Vol.2010-SLDM-147 No.4
2010/11/29

EFSM-based Weight-oriented Concolic Testing for Embedded Software

Giuseppe Di Guglielmo,[†1,†2] Masahiro Fujita,[†1,†2]
Franco Fummi,[†3] Graziano Pravadelli[†3]
and Stefano Soffia[†3]

In the context of the model-based design paradigm, this paper introduces a new concolic testing approach for embedded software modeled via extended finite state machines (EFSMs). It aims at overcoming the limited width of the search based on concrete simulation and the limited depth of the symbolic execution, by interleaving long-range constraint-based heuristics and a symbolic multi-level backjumping technique. The combined approach is based on a weight-oriented dependency analysis over EFSMs which permits to achieve a high controllability of EFSM transitions. The effectiveness of the proposed framework has been evaluated on several case studies.

1. Introduction

With the increasing popularity of model-based design for embedded software (ESW), many test approaches have been developed on the top of the model of the system[1)]. In this context, the *extended finite state machines* (EFSMs) permit to efficiently represent ESW[2)]. Indeed, ESW is an interactive system which answers to event occurrences and mixes control and data flow.

There are two main categories for classifying ESW testing techniques: *concrete execution* and *symbolic execution* methods. The concrete execution is considered a narrow-width and long-range exploration method, since it reaches deep states of the program space by executing a large number of long paths, but it is far to be an exhaustive approach. The *symbolic execution* represents an alternative for overcoming concrete-execution limitations[3)], but it suffers the solver limitations. Thus, it is considered a wide-width and short-range exploration method, since it exhaustively analyzes the nearest states of the program space, without reaching deepest state or requiring a long execution time.

†1 VLSI Design and Education Center, The Tokyo University, Japan
†2 CREST, Japan Science and Technology Agency
†3 University of Verona, Italy

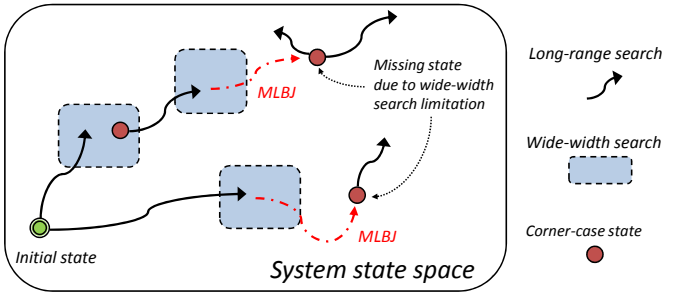


Fig. 1 The EFSM-based concolic approach.

This paper proposes an EFSM-based *concolic* (symbolic and concrete) testing approach for ESW. Figure 1 graphically represents a traditional concolic approach, which alternates long-range-concrete and wide-width-symbolical search techniques[4)]. By using this approach, some corner-case states could remain hard-to-reach, even when the symbolic technique is started in an intermediate state of the execution. To overcome this problem, we propose a multi-level backjumping strategy (MLBJ), which guides the concolic execution in covering corner cases. The main contributions of this paper are the following:

- the proposed EFSM-based methodology is addressing the problem of ESW test generation starting from a widely-used model of the system components rather than their implementation, and thus the approach can be earlier integrated into the ESW design flow;
- a concolic approach has been developed which combines biased-random test generation, symbolic execution, and exploration heuristics over the system model aiming at maximizing the EFSM transition coverage; moreover, the proposed weight-based dependency analysis over EFSMs permits to identify an effective interleaving strategy for the long-range and the wide-width approaches;
- finally, the paper empirically validates the efficiency of the proposed approach by using different case studies.

In particular, the proposed approach early identifies possible symptoms of design

errors by efficiently exploring the EFSM model of the application, and generates effective input sequences to be used in further verification steps which require to stimulate the ESW.

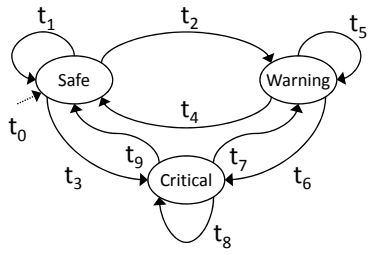
The remainder of this paper is structured as follows. Section 2 describes the EFSM model. Section 3 presents the actual methodology before Section 4 gives some experimental results. Finally, this article closes with a summary and conclusions.

2. Extended finite state machines

An *extended finite state machine* (EFSM) is a Mealy FSM augmented with a finite number of *internal variables*, which are not part of the set of explicit states. Each EFSM transition has an enabling precondition, often named *enabling function*. Traversing a transition changes the EFSM state, produces some output, and modifies the variables values. All these effects are defined by the transition *update function*.

An EFSM is defined as a 7-tuple $M = \langle S, I, O, D, E, U, T \rangle$. S is the finite set of states, I is the set of possible inputs $I = I_1 \times \ldots \times I_n$, O is the set of possible outputs $O = O_1 \times \ldots \times O_m$, and D the set of possible variables values $D_1 \times \ldots \times D_k$. Informally, I_i, O_i, and D_i represent the domain of the i-th input, output, and internal variable, respectively. $E = (I \times D \rightarrow \{0, 1\})$ is the set of enabling functions, which are a boolean predicates over inputs and internal variables. $U = (I \times D \times O \rightarrow D \times O)$ is the set of update functions. Commonly, these are implemented as sequences of assignments with the semantics that unassigned variables retain their original value. $T \subseteq S \times S \times E \times U$ is the *transition relation*. A transition, denoted as t, is an element of T, while $\mathrm{s}(t) \in S$ and $\mathrm{d}(t) \in S$ are the transition source and destination state, respectively. Two transitions t and t' are *adjacent* if $d(t) = s(t')$. A *EFSM path* is a sequence of adjacent transitions. In this work we are considering *deterministic EFSM*, thus for every state, the set of outgoing transitions have mutually exclusive enabling functions. A pair $\langle s, x \rangle \in S \times D$ is called *configuration* of M. The *reset configuration* is the pair $\langle s_0, x_0 \rangle \in S \times D$ of the *reset state* s_0 and the reset values x_0 of the internal variables.

Fig. 2 reports both a graphical and a tabular representation of a simplified in-flight safety system EFSM where $S = \{\text{Safe}, \text{Warning}, \text{Critical}\}$. The inputs $I = \{t, p, o\}$ represent the cabin temperature, pressure, and oxygen rate, respectively. The outputs are $O = \{\text{light}, \text{sound}\}$ and internal variables are $D = \{\text{tva}, \text{pva}, \text{ova}\}$. For each transition, the enabling functions and update functions are reported in the table.



T	En. Func.	Up. Func.
t_0	$rst = 1$	tva=0; ova=0; pva=0;
t_1	$t < T_M$	-
t_2	$t \geq T_M \wedge p < P_M$	tva=t; pva=p; light=1;
t_3	$t \geq T_M \wedge p \geq P_M$	tva=t; pva=p; light=1; sound=1;
t_4	$t < tva$	light=0;
t_5	$t \geq tva \wedge o \geq O_m$	-
t_6	$t \geq tva \wedge o < O_m$	ova=o; sound=1;
t_7	$o > ova$	sound=0;
t_8	$o \leq ova \wedge p \geq pva$	-
t_9	$o \leq ova \wedge p < pva$	light=0; sound=0;

Fig. 2 An EFSM of a simplified in-flight safety system.

3. EFSM-based concolic testing

In the concrete execution approaches, traversing an EFSM transition t depends both on the values of the inputs and of the internal variables. The variable values depend on the path leading from the reset configuration to t. In the following, we identify a not-yet-traversed transition t as the *target transition*. For example, one transition may assign the value 0 to a variable x while the enabling function of a later target transition requires $x > 0$ despite of the value of x not having changed between these transitions. Symbolic execution approaches typically overcome this issue by exhaustively evaluating all possible input values along one or more paths involving the target transition. However, the search range is severely limited by the complexity of the resolution procedures.

To solve this problem, this paper presents a concolic approach for ESW testing, which is based on the EFSM model of the system and integrates biased random execution, which reaches deep states of the system, and a symbolic technique that is weight oriented and ensures exhaustiveness along specific paths leading to the target transition. Algorithm 1 is a high-level description of the proposed concolic approach. The input EFSM is assumed to have all update functions

Algorithm 1: The EFSM-based concolic algorithm.

```
   input: an EFSM M; timeout, inactivity ∈ ℝ+
1  TSeq ← ∅, RInf ← ∅
2  DGph ← DependencyAnalysis(M)
3  while elapsed time < timeout do
4  |  while coverage inactivity timeout not expired do
5  |  |  (TSeq, RInf) ← LongRangeSearch(M, RInf)
6  |  (TSeq, RInf) ← MlbjWideWidthSearch(M, RInf, DGph)
7  return (TSeq, RInf)
```

expressed as a sequence of assignments. Such a normal form can be obtained from any EFSM by applying the transformations described in[5]. The output of the algorithm is the test set and the coverage information. At first, the test set *TSeq* is empty, and no reachability information (*RInf*) is available (line 1). In particular, *RInf* keeps track for each transition of the test sequences that lead to its traversal and of the corresponding EFSM configurations. Such information is used for restoring the system status when the algorithm switches between the symbolic and concrete techniques. The test generation runs until a user specified timeout expires (line 3). First the algorithm executes a long-range concrete approach, then a symbolic wide-width approach, which exploits the MLBJ to cover corner cases. The latter starts when the coverage remains steady for a user specified *inactivity* timeout (line 4). The algorithm reverts back to the first approach as soon as the wide-width search traverses a target transition. The long-range search exploits constraint-based heuristics[6], that focus on the traversal of just one transition at a time. Such approaches scale well with design size and, significantly improves the bare pure-random approach.

The main contributions of the present work are presented in the following sections. Section 3.1 proposes a weight-based analysis of the dependencies among inputs, internal variables, and EFSM paths. Such a dependency analysis permits to selectively choose a path for the symbolic execution when the concrete approach fails. Indeed, an open problem of the combined concrete and symbolic approaches is the selection of where to start the wide-width search[7]. If the switch is premature, the target transition will be too far and it will not be traversed because of the short range of symbolic methods. On the other hand, a late switch will exclude necessary paths from the search. Our heuristics backward propagates the data dependencies of the target-transition enabling function and computes a measure of how much each transition affects the value of involved internal variables. The measure is easily extended to paths. Informally, the higher is the dependence, the higher is the probability that the target transition will also be covered while traversing the path. Section 3.2 describes the multi-level backjumping technique. Such an approach addresses the selective symbolic execution of EFSM paths, which are terminating with the target transition and have a high dependence on the inputs. The high dependence on the inputs guarantee the controllability of the enabling function of the target transitions, therefore the decision procedure most likely is able to find a solution for traversing the symbolic path, which leads from an intermediate state of the concrete execution.

3.1 Dependency analysis

In the present work the dependencies between inputs, variables and EFSM transitions are expressed by means of a weights associated to EFSM paths. The higher is the value of the weight associated with a path, the greater is the input-dependency between the target transition and the path. Thus, we define how a given path π, whose last element is the target transition t, *propagates* and *consumes weight*. The weight C_π is a measure of how much the variable values at the end of π have been influenced by the inputs read along π itself. The higher is this value, the higher is the probability that the constraint solver will be able to infer a test sequence for π that satisfies t's enabling function. Actually, C_π is a measure of the input-data dependency of the target transition t with respect to the path π.

Let A be the set of possible assignments. The function Mod: $A \to (D \cup O)$ computes the target of an assignment, that is either a internal variable or an output. The function Ref: $A \to \wp(D \cup I)$ computes the set of variables and inputs referenced by an assignment. For example, the assignment $r[i] := r[j]$ modifies r and references $\{r, i, j\}$, and the assignment $r[i] := t[j]$ modifies r and references $\{t, i, j\}$.

Let the *data-dependence coefficient*[⋆1] be $\Delta\colon A \times D \to \mathbb{R}^+$ where for every assignment $a \in A$ and internal variable $d \in D$:

$$\Delta_a(d) = \begin{cases} \frac{1}{|\operatorname{Ref}(a)|} & \text{if } d \in \operatorname{Ref}(a), \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The Δ coefficient measures how much each assignment depends on the internal variables. Similarly, let the *input-dependence coefficient* be $\Upsilon\colon A \to \mathbb{R}^+$, defined as:

$$\Upsilon_a = \begin{cases} 0 & \text{if } \operatorname{Ref}(a) = \emptyset, \\ \frac{|\operatorname{Ref}(a) \cap I|}{|\operatorname{Ref}(a)|} & \text{otherwise.} \end{cases} \tag{2}$$

The Υ coefficient measures how much the value of the variable modified by the assignment a depends on the inputs. Given an EFSM, both these measures are constant, computed before the test-generation phase, and associated with each assignment in the EFSM. The computation of Δ and Υ requires time proportional to $|D|$ and to the number of assignments in target transition update function.

Let a *weight distribution* be a function which associates a non-negative real value with each internal variable of the EFSM, i.e. $w\colon D \to \mathbb{R}^+$. We define how such a weight distribution is propagated and consumed by an assignment $a \in A$. Let the *propagation function* and *consumption function* be respectively $P\colon A \times (D \to \mathbb{R}^+) \to (D \to \mathbb{R}^+)$ and $C\colon A \times (D \to \mathbb{R}^+) \to \mathbb{R}^+$ defined as:

$$\begin{aligned} P_a(w) &= w_l \Delta_a + w[l/0], \\ C_a(w) &= w_l \Upsilon_a. \end{aligned} \tag{3}$$

where $l = \operatorname{Mod}(a)$, w_l is the weight associated with the internal variable l, and $w[l/0]$ denotes the function that is equal to w on every element of its domain, except for l where it evaluates to 0. These definitions say that the weight distribution is affected by a only when there is an initial positive weight on the variable modified by a.

An EFSM update function is defined as a sequence α, in case empty, of assignments. We define how a weight distribution w is affected by such a sequence. By induction on the length of α, the empty sequence of assignments does not affect the weight propagation nor consumes weight. Let $\alpha = a :: \beta$ be the concatenation of the assignment a with the sequence β, the *propagation function* and

⋆1 In the following, the symbol $|\cdot|$ is the cardinality operator which counts the elements that occur in the set.

Algorithm 2: MultilevelBackJumping

input: target transition $\bar{t} \in T$, timeout $\in \mathbb{R}^+$

Let $\bar{w}\colon D \to \mathbb{R}^+$ be the initial weight distribution

$\forall d \in D\,.\,\bar{w}(d) = \begin{cases} 1 & \text{if } d \in \operatorname{Ref}\big(\text{EnableFunction}(\bar{t})\big), \\ 0 & \text{otherwise.} \end{cases}$

$p \leftarrow \big\{(\bar{t}, \bar{w}, 0)\big\}$

while *elapsed time* < timeout **do**

  $(t :: \pi, w, c) \leftarrow$ remove_top(p)

  if $t :: \pi$ *is satisfiable* **then**

    if $C_\pi < C_{t::\pi}$ **then**

      foreach *configuration* $k \in \operatorname{IRef}\big(\mathrm{s}(t)\big)$ **do**

        if $k \wedge t :: \pi$ *is satisfiable* **then**

          return target transition covered

      foreach $\big\{t' \in T \;\big|\; \mathrm{d}(t') = \mathrm{s}(t)\big\}$ **do**

        $\alpha \leftarrow$ UpdateFunction(t')

        push$\Big(p, \big(t' :: t :: \pi, P_\alpha(w), C_\alpha(w) + c\big)\Big)$

consumption function over sequences are defined as:

$$\begin{aligned} P_\alpha(w) &= P_a\big(P_\beta(w)\big), \\ C_\alpha(w) &= C_a\big(P_\beta(w)\big) + C_\beta(w). \end{aligned} \tag{4}$$

This says that the weight consumed by an update function is the sum of the weights consumed by each assignment. This definition can be immediately extended to paths by considering the concatenation of the update functions.

3.2 Multi-level Backjumping

When the long-range-concrete approach does not produce any new coverage point, the concolic algorithm switches to MLBJ-based wide-width approach. Typically, some hard-to-traverse transitions, whose enabling functions involve internal variables, prevent the random-based technique going further in the exploration. In this case, the MLBJ is able to selectively address paths, with high

Table 1 Characteristics of the case-studies.

DUT	LoC	CC	I	O	V	T	S	GT
Ifss	282	157	77	6	65	36	3	-
Ciitp	324	54	255	274	80	28	6	-
Lift	297	172	82	16	48	30	4	-
Atm	282	91	168	48	80	42	10	-
Inres	158	37	41	4	32	21	4	-
Elevator	3391	460	8	32	5037	775	382	0.123
Filter	256	24	66	32	293	37	21	0.015
Thermostat	142	52	58	8	56	68	2	0.048

dependency on inputs, for symbolically simulating them. Such paths are leading from an intermediate state of the execution to the target transition, thus the approach is exhaustive in a neighborhood of the corner case.

Algorithm 2 presents a structured description of the multi-level backjumping heuristics. A target transition $\bar{t}$ is selected. A progressively increasing neighborhood of $\bar{t}$ is searched for paths π leading to $\bar{t}$ having maximal consumed weight C_π. The symbolic execution is started only on transitions with a non null contribute to C_π (line 7). To further narrow the search space, we use the decision procedure to check for input-control dependencies that may prevent in advance a path from being traversed (line 6). More in details, a visit is started from $\bar{t}$ that proceeds backward in the EFSM graph. The visit uses a priority queue p that initially contains only $\bar{t}$. Elements of p are paths that end in $\bar{t}$. Each path of p is accompanied by its weight distribution w and consumed weight c. At each iteration, a path $t :: \pi$ with maximal consumed weight c is removed from p. The constraint solver is used to check if the path $t :: \pi$ can be proved unsatisfiable in advance; in this case $t :: \pi$ is discarded so the sub-tree following $t :: \pi$ will not be explored. Otherwise, if the first transition t has apported a positive consumed weight, then for each configuration k stored in the source state of t the constraint solver checks the existence of a test sequence that from k leads to the traversal of $t :: \pi$ and thus of $\bar{t}$. Finally, for each transition t' that precedes t the path $t' :: t :: \pi$ is added to p.

4. Experimental results

Eight case studies were conducted to demonstrate the feasibility of using the proposed approach to generate high quality test sequences by traversing EFSM

Table 2 Result comparisons.

	Rand		Cons		Symb		EC	
DUT	TC%	Time	TC%	Time	TC%	Time	TC%	Time
Ifss	32.31	25.62	55.81	59.10	100.00	1024.10	100.00	12.72
Ciitp	96.15	6.81	96.15	18.71	100.00	180.33	100.00	23.83
Lift	11.11	0.71	81.48	63.55	100.00	1224.52	100.00	13.97
Atm	12.50	0.20	71.10	76.00	100.00	934.17	100.00	51.02
Inres	89.47	24.11	92.73	18.40	100.00	186.82	100.00	24.01
Elevator	69.31	30.23	69.34	190.90	1.42	4914.18	81.52	4459.62
Filter	62.85	0.81	62.85	2.74	68.57	3503.11	100.00	30.52
Thermostat	33.81	8.03	47.01	170.51	47.01	3402.32	60.21	37.60

model of ESW. Five of them, *Ifss*, *Ciitp*, *Atm*, *Lift*, and *Inres*, have been deduced from the EFSM-based specifications of the ESW proposed in[8]. The other EFSMs, *Elevator*, *Filter*, and *Thermostat*, are industrial designs and they have been automatically extracted from C sources.

Table 1 describes the characteristics of the case-studies. Columns *LoC* and *CC* respectively show the number of lines of code and the *McCabe Cyclomatic Complexity*. Columns *I*, *O*, and *V* respectively show the bit size of the inputs, outputs, and internal variables. Columns *T* and *S* show the number of EFSM transitions and states. For models originally written in C, the column *GT* shows the EFSM generation time. The other EFSMs have been manually converted from tabular specifications, so their generation time is not reported.

The reported cyclomatic complexity is a widely accepted language-independent metric which provides a measure of the number of independent execution paths. In particular it is an indicator of testability: a value higher than 50 usually means a program difficult to test.

Table 2 compares the proposed EFSM-based concolic approach (EC) with a pure random approach (Rand), a constraint-based heuristics (Cons), and a pure symbolic approach (Symb). The Cons approach focuses on the traversal of just one transition at a time. Despite Cons uses a constraint solver, it can be considered a long-range concrete techniques as the Rand. For each approach, the table reports the maximum achieved transition coverage (TC%) and the execution time in seconds (Time). Each experiment was carried out with a time threshold of 5000 seconds. The execution time refers to the time when the testing engine achieved the last improvement in the transition coverage.

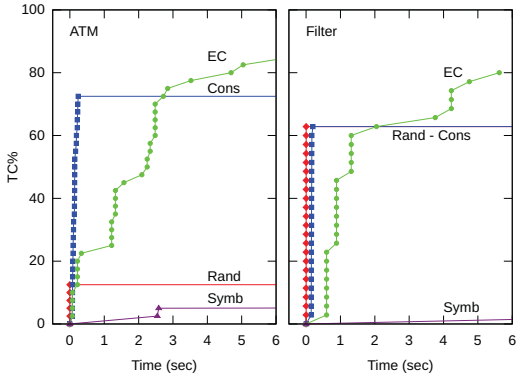


Fig. 3 The transition coverage versus the test generation time.

We observe that: (1) for the case studies derived from the specifications[8], the proposed EC approach outperforms the RAND and the CONS in terms of transition coverage. The SYMB approach reaches the same coverage as the EC, but the execution time is one or two orders of magnitude larger; (2) for the industrial case studies, the EC outperforms all the other approaches in terms of coverage. In particular, the coverage results of the SYMB approach are poor because of the complexity of the decision problems, which require execution time higher than the fixed time threshold.

Fig. 3 presents the trend of the four approaches in terms of transition coverage versus test generation time. For lack of space and for readability we selected one industrial design (*Filter*) and one (*Atm*) from the set of EFSM-based specification, and we limited the execution time up to 6 seconds. We observe that the pure random RAND and the constraint-based CONS approaches very rapidly increase the coverage, but achieve steady values, as they are unable to cover corner cases. On the contrary, the symbolic approach SYMB slowly increases the coverage because of the high execution time of the decision procedure. Finally, the EC uses a certain amount of time for the initial computation of the *input-* and *data-dependence coefficients*. Then, the coverage presents two different trends. It very rapidly increases as the long-range concrete approach is executed. As soon as it is no more able to traverse new transitions, the wide-width search (based on the *MLBJ* algorithm) is started. It is slower, because it requires more time for solving path constraints, but it allows to exit from steady conditions achieving very high transition coverage.

5. Concluding Remarks

In this paper we propose a testing approach for ESW based on EFSM, a widely used model adopted in ESW design. The model-based technique permits the earlier identification of design errors, and an efficient generation of input stimuli for the following verification phases. With the aim of maximizing the EFSM transition coverage, a concolic technique has been developed which combines biased-random test generation, symbolic execution, and exploration heuristics based on the multi-level backjumping approach. The proposed weight-based dependency analysis supports a selective symbolic-simulation strategy that overcomes pure-random, constraint-based, and traditional symbolic-execution approaches. Finally, the effectiveness of the proposed framework has been evaluated on several case studies.

References

1) DiasNeto, A., Subramanyan, R., Vieira, M. and Travassos, G.: A survey on model-based testing approaches: a systematic review, *In the Proc. of the 22th IEEE/ACM International Conference on Automated SW Engineering*, ACM, pp.31–36 (2007).
2) Sangiovanni-Vincentelli, A.: Embedded system design and hybrid systems, *Control Using Logic-Based Switching*, pp.17–38 (1997).
3) King, J.: Symbolic execution and program testing, *Communications of the ACM*, Vol.19, No.7, pp.385–394 (1976).
4) Majumdar, R. and Sen, K.: Hybrid concolic testing, *In the Proc. of the 29th International Conf. on SW Engineering*, IEEE Computer Society, pp.416–426 (2007).
5) DiGuglielmo, G., Fummi, F., Marconcini, C. and Pravadelli, G.: EFSM manipulation to increase high-level ATPG efficiency, *In the Proc. of IEEE International Symposium on Quality Electronic Design*, pp.57–62 (2006).
6) DiGuglielmo, G., Fummi, F., Pravadelli, G., Soffia, S. and Roveri, M.: Semi-Formal Functional Verification by EFSM traversing via NuSMV, *In the Proc. of IEEE International High Level Design Validation and Test Workshop*, pp.58–65 (2010).
7) Ho, P., Shiple, T., Harer, K., Kukula, J., Damiano, R., Bertacco, V., Taylor, J. and Long, J.: Smart simulation using collaborative formal and simulation engines, *In the Proc. of IEEE/ACM International Conf. on CAD*, IEEE, pp.120–126 (2000).
8) Kalaji, A., Hierons, R. and Swift, S.: A Search-Based Technique for Automatic Test Generation from an Extended Finite State Machine.