The Cook–Levin Theorem

Édouard Bonnet Codex 5.6 and 6

Abstract

We prove that binary-encoded CNF satisfiability is NP-complete under polynomial many-one reductions. The proof implements a SAT verifier and a reduction from bounded computations through Boolean circuits and gate clauses. It includes correctness, termination, and polynomial running-time bounds for the machines.

The languages are sets of finite binary words. The complexity classes and machine models are those of lax-434930. The annotations connect each definition, statement, and proof below to its formal counterpart.

1 Formulas, encodings, and certificates

Definition 1 (Conjunctive normal form)**.** A literal names a natural-number variable and its sign. A CNF formula is a list of clauses, each a list of literals. Empty clauses are false and the empty conjunction is true. Satisfiability quantifies over Boolean assignments.

Definition 2 (Binary encoding of CNF formulas)**.** Lists use a one-bit continuation marker and a zero-bit terminator. A literal consists of its variable index in unary, a zero terminator, and its sign. This encoding gives every formula a unique binary word and bounds each variable index by the word length.

Lemma 3 (Decoding an encoded formula)**.** *The binary decoder recovers every encoded CNF formula. For every formula F,* $\mathrm{decodeCNF}(\mathrm{encodeCNF}(F)) = \mathrm{some}(F)$.

Proof. Induction proves that parsing an encoded natural number or literal returns the original value and the untouched suffix. A second induction gives the corresponding property for lists. Apply it to clauses and then formulas; the final suffix is empty. □

Definition 4 (The satisfiability language and verifier)**.** SAT contains precisely the encodings of satisfiable CNF formulas. A certificate is a finite list of truth values, extended by false outside its length. The verifier accepts a paired formula encoding and certificate when every clause is satisfied. Malformed encodings are outside the language.

Lemma 5 (Satisfiability and binary encodings)**.** *An encoded formula belongs to the SAT language exactly when the formula is satisfiable. Thus* $\mathrm{encodeCNF}(F) \in \mathrm{SAT}$ *if and only if F is satisfiable.*

Proof. The decoding round trip makes the formula encoding injective. Unfold the SAT language and use injectivity to identify its witnessing formula with the given formula. □

Lemma 6 (A bounded satisfying assignment)**.** *A satisfiable formula has a certificate whose length is at most the length of its binary encoding. Only variables occurring in the formula matter. The bound is* $|y| \leq |\operatorname{encodeCNF}(F)|$.

Proof. Every variable index occurring in a formula is smaller than its encoded length. Restrict a satisfying assignment to that initial segment and extend it by false elsewhere. Every occurring literal keeps its value. Conversely, the extension of any satisfying finite certificate is a satisfying assignment. □

Lemma 7 (Correctness of the SAT verifier)**.** *Membership in SAT is equivalent to the existence of an accepted certificate of length at most the input length. Precisely,* $w \in \mathrm{SAT}$ *if and only if there is a word* y *with* $|y| \leq |w|$ *and* $\operatorname{pair}(w, y) \in \mathrm{Verifier}$.

Proof. For a SAT word, take the bounded assignment certificate for its witnessing formula. Conversely, unpack an accepted pair. Injectivity of the pairing identifies its first component with the input word, and the accepted assignment proves satisfiability. □

Lemma 8 (Polynomial time for CNF verification)**.** *A finite stack machine decodes the paired input and checks the supplied assignment. Its running time is polynomial in the input length, including on malformed encodings, using the machine model of lax-434930. In particular,* $\mathrm{Verifier} \in \mathrm{P}$.

Proof. The finite stack program decodes the paired input, parses its formula, and evaluates each literal under the supplied assignment. Its loop invariants track the unprocessed suffix, the current clause value, and the conjunction of completed clauses. Invalid encodings terminate with rejection. The proved execution bound is $100(n+1)^4$ for an input of length n; the program resets its finite control, clears its work stacks, and returns a singleton Boolean. The checked compiler transfers these executions and their bound to the deterministic stack-machine model of P. □

Lemma 9 (SAT belongs to NP)**.** *A polynomial time verifier checks a satisfying assignment of polynomial length. Hence* $\mathrm{SAT} \in \mathrm{NP}$.

Proof. Use the verifier in P, the polynomial certificate bound $p(n) = n$, and the equivalence established by verifier correctness. □

2 Circuits and gate clauses

Definition 10 (Boolean circuits)**.** A circuit is a finite sequence of input, constant, negation, conjunction, and disjunction gates. Every wire refers to an earlier gate, and the output is a gate of the circuit. A satisfying wire assignment respects every gate and makes the output true.

Definition 11 (Gate clauses)**.** The Tseitin encoding assigns one variable to each gate. At most three clauses enforce a gate's truth table, and a unit clause requires a true output. Input gates have no constraints.

Lemma 12 (Correctness of gate clauses)**.** *The clauses for one gate hold exactly when its output agrees with its Boolean operation.*

Proof. For each gate constructor, enumerate the Boolean values of its input and output wires. In every case, the conjunction of its clauses is exactly the equality required by the gate operation. □

Lemma 13 (Correctness of the circuit encoding)**.** *The gate clauses and output clause are satisfiable exactly when the circuit is satisfiable.*

Proof. Evaluation of a concatenation is conjunction, and evaluation of the flattened gate-clause list is conjunction over the gates. Apply the gate-clause identity and the output unit clause, then quantify over the common wire assignment. □

3 Polynomial reductions and NP-completeness

Definition 14 (Polynomial many-one reductions and NP-completeness)**.** A polynomial many-one reduction is one polynomial time computable function on binary words that preserves membership. A language is NP-complete if it belongs to NP and every language in NP reduces to it.

Lemma 15 (Polynomial circuit simulation of a verifier)**.** *For a fixed polynomial time verifier and polynomial certificate bound, construct a circuit whose free inputs represent the certificate. The circuit is satisfiable exactly when some bounded certificate is accepted. Its encoded gate clauses are produced in polynomial time. The machine simulation and time bound follow from a finite stack program that emits the initial layer, the transition layers, and the final acceptance constraint. For $V \in \mathrm{P}$ and $p \in \mathbb{N}[X]$, there is a family C_x such that*

$$C_x \text{ is satisfiable} \iff \exists y,\ |y| \le p(|x|) \text{ and } \mathrm{pair}(x, y) \in V.$$

The function $x \mapsto \mathrm{encodeCNF}(\mathrm{Tseitin}(C_x))$ is polynomial-time computable.

Proof. Choose an equivalent finite single-tape machine for the verifier. The certificate bound and the encoded pair length give a polynomial bound on the simulated run. Unroll the initial tape and successive configurations into an ordered circuit: free inputs describe the bounded certificate, local gates compute each transition, and the final output tests acceptance. The checked circuit invariants identify wire values with machine configurations and prove the required satisfiability equivalence. A finite stack program emits the initial layer, transition layers, gate clauses, and output constraint. Bounds on wire addresses, the number of gates, unary indices, and the emitting loops yield a polynomial running time for the exact binary output encoding. □

Lemma 16 (NP-hardness of SAT)**.** *Every language in NP has a polynomial many-one reduction to the binary language of satisfiable CNF formulas.*

Proof. For a language in NP, choose its verifier and certificate polynomial. Map each input to the binary encoding of the compiled circuit's gate clauses. The compiler makes this a polynomial-time function. Circuit correctness, correctness of the gate encoding, and the SAT encoding equivalence show that this function preserves membership. □

Theorem 17 (The Cook–Levin theorem)**.** *Satisfiability of CNF formulas is NP-complete under polynomial many-one reductions.*

Proof. Combine membership of SAT in NP with its polynomial many-one hardness for every language in NP. □