This paper has been mechanically scanned. Some errors may have been inadvertently introduced.
Sy-Control: A Tool for Syntactic Control in Temporal Logic

Akash Deshpande
Pravin Varaiya

California PATH
University of California, Berkeley

August 1994
This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation; and the United States Department of Transportation, Federal Highway Administration.

The contents of this report reflect the views of the authors who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the State of California. This report does not constitute a standard, specification, or regulation.
Sy-Control: A Tool for Syntactic Control in Temporal Logic *

Akash Deshpande and Pravin Varaiya
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, California 94720
January 15, 1994

Abstract
In this paper we describe Sy-Control, a software tool for syntactic control of discrete event systems. Given a plant behavior and some desired behavior, both described in propositional linear temporal logic (PLTL), a causal and nonblocking controller is to be synthesized that restricts the system’s closed loop behavior to a subset of the desired behavior. We summarize from [1, 2] the syntactic procedure to derive the rule representing the maximal such controller. Then we describe the Sy-Control tool and illustrate its use through a simple example. Because the complexity of Sy-Control is linear in the size of its input, it can be an important addition to the set of DES control tools.

1 Introduction
This paper presents Sy-Control, a software tool for syntactic control of discrete event systems (DES) within the framework of Propositional Linear Temporal Logic (PLTL). Sentences or formulae of PLTL describe the open loop behavior of the system, the control objective, and the closed loop behavior of the system when a control strategy is followed. In the semantic approach [1, 2, 5], these sentences are interpreted as denoting sequences of events generated by finite automata. The syntactic approach is quite different. Here the system behavior and the control objective are represented by uninterpreted PLTL sentences, and one derives the control strategy using syntactic or ‘linguistic’ rules, with no automata-theoretic interpretation.

Figure 1 shows the setup of a controlled DES.Associated to the system is a set $\mathcal{P}$ of propositions.

![Figure 1: Control System Setup](image)

Subsets of $\mathcal{P}$ constitute the events of the system. $\mathcal{P}$ itself is the disjoint union of observations $\mathcal{O}$ and actions $\mathcal{A}$. $\mathcal{O}$ comprises propositions that the controller can observe but cannot directly modify.

---

*Research supported by NSF under grants ECS 9111907 and IRI 9120074, and by the PATH program, University of California, Berkeley.*
while \( \mathcal{A} \) comprises propositions that the controller can observe and whose truth value it can set. The plant is described by a PLTL formula \( \chi \) over \( P \). In the semantic interpretation, \( \chi \) denotes all the (open loop) traces that can be generated by the plant. Only a subset of these traces is generated by the closed loop system depending on how the controller sets controllable propositions based on its observations of the plant. Different controllers will generate different sets of closed loop traces. The objective of control is specified by a PLTL formula \( \psi \) over \( P \) denoting the set of desired traces. The task of the controller is to guarantee that all closed loop traces are desired. Evidently, the largest subset of desired traces that can possibly be generated is denoted by the PLTL formula

\[
\chi \land \psi
\]

where \( \land \) means "and." However, it may not be possible to achieve \( \chi \land \psi \) with a causaland nonblocking controller. In [1, 2], a syntactic procedure was presented for deriving a formula denoting the largest subset of desired traces that can be generated by such a controller given \( \chi \) and \( \psi \).

Section 2 introduces PLTL. Section 3 formulates the notion of control. Section 4 presents the syntactic procedure for controller synthesis, and section 5 describes the Sy-Control software tool using a simple example.

## 2 PLTL Formalism

We introduce the syntax and semantics of PLTL [3].

### 2.1 Syntax

The symbols of PLTL are

1. (names for) a finite set \( P \) of atomic propositions;
2. logical connectives \( \neg \) (not), \( \land \) (and);
3. temporal connectives \( \circ \) (next), \( \cup \) (until).

A well-formed formula (wff) or sentence of PLTL is formed according to these rules (we implicitly use parentheses for unambiguous parsing and white space for formatting):

1. if \( p \in P \) then \( p \) is a wff;
2. (a) iff \( f \) is a wff, then so is \( \neg f \);
   (b) if \( f_1, f_2 \) are wff, then so is \( f_1 \land f_2 \);
3. (a) iff \( f \) is a wff, then so is \( f \circ f \);
   (b) if \( f_1, f_2 \) are wff, then so is \( f_1 \cup f_2 \).

A wff without the temporal connectives \( \circ \) and \( \cup \) is a nontemporal (or boolean) formula; otherwise it is a temporal formula.

Using these rules, a parse tree can be constructed for any wff \( f \) with atomic propositions at the leaf-nodes and logical or temporal connectives at the non-leaf nodes. Nodes of this parse tree themselves represent wffs—these are the subformulae of \( f \).

**Example 1:** Consider the wff \( f = (\neg a \land b) \cup c \). Figure 2 shows the graphical representation of the parse tree of \( f \). The root node of the parse tree (i.e., the node representing \( f \)) is the left-most node in the figure. The tree is read in a depth first manner vertically from top to bottom.

When the formula \( f \) is given as input to Sy-Control or appears as its output, it is written as follows.

---

\(^3\)Since no continuous time semantics can be given for \( \circ \), it cannot be used in describing continuous time systems.
Figure 2: Parse tree of \((-a \land b) \cup c\).

\[ ((\neg a) \land b) \text{ UNTIL } c \]

Sy-Control also shows a graphically formatted tree representation of wffs. The graphical representation of \( f \) given by Sy-Control is as follows.

\[
\begin{align*}
\text{HOT} & \\
\text{AND} & a \\
\text{UNTIL} & b \\
c & 
\end{align*}
\]

Section 5.2 gives further details on the use of Sy-Control.

2.2 Semantics

To define the semantics we need the notions of ‘structure’ and ‘satisfaction of a formula.’ Let \( T \) be \( \mathbb{R}^+ \) or \( \mathbb{Z}^+ \). A structure (or trace—we use both interchangeably) is a map

\[ s : T \to 2^P \]

where \( 2^P \) is the set of all subsets of \( P \). Let \( S \) be the set of all structures. For \( s \in S \) and \( t \in T \), \( s(t) \) is interpreted as the set of propositions that are true at \( t \). For \( r \in T \), the \( r \)-suffix of \( s \), \( s^r \), is the structure given by

\[ s^r(t) = s(t + r), \quad t \geq 0. \]

Satisfaction is the relation between formulae and structures defined by the following rules. The relation is denoted ‘\( s \models f \)’ and ‘\( s \models f \)’ is read as ‘structure \( s \) satisfies wff \( f \).’

1. \( s \models p \) for \( p \in P \) iff \( p \in s(0) \);
2. (a) \( s \models \neg f \) iff it is not the case that \( s \models f \);
   (b) \( s \models f_1 \land f_2 \) iff \( s \models f_1 \) and \( s \models f_2 \);
3. (a) In the case that \( T = \mathbb{Z}^+ \), \( s \models f \) iff \( s^1 \models f \);
   (b) \( s \models f_1 \cup f_2 \) iff \( \exists t \in T \) such that \( s^t \models f_2 \) and \( \forall r \in [0, t). s^r \models f_1 \);

where \( f, f_1, f_2 \) are wff.

The satisfaction relation yields denotational semantics: a wff \( f \) denotes the set \( S_f \) of structures that satisfy it,

\[ S_f = \{ s \mid s \models f \}. \]

\^[2]\text{It is also read as ‘structure } s \text{ models wff } f.\]

\^[3]\text{Thus a PLTL wff } f \text{ denotes a set of traces of events, i.e., a language. The languages denoted by such formulae are the same as those denoted by star-free regular expressions—these are regular expressions that do not involve the Kleene } \ast \text{ operator. Thus the expressive power of PLTL is less than that of regular expressions } [3].\]
2.3 Augmentations

It is customary and convenient to augment the syntax as follows (\(\equiv\) means syntactic equivalence):

1. (a) (or) \(f_1 \vee f_2 \equiv \neg(f_1 \wedge \neg f_2)\);
   
   (b) (implies) \(f_1 \rightarrow f_2 \equiv \neg f_1 \vee f_2\);

2. (a) (eventually) \(\Box T U f\);
   
   (b) (always) \(\Diamond \neg f\);

where \(f, f_1, f_2\) are wff, and \(T \equiv p \lor \neg p\) for any \(p \in P\). Also, let \(F \equiv \neg T\).

We define a new temporal connective \(\mathcal{B}\) (before) with the following syntax and semantics:

- if \(f_1, f_2\) are wff, then so is \(f_1 \mathcal{B} f_2\);
- for a structure \(s, s \models f_1 \mathcal{B} f_2\) iff \(\forall t \in T, st \models \neg f_2\) or \([t = \min\{r \mid s^r \models f_2\} \land \exists r \in t, s^r \models f_1\}\).

Evidently, \(\mathcal{B}\) and \(\mathcal{U}\) are duals in the following sense:

\[-(f_1 \mathcal{U} f_2) \equiv -f_1 \mathcal{B} f_2, \quad -(f_1 \mathcal{B} f_2) \equiv -f_1 \mathcal{U} f_2.\]

2.4 Closure of a Wff

For a wff \(f\), the closure \(\text{cl}(f)\) of \(f\) is a wff given as follows.

1. for \(p \in P\), \(\text{cl}(p) = p\);
2. (a) \(\text{cl}(\neg f) = \neg\text{cl}(f)\);
   
   (b) \(\text{cl}(f_1 \wedge f_2) = \text{cl}(f_1) \wedge \text{cl}(f_2)\);
3. (a) \(\text{cl}(\bigcirc f) = \bigcirc \text{cl}(f)\);
   
   (b) \(\text{cl}(f_1 \mathcal{B} f_2) = (\text{cl}(f_1) \mathcal{B} \text{cl}(f_2)) \vee \bigcirc(\text{cl}(f_1))\),\footnote{This is known as "weak until."}

where \(f, f_1, f_2\) are wff.

2.5 Canonical Form of a Wff

First we recall a few facts about nontemporal (or boolean) logic. An interpretation \(i_P\) of a set \(P\) of propositions assigns truth values to them, i.e., it is a map

\[i_P : P \rightarrow \{T, F\}.\]

A nontemporal formula \(g\) is tautologically true iff

\[\forall i_P(i_P \models g),\]

and \(g\) is tautologically false iff \(\neg g\) is tautologically true. Thus \(p \lor \neg p\) is tautologically true and \(F \wedge \neg p\) is tautologically false.

Every nontemporal formula \(g\) can be expressed in the disjunctive normal form (i.e., as a disjunction of clauses, each clause being a conjunction of atomic propositions or their negations). The nontemporal formula \(g\) in disjunctive normal form is minimal if it is expressed as \(T\) when tautologically true, as \(F\) when tautologically false, or otherwise no clause of \(g\) is tautologically false. This \(p \wedge (\neg a \lor q)\) is expressed as \((p \wedge \neg a) \lor (p \wedge q)\), \(p \lor \neg p\) as \(T\), \(p \wedge \neg p\) as \(F\), and \(q \lor (p \wedge \neg p)\) as \(q\).

A wff of PLTL is in canonical form if

- all nontemporal subformulae are in minimal disjunctive normal form,
- the connectives \(\neg\) and \(\bigcirc\) are "pushed in," and...
all negations of $\mathcal{U}$ or $\mathcal{B}$ have been removed by using the dual forms.

The $\neg$ and $\Diamond$ connectives are pushed in recursively according to the following rules.

1. First all the $\Diamond$ connectives are pushed in as follows.
   
   (a) $\Diamond(f_1 \land f_2)$ is replaced by $\Diamond f_1 \land \Diamond f_2$;
   
   (b) $\Diamond(f_1 \lor f_2)$ is replaced by $\Diamond f_1 \lor \Diamond f_2$;
   
   (c) $\Diamond \neg f$ is replaced by $\neg \Diamond f$.

2. Then all the $\neg$ connectives are pushed in as follows.
   
   (a) $\neg(f_1 \land f_2)$ is replaced by $\neg f_1 \lor \neg f_2$;
   
   (b) $\neg(f_1 \lor f_2)$ is replaced by $\neg f_1 \land \neg f_2$;
   
   (c) $\neg(\neg f)$ is replaced by $f$;
   
   (d) $\neg \Diamond f$ is replaced by $\Diamond \neg f$.

Note that these rules preserve semantic equivalence, and a wff has a unique canonical form.

3 Control

To introduce the notion control one must specify the propositions that the controller can set, and the information available to it.

3.1 Observable and Controllable Propositions

Let $\mathcal{O}$ be a finite set of observable propositions and let $\mathcal{A}$ be a finite set of controllable propositions. $\mathcal{O}$ and $\mathcal{A}$ are disjoint. We consider PLTL with $\mathcal{P} = \mathcal{O} \cup \mathcal{A}$. The following notation will prove convenient.

- Let $Y = 2^O$ denote the set of possible observations, and let $U = 2^A$ denote the set of possible controls.
- Let $y : T \rightarrow Y$ denote the observation structure, so that $y(t)$ is the observation at $t \in T$. Similarly, let $u : T \rightarrow U$ denote the control structure, so that $u(t)$ is the control assignment at $t \in T$. Let $(y, u) : T \rightarrow Y \times U$ be another notation for $s : T \rightarrow 2^P$.

Let $S_Y$ be the set of all observation structures and $S_U$ be the set of all control structures. $S = S_Y \times S_U$.

- Let $U^B = U \cup \text{block}$, where block is a special event available to the controller. Let $S_{UB}$ be the set of all extended control structures $u : T \rightarrow U^B$ such that

  \[ Vu \in S_{UB}, \quad Vt, u(t) = \text{block} \Rightarrow \forall r > t, \ u(r) = \text{block}. \]

- For a structure $s : T \rightarrow 2^P$, let $s_{|[0, t]} : 2^P$ denote the closed prefix of $s$ up to $t$. Similarly define $y_{|t}$ and $u_{|t}$, and let $(y_{|t}, u_{|t})$ be an alternative notation for $s_{|t}$.

- Given a prefix $r_t$, denote the set of structures that extend $r_t$ by $\text{extend}(r_t) = \{ s \mid s_{|t} = r_t \}$.

\[ ^5 \text{Also, we denote set complementation by } (\cdot). \]
3.2 Control Strategy

A control strategy is defined as a point-to-set map from observation traces to extended control traces:

\[ \varepsilon : \mathcal{S} \rightarrow 2^{\mathcal{U} \cup \emptyset}. \]

Thus a controller following \( \varepsilon \), when presented with an observation trace \( y \), produces a control trace in \( \varepsilon(y) \). The graph \( G_\varepsilon \) of \( \varepsilon \) is

\[ G_\varepsilon = \{ (y, u) \mid u \in \varepsilon(y) \} \subseteq \mathcal{S} \times \mathcal{U} \cup \emptyset. \]

The set of closed loop traces generated by \( \varepsilon \) is

\[ S_\varepsilon = G_\varepsilon \cap S. \]

In general, we require control strategies to have desired closed loop traces, and to be causal and nonblocking. In PLTL the desired closed loop traces are denoted by a wff \( f \). Causality depends upon the amount of time by which a controller is permitted to lag the plant. The control strategy \( \varepsilon \) is nonblocking if \( G_\varepsilon \subseteq S_\varepsilon \); otherwise it is blocking. We formalize these notions below.

**Definition 1** Fix \( a > 0 \) and a wff \( f \). A point-to-set map \( \varepsilon_f : \mathcal{S} \rightarrow 2^{\mathcal{U} \cup \emptyset} \) is an a-lag possibly blocking control strategy for \( f \) if

1. \( \varepsilon_f \subseteq \varepsilon_{cl(f)} \),

2. \( \forall r \in \mathbb{R}_+, \forall y, \hat{y} \in \mathcal{S}, y_{t+\alpha} = \hat{y}_{t+\alpha} \Rightarrow \{ u_t \mid u \in \varepsilon_f(\hat{y}) \} = \{ u_t \mid u \in \varepsilon_f(y) \} \), and

3. \( \forall y, \forall u, \forall t, \text{if } \exists t' \in \mathbb{R} \text{ such that } (y', u') \in \varepsilon_f(y, u) \text{ then } \forall u'' \in \mathcal{U} \text{ such that } (y', u'') \in \varepsilon_f(y', u') \text{ for some } \alpha > t \Rightarrow u''(t) \text{ block.} \)

Condition (1) means that \( \varepsilon_f \) guarantees satisfaction of \( cl(f) \). Condition (2) means that the choice of control at time \( t \) can depend on the observations of the plant over \([0, t + \alpha)\) and \( \hat{y} \), the controller lags the plant by time \( \alpha \). Condition (3) means that at each \( t \) if no future evolution after \( t \) satisfies \( cl(f) \), the strategy blocks.

For some index set \( I \), if \( \kappa_i, i \in I \), is an a-lag possibly blocking strategy for \( f \), then so is \( \cup_i \kappa_i \). Hence there is a largest (in the sense of set inclusion) a-lag possibly blocking strategy, \( \bigcup_i \kappa_i \).

If for an a-lag possibly blocking control strategy \( \varepsilon \), \( G_\varepsilon \subseteq S \), then we call \( \varepsilon \) an a-Zag nonblocking control strategy for \( f \), denoted as \( \varepsilon_f \).

For any \( y \) in their domain, \( U_f(y) \) and \( \bar{U}_f(y) \) increase with \( a \). The most interesting strategies are the 0-lag control strategies, defined as

\[ U_f = \cap_{\alpha > 0} U_\alpha, \bar{U}_f = \cap_{\alpha > 0} \bar{U}_\alpha. \]

The strategies \( U_f \) and \( \bar{U}_f \) assume observations of the plant behavior in an infinitesimal future.

In section 4 below we give a syntactic procedure that yields a wff \( \hat{f} \) such that \( cl(\hat{f}) \) denotes \( S_{\hat{f}} \).

4 Syntactic Procedure for Nonblocking Control

We give syntactic rules for deriving from wff \( f \) the formula \( \hat{f} \) such that \( cl(\hat{f}) \) denotes \( S_{\hat{f}} \). In order to do this, we first introduce the notion of controllability of a wff. Then we give the syntactic procedure for deriving \( \hat{f} \).

---

\( ^6 \)Hence \( \kappa_f \) is a "weak" control strategy.
4.1 Controllability of a Wff

We present two definitions of controllability of a \( \text{wff} \)— one syntactic, the other semantic. The syntactic notion is developed further. The following nontemporal example will help motivate the definitions.

**Example 2**: Consider the formula

\[ f = a \lor p \]

where \( a \in \mathcal{A} \) and \( p \in \mathcal{O} \). Suppose the plant has set \( p \) as false; then, in order to satisfy \( f \), the controller must set \( a \) as true. On the other hand, suppose the plant has set \( p \) as true. Then the controller may set \( a \) as it wishes and \( f \) will be satisfied. Thus, under all possible observations, the controller has some action which guarantees that \( f \) can be satisfied. This is our intuitive notion of controllability of a \( \text{wff} \).

Now consider

\[ f' = a \land p. \]

Here, if the plant sets \( p \) as false, there is nothing that the controller can do to satisfy \( f' \). Hence we say that \( f' \) is not controllable. \( \square \)

Given a nontemporal \( \text{wff} \) \( g \) in canonical form over propositions \( P \), derive the \( \text{wff} \) \( g_{\mathcal{O}} \) by retaining only propositions in \( \mathcal{O} \) (i.e., by removing from \( g \) all propositions in \( \mathcal{A} \) along with the associated conjunctions and negations). Thus, for example,

\[ ((p \land \neg q) \lor (p \land q))_{\mathcal{O}} \equiv p \lor (p \land q) \equiv p. \]

A nontemporal formula \( g \) is said to be semantically controllable or \( \text{se-controllable} \) iff

\[ \forall i_{\mathcal{O}} \exists i_{\mathcal{A}} (i_{\mathcal{P}} \models g). \]

Let the nontemporal formula \( g \) be in the minimal disjunctive normal form. Then \( g \) is said to be syntactically controllable or \( \text{sy-controllable} \) iff

1. There exists a clause of \( g \) entirely composed of propositions from \( \mathcal{A} \), or
2. \( g_{\mathcal{O}} \) is tautologically true.

**Fact 1**: For nontemporal formulae, \( \text{se-controllability} \) is equivalent to \( \text{sy-controllability} \).

**Proof**: First we check that the syntactic definition implies the semantic one. If case (1) holds, then let \( c \) be a clause composed entirely of propositions from \( \mathcal{A} \). Because \( c \) is not tautologically false, \( \exists i_{\mathcal{A}} \) with \( i_{\mathcal{A}} \models c \) irrespective of \( i_{\mathcal{O}} \). If case (2) holds, then given any \( i_{\mathcal{O}} \), there exists a clause \( c \) of \( g \) with \( i_{\mathcal{O}} \models c_{\mathcal{O}} \). Hence, for this \( i_{\mathcal{O}} \), \( \exists i_{\mathcal{A}} \) with \( i_{\mathcal{P}} \models c \) since \( c \) is not tautologically false.

To show that the semantic definition implies the syntactic one, suppose for the sake of contradiction that (1) \( \forall i_{\mathcal{O}} \exists i_{\mathcal{A}} (i_{\mathcal{P}} \models g) \), and (2) every clause of \( g \) has a proposition from \( \mathcal{O} \), and (3) \( g_{\mathcal{O}} \) is not tautologically true. From (3), \( \exists i_{\mathcal{O}} \) with \( \neg (\land \neg g_{\mathcal{O}}) \). Further, due to (2), \( \exists i_{\mathcal{A}} (i_{\mathcal{O}}) \) with \( i_{\mathcal{P}} \models g \). This contradicts (1). \( \square \)

Now we wish to extend the syntactic notion of controllability to temporal \( \text{wff} \). Clearly, we should define \( \mathcal{O}g \) to be \( \text{sy-controllable} \) iff \( g \) is \( \text{sy-controllable} \). Controllability of \( g = g_{1} \cup g_{2} \) is more subtle. Consider the special case where \( g_{1} \) and \( g_{2} \) are nontemporal. If neither \( g_{1} \) nor \( g_{2} \) is \( \text{sy-controllable} \), then there is no guarantee that the controller can satisfy \( g \). If \( g_{1} \) is \( \text{sy-controllable} \) but \( g_{2} \) is not, then at each time the controller can first examine whether \( g_{2} \) can be satisfied under the observation. If not, then it can choose an action to satisfy \( g_{1} \) and proceed. However, if it can satisfy \( g_{2} \) and it chooses to do so, it is subsequently free to take any action without jeopardizing the satisfaction of \( g \). Similarly, if \( g_{2} \) is \( \text{sy-controllable} \) while \( g_{1} \) is not, then at each time the controller can examine whether it can satisfy \( g_{1} \) under the observation. If it can and it chooses to do so, then it proceeds. If \( g_{1} \) is not satisfied, then it must satisfy \( g_{2} \), and its task is done. Finally, if \( g_{1} \) and \( g_{2} \) are both \( \text{sy-controllable} \), then at each time the controller can choose to satisfy either \( g_{1} \) or \( g_{2} \). If it chooses to satisfy \( g_{2} \), then
its task is done. We note that the controller uses the controllability of the nontemporal subformulas $g_1$ and $g_2$ along with some notion of history in order to choose its actions. When temporal formulae are nested, this notion of history becomes more elaborate.

With this motivation, we define $g_1 \mathcal{U} g_2$ to be sy-controllable if $g_1 \lor g_2$ is sy-controllable, where $g_1$ and $g_2$ are general wff. This notion of controllability of $g = g_1 \mathcal{U} g_2$ is “weak,” i.e., a controller for sy-controllable $g$ can guarantee closed loop traces in $S_{cl}(g)$.

We summarize below the definition for sy-controllability of any wff.

1. A nontemporal formula $g$ in the minimal disjunctive normal form is sy-controllable iff
   (a) there exists a clause of $g$ entirely composed of propositions from $\mathcal{A}$, or
   (b) $g \equiv \top$ is tautologically true;
2. $\bigvee g$ is sy-controllable iff $g$ is sy-controllable;
3. $g_1 \mathcal{U} g_2$ is sy-controllable iff $g_1 \lor g_2$ is sy-controllable;
4. $g_1 \mathcal{B} g_2$ is sy-controllable iff $g_2$ is sy-controllable.

Henceforth we refer to sy-controllability simply as controllability.

### 4.2 Syntactic Procedure for Nonblocking Control

For a wff $f$, recall that $S_{\mathcal{U}}$ is the set of closed loop traces generated by the 0-lag nonblocking strategy $\mathcal{U}$. However, the definition of $\mathcal{U}$ does not give a constructive procedure for $S_{\mathcal{U}}$. We now give a three-step syntactic procedure for generating a wff $\tilde{f}$. Informally, $\tilde{f}$ denotes the causal, controllable and nonblocking subset of $S_f$.

Assume that $f$ is in the canonical form (see section 2.5).

1. First, top-down over all subformulae $g$ of $f$ where $g = g_1 \mathcal{B} g_2$, $g$ is replaced by
   (a) if $g_1 \mathcal{B} g_2$ is not controllable, then $g_1 \land \neg g_2$;
   (b) else, itself.

   Let the formula so obtained be $f'$. Note that negations of $\mathcal{U}$ or $\mathcal{B}$ connectives may appear in $f'$ after this step. These are replaced by their equivalent dual forms so that $f'$ is again in the canonical form.

2. Then, bottom-up over all subformulae $g$ of $f'$
   (a) if the main connective of $g$ is nontemporal then $g$ is replaced by itself;
   (b) a temporal formula $g = \bigvee g'$ is replaced by
      i. if $g'$ is controllable, then itself;
      ii. else $\top$;
   (c) for temporal formula $g = g_1 \mathcal{U} g_2$,
      i. $g_1$ is replaced by
         A. if the main connective of $g_1$ is not $\mathcal{U}$, then itself;
         B. else if $g_1 = h_1 \mathcal{B} h_2$ is not controllable, then $h_2$;
         C. else itself.
      ii. $g_2$ is replaced by
         A. if the main connective of $g_2$ is not $\mathcal{U}$, then itself;
         B. else $g_2 = h_1 \mathcal{B} h_2$
            o if $h_1$ is not controllable, then $h_2$;
            o else itself.
iii. if the formula $g' \cup g''$ obtained as above is not controllable, then $g$ is replaced by $g''$.

Let the formula so obtained be $f''$.

3. Now apply the definition of controllability to $f''$. If $f''$ is controllable, $\bar{f}$ is $f''$; else, $\bar{f}$ is $F$.

In [1, 2] it is shown that $c(\bar{f})$ denotes $S_{01}$, i.e.,

$$S_{01} = S_{c(\bar{f})}.$$  

The strategy for the proof of this equality was to construct a finite state causal and nonblocking controller in the domain of semantic interpretation. Then it was shown that the syntactic procedure given above follows exactly the construction of this controller.

## 5 Discrete Event System Control using Sy-Control

Recall that the plant is described by a wff $\chi$ and the specification is given as a wff $t)$. Define the wff

$$f = \chi \rightarrow \chi \wedge \psi.$$  

The traces in $S_{\chi}$ which are allowed by the plant and which also satisfy the specification are precisely denoted by $\chi \wedge t)$. Let $\bar{f}$ be the wff obtained from $f$ using the syntactic procedure given above. The wff

$$\bar{f} \wedge \chi \wedge \psi$$  

denotes traces which are not only in $S_{\chi}$—i.e., they are generated by some causal and nonblocking control strategy—but which are also allowed by the plant and which satisfy the specification.

**Definition 2** The system $\chi$ is said to be controllable to specification $t)$ iff the wff $\bar{f} \wedge \chi \wedge \psi$ is not tautologically false. If $\chi$ is controllable to $t)$ then $\bar{f} \wedge \chi \wedge \psi$ is said to be the rule for a control strategy for $\chi$ and $t)$.

Since PLTL is decidable, standard proof systems (see [4]) can be used to determine whether a wff is tautologically false. Sy-Control implements the syntactic procedure given in Section 4 to derive $\bar{f}$ given a wff $f$. In section 5.1 we illustrate the syntactic procedure given above using a simple example. In section 5.2 we explain Sy-Control and use it to derive the rule for the control strategy for the same example.

### 5.1 Using the Syntactic Procedure to Derive a Rule for DES Control

Consider a hungry engineer with an uncooked potato, a pot of water, and a heater. He wishes to eat a soft potato. His relevant knowledge is described by the following two sentences. If the pot is heated, eventually the water will boil; if it continues to be heated, the water will continue to boil. If the potato is kept in boiling water, eventually it will turn soft; once it is soft, the potato stays soft.

The engineer's problem is to obtain a recipe—the control strategy—for making a soft potato. In the semantic approach to his problem, the engineer first captures his knowledge in an automaton involving controllable events (the heater is on or not), and uncontrollable but observable events (the water is boiling or not, the potato is soft or not). He then formulates his objective as the set of all traces or sequences of events which lead to the desired outcome (potato is soft). Finally, he derives the recipe—when to turn the heater on and off so that when the recipe is followed, only the desired outcome will emerge. In the syntactic approach, the engineer will use 'linguistic' rules to obtain the recipe, 'heat the water until the potato is soft.'

The sets of propositions are

$$P = \{h, b, s\}, \mathcal{O} = \{b, s\}, \mathcal{A} = \{h\}$$
with the pragmatics h—the heater is on, \( b \)—the water is boiling, and \( s \)—the potato is soft; \( \neg \) h means the heater is not on, etc. By definition of \( A \) and \( \Box \), the engineer can set \( h \) to ‘true’ or ‘false’ at any time, but the plant sets (the truth values of) \( b \) and \( s \).

The engineer’s knowledge can be captured in a PLTL formula \( \chi \), or by the automaton whose transition graph is drawn in Figure 3. An edge of the graph is a possible state transition and its label lists the events which enable that transition. Each event is a subset of \( P \) so that the graph is read as follows. The edge from Start to Water Boiling is labeled by two events \{b\} and \{h, b\} which means that this transition can occur under the event \{b\} (which is the same as \{lh, b, ls\} i.e., heater not on, water boiling, and potato not soft) or under the event \{h, b\} (i.e., heater on, water boiling, and potato not soft). (The event \( 0 \) is the same as \{lh, lb, ls\}.) Although it is not the case in this particular graph, two or more transitions from the same state can have the same event label, so that the automaton may be non-deterministic. Once we know how to read the figure we can calculate the traces that the automaton can generate beginning in the Start state.

These traces are also denoted by the PLTL formula

\[
\chi = hU(hU(s)) \lor (h A (\neg b V s)) \lor (h A (\neg b V s)) \lor (h A (b A (\neg s U (\Box (\neg (h V b V s))))))
\]

The subformula \( hU(hU(s)) \) denotes traces that move the automaton from Start to Potato Soft, the subformula \( (h A (\neg b V s)) \lor (h A (\neg b V s)) \) denotes traces from Start to Restart without going through Water Boiling, and the subformula \( (h A (b A (\neg s U (\Box (\neg (h V b V s)))))) \) denotes traces from Start to Restart through Water Boiling.

The desired traces are those that end up with a soft potato. These traces are denoted by the PLTL formula

\[
\psi = \Box s.
\]

The engineer must set the controllable proposition \( h \) at each time so that only traces in \( \chi \land \psi \) are generated.

Consider a control strategy in which the engineer keeps the heater on until the water starts boiling but then turns the heater off before the potato is soft. One resulting closed loop trace is that the water continues to boil and the potato eventually turns soft. (This may happen because of thermal inertia of the water or pot.) Another possible closed loop trace is that the water stops boiling and the potato does not turn soft. The first trace is desired while the second one is not, but the engineer cannot predict which trace will result. It is intuitively clear that in order to guarantee...
desired traces, the engineer has to keep the heater on until the potato is soft. This is denoted by the PLTL formula

\[ hU s. \]

It is reasonable to regard \( hU s \), or “heat until soft,” as a rule for a control strategy since it is causal and nonblocking.

Let us use the syntactic procedure to check whether the plant is controllable to the given specification.

\[
\chi = hU(bU s) \lor \left[ hA \neg (b \lor s) \lor ((h \lor b \lor s) \lor (h \lor s)) \lor (h \lor b \lor s) \right] \lor \left[ hU(bU s) \lor (h \lor b \lor s) \lor (h \lor s) \right]
\]

\[
= hU(bU s) \lor (h \lor b \lor s) U (h \lor b \lor s) \lor (h \lor b \lor s)
\]

\[
\psi = bU s.
\]

From this, we form

\[
f = \chi \land \psi \equiv \neg x \lor (\chi \land \psi)
\]

\[
\equiv \neg hB(bU s) \lor (h \lor b \lor s) B (h \lor b \lor s) \lor (h \lor b \lor s) \lor (h \lor b \lor s)
\]

After the top-down portion of the syntactic procedure, we obtain the intermediate wff \( f' \) and after the bottom-up portion of the syntactic procedure, we obtain \( f'' \). These are shown below.

\[
f' = \{ [\neg h \land b \land s] A \}
\]

\[
\equiv \neg hA \land bA \land s A (T U s)
\]

Since \( f'' \) is controllable, we obtain

\[
f = (\neg hA \land bA \land s) A (T U s).
\]

Forming \( f A \chi A \psi \) and using standard inference schemas for PLTL [4], we get \( hU s \) as the rule for the control strategy that the engineer should follow.
5.2 Using Sy-Control to Derive a Rule for DES Control

Sy-Control is a software tool for syntactic controller synthesis. It is written in the C programming language, and it uses the Unix programs lex for lexical analysis and yacc for parser generation. Sy-Control works in the following stages.

1. Parse input and build internal data structures. The input consists of $\mathcal{O}, \mathcal{A}, \mathcal{X}$ and $\psi$.
2. Form $f = \mathcal{X} \rightarrow \mathcal{X} \land \psi$, represent $f$ in canonical form and mark the controllability of each subformula of $f$.
3. Follow the top-down step of the syntactic procedure.
4. Follow the bottom-up step of the syntactic procedure.
5. Obtain $\tilde{f}$ and form $\tilde{f} \land \mathcal{X} \land \psi$.

Sections 5.2.1 and 5.2.2 describe the input and output of Sy-Control respectively, and section 5.2.3 gives the complexity of the Sy-Control algorithm.

5.2.1 The Input

The inputs $\mathcal{O}, \mathcal{A}, \mathcal{X}$ and $\psi$ are identified by the keywords OBSERVATIONS, ACTIONS, PLANT and SPECIFICATION, respectively, and must be given in that order. The keyword OBSERVATIONS is followed by a (possibly empty) white space separated list of observable propositions. Similarly, the keyword ACTIONS is followed by a list of actions. The propositions can be formed using any combination of a-z, 0-9 and '. The keyword PLANT is followed by the plant formula and the keyword SPECIFICATION is followed by the specification formula. These wffs can be formed using the propositions declared previously, the symbols T and F denoting T and F respectively, and the keywords BOT, AND, OR, IMPLIES, NEXT, UNTIL, BEFORE, EVENTUALLY and ALWAYS following the PLTL syntax given in section 2.1.

The input to Sy-Control for the example described in section 5.1 is shown in Figure 4.

5.2.2 The Output

First Sy-Control prints out the propositions, the plant formula and the specification formula parsed from its input. We omit these. We show below the intermediate output after the topdown step and after the bottom-up step of the syntactic procedure. In the graphical representation, the atomic propositions are marked as observations (O) or as actions (A), and the subformulae are marked as controllable (C) or as not controllable (NC). Amongst nontemporal subformulae, controllability is marked only for the largest ones.

The formula $f'$ obtained after the topdown step of the syntactic procedure is shown in Figure 5.

The formula $f''$ obtained after the bottom-up step of the syntactic procedure is shown in Figure 6.

Next, Sy-Control prints out $\tilde{f}$ which we omit since it is identical to the bottom-up formulashown above. Finally Sy-Control prints out $\tilde{f} \land \mathcal{X} \land \psi$, the rule for the control strategy which we also omit.

We note that Sy-Control does not implement a PLTL proof system.

5.2.3 Performance

Let the length of a wff $f$ be denoted by $|f|$. Then the time as well as the space complexity of Sy-Control is linear in $|\mathcal{X}| + |\psi|$. This is because in each step of the algorithm, each node of the parse tree is visited only once, and the size of the parse tree is the same as the length of the corresponding wff. Semantic techniques for DES controller synthesis are usually susceptible to state-space explosion.
6 Conclusion

In this paper we described Sy-Control, a software tool for syntactic control of discrete event systems. The control problem was formulated as follows. Given a plant behavior and some desired behavior, both described in propositional linear temporal logic (PLTL), a causal and nonblocking controller is to be synthesised that restricts the system’s closed loop behavior to a subset of the desired behavior. We summarized from [1, 2] the syntactic procedure to derive the rule representing the maximal such controller. Then we described the Sy-Control tool and illustrated its use through a simple example. Traditionally, since semantic techniques are often susceptible to state-space explosion, syntactic techniques have been used to prune the systems under consideration before semantic techniques are applied. Because the complexity of Sy-Control is linear in the size of its input, it is an ideal candidate for such use and can prove to be an important addition to the set of DES control tools.

References


----- top down -----

\(((\neg h)) \land (\neg(s)) \land (\neg(b)) \lor \n(((h) \until ((b) \until (s)) \land ((T) \until (s))))\\

Graphical Representation of the Top Down Formula-----

```
\begin{verbatim}
  NOT
  AND
    NOT
    s (0)
    AND (NC)
    NOT
    b (0)
    OR (C)
    h (A)
    UNTIL (C)
      b (0)
      UNTIL (NC)
        s (0)
    AND (C)
      T
    UNTIL (C)
      s (0)
\end{verbatim}
```

Figure 5: Intermediate output of Sy-Control after the topdown step

----- bottom up -----

\(((\neg h)) \land (\neg(s)) \land (\neg(b)) \lor \n(((h) \until (s)) \land ((T) \until (s))))\\

Graphical Representation of the Bottom Up Formula-----

```
\begin{verbatim}
  NOT
  AND
    NOT
    s (0)
    AND (NC)
    NOT
    b (0)
    OR (C)
    h (A)
    UNTIL (C)
      s (0)
      AND (C)
        T
      UNTIL (C)
        s (0)
\end{verbatim}
```

Figure 6: Intermediate output of Sy-Control after the bottom-up step