Modular SMT Encoding for Networks of Nonlinear Hybrid Systems

Kyungmin Bae[1], Soonho Kong[2], and Sicun Gao[3]

[1] SRI International
[2] Carnegie Mellon University
[3] Massachusetts Institute of Technology

Abstract. Formal analysis problems of general hybrid systems, which involve nonlinear ordinary differential equations (ODEs), can be reduced to SMT problems over the real numbers. However, the standard SMT encoding experiences a new *formula explosion problem* for networks of nonlinear hybrid systems. Since continuous dynamics of tightly coupled components cannot be decomposed, one cannot *modularly* encode such systems using SMT formulas over the real numbers. In order to avoid this problem, we present a novel logical framework to symbolically decompose ODE systems, which extends the standard theory over the real number by adding parameterized integration operators. The experimental results show that our techniques greatly increase the performance of SMT-based analysis for networks of hybrid systems with nonlinear ODEs.

1 Introduction

Formal analysis problems of hybrid systems can be encoded as SMT formulas over the real numbers. In this approach, we can combine state-of-the-art SMT techniques with numerical algorithms for analyzing the continuous dynamics, governed by ordinary differential equations (ODEs). The satisfaction problems of these formulas are in general undecidable for nonlinear hybrid systems. But if we take into account robustness properties under numerical perturbations, then they become *decidable* up to any user-given precision $\delta > 0$ using δ-complete decision procedures [10,12]. This is very useful in practice, since sampling exact values of physical parameters is not possible in reality.

However, for *networks of nonlinear hybrid systems*, their SMT-based analysis is often unfeasible in practice. One of the reasons is that continuous connections between different components can be *precisely* modeled using *coupled* systems of ODEs. Consider, for example, three adjacent rooms: the temperature of one room continuously affects the temperature of the other rooms. The ODE systems for one room, modeling its behavior according to its control modes, also include the variables of the other rooms, and combinations of different ODE components give the behavior of the entire system. Unlike the case of networks of restricted hybrid systems (e.g., linear hybrid systems), a typical way to modularly encode parallel compositions cannot be applied for such *tightly coupled* nonlinear components, because *connected* variables evolve simultaneously over time.

For this reason, existing SMT approaches can make use of only non-modular encodings for networks of nonlinear hybrid automata, by explicitly enumerating all combinations of ODE components. For a parallel composition $H_1 \parallel \cdots \parallel H_n$ of hybrid systems, when each component H_i includes m_i ODE systems with respect to its control modes, by using the non-modular way to encode hybrid systems in SMT, the size of the formula is $O(\prod_{i=1}^{n} m_i)$. This leads to the *formula explosion problem* that makes such SMT-based analysis practically infeasible. The goal of this paper is to provide a logical framework for SMT to effectively encode networks of nontrivial hybrid systems, involving nonlinear ODEs.

Our Contributions. We explain a new logical formulation to modularly encode networks of nonlinear hybrid systems as SMT formulas. We show that coupled systems of ODEs can be decomposed by means of universal quantification over time, by expressing continuous physical connections between components. The size of the SMT formula by the new modular encoding is $O(\sum_{i=1}^{n} m_i^2)$.

We then presents a novel logical framework for networks of nonlinear hybrid automata that provides a simple decision procedure. It extends the standard theory over the real number by adding parameterized integration operators, to logically decompose coupled ODE systems without using *universal quantification*. The satisfaction problems in the new logical theory can be reduced to ones in the standard theory of the real numbers *at no cost*.

We have implemented our technique within the dReal SMT solver [11], and performed experiments on a number of nontrivial networked nonlinear hybrid systems. The experimental results show that our techniques can dramatically increase the performance of SMT-based analysis for these networks of hybrid systems that involve multiple control modes and nonlinear ODEs. To the best of our knowledge, such networks of nonlinear hybrid systems cannot be effectively analyzed by other existing SMT-based approaches for hybrid systems.

Related Work. SMT-based verification is a fairly new direction for nonlinear hybrid systems, because of the difficulty of handling SMT formulas over the real numbers with nonlinear functions. The research direction is initiated in [18], which uses constraint solving algorithms for handling nonlinear reachability problems. Two main lines of work that explicitly formulate analysis problems as SMT formulas are based on the HySAT/iSAT solver [8,6] and the MathSAT solver [5,4]. But efficient encoding of networks of nonlinear hybrid systems has not been investigated in existing work along these lines. Our techniques are orthogonal to reachable set computation techniques such as SpaceEX [9] and Flow* [3], since they can also be used as ODE solvers for SMT solving.

The dReal SMT solver [11] can check the satisfiability of SMT formulas with nonlinear real functions and ODEs up to any precision $\delta > 0$. The dReach tool [14] encodes reachability problems of nonlinear hybrid systems as SMT problems for dReal. In order to deal with the formula explosion problem, dReach *explicitly* enumerates all mode paths of a hybrid automaton by generating many small SMT formulas for these paths. But the power of SMT techniques (such as DPLL($\mathcal{T}$) and CDCL) cannot be fully exploited in this way.

The rest of the paper is organized as follows. Section 2 gives some background on hybrid automata and SMT. Section 3 discusses modular encodings of networks of hybrid automata. Section 4 presents a new logical framework supporting the modular encodings of networks of *nonlinear* hybrid automata. Section 5 gives an overview of cast studies. Finally, Section 6 presents some conclusions.

2 Preliminaries on Hybrid Automata and SMT

Hybrid Automata. Hybrid systems are formally specified by *hybrid automata* [13,16]. Discrete states are given by a set of modes Q. Physical states are given by a finite set of real-valued variables $X = \{x_1, \ldots, x_l\}$, where the domain of its values $\boldsymbol{v} = (v_1, \ldots, v_l)$ is denoted by $val(X) = \mathbb{R}^l$. For each mode $q \in Q$, its invariant condition defines all possible values of variables X, and its flow condition defines the continuous dynamics of X. Discrete transitions between modes are given by jump conditions, identified with actions $a \in \Sigma$. Let $traj(X)$ denote a set of *trajectories* that describe the values of X over time.

Definition 1. *A* hybrid automaton *is $H = (X, Q, \Sigma, init, inv, flow, jump)$ with: (i) X a set of real-valued variables; (ii) Q a set of modes, (iii) Σ a set of actions; (iv) $init \subseteq Q \times val(X)$ a set of initial states; (v) $inv : Q \to 2^{val(X)}$ defining the set of all possible values of X in mode q; (vi) $flow : Q \to 2^{traj(X)}$ defining the set of all trajectories of X in mode q; and (vii) $jump : (Q \times val(X)) \times \Sigma \times (Q \times val(X))$ defining discrete transitions $(q, \boldsymbol{v}) \xrightarrow{a} (q', \boldsymbol{v'})$ with action a.*

The *init*, *inv*, and *jump* conditions are written as predicates over variables X: for example, $init_q(\boldsymbol{x})$, $inv_q(\boldsymbol{x})$, and $jump_{q,a,q'}(\boldsymbol{x}, \boldsymbol{x'})$ for modes $q, q' \in Q$, action $a \in \Sigma$, and vectors of variables $\boldsymbol{x}$ and $\boldsymbol{x'}$. The *flow* condition is written as a system of ordinary differential equations (ODEs) of the form $\dot{\boldsymbol{x}} = flow_q(\boldsymbol{x})$.

Networks of hybrid systems can be formally specified by parallel compositions of hybrid automata. For a parallel composition $H_1 \parallel H_2$, its mode is a pair (q_1, q_2) of H_1's mode q_1 and H_2's mode q_2, and its physical state is given by the set of real-valued variables $X_1 \cup X_2$ from H_1 and H_2. The continuous *physical* interaction between H_1 and H_2 is modeled by using shared variables in $X_1 \cap X_2$, and their discrete communication is modeled by using joint synchronous actions in $\Sigma_1 \cap \Sigma_2$. For a physical state $\boldsymbol{v} \in val(X_1 \cup X_2)$, let $\boldsymbol{v} \restriction_{X_i} \in val(X_i)$ denote the projection of $\boldsymbol{v}$ onto variables X_i for $i = 1, 2$. Similarly, for a trajectory $\boldsymbol{f} \in traj(X_1 \cup X_2)$, let $\boldsymbol{f} \restriction_{X_i} \in traj(X_i)$ denote the restriction of $\boldsymbol{f}$ by X_i.

Definition 2. *The* parallel composition *of hybrid automata H_1 and H_2 is the automaton $H_1 \parallel H_2 = (X_1 \cup X_2, Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, init, inv, flow, jump)$, where:*

- $((q_1, q_2), \boldsymbol{v}) \in init$ *iff* $(q_1, \boldsymbol{v} \restriction_{X_1}) \in init_1$ *and* $(q_2, \boldsymbol{v} \restriction_{X_2}) \in init_2$;
- $\boldsymbol{v} \in inv(q_1, q_2)$ *iff* $\boldsymbol{v} \restriction_{X_1} \in inv_1(q_1)$ *and* $\boldsymbol{v} \restriction_{X_2} \in inv_2(q_2)$;
- $\boldsymbol{f} \in flow(q_1, q_2)$ *iff* $\boldsymbol{f} \restriction_{X_1} \in flow_1(q_1)$ *and* $\boldsymbol{f} \restriction_{X_2} \in flow_2(q_2)$; *and*
- $((q_1, q_2), \boldsymbol{v}) \xrightarrow{a} ((q_1', q_2'), \boldsymbol{v'}) \in jump$ *iff for each* $i = 1, 2$*: (i) if* $a \in \Sigma_i$*, then* $(q_i, \boldsymbol{v} \restriction_{X_i}) \xrightarrow{a} (q_i', \boldsymbol{v'} \restriction_{X_i}) \in jump_i$*, and (ii) if* $a \notin \Sigma_i$*, then no transition of* H_i *is taken (that is,* $(q_i, \boldsymbol{v} \restriction_{X_i}) = (q_i', \boldsymbol{v'} \restriction_{X_i})$*).*

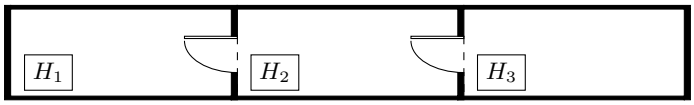

Fig. 1. Three interconnected rooms by open doors.

Example 1 (Physically Networked Thermostat Controllers). Consider a network of classical thermostat hybrid automata (the specification of a single hybrid automaton is adapted from [13]). A number of rooms are interconnected by open doors, and the temperature of each room is separately controlled by each thermostat that turns its heater on and off, as depicted in Figure 1.

The temperature x_i of room i depends on the heater's mode $q_i \in \{m_{\texttt{on}}, m_{\texttt{off}}\}$ and the temperatures of the adjacent rooms. The value of x_i changes according to the ODEs, where $A_i \subseteq \mathbb{N}$ denotes the set of its adjacent rooms:

$$\dot{x_i} = \begin{cases} K_i\big(h_i - (x_i \cdot (1 - \sum_{j \in A_i} k_{i,j}) + \sum_{j \in A_i} x_j k_{i,j})\big) & \text{if } q_i = m_{\texttt{on}}, \\ -K_i\big(x_i \cdot (1 - \sum_{j \in A_i} k_{i,j}) + \sum_{j \in A_i} x_j k_{i,j}\big) & \text{if } q_i = m_{\texttt{off}}. \end{cases}$$

The constants $K_i, h_i, k_{i,j} \in \mathbb{R}$ depend on the size of room i, the heater's power, and the size of the open door between rooms i and j, respectively. Notice that the ODEs for room i include variables x_j for adjacent rooms $j \in A_i$.

Every thermostat controller synchronously performs its discrete transitions with common action a. For each second, the heater is turned on if $x_i \leq T^i_{\min}$, and turned off if $x_i > T^i_{\max}$. To keep track of each one-second period, every automaton has a *shared* timer variable τ with the flow condition $\dot{\tau} = 1$.

The hybrid automaton $H_i = (X_i, Q_i, \Sigma_i, \mathit{init}_i, \mathit{inv}_i, \mathit{flow}_i, \mathit{jump}_i)$ is defined in an expected way. For example: (i) $X_1 = \{x_1, x_2, \tau\}$, (ii) $Q_1 = \{q_{\texttt{on}}, q_{\texttt{off}}\}$, (iii) $\Sigma_1 = \{a\}$, (iv) $\mathit{init}_1 = \{(q_{\texttt{off}}, v_1, v_2, \tau) \mid T^i_{\min} < v_1, v_2 < T^i_{\max}, \tau = 0\}$, (v) $\mathit{inv}_1(q) = \{(v_1, v_2, \tau) \mid 0 \leq \tau \leq 1\}$, (vi) $\mathit{flow}_1(q)$ given by the ODEs for $\dot{x_1}$ and $\dot{\tau}$, (vii) $(q, v_1, v_2, 1) \xrightarrow{a} (q', v_1, v_2, 0) \in \mathit{jump}$ iff $(x_i \leq T^i_{\min} \to q' = q_{\texttt{on}})$, $(x_i > T^i_{\max} \to q' = q_{\texttt{off}})$, and $(T^i_{\min} < x_i \leq T^i_{\max} \to q' = q)$. ■

Standard Encoding of Hybrid Systems in SMT. An SMT (satisfiability modulo theories) problem is to check the satisfiability of first-order formulas with respect to certain decidable logical theories. Formal analysis of hybrid systems can be reduced to the satisfiability of SMT formulas over the real numbers, along with computable real functions, including polynomials, trigonometric functions, and solutions of Lipschitz-continuous ODEs. The satisfiability of such formulas is generally undecidable for nonlinear real functions, but the satisfiability *up to a given precision* $\delta > 0$ is decidable for nonlinear real functions using δ-complete decision procedures [10,12], implemented in the dReal SMT solver [11].

Definition 3. *Given a finite set $\mathcal{F}$ of computable real functions (real constants given by 0-ary functions), $\mathcal{L}_{\mathcal{F}} = (\mathcal{F}, >)$ denotes the first-order signature over the real numbers with the real functions in $\mathcal{F}$, and $\mathbb{R}_{\mathcal{F}} = (\mathbb{R}, \mathcal{F}^{\mathbb{R}}, >^{\mathbb{R}})$ is the standard structure for the theory of the real numbers.*

The behavior of a hybrid automaton $H = (X, Q, \Sigma, \mathit{init}, \mathit{inv}, \mathit{flow}, \mathit{jump})$ can be encoded as $\mathcal{L}_{\mathcal{F}}$-formulas. As mentioned earlier, discrete jumps $(q, \boldsymbol{v}) \xrightarrow{a} (q', \boldsymbol{v'})$ are expressed as $\mathcal{L}_{\mathcal{F}}$-predicates $\mathit{jump}_{q,a,q'}(\boldsymbol{v}, \boldsymbol{v'})$. The continuous change of X's values in mode q from $\boldsymbol{v^0}$ to $\boldsymbol{v^t}$ for duration t according to the ODE system $\dot{\boldsymbol{x}} = \mathit{flow}_q(\boldsymbol{x})$ is expressed as the $\mathcal{L}_{\mathcal{F}}$-formula

$$\mathit{flow}_q(\boldsymbol{v^0}, \boldsymbol{v^t}, t) \;\equiv\; \Big(\boldsymbol{v^t} = \boldsymbol{v^0} + \int_0^t \mathit{flow}_q(\boldsymbol{x})\,\mathrm{d}t\Big)$$

where the integral $\int_0^t \mathit{flow}_q(\boldsymbol{x})\,\mathrm{d}t$ is considered as a real function over time t. To state that the invariant condition holds for any continuous trajectories of X from $\boldsymbol{v^0}$ for duration t in mode q, we write the $\mathcal{L}_{\mathcal{F}}$-formula:[4]

$$\mathit{inv}_q(\boldsymbol{v^0}, t) \;\equiv\; \forall u \in [0,t], \forall \boldsymbol{x^u}.\ \mathit{flow}_q(\boldsymbol{v^0}, \boldsymbol{x^u}, u) \;\rightarrow\; \mathit{inv}_q(\boldsymbol{x^u})$$

Example 2 (Isolated Single Thermostat). Consider a hybrid automaton for one room in Example 1, but all its doors are closed, and thus the ODEs only depend on the room's temperature x. The flow condition is expressed as the $\mathcal{L}_{\mathcal{F}}$-formulas:

$$\begin{aligned}
\mathit{flow}_{\mathtt{on}}([x^0,\tau^0],[x^t,\tau^t],t) &\equiv \big([x^t,\tau^t] = [x^0,\tau^0] + \textstyle\int_0^t [K(h-x(t)),1]\,\mathrm{d}t\big)\\
\mathit{flow}_{\mathtt{off}}([x^0,\tau^0],[x^t,\tau^t],t) &\equiv \big([x^t,\tau^t] = [x^0,\tau^0] + \textstyle\int_0^t [-Kx(t),1]\,\mathrm{d}t\big)
\end{aligned}$$

For the jump condition, $\mathit{jump}_{q,a,q'}([x,\tau],[x',\tau']) \;\equiv\; (x \le T_{\min} \rightarrow q' = q_{\mathtt{on}}) \wedge (x > T_{\max} \rightarrow q' = q_{\mathtt{off}}) \wedge (T_{\min} < x \le T_{\max} \rightarrow q' = q) \wedge (\tau = 1 \wedge \tau' = 0)$. ■

The bounded reachability problem of a hybrid automaton H up to its k-th step can be encoded as the following $\mathcal{L}_{\mathcal{F}}$-formula whose satisfiable assignments give counterexamples of length k, where $t_{\max} > 0$ is a maximal duration of single modes, and $\mathit{safe}(\boldsymbol{x})$ is a predicate to denote a safe region of X:

$$\begin{aligned}
&\exists m_0,\ldots,m_k,\ \boldsymbol{x_0^0},\ldots,\boldsymbol{x_k^0},\boldsymbol{x_0^t},\ldots,\boldsymbol{x_k^t},\ \exists t_0,\ldots,t_k \in [0,t_{\max}].\\
&\bigvee_{q\in Q}\big[m_0 = q \;\wedge\; \mathit{init}_q(\boldsymbol{x_0^0}) \;\wedge\; \mathit{flow}_q(\boldsymbol{x_0^0},\boldsymbol{x_0^t},t_0) \;\wedge\; \mathit{inv}_q(\boldsymbol{x_0^0},t_0)\big]\\
\wedge \bigwedge_{i=1}^{k} \bigvee_{\substack{q,r\in Q\\ a\in\Sigma}} &\left[\begin{array}{r} m_{i-1} = q \;\wedge\; m_i = r \;\wedge\; \mathit{jump}_{q,a,r}(\boldsymbol{x_{i-1}^t},\boldsymbol{x_i^0})\\ \wedge\, \mathit{flow}_r(\boldsymbol{x_i^0},\boldsymbol{x_i^t},t_i) \;\wedge\; \mathit{inv}_r(\boldsymbol{x_i^0},t_i)\end{array}\right] \;\wedge\; \neg \mathit{safe}(\boldsymbol{x_k^t})
\end{aligned} \tag{1}$$

Each i-th step is in mode m_i, has duration t_i, begins with $\boldsymbol{x_i^0}$, and ends with $\boldsymbol{x_i^t}$. In the initial step, for *some* mode q, the *init* condition holds on $\boldsymbol{x_0^0}$, X's values changes from $\boldsymbol{x_0^0}$ to $\boldsymbol{x_0^t}$ during t_0 according to the *flow* condition, and the *inv* condition holds in the meantime. The i-th intermediate step is reached iff for *some* modes $q, r \in Q$, a discrete jump $q \xrightarrow{a} r$ happens from the final values $\boldsymbol{x_{i-1}^t}$ of the $(i-1)$-th step to the starting values $\boldsymbol{x_i^0}$, and then X's values changes from $\boldsymbol{x_i^0}$ to $\boldsymbol{x_i^t}$ during t_i according to the *flow* and *inv* conditions of mode r.

[4] dReal can handle such universally quantified formulas over the time variable u.

3 $\mathcal{L}_{\mathcal{F}}$-Encoding for Networks of Hybrid Automata

In principle, a network of hybrid automata can always be encoded as $\mathcal{L}_{\mathcal{F}}$-formulas using Formula (1) by explicitly using their parallel composition. However, the size of the resulting formula can be huge. For a network of hybrid automata $H_1, \ldots, H_n$, if each automaton H_i has m_i modes, then a parallel composition $H_1 \parallel \ldots \parallel H_n$ has $\prod_{i=1}^{n} m_i$ modes. For k-step bounded reachability, the size of the formula by Encoding (1) is $O(k \cdot \prod_{i=1}^{n} m_i)$.

To avoid this *formula explosion problem*, we need to encode networks of hybrid automata in a modular way. For networks of *nonlinear* hybrid automata, a main difficulty comes from the physical connections between *tightly coupled* components. The standard decomposition technique, that has been widely used for *restricted* hybrid automata, does not work for physically coupled nonlinear hybrid automata. We present a new $\mathcal{L}_{\mathcal{F}}$-formulation to modularly encode such networks of nonlinear hybrid automata in a logically correct way.

Standard Decomposition. A standard way to modularly encode a parallel composition $H_1 \parallel \cdots \parallel H_n$ is to take into account synchronization conditions. For each H_j, we extend the meaning of the jump predicate $jump^j$ for general actions $\Sigma_1 \cup \cdots \cup \Sigma_n$ according to Definition 2: $jump^j_{q_j,a,q'_j}(\boldsymbol{x_j}, \boldsymbol{x'_j}) = jump^j_{q_j,a,q'_j}(\boldsymbol{x_j}, \boldsymbol{x'_j})$ if $a \in \Sigma_j$, and $q_j = q'_j \ \wedge \ \boldsymbol{x'_j} = \boldsymbol{x_j}$ if $a \notin \Sigma_j$.

Assuming that *the flow condition of each automaton can be fully encoded as $\mathcal{L}_{\mathcal{F}}$-formulas*, a k-step bounded reachability problem of $H_1 \parallel \cdots \parallel H_n$ can then be encoded as the following $\mathcal{L}_{\mathcal{F}}$-formula of size $O(k \cdot \sum_{j=1}^{n} m_j^2)$:

$$
\begin{aligned}
&\exists \{m_0^j, \ldots, m_k^j,\ \boldsymbol{x_{j0}}, \ldots, \boldsymbol{x_{jk}}, \boldsymbol{x_{j0}^t}, \ldots, \boldsymbol{x_{jk}^t}\}_{j=1}^{n},\ \exists t_0, \ldots, t_k \in [0, t_{\max}]. \\
&\bigwedge_{j=1}^{n} \bigvee_{q_j \in Q_j} \Big[m_0^j = q_j \ \wedge \ init^j_{q_j}(\boldsymbol{x_{j0}}) \ \wedge \ flow^j_{q_j}(\boldsymbol{x_{j0}}, \boldsymbol{x_{j0}^t}, t_0) \ \wedge \ inv^j_{q_j}(\boldsymbol{x_{j0}}, t_0) \Big] \\
&\quad \wedge \bigwedge_{j=1}^{n} \bigwedge_{i=1}^{k} \bigvee_{\substack{q_j, r_j \in Q \\ a \in \bigcup_{l=1}^{n} \Sigma_l}} \left[\begin{array}{r} (m_{i-1}^j, m_i^j) = (q_j, r_j) \ \wedge \ jump^j_{q_j,a,r_j}(\boldsymbol{x_{j,i-1}^t}, \boldsymbol{x_{ji}}) \\ \wedge \ flow^j_{r_j}(\boldsymbol{x_{ji}}, \boldsymbol{x_{ji}^t}, t_i) \ \wedge \ inv^j_{r_j}(\boldsymbol{x_{ji}}, t_i) \end{array} \right] \qquad (2) \\
&\qquad\qquad \wedge \neg \bigwedge_{j=1}^{n} safe(\boldsymbol{x_{jk}^t})
\end{aligned}
$$

The behavior of each hybrid automaton is separately encoded using Formula (1). Whenever a discrete jump is taken by one hybrid automaton, all components synchronize on it using synchronization conditions. Flow conditions are assumed to be synchronized upon mode changes using mode variables.

The standard decomposition technique can be applied to various classes of restricted hybrid automata, such as linear hybrid automata and piecewise affine hybrid automata. The flow conditions of a single hybrid automaton in such classes are independently expressed by real functions of specified "solved" forms, and therefore they can be fully encoded as $\mathcal{L}_{\mathcal{F}}$-formulas.

But what about a very generic class of hybrid automata involving nonlinear ordinary differential equations (ODEs)? In this cases, for a network of hybrid automata, the ODEs for one hybrid automaton H_i can be physically coupled with the ODEs for other automata; that is, the derivatives of some ODE variables are *not* specified in H_i, but in other hybrid automata. Therefore, the standard decomposition technique cannot be directly applied, since the ODEs for a single hybrid automaton cannot be (even numerically) evaluated.

Example 3. Consider the network of three thermostat controllers in Example 1. The hybrid automaton H_1 for room 1, adjacent to rooms 2, has the variable set $X_1 = \{x_1, x_2, \tau\}$, and involves the ODEs $\dot{x_1} = 7 - (0.9x_1 + 0.1x_2)$ for $q_1 = m_{\texttt{on}}$, and $\dot{x_1} = -(0.9x_1 + 0.1x_2)$ for $q_1 = m_{\texttt{off}}$, (where $K_1 = 1, h_1 = 7, k_{1,2} = 0.1$). The predicates $\mathit{flow}^1_{\texttt{on}}$ and $\mathit{flow}^1_{\texttt{off}}$ for the flow conditions are then expressed by:

$$\mathit{flow}^1_{\texttt{on}}\left(\begin{bmatrix} x_1^0 \\ x_2^0 \\ \tau^0 \end{bmatrix}, \begin{bmatrix} x_1^t \\ x_2^t \\ \tau^t \end{bmatrix}, t\right) \equiv \begin{bmatrix} x_1^t \\ x_2^t \\ \tau^t \end{bmatrix} = \begin{bmatrix} x_1^0 \\ x_2^0 \\ \tau^0 \end{bmatrix} + \int_0^t \begin{bmatrix} 7 - (0.9x_1(t) + 0.1x_2(t)) \\ \dot{x_2}(t) \\ 1 \end{bmatrix} \mathrm{d}t$$

$$\mathit{flow}^1_{\texttt{off}}\left(\begin{bmatrix} x_1^0 \\ x_2^0 \\ \tau^0 \end{bmatrix}, \begin{bmatrix} x_1^t \\ x_2^t \\ \tau^t \end{bmatrix}, t\right) \equiv \begin{bmatrix} x_1^t \\ x_2^t \\ \tau^t \end{bmatrix} = \begin{bmatrix} x_1^0 \\ x_2^0 \\ \tau^0 \end{bmatrix} + \int_0^t \begin{bmatrix} -(0.9x_1(t) + 0.1x_2(t)) \\ \dot{x_2}(t) \\ 1 \end{bmatrix} \mathrm{d}t$$

These predicates contain the *uninterpreted function* $\dot{x_2}$ in the integral terms whose interpretation is given by the other automata H_2 and H_3 (this is different from the isolated case in Example 2 where H_1 only include x_1 and τ). It is *not* possible to evaluate the integration of the ODEs with the uninterpreted function $\dot{x_2}$ by themselves. In particular, the flow predicates have no meaning in $\mathbb{R}_{\mathcal{F}}$. ■

The point is that to modularly encode networks of *tightly coupled* nonlinear hybrid automata, we need to express the physical correlations between different components. Since all variables in a system of ODEs evolve simultaneously over continuous time, naive decoupling as in Formula (2) does not work. By explicitly constructing parallel compositions, we can still use Formula (1), but at the cost of the formula explosion problem. It is very difficult to modularly encode such physical correlations as $\mathcal{L}_{\mathcal{F}}$-formulas by using *only concrete functions*, since each variable may have many solution functions for different combinations of ODEs. In Example 3, x_1 depends on x_2, which again depends on x_3, and therefore solutions functions of x_1 has $2^3 = 8$ different solved forms.

Logically Correct Flow Decomposition. For networks of general nonlinear hybrid automata, the standard decomposition technique using Formula (2) is semantically incorrect as previously stated, since flow predicates may include "undefined" functions (e.g., see Example 3). Adding extra terms to Formula (2) for defining the semantics of such uninterpreted functions (that is, the continuous interconnections between different components), we can obtain a logically correct $\mathcal{L}_{\mathcal{F}}$-formula for networks of hybrid automata. Since ODE variables may have many solved forms for different combinations of ODEs, we consider *parameterized flow predicates* to avoid the formula explosion problem. The idea presented here is also carried over into a new logical framework in Section 4.

We decompose each flow predicate into a conjunction of several formulas according to its structure. Consider a flow predicate for an l-dimensional ODEs $\mathit{flow}_q(\boldsymbol{x^0}, \boldsymbol{x^t}, t) \equiv (\boldsymbol{x^t} = \boldsymbol{x^0} + \int_0^t \mathit{flow}_q(\boldsymbol{x})\,\mathrm{d}t)$. Mathematically, a variable x in $\mathit{flow}(\boldsymbol{x})$ is a unary real function $\mathbb{R} \to \mathbb{R}$ over time, and thus $\mathit{flow}_q(\boldsymbol{x})$ can also be considered as a unary function $\mathbb{R} \to \mathbb{R}^l$. Therefore, using extra function symbols $\dot{x_1}, \ldots, \dot{x_l} : \mathbb{R} \to \mathbb{R}$ to denote the derivatives of $\boldsymbol{x} = (x_1, \ldots, x_l)$, the flow predicate can be rewritten as the logically equivalent conjunction:

$$\boldsymbol{x^t} = \boldsymbol{x^0} + \int_0^t [\dot{x_1}(t), \ldots, \dot{x_l}(t)]\,\mathrm{d}t \ \wedge\ \forall u \in [0,t].\ [\dot{x_1}(u), \ldots, \dot{x_l}(u)] = \mathit{flow}(\boldsymbol{x})(u).$$

The invariant condition *with respect to an arbitrary flow* $\boldsymbol{x}$ is similarly expressed as $\mathcal{L}_{\mathcal{F}}$-formulas using extra function symbols $\dot{\boldsymbol{x}} = (\dot{x_1}, \ldots, \dot{x_l})$. The formula $\mathit{inv}(m, \boldsymbol{x}) \equiv \bigwedge_{q \in Q} (m = q) \to \mathit{inv}_q(\boldsymbol{x})$ uniformly expresses invariant predicates in such a way that $\mathit{inv}(q, \boldsymbol{x}) \equiv \mathit{inv}_q(\boldsymbol{x})$ for mode q. Then, let us define:

$$\begin{aligned}
\mathit{flow}_{\dot{\boldsymbol{x}}}(\boldsymbol{x^0}, \boldsymbol{x^t}, t) &\equiv \boldsymbol{x^t} = \boldsymbol{x^0} + \int_0^t \dot{\boldsymbol{x}}(t)\,\mathrm{d}t, \\
\mathit{inv}_{\dot{\boldsymbol{x}}}(m, \boldsymbol{x^0}, t) &\equiv \forall u \in [0,t], \forall \boldsymbol{x^u}.\ \mathit{flow}_{\dot{\boldsymbol{x}}}(\boldsymbol{x^0}, \boldsymbol{x^u}, u) \to \mathit{inv}(m, \boldsymbol{x^u}).
\end{aligned}$$

Example 4. Consider the flow predicates $\mathit{flow}^1_{\mathtt{on}}$ and $\mathit{flow}^1_{\mathtt{off}}$ in Example 3. The parameterized flow predicate $\mathit{flow}^1_{(\dot{x_1}, \dot{x_2}, \dot{\tau})}([x_1^0, x_2^0, \tau^0], [x_1^t, x_2^t, \tau^t], t)$ using extra function symbols $\dot{\boldsymbol{x_1}} = (\dot{x_1}, \dot{x_2}, \dot{\tau})$ is defined by the formula:

$$[x_1^t, x_2^t, \tau^t] = [x_1^0, x_2^0, \tau^0] + \int_0^t [\dot{x_1}(t), \dot{x_2}(t), \dot{\tau}(t)]\,\mathrm{d}t.$$

For $\boldsymbol{x_1^0} = (x_1^0, x_2^0, \tau^0)$ and $\boldsymbol{x_1^t} = (x_1^t, x_2^t, \tau^t)$, the flow predicates $\mathit{flow}^1_{\mathtt{on}}(\boldsymbol{x_1^0}, \boldsymbol{x_1^t}, t)$ and $\mathit{flow}^1_{\mathtt{off}}(\boldsymbol{x_1^0}, \boldsymbol{x_1^t}, t)$, respectively, are equivalently rewritten to the formulas:

$$\begin{aligned}
&\mathit{flow}^1_{\dot{\boldsymbol{x_1}}}(\boldsymbol{x_1^0}, \boldsymbol{x_1^t}, t) \wedge \forall u \in [0,t].\ \dot{\tau}(u) = 1 \ \wedge\ \dot{x_1}(u) = 7 - (0.9x_1(u) + 0.1x_2(u)) \\
&\mathit{flow}^1_{\dot{\boldsymbol{x_1}}}(\boldsymbol{x_1^0}, \boldsymbol{x_1^t}, t) \wedge \forall u \in [0,t].\ \dot{\tau}(u) = 1 \ \wedge\ \dot{x_1}(u) = -(0.9x_1(t) + 0.1x_2(t))
\end{aligned}$$

The universally quantified subformulas *assign* to the function symbols $\dot{x_1}$ and $\dot{\tau}$ their logical interpretation. Notice that the interpretation of $\dot{x_2}$ can easily be assigned in a similar way by another hybrid automaton. ■

Recall that the continuous behavior of a hybrid automaton in mode q from $\boldsymbol{x^0}$ to $\boldsymbol{x^t}$ for duration t is expressed as the formula $\mathit{flow}_q(\boldsymbol{x^0}, \boldsymbol{x^t}, t) \wedge \mathit{inv}_q(\boldsymbol{x^0}, t)$, which can also be decomposed into the logically equivalent conjunction.

Lemma 1. *The following $\mathcal{L}_{\mathcal{F}}$-formulas are logically equivalent to each other:*

$$\begin{aligned}
&\mathit{flow}_q(\boldsymbol{x^0}, \boldsymbol{x^t}, t) \ \wedge\ \mathit{inv}_q(\boldsymbol{x^0}, t) \\
&\mathit{flow}_{\dot{\boldsymbol{x}}}(\boldsymbol{x^0}, \boldsymbol{x^t}, t) \ \wedge\ \mathit{inv}_{\dot{\boldsymbol{x}}}(m, \boldsymbol{x^0}, t) \wedge (m = q) \ \wedge\ \forall u \in [0,t].\ \dot{\boldsymbol{x}}(u) = \mathit{flow}_q(\boldsymbol{x})(u).
\end{aligned}$$

Proof. Immediate, because by replacing each function symbol $\dot{x} \in \dot{\boldsymbol{x}}$ according to the equalities $\dot{\boldsymbol{x}}(u) = \mathit{flow}_q(\boldsymbol{x})(u)$ in $\mathit{flow}_{\dot{\boldsymbol{x}}}(\boldsymbol{x^0}, \boldsymbol{x^t}, t) \ \wedge\ \mathit{inv}_{\dot{\boldsymbol{x}}}(q, \boldsymbol{x^0}, t)$, and by the equivalence $\mathit{inv}(q, \boldsymbol{x}) \equiv \mathit{inv}_q(\boldsymbol{x})$, we obtain $\mathit{flow}_q(\boldsymbol{x^0}, \boldsymbol{x^t}, t) \wedge \mathit{inv}_q(\boldsymbol{x^0}, t)$. □

Using this equivalence, the non-modular encoding by Formula (1) for k-step bounded reachability of $H_1 \parallel \cdots \parallel H_n$ of size $O(k \cdot \prod_{i=1}^{n} m_i)$, if each H_i has m_i modes, can be rewritten to a *logically equivalent* modular $\mathcal{L}_{\mathcal{F}}$-formula of size $O(k \cdot \sum_{j=1}^{n} m_j^2)$. Unlike the standard decomposition by Formula (2), the new encoding is logically correct for general nonlinear hybrid automata.

Theorem 1. *A k-step bounded reachability problem of a parallel composition $H_1 \parallel \cdots \parallel H_n$ is encoded as the $\mathcal{L}_{\mathcal{F}}$-formula of size $O(k \cdot \sum_{j=1}^{n} m_j^2)$, using extra function symbols $\dot{\boldsymbol{x}}_{\boldsymbol{j_i}}$ for each component $1 \le j \le n$ and each step $0 \le i \le k$:*

$$
\begin{aligned}
\exists \{m_0^j, \ldots, m_k^j,\ \boldsymbol{x_{j0}^0}, \ldots, \boldsymbol{x_{jk}^0}, \boldsymbol{x_{j0}^t}, \ldots, \boldsymbol{x_{jk}^t}\}_{j=1}^{n},\ \exists t_0, \ldots, t_k \in [0, t_{\max}].& \\
\bigwedge_{j=1}^{n} \bigwedge_{i=0}^{k} \left[\mathit{flow}^j_{\dot{\boldsymbol{x}}_{\boldsymbol{j_i}}}(\boldsymbol{x_{ji}^0}, \boldsymbol{x_{ji}^t}, t_i) \ \wedge\ \mathit{inv}^j_{\dot{\boldsymbol{x}}_{\boldsymbol{j_i}}}(m_i^j, \boldsymbol{x_{ji}^0}, t_i) \right] \ \wedge& \\
\bigwedge_{j=1}^{n} \bigvee_{q_j \in Q_j} \left[\begin{array}{r} m_0^j = q_j \ \wedge\ \mathit{init}^j_{q_j}(\boldsymbol{x_{j0}^0}) \\ \wedge\ \forall t \in [0, t_0].\ \dot{\boldsymbol{x}}_{\boldsymbol{j0}}(t) = \mathit{flow}^j_{q_j}(\boldsymbol{x_{j0}})(t) \end{array} \right] \ \wedge& \\
\bigwedge_{j=1}^{n} \bigwedge_{i=1}^{k} \bigvee_{\substack{q_j, r_j \in Q \\ a \in \bigcup_{l=1}^{n} \Sigma_l}} \left[\begin{array}{r} (m_{i-1}^j, m_i^j) = (q_j, r_j) \ \wedge\ \mathit{jump}^j_{q,a,r}(\boldsymbol{x_{ji-1}^t}, \boldsymbol{x_{ji}^0}) \\ \wedge\ \forall t \in [0, t_i].\ \dot{\boldsymbol{x}}_{\boldsymbol{j_i}}(t) = \mathit{flow}^j_{r_j}(\boldsymbol{x_{ji}})(t) \end{array} \right] \ \wedge& \\
\neg \bigwedge_{j=1}^{n} \mathit{safe}(\boldsymbol{x_{jk}^t}) &
\end{aligned}
\tag{3}
$$

Proof (Sketch). For the initial step of Formula (3), consider the parameterized subformula $\bigwedge_{j=1}^{n} \mathit{flow}^j_{\dot{\boldsymbol{x}}_{\boldsymbol{j0}}}(\boldsymbol{x_{j0}^0}, \boldsymbol{x_{j0}^t}, t_0) \wedge \mathit{inv}^j_{\dot{\boldsymbol{x}}_{\boldsymbol{j0}}}(m_0^j, \boldsymbol{x_{j0}^0}, t_0)$ and the subformula $\bigwedge_{j=1}^{n} \bigvee_{q_j \in Q_j} (m_0^j = q_j \ \wedge\ \mathit{init}^j_{q_j}(\boldsymbol{x_{j0}^0}) \ \wedge\ \forall t \in [0, t_0].\ \dot{\boldsymbol{x}}_{\boldsymbol{j0}}(t) = \mathit{flow}^j_{q_j}(\boldsymbol{x_{j0}})(t))$. Using the distributive law, the conjunction of these formulas is rewritten to the "big" disjunction of the following formulas for $(q_1, \ldots, q_n) \in Q_1 \times \cdots \times Q_n$: $\bigwedge_{j=1}^{n} \big(m_0^j = q_j \ \wedge\ \mathit{init}_{q_j}(\boldsymbol{x_{j0}^0}) \ \wedge\ \mathit{flow}^j_{\dot{\boldsymbol{x}}_{\boldsymbol{j0}}}(\boldsymbol{x_{j0}^0}, \boldsymbol{x_{j0}^t}, t_0) \ \wedge\ \mathit{inv}^j_{\dot{\boldsymbol{x}}_{\boldsymbol{j0}}}(m_0^j, \boldsymbol{x_{j0}^0}, t_0) \ \wedge\ (\forall t \in [0, t_0].\ \dot{\boldsymbol{x}}_{\boldsymbol{j0}}(t) = \mathit{flow}^j_{q_j}(\boldsymbol{x_{j0}})(t))\big)$. Using Lemma 1, every extra function symbol in $\dot{\boldsymbol{x}}_{\boldsymbol{j0}}$ can be removed from this formula. The resulting $\mathcal{L}_{\mathcal{F}}$-formula (without uninterpreted functions) exactly describes the concrete behavior of $H_1 \parallel \cdots \parallel H_n$ for the combined mode $(q_1, \ldots, q_n) \in Q_1 \times \cdots \times Q_n$ in the initial step. The case of the intermediate steps is similar. □

4 Modular Logic for Networks of Hybrid Automata

Formal analysis of networks of nonlinear hybrid automata can be encoded as logical formulas in a modular way using extra function symbols. For decidability, we use δ-complete SMT solving to check the satisfiability of such formulas up to a given precision $\delta > 0$ [10,12]. However, the use of uninterpreted real functions and universal quantification makes δ-complete SMT solving very difficult. For this reason, we present a new SMT framework to provide an efficient (δ-complete) decision procedure for the new encoding, by extending the standard theory of the real numbers by adding parameterized integration operators.

Universal Quantification and δ-Complete SMT. The satisfiability of SMT formulas over the real numbers involving nonlinear real functions can be decided up to any given precision $\delta > 0$ by δ-complete decision procedures. A *δ-complete decision procedure* for a formula ϕ returns false if ϕ is unsatisfiable, and returns true if its *syntactic numerical perturbation* of ϕ by δ is satisfiable.[5] A δ-complete SMT solving algorithm is based on the standard DPLL($\mathcal{T}$) framework, together with conflict driven clause learning (CDCL), where the underlying theory $\mathcal{T}$ is the theory of the real numbers with computable real functions.

For δ-complete SMT solving, it is difficult to identify inconsistent *universally quantified* subformulas in Formula (3) for the new encoding. Finding conflict subformulas is important for pruning the Boolean search space for DPLL($\mathcal{T}$) and for learning conflict clauses by CDCL. Without the underlying (computationally expensive) $\mathcal{T}$-solver, it is *not* possible to check whether such subformulas are consistent or not, because the consistency may depend on $\delta > 0$. For example, consider two real functions $f_1(t) = \sqrt{t+1}$ and $f_2(t) = 1 + \frac{1}{2}t - \frac{1}{8}t^2 + \frac{1}{16}x^3$ (where $f_2(t)$ is the third order taylor expansion of $\sqrt{t+1}$). For $u = 0.8$, the formulas $\forall t \in [0, u].\ \dot{x}(t) = f_1(t)$ and $\forall t \in [0, u].\ \dot{x}(t) = f_2(t)$ are inconsistent up to precision $\delta = 0.01$, but consistent up to precision $\delta = 0.1$.

It is also difficult to inform the underlying SMT solver about the structural information of hybrid automata using $\mathcal{L}_{\mathcal{F}}$-formulas. For example, each mode q of one hybrid automaton H only corresponds to one flow condition $\mathit{flow}_q(\boldsymbol{x})$. Therefore, if the truth of one flow condition for mode q is determined to be *true*, then the truths of other flow conditions are not relevant to the satisfaction of the formula. In practice, the performance of SMT solving can be increased using this knowledge. However, DPLL($\mathcal{T}$) and CDCL cannot effectively use it, since universally quantified subformulas, such as $\forall t \in [0, t_i].\ \dot{\boldsymbol{x}}(t) = \mathit{flow}_q(\boldsymbol{x})(t)$, have this structural information in Formula (3). For δ-Complete SMT solving, deriving conflict clauses from such subformulas is computationally expensive, and statically encoding such knowledge in $\mathcal{L}_{\mathcal{F}}$ is generally not possible.

Theory of the Real Numbers with Function Names. Instead of using extra function symbols and universal quantification, we consider another logical theory that extends the standard theory of the real numbers. Formal analysis of networks of nonlinear hybrid systems can still be modularly encoded in this theory by means of Formula (3). But universal quantification is not needed anymore, and thus it provides a simple decision procedure. We can also statically encode the correspondence between modes and flows in this theory.

We consider a *two-sorted* first-order logic with sorts *Real* and *Name*, where *Real* denotes the real numbers, and *Name* denotes *name constants* for unary real functions composed of the functions in $\mathcal{F}$. For example, given a set of real functions $\{1, +, \times, \sin, x, y\} \subset \mathcal{F}$ and two name constants $\mathtt{m}_1$ and $\mathtt{m}_2$ of sort *Name*, we can have two *named* unary real functions:

$$(\mathtt{m}_1)\quad \sin(x(t)) : \mathit{Real} \to \mathit{Real}, \qquad (\mathtt{m}_2)\quad [y(t)+1, z(t)^2] : \mathit{Real} \to \mathit{Real}^2.$$

[5] For example, if $\phi \equiv (x > 3) \wedge (y = z)$, then its syntactic numerical perturbation by $\delta > 0$ is $(x - 3 > -\delta) \wedge (y - z \geq -\delta) \wedge (z - y \geq -\delta)$.

A collection of *application operators* $app^n : Name \times Real \to Real^n$ connects each name constant to its underlying function with range $Real^n$, for example, $app^1(\mathtt{m}_1, 3) \equiv \sin(x(3))$ and $app^2(\mathtt{m}_2, u) \equiv [y(u)+1, z(u)^2]$.

Unlike the previous approach, we explicitly take into account a collection of *integral operators* $int^{k_1,\ldots,k_n} : Real \times Name^n \to Real^{\sum_{i=1}^n k_i}$. An integral term $int^{k_1,\ldots,k_n}(u, \nu_1, \ldots, \nu_n)$ takes time value u and a list of name constants $\nu_1, \ldots, \nu_n$, to respectively denote unary real functions $f_1, \ldots, f_n$ with ranges $Real^{k_1}, \ldots, Real^{k_n}$, and returns the value $\int_0^u [f_1(t), \ldots, f_n(t)]\,\mathrm{d}t$. For example:

$$int^2(1, \mathtt{m}_2) \equiv \int_0^1 [y+1, z^2]\,\mathrm{d}t, \qquad int^{1,2}(t, \mathtt{m}_1, \mathtt{m}_2) \equiv \int_0^t [\sin(x), y+1, z^2]\,\mathrm{d}t.$$

Notice that solutions of ODEs are *no longer* atomic functions in the new logic, but can be constructed using integral operators and function names.

Definition 4. *For a set $\mathcal{N}$ of name constants and a set $\mathcal{O}$ of operators app^n and $int^{k_1,\ldots,k_n}$, the first order signature is given by $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}} = (\mathcal{F} \cup \mathcal{N} \cup \mathcal{O}, >)$. The first order structure is given by $\mathbb{R}_{\mathcal{F}\cup\mathcal{N}} = (\mathbb{R} \cup N, \mathcal{F}^{\mathbb{R}} \cup \mathcal{N}^N \cup \mathcal{O}^{N,\mathbb{R}}, >^{\mathbb{R}})$ for a fixed finite set N of* name objects*, where the interpretation $\mathcal{N}^N$ of name constants and the interpretation $\mathcal{O}^{N,\mathbb{R}}$ of application and integral operators are given as explained above. The syntax and semantics of $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-formulas is defined by means of $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ and $\mathbb{R}_{\mathcal{F}\cup\mathcal{N}}$ in the standard way.*[6]

The satisfiability problems of $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-formulas in the new theory $\mathbb{R}_{\mathcal{F}\cup\mathcal{N}}$ can be reduced to ones in the standard theory $\mathbb{R}_{\mathcal{F}}$ at minimal cost, whose satisfiability can be determined by using δ-complete decision procedures, provided that all *Name* variables in the formulas are only existentially quantified. This reduction process for $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-formulas can be easily combined with the underlying DPLL($\mathcal{T}$) and CDCL frameworks for δ-complete SMT solving of $\mathcal{L}_{\mathcal{F}}$-formulas.

Theorem 2. *The satisfiability of $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ formulas of the form $\exists \boldsymbol{n}.\, \psi(\boldsymbol{n})$, where $\boldsymbol{n}$ are only name variables in $\psi(\boldsymbol{n})$, is decidable by δ-complete SMT solving.*

Proof. Because there are only a finite number of name objects in $\mathbb{R}_{\mathcal{F}\cup\mathcal{N}}$, by using the standard SMT solving for equalities, we can enumerate all *consistent* assignments $\boldsymbol{n} = \overrightarrow{name}_1, \ldots, \boldsymbol{n} = \overrightarrow{name}_N$ for $\boldsymbol{n}$. If there is no suchconsistent assignment for $\boldsymbol{n}$, then $\exists \boldsymbol{n}.\, \psi(\boldsymbol{n})$ is not satisfiable. Let $\hat{\psi}_i$ be the formula obtained from $\psi(\boldsymbol{n})$ by replacing each name variable by its related function according to the i-th assignment $\boldsymbol{n} = \overrightarrow{name}_i$. Then, each $\hat{\psi}_i$ is an ordinary $\mathcal{L}_{\mathcal{F}}$-formula. □

$\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-Encoding for Networks of Hybrid Automata. Formula (3) for the encoding in $\mathcal{L}_{\mathcal{F}}$ can be rewritten to a logically equivalent $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ formula. Each extra function symbol $\dot{x}_i$ is replaced by a name variable w_i, and each universally quantified formula $\forall t \in [0, t_i].\, [\dot{x}_1(t), \ldots, \dot{x}_l(t)] = [\mathit{flow}_1(\boldsymbol{x})(t), \ldots, \mathit{flow}_l(\boldsymbol{x})(t)]$ is replaced by equalities $w_1 = \mathtt{flow}_1 \wedge \cdots \wedge w_l = \mathtt{flow}_l$, where name *constant* $\mathtt{flow}_i$ denotes the function $\mathit{flow}_i(\boldsymbol{x})(t)$ (i.e., $app^l(\mathtt{flow}_i, t) \equiv \mathit{flow}_i(\boldsymbol{x})(t)$).

[6] The signature $\mathcal{L}_{\mathcal{F}}$ is a subsignature of $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$, and the structure $\mathbb{R}_{\mathcal{F}}$ for the real numbers is a substructure of $\mathbb{R}_{\mathcal{F}\cup\mathcal{N}}$.

Theorem 3. *A k-step bounded reachability problem of a parallel composition $H_1 \parallel \cdots \parallel H_n$ is encoded as the $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-formula of size $O(k \cdot \sum_{j=1}^{n} m_j^2)$, where $inv(\boldsymbol{w}, m, \boldsymbol{x^0}, t) \equiv \forall u \in [0,t], \forall \boldsymbol{x^u}.\ (\boldsymbol{x^t} = \boldsymbol{x^0} + int^{\boldsymbol{l}}(t, \boldsymbol{w})) \rightarrow inv(m, \boldsymbol{x^u})$:*

$$
\begin{aligned}
&\exists \{m_0^j, \ldots, m_k^j,\ \boldsymbol{w_0^j}, \ldots, \boldsymbol{w_k^j},\ \boldsymbol{x_{j0}^0}, \ldots, \boldsymbol{x_{jk}^t}\}_{j=1}^{n},\ \exists t_0, \ldots, t_k \in [0, t_{\max}]. \\
&\quad \textstyle\bigwedge_{j=1}^{n} \bigwedge_{i=0}^{k} \left[\boldsymbol{x_{ji}^t} = \boldsymbol{x_{ji}^0} + int^{\boldsymbol{l}}(t_i, \boldsymbol{w_i^j}) \ \wedge\ inv^j(\boldsymbol{w_i^j}, m_i^j, \boldsymbol{x_{ji}^0}, t_i) \right] \ \wedge \\
&\quad \textstyle\bigwedge_{j=1}^{n} \bigvee_{q_j \in Q_j} \left[m_0^j = q_j \ \wedge\ init_{q_j}^j(\boldsymbol{x_{j0}^0}) \ \wedge\ \boldsymbol{w_0^j} = \mathtt{flow}_{q_j}^j \right] \ \wedge \qquad (4) \\
&\textstyle\bigwedge_{j=1}^{n} \bigwedge_{i=1}^{k} \bigvee_{\substack{q_j, r_j \in Q \\ a \in \bigcup_{l=1}^{n} \Sigma_l}} \left[\begin{array}{r} (m_{i-1}^j, m_i^j) = (q_j, r_j) \ \wedge\ \boldsymbol{w_i^j} = \mathtt{flow}_{r_j}^j \\ \wedge\ jump_{q,a,r}^j(\boldsymbol{x_{ji-1}^t}, \boldsymbol{x_{ji}^0}) \end{array} \right] \ \wedge\ \neg \textstyle\bigwedge_{j=1}^{n} safe(\boldsymbol{x_{jk}^t})
\end{aligned}
$$

Proof (Sketch). This immediately follows from $\mathbb{R}_{\mathcal{F}} \subseteq \mathbb{R}_{\mathcal{F}\cup\mathcal{N}}$ and the formula rewriting process in the proof of Theorem 1. For the initial step, we have the "big" disjunction of: $\bigwedge_{j=1}^{n} \big(m_0^j = q_j \wedge init_{q_j}(\boldsymbol{x_{j0}^0}) \wedge (\boldsymbol{x_{j0}^t} = \boldsymbol{x_{j0}^0} + int^{\boldsymbol{l}}(t_0, \boldsymbol{w_0^j})) \wedge inv^j(\boldsymbol{w_0^j}, m_0^j, \boldsymbol{x_{j0}^0}, t_0)\ \wedge\ \boldsymbol{w_0^j} = \mathtt{flow}_{q_j}^j\big)$ for $(q_1, \ldots, q_n) \in Q_1 \times \cdots \times Q_n$. The formula obtained by replacing $\boldsymbol{w_0^j}$ by $\mathtt{flow}_{q_j}^j$ gives the concrete behavior for $(q_1, \ldots, q_n) \in Q_1 \times \cdots \times Q_n$ in the initial step. The other cases are similar. □

Example 5. Consider the flow predicates $flow_{\mathtt{on}}^1$ and $flow_{\mathtt{off}}^1$ in Example 3 again. The flow predicates $flow_{\mathtt{on}}^1(\boldsymbol{x_1^0}, \boldsymbol{x_1^t}, t)$ and $flow_{\mathtt{off}}^1(\boldsymbol{x_1^0}, \boldsymbol{x_1^t}, t)$, for $\boldsymbol{x_1^0} = (x_1^0, x_2^0, \tau^0)$ and $\boldsymbol{x_1^t} = (x_1^t, x_2^t, \tau^t)$ are respectively rewritten to the $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-formulas:

$$
\begin{aligned}
&\boldsymbol{x_1^t} = \boldsymbol{x_1^0} + int^{1,1,1}(t, [w_1, w_2, w_\tau]) \ \wedge\ w_\tau = \mathtt{one} \ \wedge\ w_1 = \mathtt{flow}_{\mathtt{on}}^1 \\
&\boldsymbol{x_1^t} = \boldsymbol{x_1^0} + int^{1,1,1}(t, [w_1, w_2, w_\tau]) \ \wedge\ w_\tau = \mathtt{one} \ \wedge\ w_1 = \mathtt{flow}_{\mathtt{off}}^1
\end{aligned}
$$

using name variables (w_1, w_2, w_τ) and name constants $\mathtt{one}$, denoting 1, $\mathtt{flow}_{\mathtt{on}}^1$, denoting $7 - (0.9x_1 + 0.1x_2)$, and $\mathtt{flow}_{\mathtt{off}}^1$, denoting $-(0.9x_1 + 0.1x_2)$. Another component H_2 can declare an assignment of w_2 (e.g, using $w_2 = \mathtt{flow}_{\mathtt{on}}^2$). ■

This new $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-encoding has a number of significant advantages. It requires no universal quantification with uninterpreted real function symbols for logically correct modular encoding for networks of nonlinear hybrid automata. Therefore, the decision procedure is simple and easily combined with DPLL($\mathcal{T}$) and CDCL. In particular, identifying conflict equalities $w = \mathtt{flow}_q$ and $w' = \mathtt{flow}_{q'}$ is much more efficient than identifying conflict universally quantified subformulas (which is difficult for δ-complete SMT solving as explained above). We can statically add *uniqueness lemmas* of the form $\neg(w = \mathtt{flow}_q \ \wedge\ w = \mathtt{flow}_{q'})$ for any two modes $q \neq q'$ in one hybrid automaton (because each mode has one flow condition), or for any two *incompatible* flow functions $\mathtt{flow}_q$ and $\mathtt{flow}_{q'}$. This static method can greatly reduce the Boolean search space for DPLL($\mathcal{T}$), but cannot be applied to $\mathcal{L}_{\mathcal{F}}$-encoding as previously discussed.

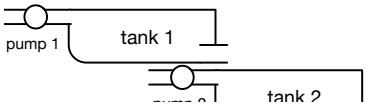

Fig. 2. Connected water tanks

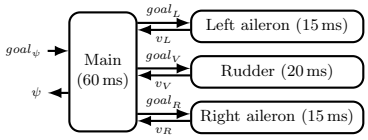

Fig. 3. Controllers for turning an airplane

5 Case Studies and Experimental Results

This section gives an overview of some case studies for SMT-based analysis of networks of *nonlinear* hybrid systems, in addition to the networked thermostats in Example 1. These case studies involve nontrivial (nonlinear) ODEs, due to the physical connections between different components. We have implemented our algorithm for $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ in the dReal SMT solver, and have performed bounded reachability analysis of these case studies. All experiments were conducted on an Intel Xeon 2.0 GHz with 64 GB memory. The case studies and the experimental evaluation are available at `http://dreal.github.io/benchmarks/networks`.

Physically Networked Water Tanks. A number of water tanks are connected by pipes as shown in Figure 2. The water level in each tank is separately controlled by the pump in the tank (adapted from [15,17]). Each water level x_i of tank i depends on the pump's mode $m_i \in \{m_{\texttt{on}}, m_{\texttt{off}}\}$ and the levels of the adjacent tanks, according to the nonlinear ODEs ($x_0 = 0$ for the leftmost tank 1):

$$\begin{aligned} A_i\dot{x}_i &= (q_i + a\sqrt{2g}\sqrt{x_{i-1}}) - a\sqrt{2g}\sqrt{x_i} && \text{if } m_i = m_{\texttt{on}} \\ A_i\dot{x}_i &= a\sqrt{2g}\sqrt{x_{i-1}} - a\sqrt{2g}\sqrt{x_i} && \text{if } m_i = m_{\texttt{off}} \end{aligned}$$

Every pipe controller synchronously performs its discrete transitions: for each second, the pump is on if $x_i \le L_{\min}$, and off if $x_i > L_{\max}$.

Turning an Airplane. We consider a networked multirate control system for turning an airplane, depicted in Figure 3 (adapted from [2]). To make a turn, an aircraft rolls by moving its ailerons, while its rudder is used to counter adverse yaw. The subcontrollers for the ailerons and the rudder separately control the respective surface angles according to modes (*up* and *down*) and the ODEs:

$$\dot{\delta_{la}} = rate_{la} \quad \text{(left)}, \qquad \dot{\delta_{ra}} = rate_{ra} \quad \text{(right)}, \qquad \dot{\delta_r} = rate_r \quad \text{(rudder)}.$$

The moving rates $rate_{la}$, $rate_{ra}$, and $rate_r$ are given by the main controller, based on the lateral dynamics of an aircraft specified as nonlinear ODEs [19]:

$$\begin{aligned} \dot{\beta} &= Y_{(\beta,\delta_{la},\delta_{ra},\delta_r)}/mV - r + V/g * \cos(\beta) * \sin(\phi) & \dot{\phi} &= p \\ \dot{p} &= (c_1 r + c_2 p) r \tan(\phi) + c_3 L_{(\beta,\delta_{la},\delta_{ra},\delta_r)} + c_4 N_{(\beta,\delta_{la},\delta_{ra},\delta_r)} & \dot{\psi} &= g/V \tan(\phi) \\ \dot{r} &= (c_8 p - c_2 r) r \tan(\phi) + c_4 L_{(\beta,\delta_{la},\delta_{ra},\delta_r)} + c_9 N_{(\beta,\delta_{la},\delta_{ra},\delta_r)} \end{aligned}$$

with variables β the yaw angle, p the rolling moment, r the yawing moment, ϕ the roll angle, and ψ the direction, and linear functions $Y_{(\beta,\delta_{la},\delta_{ra},\delta_r)}$, $L_{(\beta,\delta_{la},\delta_{ra},\delta_r)}$, and $N_{(\beta,\delta_{la},\delta_{ra},\delta_r)}$ determined by the shape and status of the aircraft.

Multiple Battery Usage. There are a number of fully charged batteries, and a control system switches load between these batteries to achieve longer lifetime (adapted from [7]). Each battery i has three modes *switchedOn*, *switchedOff*, and *dead*. The battery charge dynamics is expressed by the following ODEs, where d_i denotes its kinetic energy difference, and g_i denotes its total charge:

$$\text{(on)} \begin{bmatrix} \dot{d}_i = L/c - k \cdot d_i \\ \dot{g}_i = -L \end{bmatrix}, \qquad \text{(off)} \begin{bmatrix} \dot{d}_i = -k \cdot d_i \\ \dot{g}_i = 0 \end{bmatrix}, \qquad \text{(dead)} \begin{bmatrix} \dot{d}_i = 0 \\ \dot{g}_i = 0 \end{bmatrix},$$

If $g_i > (1-c)d_i$, then battery i is dead, and otherwise, it can be either on or off. When $k \in \mathbb{N}$ batteries are on, then load to each battery is divided by k.

Experimental Results. We have performed bounded reachability up to bound $k = 5$ (for the multirate airplane example, $k = 10$) for safety properties for each example. We consider both unsat (verified) and sat (counterexample found) cases by using different safety properties. We set a timeout of 30 hours for the experiments. The results for k-step bounded reachability are summarized in Figure 4 (see Table 1 for more details).[7] According to the results, the performance of the new encoding and the heuristics is mostly dramatically better than one of the standard encoding. For example, when we consider the thermostat and water tank examples of three components, the SMT-based analysis using the standard encoding did not terminate for 30 hours even for $k = 1$. The performance of the new encoding is similar to one of the heuristics. But for sat cases, the new encoding is much faster than the heuristics, since the heuristics explicitly generates all the mode paths to find a counterexample, whereas SMT is generally effective for finding counterexamples, thanks to DPLL($\mathcal{T}$) and CDCL.

6 Concluding Remarks

We have shown that formal analysis problems of networks of general nonlinear hybrid systems can be effectively expressed as SMT formulas in a modular way. This is useful to cope with the formula explosion problem, when analyzing networks of tightly coupled nonlinear hybrid automata. We have presented an extended logical theory for hybrid systems that can express the new encoding and provided an SMT decision procedure that involves no extra cost. We have implemented these techniques in the dReal SMT solver. Our experiments have shown that our techniques dramatically increase the performance of SMT-based analysis for networks of hybrid systems with multiple control modes and nonlinear ODEs up to precision δ.

[7] For the airplane example, we consider three variants: a single-rate nonlinear model, an approximated linear model by linear differential equations [1], and a multirate linear model in which the different controllers have different periods.

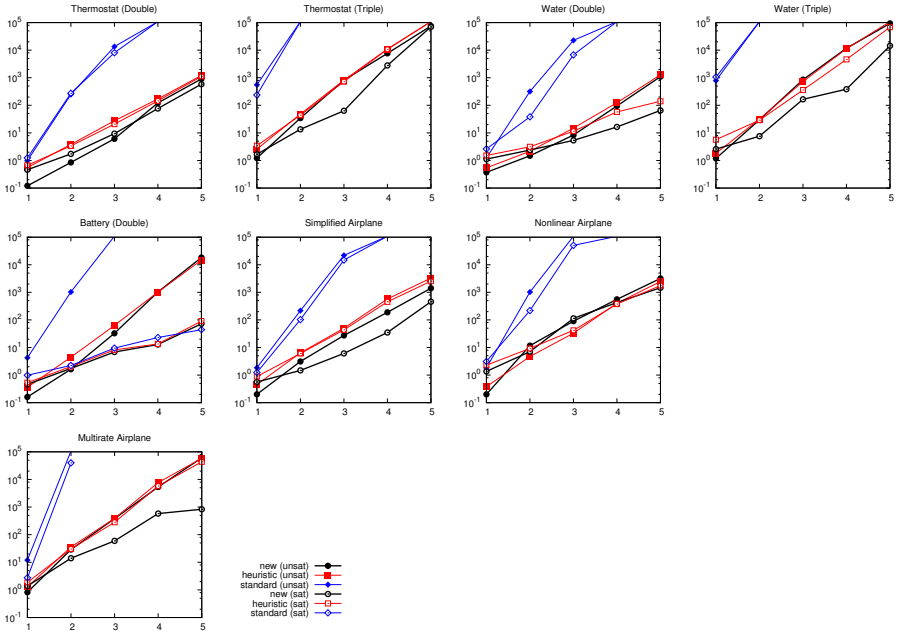

Fig. 4. Running time of k-step bounded reachability analysis, where red lines denote the new encoding, and blue lines denote the non-modular encoding.

References

1. Allerton, D.: Principles of flight simulation. John Wiley & Sons (2009)
2. Bae, K., Krisiloff, J., Meseguer, J., Ölveczky, P.C.: Designing and verifying distributed cyber-physical systems using Multirate PALS: An airplane turning control system case study. Science of Computer Programming (2014), to appear
3. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings. pp. 258–263 (2013), http://dx.doi.org/10.1007/978-3-642-39799-8_18
4. Cimatti, A., Mover, S., Tonetta, S.: A quantifier-free SMT encoding of non-linear hybrid automata. In: Formal Methods in Computer-Aided Design, FMCAD 2012, Cambridge, UK, October 22-25, 2012. pp. 187–195 (2012), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6462573
5. Cimatti, A., Mover, S., Tonetta, S.: Smt-based verification of hybrid systems. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada. (2012), http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5072
6. Eggers, A., Fränzle, M., Herde, C.: SAT modulo ODE: A direct SAT approach to hybrid systems. In: Automated Technology for Verification and Analysis, 6th International Symposium, ATVA 2008, Seoul, Korea, October 20-23, 2008. Proceedings. pp. 171–185 (2008), http://dx.doi.org/10.1007/978-3-540-88387-6_14

7. Fox, M., Long, D., Magazzeni, D., Velez, J., Hemann, G., Huang, A., Posner, I., Roy, N., Planken, L., de Weerdt, M., et al.: Plan-based policies for efficient multiple battery load management. Journal of Artificial Intelligence Research pp. 335–382 (2012)
8. Fränzle, M., Herde, C.: Hysat: An efficient proof engine for bounded model checking of hybrid systems. Formal Methods in System Design 30(3), 179–198 (2007), http://dx.doi.org/10.1007/s10703-006-0031-0
9. Frehse, G., Guernic, C.L., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: Spaceex: Scalable verification of hybrid systems. In: Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings. pp. 379–395 (2011), http://dx.doi.org/10.1007/978-3-642-22110-1_30
10. Gao, S., Avigad, J., Clarke, E.M.: δ-complete decision procedures for satisfiability over the reals. In: IJCAR. Lecture Notes in Computer Science, vol. 7364, pp. 286–300. Springer (2012)
11. Gao, S., Kong, S., Clarke, E.M.: dReal: An SMT solver for nonlinear theories over the reals. In: CADE. Lecture Notes in Computer Science, vol. 7898, pp. 208–214. Springer (2013)
12. Gao, S., Kong, S., Clarke, E.M.: Satisfiability modulo odes. In: FMCAD. pp. 105–112. IEEE (2013)
13. Henzinger, T.A.: The theory of hybrid automata. Springer (2000)
14. Kong, S., Gao, S., Chen, W., Clarke, E.M.: dReach: δ-reachability analysis for hybrid systems. In: TACAS. p. to appear (2015)
15. Kowalewski, S., Stursberg, O., Fritz, M., Graf, H., Hoffmann, I., Preußig, J., Remelhe, M., Simon, S., Treseler, H.: A case study in tool-aided analysis of discretely controlled continuous systems: the two tanks problem. In: Hybrid Systems V, pp. 163–185. Springer (1999)
16. Lynch, N., Segala, R., Vaandrager, F.: Hybrid I/O automata. Information and Computation 185(1), 105–157 (2003)
17. Raisch, J., Klein, E., Meder, C., Itigin, A., OYoung, S.: Approximating automata and discrete control for continuous systems two examples from process control. In: Hybrid systems V, pp. 279–303. Springer (1999)
18. Ratschan, S., She, Z.: Safety verification of hybrid systems by constraint propagation based abstraction refinement. In: Hybrid Systems: Computation and Control, 8th International Workshop, HSCC 2005, Zurich, Switzerland, March 9-11, 2005, Proceedings. pp. 573–589 (2005), http://dx.doi.org/10.1007/978-3-540-31954-2_37
19. Stevens, B.L., Lewis, F.L.: Aircraft control and simulation. John Wiley & Sons (2003)

Benchmark			Time (s)				
			$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
Thermo (double)	unsat	new	0.12	0.85	6.10	121.94	907.71
		heuristics	0.55	3.84	27.44	169.61	1211.51
		standard	0.99	246.73	13448.89	-	-
	sat	new	0.46	1.73	9.40	76.60	586.66
		heuristics	0.67	3.37	20.73	145.34	1114.85
		standard	1.24	273.95	8059.73	-	-
Thermo (triple)	unsat	new	1.22	34.50	816.71	7651.68	74980.37
		heuristics	2.59	48.27	812.33	11038.87	-
		standard	552.68	-	-	-	-
	sat	new	1.68	13.51	63.20	2785.33	70303.00
		heuristics	3.45	43.78	724.19	10757.28	-
		standard	236.95	-	-	-	-
Water (double)	unsat	new	0.37	1.48	8.33	94.66	1098.63
		heuristics	0.54	2.15	14.98	124.28	1278.40
		standard	1.24	317.22	22688.95	-	-
	sat	new	1.12	2.40	5.36	16.40	64.99
		heuristics	1.52	3.13	11.57	57.68	140.64
		standard	2.62	38.19	6771.18	-	-
Water (triple)	unsat	new	1.18	29.74	851.47	11736.80	93326.38
		heuristics	1.81	31.76	729.93	11228.41	-
		standard	786.58	-	-	-	-
	sat	new	2.59	7.60	165.07	383.52	14373.01
		heuristics	5.67	28.87	360.23	4630.00	68224.44
		standard	1062.09	-	-	-	-
Battery	unsat	new	0.16	1.63	32.66	1027.05	18296.20
		heuristics	0.36	4.43	63.02	1019.79	14842.02
		standard	4.24	1028.82	-	-	-
	sat	new	0.46	1.74	6.85	12.87	72.60
		heuristics	0.53	2.04	8.20	13.46	91.36
		standard	0.99	2.22	9.49	22.99	44.22
Simplified Airplane	unsat	new	0.20	3.12	27.34	188.36	1406.45
		heuristics	0.45	6.40	49.94	574.67	3279.90
		standard	1.84	217.17	21874.82	-	-
	sat	new	0.56	1.48	6.09	34.93	457.75
		heuristics	0.89	6.06	43.50	452.44	2539.44
		standard	1.24	102.51	14794.61	-	-
Nonlinear Airplane	unsat	new	0.20	11.69	90.06	555.16	3187.93
		heuristics	0.38	4.71	33.09	401.27	2465.69
		standard	1.82	1022.98	-	-	-
	sat	new	1.36	7.09	113.90	428.86	1468.34
		heuristics	2.29	9.29	41.76	375.38	1767.21
		standard	3.12	218.94	50420.57	-	-
			$k=2$	$k=4$	$k=6$	$k=8$	$k=10$
Multirate Airplane	unsat	new	0.82	29.14	382.83	5359.49	60918.94
		heuristics	1.28	35.83	389.34	7687.09	58087.00
		standard	12.01	-	-	-	-
	sat	new	1.38	14.03	59.64	580.47	834.44
		heuristics	1.88	30.21	282.89	5763.48	43635.96
		standard	2.75	40016.59	-	-	-

Table 1. Running time of k-step bounded model checking