Pergamon

Computers Math. Applic. Vol. 35, No. 5, pp. 117–124, 1998
Copyright©1998 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0898-1221/98 $19.00 + 0.00

PII: S0898-1221(98)00010-8

Efficient Minimization of Homogeneous FSMs for Fault Diagnosis

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

MARIA C. YUANG AND STEEN J. HSU
Department of Computer Science and Information Engineering
National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

(Received and accepted September 1996)

Abstract—On the base of the Finite State Machine (FSM) model, the fault diagnosis problem deals with the identification of a faulty implementation against its specification represented by an FSM. In particular, to efficiently identify potential faulty implementations caused by a transfer error in an FSM, cross-verification over the FSM is performed after all potential faulty FSMs have been minimized. Existing minimization algorithms become inefficient due to the lack of taking advantage of the homogeneity of these potential faulty FSMs. In this paper, we propose a two-phase algorithm for the efficient and simultaneous minimization of a set of homogeneous faulty FSMs. Disregarding the suspicious transition, the first phase of the algorithm performs minimization of these faulty FSMs via a common digraph of th FSMs. These faulty FSMs are considered minimized if the distinguishing sequences of all state-pairs can be discovered from the common digraph. Otherwise, the algorithm in the second phase performs minimization for each faulty FSM by individually restoring its corresponding unexpected transition in the reduced common digraph. To demonstrate the efficiency of the two-phase algorithm, we carried out experiments on a number of realistic protocol specifications, including the Alternation Bit Protocol (ABP), Transport Protocol Class 4 (TP4), and ISDN Basic Rate Interface (BRI) protocol. Experimental results show that the algorithm renders the minimization complexity greatly reduced for most of the realistic protocol FSMs.

Keywords—Minimization, Finite State Machine (FSM), Distinguishing sequences, Fault diagnosis.

1. INTRODUCTION

A formal specification model [1–3] provides an unambiguous means of modelling communication protocols defined as sets of rules governing the operations of communication networks. Among existing models, the Finite State Machine (FSM) model [4] has been widely and successfully used in areas such as protocol testing [5] and fault diagnosis [6,7]. Protocol testing ensures consistency between the implementation and the specification of the protocol. Substantially, fault diagnosis further identifies the location of faults against the specification of the protocol.

In the case of a fault occurring due to a transfer error (transition to an unexpected state), all potential faulty FSMs then correspond to FSMs with the unexpected transition progressing to states other than the expected state. Consequently, these FSMs are homogeneous and only differ in that particular unexpected transition. To identify a faulty implementation caused by such transfer error efficiently, cross-verification [7] over the FSMs is performed after all potential faulty FSMs have been minimized. The existing Hopcroft's algorithm [8], known as one of the most promising minimization algorithms, however, becomes inefficient for minimizing a group of homogeneous FSMs due to the lack of taking advantage of the homogeneity of these faulty FSMs.

In this paper, we propose a two-phase algorithm for the efficient and simultaneous minimization of a set of homogeneous faulty FSMs. In the first phase of the algorithm, the original FSM excluding the suspicious transition is first transformed into a common digraph exhibiting the behavior of all state-pairs for all faulty FSMs. These faulty FSMs are considered minimized if the distinguishing sequences of all state-pairs can be discovered from the common digraph. Otherwise, the second phase of the algorithm takes over the task. The algorithm first performs the reduction of the common digraph by means of four reduction rules. By restoring the corresponding unexpected transition in the reduced common digraph for each faulty FSM one at a time, the algorithm again performs minimization by means of finding distinguishing sequences. To demonstrate the efficiency of the two-phase algorithm, we accomplished experiments on a number of realistic protocol specifications, including the Alternating Bit Protocol (ABP), Transport Protocol Class 4 (TP4), and ISDN Basic Rate Interface (BRI) protocol. Experimental results showed that, in most protocol FSMs, the algorithm entails lower complexity for the minimization of the homogeneous faulty FSMs.

The paper is organized as follows. Section 2 defines the FSM model and the minimization problem. Section 3 then presents the two-phase algorithm. The complexity analysis and experimental results are also given in this section. Finally, Section 4 concludes the paper.

2. PRELIMINARIES—FSM MODEL AND PROBLEM DEFINITION

A protocol can be modelled as a six-tuple FSM: $(\mathbf{S}, S_1, \mathbf{I}, \mathbf{O}, \delta, \lambda)$. $\mathbf{S}$ is a nonempty set of states, S_1 is a designated state called the *initial state*, and $\mathbf{I}$ and $\mathbf{O}$ are nonempty sets of input and output symbols, respectively. The *next-state function* (δ) is defined as $\delta : \mathbf{S} \times \mathbf{I} \to \mathbf{S}$, and $\delta(S_i, a) = S_j$ denotes a transition from state S_i to state S_j as a result of given an input, a. Similarly, the *output function* (λ) is defined as $\lambda : \mathbf{S} \times \mathbf{I} \to \mathbf{O}$, and $\lambda(S_i, a) = b$ denotes the creation of output b being at state S_i if input a is applied. These two functions are combined and represented by the *transition with a label*: $S_i \to a/b \to S_j$. Graphically, an FSM can also be represented by a digraph $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \mathbf{S}$, and each edge in $\mathbf{E}$ corresponds to a transition with a label. In this paper, we assume FSMs are deterministic, strongly-connected, and completely-specified [4]. An example of an FSM is shown in Figure 1. The example will be used for the illustration of the algorithm throughout the rest of the paper.

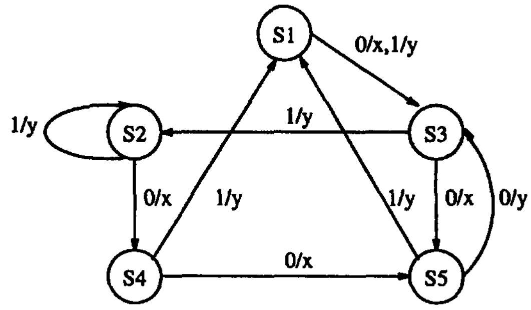

Figure 1. An FSM (G).

An FSM is minimized if and only if any two states in the FSM are always *distinguishable*. States S_i and S_j are said to be distinguishable if there exists an input sequence such that, upon applying the input sequence, states S_i and S_j produce different output sequences. For example, as shown in Figure 1, states S_2 and S_3 are distinguishable since they produce different output sequences xx and xy after applying the input sequence, 00. Input sequence 00 is thus referred to as the *distinguishing sequence* of states S_2 and S_3.

If a *transfer error* (i.e., faulty next-state) occurs in transition $S_i \rightarrow a/b \rightarrow S_j$ of a minimized n-state FSM, named as G, there are $n-1$ possible faulty FSMs (G_k, $k \neq j$) resulting from $n-1$ possible faulty next-states (S_k, $k \neq j$), respectively. This set of relevant faulty FSMs $(G_1, G_2, \dots, G_{j-1}, G_{j+1}, \dots, G_n)$ are defined as *homogeneous faulty FSMs*. As a result, it is inefficient to minimize these homogeneous FSMs one at a time using traditional minimization approaches [8–10]. Thus, the main goal of this paper is to present a novel and efficient algorithm, named as the two-phase minimization algorithm, which performs simultaneous minimization of a set of homogeneous FSMs.

3. TWO-PHASE MINIMIZATION ALGORITHM

In the following subsections, we first introduce a minimization method by means of a compound digraph. We then propose the two-phase algorithm in detail based on the constructed compound digraph.

3.1. Minimization Using Compound Digraph

A *compound digraph* ($G \times G$) of an FSM (G) is defined and constructed as follows. The input and output sets 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 node $[S_i, S_j]$ ($S_i \neq S_j$) and input a in $G \times G$, the outgoing edge of the node is determined as follows.

1. If $\lambda(S_i, a) \neq \lambda(S_j, a)$ in G, i.e., S_i and S_j are *immediately distinguishable* by input a, an edge from $[S_i, S_j]$ to the *Source* with label $a/-$ is created.
2. If $\lambda(S_i, a) = \lambda(S_j, a) = b$ and $\delta(S_i, a) = \delta(S_j, a) = S_k$ in G, an edge from $[S_i, S_j]$ to $[S_k, S_k]$ with label a/b is created. Node $[S_k, S_k]$ is referred to as a *trivial node*. Notice that it is not necessary to distinguish two identical states in a trivial node. Therefore, there is no outgoing edge from $[S_k, S_k]$ in $G \times G$.
3. If $\lambda(S_i, a) = \lambda(S_j, a) = b$, but $\delta(S_i, a) \neq \delta(S_j, a)$ in G, an edge from $[S_i, S_j]$ to $[\delta(S_i, a), \delta(S_j, a)]$ with label a/b is created. This implies the distinguishability of state-pair $[S_i, S_j]$ is dependent on the distinguishability of state-pair $[\delta(S_i, a), \delta(S_j, a)]$.

According to these three rules, the compound digraph ($G \times G$) for G in Figure 1 is constructed and shown in Figure 2. There are $(5 * 4)/2 = 10$ main nodes (state-pairs), three trivial nodes, and the *Source* node in $G \times G$. Since $\lambda(S_4, 0) \neq \lambda(S_5, 0)$, $[S_4, S_5]$ is connected to the *Source* through an edge with label $0/-$. Since $\lambda(S_2, 1) = \lambda(S_3, 1) = y$ and $\delta(S_2, 1) = \delta(S_3, 1) = S_2$ in G, $[S_2, S_3]$ is connected to trivial node $[S_2, S_2]$ through an edge with label $1/y$. In addition, since $\lambda(S_2, 0) = \lambda(S_3, 0) = x$, $\delta(S_2, 0) = S_4$, and $\delta(S_3, 0) = S_5$, an edge from $[S_2, S_3]$ to $[S_4, S_5]$ with label $0/x$ is then constructed. Notice that since the compound digraph ($G \times G$) is employed for deriving the distinguishing sequences of all state-pairs, the order of states in any pair is irrelevant, i.e., $[S_i, S_j] = [S_j, S_i]$.

Having constructed the compound digraph, $G \times G$, we are now at the stage of deriving distinguishing sequences of state-pairs. We here employ the approaches proposed by Huffman and Chen [9–11], which can be reworded in our case to say that 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 $G \times G$, and the input sequence of the path then becomes a legitimate distinguishing sequence of states S_i and S_j. To minimize the FSM (G), we perform an inverse Breadth-First Search (BFS) from the *Source* to all nodes in $G \times G$. If all nontrivial nodes are reachable, the FSM is minimized. In this case, all state-pairs are distinguishable by the distinguishing sequences established from the corresponding reachability path. Otherwise, there exists a node ($[S_i, S_j]$) which is not reachable (i.e., S_i and S_j are not distinguishable), S_i and S_j are thus considered to be *equivalent* and should be merged into one state. This minimization algorithm requires $O(pn^2)$ time complexity for $G \times G$ with $O(n^2)$ nodes and $O(pn^2)$ edges, where p and n are the numbers of input symbols and states of the FSM (G), respectively. Although this algorithm incurs higher complexity compared to

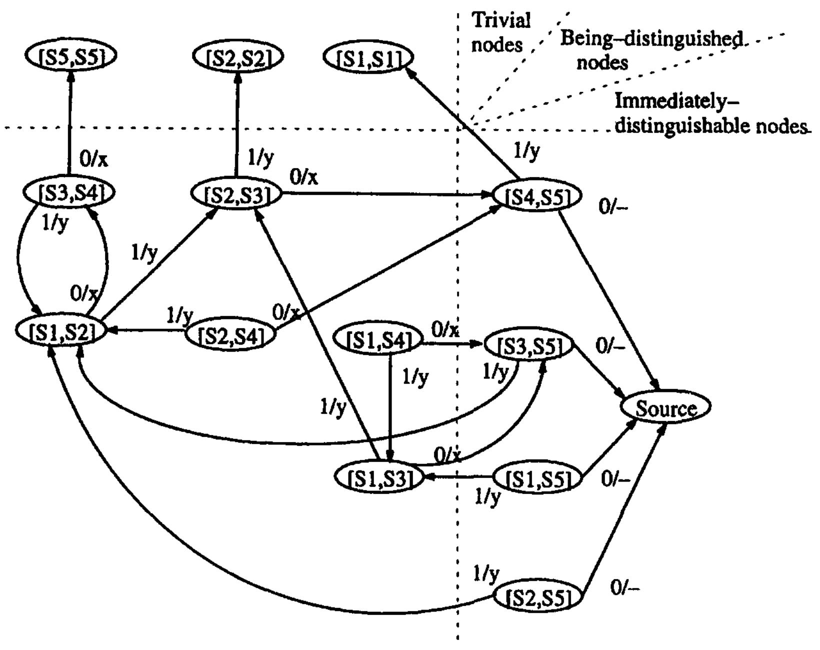

Figure 2. Compound digraph $G \times G$.

the Hopcroft's $O(pn \log n)$ algorithm [8], known as the most efficient minimization algorithm, the method of employing the compound digraph, as will be shown, is more effective for the minimization of homogeneous FSMs.

3.2. Two-Phase Algorithm—The First Phase

Assume that there exists an edge $[S_i, S_x] \to a/b \to [S_j, S_y]$, in $G \times G$, i.e., $S_i \to a/b \to S_j$ and $S_x \to a/b \to S_y$ in G. If a transfer error occurs, which results in the changes from $S_i \to a/b \to S_j$ to $S_i \to a/b \to S_k$, where $1 \leq k \leq n$ and $k \neq j$, corresponding to $n-1$ faulty FSMs G_k ($1 \leq k \leq n$, $k \neq j$), edge $[S_i, S_x] \to a/b \to [S_j, S_y]$ in $G \times G$ would be accordingly altered as $[S_i, S_x] \to a/b \to [S_k, S_y]$ in $G_k \times G_k$. For any faulty FSM, say G_k, since there are $n-1$ distinct nodes $[S_i, S_x]$ ($1 \leq x \leq n$, $x \neq i$) in $G_k \times G_k$, a set of $n-1$ edges in $G_k \times G_k$, denoted as E_k and called *modified edge set*, is different from that in $G \times G$. Consequently, removing the modified edge set from each faulty FSM, the remaining digraphs, i.e., $G_k \times G_k - E_k$, for all k, where $1 \leq k \leq n$ and $k \neq j$, become identical. The identical digraph is referred to as the *common digraph*, denoted as G_c.

From the common digraph, the distinguishing sequences of all state-pairs for any faulty FSM can be derived by applying the inverse BFS from the *Source*. If the distinguishing sequences of all state-pairs can be derived, all $n-1$ faulty FSMs are minimized and the problem is thus solved. Otherwise, the second phase of the algorithm takes over the task. Notice that, although the behavior of state-pairs related to the unexpected edge is temporarily ignored in the common digraph, any edge in modified edge set E_k progressing to the *Source*, say $[S_i, S_z] \to a/- \to$ *Source*, should stay in the common digraph, since states S_i and S_z can still be distinguished immediately by input a in any faulty FSM regardless of the occurrence of any transfer error.

The minimization of homogeneous FSMs derived from the FSM example given in Figure 1 is illustrated as follows. Consider a transfer error occurs in edge $S_3 \to 0/x \to S_5$, i.e., $S_3 \to 0/x \to S_k$, where $k \neq 5$. Then, the transfer error results in the creation of four homogeneous faulty FSMs, denoted as G_1, G_2, G_3, and G_4. For example, Figure 3 depicts the compound digraph ($G_4 \times G_4$) with modified edge set E_4 shown in bold. In addition, the common digraph for all faulty FSMs can be constructed and shown in Figure 4. To minimize G_1 through G_4, an inverse BFS from the *Source* can be performed over the common digraph. In this example,

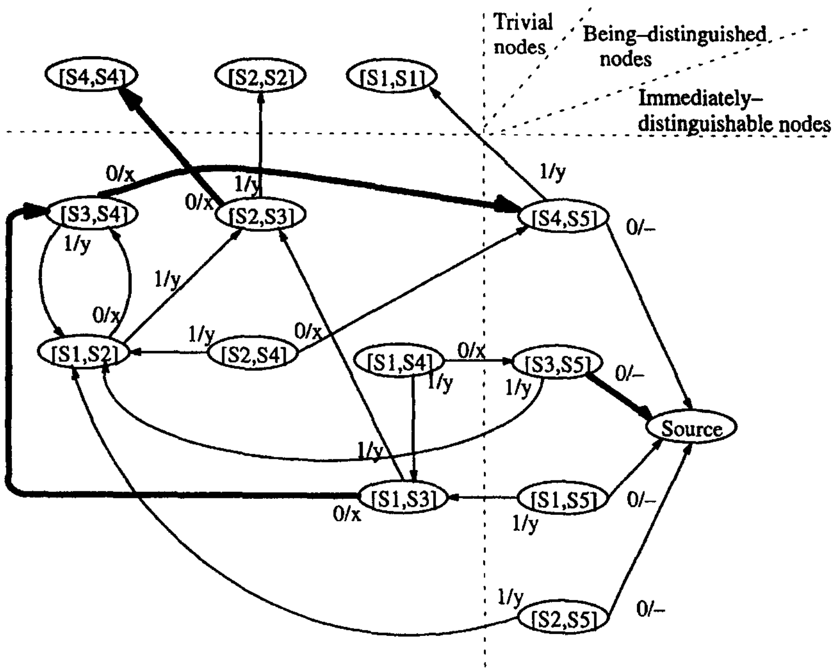

Figure 3. Compound digraph $G_4 \times G_4$.

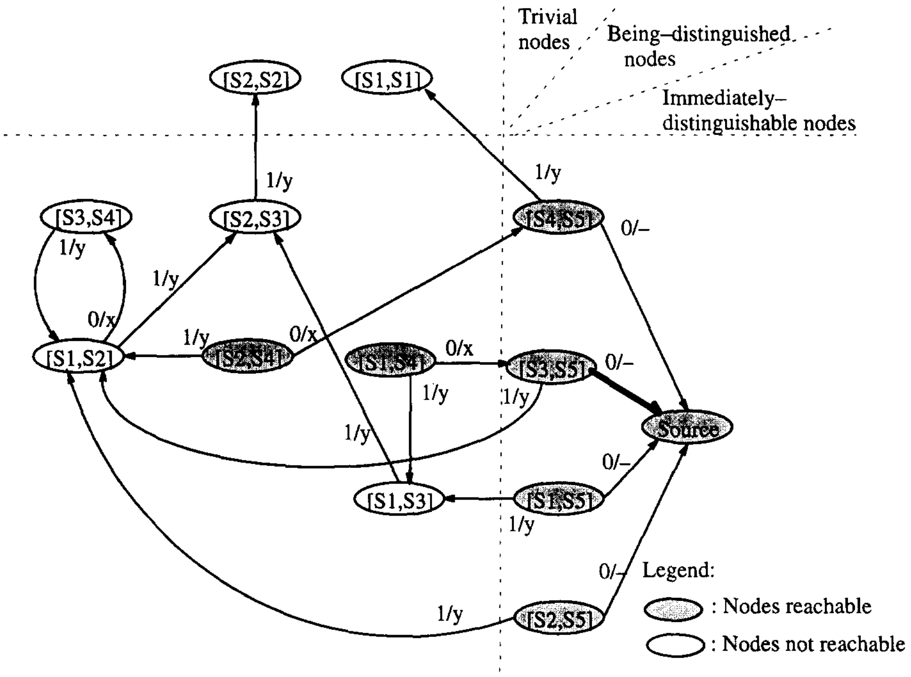

Figure 4. Common digraph G_c.

nodes $[S_2, S_4]$, $[S_4, S_5]$, $[S_1, S_4]$, $[S_3, S_5]$, $[S_1, S_5]$, and $[S_2, S_5]$ are reachable, their distinguishing sequences are thus generated and can be unanimously applied to all faulty FSMs. Since there exist nonreachable state-pairs, the minimization process is taken over by the second phase of the algorithm.

3.3. Two-Phase Algorithm—The Second Phase

To efficiently accomplish the task, we first reduce the common digraph (G_c) by means of four reduction rules (R-rules) described as follows.

- **R-rule 1:** All reachable nodes and traversed edges in the common digraph are merged and replaced by a *black node*. All the state-pairs in the black node are from now on neglected due to their distinguishability determined in the first phase of the algorithm.
- **R-rule 2:** All outgoing edges from the black node are removed. This is because we are only concerned about the reachability from other state-pairs to the black node, but not the opposite.
- **R-rule 3:** All trivial nodes and their incoming edges are removed. This is because they produce no effect on reachability due to the lack of outgoing edges from them.
- **R-rule 4:** The common digraph is further reduced to a condensed digraph [12] in which each strongly-connected component is replaced by a node, and all edges from a strongly-connected component to another are replaced by a single edge.

For example, the common digraph (G_c) in Figure 4 can be reduced to G'_c, as shown in Figure 5, based on these four R-rules.

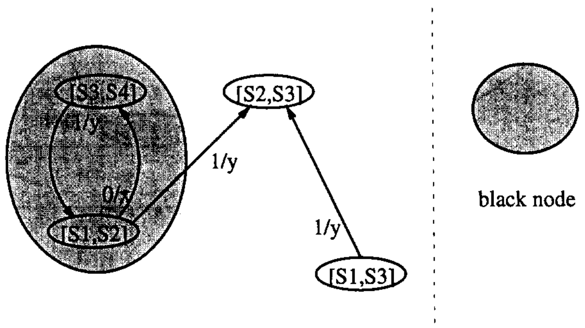

Figure 5. Reduced common digraph G'_c.

After having reduced the common digraph, the second phase of the algorithm deals with the individual minimization of each faulty FSM with the unexpected edge restored. First, for each faulty FSM, the corresponding modified edge set of the unexpected edge is restored to the reduced common digraph. Second, for each restored reduced common digraph, an inverse BFS from the black node is performed. The faulty FSM is minimized if the distinguishing sequences of all remaining state-pairs can be discovered. Otherwise, the states in any nonreachable node are not distinguishable and can be merged.

For the example given in Figure 1, after restoring modified edge set E_4 to reduced common digraph G'_c, its digraph G'_4 is constructed and depicted in Figure 6. Based on the second phase of the algorithm, we learn that the faulty FSM, G_4, is not minimized. This is because state-pair $[S_2, S_3]$ in G'_4 is not reachable and thus not distinguishable.

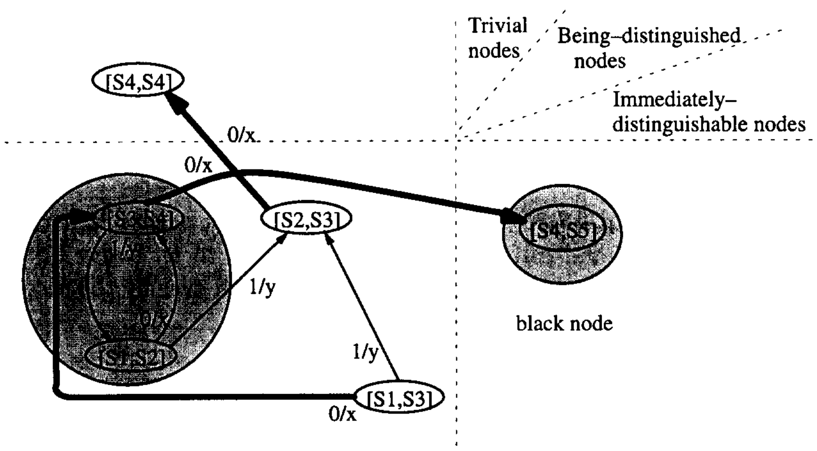

Figure 6. Digraph G'_4 after restoring E_4.

3.4. Formal Description of the Two-Phase Algorithm

ALGORITHM: The Two-Phase Algorithm

INPUT: A deterministic, strongly-connected, completely-specified, and minimized FSM (G), and a suspicious edge, say $S_i \rightarrow a/b \rightarrow S_j$.

OUTPUT: $n-1$ minimized faulty FSMs, i.e., $G_1, G_2, \ldots, G_{j-1}, G_{j+1}, \ldots, G_n$, corresponding to $n-1$ possible transferred states, respectively.

```
First phase

    Step 1: Construct common digraph $G_c$.
    Step 2: Execute an inverse BFS from the Source;
            If (all nontrivial nodes are reachable)
                then (all faulty FSMs are minimized and the algorithm terminates).

Second phase

    Step 3: Construct reduced common digraph $G'_c$ by applying R-rules 1–4.
    Step 4: For (each possible next-state $S_k$ ($S_k \neq S_j$) of the suspicious edge) do
              1. Restore modified edge set $E_k$ to construct $G'_k$;
              2. Execute an inverse BFS over $G'_k$ from the black node;
              3. If (all nontrivial nodes are reachable) then (faulty FSM $G_k$ is minimized)
                 else {*$G_k$ is not minimized*}
                 the states in each nonreachable node are equivalent and can be merged.
                 {*$G_k$ is thus minimized*}
```

The time complexity of the two-phase algorithm is analyzed as follows. Since there are $O(n^2)$ nodes and $O(pn^2)$ edges in $G_k \times G_k$ and $n-1$ edges in E_k, Step 1 thus requires $O(pn^2)$ time. Step 2 also requires $O(pn^2)$ time, owing to the fact that the execution time of the BFS is only dependent on the size of the target digraph. Therefore, if the algorithm can be successfully terminated in the first phase, the aggregate time complexity is only $O(pn^2)$. Step 3 performs four R-rules, where R-rule 1 can be executed implicitly in Step 2; R-rules 2 and 3 require $O(pn^2)$ time since there are a maximum of $O(pn^2)$ edges to be examined if they are outgoing from the black node or incoming to the $O(n)$ trivial nodes; and R-rule 4 can be executed using any existing algorithm for the construction of strongly-connected components [12]. Consequently, Step 3 thus requires a total of $O(pn^2)$ time. Step 4 consists of three substeps in the For-loop. There are $n-1$ next-states in the outer For-loop, a maximum of $n-1$ edges restored in Step 4.1, a maximum of the number of nodes and edges in G'_c plus $n-1$ restored edges traversed in Step 4.2, and a maximum of the number of nodes in G'_c nonreachable. Step 4 thus requires $O(n) * (O(n) + (|G'_c| + O(n)) + |G'_c|) = O(n^2 + n|G'_c|)$, where $|G'_c|$ denotes the size of the reduced common digraph, G'_c. The aggregate time complexity for the second phase becomes $O(pn^2 + n^2 + n|G'_c|) = O(pn^2 + n|G'_c|)$. Consequently, $|G'_c|$ determines the efficiency of the second phase of the algorithm.

Compared to the Hopcroft's $O(pn^2 \log n)$ algorithm ($n-1$ faulty FSMs and $O(pn \log n)$ for each one), the method is efficient if either the algorithm terminated in the first phase or $|G'_c|$ is less than $O(pn \log n)$. In other words, if the second phase of the algorithm is required, it becomes significant should the *reduction target* be smaller than $O(pn \log n)/O(pn^2) = O(\log n/n)$, where $O(pn^2)$ is the size of the common digraph, G_c. To justify this, we carried out experiments on a number of existing protocol specifications, including Alternating Bit Protocol [5], NBS Transport Protocol Class 4 [13], and ISDN Basic Rate Interface Protocol [14]. The experimental results, listed in Table 1, show that, in most realistic protocol FSMs, the algorithm can be successfully terminated in the first phase with all homogeneous FSMs minimized. It thus requires only $O(pn^2)$ time complexity. For other FSMs, such as the Alternating Bit Protocol as shown in Table 1, the reduction of common digraphs is significant entailing low complexity in the remaining minimization process.

Table 1. Minimization of realistic protocols.

Property \ Example		FSM 1 [6]	FSM 2 [7]	FSM 3 [15]	NBS TP4 [13]	ISDN BRI [14]	ABP [5]
Original Digraph	Node #	4	4	6	16	8	8
	Input #	3	3	2	26	14	5
	Edge #	12	12	12	53	31	10
Common Digraph	Node #	10	10	22	137	37	37
	Edge #	18	18	30	3120	392	140
Reduced Common Digraph	Node #	1	1	1	1	1	2
	Edge #	0	0	0	0	0	1
Reduction Target	Node #	10%	10%	4.5%	0.7%	2.7%	5.4%
	Edge #	0	0	0	0	0	0.7%
Algorithm terminated in the first phase with time complexity $O(pn^2)$		Yes	Yes	Yes	Yes	Yes	No

4. CONCLUSIONS

In this paper, we have proposed a two-phase algorithm for the efficient and simultaneous minimization of a set of homogeneous faulty FSMs. The first phase of the algorithm performs minimization via a common digraph of these faulty FSMs. The algorithm terminates if the distinguishing sequences of all state-pairs can be discovered from the common digraph. Otherwise, the second phase performs minimization for each faulty FSM by individually restoring its corresponding unexpected transition in the reduced common digraph. To demonstrate the efficiency of the algorithm, we performed experiments on a number of realistic protocol specifications. Experimental results showed that, in most protocol FSMs, the algorithm entails lower complexity for the minimization of the homogeneous faulty FSMs.

REFERENCES

1. Estelle: A formal description technique based on an extended state transition model, ISO/IEC 9074, (1989).
2. LOTOS—A formal description technique based on the temporal ordering of observational behavior, ISO/IEC 8807, (1988).
3. CCITT specification and description language (SDL) recommendation Z.100, CCITT SG X, (1992).
4. F.C. Hennie, *Finite-State Models for Logical Machines*, John Wiley & Sons, (1968).
5. K.K. Sabnani and A.T. Dahbura, A protocol test generation procedure, *Computer Networks and ISDN Systems* **15** (4), 285–297 (1988).
6. A. Ghedamsi, G.V. Bochmann and R. Dssouli, Multiple fault diagnostics for finite state machines, *IEEE INFOCOM*, 782–791 (1993).
7. D. Lee and K. Sabnani, Reverse-engineering of communication protocols, *IEEE/ACM ICNP*, 208–216 (1993).
8. J.E. Hopcroft, An $n \log n$ algorithm for minimizing states in a finite automation, *Theory of Machines and Computations*, (Edited by Z. Kohavi), pp. 189–196, Academic Press, (1971).
9. D.A. Huffman, The synthesis of sequential switching circuits, *J. Franklin Institute* **257**, 3–4,161–190,275–303 (1954).
10. E.F. Moore, Gedanken experiments on sequential machines, *Automata Studies*, pp. 129–153, Princeton Univ. Press, (1956).
11. 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).
12. R.E. Tarjan, Depth first search and linear graph algorithms, *SIAM Journal of Computing* **1**, 146–160 (1972).
13. D. Sidhu and T. Leung, Formal methods for protocol testing: A detail study, *IEEE Trans. Software Engineering* **15** (4), 413–426 (1989).
14. J. Zhu and S.T. Chanson, Towards evaluating fault coverage of protocol test sequences, IFIP PSTV, (1994).
15. H. Ural and K. Zhu, Optimal length test sequence generation using distinguishing sequences, *IEEE Trans. on Networking* **1** (3), 358–371 (1993).