ABSTRACT
A number of methods have been published to construct checking sequences for testing from Finite State Machine-based specifications. Many of these methods require the existence of a preset distinguishing sequence in the model. In this paper, we show that usually an adaptive distinguishing sequence is sufficient for these methods to work. This result is significant because adaptive distinguishing sequences are strictly more common and up to exponentially shorter than preset ones.

Categories and Subject Descriptors
D.2.4 [Software Engineering] Software & Program Verification – formal methods, validation; D.2.5 [Software Engineering] Testing and Debugging – testing tools (data generators, coverage testing)

General Terms
Reliability, Theory, Verification.

Keywords
Model-based testing, finite state machines, checking sequence construction, adaptive distinguishing sequences, preset distinguishing sequences.

1. INTRODUCTION
State-based systems are often specified using Finite State Machine (FSM) model or its extensions such as Specification and Description Language (SDL) or State-Charts. As a model-based testing approach, testing from an FSM has been utilized in a number of application domains such as telecommunications systems, communications protocols, embedded systems using state-based languages, object-oriented systems, web services, pattern matching and machine learning. When testing from an FSM $M$ it is common to consider a fault model $\Phi(M)$ for $M$ representing the types of faults that can occur and an implementation of FSM $M$ to be tested is taken as an unknown FSM $N$ in $\Phi(M)$. In order to determine whether a given implementation $N$ of $M$ is faulty with respect to $M$, a checking sequence (an input sequence constructed from $M$ by taking into consideration $\Phi(M)$ the fault model for $M$) is applied to $N$. Any discrepancy between the expected output sequence and the actual output sequence produced by $N$ in response to the application of the checking sequence is sufficient to determine that $N$ is a faulty implementation of $M$ [1, 2].

The methods reported in the literature for constructing a checking sequence from an FSM $M$ use a distinguishing sequence [1, 3], a set of characterizing sequences [4, 5], or a set of unique input/output (UIO) sequences [6, 7]. These methods recognize each distinct state in $N$ as a distinct state of $M$ and verify that each transition of $M$ is correctly implemented in $N$ [8]. The methods that use a set of characterizing sequences or UIO sequences either require that there is a reset that is known to have been implemented correctly or produce a checking sequence whose length is at least exponential in terms of the number of states of $M$ and the lengths of the sequences used.

Among these previous methods, we will consider the ones using a distinguishing sequence $D$ of $M$ which is an input sequence for which the response of each state of $M$ is distinct. Such methods [9, 10, 11, 12, 13] achieve recognition of a state of $N$ as a state of $M$ by applying $DT$ (i.e. the distinguishing sequence $D$ followed by a transfer sequence $T$) at that state of $N$ and perform the verification of a transition of $M$ from state $s_i$ to state $s_j$ under input $x$ in $N$ by 1) transferring $N$ to the state recognized as state $s_i$ of $M$; 2) checking the output produced by $N$ in response to $x$ to be as specified in $M$ (to detect an output fault); and 3) recognizing the state reached by $N$ after the application of $x$ as state $s_j$ of $M$ (to detect a transfer fault). Step 3) is realized by applying a $DT$ at the state reached by $N$ after the application of $x$. Step 1) is realized indirectly by making sure that the state reached by the application of $DT$ at each state is recognized by applying another $D$ at some point in the checking sequence. This facilitates the use of a $DT$ at a state such that the state reached by the application of $DT$ is the state to which $N$ needs to be transferred (i.e., state recognized as the starting state of a transition to be verified).

For the construction of a checking sequence of an FSM, these methods utilize the digraph representation of the FSM and first form three sets of paths: set $A$ of state recognition paths which are used to recognize each state of the FSM (e.g., $\alpha$-sequences, $\alpha'$-sequences or $\alpha$-elements in different methods); set $B$ of transition verification paths which are used to verify each transition of the FSM (e.g., test segments); set $C$ of transfer paths which are used to concatenate paths in $A$ and $B$. Checking sequence generation methods place various constraints on the selection of $C$. Earlier
methods use some predefined strategies to reduce the length of transfer paths [2, 14]. These strategies do not guarantee that transfer paths found yield minimized checking sequences. An optimization model has been proposed to solve this problem [9] and it is adopted by successive checking sequence generation methods [10, 11, 13].

These methods utilize a preset distinguishing sequence, that is, a sequence of inputs which is the same for every state of the FSM \( M \). There exists another type of distinguishing sequences, the adaptive distinguishing sequences that are rooted decision trees where each root to leaf path represents an input sequence that is specific to the state represented by the leaf. It has been reported that there may not be a preset or adaptive distinguishing sequence for every FSM and that to determine the existence of a preset distinguishing sequence for a given FSM is PSPACE-complete while that of an adaptive distinguishing sequence is of polynomial complexity [3]. It has been shown that if an FSM has a preset distinguishing sequence, it also has an adaptive one, but the converse is not true (see, for example, [3]). In this paper, we note that often adaptive distinguishing sequences can be used instead of preset ones. We then show that checking sequence construction methods utilizing preset distinguishing sequences can easily be adapted to work with adaptive distinguishing sequences leading to methods that yield shorter checking sequences and more widely useable than when used with preset distinguishing sequences.

The rest of the paper is organized as follows: Section 2 briefly reviews the terminology. Section 3 explains the use of adaptive distinguishing sequences for constructing a checking sequence, and presents the advantages of their use with respect to using preset ones. In Section 4, we review several specific checking sequence construction methods and show that they can be easily adapted to use adaptive distinguishing sequences instead of preset ones. Section 5 concludes the paper.

2. PRELIMINARIES

We represent a deterministic finite state machine (FSM) \( M \) as \((S, X, Y, \delta, \lambda)\), where \( S \) is a finite set of states with \( n = |S| \), \( s_i \in S \) is the initial state, \( X \) is a finite set of inputs with \( p = |X| \), \( Y \) is a finite set of outputs with \( q = |Y| \), \( \delta \) is a state transition function that maps \( S \times X \) to \( S \) and \( \lambda \) is an output function that maps \( S \times X \) to \( Y \). These two functions are extended to input sequences \( I \in X^* \) in the usual manner. A transition from state \( s_i \) to state \( s_j \) in \( M \) is denoted \( \delta(s_i, I) = s_j \).

An FSM \( M \) is considered minimal if, for every pair of states \( s_i, s_j \in S, i \neq j \), there is an input sequence \( I \in X^* \) such that \( \delta(s_i, I) \neq \delta(s_j, I) \). \( M \) is considered completely specified, if for each input \( x \in X \) and each state \( s_i \in S \), \( \delta(s_i, x) \) is defined. \( M \) can be represented by a 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 specified transitions of \( M \). Each edge \( e = (v_i, v_j, x / y) \in E \), is a state transition from state \( s_i \) to state \( s_j \) with input \( x \in X \) and output \( y \in Y \), where \( v_i \) and \( v_j \) are the starting and terminating vertices of \( e \) (states of \( I \)), and input/output (i.e., i/o) pair \( x/y \) is the label of \( e \), denoted by \( label(e) \). \( G = (V, E) \) is strongly connected, if for each pair of vertices \( v_i, v_j \in V \), there exists a path from \( v_i \) to \( v_j \).

A path \( P = (n_1, n_2, x_1/y_1)(n_2, n_3, x_2/y_2)...(n_{r-1}, n_r, x_r/y_r) \), \( r > 1 \), of \( G = (V, E) \) is a finite sequence of adjacent (not necessarily distinct) edges in \( E \), where each node \( n_i, 1 \leq i \leq r \), represents a vertex of \( V \); \( n_1 \) and \( n_r \) are called starting and terminating nodes of \( P \), and the input/output sequence \( (x_1/y_1)(x_2/y_2)...(x_r/y_r) \) is called label of \( P \). \( P \) is represented by \((n_1, n_r, 1/O)\), where \( 1/O \) is the label of \( P \). It is adopted by successive checking sequence generation methods [10, 11, 13].
subpath \((n_i, n_{i+1}; x_i/y_i)\) of \(P\) such that \(n_i\) is recognized as \(s_p\) in \(Q\), \(n_{i+1}\) is recognized as \(s_q\) in \(Q\), \(x_i/y_i = x/y\).

A subpath used to recognize a state is called a state recognition path for that state. The set of all state recognition paths for a given \(M\) and a \(D\) of \(M\) is denoted by \(P_{x}\). A subpath used to verify a transition is called a transition verification path for that transition. The set of all transition verification paths for a given \(M\) and a \(D\) of \(M\) is denoted by \(P_{y}\). Paths used to concatenate recognition/verification paths are called transfer paths and the input portions of their labels are called transfer sequences. For the methods considered in this paper, if this path starts from \(s_i\), recognizes all states of \(M\) and verifies all transitions of \(M\), the input portion of this path’s label is a checking sequence of \(M\).

Several checking sequence generation algorithms are based on the following result [9]:

**Theorem 1.** Let \(Q\) be an IO-sequence of \(M\) starting at \(v_1\) (i.e., the label of a path \(P=(n_1, n_2, Q)\) of \(G\)). If every edge of \(G\) is verified in \(Q\) then \(Q\) is a checking sequence of \(M\).

### 3. CHECKING SEQUENCES

A preset distinguishing sequence is an input sequence that can be used to distinguish each state of the model. An adaptive distinguishing sequence is not really a sequence but a decision tree where each root to leaf path represents an input sequence that is specific to the state represented by the leaf. Formally, an adaptive distinguishing sequence is a rooted tree \(\theta\) with exactly \(n\) leaves; the internal nodes are labeled with input symbols, the edges are labeled with output symbols, and the leaves are uniquely labeled with states of the FSM such that: 1) edges emanating from a common node have distinct output symbols, and 2) for every leaf of \(\theta\), if \(D_i\) and \(y_i\) are the input and output strings respectively formed by the node and edge labels on the path from the root to the leaf labeled by state \(s_i\) of the FSM then \(y_i = \lambda(s_i, D_i)\). We call \(D_i\) as the (adaptive) distinguishing sequence of state \(s_i\). The length of the sequence is the depth of the tree [3].

Adaptive distinguishing sequences have several core advantages over preset ones, while achieving the same purpose of state identification. The first and most obvious one is that a preset distinguishing sequence is also an adaptive one (for which every path from the root to the leaf has the same input portion on its label), while the reverse is not true. In other words, if an FSM has a preset distinguishing sequence, it has an adaptive distinguishing sequence as well, but there are FSMs that have adaptive distinguishing sequences but do not have a preset distinguishing sequence. That simple fact is important because it shows that if we can use adaptive distinguishing sequences instead of preset ones in a checking sequence generation method, then the method can be used on a strictly larger set of FSMs.

The second advantage of adaptive distinguishing sequences over preset ones is deciding their existence. Current algorithms that determine whether a given FSM has a preset distinguishing sequence require exponential time [1]. Moreover, this is probably unavoidable since it is PSPACE-complete to test whether a given FSM has a preset distinguishing sequence [3]. On the other hand, to decide whether a given FSM has an adaptive distinguishing sequence can be achieved in \(O(mn \log n)\) [3]. In other words, deciding whether a checking sequence generation method can be used on a given FSM is at least PSPACE-complete when the method is based on preset distinguishing sequences, while it can take polynomial time\(^1\) when the same method is adapted to use adaptive distinguishing sequences (this does not speak about the complexity of the method itself).

The third advantage is the length of the resulting sequence. It is known that there are FSMs for which the shortest preset distinguishing sequence is of exponential length [3], while an FSM that has an adaptive distinguishing sequence has one of length \(O(n^2)\) [15]. What is more, Lee and Yannakakis have proposed an \(O(kn^2)\) algorithm to produce an adaptive distinguishing sequence of length at most \(n(n-1)/2\) if there is one [3]. Hence using adaptive distinguishing sequences can also provide up to exponential reduction on the length of the checking sequences.

These advantages imply that if it is possible to adapt a checking sequence construction method to use adaptive distinguishing sequences instead of preset ones, then

1. the method will be usable on a strictly larger set of FSMs
2. deciding whether the method can be used on a given FSM will take polynomial time, while it necessarily uses exponential time in the worst case with the preset distinguishing sequences
3. constructing a checking sequence will require polynomial time in the worst case, while it required exponential time in the worst case with preset distinguishing sequences
4. the method will yield a checking sequence that is no longer than the one obtained when using preset distinguishing sequence (since at worst the preset distinguishing sequence can still be used) and can in the best case yield checking sequences that are exponentially shorter than the one produced by the same method used with preset distinguishing sequences.

This shows that if a checking sequence generation method can be altered to use adaptive distinguishing sequences, then it should be adapted and used in this way. In the following, we show that most methods can actually be adapted to use adaptive distinguishing sequences, and thus should be used this way.

### 3.1 Use of Adaptive Distinguishing Sequences

The intuition behind the use of preset distinguishing sequences is that they are “easier” to use because the input sequence is the same regardless of the state the implementation is in. However, this is only useful when there are several possibilities for the current state of the implementation and we want to identify which state it is actually in. When building a checking sequence, it is usually not required: the checking sequence should trigger the expected output from the implementation, that is, the checking sequence assumes that the implementation is at a known state at all times (and checks that the implementation indeed reacts as if it is in that state). Under these conditions, it is easy to see that when building the checking sequence, it is usually possible to replace any portion of the sequence that uses a (preset) distinguishing sequence with the adaptive distinguishing sequence of the state the implementation is supposed to be in.

\(^1\) It will take polynomial time unless some other characteristics of the FSM required by the method cannot be decided in polynomial time.
We can summarize this fact by stating that any checking sequence construction method that is based on distinguishing sequence and that does not use this distinguishing sequence for another purpose than state verification can be adapted to use adaptive distinguishing sequences instead of preset ones, because the checking sequence is built \textit{a priori} and thus must necessarily anticipate the state the implementation should be in. Note that the checking sequence itself is still a preset sequence even if it uses adaptive distinguishing sequences.

### 3.2 A Sufficient Condition

Before discussing checking sequence generation using adaptive distinguishing sequences, it is important to provide an appropriate formal basis. Since several checking sequence generation algorithms are based on Theorem 1 from [9], one possibility is to extend this result to the use of adaptive distinguishing sequences and we do this in this section. First we extend the notion of recognizing nodes to the case where we are using an adaptive distinguishing sequence.

Consider a path $P$ of $G$ representing $M$ and the nodes within it and let $0$ be an adaptive distinguishing sequence for $M$. Let $Q$ = label($P$).

1. A node $n_i$ of $P$ is 0-recognized in $Q$ as state $s_{n_i}$ of $M$ if
   a) $n_i$ is the starting node of a subpath of $P$ whose label is $D_{n_i} / \lambda(s_{n_i}, n_{i+1})$ and we say that $depth(n_i) = 0$; or
   b) $(n_{n_i}, n_{n_i}; T / \delta(s', T))$ and $(n_{n_i}, n_{n_i}; T / \delta(s', T))$ are subpaths of $P$, $n_i$ and $n_i$ are 0-recognized in $Q$ as state $s'$ of $M$, and node $n_i$ is 0-recognized in $Q$ as state $s_{n_i}$ of $M$. Here $depth(n_i) = 1 + \max\{depth(n_i), depth(n_i), depth(n_i)\}$

2. A transition $r(s, s' / x / y)$ is 0-verified in $Q$ if there is a subpath $(n_i, n_{i+1}; s / x) / y$ of $P$ such that $n_i$ is 0-recognized as $s_p$ in $Q$, $n_{i+1}$ is 0-recognized as $s_q$ in $Q$, $x / y = x / y$.

We now adapt results from [9] to prove that it is sufficient to 0-verify transitions in generating a checking sequence.

**Proposition 1.** Let $Q$ be an IO-sequence of $M$ starting at $v_1$ and let $0$ be an adaptive distinguishing sequence of $M$. If every edge of $G$ is 0-verified in $Q$ then for every vertex $v_j$ of $G$ we have that $D_j / \lambda(s_j, D_j)$ is a subsequence of $Q$.

**Proof.** By definition, $Q$ is the label of a path $P = (n_i, n_{i+1}; Q)$ of $G$. Given state $s_j$ of $M$, let $n_j$ denote a node of $P$ with minimum depth that is 0-recognized as $s_j$. Note that there must be at least one such node since $G$ is strongly connected (hence there is at least one transition leaving $s_j$) and every edge of $G$ is 0-verified. If $depth(n_j) > 0$ then there exist subpaths $(n_{n_j}, n_{n_j}, T / \lambda(s', T))$ and $(n_{n_j}, n_{n_j}, T / \lambda(s', T))$ of $P$ such that $n_j$ and $n_j$ are 0-recognized as state $s'$ of $M$, $n_j$ is 0-recognized as $s_j$ and $depth(n_j) > depth(n_j)$, contradicting the minimality of $depth(n_j)$. Thus, $depth(n_j) = 0$ and so in $P$ the node $n_j$ is followed by a subpath with label $D_j / \lambda(s_j, D_j)$ as required.

**Proposition 2.** Let $Q$ be an IO-sequence of $M$ starting at $v_1$ (i.e. the label of a path $P = (n_i, n_{i+1}; Q)$ of $G$) and let $0$ be an adaptive distinguishing sequence of $M$. Let $x_1 \ldots x_{n-1}$ be the input portion of $Q$. Suppose that every edge of $G$ is 0-verified in $Q$ and $Q$ is also an IO-sequence of $M' = (S', X, Y, \delta', \lambda')$ from $\Phi(M)$ starting at $v_1$. If node $n_i$ is 0-recognized in $Q$ as state $s_j$ of $M$ then $\lambda'(\delta'(v_1, x_1 \ldots x_{n-1}), D_j) = \lambda(\delta(v_1, x_1 \ldots x_{n-1}), D_j) = \lambda(s_j, D_j)$.

**Proof.** The proof will proceed by induction on $depth(n_i)$. The base case is $depth(n_i) = 0$. Here, $n_i$ is followed by a subpath with label $\lambda(s_j, D_j)$ and so the result follows. Inductive hypothesis: the result holds for every node of depth less than $l$ and let $depth(n_i) = l$. Since $depth(n_i) > 0$ there exist subpaths $(n_{n_i}, n_{n_i}; T / \lambda(s_{n_i}, T))$ and $(n_{n_i}, n_{n_i}; T / \lambda(s_{n_i}, T))$ of $P$ such that $n_j$ and $n_j$ are 0-recognized as $s_{n_i}$, $n_i$ is 0-recognized as $s_j$ and $depth(n_i)$ is greater than $depth(n_i)$, $depth(n_i)$, and $depth(n_i)$.

By Proposition 1 we know that $M'$ has $n$ states and 0 is an adaptive distinguishing sequence for $M'$. By the inductive hypothesis we know that $\lambda'(\delta'(v_1, x_1 \ldots x_{n-1}), D_n) = \lambda(\delta(v_1, x_1 \ldots x_{n-1}), D_n) = \lambda(s_{n_i}, D_n)$ and $\lambda'(\delta'(v_1, x_1 \ldots x_{n-1}), D_n) = \lambda(s_{n_i}, D_n)$ and so $\delta'(v_1, x_1 \ldots x_{n-1}) = \delta(v_1, x_1 \ldots x_{n-1})$ and $\delta(v_1, x_1 \ldots x_{n-1}) = \delta(v_1, x_1 \ldots x_{n-1})$. Thus, $\delta'(v_1, x_1 \ldots x_{n-1}) = \delta(v_1, x_1 \ldots x_{n-1})$ and $\delta(v_1, x_1 \ldots x_{n-1}) = \delta(v_1, x_1 \ldots x_{n-1})$. The inductive hypothesis, we have that $\lambda'(\delta'(v_1, x_1 \ldots x_{n-1}), D_n) = \lambda(\delta(v_1, x_1 \ldots x_{n-1}), D_n) = \lambda(s_j, D_j)$ and so the result follows.

**Theorem 2.** Let $Q$ be an IO-sequence of $M$ starting at $v_1$ (i.e. the label of a path $P = (n_i, n_{i+1}; Q)$ of $G$) and let $0$ be an adaptive distinguishing sequence of $M$. If every edge of $G$ is 0-verified in $Q$ then $Q$ is a checking sequence of $M$.

**Proof.** Let $M'$ be an element of $\Phi(M)$ with initial state represented by $v_1$ such that $Q$ is an IO-sequence of $M'$ and let $x_1 \ldots x_n$ denote the input portion of $Q$. By Proposition 1 we know that $M'$ has $n$ states $s_1, \ldots, s_n$ such that for all $1 \leq i \leq n$ we have that $\lambda'(s_i, D_j) = \lambda(s_i, D_j)$. It is sufficient to prove that under these conditions we have that for every transition $(s_i, s_j; x/y)$ of $M$ we have that $(s_i, s_j; x/y)$ is a transition of $M'$; from this we can deduce that $M$ and $M'$ are isomorphic. Given transition $(s_i, s_j; x/y)$ of $M$ there is a corresponding edge $(n_i, n_{i+1}; x/y)$ of $P$ such that $n_i$ is 0-recognized as $s_i$ in $Q$ and $n_{i+1}$ is 0-recognized as $s_j$ in $Q$. By Proposition 2 we know that $\lambda'(\delta'(v_1, x_1 \ldots x_{n-1}), D_n) = \lambda(\delta(v_1, x_1 \ldots x_{n-1}), D_n) = \lambda(s_j, D_j)$ and $\lambda'(\delta'(v_1, x_1 \ldots x_{n-1}), D_n) = \lambda(\delta(v_1, x_1 \ldots x_{n-1}), D_n) = \lambda(s_j, D_j)$ and so $(s_i, s_j; x/y)$ is a transition of $M'$ as required.

We have proved a result equivalent to Theorem 1 from [9] that gives a sufficient condition for a test sequence produced using adaptive distinguishing sequence to be a checking sequence. We can thus extend checking sequence generation algorithms that are based on this result to allow adaptive distinguishing sequences and in Section 4 we explain how this can be done.

### 4. THE PROPOSED METHODS

#### 4.1 The HEN64 Method

In the method proposed by Hennie [2], henceforth called HEN64, the states are recognized in a specific order which takes a
permutation of states $s_1$ to $s_j$ such that a preset distinguishing sequence $D$ followed by (a possibly empty) transfer sequence $T$ is applied at $s_i$ and $s_{i+1} = \delta(s_i, DT)$, for all $i$, $1 \leq i \leq n-1$, and for $s_{n+1} = s_1$, $D$ is applied. Thus, one state recognition path is formed as the only element of $P_\delta$. Then, the transition verification path for each $(s_j, s_{j+1} x/y)$ is formed by applying (a possibly empty) transfer sequence $T_1$ from a state $s_j$ to state $s_{j+1}$, and then applying $D$ followed by (a possibly empty) transfer sequence $T_2$ to reach $s_j$ and finally applying $xD$ where $s_j$ is the terminating state of the only path in $P_\delta$ or a transition verification path in $P_T$. This method can be adapted to adaptive distinguishing sequences [3]. When an adaptive distinguishing sequence is used, then the path that will be used to recognize the states will be formed exactly in the same manner. The only difference will be that instead of $DT$, $DT_1$ will be applied at each state $s_j$, where $D_j$ is the adaptive distinguishing sequence of $s_j$. The use of $D$ within the transition verification paths will also be replaced by adaptive distinguishing sequences. More explicitly, the transition verification path for a transition $(s_j, s_{j+1} x/y)$ will be formed by applying (a possibly empty) transfer sequence $T_1$ from a state $s_j$ to state $s_{j+1}$ and then applying $D_{j+1}$ followed by (a possibly empty) transfer sequence $T_2$ to reach $s_j$ and finally applying $xD$, where $s_j$ is the terminating state of the only path in $P_\delta$ or a transition verification path in $P_T$ and $D_j$ is the adaptive distinguishing sequence of the state $s_j = \delta(s_j, x)$.

4.2 The UWZ97 Method

The method proposed by Ural et al. [9], henceforth called UWZ97, first forms so called state recognition path candidates as concatenations of the application of $D$ followed by a $T$ at each state until the application of the last $D$ is a replication of an earlier application of $D$ at the same state in the concatenated path. Some of these $n$ state recognition path candidates can be a suffix of other state recognition path candidates. In fact, a state recognition path candidate which is not suffix of another state recognition path candidate is taken to be a state recognition path (i.e., an element of $P_\delta$), and is called an $\alpha$-sequence in [9]. The number of $\alpha$-sequences is therefore $k \leq n$. Transition verification paths are formed by applying $D$ after the transition’s input. UWZ97 finds a shortest sequence containing all $\alpha$-sequences and transition verification paths connected (possibly) by transfer paths.

When adaptive distinguishing sequences are used, the state recognition path candidates will be formed by using the adaptive distinguishing sequences of the corresponding states instead of the preset distinguishing sequence. In other words, the state recognition path candidate for state $s_j$ will have the label $D_1 T_{i_1} D_2 T_{i_2} D_3 ... D_{i_k}$, where $1 \leq i_j \leq n$, $i_j = i$, $1 \leq j \leq k$, such that $s_{j+1} = \delta(s_j, D_{i_j} T_j)$. If a state recognition path candidate is a suffix of another candidate, then it will be eliminated and the remaining candidates will similarly be called $\alpha$-sequences. The transition verification path of a transition $(s_j, s_{j+1} x/y)$ will have the label $xD T_j$, where $s_j = \delta(s_j, x)$.

4.3 The HIU06 Method

The method proposed by Hierons and Ural [13], henceforth called HIU06, is an enhanced version of the method in [9]. There are three main differences between HIU06 and UWZ97. The first one is that, while forming state recognition path candidates as concatenations of the application of $D$ at each state, HIU06 permits the application of the last $D$ in the concatenation to be a replication of an earlier application of $D$ at the same state not necessarily in the same concatenated path. Therefore, some of the $n$ state recognition path candidates can terminate once the last $D$ in the concatenation is found to be a replication of an earlier application of $D$ at the same state in another concatenated path, yielding a shorter state recognition path as an element of $P_\delta$. Similar to UWZ97, some state recognition path candidates can be a suffix of other state recognition path candidates and hence can be dropped when forming $P_\delta$. The elements of $P_\delta$ formed in this manner are called $\alpha'$-sequences in [13]. The number of $\alpha'$-sequences is $k \leq n$. The second difference relates to the optimization algorithm used and involves a change that allows optimization to occur over a larger set of checking sequences.

Using adaptive sequences does not have any complicating effect on forming $\alpha'$-sequences. Similar to the case of UWZ97, the state recognition path candidate for state $s_j$ will have the label $D_1 T_{i_1} D_2 T_{i_2} D_3 ... D_{i_k}$, where $1 \leq i_j \leq n$, $i_j = i$, $1 \leq j \leq k$, such that $s_{j+1} = \delta(s_j, D_j T_j)$. $\alpha'$-sequences will be obtained by eliminating those candidates that are a suffix of another candidate.

The third difference between UWZ97 and HiU06 is that, although a transition verification path is similarly formed by applying a $D$ after the transition’s input, a state recognition path is allowed to overlap a transition verification path, as long as the overlap is on the entire length of $D$. The method decides whether this overlapping should be used or not while forming the checking sequence. When adaptive distinguishing sequences are used, the transition verification path of a transition $(s_j, s_{j+1} x/y)$ will have the label $xD T_j$, where $s_j = \delta(s_j, x)$. If there is an $\alpha'$-sequence that starts from $s_j$, the occurrence of $D_j$ at the beginning of this $\alpha'$-sequence may or may not be overlapped with the $D_j$ at the end of the transition verification path of $(s_j, s_{j+1} x/y)$. The method will decide this as it tries to optimize the length of the checking sequence produced.

5. CONCLUSIONS

A checking sequence generated from an FSM $M$ is guaranteed to lead to failures if the implementation FSM $N$ has no more states than $M$. Many checking sequence generation methods are based on the use of a preset distinguishing sequence $D$ that distinguishes the states of $M$, despite the negative computational complexity results regarding distinguishing sequences.

This paper has investigated the use of adaptive distinguishing sequences. One of the benefits of using an adaptive distinguishing sequence, rather than a preset distinguishing sequence, is that there are FSMs for which there exists an adaptive distinguishing sequence but no preset distinguishing sequence and the converse is not the case. Further, in contrast to preset distinguishing sequences, there are polynomial time algorithms that decide whether $M$ has an adaptive distinguishing sequence and, if it does, generates such an adaptive distinguishing sequence.

We have argued that when a checking sequence is being produced, adaptive distinguishing sequences can be used in place of preset distinguishing sequences. In addition, recent checking sequence generation algorithms are based on a sufficient condition by Ural et al. [9]. We have proved that the corresponding result holds for adaptive distinguishing sequences. We have also explained how several checking sequence
generation algorithms can be altered so that they use adaptive distinguishing sequences.

Future work will consider other checking sequence generation algorithms. It will also involve implementing these new algorithms, that use adaptive distinguishing sequences, and evaluating them on FSMs.

6. ACKNOWLEDGMENTS
This work has been supported in part by grants from the Natural Sciences and Engineering Research Council of Canada, the Ontario Centers of Excellence, and Sabanci University.

7. REFERENCES