

Concept of Level-Controlled R-S Fuzzy Memory Circuit

Milan Petřík

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
petrikm@cmp.felk.cvut.cz

Abstract

In this paper we present a level-controlled fuzzy memory element based on a generalization of the classical two-valued R-S flip-flop. Our memory element is small, simple and hazardless and thus suitable for a hardware implementation. We show that the circuit is stable with respect to small errors of input signals as well as to small errors of the gates; we show that this stability is conditioned by restricting the set of fuzzy values to a finite subset of the interval $[0, 1]$.

Keywords: Many-valued logic, fuzzy logic, hardware design, logical circuit design, sequential logic, memory element, latch, flip-flop.

1 Introduction

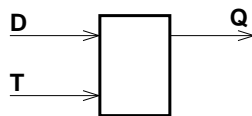


Figure 1: Scheme of a general memory element.

A *memory element*¹ is a logical circuit able to remember a logical value and to hold it on its output for a defined period of time independently of its input.

The importance of memory elements in the hardware design is crucial considering that they enable implementation of sequential logical circuits, i.e. logical circuits where the output value depends

also on the previous values of the input. Furthermore, they allow to build registers, to synchronize combinational logical circuits, etc.

Therefore if we want to design a many-valued hardware at least as strong as the classical two-valued one, we will need a way of implementing a many-valued memory element.

Fuzzy memory elements have been already studied in a number of papers². We attempt to propose a different solution and we give a comparison in the conclusion. We also try to investigate the circuit stability bearing in mind possible gate defects.

In Fig. 1 you can see a scheme of a general memory element. Input signal T (called usually *clock* or *time* signal) determines whether the memory element is *open* (the value of input D is “remembered” and shall appear on output Q) or it is *closed* (the old value of output Q is kept regardless of input D).

Two groups of memory elements are distinguished: *Level-controlled* and *edge-controlled*. Level-controlled memory elements are open or closed depending on the value of the input T while edge-controlled memory elements are open when input T changes its value from one defined value to another defined value, otherwise they are closed. Although edge-controlled memory elements have better properties, in this paper we are interested in level-controlled memory elements since their construction is simpler. The construction of edge-controlled memory elements usually requires level-controlled memory elements

¹See e.g. [1, 4, 7].

²See e.g. [2, 3, 5, 6, 8, 9, 10]

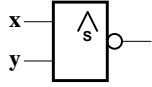


Figure 2: Gate representation of the standard many-valued Sheffer operation.

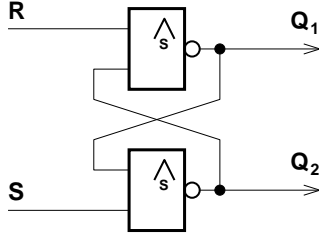


Figure 3: Scheme of a many-valued R-S circuit.

as building blocks and shall be discussed in a further paper.

2 Many-valued R-S circuit

Remark 2.1 We use apostrophes to distinguish between previous and next values of signals Q_1 and Q_2 . Thus e.g. Q'_1 is the next value of Q_1 , Q''_1 is the next value of Q'_1 , etc. ■

Definition 2.2 The standard many-valued Sheffer operation is a binary operation $\overline{\wedge}_S: [0, 1] \times [0, 1] \rightarrow [0, 1]$ defined as follows:

$$\overline{\wedge}_S(a, b) = \overline{a \wedge_S b} = 1 - \min(a, b)$$

Definition 2.3 Many-valued R-S circuit, an R-S circuit for short, is a logical circuit consisting of two gates implementing the standard many-valued Sheffer operation (see Fig. 2b), two input ports R, S , and two output ports Q_1, Q_2 , connected as shown in Fig. 3.

Theorem 2.4 The R-S circuit with input ports R, S and output ports Q_1, Q_2 satisfies: If $R \leq \overline{S}$ then $Q'_1 = \overline{R}$ and $Q'_2 = \overline{S}$. In particular, if $R = \overline{S}$ then $Q'_1 = S$ and $Q'_2 = R$. This result is independent of the initial values Q_1 and Q_2 .

Proof

1. If $R \leq Q_2^0$ then the next value of Q_1 is:
 $Q'_1 = \overline{\wedge}_S(Q_2^0, R) = \overline{R}$.

Since the circuit develops in time, we can use Q'_1 instead of Q_1^0 to compute the value of Q'_2 :
 $Q'_2 = \overline{\wedge}_S(Q_1^0, S) = \overline{\wedge}_S(Q'_1, S) = \overline{\wedge}_S(\overline{R}, S)$.

Due to $R \leq \overline{S}$, we get $\overline{R} \geq S$ and $Q'_2 = \overline{S}$.

2. If $S \leq Q_1^0$ then the situation is dual to the previous case:
3. If $R > Q_2^0$ and $S > Q_1^0$ then the next values of Q_1 and Q_2 are:
 $Q'_1 = \overline{\wedge}_S(Q_2^0, R) = \overline{Q_2^0}$,
 $Q'_2 = \overline{\wedge}_S(Q_1^0, S) = \overline{Q_1^0}$.

Now we compute the next values of Q'_1 and Q'_2 and we get:

$$Q''_1 = \overline{\wedge}_S(Q'_2, R) = \overline{\wedge}_S(\overline{Q_1^0}, R),$$

$$Q''_2 = \overline{\wedge}_S(Q'_1, S) = \overline{\wedge}_S(\overline{Q_2^0}, S).$$

Negating the inequalities $R > Q_2^0$ and $S > Q_1^0$, we get $\overline{R} < \overline{Q_2^0}$ and $\overline{S} < \overline{Q_1^0}$. Due to $\overline{R} \geq S$ and $R \leq \overline{S}$ we get $S < \overline{Q_2^0}$ and $R < \overline{Q_1^0}$ which give us the final result:

$$Q''_1 = \overline{R},$$

$$Q''_2 = \overline{S}. \quad \blacksquare$$

Theorem 2.5 The R-S circuit with input ports R, S and output ports Q_1, Q_2 satisfies: If $Q_1 = \overline{Q_2}$, $S > Q_1$ and $R > Q_2$ then $Q'_1 = Q_1$ and $Q'_2 = Q_2$.

Proof

$$Q'_1 = \overline{\wedge}_S(Q_2^0, R) = \overline{Q_2^0} = Q_1^0$$

$$Q'_2 = \overline{\wedge}_S(Q_1^0, S) = \overline{Q_1^0} = Q_2^0 \quad \blacksquare$$

3 Unstable and invalid states of the many-valued R-S circuit

Theorem 3.1 The R-S circuit with input ports R, S and output ports Q_1, Q_2 satisfies: If $R > \overline{S}$ then $\overline{R} \leq Q'_1 \leq S$ and $\overline{S} \leq Q'_2 \leq R$.

Proof We assume without loss of generality that $R \leq S$. Let us discuss all possible initial values of outputs Q_1^0 and Q_2^0 .

1. If $Q_1^0 \geq S$ and $Q_2^0 \geq R$ then:
 $Q'_1 = \overline{\wedge}_S(Q_2^0, R) = \overline{R}$

$$\begin{aligned} Q'_2 &= \overline{\delta}(Q_1^0, S) = \overline{S} \\ Q''_1 &= \overline{\delta}(Q'_2, R) = \overline{\delta}(\overline{S}, R) = S \\ Q''_2 &= \overline{\delta}(Q'_1, S) = \overline{\delta}(\overline{R}, S) = R \\ Q'''_1 &= \overline{\delta}(Q''_2, R) = \overline{\delta}(R, R) = \overline{R} \\ Q'''_2 &= \overline{\delta}(Q''_1, S) = \overline{\delta}(S, S) = \overline{S} \end{aligned}$$

Output Q_1 oscillates between \overline{R} and S as well as output Q_2 oscillates between \overline{S} and R .

2. If $Q_1^0 \geq S$ and $Q_2^0 < R$ then:

$$\begin{aligned} Q'_1 &= \overline{\delta}(Q_2^0, R) = \overline{R} \\ Q'_2 &= \overline{\delta}(Q_1^0, S) = \overline{Q_1^0} \\ Q''_1 &= \overline{\delta}(Q'_2, R) = \overline{\delta}(\overline{Q_1^0}, R) \\ Q''_2 &= \overline{\delta}(Q'_1, S) = \overline{\delta}(\overline{R}, S) = R \end{aligned}$$

Since $\overline{Q_1^0}$ can be either less or more than R we are not able to evaluate the expression $\overline{\delta}(\overline{Q_1^0}, R)$. Let us discuss both possibilities:

(a) If $\overline{Q_1^0} \geq R$ then $Q''_1 = \overline{S}$ and:

$$\begin{aligned} Q'''_1 &= \overline{\delta}(Q''_2, R) = \overline{\delta}(R, R) = \overline{R} \\ Q'''_2 &= \overline{\delta}(Q'''_1, S) = \overline{\delta}(S, S) = \overline{S} \\ Q''''_1 &= \overline{\delta}(Q'''_2, R) = \overline{\delta}(S, R) = \overline{R} \\ Q''''_2 &= \overline{\delta}(Q''''_1, S) = \overline{\delta}(\overline{R}, S) = R \\ Q'''''_1 &= \overline{\delta}(Q''''_2, R) = \overline{\delta}(R, R) = \overline{R} \\ Q'''''_2 &= \overline{\delta}(Q'''''_1, S) = \overline{\delta}(\overline{R}, S) = R \end{aligned}$$

Outputs Q_1 resp. Q_2 are both constant with the value \overline{R} , resp. R .

(b) If $\overline{Q_1^0} < R$ then $Q''_1 = Q_1^0$ and:

$$\begin{aligned} Q'''_1 &= \overline{\delta}(Q''_2, R) = \overline{\delta}(R, R) = \overline{R} \\ Q'''_2 &= \overline{\delta}(Q''_1, S) = \overline{\delta}(Q_1^0, S) = \overline{Q_1^0} \\ Q''''_1 &= \overline{\delta}(Q'''_2, R) = \overline{\delta}(\overline{Q_1^0}, R) = Q_1^0 \\ Q''''_2 &= \overline{\delta}(Q''''_1, S) = \overline{\delta}(\overline{R}, S) = R \quad Q'''''_1 = \\ &= \overline{\delta}(Q''''_2, R) = \overline{\delta}(R, R) = \overline{R} \\ Q'''''_2 &= \overline{\delta}(Q'''''_1, S) = \overline{\delta}(Q_1^0, S) = \overline{Q_1^0} \end{aligned}$$

Output Q_1 oscillates between \overline{R} and Q_1^0 and output Q_2 oscillates between $\overline{Q_1^0}$ and R .

3. If $Q_1^0 < S$ and $Q_2^0 \geq R$ then the situation is dual to the previous case.

4. If $Q_1^0 < S$ and $Q_2^0 < R$ then:

$$\begin{aligned} Q'_1 &= \overline{\delta}(Q_2^0, R) = \overline{Q_2^0} \\ Q'_2 &= \overline{\delta}(Q_1^0, S) = \overline{Q_1^0} \end{aligned}$$

$$Q''_1 = \overline{\delta}(Q'_2, R) = \overline{\delta}(\overline{Q_1^0}, R)$$

$$Q''_2 = \overline{\delta}(Q'_1, S) = \overline{\delta}(\overline{Q_2^0}, S)$$

Since $\overline{Q_1^0}$ resp. $\overline{Q_2^0}$ can be both either less or more than R , resp. S , we are not able to evaluate the expressions $\overline{\delta}(\overline{Q_1^0}, R)$ resp. $\overline{\delta}(\overline{Q_2^0}, S)$. Let us discuss all possibilities:

(a) If $\overline{Q_1^0} \geq R$ and $\overline{Q_2^0} \geq S$ then:

$$\begin{aligned} Q'''_1 &= \overline{R} \\ Q'''_2 &= \overline{S} \\ Q''''_1 &= \overline{\delta}(Q'''_2, R) = \overline{\delta}(\overline{S}, R) = S \\ Q''''_2 &= \overline{\delta}(Q''''_1, S) = \overline{\delta}(S, S) = R \\ Q'''''_1 &= \overline{\delta}(Q''''_2, R) = \overline{\delta}(R, R) = \overline{R} \\ Q'''''_2 &= \overline{\delta}(Q'''''_1, S) = \overline{\delta}(S, S) = \overline{S} \end{aligned}$$

Output Q_1 oscillates between \overline{R} and S and output Q_2 oscillates between \overline{S} and R .

(b) If $\overline{Q_1^0} \geq R$ and $\overline{Q_2^0} < S$ then:

$$\begin{aligned} Q'''_1 &= \overline{R} \\ Q'''_2 &= Q_2^0 \\ Q''''_1 &= \overline{\delta}(Q'''_2, R) = \overline{\delta}(Q_2^0, R) = \overline{Q_2^0} \\ Q''''_2 &= \overline{\delta}(Q''''_1, S) = \overline{\delta}(\overline{Q_2^0}, S) = R \\ Q'''''_1 &= \overline{\delta}(Q''''_2, R) = \overline{\delta}(R, R) = \overline{R} \\ Q'''''_2 &= \overline{\delta}(Q'''''_1, S) = \overline{\delta}(\overline{Q_2^0}, S) = Q_2^0 \end{aligned}$$

Output Q_1 oscillates between \overline{R} and $\overline{Q_2^0}$ and output Q_2 oscillates between Q_2^0 and R .

(c) If $\overline{Q_1^0} < R$ and $\overline{Q_2^0} \geq S$ then:

$$\begin{aligned} Q'''_1 &= Q_1^0 \\ Q'''_2 &= \overline{S} \\ Q''''_1 &= \overline{\delta}(Q'''_2, R) = \overline{\delta}(\overline{S}, R) = S \\ Q''''_2 &= \overline{\delta}(Q''''_1, S) = \overline{\delta}(Q_1^0, S) = \overline{Q_1^0} \\ Q'''''_1 &= \overline{\delta}(Q''''_2, R) = \overline{\delta}(\overline{Q_1^0}, R) = Q_1^0 \\ Q'''''_2 &= \overline{\delta}(Q'''''_1, S) = \overline{\delta}(S, S) = \overline{S} \end{aligned}$$

Output Q_1 oscillates between Q_1^0 and S and output Q_2 oscillates between \overline{S} and Q_1^0 .

(d) If $\overline{Q_1^0} < R$ and $\overline{Q_2^0} < S$ then:

$$\begin{aligned} Q'''_1 &= Q_1^0 \\ Q'''_2 &= Q_2^0 \\ Q''''_1 &= \overline{\delta}(Q'''_2, R) = \overline{\delta}(Q_2^0, R) = \overline{Q_2^0} \\ Q''''_2 &= \overline{\delta}(Q''''_1, S) = \overline{\delta}(Q_1^0, S) = \overline{Q_1^0} \\ Q'''''_1 &= \overline{\delta}(Q''''_2, R) = \overline{\delta}(\overline{Q_2^0}, R) = Q_1^0 \end{aligned}$$

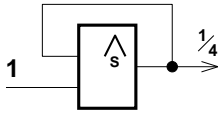


Figure 4: Example of a simple logical circuit with a feedback.

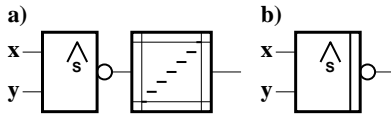


Figure 5: a - standard Sheffer operation filtered to a finite set of logical values, b - schematic symbol of a filtered standard Sheffer operation

$$Q_2''' = \overline{\overline{S}}(Q_1''', S) = \overline{\overline{S}}(\overline{Q_2^0}, S) = Q_2^0$$

Output Q_1 oscillates between Q_1^0 and $\overline{Q_2^0}$ and output Q_2 oscillates between Q_2^0 and $\overline{Q_1^0}$.

■

Corollary 3.2 Since $\overline{R} \leq Q_1' \leq S$ and $\overline{S} \leq Q_2' \leq R$ for every R and S (see Theorem 2.4 and Theorem 3.1), we can say that if we store a value in an R-S circuit and $|\overline{R} - \overline{S}| = |\overline{R} - S| = \Delta$, then $Q_1 \in [S - \Delta, S + \Delta]$ and $Q_2 \in [R - \Delta, R + \Delta]$.

Corollary 3.3 If a value is kept in an R-S circuit, $S > Q_1$, $R > Q_2$ but $Q_1 \neq \overline{Q_2}$ then output Q_1 attains a value from the interval delimited by Q_1 and $\overline{Q_2}$. Output Q_2 attains a value from the interval delimited by Q_2 and $\overline{Q_1}$.

Corollary 3.4 If any of the values of R , S , Q_1 or Q_2 has an error, then an error not greater than the sum of these errors may appear on outputs Q_1 and Q_2 .

Constructing a logical circuit the gates have an error, i.e. they give a bigger or smaller result than is the correct result of the standard Sheffer

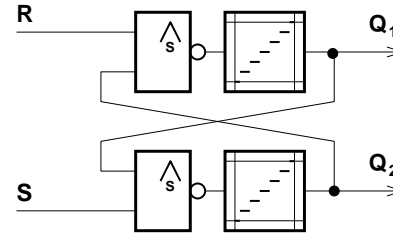


Figure 6: Scheme of a many-valued R-S circuit with added filters; the filters filtrate the output to a finite set of logical values.

operation they implement. While in two-valued logical circuits this problem is negligible in many-valued logical circuits with feedback it can harm the whole functionality of the circuit. We can illustrate this problem on an example of a simple circuit shown in Fig. 4. If the gate performs the operation of minimum without an error then the circuit keeps the value $\frac{1}{4}$ on its output. Nevertheless when the gate gives a lesser value then it is easy to imagine that the output value shall decrease until the bound 0 is reached.

The same problem may appear in our memory element. A possible solution is to add a filter at the end of both gates; this filter restricts the values from the interval $[0, 1]$ to a finite set of discrete values spread uniformly on the interval $[0, 1]$. If the distance between two neighbouring values is at least twice the error of the gates then the impact of the gate errors will be eliminated and the functionality of the R-S circuit will not be harmed. A scheme of an R-S circuit with added filters can be seen in Fig. 6.

Note that the number of logical values which may be passed out of the filter is in general arbitrary but depends on the errors of the gates.

4 Memory elements based on the many-valued R-S circuit

Definition 4.1 Many-valued R-S level-controlled memory element, an R-S latch for short, is a logical circuit consisting of four gates implementing the standard many-valued Sheffer operation, two of them filtered, three

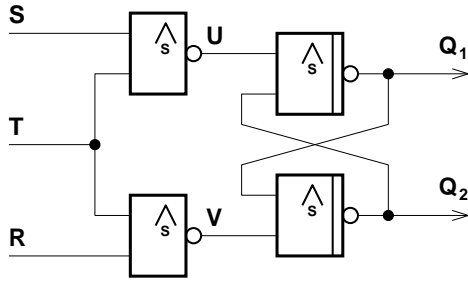


Figure 7: Scheme of a many-valued R-S level-controlled memory element.

input ports R , S , T and two output ports Q_1 , Q_2 , connected as shown in Fig. 7.

Theorem 4.2 *The R-S latch with input ports R , S , T and output ports Q_1 , Q_2 satisfies: If $R \geq \bar{S}$ and $T = 1$ then $Q_1' = S$ and $Q_2' = R$. This result is independent of the previous values of Q_1 and Q_2 .*

Proof Since $T = 1$,
 $V = \bar{S}(R, T) = \bar{S}(R, 1) = \bar{R}$ and
 $U = \bar{S}(S, T) = \bar{S}(S, 1) = \bar{S}$.

Due to $R \geq \bar{S}$, U and V satisfy $U \leq \bar{V}$ and the values are passed to the output as in the R-S circuit described in Theorem 2.4:

$$\begin{aligned} Q_1' &= \bar{U} = S, \\ Q_2' &= \bar{V} = R. \end{aligned}$$

Theorem 4.3 *The R-S latch with input ports R , S , T and output ports Q_1 , Q_2 satisfies: If R and S are constant, $R = \bar{S}$ and T is changing its value from 1 to 0, then $Q_1 = S$ and $Q_2 = R$ for the whole period of the process until $T = 0$.*

Proof We assume without loss of generality that $S \geq R$. The process can be divided into four steps:

1. $T = 1$:
 $Q_1' = S$,
 $Q_2' = R$
 (see Theorem 4.2).
2. $1 > T \geq S$:
 $U = \bar{S}(S, T) = \bar{S}$,

$$V = \bar{S}(R, T) = \bar{R}$$

and due to $R = \bar{S}$ we get:

$$Q_1'' = \bar{S}(U, Q_2') = \bar{S}(\bar{S}, R) = S,$$

$$Q_2'' = \bar{S}(V, Q_1') = \bar{S}(\bar{R}, S) = R.$$

3. $S > T \geq R$:

$$U = \bar{S}(S, T) = \bar{T},$$

$$V = \bar{S}(R, T) = \bar{R}.$$

Negating the inequality $S > T$ we get $\bar{S} < \bar{T}$:

$$Q_1''' = \bar{S}(U, Q_2'') = \bar{S}(\bar{T}, R) = \bar{S}(\bar{T}, \bar{S}) = S,$$

$$Q_2''' = \bar{S}(V, Q_1'') = \bar{S}(\bar{R}, S) = R.$$

4. $R > T \geq 0$:

$$U = \bar{S}(S, T) = \bar{T},$$

$$V = \bar{S}(R, T) = \bar{T}.$$

Negating the inequalities $S > T$, $R > T$ we get $\bar{S} < \bar{T}$, $\bar{R} < \bar{T}$:

$$Q_1'''' = \bar{S}(U, Q_2''') = \bar{S}(\bar{T}, R) = \bar{S}(\bar{T}, \bar{S}) = S,$$

$$Q_2'''' = \bar{S}(V, Q_1''') = \bar{S}(\bar{T}, S) = \bar{S}(\bar{T}, \bar{R}) = R.$$

Theorem 4.4 *The R-S latch with input ports R , S , T and output ports Q_1 , Q_2 satisfies: If $T = 0$ and $Q_1 = \bar{Q}_2$ then Q_1 and Q_2 keep their values, i.e. $Q_1' = Q_1$ and $Q_2' = Q_2$, independently of the values of inputs R and S until the value of input T is greater than 0.*

Proof If $T = 0$ then $U = 1$ and $V = 1$; Q_1 and Q_2 then keep their values as described in Theorem 2.5.

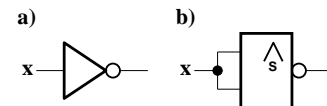


Figure 8: Gate representations of the standard negation: a - standard negation $\bar{x} = 1 - x$, b - standard negation implemented by the standard Sheffer operation $\bar{S}(x, x) = \bar{x}$.

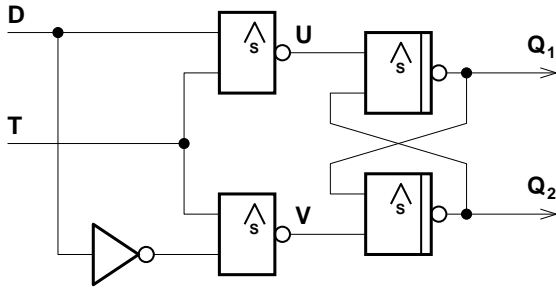


Figure 9: Scheme of a many-valued D level-controlled memory element.

Negating input R in an R-S latch by a gate implementing the standard negation (see Fig. 8) we get a *many-valued D level-controlled memory element* as shown in Fig. 9. It passes the value of input D to the output Q (i.e. $Q'_1 = D$ and $Q'_2 = \overline{D}$) when the value of input T is 1; when the value of input T is 0, the circuit keeps its old output value (i.e. $Q'_1 = Q_1$ and $Q'_2 = Q_2$).

5 Conclusion

We have designed a many-valued level-controlled memory element which can be implemented by four gates performing the standard fuzzy Sheffer operation, one gate performing the standard fuzzy negation, and two gates restricting the infinite interval $[0, 1]$ to its finite subset.

We have proven that the memory element is able to remember a many-valued logical value if the value of signal T is 1 and keeps this value if $T = 0$. We have also proven that this value is not lost when signal T changes its value continuously from 1 to 0 which is an important fact.

Comparing to the many-valued J-K memory element proposed by Ozawa, Hirota and Koczy in [2, 5, 6, 8] this solution is not based on an investigation of a characteristic equation of a memory element. Our starting point was the classical two-valued R-S circuit which we have generalized finding appropriate many-valued operations. This way we have obtained a functional

memory element implemented by two gates performing the standard Sheffer operation while the smallest memory element proposed Ozawa, Hirota and Koczy has four gates performing maximum or minimum and two negation gates. Moreover we have introduced a level-controlled memory element.

We have also discussed the problems with defective gates. We have shown that an error on the output of a gate may harm the proper function of a many-valued logical circuit with a feedback. We have proposed a solution using gates restricting the interval $[0, 1]$ to its finite discrete subset. Let us remark that to keep the stability the memory elements of Ozawa, Hirota and Koczy shall need these gates as well.

Note that analogically we may design a level-controlled memory element using gates implementing standard fuzzy negation of maximum. The behavior of this memory element will be dual to the behavior of the memory element presented here.

References

- [1] J. Bokr and V. Jáneš. *Logical Systems*. CTU, Prague, 1999, in Czech.
- [2] K. Hirota and K. Ozawa. Fuzzy flip-flop and fuzzy registers. *Fuzzy Sets and Systems*, 32:139–148, 1989.
- [3] K. Hirota and W. Pedrycz. Designing sequential systems with fuzzy J-K flip-flops. *Fuzzy Sets and Systems*, 39:261–278, 1991.
- [4] G. E. Hoernes and M. F. Heilweil. *Introduction to Boolean Algebra and Logic Design*. McGraw Hill, New York, San Francisco, Toronto, London, 1964.
- [5] L. T. Kóczy, K. Hirota, and K. Ozawa. Knowledge representation and accumulation by fuzzy flip-flops. *Fuzzy Sets and Systems*, 39:1–13, 1991.
- [6] L. T. Kóczy and K. Ömori. Algebraic fuzzy flip-flop circuits. *Fuzzy Sets and Systems*, 39:215–226, 1991.

- [7] H. Kubátová and Z. Blažek. *Logical Systems: Exercises*. CTU, Prague, 1999, in Czech.
- [8] K. Ozawa, K. Hirota, and L. T. Kóczy. *Fuzzy Logic, Implementation and Applications*, chapter Fuzzy Flip-flop, pages 197–236. Wiley and Teubner, 1996.
- [9] W. Pedrycz. *Fuzzy Sets Engineering*, chapter Fuzzy flip-flops in information processing, pages 223–252. CRC Press, 1995.
- [10] J. Virant, N. Zimic, and M. Mraz. T-type fuzzy memory cells. *Fuzzy Sets and Systems*, 102:175–183, 1999.