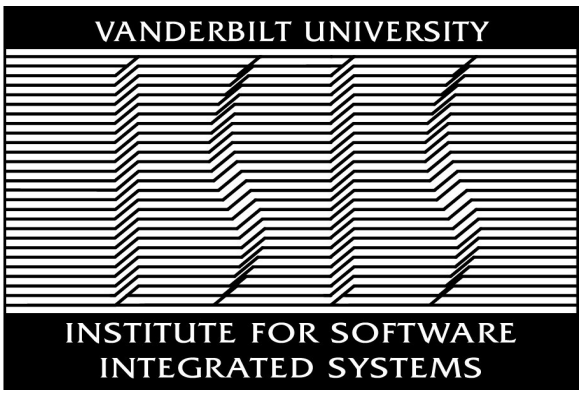

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, 37203

A Discussion on Supervisory Control Theory in Real-Time Discrete Event Systems

Abhishek Dubey

TECHNICAL REPORT

ISIS-09-112

November, 2009

A Discussion on Supervisory Control Theory in Real-Time Discrete Event Systems

Abhishek Dubey
Institute for Software Integrated Systems
Department of Electrical Engineering and Computer Science
Vanderbilt University, Nashville, TN, USA

***Abstract*— The seminal work on theory of Supervisory control in Discrete Event Systems was pioneered by Ramadge and Wonham. Taking inspiration from this work, extensive research has been done to extend the theory to real-time. The main problem in the research related to real-time supervisory control is that theories of RW framework are not directly applicable because of infiniteness of state space. This paper gives a survey of the methods of real-time supervisory control with a focus on their methods to confront the problem of infinite state space.**

I. Introduction

A Discrete event system (DES) is a discrete state, event-driven system, in which, the state evolution depends entirely on occurrence of discrete events over time [9], [27]. The event driven property implies that the state of the system changes at discrete points in time which correspond to *instantaneous* occurrence of the events. Unlike, a continuous-variable dynamic system, the state of a DES, if tracked vs. time on a graph will be a piece wise continuous function. Many systems are in fact discrete event systems, or can be modeled as discrete event systems, if only the order among the events is of consequence.

In supervisory control of DES, one seeks to restrict the behavior of a plant in such a way that the supervised system (closed loop of the supervisor and plant) meets the required specifications. These might include "safety" specifications (e.g. Prohibit the behavior that can lead to a catastrophe) and "liveness" specifications (e.g. Guarantee the eventuality of a specified goal). The seminal work on supervisory control in DES was pioneered by Ramadge and Wonham [27], [28]. In their framework[1], both plant and specification are modeled as finite state automaton. The set of events are divided into controllable, similar to user controlled input in the traditional control theory, and uncontrollable, similar to environmental disturbances. Depending upon history of executed events, the control is achieved by disabling the events that leads to undesirable behavior.

However, in some systems the elapse of time between consequent events can affect the system behavior in a significant manner. For example, in the Fig. 1 if delay of event e_2 by some t seconds causes the state to change from S_2 to S_4 and skip S_3 then we can conclude that the DES model is unsuitable for this system. Therefore, one can no longer ignore the timing of events. The set of possible time values within any given bound has values that are in real-number space. That is between any two instants of time value, a third value can be always found [13]. This shows that the set of time, even if bounded and compact i.e. closed interval, is uncountable, dense and hence has infinite state space. When introduced in the DES model the total state space becomes "infinite". This violates the basic assumption of RW framework that plant and specification are modeled as "finite" state automaton.

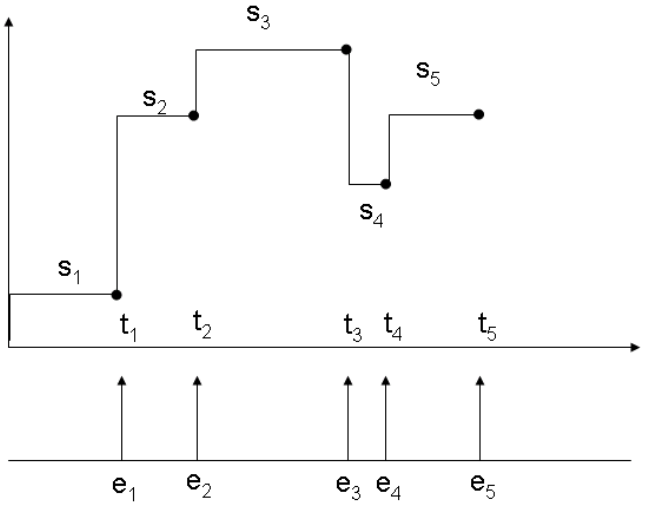

Fig. 1. A sample execution trace in a discrete event system. In this example, the states are discrete and the transition between them happens when an event occurs.

The significance of time is specifically apparent in a class of systems called real-time discrete event systems. *Real-Time* implies that the correctness of system is contingent to the correctness of logical result, and also on the time at which results are produced [8]. For example, if a missile has been targeted at some installation, anti-missile batteries only have a fixed time window in which they can shoot down the rogue missile. If this window is missed, catastrophic consequences ensue. In general, these systems share the following characteristics:

1) Events occur at discrete times. These events may imply start/end of a job.
2) Events can be enabled or disabled due to physical state of the process.
3) There are timing constraints governing the ordering between events.
4) Processes can be non-deterministic in nature.

[1]We will refer to it as RW framework in the rest of the paper.

5) Interaction between sub-processes can be synchronous or asynchronous.
6) "Hard" time deadlines, if associated with a job, must be met.

In order to extend the supervisory control to this class of systems, first one has to model the DES behavior with timing information, then the infinite state space of the model has to be translated to a finite state space by using abstraction. Only then can one apply the extensions of RW theory to design supervisor for timed discrete event systems.

In the research community, two different approaches have been used. The first approach, uses a discrete time model to specify the behavior of plant and specification. An example of such a model is Timed Transition Model [25], [16]. In the discrete time model, the set of integers are used to build a partition on the set of reals [7], [6], [25], [16]. This is done by a priori fixing the smallest measurable time unit. This can be understood as the sampling of the continuous time at the rate fixed by this smallest unit. All the continuous values that occur after the current sample and before the next sample are approximated by the next sample. Due to the use of discrete time the state space of the model remains finite. This approach has been used to formulate the real-time supervisory control theory by Brandin and Wonham [6], Daren Cofer and Vijay Garg [11]. In [15], Peyman Gohari and Wonham have extended the work of [6].

The other approach is to use a dense real-time model such as timed automaton [1]. This way of modeling time is most natural, as it imposes minimal restriction on the modeling semantics. There is no minimum measurable time unit. However, the cost of expressiveness has to be paid with computational difficulty in analysis. This is due to the infinite domain of time. However, with certain restrictions in place, it is possible to construct a bisimilar finite transition system from the infinite space of a timed automaton. This transition system is called a region automaton. RW theory can then be applied to develop supervisor for the finite region automaton. This approach was first proposed by Wong-Toi and Hoffman in [31]. In [19], Ahmed Khoumsi and Mustapha Nourelfath proposed a different method for transforming a timed automaton into a minimal and equivalent finite state automaton using two special types of events, Set and Exp. The transformed model, having a small state size compared to corresponding timed automaton, was used to design supervisor using the usual RW framework. In [22], Ratnesh Kumar et. al. have used process algebra to explain the interaction between the supervisor and plant under prioritized synchronization.

One approach that consistently differs than the ones using the RW theory is the work of Oded Maler et.al[24], [26]. They translate the control problem into a game between a supervisor (with desired specification) and a plant with the goal of driving the plant into a set of safe state that will make it impervious to the environmental disturbances that can lead to illegitimate behavior.

What follows is a review of both discrete and dense model of time. With the differences of these two models in mind, we will present a comparative study of different supervisory theories related to these two models of time. We will also present arguments that the region automaton constructed in the dense time approach can be considered similar to the timed transition model. The choice in using the either one is dictated by a trade off between the ease of specification of plant behavior in the concerned model of choice and the related computational complexities.

A. Preliminaries

Let Σ be a set of finite alphabets[2]. Then Σ^* denotes the set of all possible finite sequences, including the empty sequence denoted by $\emptyset$, that can be formed using alphabets of Σ. Let $len(s)$ denote the length of the finite sequence s. Concatenation of two sequences, $s, t \in \Sigma^*$, is defined to be a new finite sequence $s \circ t \in \Sigma^*$. In order to reduce the notations we will write the concatenated string simply as st. It should be noted that $len(st) = len(s) + len(t)$. Also, $st \leq s$ i.e. s can be extended by the alphabets from some other sequence (in this case t) to form the sequence st. A prefix of any finite sequence $s \in \Sigma^*$ is a set $\overline{s} = \{u \in \Sigma^* | (\exists v \in \Sigma^*)(uv = s)\}$. Prefix of any language, $L \subseteq \Sigma^*$, written as $\overline{L}$, is the union of prefixes of all the sequences that are a member of L. L is said to be complete and closed if $L = \overline{L}$.

Given a DES, $G = (Q, q_0, \Sigma, \delta, Q_m)$, where Q is a finite set containing all the states of DES, $Q_m \subseteq Q$ is the set of marked states, $q_0 \in Q$ is the initial state, δ is the transition map. $\delta(q_i, \sigma)!$ denotes that DES can accept the alphabet $\sigma \in \Sigma$ if it is in the state q_i. Language $L(G)$ denotes the finite sequence of strings in Σ^* that can be executed by following the transition map. Inductively, the execution due to a string $s\sigma$, where $s \in \Sigma^*$, and $\sigma \in \Sigma$ is the current alphabet, is written as $\delta(\delta(q_0, s), \sigma)$. Therefore, $L(G) = \{s \in \Sigma^* | (\delta(q_0, s)!\}$. These *languages* also signify the set of *behaviors* of a DES. Language $L_m(G)$, also known as marked language, is the set of finite sequences that if executed by the DES leads them to a marked state i.e. $L_m(G) = \{s \in \Sigma^* | (\delta(q_0, s)! \wedge \delta(q_0, s) = q' \implies q' \in Q_m\}$. G is said to be non-blocking if $\overline{L}_m(G) = L(G)$ i.e. any finite string that can be accepted by the DES can in future be completed to form a string in the marked language.

B. A Quick Review: Supervisory Control Theory in DES

In Ramadge and Wonham's theory, a supervisor controls a plant by observing its current execution string and then disabling certain possible events (one call also call them alphabets) that might lead to undesirable behavior.

Consider an unsupervised DES $G = (Q, q_0, \Sigma, \delta, Q_m)$ with corresponding pair of languages= $(L(G), L_m(G))$. The set of events (Σ) is partitioned into two set, controllable (Σ_c) and uncontrollable events Σ_u). The supervisor is a map to a control pattern, $V : \overline{L} \to \Gamma$, such that any set $\gamma \in \Gamma$, where $(\gamma = V(s))(s \in L(G))$ represents the events that should remain enabled in G, after it has already executed the events in s. It is assumed that the set of uncontrollable events are due to environmental disturbances and can never be disabled i.e. $(\forall \gamma \in \Gamma)(\Sigma_u \subseteq \gamma)$.

[2]These alphabets are used to denote possible events in a DES

Problem 1: (Supervisory Control Problem in DES) Given a specification language $K \subseteq L_m(G)$, design a supervisor DES V such that the supervised system, denoted (G/V), satisfies the following (a) $L_m(G/V) \subseteq K$ (b) $\overline{L}_m(G/V) = L(G/V)$ i.e. the supervised DES is also non-blocking.

In RW framework the solution to the above problem hinges upon the notion of controllability of the specification with respect to the unsupervised DES. This can be summarized in the following theorem:

Theorem 1 (Notion of Controllability): Given a DES G, and a specification $K \subseteq L_m(G)$. A maximally permissive supervisor V exists iff

1) $\overline{K}\Sigma_u \cap \overline{L} \subseteq \overline{K}$ i.e. K is controllable.
2) $\overline{K} \cap L_m(G) = \overline{K}$ i.e. K is $L_m(G)$ closed.

Furthermore, even if the specification K is uncontrollable, a class of languages that are subset of specification K exists such that they are controllable and are $L_m(G)$-closed. Ramadge and Wonham [27], [28] have shown that this class of controllable sub-languages forms a lattice under subset inclusion and is closed under union. This can be expressed as the following theorem:

Theorem 2 (Set of Supremal Controllable Sublanguages): Let $C = \{E \subseteq K | E$ is controllable wrt L$\}$ be the set of controllable sublanguages of specification K. Following two properties are satisfied by this set:

1) $(\forall c_1, c_2 \in C)(\exists c_3 \in C)(c_3 = c_1 \cup c_2)$.
2) A supremum element of the set C, $sup(C)$ always exists and $sup(C) \in C$.

This implies that by iteration one can always find a fixed point solution $supC(K) \subseteq K$ that is the supremal controllable sublanguage of K. Thus the supremal controllable supervisor, $V = supC(G, K, \Sigma_u)$, will be maximally permissive while still implementing a subset of the specification. They also showed that the supervisory control problem has the complexity $O(|G|^2|K|^2)$.

II. Models of Time

A. Discrete Model: Timed Transition Model

Timed transition model, introduced by Ostroff [25], is based on the discrete approximation of time. The typical DES model $G = (Q, q_0, \Sigma, \delta, Q_m)$, is augmented with time bounds for each event. The time bounds are specified as triples $(\sigma, l_\sigma, u_\sigma)$ where, $\sigma \in \Sigma$, $l_\sigma, u_\sigma \in \mathbb{Z}^+$ s.t. $l_\sigma \leq u_\sigma$. The modeling function of time bounds is straightforward: l_σ typically represents a delay in firing of the event, and u_σ is a hard deadline, imposed by physical constraints on the system. Such an augmented DES is sometimes also referred to as *Timed Discrete Event System* (TDES).

In [6], differentiation is made between events that can be delayed forever and those that cannot be delayed forever. This is because for some events there might be no hard deadline i.e. $u_\sigma = \infty$ while others have a finite upperbound. The former are called remote events (denoted by the set Σ_{rem}) since they can be delayed forever, while the latter are termed prospective events (denoted by the set Σ_{spe}).

The semantics of execution of a TDES is given as a Timed Transition Graph(TTG). Corresponding to a given TDES $G = (Q, q_0, \Sigma, \delta, Q_m)$, a TTG is written as $G' = (Q', q'_0, \Sigma', \delta', Q'_m)$ where,

- $\Sigma' = \Sigma \cup \{tick\}$. $tick$ is an event that is generated by a global clock at a fixed sampling quantum. This sampling quantum decides the resolution of time measurement in a TDES and thus maps an infinite domain of real-time into a countable sequence $< 0, tick, 2tick, 3tick, \cdots >$. Due to sampling, a compact interval of dense time will map to a finite number of terms in the tick sequence.
- Define for each σ, T_σ, the timer interval,

$$T_\sigma = \begin{cases} [0, \mathrm{u}_\sigma] & \text{if } \sigma \in \Sigma_{spe} \\ [0, \mathrm{l}_\sigma] & \text{if } \sigma \in \Sigma_{rem} \end{cases}$$

  Then the state space of TTG, $Q' = Q \times \prod\{T_\sigma | \sigma \in \Sigma\}$.
- Initial State, $q'_0 = (q_0, \prod\{t_{\sigma 0} | \sigma \in \Sigma\})$, where,

$$t_{\sigma 0} = \begin{cases} \mathrm{u}_\sigma & \text{if } \sigma \in \Sigma_{spe} \\ \mathrm{l}_\sigma & \text{if } \sigma \in \Sigma_{rem} \end{cases}$$

- Marked states, $Q'_m \subseteq \times \prod\{T_\sigma | \sigma \in \Sigma\}$

Thus a state of a TTG is of the form $x = \{q \in Q, \{t_\sigma | \sigma \in \Sigma\}\}$. The second component of the state is called the timer for a given event in state x. The timer values are changed due to transitions representing passage of times. These are called $tick$ transition. These transitions can be explained as follows: Let $(a_1, a_2 \in Q')$, where $a_1 = \{q_1 \in Q, \{t_\sigma | \sigma \in \Sigma\}\}$ and $a_2 = \{q_2 \in Q, \{t'_\sigma | \sigma \in \Sigma\}\}$. Then $\delta'(tick, a_1) = a_2 \implies q_1 = q_2 \wedge t'_\sigma = max(t_\sigma - 1, 0)$. Thus, after every $tick$ the timer value is decremented by 1.

A timer associated with a prospective event represents its current deadline. If enabled in the current state a prospective event can fire if its current timer value satisfies the following inequality: $0 \leq t_\sigma \leq u_\sigma - l_\sigma$. In order to understand this, recall that the timers for prospective events are initialized from upper bound.

Example 1: Now, consider an example of a prospective timed event, $(\sigma, 2, 5)$. This implies that sigma can fire only after waiting for at least 2 time units and at most 5 time units. The initial value of timer, t_σ is 5. After the first $tick$, $t_\sigma = 4$. After second $tick$, $t_\sigma = 3$. Note, that at n^{th} $tick$ after initialization $n = u_\sigma - t_\sigma$. When $n \geq l_\sigma$, or $t_\sigma = u_\sigma - l_\sigma$, σ can fire. Now, if at any point $t_\sigma = 0$ i.e. $n = u_\sigma$, the hard deadline has arrived and the event should be either fired or a state transition must take place to a state in which σ cannot be accepted.

A similar explanation can be given for remote events. For them, the timer value at any instant denotes the current delay (initialized to l_σ). The event can fire only when the value of t_sigma is zero.

Further point to note is that not all events are enabled in all states. Suppose, a transition is made from state $q \in Q$ to $q' \in Q$. Let $\Sigma_1 = \{\sigma \in \Sigma | \delta(\sigma, q)!\}$ and $\Sigma_2 = \{\sigma \in \Sigma | \delta(\sigma, q')!\}$. Now, timer values are inherited after a transition for the events which are in the set $\Sigma_1 \cap \Sigma_2$, except the event due to which transition took place. For rest of the events in Σ_2/Σ_1, the timers are reinitialized to either their upper bound or lower bound depending upon if they are in remote or prospective. For all the events that are disabled in q' the timer is set to 0.

Using the just mentioned transition rules one can compute the TTG of a TDES by generating the reachability graph exploring all the possible transition due to $tick$ and eligible remote and prospective events. Consider an example:

Example 2: Consider the example of an endangered pedestrian described in [6]. This example consists of two timed discrete events systems, a bus and a pedestrian. The Bus== $(\{a, g\}, \{a\}, \{pass\}, \{[a, pass, g]\}, \{g\})$ can consume an event $pass$. Its two states are a, g, where $a = approaching$ and $g = gone\ by$. The pedestrian, $Ped = (\{r, c\}, \{r\}, \{jump\}, \{[r, jump, c]\}, \{c\})$ can consume the $jump$ event. Its two states are $r = on\ roadside$ and $c = on\ curb$. Consider two timing constraints: $(pass, 2, 2)$ and $(jump, 1, \infty)$. Figure 3 shows the composed TDES. The reachability graph or the TTG for this example is shown in Fig. 2. Given the small state space, it is easy to check that this timed transition graph is correct.

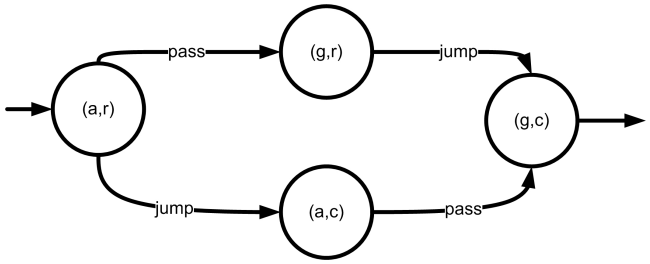

Fig. 3. The composed TDES for the bus pedestrian example. The timing constraints are $(pass, 2, 2)$ and $(jump, 1, \infty)$

The language generated by the TTG is a subset of $\{\Sigma \cup \{tick\}\}*$. The concepts for marked language, non-blocking follows straight from the DES model. The only additional constraint will be to enforce the condition that in all traces the $tick$ transition will occur infinitely often. This translates to the fact that time shall always diverge. It can be deduced that the traces of a TTG are interleaving of a number of tick transitions into the traces accepted by TDES model. Thus if the trace admitted by TDES is finite, the interleaved trace with ticks will also be finite.

Composition of two TDES, G_1, G_2 can be achieved in the same fashion as the composition of a DES and the reevaluation bounds for each event common to both TDES as, $(l_\sigma, u_\sigma) = (max(l_{1\sigma}, l_{2\sigma}), min(u_{1\sigma}, u_{2\sigma}))$

B. Dense Model: Timed Automaton

Classically, for systems with continuous timed variables, the timed automaton (TA) model [1], [18] is used for proving the correctness of system designs. This approach has also been applied to solve scheduling problems by modeling real-time tasks and scheduling algorithms as variants of timed automaton and performing reachability analysis on the model [21], [23]. The advantage of using timed automaton model is the greater expressiveness and ease of composition of subsystems.

A timed automaton consists of a finite set of states called *locations* and a finite set of real-valued clocks. It is assumed that time passes at a uniform rate for each clock in the automaton. Transitions between locations are triggered by the satisfaction of associated clock constraints known as *guards*. During a transition, a clock is allowed to be reset to zero value. These transitions are assumed to be instantaneous. At any time the value of each clock is equal to the time passed since the last reset of that clock. In order to make the timed automaton urgent, locations are also associated with clock constraints called *invariants* which must be satisfied for a timed automaton to remain inside a location. If there is no enabled transition out of a location whose invariant has been violated, the timed automaton is said to be *blocked*. Formally, a timed automaton can be defined as follows:

Definition 1 (Timed Automaton): A timed automaton is a 6-tuple TA=$< \Sigma, S, S_0, X, Inv, T >$ such that

- Σ is a finite set of alphabets which TA can accept.
- S is a finite set of locations.
- $S_0 \subseteq S$ is a set of initial locations.
- X is a finite set of clocks.
- $Inv : S \rightarrow \mathscr{C}(X)$ is a mapping called location invariant. $\mathscr{C}(X)$ is the set of clock constraints over X defined in BNF grammar by $\alpha ::= x \prec c | \neg\alpha | \alpha \wedge \alpha$, where $x \in X$ is a clock, $\alpha \in \mathscr{C}(X)$, $\prec \in \{<, \leq\}$, and c is a rational number.
- $T \subseteq S \times \Sigma \times \mathscr{C}(X) \times 2^X \times S$ is a set of transitions. The 5-tuple $< s, \sigma, \psi, \lambda, s' >$ corresponds to a transition from location s to s' via an alphabet σ, a clock constraint ψ specifies when the transition will be enabled and $\lambda \subseteq X$ is the set of clocks whose value will be reset to 0.

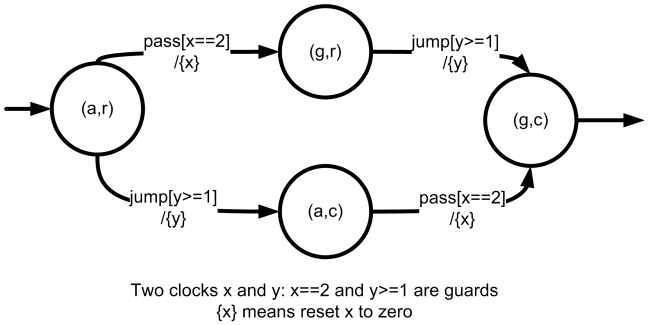

Fig. 4. The timed automaton model for Bus Pedestrian System (example 2)

The semantics of timed automaton models are described as an infinite state transition graph $A =< Q, \Sigma \cup \{\epsilon\}, Q_0, R >$. Each state in Q is a pair (s, v), where $s \in S$ and $v : X \rightarrow \mathbb{R}^+$ is clock value map, assigning each clock a positive real value. It is assumed that at any time all clocks increase with a uniform unit rate i.e. $\forall x \in X(\dot{x} = 1)$ is true. The initial state of A, Q_0 is given by $\{(q, v) | q \in S_0 \wedge \forall x \in X(v(x) = 0)\}$. Before defining the transition relations we must give some notations. For any $d \in \mathbb{R}^+$, let us define $v + d$ a clock assignment map which increases the value of each clock $x \in X$ to $v(x) + d$. For $\lambda \subseteq X$ introduce $v[\lambda := 0]$ to be the clock assignment that maps each clock $y \in \lambda$ to 0, but keep the value of all clocks $x \in X - \lambda$ same.

The transition relation R is composed of two types of transitions:

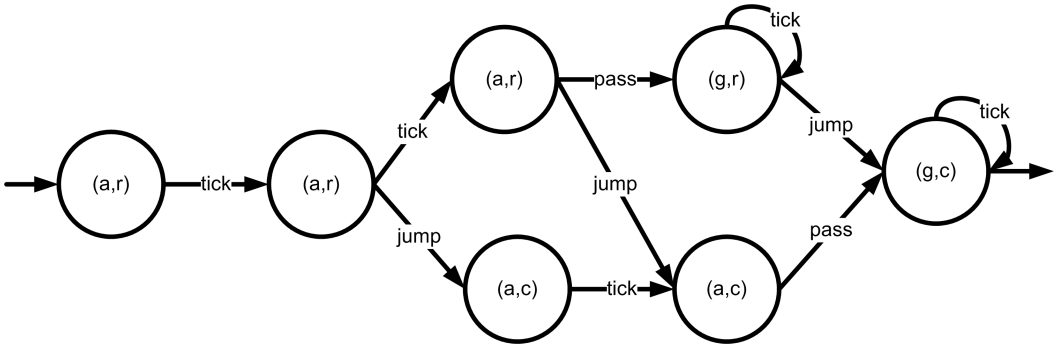

Fig. 2. The timed transition graph for the bus pedestrian example. Notice that the complete reachability graph has finite number of states.

- *Delay Transitions* refer to passage of time while staying in the same location. They are written as $(s, v) \xrightarrow{t} (s, v+t)$. The necessary condition is $v \in I(s)$ and $v + t \in Inv(s)$. This transition is represented as the occurrence of a stutter event ϵ.
- *Action Transitions* refer to occurrence of a transition from the set T. Therefore for any transition $< s, \sigma, \psi, \lambda, s' >$, we can write $(s, v) \xrightarrow{\sigma} (s', v[\lambda := 0])$, given that $v[\lambda := 0] \in I(s')$ and $v \in \psi$.

The trace of a timed automaton over $\Sigma \cup \{\epsilon\}$ is a function $\nu : I_\nu \to \Sigma \cup \{\epsilon\}$, where I_ν is an open (closed) time interval. To prevent zeno behavior, $\{t \in I_\nu | \nu(t) \in \Sigma/\epsilon\}$ i.e. the number of possible events in a time bound should be finite. An example trace for the timed automaton of bus pedestrian example (see Fig. 4)is $(\epsilon, 0)(pass, 2)(jump, 2.5)$.

Given a timed trace, an execution of the plant (timed automaton), is called a run. Thus a trace ν can be represented by a run that is a finite sequence in $(\Sigma \times \mathbb{R}^+)^*$. However, as one can envisage the state space associated with a timed automaton is infinite which makes the task of analysis including construction of supervisory controller difficult.

C. Dense Model: Timed Automaton: Region Automaton

In [1], a construction called region graph automaton is given to divide the infinite state space of clocks of timed automaton into finite number of equivalent classes. In the same paper it has been shown that this construction is bisimular to the timed automaton. Bisimular implies that for every trace of timed automaton there is a trace of region automaton and vice versa.

The main premise of constructing the equivalent classes of region automaton is the fact that the guard values in clock constraints are integers. The transition between two regions is represented by an event τ. For example, Fig. 5 shows various clock equivalence regions in the region graph associated by the timed automaton of bus and pedestrian example. One might notice that the clock regions in Fig. 5 follow a pattern. This pattern is visible if all the clock regions is divided into two sets, boundary conditions, e.g. $(x = 0, y > 1)$ and continuous region such as $(x \in (0, 1), y > 1)$. A region is called boundary region if for any clock it has a direct integral assignment. All other regions are continuous region. Now, from the figure one can notice that after a τ event a continuous region always moves to a boundary region and vice-versa. If we make one of these τ, one from continuous region to boundary region observable and make the other one unobservable, we will transform the region graph into a timed transition graph. So if events are allowed to happen only at boundary regions then we can always directly reduce the timed automaton to a TDES and construct the timed transition graph.

Using this clock equivalence region one can generate a finite transition system that reflects the behavior of timed automaton. This is called the region graph. One can say that this region graph is a DES with a language over $\{\Sigma \cup \tau\}^*$. Thus one can inherit the concept of marked language, nonblocking behavior and similar constructs from the DES theory.

III. Supervisory Control Using Discrete Time Model

The preliminary work in supervisory control using the discrete time model was done by Brandin and Wonham [6]. Their work has been implemented in a tool called TTCT [12]. The plant and specification were both specified as TDES. As noted in the earlier section the model of TDES partitions the set of events into remote and prospective events. The concept of controllable and uncontrollable events was borrowed from the RW framework. It was also noted that, Prospective events are a subset of uncontrollable events i.e. $\Sigma_{spe} \subseteq \Sigma_u$, and the remote events are a superset of controllable events i.e. $\Sigma_c \subseteq \Sigma_{remote}$. They also introduced the notion of forcible events, Σ_{for} that can if enabled can be fired, preempting the $tick$ transition.

In this framework, given any specification and a plant TDES, the first task is to compute the timed transition graphs, K and G such that $K \subseteq L_m(G)$. The supervisor is still expressed as a control pattern, $V : \overline{L} \to \Gamma$, where $\Gamma \subseteq \Sigma$ is the set of events to be is however, given any $s \in L(G)$, $\gamma = V(s)$ has to satisfy following rules: Let $Elig_G(s) = \{\sigma \in Sigma | \delta(q_0, s)!\}$. Then

1) $V(s) \cap Elig_G(s) \neq \emptyset$
2) $V(s) \supseteq \Sigma_u \cup \{tick\}$ if $V(s) \cap \Sigma_{for} = \emptyset$
3) $V(s) \supseteq \Sigma_u$ if $V(s) \cap \Sigma_{for} \neq \emptyset$

Generalizing this one can write the controllability theorem as:

Theorem 3 (Controllability in TTG): Given a timed transition graph, G, and the timed transition graph K of a

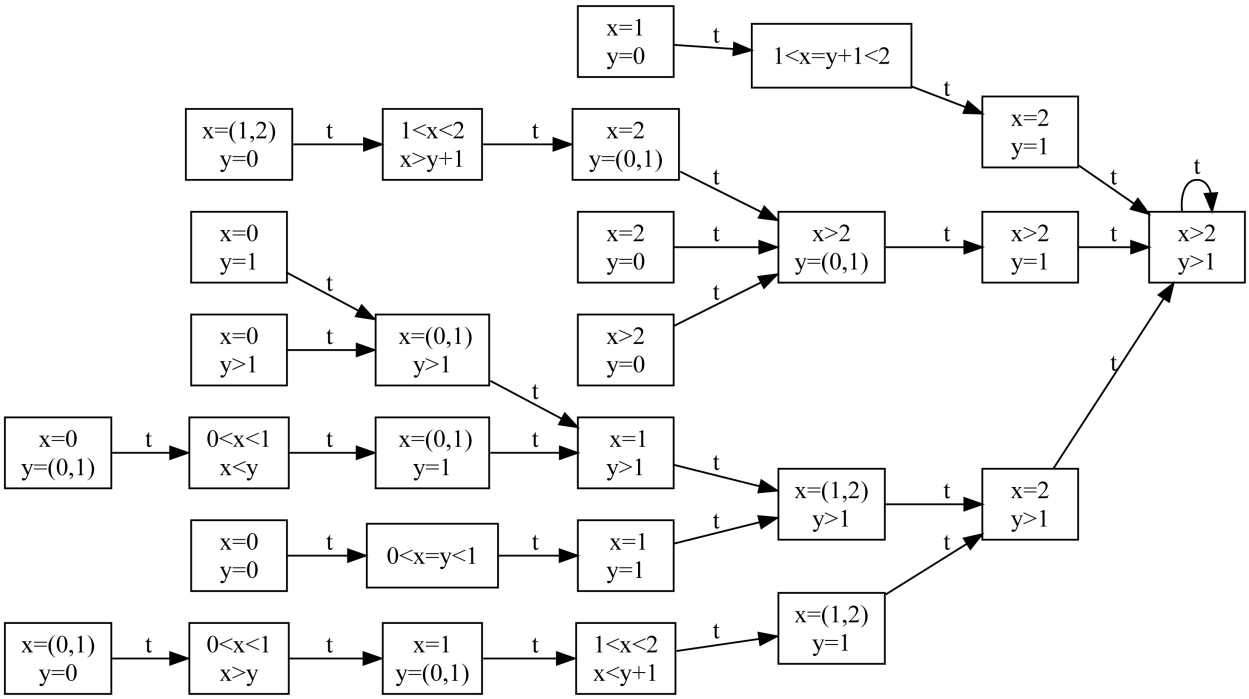

Fig. 5. Clock equivalence classes for the region graph of the timed automaton model for Bus Pedestrian System (example 2)

specification such that $K \subseteq L_m(G)$. Then, a maximally permissive supervisor V exists if and only if

1) $K = \overline{K} \cap L_m(G)$
2) $(\forall s \in \overline{K})\ (Elig_k(s) \cap \Sigma_{for} = \emptyset \implies Elig_k(s) \supseteq Elig_G(S) \cap (\Sigma_u \cup \{tick\}))$
3) $(\forall s \in \overline{K})\ (Elig_k(s) \cap \Sigma_{for} \neq \emptyset \implies Elig_k(s) \supseteq Elig_G(S) \cap \Sigma_u)$

The theorem 2 still stands true for the timed transition model. Therefore, after the construction of TTG the supremal controller can be found by using an algorithm of complexity same as that of untimed DES. However, the main problem is that the size of state space associated with any TTG can become very large soon.

Example 3: For the TDES model of example 2 a supervisor has to be designed such that the pedestrian always jumps before the bus passes. It is known that the $pass$ event is uncontrollable and $jump$ event is forcible. Fig. 6 shows the maximally permissive supervisor for this problem. One can see that the trace $tick\ tick$ is preempted by using the forcible event $jump$. This is done because the trace $tick\ tick$ would have enabled $pass$ which is uncontrollable and cannot be disabled.

In order to deal with the problem of large state space Gohari and Wonham suggested the use of abstraction in [15]. They use the idea of a reduced model, which even though has lesser states than the plant still encompasses all its behaviors. Any abstraction transformation that reduces the plant to this reduced model must have a corresponding inverse transformation, which when applied to a controller designed for reduced model gives a design for the original plant. Even though this design might not be optimal it is a trade-off against the computational complexity.

Gohari and Wonham used a transformation that scaled down the time bounds of each timed event. If the resulting bounds are not integers, they are rounded down (up) to nearest integer for lower (upper) bound. This transformation is a map that slows down the rate of ticks be aggregating a fixed number of ticks together. As the behavior of timed transition graph of reduced model is larger than the plant, any supervisor that complies with a time independent specification for reduced model will upon subjection to inverse transformation should still be a supervisor for original plant. They have successfully argued that this transformation does not increase the complexity of supervisor synthesis compared to [6]. However, since the supervisor is not optimal cases may arise when after the transformation the behavior of reduced model will be large enough so that no controllable sublanguage will exist.

The Brandin and Wonham framework was applied for real-time supervisory control of a processor for non-preemptive execution of periodic tasks by modeling the tasks as TDES in [10]. In this paper the authors showed that if a scheduler is computed as a maximally permissive supervisor then the problem of finding a schedule and determining schedulability are duals of each other.

In [11], a model that is a specialization of TDES is chosen. It is assumed that all events are remote events i.e. they can be delayed forever. In order to model these plants they used petri nets. Then a ($(max, +)$ algebra) [9] was used to define the controllable behavior as an invariance condition quantified by a lattice inequation, $x = A \otimes x \oplus v$, where x is the sequence of firing time vectors, v is a sequence of earliest allowable firing time vectors, and A is a matrix of delay functions. A set of desired behaviors or event schedules must

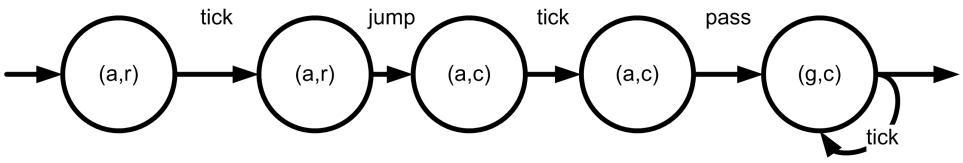

Fig. 6. Maximally permissive supervisor that makes pedestrian jump before the bus passes

satisfy this inequation for it to be realizable by only delaying controllable events. In this framework, optimal supervisors are found by computing the extremal solutions to the lattice inequation. Even though this framework does not give any computational advantages it provides an alternative formalism for the supervisory control theory in discrete timed systems.

IV. Supervisory Control Using Dense Time Model

The other approach is to use a dense real-time model such as timed automaton [1]. This way of modeling time is most natural, as it imposes minimal restriction on the modeling semantics. There is no minimum measurable time unit. However, the cost of expressiveness has to be paid with computational difficulty in analysis.

The first work in this area was described in [31] by Wong-Toi and Hoffman. They formulated the notion of controllability in the dense time domain by restricting the number of event based transitions in any bounded interval of time finite. Thus effectively their theorems are deductions of theorems presented in the Ramadge and Wonham framework. The main controllability theorem of Wong-Toi is reproduced here:

Theorem 4: Controllability Let G be a timed automaton. Let the timed traces of G form the language pair,$(L(G)^t, L(G)^t_M)$. The event set is partitioned into uncontrollable and controllable events, $\Sigma = \Sigma_c \cup \Sigma_u$. Let K^t be a timed specification s.t. $K^t \subseteq L(G)^t_M$. Then there exists a supervisor such that the supervised timed automaton, G/V, generates the language pair, $(L(G/V)^t, L(G/V)^t_M)$ such that it is non-blocking and $L(G/V)^t_M = K^t$ if and only if

1) $\overline{K^t}.(\Sigma_u \times \mathbb{R}^+) \cap \overline{L^t} \subseteq \overline{K^t}$ (Notice the similarity with untimed version)
2) $\overline{K^t} \cap L(G)^t_M = K^t$ (Ensures non-blocking)

The main algorithm of supervisor synthesis involves two operation $untime$ and $time$. The former converts a trace of timed automaton to a trace in region graph, while the latter takes the trace of region automaton and gives the corresponding set of traces of timed automaton. With these two operations the main idea is described in Fig. 7.

The main problem associated with this method is the complexity of procedure associated with region graph construction. It has been shown in [1] that this procedure in PSPACE-hard. Once the region graph is constructed the procedure is similar to the RW framework and has a complexity proportional to product of squares of size of state space of region automaton for both plant and specification.

More or less, most of the methods involved with dense time supervisory control differ in their mechanism of conversion of timed automaton to an equivalent finite state structure. In [19], Ahmed Khoumsi and Mustapha Nourelfath proposed

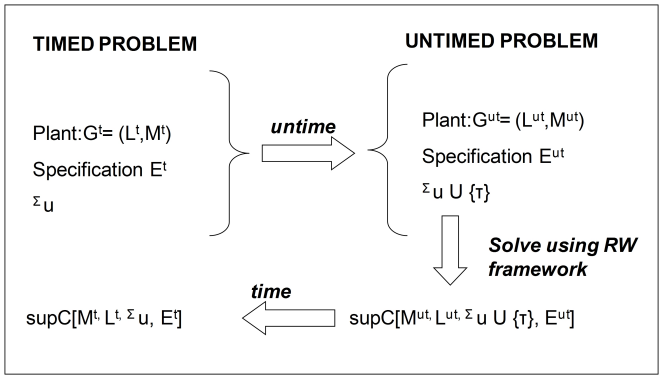

Fig. 7. Steps to compute a supervisor for a plant and specification specified as timed automaton.

a different method for transforming a timed automaton into a minimal and equivalent finite state automaton using two special types of events, Set and Exp. The main idea behind this approach is similar to the construction of timed transition graph. Since the clock value is always compared to an integer one can just measure the times to the boundary conditions of a region automaton where at least one of the clocks has an integer assignment. That means within a continuous region the enabling or disabling of an event can be decided using the predecessor boundary condition. (Refer to section II-C). Therefore, by setting the set events to expire when time reaches any boundary condition one can do away with clocks and still maintain the timed ordering of events.

Fig. 8 shows an example SE-FSA for the pedestrian bus example. Once this reduction is done, the procedure for computation of supervisor is similar to the one used by Brandin and Wonham with the knowledge that all set events are controllable and all exp events are uncontrollable. However, if an exp event is associated to a transition with another forcible event it can be preempted.

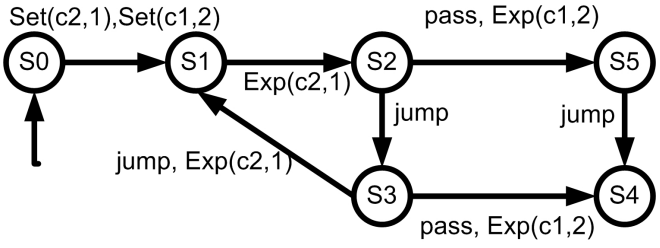

Fig. 8. The Pedestrian Bus example converted to Set Exp FSA.

Asarin et.al [24], [26] have used zone graph abstraction and posed the control problem as a game between the plant and controller. The controller wins if it can drive the plant into

a set of safe states. (invariant). This method is PSPACE-hard because the synthesis of safe state requires the construction of zone graph. In [5], Bouyer et.al. extended the Asarin's work to case when certain events are unobservable.

V. A brief introduction to supervisory control in Hybrid System Theory

Hybrid systems are usually modeled as hybrid automaton [17] and have complex continuous dynamics along with the discrete dynamics. The problem in supervisory control [20][3] of hybrid system is plagued with a problem similar to the dense real time space, infinite space. The applications vary from cruise controllers [29][14] to other more complex systems. This problem is explicitly apparent when one tries to verify the behavior of a hybrid automaton.

The basic idea for supervisory control in hybrid system remains the same. Construct a finite transition system that can conservatively approximate the behaviors of a hybrid system [2]. Then design the supervisor for the finite transition structure using a framework similar to the one used by Ramadge and Wonham.

If the construction of this finite structure is decidable as shown by Wong-Toi for linear hybrid systems in [30] the supervisory controller can be exactly computed. However, in cases where over or under approximation is required to compute the finite transition structure like in [4] the controller is not optimal.

VI. Concluding Remarks

In this paper we reviewed methods pertaining to application of supervisory control theory in real-time systems. The problems associated with denseness of time are handled either by specifying the system as a Timed Discrete Event system or by using the more expressive timed automaton formalism and then designing the supervisor using the corresponding region graph construction. The difference in complexity of the two approaches is due to the extra step involved in generating the region graph automaton from the timed automaton. This step is known to be exponential in nature.

So, which approach should be used? One can argue that in a region graph automaton the time progression event from a boundary region to a continuous region if made observable while making the timed transition from a continuous region to a boundary region unobservable, the region graph will reduce to a timed transition graph that could have been constructed by using the discrete time approximation. Thus, under such assumptions, one can choose to specify the plant as a timed transition graph if it not too big. However, if the specification difficulty has to be alleviated using the expressive timed automaton formalism, then one will have to pay the computational price as a trade off.

References

[1] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[2] R. Alur, T. Henzinger, G. Laerriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, pp. 971–984, July 2000.

[3] P. Antsaklis, X. Koutsoukos, and J. Zaytoon, "On hybrid control of complex systems: A survey," *European Journal of Automation*, vol. 32, pp. 1023–1045, 1998.

[4] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli, "Effective synthesis of switching controllers for linear systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1011–1025, 2000. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=871306

[5] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit, "Timed control with partial observability," *Lecture Notes in Computer Science*, vol. 2725, pp. 180–192, July 2003.

[6] B. Brandin and W. Wonham, "Supervisory control of timed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 39, pp. 329–342, February 1994.

[7] Y. Brave and M. Heymann, "Formulation and control of real time discrete event processes," in *Proceedings of the 27th IEEE Conference on Decision and Control*, 1988, pp. 1131–1132.

[8] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Norwell, MA, USA: Kluwer Academic Publishers, 2005.

[9] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Norwell, MA, USA: Kluwer Academic Publishers, 1999.

[10] P. C. Y. Chen and W. M. Wonham, "Real-time supervisory control of a processor for non-preemptive execution of periodic tasks." *Real-Time Systems*, vol. 23, no. 3, pp. 183–208, 2002.

[11] D. Cofer and V. Garg, "Supervisory control of real-time discrete-event systems using lattice theory," *IEEE Transactions on Automatic Control*, vol. 41, pp. 199–209, February 1996.

[12] W. W. et. al., "Xpttct." [Online]. Available: http://www.control.toronto.edu/people/profs/wonham/wonham.html

[13] E. D. Gaughan, *Introduction to Analysis*, 5th ed. Pacific Grove, CA, USA: Brooks Cole, December 1997.

[14] A. R. Girard, J. Sousa, J. A, Misener, and b. . P. a. . F. y. . . J. Karl Hedrick, title = A Control Architecture for Integrated Cooperative Cruise Control and Collision Warning Systems.

[15] P. Gohari and W. Wonham, "Reduced supervisors for timed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 48, pp. 1187–1198, July 2003.

[16] C. Golaszewski and P. Ramadge, "On the control of real-time discrete event systems," in *Proceedings of the 2rd Information Systems and Signals*, 1989, pp. 98–102.

[17] T. Henzinger, "The theory of hybrid automata," in *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS '96)*, New Brunswick, New Jersey, 1996, pp. 278–292.

[18] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic model checking for real-time systems," *Inf. Comput.*, vol. 111, no. 2, pp. 193–244, 1994.

[19] A. Khoumsi and M. Nourelfath, "An efficient method for the supervisory control of dense real-timediscrete event systems." in *The 8th International Conference on Real-Time Computing Systems and Applications (RTCSA)*, Tokyo, Japan, 2002.

[20] X. Koutsoukos, P. Antsaklis, and M. Lemmon, "Supervisory control of hybrid systems," *Proceedings of the IEEE*, vol. 88, pp. 1026 – 1049, July 2000.

[21] P. Krcál and W. Yi, "Decidable and undecidable problems in schedulability analysis using timed automata." in *TACAS*, 2004, pp. 236–250.

[22] R. Kumar and M. A. Shayman, "Supervisory control of real-time systems using prioritized synchronization." in *Hybrid Systems*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds., 1995, pp. 351–361.

[23] G. Madl, S. Abdelwahed, and G. Karsai, "Automatic verification of component-based real-time corba applications." in *RTSS*, 2004, pp. 231–240.

[24] O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems," in *STACS*, ser. Lecture Notes in Computer Science, E. W. Mayr and C. Puech, Eds., vol. 900. Springer Verlag, March 1995, pp. 229–242.

[25] J. S. Ostroff, "Deciding properties of timed transition models," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 1, no. 2, pp. 170–183, 1990. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=80145

[26] A. Pnueli, E. Asarin, O. Maler, and J. Sifakis, "Controller synthesis for timed automata," in *Proc. System Structure and Control*. Elsevier, 1998. [Online]. Available: http://citeseer.ist.psu.edu/97300.html

[27] P. Ramadge and W. Wonham, "Supervisory control of a class of discrete event processes," *Siam J. Control and Optimization*, vol. 25, no. 1, 1987. [Online]. Available: http://locus.siam.org/SICON/volume-25/art_0325013.html

[28] ——, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[29] O. Stursberg, A. Fehnker, Z. Han, and B. Krogh, "Verification of a cruise control system using counterexample-guided search," *Control Engineering Practice*, vol. 12, pp. 1269–1278, October 2004.

[30] W. H. Toi, "The synthesis of controllers for linear hybrid automata," 1997. [Online]. Available: http://citeseer.ist.psu.edu/wong-toi97synthesis.html

[31] H. Wong-Toi and G. Hoffmann, "The control of dense real-time discrete event systems," in *Proc. of 30th Conf. Decision and Control*, Brighton, UK, 1991, pp. 1527–1528.