Preprint from Proceedings of GI/ITG/GMM-Workshop "Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen", Oldenburg, Germany, February 2011

Verifying Incomplete Networks of Timed Automata *

Christian Miller and Christoph Scholl and Bernd Becker
Institute of Computer Science
Albert-Ludwigs-University
79110 Freiburg, Germany
{millerc,scholl,becker}@informatik.uni-freiburg.de

Abstract

Verification of real-time systems – e.g. communication protocols or embedded controllers – is an important task. One method to detect errors is called bounded model checking (BMC). In BMC a system is iteratively unfolded and then transformed into a satisfiability problem. If an appropriate solver finds the k-th instance to be satisfiable, a counterexample for a given safety property has been found. In this paper, we present a novel approach to apply BMC to networks of timed automata (i.e. a system of several interacting subautomata) where parts of the network are unspecified (so-called blackboxes). Here, we would like to answer the question of unrealizability, that is, is there a path of a certain length violating a safety property regardless of the implementation of the blackboxes. Focusing on two prevalent communication models for networks of interacting timed automata, we introduce two methods to solve these types of BMC problems. The first is based on fixed transitions, which in some cases is too coarse of an abstraction. In the second approach we prove unrealizability by introducing universal quantification, yielding more advanced quantified SAT-Modulo-Theory formulas.

1. Introduction

Real-time systems appear in many areas of life, such as time critical communication protocols, or in embedded controllers for automobiles. As these systems grow in complexity, verifying their correctness becomes harder, but increasingly more important. One method to prove the presence of errors in complex systems is bounded model checking (BMC) [1, 2]. BMC accomplishes this by iteratively unfolding the system k times until a predefined maximum unfolding depth is reached. After adding the negated property, the BMC instance is converted into a satisfiability problem and then solved by an appropriate solver. If the solver finds the k-th instance satisfiable, a path of length k violating the property has been found. Whereas BMC for conventional discrete circuits result in SAT-problems, BMC instances for real-time systems are encoded into so-called SAT-Modulo-Theory (SMT) formulas, since they are augmented with continuous time constraints over real-valued variables. BMC for timed automata has been studied and improved by several groups [3–6]. All this previous work has made the assumption that the entire design is specified. In contrast, this work focuses on BMC with *incomplete information*, that is designs where parts of the system are not known (so-called blackboxes). For discrete circuits this has been examined extensively in [7–9], where the encoding of the BMC instances can yield a SAT or QBF formula, depending on the abstraction level.

*This work has been partially funded by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, www.avacs.org)

In this paper, we describe our approach for analyzing incomplete designs in the continuous domain with the goal of answering the question of unrealizability, that is, for a given safety property (here, reachability of locations of the system) is there a counterexample which exists regardless of the implementation of the blackboxes? If so, we call this property unrealizable for the network of timed automata. For interactions between the subautomata in the network we focus on two prevalent models, namely *synchronization* and *bounded integer communication*. In [10] we presented an approach to prove unrealizability for each communication model separately. In contrast, we now extend our methods to handle both communication models simultaneously. First experimental results show the efficiency of our methods.
Next, in Section 2 we present some preliminary information. Section 3 defines the BMC encoding for a network of timed automata and then extends this encoding for both communication models. Then in Section 4 we introduce blackboxes into networks of timed automata and show our approach for encoding and solving these blackbox BMC problems. After presenting preliminary results in Section 5, we conclude the paper in Section 6.

2. Preliminaries

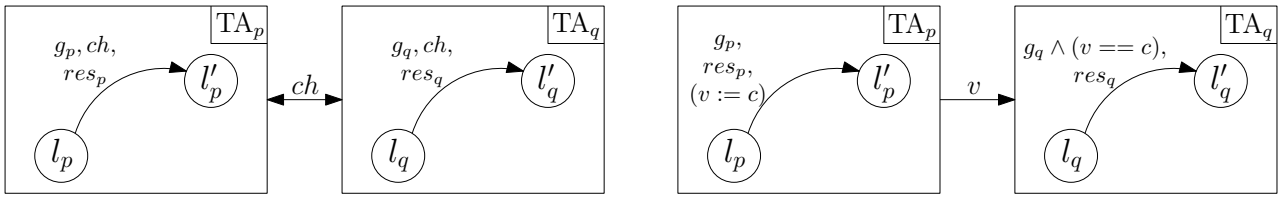
In this paper we focus on timed automata based on the model in [11]. Timed automata are an extension of conventional automata by real-valued clock variables. For these clocks, time is continuous, and transitions and invariants on locations can be labeled with constraints over clocks. The set $CC(X)$ of clock constraints over a set of clock variables X is a conjunction of comparisons of clocks to constants and is defined inductively by $g := true \mid x \sim c \mid g_1 \wedge g_2$ with $c \in \mathbb{N}$, $x \in X$, and $\sim \in \{<, \leq, >, \geq\}$. A timed automaton is defined as a tuple $\text{TA} = \langle L, l_0, X, Inv, E \rangle$ with:

- a set of locations L,
- an initial location $l_0 \in L$,
- a set of real-valued clock variables X,
- a function $Inv : L \to CC(X)$, which assigns a clock constraint to each location and
- a set of edges $E \subseteq L \times CC(X) \times 2^X \times L$.

A transition $e = \langle l, g, res, l' \rangle \in E$ can be taken if the guard $g \in CC(X)$ is satisfied. If $res \subseteq 2^X$ is not empty then all clock variables in res are reset to 0. Additionally the current assignment (after resetting clocks in res) to the clock variables must not violate the invariant of the successor location l'. A state $s = \langle l, \nu \rangle$ of a timed automaton is given by the current location l and the current assignment ν to the clock variables. Note, transitions happen instantaneously, and a timed automaton can stay in one location for an arbitrary amount of time unless the invariant of the location is violated. To prevent the invariant from being violated, the automaton is forced to take a transition leading out of its current location. If several transitions are enabled at the same time, the automaton can choose non-deterministically, which one to take.

2.1. Networks and Communication Models

Usually, the systems we are considering are networks $\mathcal{N} := \{\text{TA}_1, \ldots, \text{TA}_n\}$ of several timed automata which run in parallel and can interact in multiple ways. Note, that in networks of timed automata we allow *global* clocks, that is all clock variables can be read and reset by any subautomaton. So, instead of defining X_j for each subautomaton TA_j, one set $X_{\mathcal{N}}$ for all clock variables is introduced. For the *synchronization model*, we extend the network $\mathcal{N}$ of timed automata by a set of



(a) Synchronization (b) Bounded Integer Communication

Figure 1: Communication models

so-called channels $\text{Ch}_\mathcal{N}$. In this extended model, edges in several subautomata can be labeled together in one channel $ch \in \text{Ch}_\mathcal{N}$ if they are to be executed simultaneously. This means that an edge within a channel can only be taken if it happens synchronously with all other edges labeled with the same channel in other subautomata[1]. Of course all guards as well as all invariants of the successor states have to be satisfied for this to happen. Figure 1(a) illustrates a synchronization of two subautomata TA_p and TA_q via the channel ch. If these subautomata are in the locations l_p and l_q respectively, and both guards g_p and g_q are satisfied, then either both transitions can be taken simultaneously, or none of them is taken. In many real-world designs *bounded integer variables* are used for communication of timed automata. They can be read and written to by any timed automaton of the network, and will keep their value until they are re-assigned. Bounded integer variables have a minimum and a maximum value that they can be assigned. Furthermore, they must be initialized to a certain value (default is the lower bound). Let $V_\mathcal{N} := \{v_1, \ldots, v_m\}$ be the set of global integer variables for the network of timed automata $\mathcal{N}$, with each v_k, $1 \leq k \leq m$ having a lower bound $v_{\text{k,min}}$ and an upper bound $v_{\text{k,max}}$ and an initial value $v_{\text{k,init}}$. For the sake of convenience we limit assignments ($v_k := c$) and comparisons ($v_k == c$) of bounded integer variables to constants $c \in \mathbb{N}$ with $v_{\text{k,min}} \leq c \leq v_{\text{k,max}}$. Figure 1(b) shows an example for a bounded integer variable communication. First the transition in automaton TA_p has to be taken and thus, the variable v is assigned to a constant c. Then, an arbitrary amount of time may pass until the transition on automaton TA_q is taken.

3. Encoding of the Bounded Model Checking Problem

The BMC procedure iteratively searches for a counterexample starting at length 0, and up to a predefined length k. Using the initial state I^0 (upper index denotes the unfolding depth), the system is unfolded by a conjunction of transition relations $T^{i,i+1}$ and the negated property $\neg P^k$. This formula $I^0 \wedge T^{0,1} \wedge \ldots \wedge T^{k-1,k} \wedge \neg P^k$ is a satisfiability problem, which is satisfied if there is a path of length k leading to a state violating the property. We encode the BMC problem for a network of timed automata $\mathcal{N}$ according to an interleaving semantics similar to [11]. Additionally, unless writing conflicts on clock variables or bounded integer variables occur, we allow parallel transitions. Preserving the conventional interleaving semantics can be achieved by a slight modification of the formulas described in this section. For the locations of each subautomaton TA_p of the network, we introduce new boolean variables at_p which represent its binary encoding.

In the initial state of the network all subautomata are in their initial location and all clocks are set to zero. Furthermore, all bounded integer variables are set to their initial value:

$$I^0 \;:=\; \bigwedge_{j=1}^{n} (at_j^0 = l_{j,0}) \wedge \bigwedge_{x \in X_\mathcal{N}} (x^0 = 0) \wedge \bigwedge_{v \in V_\mathcal{N}} (v^0 = v_{\text{init}})$$

[1] In UPPAAL [12] a similar synchronization model is used, where exactly two automata synchronize on one channel (so-called binary channels).

For the transition relation, we have to differentiate between a discrete step $T^{i,i+1}_{\text{jump}}$, which describes all possibilities of changing a location, and a continuous step $T^{i,i+1}_{\text{flow}}$ describing the passing of time. We introduce boolean variables $r_{x,j}$ which are *true* if clock x is reset in subautomaton TA_j, and *false* otherwise. In this way, we can later define a *noreset*-formula which ensures that un-reset clocks keep their values. Since bounded integer variables are global variables, we can encode assignments in the same manner as the reset of clock variables by introducing new boolean variables z ($z_{v,j}$ is *true*, if the bounded integer variable v was assigned to in subautomaton TA_j). Similar to the *noreset*-formula we will then define a *noassign*-formula for bounded integer variables which do not change their values. Additionally, in our model of timed automata an edge can be labeled with one synchronization mark $ch \in \text{Ch}_j$ (to keep it simple we allow edges to be labeled with at most one channel). Synchronized transitions are encoded by again introducing a new boolean variable for each channel. We then simply set the boolean variable corresponding to ch to *true* and the variables corresponding to all other channels occurring in TA_j to *false* (for edges not labeled with a channel, all channels of TA_j are set to *false*). Using these extensions it is ensured that all transitions within ch are executed simultaneously. In the communication model using bounded integer variables, edges are extended by comparisons (as guards of transitions) and assignments (as actions of transitions) of bounded integer variables to constants. Let A_e (resp. C_e) be the set of bounded integer variables which are assigned (resp. compared) on one edge e and c_v the corresponding constant for each $v \in A_e$ (resp. $v \in C_e$). For each edge $e = \langle l, g, C_e, res, ch, A_e, l' \rangle$ in a subautomaton TA_j we encode the transition as follows:

$$\begin{aligned} T^{i,i+1}_{\text{jump}}(e,j) \;:=\; & (at^i_j = l) \wedge g^i \wedge \bigwedge_{v \in C_e} (v^i = c_v) \wedge (at^{i+1}_j = l') \wedge Inv^{i+1}(l') \wedge ch^i \wedge \bigwedge_{d \in \text{Ch}_j \setminus \{ch\}} \neg d^i \\ & \wedge \bigwedge_{x \in res} \big(r^i_{x,j} \wedge (x^{i+1} = 0)\big) \wedge \bigwedge_{x \in X_{\mathcal{N}} \setminus res} \neg r^i_{x,j} \\ & \wedge \bigwedge_{v \in A_e} \big(z^i_{v,j} \wedge (v^{i+1} = c_v)\big) \wedge \bigwedge_{w \in V_{\mathcal{N}} \setminus A_e} \neg z^i_{w,j} \end{aligned}$$

To consider the case that one subautomaton does not perform a transition, it stays in its current location and no clocks are reset, no bounded integer variables are assigned and no synchronization is performed:

$$\textit{fix}^{i,i+1}(j) \;:=\; (at^{i+1}_j = at^i_j) \wedge \bigwedge_{x \in X_{\mathcal{N}}} \neg r^i_{x,j} \wedge \bigwedge_{w \in V_{\mathcal{N}}} \neg z^i_{w,j} \wedge \bigwedge_{d \in \text{Ch}_j} \neg d^i$$

Next, we can specify the *noreset*- and *noassign*-formulas mentioned above. Furthermore a *range*-formula to ensure the bounds of the integer variables is defined:

$$\begin{aligned} \textit{noreset}^{i,i+1} \;&:=\; \bigwedge_{x \in X_{\mathcal{N}}} \Big(\big(\bigwedge_{j=1}^{n} \neg r^i_{x,j} \big) \implies (x^{i+1} = x^i) \Big) \\ \textit{noassign}^{i,i+1} \;&:=\; \bigwedge_{v \in V_{\mathcal{N}}} \Big(\big(\bigwedge_{j=1}^{n} \neg z^i_{v,j} \big) \implies (v^{i+1} = v^i) \Big) \\ \textit{range}^i \;&:=\; \bigwedge_{v \in V_{\mathcal{N}}} \big((v^i \geq v_{\text{min}}) \wedge (v^i \leq v_{\text{max}}) \big) \end{aligned}$$

Finally, the discrete step $T^{i,i+1}_{\text{jump}}$ of the whole network is encoded as follows:

$$T^{i,i+1}_{\text{jump}} \;:=\; \bigwedge_{j=1}^{n} \Big(\bigvee_{e \in E_j} T^{i,i+1}_{\text{jump}}(e,j) \vee \textit{fix}^{i,i+1}(j) \Big) \wedge \textit{noreset}^{i,i+1} \wedge \textit{noassign}^{i,i+1} \wedge \textit{range}^{i,i+1}$$

A continuous step of the network simply states that all subautomata stay in their locations and time passes equally for all clocks while no invariants are violated. Additionally, all bounded integer variables keep their values:

$$T_{\text{flow}}^{i,i+1} := \bigwedge_{j=1}^{n} \Big((at_j^{i+1} = at_j^i) \wedge \bigwedge_{l \in L_j} \big((at_j^{i+1} = l) \implies Inv^{i+1}(l) \big) \Big) \\ \wedge (\lambda^i > 0) \wedge \bigwedge_{x \in X_{\mathcal{N}}} (x^{i+1} = x^i + \lambda^i) \wedge \bigwedge_{v \in V_{\mathcal{N}}} (v^{i+1} = v^i)$$

Since we restrict ourselves to the reachability of locations of the subautomata, we allow properties P of the form $P := l \mid \neg P \mid P_1 \vee P_2$ with $l \in \bigcup_{i=1}^{n} L_i$.
In each transition step of the network, a discrete step or a continuous step can be performed. So we construct the final formula $BMC(k)$ for the k-th unfolding of the BMC problem as follows:

$$BMC(k) := I^0 \wedge \bigwedge_{i=0}^{k-1} (T_{\text{jump}}^{i,i+1} \vee T_{\text{flow}}^{i,i+1}) \wedge \neg P^k$$

If $BMC(k)$ is satisfiable, there is a run $r = \langle s_0, s_1, \ldots, s_k \rangle$ with $s_i = \langle l_i, \nu_i \rangle$, $0 \leq i \leq k$ of length k leading into a state s_k violating the property P.

4. Bounded Model Checking of Incomplete Networks of Timed Automata

In a network of timed automata we model whole subautomata as blackboxes, allowing both synchronization labels and bounded integer variables to act as interfaces between the blackboxes and the implemented system. Let $\mathcal{N}^{\text{bb}} \subset \mathcal{N}$ be the set of blackboxed automata. While the behavior of these subautomata is unknown, the interface to the remaining network is defined. More precisely, we know all synchronization channels the blackbox can possibly synchronize on ($\text{Ch}^{\text{bb}} \subseteq \text{Ch}_{\mathcal{N}}$) and all bounded integer variables which could possibly be assigned in the blackbox ($V^{\text{bb}} \subseteq V_{\mathcal{N}}$).
Here, we only build the BMC formula for $\mathcal{N} \setminus \mathcal{N}^{\text{bb}}$. Similarly, we can ignore all sets of clocks $X^{\text{bb}} \subseteq X_{\mathcal{N}}$ only occurring in the blackboxed automata. Note, that we require that clocks which are read in the implemented system are not reset in blackboxes. Otherwise, this would lead to a universal quantification of real-valued variables which is beyond the scope of this paper.

4.1. Blackbox Bounded Model Checking based on Fixed Transitions

Our first approach to prove unrealizability of incomplete networks of timed automata relies on the observation that it must be possible to find a counterexample which is *independent* of the implementation of the blackboxes. One way to achieve this, is to find a path into the bad states using transitions which are neither labeled with a blackboxed channel, nor labeled with comparisons of blackboxed bounded integer variables to constants. We call these transitions *fixed*. All other transitions may be blocked by a corresponding behavior of the blackboxes. Thus, when a counterexample is found which is only based on fixed transitions, the property is not realizable.
In our encoding, we adapt $T_{\text{jump}}^{i,i+1}(e,j)$ for any edge $e = \langle l, g, C_e, res, ch, A_e, l' \rangle$ of the remaining implemented automata as follows:

$$T_{\text{jump,bb}}^{i,i+1}(e,j) := \begin{cases} \mathit{false}, & \text{if } (ch \in \text{Ch}^{\text{bb}}) \vee (C_e \cap V^{\text{bb}} \neq \emptyset) \\ T_{\text{jump}}^{i,i+1}(e,j), & \text{else} \end{cases}$$

Additionally, all assignments to variables $v \in V^{\text{bb}}$ in the implemented automata are removed.

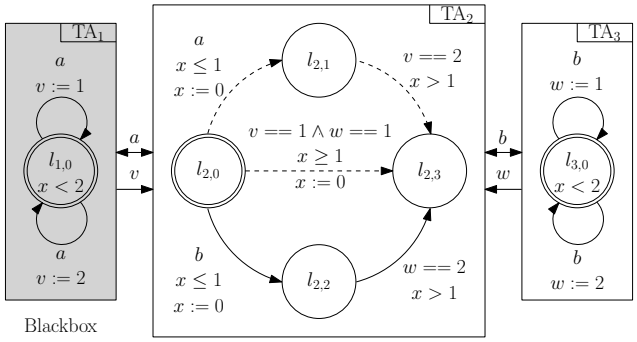


Figure 2: Incomplete network of interacting timed automata

Example

Figure 2 shows a network of three timed automata which interact over two channels a, b and two bounded integer variables v, w. There is one clock x, which is reset only in TA_2. Property P states that location $l_{2,3}$ of TA_2 can never be reached. Assume TA_1 is a blackbox, and that we only know the synchronization channel a, and the bounded integer variable v that it uses. Furthermore, in this example we have the additional constraint that v and w are within the range of 1 and 2. Now, only those transitions labeled neither with a nor with a comparison to v are considered when performing BMC (in Fig. 2 these fixed transitions are represented with solid arrows):

$$I^0 \;=\; (at_2^0 = l_{2,0}) \wedge (at_3^0 = l_{3,0}) \wedge (x^0 = 0) \wedge (w^0 = 1)$$

$$\begin{aligned} T_{\text{flow}}^{i,i+1} \;=\;& (at_2^{i+1} = at_2^i) \wedge (at_3^{i+1} = at_3^i) \wedge \big((at_3^{i+1} = l_{3,0}) \implies (x^{i+1} < 2)\big) \wedge \\ & (\lambda^i > 0) \wedge (x^{i+1} = x^i + \lambda^i) \wedge (w^{i+1} = w^i) \end{aligned}$$

$$\begin{aligned} T_{\text{jump}}^{i,i+1} \;=\;& \Big[\big[(at_2^i = l_{2,0}) \wedge (x^i \le 1) \wedge (at_2^{i+1} = l_{2,2}) \wedge r_{x,2}^i \wedge (x^{i+1} = 0) \wedge \neg z_{w,2}^i \wedge b^i\big] \\ & \vee \big[(at_2^i = l_{2,2}) \wedge (x^i > 1) \wedge (w^i = 2) \wedge (at_2^{i+1} = l_{2,3}) \wedge \neg r_{x,2}^i \wedge \neg z_{w,2}^i \wedge \neg b^i\big] \\ & \vee \big[(at_2^{i+1} = at_2^i) \wedge \neg r_{x,2}^i \wedge \neg z_{w,2}^i \wedge \neg b^i\big] \Big] \\ & \wedge \Big[\big[(at_3^i = l_{3,0}) \wedge (at_3^{i+1} = l_{3,0}) \wedge \neg r_{x,3}^i \wedge z_{w,3}^i \wedge (w^{i+1} = 1) \wedge b^i\big] \\ & \vee \big[(at_3^i = l_{3,0}) \wedge (at_3^{i+1} = l_{3,0}) \wedge \neg r_{x,3}^i \wedge z_{w,3}^i \wedge (w^{i+1} = 2) \wedge b^i\big] \\ & \vee \big[(at_3^{i+1} = at_3^i) \wedge \neg r_{x,3}^i \wedge \neg z_{w,3}^i \wedge \neg b^i\big] \Big] \\ & \wedge \Big[(\neg r_{x,2}^i \wedge \neg r_{x,3}^i) \implies (x^{i+1} = x^i)\Big] \wedge \Big[(\neg z_{x,2}^i \wedge \neg z_{x,3}^i) \implies (w^{i+1} = w^i)\Big] \\ & \wedge \Big[(w^{i+1} \ge w_{\min}) \wedge (w^{i+1} \le w_{\max})\Big] \end{aligned}$$

$$P^i \;=\; \neg(at_2^i = l_{2,3})$$

Obviously, the BMC formula for step 3 shown next is satisfied.

$$BMC(3) = I^0 \wedge (T_{\text{jump}}^{0,1} \vee T_{\text{flow}}^{0,1}) \wedge (T_{\text{jump}}^{1,2} \vee T_{\text{flow}}^{1,2}) \wedge (T_{\text{jump}}^{2,3} \vee T_{\text{flow}}^{2,3}) \wedge \neg P^3$$

One possible solution is the following run of the system (with $\langle at_2, at_3, x\rangle$):

$$\langle l_{2,0}, l_{3,0}, 0\rangle \xrightarrow{\text{jump(b)},w:=2} \langle l_{2,2}, l_{3,0}, 0\rangle \xrightarrow{\text{flow}} \langle l_{2,2}, l_{3,0}, 1.5\rangle \xrightarrow{\text{jump}} \langle l_{2,3}, l_{3,0}, 1.5\rangle$$

Since no single transition depends on a or v, this run can occur regardless of the implementation of the blackbox (thus, the property is unrealizable). Furthermore, this encoding of fixed transitions yields a conventional SMT formula and state-of-the-art SMT solvers like Yices [13] or MathSAT [14] can be used.

Consider a slight modification of the example in Fig. 2 where the edge leading from $l_{2,0}$ to $l_{2,2}$ in TA_2 is additionally labeled with the comparison $v == 2$ (a blackboxed bounded integer variable). In this situation, there is no single fixed transition leading out of the initial location $l_{2,0}$, and thus, no counterexample can be found performing BMC using the fixed transitions mentioned above. To overcome this problem, we now present a BMC encoding based on universal quantification of bounded integer variables. Note, that these problems cannot be solved with the conventional SMT solvers mentioned above.

4.2. Blackbox Bounded Model Checking based on Universal Quantification

In some situations it is not possible to prove unrealizability when considering fixed transitions based on blackboxed channels and blackboxed bounded integer variables as defined in the previous section. Since we allow for arbitrary behavior of the blackboxes, the blackboxed bounded integer variables can hold any value at any particular time. That is, we can prove unrealizability of a property, if we find a path into the bad states *for all* possible values of these variables. This can be achieved by extending the encoding by introducing universal quantification of blackboxed bounded integer variables. Thus, for T_{jump} we consider fixed transitions which are based on blackboxed synchronization channels only, and similarly to [8,9] we build a quantifier prefix for the BMC formula as follows:

$$\underbrace{\exists at^0 x^0\ \forall v_{\text{bb}}^0\ \exists v^0 ch^0 \lambda^0 r_x^0 z_v^0 \overbrace{at^1 x^1}^{T_{1,2}^{\text{jump}} \vee T_{1,2}^{\text{flow}}}}_{T_{0,1}^{\text{jump}} \vee T_{0,1}^{\text{flow}}} \forall v_{\text{bb}}^1\ \exists v^1 ch^1 \lambda^1 r_x^1 z_v^1 \underbrace{at^2 x^2}_{T_{2,3}^{\text{jump}} \vee T_{2,3}^{\text{flow}}} \forall v_{\text{bb}}^2\ \exists v^2 ch^2 \lambda^2 r_x^2 z_v^2 at^3 x^3 \ldots$$

In other words, this quantifier prefix asks, whether there exists a state of the system at depth 0 ($\langle at^0 x^0 \rangle$) such that for all blackbox outputs at depth 0 (v_{bb}^0) there is an assignment to the v^0, ch^0, λ^0, r and z variables leading to a successor state at depth 1 ($\langle at^1 x^1 \rangle$) ..., such that the BMC formula $BMC(k)$ holds.

For every depth k the corresponding successor state for each v_{bb}^k can result either from a discrete step or from a continuous step. Note, that we also have to modify the BMC formula in some way. First, all assignments to the bounded integer variables in V^{bb} are removed from the initial state I^0, and the transition relations $T_{\text{flow}}^{i,i+1}$ and $T_{\text{jump}}^{i,i+1}$. This must be done because the blackboxes can assign any value to these variables at any time. Since we are interested in all values of the blackboxed bounded integer variables, which are within the predefined bounds of these variables, we extend the BMC formula for $k > 0$ in the following way:

$$BMC_{\text{bb}}(k) := \Big(\bigwedge_{j=0}^{k-1} \text{range}(V^{\text{bb},j}) \Big) \implies BMC(k)$$

Example

Figure 3 shows the same network of timed automata as in Fig. 2 with the modification described in the previous section. Now we consider all transitions which are not labeled with a (again, these fixed transitions are represented with solid arrows). To verify that we can reach location $l_{2,3}$ regardless of

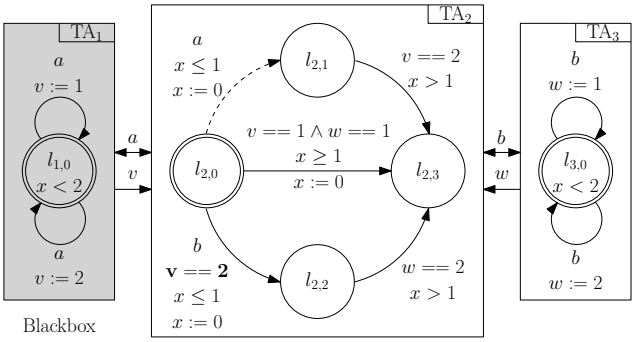


Figure 3: Modified incomplete network of interacting timed automata

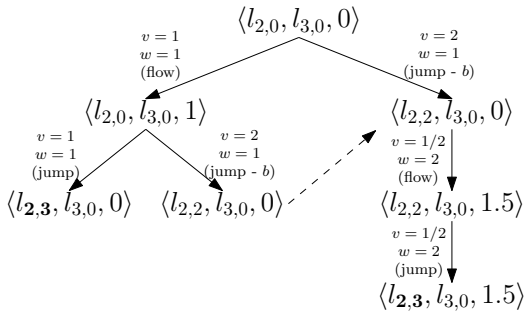


Figure 4: One possible solution tree for example in Fig. 3

the implementation of the blackbox, we build the BMC formula analog to the example in Figure 2, and let the solver try to find solutions for quantified SMT formulas as follows:

$$
\begin{aligned}
k = 0: \quad & \exists at_2^0, at_3^0, x^0 : BMC(0) \\
k = 1: \quad & \exists at_2^0, at_3^0, x^0 \forall v^0 \exists w^0, b^0, \lambda^0, r_{x,2/3,}^0, z_{w,2/3,}^0 \\
& \exists at_2^1, at_3^1, x^1 : BMC_{\text{bb}}(1) \\
k = 2: \quad & \exists at_2^0, at_3^0, x^0 \forall v^0 \exists w^0, b^0, \lambda^0, r_{x,2/3,}^0, z_{w,2/3,}^0 \\
& \exists at_2^1, at_3^1, x^1 \forall v^1 \exists w^1, b^1, \lambda^1, r_{x,2/3,}^1, z_{w,2/3,}^1 \\
& \exists at_2^2, at_3^2, x^2 : BMC_{\text{bb}}(2) \\
& \vdots
\end{aligned}
$$

Figure 4 shows one possible solution tree for this example at depth 4 (with $\langle at_2, at_3, x \rangle$). For each state, we have to find a successor state (following either a continuous or a discrete transition) for valid values of the blackboxed bounded integer variable v. So, for our example, starting from the initial state of the network $\langle l_{2,0}, l_{3,0}, 0 \rangle$, we can follow a discrete step for $v = 2$ using channel b to $\langle l_{2,2}, l_{3,0}, 0 \rangle$. For $v = 1$, we can perform a continuous transition resulting in $\langle l_{2,0}, l_{3,0}, 1 \rangle$. If we continue this, we will see that regardless of which value v is set to by the blackbox in each step, the network reaches a state violating the property after at most 4 steps.

5. Experimental Results

We implemented a prototype BMC tool to verify incomplete networks of timed automata. It uses the UTAP parser to read benchmarks in the UPPAAL file format and Yices, MathSAT and LIRA [15] as back-end solvers. To evaluate our methods for BMC based on fixed transitions we first used three instances of the Fischer mutual exclusion protocol with 10, 15 and 20 processes [16], interacting only via synchronization. The communication of the processes relies on a synchronization automaton which determines which process may enter the critical section. We inserted an error into each process by changing a clock constraint consisting of a strict inequality to a non-strict inequality and tested whether a network state can be reached where the first n processes are in the critical section at the same time. Second, we used two instances (15 and 20 processes) of a mutual exclusion protocol, where each process has its own arbiter. In contrast to the Fischer protocol, the subautomata are interacting via synchronization and bounded integer variables. Here, we verified that the first n processes can reach a safe location at the same time. All experiments were performed on an AMD Opteron

processes in total	n	k	complete system	incomplete system
10	2	12	0.23	0.05
	3	26	57.65	2.55
	4	40	9622.51	555.83
	5	-	-	-
15	2	12	0.65	0.10
	3	26	104.87	2.61
	4	40	19774.76	539.53
	5	-	-	-
20	2	12	1.41	0.16
	3	26	330.91	2.69
	4	40	29805.93	633.47
	5	-	-	-

(a) Results Fischer protocol

processes in total	n	k	complete system	incomplete system
15	3	22	8.05	0.26
	4	28	15.72	0.88
	5	34	29.87	3.59
	6	40	65.61	13.88
	7	46	129.32	42.32
	8	52	331.75	128.26
	9	58	661.99	346.98
	10	64	1686.08	1103.08
	11	70	4330.16	3807.03
	12	76	20647.73	18171.59
20	3	22	15.09	0.27
	4	28	36.37	0.95
	5	34	62.44	3.84
	6	40	127.72	12.69
	7	46	253.14	48.55
	8	52	520.18	156.49
	9	58	1317.66	640.81
	10	64	2717.79	1380.33
	11	70	5380.39	3809.41
	12	76	18557.84	9244.04

(b) Results Processes With Arbiters

Table 1: Experimental Results using Yices SMT Solver

processor running at 2.4 GHz with 4 GB of main memory and a timeout of 36,000 seconds[2].
Table 1(a) gives the results for the Fischer protocol for $n = 2, \ldots, 5$ using Yices. In the first column the total number of processes of the instance followed by the number of processes checked by the property can be found. The third column denotes the depth k where the counterexample was found. For the complete and incomplete cases where all processes not occurring in the property are black-boxed, the times are given in seconds. The results show the efficiency of abstracting all processes not occurring in the property. By using blackboxes we are able to prove unrealizability for all solvable instances in significant less time. On all benchmarks we are one to two orders of magnitude faster when verifying the incomplete system.
Table 1(b) gives the results for the processes with arbiters for $n = 3, \ldots, 12$. The columns are organized similar to Table 1(a) and all times are given in seconds. Here, in addition to the effectiveness of abstracting irrelevant processes, the results show that a higher speedup can be achieved when more processes are modeled as blackboxes.
Our first approach for BMC based on the universal quantification of bounded integer variables relies on Yices and MathSAT, as they offer an integer data type in their input language. Additionally, they allow universal and existential quantification of variables making it possible to form a quantifier prefix for the resulting SMT formula. But even for the very small example shown in Fig. 3, it was not possible to obtain a result. The solvers mentioned above return *UNKNOWN*, since they are not capable of resolving the universal quantifiers due to incompleteness. On the other hand, it also was not possible to prove unrealizability with a complete solver like LIRA, since the computational resources (out of memory) were exceeded at depth three. These results show that quantified SMT formulas resulting from the communication model based on bounded integer variables are more challenging.

[2]Note, as was introduced in [17] we use a slightly modified BMC formula alternating T^{jump} and T^{flow}, leading to an additional speedup of the whole process.

6. Conclusion

In this paper we introduced BMC for incomplete networks of timed automata. For the two communication models *synchronization* and *bounded integer communication* we presented approaches to prove unrealizability. BMC based on fixed transitions shows very promising preliminary results. For solving the class of BMC problems based on universal quantification, modifications to current solvers are necessary. At the moment we are working on transition relation methods for networks of timed automata where the bounded integer variables are binary encoded. In this case, only boolean variables would be universally quantified allowing us to use adequate data structures such as linAIGs [18] which can efficiently handle universally quantified boolean variables. Alternatively, we are working on a combination of a search-based QBF solver with a conventional SMT solver. This combination shows great promise in handling the problem class of quantified SMT formulas.

Acknowledgments

We would like to thank Karina Gitina and Matthew Lewis for fruitful discussions and Georges Morbé for providing the *processes with arbiters* benchmarks.

References

[1] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *TACAS*, 1999.

[2] E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 2001.

[3] M. Sorea. Bounded model checking for timed automata. In *ENTCS*, 2002.

[4] P. Niebert, M. Mahfoudh, E. Asarin, M. Bozga, O. Maler, and N. Jain. Verification of timed automata via satisfiability checking. In *Formal Techniques in Real-Time and FaultTolerant Systems FTRTFT*, 2002.

[5] B. Wozna, W. Penczek, and A. Zbrzezny. Checking reachability properties for timed automata via sat. *Fundamenta Informaticae*, 2002.

[6] G. Audemard, A. Cimatti, A. Kornilowicz, and R. Sebastiani. Bounded model checking for timed systems. In *Int'l Conf. on Formal Techniques for Networked and Distributed Systems*, 2002.

[7] M. Herbstritt and B. Becker. On SAT-based Bounded Invariant Checking of Blackbox Designs. In *MTV*, 2005.

[8] M. Herbstritt, B. Becker, and C. Scholl. Advanced SAT-techniques for bounded model checking of blackbox designs. In *MTV*, 2006.

[9] M. Herbstritt and B. Becker. On Combining 01X-Logic and QBF. In *EUROCAST*, 2007.

[10] C. Miller, K. Gitina, C. Scholl, and B. Becker. Bounded model checking of incomplete networks of timed automata. In *MTV*, 2010. to appear.

[11] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2000.

[12] K. G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Software Tools for Technology Transfer*, 1997.

[13] B. Dutertre and L. De Moura. The yices SMT solver. Technical report, SRI, 2006.

[14] R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, and R. Sebastiani. The mathsat 4 smt solver. In *Int'l Conf. on Computer Aided Verification*, 2008.

[15] B. Becker, C. Dax, J. Eisinger, and F. Klaedtke. LIRA: Handling constraints of linear arithmetics over the integers and the reals. In *Int'l Conf. on Computer Aided Verification*, 2007.

[16] L. Lamport. A fast mutual exclusion algorithm, 1986.

[17] E. Ábrahám, B. Becker, F. Klaedtke, and M. Steffen. Optimizing bounded model checking for linear hybrid systems. In *Int'l Conf. on Verification, Model Checking and Abstract Interpretation*, 2005.

[18] W. Damm, S. Disch, H. Hungar, S. Jacobs, J. Pang, F. Pigorsch, C. Scholl, U. Waldmann, and B. Wirtz. Exact state set representations in the verification of linear hybrid systems with large discrete state space. In *ATVA*, 2007.