On-the-fly model checking for C programs with extended CADP in FMICS-jETI

María del Mar Gallardo
Christophe Joubert
Pedro Merino
David Sanán

ICECCS 2007
Auckland, New Zealand

University of Málaga / GISUM
Technical University of Valencia / ELP
- Introduction
- The CADP environment
- C.Open and Annotator
- Integration with FMICS-jETI
- Conclusions
Introduction (I)

- Software systems becomes more difficult to analyze.
- Many tools appears to manage specification and programming languages in the context of formal methods.
- Tools integration becomes a challenge
  - Definition of intermediate languages.
  - Creation of environments to manage groups of tools.
CADP (I) - Introduction

- CADP is a toolbox for the analysis of communication protocols.
- CADP has as input language LOTOS and BCG.
- It internally works with LTSs (Labelled Transition Systems)
- A wide spread of tools:
  - Bisimulator
  - BCG_Min
  - Evaluator
  - ...
CADP (II) - Architecture

- Lotos
- Explicit LTS (BCG)
- Implicit LTS
- Caesar
- Caesar.ADT
- Caesar.Open
- Generator
- BCG_Open

- Bisimulation (Bisimulator)
- Minimization (BCG_Min)
- Verification (XTL)
- Visualization (BCG_Draw)
- Verification (Evaluator)
- Test (TGV)
- Simulation (OCIS, Simulator)
- Reduction (Reductor)
CADP (III) – Extending CADP

- CADP includes OPEN CAESAR, an interface for adding new modules to the toolbox.
- In that way, it is possible to add a new modules, well incorporating new tools to existing (as Annotator for control flow analysis) or extending the set of input languages (as C.Open for use C with the CADP tools)
Extending CADP

- **Lotos**
  - Bisimulation (Bisimulator)
  - Minimization (BCG_Min)
  - Verification (XTL)
  - Visualization (BCG_Draw)

- **Caesar**
  - Caesar.ADT
  - Caesar.Open

- **Explicit LTS (BCG)**
  - Generator

- **Implicit LTS**
  - BCG_Open

- **Verification (Evaluator)**
- **Test (TGV)**
- **Simulation (OCIS, Simulator)**
- **Reduction (Reducer)**
- **Annotator**

- **C Code**
  - C.Open
C. Open Architecture

C2XML

p1.C → p1.xml

rules.xml → Model

• C code Analysis
• Process Graph
• States and Labels impl.

XML2LTS

Implicit LTS source

graph.c  header.h  trans.m  trans.t

C compiler

Implicit LTS (Graph Module)
Annotator

- Annotator implements data flow algorithms by using boolean equation systems (BEESs).
- Annotator computes influence analyses detecting the relevant program variables in each control point.
- On-the-fly data flow analysis for:
  - Live/dead variables.
  - Very busy expressions.
  - Available expressions.
  - Reaching definitions.
- On-the-fly detection of variables preserving:
  - Reachability.
  - Assertions.
  - Temporal formulas.
  - API system calls.
Verifying software properties (I)

- We test the new CADP tools implementing the Peterson Mutual Exclusion algorithm.
- Two process access to a shared resource without conflict using only shared memory for communication.
Verifying Software Properties (II)

- We use Annotator, C.Open and Evaluator (a model checker which evaluates temporal properties) to check that the implementation satisfies the Mutual Exclusion criteria:
  - Mutual Exclusion.
  - Progress requirement.
  - Bounded waiting.
Verifying Software Properties (III) – Analysis Graph for the PME example

Diagram:
- C File
- XML File
- C File
- __Open
- C.Open
- T/F
- Evaluator
- μ Formula
- Annotator
- Generator
- BCG
- BCG_Min
- BCG
The remote execution of C.Open, Annotator and Evaluator is doing with only one SIB that remotely executes a SVL script.

Using only one SIB makes easier the construction of Service Logic Graphs as well as to use all the available tools in CADP.
Integrating CADP in FMICS-jETI (II) – SIB parameters

<table>
<thead>
<tr>
<th>Parameter Configuration</th>
<th>add Union</th>
<th>add Parameter</th>
<th>Clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATIC PARAMETER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class: java.lang.String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value: svi-path</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQUIRED PARAMETER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name: SVLFILE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class: de.unido.is5.eti.toolserver.InputFileReference</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description: SVL script file to be executed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQUIRED PARAMETER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name: INPUTFILE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class: de.unido.is5.eti.toolserver.InputFileReference</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description: Input files needed to the SVL execution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQUIRED PARAMETER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name: LOGFILE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class: de.unido.is5.eti.toolserver.OutputFileReference</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description: Log information about the SVL execution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQUIRED PARAMETER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name: OUTPUTFILE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class: de.unido.is5.eti.toolserver.OutputFileReference</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description: Output files used and generated during the SVL execution</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Service Logic Graph
Conclusions

- We have successfully integrated not only C.Open and Annotator but the set of tools included in CADP.
- To do that, we have constructed the SIB and SLG to access the CADP tools from the jABC client.
- The integration of the tools have been made in a very short time.