An International Journal
computers & mathematics with applications

PERGAMON Computers and Mathematics with Applications 37 (1999) 125–135

Synchronization Paradigm for Protocol Testing under Multiparty Configuration

RONG S. LIN
Department of Computer Science and Information Engineering
National Chiao Tung University
Hsin-Chu, Taiwan, R.O.C.
and
Chunghwa Telecommunication Laboratories
Chung-Li, Taiwan, R.O.C.

MARIA C. YUANG
Department of Computer Science and Information Engineering
National Chiao Tung University
Hsin-Chu, Taiwan, R.O.C.

(Received April 1997; revised and accepted December 1997)

Abstract—Protocol testing leads to the synchronization problem should test sequences be applied to multiple distanced testers, namely under the multiparty configuration. This paper presents a novel synchronization paradigm which seamlessly unifies two synchronization techniques, self-synchroniable sequences and external synchronization operations, by means of the *state expansion transformation*. In the paradigm, the protocol specification is first transformed into a state expansion digraph with two pieces of datum augmented. They are:

(a) a zero cost assigned to each synchronization-problem-free crossing from one transition to another transition, and
(b) a weighted cost assigned to each external synchronization operation whenever the synchronization is deemed necessary.

On the basis of the state expansion transformation, synchronizable, optimal sequences for testing can be efficiently derived. To demonstrate the viability of the proposed paradigm, we present the generations of two synchronizable sequences, namely the synchronizable preamble and the synchronizable distinguishing sequence, which have previously been used for the testing of the correctness of a protocol's transition. The paper also shows that the complexities of the two sequences generations are also polynomial-bounded, i.e., $O((np^2)\log(np))$ and $O((n^2p^2)\log(np))$, respectively, where n and p are the numbers of states and input symbols of the protocol specification. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords—Protocol testing, Multiparty configuration, Synchronization problem, Synchronizable test sequence, Preamble, Distinguishing sequence.

1. INTRODUCTION

Protocol testing [1,2] examines the conformity of an Implementation Under Test (IUT) against its protocol specification [3]. To approach this problem, a set of input/output sequences, (i.e., test cases) is initially derived from the protocol specification. Testing is then performed by applying the sequence of inputs to the IUT and verifying the sequence of outputs in response. However, predetermined test sequences may cause the synchronization problem [4] should they be applied to multiple distanced testers [1]. Such a problem arises when the i^{th} Remote Tester (RT) cannot determine when to send an input to the IUT due to ambiguity over whether the IUT has assumedly sent an output to the j^{th} ($j \neq i$) RT or not.

The synchronization problem can be resolved by two approaches:

(a) constructing *Self-Synchronizable* (SS) sequences [5,6], and
(b) employing *External-Synchronization* (ES) operations [1,7].

The first approach takes advantage of the fact that a protocol's SS sequence is considered synchronization-problem free should the sequence solely consist of the protocol's transitions. Namely, the transitions in an SS sequence are intentionally arranged so that the previous transition always informs the input RT of any transition, thereby eliminating the synchronization problem. Unfortunately, this approach has two limitations. First, SS sequences may not exist [8]. For ensuring the existence of SS sequences, requirements of assumptions [6,9] on protocols under consideration confine the method's applicability. Second, should other synchronization mechanisms, e.g., ES operations, be prohibited, such incomplete testing renders some faults undetectable [10]. In contrast to constructing SS sequences, the alternative approach relies on employing ES operations, such as via manual cooperation or extra reliable communication channels, to realize synchronization between RTs. This approach is constrained by the high cost incurred by ES operations. The fact that ES operations are inevitable but should be reduced to minimum inspires the development of a method of combining the features of SS sequences and ES operations for protocol testing.

This paper presents a novel synchronization paradigm which seamlessly unifies SS sequences and ES operations by means of the state expansion transformation. Initially in the paradigm, the protocol specification is transformed into a state expansion digraph with two pieces of datum augmented:

(a) a zero cost assigned to each synchronization-problem-free crossing from one transition to another transition, and
(b) a weighted cost, say α, assigning to each ES operation whenever the synchronization is deemed necessary.

On the basis of the state expansion transformation, synchronizable, optimal sequences for testing can be efficiently derived.

The notion of this synchronization paradigm has several advantages. First, the cost each transition is associated with can be used in the quantitative analysis for selecting SS sequences or ES operations when both types of test sequences are available. In addition, the fact that the value α can be dynamically adjusted reflects efforts to synchronize RTs in a realistic test environment. For instance, synchronizing testers within a local site is less expensive than synchronizing those locate remotely. Moreover, assigning the value α to infinite indicates the disregard for the use of ES operations. Owing to the flexibility and practicability, synchronizable, optimal sequences for testing can be derived.

To demonstrate the viability of the proposed paradigm, we present the generations of two synchronizable sequences, namely the synchronizable preamble [1] and the synchronizable distinguishing sequence [11]. Notably, a test case for testing the correctness of a protocol's transition comprises of three parts:

(a) the preamble: a sequence which directs the protocol from the initial state to the head state of the transition being tested;
(b) the transition: testing of the transition itself involves the checking of the correctness of the output symbol upon applying the input symbol; and
(c) the postamble: a sequence for verifying the correctness of the transition's tail state.

In particular, the postamble can be adopted by a set of distinguishing sequences proposed by the W-method [3,11–13]. Therefore, the second goal of the paper is the provision of the two synchronizable sequences on the basis of the proposed paradigm, realizing the synchronization-problem-free W-method.

The rest of the paper is organized as follows. Section 2 defines the protocol model and the synchronization problem. Section 3 presents the state expansion transformation of the proposed paradigm. The algorithms of the generations of the synchronizable preamble and the synchronizable distinguishing sequence are given in Sections 4 and 5, respectively. Concluding remarks are finally made in Section 6.

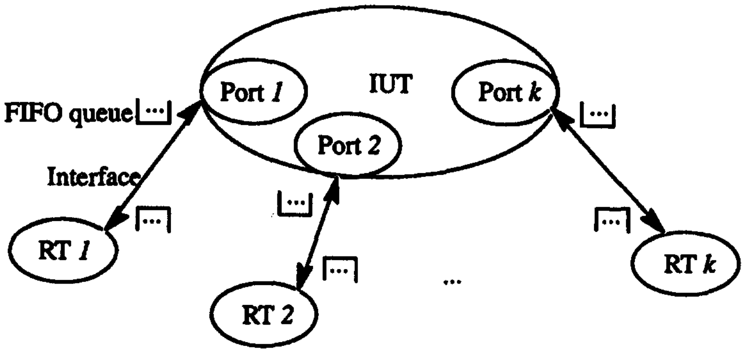

Figure 1. Multiparty-configuration testing.

2. MODEL AND PROBLEM

The test environment, referred to as the *multiparty configuration* [1], consists of k ($k \geq 2$) RTs connected to the IUT through k interfaces, as depicted in Figure 1. Each interface is comprised of two FIFO queues attached to two end points. Upon receiving an input from an RT (i.e., the *input*-RT), the IUT updates its internal state and responds with a set of outputs to multiple RTs (i.e., the *output*-RTs). The IUT is thus formally specified as a *k-port Finite State Machine* (FSM) [3,10,13] represented as a digraph $G = (S, E)$, where S denotes a nonempty set of states and each transition $T_e : [S_i, S_j; (a, m_a)/(x, m_x), (y, m_y), \ldots (z, m_z)] \in E$ represents that, as input a is received from RT m_a, the FSM generates outputs $x, y, \ldots z$ to RTs $m_x, m_y, \ldots m_z$, respectively, and progresses from state S_i to S_j. Notably, one input-RT and one or more output-RTs exist for each transition. Moreover, for later explanation of the notations, transition set E can be considered as the combining of the *next-state function* $\delta : S \times I \rightarrow S$ and the *output function* $\lambda : S \times I \rightarrow O$, where I and O are nonempty sets of input symbols and output symbols, respectively. For instance, transition T_e mentioned above is the combination of $\delta(S_i, (a, m_a)) = S_j$ and $\lambda(S_i, (a, m_a)) = (x, m_x), (y, m_y), \ldots (z, m_z)$. Finally, the protocol FSM under consideration is assumed to be deterministic, minimized, strongly-connected, and completely-specified [3,14]. In addition, there exists a particular transition, say init, which resets the IUT to a specified initial state, and is associated with a zero cost. For instance, Figure 2 presents a three-port FSM.

DEFINITION 1. *The Transition Type (TT) of transition T_e is defined as $TT(T_e) = (m_i, M_o)$, $m_i \in \{1 \ldots k\}$ and $M_o \subseteq \{1 \ldots k\}$, where m_i denotes the input-RT identification and M_o represents the set of output-RT identifications, respectively.*

DEFINITION 2. *For two adjacent transitions T_e and T_f, where $TT(T_e) = (m_i, M_{oi})$ and $TT(T_f) = (m_j, M_{oj})$, sequence T_e–T_f is synchronous, (i.e., synchronization-problem-free) if and only if $m_j = m_i$ or $m_j \in M_{oi}$. Moreover, the Enabled Set (ESet) of transition T_e is defined as ESet (T_e) $= \{m_i\} \cup M_{oi}$. Therefore, sequence T_e–T_f is synchronous if and only if $m_j \in$ ESet (T_e). As an illustration, Table 1 summarizes the associated TT and Eset of each transition of the protocol FSM in Figure 2.*

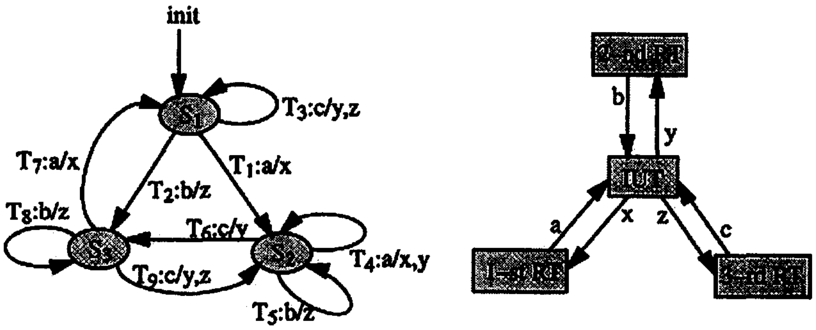

Figure 2. A schematic diagram of a three-port FSM.

Table 1. TT and ESet of each transition of the protocol FSM in Figure 2.

Transition	TT	ESet	Transition	TT	ESet
T_1	$(1, \{1\})$	$\{1\}$	T_2	$(2, \{3\})$	$\{2, 3\}$
T_3	$(3, \{2, 3\})$	$\{2, 3\}$	T_4	$(1, \{1, 2\})$	$\{1, 2\}$
T_5	$(2, \{3\})$	$\{2, 3\}$	T_6	$(3, \{2\})$	$\{2, 3\}$
T_7	$(1, \{1\})$	$\{1\}$	T_8	$(2, \{3\})$	$\{2, 3\}$
T_9	$(3, \{2, 3\})$	$\{2, 3\}$	init	$(-, \{1\})$	$\{1\}$

DEFINITION 3. *A sequence is self-synchronizable (SS) if every subsequence consisting of two adjacent transitions is synchronous.*

The synchronization problem can be resolved by constructing SS sequences or employing ES operations. For instance, sequence init $-T_1-T_4-T_5-T_6$, which directs the protocol FSM in Figure 2 from the initial state to state S_3, is self-synchronizable. Similarly, sequence init$-$ESO$(1, 2) - T_2$, where ESO(i, j), $1 \le i, j \le k$, denotes the ES Operation of the i^{th} RT informing the j^{th} RT of the valid time to send the next input, also directs the FSM from initial state to state S_3 without the synchronization problem. However, in addition to the cost of normal transitions, the latter sequence requires an extra cost of realizing ES operations.

3. STATE EXPANSION TRANSFORMATION

Generally, in the synchronization paradigm, the protocol specification is first transformed into a state expansion digraph with two pieces of datum augmented:

(a) a zero cost assigned to each synchronous crossing from one transition to another transition, and

(b) a weighted cost assigned to each external synchronization operation.

Specifically, the construction of the state expansion digraph, or the transformation, is comprised of two steps.

STEP 1. Each state S_i in the protocol FSM is replaced by a set of Attach Points (APs), where each AP is either the beginning point of an outgoing transition or the ending point of an incoming transition of S_i. The beginning AP and ending AP of transition T_e are hereinafter denoted as B_e and E_e, respectively.

STEP 2. For any two adjacent transitions T_e and T_f with common state S_i, where E_e and B_f denote the corresponding APs of S_i, a transition from E_e to B_f is created. The added transition is either a *dummy transition* with zero cost if sequence T_e–T_f is synchronous, or a *weighted transition* with cost α if sequence T_e–T_f is not synchronous.

As an illustration of the state expansion transformation, Figure 3 presents the state expansion digraph of the protocol FSM in Figure 2. We further analyze the complexity of the state expansion transformation. Assume that there are a total of n states and p legitimate input symbols in the protocol FSM, namely a total of np transitions. For the state expansion digraph, there are $2np+1$ APs because each transition in the original FSM is associated with two APs and transition init with an AP R. Furthermore, in addition to original np transitions, np^2 transitions are also created because, for each original transition, p legitimate transitions are outgoing from its ending state. Consequently, there are in total $2np+1$ APs and $np+np^2$ transitions in the state expansion digraph. In other words, there are $2*|E|+1$ APs and $|E|+p*|E|=(1+p)*|E|$ transitions in the state expansion digraph, where $|E|$ denotes the number of transitions in the original protocol FSM.

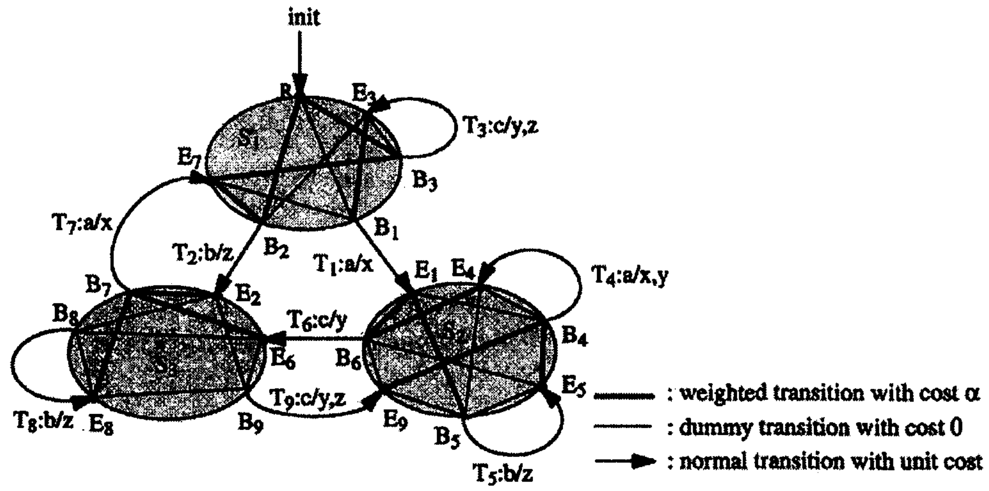

Figure 3. State expansion digraph of the FSM in Figure 2.

4. SYNCHRONIZABLE PREAMBLE

Based on the synchronization paradigm proposed above, *synchronizable preambles* can be generated. The *preamble* [15] of state S_i is the shortest (minimum-cost) I/O sequence which directs the IUT from the initial state to state S_i. A synchronizable preamble is then a synchronous preamble. The preamble stationing the IUT in state S_i is normally prefixed to a Transition Under Test (TUT) with head state S_i in the test case for testing the correctness of the transition. Notice that the synchronization problem may still occur when the IUT progresses from the synchronizable preamble to the following TUT. This fact implies that constructing a synchronous test case requires characterizing a synchronizable preamble by the ending state and the input-RT identification of the subsequent TUT. Therefore, a synchronizable preamble is represented as $\mathrm{SP}(S_i,x)$. Formally speaking, $\mathrm{SP}(S_i,x)$ is the synchronous, shortest I/O sequence which takes the IUT from the specified initial state to state S_i, and does inform the x^{th} RT of the valid time to send the next input.

The following scenario illustrates the generation of the synchronizable preamble, say $\mathrm{SP}(S_i,x)$. The state expansion transformation of the protocol FSM is first performed as mentioned in the previous section. Next, the shortest paths with associated costs from R to all APs can be derived by performing the Single Source All Shortest Paths (SSASP) algorithm [16] from AP- R. Moreover, the fact that the first attribute of target $\mathrm{SP}(S_i,x)$ is the ending state where the IUT is directed to, accounts for why only the shortest paths to ending APs are considered herein. All ending APs attached to the same state and their shortest paths can be further grouped together; thus, a total of n groups corresponding to n states are constructed. Therefore, sequentially searching for each group i allows us to select the one with the minimum cost from the shortest paths belonging to the same group and, alternately, obtain the set $\{(\mathrm{SP}(S_i,A_i),c) \mid 1 \le i \le n,\ A_i \subseteq \{1\ldots k\}\}$, where A_i denotes the *Enabled Set* of the last transition in the shortest path just selected, and c is the derived cost of the selected path.

Unfortunately, the target $\mathrm{SP}(S_i, x)$ may not be derived yet since it is possible that $x \notin A_i$ for group i. All $\mathrm{SP}(S_i, z)$, $1 \le z \le k$, for each group i should be further derived for completeness. Sequentially searching for each group i one more time is then deemed necessary to identify the shortest path for other input-RT identification z, $z \notin A_i$ and $1 \le z \le k$. Interestingly, simply picking up the shortest one from the set of paths belonging to group i and their *Enabled Set* containing input-RT identification z may yield a wrong $\mathrm{SP}(S_i, z)$. A valuable strategy in the second round is, if the cost of the considered path exceeds $\alpha + c$, where c denotes the cost of the selected shortest path in the first round for each group i, the sequence $\mathrm{SP}(S_i, z)$ derived previously should be replaced by the concatenated sequence of $\mathrm{SP}(S_i, A_i)$ and $ESO(y, z), y \in A_i$, and with cost $\alpha + c$. This strategy is because the most cost-effective approach of directing the IUT from the initial state to state S_i and enabling the z^{th} RT is via $\mathrm{SP}(S_i, A_i)$ derived in the first round. Then, $ESO(y, z)$ is performed, where $y \in A_i$.

The overall algorithm is formally specified as follows.

Algorithm: The Synchronizable Preamble Generation

INPUT. A k-port protocol FSM.

OUTPUT. The set $\{(\mathrm{SP}(S_i, z), c) \mid 1 \le i \le n,\ 1 \le z \le k\}$.

STEP 1. Construct the state expansion digraph of the protocol FSM.

STEP 2. Construct all shortest paths from R to all ending APs by performing the SSASP algorithm.

STEP 3. Execute the first-round sequential search to obtain the set $\{(\mathrm{SP}(S_i, A_i, c)) | 1 \le i \le n,$ $A_i \subseteq \{1 \dots k\}\}$, where A_i represents the *Enabled Set* of the last transition in the shortest path just selected and c is the cost of the selected path.

STEP 4. By following the above mentioned strategy, execute the second-round sequential search to identify other $(\mathrm{SP}(S_i, z), c)$, where $z \notin A_i$ and $1 \le z \le k$.

The full set $\{(\mathrm{SP}(S_i, z), c) | 1 \le i \le n, 1 \le z \le k\}$ is thus derived.

Complexity Analysis

Assume that there are n states and p legitimate input symbols in the protocol FSM. Thus, there are a total of $2np+1$ APs and $np+np^2$ transitions in the state expansion digraph. Since the SSASP algorithm requires an execution time of $O((|S|+|E|) \log |S|)$, where $|S|$ and $|E|$ denote the numbers of states and transitions of the target digraph, respectively, the algorithm thus requires a polynomial-bounded execution time of $O(((2np+1)+(np+np^2)) \log(2np+1)) = O((np^2) \log(np))$.

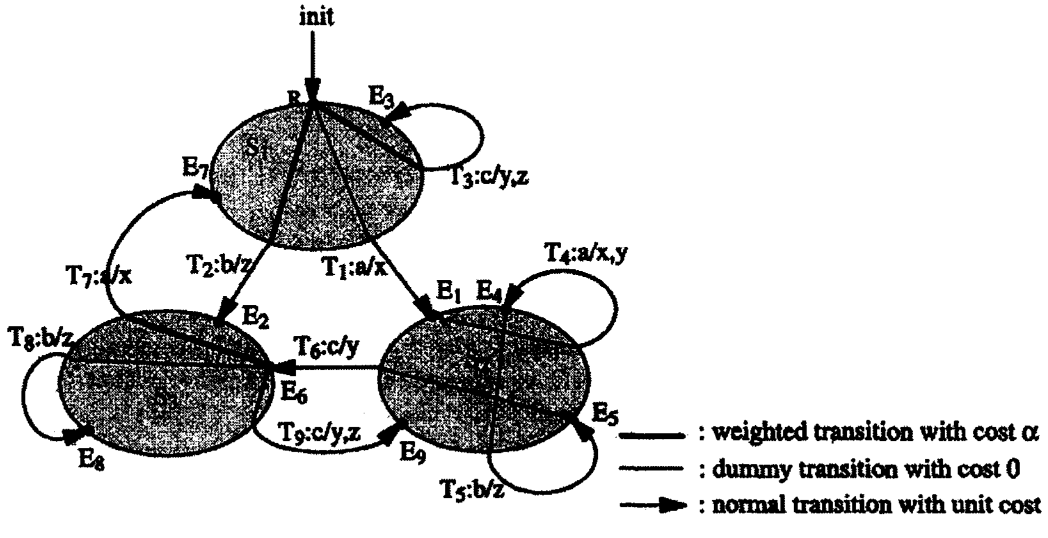
init
R
E3
T3:c/y,z
E7
T7:a/x
T2:b/z
T1:a/x
E1
E4
T4:a/x,y
E2
T8:b/z
T6:c/y
E6
E5
T9:c/y,z
E9
E8
T5:b/z
: weighted transition with cost α
: dummy transition with cost 0
: normal transition with unit cost

Figure 4. Shortest paths of the diagraph in Figure 3.

As an illustration, Figure 4 and Table 2 summarize all the shortest paths and the SPs derived from Figure 3, in which the cost of an ES operation $\alpha = 4$ is given. For instance, SP(S_3, 2) is sequence init $-T_1-T_4-T_5-T_6$ with cost 4. In addition, SP(S_3, 1) is the concatenated sequence of SP(S_3, 2) and ESO(2, 1) with aggregate cost $4 + \alpha = 8$. For each state, three SPs, i.e., SP(S_i, 1), SP(S_i, 2), and SP(S_i, 3), corresponding to three RTs are derived. Once the table is constructed, for a given ending state and an input-RT identification, the synchronizable preamble can be easily derived.

Table 2. Synchronizable preambles.

Source AP	Ending State	Ending AP	Cost	Enable Set	Derived SPs in First Round	Derived SPs in Second Round
R	S_1	I	0	$\{1\}$	(SP(S_1, $\{1\}$), 0)	(SP(S_1, 2), 4) (SP(S_1, 3), 4)
R	S_1	E_3	5	$\{2, 3\}$		
R	S_1	E_7	9	$\{1\}$		
R	S_2	E_1	1	$\{1\}$	(SP(S_2, $\{1\}$), 1)	
R	S_2	E_4	2	$\{1, 2\}$		(SP(S_2, 2), 2)
R	S_2	E_5	3	$\{2, 3\}$		(SP(S_2, 3), 3)
R	S_2	E_9	5	$\{2, 3\}$		
R	S_3	E_2	5	$\{2, 3\}$		
R	S_3	E_6	4	$\{2, 3\}$	(SP(S_3, $\{2, 3\}$), 4)	(SP(S_3, 1), 8)
R	S_3	E_8	5	$\{2, 3\}$		

5. SYNCHRONIZABLE DISTINGUISHING SEQUENCE

Based on the same synchronization paradigm, in this section we present another example of how to generate *Synchronizable Distinguishing Sequences* (SDSs). *Distinguishing sequences* of two states are used to distinguish one state from another. For protocol testing, the W-method further utilizes a set of distinguishing sequences (i.e., *characterizing set*) [13] to ascertain which state the IUT is currently in. According to the Automata Theory [3,17], distinguishing sequences of any two states in a protocol FSM always exist if the FSM is minimized. Formally speaking, states S_i and S_j are considered to be *distinguishable* if an input sequence exists such that, upon applying the input sequence, states S_i and S_j produce different output sequences. Such an input sequence is thus referred to as a distinguishing sequence of states S_i and S_j. For instance, states S_1 and S_3 in Figure 2 are distinguishable since they produce different output sequences xy and $x(y, z)$ after applying the input sequence, ac.

However, predetermined distinguishing sequences may lead to the synchronization problem should they be applied to multiple distanced testers. For instance, distinguishing sequence ac of states S_1 and S_3 incurs the synchronization problem, even causing the states to produce different output sequences. Surprisingly, another input sequence aa is also a distinguishing sequence of states S_1 and S_3 but free from the synchronization problem. Such a synchronous distinguishing sequence is thus called an SDS and notably required for protocol testing under the multiparty configuration.

In contrast to the preamble prefixed to a TUT, the distinguishing sequence, as one member subsequence of the postamble verifying the tail state of the TUT, is postfixed to the TUT. This situation implies that an important attribute of an SDS is the enabled input-RT identification, i.e., one member of the ESet of the antecedent TUT. Therefore, an SDS is characterized by the state-pair to be distinguished and the required input-RT identification of the first transition of the SDS which is confined by the ESet of the antecedent transition. An SDS is thus denoted as SDS((S_i, S_j), x), where x denotes the desired input-RT identification. Next, before applying the proposed synchronization paradigm, we introduce an algorithm for generating distinguishing sequences: the *compound digraph* approach [18–20]. Therefore, we illustrate how to perform the state expansion transformation for a compound digraph and then generate all SDSs.

A compound digraph ($G \times G$) of a protocol FSM (G) is defined and constructed as follows. The input and output symbols of $G \times G$ are the same as those of G. The node set in $G \times G$ consists of all possible state-pairs in G as well as a newly created node, called the *Source*. Considering a node, $[S_i, S_j]$ ($S_i \neq S_j$), with the input a received from RT m_a in $G \times G$, the outgoing transition of the node is determined as follows.

1. If $\lambda(S_i, (a, m_a)) \neq \lambda(S_j, (a, m_a))$, i.e., different output symbols, in G, S_i, and S_j are *immediately distinguishable* by input (a, m_a), a transition from $[S_i, S_j]$ to the *Source* with label $(a, m_a)/-$ is created.
2. If $\lambda(S_i, (a, m_a)) = \lambda(S_j, (a, m_a))$ and $\delta(S_i, (a, m_a)) = \delta(S_j, (a, m_a)) = S_k$, i.e., same output symbol and same next state, in G, a transition from $[S_i, S_j]$ to $[S_k, S_k]$ with label $(a, m_a)/\lambda(S_i, (a, m_a))$ is created. Node $[S_k, S_k]$ is referred to as a *trivial node*. Notably, distinguishing between two identical states in a trivial node is unnecessary. Therefore, no outgoing transition leaves from $[S_k, S_k]$ in $G \times G$.
3. If $\lambda(S_i, (a, m_a)) = \lambda(S_j, (a, m_a))$ but $\delta(S_i, (a, m_a)) \neq \delta(S_j, (a, m_a))$, i.e., same output symbol but different next states, in G, a transition from $[S_i, S_j]$ to $[\delta(S_i, (a, m_a)), \delta(S_j, (a, m_a))]$ with label $(a, m_a)/\lambda(S_i, (a, m_a))$ is created. This finding suggests that the distinguishability of state-pair $[S_i, S_j]$ relies on the distinguishability of state-pair $[\delta(S_i, (a, m_a)), \delta(S_j, (a, m_a))]$.

Notably, the order of states in any pair is irrelevant, i.e., $[S_i, S_j] = [S_j, S_i]$, since the compound digraph ($G \times G$) is employed for deriving distinguishing sequences of all state-pairs. Furthermore, Huffman *et al.* [18–20] proposed a novel theory in which any two states S_i and S_j are distinguishable if and only if there exists a path from $[S_i, S_j]$ to the *Source* in $G \times G$; the input sequence of the path then becomes a legitimate distinguishing sequence of states S_i and S_j. Herein, distinguishing sequences of all state-pairs are derived by merely performing an inverse Breadth-First Search from the *Source* to all nodes in $G \times G$. The fact that the FSM in mind is minimized, i.e., any two states are distinguishable allows for all required distinguishing sequences to be discovered.

Once the compound digraph is constructed, target $\mathrm{SDS}((S_i, S_j), x)$ can be generated on the basis of the synchronization paradigm, similar as in the procedures in Section 4. The complete algorithm is specified as follows.

Algorithm: The Synchronizable Distinguishing Sequences Generation

INPUT. A k-port protocol FSM.

OUTPUT. The set $\{\mathrm{SDS}((S_i, S_j), z), c) \mid 1 \le i,\ j \le n,\ i \neq j,\ 1 \le z \le k\}$, where z denotes the desired input-RT identification and c denotes the associated cost of the SDS.

STEP 1. Construct the compound digraph of the protocol FSM.

STEP 2. Construct the state expansion digraph of the derived compound digraph.

STEP 3. Construct all shortest paths from all beginning APs to the *Source* by performing the inverse SSASP algorithm from the *Source*.

STEP 4. Group together all beginning APs attached to the same state-pair as well as their shortest paths.

STEP 5. Execute the first-round sequential search to pick up the shortest one for each group and to yield legitimate $(\mathrm{SDS}((S_i, S_j), z), c)$.

STEP 6. Execute the second-round sequential search to yield other $(\mathrm{SDS}((S_i, S_j), z), c)$.

The full set $\{(\mathrm{SDS}((S_i, S_j), z), c)\}$ is thus derived.

Complexity Analysis

First, since there are n states and p legitimate input symbols in the protocol FSM, there are a total of np transitions. Second, $O(n^2)$ nodes and $O(n^2p)$ transitions are in the compound digraph. Third, $O(2*(n^2p)+1) = O(n^2p)$ APs and $O((1+p)*(n^2p)) = O(n^2p^2)$ transitions are in the expanded compound digraph. Therefore, the algorithm thus requires a polynomial-bounded execution time of $O(((n^2p) + (n^2p^2))\log(n^2p)) = O((n^2p^2)\log(np))$.

In the following, Figure 5 illustrates the compound digraph ($G \times G$) for the FSM in Figure 2. Figure 6 depicts the state expansion digraph, all shortest paths are presented in Figure 7, and Table 3 summarizes the derived SDSs, in which the cost of an ES operation $\alpha = 4$ is given. Notably, it is possible to select more than one shortest path for each group in the first-round sequential search (Step 5) if the shortest paths are associated with the same cost. Moreover, in contrast to the SP prefixed to the TUT in a test case, the SDS is merely postfixed to the TUT. This accounts for why only beginning APs are considered during the generation of SDSs.

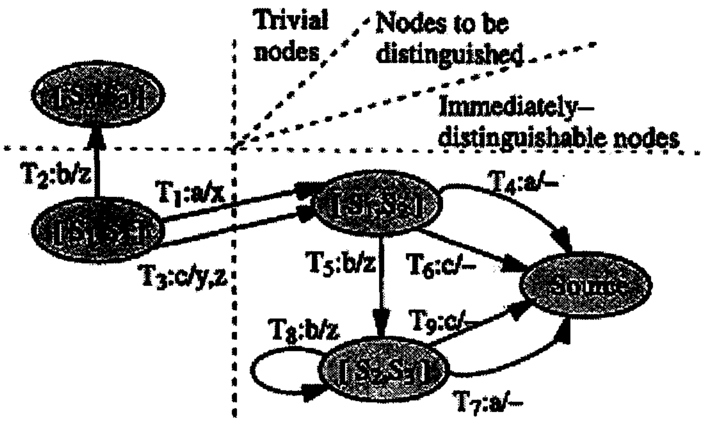

Figure 5. Compound digraph of the FSM in Figure 2.

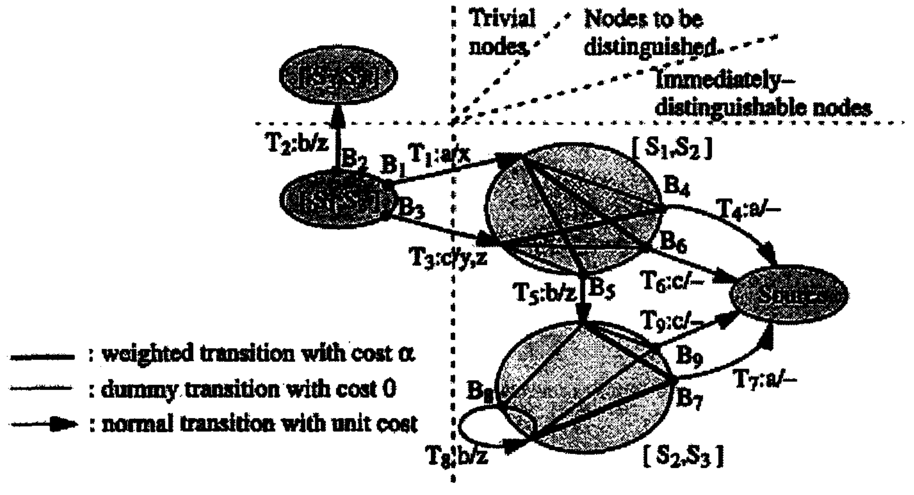
Trivial nodes
Nodes to be distinguished
Immediately–distinguishable nodes
T₂:b/z
B₂
B₁
T₁:a/x
B₃
T₃:c/y,z
[S₁,S₂]
B₄
T₄:a/–
B₆
T₅:b/z
B₅
T₆:c/–
Source
T₉:c/–
B₉
T₇:a/–
B₈
B₇
T₈:b/z
[S₂,S₃]
: weighted transition with cost α
: dummy transition with cost 0
: normal transition with unit cost

Figure 6. State expansion digraph of the digraph in Figure 5.

6. CONCLUSIONS

This paper has proposed a novel synchronization paradigm using the state expansion transformation which can be applied to generate several synchronizable, optimal sequences for protocol testing. In the paradigm, both the features of SS sequences and ES operations are seamlessly

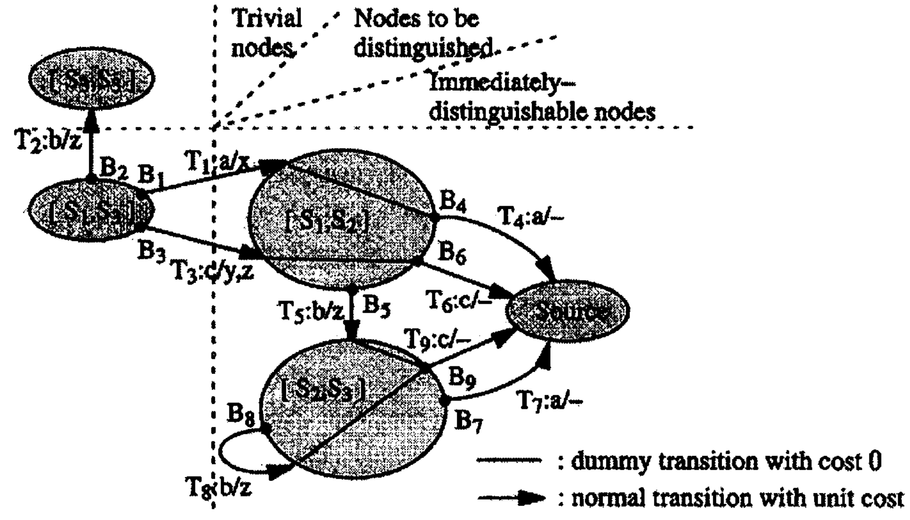
Trivial nodes
Nodes to be distinguished
Immediately-distinguishable nodes
[S1,S3]
[S1,S3]
[S1,S2]
[S2,S3]
Source
T2:b/z
T1:a/x
T3:c/y,z
T4:a/–
T5:b/z
T6:c/–
T7:a/–
T8:b/z
T9:c/–
B1
B2
B3
B4
B5
B6
B7
B8
B9
: dummy transition with cost 0
: normal transition with unit cost

Figure 7. Shortest paths of the digraph in Figure 6.

Table 3. Synchronizable distinguishing sequences.

Destination AP	Beginning State-Pair	Beginning AP	Cost	Input-RT Identification	Derived SDSs in First Round	Derived SDSs in Second Round
Source	(S_1, S_3)	B_1	2	1	$(\mathrm{SDS}((S_1,S_3),1),2)$	$(\mathrm{SDS}((S_1,S_3),2),6)$
Source	(S_1, S_3)	B_2	∞	2		
Source	(S_1, S_3)	B_3	2	3	$(\mathrm{SDS}((S_1,S_3),3),2)$	$(\mathrm{SDS}((S_1,S_3),2),6)$
Source	(S_1, S_2)	B_4	1	1	$(\mathrm{SDS}((S_1,S_2),1),1)$	
Source	(S_1, S_2)	B_5	2	2		$(\mathrm{SDS}((S_1,S_2),2),2)$
Source	(S_1, S_2)	B_6	1	3	$(\mathrm{SDS}((S_1,S_2),3),1)$	
Source	(S_2, S_3)	B_7	1	1	$(\mathrm{SDS}((S_2,S_3),1),1)$	
Source	(S_2, S_3)	B_8	2	2		$(\mathrm{SDS}((S_2,S_3),2),2)$
Source	(S_2, S_3)	B_9	1	3	$(\mathrm{SDS}((S_2,S_3),3),1)$	

combined by quantifying the execution difficulty of ES operations and explicitly augmenting them into the protocol FSM via the notion of the cost. Two typical sequences utilized in the W-method for protocol testing, i.e., the synchronizable preamble and the synchronizable distinguishing sequence, are thus derived on the basis of the proposed paradigm to demonstrate its viability. Finally, the paradigm is under the general multiparty configuration, thereby making it quite effective against the synchronization problem in a realistic test configuration for protocol testing.

REFERENCES

1. OSI Conformance Testing Methodology and Framework, Parts 1–5, ISO 9646.
2. K.K. Sabnani and A.T. Dahbura, A protocol test generation procedure, *Computer Networks and ISDN Systems* **15** (4), 285–297, (1988).
3. F.C. Hennie, *Finite-State Models for Logical Machines*, John Wiley & Sons, (1968).
4. B. Sarikaya and G.V. Bochmann, Synchronization and specification issues in protocol testing, *IEEE Transaction on Communication* **32**, 389–395, (1984).
5. W.H. Chen, C.S. Lu, L. Chen and J.T. Wang, Synchronizable protocol test sequence generation via the duplex technique, *IEEE INFOCOM*, 561–563, (1990).
6. H. Ural and Z. Wang, Synchronizable test sequence generation using UIO sequences, *Computer Communication* **16** (10), 653–661, (1993).
7. W.H. Chen and H. Ural, Synchronizable test sequences based on multiple UIO sequences, *IEEE/ACM Trans. on Networking* **3** (2), 152–157, (1995).
8. S.C. Boyd and H. Ural, The synchronization problem in protocol testing and its complexity, *Information Processing Letters* **40** (3), 131–136, (1991).

9. S. Guyot and H. Ural, Synchronizable checking sequences based on UIO sequences, *IWPTS '95*, 395–407, (1995).
10. G. Luo, R. Dssouli, G.V. Bochmann, P. Venkataram and A. Ghedamsi, Generating synchronizable test sequences based on finite state machine with distributed ports, *IWPTS '94*, 143–158, (1993).
11. Z. Kohavi, *Switching and Finite Automata Theory*, 2nd edition, McGraw-Hill, (1978).
12. M.P. Vasilevskii, Failure diagnosis of automata, *Kibernetika* **4**, 98–108, (1973).
13. T.S. Chow, Toward testing software design modeled by finite state machine, *IEEE Trans. on Software Engineering* **SE-4** (3), 178–187, (1978).
14. K.C. Tai and Y.C. Young, Port-synchronizable test sequences for communication protocols, *IWPTS '95*, 379–394, (1995).
15. E.W. Dijkstra, A note on two problems in connexion with graphs, *Numerische Mathematic* **1**, 269–271, (1959).
16. D.B. Johnson, Efficient algorithms for shortest paths in sparse networks, *Journal of the ACM* **24**, 1–13, (1977).
17. J.E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison Wesley, (1979).
18. D.A. Huffman, The synthesis of sequential switching circuits, *J. Franklin Institute* **257** (3/4), 161–190, 275–303, (1954).
19. E.F. Moore, *Automata Studies*, Princeton Univ. Press, (1956).
20. W.H. Chen and C.Y. Tang, Computing the optimal IO sequence of a protocol in polynomial time, *Information Processing Letters* **8** (40), 145–148, (1991).