Ordering Method for Reducing State Space in Compositional Verification

Wan Bok Lee and Tag Gon Kim

Department of Electrical Engineering,
Korea Advanced Institute of Science and Technology (KAIST),
373-1 Kusong-dong, Yusong-gu, Taejon 305-701, Korea
{wblee,tkim}@smslab.kaist.ac.kr

ABSTRACT

This paper proposes an efficient ordering method for solving the state explosion problem in compositional verification of discrete event systems. A system to be verified is specified by untimed DEVS (Discrete EVent system Specification) formalism. The size of the intermediate state space during the compositional verification is dependent on the sequence of composing system components. Application of this method to some classical verification problems has shown that the size of the intermediate state space during composition is markedly reduced.

1 INTRODUCTION

Man-made dynamic systems are no longer described or analyzed by the ordinary or partial differential equations. Examples of such systems are computer/communication networks, manufacturing systems and various forms of plant-controller systems. These systems might have safety critical abnormalities such as deadlock, race condition or violation of the mutual exclusive access that might result in an enormous damages or even to death. So the importance of proving the functional correctness of these systems arises and several models have been proposed to analyze such abnormalities.

Verification is a confirming process to check whether the implementation meets the specification or not. It is an indispensable process for high quality and reliable system development. An analysis technique is called compositional if the results of analyzing subsystems can be merged together to obtain the analysis results for the complete system. Thus the compositional verification method, which consists of a sequence of successive composition and concealment of the internal local events, could be a promising approach to attack the state explosion problem. Hence the size of the intermediate state space is dependent on the order of composing system components, a good ordering method is required to fullfil a fast and efficient verfication process.

The proposed method finds a good composition ordering by which the intermediate state space can be significantly reduced. Our ordering method is based on two metrics which are independent: number of the internalized events between the components and number of the observed events associated with a system property. Application of the method to some classical verification problems has shown that the size of the intermediate state space during composition was markedly reduced.

This paper is organized as follows. In Section 2, we describe our verification framework including the models of computation and related operations. Section 3 presents our composition ordering method and shows the experimental results. Finally we conclude this paper in Section 4.

2 CDEVS COMPOSITIONAL VERIFICATION FRAMEWORK

In this section, we introduce our compositional verification framework which is based on the new CDEVS formalism.

There are two elementary models in the verification process. One is an operational model (or behavior model) which describes the behavior of a system. The other is an assertional model (or property model) which specifies a property of a system. As described in Figure 1, both the models are represented by the CDEVS formalism in our framework.

The CDEVS model, newly proposed one for the analysis of DEVS models, is slightly augmented from the DEVS[8] to support following two point of necessity which are inevitable for our verification framework. The one is the capability for nondeterministic representation of a system behavior. The repetitive composition and minimization operations in our framework converts a deterministic model into a nondeterministic one. This fact was the basic motive to propose the CDEVS formalism. The other is for the closureness property under two operation, i.e. composition and minimization. The CDEVS formalism together with the two operations provides a systematic and sound means for our compositional verification framework

To show that a given specification is satisfied by the implementation model, it is usually sufficient to consider an abstraction of the global state space, because much of the internal communication and interactions are irrelevant from the observer's point of view. This abstraction may lead us to reduce the state graph drastically small compared with that of the original one by collapsing semantically equivalent states into a single representative state.

0-7803-5731-0/99/$10.00 ©1999 IEEE

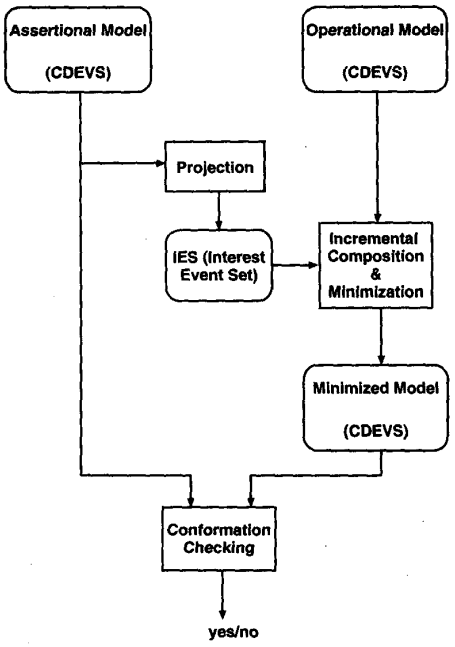


Figure 1: CDEVS Compositional Verification Framework.

It is called an interest event that is important and should not be projected out during the successive composition and minimization processes. The set of interest events are said to be Interest Events Set (IES). IES is constructed from the assertional model and both the composition and the minimization operations are dependent on it. For example, let's assume we want to show a philosopher could always eat the spaghetti whenever he tries in the classical dining philosopher problem. It is sufficient to reduce the model with respect to the events of his own while projecting out those of the others and then to verify the property is satisfied at the reduced model.

2.1 Models of Computation

Models developed in a DEVS formalism could be automatically transformed into a CDEVS models. Functional correctness are verified within the CDEVS framework while the performance issues related with time could be treated using the simulation packages such as DEVSim++[4], which is a realization of the DEVS formalism and its associated abstract simulator algorithm in C++.

The most outstanding feature of the DEVS formalism is the specification of discrete event systems in a hierarchical and modular form. Two types of models called atomic model and coupled model are specified within the DEVS formalism. Atomic model, which is not broken into a smaller one, specifies the behavioral aspect of an entity. Coupled model specifies the connectivity relation among the models, thus to form a new model using the predeveloped ones.

Because the CDEVS is devised for the purpose of system analysis, it need not have hierarchical coupling information. The coupling information among the models is flattened and transformed as the Coupling Relation (CR), which specifies the coupling information among the CDEVS components. The coupling relation CR is of the form.

$$CR \subseteq M \times Y_i \times M \times X_j$$

where

M: component models set,
X_i: input events set of the ith component,
Y_i: output events set of the ith component

An element of the coupling relation (m_1, y_1, m_2, x_2) means the connection from an outut event y_1 of a component m_1 to the input event x_2 of a component m_2 There are two assumptions for the coupling relation. One is that an input event comes from only one source; that is, an input event is uniquely connected to one output event of another process. While, an output event of a process can be connected to multiple of input events of processes. The other is that a self-loop connection is not allowed; i.e., an output of a process can not be coupled with an input event of the process itself.

The CDEVS formalism which specifies the behavior of each component models is as follows The internal transition funciton δ_{int} together with the output transition function λ of the DEVS formalism are substituted as internal transition relation denoted by T_{int}. τ is an internal hidden transition changing the current state to another state invisibly.

Definition 1 (CDEVS)

$$CDEVS = < X, Y, Q, \delta_{ext}, T_{int} >$$

where

X: *input events set,*
Y: *output events set,*
Q: *composite states set,*
$\delta_{ext} : Q \times X \longrightarrow Q$, *external transition function,*
$T_{int} \subseteq Q \times (Y \cup \{\tau\}) \times Q$, *internal transition relation,*
where $*$ *means no external output events happens.*

2.2 Communicating Semantics

Various formalisms have been developed and studied for the modeling, analysis and synthesis of discrete event systems. Notable among them, the concurrency theory including CCS[5], CSP[3] *et al.* is one of the successful. In their context the interaction between the processes and the environment is modeled by parallel composition with a specified degree of event synchronization. Heymann has classified the interactions between the processes into three categories: strict

synchronization, broadcast synchronization and prioritized synchronization[2].

Strict synchronization restricts the shared events to be either executed by both processes concurrently or by none. Under the broadcast synchronization each process can generate their events for execution and the other process will participate in their execution synchronoulsy if it can. But if it cannot, the initiating process execute the event alone. For prioritized synchronization refer to [2].

Most of the formalisms that have been defined and investigated in the literature rely on some framework of strict synchronization. In their framework, an event with the same label of distinct processes must either happen altogether or be disabled waiting all the other processes to be ready to execute it. This restriction makes those formalisms inherently inadequate for modeling the interaction of discrete event processes in which the spontaneity is an essential behavioral feature. For example broadcasts to the world (as in Ethernet protocols) are not easily modeled by the formalisms based on strict synchronization[6].

In our CDEVS model which is based on set-based system theory, events are classified into three classes: input, output and internal. Every input event is associated with its causal output as specified in the coupling relation. It occurs in a passive form, intiated by an output of the other process. Processes can spontaneously generate either an output or an internal event. No process could block the generation of an internal or an output event of other processeses. A process has only two choices; either accept the generated output event as its input event and transiting to other state if it can or simply stay at the same state not interfering the execution of it. Therefore the interaction between our CDEVS models conform to the broadcast synchronization of the Heymann's classification.

2.3 Operators for Analysis

When two or more components are interconnected and executed, its behavior is quite complex than that of the single component. To prove the functional correctness of a system we need to check all the behaviors generated by the composite model. But if we are interested in only on the sequence of events which are sufficient to prove our property named interest events, it is possible to keep the size of the intermediate state space to be much smaller than that of the conventional composition method.

Consecutive composition and minimization yield a compact CDEVS model which is observationally equivalent with the original system which consists of several components.

2.3.1 Composition: Two CDEVS models can be composed to yield another CDEVS model which exhibits the same behavior. It is a process of constructing a new model from the given two models employing our communicating semantics, broadcast synchronization.

Composition is defined as follows

Definition 2 (Composition ($M_1 \|_{CR} M_2$))
The composition operates on two CDEVS models, M_1, M_2 and on coupling relation CR, constructing a new CDEVS as

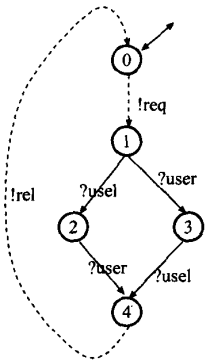
$$CDEVS = < X, Y, Q, \delta_{ext}, T_{int} >$$

where

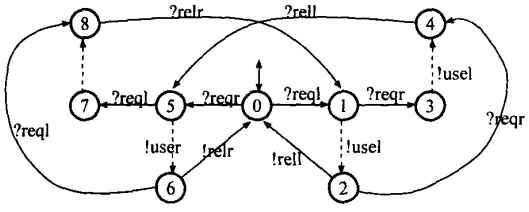
$X \subseteq \bigcup_{i \neq 1,2} M_i.Y$
$Y \subseteq M_1.Y \cup M_2.Y$
$Q \subseteq M_1.Q \times M_2.Q$
δ_{ext}: $Q \times X \rightarrow Q$
$T_{int} \subseteq Q \times (Y \cup \{\tau\}) \times Q$

Two component models in Figure 2 are composed to form a new model shown in Figure 3(b). The coupling relation used in the composition is specified in Figure 3(a).

The line between two nodes means an external transition and the dotted line denotes an internal transition. "!", "?", "*" mark specify the type of an event, i.e. output, input, and internal event, respectively.



a) CDEVS model: phil



b) CDEVS model: fork

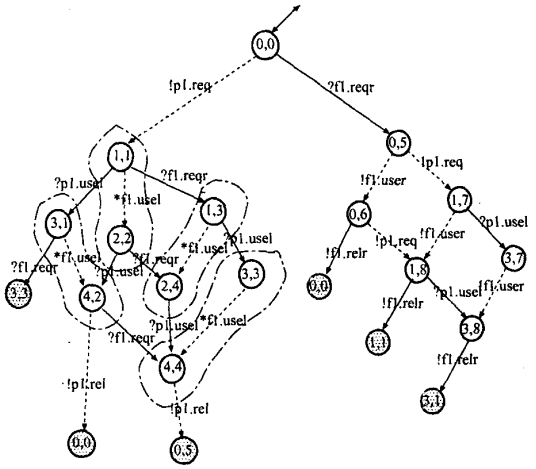
Figure 2: component model: phil, fork

A transition which is synchronized between the two components and do not influence other models is renamed as an internal hidden transition. The event "*f1.usel" at the composed model is such one. An output "usel" of the component "f1" does not influece

other compoents except the component "p1". Thus it is remarked as an internal event in the composed model.

CR = {(p1, req, f1, reql), (p1, req, f3, reqr), (p1, rel, f1, rell), (p1, rel, f3, relr), (p2, req, f2, reql), (p2, req, f1, reqr), (p2, rel, f2, rell), (p2, rel, f1, relr), (p3, req, f3, reql), (p3, req, f2, reqr), (p3, rel, f3, rell), (p3, rel, f2, relr), (f1, usel, p1, user), (f1, user, p2, usel), (f2, usel, p2, user), (f2, user, p3, usel), (f3, usel, p3, user), (f3, user, f1, usel)}

a) coupling relation



b) composed model

Figure 3: coupling relation and the composed model: pf

Composition also deletes the coupling relations related with the models used in the composition and adds new coupling relations which specify the coupling between the newly constructed model and the other components. An event which belongs to the IES should not be internalized at all during the composition, even if it could be. It is still marked as an output event. Therefore its occurrence is always observed. The more internal transitions a model has, the more reduction is possible at the minimization operation. The successive processes involving both the composition and the minimization transform the overall system into a compact CDEVS model whose events set is the interest events set.

2.3.2 Minimization: Minimization is an operation to collapse the equivalent states into a representative state. The equivalent states exhibit the same observable behavior.

A notable fact in the minimization operation is that it should guarantee the behavior of the minimized model to comply with that of before minimization.

Definition 3 (Equivalence Relation($\approx$))
Two states s_1 and s_2 are equivalent denoted by $s_1 \approx s_2$ if and only if

1-1) whenever $s_1 \xrightarrow{!y} s_1'$,
for some s_2', $s_2 \xrightarrow{\tau}{}^{*} \xrightarrow{!y} s_2'$ *and* $s_1' \approx s_2'$

1-2) whenever $s_1 \xrightarrow{?x} s_1'$,
for some s_2', $s_2 \xrightarrow{?x} s_2'$ *and* $s_1' \approx s_2'$

1-3) whenever $s_1 \xrightarrow{\tau} s_1'$,
for some s_2', $s_2 \xrightarrow{\tau}{}^{*} s_2'$ *and* $s_1' \approx s_2'$

2-1) whenever $s_2 \xrightarrow{!y} s_2'$,
for some s_1', $s_1 \xrightarrow{\tau}{}^{*} \xrightarrow{!y} s_1'$ *and* $s_2' \approx s_1'$

2-2) whenever $s_2 \xrightarrow{?x} s_2'$,
for some s_1', $s_1 \xrightarrow{?x} s_1'$ *and* $s_2' \approx s_1'$

2-3) whenever $s_2 \xrightarrow{\tau} s_2'$,
for some s_1', $s_1 \xrightarrow{\tau}{}^{*} s_1'$ *and* $s_2' \approx s_1'$

The notation $\xrightarrow{\tau}{}^{*}$ in Definition 3 means that arbitrary number of internal transition τ might happen including zero time. The equivalence relation is reflexive, symmetric and transitive.

In Figure 3, nodes in the same dashdotted line are all equivalent sates. Here let's just consider the two state (3,3) and (4,4). The state (4,4) generate the output event "!p1.rel" before it reaches the state (0,5). The state (3,3) enters the state (4,4) but it has no interaction with any outside component models untill it reaches (4,4). Thus the coupling "*f1.usel" can be localized from the viewpoint of the outside model.

3 COMPOSITION ORDERING STRATEGY

The intermediate state space which is generated in the middle step of the incremental compositional process can be varied according to its composition ordering. In the dining philosopher problem, a philosopher has interactions only with its neighboring forks. Thus composing the philosophers all together, the forks altogether and then composing these two models results in large intermediate state space. Therefore the ordering in which the philosophers and the forks are composed first and then the neighboring pairs is preferable.

In this section, we suggest an composition ordering strategy which probably reduces the intermediate state space much smaller.

3.1 The Notion: Level of Interaction

Consider composing of the two model M_1 and M_2, in which the number of states are N_1 and N_2, respectively. The number of states of the composed model M can vary in the range from $max(N_1, N_2)$ at least to $N_1 \times N_2$ at most. The minimum size arises when two models are tightly coupled and synchronized with each other at each state. While the largest size happens in case the two models have no coupling relations at all just exibiting true independency. Thus we can derive a notion of the Level Of Interaction (LOI) like

follows. Where N is the number of states of the composed model of M_1 and M_2.

Definition 4 (LOI)

$$LOI = \frac{N - max(N_1, N_2)}{N_1 \times N_2 - max(N_1, N_2)}$$

The variable LOI is zero when there is no interations at all between M_1 and M_2, and one if they communicate in the most synchronized manner. Composing the models in which the value of LOI is high can result in much reduction of the intermediate state during the overall compositional process.

The following principles should be kept to increase the LOI.

1. Components which have many events that can be internalized are better to be composed first. A pair of events which have interactions only between the two models are remarked as an internal one after composition. Because the minimization operation can be more effective with many internalized event, the composed model are likely to be reduced much smaller if it has many internalied ones.

2. Symmetric and even composition is prefered. Composition tends to construct a new model which has larger states than that of before in most cases. A model which has experienced many composition operations charges higher cost than others. Earlier researchers[7] reported that symmentric composition with hierarchical compression techniue is very efficient.

3. The models which contain many interest events are better to be composed later. An event which belongs to IES is an observed one not reduced until the final step of the compositional process. Thus, in this case little state reduction is carried at the minimization operation.

3.2 Experimentation

Let's a composition ordering denoted by $O_k = c_1, \ldots, c_i, \ldots, c_n$ means that it consists of n composition steps and the i'th composition is c_i. Assume that c_i produces a composed model m_i after composing the two model $m_{i,1}$ and $m_{i,2}$.

Two heuristic functions were defined to decide the goodness of the ordering. These functions estimate the level of interaction. The larger is the value, the more reduced is the state space at the minimization operation.

They estimate the degree of the state reduction for a given ordering without conducting the actual composition and minimization operations.

$h1$ function measures LOI using the number of internalized events betweeen the components for a composition step. For a composition step it is defined as

$$h_1(c_i) = \frac{e(m_{i,1}) + e(m_{i,2}) - e(m_i)}{depth(c_i)}.$$

where $e(m)$ means the total number of input and output events, and $depth(c_i)$ is the depth of the composition. All models before any composition have depth 1. The composed model m_i after composition of the two model $m_{i,1}$ and $m_{i,2}$ has the depth $max(depth(m_{i,1}), depth(m_{i,2})) + 1$. Composing those models which have much interaction later results in larger composition depth indicating a bad ordering.

h_2 function was designed to measure the compliance of the third principle in previous subsection. It suggests that the models which contain many interest events should be composed later. The minus sign means that the smaller absolute value of h_2 denotes the better ordering.

$$h_2(c_i) = -\frac{ies(m_i)}{depth(c_i)}.$$

Where $ies(m)$ is the number of interest events.

Now H_1 and H_2 are defined as the metrics for a composition ordering.

$$H_1(O_k) = \sum_i h_1(c_i)$$

$$H_2(O_k) = \sum_i h_2(c_i)$$

Now, we are to show the effectivenss of our method by applying it to the dining philosopher example in which six philosophers and forks are engaged. Figure 2 shows the elementary component models used in our experiment. Our model is more complex than that of Ben-Ari[1]. The fork always permits the first philosopher to use itself if it has receives requests from both neighboring philosophers.

To show that a philosopher can always eat the spaghetti whenever he requests, two event, *req* and *rel* of the first philosopher model were selected as the interest events. Figure 4 shows the final result of the successive composition and minimization process. We can easily verify that the philosopher can always eat the spaghetti whenever he tries to eat by sending requests to forks in our model.

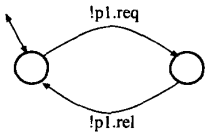


Figure 4: Finally Reduced Model.

Four composition orderings were compared and each ordering is shown in Figure 5. Table 1 shows the

values of H_1, H_2 functions together with the largest size of the intermediate state space generated during the successive composition and minimization process. H_1 function is more crucial than H_2 to select a good ordering. We recommend H_1 to be used as the primary metric and H_2 as secondary one.

The Ordering4 reveals the reckleness of composing the components in an arbitrary order. It was impossible to compute the final composed model after following the Ordering4 in our experiment. As the result shows our metric function could be effectively used to choose a good ordering without employing the time consuming composition and minimization operations in advance. The computing time spent during the process was also proportional to the size of the intermediate state space.

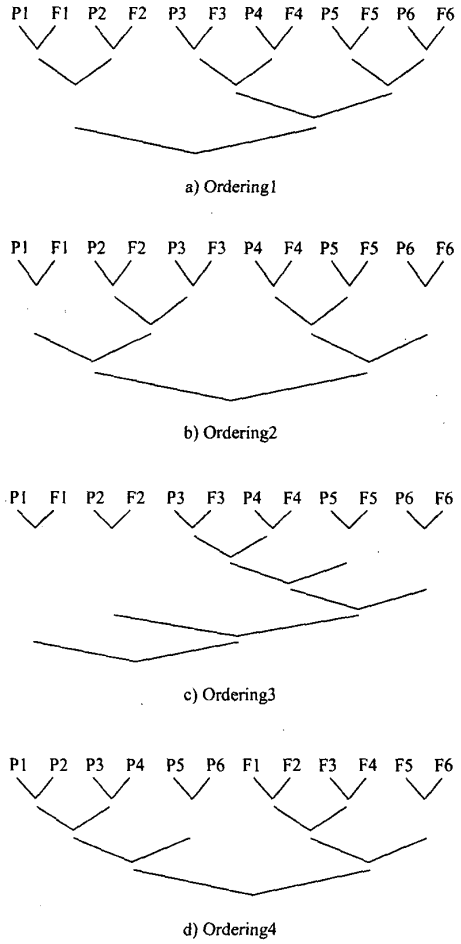


Figure 5: Four Composition Ordering.

Table 1: The Experiment Results

Ordering	Number of int. state	H_1	H_2
Ordering1	187	24.3	-2.7
Ordering2	121	21.3	-1.7
Ordering3	427	14.7	-2.9
Ordering4	more than 531441	0	-3.83

4 CONCLUSION

Our compositional framework is based on CDEVS formalism, a newly defined formalism for the analysis of the DEVS models. The DEVS formalism has been developed and investigated in the simulation areas. Our framework shows that the DEVS could be a unified formalism supporting both the performance evaluation and the logical analysis of discrete event systems.

In this paper an effiecient ordering method for compositional verification has been suggested. The intermediate state space during the stepwise composition and minimization process in the compositional framework is much dependent on the composition ordering. The experiment shows that our method could be effectively used to choose a good ordering thus saving much computational memory and time.

Currently, the complexity of the algorithm used in our minimization operation is very high. Development of a fast algorithm for it remains as a future work.

REFERENCES

[1] M. Ben-Ari. *Principles of concurrent programming.* Prentice Hall, 1982.

[2] M. Heymann. Concurrency and discrete event control. *IEEE Control Systems Magazine.*, 10(4):103–112, Jun. 1990.

[3] C. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, Aug. 1978.

[4] T. G. Kim. *DEVSim++ User's Manual: C++ Based Simulation with Hierarchical, Modular DEVS Models.* Systems Modeling Simulation Lab., KAIST, Taejon, Korea, 1994. ftp://sim.kaist.ac.kr/pub/DEVSim++-1.0/.

[5] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS.* Springer, New York, NY, USA, 1980.

[6] J. S. Ostroff. Formal methods for the specification and design of real-time safety critical systems. *Journal of Systems and Software.*, 18:33–60, Apr. 1992.

[7] A. W. Roscoe *et al.* Hierarchical compression for model-checking CSP or how to check 10^{20} dining philosophers for deadlock. In *Proceedings of the 5th International Conference, TACAS '95*, LNCS, pages 133–152. Springer, 1999.

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