Memorizing in Fuzzy Computers

MRAZ MIHA, OSELI DAMJAN, ZIMIC NIKOLAJ, VIRANT JERNEJ
Faculty of Computer and Information Science
University of Ljubljana
Trzaska 25, SI-1000 Ljubljana
SLOVENIJA

Abstract: - In this paper we present the basics of memorizing of fuzzy data, we analyze the meaning of memorizing of fuzzy data and we present basic entities used for memorizing. At the beginning we do a survey of the state of the art in the area of memorizing of fuzzy data which originates from the binary memorizing. In the continuation we present some problems and advantages which are the result of memorizing of fuzzy data with modern analogous devices. With the direct memorizing of fuzzy data the new possibilities rises for the implementation of fuzzy computer, which will deal directly with fuzzy data and fuzzy knowledge.

Key Words: - crisp bit, fuzzy bit, fuzzy computing device, fuzzy memory device, fuzzy memory cell.

1 Introduction

The computer's main features today are the possibilities of decision making, data transfer and memorizing. All of the features relate only to ordinary or crisp data. In some application cases such kind of data are not available. Instead of them only the uncertain, possible or ambiguous data and knowledge could be obtained. The typical case of application with such uncertain data and knowledge is the fire spread simulation application presented in [1].

The (ordinary) memorizing in the crisp or binary domain is founded with the main memorizing cells (flip-flops) as JK, SR, T and D are. From the viewpoint of the binary switching history the JK, SR and T flip-flops had a very big meaning due to their SET-RESET functions. From the viewpoint of today's computers architecture modeling, the main entity of memorizing is D flip-flop or the function of *delay*. Its capability is only memorizing the data for a single clock period without doing any kind of decision with logical operators.

A very fast progress of the theory of fuzzy and possibility logic in last two decades, which is based on uncertain data, has brought us also to the first research experiments and theories about fuzzy memorizing. The most of them have been done from Japanese authors Yamakawa, Hirota and Ozawa [2]. The authors have extended the classical memorizing cell JK to its fuzzy extension FJK. The authors of the present paper from Slovenia have done the extensions to SR (Set-Reset) and T memory cells ([3], [4] and [5]). Both of groups were primarily working on extensions to fuzzy variants of memory cells and to analyze their behavior.

The main lack of the today's theory of fuzzy memorizing is the explanation of its meaning. This rises questions as "how to fuzzy memorize", "what should be fuzzy memorized" and "why to fuzzy memorize something" etc.

In the present paper we'd like to propose some ideas to define the main entities of fuzzy memorizing to answer to the questions above.

2 Memorizing of crisp and fuzzy data

Let us suppose that we do some kind of processing on data, which represent the heights of the set of *n* people. In that case we have to deal with a vector of crisp values $H = (h_1, h_2, \ldots, h_n)$. In the domain of ordinary (binary) computing every value h_i from the vector is coded into its binary representation. If we limit ourselves to the integer values in vector *H* and also to the maximal height of 255 cm, the 175 cm is coded into the binary string 10101111. Eight binary memory locations are used in that case to memorize one *crisp data* from 0 to 255 cm. 0 or 1 could be stored into every memory location. The information of entity stored in a cell is called a (crisp) *bit*.

Let us suppose now that we have a fuzzy computing device, which deals only with the fuzzy data of people heights. In that case all of the input data have to be fuzzified using *m* fuzzy sets provided in advance for the fuzzification process.

For every h_i from the *H* vector the membership to individual fuzzy set is calculated. The input crisp data could be discarded after the fuzzification. Instead of the h_i (*crisp data*) after the fuzzification we got a *fuzzy data*, which is represented as a vector of memberships $V_i = (m_{\tilde{A}_j}(h_i)),\ 1 \le j \le m.$ We will use notation $\hat{h}_i$ for a fuzzy data, due to its vector structure.

Let us presume that there exists the memorizing entity with the capability of memorizing one membership value, which comes from the interval [0,1]. In that case we need *m* such memorizing entities to memorize the fuzzy data $\hat{h}_i$, represented with *m* memberships to fuzzy sets. Our proposal is that the quantity of information stored in such memorizing entity is called a *fuzzy bit*. So the quantity of fuzzy bit is a value *x* ($x \in [0,1]$). Due to Shannon's equation its information worth is much higher than of ordinary (crisp) bit. So the main difference between both kind of bits could be represented with relations:

$$\begin{aligned} &\text{(crisp) } data \rightarrow 8 \text{ (crisp) } bits, \\ &\text{(fuzzified) } fuzzy\ data \rightarrow m\ fuzzy\ bits. \end{aligned} \qquad (1)$$

The number of *m* fuzzy bits coincides with the number of fuzzy sets used for fuzzification of entering crisp data. The number of fuzzy sets determined in advance for fuzzification depends mainly on the application's specific requirements or granulation of classification process of data.

In the domain of today's binary computing the fuzzy bit is memorized as a real number stored in the exact number of crisp bits (places) determined in advance. Let us take for instance the case of the programming language C. It provides us with two basic types for real number representation of the length of 16 (`float`) or 32 (`double`) bits. Due to limited number of bits we have to know that the approximation of the real number is stored in that case, where the error of stored value is known.

In the domain of the computers of the future which will probably have the possibility of direct fuzzy processing, the fuzzy data will be stored in one "location" or memorizing entity due to the speed of reading and writing and it will be stored with the highest possible accuracy.

3 Present memorizing of fuzzy data

Let us define in advance two basic memorizing entities, memory device and memory cell. *Memory device* is the simplest form of a memorizing entity, which can memorize some data. It is a single input – single output entity. Input connection is used for writing in, while the output connection enables us to read back the stored data. The data can be read back after minimum of one clock period, since the behaviour of such entity has a delaying effect.

Memory cell is more complex entity as it can have several input and output connections. These connections allow us to integrate a functional behaviour in a cell in front of a memorizing entity. On that way the decision process is involved beside the memorizing capability into the memory cell. The decision process operates on the values of input variables and on the previos state(s) of the memory cell.

Both structures are presented on Figure 1. It is evident that the memory device is simpler than the memory cell. With other words, memory device is a part of a memory cell.

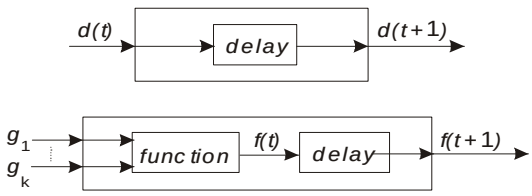

Figure 1: Memory device (above) and Memory cell (below).

Memory cells like JK, RS and T were the most important cells in the field of binary logic, because they inherited the basic concept of switching theory. The core of such cell consists of two inputs, set and reset, which are the switching inputs. As the domain of the switching is the logical value 0 and 1, the cells state can only be set in one of these states. Adding additional logic in front of the core part of the cell can improve the flexibility and diversity in the cell (e.g. JK, RS,T).

The meaning of such memory cells today is not so significant anymore, as most of the primitive and even some complex integrated devices do not use JK, RS and T cells as the memorizing entities. The only memorizing entity found in these devices is the D cell, because of its delay-memorizing effect.

In the field of fuzzy logic fuzzified versions of memory cells were proposed. The inherited fuzzified cells are FJK, FRS, FTDC, FT and the rest of them ([2], [3], [4], [5]). It is worth mentioning that each of

these cells is directly derived from its binary ancestor with the method of fuzzifying the accompanying input and output variables. As the binary logic is a subset of the fuzzy logic the reverse engineering of the knowledge from binary to the fuzzy is not uniform. Fuzzy variable can take any of the values from the interval [0,1]. The memory cell must be able to memorize any real number from this interval. The first problem is the 0.5 state lockup. The derived fuzzy memory cell *i* (e.g. FJK, FRS,…) can slip into the 0.5 state *q* in the time *t*. After that, the cell can not change its state regardless of the input value. This lockup can be overridden with the method of adjusting the cell's behavior as described in [5]. Another problem is from the theoretical background of the crisp logic. Mathematical models of all basic binary memory cells D, JK, RS and T can be derived from the main memorizing equation (2).

$$q(t+1) = g_1 q(t) + g_2 \bar{q}(t). \qquad (2)$$

Because each of the mentioned cells is derived from the same equation, we can construct every cell with the other one [6]. Fuzzified memory cells like FJK, FRS, etc. does not comply with the equation (2) anymore. Equation (3) stands for the binary D memory cell and is derived from the equation (2). Lets consider input variable *q* a fuzzy variable with the [0,1] value space. If we want to memorize this fuzzy data with the fuzzified D memory cell the equation (3) does not stand anymore. Instead of it the transformed equation presented in (4) is used.

$$q(t+1) = dq(t) + d\bar{q}(t) = d(q(t) + \bar{q}(t)) = d,$$
$$q(t) + \bar{q}(t) = 1, \qquad (3)$$

$$q(t+1) = dq(t) + d\bar{q}(t) = d(q(t) + \bar{q}(t)) \neq d,$$
$$0.5 \leq (q(t) + \bar{q}(t)) \leq 1. \qquad (4)$$

Memory equation (4) belongs to the fuzzified D memory cell, which we call FD memory cell.

All the fuzzified memory cells, which are derived from the main memory equation (2) share the same characteristic of the fuzzy complement.

$$q \vee \bar{q} = 1,$$
$$q \& \bar{q} = 0. \qquad (5)$$

$$q \vee \bar{q} \geq 0.5,$$
$$q \& \bar{q} \leq 0.5. \qquad (6)$$

In (5) the crisp logic complement properties are stated. Applying *q* a fuzzy data the (5) is not valid. The properties (6) are then valid for the fuzzy input data *q* and fuzzy memorizing. Fuzzy complement properties are described thoroughly in [8].

As explained in this section the crisp logic theoretical backgrounds of the memory cells accompany the fuzzified F memory cells. The inherited problems cause the unusual behavior of the fuzzified memory cells, which is not desired and must be overridden in some way.

4 Delaying fuzzy data

Fuzzy memorizing in its basic form must be capable of memorizing the fuzzy number as a single fuzzy bit entity. To memorize the fuzzy number a one single memorizing entity can be used. Values, the fuzzy bit can take, are real numbers from the interval [0,1]. To achieve such memorizing two primitive analog devices can be used, which are capable of memorizing values from the closed, continuous interval. These devices are

- `Sample&Hold` (S/H) and
- `DelayLine` (DL).

`Sample&Hold` device is basically a single input – single output primitive device, which is capable of reading (sampling) the input value and holding it stable at the output, until new input value overrides the previous one. Output value is delayed for a minimum time (in range of nanoseconds) regarding to input value write trigger. This kind of memorizing is of asynchronous type. Input values causes output to follow the input. Similar S/H device is used in hardware realizations of fuzzy controllers by Yamakawa. There is difference that Yamakawa memorizes output analog value, which has already passed the defuzzification process and is as such a crisp value [7].

`DelayLine` device can perform synchronous memorizing of analog input data with a constant delay.

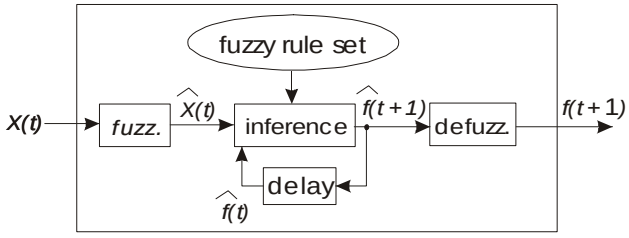

Figure 2: Fuzzy computer structure with the capability of memorizing membership grades.

In Figure 2 we schematically present a fuzzy computer. The `delay` block is used to memorize in-process membership grades to fuzzy sets. Practically we can realize such memorizing of membership grades with the *m* S/H devices, where *m* stands for the number of fuzzy sets of input variable. The memorized data are taken back to the inference engine for the processing. The `inference` block is used only for processing of the rules from the `fuzzy_rule_set.` All memorizing is moved to the `delay` block. With excluding memory functionality from the inference engine, we can build separate block of memory cells.

Structure of such separated memory block is showed in the Figure 3. Time variable t and t+1 define the delay between the input data trigger time and output value stabilization time. Exact value of this delay is not important when talking on the abstract layer. We name this delay as one clock or a one time-stamp delay.

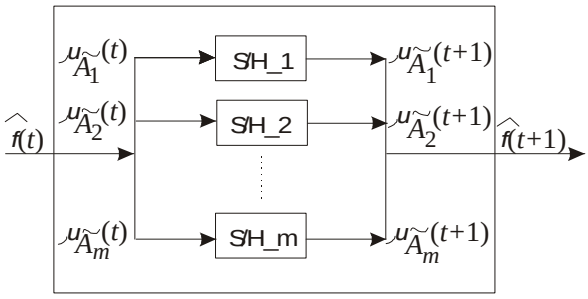

Figure 3: Memory delay block realization with `Sample&Hold` primitive devices

The size of such memory block is measured with the number of basic memorizing entities (e.g. `Sample&Hold` devices) working in parallel.

5 Conclusion

With intention to construct a general memory which could memorize fuzzy data we review the existing approaches to fuzzy memorizing and suggest separation of inference and memorizing capability as currently found joined in the fuzzy JK, RS and T cells. As demonstrated we can realize memorizing of fuzzy data with the field of S/H primitive devices. These devices have pure memorizing capabilities. Such approach is more suitable than currently popular use of memorizing and inference simultaneously as used in today's JK, RS and T cells and also their fuzzified ancestors, F cells.

References:

[1] M. Mraz, N. Zimic and J. Virant, Intelligent bush fire spread prediction using fuzzy cellular automata, *Journal of Intelligent and Fuzzy Systems*, Vol.7, 1999, pp. 203-207.

[2] K. Ozawa, K. Hirota and L. T. Koczy, *Fuzzy Flip-flop* (in *Fuzzy Logic, Implementation and Applications*), editors M. Y. Patyra and D. M. Mlynek, Wiley and Sons and B.G.Teubner Publishers, 1996, pp.197-236.

[3] J. Virant, N. Zimic and M. Mraz, T-type fuzzy memory cells, *Fuzzy sets and systems*, Vol.102, 1999, pp.175-183.

[4] J. Virant, N. Zimic and M. Mraz, *Fuzzy Sequential Circuits and Automata* (in *Fuzzy Theory Systems: Techniques and Applications*), editor C.T. Leondes, Academic Publishers, 1999, pp.1599-1653.

[5] J. Virant, *Design considerations of time in fuzzy systems*, Kluwer Academic Publishers, 2000.

[6] Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill, 1978.

[7] T. Yamakawa, Fuzzy inference on an analog fuzzy chip, *IEEE Micro*, August 1995.

[8] G.J.Klir, B.Yuan, Fuzzy sets and fuzzy logic – Theory and applications, Prentice Hall, 1995.