A Two Steps Test Suite Generation Approach based on Extended Finite State Machines

Ana Ţurlea

University of Bucharest, Romania

#fromBucharest
2017

Outline

- Problem Statement
- Proposed Strategy
- Experiments
- Conclusions
- Future work

Problem Statement

- **Input**: the specification of a system modeled as an EFSM
- **Output**: test suite - a set of feasible paths that cover all transitions and input data that trigger each path.
- **Approach**: a two steps algorithm
  - A multi-objective Genetic Algorithm for the test suite generation
  - A Hybrid Genetic Algorithm to search for test data

Extended Finite State Machine (EFSM)

An EFSM is a 6-tuple (S, s_0, V, I, O, T)
where:

- S is a non empty set of logical states;
- $s_0 \in S$ is the initial state;
- V is the finite set of internal variables;
- I and O are the set of input and output interactions, respectively;
- T is the finite set of transitions.

Extended Finite State Machine (EFSM)

An EFSM is a 6-tuple (S, s_0, V, I, O, T) where:

- S is a non empty set of logical states;
- $s_0 \in S$ is the initial state;
- V is the finite set of internal variables;
- I and O are the set of input and output interactions, respectively;
- T is the finite set of transitions.

A transition $t \in T$ has:

- start and end states
- input parameters
- guards
- a computational block (a method) - assignments and output statements.

Extended Finite State Machine (EFSM)

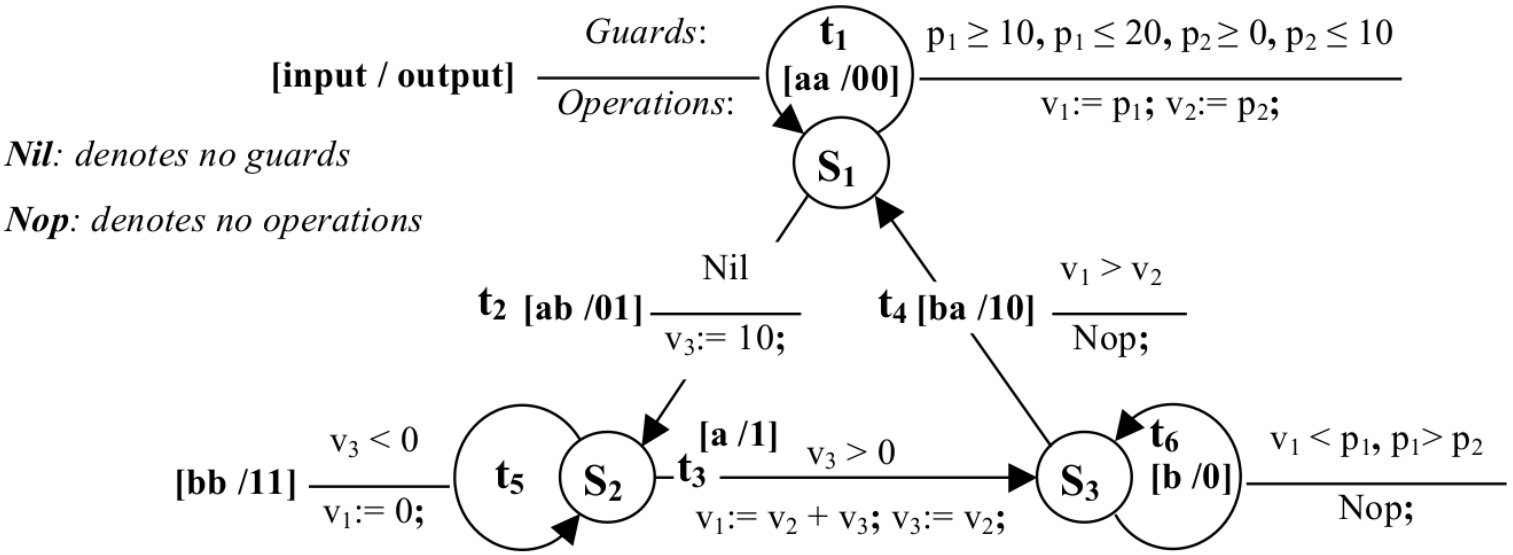

Figure 1: An EFSM example

Extended Finite State Machine (EFSM)

$$p = S_1 \xrightarrow{f_1[g_1]} S_2 \xrightarrow{f_2[g_2]} \ldots S_m \xrightarrow{f_m[g_m]} S_{m+1}$$

A path - a sequence of adjacent transitions;
Feasible path - there exist values for the input parameters to satisfy guards and to trigger all transitions for that path; infeasible path, otherwise.

Two steps algorithm:

- **Step 1:** feasible transition paths that cover all transitions using a multi-objective algorithm with 2 objectives:
  - first objective - transition coverage;
  - second objective - a feasibility metric based on transition dependences;
- **Step 2:** generate input data that validates the guards and triggers all transitions.
  - a genetic algorithm combined with a local search method.

Step 1 - Multi-objective Algorithm

Individuals Encoding:

- a chromosome - set of paths - variable length;
- a path - a sequence of integers, each one defining a transition;
- using the encoding for paths proposed by Kalaji et al. we ensure that every gene defines a valid path.

Kalaji, A.S., Hierons, R.M., Swift, S.: An integrated search-based approach for automatic testing from extended finite state machine (EFSM) models. (2011)

Step 1 - Multi-objective Algorithm

The Mutation Operator:

1. add a random generated gene
2. replace a gene from a random point
3. remove a random selected gene
4. replace a value for a random gene from a random index
5. exchange materials between two random genes

Step 1 - Multi-objective Algorithm

The Mutation Operator:

1. add a random generated gene
2. replace a gene from a random point
3. remove a random selected gene
4. replace a value for a random gene from a random index
5. exchange materials between two random genes

The Crossover Operator:

1. identify one gene in each parent at the same index and interchange genes at this position.
2. exchange materials between two genes

Step 1 - Multi-objective Algorithm

Objective Functions:

1. Transition coverage
   - if all transitions are covered - return the total number of transitions (used by all paths)
   - otherwise we return INF.

Step 1 - Multi-objective Algorithm

Objective Functions:

1. Transition coverage
   - if all transitions are covered - return the total number of transitions (used by all paths)
   - otherwise we return INF.
2. Feasibility metric - sum all the feasibility values for each path
   - the metric used by Kalaji.
   - guides the search towards transition paths that are likely to be feasible using dataflow dependencies among the transitions.

Kalaji, A.S., Hierons, R.M., Swift, S.: An integrated search-based approach for automatic testing from extended finite state machine (EFSM) models. (2011)

Step 1 - Multi-objective Algorithm

NSGA-II (Non-dominated Sorting Genetic Algorithm)

- GA with a strong strategy.

Step 1 - Multi-objective Algorithm

NSGA-II (Non-dominated Sorting Genetic Algorithm)

- GA with a strong strategy.
- after each evolution:
  - the algorithm sorts the parent population and the offspring population
  - create some fronts - non-dominance among individuals.
- apply crowding distance

Step 1 - Multi-objective Algorithm

NSGA-II (Non-dominated Sorting Genetic Algorithm)

- GA with a strong strategy.
- after each evolution:
  - the algorithm sorts the parent population and the offspring population
  - create some fronts - non-dominance among individuals.
- apply crowding distance
- select the solutions with greater crowding distance.

Step 1 - Multi-objective Algorithm

NSGA-II (Non-dominated Sorting Genetic Algorithm)

- GA with a strong strategy.
- after each evolution:
  - the algorithm sorts the parent population and the offspring population
  - create some fronts - non-dominance among individuals.
- apply crowding distance
- select the solutions with greater crowding distance.
- optimize each individual removing redundant paths and transitions (*)

Step 2: Hybrid Genetic Algorithm for Test Data Generation

- input: a path

Step 2: Hybrid Genetic Algorithm for Test Data Generation

- input: a path
- output: a list of values corresponding to all parameters of the methods, as they appear on that path

Step 2: Hybrid Genetic Algorithm for Test Data Generation

- input: a path
- output: a list of values corresponding to all parameters of the methods, as they appear on that path
- a chromosome - $x = (x_1; x_2; ...; x_n)$
- fitness function: $fitness = approach\_level + normalized\_branch\_level$

Step 2: Hybrid Genetic Algorithm for Test Data Generation

- input: a path
- output: a list of values corresponding to all parameters of the methods, as they appear on that path
- a chromosome - $x = (x_1; x_2; ...; x_n)$
- fitness function: $fitness = approach\_level + normalized\_branch\_level$
- A solution is a chromosome with fitness function 0 that triggers all transitions of the selected path and validates the guards of each transition.

Step 2: Hybrid Genetic Algorithm for Test Data Generation

- At each evolution - select best chromosome

Step 2: Hybrid Genetic Algorithm for Test Data Generation

- At each evolution - select best chromosome
- Apply local search for best chromosome using AVM

Step 2: Hybrid Genetic Algorithm for Test Data Generation

- At each evolution - select best chromosome
- Apply local search for best chromosome using AVM
- Find a better chromosome? Add new chromosome to population and continue with the next generation.

Step 2: Hybrid Genetic Algorithm for Test Data Generation

- At each evolution - select best chromosome
- Apply local search for best chromosome using AVM
- Find a better chromosome? Add new chromosome to population and continue with the next generation.
- No improvement? Continue with next generation.

Step 2: Hybrid Genetic Algorithm for Test Data Generation

- At each evolution - select best chromosome
- Apply local search for best chromosome using AVM
- Find a better chromosome? Add new chromosome to population and continue with the next generation.
- No improvement? Continue with next generation.

More details - previous approach: *Turlea, A., Ipate, F., Lefticaru, R.: A hybrid test generation approach based on extended finite state machines. SYNASC, 2016*

Experiments

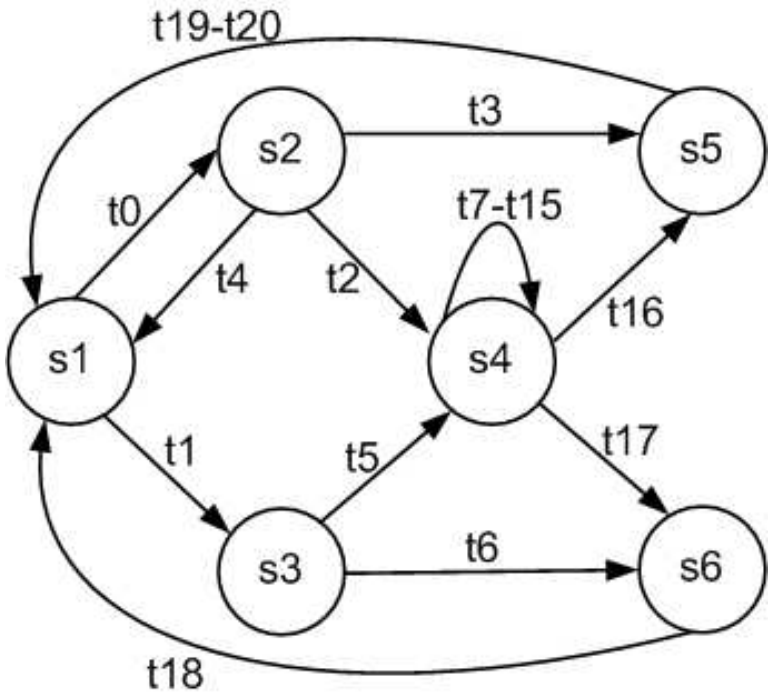

Figure 2: Class 2 transport protocol EFSM model

Experiments

Complexity	Path
4	[t0; t4; t0; t3; t20; t1; t5; t16]
18	[t1; t5; t9; t15]
34	[t1; t6; t18; t0; t4; t0; t2; t11; t12]
81	[t0; t3; t19; t1; t5; t10; t11; t7; t9; t17]
182	[t0; t4; t0; t2; t10; t13; t13; t14; t13; t8]

Table 1: Example of transition coverage set for the Class 2 Transport Protocol EFSM

Experiments

Path	Values
1	[57; -112; -64; 115; -23; 216; -138]
2	[168; 133; 54; 133]
3	[221; 194; 11; 179; -21; -89; -242; 245; -198; 8]
4	[-64; 229; 17; -153; 206; -213; 31; -116; 121; -34]
5	[15; 8; -111; 21; -229; 111; 20; 194; -55; -131; -11; 64; 65; 0]

Table 2: The input values needed to trigger the paths from Table 1

Conclusions

- two steps algorithm for test suite generation for EFSMs
- customized multi-objective algorithm to generate a transition coverage
- hybrid genetic algorithm to generate the input data that trigger each path

Conclusions

- two steps algorithm for test suite generation for EFSMs
- customized multi-objective algorithm to generate a transition coverage
- hybrid genetic algorithm to generate the input data that trigger each path
- This approach is different from the existing methods:
  - customizing the genetic operators,
  - optimizing the solutions by deleting redundant paths (all transitions are already covered)
  - shorten the paths - with smaller complexity
  - shorten the length of the set of paths
  - using the combination of genetic algorithms and local search methods

Future work

- Test on more EFSMs;
- Compare results with other model based techniques;
- Try other methods for the second step of the algorithm;
- Try to improve the multi-objective algorithm.


THANK
YOU