On Boolean Satisfiability

Abstract

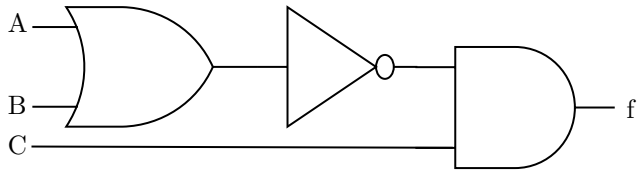
One question which has occupied mathematicians, engineers and scientists of this age is the problem of Boolean satisfiability (SAT) which asks whether a given CNF formula has at least a satisfying assignment. The goal of this work is to present a novel, efficient and general method of deciding SAT and finding all the satisfying assignments of CNF formulas. This method is powerful as it employs transformation identities in tranforming clauses without a particular variable into clauses with that variable.

SAT, satisfiable, resolution rule, transformation identity

1 Introduction

Suppose we are given the task of determining whether the output variable f of the digital circuit shown below evaluates to logic 1 for some assignments of logical values 0 and 1 to the input variables A, B, and C of the circuit. Now f is related to the input variables by

$$f = \overline{A}\,\overline{B}C.$$



The question is: *Is there a satisfying assignment of logical values* 0 *and* 1 *to* A, B *and* C, *i.e an assignment which makes* $f = \overline{A}\,\overline{B}C = 1$?. This is what the electronic engineer calls Circuit SAT.

What we commonly call SAT is by many termed Boolean satisfiability problem which imports the question as to whether a Boolean formula, e.g. $f = \overline{A}\,\overline{B}C$, has a satisfying assignment [6]. Boolean satisfiability is a propositional logic problem of interest in multiple fields, e.g., physics, mathematics, engineering and computer science. It is well-known that the SAT problem requires efficient solution methods in a variety of applications. It is believed to require exponentially growing time for an algorithm to solve for the worst-case instances. [2],[5] [4].

The goal of this paper is to present a novel, efficient and general method of solving SAT and finding all the satisfying assignments of CNF formulas. I have found beneath this emblem some of the richest and most glowing properties of satisfiability of CNF formulas.

The rest of this work is divided into two sections. Section 2 is concerned with the definitions and notations required to understand this work. Section 3 deals with the new method of solving SAT with ease as it makes use of transformation identities. I sincerely hope that this work may prove, as I intended, valuable to engineers, mathematicians, computer scientists and those involved in the design of SAT solvers.

2 Definitions and Notations

As there may be some of our readers who have little or no knowledge of Boolean Satisfiability, we deem it well to here call attention to the definitions and notations of terms associated with satisfiability.

Boolean Algebra is the algebra of truth values 0 representing *false* and 1 representing *true*. It was lauched by Boole and can be found in his work, the *Laws of Thought*, a gem, that for beauty of style and biilliancy of thought, has never been equalled on the subject of the Algebra of Logic.

Boolean variables are variables that can have the two possible truth values, 0 and 1. We use the capital letters of the English alphabet to denote variables. Thus the letters $A, B, C \ldots$ represent variables in this work. The next term on board is literal. With the common meaning which authors assigned this term, of course, we have so much to do. Some remarks, however, on the connection in which it stands, seem necessary for a proper understanding of the term. The **literals** of a variable L are its alternative forms, the unnegated L and the negated $\overline{L}$. They exhibit the properties $L\overline{L} = 0$ and $L + \overline{L} = 1$.

A **clause** is a logical sum of literals of different variables or a single literal. For example, $(A + \overline{B} + C)$ is a clause consisting of a the literals A, $\overline{B}$ and C. In arranging the literals in the clauses, we prefer the alphabetical order on account of its simplicity and ease to the reader. For instance, we shall write $(A + \overline{D} + F)$ instead of $(A + F + \overline{D})$. A clause is termed a **k-clause** if its number of literals is k. Thus, the clauses $(A), (A + \overline{B})$, and $(A + \overline{B} + C)$ are respectively 1-clause,

2-clause and 3-clause. A clause evaluates to logic 1 if a least one of its literals has been assigned a logical value of 1. If it evaluates to logic 0, it follows that each of its literals has been assigned a logical value of 0. Thus, if the clause $(A + \overline{B}) = 1$, then $A = 1$ or $\overline{B} = 1$ or both $A = 1$ and $\overline{B} = 1$. If the clause $(A + \overline{B}) = 0$ then $A = 0$ and $\overline{B} = 0$.

A **Boolean formula** is a logical expression defined over Boolean variables. Such a formula is in **conjunctive normal form** (CNF) if it is a logical product of clauses or a single clause. For example, the formula $f = (A+B)(\overline{A}+B+C)(\overline{D})$ is in CNF. A CNF formula is termed k-CNF formula when it contains clauses consisting of at most k literals.

To assist us in our reasoning, we shall invite the terms **native clauses** and **foreign clauses** to a formula f to describe respectively *clauses in* f and *clauses not in* f. Thus the native clauses of $f = (A + B)(\overline{A} + B + C)(\overline{D})$ are $(A + B), (\overline{A} + B + C)$, and $(\overline{D})$. The clause $(A + \overline{B})$ is a foreign clause to f because it is not in f.

A **Boolean assignment** to a set of Boolean variables is the set of truth values assigned to the variables or their literals in other to evaluate a Boolean formula. A **satisfying Boolean assignment** for a Boolean formula is an assignment such that the Boolean formula evaluates to logic 1. If the Boolean variables associated with a Boolean formula can be assigned logical values such that the formula turns out to be logic 1, then we say that the formula is *satisfiable*. If it is not possible to assign such values, then we say that the formula is *unsatisfiable*.

The above seem to be the most prominent terms, though not covering entirely the whole ground. There are however deeper terms of SAT that no language can describe, no illustration can reach, and no pencil can paint. And in endeavoring to bring out the valuable features of SAT, I have frequently been made sensible of the inadequacy of terms. For some good remarks on terms associated with SAT, see the paper

3 A Novel Approach for Solving SAT

It is our purpose to furnish a method for solving SAT, but before turning to the method, it will be necessary to consider two theorems needed in the method.

3.1 Two Special Theorems

Theorem 1. *Let L be a literal and $X_1, X_2, X_3, \ldots X_n$ be n clauses. Then*

$$f = (L + X_1)(L + X_2)(L + X_3) \cdots (L + X_n) = L + X_1X_2X_3, \ldots X_n.$$

Proof.

$$\begin{aligned} f &= (L + X_1)(L + X_2)(L + X_3) \cdots (L + X_n) \\ &= (LL + LX_2 + LX_1 + X_1X_2)(L + X_3) \cdots (L + X_n) \\ &= (L + X_1X_2)(L + X_3) \cdots (L + X_n) \\ &\vdots \\ &= L + X_1X_2X_3, \ldots X_n. \end{aligned}$$

□

There is a beautiful transformation identity that throws its light on SAT. This identity goes thus:

Theorem 2. *Let A, B and C be variables. Then*

$$(A + B)(\overline{A} + \overline{B})(B + C) = (A + B)(\overline{A} + \overline{B})(\overline{A} + C).$$

Mark here the transformation of clauses from those of only variable B to those of only variable A. I searched diligently in the most gigantic online libraries for a work on Boolean Satisfiability but I have never met with one embracing the use of this transformation identity. Before we pass on to its application in finding satisfying assignments of 2-CNF formulas, let us first seek the demonstration of its validity. A multitude of proofs might here be presented, but one is sufficient.

Proof. We shall employ the illustrious resolution rule. If we apply this rule to the expression

$$(A + B)(\overline{A} + \overline{B})(B + C)$$

we shall have

$$(A + B)(\overline{A} + \overline{B})(B + C)(\overline{A} + C) \tag{1}$$

where the resulting clause $(\overline{A} + C)$ is the resolvent of the parent clauses $(\overline{A} + \overline{B})$ and $(B + C)$.

We turn now to the application of the resolution rule to the second member of the identity, namely

$$(A + B)(\overline{A} + \overline{B})(\overline{A} + C).$$

The result is

$$(A + B)(\overline{A} + \overline{B})(\overline{A} + C)(B + C) \tag{2}$$

where the resulting daughter clause $(B + C)$ is the resolvent of the parent clauses $(A + B)$ and $(\overline{A} + C)$.

Now the two expressions (1) and (2) are essentially the same. Hence we reach the conclusion that the transformation identity proposed is valid.

A world of other identities can be derived from the above-mentioned identity. For if we set $A = \overline{A}$ in the identity we shall obtain the new identity

$$(\overline{A} + B)(A + \overline{B})(B + C) = (\overline{A} + B)(A + \overline{B})(A + C).$$

In this new identity, let us set $B = \overline{B}$. We get

$$(A + B)(\overline{A} + \overline{B})(\overline{B} + C) = (A + B)(\overline{A} + \overline{B})(A + C).$$

The following two examples on the testing of satisfiability of 2-CNF formulas will be enriched using the transformation identity.

Example 1. *Test the satisfiability of*

$$f = (A + B)(\overline{A} + C)(\overline{B} + \overline{C}).$$

We employ the famous resolution rule here. This introduced resolvents and f becomes

$$f = (A + B)(\overline{A} + C)(\overline{B} + \overline{C})(B + C)(A + \overline{C})(\overline{A} + \overline{B}).$$

The product $(A + B)(\overline{A} + \overline{B})$ is a transforming product. It transforms $(\overline{B} + \overline{C})$ into $(A + \overline{C})$ and $(B + C)$ into $(\overline{A} + C)$. Hence f is transformed into

$$f = (A + B)(\overline{A} + C)(A + \overline{C})(\overline{A} + C)(\overline{A} + \overline{B}).$$

which, applying Theorem 1, becomes

$$f = (A + B\overline{C})(\overline{A} + \overline{B}C)$$

which in turn becomes

$$f = A\overline{B}C + \overline{A}B\overline{C}.$$

Since f does not simplify to 0, we conclude that f is satisfiable and has the satisfying assignment of

$$\{(A = 1, \overline{B} = 1, C = 1), (\overline{A} = 1, B = 1, \overline{C})\}.$$

Let us take up the other instance.

Example 2. *Decide the satisfiability of*

$$f = (A + B)(\overline{A} + \overline{B})(A + \overline{C})(\overline{A} + C)(B + \overline{C})(\overline{B} + C).$$

The product $(A + B)(\overline{A} + \overline{B})$ is a transforming product. By it we transform $(B + \overline{C})$ into $(\overline{A} + \overline{C})$ and $(\overline{B} + C)$ into $(A + C)$. Hence, we get

$$\begin{aligned} f &= (A + B)(\overline{A} + \overline{B})(A + \overline{C})(\overline{A} + C)(\overline{A} + \overline{C})(A + C) \\ &= (A + B\overline{C}C)(\overline{A} + \overline{B}C\overline{C}) \\ &= (A)(\overline{A}) \\ &= 0. \end{aligned}$$

Since f is logic 0, we say that f is unsatisfiable.

3.2 A General Method of Solving SAT

Our chief business now is to present a general method of deciding SAT and then finding all the satisfying assignments of CNF formulas. Let $f^{(0)}$ be a given CNF formula comprising clauses with variable A and those without A. Also let $f_a^{(0)}$ be a CNF formula consisting of native clauses of $f^{(0)}$ and foreign clauses of $f^{(0)}$ with variable A such that $f_a^{(0)}$ contains transforming products capable of transforming all the native clauses without A into clauses with A. Let $f_i^{(0)}$ be the CNF expression consisting of all the introduced foreign clauses with A. We have therefore the relation

$$f_a^{(0)} = f_i^{(0)} f^{(0)}.$$

Let us enter a little more into detail. If $f_a^{(0)}$ is satisfiable, then it follows that $f^{(0)}$ is satisfiable and *all* the satisfying assignments of $f_a^{(0)}$ are *all* or *some* of the satisfying assignments of $f^{(0)}$. If *all* the satisfying assignments of $f_a^{(0)}$ are *some* of the satisfying assignments of $f^{(0)}$, then the rest must be *all* the satisfying assignments of the new CNF formula

$$f^{(1)} = \overline{f_i^{(0)}} f^{(0)}$$

for

$$f_a^{(0)} + f^{(1)} = f^{(0)}.$$

If $f_a^{(0)}$ is unsatisfiable, it does not follow that $f^{(0)}$ is unsatisfiable but that *all* the satisfying assignments of $f^{(0)}$ are the satisfying assignments of $f^{(1)}$. The given CNF formula is unsatifiable only when both $f_a^{(0)}$ and $f^{(1)}$ are unsatisfiable.

To obtain all the satisfying assignments it may be necessary to extend the computation to

$$f^{(k)} = \overline{f_i^{(k-1)}} f^{(k-1)}$$

where $k = 2, 3, 4, \ldots$ up to when one arrives at the results $f_a^{(k)} = 0$ and $f^{(k+1)} = 0$ or at a new CNF formula $f^{(k)}$ whose satisfying assignments can be found without introducing foreign clauses.

To render the work still more worthy of the public, I have presented few instances of its application.

Example 3. *Decide the satisfiability of* $f^{(0)} = (A + B)(B + C)(C + D)$. *If it is satifiable, then find all the satisfying assignments.*

To replace the native clause $(B + C)$ with a clause of A, we append the foreign clause $(\overline{A} + \overline{B})$ to $f^{(0)}$. We employ this foreign clause because it is the alternate clause to the native clause $(A + B)$ and hence its multiplication with the partial CNF expression $(A + B)(B + C)$ furnishes the CNF expression $(\overline{A} + \overline{B})(A + B)(\overline{A} + C)$. We thus have the new formula with the appended foreign clause

$$f_a^{(0)} = (\overline{A} + \overline{B})\ (A + B)(\overline{A} + C)(C + D).$$

We turn to the replacement of the native clause $(C+D)$ with a clause of A. To achieve this, we introduce the foreign clause $(A+\overline{C})$ which is the alternate of the native clause $(\overline{A}+C)$. Multiplying the foreign clause by the partial CNF expression $(\overline{A}+C)(C+D)$ gives the CNF expression $(A+\overline{C})(\overline{A}+C)(A+D)$. Hence the formula with an appended foreign clause is improved to

$$f_a^{(0)} = (A+B)(\overline{A}+\overline{B})(\overline{A}+C)(A+\overline{C})(A+D).$$

This is simplified further to

$$f_a^{(0)} = \overline{A}\,B\overline{C}D + A\overline{B}C.$$

Since $f_a^{(0)}$ is not equal to logic 0, we say that the original CNF formula $f^{(0)}$ is satisfiable. If we set $f^{(0)}=1$, we obtain three of the satisfying assignments of $f^{(0)}$, namely

$$\{(A=1,\overline{B}=1,C=1,D=\{0,1\}),(\overline{A}=1,B=1,\overline{C}=1,D=1)\}$$

To get more satisfying assignments, let us first determine the complement of the CNF formula consisting of the foreign clauses of A used in obtaining $f_a^{(0)}$. The formula is

$$f_i^{(0)} = (\overline{A}+\overline{B})(A+\overline{C}) = \overline{A}\,\overline{C} + A\overline{B}.$$

So

$$\overline{f_i^{(0)}} = (A+C)(\overline{A}+B).$$

Hence,

$$\begin{aligned} f^{(1)} &= (A+C)(A+B)(\overline{A}+B)(B+C)(C+D) \\ &= (B)(A+C)(C+D). \end{aligned}$$

This is a new CNF formula and we recommence the entire process. Thus,

$$\begin{aligned} f_a^{(1)} &= (B)(A+C)(\overline{A}+\overline{C})(\overline{A}+D) \\ &= AB\overline{C}D + \overline{A}BC. \end{aligned}$$

Hence, we obtain the three satisfying assignments

$$\{(A=1,B=1,\overline{C}=1,D=1),(\overline{A}=1,B=1,C=1,D=\{0,1\})\}.$$

For more satisfying assignments, we find the complement of the CNF formula consisting of only the foreign clauses used:

$$f_i^{(1)} = (\overline{A}+\overline{C})$$

which becomes

$$\overline{f_i^{(1)}} = (A)(C).$$

Appending this complement to $f^{(1)}$ furnishes the new CNF formula

$$f^{(2)} = (A)(C)(B)(A+C)(C+D)$$

which reduces to

$$f^{(2)} = ABC.$$

We get two more satisfying assignments:

$$\{(A = 1, B = 1, C = 1, D = \{0, 1\})\}.$$

There are altogether eight satisfying assignments of $f^{(0)}$.

Example 4. *Determine all the satisfying assignments of*

$$f^{(0)} = (A + C)(B + \overline{C})(\overline{B} + D)(C + \overline{D}).$$

We begin with the CNF formula consisting of the native clauses of A, the foreign clauses of A and the derived clauses of A obtained from the use of the transformation identities,

$$\begin{aligned} f_a^{(0)} &= (A + C)(A + B)(\overline{A} + \overline{B})(\overline{A} + \overline{C})(A + D)(\overline{A} + \overline{D}) \\ &= (A + BCD)(\overline{A} + \overline{B}\,\overline{C}\,\overline{D}) \\ &= A\overline{B}\,\overline{C}\,\overline{D} + \overline{A}BCD. \end{aligned}$$

From this we obtain two satisfying assignments of f^0:

$$\{(A = 1, \overline{B} = 1, \overline{C} = 1, \overline{D} = 1), (\overline{A} = 1, B = 1, C = 1, D = 1)\}.$$

We turn to the CNF formula of foreign clauses.

$$f_i^{(0)} = (A + B)(\overline{A} + \overline{B}) = A\overline{B} + \overline{A}B.$$

So

$$\overline{f^{(0)}}_i = (\overline{A} + B)(A + \overline{B}).$$

We append this to $f^{(0)}$ and get

$$f^{(1)} = (\overline{A} + B)(A + \overline{B})(A + C)(B + \overline{C})(\overline{B} + D)(C + \overline{D}).$$

For this new CNF formula, we employ the entire procedure again. The CNF formula consisting of the native clauses with A, foreign clauses of A and derived clauses obtained by using the transformation identities is as follows:

$$f_a^{(1)} = (\overline{A} + B)(A + \overline{B})(A + C)(A + \overline{C})(\overline{A} + D)(\overline{A} + \overline{C})(\overline{A} + \overline{D}) = 0.$$

We get no satisfying assignment here. We go further to find the CNF formula consisting of the foreign clauses. Thus we get

$$f_i^{(1)} = (\overline{A} + \overline{C})$$

which becomes

$$\overline{f^{(1)}}_i = (A)(C).$$

Hence we have the new CNF formula

$$f^{(2)} = (A)(C)(\overline{A}+B)(A+\overline{B})(A+C)(B+\overline{C})(\overline{B}+D)(C+\overline{D})$$

which simplifies to

$$f^{(2)} = ABCD.$$

We therefore get the satisfying assignments

$$\{(A=1, B=1, C=1, D=1)\}$$

.

That the reader may see the great beauty and force of this method, we present an instance involving 3-SAT.

Example 5. *Decide the satisfiability of*

$$f^{(0)} = (A+B+E)(A+C+\overline{E})(A+\overline{B}+\overline{D})(\overline{A}+D+E)(B+C+D)$$
$$(\overline{B}+C+D)(C+D+E)(\overline{C}+\overline{D}+\overline{E})$$

We start with the formula consisting of the native clauses of A, the foreign clauses of A and the derived clauses of A obtined by employing transformation identities:

$$\begin{aligned} f_a^{(0)} &= (A+B)(\overline{A}+\overline{B})(A+C)(\overline{A}+\overline{C})(A+B+E)(A+C+\overline{E})(A+\overline{B}+\overline{D}) \\ &\quad (\overline{A}+D+E)(\overline{A}+C+D)(A+C+D)(\overline{A}+D+E)(A+\overline{D}+\overline{E}) \\ &= A(D+E)(\overline{B})(C+D)(\overline{C}) + (\overline{A})(B+E)(C+\overline{E})(\overline{B}+\overline{D})(B)(C+D)(C) \\ &= A\overline{B}\,\overline{C}D + \overline{A}BC\overline{D}. \end{aligned}$$

Since $f_a^{(0)}$ is not equal to 0, we conclude that the original formula is satisfiable and has the satisfying assignments of $\{(A=1, \overline{B}=1, \overline{C}=1, D=1), (\overline{A}=1, B=1, C=1, \overline{D}=1)\}$. Other satisfying assignments can be obtained as demonstated in the previous examples.

What a glorious method is presented to our view! But we must stop here, for our limit reminds us that we must be brief.

References

[1] Aaronson S. and Wigderson A., Algebrization: a new barrier in complexity theory, in STOC, 2008, pp. 731–740

[2] Aho, Alfred V.; Hopcroft, John E.; Ullman, Jeffrey D. (1974). The Design and Analysis of Computer Algorithms. Addison-Wesley. Theorem 10.4.

[3] Baker T. P., Gill J, and Solovay R., Relativizations of the P=?NP question, SIAM Journal on Computing 4:4,431-442, 1975

[4] Cook S. A., The complexity of theorem proving procedures, Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 151-158, 1971.

[5] Davis M., Logemann G. , and Loveland D., Communications of the ACM 5, 394–397 (1962).

[6] Mironov, Ilya; Zhang, Lintao (2006). Biere, Armin; Gomes, Carla P. (eds.). "Applications of SAT Solvers to Cryptanalysis of Hash Functions". Theory and Applications of Satisfiability Testing — SAT 2006. Lecture Notes in Computer Science. Springer. *Mathematics Handbook for Science and Engineering*, Springer, New York, 2006, 5th ed.