Reachability Analysis of Timed Automata in Polynomial Time

Dirk Beyer

Software Systems Engineering Research Group,
Technical University Cottbus,
D-03013 Cottbus, Postfach 10 13 44, Germany,
Tel. +49 (3 55) 69 - 38 02, Fax. 69 - 38 10,
{db}@informatik.tu-cottbus.de

Abstract. To develop efficient algorithms for the reachability analysis of timed automata, a promising approach is to use binary decision diagrams (BDDs) as data structure for the representation. The size of a BDD is very sensitive to the ordering of the variables. This paper shows that the communication structure of the timed automata can be used to find good variable orderings to have compact BDDs, which leads to an efficient reachability analysis. We develop a discrete semantics for closed timed automata to get a finite state space required by the BDD-based representation and we prove the equivalence to the continuous semantics regarding the set of reachable locations. An upper bound for the size of the BDD representing the transition relation and an estimation for the set of reachable configurations based on the communication structure is given within this paper. Experiments on different case studies, using a tool implementing these concepts, justify our assumption: Polynomial reachability analysis seems to be possible for some classes of models.

Keywords. Timed automata, Discretization, BDDs, Formal verification, Real-time systems

1 Introduction

The demand for correct controllers in reactive systems, especially in safety-critical systems, has more and more influence on the development process. Therefore, many developers use formal methods. Model checking, i.e. the process which checks whether a particular model satisfies a given specification or not, is commonly used for verification of automata-based models. It is very popular because the verification task is done full-automatically by tools.

In this paper we use timed automata as the formalism to describe the system and reachability analysis for the verification process. To ensure safety properties, the set of reachable configurations is computed and then it is checked whether unsafe states are reachable or not.

Reachability analysis of timed automata has been implemented in tools like Kronos [BDM+98] and UppAal [LPY97] which represent the continuous part of the model (i.e. the clock valuations) as difference bound matrices. This technique has two main disadvantages: firstly, locations are enumerated explicitly, which often results in the state explosion problem, and secondly, that there is no canonical representation for clock valuations, which often avoids efficient algorithms.

Because binary decision diagrams became very popular as a data structure for model checking of automata-based models, it is obvious to use a symbolic representation based on BDDs for the discrete states in a first step. The second step towards an efficient reachability check is to use a discrete semantics for the timed automata. Using BDDs also for the representation of the continuous state space allows a uniform representation of the discrete as well as the continuous part of the model. There already exists some experience with tool implementations of this technique, e.g. using a BDD-based version of Kronos [BMPY97].

One of the most important demands for industrial use is the efficiency of the verification process. This means that our task is to find algorithms that solve the problem with polynomial space and time complexity.

In this paper we introduce a third step leading to polynomial time and space complexity of the reachability analysis for some classes of models. We use the communication structure, and also the knowledge of the developer of the model (by providing a notation for structural modelling) to compute good variable orderings for the BDD representation. Using such variable orderings compresses the BDD representation of the reachable configurations dramatically and thus, leads to more efficient verification.

Our paper is structured as follows: Section 2 introduces the formal definition of timed automata and their continuous semantics. We also explain our notation for modular modelling. Section 3 illustrates a modular model of a MOS circuit. Section 4 introduces a discrete semantics for closed timed automata and a proof of the equivalence of both semantics regarding the reachability problem. In Section 5 we explain the impact of the communication structure on the BDD representation and its implications for finding good variable orderings. Section 6 explains the results of our experiments.

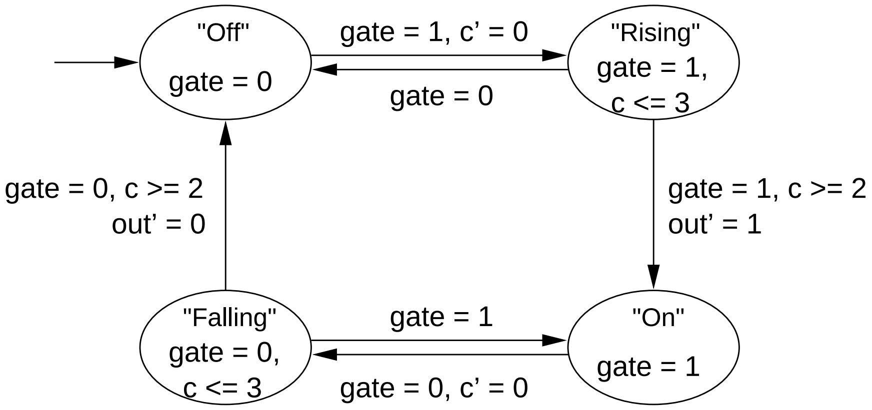


Fig. 1. Timed automaton for an nMOS transistor

2 Cottbus Timed Automata

The main goal of our modelling formalism is to combine knowledge of software engineering, i.e. hierarchical structuring of large system descriptions, and the well-investigated theoretical basis of timed automata. Thus, we use compositional modules that have well-defined interfaces and contain timed automata to describe its behaviour [BR98]. In this section we introduce timed automata informally using an example, then we introduce CTA modules which is a modelling concept providing means for modular design. After this we give a formal definition of timed automata as used in this paper.

2.1 Example

Fig. 1 shows a timed automaton which models the behaviour of an nMOS transistor. The automaton consists of four locations. Location 'Off' is the initial location of the automaton. It models the situation that the transistor is non-conducting. The transistor can stay in this situation as long as the input 'gate' is 0. When the gate becomes high the automaton takes the transition to location 'Rising' and resets clock 'c'. In this moment the transistor starts to open its channel. Location 'Rising' (as well as location 'Falling') is called unstable, because in reality the output changes during this situation. If the gate is still high after at least 2 time units the channel of the transistor can be conducting and thus, the automaton can go to location 'On' setting variable 'out' to high. The variable 'out' represents the state of the transistor. After at most 3 time units the automaton must leave location 'Rising'. A transition to location 'On' is possible, or if the gate is low meanwhile the automaton has to go to location 'Off' immediately. The automaton has an analogous behaviour for switching from conducting (out = 1) to non-conducting (out = 0).

2.2 Informal introduction to CTA modules

This section describes informally the formalism of Cottbus Timed Automata (CTA). A formal definition and the complete semantics of CTA are given in [BR99].

A CTA system description consists of a set of modules. One of them is designated as the top module. It models the whole system. The other modules are used as templates. They can be instantiated several times in different modules. Thus, it is possible to express a hierarchical structure of the system, and to define replicated components of a system just once.

Each module consists of the following components:

- An **identifier.** Identifiers are used to name the modules within the system description.
- An **interface.** The interface contains the declarations of clock variables, discrete variables and synchronization labels used by the components of the module.
  - **Synchronization labels.** Synchronization labels (shortly called signals) are used to synchronize transitions of automata contained in different modules. Synchronization labels follows the concepts of events in CSP.
  - **Variables.** Clock variables are used to model (predominantly) continuously changing components of a real-time system. Discrete variables are provided to store discrete values. The values are changed by an assignment in a transition of the automaton.

- A **timed automaton.** A module contains an automaton. This automaton consists of a finite set of states, a finite set of transitions between these states, and a alphabet of synchronization labels.
- **Initial condition.** This is a predicate over the module variables and the states of the module's automaton specifying the initial configuration.
- **Instances.** A module may contain instances of previously defined modules. This is used to model systems containing subsystems, and it is especially helpful if a subsystem occurs several times in a system. An instance consists of the following components:
  - An **identifier** is used to give a name to the instance.
  - A reference to a **module** defines which module is instantiated.
  - A **unification** of interface components of the instantiated module with declared components of the containing module defines how the instance is connected to the containing module. This may identify interface signals and interface variables of the instantiated module with signals and variables of the containing module.

2.3 Timed automata

This section gives a definition of timed automata and their continuous semantics. We use a formal definition of timed automata similar to that introduced by Alur [Alu99], because it is commonly accepted and provides a good standard.

Definition. We define **clock constraints** allowed as invariants and guards in an automaton. Let X be a set of clocks. Clock constraints over X are conjunctions of comparisons of a clock with a time constant from $\mathbb{N}$, the set of natural numbers (including 0). Formally, the following grammar generates the set of clock constraints over X:

$$\varphi := x \leq c \mid x \geq c \mid x < c \mid x > c \mid \varphi \wedge \varphi,$$

with $x \in X$ and $c \in \mathbb{N}$.

A **clock assignment** v of X is a total function from X into the set of non-negative real numbers $\mathbb{R}_+$. $Val(X)$ denotes the set of all clock assignments of X. For a clock constraint $\varphi \in \Phi(X)$, $[\![\varphi]\!]$ denotes the set of all clock assignments of X that satisfy φ.

The clock assignment which assigns the value 0 to all clocks is denoted by v^0. For $v \in Val(X)$ and $\delta \in \mathbb{R}_+$, $v + \delta$ is the clock assignment of X that assigns the value $v(x) + \delta$ to each clock x. For $v \in Val(X)$ and $Y \subseteq X$, $v[Y := 0]$ denotes the clock assignment of X that assigns the value 0 to each clock in Y and leaves the other clocks unchanged.

A **timed automaton** $\mathcal{A}$ is a tuple $(L, L^0, X, \Sigma, I, E)$, where

- L is a finite set of locations,
- $L^0 \subseteq L$ is a set of initial locations,
- X is a finite set of clocks,
- Σ is a finite set of synchronisation labels,
- I is a total function that assigns an invariant from $\Phi(X)$ to each location in L,
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$ is a set of switches. A switch (l, a, φ, Y, m) represents a transition from location l to location m labelled with synchronisation label a. The guard φ has to be satisfied to enable the location switch. The switch resets all clocks in the set Y to the value 0.

A **configuration** of a timed automaton $\mathcal{A}$ is a pair (l, v) with $l \in L$ and $v \in Val(X)$.

For a more compact notation discrete variables (which have a finite subset of the natural numbers as domain) are introduced which change their value only by location switches. Discrete variables are not considered explicitly here because they can be considered as an abbreviating notation for automata.

Semantics. The semantics of a timed automaton is defined by associating a labelled transition system with it. A **labelled transition system** $\mathcal{S}$ is a tuple $(Q, Q^0, \Sigma, \to)$ where Q is the set of configurations, $Q^0 \subseteq Q$ is a set of initial configurations, Σ is a set of labels, and $\to \subseteq Q \times \Sigma \times Q$ is a set of transitions. The system starts in an initial configuration and can change its configuration from q to q' on label a if $q \xrightarrow{a} q'$. $q \to q'$ is written if $q \xrightarrow{a} q'$ for some label a.

The **continuous semantics** $[\![\mathcal{A}]\!]_C$ of a timed automaton $\mathcal{A} = (L, L^0, X, \Sigma, I, E)$ is the labelled transition system $(L \times Val(X), L^0 \times \{v^0\}, \Sigma \cup \mathbb{R}_+, \to)$, with $\to$ containing two kinds of transitions:

- Time transitions:
  For $(l, v) \in L \times Val(X)$ and $\delta \in \mathbb{R}_+$, $(l, v) \xrightarrow{\delta} (l, v + \delta)$ if $v \in [\![I(l)]\!]$ and $v + \delta \in [\![I(l)]\!]$.

– Discrete transitions:
For $(l, v) \in L \times Val(X)$ and $(l, a, \varphi, Y, m) \in E$, $(l, v) \xrightarrow{a} (m, v[Y := 0])$ if $v \in [\![\varphi]\!]$.

Note that for all clock constraints $\varphi \in \Phi(X)$ the statements "$v \in [\![\varphi]\!]$ and $v + \delta \in [\![\varphi]\!]$" and "for all $\delta' \in \mathbb{R}$ with $0 \leq \delta' \leq \delta$, $v + \delta' \in [\![\varphi]\!]$ holds" are equivalent. This is true because only conjunctions are allowed as clock constraints.

In the following we define the runs and the reachable locations for a timed automaton $\mathcal{A} = (L, L^0, X, \Sigma, I, E)$ and a labelled transition system $\mathcal{S} = (Q, Q^0, \Sigma_{\mathcal{S}}, \rightarrow)$. Let $(q_0, q_1, ..., q_k)$ be a finite sequence of configurations and $a_0, a_1, ..., a_{k-1} \in \Sigma_{\mathcal{S}}$, such that $q_0 \in Q^0$ and $q_i \xrightarrow{a_i} q_{i+1}$ holds for all $i \in \{0, 1, ..., k-1\}$. Then $(q_0, q_1, ..., q_k)$ is a **run** of $\mathcal{A}$ with semantics $\mathcal{S}$. $Run_{\mathcal{A},\mathcal{S}}$ denotes the set of runs of $\mathcal{A}$ with semantics $\mathcal{S}$. The configuration q_k is called **reachable**. $Reach_{\mathcal{A},\mathcal{S}}$ denotes the set of reachable configurations of $\mathcal{A}$ with semantics $\mathcal{S}$. If $q_k = (l, v)$ with $l \in L$ and $v \in Val(X)$, then the location l is called **reachable**. $ReachLoc_{\mathcal{A},\mathcal{S}}$ denotes the set of reachable locations of $\mathcal{A}$ with semantics $\mathcal{S}$.

Two semantics $\mathcal{S}_1$ and $\mathcal{S}_2$ are **location-equivalent** for a timed automaton $\mathcal{A}$, iff $ReachLoc_{\mathcal{A},\mathcal{S}_1} = ReachLoc_{\mathcal{A},\mathcal{S}_2}$ holds.

The **reachability problem** for a timed automaton is the question whether for a given timed automaton $\mathcal{A}$ and a location l, $l \in ReachLoc_{\mathcal{A},[\![\mathcal{A}]\!]_C}$ holds.

Complex systems can be described as **parallel composition** of a set of timed automata which communicate through synchronisation labels.

The semantics of a composition of two timed automata $\mathcal{A}_1$ and $\mathcal{A}_2$ with disjoint sets of clocks is defined to be the semantics of the product automaton $\mathcal{A}_1 \| \mathcal{A}_2$. The locations of the product automaton are pairs of component locations, and their invariants are conjunctions of the invariants of the corresponding component locations. Two switches of different components with the same synchronisation label are synchronised. We define formally:

Let $\mathcal{A}_1 = (L_1, L_1^0, X_1, \Sigma_1, I_1, E_1)$ and $\mathcal{A}_2 = (L_2, L_2^0, X_2, \Sigma_2, I_2, E_2)$ be two timed automata, and assume that $X_1 \cap X_2 = \emptyset$. The product automaton $\mathcal{A}_1 \| \mathcal{A}_2$ is the timed automaton $(L_1 \times L_2, L_1^0 \times L_2^0, X_1 \cup X_2, \Sigma_1 \cup \Sigma_2, I, E)$ with $I(l_1, l_2) = I_1(l_1) \wedge I_2(l_2)$ and E defined as the set of the following switches:

– for $a \in \Sigma_1 \cap \Sigma_2$, for every $(l_1, a, \varphi_1, Y_1, m_1) \in E_1$ and $(l_2, a, \varphi_2, Y_2, m_2) \in E_2$ we have $((l_1, l_2), a, \varphi_1 \wedge \varphi_2, Y_1 \cup Y_2, (m_1, m_2)) \in E$,
– for $a \in \Sigma_1 \setminus \Sigma_2$, for every $(l_1, a, \varphi_1, Y_1, m_1) \in E_1$ and $l_2 \in L_2$ we have $((l_1, l_2), a, \varphi_1, Y_1, (m_1, l_2)) \in E$,
– for $a \in \Sigma_2 \setminus \Sigma_1$, for every $(l_2, a, \varphi_2, Y_2, m_2) \in E_2$ and $l_1 \in L_1$ we have $((l_1, l_2), a, \varphi_2, Y_2, (l_1, m_2)) \in E$,
– these are all transitions.

3 Examples: CTA models of a mutex protocol and a MOS circuit

In this section we introduce CTA models of two examples: Fischer's timing-based protocol for mutual exclusion for n processes [Lam87], which serves as benchmark in many publication and an AND circuit with 4 input lines. A model of this circuit using plain timed automata is used by [BMPY97] for the tool Kronos. At the end of the paper we use this example for verification and comparison with Kronos.

Fischer's protocol. The model is composed from n timed automata like the one depicted in Figure 2, each modelling one process. Each component automaton has four locations. 'Uncritical' is the initial location and represents the uncritical section of the process. The shared discrete variable k is initialised with the value 0. From this location only one transition is possible: If the shared variable ensures that no other process tries to enter the critical section ($k = 0$), the process can move to the location 'Assign'. This location expresses that a process needs at most a time units to complete the assignment $k := i$. Therefore, the clock x_i measures the staying time in this location, and the invariant forces the automaton to leave the location within a time units. Then the transition to the 'Wait' location sets the variable k to process identifier i. In this location the process has to wait at least b time units to guarantee that all other processes completed the assignment. After b time units it is allowed to enter the critical section if $k = i$. Otherwise it goes back to the uncritical section. Leaving the critical section the automaton sets k to value 0 to signify that the resource is free again.

MOS circuit. The main module modelling the behaviour of the AND gate consists of two module instances of a NAND gate and one module instance of a NOR gate as shown in the first part of Fig. 3. The environment of the AND gate is modelled by a clock 'p' for the time cycles and four variables to model the four input lines. The clock has the initial value 0, and when the value 15 is reached the automaton 'resetP' resets the clock to 0. (Automata are drawn as graph within a circle.) A module 'Input' consists of an automaton which can change the value of one of the input variables once during the first five clock ticks of 'p'. During the last ten clock ticks the input signals stay unchanged (they are stable). The binary variables 'o1' and 'o2' represent the output values of the two NAND gates and 'o' models the output of the NOR gate and thus the output of the whole circuit.

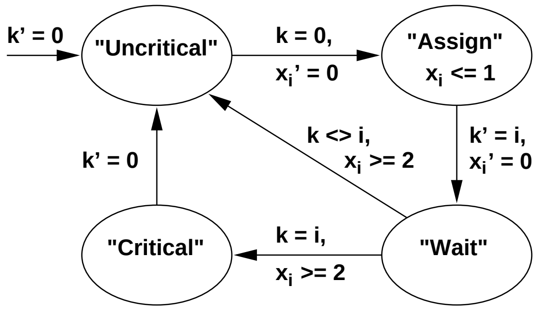


Fig. 2. Fischer's mutual exclusion protocol

The second part of Fig. 3 shows the structure of the module for NAND gates (named 'Nand' in the figure). It consists of two pMOS transistors and two nMOS transistors. Reading the variable 'out' (conducting or non-conducting) of the transistors (connected to this module as 'oP1', 'oN1', 'oP2' and 'oN2') the automaton (named 'nand' in the figure) determines the output of the NAND gate. The behaviour of the module for NOR gates is analogous.

A module for an nMOS transistor contains a clock 'c' and an automaton. An nMOS transistor takes between 2 and 3 time units to change the output after a change of the gate is detected as shown in Fig. 1. The only difference to pMOS transistors is the inverse output value and that the pMOS transistors have to react at least 4 time units after a change at the gate.

Interesting questions about the behaviour of the AND gate are for example: How many transistors switch their state together at the same point in time? (This number is proportional to the maximum of current needed by the gate.) Is it possible that a short circuit occurs in the AND gate? To answer these questions we have to compute the reachable configurations of the model. In Section 6 we show only results for the computation of reachable configurations because this the bottleneck of the reachability analysis.

The real-valuedness of the clocks leads to an infinite state space. Therefore, we use a discretization of the continuous state space as a requirement to get a finite state space. We give a formal definition of a discrete semantics in the next section.

4 Discretization

The discretization of time is possible for all timed automata [GPV94]. However, in the following we restrict ourselves to the subclass of closed timed automata to permit a discretization which is particularly simple and which allows very efficient reachability analysis. **Closed timed automata** have only clock constraints φ generated by $\varphi := x \leq c \mid x \geq c \mid \varphi \wedge \varphi$ with $x \in X$ and $c \in \mathbb{N}$, i.e. the relations $<$ and $>$ are not allowed. The product automaton of two closed timed automata is closed again.

For closed timed automata it is sufficient to use integer clock values for the computation of reachable locations. For a set of clocks X the set of integer clock assignments $Val_I(X)$ is defined to be the set of total functions from X to $\mathbb{N}$. Let $C_{\mathcal{A}}(x)$ be the greatest constant occurring in some expression constraining the variable x. For $v \in Val_I(X)$ and $\delta \in \mathbb{N}$, $v \oplus \delta$ is the clock assignment of X that assigns the value $min\,(v(x) + \delta, C_{\mathcal{A}}(x) + 1)$ to each clock x. The definition of the discrete semantics is analogous to the continuous semantics previously defined.

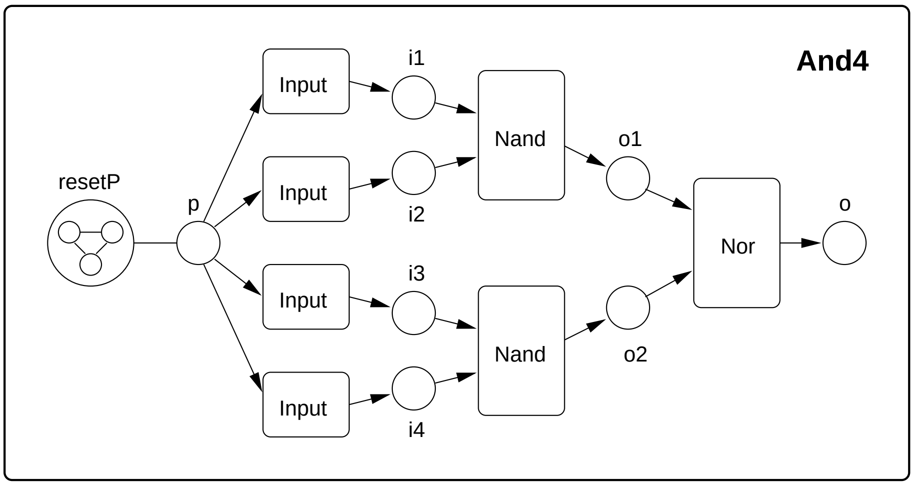
Let $\mathcal{A} = (L, L^0, X, \Sigma, I, E)$ be a closed timed automaton. The **discrete semantics** $[\![\mathcal{A}]\!]_I$ of $\mathcal{A}$ is the labelled transition system $(L \times Val_I(X), L^0 \times \{v^0\}, \Sigma \cup \mathbb{N}, \rightarrow_I)$ with the following transitions:

– For $(l, v) \in L \times Val_I(X)$ and $\delta \in \mathbb{N}$, $(l, v) \xrightarrow{\delta}_I (l, v \oplus \delta)$ if $v \in [\![I(l)]\!]$ and $v \oplus \delta \in [\![I(l)]\!]$.
– For $(l, v) \in L \times Val_I(X)$ and $(l, a, \varphi, Y, m) \in E$, $(l, v) \xrightarrow{a}_I (m, v[Y := 0])$ if $v \in [\![\varphi]\!]$.

To prove the *location equivalence* of discrete and continuous semantics, we define for a set of clocks X the relation $\succ \subseteq Val(X) \times Val_I(X)$ associating every continuous clock assignment with its possible discrete **representatives**. For $v \in Val(X)$ and $v' \in Val_I(X)$, $v \succ v'$ holds iff there exists some $\gamma \in \mathbb{R}$ with $0 \leq \gamma < 1$, such that for each clock $x \in X$:

a) $v'(x) - 1 + \gamma < v(x) \leq v'(x) + \gamma$, or
b) $v'(x) - 1 + \gamma < v(x)$ and $v'(x) = C_{\mathcal{A}}(x) + 1$.

Thus, v' is a representative of v if v' results from v by rounding off all clock values with fractional parts smaller than or equal to a certain bound and by rounding up all clock values with fractional parts greater than this bound



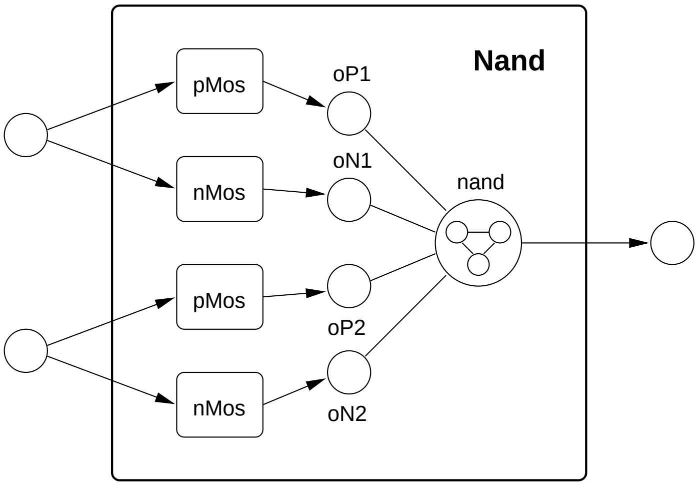


Fig. 3. Model of the AND circuit

in the first case. The second case restricts the range of the representatives to the greatest constant $C_{\mathcal{A}}(x)+1$; this is sufficient to distinguish the interesting situations.

Theorem 1. *Let $\mathcal{A}$ be a closed timed automaton with the set of clocks X and let $v \in Val(X)$ and $w \in Val_I(X)$ be clock assignments with $v \succ w$.*

1. *If v satisfies a clock constraint φ of $\mathcal{A}$, then w also satisfies φ.*
2. *For all $Y \subseteq X$, $v[Y := 0] \succ w[Y := 0]$.*

Proofs of the *location equivalence* of the discrete semantics and the continuous semantics for other formalisms than timed automata can be found in [HMP92] and [AMP98].

Lemma 2. *Let $\mathcal{A} = (L, L^0, X, \Sigma, I, E)$ be a closed timed automaton with the continuous semantics $[\![\mathcal{A}]\!]_C = (L \times Val(X), L^0 \times \{v^0\}, \Sigma \cup \mathbb{R}_+, \rightarrow_C)$ and the discrete semantics $[\![\mathcal{A}]\!]_I = (L \times Val_I(X), L^0 \times \{v^0\}, \Sigma \cup \mathbb{N}, \rightarrow_I)$. Then the following holds:*

1. *Let $(l,v'), (l,w') \in L \times Val_I(X), \delta' \in \mathbb{N}$, such that $(l,v') \xrightarrow{\delta'}_I (l,w')$ holds. Then for all $v \in Val(X)$ with $v \succ v'$ there exists a $w \in Val(X)$, such that $(l,v) \xrightarrow{\delta'}_C (l,w)$ and $w \succ w'$ holds.*
2. *Let $(l,v'), (m,w') \in L \times Val_I(X), a \in \Sigma$, such that $(l,v') \xrightarrow{a}_I (m,w')$ holds. Then for all $v \in Val(X)$ with $v \succ v'$ there exists a $w \in Val(X)$, such that $(l,v) \xrightarrow{a}_C (l,w)$ and $w \succ w'$ holds.*
3. *Let $(l,v), (l,w) \in L \times Val(X), \delta \in \mathbb{R}_+$, such that $(l,v) \xrightarrow{\delta}_C (l,w)$ holds. Then for all $v' \in Val_I(X)$ with $v \succ v'$ there exists a $\delta' \in \mathbb{N}$ and a $w' \in Val_I(X)$, such that $(l,v') \xrightarrow{\delta'}_I (l,w')$ and $w \succ w'$ holds.*
4. *Let $(l,v), (m,w) \in L \times Val(X), a \in \Sigma$, such that $(l,v) \xrightarrow{a}_C (m,w)$ holds. Then for all $v' \in Val_I(X)$ with $v \succ v'$ there exists a $w' \in Val_I(X)$ such that $(l,v') \xrightarrow{a}_I (m,w')$ and $w \succ w'$ holds.*

Proof. The statements 1 and 2 follow from the definitions of the semantics.

Statement 3: We have to distinguish the two cases of the definition of $\succ$. Let $v' \in Val_I(X)$, $v \succ v'$. Then according to the definition of the relation $\succ$ there exists some $\gamma \in \mathbb{R}$ with $0 \leq \gamma < 1$, such that the following holds for all $x \in X$:

Case a) $v'(x) + \gamma + \delta < C_{\mathcal{A}}(x) + 1$:

$$v'(x) - 1 + \gamma < v(x) \quad \leq v'(x) + \gamma.$$

Because $w = v + \delta$, the following holds for all $x \in X$:

$$v'(x) - 1 + \gamma + \delta < w(x) \quad \leq v'(x) + \gamma + \delta.$$

Let $\delta' = \lfloor \delta + \gamma \rfloor$ and $w' = v' + \delta'$. Then for all $x \in X$ the following holds:

$$w'(x) - \delta' - 1 + \gamma + \delta < w(x) \quad \leq w'(x) - \delta' + \gamma + \delta.$$

Because $0 \leq \gamma + \delta - \delta' < 1$, this implies $w \succ w'$.

Case b) $v'(x) + \gamma + \delta \geq C_{\mathcal{A}}(x) + 1$:

Using $\delta' = \lfloor \delta + \gamma \rfloor$ and $w' = v' \oplus \delta'$, analogously to Case 1 we obtain:

$$w'(x) - \delta' - 1 + \gamma + \delta < w(x),$$

and thus, $w \succ w'$.

Because v and w satisfy the invariant $I(l)$ and $v \succ v'$ and $w \succ w'$ hold, we can conclude from Theorem 1, statement 1 that v' and w' satisfy the invariant $I(l)$. Thus, we get $(l, v') \xrightarrow{\delta'}_I (l, w')$.

Statement 4 follows from Theorem 1. $\square$

Theorem 3. *For every closed timed automaton $\mathcal{A}$, $ReachLoc_{\mathcal{A},[\![\mathcal{A}]\!]_C} = ReachLoc_{\mathcal{A},[\![\mathcal{A}]\!]_I}$ holds.*

Proof. Let $\mathcal{A} = (L, L^0, X, \Sigma, I, E)$ be a timed automaton with the continuous semantics $[\![\mathcal{A}]\!]_C = (L \times Val(X), L^0 \times \{v^0\}, \Sigma \cup \mathbb{R}_+, \rightarrow_C)$ and the discrete semantics $[\![\mathcal{A}]\!]_I = (L \times Val_I(X), L^0 \times \{v^0\}, \Sigma \cup \mathbb{N}, \rightarrow_I)$.

At first, we prove $ReachLoc_{\mathcal{A},[\![\mathcal{A}]\!]_C} \subseteq ReachLoc_{\mathcal{A},[\![\mathcal{A}]\!]_I}$. We show per induction over k that for every run $((l_0, v_0), (l_1, v_1), ..., (l_k, v_k))$ in $Run_{\mathcal{A},[\![\mathcal{A}]\!]_C}$ there exists a run $((l_0, v'_0), (l_1, v'_1), ..., (l_k, v'_k))$ in $Run_{\mathcal{A},[\![\mathcal{A}]\!]_I}$, such that $v_i \succ v'_i$ holds for all $i \in \{0, 1, ..., k\}$.

Start of induction: According to the definition of run, $l_0 \in L^0$ and $v_0 = v^0$ hold. $((l_0, v^0))$ is also in $Run_{\mathcal{A},[\![\mathcal{A}]\!]_I}$, and $v_0 \succ v^0$ holds.

Inductive step: We have to show that there exists some $a' \in \Sigma \cup \mathbb{N}$ and some v'_{i+1} with $v_{i+1} \succ v'_{i+1}$, such that $(l_i, v'_i) \xrightarrow{a'}_I (l_{i+1}, v'_{i+1})$. The inductive hypothesis ensures $v_i \succ v'_i$ and there exists some $a \in \Sigma \cup \mathbb{R}_+$ with $(l_i, v_i) \xrightarrow{a}_R (l_{i+1}, v_{i+1})$. The assertion of the theorem follows from the claim of Lemma 2, statement 3, if $a \in \mathbb{R}_+$, and of statement 4, if $a \in \Sigma$. This finishes the inductive proof.

Similarly, we can show using statements 1 and 2 of Lemma 2: $ReachLoc_{\mathcal{A},[\![\mathcal{A}]\!]_I} \subseteq ReachLoc_{\mathcal{A},[\![\mathcal{A}]\!]_C}$. $\square$

5 Efficient verification using the structure of the model

Because the number of locations of a product automaton grows exponentially in the number of processes, the state explosion problem forces typically the use symbolic representation of the locations. The technique of representing sets of states as binary decision diagrams is in widespread use and also implemented in our tool. The second step to efficient verification is to use a finite set of configurations for the reachability analysis by introducing a discrete semantics. The discretization enables a unique representation of the set of configurations consisting of locations of the automata together with the discretized continuous state space of clocks. This technique is also examined in [BMPY97].

In this section we introduce a new technique for efficient verification of some classes of models. We use a variable ordering resulting from the communication structure of a system and we determined empirically the polynomial complexity of the reachability analysis of some classes of models. We prove the polynomial size of the representation of the transition relation. Because of our empirical studies we think that it is sound to infer from these results a size estimation for the representation of the set of reachable configurations. We use this estimation as a qualitative assessment of different variable orderings.

5.1 Communication graph and variable ordering

Aziz et al. proved an upper bound for the size of BDDs for transition relations of communicating finite automata [ATB94]. On the basis of this upper bound they determine good variable orderings for the set of reachable locations. In this section we use the results of that work to explain the characteristics of good variable orderings for timed automata.

The problem is to find a variable ordering for a given parallel composition of the timed automaton $\mathcal{A}$ such that the number of nodes of the BDD representation of $Reach_{\mathcal{A},[\![\mathcal{A}]\!]_I}$ is as small as possible. For this purpose we investigate the communication between the components. Two components $\mathcal{A}_j$ and $\mathcal{A}_k$ with $j,k \in \{1,...,n\}$, are **communicating**, symbolically $\mathcal{A}_j \rightleftharpoons \mathcal{A}_k$, iff $\Sigma_j \cap \Sigma_k \neq \emptyset$ and $j \neq k$. Considering the components as nodes and the communication relation $\rightleftharpoons$ as set of edges we get the **communication graph**.

We use simple examples to illustrate two general characteristics of good variable orderings. We consider three finite automata (i.e. timed automata without clocks) $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$, each having the locations l_1, l_2 and l_3.

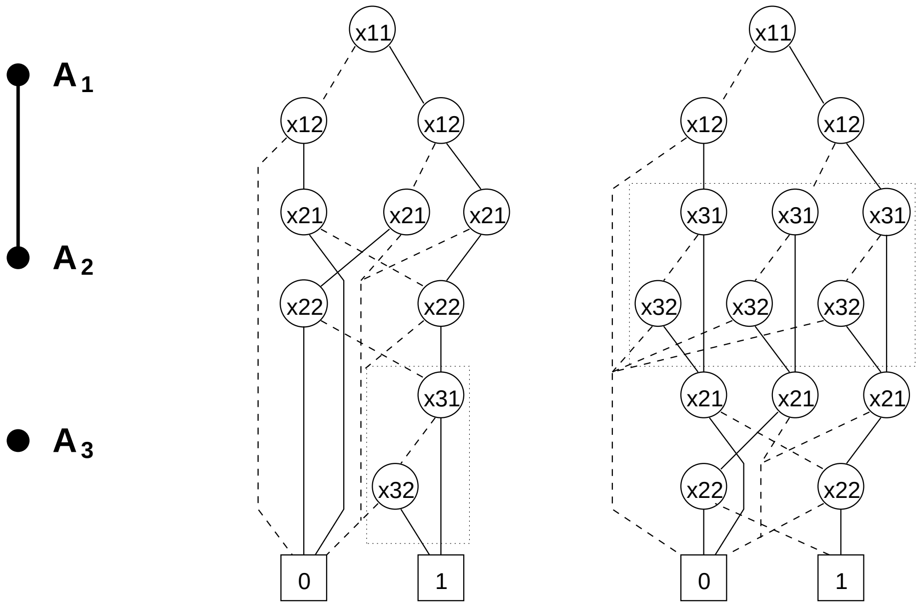


Fig. 4. Communication graph and variable ordering, example 1.

In the first example $\mathcal{A}_3$ communicates neither with $\mathcal{A}_1$ nor with $\mathcal{A}_2$. $\mathcal{A}_1$ and $\mathcal{A}_2$ ensure by communication that they stay in the same location every time. Let x_{i1} and x_{i2} encode the configuration of $\mathcal{A}_i$. Fig. 4 shows the communication graph and the BDDs of the reachable locations for the variable ordering $(x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32})$ on the left side and $(x_{11}, x_{12}, x_{31}, x_{32}, x_{21}, x_{22})$ on the right side. It illustrates the first characteristic of a good variable ordering: Communicating components have successive positions within the ordering.

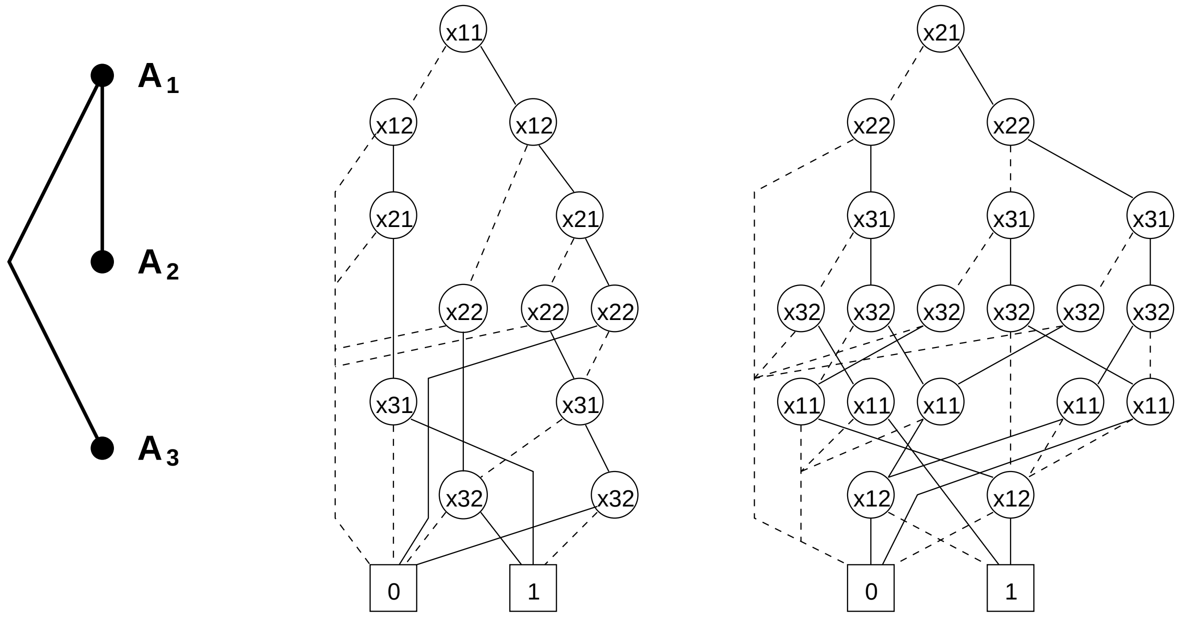


Fig. 5. Communication graph and variable ordering, example 2.

In the second example $\mathcal{A}_1$ communicates with $\mathcal{A}_2$ and $\mathcal{A}_3$, and $\mathcal{A}_2$ does not communicate with $\mathcal{A}_3$. It is ensured by communication that $\mathcal{A}_1$ and $\mathcal{A}_2$ as well as $\mathcal{A}_1$ and $\mathcal{A}_3$ stay in different locations every time. Fig. 5 shows the communication graph and the BDDs of the reachable locations for the variable orderings $(x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32})$ and $(x_{21}, x_{22}, x_{31}, x_{32}, x_{11}, x_{12})$. Both orderings do not differ with respect to the first characteristic, but the sizes of their BDDs are different. The second characteristic of good variable orderings becomes obvious: Components which communicate with many other components are at the beginning of the ordering.

To derive an algorithm for finding variable orderings we first show an upper bound of the BDD's size of a transition relation of the product automaton. The only bottleneck in our algorithms is the size of the BDD for the reachable configurations, because we do not compute the monolithic transition relation (instead we use partitioned transition relations represented by very small BDDs). But we need the upper bound for the transition relation, because good variable orderings for the transition relation are often good for the set of reachable configurations. Therefore, we derive an estimation for the size of the set of reachable configurations from the upper bound for the size of the transition relation. An algorithm for finding a good variable ordering searches for a variable ordering having a low size estimation.

To justify this argumentation we refer to results of other research groups: The result of [ATB94] is that there is a good correlation between the BDD size predicted by the bound and the actual BDD size for the transition relation. Experiments in [YBO+98] show that good variable orderings for the transition relation are also good for the set of reachable locations. In [ATB94] as well as in this paper it is demonstrated by empirical studies that this assumption is correct for many classes of systems. However, there are counterexamples with linear growth of BDDs for the transition relation but exponential growth of BDDs for the reachable configurations [McM92].

For the purpose of finding good variable orderings it is not necessary that the estimated size is absolutely close to the real size because the estimation should only reflect the relation between different variable orderings, i.e. good variable orderings should lead to better estimations than bad variable orderings. Last, not least upper bound for the BDD's size in the next section and the algorithms have the advantage that they correspond to both of the characteristics mentioned in this section, and these characteristics reflect the experience and intuition of many experts.

5.2 Upper bound for the BDD's size

In this section we prove the upper bound for the number of nodes of the BDD for the transition relation. We start with an introduction of some conventions and notations. In this section we adapt the notations introduced in Section 2.3 to be able to represent assignments by BDDs.

Range(x) is used to denote the range of a discrete variable x. Boolean variables are special discrete variables with the range $\{0, 1\}$. Let X be a set of discrete variables. The set $\Phi(X)$ of **constraints** φ is generated by the following grammar: $\varphi := x_1 \sim c \mid x_1 \sim x_2 \mid \varphi \wedge \varphi$, with $x_1, x_2 \in X$, $\sim\ \in \{\leq, \geq, <, >, =\}$, $Range(x_1) = Range(x_2)$ and $c \in Range(x_1)$.

An **assignment** v of X is a total function which assigns an element of $Range(x)$ to each variable x. The set of all assignments of X is denoted by $Val(X)$. The set of all assignments of X that satisfy a constraint $\varphi \in \Phi(X)$ is denoted by $[\![\varphi]\!]$. φ is related to different sets of variables (because e.g. $x > 5$ is a constraint for both {x} and {x, y}). Therefore, we identify two sets of assignments $V \subseteq Val(X)$ and $W \subseteq Val(X \cup Y)$ iff $W = \{w \in Val(X \cup Y) | \exists v \in Val(X)\ \forall x \in X : w(x) = v(x)\}$ holds.

Now we introduce some notations for a set of assignments $V \subseteq Val(X)$.

- For a variable $x \in X$, the **existential quantification** $\exists x.V$ is defined as set of all assignments of $X \setminus \{x\}$ with the values of all variables but x equal to the values of the same variables in an assignment in V; formally: for $w \in Val(X \setminus \{x\})$, $w \in \exists x.V$ holds iff there exists some $v \in Val(V)$, such that $v(y) = w(y)$ for all $y \in X \setminus \{x\}$.
- For two variables $x \in X$ and $y \notin X$, $V[x \leftarrow y]$ is the set of assignments which is obtained by **renaming** x by y; formally: for a $w \in Val((X \setminus \{x\}) \cup \{y\})$, $w \in V[x \leftarrow y]$ holds iff there exists a $v \in V$, such that $v(x) = w(y)$ and $v(z) = w(z)$ for all $z \in X - \{x\}$.
- For a variable $x \in X$ and a constant $c \in Range(x)$ the **cofactor** $V|_{x=c}$ is defined as $\exists x.(V \cap [\![x = c]\!])$.

A finite relation can be represented by a set of assignments by mapping the arguments of the relation to discrete variables. Let $R \subseteq R_1 \times R_2 \times ... \times R_n$ be a relation and $X = \{x_1, x_2, ..., x_n\}$ a set of discrete variables with $Range(x_i) = R_i$ for all $i \in \{1, 2, ..., n\}$. Then $R(x_1, x_2, ..., x_n)$ denotes the set of assignments of X with $v \in R(x_1, x_2, ..., x_n)$ iff $(v(x_1), v(x_2), ..., v(x_n)) \in R$.

The algorithm for computing all reachable configurations of a parallel composition of n timed automata is shown in Fig. 6. We use the abbreviations introduced in the figure also in the sequel of the paper. We deal only with closed timed automata and the integer semantics as introduced in the previous section. Transfering the results to other discretizations is possible.

Input: parallel composition $\mathcal{A} = \{L, L^0, X, \Sigma, I, E\}$
 with the discrete semantics $[\![\mathcal{A}]\!]_I = (Q, Q^0, \Sigma \cup \mathbb{N}, \rightarrow)$
 of closed timed automata $\mathcal{A}_i = (L_i, L_i^0, X_i, \Sigma_i, I_i, E_i), i \in \{1, ..., n\}$
 with the discrete semantics $[\![\mathcal{A}_i]\!]_I = (Q_i, Q_i^0, \Sigma_i \cup \mathbb{N}, \rightarrow_i)$
 and disjoint sets of clocks: $X_j \cap X_k = \emptyset$ for all $j, k \in \{1, ..., n\}$ with $j \neq k$
Output: $Reach_{\mathcal{A},[\![\mathcal{A}]\!]_I}(q_1, ..., q_n)$,
 with variable q_i corresponding to the configuration $\mathcal{A}_i$ ($Range(q_i) = Q_i$)

$R := Q^0(q_1, \ldots, q_n)$
do
 $R_{prev} := R$
 forall $a \in (\Sigma \cup \{1\})$
 $R := R \cup \big(\exists q_1 ... \exists q_n (R \ \cap \ \xrightarrow{a}(q_1, q_1', ..., q_n, q_n')) \big) [q_1' \leftarrow q_1] ... [q_n' \leftarrow q_n]$
while $R \neq R_{prev}$
return R

Fig. 6. Computation of the set of reachable configurations.

For the proof we need a formal definition for BDDs. The following definition is similar to the one from McMillan [McM92]. A BDD is identified with its root node. Let $\vec{x}$ be a vector $(x_1, x_2, ..., x_n)$ of Boolean variables. If $n = 0$, then $\mathcal{B}$ is a **binary decision diagram** over $\vec{x}$ iff $\mathcal{B}$ is the 0-terminal-node (short $\mathcal{B} = 0$) or $\mathcal{B}$ is the 1-terminal-node (short $\mathcal{B} = 1$). If $n > 0$ then $\mathcal{B}$ is a BDD over $\vec{x}$ iff

- $\mathcal{B}$ is a BDD over $(x_2, ..., x_n)$, or
- $\mathcal{B} = (x_1, \mathcal{B}_0, \mathcal{B}_1)$, where $\mathcal{B}_0$ and $\mathcal{B}_1$ are BDDs over $(x_2, ..., x_n)$. $\mathcal{B}$ is called an x_n-node, $\mathcal{B}_0$ is called low child, and $\mathcal{B}_1$ is called high child of $\mathcal{B}$.

A BDD $\mathcal{B}$ over $(x_1, x_2, ..., x_n)$ represents a set of assignments of $\{x_1, x_2, ..., x_n\}$ which are denoted by $[\![\mathcal{B}]\!]$ and defined as follows:

$$[\![\mathcal{B}]\!] = \begin{cases} \emptyset, & \text{if } \mathcal{B} = 0 \\ Val(\{x_1, x_2, ..., x_n\}), & \text{if } \mathcal{B} = 1 \\ ([\![\mathcal{B}_0]\!] \cap [\![x_i = 0]\!]) \cup ([\![\mathcal{B}_1]\!] \cap [\![x_i = 1]\!]), & \text{if } \mathcal{B} = (x_i, \mathcal{B}_0, \mathcal{B}_1) \end{cases}$$

In the sequel we consider only BDDs which are generated by applying only the following rule: Identify all equal subtrees. We do not eliminate nodes with two edges to the same sub-node. The number of nodes of such a BDD is an upper bound for the number of nodes after applying the second rule. The first rule has more impact on the reduction of the BDD and is more sensitive for the variable ordering than the second rule.

Theorem 4. *Let $\mathcal{B}$ be a BDD over the vector $(x_1, x_2, ..., x_k)$ of Boolean variables and $i \in \{1, ..., k-1\}$. Then the number of x_{i+1} nodes in $\mathcal{B}$ is less than or equal to twice the number of x_i nodes.*

Let $\mathcal{B}$ be a BDD over $(q_1, q_1', ..., q_n, q_n')$, $i \in \{1, ..., n\}$ and x the Boolean variable which is the first in the variable ordering of the variables encoding q_i. Then $|\mathcal{B}|_i$ is used to denote the number of x-nodes in $\mathcal{B}$ and $|\mathcal{B}|_{n+1}$ is used to denote the number of terminal nodes in $\mathcal{B}$. $|\mathcal{B}|$ is the number of all non-terminal nodes in $\mathcal{B}$. For an assignment $V \subseteq Val(\{q_1, q_1', ..., q_n, q_n'\})$ and a set of variables $M = \{q_{i_1}, q_{i_1}', ..., q_{i_k}, q_{i_k}'\}$ $(i_1, ..., i_k \in \{1, ..., n\})$, $V|_M$ denotes the set of all cofactors of V regarding the variables in M, i.e. $V|_M = \{V|_{q_{i_1}=c_1, q_{i_1}'=c_1', ..., q_{i_k}=c_k, q_{i_k}'=c_k'} \mid c_l, c_l' \in Q_{i_l}$ for all $l \in \{1, ..., k\}\}$. As abbreviating notation $V|_i$ denotes $V|_{\{q_1, q_1', ..., q_{i-1}, q_{i-1}'\}}$ for $i \in \{1, ..., n+1\}$. For a set M, $|M|$ is used to denote the number of elements of M.

Theorem 5. *Let $\mathcal{B}$ be a BDD over $(q_1, q_1', ..., q_n, q_n')$ and $V \subseteq Val(\{q_1, q_1', ..., q_n, q_n'\})$ be a set of assignments with $[\![\mathcal{B}]\!] = V$. Then $|\mathcal{B}|_i \leq |\ V|_i \cup \{\emptyset\}\ |$ holds for all $i \in \{1, ..., n+1\}$.*

Note: Let $x_1, ..., x_s$ be the Boolean variables encoding $q_1, q_1', ..., q_{i-1}, q_{i-1}'$. If there exists an assignment of $\{x_1, ..., x_s\}$ in $V|_i$ which is not an encoding of an assignment of $\{q_1, q_1', ..., q_{i-1}, q_{i-1}'\}$, then we have to consider the empty set as additional cofactor. This is the case if the cardinality of a set Q_k $(k \in \{1, ..., i-1\})$ is not a power of two. Otherwise $|\mathcal{B}|_i = |\ V|_i\ |$.

From the number of cofactors of an assignment we can infer the size of its BDD representation. An upper bound for the number of cofactors of the transition relation is given by the following theorem. The time transitions are taken synchronously for the clocks in all components. We would get additional edges in the communication

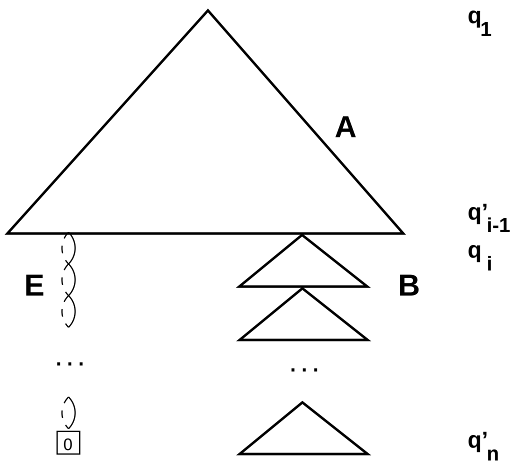


Fig. 7. BDD for Case 1.

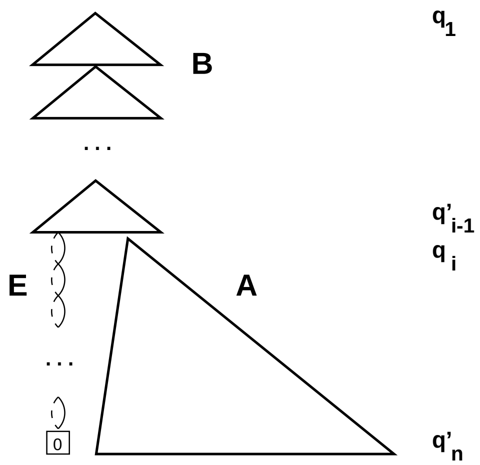


Fig. 8. BDD for Case 2.

graph connecting all automata having a clock. This does not give any hint for the variable ordering, and thus, we do not consider them here. In the sequel we consider only the relation of discrete transitions $\rightarrow' = \bigcup_{a\in\Sigma} \xrightarrow{a}$. The set $Comm_{\mathcal{A}}(i)$ contains the indices of all components of $\mathcal{A}$ which have an index less than i and communicate with a component having an index greater than or equal to i: $Comm_{\mathcal{A}}(i) = \{k \mid k < i$ and there exists an $l \geq i$ with $\mathcal{A}_k \rightleftharpoons \mathcal{A}_l\}$.

Theorem 6. *For the transition relation $\rightarrow'(q_1, q_1', ..., q_n, q_n')$ and every $i \in \{1, ..., n+1\}$, the following holds:*

$$\left|\rightarrow'(q_1, q_1', ..., q_n, q_n')|_i \;\cup\; \{\emptyset\}\right| \;\leq\; 4 \cdot \prod\nolimits_{k\in Comm_{\mathcal{A}}(i)} |Q_k|^2 + 4$$

Proof. At first we give a lemma used in our computation of the number of cofactors. For all $V, W \subseteq Val(\{q_1, q_1', ..., q_n, q_n'\})$, $i_1, ..., i_k \in \{1, ..., n\}$ and $c_l, c_l' \in Q_{i_l}$ $(l \in \{1, ..., k\})$ the following holds:

$$\begin{aligned}(V \cap W)|_{q_{i_1}=c_1, q'_{i_1}=c'_1, ..., q_{i_k}=c_k, q'_{i_k}=c'_k} \\ = V|_{q_{i_1}=c_1, q'_{i_1}=c'_1, ..., q_{i_k}=c_k, q'_{i_k}=c'_k} \;\cap\; W|_{q_{i_1}=c_1, q'_{i_1}=c'_1, ..., q_{i_k}=c_k, q'_{i_k}=c'_k}\end{aligned} \tag{1}$$

Equation 1 analogously holds for the union of sets of assignments.

We partition the transition relation $\rightarrow'$ into three subsets. From its cofactors we can conclude the cofactors of $\rightarrow'$ applying equation 1.

Case 1. Discrete transitions concerning only the components $\mathcal{A}_1$ to $\mathcal{A}_{i-1}$:

$$\begin{aligned}
&\bigcup\nolimits_{a\in\Sigma\setminus(\Sigma_i\cup...\cup\Sigma_n)} \xrightarrow{a} \\
&= \bigcup\nolimits_{a\in\Sigma\setminus(\Sigma_i\cup...\cup\Sigma_n)} \bigcap\nolimits_{k\in\{1,2,...,n\}} \begin{cases} \xrightarrow{a}_k (q_k, q_k'), \text{ if } a \in \Sigma_k \\ [\![q_k' = q_k]\!], \quad \text{otherwise} \end{cases} \\
&= \bigcup\nolimits_{a\in\Sigma\setminus(\Sigma_i\cup...\cup\Sigma_n)} \left(\begin{array}{l} \quad\bigcap\nolimits_{k\in\{1,...,i-1\}} \begin{cases} \xrightarrow{a}_k (q_k, q_k'), \text{ if } a \in \Sigma_k \\ [\![q_k' = q_k]\!], \quad \text{otherwise} \end{cases} \\ \cap\; \bigcap\nolimits_{k\in\{i,...,n\}} [\![q_k' = q_k]\!] \end{array} \right) \\
&= \quad \bigcup\nolimits_{a\in\Sigma\setminus(\Sigma_i\cup...\cup\Sigma_n)} \bigcap\nolimits_{k\in\{1,...,i-1\}} \begin{cases} \xrightarrow{a}_k (q_k, q_k'), \text{ if } a \in \Sigma_k \\ [\![q_k' = q_k]\!], \quad \text{otherwise} \end{cases} \\
&\quad \cap \;\; \bigcap\nolimits_{k\in\{i,...,n\}} [\![q_k' = q_k]\!]
\end{aligned}$$

Regarding equation 1 for the cofactors we get:

$$\left(\bigcup\nolimits_{a\in\Sigma\setminus(\Sigma_i\cup...\cup\Sigma_n)} \xrightarrow{a}\right)\Big|_i \;\subseteq\; \left\{\emptyset,\; \bigcap\nolimits_{k\in\{i,...,n\}} [\![q_k = q_k']\!]\right\}.$$

A sketch of the BDD for these cofactors is shown by Fig. 7. In the figure, 'A' denotes a BDD for the part where transitions change the assignments for the variables. 'E' denotes the BDD for the empty set and 'B' denotes the BDD for the assignments with $q_k = q_k'$.

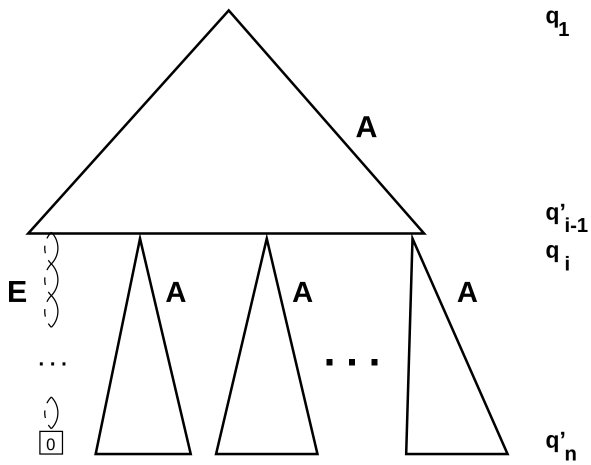


Fig. 9. BDD for Case 3.

Case 2. Discrete transitions concerning only the components $\mathcal{A}_i$ to $\mathcal{A}_n$:

$$\bigcup_{a\in\Sigma\setminus(\Sigma_1\cup\ldots\cup\Sigma_{i-1})} \xrightarrow{a}$$

$$= \bigcap_{k\in\{1,\ldots,i-1\}} [\![q'_k = q_k]\!]$$

$$\cap \bigcup_{a\in\Sigma\setminus(\Sigma_1\cup\ldots\cup\Sigma_{i-1})} \bigcap_{k\in\{i,\ldots,n\}} \begin{cases} \xrightarrow{a}_k (q_k, q'_k), \text{ if } a \in \Sigma_k \\ [\![q'_k = q_k]\!], \quad \text{otherwise} \end{cases}$$

Denoting the second term of the intersection by T, using equation 1 follows: $\left(\bigcup_{a\in\Sigma\setminus(\Sigma_1\cup\ldots\cup\Sigma_{i-1})} \xrightarrow{a}\right)\Big|_i \subseteq \{\emptyset,\ T\}$. The BDD representation is shown in Fig. 8.

Case 3. Discrete transitions concerning components before $\mathcal{A}_i$ as well as components from $\mathcal{A}_i$:

$$\bigcup_{a\in(\Sigma_1\cup\ldots\cup\Sigma_{i-1})\cap(\Sigma_i\cup\ldots\cup\Sigma_n)} \xrightarrow{a}$$

$$= \bigcap_{k\in\{1,\ldots,i-1\}\setminus Comm_{\mathcal{A}}(i)} [\![q'_k = q_k]\!]$$

$$\cap \bigcup_{a\in(\Sigma_1\cup\ldots\cup\Sigma_{i-1})\cap(\Sigma_i\cup\ldots\cup\Sigma_n)} \bigcap_{k\in Comm_{\mathcal{A}}(i)\cup\{i,\ldots,n\}} \begin{cases} \xrightarrow{a}_k (q_k, q'_k), \text{ if } a \in \Sigma_k \\ [\![q'_k = q_k]\!], \quad \text{otherwise} \end{cases}$$

Denoting the first term of the intersection as T_1 and the second term as T_2 we get $T_1|_i \subseteq \{\emptyset, Val(\{q_i, q'_i, ..., q_n, q'_n\})\}$, and, because $T_2 \subseteq Val(\ \{q_k, q'_k \mid k \in Comm_{\mathcal{A}}(i)\} \cup \{q_i, q'_i ..., q_n, q'_n\}\ )$, the following holds: $\mid T_2|_i \mid\ \leq\ \prod_{k\in Comm_{\mathcal{A}}(i)} |Q_k|^2$.

Applying equation 1 $\left|\ \left(\bigcup_{a\in(\Sigma_1\cup\ldots\cup\Sigma_{i-1})\cap(\Sigma_i\cup\ldots\cup\Sigma_n)} \xrightarrow{a}\right)\Big|_i\ \cup\ \{\emptyset\}\ \right|\ \leq\ \prod_{k\in Comm_{\mathcal{A}}(i)} |Q_k|^2 + 1$ holds.

Fig. 9 shows this most interesting part as BDD representation.

Using equation 1 for the union of the three parts of the transition relation the claim follows. □

From the upper bound of the number of cofactors we derive an upper bound for the BDD's size now. We use $|q_i|$ to denote the number of Boolean variables encoding q_i.

Theorem 7. *Let $\mathcal{B}$ be the BDD over $(q_1, q'_1, ..., q_n, q'_n)$ with $[\![\mathcal{B}]\!] = \to' (q_1, q'_1, ..., q_n, q'_n)$. Then the following holds:*

$$|\mathcal{B}|\ \leq\ \sum_{i=1}^{n} \left(2^{2|q_i|} - 1\right) \cdot \left(4 \cdot \prod_{k\in Comm_{\mathcal{A}}(i)} |Q_k|^2 + 4\right)$$

Proof. Theorem 6 states for every $i \in \{1, ..., n\}$:

$$\left| \to' (q_1, q'_1, ..., q_n, q'_n)|_i\ \cup\ \{\emptyset\} \right|\ \leq\ 4 \cdot \prod_{k\in Comm_{\mathcal{A}}(i)} |Q_k|^2 + 4.$$

Using Theorem 5, for every $i \in \{1, ..., n\}$

$$|\mathcal{B}|_i\ \leq\ 4 \cdot \prod_{k\in Comm_{\mathcal{A}}(i)} |Q_k|^2 + 4$$

follows. Finally, using Theorem 4, we get the upper bound for the number of all BDD nodes which code q_i and q'_i:

$$\sum_{l=0}^{2|q_i|-1} 2^l \cdot \left(4 \cdot \prod_{k\in Comm_{\mathcal{A}}(i)} |Q_k|^2 + 4\right) = \left(2^{2|q_i|} - 1\right) \cdot \left(4 \cdot \prod_{k\in Comm_{\mathcal{A}}(i)} |Q_k|^2 + 4\right).$$

Applying the sum over all $i \in \{1, ..., n\}$ we get the claim. □

The statement of this upper bound for the BDD's size obviously reflects the rules of the previous section: If we order communicating components on neighbouring positions in the variable ordering and if we place components communicating with many other components at the beginning of the ordering, then the sets $Comm_{\mathcal{A}}(i)$ have only few elements and the upper bound is relative small.

In the upper bound we used the assumption that q and q' of one component have successive positions within the variable ordering. This makes sense because usually every bit of the successor configuration depends on all bits of the current configuration and thus, there exist a lot of communication within a component. Because a bit of the successor configuration usually is most inter-related to the corresponding bit of the current configuration, we use an interleaved ordering, i.e. each bit of the current configuration is directly followed by the corresponding bit of the successor configuration.

To use the upper bound for the transition relation as size estimation for the BDD representing all reachable configurations we need a modification of Theorem 7. Furthermore, the aim is not to compute an upper bound but to compute an estimation of the BDD's size for comparison of different variable orderings by an algorithm. Therefore, the size estimation should correspond as good as possible to the actual BDD size induced by the variable ordering.

Differently from the BDDs for transition relations, BDDs for sets of reachable configurations do not contain ticked variables for successor configurations. This leads to the following estimation:

$$\sum_{i=1}^{n} \left(2^{|q_i|} - 1\right) \cdot \left(4 \cdot \prod_{k \in Comm_{\mathcal{A}}(i)} |Q_k| + 4\right).$$

Let $\mathcal{B}$ be the BDD over $(q_1, ..., q_n)$ with $[\![\mathcal{B}]\!] = Reach_{\mathcal{A}, [\![\mathcal{A}]\!]_I}(q_1, ..., q_n)$. For $i \in \{1, ..., n\}$, let the variable q_i be encoded by the Boolean variables $x_{i,1}, ..., x_{i,|q_i|}$, such that $\mathcal{B}$ is a BDD over $(x_{1,1}, ..., x_{1,|q_1|}, ..., x_{n,1}, ..., x_{n,|q_n|})$. The estimation contains the worst case assumption that the number of $x_{i,k+1}$ nodes in $\mathcal{B}$ is twice the number of $x_{i,k}$ nodes $(1 \leq k < |q_i|)$. This assumption is not realistic for variables q_i with large number of bits and the estimation is not very similar to the actual size. But for our purpose the estimation is sufficient because it reflects the relation between different variable orderings approximately.

5.3 Finding good variable orderings for CTA models

An algorithm for finding the variable ordering of the best estimation, which considers the bits encoding one component as a unit, must compute the size estimation for all permutations of the sequence $q_1, \ldots, q_n$. Such an algorithm is of exponential time complexity and therefore, it is not relevant for practical use. With a computation of the estimation in $O(n^2)$, the time complexity would be $O(n^2 n!)$

Dynamic programming reduces this complexity to $O(n^3 2^n)$ by storing the results already computed in former iterations for parts of an ordering. But we need an algorithm of polynomial time complexity. Because of this, exact algorithms are not acceptable, especially for large numbers of components. Thus, we have to accept heuristic solutions, e.g. we can use the arbitrary insertion heuristic [LLKS85]. Using this technique we get a time complexity in $O(n^3)$ which is sufficient for our purpose.

Using the structure of CTA models. Cottbus Timed Automata can be considered as composition of timed automata and thus, the algorithms described above can be used. Nevertheless, it is very promising to use special techniques for CTA which use the information about the hierarchical structure as hints for good variable orderings.

The main idea is that the bit-encodings of all objects (variables, automata) contained in the same module instance have successive positions in the variable ordering because objects which are considered as strongly coupled are put together by the model's developer. In the module hierarchy, it is considered that synchronization labels and variables are accessible only where they are really needed. The modules' interfaces should be as small as possible. These considerations lead to a model in which the communication (in the sense of Section 5.1) within a module normally is stronger than the communication of the module with its environment.

We can not guarantee that an algorithm using the hierarchical structure always computes better variable orderings than the algorithm for plain composition of timed automata, but using the structure has the following main advantages:

- The modeller's knowledge about the system is used.
- The problem of variable ordering is partitioned into smaller sub-problems, which results in smaller computation times or that exact algorithms are applicable for these smaller problems, and thus, one might be able to get better variable orderings.

6 Experimental results

To demonstrate the high performance of our approach we give two examples. We use an algorithm for mutual exclusion to examine the computation of the estimation of the BDD's size and we validate the quality of the variable orderings found using our heuristics by measuring the time and the number of nodes needed for verification of the mutex property. The second example is an AND circuit. We consider the module structure to find good variable orderings for the analysis of this model.

Fischer's protocol. Fischer's protocol is a timing-based mutual exclusion protocol. We verified the mutual exclusion property for Fischer's protocol for n processes. The automaton modelling one process has a location for the critical section. The verification task is to compute all reachable configurations and to check whether there exists a reachable situation in which at least two processes are in the critical section. The communication graph of Fischer's protocol for n processes is shown in Fig. 10. Changing the positions of the process automata in the variable ordering has no effect on the estimation of the BDD's size; only the position of the variable k is important and that the encodings of clock and location of an automaton have neighbouring positions. Table 1 reports the results of our experiments with different variable orderings. We give the computation times in seconds on a SUN Ultra-Sparc 1 with 200 MHz processor. We examined five combinations of tools and strategies. The first row of an experiment in the table contains the computation time in seconds and for experiments with our tool the second row displays the growth of the maximal size of the BDD representing reachable configurations.

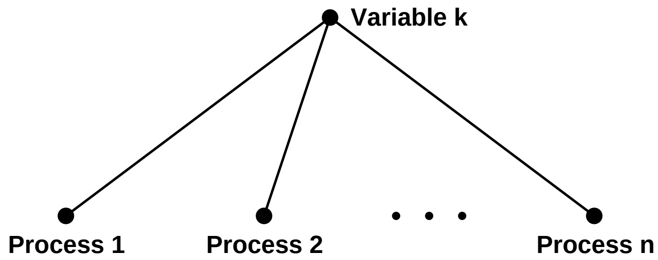


Fig. 10. Communication graph of Fischer's protocol.

In the third experiment we used a variable ordering violating the rule that the variables of a component have successive positions. We used the variable ordering (variable k, automaton 1, ..., automaton n, clock 1, ..., clock n). It leads to a very strong growth of the BDD's size.

If we place the variable k on the last position we get $Comm_1 = Comm_{n+2} = \emptyset$ and $Comm_i = \{1, ..., i-1\}$ for $i \in \{2, ..., n+1\}$. The estimation for $|\mathcal{B}|_1$ and $|\mathcal{B}|_{n+2}$ is 1 again, but for $|\mathcal{B}|_i$ ($i \in \{2, ..., n+1\}$) we get 12^{i-1}. The estimation for $|\mathcal{B}|$ therefore is in $O(12^n)$. The fourth experiment shows that the BDD's size actually grows exponentially.

Placing variable k on first position we have $Comm_1 = Comm_{n+2} = \emptyset$, $Comm_2 = ... = Comm_{n+1} = \{1\}$. For the BDD $\mathcal{B}$ of all reachable configurations we get the estimation 1 for $|\mathcal{B}|_1$ and $|\mathcal{B}|_{n+2}$, and the estimation $n+1$ for $|\mathcal{B}|_2, ..., |\mathcal{B}|_{n+1}$. Thus, the estimation for the size of $\mathcal{B}$ is in $O(n^2)$. The last experiment shows that the estimation matches the actual size very good.

Table 1 also contains a comparison with the most popular tools for the verification of timed automata, Kronos and UppAal. These tools use difference bound matrices to represent sets of clock assignments. The first and second experiment in the table show the computation times of our experiments with these tools. The results show that the computation times of Kronos and UppAal seem to be at least exponential in n, while the computation time of our tool *Rabbit* seems to be polynomial using a good variable ordering (fifth row).

MOS circuit. This section applies our algorithms to the model 'And4' introduced in Section 3. This model has a more complicated communication graph than Fischer's protocol. Table 2 contains the results of our measurements. The computation times are given also in seconds of CPU time on a SUN Ultra-Sparc 1 with 200 MHz processor.

The first row contains a result of the BDD-based version of Kronos. This result, also obtained using a SUN Ultra-Sparc 1, is published in [BMPY97]. The second row of the table shows the computation time needed by our tool *Rabbit* to compute the whole set of reachable configurations. The number of nodes needed to represent this set is given in the third row. (Using a 'stupid' random variable ordering the number of nodes is about 4,000,000 for four input signals.)

Number of processes	4	5	6	7	8	10	12	14	16	32	64
Kronos (DBM)	3.0	191	MO								
UppAal (DBM)	0.5	13.0	657	MO							
Rabbit: Proc. not together	0.3	1.0	5.0	21.6	110	MO					
Number of nodes	828	3053	10983	38515	132245						
Rabbit: Var. k at end	0.3	0.6	1.6	3.9	9.4	46.3	249	MO			
Number of nodes	456	1003	2119	4625	9158	36405	145438				
Rabbit: Var. k in front	0.3	0.4	0.8	1.3	2.3	4.0	8.9	13.6	22.7	208	1920
Number of nodes	326	544	812	1129	1497	2375	3450	4720	6190	24983	100200

Table 1. Computation times for the verification of the mutex property of Fischer's protocol. MO means that more than 64 MB memory were needed.

Number of input signals	2	4	8	16
Kronos (BDD): Computation time	N/A	324.7	N/A	N/A
Rabbit: Computation time	0.5	6.0	79.6	1208.7
Rabbit: Number of BDD's nodes	2007	15722	119870	789835

Table 2. Times and BDD's size for computation of all reachable configurations of the 'And4' model. 'N/A' indicates that measured values are not available for that model.

7 Summary

To provide efficient verification, symbolic representation of the locations of a timed automaton using a BDD-based representation is the first step. The second step is a finite semantics to be able to use BDDs also for the representation of the continuous part of the model. In extension of [ABK+97,AMP98] we gave a formal definition of a discrete semantics for closed timed automata. We proved the correctness of using this discrete semantics for the computation of all reachable locations.

As the next step towards efficient reachability analysis we use the communication structure of the system to find good variable orderings. Based on the concepts described in this paper we implemented *Rabbit* which is a tool for the BDD-based reachability analysis of closed timed automata as an extension of our existing model checker using matrices [BR00]. In order to use the modeller's knowledge for finding a good variable ordering we use the modular modelling notation Cottbus Timed Automata [BR98]. This notation enables us to build hierarchical structures of timed automata-based models.

The main result of our paper is that we can compute good variable orderings and that the verification of timed automata using our technique is very efficient. Our experimental results show that for some classes of models the reachability analysis seems to be of polynomial time and space complexity.

Acknowledgements

We thank Andreas Noack for his conceptual ideas and the implementation of the BDD engine, Michael Vogel for developing CTA versions of the case studies, and Heinrich Rust for his valuable hints for improvement of the paper.

References

[ABK+97] Eugene Asarin, Marius Bozga, Alain Kerbat, Oded Maler, Amir Pnueli, and Anne Rasse. Data-structures for the verification of timed automata. In O. Maler, editor, *Proceedings of the 1st International Workshop on Hybrid and Real-Time Systems (HART'97)*, LNCS 1201, pages 346–360. Springer-Verlag, 1997.

[Alu99] Rajeev Alur. Timed automata. In N. Halbwachs and D. Peled, editors, *Proceedings of the 11th International Conference on Computer-Aided Verification (CAV'99)*, LNCS 1633, pages 8–22. Springer-Verlag, 1999.

[AMP98] Eugene Asarin, Oded Maler, and Amir Pnueli. On discretization of delays in timed automata and digital circuits. In R. de Simone and D. Sangiorgi, editors, *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98)*, LNCS 1466, pages 470–484. Springer-Verlag, 1998.

[ATB94] Adnan Aziz, Serdar Tasiran, and Robert K. Brayton. BDD variable ordering for interacting finite state machines. In *Proceedings of the 31st ACM/IEEE Design Automation Conference (DAC'94)*, pages 283–288, 1994.

[BDM+98] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: a model-checking tool for real-time systems. In A.J. Hu and M.Y. Vardi, editors, *Proceedings of the 10th International Conference on Computer-Aided Verification (CAV'98)*, LNCS 1427, pages 546–550. Springer-Verlag, 1998.

[BMPY97] Marius Bozga, Oded Maler, Amir Pnueli, and Sergio Yovine. Some progress on the symbolic verification of timed automata. In O. Grumberg, editor, *Proceedings of the 9th International Conference on Computer Aided Verification (CAV'97)*, LNCS 1254, pages 179–190. Springer-Verlag, 1997.

[BR98] Dirk Beyer and Heinrich Rust. Modeling a production cell as a distributed real-time system with cottbus timed automata. In Hartmut König and Peter Langendörfer, editors, *Tagungsband Formale Beschreibungstechniken für verteilte Systeme (FBT'98)*, pages 148–159, Aachen, June 1998. Shaker Verlag.

[BR99] Dirk Beyer and Heinrich Rust. A formalism for modular modelling of hybrid systems. Technical Report 10/1999, BTU Cottbus, 1999.

[BR00] Dirk Beyer and Heinrich Rust. Modular modelling and verification with cottbus timed automata. In Charles Rattray and Miroslav Sveda, editors, *Proceedings of the IEEE/IFIP Joint Workshop on Formal Specifications of Computer-Based Systems (FSCBS 2000)*, pages 17–24, Edinburgh, April 2000.

[GPV94] Aleks Göllü, Anuj Puri, and Pravin Varaiya. Discretization of timed automata. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pages 957–958, 1994.

[HMP92] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. What good are digital clocks? In *Proceedings of the 19th International Colloquium on Automata, Languages, and Programming (ICALP'92)*, LNCS 623, pages 545–558. Springer-Verlag, 1992.

[Lam87] Leslie Lamport. A fast mutual exclusion algorithm. *ACM Transactions on Computer Systems*, 5(1):1–11, 1987.

[LLKS85] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem*. John Wiley & Sons, 1985.

[LPY97] Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a Nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2):134–152, October 1997.

[McM92] Kenneth L. McMillan. *Symbolic Model Checking: an approach to the state explosion problem*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1992. Technical report CMU-CS-92-131.

[YBO+98] Bwolen Yang, Randal E. Bryant, David R. O'Hallaron, Armin Biere, Olivier Coudert, Geert Janssen, Rajeev K. Ranjan, and Fabio Somenzi. A performance study of BDD-based model checking. In Ganesh Gopalakrishnan and Phillip J. Windley, editors, *Proceedings of the 2nd International Conference on Formal Methods in Computer-Aided Design (FMCAD'98)*, LNCS 1522, pages 255–289. Springer-Verlag, 1998.