# Can Decision Diagrams Overcome State Space Explosion in Real-Time Verification?

Dirk Beyer and Andreas Noack

Software Systems Engineering Research Group,
Brandenburg Technical University at Cottbus, Germany,
{db|an}@informatik.tu-cottbus.de

**Abstract.** In this paper we analyze the efficiency of binary decision diagrams (BDDs) and clock difference diagrams (CDDs) in the verification of timed automata. Therefore we present analytical and empirical complexity results for three communication protocols. The contributions of the analyses are: Firstly, they show that BDDs and CDDs of polynomial size exist for the reachability sets of the three protocols. This is the first evidence that CDDs can grow only polynomially for models with non-trivial state space explosion. Secondly, they show that CDD-based tools, which currently use at least exponential space for two of the protocols, will only find polynomial-size CDDs if they use better variable orders, as the BDD-based tool Rabbit does. Finally, they give insight into the dependency of the BDD and CDD size on properties of the model, in particular the number of automata and the magnitude of the clock values.

## 1 Introduction

Timed automata [1] are a popular modeling formalism for distributed real-time systems. Several tools for the verification of timed automata exist, e.g. Kronos [19] and Uppaal [4]. One of the main problems in the application of these tools is the exploding consumption of time for the computation and memory for the representation of the reachable configurations. Thus a key issue in achieving practical relevance of verification tools is to investigate efficient data structures for sets of configurations.

Sets of configurations of timed automata consist of locations and associated sets of clock assignments. The most common data structure for the representation of clock assignments are difference bound matrices (DBMs) [14], as used in Kronos and Uppaal. But DBMs are not efficient for non-convex sets, and the use of different data structures for locations and clock assignments often leads to an inefficient representation of configuration sets with many locations.

More recently, binary decision diagrams (BDDs) [12] and clock difference diagrams (CDDs) [7] (and their variants difference decision diagram [17] and clock restriction diagram [18]) were used to represent sets of configurations and were shown to be more efficient than DBMs for many models [7, 9, 11, 17, 18]. Although empirical performance results were published, important questions

H. König, M. Heiner, and A. Wolisz (Eds.): FORTE 2003, LNCS 2767, pp. 193–208, 2003.

remained unanswered: (1) For which models are CDDs smaller than BDDs, and vice versa? (2) Can CDDs cope with state space explosion, i.e. are there polynomial-size CDDs for non-trivial sets of configurations with an exponential number of locations? (3) How large is the gap between the size of the representations which are actually computed by the tools and the size of the best possible representations? (4) What can be done to narrow this gap?

To answer these questions, our strategy is to study instructive examples and use methods that allow generalizations to other practically relevant models. Therefore, we examine the performance of BDDs and CDDs in the analysis of three models of communication protocols, two of them exhibiting state space explosion. For each model, we present not only empirical performance results, but also a detailed analysis of the BDD and the CDD representation of the reachability set. Before this main part, the next two sections give a definition of timed automata and their semantics, and introduce the data structures BDD and CDD.

## 2 Timed Automata

This section starts with a formal definition of timed automata similar to that introduced by Alur [1]. It gives definitions of the continuous and discrete semantics of timed automata. A CDD represents a set of configurations of the continuous semantics, while a BDD represents a set of configurations of the discrete semantics. A theorem at the end of this section states that both semantics are equivalent with respect to the reachable locations for timed automata without strict clock constraints.

### 2.1 Definition

At first, we define *clock constraints*, which are allowed as invariants and guards of a timed automaton. Let $X = \{x_1, \dots, x_n\}$ be a set of clocks. Atomic clock constraints over $X$ are comparisons of a clock with a time constant from $\mathbb{N}$, the set of natural numbers (including 0). Clock constraints are conjunctions of atomic clock constraints. Formally, the set $\Phi(X)$ of clock constraints over $X$ is generated by the grammar $\varphi := x \sim c \mid \varphi \wedge \varphi \mid \mathit{true}$, with $x \in X$, $c \in \mathbb{N}$ and $\sim \in \{\leq, \geq, <, >\}$.

A *clock assignment* of $X$ is a total function from $X$ into $\mathbb{R}_+$, the set of non-negative real numbers. $\mathit{Val}(X)$ denotes the set of all clock assignments of $X$. The semantics $[\![\varphi]\!]$ of a clock constraint $\varphi$ is the set of all clock assignments of $X$ that satisfy $\varphi$. The clock assignment which assigns the value 0 to all clocks is denoted by $v^0$. For $v \in \mathit{Val}(X)$ 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 $v \in \mathit{Val}(X)$ 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*,
- $\Sigma$, with $\Sigma \cap \mathbb{R}_+ = \emptyset$, is a finite set of *synchronization 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 $(l, a, \varphi, Y, m)$ represents a transition labeled with synchronization label $a$ from location $l$ to location $m$.

Complex systems can be described as *parallel composition* of several timed automata which communicate through synchronization labels. A composition of two timed automata with disjoint sets of clocks can be transformed into a single timed automaton by constructing the product automaton. The locations of the product automaton are pairs of component locations, and the invariant of a compound location is the conjunction of the invariants of the component locations. Each two switches of different components with the same synchronization label are synchronized.

### 2.2 Continuous Semantics

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

- *Time transitions:* For $(l, v), (m, w) \in L\times \mathit{Val}(X)$ and $\delta\in\mathbb{R}_+$, $(l, v) \xrightarrow{\delta} (m, w)$ holds iff $l = m$, $w = v + \delta$, $v \in [\![I(l)]\!]$ and $w \in [\![I(l)]\!]$.
- *Discrete transitions:* For $(l, v), (m, w) \in L\times \mathit{Val}(X)$ and $a\in\Sigma$, $(l, v) \xrightarrow{a} (m, w)$ holds iff there exists an $(l, a, \varphi, Y, m) \in E$ with $v \in [\![\varphi]\!]$ and $w = v[Y := 0]$.

Note that discrete transitions to configurations that violate an invariant are possible, but no time is allowed to pass in such configurations.

For a timed automaton $\mathcal{A} = (L, L^0, X, \Sigma, I, E)$ and its continuous semantics $[\![\mathcal{A}]\!]_C = (L\times \mathit{Val}(X), L^0\times\{v^0\}, \Sigma\cup\mathbb{R}_+, \rightarrow)$ we define the following notions: Let $(q_0, q_1, ..., q_k)$ be a sequence of configurations from $L\times \mathit{Val}(X)$, $a_0, a_1, ..., a_{k-1} \in \Sigma\cup\mathbb{R}_+$, $q_0\in L^0\times\{v^0\}$, and $q_i \xrightarrow{a_i} q_{i+1}$ for all $i \in \{0, 1, ..., k-1\}$. Then the configuration $q_k$ is *reachable*. A location $l \in L$ is *reachable* iff there exists a clock assignment $v \in \mathit{Val}(X)$ such that $(l, v)$ is reachable. $\mathit{ReachLoc}([\![\mathcal{A}]\!]_C)$ denotes the set of reachable locations of $\mathcal{A}$ regarding semantics $[\![\mathcal{A}]\!]_C$. We apply analogous notions for other semantics of $\mathcal{A}$.

### 2.3 Discrete Semantics

The continuous semantics of a timed automaton has infinitely many configurations. To represent sets of configurations with data structures for finite sets, equivalent semantics with finitely many configurations are needed. For proving that certain locations are not reachable, it is sufficient that the continuous and the finite semantics have the same set of reachable locations. There exist two ways to transform the continuous semantics into finite semantics: partitioning the configurations into equivalence classes [2], and discretization of time. Partitioning into equivalence classes is the basis for the use of DBMs and CDDs, while discretization is the basis for the use of BDDs, and thus both are needed for the complexity analyses in Sect. 4. We only sketch the less known discretization of time in the following.

A discretization of time which is equivalent to the continuous semantics with respect to the reachable locations exists for all timed automata [15]. However, we restrict ourselves to the subclass of closed timed automata to allow for a discretization which is particularly simple and enables efficient reachability analysis. This restriction is of technical nature, and we did not find examples within our application area of real-time algorithms and embedded systems for which it is difficult to construct models using only non-strict constraints.

*Closed timed automata* have only clock constraints $\varphi$ 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 only integer clock values for the computation of the 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 $\mathbb{N}$. For a timed automaton $\mathcal{A}$ with a clock $x$, $C_{\mathcal{A}}(x)$ denotes the greatest constant $x$ is compared with in a clock constraint of $\mathcal{A}$. 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 $x$.

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 transition system $(L \times Val_I(X), L^0 \times \{v^0\}, \Sigma \cup \mathbb{N}, \rightarrow_I)$ with:

- *Time transitions:* For $(l, v), (m, w) \in L \times Val_I(X)$ and $\delta \in \mathbb{N}$, $(l, v) \overset{\delta}{\rightarrow}_I (m, w)$ holds iff $l = m$, $w = v \oplus \delta$, $v \in [\![I(l)]\!]$ and $w \in [\![I(l)]\!]$.
- *Discrete transitions:* For $(l, v), (m, w) \in L \times Val_I(X)$ and $a \in \Sigma$, $(l, v) \overset{a}{\rightarrow}_I (m, w)$ holds iff there exists an $(l, a, \varphi, Y, m) \in E$ with $v \in [\![\varphi]\!]$, $w = v[Y := 0]$.

This discrete semantics is equivalent to the continuous semantics defined in Sect. 2.2 with respect to the reachable locations.

**Theorem 1.** *For every closed timed automaton $\mathcal{A}$, $ReachLoc([\![\mathcal{A}]\!]_C) = ReachLoc([\![\mathcal{A}]\!]_I)$ holds.*

The proof for Theorem 1 is given in [9]. Proofs of the location equivalence of the integer semantics and the continuous semantics for other formalisms than timed automata can be found in [16] and [5].

# 3 Data Structures for Reachability Sets

The storage and processing of large sets of configurations are the most expensive tasks in the verification of timed automata. Consider, for example, proving that a timed automaton cannot reach a forbidden location. This can be done by computing the set of all reachable configurations and checking that this set contains no configuration with the forbidden location. Therefore, the reachability set has to be represented by a data structure, and this representation is usually large even for medium-size models. The most common data structure for representing sets of configurations are Difference Bound Matrices (DBMs) [14]. More recently, Binary Decision Diagrams and Clock Difference Diagrams were applied to the verification of timed automata and were shown to be more efficient for many models [7, 9, 11, 17, 18]. In the following, these two data structures are introduced.

## 3.1 Binary Decision Diagrams

A binary decision diagram (BDD) [12] represents a set of assignments for a set of Boolean variables. In the discrete semantics for timed automata defined in Sect. 2.3, a configuration consists of the location and the integer values of the clocks. These can be encoded by bit strings, and thus sets of configurations can be represented by BDDs.

A BDD is a rooted directed acyclic graph. It consists of decision nodes, and two terminal nodes called 0-terminal and 1-terminal. Each decision node is labeled by a Boolean variable and has two children called low child and high child. A BDD is maximally reduced with respect to two rules: Merge any isomorphic subgraphs, and eliminate any node whose two children are isomorphic.

The assignments represented by a BDD correspond to the paths from the root node to the 1-terminal. The variable of a node has the value 0 if the path descends to the low child and the value 1 if it descends to the high child.

We only deal with ordered BDDs which means that the variables occur in the same order on any path from the root to a terminal node. For a given variable order, the representation of a set of assignments is unique.

Figure 2 shows BDD representations for two different variable orders of the reachability set of the two timed automata in Fig. 1. The locations of the automata are encoded by binary variables $s_i$, and the clocks are encoded by binary

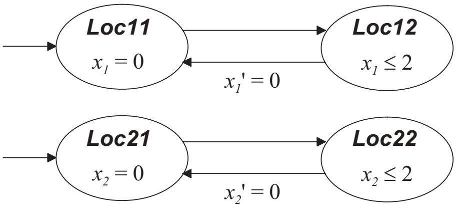


**Fig. 1.** Two simple timed automata

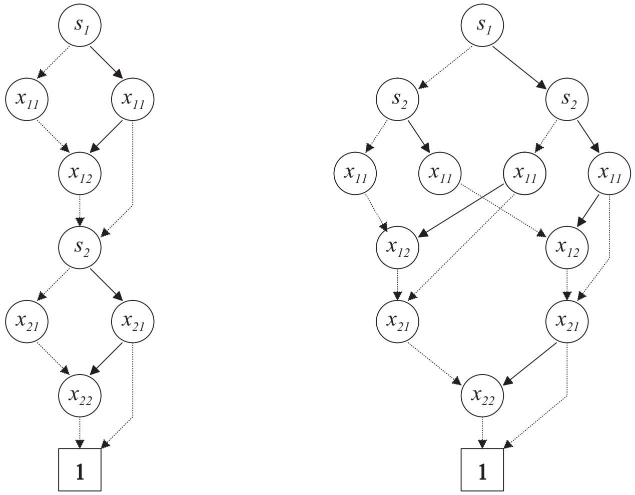


**Fig. 2.** Two BDD representations of the reachability set of the model in Fig. 1

variables $x_{i1}$ and $x_{i2}$ ($i \in \{1, 2\}$). Edges to low children are shown as dotted lines, and edges to the 0-terminal are omitted.

### 3.2 Clock Difference Diagrams

Clock Difference Diagrams (CDDs) were derived from BDDs by Behrmann et al. [7]. Minor variations of CDDs include Clock Restriction Diagrams (CRDs), which are used in the tool RED [18], and Difference Decision Diagrams (DDDs) [17]. The only difference is that all CRD and DDD nodes have two outgoing edges, while CDD nodes can have more than two outgoing edges. But a CDD node with $k$ outgoing edges can be transformed into $k-1$ nodes with two outgoing edges each. So we only need to consider nodes with two outgoing edges and use the unique name Clock Difference Diagram (CDD) in the following.

Nodes in CDDs can not only be labeled with Boolean variables (like BDD nodes), but also with inequalities of the form $x - y < c$, where $x$ and $y$ are clocks or 0, and $c$ is an integer constant. Figure 3 shows CDD representations for two different variable orders of the reachability set of the two timed automata in Fig. 1.

For a fixed variable order, a set of configurations usually has several CDD representations. So even if for a given variable order the smallest CDD representation of a reachability set is significantly smaller than the (unique) BDD representation, it is not guaranteed that CDD-based tools actually find a representation that is smaller than the BDD.

In contrast to BDDs, there is currently no evidence that CDDs can cope with an explosion of the reachable locations. The only model with an exponential number of reachable locations for which a polynomial-size CDD representation has been published is Milner's scheduler in [17]. However, the state space explosion in

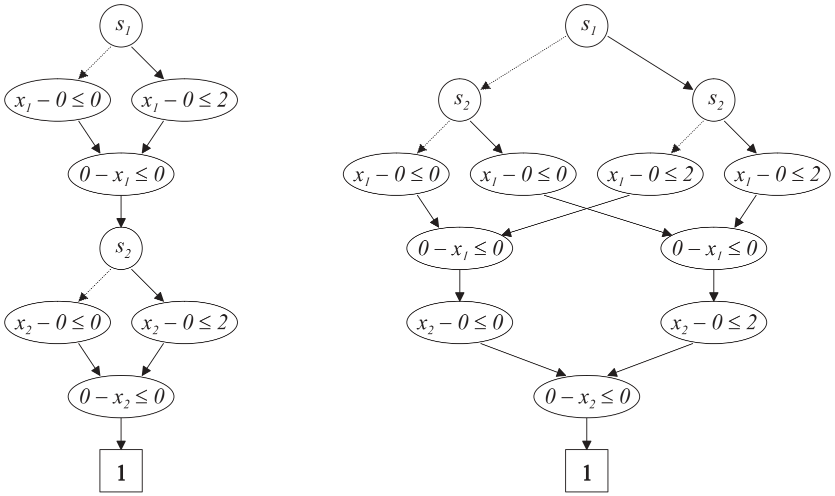


**Fig. 3.** Two CDD representations of the reachability set of the model in Fig. 1

this model is trivial because there is only a *linear* number of reachable locations with pair-wise different reachable clock assignments. In Sect. 4, we show that there exist non-trivial reachability sets with exponential number of locations, but polynomial CDD representation.

### 3.3 Variable Order

The size of BDDs and CDDs is highly dependent on the variable order. Consider a generalization of Fig. 1 to $n$ automata. For the variable order $s_1, x_1, ..., s_n, x_n$ the BDD and the CDD representation of the reachability set have $O(n)$ nodes, while for the variable order $s_1, ..., s_n, x_1, ..., x_n$ the BDD and the CDD have $O(2^n)$ nodes.

We denote the second type of variable order, where the locations of the automata precede the clock values, as *location-first* variable orders. The simple example indicates a general problem: The number of reachable locations is often exponential in the number of automata. If the reachable clock assignments are different for most of the reachable locations, the size of the BDD or CDD representation using location-first variable orders grows at least as fast as the number of reachable locations. The CDD-based tool RED [18] and the CDD implementation described in [7] use location-first variable orders, while the BDD-based tool Rabbit [8] applies techniques for finding good variable orders.

Because finding the optimal variable order of a BDD is algorithmically intractable [10], Rabbit applies heuristics. Like Aziz et al.'s approach for communicating finite automata [6], the heuristic optimizes the variable order with respect to a size estimate for the BDD representation of the reachability set. This estimate is derived from an upper bound for the size of the transition relation which is proven in [9].

Research in the verification of finite automata has shown that besides variable ordering, several other techniques can considerably improve the efficiency of BDD implementations [13]. Some of these techniques, like partitioning transition relations, and modified breadth first search to minimize the BDD size of intermediate results, were adapted for the use in Rabbit [9].

## 4 Case Studies: Three Protocols

In this section, we examine the sizes of BDDs and CDDs for the reachability sets of three models of communication protocols. To show how the data structures deal with state space explosion, we chose two models (Fischer, CSMA/CD) which exhibit exponential growth of the reachable locations. For comparison, we used one more deterministic model (FDDI) with linear growth of the reachable locations.

As empirical results, we present memory requirements and runtimes for the tools RED [18] version 3.1, which is based on location-first CDDs, Rabbit [8] version 2.1, which is based on BDDs with automatic variable ordering, and (for comparison) Uppaal2k [4] version 3.2.4 (without approximation), a popular and highly optimized DBM-based tool. The runtimes are given in seconds of processor time on a Linux PC with 1 GHz AMD Athlon processor. The memory requirements for the representation of the reachability sets are given in BDD nodes for Rabbit. (One BDD node including supporting data structures takes about 26 bytes.) Because we have no access to such precise measures for RED and Uppaal, the overall memory requirements in MByte are given for these tools. Empty table entries mean that more than 7200 seconds of processor time or more than 400 MBytes of memory were consumed.

Besides empirical results, we give for each model a detailed analysis of the size of the smallest BDD, the smallest CDD, and the smallest location-first CDD representation of the reachability set. The analyses complement the empirical results in three ways. First, they explain the empirical results. Knowledge of the causes of inefficiencies is essential for systematic improvement. Second, they enable us to assess the potential of techniques that are not yet implemented, like variable ordering for CDDs. Third, analytical results can be generalized from the analyzed models to other models which allow the same (or a similar) argumentation.

The size of the BDD or CDD of the reachability set is not the only factor that determines the performance of reachability analysis. Other important factors include the size of intermediate BDDs or CDDs and the representation of the transition relation. It is possible to apply our analysis techniques also for intermediate BDDs and CDDs, but for brevity, we do not consider these factors explicitly here. The runtime measurements for all three models show that the overall performance, including the computation of intermediate representations, does not deviate drastically from the analytical results for the reachability sets.

### 4.1 Fischer's Protocol for Mutual Exclusion

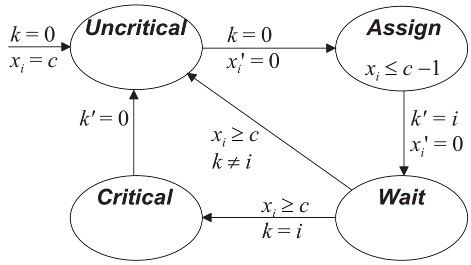
**Model description.** The model of Fischer's timing-based mutual exclusion protocol is composed from $n$ instances of the timed automaton depicted in Fig. 4, each modeling one process. They communicate through a shared variable $k$ $(0 \leq k \leq n)$. The variable $k$ could be modeled as additional automaton, but we prefer a more compact notation. The initial value of $k$ is 0, which means that no process tries to enter the critical section. When $k \neq 0$, the $k$th process is scheduled to enter the critical section or already stays there.

Each automaton has four locations. Initially it is in *Uncritical*. If no other process tries to enter the critical section ($k = 0$), it can move to *Assign*. Process $i$ needs less than $c$ time units for the assignment $k := i$. This is modeled by the clock $x_i$, which measures the time spent in *Assign*, and the invariant of *Assign*, which forces the automaton to leave the location before $x_i$ is $c$. The transition to *Wait* sets $k$ to $i$. After the process has stayed in *Wait* for $c$ time units, it is guaranteed that no process is in location *Assign*. The process is allowed to enter the critical section if the value of $k$ still is its identifier $i$, otherwise it has to go back to *Uncritical*.

**Complexity analysis.** Consider the protocol with $n \geq 2$ processes. The location of each process automaton can be encoded by 2 bits and the value of $k$ by $\lceil \mathrm{ld}(n+1) \rceil$ bits for a total of $\Theta(n)$ bits. (ld denotes the logarithm to base 2.)

For each clock $x_i$, the values greater or equal $c$ do not need to be distinguished (see Sect. 2.3), and are therefore represented by the single value $c$. So $\Theta(n \text{ ld } c)$ bits are needed to encode the $n$ clocks in the BDD representation.

An (up to a constant factor) optimal BDD variable order is: $k$, location of process 1, $x_1$, ..., location of process $n$, $x_n$. (See [6, 9] for information about



| # processes | 4 | 5 | 6 | 7 | 8 | 10 | 12 | 14 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Uppaal, sec | 0.06 | 1.44 | 181 | 32488 | | | | | | | | |
| MByte | 1.2 | 3.7 | 24.6 | 352 | | | | | | | | |
| RED, sec | 1.64 | 6.78 | 21.7 | 60.7 | 168 | 1400 | | | | | | |
| MByte | 1.9 | 1.9 | 2.1 | 2.4 | 3.1 | 8.9 | | | | | | |
| Rabbit, sec | 0.04 | 0.08 | 0.15 | 0.26 | 0.50 | 1.35 | 1.61 | 3.81 | 6.50 | 61.4 | 559 | 5200 |
| BDD nodes | 326 | 544 | 812 | 1129 | 1497 | 2375 | 3450 | 4720 | 6190 | 24983 | 100200 | 401161 |

**Fig. 4.** Fischer's protocol: timed automaton for the $i$th process and performance results

variable ordering.) A corresponding CDD variable order is obtained by replacing each $x_i$ by $x_i - 0$, $0 - x_i$, and all $x_i - x_j$, $x_j - x_i$ with $j \in \{i{+}1, ..., n\}$.

The model can reach the following configurations:

1. $k = 0$

   When $k = 0$, every configuration is reachable where every process $i$ ($1{\leq}i{\leq}n$) is in one of the locations *Uncritical* with $x_i = c$, *Assign* with $x_i \leq c - 1$, or *Wait* with $x_i = c$. However, at least one process must be outside *Wait*, else $k \neq 0$.

   So the only fact about the configurations of the processes $1, ..., i-1$ ($1{<}i{\leq}n$) that influences the possible configurations of the remaining processes $i, ..., n$ is if any of the processes $1, ..., i-1$ is outside *Wait*. Because there are only two possibilities, the set of configurations can be represented by a BDD of $\Theta(2n \text{ ld } c) = \Theta(n \text{ ld } c)$ nodes or a CDD of $\Theta(2n) = \Theta(n)$ nodes.

   A *location-first* CDD representation has $\Omega(2^n)$ nodes, because $2^n$ of the $3^n{-}1$ reachable locations have different reachable clock assignments.
2. $1 \leq k \leq n$

   Then process $k$ must be in the location *Wait* or *Critical*.

   (a) Process $k$ is in *Wait*.

   Then every process $i$ ($i \neq k$) can be in *Uncritical* with $x_i = c$, in *Wait* with $x_i \geq x_k$, or in *Assign* with $x_i \geq x_j$, where $j$ is the process that has the maximum clock value among all processes in location *Wait* having clock values smaller than $c$.

   To ensure that the processes $i, ..., n$ satisfy these constraints, knowledge of the value of $k$ and three clock values of the processes $1, ..., i-1$ is needed: the smallest clock value of the processes in location *Assign*, and the smallest and largest clock value of the processes in location *Wait*. So the BDD representation has $\Theta(c^3 n^2 \text{ ld } c)$ nodes.

   In the smallest CDD, the representation of the processes depends on the value of $k$ and the identifier $j \in \{1, ..., n\}$ of the process in location *Assign* with the smallest clock value (or alternatively the identifier of the process in location *Wait* with the largest clock value smaller than $c$). So the smallest CDD representation has $\Theta(n^3)$ nodes.

   (b) Process $k$ is in *Critical*.

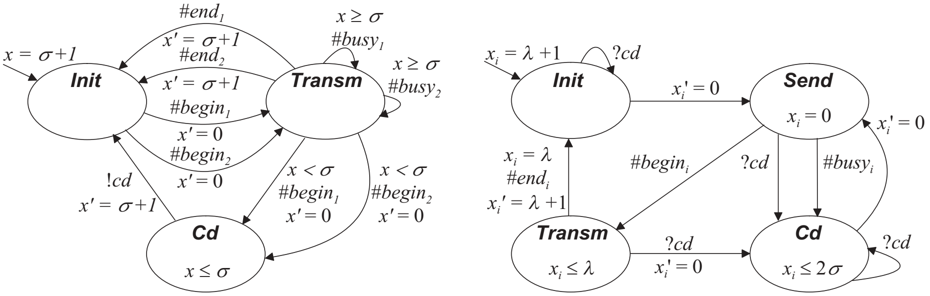
   Every other process $i$ ($i \neq k$) can be in *Uncritical* or *Wait* with $x_i = c$. Thus the asymptotic complexity is for the BDD $\Theta(n \text{ ld } c)$ and for the CDD $\Theta(n)$.

So the overall size is $\Theta(c^3 n^2 \text{ ld } c)$ for the BDD, $\Theta(n^3)$ for the smallest CDD and $\Omega(2^n)$ for the location-first CDD. The time and space used by the BDD-based tool Rabbit (see Fig. 4, $c = 2$) conforms to the analytical result for BDDs, and the measurement data for the CDD-based tool RED conform to the analytical result for location-first CDDs. The large gap between the analytical results for the smallest and the location-first CDD shows that better variable orders are necessary to cope with the explosion of the reachable locations.

### 4.2 CSMA/CD Protocol

**Model description.** The model of the CSMA/CD protocol consists of one automaton for the medium and $n$ automata for senders, as depicted in Fig. 5 (cf. [19]). The signals $begin_i$, $end_i$ and $busy_i$ exist for each sender $i$ for communication with the medium. The shared signal $cd$ indicates a collision. As long as the medium is unused, the automaton is in the location *Init*. The medium synchronizes with signal $begin_i$, if the sender $i$ wants to use the medium. The clock $x$ measures the time that elapsed since the start of the transmission. If another sender starts to transmit within the propagation time of $\sigma$ time units, then the medium switches to location *Cd* and both senders receive the signal $cd$ indicating a collision. After no other sender started to transmit within $\sigma$ time units, the medium answers requests from another sender $j$ by the signal $busy_j$.

If a sender in the location *Init* wants to transmit data, it switches to the location *Send*. Because of the invariant of this location it has to move to the next location immediately. If the medium is not used (i.e. it is in state *Init*), then the sender and the medium synchronously switch to location *Transm*. Now the sender can transmit data for $\lambda$ time units ($\lambda > \sigma$), measured by clock $x_i$. After finishing the transmission the sender releases the medium with signal $end_i$. If another sender enters location *Transm* within the propagation time $\sigma$ then all senders receive the signal $cd$, and the transmitting senders switch to the location *Cd*. After a random delay of at most $2\sigma$ time units the senders try again to access the medium.



| # senders | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Uppaal, sec | 0.01 | 0.03 | 5.1 | | | | | | | | |
| MByte | 1.2 | 1.2 | 6.1 | | | | | | | | |
| RED, sec | 0.05 | 0.28 | 1.15 | 5.88 | 41.4 | 516 | | | | | |
| MByte | 1.9 | 2.1 | 2.4 | 4.5 | 15.9 | 75 | | | | | |
| Rabbit, sec | 0.02 | 0.08 | 0.23 | 0.49 | 0.82 | 1.28 | 1.83 | 2.69 | 12.6 | 62.9 | 367 |
| BDD nodes | 103 | 329 | 561 | 793 | 1025 | 1257 | 1489 | 1721 | 3577 | 7289 | 14713 |

**Fig. 5.** CSMA/CD protocol: timed automata for the medium for two senders (left) and for the $i$th sender (right); performance results

**Complexity analysis.** Consider the protocol with $n > 2$ senders. The locations of the medium and the $n$ senders can be encoded by $\Theta(n)$ bits, and the values of the $n$ clocks are encoded by $\Theta(n \text{ ld } \lambda)$ bits in the BDD representation.

An (up to a constant factor) optimal variable order is: location of the medium, $x$, location of sender 1, $x_1$, ..., location of sender $n$, $x_n$.

The model can reach the following configurations:

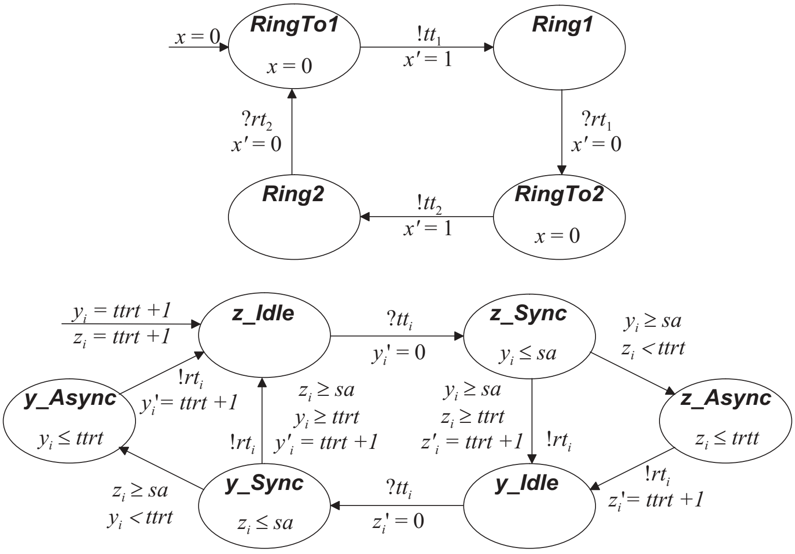
1. The medium is in *Init*.
   Then every sender $i$ $(1 \le i \le n)$ can be in any of the locations *Init* with $x_i = \lambda + 1$, *Send* with $x_i = 0$, and *Cd* with $0 \le x_i \le 2\sigma$.
   These configurations can be represented by a BDD with $\Theta(n \text{ ld } \lambda)$ nodes or a CDD with $\Theta(n)$ nodes.
   However, all of the $3^n$ reachable locations have different sets of reachable clock assignments, so the smallest location-first CDD has $\Omega(3^n)$ nodes.
2. The medium is in *Transm*.
   Then exactly one sender $k$ is in *Transm*, and the clock $x_k$ of this sender equals the clock $x$ of the medium. Again, every sender $i$ $(i \neq k)$ can be in any of the locations *Init* with $x_i = \lambda + 1$, *Send* with $x_i = 0$, and *Cd*. However, because no sender can move to location *Cd* while $x < \sigma$ (without forcing the medium to leave *Transm*), the possible values of $x_i$ depend on $x$ if sender $i$ is in *Cd*.
   These clock dependencies only change the constant factor in the smallest CDD size, so it is $\Theta(n)$. But they contribute a factor of $\Theta(\sigma)$ to the BDD size, yielding $\Theta(n\sigma \text{ ld } \lambda)$.
3. The medium is in *Cd*.
   Then exactly two senders $k_1$ and $k_2$ are in *Transm*. Let $x_{k_1} \ge x_{k_2}$. Because the medium could only move from *Transm* to *Cd* when $x = x_{k_1} < \sigma$, we have $x_{k_1} < x + \sigma$ and $x_{k_2} < x + \sigma$.
   Every sender $i \notin \{k_1, k_2\}$ can be in any of the locations *Init* with $x_i = \lambda + 1$, *Send* with $x_i = 0$, and *Cd* with $x_i \ge x_{k_1}$ (because no sender entered *Cd* since the medium moved from *Init* to *Transm*).
   To ensure that the senders $i, ..., n$ $(1 < i \le n)$ satisfy these constraints, knowledge of the value of $x$ and two clock values of the processes $1, ..., i-1$ is needed: the value of the largest clock value of a sender in *Transm*, and the value of the smallest clock value of a sender in *Cd*. This leads to a BDD with $\Theta(n\sigma^3 \text{ ld } \lambda)$ nodes.
   The dependencies on $x$ do not significantly increase the size of the smallest CDD, but the representation of every sender depends on the identifier $j \in \{1, ..., n\}$ of the sender in location *Transm* with the largest clock value (or alternatively the identifier of the sender in location *Cd* with the smallest clock value). So the smallest CDD representation has $\Theta(n^2)$ nodes.

So the overall size is $\Theta(n\sigma^3 \text{ ld } \lambda)$ for BDDs, $\Theta(n^2)$ for smallest CDDs and $\Omega(3^n)$ for location-first CDDs. The analytical result for BDDs conforms to the empirical data for Rabbit in Fig. 5 ($\sigma = 1$, $\lambda = 4$), and the result for location-first CDDs roughly conforms to the empirical data for RED in this table. Like for Fischer's protocol, the number of reachable locations explodes with growing $n$, and consequently the location-first CDDs are much greater than the smallest CDDs for better variable orders.

### 4.3 Token Ring FDDI Protocol

**Model description.** The model of the FDDI protocol consists of one automaton for the token ring and $n$ automata for the stations, as depicted in Fig. 6 (cf. [20]). Station $i$ gets the token by receiving the signal $tt_i$ (*take token*). After the data transmission, it returns the token to the ring by sending the signal $rt_i$ (*release token*). The clock $x$ and the invariants of the locations *RingTok* ensure that the ring immediately transmits the token to station $i + 1$ (mod $n$).

Each station has the modes *waiting for token*, *synchronous transmission* and *asynchronous transmission*. According to the roles of the two clocks $x_i$ and $y_i$, we distinguish six locations: In the locations *z_Idle*, *z_Sync* and *z_Async* the clock $z_i$ measures the time that elapsed since the start of the previous transmission of station $i$. The clock $y_i$ is reset when receiving the signal $tt_i$ and ensures that



| # stations | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 32 | 64 |
|---|---|---|---|---|---|---|---|---|---|---|
| Uppaal, sec | 0.01 | 0.03 | 0.16 | 1.42 | 18.2 | 279 | 4530 | | | |
| MByte | 1.2 | 1.2 | 2.2 | 3.0 | 7.4 | 30.3 | 151 | | | |
| RED, sec | 0.02 | 0.09 | 0.26 | 0.61 | 1.18 | 2.16 | 3.62 | 6.31 | 157 | 2550 |
| MByte | 1.30 | 2.07 | 2.26 | 2.49 | 2.79 | 3.13 | 3.47 | 3.92 | 9.84 | 35.8 |
| Rabbit, sec | 0.04 | 0.25 | 0.99 | 4.20 | 11.4 | 26.9 | 49.8 | 142 | | |
| BDD nodes | 350 | 4211 | 18501 | 64505 | 151737 | 305523 | 501305 | 993673 | | |

**Fig. 6.** FDDI protocol: timed automata for the ring for two stations (top) and for the $i$th station (bottom); performance results

the time for the synchronous transmission in location *z_Sync* does not exceed $sa$. After the synchronous transmission, the station is permitted to transmit data in the asynchronous mode if the time since the previous transmission (as measured by $z_i$) is smaller than $ttrt$ (*target token rotation time*). In the locations *y_Idle*, *y_Sync* and *y_Async* the clocks $y_i$ and $z_i$ swap their roles.

**Complexity analysis.** Consider the protocol with $n \geq 2$ stations. The locations of the ring and the $n$ stations can be encoded by $\Theta(n)$ bits, and the values of the $2n$ clocks are encoded by $\Theta(n \text{ ld } ttrt)$ bits in the BDD representation.

An (up to a constant factor) optimal variable order is: location of the ring, $x$, location of station 1, $y_1$, $z_1$, ..., location of station $n$, $y_n$, $z_n$.

We start with the characterization of the reachable locations and then proceed with the reachable clock assignments. Regarding the locations, let the ring automaton be in *RingTok* or *Ringk*. Consider two cases concerning the location of station 1:

1. Station 1 is in *z_Idle*, *z_Sync* or *z_Async*.
   If $k = 1$, then all stations $i$ with $i > 1$ are in *z_Idle*. If $k > 1$, then all stations $i$ with $i < k$ are in *z_Idle*, and all stations $i$ with $i > k$ are in *y_Idle*. Station $k$ is in *y_Idle* if the ring is in *RingTok*, and in *y_Sync* or *y_Async* if the ring is in *Ringk*.
2. Station 1 is in *y_Idle*, *y_Sync* or *y_Async*. (Analogous to the first case.)

So only $6n$ locations are reachable, and the possible locations of the stations $i, ..., n$ $(i \geq 2)$ are completely determined by the location of the ring and station 1. Consequently, the representation of the locations has $\Theta(n^2)$ BDD nodes.

Let us now consider the clock values. Again, let the ring automaton be in *RingTok* or *Ringk*.

1. Station $k$ is in *z_Idle*, *z_Sync* or *z_Async*.
   Then $y_{k+1} = ... = y_n = z_1 = ... = z_{k-1} = ttrt+1$ and $y_k \leq y_{k-1} \leq ... \leq y_1 \leq z_n \leq ... \leq z_k$. Although this characterization is not entirely precise (e.g., even $y_k + sa \leq y_{k-1}$ holds), it is sufficient for the estimation of the BDD and CDD size. It implies two dependencies between the clocks of the stations $1, ..., i-1$ $(1 < i \leq n)$ and the clocks of the stations $i, ..., n$:
   - If $i \leq k$ then $y_i \leq y_{i-1}$ else $z_i \leq z_{i-1}$, and
   - $y_1 \leq z_n$.
2. Station $k$ is in *y_Idle*, *y_Sync* or *y_Async*. (Analogous to the first case.)

In the BDD representation, the two clock dependencies contribute a factor of $\Theta(ttrt^2)$ to the number of nodes, yielding an overall size of $\Theta(n^2 ttrt^2 \text{ ld } ttrt)$. The data in Fig. 6 was created with $ttrt = 2n + 1$ and $sa = 1$, so we expect a BDD size of $\Theta(n^4 \text{ ld } n)$, which agrees with the BDD size data for Rabbit.

In the smallest CDD, the representation of the clock dependencies only requires a constant-factor extension of the representation of the locations, so the overall number of nodes is in $\Theta(n^2)$. Fig. 6 shows that the tool RED, which is based on location-first CDDs, actually achieves polynomial performance. Because the number of reachable locations is small, the location-first variable order is appropriate here.

## 5 Conclusion

We examined for three models of communication protocols how the size of the BDD and the CDD representation of the reachability set depends on two properties of the models: the number of automata and the magnitude of the clock values. The results are summarized in Fig. 7. These analyses answer the four questions of the introduction:

(1) While the size of CDDs is almost independent of the magnitude of the clock values, the size of BDDs is very sensitive to large clock values. (The latter was observed before e.g. in [3].) On the other hand, BDDs with good variable orders are least sensitive to the number of automata, but smallest CDDs with good variable orders are only slightly worse. (2) So there exist polynomial-size CDDs for non-trivial sets of configurations with an exponential number of locations. (3) But, in contrast to the BDD-based tool Rabbit, the time and space used by the CDD-based tool RED is usually at least exponential for these models. (4) This aspect of the performance will not be competitive to BDDs until CDD-based tools use techniques for finding better variables orders, as Rabbit does.

So there exist CDDs which combine the advantages of DBMs and BDDs – independence of the magnitudes of clock values and robustness against state space explosion – in the verification of practical distributed systems. Work on variable ordering is a necessary step towards tools that find them.

| Data structure | BDD | CDD | CDD |
|---|---|---|---|
| Variable order | interleaved | interleaved | location-first |
| Fischer (Sect. 4.1) | $\Theta(n^2c^3 \text{ ld } c)$ | $\Theta(n^3)$ | $\Omega(2^n)$ |
| CSMA/CD (Sect. 4.2) | $\Theta(n\sigma^3 \text{ ld } \lambda)$ | $\Theta(n^2)$ | $\Omega(3^n)$ |
| Token Ring FDDI (Sect. 4.3) | $\Theta(n^2 ttrt^2 \text{ ld } ttrt)$ | $\Theta(n^2)$ | $\Theta(n^2)$ |

**Fig. 7.** Summary of the decision diagram sizes. $n$ denotes the number of processes (Fischer) or senders (CSMA/CD, FDDI), and $c$, $\sigma$, $\lambda$, and $ttrt$ are constants from clock constraints of the models.

## References

1. Rajeev Alur. Timed automata. In *Proc. CAV'99*, LNCS 1633, pages 8–22. Springer, 1999.
2. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. Rajeev Alur, Alon Itai, Robert P. Kurshan, and Mihalis Yannakakis. Timing verification by successive approximation. *Information and Computation*, 118(1):142–157, 1995.
4. Tobias Amnell, Gerd Behrmann, Johan Bengtsson, Pedro R. D'Argenio, Alexandre David, Ansgar Fehnker, Thomas Hune, Bertrand Jeannet, Kim G. Larsen, M. Oliver Möller, Paul Pettersson, Carsten Weise, and Wang Yi. Uppaal - now, next, and future. In *Proc. MOVEP'00*, LNCS 2067, pages 99–124. Springer, 2001.

5. Eugene Asarin, Oded Maler, and Amir Pnueli. On discretization of delays in timed automata and digital circuits. In *Proc. CONCUR'98*, LNCS 1466, pages 470–484. Springer, 1998.
6. Adnan Aziz, Serdar Tasiran, and Robert K. Brayton. BDD variable ordering for interacting finite state machines. In *Proc. DAC'94*, pages 283–288. ACM Press, 1994.
7. Gerd Behrmann, Kim G. Larsen, Justin Pearson, Carsten Weise, and Wang Yi. Efficient timed reachability analysis using clock difference diagrams. In *Proc. CAV'99*, LNCS 1633, pages 341–353. Springer, 1999.
8. Dirk Beyer. Efficient reachability analysis and refinement checking of timed automata using BDDs. In *Proc. CHARME'01*, LNCS 2144, pages 86–91. Springer, 2001.
9. Dirk Beyer. Improvements in BDD-based reachability analysis of timed automata. In *Proc. FME'01*, LNCS 2021, pages 318–343. Springer, 2001.
10. Beate Bollig and Ingo Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on Computers*, 45(9):993–1002, 1996.
11. Marius Bozga, Oded Maler, Amir Pnueli, and Sergio Yovine. Some progress in the symbolic verification of timed automata. In *Proc. CAV'97*, LNCS 1254, pages 179–190. Springer, 1997.
12. Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transaction on Computers*, C-35(8):677–691, 1986.
13. Jerry R. Burch, Edmund M. Clarke, David E. Long, Kenneth L. McMillan, and David L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on CAD*, 13(4):401–424, 1994.
14. David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proc. Automatic Verification Methods for Finite State Systems*, LNCS 407, pages 197–212. Springer, 1990.
15. Aleks Göllü, Anuj Puri, and Pravin Varaiya. Discretization of timed automata. In *Proc. Decision and Control*, pages 957–958, 1994.
16. Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. What good are digital clocks? In *Proc. ICALP'92*, LNCS 623, pages 545–558. Springer, 1992.
17. Jesper Møller, Jakob Lichtenberg, Henrik R. Andersen, and Henrik Hulgaard. Difference decision diagrams. In *Proc. CSL'99*, LNCS 1683, pages 111–125. Springer, 1999.
18. Farn Wang. Symbolic verification of complex real-time systems with clock-restriction diagram. In *Proc. FORTE'01*, pages 235–250. Kluwer, 2001.
19. Sergio Yovine. Kronos: A verification tool for real-time systems. *Software Tools for Technology Transfer*, 1(1-2):123–133, 1997.
20. Sergio Yovine. Model checking timed automata. In *Lectures on Embedded Systems*, LNCS 1494, pages 114–152. Springer, 1998.