Efficient TCTL Model Checking Algorithm for Timed Actos

Ehsan Khamespanah, Ramtin Khosravi, and Marjan Sirjani
University of Tehran in Iran and Reykjavik University in Iceland
AGERE@SPLASH
October 2014
Analysis Support

- Floating Time Transition System for Schedulability and Deadlock-Freedom analysis
- Transforming to Erlang to simulate
- Event-Based Property Analysis using Simulation Engine of Timed Rebeca
- TCTL verification of Timed Rebeca models
Difficulties of TCTL Model Checking

• There is no efficient algorithm

• Only a subset of TCTL can be check efficiently

• Does having discrete time make it easier?
  • Duration transition graph (DTG) is a good alternative
Semantics Of Timed Rebeca in Timed Transition System

- The tuple $TTS = (S, s_0, Act, \rightarrow, AP, L)$ is the timed transition system of a Timed Rebeca model where:
  - $S$ is a set of states $\prod_{i \in I} (V_{s,i}, B_{s,i}, pc_{s,i}, res_{s,i}, now_s)$
  - $Act$ is a set of actions $\bigcup_{i \in I} ((I \times i \times M_i) \times \mathbb{N} \times \mathbb{N}) \cup \{\tau\} \cup \mathbb{N}$
  - Three types of transitions
    - Taking an event
    - Internal transition
    - progress of time
Simple Timed Rebeca Model

reactiveclass RC1 (3) {
  knownrebecs {
    RC2 r2;
  }
  RC1() {
    self.m1();
  }
  msgsrv m1() {
    delay(2);
    r2.m2();
    delay(2);
    r2.m3();
    self.m1() after (10);
  }
}

reactiveclass RC2 (4) {
  knownrebecs {
    RC1 r1;
  }
  RC2() {
  }
  msgsrv m2() {
  }
  msgsrv m3() {
  }
}

main {
  RC1 r1(r2):();
  RC2 r2(r1):();
}
Simple Timed Rebeca Model

```rebeca
reactiveclass RC1 (3) {
    knownrebecs {
        RC2 r2;
    }
    RC1() {
        self.m1();
    }
    msgsrv m1() {
        delay(2);
        r2.m2();
        delay(2);
        r2.m3();
        self.m1() after (10);
    }
}

reactiveclass RC2 (4) {
    knownrebecs {
        RC1 r1;
    }
    RC2() {
    }
    msgsrv m2() {
    }
    msgsrv m3() {
    }
}

main {
    RC1 r1(r2):();
    RC2 r2(r1):();
}
```

Line number as program counter
msgsrv m1() {
  1 delay(2);
  2 r2.m2();
  3 delay(2);
  4 r2.m3();
  5 self.m1() after (10);
}
msgsrv m1() {
  1    delay(2);
  2    r2.m2();
  3    delay(2);
  4    r2.m3();
  5    self.m1() after (10);
}
msgsrv m1() {
  1 delay(2);
  2 r2.m2();
  3 delay(2);
  4 r2.m3();
  5 self.m1() after (10);
}
Two different semantics

• Jump semantics

• Continuous semantics

Duration Transition Graph

- Example of a DTG

- Two different semantics

- Jump semantics

- Continuous semantics
TCTL Model Checking for DTG

- There is a polynomial time model checking algorithm for TCTL<> properties but model checking of TCTL\(_\leq\) is NP-Complete

- Combination of CTL model checking and shortest/longest path search
  
  - Try to find a set of states which satisfy a given TCTL formula without any timed constraint

- Check timed constraints
How the Algorithm Works for $\exists \Phi_1 U^{\leq c} \Phi_2$

- Assume $\text{DTG}^\text{sub}_M = (S', s'_0, \text{Act}, \rightarrow', \text{AP}', L')$ as a reduced version of $\text{DTG}_M$ which satisfies $\exists \Phi_1 U \Phi_2$

- Any state $s$ in $\text{DTG}^\text{sub}_M$ satisfies the time version of the formula if and only if there is path from $s$ to state $s'$ such that the length of path is less than the time constraint.

- This can be checked using $O(n^2)$ algorithm
TCTL<> Model Checking of Timed Rebeca Models

- TTS of a Timed Rebeca model is a DTG

- All the transitions are assumed to be internal transitions (no action label) with zero time duration

- Progress-of-Time transitions are assumed as transitions with tight duration

- Model checking Timed Rebeca models against TCTL<> properties is possible in $O(n^2)$
A New Reduction Technique

- There are some transient states in the state space (Residual time is zero in transient states)
- Transient behavior is not interested in some systems
  - It is possible to eliminate transient states from the state space
  - The overhead of reduction must be smaller than the gain of model checking on smaller state space
Example of How Reduction Technique Works
Cost Free Reduction Technique!

- Timed Rebeca models must be checked to be Zeno free
- It can be checked by using DFS search to detect cycles without progress-of-time states
- Timed model is discrete in Timed Rebeca

Algorithm 1: ZenoFree(s) analyzes the model for Zeno-freedom.

```
Input: State s of a timed transition system T
Output: The part of T reachable from s is Zeno-free or not

visited ← ∅
for all the state s' ∈ Successors(s) do
    if s' ∈ visited then
        visited ← visited ∪ {s'}
        recStack(s') ← true
    if ZenoFree(s') = false then
        return false
    recStack(s') ← false
else
    if recStack(s') = true ∧ now(s') = now(s) then
        return false
return true
```

13
Cost Free Reduction Technique!

• Reduction technique requires BFS traversal to figure out the set of next level progress-of-time states of each state

• Combination of BFS and DFS is required
  • Explore the states among two consequent progress-of-time states by bounded-DFS
  • Then go to the next level
Algorithm 2: BoundedZenoCheck(s) makes sure that there is no cycle among reachable states from s to npts(s). Also sets the nearest progress-of-time states of all the reachable states from s to npts(s).

Input: State s of a timed transition system
Output: The bounded reachable part of the transition system is Zeno-free or not

visited ← ∅

forall the state s' ∈ Successors(s) do
  if s' ≠ visited then
    visited ← visited ∪ {s'}
    if s' is progress-of-time then
      //DFS has reached one of its boundaries
      npts(s) ← npts(s) ∪ {s'}
    else
      if now(s) = now(s') then
        recStack(s') ← true
        childsNPTS ← BoundedZenoCheck(s')
        recStack(s') ← false
        if childsNPTS = ∅ then
          return ∅
        else
          npts(s) ← npts(s) ∪ childsNPTS
      else
        //Back-edge is detected
        npts(s) ← npts(s) ∪ npts(s')
  else
    if recStack(s') = true then
      //There is cycle which shows Zeno behavior
      return ∅

return npts(s)

Algorithm 3: FTS(TTS_M) creates the corresponding FTS of a given TTS or returns ∅ in the case of Zeno behavior in the model.

Input: Timed transition system
\( TTS_M = (S, s_0, Act, →, AP, L) \)
Output: Folded timed transition system of M

S' ← \{s_0\}
Act' ← ∅
→ ← ∅
AP' ← AP
L' ← L
openBorderStates ← \{s_0\}
nextLevelStates ← ∅

repeat
  while openBorderStates ≠ ∅ do
    remove s from openBorderStates
    NPTS ← BoundedZenoCheck(s)
    if NPTS = ∅ then
      return ∅
    else
      nextLevelStates ← nextLevelStates ∪ NPTS
      S' ← S' ∪ NPTS
      foreach s' ∈ NPTS do
        → ← U \{(s, act', s')\}
        Act' ← Act' ∪ {act'}
      end for
      openBorderStates ← nextLevelStates
      nextLevelStates ← ∅
  until openBorderStates ≠ ∅

return (S', s_0, Act', →, AP, L)
More Than Smaller State Space

- Using reduction technique, we are able to efficiently check the model for $\text{TCTL}_= \text{ properties}$

- There is a pseudo-polynomial algorithm for finding exact path among two nodes of the graph

- Preprocessing a graph to uniform the weights of edges by an $O(W^2 \cdot n^3)$ algorithm

- Looking for exact path by an $O(|k| \cdot \min\{|k|, w\} \cdot n^2)$ algorithm
Polynomial Algorithm for Model Checking of TCTL$_\leq$ Properties

• All the weights in TTS are positive integers, so there is no need for an $O(W^2 n^3)$ relaxation algorithm

• In model checking problem, “k” related to the maximum bound number which is used in TCTL$_\leq$ formula

  • In many cases “k” can be assumed as an small constant

  • The complexity of finding exact path is reduced to $O(|k|^2 n^2) = O(n^2)$
Experimental Results

- Four different models are used
- 90% reduction in the state space size in some cases

<table>
<thead>
<tr>
<th>Problem</th>
<th>Size</th>
<th>State Space Size</th>
<th>Reduced State Space Size</th>
<th>Percentage of Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ticket Service</td>
<td>2 customers</td>
<td>77</td>
<td>10</td>
<td>87%</td>
</tr>
<tr>
<td></td>
<td>3 customers</td>
<td>360</td>
<td>39</td>
<td>89%</td>
</tr>
<tr>
<td></td>
<td>4 customers</td>
<td>1825</td>
<td>184</td>
<td>90%</td>
</tr>
<tr>
<td></td>
<td>5 customers</td>
<td>10708</td>
<td>1045</td>
<td>90%</td>
</tr>
<tr>
<td></td>
<td>6 customers</td>
<td>73461</td>
<td>6996</td>
<td>90%</td>
</tr>
<tr>
<td></td>
<td>7 customers</td>
<td>581962</td>
<td>54019</td>
<td>91%</td>
</tr>
<tr>
<td>WSAN</td>
<td>-</td>
<td>1920</td>
<td>818</td>
<td>57%</td>
</tr>
<tr>
<td>Yarn</td>
<td>-</td>
<td>533</td>
<td>172</td>
<td>68%</td>
</tr>
<tr>
<td>8x8 NoC</td>
<td>-</td>
<td>74192</td>
<td>6068</td>
<td>92%</td>
</tr>
</tbody>
</table>
Conclusion

• We can model check Timed Rebeca models against TCTL$_{<>}$ properties in polynomial time.

• A combination of checking for Zeno freedom and reduction technique is proposed
  
  • Reducing the state space size without overhead

• We propose an approach which works for model checking of wide range of TCTL$_{=} \quad$ formulas in polynomial time
Thank you