Methods of decomposition of Boolean functions

N I Gdanskiy, A A Denisov and M L Rysin

Federal State Budget Educational Institution of Higher Education «K.G. Razumovsky Moscow State University of technologies and management (the First Cossack University), 73, Zemlyanoy Val, Moscow, 109004, Russia

E-mail: aadenisov88@gmail.com

Abstract. The article considers on of the main types of Boolean functions conversion – their decomposition. The decomposition is being used in many applications for studying and designing complex objects and systems with logical methods. Firstly, the article observes decomposition of Boolean functions represented in the truth table. Among traditional decomposition approaches, the article describes formulas presented by Boole and Shannon and special methods of orthonormal expansions. Then, the paper considers contemporary decomposition techniques and their classification. Next, the article discusses representation of Boolean functions in normal forms and essential features of that representation, important for decomposition. After that, the article reviews decomposition techniques for boolean functions represented in conjunctive normal forms (CNF). The study mentions CNF partitioning as hypergraph partitioning. Then the paper discusses applications generating large CNFs. Minimization of large CNFs with traditional methods is impossible. Their minimization is possible only using decomposition techniques. The separate class of tasks is minimization of Horn formulas. Finally, at the conclusion, the study discusses decomposition for parallelization of SAT problem.

1. Introduction

A common way of studying large and complex systems and objects is their decomposition (in particular, software decomposition) [1, 2, 3]. Boolean formula in Conjunctive Normal Form (CNF) is a representative case of such a complex object: it has two-level hierarchical structure. The CNF used in many practical application can be very large and inconvenient for direct analysis. The article presents an overview of decomposition methods for studying such CNFs. Boolean functions can be decomposed in many different ways.

In [4] the possible types of decomposition are categorized by:

• nesting depth in the resulting formula (“two-level” or “multi-level”),
• the sharing of variables between different variables sets (“disjoint” or “non-disjoint”),
• the purpose of the decomposition is performed: algebraic, logical or others.

Various decomposition techniques are based on different representations of Boolean function, such as: truth tables, normal forms (CNF, in particular), binary decision diagrams (BDDs), and so on.

Boolean functions are being used in studying and designing distinct objects and systems by logical modeling them. Therefore, when choosing the method of decomposition to apply, it is necessary to identify those properties of Boolean functions that describe essential features of objects being modeled.
Decomposition is being used both for functions represented in truth table (the function has a single truth table) and functions represented in formulas of certain type. In the second case, the same function can have many equivalent representations.

2. Decomposing Boolean functions represented in the truth table
In [5] George Boole proposed an idea of function decomposition.

Boole proposed an equivalent transformation of the function \( f(x_1, x_2, ..., x_n) \):
\[
f(x_1, ..., x_i, ..., x_n) = x_i \& f(x_1, ..., x_i = 1, ..., x_n) \lor \neg x_i \& f(x_1, ..., x_i = 0, ..., x_n).
\]

This property holds for exclusive or as well. In [6] Boole generalized this formula for multiple variables.

Later Claude Shannon developed a theory of expansions of Boolean functions [7]. In particular, Shannon developed a dual form of decomposition:
\[
f(x_1, ..., x_i, ..., x_n) = (x_i \lor f(x_1, ..., x_i = 1, ..., x_n)) \& (\neg x_i \lor f(x_1, ..., x_i = 0, ..., x_n)).
\]

The validity of such decomposition follows from the possibility of specifying any Boolean functions using the truth table.

One of the possible decompositions of Boolean functions is given by orthonormal expansions [8]. In particular, the Löwenheim expansion is well-known [9].

Different types of Boolean function decomposition can be shown using decomposition schemes. Examples of such notation are given in [4].

The truth table can be used only for small functions with little number of variables. For functions with large number of variables the most practically important representation is normal form.

3. Representing Boolean functions using normal forms
A normal form is standard two-level structure, where the internal functions, applied to literals (variables or its negations), are interconnected by external functions. A formula in normal form contains only threshold functions. The value of these functions on input vector is the same except just one position - either at the beginning of the truth table, or at the end. Threshold functions are being used in electronics because they are the most convenient for hardware and software implementation.

In the case of one variable, such functions are identity and negation. For two variables, the threshold functions are the following: conjunction, disjunction, Webb function (Peirce's arrow, negation of disjunction) and Sheffer stroke (negation of conjunction). Their generalizations for large number of variables are used as well [10].

The most commonly used normal forms are conjunctive and disjunctive normal forms (CNF and DNF). They use the same complete set of connectives \( \{\neg, \&, \lor\} \) in a different way. Any Boolean function, (except constant true and constant false respectively), can be represented in CNF and DNF. The use of CNF is more preferable for functions having more ones than zeros in output vector, because in that case the CNF has fewer internal functions – clauses. If the output vector has more zeros than ones, then for that function the DNF is more preferable [10].

Alternatives to CNF and DNF are considered in [10]. These are normal forms based on the Sheffer (\( \uparrow \)) and Webb (\( \uparrow \)) functions, which use the complete set of connectives \( \{\neg, \uparrow\} \) and \( \{\neg, \uparrow\} \) respectively. Since the Sheffer and Webb functions themselves are complete single-functional systems, Boolean functions can be represented using only \( \uparrow \) and \( \uparrow \).

Any boolean function can have infinite number of equivalent representations in a certain normal form. Therefore when using normal forms it is possible to select the best one regarding to some criterion (mathematical optimization). The most common criterion is the minimum number of literals in that normal form. In that case, that optimization is called minimization. The task of minimization can have more than one solution. The traditional methods for normal form minimization are Quine method [11, 12], Quine–McCluskey algorithm [13] and Karnougah Maps [14, 15].

There are other types of formulas for Boolean functions along with normal forms. In particular, Zhegalkin polynomial is well-known [16]. In additional to threshold function \& , it uses symmetrical functions \( \{0, 1, \oplus\} \). Unlike normal forms, Zhegalkin polynomial gives the only one representation of a Boolean function.
Among all the types of normal forms, the most commonly used is conjunctive.

4. Decomposing Boolean functions represented in Conjunctive Normal Forms
Decomposition is one of the preprocessing activities, simplifying further CNF analysis.
The main goal of decomposition is to maximize the preservation of relations between the elements of the original system. This fact largely determines the effectiveness of the subsequent analysis of the decomposed system.

CNF representations are used in different applications, including the SAT problem.
One of the possible forms of decomposition is representing CNF partitioning as hypergraph partitioning [17]. Many CNF applications generate very large formulas difficult to process and analyze. For example, large CNF appear in such applications as: covers in the relational database model [18] expert systems [19] representation of directed hypergraphs [20] and others. It is necessary to minimize these formulas. The minimal formula is a formula with minimum number of clauses or minimum number of literals.

Large formulas are beyond the capabilities of traditional minimizations algorithms - Quine, Quine–McCluskey, Karnagugh Maps. To minimize such a kind of formulas, the decomposition method is usually used [18, 19, 20]. Also, decomposition methods may be applied to a large CNF formula for proving its minimality [21].

A separate wide class of decomposition problems is minimization of the Horn formulae [22, 23, 24, 25].

Along with minimization, CNF decomposition method are used to parallelize the solution of the SAT problem using multiprocessor computing systems [26, 27].

SAT-solvers actually decompose CNF and the set of possible solutions. Usually, solvers decompose CNF the same way, as Boole and Shannon proposed: just decreasing the size of the task. To minimize the assignments set to be analyzed, they use different heuristic techniques without changing the basic principle: going through the set of possible solutions.

5. Conclusion
Decomposition is one of the main tools for studying Boolean functions, represented both in truth table and in normal forms. In performing the decomposition, it is necessary to consider the properties of the objects or systems being modeled, so that the decomposition clearly shows essential features of the system being modeled.

Methods of Boolean functions decomposition are considered with perspective of SAT problem, because it is mostly of theoretical and practical interest.

Traditional methods, presented by Boole and Shannon, defines universal recursive decomposition of Boolean functions. These methods allow to decrease the size of the task by switching to functions with fewer number of variables. In order to apply these ideas to effective solution of the SAT problem during CNF decomposition, it is needed to consider the structure of the formulas.

For that it is useful to apply general methods of decomposition proposed in [4].

Use of Boole’s and Shannon’s ideas of decreasing the size of Boolean functions, even with heuristic techniques does not change the basic principle – going through the set of possible solutions that grows exponentially with growing number of variables.

Graph methods of CNF decomposition allows to use the specifics of the structure of the certain CNFs. But using that methods does not sufficiently simplify the main task.

In particular, when decomposing CNF for SAT problem, it is preferable to keep only those connections that come from the CNF itself. It allows to significantly reduce the number of assignment to be analyzed.

It is optimal to select just part of the CNF and part of clauses. One of the way to do that is to equivalently convert the CNF to split it into blocks, for which getting the partial assignment is simpler.
References

[1] Dahl O J, Dijkstra E W, and Hoare C A R 1972 Structured Programming (GBR: Academic Press Ltd)
[2] Booch G 2004 Object-Oriented Analysis and Design with Applications (3rd Edition) (USA: Addison Wesley Longman Publishing Co Inc)
[3] Reilly E D 2004 Concise Encyclopedia of Computer Science (Hoboken: John Wiley & Sons Inc)
[4] Martinelli A 2006 Advances in Functional Decomposition: Theory and Applications
[5] Boole G 2018 The Mathematical Theories of Logic and Probabilities (New Delhi: Heritage Publishers)
[6] Boole G 2010 An Investigation of the Laws of Thought (Seaside: Watchmaker Publishing)
[7] Shannon C E 1949 The synthesis of two-terminal switching circuits Bell System Technical Journal 28 59–98
[8] Markham F B 1990 Boolean Reasoning: The Logic of Boolean Equations (USA: Springer)
[9] Löwenheim L 1910 Über die auflösung von gleichungen im logischen gebietekalkul Mathematische Annalen 68(2) 169–207
[10] Gdansky N 2011 Prikladnaya diskretnaya matematika. Logika. Grafy. Avtomaty. Algoritm. Kodirovanie (Moscow: Vuzovskaya Kniga)
[11] Quine W V 1952 The problem of simplifying truth functions The American Mathematical Monthly 59(8) 521
[12] Quine W V 1955 A way to simplify truth functions The American Mathematical Monthly 62(9) 627
[13] McCluskey E J 1956 Minimization of boolean functions* Bell System Technical Journal 35(6) 1417–44
[14] Karnaugh M 1953 The map method for synthesis of combinational logic circuits. Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics 72(5) 593–9
[15] Mondal S S, Nandy A, Agrawal R and Karnet D S: An efficient boolean function simplifier
[16] Steinbach S B, Posthoff C 2009 Logic functions and equations: examples and exercises Springer, c2009. xxii, 231 p. : ill. ; 25 cm. QA9 .S745 ISBN: 9781402095948 (hbk.)1402095945 (ebk.)9781402095955 (ebk.)
[17] Park T J and Gelder A V 2000 Partitioning methods for satisfiability testing on large formulas Inf. Comput. 162(1) 179–84
[18] Maier D 1980 Minimum covers in relational database model J. ACM 27(4) 664–74
[19] Peter L Hammer and Alexander Kogan 1994 Knowledge compression-logic minimization for expert systems.
[20] Ausiello G D’Atri A and Sacca D 1986 Minimal representation of directed hypergraphs. SIAM Journal on Computing 15(2) 418–31
[21] Boros E, Cepek O and Kuˇcera P 2013 A decomposition method for CNF minimality proofs Theoretical Computer Science 510 111–26
[22] Boros E, Cepek O and Kogan A 1998 Horn minimization by iterative decomposition. Annals of Mathematics and Artificial Intelligence 23(3–4) 321–43
[23] Cepek O and Kucera P 2008 On the complexity of minimizing the number of literals in horn formulæ
[24] Dowling W F and Gallier J H 1984 Linear-time algorithms for testing the satisfiability of propositional horn formulæ. The Journal of Logic Programming 1(3) 267–84
[25] Minoux M 1988 Ltur: A simplified linear-time unit resolution algorithm for horn formulæ and computer implementation Inf. Process Lett 29(1) 1–12
[26] El Khalek Y A, Safar M, and El-Kharashi M W 2015 On the parallelization of SAT solvers In 2015 Tenth International Conference on Computer Engineering & Systems (ICCES) IEEE
[27] Barrón-Romero C 2017 The fast parallel algorithm for CNF SAT without algebra CoRR abs/1701.04777