# Using MTBDDs for Discrete Timed Symbolic Model Checking

Thomas Kropf Jürgen Ruf

Institute of Computer Design and Fault Tolerance (Prof. D. Schmid)
University of Karlsruhe, Kaiserstr. 12, 76128 Karlsruhe, Germany
{Thomas.Kropf,Juergen.Ruf}@informatik.uni-karlsruhe.de
http://goethe.ira.uka.de/hvg/

**Abstract**

*The verification of timing properties is an important task in the validation process of embedded and real time systems. Temporal logic model checking is one of the most successful techniques as it allows the complete automation of the verification.*

*In this paper, we present a new approach to symbolic QCTL (Quantitative CTL) model checking. In contrast to previous approaches we use an intuitive QCTL semantics, provide an efficient model representation and the new algorithms require less iteration steps compared to translating the QCTL problem into CTL and using standard CTL model checking techniques. The new model checking algorithm is based on a MTBDD representation. Some experimental results show the efficiency of the new approach.*

## 1 Introduction

Formal verification is an important step in the system design process. Using model checking as verification technique, the validity of a temporal logic specification has to be shown with regard to a model, representing the system to be verified. Techniques have been developed to efficiently represent state sets and transitions relations symbolically by using binary decision diagrams (ROBDDs, [13]). Proceeding this way systems with a large state space have become verifiable [8].

Standard CTL is not sufficient to cope with quantitative timing constraints. Thus various timed temporal logics have been presented to argue about timing. Logics based on real numbers model time as a continuous quantity[11]. Specifications are given by a timed variant of CTL called TCTL. This approach is very expressive and allows a very accurate system modeling. However, the price to be paid is an exponential runtime with regard to the number of clocks and timing constraints, although symbolic model checking is possible in principle [15].

In many cases it is sufficient to use a discrete time model, e.g. for clocked systems like traffic light controllers or communication protocols. Thus only natural numbers are assigned as transition times to state transitions. CTL is extended appropriately by allowing quantitative CTL operators, leading to languages like QCTL. Model checking is either based on a translation into a standard CTL model checking problem or by using dedicated representation techniques [14, 6]. The translation approach requires the mapping of timed structures to unit delay structures, leading to a state explosion if large transition times occur. This can be avoided when timed structures are used directly. In this case, however, the intuitive semantics of QCTL gets lost when composing temporal operators. Thus the resulting model checking algorithm is of limited use only (see section 4.1).

Hence it is desirable to find techniques for compactly representing timed structures by simultaneously keeping a clean QCTL semantics. This paper describes a solution to this problem. In addition to solving the described dilemma, to our knowledge for the first time multi-valued BDDs [2] are used to efficiently represent timed state transitions and state sets.

Due to space limitations this paper only contains an informal presentation of the basic ideas of our approach. A detailed and formal description with all model checking algorithms can be found in [10].

## 2 Binary Decision Diagrams

Symbolic model checking makes excessive use of Boolean functions, which may be compactly and efficiently represented by ROBDDs [13]. In contrast to ROBDDs, *Multi terminal binary decision diagrams* (MTBDDs [2]) represent *pseudo Boolean functions*. These functions map bit vectors to a finite set of elements.

**Definition 2.1 (Pseudo Boolean Function)** *Given the set* $\mathbb{B}$ *containing the Boolean values* $\{0,1\}$,

ED&TC '97 on CD-ROM

*and a finite set $A$, then each function $f : \mathbb{B}^n \to A$ is a pseudo Boolean function.*

Like ROBDDs, MTBDDs may be represented by directed acyclic graphs where the leaf values contain elements of $A$. The usual ROBDD algorithms for terminal defined functions and for composing two ROBDDs can be extended to MTBDDs.

# 3 QCTL

Quantitative computation tree logic (QCTL) is an extension of CTL [3] by quantitative temporal operators. The logic has been introduced in [14, 6]. In QCTL each temporal operator of CTL carries a temporal scope as additional information, e.g. EF[5]$f$ means there exists a path such that $f$ holds in one of the next five states.

Although QCTL has the same expressiveness as CTL, specifications containing quantitative timing can be given in a more natural and succinct way. The semantics of QCTL may be given with regard to a standard unit-delay temporal structure. In fact, fragments of QCTL have been incorporated into the SMV model checker [9]. However, the virtues of QCTL are exhibited only when timed temporal structures are used for model checking. In this case it is possible to interpret the semantics of the QCTL operators directly, without translating them into the unit-delay model. Moreover timed temporal structures allow a more compact representation of the state space and the transition relation, especially if large delay times are used. As the unit delay model needs additionally states, the number of variables encoding the states may increase.

## 3.1 State of the Art

To represent temporal structures Campos and Clarke suggested to use a timed transition relation, where delay times are encoded using additional Boolean values [14]. Another recent suggestion is to partition the transition relation, grouping together transitions with the same delay in a separate ROBDD [6].

Unfortunately, if no unit-delay model is used, the QCTL semantics diverges from its CTL counterpart and is not what one would expect [5]. This hinders the use of nested QCTL expressions.

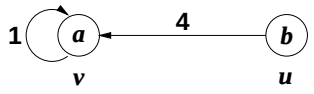


Figure 1: A timed temporal structure

**Example 3.1** *Given the timed temporal structure of figure 1. Consider the following three formulas and the sets of states where these formulas hold:*

$$\begin{array}{lcl} \mathrm{EF}[3]a & \widehat{=} & \{v\} \\ \mathrm{EF}[3]\mathrm{EF}[3]a & \widehat{=} & \{v\} \\ \mathrm{EF}[6]a & \widehat{=} & \{u, v\} \end{array}$$

*The last two equations are not intuitive, since one expects that* EX[3]EX[3] *and* EX[6] *yield in the same result.*

The reason for this effect is that these approaches consider only the existing states, but there are implicitly more states to be considered than are defined in the temporal structure. To avoid this counter intuitive semantics, we redefine it.

## 3.2 Semantics of QCTL

**Definition 3.1 (Timed Temporal Structure)** *A timed temporal structure is a quintuple* $\mathcal{M} = (AP, S, T, L_S, L_T)$.

- $AP$, *the set of atomic propositions*
- $S$, *the set of states*
- $T \subseteq S \times S$, *the set of transitions with* $\forall s \in S.\exists s' \in S.(s, s') \in T$
- $L_S : S \to \wp(AP)$, *the state labeling function*
- $L_T : T \to \mathbb{N}_+$ *a delay time for each transition*

The standard CTL structure is extended by $L_T$, representing time delays at the edges. The delay times are non-zero natural numbers. The semantics of extended structures can be given in terms of single transition models by introducing new states (intermediate states) on timed edges carrying the same labels as the respective transition source state.

Next, we introduce derived temporal structures. These structures contain additional states to represent a unit delay derivate of a timed temporal structure. This derived structure contains more information than a pure translation into a CTL structure.

**Definition 3.2 (Derived Temporal Structure)** *A derived temporal structure is a 6-tuple* $\widetilde{\mathcal{M}} = (AP, S_\prec, H, T, L_H, \tilde{L}_T)$.

- $AP$, *the set of atomic propositions*
- $S_\prec$, *the set of states, with partial ordering* ($\prec$) *on the states*
- $H \subseteq S_\prec$, *the set of main states*
- $T \subseteq H \times H$, *the set of transitions with* $\forall s \in H.\exists s' \in H.(s, s') \in T$
- $L_H : H \to \wp(AP)$, *the main state labeling function*
- $\tilde{L}_T : T \to \wp(S_\prec) \setminus \{\emptyset\}$, *the transition labeling function. The order* $\prec$ *is total on* $\tilde{L}_T(t)$.

*For all $v \in S_\prec \setminus H$ exists a $u \in H$ such that $u \prec v$.*

The partial ordering ($\prec$: "comes earlier than") on the state space defines state chains for the transitions. These chains represent the delay time for the transition between two main states. States are only comparable if they are on the same transition. The least element of every partial ordering is the source main state of the transition. The main states of the derived model represent the states of the timed model. Only main states carry labels. The labels of the other states (intermediate states) are implicitly given by the predecessor main state. The labeling function $\tilde{L}_T$ maps transitions to the set of states lying on it.

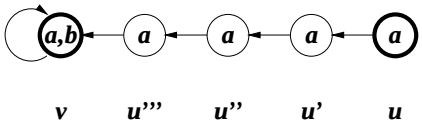


Figure 2: The derived temporal structure of figure 1

**Example 3.2** *Given the timed structure of of figure 1. The derived temporal structure (figure 2) is formally defined by:*
$\widetilde{\mathcal{M}} = (\ \{a,b\}, \{u,u',u'',u''',v\}, \{u,v\}, \{(u,v),(v,v)\}, L_H, L_T)$

*The states are partially ordered:* $u \prec u' \prec u'' \prec u'''$.
$v$ *is not compareable with either* $u, u', u'', u'''$.
*The state labeling function is as follows:*
$L_H(u) = \{a\}, L_H(v) = \{a,b\}$.
*The transition labeling function is:*
$\tilde{L}_T(u,v) = \{u,u',u'',u'''\}$,
$\tilde{L}_T(v,v) = \{v\}$, $\tilde{L}_T(u,u) = \emptyset$, $\tilde{L}_T(v,u) = \emptyset$
*Note that* $v \notin \tilde{L}_T(u,v)$ *but* $u \in \tilde{L}_T(u,v)$.

To define the new semantics of QCTL, we introduce the derived state labeling function which assigns labels to all states.

**Definition 3.3 (Derived State Labeling)**
*Given the derived temporal structure $\widetilde{\mathcal{M}}$, the derived state labeling function is:*

$$\tilde{L}_{S_\prec}(s) \quad := \quad \begin{cases} L_H(s) & \text{if } s \in H \\ L_H(u) & \text{if } u \in H \text{ and } u \prec s \end{cases}$$

Since the set of states on a transition is finite and since the ordering ($\prec$) is total, there exists a minimum state (a main state) and a maximum state for each transition chain.

**Definition 3.4 (Transition Maximum State)**
*Given a transition $(u,v) \in T$ of a derived structure $\widetilde{\mathcal{M}}$, the maximum state is*

$$\max_\prec(u,v) \quad := \quad w \in \tilde{L}_T(u,v) \text{ where no } v' \in \tilde{L}_T(u,v) \text{ exists such that } w \prec v'$$

The derived transition relation connects all states of $S$ according to $\widetilde{\mathcal{M}}$ in a unit-delay manner.

**Definition 3.5 (Derived Transition Relation)**
*Given the derived temporal structure $\widetilde{\mathcal{M}}$, the derived transition relation $\widetilde{T} \subseteq S_\prec \times S_\prec$ is:*

$$\widetilde{T} = \left\{ (u,v) \;\middle|\; \begin{array}{l} (u,v) \in T \text{ and } \tilde{L}_T(u,v) = \{u\} \text{ or} \\ u \prec v \text{ and there exists no } w \in S_\prec \\ \quad \text{such that } u \prec w \prec v \text{ or} \\ u \not\prec v \text{ and there exists } w \in H \text{ such} \\ \quad \text{that } (w,v) \in T \text{ and} \\ \quad u = \max_\prec(w,v) \end{array} \right\}$$

The first case connects two main states with no intermediate states. The second case connects the states on a transition. The third case connects the greatest intermediate state with the following main state.

**Definition 3.6 (Derived Path)**
*A derived path $\tilde{p} := (s^0, s^1, \ldots)$ of $\widetilde{\mathcal{M}}$ is a sequence of states with $\forall i \geq 0.(s^i, s^{i+1}) \in \widetilde{T}$.*

**Definition 3.7 (Semantics of QCTL)**
*Given the derived model $\widetilde{\mathcal{M}}$, a state $s$ of the model and the QCTL formulas $f$ and $g$. We write $s \models f$ for $\widetilde{\mathcal{M}}, s \models f$. The operators true, $\neg$, $\wedge$, EG and EU are defined analogously to the corresponding CTL operators.*

$s \models f \ :\Leftrightarrow\ f \in \tilde{L}_{S_\prec}(s)$ *if* $f \in AP$
$s \models \mathrm{EX}[n]f \ :\Leftrightarrow$ *there exists a derived path* $\tilde{p} = (s^0, s^1, \ldots)$ *in* $\widetilde{\mathcal{M}}$ *with* $s^n \models f$
$s \models \mathrm{EG}[n]f \ :\Leftrightarrow$ *there exists a derived path* $\tilde{p} = (s^0, s^1, \ldots)$ *in* $\widetilde{\mathcal{M}}$ *with* $\forall 0 \leq i \leq n.s^i \models f$
$s \models \mathrm{E}(f\ \mathrm{U}[n]\ g) \ :\Leftrightarrow$ *there exists a derived path* $\tilde{p} = (s^0, s^1, \ldots)$ *in* $\widetilde{\mathcal{M}}$ *with* $\exists 0 \leq i \leq n.(s^i \models g \text{ and } \forall 0 \leq j < i.(s^j \models f))$

The semantics of this QCTL variation is more intuitive, since we consider the intermediate states.

**Example 3.3** *Revisit the example 3.1 and the derived temporal structure of figure 2. We now have the following*

$$\begin{array}{lcl} \mathrm{EF}[3]a & \widetilde{=} & \{v,u',u'',u'''\} \\ \mathrm{EF}[3]\mathrm{EF}[3]a & \widetilde{=} & \{v,u,u',u'',u'''\} \\ \mathrm{EF}[6]a & \widetilde{=} & \{v,u,u',u'',u'''\} \end{array}$$

To show the flexibility of this semantics, we consider a QCTL extension with intervalls.

$s \models \mathrm{EG}[a,b]f \;:\Leftrightarrow$ there exists a path $p = (s^0, s^1, \ldots)$ such that $\forall a \leq i \leq b . s^i \models f$

This interval operator can easily be defined by the QCTL-formula: $\mathrm{EG}[a,b] := \mathrm{EX}[a]\mathrm{EG}[b-a]f$

# 4 QCTL Model Checking Algorithms

## 4.1 Representation of State Sets

Our approach is based on the observation, that main states and intermediate states can be treated differently: main states have any number of predecessor states, intermediate states have exactly one, main states have at least one successor state, intermediate states have exactly one and main states have arbitrary labels, intermediate states have the same labels as their predecessor main state.

The idea is to represent states using *extended characteristic functions*. These characteristic functions attributes transitions with state sets. In contrast to standard CTL model checking, we raise the *characteristic function* from the level of states to the level of transitions. The *extended characteristic function* maps transitions to the set of states lying on it:

$$\tilde{c}(u,v) \quad := \quad \begin{cases} \wp(\tilde{L}_T(u,v)) & \text{if } (u,v) \in T \\ \bot & \text{if } (u,v) \notin T \end{cases}$$

The state set is the union of all transition sets.

For a compact implementation, the set of states along a transition is represented by a bit vector. Every state on the transition is represented by one bit. The total ordering of states along a transition is reflected by the linear arrangement of the bits.

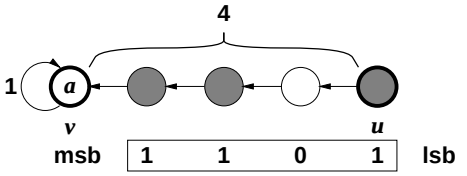


Figure 3: The encoding of states with bit vectors

**Example 4.1** *In figure 3 an example encoding of states by bit vectors is shown.*

The transitions are encoded by a pair of main states and main states are encoded by Boolean types $\mathbb{B}^n$ which are the labels carried in the states. If we assume a maximal cardinality $k$ of states on a transitions then the transitions are given by a function $\tilde{c} : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}^k$.

In order to reason on this representation, we have to define some basic set operations for extended characteristic functions. As already explained in section 3, the union of two extended characteristic functions is defined as:

$$(\tilde{c}_1 \cup \tilde{c}_2)(u,v) := \begin{cases} \tilde{c}_1(u,v) & \text{if } \tilde{c}_2(u,v) = \bot \\ \tilde{c}_2(u,v) & \text{if } \tilde{c}_1(u,v) = \bot \\ \tilde{c}_1(u,v) \cup \tilde{c}_2(u,v) & \text{otherwise} \end{cases}$$

The intersection of two extended characteristic functions can be computed analogously.

An adequate representation of QCTL state sets, i.e. extended characteristic functions, are multi terminal ROBDDs (MTBDDs). The transition encoding variables (a pair of labels) are stored in the internal nodes (variables) of the MTBDD. The resulting attribute of a transition (the set of states belonging to a transition encoded by a bit vector) is stored in the leaves of the MTBDD.

## 4.2 Representation of Timed Structures

The encoding of the transition relation of a timed structure differs from the representation of sets of states as the intermediate states do not have to be represented. We only have to know which time delay a transition has. Thus, the corresponding extended characteristic function for the transition relation is of type $\tilde{c}_{T'} : H \times H \to \mathbb{N}$. It results of the transition relation ($T$) and the transition labeling function ($L_T$) as an extended characteristic function.

However, to achieve a homogenous representation, we map from transitions to states in $S_\prec$. Since we do not consider sets of states in the transition representation, we map to the maximum state of every transition. This simplifies the model checking algorithm, as shown in the next section.

Analogous to the state set representation, we encode the maximum state by a bit vector. When using this representation in the algorithm, we interpret the most significant bit of the transition representation as the last state of the intermediate state chain. A nonexisting transition is represented by a zero bit vector $(0 \ldots 0)$.

**Example 4.2** *Figure 4 shows the MTBDDs for the set of states and the transition relation of figure 3.*

## 4.3 Algorithms

We can modify the standard CTL model checking algorithms [8]. Only the basic operators $\wedge$, $\neg$ and **EX** have to be considered for the adaption to the new data structure. The other operators can be implemented by these basic operators.

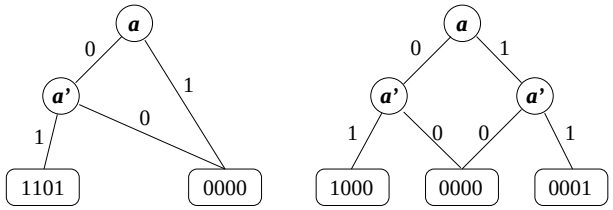


Figure 4: The MTBDDs for the set of states and the transition relation of figure 3

The result of the $\wedge$-operator can be computed by the intersection of the state sets of both operands $\tilde{c}_{f\wedge g} = \tilde{c}_f \cap \tilde{c}_g$.

The negation of a formula $f$ is true in all states of $S_\prec$ without the states holding $f$. The following formula computes the set of states for the negation:

$$\tilde{c}_{\neg f}(u,v) = \begin{cases} \tilde{c}_{S_\prec}(u,v) \setminus \tilde{c}_f(u,v) & \text{if } \tilde{c}_f(u,v) \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

The states holding EX[1]$f$ are direct predecessors of states holding $f$. In order to find the predecessors, it is necessary to distinguish between main and intermediate states. Because of the linear chains of intermediate states, it is easy to find their predecessors. With the state set representation with bit vectors we compute the predecessors by a simple shift right oeration on every MTBDD leaf.

There exists no predecessor of the least element (main state) on a transition. The predecessor states of main states are the greatest states of predecessor transitions. To extract these states out of the set of all states it is necessary to find the predecessor transitions and then extract the greatest states of them.

Extracting the predecessors of the main states is done in three steps:

1. extracting transitions with main states holding $f$
2. extracting the labels of the main states holding $f$
3. finding the predecessor transitions of the main states in 2. and extract the maximum state of this transition

These three steps can by implemented by new MTBDD operators [10].

The set for EX[$n$] is obtained by executing the EX[1] operator $n$ times. The EG[$n$] and the EU[$n$] operators use the standard fixed point iteration based on the new EX[1] operator. The terminal condition is extended by a time scope test, i.e. reaching a fixed point or exceeding the time scope terminates the algorithm.

## 5 Experimental Results

The overhead of the MTBDDs is justified mainly when dealing with large delay times. The translation of timed structures into CTL structures causes additional intermediate states. These additional states blow up the ROBDD representation of the CTL structures. On the other hand, the number of MTBDD variables in the derived temporal structure representation is independent of the number of intermediate states.

For better testing, we have chosen a scalable model. We decided to check one time property of the *priority inheritance* protocol [1, 12]. For the modeling of this protocol, we used a time slice of 60 time units. All processes were modeled together in one timed temporal structure. The property we have verified is:

$$P = \mathrm{AG}(process[max].state = try \rightarrow \mathrm{AF}[70]process[max].state = active)$$

This property insures that the process with the highest priority will enter the critical section at latest 70 time steps after the trial to enter.

For the tests (The QCTL-Model Checker is a part of the C@S-System and is on-line testable at http://goethe.ira.uka.de/hvg/cats) we used an ULTRAsparc station with 167 MHz. As model checker for the single transition model, we used SMV [9]. There we have to translate the timed structures into CTL structures. The specification formulas also have to be transformed in a non quantitative form. This is done by explicit unrolling the fixpoint iteration. For example the formula EG[3]$f$ becomes $f \wedge (\mathrm{EX}f \wedge (\mathrm{EX}f \wedge (\mathrm{EX}f)))$.

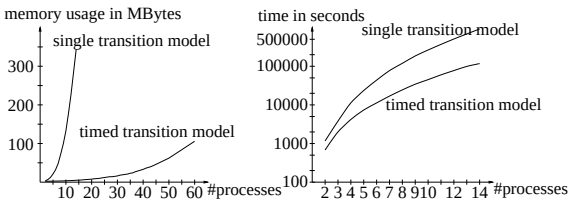


Figure 5: The usage of memory and the runtimes

Figure 5 shows the usage of memory and the runtimes for the verification of the property P.

This specific example shows, that there exist problems which are better suited for QCTL model checking than for CTL model checking. In order to extract the parameters, when problems benefit from the QCTL representation, we studied both approaches with a large number of randomly generated models of different size. The QCTL formula checked against these

models was: EX[20]$q$, with one of the atomic propositions $q$ in the model. We have chosen the EX operator, since all other temporal operator are based on it. Moreover, the EX operator is a worst case test, since the other temporal operators in QCTL can terminate before reaching the time scope.

Figure 6 compares the memory usage and the run times of both model checkers. For one point in the plane we have constructed 20 different randomly generated temporal structures and have computed the average.

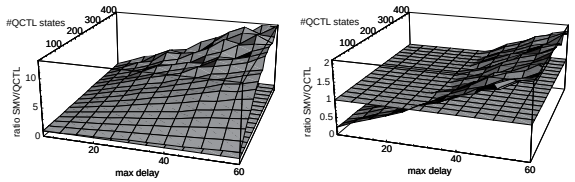


Figure 6: Relative memory requirements and run times

## 6 Conclusions

Our aim was to find a representation of timed structures which requires less memory for the verification by correctly considering of all occurrences of intermediate states. We use an extension of characteristic functions which map transitions to sets of states. With these functions we have considered all intermediate states without explicitely representing them. As data structure for these characteristic functions we have used MTBDDs. Then we have extended the basic model checking routines to the MTBDD representation.

Our experiments showed that the representation of timed models is more compact than the single transition model. The representation of timed transition graphs with MTBDDs allow to verify bigger models as in the case of single transition representation. We also showed, that the basic routines for QCTL model checking work faster if they cross a specific value for the maximal time delay.

## Acknowledgments

We are grateful to Hans Eveking, who made us aware of the counterintuitive semantics of QCTL which results if intermediate states are not considered [5].

## References

[1] S. Davari and L. Sha. Sources of unbounded priority inversion in real-time systems and a comparative study of possible solutions. In *Operating Systems Review*, pages 110–120. ACM, April 1992.

[2] M. Fujita E. Clarke and X. Zhao. Applications of multi-terminal binary decision diagrams. Technical Report CMU-CS-95-160, School of Computer Science Carnegie Mellon University, April 1995.

[3] E.Clarke, O. Grumberg, and D. Long. Verification Tools for Finite State Concurrent Systems. In de Bakker, de Roever, and Rozenberg, editors, *A Decade of Concurrency-Reflections and Perspectives*, LNCS volume 803 pages 124–175, June 1993. Springer-Verlag.

[4] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.

[5] H. Eveking. Private communication, March 1996.

[6] J. Frößl, J. Gerlach, and T. Kropf. An Efficient Algorithm for Real-Time Model Checking. In *In Proccedings of the European Design and Test Conference*, pages 15–21, Paris, March 1996.

[7] J. Lipson, editor. *Elements of Algebra and Algebraic Computing*. The Benjamin/Cummings Publishing Company, Inc., 1981.

[8] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic Model Checking: $10^{20}$ States and Beyond. In *Proceedings IEEE Symposium on Logic in Computer Science*, pages 1–33, Washington, D.C., June 1990.

[9] K.L. McMillan. The SMV system, symbolic model checking - an approach. Technical Report CMU-CS-92-131, Carnegie Mellon University, 1992.

[10] T. Kropf and J. Ruf. Using MTBDDs for Discrete Timed Symbolic Model Checking. Tech. Report, ftp://goethe.ira.uka.de/pub/techreports/SFB358-C2-5-96.ps.gz, August 1996.

[11] R. Alur, C. Courcoubetics, and D.L. Dill. Model Checking for Real-Time Systems. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 414–425, June 1990.

[12] R. Rajkumar. *Task synchronisation in real-time systems*. PhD thesis, Carnegie Mellon University, 1989.

[13] R.E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[14] S.V. Campos and E. Clarke. Real-Time Symbolic Model Checking for Discrete Time Models. In T. Rus and C. Rattray, editors, *Theories and Experiences for Real-Time System Develpment*, AMAST Series in Computing. May 1994.

[15] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic Model Checking for Real-Time Systems. In *7th. Symposium of Logics in Computer Science*, pages 394–406, Santa-Cruz, California, June 1992. IEEE Computer Scienty Press.