Chapter 8

Nondeterministic Polynomial Time

8.1. Satisfiability of Propositional Formulas

In this section we present a new proof that SAT is $\mathcal{NP}$-complete. Given the machinery that we have developed, this is much simpler than the one originally given by Cook [4].

Definition 8.1: A circuit is in *normal form* if and only if the NOT gates have as inputs only circuit inputs.

Exercise 8.2: Show that any circuit can be put into normal form with an increase of 1 in depth and doubling in size.

Definition 8.3: $CSAT = \{C \mid C \text{ is a circuit in normal form with } L(C) \neq \emptyset\}$.

Theorem 8.4: $CSAT$ is $\leq_m^{\mathcal{L}}$–complete for $\mathcal{NP}$.

Proof:

1. $CSAT \in \mathcal{NP}$: Given a circuit C with k inputs, guess input values $(b_1, b_2, \ldots, b_k) \in \{0,1\}^k$ and accept if and only if C outputs 1 on input $(b_1, b_2, \ldots, b_k)$. The circuit evaluation is done in polynomial time as in Theorem 7.11.

2. $CSAT$ is $\leq_m^{\mathcal{L}}$–hard for $\mathcal{NP}$: Let N be a nondeterministic Turing machine that accepts a language L in polynomial time $p(n)$. Assume without loss of generality that each configuration of N has at most two immediate successors. N can be simulated by a nondeterministic Turing machine N' that behaves as follows. On input x, N' computes $p(n)$ (which can be done deterministically), existentially chooses $y \in \{0,1\}^{p(n)}$, and writes $\#y$ after x on the input tape. N' then returns its head to the left end of x and simulates some deterministic Turing machine D on input $x\#y$, where D simulates N, but consumes another bit of y whenever it needs to simulate a nondeterministic choice. By Theorem 4.9, there is an alternating Turing machine A that uses space $O(\log n)$ and accepts the same language as D.

CONSTRUCTION: We must construct a deterministic Turing machine M that, on input x, outputs some circuit C that is in $CSAT$ if and only if there exists a y such that A accepts $x\#y$. M first computes $p(n)$, and then proceeds exactly as in Theorems 7.11 and 7.17, combining the input conventions of those two proofs as follows:

- If P is a configuration in state q_{read} with ia on its index tape, then:
  - if $i \leq |x|+1$, then u_P is a constant gate with value
    - 1, if $a = (x\#)_i$, and
    - 0, otherwise;
  - if $i > |x|+1$, then u_P is
    - an input vertex labeled $y_{i-|x|-1}$, if $a = 1$,
    - the negation $\neg y_{i-|x|-1}$ of an input vertex, if $a = 0$, and
    - the constant 0, if $a \notin \{0,1\}$.

CORRECTNESS:

N accepts x	if and only if	there exists a y of length $p(n)$ such that A accepts $x\#y$
	if and only if	there exists a y of length $p(n)$ such that C outputs 1 on input y
	if and only if	$C \in CSAT$.

The second equivalence is proved in the correctness proof of Theorem 7.11.

ANALYSIS: M can compute the binary representation of $p(n)$ in space $O(\log n)$. Once this is done, the remainder of the analysis is the same as in Theorem 7.11. □

Given Theorem 8.4, it is reasonably straightforward to prove that SAT is complete for $\mathcal{NP}$. In fact, Cook [4] considered the following much more restrictive version of SAT:

Definition 8.5: A propositional formula is in *conjunctive normal form* if and only if it is the conjunction of "clauses", each of which is the disjunction of "literals", each of which is either a propositional variable or its negation.

Definition 8.6: *3SAT* is the set of satisfiable propositional formulas in conjunctive normal form with at most three literals per clause.

Theorem 8.7 (Cook [4]): *3SAT* is $\leq_m^{\mathcal{L}}$–complete for $\mathcal{NP}$.

Proof:

1. *3SAT* $\in \mathcal{NP}$: Given a formula F with k variables, nondeterministically choose a truth assignment $A \in \{0,1\}^k$ and accept if and only if A satisfies F. This formula evaluation can be done deterministically in polynomial time (in fact, in ATIME($\log n$) [2]).

2. $CSAT \leq_m^{\mathcal{L}} 3SAT$:

CONSTRUCTION: Given a circuit C, the reduction outputs a formula F with one variable x_w for each vertex w of C. F is the conjunction of the following:

- For each OR gate w with inputs u and v, output the clauses

$$(\neg x_u \vee x_w) \wedge (\neg x_v \vee x_w) \wedge (\neg x_w \vee x_u \vee x_v).$$

- For each AND gate w with inputs u and v, output the clauses

$$(\neg x_u \vee \neg x_v \vee x_w) \wedge (\neg x_w \vee x_u) \wedge (\neg x_w \vee x_v).$$

- For each NOT gate w with input u, output the clauses

$$(x_u \vee x_w) \wedge (\neg x_u \vee \neg x_w).$$

- If w is the output gate, output the singleton clause (x_w).

CORRECTNESS: We must prove that $C \in CSAT$ if and only if $F \in 3SAT$.

"only if" clause: Suppose C outputs 1 on input $(b_1, b_2, \ldots b_k)$. Then F has a satisfying assignment $A : \{x_w\} \to \{0,1\}$, where $A(x_w) = \text{val}(w)$.

"if" clause: Suppose the truth assignment $A : \{x_w\} \to \{0,1\}$ satisfies F. Then $\text{val}(w) = A(x_w)$ is consistent and has $\text{val}(w) = 1$ for the output gate w.

ANALYSIS: Space $O(\log n)$ suffices to hold u, v, and w. □

For a detailed discussion of completeness for $\mathcal{NP}$ and a comprehensive list of the diverse problems that are complete for $\mathcal{NP}$, see the book by Garey and Johnson [10].

8.2. Exercises

1. Do Exercise 8.2.

   (Hint: Because a single gate may be the input to more than one other gate, it is insufficient to apply deMorgan's laws in any naive way.)