LETTER

Simulation Modeling of SAM Fuzzy Logic Controllers

Hae Young LEE[†], Member, Seung-Min PARK[†], Nonmember, and Tae Ho CHO^{††a)}, Member

SUMMARY This paper presents an approach to implementing simulation models for SAM fuzzy controllers without the use of external components. The approach represents a fuzzy controller as a composition of simple simulation models which involve only basic operations.

key words: fuzzy logic controllers, standard additive models, modeling & simulation, discrete event system specification, embedded systems

1. Introduction

One particular use of modeling and simulations (M&S) is in the development of embedded systems [1]. The construction of embedded system models and their analysis through simulation reduces both end costs and risks, while enhancing system capabilities and improving the quality of the final products [2]. Especially, the discrete event system specification (DEVS) [3], a formal M&S methodology, has recently gained popularity due to the fact that it enables not only an interactive simulation but also a smooth transformation from models to executing code or other models in real-time environments [4].

In order to develop DEVS models for fuzzy logic controllers, external fuzzy logic components, such as the Free Fuzzy Logic Library (FFLL) [5], are typically used. The use of such external components, however, makes transforming models very difficult or impossible. Thus, the development of DEVS models for fuzzy controllers that do not use any external components would be useful. The existing research efforts [6], [7] to model fuzzy controllers without employing external components have focused on modeling of Mamdani fuzzy controllers [8], which is the most popular approach. However, most Mamdani controllers involve heavy computation in terms of resource-constrained embedded systems, such as sensor nodes.

This paper presents an approach to model the standard additive model (SAM) fuzzy logic controllers [9] based on DEVS. Compared to Mamdani controllers, SAM controllers are more suited for embedded systems due to its lower computation overhead. Our approach represents a SAM controller as a composition of simple DEVS models so that the

models would be smoothly transformed to code. Basically, our approach inherits the merits of SAM controllers. That is, it can produce lightweight DEVS models for fuzzy controllers in terms of computation. Moreover, it can reduce spatial and communicational overhead of models, compared to the existing approaches. Thus, it would be more suitable for the M&S based development of embedded systems.

2. Background

2.1 SAM Fuzzy Controller

The SAM, which is a general fuzzy logic controller, was introduced by B. Kosko in 1996 [9]. Suppose a SAM fuzzy controller that describes a mapping from $X \times Y$ to Z contains rules in the form of:

IF x is A_i AND y is B_i THEN z is C_i ,

where A_i , B_i , and C_i are the fuzzy membership functions. Then the model's output is:

$$z = \frac{\sum_{i=1}^n (\mu_{A_i}(x) \times \mu_{B_i}(y)) \times a_i \times g_i}{\sum_{i=1}^n (\mu_{A_i}(x) \times \mu_{B_i}(y)) \times g_i}, \quad (1)$$

where $\mu_A(x)$ is the membership degree of x in the membership function A , n is the number of if-then rules, a_i is the area under the i -th rule's conclusion C_i (i.e., the area of the membership function C_i , e.g., a in Fig. 3), and g_i is the centroid of C_i . The main advantage of SAM is the efficiency of its computation, because both a_i and g_i in (1) can be pre-computed (i.e., they are constants once the rules are defined) [10].

2.2 DEVS

The DEVS formalism is a theoretically well-grounded means of expressing modular discrete event simulation models developed by Zeigler [3]. A DEVS atomic model is defined as a 7-tuple:

$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle,$$

where

X is the set of input events,

Y is the set of output events,

Manuscript received October 22, 2009.

Manuscript revised February 20, 2010.

[†]The authors are with Electronics and Telecommunications Research Institute (ETRI), Daejeon 305-700, Republic of Korea.

^{††}The author is with School of Information and Communication Engineering, Sungkyunkwan University, Suwon 440-746, Republic of Korea (corresponding author).

a) E-mail: taecho@ece.skku.ac.kr

DOI: 10.1587/transinf.E93.D.1984

S is the set of sequential states,
 δ_{ext} is the external transition function,
 δ_{int} is the internal transition function,
 λ is the output function,
 ta is the time advanced function.

The DEVS environment supports building models in a hierarchical and modular manner, where the term “modular” means the description of a model in such a way that it has recognized input and output ports through which all interaction with the external world is mediated. This property enables the hierarchical construction of models so that complex models can be easily developed.

3. DEVS Modeling of Fuzzy Controllers

In our approach, a SAM fuzzy controller containing i input membership functions (MFs), j rules, and k output MFs, with l inputs and m outputs is implemented as a DEVS coupled model with i inputs and m outputs. The coupled model contains $i + j + k + m$ DEVS atomic models: i input MF models, j rule models, k output MF models, and m defuzzification models.

- **Input MF models (IMs):** each input MF of the fuzzy controller is implemented as a DEVS atomic model (Fig. 1 (a)) that receives a value x as input and immediately generates the membership degree of x in the MF I (i.e., $\mu_I(x)$) as output.
- **Rule models (RMs):** Each if-then rule of the fuzzy controller corresponds to a RM (Fig. 1 (b)). The RM corresponding to a rule R receives membership degrees, $d_1, d_2, \dots, d_i$, from the associated IMs of R (i.e., the MFs in the if-part of R). Then the model immediately sends the rule matching degree of R (i.e., $\prod d$) to the models corresponding to the output MFs of R (i.e., the MFs in the then-part). For example, the RM corresponding to a rule:

IF x is A AND y is B THEN z is C ,

receives $\mu_A(x)$ and $\mu_B(y)$ from the IMs corresponding to A and B , and then sends $\mu_A(x) \times \mu_B(y)$ to the model corresponding to C .

Fig. 1 DEVS atomic models for SAM fuzzy controller.

- **Output MF models (OMs):** each output MF of the fuzzy controller is implemented as a DEVS atomic model (Fig. 1 (c)) that receives the rule matching degrees from the associated RMs and immediately generates a pair $(d_R \cdot a_O \cdot g_O, d_R \cdot g_O)$ as output for each matching degree d_R , where a_i is the area of the MF and g_i is the centroid of the MF.
- **Defuzzification models (DMs)** (Fig. 1 (d)): Each output of the fuzzy controller corresponds to a defuzzification model. A DM (Fig. 1 (d)) receives pairs $(n_1, e_1), (n_2, e_2), \dots, (n_i, e_i)$ from the associated OMs and immediately generates a defuzzified value (i.e., $\sum n / \sum e$) as output.

Once a RM, an OM, and a DM have been developed, they can be easily reused without modification of their behavior. An IM for a MF type can also be reused in the development of IMs for new MF types with few modifications. In our approach, the only difference between IMs for different MF types is the definition of the external transition function. Thus, a new IM for the function can be developed just by redefining the external transition function of an existing IM. Note that every IM initially starts with the passive state. In the passive state, it never produces any membership degrees until the first input value arrives.

Table 1 shows the fuzzy if-then rules of a sample fuzzy controller for a toy washing machine, and Fig. 2 shows

Table 1 Fuzzy if-then rules of a sample fuzzy controller.

Rule No.	Input		Output
	Quantity	Softness	Cycle
1	Small	Soft	Light
2	Medium	Soft	Light
3	Large	Soft	Normal
4	Small	Normal	Light
5	Medium	Normal	Normal
6	Large	Normal	Strong
7	Small	Hard	Normal
8	Medium	Hard	Strong
9	Large	Hard	Strong

Fig. 2 DEVS model for the sample fuzzy controller.

Fig. 3 Development of a new fuzzy controller model.

Table 2 Overhead analysis for three approaches.

Overhead	Ours	Jamshidi's	Lee & Kim's
Space	$i + j + k + m$	$j \times l + j + j \times m + 2m$	$i + j + k + 2m$
Communication	$i + j \times l + j + l + m$	$l \times j + j \times l + j \times m + j \times m + 2m$	$i + j \times l + j + k + 2m$
Inference	Multiply	Find a minimum + Clip or scale a MF	
Combining	Add	Merge MFs	
Defuzzification	Multiply	Find the mean of the maximum or the centroid of an area	

a DEVS model of the controller, which is implemented using our approach. Most of the atomic models involve only basic operations such as addition or multiplication. Thus, they can be smoothly transformed to code.

A DEVS model M of a new fuzzy controller containing i input MFs, j rules, and k output MFs, with l inputs and m outputs can be implemented with the following steps (Fig. 3):

- Choose i IMs based on the MF types of the inputs and put them into M . Then assign their parameters (e.g., α , β , and γ in an IM for a triangular MF).
- Put j RMs into M .
- Put k OMs into M and assign the area a and the centroid g of each of them.
- Put m DMs into M .
- Link the models based on the if-then rules of the fuzzy controller.

Table 2 shows the space, communication, and computation overhead of fuzzy controller models built using three approaches: ours, Jamshidi's, and Lee & Kim's. Each controller contains i input MFs, j rules, and k output MFs, with l inputs and m outputs. In our approach, a controller DEVS model consists of $i + j + k + m$ sub-models, while more number of sub-models are required to implement a controller mod-

els in other approaches (see [7]). Due to the reduction of space overhead, our approach can also reduce communication overhead between atomic modes of a controller model (i.e., the number of messages generated by inter-communication). While Mamdani controllers, which can be modeled by the existing approaches, are widely used, they usually involve more complex operations, such as clipping of and merging of MFs and finding centroids, compared to SAM controllers. Such complex operations might be too heavy on resource-constrained embedded systems. On the other hand, our approach can model SAM controllers, which involve simple operations. Thus, our approach would be suitable for the M&S based development of embedded systems.

4. Conclusions and Future Work

This paper presents an approach to model SAM fuzzy controllers based on DEVS, in which a fuzzy controller can be modeled without the use of any external components. The sub-models can be easily reused with few modifications. They can be transformed to code with ease since they involve only basic operations such as addition or multiplication. Compared to the existing approaches, our approach can produce more lightweight models for fuzzy controllers, which is especially useful in embedded system development. We have developed the atomic models for DEVS Object C++ and DEVS Scheme, and are going to develop the models for other environments, such as DEVS Sim++ and DEVS JAVA.

References

- [1] Y.H. Yu and G. Wainer, "eCD++: An engine for executing DEVS models in embedded platforms," Proc. SCSC, pp.323–330, 2007.
- [2] G. Wainer and E. Glinsky, "Model-based development of embedded systems with RT-CD++," <http://www.cs.virginia.edu/rtas04/wip/wip27.pdf>
- [3] B.P. Zeigler, T.G. Kim, and H. Praehofer, Theory of Modeling and Simulation, 2nd ed., Academic Press, 2000.
- [4] H. Shang and G. Wainer, "Dynamic structure DEVS: Improving the real-time embedded systems simulation and design," Proc. ANSS, pp.271–278, 2008.
- [5] The Free Fuzzy Logic Library. <http://ffl.sourceforge.net/>
- [6] M. Jamshidi, S. Sheikh-Bahaei, J. Kitzinger, P. Sridhar, S. Xia, Y. Wang, J. Liu, E. Tunstel, Jr., M. Akbarzadeh, A. El-Osery, M. Fathi, X. Hu, and B.P. Zeigler, "A distributed intelligent discrete-event environment for autonomous agents simulation," Applied System Simulation: Methodologies and Applications, pp.241–274, Springer, 2003.
- [7] H.Y. Lee and H.J. Kim, "Reducing the complexity of DEVS-based mamdani models for enhancing privacy," Proc. ISIS, pp.281–283, 2009.
- [8] E.H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," International Journal of Machine Studies, vol.7, no.1, pp.1–13, 1975.
- [9] B. Kosko, Fuzzy Engineering, Prentice Hall, 1997.
- [10] J. Yen and R. Langari, Fuzzy Logic: Intelligence, Control, and Information, Prentice Hall, 1999.