CREDES related research at the Dept. of Computer Science TUT

Jüri Vain

CREDES Kick-off Meeting Tallinn, June 05, 2009

Research lines

- (Timed) automata learning
- Proof techniques for swarm coordination algorithms
- State isomorphism based symmetry reduction
- Iterated search refinement with bit-state pruning
- Test trace generation using MC
- Synthesis of test goal directed reactive testers

- model-based planning in...
- model checking using...
- model-based test generation in...
- interacting agent systems
- multi-agent systems
- state space reduction by abstraction
- determinist systems
- non-deterministic systems
- FMs for model-based planning in...
State isomorphism based symmetry reduction

- Solution
  - Given a model in terms of unordered data structures, like sets and maps, and objects:
    - Construct the state graphs of the states at run time and
    - Use the computation of graph isomorphism to determine whether a state with similar structure has been seen.

- Results developed with Margus Veanes and Colin Campell and published at FORTE 2007 (June 2007) and in the PhD thesis of J. Ernits (Nov. 2007)
State isomorphism based symmetry reduction

Problem
- Exploit symmetries to reduce the effects of state space explosion in explicit state reachability

Applications:
- Method is implemented in model-testing toolkit called NModel (by Margus Veanes et al, MS Research Redmond)
- There is similar implementation developed independently by Arend Rensink in the GROOVE project
Research lines

- (Timed) automata learning
- Proof techniques for swarm coordination algorithms
- State isomorphism based symmetry reduction
- Iterated search refinement with bit-state pruning
- Test trace generation using MC
- Synthesis of test goal directed reactive testers

Iterated search refinement with bit-state pruning

- Problem
  - Calculate reachability in models that
    - Run out of resources without producing results with known methods
    - Produce too long traces to the error/desired state with known methods
  - Such reachability problems can be used for solving problems related to
    - Scheduling
    - Hardware synthesis
    - Offline test generation
Iterated search refinement with bit-state pruning

Solution
- Combine two well known methods in a new way:
  - Iterated Search Refinement, and
  - Bitstate hashing
- The key idea is to use collisions in bitstate hash table to randomly filter the search space.
  - Each iteration requires very small amounts of memory
  - A lot of prefixes of paths are covered multiple times, but
  - Provides good results in several examples.
- Results published in the PhD thesis of J. Ernits (Nov. 2007)

Iterated search refinement with bit-state pruning

Practical applications
- 2 well researched case studies:
  - Synthesis of a memory arbiter for a radar memory case study (J. Ernits, 2005)
  - Calculating offline test sequences for model-based testing (J. Ernits, A. Kull, K. Raiend, J. Vain, 2006)
- The method cannot be used for disproving the reachability, but provides a new alternative for finding a witness trace.
Pre-set test trace generation using MC

Case study: modified INRES protocol

Results
- Time spent for finding sequences for the 2-switch coverage criterion
- Experiments made with Uppaal Cora

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>BC</th>
<th>Search order</th>
<th>Trace Cutting</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>-</td>
<td>depth first</td>
<td>some</td>
</tr>
<tr>
<td>IterDF</td>
<td>-</td>
<td>depth first</td>
<td>some</td>
</tr>
<tr>
<td>IterRBDF</td>
<td>-</td>
<td>random</td>
<td>best uniform</td>
</tr>
<tr>
<td>IterRBDF tuned</td>
<td>-</td>
<td>random</td>
<td>best tuned</td>
</tr>
</tbody>
</table>
Synthesis of test goal directed reactive testers

- **Problem:**
  - Given a non-deterministic EFSM $M_{SUT}$ of System Under Test
  - testing goal in terms of a set $Trp$ of $M_{SUT}$ transitions

- Find EFSM $M_{TSR}$ s.t.:
  - $M_{TSR}$ is I/O compliant with $M_{SUT}$
  - any trace of $M_{SUT} \parallel M_{TSR}$ includes labels of all transitions in $Trp$
  - $M_{TSR}$ chooses a transition from those labelled with current input from $M_{SUT}$ that maximizes some gain function $G$. 
Synthesis of test goal directed reactive testers

- Solution:
  - Algorithm of constructing $M_{TSR}$

- Complexity:
  - For all controllable transitions of the $M_{TSR}$, the upper bound of the complexity of the computations of the gain functions is $O(|E_{SUT}|^3)$.
  - At runtime each choice by the tester takes $O(|E_{SUT}|^2)$ arithmetic operations to evaluate the gain functions

(ASE2007, Vain, Raiend, Kull, Ernits)

---

**Synthesis of test goal directed reactive testers**

- Experimental results

<table>
<thead>
<tr>
<th>Algorithm of the tester</th>
<th>Model 1 (8 trans.)</th>
<th>Model 2 (16 trans.)</th>
<th>Model 3 (32 trans.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random choice</td>
<td>56 ± 36</td>
<td>295 ± 130</td>
<td>1597 ± 1000</td>
</tr>
<tr>
<td>Antis</td>
<td>21 ± 4</td>
<td>53 ± 13</td>
<td>218 ± 81</td>
</tr>
<tr>
<td>Reactive planner</td>
<td>17 ± 3</td>
<td>37 ± 6</td>
<td>80 ± 10</td>
</tr>
</tbody>
</table>

---

**One Transition Test Purpose**

<table>
<thead>
<tr>
<th>Algorithm of the tester</th>
<th>Model 1 (8 trans.)</th>
<th>Model 2 (16 trans.)</th>
<th>Model 3 (32 trans.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random choice</td>
<td>34 ± 35</td>
<td>120 ± 114</td>
<td>699 ± 719</td>
</tr>
<tr>
<td>Anti-antis</td>
<td>14 ± 7</td>
<td>36 ± 19</td>
<td>140 ± 70</td>
</tr>
<tr>
<td>Reactive planner</td>
<td>5 ± 2</td>
<td>8 ± 3</td>
<td>11 ± 3</td>
</tr>
</tbody>
</table>
State-of-the-art

- CS TUT test generator is used for TTCN-3 test generation
- Feeder Box Control Unit - System Under Test
- Measurement and control devices make up system adapter HW part
- LabVIEW is used for implementing system adapter SW part

Research lines

- (Timed) automata learning
- Proof techniques for swarm coordination algorithms
- State isomorphism based symmetry reduction
- Iterated search refinement with bit-state pruning
- Test trace generation using MC
- Synthesis of test goal directed reactive testers

- FMs for model based planning in...
- model checking using...
- model based test generation in...

- single agent systems
- multi-agent systems
- state space reduction by abstraction
- determinitic systems
- non-deterministic systems
- model checking using...
- model based test generation in...
Proof techniques for swarm coordination algorithms

Problem:
- proving properties of swarm (distributed) coordination algorithms
- self-stabilization, convergence, ... can be reduced to reachability analysis
- reachability analysis collides with scalability barrier

Some solutions:
- Pattern based problem encoding provides right level of abstraction (spec. patterns are problem oriented)
- General principles of proving:
  - Compositional approach combining different techniques of component proofs – MC, deduction,...
  - Global properties by structural induction on the size of swarm/task
    - Base: prove sufficiency of assumptions on a swarm fragment of some tractable size, e.g., using MC with symmetry reduction, (typically the size of fragment can’t be trivial either) (see BEC2008, Vain, Kuusik, Tammet, Juurik)
    - Step: Show that the induction step does not violate the assume part for the new fragment defined by the step
Proof techniques for swarm coordination algorithms

- Demo: dynamic cleaning problem
  - different zones of the room deteriorate with different rate
  - a swarm of cleaning robots should keep deterioration below threshold
  - robots of the swarm communicate through pheromone trace
- Problem: given a threshold $C_{\text{Tresh}}$ and initial value of the deterioration vector $S^{det}(0)$, s.t. $S^{det}_i(0) > C_{\text{Tresh}}$ for all $i$,
- prove that
  1) the swarm is always able to reach the state $S^{det}(t)$, s.t. $S^{det}_i(t) < C_{\text{Tresh}}$ for all $i$,
  and
  2) for all $t' > t$, condition $S^{det}(t') < C_{\text{Tresh}}$ is invariantly true.

Research lines

- (Timed) automata learning
- Proof techniques for swarm coordination algorithms
- State isomorphism based symmetry reduction
- Iterated search refinement with bit-state pruning
- Test trace generation using MC
- Synthesis of test goal directed reactive testers
- FMs for model based planning in...
- model checking using...
- model based test generation in...
- interacting agent systems
- multi-agent systems
- state space reduction by abstraction
- deterministic systems
- non-deterministic systems
- deterministic systems
- non-deterministic systems
- interacting agent systems
- multi-agent systems
- state space reduction by abstraction
- deterministic systems
- non-deterministic systems
- interacting agent systems
- multi-agent systems
- state space reduction by abstraction
- deterministic systems
- non-deterministic systems
HRI automata learning

- Problem: Planner synthesis for a human adaptive Scrub Nurse Robot (SNR)
  - human adaptive = on-line learning from H-H/H-R interactions
  - learning = constructing/modifying a human motion model

(Timed) automata learning

- Context: SNR control architecture
(Timed) automata learning

- Learning algorithm
  - Input:
    - Time-stamped sequences of observations of HRI (set of timed traces $Tr^T(Obs)$)
    - Parameters for
      - state rescaling operator - $R$
      - distinguishing model observables/controllables - $|_{TAIO}$
      - defining quotient state space
  - Output:
    - Extended (Uppaal version) timed automaton $TA$ s.t.
      $Tr^T(TA) = R(Tr^T(Obs))|_{TAIO} / \sim_{RTIOCO}$ of timed traces

(Vain, Miyawaki, ICCAS-SICE2009, Fukuoka, Japan)

(Timed) automata learning (example)
Interaction model constructed

MB Test & Verification

<table>
<thead>
<tr>
<th>SoA</th>
<th>Beyond SoA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB test generation:</td>
<td>on-line learning of SUT models</td>
</tr>
<tr>
<td>M construction off-line</td>
<td></td>
</tr>
<tr>
<td>Testing non-deterministic systems:</td>
<td>Random walk, anti-ant, .. goal-oriented on-line planning testers: RPT</td>
</tr>
<tr>
<td>Test data construction:</td>
<td>on-the-fly constraint &amp; solving</td>
</tr>
<tr>
<td>off-line constraint solving</td>
<td></td>
</tr>
<tr>
<td>Distributed testing:</td>
<td>fully distributed control</td>
</tr>
<tr>
<td>with centralized control</td>
<td>(coordination &amp; synchronization)</td>
</tr>
</tbody>
</table>
Thank You!