Test Generation based on Abstraction and Test purposes to complement Structural tests

F. Bouquet, P.-C. Bué, J. Julliand, P.-A. Masson

A-MOST 2010 - April 6, 2010 - Paris
Context

Model-Based Testing

- MBT from static selection criterion
- MBT from dynamic selection criterion
Context

Model-Based Testing

- MBT from static selection criterion
- MBT from dynamic selection criterion
Problem and Motivations

Motivations
- Complete the tests from static criteria with tests from dynamic criteria
- Evaluate the coverage of the generated tests

Problem
- Combinatorial explosion of the size of an LTS

Solution
- Abstraction to reduce the size of the state space
Problem and Motivations

Process overview

Process application
- The robot example
- Definition of an abstraction
- Synchronization between the abstraction and the test purpose
- Test generation
- Instantiation of tests

Experimental results
- Abstraction computation
- Test generation

Conclusion and Future works
Our process

Differences

- Introduction of an abstraction computation
- Replace the model by an abstraction
A conveying system: Robot

Specifications

- A gripper moves parts in two dimensions
- 3 sort of parts: \( T_1, T_2, T_3 \)
- Operations: Arrival, Load, Unload, Up, Down, Left, Right, LeftEvac, RightEvac
- Unloading rules: \( T_1 \) left, \( T_2 \) right, \( T_3 \) regardless
- \( T_3 \) parts: Left unloading priority
A test purpose for Robot

What is a test purpose?

- Description of a test intention
- Combine operation calls and states to reach
- Formalization: state machine or regular expression

Left unloading priority

F. Bouquet, P.-C. Bué, J. Julliand, P.-A. Masson
Definition of an abstraction

What is an abstraction?
- Gives a thumbnail of the model
- Simulates executions of the model

Abstraction computation with GeneSyst (Bert, Potet, Stouls)
- Disjunction of state predicates defined by user
- One predicate $\iff$ one symbolic state of the abstraction
The **GeneSyst** tool

### Principles

~ How does **GeneSyst** compute an abstraction?

- Predicate abstraction
- Computation of weakest precondition and resolution of satisfiability problems with prover technology
- Proof failure for a transition $\Rightarrow$ Transition is kept in the abstraction
- Filter transitions with **CLP**

~ How to define state predicates?
Defining a set of abstraction predicates

- Predicates which reveal changes in test purpose
- Which ones do we propose?
  - Present used to define target states in test purpose
  - Involving on variables modified by all operations of test purpose
  - Defining a partition of the state space

Left unloading priority

\[(\text{op}^+ \leadsto (LE \in \{T_1, T_3\} \land G = T_3))\]

- Right
- LeftEvac
- \[(\text{op}^+ \leadsto (G = T_3))\]
- Unload
Defining a set of abstraction predicates

- Predicates which reveal changes in test purpose
- Which ones do we propose?
  - Present used to define target states in test purpose
  - Involving on variables modified by all operations of test purpose
  - Defining a partition of the state space

Left unloading priority

\[
LE \in \{T_1, T_3\} \\
G \in T_3
\]

\[(\$op)^+ \leadsto (LE \in \{T_1, T_3\} \land G = T_3)\]

- Right
- LeftEvac
- \((\$op)^+ \leadsto (G = T_3)\)
- Unload
Defining a set of abstraction predicates

- Predicates which reveal changes in test purpose
- Which ones do we propose?
  - Present used to define target states in test purpose
  - Involving on variables modified by all operations of test purpose
  - Defining a partition of the state space

\[
LE = \text{free} \\
G = \text{free}
\]

**Left unloading priority**

\[
($op)^+ \leadsto (LE \in \{ T_1, T_3 \} \land G = T_3)
\]

- **Right**
- **LeftEvac**
- \((($op)^+ \leadsto (G = T_3))
- **Unload**
Defining a set of abstraction predicates

- Predicates which reveal changes in test purpose
- Which ones do we propose?
  - Present used to define target states in test purpose
  - Involving on variables modified by all operations of test purpose
  - Defining a partition of the state space

### Left unloading priority

\[(\text{lop})^+ \leadsto (\text{LE} \in \{T_1, T_3\} \land G = T_3)\]

- Right
- LeftEvac
- \((\text{lop})^+ \leadsto (G = T_3)\)
- Unload
Abstraction computation

\[
\begin{align*}
\text{LE} & \quad \{\text{free}\}, \{T_2\}, \{T_1, T_3\} \\
\text{G} & \quad \{\text{free}\}, \{T_1, T_2\}, \{T_3\}
\end{align*}
\]
Synchronization between the abstraction and the TP

Abstraction

Test purpose

The robot example
Definition of an abstraction
Synchronization between the abstraction and the test purpose
Test generation
Instantiation of tests
The robot example
Definition of an abstraction
Synchronization between the abstraction and the test purpose
Test generation
Instantiation of tests

Synchronization between the abstraction and the TP

Synchronization

- Instantiation of \((\text{op})^+\)

\[ (\text{op})^+ \sim (LE \in \{T_1, T_3\} \land G = T_3) \]
- Right
Generation and Instantiation of tests

Generation of a set of symbolic tests
- *All-edges* criterion on synchronized product
- Application of the Chinese Postman Algorithm

Instantiation of a set of symbolic tests
- Why isn’t instantiation complete?
  - Tests use over-approximated transitions in abstraction
- How do we deal with that?
  - Interleaving of reflexive operations calls in tests
Instantiation of symbolic tests

Symbolic test

\[ \ldots \leadsto (G = \text{free}) \]

. Load \( \leadsto (G = T_3) \)

Instantiated test

\[ \ldots \leadsto (G = \text{free}) \]

. Arrival \( \leadsto (G = \text{free}) \)

. Load \( \leadsto (G = T_3) \)

\[ \ldots \ldots \ldots \]
Instantiation of symbolic tests

Symbolic test

\[
\ldots \leadsto (G = free) \\
\text{Load} \leadsto (G = T_3)
\]

Instantiated test

\[
\ldots \leadsto (G = free) \\
\text{Arrival} \leadsto (G = free) \\
\text{Load} \leadsto (G = T_3) \\
\ldots \ldots
\]
Instantiation of symbolic tests

Symbolic test

\[ \ldots \leadsto (G = \text{free}) \]
\[ \text{Load} \leadsto (G = T_3) \]

Instantiated test

\[ \ldots \leadsto (G = \text{free}) \]
\[ \text{Arrival} \leadsto (G = \text{free}) \]
\[ \text{Load} \leadsto (G = T_3) \]
\[ \ldots \ldots \]
Abstraction computation - Synchronization

- Reduction of the size of the models
- 1 abstraction per 1 test purpose
- Abstraction computation : time consuming

<table>
<thead>
<tr>
<th>B Model</th>
<th>#States</th>
<th>Abstraction computation</th>
<th>Synchronization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#States</td>
<td>#Trans.</td>
</tr>
<tr>
<td>Qui-Donc</td>
<td>13</td>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td>Robot</td>
<td>372</td>
<td>6</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>DeMoney</td>
<td>$10^{30}$</td>
<td>3</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>228</td>
</tr>
</tbody>
</table>

F. Bouquet, P.-C. Bué, J. Julliand, P.-A. Masson

Abstractions of models for test
### Experimental results

#### Test generation - Instantiation

- 😊 Number of instantiated tests increase with filtering
- 😞 Instantiation and filtering are time consuming with large example

<table>
<thead>
<tr>
<th>B Model</th>
<th>Without filtering</th>
<th>With filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Instant. tests / #Symbolic tests</td>
<td>Time of instantiation</td>
</tr>
<tr>
<td>Qui-Donc</td>
<td>4/10 (40%)</td>
<td>≤ 1 s.</td>
</tr>
<tr>
<td></td>
<td>5/15 (33%)</td>
<td></td>
</tr>
<tr>
<td>Robot</td>
<td>7/12 (58%)</td>
<td>3 min.</td>
</tr>
<tr>
<td></td>
<td>8/23 (35%)</td>
<td>5 min.</td>
</tr>
<tr>
<td>DeMoney</td>
<td>0/32 (0%)</td>
<td>2 h.</td>
</tr>
<tr>
<td></td>
<td>0/32 (0%)</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion and Future works

Conclusion

😊 Automatic extraction of abstraction predicates
😊 Abstraction related to test purpose
😊😊 1 test purpose per 1 abstraction
😊 Computation time of abstraction
😊 Too much over-approximation for efficient test instantiation

Future works

○ Common abstraction (1 abstraction for N tests purposes)
○ Under-approximations rather than over-approximations
  ○ *Tri-modal Transitions System* generation (*Godefroid, Ball*)
Thanks for your attention

Questions?