Proceedings of the 33rd
Conference on Decision and Control
Lake Buena Vista, FL - December 1994

Discretization of Timed Automata *

Aleks Göllü, Anuj Puri, Pravin Varaiya
gollu@eecs.berkeley.edu, anuj@eecs.berkeley.edu, varaiya@eecs.berkeley.edu
Department of EECS
University of California at Berkeley
Berkeley, CA 94720

Abstract

We construct two discretizations of dense time automata which generate the same untimed language as the dense time automata.

1 Introduction

A timed automaton (TA) is an automaton together with a finite set of clocks that constrain the occurrence of transitions. The timed language of the TA consists of pairs of sequences $\{(\sigma_i, \tau_i)\}$, where τ_i is the timestamp of the event σ_i. If τ_i can take any real value, one obtains the dense time language $\mathcal{L}_{CT}$. If τ_i can take discrete values $0, \Delta, 2\Delta, ...$, one obtains the discrete time language $\mathcal{L}_{DT}$. The untimed languages $\mathcal{L}_{CU}$ and $\mathcal{L}_{DU}$ are obtained by dropping the timestamps. We prove for two different discretizations that $\mathcal{L}_{DU} = \mathcal{L}_{CU}$, extending the result of [2]. TA's form the first step in a descriptive hierarchy between (untimed) automata and hybrid systems.

2 Timed Automata

Recall the definition of a TA [1]. It is an automaton together with a set $C = \{1 \cdots K\}$ of timers (clocks) with state space $\mathcal{R}_+^K$. Timer values constrain the occurrence of transitions; and a transition resets some timers to zero. The constraints and resets are expressed as part of the edge labels of the automaton's transition graph. This is formalized as follows.

Let Φ be a finite set of timer *constraints* of the form $\delta := x_i \leq c | x_i \geq c | \delta_1 \wedge \delta_2 | \neg\delta$, where x_i, x_j are values of timers i, j and $c \in Z_+$. We define three basic operators on the timer state space. For $v \in \mathcal{R}_+^K$

- *succ*: $v \xrightarrow{succ} v'$ provided $v_i' = v_i + t$; where $t \in \mathcal{R}_+ \setminus \{0\}$ is the time to execute *succ*
- $\delta \in \Phi$: $v \xrightarrow{\delta} v$ provided $v \in \delta$
- $\lambda \in 2^C$: $v \xrightarrow{\lambda} v'$ provided $v_i' = 0$ for $i \in \lambda$ and $v_i' = v_i$ otherwise

Let $\Gamma = succ \times \Phi \times 2^C$ be the set of timing labels. $\gamma = (succ, \delta, \lambda) \in \Gamma$ is an operator which works from the left, i.e., *succ* first moves timer values by allowing time to pass; δ then checks that the timer state is within a certain region; finally, the timers in λ are **reset** to zero.

A TA is a tuple $\langle \Sigma, S, S_0, C, E, C_0 \rangle$, where Σ is a finite alphabet, S are the automaton states, $S_0 \subseteq S$ are the start states, C are the timers, $E \subseteq S \times [\Sigma \cup \{\epsilon\}] \times \Gamma \times S$ are the state transitions (edges), and C_0 are the initial timer values.

*Research supported in part by the National Science Foundation and the California PATH program in cooperation with the State of California and US DOT. The contents of this report reflects the views of the authors who are responsible for the facts and accuracy of the data presented herein

A *dense time trace* is a pair (σ, τ) where $\sigma \in \Sigma^\omega$ and τ is a progressing time sequence; (σ_i, τ_i) represents the event σ_i occurring at time $\tau_i \in \mathcal{R}_+$.

The dense time trace (σ, τ) generates the *run*

$$(q_0, v_0) \xrightarrow[\tau_0]{(\sigma_0,\gamma_0)} (q_1, v_1) \xrightarrow[\tau_1]{(\sigma_1,\gamma_1)} (q_2, v_2) \xrightarrow[\tau_2]{(\sigma_2,\gamma_2)} \cdots \quad (1)$$

where $v_i \in \mathcal{R}_+^K$, $\tau_i \in \mathcal{R}_+$, provided $(q_i, (\sigma_i, \gamma_i), q_{i+1}) \in E$, $v_i \xrightarrow{\gamma_i} v_{i+1}$, τ_i is the time at which the run enters the state (q_{i+1}, v_{i+1}), and $\tau_{i+1} - \tau_i$ is the time taken to execute γ_i. The timed and untimed languages are defined as: $\mathcal{L}_{CT} = \{(\sigma, \tau) | (\sigma, \tau)$ has a run in dense time semantics$\}$; $\mathcal{L}_{CU} = \{\sigma | \exists (\sigma, \tau) \in \mathcal{L}_{CT}\}$.

In [1] an equivalence relation is defined that partitions the timer state space into *timer regions*: (q, x) and (q, y) in $S \times \mathcal{R}_+^K$ are equivalent if

$$\begin{array}{lllllll} int[x_i] & = & int[y_i] & & & & \\ fr(x_i) & < & fr(x_j) & \iff & fr(y_i) & < & fr(y_j) \\ fr(x_i) & = & fr(x_j) & \iff & fr(y_i) & = & fr(y_j) \\ fr(x_i) & = & 0 & \iff & fr(y_i) & = & 0 \end{array} \quad (2)$$

Here *int* and *fr* are the integer and fractional parts of a timer value. The timer region containing x then has the following canonical form:

$$\begin{array}{l} [x] : 0 \sim fr(x_{a_1}) \sim \cdots fr(x_{a_k}) < 1, \\ int(x_{a_i}) = K_i \end{array} \quad (3)$$

Here $\sim \in \{=, <\}$ and the indices a_i order the timers. The immediate succesor of a timer region under the *succ* operator is denoted **next**$([x])$. In (3), if $0 = fr(x_{a_1})$ then **next**$([x])$ is:

$$\begin{array}{l} [x'] : 0 < fr(x_{a_1}) \sim \cdots fr(x_{a_k}) < 1 \\ int(x_{a_i}) = K_i \end{array} \quad (4)$$

Assuming $fr(x_{a_{k-1}}) < fr(x_{a_k})$ then **next**$([x'])$:

$$\begin{array}{l} \textbf{next}([x']) : 0 = fr(x_{a_k}) \sim \cdots fr(x_{a_{k-1}}) < 1 \\ int(x_{a_k}) = K_k + 1;\ int(x_{a_i}) = K_i \ \text{ for } \ i \neq k \end{array} \quad (5)$$

Two discrete time semantics are given later. In both cases the timer state space is a discrete grid $\mathcal{D}$. Timer values and timestamps (v_i and τ_i in (1)) are restricted to points on this grid. In both cases timer regions are given by (3). We use $[x]_\mathcal{D}$ to denote points on the grid in a given timer region. Initial timer values C_0 are then picked as the largest point in $[C_0]_\mathcal{D}$. The timed and untimed languages are defined as

$\mathcal{L}_{DT} = \{(\sigma, \tau) \mid (\sigma, \tau)$ has a run in discrete time semantics$\}$; $\mathcal{L}_{DU} = \{\sigma | \exists (\sigma, \tau) \in \mathcal{L}_{DT}\}$.
In both cases we show that $\mathcal{L}_{DU} = \mathcal{L}_{CU}$.

0-7803-1968-0/94$4.00©1994 IEEE

3 Discretization 1

The desired result is obtained by proper specification of the *succ* operator. For a TA with K timers we use the discrete grid $\mathcal{D} = \mathcal{Z}^K_{K+1}$ as the state space for the discrete time automaton. The successor of $d \in \mathcal{D}$ under time passage is denoted by $\texttt{tick}(d)$. Given d satisfying (3) $\texttt{tick}(d)$ is defined as follows.[1]

- Rule 1: Pick the timer with lowest fractional value that can be incremented without violating (changing) any of the $\sim$ relations in (3) and increment it by $\frac{1}{K+1}$. If there is more than one timer with equal, lowest fractional value, increment them simultaneously.
- Rule 2: If no such timer exists, i.e., an increment of any timer would violate some relation in (3), examine the first relation $0 \sim fr(x_{a_1})$.
  - If $\sim$ is "=", advance timer a_1 (and all timers with same fractional component as a_1) by $\frac{1}{K+1}$ changing the relation to "<".
  - If it is "<", advance timer a_k (and all timers with equal fractional component as a_k) by $\frac{1}{K+1}$. This sets $fr(x_{a_k}) = 0$ and $int(x_{a_k}) = K_k + 1$.

For an intuitive illustration of time passage, see Figure 1 in which the discrete time timer trajectories have the staircase shape.

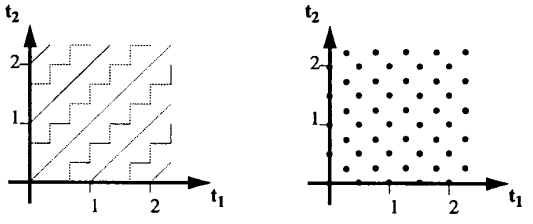

Figure 1: Left: Discretization 1 trajectories; Right: Discretization 2 Grid Points

With these timer semantics, the *succ* operator is defined recursively by successive applications of the tick operator:

- succ: $x \xrightarrow{succ} x$; $x \xrightarrow{succ} y$ provided, $\exists z$ such that, $z = \texttt{tick}(x)$ and $z \xrightarrow{succ} y$
- δ and λ operators for discrete time have similar semantics as in dense time

Note that in this definition *succ* can have zero duration. In particular, two state transitions can take place at the same time. This is not a restriction of the automaton model, but rather a side-effect of the discretization. Since the main goal is the verification of *untimed* event sequeneces the exact time valuations used to generate those sequences are not relevant, as long the time order is preserved.

Our proof is based on the results in [1]. We show that a discrete time trajectory traverses the same sequence of timer regions as its corresponding dense time trajectory. We show that each timing operator preserves this sequence. Since $\gamma \in \Gamma$ is a composition of these operators the result follows. The proofs are omitted.

[1] Based on the largest integer in the timer constraints Φ, we may bound the $\mathcal{D}$ values that we need to consider.

Lemma 1 *If $d \in [v]_{\mathcal{D}}$, $\texttt{tick}(d)$ is either in $[v]_{\mathcal{D}}$, or in $(\texttt{next}([v]))_{\mathcal{D}}$. Furthermore $\exists d' \in (\texttt{next}([v]))_{\mathcal{D}}$ such that $d \xrightarrow{succ} d'$.*

Lemma 2 *If $v \xrightarrow{succ} v'$ and $z \in [v]_{\mathcal{D}}$ then $\exists z' \in [v']_{\mathcal{D}}$ such that $z \xrightarrow{succ} z'$.*

Lemma 3 *If $v \xrightarrow{\delta} v$ for $\delta \in \Phi$ then $\exists z \in [v]_{\mathcal{D}}$, such that $z \xrightarrow{\delta} z$.*

Lemma 4 *If $v \xrightarrow{\lambda} v'$ for $\lambda \in 2^C$ and $z \in [v]_{\mathcal{D}}$ then $\exists z' \in [v']_{\mathcal{D}}$ such that $z \xrightarrow{\lambda} z'$.*

Theorem 1 *The discrete time grid can be used to generate the untimed event language of any timed automaton. In particular, $\mathcal{L}_{DU} = \mathcal{L}_{CU}$.*

4 Discretization 2

The desired result is obtained by proper specification of the reset of a timer under the λ operator. We use a subset of the discrete grid $\mathcal{Z}^K_{\in\mathcal{K}}$. It consists of $\mathcal{Z}^K_K$ together with its translation by the vector $\mathcal{V} = (\frac{1}{2K}, \frac{1}{2K}, \ldots, \frac{1}{2K})$, as in Figure 1. Define $\texttt{tick}(d) = d + \mathcal{V}$. The definition of the *succ* then is similar as in dense time, but restricted to the discrete grid. Note that *succ* can take zero time.

- *succ*: $x \xrightarrow{succ} y$ provided $y = x + l$ where $l \in \mathcal{Z}_{2K}$.
- δ operator for discrete time has similar semantics as dense time.
- λ is defined as succesive resets of the specified timers.

We now define reset.

Let a timer state satisfy (3). The reset of a timer, a_i, results in the timer region $0 = fr(x_{a_i}) \sim fr(x_{a_1}) \sim \ldots \sim fr(x_{a_n}) < 1$ where all fractional timer values of $x_{a_j}, j \neq i$, are picked from the set $\{0, \frac{2}{2K}, \frac{4}{2K}, \ldots, \frac{2K-2}{K}\}$ such that all of the remaining $\sim$ are satisfied. To make the choice unique we pick the largest such possible x_{a_j}.

Note that both in dense and discrete time, the λ operator is equivalent to a sequence of resets on the timers in λ.

The proof is similar to the first discretization. The same sequence of four lemmas are used to show that each timing operator preserves the sequence of timer regions traversed. The lemmas lead to the stated theorem.

Theorem 2 *The discrete time grid can be used to generate the untimed event language of any timed automata. In particular, $\mathcal{L}_{DU} = \mathcal{L}_{CU}$.*

We have shown that proper modification of discrete time semantics allows us to reduce timed automata to discrete time systems. Our results may provide insight into expressiveness, verification and simulation of timed automata.

References

[1] Rajeev Alur, David Dill; Automata For Modeling Real-Time Systems. In *Automata, Languages and Programming: Proceedings of the 17th ICALP*, Lecture Notes in Computer Science 443, pages 322-335. Springer-Verlag, 1990.

[2] T. Henzinger, Z. Manna, A. Pnueli. What good are digital clocks?. In *Automata, Languages and Programming: Proceedings of the 19th ICALP*, W. Kuich editor, Lecture Notes in Computer Science Volume 623. Springer-Verlag, 1992.