Optimal Test Input Sequence Generation for Finite State Models and Pushdown Systems

Ajay Chander
DOCOMO USA Labs
Palo Alto CA 94304, USA
chander@docomolabs-usa.com

Dinakar Dhurjati
DOCOMO USA Labs
Palo Alto CA 94304, USA
dhurjati@docomolabs-usa.com

Koushik Sen
EECS, UC Berkeley
Berkeley CA, USA
ksen@cs.berkeley.edu

Dachuan Yu
DOCOMO USA Labs
Palo Alto CA 94304, USA
yu@docomolabs-usa.com

***Abstract*—Finite state machines and pushdown systems are frequently used in model based testing. In such testing, the system under test is abstractly modeled as a finite state machine having a finite set of states and a labeled transition relation between the states. A pushdown system, additionally, has an unbounded stack. Test inputs are then generated by enumerating a set of sequences of transitions labels from the model. There has been a lot of research that focussed on generation of test input sequences satisfying various coverage criteria. In this paper, we consider the problem of generating a set of test input sequences that satisfy certain coverage criteria—cover all transition labels or cover all length-n transition label sequences at least once—while minimizing the sum of the length of the sequences in the set. We show that these optimal test input generation problems can be reduced to integer linear programming (ILP) problems. We also prove that our optimal test input generation problems are NP-Complete. We report our experimental results on a prototype implementation for finite states machines.**

I. Introduction

Finite state machines (FSM) [1], [2], [3], [4], [5] and pushdown systems (PDS) with labeled transitions are often used in model based testing. Such models have a finite set of states representing the abstract states of the system under test. Pushdown systems [6], [7] additionally have an unbounded stack. The transitions are labeled by a finite set of symbols. The labels on the transitions denote inputs to the system. A sequence of labels generated from such models are often used for testing. Such sequences are called test input sequences. We consider a particular problem of test input sequence generation for such models—we want to generate a set of test input sequences such that the following two conditions are met:

- the test input sequences must cover all labels of the system, and
- the sum of the length of the test input sequences must be minimal.

We call this the *optimal test input sequence generation problem.*

Our particular test input generation problem is motivated by model based testing of graphical user interfaces (GUIs). A graphical user interface (GUI) is an event-driven system where the system gets input from the user and changes its state in response to each user input. The inputs are called events. Model based testing approaches are often employed to test graphical user interfaces [8], [9], [10], [11]. Such approaches model the behavior of the GUI abstractly using a suitable formalism such as event-flow graphs [8], [9], finite state machines [12], [10], or Petri nets [11]. The models are then used to automatically generate a set of sequences of inputs (or events), called a test-suite. The constituents of a test-suite depend on the coverage criteria used for the generation of the test inputs, where a coverage criterion is a set of rules that determine if a test-suite has adequately tested a GUI. Such coverage criteria include state coverage, event coverage [13], [14], and length-n event sequence coverage [13].

In our setting, we wanted to test the user interfaces of various mobile applications adequately with respect to some coverage criteria. An important restriction that we have is that the number of user interactions or user inputs must be minimal during testing. Moreover, we do not have access to the source code of such mobile applications. Therefore, we need to use a model based testing approach where the application is treated as a black box. We used finite state machines to model the behaviors of the mobile-phone application GUIs, where the states of a model represent the abstract states of the GUI and the transitions in the finite state machine denote various key or touch inputs that can be performed on the GUI. For certain GUIs, we found that finite state machines are not sufficient for modeling purposes; therefore, we used recursive state machines such as pushdown systems as models. Note that pushdown systems are more expressive than finite state machines—such systems have infinite state space due to the presence of an unbounded stack. One coverage criterion that we use in the generation of test inputs is that all labels or inputs in the model must be exercised at least once. This real-world setup motivated our optimal test input generation problem.

In this paper, we describe algorithms that reduce the optimal test input generation problem for finite state machines and pushdown systems to an integer linear programming (ILP) problem, an optimization problem known to be NP-complete. We also prove that our optimal test generation problem is NP-complete; therefore, we cannot develop an asymptotically better algorithm than the one we propose. Apart from its real-world motivation, our problem of optimal test input generation for finite state machines and pushdown systems is in itself an interesting theoretical problem. We believe that these particular problems and their solutions could be applied to other domains.

Note that in our particular problem setting, we consider a minimal coverage criteria: all inputs must be covered at least once. However, this minimal coverage criteria may not be adequate under other circumstances. Previous research [13] has proposed the use of other powerful coverage criteria such as one where all length-n input sequences must be covered at least once. In this paper, we also consider the problem of generating a minimal set of test input sequences for a finite state machine so that all length-n input sequences are covered at least once. We prove that this problem is also NP-complete and show that the problem can be reduced to an integer linear programming problem.

We have only implemented the optimal test input generation algorithm for finite state machines in a prototype tool and our experiments on a few models show that we can effectively generate optimal test input sequences for these models within a few seconds despite the fact that the problem is NP-complete. A key advantage of reducing our optimal test generation problem to integer linear programming problem is that there are heuristics based efficient solvers available both freely and commercially and we can readily use those solvers to make our test generation implementation efficient.

Related Work

Our optimal test input generation problem for FSM is most closely related to two problems: the Chinese postman problem [15] and the Rural Chinese Postman problem [16]. In Chinese postman problem [15] one needs to cover all transitions irrespective of their labels. The problem is solvable in polynomial time. However, unlike our problem, a solution to the Chinese postman problem results in longer test input sequences. The Rural Chinese postman problem is a well known NP-hard problem where a subset of transitions are required to be traversed at minimal cost [16]. Our optimization problem is different—we require at least one transition for each label to be traversed, but we do not specify exactly which transitions need to be traversed. The above two problems have been formulated and solved for finite state machines and have not been considered for pushdown systems.

State machines have been commonly used to model and generate tests for software designs in general and GUIs in particular [17], [18]. The key idea here is to model GUIs as a state machine with each GUI event triggering a transition in the machine. A path of transitions in the state machine represents a test case for the GUI. The test coverage criteria used is typically event pair coverage or more generally covering all event sequences of length-n.

Early research on testing of finite state machines has focused on conformance testing (see [19] for a survey) where the goal of the testing is to check if an implementation, i.e. a "black box", conforms to a specification given in the form of a finite state machine with inputs and outputs (e.g. Mealy machines) under certain assumptions. Another line of research has focussed on the state identification problem [19]—if a finite state machine with inputs and outputs is in an unknown state, what sequence of inputs can be applied so that the unknown state can be identified from its input/output behavior. Both of these testing methods are concerned with models with inputs and outputs and they test if the input/output behavior matches the specification. In our optimal test generation problem, we are only concerned with input sequence generation and their optimality. We expect that the output will be checked manually (or visually in the case of a GUI). Therefore, none of the solutions from conformance testing can be applied to our problem.

II. Model Definitions

We formally define finite state machines and pushdown systems.

Definition 1 (Finite State Machines): A *finite state machine* (FSM) is a tuple $T = (Q, L, \longrightarrow, q_0)$, where

- Q is a finite set of states,
- L is a finite set of labels,
- $\longrightarrow \subseteq Q \times L \times Q$ is a transition relation, and
- $q_0 \in Q$ is the initial state.

We also define the following notations and terms:

1) We use $q \xrightarrow{a} q'$ to denote $(q, a, q') \in \longrightarrow$.
2) A string of labels $\tau = a_1 a_2 \ldots a_n$ is a *trace* of an FSM $T = (Q, L, \longrightarrow, q_0)$ iff there exists states $q_1, q_2, \ldots, q_n \in Q$ such that $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} q_n$.
3) We use $|\tau|$ to denote the length of the string $\tau \in L^*$.
4) We use $\tau[i]$ to denote the label at location i in τ.
5) We use $[\tau]$ to denote the set of all labels present in τ.
6) We use $\tau_1.\tau_2$ to denote the string obtained by concatenating the strings τ_1 and τ_2.
7) We say τ' is a sub-trace of τ, if and only if there exists $\tau_1, \tau_2 \in L^*$ such that $\tau = \tau_1.\tau'.\tau_2$.
8) We say τ is a sub-trace of the FSM T, if there exists a trace τ' of T such that τ is a sub-trace of τ'.
9) We use $[\tau]_n$ to denote the set of all sub-traces of τ whose length is n.

Definition 2 (Pushdown Systems): A *pushdown system* (PDS) [6] is a tuple $P = (Q, \Gamma, L, \hookrightarrow, q_0, \gamma_0)$, where

- Q is a finite set of control states,
- Γ is a finite stack alphabet,
- L is a finite set of labels,
- $\hookrightarrow \subseteq (Q \times \Gamma) \times L \times (Q \times \Gamma^*)$ is a transition relation,
- $q_0 \in Q$ is the initial state, and
- $\gamma_0 \in \Gamma$ is the initial stack content.

We next define a few notations and terms for pushdown systems:

1) A *configuration* of P is a pair (q, w) where $q \in Q$ is a control state and $w \in \Gamma^*$ is a stack content.
2) We use $(q, \gamma) \overset{a}{\hookrightarrow} (q', w)$ to denote $((q, \gamma), a, (q', w)) \in \hookrightarrow$.
3) If $(q, \gamma) \overset{a}{\hookrightarrow} (q', w)$ and $v \in \Gamma^*$, then the pushdown system can transition from configuration $(q, \gamma v)$ to configuration (q, wv) in one step and we denote this transition by $(q, \gamma v) \xrightarrow{a} (q', wv)$.
4) A string of labels $\tau = a_1 a_2 \ldots a_n$ is a *trace* of a PDS $P = (Q, \Gamma, L, \longrightarrow, q_0, \gamma_0)$ iff there exists configurations $C_1, C_2, \ldots, C_n$ and labels $a_1, a_2, \ldots, a_n$ such that $(q_0, \gamma_0) \xrightarrow{a_1} C_1 \xrightarrow{a_2} C_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} C_n$.
5) A configuration C is reachable iff there exists for some $n \geq 0$ configurations $C_1, C_2, \ldots, C_{n-1}$ and labels $a_1, a_2, \ldots, a_n$ such that $(q_0, \gamma_0) \xrightarrow{a_1} C_1 \xrightarrow{a_2} C_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} C$.
6) We use $\mathcal{C}_P$ to denote the set of all reachable configurations with non-empty stack contents of PDS P. Note that a PDS can reach a configuration whose stack is empty. We do not include such configurations in $\mathcal{C}_P$.

III. Problem Definition

Without loss of generality, let us assume that all states and labels of an FSM are reachable from the initial state.[1] Similarly, we assume that all control states and labels of a pushdown system are reachable. Once we have a finite state machine T (or a pushdown system P), we generate a set of traces such that the sum of the length of all traces in the set is minimal and either (1) each label in L is present in at least one trace in the set, or (2) each sub-trace of T (or P) of length n, where n is a given finite positive integer, is present in at least one trace in the set. We next define these problems formally.

Problem 3: (Optimal Trace Generation for an FSM) Given an FSM $T = (Q, L, \longrightarrow, q_0)$, find a set of traces $\{\tau_1, \tau_2, \ldots, \tau_m\}$ of T such that $L = [\tau_1] \cup [\tau_2] \cup \ldots \cup [\tau_m]$ and $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ is minimal.

For example, an optimal set of traces for the model in Figure 1 is {`select unselect edit unedit, select edit copy edit cut edit paste`}.

Problem 4: (Trace Generation Decision for an FSM) Given an FSM $T = (Q, L, \longrightarrow, q_0)$ and an integer k, find a set of traces $\{\tau_1, \tau_2, \ldots, \tau_m\}$ of T such that $L = [\tau_1] \cup [\tau_2] \cup \ldots \cup [\tau_m]$ and $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ is less than or equal to k.

Note that the optimal trace generation problem is concerned only with covering all the labels in L. Such an optimal set of label sequences will guarantee event-coverage as defined in [13]. The optimal trace generation problem can also be generalized to length-n event coverage [13] as follows.

[1]If for an FSM $T = (Q, L \longrightarrow, q_0)$, only $Q' \subset Q$ and $L' \subset L$ is reachable from the initial state q_0, then we generate test input sequences for the FSM $T' = (Q', L', \longrightarrow', q_0)$, where $\longrightarrow' = \longrightarrow \cap (Q' \times L' \times Q')$.

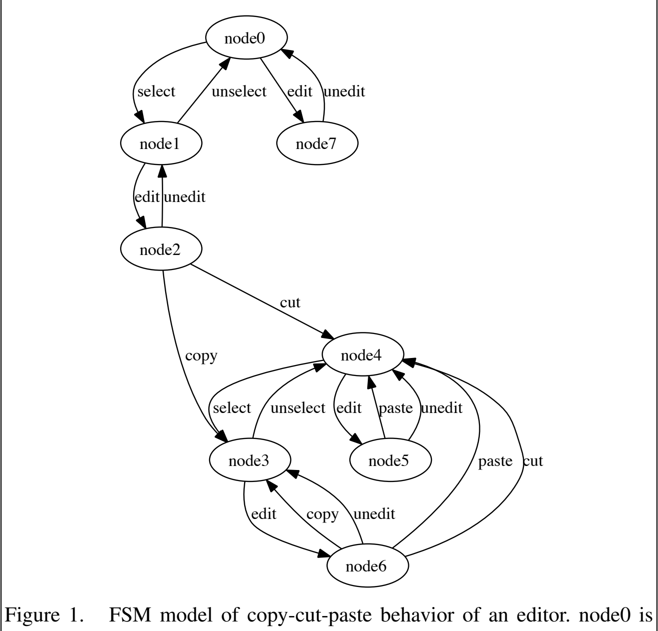

Figure 1. FSM model of copy-cut-paste behavior of an editor. node0 is the initial state.

Problem 5: (Generalized Optimal Trace Generation for an FSM) Given an FSM $T = (Q, L, \longrightarrow, q_0)$ and a constant n, find a set of traces $\{\tau_1, \tau_2, \ldots, \tau_m\}$ of T such that $L' = [\tau_1]_n \cup [\tau_2]_n \cup \ldots \cup [\tau_m]_n$ and $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ is minimized, where L' is the set of all sub-traces of T of length n.

Note that the optimal trace generation problem (i.e. Problem 3) is a special case of the generalized optimal trace generation problem (i.e. Problem 5) with $n = 1$.

The decision problem corresponding to the generalized optimal trace generation problem is as follows.

Problem 6: (Generalized Trace Generation Decision for an FSM) Given an FSM $T = (Q, L, \longrightarrow, q_0)$ and a constant n and a positive integer k, find a set of traces $\{\tau_1, \tau_2, \ldots, \tau_m\}$ of T such that $L' = [\tau_1]_n \cup [\tau_2]_n \cup \ldots \cup [\tau_m]_n$ and $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ is less than or equal to k, where L' is the set of all sub-traces of T of length n.

Similarly, we can define the optimal trace generation problem for a PDS as follows.

Problem 7: (Optimal Trace Generation for a PDS) Given a PDS $P = (Q, \Gamma, L, \longrightarrow, q_0, \gamma_0)$, find a set of traces $\{\tau_1, \tau_2, \ldots, \tau_m\}$ of P such that $L = [\tau_1] \cup [\tau_2] \cup \ldots \cup [\tau_m]$ and $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ is minimized.

The decision version of this problem is as follows.

Problem 8: (Trace Generation Decision for a PDS) Given a PDS $P = (Q, \Gamma, L, \longrightarrow, q_0, \gamma_0)$ and an integer k, find a set of traces $\{\tau_1, \tau_2, \ldots, \tau_m\}$ of P such that $L = [\tau_1] \cup [\tau_2] \cup \ldots \cup [\tau_m]$ and $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ is less than or equal to k.

IV. Optimal Trace Generation Algorithm for an FSM

We next describe algorithms to solve both the optimal trace generation and the generalized optimal trace generation problems and prove that both problems are NP-complete. For simplicity of exposition, we first describe the reduction for the optimal trace generation problem, which is a special case of the generalized optimal test generation problem. In our tool we have implemented this reduction. The reduction works as follows. If $\{\tau_1, \tau_2, \ldots, \tau_m\}$ is an optimal set of traces, then we first show that there is a bound on m and the maximum length of a trace. We then create a set of 0-1 variables to denote that a label is present at location i in the j^{th} trace. We set up a set of constraints on these variables to ensure that the traces are generated by the FSM and minimize the sum of the length of these traces. The bound on m and the maximum length of a trace is given by the following simple lemma.

Lemma 9: Given an FSM $T = (Q, L, \longrightarrow, q_0)$, if there exists a set of traces $\{\tau_1, \tau_2, \ldots, \tau_m\}$ of T such that $L = [\tau_1] \cup [\tau_2] \cup \ldots \cup [\tau_m]$ and $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ is minimized, then $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ is less than or equal to $|Q|.|L|$ and $m \leq |L|$.

Proof: Any state $q \in Q$ can reached by a trace of length at most $|Q|$ since we assume that all states and transitions are reachable from the initial state in the FSM. Therefore, the minimum length of any trace containing a label $a \in L$ is bounded by $|Q|$. Since in the worst case, we will have a trace for each label in L, $m \leq |L|$ and $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ is less than or equal to $|Q|.|L|$. ■

We next describe an algorithm to solve the optimal trace generation problem. The algorithm first creates and solves an integer linear programming (ILP) problem and then uses the solution of the ILP problem to compute the optimal set of traces. Let $T = (Q, L, \longrightarrow, q_0)$ be an FSM. We want to find a set of traces $\{\tau_1, \tau_2, \ldots, \tau_m\}$ of T such that $L = [\tau_1] \cup [\tau_2] \cup \ldots \cup [\tau_m]$ and $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ is minimized. Let k be an upper bound on the optimization function $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ and p be an upper bound on m. Then by Lemma 9, $k = |Q|.|L|$ and $p = |L|$. For each $q, q' \in Q$, $1 \leq i \leq k$, $1 \leq j \leq p$, and $a \in L$ create a variable $X_{qq'aij}$. Any $X_{qq'aij}$ can assume a value of 0 or 1. Intuitively, if $X_{qq'aij}$ is 1, then it denotes that the j^{th} trace in the set of optimal traces has the label a at location i. Consider the integer linear programming (ILP) problem shown in Figure 2. We next state the intuition behind each constraint in the ILP.

- Constraint (1) and (2) ensure that the solution to ILP considers only valid traces of the FSM i.e. if a transition does not exist in the FSM it is not considered in the solution of the ILP.
- Constraint (3) ensures that the first transition in any trace must be from the state q_0.
- Constraint (4) ensures that for a given i and j, there is at most one $X_{qq'aij}$ such that $X_{qq'aij} = 1$.
- Constraint (5) ensures that if the j^{th} trace has a label a' at location i and $i > 1$, then it must have some label a at location $(i-1)$ i.e. a trace constructed from the solution of the ILP is valid trace of the FSM.
- Constraint (6) ensures that each label $a \in L$ is covered by at least one trace.

Let a solution of this ILP assign $x_{qq'aij}$ to each $X_{qq'aij}$. Then the optimal set of traces is

$$\begin{aligned}\mathcal{T} = \{\tau_j \mid\ & \exists a_1, \ldots, a_l \in L, q_1, \ldots, q_l \in Q \\ & \text{such that } \tau_j = a_1 \ldots a_l \\ & \text{and for all } i \in [1, l]\ .\ x_{q_{i-1}q_i a_i ij} = 1 \\ & \text{and } (l = k \text{ or } \forall q \in Q, a \in L\ .\ x_{q_l q a (l+1) j} = 0)\}\end{aligned}$$

We next prove the correctness of the algorithm.

Theorem 10: The trace generation decision problem for an FSM is in NP.

Proof:

[Claim 1] Each sequence of labels in $\mathcal{T}$ is a trace of T. By constraints (1) and (2) of the ILP, $x_{qq'aij} = 1$ implies that $q \xrightarrow{a} q'$. Since for all $i \in [1, l]\ .\ x_{q_{i-1}q_i a_i ij} = 1$, we have $q_0 \xrightarrow{a_1} q_1 \ldots \xrightarrow{a_l} q_l$. Therefore, each $a_1 \ldots a_l$ is a trace of T.

[Claim 2] For each $i \in [1, k], j \in [1, p]$, if $x_{qq'aij} = 1$ for some $q, q' \in Q, a \in L$, then for all $q_1, q_2 \in Q, a' \in L$, if $q \neq q_1$ or $q' \neq q_2$ or $a \neq a'$, then $x_{q_1 q_2 a' ij} = 0$. This follows from constraint (4) of the ILP.

[Claim 3] If for some $q, q' \in Q, a \in L, i \in [1, k], j \in [1, p]$, we have $x_{qq'aij} = 1$, then the trace $\tau_j \in \mathcal{T}$ has a at location i. The proof is as follows. Since $x_{qq'aij} = 1$, by constraint (5) and (2) of the ILP, if $i > 1$, we can find $q'_0, q'_1, \ldots, q'_{i-1}, q'_i \in Q$ and $a_1, a_2, \ldots, a_{i-1}, a_i \in L$ such that $q = q'_{i-1}$, $q' = q'_i$, $a = a_i$, and $x_{q'_0 q'_1 a_1 1 j} = 1$, $x_{q'_1 q'_2 a_2 2 j} = 1$, …, $x_{q'_{i-1} q'_i a_i ij} = 1$. Further, if $x_{q'_r q'_{r+1} a_{r+1} (r+1) j} = 1$, then for all $q'', q''' \in Q, a' \in L$, if $q'_r \neq q''$ or $q'_{r+1} \neq q'''$ or $a' \neq a_{r+1}$, then $x_{q'' q''' a' (r+1) j} = 0$ by **[Claim 2]**. Therefore, if $q'_0 = q_0$, then $a_1 a_2 \ldots a_i$ is the prefix of the trace τ_j. The case $q'_0 \neq q_0$ is not possible by constraint (3) of the ILP. Therefore, the trace τ_j has a at location i.

[Claim 4] The sum of the length of all traces in $\mathcal{T}$ is equal to the sum of all $x_{qq'aij}$. By the construction of $\mathcal{T}$ (i.e. for each a_i in τ_j, we have $x_{q_{i-1}q_i a_i j} = 1$), we have the sum of the length of all traces is less than or equal to the sum of all $x_{qq'aij}$. By **[Claim 2]** and **[Claim 3]**, the sum of the length of all traces is greater than or equal to the sum of all $x_{qq'aij}$. Therefore, the sum of the length of all traces in $\mathcal{T}$ is equal to the sum of all $x_{qq'aij}$.

[Claim 5] For each $a \in L$ there is a trace in $\mathcal{T}$ that contains a. By constraint (6) of the ILP, there must exist $q, q' \in Q, i \in$

$$\text{minimize} \sum_{q,q' \in Q, a \in L, i \in [1,k], j \in [1,p]} X_{qq'aij}$$

$$\text{subject to}$$

$$X_{qq'aij} = 0 \text{ for all } q, q' \in Q, a \in L, i \in [1,k], \text{ and not } q \xrightarrow{a} q' \quad (1)$$

$$X_{qq'aij} \leq 1 \text{ for all } q, q' \in Q, a \in L, i \in [1,k], \text{ and } j \in [1,p] \quad (2)$$

$$X_{qq'a1j} = 0 \text{ for all } q \in Q \setminus \{q_0\}, q' \in Q, j \in [1,p], \text{ and } a \in L \quad (3)$$

$$\sum_{q,q' \in Q, a \in L} X_{qq'aij} \leq 1 \text{ for all } i \in [1,k] \text{ and } j \in [1,p] \quad (4)$$

$$X_{qq'aij} \geq X_{q'q''a'(i+1)j} \text{ for all } q, q', q'' \in Q, a, a' \in L, i \in [1,k-1], \text{ and } j \in [1,p] \quad (5)$$

$$\sum_{q,q' \in Q, i \in [1,k], j \in [1,p]} X_{qq'aij} \geq 1 \text{ for all } a \in L \quad (6)$$

Figure 2. Integer Linear Program for the Optimal Trace Generation Problem

$[1,k], j \in [1,p]$ such that $x_{qq'aij} = 1$. Therefore, by **[Claim 3]**, there is a trace in $\mathcal{T}$ that contains a.

[Claim 6] The sum of the length of all traces in $\mathcal{T}$ is optimal. We prove this by contradiction. By **[Claim 4]** and **[Claim 5]**, $\mathcal{T}$ is a set of traces of T such that for each label $a \in L$, there is a trace in $\mathcal{T}$ that contains a and the sum of the length of all traces in $\mathcal{T}$ is minimized in the ILP. However, it does not imply $\mathcal{T}$ is the set of traces of T whose sum of length of traces is minimal. Assume that $\mathcal{T}' = \{\tau_1', \ldots, \tau_r'\}$ be a set of traces such that $\mathcal{T}'$ is a solution of the optimal trace generation problem of T and the sum of the length of all traces in $\mathcal{T}'$ is strictly less than that of $\mathcal{T}$. Let a trace τ_j' be of the form $a_1^j \ldots a_{m_j}^j$. Then we can find $q_1^j, \ldots, q_{m_j}^j \in Q$ such that $q_0 \xrightarrow{a_1^j} q_1^j \ldots \xrightarrow{a_{m_j}^j} q_{m_j}^j$. Set $X_{q_{i-1}^j q_i^j a_i^j ij}$ to 1 if and only if $j \in [1,r]$ and $i \in [1,m_j]$. Any other $X_{qq'aij}$ is set to 0. It is easy to see that such an assignment satisfies all six constraints of the ILP. Therefore, it forms a solution of the ILP. This implies that the original solution from which we constructed $\mathcal{T}$ is not minimum—a contradiction.

By claims 5 and 6, we proved that $\mathcal{T}$ is a set of optimal traces for T. Since finding a solution of an ILP is NP-complete and the size of ILP is polynomial in the size of the FSM (i.e. the reduction from the optimal trace generation problem to ILP is polynomial time), the optimal trace generation problem is in NP. ■

Theorem 11: The trace generation decision problem for an FSM is NP-hard.

Proof: The weighted set cover problem is defined as follows. Given a universe X and a family S of subsets of X and an integer k, find a subfamily $C \subseteq S$ such that $X = \bigcup_{c \in C} c$ and $\sum_{c \in C} cost(c) \leq k$, where $cost : S \to \mathbb{R}$ maps each set in S to a positive real. The weighted set cover problem is NP-complete [20].

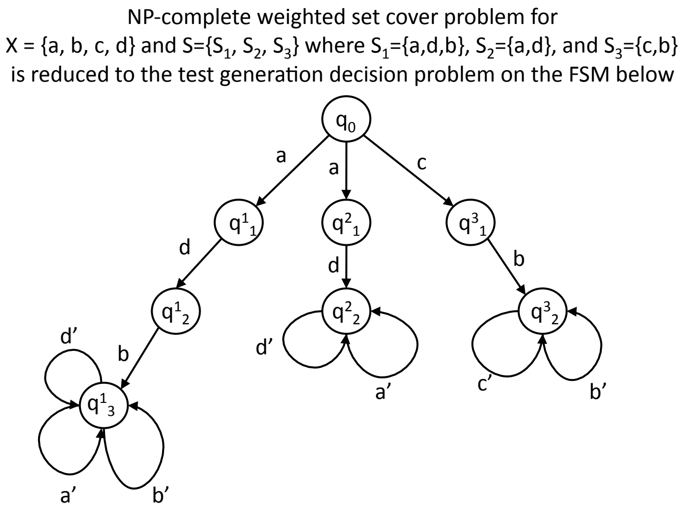
NP-complete weighted set cover problem for
X = {a, b, c, d} and S={S1, S2, S3} where S1={a,d,b}, S2={a,d}, and S3={c,b}
is reduced to the test generation decision problem on the FSM below
q0
a
a
c
q11
q21
q31
d
d
b
q12
q22
q32
d'
b
d'
a'
c'
b'
q13
a'
b'

Figure 3. Construction example for NP-hardness proof. A weighted set cover problem reduced to test generation decision problem on an FSM.

Consider a weighted set cover problem (X, S, k) where $cost(c)$ is defined as the cardinality of the set c. We will reduce this weighted set cover problem to a trace generation decision problem.

Construct an FSM $T = (Q, L, \longrightarrow, q_0)$ as follows. Let $L = X \cup X'$ where $e \in X$ iff $e' \in X$. Suppose $S = \{S_1 \ldots, S_n\}$ and suppose each S_i be $\{e_1^i, \ldots, e_{m_i}^i\}$. Then let $Q = \{q_0\} \cup \bigcup_{i \in [1,n]} \bigcup_{j \in [1,m_i]} \{q_j^i\}$. For each $i \in [1,n]$, let $q_{m_i}^i \xrightarrow{a'} q_{m_i}^i$ for each $a \in S_i$. For each $i \in [1,n]$ and for each $j \in [1, m_i - 1]$, let $q_j^i \xrightarrow{e_{j+1}^i} q_{j+1}^i$. For each $i \in [1,n]$, let $q_0 \xrightarrow{e_1^i} q_1^i$. An example of this reduction is shown in Figure 3. Next we prove that the trace generation decision

$$\text{minimize} \sum_{q,q' \in Q, a \in L, i \in [1,k], j \in [1,p]} X_{qq'aij}$$

subject to

$$X_{qq'aij} = 0 \text{ for all } q, q' \in Q, a \in L, i \in [1,k], \text{ and not } q \xrightarrow{a} q' \quad (1)$$

$$X_{qq'aij} \leq 1 \text{ for all } q, q' \in Q, a \in L, i \in [1,k], \text{ and } j \in [1,p] \quad (2)$$

$$X_{qq'a1j} = 0 \text{ for all } q \in Q \setminus \{q_0\}, q' \in Q, j \in [1,p], \text{ and } a \in L \quad (3)$$

$$\sum_{q,q' \in Q, a \in L} X_{qq'aij} \leq 1 \text{ for all } i \in [1,k] \text{ and } j \in [1,p] \quad (4)$$

$$X_{qq'aij} \geq X_{q'q''a'(i+1)j} \text{ for all } q, q', q'' \in Q, a, a' \in L, i \in [1,k-1], \text{ and } j \in [1,p] \quad (5)$$

$$0 \leq Y_{ij\tau} \leq 1 \text{ for all } i \in [1,k-n], j \in [1,p], \text{ and } \tau \in L' \quad (6)$$

$$\sum_{q,q' \in Q, r \in [1,n]} X_{qq'\tau[r](i+r)j} \geq n.Y_{ij\tau} \text{ for all } i \in [1,k-n], j \in [1,p], \text{ and } \tau \in L' \quad (7)$$

$$\sum_{i \in [1,k], j \in [1,p]} Y_{ij\tau} \geq 1 \text{ for all } \tau \in L' \quad (8)$$

Figure 4. Integer Linear Program for the Generalized Optimal Trace Generation Problem

problem on T and $k+|X|$ has a solution if an only if the set cover problem for X, S, and k has a solution. A proof of the above claim will prove that the trace generation problem is NP-hard.

(proof of [only if]). First, we show that if the trace generation decision problem on T and $k+|X|$ has a solution, then there exists another solution to the same problem where

1) **[Condition 1].** the traces in the solution have exactly one occurrence of each label in X', and
2) **[Condition 2].** each trace in the solution is of the form $\tau_1.\tau_2$ (i.e. concatenation of τ_1 and τ_2), where τ_1 has labels from X and τ_2 has labels from X'. Further, there exists an $S_i \in S$ such that $[\tau_1] = S_i$.

If the traces in a solution on T and $k + |X|$ has more than one occurrence of a label, say $a' \in X'$, then we can safely remove all such labels except one from the traces. The resultant traces will still be a solution of T and $k+|X|$ because each transition with label $a' \in X'$ creates a self loop on some state of the form $q^i_{m_i}$, where $i \in [1,n]$ and such transitions could be made the last transition of a trace.

Note that each trace of T has the form $\tau_1.\tau_2$, where the labels in τ_1 are in X, the labels in τ_2 are in X', $|\tau_1| \geq 1$, and $|\tau_2| \geq 0$. This is because by the construction of the FSM, a trace must have a prefix $e^i_1 e^i_2 \ldots e^i_p$ (i.e. $q_0 \xrightarrow{e^i_1} q^i_1 \xrightarrow{e^i_2} \ldots \xrightarrow{e^i_p} q^i_p$) for some $i \in [1,n]$ and $p \leq m_i$. If $p = m_i$, then the rest of the trace will loop over the state $q^i_{m_i}$ and will be a string in $\{e'^i_1, \ldots, e'^i_{m_i}\}^*$.

Assume the solution on T and $k + |X|$ has a trace $\tau_1.\tau_2$, such that $[\tau_1] \neq S_i$ for all $i \in [1,n]$. Therefore, τ_1 is of the form $e^i_1 e^i_2 \ldots e^i_p$ for some $i \in [1,n]$ and $p \leq m_i$. If $p = m_i$, the $[\tau_1] = S_i$—a contradiction. Therefore, $p < m_i$ and τ_2 is empty. In this case, we claim that we can safely remove the trace $\tau_1.\tau_2$ from the solution and get another solution.

If the above claim is not true, then there is a label $a \in [\tau_1]$ which is not present in the remaining traces of the solution. However, since τ_2 is empty, there must be trace $\tau'_1.\tau'_2$ in the remaining traces that contains a', the primed version of a. If a' is present in the trace, then by the construction of the FSM, a must be present in τ'_1—a contradiction. Therefore, our claim holds. This implies that we can remove all traces τ, where τ has labels only in X and $[\tau_1] \neq S_i$ for all $i \in [1,n]$, from the solution and obtain a smaller solution. Therefore, if there is a solution on T and $k+|X|$, then there is another solution on T and $m+|X|$ where the above two conditions hold.

Now we show that if there is a solution on T and $k+|X|$ that satisfies the above three conditions, then there is a solution of the set covering problem on X, S, and k. By **[Condition 2]**, each trace in the solution is of the form $\tau_1.\tau_2$ where $[\tau_1] = S_i$ for some $i \in [1,n]$. Let the traces in the solution be $\tau^1_1.\tau^1_2, \ldots, \tau^r_1.\tau^r_2$. Then each $[\tau^j_1] = S_i$ for some $j \in [1,r]$ and $i \in [1,n]$. Further, the traces being a solution must contain all labels in $X \cup X'$. Therefore, $[\tau^1_1] \cup \ldots \cup [\tau^r_1] = X$. Therefore, $[\tau^1_1], \ldots, [\tau^r_1]$ is a set covering of X, S. The set covering has a cost of at most k because $|\tau^1_1| + \ldots + |\tau^r_1| = (k + |X|) - |X|$ by **[Condition 1]**. Therefore, we have a solution of the set covering problem on X, S, and k.

(proof of [if]). Let $\{S_{i_1}, \ldots, S_{i_r}\}$ be a solution of the set covering problem on X, S, and k. Given each S_{i_j} is in S, $e^{i_j}_1, \ldots, e^{i_j}_{m_{i_j}}, e'^{i_j}_1, \ldots, e'^{i_j}_{m_{i_j}}$ is a trace of T by construction of the FSM. The set of all such traces contain all labels in X and X'. Further, the cost of all traces is $2k$. From these trace we can remove labels in X' so that each label in X' occurs exactly once in the traces. The resultant traces will have a total cost of $k + |X|$. Therefore, the traces form a solution of the trace generation decision problem on X, S, and $k + |X|$. ■

V. Generalized Optimal Trace Generation Algorithm for an FSM

Consider a generalized optimal trace generation problem, i.e. given an FSM $T = (Q, L, \longrightarrow, q_0)$ and a constant n, find a set of traces $\{\tau_1, \tau_2, \ldots, \tau_m\}$ of T such that $L' = [\tau_1]_n \cup [\tau_2]_n \cup \ldots \cup [\tau_m]_n$ and $|\tau_1| + |\tau_2| + \ldots + |\tau_m|$ is minimized, where L' is the set of all sub-traces of T of length n. In order to solve this problem, we construct another FSM $T' = (Q, L', \longrightarrow', q_0)$ as follows.

Define $\longrightarrow_0 = \longrightarrow$. Construct $\longrightarrow_i$ for $i > 0$ as follows.

$$\begin{aligned}\longrightarrow_i = \{ \quad & (q, \tau.a, q') \mid \exists q'' \in Q \text{ such that} \\ & (q, \tau, q'') \in \longrightarrow_{i-1} \text{ and } (q'', a, q') \in \longrightarrow \}\end{aligned}$$

Note that this construction can be done using dynamic programming. Then define $\longrightarrow' = \longrightarrow_n$ and L' as follows.

$$L' = \{\tau \in L^n \mid \exists q, q' \in Q \text{ and } (q, \tau, q') \in \longrightarrow'\}$$

L' represents the set of all sub-traces of length n of the FSM T. The construction of T' takes $O(|\longrightarrow|^n)$ time (note that n is constant). Further, $|L'| \leq L^n$ as $L' \subseteq L^n$ and $|\longrightarrow'| \leq |Q|.|L'|.|Q|$ as $\longrightarrow' \subseteq Q \times L' \times Q$. Therefore, the size of T' is polynomial in the size of T. Set $k = |L'|.|Q|$ and $p = |L'|$.

For each $q, q' \in Q$, $1 \leq i \leq k$, $1 \leq j \leq p$, and $a \in L$ create a variable $X_{qq'aij}$. Further, for each $1 \leq i \leq k$, $1 \leq j \leq p$, and $\tau \in L'$, create a variable $Y_{ij\tau}$. Consider the integer linear programming (ILP) problem shown in Figure 4. This ILP is same as the ILP in Figure 2, except that we replace constraint (6) in the latter by constraints (6), (7), and (8) in the former. These new three constraints ensure that each sub-trace of T of length n, i.e. each sub-trace in L' is present at least once in the optimal set of traces. More specifically, a variable $Y_{ij\tau}$ can assume a value of 0 or 1, which is ensured by constraint (6) of the ILP. If $Y_{ij\tau} = 1$, then the j^{th} trace has the sub-trace τ starting at the location i. Constraint (7) ensure this condition. Constraint (8) in the ILP ensures that for each $\tau \in L'$, there is at least one trace that contains τ.

Let a solution of this ILP assign $x_{qq'aij}$ to each $X_{qq'aij}$. Then the optimal set of traces is

$$\begin{aligned}\mathcal{T} = \{\tau_j \mid \quad & \exists a_1, \ldots, a_l \in L, q_1, \ldots, q_l \in Q, j \in [1, p] \\ & \text{such that } \tau_j = a_1 \ldots a_l \text{ and} \\ & \text{for all } i \in [1, l] \ . \ x_{q_{i-1}q_ia_iij} = 1 \text{ and} \\ & (l = k \text{ or } \forall q \in Q, a \in L \ . \ x_{q_lqa(l+1)j} = 0)\}\end{aligned}$$

Theorem 12: The generalized trace generation decision problem for an FSM is in NP.

The proof of this theorem is very similar to the proof of the Theorem 10—we need to modify **[Claim 4]** appropriately. We skip the proof in the interest of space.

Theorem 13: The generalized trace generation decision problem for an FSM is NP-hard.

Proof: This follows from the proof of Theorem 11 because the trace generation decision is a generalized trace generation problem with $n = 1$. ■

VI. Optimal Trace Generation Algorithm for a PDS

We next describe a reduction of the optimal trace generation problem for a PDS to an ILP. The reduction depends on the following two crucial lemmas, which upper bound the sum of the length of the optimal set of traces.

Lemma 14: Given a $q \in Q$ and a $\gamma \in \Gamma$, if a configuration of the form $(q, \gamma w)$ is reachable in the PDS $P = (Q, \Gamma, \hookrightarrow, q_0, \gamma_0)$ from the initial configuration (q_0, γ_0), then there exists a configuration of the form $(q, \gamma v)$ such that the configuration can be reached in at most $(|Q|.|\Gamma| + 1)^2.(|\Gamma| + 1)$ steps.

Proof: Given a pushdown system $P = (Q, \Gamma, \hookrightarrow, q_0, \gamma_0)$, we use a P-automaton to represent the set of reachable configurations with non-empty stack contents.

A P-automaton is a tuple $A = (S, \Gamma, \delta, f)$, where S is a finite set of states such that $Q \times \Gamma \subseteq S$, $\delta \subseteq S \times (\Gamma \cup \{\epsilon\}) \times S$ is a transition relation, and $f \in S$ is the final state. We define the transition relation $\rightsquigarrow \ \subseteq S \times \Gamma^* \times S$ as the smallest relation satisfying:

- $s \overset{\epsilon}{\rightsquigarrow} s$ for each $s \in S$,
- if $(s, \gamma, s') \in \delta$ then $s \overset{\delta}{\rightsquigarrow} s'$, and
- if $s \overset{w}{\rightsquigarrow} s''$ and $s'' \overset{\gamma}{\rightsquigarrow} s'$ then $s \overset{w\gamma}{\rightsquigarrow} s'$.

A P-automaton accepts a configuration $(q, \gamma w)$ iff $(q, \gamma) \overset{w}{\rightsquigarrow} f$.

We construct a P-automaton $A = (S, \Gamma, \delta, f)$ that accepts all reachable configurations with non-empty stacks of P (i.e. accepts $\mathcal{C}_P$) as follows. We assume that each transition rule $(q, \gamma) \overset{a}{\hookrightarrow} (q, w)$ of P satisfies $|w| \leq 2$. Note that we do not loose generality due to this restriction because any pushdown system can be transformed into an equivalent PDS of this form. We denote the relation $(\overset{\epsilon}{\rightsquigarrow})^* \overset{\gamma}{\rightsquigarrow} (\overset{\epsilon}{\rightsquigarrow})^*$ by $\overset{\gamma}{\Longrightarrow}$.

Initially, S is the empty set. For each $q \in Q$ and $\gamma \in \Gamma$, add (q, γ) to the set S. Add f to S.

Add $((q_0, \gamma_0), \epsilon, f)$ to δ. Add new transitions to δ according to the following rules *iteratively* until no more new transition can be added to δ:

1) If $(q, \gamma) \overset{a}{\hookrightarrow} (q', \epsilon)$ and $(q, \gamma) \overset{w}{\rightsquigarrow} f$ for some $w \in \Gamma^*$, then for all $s \in S$ such that $(q, \gamma) \overset{\gamma'}{\rightsquigarrow} s$, add $((q', \gamma'), \epsilon, s)$ to δ.
2) If $(q, \gamma) \overset{a}{\hookrightarrow} (q', \gamma')$ and $(q, \gamma) \overset{w}{\rightsquigarrow} f$ for some $w \in \Gamma^*$, then add $((q', \gamma'), \epsilon, (q, \gamma))$ to δ.
3) If $(q, \gamma) \overset{a}{\hookrightarrow} (q', \gamma'\gamma'')$ and $(q, \gamma) \overset{w}{\rightsquigarrow} f$ for some $w \in \Gamma^*$, then add $((q', \gamma'), \gamma'', (q, \gamma))$ to δ.

The P-automaton thus constructed accepts all reachable configurations of the PDS $P = (Q, \Gamma, \hookrightarrow, q_0, \gamma_0)$.

Maximum number of transitions that can be added to δ is $(|Q|.|\Gamma|+1)^2.(|\Gamma|+1)$. This is because S has $|Q|.|\Gamma|+1$ states and between any pair of states there can be $|\Gamma|+1$ transitions. Therefore, the above 3 rules can be applied at most $(|Q|.|\Gamma|+1)^2.(|\Gamma|+1)$ times. If a transition of the form $((q,\gamma),\gamma',s)$ is added in the k^{th} iteration, then by induction on the number of iterations we conclude that a configuration of the form $(q,\gamma v)$ is reachable in the PDS in at most k steps. k is upper bounded by $(|Q|.|\Gamma|+1)^2.(|\Gamma|+1)$. ■

Lemma 15: Given a PDS $P = (Q, \Gamma, \hookrightarrow, q_0, \gamma_0)$, if there exists a set of traces $\{\tau_1, \tau_2, \ldots, \tau_m\}$ of P such that $L = [\tau_1] \cup [\tau_2] \cup \ldots \cup [\tau_m]$ and $|\tau_1|+|\tau_2|+\ldots+|\tau_m|$ is minimized, then $|\tau_1|+|\tau_2|+\ldots+|\tau_m|$ is less than or equal to $|Q|^2.(|\Gamma|+1)^3.|L|$ and $m \leq |L|$.

Proof: Given a $q \in Q$, a $\gamma \in \Gamma$, and a transition of the form $((q,\gamma) \overset{a}{\hookrightarrow} (q',w))$, if a configuration of the form $(q,\gamma v)$ is reachable in the PDS in at most k steps, then the transition can be reached in at most $k+1$ steps. Therefore, by Lemma 14, any transition can be reached in $(|Q|.|\Gamma|+1)^2.(|\Gamma|+1)+1$ steps. Since in the worst case, we will have a trace to reach each label in L, $m \leq |L|$ and $|\tau_1|+|\tau_2|+\ldots+\tau_m| \leq ((|Q|.|\Gamma|+1)^2.(|\Gamma|+1)+1).|L|$ which is less than or equal to $|Q|^2.(|\Gamma|+1)^3.|L|$. ■

As in FSM, the optimal trace generation algorithm for PDS first creates and solves an integer linear programming (ILP) and then uses the solution of the ILP to compute the optimal set of traces. Let $P = (Q, \Gamma, \hookrightarrow, q_0, \gamma_0)$ be a PDS. We want to find a set of traces $\{\tau_1, \tau_2, \ldots, \tau_m\}$ of P such that $L = [\tau_1] \cup [\tau_2] \cup \ldots \cup [\tau_m]$ and $|\tau_1|+|\tau_2|+\ldots+|\tau_m|$ is minimized. Let K be an upper bound on the optimization function $|\tau_1|+|\tau_2|+\ldots+|\tau_m|$ and p be an upper bound on m. Then by Lemma 15, $K = |Q|^2.(|\Gamma|+1)^3.|L|$ and $p = |L|$.

We create a set of 0 or 1 valued variables as follows.

- For each $1 \leq i \leq K+1$, $1 \leq j \leq K+2$, $1 \leq k \leq p$, and $\gamma \in \Gamma \cup \{\top\}$ create a variable $X_{ijk\gamma}$. $X_{ijk\gamma} = 1$ denotes that the i^{th} configuration of the k^{th} trace has the stack symbol γ at the stack position j. $\top$ denotes the empty stack symbol.
- For each $q \in Q$, $i \in [1, K+1]$, and $k \in [1,p]$, create a variable Z_{qik}. $Z_{qik} = 1$ denotes that the i^{th} configuration in the k^{th} trace has the state q.
- For each $i \in [1,K]$, $j \in [1,K+1]$, $k \in [1,p]$, and each rule r in $\hookrightarrow$, create a variable Y_{ijkr}. $Y_{ijkr} = 1$ denotes that in the k^{th} trace the i^{th} transition is triggered by the rule r and the stack has j elements at the time of transition.
- For each $i \in [1,K]$, $j \in [1,K+1]$, $k \in [1,p]$, $q \in Q$, and $\gamma \in \Gamma \cup \{\top\}$, create a variable $W_{ijkq\gamma}$. $W_{ijkq\gamma} = 1$ means that the in the k^{th} trace the i^{th} transition does not change the configuration. The first occurrence of $W_{ijkq\gamma} = 1$ in the k^{th} trace thus denotes that the k^{th} trace has a length of $i-1$.

Consider the integer linear programming (ILP) problem shown in Figure 5. We next state the intuition behind each constraint in the ILP.

- The length of any trace is bounded by K. Therefore, the stack can have at most $K+1$ elements in a trace.
- The first three constraints ensure that the initial configuration of any trace is (q_0, γ_0).
- Contraint (4) ensures that a configuration at a location i in the k^{th} trace has a unique state.
- Constraints (5) ensures that if the configuration at the i^{th} location in the k^{th} trace has a stack with j elements, then all the elements in the stack above j elements are $\top$, i.e. the empty stack element.
- Constraint (6) ensures that the configuration at the i^{th} location in the k^{th} trace has a unique stack element at the j^{th} location in the stack of the configuration.
- Constraints (7), (8), and (9) encodes the transition rules in $\hookrightarrow$.
- Constraints (10) and (13) ensure that if the end of the k^{th} trace is reached after $(i-1)$ transitions, then the configuration remains unchanged from i^{th} location in the trace.
- Constraints (11) and (12) ensure that the stack remains unchanged after each transition except for the top element and elements beyond the top element in the stack.
- Constraint (14) ensures that the i^{th} transition in the k^{th} trace is unique.
- Constraint (15) ensures that a trace contains a transition whose label is a for each $a \in L$.

Let a solution of this ILP assign y_{ijkr} to each Y_{ijkr}. Then the optimal set of traces is

$$\begin{aligned}\mathcal{T} = \{\tau_k \mid\ & \exists a_1, \ldots, a_l \in L, k \in [1,p] \text{ such that} \\ & \tau_k = a_1 \ldots a_l \text{ and for all } i \in [1,l] \ .\ y_{ijkr} = 1 \\ & \text{for some } j \text{ and label of } r \text{ is } a_i \text{ and} \\ & (l = K \text{ or } w_{(l+1)jkq\gamma} = 1 \text{ for some } j,\, q, \text{ and } \gamma\}\end{aligned}$$

Theorem 16: The trace generation decision problem for a PDS is in NP.

The proof of this theorem follows from the above construction of the integer linear programming. The integer linear programming has size polynomial in K and p. Therefore, the reduction takes polynomial amount of time. We skip the formal proof of this theorem in the interest of space—the intuitive explanation of the constraints in the ILP provides information to carry out the formal proof.

Theorem 17: The trace generation decision problem for a PDS is NP-hard.

Proof: This follows from the proof of Theorem 11 because an FSM is a spacial case of a PDS where each transition is of the form $(q,\gamma_0) \overset{a}{\hookrightarrow} (q',\gamma_0)$, i.e. the stack remains unchanged. ■

$$\text{minimize} \sum_{i\in[1,K],j\in[1,K+1],k\in[1,p], \text{ and for each rule } r} Y_{ijkr}$$

subject to

$$Z_{q_0 1k} = 1 \text{ for all } k \in [1,p] \quad (1)$$

$$X_{11k\gamma_0} = 1 \text{ for all } k \in [1,p] \quad (2)$$

$$X_{12k\top} = 1 \text{ for all } k \in [1,p] \quad (3)$$

$$\sum_{q\in Q} Z_{qik} = 1 \text{ for all } i \in [1,K+1] \text{ and } k \in [1,p] \quad (4)$$

$$X_{ijk\top} \le X_{i(j+1)k\top} \text{ for all } i \in [1,K+1], j \in [1,K+1], \text{ and } k \in [1,p] \quad (5)$$

$$\sum_{\gamma\in\Gamma\cup\{\top\}} X_{ijk\gamma} = 1 \text{ for all } i \in [1,K+1], j \in [1,K+2], \text{ and } k \in [1,p] \quad (6)$$

$$Z_{qik} + Z_{q'(i+1)k} + X_{ijk\gamma} + X_{i(j+1)k\top} + X_{(i+1)jk\top} \ge 5Y_{ijkr}$$
$$\text{for all } i \in [1,K], j \in [1,K+1], k \in [1,p], \text{ and each rule } r \text{ of the form } (q,\gamma) \overset{a}{\hookrightarrow} (q',\epsilon) \quad (7)$$

$$Z_{qik} + Z_{q'(i+1)k} + X_{ijk\gamma} + X_{i(j+1)k\top} + X_{(i+1)jk\gamma'} + X_{(i+1)(j+1)k\top} \ge 6Y_{ijkr}$$
$$\text{for all } i \in [1,K], j \in [1,K+1], k \in [1,p], \text{ and each rule } r \text{ of the form } (q,\gamma) \overset{a}{\hookrightarrow} (q',\gamma') \quad (8)$$

$$Z_{qik} + Z_{q'(i+1)k} + X_{ijk\gamma} + X_{i(j+1)k\top} + X_{i(j+2)k\top}$$
$$+X_{(i+1)jk\gamma''} + X_{(i+1)(j+1)k\gamma'} + X_{(i+1)(j+2)k\top} \ge 8Y_{ijkr}$$
$$\text{for all } i \in [1,K], j \in [1,K], k \in [1,p], \text{ and each rule } r \text{ of the form } (q,\gamma) \overset{a}{\hookrightarrow} (q',\gamma'\gamma'') \quad (9)$$

$$Z_{qik} + Z_{q(i+1)k} + X_{ijk\gamma} + X_{i(j+1)k\top} + X_{(i+1)jk\gamma} + X_{(i+1)(j+1)k\top} \ge 6W_{ijkq\gamma}$$
$$\text{for all } q \in Q, \gamma \in \Gamma \cup \{\top\}, i \in [1,K], j \in [1,K+1], \text{ and } k \in [1,p] \quad (10)$$

$$X_{i(j-1)k\gamma} - X_{(i+1)(j-1)k\gamma} \le 1 - \Big(\sum_{\ell\in[j,K+1] \text{ and for each rule } r} Y_{i\ell kr} + \sum_{\ell\in[j,K+1],q\in Q,\gamma'\in\Gamma\cup\{\top\}} W_{i\ell kq\gamma'}\Big)$$
$$\text{for all } i \in [1,K], j \in [2,K+1], k \in [1,p], \text{ and } \gamma \in \Gamma \quad (11)$$

$$X_{(i+1)(j-1)k\gamma} - X_{i(j-1)k\gamma} \le 1 - \Big(\sum_{\ell\in[j,K+1] \text{ and for each rule } r} Y_{i\ell kr} + \sum_{\ell\in[j,K+1],q\in Q,\gamma'\in\Gamma\cup\{\top\}} W_{i\ell kq\gamma'}\Big)$$
$$\text{for all } i \in [1,K], j \in [2,K+1], k \in [1,p], \text{ and } \gamma \in \Gamma \quad (12)$$

$$W_{ijkq\gamma} \le W_{(i+1)jkq\gamma} \text{ for all } q \in Q, \gamma \in \Gamma \cup \{\top\}, i \in [1,K], j \in [1,K+1], \text{ and } k \in [1,p] \quad (13)$$

$$\sum_{j\in[1,K+1] \text{ and for each rule } r} Y_{ijkr} + \sum_{j\in[1,K+1],q\in Q,\gamma\in\Gamma\cup\{\top\}} W_{ijkq\gamma} = 1 \text{ for all } i \in [1,K] \text{ and } k \in [1,p] \quad (14)$$

$$\sum_{i\in[1,K],j\in[1,K+1],k\in[1,p], \text{ and for each rule } r \text{ with label } a} Y_{ijkr} \ge 1 \quad (15)$$

Figure 5. Integer Linear Program for the Optimal Trace Generation Problem for a PDS. Each variable in the ILP can have a value 0 or 1.

GUI	FSM Size total (states, transitions)	Time to solve ILP (ms)	Total time (ms)	optimal sequence size
Registration	98 (17, 81)	3811	4139	6
Editor clipboard	31 (10, 21)	1359	1655	11
Mail composer	32 (12, 20)	1187	1499	5
Bookmark manager	60 (19 , 41)	2328	2672	12
Browser navigation	108 (21 , 85)	4139	4529	8
Phone book manager	152 (44, 108)	5982	6388	15
Mail viewer	215 (39, 176)	8576	8981	14

Table I
EXPERIMENTAL RESULTS

VII. EVALUATION

We have implemented the optimal test sequence generation algorithm in a tool. We only focussed on label coverage for an FSM and implemented the algorithm described in Section IV. The tool could be extended to implement the algorithms in Sections V and VI.

In our evaluation, we created FSM models for seven mobile applications: *Registration, Editor clipboard, Mail composer, Mail viewer, Phone book manager, Browser navigation,* and *Bookmark manager*.

We applied our tool to generate optimal input sequences for these seven models. In our experiments, we mainly tried to evaluate the time taken by our tool to generate optimal input sequences for these GUI models.

Table I shows the results of our experiments on a Pentium IV with 2GB RAM. The first column in the table reports the name of the model. The second column reports the size of the FSM model. The third and fourth columns report the time taken in milliseconds to solve the ILP and the total time taken to generate the optimal set of test sequences, respectively. The last column reports the sum of the length of all input sequences in the optimal solution. The table shows that the total time taken for optimal input sequence generation is less than 9 seconds for all the models. This shows that we can generate optimal input sequences for these models in a reasonable amount of time.

VIII. Conclusion

We considered the theoretical problem of generating a set of test input sequences that cover all transition labels or cover all length-n transition label sequences at least once while minimizing the sum of the length of the sequences in the set. We showed that this optimal test input generation problem is NP-complete for both finite state machines and pushdown systems. We experimentally demonstrated that the optimal test input sequence generation algorithm for FSM works efficiently for some benchmark models. In future, we would like to solve the optimal test generation problem for other kinds of coverage criteria. We would also like to develop and evaluate some approximation algorithms for the optimal test generation problems.

References

[1] T. S. Chow, "Testing software design modeled by finite-state machines," *IEEE Transactions on Software Engineering*, vol. 4, no. 3, pp. 178–187, 1978.

[2] J. Offutt, S. Liu, A. Abdurazik, and P. Ammann, "Generating test data from state-based specifications," *The Journal of Software Testing, Verification and Reliability*, vol. 13, pp. 25–53, 2003.

[3] M. Veanes, C. Campbell, W. Grieskamp, W. Schulte, N. Tillmann, and L. Nachmanson, "Model-based testing of object-oriented reactive systems with spec explorer," 2008, pp. 39–76.

[4] J. Dick and A. Faivre, "Automating the generation and sequencing of test cases from model-based specifications," in *Formal Methods Europe on Industrial-Strength Formal Methods (FME 93)*, 1993, pp. 268–284.

[5] S. Helke, T. Neustupny, and T. Santen, "Automating test case generation from Z specifications with Isabelle," in *Lecture Notes in Computer Science*, 1997, pp. 52–71.

[6] J.-M. Autebert, J. Berstel, and L. Boasson, "Context-free languages and pushdown automata," pp. 111–174, 1997.

[7] S. Schwoon, "Model-checking pushdown systems," Ph.D. dissertation, TU München, 2002.

[8] A. M. Memon, "An event-flow model of GUI-based applications for testing," *Software Testing, Verification and Reliability*, vol. 17, no. 3, pp. 137–157, 2007.

[9] A. M. Memon and Q. Xie, "Studying the fault-detection effectiveness of GUI test cases for rapidly evolving software," *IEEE Transactions on Software Engineering*, vol. 31, no. 10, pp. 884–896, 2005.

[10] R. K. Shehady and D. P. Siewiorek, "A methodology to automate user interface testing using variable finite state machines," in *FTCS*, 1997, pp. 80–88.

[11] H. Reza, S. Endapally, and E. Grant, "A model-based approach for testing GUI using hierarchical predicate transition nets," in *International Conference on Information Technology (ITNG'07)*. IEEE Computer Society, 2007, pp. 366–370.

[12] W. M. Newman, "A system for interactive graphical programming," in *Proceedings of the spring joint computer conference (AFIPS '68 (Spring))*. ACM, 1968, pp. 47–54.

[13] A. M. Memon, M. L. Soffa, and M. E. Pollack, "Coverage criteria for GUI testing," in *8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering*. ACM, 2001, pp. 256–267.

[14] F. Belli and C. J. Budnik, "Test minimization for human-computer interaction," *Applied Intelligence*, vol. 26, no. 2, pp. 161–174, 2007.

[15] H. Thimbleby, "The directed Chinese Postman Problem," *In journal of Software: Practice and Experience*, vol. 33, pp. 1081–1096, 2003.

[16] H. A. Eiselt, M. Gendreau, and G. Laporte, "Arc routing problems, part ii: The rural postman problem," *Operations Research*, vol. 43, no. 3, pp. 399–414, 1995.

[17] L. White and H. Almezen, "Generating test cases for GUI responsibilities using complete interaction sequences," in *11th International Symposium on Software Reliability Engineering (ISSRE'00)*. IEEE Computer Society, 2000, p. 110.

[18] F. Belli, "Finite-state testing and analysis of graphical user interfaces," in *12th International Symposium on Software Reliability Engineering (ISSRE'01)*. IEEE Computer Society, 2001, p. 34.

[19] D. Lee and M. Yannakakis, "Principles and methods of testing FSMs: A survey," *Proceedings of IEEE*, vol. 84, pp. 1089–1123, 1996.

[20] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.