Discrete Event Dynamic Systems: Theory and Applications, 10, 87–113 (2000)
© 2000 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Dynamical Properties of Timed Automata

ANUJ PURI anuj@eclair.eecs.berkeley.edu
University of California, 195 M Cory Hall, Berkeley, CA 94720

Abstract. Timed automata are an important model for specifying and analyzing real-time systems. The main analysis performed on timed automata is the reachability analysis. In this paper we show that the standard approach for performing reachability analysis is not correct when the clocks drift even by a very small amount. Our formulation of the reachability problem for timed automata is as follows: we define the set $R^*(T, Z_0) = \cap_{\epsilon>0} Reach(T_\epsilon, Z_0)$ where T_ϵ is obtained from timed automaton T by allowing an ϵ drift in the clocks. $R^*(T, Z_0)$ is the set of states which can be reached in the timed automaton T from the initial states in Z_0 when the clocks drift by an infinitesimally small amount. We present an algorithm for computing $R^*(T, Z_0)$ and provide a proof of its correctness. We show that $R^*(T, Z_0)$ is robust with respect to various types of modeling errors. To prove the correctness of our algorithm, we need to understand the dynamics of timed automata—in particular, the structure of the limit cycles of timed automata.

Keywords: Timed automata, dynamical systems, verification

1. Introduction

Real-time systems play an increasingly important role in applications ranging from telecommunications to computer controlled physical systems. An important model for analyzing real-time systems is the timed automata model (Alur and Dill, 1990). A timed automaton is an automaton coupled with a finite number of continuous clocks. The clocks move uniformly at rate one and a jump is made from one control location of the automaton to another based on the value of the clocks. Timed automata have been used for specifying and analyzing numerous real-time systems from a variety of application domains and a number of computer tools are now available for their analysis (Alur and Dill, 1990; Alur et al., 1995; Alur and Kurshan, 1996; Bengtsson et al., 1996; Courcoubetis and Yannakakis, 1992; Daws et al., 1996; Dill, 1989; Henzinger et al., 1992; Yannakakis and Lee, 1993).

The main analysis performed on timed automata is the reachability analysis. The reachability analysis checks whether some undesirable states of the system are reachable from the initial state. In Alur and Dill (1990), it was shown that a finite *region graph* can be constructed from the timed automata and the reachability problem can be solved using this graph. But timed automata are continuous systems that maybe viewed as dynamical systems. The approach of Alur and Dill (1990) ignored all the dynamical features of timed automata and solved the reachability problem by creating the finite region graph.

In this paper we show that the approach of Alur and Dill (1990) is not robust. Reachability analysis done using the approach of Alur and Dill (1990) will not be correct under the assumption of a small drift in clocks. This is disturbing because we would like small errors in our model to create only small errors in the results of our analysis. On the other hand, the behavior of a system when the clocks drift even a little bit maybe quite different from what happens when the system has perfect clocks.

A small drift in clocks can cause a drastic change in the behavior of timed automata because states near the reachable states of our original model can be reached also. How these states behave over long term is not accounted for in Alur and Dill (1990). To understand the long term behavior of these states, we will need to understand the dynamics of timed automata.

Let us describe the problem we want to solve in this paper. Define $\mathit{Reach}(T, Z_0)$ to be the reachable states in the timed automaton T starting from the initial states in Z_0. Similarly define $\mathit{Reach}(T_\epsilon, Z_0)$ to be the reachable states in T_ϵ where T_ϵ is obtained by allowing an ϵ drift in the clocks in T. Now $\mathit{Reach}(T, Z_0) \subset \mathit{Reach}(T_\epsilon, Z_0)$. We will be interested in computing the limit $R^*(T, Z_0) = \cap_{\epsilon>0} \mathit{Reach}(T_\epsilon, Z_0)$. The set $R^*(T, Z_0)$ is the limit of $\mathit{Reach}(T_\epsilon, Z_0)$ as $\epsilon \longrightarrow 0$. So $R^*(T, Z_0)$ is the set of states which can be reached in the timed automaton T from the initial states in Z_0 when the clocks drift by an infinitesimally small amount. In general, $R^*(T, Z_0) \neq \mathit{Reach}(T, Z_0)$. In this paper we present an algorithm for computing $R^*(T, Z_0)$. To prove the correctness of our algorithm, we will need to understand the dynamics of timed automata—in particular, the structure of the limit cycles of timed automata. We will show that unlike $\mathit{Reach}(T, Z_0)$, $R^*(T, Z_0)$ is robust against various types of modeling errors and drifts in clocks.

Our main contributions in this paper are as follows:

1. We present an algorithm for computing $R^*(T, Z_0)$.

2. We present an analysis of the dynamics of timed automata. In particular, we present a complete analysis of the limit cycles of timed automata.

3. We provide a method for establishing the relationship between the trajectories of the timed automaton T and the perturbed model T_ϵ.

In Gupta et al. (1997), a notion of robustness for timed automata is also described. But unlike our approach, their approach involves looking at fewer executions of the timed automaton. In particular, their approach does not apply to reachability analysis in the presence of small drifts in clocks.

In Section 2, we introduce our notation. In Section 3, we describe the timed automata model and review the work of Alur and Dill (1990). In Section 4, we show that the reachability analysis based on the approach of Alur and Dill (1990) is not robust. We then present our formulation of the reachability problem. In Section 5, we present the algorithm for computing $R^*(T, Z_0)$. Proving correctness of the algorithm relies on understanding the structure of the limit cycles of timed automata and on understanding the relationship between the trajectories of T and T_ϵ. In Section 6, we discuss the relationship between zones and regions. In Section 7, we provide a complete analysis of the limit cycles of timed automata. In Section 8, we discuss the relationship between the trajectories of T and T_ϵ. In Section 9, we show that $R^*(T, Z_0)$ is a robust set. Section 10 discusses the complexity of computing $R^*(T, Z_0)$ and Section 11 is the conclusion.

2. Notation

2.1. Preliminaries

The set $\mathbb{R}$ ($\mathbb{R}^+$) are the (positive) reals and $\mathbb{Z}$ ($\mathbb{Z}^+$) are the (positive) integers. For $x \in \mathbb{R}^n$, we write x_i for the ith component of x. For $x, y \in \mathbb{R}^n$, define $\|x - y\| = \max_i |x_i - y_i|$. For a set $S \subset \mathbb{R}^n$ and $x \in \mathbb{R}^n$, define $dist(x, S) = \inf_{s \in S} \|x - s\|$. For $S \subset \mathbb{R}^n$, $cl(S)$ is the closure of S and $conv(S)$ is the convex hull of S. For sets X and Y, $X + Y = \{x + y \mid x \in X, y \in Y\}$. A set-valued map is $f\colon A \longrightarrow B$ where $f(a)$ is a set. We define $f^{i+1}(x) = \{z \mid z \inf(y) \text{ and } y \in f^i(x)\}$.

2.2. Graphs

A graph is $G = (V, \longrightarrow_G)$ where V is the set of vertices and $\longrightarrow_G \subset V \times V$ is the set of edges (i.e., there is an edge from v to w provided $v \longrightarrow_G w$). We write $v \Longrightarrow_G w$ provided there is a path from v to w in G.

A non-trivial strongly connected component (SCC) of G is a maximal set $S \subset V$ such that for any $v, w \in S$, $v \Longrightarrow_G w$ and every vertex $v \in S$ is contained in a cycle. In this paper we will only be interested in non-trivial strongly connected components. There are well known algorithms for computing the non-trivial strongly connected components of a graph (Cormen et al., 1990).

2.3. Zones

A $\mathbb{Z}$-zone ($\mathbb{R}$-zone) $Z \subset \mathbb{R}^n$ is a closed set defined by inequalities of the form

$$x_i - x_j \leq u_{ij}, \qquad l_i \leq x_i \leq u_i \tag{1}$$

where $i, j \in \{1, \ldots, n\}$ and $u_{ij}, l_i, u_i \in \mathbb{Z}$ ($u_{ij}, l_i, u_i \in \mathbb{R}$). We will refer to $\mathbb{Z}$-zones simply as zones.

Example 2.1: A zone $Z \subset \mathbb{R}^2$ defined by the inequalities

$$1 \leq a \leq 3, \qquad 0 \leq b \leq 3, \qquad 0 \leq a - b \leq 2 \tag{2}$$

is shown in Figure 1.

3. Timed Automata

A timed automaton (Alur and Dill, 1990) is an automaton with a finite number of real-valued clocks, a finite number of control locations and edges between the control locations. In the control locations, the clocks move uniformly at rate one. A jump from one location to another can be made when the clocks satisfy the guards on the edges. During the jump, some of the clocks may get reset to zero.

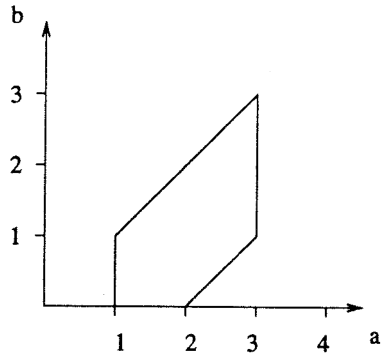

Figure 1. Zone defined by inequalities of Equation 2.

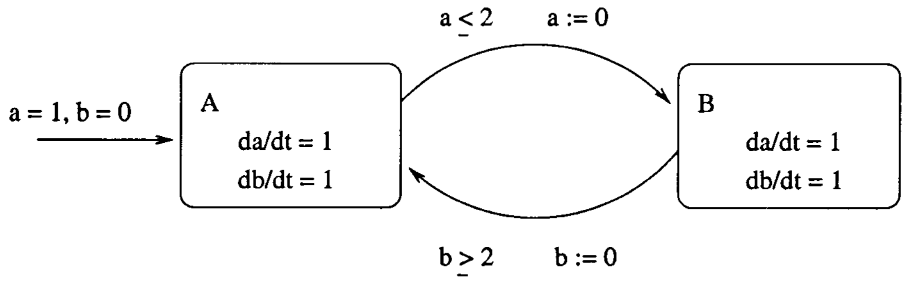

Figure 2. A timed automaton.

More formally, a timed automaton is $T = (L, E)$ where L is the set of control locations and E is the set of edges. The state of a timed automaton with n clocks is (l, x) where l is a control location and $x \in \mathbb{R}^n$. An edge from location l to location m is $e = (l, g, r, m)$ where g is the guard and $r \subset \{1, \ldots, n\}$ are the clocks which are reset on the jump from l to m. The guard g is a closed zone in $\mathbb{R}^n$. The state space of the timed automaton is $Q_T \subset L \times [0, M]^n$ where $Q_T = \cup_{l \in L}(l, Z_l)$ and Z_l is a closed zone. Notice the state space of our timed automaton is a closed bounded set and the guards are closed sets.

Figure 2 is an example of a timed automaton with clocks a and b, control locations A and B, and state space $Q_T = (A, 0 \leq b \leq a \leq 2) \cup (B, 0 \leq a \leq b \leq 2)$.

We now review the work of Alur and Dill (1990) who presented a method for solving the reachability problem in timed automata.

3.1. Transition System

The transition system of the timed automaton describes how the state of the timed automaton moves. The transition system of the timed automaton is $T = (Q_T, \longrightarrow, \Sigma)$ where Q_T is the state space of the timed automaton, $\longrightarrow \subset nQ_T \times \Sigma \times Q_T$ is the transition relation and $\Sigma = \mathbb{R}^+ \cup \{d\}$ is the set of moves.

The state of the timed automaton moves by either staying in the same control location and letting time elapse, or by making a jump from one control location to another.

Definition 3.1. The transition system of timed automaton $T = (L, E)$ is $T = (Q_T, \longrightarrow, \mathbb{R}^+ \cup \{d\})$ where

1. $(l, x) \xrightarrow{t} (l, y)$ provided $y = x + t$.

2. $(l, x) \xrightarrow{d} (m, y)$ provided $(l, g, r, m) \in E$, $x \in g$, $y_i = 0$ when $i \in r$, and $y_i = x_i$ for $i \notin r$.

We define the relation $\rightarrow \subset Q_T \times Q_T$ where $q \rightarrow q'$ provided $q \xrightarrow{t} q'$ or $q \xrightarrow{d} q'$. We define the relation $\Rightarrow$ to be the reflexive and transitive closure of $\rightarrow$. So $q_0 \Rightarrow q_N$ provided there are $q_1, \ldots, q_{N-1}$ such that $q_i \rightarrow q_{i+1}$ for $i = 0, \ldots, N-1$.

Definition 3.2. For a timed automaton $T = (L, E)$ and an initial closed zone Z_0, define $\mathit{Reach}(T, Z_0) = \{q \mid q_0 \Rightarrow q,\ q_0 \in Z_0\}$.

$\mathit{Reach}(T, Z_0)$ is the set of states which can be reached in the timed automaton T starting from a state in the initial zone Z_0. The *reachability problem* is to determine whether a state $q \in \mathit{Reach}(T, Z_0)$. The *control location reachability problem* is to determine whether a control location l_f is reachable (i.e., whether $(l_f, x) \in \mathit{Reach}(T, Z_0)$ for some x).

3.2. Trajectories of Timed Automata

A trajectory of the timed automaton is $\pi = (q_0, t_0)(q_1, t_1) \ldots (q_k, t_k)$ where either $q_i \xrightarrow{d} q_{i+1}$ and $t_{i+1} = t_i$, or $\tau_i = t_{i+1} - t_i$ and $q_i \xrightarrow{\tau_i} q_{i+1}$. We will sometimes also refer to this trajectory as $\pi[t_0, t_k]$ and write $\pi(t_i)$ instead of q_i. The trajectory is *stutter-free* provided it is not the case that $q_i \xrightarrow{\tau_i} q_{i+1}$ and $q_{i+1} \xrightarrow{\tau_{i+1}} q_{i+2}$ for some i.

Example 3.1: A trajectory of the timed automaton of Figure 2 is $(A, a = 1, b = 0)(A, a = 1.5, b = 0.5)(B, a = 0, b = 0.5)(B, a = 1.7, b = 2.2)(A, a = 1.7, b = 0)$.

3.3. Equivalence Classes

For $c \in \mathbb{R}$, define $\lfloor c \rfloor$ to be its integer part and $\langle c \rangle$ to be its fractional part.

Definition 3.3. Define a relation $\sim \subset [0, M]^n \times [0, M]^n$ where $x \sim y$ provided for every i and j

- $\langle x_i \rangle = 0$ iff $\langle y_i \rangle = 0$
- $\langle x_i \rangle < \langle x_j \rangle$ iff $\langle y_i \rangle < \langle y_j \rangle$
- $\langle x_i \rangle = \langle x_j \rangle$ iff $\langle y_i \rangle = \langle y_j \rangle$
- $\lfloor x_i \rfloor = \lfloor y_i \rfloor$ for every i

It can be checked that $\sim$ is an equivalence relation. We next describe the equivalence classes of the relation $\sim$. For a vector $x \in \mathbb{R}^n$, we define $\langle x \rangle$ to be the ordering of the fractional parts of its components. For example, for $x = (0.3\ 4.7\ 3.2\ 8.3)$, $\langle x \rangle = (0 < \langle x_3 \rangle < \langle x_1 \rangle = \langle x_4 \rangle < \langle x_2 \rangle)$. Similarly, $\lfloor x \rfloor$ is a vector of integer parts of the components. For $x = (0.3\ 4.7\ 3.2\ 8.3)$, $\lfloor x \rfloor = (0\ 4\ 3\ 8)$.

LEMMA 3.1 *For the relation* $\sim$, *the equivalence class* $]x[= \{y \mid \langle x \rangle = \langle y \rangle$ *and* $\lfloor x \rfloor = \lfloor y \rfloor\}$.

Because in this paper, we have assumed that the guards, the initial set and the state space of the timed automaton are closed, we only need to deal with the closure of the equivalence classes. Let us *define* $[x] = cl(]x[)$. For example, for $x = (0.3\ 4.7\ 3.2\ 8.3)$, $[x] = \{y \mid \lfloor y \rfloor = (0\ 4\ 3\ 8)$ and $(0 \le \langle y_3 \rangle \le \langle y_1 \rangle = \langle y_4 \rangle \le \langle y_2 \rangle)\}$.

The relation $\sim$ and $[\ ]$ extends in a natural manner to Q_T. For $q \in Q_T$, we call $[q]$ a region. Let us define $\mathcal{C} = \{[q] \mid q \in Q_T\}$ to be the set of regions. Because Q_T is bounded, $\mathcal{C}$ is a finite set.

For a zone $Z \subset Q_T$, we define $[Z] = \{[q] \mid q \in Z\}$ to be regions making up Z. For a set $K \subset \mathcal{C}$, we define $States(K) \subset Q_T$ to be the set of states making up the regions in K. So for a closed zone $Z \subset Q_T$, $States([Z]) = Z$. When it is clear from the context that we are talking about $States(K)$, we may simply write it as K.

The following theorem states the basic property of regions.

THEOREM 3.1 *Suppose* $q \to q'$ *for* $q, q' \in Q_T$. *Then for any* $s \in [q]$, *there exists* $s' \in [q']$ *such that* $s \to s'$; *and for any* $w' \in [q']$, *there exists* $w \in [q]$ *such that* $w \to w'$.

3.4. The Region Graph

The equivalence relation $\sim$ partitions the state space Q_T into a finite set. Taking closure of the elements of this set, we obtain $\mathcal{C}$. We call the members of $\mathcal{C}$ regions. Associated with a timed automaton is a finite *region graph* whose vertices are $\mathcal{C}$ and there is an edge from vertex $[q]$ to vertex $[q']$ provided some point in $]q[$ can move to a point in $]q'[$.

Definition 3.4. For the timed transition system $T = (Q_T, \longrightarrow, \Sigma)$, we define the region graph $G = (\mathcal{C}, \longrightarrow_G)$ where the vertices are $\mathcal{C}$ and the edges are $\longrightarrow_G \subset \mathcal{C} \times \mathcal{C}$. We define $[q] \longrightarrow_G [q']$ provided $q \to q'$ and $[q] \neq [q']$.

In the region graph G, we say $[q']$ is an *immediate successor* of $[q]$ provided either $q \xrightarrow{d} q'$ or $q \xrightarrow{t} q'$ and for $\tau \leq t$, if $q \xrightarrow{\tau} q''$ then $[q''] = [q]$ or $[q''] = [q']$.

The region graph of the timed automaton of Figure 2 is shown in Figure 3 (the figure only shows edges between immediate successors).

Let us define $\Longrightarrow_G$ to be the reflexive and transitive closure of $\longrightarrow_G$. So $u \Longrightarrow_G w$ provided there is a path in the graph G from vertex u to the vertex w.

Definition 3.5. For the graph $G = (\mathcal{C}, \longrightarrow_G)$, define $Reach(G, S) = \{v \mid v_0 \Longrightarrow_G v, v_0 \in S\}$.

$Reach(G, S)$ is the set of vertices which can be reached in the graph G starting from a vertex in S. The region graph G is in some sense a finite description of the underlying timed automaton. In particular, as the next theorem shows the reach set of the timed automaton can be computed using the region graph G.

THEOREM 3.2 (Alur and Dill, 1990) *For a timed automaton $T = (L, E)$, let $G = (\mathcal{C}, \longrightarrow_G)$ be the underlying region graph. Then $Reach(T, Z_0) = Reach(G, [Z_0])$.*

Figure 4 shows the states which are reached in the timed automaton of Figure 2 starting from the initial state $(A, (a = 1) \wedge (b = 0))$. This set can be computed by finding the reachable vertices in the region graph G of Figure 3 from the initial vertex $(A, (a = 1) \wedge (b = 0))$.

Definition 3.6. A progress cycle in the region graph is a cycle in which each clock is reset at least once.

Assumption 3.1. We make the assumption that for the timed automata we consider, every cycle in the region graph is a progress cycle.

In a progress cycle, time can pass if we follow the progress cycle. The assumption simplifies our presentation.

4. A Problem with Timed Automata

In this section, we discuss a problem with computing the reach set of the timed automata using the approach of the previous section. We show that even an infinitesimally small drift in the clocks can invalidate the results obtained using the analysis of the previous section. We then present our formulation of the reachability problem for timed automata.

4.1. Problem

Consider again the timed automaton of Figure 2. Let us now assume that clocks a and b are not perfect, but they may drift a little bit. We model the clocks as $\dot{a} \in [1-\epsilon, 1+\epsilon]$ and

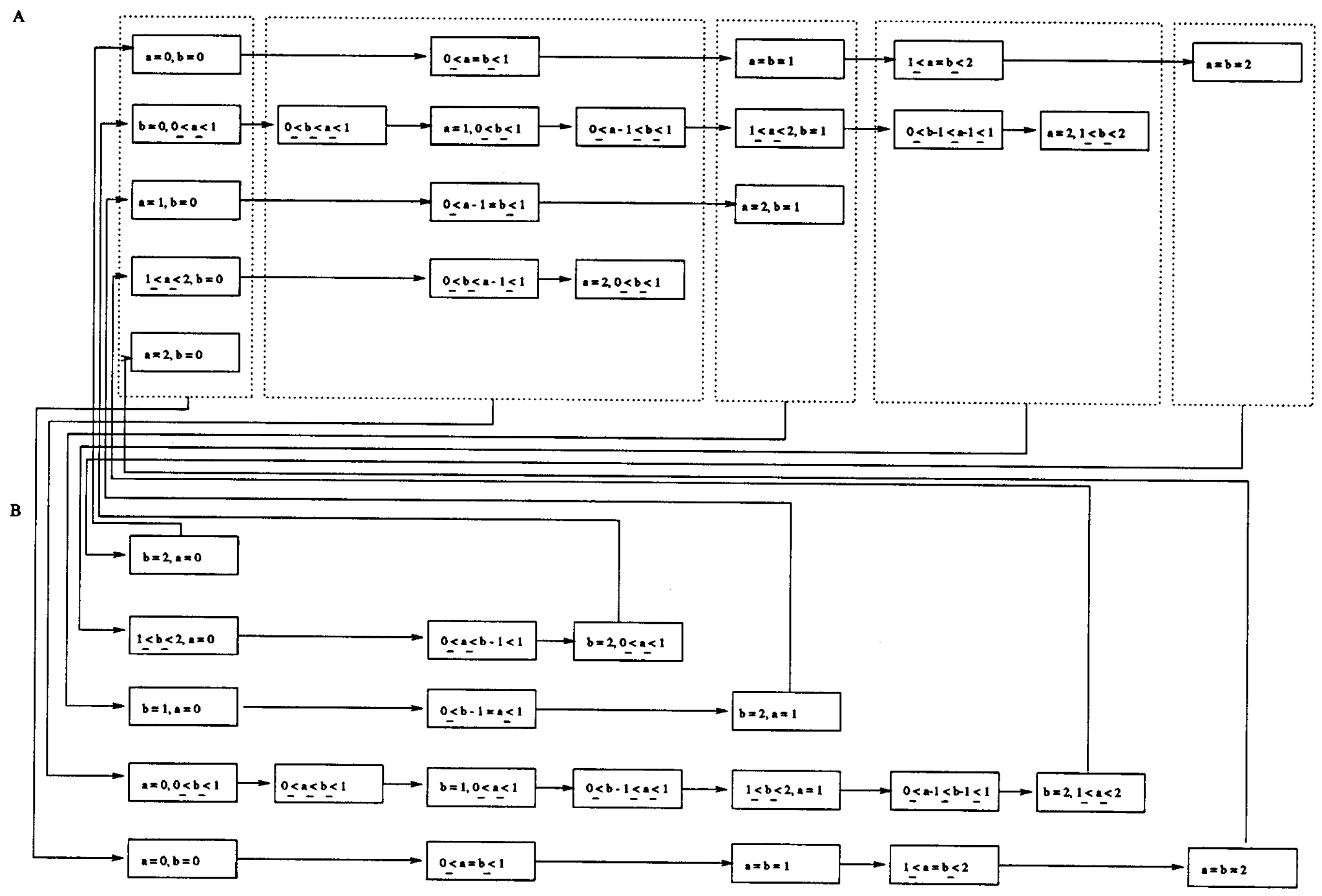

Figure 3. The region graph of timed automaton of Figure 2.

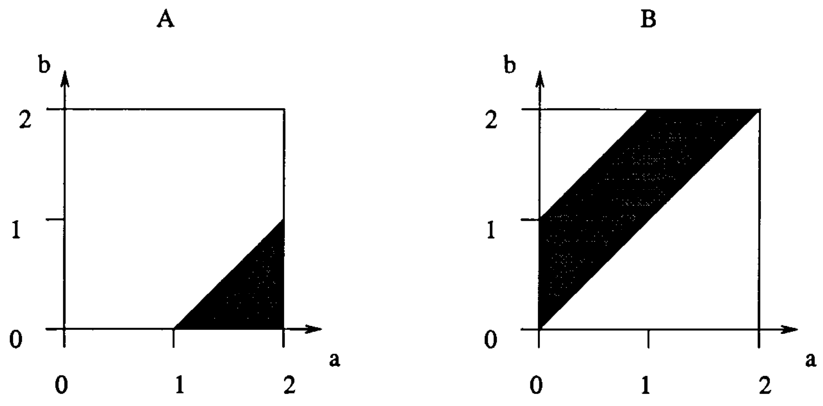

Figure 4. The set *Reach*(T, Z_0) for timed automaton of Figure 2.

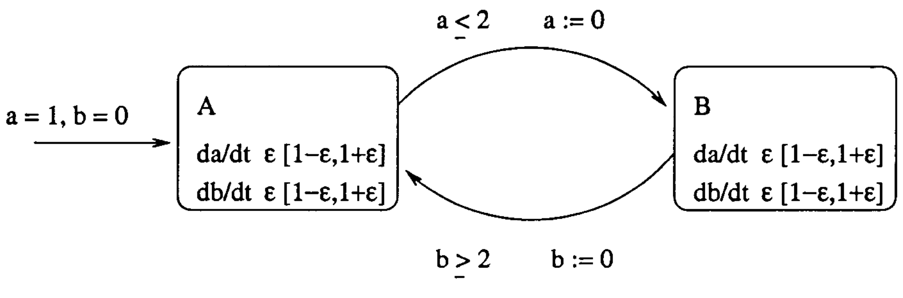

Figure 5. The timed automaton T_ϵ.

$\dot{b} \in [1 - \epsilon, 1 + \epsilon]$ where ϵ bounds the amount of drifting. Let us call this new automaton shown in Figure 5, T_ϵ.

We will now compute the reachable states in T_ϵ. Clearly all the states in *Reach*(T, Z_0) can also be reached in T_ϵ. In addition, the following phenomenon occurs: in the first jump from A to B, the state $(B, (b = 1 + \epsilon) \wedge (a = 0))$ is reached; and on the subsequent jump from B to A, the state $(A, (a = 1 - \epsilon) \wedge (b = 0))$ is reached. In the next jump from A to B, the state $(B, (b = 1 + 2\epsilon) \wedge (a = 0))$ is reached; and on the jump from B to A, the state $(A, (a = 1 - 2\epsilon) \wedge (b = 0))$ is reached. Continuing is this manner, we find that any state $(B, (b = 1 + k\epsilon) \wedge (a = 0))$ and $(A, (a = 1 - k\epsilon) \wedge (b = 0))$ can be reached. In fact, it can be checked that the whole set $R^*(T, Z_0)$ shown in Figure 6 can be reached under the assumption of drifting clocks for any $\epsilon > 0$. On the other hand, if $\epsilon = 0$ then the set of reachable states is exactly the one shown in Figure 4.

Assuming perfect clocks in a timed system with continuous clocks is clearly not a reasonable assumption. Therefore, an analysis based on the approach of the previous section can lead to an incorrect conclusion about the behaviour of the system (as for example in the timed automaton of Figure 2).

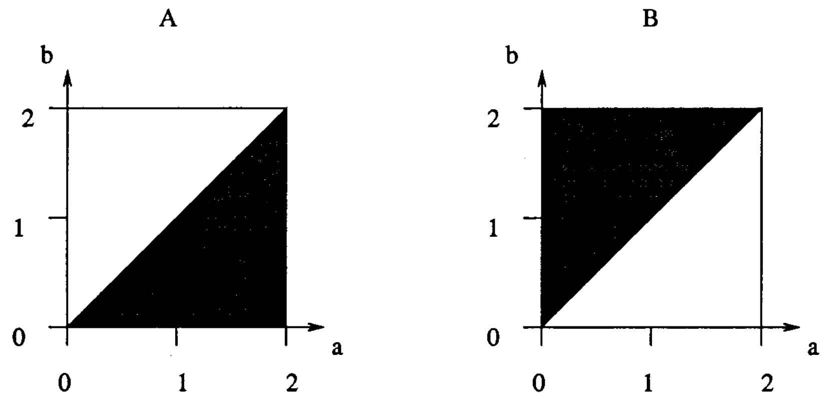

Figure 6. The set $R^*(T, Z_0)$ for timed automaton of Figure 2.

4.2. Problem Formulation

Let us now appropriately formulate the problem we want to solve in this paper. For a timed automaton T, let us define the timed automaton T_ϵ. The timed automaton T_ϵ is obtained from T by replacing the clock $\dot{a} = 1$ in T with the drifting clock $\dot{a} \in [1 - \epsilon, 1 + \epsilon]$ in T_ϵ. Figure 5 shows the timed automaton T_ϵ obtained from the timed automaton T in Figure 2.

Definition 4.1. The transition system of timed automaton T_ϵ is $T_\epsilon = (Q_T, \longrightarrow, \mathbb{R}^+ \cup \{d\})$ where

1. $(l, x) \xrightarrow{t} (l, y)$ provided $(1 - \epsilon)t \leq y_i - x_i \leq (1 + \epsilon)t$ for each i.

2. $(l, x) \xrightarrow{d} (m, y)$ provided $(l, g, r, m) \in E$, $x \in g$, $y_i = 0$ when $i \in r$, and $y_i = x_i$ for $i \notin r$.

For the timed automaton T_ϵ, define the relation $\rightarrow \subseteq Q_T \times Q_T$ where $q \rightarrow q'$ provided $q \xrightarrow{t} q'$ or $q \xrightarrow{d} q'$ in T_ϵ. Define the relation $\Longrightarrow$ to be the reflexive and transitive closure of $\rightarrow$.

Definition 4.2. For a timed automaton T_ϵ and an initial zone Z_0, define *Reach*$(T_\epsilon, Z_0) = \{q \mid q_0 \Rightarrow q \text{ in } T_\epsilon \text{ and } q_0 \in Z_0\}$.

Reach(T_ϵ, Z_0) is the set of states which can be reached in T_ϵ. Clearly for $\epsilon_2 \leq \epsilon_1$, *Reach*$(T_{\epsilon_2}, Z_0) \subset$ *Reach*(T_{ϵ_1}, Z_0). We will be interested in computing the set $R^*(T, Z_0) = \bigcap_{\epsilon > 0}$ *Reach*(T_ϵ, Z_0). The set $R^*(T, Z_0)$ is the limit of *Reach*(T_ϵ, Z_0) as $\epsilon \rightarrow 0$. The set $R^*(T, Z_0)$ for the timed automaton of Figure 2 is shown in Figure 6. As this example shows, $R^*(T, Z_0)$ is not necessarily equal to *Reach*(T, Z_0).

Our main result in this paper is an algorithm for computing $R^*(T, Z_0)$ for a timed automaton T and an initial zone Z_0. In Section 5, we will present the algorithm for computing $R^*(T, Z_0)$. In Section 9, we will show that unlike $Reach(T, Z_0)$, $R^*(T, Z_0)$ is robust against errors in guards, drifts in clocks and other types of modeling errors.

5. The New Algorithm

We now present the algorithm for computing $R^*(T, Z_0)$ for a timed automaton T and a closed initial zone Z_0. The algorithm works on the underlying region graph $G = (\mathcal{C}, \longrightarrow_G)$ of the timed automaton T.

Algorithm for computing $R^*(T, Z_0)$:

```
1. Construct the region graph $G = (\mathcal{C}, \longrightarrow_G)$ of timed automaton $T$
2. Compute $SCC(G)$ = strongly connected components (SCC) of graph $G$
3. $J^* := [Z_0]$
4. $J^* := Reach(G, J^*)$
5. If for some $S \in SCC(G)$, $S \not\subseteq J^*$ and $States(J^*) \cap States(S) \neq \emptyset$
      then $J^* := J^* \cup S$
         Goto 4
      else STOP
```

The set J^* is computed using two operations: if a SCC S is "touching" J^*, then the new $J^* := J^* \cup S$; and from a given set J^*, the new $J^* := Reach(G, J^*)$. Starting from the set $[Z_0]$, these two operations are repeated until no new states are added to J^*.

Example 5.1: Consider the timed automaton of Figure 2 and its region graph in Figure 3. Let us compute the set $R^*(T, Z_0)$ for $Z_0 = (A, (a = 1) \wedge (b = 0))$ using our algorithm. The set $SCC(G) = \{\{A1, B1\}, \{A2, B2\}, \{A3, B3\}, \{A4, B4\}, \{A5, B5\}\}$. Now in Step 3 $J^* = Reach(G, [Z_0]) = \{A3, B3, A4, B4, A5, B5\}$. Since the SCC $\{A2, B2\}$ is "touching" J^*, the new $J^* = \{A2, B2, A3, B3, A4, B4, A5, B5\}$. And since the SCC $\{A1, B1\}$ also "touches" J^*, $J^* = \{A1, B1, A2, B2, A3, B3, A4, B4, A5, B5\}$. The set J^* computed by our algorithm is exactly the set $R^*(T, Z_0)$ shown in Figure 6.

For a timed automaton T, the set J^* computed by our algorithm is exactly $R^*(T, Z_0)$. The correctness of the algorithm relies on the following two theorems. We assume that the timed automaton T has state space $Q_T \subset L \times [0, M]^n$ and W is the number of regions in Q_T.

THEOREM 7.3 *Suppose S is a SCC of the region graph G, and x, y are two points in $States(S)$. Then $x \Rightarrow y$ in T_ϵ for any $\epsilon > 0$.*

Proof: See Section 7 ∎

THEOREM 8.3 *Suppose $x \in States(J^*)$ and $x \Rightarrow y$ in timed automaton T_ϵ with $\epsilon < \frac{\alpha}{4(n+W+1)(M+1)}$ where $\alpha < \frac{1}{4}$. Then $dist(y, J^*) < \alpha$.*

Proof: See Section 8 ■

Theorem 7.3 says that in a SCC S of the region graph G of timed automaton T, it is possible to move from any point $x \in States(S)$ to any other point $y \in States(S)$ in the timed automaton T_ϵ. To prove this result, we will need to understand the structure of the limit cycles of the timed automaton. We do this in Section 7.

Theorem 8.3 states that from a point $x \in States(J^*)$, it is not possible to get to any other point which is more than distance α away from J^* in the timed automaton T_ϵ for sufficiently small ϵ. To prove this, we need to better understand the relationship between trajectories of T_ϵ and T. We do this in Section 8.

The correctness of our algorithm for computing $R^*(T, Z_0)$ follows straightforwardly from Theorem 7.3 and Theorem 8.3.

THEOREM 5.1 *$R^*(T, Z_0) = J^*$.*

Proof: For $K \subset \mathcal{C}$, if $K \subset R^*(T, Z_0)$ then $Reach(G, K) \subset R^*(T, Z_0)$. And if $K \subset R^*(T, Z_0)$ and $States(K) \cap States(S) \neq \emptyset$ for a SCC S, then from Theorem 7.3, $K \cup S \subset R^*(T, Z_0)$. Since $[Z_0] \subset R^*(T, Z_0)$ and J^* is obtained from $[Z_0]$ by only repeated application of the above two operations, it follows that $J^* \subset R^*(T, Z_0)$.

Since $States(J^*)$ is a closed set containing Z_0, from Theorem 8.3, for any $y \in Reach(T_\epsilon, Z_0)$, $dist(y, J^*) \leq \alpha$ where α can be made arbitrarily small. It follows that $R^*(T, Z_0) \subset J^*$. ■

6. Properties of Zones and Regions

We defined zones in Section 2.3 and regions in Section 3.3. Here we make the relationship between zones and regions, and discuss some of their properties. We will need these results in Sections 7 and 8.

6.1. Regions

A region $r = \{x \mid \lfloor x \rfloor = \alpha \text{ and } \langle x \rangle = \beta\}$ where $\alpha \in \mathbb{Z}^n$ and β is an ordering of the fractional components. Each region r can be equivalently described by a set of inequalities.

Example 6.1: Consider the region $r = \{y \mid \lfloor y \rfloor = (0\ 4\ 3\ 8) \text{ and } (0 \leq \langle y_3 \rangle \leq \langle y_1 \rangle = \langle y_4 \rangle \leq \langle y_2 \rangle)\}$. The region r is equivalently described by the following set of inequalities:

$$3 \leq y_3, \quad y_3 - 3 \leq y_1 - 0, \quad y_1 - 0 = y_4 - 8, \quad y_4 - 8 \leq y_2 - 4, \quad y_2 \leq 5.$$

The vertices of a region can be determined from the description of the region. The vertices of the region r in Example 6.1 are $Vertex(r) = \{(0\ 4\ 3\ 8), (0\ 5\ 3\ 8), (1\ 5\ 3\ 9), (1\ 5\ 4\ 9)\}$. A m-dimensional region has $m + 1$ vertices.

THEOREM 6.1 *Each region is a simplex and every face of a region is itself a region.*

Two consequences of Theorem 6.1 are of interest: in a region r, a point $x \in r$ is a unique convex combination of the vertices of r; and for any $V \subset \mathit{Vertex}(r)$, $\mathit{conv}(V)$ is a face of r and hence is a region.

6.2. Zones

A zone ($\mathbb{R}$-zone) $Z \subset \mathbb{R}^n$ is defined by a set of inequalities of the form

$$x_i - x_j \leq u_{ij}, \quad l_i \leq x_i \leq u_i \tag{3}$$

where $i, j \in \{1, \ldots, n\}$ and $u_{ij}, l_i, u_i \in \mathbb{Z}$ ($u_{ij}, l_i, u_i \in \mathbb{R}$).

LEMMA 6.1 *The following properties relate zones and regions:*

1. *Each region is a zone.*
2. *If Z is a zone then $Z = \cup_i r_i$ where each r_i is a region.*
3. *If $W = \cup_i Z_i$ is convex and each Z_i is a zone, then W is a zone.*

LEMMA 6.2 *For zones ($\mathbb{R}$-zones) Z_1 and Z_2*

1. *$Z_1' = \{y \mid y = x + t, x \in Z_1,$ and $t \geq 0\}$ is a zone ($\mathbb{R}$-zone).*
2. *$Z_1 \cap Z_2$ is a zone ($\mathbb{R}$-zone).*
3. *Projection of Z_1 onto a set of variables is a zone ($\mathbb{R}$-zone).*

A vertex of a $\mathbb{R}$-zone is obtained by solving n simultaneous equations of the form $x_i - x_j = u_{ij}$ and $x_i = u_i$ or $x_i = l_i$. It is easy to check that the vertex then satisfies the following property.

LEMMA 6.3 *Suppose v is a vertex of a $\mathbb{R}$-zone Z from Equation 3. Then v_i is a sum of at most n coefficients u_{ij}, l_i, u_i.*

In particular, if v is a vertex of a zone then $v \in \mathbb{Z}^n$. The next lemma states that for two non-intersecting zones, the points in the two zones are at least $\frac{1}{2}$ distance away.

LEMMA 6.4 *Suppose Z_1 and Z_2 are zones such that $Z_1 \cap Z_2 = \emptyset$. Then for any $x \in Z_1$ and $y \in Z_2$, $\|x - y\| \geq \frac{1}{2}$.*

Proof: Since $Z_1 \cap Z_2 = \emptyset$, it must be the case for some i, j, $x_i - x_j \leq k$ and $y_i - y_j \geq k+1$. Therefore $\|x - y\| \geq \frac{1}{2}$. ■

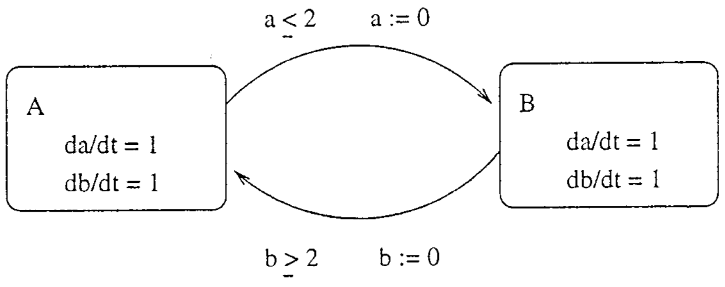

Figure 7. An example timed automaton.

6.3. Zones and Trajectories

Consider a path $p = p_0 p_1 \ldots p_N$ in the region graph of the timed automaton T. Define the set-valued map R: $p_0 \rightarrow p_N$ where $R(q_0) = \{q_N \mid \pi = (q_0, t_0)(q_1, t_1) \ldots (q_N, t_N)$ is a trajectory of T and $q_i \in p_i\}$. For $q_0 \in p_0$, $R(q_0)$ is the set of points which can be reached in p_N from q_0 by passing through the regions $p_1 \ldots p_N$.

LEMMA 6.5 *For $q_0 \in p_0$, $R(q_0)$ is a $\mathbb{R}$-zone, and for a region $r_0 \subset p_0$, $R(r_0)$ is a region.*

Proof: Using Lemma 6.2. ■

Definition 6.1. For $p = p_0 p_1 \ldots p_N$ where $p_i \in \mathcal{C}$ define the set $F_T(p) = \{(-q_0, t_0, t_1, \ldots, t_N) \mid \pi = (q_0, t_0)(q_1, t_1) \ldots (q_N, t_N)$ is a trajectory of T and $q_i \in p_i\}$.

LEMMA 6.6 *$F_T(p)$ is a zone.*

Proof: See the discussion in Example 6.2. ■

Example 6.2: Consider the timed automaton of Figure 7 and $p = p_0 p_1 p_2 p_3$ where $p_o = (A, (0 \le a \le 1) \wedge (0 \le b \le 1) \wedge (b \le a))$, $p_1 = (A, (1 \le a \le 2) \wedge (1 \le b \le 2) \wedge (b \le a))$, $p_2 = (B, (a = 0) \wedge (1 \le b \le 2))$ and $p_3 = (B, (0 \le a \le 1) \wedge (b = 2))$. A trajectory is $\pi = (A, a_0, b_0, t_0)(A, a_1, b_1, t_1)(B, a_2, b_2, t_2)(B, a_3, b_3, t_3)$. The zone $F_T(p)$, expressed in terms of the variables $-a_0, -b_0, t_0, t_1, t_2, t_3$, is given by the following inequalities:

$$
\begin{aligned}
&-1 \le -a_0 \le 0, \quad -1 \le -b_0 \le 0, \quad -a_0 \le -b_0 \\
&1 \le t_1 - (-a_0) \le 2, \quad 1 \le t_1 - (-b_0) \le 2, \quad -a_0 \le -b_0, \quad t_0 \le t_1 \\
&t_2 = t_1 \\
&0 \le t_3 - t_2 \le 1, \quad t_3 - (-b_0) = 2, \quad t_2 \le t_3
\end{aligned}
$$

The variables $a_1, b_1, a_2, b_2, a_3, b_3$ can be expressed in terms of the variables $a_0, b_0, t_0, t_1, t_2, t_3$. For example, $a_1 = t_1 + a_0, b_1 = t_1 + b_0, a_2 = 0, b_2 = t_2 + b_0, a_3 = t_3 - t_2$ and $b_3 = t_3 + b_0$.

7. Limit Cycles of Timed Automata

In this section, we will study the limit cycles of timed automata. Our main result will be Theorem 7.3. The theorem states that in a SCC S of the region graph G of the timed automaton T, it is possible to move from any state $x \in \mathit{States}(S)$ to any other state $y \in \mathit{States}(S)$ in the timed automaton T_ϵ. To prove this result, we will need to understand the structure of the limit cycles of the timed automaton T.

A state q has a limit cycle through it provided there is a trajectory $\pi[0, t]$ with $t > 0$ such that $\pi(0) = q$ and $\pi(t) = q$. We call $\pi[0, t]$ a limit cycle (Hirsh and Smale, 1974). Clearly for state q to have a limit cycle through it, there must be a cycle through $[q]$ in the region graph.

Our approach to analyzing the limit cycles will be to focus on a cycle $c = c_0 c_1 \ldots c_N$ with $c_N = c_0$ in the region graph. We then look at the return map R: $c_0 \to c_0$ where $R(x)$ is the set of states in c_0 to which it is possible to return after one cycle starting from state $x \in c_0$. We analyze the limit cycles using the return map R. One of our main results is that the set of points in c_0 with limit cycles forms a region. Theorem 7.3 will be a simple consequence of our results about limit cycles and the return map R.

In Section 7.1, we define the return map. Section 7.2 shows some examples of limit cycles. Section 7.3 describes the properties of the return map. In Section 7.4, we show that the behavior of the return map is described completely by its behavior on the vertices of the region c_0. We also show that the set of points with limit cycles forms a region. In Section 7.5, we show that for the class of simple timed automata, every point $q \in c_0$ has a limit cycle through it. In Section 7.6, we show that this is not the case for timed automata in general. In Section 7.7 we prove Theorem 7.3.

7.1. The Return Map

Consider a cycle $c = c_0 c_1 c_2 \ldots c_N$ in the region graph where $c_N = c_0$. Define the *return map* R: $c_0 \to c_0$ where $R(q_0) = \{q_N \mid \pi = (q_0, t_0)(q_1, t_1) \ldots (q_N, t_N)$ is a trajectory of the timed automaton and $q_i \in c_i\}$. R is a set-valued map where for $x \in c_0$, $R(x)$ is the set of points to which it is possible to return in c_0 after passing through the regions $c_1, \ldots, c_N$. From Lemma 6.5, we know that for $x \in c_0$, $R(x)$ is a $\mathbb{R}$*-zone*, and for a region $r_0 \subset c_0$, $R(r_0)$ is a region.

We define $L_1 = \{x \mid x \in R(x)\}$, $L_2 = \{x \mid x \in R^2(x)\}$ and more generally $L_m = \{x \mid x \in R^m(x)\}$. So L_m is the set of points which can return back to themselves after m cycles. The set of points with limit cycles is $L = \cup_i L_i$.

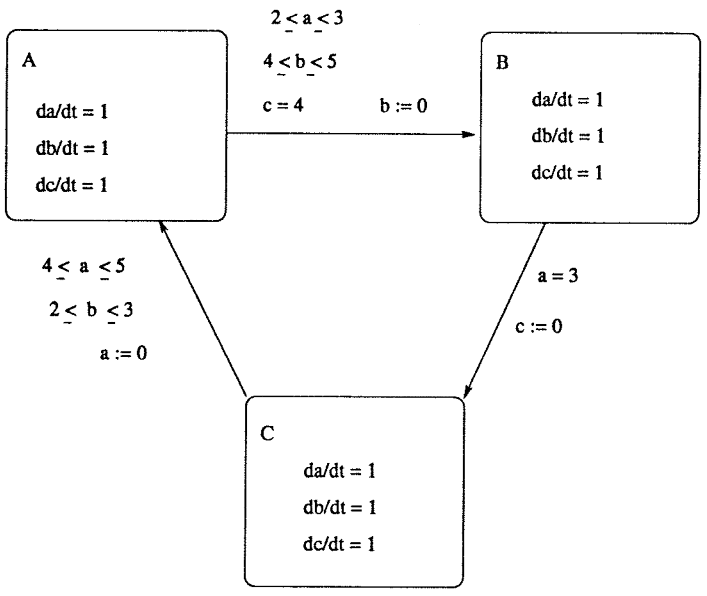

Figure 8. A timed automaton.

7.2. Examples of Limit Cycles

Figure 8 shows a timed automaton. Consider the region $c_0 = (A, r)$ where $r = \{x \mid \lfloor x \rfloor = 0\ 2\ 1)$ and $\langle x \rangle = (0 = \langle a \rangle \leq \langle b \rangle \leq \langle c \rangle)\}$. The reader can check that the region c_0 has only one cycle through it in the region graph which goes through the control locations A, B and C only once.

Now consider the return map R: $c_0 \rightarrow c_0$. The set $L_1 = \{A\} \times \{x \mid 2c = b+1 \text{ and } x \in c_0\}$ and $L_2 = c_0$. The sets L_1 and L_2 are shown in Figure 9. Since $L = L_2 = c_0$, every point of c_0 has a limit cycle through it.

7.3. Properties of the Return Map

We next discuss some properties of the return map. We will need the following lemmae first.

LEMMA 7.1 *Suppose $r \longrightarrow_G r'$ in the region graph G, and $x \rightarrow x'$ and $y \rightarrow y'$ where $x, y \in r$ and $x', y' \in r'$. Then $\lambda x + (1-\lambda)y \rightarrow \lambda x' + (1-\lambda)y'$ for $0 \leq \lambda \leq 1$.*

LEMMA 7.2 *Suppose $r \longrightarrow_G r'$, $z = \lambda x + (1-\lambda)y$ and $z \rightarrow z'$ where $x, y \in r$ and $z' \in r'$. Then there are $x', y' \in r'$ such that $z' = \lambda x' + (1-\lambda)y'$ and $x \rightarrow x'$ and $y \rightarrow y'$.*

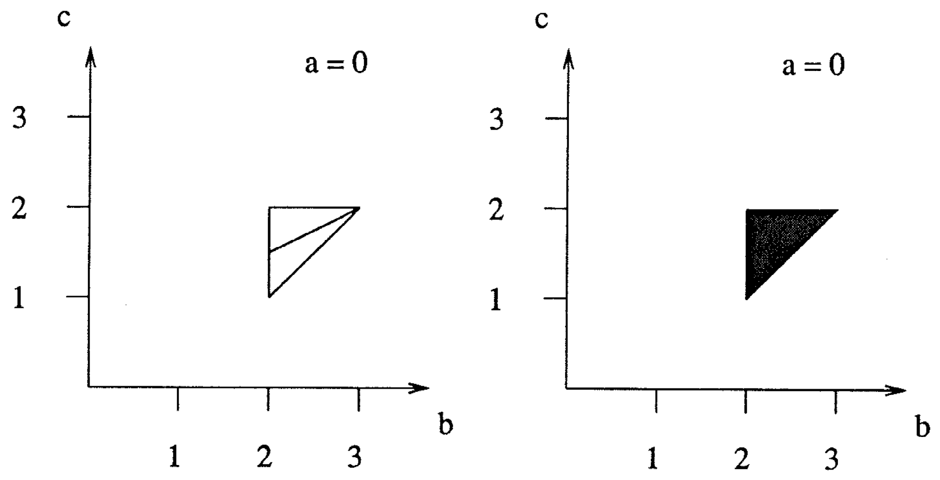

Figure 9. The sets L_1 and L_2.

Proof (Sketch): We will only show that $z \xrightarrow{t} z'$ implies that there are $x', y' \in r'$ such that $z' = \lambda x' + (1 - \lambda)y'$, $x \xrightarrow{t_1} x'$ and $y \xrightarrow{t_2} y'$. We will only need to show this for the case where r' is an immediate successor of r.

So suppose $z \xrightarrow{t} z'$ and $\langle z_m \rangle \geq \langle z_i \rangle$ for each i. Then define

$$x' = x + \frac{t(1 - \langle x_m \rangle)}{1 - \langle z_m \rangle}$$

and

$$y' = y + \frac{t(1 - \langle y_m \rangle)}{1 - \langle z_m \rangle}.$$

It can be checked that

$$\lambda x' + (1 - \lambda)y' = \lambda x + (1 - \lambda)y + t = z + t = z'$$

and $x' \in [z']$ and $y' \in [z']$. ■

THEOREM 7.1 *$R(\lambda x + (1 - \lambda)y) = \lambda R(x) + (1 - \lambda)R(y)$ for $0 \leq \lambda \leq 1$.*

Proof: Using Lemma 7.1 and Lemma 7.2. ■

We next note that the set of points with limit cycles through them is a convex set.

LEMMA 7.3 *L is a convex set.*

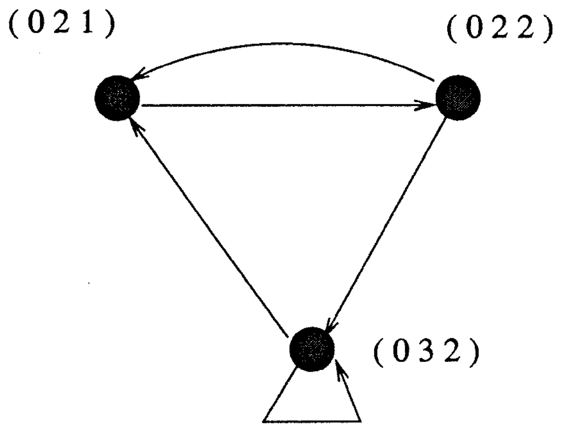

Figure 10. Orbit graph.

Proof: Suppose $x, y \in L$. Then $x \in L_k$ and $y \in L_l$ for k and l. Therefore, $x, y \in L_m$ for $m = kl$ (i.e., $x \in R^m(x)$ and $y \in R^m(y)$). And hence, $\lambda x + (1 - \lambda)y \in R^m(\lambda x + (1 - \lambda)y)$. ■

7.4. Orbit Graph of Vertices of a Region

Consider again a cycle $c = c_0c_1 \dots c_N$ in the region graph. Define the orbit graph $\Theta = (V_\theta, \rightarrow_\theta)$ where the vertices V_θ are the vertices of the region c_0 and for $v, w \in V_\theta$, there is an edge $v \rightarrow_\theta w$ provided $w \in R(v)$.

LEMMA 7.4 *Suppose $r \longrightarrow_G r'$ in the region graph G. Then for each vertex u of region r, there exists a vertex v of region r' such that $u \rightarrow v$; and for each vertex v' of r', there exists a vertex u' of r such that $u' \rightarrow v'$.*

From Lemma 7.4, each vertex of Θ has at least one incoming edge and at least one outgoing edge.

Example 7.1: Consider again the timed automaton of Figure 8 and the region $c_0 = (A, r)$ where $r = \{x \mid \lfloor x \rfloor = (0\ 2\ 1)$ and $\langle x \rangle = (0 = \langle a \rangle \leq \langle b \rangle \leq \langle c \rangle)\}$. The vertices of region c_0 are $V_\theta = \{A\} \times \{(0\ 2\ 1), (0\ 2\ 2), (0\ 3\ 2)\}$. The orbit graph for this example is shown in Figure 10.

LEMMA 7.5 *For $v \in V_\theta$, $R(v)$ is a region and $R(v) = conv\{w \mid v \rightarrow_\theta w\}$.*

Proof: From Lemma 6.5, $R(v)$ is a region. It also follows $R(v) = conv\{w \mid v \rightarrow_\theta w\}$. ■

From Theorem 7.1, $R(x)$ is then a convex combination of $R(v_i)$, $v_i \in V_\theta$.

LEMMA 7.6 *Suppose $y \in R(x)$ where $x = \Sigma_i \lambda_i v_i$, $v_i \in V_\theta$, $\lambda_i \geq 0$, $\Sigma_i \lambda_i = 1$. Then $y = \Sigma_i \lambda_i \Sigma_j p_{ij} w_{ij}$ where $w_{ij} \in V_\theta$, $p_{ij} \geq 0$, $\Sigma_j p_{ij} = 1$ and $p_{ij} = 0$ if there is no edge from v_i to w_{ij} in Θ.*

Proof: $y \in R(\Sigma_i \lambda_i v_i) = \Sigma_i \lambda_i R(v_i)$ from Theorem 7.1. So $y = \Sigma_i \lambda_i s_i$ where from Lemma 7.5 $s_i = \Sigma_j p_{ij} w_{ij}$ where $w_{ij} \in V_\theta$ and $p_{ij} = 0$ if there is no edge from v_i to w_{ij} in Θ. Thus $y = \Sigma_i \lambda_i \sigma_j p_{ij} w_{ij}$. ■

Example 7.2: Consider the return map of Example 7.1 and

$$x = \frac{1}{3}(0, 2, 1) + \frac{1}{6}(0, 2, 2) + \frac{1}{2}(0, 3, 2).$$

Then $y \in R(x)$ from Lemma 7.6 and Figure 10 for

$$y = \frac{1}{3}((0, 2, 2)) + \frac{1}{6}\left(\frac{1}{2}(0, 2, 1) + \frac{1}{2}(0, 3, 2)\right) + \frac{1}{2}\left(\frac{2}{3}(0, 3, 2) + \frac{1}{3}(0, 2, 1)\right).$$

The limit cycles of points in c_0 can also be described in terms of the limit cycles of points in V_θ.

LEMMA 7.7 *$v \in V_\theta$ has a limit cycle iff v has a cycle through it in the orbit graph Θ.*

THEOREM 7.2 *$L = conv(V)$ where $V \subset V_\theta$ are the vertices of Θ with a cycle.*

Proof: Clearly $conv(V) \subset L$. Now assume $x \in L$ (i.e., $x \in R^k(x)$ for some k). Since $x = \Sigma_i \lambda_i v_i$ where $\lambda_i > 0$, $\Sigma_i \lambda_i = 1$ and $v_i \in V_\theta$, from Lemma 7.6 and Theorem 6.1, each $v_i \in V$. Therefore $x \in conv(V)$. ■

From Theorem 6.1, it follows that L is a region.

7.5. Simple Timed Automata

A timed automaton $T = (L, E)$ is *simple* provided each guard $g \subset h_i$ where $h_i = \{x \mid x_i = k\}$. That is, each guard g is contained in a horizontal hyperplane. Figure 11 is an example of a simple timed automaton.

Let us now consider a cycle $c = c_0 c_1 \ldots c_N$ in the region graph of a simple timed automaton T. Without loss of generality, we assume that c_0 is contained in a horizontal hyperplane. We study the orbit graph $\Theta = (V_\theta, \rightarrow_\theta)$ and the limit cycles.

Because T is simple, $R(x)$ for $x \in c_0$ is a singleton. Hence, each vertex of V_θ has exactly one outgoing edge. From Lemma 7.4, each vertex also has at least one incoming edge. Therefore, each vertex of V_θ has exactly one incoming and one outgoing edge. As a consequence, each vertex of V_θ has a cycle through it.

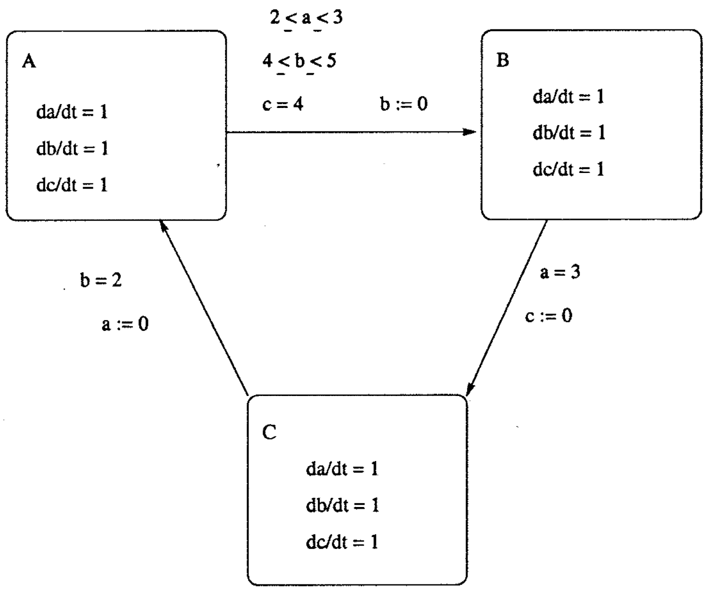

Figure 11. A simple timed automaton.

LEMMA 7.8 *In the orbit graph $\Theta = (V_\theta, \rightarrow_\theta)$ of a simple timed automaton, each vertex of V_θ has a cycle through it.*

LEMMA 7.9 *In a cycle $c = c_0 c_1 \ldots c_N$ in the region graph of a simple timed automaton, the set of points in c_0 with limit cycles is $L = c_0$.*

Example 7.3: Figure 11 shows a simple timed automaton. Consider the region $c_0 = (A, s)$ where $s = \{x \mid \lfloor x \rfloor = (0\ 2\ 1)$ and $\langle x \rangle = (0 = \langle a \rangle = \langle b \rangle \leq \langle c \rangle)\}$. The reader can check that c_0 has exactly one cycle through it in the region graph. The vertices of c_0 are $V_\theta = \{A\} \times \{(0\ 2\ 1), (0\ 2\ 2)\}$. The orbit map for this example is shown in Figure 12. The set $L_1 = \{A\} \times \{\frac{1}{2}(0\ 2\ 1) + \frac{1}{2}(0\ 2\ 2)\} = (A, (0, 2, 1.5))$ and the set $L_2 = c_0$. Since $L = L_2 = c_0$, each point in c_0 has a limit cycle through it.

7.6. General Timed Automata

Given a cycle $c = c_0 c_1 \ldots c_N$ in the region graph, from Theorem 7.2 we know that L is a region and $L \subset c_0$. In Section 7.5, we showed that for simple timed automata $L = c_0$. Is this always the case? We next show with an example that this is not always the case. That is, there are cases where L is a proper subset of c_0.

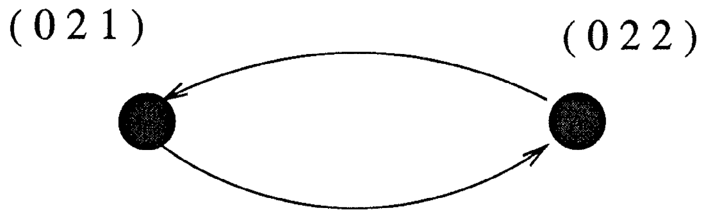

Figure 12. Orbit graph of timed automaton of Figure 11.

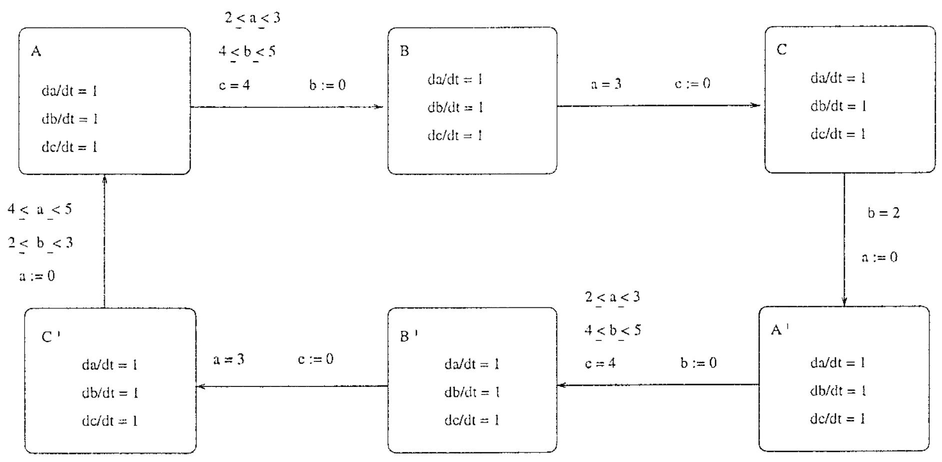

Figure 13. Timed automaton obtained by "concatenating" the automata of Figure 8 and Figure 11.

Example 7.4: Consider the timed automaton of Figure 13. The automaton is obtained by "concatenating" the automaton of Figure 8 with the automaton of Figure 11. Consider the region $c_0 = (A, r)$ where $r = \{x \mid \lfloor x \rfloor = (0\ 2\ 1)$ and $\langle x \rangle = \langle 0 = \langle a \rangle \leq \langle b \rangle \leq \langle c \rangle)\}$. The reader can check that c_0 has exactly one cycle through it in the region graph of timed automaton of Figure 13. The vertices of c_0 are $V_\theta = \{A\} \times \{(0\ 2\ 1), (0\ 2\ 2), (0\ 3\ 2)\}$. The orbit map for this example is shown in Figure 14. The set $L = \{A\} \times \{x \mid a = 0, b = 2$ and $1 \leq c \leq 2\}$. So L is a region and it is a proper subset of c_0. In particular, the point $(A, (0, 3, 2))$ does not have a limit cycle through it.

Example 7.4 shows that even though c_0 has a cycle through it in the region graph, there are points in c_0 which do not have limit cycles. We next show that for any point $x \in c_0$ there are points u and v in L such that $x \Rightarrow u$ and $v \Rightarrow x$ in T.

LEMMA 7.10 *Consider a cycle $c = c_0 c_1 \ldots c_N$ in the region graph. Then for any $x \in c_0$, there exist $u, v \in L$ such that $x \Rightarrow u$ and $v \Rightarrow x$ in T.*

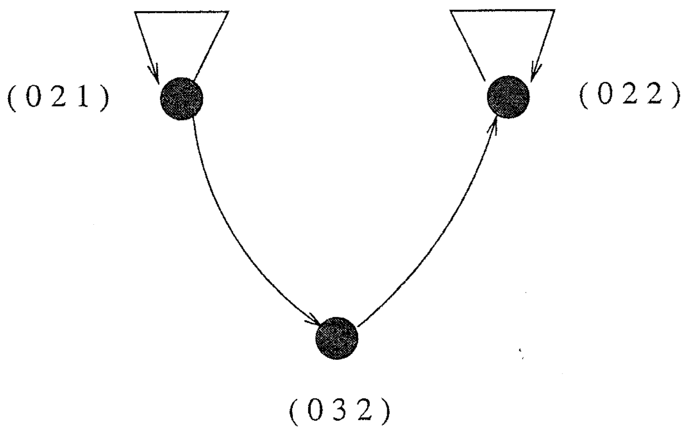

Figure 14. Orbit graph of timed automaton of Figure 13.

Proof: By writing $x = \Sigma_i \lambda_i v_i$ where $v_i \in V_\theta$, and then using Lemma 7.6 and the properties of the orbit graph. ■

7.7. Main Result

We now show that in a strongly connected component S of the region graph of T, $x \Rightarrow y$ in T_ϵ for any $x, y \in States(S)$. Let us first focus on a cycle $c = c_0 c_1 \ldots c_N$ in the region graph.

LEMMA 7.11 *Consider a cycle $c = c_0 c_1 \ldots c_N$ in the region graph. Then for $u, v \in L$, $u \Rightarrow v$ in T_ϵ.*

Proof: L is a convex set in which each point has a limit cycle through it. In timed automaton T_ϵ, we can reach state $v \in L$ from state $u \in L$ by drifting from one limit cycle to another until we reach state v.

More formally, since L is convex, $w = \lambda v + (1 - \lambda) u$ contains a limit cycle for $0 \leq \lambda \leq 1$. It can also be shown that there is a $\delta > 0$ such that if $\pi[0, t]$ is a limit cycle through a point, then its period $t \geq \delta$. For every $\epsilon > 0$, there is a $\beta_1 > 0$ such that for any $\beta_2 < \beta_1$, it is possible to move from $\lambda v + (1 - \lambda) u$ to $(\lambda + \beta_2) v + (1 - (\lambda + \beta_2)) u$ in T_ϵ. Hence, it is possible to move from u to v in T_ϵ. ■

LEMMA 7.12 *For $x, y \in c_0$, $x \Rightarrow y$ in timed automaton T_ϵ.*

Proof: Using Lemma 7.10 and Lemma 7.11. ■

THEOREM 7.3 *Suppose S is a SCC of the region graph G, and x, y are two points in S. Then $x \Rightarrow y$ in T_ϵ for any $\epsilon > 0$.*

Proof: By considering a cycle of S which contains x and y and using Lemma 7.12. ■

8. Relationship Between Trajectories of T and T_ϵ

In this section we discuss the relationship between the trajectories of T and T_ϵ. We show that for any trajectory of T_ϵ, there is a trajectory of T which is close to it. In particular, we show that in a timed automaton with state space $Q_T \subset L \times [0, M]^n$, for any trajectory π' of T_ϵ which makes k jumps, there is a trajectory π of T which makes the same k jumps and is at most $4(k+n)(M+1)\epsilon$ distance away from π'.

We will use this result to prove Theorem 8.3: starting from a point in J^*, for any $\alpha > 0$, it is not possible to get more than distance α away from J^* in T_ϵ for sufficiently small ϵ.

8.1. Zones Z and Z_β

In proving our results, we will need to understand the relationship between the solutions to the zone Z and the solutions to the $\mathbb{R}$-zone Z_β. Z_β is obtained from Z by replacing each inequality $x_i - x_j \le u_{ij}$ ($l_i \le x_i \le u_i$) in Z with the inequality $x_i - x_j \le u_{ij} + \beta$ ($l_i - \beta \le x_i \le u_i + \beta$) in Z_β. The following theorem makes the relationship between solutions of Z and Z_β.

THEOREM 8.1 *Suppose Z is a zone with variables x_i, $i = 1, \ldots, K$. We define the $\mathbb{R}$-zone Z_β where each inequality $x_i - x_j \le u_{ij}$ ($l_i \le x_i \le u_i$) in Z is replaced with the inequality $x_i - x_j \le u_{ij} + \beta$ ($l_i - \beta \le x_i \le u_i + \beta$) in Z_β. Then for $\beta < \frac{1}{2K+1}$, for any $d \in Z_\beta$, there exists a $d' \in Z$ such that $|d - d'| \le K\beta$.*

Proof: We only need to show that for any vertex $d \in Z_\beta$, there exists a $d' \in Z$ such that $|d - d'| \le K\beta$. Suppose d is a vertex of Z_β. Then from Lemma 6.3 $d_i = m_i + k_i\beta$ where $|k_i| \le K$ and $m_i \in \mathbb{Z}$. Define $d'_i = m_i$. Then $d' \in Z$. This is because $d_i - d_j = m_i - m_j + (k_i - k_j)\beta \le u_{ij} + \beta$. Since m_i, m_j, u_{ij} are integers, $\beta < \frac{1}{2K+1}$ and $|k_i - k_j - 1| \le 2K + 1$, $d'_i - d'_j = m_i - m_j \le u_{ij}$. ■

8.2. Trajectories of T_ϵ and $F_{T_\epsilon}(p)$

For $p = p_0 p_1 \ldots p_k$ where each p_i is a region, define $F_{T_\epsilon}(p) = \{(-q'_0, t'_0, \ldots, t'_k) \mid \pi' = (q'_0, t'_0)(q'_1, t'_1) \ldots (q'_k, t'_k)$ is a trajectory of T_ϵ and $q'_i \in p_i\}$. It can then be shown that if the variables $(-q_0, t_0, \ldots, t_k)$ satisfy the inequalities

$$a \le t_i - t_j \le b, \quad a \le t_i - (-q_{0j}) \le b$$

in $F_T(p)$, then the variables $(-q_0', t_0', \ldots, t_k')$ satisfy the inequalities

$$a - 2(M+1)\epsilon \leq t_i' - t_j' \leq b + 2(M+1)\epsilon,$$
$$a - (M+1)\epsilon \leq t_i' - (-q_{oj}') \leq b + (M+1)\epsilon$$

in $F_{T_\epsilon}(p)$ for sufficiently small ϵ. Lemma 8.1 is a consequence of this observation.

LEMMA 8.1 *Suppose $Z = F_T(p)$ where $Q_T \subset L \times [0, M]^n$. Then for sufficiently small ϵ, $Z \subset F_{T_\epsilon}(p) \subset Z_\beta$ for $\beta \geq 2(M+1)\epsilon$.*

We will now make the relationship between the trajectories of T and T_ϵ.

THEOREM 8.2 *Suppose we are given a $0 < \delta < 1$ and a trajectory $\pi' = (q_0', t_0') \ldots (q_k', t_k')$ of T_ϵ where $\epsilon < \frac{\delta}{4(k+n)(M+1)}$. Then there is a trajectory $\pi = (q_0, t_0) \ldots (q_k, t_k)$ of T such that $q_i \in [q_i']$, $|t_i' - t_i| < \frac{\delta}{2}$ and $|q_i' - q_i| < \delta$.*

Proof: Consider $p = p_0 p_1 \ldots p_k$ where $p_i = [q_i']$. The result follows by making the relationship between $F_{T_\epsilon}(p)$ and $F_T(p)$ using Lemma 8.1 and Theorem 8.1. ■

A consequence of Theorem 8.2 is that for a state $q \in R^*(T_\epsilon, Z_0)$ but $q \notin \mathit{Reach}(T, Z_0)$, the time it takes to reach q in T_ϵ becomes longer as ϵ becomes smaller.

8.3. Main Result

We are now ready to prove the main result of this section: starting from a point In J^*, it is not possible to get more than α distance away from J^* in the timed automaton T_ϵ for sufficiently small ϵ.

The state space of timed automaton T is $Q_T \subset L \times [0, M]^n$ and W is the number of regions in Q_T.

THEOREM 8.3 *Suppose $x \in J^*$ and $x \Rightarrow y$ in timed automaton T_ϵ with $\epsilon < \frac{\alpha}{4(n+W+1)(M+1)}$ where $\alpha < \frac{1}{4}$. Then $\mathit{dist}(y, J^*) < \alpha$.*

Proof: Suppose $\pi' = (q_0', t_0')(q_1', t_1') \ldots (q_m', t_m')$ is a stutter-free trajectory of T_ϵ where $q_0' = x$, $q_m' = y$ and $q_0' \in J^*$. We need to show that $\mathit{dist}(q_m', J^*) < \alpha$. The proof will be by using induction. Using Theorem 8.2, $\mathit{dist}(q_i', J^*) < \alpha$ for $i \leq W+1$. Now assume that $\mathit{dist}(q_i', J^*) < \alpha$ for $i \leq N$ where $N \geq W+1$. By our induction hypothesis $\mathit{dist}(q_j', J^*) < \alpha$ for $j = (N+1) - (W+1)$. Consider $p = p_j p_{j+1} \ldots p_{N+1}$ where $p_i = [q_i']$. From Theorem 8.2, there is a trajectory $\eta = (q_j, t_j)(q_{j+1}, t_{j+1}) \ldots (q_{N+1}, t_{N+1})$ in T where for $j \leq i \leq N+1$, $\|q_i - q_i'\| < \alpha$ and $q_i \in [q_i']$. Now $\mathit{dist}(q_j, J^*) < \frac{1}{2}$ because $\mathit{dist}(q_j', J^*) < \alpha$ and $\|q_j - q_j'\| < \alpha$. Therefore from Lemma 6.4 $\mathit{States}([q_j]) \cap \mathit{States}(J^*) \neq \emptyset$. By construction of J^* (since the path $[q_j][q_{j+1}] \ldots [q_{N+1}]$ contains a cycle, $[q_{N+1} \subset J^*$), it follows that $q_{N+1} \in J^*$. Therefore $\mathit{dist}(q_{N+1}', J^*) < \alpha$. ■

9. Robustness of $R^*(T, Z_0)$ and Approximations

In Section 5, we presented an algorithm for computing the set $R^*(T, Z_0)$. We next show that $R^*(T, Z_0)$ is robust against various types of modeling errors. We then describe an approximation method which allows us to determine whether a control location l_f is reachable in T_ϵ without computing $R^*(T, Z_0)$.

9.1. Modeling Errors

Consider a timed automaton T with state space $Q_T \subset L \times [0, M]^n$ and W regions. We first show that control location l_f is reachable in T_ϵ for sufficiently small ϵ iff it is reachable in $R^*(T, Z_0)$.

LEMMA 9.1 *In a timed automaton T, control location l_f is reachable in T_ϵ for $\epsilon < \frac{1}{16(n+W+1)(M+1)}$ iff l_f is reachable in $R^*(T, Z_0)$.*

Proof: Using Lemma 6.4 and Theorem 8.3. ■

Define the automaton T_ϵ^δ as being obtained from the timed automaton T by replacing clock $\dot{x} = 1$ with the clock $\dot{x} \in [1 - \epsilon, 1 + \epsilon]$ and by replacing the guard $a \leq x \leq b$ in T with the guard $a - \delta \leq x \leq b + \delta$.

THEOREM 9.1 $\cap_{\delta>0} \cap_{\epsilon>0} Reach(T_\epsilon^\delta, Z_0) = R^*(T, Z_0)$.

Proof: The proof is similar to that of Theorem 5.1. ■

The theorem states that for sufficiently small ϵ and δ, $Reach(T_\epsilon^\delta, Z_0)$ is close to $R^*(T, Z_0)$. Hence, $R^*(T, Z_0)$ is robust against errors in guard and drifts in clocks.

9.2. Approximations

We next show that for the control location reachability problem, if the control location l_f is not reachable in T_0^δ then for sufficiently small ϵ, it is also not reachable in T_ϵ^0. This implies that to check that l_f is not reachable in T_ϵ^0, it is sufficient to check this for T_0^δ.

LEMMA 9.2 *Suppose the control location l_f is not reachable in the timed automaton T_0^δ. Then l_f is also not reachable in T_ϵ^0 for $\epsilon \leq \frac{\delta}{2(M+1)}$.*

Proof: Using Lemma 8.1. ■

10. Complexity

We next consider the complexity of computing $R^*(T, Z_0)$. In Courcoubetis and Yannakakis (1992), it is shown that determining whether a state $q \in Reach(T, Z_0)$ is PSPACE-

Complete. We note that determining whether a state $q \in R^*(T, Z_0)$ is also PSPACE-Complete.

THEOREM 10.1 *For a timed automaton T and an initial zone Z_0, determining whether a state $q \in R^*(T, Z_0)$ is PSPACE-Complete.*

Proof: Membership in PSPACE follows from our algorithm for computing $R^*(T, Z_0)$. The hardness result follows by noting that determining whether $q \in \mathit{Reach}(T, Z_0)$ is PSPACE-hard (Courcoubetis and Yannakakis, 1992), and for the class of timed automata used to show this in (Courcoubetis and Yannakakis, 1992), $\mathit{Reach}(T, Z_0) = R^*(T, Z_0)$. ■

Given a timed automaton T and an initial zone Z_0, is it easy to tell whether $R^*(T, Z_0) = \mathit{Reach}(T, Z_0)$? We next note that this problem is also PSPACE-complete.

THEOREM 10.2 *For a timed automaton T and an initial zone Z_0, determining whether $\mathit{Reach}(T, Z_0) = R^*(T, Z_0)$ is PSPACE-Complete.*

Proof (Sketch): We show hardness by showing that the reachability problem for timed automata can be translated to our problem. We take the timed automaton T from Courcoubetis and Yannakakis (1992) for which $\mathit{Reach}(T, Z_0) = R^*(T, Z_0)$ and create a new automaton T' such that $q \in \mathit{Reach}(T, Z_0)$ iff $\mathit{Reach}(T', Z_0) \neq R^*(T', Z_0)$. This is done by concatenating from T "at state q" the automaton of Figure 2 to create T'. Then $q \in \mathit{Reach}(T, Z_0)$ iff $\mathit{Reach}(T', Z_0) \neq R^*(T', Z_0)$.

Membership in PSPACE follows because determining if $q \in \mathit{Reach}(T, Z_0)$ is in PSPACE and determining whether $q \in R^*(T, Z_0)$ is in PSPACE. ■

Although for a timed automaton T, determining whether $R^*(T, Z_0) = \mathit{Reach}(T, Z_0)$ is PSPACE-hard, it is of interest to find some simple sufficient conditions which guarantee this.

11. Conclusion

Our main contribution in this paper is an algorithm for computing $R^*(T, Z_0)$ and a proof of its correctness. We showed that unlike $\mathit{Reach}(T, Z_0)$, $R^*(T, Z_0)$ is robust against various types of modeling errors.

We also presented an analysis of the dynamics of timed automata. In particular, we presented a complete analysis of the limit cycles of timed automata.

Although in this paper we have focused on timed automata, it should be clear that the same problems also extend to hybrid systems (Alur et al., 1995; Henzinger et al., 1997; Puri and Varaiya, 1996; Puri, 1995). An analysis of hybrid systems using the approaches of Alur et al. (1995) and Henzinger et al. (1997) will not be robust. But for the class of decidable hybrid systems (Puri and Varaiya, 1996; Puri, 1995), the methods we have presented in this paper should be applicable.

Open issues which require further study include designing efficient algorithms for computing $R^*(T, Z_0)$ and finding simple sufficiency conditions which guarantee that $R^*(T, Z_0) = \mathit{Reach}(T, Z_0)$.

Acknowledgements

I thank reviewers for their helpful comments.

References

Alur, R., and Dill, D. 1990. Automata for modeling real-time systems. *Proc. 17th ICALP*. LNCS 443, Springer-Verlag.

Alur, R., et al. 1995. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*.

Alur, R., Itai, A., Kurshan, R., and Yannakakis, M. 1995. Timing verification by successive approximation. *Information and Computation* 118: 142–157.

Alur, R., and Kurshan, R. 1996. Timing analysis in COSPAN. *Hybrid Systems III*. LNCS 1066, Springer-Verlag.

Bengtsson, J., Larsen, K. G., Larsson, F., Pettersson, P., and Yi, W. 1996. UppAal: a tool-suite for automatic verification of real-time systems. *Hybrid Systems III*. LNCS 1066, Springer-Verlag.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L. 1990. *Introduction to Algorithms*. MIT Press.

Courcoubetis, C., and Yannakakis, M. 1992. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design* 1: 385–415.

Daws, C., Olivero, A., Tripakis, S., and Yovine, S. 1996. The tool Kronos. *Hybrid Systems III*. LNCS 1066, Springer-Verlag.

Dill, D. 1989. Timing assumptions and verification of finite state concurrent systems. *Automatic Verification Methods for Finite-State Systems*. LNCS 407, Springer-Verlag.

Gupta, V., Henzinger, T., and Jagadeesan, R. 1997. Robust timed automata. *First International Workshop on Hybrid and Real-time Systems (HART 1997)*. LNCS 1201, Springer-Verlag.

Henzinger, T., Ho, P.-H., and Wong-Toi, H. 1997. HyTech: A model checker for hybrid systems. *Computer Aided Verification*. LNCS 1254, Springer-Verlag.

Henzinger, T., Nicollin, X., Sifakis, J., and Yovine, S. 1992. Symbolic model-checking for real-time systems. *Proc. 7th IEEE Symp. on Logic in Computer Science*.

Hirsh, M. W., and Smale, S. 1974. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, Inc.

Puri, A., and Varaiya, P. 1996. Decidable hybrid systems. *Computer and Mathematical Modeling* 23: 191–202.

Puri, A. 1995. Theory of hybrid systems and discrete event systems. Ph.D. thesis, University of California, Berkeley.

Yannakakis, M., and Lee, D. 1993. An efficient algorithm for minimizing real-time transition systems. *Computer Aided Verification*. Springer-Verlag.