BACH: Path-oriented Reachability Checker of Linear Hybrid Automata

Xuandong Li
Department of Computer Science and Technology,
Nanjing University, P.R.China
Outline

- Preliminary Knowledge
- Path-oriented Reachability Checking
- IIS-Based Bounded Checking
- Shallow Semantic Based Compositional Checking
- Unbounded Proof Derivation
- Conclusion
Outline

- Preliminary Knowledge
- Path-oriented Reachability Checking
- IIS-Based Bounded Checking
- Shallow Semantic Based Compositional Checking
- Unbounded Proof Derivation
- Conclusion
Hybrid System

- Systems containing **both discrete and continuous components**

- Practical Examples:
  - Embedded System Controller
  - VLSI circuits
  - System Biology

- Safety Critical Area

- Formal Verification
  - Formal Model: Hybrid Automata
Hybrid Automata

- Widely studied formal models for hybrid systems
  \[ H = (X, \Sigma, V, E, V^0, \alpha, \beta, \gamma) \]

- They consist of
  - A finite state transition system
  - Differential equations in each location

Example

Linear Hybrid Automata
Reachability Analysis

- **Approach**
  - Over-approximation
  - Geometric Computation

- **Performance**
  - Undecidable
  - Imprecise
  - Low dimension
Reachability Analysis

**Bounded Model Checking**

- Search for a potential behavior within $k$ step
- Usually solved by SMT techniques
  - SMT: satisfiability modulo theories
- Need to encode all the potential bounded behavior firstly
- Medium bound $\rightarrow$ Large SMT problem

Control The Complexity!
Outline

- Preliminary Knowledge
- **Path-oriented Reachability Checking**
- IIS-Based Bounded Checking
- Shallow Semantic Based Compositional Checking
- Unbounded Proof Derivation
- Conclusion
Reachability Analysis

- Path-oriented Based Bounded Model Checking
  - Check the reachability of one abstract path using Linear Programming (LP)
  - Enumerate all the candidate paths in bound by Depth First Search (DFS)
Path, Behavior, Encoding

**Path:**

\[ \langle v_0 \rangle \xrightarrow{(\varphi_0, \psi_0)} \langle v_1 \rangle \xrightarrow{(\varphi_1, \psi_1)} \cdots \xrightarrow{(\varphi_{n-1}, \psi_{n-1})} \langle v_n \rangle \]

**Behavior:**

\[ \begin{bmatrix} v_0 \\ t_0 \end{bmatrix} \xrightarrow{(\varphi_0, \psi_0)} \begin{bmatrix} v_1 \\ t_1 \end{bmatrix} \xrightarrow{(\varphi_1, \psi_1)} \cdots \xrightarrow{(\varphi_{n-1}, \psi_{n-1})} \begin{bmatrix} v_n \\ t_n \end{bmatrix} \]

**Encoding**

\[ g_1(y) = \delta_1(x) = \gamma_1(x) \]

\[ \delta_2(x) = \gamma_2(x) \]

\[ \delta_3(x) = \gamma_3(x) \]

\[ (\varphi, \psi) = (\varphi, \psi) + t \]

\[ (x) + t (x) + t (x) + 2t \]

\[ t_0 \leq 0; \quad (y) < 5; \quad (y) < 5 \]
DFS-Based Bounded Model Checking

- Eager-DFS-BMC
  - check each path $\rho$ in the given bound
  - If $\rho$ is infeasible, backtrack to the last location

- BACH: $B$ounded reAChability $C$hecker

- [http://seg.nju.edu.cn/BACH/](http://seg.nju.edu.cn/BACH/)
Outline

- Preliminary Knowledge
- Path-oriented Reachability Checking
- IIS-Based Bounded Checking
- Shallow Semantic Based Compositional Checking
- Unbounded Proof Derivation
- Conclusion
Eager - DFS - BMC

- Eager - DFS - BMC
  - Check each path $\rho$ in the given bound
  - Lots of redundant work

Example

- Target $v_5$
  - $v_0 \rightarrow v_1$
  - $v_0 \rightarrow v_1 \rightarrow v_2$
  - $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3$

Most of the time are spent in LP solving
Lazy DFS + LP

- Lazy DFS + LP
  - Only check the path $\rho$ when it reaches the target

Where to backtrack?
Using IIS to locate infeasible path segment core to accelerate the backtracking

An irreducible infeasible set (IIS) of an infeasible linear constraint set is an unsatisfiable set of constraints that becomes satisfiable if any constraint is removed.

\{A, B, C\} is an IIS
**Recall**: We use an LP based approach to check the feasibility of a path $\rho$

- IIS technique can be used to locate the minimal inconsistent set
- Such inconsistent set can be mapped back to an path segment. All the paths containing such path segments are not feasible for sure.
Example

Example

- $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_1 \rightarrow v_5$
- $v_3 \rightarrow v_4 \rightarrow v_1 \rightarrow v_5$ is the IIS path segment
- Backtrack to $v_1$
- Once DFS found a new path containing $v_3 \rightarrow v_4 \rightarrow v_1 \rightarrow v_5$ it will backtrack to $v_1$ directly without call LP solver

Bound 100, Lazy DFS+IIS -> 25 paths only call LP solver 2 times

Problem:
These paths containing the IIS are not feasible for sure. Can we don’t waste time in enumerating such paths?
The transition relation graph can be encoded as propositional formulas

- Encode the bounded graph structure of an LHA into a propositional formula set
- Find a truth assignment using a SAT solver
  - SAT: Boolean satisfiability problem
- Decode the truth assignment to get a path in the graph
SAT Encoding of the Bounded Graph

- Consist of four clauses

\[ \text{NEXT} := \bigwedge_{q \in V} (\text{loc} = q \rightarrow \bigvee_{(q, q') \in N} \text{loc}' = q') \]

\[ \text{EXCLUDE} := \bigwedge_{q \in V} (\text{loc} = q \rightarrow \bigwedge_{q' \in V \land q' \neq q} \text{loc} \neq q') \]

\[ \text{INIT} := (\text{loc} = v_f) \land \text{EXCLUDE} \]

\[ \text{TARGET} := (\text{loc} = v_T) \]

- The bounded graph formula set with bound \( k \)

\[ \text{BG}^k := \text{INIT}^0 \land \bigwedge_{0 \leq i \leq k-1} \text{NEXT}^i \land \bigwedge_{1 \leq i \leq k} \text{EXCLUDE}^i \land \bigvee_{0 \leq i \leq k} \text{TARGET}^i \]
Decode From The Truth Assignment

- The superscript of the name of variables represents the order of the nodes in the path.
- Suppose we get a truth valuation: \(v_0^0, v_1^1, v_2^2\) from the SAT encoding, the corresponding path in the graph is \(\langle v_0 \rangle \xrightarrow{e_0} \langle v_1 \rangle \xrightarrow{e_5} \langle v_5 \rangle\).
Include a *IIS* clause to prevent the SAT from enumerating paths which contain an infeasible path segment.

\[
IIS := \bigwedge_{\rho' \in IIS\ Path} IIS^k(\rho')
\]

\[
BG^k := BG^k \land IIS
\]
Example

- The previous checked path

\[ \rho = \langle v_0 \rangle \xrightarrow{e_0} \langle v_1 \rangle \xrightarrow{e_1} \langle v_2 \rangle \xrightarrow{e_2} \langle v_3 \rangle \xrightarrow{e_3} \langle v_4 \rangle \xrightarrow{e_4} \langle v_1 \rangle \xrightarrow{e_5} \langle v_5 \rangle \]

- The infeasible path segment

\[ \rho' = \langle v_3 \rangle \xrightarrow{e_3} \langle v_4 \rangle \xrightarrow{e_4} \langle v_1 \rangle \xrightarrow{e_5} \langle v_5 \rangle \]

- The IIS clause

\[ IIS^k(\rho') := \bigwedge_{0 \leq i \leq k-leng+1} \left( v_i^i \land v_i^{i+1} \land v_i^{i+2} \rightarrow \neg v_i^{i+3} \right) \]
Example

Bound 100, $v_5$

DFS+IIS $\rightarrow$ 25 paths
(call LP 2 times)

SAT+IIS $\rightarrow$ 2 paths
Performance

Performance Data On The Highway System With 500 Vehicles

System Size: 502 locations, 500 variables

<table>
<thead>
<tr>
<th>Bound</th>
<th>Tech.</th>
<th>BACH-SAT</th>
<th>BACH-DFS</th>
<th>MathSAT</th>
<th>Z3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>Memory</td>
<td>Time</td>
<td>Memory</td>
</tr>
<tr>
<td>3</td>
<td>BACH-SAT</td>
<td>53.2s</td>
<td>&lt;1000m</td>
<td>12.3s</td>
<td>&lt;600m</td>
</tr>
<tr>
<td>100</td>
<td>BACH-DFS</td>
<td>62.2s</td>
<td>&lt;2500m</td>
<td>OOT</td>
<td>&lt;4096m</td>
</tr>
<tr>
<td>200</td>
<td>Z3</td>
<td>74.2s</td>
<td>&lt;4096m</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

- **Large Scale System** 500 locations, 500 variables
- Classical SMT-style BMC, **OOM** (Out of Memory) with bound 3
- **BACH**:  
  - Path-oriented, complexity well controlled  
  - With the help of IIS, 200 steps in only 74 seconds!

Scalable Highway System
Outline

- Preliminary Knowledge
- Path-oriented Reachability Checking
- IIS-Based Bounded Checking
- Shallow Semantic Based Compositional Checking
- Unbounded Proof Derivation
- Conclusion
Compositional LHA System

- Compositional LHA Systems

\[ S \ xrightarrow{a} s_1 \ xrightarrow{b} s_2 \ xrightarrow{e} s_3 \ xrightarrow{f} s_4 \ xrightarrow{} s_5 \]

\[ T \ xrightarrow{b} t_1 \ xrightarrow{d} t_2 \ xrightarrow{e} t_3 \ xrightarrow{g} t_4 \ xrightarrow{} t_5 \]

\[ K \ xrightarrow{c} k_1 \ xrightarrow{e} k_2 \ xrightarrow{h} k_3 \ xrightarrow{f} k_4 \ xrightarrow{} k_5 \]
Current Status

Low Dimension Few Components
Shallow Synchronization Semantic
Bounded Reachability

- Find and verify all the path sets in the given bound limit

- Reduce the number of potential path sets which needs to be verified.

- Share label sequence guided DFS
Share Label Sequence Guided DFS

\[
T_0 \xrightarrow{\text{approach}} T_1 \xrightarrow{\text{in}} T_2
\]

\[
G_0 \xrightarrow{\text{lower}} G_1 \xrightarrow{\text{down}} G_2
\]

\[
C_0 \xrightarrow{\text{approach}} C_1 \xrightarrow{\text{lower}} C_0 \xrightarrow{\text{exit}} C_2
\]

\[
\begin{align*}
T_0 & \quad x = [0.9, 1.1] \quad x \leq 5 \\
T_1 & \quad x = [0.9, 1.1] \quad x \leq 5 \\
T_2 & \quad x = [0.9, 1.1] \\
T_3 & \quad x = [0.9, 1.1] \\
G_0 & \quad y = [0.9, 1.1] \quad y \leq 1 \\
G_1 & \quad y = [0.9, 1.1] \\
G_2 & \quad y = [0.9, 1.1] \\
G_3 & \quad y = [0.9, 1.1] \\
C_0 & \quad z = [0.9, 1.1] \\
C_1 & \quad z = [0.9, 1.1] \\
C_2 & \quad z = [0.9, 1.1] \quad z \leq 1 \\
C_3 & \quad z = [0.9, 1.1] \quad z \leq 1
\end{align*}
\]
Performance
Outline

- Preliminary Knowledge
- Path-oriented Reachability Checking
- IIS-Based Bounded Checking
- Shallow Semantic Based Compositional Checking
- Unbounded Proof Derivation
- Conclusion
Previous Example

Water-Level Monitor System

- Is $v_5$ reachable within 10 steps?
- Path: $v_0 \rightarrow v_1 \rightarrow v_5$
- IIS: $v_0 \rightarrow v_1 \rightarrow v_5$
- Path: $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_1 \rightarrow v_5$
- IIS: $v_3 \rightarrow v_4 \rightarrow v_1 \rightarrow v_5$

Potential path can not contain

$v_3 \rightarrow v_4 \rightarrow v_1 \rightarrow v_5$
$v_0 \rightarrow v_1 \rightarrow v_5$

No more potential paths, not reachable!
Key insights

Avoiding IIS path segments may make the target location not reachable in the unbounded state space.

Goal

- Prove whether there exists a path which can reach the target location without touching certain path segments.

Solution

- LTL model checking
  - LTL: linear temporal logic
We propose to model the graph structure of an LHA with a finite-state transition system (TS).
Avoid containing an IIS path segment

- Suppose there is an IIS path segment:
  \[ \rho' = v_i \rightarrow v_{i+1} \rightarrow \ldots \rightarrow v_j \]
  \[ p_{v_i} p_{v_{i+1}} \ldots p_{v_j} \]

- The LTL formula which can represent \( \rho' \):
  \[ IIS_{\rho'} = p_{v_i} \& X p_{v_{i+1}} \& \ldots \& X X \ldots X p_{v_j} \]

- A path which does not contain \( \rho' \):
  \[ G(\neg IIS_{\rho'}) \]
Reach target without any IIS path segment

- The target location $q_{bad}$ is finally reached:

  $v_i v_{i+1} \cdots q_{bad}$

  $p_{v_i} p_{v_{i+1}} \cdots p_{q_{bad}} \rightarrow F p_{q_{bad}}$

- The LTL formula which is true for path reaching the target without containing any IIS path segment

  \[
  \{\rho_1, \rho_2, \cdots, \rho_n\} : (G( \bigwedge_{1 \leq i \leq n} \neg \text{IIS}_{\rho_i})) \land F p_{q_{bad}}
  \]

- As our target is to prove the nonexistence of such a path, the final LTL specification:

  $\neg((G( \bigwedge_{1 \leq i \leq n} \neg \text{IIS}_{\rho_i})) \land F p_{q_{bad}})$
Workflow of Unbounded Proof Derivation

- LHA model
- BMC Procedure
- Reachable?
- LTL Formula Encoding
- LTL Specification Checking
- Satisfiable?
- Report Generally Not Reachable
- Report k-Bounded Not Reachable
- Report Reachable
## Experiment

<table>
<thead>
<tr>
<th>System</th>
<th>#locs</th>
<th>#vars</th>
<th>#IIS</th>
<th>BACH (NuSMV)</th>
<th>BACH (IC3)</th>
<th>SpaceEx (PHA.)</th>
<th>SpaceEx (Supp.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Time (s)</td>
<td>Mem. (MB)</td>
<td>Time (s)</td>
<td>Mem. (MB)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>water</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>0.94_U</td>
<td>&lt;1</td>
<td>0.87_U</td>
<td>30.4</td>
</tr>
<tr>
<td>tcs</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>0.97_U</td>
<td>&lt;1</td>
<td>0.98_U</td>
<td>16.4</td>
</tr>
<tr>
<td>sample</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0.96_B</td>
<td>26.8</td>
<td>0.41_B</td>
<td>21.2</td>
</tr>
<tr>
<td>train</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>1.02_U</td>
<td>&lt;1</td>
<td>0.3_U</td>
<td>&lt;1</td>
</tr>
<tr>
<td>motorcade_5</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>0.05_U</td>
<td>&lt;1</td>
<td>0.4_U</td>
<td>&lt;1</td>
</tr>
<tr>
<td>motorcade_10</td>
<td>12</td>
<td>10</td>
<td>9</td>
<td>0.12_U</td>
<td>&lt;1</td>
<td>0.6_U</td>
<td>16.9</td>
</tr>
<tr>
<td>motorcade_20</td>
<td>22</td>
<td>20</td>
<td>19</td>
<td>0.53_U</td>
<td>60.8</td>
<td>1.1_U</td>
<td>25.4</td>
</tr>
<tr>
<td>motorcade_100</td>
<td>102</td>
<td>100</td>
<td>99</td>
<td>6.66_U</td>
<td>163.9</td>
<td>15.7_U</td>
<td>389</td>
</tr>
<tr>
<td>motorcade_200</td>
<td>202</td>
<td>200</td>
<td>199</td>
<td>61.8_U</td>
<td>652.7</td>
<td>115.3_U</td>
<td>3299</td>
</tr>
</tbody>
</table>

Try the task of unbounded proof by the cost of BMC!
Outline

- Preliminary Knowledge
- Path-oriented Reachability Checking
- IIS-Based Bounded Checking
- Shallow Semantic Based Compositional Checking
- Unbounded Proof Derivation
- Conclusion
Framework

- Path oriented 
- Bounded
- Single HA 
- Composed HA
- Linear HA 
- Nonlinear HA
- Hybrid System 
- Cyber Physical
Current Achievement

- **Tool: BACH**
  - Graphical Editor, Model Checker, Eclipse Plugin, Web Application… more than 8 components and 20 versions
  - More than 200 Globally Download, including researchers from UCB, CMU, UBC and engineers from industry.
  - BMC Area Chair of ARCH Competition 2017, 2018

- **Publications**
  - Around 40 papers: IEEE TC, IEEE TPDS, ACM TCPS, FMSD, STTT, RTSS, CAV, FMCAD, DSN, ICCPS, DATE, VMCAI, FORTE and so on
  - 11 Software Copyrights, 8 Chinese Patents
Selected Application: CPS

- Real-life CPS show high nondeterministic behavior
  ➔ classical offline model checking does not work
- Our solution:
  - Parametric hybrid system modeling, Online Concretization
  - Online periodical real-time hybrid systems model checking of time-bounded future!
- Implemented a special version BACH_{OL} for CPS online verification
- Deployed on National Engineering Research Center of Rail Transportation Operation and Control System
Selected Application: IoT

- IFTTT-style event triggering IoT system is widely believed to be an important enabling building block of IoT
- Will an IoT app meet an user’s expectations? Will there be any unsafe consequences?
- We propose a framework of Modeling, Verification and Fixing of Smart Home System as Real time hybrid system automatically
- BACH is the underlying checker
- Selected into Microsoft TechFest’15 for technology transfer
Conclusion

- By isolating the discrete path and related continuous behavior into different layers, the **complexity** of our approach is well-controlled.

- By integrating SAT, LP and IIS, the **performance** of our tool outperforms the state-of-the-art SMT solvers significantly.

- Use the byproduct of BMC, IIS, to derive an unbounded result (**Extra Benefit!**)

- **On going work:** **Code Verification**
  - Software code shares similar feature with hybrid system
    - Transition system with constraints, infinite state space…

- **Public available** from http://seg.nju.edu.cn/BACH/
Thanks

Questions?