TGV
Génération de tests de conformité à partir de modèles formels

Thierry Jéron (INRIA / IRISA)
Wendelin Serwe (INRIA / LIG)
5ème Forum Méthodes Formelles, Toulouse, 16 juin 2015
TGV

Generation of Conformance Tests from Formal Models

Thierry Jéron (INRIA / IRISA)

Wendelin Serwe (INRIA / LIG)

5ème Forum Méthodes Formelles, Toulouse, 16 juin 2015
Conformance Testing

- Check conformance between
  - Formal specification \((S)\) as reference or oracle:
    Input/Output labeled transition system \((IOLTS)\)
  - Implementation under test \((IUT)\):
    a black box, interaction only via known points of control and observation \((PCO)\)
- IUT conforms to \(S\) if it passes tests
- Different approaches: online / offline / a posteriori

\[
\begin{align*}
\text{specification} & \quad \Rightarrow \quad \text{control} \quad ? \quad \text{implementation under test} \\
\text{observation} & \quad \Rightarrow \quad \text{control} \quad ? \quad \text{observation}
\end{align*}
\]
Online Conformance Testing

- Simultaneous execution of
  - the specification (\textit{tester}) $S$ and
  - the implementation under test $IUT$

- Synchronize control of $IUT$ with observation of $S$ (and vice versa)

- Stop when an error is found

---

**Diagram:**

- **specification**
  - control
  - observation
  - verdict (fail, stop)

- **implementation under test**
  - control
  - observation
Offline Conformance Testing

- **Test purpose**: functionality to be tested

- **Verdicts**:
  - **Fail**: IUT not conform to the specification
  - **Pass**: test purpose reached
  - **Inconclusive**: no error, but test purpose not reached

```
implementation
```

```
test case(s)
```

```
test purpose (selection directive)
```

```
control
```

```
observation
```

```
verdict (pass, fail, inconclusive)
```
A posteriori conformance testing
Generate execution traces
Validate traces with respect to the specification or expected properties
Background
Formal Model of Behavior

- For specification and implementation under test
- Input-Output Labeled Transition System (IOLTS) $(Q, A, \rightarrow, q_0)$
  - $Q$: enumerable set of states
  - $A = A_I \cup A_O \cup \{\tau\}$: transition labels (actions)
    - $A_I$: inputs, controllable by the tester, prefix “?”
    - $A_O$: outputs, observable by the tester, prefix “!”
    - $\tau$: internal action
  - $\rightarrow \subseteq Q \times A \times Q$: transition relation
- Other models: Mealy machines

Conformance Test Generation
Notions

- Execution, trace, run
- Quiescence ($\delta$): no further output from the IUT
  - outputlock (includes deadlock): wait for input
  - livelock: loop of internal actions
- Suspended trace: execution up to quiescence

Properties of a test suite (set of test cases)
- sound/correct: tests reject only a non-conform IUT
- exhaustive: rejection of all non-conform IUTs
- complete: sound and exhaustive
Conformance Relation

- Depends on the control and observation capabilities of the tester
- Many choices: isomorphism, bisimulation, testing equivalence, trace equivalence, ...
- Reasonable compromise (Jan Tretmans): \textit{ioco}
  “IUT \textit{ioco} S” if after each suspended trace IUT exhibits only outputs and quiescences present in S
ioco: Correct Examples

specification

implementation choice

implementation of a partial specification
ioco: Incorrect Examples

specification

forbidden output

forbidden quiescence
Test Selection

- Exhaustiveness unachievable in practice:
  Produce a “limit-exhaustive” suite of sound tests
- Tradeoff between test quality and cost/time
- Focus on “corner cases”
- Measure “coverage”

Different approaches

- Random (online testing)
- Domain specific knowledge (test purposes)
- Model-based (structural coverage criteria)
Online Testing: Example Case Study
FAME (Flexible Architecture for Multiple Environments)

CC-NUMA architecture for Bull's high-end servers based on Intel's Itanium-2
Focus on most critical, asynchronous parts

- Chipset components for an early prototype of FAME based on Itanium-1 ("Merced") processors:
  - CCS (Core Chip Set)
  - NCS (Network Chip Set)
- B-SPS / FSS (Fame Scalability Switch)
  - core of the FAME architecture
  - implements message routing and cache coherency protocol
  - contains several "units", which themselves contain "blocks"
Online Conformance Testing

- Various coverage criteria
  - Petri net transitions
  - LOTOS visible labels and their offers
- Combination of random and directed approaches
  - Random firing of tau transitions
  - History-based guidance to maximize coverage
Offline Testing with the TGV tool
Test Purpose

- IOLTS with the same actions as the specification
- **Accept** states to be reached by the test
- **Refuse** states to stop test execution (inconclusive)
- Deterministic
- Complete: each state offers all actions
Abstract Test Case

- IOLTS with verdict states (pass, fail, inconclusive)
- No internal actions
- Outputs = inputs of the specification/IUT
- Inputs = outputs of the specification/IUT + \{\delta\}
- From all states, a verdict is reachable
- Fail/inconclusive directly reachable only by inputs
- Input-complete: accepts all outputs of the IUT
- Controllable
  - no choice between two outputs or an input and an output
  - otherwise: complete test graph
- Requires refinement to connect to the IUT
Conformance Test Generation

property

\( \vdash \) (satisfies)

specification

\( \approx \) (conforms to)

implementation

\( \Rightarrow \)

\( \Rightarrow \)

test purpose (selection directive)

TGV test generation

test case(s)

verdict
TGV: advanced options

- Quiescence detection using two timers
  - TAC: no quiescence expected
    - timeout yields fail verdict
  - TNOAC: quiescence expected

- Postambles
  - reinitialisation of the IUT after passing the test purpose
  - pass-first verdict

- Hiding/Renaming

- Implicit completion of test purposes
Some Case Studies with TGV
PolyKid Multiprocessor Architecture

- PowerPC processors
- CC-NUMA memory model

lower level: SMP snoop-based cache coherence

higher level: loosely coupled directory-based cache coherence
PolyKid: Specification and Verification

- Several specifications developed
  - Polykid architecture: 4,000 lines of LOTOS
  - Cache coherency rules: 2,000 lines of LOTOS

- Validation by simulation and model checking on abstracted subsets (2,000 lines of LOTOS, 10 concurrent processes)

- Several problems (deadlocks, memory consistency violation, undocumented behaviours) found:
  - phase 1: 55 questions
  - phase 2: 20 questions, 7 serious issues
  - phase 3: 13 serious issues
PolyKid: Test Generation Results

- 75 tests (> 400 states each) generated in 1 man.month
- Development of tools for automated test execution
- Test execution in less than 20 hours
- 5 new bugs discovered in VHDL design


http://cadp.inria.fr/case-studies/98-c-ccnuma.html

http://cadp.inria.fr/case-studies/00-c-polykid.html
Diagnosis System of Vehicles

- Model Transformation: UML statecharts to LOTOS
- Focus on automation of test purpose generation

Conformance Test Generation
Diagnosis System of Vehicles

- Lengthy test cases due to high branching factor and search order (depth-first rather than breadth-first)
- Coverage criteria for the UML statecharts
- Redundancies in test cases


- [http://cadp.inria.fr/case-studies/09-j-test-automotive.html](http://cadp.inria.fr/case-studies/09-j-test-automotive.html)
Further Case Studies with TGV

- **DREX** (military version of the ISDN D protocol)

- **SSCOP (Service Specific Connection Oriented Protocol) / FranceTelecom R&D**
  - [http://link.springer.com/chapter/10.1007/978-0-387-35516-0_1](http://link.springer.com/chapter/10.1007/978-0-387-35516-0_1)

- **Conference Protocol**
  - [http://cadp.inria.fr/case-studies/00-g-conference.html](http://cadp.inria.fr/case-studies/00-g-conference.html)

- **Agent-Based Online Auction**
  - [http://cadp.inria.fr/case-studies/03-f-auction.html](http://cadp.inria.fr/case-studies/03-f-auction.html)

- **Teleoperation**
  - [http://cadp.inria.fr/case-studies/03-g-teleoperation.html](http://cadp.inria.fr/case-studies/03-g-teleoperation.html)

- **Fault-based testing of communication protocols**
  - [http://cadp.inria.fr/case-studies/04-g-fault-based-testing.html](http://cadp.inria.fr/case-studies/04-g-fault-based-testing.html)

- **Session Initiating Protocol**
  - [http://cadp.inria.fr/case-studies/07-b-sip.html](http://cadp.inria.fr/case-studies/07-b-sip.html)

- **AMBA 4 ACE Cache Coherency**: next talk by Massimo Zendri (STMicroelectronics)

- See also [http://cadp.inria.fr/case-studies](http://cadp.inria.fr/case-studies)
Trace Validation
FAME (Flexible Architecture for Multiple Environments)

CC-NUMA architecture for Bull's high-end servers based on Intel's Itanium-2
Trace Validation: Former Approach

Goals

- find bugs in traces of bus transactions
- measure coverage of test effort

- Large, complex traces (> 10,000 nested bus transactions)
- Costly development of a dedicated “verifier”
- What about correctness of the “verifier”?
Trace Validation: Formal Approaches

- **Goal:** reuse the LOTOS specification to check traces
  - **BISIMULATOR:** trace inclusion
  - **EXHIBITOR:** regular expression matching
  - **EVALUATOR:** temporal formula satisfaction

- **What about coverage?**

---

Conformance Test Generation
Trace Validation with Coverage

- μ-calculus formulas generated from state/transitions tables
- **Markers** indicating if a formula is **activated** by a given trace
- Formula activated by no trace → more traces needed for coverage
- **Functional coverage** (with respect to the specification)
- Different from structural coverage (wrt Verilog design)
Trace validation with coverage

Main results
- Major bug: ambiguity of informal specification (also found by the “Verifier” of the former approach)
- Collision traces (∼ 24,000 transactions, 130 Mbytes): OK
- Interface traces (761 properties verified, 216 not covered): 2 missing tests added in 2001
- Directory traces (518 properties verified, 196 not covered): 1 missing test added in 2001

Used at every revision: official part of design methodology

Performance
- 7.4 millions of model checking jobs
- 23 hours (PC, Pentium III 700 MHz, 1 GB RAM)

http://dx.doi.org/10.1007/s100090100044
Conclusion
Related Work and Tools

- Axini Test Manager (http://www.axini.com/?lang=en)
- Agatha (http://dx.doi.org/10.1007/3-540-36577-X_43)
- FSM-based test generation (http://dx.doi.org/10.1049/sej.1991.0040)
- Java PathFinder (http://babelfish.arc.nasa.gov/trac/jpf/wiki)
- JTorX (https://fmt.ewi.utwente.nl/redmine/projects/jtorx/wiki)
- SpecExplorer (http://research.microsoft.com/en-us/projects/specexplorer/)
- TestComposer (http://www.canamsoftware.com/Products/CAGenSolutions/TestComposer%E2%84%A2/Overview.aspx)
- TestGen (http://freecode.com/projects/testgen)
- Test generation based on model-checking: http://dx.doi.org/10.1007/s100090050044
- UPPAAL Co√er (http://www.hessel.nu/CoVer/)
- UPPAAL TRON (http://people.cs.aau.dk/~marius/tron)
- ...
Conclusion

- Model-based testing applicable to various domains
- Large state spaces manageable
- Different approaches: online, offline, trace validation
- Design of test purposes crucial for offline testing
  - easier if requirements available
  - refinement needed to enable test case generation
  - control length of test cases
- Guarantees: limit-exhaustive suite of sound tests
- Orthogonal to coverage based techniques
- Extensions: symbolic, time, ...
References


For more information: [http://cadp.inria.fr](http://cadp.inria.fr)