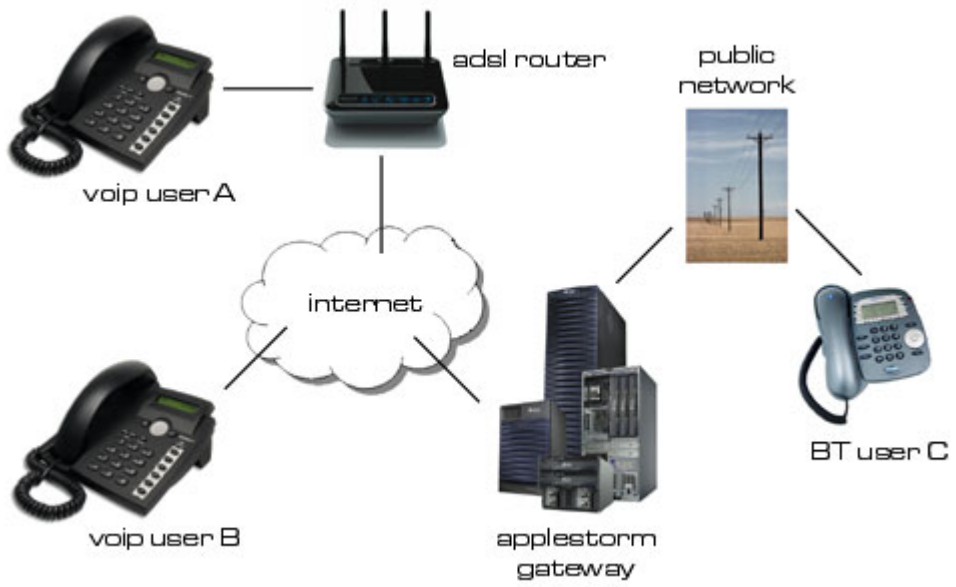
Symbolic analysis of EFSM models for test generation using Z3

Marko Kääramees
Tallinn University of Technology

1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Model-based black-box conformance testing

- System can communicate to its environment according to specification/protocol
- Testing @ interface
- Model represents the correct behaviour of the IUT
  - Non-deterministic models considered
- Embedded systems, services, communication devices

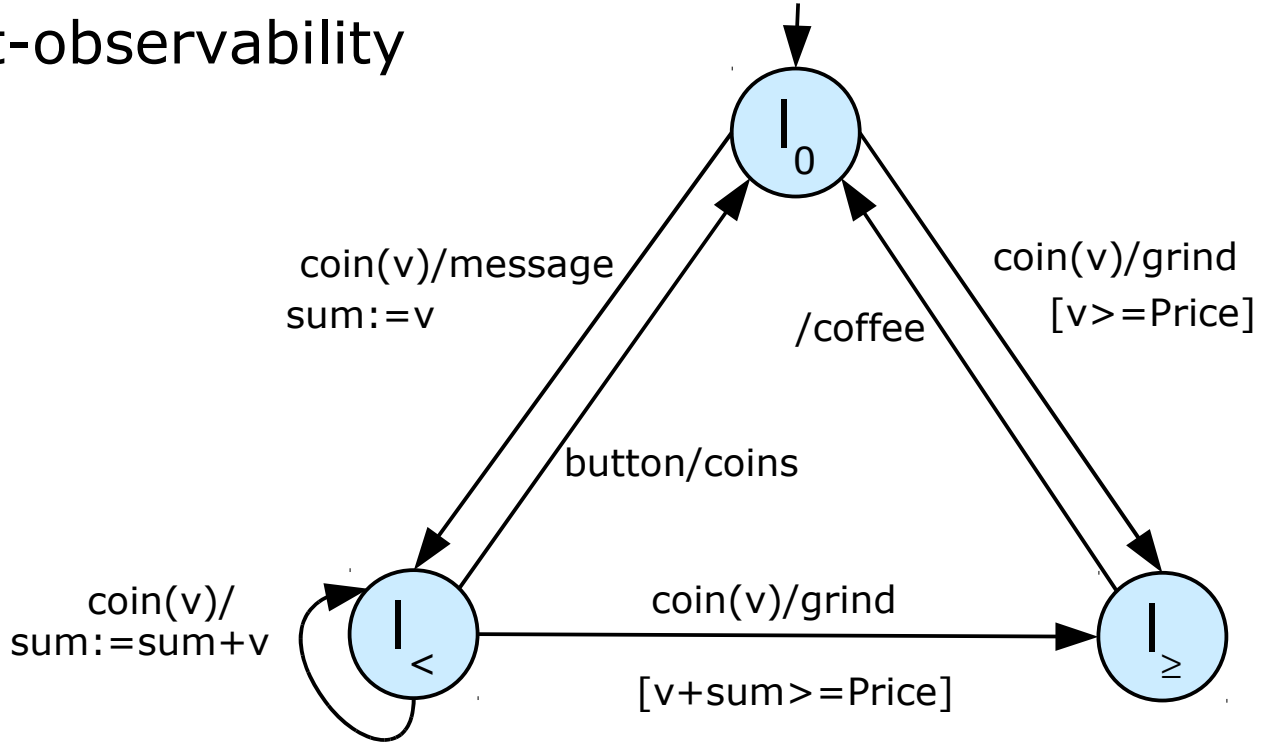


Modelling formalism: I/O-EFSM

- Set of locations L and initial location l_0
- Set of variables $X = Xs \cup Xi \cup Xo$ consisting of state, input, and output variables
- Domain constraint D is a predicate on variables X
- I and O are the sets of input and output labels every label may have associated a tuple of parameters $(x_0 \ldots x_n)$
- E is a set of edges with
  - Source location l and target location l'
  - Guard g
  - Input port $i(\bar{x})$ and output port $o(\bar{x})$ with their actual parameters $\bar{x} \in Xi \cup Xo$
  - Updates U
- Background first order theory Th

Non-deterministic model

- Practical non-determinism
  - Input-nondeterminism
  - Output-observability



Testing non-deterministic systems

- A test suite cannot be represented by a finite set of test cases
  - Several different outputs are correct for an input
  - Next input depends on the behaviour of IUT

- Symbolic test strategy
  - The strategy must choose an input for a current state of the IUT to lead it towards some of the test goals.
    Strategy is a function from state and test goal to input

$$St: S \times G \rightarrow I$$

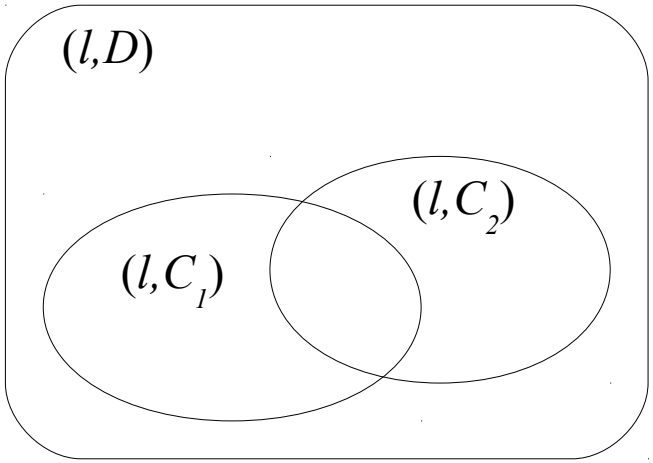
  - State space represented by an I/O-EFSM model is usually very large or infinite. Symbolic representation is needed
  - Strategy must be efficient
    - Extensive search and planning is not possible on-line
    - Industrial requirements: 10-100 ms for each step

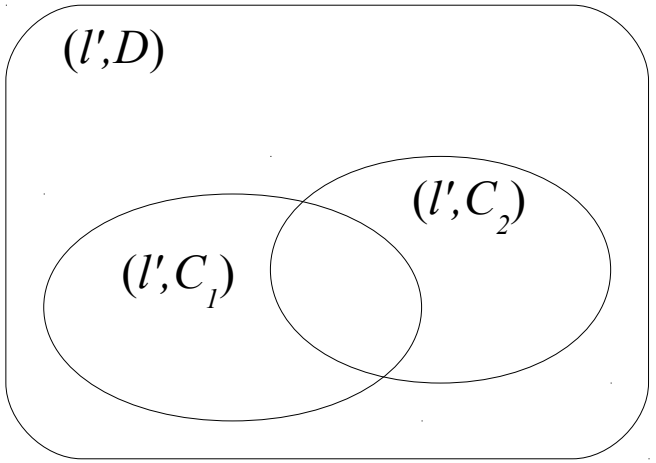
Expressing test goals

- Several usual coverage criteria
  - All transitions
  - Pairs of transitions
  - Guard border conditions
- Expressing test goals using *traps*
  - A *trap* is a pair of an edge and predicate on state and input variables
- Can express
  - transition coverage
    every edge has a trap
  - transition sequence
    trap condition with reference to other traps
  - repeated pass using auxiliary variable
    trap condition with reference to auxiliary variables

Symbolic state representation

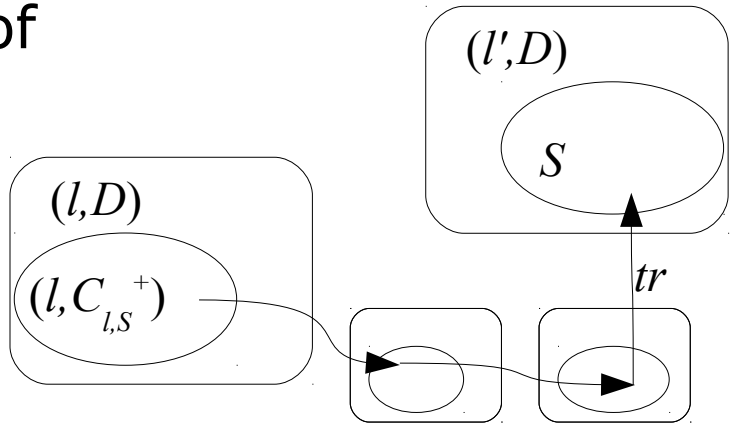
- State $s = (l, \alpha)$ is a pair of a location and assignment to state variables X_s
- Sybolic state $S = (l, C)$ is a pair of a location and a constraint. The constraint is a formula on state variables X_s
  - A constraint represents a set of assignments
  - Locations are represented explicitly





Symbolic representation of reachability

- We can represent the reachability of traps by the following constraints and distances
- $C^{+}_{l,tr}$ – there is a run of automaton that starts from state $(l, C^{+}_{l,tr})$ and ends with a transition that covers the trap tr. $L^{+}_{l,tr}$ is the length of the longest of such runs.
- $C^{+}_{l,tr}$ is a quanifier free formula on state variables X_s
- Reachability constraints are calculated by repeated application and combination of pre-image calculation procedure



Symbolic representation of runs

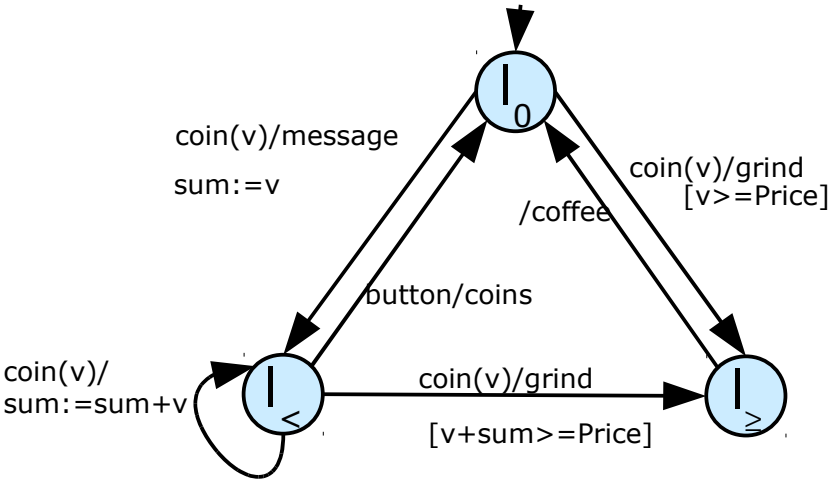
- Guiding constraints are needed for finding an input and transition that leads to the chosen trap

- $C^{g}_{e,tr}$ – constraint on state and input variables that an edge *e* is the initial transition of the shortest run that ends with a transition that cover the trap tr.

  A guarding constraint for the

  edge $l_0 \rightarrow l_{\geq}$ is $v >= Price$

  edge $l_0 \rightarrow l_{<}$ is $v < Price$



Reachability analysis

- Breath-first backwards symbolic traversal of the automaton starting from the trap edge and condition

initialise C to *false*, L to 0

$C^{+}_{tr,tr} = guard_{tr} \wedge condition_{tr}$

while fixpoint, initial state or search *depth* is reached

  for each state s on the *depth* level do

    $C^{+'}_{l,tr} := \exists I: \vee C^{+}_{e,tr}$ // e - *edge* leaving from l; I - *input*

    if weaker($C^{+'}_{l,tr}$, $C^{+}_{l,tr}$) // C^{*}_{s} changed

      $C^{+}_{l,tr} := \text{compact}(C^{+}_{l,tr} \vee C^{+'}_{l,tr})$

      $L^{+}_{l,tr} := depth$

      for each transition e coming to l

        $C^{+}_{e,tr} := guard_{e} \wedge wp(update_{e}, C^{+}_{l,tr})$

        $C^{g}_{e,tr} := \text{compact}(C^{g}_{e,tr} \vee (C^{+}_{e,tr} \wedge \neg C^{+}_{source(e),tr}))$

Compacting symbolic representation

- Compat representation of the symbolic states is crucial to the efficiency of the application
- > 90 % of the time of the algorithm spent in *compact*()
- Uses a combination of Z3 simplification functions

```
ELIM_QUANTIFIERS = true
STRONG_CONTEXT_SIMPLIFIER = false
CONTEXT_SIMPLIFIER = false
simplify()
STRONG_CONTEXT_SIMPLIFIER = true
simplify()      // double strong simplification
simplify()
STRONG_CONTEXT_SIMPLIFIER = false
CONTEXT_SIMPLIFIER = true
simplify()
```

Simplification parameter tuning

- Have tried to play with different tuning parameters:

  ARITH_PROCESS_ALL_EQS = true

  ARITH_EQ_BOUNDS = true

  ARITH_ADAPTIVE = true

  ARITH_PROP_STRATEGY = #

  ELIM_BOUNDS = TRUE

  FWD_SR = true

  PROPAGATE_BOOLEANS = true

- No additional reduction
- Or a little reduction in expense of much longer computation time on some examples

Convergence checking

- Check if the newly generated symbolic state weakens the previous symbolic state (constraint) for the location
  - Checked using satisfiability check of the implication

$$\text{weaker}(C^{+'}_{l,tr}, C^{+}_{l,tr}) \equiv \text{SAT}(\neg(C^{+}_{l,tr} \Rightarrow C^{+'}_{l,tr}))$$

- Easier than compacting
  - but regular compacting of relevant components makes it feasible
  - Done on un-compacted constraints and only weakening constraints are compacted

On-line test generation

$l = l_0$ //start from the initial location

while exist uncovered traps

select nearest reachable trap *tr*

with $C^+_{l,tr}[X_s/\alpha]$ satisfiable and minimal $L^+_{l,tr}$

input := select input for moving towards trap *tr*

by finding a satisfying model for $C^g_{source(l),tr}[X_s/\alpha]$

or doing constraint solving

output := communicate_SUT(*input*)

simulate *input/output* on model and determine next location *l*

if the output of does not conform to the model

stop(test_failed)

end while

stop(test_passed)

Telecom Billing Case-Study

- Model: 13 locations, 47 transitions
- Path length to trap from initial state: 189
- Size of ASCII representation of the strategy: 34MB
- Time for test generation (symbolic analysis + input) [1 GHz Opteron]
  - 66 minutes for constraint generation to initial state
  - < 2 minutes with constraint generation to depth 10 and heuristic on-line test generation
  - Average time for a *compact*() operation ~1.3 sec
  - Average time for a *SAT*() operation ~0.09 sec

Past/Ongoing/Future work

- Alternatives considered
  - RedLog for compacting, quantifier ellimination
  - CVC3 for SAT solving
- Invariant discovery
- Extend modelling frameworks
  - Hierarchical automaton (subset of Statecharts)
  - Distributed systems
  - Timed systems
- Wider background theory and modelling language
  - Arrays
  - Recursive data-types

Conclusions

- Thanks to all developers and supporters of Z3
- Issues
  - Some encountered
  - None at the moment
- Feature requests
  - Better documentation for tuning parameters
    Would be nice to know (reference) what is behind each parameter to be able to suitability for the application
  - Interface for custom rewriting/simplification rules
- Conclusions
  - Z3 does a good job and has a quite reasonable set of defaults for the parameters and heuristics