

CMPT 710 - Complexity Theory: Week 4

Valentine Kabanets

September 25, 2007

1 NP Completeness of SAT

Next we consider the following version of Circuit-SAT, and prove its NP-completeness.

$$SAT = \{\phi \mid \phi \text{ is a satisfiable Boolean formula}\}$$

Theorem 1 (Cook-Levin). *SAT is NP-complete.*

Proof. SAT is in NP (easy). To prove NP-hardness, we will show that Circuit-SAT is reducible to SAT.

Let C be an arbitrary Boolean circuit with gates g_1, \dots, g_m , where g_1, \dots, g_n are input gates and g_m is the output gate. For each g_j , introduce a Boolean variable y_j . For every $i > n$, define the Boolean formula $gate_i$ expressing that the value of y_i is equal to the value of the gate g_i . That is, if gate g_i is an AND gate with inputs g_{i_1} and g_{i_2} , then $gate_i$ is True iff $y_i \equiv y_{i_1} \wedge y_{i_2}$; similarly, for OR, and NOT gates.

Our final formula ϕ_C is defined as

$$\bigwedge_{i=n+1}^m gate_i \wedge "y_m \equiv 1"$$

It is left as an exercise to verify that C is satisfiable iff ϕ_C is satisfiable. □

2 Importance of the Cook-Levin Theorem

There is a trivial NP-complete language:

$$L_u = \{(M, x, 1^k) \mid \text{NTM } M \text{ accepts } x \text{ in } \leq k \text{ steps}\}$$

Exercise: Show that L_u is NP-complete.

The language L_u is not particularly interesting, whereas SAT is extremely interesting since it's a well-known and well-studied natural problem in logic. After Cook and Levin showed NP-completeness of SAT, literally hundreds of other important and natural problems were also shown to be NP-complete. It is this abundance of natural complete problems which makes the notion of NP-completeness so important, and the "P vs. NP" question so fundamental.

3 Viewing NP as a game

Nondeterministic computation can be viewed as a two-person game. The players are Prover and Verifier. Both get the same input, e.g., a propositional formula ϕ . Prover is all-powerful (but not trust-worthy). He is trying to convince Verifier that the input is in the language (e.g., that ϕ is satisfiable). Prover sends his argument (as a binary string) to Verifier. Verifier is computationally bounded algorithm. In case of NP, Verifier is a deterministic polytime algorithm.

It is not hard to argue that the class NP of languages L is exactly the class of languages for which there is a pair (Prover, Verifier) with the property: For inputs in the language, Prover convinces Verifier to accept; for inputs not in the language, any string sent by Prover will be rejected by Verifier. Moreover, the string that Prover needs to send is of length polynomial in the size of the input.

Once we have this set-up, we can vary computational resources allowed for Verifier to use. For instance, we can let Verifier be a randomized polytime algorithm, thus allowing some small probability that Verifier is incorrectly convinced the input outside the language. We can also allow Verifier random access to the string sent by Prover, i.e., Verifier can choose a few (say, 3) locations in the string and look only at the corresponding 3 bits. What class of languages can we capture with this kind of randomized protocols where Verifier reads only 3 bits of Prover's string? It turns out all of NP! This is basically the content of the famous PCP Theorem (where "PCP" stands for "Probabilistically Checkable Proofs"). That is, for every language in NP, there is a pair (Prover, Verifier) such that, for inputs in the language Prover sends a polynomial-length string that is accepted by Verifier with probability 1, while for inputs not in the language any such string is rejected by Verifier with constant probability (say, 1/2). Moreover, Verifier always reads (randomly selected) constantly many (say, 3) bits of Prover's string.

We will talk more about this alternative characterization of NP and its applications later in the course.