Using Distinguishing and UIO Sequences Together in a Checking Sequence

M. Cihan Yalcin[1] and Husnu Yenigun[1]

Faculty of Engineering and Natural Sciences, Sabanci University, Tuzla 34956, Istanbul, Turkey

Abstract. If a finite state machine M does not have a distinguishing sequence, but has UIO sequences for its states, there are methods to produce a checking sequence for M. However, if M has a distinguishing sequence $\bar{D}$, then there are methods that make use of $\bar{D}$ to construct checking sequences that are much shorter than the ones that would be constructed by using only the UIO sequences for M. The methods to applied when a distinguishing sequence exists, only make use of the distinguishing sequences. In this paper we show that, even if M has a distinguishing sequence $\bar{D}$, the UIO sequences can still be used together with $\bar{D}$ to construct shorter checking sequences.

1 Introduction

Finite state machines (FSM) have been successfully used to model the externally observable behavior of systems [1]. Based on the FSM model M of a system under test (SUT) N, a test sequence can be constructed to check if N is implemented correctly [2, 3].

Such a test sequence, which will be called a checking sequence, is a sequence of inputs such that, if N produces the expected outputs then this information provides sufficient evidence to conclude that N is a correct implementation of M. Of course, such a checking sequence cannot be found in general. Two important assumptions are made on N in practice. First assumption is that N is deterministic and does not change during the experiments. The second assumption is that N has at most the same number of states as M. Although the latter assumption seems to be restrictive, this assumption provides a basis to construct a checking sequence. Based on the methods that can generate checking sequences under this assumption, it is possible to extend these methods to generate checking sequences when this assumption is relaxed and N is assumed to have at most $n + \Delta$ states for some constant Δ, where n is the number of states in M (e.g. see [4]).

Basically, a checking sequence consists of parts that challenge N to provide evidence for the correct implementation of every transition in M. To do this, the checking sequence brings N to a state, applies an input at that state (to see if it would produce the correct output), and then it applies a sequence of inputs to recognize the state reached. As we will explain, bringing N to a certain state

is also based on recognizing states, which can only be performed by observing distinct outputs produced to the same input sequence by different states.

Recognizing states can be based on distinguishing sequences [3], a characterization set [3] or unique input-output (UIO) sequences [5]. It is known that a distinguishing sequence may not exist for every minimal FSM [6], and that determining the existence of a distinguishing sequence for an FSM is PSPACE-complete [7]. However, if M has a distinguishing sequence, there are methods already available in the literature (e.g. [3, 8, 9]) to produce a checking sequence in which distinguishing sequences are used to recognize the states. It is quite easy to understand why a distinguishing sequence $\bar{D}$ can be used to recognize a state, since all the states in M produces a different output sequence to the same input sequence $\bar{D}$.

If an FSM M does not have a distinguishing sequence, it is still possible to construct a checking sequence for M. For example in [5] and in [10], it is shown how a checking sequence can be constructed by using UIO sequences, which are sequences that may exist even when a distinguishing sequence is not available. However, the authors of [11] show that, the original method proposed in [5] is not sufficient, and they propose the UIOv method to fix the problems of the method given in [5]. Since the UIO sequences of the states are not necessarily the same, although the response of a state to is UIO $\bar{U}$ is unique in the specification, we have to make sure that no other state produces the same response to $\bar{U}$ in N. As this must be guaranteed for the UIO sequences of all the states, checking sequences based on UIO sequences tend to be longer. Hence the UIOv and the other UIO based methods are considered only when a distinguishing sequence does not exist.

In this paper we propose that, even if there exists a distinguishing sequence for an FSM M, UIO sequences for the states of M (which are guaranteed to exist since M is known to have a distinguishing sequence) can also be used to construct a checking sequence in conjunction with the distinguishing sequence. We explain a method to show how to construct such a checking sequence. We also give an example for which the length of the checking sequence based on the distinguishing sequence and UIO sequences is less than the length of the checking sequence based on the distinguishing sequence only.

The rest of the paper is organized as follows. Section 2 introduces the concepts used in constructing checking sequences. In Section 3, an existing method to construct checking sequences based on distinguishing sequences is given. Section 4 explains the conditions under which a UIO sequence can be used to recognize states in a checking sequence. In Section 5, we give a modification of the method in Section 3 that constructs checking sequences in which UIO sequences are also used for state recognition. Finally, Section 6 concludes the paper and provides future research directions on the topic.

2 Preliminaries

We directly adopt the formalism and the notation for finite state machines from [12] and include it below for completeness. A deterministic FSM M is defined by a tuple $(S, s_1, X, Y, \delta, \lambda)$ where

- S is a finite set of *states*,
- $s_1 \in S$ is the *initial state*,
- X is the finite *input alphabet*,
- Y is the finite *output alphabet*,
- $\delta : S \times X \to S$ is the *next state function*, and
- $\lambda : S \times X \to Y$ is the *output function*.

Throughout the paper, we use barred symbols (e.g. $\bar{x}, \bar{P}, \ldots$) to denote sequences, and juxtaposition to denote concatenation. The next state function δ and the output function λ can be extended to sequences in a straightforward manner as, for an input symbol $a \in X$, a sequence of inputs $\bar{x} \in X^\star$, and a state $s \in S$,

$$\delta(s, a\bar{x}) = \delta(\delta(s, a), \bar{x}) \text{ and } \lambda(s, a\bar{x}) = \lambda(s, a)\lambda(\delta(s, a), \bar{x})$$

The number of states of M is denoted n and the states of M are enumerated, giving $S = \{s_1, s_2, \ldots, s_n\}$. An FSM is *completely specified* if the functions λ and δ are total.

An FSM, that will be denoted M_0 throughout this paper, is described in Figure 1. Here, $S = \{s_1, s_2, s_3\}$, $X = \{a, b\}$ and $Y = \{0, 1\}$.

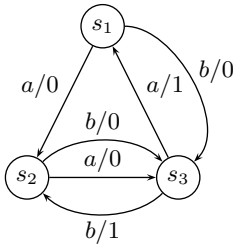

Fig. 1. The FSM M_0

In an FSM M, $s_i \in S$ and $s_j \in S$, $s_i \neq s_j$, are *equivalent* if, $\forall \bar{x} \in X^*$, $\lambda(s_i, \bar{x}) = \lambda(s_j, \bar{x})$. If $\exists \bar{x} \in X^*$ such that $\lambda(s_i, \bar{x}) \neq \lambda(s_j, \bar{x})$ then $\bar{x}$ is said to *distinguish* s_i and s_j. An FSM M is said to be *minimal* if none of its states are equivalent.

A *distinguishing sequence* for an FSM M is an input sequence $\bar{D}$ for which each state of M produces a distinct output. More formally, for all $s_i, s_j \in S$

if $s_i \neq s_j$ then $\lambda(s_i, \bar{D}) \neq \lambda(s_j, \bar{D})$. Thus, for example, M_0 in Figure 1 has distinguishing sequence aa.

A *unique input output sequence* (a UIO sequence, or simply a UIO) for a state s_i of an FSM M is an input sequence $\bar{U}_i$ which distinguishes s_i from the other states. More formally, $\bar{U}_i$ is a UIO for s_i if for all $s_j \in S$, if $s_j \neq s_i$, then $\lambda(s_i, \bar{U}_i) \neq \lambda(s_j, \bar{U}_i)$. Thus, for example, s_3 of M_0 has UIO $\bar{U}_3 = b$.

It is known that some FSMs do not have a distinguishing sequence, and some states do not have UIO sequences. However, when we consider a machine $M = (S, s_1, X, Y, \delta, \lambda)$ with a distinguishing sequence $\bar{D}$ (let $\bar{D}$ be a shortest such sequence), and a state $s_i \in S$ with a UIO sequence $\bar{U}_i$ (let $\bar{U}_i$ be a shortest such sequence), we can easily observe the following fact: $\bar{D}$ distinguishes between all pairs of states (s_i and s_j, $\forall s_i, s_j \in S$), whereas $\bar{U}_i$ distinguishes only between certain pairs of states (s_i and s_j, $\forall s_j \in S$). Hence, $\bar{U}_i$ must be at most as long as $\bar{D}$. In fact, any distinguishing sequence is also a UIO sequence for all the states by definition.

For example, for the state s_3 in M_0 of Figure 1, $\bar{U}_3 = b$ is shorter than the distinguishing sequence $\bar{D} = aa$. However, for the states s_1 and s_2, shortest UIO sequences are of length 2, which is the same as the length of the distinguishing sequence.

Therefore, when we do have a distinguishing sequence for an FSM M, we may be able to find shorter UIO sequences for the states of M. It is this observation that will allow us to form shorter checking sequences, as explained in the rest of the paper.

An FSM M can be represented by a directed graph (*digraph*) $G = (V, E)$ where a set of vertices V represents the set S of states of M, and a set of directed edges E represents all transitions of M. Each edge $e = (v_j, v_k, x/y) \in E$ represents a transition $t = (s_j, s_k, x/y)$ of M from state s_j to state s_k with input x and output y where $s_j, s_k \in S$, $x \in X$, and $y \in Y$ such that $\delta(s_j, x) = s_k$, $\lambda(s_j, x) = y$.

A sequence $\bar{P} = (n_1, n_2, x_1/y_1)(n_2, n_3, x_2/y_2)\ldots(n_{k-1}, n_k, x_{k-1}/y_{k-1})$ of pairwise adjacent edges from G forms a *path* in which each *node* n_i represents a vertex from V and thus, ultimately, a state from S. Here $initial(\bar{P})$ denotes n_1, which is the *initial node* of $\bar{P}$, and $final(\bar{P})$ denotes n_k, which is the *final node* of $\bar{P}$. Two paths $\bar{P}_1$ and $\bar{P}_2$ can be concatenated as $\bar{P}_1\bar{P}_2$ only if $final(\bar{P}_1) = initial(\bar{P}_2)$.

The sequence $\bar{Q} = (x_1/y_1)(x_2/y_2)\ldots(x_{k-1}/y_{k-1})$ is the *label* of $\bar{P}$ and is denoted $label(\bar{P})$. In this case, $\bar{Q}$ is said to *label* the path $\bar{P}$. $\bar{Q}$ is said to be a *transfer sequence* from n_1 to n_k. The path $\bar{P}$ can be represented by the tuple $(n_1, n_k, \bar{Q})$ or by the tuple $(n_1, n_k, \bar{x}/\bar{y})$ in which $\bar{x} = x_1x_2\ldots x_{k-1}$ is the *input portion* of $\bar{Q}$ and $\bar{y} = y_1y_2\ldots y_{k-1}$ is the *output portion* of $\bar{Q}$.

A *tour* is a path whose initial and final nodes are the same. Given a tour $\bar{\Gamma} = e_1e_2\ldots e_k$, $\bar{P} = e_je_{j+1}\ldots e_ke_1e_2\ldots e_{j-1}$ is a path formed by *starting* $\bar{\Gamma}$ with edge e_j, and hence by *ending* $\bar{\Gamma}$ with edge e_{j-1}. An *Euler Tour* is a tour that contains each edge exactly once. A set E' of edges from G is *acyclic* if no tour can be formed using the edges in E'.

A digraph is *strongly connected* if for any ordered pair of vertices (v_i, v_j) there is a path from v_i to v_j. An FSM is *strongly connected* if the digraph that represents it is strongly connected. It will be assumed that any FSM considered in this paper is deterministic, minimal, completely specified, and strongly connected.

Given an FSM M, let $\Phi(M)$ be the set of FSMs each of which has at most n states and the same input and output alphabets as M. Let N be an FSM of $\Phi(M)$. N is *isomorphic* to M if there is a one-to-one and onto function f on the state sets of M and N such that for any state transition $(s_i, s_j, x/y)$ of M, $(f(s_i), f(s_j), x/y)$ is a transition of N. A *checking sequence* of M is an input sequence starting at the initial state s_1 of M that distinguishes M from any N of $\Phi(M)$ that is not isomorphic to M. In the context of testing, this means that in response to this input sequence, any faulty implementation N from $\Phi(M)$ will produce an output sequence different from the expected output, thereby indicating the presence of a fault/faults. As stated earlier, a crucial part of testing the correct implementation of each transition of M in N from $\Phi(M)$ is recognizing the starting and terminating states of the transition which lead to the notions of state recognition and transition verification used in algorithms for constructing checking sequences (for example, [9, 13]).

3 An Existing Approach

In this section, we will present an existing approach for generating checking sequences. The approach is based on distinguishing sequences only, and directly imported from [12] for completeness. After understanding the components (and their purpose) that are put together to form a checking sequence by this approach, it will be easier to understand how we can use UIO sequences instead of some of these components, that will hopefully make the generated checking sequences shorter. In fact, the algorithm for generating a checking sequence that will be proposed in this paper is a modification on the algorithm of [12], which was first given in [13].

3.1 Basics

The checking sequence $\bar{C}$ will be a sequence of inputs to be applied to SUT N, that will identify whether N is a correct implementation of M or not, i.e. whether N is isomorphic to M or not. Suppose that we trace $\bar{C}$ on the digraph $G = (V, E)$ representing M. Since M is deterministic, the trace will correspond to a unique path $\bar{P} = (n_1, n_2, x_1/y_1)(n_2, n_3, x_2/y_2)\ldots(n_{k-1}, n_k, x_{k-1}/y_{k-1})$. Below we will refer to the checking sequence $\bar{C}$ as the input portion of the input/output sequence $\bar{Q}$ which is the label of the path $\bar{P}$.

$\bar{P}$ can also be viewed as the application of $\bar{C}$ to N. In this view, the nodes $n_1, n_2, \ldots, n_k$ (or equivalently the states of N visited during this application) are not known. A checking sequence $\bar{C}$, or equivalently $\bar{P}$, should be designed in such a way that, the inputs and the corresponding outputs should provide

sufficient evidence to let us identify these unknown states that are visited during the application of $\bar{C}$ to N.

If M has a distinguishing sequence $\bar{D}$, then $\bar{D}$ can be used in $\bar{C}$ to help to identify the states. Let us call $\bar{T}_i = \bar{D}/\lambda(s_i, \bar{D})\bar{B}_i$ as a *T–sequence*, where $\bar{B}_i = \bar{I}_i/\lambda(\delta(s_i, \bar{D}), \bar{I}_i)$ for a possibly empty transfer sequence $\bar{I}_i$. For example, for FSM M_0 in Figure 1, if we take $\bar{I}_1$,$\bar{I}_2$ and $\bar{I}_3$ as empty sequences, $\bar{T}_1 = aa/00$, $\bar{T}_2 = aa/01$, $\bar{T}_3 = aa/10$.

Inference Rule IR1: Let $\bar{R}_i = (n_p, n_q, \bar{T}_i)$ be a subpath in $\bar{P}$. Since the response of N to $\bar{D}$ at n_p is $\lambda(s_i, \bar{D})$, this unknown state n_p of N at step p, has some relation to the state s_i of M. Of course, this does not guarantee that n_p is equivalent to the state s_i under the light of this evidence only. N may be a faulty implementation of M, yet it may still have a state that produces the same output $\lambda(s_i, \bar{D})$ to $\bar{D}$. Therefore we only say that, if n_p produces the same output to $\bar{D}$ as s_i, then n_p is *recognized* as state s_i of M in $\bar{Q}$.

Based on the assumption that N does not change during the experiments, the following inference rule can also be used.

Inference Rule IR2: If $\bar{P}_1 = (n_p, n_q, \bar{x}/\bar{y})$ and $\bar{P}_2 = (n_r, n_s, \bar{x}/\bar{y})$ are two subpaths of $\bar{P}$ such that n_p and n_r are recognized as state s_i of M and n_q is recognized as state s_j of M, then n_s is said to be recognized (in $\bar{Q}$) as state s_j of M. Intuitively, this rule says that if $\bar{P}_1$ and $\bar{P}_2$ are labeled by the same input/output sequence and their starting vertices are both recognized as the same state s_i of M, then their terminating vertices correspond to the same state s_j of M.

For N to be a correct implementation of M, first of all, for each state s_i of M, N must have a state which is recognized as s_i. If P has subpaths $\bar{R}_i = (n_p, n_q, \bar{T}_i)$ for all $i \in \{1, 2, \ldots, n\}$, then it will check existence of the corresponding states in N. If N does not produce the expected outputs, then N is a faulty implementation of M. However, if N produces the expected outputs, then for each state s_i in M, N must have at least one state corresponding to (recognized as) s_i. When combined with the assumption that N has at most n states, this will form a one–to–one correspondence between the states of M and the states of N.

As explained in the paragraph above, for each $\bar{T}_i$, $\bar{P}$ will have at least one subpath $\bar{R}_i = (n_p, n_q, \bar{T}_i)$. Based on IR1, $initial(\bar{R}_i)$ will be recognized as s_i. Note that, if there exists another subpath $\bar{R}'_i = (n'_p, n'_q, \bar{T}_i)$, $initial(\bar{R}'_i)$ will again be recognized as s_i. In other words, for every subpath with the label $\bar{T}_i$, the initial node of the subpath will be recognized as s_i. We will abuse the notation and let $initial(\bar{T}_i)$ denote the state s_i. Since, N is deterministic and does not change during experiments, we can also argue that for any subpath $\bar{R}_i$ with the label $\bar{T}_i$, $final(\bar{R}_i)$ will be recognized as the same state s_j, where $s_j = \delta(s_i, \bar{D}\bar{I}_i)$. We will use $final(\bar{T}_i)$ to denote this state s_j. Below we explain how $final(\bar{R}_i)$ can be recognized as well.

In order to recognize $final(\bar{R}_i)$, $\bar{P}$ will include subpaths with the labels as explained below. Let α'–set $A = \{\bar{\alpha}'_1, \bar{\alpha}'_2, \ldots \bar{\alpha}'_q\}$ be a set of input/output sequences such that $\bar{\alpha}'_k$ $(1 \leq k \leq q)$ is the sequence $\bar{T}_{k_1}\bar{T}_{k_2} \ldots \bar{T}_{k_{r_k}}$, for some $1 \leq k_1, k_2, \ldots, k_{r_k} \leq n$, such that $\forall i \in \{1, 2, \ldots r_k - 1\}$, $initial(\bar{T}_{k_{i+1}}) =$

$final(\bar{T}_{k_i})$. Each $\bar{\alpha}'_k$ is called an α'–sequence, and an α'–set A satisfies the following condition [13]: For all $i \in \{1, 2, \ldots, n\}$, there exists a $j \in \{1, 2, \ldots, n\}$ and a $k \in \{1, 2, \ldots, q\}$, such that $\bar{T}_i\bar{T}_j$ is a subsequence of $\bar{\alpha}'_k$. For example $\{\bar{T}_1\bar{T}_3, \bar{T}_3\bar{T}_2, \bar{T}_2\bar{T}_1\}$ is an α'–set for FSM M_0 given in Figure 1.

Lemma 1. *Let $\mathcal{T} = \{\bar{T}_1, \bar{T}_2, \ldots, \bar{T}_n\}$ be a T–set, and $A = \{\bar{\alpha}'_1, \bar{\alpha}'_2, \ldots, \bar{\alpha}'_q\}$ be an α'–set based on $\mathcal{T}$. If $\bar{Q} = label(\bar{P})$ includes all $\bar{\alpha}'_k$, $1 \leq k \leq q$, as a subsequence then:*

1. *For all $k \in \{1, 2, \ldots q\}$, if $(n_p, n_q, \bar{\alpha}'_k)$ is a subpath in $\bar{P}$, then n_p is recognized.*
2. *For all $i \in \{1, 2, \ldots, n\}$, $\bar{T}_i$ is a subsequence in $\bar{Q}$.*
3. *For all $i \in \{1, 2, \ldots, n\}$, if $(n_r, n_s, \bar{T}_i)$ is a subsequence in $\bar{P}$, then n_s is recognized in $\bar{P}$.*
4. *For all $k \in \{1, 2, \ldots q\}$, if $(n_p, n_q, \bar{\alpha}'_k)$ is a subpath in $\bar{P}$, then n_q is recognized.*

Proof. 1. Since $\bar{\alpha}'_k$ starts with a $\bar{T}_i$ that has a prefix $\bar{D}/\lambda(s_i, \bar{D})$, n_p is recognized as s_i in $\bar{Q}$ (IR1).
2. Since for each $\bar{T}_i$, there exists a $\bar{T}_j$ such that $\bar{T}_i\bar{T}_j$ is a subsequence of some $\bar{\alpha}'_k$, which in turn is a subsequence in $\bar{Q}$, $\bar{T}_i$ is a subsequence in $\bar{Q}$.
3. There exists a $\bar{T}_j$ such that $\bar{T}_i\bar{T}_j$ is a subsequence of some $\bar{\alpha}'_k$, which in turn is a subsequence in $\bar{Q}$. In other words, there exists a subpath $(n_p, n_t, \bar{T}_i\bar{T}_j)$ in $\bar{P}$. After dividing this path into two as $(n_p, n_q, \bar{T}_i)(n_q, n_t, \bar{T}_j)$, it is easy to see that, n_p and n_q are recognized as states s_i and s_j respectively. But then, we can use IR2 on $(n_p, n_q, \bar{T}_i)$ and $(n_r, n_s, \bar{T}_i)$ to deduce that n_s is recognized as s_j.
4. Since $\bar{\alpha}'_k$ ends with a $\bar{T}_i$, based on the discussion given in (3) above, n_q is recognized. □

Different α' sets can be found for a given set of T–sequences $\{\bar{T}_1, \bar{T}_2, \ldots, \bar{T}_n\}$. For example $\{\bar{T}_3\bar{T}_2\bar{T}_1\bar{T}_3\}$ and $\{\bar{T}_1\bar{T}_3\bar{T}_2\bar{T}_1\}$ are also α'–sets for M_0 of Figure 1. Since $\bar{C}$ will have the input portion of α'–sequences as subsequences, it may be desirable to minimize the total length of α'–sequences. Note that, this is just a heuristic to minimize the length of $\bar{C}$. In [14] authors explain how to find a set of α'–sequences with a minimal total length from a given set of T–sequences.

Besides these components to recognize the states in N, a checking sequence will also have components to check if the transitions are implemented correctly. We say that the transition $(s_i, s_j, x/y)$ of M is *verified* in $\bar{Q} = label(\bar{P})$ if $(n_p, n_q, x/y)$ is a subpath of $\bar{P}$, n_p is recognized as s_i and n_q is recognized as s_j. n_p will have to be recognized using IR2. n_q can be recognized using IR1, by applying a $\bar{T}_i$. Since α'–sequences start with $\bar{T}_i$'s, they can also be used to recognize the end state of the transitions [13].

In the next section we will explain a method to generate a checking sequence, which is based on Theorem 1.

Theorem 1. *(Theorem 1, [9]) Let $\bar{Q}$ be the label of a path $\bar{P}$ on G representing an FSM M that starts at s_1. If every transition of M is verified in $\bar{Q}$, then the input portion of $\bar{Q}$ is a checking sequence of M.*

3.2 Checking Sequence Construction

In [13], the following method is explained to produce a checking sequence. Given $G = (V, E)$ corresponding to an FSM M, a T–sequence set $\mathcal{T} = \{\bar{T}_1, \bar{T}_2, \ldots, \bar{T}_n\}$, and an α'–set $A = \{\bar{\alpha}'_1, \bar{\alpha}'_2, \ldots, \bar{\alpha}'_q\}$, first another digraph $G' = (V', E')$ is produced by augmenting the digraph G as follows (Figure 2 is the digraph G' corresponding to the digraph G of FSM M_0 given in Figure 1):

a) $V' = V \cup U'$ where $U' = \{v' : v \in V\}$, i.e. for each vertex v in G, there are two copies of v in G'. In Figure 2, the nodes on the left are the nodes in V, and the nodes on the right are the nodes in U'.
b) $E' = E_C \cup E_T \cup E_{\alpha'} \cup E''$ where
   i) $E_C = \{(v'_i, v_j, x/y) : (v_i, v_j, x/y) \in E\}$. The solid edges leaving the nodes on the right in Figure 2 are the edges in E_C.
   ii) $E_T = \{(v_i, v'_j, \bar{T}_i) : \bar{T}_i \in \mathcal{T}, s_i = initial(\bar{T}_i), s_j = final(\bar{T}_i)\}$. For example, since $initial(\bar{T}_1) = s_1$ and $final(\bar{T}_1) = s_2$, there is an edge $(v_1, v'_2, \bar{T}_1)$ in Figure 2.
   iii) $E_{\alpha'} = \{(v_i, v'_j, \bar{\alpha}'_k) : \bar{\alpha}'_k \in A, \bar{\alpha}'_k = \bar{T}_i \ldots \bar{T}_l, initial(\bar{T}_i) = s_i, final(\bar{T}_l) = s_j\}$. For example, in Figure 2 we consider a singleton α'–set $A = \{\bar{\alpha}'_1 = \bar{T}_1\bar{T}_3\bar{T}_2\bar{T}_1\}$. There is an edge $(v_1, v'_3, \bar{\alpha}'_1)$ in Figure 2 since $initial(\bar{T}_1) = s_1$ (the first T–sequence in $\bar{\alpha}'_1$ is $\bar{T}_1$), and $final(\bar{T}_1) = s_3$ (the last T–sequence in $\bar{\alpha}'_1$ is $\bar{T}_1$).
   iv) $E'' \subseteq \{(v'_i, v'_j, x/y) : (v_i, v_j, x/y) \in E\}$. E'' is a subset of the copies of the edges in E placed between the corresponding nodes in U'. E'' is selected in such a way that, $G'' = (U', E'')$ does not have a tour and G' is strongly connected.

We would like to highlight the followings about G':

- The edges in E_C represent the transitions to be verified.
- On a path in G', an edge in E_C will have to be followed by an edge in E_T or $E_{\alpha'}$. Since an α'–sequence also starts with a T–sequence, this means that a transition will always be followed by a T–sequence, hence the end state of the transition will be recognized.
- On a path in G', the nodes in U' will be recognized. If a node v' in U' is reached by using an edge in E_T or an edge $E_{\alpha'}$, it is easy to show that v' is recognized since the final states of T–sequences and α'–sequences are recognized as explained previously in Lemma 1. As long as $G'' = (U', E'')$ is acyclic, it is also guaranteed that v' will be recognized if it is reached by using an edge in E'' (please see the proof of Theorem 2 in [9] for the sketch of a proof of this claim).
- Based on the previous claim, the initial states of the transitions will also be recognized in a path $\bar{P}$ in G', since the edges in E_C representing the transitions always have their initial nodes in U'.

Suppose that we form a path $\bar{P}$ in G' that starts from and ends at v_1 such that, it includes all the edges in $E_{\alpha'}$ (so that the states are recognized), and it also includes all the edges in E_C (so that the transitions are verified). On the

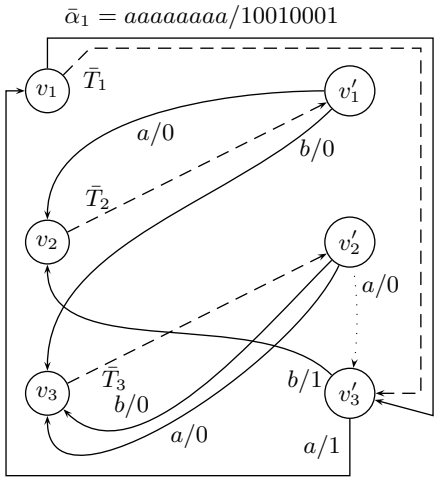

Fig. 2. G' for M_0

basis of Theorem 1, it is argued in [13] that the input portion of the label of such a path $\bar{P}$ which is followed by $\bar{D}$ is a checking sequence of M.

In fact, since we would like to keep the length of the checking sequence small, an optimization is used to find a short path. The approach given in [13] forms a minimal symmetric augmentation G^* of the digraph induced by $E_{\alpha'} \cup E_C$ by adding replications of edges from E' . If G^*, with its isolated vertices removed, is connected, then G^* has an Euler tour. Otherwise, a heuristic such as the one given in [9] is applied to make G^* connected and an Euler tour of this new digraph is formed to find a path from v_1 to v_1.

$$(v_1, v_3', \bar{\alpha}_1')(v_3', v_2, b/1)(v_2, v_1', \bar{T}_2)(v_1', v_2, a/0)(v_2, v_1', \bar{T}_2)(v_1', v_3, b/0)(v_3, v_2', \bar{T}_3)$$
$$(v_2', v_3, a/0)(v_3, v_2', \bar{T}_3)(v_2', v_3, b/0)(v_3, v_2', \bar{T}_3)(v_2', v_3', a/0)(v_3', v_1, a/1)$$

Fig. 3. An tour in G'

The checking sequence constructed based on the tour given in Figure 3 would be the label of the path of Figure 3 followed by $\bar{D}$. Hence the length of the checking sequence is 27.

4 Using UIO Sequences for State Recognition

The method explained in Section 3 uses a distinguishing sequence to recognize the end state of a transition $(s_i, s_j, x/y)$ by applying $\bar{D}$ after the execution of the transition, and by observing the output $\lambda(s_j, \bar{D})$ which is unique among all the states. The purpose of an edge $(v_i, v'_j, \bar{T}_i)$ in G' is twofold: (i) it recognizes the final state of a transition, and (ii) it also recognizes the final state of itself (see Lemma 1). In other words, when the input portion of $\bar{T}_i$ is applied to SUT N and the expected output is observed, we do not only recognize the state before the application, but we also recognize the state that is reached after the application of the input part of $\bar{T}_i$. This is obviously based on the fact that, the input portion of all the α'–sequences are also applied and the expected outputs are observed from N.

A UIO sequence $\bar{U}_j$ for a state s_j also provides a similar information. In other words, to recognize the end state of a transition $(s_i, s_j, x/y)$, one can apply $\bar{U}_j$ after the execution of the transition, and observe the output $\lambda(s_j, \bar{U}_j)$ which is also unique among all the states. Since $\bar{U}_j$ will be at most as long as $\bar{D}$, using UIO sequences instead of distinguishing sequences may shorten the overall checking sequence.

However, for a UIO sequence $\bar{U}_j$ for a state s_j, suppose that $\bar{P}$ contains $(n_p, n_q, \bar{U}_j/\lambda(s_j, \bar{U}_j))$ as a subpath. (i) Can we conclude that n_p must be recognized as s_j? (ii) Can we conclude that n_q must be recognized as $\delta(s_j, \bar{U}_j)$? Below we explain under what conditions both of these questions can be answered positively.

For a sequence $\bar{x} \in X^*$, let $symb(\bar{x}) \subseteq X$ denote the set of input symbols that appear in $\bar{x}$. For example, if $\bar{x} = aba$, then $symb(\bar{x}) = \{a, b\}$.

Theorem 2. *Let $\bar{Q}$ be the label of a path $\bar{P}$ in $G = (V, E)$ corresponding to an FSM M, and $\bar{U}_j$ be a UIO for a state s_j in M. Assume that $\forall x \in symb(\bar{U}_j)$ and for all states s in M, the transition $(s, \delta(s, x), x/\lambda(s, x))$ is verified in $\bar{Q}$. If $(n_p, n_q, \bar{U}_j/\lambda(s_j, \bar{U}_j))$ is a subpath of $\bar{P}$, then n_p is recognized as s_j and n_q is recognized as $\delta(s_j, \bar{U}_j)$.*

We will need the following result to prove Theorem 2.

Lemma 2. *Let $\bar{Q}$ be the label of a path $\bar{P}$ in $G = (V, E)$ corresponding to an FSM M, and $\bar{x}' \in X^*$ be an input sequence. Assume that $\forall x \in symb(\bar{x}')$ and for all states s in M, the transition $(s, \delta(s, x), x/\lambda(s, x))$ is verified in $\bar{Q}$. If $(n_r, n_s, \bar{x}'/\lambda(s', \bar{x}'))$ is a subpath of $\bar{P}$ and n_r is recognized as s', then n_s is recognized as $\delta(s', \bar{x}')$.*

Proof. The proof is based on induction on the length of $\bar{x}'$. When the length of $\bar{x}'$ is 1, i.e. when $\bar{x}' = a$ for some $a \in X$, we have $\bar{P}_1 = (n_r, n_s, a/\lambda(s', a))$ as a subpath in $\bar{P}$. Since $\forall x \in symb(\bar{x}') = \{a\}$ and for all states s in M, the transition $(s, \delta(s, x), x/\lambda(s, x))$ is verified in $\bar{Q}$, there must exist a subpath $\bar{P}_2 = (n_p, n_q, a/\lambda(s', a))$ in $\bar{P}$ such that n_p is recognized as s', and n_q is recognized as $\delta(s', a)$. Using $\bar{P}_1$ and $\bar{P}_2$ and the inference rule IR2, we can deduce that n_s

is recognized as $\delta(s', a)$.
For the inductive step, assume that $\bar{x}' = a\bar{x}''$, in other words we have a subpath $\bar{P}_1 = (n_r, n_s, a\bar{x}''/\lambda(s', a\bar{x}''))$, or equivalently by dividing $\bar{P}_1$ into two, we have the subpaths $\bar{P}_{11} = (n_r, n_t, a/\lambda(s', a))$, $\bar{P}_{12} = (n_t, n_s, \bar{x}''/\lambda(\delta(s', a), \bar{x}''))$. Based on the discussion given in the base step of the proof, n_t is recognized as $\delta(s', a)$. This completes the proof, since n_t is recognized, and $\bar{x}''$ is shorter than $\bar{x}'$. □

We can now go back to the proof of Theorem 2:

Proof (of Theorem 2).
We know that the transitions of all the states for all the input symbols in $\bar{U}_j$ are implemented correctly. Since $\bar{U}_j$ is a UIO sequence for s_j, this means that only the state that should be recognized as state s_j in N produces the output $\lambda(s_j, \bar{U}_j)$ to $\bar{U}_j$. Hence, for the subpath $(n_p, n_q, \bar{U}_j/\lambda(s_j, \bar{U}_j))$ of $\bar{P}$, n_p must be recognized as s_j.
When n_p is recognized, we can use Lemma 2 to show that n_q is also recognized. □

What Theorem 2 suggests is that, when it is guaranteed that the transitions of the states for the input symbols that appear in a UIO sequence $\bar{U}_j$ are verified, then $\bar{U}_j/\lambda(s_j, \bar{U}_j)$ can be used in a checking sequence exactly in the same way and for the same purpose as the T–sequence $\bar{T}_j$. Based on this observation, we will propose a modification on the method given in Section 3.2 for constructing checking sequences.

5 Modified Method for Checking Sequence Construction

The modification will actually be quite intuitive, and very simple for a reader who understands the purposes of the components of the digraph G' given in Section 3.2.

Let us explain the modified method on our running example first. We will provide the method formally later. Consider M_0 in Figure 1, and the digraph G' for M_0 given in Figure 2, and let us focus on the edge $(v'_2, v_3, a/0)$. In G', this edge will have to followed by the edge $(v_3, v'_2, \bar{T}_3)$, which would both recognize v_3 as s_3, and also recognize v'_2 as s_2.

The state s_3 in M_0 has the UIO $\bar{U}_3 = b$. Based on the discussions given Section 4, we can add outgoing edge to $(v_3, v'_2, \bar{U}_3/\lambda(s_3, \bar{U}_3))$ in G', since $\bar{U}_3/\lambda(s_3, \bar{U}_3)$ can also be used in a similar way as $\bar{T}_3$ is used in G' (Figure 4).

However, we also require that the input symbols that appear in the UIO sequences that are used to recognize states to be verified. We have to avoid verifying an edge depending on the correctness of itself. In other words, there are some transitions with the input b whose final states are s_3. Namely the edges $(v'_1, v_3, b/0)$ and $(v'_2, v_3, b/0)$ in Figure 4. The verification of the corresponding transitions of these edges will have to be performed in the conventional way. In other words, we will need to force to use the edge with the label $\bar{T}_3$ when these

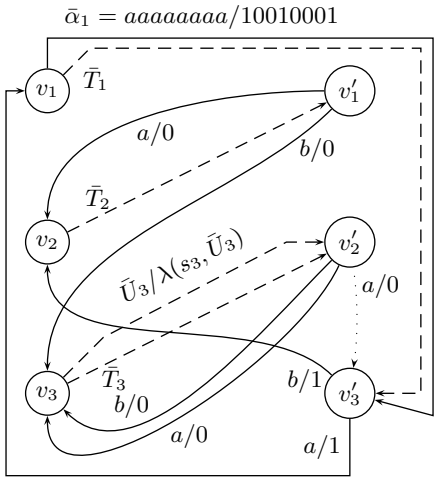

Fig. 4. The first (unseccuessful) attempt for the modification

two edges are used to reach v_3, to guarantee that b transitions of s_2 and s_3 are verified.

This can be achieved by having two copies of v_3 in G'. One copy of v_3 will be the usual v_3 that already exists in G', and have the outgoing edge with label $\bar{T}_3$. The other copy of v_3 (say v_3^U) will have an outgoing edge with the label $\bar{U}_3/\lambda(s_3, \bar{U}_3)$. Note that, having this edge as the only outgoing edge of v_3^U would force $\bar{U}_3$ to be used to recognize the node v_3^U. However, if we also add an edge (v_3^U, v_3, ϵ), this would introduce the possibility and the flexibility of using $\bar{T}_3$ (and any α'–sequence originating from v_3 if there were any) to recognize the node v_3^U. The final digraph G' that will be used for our example is given in Figure 5.

We now explain the modified method more formally. Given $G = (V, E)$ corresponding to an FSM M, a T–sequence set $\mathcal{T} = \{\bar{T}_1, \bar{T}_2, \ldots, \bar{T}_n\}$, and an α'–set $A = \{\bar{\alpha}'_1, \bar{\alpha}'_2, \ldots, \bar{\alpha}'_q\}$, we will again generate a digraph $G' = (V', E')$ by augmenting G. Assume that we are also given a set of UIO sequences for some of the states to recognize these states. Let $\mathcal{U} = \{\bar{U}_{i_1}, \bar{U}_{i_2}, \ldots, \bar{U}_{i_k}\}$ be such a set of UIO sequences. Suppose that the UIO sequence $\bar{U}_{i_j} \in \mathcal{U}$ is a UIO sequence for the state s_{i_j}. Let $symb(\mathcal{U}) = symb(\bar{U}_{i_1}) \cup symb(\bar{U}_{i_2}) \cup \cdots \cup symb(\bar{U}_{i_k})$ below.

a) $V' = V \cup V^U \cup U'$ where
   i) $U' = \{v' : v \in V\}$. For each $v \in V$, we have a copy of v in U'.
   ii) $V^U = \{v_j^u : v_j \in V, j \in \{i_1, i_2, \ldots, i_k\}\}$
   If $\mathcal{U}$ includes a UIO sequence $\bar{U}_j$ for the state s_j, then for the corresponding node $v_j \in V$, we create a copy v_j^U in V^U.

b) $E' = E_C \cup E_T \cup E_U \cup E_\epsilon \cup E_{\alpha'} \cup E''$ where

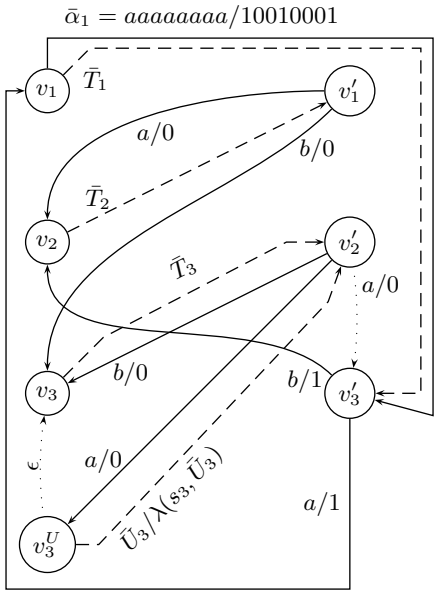

Fig. 5. G' after modification

i) $E_C = \{(v_i', v_j^U, x/y) : (v_i, v_j, x/y) \in E, x \notin symb(\mathcal{U}), j \in \{i_1, i_2, \ldots, i_k\}\} \cup \{(v_i', v_j, x/y) : (v_i, v_j, x/y) \in E, (x \in symb(\mathcal{U}) \text{ or } j \notin \{i_1, i_2, \ldots, i_k\})\}$. E_C will again correspond to the transitions to be verified. However, we have now two different types of edges in E_C. If the input symbol of the transition is not one of the input symbols in $symb(\mathcal{U})$ (i.e. it does not appear in any of the UIO sequences provided), and there exists a UIO sequence $\bar{U}_j \in \mathcal{U}$ for the recognition of final state s_j of the transition, then the edge is connected to the node v_j^U. Otherwise, the edge will be connected to the node v_j.

ii) $E_T = \{(v_i, v_j', \bar{T}_i) : \bar{T}_i \in \mathcal{T}, s_i = initial(\bar{T}_i), s_j = final(\bar{T}_i)\}$. There is no change in this component.

iii) $E_U = \{(v_i^U, v_j', \bar{U}_i/\lambda(s_i, \bar{U}_i)) : \bar{U}_i \in \mathcal{U}, s_j = \delta(s_i, \bar{U}_i)\}$. If $\bar{U}_i \in \mathcal{U}$ is a UIO sequence for a state s_i, then we place the outgoing edge from v_i^U for the UIO recognition, hence it has the label $\bar{U}_i/\lambda(s_i, \bar{U}_i)$.

iv) $E_\epsilon = \{(v_i^U, v_i', \epsilon) : \bar{U}_i \in \mathcal{U}\}$. If $\bar{U}_i \in \mathcal{U}$ is a UIO sequence for a state s_i, then we insert an ϵ edge from v_i^U to v_i for increased flexibility of using $\bar{T}_i$ from v_i (or an α'–sequence outgoing from v_i, if exists) for recognizing the end state of an edge in E_C that ends in v_i^U.

v) $E_{\alpha'} = \{(v_i, v_j', \bar{\alpha}_k') : \bar{\alpha}_k' \in A, \bar{\alpha}_k' = \bar{T}_i \ldots \bar{T}_l, initial(\bar{T}_i) = s_i, final(\bar{T}_l) = s_j\}$. There is no change in this component.

vi) $E'' \subseteq \{(v_i', v_j', x/y) : (v_i, v_j, x/y) \in E\}$. E'' is again a subset of the copies of the edges in E placed between the corresponding nodes in U'. E'' is

selected in such a way that, $G'' = (U', E'')$ does not have a tour and G' is strongly connected.

As in the case of the previous method, a tour is found in G' that includes all the edges in $E_{\alpha'} \cup E_C$. Figure 6 shows a tour in G' given in Figure 5. The tour includes the necessary edges, and hence can be used to form a checking sequence as explained below.

$$(v_1, v_3', \bar{\alpha}_1')(v_3', v_2, b/1)(v_2, v_1', \bar{T}_2)(v_1', v_2, a/0)(v_2, v_1', \bar{T}_2)(v_1', v_3, b/0)(v_3, v_2', \bar{T}_3)$$
$$(v_2', v_3^U, a/0)(v_3^U, v_2', \bar{U}_3/\lambda(s_3, \bar{U}_3))(v_2', v_3, b/0)(v_3, v_2', \bar{T}_3)(v_2', v_3', a/0)(v_3', v_1, a/1)$$

Fig. 6. An tour in the modified G'

The checking sequence constructed based on the tour given in Figure 6 would be the label of the path of Figure 6 followed by $\bar{D}$. Hence the length of the checking sequence is 26. The length of the new checking sequence is 1 less than the length of the checking sequence produced by the previous method.

6 Conclusion and Future Work

We have shown that, for a FSM M with a distinguishing sequence, UIO sequences for states can also be used to recognize states in a checking sequence. Existing methods in the literature use only distinguishing sequences to recognize states in a checking sequence when M has a distinguishing sequence. However, when an FSM M has a distinguishing sequence, the states of M may have shorter UIO sequences. Therefore using UIO sequences instead of distinguishing sequences may result in shorter checking sequences. We have given an example of such a case, where the length of the checking sequence is reduced.

We have also shown how a checking sequence that uses UIO sequences for state recognition can be constructed by modifying an already existing checking sequence construction technique, which is based on using distinguishing sequences only for state recognition.

It is assumed that we are given a set of UIO sequences to be used for state recognition. Further research is required to compute a set of UIO sequences for an FSM M, that will help shortening the length of a checking sequence. Intuitively, if for a state s_j, there is a large number of transitions incoming into the state s_j, and if we can find a UIO $\bar{U}_j$ for s_j such that a small number of different input symbols appear in $\bar{U}_j$, then heuristically, using $\bar{U}_j$ for recognizing s_j seems to be promising to reduce the length of the checking sequence.

This paper shows that it is possible to decrease the length of a checking sequence using the method proposed. However, an experimental study would also be useful to understand the magnitude of a typical reduction.

References

1. Tanenbaum, A.S.: Computer Networks. 3rd edn. Prentice Hall International Editions, Prentice Hall (1996)
2. Gill, A.: Introduction to the Theory of Finite–State Machines. McGraw–Hill, New York (1962)
3. Hennie, F.C.: Fault–detecting experiments for sequential circuits. In: Proceedings of Fifth Annual Symposium on Switching Circuit Theory and Logical Design, Princeton, New Jersey (1964) 95–110
4. Lee, D., Yannakakis, M.: Principles and methods of testing finite–state machines – a survey. Proceedings of the IEEE **84**(8) (1996) 1089–1123
5. Sabnani, K., Dahbura, A.: A protocol test generation procedure. Computer Networks **15** (1988) 285–297
6. Kohavi, Z.: Switching and Finite Automata Theory. McGraw–Hill, New York (1978)
7. Lee, D., Yannakakis, M.: Testing finite state machines: state identification and verification. IEEE Trans. Computers **43**(3) (1994) 306–320
8. Gonenc, G.: A method for the design of fault detection experiments. IEEE Transactions on Computers **19** (1970) 551–558
9. Ural, H., Wu, X., Zhang, F.: On minimizing the lengths of checking sequences. IEEE Transactions on Computers **46**(1) (1997) 93–99
10. Aho, A., Dahbura, A., Lee, D., Uyar, M.: An optimization technique for protocol conformance test generation based on UIO sequences and rural chinese postman tours. IEEE Transactions on Communications **39**(11) (1991) 1604–1615
11. Chan, W., Vuong, C., Otp, M.: An improved protocol test generation procedure based on UIOS. ACM SIGCOMM Computer Communication Review **19**(4) (1989) 283–294
12. Tekle, K.T., Ural, H., Yalcin, M.C., Yenigun, H.: Generalizing redundancy elimination in checking sequences. In: 20th International Symposium on Information and Computer Sciences (ISCIS). Volume 3733 of Lecture Notes in Computer Science., Istanbul, Turkey (2005) 915–926
13. Hierons, R.M., Ural, H.: Reduced length checking sequences. IEEE Transactions on Computers **51**(9) (2002) 1111–1117
14. Hierons, R.M., Ural, H.: Optimizing the length of checking sequences. IEEE Transactions on Computers (2004) accepted for publication.