数理解析研究所講究録
996巻 1997年 222-242

The symbolic model-checking methods for real-time systems

April 22, 1996

Satoshi Yamane
Dept. of Computer Science, Shimane University
1060 Nishikawatu, Matue city, Japan
Email:yamane@cis.shimane-u.ac.jp

It is important to verify timing conditions in real-time systems. The verification methods such as model checking and language inclusion algorithm, bisimulation method have been researched. Especially symbolic model checking is promising for verifying a large system. Time models are classified into discrete time model and dense time model. In discrete time model, symbolic model checker based on BDD(Binary Decision Diagram) has been developed. But in dense time model, symbolic model checking based on BDD causes the state-explosion problem because of generating region graph from specification. In this paper, we propose symbolic model checking based on BDD in dense time model, which do not use region graph. In our proposed symbolic model checker, we represent state spaces by both BDD and DBM(Difference Bound Matrices). We have realized effective symbolic model checker based on BDD in dense time model by proposed method.

Key word real-time systems, verification, BDD, symbolic model checking

1 Introduction

It is important to formally verify whether specification satisfies verification properties or not in real-time systems, such as operating systems and commu-

nication protocols, logical circuits[1]. In dense time model, formal verification methods are classified into language inclusion algorithm and model checking as follows[2].

1. If both specification and verification specification are described by timed automaton, verification problem reduces to language inclusion problem in formal language theory[3]. Language inclusion problem is decided if verification specification language is closed under complementation. Timed automaton is based on the ideas on coupling ω-automaton with timing constraints in dense time domain.

2. If specification is described by timed Kripke structure and verification specification is described by real-time temporal logic, verification problem reduces to real-time model checking[4].

In this paper, we focus on model checking ,because many interesting verification properties are expressive. Especially, we focus on symbolic model checking[5] based on BDD(Binary Decision Diagram)[6], bacause we can avoid the state-explosion problem.

On the other hand, there are discrete time model and fictitious clock time model, dense time model[2]. In discrete time model and fictitious clock time model, symbolic model checking systems such as [7] and [8] have been developed. But in discrete time model and fictitious clock time model, asynchronous real-time systems can not be specified and verified[9]. In this paper, we try to specify real-time systems by dense time model and formally verify specification using model checking, especially symbolic model checking. In dense time model, symbolic model checkers such as the verifier of multi-clock automaton[10] and HYTECH[11], KRONOS[12] have been developed. But there are some problems in these symbolic model checkers as follows.

1. The verifier of multi-clock automaton requires large verification cost because of generating region graph[13].

2. HYTECH is implemented by both symbolic representation and semi-decision procedure using mathematica. But HYTECH is not implemeted by BDD.

3. KRONOS is implemented by both symbolic representation and DBM. But KRONOS is not implemeted by BDD.

From 1. and 2., 3., the dense time symbolic model checking based on BDD without region graph have not yet been developed. For this reason, 1. and 2., 3. cause the state- explosion problem.
In this paper, we develope symbolic model checking based on BDD. In general, symbolic model checking is more effective than model checking because of image computation and BDD. But in dense time model, it is difficult to verify systems because of timing constraints. We try to store state transitions and timing constraints by the form (s, x) where s is a set of states represented by BDD and x is a set of timing constraints represented by DBM. This form allows us to verify systems using dense time symbolic model checking. This form have been proposed for language inclusion algorithm by Dill D. and Wong-Toi H[14]. They have realized the real-time symbolic verification system based on BDD and DBM. We develope our real-time symbolic model checking based on their ideas. Our approach for verification of real-time systems is as follows.

1. System specification, which is described by parallel composition of timed automata, is automatically transformed into timed Kripke structure.

2. Dense time symbolic model checking is based on both BDD and DBM. For DBM, we can avoid the state- explosion problem.

In section 2 real-time specification is introduced. In section 3 real-time temporal logic is introduced. In section 4 real-time symbolic model checking is introduced. In section 5 examples of formal specification and timing verification are introduced. In section 6 conclusion is introduced.

2 Specification method for real-time systems

2.1 Specification by timed Buchi automaton

Each process is specified by timed Buchi automaton[15] and system specification is the product automaton of timed Buchi automata. Because timed Buchi automaton is closed under union and intersection. Timed Kripke structure is automatically transformed from system specification.

Definition 1 (timed Buchi automaton) *Timed Buchi automaton is a* $A=(\Sigma, S, S_0, C$ *where*

1. Σ*: a finite set of events*
2. S *: a finite set of states*
3. $S_0 \subseteq S$ *: a finite set of start states*
4. C *: a finite set of clocks*
5. $E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C)$ *: a set of transitions*
6. $F \subseteq S$ *: accepting states*
7. $\Phi(C)$ *represents timing constraints* δ *of clock* C*, and is recursively defined by a set* X $(x \in X)$ *of clock variables and a time constant* D *as follows.*

$$\delta ::= x \leq D | D \leq x | \neg\delta | \delta_1 \wedge \delta_2$$

An edge $(s, s', a, \lambda, \delta)$ *represents a transition from state* s *to state* s' *on input symbol* a*. We represent this transition as follows.*

$$s \xrightarrow{a,\lambda,\delta} s'$$

The set $\lambda \subseteq C$ *gives the clocks to be reset with this transition. A run* r *of timed automaton over a word* $\sigma \in \Sigma^\omega$ *is an accepting run iff* $inf(r) \cap F \neq 0$.

Definition 2 (intersection of timed Buchi automata) *Consider timed Buchi automaton* $Ai=(\Sigma, S_i, s_{0i}, C_i, E_i, F_i)$*, i=1,2, ... ,n. Intersection can be implemented by a trivial modification of the standard product construction for Buchi automata as follows.*

1. *The set of clocks for the product automaton* A *is* $\cup C_i$.
2. *The states of* A *are of the form* $(s_{j1} \times s_{j2} \times .... \times s_{jn})$ *,where each* $s_{ji} \in S_i$ *, and i=1,2, ... ,n, j=1, ... ,m .*
3. *The initial state is of the form* $(s_{01} \times s_{02} \times .... \times s_{0n})$*,where each* $s_{0i} \in S_i$ *, and* $i = 1, 2, \ldots, n$.

4. *The set of transitions consists of* $E_1 \times E_2 \times .... \times E_n$. *The transition of A is obtained by coupling the transitions of the individual automaton having the same label. Let* $\{\langle s_{ji}, s_{ki}, a, \lambda i, \delta i\rangle \in E_i | i = 1, 2, \ldots, n\}$ *be a set of transitions with the same label a. Corresponding to this set, there is a joint transition of A out of each state of the form* $(s_{j1} \times s_{j2} \times .... \times s_{jn})$ *labeled with a. The new state is* $(s_{k1} \times s_{k2} \times .... \times s_{kn})$ *with j=k+1 mod n if* $s_k \in F_k$ *and j=k otherwise.*

5. *The set of clocks to be reset with the transition is* $\cap\lambda i$, *and the associated clock constraint is* $\wedge\delta i$.

6. *The accepting set of A consists of* $F_1 \times F_2 \times \ldots \times F_n$.

Theorem 1 (closure under intersection) *Timed Buchi automaton is closed under intersection.*
(outline of proof)
According to [Definition 2], the class of timed language $L(A_1) \cap L(A_2)$ *accepted by timed Buchi automaton* $A_1 \times A_2$ *is generated, where* $L(A_i)$ *is the class of timed language accepted by timed Buchi automaton* A_i .

2.2 Generation of timed Kripke structure

It is necessary to generate timed Kripke structure from timed automaton in order to realize model checking. The generation method is the same as the reference[16]. A timed Kripke structure T corresponding to a timed automaton A is defined to be a timed Kripke structure such that there exists one to one correspondence between the state-input sequences of A and the paths from one of an initial state. Next, we formally define the generation method of timed Kripke structure as follows.

Definition 3 (timed Kripke structure) $T=(S', \mu', R', \pi')$ *be a timed Kripke structure. where*

1. S' : *a finite set of states*

2. $\mu' : S' \rightarrow 2^P$ *assigns to each state the set of atomic propositions true in that state.*

3. R' : *a binary relation on* S' ($R' \subseteq S' \times S'$) *which gives the possible transitions between states.*

4. π' : $S' \rightarrow 2^C \times \Phi(C)$ *assigns to each state the set of clocks.*

Next, we define operational semantics for a timed Kripke structure T in terms of a transition system. A timed-state of the system is a pair q=(si', xi), i=0, ... ,n, where $s'_i \in S'$ is a state and x_i is a vector of clock values.

Definition 4 (semantics of timed Kripke structure) *We define operational semantics for a timed Kripke structure T as follows.*

1. *The set* q_0 *of initial states is the set of all timed-states whose state component is an initial state in T, and whose clocks values are all equal to 0, as given by q0=*{ $(s'_0, 0)$ $|s'_0$ *is the set of initial states* }

2. *For each transition* $s'_i \rightarrow s'_{i+1}$, *let*
$R' =$ { (s'_i, x_i), (s'_{i+1}, x_{i+1}) $|x_i \in \pi'$ (s'_i) *and* $x_{i+1} \in \pi'$ (s'_{i+1}) ,$s'_i \times s'_{i+1} \in R'$}

Next, we define the generation of timed Kripke structure from a timed automaton.

Definition 5 (the generation of timed Kripke structure) *Let* $A=(\Sigma, S, S_0, C, E, I$ *be a timed automaton , and* $T=(S', \mu', R', \pi')$ *be a timed Kripke structure. The generation method is as follows.*

1. $S' = E$

2. $R' \subseteq E \times E$

3. $\mu' = E \rightarrow 2^\Sigma = S' \rightarrow 2^P$

4. $\pi' = E \rightarrow 2^C \times \Phi(C) = S' \rightarrow 2^C \times \Phi(C)$

The example of the generation of timed Kripke structure is as shown in Fig.1.

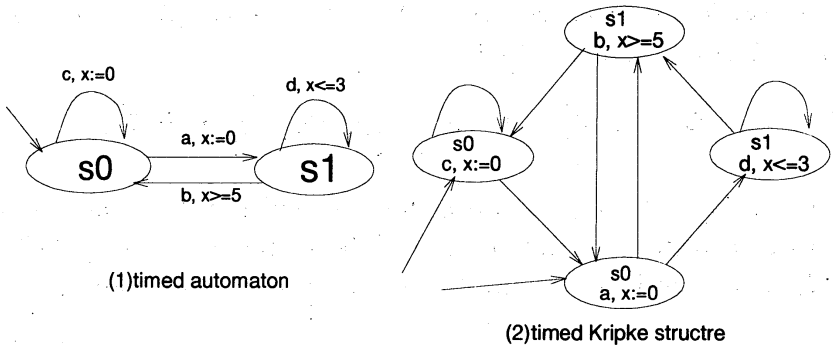

Fig. 1 Generation method of timed Kripke structure

Next, we explain that this transformation is compatible with the definition of truth of temporal logic.

Theorem 2 (compatibility) *For all timed automaton A and temporal logic ϕ , timed Kripke structures T, it holds that $\models_A \phi$ iff $\models_T \phi$*
(proof)
From the definition of A, it follows that for each infinite sequence

$$s_0 \xrightarrow{a_0,\lambda_0,\delta_0} s_1 \xrightarrow{a_1,\lambda_1,\delta_1} s_2 \xrightarrow{a_2,\lambda_2,\delta_2} \cdots$$

of transitions of A there is a path $(s'_0, s'_1, s'_2, \ldots)$ of T,

such that

$$s_i \xrightarrow{a_i,\lambda_i,\delta_i} s_{i+1} \sim s'_i$$

for all

$$s_i \xrightarrow{a_i,\lambda_i,\delta_i} s_{i+1}$$

Conversly, for each infinite sequence $(s'_0, s'_1, s'_2, \ldots)$ of T, there is a path

$$s_0 \xrightarrow{a_0,\lambda_0,\delta_0} s_1 \xrightarrow{a_1,\lambda_1,\delta_1} s_2 \xrightarrow{a_2,\lambda_2,\delta_2} \cdots$$

of A
such

$$s_i \xrightarrow{a_i,\lambda_i,\delta_i} s_{i+1} \sim s'_i$$

for all s'_i in the first sequence.
Next observe that if

$$s_i \xrightarrow{a_i,\lambda_i,\delta_i} s_{i+1} \sim s'_i$$

then $s_{i+1} \models_A \phi$ iff $si' \models_T \phi$. Using this observation and the above correspondence between sequences of transitions of A and sequences of states of T, we can induction over ϕ prove that for all transtions of A and states of T.

3 Real-time temporal logic

Verification property specification is described in RTCTL(Real-Timed CTL) , which expands TCTL(Timed CTL)[13] with next state operator as follows.

Definition 6 (syntax of RTCTL) *The formulas ϕ of RTCTL are inductively defined as follows.*
$\phi ::= p|\neg\phi|\phi_1 \rightarrow \phi_2|EX_{\sim}c\ \phi_1|\ E\ (\phi_1\ U_{\sim c}\ \phi_2)\ |EG_{\sim c}\ \phi_1$

1. *E : for some sequence of states a formula holds*
2. *X : next states operator*
3. *U : until operator*
4. *G : always operator*
5. *$p \in$ (atomic proposition)*
6. *$c \in N$ (natural number)*
7. *$\sim$ is binary relation $<, \leq, =, \geq, >$*

Informally, E $(\phi_1\ U_{<c}\ \phi_2)$ means that for some sequence of states $s'_0, s'_1, s'_2, \ldots$ there exits a sequence of states of time length less than C such that ϕ_2 holds at the last state and ϕ_1 holds at all its intermediate states.

Definition 7 (syntactic abbreviations for RTCTL) *We can specify all the temporal formulas using following syntactic abbreviations.*

1. $EF_{\sim c}\ \phi_1 = E\ (true U_{\sim c}\ \phi_1)$
2. $AX_{\sim c}\ \phi_1 = \neg E \neg X_{\sim c}\ \phi_1$
3. $AG_{\sim c}\ \phi_1 = \neg E F_{\sim c}\ \neg\phi_1$
4. $A\ \phi_1\ U_{\sim c}\ \phi_2 = E[\neg\phi_2\ U_{\sim c}\ \neg\phi_1 \wedge \neg\phi_2] \wedge \neg EG_{\sim c}\ \neg\phi_2$

Here we define RTCTL-semantics in order to interpret RTCTL-formula based on timed Kripke structure as follows.

Definition 8 (semantics of RTCTL) *For a timed Kripke strucutre* $T=(S', \mu', R', \pi')$ *, a state* $s'_0 \in S'_0$ *, a sequence of states* $s'_0, s'_1, \ldots, s'_n$ *and a RTCTL-formula* ϕ*, the satisfaction relation* $(T, s'_0) \models \phi$ *is defined inductively as follows.*

1. $(T, s'_0) \models p$ *iff* $p \in \mu'\ (s'_0)$.
2. $(T, s'_0) \models \neg\phi_1$ *iff* $(T, s'_0) \models \phi_1$ *is unsatisfiable.*
3. $(T, s'_0) \models \phi_1 \rightarrow \phi_2$ *iff* $(T, s'_0) \models \phi_1$ *is unsatisfiable or* $(T, s'_0) \models \phi_2$.
4. $(T, s'_0) \models EX_{\sim c}\ \phi_1$ *iff for some state* s'_1 *such that* $(s'_0, s'_1) \in R'$, $s'_1 \models \phi_1$ *and* $\sim c$ *is satisfiable with* $\mu'\ (s'_1)$.
5. $(T, s'_0) \models E\ (\phi_1\ U_{\sim c}\ \phi_2)$ *iff for some sequence of states* $(s'_0, s'_1, \ldots, s'_n)$, $\exists i[i \geq \wedge\ (T, s'_i) \models \phi_2 \wedge \sim c$ *is satisfiable with* $\mu'\ (s'_i)\ \wedge \forall j[\ 0 \leq j < i \rightarrow (T, s'_j) \models \phi_1 \wedge \sim c$ *is satisfiable with* $\mu'\ (s'_j)\ ]\ ]$
6. $(T, s'_0) \models EG_{\sim c}\ \phi_1$ *iff for some sequence of states* $(s'_0, s'_1, \ldots, s'_n)$, $\forall i[0 \leq i \rightarrow (T, s'_i) \models \phi_1 \wedge \sim c$ *is satisfiable with* $\pi'\ (s'_i)\ ]$

4 Verification algorithm for real-time symbolic model checking

In real-time symbolic model checking, for a timed Kripke structure T, we represent state transitions relation R' as BDD and a set of states as the form (s'_i, x_i) $(i = 0, 1, \ldots, n)$, where $s'_i \in S'$ is a state represented by BDD and x_i is a vector of clock values represented by DBM(differences bounds matrix). In order to realize real-time symbolic model checking, we compute a set of states

that satisfy the formulas by inverse image computation and test whether a set of states satisfy timing constraints using DBM.

We define real-time symbolic model checking algorithm after defining inverse image computation and DBM.

4.1 Inverse image computation

Many of the idea used in symbolic model checking can be explained by considering the problem of computing reachable state sets, since reachable state computations are at the heart of model checking[16]. Let s_i' be a set of states represented by the BDD s_i' (V). We wish to compute a BDD s_j' (V') that represents the states reachable from s_i' by the transitions in the transition relation R' :
s_j' $(V') = \exists$ V. $[s_i'$ (V) $\wedge$ R' (V, V') $]$.
This is called image computation. But in real-time symbolic model checking, we use inverse image computation. In inverse image computation, we compute a BDD s_i' (V) that represents the states backward reachable from s_j' by the transitions in the transition relation R' :
s_i' (V)=$\exists V'.[s_j'$ (V') $\wedge$ R' (V, V') $]$.

4.2 DBM(differences bounds matrix)

4.2.1 reachability analysis(test timing constraints)

We can compute a set of states that satisfy the formulas using inverse image computation. But we must test whether a set of states satisfy timing constraints. We will test timing constraints(reachability analysis) using DBM[18,19] as follows.

Definition 9 (DBM(Difference Bounds Matrices))) *DBM consists of the matrix of timer valuations. Timer valuations are defined as follows.*
$\forall i, j \in C : t_i - t_j \leq d_{ij}$
where

1. t_i : *clock variable*
2. t_j : *clock variable*
3. d_{ij} : *clock constant*

The (i, j)-element of DBM is equal to d_{ij}. A fictitious clock t0 that is always exactly zero is introduced. $d_{ij} \subseteq \{..., -2, -1, 0, 1, 2, ...\} \cup \{..., -2^-, -1^-, 0^-, 1^-, 2^-, ...\} \cup \{-\infty\} \cup \{\infty\}$. *The ordering ¡ over the integers is extended to dij by the following law: for any interger a,* $-\infty < a^- < a < (a+1)^- < \infty$.

Next, we define reachability analysis using DBM.

Definition 10 (reachability analysis) *Generate the intersection of canonical DBMs, check reachability between two states. Here we check whether state* $D \rightarrow D'$ *is possible or not.*

(1)Perform canonical DBM by Floyd-Warshall' algorithm *The each inequality of DBM is of the form* $t_i - t_j \leq d_{i,j}$. *An alternative formulation of it allows the construction of a constraint graph for a given set of inequalities. Each variable is represented as a node in the graph, and an inequality* $t_i - t_j \leq d_{i,j}$ *is represented by a directed edge with weight* $d_{i,j}$ *connecting* t_i *to* t_j *as shown in Fig.2. For this reason, we can get canonical DBM by Floyd-Warshall' algorithm[18].*

(2)Intersect canonical DBMs *intersection DBM =* $min\ \{d_{ij}, d'_{ij}\}$ *where*

1. $[d_{ij}]$ *: canonical DBM of state D*
2. $[d'_{ij}]$ *: canonical DBM of state D'*

In dense time model, in order to reach D' from D, there is intersection DBM between D and D'[19].

(3)Test intersection DBM *If there is a negative-cost cycle in intersection DBM, it is impossible to reach D' from D. If there is no negative-cost cycle in intersection DBM, it is possible to do so.*

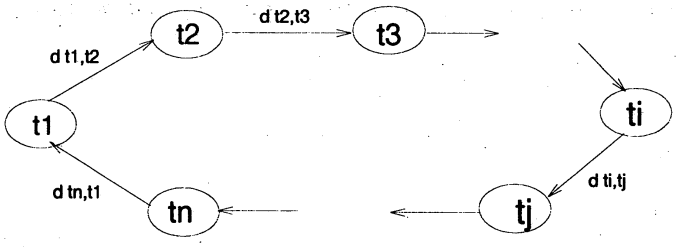

Fig. 2 the graph representation of DBM

Next, we explain the validity of reachability analysis as follows.

Theorem 3 (the validity of reachability analysis) *If there is a negative-cost cycle in intersection DBM of D and D' , it is impossible to reach D' from D.*
(proof)
We call a sequence of clock variables $t_1, t_2, \ldots, t_n$. *The cost of the path in intersection DBM is* $d_{t1,t2}+d_{t2,t3}+\ldots+d_{tn,t1}$. *We can define* $d_{t1,t2}, d_{t2,t3}, \ldots, d_{tn,t1}$ *as follows.*
$t_1 - t_2 \leq d_{t1,t2}$
$t_2 - t_3 \leq d_{t2,t3}$
$\ldots$
$t_n - t_1 \leq d_{tn,t1}$
The cost=$(t_1 - t_2) + (t_2 - t_3) + \ldots + (t_n - t_1) = t_1 - t_1$
If there is a negative-cost cycle $(t_1 - t_1 < 0)$, *it is impossible to reach D' from D.*

4.2.2 Testing whether a set of states satisfy $\sim c$ *in* $\phi 1\ U_{\sim c}\ \phi_2$

We must test whether a set of states satisfy $\sim c$ *in* $\phi_1\ U_{\sim c}\ \phi_2$. In other words, we test whether the time elapsed in traversing a sequence between ϕ_1 and ϕ_2 satisfies $\sim c$. We test it using a clock variable in DBM as follows.

Definition 11 (the computation of the time elapsed in traversing a sequence) *We define the computation of the time elapsed in traversing a sequence between* s'_j *and* s'_k $(j < k)$. *We focus on some clock variable x in DBM.*

(1)when x is not reset between s'_j **and** s'_k *The timing constraint is* $x \leq d$ *or* $x \geq d$ *at* s'_j *and* $x \leq h$ *or* $x \geq h$ *at* s'_k *,where* $d \leq h$. *We compute the time elapsed in traversing a sequence between* s'_j *and* s'_k *as follows in Fig.3.*

1. *case* $x \leq d$ *and* $x \leq h$: *The elapsed time t is* $t \leq$ h-d.
2. *case* $x \geq d$ *and* $x \geq h$: *The elapsed time t is* $t \geq$ h-d.
3. *case* $x \leq d$ *and* $x \geq h$: *The elapsed time t is* $t \geq h$.
4. *case* $x \geq d$ *and* $x \leq h$: *The elapsed time t is* $t \leq$ h-d.

(2)when x is reset between s'_j and s'_k *Assuming that x is reset at a state s'_l $(j < l < k)$. We compute the time elapsed in traversing a sequence between s'_j and s'_l and the time elapsed in traversing a sequence between s'_l and s'_k . We compute the elaped times using the same way as (1). Finally, we add the time elapsed in traversing a sequence between s'_j and s'_l and the time elapsed in traversing a sequence between s'_l and s'_k*

From (1) and (2), we can compute the time elapsed in traversing a sequence between s'_j and s'_k .

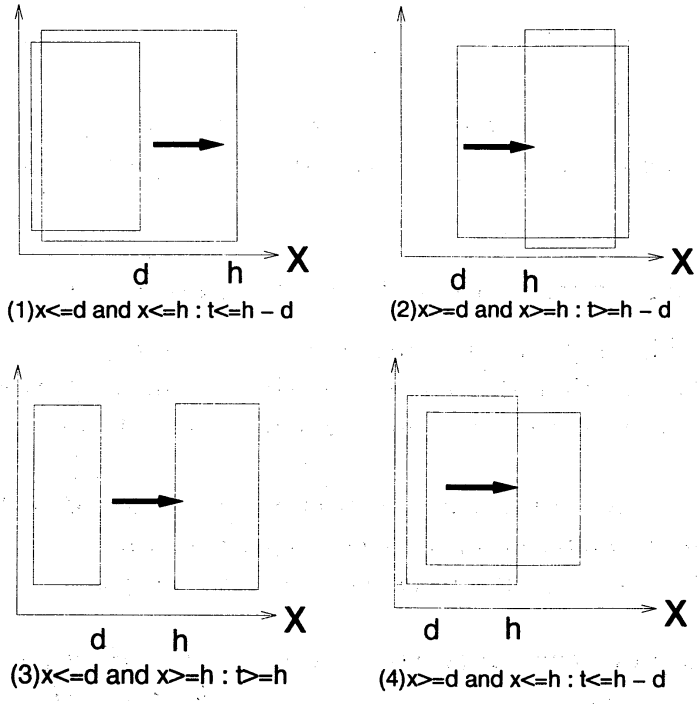

Fig. 3 the time elapsed in state transitions

4.3 Real-time symbolic model checking

Finally, we define real-time symbolic model checking as follows.

Definition 12 (real-time symbolic model checking) *The real-time symbolic model checking consists of following procedures.*

1. *Firstly, we convert system specification into timed Kripke structure.*

2. *Secondly, we represent state transtions relation R' as BDD and a set of states as the form* (s'_i, x_i) $(i = 0, 1, \ldots, n)$ *, where* $s'_i \in S'$ *is a state represented by BDD and* x_i *is a vector of clock values represented by DBM(differences bounds matrix).*

3. *Next, we compute the set of states that satisfy every subformula using inverse image computation and we test whether the set of states satisfy timing constraints or not using DBM. We test whether the time elapsed in traversing a sequence satisfy timing constraints in a formula(for example,* $\sim c$ *in* $E\ (\phi 1\ U_{\sim c}\ \phi 2)$.

4. *Finally, after determining the set S of states that satisfy the formula f, we test whether* s'_0 *is a subset of S(that is, whether* $\neg s'_0$ *(V)* $\vee$ *S(V) is the BDD representing true.) If it is, then the timed Kripke structure satisfies f.*

Next, we define real-time symbolic model checking algorithm as follows.

Definition 13 (real-time symbolic model checking algorithm) *For given a structure* $T=(S', \mu', R', \pi')$ *and a temporal logic formula f, we determine whether* $\models_T f$. *The algorithm is based on inverse image computation. Firstly, we compute the set of states that satisfy all subformulas of f of length 1, the second stage compute the set of states that satisfy all subformulas of f of length 2, and so on. At the end of ith stage, the set of states that satisfy the set of all subformula of length* $\leq i$ *will be computed. To perform computing the set of states at stage i, the set of states gathered in earlier stages is used. One can conclude that* $\models_T f$ *if the initial state* (s'_0) *is a subset of the set of states. Let* ϕ *be a subformula of f. We compute the set of states that satisfy all subformulas* ϕ *of f of length 1 as follows.*

1. $\phi \in$ *p(atomic proposition)*
   nothing to do

2. $\phi = \neg\phi_1$
   return $\neg s\phi_1$ *(V)*
   where $s\phi_1$ *(V) means the set of states that satisfy* ϕ_1 *represented by BDD*

3. $\phi = \phi_1 \rightarrow \phi_2$
   return $\neg\ s\ \phi_1\ (V) \vee s\ \phi_2\ (V)$

4. $\phi = EX_{\sim c}\phi_1$
   return $functionEX\phi_1\ (EX\phi_1(V), \sim c)$
   $functionEX\ \phi_1\ (EX\ \phi_1\ (V)\ , \sim c\ )$
   $EX\phi_1\ (V) = \exists V'\ .[\ R'\ (V, V')\ \wedge \phi 1\ (V)]$;
   If $\sim c$ *is not satisfiable with* $\phi'\ (EX\ \phi_1\ (V)')$, *we compute* $EX\ \phi_1\ (V)$ *as follows.*
   $EX\phi_1\ (V) := EX\phi_1\ (V)$ - $EX\ \phi_1\ (V)\ '$;
   We test whether $EX\phi_1\ (V)$ *is reachable from* $\phi_1\ (V)$ *satisfying timing constraints ;*
   If there is the set of states $EX\phi_1\ (V)\ '$ *that does not satisfy timing constraints,*
   we compute $EX\ \phi_1\ (V)$ *as follows.*
   $EX\phi_1\ (V) := EX\phi_1\ (V)$ - $EX\phi_1\ (V)\ '$;
   return $EX\phi_1\ (V)$;
   end $functionEX\phi_1$;

5. $\phi = E\phi_1\ U_{\sim c}\ \phi_2$
   $T(V) := \phi_2\ (V)$;
   repeat {
   $U(V) := \phi_1\ (V) \wedge \exists V'.[R'(V, V') \wedge T(V')]$;
   If there is the set of states $\pi'\ (U(V)')$ *that does not satisfy* $\sim c$, *we compute* $U(V)$ *as follows.*
   $U(V) := U(V)$ - $U(V)\ '$;
   If there is the set of states $U(V)\ '$ *that does not satisfy timing constraints, we compute* $U(V)$ *as follows.*
   $U(V) := U(V) - U(V)\ '$;
   If $U(V)$ *is included in* $\phi_1\ (V)$, *return* $\phi_1\ (V)$;
   If $U(V)$ *is not included in* $\phi_1\ (V)$, $T(V) := U(V) + T(V)$;
   }

6. $\phi = EG_{\sim c}\phi_1$
   $T(V) := \phi_1\ (V)$;
   repeat {
   $U(V) := \phi_1\ (V) \wedge functionEX\ \phi_1\ (T(V), \sim c\ )$;
   If $U(V)$ *is equal to* $\phi_1\ (V)$, $\phi_1\ (V) := T(V)$ *and return* $\phi_1\ (V)$;

T(V):=U(V) ;
}

5 The verification system

5.1 Configuration of the verification system

We have developed the veirfication system based on this method using SBDD library[21] as shown in Fig.4. It runs on SUN4/IP(12MB). The veirfication system consists of compiler(1kstep) and real-time symbolic model checker(3kstep), which are implemented in C language.

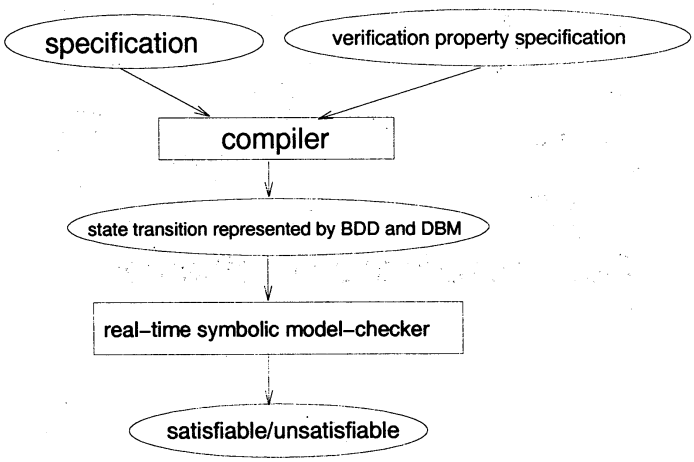

Fig. 4 Configuration of verification system

5.2 Verification example

5.2.1 Specification

We present here the timed automata for the senders and the receivers of the CSMA/CD protocol[22]. The specification of sender and receiver is shown in Fig.5. The sender stays in initial state S0 until it receives a message. Then, it tests the bus to see if it is ready or busy, collision detection. In receiver, at initial state R0, it is ready to be in the transmission of a message. If one of the senders starts sending, the receiver sets to zero the timer y. When y is less than or equal to 5,the bus is sensed idle for the other sender.

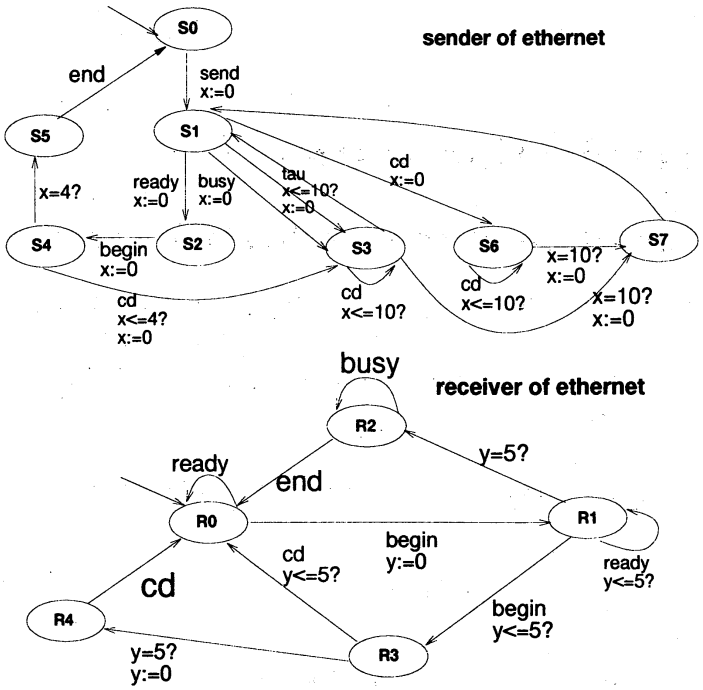

Fig. 5 specification of ethernet

5.2.2 Verification

We have verified using real-time symbolic model-checker whether verification properties by RTCTL are satisfiable in specification. We input timed automata into compiler by the programming language format as shown in Fig.6.

```
System specification ;
System configuration ;
  system = sender * receiver ;
Process specification(sender) ;
State definition part s0,s1,...,s7 ;
Event definition part send, tau, ready, cd, busy, begin, end ;
Initial state definition part s0 ;
State transition part
  s0 ⇒send, x:=0  ⇒s1 ;
  s1 ⇒cd, x:=0  ⇒s6 ;
       :
       :
end ;
Process specification(receiver) ;
State definition part R0, R1, ...,R4 ;
Event definition part begin, cd, ready, cd ;
Initial state definition part R0 ;
State transition definition part
  R0 ⇒ready, y:=0  ⇒R1 ;
  R0 ⇒begin, y:=0  ⇒R1 ;
  R1 ⇒begin, y<=5  ⇒R3 ;
       :
       :
end;
```

Fig. 6 Example of input format into compiler

The verification properties are (1)EF send and (2)EF(send $\vee$ E (ready U $_{\leq 5}$ end)), (3)EF(send $\vee$E(ready U $_{\leq 10}$ end)), (4)EG($send \vee ready \wedge begin$) as shown in Table 1. In order to compare real-time symbolic model-cheker and real-time model-checker, we have verified using real-time model-checker whether verification properties by RTCTL are satisfiable in specification. We have already reported real-time model-checker [4]. When we have verified it using real-time model-checker, we cannot verify 5665 states because of being not enough memory. But we can verify more than 14588 states using real-time symbolic model-checker. For this, we can avoid the state-explosion problem. We can show symbolic model-checking fro dense time real-time systems is effective.

Table 1 evaluation of verification costs

verification specification	true/ false	specification		symbolic mode-checking		model-checking	
		states	transitions	memory	CPU time	memory	CPU time
EF send	true	122	338	311kb	0.1sec	94kb	1.0sec
		870	3801	348kb	8.9sec	337kb	10.3sec
		1213	5472	408kb	28.7sec	382kb	18.9sec
		2792	14627	423kb	109.4sec	460kb	100.0sec
		5665	32445	431kb	553.9sec	not enough memory	
		14588	87724	463kb	2736.2sec	not enough memory	
EF(send \/E(ready U<=5 end))	true	122	338	335kb	0.5sec	96kb	1.osec
		870	3801	350kb	16.5sec	301kb	10.1sec
		1213	5472	406kb	52.5sec	367kb	19.0sec
		2792	14627	444kb	170.6sec	491kb	101.8sec
		5665	32445	454kb	924.5sec	not enough memory	
		14588	87724	457kb	3341.5sec	not enough memory	
EF(send \/E(begin U<=10 end))	true	122	338	303kb	1.3sec	93kb	1.1sec
		870	3801	351kb	31.4sec	336kb	10.2sec
		1213	5472	409kb	98.2sec	369kb	19.1sec
		2792	14627	450kb	307.3sec	369kb	100.8sec
		5665	32445	453kb	1828.3sec	not enough memory	
		14588	87724	456kb	7154.8sec	not enough memory	
EG(send \/ready /begin)	false	122	338	320kb	0.1sec	96kb	1.1sec
		870	3801	348kb	3.4sec	343kb	10.2sec
		1213	5472	380kb	14.5sec	380kb	19.1sec
		2792	14627	387kb	47.7sec	477kb	101.sec
		5665	32445	433kb	220.5sec	not enough memory	
		14588	87724	459kb	1064.5sec	not enough memory	

6 Conclusion

In this paper, we have proposed real-time symbolic model checking method based on both BDD and DBM. We have developed the verification system and shown it effective by the CSMA/CD protocol. We can avoid the state-explosion problem. But we cannot verify 1020 states such as [5]. This shows the verification system for dense time systems is high cost. But the dense time has a desirable feature for representing two causally independent events in asynchronous real-time systems. In order to verify very large systems, we are developing the compositional verification system such as [5].

7 References

References

[1] Kavi K.M.:Real-time Systems, Abstraction, Languages and Design Methodologies, P.660,IEEE Computer Society (1992)

[2] Alur R. Henzinger H.A.:"Logics and Models of Real Time:A Survey",LNCS 600,pp.74-106(1992)

[3] Yamane S.:"Formal timing verification techniques for distributed system",5th IEEE CS Workshop on Future Trends of Distributed Computing Systems(FTDCS'95),pp.454-460,IEEE Computer Society(1995)

[4] Yamane S.:"Verification system for real-time specification based on extended real-time logic",International Workshop on Real-Time Computing Systems and Applications(RTCSA),pp.192-196,IEEE Computer Society(1995)

[5] McMillan K.L.:Symbolic Model Checking,Kluwer,P.194(1993)

[6] Bryant R.E.:"Graph-Based Algorithms for Boolean Function Manipulation",IEEE Transactions on Computers, Vol.C-35,No.8,pp.677-691(1986)

[7] Campos S.V. Clarke E.M.:"Real-time symbolic model checking for discrete timemodels",Proc. first AMAST Inter. Workshop in Real-time systems,pp.129-145,World Scientific(1994)

[8] Yang J. Mok A.K. Wang F.:"Symbolic model checking for event-driven real-timesystems",Proc. real-time systems symposium,pp.23-32(1993)

[9] Alur R.:Techniques for automatic verification of real-time systems,Phd thesis,Stanford university(1991)

[10] Wang F. Mok A.K. Emerson E.A.:"Symbolic model checking for distributed real-time systems",LNCS 670,pp.632-651(1993)

[11] Alur R. Henzinger T.A. Pei-Hsin Ho:"Automatic symbolic verification of embedded systems",Proc. real-time systems symposium,pp.2-11(1993)

[12] Henzinger T.A. Nicollin X. Sifakis J. Yovine S.:"Symbolic model checking for real-time systems",IEEE symp. on Logic in Computer Science,pp.394-406(1992)

[13] Alur R. Courcoubetis C. Dill D.:"Model-Checking for Real-Time Systems",Proc. 5th IEEE Logic in Computer Science,pp.414-425(1990)

[14] Dill D. Wong-Toi H.:"Verification of real-time systems by successive over and under approximation",LNCS939,pp.409-422(1995)

[15] Alur R. Dill D. :"The Theory of Timed Automata",LNCS 600,pp.45-73(1992)

[16] Hiraishi H.:"Design verification of sequential machines based on ε -free regular temporal logic",Computer hardware description languages and their applications",pp.249-263,Elsevier science publishers(1990)

[17] Burch J.R. Clarke E.M. Long D.E. Mcmillan K.L. Dill D.:"Symbolic Model Checking for Sequential Circuit Verification",IEEE Trans. CAD of ICS,Vol.13,N0.4, pp.401-424(1994.4)

[18] Dill D.:"Timing assumptions and verification of finite-state concurrent systems",LNCS 407,pp.197-212(1989)

[19] Alur R. Courcoubetis C. Dill D. Halbwachs N. Wong-Toi H.:"An Implementation of Three Algorithms for Timing Verification Based on Automata Emptiness",Proc. Real-Time Systems Symposium,pp.157-166(1992)

[20] Emerson E.A.:"Temporal and modal logic", Handbook of Theoretical Computer Science,Vol.B,pp.997-1072(1990)

[21] Minato S. Ishiura N. Yajima S.:"Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation",Proc. 27th Design Automation Conference,pp.52-57(1990)

[22] IEEE ANSI/IEEE 802.3, ISO/DIS 8802/3. IEEE Computer Society Press(1985)

[23] Nicollin X. Sifakis J. Yovine S. :"Compiling real-time specifications into extended automata",IEEE trans. on SE,Vol.18,No.9,pp.794-804(1992)

Preface

This volume contains the proceedings of the RIMS Workshop in Computing titled Concurrency Theory and Applications '96 which took place at Kyoto University in July 1996. The aim of the workshop was to bring together researchers and practitioners interested in concurrency in order to discuss recent developments and trends in concurrency theory, exchange experiences and opinions on applications of concurrency and to establish research contacts.

In response to the call for papers, fifteen papers were accepted to be presented at the workshop and to appear in the proceedings. Also, four speakers from abroad were invited: Professor Matthew Hennessy of the University of Sussex, UK, Professor Scott Smolka of SUNY at Stony Brook, USA, Dr Huimin Lin of the Chinese Academy of Sciences, Beijing, China and Dr Mark Harman of the University of North London, UK.

The papers appearing in this volume span a wide range of topics, including timed and classical process algebras and their applications, semantics for CML, semantics and types for the π-calculus, model checking and description of real time and multimedia systems, program slicing and temporal logic.

I would like to thank all those who presented their research at the workshop as well as those who only attended it. Moreover, I particularly wish to thank Yasuhiko Minamide, Shoji Yuen and Susumu Nishimura for an invaluable help throughout the preparation and organisation of the workshop.

Irek Ulidowski