Information and logical transformations in Schaeffer and Pierce Bases in Maple

Aleksandr Olenev[1*], *Ekaterina* Potekhina[1], *Alexey* Khabarov[2], and *Larisa* Zvereva[1]

[1]Stavropol State Pedagogical Institute, 417A, Lenina str., Stavropol, 355029, Russia
[2]Stavropol State Agrarian University, 12, Zootechnichesky, Stavropol, 355017, Russia

Abstract. One of the tasks of synthesis and analysis of the functioning of combinational devices is the representation of a Boolean function in various bases. The absolute completeness of the use and implementation of various transformations of Boolean functions is impossible without the implementation of transformations into universal bases of Schaeffer (AND-NOT) or Webb (OR-NOT). The Logic library of the Maple computer algebra system allows you to use only the operations &nand (Schaeffer basis) and &nor (Webb basis) to build a model of a digital device. Therefore, there is a need to expand the functionality of the Logic library of the Maple computer algebra system by representing Boolean functions in Schaeffer and/or Pierce (Webb) bases. In this regard, the authors of the article have developed procedures that allow the representation of disjunctive or conjunctive normal forms in the Schaeffer basis or in the Webb basis. The development was carried out taking into account the existing data processing functions and procedures. The article describes the technology of extending the Logic library of the Maple computer algebra system by developed procedures that implement the transformation of a given or received Boolean function in Schaeffer or Webb bases.

1 Introduction

Computer algebra systems (CSA) with support for performing Boolean algebra operations can be divided into two groups. The first group includes computer algebra systems that were created mainly to solve problems of propositional logic, for example, Matcad, Derive, etc. [1-4]. In such systems, the emphasis is on the representation or transformation of logical expressions in a general way. The second group includes universal computer algebra systems, such as Maple, Mathematica, etc., which allow solving problems not only of propositional logic, but also Boolean algebra, as well as making transitions from one representation to another [5-7]. It is to this group that the Maple computer algebra system belongs, the peculiarities of expanding transformations of Boolean functions to which this work is devoted.

Maple has significant differences from other computer algebra systems that allow the conversion of Boolean functions, in particular [8-11]:

– the use of mathematical notation of logical expressions;

* Corresponding author: olenevalexandr@gmail.com

- convenience of defining arbitrary forms of representation;
- the possibility of using And-NOT and OR-NOT operations.

At the same time, Maple SKA has significant drawbacks. In particular, in all versions (at the moment there are more than 40 of them) there is no possibility of converting a Boolean function presented in disjunctive or conjunctive form into Schaeffer or Webb bases. Accordingly, the functions of Schaeffer's Stroke and Pierce's arrow are used for this. In addition, the program structure of the system itself makes it impossible to hope for the implementation of these transformations in the near future.

In addition, there is practically no documentation on the use of the operations &nand (Schaeffer basis) and &nor (Webb basis) of the Logic library of the Maple computer algebra system for constructing models of Boolean functions. On the website of the system itself (https://www.maplesoft.com/support/help/Maple/view.aspx?path=Logic) a fairly large set of examples is presented, but there are no specific examples of using the above functions, especially transformations (https://www.maplesoft.com/support/ help/ Maple/view.aspx?path=Logic%2foperators). There is no significant information on this issue in specialized research articles in which you can find a possible application of the Maple system [12-14].

In this paper, we propose the implementation of procedures for converting disjunctive and conjunctive normal forms into logical expressions represented using the Schaeffer Stroke and Pierce Arrow operations, which allows us to expand the functionality of the Logic library of the Maple computer algebra system.

2 Problem statement and known results

Analyzing the capabilities of mathematical systems [15-16], one can come to the conclusion: there are not enough built-in functions in mathematical packages to work with various representations of Boolean function models, which means that it is necessary to create auxiliary functions for each task or use specially created programs based on the means of a mathematical package.

In addition to computer algebra systems, there are and are being developed programs specifically designed to work with Boolean functions, for example, Boolean Functions, which was created by N.A. Kolomeets and A.V. Pavlov [17]. After analyzing the possibilities of Boolean Functions, we can conclude that this system does not allow us to represent Boolean functions in universal bases of Schaeffer's Stroke or Pierce's Arrow.

Another example of a program specially designed to work with Boolean algebra is a software package created by I.S. Lartsov [18]. The program is an MDI application and allows you to solve such problems of Boolean algebra as: building logic circuits and truth tables of arbitrary Boolean functions, converting Boolean functions into normal forms, representing a Boolean function as a Zhegalkin polynomial, performing other transformations, but not related to representations in Schaeffer and Webb bases.

The analysis of the functionality of the considered complex allows us to conclude that it is focused on solving educational problems in discrete mathematics and mathematical logic, but it is not possible to consider the solution of a number of problems related to the transformation and representation of Boolean functions using the Schaeffer Stroke and Pierce Arrow operations in programs.

In addition to the programs considered, there are other programs [19-20] that also work with Boolean functions, but their peculiarity is that they are not complexes for working with such functions themselves and, as a rule, implement only individual algorithms necessary for the creators to perform some task.

Thus, the analysis of currently available software tools for working with Boolean functions suggests that there are tools for working with Boolean functions designed to solve

problems, including educational ones. However, these programs are not focused on solving problems related to the representation of Boolean functions using the universal elements of Schaeffer's Stroke and Pierce's Arrow. Therefore, creating a software tool that is convenient for studying and working with the representation of Boolean functions in Schaeffer and Web bases is an urgent and in-demand task.

3 Method

The results of transformations of Boolean functions, the use of the laws of logic algebra, the implementation of representations in various normal forms, the minimization of Boolean functions, all this has found wide application in the design of electronic circuits. In particular, the task of representing arbitrary logical functions using formulas containing logical operations from a given set is important. Such sets of operations are called functionally complete. The most studied functional sets of logical operations are the standard (basic) basis ("¬", "∧", "∨"), in which logical functions are represented as conjunctive and disjunctive normal forms, and the Zhegalkin basis ("⊕", " ∧", "1"), in which logical functions are the only thus are represented by Zhegalkin polynomials [21-23].

Functionally complete sets of logical operations containing only one operation are the Schaeffer basis, which includes the Schaeffer stroke function "| ", and the Webb (Pierce) basis, which includes the Pierce arrow function "↓" [24-26].

To solve the problem of constructing a program code for the transition from one form of representation to another (from the standard to the Webb or Schaeffer basis), it is necessary to consider the main features of these operations [27-30].

The approach to the transition to the Schaeffer and Pierce basis is to use de Morgan's laws and double negation.

Consider a logical function given by a conjunctive monomial.

$$\overline{\boldsymbol{x_1 \wedge x_2 \ldots \wedge x_n}} = \boldsymbol{\overline{x_1} \vee \ldots \vee \overline{x_n}} \quad (1)$$

and we give examples of its representation in the Schaeffer basis for the cases n =1, n = 2, n= 3.

1) Let n = 1. Then, using de Morgan's law, we get:
$$\overline{x_1} = x_1 | x_1$$

2) Let n=2. Then, using de Morgan's law, we get:
$$\overline{x_1 \wedge x_2} = \overline{x_1} \vee \overline{x_2} = x_1 | x_2$$

3) Let n=3. Then, using de Morgan's law, we get:
$$\overline{x_1 \wedge x_2 \wedge x_3} = \overline{x_1} \vee \overline{x_2} \vee \overline{x_3} = x_1 | x_2 | x_3 \quad (2)$$

Similarly, we will analyze the disjunctive monomial:
$$\overline{x_1 \vee x_2 \vee \ldots \vee x_n} = \overline{x_1} \wedge \ldots \wedge \overline{x_n} \quad (3)$$

We give similar expressions for the representation of a function in the Webb basis (Pierce Arrow) for the cases n =1, n =2, n=3:

1) Let n =1. Then, using de Morgan's law, we get:
$$\overline{x_1} = x_1 \downarrow x_1$$

2) Let n=2. Then, using de Morgan's law, we get:
$$\overline{x_1 \vee x_2} = \overline{x_1} \wedge \overline{x_2} = x_1 \downarrow x_2$$

3) Let n=3. Then, using de Morgan's law, we get:
$$\overline{x_1 \vee x_2 \vee x_3} = \overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3} = x_1 \downarrow x_2 \downarrow x_3 \quad (4)$$

3.1 The algorithm of transition from MDNF and MKNF to the schaeffer and Webb basis

We present an algorithm for the transition from the minimal disjunctive normal form (DNF) to the Schaeffer basis. Consider a function for n variables [31-33]:

$$f(x_1, \dots, x_n) = \bigvee_{i=1}^{k} \lambda_l,$$

where $\lambda_i = \wedge_{m=1}^{n} \alpha_i \, q_i, \alpha_i \begin{cases} 0, if\ q\_m\ is\ not\ contained\ in\ \lambda_i, \\ 1, if\ q\_m\ is\ contained\ in\ \lambda\_i\ в\ \lambda_i, \end{cases}$ q_m – variable or its negation

Using the laws of double negation and de Morgan, the function can be represented as follows:

$$f(x_1, \dots, x_n) = \lambda_1 \vee \dots \vee \lambda_k = \overline{\overline{\lambda_1 \vee \dots \vee \lambda_k}} = \overline{\overline{\lambda_1} \wedge \dots \wedge \overline{\lambda_k}}$$

Then, applying the previously given relations for the Schaeffer basis, the function can be written as:

$$f(x_1, \dots, x_n) = \overline{\overline{\lambda_1} \wedge \dots \wedge \overline{\lambda_k}} = \overline{\lambda_1} \,|\, \overline{\lambda_2} \,|\, \dots \,|\, \overline{\lambda_k} \quad (5)$$

The resulting expression is the basis of the algorithm for the transition from the disjunctive form to the Schaeffer basis, presented below.

Step one. We carry out the transformation (verification of correctness) of the representation of a logical function in a disjunctive form with the preservation of inversion. If a literal with an inversion is included in the polynomial, then the transformation is performed as $\overline{x_1} = x_1 | \, x_1$. We carry out the transformation (formation) list of the conjunctive polynomial.

Step two. We obtain a conjunctive polynomial in a given form. In this case, in the Schaeffer basis.

Step three. We obtain conjunctive polynomials in the form of a list.

Step four. We accumulate conjunctive polynomials. We check for special cases. The conjunctive polynomial is represented by only one literal or its inversion, or the disjunctive form includes only one conjunctive polynomial.

Step five. We get the final result.

By analogy, we obtain an algorithm for the transition from the minimal conjunctive normal form (SKNF) to the basis of the Pierce Arrow.

Based on the function for n variables of the following form:

$$f(x_1, \dots, x_n) = \bigwedge_{i=1}^{k} \lambda_i$$

where $\lambda_i = \vee_{m=1}^{n} \alpha_i \, q_i, \alpha_i \begin{cases} 0, if\ q_m is\ not\ contained\ in \lambda_i, \\ 1, \text{e}if\ q\_m\ is\ contained\ in\ \lambda_i, \end{cases}$ q_m – variable or its negation

Using the laws of double negation and de Morgan, the function can be represented as follows:

$$f(x_1, \dots, x_n) = \lambda_1 \wedge \dots \wedge \lambda_k = \overline{\overline{\lambda_1 \wedge \dots \wedge \lambda_k}} = \overline{\overline{\lambda_1} \vee \dots \vee \overline{\lambda_k}}$$

Then, applying the previously given relations for the Webb basis (Pierce Arrow), the function can be represented as follows:

$$f(x_1, \dots, x_n) = \overline{\overline{\lambda_1} \vee \dots \vee \overline{\lambda_k}} = \overline{\lambda_1} \downarrow \dots \downarrow \overline{\lambda_k} \quad (6)$$

The resulting expression is the basis of the algorithm for the transition from the conjunctive normal form in the Webb basis (Pierce Arrow), presented below.

Step one. We carry out the transformation (verification of correctness) of the representation of a logical function in a conjunctive form with the preservation of inversion. If a literal with an inversion is included in the polynomial, then the transformation is performed as $\overline{x_1} = x_1 \downarrow x_1$.

Step two. We obtain a disjunctive polynomial in a given form. In this case, in the Webb basis.

Step three. We obtain disjunctive polynomials in the form of a list.

Step four. We accumulate disjunctive polynomials. We check for special cases. A disjunctive polynomial is represented by only one literal or its inversion, or the disjunctive form includes only one disjunctive polynomial.

Step five. We get the final result.

4 Implementation of the transformation in the maple system

Transformations of a Boolean function into the selected basis should begin by entering a prepared or received logical expression.

To fulfill this requirement, you can use the Logic – **Normalize** library function. To get the selected conjunctions or disjunctions and represent them as a list, the following construction is used **[op(Normalize (Y))]**. The use of such a construction makes it possible to select the necessary conjunction or disjunction of the normal form, transfer it to a separate processing (implementation of transformations taking into account special cases) and, upon receipt of this transformed disjunction or conjunct, accumulate them in a cycle. Iterations of the loop are limited by the number of elements in disjunctive or conjunctive normal form.

Listing 1: Representation of the conjunct in the form of a list, taking into account the inversion operations implemented in the Schaeffer basis (formula (1) – (2)).

```
> Disjunction:= proc(a)
local K, i, C;
K:= [op(Normalize(a))];
for i from 1 to nops(K) do
if nops(K)>1 and  op(0,K[i])=`&not` then K[i]:= op(1,K[i]) &nand op(1,K[i]);
elif nops(K)>1 and  op(0,K[i])<>`&not` then K[i]:= op(1,K[i]);
end if;
end do:
C:=&nand(op(K));
if nops(a)=1 and op(0,a)=`&not` then C:=op(1,a);
elif nops(a)=1 and op(0,a)<>`&not` then C:= op(1,a)&nand op(1,a);
end if;
return C;
end proc:
```

Listing 2: Representation of a Boolean function in the Schaeffer Basis (disjunctive normal form), implementation of formula (5).

```
>  DNF_SHSH:=proc(b)
local W, i, R;
W:= [op(Normalize(b))];
for i from 1 to nops(W) do
W[i]:=Disjunction(W[i]);
end do;
R:=&nand(op(W));
if nops(b)>1 and op(0,b)=`&and` then R:=Disjunction(b)&nand Disjunction(b);
elif nops(b)=1 then R:=Disjunction(b)&nand Disjunction(b);
end if;
return R;
end proc:
```

Similarly for the Webb basis.

Listing 3: Representation of the disjunction in the form of a list, taking into account the inversion operations implemented in the Webb basis (formula (3)-(4)).

```
> Conjunct:= proc(a)
local K, i, C;
K:= [op(Normalize(a, form=CNF))];
for i from 1 to nops(K) do
if nops(K)>1 and  op(0,K[i])=`&not` then K[i]:= op(1,K[i]) &nor op(1,K[i]);
elif nops(K)>1 and  op(0,K[i])<>`&not` then K[i]:= op(1,K[i]);
end if;
end do:
C:=&nor(op(K));
if nops(a)=1 and op(0,a)=`&not` then C:=op(1,a);
elif nops(a)=1 and op(0,a)<>`&not` then C:= op(1,a)&nor op(1,a);
end if;
return C;
end proc:
```

Listing 4: Representation of a Boolean function in the Webb Basis (conjunctive normal form), implementation of formula (6).

```
> KNF_SP:=proc(b)
local W, i, R;
W:= [op(Normalize(b, form=CNF))];
for i from 1 to nops(W) do
W[i]:=Conjunct(W[i]);
end do;
R:=&nor(op(W));
if nops(b)>1 and op(0,b)=`&or` then R:=Conjunct(b)&nor Conjunct(b);
elif nops(b)=1 then R:=Conjunct(b)&nor Conjunct(b);
end if;
return R;
end proc:
```

5 Example of application of the developed procedures

The algorithm and program code are implemented in the Maple 2014 32 bit environment. Calculations were performed on an Intel(R) Core(TM) i5-3300 S CPU 2.70 GHz 32bit, 4.00 GB RAM running Windows 7 Corporate SPI. The counting time was less than 0.2 seconds.

Example 1. It is necessary to represent the following logical function in the Schaeffer basis:

$$\overline{(c \wedge g \wedge h)} \vee (d \wedge j)$$

Decision. Using the developed functions, we will carry out the transformation into a universal Schaeffer basis:

```
> DNF_SHSH((&not c &and g &and h) &or (d &and j));
```

We received the following response:

$$(d \;\&nand\; j) \;\&nand\; ((g \;\&nand\; h) \;\&nand\; (c \;\&nand\; c))$$

Using the **Equivalent** function of the Logic library of the Maple computer algebra system, we will check the correctness of the transformation of a given function:

```
> Equivalent(%, &not c &and g &and h &or (d &and j));
```

We received the following response:

true

The result obtained indicates the correctness of the developed procedures.

Example 2. It is necessary to represent the following logical function in the Webb basis:

$$(x \vee \overline{y} \vee z) \wedge a$$

Decision. Let's imagine the above expression using the Maple language:

Using the developed functions, we will transform into a universal Webb basis:

> KNF_SP((&not y &or x &or z)&and (a));

We received the following response:

$$(a \,\&nor\, a)\&nor\, ((x \,\&nor\, z)\&nor\, (y \,\&nor\, y))$$

Using the **Equivalent** function of the Logic library of the Maple computer algebra system, we will check the correctness of the transformation of a given function:

> Equivalent(%,(x &or &not y &or z)&and a);

As a result of the inspection , we received the following result:

true

The result obtained indicates the correctness of the developed procedures.

6 Conclusion

The created procedures allow you to expand the capabilities of the Logic package of the Maple computer algebra system. Thus, it was possible to carry out an automated transformation of a Boolean function in the basis of Schaeffer (Schaeffer's Stroke) and Webb (Pierce's Arrow). It should also be noted that the construction of a Boolean function model using the universal Schaeffer and Webb bases was justified, since the implemented algorithms allow performing these transformations and comparing the obtained forms with those obtained earlier. In addition, it makes it possible to significantly expand the possibilities of using this library.

That is, the built-in language of the Maple computer algebra system and its application makes it possible to adequately express an algebraic construction in universal bases and confirms the need to expand the Logic library and conduct formal proofs of such a representation.

We thank the members of the editorial board for their patience, timely, complete technical editing and editing of the text of the article.
We will be glad to have any questions, comments and suggestions for improving the article.

References

1. D.V. Kiryanov, *Mathcad 15/Mathcad Prime 1.0* (BHV-Petersburg, St. Petersburg, 2012)
2. A. Polovko, *Derive for a student* (BHI-Petersburg, St. Petersburg, 2005)
3. A.Yu. Golubkov, A.I. Zobnin, O.V. Sokolova, *Computer algebra in the Sage system* (MSTU named after N.E. Bauman, Moscow, 2013)
4. V.A. Dalinger, S.D. Simonzhenkov, *Computer Science and Mathematics. Solving equations and optimization in Matcad and Maple* (Urayt, Moscow, 2023)
5. M.B. Monagan, K.O. Geddes, K.M. Heal, G. Labahn, S.M. Vorkoetter, J. McCarron, and P. DeMarco, *Maple Introductory, Programming Guide* (Maplesoft, a division of Waterloo Maple Inc., 2009)
6. N.A. Vavilov, V.G. Khalin, A.V. Yurkov, *Mathematica for non-mathematicians* (Electronic edition, MTsNMO, Moscow, 2021)
7. R.J. Lopez, *Maple via Calculus: A Tutorial Approach* (Springer Science & Business Media, 2012)

8. H. Hamid, N. Angkotasan, A. Jalal, D. Muhtadi, Sukirwan, J. Phys.: Conf. Ser. **1613,** 0120252020 (2009)
9. Y.L. Ningsih, R. Paradesa, J. Phys.: Conf. Ser. **948,** 012034 (2018)
10. J.L. Pardo Gómez, *Introduction to Cryptography with Maple* (Springer, New York, 2013)
11. R.E. Klima, N.P. Sigmon, *Cryptology: Classical and Modern with Maplets* (Chapman & Hall/CRC, New York, 2012)
12. M. Durcheva, E. Nikolova*,* AIP Conference Proceedings, **2048(1)**, 060010 (2018)
13. M. Durcheva, E. Varbanova, Math.Comput.Sci. **11,** 305-314 (2017)
14. A.A. Olenev, L.G. Zvereva, T.H. Saieg, Journal of Higher Education Theory and Practice **22(8)**, 51-57 (2022)
15. N.A. Kolomeets, A.V. Pavlov, Computational methods in discrete mathematics. **Appendix 4**, 67-68 (2011)
16. I.S. Lartsov*, Software package for working with Boolean algebra* [Electronic resource]. Laboratory work. Faculty of Computational Mathematics and Cybernetics. Nizhny Novgorod State University named after. N.I. Lobachevsky. Nizhny Novgorod (2011). irror.transact.net.au/sourceforge/t/project/ta/tablecloth/.../1.../otchet.pdf.
17. A.A. Olenev, I.A. Kalmykov, K.A. Kirichek, I.A. Provornov, A program for visualizing the construction of perfect normal forms of logic algebra formulas. Certificate of registration of a computer program, 2023614273 (2023)
18. K.A. Kirichek, E.V. Potekhina, Program for visualization of basic logical operations. Certificate of registration of the computer program, 2021613965 (2021)
19. A.A. Olenev, I.A. Kalmykov, K.A. Kirichek, Software & Systems **36(3**), 349-360 (2023)
20. K.H. Rosen, *Discrete mathematics and its applications* (McGraw-Hill, New York, 2012)
21. A. Shingareva, C. Lizarraga-Celaya*, Maple and Mathematica. A Problem Solving Approach for Mathematics* (Springer-Verlag, New York, 2009)
22. V.A. Gorbatov*, Fundamental foundations of discrete mathematics* (Nauka, Moscow, 2000)
23. O.P. Kuznetsov, *Discrete mathematics for the engineer* (Lan, St. Petersburg, 2009)
24. O.A. Logachev, A.A. Salnikov, V.V. Yashchenko*, Boolean functions in coding theory and cryptology* (MTsNMO, Moscow, 2004)
25. V.P. Suprun, Cybernetics **1**, 116-117 (1987)
26. G. Borowik, G. Labiak, A. Bukowiec, *FSM-Based Logic Controller Synthesis in Programmable Devices with Embedded Memory Blocks* (Springer, London, 2015)
27. V. Riznyk, M. Solomko, P. Tadeyev, V. Nazaruk, L. Zubyk, V. Voloshyn, Eastern-European Journal of Enterprise Technologies **3(4(105))**, 43-60 (2020)
28. S. Baranov, A. Karatkevich, Int. Journal of electronics and telecommunications **64(3)**, 373-378 (2018)
29. M. Maxfield, Implementing Logic Functions Using Only NAND or NOR Gates (2018). https://www.eeweb.com/implementing-logic-functions-using-only-nand-or-nor-gates/
30. E.H. Shaik, N. Rangaswamy, Journal of Optics **47(1)**, 1-14 (2017)
31. V. Riznyk, M. Solomko*,* Technology Audit and Production Reserves **5(2(43))**, 42-55 (2018)
32. V. Riznyk, M. Solomko*,* Technology Audit and Production Reserves **6(2(38))**, 60-76 (2017)

33. A.A. Olenev, E.M. Petlina, A.V. Shuvaev, N.V. Grivennaya, A.N. Khabarov, *Maple Information Tools in the Study of Mathematical Logic Questions*. In P. Stanimorovic, A. A. Stupina, E. Semenkin, & I.V. Kovalev (Eds.), Hybrid Methods of Modeling and Optimization in Complex Systems, vol. 1. European Proceedings of Computers and Technology. European Publisher. 22-24 November 2022, Krasnoyarsk, Russia (2023)