The DEVS Formalism : A Framework for Logical Analysis and Performance Evaluation for Discrete Event Systems *

Gyung Pyo Hong and Tag Gon Kim
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology
373-1 Kusung-dong Yusung-gu, Taejon 305-701, Korea.

Abstract

This paper proposes a framework which supports performance evaluation and logical analysis of discrete event systems using a unified formalism, i.e., the DEVS(Discrete Event System Specification) formalism. For performance evaluation, DEVSim++, a realization of the DEVS formalism and the associated simulation algorithms in C++, is used. For logical analysis, the dual language approach is adopted. We use the DEVS formalism as an operational formalism to describe system's behavior. Temporal Logic(TL) is employed as an assertional formalism to specify system's properties. To reduce states space in logical analysis, we exploit a projection mechanism. The method is a mapping of a set of states in models into a state which obtained from TL assertions. An example of logical analysis for Alternating Bit Protocol is given.

1 Introduction

Systems development process consists of a formal specification of requirements, modeling from the specification, validation of the models, performance evaluation and implementation. A model for performance evaluation should have time informations between the communicating entities to analyze average delay time, throughput, and so on. On the other hand, a model for validation should have logical informations to prove that there are no logical conflicts in procedure rules. From these different features, the designer has to develop two kinds of models to perform the performance evaluation and the logical analysis. This is a very tedious job.

This paper presents a framework which supports both performance evaluation and logical analysis within a unified formalism. Logical analysis techniques can be divided into two approachs, *single language approach* and *dual language approach*. Reachability analysis is a well known single language approach. It is practically impossible to perform complete analysis for complex systems[2]. On the other hand, the dual language approach uses two formalisms: *operational formalism* which describes the behavior of a system and *assertional formalism* which specifies the property of a system. We adopt Temporal Logic(TL) as an assertional language and DEVS formalism as a description language.

This paper is organized as follows. Section 2 describes the proposed framework for the dual language approach. In section 3, we describe the assertional language TL and its expansion procedure. And a projection mechanism for atomic DEVS models and a validation procedure are also described. We conclude this paper in section 4.

2 Proposed Framework

Figure 1 proposes a framework for logical analysis and performance evaluation within the unified DEVS formalism, where the logical analysis exploits the dual language approach. A modeler should develop DEVS models to perform performance evaluation. This is refined descriptions from informal requirements of a system. The DEVS formalism and the associated simulation algorithms provide sound modeling semantics and a simulation methodology[8]. Therefore the modeling and the performance evaluation processes are easily accomplished by using the DEVSim++[3].

*ISBN 0-8186-6440-1.

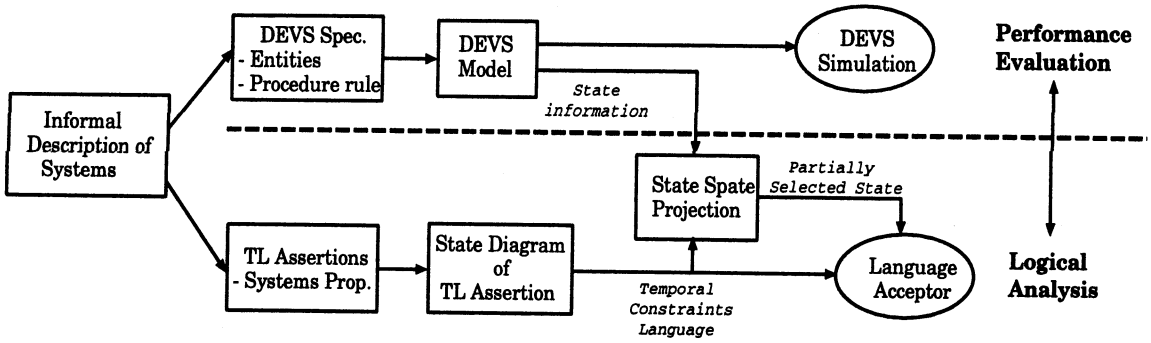


Figure 1: Overall System Configuration

The main advantage of the dual language approach is in its flexibility; the use of specification language provides a uniform notation for expressing a wide variety of correctness properties and it separate models from reachability assertions[6]. But the dual language approach also has the state explosion problem for complex systems. Therefore partial proof against given specifications is a reasonable solution to these problems[4]. The dual language approach with a projection mechanism can be an efficient method for the validation of large systems.

Temporal constraints that present temporal properties of the requirements are also required for logical analysis. These constraints are expressed by using Temporal Logic(TL). The temporal logic formula is translated into a finite state automaton. The state informations are obtained from the automata. These informations are applied to the DEVS model, developed for performance evaluation, to obtain a projected state space. Logical analysis is performed by using the projected state space and the finite automata of TL formulas.

3 Logical Analysis

The DEVS models for performance evaluation does not have any constraints about the desired states/events set and global state information of a system. For logical analysis, such information should be added to the DEVS models. The logical constraints which specify the state sequences of a system can be expressed in terms of temporal logic formulas. Timing information does not need for logical analysis. Thus, the time advance function fro an atomic DEVS model may not be used for logical analysis. The logical analysis is basically a searching procedure to find illegal states. For efficient analysis, there should be a mechanism to reduce the state space. We accomplish it by an extension of the DEVS formalism, *i.e*, add a projection function and state set informations derived from TL assertions into atomic DEVS models. A facility for global states manipulation is also added to coupled DEVS models.

3.1 Expression of Temporal Logic

Temporal logic assertions express sequence of states that should be satisfied during system execution. Therefore the state information of TL assertions can be used to project with respect to related states from the DEVS model of a target system. Temporal logic is an extended logic by adding the temporal operators to describe the timing relationship between entities. It has been widely used to specify discrete event systems such as concurrent system and communication protocol. The temporal operators and their meanings are as follows[1].

- $\Box(always)A$: A is true now and will always be true in the future.
- $\Diamond(sometimes)A$: A is true now or will be true sometimes in the future.
- $A \bigcup(until)B$: B is true now or A is true until B will be true.
- $\bigcirc(next)A$: A will be true next time.

To establish correspondence between TL assertions and a DEVS model, TL assertions are described by using the state variables and their values defined in the DEVS model. Let grammar G of the temporal expression be $\langle V_T, V_N, P, S\rangle$. Terminal V_T is a set of atomic formulas which contain no temporal operator. The non-terminal V_N follows the next operator $\bigcirc$. A given temporal expression, a non-terminal set and a

FINAL state constitute a states set S of the grammar. The production rules P for a set of temporal operators are as follows, where an expression in { } is a language accepted by the projection rules.

- $\Box A \Longrightarrow A \cdot \bigcirc(\Box A)\{A^*\}$
- $\Diamond A \Longrightarrow A \mid \neg A \cdot \bigcirc(\Diamond A)\{(\neg A)^* A\}$
- $A \bigcup B \Longrightarrow B \mid A \wedge \neg B \cdot \bigcirc(A \bigcup B))\{(A \wedge \neg B)^* D\}$
- $\neg\Box A \Longrightarrow \neg A \mid \bigcirc(\neg\Box A)\{(\neg A)^*\}$
- $\neg\Diamond A \Longrightarrow \neg A \mid A \cdot \bigcirc(\neg\Box A)\{A^*\neg A\}$
- $\neg(A \bigcup B) \Longrightarrow$
  $\neg B \mid B \wedge \neg A \cdot \bigcirc(\neg(A \bigcup B))\{(B \wedge \neg A)^*\neg B\}$

A temporal expression which describes a sequence of states is expanded to a current state condition and a next state condition using the grammar. This is based on the decision procedure in [7]. The expansion procedure for temporal expression is as follows.

(i) Start with application of the production rules shown above to TL assertions.

(ii) Set the non-terminal to a next state and the terminal to a transition condition. Any formula that contains only terminals becomes a transition condition for the **FINAL** state.

(iii) The outmost operator $\bigcirc$ for a next state is removed.

(iv) If a new state does not appears, then terminate. Otherwise go to (i).

After execution of the expansion procedure, a TL assertion is translated into a finite state automaton **R** as

$$\mathbf{R} = < \mathbf{S}, \mathbf{A}, \delta_{\mathbf{R}} >$$

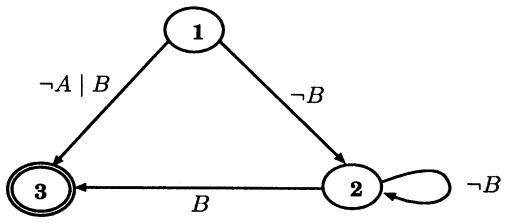
S : sequential states set;
A : logical assertions set;
δ : state transition function;
with the following constraints
S, A : finite set;
$\delta_R : S \times 2^A \rightarrow 2^S$;

In this paper, we use the alternating bit protocol(ABP) as an example system. A detailed description of the ABP appears in [5]. Consider the following property for the ABP: *" Error_free Transmission : If no transmission errors exist between Sender and Receiver, then messages are sent infinitely and they have alternate control bit"*. The TL assertions of this property are as follows.

(i) $\Box(S.Error \wedge R.Error = false)$
(ii) $\Box((S.Phase = FM \wedge S.st = 0)$
$\rightarrow \Diamond(R.Pahse = FA \wedge R.at = 0))$
(iii) $\Box((S.Phase = FM \wedge S.st = 1)$
$\rightarrow \Diamond(R.Pahse = FA \wedge R.at = 1))$
(iv) $\Box((S.Phase = FM \wedge R.Phase = WM)$
$\rightarrow (S.Phase = FM \wedge R.Phase = WM)$
$\bigcup (S.Phase = WA \wedge R.Phase = FA))$
(v) $\Box((S.Phase = WA \wedge R.Phase = FA)$
$\rightarrow (S.Phase = WA \wedge R.Phase = FA)$
$\bigcup (S.Phase = FM \wedge R.Phase = WM))$

The model of the ABP should be correct if it satisfies the above TL assertions. To prove the property, we translate TL assertions into a finite state automaton. Figure 2 shows the resultant automata for the TL assertion (*iii*). The expansion procedure for this assertion is as follows. Let $S.Phase = FM \wedge S.at = 1$ be A, and $R.Phase = FA \wedge R.at = 1$ be B. The following is the expansion procedure for the given property.

1st step : apply expansion rule for $\Diamond B$
$\neg A \mid \Diamond B = \neg A \mid B \mid \neg B \cdot \bigcirc(\Diamond B)$
2nd step : determine transition conditions
transition to **FINAL** state : $\neg A \mid B$
transition to next state($\Diamond B$) : $\neg B$
3th step : apply expansion rules to $\Diamond B$
$\Diamond B = B \mid \neg B \cdot \bigcirc(\Diamond B)$
4th step : determine transition conditions
transition to **FINAL** state : B
transition to next state($\Diamond B$) : $\neg B$
5th step : ($\Diamond B$) appears again : stop expansion



$A = (S.Phase = FM \wedge S.st = 1)$
$B = (R.Phase = FA \wedge R.at = 1)$

Figure 2: Finite State Automaton

The state values set which is related to the given properties can be extracted from the transition condition $\mathbf{2^A}$ of **R**. This is done by the grouping the values set used as transition conditions. The state variables of **SENDER** and **RECEIVER** for the

above property can be grouped as follows: $S.Phase = \{\{WA\}\}$, $S.st = \{\{0\}\}$, $S.Error = \{\ \}$, $R.Phase = \{FA, \{WA\}\}$, $R.at = \{1, \{0\}\}$ and $R.Error = \{\ \}$. Therefore, states that have the same values except for *Error* can be treated as an equivalent state. Table 1 represents the values set of state variables of the ABP ofr the *Error_free Transmission* property.

Spec #	sv	SENDER	RECEIVER
1	Error	false	false
2	Phase	{WA}	FA, {WM}
	st	{1}	—
	at	—	0, {1}
3	Phase	{WA}	FA, {WM}
	st	{0}	—
	at	—	1, {0}
4	Phase	{WA}, WA, FM	FA, {FA}, WM
5	Phase	FM, {FM}, WA	{WA}, WM, FA

Table 1. Grouped state variables for TL assertions

3.2 Projection Procedure

The projection is an efficient method to reduce the space/time complexity of logical analysis. Assume that a model supports various properties and some of them are disjoint. Then TL assertions for a certain property use only a subset of domain of a state variable. Therefore the states which have the unused values can be grouped in one state.

Definition. 1 *Let $M(S_i)$ be a state variable of an atomic DEVS model and $M(V_i)$ be domain of $M(S_i)$. Then the set of states of the model is $S_M = M(V_1) \times M(V_2) \times \cdots M(V_n)$.*

Definition. 2 *Let $R(S_i)$ be a state variable that is used as transition conditions in the automata* **R** *and $R(V_i)$ be a set of grouped value set. Then the set of states of requirements is $S_R = R(V_1) \times R(V_2) \times \cdots R(V_n)$.*

Definition. 3 *Projection is a mapping of a set of states in S_M into a state in S_R based on $R(V_s)$.*
(i) The states in S_M which contain a value in the used value set $\bigcup_{1 \le i} R(V_i)$ is aggregated into a state in S_R.
(ii) The states in S_M which contain values that does not appear in the used value set $\bigcup_{1 \le i} R(V_i)$ are removed from S_M. The resultant isolated states are also removed.
(iii) If i state variables do not used in the assertions, then the n dimensional state space is mapped into the $n - i$ dimensional image.

Consider a system shown in Figure 3 (a) that is described by 2 state variables $v_1 = \{true, false\}$, $v_2 = \{n \mid n \ge 1\}$. Let states s_2, s_3 and s_4 be $s_2 = \{v_1 = false, v_2 = 2\}$, $s_3 = \{v_1 = false, v_2 = 3\}$ and $s_3 = \{v_1 = false, v_2 = 3\}$. Assume that state transition conditions which are obtained from the expansion procedure have a value group $v_2 = \{1, \{2, 3, 4\}\}$. If a transition condition for TL assertions satisfies $v_1 = false$, then the system can transit to these states. Therefore these states are equivalent and can be grouped. Figure 3 (b) shows the results after grouping.

If TL assertions do not use the value $v_1 = false$, then the system never transit to the states that contain this value. So, these states can be removed from the original system. Figure 3 (c) shows the result after removing those states.

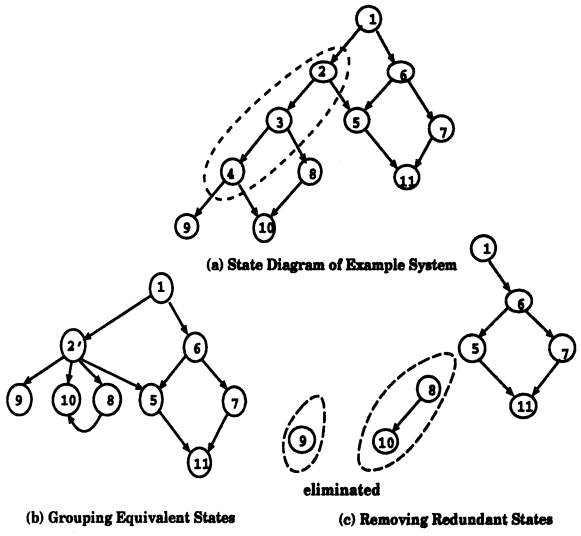


Figure 3: Projection of Example System

The DEVS formalism is extended for logical analysis and projection. The projection function is a mapping of states from an atomic DEVS model to a state in a finite state automaton of a TL assertion. S_R is the states set that is equivalent to the grouped states of the DEVS model. Formally, the specification of an atomic mode M is as follows.

$$\mathbf{M} = < \mathbf{X}, \mathbf{S_M}, \mathbf{Y}, \delta_{\mathbf{int}}, \delta_{\mathbf{ext}}, \lambda, \mathbf{ta}, \mathbf{f_R}, \mathbf{S_R} >$$

X : input events set;
S_M : sequential states set;
Y : output events set;
δ_{int} : internal transition function;
δ_{ext} : external transition function;

λ : output function;
ta : time advance function;
f_R : projection function for requitements;
S_R : states set of requirements;
with the following constraints,
X, Y, S_M : infinite but countable set;
$\delta_{int} : S_M \rightarrow S_M$;
$\delta_{ext} : Q \times X \rightarrow S_M$;
$Q = \{(s, e) \mid s \in S_M, 0 \leq e \leq ta(s)\}$;
Q: total state of M,
e: elapsed time after scheduling;
$\lambda : S_M \rightarrow Y$;
$ta : S_M \rightarrow Real$;
$f_R : 2^{S_M} \rightarrow S_R$;

To validate, the gathering and tracking facilities of global states of the coupled DEVS model are required. So a means for manipulating the global states set is added to the coupled DEVS formalism.

$$\mathbf{DN} = < \mathbf{D}, \{\mathbf{M_i}\}, \{\mathbf{I_i}\}, \{\mathbf{Z_{i,j}}\}, \mathbf{SELECT}, \mathbf{S_G}, \delta_{\mathbf{DN}} >$$

D : component names set;
for each i in D,
M_i : DEVS component i in D;
I_i : set of influencees of i;
for each j in I_i,
$Z_{i,j} : Y_i \rightarrow X_j$
: i-to-j output translation function;
$SELECT : 2^D \rightarrow D$: tie-breaking selector;
$S_G : \times S_{R_i}$: global states set;
$\delta_{DN} : S_G \times 2^A \rightarrow 2^{S_G}$: state transition function;

Figure 4 (a) shows state diagrams of the atomic models SENDER and RECEIVER. The global state diagram for the projected atomic models is shown in (b). The dotted area presents the projected states with respect to the property *Error_free Transmission*. The global state diagram obtained from the projected atomic models is equivalent to the projection result of the original coupled model. Therefore application of the projection on the atomic DEVS model is more efficient because the state space complexity is reduced.

3.3 Validation Procedure

The logical analysis is performed by an acceptance checking : check whether a TL assertion accepts the state transition sequences of a coupled DEVS model. If the model is correct, then a sequence of states in a

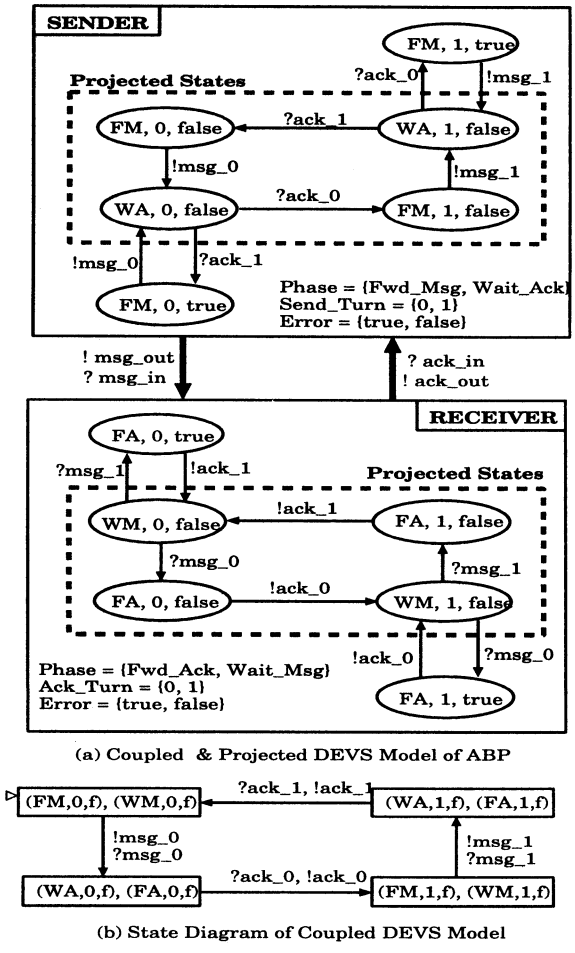


Figure 4: Global State Diagram of Projected ABP

cycle of the model would be accepted by the temporal constraints language. The validation algorithm is as follows.

Validation Algorithm

Var
Stack: stack of G;
g_i *: global states set :* $S_i \times S_{G_i}$;
S_n *: next states set of FSA;*
t_n *: transition condition of FSA;*
S_{G_n} *: next states set of MODEL;*

begin
Stack := { };
$g_i := g_0$;
push(g_0, *Stack*);
while not_empty(Stack) do begin
$S_n := \delta(S_i, t_i)$;
$S_{G_n} := \delta_{DN}(S_{G_i}, t_i)$;

```
        if S_{G_n} = { } then
            go to Label;
        g_n := S_n × S_{G_n};
        for ∀g_n do
            if (g_n ∉ Stack) then
                push(g_n, Stack);
            else if acceptance_check(g_n) = true then
                terminate(model is correct);
        end for;
        Label : g_i = pop(Stack);
    end while;
terminate(conflict exist);
end;
```

By using the above algorithm, we can find a loop from the transitions of a FSA(Finite State Automaton) and a DEVS model in the Figures 2 and 4. The possible next states $g_1([2,((WA,0,f),(FA,0,f))])$ and $g_2([3,((WA,0,f),(FA,0,f))])$ are obtained from the initial $state_0([1,((FM,0,f),(WM,0,f))])$. Next state of g_2 becomes $g_3([1,((WA,0,f),(FA,0,f))])$ although g_2 is an acceptance state. This is because a cycle of sequences of global states does not exists. And g_1 transit into $g_4([2,((WA,0,f),(FA,0,f))])$ because next state of the model can accept the transition condition of the FSA. The state transition of the model based on transition condition of the FSA is made until a cycle of state sequences of the model is detected and the FSA reaches the acceptance state. If a model has a deadlock, then it can not proceed at that state. Therefore the model can not reach the acceptance state until the algorithm is terminated. The validation algorithm can also be applied to the negation of a TL assertion. Such negation of a given TL assertion may increase validation speed for some cases.

4 Conclusion

This paper presents a unified framework for performance evaluation and logical analysis within the DEVS formalism. A performance analysis model has timing informations between the communicating entities. That is used to analyze time related performance such as average delay time and throughput, *etc.* A validation model has logical informations which is used to prove systems properties or procedure rules. This is accomplished by an extension of the DEVS formalism.

To solve the state space explosion problem during the validation phase, we exploit a projection mechanism using external TL assertions. This is a very efficient method because state reduction is made before the atomic models are coupled. Then various validation techniques which are used in the dual language approach can be applied.

Acknowledgement

The authors would like to thank ETRI(Electronics and Telecommunications Research Institute) for supporting this research. This research was done in the context of the ETRI project on ATM network performance study.

References

[1] Reinhard Gotzhein, " Temporal logic and applications - a tutorial," *Computer Networks and ISDN Systems 24*, 1992.

[2] Gerard J. Holzmann, *Design and Validation of Computer Protocols*, Prentice-Hall, Inc., 1991.

[3] Tag G. Kim, "DEVSIM++ User's Manual: C++ Based Simulation with Hierarchical Modular DEVS Models", Computer Engineering Lab., Dept. of Electrical Engineering, KAIST, 1994.

[4] Simon S. Lam, A. Udaya Shankar, "Protocol Verification via Projections", *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 4, July 1984,

[5] Jawahar Malhotra, Scott A. Smolka, Alessandro Giacalone, Robert Shapiro, "Winston : A Tool for Hierarchical Design and Simulation of Concurrent Systems, "

[6] Jonathan S. Ostroff, *Temporal Logic for Real-Time Systems, Advanced Software Development Series. 1*, Research Studies Press Ltd., 1989.

[7] Pierre Wolper, "Temporal logic can be more expressive," *22nd Annual Symposium on Foundation of Computer Science*, pp.340-348, 1981.

[8] B.P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation:* Academic Press, Orlando, FL, 1984.