A TOOLBOX FOR FUZZY LOGIC FUNCTIONS SYNTHESIS ON A CHOICE TABLE

Dmitry A. Zaitsev
Computer Science Department
Vistula University
st. Stoklosy, 3
Warsaw 02-787, Poland
E-mail: daze@acm.org

KEYWORDS

Fuzzy logic function; choice table; synthesis; constituent of maximum; disjunctive normal form; partitioning

ABSTRACT

A toolbox for fuzzy logic functions synthesis on a choice table, available for free download on GitHub, has been implemented in C language. The early published method of a continuous (fuzzy) logic function synthesis on a choice table has been adjusted for fast partitioning the source choice table with a set of fuzzy logic functions. The toolbox implements a command line style of programming using data located in textual files of simple intuitive formats. The toolbox can process big data rapidly and can be easily integrated into fuzzy logic frameworks as a synthesis engine for developing graphical environment of fuzzy (control) systems design. Formal aspects of optimality (minimalism) are directions for future research.

INTRODUCTION

Fuzzy sets and fuzzy logic find wide application in intelligent and control systems design (Kaufmann 1977; Kandel 1986; Novak et al 2016). Since a fuzzy logic function (FLF) of Zadeh (Zadeh 1965) takes value of one of its arguments or a negation of an argument, the function can be given by a choice table (CT) (Volgin and Levin 1990) where all the variants of ordering of arguments and their negations are listed and for each variant the function value is pointed out. Without loss of generality, we suppose that an FLF is represented in disjunctive normal form (DNF). The choice table is considered as an analog to the truth table of the conventional binary logic (Kleene 1967) because it gives a simple way to compare FLFs via building and comparing their CTs. However, not any choice table defines a (single) fuzzy logic function; a table can be partitioned and covered by a set of fuzzy logic functions valid on subdomains (Zaitsev et al 1998). Topics of fuzzy logic synthesis from the behavioral description have been studied in (Wielgus 2004).
In (Zaitsev et al 1998), a criterion to tell whether a given choice table defines a fuzzy logic function has been introduced and an algorithm of a fuzzy logic function synthesis has been constructed. The total time complexity of the algorithm is linear in the table length. However, the calculation of the criterion is square in time with respect to the table length. In case a table does not define a fuzzy logic function, a sophisticated procedure is stipulated to find subsets of the table rows which are not mutually overlapped. The time complexity of this procedure can be rather great taking into consideration that the table length is an exponent in the number of the function arguments. Fuzzy logic operations enter the state equation of Petri nets with multichannel transitions (Zaitsev and Sleptsov 1997) which have been recently applied as a concurrent programming language (Zaitsev and Jurjens 2016). Colored Petri nets (Zaitsev and Shmeleva 2006) use similar concepts; their recent application for modeling grid structures (Zaitsev et al 2016), based on a substrate (Shmeleva et al 2009), revealed complex deadlocks.
The present work introduces a simple heuristic technique which accomplishes the results of (Zaitsev et al 1998) regarding partitioning a given choice table into a set of subdomains (subsets of the table rows) and synthesizing a separate fuzzy logic function for each partition. The total time complexity of the technique is linear in the number of the table rows. A toolbox, available for free download on GitHub, has been developed; it contains the following tools: synthesize a DNF on a choice table; create a choice table on a DNF; check whether two functions/tables coincide; partition a choice table into subdomains supplied with a set of DNFs valid for each subdomain; generate a random choice table. The partitioning technique has been justified statistically on sets of random choice tables. The formal aspects of optimality (minimalism) are left beyond the scope of the present paper as a direction for future research.

BASIC CONCEPTS AND NOTIONS

For a *domain* represented with an interval of real numbers $D = [0,1]$, operations of *conjunction, disjunction, and negation*, are introduced as follows:

$$\begin{aligned} x \wedge y &= \min(x,y), \\ x \vee y &= \max(x,y), \\ \bar{x} &= 1 - x, \end{aligned} \tag{1}$$

respectively, where $x, y \in D$.
A *fuzzy logic function* of n arguments is a function $f: D^n \to D$ obtained as a result of superposition of operations (1) on independent variables $x_1, x_2, \dots, x_n \in D$. Note that according to the above definition, an FLF takes value of an argument or a negation of an argument.
For FLFs, the following *basic laws* of Boolean algebra (Kleene 1967) are valid: commutative, associative, distributive for both conjunction and disjunction, absorption, double negation, idempotency of elements, Kleene and de-Morgan. However, the exception of the third law, $x \wedge \bar{x} \neq 0$ and $y \vee \bar{y} \neq 1$, is not valid. Though it follows from (1) that $x \wedge \bar{x} < M$, $y \vee \bar{y} > M$, where the

central point $M = 1/2 = 0.5$ of the section D is called a *median*. *Disjunctive and conjunctive normal forms* (DNFs and CNFs) are introduced same as in Boolean algebra (Kleene 1967) with the only exception that a conjunct (disjunct) may contain a variable and its negation. A *minimization technique* for an FLF given by its DNF is studied in (Kabecode 1981; Kandel 1986; Wielgus 2014). To compare two given FLFs, we can compute and compare their values on all the variants of ordering arguments and their negations; the corresponding table is called a *choice table*. In Table 1, a choice table of function $f_1 = x_1\bar{x}_2 \vee \bar{x}_1 x_2$ is shown. We denote areas covering the function domain D^n as $A_i, 1 \le i \le L$, where L denotes the table's length. Fig. 1 shows the areas inside the unit square. In the general case, areas are formed as a result of hyperplanes $x_i = \bar{x}_j$ $(1 \le j \le n, 1 \le j \le n)$ intersection inside the unit hypercube.

Table 1: A choice table of function f_1

Number of area	Specification of area	Value of function f_1
1	$x_1 \le x_2 \le \bar{x}_2 \le \bar{x}_1$	x_2
2	$x_1 \le \bar{x}_2 \le x_2 \le \bar{x}_1$	x_2
3	$\bar{x}_1 \le x_2 \le \bar{x}_2 \le x_1$	$\bar{x}_2$
4	$\bar{x}_1 \le \bar{x}_2 \le x_2 \le x_1$	$\bar{x}_2$
5	$x_2 \le x_1 \le \bar{x}_1 \le \bar{x}_2$	x_1
6	$x_2 \le \bar{x}_1 \le x_1 \le \bar{x}_2$	x_1
7	$\bar{x}_2 \le x_1 \le \bar{x}_1 \le x_2$	$\bar{x}_1$
8	$\bar{x}_2 \le \bar{x}_1 \le x_1 \le x_2$	$\bar{x}_1$

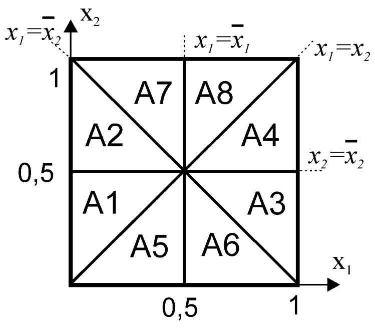

Figure 1: Areas of Table 1 inside the unit square (2-dimensional case)

In further notations, we omit symbol of variable "x" and consider vectors of indices. Denoting index of $\bar{x}_i$ as $-i$, we represent a sequence of indices, which specifies an area of the domain D^n, with a vector of 2n elements $\vec{\imath} = (i_1, i_2, \dots, i_{2n})$, $i_j \in \{-n, \dots, -1, 1, \dots, n\}$, $1 \le j \le 2n$. For brevity, we use negative indices i_j as well introducing the following notation:

$$x^i = \begin{cases} x_i, & i > 0 \\ \bar{x}_{|i|}, & i < 0. \end{cases} \qquad (2)$$

Thus, a vector $\vec{\imath}$ represents a domain area specified with $x^{i_1} \le x^{i_2} \le \dots \le x^{i_{2n}}$. A *choice table* T is a set of rows $T = \{t\}$, where the table row has the following form $t = (\vec{\imath}, a)$ which means that the function takes value x^a on the area specified by $\vec{\imath}$. When additional specifications are absent, we suppose that all the areas are listed in a choice table. Though studying a partially defined table could be useful, especially for the minimization purposes.

For example, the choice table, shown in Table 1, is specified as

$$T_1 = \{((1,2,-2,-1),2), ((1,-2,2,-1),2), ((-1,2,-2,1),-2), ((-1,-2,2,1),-2), ((2,1,-1,-2),1), ((2,-1,1,-2),1), ((-2,1,-1,2),-1), ((-2,-1,1,2),-1)\}.$$

Note that a tuple $\vec{\imath}$ is symmetric with respect to its middle because $x \le y$ implies $\bar{y} \le \bar{x}$; thus, only one half of vector $\vec{\imath}$, for instance the first, can be actually stored. It means that the number of the choice tables of n arguments L(n) is defined by the number of permutations of n numbers multiplied by the number of variants for assigning signs to each of permutations; thus,

$$L(n) = n! \cdot 2^n. \qquad (3)$$

Note that in examples we use functions of two arguments which CTs contain 8 rows; for 3 arguments the number of rows equals 48 and such examples are rather bulky; an example of synthesizing an FLF of 3 arguments is considered in Appendix. Asymptotically, the number of choice tables considerably exceeds the number of FLFs (Volgin and Levin 1990), which means that not any choice table defines an FLF. When the order of areas specified with vectors $\vec{\imath}$ is fixed, for instance same as in Table 1, we can represent a table as a vector of L(n) values $a_{\vec{\imath}}$. For example, the choice table $T_2 = (2,-1,2,-2,1,-1,-2,-1)$ does not define an FLF. However, each choice table can be partitioned in subdomains each of which defines an FLF (Volgin and Levin 1990). Note that not more than L(n) areas are required when specifying each function with a DNF consisting of a single argument (negation of an argument).

PARTITIONING A CHOICE TABLE WITH FUZZY LOGIC FUNCTIONS

We adjust the method of synthesis (Zaitsev et al 1998) to obtain a simple procedure for partitioning a given CT with a set of FLFs represented by DNF. The paper introduces a criterion when a given CT specifies an FLF based on a notion of *overlapping rows* of the table.

Here, we extended the technique (Zaitsev et al 1998) on partial tables and replace a formal application of the criterion by a simple iterative procedure which consists of the following steps:

1) Synthesize a DNF on a given table (part of table).
2) Build a CT on the obtained DNF and compare the function values.
3) If the function values coincide then the sought FLF is specified by the obtained DNF.
4) Otherwise proceed from the step 1) for the part of table which contains rows where the function values do not coincide (the "difference table").

In (Zaitsev et al 1998), it was proven that if a CT defines an FLF, then the FLF is represented with a DNF consisting of disjunctions of *constituents of maximum* on the rows of the CT, where a constituent of maximum for a table row is equal to conjunctions of variables starting from that which equals the function value to the last variable, inclusive. Using the CT notation from Section 2 for a table row $t = (\vec{\imath}, a)$, its constituents of maximum, denoted φ_t, is calculated as

$$\varphi_t = \bigwedge_{x^a \le x^i} x^i \,. \tag{4}$$

Then, the synthesized DNF has the following form

$$f = \bigvee_{t \in T} \varphi_t \,. \tag{5}$$

We leave the issues of formal minimization of FLFs (Kabecode 1981; Kandel 1986; Wielgus 2014) beyond the scope of the present paper only applying a simple DNF reduction with the tautology and absorption laws

$$x \vee x = x, x \vee (xy) = x.$$

Note that a constituent length does not exceed $2n$ and the maximal table length is $L(n)$ specified with (2). Thus, the algorithm complexity is $O(2n \cdot L)$. Though it is exponential in n, taking into consideration (2), its characterization as linear in the table length sounds more optimistic.
Let us consider examples of FLFs synthesis. First, we synthesize a function on the CT shown in Table 1. According to (5)

$$f_1 = \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4 \vee \varphi_5 \vee \varphi_6 \vee \varphi_7 \vee \varphi_8 \,,$$

where according to (4)

$$\varphi_1 = x_2\bar{x}_2\bar{x}_1,\ \varphi_2 = x_2\bar{x}_1,\ \varphi_3 = \bar{x}_2x_1,\ \varphi_4 = \bar{x}_2x_2x_1,$$
$$\varphi_5 = x_1\bar{x}_1\bar{x}_2,\ \varphi_6 = x_1\bar{x}_2,\ \varphi_7 = \bar{x}_1x_2,\ \varphi_8 = \bar{x}_1x_1x_2.$$

Since φ_7 equals φ_2 , φ_6 equals φ_3, φ_2 absorbs φ_1 and φ_8, φ_3 absorbs φ_4 and φ_5, we obtain

$$f_1 = \varphi_2 \vee \varphi_3 = x_2\bar{x}_1 \vee \bar{x}_2x_1 \,,$$

that coincides with the initial expression which Table 1 has been constructed on.
Second, we synthesize an FLF (FLFs) on Table 2. We use a hint that the table defines a set of FLF introducing the following notation where the upper index specifies the function number in the source CT partitioning. We compose,

$$f_2^1 = \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4 \vee \varphi_5 \vee \varphi_6 \vee \varphi_7 \vee \varphi_8 \,,$$

Where

$$\varphi_1 = x_2\bar{x}_2\bar{x}_1,\ \varphi_2 = \bar{x}_1,\ \varphi_3 = x_2\bar{x}_2x_1,$$
$$\varphi_4 = \bar{x}_2x_2x_1, \varphi_5 = x_1\bar{x}_1\bar{x}_2,\ \varphi_6 = \bar{x}_1x_1\bar{x}_2,$$
$$\varphi_7 = \bar{x}_2\, x_1\bar{x}_1x_2,\ \varphi_8 = \bar{x}_1x_1x_2.$$

Since φ_3 equals φ_4 , φ_5 equals φ_6, φ_2 absorbs φ_1, φ_5, φ_7, φ_8, we obtain

$$f_2^1 = \varphi_2 \vee \varphi_3 = \bar{x}_1 \vee x_2\bar{x}_2x_1 \,.$$

However, a CT of f_2^1 coincides with the source CT of function f_2 only on the areas A_2, A_3, A_4, A_6, A_8.
For the rest of the table, where the values of f_2^1 do not coincide with the corresponding values of f_2, we proceed with the same procedure composing

$$f_2^2 = \varphi_1 \vee \varphi_5 \vee \varphi_7.$$

Since φ_1 equals φ_7, we obtain

$$f_2^2 = \varphi_1 \vee \varphi_5 = x_2\bar{x}_2\bar{x}_1 \vee x_1\bar{x}_1\bar{x}_2.$$

Thus, we specify function f_2, given by CT shown in Table 2, as

$$f_2 = \begin{cases} f_2^1 = \bar{x}_1 \vee x_2\bar{x}_2x_1, & \vec{x} \in A_2, A_3, A_4, A_6, A_8, \\ f_2^2 = x_2\bar{x}_2\bar{x}_1 \vee x_1\bar{x}_1\bar{x}_2, & \vec{x} \in A_1, A_5, A_7. \end{cases}$$

A TOOLBOX FOR SYNTHESIS OF FUZZY LOGIC FUNCTIONS ON A CHOICE TABLE

Using the technique described in Section 3, a toolbox fzy_syn has been implemented in C language and placed for free download on GitHub. It uses the following abbreviations: "fzy" denotes fuzzy logic, "tab" denotes a choice table (either complete or partial), "dnf" denotes a DNF, "syn" denotes synthesis, "cmp" denotes comparison.
The toolbox contains the following command line tools:

- fzy_tab_syn partitions a given choice table with a series of synthesized fuzzy logic functions (DNFs);
- fzy_tab_dnf synthesizes a fuzzy logic function (DNF) on a given choice table;
- fzy_dnf_tab builds a choice table on a given fuzzy logic function (DNF);
- fzy_cmp_tab_dnf compares a choice table with a fuzzy logic function (DNF);
- fzy_gen_tab generates a random choice table for a given number of variables.

fzy_tab_syn implements a repeated combination of fzy_tab_dnf and fzy_cmp_tab_dnf. In the general case, a DNF synthesized with fzy_tab_dnf satisfies only a part of the CT. Using fzy_cmp_tab_dnf, the CT is partitioned into two tables: the first table contains rows where DNF coincides with the source table; the second table contains rows where DNF does not coincide with the source table. Then the process is repeated with the difference table until it will be empty.

Table 2: A choice table of function f_2

Number	Area	f_2	f_2^1	f_2^2
1	$x_1 \le x_2 \le \bar{x}_2 \le \bar{x}_1$	x_2		x_2
2	$x_1 \le \bar{x}_2 \le x_2 \le \bar{x}_1$	$\bar{x}_1$	$\bar{x}_1$	
3	$\bar{x}_1 \le x_2 \le \bar{x}_2 \le x_1$	x_2	x_2	
4	$\bar{x}_1 \le \bar{x}_2 \le x_2 \le x_1$	$\bar{x}_2$	$\bar{x}_2$	
5	$x_2 \le x_1 \le \bar{x}_1 \le \bar{x}_2$	x_1		x_1
6	$x_2 \le \bar{x}_1 \le x_1 \le \bar{x}_2$	$\bar{x}_1$	$\bar{x}_1$	
7	$\bar{x}_2 \le x_1 \le \bar{x}_1 \le x_2$	$\bar{x}_2$		$\bar{x}_2$
8	$\bar{x}_2 \le \bar{x}_1 \le x_1 \le x_2$	$\bar{x}_1$	$\bar{x}_1$	

On a given CT, fzy_tab_dnf builds a DNF. The CT can be either complete or partial. A DNF is reduced with the application of the tautology and absorption laws. It is possible that the obtained DNF be valid for a part of the table only. The corresponding test can be implemented with fzy_cmp_tab_dnf.
On a given DNF of an FLF, fzy_dnf_tab builds a complete CT. The CT contains all L(n) areas specified with (3), where n is the number of FLF arguments. When enumerating the areas, it proceeds first with ascending order of permutations of n arguments, and second with all the combinations of arguments with and without negation; for a definite permutation, the ascending order of enumerating the negations corresponds to treating an argument without negation as a zero and an argument with negation as a unit. Thus, we enumerate n! permutations and for each permutation 2^n combinations of negations.
On the areas of a given CT, fzy_cmp_tab_dnf compares the function values with the values computed according to a given DNF. The CT can be either complete or partial. Two tables are written: the first table for the same values of function and the second table for different values of function taken from the source table.
fzy_gen_tab builds a complete CT with random values of function equal to arguments and negations of arguments. The CT specifies a complete function and contains all L(n) areas specified with (3), where n is the number of FLF arguments.
The toolbox implements a command line interface with input and output data located in textual files having special format; who types of files are supported: a file of CT and a file of a DNF of an FLF.
Command lines to launch the tools, has the following format:

```
>fzy_tab_syn tab_file result_files_prefix
>fzy_tab_dnf tab_file dnf_file
>fzy_dnf_tab dnf_file tab_file
>fzy_cmp_tab_dnf tab_file dnf_file
       comm_tab_file diff_tab_file
>fzy_gen_tab n tab_file [ rand_seed ]
```

The command line parameters are specified as follows:

- n is the number of FLF arguments;
- tab_file is a file which contains a CT either complete or partial;
- dnf_file is a file which contains a DNF of an FLF;
- comm_tab_file contains rows of the source table where values of the functions coincide;
- diff_tab_file contains rows of the source table where values of the functions are different;
- result_files_prefix is a prefix to create file names of subdomains using the following suffices: "-tab-<i>" – a subdomain, "-dnf-<i>" – its DNF.
- rand_seed is an integer which is used as a seed to generate random values; on default, the system time is used as a seed.

Let us specify formats of files. A choice table file has the following format:

```
FZYTAB n
Aᵢ    vᵢ
...
FZYTABEND [(L rows)]
```

Where:
- "FZYTAB" is a label of the table file beginning;
- n is the number of arguments;
- L is the number of table rows;
- "FZYTABEND" is a label of the table file end;
- A_i is the current area ($1 \le i \le L$);
- v_i is the function value on the current area.

The current area A_i is specified as follows:

A_i : : j_1 j_2 ... j_{2n}

where the values range is

$$-n \le j_k < 0, 0 < j_k \le 2n, 1 \le k \le 2n.$$

The term j_k defines k-th item in the list A_i which equals to x_{j_k} if $j_k > 0$ and equals to the negation $\bar{x}_{|j_k|}$ if $j_k < 0$.
An example of the choice table file (f1_tab) for the function f_1 (according to Table 1) follows:

```
FZYTAB 2
 1  2 -2 -1      2
 1 -2  2 -1      2
-1  2 -2  1     -2
-1 -2  2  1     -2
 2  1 -1 -2      1
 2 -1  1 -2      1
-2  1 -1  2     -1
-2 -1  1  2     -1
FZYTABEND (8 rows)
```

A DNF file has the following format:

```
FZYDNF n
   Cᵢ
...
FZYDNFEND [(L conjuncts)]
```

Where:

- "FZYDNF" is a label of the DNF file beginning;
- n is the number of arguments;
- L is the number of conjuncts;
- "FZYDNFEND" is a label of the DNF file end;
- C_i is the current conjunct ($1 \le i \le L$).

The current conjunct C_i is specified as follows:

$C_i :: \; j_1 \; j_2 \quad \dots \quad j_{m_i}$

where the values range is

$$-n \le j_k < 0, 0 < j_k \le 2n, 1 \le k \le m_i;$$

m_i is the number of items (variables and negations of variables) in the current conjunct C_i; the term j_k defines k-th item in the list C_i which equals to x_{j_k} if $j_k > 0$ and equals to the negation $\bar{x}_{|j_k|}$ if $j_k < 0$.
An example of the DNF file (f1_dnf) for the function f_1 follows:

```
FZYDNF 2
1 -2
-1 2
FZYDNFEND (2 conjuncts)
```

RUNNING TOOLS AND ANALYSING RESULTS

Let us consider examples of command lines. Suppose the source data file f1_dnf specifying DNF of f_1 has been created by hand in a text editor. The command line

```
>fzy_dnf_tab f1_dnf f1_tab
```

creates a CT of f_1 on its DNF file f1_dnf and saves the resulting CT in file f1_tab. We can check visually that it coincides with the table file shown above.
Suppose the source data file f2_tab specifying CT of f_2 has been created by hand in a text editor. The command line

```
>fzy_tab_dnf f2_tab f2_1_dnf
```

synthesizes a DNF file f2_1_dnf on the CT file f2_tab. The obtained DNF file f2_1_dnf follows:

```
FZYDNF 2
-1
2 -2 1
FZYDNFEND (2 conjuncts)
```

The command line

```
>fzy_cmp_tab_dnf f2_tab f2_1_dnf
              f2_tab_comm f2_tab_diff
```

compares a function given by CT file f2_tab with an FLF given by DNF file f2_1_dnf, writes the coinciding rows to the file f2_tab_comm, and writes the different rows to the file f2_tab_diff with the function values according to the source CT file (f2_tab). The obtained difference file f2_tab_diff follows:

```
FZYTAB 2
 1  2 -2 -1     2
 2  1 -1 -2     1
-2  1 -1  2    -2
FZYTABEND (3 rows)
```

It is not empty; consequently f_2 is not an FLF. The obtained coinciding part of the CT represented with the file f2_tab_comm follows:

```
FZYTAB 2
 1 -2  2 -1    -1
-1  2 -2  1     2
-1 -2  2  1    -2
 2 -1  1 -2    -1
-2 -1  1  2    -1
FZYTABEND (5 rows)
```

In fact it contains the rows where the function f_2^1 specified with f2_1_dnf represents the source function f_2. Then, we can synthesize a DNF for the difference file with

```
>fzy_tab_dnf f2_tab_diff f2_2_dnf
```

to obtain the second DNF file f2_2_dnf to cover the source CT of f_2:

```
FZYDNF 2
2 -2 -1
1 -1 -2
FZYDNFEND (2 conjuncts)
```

However, it is more convenient to use fzy_tab_syn which partitions the source CT automatically. The command line

```
>fzy_tab_syn f2_tab f2_syn
```

partitions the source CT f2_tab into subdomains and synthesizes an FLF represented with a DNF for each subdomain. The following files are created: f2_syn-tab-0, f2_syn-dnf-0 and f2_syn-tab-1, f2_syn-dnf-1 specifying the obtained partitioning. File f2_syn-tab-0 coincides with the above f2_tab_comm and f2_syn-dnf-0 coincides with the above f2_1_dnf. File f2_syn-tab-1 coincides with the above f2_tab_diff and f2_syn-dnf-1 coincides with the above f2_2_dnf.
To try the toolbox on big random data we use fzy_gen_tab to create random CTs of specified number of arguments. The command line

```
>fzy_gen_tab 6 F3_rand_tab
```

creates a CT of a function of 6 arguments with random values.

CONCLUSIONS

A toolbox for synthesis of fuzzy logic functions on a choice table has been implemented in C language; it is available for free download on GitHub. The method

described in (Zaitsev et al 1998) has been adjusted for fast partitioning the source choice table with a set of fuzzy logic functions. No formal minimization of fuzzy logic functions has been implemented though the DNF transformation using the tautology and absorption laws allows its considerable reduction. Formal aspects of optimality (minimalism) are a direction for future research.
The toolbox implements a command line style of programming using data located in textual files of simple intuitive formats. Though it lacks graphical user interface, the toolbox can process rather big data fast. One more benefit is possibility of its easy integration into fuzzy logic frameworks. Thus the toolbox can be used as a synthesis engine to develop graphical systems for fuzzy (control) systems design.

REFERENCES

Kabekode, V.S.B. 1981. “Minimization of disjunctive normal forms of fuzzy logic functions”. *Journal of the Franklin Institute*, 311(3), 171-185.
Kandel, A. 1986. *Fuzzy Mathematical Techniques with Applications*. Addison-Wesley.
Kaufmann, A. 1977. *Introduction a la theorie des sous-ensembles flous*. Masson.
Kleene, S.C. 1967. *Mathematical Logic*. Wiley.
Novak, V,; I. Perfilieva; and A. Dvorak. 2016. *Insight into Fuzzy Modeling*. Wiley.
Shmeleva, T.R.; Zaitsev, D.A.; and Zaitsev, I.D. 2009. “Verification of square communication grid protocols via infinite Petri nets”. In *Proceedings of 10th Middle Eastern Simulation Multiconference*, 53-59.
Volgin, L.I. and V.I. Levin. 1990. *Continuous Logic. Theory and Applications*. Tallinn, AS of Estonia.
Wielgus, A. 2014. "Heuristic algorithm of two-level minimization of fuzzy logic functions", In *Proceedings of 17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, 302-305.
Wielgus, A. 2004. "Algorithms of multilevel synthesis of fuzzy functions", In *Proceedings of IEEE International Conference on Fuzzy Systems*, 929-934.
Zadeh, L.A. 1965. "Fuzzy sets", *Information and Control*, 8(3), 338-353.
Zaitsev, D.A.; T.R. Shmeleva; W. Retschitzegger; and B. Proell. 2016. “Security of grid structures under disguised traffic attacks”. *Cluster Computing*, 19(3), 1183-1200.
Zaitsev, D.A. and J. Jurjens. 2016. “Programming in the Sleptsov net language for systems contro”. *Advances in Mechanical Engineering*, 8(4), 1–11.
Zaitsev, D.A. and T.R. Shmeleva. 2006. “Switched Ethernet Response Time Evaluation via Colored Petri Net Model”. In *Proceedings of International Middle Eastern Multiconference on Simulation and Modelling*, Alexandria, Egypt. 68-77.
Zaitsev, D.A.; V.G. Sarbei; anf A.I. Sleptsov. 1998. “Synthesis of continuous-valued logic functions defined in tabular form”. *Cybernetics and Systems Analysis*, 34(2), 1998, 190–195.
Zaitsev, D.A. and A.I. Sleptsov. 1997. “State equations and equivalent transformations for timed Petri nets”. *Cybernetics and Systems Analysis*, 33(5), 659-672

WEB REFERENCES

github.com/dazeorgacm/fzy_syn

APPENDIX: AN EXAMPLE OF SYNTHESIS OF AN FLF OF 3 ARGUMENTS

A given CT file (f3_tab):

```
 FZYTAB 3
 1  2  3 -3 -2 -1    -2
 1  2 -3  3 -2 -1    -2
 1 -2  3 -3  2 -1    -3
 1 -2 -3  3  2 -1     3
-1  2  3 -3 -2  1    -3
-1  2 -3  3 -2  1     3
-1 -2  3 -3  2  1     2
-1 -2 -3  3  2  1     2
 1  3  2 -2 -3 -1    -3
 1  3 -2  2 -3 -1    -3
 1 -3  2 -2  3 -1    -2
 1 -3 -2  2  3 -1     2
-1  3  2 -2 -3  1    -2
-1  3 -2  2 -3  1     2
-1 -3  2 -2  3  1     3
-1 -3 -2  2  3  1     3
 2  1  3 -3 -1 -2    -1
 2  1 -3  3 -1 -2    -1
 2 -1  3 -3  1 -2    -3
 2 -1 -3  3  1 -2     3
-2  1  3 -3 -1  2    -3
-2  1 -3  3 -1  2     3
-2 -1  3 -3  1  2     1
-2 -1 -3  3  1  2     1
 2  3  1 -1 -3 -2    -3
 2  3 -1  1 -3 -2    -3
 2 -3  1 -1  3 -2    -1
 2 -3 -1  1  3 -2     1
-2  3  1 -1 -3  2    -1
-2  3 -1  1 -3  2     1
-2 -3  1 -1  3  2     3
-2 -3 -1  1  3  2     3
 3  1  2 -2 -1 -3    -1
 3  1 -2  2 -1 -3    -1
 3 -1  2 -2  1 -3    -2
 3 -1 -2  2  1 -3     2
-3  1  2 -2 -1  3    -2
-3  1 -2  2 -1  3     2
-3 -1  2 -2  1  3     1
-3 -1 -2  2  1  3     1
 3  2  1 -1 -2 -3    -2
 3  2 -1  1 -2 -3    -2
 3 -2  1 -1  2 -3    -1
 3 -2 -1  1  2 -3     1
-3  2  1 -1 -2  3    -1
-3  2 -1  1 -2  3     1
-3 -2  1 -1  2  3     2
-3 -2 -1  1  2  3     2
FZYTABEND (48 rows)
```

A command line:

```
>fzy_tab_dnf f3_tab f3_dnf
```

The obtained DNF file (f3_dnf):

```
FZYDNF 3
-2 -1
2 1
-3 -1
3 1
-3 -2
3 2
FZYDNFEND (6 conjuncts)
```

The obtained FLF:

$$f_3 = \bar{x}_2\bar{x}_1 \vee x_2x_1 \vee \bar{x}_3\bar{x}_1 \vee x_3x_1 \vee \bar{x}_3\bar{x}_2 \vee x_3x_2 \ .$$