

## FINITE STATE MACHINE TESTING FROM AN OR STATE REFINEMENT DESIGN

Florentin Ipate, Tudor Bălănescu

*Department of Mathematics and Computer Science, University of Pitesti*

fipate@ifsoft.ro, tudor.balanescu@yahoo.com

**Abstract** The paper formalizes the OR refinement of finite state machines and develops an efficient method to integrate the test sets of the component machines in order to test the flattened machine. The approach supports component-software development since it constructs test sets for an assembly of systems from readily available test sets of the prefabricated parts.

**Keywords:** finite state machine, OR state refinement, test generation, verification,  $W$ -method  
**2000MSC:** 68N32

### 1. INTRODUCTION

The test set generation problem for protocols modeled by *deterministic finite state machines* has attracted a great deal of interest and has generated a substantial literature. Given two deterministic finite state machines  $S$  and  $I$ , the former representing the specification and the latter the implementation, a *test set* is a set of input sequences that, if  $S$  and  $I$  are not equivalent, contains at least one sequence that produces different results when applied to the two machines. The test set is generated from the specification  $S$  and, in principle, little information is available about the implementation  $I$ . For instance, in the so called  $W$ -method [7] the only information about the implementation is an upper bound on the number of its states. Originally, the  $W$ -method was developed in the context of *completely specified* deterministic finite state machines and was recently revised for the case in which both the specification and the implementation may be *partially specified*. The  $W$ -method has also been adapted to finite state machines based specification languages, such as stream X-machines [11], [10] and Harel statecharts [3], [4].

On the other hand, in practice a finite state machine specification is usually constructed through a process of refinement; the equivalent flattened finite state machine of such a refinement can be constructed and the  $W$ -method can be applied to test the flattened machine. This approach might not always be practical, for large scale systems the flattened machine can be extremely complex and this will result in a test set of unmanageable size.

The paper formalizes the OR refinement of Finite State Machines (in section 4) and develops an efficient testing strategy for this kind of refinement.

## 2. PRELIMINARIES

**Definition 2.1** A Deterministic Finite State Machine (DFSM for short) is a quintuple  $M = (Q, Li, Lo, h, M_0)$ , where  $Q$  is a finite set of states,  $Li$  is a finite set of input symbols,  $Lo$  is a finite set of output symbols,  $h : Q \times Li \rightarrow Q \times Lo$  is the behavior (partial) function,  $M_0$  is the initial state.

A DFSM is often described by a state-transition diagram as in Example 2.1.

Let  $q, r \in Q, a \in Li, o \in Lo$ . We write  $q \rightarrow a/o \rightarrow r$  to denote a transition from  $q$  to  $r$  with label  $a/o$  if  $h(q, a) = (r, o)$ . As a response to an arbitrary input sequence  $x = a_1 \cdots a_n, n \geq 0, a_i \in Li$  for all  $i, 1 \leq i \leq n$ , a state  $q \in Q$  of the DFSM  $M$  may produce an output sequence  $t = o_1 \cdots o_n, o_i \in Lo$  for all  $i, 1 \leq i \leq n$ , if there are the following  $n$  transitions  $q_i \rightarrow a_i/o_i \rightarrow q_{i+1}, 1 \leq i \leq n$  and  $q_1 = q$ . It is said that the input  $a_1 \cdots a_n$  leads the machine from the state  $q$  to the state  $r$  or that the state  $r$  is reachable from  $q$ . We denote this by  $q \Rightarrow a_1/o_1 \cdots a_n/o_n \Rightarrow q_{n+1}$  or simply by  $q \Rightarrow a_1/o_1 \cdots a_n/o_n \Rightarrow$  if the reached state  $q_{n+1}$  may be omitted. Moreover, if the produced output string  $o_1 \cdots o_n$  is not significant, it may be omitted, writing either  $q \Rightarrow a_1 \cdots a_n \Rightarrow q_{n+1}$  or  $q \Rightarrow a_1 \cdots a_n \Rightarrow$ .

For an input sequence  $a_1 \cdots a_n$ , we say that two states  $q, r \in Q$  give *identical response* to  $a_1 \cdots a_n$  if:  $q \Rightarrow a_1/o_1 \cdots a_n/o_n \Rightarrow$  iff  $r \Rightarrow a_1/o_1 \cdots a_n/o_n \Rightarrow$  for all  $o_1, \dots, o_n \in Lo$ . Note that the definition includes the case where no such  $o_1, \dots, o_n$  exist (i.e. for partially specified DFSM).

Let  $A \subseteq Li^*$  be an arbitrary set of input sequences. Two states  $q, r \in Q$  are said to be *A-equivalent* if  $q$  and  $r$  give identical responses for each input sequence  $x \in A$ . We denote this by  $q \cong_A r$ . Otherwise, they are said to be *A-distinguishable* (denoted by  $q \not\cong_A r$ ). If  $A = Li^*$  (i.e.  $q$  and  $r$  give identical responses for any input sequence) then  $q$  and  $r$  are said to be *equivalent*.

Let  $M$  and  $N$  be two DFSMs with the same sets of input ( $Li$ ) and output ( $Lo$ ) symbols and  $A \subseteq Li^*$ . Then  $M$  and  $N$  are said to be *A-equivalent* if their initial states  $M_0$  and  $N_0$  are *A-equivalent*. Otherwise,  $M$  and  $N$  are said to be *A-distinguishable*. If  $A = Li^*$  then  $M$  and  $N$  are said to be *equivalent*.

A finite set  $R \subseteq Li^*$  is said to be a *state cover* of a DFSM  $M$  if the empty sequence  $\epsilon$  is contained in  $R$ , and, for every state  $q \in Q$  other than  $M_0$ ,  $R$  contains an input sequence  $x \in R$  that may lead the machine from the initial state  $M_0$  to  $q$  (i.e.  $M_0 \Rightarrow x \Rightarrow q$ ). A finite set  $W \subseteq Li^*$  is said to be a *characterization set* of a DFSM  $M$  if any two distinct states  $q, r \in Q$  are *W-distinguishable*.

**Example 2.1** Consider a simple tape recorder capable of playing and recording a tape. For simplicity, we consider the tape to be infinite. The DFSM

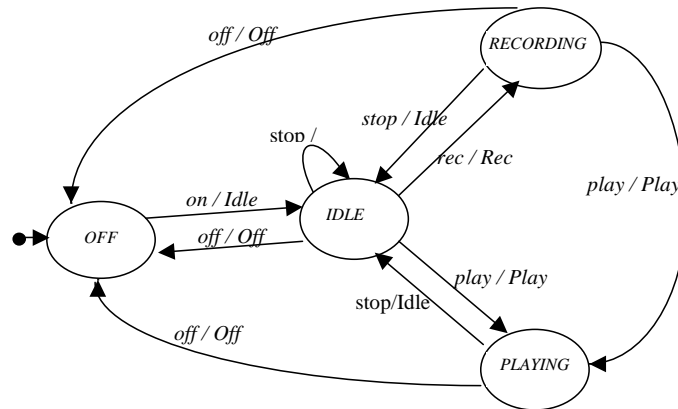


Fig. 1. DFSM model of a tape recorder.

presented in fig. 1 models the control of this tape recorder. The inputs to this DFSM are events:  $Li = \{\text{play}, \text{stop}, \text{rec}, \text{on}, \text{off}\}$ . The outputs are the operations performed by the tape recorder:  $Lo = \{\text{Play}, \text{Idle}, \text{Rec}, \text{Off}\}$ . The initial state *OFF* is pointed at by a transition from a blob.

Then  $R = \{\epsilon, \text{on}, \text{on} \cdot \text{rec}, \text{on} \cdot \text{play}\}$  is a state cover of this DFSM and  $W = \{\text{on}, \text{rec}, \text{play}\}$  is a characterization set.

A DFSM  $M$  is said to be *minimal* if any other equivalent DFSM has at least the same number of states as  $M$ ; a DFSM  $A$  is minimal if and only if there exists a state cover and a characterization set of  $A$  [8].

Let  $M = (Q, Li, Lo, h, M_0)$  and  $M' = (Q', Li, Lo, h', M'_0)$  be two DFSMs. Then a bijective function  $f : Q \rightarrow Q'$  is called an *isomorphism* from  $M$  to  $M'$  if for every states  $q_1, q_2 \in Q$ , input symbol  $a \in Li$  and output symbol  $o \in Lo$ ,  $q_1 \rightarrow a/o \rightarrow q_2$  if and only if  $f(q_1) \rightarrow a/o \rightarrow f(q_2)$ . Any two equivalent minimal DFSMs are isomorphic [8], so for any DFSM  $M$ , there exists an unique (up to an isomorphism) minimal DFSM  $M'$  equivalent to  $M$ .  $M'$  is called the *minimal DFSM* of  $M$ .

### 3. THE W-METHOD

Let  $S$  be a DFSM specification and  $I$  a DFSM model of the implementation. The  $W$ -method generates a test set (a finite set of sequences) that can determine any error in  $I$  with respect to  $S$ , provided that the number of states of  $I$  is bounded by an integer  $m$  that may be larger than the number  $n$  of states of  $S$ . The set is generated from the specification  $S$  and no information is available about  $I$  except the above mentioned upper bound.

The  $W$ -method was first developed in the context of *completely specified* DFSMs [7]. In [1], the method was extended to cope with (possibly) *partially specified* DFSMs.

**Definition 3.1** Let  $S = (Q_S, Li, Lo, h_S, S_0)$  and  $I = (Q_I, Li, Lo, h_I, I_0)$  be two DFSMs. A finite set  $Y \subseteq Li^*$  is called a test set for  $S$  and  $I$  if:  $S$  and  $I$  are  $Y$ -equivalent  $\implies S$  and  $I$  are equivalent.

The  $W$ -method involves the selection of two sets of input sequences:

- $W \subseteq Li^*$ , a characterization set of  $S$
- $R \subseteq Li^*$ , a state cover set of  $S$ .

The method makes the following assumptions about the specification  $S$  and the implementation  $I$ :

- $S$  is deterministic and minimal
- $I$  is deterministic and the number of states of  $I$  is bounded by an integer  $m$ .

For the case where  $S$  and  $I$  are assumed to be *completely specified* the  $W$ -method provides a test set

$$Y_{cs} = R \cdot Li[m - n + 1] \cdot W,$$

with  $Li[j] = \{\epsilon\} \cup Li \cup \dots \cup Li^j$  for  $j \geq 0$  [7].

However,  $Y$  is not always a valid test set when partially specified DFSMs are considered, as shown by the following example. Consider the partially specified DFSM specification  $S = (\{0, 1\}, \{a, b\}, \{a, b\}, h, 0)$  with  $h(0, a) = (1, a)$  and  $h$  undefined elsewhere. Then  $R = \{\epsilon, a\}$  is a *state cover* of  $S$  and  $W = \{a\}$  is a *characterization set* of  $S$ . Let  $I = (\{0, 1\}, \{a, b\}, \{a, b\}, h', 0)$  be an implementation of  $S$  with  $h'(0, a) = (1, a)$ ,  $h'(0, b) = (1, b)$  and  $h'$  undefined elsewhere. Since  $S$  and  $I$  have the same number of states ( $n = m = 2$ ), the  $W$ -method gives  $Y_{cs} = R \cdot Li[1] \cdot W = \{a, a \cup a, b \cup a, a \cup a \cup a, a \cup b \cup a\}$ . However, the two DFSMs are  $Y_{cs}$ -equivalent but they are not equivalent. Indeed, the single word which distinguishes them is  $b$ , which is not contained in  $Y_{cs}$ .

This situation occurs because when  $S$  and  $I$  are completely specified whenever  $S$  and  $I$  are  $\{s\}$ -equivalent for an input sequence  $s$ , they are also  $\{t\}$ -equivalent for any prefix  $t$  of  $s$ . This is no longer true when at least one of  $S$  and  $I$  is partially specified. Therefore, one way of extending the  $W$ -method to the general case where the machines may be partially specified is to take all prefixes of the sequences in  $Y_{cs}$ . It has been proved, however, that a smaller subset of prefixes is sufficient [1], this is

$$Y = R \cdot Li[m - n + 1] \cdot (W \cup \{\epsilon\}).$$

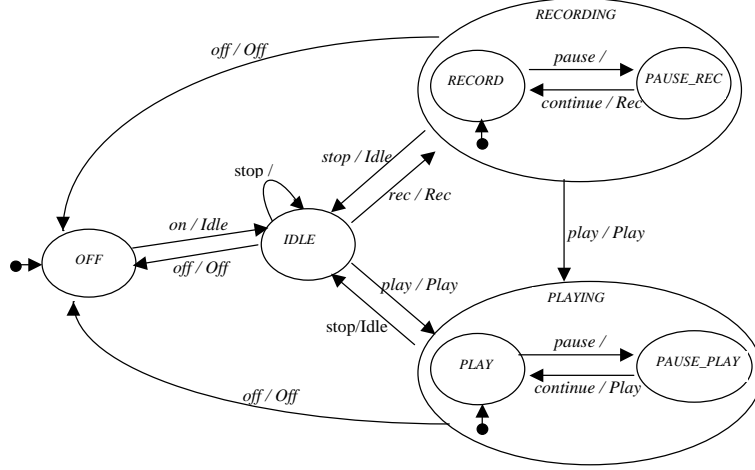


Fig. 2. Refinement of OR states.

#### 4. OR STATE REFINEMENT

One way of refining an existing DFSM specification is to replace one or more states with other DFSMs. The semantics of this transformation is that given by the expansion of OR states in state charts [3] and is illustrated by the following example.

**Example 4.1** *The original model in Example 2.1 can be refined by allowing the tape recorder to pause while playing or recording. For instance, the behavior of the RECORDING state is a two-state machine describing whether the tape recorder is actively recording or waiting for the user to press the continue command. The refinement of the two states, PLAYING and RECORDING (called in what follows OR states), is graphically illustrated in fig. 2. The flattened DFSM for the refined model is presented in fig. 3. Any transition arriving to the OR state in the original specification will now be directed to the initial state of the DFSM that replaces that state. Any transition leaving the OR state will now leave all the states of the replacement DFSM.*

**Definition 4.1** *A DFSM  $M' = (Q', Li', Lo', h', M'_0)$  is called an OR state refinement of a DFSM  $M = (Q, Li, Lo, h, M_0)$  if  $Q \subseteq Q'$ ,  $Li \subseteq Li'$ ,  $Lo \subseteq Lo'$ ,  $M_0 = M'_0$  and there is a partition  $E = \{E_q\}_{q \in Q}$  of  $Q'$  with  $q \in E_q$  for all  $q \in Q$  such that the following hold:*

- $h(q, a) = h'(q', a)$  for all  $q \in Q$ ,  $a \in Li$ ,  $q' \in E_q$
- $\pi_1(h'(q', a)) \subseteq E_q$  for all  $q \in Q$ ,  $a \in Li' - Li$ ,  $q' \in E_q$ , where  $\pi_1 : Q' \times Lo' \rightarrow Q'$  denotes the projection function

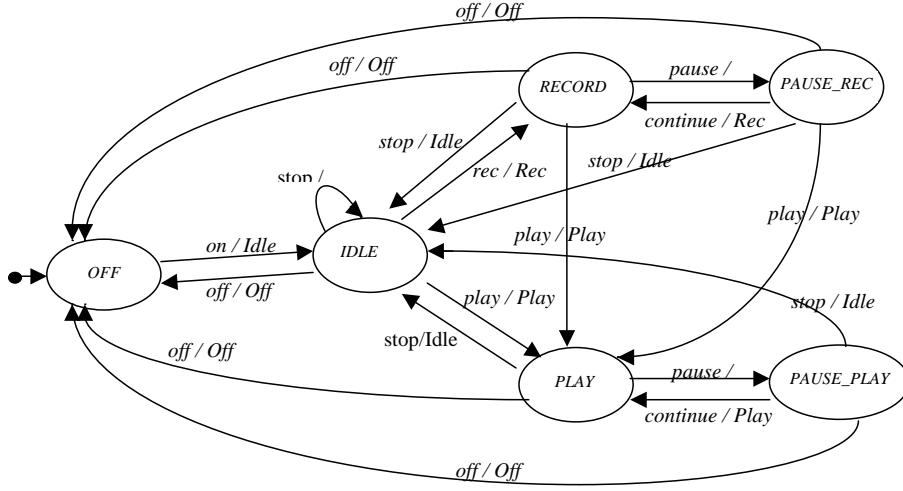


Fig. 3. Flattened DFSM for OR state refinement.

$Li_0 = Li' - Li$  is called the refining input alphabet.

For  $q \in Q$ , the DFSM  $M_q = (E_q, Li_0, Lo', h_q, q)$ , where  $h_q$  is the restriction of  $h'$  to  $E_q \times Li_0$ , is called the refining machine for  $q$ .

For simplicity, in Definition 4.1, the refined state and the initial state of the corresponding refining machine are considered to be identical. For clarity, they are given different names in our examples, where only the states *PLAYING* and *RECORDING* are refined. So the partition  $E$  is defined by  $E_{PLAYING} = \{PLAY, PAUSE\_PLAY\}$ ,  $E_{RECORDING} = \{RECORD, PAUSE\_REC\}$  and  $E_q = \{q\}$  for the other states. The refining input alphabet is  $Li_0 = \{continue, pause\}$ .

## 5. OR STATE REFINEMENT TESTING

Consider the problem of generating test sets from a specification of the form of an OR state refinement of a DFSM  $S = (Q_S, Li, Lo, h_S, S_0)$ . The  $W$  method can be directly applied to the flattened specification  $S' = (Q'_S, Li', Lo', h'_S, S'_0)$  and, furthermore, a state cover  $R_{S'}$  and a characterization set  $W_{S'}$ , can be constructed from a state cover  $R_S$  and a characterization set  $W_S$  of the original DFSM  $S$  and state covers  $R_q, q \in Q_S$  and characterization sets  $W_q, q \in Q_S$  of the refining machines using the following formulae

$$R_{S'} = R_S \otimes \mathcal{R},$$

where  $\mathcal{R} = \{R_q \mid q \in Q_S\}$  is a set that contains a state cover  $R_{M_q}$  for each state  $q \in Q_S$  and for  $A \subseteq Li^*$  and  $\mathcal{B} = \{B_q \mid q \in Q_S\} \subseteq 2^{Li^*}$ ,  $A \otimes \mathcal{B} = \{a \cdot b \mid a \in A, b \in B_q, S_0 \Rightarrow a \Rightarrow q\}$ . That is  $R_{S'}$  is obtained by concatenating each

sequence in  $R_S$  that forces  $S$  into  $q$  from the initial state  $S_0$  with all sequences in  $R_q$

$$W_{S'} = W \cup \left( \bigcup_{q \in Q_S} W_q \right).$$

The proofs are straightforward.

On the other hand, since the specification has been developed through a refinement process, one would also expect the implementation to be build incrementally: initially a developer may only have a skeleton of the system then step by step fill it with details. For instance the DFSM in fig. 1 can be built first. Afterwards, when the functionality contained in the *RECORDING* (or *PLAYING*) refining machine is added it can be considered not to have an effect outside this state, (i.e. any transition leaving an OR state in  $I$  will leave all the states of the refining machine in  $I'$  and have the same destination as the original transition), so implementation can also be modelled by an OR state refinement  $I' = (Q'_I, Li', Lo', h'_I, I'_0)$  of some DFSM  $I = (Q_I, Li, Lo, h_I, I_0)$ . We can now take advantage of this situation and show how the test sets for  $S$  and the refining machines can be reused in the construction of the new test set.

Let  $S'$ , the specification, be an OR state refinement of  $S$  and  $I'$ , the implementation, be an OR state refinement of  $I$ , as above. Our aim is to construct a test set  $Y'$  for  $S'$  and  $I'$ . Let  $n$  be the number of states of  $S$  and  $m$  the upper bound for the number of states of  $I$ . For  $q \in Q_S$  let  $S_q$  be the refining machine for  $q$  and for  $p \in Q_I$  let  $I_p$  be the refining machine for  $p$ . Let  $n_q$  be the number of states of  $S_q$ ,  $m_p$  the upper bound for the number of states of  $I_p$  and  $k = \max_{p \in Q_I} m_p$ . We assume that  $S$  is minimal and  $S_q$  is minimal for all  $q \in Q_S$ . Without loss of generality all states of  $I$  are assumed to be reachable from the initial state  $I_0$  (otherwise the non-reachable states may be removed). Similarly, for  $p \in Q_I$ , all states of  $I_p$  are assumed to be reachable from the initial state  $p$ .

Then the test set  $Y'$  we are after is

$$Y' = Y \cup U,$$

where  $Y = R \cdot Li[m - n + 1] \cdot (W \cup \{\epsilon\})$  is a test set for  $S$  and  $I$  and  $U$  is constructed using the following sets:

- $Y_q = R_q \cdot Li_0[k - n_q + 1] \cdot (W_q \cup \{\epsilon\})$  for all  $q \in Q_I$ , where  $R_q$  is a state cover and  $W_q$  a characterization set of  $S_q$ ; then  $Y_q$  is a test set for  $S_q$  and  $I_p$  for all  $p \in Q_p$
- $T = R \cdot Li[m - n]$

Then

$$U = T \otimes \mathcal{Y},$$

where  $\mathcal{Y} = \{Y_q \mid q \in Q_S\}$  is a set that contains a set  $Y_q$  for each state  $q \in Q_S$ .

For our example consider  $m = 5$  and  $k = 2$ . Then  $T = R \cdot Li[1] = \{\epsilon, on, on \cdot rec, on \cdot play\} \cdot \{\epsilon, play, stop, rec, on, off\}$ ,  $R_{RECORDING} = R_{PLAYING} = \{\epsilon, pause\}$ ,  $W_{RECORDING} = W_{PLAYING} = \{pause\}$  so  $Y_{RECORDING} = Y_{PLAYING} = \{\epsilon, pause\} \cdot Li_0[1] \cdot \{pause\}$ ;  $R_{OFF} = R_{IDLE} = W_{OFF} = W_{IDLE} = \{\epsilon\}$ , so  $Y_{OFF} = Y_{IDLE} = Li_0[2]$ . Thus  $U = \{\epsilon, on \cdot off, on \cdot rec \cdot off, on \cdot play \cdot off\} \cdot Y_{OFF} \cup \{on, on \cdot stop, on \cdot rec \cdot stop, on \cdot play \cdot stop\} \cdot Y_{IDLE} \cup \{on \cdot rec\} \cdot Y_{RECORDING} \cup \{on \cdot play\} \cdot Y_{PLAYING}$ .

**Lemma 5.1** *For  $S$  and  $I$  as above, if  $S$  and  $I$  are equivalent, then  $T$  is a state cover of  $I$ .*

*Proof.* We prove by induction on  $0 \leq j \leq m - n$  that  $T_j = R \cdot Li[j]$  reaches at least  $n + j$  states of  $I$ . For  $j = 0$  the statement follows since  $R$  is a state cover of  $S$ , which is the minimal DFSM of  $I$ . Assume the statement true for  $j$ . Then either  $T_j$  reaches all the states of  $I$  or  $T_{j+1}$  will reach at least one more state than  $T_j$ . Hence  $T_{j+1}$  reaches at least  $n + j + 1$  states of  $I$ .  $\square$

**Theorem 5.1** *For  $S, S', I, I'$  and  $Y'$  as above,  $Y'$  is a test set for  $S'$  and  $I'$ .*

*Proof.* Assume that  $S'$  and  $I'$  are  $Y'$ -equivalent. Then  $S'$  and  $I'$  are  $Y$ -equivalent, hence  $S$  and  $I$  are  $Y$ -equivalent. Since  $Y$  is a test set for  $S$  and  $I$ ,  $S$  and  $I$  are equivalent. Furthermore, from Lemma 5.1 it follows that  $T$  is a state cover of  $I$ .

Let  $q \in Q_S$  and  $p \in Q_I$  be two equivalent states of  $S$  and  $I$ , respectively. Then there exists  $t \in T$  such that  $S_0 \Rightarrow t \Rightarrow q$  and  $I_0 \Rightarrow t \Rightarrow p$ . Since  $S'$  and  $I'$  are  $U$ -equivalent it follows that  $S_q$  and  $I_p$  are  $Y_q$ -equivalent. Hence  $S_q$  and  $I_p$  are equivalent.

Therefore, we have proved that  $S$  and  $I$  are equivalent and for any pair of equivalent states  $q \in Q_S$ ,  $p \in Q_I$ ,  $S_q$  and  $I_p$  are equivalent. Then from Definition 4.1 it follows that  $S'$  and  $I'$  are equivalent.  $\square$

It can be shown that the implementation refinement approach considerably reduces the size of the test set generated.

## 6. CONCLUSIONS

The paper formalizes the OR refinement for finite state machines and it shows how if the implementation is also constructed through a process of refinement, then the test process can be distributed into smaller chunks (i.e. the original machine and the refining machines), thus diminishing the effort devoted to testing and the size of the final test set.

The partial OR (p-OR) state refinement of finite state machines will also be considered in a future paper. The generalization of these test generation methods for refined DFSMs to other finite state machine based specification languages, such as Harel statecharts, will also be investigated.

## References

- [1] Bălănescu, T., Gheorghe, M., Ipate, F., Holcombe, M., Formal black box testing for partially specified deterministic finite state machines, *Foundations of Computing and Decision Systems*, **28** (2003).
- [2] Bălănescu, T., Ipate, F., The Wp method for partially specified deterministic finite state machines, *Annals of Bucharest University, Informatica*, **52**, **1** (2004), 47-61.
- [3] Bogdanov, K., *Automatic testing of Harel's statecharts*, PhD thesis, The University of Sheffield, 2000.
- [4] Bogdanov, K., Holcombe, M., *Statechart testing method for aircraft control systems*, J. Softw. Test. Verific. Reliability, **11** (2001), 39-54.
- [5] Boiten, E., Derrick, J., *Unifying concurrent and relational refinement*, Electronic Notes in Theoretical Computer Science, **70** (2002), 35 pp.
- [6] Brookes, S.D., Roscoe, A.W., *An improved failures model for communicating processes*, Pittsburgh Symposium on Concurrency, Lecture Notes in Computer Science, **197** (1985), 281-305.
- [7] Chow, T.S., *Testing software design modeled by finite-state machines*, IEEE Trans. Softw. Engrg, **4** (1978), 178-187.
- [8] Eilenberg, S., *Automata, languages and machines*, Academic, New York, 1974.
- [9] Fujiwara, S., von Bochmann, G., Khendek, F., Amalou, M. and Ghedamsi, A., *Test selection based on finite state models*, IEEE Trans. Softw. Engrg, **17** (1991), 591-603.
- [10] Holcombe, M. and Ipate, F., *Correct systems: Building a business process solution*, Springer, Berlin, 1998.
- [11] Ipate, F. and Holcombe, M., *An integration testing method that is proved to find all faults*, Intern. J. Comput. Math, **69** (1997), 159-178.
- [12] Ipate, F. and Holcombe, M., *An integrated refinement and testing method for stream X-machines*, Appl. Algebra Engrg., Comm. Comput, **13** (2002), 67-91.
- [13] Lee, D. and Yannakakis, M., *Principles and methods of testing finite state machines - A survey*, Proceed. IEEE, **84** (1996), 1090-1123.
- [14] Luo, G., v. Bochmann, G. and Petrenko, A., *Test selection based on communicating nondeterministic finite-state machines using a generalised Wp-method*, IEEE Trans. Softw. Engrg, **20** (1994), 149-161.
- [15] Milner, R. *Communication and Concurrency*, Prentice-Hall, 1989.
- [16] Petrenko, A., *Fault model-driven test derivation from finite state models: Annotated bibliography*, Modelling and Verification of Parallel Processes, 4th Summer School, MOVEP 2000, Springer Verlag, Lecture Notes in Comput. Sci. **2067** (2001), 196-205.
- [17] Urbasek, M., *Net Transformations for Petri Net Technology*, Bulletin of the EATCS, **80** (2003), 77-94.
- [18] v. Bochmann, G., Petrenko, A., *Protocol testing: Review of methods and relevance for software testing*, ACM International Symposium on Software Testing and Analysis, Seattle USA, 1994, 109-123.

