19 March 2018 — The SAT Problem

SAT is a family of problems in Boolean logic.

① CNF-SAT. (CNF stands for "conjunctive normal form")

Given n Boolean variables $x_1, x_2, \dots, x_n$

m clauses

Each clause is a disjunction (Boolean OR) of one or more literals (a variable or its negation)

Example. $(\overline{x_2} \vee x_4 \vee \overline{x_6} \vee x_7)$

Does there exist a truth assignment of the variables that satisfies every clause? (A clause is satisfied if at least one of its literals evaluates to TRUE.)

② CIRCUIT SAT.

Given n Boolean variables and a program consisting of m steps each of which is either

- look up the value of a variable x_i
- perform the operation AND, OR, or NOT on operands which are the outcomes of earlier steps

Decide whether $\exists$ a truth assignment of the variables that makes the last line of the program evaluate to TRUE.

Variables: x_1, x_2, x_3, x_4

Program:

1: $y_1 = x_2$
2: $y_2 = x_4$
3: $y_3 = \overline{y_1}$
4: $y_4 = x_3$
5. $y_5 = y_3 \wedge y_4$

CIRCUIT SAT reduces in polynomial time to CNF SAT: given a circuit sat problem instance, create a CNF-SAT instance with variables $x_1, \dots, x_n, y_1, \dots, y_m$.

Create clauses as follows:

1. $y_i = x_j$ becomes $(y_i \vee \overline{x_j}) \wedge (\overline{y_i} \vee x_j)$
2. $y_i = \overline{y_j}$ becomes $(y_i \vee y_j) \wedge (\overline{y_i} \vee \overline{y_j})$
3. $y_i = y_j \vee y_k$ becomes $(\overline{y_i} \vee y_j \vee y_k) \wedge (y_i \vee \overline{y_j}) \wedge (y_i \vee \overline{y_k})$
4. $y_i = y_j \wedge y_k$ becomes $(\overline{y_i} \vee y_j) \wedge (\overline{y_i} \vee y_k) \wedge (y_i \vee \overline{y_j} \vee \overline{y_k})$
5. Final clause asserts y_m.

CNF SAT reduces (also in linear time) to CIRCUIT SAT just by writing down a program to evaluate the clauses of the CNF formula one at a time and then evaluate their conjunction.

E.g. $(x_1 \vee \overline{x_2}) \wedge (x_1 \vee x_2 \vee x_3)$ becomes

$y_1 = x_1$
$y_2 = x_2$
$y_3 = \overline{y_2}$
$y_4 = y_1 \vee y_3$ // 1st clause
$y_5 = y_1 \vee y_2$
$y_6 = x_3$
$y_7 = y_5 \vee y_6$ // 2nd clause
$y_8 = y_4 \wedge y_7$

A truth assignment of $x_1, \dots, x_3$ satisfies the last line of the program if and only if it satisfies every CNF clause.

Important remark. The reduction CIRCUIT SAT to CNF SAT on previous page produces clauses with ≤ 3 literals per clause.

The special case of CNF SAT where each clause has ≤ 3 literals is denoted by 3SAT. And the reductions on this page and the previous one show that 3SAT is at least as hard (computationally) as the general case of CIRCUIT SAT and CNF SAT.

If 3SAT is easy then CNF SAT is easy.
(By combining preceding 2 reductions.)
If CNF SAT is easy then 3SAT is easy.
(Because a 3SAT problem is a special case of a CNF SAT problem.)
∴ CNFSAT and 3SAT are "equivalent under polynomial time reductions"
(CIRCUIT SAT is also equivalent to both of them, by analogous reasoning.)

Next up: Reduce INTEGER FACTORING to CIRCUIT SAT.

Addition of binary numbers can be simulated by a CIRCUIT SAT program.

"carry digits"

$$\begin{array}{cccccc} & d_1 & d_2 & \cdots & d_{n-1} & \\ & a_1 & a_2 & \cdots & & a_n \\ + & b_1 & b_2 & \cdots & & b_n \\ \hline c_0 & c_1 & \cdots & & & c_n \end{array}$$

Define the $\oplus$ operation by specifying that $y_i = y_j \oplus y_k$ is shorthand for

$$y_i' = y_j \wedge \overline{y_k}$$
$$y_i'' = \overline{y_j} \wedge y_k$$
$$y_i = y_i' \vee y_i''$$

Adder:

$$c_n = a_n \oplus b_n$$
$$d_{n-1} = a_n \wedge b_n$$
$$c_{n-1} = (d_{n-1} \oplus a_{n-1}) \oplus b_{n-1}$$
$$d_{n-2} = \cdots$$
$$\vdots$$

shorthand for a pair of lines $c_{n-1}' = d_{n-1} \oplus a_{n-1}$, $c_{n-1} = c_{n-1}' \oplus b_{n-1}$

And so on. The whole addition algorithm translates into Boolean logic.

Multiplication: fill in a table using AND function, then do iterated addition.

$$\begin{array}{ccccc} & & a_1 & a_2 & a_3 \\ & & b_1 & b_2 & b_3 \\ \hline & & (a_1 \wedge b_3) & (a_2 \wedge b_3) & (a_3 \wedge b_3) \\ & (a_1 \wedge b_2) & (a_2 \wedge b_2) & (a_3 \wedge b_2) & \\ (a_1 \wedge b_1) & (a_2 \wedge b_1) & (a_3 \wedge b_1) & & \\ \hline \end{array}$$

Add up these 3 lines

Factoring a n-bit number $c_1\ c_2 \cdots c_n$ becomes a circuit sat problem by writing a program that takes 2 integers in binary

$(a_1\ a_2 \ldots a_n)$ $(b_1 \ldots b_n)$

and verifies:

1. their product is $(c_1 \ldots c_n)$
2. $a_1 \ldots a_n \neq 000 \ldots 1$
3. $b_1 \ldots b_n \neq 000 \ldots 1$

A satisfying truth assignment of this circuit is equivalent to a factorization of $(c_1 \cdots c_n)$ into integers > 1.

CIRCUIT SAT is believed to be much harder than factoring.