TECHNICAL
RESEARCH
REPORT

SYSTEMS
RESEARCH
CENTER

Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Language Stability and Stabilizability of Discrete Event Dynamical Systems

by R. Kumar, V. Garg and S.I. Marcus

TR 91-112r1

Language Stability and Stabilizability of Discrete Event Dynamical Systems [1]

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

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

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

July 21, 1992

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

Abstract

This paper studies the stability and stabilizability of Discrete Event Dynamical Systems (DEDS's) modeled by state machines. We define stability and stabilizability in terms of the behavior of the DEDS's, i.e. the language generated by the state machines (SM's). This generalizes earlier work where they were defined in terms of legal and illegal states rather than strings. The notion of reversal of languages is used to obtain algorithms for determining the stability and stabilizability of a given system. The notion of stability is then generalized to define the stability of infinite or sequential behavior of a DEDS modeled by a Büchi automaton. The relationship between the stability of finite and stability of infinite behavior is obtained and a test for stability of infinite behavior is obtained in terms of the test for stability of finite behavior. We present an algorithm of linear complexity for computing the regions of attraction, which is used for determining the stability and stabilizability of a given system defined in terms of legal states. This algorithm is then used to obtain efficient tests for checking sufficient conditions for language stability and stabilizability.

Keywords: Discrete Event Dynamical Systems, Automata Theory, Supervisory Control, Stability, Stabilizability

AMS(MOS) Subject Classification: 93

1 Introduction

Ramadge and Wonham in their work [21] on supervisory control of discrete event dynamical systems (DEDS) have modeled a DEDS, also called a plant, by a State Machine (SM), the event set of which is finite and is partitioned into sets of controllable and uncontrollable events. The language generated by such a SM is used as a model to describe the behavior of the plant at the logical level. The control task is to synthesize of a controller, also called a supervisor, which disables some of the controllable events in the plant so that the closed loop behavior equals some prespecified desired behavior, also called legal behavior. Supervisors which do not prevent any uncontrollable events from occurring are are called complete. Thus there may not always exist a complete supervisor so that the closed loop system has a prespecified desired behavior. Attention is then restricted to designing a complete supervisor that is *minimally restrictive* [21, 20, 10, 1, 11] so that the closed loop system can engage in some maximal behavior and still maintain the prescribed behavioral constraint. Thus the control objective is usually described as the synthesis of a minimally restrictive supervisor so that the controlled system has a maximally permissive legal behavior.

Sometimes such a constraint on the system behavior leads to the design of a supervisor which results in a very restrictive behavior [14, 15]. Recently there has been work [14, 15] on posing a supervisory control problem that allows the system to engage in some illegal behavior which can be tolerated. In this paper, we also allow the possibility of the system behaving illegally. The supervisor is synthesized so that the behavior of the supervised system is "asymptotically legal". In other words, the system is initially allowed to make illegal transitions but after a finite number of transitions the supervised system makes only legal transitions. With the above motivation, we define the stability and stabilizability of DEDS's in terms of their legal behavior.

In [16, 18, 2, 4, 3] the notion of stability and stabilizability of DEDS's has been presented in terms of the legal and illegal states of the system. In [2, 3] a stable system is one that starts from any arbitrary initial state and after finitely many transitions goes to one of the legal states and stays there; a stabilizable system is one for which there exists a supervisor so that the supervised system is stable. In [18] a system is said to be stable if after starting from any arbitrary initial state it visits the legal subset of states infinitely often; a system that can be made stable in the above context by the synthesis of an appropriate supervisor is called stabilizable. We define a system to be language-stable[1]. if its eventual behavior remains confined to the legal behavior; if a supervisor exists such that the supervised system is language-stable, then the system is called language-stabilizable. We show below that the existence of an eventually reachable legal set of states implies the existence of an eventually reachable legal behavior, whereas the converse is not always true. Thus the notion of stability presented here is finer than those in [18, 2, 3] in the sense that there need not exist any fixed set of legal states. A state can eventually be reached by legal as well as illegal strings, so none of the states can be predefined to be legal. In order to illustrate this point consider for

[1]We use the term language-stability to emphasize the fact that it is defined in terms of legal behavior rather than legal states in which case it may be called state-stability

example an elevator which moves between three floors - bottom, middle and top. Assume that a passenger requests service at the top floor. We can view the top floor to be the legal state, and require that the elevator should eventually reach it. However, a "finer" constraint that the elevator should reach the top floor in no more than two moves (there are total three floors) may be desired. In this case the top floor may be reached by legal as well as illegal sequence of moves of the elevator. Thus in this example the stability based on legal states cannot capture the desired behavior of the elevator.

In [18, 2, 3], the supervisors considered for stabilizing a system are assumed to be of *static feedback* type in which the next control action is determined by the current state of the system. In general a supervisor can be of *dynamic feedback* type where the next control action is determined not necessarily by the current state but by the "path" taken to reach the current state. We refine the notion of stability and stabilizability by defining it in terms of languages rather than states, and show that in some cases a static feedback type supervisor cannot stabilize a system and a more general dynamic feedback type supervisor is needed for stabilization. In [16], the stability of systems under partial observation is studied. In this case, the supervisor is of dynamic feedback type; it can be represented as a cascade of a dynamic state observer followed by a static feedback type controller. The supervisor considered for eventually restrictable systems in [17] is also of dynamic feedback type.

We start with the description of DEDS's and present some of the notions of stability defined in terms of states. The computational complexity of the algorithms presented in [2, 3] for determining the stability and stabilizability of DEDS's based on computing the *regions of attraction* is quadratic in the number of states of the system. We present an algorithm that is linear in the number of states of the system and is thus computationally more efficient. We then introduce the notion of stability in terms of languages and provide algorithms for determining the stability and stabilizability of a given system by considering an equivalent problem defined in terms of *reversal* of languages. We also discuss the computational complexity of these algorithms. Later, we provide computationally more efficient algorithms for testing the sufficiency of stability and stabilizability of systems based on our algorithm for computing the regions of attraction. In all this, we assume that perfect observation of the system behavior is possible so that the control actions are determined on the basis of observing the system evolution perfectly. We also introduce a weaker notion of language stability that is preserved under union and provide a technique for constructing the minimally restrictive stabilizing supervisor in this weaker sense of language stability.

The notion of language stability is then generalized to study the stability of sequential behaviors of DEDS's modeled by Büchi automata. The notions of ω-stability and ω-stabilizability are introduced in this context, and tests for verifying stability and stabilizability of sequential behavior are obtained by reducing the problem of testing them to the problem of testing language stability. We introduce a equivalence relation on the space of infinite strings and obtain a necessary condition of ω-stability in terms of this equivalence relation.

2 Notation and Terminology

A DEDS to be controlled, called a *plant*, is modeled as a deterministic trim [8] state machine (SM) following the framework of [21]. Let the quintuple

$$P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$$

denote a SM representing a plant, where X denotes the state set; Σ denotes the finite event or alphabet set; $\alpha : \Sigma \times X \rightarrow X$ denotes the partial state transition function; $x_0 \in X$ denotes the initial state; and $X_m \subseteq X$ denotes the set of marked states. The transition function $\alpha(\cdot, \cdot)$ is extended to $\Sigma^\star \times X$ in the natural way, where $\Sigma^\star$ denotes the set of all finite sequences of events belonging to Σ. The notation $\epsilon \in \Sigma^\star$ is used to denote the empty string. The behavior of P is described by the language $L(P) \subseteq \Sigma^\star$ that it *generates* and $L_m(P) \subseteq L(P)$ that it *marks* or *recognizes*. Formally,

$$L(P) = \{s \in \Sigma^\star \mid \alpha(s, x_0)!\}; L_m(P) = \{s \in L(P) \mid \alpha(s, x_0) \in X_m\},$$

where the notation "!" is used to denote "is defined". By definition, $L(P)$ is prefix closed and also since P is trim, $\overline{L_m(P)} = L(P)$ [8].

The event set is partitioned into $\Sigma = \Sigma_u \cup \Sigma_c$, the set of *uncontrollable* and *controllable* events. A supervisor S for controlling a plant is another DEDS, also represented as a SM,

$$S \stackrel{\text{def}}{=} (Y, \Sigma, \beta, y_0, Y_m)$$

S operates synchronously with P, thus allowing only the synchronous transitions to occur in the closed loop system described by the SM [10, 11]

$$P \Box S \stackrel{\text{def}}{=} (Z, \Sigma, \gamma, z_0, Z_m),$$

where $z_0 = (x_0, y_0)$; for $s \in \Sigma^\star, \gamma : \Sigma^\star \times Z \rightarrow Z$ is defined as: $\gamma(s, z_0)) = (\alpha(s, x_0), \beta(s, y_0))$ if $\alpha(s, x_0)!$ and $\beta(s, y_0)!$, undefined otherwise; $Z = \{z \in X \times Y \mid \exists s \in \Sigma^\star \text{ s.t. } \gamma(s, z_0) = z\}$; and $Z_m = Z \cap (X_m \times Y_m)$. Thus $Z \subseteq X \times Y$ is the set of states that are reachable from the initial state z_0, and $Z_m \subseteq Z$ is the set of those reachable states that have both their "co-ordinates" marked.

The following states the control achieved by the synchronous operation of P and S.

Remark 2.1 [11, 10] Let L(P□S) be the language generated and $L_m(P \Box S)$ the language marked by P□S; then $L(P \Box S) = L(P) \cap L(S)$, and $L_m(P \Box S) = L_m(P) \cap L_m(S)$, where $L(S), L_m(S)$ denote the languages generated, recognized by S respectively.

Also, since S can disallow only the controllable events from occurring, $L(P) \cap \Sigma_u^\star \subseteq L(P \Box S)$, where $\Sigma_u^\star$ is the set of finite sequences of events belonging to Σ_u.

The supervisor as defined above represents a closed loop control policy. This differs from open loop control policy in which control actions are all prespecified; in closed loop control, control actions are determined by observing all or part of the history of the system evolution.

Definition 2.2 Let the map $f : L(P) \rightarrow 2^{\Sigma}$ denote a *control policy* as described in [21], i.e. for each string $s \in L(P)$ generated by the plant P, $f(s) \subseteq \Sigma$ is the set of events that are not disabled by a supervisor. Then the control exercised by the synchronous operation of a supervisor and the plant, as described above, defines the following control policy over the set of strings generated by the plant:

$$f(s) \stackrel{\text{def}}{=} \begin{cases} \{\sigma \in \Sigma | \gamma(s\sigma, z_0)!\} & \text{if } \gamma(s, z_0)! \\ \text{undefined} & \text{otherwise} \end{cases}$$

where the string $s \in L(P)$.

Closed loop controllers can further be classified into static and dynamic control type. Given a deterministic SM, $V \stackrel{\text{def}}{=} (Q, \Sigma, \delta, q_0, Q_m)$, there is a natural equivalence relation R_V [8, 6, 11, 10] induced by V on $\Sigma^\star$, which is defined by $s \cong t(R_V) \Leftrightarrow \delta(s, q_0) = \delta(t, q_0)$ (this is meant to include the condition that $\delta(s, q_0)$ is undefined $\Leftrightarrow \delta(t, q_0)$ is undefined), where $s, t \in \Sigma^\star$. Thus all those strings which upon execution result in the same state in V belong to the same equivalence class. We use $[s](R_V)$ to denote the equivalence class under the equivalence relation R_V, containing the string s.

Definition 2.3 Consider the control policy $f : L(P) \rightarrow 2^{\Sigma}$ defined by the synchronous composition operator as described in Definition 2.2. We say that a closed loop control policy is *static* if $s \cong t(R_P) \Rightarrow f(s) = f(t)$ whenever both $f(s), f(t)$ are defined.

In other words, in a static feedback type control, the same control action is applied after the execution of all strings that lead to the same state in the plant. Next we show that if a supervisor exercises a static closed loop control, then it can be represented as a SM having structure similar to that of the plant.

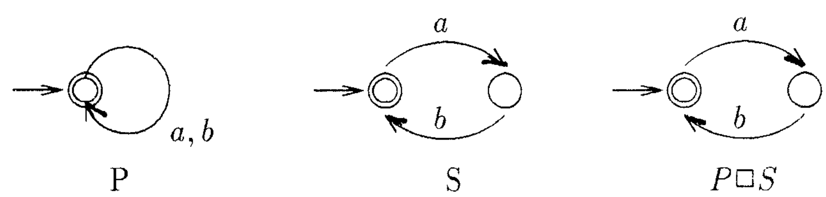
Definition 2.4 Let $V_1 \stackrel{\text{def}}{=} (Q_1, \Sigma, \delta_1, q_{0_1}, Q_{m_1})$ and $V_2 \stackrel{\text{def}}{=} (Q_2, \Sigma, \delta_2, q_{0_2}, Q_{m_2})$ be two SM's. V_1 is said to be a *subautomaton* [5] of V_2 if there exists a one-to-one map $h : Q_1 \rightarrow Q_2$ such that $h(\delta_1(s, q_{0_1})) = \delta_2(s, q_{0_2})$ for each $s \in L(V_1)$.

Thus if V_1 is a subautomaton of V_2, then $L(V_1) \subseteq L(V_2)$. Note that if the map h in Definition 2.3 is also onto, then V_1 and V_2 are structurally identical.

Proposition 2.5 [9] The following are true:

1. If S is a subautomaton of P, then the control policy $f : L(P) \rightarrow 2^{\Sigma}$ defined by S is static.
2. If $f : L(P) \rightarrow 2^{\Sigma}$ is a static control policy, then there exists S which defines the same control policy as f and is a subautomaton of P.

Definition 2.6 A closed loop control policy is said to be *dynamic* if it is not static.



$$L_m(P\Box S) = L_m(S) = (ab)^\star$$

Figure 1: Diagram illustrating Example 2.7

Example 2.7 Consider for example a plant P, with language $L(P) = (a + b)^\star$ defined over the event set $\Sigma = \{a, b\}$. Assume that $\Sigma_c = \Sigma$ (see Figure 1; "◯" denotes the states, an entering arrow "⟶" to "◯" represents the initial state, and "◎" denotes the marked states). Then the language generated by the coupled system under a static feedback control policy could be one of the following: $L(P\Box S) = (a + b)^\star$ or $a^\star$ or $b^\star$ or ϵ depending on whether the events disabled in the only state of the system are $\emptyset, \{b\}, \{a\}$ or $\{a, b\}$.

On the other hand, the language marked by the coupled system can be made to be any sublanguage $K \subseteq (a + b)^\star$ by using a dynamic feedback control policy. This can be done because all the events are controllable [10, 11] (pick the supervisor S, so that L(S) = K). An example for the case $K = (ab)^\star$ is shown in Figure 1.

3 Stability: Region of Attraction

With the above introduction on our supervisory control model, we next consider the stability issues for DEDS's. First we discuss the definitions and results of some of the earlier works, in which the stability is defined in terms of a set of legal states of the system. Later, we present our own notions of stability defined in terms legal behavior of the system.

Consider a plant $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$. Let $\hat{X} \subseteq X$ be the prescribed subset of states or the legal states. The notions of *strong* and *weak attraction* [2, 3] are defined as follows: A state $x \in X$ is said to be *strongly attractable* to $\hat{X}$, if after starting from the state x, the system always reaches a state in the set $\hat{X}$ after a finite number of transitions. The set of all the strongly attractable states is called the *region of strong attraction* of $\hat{X}$ and is denoted by $\Omega(\hat{X})$. Formally, let for $s \in \Sigma^\star, |s|$ denote the length of s, and for $X' \subseteq X, |X'|$ denote the number of states in the set X'.

Definition 3.1 $x \in X$ is strongly attractable to $\hat{X}$ if for all s such that $\alpha(s, x)!$ and $|s| \geq |X - \hat{X}|$ there exists a prefix $u_s \in \Sigma^\star$ of s with $|u_s| \leq |X - \hat{X}|$ so that $\alpha(u_s, x) \in \hat{X}$ [2, 3].

Definition 3.2 A state $x \in X$ is said to be *weakly attractable* to $\hat{X}$, if there exists a supervisor S such that x is strongly attractable to $\hat{X}$ in the coupled system $P\Box S$. The set of all the weakly attractable states is called the *region of weak attraction* and is denoted by $\Lambda(\hat{X})$.

Clearly, $\Omega(\hat{X}) \subseteq \Lambda(\hat{X})$. If $\Omega(\hat{X}) = X$, then P is said to be *stable* with respect to $\hat{X}$ and if $\Lambda(\hat{X}) = X$, then P is said to be *stabilizable* with respect to $\hat{X}$. Thus in order to test whether a given system is stable (stabilizable) with respect to a given set of legal states, one needs to compute the region of strong (weak) attraction. The definitions of strongly and weakly attractable states are the same as those of *prestable* and *prestabilizable* states, respectively [18].

Remark 3.3 Algorithms for constructing the regions of strong and weak attraction are presented in [18, 2, 3]. The complexity of these algorithms is quadratic in number of states of the system. An algorithm of linear time complexity in number of states of the system for constructing the regions of strong and weak attraction is presented in the Appendix A of this paper. This algorithm is used later for arriving at computationally more efficient test for determining a sufficient condition of stability and stabilizability introduced below.

4 Language-Stability

So far we have discussed stability of DEDS's defined in terms of their legal states and provided an efficient algorithm for testing it by computing the regions of attraction (refer to Appendix A for the algorithm). Next we provide motivation for a more general notion of stability which we call language-stability and discuss some of the issues related to stability in this framework.

In some cases, it might be desirable that the eventual behavior (rather than the whole behavior) of the system be legal, so the whole behavior of the system need not be confined to a legal language as in [21, 20]. Thus in these cases the control task can be formulated as the synthesis of a supervisor such that the behavior of the supervised system is eventually legal. This leads to the design of supervisors that are less restrictive and as a result, the behavior of the supervised system is a larger language. Hence, we will formalize the notion of eventual behavior of the systems and define stability and stabilizability of systems in terms of their behavior. As discussed in the previous sections, the notions of stability defined in terms of languages can also be viewed as a generalization to the ones defined in terms of states [18, 16, 2, 4, 3].

Example 4.1 Consider the machine P shown in Figure 2. P can either be in "idle", "working", "broken" or "display" state. Assume that initially it is in the idle state and goes to the working state when the action "start" is executed. While in the working state, P can either "stop" and go back to the idle state or can "fail" and go to the broken state. In the broken state it can execute either the action "repair" and go to the display state or the action "replace" and get back to the initial idle state. While in the display state, the action "reject" or "approve" can be executed, so the resulting state of P can either be broken (if reject is executed) or idle (if approve is executed).

Consider the above example for the stability analysis in the framework of [18, 16, 2, 4, 3]. The states idle and working are the "good" or legal states of P. The actions start, repair and

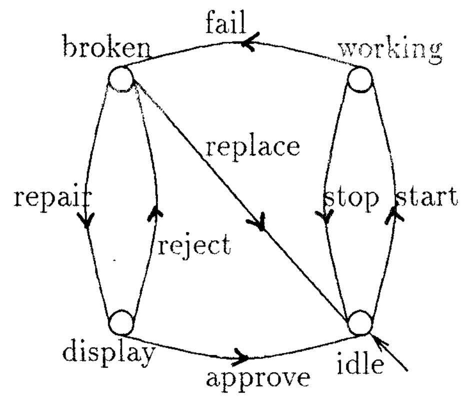


Figure 2: Machine P of Example 4.1

replace are the controllable actions, whereas the actions stop, fail, reject and approve are the uncontrollable actions. Clearly, P is not *stable* with respect to its legal states (once P executes fail, it is not guaranteed to get back to the legal states). To show that P is *stabilizable*: once it executes fail and goes to the broken state, it must execute the controllable action replace to go back to the legal state either permanently (as in [4, 2, 3]) or temporarily (as in [18]). Suppose instead, it executes the controllable action repair and goes to the display state; there it might not execute the uncontrollable action approve in which case it would remain in the illegal state. Hence the only way P can be stabilized is by executing the action replace after it executes fail. This however, may not be desired, for replacing (and not repairing) P whenever it fails might be cost ineffective. Thus in this example, the framework of [18, 16, 3] may be too restrictive for *stabilizing* the machine P.

We would like the desired behavior of P to be such that it allows P to execute the repair–reject sequence for a finite number of times. In other words, the desired behavior of P is that if it executes fail, it should execute replace or approve after a finite number of executions of the repair–reject sequence; otherwise it should execute the start–stop sequence. The way P is designed, after executing fail, it might never execute replace or approve and continue executing the repair–reject sequence, in which case the desired behavior is not achieved. We note that the desired behavior of P as described above cannot be achieved by use of a static feedback controller.

Moreover, in the above example, P is allowed to execute "illegal" actions (the repair–reject sequence) after it executes fail, provided it eventually executes one of the "legal" actions (replace or approve). Thus the whole behavior of the system need not always be confined to a legal language as in [21, 20]. With these motivations, the notions of stability of systems is formally defined in terms of their legal behavior:

With this motivation, we formally define stability of systems in terms of their legal behavior. For $n \in \mathcal{N}$, let Σ^n denote the set of strings, each of length n, of events belonging to Σ. We use $\Sigma^{\leq N}$ to denote $\bigcup_{n \leq N} \Sigma^n$ for each $N \in \mathcal{N}$.

Definition 4.2 Let $L, K \subseteq \Sigma^\star$ be two languages. L is said to be *language stable (ℓ-stable)* with respect to K if there exists $N \in \mathcal{N}$ such that $L \subseteq \Sigma^{\leq N} K$.

Since $\Sigma^{\leq N} \subseteq \Sigma^{\leq N'}$ whenever $N \leq N'$ $(N, N' \in \mathcal{N})$, it follows that if L is ℓ-stable with respect to K, then there exists a smallest integer $N_0 \in \mathcal{N}$ such that $L \subseteq \Sigma^{\leq N_0} K$. Given a string $s \in \Sigma^\star$, let $u_n \in \Sigma^\star$ be the prefix of length n of s $(n < |s|)$, and let $v_n \in \Sigma^\star$ be such that $s = u_n v_n$. We define a map $\Pi_n : \Sigma^\star \to \Sigma^\star$ in the following manner:

$$\Pi_n(s) = \begin{cases} v_n & \text{for } n < |s| \\ \epsilon & \text{otherwise} \end{cases}$$

Thus the effect of the map $\Pi_n(\cdot)$ on a string s is to remove the initial n symbols of s.

It follows from Definition 4.2 that $L \subseteq \Sigma^\star$ is ℓ-stable with respect to $K \subseteq \Sigma^\star$ if and only if there exists $N \in \mathcal{N}$ such that for every string $s \in L$ there exists a prefix $u_s \in \Sigma^\star$ of s with $|u_s| \leq N$ such that $\Pi_{|u_s|}(s) \in K$. Thus L is ℓ-stable with respect to K if after removing a prefix of length at most N from a string in L, it matches some string in K. The language L can be thought to be representing the plant behavior and the language K can be thought to be representing the eventual legal behavior of plant. If L is not ℓ-stable with respect to K, then it is said to be *ℓ-stabilizable* with respect to K if there exists a supervisor S such that the closed loop behavior is ℓ-stable with respect to K. Formally,

Definition 4.3 Consider $L, K \subseteq \Sigma^\star$. L is said to be *ℓ-stabilizable* with respect to K if there exists a nonempty controllable [21] sublanguage $H \subseteq L$ such that H is ℓ-stable with respect to K.

Assume that L is recognized by a plant P, i.e. $L_m(P) = L$. Let S be a supervisor such that the language recognized by the closed loop system $L_m(P \square S)$ is ℓ-stable and controllable with respect to K; then clearly L is ℓ-stabilizable with respect to K with $H = L_m(P \square S)$. It is know that the closed loop behavior $L_m(P \square S)$ is controllable if and only if S is a *complete*[2] supervisor [21, 11, 10]. Thus Definition 4.3 can equivalently be stated as: L is said to be *ℓ-stabilizable* with respect to K if there exists a complete supervisor S such that $L_m(P \square S)$ is ℓ-stable with respect to K.

Proposition 4.4 If $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$ is stable (stabilizable) with respect to $\hat{X} \subseteq X$, then $L_m(P)$ is ℓ-stable (ℓ-stabilizable) with respect to $\bigcup_{x \in \hat{X}} L_m(P, x)$, where $L_m(P, x)$ is the language marked by P assuming the initial state to be x.

Proof: Assume that the SM, $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$ is stable with respect to the legal set $\hat{X} \subseteq X$. Let $L = L_m(P)$, and $K = \bigcup_{x \in \hat{X}} L_m(P, x)$. Define $N \stackrel{\text{def}}{=} |X - \hat{X}|$. We will show that $L \subseteq \Sigma^{\leq N} K$. Consider $s \in L$. If $|s| \leq N$, then $s \in \Sigma^{\leq N}$; hence $s \in \Sigma^{\leq N} K$. If $|s| > N$, then there exists a prefix $u_s < s$, $|u_s| \leq N$, such that $\alpha(u_s, x_0) \in \hat{X}$ (follows from the fact that

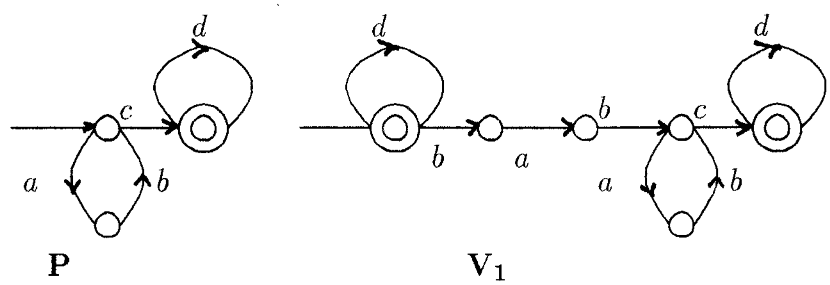
[2]A supervisor S is said to be *complete* if for all $s \in \Sigma^\star, \sigma_u \in \Sigma_u : s \in L(P \square S), s\sigma_u \in L(P) \Rightarrow s\sigma_u \in L(P \square S)$.

x_0 is strongly attractable to $\hat{X}$). Thus $\Pi_{|u_s|}(s) \in K$ (by definition of K). Hence $s \in \Sigma^{\leq N}K$; which shows that L is ℓ-stable with respect to K.

Similarly, it can be shown that if P is stabilizable with respect to $\hat{X}$, then L is ℓ-stabilizable with respect to K. □

Proposition 4.4 shows that stability (stabilizability) in terms of states in some sense implies ℓ-stability (ℓ-stabilizability). We show in the next example that the converse does not necessarily hold, thus showing that the notion of ℓ-stability (ℓ-stabilizability) is finer than that of stability (stabilizability).

Example 4.5 Let $\Sigma = \Sigma_u = \{a, b, c, d\}$. Consider the languages $L, K_i (i \geq 1) \subseteq \Sigma^\star$ given by: $L = (ab)^\star cd^\star$ and and $K_i = d^\star + b(ab)^i(ab)^\star cd^\star$. Generators for L and $K_1 = d^\star + bab(ab)^\star cd^\star$ are shown in Figure 3. Letting $N \stackrel{\text{def}}{=} 2i + 1$, it can be easily verified that $L \subseteq \Sigma^{\leq N}K_i$ for each $i \geq 1$, and also that N is the smallest integer for which the last inclusion holds. Fix for example $i = 1$. We show that $L \subseteq \Sigma^{\leq 3}K_1$. L consists of strings $cd^\star$ (no ab followed by $cd^\star$), $abcd^\star$ (one ab followed by $cd^\star$), and $(ab)^{\geq 2}cd^\star$ (two or more ab followed by $cd^\star$). First consider the strings in $cd^\star \subseteq L$. Then $\Pi_1(cd^\star) = d^\star \subseteq K_1$. Next consider the strings in $abcd^\star \subseteq L$. Then $\Pi_3(abcd^\star) = d^\star \subseteq K_1$. Finally consider $(ab)^{\geq 2}cd^\star \subseteq L$. Then $\Pi_1((ab)^{\geq 2}cd^\star) = b(ab)^{\geq 1}cd^\star = bab(ab)^\star cd^\star \subseteq K_1$. Thus it follows that $L \subseteq \Sigma^{\leq 3}K_1$.



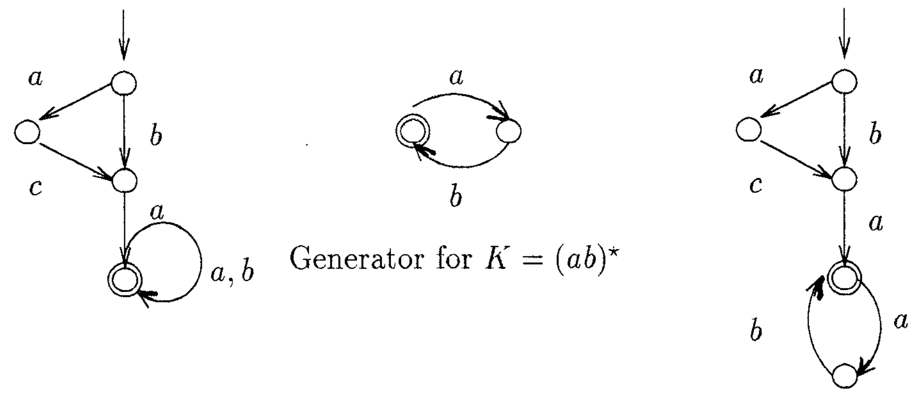
$L = L_m(P) = (ab)^\star cd^\star$ $\quad$ $K_1 = L_m(V_1) = d^\star + bab(ab)^\star cd^\star$

Figure 3: Diagram illustrating Example 4.5 with $i = 1$

Since L is ℓ-stable with respect to each K_i it follows that L is also ℓ-stabilizable with respect to each K_i.

Let P, V_i be the minimal SM's generating L, K_i respectively. Then P, V_i must have $3, 2i+4$ states respectively (refer to Figure 3 for $i = 1$). It can be easily seen that P is not stable with respect to any of its subset of states. Since $\Sigma_u = \Sigma$, P is not stabilizable with respect to any of its subset of states either.

Example 4.6 Consider the languages $L = (ac + b)a(a + b)^\star$ and $K = (ab)^\star$ defined over $\Sigma = \Sigma_c = \{a, b, c\}$. We will show that L is not ℓ-stable with respect to K, i.e. there exists no $N \in \mathcal{N}$ such that $L \subseteq \Sigma^{\leq N}K$. To prove this, we assume for contradiction that there exists $N_0 \in \mathcal{N}$ is such that $L \subseteq \Sigma^{\leq N_0}K$. Consider the string $baa^{N_0} \in L$. Any substring of it obtained by removing an initial finite segment of length less than N_0 does not match any string in K (a string in K contains the symbol b at the end, whereas the string baa^{N_0} ends with the symbol a).



Generator for $K = (ab)^\star$

Generator for $L = (ac + b)a(a + b)^\star$ Generator for $H = (ac + b)a(ab)^\star$

Figure 4: Diagram illustrating Example 4.6

Consider a sublanguage $H = (ac+b)a(ab)^\star \subseteq L$ as shown in Figure 4. Since $\Sigma_c = \Sigma$, H is controllable with respect to L. It can be easily seen that $H = (ac+b)a(ab)^\star$ is ℓ-stable with respect to $K = (ab)^\star$ (consider any string from H and remove the initial segment, either aca or ba, whichever is appropriate; the resulting string belongs to K). Thus L is ℓ-stabilizable to K.

In this example, it is clear that a dynamic feedback type supervisor has been used to ℓ-stabilize the given language. Also, a static feedback type control cannot be used to stabilize $L = (ac+b)a(a+b)^\star$ with respect to $K = (ab)^\star$. This follows since any string in K contains an equal number of a's and b's, and L cannot be restricted to a language $H \subseteq L$ with all its strings having an equal number of a's and b's at its end by using a static supervisor (refer to Example 2.7). In [18, 2, 3], where stability is defined in terms of the legal states, the supervisors considered for stabilizing DEDS's are all assumed to be of static feedback type. Thus a more general type of control is needed to ℓ-stabilize the behavior of a given system, which also shows that the notion of ℓ-stability (ℓ-stabilizability) is a finer notion.

Example 4.7 Consider a system consisting of a single buffer of unbounded capacity. Only two types of events: arrival, denoted a, and departure, denoted b, occur in this system, i.e. $\Sigma = \{a, b\}$. The behavior of this system can be described the language:

$$L = \{s \in \Sigma^\star \mid \#(a, s) \geq \#(b, s)\},$$

where the symbol $\#(x, y)$ is used to denote the number of times the symbol x occurs in the string y. We may be interested in determining whether there exists some number $N \in \mathcal{N}$ such that after execution of all strings of length larger than N, the buffer content is bounded above by a fixed number $N_0 \in \mathcal{N}$. The above problem can be posed as a ℓ-stability problem with the "eventually reachable" language $K \subseteq \Sigma^*$ defined as:

$$K = \{s \in \Sigma^* \mid \#(a, s) - \#(b, s) \leq N_0\}.$$

K corresponds to the content of the buffer being bounded above by N_0.

It is easy to see that L is not ℓ-stable with respect to K, i.e. there does not exist any N such that after execution of all strings of length larger than N, the buffer content is bounded above by N_0. Note that in this example $K \subseteq L$, and if the arrival event a is controllable, then L can be restricted to the language K by disabling a whenever the buffer content becomes equal to N_0. This proves that L is ℓ-stabilizable with respect to K.

Next we present algorithms for testing ℓ-stability and ℓ-stabilizability of a language L with respect to another language K.

4.1 Algorithms for testing ℓ-stability and ℓ-stabilizability

In order to test whether a language L is ℓ-stable (ℓ-stabilizable) with respect to another language K, we need to test whether there exists an integer $N \in \mathcal{N}$ such that $L \subseteq \Sigma^{\leq N} K$ ($H \subseteq \Sigma^{\leq N} K$, where $H \subseteq L$). This problem can equivalently be posed in terms of the *reversal* [1] of languages that we define next.

Definition 4.8 Given a string $s \in \Sigma^*$, its *reversal* $s^R \in \Sigma^*$, is the string obtained by reversing s. Given a language $L \subseteq \Sigma^*$, its reversal $L^R \subseteq \Sigma^*$ is defined to be: $L^R \stackrel{\text{def}}{=} \{s^R \in \Sigma^* | s \in L\}$.

Next we discuss some of the properties of the reversal operator. We use L, L_1, L_2 to denotes languages defined on Σ.

Lemma 4.9

1. Reversal preserves regularity, i.e. if L is regular, then so is L^R.
2. $(L^R)^R = L$.
3. Reversal is monotone, i.e. if $L_1 \subseteq L_2$, then $L_1^R \subseteq L_2^R$.
4. $(L_1 L_2)^R = L_2^R L_1^R$.

Proof: 1. The proof is based on constructing a FSM that recognizes L^R using a FSM realization for L, and can be found in [8].

2. Follows from the definition of the reversal of languages and the fact that for any string $s \in \Sigma^*$, $(s^R)^R = s$.

3. Pick $s \in L_1^R$; then $s^R \in L_1$. Since $L_1 \subseteq L_2$, it follows that $s^R \in L_2$, i.e. $(s^R)^R = s \in L_2^R$.

4. We first show that $(L_1L_2)^R \subseteq L_2^R L_1^R$. Pick $s \in (L_1L_2)^R$; then $s^R \in L_1L_2$, i.e. there exist $u_s \in L_1$ and $v_s \in L_2$ such that $u_sv_s = s^R$. Hence $s = (s^R)^R = (u_sv_s)^R = v_s^R u_s^R \in L_2^R L_1^R$.

Next we show that $L_2^R L_1^R \subseteq (L_1L_2)^R$. Pick $s \in L_2^R L_1^R$; then there exist $v_s \in L_2$ and $u_s \in L_1$ such that $v_s^R u_s^R = s$. Hence $s = (s^R)^R = ((v_s^R u_s^R)^R)^R = (u_sv_s)^R \in (L_1L_2)^R$. □

Corollary 4.10 $L \subseteq \Sigma^{\leq N} K$ if and only if $L^R \subseteq K^R \Sigma^{\leq N}$, where $L, K \subseteq \Sigma^\star$ and $N \in \mathcal{N}$.

Proof: Assume that $L \subseteq \Sigma^{\leq N} K$; then it follows from part 3 of Lemma 4.9 $L^R \subseteq (\Sigma^{\leq N} K)^R$. Since $(\Sigma^{\leq N})^R = \Sigma^{\leq N}$, it follows from part 4 of Lemma 4.9 that $L^R \subseteq K^R \Sigma^{\leq N}$.

Assume next that $L^R \subseteq K^R \Sigma^{\leq N}$; then from part 3 of Lemma 4.9 it follows that $(L^R)^R \subseteq (K^R \Sigma^{\leq N})^R$. Thus from part 4 of Lemma 4.9 we obtain $(L^R)^R \subseteq \Sigma^{\leq N} (K^R)^R$. It then follows from part 2 of Lemma 4.9 that $L \subseteq \Sigma^{\leq N} K$. □

Thus the problem of testing ℓ-stability of a language L with respect to another language K can be equivalently posed as that of determining an integer $N \in \mathcal{N}$, if it exists, such that $L^R \subseteq K^R \Sigma^{\leq N}$. Hence, given two languages $L, K \subseteq \Sigma^\star$, we next analyze the problem of determining an integer $N \in \mathcal{N}$, if it exists, such that $L^R \subseteq K^R \Sigma^{\leq N}$.

Let $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$ and $V \stackrel{\text{def}}{=} (Q, \Sigma, \delta, q_0, Q_m)$ be two SM's such that $L_m(P) = L^R$ and $L_m(V) = K^R$. Assume further that P is trim [8] so that $L(P) = \overline{L_m(P)} = \overline{L^R}$, and V is such that $L(V) = \Sigma^\star$, i.e. V is a SM that recognizes K^R and has an additional dump state in order to generate $\Sigma^\star$. Consider a slightly different synchronous composition of P and V, denoted $P\Box'V$, given by the 5-tuple:

$$P\Box'V \stackrel{\text{def}}{=} (R, \Sigma, \rho, r_0, R_m)$$

where the state set R, the transition function $\rho(\cdot,\cdot)$, and the initial state r_0 are defined as in the definition of synchronous composition in section 2, and $R_m = \{r \in X_m \times Q \mid \exists s \in \Sigma^\star \text{ s.t. } \rho(s, r_0) = r\}$. This is a slight variation to the earlier definition of marked states in synchronous composition of two state machines. Note that R_m consists of those states in $X_m \times Q$ that are reachable from the initial state r_0, hence $R_m \subseteq X_m \times Q$. Also, note that all transitions are defined in all states of V, i.e. given any event $\sigma \in \Sigma$ and any state $q \in Q$, $\delta(\sigma, q)!$. Hence for any event $\sigma \in \Sigma$ and state $r = (x, q) \in R$, $\rho(\sigma, (x, q))$ is defined if and only if $\alpha(\sigma, x)$ is defined.

Lemma 4.11 Let P and V be the two SM's as defined above. Then $L_m(P\Box'V) = L_m(P)$, and $L(P\Box'V) = L(P)$.

Proof: First we show that $L_m(P\Box'V) \subseteq L_m(P)$. Pick $s \in L_m(P\Box'V)$; then $\rho(s, r_0) \in R_m$. Since $\rho(s, r_0)$ is defined if and only if $\alpha(s, x_0)$ is defined, and $R_m \subseteq X_m \times Q$, it follows that $\alpha(s, x_0) \in X_m$. Thus $s \in L_m(P)$. Next we show that $L_m(P) \subseteq L_m(P\Box'V)$. Pick $s \in L_m(P)$; then $\alpha(s, x_0) \in X_m$. Again, since $\alpha(s, r_0)$ is defined if and only if $\rho(s, r_0)$ is defined, $\rho(s, r_0) \in X_m \times Q$. The state $\rho(s, r_0)$ is clearly a reachable state from r_0, hence $\rho(s, r_0) \in R_m$, which shows that $s \in L_m(P\Box'V)$.

Since $L(P\Box'V) = L(P) \cap L(V) = L(P) \cap \Sigma^\star$=L(P), the other result follows. □

Given two languages $L, K \subseteq \Sigma^\star$, next we present a necessary and sufficient condition to determine whether there exists an integer $N \in \mathcal{N}$ such that $L^R \subseteq K^R\Sigma^{\leq N}$ in terms of the graphical structure of SM's recognizing the languages L^R, K^R.

Consider R, the state set of $P\Box' V$. Let $R^\star$ denote the set of all finite sequences of states belonging to R. Consider $p \in R^\star$ such that $p = (r_1 r_2 \ldots r_i \ldots r_n) \in R^\star$, where $r_i \in R$ for each $1 \leq i \leq n$ and $n \in \mathcal{N}$. Then p is said to be a *path* starting at r_1 and ending at r_n in $P\Box V$, if there exist a string $s_p \in \Sigma^\star, s_p = \sigma_1\sigma_2 \ldots \sigma_i \ldots \sigma_{n-1}$, where $\sigma_i \in \Sigma$ for each $1 \leq i \leq n-1$, such that $\rho((\sigma_1 \ldots \sigma_{i-1}), r_1) = r_i$ for each $1 < i \leq n$. $s_p \in \Sigma^\star$ as described above is called the string corresponding to path p. Thus given a path p in $P\Box V$, there exists at least one string $s_p \in \Sigma^\star$ corresponding to p. A state $r \in R$ is said to be a *path-state* of the path p if $r = r_i$ for some $1 \leq i \leq n$. p is said to be a *loop-path* if there exist i, j with $1 \leq i < j \leq n$ such that $r_i = r_j$; in which case the portion $r_i \ldots r_j$ of p is called the *loop-portion* of p. p is said to be a *loopfree-path* if p is not a loop-path.

Theorem 4.12 Let $L^R, K^R \subseteq \Sigma^\star$ be the languages recognized by the SM's P, V respectively as described above. Then there exists an integer $N \in \mathcal{N}$ such that $L^R \subseteq K^R\Sigma^{\leq N}$ if and only if the following hold in the SM $P\Box' V$:

C1 For each $r_m \in R_m$ and for every path p in $P\Box' V$ that starts at r_0 and ends at r_m, there exists a path-state $r = (x, q) \in X \times Q$ of p such that $q \in Q_m$.

C2 For each $r = (x, q) \in X \times Q_m$ and each $r_m \in R_m$, if a path p in $P\Box' V$ that starts at r and ends at r_m has none of its path-states in $X \times Q_m$ (other than the one at which it starts), then p is a loop-free path.

Proof: Assume that there exists an integer $N \in \mathcal{N}$ such that $L^R \subseteq K^R\Sigma^{\leq N}$; then we first show that C1 holds.

Fix a path p in $P\Box' V$ such that p starts at r_0 and ends at $r_m \in R_m$. Then there exists a string $s_p \in L_m(P\Box' V)$ such that $\rho(s_p, r_0) = r_m$. Since $L_m(P\Box' V) = L_m(P) = L^R$ (Lemma 4.9 and definition of P), $s_p \in L^R$. Thus it follows from the assumption that $s_p \in K^R\Sigma^{\leq N}$, i.e. there exist $u_{s_p} \in K^R$ and $v_{s_p} \in \Sigma^{\leq N}$ such that $s_p = u_{s_p}v_{s_p}$. Consider the path-state $r = (x, q) = \rho(u_{s_p}, r_0)$ of p. Since $u_{s_p} \in K^R$, the state q reached by accepting u_{s_p} in V belongs to Q_m, i.e. $r = (x, q) \in X \times Q_m$.

Next we show that C2 holds. Fix a path p in $P\Box' V$ such that p starts at $r = (x, q) \in X \times Q_m$ and ends at $r_m \in R_m$ and none of the path-states of p other than the first one are in $X \times Q_m$. Assume for contradiction that C2 is false, i.e. p is a loop-path. Consider the string $s \in L(P\Box' V)$ such that $\rho(s, r_0) = r = (x, q)$. Since $q \in Q_m$, $s \in L_m(V) = K^R$. Let $t_p = u_p v_p w_p \in \Sigma^\star$ be a string corresponding to the path p, where v_p represents the string corresponding to the loop-portion of p. Then $st_p = su_p v_p w_p \in L_m(P\Box' V) = L^R$ (since $\rho(st_p, r_0) = r_m \in R_m$). Hence the string $tu_p(v_p)^{N+1}w_p \in L^R$. Then there exists no prefix $s' \in K^R$ of the string $su_p(v_p)^{N+1}w_p$ such that $\Pi_{|s'|}(su_p(v_p)^{N+1}w_p) \in \Sigma^{\leq N}$, which contradicts the fact that $L^R \subseteq K^R\Sigma^{\leq N}$. This completes the proof of the fact that C1 and C2 are necessary conditions for an integer $N \in \mathcal{N}$ to exist such that $L^R \subseteq K^R\Sigma^{\leq N}$. It remains to show that C1 and C2 are sufficient conditions also.

Assume then that C1 and C2 hold for SM $P\Box'V$. Since C2 holds, any path p in $P\Box'V$, that starts at $r = (x,q) \in X \times Q_m$ and ends at $r_m \in R_m$ with none of its path-states (other than the first one) in $X \times Q_m$, is a loopfree-path. Let $\mathcal{P}$ denote the collection of all such paths (paths that satisfy condition C2). Define $N \stackrel{\text{def}}{=} \max_{p\in\mathcal{P}} |p|$, where $|p|$ denotes the length of path p. Then we will show that $L^R \subseteq K^R\Sigma^{\leq N}$. Note that since C2 holds, all the paths $p \in \mathcal{P}$ are loopfree-paths, hence the maximum in the definition of N exists. In order to show that $L^R \subseteq K^R\Sigma^{\leq N}$, pick $s \in L^R$. Then $s \in L_m(P\Box'V)$. Let $\rho(s, r_0) = r_m \in R_m$. Consider the path p_s in $P\Box'V$ corresponding to string s. Since $P\Box'V$ is deterministic, p_s is unique. Also, p_s starts at r_0 and ends at $r_m \in R_m$. Hence by C1, there exists a path-state $r = (x,q)$ of p_s such that $r = (x,q) \in X \times Q_m$. Let r' be the last such path-state of p_s, i.e. $r' \in X \times Q_m$ and all the path-states of p_s that follow r' do not belong to $X \times Q_m$. Let the portion of p_s that starts at r' and ends at r_m be denoted by p'; then from C2 p' is a loopfree-path, also $p' \in \mathcal{P}$. It follows from the definition of N that $|p'| \leq N$. Let $u' \in \Sigma^*$ be the prefix of s such that $\rho(u', r_0) = r'$, then $u' \in K^R$ (since $r' \in X \times Q_m$), and $\Pi_{|u'|}(s) \in \Sigma^{\leq N}$. Thus $s \in K^R\Sigma^{\leq N}$. This completes the proof of Theorem 4.12. □

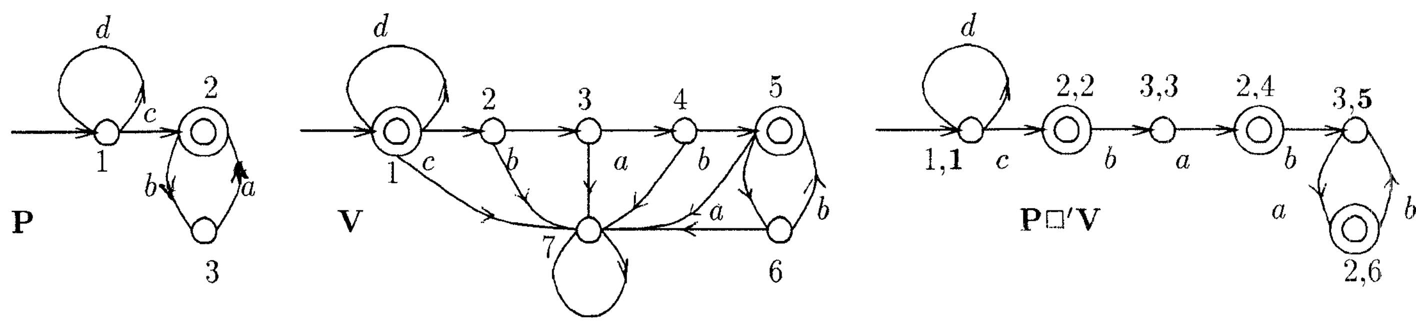
Remark 4.13 The conditions C1 and C2 can be tested in $P\Box'V$ in the following manner:

1. Consider the state set R of $P\Box'V$ and remove all the states $r = (x,q) \in R$ (and the transitions entering or leaving these states) for which $q \in Q_m$. Then for C1 to hold, there must not exist any path connecting r_0 to any $r_m \in R_m$ in the machine obtained by removing the above states. Thus C1 can be verified by performing a *single* reachability test on the reduced machine as described above.

2. Next fix a state $r = (x,q) \in R$ with $q \in Q_m$ and remove from $P\Box'V$ all the other states $r' = (x', q') \in R$ (and the transitions entering or leaving these states) having $q' \in Q_m$. Then for C2 to hold for this state r, any path connecting r to any $r_m \in R_m$ in the machine obtained by removing the above states must be acyclic. Repeat the above for every state $r'' = (x'', q'') \in R$ with $q'' \in Q_m$ and test for acyclicity. Since for each such state acyclicity can be tested by computing its reachability set, testing C2 requires at most $|P\Box'V|$ rechability tests to be performed, where $|P\Box'V|$ denotes the number of states in $P\Box'V$.

Thus it follows from above that testing C1 and C2 requires at most $|P\Box'V|+1$ reachability tests to be performed. Since the reachability set can be computed in $O(|P\Box'V|)$ time, the complexity of testing C1 and C2 is of order $O(|P\Box'V|^2)$. Let $m, n \in \mathcal{N}$ be the number of states in the minimal SM's recognizing L, K respectively, then the number of states in SM's P, V recognizing L^R, K^R respectively is $2^m, 2^n$ respectively (reversal operation requires nondeterministic to deterministic conversion of SM's). Hence the computational complexity of testing ℓ-stability of L with respect to K is $O(2^{2(m+n)})$.

Example 4.14 Consider the languages $L, K_1 \subseteq \{a,b,c,d\}^*$ as in Example 4.5: $L = (ab)^*cd^*$ and $K_1 = d^* + bab(ab)^*cd^*$. Recognizers for L and K_1 are shown in Figure 3. Then $L^R = d^*c(ba)^*$ and $K_1^R = d^* + d^*c(ba)^*bab = d^* + d^*cbab(ab)^*$. Let P, V be state machines such that $L_m(P) = L^R, L_m(V) = K_1^R$, and $L(P) = \overline{L_m(P)}, L(V) = \Sigma^\star$ respectively as in Theorem 4.12. Construct $P\Box' V$ as described above. Recognizers for $P, V, P\Box' V$ are shown in Figure 5. in Figure 5 A state (x, q) in the state set R of $P\Box' V$ is marked if and only if the state



for clarity, transitions leading to "dump" state 7 are not labeled

$L_m(P) = L^R = d^\star c(ba)^\star$ $\quad$ $L_m(V) = K_1^R = d^\star + d^\star cbab(ab)^\star$ $\quad$ $L_m(P\Box' V) = d^\star c(ba)^\star$

Figure 5: Diagram illustrating Example 4.14

$x \in X_m$. Since state $2 \in X$ is the only state marked in P, the states $(2,2), (2,4)$ and $(2,6)$ are the marked states in $P\Box' V$. We now check whether conditions C1 and C2 of Theorem 4.12 hold. Consider any path in $P\Box' V$ starting from the initial state $(1,1)$ and ending at one of the marked states $(2,2)$ or $(2,4)$ or $(2,6)$. Then this path obviously visits the state $(1,1)$. Since $(1,1) \in X \times Q_m$ (state 1 is marked in V), condition C1 holds. In order to show that C2 holds consider any path in $P\Box' V$ starting at the initial state $(1,1)$ and ending at one of the marked states either $(2,2)$ or $(2,4)$ or $(2,6)$. If the path ends at $(2,2)$, then the last state (x,q) such that $q \in Q_m$ visited along this path is $(1,1)$. Consider the path segment between $(1,1)$ and $(2,2)$; it is loop-free. If the path ends at $(2,4)$, then the last state (x,q) with $q \in Q_m$ visited along this path is again $(1,1)$, and again the path segment between $(1,1)$ and $(2,4)$ is loop-free. Finally, if the path ends at $(2,6)$, then the last state (x,q) with $q \in Q_m$ visited is $(3,5)$, and the path segment between $(3,5)$ and $(2,6)$ is again loop-free. Thus condition C2 also holds. It then follows from Theorem 4.12 that L is ℓ-stable with respect to K_1 as expected (refer to discussion in Example 4.5.

Corollary 4.15 Consider two regular languages $L, K \subseteq \Sigma^\star$. Let $m, n \in \mathcal{N}$ be the number of states in the minimal SM's recognizing L, K respectively. L is ℓ-stable with respect to K, if and only if $L \subseteq \Sigma^{\le 2^{m+n}} K$.

Proof: For proving the "if" part we need to show that $L \subseteq \Sigma^{\le 2^{m+n}} K$ implies L is ℓ-stable with respect to K. This is trivially true: set $N = 2^{m+n}$ in the definition of ℓ-stability. In order to prove the "only if" part we need to show that if L ℓ-stable with respect to K, then $L \subseteq \Sigma^{\le 2^{m+n}} K$. Since L is ℓ-stable with respect to K, it follows from Corollary 4.10 that there exists $N \in \mathcal{N}$ such that $L^R \subseteq K^R \Sigma^{\le N}$. Let P, V be machines recognizing L^R, K^R respectively as in Theorem 4.12. It then follows from Theorem 4.12 that $N \le |P\Box' V|$ (refer to the second part of the proof of Theorem 4.12 where N is defined to be $N \stackrel{\text{def}}{=} \max_{p\in\mathcal{P}} |p|$; since each $p \in \mathcal{P}$ is loop-free, $|p| \leq |P\Box' V|$, hence $N \leq |P\Box' V|$). Since the number of states in SM's recognizing L, K is m, n respectively, the number of states in P, V is $2^m, 2^n$ respectively (reversal operation requires nondeterministic to deterministic conversion of SM's [1]). Thus it follows that $N \leq |P\Box' V| = (2^m)(2^n) = 2^{m+n}$. □

Remark 4.16 Thus ℓ-stability of a given language L with respect to another language K can also be determined by testing whether $L \subseteq \Sigma^{\leq 2^{m+n}} K$, where $m, n \in \mathcal{N}$ are the numbers of states present in SM's recognizing L, K respectively.

Next we consider the problem of testing ℓ-stabilizability of a given language $L \subseteq \Sigma^*$ with respect to another language $K \subseteq \Sigma^*$. Let P, V be the SM's recognizing language L, K respectively. The supervisor that disables all the controllable transitions of P (treated as a plant) is called the *maximally restrictive supervisor*. The behavior of P under the maximally restrictive control is given by $L \cap \Sigma_u^*$. Note that since $L \cap \Sigma_u^*$ is the closed loop behavior under the control of the maximally restrictive supervisor, then any nonempty controllable sublanguage $H \subseteq L$, $L \cap \Sigma_u^* \subseteq H$. Also, note that $L \cap \Sigma_u^*$ is controllable, for $(\overline{L \cap \Sigma_u^*})\Sigma_u \cap L(P) = \overline{L \cap \Sigma_u^*}$.

Theorem 4.17 L is ℓ-stabilizable with respect to K if and only if $L \cap \Sigma_u^*$ is nonempty and ℓ-stable with respect to K.

Proof: Assume that L is ℓ-stabilizable with respect to K. Then there exists $N \in \mathcal{N}$ and a nonempty controllable sublanguage $H \subseteq L$ such that $H \subseteq \Sigma^{\leq N} K$. Note that $L \cap \Sigma_u^* \subseteq H$ (by definition of maximally restrictive control). Hence $L \cap \Sigma_u^* \subseteq \Sigma^{\leq N} K$. Thus $L \cap \Sigma_u^*$ is ℓ-stable with respect to K.

Next assume that $L \cap \Sigma_u^*$ is nonempty and ℓ-stable with respect to K. Since $L \cap \Sigma_u^*$ is controllable, it follows that L is ℓ-stabilizable with respect to K. □

Remark 4.18 Thus ℓ-stabilizability of a given language L with respect to another language K can be determined by testing whether $L \cap \Sigma_u^*$ is nonempty and ℓ-stable with respect to K.

As stated in Remark 4.13, the algorithm for testing ℓ-stability of L with respect K is of computational complexity that is exponential in the number of states present in SM's recognizing L and K. Hence so is the complexity of the algorithm that tests the ℓ-stabilizability of L with respect to K. Next we present a sufficient condition for ℓ-stability of L with respect to K that can be tested in polynomial time. Let $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$ and $V \stackrel{\text{def}}{=} (Q, \Sigma, \delta, q_0, Q_m)$ be two SM's recognizing L and K respectively. Define the following subset of states $X_S \subseteq X$:

$$X_S = \{x \in X \mid L_m(P, x) \subseteq K\}$$

where $L_m(P, x)$ is the language recognized by P assuming its initial state to be $x \in X$.

Proposition 4.19 Consider SM's P, V as defined above. If $x_0 \in \Omega(X_S)$, then L is ℓ-stable with respect to K.

Proof: Define $N \stackrel{\text{def}}{=} |X - X_S|$; then to prove ℓ-stability of L with respect to K, we need to show that $L \subseteq \Sigma^{\leq N} K$. Consider $s \in L$. If $|s| \leq N$, then clearly $s \in \Sigma^{\leq N} K$. So let $s \in L$ be such that $|s| > N$. Then it follows from the definition of region of strong attraction that there exists a prefix $u_s \in \Sigma^*$, $|u_s| \leq N$, of s such that $\alpha(u_s, x_0) \in X_S$. Also, by the definition of X_S, $\Pi_{|u_s|}(s) \in K$, which shows that $s \in \Sigma^{\leq N} K$. □

Thus if x_0 is strongly attractable to a state in X_S, then P after starting from x_0 reaches a state in X_S in at most $|X - X_S|$ transitions, and then onwards follows a string in K. The following algorithm checks the sufficient condition of ℓ-stability of Proposition 4.19:

Algorithm 4.20

1. Determine the subset of states $X_S \subseteq X$ defined above.
2. Compute $\Omega(X_S)$ using Algorithm A.1.
3. If $x_0 \in \Omega(X_S)$, then L is ℓ-stable with respect to K.

Let P, V be the minimal SM's recognizing L, K respectively and let $m, n \in \mathcal{N}$ be the number of states in P, V respectively. Then step 1 of Algorithm 4.20 can be determined in $O(m^2 n)$ time, and step 2 and 3 can both be determined in $O(m)$ time (refer to Theorem A.2). Hence the computational complexity of Algorithm 4.20 is $O(m^2 n)$ which is polynomial in m, n. Note that Algorithm 4.20 tests only for the sufficiency condition of ℓ-stability. Hence if the condition in step 3 of Algorithm 4.20 is not satisfied, ℓ-stability of L with respect to K is determined by testing conditions C1 and C2 of Theorem 4.12 as described in Remark 4.13. Next we present a sufficient condition for ℓ-stabilizability of L with respect to K, which can also be tested in polynomial time.

Proposition 4.21 Consider the SM's P, V. Let $X'_S \stackrel{\text{def}}{=} \{x \in X \mid L_m(P, x) \cap \Sigma_u^* \subseteq K\}$. If $x_0 \in \Lambda(X'_S)$, then L is ℓ-stabilizable with respect to K.

Proof: Similar to the proof of Proposition 4.19. □

The following algorithm can be used for testing the sufficient condition of ℓ-stabilizability of Proposition 4.21:

Algorithm 4.22

1. Compute $X'_S \subseteq X$.
2. Compute $\Lambda(X'_S)$ using the modification to Algorithm A.1 described in Remark A.3.
3. If $x_0 \in \Lambda(X'_S)$, then L is ℓ-stabilizable with respect to K.

The computational complexity of Algorithm 4.22 is also $O(m^2 n)$, where m, n is the number of states in P, V respectively.

5 Weakly Stabilizing Supervisors

In the previous section we showed that given a plant P with physical behavior $L \subseteq \Sigma^\star$ and desired eventual behavior $K \subseteq \Sigma^\star$, it can be verified whether or not L is ℓ-stable or ℓ-stabilizable with respect to K. In case L is ℓ-stable with respect to K, the eventual behavior of P is contained in K; hence no supervisor is needed. If L is not ℓ-stable but is ℓ-stabilizable with respect to K, then a supervisor must be constructed to insure that the eventual closed loop behavior of the system is a sublanguage of K. The ℓ-stabilizability of L guarantees the existence of a stabilizing supervisor, but a minimally restrictive stabilizing supervisor need not in general exist. This is evident from the following proposition:

Proposition 5.1 ℓ-stability is not preserved under union.

Proof: We show by the following example that ℓ-stabilizability is not preserved under union. Let $\Sigma = \Sigma_c = \{a, b\}$, $L = a^\star b^\star$ denote the plant behavior and $K = b^\star$ denote the desired eventual behavior. Then there does not exist any integer $N \in \mathcal{N}$ such that $L \subseteq \Sigma^{\leq N} K$, i.e. L is not ℓ-stable with respect to K.

Next consider the following family of sublanguages $\{L_i\}_{i \in \mathcal{N}}$ of L with $L_i = a^i b^\star$ for each $i \in \mathcal{N}$. Then it is clear that for each $i \in \mathcal{N}$, L_i is controllable (since $\Sigma_c = \Sigma$) and also ℓ-stable (since $L_i \subseteq \Sigma^{\leq i} K$) sublanguage of L. But $\bigcup_{i \in \mathcal{N}} L_i = L$ is not ℓ-stable with respect to K; thus showing that ℓ-stability is not preserved under union. □

The implication of Proposition 5.1 is that if the plant behavior L is not ℓ-stable with respect to the desired eventual behavior K, then the minimally restrictive stabilizing supervisor, which will restrict the plant behavior to the supremal ℓ-stable sublanguage of L, cannot in general be constructed. Next we define a weaker notion of language stability that we call *weak ℓ-stability* which is preserved under union so that the minimally restrictive stabilizing supervisor can be constructed.

Definition 5.2 A language $L \subseteq \Sigma^\star$ is said to be *weakly ℓ-stable* with respect to another language $K \subseteq \Sigma^\star$ if $L \subseteq \Sigma^\star K$. If there exists a nonempty controllable sublanguage $H \subseteq L$ such that H is weakly ℓ-stable with respect to K, then L is said to be *weakly ℓ-stabilizable* with respect to K.

Thus if L is weakly ℓ-stable with respect to K, then every string in L after removing a prefix from it, matches some string in K. Notice that here no uniform bound on the size of prefix to be removed from a string in L is assumed.

Remark 5.3 Since $\Sigma^{\leq N} \subseteq \Sigma^\star$ for any $N \in \mathcal{N}$, it follows that ℓ-stability implies weak ℓ-stability. However, the converse does not hold in general. Consider for example the languages $L = a^\star b^\star$ and $K = b^\star$ defined over the event set $\Sigma = \{a, b\}$. Then as stated in the proof of Proposition 5.1, L is not ℓ-stable with respect to K. But clearly L is weakly ℓ-stable with respect to K, for $a^\star b^\star \subseteq \Sigma^\star b^\star$.

The following result analogous to that stated in Theorem 4.17 holds also for weak ℓ-stabilizability.

Theorem 5.4 L is weakly ℓ-stabilizable with respect to K if and only if $L \cap \Sigma_u^\star$ is nonempty and weakly ℓ-stable respect to K.

Proof: Similar to the proof of Theorem 4.17. □

Next we discuss how to verify weak ℓ-stability and weak ℓ-stabilizability of a given plant behavior with respect to its desired eventual behavior. Let $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m), V \stackrel{\text{def}}{=} (Q, \Sigma, \delta, q_0, Q_m)$ be the minimal SM's recognizing the languages L, K respectively. Assuming that the languages L, K are regular, let m, n be the number of states in P, V respectively. A SM that recognizes $\Sigma^\star K$ is constructed by first adding the self-loop corresponding to $\Sigma^\star$ at the initial state of V and then converting it to a deterministic SM. Let this SM be denoted by V'; then the number of states in V' is 2^n.

Remark 5.5 The weak ℓ-stability of L with respect to K can be verified by determining whether $L_m(P) \subseteq L_m(V')$. Since the number of states in P, V' is $m, 2^n$ respectively, the computational complexity of verifying weak ℓ-stability of L with respect to K is $O(m2^n)$. It also follows, in view of Theorem 5.4, that the computational complexity of testing weak ℓ-stabilizability of L with respect to K is again $O(m2^n)$.

Since ℓ-stability (ℓ-stabilizability) implies weak ℓ-stability (weak ℓ-stabilizability), the condition in Proposition 4.19 (Proposition 4.21) is sufficient for weak ℓ-stability (weak ℓ-stabilizability). Thus Algorithm 4.20 (Algorithm 4.22) can be employed to test this sufficient condition for weak ℓ-stability (weak ℓ-stabilizability), the computational complexity of which is polynomial in m, n.

Next we prove that weak ℓ-stability is preserved under union, i.e. the supremal weakly ℓ-stable sublanguage of a given language exists.

Proposition 5.6 The supremal weakly ℓ-stable sublanguage of a given language exists and is unique.

Proof: Let L, K denote the plant, desired eventual behavior respectively. Let Λ be an indexing set such that the family of weakly ℓ-stable sublanguages of L is given by $\{L_\lambda\}_{\lambda \in \Lambda}$, i.e. L_λ is weakly ℓ-stable sublanguage of L for each $\lambda \in \Lambda$. Such a family is nonempty because $\emptyset$ is weakly ℓ-stable sublanguage of L. Consider the language $H \stackrel{\text{def}}{=} \bigcup_{\lambda \in \Lambda} L_\lambda$; then clearly $H \subseteq L$ and H is weakly ℓ-stable. The last assertion follows from the fact that $L_\lambda \subseteq \Sigma^\star K$ for each $\lambda \in \Lambda$ which implies that $\bigcup_{\lambda \in \Lambda} L_\lambda = H \subseteq \Sigma^\star K$. This completes the proof of Proposition 5.6. □

Corollary 5.7 The supremal controllable and weakly ℓ-stable sublanguage of a given language exists and is unique.

Proof: Follows from Proposition 5.6 and the fact that controllability is preserved under union [21, 20]. □

We proved the existence and uniqueness of the supremal controllable and weakly ℓ-stable sublanguage of a given language. Next we present a closed form expression for it. We use the notation $H^\uparrow$ to denote the supremal controllable sublanguage of a given language $H \subseteq \Sigma^\star$ [20, 1, 10].

Theorem 5.8 Let $L, K \subseteq \Sigma^*$ denote the plant, desired eventual behavior respectively. Then the supremal controllable and weakly ℓ-stable sublanguage of L is given by $(L \cap \Sigma^* K)^\uparrow$.

Proof: Let $H \subseteq \Sigma^*$ denote the supremal controllable and weakly ℓ-stable sublanguage of L with respect to K. Then we need to show that $H = (L \cap \Sigma^* K)^\uparrow$.

First we show that $(L \cap \Sigma^* K)^\uparrow \subseteq H$. Since H is the supremal controllable and weakly ℓ-stable sublanguage of L, it suffices to show that $(L \cap \Sigma^* K)^\uparrow$ is a controllable and weakly ℓ-stable sublanguage of L. By its definition, $(L \cap \Sigma^* K)^\uparrow$ is a controllable sublanguage of L. Also, since $(L \cap \Sigma^* K)^\uparrow \subseteq L \cap \Sigma^* K \subseteq \Sigma^* K$, it follows that $(L \cap \Sigma^* K)^\uparrow$ is weakly ℓ-stable with respect to K. Thus $(L \cap \Sigma^* K)^\uparrow$ is a controllable and weakly ℓ-stable sublanguage of L.

Next we prove that $H \subseteq (L \cap \Sigma^* K)^\uparrow$. Since H is weakly ℓ-stable, it follows that $H \subseteq \Sigma^* K$; also, $H \subseteq L$, hence $H \subseteq L \cap \Sigma^* K$. Note that H is controllable also. Thus H is controllable and is contained in $L \cap \Sigma^* K$. Since $(L \cap \Sigma^* K)^\uparrow$ is the supremal controllable sublanguage contained in $L \cap \Sigma^* K$, it follows that $H \subseteq (L \cap \Sigma^* K)^\uparrow$. □

Thus if L is not ℓ-stable with respect to K, but is weakly ℓ-stabilizable with respect to K, then a minimally restrictive stabilizing supervisor can be constructed so that the behavior of the closed loop system is given by $(L \cap \Sigma^* K)^\uparrow$. Note that the result of Theorem 5.8 is not surprising, as we are interested in finding the supremal sublanguage $H \subseteq L$ such that H is weakly stable, i.e. $H \subseteq \Sigma^* K$ and H is controllable. Since $H \subseteq L$ and $H \subseteq \Sigma^* K$, it follows that $H \subseteq L \cap \Sigma^* K$. Thus we are interested in finding the supremal controllable sublanguage $H \subseteq L \cap \Sigma^* K$, which obviously equals $(L \cap \Sigma^* K)^\uparrow$. This, however, offers an alternative interpretation of minimally restrictive weakly stabilizing supervisors: problem of finding the minimally restrictive weakly stabilizing supervisor for a plant with behavior L and *desired eventual behavior* K is equivalent to the problem of finding the minimally restrictive supervisor for the same plant with *desired behavior* $L \cap \Sigma^* K$. Hence techniques developed in [21, 20, 1, 11] etc. can be used to solve the problem.

6 Stability of Sequential Behavior

So far we have discussed the stability of the *finite* behavior of a DEDS. We will show how the notions of ℓ-stability and ℓ-stabilizability defined above can be easily generalized to describe the stability of *infinite* or *sequential* behaviors of DEDS's. In this section, we introduce the notion of *ω-stability* for formally describing the the notion of eventual sequential behavior.

In [19, 22, 13, 12, 23] the supervisory control problem for controlling the sequential behavior of a DEDS is studied, and conditions under which a supervisor can be constructed so that the sequential behavior of the controlled system is equal to some desired sequential behavior are obtained. As discussed above, such a control problem formulation may lead to synthesis of a very restrictive supervisor. In some cases, it might suffice to design a supervisor which would ensure that the sequential behavior of the controlled system is eventually contained in the desired sequential behavior. So we introduce the notion of the desired eventual sequential behavior and obtain conditions under which the plant's sequential behavior is eventually contained in this sequential behavior. We follow the framework of [19] for addressing the supervisory control problem of sequential behavior.

Let Σ^ω denote the set of all infinite strings of events belonging to Σ. An *infinite* or *ω-language* is a sublanguage of Σ^ω. Let $e^n \in \Sigma^\star$ denote the prefix of size n of the infinite string $e \in \Sigma^\omega$. A suitable metric can be defined on the space Σ^ω [7]. Given two infinite strings $e_1, e_2 \in \Sigma^\omega$, the distance $d(e_1, e_2)$ between the two infinite strings is defined to be:

$$d(e_1, e_2) \stackrel{\text{def}}{=} \begin{cases} 1/(n+1) & \text{if } e_1^n = e_2^n \text{ and } e_1^{n+1} \neq e_2^{n+1} \ (n \in \mathcal{N}) \\ 0 & \text{if } e_1 = e_2 \end{cases}$$

Given a language $L \subseteq \Sigma^\star$, its *limit*, denoted as L^∞, is the ω-language defined as:

$$L^\infty \stackrel{\text{def}}{=} \{e \in \Sigma^\omega \mid e^n \in L \text{ for infinitely many } n \in \mathcal{N}\}$$

We will use $t \leq s$ to denote that $t \in \Sigma^\star$ is a prefix of $s \in \Sigma^\star \cup \Sigma^\omega$. If t is a proper prefix of s, then it is written as $t < s$. Given an infinite sequence of strings $s_1 < s_2 < \ldots < s_n < \ldots$ with $s_n \in \Sigma^\star$ for each n, there exists a unique infinite string $e \in \Sigma^\omega$ such that $s_n < e$ for each n. In this case, the infinite string e is also written as $e = \lim_{n \to \infty} s_n$. Given an ω-language $\mathcal{L} \subseteq \Sigma^\omega$, its *prefix*, denoted by $pr\mathcal{L}$, is the language:

$$pr\mathcal{L} \stackrel{\text{def}}{=} \{s \in \Sigma^\star \mid \exists e \in \mathcal{L} \text{ s.t. } s < e\}$$

Note that $pr\mathcal{L} = pr\overline{\mathcal{L}}$, where $\overline{\mathcal{L}}$ denotes the topological closure[3] of $\mathcal{L}$ in the metric space (Σ^ω, d) [7]. It can be proved [7] that for a ω-language $\mathcal{L} \subseteq \Sigma^\omega$,

$$(pr\mathcal{L})^\infty = \overline{\mathcal{L}}$$

With the above preliminary notions we can address the issue of stability of the infinite behavior of a given DEDS. Let $P \equiv (X, \Sigma, \alpha, x_0, X_m)$ denote the plant. Then as defined above, $L_m(P), L(P) \subseteq \Sigma^\star$ denote its (finite) marked, generated languages respectively. The ω-language generated by P, denoted by $\mathcal{L}(P)$, is defined to be:

$$\mathcal{L}(P) \stackrel{\text{def}}{=} \{e \in (L(P))^\infty \mid \exists \text{ infinitely many } n \in \mathcal{N} \text{ s.t. } \alpha(e^n, x_o) \in X_m\} = (L_m(P))^\infty$$

Note that the ω-language $\mathcal{L}(P)$ generated by P as defined above is also the ω-language generated by P viewed as a Büchi automaton [7]. P is said to *nonblocking* if $pr\mathcal{L}(P) = L(P)$. Let $S \equiv (Y, \Sigma, \beta, y_0, Y_m)$ denote the supervisor that controls P by synchronization as defined above. Then the ω-language generated by the closed loop system $P \Box S$ is defined to be:

$$\mathcal{L}(P \Box S) \stackrel{\text{def}}{=} (L(P \Box S))^\infty \cap \mathcal{L}(P)$$

Let $\mathcal{K} \subseteq \mathcal{L}(P)$ be the desired ω-language. It is shown in [19] that a complete, nonblocking supervisor exists for achieving the desired sequential behavior if and only if $\mathcal{K}$ is *ω-controllable* with respect to P:

[3]The notation $\overline{\mathcal{L}}$ is used to denote topological closure whenever $\mathcal{L} \subseteq \Sigma^\omega$, and the notation $\overline{L}$ is used to denote the prefix closure whenever $L \subseteq \Sigma^\star$.

Definition 6.1 An ω-language $\mathcal{K} \subseteq \Sigma^\omega$ is said to be *ω-controllable* with respect to the plant P if $pr\mathcal{K}$ is controllable with respect to P, and $\mathcal{K}$ is topologically closed with respect to $\mathcal{L}(P)$; i. e.

1. $pr(\mathcal{K})\Sigma_u \cap L(P) \subseteq pr\mathcal{K}$, and
2. $\overline{\mathcal{K}} \cap \mathcal{L}(P) = \mathcal{K}$.

It is further shown in [19] that if $\mathcal{K}$ is not ω-controllable, but is topologically closed with respect to $\mathcal{L}(P)$, then the *supremal ω-controllable* sublanguage, denoted by $\mathcal{K}^\dagger$, of $\mathcal{K}$ exists[4]. Thus the construction of the *minimally restrictive supervisor* is possible. A closed form expression for the supremal ω-controllable sublanguage, as well as an efficient algorithm for computing it, is presented in [13, 12].

Next, let $\mathcal{K} \subseteq \Sigma^\omega$ represent the desired eventual sequential behavior of the plant $P \equiv (X, \Sigma, \alpha, x_0, X_m)$. The notion of ω-stability is defined as follows:

Definition 6.2 The plant sequential behavior $\mathcal{L}(P)$ is said to be *ω-stable* with respect to the desired eventual sequential behavior $\mathcal{K}$ if there exists an integer $N \in \mathcal{N}$ such that $\mathcal{L}(P) \subseteq \Sigma^{\leq N}\mathcal{K}$. $\mathcal{L}(P)$ is said to be *ω-stabilizable* with respect to $\mathcal{K}$ if there exists a nonempty ω-controllable sublanguage $\mathcal{H} \subseteq \mathcal{L}(P)$ such that $\mathcal{H}$ is ω-stable with respect to $\mathcal{K}$.

Let $e \in \Sigma^\omega$ be a infinite string and for each $n \in \mathcal{N}$, let $f_n \in \Sigma^\omega$ be such that $e = e^n f_n$. Then the *projection* operator $\Pi_n : \Sigma^\omega \to \Sigma^\omega$ $(n \in \mathcal{N})$ is defined in the following manner:

$$\Pi_n(e) = f_n$$

In other words, given a infinite string $e \in \Sigma^\omega$, its projection $\Pi_n(e)$ is obtained by deleting its prefix of size n from it. Thus if $\mathcal{L}(P)$ is ω-stable with respect to $\mathcal{K}$, then for each $e \in \mathcal{L}(P)$ there exists an integer $n_e \leq N$ such that $\Pi_{n_e}(e) \in \mathcal{K}$. In other words, each infinite string in $\mathcal{L}(P)$ after removing a prefix of size at most N matches a infinite string in $\mathcal{K}$. The ω-language $\mathcal{K}$ thus can be thought of to be representing the desired eventual sequential behavior. If $\mathcal{L}(P)$ is not ω-stable but ω-stabilizable with respect to $\mathcal{K}$, then there exists a nonempty ω-controllable sublanguage $\mathcal{H} \subseteq \mathcal{L}(P)$ which is ω-stable with respect to $\mathcal{K}$ also. Thus a nonblocking and complete [19] supervisor, that can restrict the sequential behavior of the plant to $\mathcal{H}$ which "stabilizes" to the desired eventual sequential behavior $\mathcal{K}$, can be constructed.

6.1 Tests for ω-stability and ω-stabilizability

In this subsection we show that under certain assumptions ω-stability can be tested by performing the test for ℓ-stability. First we define the notion of *complete* languages which is useful in the context of studying the stability of infinite behaviors.

[4]The notation $\mathcal{K}^\dagger$ is used to denote the supremal ω-controllable sublanguage of $\mathcal{K} \subseteq \Sigma^\omega$, and the notation $K^\uparrow$ is used to denote the supremal controllable sublanguage of $K \subseteq \Sigma^*$.

Definition 6.3 Consider a language $L \subseteq \Sigma^\star$. A string $s \in L$ is said to have an *extension* in L if there exists a $t \in L$ such that $s < t$. L is said to be *complete*[5] if for every string $s \in L$, there exists an extension in L.

Note that a language is complete if and only if a trim SM recognizing it is *live* (has at least one transition defined at each of its states) [13]. First we show that ℓ-stability of a given language with respect to another implies ω-stability of the limit of the given language with respect to the limit of the other.

Theorem 6.4 Consider $L, K \subseteq \Sigma^\star$. If L is ℓ-stable with respect to K, then L^∞ is ω-stable with respect to K^∞.

We prove the following lemma before proving the result of Theorem 6.4.

Lemma 6.5 Consider $L \subseteq \Sigma^\star$. Then for any $N \in \mathcal{N}$, $(\Sigma^{\leq N} L)^\infty = \Sigma^{\leq N} L^\infty$.

Proof: First we show that $\Sigma^{\leq N} L^\infty \subseteq (\Sigma^{\leq N} L)^\infty$. Pick $e \in \Sigma^{\leq N} L^\infty$. Then e can be written as $e = e^n f$, where $n \leq N$ and $f \in L^\infty$. Thus there exist infinitely many $m \in \mathcal{N}$ such that $f^m \in L$. Then the strings $e^n f^m \in \Sigma^{\leq N} L$ for each $m \in \mathcal{N}$. Hence $\lim_{m \to \infty} e^n f^m \in (\Sigma^{\leq N} L)^\infty$. Also, since $e^n f^1 < e^n f^2 < \ldots < e^n f^m < \ldots < e$, it follows that $\lim_{m \to \infty} e^n f^m = e$: which shows that $e \in (\Sigma^{\leq N} L)^\infty$.

Next we show that $(\Sigma^{\leq N} L)^\infty \subseteq \Sigma^{\leq N} L^\infty$. Pick $e \in (\Sigma^{\leq N} L)^\infty$. Then there exist infinitely many $n \in \mathcal{N}$ such that $e^n \in \Sigma^{\leq N} L$. Thus each e^n can be written as $e^n = u_n v_n$, where $u_n \in \Sigma^{\leq N}$ and $v_n \in L$. Since the set $\Sigma^{\leq N}$ is finite, it follows that there exists at least one integer $n_0 \in \mathcal{N}$ such that $u_{n_0} = u_n$ for infinitely many n. Let $\{n_k\}_{k \in \mathcal{N}}$ be a subsequence such that $u_{n_1} = u_{n_2} = \ldots = u_{n_k} = \ldots = u_{n_0}$. Then $e^{n_k} = u_{n_0} v_{n_k}$ for each $k \in \mathcal{N}$. Hence $e = \lim_{k \to \infty} e^{n_k} = u_{n_0} \lim_{k \to \infty} v_{n_k}$. Since $u_{n_0} \in \Sigma^{\leq N}$ and $v_{n_k} \in L$ for each $k \in \mathcal{N}$, it follows that $e \in \Sigma^{\leq N} L^\infty$. □

Proof (of Theorem 6.4): Since L is ℓ-stable with respect to K, there exists an integer $N \in \mathcal{N}$ such that $L \subseteq \Sigma^{\leq N} K$. Hence, by taking limits on both sides of the last inclusion, we obtain $L^\infty \subseteq (\Sigma^{\leq N} K)^\infty$. It then follows from Lemma 6.5 that $L^\infty \subseteq \Sigma^{\leq N} K^\infty$; which shows that L^∞ is ω-stable with respect to K^∞. □

Example 6.6 Consider languages L, K_1 of Example 4.5. Then $L^\infty = ((ab)^\star cd^\star)^\omega = (ab)^\star cd^\omega$ and $(K_1)^\infty = (d^\star + bab(ab)^\star cd^\star)^\infty = d^\omega + bab(ab)^\star cd^\omega$. Using arguments similar to those in Example 4.5 it can be easily verified that $L^\infty \subseteq \Sigma^{\leq 3}(K_1)^\infty$. This shows as expected from the result of Theorem 6.4 that L^∞ is ω-stable with respect to $(K_1)^\infty$. However, the converse of Theorem 6.4 does not hold in general. Consider for example languages $L, K \subseteq \{a, b\}^\star$: $L = (ab)^\star$ and $K = (ba)^s tar$. Then $L^\infty = (ab)^\omega$ and $K^\infty = (ba)^\omega = (ab)^\omega$. Since $(ab)^\omega = a(ba)^\omega$, it is obvious that L^∞ is ω-stable with respect to K^∞ ($L^\infty = aK^\infty$). It can also be easily checked that L is not ℓ-stable with respect to K: a string in L ends with the symbol b, whereas a string in K ends with the symbol a. Thus given any string in L, no suffix of it matches any string in K.

[5]Completeness is also defined to be a property of supervisors; here we define it to be a property of languages. The two definitions are unrelated and not to be confused with.

Next we prove that under certain assumptions the converse of Theorem 6.4 holds.

Theorem 6.7 Consider $L, K \subseteq \Sigma^\star$. Assume that L is complete and K is prefix closed. Then ω-stability of L^∞ with respect to K^∞ implies ℓ-stability of L with respect to K.

Before proving the result of Theorem 6.7, we prove the following lemma.

Lemma 6.8 Consider two languages $L_1, L_2 \subseteq \Sigma^\star$. Assume that L_1 is complete and L_2 is closed. Then $(L_1)^\infty \subseteq (L_2)^\infty$ if and only if $L_1 \subseteq L_2$.

Proof: It is clear that $L_1 \subseteq L_2$ implies $L_1^\infty \subseteq L_2^\infty$. Hence it suffices to show that if $(L_1)^\infty \subseteq (L_2)^\infty$, then $L_1 \subseteq L_2$. Pick $s \in L_1$. Since L_1 is complete, there exists a sequence of strings $s_1 < s_2 < \ldots < s_n < \ldots$ such that $s_n \in L_1$ for each $n \in \mathcal{N}$ and $s < s_1$. Let $e = \lim_{n\to\infty} s_n$; then $e \in (L_1)^\infty$. It then follows from the assumption that $e \in (L_2)^\infty$. Hence there exist infinitely many $n \in \mathcal{N}$ such that $e^n \in L_2$. Pick $m \in \mathcal{N}$ such that $s < e^m$. Since $e^m \in L_2$ and L_2 is closed, it follows that $s \in L_2$. □

Proof (of Theorem 6.7): Assume that L^∞ is ω-stable with respect to K^∞. Then there exists an integer $N \in \mathcal{N}$ such that $L^\infty \subseteq \Sigma^{\leq N} K^\infty$. Thus it follows from Lemma 6.5 that $L^\infty \subseteq (\Sigma^{\leq N} K)^\infty$. Note that since $\Sigma^{\leq N}$ is closed, and prefix closure is preserved under concatenation of languages $\Sigma^{\leq N} K$ is a closed language (by assumption K is closed). Since L is complete (by assumption) and $\Sigma^{\leq N} K$ is closed, we obtain from Lemma 6.8 that $L^\infty \subseteq (\Sigma^{\leq N} K)^\infty$ if and only if $L \subseteq \Sigma^{\leq N} K$. □

Example 6.9 Consider the languages $L = (ab)^\star$ and $K = (ba)^\star$ of Example 6.6. It was noted in Example 6.6 that L^∞ is ω-stable with respect to K^∞, however, L is not ℓ-stable with respect to K. The reason is that although L is a complete language, K is not prefix closed. Let us replace K by its prefix closure, i.e. consider $K' = \overline{K} = \overline{(ba)^\star} = (ba)^\star + b(ab)^\star$. Then clearly L is ℓ-stable with respect to K' ($L = (ab)^\star = ab + ab(ab)^\star \subseteq \Sigma^{\leq 2} K'$).

The results of Theorem 6.4 and Theorem 6.7 can be combined to arrive at a test for ω-stability based on the test for ℓ-stability (Theorem 4.12).

Theorem 6.10 Let $\mathcal{L}(P) = (L_m(P))^\infty \subseteq \Sigma^\omega$ denote the plant ω-behavior and $\mathcal{K} \subseteq \Sigma^\omega$ denote the desired eventual behavior. If P is live and $\mathcal{K}$ is topologically closed, then $\mathcal{L}(P)$ is ω-stable with respect to $\mathcal{K}$ if and only if $L_m(P)$ is ℓ-stabe with respect to $pr\mathcal{K}$.

Proof: Since P is live, $L_m(P)$ is complete. Also, since $\mathcal{K}$ is topologically closed, $\mathcal{K} = \overline{\mathcal{K}} = (pr\mathcal{K})^\infty$. Thus $\mathcal{L}(P)$ is limit of the complete language $L_m(P)$ and $\mathcal{K}$ is limit of the prefix closed language $pr\mathcal{K}$. Hence it follows from Theorem 6.4 and Theorem 6.7 that $\mathcal{L}(P)$ is ω-stable with respect to $\mathcal{K}$ if and only if $L_m(P)$ is ℓ-stable with respect to $pr\mathcal{K}$. □

Next we relate the notion of ω-stabilizability to that of ω-stability through the following theorem.

Theorem 6.11 $\mathcal{L}(P)$ is ω-stabilizable with respect to $\mathcal{K}$ if and only if $\mathcal{L}(P) \cap \Sigma_u^\omega$ is nonempty and ω-stable with respect to $\mathcal{K}$, where $\Sigma_u^\omega = (\Sigma_u^\star)^\infty$.

Proof: We first show that $\mathcal{L}(p) \cap \Sigma_u^\omega$ is the infimal ω-controllable sublanguage of $\mathcal{L}(P)$, i.e. it is the sequential behavior of P under the control of maximally restrictive complete and nonblocking supervisor [19]. Consider the supervisor that disables all the controllable events in P. Then the behavior of the closed loop system under this control law is given by $L(P) \cap \Sigma_u^*$. Hence the sequential behavior of the closed loop system is given by $(L(P) \cap \Sigma_u^*)^\infty \cap \mathcal{L}(P) = (L(P))^\infty \cap (\Sigma_u^*)^\infty \cap \mathcal{L}(P) = \mathcal{L}(P) \cap \Sigma_u^\omega$, where the first equality follows from the fact that $L(P), \Sigma_u^*$ are both closed languages and the second equality follows from the fact that $\mathcal{L}(P) \subseteq (L(P))^\infty$ and $(\Sigma_u^*)^\infty = \Sigma_u^\omega$. Note that the supervisor that disables all the controllable transitions in P is complete (it never disables any uncontrollable transition) and nonblocking (since $pr(\mathcal{L}(P) \cap \Sigma_u^\omega) = L(P) \cap \Sigma_u^*$). Hence $\mathcal{L}(P) \cap \Sigma_u^\omega$ is ω-controllable [19]. Since it is the sequential behavior under the maximally restrictive complete and nonblocking control law, if $\mathcal{H} \subseteq \mathcal{L}(P)$ is any ω-controllable sublanguage of $\mathcal{L}(P)$, then $\mathcal{L}(P) \cap \Sigma_u^\omega \subseteq \mathcal{H}$.

Assume then that $\mathcal{L}(P)$ is ω-stabilizable with respect to $\mathcal{K}$. Then by the definition of ω-stabilizability, there exists a nonempty ω-controllable sublanguage $\mathcal{H} \subseteq \mathcal{L}(P)$ and an integer $N \in \mathcal{N}$ such that $\mathcal{H} \subseteq \Sigma^{\leq N}\mathcal{K}$. Since $\mathcal{L}(P) \cap \Sigma_u^\omega \subseteq \mathcal{H}$, it follows that $\mathcal{L}(P) \cap \Sigma_u^\omega \subseteq \Sigma^{\leq N}\mathcal{K}$; which shows that $\mathcal{L}(P) \cap \Sigma_u^\omega$ is ω-stable with respect to $\mathcal{K}$.

Assume next that $\mathcal{L}(P) \cap \Sigma_u^\omega$ is nonempty and ω-stable with respect to $\mathcal{K}$. Since $\mathcal{L}(P) \cap \Sigma_u^\omega \subseteq \mathcal{L}(P)$ and is ω-controllable (proved above), it follows that $\mathcal{L}(P)$ is ω-stabilizable with respect to $\mathcal{K}$. □

Remark 6.12 Note that $\mathcal{L}(P) \cap \Sigma_u^\omega = (L_m(P))^\infty \cap (\Sigma_u^*)^\infty = (L_m(P) \cap \Sigma_u^*)^\infty$, where the last equality follows from the fact that Σ_u^* is prefix closed. Thus, if P is live and $\mathcal{K}$ is topologically closed, then from Theorem 6.10 and Theorem 4.17 it follows that the ω-stabilizability of $\mathcal{L}(P)$ with respect to $\mathcal{K}$ is equivalent to ℓ-stabilizability of $L_m(P)$ with respect to $pr\mathcal{K}$.

Remark 6.13 A necessary condition for ω-stability is obtained using an equivalence relation on the space Σ^ω introduced in Appendix B. It is also shown in Appendix B that if a weaker definition of ω-stability is used the necessary condition obtained in terms of the equivalence relation is also a sufficient condition.

7 Conclusion

In this paper, we have introduced the notions of stability and stabilizability of DEDS's in terms of their behavior. In many situations, since the behavior rather than the states of the system is observed directly, it is more natural to study the stability of systems in terms of their behavior. Also, in some cases, it might be desired that the eventual (rather than the whole) behavior of the system be legal, so it is necessary to define formally the notion of language stability. Earlier works concerning stability of DEDS's [18, 2, 3] are all based in terms of the states of the systems and can be viewed as a special case of the work presented here (refer to Proposition 4.4). The earlier works [18, 2, 3] on stability in terms of states assume the control to be of static feedback type; however, more general supervisors that exercise dynamic feedback have been used here for making the systems ℓ-stable.

We have shown that the problem of determining ℓ-stability (ℓ-stabilizability) of a given language with respect to another language is equivalent to another problem posed in terms of the reversal of languages (refer to Corollary 4.10) and have provided a solution to this equivalent problem (refer to Theorem 4.12 and Theorem 4.17). We have also provided an upper bound to the value of the integer N in the definition of ℓ-stability (ℓ-stabilizability) using the solution to the equivalent problem (refer to Corollary 4.10). Next we have presented a weaker notion of language stability in which no uniform upper bound on the length of the prefix to be removed from a string in a language (for it to ℓ-stable with respect to another language) exists and have provided the construction of the *minimally restrictive supervisor* [10, 21, 20, 11] to ℓ-stabilize a given language in this weaker sense of language stability.

The notion of ℓ-stability and ℓ-stabilizability is then generalized to describe the notion of stability of sequential behavior of DEDS's and the notions of ω-stability and ω-stabilizability is introduced in this context. We have introduced an equivalence relation on the space of infinite strings and have obtained a necessary condition of ω-stability in terms of this relation. A necessary and sufficient condition for ω-stability is obtained in terms of ℓ-stability, which is used to arrive at tests for ω-stability and ω-stabilizability.

References

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

[2] Y. Brave and M. Heymann. On stabilization of discrete event processes. In *IEEE Proceedings of 28th Conference on Decision and Control*, pages 2737–2742, Tampa, FL, December 1989.

[3] Y. Brave and M. Heymann. On stabilization of discrete event processes. Technical report, Department of EE, Technion-Israel Institute of Technology, Hafia 32000, Israel, 1989.

[4] Y. Brave and M. Heymann. On optimal attraction in discrete event processes. Technical Report CIS Report 9019, Department of CS, Technion-Israel Institute of Technology, Hafia 32000, Israel, 1990.

[5] H. Cho and S. I. Marcus. On supremal languages of class of sublanguages that arise in supervisor synthesis problems with partial observations. *Mathematics of Control Signals and Systems*, 2:47–69, 1989.

[6] H. Cho and S. I. Marcus. Supremal and maximal sublanguages arising in supervisor synthesis problems with partial observations. *Mathematical Systems Theory*, 22:177–211, 1989.

[7] S. Eilenberg. *Automata, Languages, and Machines: Volume A.* Academic Press, New York, NY, 1974.

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

[9] R. Kumar. *Supervisory Synthesis Techniques for Discrete Event Dynamical Systems: Transition Model Based Approach.* PhD thesis, Department of ECE, University of Texas at Austin, 1991.

[10] R. Kumar, V. K. Garg, and S. I. Marcus. Supervisory control of discrete event systems: supremal controllable and observable languages. In *Proceedings of 1989 Allerton Conference*, pages 501–510, Allerton, IL, September 1989.

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

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

[13] R. Kumar, V. K. Garg, and S. I. Marcus. On supervisory control of sequential behaviors. *IEEE Transactions on Automatic Control*, 1991. To appear.

[14] S. Lafortune and E. Chen. On the infimal closed and controllable superlanguage of a given language. *IEEE Transactions on Automatic Control*, 35(4):398–404, 1990.

[15] S. Lafortune and F. Lin. On tolerable and desirable behaviors in supervisory control of discrete event systems. *Discrete Event Dynamical System: Theory and Application*, (1):61–92, 1991.

[16] C. M. Ozveren, A. S. Willsky, and P. J. Antsaklis. Output stabilizability of discrete event dynamical systems. In *IEEE Proceedings of 28th Conference on Decision and Control*, pages 2719–2724, Tampa, FL, December 1989.

[17] C. M. Ozveren, A. S. Willsky, and P. J. Antsaklis. Tracking and restrictability in discrete event dynamical systems. *SIAM Journal of Control and Optimization*, 1990. Submitted.

[18] C. M. Ozveren, A. S. Willsky, and P. J. Antsaklis. Stability and stabilizability of discrete event dynamical systems. *Journal of ACM*, 1991. To appear.

[19] P. J. Ramadge. Some tractable supervisory control problems for discrete event systems modeled by buchi automata. *IEEE Transactions on Automatic Control*, 34(1):10–19, 1989.

[20] P. J. Ramadge and W. M. Wonham. On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, 25(3):637–659, 1987.

[21] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[22] J. G. Thistle and W. M. Wonham. On the synthesis of supervisors subject to ω-language specifications. In *Proceedings of 22nd Annual Conference on Information Sciences and Systems*, pages 440–444, Princeton, NJ, 1988.

[23] S. Young, D. Spanjol, and V. K. Garg. Control of discrete event systems modeled with infinite strings. Technical report, University of Texas at Austin, Austin, Texas 78712, 1990.

A Algorithm for constructing $\Omega(\hat{X})$ and $\Lambda(\hat{X})$

As before, let $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$ be the plant and $\hat{X} \subseteq X$ be the set of legal states. The following algorithm can be used to compute $\Omega(\hat{X})$ (we assume that the plant P has finite number of states so that the algorithm terminates in finite number of steps):

Algorithm A.1

1. **Initiation step:**
   Set $\Omega_{-1}(\hat{X}) = \emptyset, \Omega_0(\hat{X}) = \hat{X}$, and $k = 0$.

2. **Iteration step:**

   (a) Let $X_k \subseteq X$ be the set of states from which $\Omega_k(\hat{X}) - \Omega_{k-1}(\hat{X})$ can be reached in a single transition, i. e.

   $$X_k = \{x \in X \mid \exists \sigma \in \Sigma \text{ s.t. } \alpha(\sigma, x) \in \Omega_k(\hat{X}) - \Omega_{k-1}(\hat{X})\}$$

   Determine the set X_k by considering the SM $P^{-1} \stackrel{\text{def}}{=} (X, \Sigma, \alpha^{-1}, x_0, X_m)$, where $\alpha^{-1}(\sigma, x_2) \stackrel{\text{def}}{=} \{x_1 \in X \mid \alpha(\sigma, x_1) = x_2\}$ (P^{-1} is the SM obtained by reversing all the transitions of P), and by finding the states that can be reached from $\Omega_k(\hat{X}) - \Omega_{k-1}(\hat{X})$ by a single transition in P^{-1}.

   (b) Consider $x \in X_k$. If all the transitions from x lead to $\Omega_k(\hat{X})$, then $\Omega_{k+1}(\hat{X}) = \Omega_k(\hat{X}) \cup \{x\}$. Repeat this for all $x \in X_k$. Thus, if all the transitions from a state $x \in X_k$ lead to states in $\Omega_k(\hat{X})$, then x is a strongly attractable state, i. e.

   $$\Omega_{k+1}(\hat{X}) = \Omega_k(\hat{X}) \cup \{x \in X_k \mid \alpha(\sigma, x) \in \Omega_k(\hat{X}) \text{ for all } \sigma \in \Sigma(P)(x)\}$$

   where $\Sigma(P)(x) \subseteq \Sigma$ is the set of all the transitions that are defined in the state $x \in X$ in P and is given by, $\Sigma(P)(x) = \{\sigma \in \Sigma \mid \alpha(\sigma, x)!\}$.

3. **Termination step:**
   If $\Omega_{k+1}(\hat{X}) = \Omega_k(\hat{X})$, then stop and set $\Omega(\hat{X}) = \Omega_k(\hat{X})$; else set $k = k + 1$ and go to step 2.

Theorem A.2 Algorithm A.1 computes the region of strong attraction $\Omega(\hat{X})$ of the set of legal states $\hat{X} \subseteq X$.

Proof: The proof that the Algorithm A.1 computes $\Omega(\hat{X})$ is based on the following two facts:

Firstly, the above algorithm computes $\Omega(\hat{X})$ if in step 2, $\Omega_k(\hat{X}) - \Omega_{k-1}(\hat{X})$ is replaced by $\Omega_k(\hat{X})$ (for proof refer to Proposition 2.7 of [18]).

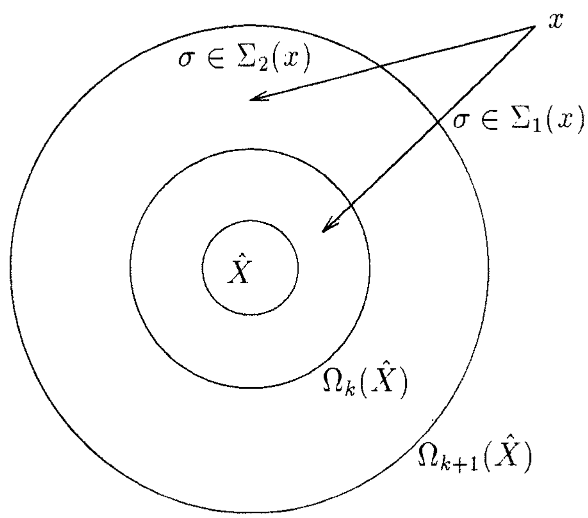


Figure 6: Constructing region of strong attraction

Secondly, at the end of the kth iteration, to determine the states that might be strongly attractable, we just need to consider the states that have transitions leading into the set $\Omega_{k+1}(\hat{X}) - \Omega_k(\hat{X})$ (rather than into the set $\Omega_{k+1}(\hat{X})$) in P, so that the replacement as described above is justified (see Figure 6). In other words, we must show that at the end of kth iteration, if all the transitions in $\Sigma(P)(x)$ from the state $x \in X - \Omega_{k+1}(\hat{X})$ lead to the set $\Omega_{k+1}(\hat{X})$, then there exists $\sigma \in \Sigma(x)$ such that $\alpha(\sigma, x) \in \Omega_{k+1}(\hat{X}) - \Omega_k(\hat{X})$. To show this, we first partition $\Sigma(P)(x)$ into the set $\Sigma_1(P)(x) \cup \Sigma_2(P)(x)$, the set $\Sigma_1(P)(x)$ of transitions leading to $\Omega_k(\hat{X})$ and the set $\Sigma_2(P)(x)$ of transitions leading to $\Omega_{k+1}(\hat{X}) - \Omega_k(\hat{X})$. Then it is enough to show that the set $\Sigma_2(x)$ is nonempty. Assume that it is empty; then $x \in \Omega(\Omega_k(\hat{X}))$ and therefore it belongs to the set $\Omega_{k+1}(\hat{X})$, which is contradictory to the fact that $x \in X - \Omega_{k+1}(\hat{X})$. This proves the second claim. □

Remark A.3 In order to determine the region of weak attraction $\Lambda(\hat{X})$ of $\hat{X}$, we replace step 2(b) in the iteration step of the previous algorithm by the following step $2(b')$:

$2(b')$ Consider $x \in X_k$. If all the uncontrollable transitions from x lead to $\Omega_k(\hat{X})$, then $\Omega_{k+1}(\hat{X}) = \Omega_k(\hat{X}) \cup \{x\}$, i. e.

$$\Omega_{k+1}(\hat{X}) = \Omega_k(\hat{X}) \cup \{x \in X_k \mid \alpha(\sigma, x) \in \Omega_k(\hat{X}) \text{ for all } \sigma \in \Sigma_u(P)(x)\},$$

where $\Sigma_u(P)(x) = \Sigma(P)(x) \cap \Sigma_u$.

This can be tested by considering the transitions in $P\mid_{\Sigma_u}$ (P with all its controllable transitions deleted). Formally, $P\mid_{\Sigma_u} \stackrel{\text{def}}{=} (X, \Sigma_u, \alpha\mid_{\Sigma_u \times X}, x_0, X_m)$.

This would result in the construction of the region of weak attraction $\Lambda(\hat{X})$ of $\hat{X}$. Notice that with an abuse of notation we have used $\Omega_k(\hat{X})$ in the algorithm for determining $\Lambda_k(\hat{X})$.

Theorem A.4 The time complexity of Algorithm A.1 for constructing $\Omega(\hat{X})$ and $\Lambda(\hat{X})$ is $O(|\Sigma|n)$, where $|\Sigma|$ denotes the number of events in the event set Σ and n is the number of states in P.

Proof: Assume that at the end of kth iteration, the number of transitions (of length one) leading into the set $\Omega_{k+1}(\hat{X}) - \Omega_k(\hat{X})$ from $X - \Omega_{k+1}(\hat{X})$ is e_k. We show that step 2 of the algorithm can be computed in $O(e_k)$ time, as follows.

Firstly, the states in the set X_k can be computed in $O(e_k)$ time, for in order to determine the states reachable from the states in the set $\Omega_{k+1}(\hat{X}) - \Omega_k(\hat{X})$ by a single transition in P^{-1}, we need consider only the e_k transitions. Secondly, since there could be at most e_k such states, the states in the set $\Omega_{k+1}(\hat{X})$ can also be computed in $O(e_k)$ time. This is true because to test whether a state $x \in X_k$ belongs to $\Omega_{k+1}(\hat{X})$ requires only $O(|\Sigma|)$ time which is constant.

Since the sets $\Omega_{k+1}(\hat{X}) - \Omega_k(\hat{X})$ for each value of k are all disjoint, the transitions (of length one) leading into them from $X - \Omega_{k+1}(\hat{X})$ are also all disjoint. Hence the computational complexity of Algorithm A.1 is of order $O(\sum_k e_k) = O(e)$, where e is the number of transitions in P. Since P is deterministic, $e \leq |\Sigma|n$, hence the theorem follows. Similarly, the complexity of the algorithm for determining $\Lambda(\hat{X})$ is also $O(|\Sigma|n)$. □

This is significant improvement over the computational complexity of the algorithm given in [2, 3], which is $O(n^2)$. Notice that our algorithm requires the construction of the SM P^{-1} which could be nondeterministic, but has same number of transitions as P.

The above algorithm can also be used to construct the *prestable* and *prestabilizable* states of a given *invariant* state set as defined in [18]. In fact, the set of prestable states and the set of prestabilizable states with respect to a given invariant or legal set of states is the same as $\Omega(\hat{X})$ and $\Lambda(\hat{X})$ respectively, where $\hat{X}$ denotes the set of invariant states. The computational complexity of the algorithms provided in [18] is also quadratic in the number of states of P.

B An Equivalence Relation on Σ^ω and ω-Stability

A necessary condition for ω-stability of a given ω-language with respect to another can be obtained in terms of an equivalence relation defined on the space Σ^ω. In this appendix we define this relation and show its close relation to the notion of ω-stability.

Definition B.1 For $e_1, e_2 \in \Sigma^\omega$, $e_1 \cong e_2$ if and only if there exist $m, n \in \mathcal{N}$ such that $\Pi_m(e_1) = \Pi_n(e_2)$.

Note that for each $n \in \mathcal{N}$, $\Pi_n : \Sigma^\omega \to \Sigma^\omega$ is the map such that for $e \in \Sigma^\omega$, $\Pi_n(e)$ is the infinite string obtained by removing the prefix of length n from e.

Theorem B.2 The relation $\cong$ as defined is an equivalence relation.

Proof: We need to show that the relation $\cong$ is reflexive, symmetric and transitive.

It is clear that for any vector $e_1 \in \Sigma^\omega$, $e_1 \cong e_1$, i.e. $\cong$ is reflexive. Also, if $e_1 \cong e_2$, then clearly $e_2 \cong e_1$ for any two vectors $e_1, e_2 \in \Sigma^\omega$, i.e. $\cong$ is symmetric. It remains to show that the relation $\cong$ is transitive. Pick any $e_1, e_2, e_3 \in \Sigma^\omega$. We will show that $e_1 \cong e_2$ and $e_2 \cong e_3$ implies $e_1 \cong e_3$. Let $m, n, p, q \in \mathcal{N}$ be such that $\Pi_m(e_1) = \Pi_n(e_2)$ and $\Pi_p(e_2) = \Pi_q(e_3)$. We may have either $n \leq p$ or $p \leq n$. If $n \leq p$, then $\Pi_{m+(p-n)}(e_1) = \Pi_q(e_3)$, i.e. $e_1 \cong e_3$; if $p \leq n$, then $\Pi_m(e_1) = \Pi_{q+(n-p)}(e_3)$, i.e. $e_1 \cong e_3$. □

A necessary condition for ω-stability can be obtained using the equivalence relation defined above.

Proposition B.3 If plant sequential behavior $\mathcal{L}(P)$ is ω-stable with respect to the desired eventual behavior $\mathcal{K}$, then for each $e \in \mathcal{L}(P)$, there exists $e' \in \mathcal{K}$ such that $e \cong e'$.

Proof: Assume $\mathcal{L}(P)$ is ω-stable with respect to $\mathcal{K}$, i.e. there exist $N \in \mathcal{N}$ such that $\mathcal{L}(P) \subseteq \Sigma^{\leq N}\mathcal{K}$. Then given $e \in \mathcal{L}(P)$, there exists $n \leq N$ and $e' \in \mathcal{K}$ such that $e = e^n e'$. Thus $\Pi_n(e) = e'$, i.e. $e \cong e'$. □

Remark B.4 Proposition B.3 gives a necessary condition for ω-stablity. This condition will be a necessary as well as sufficient condition if a weaker definition of ω-stability is used. Let the projection operator be extended to the space 2^{Σ^ω} in the obvious manner, i.e. for any $n \in \mathcal{N}$, $\Pi_n : 2^{\Sigma^\omega} \to 2^{\Sigma^\omega}$ is defined to be:

$$\Pi_n(\mathcal{L}) = \{e \in \Sigma^\omega \mid \exists e' \in \mathcal{L} \text{ s.t. } \Pi_n(e') = e\}$$

where $\mathcal{L} \subseteq \Sigma^\omega$. We use $\Pi_\star(\cdot)$ to denote the operator $\bigcup_{n \in \mathcal{N}} \Pi_n(\cdot)$. The plant sequential behavior $\mathcal{L}(P)$ is said to be *weakly ω-stable* with respect to the desired eventual sequential behavior $\mathcal{K}$ if $\mathcal{L}(P) \subseteq \Sigma^\star \Pi_\star(\mathcal{K})$. Thus if $\mathcal{L}(P)$ is weakly ω-stable with respect to $\mathcal{K}$, then for every $e \in \mathcal{L}(P)$ there exist $n, m \in \mathcal{N}$ and $e' \in \mathcal{K}$ such that $\Pi_n(e) = \Pi_m(e')$. It is clear that ω-stability implies weak ω-stability. It can easily be verified that $\mathcal{L}(P)$ is weakly ω-stable with respect to $\mathcal{K}$ if and only if given any $e \in \mathcal{L}(P)$ there exists $e' \in \mathcal{K}$ such that $e \cong e'$.