Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Nonblocking Directed Control of Discrete Event Systems

Jing Huang and Ratnesh Kumar
Department of Electrical and Computer Engineering
Iowa State University, Ames, Iowa 50011

***Abstract*— We introduce the notion of *directed control*, where a directed controller, simply referred as *director*, is one that selects at most one controllable event to be enabled at any instant. This is in contrast to *supervisory control*, where a supervisory controller, simply referred as *supervisor*, enables a maximum allowable set of controllable events at any instant, i.e., no specific selection for executing an enabled event is made. While a supervisor design is meaningful for plants that are *generator* of controllable events, a director design makes more sense for plants that are *executor* of controllable events. In this paper we prove that a nonblocking director exists if and only if a nonblocking supervisor exists, thereby proving the polynomiality of verifying existence. We also develop a set of algorithms of polynomial complexity to compute a nonblocking director.**

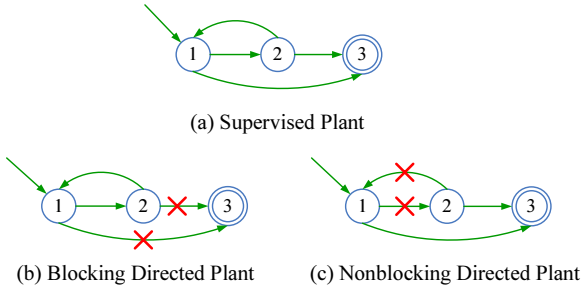
***Index Terms*— discrete event system, directed control, director, supervisory control, supervisor, nonblocking, uncontrollable disturbance input, uncontrollable sensor output, maximally permissive supervisor, automata**

I. Introduction

Most prior work on logical control of discrete event systems deals with supervisory control [6], [4], [2], a goal of which is to enforce a given specification by minimally restricting the behavior of a given discrete event system (called plant). In a transportation system, for example, a supervisory control action will specify a maximal set of permissible routes for a vehicle. However, what is more appropriate is a directed control action commanding the vehicle to follow a specific route. So for systems that are executor of events, it is more meaningful to issue a command consisting of a single possible controllable event, rather than a set of controllable events as issued by a supervisor.

In [1], an antenna rotor control system (ARCS) has been designed where a controller enforces the given safety, liveness, and real-time control constraints, while selecting a single controllable event at each state of the system. The controllable event is selected from the ones allowed by a maximally permissive supervisor. This selection, however, is done on an *ad hoc* basis. Similar ad hoc selection of controllable event is made in another application consisting of an educational assembly line [3].

While it is possible to arbitrarily select one of the controllable events among the ones enabled by a supervisor to "extract" a director, such an ad hoc selection can lead to blocking. For example, consider a plant under the control of a supervisor shown in Figure 1(a). An arbitrary disablement of all but one controllable event to obtain a director may result in blocking, as shown in Figure 1(b). On the other hand, another way of disabling all but one controllable event results in a nonblocking director, as shown in Figure 1(c). Thus it is clear that one needs an algorithmic approach to search for a director. A contribution of the paper is to provide a polynomial complexity algorithm to obtain a nonblocking director. We also show that a nonblocking director exists if and only if a nonblocking supervisor exists, i.e., existence and synthesis are both polynomially solvable. (Recall that existence of a nonblocking supervisor is polynomially verifiable.)



(a) Supervised Plant

(b) Blocking Directed Plant

(c) Nonblocking Directed Plant

Fig. 1. From Supervisor to Director

The remainder of this paper is organized as follows. Section II gives the basic notation and preliminaries. Section III introduces the notion of directed control. Section IV presents an existence condition of a nonblocking director. Section V presents an algorithm to compute a nonblocking director, along with some examples to aid the understanding of the approach, while Section VI concludes the paper. An appendix is provided containing some relevant algorithms.

II. Notation and Preliminaries

A DES to be controlled, called plant, is modeled as an automaton, denoted by a five tuple $G := (X, \Sigma, \alpha, x_0, X_m)$, where X denotes the set of states, Σ denotes the finite set of events, $\alpha : X \times \Sigma \to X$ denotes the partial deterministic state transition function, $x_0 \in X$ denotes the initial state, and $X_m \subseteq X$ denotes the set of marked states. Σ^* is used to denote set of all finite-length sequences of events, called traces, which include the zero-length trace ϵ. A subset of Σ^* is called a language. The generated language of G, denoted as $L(G) \subseteq \Sigma^*$, contains all traces s for which $\alpha(x_0, s)$ is defined. The marked language of G, denoted as $L_m(G)$, contains all traces $t \in L(G)$ such that $\alpha(x_0, t) \in X_m$. For

This work was supported in part by the National Science Foundation under the grants NSF-ECS-0218207, NSF-ECS-0244732, NSF-EPNES-0323379, NSF-0424048 and a DoD-EPSCoR grant through the Office of Naval Research under the grant N000140110621.

$x \in X$, we use $\Sigma(x) \subseteq \Sigma$ to denote the set of events defined at x, i.e., $\Sigma(x) := \{\sigma \in \Sigma \mid \alpha(x, \sigma) \text{ is defined}\}$. We use $K\backslash s$ to denote the set of traces that can occur in the language K after the trace s has occurred, i.e., $K\backslash s := \{t \in \Sigma^* \mid st \in K\}$. For traces s and t, we use $s \leq t$ to denote that s is a prefix of t and $s < t$ to denote that s is a proper prefix of t.

For control purposes, the event set of G is partitioned into the set of controllable events $\Sigma_c \subseteq \Sigma$ and the set of uncontrollable events $\Sigma_u \subseteq \Sigma$. An uncontrollable event can be either a disturbance input or a sensor output. Occurrence of a disturbance input is uncertain while that of a sensor output is something expected. The set of uncontrollable events that are disturbance inputs is denoted as $\Sigma_d \subseteq \Sigma_u$. A supervisory controller is a map $S : L(G) \rightarrow 2^{\Sigma - \Sigma_u}$ that determines the set of events $S(s) \subseteq (\Sigma - \Sigma_u)$ to be disabled after the occurrence of a trace $s \in L(G)$. Events not belonging to the set $S(s)$ remain enabled at trace s. In particular, the uncontrollable events remain enabled.

III. Notion of Directed Control

A director enables at most one controllable event at each state. This control selection is what distinguishes a director from a supervisor. A useful class of directors consists of those that enable exactly one controllable event following certain plant traces. These traces are candidates for control (i.e., one or more controllable events are executable in the next step) and all executable uncontrollable events are disturbance inputs. For such traces, disabling all controllable events is not a desirable option since this will make the controlled system either wait for a disturbance input to occur before evolving further, or idle forever if no disturbance input is feasible.

Definition 1: A trace $s \in L(G)$ is a control-candidate if $L(G)\backslash s \cap \Sigma_c \neq \emptyset$; it is called a disturbance trace if $L(G)\backslash s \cap \Sigma_u \subseteq \Sigma_d$. We represent the set of all control-candidate traces that are also disturbance traces as $L_{CD}(G) \subseteq L(G)$, i.e.,

$$L_{CD}(G) := \{s \in L(G) \mid L(G)\backslash s \cap \Sigma_c \neq \emptyset \text{ and } L(G)\backslash s \cap \Sigma_u \subseteq \Sigma_d\}.$$

Then a director is a map $D : L(G) \rightarrow 2^{\Sigma_c}$ such that

$$\forall s \in L(G) : |D(s)| \leq 1 \text{ and } \forall s \in L_{CD}(G) : |D(s)| = 1.$$

The set of events enabled by such a director following a trace $s \in L(G)$ is given by $D(s) \cup \Sigma_u$. After the execution of a trace $s \in L(G)$, the director enables at most one controllable event unless $s \in L_{CD}(G)$, in which case the director enables exactly one controllable event. Also note that no control decision is defined with respect to uncontrollable events; such events remain enabled.

The directed plant is denoted by G^D, and the languages generated and marked by the directed plant are denoted by $L(G^D)$ and $L_m(G^D)$ respectively, which are defined as follows:

$\epsilon \in L(G^D)$;
$[s \in L(G^D), \sigma \in D(s) \cup \Sigma_u, s\sigma \in L(G)] \Leftrightarrow [s\sigma \in L(G^D)]$;
$L_m(G^D) := L(G^D) \cap L_m(G)$.

It is clear that $pr(L_m(G^D)) \subseteq L(G^D)$. A director D is said to be nonblocking if $pr(L_m(G^D)) = L(G^D)$.

For simplicity, we will consider state-based specification and control. It is known that a language-based specification (resp. control) can be converted to a state-based specification (resp. control) on a suitably refined plant model. A state-based specification is given using a set of illegal states $X_i \subseteq X$ that must never be visited, whereas a director D is state-based if it computes control action as a function of plant state, i.e., $D : X \rightarrow 2^{\Sigma_c}$.

Definition 2: A state $x \in X$ is said to be a control-candidate if it has at least one controllable event defined; it is called a disturbance state if all uncontrollable events defined at the state are disturbance inputs. We represent the set of all control-candidate states that are also disturbance states as $X_{CD} \subseteq X$, i.e.,

$$X_{CD} := \{x \in X \mid \Sigma(x) \cap \Sigma_c \neq \emptyset \text{ and } \Sigma(x) \cap \Sigma_u \subseteq \Sigma_d\}.$$

Then a state-based director is a map $D : X \rightarrow 2^{\Sigma_c}$ such that

$$\forall x \in X : |D(x)| \leq 1 \text{ and } \forall x \in X_{CD} : |D(x)| = 1.$$

Under the control of a state-based director D, the controlled plant is a subgraph of the plant graph,

$$G^D := (X, \Sigma, \alpha^D, x_0, X_m),$$

where the state-transition function $\alpha^D(x, \sigma)$ is defined as follows.

$$\alpha^D(x, \sigma) := \begin{cases} \alpha(x, \sigma) & \text{if } \sigma \in D(x) \cup \Sigma_u \text{ and } \alpha(x, \sigma) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

For a state-based director to be nonblocking, the following must hold: if $x \in X$ is such that there exists $s \in \Sigma^*$ with $\alpha^D(x_0, s) = x$, then there exists $t \in \Sigma^*$ such that $\alpha^D(x_0, st) \in X_m$.

Definition 3: Given a plant $G := (X, \Sigma, \alpha, x_0, X_m)$, a component $(\hat{X}, \hat{\alpha})$ of G is a subgraph of G satisfying $\hat{X} \subseteq X$ and $\hat{\alpha} \subseteq \alpha|_{\hat{X}}$.

The set of all possible directors for a component $(\hat{X}, \hat{\alpha})$ is denoted as $\mathcal{D}(\hat{X}, \hat{\alpha})$.

Central to our algorithm for synthesizing a nonblocking director is the observation that any given graph, including a controlled plant graph, can be partitioned into a set of strongly-connected components (SCCs), over which the given graph possesses a tree structure. In case of a directed plant graph, the leaf nodes of such a tree must be *strongly-connected*, *legal*, *invariant*, *nonblocking* and *control-consistent* components ($SLINC$s) while the non-leaf nodes must be *strongly-connected*, *legal*, $SLINC$-*attractable* and *control-consistent* components ($SLAC$s). These notions are formalized in the following.

Definition 4: A component $(\hat{X}, \hat{\alpha})$ is called

1) *strongly-connected* if there exists a path lying entirely within the component between any pair of states of the component, i.e.,

$$\forall x_1, x_2 \in \hat{X},\ \exists s \in \Sigma^* \text{ s.t. } \hat{\alpha}(x_1, s) = x_2 \text{ and } \forall t \leq s : \hat{\alpha}(x_1, t) \in \hat{X}.$$

2) *legal* if there is no illegal state inside the component, i.e.,

$$\hat{X} \cap X_i = \emptyset.$$

3) *control-consistent* if there is at most one controllable event defined at each state and exactly one controllable event defined at each control-candidate state that is also a disturbance state, i.e.,

$$\forall x \in \hat{X} : |\Sigma_c^{\hat{\alpha}}(x)| \leq 1 \text{ and } \forall x \in \hat{X} \cap X_{CD} : |\Sigma_c^{\hat{\alpha}}(x)| = 1$$
$$\text{where } \Sigma_c^{\hat{\alpha}}(x) = \{\sigma \in \Sigma_c \mid \hat{\alpha}(x, \sigma) \text{ is defined}\}.$$

4) *nonblocking* if from any state of the component, a marked state can be reached within the component, i.e.,

$$\forall x \in \hat{X},\ \exists s \in \Sigma^* \text{ s.t. } \hat{\alpha}(x, s) \in X_m \text{ and } \forall t \leq s : \hat{\alpha}(x, t) \in \hat{X}.$$

5) *invariant* if the state transitions of $(\hat{X}, \hat{\alpha})$ can be confined within $\hat{X}$ under directed control, i.e.,

$$\hat{\alpha}(\hat{X}, \Sigma_u) \subseteq \hat{X} \text{ and } \forall x \in \hat{X} \cap X_{CD} : \hat{\alpha}(x, \Sigma_c) \cap \hat{X} \neq \emptyset.$$

Otherwise, the component is called *variant*. We represent the set of states in $\hat{X}$ violating any of the above conditions, i.e. the set of *variant* states, as $V(\hat{X}, \hat{\alpha}) \subseteq \hat{X}$.

6) X_r*-attractable*, where X_r is a reference state set, if there exists a component $(\widetilde{X}, \widetilde{\alpha}) \supseteq (\hat{X}, \hat{\alpha})$ such that every state of $(\widetilde{X}, \widetilde{\alpha})$ can reach a state of X_r over transitions within $\widetilde{X} \cup X_r$, and the state transitions of $(\widetilde{X}, \widetilde{\alpha})$ can be confined within $\widetilde{X} \cup X_r$ under directed control, i.e.,
   a) $\forall x \in \widetilde{X}, \exists s \in \Sigma^*$ such that $\widetilde{\alpha}(x, s) \in X_r$ & $\forall t < s : \widetilde{\alpha}(x, t) \in \widetilde{X}$, and
   b) $\widetilde{\alpha}(\widetilde{X}, \Sigma_u) \subseteq \widetilde{X} \cup X_r$, and
   c) $\forall x \in \widetilde{X} \cap X_{CD} : \widetilde{\alpha}(x, \Sigma_c) \cap [\widetilde{X} \cup X_r] \neq \emptyset$.

   In case $(\widetilde{X}, \widetilde{\alpha})$ can be chosen as the component $(\hat{X}, \hat{\alpha})$ itself, $(\hat{X}, \hat{\alpha})$ is said to be *singularly* X_r*-attractable*. Also, the notion $U((\hat{X}, \hat{\alpha}), X_r) \subseteq \hat{X}$ is used to denote the set of states in $\hat{X}$ violating any of the 3 conditions in the definition of singular X_r-attractability.

Definition 5: Given a component $(\hat{X}, \hat{\alpha})$ and a reference state set $X_r \subseteq X$, we represent the set of maximal subcomponents of $(\hat{X}, \hat{\alpha})$ that are

1) *strongly-connected* as, $\mathcal{S}(\hat{X}, \hat{\alpha})$;
2) *strongly-connected, legal, invariant and nonblocking* as, $\mathcal{SLIN}(\hat{X}, \hat{\alpha})$;
3) *strongly-connected, legal and singularly* X_r*-attractable* as, $\mathcal{SLA}((\hat{X}, \hat{\alpha}), X_r)$.

Algorithmic computation of $\mathcal{S}(\hat{X}, \hat{\alpha})$ is well-known (and so not presented in the paper); the algorithms to compute $\mathcal{SLIN}(\hat{X}, \hat{\alpha})$ and $\mathcal{SLA}((\hat{X}, \hat{\alpha}), X_r)$ are presented in the Appendix for reference.

IV. Existence of Directed Controller

Given a plant $G := (X, \Sigma, \alpha, x_0, X_m)$, the control goal is to find a state-based nonblocking director $D : X \to 2^{\Sigma_c}$ such that illegal states are never visited. It turns out that the existence of such a director can be determined by checking the existence of a nonblocking supervisor. Since a nonblocking supervisor exists if and only if a maximally permissive nonblocking supervisor exists [6], we first present an algorithm to compute such a supervisor, taken from [5]. Again the central idea is that the graph of a maximally permissive nonblocking supervised plant is partitionable into SCCs, over which it possesses a tree-structure (see Figure 2(a)). The leaf nodes of the tree are SCCs that are legal, invariant and nonblocking. Other SCCs are legal and attractable to leaf nodes.

The algorithm first identifies strongly-connected, legal, invariant and nonblocking components ($SLIN$s) as the leaf nodes, and then iteratively searches backwards to identify strongly-connected, legal and $SLIN$-attractable components (SLAs) for non-leaf nodes. The iterative backward search terminates when either the *root* node (i.e., a SLA containing the initial state) is found, or no further SLAs can be added as nodes to the tree. In the former case, a nonblocking supervisor exists, whereas in the latter case, a nonblocking supervisor does not exist.

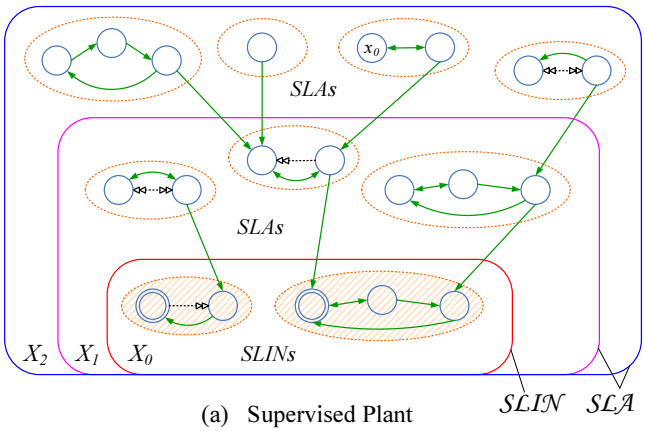
Algorithm 1: Given a plant graph (X, α) containing some marked and illegal states, the following steps computes a maximally permissive nonblocking supervisor [5].

1) Compute $\mathcal{SLIN}(X, \alpha)$. Let X_0 denote the set of states of all those components; $k = 0$.
2) a) Compute singularly X_k-attractable region in the remainder of the plant, i.e., compute $\mathcal{SLA}((X - X_k, \alpha|_{X - X_k}), X_k)$; Let $\widetilde{X}$ denote the set of states of all those components.
   b) Augment X_k with $\widetilde{X}$ to get X_{k+1}, i.e., $X_{k+1} = X_k \cup \widetilde{X}$.
3) Repeat Step 2 with $k = k + 1$ until
   a) $x_0 \in X_k$, in which case a maximally permissive nonblocking supervisor is found, or
   b) $X_{k+1} = X_k$, in which case no nonblocking supervisor exists.

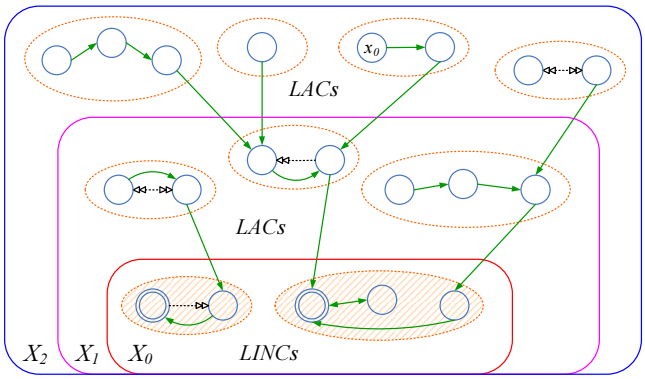
In the remainder of this section, we show that the director existence is equivalent to the supervisor existence. We need the following lemmas to establish our existence result.

Lemma 1: For any strongly-connected, legal, invariant and nonblocking component $(\hat{X}, \hat{\alpha})$, there exists one $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ such that $(\hat{X}, \hat{\alpha}^D)$ is legal, invariant, nonblocking and control-consistent.

Lemma 2: For any strongly-connected, legal and singularly X_r-attractable component $(\hat{X}, \hat{\alpha})$, there exists one



(a) Supervised Plant



(b) Directed Plant

Fig. 2. Structure of Plants under Control

$D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ such that $(\hat{X}, \hat{\alpha}^D)$ is legal, singularly X_r-attractable and control-consistent.

With the above Lemma 1 and 2 at hand, we can establish the following theorem.

Theorem 1: There exists a nonblocking director for a plant G if and only if there exists a maximally permissive nonblocking supervisor for G.

Proof: Suppose a maximally permissive nonblocking supervisor exists and is computed by Algorithm 1. Then from Lemma 1, we can transform each component $(\hat{X}, \hat{\alpha}) \in \mathcal{SLIN}(X, \alpha)$ to a legal, invariant, nonblocking and control-consistent component ($LINC$), and similarly by Lemma 2, we can transform each component $(\hat{X}, \hat{\alpha}) \in \mathcal{SLA}((X - X_k, \alpha|_{X-X_k}), X_k)$ to a legal, $LINC$-attractable and control-consistent component (LAC). Following such transformations, a graph such as in Figure 2(a) is converted to a graph such as in Figure 2(b), yielding a tree structure with leaf nodes consisting of $LINC$s (which are partitionable into $SLINC$s and $SLAC$s) and non-leaf nodes consisting of LACs (which are partitionable into $SLAC$s), thereby yielding a desired nonblocking directed plant graph.

Conversely, if a nonblocking director exists, then since a director is also a supervisor, a nonblocking supervisor exists. Finally since a nonblocking supervisor exists if and only if the maximally permissive nonblocking supervisor exists (controllability and relative-closure are preserved under union), the assertion of the theorem follows. ■

V. Synthesis of Directed Controller

We use the ideas developed in the previous section to present a set of algorithms for synthesizing a nonblocking director. The first algorithm transforms a $SLIN$ to a $LINC$; the second transforms a SLA to a LAC, and the final algorithm performs backward search over the tree of SCCs in the graph of a maximally permissive nonblocking supervised plant to find a desired director.

Algorithm 2: Given a strongly-connected, legal, invariant and nonblocking component $(\hat{X}, \hat{\alpha})$, the following steps compute a director $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ such that $(\hat{X}, \hat{\alpha}^D)$ is legal, invariant, nonblocking and control-consistent.

1) $\widetilde{X} = \hat{X}$; $X_0 = \emptyset$; $X_1 := \hat{X} \cap X_m$; $k := 1$;
2) For each $x \in X_k$, set the control action as

$$D(x) := \begin{cases} \{\sigma\}, \text{ where } \sigma \in \Sigma_c \text{ is } \textit{any} \text{ controllable event s.t. } \hat{\alpha}(x,\sigma) \in X_{k-1} & \text{if } \hat{\alpha}(x, \Sigma_c) \cap X_{k-1} \neq \emptyset \\ \emptyset & \text{if } \hat{\alpha}(x, \Sigma_c) = \emptyset \\ \{\sigma\}, \text{ where } \sigma \in \Sigma_c \text{ is } \textit{any} \text{ controllable event s.t. } \hat{\alpha}(x,\sigma) \in \hat{X} & \text{otherwise} \end{cases}$$

3) $\widetilde{X} = \widetilde{X} - X_k$; If $\widetilde{X} = \emptyset$, then terminate the algorithm, else continue the following steps;
4) $X_{k+1} := \{x \in \widetilde{X} \mid \hat{\alpha}(x, \Sigma) \cap X_k \neq \emptyset\}$; $k := k + 1$
5) Go back to Step 2.

We present an example to aide the understanding of Algorithm 2.

Example 1: Consider a $SLIN$ $(\hat{X}, \hat{\alpha})$ shown in Figure 3(a), where we represent illegal states by crossing them. The edges with solid line and single arrow represent transitions on controllable events while those with dashed line and double arrows represent transitions on uncontrollable events. For simplicity of discussion, we represent a component of interest by its state set. The associated state transition function can be readily identified from the figures. The figures show how one director $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ is computed by Algorithm 2 iteratively such that $(\hat{X}, \hat{\alpha}^D)$ is a $LINC$. The resulting $(\hat{X}, \hat{\alpha}^D)$ is shown in Figure 3(d). Note that a controllable event, if disabled by the underlying director, is omitted from the corresponding figures.

Remark 1: Let $|\hat{X}|$ be the number of state in a $SLIN$. Due to determinism, there are at most $|\hat{X}||\Sigma|$ transitions, and it can be verified that complexity of Algorithm 2 is of order $O(|\hat{X}||\Sigma|)$.

Next we present an algorithm to transform a SLA to a LAC.

Algorithm 3: Given a reference set $X_r \in X$ and a strongly-connected, legal, singularly X_r-attractable component $(\hat{X}, \hat{\alpha})$, the following steps compute a director $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ such that $(\hat{X}, \hat{\alpha}^D)$ is legal, singularly X_r-attractable and control-consistent.

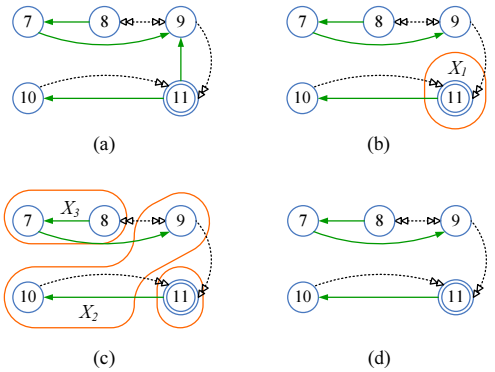


Fig. 3. Transformation from $SLIN$ to $LINC$

1) $\widetilde{X} = \hat{X}$; $k := 0$; $X_0 := X_r$;
2) $X_{k+1} := \{x \in \widetilde{X} \mid \hat{\alpha}(x, \Sigma) \cap X_k \neq \emptyset\}$; $k := k + 1$;
3) For each $x \in X_k$, set the control action as

$$D(x) := \begin{cases} \{\sigma\}, \text{ where } \sigma \in \Sigma_c \text{ is } \textit{any} \text{ controllable event s.t. } \hat{\alpha}(x,\sigma) \in X_{k-1} & \text{if } \hat{\alpha}(x, \Sigma_c) \cap X_{k-1} \neq \emptyset \\ \emptyset & \text{if } \hat{\alpha}(x, \Sigma_c) = \emptyset \\ \{\sigma\}, \text{ where } \sigma \in \Sigma_c \text{ is } \textit{any} \text{ controllable event s.t. } \hat{\alpha}(x,\sigma) \in \hat{X} & \text{otherwise} \end{cases}$$

4) $\widetilde{X} = \widetilde{X} - X_k$; If $\widetilde{X} = \emptyset$, then terminate the algorithm, else go back to Step 2.

We present an example to aide the understanding of Algorithm 3.

Example 2: Consider a SLA $(\hat{X}, \hat{\alpha})$ circled in Figure 4(a) and a reference state set X_r. The figures shows how one director $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ is computed by Algorithm 3 iteratively such that $(\hat{X}, \hat{\alpha}^D)$ is a LAC. The resulting $(\hat{X}, \hat{\alpha}^D)$ is circled in Figure 4(d).

Remark 2: Similar to the complexity of Algorithm 2, the complexity of Algorithm 3 is also of order $O(|\hat{X}||\Sigma|)$, where $|\hat{X}|$ is the number of states in a SLA.

Now we are ready to present the final algorithm that performs backward search over the tree of SCCs in the graph of a maximally permissive nonblocking supervised plant to obtain a nonblocking director.

Algorithm 4: Given a plant $G := (X, \Sigma, \alpha, x_0, X_m)$, the following steps compute a nonblocking director.

(I) **Initiation:**

1. Compute $\mathcal{SLIN}(X, \alpha)$. If $\mathcal{SLIN}(X, \alpha) = \emptyset$, then go to Step III.1; else do the following.
2. Set $k := 0$ and let $(X_k, \alpha_k) := \bigcup_{(\hat{X},\hat{\alpha}) \in \mathcal{SLIN}(X,\alpha)} (\hat{X}, \hat{\alpha}^D)$, where $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ is a director computed by Algorithm 2 for each $(\hat{X}, \hat{\alpha}) \in \mathcal{SLIN}(X, \alpha)$.

(II) **Iteration:**

1. If $x_0 \in X_k$, then go to Step III.2; else do the following.

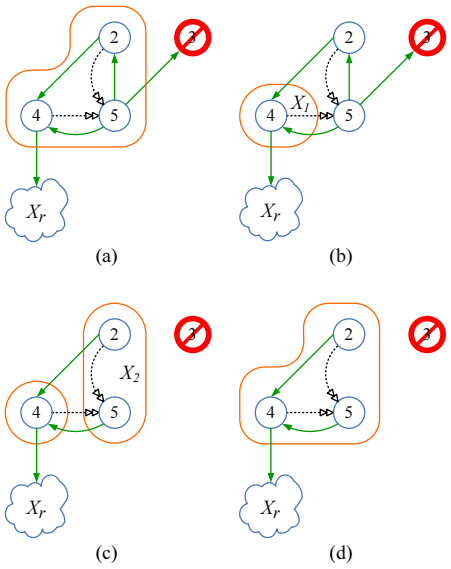


Fig. 4. Transformation from SLA to LAC

2. Let $X' := X - X_k$; $\alpha' := \alpha|_{X'}$. Compute $\mathcal{SLA}((X', \alpha'), X_k)$. If $\mathcal{SLA}((X', \alpha'), X_k) = \emptyset$, then go to Step III.1; else do the following.
3. $(\widetilde{X}, \widetilde{\alpha}) := \bigcup_{(\hat{X},\hat{\alpha}) \in \mathcal{SLA}((X',\alpha'),X_k)} (\hat{X}, \hat{\alpha}^D)$, where $D \in \mathcal{D}(\hat{X}, \hat{\alpha})$ is a director computed by Algorithm 3 for each $(\hat{X}, \hat{\alpha}) \in \mathcal{SLA}((X', \alpha'), X_k)$;
4. $(X_{k+1}, \alpha_{k+1}) := (X_k, \alpha_k) \cup (\widetilde{X}, \widetilde{\alpha})$;
5. Go back to Step II.1 with $k := k + 1$.

(III) **Termination:**

1. Stop since no nonblocking director exists.
2. Stop since a nonblocking director is found.

We present an example to aide the understanding of Algorithm 4.

Example 3: Consider a plant $G = (X, \Sigma, \alpha, x_0, X_m)$ shown in Figure 5(a). The figures show how a nonblocking director is computed by Algorithm 4 iteratively. The resulting nonblocking director is circled in Figure 5(h). Note that this plant includes the components used in Example 1 and 2, so some constructions performed previously are reused to facilitate the computation, which are circled and shaded in Figure 5(d) and (f), respectively. Also note that the transition from state 6 to state 5 is on an uncontrollable sensor output, which is illustrated by dashed line with double *solid* arrows to distinguish it from the transitions on uncontrollable disturbance input, which are represented by dashed line with double *hollow* arrows.

Remark 3: From [5], we know the overall complexity of computing a maximally permissive nonblocking supervisor is quadratic, namely, of order $O(|\hat{X}|^2|\Sigma|^2)$. In addition, we also show in the Remark 1 and 2 that the complexity of additional steps to transform $SLIN/SLA$ to $LINC/LAC$ is linear. Thus the complexity of the above set of algorithms that computes a nonblocking director is also of order $O(|\hat{X}|^2|\Sigma|^2)$.

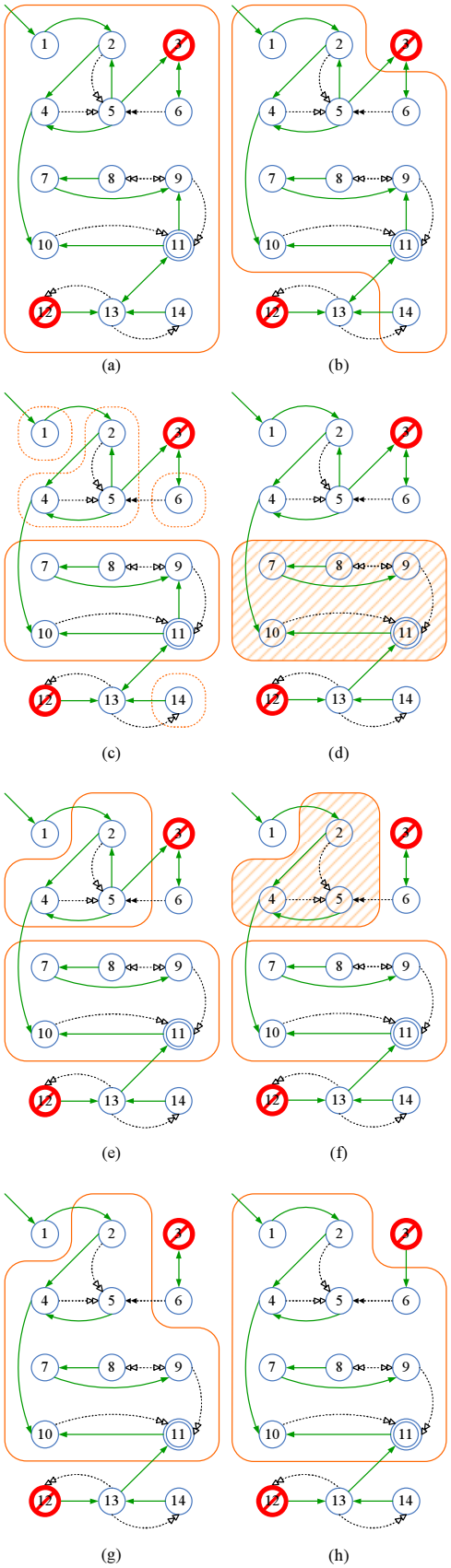


Fig. 5. Computing a Nonblocking Directed Controller

VI. Conclusion

In this paper, we introduce the notion of directed control and develop a set of algorithms to compute a nonblocking directed controller. A directed control makes more sense when plant is an executor of controllable events rather a generator. We show that a directed controller exists for a plant if and only if a nonblocking supervisor exists. The complexity of existence and synthesis of a nonblocking directed controller is polynomial in the size of the plant, and is the same as that of a nonblocking supervisor. Future work will extend to include partial observation and optimal nonblocking directed controller.

VII. Appendix

Algorithm 5: Given a component $(\hat{X}, \hat{\alpha})$, the following steps compute $\mathcal{SLIN}(\hat{X}, \hat{\alpha})$.

1. $(X_0, \alpha_0) := (\hat{X} - X_i, \hat{\alpha}|_{\hat{X}-X_i})$; $k := 0$.
2. $X_{k+1} := X_k - V(X_k, \alpha_k)$; $\alpha_{k+1} := \alpha_k|_{X_{k+1}}$.
3. Repeat Step 2 with $k := k + 1$ until $X_{k+1} = X_k$.
4. Compute $\mathcal{S}(X_k, \alpha_k)$.
5. For each $(\widetilde{X}, \widetilde{\alpha}) \in \mathcal{S}(X_k, \alpha_k)$, if it is
   (a) invariant and nonblocking, then $(\widetilde{X}, \widetilde{\alpha}) \in \mathcal{SLIN}(\hat{X}, \hat{\alpha})$;
   (b) variant and nonblocking, then go to Step 1 with $(X_0, \alpha_0) := (\widetilde{X}, \widetilde{\alpha})$;
   (c) blocking, then $(\widetilde{X}, \widetilde{\alpha}) \notin \mathcal{SLIN}(\hat{X}, \hat{\alpha})$.

Algorithm 6: Given a reference state set $X_r \subseteq X$ and a component $(\hat{X}, \hat{\alpha})$, the following steps compute $\mathcal{SLA}((\hat{X}, \hat{\alpha}), X_r)$.

1. $(X_0, \alpha_0) := (\hat{X} - X_i, \hat{\alpha}|_{\hat{X}-X_i})$; $k := 0$.
2. $X_{k+1} := X_k - U((X_k, \alpha_k), X_r)$; $\alpha_{k+1} := \alpha_k|_{X_{k+1}}$.
3. Repeat Step 2 with $k := k + 1$ until $X_{k+1} = X_k$.
4. Compute $\mathcal{S}(X_k, \alpha_k)$.
5. For each $(\widetilde{X}, \widetilde{\alpha}) \in \mathcal{S}(X_k, \alpha_k)$, if it is singularly X_r-attractable, then $(\widetilde{X}, \widetilde{\alpha}) \in \mathcal{SLA}\ ((\hat{X}, \hat{\alpha}), X_r)$; otherwise go to Step 1 with $(X_0, \alpha_0) = (\widetilde{X}, \widetilde{\alpha})$.

References

[1] M. Barbeau, M. Frappier, F. Kabaza, and R. St.-Denis. A supervisory control synthesis case study: The antenna control system. In *Proceedings of 1997 Allerton Conference on Communication, Control, and Computing*, pages 533–542, 1997.

[2] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, Boston, MA, 1999.

[3] V. Chandra, Z. Huang, and R. Kumar. Automated control syntesis for and assembly line using discrete event system control theory. *IEEE Transactions on Systems, Man, and Cybernetics: Part C*, 33(2):284–289, 2003.

[4] R. Kumar and V. K. Garg. *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, Boston, MA, 1995.

[5] R. Kumar, S. Takai, M. Fabian, and T. Ushio. Maximally permissive mutually & globally nonblocking supervision with application to switching control. *Automatica*, 41(8):1299–1312, August 2005.

[6] 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.