17

On the design for testability of communication protocols

N. Yevtushenko[1], A. Petrenko[2], R. Dssouli[2], K. Karoui[2], S. Prokopenko[1]

1 - Tomsk State University,
36 Lenin str., Tomsk, 634050, RUSSIA.
2 - Université de Montréal,
C.P. 6128, succ. Centre-Ville, Montréal, H3C 3J7, CANADA,
Phone: (514) 343-7535, Fax: (514) 343-5834,
{petrenko, dssouli, karoui}@iro.umontreal.ca

Abstract

Design For Testability (DFT) is understood as the process of introducing some features into a protocol entity that facilitate the testing process of protocol implementations. DFT at the implementation level deals with a particular realization on a given platform, whereas DFT at the specification level affects all possible implementations regardless of the implementation process. The fact that protocols are usually specified only partially, facilitates DFT at the specification level. In this paper, we address one particular problem of DFT, the problem of finding a minimal augmentation of the given protocol behavior (an FSM) such that a newly obtained specification is more testable than the original one, while maintaining sets of defined states and events. We propose an approach to augmenting a partially specified FSM such that a test suite for the resulting FSM with guaranteed fault coverage is shorter than that for the original FSM.

Keywords

Conformance testing, protocol testability, design for testability, partial FSMs, test derivation

1 INTRODUCTION

Behavior of a protocol entity is usually described for a number of situations. A situation in the entity can be understood as a combination of its state and a current input event from a peer entity or from the user of this entity. A service definition and protocol specification explicitly define situations, sometimes called valid, which can happen during normal or abnormal course of communication. If the protocol behavior is unspecified in a valid situation then the protocol is said to have an error, called an unspecified reception. A well-formed protocol is neither under- nor over-specified. An over-specified protocol has unreachable states or transitions which can never be executed. However, even well-formed protocols leave certain situations undefined. These situations are regarded as invalid, in the sense that some events can never happen in a particular state. There are also certain signals from the user which do not require any communication with a remote entity and are processed locally. Local behavior is usually not standardized and can be implemented in various ways. For this reason, protocols are widely

A. Cavalli et al. (eds.), *Protocol Test Systems VIII*

recognized as partially specified systems [BoPe94], [PBD93], [SiLe89]. This feature facilitates the process of implementing a protocol. Undefined situations are utilized for optimizing a protocol implementation on a given platform. Various criteria can be applied for optimization. For complex protocols, testability of their implementations becomes a primary concern of communication software designers.

In general terms, testability of a protocol entity means that it has some features that will facilitate the testing process [DsFo91], [VLC93], [PDK93]. Design for testability (DFT) is understood as the process of introducing these features into the protocol entity. DFT at the implementation level deals with a **particular** realization on a given platform, whereas DFT at the specification level affects **all** possible implementations regardless of the implementation process. Unfortunately, most existing protocols have been designed and documented without testing requirements in mind [VLC93]. To improve the protocol's testability, one should rather rely only on those combinations of states and events for which the protocol behavior is not defined. Assuming that the behavior in such situations can be arbitrary defined, the problem of DFT at the specification level can be formulated as follows.

We must find a minimal augmentation of the given protocol behavior such that a newly obtained specification is more testable than the original one. A measure of testability of a protocol entity is assumed to be inversely proportional to the shortest length of a test suite needed to achieve guaranteed (complete) coverage of certain faults [PDK93]. To the best of our knowledge, formal methods for DFT of protocols have not yet been explored. Note that the existing formal methods and techniques for improving testability which have been developed mainly in the hardware area, see, for example, [ShLe94], [ABF90], [Jose78], cannot be applied in this domain, since they either rely on the structure of an implementation, or they change the set of states, or the sets of input/output events, in a way that is similar to the adding of a read-state message in protocol engineering.

In this paper, we consider a simple FSM model of a protocol machine (at least its control portion) and assume the following scenario of DFT. In the first step, an initial FSM specification is derived from the given requirements. If the FSM is not completely specified then its undefined transitions are "don't care" transitions which model situations where the requirements do not restrict any further protocol behavior. The problem now is how to augment the partially specified FSM by converting "don't care" transitions into defined transitions such that a test suite for the resulting FSM with guaranteed fault coverage is shorter than that for the original FSM. In the last step, the augmented specification of the protocol and its test suite are released for implementation. Even if the scenario seems somewhat idealistic, we believe that such an approach should spare efforts required to produce conforming implementations.

This paper is structured as follows. In Section 2, we present some basic definitions and concepts. In Section 3, we discuss the influence of machine's parameters on the size of a complete test suite and derive several formulae for estimating its length. Based on this discussion, we introduce the basic idea underlying our method for assigning "don't care" transitions presented first for a machine with a single input in Section 4 and then generalized for an arbitrary machine in Section 5. Section 6 contains application examples, including the INRES protocol. We conclude in Section 7 by presenting some open research issues.

2 BASIC NOTIONS AND DEFINITIONS

Throughout this paper we make use of the following definitions. A *partial Finite State Machine* (FSM) A is a 6-tuple $(S, X, Y, \delta, \lambda, D_A)$, where S is a set of n states; X and Y are finite sets of inputs and outputs; δ and λ are transition and output functions; D_A is a set of *defined* transitions of A, that is a subset of $S{\times}X$. We assume that each input labels at least one defined transition. An initialized machine also has a designated initial state s_0. A becomes a *complete* (completely specified) machine if $D_A = S{\times}X$. Transitions in $(S{\times}X)\backslash D_A$ are undefined or *"don't care"* transitions. Here, we assume the so-called "undefined by default" convention for undefined transitions [PBD93], that is, if $(s, x) \in (S \times X) \backslash D_A$ then $\delta(s, x)$ and $\lambda(s, x)$ can be set to (assigned) any s', $s' \in S$ and any y, $y \in Y$, respectively.

A sequence $x_1..x_k$ of the set X^* of all possible input sequences is called an *acceptable* input sequence for state s if there exist k states $s_1,...,s_k$ from S such that $\delta(s, x_1) = s_1$ and $\delta(s_i, x_{i+1}) = s_{i+1}$, $i = 1, ..., k$-1. We use X_i^* to denote a set of all input sequences acceptable for state s_i and X_A^* for the state s_0. Two states s_i and s_j of FSM A are said to be *distinguishable* if there is an input sequence $\alpha \in X_i^* \cap X_j^*$ such that $\lambda(s_i, \alpha) \neq \lambda(s_j, \alpha)$. If any two states of A are distinguishable then A is a *reduced* machine. As usual, states s_i and s_j of A are *equivalent* iff $X_i^* = X_j^*$ and $\lambda(s_i, \alpha) = \lambda(s_j, \alpha)$ for every sequence α.

Let A be a reduced partial FSM. Given a pair of states s_i and s_j, we choose a sequence α_{ij} that distinguishes them, and form a set $W_i = \{\alpha_{ij} \mid s_j \in S \text{ and } j \neq i\}$. The set W_i is called an *identifier* of state s_i. If the W_i has just a single identifying sequence, i.e. $|W_i|=1$ then it is often called a UIO-sequence. If there exists a set W of input sequences such that any sequence from W is acceptable for any state and W is a state identifier of any state then we refer to the set W as a *characterization* set of A. In the case where the W set consists of just a single identifying sequence, we refer to this sequence as a *distinguishing* sequence [Henn64}. Machines with such sequences usually possess very short tests.

We say that the FSM A is *connected* if for any state s there is an input sequence β such that $\delta(s_0, \beta) = s$. A set $V = \{\varepsilon, \beta_1,...,\beta_{n-1}\}$ of input sequences is said to be a *state cover* if for any state s_i of S there is a sequence $\beta_i \in V$ which takes FSM A from its initial state into state s_i, where ε is the empty sequence: $\delta(s_0, \varepsilon) = s_0$, thus $V \ni \varepsilon$.

Let a connected reduced FSM A have a characterization set W. If $D_A = S \times X$ then W always exists. Assume the sets V and W have the following properties:

1) if a sequence β belongs to V then any prefix of it also belongs to V, i.e.

if $\beta x \in V$ then $\beta \in V$; (2.1)

2) if a sequence α belongs to W then any suffix of it also belongs to W, i.e.

if $x\alpha \in W$ then $\alpha \in W$. (2.2)

As an example, a homogeneous distinguishing sequence satisfies (2.2), a sequence is *homogeneous* if it is a sequence of the same symbol, i.e. $x^r = xx^{r-1}$, $r \geq 1$.

It is known that if V and W have the properties (2.1) and (2.2) then the set $TS = TC@W$ is a *complete* test suite for the FSM A in the class $\Im_n$ of all FSMs with up to n states [Vasi73]. Here "@" stands for concatenation operation on sets, and the set TC is a *transition cover* that contains V as well as any sequence β_i from V concatenated with any input that is acceptable for state $s_i = \delta(s_0, \beta_i)$. [Vasi73] gives this version of the so-called W-method [Chow78].

A complete test suite can be shortened if each sequence of the set W is applied after any sequence from V. However, after any sequence from TC that is not in V, only a part of the set W is applied, namely, a corresponding state identifier. This is the main idea of the Wp-method [FBK91]. We use the following notation for such test suites: $TS = V@W \cup TC \otimes W$. Here we consider slightly generalized versions of these methods to cover partial reduced machines. For more discussions on partial FSMs the reader is referred to [Petr91], [PBD93], [BoPe94].

3 LENGTH OF A TEST SUITE AND "DON'T CARE" TRANSITIONS

3.1 Estimating a test size

Generating a complete test suite for a given FSM involves many choices, most of which are left without any guidance in most existing test derivation methods. This makes it extremely difficult to determine length of a complete test suite until the test suite is actually derived. Its length depends on many factors [BPY94]. Among them, properties of the state cover, transition cover and characterization set chosen for the test derivation seem most essential. We note that the known bounds [Vasi73] are derived for the case where state covers and characterization sets satisfy the properties (2.1) and (2.2). These properties facilitate the test length estimation as well. Here we look for other properties of sets V and W that provide shorter test suites. We first consider an example.

Example. The FSM A shown in Figure 3.1 possesses several state covers with the property (2.1). We choose two of them of different lengths: $V_1 = \{\varepsilon, a, aa, b\}$ and $V_2 = \{\varepsilon, a, aba, ab\}$. V_1's length is four, whereas V_2 contains six symbols.

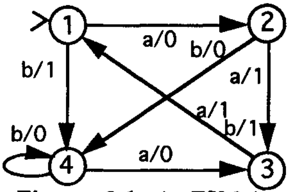


Figure 3.1 An FSM A.

The machine also possesses different characterization sets with the property (2.2): $W = \{aaa\}$, $W' = \{a, b\}$. In the first case, there is a homogeneous distinguishing sequence aaa, from which it is easy to obtain state identifiers as follows: $W_1 = \{aaa\}$, $W_2 = \{aa\}$, $W_3 = \{aa\}$, $W_4 = \{aaa\}$. In the second case, $W'_1 = W'_2 = W'_3 = W'_4 = \{a, b\}$. Total length $L(W)$ of W is three; $L(W') = 2$, but it contains two sequences. Based on these sets, it is possible to construct various test suites complete in the class $\mathfrak{I}_n$, where n=4. The Wp-method gives the following results:

1) $V_1 = \{\varepsilon, a, aa, b\}$, $W = \{aaa\}$. $TS1 = \{aaaaaa, aabaaa, abaaa, baaa, bbaaa\}$. $L(TS1)$= 26.

2) $V_1 = \{\varepsilon, a, aa, b\}$, $W' = \{a, b\}$.
$TS2 = \{aaaa, aaab, aaba, aabb, aba, abb, baa, bab, bba, bbb\}$. $L(TS2) = 34$.

3) $V_2 = \{\varepsilon, a, aba, ab\}$, $W=\{aaa\}$. $TS3 = \{aaaa, abaaaaa, ababaaa, abbaaa, baaa\}$. $L(TS3) = 28$.

4) $V_2 = \{\varepsilon, a, aba, ab\}$, $W' = \{a, b\}$.
$TS4 = \{aaa, aab, abaaa, abaab, ababa, ababb, abba, abbb, ba, bb\}$. $L(TS4) = 38$. □

As can be seen from this example, a characterization set with fewer sequences yields a shorter test suite even if its total length is not minimal. The number of transfer sequences in a state cover cannot be reduced, however, a state cover of a shorter length usually leads to a shorter test suite. We now present some estimations that confirm these observations in the general case.

We consider a test suite $TS = TC@W$ of an FSM A which has a characterization set W assuming that the sets V and W satisfy properties (2.1) and (2.2), respectively. We have seen from the above example that, all things being equal, length of this test suite is determined by lengths of sets V and W and by the cardinality of W. If A is a partial machine then the size of a test suite depends also on the number of defined transitions. Let p be the number of defined transitions of A, i.e. $p \le mn$, where m is a number of inputs.

Proposition 3.1. Given an FSM A with n states, a state cover V with the property (2.1), a transition cover TC of p defined transitions, and a characterization set W with the property (2.2), the total length $L(TS)$ of a complete test suite $TS = TC@W$ does not exceed

$$L(TC)|W| + L(W)(p-n+1). \qquad (3.1)$$

Proof. In fact, the total length of a set $TC@W$ does not exceed the value of

$$\sum_{v_i \in TC, w_j \in W} [L(v_i) + L(w_j)].$$

Then $L(TC@W) \leq [L(v_1) + L(w_1)] + ... + [L(v_r) + L(w_t)] = [L(v_1) + ... + L(v_r)]\ |W| + L(W)r$, where r is a number of sequences in TC. Thus
$L(TC@W) \leq L(TC)|W| + L(W)|TC|$.

Now it is sufficient to show that the number of sequences in TC is $(p - n + 1)$. Consider the set of input sequences TC and a corresponding successors tree of A with its initial state as a root of this tree. Due to (2.1), the tree has exactly p edges from its internal nodes, therefore this tree has p nodes excluding the root of this tree. Since n nodes are internal, there are exactly $(p - n + 1)$ terminal nodes, i.e. $|TC| = p - n + 1$. ❑

Consider again the FSM A shown in Figure 3.1. In the case of a completely specified FSM, $p = mn$. Assume that V_1 and W are chosen for test derivation. For the test suite $TS1$ we have $L(TC) = 12$, $p\text{-}n\text{+}1 = 5$, $|W| = 1$, $L(W) = 3$, and therefore the length is 27. It is close to the actual length of 26. In the case where V_2 and W' are chosen, we have $L(TC) = 14$, $p\text{-}n\text{+}1 = 5$, $|W'| = 2$, $L(W) = 2$, and $14 \cdot 2 + 5 \cdot 2 = 38$. This is length of the test suite $TS4$.

Since $(p\text{-}n\text{+}1)$ is the number of sequences in the set TC, we usually have $L(TC) >> (p\text{-}n\text{+}1)$. For this reason, **length of a complete test suite primarily depends on the number of sequences in the set W, rather than on its total length.**

It is also worth noting than based on (3.1), it is possible to derive the least upper bound on the length of a complete test suite. Assume for simplicity that the machine is completely specified. Then it is known that the bound is $O(mn^3)$, where m is the number of inputs [Vasi73], [LeYa94]. We need a more precise estimation. [TyBa75] gives $L(W) \leq n(n\text{-}1)/2$, $|W| \leq n\text{-}1$, $L(V) \leq n(n\text{-}1)/2$ provided that the properties (2.1) and (2.2) hold. It is not difficult to check that $L(TC) \leq n(m\text{-}1) + n(n\text{-}1)/2 + 1$. $(p\text{-}n\text{+}1) = nm - n + 1$. Then
$L(TS) \leq [n^2m + 2nm - 2n + 2](n\text{-}1)/2 = L_{\max}$.

Thus, $L(TS) \leq L_{\max} < mn^3$ for completely specified reduced machines with $n \geq 2$ states and $m \geq 2$ inputs. The least upper bound on tests for partially specified machines remains unknown, as they are still the subject of active research [BPY94], [BoPe94].

Next, we estimate length of a test suite for a partial FSM that possesses homogeneous identifying sequences.

Proposition 3.2. Given an FSM A with n states, a state cover V with the property (2.1), a transition cover TC of p defined transitions, and a characterization set W with k homogeneous identifying sequences of length up to h, there exists a complete test suite of length not more than
$[L(V) + hn]k + L(TC) + h(p\text{-}n\text{+}1)$. (3.2)

Proof. Based on the Wp-method, we construct a complete test suite of the form $TS = V@W \cup TC \otimes W$. The set $V@W$ means that every sequence from V is concatenated by k sequences of length up to h. In the worst case, this part of the test suite is no longer than $L(V)k + hnk$. The set $TC \otimes W$ means that just a single identifying sequence of length up to h is applied to the state reached after any sequence from the transition cover TC. Similar to the previous proof, $|TC| = p\text{-}n\text{+}1$, and the total length of sequences of the set $TC \otimes W$ does not exceed the value $\sum_{v_i \in TC} [L(v_i) + h] = \sum_{v_i \in TC} L(v_i) + h|TC| = L(TC) + h(p\text{-}n\text{+}1)$. ❑

Corollary 3.3. The length of a complete test suite of an FSM A with n states and a transition cover TC of p defined transitions is no less than
$L(TC) + (p\text{-}n\text{+}1) = L_{\min}$. (3.3)

(3.1)-(3.3) can be used to determine the expected size of a test suite which is complete for the given machine in the class of all machines with an equal or fewer number of states.

3.2 A criterion for assigning "don't care" transitions

Practice has not yet provided us with protocol machines such that the length of complete test suites approaches the least upper bound $L_{\max}$. However, there exists a class of FSMs for which the length of a complete test suite meets the lower bound $L_{\min}$. These are machines with a distinguishing sequence of length one. Unfortunately, protocol machines seldom fall into this class. It is known that certain machines have neither distinguishing nor state identifying seqences, and there are machines whose states have identifying (UIO) sequences, but only of exponential length [LeYa94]. Their least upper bound remains unknown, especially for partial FSMs. In the general case, n states may require identifiers consisting of up to n-1 sequences, and it is possible to construct a sequence of length up to $n(n-1)/2$ distinguishing two states in a given partial reduced FSM. Thus, in most cases, we deal with partial FSMs which are not easily testable in the sense that they require a complete test suite of length far from the best possible.

If a protocol machine is specified completely then some measures can be taken to improve the protocol testability at the implementation level only. However, if it is specified only partially then its testability can be improved at the specification level ensuring that all implementations derived from the augmented specification are more testable than that derived from the original specification. Given a partial FSM with n states, t inputs, m outputs and p defined transitions, there are $(nt\text{-}p)$ "don't care" transitions. Each undefined transition can be either left undefined or transformed into a defined one in nm different ways. Thus, there exist $(nm+1)^{(nt-p)}$ FSMs which are quasi-equivalent to the given machine [Gill62]. We call them *augmented machines* with respect to the original machine. An exhaustive procedure would enumerate all $(nm+1)^{(nt-p)}$ machines, derive a complete test suite for each of them and search for the one with the shortest test suite. We wish to avoid such a brute force search, called also *"perebor"*, and should find a criterion to guide the process of assigning "don't care" transitions.

As follows from the discussion of Section 3.1, a homogeneous distinguishing sequence ensures $|W|=1$ and leads to a short test suite. Since such a sequence may not always exist, the next best case is when each state possesses a homogeneous identifying sequence which might be common for several states. Thus, the transformation of "don't care" transitions that maximizes the number of states possessing homogeneous identifying sequences can be regarded as a successful transformation. The approach we take is based on these heuristics.

However, the numerical characteristics of homogeneous identifying sequences alone such as their number and total length are not sufficient to choose the best assignment of "don't care" transitions. The problem is that their assignment affects length of a test suite in two opposite ways. On the one hand, if newly added transitions create short identifying or even diagnostic sequences of the machine which had no such sequences prior to the assignment then the size of the test suite most probably will be reduced. On the other hand, the number of defined transitions increases, thus, additional test sequences are required to test new transitions. If the increase in length caused by an assignment exceeds the savings gained then the assignment would deteriorate the testability of the given machine. For this reason, the expected or actual length of a complete test suite should eventually be used to estimate the effect of assignments.

We demonstrate later in this paper that if a state has at least one "don't care" transition then it is always possible to construct an augmented machine such that this state has a homogeneous identifying sequence. Moreover, if the necessary and sufficient conditions established below are satisfied then there exists an augmented machine with a homogeneous distinguishing sequence. In the worst case, the length of identifying sequences reaches n, the number of states.

4 AN FSM WITH A SINGLE INPUT

In this section, we assume that a given machine has just one input and propose a method for finding assignments of all "don't care" transitions such that their initial states possess identifying sequences in an augmented machine. We also show that under certain conditions the augmented machine has a distinguishing sequence. By the construction, the obtained sequences are also homogeneous for the original machine which contains the machine with a single input as its submachine.

4.1. Auxiliary notions

Let B be a reduced partial FSM. Consider its submachine A which is obtained by restricting its input set X to an arbitrary input $x \in X$ which labels at least one "don't care" transition. Thus, $A = (S, \{x\}, Y, \delta, \lambda, D_A)$ is an FSM with a single input x.

The state transition graph of the FSM A has a *cycle* (s_1->s_2->...->s_k) if $\delta(s_i, x) = s_{i+1}$ for $i = 1,\dots,k-1$, $k \geq 1$; and $\delta(s_k, x) = s_1$. For each state of the cycle, any input sequence is acceptable. If length of an input sequence ω is a multiple of k, i.e. $\omega = x^{mk}$, $m \geq 1$, then

$$\delta(s_i, x^{mk}) = s_i \text{ for all } i = 1,\dots,k. \tag{4.1}$$

Given a sequence x^{mk-1} of length mk-1, we also have

$$\delta(s_i, x^{mk-1}) = s_{i-1} \text{ for all } i = 2,\dots,k \text{ and } \delta(s_1, x^{mk-1}) = s_k. \tag{4.2}$$

State s is said to be a *starting* state of A if there is no transition leading to this state. Consider an arbitrary path s_1->s_2->...->s_k from a starting state s_1. We say that the path *terminates in state* s_k (a terminating path) if (s_k, x) is a "don't care" transition; or the path *cycles* (a cycling path) if $\delta(s_k, x) = s_{i+1}$ for some $k > i \geq 1$, i.e. s_{i+1}->...->s_k is a cycle. The behavior of the FSM A is defined in every state of a cycling path, thereby such path does not traverse any state with a "don't care" transition.

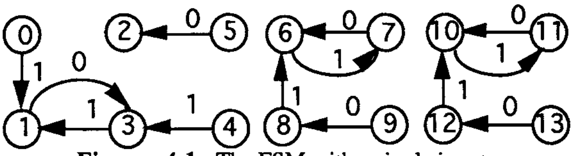


Figure 4.1 The FSM with a single input.

Example. Consider an FSM shown in Figure 4.1. Its transitions are labeled with outputs only, since the machine has just one input. Starting states of A are 0, 4, 5, 9, 13. State 2 has a "don't care" transition. There is only one terminating path from the starting state 5 which terminates in state 2. Four other paths are cycling. All states except 2 and 5 accept all input sequences. The FSM has the following sets of equivalent states: {1, 7, 11}, {0, 3, 6, 10}, {4, 8, 12}, {9, 13}. Notice that starting states 9 and 13 are equivalent, but each of them is not equivalent to any other state of its path. Moreover, they are not equivalent to any other state of this machine. ❑

Based on this observation, we claim a more general property of an FSM with a single input.

Proposition 4.1. Let the FSM A have a cycling path $P = (s_1$->...->$s_k)$ from a starting state s_1 that is not equivalent to any other state of this path. Then A has a cycling path $P' = (s'_1$->...->$s'_k)$ from a starting state s'_1 that is not equivalent to any other state of A which is not a starting state.

Proof. If the starting state s_1 of P is not equivalent to any other state of A then P itself is the path of the proposition. Suppose therefore that there is another state s'_j equivalent to s_1 and s'_j is an intermediate state of a cycle (s'_1->...->s'_t) or of a cycling path (s'_1->...s'_j->...->s'_t), where $j>1$. For any input sequence, the successors of the equivalent states are also equivalent, therefore states $\delta(s_1, x^t)$ and $\delta(s'_j, x^t)$ are equivalent.

If s'_j is an intermediate state of a cycle (s'_1->...->s'_t) then, because of (4.1), $\delta(s'_j, x^t) = s'_j$. In this case, state s_1 and state $\delta(s_1, x^t)$ of the same path are also equivalent. This contradict our assumption that the state s_1 that is not equivalent to any other state of this path.

Assume then s'_j is an intermediate state of a cycling path (s'_1->...s'_j->...->s'_t), where $j>1$. Now, we must show that the starting state s'_1 is not equivalent to any state of $P = (s_1$->...->$s_k)$. In fact, if states s'_1 and s_p, $p \geq 1$ are equivalent, so are states $\delta(s'_1, x^{j-1})$ and $\delta(s_p, x^{j-1})$. $\delta(s'_1, x^{j-1}) = s'_j$ and is equivalent to s_1. Now, states $\delta(s_p, x^{j-1})$ and s_1 are required to be equivalent as well. This is the contradiction. We can exclude the path (s_1->...->s_k) from our consideration. As a result, we could only find another cycling path whose starting state is equivalent to the starting state s_1 of the given path P. □

A cycling path whose starting state is not equivalent to any other starting state of cycling paths is termed a ***dominant*** cycling path of the FSM A. In the above example, among four cycling paths, there are two dominant cycling paths: (9,8,6,7) and (13,12,10,11). As will be shown in next section, dominant cycling paths play an important role for the assignment of "don't care" transitions.

4.2 A single transition

We consider in this section the case where the FSM A with a single input has only one "don't care" transition (s, x) and show that it is always possible to assign this transition such that state s becomes distinguishable from any other state in the newly obtained completely specified FSM.

Given an FSM $A = (S, \{x\}, Y, \delta, \lambda, D_A)$, where $(SxX)\backslash D_A = \{(s, x)\}$, we consider its state transition graph and determine all its cycles, terminating paths, cycling paths, and dominant cycling paths. Note that a terminating path can only end in state s, but it might be empty if s has no incoming transition in A. There are four possible cases each of which requires a distinct assignment of the "don't care" transition (s, x):

1) There is no cycle in A, so all paths terminate in state s.
2) In A, there are only cycles and paths terminating in state s.
3) A has also cycling paths, but it has no dominant cycling paths.
4) A has a dominant cycling path.

Next we consider how the "don't care" transition (s, x) should be assigned in each of these cases in order to obtain a homogeneous identifying sequence of state s.

Case 1

(s, x) is a "don't care" transition. All paths terminate in s. A has no cycle.
We determine the longest path s_1->s_2->...->s_k->s and define a transition in state s on input x $\delta(s, x) = s_1$ and $\lambda(s, x) = y$, where output y is such that the sequence $\lambda(s_1, x), ..., \lambda(s_k, x)y$ cannot be represented as any of its proper prefixes repeated several times. If $|Y|>1$ then it is always possible to find such output. Assigning the transition (s, x), we obtain a completely specified FSM A'.

Proposition 4.2. In case 1, state s is distinguishable from any other state of A'.

Proof. In fact, due to the chosen output assignment, state s cannot be equivalent to any other state in the obtained cycle (s_1->s_2->...->s_k->s). Assume therefore, that s is equivalent to state s'_j, $1 \le j \le t$, which belongs to another path s'_1->...->s'_j...->s'_t->s->s_1->...->s_k. If states s and s'_j are equivalent then states $\delta(s'_j, x^{t-j+1}) = s$ and $\delta(s, x^{t-j+1})$ are also equivalent. State $\delta(s, x^{t-j+1})$ belongs to the cycle s->s_1->...->s_k and it is not state s because $t \le k$ and $j \ge 1$. Then state s should be equivalent to another state of the cycle, but this is impossible. □

Example 1. Consider the FSM shown in Figure 4.2.

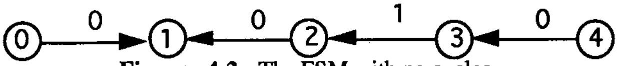


Figure 4.2 The FSM with no cycles.

This machine has a single "don't care" transition in state s_1. There are two paths: s_0->s_1 and s_4->s_3->s_2->s_1 which terminate in state s_1. We choose the longest one and assign $\delta(s_1, x) = s_4$. The output set has two symbols, 0 and 1. If $\lambda(s_1, x)$ is assigned 1 then $\lambda(s_4, x)\lambda(s_3, x)\lambda(s_2, x)\lambda(s_1, x) = 0101$ and it can be represented as its proper prefix 01 repeated two times, i.e. 0101 = (01)(01). In this case, state s_1 would become equivalent to s_3, and so would s_2 and s_4. To distinguish state s_1 from other states we must define $\lambda(s_1, x) = 0$. We obtain a new transition:

s_1-x/0->s_4. Now sequence x distinguishes state s_1 from s_3; sequence xx distinguishes state s_1 from s_4; and xxx distinguishes s_1 from s_0 and s_2. Thus, the sequence xxx is a homogeneous identifying sequence of state s_1 in the augmented FSM. ❑

Case 2

(s, x) is a "don't care" transition. A has cycles, but its paths terminate in s. In this case, we take an arbitrary cycle (s_1->...->s_k) and define a transition from state s to the state s_1, i.e. $\delta(s, x) = s_1$, with the output $\lambda(s, x) = y$ such that $y \neq \lambda(s_k, x)$. By assigning the transition (s, x), we again obtain a new completely specified FSM A'.

Proposition 4.3. In case 2, state s is distinguishable from any other state of A'.

Proof. The obtained FSM A' has some cycles and cycling paths with a cycle (s_1->...->s_k) only. Let s'_1->...->s'_t->s->s_1->...->s_k be such a path. If state s is equivalent to a state in the path then it is also equivalent to another state in the cycle (s_1->...->s_k) since the successors of equivalent states are equivalent for any input sequence. Therefore, we may assume that state s is equivalent to a state s'_1 in a cycle (s'_1->...->s'_t). Consider now an input sequence x^{tk} of length tk. State $\delta(s, x^{tk})$ and state $\delta(s'_1, x^{tk})$ are equivalent states, as they are successors of s and s'_1. By virtue of (4.2) and (4.1), $\delta(s, x^{tk}) = \delta(s_1, x^{tk-1}) = s_k$ and $\delta(s'_1, x^{tk}) = s'_1$. This means s and s_k should be equivalent states, but this is not possible, because $\lambda(s, x) \neq \lambda(s_k, x)$. ❑

Example 2. Consider the FSM with two cycles shown in Figure 4.3.

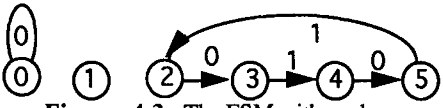


Figure 4.3 The FSM with cycles.

Based on the cycle (s_2, s_3, s_4, s_5), we define $\delta(s_1, x) = s_2$ and $\lambda(s_1, x) = 0$, since $\lambda(s_5, x) = 1$. We obtain a new transition: s_1-x/1->s_2. The machine augmented with this transition has a sequence xxx that distinguishes state 1 from any other state. In fact, $\lambda(s_0, xxx) = 000$, $\lambda(s_1, xxx) = 001$, $\lambda(s_2, xx) = 010$, $\lambda(s_3, xxx) = 101$, $\lambda(s_4, xxx) = 010$, $\lambda(s_5, xxx) = 101$. Alternatively, we could assign $\delta(s_1, x) = s_0$ and $\lambda(s_1, x) = 1$, since $\lambda(s_0, x) = 0$. In either case, the sequence xxx can be used as a homogeneous identifying sequence of s_1 in the completely specified machine. The original machine had no identifying sequence for this state. ❑

Case 3

A has cycling paths, but none of them is dominant. In this case, the starting state of each cycling path is equivalent to another state of this path. To assign the "don't care" transition (s, x) we choose an arbitrary path (s_1->...->s_{i+1}->...->s_k) which ends with the cycle (s_{i+1}->...->s_k), where $i \geq 1$. Since state s_1 is equivalent to a certain state of the path, it is also equivalent to a state s_j of this cycle, $i+1 \leq j \leq k$. Then we assign $\delta(s, x) = s_j$ with the output $\lambda(s, x) = y$ such that $y \neq \lambda(s_{j-1}, x)$. If j=1 then $y \neq \lambda(s_k, x)$. As a result, the augmented machine A' is obtained.

Proposition 4.4. In case 3, state s is distinguishable from any other state of A'.

Proof. We have $\delta(s, x) = s_j$ and $\lambda(s, x) \neq \lambda(s_{j-1}, x)$. Let state s be equivalent in A' to a state s'_p. State s'_p belongs either to a cycle (s'_1->...->s'_t), where $p \leq t$, or to a cycling path s'_1->...->s'_t with a cycle (s'_{r+1}->...->s'_t), where $1 \leq r \leq t$.

In the first case, the equivalence of s and s'_p implies the equivalence of s_j and s'_{p+1} (or s_j and s'_1 if $p=t$). Moreover, states $\delta(s_j, x^{tk-1})$ and $\delta(s'_{p+1}, x^{tk-1})$ are also equivalent. Because of (4.2), $\delta(s_j, x^{tk-1}) = s_{j-1}$, $\delta(s'_{p+1}, x^{tk-1}) = s'_p$. Thus, if states s_{j-1} and s'_p are equivalent, then states s and s_j should be equivalent as well, but this is impossible, because $\lambda(s, x) \neq \lambda(s_{j-1}, x)$. In the second case, state s is equivalent to state s'_p from the path s'_1->...->s'_t which terminates in cycle (s'_{r+1}->...->s'_t), where $1 \leq r \leq t$. If this path is defined in A then, by the assumption, its starting state is equivalent to another state of the path, and we have the situation considered above. Assume finally that state s is equivalent to state s'_p from the path (s'_1->...->s'_r->...->s->s_1->...->s_k) obtained in A'. Again, state s becomes equivalent to a state of a cycle, and we have exactly the same situation as above. ❑

Example 3. Consider the FSM shown in Figure 4.4.

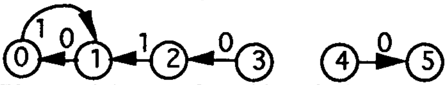


Figure 4.4 The FSM with equivalent states.

There is a cycling path. Its starting state s_3 is equivalent to state s_1. We add a transition from state s_5 to state s_1 labeled with output $\lambda(s_5, x) = 0 \neq \lambda(s_2, x)$. Then state s_5 becomes distinguishable from any other state by the sequence xxx. ❑

Case 4

A has a dominant cycling path. In this case, there exists a cycling path P = (s_1->...s_{i+1}->...->s_k), $i \geq 1$ such that s_1 is not equivalent to any other state of P, moreover, in accordance with Proposition 4.1, only another starting state of a cycling path might be equivalent to s_1.

We transform the "don't care" transition (s, x) into a defined one in the following way. $\delta(s, x) = s_1$ and $\lambda(s, x)$ is assigned any $y \in Y$. Similarly to the cases considered above, we claim that state s is now distinguishable from any other state of the augmented FSM A'.

Proposition 4.5. In case 4, state s is distinguishable from any other state of A'.

Proof. Assume s is equivalent to a state s'_j. If s'_j is involved in a path (s'_1->...->s'_t) in A then s_1 of P can be equivalent to a starting state of another dominant cycling path, because P is also a dominant path. State s cannot be equivalent to any state of such a path. In this case, s can only be equivalent to some state of a cycling path (s'_1->...->s->s_1->...s_{i+1}->...->s_k). However, s_1 becomes equivalent to some state of the cycle (s_{i+1}->...->s_k). This contradicts our assumption that P is a dominant cycling path. ❑

Example 4. Consider the FSM shown in Figure 4.1. This machine has two dominant cycling paths with states (9, 8, 6, 7) and (13, 12, 10, 11). We can choose the first path and define transition from state 2 to state 9. Regardless of the output of this transition, the sequence xxx becomes an identifying sequence of state 2. ❑

Propositions 4.2 - 4.5 implies the following theorem.

Theorem 4.6. Given an FSM $A = (S, \{x\}, Y, \delta, \lambda, D_A)$, where $(S \times X) \backslash D_A = \{(s, x)\}$, it is always possible to assign its "don't care" transition such that state s of the augmented completely specified FSM is distinguishable from any other state and has a homogeneous state identifying sequence of length not exceeding the number of states. ❑

In certain cases, the augmented FSM has a homogeneous distinguishing sequence, as the following theorem shows. States s_i and s_j are said to be *converging* iff $\delta(s_i, x) = \delta(s_j, x)$ and $\lambda(s_i, x) = \lambda(s_j, x)$.

Theorem 4.7. Given an FSM $A = (S, \{x\}, Y, \delta, \lambda, D_A)$, where $(S \times X) \backslash D_A = \{(s, x)\}$, it is always possible to assign its "don't care" transition such that the augmented completely specified FSM has a homogeneous distinguishing sequence of length not exceeding the number of states iff A has neither converging nor equivalent states.

Proof. If A has at least two converging states then regardless of the transition's assignment, these states would become equivalent in any augmented machine. Assume now that we have assigned the transition (s, x) and state s becomes distinguishable from any other state. Theorem

4.6 states that it is always possible. If two states s_i and s_j are such that any input sequence is acceptable for each of them then they are nonequivalent in A as well as in an augmented FSM. Suppose therefore that for one of these states, say, for state s_i, not all sequences are acceptable. In this case, there is an acceptable sequence ω for s_i of A such that $\delta(s_i, \omega) = s$, since (s,x) is the only "don't care" transition in A. Now state $\delta(s_j, \omega)$ should be equivalent to state s. Then $\delta(s_j, \omega) = s$. The latter is possible only if A has converging states.

The completed FSM has only cycling paths, and the cycles are no longer than n. By construction, every two states are distinguishable by a sequence of length k, where k is the length of a cycle. Thus, $k \leq n$. ❑

We have examined all configurations possible in a given FSM with a single "don't care" transition in state s and thus, we have devised a technique for converting such a partial FSM into a completely specified FSM where state s possesses an identifying sequence. Under certain conditions, the resulting identifying sequence may also be a distinguishing sequence. Next, this technique will be generalized to cover the case where there exist several "don't care" transitions.

4.3 Several transitions

Given an FSM $A = (S, \{x\}, Y, \delta, \lambda, D_A)$, where $|(S \times X) \setminus D_A| \geq 1$, let the subset S_u contain all states with "don't care" transitions, i.e. $S_u = \{s \mid (s, x) \notin D_A\}$. We also define a subset S_d of states which accept all possible input sequences $\{x\}^*$; these states are involved in cycles or in cycling paths. The set S_d might be empty. For a state $s_i \in S_u$, let S_i denote the set of states from which state s_i is reachable, $s_i \in S_i$. Clearly, $S_i \cap S_j = \emptyset$ for all $i \neq j$ and $S_i \cap S_d = \emptyset$, since A is a deterministic machine. Based on the set $S_d \cup S_i$, we construct a submachine $A_i = (S_d \cup S_i, \{x\}, Y, \delta_i, \lambda_i, D_i)$ of the FSM A by deleting from A all states $S \setminus (S_d \cup S_i)$ along with their transitions. A_i has exactly one "don't care" transition. The technique of Section 4.2 can now be applied.

We present an algorithm for augmenting a given FSM with a single input in order to obtain an identifying sequence.

Algorithm 1.

Input: A partial FSM $A = (S, \{x\}, Y, \delta, \lambda, D_A)$ with a single input and $|S_u| \geq 1$ "don't care" transitions.

Output: An augmented completely specified FSM $A' = (S, \{x\}, Y, \delta', \lambda')$. Each state of S_u has an identifying sequence.

Step 1. Construct the subset S_d for A.

Step 2. Choose a state $s_i \in S_u$ with the maximal $|S_i|$

Construct a submachine $A_i = (S_d \cup S_i, \{x\}, Y, \delta_i, \lambda_i, D_i)$.

Step 3. Call the technique of Section 4.2 to assign (s_i, x) in A_i (and therefore in A).

Step 4. $S_d := S_d \cup S_i$

$S_u := S_u \setminus \{s_i\}$

If $S_u \neq \emptyset$ then GO TO Step 2. ❑

The resulting machine can be characterized by the following two theorems which are generalized from Theorems 4.6 and 4.7 and are proven in a similar manner.

Theorem 4.8. Suppose that $A = (S, \{x\}, Y, \delta, \lambda, D_A)$, where $|(S \times X) \setminus D_A| \geq 1$ is a given FSM and an FSM A' is the output of Algorithm 1. Then every initial state of "don't care" transitions is distinguishable from any other state in A' and has a homogeneous identifying sequence of length not exceeding the number of states. ❑

Theorem 4.9. Suppose that $A = (S, \{x\}, Y, \delta, \lambda, D_A)$, where $|(S{\times}X)\backslash D_A|\geq 1$ is a given FSM and an FSM A' is the output of Algorithm 1. Then the augmented completely specified FSM A' has a homogeneous distinguishing sequence of length not exceeding the number of states iff A has neither converging nor equivalent states. □

Example. Consider the FSM A shown in Figure 4.5a. The necessary and sufficient conditions of Theorem 4.9 are satisfied, since the machine has no converging or equivalent states, so Algorithm 1 should augment it in such a way that the resulting machine has a homogeneous distinguishing sequence of length not exceeding the number of states.

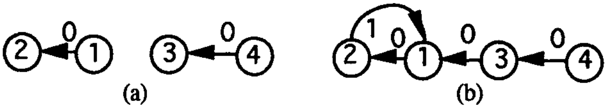


Figure 4.5 The FSM and its augmented FSM.

States 2 and 3 have "don't care" transitions, so the set $S_u = \{2, 3\}$. No state of A accepts all input sequences, the set $S_d = \emptyset$. We choose state 2 and find the set $S_2 = \{1, 2\}$. A submachine contains the transition 1->2. It is the case 1 of the technique from Section 4.2. We assign $\delta(2, x) = 1$ and $\lambda(2, x) = 1$. Now $S_d = \{1, 2\}$, $S_u = \{3\}$. $S_3 = \{3, 4\}$. We have the case 2 of Section 4.2. $\delta(3, x) = 1$ and $\lambda(3, x) = 0$. The augmented machine is shown in Figure 4.5b. It has a homogeneous distinguishing sequence xxx. The identifying sequence for state 1 is xx, for 2 - x, and for 3 and 4 - xxx.

5 ASSIGNING "DON'T CARE" TRANSITIONS

We now present an algorithm for augmenting a partial reduced FSM with several inputs. The algorithm uses formulae (3.1) - (3.3) to estimate the expected length of a test suite. If it exceeds the lower bound $L_{\min}$ defined by (3.3), the algorithm repeatedly tries all inputs labeling "don't care" transitions and calls Algorithm 1. The Wp-method [FBK91] is used to derive a resulting test suite which is complete in the class of machines with an equal or fewer number of states.

Algorithm 2.

```
Input: A partial reduced FSM $A$.
Output: An augmented FSM $A'$ and a complete test suite of length not greater than that of $A$.
Step 1. Calculate the expected length $L_A$ of a test suite for $A$.
If $L_A = L_{\min}$ then GO TO STEP 3.
Step 2. $C := A$.
Step 2. Let $X_u = \{x_1, ..., x_q\}$ be the set of inputs labeling "don't care" transitions in $C$.
For each $x_i \in X_u$
        Call Algorithm 1 to assign "don't care" transitions labeled with the input $x_i$.
        Add newly defined transitions into the FSM $C$.
        Let the augmented machine be $C_i$.
        Calculate the expected length of a test suite for $C_i$.
Step 3. Let $C^*$ be an FSM $C_i$ or $C$ with the shortest expected test suite.
If $C^* = C$ then GO TO STEP 4.
$C := C^*$. $X_u = X_u\backslash\{x_i\}$. If $X_u \neq \emptyset$ then GO TO STEP 2.
Step 4. $A' := C^*$. Call the Wp-method to derive a complete test suite for the machine $A'$.   □
```

Remarks on Algorithm 2:

1) We assume that an FSM is given in its reduced form; however, this assumption is indeed not restrictive. The algorithm also accepts FSMs that are not reduced, i.e. that have compatible states [Gill62]. However, under the "undefined by default" convention for "don't care"

transitions, it is recommended first to reduce such a machine by merging compatible states, and then to apply Algorithm 2 to its reduced form. This is because a machine with fewer states usually requires shorter tests. There may exist several reduced forms of a nonreduced partial FSM, unlike the case of complete FSMs; and it is desirable to choose the most testable reduced form in this case. More research is required in this direction.

2) The resulting FSM is not necessarily a completely specified machine, some "don't care" transitions might be left intact. As discussed in Section 3, a shorter state cover usually leads to a shorter test suite. Undefined transitions can be assigned to reduce the total length of a state cover of the machine and eventually that of a complete test suite.

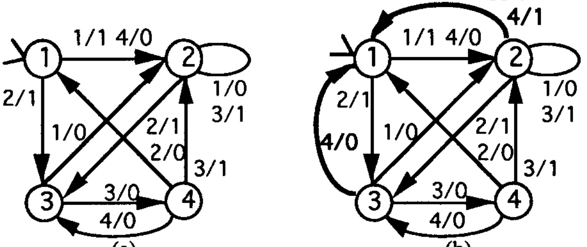
3) The algorithm tries all inputs which label "don't care" transitions. To facilitate its early termination we can arrange inputs such that the overall number of converging and equivalent states for a corresponding input form a non-decreasing sequence. In particular, if there exists an input, such that the necessary and sufficient conditions for the existence of a homogeneous distinguishing sequence are satisfied, then Algorithm 2 assigns the "don't care" transitions labeled with that input.

4) Comparison of possible augmentations with respect to different inputs is based on the expected test suite length. If instead, a test derivation method, such as the Wp-method, is called to derive a test suite whenever its length is required to make a decision, the user can stop the process once a test suite of an acceptable size is obtained. In the worst-case scenario, the method would be called $q(q+1)/2$ times, where q is the number of inputs labeling "don't care" transitions in the original machine.

6 APPLICATION EXAMPLES

Example 6.1

Consider the FSM A shown in Figure 6.1a. It is reduced and partially specified. There are five "don't care" transitions in this machine. Each of the transitions can lead to one of five states with output 0 or 1; alternatively, it can be left intact. Altogether, there exist $11^5 = 161051$ completely and partially specified machines that are augmentations of the given FSM A. A "perebor", i.e. an exhaustive procedure must try all of them, derive a complete test suite for each, and choose a machine with the shortest test suite. Instead, we apply our method.



(a) (b)

Figure 6.1 The FSM and its augmented FSM.

First we derive a complete test suite for the given machine using the Wp-method. Assuming state 1 as an initial state, the state cover is $V = \{\varepsilon, 1, 2, 23\}$. The transition cover is $TC = \{\varepsilon, 1, 2, 4, 11, 12, 13, 21, 23, 232, 233, 234\}$. The set $W = \{1, 2, 3\}$ is a characterization set of A. The state identifiers are: $W_1 = \{1, 2\}$, $W_2 = \{1, 2, 3\}$, $W_3 = \{1, 3\}$, $W_4 = \{1, 3\}$. The resulting test suite complete in the class of FSMs with up to four states is: {111, 112, 113, 121, 123, 131, 132, 133, 211, 212, 213, 2321, 2322, 2331, 2332, 2333, 2341, 2343, 41, 42, 43}. There are 21 test cases of total length 67. The formula (3.1) gives the expected length of 90. (3.3) returns the lower bound $L_{\min} = 30$.

Every input labels at least one "don't care" transition, but only for input 4 are there no converging states. We choose this input and construct a submachine of A. It is, in fact, the machine shown in Figure 4.5a. The corresponding augmented submachine is the one shown in Figure 4.5b. It has a homogeneous distinguishing sequence 444. According to this submachine, two transitions must be added to the original machine, namely 2-4/1->1 and 3-4/0->1. We include them into the FSM *A* and obtain the augmented FSM *A'* shown in Figure 6.1b. The additional transitions are depicted in bold. Notice that three other "don't care" transitions are left intact. *A'* has the following state identifiers (as constructed in Section 3.3): $W'_1 = \{44\}$, $W'_2 = \{4\}$, $W'_3 = \{444\}$, $W'_4 = \{444\}$. Based on the obtained identifiers, we can derive a complete test suite of length 39. As a result, the length is reduced by about 40%. The "don't care" transitions labeled with inputs 1, 2 and 3 remain, since the length cannot be further reduced.

Example 6.2. The INRES protocol

To illustrate the proposed approach to improving the testability of a partially specified protocol machine, we consider the INRES protocol [Hogr91]. The behavior of the responder part of this protocol can be specified by an FSM given in Figure 6.2 (plain lines only).

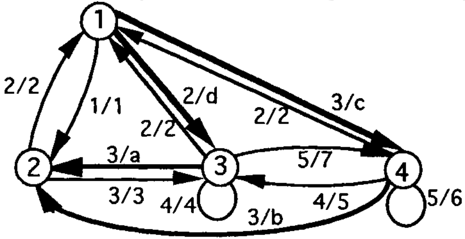


Figure 6.2 The INRES Responder.

The input alphabet is: 1- CR, 2 - IDISr, 3 - ICONrsp, 4 - DT0, 5 - DT1. The output alphabet is 1 - ICONi; 2 - DR; 3 - CC; 4 - ACK0; 5 - ACK0, IDATi; 6 - ACK1; 7 - ACK1, IDATi; 8 - null. The machine is partially specified. The traditional way of augmenting such a machine is based on the completeness assumption [SiLe89], [PBD93]. In particular, all "don't care" transitions are replaced by looping transitions with the null output (not depicted). Following this approach, we first obtain a completely specified FSM and derive a test suite complete for implementations with up to four states, as follows. A state cover is $V = \{\varepsilon, 1, 13, 135\}$. $W = \{41\}$. We apply the Wp-method and obtain a test suite with 19 test cases of total length 76.

Next, we assume that the behavior of the responder for all service primitives from the INRES user can be defined in an arbitrary way, whereas the completeness assumption should still be applied for all incoming PDUs. In particular, the transitions (1, 2), (1, 3), (3, 3), (4, 3) in states 1 and 3 on inputs 2 (IDISr) and 3 (ICONrsp) are "don't care" transitions. This machine requires a complete test suite with 15 test cases of total length 61.

Now we follow the proposed approach to find an augmented machine with a shorter test suite. Applying Algorithm 2 we can obtain the transitions 1-3/*c*->4, 3-3/*a*->2, 4-3/*b*->2 shown in Figure 6.2 as bold lines. Here *a*, *b*, and *c* are different ouputs which can be arbitrary chosen from the set {1, 2, 4, 5, 6, 7, 8}. The obtained machine has a homogeneous distinguishing sequence, that is $W' = \{3\}$ (ICONrsp). There is only one "don't care" transition left in state 1 on input 2. We define a transition 1-2/*d*->3, where *d* is an arbitrary output in order to reduce a state cover. It now has fewer symbols: $V' = \{\varepsilon, 1, 2, 3\}$. Given the sets *W'* and *V'*, we now have a complete test suite (produced by the same method): *TS* = {41, 541, 441, 341, 241, 1241, 13241, 135241, 141, 1541, 1441, 1141, 1344, 1334, 1314, 13544, 13554, 13534, 13514}. It comprises 17 test cases of total length 49. Thus, the obtained version of the INRES responder is more testable than the original one and the version based the completeness assumption. This assumption widely cited in the literature may deteriorate the testability of a protocol, as our example shows.

To assess the effectiveness of the method we have conducted the following experiment. A tool was designed to enumerate all $(1+4\cdot 8)^4 = 1185921$ of the possible augmented machines for the INRES responder, derive a complete test suite for each of them, and find an FSM with the shortest one. A test derivation tool used to generate test suites implements the method developed for partial FSMs in [Petr91]. The experiment shown that all augmented FSMs require no fewer than 49 test events for a complete test suite.

7 CONCLUSION

In this paper, we have addressed one particular problem of design for testability of protocols on the specification level. We have developed an approach to improving testability of the given protocol taking advantage of the fact that a protocol is usually specified only partially and certain state/input combinations can be set in an arbitrary way. The feasibility of the approach was proven on partially specified FSMs with "don't care" transitions. Its effectiveness was demonstrated by conducting an experiment on the INRES protocol.

Though an algorithm given in this paper guarantees that the identifying sequences in the resulting FSM are quite short (their lengths do not exceed the number of states), it does not yet guarantee to produce the shortest possible ones. Thus, our algorithm can be further refined to construct an augmented machine with near-optimal identifying sequences. We continue our research in this direction. The work in progress also concerns the adaptation of the basic ideas underlying the proposed approach to nondeterministic and extended finite state machines.

In this paper, we have also presented some useful estimations of the expected length of complete test suites which are used to guide the process of augmenting partially specified machines. These estimations can also be used to select parameters of transition covers and characterization sets usually left without any guidance by most existing test derivation methods.

We have considered DFT in the context of the test derivation methods that rely on a reset facility, however the presented algorithms can be used in conjunction with other methods which do not use the reset. By augmenting a partial machine, a variety of UIO's or even distinguishing sequences are usually created. A nice property of the resulting machine is that lengths of identifying sequences never exceed the number of states. Therefore, any UIO-based method should yield a short test sequence. The presented approach can also be easily generalized to incorporate additional factors influencing the testability, such as length of transfer sequences (test preambles and postambles), the cost assigned to protocol messages, and others.

Acknowledgments

This work was partly supported by the HP-NSERC-CITI Industrial Research Chair on Communication Protocols at Université de Montréal, the NSERC Individual Research Grant (R. Dssouli) #20629188, and by the Russian Found for Fundamental Research. The authors wish to thank Q. M. Tan who has designed a test derivation tool for his help in experiments and S. A. Ezust for comments.

8 REFERENCES

[ABF90] M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design, Computer Science Press, Oxford, England, 1990.

[BoPe94] G. v. Bochmann and A. Petrenko, "Protocol Testing: Review of Methods and Relevance for Software Testing", ISSTA'94, ACM International Symposium on Software Testing and Analysis, Seattle, U.S.A., 1994, pp. 109-124.

[BPY94] G. v. Bochmann, A. Petrenko, and M. Yao, "Fault Coverage of Tests Based on Finite State Models", the Proceedings of IFIP TC6 Seventh IWPTS'94, Japan.

[Chow78] T. S. Chow, "Testing Software Design Modeled by Finite-State Machines", IEEE Transactions on Software Engineering, Vol. SE-4, No.3, 1978, pp.178-187.

[DsFo91] R. Dssouli and R. Fournier, "Communication Software Testability", IFIP Transactions, Protocol Testing Systems III (the Proceedings of IFIP TC6 Third International Workshop on Protocol Test Systems), Ed. by I. Davidson and W. Litwack, North Holland, 1991, pp.45-55.

[FBK91] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, A. Ghedamsi, "Test Selection Based on Finite State Models", IEEE Transactions on Software Engineering, Vol. SE-17, No.6, 1991, pp.591-603.

[Gill62] A. Gill, Introduction to the Theory of Finite-State Machines, McGraw-Hill, 1962.

[Henn64] F. C. Hennie, "Fault Detecting Experiments for Sequential Circuits", IEEE 5th Ann. Symp. on Switching Circuits Theory and Logical Design, 1964, pp. 95-110.

[Hogr91] D. Hogrefe, "OSI Formal Specification Case Study: The Inres Protocol and Service", University of Berne, Technical Report IAM-91-012, University of Berne, 1991.

[Jose78] J. Joseph, "On Easily Diagnosable Sequential Machines", IEEE Transactions on Computers, Vol. C-27, February, 1978, pp.159-162.

[LeYa94] D. Lee and M. Yannakakis, "Testing Finite-State Machines: State Identification and Verification", IEEE Trans. on Computers, Vol. 43, No. 3, 1994, pp. 306-320.

[Petr91] A. Petrenko, "Checking Experiments with Protocol Machines", IFIP Transactions, Protocol Test Systems, IV (the Proceedings of IFIP TC6 Fourth International Workshop on Protocol Test Systems, 1991), Ed. by Jan Kroon, Rudolf J. Heijink and Ed Brinksma, 1992, North-Holland, pp. 83-94.

[PBD93] A. Petrenko, G. v. Bochmann, and R. Dssouli, "Conformance Relations and Test Derivation", IFIP Transactions, Protocol Test Systems, VI, (the Proceedings of IFIP TC6 Fifth International Workshop on Protocol Test Systems, 1993), Ed. by O. Rafiq, 1994, North-Holland, pp.157-178.

[PDK93] A. Petrenko, R. Dssouli, and H. Konig, "On Evaluation of Testability of Protocol Structures", IFIP Transactions, Protocol Test Systems, VI, (the Proceedings of IFIP TC6 Fifth International Workshop on Protocol Test Systems, 1993), Ed. by O. Rafiq, 1994, North-Holland, pp.111-123.

[ShLe94] M. L. Sheu and C. L. Lee, "Symplifying Sequential Circuit Test Generation", IEEE Design and Test of Computers, Fall 1994, pp. 28-38.

[SiLe89] D. P. Sidhu and T. K. Leung, "Formal Methods for Protocol Testing: A Detailed Study", IEEE Trans. on Software Engineering, Vol. SE-15, No.4, 1989, pp.413-426.

[TyBa75] T. Tylaska and J. D. Bargainer, "An Improved Bound for Checking Experiments that Use Simple Input-Output and Characterizing Sequences", IEEE Transactions on Computers, Vol. C-24, No.6, 1975, pp. 670-673.

[Vasi73] M. P. Vasilevski, "Failure Diagnosis of Automata", Cybernetics, Plenum Publishing Corporation, New York, No.4, 1973, pp.653-665.

[VLC93] S. T. Vuong, A. A. F. Loureiro, and S. T. Chanson, "A Framework for the Design for Testabilitiy of Communication Protocols", in the Proceedings of IFIP TC6 Fifth IWPTS'93, Ed. by O. Rafiq, 1994, North-Holland, pp.89-108.

9 BIOGRAPHY

Nina Yevtushenko received the Dipl. degree in radio-physics in 1971 and Ph. D. in computer science in 1983, both from the Tomsk State University, Russia. She is now a Professor at that University. Her research interests include the automata and FSM theory and testing problems.

Alexandre Petrenko received the Dipl. degree in electrical and computer engineering from Riga Polytechnic Institute in 1970 and the Ph.D. in computer science from the Institute of Electronics and Computer Science, Riga, USSR, in 1974. Since 1992, he has been with the Université de Montréal, Canada. His current research interests include communication software engineering, protocol engineering, conformance testing, and testability.

Rachida Dssouli received the Doctorat d'université degree in computer science from the Université Paul-Sabatier of Toulouse, France, in 1981, and the Ph.D. degree in computer science in 1987, from the Université de Montréal, Canada. She is currently an Associate professor at the University of Montréal. Her research area is software engineering and her research interests include software specification and testability, protocol testing and observation.

Kamel Karoui is a Ph.D. student of the Université de Montréal, Canada.

Svetlana Prokopenko is a Ph.D. student of the Tomsk State University, Russia.