Improvements for the Symbolic Verification of Timed Automata*

Rongjie Yan[1,2], Guangyuan Li[1], Wenliang Zhang[1,2], and Yunquan Peng[1,2]

[1] State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences,Beijing,100080,China
[2] Graduate School of the Chinese Academy of Sciences,Beijing,100039,China
{yrj,ligy}@ios.ac.cn

Abstract. Based on the equivalence relation for location based reachability between continuous and integer semantics of closed timed automata, Beyer et al. have implemented the verifier Rabbit, with the uniform representation of reachable configurations. However, the growth of maximal constant of clock variables will decline the performance of Rabbit. The paper proposes an improved symbolic method, using binary decision diagrams (BDDs) to store the symbolic representation of discretized states, for the verification of timed systems. Compared with Rabbit, experiments demonstrate that besides the memory reduction, our implementation is also less sensitive to the size of clock domain.

Keywords: verification, timed systems, symbolic method, BDD.

1 Introduction

Formal verification is one of the effective methods to ensure the correctness of real-time systems. Timed automata (TAs) [1] provide a formal framework for the automatic analysis and verification of real-time systems, and in the past few years several tools for the model checking of TAs have been developed and used, including Uppaal [13], Kronos [8], Red [16], Rabbit [6] and FPTA [17], which have implemented the computation for the set of all reachable configurations by reachability analysis. However, the exploding increase of time consumption for the computation and memory consumption for the representation of the reachable configurations is still a main problem.

Within the model checking community, many works were based on symbolic representations of the state space. The *region equivalence* of [1] is the precursor of the symbolic methods in which the state space is covered using regions with the same integer parts of clock values and the ordering of fractional parts. Currently, most of real-time verifiers apply abstractions based on *zones* (the constraint sets) in order to be coarser. Difference bound matrices (DBMs) [4] are a common data structure to describe zones. However, this structure cannot

* Supported by 973 Program of China under Grant No. 2002cb312200; and the National Natural Science Foundation of China under Grant Nos. 60673051, 60421001.

J. Derrick and J. Vain (Eds.): FORTE 2007, LNCS 4574, pp. 196–210, 2007.

unify the representation of configurations which consist of locations and clock valuations.

Besides DBMs, clock difference diagrams (CDDs) [3] and their variants [16,15] were used to combine the representation of locations and clock valuations in zones. Their common disadvantage is that the lack of a unique canonical representation may hinder the containment relation detection.

The work in [9] introduced the BDD representation of reachable configurations based on the methods of time discretization [10]. The work in [2] proposed that closed timed automata (CTAs), whose clock constraints only contain $\geq, \leq$ relations, can just consider integer clock valuations for the reachability analysis. Based on the observation in [2], the work in [5] implemented BDD-based reachability analysis, which formally defined the integer semantics of closed automata and proved the equivalence between integer and continuous semantics for location based reachability. All these BDD-based verifiers share the same problem of BDD's: they are sensitive to the size of clock domain.

Based on the work of [5], we introduce symbolic structures for the representation of reachable configurations, in the integer semantics of closed timed automata, which is similar to the work of [17]. To reduce the memory consumption, BDD is applied to store the reachable symbolic sets. The combination not only reduces the sensitivity to the scale of clock constants, but also unifies the representation of locations and clock valuations.

The paper is organized as follows. In section 2, we briefly recall the definition of TAs, CTAs and their semantics. In section 3, we present the new symbolic data structures and the reachability analysis algorithm for the integer semantics. In section 4, we demonstrate the performance of our prototype implementation. Section 5 concludes and discusses future work.

2 Preliminaries

The section introduces the definition of TAs and their continuous and integer semantics.

A timed automaton (TA), proposed by Alur and Dill [1], is a finite state automaton extended with a finite set of real-valued clock variables.

Definition 1. *(Syntax of Timed Automata).*
Let X be a finite set of clocks, and $C(X)$ be the clock constraint set over X, given by the syntax:

$$\phi \ ::= \ (x \sim c) \mid \phi_1 \wedge \phi_2 \mid true$$

where $x \in X$, $\sim \in \{<, \leq, >, \geq\}$ and $c \in \mathbb{N}^+$ ($\mathbb{N}^+$ is the set of non-negative integers).

A timed automaton *over X is a tuple $A = \langle L, l_0, \Sigma, X, I, E \rangle$, where*

- *L is a finite set of locations, and $l_0 \in L$ is the initial location,*
- *I is a mapping that labels each location $l \in L$ with some constraint in $C(X)$,and $I(l)$ is called the invariant of l,*

- Σ *is a finite set of synchronization labels, and*
- $E \subseteq L \times C(X) \times \Sigma \times 2^X \times L$ *is the set of transitions.*

A transition $(l, g, \sigma, Y, l') \in E$ means that one can move from the location l to l' through a transition labelled with $\sigma \in \Sigma$. Moreover, g the guard must be satisfied by the current clock values, and all the clocks in Y ($Y \subseteq X$) are reset to 0.

Closed timed automata [2] restrict the clock constraints. The restricted constraints ϕ over X is:

$$\phi ::= x \leq c | x \geq c | \phi_1 \wedge \phi_2,$$

where $x \in X$, and $c \in \mathbb{N}^+$.

2.1 Continuous Semantics of TA

In continuous semantics, clock variables have non-negative real valuations. A clock valuation is a function $\mu : X \mapsto \mathbb{R}^+$, where $\mathbb{R}^+$ is the set of non-negative reals. μ_X denotes the set of all clock valuations over X. For $t \in \mathbb{R}^+$, $\mu + t$ denotes the clock valuation such that $\mu(x + t) = \mu(x) + t$, for all $x \in X$. For $Y \subseteq X$, $\mu[Y := 0]$ denotes the clock valuation such that $\mu[Y := 0](x) = 0$, for all $x \in Y$ and otherwise $\mu[Y := 0](x) = \mu(x)$. μ satisfies a constraint $\phi \in C(X)$, denoted by $\mu \models \phi$, if ϕ evaluates to *true* under the assignment given by μ.

The continuous semantics of a timed automaton $A = \langle L, l_0, \Sigma, X, I, E \rangle$ over X is defined as a transition system $\langle S, s_0, \Sigma \cup \mathbb{R}^+, \rightarrow \rangle$, where $S = L \times \mu_X$; $s_0 = (l_0, \mu_0)$ is the initial state where $\mu_0(x) = 0$ for all $x \in X$; and the transition relation $\rightarrow$ comprises two kinds of moves:

- delay transition: $(l, \mu) \xrightarrow{\delta} (l, \mu + \delta)$, if $\delta \in \mathbb{R}^+$ and $\mu \models I(l)$ and $\mu + \delta \models I(l)$;
- discrete transition: $(l, \mu) \xrightarrow{\sigma} (l', \mu[Y := 0])$, if $(l, g, \sigma, Y, l') \in E$ and $\mu \models g$ and $\mu[Y := 0] \models I(l')$.

Let A be a TA. For a state $s_k = (l, \mu)$ where $l \in L, \mu \in \mu_X$. If there is a finite state sequence such that $s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{k-1}} s_k$, then s_k is called reachable and l the reachable location in the continuous semantics of A, where $\alpha_i \in \Sigma \cup \mathbb{R}^+,\ 0 \leq i < k$.

2.2 Integer Semantics of TA

The differences between integer and continuous semantics are the definitions of clock valuations and transition relations. In integer semantics, clock variables have integer valuations. A clock valuation is a function $\nu : X \mapsto \mathbb{N}^+$, where $\mathbb{N}^+$ is the set of non-negative integers. ν_X denotes the set of all clock valuations over X.

The integer semantics of a timed automaton [5] $A = \langle L, l_0, \Sigma, X, I, E \rangle$ is defined as a transition system $\langle S, s_0, \Sigma \cup \mathbb{N}^+, \rightarrow_{\mathcal{I}} \rangle$, where $S = L \times \nu_X$, $s_0 = (l_0, \nu_0)$ is the initial state where $\nu_0(x) = 0$ for all $x \in X$, and the transition relation $\rightarrow_{\mathcal{I}}$ comprises two kinds of moves:

- delay transition: $(l,\nu) \xrightarrow{\delta} (l,\nu \oplus \delta)$, if $\delta \in \mathbb{N}^+$ and $\nu \models I(l)$, $\nu \oplus \delta \models I(l)$;
- discrete transition: $(l,\nu) \xrightarrow{\sigma} (l',\nu[Y := 0])$, if $(l,g,\sigma,Y,l') \in E$ and $\nu \models g$, $\nu[Y := 0] \models I(l')$.

where $(\nu \oplus \delta)(x) = \min\{\nu(x) + \delta,\ c_{\mathcal{A}}(x) + 1\}$, $c_{\mathcal{A}}(x)$ is the maximal constant compared with x in the clock constraints of A.

Let A be a TA. For a state $s_k = (l,\nu)$ where $l \in L, \nu \in \nu_X$. If there is a finite state sequence such that $s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{k-1}} s_k$, then s_k is called reachable and l the reachable location in the integer semantics of A, where $\alpha_i \in \Sigma \cup \mathbb{N}^+,\ 0 \le i < k$.

The work in [5] proved the equivalence relation for the set of reachable locations between integer and continuous semantics of CTAs, which formed the basis of BDD-based reachability analysis.

3 Reachability Analysis for CTAs

In the integer semantics, the number of reachable configurations and the time consumption grow greatly with the increasing size of clock domain. Though BDD can reduce the memory consumption by data sharing, dealing with such enormous reachable sets will slow down the verification process.

Based on the symbolic representation for integer clock valuations in [17], we apply the symbolic method to record the reachable configurations of CTAs during the verification process. Meanwhile, we use BDD to record the symbolic sets to increase the data sharing and reduce the memory consumption.

3.1 Delay Sequence

Reachability is one of the most common properties being checked by verifiers. There are two kinds of search strategies for reachability analysis during state space exploration: forward and backward search. Currently our tool uses the forward search technique.

The forward analysis of the reachable configurations starts from the initial state (l_0, v_0). Whenever allowed by the invariant of l_0, time delays can form the sequence $(l_0, v_0 \oplus 0) \xrightarrow{1} (l_0, v_0 \oplus 1) \xrightarrow{1} \cdots$, where $v_0 \oplus i \models I(l_0)$. For example, given $A = \langle L, l_0, \Sigma, X, I, E\rangle$, let $l_0 \in L$ be the initial location with the invariant $x \le 10^6$, where $x \in X$. Then there may be a sequence: $(l_0, 0) \xrightarrow{1} (l_0, 1) \xrightarrow{1} \cdots \xrightarrow{1} (l_0, 10^6)$. Even with BDD representation for the set of states in this sequence, the frequent operations with the increasing number of reachable configurations are burdensome. To relieve this problem, here we introduce a symbolic representation for this kind of sequence.

Definition 2. *(Symbolic Representation of Delay Sequence)*
Given location l and clock valuation v, let $< l, v >$ denote the set of states $\{(l,v') | v' = v \oplus i,$ where $i \ge 0,$ and $v' \models I(l)\}$. Based on the maximal constant abstraction, for every $x \in X$, all the clock valuations greater than $c_{\mathcal{A}}(x)+1$ are

treated as $c_{\mathcal{A}}(x)+1$. *Therefore, though time can progress infinitely, the number of states in the delay sequence is finite.*

Therefore, a delay sequence (DS) generated by delay transitions from the state $s=(l,v)$ *can be denoted by* $<l,v>$. *And the number of states in* $<l,v>$ *can be determined by* $I(l)$ *and clock valuation* v.

Let $A=\langle L,l_0,\Sigma,X,I,E\rangle$ be a timed automaton. For a state $s=(l,v)$ and a transition $e=(l,g,\sigma,Y,l')\in E$, where $l\in L$ and $v\in\nu_X$, $post(s,e)$ denotes the set of states $\{<l',v'>$ |if $\exists i\in\mathbb{N}^+$ such that $v\oplus i\models I(l)\wedge g,\ v'=v\oplus i[Y:=0]$ and $v'\models I(l')\}$.

Given the symbolic representation of DS, the symbolic semantics can be defined as follows.

Definition 3. *(Symbolic Semantics)*
Let $A=\langle L,l_0,\Sigma,X,I,E\rangle$ *be a timed automaton. The symbolic semantics of* A *is based on the transition system* $\langle S,s_0,\rightsquigarrow\rangle$, *where* $S=L\times\nu_X,\ s_0=<l_0,v_0>$, *and* $\rightsquigarrow$ *is defined by the following rule:*
$<l,v>\rightsquigarrow<l',v'>$, *if there exist a transition* $(l,g,\sigma,Y,l')\in E$ *and an* $i\in\mathbb{N}^+$, *such that* $v\oplus i\models I(l)\wedge g,\ v'=v\oplus i[Y:=0]$ *and* $v'\models I(l')$.

Given a time automaton $A=\langle L,l_0,\Sigma,X,I,E\rangle$, and a state (l,v). l is reachable in the integer semantics of A, iff it is reachable in the symbolic semantics $\langle S,s_0,\rightsquigarrow\rangle$.

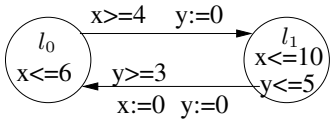

Fig. 1. A simple example

Example 1. The simple timed automaton in Figure 1 is to illustrate the application of DS during the reachability analysis. Every state is denoted by $(l_i,(v(x),v(y)))$, where $(v(x),v(y))$ are two clock valuations of the timed automaton.

One of the runs in the example is: $(l_0,(0,0))\xrightarrow{1}(l_0,(1,1))\xrightarrow{1}(l_0,(2,2))\xrightarrow{1}(l_0,(3,3))\xrightarrow{1}(l_0,(4,4))\rightarrow(l_1,(4,0))\xrightarrow{1}(l_1,(5,1))\cdots$. We list the unfolded state space in Figure 2. The state sequence generated from $(l_0,(0,0))$ by delay transitions can be represented by $<l_0,(0,0)>$, and the sequence generated from $(l_1,(4,0))$ by delay transitions can be represented by $<l_1,(4,0)>$. Therefore, with DS representation, the size of state space can be reduced. Figure 3 shows the reduced state space.

3.2 Series of Delay Sequences

In a delay sequence, when some states satisfy the guard of a transition, the corresponding discrete transition can be taken, leading to the new states. From these

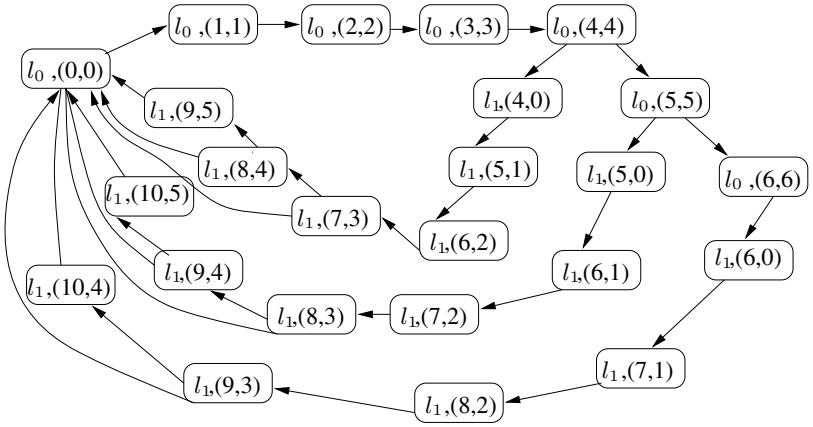

Fig. 2. Unfolded state space of the example

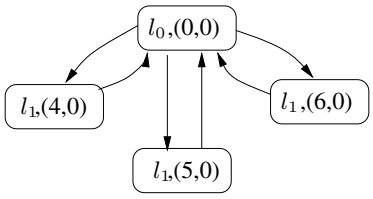

Fig. 3. State space represented by DS

new states, the execution of delay or discrete transitions will be continued. For instance, in the example of Figure 1, some states in delay sequence $< l_0, (0,0) >$ can trigger the discrete transition from l_0 to l_1. Then the corresponding successor states are $(l_1, (4,0)), (l_1, (5,0)), (l_1, (6,0))$. After the discrete transition, only valuations of reset clocks are different from their precursors. If we ignore the reset clocks, we will find that other clock valuations in these successors still obey the rule of "$\oplus$" operation.

Then given a state generated by the discrete transition, we can compute all other new successors from the states in the same delay sequence. Let v_r be the clock valuation after a discrete transition, where Y is the reset clock set. The clock valuaton of the ith state from v_r is $v_{ir} = v_r \circledast i$, where $(v_r \circledast i)(x) = \begin{cases} v_r(x) + i & x \notin Y \\ 0 & x \in Y \end{cases}$.

Based on this observation, we can define a coarser data structure, which comprises more than one DS.

Definition 4. *(Series of Delay Sequences)*
Let $((l, v), k, Y)$ be the symbolic representation for the set of states $\{< l, v' > | v' = v \circledast i, 0 \leq i < k\}$. We call this representation the series of delay sequence (SDS). Let $s_0 = (l, v)$ be the so-called start state, then the SDS is denoted by (s_0, k, Y). (s_0, k, Y) is denoted by $(s_0, 1, \emptyset)$ if $Y = \emptyset$.

Then in Example 1, delay sequences $< l_1, (4,0) >$, $< l_1, (5,0) >$, $< l_1, (6,0) >$ computed from $< l_0, (0,0) >$ by the discrete transition can be represented by $((l_1, (4,0)), 3, \{y\})$. The state space is further reduced.

3.3 Reachability Analysis

During the forward search of the reachable configurations in integer semantics, we use DS to compute the successors and record the set of visited states. Given a DS t, the process for successor computation from t is: the delay sequence from t can trigger the discrete transition e when some states in the sequence satisfy its guard. Then the set of new configurations is computed. In other words, if t is not in P the set of visited configurations, then by the discrete transition e, the delay sequence from t can generate the set of configurations $T = post(s, e)$. And the new successors will be added to the waiting list W for the computation loop. The verification will stop when W is empty or the property is satisfied.

Because the structure of a DS is similar to that of a discrete state, to save the memory consumption during the verification process, we use BDD to represent the set of reachable DSs. The generalized algorithm for reachable analysis is as follows:

1: **Reachability()**
2: $W = \{< l_0, v_0 >\}$;
3: **while** $W \neq \emptyset$ **do**
4: get $s =< l, v >$ from W;
5: **if** $s \in P$ **then**
6: *continue*
7: **end if**
8: **for all** $e \in \{(l, g, \sigma, Y, l')\}$ **do**
9: $T = post(s, e)$;
10: **if** $\exists t \in T, t \models \phi$ **then**
11: return *true*
12: **end if**
13: **if** $T \nsubseteq W$ **then**
14: $W = W \cup T$
15: **end if**
16: $P = P \cup \{s\}$;
17: **end for**
18: **end while**
19: return *false*

3.4 The Application of SDS

Because the result of $post(s, e)$ is a set of interrelated DSs, we can use SDS to represent the set of DSs. Then W can be organized as the list of SDSs to reduce the occupied memory during the verification process. Meanwhile, time

consumption for computing new reachable configurations from DSs in the same SDS can be saved for their interrelation.

To explain how to compute successors for a SDS, we firstly show the corresponding computation for a DS. Then we discuss the relation between two SDSs with the same locations. Finally we present the SDS-based reachability analysis algorithm.

3.4.1 Successor Computation for DS

Given a timed automaton $A = \langle L, l_0, \Sigma, X, I, E\rangle$, a state (l, v) and $e = (l, g, \sigma, Z, l') \in E$, where $l \in L$ and $v \in \nu_X$. To compute the successors, we need to consider the constraints involving $I(l)$ and g. That is, to trigger the discrete transition, the states should satisfy the constraint $I(l) \wedge g$.

For convenience, given a clock valuation $v \in \nu_X$, let $\theta_X^v = \max\{\ c_{\mathcal{A}}(x) - v(x) | x \in X\}$. Given a constraint ϕ, we define

- X_ϕ is the set of all clock variables occuring in ϕ.
- $\phi_l = \bigwedge\{x \leq c | x \leq c \text{ is in } \phi\}$.
- $\phi_g = \bigwedge\{x \geq c | x \geq c \text{ is in } \phi\}$.
- c_ϕ is the maximal constant occuring in ϕ.
- $c_\phi(x)$ be the maximal constant compared with x in ϕ.

In the following computation, for constraint $\varphi = I(l) \wedge g$, let

$$m = \max\{\{c_{\varphi_g}(x) - v(x) | x \in X_{\varphi_g}\}, 0\} \tag{1}$$

$$n = \min\{\{c_{\varphi_l}(x) - v(x) | x \in X_{\varphi_l}\}, \theta_X^v\} \tag{2}$$

Then in the delay sequence $< l, v >$, set of states $\{(l, v \oplus j) | j \in [m, n]\}$ can take the discrete transition $e = (l, g, \sigma, Z, l')$.

Example 2. For the timed automaton in Figure 1, given the initial state $(l_0, (0, 0))$ and the discrete transition from l_0 to l_1. Firstly, $\theta_X^v = 10$ and $\varphi = x \geq 4 \wedge x \leq 6$. According to the definition, we get that $\varphi_l = x \leq 6$, $c_{\varphi_l}(x) = 6$, $\varphi_g = x \geq 4$, and $c_{\varphi_g}(x) = 4$. Then $m = 4$ and $n = 6$, the set $\{(l_0, (4, 4)),\ (l_0, (5, 5)),\ (l_0, (6, 6))\}$ in $< l_0, (0, 0) >$ can take the discrete transition from l_0 to l_1.

Therefore, to get successors from a DS, we have to consider the computation and judgement between clock valuations and constraints.

3.4.2 Successor Computation for SDS

If all successors of DSs in a SDS can be computed according to the successors of one DS, the effort can be saved by avoiding the repeated computation between clock valuations and constraints.

Given a SDS $d = ((l, v), k, Y)$ and $e = (l, g, \sigma, Z, l')$, we have observed that $v(x) \oplus i = v(x) \circledast i$ for all $x \in X - Y$. So some states in the ith delay sequence $< l, v \circledast i >$ and the delay sequence $< l, v >$ may have same valuations except for the clocks in Y.

With this observation, we discuss the corresponding process for computing new configurations. We firstly determine the set of states that can trigger the discrete transition for the start state (l, v).

For the constraint φ and SDS d, let

$$a_1 = \max\{\{c_{\varphi_g}(x) - v(x) | x \in Y \cap X_{\varphi_g}\}, 0\} \tag{3}$$

$$b_1 = \min\{\{c_{\varphi_l}(x) - v(x) | x \in Y \cap X_{\varphi_l}\}, \theta_Y^v\} \tag{4}$$

$$a_2 = \max\{\{c_{\varphi_g}(x) - v(x) | x \in (X - Y) \cap X_{\varphi_g}\}, 0\} \tag{5}$$

$$b_2 = \min\{\{c_{\varphi_l}(x) - v(x) | x \in (X - Y) \cap X_{\varphi_l}\}, \theta_{(X-Y)}^v\} \tag{6}$$

Then $m = \max\{a_1, a_2\}$, and

$$n = \begin{cases} \max\{b_1, b_2\} & \text{if } X_{\varphi_l} = \emptyset \\ b_2 & \text{if } (X_{\varphi_l} \cap Y) = \emptyset \\ b_1 & \text{if } (X_{\varphi_l} \cap X - Y) = \emptyset \\ \min\{b_1, b_2\} & \text{if } (X_{\varphi_l} \cap Y) \neq \emptyset \wedge (X_{\varphi_l} \cap X - Y) \neq \emptyset \end{cases}.$$

Set of states $\{(l, v \oplus j) | j \in [m, n]\}$ in the delay sequence $< l, v >$ can trigger the discrete transition $e = (l, g, \sigma, Z, l')$.

Example 3. Now we use SDS $((l_1, (4, 0)), 3, \{y\})$ and the transition from l_1 to l_0 in Figure 1 as the example. For the start state $(l_1, (4, 0))$ and guard $y \geq 3$ in the transition, we get that $\varphi = x \leq 10 \wedge y \leq 5 \wedge y \geq 3$. According to the Equation 3 $\sim$ 6, $a_1 = 3$, $b_1 = 5$, $a_2 = 0$, and $b_2 = 6$ respectively. Because $X_{\varphi_l} = \{x, y\}$ and $Y = \{y\}$, neither $X_{\varphi_l} \cap Y$ nor $X_{\varphi_l} \cap X - Y$ is empty. Therefore $m = 3,\ n = 5$, the set of states $\{(l_1, (4, 0) \oplus j) | j \in [3, 5]\}$ in the delay sequence $< l_1, (4, 0) >$ can take the discrete transition.

After the computation for the successors of the start state in SDS, we can get the set of successors from other DSs in the same SDS according to the feature of SDS.

For the delay sequence of $< l, v \circledast i >$ where $0 \leq i < k$, $m = \max\{a_1, a_2 - i\}$, and

$$n = \begin{cases} \max\{b_1, b_2\} & \text{if } X_{\varphi_l} = \emptyset \\ \max\{b_2 - i, 0\} & \text{if } (X_{\varphi_l} \cap Y) = \emptyset \\ b_1 & \text{if } (X_{\varphi_l} \cap X - Y) = \emptyset \\ \min\{b_1, \max\{b_2 - i, 0\}\} & \text{if } (X_{\varphi_l} \cap Y) \neq \emptyset \wedge (X_{\varphi_l} \cap X - Y) \neq \emptyset \end{cases},$$

states that can trigger the discrete transition e are the set $\{(l, (v \circledast i) \oplus j) | j \in [m, n]\}$.

Example 4. For SDS $((l_1, (4, 0)), 3, \{y\})$, we have obtained that $a_1 = 3,\ b_1 = 5$, $a_2 = 0$, and $b_2 = 6$. Then

1. For the delay sequence $< l_1, (4,0) \circledast 1 >$, set of states $\{(l_1, (5,0) \oplus j) | j \in [3,5]\}$ can trigger the discrete transition.
2. For the delay sequence $< l_1, (4,0) \circledast 2 >$, set of states $\{(l_1, (6,0) \oplus j) | j \in [3,4]\}$ can trigger the discrete transition.

3.4.3 SDS-Based Reachability Analysis

The following is the reachability analysis algorithm by the application of SDS.

```
1: ReachabilitySDS
2: SDS d';
3: stack of SDS W;
4: W.push((l_0, v_0), 1, ∅);
5: while W ≠ ∅ do
6:    get d = (s_0, k, Y) from W;
7:    for all e = (l, g, σ, Y', l') enabled at d do
8:       SuccessorN(a_1, b_1, a_2, b_2, d, e);
9:       for i = 0; i < k; i++ do
10:         s_i = s_0 ⊛ i;
11:         if s_i ∈ P then
12:            continue
13:         end if
14:         distance = getdistance(i, φ, a_1, b_1, a_2, b_2);
15:         if distance < 0 then
16:            break
17:         end if
18:         s' = (l', (s_i.v ⊕ max{a_1, a_2 − i})[Y' := 0]);
19:         d' = (s', distance + 1, Y');
20:         if ∃s ∈ d', s ⊨ φ then
21:            return true
22:         end if
23:         if d' ∉ W then
24:            W.push(d')
25:         end if
26:         P = P ∪ {s_i};
27:      end for
28:   end for
29: end while
30: return false
```

Line 8 *SuccessorN* computes the related ranges for the start state according to the certain discrete transition. Line 9-27 compute all the successors of d, and Line 13 *getdistance* is to get the number of states in every DS which is capable of taking the discrete transition (the number of successors for every DS). Then Line 18, 19 generate a new set of DSs.

3.5 Inclusion Relation of SDS

When we get a new set of reachable configurations, we need to judge its relation with those SDSs in W to avoid the repeated computation and ensure the termination of checking. The relation between the new set d' and $d \in W$ is:

- *equivalence*, if their start states, the number of DSs, and the set of reset clocks are equal, which is a special case of inclusion. Or
- *intersection*, if two sets of reset clocks are equal, and there exists $0 \leq i < k,\ 0 \leq j < k'$, such that all the valuations of $s_0 \circledast i$ and $s'_0 \circledast j$ are the same state. Or
- *irrelevance*, neither with equivalence nor intersection relation.

The algorithm for judging SDS relations is as follows. The idea of the algorithm is: firstly we should judge whether the differences between two clock valuations are the same. If some clock differences are different from others, there is no equivalence or intersection relation. If all clock valuations except for reset ones have the same difference, two SDSs may intersect, or one is a subset of the other.

1: **SDSRelation**(d, d')
2: select an x which $x \notin Y \wedge s_0(x) < c_{\mathcal{A}}(x) \wedge s'_0(x) < c_{\mathcal{A}}(x)$;
3: diff=$s_0(x) - s'_0(x)$;
4: **for all** $x \in X - Y$ **do**
5: **if** $s_0(x) - s'_0(x) \neq$ diff **then**
6: return *irrelevance*
7: **end if**
8: **end for**
9: **if** diff $\leq 0 \wedge$ diff $+ k \geq k'$ **then**
10: return $d \supseteq d'$
11: **else**
12: **if** diff $\geq 0 \wedge$ diff $+ k \leq k'$ **then**
13: return $d \subseteq d'$
14: **end if**
15: **end if**
16: return *intersection*

---

4 Experiments

Based on the symbolic data structure, we have implemented a prototype to support the verification of real-time systems with multi-processes, synchronizations, and broadcasts. The tool is available at http://lcs.ios.ac.cn/∼ligy/tools/. We compare the experiment results with those of Rabbit. All experiments were performed on a 2.6GHz Pentium 4 with 512MB of memory. And experiments were limited to 30 minutes of CPU time.

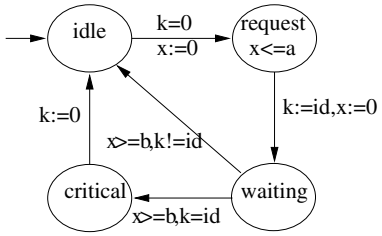

Fig. 4. Fischer's mutual exclusive protocol

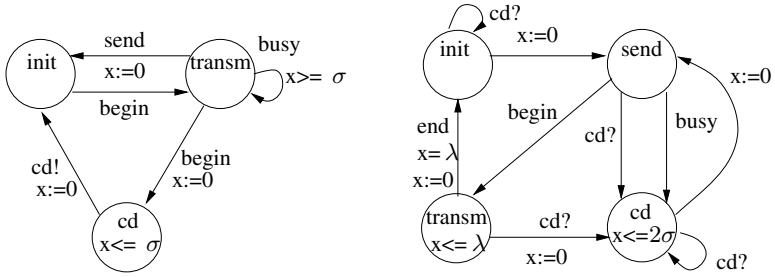

Fig. 5. CSMA/CD protocol

We use Fischer's mutual exclusive protocol(f) [12] (see Figure 4), CSMA/CD protocol(c) [7] (see Figure 5) and two industrial case studies (Gear Controller [14], and an Audio/Video Protocol [11]) as the examples. In the following tables, we list the time consumption(t) in seconds and the number of reachable configurations (*state* for Rabbit, and DS for our implementation). BDD is the number of nodes in the BDD representation for the whole reachable configuration when the verification finishes, which is in direct proportion to the memory consumption. "-" indicates that the result is unavailable.

To compare the sensitivity of tools to clock constants, we demonstrate experiment results for different valuations of a and b in Fischer's mutual exclusive protocol, λ and σ in CSMA/CD protocol. Here the complete state space was generated.

As we know, the greater the maximal constant, the more the number of reachable configurations will be. Table 1 and 2 list results of Rabbit and our prototype under different valuations for two protocols. Comparing results in two tables, the number of reachable configurations in Rabbit and the prototype increases rapidly, as well as the time consumption. However, w.r.t. the finial number of BDD nodes and the number of reachable configurations, the increase of our prototype are quite less than that of Rabbit. For both Fischer's protocol and CSMA/CD protocol, our prototype scales better with the growth of constants.

For the industrial examples, the complete state space of Gear Controller was explored in 177.22 seconds with 64872 BDD nodes recording all the reachable configurations; and the complete state space of Audio/Video protocol was

Table 1. Rabbit's results for Fischer's (f) and CSMA/CD (c)

No.	a=2,b=4			a=4,b=8			a=8,b=16			a=16,b=32		
	t	state	BDD	t	state	BDD	t	state	BDD	t	state	BDD
f2	0	193	133	0	467	216	0	1399	374	0	4799	684
f3	0	1893	605	0	7095	1318	0	36939	3385	0	234867	9998
f4	0	17577	1956	0	102291	5531	1	922479	19741	4	1.08322e+07	85485
f5	0	158449	4720	0	1.43358e+06	16218	3	2.2305e+07	75043	29	4.8236e+08	450942
f6	1	1.40518e+06	8751	2	1.97584e+07	34284	8	5.2812e+08	191567	-	-	-
f7	0	1.23492e+07	13821	2	2.69305e+08	58725	22	1.23136e+10	371250	-	-	-
f8	1	1.07952e+08	19897	5	3.63936e+09	88780	-	-	-	-	-	-
f9	1	9.40233e+08	26960	10	4.88237e+10	124248	-	-	-	-	-	-
f10	2	8.16454e+09	35014	17	6.50652e+11	165133	-	-	-	-	-	-
	$\lambda = 4, \sigma = 1$			$\lambda = 8, \sigma = 2$			$\lambda = 16, \sigma = 4$			$\lambda = 32, \sigma = 8$		
c2	0	157	213	0	358	459	0	982	994	1	3118	2195
c3	0	1446	620	0	4609	1614	2	19569	4308	2	105025	12579
c4	2	11225	1237	1	49645	3636	2	325631	11369	10	2.9609e+06	44015
c5	1	84140	2035	1	502835	7792	8	5.02002e+06	32181	-	-	-
c6	0	594174	2851	3	4.75103e+06	13836	19	7.22529e+07	70441	-	-	-
c7	1	4.01893e+06	3667	5	4.27544e+07	20989	-	-	-	-	-	-
c8	2	2.63267e+07	4483	9	3.71571e+08	28482	-	-	-	-	-	-

Table 2. Our results for Fischer's (f) and CSMA/CD (c)

No.	a=2,b=4			a=4,b=8			a=8,b=16			a=16,b=32		
	t	DS	BDD	t	DS	BDD	t	DS	BDD	t	DS	BDD
f2	0.05	22	62	0.05	22	72	0.05	22	82	0.05	22	92
f3	0.08	107	151	0.16	119	181	0.09	143	211	0.16	191	241
f4	0.14	476	273	0.19	588	333	0.33	812	393	0.80	1260	453
f5	0.44	1970	416	0.78	2620	512	1.95	3920	608	5.97	6520	704
f6	1.97	7679	583	4.13	10717	721	11.63	16793	859	38.59	28945	997
f7	9.95	28551	772	22.13	41123	958	63.48	66267	1144	222.56	116555	1330
f8	43.75	102382	987	102.95	150610	1227	313.00	247066	1467	1063.27	439978	1707
f9	190.83	357176	1222	466.16	533102	1522	1398.83	884954	1822	-	-	-
f10	932.52	1220153	1481	-	1839819	1847	-	3079151	2213	-	-	-
	$\lambda = 4, \sigma = 1$			$\lambda = 8, \sigma = 2$			$\lambda = 16, \sigma = 4$			$\lambda = 32, \sigma = 8$		
c2	0.08	31	86	0.08	46	106	0.09	78	149	0.09	142	226
c3	0.16	202	180	0.17	387	311	0.23	885	551	0.47	2385	2141
c4	0.34	1038	344	0.61	2123	787	2.25	5507	1965	18.06	16643	5730
c5	1.70	4479	619	3.84	9814	1578	31.59	28672	4381	659.59	101732	14177
c6	8.67	17786	910	28.66	41580	2441	455.17	139100	7046	-	634740	23636
c7	37.45	67046	1202	183.61	166701	3308	-	-	-	-	-	-
c8	179.45	243741	1494	1243.05	641407	4175	-	-	-	-	-	-

unfolded in 917.67 seconds with 2351 BDD nodes. However, Rabbit failed in the limited memory. The performance of our prototype is dramatic compared with Rabbit.

Therefore, our tool's sensitivity to constant valuations is lower than that of Rabbit. The reason is that BDD representation for DS is coarser than that for explicit states in integer semantics.

5 Conclusions and Further Work

In this paper we propose a new symbolic structure for the discrete states in the integer semantics of closed timed automata for the reachability analysis. Concluded from the experiment results, our structure is better than the pure BDD representation for explicit configurations w.r.t. the influence of the magnitude of clock constants. And the memory consumption is greatly reduced, benefited from the data sharing ability of BDD.

However, the prototype does not use BDD to represent the transition relations yet, which results in the transformation from DS to BDD frequently. The transformation and judgement between DS and BDD waste lots of time. So our time consumption is higher than that of Rabbit. For further work, we need to investigate the combination of BDD representation for transition relations and our symbolic data structure to improve the performance of our prototype.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
2. Asarin, E., Maler, O., Pnueli, A.: On discretization of delays in timed automata and digital circuits. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 470–484. Springer, Heidelberg (1998)
3. Behrmann, G., Larsen, K.G., Pearson, J., Weise, C., Yi, W.: Efficient timed reachability analysis using clock difference diagrams. In: Computer Aided Verification, pp. 341–353 (1999)
4. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton (1957)
5. Beyer, D.: Improvements in BDD-based reachability analysis of timed automata. In: Oliveira, J.N., Zave, P. (eds.) FME 2001. LNCS, vol. 2021, pp. 318–343. Springer, Heidelberg (2001)
6. Beyer, D., Lewerentz, C., Noack, A.: Rabbit: A tool for BDD-based verification of real-time systems. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 122–125. Springer, Heidelberg (2003)
7. Beyer, D., Noack, A.: Can decision diagrams overcome state space explosion in real-time verification? In: König, H., Heiner, M., Wolisz, A. (eds.) FORTE 2003. LNCS, vol. 2767, pp. 193–208. Springer, Heidelberg (2003)
8. Bozga, M., Daws, C., Maler, O., Olivero, A., Tripakis, S., Yovine, S.: Kronos: A model-checking tool for real-time systems. In: Hu, A.J., Vardi, M.Y. (eds.) CAV 1998. LNCS, vol. 1427, pp. 546–550. Springer, Heidelberg (1998)
9. Asarin, E., Bozga, M., Kerbrat, A., Maler, O., Pnueli, A., Rasse, A.: Data structures for the verification of timed automata. In: Maler, O. (ed.) Hybrid and Real-Time Systems, Grenoble, France. LNCS, pp. 346–360. Springer Verlag, Heidelberg (1997)
10. Gollü, A., Puri, A., Varaiya, P.: Discetization of timed automata. In: Proceedings of the 33rd IEEE conferene on decision and control, pp. 957–958 (1994)
11. Havelund, K., Skou, A., Larsen, K.G., Lund, K.: Formal modeling and analysis of an audio/video protocol: An industrial case study using UPPAAL. In: Proc. of the 18th IEEE Real-Time Systems Symposium, pp. 2–13. IEEE Computer Society Press, Los Alamitos (1997)

12. Lamport, L.: A fast mutual exclusion algorithm. ACM Trans. Comput. Syst. 5(1), 1–11 (1987)
13. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a nutshell. International Journal on Software Tools for Technology Transfer 1(1-2), 134–152 (1997)
14. Lindahl, M., Pettersson, P., Yi, W.: Formal design and analysis of a gear controller. In: Steffen, B. (ed.) ETAPS 1998 and TACAS 1998. LNCS, vol. 1384, pp. 281–297. Springer, Heidelberg (1998)
15. Møller, J., Lichtenberg, J., Andersen, H.R., Hulgaard, H.: Difference decision diagrams. In: Flum, J., Rodríguez-Artalejo, M. (eds.) CSL 1999. LNCS, vol. 1683, pp. 111–125. Springer, Heidelberg (1999)
16. Wang, F.: Efficient verification of timed automata with bdd-like data-structures. In: VMCAI 2003. Proceedings of the 4th International Conference on Verification, Model Checking, and Abstract Interpretation, London, UK, pp. 189–205. Springer-Verlag, Heidelberg (2003)
17. Yan, R., Li, G., Tang, Z.: Symbolic model checking of finite precision timed automata. In: Van Hung, D., Wirsing, M. (eds.) ICTAC 2005. LNCS, vol. 3722, pp. 272–287. Springer, Heidelberg (2005)