Extended Finite-State Machine Induction using SAT-Solver

Vladimir Ulyantsev, Fedor Tsarev
ulyantsev@rain.ifmo.ru, tsarev@rain.ifmo.ru

St. Petersburg National Research University of IT, Mechanics and Optics
Computer Technologies Department

14th IFAC Symposium on Information Control Problems in Manufacturing
May 25, 2012
Automata-based Programming

• Programs with complex behavior should be designed using automated controlled objects
Extended Finite-State Machine and Test Scenarios

- **EFSM:**
  - input events
  - input Boolean variables
  - output actions

- Test scenario is a sequence of triples \(<e, f, A>\)
  - \(e\) – input event
  - \(f\) – guard condition – Boolean formula of input variables
  - \(A\) – sequence of output actions

- EFSM on the picture complies with \(<A, \neg x, (z2)> <A, x, (z1)>\)
- EFSM on the picture does not comply with \(<A, x, (z2)>\)
EFSM Example

- Alarm clock
- Four events
  - H – button “H” pressed
  - M – button “M” pressed
  - A – button “A” pressed
  - T – occurs on each time tick
- Two input variables
- Seven output actions
Goal of the Work

• Focus on automata-based programs with only one automated controlled object

• Given:
  – Set of test scenarios (Sc)
  – Number of EFSM states ($C$)

• Need to find a EFSM with $C$ states complying with all scenarios
Works of Other Authors

• DFA and FST induction with genetic algorithms:

• DFA induction using SAT-solvers
Main Idea

• Each scenario is similar to “linear” automaton

• Scenarios “coloring”
  – Each “state” of each scenario is to be mapped to some state of resulting EFSM
  – States of resulting EFSM <-> colors
1. Scenarios tree construction

2. Consistency graph construction

3. Boolean CNF-formula construction

4. SAT-solver invocation

5. EFSM construction from satisfying assignment
Precomputations

• For each pair of guard conditions from scenarios compute:
  – If they are same as Boolean functions
  – If they have common satisfying assignment

• Time complexity:
  – $O(n^2 2^{2m})$ where $n$ is total size of scenarios, $m$ is maximal number of input variables occurring in guard condition (in practice $m$ is not greater than 5)
1. Scenarios Tree Construction

- Similar to syntax tree construction algorithm
- If contradiction is found, process is terminated
2. Consistency Graph Construction

- Vertices are same as in scenarios tree
- Two vertices are connected by an edge if there is a sequence telling them apart
- Sets of inconsistent vertices are constructed for each tree vertex starting from leaves using dynamic programming
3. Boolean CNF-formula construction (1)

• Variables:
  – $x_{v,i}$ – is it true that vertex $v$ has color $i$
  – $y_{a,b,e,f}$ – is it true that in resulting EFSM exists a transition from state $a$ to state $b$ labeled with event $e$ and formula $f$
3. Boolean CNF-formula construction (2)

- Types of clauses:
  - \((x_{v,1} \lor \ldots \lor x_{v,C})\) – each vertex should be colored with some color
  - \((\neg x_{v,i} \lor \neg x_{v,j})\) – no vertex can be colored with two colors simultaneously
  - \((\neg x_{v,i} \lor \neg x_{u,i})\) – no pair of inconsistent vertices can be colored with same color
  - \((\neg y_{i,j,e,f} \lor \neg y_{i,k,e,f})\) – there is no more than one transition from each state of resulting EFSM marked with same event \((e)\) and Boolean formula \((f)\)
3. Boolean CNF-formula construction (3)

• Types of clauses:
  
  – \((y_{ij,ef} \lor \neg x_{vi} \lor \neg x_{uj})\) – each edge of scenarios tree must be present in resulting EFSM
  
  – \((\neg y_{ij,ef} \lor \neg x_{vi} \lor x_{uj})\) – vertex colors should not contradict with edges of scenarios tree
4. SAT-solver invocation

• CNF-formula is represented using DIMACS CNF format

• We use cryptominisat SAT-solver – winner of SAT RACE 2010
5. EFSM construction from satisfying assignment (1)

- Scenarios tree coloring
- Each vertex gets a color according to $x_{v,i}$ values
5. EFSM construction from satisfying assignment (2)

• All vertices with the same color are merged
5. EFSM construction from satisfying assignment (3)

- Coloring is not necessarily unique
Experiments

- First experiment – EFSM for alarm clock:
  - 38 scenarios of total length 242
  - Running time – 0.25 seconds
  - Genetic algorithm ~ 4 minutes
Second experiment

- Random EFSM $A_1$ with $n$ states generation
- Test scenarios generation (random paths in $A_1$) with total size $l$
- EFSM $A_2$ with $n$ states induction
- “Forward check”
  - $1000n$ random scenarios of length $4n$ are generated from $A_1$
  - $A_2$ is checked against each of these scenarios
  - The part of scenarios $A_2$ complies with is recorded
- 1000 runs for each $n$ and $l$
Median execution time
Median “forward check” percent

[Graph showing the median "forward check" percent for different numbers of states (5, 10, 15, and 20 states) against the summary scenerios size. The graph illustrates the percentage of forward checks as the number of states and summary scenerios size increase.]
Future work

- Use CSP-solver to fix errors in scenarios
- Use Ant Colony Optimization Algorithms for EFSM induction (ANTS’12)
- Negative scenarios
- Verification
Results

• A method for EFSM induction based on reduction to SAT problem was proposed
• It was tested and proved to be much faster than genetic algorithm for the same problem
Thank you!

Extended Finite-State Machine Induction using SAT-Solver

Vladimir Ulyantsev