1 NP Problems

Sat is the first of hundreds of combinatorial problems shown to be solvable in *Nondeterministic Polynomial time* (NP). Recall that for us Sat is the subset of Formulas that are satisfiable, symbolically:

$$\textbf{Sat} = \{X : Form | \exists a : Var(X) \to \mathbb{B}.Bval(X, a) = t \ in \ \mathbb{B}\}.$$

Recall also that Bval is the recursive function we defined in Lecture 7 which takes X as its first argument. We will use the convention that if we supply X and write Bval(X), the result is a function from $Var(X) \to \mathbb{B}$ into $\mathbb{B}$. This is the Boolean valuation that we use in practice. Our definition in Lecture 7 is:

Bval(X,v) = **case** X is var(y) $\to$ v(y); neg(U) $\to$ bnot(Bval(U,v));
op(U,V) $\to$ bop(Bval(U,v),Bval(V,v)) **end**

Consider a *problem* to be a set S of elements from a discrete type U. S belongs to the class of NP problems if and only if there is a polynomial time algorithm R on U and on another discrete set T and a polynomial p such that

$$x \text{ in S iff } \exists t : T.(|t| \leq p(|x|) \& R(x, t))$$

and the running time of R(x, t) is bounded by $c \times (|t| + |x|)^d$ where $|x|$ and $|t|$ are the length of x and t (think of x and t as strings of symbols), and c and d are positive integers. We call t a *certificate* for x.

To see that **Sat** belongs to NP, let U be the set of propositional formulas, Form.

$$\text{X in } \textbf{Sat} \text{ iff } \exists a : Var(X) \ \to \mathbb{B}.Bval(X, a) = t \ in \ \mathbb{B}.$$

So the *certificate* is the assignment a. We have analyzed the recursive computation of *Bval(X,a)* in class and observed that the number of steps taken is a constant times the length of the formula X, so $|X|$ is len(X). The *assignment* (or interpretation) a is bounded by twice the number of variables in X.

The problem often referenced in the literature as SAT is the **Sat** problem

for formulas in Conjunctive Normal Form, CNF. Sometimes this is called CNF-**Sat**.

2 NP Complete Problems

Some problems in NP are special because they are the hardest problems in the class in the sense that if they can be solved by a polynomial time algorithm, then so can any other NP problem. To make this idea precise, we say that a problem S_1 is reducible in P-time to a problem S_0, symbolically $S_1 \leq_P S_0$, iff there is an algorithm f_{S_1} solving membership in S_1 which runs in P-time using a membership algorithm for S_0, call it f_{S_0} at essentially "no cost", that is when we count the number of steps used in the computation, we count the cost of any application of f_{S_0} as *one step*. The cost of producing the input to this function is part of the cost of the computation, but the cost of producing the Boolean output is one step. So we think of f_{S_0} as an **oracle** for S_0 or a "black box" that produces the answer about whether a value Z belongs to S_0 in one step, and we don't necessarily see the computation of $f_{S_0}(z)$ for any z. Here are some basic facts we proved in the lecture and visited again in Lecture 10.

1. $Sat \leq_P SAT$, that is, the general satisfiability problem is reducible to CNF-Sat. In the lecture notes from 2009, we explained this reduction in terms of non-deterministic Turing machines. You can read our account in CS4860 Spring 2009, Efficient SAT Solving, Lecture 4.

2. $SAT \leq_P 3SAT$ where 3SAT are the CNF formulas with exactly three literals per clause. Recall that a literal is a variable or its negation, say p_i or $\ p_i$; we also write $\ p_i$ as $\bar{p}_i$.

3. $SAT \leq_P Clique$ where clique is the question of whether a graph G has a clique of size k, which is a set of k vertices that are all connected by edges. We sketched this reduction in lecture, and it is found in most textbooks that treat the topic of NP completeness.

4. $SAT \leq_P k-colorablegraph$ where k-colorable means that the vertices can be assigned k colors such that no two adjacent vertices have the same color. We looked at this problem in studying compactness and discussed the 3-coloring of planar graphs.

3 Exercise

Translate the following expressions into English.

Let $\mathbb{P}$ equal ***Prop***, the type of propositions. Let $P : Form \to \mathbb{P}$ be the propositional functions on Formulas. Recall that SAT(S) for an infinite set of formulas S means that they are all simultaneously satisfiable using an assignment (valuation) of Var.

1. $\forall X : Form.\ \forall v : Var(X) \to \mathbb{B}.\ Boolean(Bval(X), SubForm(X))$.

2. $\forall X : Form.\ (\ TAUT(X) \Rightarrow\ \exists T : Tableau(FX).$
   $(Completed(T)\ \&\ Closed(T))\ )$.

3. $\forall e : \mathbb{N}\ \to Form.\ \forall n : \mathbb{N}.\ SAT(\{e(1), ..., e(n)\})\ \Rightarrow\ \exists v : Var \to \mathbb{B}.$
   $\forall i : \mathbb{N}.\ bval(e(i), v) = t\ in\ \mathbb{B}$.

4. $\forall P : Form \to \mathbb{P}.\ (\ \forall X : Form.$
   $(\ (\forall Y : SubForm(X).\ P(Y))\ )\ \Rightarrow P(X)\ )\ \Rightarrow\ \forall X : Form.\ P(X)$.

5. $\forall X : Form.\ PROV(X) \Leftrightarrow (\ \exists T : Tableau(FX).Closed(T))$.

6. $\exists Decide : Form \to \mathbf{B}.\ \forall X : Form.(Decide(X) = t\ in\ \mathbb{B})\ \Leftrightarrow\ PROV(X)$.