1

Abstract

Description of a polynomial time reduction of SAT to 2-SAT of polynomial size.

Reducing SAT to 2-SAT

Sergey Gubin

February 16, 2013

1 Introduction

Among all dimensions, 2-SAT possesses many special properties unique in the sense of computational complexity [1, 2, 3, 4, 5]. But in light of works [6, 8, 7, 9] a problem arose: either those properties are accidental or there are polynomial time reductions of SAT to 2-SAT of polynomial size. This article describes one such reduction.

2 Presenting SAT with XOR

In [6] was described one of the ways to present SAT with a conjunction of XOR. Let us summarize it.

Let Boolean formula f define a given SAT instance:

$$f = c_1 \wedge c_2 \wedge \ldots \wedge c_m. \tag{1}$$

Clauses c_i are disjunctions of literals:

$$c_i = L_{i1} \vee L_{i2} \vee \ldots \vee L_{in_i}, \ i = 1, 2, \ldots, m$$

- where n_i is the number of literals in clause c_i; and L_{ij} are the literals. Using distributive laws, formula (1) can be rewritten in disjunctive form:

$$f = d_1 \vee d_2 \vee \ldots d_p, \ p = n_1 n_2 \ldots n_m.$$

Clauses d_k in this presentation are conjunctions of m literals - one literal from each clause $c_i, \ i = 1, 2, \ldots, m$:

$$d_k = L_{1k_1} \wedge L_{2k_2} \wedge \ldots \wedge L_{mk_m}, \ k = 1, 2, \ldots, p. \tag{2}$$

---

*Author's email: sgubin@genesyslab.com

It is obvious that formula (1) is satisfiable iff there are clauses without complimentary literals amongst conjunctive clauses (2). Disjunction of all those clauses is the disjunctive normal form of formula (1). Thus, formula (1) is satisfiable iff there are members in its disjunctive normal form.

There is a generator for conjunctive clauses (2):

$$g = \bigwedge_{i=1}^{m} (\xi_{i1} \oplus \xi_{i2} \oplus \ldots \oplus \xi_{in_i}) = true, \tag{3}$$

- where Boolean variable $\xi_{\mu\nu}$ indicates whether literal $L_{\mu\nu}$ participates in conjunction (2). Solutions of equation (3) generate conjunctive clauses (2). Let's call the variables ξ the indicators. To select from all solutions of equation (3) those without complimentary clauses, let's use another Boolean equation.

For each of the combination of clauses (c_i, c_j), $1 \leq i < j \leq m$, let's build a set of all couples of literals participating in the clauses:

$$A_{ij} = \{ (L_{i\mu}, L_{j\nu}) \mid c_i = L_{i\mu} \vee \ldots; \ c_j = L_{j\nu} \vee \ldots \}.$$

Let B_{ij} be a set of such couples of indicators $(\xi_{i\mu}, \xi_{j\nu})$, that the literals they present are complimentary:

$$B_{ij} = \{ (\xi_{i\mu}, \xi_{j\nu}) \mid (L_{i\mu}, L_{j\nu}) \in A_{ij}, \ L_{i\mu} = \bar{L}_{j\nu} \}.$$

There are C_m^2 sets B_{ij}, $1 \leq i < j \leq m$, and

$$|B_{ij}| \leq \min\{n_i, n_j\}.$$

Let's mention that some of the sets can be empty. Then, the following equation will select from all solutions of equation (3) those without complimentary clauses:

$$h = \bigwedge_{1 \leq i < j \leq m} \bigwedge_{(\xi,\zeta) \in B_{ij}} (\bar{\xi} \vee \bar{\zeta}) = true. \tag{4}$$

Due to the above estimations of the number of sets B_{ij} and of their sizes, the number of clauses in formula (4) is

$$n = O(t_2 m^2),$$

- where t_2 is the second number in the row of clauses' sizes sorted by value:

$$t_1 = \max\{n_1, n_2, \ldots, n_m\}, \ t_2 = \max_{i<j} \min\{n_i, n_j\}, \ \ldots$$

Because satisfiability of formula (1) means that the disjunctive normal form of formula (1) has conjunctive clauses, formula (1) is satisfiable iff the following formula/equation is satisfiable:

$$g \wedge h = true. \tag{5}$$

The reasons for replacing formula (1) with formula (5) are explained in [6]. The number of *true*-strings in truth-tables of XOR clauses of formula (3) is linear over initial input. The number of *true*-strings in truth-tables of disjunctive clauses of formula (4) is just 3. The number of all clauses in (5) is cubic over initial input. It can be estimated as

$$m + n = O(t_2 m^2).$$

Thus, application of the simplified compatibility matrices method [6] to equation (5) will produce a polynomial time algorithm for SAT. But let's return to the reduction.

3 SAT vs. 2-SAT

Let's apply the simplified method of compatibility matrices [6] to equation (5). The method consists of sequential Boolean transformations of compatibility matrices of equation (5). Let's mention that after m iterations, due to the allocation of formula (4) at the end of formula (5), there will only be compatibility matrices of equation (4) left in play. They will be grouped in an upper triangular box matrix

$$S = (F_{m+\mu, m+\nu})_{1 \leq \mu < \nu \leq n}. \tag{6}$$

The matrix is displayed below:

$F_{m+1,m+2}$	$F_{m+1,m+3}$	$\cdots$	$F_{m+1,m+n}$
	$F_{m+2,m+3}$	$\cdots$	$F_{m+2,m+n}$
		$\ddots$	$\vdots$
			$F_{m+n-1,m+n}$

If there are no complimentary literals in different clauses of formula (1), then formula (4) is just missing. The size of matrix (6) is 0×0. In this case, formula (1) is reducible to 1-SAT instance

$$\omega_1 \wedge \omega_2 \wedge \ldots \wedge \omega_m,$$

- where
$$\omega_i = \xi_{i1} \oplus \xi_{i2} \oplus \ldots \oplus \xi_{in_i}, \ i = 1, 2, \ldots, m.$$

This singularity belongs to the set of all 2-SAT instances.

If, during the first m iterations, a pattern of unsatisfiability arises (one of the compatibility matrices becomes filled with *false* entirely), then formulas

(5) and (1) are both unsatisfiable [6]. This case may be thought of as a case of formula (1) being reduced to an unsatisfiable formula

$$false.$$

Let's include this singularity in the set of all 2-SAT instances.

Otherwise, boxes $F_{m+\mu,m+\nu}$ in matrix (6) are what is left of the compatibility matrices of equation (4) after the first m iterations of the method.

Due to their construction [6], the boxes are 3×3 matrices:

$$F_{m+\mu,m+\nu} = (x_{ij})_{3\times3}, \ 1 \leq \mu < \nu \leq n \tag{7}$$

- where $x_{ij} \in \{false, true\}$. The number of boxes is C_n^2. Thus, the number of all elements in matrix (6) is

$$e = 9C_n^2 = O(t_2^2 m^4).$$

Let's enumerate the elements arbitrarily:

$$y_1, \ y_2, \ldots, y_e.$$

Then, distribution of $true/false$ in matrix (6) can be described with a 1-SAT formula/equation

$$w = \eta_1 \wedge \eta_2 \ldots \wedge \eta_e = true, \tag{8}$$

- where η_i are literals over a set of Boolean variables

$$\{ \ b_1, \ b_2, \ldots, b_e \ \}.$$

The literals are

$$\eta_i = \begin{cases} b_i, & y_i = true \\ \bar{b_i}, & y_i = false \end{cases}, \ \ i = 1, 2, \ldots, e.$$

Let's take the following 2-SAT instance:

$$h \wedge w. \tag{9}$$

Box matrix (6) is an initialization of the modified method of compatibility matrices [6] for formula (9): compatibility matrices of formula (4) are depleted to satisfy equation (8). Thus, continuation of the simplified method of compatibility matrices for equation (5) from its Step $m+1$ to its finish is an application of the modified method of compatibility matrices to system (9) from its Step 1 to its finish [6]. After $n-2$ iterations, both methods must result with the same version of satisfiability of formula (1). Thus, formulas (5) and (1) are satisfiable iff 2-SAT formula (9) is satisfiable. The number of clauses in formula (9) is

$$e + n = O(t_2^2 m^4).$$

According to [6], the time to deduce formula (9) can be safely estimated as

$$O(t_1^4 t_2^4 m^6)$$

4   SAT vs. 1-SAT

Let's take one step further. Applying to formula (1)/(5) either of the variations of the compatibility matrices method [6] will produce a Boolean matrix. Let it be a matrix R:

$$R = (r_{ij})_{a \times b}.$$

Size of the matrix depends on the method's variation and the order of clauses in formula (1). The size can be changed if permute the clauses and repeat the method [6]. The formula (1) is satisfiable iff matrix R contains $true$-elements [6] (elements which are $true$). The existence/absence of the $true$-elements is the only invariant.

   If formula (1) is unsatisfiable, then that formula is reducible to formula "$false$". Otherwise, formula (1) is reducible to a 1-SAT instance.

Proof. Let's enumerate elements of matrix R in arbitrarily order:

$$z_1, z_2, \ldots, z_{ab}.$$

Let B be a set of $t = ab$ Boolean variables:

$$B = \{ \, b_i \in \{false, true\} \mid i = 1, 2, \ldots, t \, \}.$$

Then the following 1-SAT formula describes distribution of $true/false$ in matrix R:

$$\theta_1 \wedge \theta_2 \wedge \ldots \wedge \theta_t, \tag{10}$$

- where literals θ_i are

$$\theta_i = \begin{cases} b_i, & z_i = true \\ \overline{b}_i, & z_i = false \end{cases}, \quad i = 1, 2, \ldots, t.$$

Thus, the compatibility matrices method reduces satisfiable formula (1) to 1-SAT formula (10).   □

   In its turn, formula (10) can be rewritten as SAT of any dimension by appropriate substitution of variables.

   If use the simplified method of compatibility matrices, then matrix R is a 3×3 Boolean matrix [6]. Let there be two clauses shorter than 3 in formula (1). Let's permute all clauses and make those shortest clauses to be the last ones in formula (1). Then, result of the modified method [6] will be a matrix R of size less than 3×3. That proves the following theorem.

   Theorem 1. *Any SAT instance is reducible to a 1-SAT instance with* 9 *variables or less. A SAT instance is unsatisfiable iff its 1-SAT presentation is "$false$" - there is not any variables in its 1-SAT presentation.*

5   Conclusions

Formula (1) may be thought of as a "Business Requirements". And any appropriate computer program may be thought of as a solution of the SAT instance. Then, theorem 1 can be an explanation of the remarkable efficiency of the "natural programs". From this point of view, the iterations of the method of compatibility matrices may be thought of as a learning/modeling of the business domain. In the artificial programming, the calculation of the compatibility matrices - a virtual business domain - could be a conclusion of the stage "Business Requirements Analysis/Mathematical Modeling". That would improve the programs' performance. The resulting compatibility matrices may be thought of as a fussy logic's tables of rules for the domain.

The whole solution of formula (1) can be achieved, with one of the following approaches, for example. ANN approach is the applying of the compatibility matrices method backward, starting from matrix R. An example of that can be found in [7]. DTM approach is the looping trough of the following three steps: selection of any $true$-element from matrix R; substitution of the appropriate $true$-assignments in formula (1); and repeating of the compatibility matrices method. The last method is an implication of the self-reducibility property of SAT [5].

In certain sense, theorem 1 may be seen as an answer to the Feasibility Thesis [2].

References

[1] Stephen Cook. The complexity of theorem-proving procedures. In Conference Record of Third Annual ACM Symposium on Theory of Computing. p.151-158, 1971

[2] Stephen Cook. The P versus NP problem. http://www.claymath.org/millennium/P_vs_NP/pvsnp.pdf

[3] Richard M. Karp. Reducibility Among Combinatorial Problems. In Complexity of Computer Computations, Proc. Sympos. IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y. New York: Plenum, p.85-103, 1972.

[4] M.R. Garey and D.S. Johnson. Computers and Intractability, a Guide to the Theory of NP-Completeness. W.H. Freeman and Co. San Francisco, 1979.

[5] Lane A. Hemaspaandra, Mitsunori Ogihara. The Complexity Theory Companion. Springer-Verlag Berlin Heidelberg, 2002.

[6] Sergey Gubin. A Polynomial Time Algorithm for SAT. http://www.arxiv.org/pdf/cs/0703146

[7] Sergey Gubin. A Polynomial Time Algorithm for 3-SAT. Examples of use. http://www.arxiv.org/pdf/cs/0703098

[8] Sergey Gubin. A Polynomial Time Algorithm for 3-SAT. http://www.arxiv.org/pdf/cs/0701023

[9] Sergey Gubin. A Polynomial Time Algorithm for The Traveling Salesman Problem. http://www.arxiv.org/pdf/cs/0610042