On the complexity of SAT

Fabio Romano

March 23, 2022

Abstract

In this paper, we prove that no deterministic algorithm can solve SAT in polynomial time in the number of boolean variables.

1 First definitions

Definition 1.1 (Set of input symbols $\Sigma$). The set of input symbols $\Sigma$ for SAT language is made-up by:

- The symbols $x, 0, 1$.
- Binary operators $\land$ and $\lor$, standing for the logical AND and OR of two expressions.
- Unary operator $\neg$, standing for logical negation.
- Parentheses to group operators and operands, if necessary to alter the default precedence of operators: $\neg$ highest, then $\land$ and finally $\lor$.

Cf. [1, sect. 10.2].

Definition 1.2 (Boolean variables). If $w \in \{0, 1\}^*$ is a string starting with 1, then $xw$ is a boolean variable, where $xw$ is the concatenation of symbol $x$ with $w$. $w$ is the binary representation of the index of variable $xw$.

Cf. [1, par. 10.2.2].

Definition 1.3 (Boolean expressions).

- Every boolean variable is a boolean expression.
- If $A$ and $B$ are boolean expressions, then $(A)$, $\neg A$, $A \land B$ and $A \lor B$ are boolean expressions.

Cf. [1, par. 10.2.1].

Definition 1.4 (Length of a boolean expression). The length of a boolean expression $E$ is the number of positions for symbols in $E$. We denote the length of $E$ with $|E|$. 
Definition 1.5 (Set of boolean expressions not longer than \(n\)). For every \(n \in \mathbb{N}\), the set of boolean expressions not longer than \(n\) is defined as follows:

\[
\text{BoolExp}_{\leq}(n) \overset{\text{def}}{=} \{E \mid E \text{ is a boolean expression s. t. } |E| \leq n\}
\]

From now on, we assume that \(0 \notin \mathbb{N}\).

Definition 1.6 (Truth assignment). A truth assignment for a given boolean expression \(E\) assigns either 1 or 0 to each of the variables mentioned in \(E\). The value of expression \(E\) given a truth assignment \(T\), denoted \(E(T)\), is the result of evaluating \(E\) with each variable \(x\) replaced by the value \(T(x)\) that \(T\) assigns to \(x\) (cf. [1, par. 10.2.1]).

Definition 1.7 (Satisfiability). A truth assignment \(T\) satisfies a boolean expression \(E\) if \(E(T) = 1\); i.e., the truth assignment \(T\) makes expression \(E\) true. A boolean expression \(E\) is said to be satisfiable if there exists at least one truth assignment \(T\) that satisfies \(E\) (cf. [1, par. 10.2.1]).

Definition 1.8 (SAT \(\subseteq \Sigma^*\)). SAT is the set of all and only boolean expressions that are satisfiable (cf. [1, par. 10.2.1]).

Definition 1.9 (Language of a dTM). Given a deterministic TM \(M\), the language of \(M\) \(L(M)\) is the set of coded input strings accepted by \(M\) (cf. [1, sect. 8.2]).

Definition 1.10 (Time complexity). A dTM \(M\) is said to be of time complexity \(T(w)\) (or to have running time \(T(w)\)) if whenever \(M\) is given an input \(w\), \(M\) halts after making at most \(T(w)\) moves.

Definition 1.11 (The class \(\mathcal{P}\)). We say a language \(L \in \mathcal{P}\) iff there is some polynomial \(P(n)\) such that \(L = L(M)\) for some deterministic TM of time complexity \(P(n)\) in the length of input (cf. [1, par. 10.1.1]).

2 Issues with definition of complexity for SAT

At first sight, it might be thought that SAT \(\in \mathcal{P}\) is a sufficient condition to affirm that SAT is a tractable problem, in fact, up to now it has been so. However, a previous attempt we made to prove that SAT \(\notin \mathcal{P}\) has pointed out some problems in this regard.

The fundamental trouble concerns the concept of time complexity: by definition \(\text{[11]}\), the running time of an algorithm is a function \(T(w)\) on the input. So, if SAT \(\in \mathcal{P}\), then, by definition \(\text{[11]}\) we would have a SAT solver that runs in polynomial time \(P(|w|)\) in the length of an input \(w\). But despite this, we don’t know anything about relation between the length of \(w\) and the number of its boolean variables.

Since a boolean expression always represents a boolean function, the shortest expression that can represent a boolean function of \(n\) variables could virtually have a length \(l\) that is exponential in \(n\). So, if \(l = 2^n\), the SAT solver can run in
time $P(2^n)$, which is obviously exponential in the number of variables, although it is polynomial in the length of input.

Therefore, it follows that the definition of $P$ is not satisfying enough to ensure that SAT is tractable, so SAT could be exponential even if SAT $\in P$. Later in the paper, we will see that this is how things really are.

For this reason, we have to attack the problem with techniques of boolean function theory.

### 3 Boolean Function Theory

**Definition 3.1** (Boolean domain). $\mathbb{B} \overset{\text{def}}{=} \{0, 1\}$ is the set of truth values.

**Definition 3.2** (Boolean function). Given an $n \in \mathbb{N}$, a boolean function $f$ is a total function $f : \mathbb{B}^n \to \mathbb{B}$, i.e. it assigns to each $n$-tuple of elements of $\mathbb{B}$ an element of $\mathbb{B}$.

**Definition 3.3** (Equivalence of boolean functions). Let $f : \mathbb{B}^n \to \mathbb{B}$ and $g : \mathbb{B}^m \to \mathbb{B}$ be two boolean functions.

- If $n \geq m$:
  
  \[ f \equiv g \text{ iff, for every } \langle v_1, \ldots, v_n \rangle \in \mathbb{B}^n, \ f(v_1, \ldots, v_n) = g(v_1, \ldots, v_m) \text{ holds.} \]

- If $n < m$:
  
  \[ f \equiv g \text{ iff } g \equiv f. \]

If $f \equiv g$, we say that $f$ and $g$ are equivalent. It’s trivial to prove that $\equiv$ is an equivalence relation.

**Definition 3.4** (Interpretation of boolean expressions). Given a boolean expression $E$ that contains $x_{k_1}, \ldots, x_{k_n}$ distinct variables sorted by index, $I_E : \mathbb{B}^{k_n} \to \mathbb{B}$ is a boolean function such that:

\[ I_E(v) \overset{\text{def}}{=} E(T_v) \]

where $v = \langle v_1, \ldots, v_{k_n} \rangle \in \mathbb{B}^{k_n}$ and $T_v : \{x_{k_1}, \ldots, x_{k_n}\} \to \mathbb{B}$ is the truth assignment such that, for every $i \in \{1, \ldots, n\}$, $T_v(x_{k_i}) = v_{k_i}$.

We say that $I_E$ is the interpretation of $E$ as a boolean function.

**Definition 3.5** (Length of a boolean function). Given a boolean function $f : \mathbb{B}^n \to \mathbb{B}$, the length of $f$ is defined as follows:

\[ \text{length}(f) \overset{\text{def}}{=} \min\{|E| \mid I_E \equiv f, \ E \text{ is a boolean expression}\} \]

That is, the length of a boolean function $f$ is the length of the shortest boolean expression $E$, such that its interpretation is equivalent to $f$. 

3
Definition 3.6 (Set of n-ary boolean functions). For every \( n \in \mathbb{N} \), the set of all \( n \)-ary boolean functions is defined as follows:

\[
\text{BoolFunctions}(n) \overset{\text{def}}{=} \{ f : \mathbb{B}^n \rightarrow \mathbb{B} | f^{-1}(\mathbb{B}) = \mathbb{B}^n \}
\]

The condition \( f^{-1}(\mathbb{B}) = \mathbb{B}^n \) stands to mean that \( f \) is a total function, i.e. the inverse image of \( f \) coincides with its domain.

4 Proof

Proposition 1. For every \( n \in \mathbb{N} \):

\[
\text{BoolExp}_\leq(n) \subseteq \bigcup_{k=0}^{n} \Sigma^k
\]

Proof. If \( E \in \text{BoolExp}_\leq(n) \), then, by def. 1.5, \( E \) is a boolean expression such that \( |E| \leq n \), and since \( E \in \Sigma^* \), exists a \( k \in \{0, \ldots, n\} \) such that \( E \in \Sigma^k \); so \( E \in \bigcup_{k=0}^{n} \Sigma^k \).

Lemma 1. If \( P \) is a polynomial, then exists an \( N_P \in \mathbb{N} \) such that:

\[
|\text{BoolExp}_\leq(P(N_P))| < |\text{BoolFunctions}(N_P)|
\]

Proof. Let’s consider the cardinality of \( |\text{BoolExp}_\leq(P(n))| \):

\[
|\text{BoolExp}_\leq(P(n))| \leq \sum_{k=0}^{P(n)} |\Sigma|^k
\]

by prop. 1

\[
= \sum_{k=0}^{P(n)} |\Sigma|^k \text{ since, for each } i \neq j, \Sigma^i \cap \Sigma^j = \emptyset
\]

\[
= \sum_{k=0}^{P(n)} |\Sigma|^{P(n)+1} - 1
\]

since is a geometric series

\[
= \frac{|\Sigma|^{P(n)+1} - 1}{|\Sigma| - 1}
\]

\[
= \frac{8^{P(n)+1} - 1}{8 - 1}
\]

since \( \Sigma \) contains 8 elements

\[
= \frac{8}{7} \cdot 8^P(n) - \frac{1}{7}
\]

\[
= O(8^{P(n)})
\]

So, \( |\text{BoolExp}_\leq(P(n))| = O(8^{P(n)}) \).

Now, let’s consider the set \( \text{BoolFunctions}(n) \). By def. 3.6 \( \text{BoolFunctions}(n) \) is the set of \( n \)-ary boolean functions. Let be \( f \in \text{BoolFunctions}(n) \).

Since \( |\mathbb{B}^n| = 2^n \), then the cardinality of the domain of \( f \) is \( 2^n \), and since, for each \( n \)-tuple of \( \mathbb{B}^n \), \( f \) can assign either 0 or 1, then the number of different ways \( f \) can assign boolean values to the \( n \)-tuples is \( 2^{|\mathbb{B}^n|} = 2^{2^n} \), so \( |\text{BoolFunctions}(n)| = 2^{2^n} \).
Now, let’s compare the growth of cardinalities of $\text{BoolExp}_\leq(P(n))$ and $\text{BoolFunctions}(n)$ using their logarithms to base 2, which are roughly the number of bits necessary to represent the cardinalities.

$$\log |\text{BoolExp}_\leq(P(n))| = O(\log 8^{P(n)}) = O(P(n) \cdot \log 8) = O(P(n))$$

$$\log |\text{BoolFunctions}(n)| = \log 2^{2^n} = 2^n \cdot \log 2 = 2^n$$

This means that the number of bits to represent $|\text{BoolFunctions}(n)|$ grows exponentially in the arity of boolean functions, while the number of bits to represent $|\text{BoolExp}_\leq(P(n))|$ grows polynomially. So, by interpolation, we can always find an $N_P \in \mathbb{N}$ such that:

$$|\text{BoolExp}_\leq(P(N_P))| < |\text{BoolFunctions}(N_P)|$$

**Proposition 2.** If $f: \mathbb{B}^n \rightarrow \mathbb{B}$ and $g: \mathbb{B}^n \rightarrow \mathbb{B}$ are two boolean functions s. t. $f \neq g$, and $F, G$ two boolean expressions s. t. $I_F \equiv f$ and $I_G \equiv g$, then $F \neq G$.

**Proof.** If $F = G$, then $I_F = I_G$, so, by equivalence, $I_F \equiv f$ and $I_G \equiv g$ entail $f \equiv g$, and since $f$ and $g$ have the same arity, by def. three $f = g$ holds, contradicting $f \neq g$. □

**Lemma 2.** If $P$ is a polynomial, then exists a boolean function $\mathcal{F}: \mathbb{B}^{N_P} \rightarrow \mathbb{B}$ such that $P(N_P) < \text{length}(\mathcal{F})$.

**Proof.** By lemma 1, there is an $N_P \in \mathbb{N}$ such that:

$$|\text{BoolExp}_\leq(P(N_P))| < |\text{BoolFunctions}(N_P)|$$

Let be $g, h \in \text{BoolFunctions}(N_P)$ s. t. $g \neq h$, and $G, H$ two boolean expressions s. t. $I_G \equiv g$ and $I_H \equiv h$. Since $g$ and $h$ are both $N_P$-ary functions, by prop. two $G \neq H$.

This entails that we can define a partial function $\mathcal{M}: \text{BoolExp}_\leq(P(N_P)) \rightarrow \text{BoolFunctions}(N_P)$ as follows:

$$\mathcal{M}(E) \overset{\text{def}}{=} f, \text{ such that } I_E \equiv f \in \text{BoolFunctions}(N_P)$$

Now, suppose by contradiction that, for every $f \in \text{BoolFunctions}(N_P)$, there is an $E_f \in \text{BoolExp}_\leq(P(N_P))$ such that $f = \mathcal{M}(E_f)$. Then, $\mathcal{M}$ would be surjective, but this involves that $|\text{BoolExp}_\leq(P(N_P))| \geq |\text{BoolFunctions}(N_P)|$, contradicting (1).

This means that $\mathcal{M}$ cannot be surjective, so, by (2), exists a boolean function $\mathcal{F}: \mathbb{B}^{N_P} \rightarrow \mathbb{B}$ s. t., for every $E \in \text{BoolExp}_\leq(P(N_P))$, $I_E \neq \mathcal{F}$ holds.

By functional completeness of $\{\neg, \land, \lor\}$ and by def. three there must be a boolean expression $F$ such that $I_F \equiv \mathcal{F}$ and $|F| = \text{length}(\mathcal{F})$. Since, for every $E \in \text{BoolExp}_\leq(P(N_P))$, $I_E \neq \mathcal{F}$ holds, then $F \not\in \text{BoolExp}_\leq(P(N_P))$, so, by def. five $P(N_P) < |F|$, thus $|F| = \text{length}(\mathcal{F})$ entails that $P(N_P) < \text{length}(\mathcal{F})$.

Summing up, there is a boolean function $\mathcal{F}: \mathbb{B}^{N_P} \rightarrow \mathbb{B}$ such that $P(N_P) < \text{length}(\mathcal{F})$, which is the thesis of this lemma. □
Lemma 3. Let be \( f: \mathbb{B}^n \to \mathbb{B} \) a boolean function, and \( F \) a boolean expression that contains \( x_{k_1}, \ldots, x_{k_m} \) distinct variables sorted by index, s. t. \( I_F \equiv f \) and \( k_m > n \). Then, exists a boolean expression \( F' \) with its variable indexes less or equal to \( n \), s. t. \( I_{F'} \equiv f \) and \( |F'| < |F| \).

Proof. Since \( I_F \equiv f \), \( F \) contains \( k_m \) variables and \( k_m > n \), then, by def. 3.3, for every \( \langle v_1, \ldots, v_n \rangle \in \mathbb{B}^n \):

\[
f(v_1, \ldots, v_n) = I_F(v_1, \ldots, v_n, \underbrace{0, \ldots, 0}_{k_m - n \text{ times}})
\]

Then, by def. 3.3 \( I_F \equiv f \) entails that, for every \( \langle v_1, \ldots, v_{k_m} \rangle \in \mathbb{B}^{k_m} \):

\[
I_F(v_1, \ldots, v_{k_m}) = I_F(v_1, \ldots, v_n, \underbrace{0, \ldots, 0}_{k_m - n \text{ times}})
\]

This means the truth of \( F \) does not depend by the values of \( x_{n+1}, \ldots, x_{k_m} \), so we can replace them in \( F \) with 0, and then simplify \( F \) until we obtain a new boolean expression, with the rules below:

\[
\begin{array}{cccc}
E & E[-1/0] & E[-0/1] \\
E[A \land 1/A] & E[1 \land A/A] & E[A \lor 1/1] & E[1 \lor A/1] \\
E[A \land 0/0] & E[0 \land A/0] & E[A \lor 0/A] & E[0 \lor A/A]
\end{array}
\]

where \( E[A/B] \) is the result of replacing in \( E \) every occurrence of \( A \) with \( B \). If simplification results in a boolean value, then 1 can be replaced by \( x_1 \lor \neg x_1 \) and 0 by \( x_1 \land \neg x_1 \).

The result of simplifying \( F \) is clearly a boolean expression \( F' \) with its variable indexes less or equal to \( n \), s. t. \( I_{F'} \equiv f \) and \( |F'| < |F| \). \( \square \)

Theorem 1. There is no dTM \( M \), with polynomial time complexity in the number of boolean variables of input, such that \( L(M) = \text{SAT} \).

Proof. Suppose by contradiction that exists a dTM \( M \), with polynomial time complexity \( P(n) \) in the number of boolean variables of input, such that \( L(M) = \text{SAT} \). Then, by lemma 2 there is a boolean function \( f: \mathbb{B}^{N_P} \to \mathbb{B} \) such that \( P(N_P) < \text{length}(f) \).

By functional completeness of \( \{\neg, \land, \lor\} \) and by def. 3.5 there is a boolean expression \( F \) such that \( I_F \equiv f \) and \( |F| = \text{length}(f) \), so \( P(N_P) < \text{length}(f) \) entails that \( P(N_P) < |F| \).

Since, by def. 3.5 \( F \) is the shortest boolean expression such that \( I_F \equiv f \), and since \( f \) is an \( N_P \)-ary function, then, by lemma 3 \( F \) contains at most \( N_P \) variables, and since \( P \) is a polynomial, \( P(N_P) \) is certainly an upper bound of the running time of \( M \) with \( F \) as input.
But, on the other hand, to verify that $F$ is a boolean expression, $M$ is forced to read the entire input, so $M$ must make at least $|F|$ moves, therefore $P(N_P) \geq |F|$, contradicting $P(N_P) < |F|$.

5 Conclusions

Although we didn’t prove that SAT $\notin \mathcal{P}$, anyway we understand an important fact: even if SAT were in $\mathcal{P}$, over a certain $n \in \mathbb{N}$, the quantity of $n$-ary boolean functions that are writable with polynomial length is infimum, if compared to the totality of $n$-ary boolean functions. So, even in that case, a minimal part of problem instances would be tractable, while the overwhelming majority would still be effectively unsolvable in polynomial time (in the number of variables).

References

[1] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Ed. by Pearson. 3rd ed. 2006.