Rapid Prototyping for Fuzzy Systems*

Chantana Chantrapornchai   Sissades Tongsima   Edwin H.-M. Sha
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556

Abstract

One of the common problems for fuzzy system implementation arises from the complications of the fuzzy inference process. Extra computations are required to deduce a consequence due to nature of fuzzy sets. Furthermore, considerable simulations need to be performed to verify system functions. In order to reduce the prototyping time, the fuzzy system is partitioned into hardware and software portions. The model, called Fuzzy Rule-based Automata (FRA), is proposed to simplify fuzzy rule base. Since most rule base are rarely changed, they can be implemented in hardware to speedup the running time. Special computations for inference process is taken care by software to reduce hardware complications which yields prototype flexibility.

1. Introduction

Nowadays fuzzy logic [8, 23, 24] is commonly used in expert systems, especially in control systems. In these systems, a consequence is inferred using rule base so that a control action can be performed. Implementing a fuzzy system is complex due to a large number of rules and fuzzy inference computations in the system. Since fuzzy systems are designed from human language, the performance of the systems is as good as represented knowledge which are given by specialists. Before building a system, many real-time experiments need to be performed to verify system performance. Therefore, rapid prototyping becomes an important issue here.

To achieve rapid prototyping, the implementation of fuzzy rule base system is partitioned into two portions: hardware and software. Fuzzy rule base are implemented in the hardware part since these rule base are rarely changed. By so doing, a system simulation time can be reduced. The second part, also called software inference engine, handles some necessary computations for fuzzy logic which includes built-in operations such as membership functions and fuzzy implications. This is best to be implemented in software because these operations can easily be updated for numerous experiments. According to such inputs, the rule-based hardware produces proper simulation outputs which will be used by software inference engine. The software inference engine, then, collects results from the hardware simulations, and computes them by using built-in operations. Thereafter, it defuzzifies such output results where appropriate consequence is produced.

Considerable work has been done on developing hardware implementations for fuzzy control systems. Special hardware chips have been invented to speedup a fuzzy inference [9, 16]. Some researchers focus on designing processor architectures for fuzzy logic controllers [2, 4, 5, 13, 17, 19]. However, these pure hardwares are complex. Fuzzy automata model [6, 12, 20] proposed by Wee has been frequently used for pattern recognition and learning applications [18, 21]. They, nonetheless, have not applied the fuzzy automaton for fuzzy control systems.

In this research, we propose a new model called Fuzzy Rule-based Automata (FRA) which is used to design a circuit. The rule base are transformed into such a model and systematically converted into hardware. Obtaining hardware, using the user-defined membership functions, fuzzy implications and the fact that whether the rules are conjunctive or disjunctive, the software accumulates each output and translates them properly to obtain a conclusion. Using our methods to implement fuzzy systems have some advantages. First, the FRA model facilitates the hardware implementation of fuzzy rule base. Secondly, embedding rule base in hardware can speedup simulation time. Further, by implementing other operations, e.g. membership functions, fuzzy implication, in software, they can be easily modified for different tests without changing the hardware. This software implementation yields flexibility in doing experiments before setting up a real system.

Figure 1(a) shows an example of rule base for a simple temperature control system. The corresponding FRA is shown in Figure 1(b). This automaton has 3 internal

---

*This work was supported by NSF grant MIP 95-01006 and Royal Thai Government Scholarship.
states \( s_1, s_2, s_3 \), two input variables \( \text{cold}, \text{hot} \), and two output variables \( \text{slow}, \text{fast} \). The initial internal state is \( s_1 \). The corresponding membership functions are \( \mu_{\text{cold}}(t) \), \( \mu_{\text{hot}}(t) \), \( \mu_{\text{slow}}(t) \), and \( \mu_{\text{fast}}(t) \). Figure 2 presents a block diagram of the entire inference engine. The dash block on the left-hand side shows the hardware implementations for rule base. Since the FRA has 2 input linguistic variables, 2 output linguistic variables, and 3 states, we can use 1 bit to encode inputs and 1 bit to encode outputs. Two bits can be used to encode internal states. To perform a fuzzy inference process, for each input and current internal state, the circuits select the next state and output. To obtain a conclusion, this hardware circuit will be applied at most 6 times (3 states, 2 inputs). The remaining blocks of the diagram are implemented in software. User-defined output operations integrate membership functions as well as fuzzy implication and calculate proper output values. State operations compute max-min operation against the current state and inputs to obtain a set of next states. The temperature input, fuzzified by using membership functions \( \{\mu_{\text{cold}}(t), \mu_{\text{hot}}(t)\} \), is fed to compute the next states by state operations, and to calculate membership values of fan speed by user-defined output operations. Finally, the results are defuzzified and the control is sent to the fan speed controller.

This paper is organized as following: Section 2 presents some backgrounds on fuzzy set theory as well as fuzzy reasoning, and the original fuzzy automata model. Section 3 discusses the formal definitions of our model (FRA). The implementation issues and fuzzy inference procedure based on this model are proposed in Section 4. The concluding remarks are presented in Section 5.

2. Preliminaries

In this section, some backgrounds on fuzzy set theory, fuzzy inference and the original fuzzy automata model are provided.

Fuzzy sets, proposed by Zadeh, represent a set with imprecise boundary [23, 24]. In classical (crisp) sets, an element can either be a member of a set or not at all; hence, its membership degree is either 1 or 0. A fuzzy set is defined by assigning each element in a universe of discourse its membership degree which is in the unit interval \([0, 1]\), conveying to what degree \( x \) is a member in the set. This membership value can be defined as a membership function of an element in the set, \( \mu_A(x) : x \rightarrow [0, 1] \).

Because of the non-binary membership degree, normal set operation such as union, intersection, or complement are extended so that the membership value of a result set can be calculated appropriately. Let \( A \) and \( B \) be fuzzy sets from the same universe.

\[
\begin{align*}
\text{Union} & : \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \\
\text{Intersection} & : \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \\
\text{Complement} & : \mu_A^C(x) = 1 - (\mu_A(x))
\end{align*}
\]

The followings are definitions for fuzzy relations: Let \( \sim \) and \( \approx \) be fuzzy relations on the Cartesian space \( x \times y \), \( \preceq \) be a fuzzy relations on the Cartesian space \( y \times z \), and \( \bowtie \) be a fuzzy relations on the Cartesian space \( x \times z \). \( \mu_g(x, y) \) denotes a membership value of a fuzzy relation \( \bowtie \) for given elements \( x \) and \( y \).
2.2. Fuzzy inference

A rule-based system consists of a set of rules in the form

If antecedent (premise) THEN consequent (conclusion)

In classical logic, the evaluation of the antecedent part can return either true or false (0 or 1). For fuzzy logic, premise and conclusion contain linguistic variable(s) that have corresponding membership function(s). For example, If the temperature is hot, THEN speed up the fan. hot is a linguistic variable which corresponds to the membership function \( \mu_{\text{hot}}(x) \), mapping from temperature \( x \) to a value in \([0, 1]\), describing the degree of hotness of an instance \( x \). Hence, to conclude the consequent, special computations is required for these membership values.

For simplicity in explanation, we describe a set of rules in a canonical form [1].

\[
\begin{align*}
R_1: & \text{ If } x_1 = x_1^1 \text{ and } \ldots \text{ and } x_m = x_m^1, \text{ THEN } y = y^1 \\
R_2: & \text{ If } x_1 = x_1^2 \text{ and } \ldots \text{ and } x_m = x_m^2, \text{ THEN } y = y^2 \\
& \vdots \\
R_n: & \text{ If } x_1 = x_1^n \text{ and } \ldots \text{ and } x_m = x_m^n, \text{ THEN } y = y^n
\end{align*}
\]

\( x \) is a linguistic variable from the universe of discourse \( U_j \). An element \( x_j \) denotes an instance in the universe and \( \mu_{\text{in}}(x_j) \) represents its corresponding membership value. \( y \) is a linguistic variable from the consequent from the universe of discourse \( V \) with the membership function \( \mu_{\text{out}}(y) \).

To evaluate the system, \( m \) non-interactive inputs, \( \{X_1, \ldots, X_m\} \) are required and a single fuzzy output \( Y' \) is produced. If the rules are disjunctive, \( Y' \) is obtained from

\[
Y' = \mu_{\text{or}}(Y_1, \ldots, Y_m) = \mu_{\text{or}}(\mu_{\text{in}}(y_1), \ldots, \mu_{\text{in}}(y_m))
\]

where

\[
\mu_{\text{or}}(y) = \max_{1 \leq i \leq m} \text{min}(\mu_{\text{in}}(y_i), \mu_{\text{or}}(y_{i-1}))
\]

\( Y' \) is the fuzzy output of \( Y' = \mu_{\text{or}}(Y_1, \ldots, Y_m) \) where \( Y_i \) is a relation for rule \( i \) computed by

\[
\mu_{\text{in}}(x_1, \ldots, x_m) = \mu_{R_i}(x_1, \ldots, x_m)
\]

\( R \) is a fuzzy implication operator [7]. Different fuzzy implications such as Mamdani's, Zadeh's, Lukasiewicz's etc. have different properties [10, 11, 22]. If \( R \) is \text{min} operation or Mamdani 's implicator, Equation (1) will consist of only \text{max-min} operations.

2.3. Fuzzy automata

A fuzzy automata was originally proposed by W. G. Wee as a model of pattern recognitions and learning system [21]. It is defined by five tuples [7]: \( A = (X, Y, Z, S, R) \) where

- \( X \) is a nonempty finite set of input states,
- \( Y \) is a nonempty finite set of output states,
- \( Z \) is a nonempty finite set of internal states,
- \( S \) is a fuzzy relation on \( X \times Z \times Y \),
- \( R \) is a fuzzy relation on \( Z \times Y \).

\( S \) defines a transition function for a given input and current state as in the original finite state machine. \( R \) defines an output issued by a given state. Assume that \( X = \{x_1, \ldots, x_n\} \), \( Y = \{y_1, \ldots, y_m\} \) and \( Z = \{z_1, \ldots, z_q\} \).

Let \( Z' \) denote current internal states (t). For a given input \( X' \), first a transition relation \( Z \) is converted into 2-dimension relation \( Z \times Z \) by the following formula:

\[
S(x, z, t) = \max(\min(s_1, S(x_1, z_1, t)), \ldots, \min(s_m, S(x_m, z_q, t)))
\]

for all \( (z_i, z_j) \). Therefore, the next state and the next output are given by [7]

\[
Z_i^{t+1} = Z \circ S \quad Y_i^{t+1} = Z \circ R
\]

In order to define a reasonable transition relation, \( S(x_k, z_i, z_j) > 0 \) for at least one \( z_j \) for each pair \( (x_k, z_i) \in X \times Z \) and \( R(z_i, y_l) > 0 \) for at least \( y_l \in Y \) for each \( z_i \).

The transition and output relations, \( S \) and \( R \) are said to be crisp deterministic if for each \( (x_k, z_i) \in X \times Z \), \( S(x_k, z_i, z_j) = 1 \), at only one \( z_j \in Z \) and \( R(z_i, y_l) = 1 \), exactly one \( y_l \in Y \) for each \( z_i \). That is, for a specific input and current state, there is only one possible next state and output.

3. Fuzzy rule-based automata (FRA)

In this section, we formalize the FRA, the next state operations as well as the user-defined output operations for a fuzzy inference process.
Definition 3.1 A Fuzzy Rule-based Automata (FRA) \( \mathcal{A} \) is a quintuple \( (I, \mathcal{O}, S, \delta, \gamma) \) where \( I \) is a set, \( \{i_1, i_2, \ldots, i_k\} \), and each \( i_k \) is an input linguistic variable which corresponds to a membership value \( \mu_k(x) \). \( \mathcal{O} \) is a set \( \{o_1, o_2, \ldots, o_m\} \), where each \( o_m \) is an output linguistic variable which corresponds to a membership value \( \mu_{om}(y) \), and \( S \) is a set of states \( \{s_1, s_2, \ldots, s_q\} \) corresponding to a state vector \( [v_{s_1}, v_{s_2}, \ldots, v_{s_q}] \). \( v_{s_k} \) is a value ranged \([0, 1]\), representing the likelihood of being in state \( s_k \). \( \delta \) is a crisp deterministic transition relation \( S \times S \times I \) and \( \gamma \) is a crisp deterministic output relation on \( S \times O \times I \), defined by:

\[
\delta(s_i, s_j, i_k) = \begin{cases} 
1 & \text{for } i_k, \exists \text{ a transition } s_i \rightarrow s_j \\
0 & \text{otherwise}
\end{cases}
\]

and

\[
\gamma(s_k, o_m, i_p) = \begin{cases} 
1 & \text{at } (s_k, o_m, i_p) \text{ output } o_k \\
0 & \text{otherwise}
\end{cases}
\]

With the conditions that for each \( (i_k, s_i) \) if \( \delta(s_i, s_j, i_p) = 1 \), \( \delta(s_i, s_j, i_k) = 0 \) for \( k \neq p \); for each \( (i_k, s_i) \) if \( \gamma(s_i, o_m, i_k) = 1 \), \( \gamma(s_i, o_j, i_k) = 0 \) for \( j \neq p \), and for any two input variables \( i_p \) and \( i_q \) which are in the same universe, \( \gamma(s_i, o_j, i_p) \neq \gamma(s_i, o_j, i_q) \).

For the system in Figure 1(a), \( I = \{\text{cold, hot}\} \) and \( O = \{\text{slow, fast}\} \). 3 states, \( s_1, s_2, \) and \( s_3 \), are defined corresponding to the initial state, hot state and cold state respectively. The followings show parts of the transition relation and output relation according to the previous definition:

\[
\begin{align*}
\delta(s_1, s_1, \text{hot}) &= 0 \\
\delta(s_1, s_2, \text{hot}) &= 1 \\
\delta(s_2, s_1, \text{hot}) &= 0 \\
\delta(s_2, s_2, \text{hot}) &= 0 \\
\delta(s_3, s_1, \text{hot}) &= 0 \\
\delta(s_3, s_2, \text{hot}) &= 1 \\
\delta(s_3, s_3, \text{hot}) &= 0 \\
\gamma(s_1, \text{slow, cold}) &= 1 \\
\gamma(s_1, \text{fast, cold}) &= 0 \\
\gamma(s_1, \text{slow, hot}) &= 0 \\
\gamma(s_1, \text{fast, hot}) &= 1 \\
\gamma(s_2, \text{slow, cold}) &= 1 \\
\gamma(s_2, \text{fast, cold}) &= 0
\end{align*}
\]

The following definitions specify details of the next state operations and user-defined output operations.

Definition 3.2 Given an FRA \( \mathcal{A} = (I, O, S, \delta, \gamma) \), at instance \( t \), the current input \( x \) has membership values \( l_i = ([\mu_{i_1}(x), \mu_{i_2}(x), \ldots, \mu_{i_k}(x)] \) for each linguistic variable \( i_k \) and \( S^t \) denotes the current internal state, corresponding to the vector \( [v_{s_1}, v_{s_2}, \ldots, v_{s_m}] \), the next internal state values, \( S^{t+1} \) are computed by,

\[
S^{t+1} = S^t \circ \varphi_t \quad \text{where} \quad \varphi_t(s_i, s_j) \quad \text{is (3)}
\]

\[
\max[\min(\mu_{i_1}(x), \delta(s_i, s_j, i_1)) \ldots \min(\mu_{i_k}(x), \delta(s_i, s_j, i_n))]
\]

and the membership values for output are computed by,

\[
O^{t+1} = \Delta(O^t, S^t, l_i \geq 0)
\]

\( \Delta \) is a user-defined operation to calculate the cut for each output linguistic variable \( o_j \in O \) which may depend on the current output, state and all previous input values. \( \circ \) is a fuzzy relation composition.

In the system in Figure 1(b), the initial state is \( s_1 \); hence, the corresponding state vector contains values \([1, 0, 0]\). Suppose we choose a min operation as a fuzzy implication and two rules in Figure 1(a) are disjunctive. In the original inference process, for each \( l_i \), \( \min(\mu_{\text{cold}}(l_i), \mu_{\text{hot}}(l_i)) \) will be the cut of the output fuzzy set of in the universe of fan speed as shown in Figure 3. Hence we may use a user-defined \( \Delta \) as \( O^t \) for each output linguistic variable \( o_j \) as \( O^t(o) = \mu_{ij}(x) \) for the current input \( x \) that activates the input linguistic variable \( i_j \) and outputs \( o \).

![Figure 3. A traditional fuzzy inference for the example in Figure 1](image)

4. Implementation issues and fuzzy inference procedure

In this section, we describe the implementation of fuzzy rule base in hardware and propose the algorithm for fuzzy inference based on this hardware.

4.1. Implementation of fuzzy rules

Given fuzzy rule base, the following steps organize the rules and wire them.

**step 1** Convert the rules into the canonical form such as in Section 2.2. Next, group the rules which have the same universe output. Order these groups into stages according to the input-output dependency such that group \( G_u \) precedes \( G_v \) if its universe output required to be an input by group \( G_v \).

**step 2** Transform the rules into an FRA for each group \( G_u \). Here empty action (denoted by \( \phi \)) is introduced which implies that no actions are needed at this moment. Let
\{i_{k,l}\} be each input linguistic variable for rule \(R_k\) and \(l^{th}\) input variable when there is \(m\) rules in the group \(G_u\) and each rule has \(n\) input linguistic variables. For simplicity, we assume that inputs arrive in order \([1 \ldots n]\) defined the rule. Each rule \(R_k\) in the group \(G_u\) is conjunctive. Define the initial state \(s_0\). For every rule in group \(G_u\), and for each input variable \(i_{k,l}\), define a transition from \(s_i\) to \(s_j\) and label input \(i_{k,l}\). If \(l = n\), the output is \(o_k\). Otherwise, the output is \(\phi\). For the next input \(i_{k,l+1}\), construct a transition from state \(s_j\) and so on. For the first group \(G_0\), \(s_i = s_0\).

step 3 Convert the FRA into a combinational logic. The minimization of the classical automata may be applied so that the number of states is minimum. The state, input, output encodings may play roles for the minimum number of gates. The derived circuit implicitly represents the state transition relation and output relation.

Property 4.1 Given an FRA representing rule base, the derived hardware circuit implicitly contains the same set of rules but in the different representation.

Given a rule base, an equivalent FRA can be derived and a corresponding hardware circuit can be obtained. This hardware part encodes all linguistic variables into binary representations. Inputs to the system, converted into a binary representation, are fed to the logic gates which compute proper output bits that form the output linguistic variable according to the defined FRA.

From Figure 2, the state operations and user-defined output operations have to be identified in order to infer consequences. Obtaining the rule-based hardware, membership functions, defuzzification method, fuzzy implication, a user-defined output operation \(\Delta\) can be defined to obtain the consequences as in the traditional fuzzy reasoning. The state operations can be defined to compute next states as Definition 3.2.

4.2. Fuzzy inference procedure

Assume that each group of rules \(G_u\) requires \(d\) non-interactive inputs arriving in the order as in the antecedent part of the canonical-form rules. For one \(G_u\), its corresponding FRA has \(I = \{l_1, l_2, \ldots, l_k\}\) as input linguistic variables. \(S = \{s_1, s_2, \ldots, s_q\}\) are internal states and \(\mu_{ij}\) is a membership function of an input linguistic variable \(i_j\). Let \(S_1\) be a current state vector with length \(q\), initially store \(v_{n0}\), where \(v_{n0} = 1\) if \(s_1\) is initial state and the rest are zero. Let \(S_2\) be a next state vector, initially stored zero. Let output buffer be \(O\), initially stored zero, and \(X\) be a buffer which stores previously arrived inputs. The following procedure presents the inference process an input \(x\) for one group \(G_u\).

Inference(x)

1. for each \(i_j, 1 \leq j \leq k\)
2. do for each \(s_n, 1 \leq n \leq q\)
3. do if \(\mu_{lj}(x) > 0\) \& \& \& \(S_1[n] > 0\)
4. then \(m = \text{next}\_state(j, n)\)
5. if \(x\) is the last arrival input
6. then \(p = \text{output}(j, n)\)
7. \(O[p] = 1\)
8. \(\text{tmp} = \min(\mu_{lj}(x), S_1[n])\)
9. \(S_2[m] = \max(S_2[m], \text{tmp})\)
10. swap \((S_1, S_2)\) and reset \(S_2\) to all zeros
11. \(X = X\_U(x)\) /* accumulate previous inputs */
12. if \(x\) is the last arrival input
13. then calculate output \(O\) using \(\Delta(O, X, S1)\)
14. defuzzify using \(O\) and give a conclusion

In this procedure, \(\text{next}\_state\) sends inputs \((j, n)\) to hardware and returns the next state index \(m\). Similarly, \(\text{output}\) function returns index \(p\) indicating which output variable to be activated. Because of the fuzzy nature, each input \(x\) can be related to more than one membership function \(\mu_{lj}(x)\). Current input \(x\) can generate more than one outputs \((i_j)\) to the FRA hardware. Since for each \(x\), there can be more than one rules to fire, the FRA can generate more than one next states \((s_n)\). Thus, all possible states and outputs have to be gathered for each \((j, n)\).

The following theorem shows that this above inference procedure is equivalent to the traditional inference process.

Theorem 4.1 For a group of rules \(G_u\), given an FRA for rule base having \(d\) non-interactive inputs and one output, INFERENCE algorithm performs the same inference process as in the traditional fuzzy and after \(d \times q \times k\) iterations, the consequence can be obtained.

The derived circuit from this FRA model contains only combinational logic gates; therefore, it is easy to implement with low cost. We can duplicate the hardware to speedup the simulation process. The hardware circuit can be duplicated as many as the number of internal states times the number of input linguistic variables \((q \times k)\) so that we apply all of them in parallel. Thus, the number of iterations for rules in group \(G_u\) is reduced to \(d\). In order to perform the parallel inference on this model, extra hardwares such as multiplexers may be required to select which inputs and states should be fed into which circuit. The software is also modified in such a way that it can gather all outputs back correctly.

Let us briefly trace the inference procedure for the example in Figure 1 which has only one stage (or one group). This system has 3 states and 2 input variables; therefore, the two loops become \(1 \leq k \leq 2, 1 \leq q \leq 3\) respectively. The initial state vector is \([1, 0, 0]\). If a temperature input \(t\) is 65 and the membership values \([\mu_{cold}(t), \mu_{hot}(t)]\) are \([0.7, 0.2]\), next state vector \(S_2\) will be \([0, 0.2, 0.7]\) after the outer loop.
5. Conclusion

Most fuzzy rule-based system implementations are complicated because the nature of fuzzy sets requires special computation to derive a consequence. To implement a system, we partition the system into two parts: hardware and software to reduce the difficulties and yield rapid prototyping for fuzzy system implementations. The extra computations in fuzzy reasoning are implemented in software which gives more flexibility in system simulations. Hence, one can easily change the membership functions or fuzzy implications by editing software part. Because the rule base is seldom changed, they can be implemented in hardware to speed up the inference. The Fuzzy Rule-based Automata (FRA) is presented to model the rule base so that they can be rapidly transformed into simple hardware logic gates with low cost. We present a software inference procedure that controls this rule-based hardware and performs computations for the fuzzy inference process. This software incorporates built-in operations for calculating outputs which includes membership functions, fuzzy implicators, and defuzzification method. It sends proper inputs to hardware and gathers back outputs and next states. A user-defined output operation computes consequences, by using inputs as well as current outputs, which will be defuzzified at the last step.