Analyzing Data Structure Choices for On-The-Fly Real Time Model Checking

Peter Fontana and Rance Cleaveland
University of Maryland, College Park
Work In Progress, April 28, 2011
Real-Time Model Checking

TCTL (Invalid): $\text{AF}_\infty^{\near \vee \text{in}}$

TCTL (Valid): $\text{AG}_\infty^{\near \rightarrow \text{AF}_{\leq \text{TP} + \text{TDU}}^{\text{far}}}$
Background

- Timed Automata model checkers
  - UPPAAL, RED, KRONOS
  - Restricted sets of properties
- Predicate Equation Systems (PES) [Zhang, Cleaveland, 2005]
  - First order logic with fixpoint formulae
  - General framework for on-the-fly model checking
On-The-Fly Model Checking

- Goal-directed proof construction
- Uses circularity to detect fixpoints
- For timed automata:
  - *Clock zones* represent sets of states concisely
  - Clock zone data structures important for performance
Goals

- Investigate the impact of clock zone data structures of on-the-fly model checking performance

**Context:** use PES engine to model check a subset of SIMULINK
Clock Zones

- **Example:** \( x_1 = 2 \land x_2 < 3 \land x_1 - x_3 \leq 1 \)

- **Clock Zone** = *convex* set of clock constraints

- **Definition:**

  \[
  z := x < c \mid x > c \mid x \leq c \mid x \geq c \mid x - y < c \\
  \mid x - y > c \mid x - y \leq c \mid x - y \geq c \mid z_1 \land z_2
  \]
Clock Zone Implementations

- **DBM**: Matrix (Difference Bound Matrix)

- **CRDZone**: Linked list, nodes in lexicographical order (omit implicit nodes)
Experiment

- **Purpose:** Analyze performance of DBM, CRDZone on PES-based on-the-fly model checking

- **Hypothesis:** The CRDZone will improve *time* and space performance

- **Setup:**
  - Replace DBM with CRDZone in model checker
  - Compare time, space on various benchmarks
Benchmark Suite

- **A**: valid specification, correct system
- **B**: invalid specification, correct system
- **C**: valid specification, buggy system
- 21 model-checker invocations per category
### Preliminary Data Analysis

- **Compare paired differences** between DBM and CRDZone

**Conclusions:**
- CRDZone performs slightly faster for majority
- Huge variation

<table>
<thead>
<tr>
<th>Statistic</th>
<th>DBM – CRDzone (time - s)</th>
<th>DBM – CRDZone (space – MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Benchmark</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>Mean</td>
<td>0.42</td>
<td>-104.0</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>1001.40</td>
<td>298.2</td>
</tr>
<tr>
<td>95% CI (Mean)</td>
<td>-333.67 – 334.10</td>
<td>-203.4 – (-4.6)</td>
</tr>
<tr>
<td>P-Value for Mean ≠ 0</td>
<td>0.999</td>
<td>0.033</td>
</tr>
<tr>
<td>Median</td>
<td>7.21</td>
<td>-0.5</td>
</tr>
<tr>
<td>P-Value for Median ≠ 0</td>
<td>0.012</td>
<td>0.157</td>
</tr>
</tbody>
</table>
Future Work

- Expand checkable specification range
- Continue optimizing code for performance
- Further uses for PES Engine
  - SIMULINK
  - Vacuity checking