Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG- 2010-IC-002

BACH 2 : Bounded ReachAbility CHecker for Compositional Linear Hybrid Systems

Lei Bu, You Li, Linzhang Wang, Xin Chen, Xuandong Li

Postprint Version. Originally Published in: ACM Press, 2010, pp.1512-1517.

BACH 2 : *B*ounded Reach*A*bility *CH*ecker for Compositional Linear Hybrid Systems

Lei Bu, You Li, Linzhang Wang, Xin Chen, and Xuandong Li
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, P.R.China 210093
Department of Computer Science and Technology, Nanjing University, Nanjing, Jiangsu, P.R.China 210093
Email: {bl|leo86}@seg.nju.edu.cn, {chenxin|lzwang|lxd}@nju.edu.cn

***Abstract*—Existing reachability analysis techniques are easy to fail when applied to large compositional linear hybrid systems, since their memory usages rise up quickly with the increase of systems' size. To address this problem, we propose a tool BACH 2 that adopts a path-oriented method for bounded reachability analysis of compositional linear hybrid systems. For each component, a path is selected and all selected paths compose a path set for reachability analysis. Each path is independently encoded to a set of constraints while synchronization controls are encoded as a set of constraints too. By merging all the constraints into one set, the path-oriented reachability problem of a path set can be transformed to the feasibility problem of this resulting linear constraint set, which can be solved by linear programming efficiently. Based on this path-oriented method, BACH 2 adopts a shared label sequence guided depth first search (SLS-DFS) method to perform bounded reachability analysis of compositional linear hybrid system, where all potential path sets within the bound limit are identified and verified one by one. By this means, since only the structure of a system and the recently visited one path in each component need to be stored in memory, memory consumption of BACH 2 is very small at runtime. As a result, BACH 2 enables the verification of extremely large systems, as is demonstrated in our experiments.**

I. Introduction

Hybrid automata [1] are well studied formal models for hybrid systems with both discrete and continuous state changes. Unfortunately, hybrid automata are very difficult to analyze. Even for a simple class such as *linear hybrid automata* (LHA), the reachability problem is undecidable [1][2]. Existing techniques do not scale well with the problem sizes of practical interest as they need to perform expensive polyhedral computation, and the complexity of their algorithms is exponential in the number of variables in the automata. Especially when performing the reachability analysis for compositional linear hybrid systems, they need to compute the Cartesian Product of all the components at first which will cause state explosion very easily therefore make their performance even worse [8][9].

Recently, benefited from the progress of the boolean satisfiability problem (SAT), bounded model checking (BMC) [3] has been proposed as an alternative technique for BDD-based model checking as it is able to find witnesses in situations where other techniques fail completely. As a successful extension of SAT, Satisfiability Modulo Theories (SMT) has been widely applied to lots of real cases, including linear hybrid automata [4][5]. However, when encoding a compositional system, classical SMT-style BMC method [4][5] has to keep a global bound and a global step for all component automata. This always results in "stutter transitions" in the encoding when a component automaton does nothing in the corresponding step. It will cause transition interleaving and result in a huge problem space for the solver to solve when the system size, e.g., the number of component automaton, is large. This limitation restricts the size of the problem that can be solved.

In our previous study [6], we proposed a prototype tool BACH to do bounded reachability checking of a single linear hybrid automata. It traverses the structure of a linear hybrid automata using depth first search and verifies the abstract path by linear programming (LP). In this paper, we propose a new tool – BACH 2– for bounded reachability analysis of compositional LHA systems. Different from BACH, BACH 2 introduces a new method to encode the synchronization of automata and uses an efficient way to explore the state space. The main contributions of BACH 2 are given below:

- A new encoding is introduced for checking the path-oriented reachability problem of a path set [7]. Each path set consists of one path for each component in the compositional LHA system. In our encoding, each path in the path set is encoded to a group of linear constraints independently and a special set of constraints is added for synchronization control. Based on this encoding, the path-oriented reachability problem of the path set can be verified by solving the feasibility of the constraint set using linear programming very efficiently, without using neither Cartesian Product in general reachability analysis techniques, nor "stutter transitions" in SMT encoding for bounded reachability analysis.
- A tailored depth first search algorithm, named as SLS-DFS, is designed. It drops many path sets where sequences obtained by projecting each path on shared labels are not consistent with each other, and thus greatly reduce the number of path sets for checking. Driven by this algorithm, all potential path sets within a bound limit are picked out and checked one by one. Thus, we can tackle the problem of bounded reachability analysis of compositional linear hybrid system.

The rest of the paper is organized as follows: Sec. II gives a simple description of the underlying techniques of BACH 2. Sec. III describes several case studies to show the performance

978-3-9810801-6-2/DATE10 © 2010 EDAA

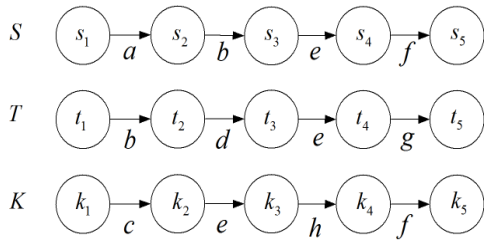

Fig. 1. Sample Automata

of our tool. Finally the conclusion is stated in Sec. IV.

II. The Underlying Techniques

A. Novelty of BACH 2

Path-Oriented Encoding. As mentioned in Sec. I, the main challenges for the current techniques for reachability analysis of compositional LHA systems are the state explosion caused by Cartesian Product in general techniques and the transition interleaving caused by stutter transitions in SMT-based BMC encoding. Instead of using the above two methods, in BACH 2 we implement our path-oriented reachability analysis method [7] which encodes each path in a path set separately and controls the synchronization by adding a special group of constraints. In this manner, we can solve the bounded reachability problem of the compositional LHA system by solving the feasibility of the small group of linear constraints and avoid the state explosion problem effectively. Here, we'll use a simple system given in Fig. 1 to illustrate the Cartesian Product-based general reachability analysis, the SMT-based BMC encoding and our path-oriented encoding introduced in BACH 2. This system consists of three subsystems: S, T, and K. These three subsystems synchronize with each other by shared labels b, e and f.

- In the traditional analysis procedure, most pieces of model checking work related to compositional LHA analysis usually combine all the participant automata together by Cartesian Product, which causes a critical problem of state explosion, thus greatly restricts the problem size, namely the number of participant automata in this context. Even if these three subsystems shown in Fig. 1 are all very simple, the size of the resulting automata is still quite large as we can see from Fig.2.A. This resulting system has 14 states and 12 paths to traverse if we want to check it.
- In SMT-based BMC encoding, the behavior of the system will be unrolled for k times. As the encoding keeps a global step for the whole system, it will introduce "stutter transitions" in the encoding when a component automaton does nothing but waits for synchronization as shown by the dashed arrows in Fig.2.B. Each dashed arrow stands for firing a stutter transition for k' times ($0 \leq k' \leq k$). As the exact number of stutter transitions can't be decided in advance, in general such a transition has to be encoded for k times, which introduces transition interleaving in the state space of the target SMT problem. When the size of the system is large, this will bring many variables and constraints into the target SMT problem that need to be solved, hence restrict the size of the problem that can be analyzed.
- While using our path-oriented approach, we can encode this system to a linear constraint set in the following way:
  1) For each path, we abstract the timed transition in each location using a nonnegative real variable which stands for the time spent on the location. For example, we use variable δ_{s_1} to indicate the time that the system S stayed in location s_1.
  2) Generate a set of constraints to guarantee all the system variables still satisfy all the location invariants and transition guards of each path after the system spent some amount of time in each location.
  3) Control the synchronization of the behavior of each component by adding constraints according to the time when certain transitions with shared labels are fired. For example, b is a label shared by S and T, then we have constraint $\delta_{s_1} + \delta_{s_2} = \delta_{t_1}$. In such a manner, we can ensure these three components cooperate accurately according to the synchronization events since they fire the transition with the same label at the same time spot, which are represented by the dashed lines in Fig.2.C.

  In general, this group of constraints generated in step 2 and step 3 with respect to the 3 paths in Fig.2.C can cover all the legal behaviors according to the 12 paths in Fig.2.A and all the possible interleaving in Fig.2.B. Thus, the compositional reachability problem of these three subsystems is transformed to the feasibility problem of a small group of linear constraints, which can be solved by linear programming efficiently.

SLS-DFS. As we already have an efficient approach to verify the reachability of a path set, if all the path sets of the system within a bound limit can be checked one by one, the bounded reachability problem of this system can be verified accordingly. However, when the number of component automata increases, the number of path sets could blow up quickly, which causes the enumeration of path by plain DFS doesn't work. In order to alleviate this composition explosion, we propose a novel shared label sequence guided depth first search method (SLS-DFS) to decrease the number of path sets needed to be checked.

The main idea of this DFS method is, given an arbitrary path set, if the projection of each path on their shared labels, denoted as shared label sequence (SLS), are not consistent with each other, this path set cannot compose a legal behavior of the whole system. Take the compositional system in Fig. 1 for example, given path set $\rho^* = \{\rho_S = \langle s_1\rangle \xrightarrow[a]{} \langle s_2\rangle \xrightarrow[b]{} \langle s_3\rangle$, $\rho_T = \langle t_1\rangle \xrightarrow[b]{} \langle t_2\rangle \xrightarrow[d]{} \langle t_3\rangle \xrightarrow[e]{} \langle t_4\rangle$, $\rho_K = \langle k_1\rangle \xrightarrow[c]{} \langle k_2\rangle \xrightarrow[e]{} \langle k_3\rangle\}$, as S communicates with T by shared labels b and e, the SLS of ρ_S is b, while the SLS of ρ_T is $b \rightarrow e$. These two SLSs are not consistent with each other, so the path set ρ^* is unreachable for sure. According to this rule, we can decrease the number

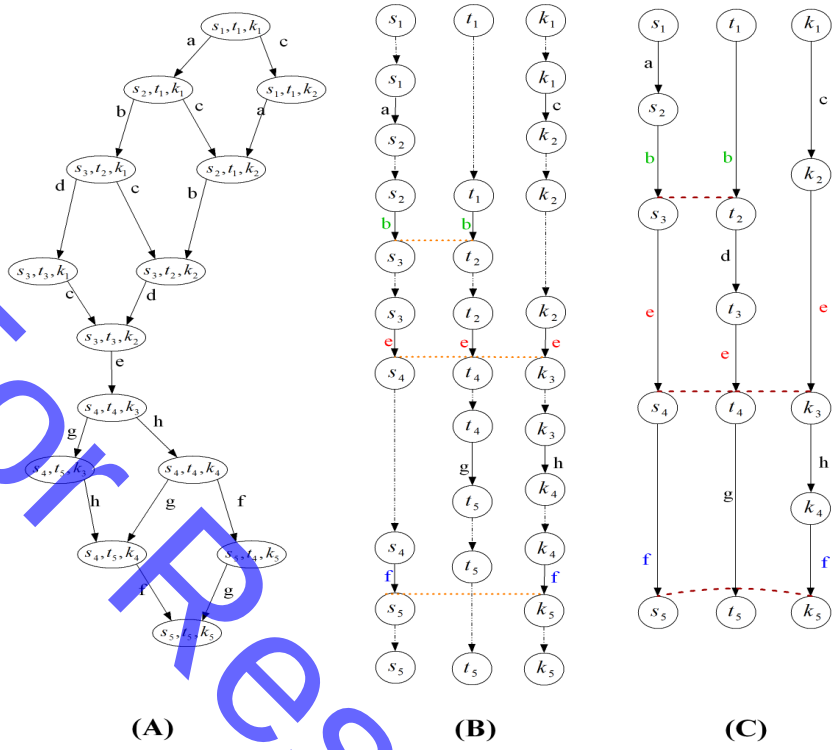

Fig. 2. Compositional State Space Interpretations for Sample Automata

of path sets which needed to be checked enormously.

B. Scenario Illustration

The main novelties of BACH 2 are explained in Sec.II-A already. Now, we use one real case scenario called TRAIN–GATE CONTROLLER (TGC) from study [2] to demonstrate the main underlying techniques of BACH 2. This system is composed of three components: TRAIN, GATE and CONTROLLER as shown in Fig. 3 (hereafter T, G and C). Each component has certain local time constraints that must be satisfied during the system execution. The synchronization between component automata is controlled by shared labels. For example, LHA T communicates with LHA C by two shared labels *approach* and *exit*.

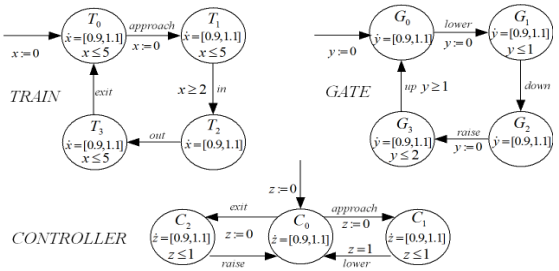

Fig. 3. Train-gate controller

Path-Oriented Reachability Analysis. In order to illustrate our approach succinctly, the path we choose for each component visits each transition only once, as indicated in Fig. 4.

$$\text{T:}\quad \langle T_0\rangle \xrightarrow[approach]{} \langle T_1\rangle \xrightarrow[in]{} \langle T_2\rangle \xrightarrow[out]{} \langle T_3\rangle \xrightarrow[exit]{} \langle T_0\rangle$$

$$\text{G:}\quad \langle G_0\rangle \xrightarrow[lower]{} \langle G_1\rangle \xrightarrow[down]{} \langle G_2\rangle \xrightarrow[raise]{} \langle G_3\rangle \xrightarrow[up]{} \langle G_0\rangle$$

$$\text{C:}\quad \langle C_0\rangle \xrightarrow[approach]{} \langle C_1\rangle \xrightarrow[lower]{} \langle C_0\rangle \xrightarrow[exit]{} \langle C_2\rangle \xrightarrow[raise]{} \langle C_0\rangle$$

Fig. 4. Path example of Train-gate controller Model

The problem we are concerned with is to check the reachability along these three paths, which can be reduced to a linear program whose linear inequality set is defined as follows:

1) Abstract the timed transition in each location by a nonnegative real variable δ_Y^X and represent the behavior of each path in the form of Fig. 5, where δ_Y^X stands for the time spent in the Yth location of LHA X.

$$\text{T:}\quad \left\langle \begin{matrix} T_0 \\ \delta_1^T \end{matrix} \right\rangle \xrightarrow[approach]{} \left\langle \begin{matrix} T_1 \\ \delta_2^T \end{matrix} \right\rangle \xrightarrow[in]{} \left\langle \begin{matrix} T_2 \\ \delta_3^T \end{matrix} \right\rangle \xrightarrow[out]{} \left\langle \begin{matrix} T_3 \\ \delta_4^T \end{matrix} \right\rangle \xrightarrow[exit]{} \left\langle \begin{matrix} T_0 \\ \delta_5^T \end{matrix} \right\rangle$$

$$\text{G:}\quad \left\langle \begin{matrix} G_0 \\ \delta_1^G \end{matrix} \right\rangle \xrightarrow[lower]{} \left\langle \begin{matrix} G_1 \\ \delta_2^G \end{matrix} \right\rangle \xrightarrow[down]{} \left\langle \begin{matrix} G_2 \\ \delta_3^G \end{matrix} \right\rangle \xrightarrow[raise]{} \left\langle \begin{matrix} G_3 \\ \delta_4^G \end{matrix} \right\rangle \xrightarrow[up]{} \left\langle \begin{matrix} G_0 \\ \delta_5^G \end{matrix} \right\rangle$$

$$\text{C:}\quad \left\langle \begin{matrix} C_0 \\ \delta_1^C \end{matrix} \right\rangle \xrightarrow[approach]{} \left\langle \begin{matrix} C_1 \\ \delta_2^C \end{matrix} \right\rangle \xrightarrow[lower]{} \left\langle \begin{matrix} C_0 \\ \delta_3^C \end{matrix} \right\rangle \xrightarrow[exit]{} \left\langle \begin{matrix} C_2 \\ \delta_4^C \end{matrix} \right\rangle \xrightarrow[raise]{} \left\langle \begin{matrix} C_0 \\ \delta_5^C \end{matrix} \right\rangle$$

Fig. 5. Behavior example of Train-gate controller Model

2) For each local constraint, generate the corresponding

linear constraints as follows: take the constraint $x > 2$ in the transition *in* of T for example, as the flow condition of x in T_1 is $[0.9, 1.1]$ and x is reset to 0 in the transition *approach*, we can generate the following constraints $x_{T_1} > 2$, $\delta_2^T \times 0.9 \leq x_{T_1} \leq \delta_2^T \times 1.1$, where x_{T_1} stands for the value of x after the LHA T stayed at state T_1 for time δ_2^T.

3) The sum of time spent on each path are required to be equal, e.g., for G and C, we can generate constraint: $\delta_1^G + \delta_2^G + \delta_3^G + \delta_4^G + \delta_5^G = \delta_1^C + \delta_2^C + \delta_3^C + \delta_4^C + \delta_5^C$.
4) For each shared label, we will also generate the constraints, e.g., for *exit* in T and C, we can generate constraint: $\delta_1^T + \delta_2^T + \delta_3^T + \delta_4^T = \delta_1^C + \delta_2^C + \delta_3^C$.

This is a simple illustration of how our solution can be used, and this example is transformed into a LP problem with 68 constraints and 22 variables which can be solved by an off-the-shelf LP solver in milliseconds.

Bounded Reachability Analysis. Here we still use the TGC model to illustrate our bounded reachability method, especially the SLS-DFS method, as follows. The reachability target under verification is whether the location (T_0, G_0, C_0) can be reached within the bound limit $\{[2,5],[2,5],[2,5]\}$ for the three components, where the bound means the number of discrete locations in the path.

1) Generate the shared label set $\{approach, exit, lower, raise\}$. Generate the initial empty path for each component respectively: ρ_T, ρ_G, ρ_C and the path set $\rho^* = \{\rho_T, \rho_G, \rho_C\}$. Generate and initialize the SLS set ζ^* to be empty.
2) Using plain DFS, find the next ρ_T in LHA T, denoted as ρ'_T, which can reach T_0 within bound [2,5]. The first path can be found is $\langle T_0\rangle \underset{approach}{\longrightarrow} \langle T_1\rangle \underset{in}{\longrightarrow} \langle T_2\rangle \underset{out}{\longrightarrow} \langle T_3\rangle \underset{exit}{\longrightarrow} \langle T_0\rangle$. Update ρ_T and ρ^* with ρ'_T.
3) Generate SLS ζ_T from $\rho_T : approach \rightarrow exit$, and add ζ_T to ζ^*.
4) Similarly, find path $\rho'_G = \langle G_0\rangle \underset{lower}{\longrightarrow} \langle G_1\rangle \underset{down}{\longrightarrow} \langle G_2\rangle \underset{raise}{\longrightarrow} \langle G_3\rangle \underset{up}{\longrightarrow} \langle G_0\rangle$ from LHA G. Update ρ_G and ρ^* with ρ'_G,
5) Get SLS ζ_G from $\rho_G : lower \rightarrow raise$, add ζ_G to ζ^*.
6) Find path $\rho'_C = \langle C_0\rangle \underset{approach}{\longrightarrow} \langle C_1\rangle \underset{lower}{\longrightarrow} \langle C_0\rangle \underset{exit}{\longrightarrow} \langle C_2\rangle \underset{raise}{\longrightarrow} \langle C_0\rangle$ from LHA C according to ζ^* and the reachability target. Update ρ_C and ρ^* with ρ'_C,
7) Since LHA C is the last component, check the reachability of ρ^* by the path-oriented method mentioned in the last paragraph.
8) If ρ^* is reachable, stop.
9) Otherwise, continue performing DFS on LHA C from ρ_C. As no more path can be found in this case, backtrack to LHA G to search for the next ρ_G, again no such path exists. Keep backtracking to LHA T, the next ρ_T can not be found neither. Since the first component has been backtracked, we stop and report that the bounded reachability target of the system can not be satisfied.

Note that, for either proving or disproving the given bounded reachability specification of this TGC system, only one path set needs to be generated and verified. This small example gives a simple but clear demonstration of how our approach can decrease the number of potential path sets which need to be generated. By taking advantage of our path-oriented reachability verification approach, we can traverse all potential path sets within the bound limit efficiently and finish the bounded reachability checking of compositional LHA systems without performing Cartesian Product or introducing stutter transition.

III. Case Studies

As a successor of BACH, BACH 2 is also implemented in Java, and can be downloaded from http://seg.nju.edu.cn/BACH/. BACH 2 shares the Graphical LHA Editor with BACH and extends two kinds of reachability verification: path-oriented and bounded reachability verification for compositional LHA systems. We briefly evaluate the performance of BACH 2 with two well-known examples: the Fischer Protocol and the Nuclear Reactor system. The experiments are conducted on a DELL workstation (Intel Core2 Quad CPU 2.4GHz, 4GB RAM).

The Fischer Protocol system consists of several competing processes which all attempt to enter the critical section. The automaton we use to model a single process is shown in Fig. 6. As this is a classical shared variable problem, in order to handle it in the context of our tool (synchronize with shared labels), we build a LHA: Shared Variable (SV) to represent all the evaluation and reset actions on the shared variable. The Nuclear Reactor System controls a nuclear reactor with n rods whose models are shown in Fig. 7. The system use these rods to absorb neutrons one by one. Each rod that has just been moved out of the heavy water must stay out of the water and cool for several time units. The size of these two examples can easily scale up by increasing the number of processes(rods), thus they are particularly appropriate to evaluate BACH 2.

For the path-oriented analysis part, we conduct a group of experiments based on these two systems with many components, i.e. 40 processes/rods. The path we select for each component is a long path which traverses all discrete locations in this component automaton several times. The experiment data are shown in Table I and Table II, e.g., for the fischer protocol system, the largest problem we solve is a system having 40 processes with each path traverses the structure of the system for 15 times. As the size of the linear program our approach generates is linear in the size of system (number of paths and locations in each path), this indicates a possibility of solving a system containing many more component automata with shorter path for each component.

We also evaluate the processing ability of the Bounded Reachability Checking (BRC) in BACH 2 using the same models by comparing with a group of state-of-the-art SMT solvers which support linear real arithmetic (LRA) and took part in the QF-LRA division of SMT-COMP'08. The solvers we compare with are MathSAT [10], Yices [11], CVC3 [12], and Barcelogic [13]. All the solvers we use in comparison

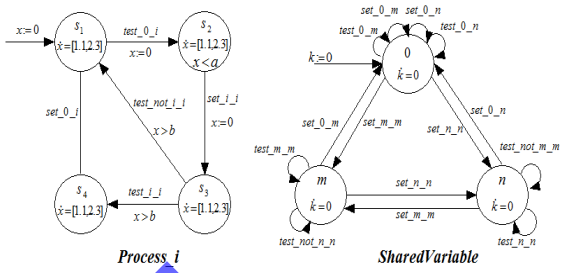

Fig. 6. Fischer Protocol

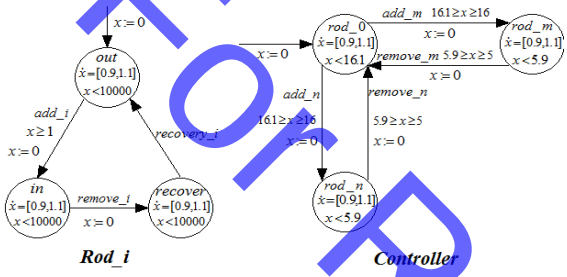

Fig. 7. Nuclear Reactor System

TABLE I
FISCHER PROTOCOL, 40 PROCESSES

Path	Pro_1	$(s_1\hat{}test\_0\_1\hat{}\ldots\hat{}s_4\hat{}set\_0\_1)^k\hat{}s_1$		
	Pro_2	$(s_1\hat{}test\_0\_2\hat{}\ldots\hat{}s_4\hat{}set\_0\_2)^k\hat{}s_1$		
	...	...		
	Pro_40	$(s_1\hat{}test\_0\_40\hat{}\ldots s_4\hat{}set\_0\_40)^k\hat{}s_1$		
	SV	$(0\hat{}test\_0\_1\hat{}\ldots\hat{}0\hat{}set\_0\_1)^k\hat{}0$		
k	Cons.	Vari.	Mem.(M)	Time(s)
5	4962	2041	256	160.661
10	9762	4041	1024	1233.208
15	14562	6041	2048	4099.463

TABLE II
NUCLEAR REACTOR SYSTEM, 40 RODS

Path	Rod_1	$(out\hat{}add\_1\hat{}\ldots\hat{}rec\hat{}rec\_1)^k\hat{}out$		
	Rod_2	$(out\hat{}add\_2\hat{}\ldots\hat{}rec\hat{}rec\_2)^k\hat{}out$		
	...	...		
	Rod_40	$(out\hat{}add\_40\hat{}\ldots\hat{}rec\hat{}rec\_40)^k\hat{}out$		
	Con.	$(rod\_0\hat{}add\_1\hat{}\ldots\hat{}rod\_40)^k\hat{}rod\_0$		
k	Cons.	Vari.	Mem.(M)	Time(s)
4	6246	1682	256	235.066
8	12166	3282	1024	1602.527
12	18086	4882	2048	5334.934

are in the latest version available online. In addition, we also compare our tool with a state-of-the-art SAT-style bounded model checker for hybrid automata HySAT [4]. In the experiments, the timeout was set to 24 hours, the SMT and HySAT encoding techniques are from study [4] and study [5]. For more information, our encoding of the examples, in the format of .ha, .hys, .msat and .smt, are all available on http://seg.nju.edu.cn/BACH/.

The reachability specification for the Fischer Protocol we want to check is whether the system can reach a state that all the processes are in the critical section. The reachability specification for Nuclear Reactor System we check is whether the system can reach a state in which all the rods are in the recover location. By adjusting parameters, we build two kinds of models for each system – unreachable models and reachable models. "Unreachable model" means the reachability specification cannot be satisfied by any path set in the model while "reachable model" means every potential path set in the model satisfy the reachability specification.

We conduct the experiments by setting the bound to the smallest number of the discrete locations for each component to reach the target. The experiment data on unreachable models are shown in Fig. 8 and Fig. 9, while the experiment data on reachable models are shown in Fig. 10 and Fig. 11. From these data, we can see that:

1) For unreachable models, we first consider the memory usage. The memory usage of SMT solvers blow up quickly with the increasing of system size. Several SMT solvers throw the memory allocation exception when tackling problem with around 10 components and more (> 3 GB), which means the size of system they can solve can not be larger even more time is given. However, the memory footprint of BACH 2 for the same problem is quite small (< 64 MB). We believe if time permits, BACH 2 can solve a larger problem when all the SMT solvers are failed owing to the memory explosion. On the other hand, from the aspect of time usage, as all the path sets within the bound limit don't satisfy the reachability specification, BACH 2 needs to exhaust all the path sets, which turns out to be time consuming. For example, on the nuclear reactor system, it took BACH 2 several hours to prove that the reachability specification can not be satisfied within the bound limit when solving problems with 10 rods. While with the help of the conflict clause learning method, Yices and Barcelogic successfully tackled the problem consisting of 11 rods. Note that, as the main issue of compositional analysis is the memory problem introduced by state explosion, and BACH 2 is not restricted by memory limitation, we believe the size of the problem that BACH 2 can solve will outperform SMT solvers in general.
2) For reachable models, BACH 2 only needs to find one path set which turn out to be reachable after path-oriented verification. From Fig. 10 and Fig. 11, we can see that BACH 2 can handle an extremely large system consisting of 320 processes/rods within 1 hour! At the same time, the performance of SMT solvers are restricted by the problem size. The largest problem SMT solvers solved in one day is a Nuclear Reactor System consisting of 18 rods which took Barcelogic 25648.9 seconds. Furthermore, similarly with the data on unreachable model, the memory usage for the same problem of BACH 2 (< 64 MB) is far less than SMT solvers (> 3 GB), which shows a clear potential of solving huge problems by BACH 2 in the future. Compared with data on unreachable models, the data on reachable models are more qualified to illustrate the advantage of BACH 2 in the context of BMC: the ability to find witnesses in situations where other techniques fail

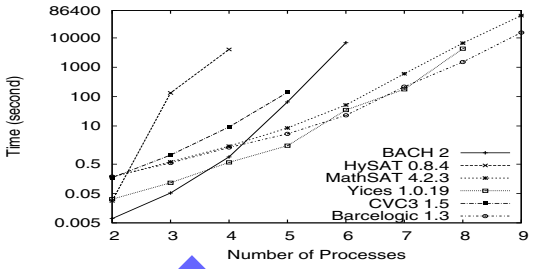

Fig. 8. BRC Results of Fischer Protocol on Unreachable Model

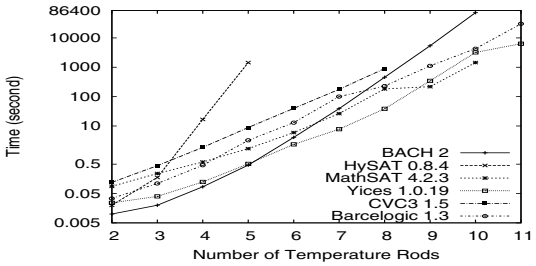

Fig. 9. BRC Results of Nuclear Reactor on Unreachable Model

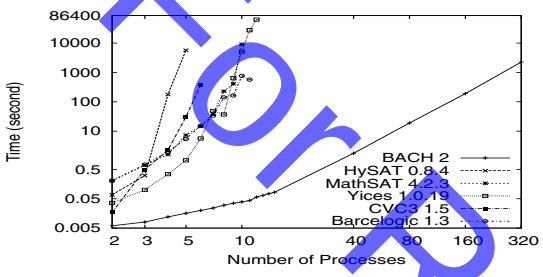

Fig. 10. BRC Results of Fischer Protocol on Reachable Model

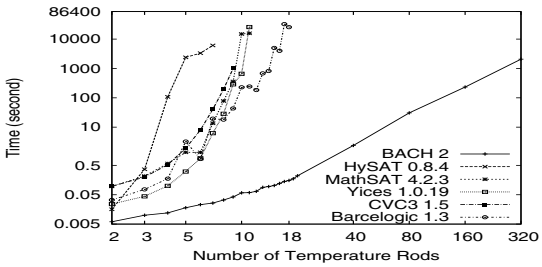

Fig. 11. BRC Results of Nuclear Reactor on Reachable Model

completely.

In summary, compared with SMT solvers, BACH 2 can take bounded reachability checking with much smaller memory usage and similar computation performance. To some extent, BACH 2 is immune to the memory exhaustion problem and thus can be used for extremely large systems.

IV. Conclusion

In this paper, we present a successor of our bounded reachability checker BACH for single LHA– BACH 2 for compositional LHA systems.

BACH 2 extends a powerful path-oriented reachability checker to analyze a path set in compositional LHA systems by encoding the path set to a set of linear constraints, then solving this constraint set by linear programming. BACH 2 also provides a bounded reachability checker using our SLS-DFS method to traverse all potential path sets within the bound limit and verify each of them using the path-oriented method to tackle the problem of bounded reachability analysis of compositional LHA system.

The experiments show that when solving problems of the same size, BACH 2 outperforms most SMT solvers significantly on the aspect of memory usage, at the same time maintaining the time usage on the same level, which benefits the verification of extremely large systems, e.g., a Fischer Protocol system consisting of 320 processes.

Acknowledgment

We would like to thank Dr. Alessandro Cimatti, Dr. Stefano Tonetta, and anonymous reviewers for their valuable comments and suggestions of this paper. This work is supported by the National Natural Science Foundation of China (No.90818022 and No.60721002), the National 863 High-Tech Programme of China (No.2009AA01Z148 and No.2007AA010302), and by the National Grand Fundamental Research 973 Program of China (No.2009CB320702).

References

[1] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of LICS 1996*, IEEE Computer Society, 1996, pp. 278-292.

[2] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H.Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine. The algorithmic analysis of hybrid systems. In *Theoretical Computer Science*, 138(1995), pp.3-34.

[3] A. Biere, A. Cimatti, E. Clarke, O. Strichman, Y. Zhu. Bounded Model Checking. In *Advance in Computers*, Vol.58, Academic Press, 2003, pp.118-149

[4] M. Fränzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. In *Journal on Satisfiability, Boolean Modeling and Computation*, 2007, vol.1, pp.209-236

[5] G. Audemard, M. Bozzano, A. Cimatti, R. Sebastiani. Verifying Industrial Hybrid Systems with MathSAT. In *Proceedings of the Second International Workshop on Bounded Model Checking (BMC04)*, Vol.119, No.2, 2005, pp.17-32.

[6] L. Bu, Y. Li, L. Wang and X. Li. BACH: Bounded Reachability Checker for Linear Hybrid Automata. In *Proceedings of the 8th International Conference on Formal Methods in Computer Aided Design*, IEEE Computer Society, pp.65-68,2008.

[7] L. Bu and X. Li. Path-Oriented Bounded Reachability Analysis of Compositional Linear Hybrid Systems. http://seg.nju.edu.cn/BACH/publications/prcsttt3.pdf, manuscript submitted, 2008.

[8] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: a model checker for hybrid systems. In *Software Tools for Technology Transfer*, 1:110-122, Springer, 1997.

[9] G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems past HyTech. In *Proceedings of International Conference on Hybrid Systems: Computation and Control '05*, LNCS 2289, pp.258-273.

[10] R. Bruttomesso, A. Cimatti, A. Franzen, A. Griggio, R. Sebastiani. The MathSAT 4 SMT Solver. In *Proceedings of International Conference on Computer Aided Verification 08*. LNCS 5123, pp.299-303.

[11] B. Dutertre and L. de Moura. The Yices SMT Solver, http://yices.csl.sri.com/tool-paper.pdf, 2006.

[12] C. Barrett and C. Tinelli. CVC3. In *Proceedings of International Conference on Computer Aided Verification 07*, LNCS 4590, pages 298-302. Springer, July 2007.

[13] M. Bofill, R. Nieuwenhuis, A. Oliveras, E. Carbonell, A. Rubio. The Barcelogic SMT Solver. In *Proceedings of International Conference on Computer Aided Verification 08*. LNCS 5123, 2008.