Test Case Generation Using PDA

Puneet Batheja

Test Case Generation Using PDA

Puneet Bhateja
IRISA / INRIA Rennes
Campus Universitaire de Beaulieu
35042 Rennes Cedex - FRANCE
puneet.bhateja@inria.fr

***Abstract*—IOLTS (input output labeled transition system) is a versatile model and is frequently used in model based testing to model the functional behavior of an IUT (implementation under test). However when a system is tested remotely, its observed behavior can be different from its actual functional behavior. In [2], we defined a notion of remotely observed behavior of an IOLTS in terms of its actual behavior. This paper contributes by proposing a methodology to simulate a PDA (push down automaton) from the given IOLTS such that the simulated PDA precisely expresses the remotely observed behavior of the IOLTS. The simulated PDA can be thought of as an automatic test generator for remote testing.**

***Keywords*-Test generation, static testing, dynamic testing.**

I. Introduction

Most of the reactive systems are safety critical, thus it is vital to test them before their deployment. Testing in its most basic form comprises two steps. In the first step the tester simulates the IUT with some inputs and observes the response of the IUT. Next, the tester appraises the response of the IUT vis-a-vis the given specification.

In model based testing, first of all a model is created for the IUT , and then test generation is carried out based on this model. There are primarily two advantages to model based testing. One, a model serves as a unifying point of reference for everyone involved in the project. Two, most of the models have rich underlying theory, which makes the IUT amenable to formal analysis.

IOLTS is one such model which is commonly used to depict the functional behavior of an IUT. We consider a case wherein the functional behavior represented by an IOLTS is observed remotely through a pair of FIFO queues— input queue and the output queue. What makes the remotely observed behavior different from the actual behavior is the fact that the two queues are presumably disjoint, and therefore the actions through them can be observed in an order different from the one in which they actually happened. To this end, in [2] we formulated a notion of remotely observed behavior of a given IOLTS. Further in [1], we proved that the remote behavior of an IOLTS can be captured by a context free grammar. Based on the well-known result from the theory of formal languages that any language expressed by a context free grammar can also be automatically generated by a push-down automaton (PDA) [3], this paper contributes by proposing a methodology to simulate a PDA from the given IOLTS. The simulated PDA can generate test cases required to test the IOLTS remotely.

The paper is organised as follows: Section 1 comprises the ongoing introduction; Section 2 formally defines the IOLTS and some notations relevant to it; Section 3 recalls the notion of remote behavior of an IOLTS; Section 4 explains how to define a PDA corresponding to an IOLTS such that the PDA precisely captures the remote behavior of the IOLTS; and finally Section 5 concludes the paper.

II. IOLTS: definition and notations

An IOLTS (input output labeled transition system) is a state-based model which is widely used to explain the behavior of an interactive system [4], [5]. A characteristic feature of an IOLTS is that it exhibits in a mutually distinctive manner the actions whereby the system sends/receives messages to/from its environment. Formally, an IOLTS can be defined as follows:

***Definition* 1:** *An IOLTS is a quadruple $M = (Q^M, A^M, \rightarrow_M, q_0^M)$, where Q^M is a non empty set of states; A^M is a set of actions, and is further partitioned into an input alphabet A_I^M and an output alphabet A_O^M; $q_0^M \in Q^M$ is the initial state of the IOLTS; and $\rightarrow_M \subseteq Q^M \times A^M \times Q^M$ is a transition relation.*

We now state some definitions and notations with respect to an IOLTS $M = (Q^M, A^M, \rightarrow_M, q_0^M)$.

- $\forall q, q' \in Q^M, \forall a \in A^M : (q \xrightarrow{a}_M q')$ is true iff $(q, a, q') \in \rightarrow_M$. $\forall q \in Q^M, \forall a \in A^M : (q \xrightarrow{a}_M)$ is true iff $\exists q' \in Q^M : (q, a, q') \in \rightarrow_M$. We can generalize this to strings of all lengths. $\forall q, q' \in Q^M, \forall a_1.a_2.\cdots.a_n \in (A^M)^\star : q \xrightarrow{a_1.a_2\cdots.a_n}_M q'$ is a run of M iff $\exists q_1, q_2, \cdots, q_{n-1} : (q \xrightarrow{a_1}_M q_1) \wedge (q_1 \xrightarrow{a_2}_M q_2) \wedge \cdots \wedge (q_{n-1} \xrightarrow{a_n}_M q')$.

- For $\sigma \in (A^M)^\star$ and $X \subseteq A^M, \sigma \downarrow_X$ is called the projection of σ on X. It can be defined inductively. Base case: $\epsilon \downarrow_X = \epsilon$. Induction step: $\sigma \downarrow_X = a.(\sigma' \downarrow_X)$,

when $a \in X$ and $\sigma = a.\sigma'$; $\sigma \downarrow_X = \sigma' \downarrow_X$, when $a \notin X$ and $\sigma = a.\sigma'$

- A state is called deadlocked, if the system cannot perform any output action in that state. Formally, $q \in Q^M$ is called deadlocked iff $\forall x \in A_O^M : \neg(q \xrightarrow{x}_M)$

We make two structural assumptions about each IOLTS that we consider in this paper.

1) An IOLTS does not have a loop comprising only output symbols. Formally $\forall q \in Q^M, \forall x_1.x_2 \cdots x_n \in (A_O^M)^+ : q \xrightarrow{x_1.x_2 \cdots x_n}_M q$ is false. This restriction is quite justified, because such a loop indicates that the modeled system could produce an infinite behavior spontaneously. On the other hand, imposition of this restriction ensures that each IOLTS has at least one deadlocked state.

2) An IOLTS in a deadlocked state is ready to accept every input from its environment. This restriction is also justified, because when a system is not in a state to produce any output for its environment, then it should be ready to accept every input from its environment. One particular state of an IOLTS is designated as a dead state and every deadlocked state has a transition on every input symbol $a \in A_I^M$ to that dead state. The dead state in turn has a self loop on every input symbol in the set A_I^M. To avoid clutter, the dead state is note shown in the IOLTS diagrams. Nevertheless, the formal definition of the remotely observed behavior assumes that such a state exists.

III. Remote testing

In remote testing, the tester and the IUT are separated by a medium which is modeled as a pair of FIFO queues. The tester and the IUT interact with each other complementarily, that is, the input (output) queue of the tester is the output (input) queue of the IUT. Based on how the behavior of the IUT is observed, remote testing can be classified as static testing or dynamic testing.

A. Static testing

In static testing the tester supplies the entire input upfront to the IUT and then waits for the entire response of the IUT. When a system is subjected to static testing, it exhibits a static test behavior which is formally defined as follows:

***Definition* 2:** For a given IOLTS $M = (Q^M, A^M, \rightarrow_M, q_0^M)$, the static test behavior $STB(M)$ is the set of pairs $(u, v) \in (A_I^M)^\star \times (A_O^M)^\star$ such that there is a run $q_0 \xrightarrow{w}_M q_d$ in M, where:

- q_0 is the initial state and q_d is a deadlocked state.

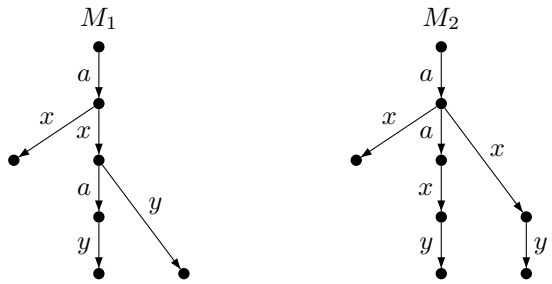

Figure 1. M_1 and M_2 with different actual behaviors but same static test behavior.

- $u = w \downarrow_{A_I^M}$.
- $v = w \downarrow_{A_O^M}$.

Intuitively, $(u, v) \in STB(M)$ means that when the tester simulates the IUT with input u, it gets output v as response.

***Example* 1:** Figure 1 shows two systems M_1 and M_2 over $A_I^{M_1} = A_I^{M_2} = \{a\}$ and $A_O^{M_1} = A_O^{M_2} = \{x, y\}$. M_1 and M_2 have different actual behaviors but same static test behavior which is given below using regular expressions.

$$STB(M_1) = STB(M_2) = (\epsilon, \epsilon) + (a, x) + (aa, xy) + (a, xy) + (aa^+, x) + (aaa^+, xy) + (aa^+xy)$$

Both in M_1 and M_2, the deadlocked states comprises the root node, the left leaf node, the middle leaf node, and the right leaf node. The static test behavior w.r.t the root node is (ϵ, ϵ), and with respect to the left leaf node is $(a, x) + (aa^+, x)$ and so on.

B. Dynamic testing

In dynamic testing, the tester has an additional discrimination power. Unlike in static testing, wherein it supplies all the inputs upfront to the system, here it supplies the inputs incrementally. The next sequence of inputs to be supplied is decided on the fly. When a system is subjected to dynamic testing, it exhibits a dynamic test behavior which is formally defined as follows:

***Definition* 3:** For a given IOLTS $M = (Q^M, A^M, \rightarrow_M, q_0^M)$, the dynamic test behavior $DTB(M)$ is the set of sequences of pairs $(u_1, v_1).(u_2, v_2) \cdots (u_k, v_k) \in ((A_I^M)^\star \times (A_O^M)^\star)^\star$ such that there is a run $q_0 \xrightarrow{w_1}_M q_1 \xrightarrow{w_2}_M \cdots \xrightarrow{w_k}_M q_k$ in M, where:

- q_0 is the initial state, and $q_1, q_2, \cdots, q_k$ are some of the (not necessarily all) deadlocked states on the run.
- $v_j = w_j \downarrow_{A_O^M}$ for all $1 \leq j \leq k$
- $u_j = w_j \downarrow_{A_I^M}$ for all $1 \leq j < k$

***Example* 2:** Figure 2 shows two systems M_1 and M_2 over $A_I^{M_1} = A_I^{M_2} = \{a\}$ and $A_O^{M_1} = A_O^{M_2} = \{x, y\}$ which

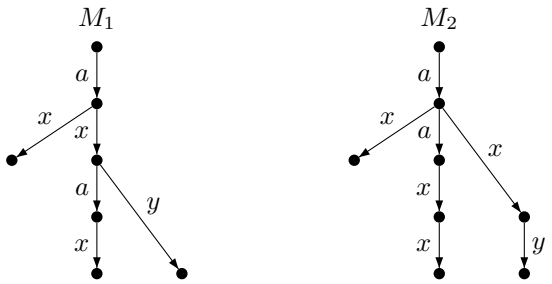

Figure 2. M_1 and M_2 with different actual behaviors but same dynamic test behaviors.

have different actual behaviors but same dynamic behavior. The dynamic test behavior of the two systems is given below using regular expressions.

$DTB(M_1) = DTB(M_2) = (\epsilon,\epsilon) + \{(a^+,x) + (a^+,xy) + (aa^+,xx)\}.(a^+\epsilon)^\star + (\epsilon,\epsilon).\{(a^+,x) + (a^+,xy) + (aa^+,xx)\}.(a^+,\epsilon)^\star$

We would like to mention that even though the systems dealt with in this section were finite systems, the behavior exhibited by them was infinite. This is because of the structural assumption on the IOLTS that there is an implicit edge from every deadlocked state, on every missing input, to a dead state. This dead state in turn has a self loop on every input symbol in A_I^M.

IV. Test generation

This section describes how to simulate a PDA from the given IOLTS. We will consider two cases. In the first case the simulated PDA generates test cases for the static testing, and in the second case the simulated PDA generates test cases for the dynamic testing.

A. Static testing

Given an IOLTS $M = (Q^M, A^M, \rightarrow_M, q_0^M)$, we can define a PDA $N = (Q^N, A^N, \Gamma, q_0^N, \perp, q_f, \rightarrow_N)$, where:

- Q^N is the set of states of the PDA such that $Q^N = Q^M$.
- A^N is the tape alphabet of the PDA such that $A^N = A^M$.
- Γ is the stack alphabet of the PDA such that $\Gamma = \{\perp\} \cup A_O^M$.
- q_0^N is the initial state of the PDA such that $q_0^N = q_0^M$.
- $\perp$ is the initial stack symbol of the PDA.
- q_f is the final state of the PDA.
- $\rightarrow_N \subseteq (Q^N \times A^N \cup \{\epsilon\} \times \Gamma \times Q^N \times \Gamma^\star)$ is the transition relation of the PDA defined by the following four rules $\mathbf{R_1} - \mathbf{R_4}$.

Before we state the rules, it should be noted that the interpretation of an element of the set $\rightarrow_N$ is standard [3]. For example, $(q,a,\alpha,q',\beta) \in \rightarrow_N$ means the following: The PDA in state q scans an input symbol a, while the top symbol of the stack is α. As a result, the PDA changes its state to q', moves paste the input symbol a, and replaces the stack's top symbol $\alpha \in \Gamma$ with a string $\beta \in \Gamma^\star$. And, had it been ϵ in place of a, it would mean that the PDA does everything similar, except that it does not move paste the current symbol being scanned.

$\mathbf{R_1}$ $\forall a \in A_I^M, \forall q,q' \in Q^M$: If $(q,a,q') \in \rightarrow_M$, then $(q,a,\alpha,q',\alpha) \in \rightarrow_N$

$\mathbf{R_2}$ $\forall x \in A_O^M, \forall q,q' \in Q^M$: if $(q,x,q') \in \rightarrow_M$, then $(q,\epsilon,\alpha,q',x.\alpha) \in \rightarrow_N$

$\mathbf{R_3}$ $\forall q \in Q^M, \forall x \in A_O^M$: if q is deadlocked, then $(q,x,x,q,\epsilon) \in \rightarrow_N$

$\mathbf{R_4}$ $\forall q \in Q^M$: if q is deadlocked, then $(q,\epsilon,\perp,q_f,\perp) \in \rightarrow_N$.

As is customary, a tuple (q,w,α) denotes a configuration of the PDA, where q is the control state of the PDA, w is string scanned till now, and α is the contents of the stack (left most symbol of α indicates the top of the stack). If a PDA makes a transition on some string $\sigma \in (A^N)^\star$ from a configuration (q,w,β) to (q',w',β'), then we denote it by $(q,w,\beta) \overset{\sigma}{\Rightarrow} (q',w',\beta')$.

***Lemma* 1:** For all $w \in (A^M)^\star$, there is a run $q_0 \overset{w}{\rightarrow_M} q_d$ in the IOLTS M such that $w \downarrow_{A_I^M} = u_1.u_2 \cdots u_m$ and $w \downarrow_{A_O^M} = v_1.v_2 \cdots v_n$ if and only if $(q_0,\epsilon,\perp) \overset{u_1.u_2 \cdots u_m}{\Longrightarrow} (q_d, u_1.u_2 \cdots u_m, v_n.v_{n-1} \cdots v_1.\perp)$.

Proof: We can prove this by induction on $|w|$, that is, the length of w. Base Case: Let $|w| = 0$. It means that $q_0 \overset{\epsilon}{\rightarrow_M} q_d$ is the run of the IOLTS M. Since a transition on ϵ is not defined in the IOLTS M, therefore we have $q_0 = q_d$. Since it is the case that $\epsilon \downarrow A_I^M = \epsilon \downarrow_{A_O^M} = \epsilon$, we have $(q_0,\epsilon,\perp) \overset{\epsilon}{\Rightarrow} (q_0,\epsilon,\perp)$. This completes the base case step of the proof.

Induction Step: Let $q_0 \overset{w}{\rightarrow_M} q \overset{a}{\rightarrow_M} q'$ is the run of the IOLTS M such that $w \in (A^M)^\star, q \in A^M$. If $w \downarrow_{A_I^M} = u$ and $w \downarrow_{A_O^M} = v$, then by the induction hypothesis we have $(q_0,\epsilon,\perp) \overset{u}{\Rightarrow} (q,u,v^R.\perp)$, where v^R is the reverse of the string v. Now two cases arise. (1) If $a \in A_I^M$, then by the rule $\mathbf{R_1}$ we get $(q_0,\epsilon,\perp) \overset{u}{\Rightarrow} (q,u,v^R.\perp) \overset{a}{\Rightarrow} (q',u.a,v^R.\perp)$. (2) If $a \in A_O^M$, then by $\mathbf{R_2}$ we get $(q_0,\epsilon,\perp) \overset{u}{\Rightarrow} (q,u,v^R.\perp) \overset{\epsilon}{\Rightarrow} (q',u,a.v^R.\perp)$. Both the cases thus prove that the left hand side implies the right hand side. Similarly, we can prove the converse, by applying

induction on $|u_1.u_2 \cdots u_m| + |v_1.v_2 \cdots v_n|$.

***Theorem* 1:** For a given IOLTS M, $\forall u \in (A_I^M)^\star, \forall v \in (A_O^M)^\star : (u, v) \in STB(M)$ if and only if $u.v^R \in L(N)$, where v^R is the reverse of the string v and $L(N)$ is the language generated by the PDA N.

Proof: Suppose that $(u, v) \in STB(M)$. By the Definition 2, it means that there is a run $q_0^M \xrightarrow{w}_M q_d$ in M such that (1) q_0^M is the initial state, (2) q_d is the deadlocked state, and (3) $w \downarrow_{A_I^M} = u$ and $w \downarrow_{A_O^M} = v$. If $v = v_1.v_2 \cdots v_n$, then by Lemma 1, we have $(q_0, \epsilon, \perp) \overset{u}{\Rightarrow} (q_d, u, v_n.v_{n-1} \cdots v_1.\perp)$. Now if the rule $\mathbf{R_3}$ is applied n times continuously, we get $(q_d, u, v_n.v_{n-1} \cdots v_1.\perp) \overset{v_n.v_{n-1} \cdots v_1}{\Longrightarrow} (q_d, u, \perp)$. Finally by the virtue of the rule $\mathbf{R_4}$, we have $(q_d, u, \perp) \overset{\epsilon}{\Rightarrow} (q_f, u, \perp)$.

Now let us prove the converse. Suppose that $u.v^R \in L(N)$. It means that we have $(q_0, \epsilon, \perp) \overset{u.v^R}{\Rightarrow} (q_f, u.v^R, \beta)$, where $\beta \in \Gamma^\star$. By $\mathbf{R_4}$, we know that the PDA can reach the final state q_f only when the top of the stack is $\perp$ and the PDA is in a deadlocked state. So, we have $(q_0, \epsilon, \perp) \overset{u.v^R}{\Rightarrow} (q_d, u.v^R, \perp) \overset{\epsilon}{\Rightarrow} (q_f, u.v^R, \perp)$, where q_d is a deadlocked state. Now by $\mathbf{R_3}$, we have $(q_0, \epsilon, \perp) \overset{u}{\Rightarrow} (q_d, u, v^R.\perp) \overset{v^R}{\Rightarrow} (q_d, u.v^R, \perp) \overset{\epsilon}{\Rightarrow} (q_f, u.v^R, \perp)$. Finally by the Lemma 1, $q_0 \xrightarrow{w}_M q_d$ such that $w \downarrow_{A_I^M} = u$ and $w \downarrow_{A_O^M} = v$. Hence, by the Definition 2 $(u, v) \in STB(M)$. With this, the proof is over.

The PDA simulated above can be thought of as a test generator. For example, if the PDA generates a string $u.v^R$, it means that the tester should simulate the IUT with an input u and should positively expect v as the corresponding response of the IUT.

B. Dynamic testing

Similar to static testing, we can also simulate a PDA corresponding to a given IOLTS such that the simulated PDA expresses precisely the dynamic behavior of the IOLTS.

Given an IOLTS $M = (Q^M, A^M, \rightarrow_M, q_0^M)$, we can define a PDA $N = (Q^N, A^N, \Gamma, q_0^N, \perp, q_f, \rightarrow_N)$, where:

- Q^N is the set of states of the PDA such that $Q^N = Q^M$.
- A^N is the tape alphabet of the PDA such that $A^N = A^M \cup \{\$\}$.
- Γ is the stack alphabet of the PDA such that $\Gamma = \{\perp, \$\} \cup A_O^M$.
- q_0^N is the initial state of the PDA such that $q_0^N = q_0^M$.
- $\perp$ is the initial stack symbol of the PDA.
- q_f is the final state of the PDA.
- $\rightarrow_N \subseteq (Q^N \times A^N \cup \{\epsilon\} \times \Gamma \times Q^N \times \Gamma^\star)$ is the transition relation of the PDA defined by the following six rules $\mathbf{R_1'} - \mathbf{R_6'}$.

  $\mathbf{R_1'}$ $\forall a \in A_I^M, \forall q, q' \in Q^M$: If $(q, a, q') \in \rightarrow_M$, then $(q, a, \alpha, q', \alpha) \in \rightarrow_N$

  $\mathbf{R_2'}$ $\forall x \in A_O^M, \forall q, q' \in Q^M$: if $(q, x, q') \in \rightarrow_M$, then $(q, \epsilon, \alpha, q', x.\alpha) \in \rightarrow_N$

  $\mathbf{R_3'}$ $\forall q \in Q^M, \forall x \in A_O^M$: if q is deadlocked, then $(q, x, x, q, \epsilon) \in \rightarrow_N$

  $\mathbf{R_4'}$ $\forall q \in Q^M$: if q is deadlocked, then $(q, \epsilon, \perp, q_f, \perp) \in \rightarrow_N$.

  $\mathbf{R_5'}$ $\forall q \in Q^M$: if q is deadlocked, then $(q, \$, \alpha, q, \$\alpha) \in \rightarrow_N$.

  $\mathbf{R_6'}$ $\forall q \in Q^M$: $(q, \$, \$, q, \epsilon) \in \rightarrow_N$.

***Theorem* 2:** $u_1.\$.u_2.\$...\$.u_k.\$.\$.v_k^R.\$...\$.v_2^R.\$.v_1^R \in L(N)$ iff $(u_1, v_1).(u_2, v_2)...(u_k, v_k) \in DTB(M)$, where $\forall 1 \leq j \leq k : u_j \in (A_I^M)^\star, v_j \in (A_O^M)^\star$.

Proof: We omit the proof owing to space constraints, however it is quite similar to that of Theorem 1.

V. Conclusion

PDA is a computational model which is mostly used to depict the control flow of a recursive programme. In this paper, we have explained the use of PDA in test generation, which hitherto is unheard of.

References

[1] Puneet Bhateja. Grammar based asynchronous testing. In *ISEC*, pages 105–110, 2009.

[2] Puneet Bhateja, Paul Gastin, and Madhavan Mukund. A fresh look at testing for asynchronous communication. In *ATVA*, pages 369–383, 2006.

[3] John E. Hopcroft and Jefferey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Adison-Wesley Publishing Company, Reading, Massachusets, USA, 1979.

[4] Nancy A. Lynch and Mark R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2:219–246, 1989.

[5] Machiel van der Bijl and Fabien Peureux. I/O-automata based testing. In *Model-Based Testing of Reactive Systems*, pages 173–200, 2004.