On a simplified untiming procedure for supervisory control of timed automata when the time increases strictly monotonically

MICHAEL P. SPATHOPOULOS†

Given a timed automaton G accepting the timed language L^T a finite state machine G' can be constructed, known as the region automaton, which accepts the untimed language $\text{Untime}(L^T)$. In this paper we construct an alternative finite state machine which also accepts the language $\text{Untime}(L^T)$, but has fewer states than G' . This is shown for languages of both finite and infinite traces given that the time values in the time sequence increase strictly monotonically. The supervisory control design for timed automata, when the simplified untiming procedure is used and the time is strictly positive, is studied.

1. Introduction

Timed discrete event systems are a generalization of discrete event systems in which each event is considered to have occurred at a certain point in time. Primarily, research in this field has concentrated on the logical sequencing of events and abstracted away the actual timing delays between events. However, the correct behaviour of hard real-time systems such as manufacturing systems, transport systems and computer networks depends on the actual delay values between events. Various formalisms have been developed to model timed discrete event systems, including Alur and Dill (1994) and Brandin and Wonham (1994). Work in Brandin and Wonham (1994) corresponds to discrete time using a single system timer as opposite to Alur and Dill (1994) where the dense time model includes several asynchronous clocks and is more expressive than other formalisms allowing composition of timed processes and independent timing conditions for each system component. Here, we are concerned with timed automata as defined in Alur and Dill (1994).

The timed automaton defines (or *accepts*) a language of *timed words*. A timed word is defined as follows: given an ordinary word $\sigma_1\sigma_2\ldots\sigma_n$, we pair each letter σ_i with a non-negative real number $t_i \in \mathbf{R}^{\geq 0}$, subject to the condition that $t_i < t_{i+1}$; thus a timed word over an alphabet Σ is a finite sequence $(\sigma_1, t_1) \ldots (\sigma_n, t_n)$ with $\sigma_i \in \Sigma$ and the numbers t_i are increasing strictly monotonically. It is convenient to define $\text{Untime}((\sigma_1, t_1) \ldots (\sigma_n, t_n)) = (\sigma_1 \ldots \sigma_n)$. A *timed language* over Σ is then a set of timed words over Σ . Languages of infinite timed words are similarly defined; an infinite (timed) word is called a

(timed) ω -word, and a language of (timed) infinite words is called a (timed) ω -language.

We interpret t_i to be the time at which the event σ_i occurs; it is a convention in the theory of timed discrete event systems that an event occurs at a single point in time, whereas a system may be in a particular state for a non-trivial time interval.

Given a timed automaton G accepting the timed language L^T , it was shown in Alur and Dill (1994) that a finite state automaton G' can be constructed, known as the *region automaton*, which accepts the language $\text{Untime}(L^T)$.

The main result of this paper concerns the definition of a finite state automaton G^* that also accepts the language $\text{Untime}(L^T)$, but has significantly fewer states than G' and is therefore easier to construct. We require that the time increases *strictly* monotonically, while in Alur and Dill (1994) time is first assumed to satisfy the same monotonicity constraint (Definition 3.1), this condition is later relaxed. We first consider timed automata as acceptors of *finite* timed words, and in §5 we generalize our results to timed ω -words. Indeed, we need to distinguish languages of finite and infinite strings since the derived finite automata in each case are different, whereas in Alur and Dill (1994) the results are the same for both cases. Although the result above is of interest in computer science, we show that our result is useful for the control of real time discrete-event systems as well.

The second result concerns the application of the simplified construction of the automaton G^* for the supervisory control of timed discrete event systems. Supervisory control for timed discrete event systems has been studied in Hoffmann and Wong-Toi (1992), Brandin and Wonham (1994), Asarin *et al.* (1998), Laurence and Spathopoulos (1998) and Spathopoulos (2003). In Hoffmann and Wong-Toi (1992) and Laurence and Spathopoulos (1998) the methods of Ramadge and Wonham (1987) are adapted to construct a supervisor for a dense real-time discrete event system

Received 1 April 2003. Revised and accepted 28 September 2003.

†System Dynamics and Control Department of Mechanical Engineering, University of Strathclyde, James Weir Building, 75 Montrose Street, Glasgow G1 1XJ, UK. e-mail: mps@mecheng.strath.ac.uk

modelled by a timed automaton using untiming procedures. In Spathopoulos (2003) the non-blocking supervisory control problem with urgency has been studied and in Asarin *et al.* (1998) the same problem is addressed without using untiming procedures.

When the time increases strictly monotonically, as is always the case in reality, the simplified automaton derived below, which applies for the finite string case, can be used for the solution of the supervisory control problem. We show that when this simplified technique is used and the time is strictly positive it is necessary for the controlled behaviour to become more conservative. Intuitively this is to be expected since allowing the time to be zero gives more flexibility to the model.

The paper is organized as follows. In §§ 2 and 3 the timed automata and the construction of the region (untimed) automaton, as given in Alur and Dill (1994), are briefly introduced.

The main result of the paper is given in § 4 and in § 5 the result is extended for infinite strings. Upper bounds for the clock regions of the derived automata are given in § 6. The second result is given in § 7 where the simplified untiming technique is used for the supervisory control problem of real time discrete-event systems. An illustrated example is used in order to derive the conditions under which the simplified construction results in the least restrictive controlled behaviour. Finally, conclusions are given in § 8.

2. The timed automaton

In this section we give a short overview of the timed automaton as defined in Alur and Dill (1994), where the reader is referred for more details. Our purpose here is not to give intuitive explanations of definitions in timed automata and interested readers should be familiar with them.

For simplicity we consider timed automata as acceptors of finite words. Our results are extended for languages of infinite words in § 5.

Definition 1: A timed automaton is a tuple

$$G = (\Sigma, S, C, E, S_0, S_m) \quad (1)$$

where Σ , as usual, is a finite alphabet, S is a finite set of states, C is a finite set of *clocks* and E is a finite set of *edges*. The sets $S_0, S_m \subseteq S$ are called the *initial* and *final* (or *accepting*) state sets respectively.

An edge is a tuple $(s, s', \sigma, \delta, C')$, alternatively written

$$s \xrightarrow[\delta]{\sigma, C' := 0} s' \quad (2)$$

Here $C' \subseteq C$ and δ is what is known as a *clock constraint*. A clock constraint is a predicate (that is, a condition) built from the atomic predicates $c * q$ (where

$c \in C$ and $*$ is one of $\leq, \geq$ or $=$ and $q \in \mathbf{Z}$) using $\neg$ (not) and the connectives $\wedge$ (and) and $\vee$ (or). For instance, if $c, d \in C$ then $c \geq 1 \vee d = 2$ is an example of a predicate.

We need further definitions as follows.

Definition 2: A (*clock*) *valuation* is a function $v : C \rightarrow \mathbf{R}^{\geq 0}$.

Definition 3: A clock valuation $v : C \rightarrow \mathbf{R}^{\geq 0}$ is said to satisfy a clock constraint δ if replacing each clock c in δ by $v(c)$ gives a true statement.

Definition 4: If $v : C \rightarrow \mathbf{R}^{\geq 0}$ is a clock valuation and $t \in \mathbf{R}$, then $v + t$ is the clock valuation given by $(v + t)(c) = v(c) + t$ for all clocks c . Also, if $t > 0$ then $v + t$ is said to be a *successor* of v .

Definition 5: If $v : C \rightarrow \mathbf{R}^{\geq 0}$ is a clock valuation and $C' \subseteq C$ then $[C' = 0]v$ is the clock valuation which sends all clocks in C' to zero and sends all the others to the same value as v .

Essentially, the timed automaton is an infinite state automaton whose states are ordered pairs (s, v) where $s \in S$ is a state and $v : C \rightarrow \mathbf{R}^{\geq 0}$ is a clock valuation. We will call such pairs *generalized (or extended) states*. When the timed automaton is G regarded as an infinite state automaton, its initial (generalized) states are those ordered pairs (s, v) where $s \in S_0$ and the valuation sends every clock to zero. The final states are those for which $s \in S_m$. The set of transitions between generalized states is defined as follows.

Given the edge in (2) and clock valuations v, v' , there is a legal transition with label σ from (s, v) to (s', v') if there exists $t > 0$ and $v' = [C' = 0](v + t)$. We call t the *time jump* of the transition at the state s . We also define the empty timed word $(\varepsilon, 0)$, where ε is the empty word. All transactions are of this type, or are trivial; that is, there is a transition with label ε from every generalized state to itself.

It is worth mentioning again that it is possible to allow zero time jumps in the definition of a transition. This was considered by Alur and Dill (1994) in order to allow the same time value to be associated with many consecutive events in the time sequence. However, in this case the main result of this paper, the construction of an untimed automaton G' with fewer states than the region automaton is impossible, since Lemma 3 of § 4 is false.

Having defined G as a state machine, we now explain how G can be used to define a timed language. Let $(\sigma_1, t_1) \dots (\sigma_n, t_n)$ be a finite timed word. Then we say that G *accepts* $(\sigma_1, t_1) \dots (\sigma_n, t_n)$ if G accepts the untimed word $\sigma_1 \sigma_2 \dots \sigma_n$ as a state machine and if

$(s_0, v_0) \xrightarrow{\sigma_1} (s_1, v_1) \xrightarrow{\sigma_2} (s_2, v_2) \cdots (s_{n-1}, v_{n-1}) \xrightarrow{\sigma_n} (s_n, v_n)$ is a corresponding accepting path for $\sigma_1 \sigma_2 \cdots \sigma_n$ (recall that this means that $s_0 \in S_0$, $s_n \in S_m$ and the valuation v_0 sends every clock to zero) then $t_i - t_{i-1}$ is the time jump for the transition (between generalized states) $(s_{i-1}, v_{i-1}) \xrightarrow{\sigma_i} (s_i, v_i)$. The timed language accepted by G is then defined as the set of timed words accepted by G . If G accepts the *timed* language L^T , then we also say that G accepts the *untimed* language $\text{Untime}(L^T)$.

3. The Alur–Dill untiming construction

In this section for the sake of completeness we state briefly the region automaton as constructed in Alur and Dill (1994).

The timed automaton has one important disadvantage in computations; even if the state set S is finite, there will necessarily be infinitely many generalized states (assuming there is at least one clock) simply because there are infinitely many clock valuations. This problem is avoided by partitioning the set of clock valuations into equivalence classes in such a way that there are finitely many such classes and the essential structure of the timed automaton is captured.

An equivalence relation, $\equiv$, is defined on the set Val of clock valuations of A having the following two properties: there are finitely many equivalence classes, and if there exist clock valuations v_1, v_2 and κ_1 with $v_1 \equiv v_2$ and there is a legal transition from (s, v_1) to (s', κ_1) , then there is a clock valuation κ_2 with $\kappa_1 \equiv \kappa_2$ and a legal transition from (s, v_2) to (s', κ_2) .

Definition 6: Let G be the timed automaton of (1) in Definition 1.

For each clock $x \in C$, let c_x be the largest integer with which x is compared in any clock constraint in any edge.

Define an equivalence relation $\equiv$ on the set of clock valuations as follows: given two valuations $v, \kappa : C \rightarrow \mathbb{R}^{\geq 0}$, $v \equiv \kappa$ iff:

- for all clocks $x \in C$, either $\text{int}(v(x)) = \text{int}(\kappa(x))$ or $v(x), \kappa(x) > c_x$;
- for all clocks $x \in C$, $v(x)$ is an integer $\leq c_x$ if and only if $\kappa(x)$ also satisfies this condition;
- for all clocks $x, y \in C$, if $v(x), \kappa(x) \leq c_x$ and $v(y), \kappa(y) \leq c_y$ then $\text{frac}(v(x)) \leq \text{frac}(v(y))$ implies $\text{frac}(\kappa(x)) \leq \text{frac}(\kappa(y))$.

Then the equivalence classes defined by $\equiv$ are known as *clock regions*. If v is a valuation, then $\text{reg}(v)$ is the clock region containing v . It should be noted that $C' \subseteq C$ if is a set of clocks then $v \equiv \kappa$ implies $[C' = 0]v \equiv [C' = 0]\kappa$.

The following lemma is fundamental to the motivation of the definition of the region automaton G' and of Definition 8 of §4.

Lemma 1: Let r_1, r_2 be clock regions and let $v, \kappa \in r_1$. If $v + t \in r_2$, for some $t \geq 0$, then there exists $t' \geq 0$ such that $\kappa + t' \in r_2$.

Proof: We will prove the lemma by induction on $|\{\text{reg}(v + a') \mid a' \in [0, t]\}|$. If this set is empty then $r_1 = r_2$ and the lemma is obvious. Hence we may assume that $r_1 \neq r_2$. Thus necessarily $v(x) \leq c_x$ for at least one $x \in C$. We define $K = \{x \in C \mid v(x) \leq c_x \text{ and } v(x) \in \mathbb{Z}\}$ we consider two cases.

1. $K = \emptyset$. Define

$$t_1 = \min\{\text{int } v(x) + 1 - v(x) \mid x \in C$$

and $v(x) \leq c_x\}$ and let

$$t'_1 = \min\{\text{int } \kappa(x) + 1 - \kappa(x) \mid x \in C \text{ and } \kappa(x) \leq c_x\}$$

Then $v + t_1 \equiv \kappa + t'_1$.

2. $K \neq \emptyset$. Choose any positive

$$t_1 < \min(t, \min\{\text{int } v(x) + 1 - v(x) \mid x \in C \text{ and } v(x) \leq c_x\})$$

and positive

$$t'_1 < \min\{\text{int } \kappa(x) + 1 - \kappa(x) \mid x \in C \text{ and } \kappa(x) \leq c_x\}$$

Here also $v + t_1 \equiv \kappa + t'_1$.

In both cases we have $\text{reg}(v) \neq \text{reg}(v + t_1) = \text{reg}(v + t'_1)$. The result now follows from the inductive hypothesis applied to the valuations $v + t'_1, \kappa + t'_1$ and $t - t_1$ in place of t since $|\{\text{reg}(v + t_1 + a') \mid a' \in [0, t - t_1]\}| < |\{\text{reg}(v + a') \mid a' \in [0, t]\}|$. $\square$

Lemma 2: Given two states $s, s' \in S$ and clock valuations v_1, v_2, κ_1 with $v_1 \equiv v_2$ and a legal transition between generalized states

$$(s, v_1) \xrightarrow{\sigma} (s', \kappa_1)$$

then there exists a clock valuation κ_2 with $\kappa_2 \equiv \kappa_1$ and a legal transition

$$(s, v_2) \xrightarrow{\sigma} (s', \kappa_2)$$

Proof: This follows immediately from Lemma 1 and the definition of a transition between generalized states in a timed automaton. $\square$

From this result it follows that we can replace the set of clock valuations by the set of clock regions, thus motivating the following definition.

Definition 7: Let G be the timed automaton defined in (1) in Definition 1. We define G' to be the finite state machine

$$G'(\Sigma, (S \times R), T, S_0 \times \mathbf{0}, S_m \times R)$$

where R is the set of clock regions of G and $\mathbf{0}$ is the clock region whose unique valuation sends all clocks to zero.

We define the set of T transitions as follows. There is a transition

$$(s, r_1) \xrightarrow{\sigma} (s', r_2)$$

for two clock regions r_1, r_2 iff there exist clock valuations v_1, v_2 with $v_i \in r_i$ and there is a transition in G between generalized states

$$(s, v_1) \xrightarrow{\sigma} (s', v_2)$$

G' is called the *region automaton*.

The following is the main result of this section. The proof, which may be inferred from Lemma 2, is omitted.

Theorem 1: Let L^T be the timed language accepted by G . Then $\text{Untime}(L^T)$ is the language accepted by the finite state automaton G' .

The importance of G' is that there are finitely many clock regions (this follows from Definition 6 and the fact that the clock set C is finite), and so G' always has finitely many states.

4. A new finite automaton for languages of finite strings

This partition of the set of clock valuations Val defined above has the following properties:

- There are finitely many clock regions.
- Let $v \in \text{Val}$ and $\omega \in \text{reg}(v)$. Then ω and v satisfy the same set of lock constraints. If $t \geq 0$ then there exists $t' \geq 0$ such that $\text{reg}(v + t) = \text{reg}(\omega + t')$.
- if $0 \leq t_1 \leq t_2 \leq t_3$ and $v \in \text{Val}$ and $\text{reg}(v + t_1) = \text{reg}(v + t_3)$ then $v + t_2 \in \text{reg}(v + t_3)$.
- if $\omega \in \text{reg}(v)$ and $C' \subseteq C$ is a set of clocks then $[C' = 0]\omega \in \text{reg}[C' = 0]v$.

Given a partition satisfying these properties, and a clock region $r \in R$, we define $\text{succ}(r) \in R$, the successor region to $r \in R$, as follows. If $t \geq 0$ implies $\text{reg}(v + t) = \text{reg}(v)$ then $\text{succ}(r)$ is undefined. Otherwise let $\text{succ}(r) = \text{reg}(v + t)$ for $v \in r$ and $t \geq 0$ chosen such that $v \notin \text{reg}(v + t)$ and $0 \leq t' \leq t$ implies

that $\text{reg}(v + t') = \text{reg}(v + t)$ or $\text{reg}(v + t') = \text{reg}(v)$. Owing to the third property listed above and the fact that there are finitely many clock regions, such a value of $t \geq 0$ can be found; and owing to the second property listed, the choice of $v \in r$ is irrelevant. We will write $\text{succ}^n(r) = \text{succ}(\text{succ}^{n-1}(r))$ for $n \geq 1$.

The following definition introduces the notion of *closed* and *non-closed* regions and will be used in this section for the simplification of the untiming procedure.

Definition 8: Let r and r' be clock regions. We say that r' is a successor of r if r' contains a valuation which is a successor of a valuation lying in r . By Lemma 1, this implies that every element of r has a successor in r' . We define $\text{next}(r)$ to be the *minimal* successor of r ; that is, the unique successor of r such that all other successors of r are also successors of $\text{next}(r)$.

If any (and hence every) valuation in r maps any clock $x \in C$ to an integer $\leq c_x$ and $v \in r$, then $\text{next}(r)$ is defined as the clock region containing the valuations $\{v + \varepsilon \mid 0 < \varepsilon < 1 - \text{frac}(v(y)), \forall y \in C \text{ such that } v(y) < c_y\}$. In other words, for the clock region in which any valuation $v \in r$ maps any clock to an integer, $\text{next}(r)$ is the unique *minimal* time successor region of r . We call such clock regions *closed*.

On the other hand, if the valuations in r do not map any clock $x \in C$ to an integer $\leq c_x$, then we define $\text{next}(r) = r$. In other words, in this case, the clock region r has itself a minimal time successor region. We call such clock regions *non-closed*. Intuitively a region is non-closed or open when time can pass without leaving the region.

Observe that $\text{next}(\text{next}(r)) = \text{next}(r)$ for all clock regions r .

Example 1: Consider a timed automaton with two states s_1, s_2 , one clock x and possible transition $s_1 \xrightarrow{a} s_1$, $1 < x < 2$. *Closed* regions are $\{x = 0\}$, $\{x = 1\}$, $\{x = 2\}$ and *non-closed* regions are $\{0 < x < 1\}$, $\{1 < x < 2\}$ and $\{2 < x\}$. Moreover $\text{next}\{x = 0\} = \{0 < x < 1\}$, $\text{next}\{x = 2\} = \{1 < x < 2\}$ and $\text{next}\{x = 2\} = \{2 < x\}$. For a more general case of two locks the reader is referred to figure 1, where the bisimilar quotient state space (the clock regions) associated with Example 4.4 of Alur and Dill (1994) is given (the same example is used again in §6). The *closed* regions are associated with line segments (except for the line segments 2 and 5) and points. The *non-closed* regions are associated with the inside area of the rectangles and the line segments 2 and 5 that have slope 1. The arrow indicate the next (non-closed) region (minimal successor) of the associated *closed* region. For example $\text{next}\{(x, y)/x = y = 0\} = \{(x, y)/0 < x < 1, x = y\}$ and $\text{next}\{(x, y)/x = 0, 0 < y < 1\} = \{(x, y)/0 < y < 1, 0 < x < y\}$.

Below we give a different procedure for defining the untimed language of G . We prove that the clock regions of G which are *closed* may be deleted from G' , provided certain modifications are made to the transitions of G' . Thus, we define a new finite automaton G^* that has considerable fewer states than G' .

The main result of this paper is based on the following lemma.

Lemma 3:

1. Suppose that there is a transition of the following form between the states of G'

$$(s, r_1) \xrightarrow{\sigma} (s', r_2) \quad (3)$$

Then there is also a transition

$$(s, \text{next}(r_1)) \xrightarrow{\sigma} (s', r_2) \quad (4)$$

2. Conversely, the existence of the transition in (4) implies the existence of that in (3).

Proof: We first assume that the transition (3) exists and prove that (4) does. Following (3) there are valuations τ_i for $i = 1, 2$ such that $\tau_1 \in r_1$, $\tau_2 = \tau_1 + t$, $t > 0$ and a valuation $\kappa = [C' = 0]\tau_2$ which means that $\kappa \in r_2$. Clearly $\text{reg}(\tau_2)$ is a successor of r_1 and since the time jump t is strictly positive it is also a successor of $\text{next}(r_1)$. This implies that there is a transition from $\text{next}(r_1)$ to r_2 . More specifically let $0 < \varepsilon < t$. Then we have the following regions in the order below

$$\begin{aligned} r_1 = \text{reg}(\tau_1) &\rightarrow \text{next}(\text{reg}(\tau_1)) \rightarrow \text{reg}(\tau_1 + \varepsilon) \\ &\rightarrow \text{reg}(\tau_2) \rightarrow r_2 \end{aligned}$$

and this shows that there exists a transition from $\text{next}(r_1)$ to r_2 . Note that it might be that $\text{next}(r_1) = \text{reg}(\tau_1 + \varepsilon)$. The converse statement follows from the fact that any successor of $\text{next}(r_1)$ is also a successor of r_1 .

Definition 9: Define the finite state automaton G^* as follows. G^* is obtained from G' by *deleting* all states (s, r) such that the region r is *closed*, *deleting* all transitions incident to these states and *adding* in transitions of the form

$$(s, \text{next}(r_1)) \xrightarrow{\sigma} (s', \text{next}(r_2))$$

for every transition in G' of the form

$$(s, r_1) \xrightarrow{\sigma} (s', r_2)$$

The following is the main result of this section.

Theorem 2: G^* and G' accept the same language.

Proof: Let $w = \sigma_1 \cdots \sigma_n$ be accepted by G' . This means that there is a sequence of transitions in G'

$$(s_0, \mathbf{0}) \xrightarrow{\sigma_1} (s_1, r_1) \cdots (s_{n-1}, r_{n-1}) \xrightarrow{\sigma_n} (s_n, r_n)$$

for clock regions r_i of G and with $s_0 \in S_0, s_n \in S_m$.

By the definition of the automaton G^* , there is a sequence of transitions in G^*

$$(s_0, \text{next}(\mathbf{0})) \xrightarrow{\sigma_1} (s_1, \text{next}(r_1)) \cdots (s_{n-1}, \text{next}(r_{n-1})) \xrightarrow{\sigma_n} (s_n, \text{next}(r_n))$$

and thus w is accepted by G^* .

To show the converse, again let $w = \sigma_1 \cdots \sigma_n$ and assume that w is accepted by G^* . By the definition of the automaton G^* , this means that there is a sequence of transitions in G^*

$$(s_0, \text{next}(\mathbf{0})) \xrightarrow{\sigma_1} (s_1, \text{next}(r_1)) \cdots (s_{n-1}, \text{next}(r_{n-1})) \xrightarrow{\sigma_n} (s_n, \text{next}(r_n)) \quad (5)$$

for clock regions r_i of G and with $s_0 \in S_0, s_n \in S_m$. We now show by induction on i that for all $i \leq n$ there is the following sequence of transitions in G' , for clock regions r'_j satisfying $\text{next}(r'_j) = \text{next}(r_j)$

$$(s_0, \mathbf{0}) \xrightarrow{\sigma_1} (s_1, r'_1) \cdots (s_{i-1}, r'_{i-1}) \xrightarrow{\sigma_i} (s_i, r'_i) \quad (6)$$

For $i = 0$ this is trivial. Assume (6) holds for some $i < n$. From (5) and the definition of G^* , there is a transition $(s_i, r''_i) \xrightarrow{\sigma_{i+1}} (s_{i+1}, r'_{i+1})$ in G' with $\text{next}(r''_i) = \text{next}(r_i) = \text{next}(r'_i)$ and $\text{next}(r'_{i+1}) = \text{next}(r_{i+1})$. By Lemma 3.1 there is a transition $(s_i, \text{next}(r''_i)) \xrightarrow{\sigma_{i+1}} (s_{i+1}, r'_{i+1})$ in G' ; hence by Lemma 3.2 there is a transition $(s_i, r'_i) \xrightarrow{\sigma_{i+1}} (s_{i+1}, r'_{i+1})$ in G' . Thus we have proved the inductive step, and hence proved that w is accepted by G' . $\square$

Remark: Our result can be summarized as follows. For any *closed* region, since time must take place (leading to $\text{next}(r)$) before any further transition, we can consider r and $\text{next}(r)$ to be *bisimilar*.

Transitions that are enabled at exactly the *closed* region r are covered by regions ‘*after*’ r , i.e. for r being $\{x = 1\}$, the region $\{1 < x < 2\}$ will include all transitions which are enabled at $\{x = 1\}$. Therefore we may consider the two regions together given by $\{1 \leq x < 2\}$. In other words, we may identify together the (extended) states whose clock regions have the same *next*-value.

Example 2: Consider a timed automaton with two states s_1 and s_2 , one clock x and possible transitions $s_1 \xrightarrow{a} s_1$, $1 < x < 2$ and $s_1 \xrightarrow{b} s_2$, $x = 1$. Initially the state s_1 is active and $x = 0$. The new finite automaton G^* will involve the transitions $(s_1, \{0 \leq x < 1\}) \xrightarrow{a} (s_1, \{1 \leq x < 2\})$, $(s_1, \{1 \leq x < 2\}) \xrightarrow{b} (s_1, \{1 \leq x < 2\})$, $(s_1, \{0 \leq x < 0\}) \xrightarrow{b} (s_2, \{1 \leq x < 2\})$

5. Extension of results to acceptors of infinite words

In this section we construct another automaton, similar to G^* , in order to simplify the untiming procedure in the case where the timed automata are regarded as acceptors of ω -words.

Definition 10: A *timed Buechi automaton* (Alur and dill 1994, Definition 3.10) is a tuple

$$H = (\Sigma, S, C, E, S_0, S_m)_B$$

where the symbols have the same meanings as in (1) of Definition 1. But H is regarded as an acceptor of timed ω -words as follows: let $w = (\sigma_1, t_1) \cdots (\sigma_i, t_i) \cdots$ be a timed ω -word. Then H accepts w provided the following two conditions hold:

1. There is an infinite sequence of transitions $(s_0, 0) \xrightarrow{\sigma_1} (s_1, t_1) \cdots (s_{i-1}, t_{i-1}) \xrightarrow{\sigma_i} (s_i, t_i) \cdots$ through H with $s_0 \in S_0$ and $s_i \in S_m$ for infinitely many values of i .
2. $\lim_{i \rightarrow \infty} t_i = \infty$. This is called the *progress condition*.

The timed ω -language accepted by H is then defined to be the set of timed ω -words accepted by H . If H defines the timed ω -language L^T then we say that H accepts the *untimed* ω -language $\text{Untime}(L^T)$, by analogy with ordinary timed automata.

Lemma 4: Let $\cdots(\sigma_i, t_i) \cdots$ be an infinite timed word which defines a path through the sequence of generalized states $\cdots(s_i, v_i) \cdots$ of H . Then there is a word $\cdots(\sigma_i, t'_i) \cdots$ passing through the same sequence of states of H and satisfying the progress condition if and only if for every clock $x \in C$, we have $v_i(x) = 0$ or $v_i(x) > c_x$ for infinitely many values of i . This is shown in Alur and Dill (1994, Lemma 4.13).

Definition 11: A *timed Muller automaton* is a tuple $M = (\Sigma, S, C, E, S_0, F)_M$ which is identical to a Buechi timed automaton except that the accepting set S_m is replaced by a collection F of subsets of the state set S , and instead of the condition (1), to be accepted by M a timed word must have a path through M which passes infinitely many times through the sets in $P \subseteq S$ and finitely many times through the sets in $S - P$, for some set $P \in F$.

Next we define the Muller automaton $M' = (\Sigma, S \times R, T, S_0 \times \mathbf{0}, F)_M$ where F contains the following subsets of the state set $S \times R$; F contains all those subsets of $S \times R$ which intersect non-trivially with $S_m \times R$ and with all sets $S \times R_x$ for $x \in C$, where R_x is the set of all clock regions whose valuations map the clock x to zero or to a number greater than c_x .

Theorem 3: M' accepts the same ω -language as H .

Proof: Let $\sigma_1 \cdots$ be an untimed ω -word. Then both conditions of Definition 10 are satisfied for some time sequence $\cdots t_i \cdots$ if and only if there is a path with label $\sigma_1 \cdots$ through M' starting at $S_0 \times \mathbf{0}$ and also satisfying the following conditions: the path passes infinitely often through the state set $S_m \times R$ and it passes infinitely often through the sets $S \times R_x$ for all $x \in C$, by Lemma 4. Thus the result follows immediately from the definition of M' .

We mimic the construction of M' using a modified version of the definition of $\text{next}(r)$.

We define a Muller automaton M^* obtained from M' in a similar way in which G^* was obtained from G' . $\square$

Definition 12: The Muller automaton M^* is defined as $M^* = (\Sigma, S^*, T^*, S_0^*, S_m^*)_M$ where

- the state set S^* contains all elements of $S \times R$ of the form $(s, \text{next}(r))$, where r is any clock region, or of the form (s, r) , where r is a clock region whose valuations send any clock to zero.
- The transition set T^* is as follows: for every transition in the region automaton G' of the form $(s, r_1) \xrightarrow{\sigma} (s', r_2)$ there is a transition in T^* of the form $(s, \text{sec}(r_1)) \xrightarrow{\sigma} (s', \text{sec}(r_2))$ where the clock region $\text{sec}(r)$ is defined to be $\text{next}(r)$ unless r maps any clock to zero, in which case $\text{sec}(r) = r$.
- $S_0^* = S_0 \times \mathbf{0}$ is the set of initial states.
- The accepting state sets are those which intersect non-trivially with $S_m \times \text{sec}(R)$ and with all sets $S \times \text{sec}(R_x)$ for all $x \in C$.

Thus M^* 'lies between' G' and G^* in the sense that it is obtained from G' by eliminating *some* of the non-closed clock regions.

Theorem 4: H and M^* accept the same language.

Proof: Since the clock region $\text{sec}(r)$ is always either $\text{next}(r)$ or r , Lemma 3 still holds with $\text{next}(r)$ replaced by $\text{sec}(r)$. The result therefore follows from an argument similar to that in Theorem 2. $\square$

6. Upper bounds for numbers of clock regions

In Alur and Dill (1994, Lemma 4.5) an upper bound of

$$|C|!2^{|C|-1} \prod_{x \in C} (2c_x + 2)$$

is given for the number of clock regions of the timed automaton G . The number of *non-closed* clock regions

for the finite string case can similarly be shown to be bounded by

$$|C|!2^{|C|-1} \prod_{x \in C} (c_x + 1)$$

To prove this, observe that the clock region of G is uniquely specified by the following:

- (a) for each clock x , one clock constraint from the set

$$\{x = c | c = 0, \dots, c_x\} \cup \{x \in (c - 1, c) | c = 1, \dots, c_x\} \cup \{x > c_x\}$$

- (b) an ordering of the integers $\{1, 2, \dots, |C|\}$ to indicate an ordering of the numbers $\text{frac}(x)$ for $x \in C$,
- (c) for each integer $n \in \{2, 3, \dots, |C|\}$, whether $\text{frac}(x) > \text{frac}(y)$ or $\text{frac}(x) = \text{frac}(y)$, where x and y are the n th and $n - 1$ th clocks in the ordering given in (b).

There are $|C|!$ choices in (b) and $2^{|C|-1}$ in (c). Case (a) gives $\prod_{x \in C} (2c_x + 2)$ choices for all regions, but if only closed regions need to be listed, then the first set given in (a) is irrelevant, giving $\prod_{x \in C} (c_x + 1)$ choices. Multiplying the results gives the upper bounds claimed above. Thus replacing G' by G^* reduces the upper bound on the number of states by a factor of $2^{|C|}$.

We consider the timed automaton example with two clocks given in Alur and Dill (1994, Example 4.4). The region automaton involves 28 clock regions, of which only 10 are closed. Therefore in the new automaton we may delete the *closed* regions or consider them together with the associated *non-closed* regions. In figure 1, the associated *non-closed* regions just ‘after’ the *closed* regions are shown using arrows. The 10 (non-closed) regions needed for the new finite automaton are illustrated in figure 1 using bold numbers. Note that only 2 and 5 are line segments. The derived upper bounds are rather conservative. Thus, in practice and for specific problems the non-closed regions are much fewer than

Figure 1.

the closed ones. This results in deriving significantly fewer extended states for the new finite automaton.

It is not difficult to conclude that the number of *non-closed* clock regions in the infinite string case is bounded by $|C|2^{|C|-1} \prod_{x \in C} (c_x + 2)$. Thus replacing G' by M^* reduces again the upper bound on the number of states by a factor of almost $2^{|C|}$. For the above examples we must now include all *closed* regions associated with the clock axis x and y and illustrated in figure 1 as 1i–9i. This allows the valuations to be reset infinitely many times to zero required by Lemma 4. Now 19 clock regions are needed.

7. On supervisory control for dense real-time discrete-event systems

In this section the application of the derived simplified technique to the supervisory control design and its effect in obtaining the least restrictive controlled behaviour are given.

For simplicity we consider the timed automaton $A = (S, s_0, \Sigma, E, C)$ similar to the one defined in (1) but without accepting (marked) states. In this section, we are only interested in timed automata as acceptors of languages of *finite* timed words. We assume that A is *time deterministic*; that is, it satisfies the following condition; besides having only one initial state, for any so defined *extended state* (s, v) , where $v : C \rightarrow R^{\geq 0}$ are the *clock valuations*, and for any event $\sigma \in \Sigma$ there is a maximum of one edge $(s, s', \sigma, \delta, C')$ in A such that v satisfies the clock constraint δ . This implies that for any timed word $(\sigma_1, t_1) \cdots (\sigma_n, t_n)$ there is a maximum of one path starting at the initial state s_0 over the timed word $(\sigma_1, t_1) \cdots (\sigma_n, t_n)$ through A .

We denote Ω the set of extended states of A and as in the case of a finite state automaton, we partition the event alphabet into controllable and uncontrollable events: $\Sigma = \Sigma_c \cup \Sigma_u$. The *control function* for A is defined as $\mu: \Omega \times \Sigma \rightarrow \{0, 1\}$ satisfying $\mu(e, \sigma) = 1$ if $\sigma \in \Sigma_u$ and e is any extended state. We call the pair (A, μ) a *controlled timed automaton*. We define the timed language $\text{Ltimed}(A, \mu)$ to be the set of all timed words $(\sigma_1, t_1) \cdots (\sigma_n, t_n)$ such that there is a path

$$(s_0, \mathbf{0}) \xrightarrow{(\sigma_1, t_1)} (s_1, v_1) \cdots \xrightarrow{(\sigma_n, t_n)} (s_n, v_n)$$

where the clock valuation $\mathbf{0}$ sends all clocks to zero, satisfying the condition $\mu(s_{i-1}, v_{i-1} + t_i, \sigma_i) = 1$ for all i . If such a path exists we say that the extended states (s_i, v_i) are *accessible* under μ .

Let F (for *forbidden*) be a set of extended states (possibly infinite, since the number of valuations are infinite) of A . The supervisory control problem that we are dealing is the following (forbidden control problem):

Find the control function μ such that every element of F is inaccessible under μ and such that $\text{Ltimed}(A, \mu)$ is maximal for all such control functions μ .

For simplicity the non-blocking condition is not included assuming that the languages are closed. The interested reader is referred to Spathopoulos (2003) for the non-blocking forbidden control problem.

It has been shown (Hoffman and Wong-Toi 1992, Laurence and Spathopoulos 1988) that this can be solved considering the control problem for the finite state automaton $\text{Untime}(A)$ (as defined in the following definition) provided that the forbidden state set F satisfies a certain condition stated in Theorem 5.

Definition 13: Let R be the set of clock regions of A . Then the finite state automaton $\text{Untime}(A)$ has state set $S \times R$, initial state $(s_0, \mathbf{0})$, event alphabet $\Sigma \cup \{\tau\}$ and transitions of two kinds. There are transitions $(s, r) \xrightarrow{\tau} (s, \text{succ}(r))$ from every state (s, r) where $\text{succ}(r)$ is the successor region to r and there are transitions $(s, r) \xrightarrow{\sigma} (s', [C' = 0](r'))$ for every edge $(s, s', \sigma, \delta, C')$ in A such that the elements of the clock region r satisfy the clock constraint δ .

Remark: The automaton $\text{Untime}(A)$ is similar to G' where the event τ that models the 'time passage' has been added in the alphabet. Basically we use the *time-abstract* transition system for which the set of events is $\Sigma \cup \{\tau\}$. The time passage is treated in this paper as an uncontrollable event. The reader is referred to Spathopoulos (2003) where the notion of urgency is introduced and the time passage may be defined as a controllable event as well.

It can be shown that A *time deterministic* implies $\text{Untime}(A)$ *deterministic*. This was the reason for requiring A to be time deterministic.

It is useful to make the following definition: if (s, v) is an extended state then let $\rho(s, v) = (s, \text{reg}(v))$.

Lemma 5:

- (1) If there is a legal transition $(s, v) \xrightarrow{\sigma} (s', v')$ in A with time jump t then there is a path

$$(s, \text{reg}(v)) \xrightarrow{\tau} (s, \text{succ}(\text{reg}(v))) \xrightarrow{\tau} \dots \xrightarrow{\tau} (s, \text{succ}^n(\text{reg}(v))) \xrightarrow{\sigma} (s', \text{reg}(v'))$$

through $\text{Untime}(A)$ for $\text{succ}^n(\text{reg}(v)) = \text{reg}(v + t)$.

- (2) If there is a transition $(s, r) \xrightarrow{\sigma} (s', r')$ in $\text{Untime}(A)$ with $\sigma \in \Sigma$ and $v \in r^*$ for some $v \in \text{Val}, r^* \in R$ and $\text{succ}^n(r^*) = r$ then there is a transition $(s, v) \xrightarrow{\sigma} (s', v')$ for some $v' \in r'$.

Proof: Follows from Definition 13 and Lemma 2. $\square$

Given a timed word $(\sigma_1, t_1) \dots (\sigma_n, t_n)$ defining a path $\dots (s_i, v_i) \xrightarrow{\sigma_i} (s'_{i+1}, v'_{i+1}) \dots$ through A , the *projection* of this word is defined to be the label of the corresponding path in $\text{Untime}(A)$ given by combining the path-segments of the form given in Lemma 5(1).

The following theorem states that the supervisory control problem for timed automata can be solved using the automaton $\text{Untime}(A)$.

Theorem 5: Let μ be a control function for the timed automaton A , and let F be a set of extended states of A such that if $(s, v) \in F$ and $v' \in \text{reg}(v)$ then $(s, v') \in F$ (or equivalently, $\rho^{-1}(\rho(F)) = F$) and let $\pi: S \times R \times (\Sigma \cup \{\tau\}) \rightarrow \{0, 1\}$ be a control function for the finite state automaton $\text{Untime}(A)$, with the 'time passage' event τ classed as uncontrollable. Define the control function $\mu: \Omega \times \Sigma \rightarrow \{0, 1\}$ as $\mu(s, v, \sigma) = \pi(s, \text{reg}(v), \sigma)$. Assume that the states $\rho(F)$ are inaccessible in $(\text{Untime}(A), \pi)$; then the extended states in F are inaccessible in (A, μ) ; furthermore, if $\text{L}(\text{Untime}(A), \pi)$ is maximal with this property, then so is $\text{Ltimed}(A, \mu)$.

Proof: Suppose that an extended state (s_n, v_n) is accessible in (A, μ) ; then there is a path

$$(s_0, \mathbf{0}) \xrightarrow{(\sigma_1, t_1)} (s_1, v_1) \dots \xrightarrow{(\sigma_n, t_n)} (s_n, v_n)$$

through A satisfying $\mu(s_{i-1}, v_{i-1} + t_i, \sigma_i) = 1$ for all i . Thus from multiple applications of Lemma 5(1) there is a path

$$(s_0, \mathbf{0}) \dots (s_{i-1}, \text{reg}(v_{i-1})) \xrightarrow{\tau} (s_{i-1}, \text{succ}(\text{reg}(v_{i-1}))) \xrightarrow{\tau} \dots \xrightarrow{\tau} (s_{i-1}, \text{succ}^{n_{i-1}}(\text{reg}(v_{i-1}))) \xrightarrow{\sigma_i} (s_i, \text{reg}(v_i)) \dots (s_n, \text{reg}(v_n))$$

through $\text{Untime}(A)$ for $\text{succ}^{n_i}(\text{reg}(v_i)) = \text{reg}(v_i + t_{i+1})$ with $\pi(s_{i-1}, \text{reg}(v_{i-1} + t_i), \sigma_i) = 1$ for all i . Thus $(s_n, \text{reg}(v_n)) = \rho(s_n, v_n)$ is accessible in $(\text{Untime}(A), \pi)$ and this proves the first part of the theorem.

Now assume that the timed language $\text{Ltimed}(A, \mu)$ is not maximal with this property: that is, that there is a control function μ^* such that $\text{Ltimed}(A, \mu^*) \supsetneq \text{Ltimed}(A, \mu)$, there is a timed word $(\sigma_1, t_1) \dots (\sigma_n, t_n)$ lying in $\text{Ltimed}(A, \mu^*) - \text{Ltimed}(A, \mu)$ and the extended states in F are also inaccessible in (A, μ^*) . Let

$$(s_0, \mathbf{0}) \xrightarrow{(\sigma_1, t_1)} (s_1, v_1) \dots \xrightarrow{(\sigma_n, t_n)} (s_n, v_n)$$

be the path in A defined by the timed word $(\sigma_1, t_1) \dots (\sigma_n, t_n)$ starting at $(s_0, \mathbf{0})$. For all $i \leq n$, there does not exist a path over any word in Σ_u^* starting at (s_i, v_i) and ending at an element of F . Furthermore, for at least one value of $i < n$, we must have

$$\pi(s_i, \text{reg}(v_i + t_{i+1}), \sigma_{i+1}) = \mu(s_i, v_i + t_{i+1}, \sigma_{i+1}) = 0$$

(since $(\sigma_1, t_1) \dots (\sigma_n, t_n) \notin \text{Ltimed}(A, \mu)$) and so the projection of the timed word $(\sigma_1, t_1) \dots (\sigma_n, t_n)$ onto

Untime(A) starting at $(s_0, \mathbf{0})$ does not lie in $L(\text{Untime}(A), \pi)$. Let $\pi^*: S \times R \times \Sigma \cup \{\tau\} \rightarrow \{0, 1\}$ be the control function for $\text{Untime}(A)$ defined as follows. If a state (s, r) is accessible in $\text{Untime}(A)$, π then $\pi^*(s, r', \sigma) = \pi(s, r', \sigma)$ for any time successor r' of r , unless $(s, r', \sigma) = (s_i, \text{reg}, (v_i + t_{i+1}), \sigma_{i+1})$ for some $i < n$, in which case $\pi^*(s_i, \text{reg}(v_i + t_{i+1}), \sigma_{i+1}) = 1$. If $\sigma \in \Sigma_u \cup \{\tau\}$ then necessarily $\pi^*(s, r, \sigma) = 1$. In all other cases $\pi^*(s, r, \sigma) = 0$.

It follows by induction on word length that $L(\text{Untime}(A), \pi) \subseteq L(\text{Untime}(A), \pi^*)$; furthermore, the projection of the timed word $(\sigma_1, t_1) \cdots (\sigma_n, t_n)$ lies in $L(\text{Untime}(A), \pi^*)$, but not in $L(\text{Untime}(A), \pi)$, so the inclusion is strict.

Owing to the maximality condition on $L(\text{Untime}(A), \pi)$, an element of $\rho(F)$ is accessible in $(\text{Untime}(A), \pi^*)$. Therefore there is a path p in $\text{Untime}(A)$ from $(s_0, \mathbf{0})$ to an element of $\rho(F)$ which is permitted by π^* , but not, of course, by π . Thus p must pass through a transition

$$(s_i, \text{reg}(v_i + t_{i+1})) \xrightarrow{\sigma_{i+1}} (s_{i+1}, \text{reg}(v_{i+1}))$$

for some $i < n$, since all transitions apart from these ones are permitted by π . Let p' be a final segment of p , starting at the state $(s_{i+1}, \text{reg}(v_{i+1}))$ for some $i < n$, and not passing through any transitions of the kind displayed above. Thus (by the definition of π^*) $\pi(s, r, \sigma) = \pi^*(s, r, \sigma) = 1$ for all edges $(s, r) \xrightarrow{\sigma} (s', r')$ through which p' passes. We will show that $\sigma \in \Sigma_u \cup \{\tau\}$ for all such edges. Assume that this is not the case and let $(s, r) \xrightarrow{\sigma} (s', r')$ be the last edge in p' for which $\sigma \in \Sigma_c$. The definition of π^* implies that (s, r) is accessible in $(\text{Untime}(A), \pi)$, which leads to a contradiction since the endpoint of p , which lies in $\rho(F)$, would be accessible in $(\text{Untime}(A), \pi)$. We can lift p' to a path q , using Lemma 5(2) repeatedly, starting at (s_{i+1}, v_{i+1}) and ending at an extended state $f \in \rho^{-1}\rho(F) = F$. The path q is legal in (A, μ^*) , since all its events lie in Σ_u . However, the extended state (s_{i+1}, v_{i+1}) is accessible in (A, μ^*) ; thus the forbidden state f is accessible in (A, μ^*) , contrary to hypotheses. $\square$

Remark: Following Theorem 5, when the time increases strictly monotonically, the automaton $\text{Untime}(A)$ can be replaced by the simplified automaton $\text{Untime}(A^*)$ that is similar to G^* where the event τ has been added to its alphabet.

Example 3: Consider a simple timed automaton given in figure 2a involving only two states, one controllable transition and initial state s . Assume that the forbidden state is the state $(s', x = 1)$. In order to solve the control problem, the $\text{Untime}(A)$ given in figure 2b (τ -transitions are shown as well) is derived and clearly all

Figure 2a.

Figure 2b.

Figure 2c.

transitions leading to this state should be disabled. If time is strictly positive the simplified automaton $\text{Untime}(A^*)$, given in figure 2b, is derived. Then clearly both extended states $(s', x = 1)$ and $(s', 1 < x < 1)$ should be disabled. Intuitively this is the price someone has to pay when the passing of time

is strictly positive and it is natural to expect that the controlled behaviour should become more conservative. In other words, when the time is not allowed to be zero, the automaton would not be permitted to visit *non-closed* states associated with forbidden *closed* states. This is due to the fact that, when the simplified automaton is used, if $\text{next}(x) = \text{next}(y)$ then, when (s, x) is forbidden, (s, y) also becomes forbidden since there can be no difference between extended states with the same next-value.

From the above it follows that when the time jump is strictly positive the controlled behaviour is more conservative. The following corollary gives a condition under which the controlled behaviour is the least restrictive one derived using theorem 5.

Corollary 1: The forbidden state control problem using the simplified untiming procedure gives the least restrictive controlled behaviour as stated by Theorem 5 under the condition that the forbidden states are not *closed* regions. In other words the atomic predicates specifying the forbidden extended states should be of the form x^*c for a clock $x \in C$ where $*$ is one of $\leq$, $<$, $\geq$ or $>$ (but not $=$).

8. Conclusions

Given a timed automaton G , we have defined a new finite state automaton G^* that accepts the same untimed language as G and has significantly fewer states than the region automaton G' . Given a timed Buchi automaton, we have proved a similar result.

Next, this result is used in connection with the forbidden state control problem for timed discrete event systems. We study this problem and show that the simplified untiming construction leads to a simplified supervisory control design for real time discrete-event systems modelled by timed automata. Conditions under which the least restrictive solution is obtained are given.

Due to its complexity, the untiming procedure for dense time automata results in a large number of regions for the untimed automaton. The efficiency of the involved supervisory design depends on the number of regions of the untimed automaton. Therefore, simplifications and computational approximations for timed automata are of importance. For the simplified untiming procedure addressed here, the number of regions in the graph still depend exponentially on the number of clocks. In practice, as explained in §6, the reduction of regions is significant. In Laurence and Spathopoulos (1998), the simplified procedure reduces the exponential number of regions. However, it works only for a special case that essentially corresponds to the case where the time is discretized. Further work with reference to the notion of urgency introduced in Spathopoulos (2003) will be reported elsewhere.

References

- ALUR, R., and DILL, D., 1994, A theory of timed automata. *Theoretical Computer Science*, **126**, 183–235.
- ASARIN, E., MALER, O., PNEULI, A., and SIFAKIS, J., 1998, Controller synthesis for timed automata. *Proceedings of the IFAC Symposium on System Structure and Control* (Elsevier), pp. 469–474.
- BRANDIN, B. A., WONHAM, W. M., 1994, Supervisory control of timed discrete-event systems. *IEEE Transactions on Automatic Control*, **39**, 329–342.
- HOFFMANN, G., and WONG-TOI, H., 1992, *The Control of Dense Real-time Discrete Event Systems*, Stanford University CS Report, No. STAN-CS-92-1411.
- LAURENCE, M. R., and SPATHOPOULOS, M. P., 1998, Forbidden state problems in timed automata. *Proceedings of the 4th International Workshop on Discrete-Event Systems (WODES 98)*, Cagliari, Italy, August, pp. 15–23.
- RAMADGE, P. J., and WONHAM, W. M., 1987, Supervisory control of a class of discrete event processes, *SIAM Journal of Control and Optimization*, **25**, 206–230.
- SPATHOPOULOS, M. P., 2003, On supervisory control for timed automata using urgency. *Proceedings of the 9th IEEE International Conference on Methods and Methods in Automation and Robotics*, Miedzyzdroje, Poland, August, pp. 863–869.