Parallel Reachability Analysis for Hybrid Systems

Amit Gurung*

Joint work with

Arup Deka*, Ezio Bartocci**, Sergiy Bogomolov***, @, Radu Grosu**

and

Rajarshi Ray*

*(National Institute of Technology Meghalaya, India)

**(Vienna University of Technology, Austria)

*** (Institute of Science and Technology, Austria)

@(Australian National University, Australia)

MeMoCoDe, Nov 18-20, 2016
Overview

1 Motivation of Parallel Reachability Analysis

2 Sequential BFS Reachability Algorithm on Hybrid Systems

3 Parallel BFS Algorithms on Hybrid Systems
   - Adapted Gerard J Holzmann’s PBFS Algorithm
   - Load Balancing TP-BFS Algorithm

4 Conclusion
Motivation of Parallel Reachability Analysis

- Reachability analysis is the process of computing all reachable states from a given set of initial states.
- It helps in analysing the behaviour (e.g. safety) of systems.
- **Our Focus**: Hybrid Systems where the continuous component has linear dynamics of the form
  \[
  \dot{x} = Ax(t) + u(t), \quad u(t) \in U, \quad x(0) \in X_0
  \]

**Figure**: Reachable state-space of a 28 dimensional Helicopter Controller System generated using the tool SpaceEx.

- Computation of reachable states with a reasonable precision is an expensive operation even in the state-of-the-art tool.

- **Goal**: exploit the computational power of multi-core architectures
Symbolic State: reachable states are represented using Symbolic State so as to enable operations such as union, difference, intersection, emptiness check etc.

For eg, $s = (D, C)$ is a symbolic state. $D \subseteq \text{Loc}$ set of discrete states and $C \subseteq \mathbb{R}^n$ a set of continuous states

$PostC(s)$: the set of states reachable from a given initial symbolic state $s$, by letting an arbitrary amount of time elapse (also referred as flowpipe)

$PostD(f)$: generates a new list of symbolic states (to be explored or reachable) on the application of discrete transitions on the flowpipe $f$. 
Data structure

- $Wlist$ consists a list of symbolic states to be processed at next iterations.
- $R$ is the union of all reachability region explored.

---

**Algorithm 1 Reachability Algorithm for Hybrid Automata**

1: procedure REACH-HA(ha, Init)
2:     $Wlist \leftarrow $ Init; $R \leftarrow \emptyset$;
3:     while $Wlist \neq \emptyset$ do
4:         delete $S$ from $Wlist$; $R' \leftarrow PostC(S)$
5:         $R \leftarrow R \cup R'$
6:         $R'' \leftarrow PostD(R')$
7:         if $R'' \subseteq R$ then go to step 3
8:         else add $R''/R$ to $Wlist$
9:     end if
10:    end while
11: end procedure
Parallel BFS Algorithms
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

<table>
<thead>
<tr>
<th>Wlist[t] :- read</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4</td>
</tr>
<tr>
<td>1 2 3 4</td>
</tr>
<tr>
<td>1 2 3 4</td>
</tr>
<tr>
<td>1 2 3 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Wlist[1-t] :- write</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4</td>
</tr>
<tr>
<td>1 2 3 4</td>
</tr>
<tr>
<td>1 2 3 4</td>
</tr>
<tr>
<td>1 2 3 4</td>
</tr>
</tbody>
</table>

1. Central idea uses data structure `Wlist` as 3 dimensional array as `Wlist[2][N][N]`. `N` is the number of cores available.

2. 1st dimension is for separate read/write index for a lock-free operation.

3. `N × N` matrix is for distributing randomly the symbolic states for proper load balancing.
Initial state of the \textit{wlist} with only one symbolic state to explore (with $t=0$)

\[
\begin{align*}
\text{Wlist}[t] &: \text{- read} \\
\text{Wlist}[1-t] &: \text{- write}
\end{align*}
\]

\begin{tabular}{|c|c|c|c|}
\hline
1 & 2 & 3 & 4 \\
\hline
1 & & & & \\
\hline
2 & & & & \\
\hline
3 & & & & \\
\hline
4 & & & & \\
\hline
\end{tabular}

\begin{tabular}{|c|c|c|c|}
\hline
1 & 2 & 3 & 4 \\
\hline
1 & & & & \\
\hline
2 & & & & \\
\hline
3 & & & & \\
\hline
4 & & & & \\
\hline
\end{tabular}
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

![Diagram of Wlist[t] and Wlist[1-t]]

- **Wlist[t] :- read**

- **Wlist[1-t] :- write**
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

\[ Wlist[t] :- read \]

\[ Wlist[1-t] :- write \]

Amit Gurung (NIT Meghalaya)
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

Now switch the read/write variable as: \( t = 1 - t \)

---

Amit Gurung (NIT Meghalaya)
Now the write data structure will become the read and vice versa.

\[
\text{Wlist}[t] : \text{read} \\
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & & & \\
2 & & & \\
3 & & & \\
4 & & & \\
\end{array}
\]

\[
\text{Wlist}[1-t] : \text{write} \\
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & & & \\
2 & & & \\
3 & & & \\
4 & & & \\
\end{array}
\]
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

\[
\begin{align*}
Wlist[t] & :- \text{read} \\
1 & \enspace 2 & \enspace 3 & \enspace 4 \\
1 & & & \\
2 & & & \\
3 & & & \\
4 & & & \\
\end{align*}
\]

\[
\begin{align*}
Wlist[1-t] & :- \text{write} \\
1 & \enspace 2 & \enspace 3 & \enspace 4 \\
1 & & & \\
2 & & & \\
3 & & & \\
4 & & & \\
\end{align*}
\]
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

\[
\begin{array}{c|c|c|c|c}
 & 1 & 2 & 3 & 4 \\
\hline
1 & & & & \\
2 & & & & \\
3 & & & & \\
4 & & & & \\
\end{array}
\]

Wlist[t] :- read

1
2
3
4

Wlist[1-t] :- write

1
2
3
4

Amit Gurung (NIT Meghalaya)

XSpeed: Parallel Reachability Analysis Tool

MeMoCoDe Nov 18-20, 2016
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

Wlist[t] :- read

1 2 3 4
1 2 3 4

Wlist[1-t] :- write

1 2 3 4
4

Amit Gurung (NIT Meghalaya)
XSpeed: Parallel Reachability Analysis Tool
MeMoCoDe Nov 18-20, 2016
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

- **Wlist[t] :- read**
- **Wlist[1-t] :- write**

Amit Gurung (NIT Meghalaya)
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

\[ Wlist[t] : \text{read} \]

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & & & \\
2 & & & \\
3 & & & \\
4 & & & \\
\end{array}
\]

\[ Wlist[1-t] : \text{write} \]

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & & & \\
2 & & & \\
3 & & & \\
4 & & & \\
\end{array}
\]
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

Wlist[t] :- read

Wlist[1-t] :- write
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

Wlist[t] :- read

Wlist[1-t] :- write
Now we switch again the read/write variable as: $t = 1 - t$
1) Adaptation of Parallel-BFS by Gerard J. Holzmann on Hybrid Systems

- Status of the $Wlist$ data-structure after exploration of 2 levels during BFS
- Each core/thread have at least one symbolic state to process (and 2 for thread id=1).

\[
\begin{array}{cccc}
Wlist[t] & \text{read} & \text{wlist}[1-t] & \text{write} \\
1 & 2 & 3 & 4 \\
1 & \bullet & \bullet & \bullet \\
2 & \bullet & \bullet &  \\
3 & \bullet &  &  \\
4 &  &  &  \\
\end{array}
\]
Comparison of Adapted Gerard J. Holzmann’s algorithm on Hybrid Systems

- SpaceEx (LGG): implementation of the LeGuernic-Girard (LGG) algorithm in the tool SpaceEx
- XSpeed (Seq-BFS): Sequential implementation of BFS algorithm in the tool XSpeed
- XSpeed (GJH-PBFS): Adaptation of Holzmann’s PBFS algorithm implemented in the tool XSpeed

<table>
<thead>
<tr>
<th>Models</th>
<th>Breadths</th>
<th>Time in Secs</th>
<th>CPU Utilization (%)</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SpaceEx (LGG)</td>
<td>XSpeed (Seq-BFS)</td>
<td>XSpeed (GJH-PBFS)</td>
</tr>
<tr>
<td>Circle</td>
<td>4</td>
<td>37.68</td>
<td>29.39</td>
<td>33.92</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>86.26</td>
<td>80.47</td>
<td>62.80</td>
</tr>
<tr>
<td>Nav 3x3 (inst# 1)</td>
<td>3</td>
<td>61.42</td>
<td>9.17</td>
<td>5.72</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>199.27</td>
<td>44.37</td>
<td>24.24</td>
</tr>
<tr>
<td>Nav 5x5</td>
<td>5</td>
<td>215.46</td>
<td>50.77</td>
<td>31.39</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>1232.31</td>
<td>253.63</td>
<td>104.30</td>
</tr>
</tbody>
</table>

Amit Gurung (NIT Meghalaya)

XSpeed: Parallel Reachability Analysis Tool
Load Balancing Problem Identified on some Benchmarks of Hybrid Systems

Observation

- Non-uniform flowpipe computation cost give rise to improper load balancing and under utilization of CPU.

Figure: Flowpipe after 2 levels of BFS

Figure: Flowpipe after 3 levels of BFS
Load Balancing Problem Identified on some Benchmarks of Hybrid Systems

Observation

- Non-uniform flowpipe computation cost give rise to improper load balancing and under utilization of CPU.

![Flowpipe after 2 levels of BFS](image1)

![Flowpipe after 3 levels of BFS](image2)

Proposed Solution: All flowpipes computation can be grouped as one big task (which is a collection of large number of **atomic tasks**).
Load Balancing Problem Identified on some Benchmarks of Hybrid Systems

Observation
- Non-uniform flowpipe computation cost give rise to improper load balancing and under utilization of CPU.

Figure: Flowpipe after 2 levels of BFS

Figure: Flowpipe after 3 levels of BFS

Proposed Solution: All flowpipes computation can be grouped as one big task (which is a collection of large number of **atomic tasks**)

Then it can be distributed evenly on to the available cores.
Support Functions as Set Representation

- The support function of any compact convex set \( S \subseteq \mathbb{R}^d \), denoted by \( \rho_S : \mathbb{R}^d \rightarrow \mathbb{R} \) defined as
  \[
  \rho_S(\ell) = \max_{x \in S} (\ell \cdot x)
  \]
  (1)

- A compact convex set \( S \) can be represented by its support function as:
  \[
  S = \bigcap_{\ell \in \mathbb{R}^n} (\ell \cdot x) \leq \rho_S(\ell)
  \]
  (2)
Polyhedral approximations are computed from the support function representation using finite samplings in bounding directions.

**Observation:**

- Flowpipe computation involves a collection of support function sampling.
- Each support function sampling could be computed independent of each other and can be considered as an atomic task.
2) Load Balancing Algorithm (TP-BFS)

- A simplified version of task-parallel BFS (TP-BFS) algorithm.
- Steps involved in computing the PostC and PostD operations in the Load balancing algorithm for an $i^{th}$-BFS level.
2) Load Balancing Algorithm (TP-BFS)

- **Wlist[t]**: Flowpipe cost Computation
- **Atomic tasks**: Task distribution for parallel execution
- **Merge result to create Flowpipe**

**i\textsuperscript{th} BFS iteration**

- **Wlist[1-t]**: Generate symb states
- **Task distribution for parallel execution**
- **Atomic tasks**
- **PostD cost Computation**

**PostC**

**PostD**

**(i+1)\textsuperscript{th} BFS iteration**

- **Wlist[t]**: Flowpipe cost Computation
- **Atomic tasks**: Task distribution for parallel execution
- **Merge result to create Flowpipe**

- **Wlist[1-t]**: Generate symb states
- **Task distribution for parallel execution**
- **Atomic tasks**
- **PostD cost Computation**

**PostC**

**PostD**

Amit Gurung (NIT Meghalaya)
Proposed a cheap flowpipe cost estimating approach with an error bounded by $|t - t'| < \delta_F$.

where $t$ is the exact time, $t'$ is the computed approximate time and $\delta_F$ is the fine time-step.

**Algorithm 4** Detecting time of invariant crossing with varying time-step

1: **procedure** INARIANT-CROSSING TIME DETECTION($\mathcal{I}$, $X_0$, $T$)
2: $\text{discretization} = 10, \tau = T/\text{discretization}$ $\triangleright$ Coarse Time-step
3: $i = 0, R(0) = X_0$
4: while $R(\tau.i) \vdash \mathcal{I}$ do $i = i + 1$ $\triangleright$ Widened Search
5: end while
6: if $i > 1$ then $t_1 = \tau \ast (i - 1)$
7: else return 0
8: end if
9: $\tau = \tau/\text{discretization}, i = 0$ $\triangleright$ Fine Time-step
10: while $R(t_1 + i \ast \tau) \vdash \mathcal{I}$ do $i = i + 1$ $\triangleright$ Narrowed Search
11: end while
12: return $t_1 + i \ast \tau$ $\triangleright$ An upper bound on invariant crossing time
13: **end procedure**

**Figure**: Coarse and Fine time-step image mapping
Experimental Results: Performance comparison

**Setup:** 12 Core Intel Xeon CPU 2.30GHz with 62GB RAM (hyper-threading disabled)

(a) Switching Oscillator Model

(b) Timed Bouncing Ball Model

(c) Circle Model

(d) Nav 3x3 (#inst 1) Model
Experimental Results: Performance comparison

Performance Speed-up in Navigation 3x3 (#inst 2) Model

(e) Nav 3x3 (#inst 2) Model

Performance Speed-up in Navigation 3x3 (#inst 3) Model

(f) Nav 3x3 (#inst 3) Model

Performance Speed-up in Navigation 5x5 Model

(g) Nav 5x5 Model

Performance Speed-up in Navigation 9x9 Model

(h) Nav 9x9 Model
Experimental Results: CPU utilization

(i) Switching Oscillator Model

(j) Timed Bouncing Ball Model

(k) Circle Model

(l) Nav 3x3 (#inst 1) Model
Experimental Results: CPU utilization

CPU Utilization in Navigation 3x3 (#inst 2) Model

(m) Nav 3x3 (#inst 2) Model

CPU Utilization in Navigation 3x3 (#inst 3) Model

(n) Nav 3x3 (#inst 3) Model

CPU Utilization in Navigation 5x5 Model

(o) Nav 5x5 Model

CPU Utilization in Navigation 9x9 Model

(p) Nav 9x9 Model
Results: Performance comparison between G. J. Holzmann and Load Balancing algorithm

Figure: 9 × 9 Navigation model with sampling time of $1e^{-3}$
Conclusion

- Link to the benchmarks description is the tool XSpeed can be downloaded from http://nitmeghalaya.in/nitm_web/fp/cse_dept/XSpeed/downloads.html.

- We have presented two parallel BFS algorithms for reachability analysis of hybrid systems and implemented them in the model checker XSpeed.

- The first algorithm A-GJH, is an adaption of G.J. Holzmann (SPIN model checker) and second TP-BFS an improvement over A-GJH.

- When the number of symbolic states to be explored is less than the number of cores available or the cost of flowpipe computation is uneven, A-GJH fails to perform efficient load balancing.

- The TP-BFS algorithm have shown an efficient load balancing for all models of hybrid systems and yields better performance and CPU utilization than the A-GJH algorithm.

- In cases when the number of symbolic states is significantly large, A-GJH outperformed TP-BFS in performance/CPU utilization due to the overheads involved in the load balancing tasks.
Rajarshi Ray and Amit Gurung
Parallel state space exploration of linear systems with inputs using XSpeed

Rajarshi Ray, Amit Gurung, Binayak Das, Ezio Bartocci, Sergiy Bogomolov and Radu Grosu
XSpeed: Accelerating Reachability Analysis on MultiCore Processors
In Hardware and Software: Verification and Testing- 11th International Haifa Verification Conference, HVC Nov-2015, Haifa, Israel.

Colas Le Guernic and Antoine Girard
Reachability Analysis of Hybrid Systems using Support Functions
Thank You