Garakabu2: A Formal Verification Tool for ZIPC

Weiqiang Kong, Kyushu University, Japan

[Jointly developed by]
Fukuoka IST & CATS Co. Ltd. & Kyushu University of Japan

[With technical supports from]
AIST of Japan

Industry Day Program of FM2012
August 30, 2012
Outline

- Garakabu2 and ZIPC – A general introduction
- The SMT-based BMC approach in Garakabu2
- Towards practical usability for on-site software engineers
Outline

- Garakabu2 and ZIPC – A general introduction

  The SMT-based BMC approach in Garakabu2

  Towards practical usability for on-site software engineers
Garakabu2 in General

- Software designs in ZIPC-HSTM
- Properties in LTL
- Checking the properties against the design with BMC
- Tracking and understanding error reasons
- Revising the design and recheck...

ZIPC

Garakabu2
Features of Garakabu2

- Designs developed with ZIPC can be checked as they are
  - Easy to use: no need to learn a 2nd MC specific description language

- Traces of CX can be illustrated in ZIPC
  - Easy to understand: relatively easier to understand the reasons for errors

- Previous MC results can be saved and replayed
  - Easy to make a later confirmation for previous designs/checks (traceability)

- LTL properties can be specified more intuitively
  - Write/Draw LTL properties by patterns or figures (with SpecEditor of AIST)
Our intention:

Make Garakabu2 an easy-to-use formal verification tool for on-site software engineers who have not much knowledge in formal methods.
ZIPC in General

A CASE Tool (or ... model-based development tool)

- No overlooking of possibly abnormal cases;
- Automatically generating source codes (framework);
- Syntax check, graphical simulation;
- ...

➢ No formal verification supports
Hierarchical State Transition Matrix (HSTM)

- STM is a table that captures an object's behaviors under event-state match;
- An HSTM is a set of STMs organized in a hierarchical structure with calling-to and returning-from the execution of Child STMs;
- An HSTM design is a set of HSTMs executing in an interleaving manner.
A Simplified Money-Exchange Machine (MEM) in HSTM

HSTM1: Specifies interaction with users and actual bill exchange functionality;
HSTM2: Specifies exchanged bill payback functionality.
ZIPC – Image Demo (Japanese)
Outline

Garakabu2 and ZIPC – A general introduction

- The SMT-based BMC approach in Garakabu2

Towards practical usability for on-site software engineers
SMT solving and SMT solvers

- SMT (Satisfiability Modulo Theories) Solving Technique
  A technique of deciding satisfiability of a given quantifier-free formula, i.e., finding a variable-assignment that makes the formula TRUE.

- SMT Solvers
  CVC3 (New York Univ.), Yices (SRI International), Z3 (Microsoft), etc.
Basic ideas for the encoding in Garakabu2

Satisfiable: Counterexample;
Unsatisfiable: holds within bound

Can formula: \( \text{step}_0 \text{ and } \ldots \text{and } \text{step}_n \text{ and } \neg(\text{prop}) \) be satisfied/true?

LTL Properties

HSTM

CVC3
Basic Ideas (Informal) for Encoding an HSTM Design

Basic Encoding Rules

- **CELL** := \((\text{cell.event} \land \text{cell.status}) \Rightarrow (\text{cell.action} \land \text{cell.statusTransit} \land \text{untouchedVars})\)
- **STM** := \((\text{CELL1} \land \text{flagC1}) \lor (\text{CELL2} \land \text{flagC2}) \lor \ldots \lor (\text{CELLN} \land \text{flagCN})\)
- **HSTM** := \((\text{STM1} \land \text{flagS1}) \lor (\text{STM2} \land \text{flagS2}) \lor \ldots \lor (\text{STMP} \land \text{flagSN})\)
- **DESIGN** := \((\text{HSTM1} \lor \text{HSTM2} \lor \ldots \lor \text{HSTMQ}) \land \text{asynchConstraints}\)

- Formula **asynchConstraints** is defined on flag variables to restrict the interleaving execution manner.

- There are many subtle, but not technically difficult, details for encoding.
Basic Ideas (Informal) for Encoding an HSTM Design

Step 0 Formula (representing Initial States)

InitState := (var_{1_0} = initValueVar_{1}) \land (var_{2_0} = initValueVar_{2}) \land \ldots \land (var_{M_k} = initValueVar_{M})

Step k Formula (representing the set of states reachable at step k)

CELL_{k} := (cell.event_{k-1} \land cell.status_{k-1}) \implies (cell.action_{k} \land cell.status{\text{Transit}}_{k} \land untouchedVars_{k})

STM_{k} := (CELL_{1_k} \land flagC_{1_k}) \lor (CELL_{2_k} \land flagC_{2_k}) \lor \ldots \lor (CELL_{N_k} \land flagC_{N_k})

HSTM_{k} := (STM_{1_k} \land flagS_{1_k}) \lor (STM_{2_k} \land flagS_{2_k}) \lor \ldots \lor (STM_{P_k} \land flagS_{N_k})

DESIGN_{k} := (HSTM_{1_k} \lor HSTM_{2_k} \lor \ldots \lor HSTM_{Q_k}) \land asynchConstraints_{k}

- Formulas cell.action_{k} and untouchedVars_{k} are defined using variables that belong to either step k or step k-1.
- Hierarchical structure is represented by flag variables and changes of their values.
Basic ideas for accelerating BMC in Garakabu2

Avoid encoding all transitions in step_k by explicitly pre-traversing the state space.
Stateless traversing; only interested in transitions executable in step_k.

Saturation may happen for deep BMC bounds.
Generally becomes faster, especially for safety properties.
Outline

Garakabu2 and ZIPC – A general introduction

The SMT-based BMC approach in Garakabu2

• Towards practical usability for on-site software engineers (with demo of Garakabu2)
Step 1: Input ZIPC-HSTM designs into Garakabu2

- Syntax errors or specifications that could not be handled by Garakabu2 will be reported and pinpointed.

Garakabu2 – Image Demo
Step 2: Select STMs to be checked

- It is possible to select and check partial designs that are of interesting

- Partial selection that violates predefined rules are not allowed
  - E.g., it is not allowed to select a child STM, whose parent STM is not selected but STMs of other tasks are selected.

Garakabu2 – Image Demo
Step 3: Set initial/threshold values for variables

- Garakabu2 automatically reads initial values defined in ZIPC RAM file;
- It is possible to set the min/max values of variables (optional)

<table>
<thead>
<tr>
<th>Variable Types</th>
<th>Range Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
</tr>
<tr>
<td>Bool</td>
<td>False</td>
</tr>
<tr>
<td>Byte</td>
<td>-128</td>
</tr>
<tr>
<td>Char</td>
<td>-128</td>
</tr>
<tr>
<td>Short</td>
<td>-32768</td>
</tr>
<tr>
<td>Int</td>
<td>-2147483648</td>
</tr>
</tbody>
</table>
Step 4: Input properties to be checked

- Supported properties
  - Reachability of Invalid Cells;
  - General LTL properties;
  - Deadlock;
  - Range values violation;
  - HSTM-specific properties:
    - Correlation between status of different STMs

Garakabu2 – Image Demo
Specify LTL properties by patterns

- It is difficult to specify LTL properties correctly
  - Property in text:
    
    Before the bill exchanged for a previous session is taken, no new to-be-exchanged bills could be inserted into the machine.
  
  - Property in LTL:
    
    Before the bill exchanged for a previous session is taken, no new to-be-exchanged bills could be inserted into the machine.

- SpecEditor – under-development in AIST of Japan
  - Specify LTL properties with Dwyer’s LTL patterns
  - Specify LTL properties with graph drawing (AIST LTL Notations)

SpecEditor – Image Demo
Step 5: Read/Track the check results

- After checking, the results in
  - Black indicates no counterexamples (within the bound)
  - Red indicates a counterexample

- By clicking a check result in red
  - Trace of the counterexample could be illustrated in ZIPC environment,
  - By which to confirm the undesired behaviors that violate the property

Garakabu2 – Image Demo
Step 6: Read/Replay previous check results

- Previous check results are recorded for further confirmation.
  - Select the history management label,
  - Double-click a checking item to confirm the model, property, and results to help understand design revision history.

Garakabu2 – Image Demo
This is all Garakabu2 about.

Simple and Easy-to-use are what we expect!
Future work

- Accelerating BMC implemented in Garakabu2;
  - Further integrating explicit model checking techniques into BMC
    - Preprocessing with explicit traversing and applying explicit abstraction techniques, e.g., partial order reductions etc.

- Extending HSTM components checkable by Garakabu2;
  - E.g., Concurrent states are not checkable by current Garakabu2
    - According to comments from practical users of ZIPC and Garakabu2

- (Further) Implementing SpecEditor;

- ......
Questions and Comments?