Computing supremal minimum-weight controllable and normal sublanguages

Citation for published version (APA):

Document status and date:
Published: 01/01/2009

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
SE Report: Nr. 2009-03

Computing Supremal
Minimum-Weight Controllable and
Normal Sublanguages

Rong Su, Jan H. van Schuppen,
Jacobus E. Rooda and Mihály Petreczky

ISSN: 1872-1567

SE Report: Nr. 2009-03
Eindhoven, April 2009
SE Reports are available via http://se.wtb.tue.nl/sereports
Abstract

In practical applications we are frequently required to find a supervisor that can achieve certain optimal performance. Some performance such as maximum throughput or minimum execution time/cost can be specified in terms of weights. In this paper we first define a minimum-weight supervisory control problem on weighted discrete-event systems. Then we show that, the supremal minimum-weight controllable and normal sublanguages exist, and can be computed by a terminable algorithm.
1 Introduction

In many practical applications we are frequently required to develop a controller that can not only enforce safety and liveliness related specifications, but also achieve certain optimal performance. For example, in semiconductor industry cluster tools are used to process wafers. A typical requirement is that, the system should not deadlock or livelock, and should achieve as high a throughput as possible [33]. A similar requirement appears in highway traffic control. At a certain abstraction level, such a system can be modeled as a weighted automata, which carries weights on its edges. Each weight can represent execution time or cost. The problem is to synthesize a supervisor which guarantees that, the closed-loop behavior complies with specifications and is controllable and observable[2] [3], and furthermore, the supervisor can drive the system towards a desirable state with minimum time/cost in terms of an appropriate sum of weights.

In this paper, controllability, normality and supremality are adopted. We first define a weight over each nonblocking controllable and normal sublanguage as the weight of the longest string in this sublanguage. Then we present two supervisor synthesis problems as to find the supremal (i.e. largest in terms of set inclusion) nonblocking controllable sublanguage and controllable and normal sublanguage respectively that have the minimum finite weight. After that we provide for each problem a terminable algorithm to compute the relevant supremal minimum-weight sublanguage. Our contributions in this paper can be described in two aspects: first, we propose a weighted supervisory control problem under partial observation; second, we present an efficient computational algorithm to solve the problem.

Synthesis problems involving quantitative costs have been discussed in the literature, especially in optimal supervisory control, e.g. [9] [7] [8] [10] [11] [12]. These approaches are aimed to find a supervisor that can drive a deterministic plant from the initial state to a target state set with the minimum cost. Although their general goals are the same as ours, they are different from ours in various aspects besides their deterministic setup. In [9] weights are assigned to transitions, but all events are assumed to be controllable, and the supervisor need not be the least restrictive. Furthermore, partial observation is not considered. In [7] their setup is very close to ours, except that it is for deterministic systems and no partial observation is under consideration. In [8] the weights are not assigned to transitions, instead, they are considered in terms of cost of event disabling, reaching undesirable states and not being able to reach desirable states. Thus, their optimal control problem is different from ours. Besides, they adopt the state-feedback control instead of even-based control used in this paper. In [10] weights are assigned to events and states, not on transitions. The cost function is defined as the sum of event weights and event disabling costs at certain states. If event disabling costs are assumed to be zero, then their problem is the same as our first problem under full observation. But their computational procedure is completely different from ours. Furthermore, partial observation is not considered. [11] is an extension of [10], where partial observation is taken into consideration. But their approach is to first project out all unobservable events from the plant model, then solve the optimal control problem in the observable abstraction of the original plant model by using the technique proposed in [10]. The resultant supervisor is least restrictive with respect to the observable abstraction, but not with respect to the original model. This differs from our goal of achieving the least restrictive supervisor with respect to the original plant model. In [12] the weights are assigned to transitions and event disabling. But they define the weight of a (controllable) sublanguage as the signed real measure, which is roughly the sum of all string weights, instead of the maximum string weight used in our paper, making their optimal control
Synthesis problems involving quantitative costs have also been discussed in supervisory control of timed automata \([4, 5, 6, 25, 31]\). They comply with the specification and can reach a certain desirable state (e.g., a marker state) with a cost no more than some pre-specified value, even when uncontrollable events happen. Among these approaches, \([6]\) is an extension of \([4]\) and \([5]\). \([25]\) deals with a special control strategy, where a winning path must follow a special pattern of alternating controllable and uncontrollable events. \([31]\) deals with acyclic timed automata. In \([4, 5, 6, 25, 31]\) a game theoretic approach is used, where uncontrollable events play the role of the opponent, which tries to maximize the cost. The controllable events then correspond to the player who tries to minimize the cost. A solution of the control problem corresponds then to a strategy of the player corresponding to controllable events. The arising game-theoretic problem is a two-person zero-sum dynamic game. It was noted above, controllable actions as the actions of the first player, and by modeling similar the literature, see \([14, 21]\). Games on graphs and finite-state Markov-chains have been studied before, see \([23, 20, 13, 15, 16, 24, 18, 19]\). However, none of the cited results seem to be directly applicable to the problem considered in this paper. The problem in \([13, 24, 19]\) is similar to this problem of this paper, however, in \([13, 24, 19]\) only the cost of the terminal state, but not the path leading to it is considered. In \([23, 20, 22]\) stochastic finite-state games are considered. In \([18]\) finite-state dynamic games with mean payoff are considered. However, the payoff function in our setting is not mean-payoff. The class of repeated games with overtaking payoff \([21]\) seems to be the closest to our setting. However, we did not find any solution algorithm for the infinite horizon case in the literature. In \([17]\), combined parity and mean-payoff games were considered. Control problem in the The qualitative costs considered here are not mean-payoff.

Besides the difference of supervisory control problem formulations including that we deal with systems with partial observations, which are not considered in the existent approaches, pursuing supremal control strategies makes our approach different from most game-theory-based approaches. Although in \([6]\) the authors also aim at the supremal strategy and their computational algorithm is close to ours under an appropriate formulation transformation, no specific upper bound for the termination of the algorithm is given in \([6]\) because of the involvement of time, and the algorithm does not take the conformation of specifications into consideration, which, as a contrast, is dealt in our algorithm. Weighted automata also appear in supervisory control or fault diagnosis of probabilistic/stochastic discrete-event systems, e.g. \([26, 28, 32]\), speech recognition, e.g. \([30, 29]\), and verification, e.g. \([27]\). But their target problems are completely different from ours.

This report is organized as follows. In Section II we first provide all relevant necessary concepts about languages and automata, then introduce a minimum-weight supervisory control problem. After that we present a terminable algorithm in Section III, which computes the supremal minimum-weight controllable sublanguages. In Section IV we discuss how to handle partial observation by introducing the concept of supremal minimum-weight controllable and normal sublanguages, and providing an algorithm to compute them. Conclusions are drawn in Section V. All long proofs are provided in the Appendix.
2 Minimum-Weight Supervisory Control Problems

In this section we first review basic concepts of languages and weighted finite-state automata. Then we present two minimum-weight supervisory control problems that take respectively full and partial observation into consideration.

Let $\Sigma$ be a finite alphabet, and $\Sigma^*$ denote the Kleene closure of $\Sigma$, i.e. the collection of all finite sequences of events taken from $\Sigma$. Given two strings $s,t \in \Sigma^*$, $s$ is called a prefix substring of $t$, written as $s \leq t$, if there exists $s' \in \Sigma^*$ such that $ss' = t$, where $ss'$ denotes the concatenation of $s$ and $s'$. We use $e$ to denote the empty string of $\Sigma^*$ such that for any string $s \in \Sigma^*$, $es = se = s$. We use $|s|$ to denote the length of $s$, e.g. suppose $s = aab$ then $|s| = 3$. In particular, $|e| = 0$. A subset $L \subseteq \Sigma^*$ is called a language. $\overline{L} = \{s \in \Sigma^* | \exists t \in L \ s \leq t\} \subseteq \Sigma^*$ is called the prefix closure of $L$. $L$ is called prefix closed if $L = \overline{L}$. Given two languages $L, L' \subseteq \Sigma^*$, $LL' := \{ss' \in \Sigma^* | s \in L \land s' \in L'\}$. When $L$ is a singleton, say $L = \{s\}$, then we simply use $sL'$ to denote $\{s\}L'$.

A weighted finite-state automaton is a pair $(G = (X, \Sigma, \xi, x_0, X_m, f))$, where $G$ denotes a deterministic finite-state automaton with $X$ for the state set, $\Sigma$ for the alphabet, $\xi : X \times \Sigma \rightarrow X$ for the (partial) transition function, $x_0$ for the initial state, and $X_m$ for the marker state set, and $f : X \times \Sigma \rightarrow \mathbb{R}^+$ is the weight function, where $\mathbb{R}^+$ denotes the set of positive reals. We use $\xi(x,\sigma)$ to denote that, the transition $\xi(x,\sigma)$ is defined, and $\neg \xi(x,\sigma)$ if $\xi(x,\sigma)$ not being defined. As usual, we extend the domain of $\xi$ from $X \times \Sigma$ to $X \times \Sigma^*$. Let $L(G) := \{s \in \Sigma^* | \xi(x_0, s)\}$ be the closed behavior of $G$ and $L_m(G) := \{s \in L(G) | \xi(x_0, s) \in X_m\}$ for the marked behavior of $G$. Given a language $K \subseteq \Sigma^*$, let $\overline{K} := \{s \in \Sigma^* | (\exists s' \in K) s \leq s'\}$ be the prefix closure of $K$. We use $\phi(\Sigma)$ to denote the collection of all finite-state automata, whose alphabet is $\Sigma$.

To associate a weight to each sublanguage, we first define a weight for each string. To this end, let $\theta_G : X \times \Sigma^* \rightarrow \mathbb{R}^+$ be a map, where

1. $\theta_G(x, e) = 0$

2. $\forall s \sigma \in \Sigma^* \theta_G(x, s \sigma) := \begin{cases} \theta_G(x, s) + f(\xi(x, s), \sigma) & \text{if } \xi(x, s)! \text{ and } \xi(x, s \sigma)! \\ \text{undefined} & \text{otherwise} \end{cases}$

In other words, the weight of a string is simply the sum of weights of transitions appearing in this string. For each sublanguage $K \subseteq L(G)$, the weight of $K$ with respect to $G$ is defined as follows:

$$\omega_G(K) := \begin{cases} \max_{s \in K} \theta_G(x_0, s) & \text{if } K \neq \emptyset \text{ and } K \text{ is finite} \\ +\infty & \text{otherwise} \end{cases}$$

The motivation of assigning an infinite weight to the empty set can be explained as follows. Usually a set of strings is associated with a property, e.g. $K \subseteq L_m(G)$ is a collection of strings that can reach marker states. When a set is empty, it means there is no finite string satisfying that property. Thus, the weight of the empty set should not be finite.

To present our first supervisory control problem, we need the concept of controllability. Let $\Sigma = \Sigma_c \cup \Sigma_{uc}$, where disjoint subsets $\Sigma_c$ and $\Sigma_{uc}$ denote respectively the set of controllable events and the set of uncontrollable events.
Definition 2.1. [34] Given $G \in \phi(\Sigma)$, a language $K \subseteq L(G)$ is controllable with respect to $G$ if $K \Sigma_{uc} \cap L(G) \subseteq K$. □

Given another automaton $E \in \phi(\Sigma)$, let

$WC(G, E) := \{K \subseteq L_m(G) \cap L_m(E)|K \text{ is controllable w.r.t. } G \land \omega_G(K) < \infty\}$

be the collection of controllable sublanguages $K \subseteq L_m(G) \cap L_m(E)$, whose weights are finite. It is possible that, $WC(G, E) = \emptyset$. Because $\min_{(x, \sigma) \in X \times \Sigma: \xi(x, \sigma)!(x, \sigma) > 0}$ for any $K \in WC(G, E)$ we can derive that the set $\{K' \in WC(G, E)|\omega_G(K') \leq \omega_G(K)\}$ is finite. Thus, there exists $K^* \in WC(G, E)$ such that

$(\forall K \in WC(G, E)) \omega_G(K^*) \leq \omega_G(K)$

Since an arbitrary union of controllable sublanguages is still controllable, we have that

$\bigcup_{K \in WC(G, E): \omega_G(K) = \omega_G(K^*)} K \in WC(G, E)$

which is called the supernal minimum-weight controllable sublanguage of $G$ with respect to $E$, and denote it as $\sup WC(G, E)$. We now introduce one more concept before we define a supervisor synthesis problem.

Definition 2.2. $G = (X, \Sigma, \xi, x_0, X_m)$ is marking deadlock if $(\forall x \in X_m)(\forall \sigma \in \Sigma) - \xi(x, \sigma)!$. □

A marking deadlock automaton is the one, whose marker states are deadlock states. The reason that we are interested in marking deadlock automata is because we want to design a controller such that, the first time of reaching a marker state in the closed-loop system should be as early as possible. As long as a marker state is reached, whether the plant continues or stops is not of our interest. In reality, after a marker state is reached, the system can be reset to repeat the same sequence again, as commonly seen in manufacturing systems. We now state the first problem that we will solve in this paper:

Problem 2.3. Given a plant $G \in \phi(\Sigma)$, which is marking deadlock, and a specification $E \in \phi(\Sigma)$, how to compute $\sup WC(G, E)$? □

In Problem 1 we consider every event to be observable. This may not be true in reality. To deal with partial observation, we first introduce the concept of normality. Let $G \in \phi(\Sigma)$ and $\Sigma = \Sigma_o \cup \Sigma_uo$, where the disjoint subsets $\Sigma_o$ and $\Sigma_uo$ denote respectively the set of observable events and the set of unobservable events. Let $\Sigma' \subseteq \Sigma$. A mapping $P : \Sigma^* \rightarrow \Sigma'^*$ is called the natural projection with respect to $(\Sigma, \Sigma')$, if

1. $P(\epsilon) = \epsilon$

2. $(\forall \sigma \in \Sigma) P(\sigma) := \begin{cases} \sigma & \text{if } \sigma \in \Sigma' \\ \epsilon & \text{otherwise} \end{cases}$

3. $(\forall s\sigma \in \Sigma'^*) P(s\sigma) = P(s)P(\sigma)$

5 Minimum-Weight Supervisory Control Problems
Given a language \( L \subseteq \Sigma^* \), \( P(L) := \{ P(s) \in \Sigma^* \mid s \in L \} \). The inverse image mapping of \( P \) is 
\[
P^{-1} : 2^{\Sigma^*} \rightarrow 2^{\Sigma^*} : L \mapsto P^{-1}(L) := \{ s \in \Sigma^* \mid P(s) \in L \}
\]
Let \( P_o : \Sigma^* \rightarrow \Sigma_o^* \) be the natural projection. We have the following definition.

**Definition 2.4.** Let \( K \subseteq L(G) \). We say \( K \) is normal with respect to \( G \) and \( P_o \) if 
\[
K = L(G) \cap P_o^{-1}(P_o(K)).
\]
This definition is different from the one defined in [34], where normality is a property on the language itself, not on its prefix closure as used in Def. 2.4. We hope this slight abuse of notation will not cause any confusion for readers. Given another automaton \( E \in \phi(\Sigma) \), let 
\[
WCN(G, E) := \{ K \in WC(G, E) \mid K \text{ is normal with respect to } G \text{ and } P_o \}
\]
For notational simplicity, from now on we say a sublanguage \( K \subseteq L(G) \) is controllable and normal with respect to \( G \), if \( K \) is controllable with respect to \( G \) and \( \Sigma uc \), and normal with respect to \( G \) and \( P_o \). Thus, \( WCN(G, E) \) is the collection of all controllable and normal sublanguages of \( L_m(G) \cap L_m(E) \) with respect to \( G \), whose weights are finite. Since an arbitrary union of controllable and normal sublanguages is still controllable and normal, by using a similar argument as showing the existence of supremal minimum-weight controllable sublanguage \( WC(G, E) \), we can derive, there exists a unique element \( K^* \in WCN(G, E) \) such that,
\[
(\forall K \in WCN(G, E)) \omega_G(K^*) \leq \omega_G(K) \land [\omega_G(K) = \omega(K^*) \Rightarrow K \subseteq K^*]
\]
We call \( K^* \) the supremal minimum-weight controllable and normal sublanguage of \( G \) with respect to \( E \), and denote it as \( supWCN(G, E) \). Our second problem is specified as follows:

**Problem 2.5.** Given a plant \( G \in \phi(\Sigma) \), which is marking deadlock, and a specification \( E \in \phi(\Sigma) \), how to compute \( supWCN(G, E) \)?

Next, we provide terminable procedures to compute \( supWCN(G, E) \) and \( supWCN(G, E) \), and show their correctness.

### 3 Computing Supremal Minimum-Weight Controllable Sublanguages

The basic idea is that, we first compute the supremal controllable sublanguage of \( G \) under \( E \), then we search for the supremal minimum-weight controllable sublanguage (SMWCS) of \( G \) from the previously computed supremal controllable sublanguage. By doing this we can make sure that, the computed controllable sublanguage complies with the specification \( L_m(E) \). Searching for the SMWCS can be done in a way similar to searching for the optimal winning strategy in timed game theory [6], or the Dijkstra’s algorithm for solving the single-source shortest-paths problem [1], except that the Dijkstra’s algorithm does not distinguish controllable and uncontrollable edges, and updates the cost of a state based on costs of its ancestor states instead of costs of its descendent states. Since the algorithm may encounter infinitely large weights, we treat \( +\infty \) as a number and make
the following rule: for any $a \in \mathbb{R}^+$, $(+\infty) + a = +\infty$.

**Procedure for Supremal Minimum-Weight Controllable Sublanguages (PSMWCS):**

1. Input: a marking deadlock $G = (X, \Sigma, \xi, x_0, X_m, f)$ and a specification $E \in \phi(\Sigma)$
2. Initialization:
   (a) Compute the supremal controllable sublanguage $K \subseteq L_m(G) \cap L_m(E)$ of $G$.
   (b) If $K = \emptyset$ then $K := \emptyset$ and go to Step (5).
   (c) Construct a weighted automaton $(S = (Y, \Sigma, \eta, y_0, Y_m), f')$ such that $L_m(S) = K$, $L(S) = L_m(S)$ and
      $$(\forall s, s' \in \Sigma^*) \eta(y_0, s) = \eta(y_0, s) \Rightarrow \xi(x_0, s) = \xi(x_0, s')$$
      and the weight function $f' : Y \times \Sigma \to \mathbb{R}^+$ is defined as follows:
      $$(\forall s \in L(S))(\forall \sigma \in \Sigma) f'(\eta(y_0, s), \sigma) := f(\xi(x_0, s), \sigma)$$
   (d) For each $y \in Y_m$, set $\kappa_0(y) = 0$
   (e) For each $y \in Y - Y_m$, define $\kappa_0(y) := +\infty$
3. Iterate on $k = 1, 2, \cdots$, as follows:
   (a) For each $y \in Y_m$, $\kappa_k(y) := 0$
   (b) For each $y \in Y - Y_m$ we have
      $$\kappa_k(y) := \begin{cases} \max_{\sigma \in \Sigma} \kappa_{k-1}(\eta(y, \sigma)!) (f'(y, \sigma) + \kappa_{k-1}(\eta(y, \sigma)!)) & \text{if } (\exists \sigma \in \Sigma) \eta(y, \sigma)! \\ \min_{\sigma \in \Sigma} \kappa_{k-1}(\eta(y, \sigma)!) (f'(y, \sigma) + \kappa_{k-1}(\eta(y, \sigma)!)) & \text{if } \emptyset \neq \{ \sigma \in \Sigma \mid \eta(y, \sigma)! \} \subseteq \Sigma_c \\ \kappa_{k-1}(y) & \text{otherwise} \end{cases}$$
   (c) Termination when: $(\exists r \in \mathbb{N}) (\forall y \in Y) \kappa_{r-1}(y) = \kappa_r(y)$
4. If $\kappa_r(y_0) = +\infty$, then $K := \emptyset$ and go to step (5). Otherwise, let $S' = (Y', \Sigma, \eta', y_0, Y'_m)$ where
   (a) $Y' := \{ y \in Y | \kappa_r(y) < +\infty \}$
   (b) $Y'_m := Y' \cap Y_m$
   (c) $\eta' : Y' \times \Sigma \to Y'_m$, where for any $(y, \sigma) \in Y' \times \Sigma$,
      $$\eta'(y, \sigma) := \begin{cases} \eta(y, \sigma) & \text{if } \eta(y, \sigma) \in Y' \text{ and } f'(y, \sigma) + \kappa_r(\eta(y, \sigma)) \leq \kappa_r(y) \\ \text{not defined} & \text{otherwise} \end{cases}$$
      and let $K := L_m(S')$.
5. Output: $K$

The definition of $S$ guarantees that the weight function $f'$ is well defined. As long as $K \neq \emptyset$, we can always find such an automaton $S$. For example, we can first find an arbitrary recognizer of $K$, say $\hat{S}$ with $L(\hat{S}) = L_m(\hat{S})$. Then let $S := G \times \hat{S}$, which, we can check, satisfies the definition. What PSMWCS does is that, first the supremal controllable sublanguage $K \subseteq L_m(G) \cap L_m(E)$ with respect to $G$ is computed. If $K = \emptyset$ then we know that WC(G, E) = \emptyset. When $K \neq \emptyset$, we start to search the largest sublanguage of $K$, which is controllable with respect to $G$ with the minimum finite weight. The search is based on a recognizer $S$ of $K$. At each stage $k$, the weight $\kappa_k(y)$ at each state $y$ of $S$.
actually is equal to the minimum worst-case accumulated weight from \( y \) to a marker state reachable from \( y \) within no more than \( k \) transitions. The updating rule says that: at each stage \( k \), the weight of each marker state is always zero; the weight of a non-marker state is updated based on weights of its descendants in the previous stage \( k-1 \). When the termination condition holds at \( k \), we first check whether \( \kappa_k(y_0) \) is finite. If \( \kappa_k(y_0) = +\infty \), then it means it is not possible to drive the plant from the initial state \( y_0 \) to a marker state within a finite number of transitions by simply disabling controllable events. If \( \kappa_k(y_0) < +\infty \), then we construct \( S' \), whose transition maps indicates that, only transitions that are part of paths, whose weights are no more than the minimum worst-case weight of the initial state are allowed. We will show that, \( L_m(S') = \sup_{W} WC(G, E) \). To this end, we first show that, the procedure PSMWCS terminates no later than \( k = |Y| \), where \( |Y| \) denotes the size of \( Y \). To this end, we need to introduce a few more concepts and lemmas.

**Definition 3.1.** Suppose \( S \) is computed in PSMWCS. For any two strings \( s, s' \in \Sigma^* \) and \( y \in Y \), we say \( s' \) is an uncontrollable bypath of \( s \) with respect to \( y \), denoted as \( s \succ_y s' \), if \( \eta(y, s) \in Y_m \) and \( \eta(y, s') \in Y_m \) and the following condition holds,

\[
(\exists s_1, s_2, s_3 \in \Sigma^*)(\exists \sigma \in \Sigma_{uc}) s = s_1s_2 \land s' = s_1\sigma s_3
\]

\[\square\]

What Def. 3.1 says is that, an uncontrollable bypath \( s' \) of \( s' \) with respect to \( y \) shares a substring \( s_1 \) with \( s \), and departs from \( s \) by an uncontrollable event \( \sigma \), and both strings reach a marker state from state \( y \). Next, we introduce the concepts of chains and maximal chains.

**Definition 3.2.** Suppose \( S \) is computed in PSMWCS. For each \( y \in Y \) and \( s \in \Sigma^* \), a set \( c(y, s) \subseteq \{ s' \in \Sigma^* | s \succ_y s' \} \cup \{ s \} \) with \( s \in c(y, s) \) is a chain of \( s \) with respect to \( y \) if

\[
(\forall t \in c(y, s)) (|\{ \sigma \in \Sigma \mid \sigma \in c(y, s) \} \cap \Sigma_{uc}| \leq 1)
\]

We say \( c(y, s) \) is maximal if

\[
(\forall t \in c(y, s)) (\forall \sigma \in \Sigma_{uc}) \eta(y, t\sigma)! \Rightarrow t\sigma \in c(y, s)
\]

\[\square\]

From Def. 3.2 we get that, any prefix substring of a string in a chain can have at most one controllable extension within the chain. If the chain is maximal, then any uncontrollable extension of a prefix substring that is allowed in \( S \) should still be a prefix substring of that chain. This property actually suggests that, each maximal chain is controllable with respect to \( S \) (thus, with respect to \( G \) as well because \( S \) is controllable with respect to \( G \)). Let \( \gamma(y, s) \) be the collection of all maximal chains of \( s \) with respect to \( y \). For each \( k \in \mathbb{N} \) we define

\[
\varrho(y, k) := \{ C \in \gamma(y, s)| \eta(y, s) \in Y_m \land \sup_{s' \in C} |s'| \leq k \}
\]

Thus, \( \varrho(y, k) \) is the collection of all maximal chains whose length is no more than \( k \). Let

\[
v_k(y) = \begin{cases} 
\min_{C \in \varrho(y, k)} \max_{s' \in C} \theta_S(y, s') & \text{if } \varrho(y, k) \neq \emptyset \\
+\infty & \text{otherwise}
\end{cases}
\]

Informally speaking, \( v_k(y) \) denotes the minimum worst-case weight from \( y \) to a marker state within \( k \) transitions, where 'worst-case' means that uncontrollable transitions may pull the transition away from the path with the absolute minimum weight. We have the
following lemmas.

**Lemma 3.3.** Suppose \( S \) is computed by PSMWCS. For any \( y \in Y \) and \( k \in \mathbb{N} \), we have \( \kappa_k(y) = \upsilon_k(y) \).

The proof of Lemma 3.3 is provided in the Appendix. It relates the weight of a state in PSMWCS to the weight of a maximal chain, and the number of iterations \( k \) to the length of the longest string in that maximal chain.

**Lemma 3.4.** Suppose \( S \) is computed by PSMWCS. For any \( y \in Y \) and \( C \in \gamma(y, |Y|) \), there exists \( \hat{C} \in \gamma(y, |Y| - 1) \) such that \( \max_{u \in \hat{C}} \theta_S(y, \hat{u}) \leq \max_{u \in C} \theta_S(y, u) \).

The proof of Lemma 3.4 is presented in the Appendix. We now use Lemma 3.3 and Lemma 3.4 to show the following theorem.

**Theorem 3.5.** In PSMWCS, for any \( y \in Y \) we have \( \kappa_{|Y| - 1}(y) = \kappa_{|Y|}(y) \).

**Proof:** By Lemma 3.3 we have \( \kappa_{|Y|}(y) = \upsilon_{|Y|}(y) \). Since \( g(y, |Y| - 1) \subseteq g(y, |Y|) \) and by Lemma 3.4 we have

\[
v_{|Y|}(y) = \begin{cases} 
\min_{\hat{C} \in \gamma(y, |Y| - 1)} \max_{u \in \hat{C}} \theta_S(y, \hat{u}) & \text{if } g(y, |Y| - 1) \neq \emptyset \\
+\infty & \text{otherwise}
\end{cases}
\]

\[
v_{|Y| - 1}(y) = \begin{cases} 
\min_{\hat{C} \in \gamma(y, |Y| - 1)} \max_{u \in \hat{C}} \theta_S(y, \hat{u}) & \text{if } g(y, |Y| - 1) \neq \emptyset \\
+\infty & \text{otherwise}
\end{cases}
\]

By Lemma 3.3, we have \( \kappa_{|Y| - 1}(y) = \kappa_{|Y|}(y) \), and the theorem follows.

**Theorem 3.5** indicates that, the termination condition in PSMWCS will be satisfied no later than \( k = |Y| \). Next, we need to show that, the output of PSMWCS is indeed what we want. To this end we first need the following lemma.

**Lemma 3.6.** For any \( s \in L_m(S) \), every maximal chain \( C \in \gamma(y_0, s) \) is controllable with respect to \( G \); and every nonempty controllable sublanguage \( K \subseteq L_m(S) \) with respect to \( G \) contains one maximal chain \( C \in \gamma(y_0, s) \) for some \( s \in K \).

The proof of Lemma 3.6 is provided in the Appendix. Lemma 3.6 is used in the proof of the following theorem.

**Theorem 3.7.** Given a weighted finite-state automaton \((G, f)\) with a marking deadlock \( G \in \phi(\Sigma) \) and a specification \( E \in \phi(\Sigma) \), let \( \hat{K} \) be computed by PSMWCS. Then (1) \( \hat{K} = \emptyset \) if and only if \( WC(G, E) = \emptyset \); (2) When \( \hat{K} \neq \emptyset \), we have \( \hat{K} = \sup WC(G, E) \).
The proof of Theorem 3.7 is presented in the Appendix. Next, we use a simple example to illustrate the procedure. Suppose the plant model $G$ is depicted in Figure 1, where $\Sigma = \{a, b, c, d\}$ and $\Sigma_c = \{a, b\}$. The initial state is $x_0$ and the marker states are $x_3$ and $x_4$. The cost function $f$ is as follows:

$f(x_0, a) = 1, f(x_1, a) = 2, f(x_1, b) = 2, f(x_1, d) = 6, f(x_2, a) = 2, f(x_2, c) = 1, f(x_2, d) = 1$

For convenience we directly show the cost values on relevant edge in Figure 1. For example, the edge $a/1$ denotes that, the event is $a$ and the corresponding cost value of this transition is 1. The specification $E$ is depicted in Figure 2. We can easily check that, $L_m(G) \cap L_m(E)$ is controllable with respect to $G$. Let $S$ be an automaton depicted in Figure 3, which recognizes $L_m(G) \cap L_m(E)$. The cost function $f'$ for $S$ is defined as...
follows:
\[ f'(y_0, a) = 1, f'(y_1, b) = 2, f'(y_1, d) = 6, f'(y_2, c) = 1, f'(y_2, d) = 1 \]

We now apply PSMWCS on \( S \), and the computational results are listed as follows.

1. \( k = 0 \): \( \kappa_0(y_3) = \kappa_0(y_4) = 0, \kappa_0(y_0) = +\infty, \kappa_0(y_1) = +\infty, \kappa_0(y_2) = +\infty \)
2. \( k = 1 \): \( \kappa_1(y_3) = \kappa_1(y_4) = 0, \kappa_1(y_0) = +\infty, \kappa_1(y_1) = 6, \kappa_1(y_2) = +\infty \)
3. \( k = 2 \): \( \kappa_2(y_3) = \kappa_2(y_4) = 0, \kappa_2(y_0) = 7, \kappa_2(y_1) = 6, \kappa_2(y_2) = +\infty \)
4. \( k = 3 \): \( \kappa_3(y_3) = \kappa_3(y_4) = 0, \kappa_3(y_0) = 7, \kappa_3(y_1) = 6, \kappa_3(y_2) = 8 \)
5. \( k = 4 \): \( \kappa_4(y_3) = \kappa_4(y_4) = 0, \kappa_4(y_0) = 7, \kappa_4(y_1) = 6, \kappa_4(y_2) = 8 \)

Since at \( k = 4 \), for any \( y \in Y \) we have \( \kappa_3(y) = \kappa_4(y) \). The termination condition holds. Since \( \kappa_4(y_0) = 7 < +\infty \), we construct \( S' \) as follows:

1. \( Y' = \{ y_0, y_1, y_2, y_3, y_4 \} \)
2. \( Y'_m = Y_m \cap Y' = \{ y_3, y_4 \} \)
3. The transition \( \eta' \) is almost the same as \( \eta \), except that \( \eta'(y_1, b) \) is not defined because, although \( \eta(y_1, b) \in Y' \), the condition \( \kappa_4(y_1) \geq f'(y_1, b) + \kappa_4(\eta(y_1, b)) \) does not hold.

The final \( S' \) is depicted in Figure 4, where states \( y_2 \) and \( y_3 \) are unreachable from \( y_0 \). Thus, \( \hat{K} = L_m(S') = \{ ad \} \). We can easily verify that, the supremal minimum-weight controllable sublanguage \( \sup WC(G, E) = \{ ad \} \). Thus, \( \hat{K} = \sup WC(G, E) \), as predicted by Theorem 3.7.

Next, we describe how to solve Problem 2, namely to compute the supremal minimum-weight controllable and normal sublanguage \( \sup WCN(G, E) \).

11 Computing Supremal Minimum-Weight Controllable Sublanguages
4 Computing Supremal Minimum-Weight Controllable and Normal Sublanguages

We first present a terminable algorithm below and provide an intuitive explanation for it. Then we show that the procedure fulfils our expectation for solving Problem 2.

Procedure for Supremal Minimum-Weight Controllable and Normal Sublanguages (PSMWCNS):

1. Input: a marking deadlock $G = (X, \Sigma, \xi, x_0, X_m, f)$ and a specification $E \in \phi(\Sigma)$

2. Initialization:
   
   (a) Compute the supremal controllable and normal sublanguage $K \subseteq L_m(G) \cap L_m(E)$ with respect to $G$.
   
   (b) If $K = \emptyset$ then set $K_{CN} = \emptyset$ and go to step (6).

   (c) Construct a weighted automaton $(S = (Y, \Sigma, \eta, y_0, Y_m), f')$ such that $L_m(S) = K$, $L(S) = L_m(S)$ and
   
   $$(\forall s, s' \in \Sigma^*) \eta(y_0, s) = \eta(y_0, s') \Rightarrow \xi(x_0, s) = \xi(x_0, s')$$

   and the weight function $f': Y \times \Sigma \rightarrow \mathbb{R}^+$ is defined as follows:

   $$f'(Y, \Sigma, \eta(y_0, s), \sigma) := f(\xi(x_0, s), \sigma)$$

   (d) For each $y \in Y_m$, set $\kappa_0(y) = 0$

   (e) For each $y \in Y - Y_m$, define $\kappa_0(y) := +\infty$

3. Iterate on $k = 1, 2, \cdots$, as follows:

   (a) For each $y \in Y_m$, $\kappa_k(y) := 0$

   (b) For each $y \in Y - Y_m$ we have

   $$\kappa_k(y) := \begin{cases} 
   \max_{\sigma \in \Sigma_{uc}, \eta(y, \sigma)} f'(y, \sigma) + \kappa_{k-1}(\eta(y, \sigma)) & \text{if } (\exists \sigma \in \Sigma_{uc}) \eta(y, \sigma)!
   
   \min_{\sigma \in \Sigma_{uc}, \eta(y, \sigma)} f'(y, \sigma) + \kappa_{k-1}(\eta(y, \sigma)) & \text{if } \emptyset \neq \{\sigma \in \Sigma \mid \eta(y, \sigma)\} \subseteq \Sigma_{uc}
   
   \kappa_{k-1}(y) & \text{otherwise}
   \end{cases}$$

   (c) Termination when: $(\exists r \in \mathbb{N})(\forall y \in Y) \kappa_{r-1}(y) = \kappa_r(y)$

4. If $\kappa_r(y_0) = +\infty$, $K_{CN} := \emptyset$ and go to Step (6). Otherwise, let $S' = (Y', \Sigma, \eta', y_0, Y_m')$ where

   (a) $Y' := \{y \in Y \mid \kappa_r(y) < +\infty\}$

   (b) $Y_m' := Y' \cap Y_m$

   (c) $\eta' : Y' \times \Sigma \rightarrow Y'$, where for any $(y, \sigma) \in Y' \times \Sigma$,

   $$\eta'(y, \sigma) := \begin{cases} 
   \eta(y, \sigma) & \text{if } \eta(y, \sigma) \in Y'
   
   \text{not defined} & \text{otherwise}
   \end{cases}$$

   compute the largest controllable and normal sublanguage $K \subseteq L_m(S')$ with respect to $G$. If $K = \emptyset$ then $K_{CN} := \emptyset$ and go to step (6). Otherwise, continue.

5. Set $K_0 := K$ and iterates on $r = 0, 1, \cdots$, as follows:

   (a) Search a set $\psi(K_r) := \{s \in K_r \mid \theta_S(y_0, s) = \omega_S(K_r)\}$. 
(b) Compute the largest controllable and normal sublanguage \( \hat{K}_{r+1} \subseteq \hat{K}_r - \psi(\hat{K}_r) \) with respect to \( G \).

(c) If \( \hat{K}_{r+1} = \emptyset \) then set \( K_{CN} := \hat{K}_r \) and go to Step (6). Otherwise, continue on \( r + 1 \).

6. Output: \( K_{CN} \)

What PSMWCNS does is that, we first compute the largest language \( L_m(S') \subseteq L_m(G) \cap L_m(E) \), which contains every string allowing a finite reach from the initial state \( x_0 \) to a marker state \( x \in X_m \), even when uncontrollable events happen. From the previous section we can derive that, \( L_m(S') = \cup_{s \in L_m(S') \cap \{y_0, s\}} \), and furthermore, \( L_m(S') \) is controllable. But it may not be normal. Thus, at step (4) in PSMWCNS we check whether the largest controllable and normal sublanguage \( \hat{K} \) of \( L_m(S') \) is nonempty. If \( \hat{K} = \emptyset \), then we know that, supWCN\((G, E) = \emptyset \). Otherwise, we continue to search at step (5) for the largest controllable and normal sublanguage of \( L_m(S') \) with respect to \( G \) that has the minimum weight. What step (5) does is that, we first remove all strings with the maximum weight from \( \hat{K}_r \), where the set of all strings with the maximum weight \( \psi(\hat{K}_r) \) is computable by the Dijkstra algorithm, except that, instead of searching for the minimum-weight paths in the Dijkstra algorithm, we search for the maximum-weight paths. Then we compute the largest controllable and normal sublanguage \( \hat{K}_{r+1} \subseteq \hat{K}_r - \psi(\hat{K}_r) \) with respect to \( G \).

Theorem 4.1. Given a weighted finite-state automaton \((G, f)\) with a marking deadlock \( G \in \phi(\Sigma) \) and a specification \( E \in \phi(\Sigma) \), let \( K_{CN} \) be computed by PSMWCNS. Then (1) \( K_{CN} = \emptyset \) if and only if WCN\((G, E) = \emptyset \); (2) When \( K_{CN} \neq \emptyset \), we have \( K_{CN} = \supWCN(G, E) \).

The proof of Theorem 4.1 is provided in the Appendix. To illustrate the procedure PSMWCNS, we consider the following example depicted in Figure 5, where \( \Sigma = \{a, b, c, d, e\} \), \( \Sigma_c = \{a, b, e\} \) and \( \Sigma_o = \{a, c, e\} \). The cost function \( f \) is as follows:

\[
\begin{align*}
  f(x_0, a) & = 1, f(x_0, c) = 10, f(x_1, a) = 2, f(x_1, b) = 2, \\
  f(x_1, d) & = 6, f(x_2, a) = 2, f(x_2, c) = 1, f(x_2, d) = 1
\end{align*}
\]

The specification \( E \) is depicted in Figure 6. We first compute the largest controllable and normal sublanguage \( K \subseteq L_m(G) \cap L_m(E) \) with respect to \( G \), which is depicted in Figure 7. The cost function \( f' \) for \( S \) is defined as follows:

\[
\begin{align*}
  f'(y_0, a) & = 1, f'(y_0, c) = 10, f'(y_1, b) = 2, f'(y_1, d) = 6, f'(y_2, c) = 1, f'(y_2, d) = 1, f'(y_5, e) = 10
\end{align*}
\]

We now apply PSMWCNS on \( S \). Since there is no loop in \( S \), every state in \( S \) has a finite weight. Thus, \( S' \) is the same as \( S \), depicted in Figure 8. Since \( a \) is controllable and observable, we can check that \( L_m(S') \) is controllable and normal with respect to \( G \). Thus, \( \hat{K} = L_m(S') \). By using an algorithm similar to the Dijkstra’s algorithm for solving the single-source shortest-paths problem [1], we can find all strings in \( L_m(S') \) with the maximum weight, which is \( s = abde \) and the weight is 14. Thus, \( \psi(\hat{K}) = \{abde\} \), and \( K' = \hat{K} - \psi(\hat{K}) \) is depicted in Figure 9. We now compute the largest controllable and
normal sublanguage \(K^{''} \subseteq K'\) with respect to \(G\). It turns out that, \(K^{''} = \{e\}\). The reason is as follows. Since \(ab \in K',\ abd \notin K',\ abd \in L(G)\) and \(d \in \Sigma_{uc}\), we get that \(ab \notin K^{''}\). Otherwise, \(K^{''}\) is not controllable with respect to \(G\). Since \(ab \notin K^{''}\) and \(P_o(ab) = P_o(ad)\), we have \(ad \notin K^{''}\). Otherwise, \(K^{''}\) is not normal with respect to \(G\) and \(P_o\). Since \(ad \notin K^{''}\) and \(d \in \Sigma_{uc}\), we get that \(a \notin K^{''}\). Otherwise, \(K^{''}\) is not controllable with respect to \(G\). We set \(K = K^{''}\) and repeat the previous computation. Clearly, after taking out \(e\), we get \(K' = \emptyset\), which means \(K^{''} = \emptyset\). Thus, the final output is \(\hat{K} = \{e\}\), and the corresponding weight is 10.

5 Conclusions

In this paper we first define the concepts of supremal minimum-weight controllable sublanguages and supremal minimum-weight controllable and normal sublanguages, and present two minimum-weight supervisory control problems, where full and partial observation are considered respectively. Then we provide an algorithm PSMWCS to compute the supremal minimum-weight controllable sublanguages. We have shown that, the algorithm terminates within a number of steps no more than the number of states of the plant model. After that, we present a terminable algorithm PSMWCNS to compute the supremal minimum-weight controllable and normal sublanguages. The supervisory control problems are formulated in a centralized manner, namely we have one plant and one specification. In reality, we may encounter high computational complexity during centralized synthesis. Thus, it is of our primary interest to investigate whether a similar approach can be applied to a hierarchical and distributed system, which will be addressed in our future papers.
Figure 7: Example 2: Automaton Model S

Figure 8: Example 2: Automaton Model S'

Acknowledgement:

We would like to thank Dr. Albert T. Hofkamp of the Systems Engineering Group at Eindhoven University of Technology for coding all algorithms mentioned in this paper.

Appendix

1. Proof of Lemma 3.3: We use induction. When $k = 0$, for any $y \in Y_m$ we have $\theta(y, 0) = \{\{\epsilon\}\} \neq \emptyset$; and for any $y \in Y - Y_m$ we have $\theta(y, 0) = \emptyset$. When $\theta(y, 0) \neq \emptyset$, by the definition we have $\theta_S(y, \epsilon) = 0$. Thus, the claim holds for $k = 0$. Suppose it holds for $k \geq 0$. We need to show that it holds for $k + 1$. To this end, we consider three cases.

Case 1: suppose there exists $\sigma \in \Sigma_{uc}$ such that $\eta(y, \sigma)!$. Then

$$\kappa_{k+1}(y) = \max_{\sigma \in \Sigma_{uc} \land \eta(y, \sigma)!} (f'(y, \sigma) + \kappa_k(\eta(y, \sigma)))$$
Let $y'_r = \eta(y, \sigma)$. By the induction hypothesis we have

$$\kappa_k(y'_r) = v_k(y'_r) = \begin{cases} \min_{v \in \mathcal{E}(y'_r, k)} \max_{s' \in W} \theta_S(y'_r, s') & \text{if } g(y'_r, k) \neq \emptyset \\ +\infty & \text{otherwise} \end{cases}$$

If there exists $\sigma \in \Sigma_{uc}$ such that $g(y'_r, k) = \emptyset$ then, since $S$ is nonblocking, we get that, for any $C' \in \cup_{r \in \mathcal{N}} g(y'_r, r)$, there exists $s' \in C'$ such that $|s'| > k$. We now show that, $g(y, k + 1) = \emptyset$. Suppose it is not true. Then let $C \in g(y, k + 1)$. Since $\sigma \in \Sigma_{uc}$, by the definition of $\succ_y$ we get that, there exists $C' \in \cup_{r \in \mathcal{N}} g(y'_r, r)$ such that $\sigma C' \subseteq C$. Thus, there exists $\sigma' \in C$ such that $|\sigma'| = |s'| + 1 > k + 1$, which contradicts the assumption that $C \in g(y, k + 1)$. Thus, $g(y, k + 1) = \emptyset$, which means $v_{k+1}(y) = +\infty$. On the other hand, since $\kappa_k(y'_r) = +\infty$ we get that $\kappa_k(y'_r + 1) = +\infty$. Thus, $\kappa_k(y_r + 1) = \kappa_k(y'_r + 1)$.

If for any $\sigma \in \Sigma_{uc}$ with $\eta(y, \sigma) \neq \emptyset$, we have $g(\eta(y, \sigma), k) \neq \emptyset$, then for any $s = \sigma' t$ with $\sigma' \in \Sigma_{uc}$, and any $C \in \sigma y (s) \cap g(y, k + 1)$, there exists $s' = \sigma'' t'$ with $\sigma'' \in \Sigma_{uc}$ such that there exists $C'' \in \sigma (y, s') \cap g(y, k + 1)$ with $C'' \subseteq C$. Thus,

$$v_{k+1}(y) = \min_{C \in \sigma (y, s'')} \max_{s' \in C} \theta_S(y, s') = \min_{C \in \sigma (y, s') \cap g(y, k + 1) \cap \sigma (y, s'')} \max_{s' \in C} \theta_S(y, s') \quad (1)$$

Thus, $v_{k+1}(y)$ can be determined only based on strings starting with an uncontrollable event. For any $C \in \sigma (y, s') \cap g(y, k + 1)$ with $s' \in \Sigma_{uc} \Sigma^*$, by the definition of $\succ_y$ we can derive that, for each $\sigma \in \Sigma_{uc}$ there exists $C''(\sigma) \in g(\eta(y, \sigma), k)$ such that $C = \cup_{\sigma \in \Sigma_{uc}} \sigma C''(\sigma)$. For notation simplicity, let $d(C''(\sigma)) := \max_{s' \in C''(\sigma)} \theta_S(\eta(y, \sigma), s')$. By Equation (1) We have

$$v_{k+1}(y) = \min_{C''(\sigma) \in g(\eta(y, \sigma), k) \cap \sigma \in \Sigma_{uc}} \max_{\sigma \in \Sigma_{uc} \cup \eta(y, \sigma)!} (f'(y, \sigma) + d(C''(\sigma))) \quad (2)$$

We will show that, from Equation (2) we can derive

$$v_{k+1}(y) = \max_{\sigma \in \Sigma_{uc} \cup \eta(y, \sigma)!} (f'(y, \sigma) + \min_{C''(\sigma) \in g(\eta(y, \sigma), k)} d(C''(\sigma))) \quad (3)$$

To this end, suppose

$$f'(y, \sigma^*) + d(C''(\sigma^*)) = \min_{C''(\sigma) \in g(\eta(y, \sigma), k)} \max_{\sigma \in \Sigma_{uc} \cup \eta(y, \sigma)!} (f'(y, \sigma) + d(C''(\sigma)))$$

Then we have that, for any $g(\eta(y, \sigma), k) \cap \sigma \in \Sigma_{uc}$,

$$f'(y, \sigma^*) + d(C''(\sigma^*)) \leq \max_{\sigma \in \Sigma_{uc} \cup \eta(y, \sigma)!} (f'(y, \sigma) + d(C''(\sigma)))$$

Figure 9: Example 2: Automaton that Recognizes $K'$
In particular, we choose $C'(\sigma)$ as arg\(\min_{C'(\sigma)\in\mathcal{E}(\eta(y,\sigma),k)} d(C'(\sigma))\). Thus, we have
\[
\min_{C'(\sigma)\in\mathcal{E}(\eta(y,\sigma),k)} \max_{\sigma\in\Sigma_{uc}} (f'(y,\sigma) + d(C'(\sigma))) \\
\leq \\
\max_{\sigma\in\Sigma_{uc} \cap \eta(y,\sigma)} (f'(y,\sigma) + \min_{C'(\sigma)\in\mathcal{E}(\eta(y,\sigma),k)} d(C'(\sigma)))
\]
To show the opposite direction of inequality, let
\[
f'(y,\sigma^*) + d(C^*(\sigma^*)) = \max_{\sigma \in \Sigma_{uc} \cap \eta(y,\sigma)} (f'(y,\sigma) + \min_{C'(\sigma)\in\mathcal{E}(\eta(y,\sigma),k)} d(C'(\sigma)))
\]
Thus, we have that, for any $C^*(\sigma) \in \mathcal{E}(\eta(y,\sigma),k)$,
\[
f'(y,\sigma^*) + d(C^*(\sigma^*)) \leq \max_{\sigma \in \Sigma_{uc} \cap \eta(y,\sigma)} (f'(y,\sigma) + d(C^*(\sigma)))
\]
In particular, we choose \(\{C'(\sigma) \in \mathcal{E}(\eta(y,\sigma),k) | \sigma \in \Sigma_{uc}\}\) as
\[
\arg\min_{C'(\sigma)\in\mathcal{E}(\eta(y,\sigma),k)} \max_{\sigma \in \Sigma_{uc} \cap \eta(y,\sigma)} (f'(y,\sigma) + d(C'(\sigma)))
\]
Thus, we have
\[
\min_{C'(\sigma)\in\mathcal{E}(\eta(y,\sigma),k)} \max_{\sigma \in \Sigma_{uc} \cap \eta(y,\sigma)} (f'(y,\sigma) + d(C'(\sigma))) \\
\geq \\
\max_{\sigma \in \Sigma_{uc} \cap \eta(y,\sigma)} (f'(y,\sigma) + \min_{C'(\sigma)\in\mathcal{E}(\eta(y,\sigma),k)} d(C'(\sigma)))
\]
which means
\[
\min_{C'(\sigma)\in\mathcal{E}(\eta(y,\sigma),k)} \max_{\sigma \in \Sigma_{uc} \cap \eta(y,\sigma)} (f'(y,\sigma) + d(C'(\sigma))) \\
= \\
\max_{\sigma \in \Sigma_{uc} \cap \eta(y,\sigma)} (f'(y,\sigma) + \min_{C'(\sigma)\in\mathcal{E}(\eta(y,\sigma),k)} d(C'(\sigma)))
\]
Thus, Equation (3) is true, from which we can derive that,
\[
v_{k+1}(y) = \max_{\sigma \in \Sigma_{uc} \cap \eta(y,\sigma)} (f'(y,\sigma) + \kappa_k(\eta(y,\sigma)))
\]
By using the hypothesis of induction we get
\[
v_{k+1}(y) = \max_{\sigma \in \Sigma_{uc} \cap \eta(y,\sigma)} (f'(y,\sigma) + \kappa_k(\eta(y,\sigma))) = \kappa_{k+1}(y)
\]
Thus, the lemma holds for Case 1.

Case 2: suppose there are only $\sigma \in \Sigma_c$ such that $\eta(y,\sigma)!$. By the definition we have
\[
\kappa_{k+1}(y) = \min_{\sigma \in \Sigma_c \cap \eta(y,\sigma!)} (f'(y,\sigma) + \kappa_k(\eta(y,\sigma!)))
\]
By the induction hypothesis we have
\[
\kappa_{k+1}(y) = \min_{\sigma \in \Sigma_c \cap \eta(y,\sigma!)} (f'(y,\sigma) + v_k(\eta(y,\sigma!)))
\]
By the definition of $v_{k+1}(y)$ and we have
\[
v_{k+1}(y) = \begin{cases} 
\min_{C \in \mathcal{E}(\eta(y,k+1))} \max_{s \in C} \theta_S(y,s) & \text{if } g(y,k+1) \neq \emptyset \\
+\infty & \text{otherwise}
\end{cases}
\]
For any $C \in \mathcal{E}(\eta(y,k+1))$ there exists $\sigma'! \in \Sigma^*$ with $\sigma' \in \Sigma_c$ such that $C \in \{\sigma' C' / C' \in \gamma(\eta(y,\sigma'),s')\}$. Thus, $\max_{C' \in C} \theta_S(y,s) = f'(y,\sigma') + \max_{\sigma \in \Sigma_c} \theta_S(\eta(y,\sigma'),s')$. Furthermore, $g(y,k+1) \neq \emptyset$ if and only if for some $\sigma' \in \Sigma_c$ with $\eta(y,\sigma!)$, $g(\eta(y,\sigma!),k) \neq \emptyset$. If $g(y,k+1) = \emptyset$, then we have that, for any $\sigma' \in \Sigma_c$ with $\eta(y,\sigma!)$, $g(\eta(y,\sigma!),k) = \emptyset$, namely $v_k(\eta(y,\sigma!)) = +\infty$. Thus, by the induction hypothesis we have $\kappa_k(\eta(y,\sigma!)) = +\infty$, which means $\kappa_{k+1}(y) = +\infty$. On the other hand, since $g(y,k+1) = \emptyset$, we have $v_{k+1}(y) = +\infty$. Thus, $\kappa_{k+1}(y) = v_{k+1}(y)$. If $g(y,k+1) \neq \emptyset$, we can derive that,
\[
\min_{C \in \mathcal{E}(\eta(y,k+1))} \max_{\sigma \in C} \theta_S(y,s) = \min_{\sigma \in \Sigma_c \cap \eta(y,\sigma!)} (f'(y,\sigma') + \min_{C' \in \mathcal{E}(\eta(y,\sigma!),k)} \max_{C' \in C'} \theta_S(\eta(y,\sigma'),s'))
\]
Thus, by the induction hypothesis, we can still get that \( \kappa_{k+1}(y) = v_{k+1}(y) \).

Case 3: suppose there is no \( \sigma \in \Sigma \) such that \( \eta(y, \sigma)! \). Since \( S \) is nonblocking, we have that \( y \in Y_m \). Clearly, \( g(y, k+1) = g(y, k) = \{\{\epsilon\}\} \). Thus, \( v_{k+1}(y) = v_k(y) = 0 \). By the definition we have \( \kappa_{k+1}(y) = \kappa_k(y) \). By the induction hypothesis we have \( v_k(y) = \kappa_k(y) \). Thus, \( \kappa_{k+1}(y) = v_{k+1}(y) \), which means the lemma holds for Case 3.

Since in any case the lemma holds, we can conclude that the lemma is true.

2. Proof of Lemma 3.4: For any \( C \in g(y, |Y|) \), there are two possibilities: either \( C \in g(y, |Y| - 1) \), or \( C \notin g(y, |Y| - 1) \). For the latter case, there exists \( s \in \Sigma^* \) such that \(|s| = |Y|\). Let \( C' \in \gamma(y, s) \) be a maximal chain with respect to \( y \), and \( C' \subset C \). Such a \( C' \) must exist because \( C \) is a maximal chain. Let \( \hat{S} = (\hat{Y}, \Sigma, \hat{\eta}, \hat{y}_0, \hat{Y}_m) \) be a tree automaton such that

1. \( \hat{Y} := \overline{C} \)
2. \( \hat{Y}_m := C' \) and \( \hat{y}_0 := \epsilon \)
3. \( \hat{\eta} : \hat{Y} \times \Sigma \to \hat{Y} \), where for any \( \hat{y}, \hat{y}' \in \hat{Y} \) and \( \sigma \in \Sigma \), \( \hat{y}' \in \hat{\eta}(\hat{y}, \sigma) \) if \( \hat{y}' = \hat{y}\sigma \)

Since \( C' \in g(y, |Y|) \), the state set \( \hat{Y} \) is finite. Thus, \( \hat{S} \) is well defined. We can easily check that, \( L_m(\hat{S}) = C' \) and \( L(\hat{S}) = \overline{C} \). Let \( T = (\hat{Y}, \Sigma \cup \{\tau\}, \delta, \hat{y}_0, \hat{Y}_m) \) be a new tree automaton, where \( \tau \notin \Sigma \) and \( \delta \) is equal to \( \hat{\eta} \) when restricted to \( \hat{Y} \times \Sigma \). For any \( \hat{y}, \hat{y}' \in \hat{Y} \), if \( \hat{\eta}(\hat{y}, \hat{y}') = \hat{\eta}(\hat{y}, \hat{y}') \) and there exists \( \sigma \in \Sigma_c \) such that \( \hat{y}\sigma \leq \hat{y}' \) (remember that \( \hat{y} \) and \( \hat{y}' \) are strings), then add an edge \( \tau \) from \( \hat{y} \) to \( \hat{y}' \) in \( T \), namely \( \delta(\hat{y}, \tau) = \hat{y}' \). We now use the following procedure to modify \( T \).

1. Initially set \( T_0 = (\hat{Y}_0, \Sigma \cup \{\tau\}, \delta_0, \hat{y}_0, \hat{Y}_{m,0}) := T \)
2. Suppose we have \( T_k = (\hat{Y}_k, \Sigma \cup \{\tau\}, \delta_k, \hat{y}_0, \hat{Y}_{m,k}) \) we construct \( T_{k+1} \) as follows:
   
   (a) Pick two states \( \hat{y}_k, \hat{y}_k' \in \hat{Y}_k \) such that \( \delta_k(\hat{y}_k, \tau) = \hat{y}_k' \). Let \( \hat{g}(\hat{y}_k, \hat{y}_k') := \{\hat{y}_u' \in \hat{Y}_k | (\exists u, u' \in \Sigma^*) u \neq \epsilon \land \delta_k(\hat{y}_k, u) = \hat{y}_u' \land \delta_k(\hat{y}_k', u') = \hat{y}_k'\} \)
   
   If no such two states exist, then \( \hat{g}(\hat{y}_k, \hat{y}_k') := \emptyset \)
   
   (b) \( \hat{Y}_{k+1} := \hat{Y}_k \setminus \hat{g}(\hat{y}_k, \hat{y}_k') \) and \( \hat{Y}_{m,k+1} := \hat{Y}_{m,k} \cap \hat{Y}_{k+1} \)
   
   (c) \( \delta_{k+1} : \hat{Y}_{k+1} \times (\Sigma \cup \{\tau\}) \to \hat{Y}_{k+1} \), where for any \( \hat{y}, \hat{y}' \in \hat{Y}_{k+1} \) and \( \sigma \in (\Sigma \cup \{\tau\}) \), if \( \delta_k(\hat{y}_k, \sigma) = \hat{y}_k' \) then \( \delta_{k+1}(\hat{y}_k, \sigma) = \hat{y}_k' \); if \( \delta_k(\hat{y}_k', \sigma) \) then \( \delta_{k+1}(\hat{y}_k, \sigma) := \delta_k(\hat{y}_k', \sigma) \)

   Terminate when there exists \( r \in \mathbb{N} \) such that \( T_{r-1} = T_r \).

3. In \( T_r \) for each \( \hat{y} \in \hat{Y}_r \) if there exists \( \sigma \in \Sigma_{mc} \) such that \( \delta_r(\hat{y}, \sigma)! \), then remove all controllable edges at \( \hat{y} \). Let \( \hat{T} \) be the resultant reachable tree.

Since \( T \) is a tree, the above procedure will terminate in a finite number of iterations. We now show that, \( \hat{C} := L_m(\hat{T}) \) is a maximal chain in \( g(y, |Y| - 1) \). Since \( C' \) is a maximal chain, we can derive that,

\[
(\forall u \in \overline{C}) \{ \sigma \in \Sigma | u\sigma \in \overline{C} \} \cap \Sigma_c \leq 1
\]

By the construction of \( \hat{T} \) and the fact that \( C' \) is a maximal chain, we can derive that

\[
(\forall t \in \overline{C})(\forall \sigma \in \Sigma_{mc}) \eta(y, t\sigma)! \Rightarrow t\sigma \in \overline{C}
\]
Thus, \( \hat{C} \) is a maximal chain in \( g(y, |Y|) \). We now need to show that, \( \hat{C} \in g(y, |Y| - 1) \). Suppose it is not true. Then there exists \( t \in \hat{C} \) such that \( |t| = |Y| \). By the Pumping Lemma, there must exist \( t_1, t_2, t_3 \in \Sigma^* \) such that \( t = t_1 t_2 t_3 \), \( t_2 \neq \epsilon \) and \( \eta(t_1 t_3) = \eta(y, t_1 t_2) \). By the construction of \( \hat{T} \) we know that, there exists \( \sigma \in \Sigma_{uc} \) and \( t'_2 \in \Sigma^* \) such that \( t_2 = \sigma t'_2 \). Furthermore, by the definition of \( \hat{T} \) we can derive that, for any \( t'_2 \sigma' \leq t'_2 \), either \( \sigma' \in \Sigma_{uc} \) or for any \( \sigma'' \in \Sigma \), if \( \delta_r(t'_2 \sigma', \sigma'') \) then \( \sigma'' = \sigma' \) (namely \( \sigma' \) is the only exit transition at the state \( \delta_r(t'_2 \sigma', \sigma'') \)) if \( \sigma' \in \Sigma_c \). This means that, for any string \( s' \in \Sigma^* \), if \( \eta(y, s') = \eta(y, \delta_r(t_1 t_3)) \), then for any \( n \in \mathbb{N} \), \( s'(t_2)^n \in \Sigma^* \). But this contradicts the assumption that \( C' \in g(y, |Y|) \). Therefore, \( \hat{C} \in g(y, |Y| - 1) \). Clearly, \( \hat{C} \) is a maximal chain in \( g(y, |Y| - 1) \).

Thus, Lemma 2 is true.

3. Proof of Lemma 3.6: Let \( C \subseteq g(y_0, s) \). If \( C \) is not controllable with respect to \( G \), then there exists \( s' \in \Sigma^* \) and \( \sigma \in \Sigma_{uc} \) such that \( s' \sigma \in L(G) \) but \( s' \sigma \notin \Sigma^* \). Since \( C \subseteq L_m(S) \), which is controllable with respect to \( G \), we get that \( s' \sigma \in L_m(S) \). But this contradicts the assumption that \( C \) is a maximal chain.

Let \( K \subseteq L_m(S) \) be a nonempty controllable sublanguage with respect to \( G \). Pick \( s \in K \) and let \( c(y_0, s) \) be a maximal chain of \( s \) with respect to \( y_0 \). Suppose \( c(y_0, s) \in K \neq \emptyset \). For any \( s' \in c(y_0, s) \), clearly, \( s' \in c(y_0, s) \) but \( s' \notin K \). Since \( s \in c(y_0, s) \), we have \( \epsilon \in c(y_0, s) \). Thus, there exist \( t, t' \in \Sigma^* \) and \( \sigma \in \Sigma \) such that \( s' = \sigma t t' \), \( t \in K \) and \( t \notin K \). Since \( K \) is controllable with respect to \( G \), we get that \( \sigma \in \Sigma_c \). On the other hand, we have \( s' \in L_m(S) \). Thus, there exists \( t'' \in \Sigma^* \) such that \( s'' = \sigma t t'' \in K \). Clearly \( c(y_0, s) := (c(y_0, s) \cup \{ s'' \}) - \{ s' \} \) is also a maximal chain of \( s \) with respect to \( y_0 \). If we define a choice map \( q : \Sigma^* \to \Sigma^* \), which maps each \( s' \in c(y_0, s) \) to a string \( s'' \in K \), as described above, we can show that, \( c(y_0, s) := (c(y_0, s) \cup \{ g(s') \mid s' \in c(y_0, s) - K \} \) is a maximal chain of \( s \) with respect to \( y_0 \), and \( c(y_0, s) \subseteq K \). Thus, the lemma is true.

4. Proof of Theorem 3.7: (1) We first show that, if \( K = \emptyset \), then \( WC(G, E) = \emptyset \). By PSMWCS, if \( K = \emptyset \), then either \( L_m(G) \cap L_m(E) \) has no controllable sublanguage or \( \kappa_k(y_0) = +\infty \). For the former case, clearly, \( WC(G, E) = \emptyset \). For the latter case, by the proof of Theorem 3.5 we get that, \( \nu_k(y_0) = +\infty \), which, by Lemma 3.6, indicates that, for any controllable sublanguage \( K \subseteq L_m(G) \cap L_m(E) \) we have \( \omega_C(K) = \omega_S(K) = +\infty \). Thus, \( WC(G, E) = \emptyset \).

We now show that, if \( WC(G, E) = \emptyset \), then \( K = \emptyset \). Since \( WC(G, E) = \emptyset \), either \( L_m(G) \cap L_m(E) \) has no controllable sublanguage or there is no controllable sublanguage with a finite weight. For the former case, clearly \( K = \emptyset \) (see step (2.1)-(2.2) in PSMWCS). For the latter case, by Lemma 3.3 and Lemma 3.6 we have \( \kappa_k(y_0) = +\infty \). Thus, \( K = \emptyset \).

(2) Next, suppose \( K \neq \emptyset \) and we want to show that \( \hat{K} = \sup WC(G, E) \). To this end, we first show that \( \hat{K} \in WC(G, E) \). Clearly, \( \hat{K} \subseteq L_m(S) = L_m(G) \cap L_m(E) \). Furthermore, \( \omega_C(\hat{K}) = \omega_S(\hat{K}) = \kappa_k(y_0) < +\infty \). Thus, we only need to show that, \( \hat{K} \) is controllable with respect to \( G \) and \( \Sigma_{uc} \). To this end, for each state \( y \in Y' \) we have

\[
\kappa_k(y) = \begin{cases} 
\max_{\sigma \in \Sigma_{uc}} \eta(y, \sigma) (j'(y, \sigma) + \kappa_{k-1}(\eta(y, \sigma))) & \text{if } (\exists \sigma \in \Sigma_{uc}) \eta(y, \sigma) \\
\min_{\sigma \in \Sigma_{uc}} \eta(y, \sigma) (j'(y, \sigma) + \kappa_{k-1}(\eta(y, \sigma))) & \text{else if } (\exists \sigma \in \Sigma_{uc}) \eta(y, \sigma) \\
\kappa_{k-1}(y) & \text{otherwise}
\end{cases}
\]

For any \( \sigma \in \Sigma_{uc} \), if \( \eta(y, \sigma) \) then \( \kappa_k(\eta(y, \sigma)) = \kappa_{k-1}(\eta(y, \sigma)) \). Thus, by the definition of \( S' \) we get that, \( \eta(y, \sigma) \in Y' \). Furthermore, by induction we can show that,
there exist \( y_1, \cdots, y_n \in Y' \) and \( \sigma_1, \cdots, \sigma_n \in \Sigma \) with \( y_n \in Y_m \) such that
\[
y_1 = \eta'(y, \sigma_1) \land (\forall i \in \{2, 3, \cdots, n\}) \ y_i = \eta'(y_{i-1}, \sigma)
\]

Therefore, \( \bar{K} = \overline{L_m(S')} = L(S) \). Thus, \( \bar{K} = L_m(S') \) is controllable with respect to \( G \).

Next, we show that \( \omega_G(\bar{K}) = \omega_G(\sup WCN(G, E)) \). By the construction of \( S \) we get that, \( \omega_G(\bar{K}) = \omega_S(\bar{K}) \) and \( \omega_G(\sup WCN(G, E)) = \omega_S(\sup WCN(G, E)) \). For any \( K \in WCN(G, E) \subseteq L_m(S) \), it is controllable with respect to \( G \). Thus, by Lemma 3.6, \( K \) contains a maximal chain \( c(y_0, s) \) for some \( s \in L_m(S) \). Since \( \omega_G(\bar{K}) < +\infty \), we get that, there exists \( r \in \mathbb{N} \) such that \( c(y_0, s) \in \wp(y_0, r) \). If \( r \leq k \), from the proof of Theorem 3.5 we get that,
\[
\omega_G(\bar{K}) = v_k(y_0) = \min_{C \in \wp(y_0, k), s' \in G} \max_{s \in c(y_0, s)} \theta_S(y_0, s') = \omega_G(c(y_0, s)) \leq \omega_G(K)
\]

When \( r > k \), by the proof of Theorem 3.5 we get that, \( v_r(y_0) = \kappa_r(y_0) = \kappa_k(y_0) = v_k(y_0) \). Thus,
\[
\omega_G(\bar{K}) = v_k(y_0) = v_r(y_0) = \min_{C \in \wp(y_0, r), s' \in G} \max_{s \in c(y_0, s)} \theta_S(y_0, s') \leq \omega_G(c(y_0, s)) \leq \omega_G(K)
\]

Thus, in any case we have \( \omega_G(\bar{K}) \leq \omega_G(\sup WCN(G, E)) \). On the other hand, since \( \bar{K} \in WCN(G, E) \), we get that \( \omega_G(\bar{K}) \geq \omega_G(\sup WCN(G, E)) \). Thus, \( \omega_G(\bar{K}) = \omega_G(\sup WCN(G, E)) \).

Finally, we show that \( \bar{K} = \sup WCN(G, E) \). We have shown that \( \bar{K} \in WCN(G, E) \), which means \( \bar{K} \subseteq \sup WCN(G, E) \). Thus, we only need to show the opposite inclusion. For any \( s \in \sup WCN(G, E) \), let \( c(y_0, s) \) be a maximal chain of \( y_0 \). Clearly, \( c(y_0, s) \subseteq \sup WCN(G, E) \) because, otherwise \( \sup WCN(G, E) \) is not controllable. So we have
\[
\omega_S(c(y_0, s)) \leq \omega_S(\sup WCN(G, E)) = \omega_S(\bar{K})
\]

Thus, \( c(y_0, s) \subseteq L_m(S') = \bar{K} \), which means \( s \in \bar{K} \). Therefore, \( \sup WCN(G, E) \subseteq \bar{K} \).

5. Proof of Theorem 4.1: (1) Suppose \( K_{CN} = \emptyset \). Then we have three cases to consider:
Case 1.1: \( K = \emptyset \), namely \( L_m(G) \cap L_m(E) \) has no controllable and normal sublanguage;
Case 1.2: Step 3 terminates at \( k \) and \( \kappa_k(y_0) = +\infty \); Case 1.3: \( L_m(S') \neq \emptyset \) but \( \bar{K} = \emptyset \), namely \( L_m(S') \) has no controllable and normal sublanguage with respect to \( G \). For Case 1.1, clearly, \( WCN(G, E) = \emptyset \). For Case 1.2, by the proof of Theorem 3.7 we get that there is no controllable and normal sublanguage of \( L_m(G) \cap L_m(E) \) with respect to \( G \) with a finite weight. Thus, \( WCN(G, E) = \emptyset \). For Case 1.3, again it means there is no controllable and normal sublanguage of \( L_m(G) \cap L_m(E) \) with a finite weight. Thus, \( WCN(G, E) = \emptyset \). On the other hand, if \( WCN(G, E) = \emptyset \), then either \( L_m(G) \cap L_m(E) \) has no controllable and normal sublanguage or there is no controllable and normal sublanguage with a finite weight. In the former case, clearly, \( K_{CN} \neq \emptyset \) because \( K = \emptyset \). In the latter case, since every controllable and normal sublanguage of \( L_m(G) \cap L_m(E) \) is a controllable sublanguage of \( K \), we have two subcases to consider. Subcase 1: there is no controllable sublanguage of \( K \) with a finite weight. Then by Lemma 3.3 and Lemma 3.6, when the Step 3 of PSMWCNS terminates at \( k \), we have \( \kappa_k(y_0) = +\infty \). Thus, \( K_{CN} = \emptyset \).
Subcase 2: Every controllable sublanguage of \( K \) with a finite weight contains no controllable and normal sublanguage with a finite weight. In this subcase we can derive that, \( \bar{K} = \emptyset \). Thus, again \( K_{CN} = \emptyset \).

(2) Suppose \( K_{CN} \neq \emptyset \). We want to show that, \( K_{CN} = \sup WCN(G, E) \). Clearly, \( \bar{K}_0 = \bar{K} \in WCN(G, E) \). Furthermore, for any \( W \subseteq WCN(G, E) \) we have \( W \subseteq \bar{K} = \bar{K}_0 \). In particular, \( \sup WCN(G, E) \subseteq \bar{K}_0 \). Suppose Step (5) terminates at \( r + 1 \) with \( r \geq 0 \). We use induction to show that, \( \sup WCN(G, E) \subseteq \bar{K}_r \). It is true for \( l = 0 \). We assume that it is also true for \( l \) and we need to show that, it is true for \( l + 1 \leq r \). Since \( \sup WCN(G, E) \subseteq \bar{K}_l \), we have
\[
\omega_S(\bar{K}_l) \geq \omega_S(\sup WCN(G, E))
\]
For any \( s \in \hat{K}_l \) with \( \theta_U(y_0, s) = \omega_S(\hat{K}_l) \), we have \( s \in \psi(\hat{K}_l) \). Since \( \hat{K}_l \neq \emptyset \) and \( l < r \), we have \( \hat{K}_l - \psi(\hat{K}_l) \neq \emptyset \). Then

\[
\omega_S(\hat{K}_l - \psi(\hat{K}_l)) < \omega_S(\hat{K}_l)
\]

Let \( \hat{K}_{l+1} \) be the largest controllable and normal sublanguage of \( \hat{K}_l - \psi(\hat{K}_l) \) with respect to \( G \). Since \( l + 1 \leq r \), we have \( \hat{K}_{l+1} \neq \emptyset \). Thus, \( \hat{K}_{l+1} \in WCN(G, E) \), which means

\[
\omega_G(\sup WCN(G, E)) = \omega_S(\sup WCN(G, E)) \leq \omega_S(\hat{K}_{l+1}) < \omega_S(\hat{K}_l)
\]

Since \( \sup WCN(G, E) \subseteq \hat{K}_l \) and for any controllable and normal sublanguage \( W \subseteq \hat{K}_l \) with respect to \( G \), we have

\[
\omega_S(W) < \omega_S(\hat{K}_l) \Rightarrow W \subseteq \hat{K}_{l+1}
\]

we have

\[
\sup WCN(G, E) \subseteq \hat{K}_{l+1}
\]

Thus, the induction is true, namely \( \sup WCN(G, E) \subseteq \hat{K}_r \). Since \( \hat{K}_{r+1} = \emptyset \), for any controllable and normal sublanguage \( W \subseteq \hat{K}_r \) with respect to \( G \), we get that, \( W \cap \psi(\hat{K}_r) \neq \emptyset \), which means \( \Omega_S(W) = \Omega_S(\hat{K}_r) \). Since \( \sup WCN(G, E) \subseteq \hat{K}_r \), we have \( \omega_S(\sup WCN(G, E)) = \omega_S(\hat{K}_r) \). Thus, \( \sup WCN(G, E) = \hat{K}_r = K_{CN} \), and the theorem follows. ■


24 Bibliography