Simulation Modeling of TSK Fuzzy Systems for Model Continuity

Hae Young Lee, Jin Myoung Kim, Ingeol Chun, Won-Tae Kim and Seung-Min Park

Abstract This paper presents an approach to formally model Takagi–Sugeno–Kang (TSK) fuzzy systems without the use of any external components. In order to keep the model continuity, the formal simulation model for a TSK fuzzy system is comprised of three types of reusable sub-models involving primitive operations. Thus, the model can be executed even on limited computational platforms, such as embedded controllers.

Keywords Modeling and simulation · Model continuity · Fuzzy logic · Discrete event system specification · Embedded systems

1 Introduction

Modeling and simulation (M&S) technologies have been widely used in industry to assist in system development [1]. One particular use of these technologies is in the development of embedded controllers since they usually have time constraints [2, 3]. When modelers build simulation models for embedded fuzzy controllers, they typically embed external fuzzy components in their models [4, 5]. These models, however, may not be used throughout all of the design phases since M&S

This work was supported by the IT R&D Program of MKE/KEIT [10035708, "The Development of CPS (Cyber-Physical Systems) Core Technologies for High Confidential Autonomic Control Software"].

H. Y. Lee (✉) · J. M. Kim · I. Chun · W.-T. Kim · S.-M. Park
CPS Research Team, ETRI, Daejeon 305-700, Republic of Korea
e-mail: haelee@ieee.org

J. J. Park et al. (eds.), *IT Convergence and Services*,
Lecture Notes in Electrical Engineering 107, DOI: 10.1007/978-94-007-2598-0_8,

environments do not support the use of some external components. Also, the use of external components may make the transformation of simulation models difficult or impossible [6]. Therefore, simulation models should not contain any external components to keep their continuity.

Several research efforts [6–8] have been made to build 'pure' simulation models for fuzzy controllers. In [7], Jamshidi et al. proposed an approach to model the Mamdani fuzzy systems [9] based on pararell discrete event system specification (P-DEVS) [10]. The modeling approach proposed by Lee and Kim [8] can reduce the complexity of the Mamdani P-DEVS models. The Mamdani model has a great advantage in terms of expression power, though it involves some complex computation. The standard additive model (SAM) fuzzy systems [11] can be built with P-DEVS models based on the approach proposed in [6]. The main advantage of the SAM is computational efficiency since most parameters can be precomputed. However, simulation modeling of Takagi–Sugeno–Kang (TSK) fuzzy systems [12, 13] has not been addressed yet. Compared to the Mamdani model, the TSK model can reduce the number of rules, especially for complex and high-dimensional problems.

This paper presents an approach to build simulation models for TSK fuzzy systems based on P-DEVS. A P-DEVS model of a TSK fuzzy system is a coupled model consisting of three types of sub-models: an input membership function model, rule model, and defuzzification model. Since the models are all pure simulation models involving only addition and multiplication, they could be executed even on embedded platforms. Consequently, their continuity can be maintained. Compared to the existing approaches for the modeling of fuzzy systems, the proposed approach can model a TSK fuzzy system with a smaller number of sub-models.

2 Background

In this section, we briefly describe the backgrounds of TSK fuzzy systems and P-DEVS.

2.1 TSK Fuzzy Systems

In general, a rule in a TSK model has the following form:

$$\text{IF } x_1 \text{ is } A_{i1} \text{ and } x_2 \text{ is } A_{i2} \text{ and, } \ldots, \text{ and } x_k \text{ is } A_{ik}$$

$$\text{THEN } y = a_{i0} + a_{i1} \times x_1 + \cdots + a_{ik} \times x_k$$

where x_1, x_2,…, x_k are input parameters, A_{i1}, A_{i2},…, A_{ik} are the membership functions of ith rule, a_{i0}, a_{i1},…, a_{ik} are real-valued parameters, and y is the output parameter. The total output, y, of the model is given by Eq. (1), where α_i is the matching degree of the i-th rule.

$$y = \frac{\sum_{i=1}^{j} \alpha_i (a_{i0} + a_{i1}x_1 + \cdots + a_{ik}x_k)}{\sum_{i=1}^{j} \alpha_i} \tag{1}$$

The great advantage of the TSK model is its representative power. Moreover, due to the explicit functional representation form, it is convenient to identify its parameters using learning algorithms [14].

2.2 Parallel DEVS

The basic formalism of a P-DEVS model is:

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

where

X is the set of input events,
Y is the set of output events,
S is the set of sequential states,
δ_{ext}: $Q \times X^b \rightarrow S$ is the external transition function,

where $Q = \{(s, e) \mid s \in S, 0 < e < ta\ (s)\}$, e is the elapsed time since the last state transition, and X^b is a set of bags over the elements in X,

δ_{int}: $S \rightarrow S$ is the internal transition function,
δ_{con}: $Q \times X^b \rightarrow S$ is the confluent transition function, subject to $\delta_{con}(s, \varnothing) = \delta_{int}(s)$,
λ: $S \rightarrow Y^b$ is the output function,
ta is the time advanced function.

3 P-DEVS Modeling of TSK Fuzzy Systems

In the proposed approach, a TSK fuzzy system containing i input membership functions and j rules, with k inputs and a single output is represented as a P-DEVS coupled model with k input ports and a single output port. The coupled model contains $i + j + 1$ P-DEVS atomic models: i input membership function models (IMs), j rule models (RMs) and a single defuzzification model (DM). Figure 1 shows the P-DEVS model of a fuzzy system containing four input membership functions and four rules with two inputs and a single output (i.e., $i = 4$, $j = 4$, $k = 2$).

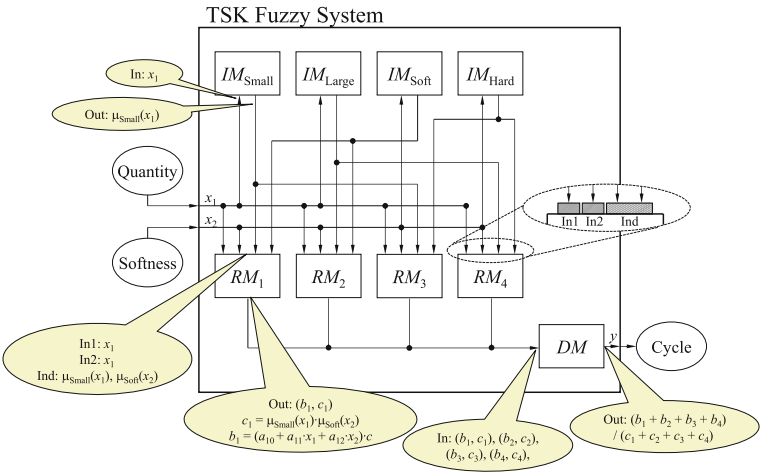

Fig. 1 A model structure for a TSK fuzzy system

3.1 Input Membership Function Models (IMs)

Each input membership function of the fuzzy system is represented as an IM M that is defined as:

$$M = \langle X_M, Y_M, S, \delta_{ext}, \delta_{\text{int}}, \delta_{con}, \lambda, ta \rangle,$$

where

$InPorts = \{\text{“In”}\}$, $X_{\text{In}} = \Re$,
$OutPorts = \{\text{“Out”}\}$, $Y_{\text{Out}} = [0, 1]$,
$X_M = \{(p, v) \mid p \in InPorts, v \in X_p\}$,
$Y_M = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$,
$S = \{\text{“passive”}, \text{“active”}\} \times \Re$,
δ_{ext} (“passive”, d, e, (“In”, x)) = (“active”, μ (x)),
δ_{int} (“active”, d) = (“passive”, d),
δ_{con} (s, ta (s), x) = δ_{ext} (δ_{int} (s), 0, x)
λ (“active”, d) = (“Out”, d),

$$ta(phase, d) = \begin{cases} 0 & \text{if } phase = \text{``active''}; \\ \infty & \text{otherwise.} \end{cases}$$

Every IM produces a matching degree of the corresponding membership function for each input value. It initially starts with its state = (“passive”, d), where d is an arbitrary real value. When the IM for an input membership function I receives a real value x as an input, it transitions its state to (“active”, μ_I (x)). Immediately, the IM generates μ_I (x) as its output and transitions to the passive state.

Once an IM for a membership function type has been implemented, it can be easily reused just by setting the parameters of the membership functions in the same type. Even if any IM for a certain type does not exist, it can be implemented just by redefining δ_{ext} of the existing one. IMs are independent from fuzzy inference models; TSK, SAM, and Mamdani use the same IMs in the approach.

3.2 Rule Models (RMs)

Each if-then rule of the fuzzy system corresponds to an RM. An RM is defined as:

$$M = \langle X_M, Y_M S, \delta_{ext}, \delta_{\text{int}}, \delta_{con}, \lambda, ta \rangle,$$

where

$InPorts = \{\text{“In1”},\ldots, \text{“In}k\text{”}, \text{“Ind”}\}$, $X_{\text{In1}} = \cdots = X_{\text{In}k} = X_{\text{Ind}} = \Re$,
$OutPorts = \{\text{“Out”}\}$, $Y_{\text{Out}} = \Re^2$,
$X_M = \{(p, v) \mid p \in InPorts, v \in X_p\}$,
$Y_M = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$,
$S = \{\text{“passive”}, \text{“active”}\} \times \Re^{k+3}$,
δ_{ext} (“passive”, a_0,…, a_k, b, c, e, ((“In1”, x_1),…, (“Ink”, x_k), (“Ind”, d_1),…, (“Ind”, d_k))
= (“active”, a_0,…, a_k, $a_0 + \cdots + a_k \cdot x_k$, min ($d_1$,…, d_k))
or
= (“active”, a_0,…, a_k, $a_0 + \cdots + a_k \cdot x_k$, $d_1 \times \cdots \times d_k$),
δ_{int} (“active”, a_0,…, a_k, b, c) = (“passive”, a_0,…, a_k, b, c),
δ_{con} (s, ta (s), x) = δ_{ext} (δ_{int} (s), 0, x)
λ (“active”, a_0,…, a_k, b, c) = (“Out”, ($b \times c$, c)),

$$ta(phase, a_0, \ldots, a_k, b, c) = \begin{cases} 0 & \text{if } phase = \text{``active ''}; \\ \infty & \text{othewise.} \end{cases}$$

Each RM produces a conclusion of the corresponding rule, based on input values: all input values of the fuzzy system and the membership degrees from the associated IMs. It has k input ports, “In1”,…, “Ink”, used to receive the k input values, x_1,…, x_k, of the fuzzy system; an additional input port, “Ind”, used for k membership degrees, d_1,…, d_k, from the associated IMs; and a single output port, “Out.” The RM corresponding to rule R starts with the initial state = (“passive”, a_0,…, a_k, b, c), where a_0,…, a_k are the constant values defined in the consequent part (then-part) of R, and b and c are arbitrary real values. When the RM receives x_1,…, x_k, through the ports “In1”,…, “Ink”, respectively, together with d_1,…, d_k through the port “Ind”, it stores $b = a_0 + \cdots + a_k \cdot x_k$ and c = min (d_1,…, d_k) or ($d_1 \times \cdots \times d_k$). Finally, the RM outputs ($b \times c$, c) via the output port and transitions to the passive state.

The RM can be reused repeatedly once it has been implemented. The reuse can be done simply through the creation of RM instances and assigning $k + 1$ parameters $a_0,\ldots, a_k$, of each instance.

3.3 Defuzzification Model (DM)

A DM is an application-independent atomic-model that generates the outputs of a fuzzy system. It is formally defined as:

$$M = \langle X_M, Y_M, S, \delta_{ext}, \delta_{\text{int}}, \delta_{con}, \lambda, ta \rangle$$

where

$InPorts = \{\text{“In”}\}$, $X_{\text{In}} = \Re^2$,
$OutPorts = \{\text{“Out”}\}$, $Y_{\text{Out}} = \Re$,
$X_M = \{(p, v) \mid p \in InPorts, v \in X_p\}$,
$Y_M = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$,
$S = \{\text{“passive”}, \text{“active”}\} \times \Re$,
δ_{ext} $(phase,$ $y,$ $e,$ $(b_1,$ $c_1),\ldots,$ $(b_j,$ $c_j)) = (\text{“active”},$ $(b_1 + \cdots + b_j)/(c_1 + \cdots + c_j))$,
δ_{int} (“active”, y) = (“passive”, y),
λ (“active”, y) = (“Out”, y),

$$ta(phase, y) = \begin{cases} 0 & \text{if } phase = \text{‘‘active’’;} \\ \infty & \text{otherwise.} \end{cases}$$

The DM produces a final conclusion of the fuzzy system based on the collection of conclusions of the rules. It starts with the passive state = (“passive”, y), where y is an arbitrary real value. When the DM receives $(b_1, c_1),\ldots, (b_j, c_j)$ from all RMs, it transitions its state to (“active”, $(b_1 + \cdots + b_j)/(c_1 + \cdots + c_j)$). Then, the DM outputs y and transitions its state back to the passive state.

Since any implementation of the DM is application-independent, it is reused for every TSK fuzzy system. Also, it is identical to the DM of SAM described in [6].

3.4 Models Couplings

The coupled model has i input ports and a single output port. Each of the input ports is connected with the associated input ports (e.g., input port “In1” for input x_1, “In2” for x_2,…) of all RMs. The port is also coupled with the input ports of the associated IMs. Consider the following fuzzy if-then rules of a TSK fuzzy system that receives quantity x_1 and softness x_2:

Table 1 Overhead in four modeling approaches

Complexity	TSK (proposed)	SAM [6]	Mamdani [7]	Mamdani [8]
Sub-models	$i + j + 1$	$i + j + l + 1$	$j \times k + 2j + 2$	$i + j + l + 2$
Communications	$i + 2j \times k + j + 1$	i $+ j \times k + j + k + 1$	$2j \times k + 2j + 2$	$i + j \times k +$ j $+ l + 2$
Inference	Multiplication + addition	Multiplication	Finding a minimum + clipping of or scaling a MF	
Combining	Addition		Merging MFs	
Defuzzification	Multiplication		Finding the mean of the maximum or the centroid of an area	

$$\text{Rule 2 : IF } x_1 \text{ is Large and } x_2 \text{ is Soft}$$

$$\text{THEN } y = 1 + 2 \cdot x_1 + 2 \cdot x_2$$

$$\text{Rule 3 : IF } x_1 \text{ is Small and } x_2 \text{ is Hard}$$

$$\text{THEN } y = 1 + x_1 + 2 \cdot x_2$$

In the above example, the port that receives x_1 (quantity value) would be connected with the input ports of IM_{Small} and IM_{Large}. Note that each IM has a single input port "In". The output port "Out" of each IM is coupled with input port "Ind" of each associated RMs. In the example, "Out" of IM_{Small} would be connected with "Ind" of RM_3, which have a linguistic variable 'Small' in the if-part. The output port of every RM is coupled with the input port of the DM. And the output port of the DM is connected with that of the coupled model.

3.5 Overhead Analysis

Table 1 shows an overhead analysis for the approach and the three existing approaches [6–8]. Each fuzzy system consists of i input membership functions, j rules, and l output membership functions, with k inputs and a single output. In the approach, a coupled model for a fuzzy system contains $i + j + 1$ atomic models, while a higher number of sub-models is required to build a coupled model for a fuzzy system in other approaches. The complex couplings among the sub-models in the approach make the communications overhead increase. However, the overhead of the approach is still smaller than that of [7]. Moreover, TSK can describe a highly nonlinear system using a small number of rules [14]. That is, j of TSK could be much smaller than that of SAM or Mamdani. While Mamdani fuzzy systems are widely used, they usually involve complex operations, such as the clipping and merging of membership functions and finding their centroids. Such

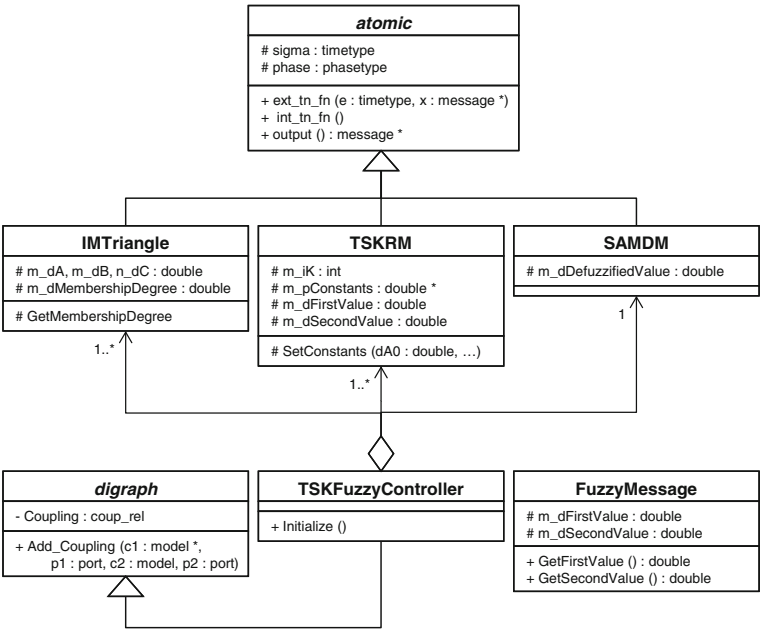

Fig. 2 Simplified UML diagrams of TSK simulation models

complex operations might be too heavy on resource-constrained systems. Similar to [6], the approach can model TSK fuzzy systems, which involve only primitive operations. Thus, the approach will be suitable for the M&S-based engineering of embedded software systems.

4 Implementation Status

The P-DEVS models for TSK systems described in Sect. 3 were implemented in C++ for our simulation environment, the DEVS object C++ (DOC) environment. The IM for the triangular membership function type was implemented as an IMTraingle class. The RM and DM were implemented as TSKRM and SAMDM classes, respectively. As shown in Fig. 2, these classes inherit the atomic class of DOC class, which corresponds to the basic model of P-DEVS. The essential member functions of atomic are ext_tn_fn (the implementation of δ_{ext}), int_tn_fn (δ_{int}), and output (λ). By overriding these functions, the behavior of a subclass is defined. The FuzzyMessage class is used for internal communications between the atomic models of fuzzy systems. In order to facilitate the modeling process, we have also developed a prototype of a visual modeling tool. The simulation modeling of fuzzy systems can be done with ease using the tool. However, they

can also be manually constructed without the use of the tool, thanks to the hierarchical and modular model-composition provided by the P-DEVS environments.

5 Conclusions and Future Work

In this paper, we presented an approach for representing P-DEVS models of TSK fuzzy systems without the use of any external components. Exclusion of external components from simulation models would improve the continuity of the models so that the user can efficiently manage software complexity and maintain consistency throughout the design phase. A P-DEVS model of a fuzzy system is comprised of easy-to-reuse atomic models: IMs, RMs, and a DM. Since each atomic model involves primitive operations, such as addition or multiplication, the model works on target platforms and can be smoothly transformed into other forms of models or languages. A coupled model for a TSK fuzzy system requires a smaller number of sub-models, compared to that of a Mamdani or SAM fuzzy system. Thus, it will be more compatible with embedded platforms. We have implemented P-DEVS models on DOC, for fuzzy systems including TSK, SAM, and Mamdani. To facilitate the modeling of fuzzy systems, a GUI-based modeling tool prototype was developed. We will implement the models for other DEVS environments, such as eCD++ [2].

References

1. Hu X (2004) A simulation-based software development methodology for distributed real-time systems. Doctoral dissertation, The University of Arizona
2. Moallemi M, Gutierrez-Alcaraz JM, Wainer G (2008) ECD++ A DEVS based real-time simulator for embedded systems. In: Proceedings of the spring simulation multiconference, article no. 12
3. Park J, Yoo J (2010) Hardware-aware rate monotonic scheduling algorithm for embedded multimedia systems. ETRI J 32:657–664
4. Garcia AM, Baumgartner B, Schreiber U, Krane M, Knoll A, Bauernschmitt R (2009) Automedic: fuzzy control development platform for a mobile heart-lung machine. IFMBE Proc 25:685–688
5. Muruganandam M, Madheswaran M (2009) Modeling and simulation of modified fuzzy logic controller for various types of DC motor drives. In: Proceedings of international conference on control, automation, communication and energy conservation, pp 1–6
6. Lee HY, Park SM, Cho TH (2010) Simulation modeling of SAM fuzzy logic controllers. IEICE Trans Inf Syst E93-D:1984–1986
7. Jamshidi M, Sheikh-Bahaei S, Kitzinger J, Sridhar P, Beatty S, Xia S, Wang Y, Song T, Dole U, Lie J (2003) V-LAB–A distributed intelligent discrete-event environment for autonomous agents simulation. Intell Autom Soft Comput 9:181–214
8. Lee HY, Kim HJ (2009) Reducing the complexity of DEVS-based mamdani models for enhancing privacy. Proceedings of international symposium on advanced intelligent systems, pp 281–283

9. Mamdani EH (1974) Application of fuzzy algorithms for control of simple dynamic plant. IEEE Proc 121:1585–1588
10. Zeigler BP, Kim TG, Praehofer H (2000) Theory of modeling and simulation, 2nd edn. Academic Press, New York
11. Kosko B (1997) Fuzzy engineering. Prentice Hall, Upper Saddle River
12. Takagi T, Sugeno M (1985) Fuzzy identification of systems and its application to modeling and control. IEEE Trans Syst Man Cybern 15:116–132
13. Sugeno M, Kang KT (1988) Structure identification of fuzzy model. Fuzzy Sets Syst 28: 15–33
14. Yen J, Langari R (1999) Fuzzy logic: intelligence control and information. Prentice Hall, Englewood Cliffs