Timed I/O Test Sequences for Discrete Event Model Verification

Ki Jung Hong and Tag Gon Kim

Dept. of EECS,
Korea Advanced Institute of Science and Technology (KAIST),
373-1 Guseong-dong, Yuseong-gu, Daejeon, Republic of Korea
kjhong@smslab.kaist.ac.kr, tkim@ee.kaist.ac.kr

Abstract. Model verification examines the correctness of a model implementation with respect to a model specification. While being described from model specification, implementation prepares to execute or evaluate a simulation model by a computer program. Viewing model verification as a program test this paper proposes a method for generation of test sequences that completely covers all possible behavior in specification at an I/O level. Timed State Reachability Graph (TSRG) is proposed as a means of model specification. Graph theoretical analysis of TSRG has generated a test set of timed I/O event sequences, which guarantees 100% test coverage of an implementation under test.

1 Introduction

Model verification examines the correctness of a model implementation with respect to a model specification. As discrete event simulation models are getting more and more complicated verification of such models is extremely complex. Thus, automatic verification of such a simulation model is highly desirable [1].

Since a model specification is implemented in a simulation program model verification can be viewed as a program test. Thus, model verification starts from generation of input/output sequences for an implementation, which covers all possible behaviors of a specified model. Untimed discrete event model can be specified by finite state machine (FSM). FSM can be verified by conformance test [5]. Test sequences of conformance test can be built by the UIO method [3, 4], and others. Test sequences generation of timed discrete event models can be obtained by timed Wp-method [6], which is based on timed automata and region automata.

This paper proposes a new approach to select test cases for a module-based testing of a discrete event simulation program at an I/O level. We assume that specification of each module is known and an implementation is unknown as a black box. Time State Rechability Graph (TSRG) is proposed to specify modules of a discrete event model. TSRG represents a discrete event model in terms of nodes and edges. Each node represents a state of discrete event model associated with which is a time interval. On the other hand, each edge represents transition between nodes with input, output or null event in the specified time

T.G. Kim (Ed.): AIS 2004, LNAI 3397, pp. 275–284, 2005.
© Springer-Verlag Berlin Heidelberg 2005

interval. Graph theoretical analysis of TSRG generates all possible timed I/O sequences from which a test set of timed I/O sequences with 100 % coverage can be constructed.

An obstacle of the test method lies in different numbers of states between specification and implementation. This is because we assume that an exact number of states in implementation is unknown. However, an assumption on a maximum number of states used in an implementation can overcome the obstacle. Note that the number does not need to be exact. Instead, it may be any number that is greater than or equal to one used in implementation, which only determines complexity of testing. This paper is organized as follows. Section 2 proposes TSRG. Section 3 introduces TSRG related definitions and theorems, and proposes the generation method of test input/output sequences. Finally, conclusions are made in section 5.

2 Timed State Reachability Graph

Timed State Reachability Graph (TSRG) is a modeling means which specifies a discrete event system in timed input/output behavior. Nodes represent time constraints states; edges represent conditions for transitions between nodes. The graph starts from an initial state of a discrete event model and generates edges/nodes which are reachable from the state. The formal semantics of TSRG is given below:

$$
\begin{aligned}
TSRG &= \langle N, E, \theta_N, \theta_E \rangle \\
N &: \text{States set : Node} \\
E &: N \times N : \text{Edge} \\
\theta_N &: N \rightarrow \Re^{+}_{(0,\infty)} \times \Re^{+}_{(0,\infty)} \\
&: \text{ Node attribute function} \\
\theta_E &: E \rightarrow (X \cup Y \cup \{\tau\}) \times Boolean \\
&: \text{ Edge attribute function} \\
X &: \text{Input events set} \\
Y &: \text{Output events set} \\
\tau &: \text{null output event} \\
Boolean &\in \{true, false\} : \text{CONTINUE or NOT}
\end{aligned}
$$

TSRG has two types of an edge: input transition edge and output transition edge. While one or none output transition edge can be attached at a node, one or more input transition edges can be attached at the same node. An attribute associated with a node is interpreted as a waiting time for the state to be transit. An edge attribute include a Boolean of **CONTINUE**. Meaning of **CONTINUE** is that (1) an input transition occurs at a state before a deadline of a maximum elapsed time defined at the state, and (2) a new state continues keeping the deadline defined at the previous state.

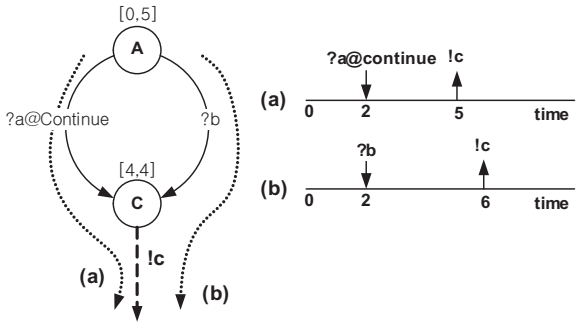


Fig. 1. Event time diagram of continue example.

Let us explain **CONTINUE** in more detail using an example shown in Fig 1. In the Fig, there are two paths from state A to state C. The first path is A $\xrightarrow{?a@_{[0,5]},\textbf{continue}}$ C, and the second one is A $\xrightarrow{?b@_{[0,5]}}$ C. A waiting time of state A is the same value 5 for both paths, but a waiting time of state C is different for each path, because of **continue** in the first path. If ε is an elapsed time for an arrival event ?a of state A, a waiting time of state C for the first path is 5-ε, $0 < \varepsilon \leq 5$, and a waiting time of state C for the second path is 4. Briefly, **CONTINUE** means that the waiting timer is continued from the previous state, without resetting for the next state. Due to such timer semantics of **CONTINUE**, a pair of nodes connected by an edge with **CONTINUE** should have a finite waiting time and an output edge.

3 Theorem and Definition: TSRG

The following equivalent node's definition in TSRG is prerequisite to define the minimization of TSRG.

Definition 1 (Equivalent node). *Let $s_1, s_2 \in N$ be nodes. Node s_1 and s_2 are equivalent, i.e., $s_1 \equiv s_2$, when the following condition is satisfied:* $\theta_N(s_1) = \theta_N(s_2) \wedge (\forall e_1 = (s_1, \acute{s_1}) \in E, \exists e_2 = (s_2, \acute{s_2}) \in E, \theta_E(e_1) = \theta_E(e_2) \wedge \acute{s_1} \equiv \acute{s_2}) \wedge (\forall e_2 = (s_2, \acute{s_2}) \in E, \exists e_1 = (s_1, \acute{s_1}) \in E, \theta_E(e_1) = \theta_E(e_2) \wedge \acute{s_1} \equiv \acute{s_2})$.

Definition 2 (Minimization of TSRG). *TSRG is minimized if and only if there is no equivalent relation for any two nodes in the node set.*

Figure 2 shows an example of equivalent nodes. Timed input/output (TIO) event trace of model (a) is repetitive sequences of ?a@$[0,t_A]$·!b@$[t_B,t_B]$·?c@$[0,t_C]$, which is extracted from a state trace with a TIO event A$\xrightarrow{?a@_{[0,t_A]}}$B $\xrightarrow{!b@_{[t_B,t_B]}}$C $\xrightarrow{?c@_{[0,t_C]}}$A. The TIO event trace of model (b) is repetitive sequences of ?a@$[0,t_A]$· !b@$[t_B,t_B]$· ?c@$[0,t_C]$, extracted from A$\xrightarrow{?a@_{[0,t_A]}}$B $\xrightarrow{!b@_{[t_B,t_B]}}$C $\xrightarrow{?c@_{[0,t_C]}}$D $\xrightarrow{?a@_{[0,t_A]}}$B. Thus, model (a) and (b) in figure 2 have the same TIO event trace.

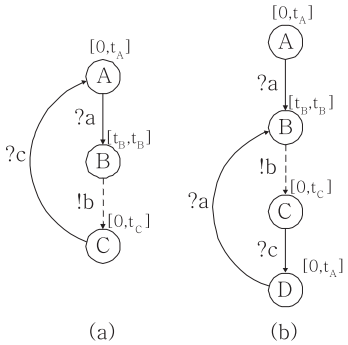


Fig. 2. Equivalent states.

While model (a) is minimized, model (b) has equivalent nodes, A ≡ D. In addition, the number of nodes of each model is different.

Test sequences for TSRG visit all edges in a TSRG model through a path from a start node to an end node. Such a path is defined as:

Definition 3 (Path). *Let $i, n \in Z$ be integers with $i < n$, $s_i \in N$ be a node, $e_i \in E$ be an edge with $e_i = (s_i, , s_{i+1})$, and $t_i = \theta_N(s_i)$ be a node attribute function. A path P from s_0 to s_n expressed as $P(s_0, s_n) = (e_0, t_0)(e_1, t_1) \cdots (e_{n-1}, t_{n-1})$, which is the sequence of all pairs of a visited edge e_i and its waiting time t_i from the start node to the end node.*

Each small paths can concatenate to the big one. Path concatenation operator is defined formally to describe such concatenation behavior.

Definition 4 (Path concatenation operator). *Let $s_i, s_j, s_k \in N$ be nodes and $P(s_i, s_j), P(s_j, s_k)$ be paths. Path concatenation operator $\bullet$ is defined as $P(s_i, s_j) \bullet P(s_j, s_k) = P(s_i, s_k)$ with the following properties.*

$$P(s_j, s_k) \bullet P(s_i, s_j) = \phi$$
$$P(s_i, s_j) \bullet \phi = P(s_i, s_j)$$
$$\phi \bullet P(s_i, s_j) = P(s_i, s_j)$$

For full coverage of states and transitions in model verification all states and edges in TSRG should be reachable, or strongly connected, defined in the following.

Definition 5 (Reachable). *TSRG is reachable if and only if there exists one or more path between any two nodes in TSRG.*

If TSRG is reachable, it is possible to visit all edges and nodes to verify TSRG. Otherwise, TSRG is not verifiable. The visit of all nodes can be covered through the visit of all edges in reachable TSRG. Thus, a loop is introduced for the visit of all edge in reachable TSRG. If TSRG is a strongly connected graph, there exist one or more loop paths which can visit any edge and node in the

graph. There exists a set of loop paths which can visit all edges and nodes in the graph. If there exists an intersection between two loops an adjacency set to be defined contains the intersection relation between loops. A traversal from an initial state to any other state can be contained by some interconnected loops. Interconnected loops can be bound by an adjacent loop chain.

Theorem 1 (Loop). *Let $TSRG$ be reachable. Any node in $TSRG$ can be traversed by one or more loop paths.*

Proof. $\forall s_i, s_j \in N, \exists P(s_i, s_j)$ *and* $P(s_j, s_i)$ *s.t.* $P(s_i, s_i) = P(s_i, s_j) \bullet P(s_j, s_i)$

Definition 6 (Loop set). *In a reachable TSRG, a loop set is a collections of loops which cover all edges in TSRG.*

All edges of TSRG is covered from the definition of a loop set. If a test target has an equivalent node, visiting routes of all possible edges are made from the loop set of TSRG by the following loop joining concatenation.

Definition 7 (Adjacency set). *Any two loops in a loop set L has adjacent relation if and only if these loops visit the same node. An adjacent relation set, called an adjacency set, has the condition: $M_{adj} \subseteq L \times L$.*

Definition 8 (Loop joining concatenation operator). *Let L be a loop set, M_{adj} be an adjacency set, p_1, p_2 be paths in TSRG, s_1, s_2, s_i be in N, s_i be the first met adjacent node, and $l_1, l_2 \in L$ be loop paths. Then, the following operators are hold.*

$$
\begin{aligned}
& l_1 \bullet \phi = l_1, \phi \text{ is identity} \\
& \phi \bullet l_1 = l_1 \\
& l_1 \bullet l_2 = \phi, \text{ if } (l_1, l_2) \notin M_{adj} \\
& l_1 \bullet l_2 = l_3, \text{ if } (l_1, l_2) \in M_{adj} \\
& \quad l_1 = p_1(s_1, s_i) \bullet p_1(s_i, s_1) \\
& \quad l_2 = p_2(s_2, s_i) \bullet p_2(s_i, s_2) \\
& \quad l_3 = p_1(s_1, s_i) \bullet p_2(s_i, s_2) \bullet p_2(s_2, s_i) \bullet p_1(s_i, s_1)
\end{aligned}
$$

Definition 9 (Loop set joining concatenation operator). *Let L_0, L_1 be loop sets.*

$$L_1 \bigotimes L_2 = \{l_0 \bullet l_1 | \forall l_0 \in L_0, \forall l_1 \in L_1\}$$

Definition 10 (All possible I/O sequences). *Let $TSRG$ be reachable and L be all possible loop sets of $TSRG$. All possible input/output sequences, $\Omega(h)$, is $\Omega(h) = \bigotimes_{i=0}^{h} L$.*

The function $\Omega(h)$ of all possible I/O sequences is to explore all possible state trajectories from an initial state through a loop set under adjacency relation. h in $\Omega(h)$ represents a depth of adjacency tree exploration. Figure 3 shows an example of adjacency tree exploration. In figure 3, since loops 'A' and 'B' have

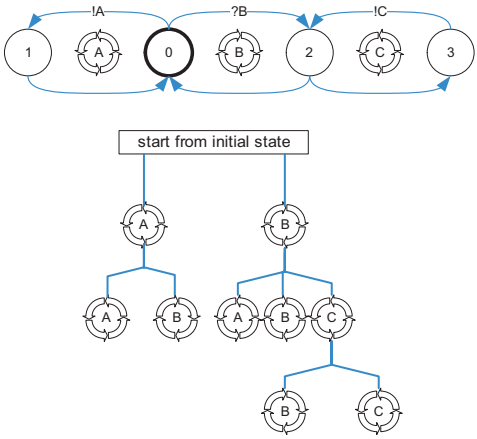


Fig. 3. Adjacency tree exploration.

an initial state at their paths, they only can be selected at the first time when the tree is just explored. After selection of all such loops, next loops may be selected only by the adjacency relation until finding synchronized event sequences.

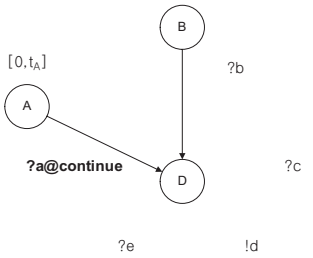
In the view of test sequences, all possible sequences $\Omega(h)$ can cover all nodes and edges with time intervals. However, $\Omega(h)$ has very high complexity both in time and in space. This is because the size of all possible loop set of TSRG is increased at an exponential rate by the number of edges. To reduce the complexity of $\Omega(h)$, a basic loop set is introduced as the following definition.

Definition 11 (Path inclusion). *Let A, B be paths in TSRG. The path A includes the path B, i.e. $A \supseteq B$, if and only if the path A visits all edges in the path B.*

Definition 12 (Loop independency). *Let L be a loop set and $l_i \in L$. The loop set L is independent if only if there is no $l_i \in \{l_1, l_2, \cdots, l_n\}$, such that $l_i \subseteq \bullet_{k=1,k\neq i}^{n} l_k$.*

Definition 13 (Basic loop set). *Let L be a loop set. L is a basic loop set if and only if a loop set $l_1 \in L$ has an independent path for other loop paths in the loop set, i.e., the remaining loop paths without any selected loop path l_1 can not cover all nodes and edges in TSRG.*

A basic loop set passes through all edges in TSRG. However, since a basic loop set is a subset of a loop set its size is less than that of a loop set. It implies that redundancy of visiting edges is reduced by using loops in a basic loop set. Since a previous edge with **CONTINUE** affects time scheduling of a current edge, a minimal independent loop set to visit all edge can not verify time scheduling of all nodes. To solve this, a TSRG model with **CONTINUE** is modified by the following rules. If TSRG has one or more edges with **CONTINUE**, all edges with **CONTINUE** are merged into a newly inserted equivalent node, and they


B
[0,tA]
?b
A
?a@continue
D
?c
?e
!d

Definition 15 (Mergeable node). *Assume that a node s_0 with waiting time $T(s_0)$ connected to edges with transition rules. Then, the node s_0 is mergeable if and only if there exists a node s_i in TSRG with the following conditions :* $\forall e_1 = (s_0, s_1), e_2 = (s_i, s_j) \in E, s_i \neq s_0 \wedge \theta_N(s_0) = \theta_N(s_i) \wedge \theta_N(s_1) = \theta_N(s_j) \wedge ((\theta_E(e_1), \theta_E(e_2) \in ((Y \cup \{\tau\}) \times boolean) \wedge \theta_E(e_1) = \theta_E(e_2)) \vee ((x_1, \perp) = \theta_E(e_1), (x_2, \perp) = \theta_E(e_2) \in (X \times boolean) \wedge (x_1 \neq x_2 \vee s_1 \equiv s_j)))$.

To cover mergeable nodes, the following defines the concept of a unique timed I/O (UTIO) which is similar to UIO and UIOv [3].

Definition 16 (UTIO). *An UTIO sequence which discriminates a node s_0 from the other one is an input or time-out sequence, x_0. The output sequence produced in response to x_0 from any node other than s_0 is different from that responded from s_0. That is, for $y_i, y_0 \in Y, e_{ij} = (s_i, s_j), e_{01} = (s_0, s_1) \in E$, $(y_i, false) = \theta_E(e_{ij})$, $(y_0, false) = \theta_E(e_{01})$, $\forall s_i \in N \wedge s_i \neq s_0$, $\theta_N(s_i) \neq \theta_N(s_0) \vee y_i \neq y_0 \vee \theta_N(s_j) \neq \theta_N(s_1)$. And, an ending point of the UTIO sequence at node s_0 is s_0, which means UTIO sequence at node s_0 is a loop path.*

4 Simple Example: Verification of Buffer

The UTIO sequence of a mergeable node should be a loop path, which can be easily attached to basic loop paths. Consider an example of a buffer model of length 2 in figure 5. As shwon in the figure state space of the buffer is two dimensional: processing status and queue length. Processing status, either in Busy or Free, is a status of a processor cascaded to the buffer. Queue length is a maximum size of the buffer. Let us find mergeable nodes and UTIO sequences. Mergeable nodes S_4, S_6, UTIO sequences of nodes S_4 and S_6 are shown in table 1. Two UTIO sequences are attached into the basic loop set of figure 5. The attached basic loop set is shown in table 1. The four basic loop paths can visit all nodes and edges of the buffer model shown in figure 5. Let us further consider another example of a buffer model of length 2 with equivalent nodes. Figure 6 shows two such models: (a) one with three equivalent states and (b) the other

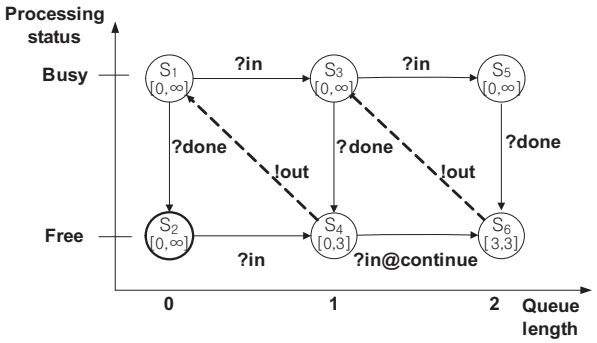


Fig. 5. TSRG of buffer model of length 2.

Table 1. Basic loop paths for buffer of length 2.

UTIO Node	UTIO Seq.
S_4	$!out@[3,3]{\cdot}?done@[0,\infty]{\cdot}?in@[0,3]$
S_6	$\overline{!out@[3,3]{\cdot}?done@[0,\infty]{\cdot}?in@[0,3]}$

No.	Loop paths	Nodes	UTIO Nodes
l_1	$S_2 \rightarrow ?in@[0,3]{\cdot}!out@[3,3]{\cdot}?done@[0,\infty]$	S_2, S_4, S_1	NA
l_2	$S_4 \rightarrow !out@[0,3]{\cdot}\ ?in@[0,\infty]{\cdot}\ ?done@[0,\infty]{\cdot}$ $!out@[3,3]{\cdot}?done@[0,\infty]{\cdot}?in@[0,3]$	S_4, S_1, S_3	S_4
l_3	$S_4 \rightarrow ?in@[0,3], \mathbf{C}{\cdot}$ $\overline{!out@[3,3]{\cdot}?done@[0,\infty]{\cdot}?in@[0,3]}{\cdot}$ $!out@[3,3]{\cdot}?done@[0,\infty]{\cdot}$ $!out@[3,3]{\cdot}?done@[0,\infty]{\cdot}?in@[0,3]$	S_4, S_6, S_3	S_6, S_4
l_4	$S_6 \rightarrow !out@[3,3]{\cdot}\ ?in@[0,\infty]{\cdot}\ ?done@[0,\infty]{\cdot}$ $!out@[3,3]{\cdot}?done@[0,\infty]{\cdot}?in@[0,3]$	S_6, S_3, S_5	S_6

with three equivalent nodes and one transition fault in $S_1' \rightarrow S_5$. The variable h of $\Gamma(h)$ is assigned to 1. Test sequences set, generated from $\Gamma(1)$ of figure 5, is $\{l_1 \bullet l_1,\ l_1 \bullet l_2,\ l_1 \bullet l_3,\ l_2 \bullet l_1,\ l_2 \bullet l_2,\ l_2 \bullet l_3,\ l_3 \bullet l_1,\ l_3 \bullet l_2,\ l_3 \bullet l_3,\ l_3 \bullet l_4,\ l_4 \bullet l_2,\ l_4 \bullet l_3,\ l_4 \bullet l_4\}$. The test sequence $l_3 \bullet l_2$ detects the difference between (a) and (b) of figure 6. The test sequence $l_3 \bullet l_2$ visits the nodes in figure 6(a) through the following order: $S_4 \rightarrow S_1' \rightarrow S_3 \rightarrow S_4 \rightarrow S_6 \rightarrow S_3 \rightarrow S_4$. However, for figure 6(b), the order of node visits of $l_3 \bullet l_2$ is $S_4 \rightarrow S_1' \rightarrow S_5 \rightarrow S_6 \rightarrow S_3 \rightarrow S_4$. Consequently, the UTIO sequence of S_4 detects transition fault of figure 6(b).

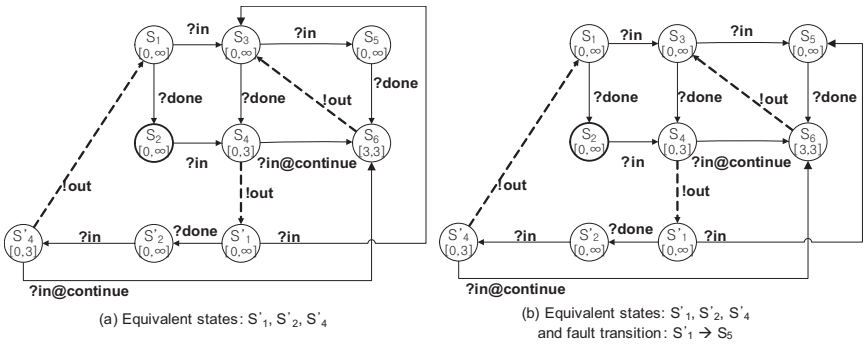


Fig. 6. Buffer model of length 2 with equivalent states.

5 Conclusion

This paper introduced a method for verification of a discrete event model using a test set of timed I/O event sequences. TSRG was employed as a means of modeling of discrete event systems; an implementation of TSRG was assumed to be a

black box with a maximum number of states known. A graph-theoretical analysis of TSRG showed all possible timed I/O sequences $\Omega(h)$ which covers all edges and nodes of TSRG. However, due to some redundancy in visiting nodes/edges of TSRG complexity of $\Omega(h)$ is too high to apply practical verification problems. To solve the problem a basic loop set of TSRG was introduced based on which a minimal test set $\Gamma(h)$ of timed I/O event sequences was extracted. Introduction of predefined UTIOs attached to basic loop paths for mergeable nodes guaranteed that a test coverage of discrete event models using $\Gamma(h)$ was 100 %.

References

1. J. Banks, D. Gerstein, and S. P. Searles, "Modeling process, validation, and verification of complex simulations: A survey", *S.C.S Methodology and Validation, simulation series*, Vol. 19, No. 1, pp 13-18, 1988.
2. Concepcion, and B. P. Zeigler, "DEVS formalism: a framework for hierarchical model development," *IEEE Trans. Software Eng.*, vol. 14, no. 2, pp. 228-241, Feb. 1988.
3. W. Y. L. Chan, C. T. Vuong, M. R. Otp, "An improved protocol test generation procedure based on UIOS," *ACM SIGCOMM Comp. Commun. Review,Symposium proc. communi. arch. & protocols* , vol. 19, pp. 283-294, Aug. 1989.
4. A. T. Dahbura, K. K. Sabnani, and M. Ümit Uyar, "Formal nethods for generating protocol conformance test sequences," *Proc. of the IEEE*, vol. 78, pp. 1317-1325, Aug. 1990.
5. D. Lee, and M. Yannakakis, "Principles and methods of testing finite state machines-A survey," *Proc of the IEEE*, vol. 84, pp. 1090-1123, Aug. 1996.
6. A. En-Nouaary, R. Dssouli, and F. Khendek, "Timed Wp-method: Testing real-time systems," *IEEE Trans. Software Eng.*, vol. 28, pp. 1023-1038, Nov. 2002.
7. K. J. Hong, and T. G. Kim, "A Verification Of Time Constrained Simulation Model Using Test Selection:A New Approach", *In preparation*, 2004.