2008 American Control Conference
Westin Seattle Hotel, Seattle, Washington, USA
June 11-13, 2008

WeA14.3

Supervisory Control of Timed State Tree Structures

A. Saadatpoor, C. Ma and W. M. Wonham

Abstract— **It is well known that the nonblocking supervisory control problem is NP-hard, subject in particular to state space explosion that is exponential in the number of system components. The problem of state explosion is more challenging in Timed DES than in untimed DES. In this paper we propose to manage complexity by organizing the system as a Timed State Tree Structure (TSTS). Based on TSTS we present an efficient recursive algorithm that can perform nonblocking supervisory control design (in reasonable time and memory) for systems of state size 10^{12} and higher.**

I. INTRODUCTION

In the last two decades, Discrete Event Systems (DES) have been studied by researchers from different fields, with respect to modeling, analysis and control. Several models have been proposed and investigated. These models can be classified as *untimed* DES models and *timed* DES models [1]. In an untimed model, when considering the state evolution, only the sequence of states visited is of concern. In a timed model, both logical behavior and timing information are considered. The main problem of supervisory control theory (SCT) is that of optimal nonblocking supervisory control [1]. This problem is well known to be NP-hard [2]. This means that realistic industrial problems (say with state set sizes of 10^{20} and higher), if formulated naively, may well exceed the computational capacity available. It is therefore attractive to explore structured system architectures with the property that, if the given system can be modelled in the selected framework, the required computations can be carried out with greater efficiency. In addition, structured modelling may confer advantages of model transparency and modifiability. One instance is Leduc's [3] Hierarchical Interface-based Supervisory Control theory (HISC).
Statecharts [4] offer a compact representation of hierarchy and concurrency in finite state machines (FSM). Here the system state set is structured top-down into successive layers of cartesian products (AND superstates) alternating with disjoint unions (OR superstates). On this basis, Wang [5] introduced State Tree Structures (STS) consisting of a hierarchical state space or State Tree, equipped with dynamic modules called holons . Gohari [6] formalized Wang's model in linguistic terms. In [5], however, AND states had to be converted by synchronous product of factors into OR states at a higher level before computations could effectively be carried out; and [6] was similarly restricted to a purely OR state expansion. By contrast, in [7] both AND and OR states are treated on an equal footing, and AND states allow shared events among the factors. Timed Statecharts were introduced by Kesten and Pnueli [8]. They extend the traditional statecharts by specifying time bounds for execution of transitions. The semantics is defined with reference to a dense (real) time domain. In this work we use *tick* as in [9], to represent the 'tick' of a global clock.
Borrowing from symbolic model checking [10] we employ binary decision diagrams [11] (BDDs). BDD is a well-known computational representation of the state set, a directed graph representation of propositional logic formulas. A BDD is appealing because, for a fixed order of arguments (variables), it is a canonical form, and makes such tasks as testing for set membership or set equality highly efficient. After representing sets by BDD, the computational complexity of our synthesis is no longer polynomial in the model's state size, but in the number of nodes in the relevant BDD. As with other approaches to solving NP-hard problems, in the worst case the number of BDD nodes ($|nodes|$) grows exponentially in the number of variables and, therefore, is comparable to the number of states ($|states|$) of the set it represents. However, with TSTS, we propose an efficient encoding scheme such that in many cases $|nodes| \ll |states|$. In [12], [7], a complete symbolic approach for the optimal supervisor design of untimed STS was proposed. In [13], BDDs were used for Timed DES but the scope was limited to flat system models. The aim of this paper is to extend the STS framework of [12] to timed models. We use definitions from [12] , to which readers are referred for further details. The paper is organized as follows. Section II introduces the state-based framework of TSTS. Section III presents the symbolic representation of TSTS. In Section IV controllability and state feedback control are defined for TSTS. Section V introduces the symbolic synthesis algorithms for TSTS. We report on a complex example in Section VI, and provide conclusions in Section VII.

II. TIMED STATE TREE STRUCTURES

Let X be a finite collection of finite sets, called *states*. Let $x \in X$ and $Y = \{x_1, x_2, ..., x_n\} \subset X$, $x \notin Y$. If x can be represented by the union (Cartesian product) of states in Y , we call x an *OR (AND)* superstate and each x_i an OR (AND) component of x. Also call x a parent of any x_i and x_i a child of x. All other states in X are called simple states. Say X is a *structured state set*. Formally, we define $\mathcal{M} : X \rightarrow \{and, or, simple\}$ as the *mode* function and $\mathcal{E} : X \rightarrow 2^X$ as the *expansion* function, with $\mathcal{E}(x) := \begin{cases} Y, & \text{if } \mathcal{M}(x) \in \{and, or\}; \\ \emptyset, & \text{if } \mathcal{M}(x) = simple. \end{cases}$
, where $\emptyset$ denotes the empty set. Let $R \subset X$. Write $\mathcal{M}_R : R \rightarrow \{and, or, simple\}$ as the restriction of $\mathcal{M}$ to R, and $\mathcal{E}_R : R \rightarrow 2^R$ as the restriction of $\mathcal{E}$. The *reflexive* and *transitive closure* of $\mathcal{E}$ is written $\mathcal{E}^*$. That is, $\mathcal{E}^*(x) - \{x\}$

is the set of all *descendants* of x, while x is an *ancestor* of states in $\mathcal{E}^*(x) - \{x\}$. Now we can define the *state tree* by recursion.

Definition 1 (State Tree) A *state tree* is a 4-tuple $(X, x_o, \mathcal{M}, \mathcal{E})$, where X is a finite structured state set with $X = \mathcal{E}^*(x_o)$ and $x_o \in X$ is the root state.

$\mathbf{ST} = (X, x_o, \mathcal{M}, \mathcal{E})$ is a state tree if

1. (terminal case) $X = x_o$, or

2.(recursive case) $(\forall x_i \in \mathcal{E}(x_0))\mathbf{ST}^{x_i} = (\mathcal{E}^*(x_i), x_i, \mathcal{M}_{\mathcal{E}^*(x_i)}, \mathcal{E}_{\mathcal{E}^*(x_i)})$ is also a state tree, where $(\forall x_i, x_j \in \mathcal{E}(x_0), x_i \neq x_j)\mathcal{E}^*(x_i) \cap \mathcal{E}^*(x_j) = \emptyset$ and $\bigcup_{x_i \in \mathcal{E}(x_0)} \mathcal{E}^*(x_i) = X - \{x_0\}$. Say $\mathbf{ST}^{x_i}$ is a child state tree of x_o in $\mathbf{ST}$, rooted by x_i.

A *well-defined* state tree must also satisfy $(\forall x \in X)\mathcal{M}(x) = and$ & $x_i \in \mathcal{E}(x) \Rightarrow \mathcal{M}(x_i) =$ *or*. That is, all AND

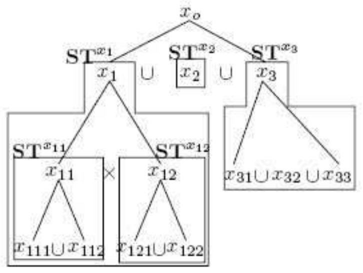

Fig. 1. Recursive definition of state tree

components must be OR superstates. We assume all state trees are well-defined. In addition, we declare that $\mathbf{ST}$ is the *empty state tree* if $X = \emptyset$, and write $\mathbf{ST} = \emptyset$. An example is shown in Fig 1.

Definition 2 (Sub State Tree) Let $\mathbf{ST} = (X, x_o, \mathcal{M}, \mathcal{E})$ be a state tree, and let $Y \subseteq X$. $\mathbf{subST} = (Y, x_o, \mathcal{M}_Y, \mathcal{E}')$ is a *sub state tree* of $\mathbf{ST}$ if $\mathbf{subST}$ is an empty state tree $\emptyset$ or a well-defined state tree with $\mathcal{E}' : Y \rightarrow 2^Y$ defined by

$$\begin{cases} \mathcal{E}'(y) := \mathcal{E}(y), & \text{if } \mathcal{M}_Y(y) \in \{and, simple\}; \\ \emptyset \subset \mathcal{E}'(y) \subseteq \mathcal{E}(y), & \text{if } \mathcal{M}_Y(y) = or. \end{cases}$$

for all $y \in Y$. Trivially, $\mathbf{ST}$ itself is a sub state tree of $\mathbf{ST}$. Denote by $\mathcal{ST}(\mathbf{ST})$ the set of all sub state trees of $\mathbf{ST}$. Let $\mathbf{ST_1}, \mathbf{ST_2} \in \mathcal{ST}(\mathbf{ST})$. Define $\mathbf{ST_1} \leq \mathbf{ST_2}$ iff $\mathbf{ST_1} \in \mathcal{ST}(\mathbf{ST_2})$. To measure the size of $\mathcal{ST}(\mathbf{ST})$, we use the function *count*, defined recursively along the sub state tree $\mathbf{ST}_1 \in \mathcal{ST}(\mathbf{ST})$ such that: count($\mathbf{ST_1}$) =

$$\begin{cases} \prod_{\forall x_i \in \mathcal{E}(x_0)} count(\mathbf{ST}_1^{x_i}), & \text{if } \mathcal{M}(x_0) = and \\ \sum_{\forall x_i \in \mathcal{E}(x_0)} count(\mathbf{ST}_1^{x_i}), & \text{if } \mathcal{M}(x_0) = or \\ 1, & \text{if } \mathcal{M}(x_0) = simple \end{cases}$$

Trivially, count($\emptyset$) $= 0$. Say $\mathbf{subST}$ is a *basic* sub state tree of $\mathbf{ST}$ if count($\mathbf{subST}$) $= 1$. Write $\mathcal{B}(\mathbf{ST})$ for the set of all basic sub state trees of $\mathbf{ST}$. A basic sub state tree is the "smallest" nonempty element in $\mathcal{ST}(\mathbf{ST})$. It is equivalent to a state of FSM in describing system behavior. A simple way of defining the STS behavior is to assign transitions to each element in $\mathcal{B}(\mathbf{ST})$. However, the size of $\mathcal{B}(\mathbf{ST})$ can be so large for complex systems that the assignment may be infeasible to carry out. Instead, we introduce the concept of *timed holon*, the local behavior, and then build the global behavior structurally.

Definition 3 (Timed Holon) A Timed Holon H is a 5-tuple $H := (X, \Sigma, \delta, X_o, X_m)$, where X, the finite state set, is the disjoint union of *external state set* X_E and *internal state set* X_I; and Σ, the event set, is the disjoint union of *boundary event set* Σ_B and *internal event set* Σ_I . The event *tick* which represents the 'tick' of the global clock is included in Σ. The transition structure $\delta : X \times \Sigma \rightarrow X$ is a partial function; it is the disjoint union of the *internal transition structure* $\delta_I : X_I \times \Sigma_I \rightarrow X_I$ and the *boundary transition structure* δ_B; δ_B is again the disjoint union of two transition structures :

- $\delta_{BI} : X_E \times \Sigma_B \rightarrow X_I$ (incoming boundary transitions)
- $\delta_{BO} : X_I \times \Sigma_B \rightarrow X_E$ (outgoing boundary transitions)

Write $\delta(x, \sigma)!$ if $\delta(x, \sigma)$ is defined. We require the transition structure to be deterministic. $X_o \subseteq X_I$ is the *initial state set*, where X_o has only the target states of incoming boundary transitions if δ_{BI} is defined. Otherwise X_o is a selected nonempty subset of X_I. From now on , write $\delta_{BI} : X_E \times \Sigma_B \rightarrow X_o$ (pfn). $X_m \subseteq X_I$ is the *marker* state set, where X_m has only the source states of the outgoing boundary transitions if δ_{BO} is defined. Otherwise X_m is a selected nonempty subset of X_I. Write $\delta_{BO} : X_m \times \Sigma_B \rightarrow X_E$ (pfn).

Note that an event in Σ can be controllable or uncontrollable or the *tick* event. ($\Sigma = \Sigma_u \cup \Sigma_c \cup \{tick\}$). We have also a subset of events called *forcible* events $\Sigma_{for} \subset \Sigma_u \cup \Sigma_c$. A forcible event is one that can preempt a *tick* of the clock (Chapter 9 of [1]).

To complete the general definition of timed holon we impose a final technical condition, to exclude the physically unrealistic possibility that a *tick* transition might be preempted indefinitely by repeated execution of an activity loop within a fixed unit time interval. A timed holon is said to have an *activity loop* if $(\exists x \in X)(\exists s \in (\Sigma - \{tick\})^+)\delta(x, s) = x$. ($\Sigma^+$ denotes the set of all finite symbol sequences, of the form $\sigma_1\sigma_2...\sigma_k$ where $k \geq 1$ and $\sigma_i \in \Sigma$.)

We declare that all timed holons must be *activity-loop-free*, namely $(\forall x \in X)(\forall s \in (\Sigma - \{tick\})^+)\delta(x, s) \neq x$.

Note that a timed holon never *stops the clock*, meaning that at any state some transition $\sigma \in \Sigma$ is eligible which can be a *tick* or some other event.

An example of a timed holon is displayed in Fig. 2. In this example $X_I = \{0, 1, 2\}, X_E = \{3, 4, 5\}, \Sigma_B = \{d, e, f\}, \Sigma_I = \{a, c, tick\}, X_0 = \{0, 2\}$ and $X_m = \{1\}$.

For each OR superstate y in the state tree $\mathbf{ST} = (X, x_o, \mathcal{M}, \mathcal{E})$, a timed holon $H^y = (X_E^y \cup X_I^y, \Sigma^y, \delta^y, X_o^y, X_m^y)$ is said to be *matched* to y if

. internal structure matches, i.e., $X_I^y = \mathcal{E}(y)$.

. external structure matches. Let x be the nearest OR ancestor of y on $\mathbf{ST}$, i.e., $x < y$ and $\mathcal{M}(x) = or$. Then

$$\begin{cases} X_E^y = \emptyset, & \text{if } x \text{ does not exist} \\ X_E^y \subset \mathcal{E}(x), & \text{if } x \text{ exists} \end{cases} .$$

Suppose a timed holon H^x is also matched to the OR superstate x. We say H^x is the *parent timed holon* of H^y

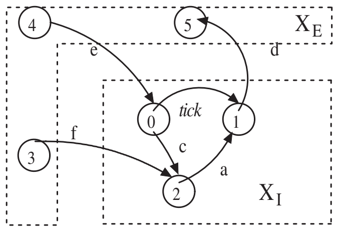

Fig. 2. An example of a Timed Holon

and H^y the *child timed holon* of H^x. Notice that if x exists, $X_E^y \subset \mathcal{E}(x)$ implies that the superstate y cannot be a boundary state in $(X_o^x \cup X_m^x)$, i.e., all boundary states of matched timed holons must be simple states. Matched timed holons limit vertical communication to be only between parent/child timed holons.

Definition 4 (Boundary Consistency) Let $H^x = (X^x, \Sigma^x, \delta^x, X_o^x, X_m^x)$ and $H^y = (X^y, \Sigma^y, \delta^y, X_o^y, X_m^y)$ be the timed holons matched to x and y, respectively. H^x is the parent timed holon of H^y. As illustrated in Fig. 3, there are only two possible cases

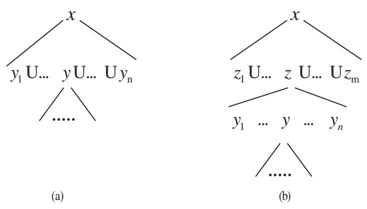

Fig. 3. x and y relation

1. $y \in \mathcal{E}(x) = X_I^x$ as in case (a) of Fig. 3, or
2. $(\exists z, \mathcal{M}(z) = and) y \in \mathcal{E}(z) \& z \in \mathcal{E}(x) = X_I^x$ as in (b).

In both cases, there is exactly one *representative superstate* of y in X_I^x . Denote the representative superstate $\hat{y}$ by

$$\hat{y} = \begin{cases} y, & \text{if } y \in X_I^x \\ z, & \text{if } y \in \mathcal{E}(z) \text{ and } z \in X_I^x \end{cases}.$$

Then the pair (H^x, H^y) is *boundary consistent* if

- (State consistency) The external states of H^y are those connected with the superstate $\hat{y}$ at H^x.
- (Event consistency) The boundary events of H^y are internal events of H^x, i.e., $\Sigma_B^y \subseteq \Sigma_I^x$. More precisely, the boundary events are those events which point to or leave the superstate $\hat{y}$ at H^x.
- (Boundary transition consistency) The incoming/outgoing boundary transitions of H^y are consistent with those of the superstate $\hat{y}$ at H^x.

Definition 5 (Timed State Tree Structure (TSTS)) A timed state tree structure (TSTS) is a 6-tuple $(\mathbf{ST}, \mathcal{H}, \Sigma, \Delta, \mathbf{ST_o}, \mathcal{ST}_m)$, where

- $\mathbf{ST} := (X, xo, \mathcal{M}, \mathcal{E})$ is a state tree;
- $\mathcal{H} := \{H^a | a \in X \& \mathcal{M}(a) = or\}$ is the set of timed holons assigned to the OR superstates in ST;
- Σ is the set of events occurring in $\mathcal{H}$;
- $\Delta : \mathcal{ST}(\mathbf{ST}) \times \Sigma \to \mathcal{ST}(\mathbf{ST})$ is the transition function;
- $\mathbf{ST}_o \in \mathcal{ST}(\mathbf{ST})$ is the initial state tree;
- $\mathcal{ST}_m \subseteq \mathcal{ST}(\mathbf{ST})$ is the marker state tree set.

$\mathbf{G} = (\mathbf{ST}, \mathcal{H}, \Sigma, \Delta, \mathbf{ST_o}, \mathcal{ST}_m)$ is a timed state tree structure if

1. (Boundary consistency) all parent-child pairs in $\mathcal{H}$ are boundary consistent, and
2. (No-Activity coupling) No events of an inner transition structure except *tick* can be shared among those timed holons matched to the children of an AND superstate. Formally, for all superstates $a \neq b$ with matching timed holons $H^a, H^b \in \mathcal{H}$, we require $\Sigma_I^a \cap \Sigma_I^b = \{tick\}$.

It has been shown by an example in [1] that if two timed holons have shared events other than *tick*, then their synchronous product may stop the clock, i.e. there may exist some states in their synchronous product where no events are defined. It has also been shown [1] that if two timed holons have no shared events other than *tick*, then their synchronous product will not stop the clock. So the no-activity coupling condition guarantees the following proposition.

Proposition 1 A TSTS never stops the clock. □

A simple example is shown in Fig. 4. We use the graphical notation of statecharts (see [4]) to draw our TSTS model.

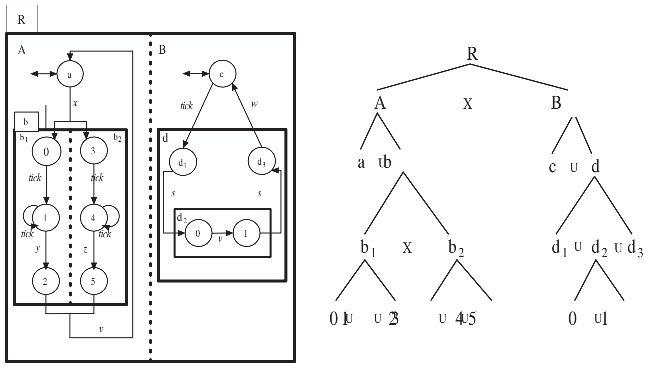

Fig. 4. TSTS model and its State Tree

The dynamics of TSTS, given by a function Δ, is defined as follows.

Definition 6 (Δ) Let $\mathbf{ST}_1 \in \mathcal{ST}(\mathbf{ST})$ and $\sigma \in \Sigma$. Define the total function

$$\Delta(\mathbf{ST}_1, \sigma) := ReplaceSource_{\mathbf{G},\sigma}(\mathbf{ST}_1 \wedge Elig_{\mathbf{G}}(\sigma))$$

$Elig_{\mathbf{G}}(\sigma) \in \mathcal{ST}(\mathbf{ST})$ is the largest sub state tree of $\mathbf{ST}$ that allows σ to happen. $\mathbf{ST}_1 \wedge Elig_{\mathbf{G}}(\sigma)$ is also a sub state tree because $(\mathcal{ST}(\mathbf{ST}), \leq)$ is a lattice. If p is any source state on the tree $\mathbf{ST}_1 \wedge Elig_{\mathbf{G}}(\sigma)$ such that $\delta^x(p, \sigma)!$ for some timed holon H^x, the function $ReplaceSource_{\mathbf{G},\sigma}(.)$ just replaces p by its target state $\delta^x(p, \sigma)$ to yield another sub state tree. In other words, $\Delta(\mathbf{ST}_1, \sigma)$ is the largest sub state tree (wrt $\leq$) in which the system could reside if event σ occurred at $\mathbf{ST}_1$. Because in general count$(\mathbf{ST}_1) \geq 1$, one can look on $\mathbf{ST}_1$ as the symbol of $\mathcal{B}(\mathbf{ST}_1)$, its set of basic sub state trees. So $\delta^x(p, \sigma)$ embodies all transitions labelled by σ in the set $\mathcal{B}(\mathbf{ST}_1)$, i.e., our Δ function is computationally more efficient than the δ function of FSM, which computes only one transition at a time. An example of the Δ function for the TSTS in Fig 4 is given in Fig 5. In this example, $\Delta(\mathbf{ST}_1, tick)$ embodies all transitions labelled by *tick* event in the set $\mathcal{B}(\mathbf{ST}_1)$, i.e. $\delta_I^{b_1}(0, tick) = 1$, $\delta_I^{b_2}(3, tick) = 4$ and $\delta_{BI}^{d}(c, tick) = d_1$.

Following a dual route, we can define the function Γ, which

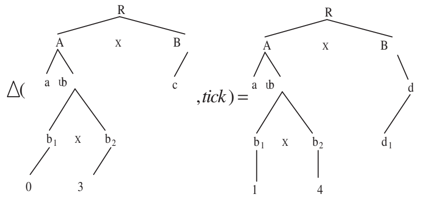

Fig. 5. $\Delta(\mathbf{ST}_1, tick)$

is more important for the synthesis of TSTS.

Definition 7 (Γ) *Backward Transition Function* Let $\mathbf{ST}_1 \in \mathcal{ST}(\mathbf{ST})$ and $\sigma \in \Sigma$. Define

$$\Gamma(\mathbf{ST}_1, \sigma) := ReplaceTarget_{\mathbf{G},\sigma}(\mathbf{ST}_1 \wedge Next_{\mathbf{G}}(\sigma))$$

$Next_{\mathbf{G}}(\sigma) = \Delta(\mathbf{ST}, \sigma)$ is the largest sub state tree of $\mathbf{ST}$ that the event σ is targeting. The function $ReplaceTarget_{\mathbf{G},\sigma}(.)$ just replaces target states with source states to obtain a new sub state tree. Thus, $\Gamma(\mathbf{ST}_1, \sigma)$ is the largest sub state tree of $\mathbf{ST}$ that could reach a sub state tree of $\mathbf{ST}_1$ if event σ occurs. Normally $\Gamma(\Delta(\mathbf{ST}_1, \sigma), \sigma) \neq \mathbf{ST}_1$.

III. Symbolic Representation of TSTS

A. State Space

Let $\mathbf{G} = (\mathbf{ST}, \mathcal{H}, \Sigma, \Delta, \mathbf{ST_o}, \mathcal{ST}_m)$ be a timed state tree structure with $\mathbf{ST} = (X, x_o, \mathcal{M}, \mathcal{E})$. A *predicate* P defined on $\mathcal{B}(\mathbf{ST})$ is a function $P : \mathcal{B}(\mathbf{ST}) \to \{0, 1\}$. Also a predicate can be identified by a set of basic state trees, say B_P , such that $B_P := \{b \in \mathcal{B}(\mathbf{ST}) | P(b) = 1\}$.

Let $b \in \mathcal{B}(\mathbf{ST})$. We say predicate P *holds*, or is *satisfied* for b, i.e. $b \models P$ if and only if $b \in B_P$. Let $ST_1 \leq ST$. We say predicate P *holds*, or is *satisfied* for ST_1, i.e. $ST_1 \models P$ if and only if $\mathcal{B}(ST_1) \subseteq B_P$. Write $Pred(ST)$ as the set of all predicates defined on ST. Consistently, write $(\forall P \in Pred(ST))\emptyset \models P$, where $\emptyset$ is the empty state tree.

We define a partial order on $Pred(ST)$ by use of subset containment: $P_1 \preceq P_2$ iff $P_1 \wedge P_2 = P_1$. That is, P_1 precedes P_2, or is stronger than P_2, if and only if $(\forall b) b \models P_1 \Rightarrow b \models P_2$. Say P_1 is a *subpredicate* of P_2.

The following function θ encodes a sub state tree of $\mathbf{ST}$ to the predicate it satisfies. A *state variable* for a timed holon H(or an OR superstate x) is a variable whose range is the internal state set of H(or the set of all children of x).

Definition 8 (θ) Denote by v_x the state variable for the OR superstate x. Let $\mathbf{ST}_1 = (X_1, x_o, \mathcal{M}_1, \mathcal{E}_1)$ be a sub state tree of $\mathbf{ST}$. Define $\theta : \mathcal{ST}(\mathbf{ST}) \to Pred(\mathbf{ST})$ recursively by $\theta(\mathbf{ST}_1) :=$

$$\begin{cases} \bigwedge_{y \in \mathcal{E}_1(x_0)} \theta(\mathbf{ST}_1^y), & \text{if } \mathcal{M}(x_0) = and; \\ \bigvee_{y \in \mathcal{E}_1(x_0)} ((v_{x_0} = y) \wedge \theta(\mathbf{ST}_1^y)), & \text{if } \mathcal{M}(x_0) = or; \\ 1, & \text{if } \mathcal{M}(x_0) = simple. \end{cases}$$

where $\mathbf{ST}_1^y$ denotes the child state tree of $\mathbf{ST}_1$ that is rooted by y. Notice that if x_o is an OR superstate, we can exploit the tautology $(\bigvee_{y \in \mathcal{E}(x_0)} (v_{x_0} = y) \equiv 1)$ to simplify $\theta(\mathbf{ST_1})$. For any set B_P of basic state trees, its predicate representation is given by $P := \bigvee_{b \in B_P} \theta(b)$. If $B_P = \mathcal{B}(\mathbf{ST}_1)$, then $P := \theta(\mathbf{ST}_1)$.

In the TSTS $\mathbf{G}$, the *initial predicate* $P_o := \theta(\mathbf{ST}_o)$, and the *marker predicate* $P_m := \bigvee_{\forall \mathbf{ST}_i \in \mathcal{ST}_m} \theta(\mathbf{ST}_i)$. Now we can rewrite the plant TSTS by $\mathbf{G} = (\mathbf{ST}, \mathcal{H}, \Sigma, \Delta, P_o, P_m)$, where $\mathbf{ST}_o$ and $\mathbf{ST}_m$ are replaced by their predicate counterparts.

B. Encode Γ

Let x be an OR superstate. Call v_x the normal state variable of x. Also denote by v'_x the prime state variable of x. In a transition relation, v_x will be used to record target state information, while v'_x is for the source states.

Following the method of [12], we encode the entire set of transitions labelled by a given event σ as a triple $(N_\sigma, \mathbf{v}_{\sigma,S}, \mathbf{v}_{\sigma,t})$. $N_\sigma(\mathbf{v}'_{\sigma,S}, \mathbf{v})$ is the transition relation with $\mathbf{v}'_{\sigma,S}$ the set of prime variables for the source states where $\delta(., \sigma)!$ and $\mathbf{v}$ the set of all variables in $\mathbf{G}$. $\mathbf{v}_{\sigma,t}$ is the set of normal variables for those target states which $\delta(., \sigma)$ hits.

Definition 9 ($\hat{\Gamma}$) Let $\sigma \in \Sigma$ and $(N_\sigma, \mathbf{v}_{\sigma,S}, \mathbf{v}_{\sigma,t})$ represent the transitions labelled with σ. $\hat{\Gamma} : Pred(\mathbf{ST}) \times \Sigma \to Pred(\mathbf{ST})$ is defined by[1]

$$\hat{\Gamma}(P, \sigma) := (\exists \mathbf{v}_{\sigma,T}(P \wedge N_\sigma))[\mathbf{v}'_{\sigma,S} \to \mathbf{v}_{\sigma,S}]$$

Notice that $P[\mathbf{v}'_{\sigma,S} \to \mathbf{v}_{\sigma,S}]$ means replacing all prime variables in $\mathbf{v}'_{\sigma,S}$ by their respective normal variables in $\mathbf{v}_{\sigma,S}$.

IV. Controllability and State Feedback Control

Let $\mathbf{G} = (\mathbf{ST}, \mathcal{H}, \Sigma, \Delta, P_o, P_m)$ be a timed state tree structure. The *reachability* predicate $R(\mathbf{G}, P)$ is defined to designate all the basic state trees that can be reached from initial basic state trees via state trees satisfying P [12].

The *weakest liberal precondition* is the predicate transformer $M_\sigma : Pred(\mathbf{ST}) \to Pred(\mathbf{ST})$ defined for basic state trees as follows: $b \models M_\sigma(P)$ iff $\Delta(b, \sigma) \models P$. Notice that if $\Delta(b, \sigma) = \emptyset$, then for any P, $\Delta(b, \sigma) \models P$.

Before defining controllability, we need to define a new predicate called *forcing-free* predicate($\mathcal{F}$). We adopt the concept of *forcing* from [9] and adjoin it to the STS framework of [12].

Definition 10 ($\mathcal{F}$) *Forcing-free predicate*. For all $b \in \mathcal{B}(\mathbf{ST})$

$$\mathcal{F}(b) = \begin{cases} 1, & \text{if } (\nexists \sigma \in \Sigma_{for})\ \Delta(b, \sigma)!\ ; \\ 0, & \text{otherwise.} \end{cases}$$

Any state b is forcing-free if there is no forcible event defined there. We also let $f : \mathcal{B}(\mathbf{ST}) \to \Sigma$ denote the *state feedback control* (SFBC) for $\mathbf{G}$, where for $b \in \mathcal{B}(ST)$: $f(b) \supseteq \begin{cases} \Sigma_u, & \text{if } \mathcal{F}(b) = 0\ ; \\ \Sigma_u \cup \{tick\}, & \text{if } \mathcal{F}(b) = 1 \end{cases}$. The event σ is *enabled* at b if $\sigma \in f(b)$, and is *disabled* otherwise. For

[1]If v_x is a state variable appearing in predicate P and the range of v_x is $\mathcal{E}(x) = \{y_1, y_2, ..., y_n\}$, $\exists v_x P = \bigvee_{1 \leq i \leq n} P[y_i/v_x]$ where $P[y_i/v_x]$ is the resulting predicate after assigning y_i to v_x.

event σ introduce the predicate $f_\sigma : \mathcal{B}(ST) \to \{0,1\}$ defined by $f_\sigma(b) = 1$ iff $\sigma \in f(b)$. Thus the SFBC f can be implemented by the set of predicates $\{f_\sigma | \sigma \in \Sigma\}$. The closed-loop transition function induced by the SFBC f is given by $\Delta^f(b,\sigma) = \begin{cases} \Delta(b,\sigma), & \text{if } f_\sigma(b) = 1; \\ \emptyset, & \text{otherwise} \end{cases}$. Unlike Δ, Δ^f need be defined only on basic state trees. We write the controlled TSTS as $\mathbf{G}^f = (\mathbf{ST}, \mathcal{H}, \Sigma, \Delta^f, P_0^f, P_m)$ with $P_o^f \preceq P_o$ for the closed system supervised by the SFBC f. Notice that in general some initial basic state trees are excluded from G^f. To choose $P_o^f (\preceq P_o)$, the allowable initial basic state trees, is also the responsibility of the synthesizer.

Definition 11 A predicate $P \in Pred(\mathbf{ST})$ is *weakly controllable* wrt. $\mathbf{G}$ if (1)$(\forall \sigma \in \Sigma_u) P \preceq M_\sigma(P)$, and (2)$(P \wedge \mathcal{F}) \preceq M_{tick}(P)$.

Thus a predicate P is weakly controllable if at a basic state tree b that satisfies P an uncontrollable event occurs, that uncontrollable event changes the state of the system to another basic state tree that also satisfies P. Also the occurrence of *tick* if there is no forcible event defined at b should lead to a basic state tree that also satisfies P.

The reason we drop the reachability condition $P \preceq R(\mathbf{G}, P)$in [13] is that the computation of reachable predicate is expensive and, as will be seen later, unnecessary for the synthesis of SFBC f.

Proposition 2 Let P be a weakly controllable predicate of $\mathbf{G}$. Then $R(\mathbf{G}, P)$ is controllable. □

Given a predicate, it is easier to verify its weak controllability than to verify its controllability. What one needs is just some local information, as to obtain $M_\sigma(P)$ only the one step transition function Δ is called. By avoiding the expensive computation of the reachable predicate $R(\mathbf{G}, P)$, we gain computational efficiency. By Theorem 1 we will show that it is adequate just to get a weakly controllable predicate . This result guarantees that given a weakly controllable predicate P, there exists a SFBC f to implement its reachable subpredicate $R(\mathbf{G}, P)$. Therefore, this theorem underlies the synthesis procedure. A similar result was given in [13]. However, that result requires a controllable predicate P, which is computationally more expensive to verify.

Theorem 1 Let $P \in Pred(\mathbf{ST})$ and $P \wedge P_0 \neq false$. Then P is weakly controllable if and only if there exists a SFBC f for $\mathbf{G}$ such that $R(\mathbf{G}^f, true) = R(\mathbf{G}, P)$, where $\mathbf{G}^f = (ST, \mathcal{H}, \Sigma, \Delta^f, P_0^f, P_m)$ and $P_0^f = P \wedge P_0$. □

Note that even though P may not be reachable, the weak controllability of P guarantees that its reachable subpredicate $R(\mathbf{G}, P)$ can be implemented by a SFBC f. Thus the expensive computation of $R(\mathbf{G}, P)$ can be completely avoided. Now suppose P is not weakly controllable. Denote the family of weakly controllable subpredicates that are stronger than P by

$$\mathcal{CP}(P) = \{K \in Pred(ST) | K \preceq P \,\&\, K \text{weakly controllable}\}$$

Proposition 3 $\mathcal{CP}(P)$ is noneempty and is closed under arbitrary disjunctions. In particular $\mathcal{CP}(P)$ contains a (unique) supremal element $sup\mathcal{CP}(P)$. □

To compute $sup\mathcal{CP}(P)$, we define a predicate transformer [.] in $\mathbf{G}$ by $R \to [R]$, inductively as follows:
1. $b \models R \Rightarrow b \models [R]$
2. $b \models [R]$ & $b \neq \emptyset$ & $\sigma \in \Sigma_u$ &$\Delta(b', \sigma) = b \Rightarrow b' \models [R]$
3. $b \models [R]$ & $b \neq \emptyset$ & $b' \models \mathcal{F}$ & $\Delta(b', tick) = b \Rightarrow b' \models [R]$
4. No other basic state trees b satisfy $[R]$.

In other words, $[R]$ holds all the basic state trees that can reach R only by uncontrollable paths, i.e. uncontrollable event or *tick* when there is no forcible event.

Theorem 2 $sup\mathcal{CP}(P) = \neg[\neg P]$ □

The *coreachability* predicate is defined to hold for all basic state trees from which some $b_m \models P_m$ can be reached via trees satisfying P [12]. A predicate P is *nonblocking* for $\mathbf{G}$ if $R(\mathbf{G}, P) \preceq CR(\mathbf{G}, P)$, i.e., each basic state tree, reachable from an initial basic state tree by a path on which all basic state trees satisfy P, can reach a marker basic state tree by a path on which all trees satisfy P.

A predicate P is *coreachable* for $\mathbf{G}$ if $P \preceq CR(\mathbf{G}, P)$. Notice that the coreachability of P implies the nonblocking of P, as in general $R(\mathbf{G}, P) \preceq P$, but verifying the coreachability of P requires less computation. A SFBC f for $\mathbf{G}$ is *nonblocking* if $R(\mathbf{G}^f, true) \preceq CR(\mathbf{G}^f, true)$.

Theorem 3 Let $P \in Pred(\mathbf{ST})$ and $P \wedge P_0 \neq false$. Then there exists a nonblocking SFBC f for $\mathbf{G}$ such that $R(\mathbf{G}^f, true) = R(\mathbf{G}, P)$ if P is weakly controllable and coreachable. □

Again suppose P is not weakly controllable or not coreachable. Denote the family of weakly controllable and coreachable subpredicates that are stronger than P by $\mathcal{C}^2\mathcal{P}(P)$. To compute $sup\mathcal{C}^2\mathcal{P}(P)$, define a predicate transformer $\Omega_P : Pred(\mathbf{ST}) \to Pred(\mathbf{ST})$ in $\mathbf{G}$ by

$$\Omega_P(K) = P \wedge CR(\mathbf{G}, sup\mathcal{CP}(K)) = P \wedge CR(\mathbf{G}, \neg[\neg K])$$

Proposition 4 Let $K_0 = P$ and $K_{i+1} = \Omega_P(K_i)$. Then $K = \lim_{i \to \infty} K_i$ exists and $sup\mathcal{C}^2\mathcal{P}(P) = K$. □

Here the limit is achieved in a finite number of steps.

V. Symbolic Synthesis

Let $\mathbf{G} = (\mathbf{ST}, \mathcal{H}, \Sigma, \Delta, P_o, P_m)$ be a timed state tree structure. A specification $\mathcal{S} \subseteq \mathcal{ST}(\mathbf{ST})$ is given as a set of illegal sub state trees of $\mathbf{ST}$, which the supervisor must forbid the system from visiting.

Of course, $\mathcal{S}$ can be encoded by $S = \bigvee_{\forall \mathbf{T} \in \mathcal{S}} \theta(\mathbf{T})$. Then, our synthesis objective is to find the largest subpredicate of $\neg S$ that is both nonblocking and controllable.

Based on Proposition 4, we have a synthesis algorithm for computing $sup\mathcal{C}^2\mathcal{P}(P)$:

Algorithm 1
```
1) $K_0 = P$
2) $K_{i+1} = \Omega_P(K_i) = P \wedge CR(\mathbf{G}, \neg[\neg K_i])$
3) If $K_{i+1} = K_i$, then $sup\mathcal{C}^2\mathcal{P}(P) = K_i$. Otherwise, go back to step 2.
```

The algorithm terminates because $Pred(\mathbf{ST})$ is a finite set. In Algorithm 1, each computation of $\Omega_P(K_i)$ needs to call [.] and $CR(\mathbf{G}, .)$ once. So the faster these two functions execute, the better. The following algorithms are immediate from the definition of $[P]$ and $CR(\mathbf{G}, P)$.

Algorithm 2: $[P]$

1)$K_0 = P$

2)$K_{i+1} = K_i \vee (\bigvee_{\sigma_u \in \Sigma_u} \hat{\Gamma}(K_i, \sigma_u))$

3)$K_{i+1} = K_{i+1} \vee (\mathcal{F} \wedge \hat{\Gamma}(K_{i+1}, tick))$

4) If $K_{n+1} = K_n$, then $[P] = K_n$. Otherwise, go back to step 2.

In Algorithm 2, line 2 deals with uncontrollable events while line 3 considers the *tick* transitions whose source states are forcing-free.

Algorithm 3: $CR(\mathbf{G}, P)$

1) $K_0 = P \wedge P_0$

2) $K_{i+1} = K_i \vee (P \wedge \bigvee_{\sigma \in \Sigma} \hat{\Gamma}(K_i, \sigma))$

3)If $K_{n+1} = K_n$, then $CR(\mathbf{G}, P) = K_n$. Otherwise, go back to step 2.

By employing BDD as the representation of predicates, we can compute $sup\mathcal{C}^2\mathcal{P}(P)$ by combining Algorithms 1,2 and 3.

VI. Example

Consider the workcell shown in Fig. 6, consisting of six machines and a buffer of size n. A workpiece produced by M_1, M_2 or M_3 is placed in the buffer and is consequently available for further work by M_4, M_5 or M_6. The required specification for the buffer is not to overflow or underflow. We first model each machine as a timed holon with about

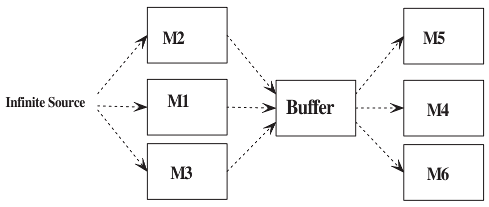

Fig. 6. 6-machine workcell

90 states. These timed holons are not the same because each machine has its own temporal behavior. The only shared event between these holons is *tick*. So we can model the whole system by TSTS. We modelled the cell as a TSTS of size 10^{12}.The state tree of this TSTS is shown in Fig 7. Each machine M_i and the buffer are OR superstates. Our BDD-based program computed the controller in about 5 minutes on a personal computer with centrino CPU and 1 GB of RAM while the TDES existing software TTCT (Chapter 9 of [1]) cannot possibly find the controller for a system of this size. Our program also provides the control function for each controllable event. This function can determine if that event is disabled or forced at each state of the system.

VII. Conclusions

In this paper, we introduced a timed version of STS [12], TSTS. The model includes a global clock and we proved that TSTS will not stop the clock. In order to perform control design efficiently,we employed a symbolic representation of TSTS and developed a recursive algorithm. The state

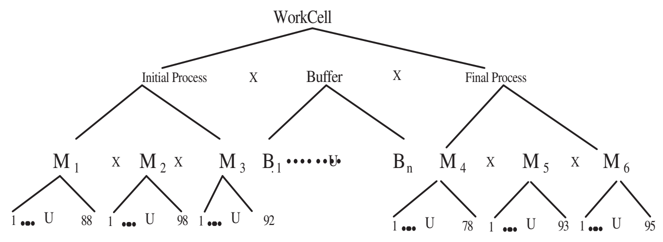

Fig. 7. State Tree of 6-machine workcell

space explosion problem is effectively managed. TSTS can model complex real time systems with both hierarchical and concurrent structures but without shared events among the modules except *tick*. It is hoped to relax this restriction in future work. Our symbolic synthesis can compute the optimal nonblocking supervisor for some large TDES, with reasonable usage of memory and time. The controller can be implemented in a straightforward way by control functions. The symbolic approach should clear the way for the industrial application of the TSTS framework.Work continues on new recursive algorithms to make the synthesis faster and less demanding of memory.

References

[1] W. M. Wonham, "*Supervisory Control of Discrete-Event Systems*," *Department of Electrical and Computer Engineering, University of Toronto, 2007*, available at http://www.control.utoronto.ca/DES.

[2] P. Gohari and W. M. Wonham, "*On the complexity of supervisory control design in the RW framework*," *IEEE Transactions on Systems, Man and Cybernetics, Special Issue on DES*, vol. 30, no. 5, pp. 643–652, 2000.

[3] R. J. Leduc, M. Lawford, and W. M. Wonham, "*Hierarchical Interface-based supervisory control: AIP example* ," *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing*, pp. 396–405, 2001.

[4] D. Harel, "*Statecharts: A visual formalism for complex systems* ," *Science of Computer Programming*, vol. 8, pp. 231–274, June 1987.

[5] B. Wang, "*Top-down design for RW supervisory control theory*," *MASc thesis, Department of Electrical and Computer Engineering, University of Toronto*, 1995.

[6] P. Gohari and W. M. Wonham, "*A linguistic framework for controlled hierarchical DES*," *in 4th International Workshop on Discrete Event Systems*, pp. 207–212, 1998.

[7] C. Ma and W. M. Wonham, "*Nonblocking supervisory control of state tree structures*," *IEEE Transactions on Automatic Control,*, vol. 51, pp. 782–793, May 2006.

[8] Y. Kesten and A. Pnueli, "*Timed and Hybrid Statecharts and their Textual Representation*," *In J. Vytopil, editor, Formal Techniques in Real-Time and Fault-Tolerant Systems, volume 571 of Lecture Notes in Computer Science. Springer Verlag*, 1992.

[9] B. Brandin and W. M. Wonham, "*Supervisory control of timed discrete event systems*," *IEEE Transactions on Automatic Control*, vol. 39, no. 2, pp. 329–342, 1994.

[10] J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang, "*Symbolic Model Checking:* 10^{20} *states and beyond*," *Information and Computation*, vol. 98, pp. 142–170, June 1992.

[11] R. E. Bryant, "*Graph-based algorithms for boolean function manipulation*," *IEEE Transactions on Computers*, vol. 12, no. 8, pp. 677–691, August 1986.

[12] C. Ma and W. M. Wonham, "*Nonblocking Supervisory Control of State Tree Structures*," *Berlin, Germany: Springer-Verlag*, vol. 317, 2005, LNCIS.

[13] A. Saadatpoor, "*State-Based Control of Timed Discrete Event Systems using Binary Decision Diagrams*," *MASc thesis, Department of Electrical and Computer Engineering, University of Toronto*, June 2004.