Model Checking for Rectangular Hybrid Systems: A Quantified Encoding Approach

Luan V. Nguyen
Department of Computer Science
University of Dayton
OH, USA
lnguyen1@udayton.edu

Wesam Haddad
Department of Computer Science
University of Dayton
OH, USA
haddadw1@udayton.edu

Taylor T. Johnson
Department of Computer Science
Vanderbilt University
TN, USA
taylor.johnson@vanderbilt.edu

Satisfiability Modulo Theories (SMT) solvers have been successfully applied to solve many problems in formal verification such as bounded model checking (BMC) for many classes of systems from integrated circuits to cyber-physical systems. Typically, BMC is performed by checking satisfiability of a possibly long, but quantifier-free formula. However, BMC problems can naturally be encoded as quantified formulas over the number of BMC steps. In this approach, we then use decision procedures supporting quantifiers to check satisfiability of these quantified formulas. This approach has previously been applied to perform BMC using a Quantified Boolean Formula (QBF) encoding for purely discrete systems, and then discharges the QBF checks using QBF solvers. In this paper, we present a new quantified encoding of BMC for rectangular hybrid automata (RHA), which requires using more general logics due to the real (dense) time and real-valued state variables modeling continuous states. We have implemented a preliminary experimental prototype of the method using the HyST model transformation tool to generate the quantified BMC (QBMC) queries for the Z3 SMT solver. We describe experimental results on several timed and hybrid automata benchmarks, such as the Fischer and Lynch-Shavit mutual exclusion algorithms. We compare our approach to quantifier-free BMC approaches, such as those in the dReach tool that uses the dReal SMT solver, and the HyComp tool built on top of nuXmv that uses the MathSAT SMT solver. Based on our promising experimental results, QBMC may in the future be an effective and scalable analysis approach for RHA and other classes of hybrid automata as further improvements are made in quantifier handling in SMT solvers such as Z3.

1 Introduction

Boolean Satisfiability (SAT) is the canonical NP-complete problem and is to determine if a given Boolean formula is satisfiable, i.e., check if there exists an assignment of values to variables where the formula is true. A Boolean formula is given in Conjunctive Normal Form (CNF), that is, a conjunction of clauses, each of which is a disjunction of literals. Satisfiability Modulo Theories (SMT) is a generalization of SAT, where literals are interpreted with respect to a background theory (e.g., linear real arithmetic, nonlinear integer arithmetic, bit-vectors, etc.).

SMT-based techniques have been developed to formally verify hybrid systems [2, 9, 13, 16, 24, 30]. Typically, these SMT-based methods are used in bounded model checking (BMC), which is to check for a transition system A and a specification P whether $I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge (\bigvee_{i=0}^{k} P(V_i))$ is satisfiable. Here, $I(V_0)$ encodes an initial set of states over a set of variables V_0, $T(V_i, V_{i+1})$ represents the transition relation from iteration i to $i+1$ over sets of variables V_i and V_{i+1}, and $P(V_i)$ encodes the specification at step i. An SMT solver either returns SAT if there is a sequence of states leading the transition system A from a state in I to a state in P, or UNSAT if a state in P can not be reached in k steps. In principle, BMC is complete as it can prove that no bad state can be reached if a large enough bound k is used.

Remke & Tran (Eds.): Symbolic-Numeric Methods
for Reasoning about CPS and IoT (SNR'21)
EPTCS 361, 2022, pp. 9–23, doi:10.4204/EPTCS.361.4

However, as k increases, SMT-based BMC may require excessive memory due to the underlying complex combination theories in a SMT solver. To increase scalability, it is essential to reduce the memory usage while performing SMT-based BMC for large hybrid systems.

Hybrid automaton is a modeling formalism used to verify dynamical systems including both continuous states and dynamics as well as discrete states and transitions. Examples of systems naturally modeled by hybrid automata arise in the interaction of physical plants and software controllers in real-time systems and cyber-physical systems (CPS). In essence, hybrid automata augment finite state machines with a set of real-valued variables that evolve continuously over intervals of real time. In hybrid automata, a transition relation $T = D \cup \mathscr{T}$ encodes both discrete transitions D and continuous trajectories $\mathscr{T}$ over intervals of real-time. Rectangular hybrid automata (RHA) are a special class of hybrid automata with continuous dynamics described by rectangular differential inclusions and where all other quantities (guard conditions, invariants, resets, etc.) of the automata are linear inequalities over constants [19, 22]. Sets of states, as well as discrete transitions and continuous trajectories of RHA, can be symbolically represented by SMT formulas over real and Boolean variables.

Depending on the underlying logics supported, SMT solvers may or may not support quantifiers. While quantifiers may make the language more expressive, they often increase the complexity of computations like checking satisfiability and may also lead to undecidability. Techniques allowing quantifiers, such as in quantified Boolean formula (QBF) solvers, have been developed for the BMC of purely discrete systems, such as finite state machines [25, 28]. However, to the best of our knowledge, there has been no effort to develop quantified BMC (QBMC) methods for timed or hybrid automata, which we develop in this paper. Of course, this is partially because the underlying SMT solver requires support for complex combination theories and efficient algorithms to check quantified formulas, which until recently, were either not available or not scalable.

In this paper, we propose a SMT-based verification technique that encodes the BMC problem for timed automata and RHA in a quantified form, which we call QBMC (quantified bounded model checking). As the logic encoding requires some finite sort for the discrete states (such as a enumerated type or bit-vectors) and reals for the continuous states and trajectories, we uses LRABV (linear real arithmetic with bit-vectors) for encoding QBMC for timed automata and RHA. We note that general hybrid automata would need NRABV (nonlinear real arithmetic with bit-vectors) or beyond, such as those whose solutions involve special (transcendental) functions like sin, cos, exp, etc. While none of these logics are officially supported in the SMT-LIB2 standard as of the time of this writing [4], several solvers do have unofficial support for this combination theory, such as the latest versions of Z3, which is the SMT solver used in this paper [12].

For implementation, we take hybrid automata in the SpaceEx format [14], which are then translated to the QBMC encoding sequence proposed in this paper using the HyST model transformation tool [3]. HyST allows the same model to be analyzed simultaneously in several hybrid systems analysis tools, so it is convenient to compare the performance of our proposed approach with existing works. The QBMC is performed by querying the Z3 SMT solver via its Python API and using its quantifier-handling procedures [12]. We present preliminary experimental results where the QBMC approach and Z3 perform competitively to (a) the dReach tool that performs BMC using an SMT check by querying the dReal δ-decidable SMT solver [15, 16, 26], and (b) the HyComp tool built on top of nuXmv that uses the MathSAT SMT solver [10]. The examples include standard ones such as Fischer and Lynch-Shavit mutual exclusion, as well as an illustrative example to describe the encoding. Overall, the main contribution of this paper is the first encoding of BMC as a quantified problem for RHA that has less memory requirement and competitive execution time when compared to state-of-the-art SMT solvers. Our results subsume the case for timed automata, as RHA are more expressive than timed automata, and we note this is also the first QBMC approach for timed automata.

2 Related Work

When defining the semantics of hybrid automata, first-order or higher logic is typically used and quantifiers typically show up in several places. Existential quantifiers over reals are used to specify that some amount of real time may elapse in a given location of the hybrid automaton. Universal quantifiers over reals representing real time are used to construct invariants that are enforced at all times, while in a given location of the hybrid automaton; otherwise real time is not allowed to advance, and a discrete transition must be taken, if any are enabled based on the current state and guards of the transitions. Alternative approaches to the one described in this paper have previously been developed, where the universal quantifiers used to define invariant semantics are explicitly removed from the SMT expressions to create quantifier-free formulas. That allows the use of existing SMT-based procedures and avoids quantifier-elimination and other quantifier-handling procedures [11, 22]. We note those approaches do not use quantifiers on the number of steps $k \geq 0$ in the BMC computation, which we do in this paper. Specifically, we suggest that effectively encoding the BMC problem in a quantified form over the number of steps k may provide a more scalable approach in the future as quantifier handling procedures are improved in the underlying solvers. We accomplish this by extending existing results for the BMC of discrete systems with QBF solvers [28] to timed and hybrid automata, specifically RHA.

Typical approaches that analyze timed and hybrid automata use symbolic representations of states such as difference bound matrices (DBMs) to represent clock regions in Uppaal [6], HybridSAL [31], or polyhedra in HyTech [20]. Several other formal verification tools for hybrid automata focus on performing reachability computations, and overapproximate the set of reachable states using various data structures to symbolically represent geometric sets of states, such as Taylor models in Flow* [8] and support functions in SpaceEx [14]. Reachability analysis tools like Flow* and SpaceEx focus on computing reachable states, although there is a direct equivalence between time-bounded reachability computations and BMC.

Several SMT-based approaches can verify properties of timed and hybrid automata. dReal is an SMT-solver for first-order logic formulas over the reals, and uses a δ-complete decision procedure [15]. dReach is a BMC tool that queries dReal to check satisfiability of SMT formulas encoding the transitions and trajectories for hybrid automata [16]. HyComp is a verification tool for networks (parallel compositions) of hybrid automata with polynomial and other dynamics [9] and is built on top of nuXmv [7]. HyComp supports several verification modes, including a BMC analysis mode, k-induction, and IC3. For k-induction and IC3, HyComp may perform unbounded model checking, but in the BMC mode, it also allows a limit on the number of steps, and also encodes the semantics of the network of hybrid automata's transition relation and trajectories. In this paper, we will compare our QBMC approach to dReach and HyComp in terms of performing BMC for hybrid systems.

A very closely related approach to this paper also encodes BMC problems for timed automata using quantified formulas, but this quantification is to encode unknown or incomplete components, and is not a quantification over the BMC length [29]. Passel is a parameterized verification tool for networks of RHA that may prove properties regardless of the number N of automata in the network [22]. Passel implements an extension to hybrid automata of the invisible invariants approach for parameterized verification, and consists of an invariant synthesis procedure [23] that relies on reachability computations [24]. Passel encodes the semantics of networks of hybrid automata as SMT formulas and checks satisfiability and validity using the Z3 SMT solver. When performing reachability computations, Passel makes use of quantifier elimination procedures over the reals and bit-vectors [24]. MCMT is another SMT-based verification tool equipped with an extensive quantifier instantiation to verify the safety properties of parametrized systems [18]. In contrast, our proposed approach straightforwardly deals with quantified SMT formulas.

3 Preliminaries

In this section, we introduce the preliminaries that are needed for this work. We first define a hybrid automaton model, discuss its semantics and safety specification. Then we present the traditional encoding (quantifier-free) of the BMC for hybrid automata.

3.1 Hybrid Automata

Syntax. A hybrid automaton is essentially a finite state machine extended with a set of real-valued variables that evolve continuously over intervals of real-time. The syntactic structure of a hybrid automaton is formally defined as follows. A hybrid automaton $\mathscr{H}$ is a tuple, $\mathscr{H} \triangleq \langle Loc, Var, Inv, Flow, Trans, Init\rangle$, with the components as follows.

- *Loc* is a finite set of discrete locations.
- *Var* is a finite set of n continuous, real-valued variables, and $\mathscr{Q} \triangleq Loc \times \mathbb{R}^n$ is the state-space.
- *Inv* is a finite set of invariants, one for each discrete location, and for each location $\ell \in Loc$, $Inv(\ell) \subseteq \mathbb{R}^n$.
- *Flow* is a finite set of ordinary differential inclusions, one for each continuous variable $x \in Var$, and $Flow(\ell, x) \subseteq \mathbb{R}^n$ describes the continuous dynamics in each location $\ell \in Loc$.
- *Trans* is a finite set of transitions between locations. Each transition is a tuple $\tau \triangleq \langle \ell, \ell', g, u\rangle$, where ℓ is a source location and ℓ' is a target location that may be taken when a guard condition g is satisfied, and the post-state is updated by an update map u.
- *Init* is an initial condition, which consists of a set of locations in *Loc* and a formula over *Var*, so that $Init \subseteq \mathscr{Q}$.

For RHA, all the expressions appearing in invariants, guards, and updates must be boolean combinations of constant inequalities, and the flows are rectangular differential inclusions e.g., $\dot{x} \in [a, b]$ for $a \leq b$. Timed automata is the important subclass of RHA in which every continuous variable is a precise clock e.g., $\dot{x} = 1$. We use the dot (.) notation to refer to different components of tuples e.g., $\mathscr{H}.Inv$ refers to the invariants of automaton $\mathscr{H}$ and $\tau.g$ refers to the guard of a transition τ. If clear from context, we drop $\mathscr{H}$ and τ and refer to the individual components of the tuple.

Semantics. The semantics of a hybrid automaton $\mathscr{H}$ are defined in terms of executions, which are sequences of states. A *state* q of $\mathscr{H}$ is a tuple $q \triangleq \langle \ell, v\rangle$, where $\ell \in Loc$ is a location, and $v \in \mathbb{R}^n$ is a valuation of all variables in *Var*. Formally, for a set of variables *Var*, a *valuation* is a function mapping each $x \in Var$ to a point in its type—here, $\mathbb{R}$. The state-space $\mathscr{Q}$ is the set of all states of $\mathscr{H}$. Updates of states are described by a transition relation $T \subseteq \mathscr{Q} \times \mathscr{Q}$. For a transition $\langle q, q'\rangle \in T$ where $q \triangleq \langle \ell, v\rangle$ and $q' \triangleq \langle \ell', v'\rangle$, we denote $q \to q' \in T$ as the transition between the current state q and the next state q'. The transition relation T is partitioned into disjoint sets of discrete transitions and continuous trajectories that respectively describe the discrete and continuous behaviors of the automaton. Thus,

$T \triangleq D \cup \mathscr{T}$, where: (a) $D \subseteq \mathscr{Q} \times \mathscr{Q}$ is the set of discrete transitions that describe instantaneous updates of state, (b) $\mathscr{T} \subseteq \mathscr{Q} \times \mathscr{Q}$ is the set of continuous trajectories that describe updates of state over real time intervals.

Discrete transitions. A discrete transition $q \to q' \in D$ models an instantaneous update from the current state q to the next state q'. There is a discrete transition $q \to q' \in D$ if and only if (iff): $\exists \tau \in Trans : q.v \models \tau.g \wedge q'.v' \models \tau.u$, where $\tau.g$, and $\tau.u$ are the guard condition and the update map of the discrete transition τ, respectively.

Continuous trajectories. A continuous trajectory $q \to q' \in \mathscr{T}$ models the update of state q to q' over an interval of real time. The set-valued function Δ returns a set of states and is defined as: $\Delta(q.\ell, q.v, x, t) \in q.v.x + \int_{\delta=t_0}^{t} f(q.\ell, x) d\delta$, where $f \in Flow$ is a flow rate—a formula over $Var \cup \dot{Var}$ that describes the evolution of a real variables $x \in Var$ over a real time interval $J = [t_0, t]$— and $q.v.x$ is the value of continuous variable x of the state q at $t = t_0$. Then, there is a trajectory $q \to q' \in \mathscr{T}$ iff: $\exists t_\alpha \in \mathbb{R}_{\geq 0}\ \forall t_\beta \in \mathbb{R}_{\geq 0}\ \exists \ell \in Loc\ : t_\beta \leq t_\alpha\ \wedge\ \Delta(q.\ell, q.v, Var, t_\beta) \models Inv(\ell)\ \wedge\ q'.v'.Var \in \Delta(q.\ell, q.v, Var, t_\alpha)$. For each real variable x, $q.v.x$ must evolve to the valuation $q'.v'.x$ at precisely time t_α and corresponding to the flow rate of x in location ℓ. Additionally, all states along the trajectory must satisfy the invariant $Inv(\ell)$ i.e., at every point in the interval of real time $t_\beta \leq t_\alpha$.

Executions. An *execution* of $\mathscr{H}$ is a sequence $\pi \triangleq q_0 \to q_1 \to q_2 \to \ldots$, such that: (a) $q_0 \in Init$ is an initial state, and (b) either $q_i \to q_{i+1} \in D$ is a discrete transition or $q_i \to q_{i+1} \in \mathscr{T}$ is a continuous trajectory for each consecutive pair of states in the sequence π. A state $q_k \triangleq \langle \ell_k, v_k \rangle$ is *reachable* from initial state $q_0 \triangleq \langle \ell_0, v_0 \rangle \in Init$ iff there exists a finite execution $\pi \triangleq q_0 \to q_1 \to \ldots \to q_k$.

Safety specifications. In this paper, we develop the QBMC procedure to check whether safety properties of hybrid automata are satisfied up to iteration k. A *safety specification* ϕ is a formula over *Loc* and *Var* that describes a set of states $[\![\phi]\!] \subseteq \mathscr{Q}$, where $[\![\cdot]\!]$ is the set of states satisfying ϕ. For an automaton $\mathscr{H}$ and a safety specification ϕ, the automaton satisfies the specification, denoted $\mathscr{H} \models \phi$, iff for every execution π, for every state $q_0, q_1, \ldots, q_k$ in the execution π, we have $\pi.q_k \in [\![\phi]\!]$. If $\mathscr{H} \models \phi$ for every $i \in \{0, \ldots, k\}$, then the system is safe up to iteration k. If $\mathscr{H} \models \phi$ for any k, then the system is safe. For a safety specification ϕ, a *counterexample* is an execution π where some state $q \in \pi$ violates ϕ, i.e., $q \not\models \phi$, or equivalently, $q \notin [\![\phi]\!]$.

3.2 Quantifier-Free BMC for Hybrid Automata

BMC has been used widely in verification and falsification of safety and liveness properties of various classes of systems, from finite state machines to hybrid automata. The key idea is to search for a counterexample execution whose length is bounded by a number of steps k. In other words, BMC will explore all executions from any initial state of a system to detect whether there is a way to reach a bad state that violates a given property (or to find a loop in the case of liveness). Then this path is considered as a counterexample to the property that may help the user to debug the system. For finite state systems, BMC can be encoded as a propositional formula to be checked as satisfiable or unsatisfiable using a Boolean SAT solver. For hybrid automata, BMC can be encoded as a formula over reals and finite sorts (such as Booleans, bit-vectors, or enumerated types). In this paper, we focus only on hybrid automata with rectangular differential inclusion dynamics ($\dot{x} \in [a, b]$ for real constants $a \leq b$), and for this class of automata, the formulas are within linear real arithmetic (LRA). Before presenting the QBMC approach

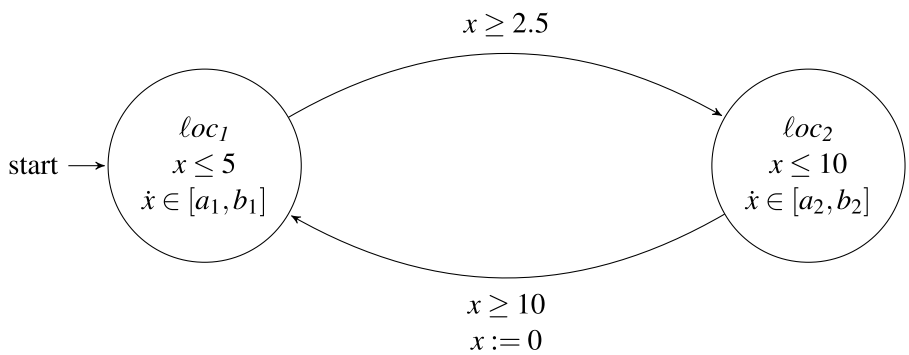

Figure 1: The hybrid automaton $\mathscr{H}$ for Example 3.2.

in Section 4, which is the main contribution of this paper, we first illustrate BMC for hybrid automata using the traditional quantifier-free encoding.

Let P be a set of given specifications of the hybrid automata, the BMC problem will determine whether a specification $P(q_k) \in P$ is safe after k steps, and it is:

$$\Phi(k) \triangleq I(V_0) \wedge \bigwedge_{i=0}^{k-1} T_i(V_i, V_{i+1}) \wedge (\bigvee_{i=0}^{k} P(V_i)), \tag{1}$$

where V_i corresponds to the set of variables *Var* of the automaton $\mathscr{H}$ appropriately renamed. In Equation 1, $I(V_0)$ encodes the initial set of states, $T_i(V_i, V_{i+1})$ encodes the transition between consecutive pairs of sets of states, and $P(V_i)$ is a safety specification at iteration i. We note that the sets of variables V_i for each iteration i are implicitly existentially quantified, e.g., we could equivalently prefix $\exists V_0, V_1, \ldots, V_k$. We drop the sets of variables for a shorter notation, e.g., Equation 1 is equivalent to $I_0 \wedge \bigwedge_{i=0}^{k-1} T_i \wedge (\bigvee_{i=0}^{k} P_i)$.

Illustrative Example. Consider the hybrid automaton $\mathscr{H}$ shown in Figure 1. Assume that the automaton starts at location ℓoc_1, and the initial value of x is 0. Two intervals $[a_1, b_1]$ and $[a_2, b_2]$ describe the rectangular differential inclusions for locations ℓoc_1, and ℓoc_2, respectively. This automaton would be a *timed automaton* if all of the constants values are equal, i.e., $a_1 = b_1 = a_2 = b_2$. This automaton would be a *multi-rate timed automaton* if $a_1 = b_1$ and $a_2 = b_2$ but possibly $a_1 \neq a_2$. Otherwise, this automaton is a *rectangular hybrid automaton*. Suppose that $a_1 = 1$, $b_1 = 2$, $a_2 = 3$, and $b_2 = 4$. We introduce $k+1$ copies $x_0, x_1, ..., x_k$ and $\ell_0, \ell_1, ..., \ell_k$, where the variable x_i gives the value of the variable x, and ℓ_i indicates the location at the state q_i, representing the i^{th} step of the BMC computation for the automaton shown in Figure 1. The set of bad states are defined by:

$$P \triangleq \bigvee_{i=0}^{k} q_i.\ell_i = \ell oc_2 \ \wedge \ x < 2.5. \tag{2}$$

The BMC computation of $\mathscr{H}$ for each k up to 2 can be encoded as:

- $k = 0$: $I_0 := (\ell_0 = \ell oc_1 \wedge x_0 = 0)$;
- $k = 1$ (D_0): $(\ell_0 = \ell oc_1 \wedge \ell_1 = \ell oc_2 \wedge x_0 \leq 5 \wedge x_0 \geq 2.5 \wedge x_1 = x_0)$,
- $k = 1$ ($\mathscr{T}_0$): $(\ell_0 = \ell oc_1 \implies (\ell_1 = \ell_0 \wedge x_0 + a_1\delta \leq x_1 \wedge x_1 \leq x_0 + b_1\delta \wedge x_1 \leq 5))$,

- $k = 2$ (D_1): $(\ell_1 = \ell oc_1 \wedge \ell_2 = \ell oc_2 \wedge x_1 \leq 5 \wedge x_1 \geq 2.5 \wedge x_2 = x_1)$,
- $k = 2$ ($\mathscr{T}_1$): $(\ell_1 = \ell oc_1 \implies (\ell_2 = \ell_1 \wedge x_1 + a_1\delta \leq x_2 \wedge x_2 \leq x_1 + b_1\delta \wedge x_2 \leq 5))$,

where δ is a fresh, real constant. In general, a universally quantified assertion that the invariant is satisfied for every real time along the trajectory from time t_0 to time $t_0 + \delta$, although this is unnecessary for rectangular differential inclusions with linear guards and invariants for convexity reasons [9, 22], which makes this assertion fall into the combination theory of linear real arithmetic with bit-vectors (or some finite sort to encode the locations). We split the discrete transitions and trajectories for clarity, but the entire formula to be checked for iteration $k = 1$ would just be the disjunction of these conjuncted with the formula representing $k = 0$ and the bad set of states, i.e., $I_0 \wedge (D_0 \vee \mathscr{T}_0) \wedge P$. For $k = 2$, this full formula would be $I_0 \wedge (D_0 \vee \mathscr{T}_0) \wedge (D_1 \vee \mathscr{T}_1) \wedge P$.

For $k = 1$, we dropped the obviously infeasible transition from ℓoc_2 to ℓoc_1 from D_0, which would be found as being unsatisfiable since $\ell_0 \neq \ell oc_2$. However, the transition from ℓoc_1 to ℓoc_2 also cannot occur since $x_0 = 0$, but $x_0 \ngeq 2.5$, so that this part is unsatisfiable and no discrete transitions may be taken from the set of initial states. We also dropped the continuous dynamics for ℓoc_2 from $\mathscr{T}_0$ since this would also be infeasible since $\ell_0 \neq \ell oc_2$. However, real time may elapse, and as encoded, would correspond to any choice of time δ such that $x_1 \in [a_1\delta, b_1\delta]$ and $x_1 \leq 5$. Since $a_1 = 1$ and $b_1 = 2$, at most between 2.5 and 5 seconds of real time could elapse, and either case would yield $x_1 \in [0, 5]$.

For $k = 2$, we also dropped the infeasible transition and trajectory for clarity. In this case, the transition from ℓoc_1 to ℓoc_2 is enabled since $x_1 \in [0, 5]$, so the update to ℓoc_2 may occur. However, now the continuous trajectory would be infeasible since x_1 could already be 5 and the invariant requires $x_2 \leq 5$, so no real-time $\delta > 0$ may elapse, as otherwise $x_1 + a_1\delta \leq 5$ is unsatisfiable for $x_1 = 5$. So, the only state update would be to ℓoc_2 owing to the discrete transition.

4 Quantified BMC for Hybrid Automata

The idea of the quantified SMT-based BMC for hybrid automata presented in this paper was inspired from the compact QBF encodings for addressing the BMC problem of purely discrete systems [21, 25, 28]. To construct a quantified formula $\Omega(k)$ for the BMC of $\mathscr{H}$ of length k, we introduce a *bit-vector* $\vec{t} = \langle t_1, t_2, ..., t_{\lceil \log_2 k \rceil} \rangle$ to index each iteration of the BMC. The next-state of each iteration is connected to the current-state of the next iteration using two multiplexers, where the vector $\vec{t}$ functions as common select lines. We do not describe those multiplexers here but refer to its similar presentation in [28]. Hence, depending on the truth assignment given to the vector $\vec{t}$, the single copy of a transition relation T simulates different iteration of the BMC. The QBMC formula is then given as:

$$\Omega(k) \triangleq \exists V_0, V_1, ..., V_k, \delta \, \forall \vec{t} \, \exists V, V' \mid I(V_0) \wedge T(V, V') \wedge \bigwedge_{i=0}^{k-1} t^k(i) \rightarrow [(V = V_i) \wedge (V' = V_{i+1})] \wedge (\bigvee_{i=0}^{k} P(V_i)), \quad (3)$$

where we note that the existential δ encodes the real time elapse and would appear in the trajectories $\mathscr{T}$ of the disjunct $T = D \vee \mathscr{T}$.

In the prefix of $\Omega(k)$, the existential variables V_i is understood to *dominate* the universal $\vec{t}$ to ensure state contiguity. Intuitively, if there exists a truth assignment to each existential variable V_i, then the quantified first order formulas in SMT will be satisfied for all universal variable assignments. Here, the current state q and the next state q' under the transition relation $T(V, V')$ are connected to the current state and the next state for each particular iteration $t^k(i)$, which is associated with a truth assignment given to $\vec{t}$. This allows the QBMC of hybrid automaton $\mathscr{H}$ to be compactly encoded using only a single copy of a transition relation with adding some state variable equalities at each time. Whereas, the quantifier-free encoding in Equation 1 requires k copies of the transition relation T. This advantage of non-coping transition relation encoding has been proved to significantly reduce the problem size of the QBF encoding in [28]. Similarly, the proposed QBMC method also requires less memory usage compared to other quantifier-free SMT-based BMC approaches, that will be demonstrated in details later in Section 5.

To illustrate Equation 3, we consider the QBMC of the hybrid automaton of Example 3.2 with $k = 3$ as follows:

$$\begin{aligned}\Omega(3) = \exists V_0, V_1, V_2, V_3, \delta \forall t_1, t_2 \exists V, V' \mid\, & I(V_0) \wedge T(V, V') \\ & \wedge \{\bar{t_1} \rightarrow [(V = V_0) \wedge (V' = V_1)]\} \\ & \wedge \{t_1 \wedge \bar{t_2} \rightarrow [(V = V_1) \wedge (V' = V_2)]\} \\ & \wedge \{t_1 \wedge t_2 \rightarrow [(V = V_2) \wedge (V' = V_3)]\} \\ & \wedge (P(V_0) \vee P(V_1) \vee P(V_2) \vee P(V_3)),\end{aligned} \tag{4}$$

where $V = V'$ is a shorthand indicating every variable $v \in V$ equals its corresponding counterpart $v' \in V$. In Equation 4, if the value of t_1 is 0, then there is a continuous trajectory that evolves from the initial state q_0, where $q_0.\ell_0 = \ell oc_1$ and $x_0 = 0$, to the next state q_1, where $q_1.\ell_1 = \ell oc_1$ and $x_1 \leq 5$. When $t_1 = 1$ and $t_2 = 0$, the system takes the discrete transition from the current state q_1 to the next state q_2, where $q_2.\ell_2 = \ell oc_2$ and the value of x_2 is not higher than 10. At $k = 3$, both t_1, and t_2 are true, then q_2 becomes the current state, and q_3 is the next state, where $q_3.\ell_3 = \ell oc_1$, and $x_3 \leq 5$. The discrete transition taken from q_2 to q_3 when $x \geq 10$ will reset the value of x to 0.

If it terminates, an SMT solver supporting the combined theory of bitvectors and reals with quantifiers will return SAT for the QBMC formula if there exists an execution from an initial state to a bad state, i.e., if a bad state is reachable. Otherwise, if it terminates, it will return UNSAT if a bad state is not reachable in k steps. We note that the combination theory of linear real arithmetic with bitvectors is decidable, and Z3 is in essence a decision procedure for this theory. For instance, the example from Equation 4 has the state q, where $q.\ell = \ell oc_2$, and $x < 2.5$ is unreachable. Up to $k = 3$, if the system has any safety specification containing the state q, that means $q \models \bigvee_{i=0}^{3} P(q_i)$, and an SMT solver will return SAT on checking the formula $\Omega(3)$. Thus, we can assert that the system is unsafe for $k = 3$, which in this case, was just illustrating a reachability query.

5 Experimental Results

We implement the QBMC method described in Section 4 as a module within HyST [3]. HyST takes as input a hybrid automaton model in an extended form of the SpaceEx XML format [14] (supporting e.g., nonlinear functions instead of only affine ones). Then it will generate a Python script that includes the transition relations of a hybrid system expressed as quantified SMT formulas using the Z3 Python API. We evaluate the proposed QBMC method on several instances of Fischer and Lynch-Shavit mutual exclusion protocols. We compare the results to that of dReach, which is a state-of-the-art BMC tool for nonlinear hybrid automata [17], and with that of HyComp using the MathSAT SMT solver [9] in terms of execution time and memory consumption. We note that the comparison focuses only the BMC feature of dReach and HyComp, and all of the models of them were generated using HyST. The experiments

Table 1: The performance comparison between QBMC, HyComp and dReach in solving the BMC of Example 3.2. k is a number of steps in the BMC computation. "Time" and "Mem" are the time and memory usages measured in second and MB, respectively.

Tools	$k \leq 32$		$k \leq 64$		$k \leq 128$	
	Time	Mem	Time	Mem	Time	Mem
QBMC	0.81	17.3	3.22	27.7	5.36	41.2
dReach	82.6	106.7	1128.2	275.5	19896	812.6
HyComp	0.4	98.1	0.6	105.8	1.62	128.7

are performed on Intel I5 2.4GHz processor with 4GB RAM, executing the method described in this paper and dReach in a VirtualBox virtual machine running Ubuntu 64-bit. Z3 version 4.3.2 was used in the evaluation. We collect the execution times in second and the peak memory usages in MB for different examples. The preliminary implementation described in this paper, along with all the examples, is available online at: `https://github.com/LuanVietNguyen/QBMC`

We first evaluate our QBMC encoding on the illustrative hybrid automata presented in Example 3.2, and compare the results to those of dReach and HyComp. The performances of those three different methods are shown in Table 1. The constant values of the rectangular differential inclusion dynamics are given as: $a_1 = 0, b_1 = 1, a_2 = 0$, and $b_2 = 2$. The set of bad state used in this experiment is given in Equation 2, where all states in this set are unreachable and the system is always safe. For all of the values of k up to 128, the BMC results of QBMC, HyComp and dReach are all UNSAT, illustrating the correctness of the BMC procedure. The time and memory consumptions shown in Table 1 preliminarily indicate that our QBMC approach is capable of solving BMC faster than dReach, but slower than HyComp. However, our approach is more scalable as it requires significantly less memory usage compared to dReach and HyComp.

5.1 Fischer mutual exclusion protocol

Next, we evaluate QBMC with several scenarios using the Fischer mutual exclusion protocol [22]. Fischer mutual exclusion is a timed distributed algorithm that ensures a mutual exclusion safety property, namely that at most one process in a network of N processes may enter a critical section simultaneously. Figure 2 shows the model of Fischer protocol in including four discrete locations $\mathsf{Loc} \triangleq \{\mathsf{rem}, \mathsf{try}, \mathsf{wait}, \mathsf{cs}\}$, where Δ_1 and Δ_2 are two real timing parameters. Here, if $\Delta_1 < \Delta_2$ a mutual exclusion is guaranteed. The set of bad states is defined by:

$$\phi \triangleq \neg\forall i, j \in \{1, \ldots, N\} \mid (i \neq j \wedge q_i = cs) \implies q_j \neq cs,$$

where q_i and q_j are variables modeling the discrete location of the automata, cs is the critical section location. To evaluate QBMC, we perform the BMC for both of the safe and unsafe version of Fischer protocol. In the safe version, a state where the set of bad states ϕ is satisfied is not reachable, while in the

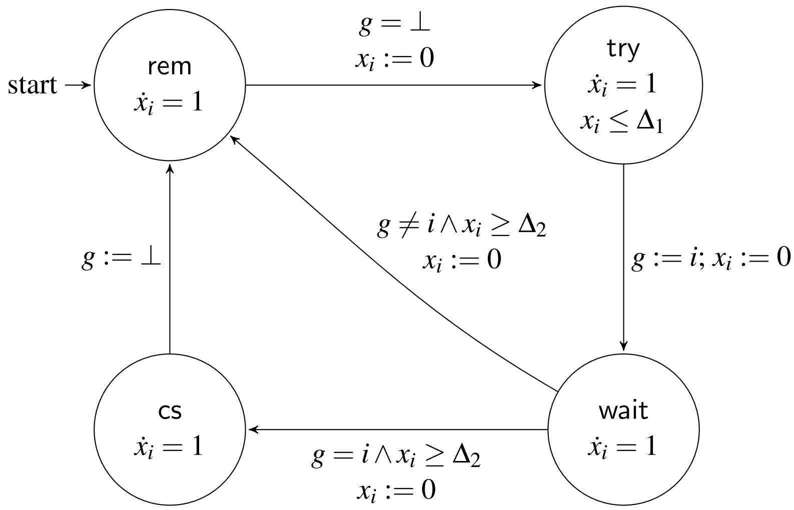

Figure 2: Fischer's mutual exclusion algorithm for a process with identifier $i \in \{1,\ldots,N\}$. Here, g is a global variable of type $\{\bot,1,\ldots,N\}$, x_i is a local variable of type $\mathbb{R}$, and both Δ_1 and Δ_2 are constants of type $\mathbb{R}$.

unsafe one, a state where ϕ is satisfied is reachable. We then also compare the performance of QBMC in solving the BMC of Fischer protocol with HyComp and dReach.

Table 2 shows the execution time and memory usage comparison among HyComp, dReach and QBMC for Fischer protocol with different numbers of processes, where NoL is the number of locations; FS, FU denote the safe and unsafe versions of Fischer protocol, respectively, and the number following the hyphen (-) describes a number of processes for each version. For instance, FS-2, FU-2 are the safe and unsafe versions of the Fischer protocol with 2 processes, respectively. We choose $\Delta_1 = 5$, $\Delta_2 = 70$ for a safe version, and $\Delta_1 = 75$, $\Delta_2 = 70$ for an unsafe one. In Table 2, T/O means the computation time out ($\geq$ 12 hours), M/O represents that the peak memory usage is higher than 4GB, and N/A denotes that the information of times or memory usages cannot be captured due to either T/O or M/O. Also, $\times$ means that the BMC procedure cannot terminate.

Compared to HyComp. According to Table 2, we can see that HyComp is generally faster than QBMC, but it requires a higher memory consumption than QBMC. For instance, up to $k = 8$, HyComp can address the BMC of the safe version of Fischer protocol with 5 processes in 172.4 seconds, which is approximately twice faster than QBMC (terminates after 344 seconds). However, HyComp consumes 1405.9 MB of memory, which is more than five times higher than QBMC (using only 254.4 MB). Additionally, with $k \leq 16$, the BMC of the unsafe version of Fischer protocol with 5 processes cannot terminate in HyComp due to out of memory (requiring more than 4GB). However, QBMC can solve it using less than 500 MB. Hence, we can deduce that QBMC is superior than HyComp with respect to memory consumption.

Compared to dReach. Generally, Table 2 also indicates that QBMC outperforms dReach in solving the BMC of Fischer protocol. As an example, with $k \leq 16$, dReach solves the BMC of the unsafe version

Table 2: The performance of the BMC of Fischer mutual exclusion protocol using QBMC, HyComp, and dReach.

Example	NoL	k (≤)	QBMC			HyComp			dReach		
			Time	Mem	Result	Time	Mem	Result	Time	Mem	Result
FU-2	4^2	8	1.1	24.7	SAT	0.4	100.9	SAT	48.4	28.9	SAT
		16	1.52	28.2	SAT	0.5	101.4	SAT	50.3	30.7	SAT
FS-2		8	1.6	25.2	UNSAT	0.5	101.4	UNSAT	64.1	120.8	UNSAT
		16	6.4	30	UNSAT	2.8	107.3	UNSAT	T/O	N/A	×
FU-3	4^3	8	6.9	48.7	UNSAT	2.1	131.8	UNSAT	270	214	UNSAT
		16	22.7	49.7	SAT	6.7	149.6	SAT	959.3	235.3	SAT
FS-3		8	8.3	48.7	UNSAT	2.2	131.8	UNSAT	T/O	N/A	×
		16	52.4	52.4	UNSAT	55.8	214.4	UNSAT	T/O	N/A	×
FU-4	4^4	8	40.1	73.2	UNSAT	13.3	318.2	UNSAT	T/O	N/A	×
		16	119.1	156.2	UNSAT	569.4	895.4	UNSAT	T/O	N/A	×
FS-4		8	76.1	74.1	UNSAT	9.9	319.1	UNSAT	T/O	N/A	×
		16	T/O	N/A	×	788	1010.4	UNSAT	T/O	N/A	×
FU-5	4^5	8	288.8	249.9	UNSAT	109.1	1345.4	UNSAT	T/O	N/A	×
		16	21456	473.8	UNSAT	N/A	M/O	×	T/O	N/A	×
FS-5		8	344.4	254.4	UNSAT	172.4	1405.9	UNSAT	T/O	N/A	×
		16	T/O	N/A	×	N/A	M/O	×	T/O	N/A	×

of Fischer protocol with 3 processes in 959.3 seconds using 235.3 MB of memory, while QBMC only terminates in 22.7 seconds ($\approx$ 40 time faster) using only 49.7 MB. Moreover, the BMC of both unsafe and safe versions of Fischer protocol with more than 3 processes is not able to terminate in dReach. Although dReach is capable to address the BMC of a wide range of nonlinear hybrid systems, handling scalability is not its strength.

Overall, QBMC has a competitive execution time and is more scalable to other state-of-the-art SMT solvers. Due the state-space (and formula) explosion of BMC, the reduction of memory consumption is one of the major challenges to address. Since QBMC requires a smaller amount of memory usage than other quantifier-free BMC approaches, it is effective in solving the BMC of large scale problems. According to Table 2, the BMC of the unsafe version of 4-processes Fischer protocol can be checked completely using QBMC up to $k = 32$ with only 254.1 MB of memory consumption. This result indicates that QBMC is effective for bug detection. However, as k increases, the higher execution time and the larger memory usage are required for the quantified encoding of BMC due to the increasing number of all possible paths from an initial state in the set of initial states to a bad state that does not satisfy the set of safety specifications.

5.2 Lynch-Shavit mutual exclusion protocol

The Lynch-Shavit protocol is a modified version of Fischer protocol where the mutual exclusion property is time-independent [27]. Intuitively, the protocol ensures that mutual exclusion is always satisfied even if the timing constraints are violated. We first modeled the hybrid automata of Lynch-Shavit protocol [1] in an extended form of the SpaceEx XML format. Then, we used the HyST tool to translate it to other model formats (e.g., for HyComp and dReach), and create the QBMC encoding. Each process of Lynch-Shavit protocol has 9 discrete locations, so the Lynch-Shavit protocol with 4 processes includes 6561 locations. The set of bad states of Lynch-Shavit protocol is defined similar to Fischer protocol, where two processes may be in the critical section.

The performance analyzing the BMC of Lynch-Shavit protocol using QBMC and HyComp are shown in Table 3, in which the BMC results of QBMC and HyComp are both UNSAT. Due to scalability problem, the BMC of Lynch-Shavit protocol using dReach could not terminate correctly. Thus, we do not demonstrate the time and memory consumptions for the BMC of Lynch-Shavit protocol using dReach in Table 3. However, we still provide the model files in our preliminary implementation package for interested readers to investigate. Again, we can see the trade-off between the two approaches. HyComp is generally faster than QBMC, but requires a much higher memory usage. As a result, the BMC of Lynch-Shavit protocol with 4 processes can be solved by QBMC up to $k = 16$ with consuming approximately 1GB memory, but cannot be solved in HyComp up to $k = 8$ due to out of memory. Hence, that demonstrates the proposed QBMC approach is efficient and scalable in addressing the BMC of large hybrid systems.

6 Conclusion and Future Works

In this paper, we present a new SMT-based technique that encodes, in a quantified form, the BMC problem for rectangular hybrid systems which also subsumes this encoding for timed systems. The preliminary results for the Fischer mutual exclusion protocol and Lynch-Shavit protocol indicate the capability of our method to solve the BMC problem for hybrid systems including more than thousand locations. We compare the QBMC approach to the quantifier-free BMC approaches in the dReach tool

Table 3: The performance comparison between QBMC and HyComp in solving the BMC of Lynch-Shavit mutual exclusion protocol.

Tools	NoL	$k \leq 4$		$k \leq 8$		$k \leq 16$	
		Time	Mem	Time	Mem	Time	Mem
QBMC	9^2	3.7	52.2	5.1	52.3	25.9	52.7
	9^3	15.5	65.6	31.3	87.5	1091.5	144.5
	9^4	256.1	702.8	1062.1	708.9	43178	1196.2
HyComp	9^2	0.8	121.9	1.33	132.8	9.5	170.5
	9^3	2.7	307.9	12.81	380.8	192.8	771.4
	9^4	63.9	2655.4	N/A	M/O	N/A	M/O

that uses the dReal SMT solver, and the HyComp tool built on top of nuXmv that uses the MathSAT SMT solver. The experimental results demonstrate that our approach is competitive to these tools in terms of execution time and memory consumption.

Future works. As solvers for fragments of many-sorted first-order logic such as LRA, NRA, etc., continue to improve, QBMC encodings such as the one described in this paper will become more effective, similar to how QBMC for discrete systems has been shown to be effective with QBF encodings [28]. In future, we plan to conduct additional experiments for other benchmarks to cover a wider range of applications, and compare the results to more tools and techniques, such as Uppaal and tools built on top of CVC4 [5]. We intend to compare the proposed QBMC with IC3-based algorithms that might provide trade-offs between memory consumption and execution time. We will also investigate more general classes of hybrid automata whose dynamics are expressed by linear or polynomial differential equations.

Acknowledgements

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant numbers 1918450 and 2028001. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of NSF. This paper is based on an earlier workshop paper that did not appear in an archival proceedings: Luan Viet Nguyen, Djordje Maksimovic, Taylor T. Johnson, Andreas Veneris, ”Quantified Bounded Model Checking for Rectangular Hybrid Automata,” which was presented at the 9th International Workshop on Constraints in Formal Verification (CFV 2015), Austin, Texas, November 2015, colocated with the 34th IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2015).

References

[1] Parosh Aziz Abdulla, Aurore Collomb-Annichini, Ahmed Bouajjani & Bengt Jonsson (2004): *Using forward reachability analysis for verification of lossy channel systems*. Formal Methods in System Design 25(1), pp. 39–65, doi:10.1023/$B:FORM$.0000033962.51898.1a.

[2] Kyungmin Bae & Sicun Gao (2017): *Modular smt-based analysis of nonlinear hybrid systems*. In: *2017 Formal Methods in Computer Aided Design (FMCAD)*, IEEE, pp. 180–187, doi:10.23919/$FMCAD$.2017.8102258.

[3] Stanley Bak, Sergiy Bogomolov & Taylor T. Johnson (2015): *HyST: A Source Transformation and Translation Tool for Hybrid Automaton Models*. In: *Proc. of the 18th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*, ACM, doi:10.1145/2728606.2728630.

[4] Clark Barrett, Pascal Fontaine & Cesare Tinelli (2017): *The SMT-LIB Standard: Version 2.6*. Technical Report, Department of Computer Science, The University of Iowa. Available at `www.SMT-LIB.org`.

[5] Clark W. Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovic, Tim King, Andrew Reynolds & Cesare Tinelli (2011): *CVC4*. In Ganesh Gopalakrishnan & Shaz Qadeer, editors: *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, *Lecture Notes in Computer Science* 6806, Springer, pp. 171–177, doi:10.1007/978 − 3 − 642 − 22110 − 1_14.

[6] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson & Wang Yi (1996): *UPPAAL: A tool suite for automatic verification of real-time systems*. In Rajeev Alur, Thomas Henzinger & Eduardo Sontag, editors: *Hybrid Systems III*, *LNCS* 1066, Springer, pp. 232–243, doi:10.1007/BFb0020949.

[7] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri & Stefano Tonetta (2014): *The nuXmv Symbolic Model Checker*. In Armin Biere & Roderick Bloem, editors: *Computer Aided Verification*, *Lecture Notes in Computer Science* 8559, Springer International Publishing, pp. 334–342, doi:10.1007/978 − 3 − 319 − 08867 − 9_22.

[8] Xin Chen, Erika Abraham & Sriram Sankaranarayanan (2013): *Flow*: An Analyzer for Non-linear Hybrid Systems*. In Natasha Sharygina & Helmut Veith, editors: *Computer Aided Verification*, *Lecture Notes in Computer Science* 8044, Springer Berlin Heidelberg, pp. 258–263, doi:10.1007/978 − 3 − 642 − 39799 − 8_18.

[9] Alessandro Cimatti, Alberto Griggio, Sergio Mover & Stefano Tonetta (2015): *HyComp: An SMT-Based Model Checker for Hybrid Systems*. In Christel Baier & Cesare Tinelli, editors: *Tools and Algorithms for the Construction and Analysis of Systems*, *Lecture Notes in Computer Science* 9035, Springer Berlin Heidelberg, pp. 52–67, doi:10.1007/978 − 3 − 662 − 46681 − 0_4.

[10] Alessandro Cimatti, Alberto Griggio, Sergio Mover & Stefano Tonetta (2015): *HyComp: An SMT-Based Model Checker for Hybrid Systems*. In: *Tools and Algorithms for the Construction and Analysis of Systems*, Springer, pp. 52–67, doi:10.1007/978 − 3 − 662 − 46681 − 0_4.

[11] Alessandro Cimatti, Sergio Mover & Stefano Tonetta (2013): *SMT-based scenario verification for hybrid systems*. *Formal Methods in System Design* 42(1), pp. 46–66, doi:10.1007/s10703 − 012 − 0158 − 0.

[12] Leonardo De Moura & Nikolaj Bjørner (2008): *Z3: An Efficient SMT Solver*. In: *Proc. of 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS '08/ETAPS '08, Springer-Verlag, pp. 337–340, doi:10.1007/978 − 3 − 540 − 78800 − 3_24.

[13] Andreas Eggers, Martin Fränzle & Christian Herde (2008): *SAT Modulo ODE: A Direct SAT Approach to Hybrid Systems*. In Sungdeok Cha, Jin-Young Choi, Moonzoo Kim, Insup Lee & Mahesh Viswanathan, editors: *Automated Technology for Verification and Analysis*, *Lecture Notes in Computer Science* 5311, Springer Berlin / Heidelberg, pp. 171–185, doi:10.1007/978 − 3 − 540 − 88387 − 6_14.

[14] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang & Oded Maler (2011): *SpaceEx: Scalable Verification of Hybrid Systems*. In: *Computer Aided Verification (CAV)*, LNCS, Springer, doi:10.1007/978 − 3 − 642 − 22110 − 1_30.

[15] Sicun Gao, J. Avigad & E.M. Clarke (2012): *Delta-Decidability over the Reals*. In: *Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on*, pp. 305–314, doi:10.1109/LICS.2012.41.

[16] Sicun Gao, Soonho Kong & Edmund Clarke (2013): *Satisfiability Modulo ODEs*. In: *International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, doi:10.1109/FMCAD.2008.ECP.14.

[17] Sicun Gao, Soonho Kong & Edmund M Clarke (2013): *dReal: An SMT solver for nonlinear theories over the reals*. In: *Automated Deduction–CADE-24*, Springer, pp. 208–214, doi:10.1007/978-3-642-38574-2_14.

[18] Silvio Ghilardi & Silvio Ranise (2010): *MCMT: A Model Checker Modulo Theories.* In: *IJCAR*, 10, Springer, pp. 22–29, doi:10.1007/978-3-642-14203-1_3.

[19] T. A. Henzinger (1996): *The theory of hybrid automata*. In: *IEEE Symposium on Logic in Computer Science (LICS)*, IEEE Computer Society, Washington, DC, USA, p. 278, doi:10.1109/LICS.1996.561342.

[20] Thomas A. Henzinger, Pei-Hsin Ho & Howard Wong-Toi (1997): *HyTech: A model checker for hybrid systems*. *Journal on Software Tools for Technology Transfer* 1, pp. 110–122, doi:10.1007/s100090050008.

[21] Mikoláš Janota, William Klieber, Joao Marques-Silva & Edmund Clarke (2012): *Solving QBF with counterexample guided refinement*. In: *Theory and Applications of Satisfiability Testing–SAT 2012*, Springer, pp. 114–128, doi:10.1007/978-3-642-31612-8_10.

[22] Taylor T. Johnson & Sayan Mitra (2012): *A Small Model Theorem for Rectangular Hybrid Automata Networks*. In: *Proceedings of the IFIP International Conference on Formal Techniques for Distributed Systems, Joint 14th Formal Methods for Open Object-Based Distributed Systems and 32nd Formal Techniques for Networked and Distributed Systems (FMOODS-FORTE)*, *LNCS* 7273, Springer, doi:10.1007/978-3-642-30793-5_2.

[23] Taylor T. Johnson & Sayan Mitra (2013): *Invariant Synthesis for Verification of Parameterized Cyber-Physical Systems with Applications to Aerospace Systems*. In: *Proceedings of the AIAA Infotech at Aerospace Conference (AIAA Infotech 2013)*, Boston, MA, doi:10.2514/6.2013-4811.

[24] Taylor T. Johnson & Sayan Mitra (2014): *Anonymized Reachability of Rectangular Hybrid Automata Networks*. In: *Formal Modeling and Analysis of Timed Systems (FORMATS)*, doi:10.1007/978-3-319-10512-3_10.

[25] Toni Jussila & Armin Biere (2007): *Compressing BMC encodings with QBF*. *Electronic Notes in Theoretical Computer Science* 174(3), pp. 45–56, doi:10.1016/j.entcs.2006.12.022.

[26] Soonho Kong, Sicun Gao, Wei Chen & Edmund Clarke (2015): *dReach: Delta-reachability analysis for hybrid systems*. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, pp. 200–205, doi:10.1007/978-3-662-46681-0_15.

[27] Nancy Lynch & Nir Shavit (1992): *Timing-based mutual exclusion*. In: *Real-Time Systems Symposium, 1992*, IEEE, pp. 2–11, doi:10.1109/REAL.1992.242681.

[28] H. Mangassarian, A. Veneris & M. Benedetti (2010): *Robust QBF Encodings for Sequential Circuits with Applications to Verification, Debug, and Test*. *Computers, IEEE Transactions on* 59(7), pp. 981–994, doi:10.1109/TC.2010.74.

[29] C. Miller, K. Gitina & B. Becker (2011): *Bounded Model Checking of Incomplete Real-time Systems Using Quantified SMT Formulas*. In: *Microprocessor Test and Verification (MTV), 2011 12th International Workshop on*, pp. 22–27, doi:10.1109/MTV.2011.13.

[30] Fedor Shmarov, Nicola Paoletti, Ezio Bartocci, Shan Lin, Scott A Smolka & Paolo Zuliani (2017): *SMT-based synthesis of safe and robust PID controllers for stochastic hybrid systems*. In: *Haifa Verification Conference*, Springer, pp. 131–146, doi:10.1007/978-3-319-70389-3_9.

[31] Ashish Tiwari (2012): *HybridSAL Relational Abstracter*. In P. Madhusudan & SanjitA. Seshia, editors: *Computer Aided Verification*, *Lecture Notes in Computer Science* 7358, Springer Berlin Heidelberg, pp. 725–731, doi:10.1007/978-3-642-31424-7_56.