Automatic Test Case Generation of the Behavior of Communication Software Systems

Ana Maria Ambrosio[1]

[1]Ground Systems Development Division (DSS)
ana@dss.inpe.br

Solon V. de Carvalho[2]
Nandamudi L. Vijaykumar[2]

[2]Associated Laboratory of Computing and Applied Mathematics (LAC)
National Institute for Space Research (INPE)
{solon, vijay}@lac.inpe.br

Eliane Martins[3]

[3]Institute of Computing (IC)
State University of Campinas (UNICAMP)
eliane@ic.unicamp.br

Abstract

This paper addresses the problem of automatically generating test cases of the behavior of communication software systems, the protocols, which are usually specified in Extended Finite State Machines. The specification technique Statecharts is considered here for specifying the protocol behavior with the objective of using its descriptive power of the hierarchy, orthogonality and synchronization features. To deal with the state explosion problem a strategy is proposed for test generation in steps (separated phases) based on the concepts of independent and synchronizing transitions. Another topic of study is related to the formal aspects of the specification into matrix algebra and Kronecker operators. Both the proposed strategy and mapping of the specification into matrix algebra are in the initial phase.

1. Introduction

In the space system software technology area, communication systems play an important role especially with the progress in data communication protocol. The standardization of space protocols and interfaces lead to facilitate cross-supporting among agencies and even the re-use of common systems in different missions.

A set of space protocols has been standardized, due to the efforts of the CCSDS committee, whose standards may be found in [http://www.ccsds.org/all-books.html]. These protocols are formally specified in Extended Finite State Machine (presented in the form of state x event tables) so that any space agency or any company in the world may implement and use the space protocols benefiting from the standardization advantages.

In order to assure the correct communication between the computer systems from different manufacturers it must be possible to ascertain that the implemented protocols really conform to the standard protocol specification. This activity is known as protocol conformance testing. An overview of the state-of-art of protocol testing may be found in [Bochmann et al, 1994], [Dssouli, 1999] and [Lai, 2002]. However, protocol conformance testing is not an easy task. The difficulties in conformance testing are related to insufficient techniques and support to test specification, besides that tests take too much time and effort. It may consume up to 50% of the project resources in a software development project [Tretmans, 1999].

Many efforts have been concentrated in automating the test case generation from protocol specifications in order to cover the conformance testing activity. The literature has pointed out several approaches of automating the test case generation for FSM-based protocol specifications [Tan, 1996]. In situations where data and conditions of transitions have to be considered, EFSM-based protocol specifications are best suited [Bourhfir, 1997], [Martins,1999].

In fact, communication protocols are considered as reactive systems which deal with distribution, communication and synchronization features. Because of that, the use of Statecharts as defined in [Harel, 1987] is proposed to represent the protocol specification.

A Statecharts-based specification may represent the behavior of different components in parallel, in depth and allows broadcasting of events [Harel, 1987]. Some approaches in generating test cases from Stetecharts can be found in [Bogdanov, 1999], [Hong,2001], [Kim, 1999], [Fabri, 1999].

Statecharts greatly simplify the system behavior specification, through the philosophy of top-down approach; the system behavior is specified by the behavior of each component of the system. In case of protocol specification, for example, each virtual channel could be represented by one component. For protocol with more than one level, each level could be specified by one component, and the interaction between each level could be represented by synchronizing events.

However, in order to generate test cases from a Statecharts-based specification, all possible configurations (system global state) that are produced while converting Statecharts into an equivalent Finite State Machine must be taken into consideration. In doing that one has to deal with the state explosion problem. The number of Statecharts configurations is usually equal (in the worst case) to the product of the number of states of each system component. In order to automatically generate a tractable number of test cases from a Statecharts-based specification, Hong [Hong, 2001] proposed to generate test cases as counterexamples during the model checking of the specification.

In the approach discussed here, a method is proposed to incrementally generate test cases based on the transition classification. The transitions may be independent or synchronized. The method suggests to generate test cases first, considering each component independently. Later only the synchronizing transitions are considered. The system behavior is represented in matrix algebra. To generate test case, in both phases, we will make use of an existing tool named ConData [Sabião, 1999], which implements a transition-tour-based method to create the behavioral test cases from the specification.

The paper is organized as follows: section 2 presents the formal description of two models used to specify the protocol behavior, the EFSM that has been used for this purpose and the Statecharts that is proposed here to specify the protocol synchronization aspects. Section 3, summarizes some concepts that have been explored in order to formalize the formal representation of the synchronizing transitions into matrices. Section 4 concludes the paper as well as pointing out future works.

2. Protocol Behavior Modeling

Generally, the behavior of a protocol is specified in a Finite State Machine (FSM), which is a formal way to define how of the communication rules should be implemented. A finite state machine contains a finite number of states and according to the received input or stimulus it generates the outputs. Because of the practical importance of this specification technique, there is a great number of research in automatic testing on FSM-based specification. In [Lee, 1996] the FSM-based methods, principles and problems are discussed.

A FSM-base covers only the behavior aspect of the specified system. However, in practice, the protocol specification includes variables, whose values are checked and changed as the protocol state changes. So, it is used the *Extended Finite State Machine* – EFSM in order to include control and data aspects of the protocol at is specification.

The EFSM includes the variable handling and the concept of condition associated to a transition. A EFSM is formally defined as a 8tuple $M = (\Sigma, Q, \delta, q_0, F, V, P, A)$ where:

- Σ = input symbol alphabet
- Q = finite set of possible states
- δ = state transition function - $\delta: Q \times \Sigma \times P(V) \rightarrow Q \times O \times A(V)$
- q_0 = initial state; $q_0 \in Q$
- F = output interactions set

- V = variables set
- P = set of predicates expressed in terms of variables of V, parameters of Σ and constants;
- A = set of actions related to variables.

In the EFSM, a transition is represented as

t = *(source-state, target-state, input, predicate, output)*, where: *source-state* and *target-state* ∈ Q; *input* ∈ Σ; *predicate* ∈ P and *output* ∈ F.

Statecharts extend the finite state machine (FSM) with the concepts of hierarchical state decomposition (depth/abstraction), orthogonality (representation of parallel activities) and interdependency (broadcast communication). The basic elements of Statecharts to represent a system are: *configuration* (the global state of the system, which means the set of the active basic state of each orthogonal component), *event* (external – explicitly stimulated; and internal – automatically stimulated by the internal logic of Statecharts), *action* (which may cause an output or a variable change or yet trigger another event), and the concepts of *condition*, *transition*, *expressions*, *variables* and *labels*. The general notation of a transition label in Statecharts is *event[condition]/action*. Action may be a change of an expression, a change of a variable or even events that are triggered in other orthogonal components. Internal events are: *true (condition)*, *false (condition)*, *entered(X)* and *exit(X)*.

An example of a statecharts-based specification is illustrated in Figure 1, where a producer – consumer protocol is specified. In this example there are three orthogonal components, M_1, Buffer (B) and M_2. M_1 comprises three basic states: W_1, T_1 and F_1. In the transition from the state W_1, to T_1, the expression *[not inB.2]* is a condition associated to the event *a1*, which means that the transition will be trigged only if the component Buffer is not in the state 2. And, in the transition from T_1 to W_1 the action *inc* will cause a transition in the component B. More detail of the Statecharts notation may be found in [Harel at al, 1998].

3. Statecharts synchronizing transitions in Matrix Algebra

In this section the concepts of independent and synchronizing transitions are described. In the following, the concept of transition descriptor to represent synchronizing transitions is shown as Kronecker product of boolean matrices [Eisele and Mason, 1970]. The sum of those descriptors generate a reduced state machine from which test cases will be extracted, the main subject of this thesis.

Synchronizing transition concept

A synchronizing transition is a transition from one basic state to another in a Statecharts component that comprises at least one of the following features:

- a broadcast event – an event that is received by more than one component,
- an event generated by an action of other component,
- a state condition – a condition (if not satisfied) that avoids the transition firing namely. The approach discussed in this paper considers only "not in state" condition.

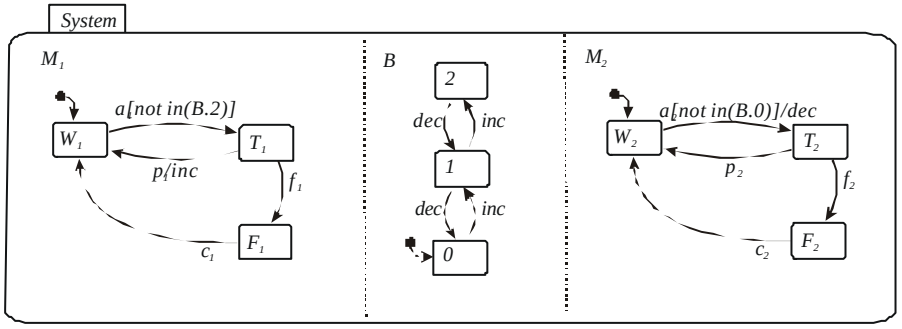

Figure 1 – Statecharts-based specification of producer-consumer protocol

In order to illustrate the concepts presented above, observe the Statecharts-based specification of a producer – consumer protocol showned in Figure 1. In this example there are six synchronizing transitions: (W_1, a_1 [not in B.2], T_1), (T_1, p_1/inc, W_1), (W_2, a_2 [not in B.0], T_2), (B.2, dec, B.1), (B.1, dec, B.0), (B.0, inc, B.1), (B.1, inc, B.2).
The independent transitions are (T_1, f_1, F_1), (F_1, c_1 , W_1), (T_2, f_2, F_2) , (F_2, c_2, W_2), (T_2, p_2, W_2).

Matrix representation of the transition descriptor

Any Finite State Machine may be written as a boolean matrix.. In such a matrix, $M_{|state| \times |state|}$, the lines are the source states, the columns the target states and the elements may be 1 if there is a transition (S, *e*, S'), where S and S' are states of the machine and *e* a transition from S to S', and 0 otherwise. For the component M_1 shown in Figure 1, which comprises the transitions (W_1, *a_1 [not in B.2]*, T_1), (T_1, *p_1/inc*, W_1), (T_1, *f_1*, F_1), (F_1, *c_1* , W_1) the corresponding matrix is shown in Figure 2.

$$\begin{array}{c} \\ W_1 \\ T_1 \\ F_1 \end{array} \begin{array}{c} \begin{array}{ccc} .....W_1 & ..T_1 & ..F_1 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 2. Boolean Matrix of the Component M_1

As synchronizing transitions involve communication among components, one single matrix representing one component is not enough. It is necessary to take into account the behavior of other components related to such transitions based on actions, restrictive conditions (not in state) and events that are fired in more than one component. Taking as an example, consider the transition from W_1 to T_1 in M_1 component. Event a_1 is tied to a condition (not in state) directly related to the behavior in the B component. How to represent this dependence in a matrix form?

M_1 component is represented by the states that are involved in the transition, i.e., W_1 and T_1 (2 x 2 matrix is enough). This means that states that are not related in the transition are not necessary to be represented or are represented by the identity matrix. Matrix representing B component is described as an identity matrix except the line (column) corresponding to the state mentioned in the condition is assigned to zero. The algebraic expression of the matrices representing the synchronizing transition of the system as a whole is shown in Figure 3 as a Kronecker product of the component matrices.

The Kronecker (or tensor) product [Eisele and Mason, 1970] is suitable because of the difference among the matrix dimension.

The result of the product, illustrated in figure 4, is also a boolean matrix named, here, *transition descriptor.*

$$\begin{array}{l} W_1B0W_2 \\ W_1B0F_2 \\ W_1B1W_2 \\ W_1B1T_2 \\ W_1B2W_2 \\ W_1B2T_2 \\ T_1B0W_2 \\ T_1B0F_2 \\ T_1B1W_2 \\ T_1B1F_2 \\ T_1B2W_2 \\ T_1B2F_2 \end{array} \begin{bmatrix} & & 1 & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 1 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 1 & 0 & 0 \\ & & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline & & & & & & & \\ & & & & & & & \\ & 0 & & & 0 & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \end{bmatrix}$$

Figure 4. (W_1,*a_1 [not in B.2]*,T_1) transition descriptor

The lines and columns of the descriptor represents a configuration, the state of the system state as a whole, as (W_1, B.0, W_2), (W_1, B.0, T_2), etc.. and the elements may value 1 whenever there are a transition from the configuration in line to the configuration in the column and 0 otherwise.

$$D(a_1[NotInB_2]) = \overset{M_1}{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}} \otimes \overset{B}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}} \otimes \overset{M_2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}$$

Figure 3. Transition Descriptor of the synchronizing transition (W_1, a_1[not in B.2],T_1).

Reduced State Machine

The complete representation of the system behavior only considering the communication aspects, e.g., the synchronizing transitions, is obtained by adding the transition descriptors as a scalar matrix sum. The elements of the resulting matrix correspond to the transitions from configuration to configuration, so it represents the equivalent FSM for the unfolded Statecharts specification only considering the intercommunication features. An algorithmic conversion from a Statechart-based specification to a finite state machine is given in [Vijaykumar, 1999] and [Vijaykumar et al, 2002]. However, the presented solution through transition matrix appears to be more precise and one may find more efficient algorithms to store and tour the specification as they are highly sparse matrices [Fernandes, 1998] [Ciardo, 2001].

To automatically generate test case from the resulting FSM comprising the synchronization components of the protocol Statecharts-based specification, it is intended to use the Condata. This available tool is able to extract test cases based on the tour transition method.

4. Comments and Future Plans

A strategy to automatically generate test cases to a Statecharts-based specification considering the synchronization aspects of the system behavior is presented. In fact, the proposed approach needs to be tested. There are still some problems to be solve while generating test cases with ConData from the resulting matrix with the synchronizing transitions. Some of them are: to guarantee that the resulting machine is a strongly connected graph and that the initial state belongs to the resulting machine. Moreover, it is clear that some configurations will not be a part of the resulting matrix, in other words, the behavior of the protocol specification is a partial one, so investigations need to be done in order to evaluate the efficiency of the test case suite generated by the proposed method.

In order to evaluated the efficiency of the test case suite generated to the reduced machine, the following experiment will be taken: (i) the example of the Statecharts specification of Producer x Consumer will be implemented in C; (ii) generate a set of interface mutants based on the work of [Delamaro, 1997], (iii) generate the tests based on the complete specification; (iv) generate the tests based on the reduced machine; (v) apply the test of the (iv) and (iii) over the mutants; (vi) compare the power of fault detection of the tests generated in (iv).

References

Bogdanov, K. ; Holcombe, M.; Singh, H. Automated Test Set Generation for Statecharts. Lectures Notes in Computer Science 1641 (1999) 107-121

[Bochmann et al, 1994] Bochmann, G.; Petrenko, A . – Protocol Testing: Review of Methods and Relevance for Software Testing – Proceedings of the 1994 International Symposium on Software Testing and Analysis, p.109-124, August 17-19, 1994 – Seatle, Washington, USA.

Bourhfir, C.; Dssouli, R.; Aboulhamid, El M.; Rico, N. Automatic Executable Test Case Generation for Extended Finite State Machine Protocols. IFIP International Workshop on Testing Communicating Systems, Korea, 1997.

Ciardo, G - What a Structural World – invited paper of Aachen International Multiconference on Measurement, Modeling and Evaluation of Computer-Communication Systems, September, 2001.

Delamaro, M.E. – Mutação de Interface: Critério de Adequação Interprocedural para o Teste de Integração – Ph D. Thesis – Instituto de Física de São Carlos - Universidade de São Paulo, 1997.

Dssouli, H.; Salek, K.; Aboulhamid, E ; En-Nouaary, A ; Bourhfir, C - Test Development for Communication Protocols: Towards Automation. Computer Networks 31, 1999 1835-1872.

Eisele, J.A ; Mason, R.M. – Applied Matrix and Tensor Analysis – John Wiley & Sons, 1970.

Fabri, SCPF; Maldonado, J.C.; Delamaro, M.E.; Masieiro P.C. Mutation Testing Applied to Validate Specification based on Statecharts. Proceedings of the Tenth International Symposium on Software Reliability Engineering, Flórida 1999, 210-219.

Fernades, P; Plateau, B; Steart, W. - Efficient Descriptor-Vector Multiplications in Stochastic Automata Networks – Journal of ACM, v. 45, n. 3 May 1998, 381-414.

Harel, D. Statecharts: a visual formalism for complex systems. Science of Computer Programming, 8, (1987) 231-274

Harel, D.; Politi, M. – Modeling Reatctive Systems with Statecharts – The Statemate Approach – MacGraw-Hill, 1998

Hong, H.S.; Lee, I.; Sokolsky, O.; Cha, S.D. - ``Automatic Test Generation from Statecharts Using Model Checking,'' Proceedings of the First Workshop on Formal Approaches to Testing of Software (FATES '01), pp. 15-30, Aalborg, Denmark, Aug. 2001.

Huzar, Z.; Magott, J. – New semantics for Markovian Statecharts – http;//www.dur.ac.uk/nigel.thomas/UKPEW2000/ online-proceedings.html

Kim, Y.G.; Hong, H.S.; Cho, S.M.; Bae, D.H.; Cha S.D. – Test Case Generation from UML State Diagrams – IEE Proceedings software, V. 146, N. 4, , Aug. 1999, 187-192.

Lai, R. – A survey of communication protocol testing – The Journal of Systems and Software, 62, 2002, pp. 21-46.

Lee, D.; Yannakakis, M. Principles and Methods of Testing Finite State Machines – a Survey – Proceedings IEEE, 84 (8): 1090-1123. 1996.

Martins, E. Sabião, S.B.; Ambrosio, A. M. - ConData: a Tool for Automating Sapecification-based Test Case Generation for Communication Systems. Software Quality Journal, 8 (4) (1999) 303-319.

Sabião, S. B. A Method for Test Case Generation based on Extended Finite State Machines combining Black-Box Testing Techniques. Master thesis. Institute of Computing, State University of Campinas, Brazil, 1999.

Tan, Q.M.; Petrenko, A.; Bochmann, G. A Test Generation Tool for Specifications in the Form of State Machines. Proceedings of the International Communications Conference ICC 96, Texas, 1996, 225-229

Tretmans, J.; Belinfante, A. – Automatic Testing with Formal Methods – In Proceedings of the Conference on Software Testing, Analysis and Review. EuroSTAR´99, November, 1999.

Vijaykumar, N. L. Statecharts: Their Use in Specifying and Dealing with Performance Models. Ph.D. Thesis – Aeronautics Inst. of Technology (ITA). S. J. Campos, Brazil, 1999.

Vijaykumar, N. L.; Carvalho, S. V.; Abdurahiman, V. On proposing Statecharts to specify Performance Models. International Transactions in Operational Research, 9(3), (2002) 321-336.