The paper

- Claude Jard, Thierry Jeron
- Published online 2004, Springer-Verlag
Test generation with verification technology

automatic synthesis of conformance test cases from a formal specification of a (non-deterministic) reactive system

"on-the-fly" synthesis
Outline

1 Basics
   - TGV overview
   - IOLTS
   - Formal test purposes

2 Principles and algorithms
   - Synchronous product
   - Extracting visible behaviour
   - Test selection
   - Controllability
   - On-The-Fly synthesis

3 The Tool

4 Conclusion
Definitions

- **Test case**: testing particular functionality
- **Test suite**: a set of tests
- **Test**: outputs are stimuli for IUT, inputs are observations of IUT’s outputs
- **Fail verdict**: IUT is rejected
- **Pass verdict**: IUT is accepted
- **Inconclusive verdict**: correct behaviour is observed but test purpose can’t be reached
- **Soundness**: test cases only reject non-conformant IUT’s
- **Exhaustiveness**: all non-conformant IUT’s are rejected
Functional view
IOLTS

- $M = (Q^M, A^M, \rightarrow^M, q_0^M)$
- $A^M = A^M_I \cup A^M_O \cup I^M$
- $a(i) \in A^M \setminus I^M$
- $\tau(i) \in I^M$
- **Fireable actions:** $\Gamma(q)$
- **Transitions:** $\xrightarrow{a}$
- **Visible behaviour:** $\Rightarrow$
- **Input:** $?a$
- **Output:** $!x$
- **$\text{det}(M)$:** $M$ without internal actions
IOLTS cont.
Quiescence

- Deadlock: $\Gamma(q) = \emptyset$
- Output quiescence: $\Gamma(q) \subseteq A_f^M$
- Livelock
  - $\text{deadlock}(M) \subseteq \text{outputlock}(M)$
  - $\text{quiescent}(M) = \text{livelock}(M) \cup \text{outputlock}(M)$
Suspension automaton $\Delta(S)$
\[ \det(\Delta(S)) = S^{VIS} \]
Test purpose

- $TP = (Q^{TP}, A^{TP}, \rightarrow_{TP}, q_{0}^{TP})$
- $Accept^{TP}$: select target behaviour
- $Refuse^{TP}$: cut down specification
- Allows efficient test selection on-the-fly
- Smaller than specification but complete (...?)
Test purpose cont.
Principles and algorithms
Synchronous product

- Mark behaviours of $S$ by $Accept$, $Refuse$
- Accepted behaviour of SP are accepted behaviours of $S$ by TP
\[ S \times TP = SP \]
$SP^{VIS} = det(\Delta(SP))$
Test selection

- Extracting test case by selection of accepted behaviours
  \[ A^{CTG}_O \subseteq A^{VIS}_I \]
  \[ A^{CTG}_I = A^{VIS}_O \]
- **L2A**: all states that lead to accept
- **Pass**: \( \text{Accept}^{VIS} \)
- **Inconc**: direct successors of states in L2A by output in \( SP^{VIS} \)
- **Fail**: else
- **Result**: CTG (Complete Test Graph)
- **TGVLoop**, based on Tarjan’s algo \( (O(n), S(n)) \)
Test selection
Controllability

- Extract a controllable subgraph of CTG
- Get rid of choices between I/O (pruning)
- This can happen during TGVLoop (partially)
A test case
On-The-Fly synthesis

- perform lazy construction of subgraphs of $S$, $SP$, $SP^{VIS}$
- needed functions
  - init
  - fireable
  - succ
  - Comparison function
  - Function to compute membership of Accept/Refuse
- goal: reduce size of graphs
Several software layers
- Communicate through APIs
- Each one is simulation of IOLTS (allows graph traversal)
- Each level implements one of the algorithms
- Output: test cases in TTCN or graph formats (.aut, .bcg)
- Can be used to verify manually created test cases
- SunOS 5, Windows XP, Linux
Architecture
Timers
Conclusion

- TGV can synthesize tests from industrial size specs
- Drawback: manual creation of test purpose
- Still better than manual test case creation
- Future: distributed tests, improvements of algorithms