CSC2401F Introduction To Computational Complexity
Lecture 10 / Tuesday, October 16, 2001
Instructor: Allan Borodin
Scribe: Peter Kanareitsev

1 Context and plan of this lecture

In the last lecture, we proved that the *Circuit-SAT* problem is NP-complete. Note that the proof we used shows that the Circuit Value Problem (CVP) is hard for P, and indeed P-complete (with respect to $\leq_{LOGSPACE}$ reducibility).

We will also exhibit a transformation that reduces *Circuit-SAT* to 3-*SAT* (i.e. to 3-CNF satisfiability) and briefly look at another transformation that reduces 3-*SAT* to a graph theory problem (Vertex Cover).

In the upcoming lectures, we will look at reducibility transformations for other natural problems and, hopefully, get into the current and interesting problems of search, optimization and approximation. We will also define the notion of weak vs. strong NP-hardness, something that the textbook does not mention although it does mention the closely related concept of pseudo polynomial time.

2 CVP is P-complete

2.1 Review of transformation to *Circuit-SAT*

Recall that in the last lecture, we looked at the language

$$\begin{array}{rcll} \textit{Circuit-SAT} & = & \{\hat{C} \mid & \hat{C} \text{ is an encoding of a Boolean circuit } C \\ & & & \text{having a number of inputs } y_1, \cdots, y_m \text{ and} \\ & & & \text{there are input settings } a_1, \cdots, a_m \in \{0,1\} \\ & & & \text{such that } C \text{ accepts } \forall i y_i = a_i\} \end{array}$$

Recall also how we reduced an arbitrary language in NP to *Circuit-SAT*. We used the alternative definition of NP as the class of languages of the form

$$L = \{x \mid \exists y \cdot |y| \leq q(|x|) \text{ and } R(x,y) = true\}$$

for some polynomial q, where y is a certificate (describing the guessed path in a NTM's computation tree) and $R(x,y)$ is a verification predicate, computable in polynomial time on some *deterministic* TM M (say having 1 tape).

We then noticed that for any fixed x, the computation of a Turing macnihe M which decides $\langle x,y\rangle$ can be described by a big Boolean circuit, thanks to the "localized" property of the Turing Machine model of computation (i.e. the next configuration is always fully determined by the current state and the 3 tape symbols at the squares near the head). Furthermore, we found that the size of the big circuit was uniformly bounded by a polynomial in $|x|$. Refer to last lecture for a diagram of this circuit.

2.2 Transformation of generic $L \in P$ to CVP

We now modify this argument to show that any $L \in P$ can be reduced to

$$CVP \;=\; \{\langle \hat{C}, w\rangle \mid w \text{ is accepted by circuit } C \text{ encoded by } \hat{C}\}$$

Let $\tilde{M}$ be a DTM recognizing L (i.e. $L = \mathcal{L}(\tilde{M})$) in polynomial time. We can transform a pair $\tilde{M}, x$ into $\langle \hat{C}, x\rangle$ (a pairing of a circuit and its input) so that x is accepted by $\tilde{M}$ if and only if $\langle \hat{c}, x\rangle \in CVP$. The circuit is created the same way as for $Circuit\text{-}SAT$ except that the deterministic case is simpler because there is no certificate y appearing as input. We insist this transformation can be done in $LOGSPACE$.

This shows that $\forall L \in P \;\; L \leq_{LOGSPACE} CVP$; that is to say, CVP is hard for P with respect to $\leq_{LOGSPACE}$. Clearly, $CVP \in P$, so we can say that CVP is complete for P with respect to $\leq_{LOGSPACE}$, or P-complete.

2.3 Relating time and space classes

The foregoing means $P \subseteq LOGSPACE$ if and only if $CVP \in LOGSPACE$. Also, $P \subseteq POLYLOGSPACE$ if and only if $CVP \in DSPACE((\log n)^k)$ for some k (where $POLYLOGSPACE = \cup_k DSPACE((\log n)^k)$). Later in the course, we will show that $POLYLOGSPACE$ is the same as $POLYLOG$ "parallel time", a concept not yet defined. So far we have only considered sequential machines, rather than a varying number of machines working simultaneously. To do something in $POLYLOG$ "parallel time" will turn out to be equivalent to being able to do it in $POLYLOG$ space.

3 3-SAT is NP-complete

3.1 Proof approach

We now return to the subject of NP-completeness and show a transformation that reduces $Circuit\text{-}SAT$ (already shown NP-complete) to 3-SAT. This approach to proving NP-completeness of 3-SAT is simpler than (Cook's) direct proof, because the transformation of a generic language to $Circuit\text{-}SAT$ possesses a certain natural simplicity.

▶ From now on, we will adopt a notation convention that $\leq$ means $\leq_{LOGSPACE}$, unless we explicitly state otherwise.

We will show that $Circuit\text{-}SAT \leq$ 3-SAT.

3.2 Definition of 3-SAT

First, we define 3-SAT. Let a literal be a propositional variable or its negation (denoted $x, \bar{x}$). Let a clause be a disjunction of literals. Then a CNF is a conjunction of clauses. An k-CNF is a CNF with exactly (by some accounts, at most) k literals per clause. For example, the following is a 3-CNF:

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge \ldots \wedge (x_2 \vee x_5 \vee x_9)$$

Note that every Boolean function on n variables can be represented by an n-CNF formula (CNF stands for Conjunctive Normal Form).

Let 3-SAT $=$ $\{F \mid F$ is a satisfiable 3-CNF formula$\}$ where satisfiable means there is a truth assignment to the propositional variables which makes the formula evaluate to true.

3.3 Transformation of $Circuit$-SAT to 3-SAT

Now, we look at $Circuit$-SAT. The Boolean circuit C has a number of inputs $y_1, \ldots, y_m$ and has a single output gate (although internal gates can have any fanout). We can view C as a directed acyclic graph. For such a graph, there is a topological ordering of its vertices, so we can number all gates in C in such a way that later gates have higher numbers. Note that there may be some freedom in the choice of this ordering, as unless there is directed path between two gates, there is no implied order between them.

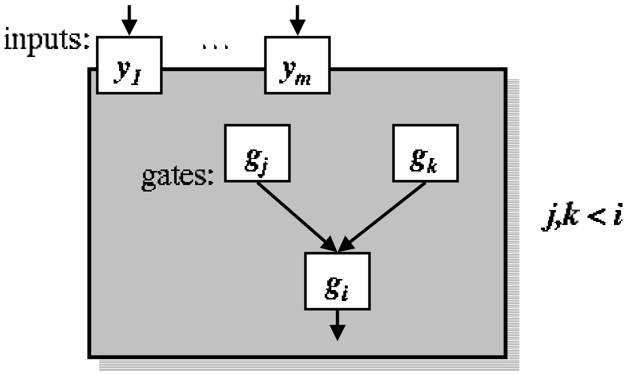

Figure 1: Topological ordering of gates in a Boolean circuit.

We will now transform the circuit C to a 3-CNF formula by introducing propositional variables to describe the output signal of each gate. Let v_i describe the output signal of the i-th numbered gate g_i and let z_i describe the signal at the i-th input y_i. Then circuit C accepts some input settings if and only if the set of propositional formulas describing signal manipulation at each gate, together with the formula v_{output}, is satisfiable.

But this set has a 3-CNF, obtained by conjoining together the 3-CNF's describing the signal constraints at each gate. Since any gate g_i in a Boolean circuit has 2 inputs (the outputs of some gates g_j, g_k, with $j, k < i$), the signal constraint at gate g_i has the form $v_i \leftrightarrow (v_j\, OP_i\, v_k)$ where OP_i is the boolean operation performed by g_i. Since only 3 gates (variables) are involved, this always has a 3-CNF.

For example, if g_i is an OR-gate, we have

$$v_i \leftrightarrow (v_j \vee v_k)$$

which is equivalent to

$$(v_i \rightarrow (v_j \vee v_k)) \wedge ((v_j \vee v_k) \rightarrow v_i)$$

which can be simplified to 3-CNF:

$$(\bar{v}_i \vee v_j \vee v_k) \wedge (\bar{v}_j \vee v_i) \wedge (\bar{v}_k \vee v_i)$$

To summarize, we have reduced satisfiability of circuit C to satisfiability of the following 3-CNF:

$$v_{output} \wedge \bigwedge_{1 \leq i \leq \sharp gates} (\text{3-CNF expansion of } v_i \leftrightarrow \cdots)$$

3.4 The 3 in 3-SAT and other curiosities

What is so special about 3 in 3-SAT? Well, 4-SAT is also NP-complete, since 3-SAT $\leq$ 4-SAT, as evidenced by the transformation

$$(x \vee y \vee z) \mapsto (x \vee y \vee z \vee w) \wedge (x \vee y \vee z \vee \bar{w})$$

where w is a new variable. Also, for $k > 3$, the reducibility works in both directions: k-SAT $\leq$ 3-SAT (k-CNF can be viewed as a Boolean circuit and we have just shown how to transform it to 3-CNF). However, 2-SAT is not NP-hard. 2-$SAT \in P$ and furthermore, 2-$SAT \in NL$, as we will show in the next lecture by viewing 2-SAT as a connectivity problem on a graph of literals.

2 vs. 3 phenomena are pervasive in Complexity theory. Quite often, natural problems with a natural-number parameter are easy when the parameter is 2 but NP-complete when it is 3. For example: 2-SAT vs. 3-SAT, 2-colouring vs. 3-colouring, 2-PARTITION vs 3-PARTITION, 2-dimensional matching vs 3-dimendsional matching.

4 Other natural problems and their transformations

4.1 Overview

By now, we have shown that $L \leq$ *Circuit*-SAT $\leq$ 3-SAT (where $L \in NP$ is a generic language). This puts us in position to show that many other natural problems are hard for NP. The following reducibilities are described in the textbook:

3-SAT $\leq$ Vertex Cover $\leq$ Independent Set $\leq$ Clique
3-SAT $\leq$ Hamiltonian Circuit $\leq$ Traveling Salesman
3-SAT $\leq$ Subset Sum

In the next lecture, we will see that Subset Sum (SS) is "weakly" NP-hard, while Traveling Salesman (TSP) is "strongly" NP-hard.

Corman, Leiserson and Rivest (CLR) gives a somewhat different chain of reductions:

$$(\text{generic } L \in NP) \leq \textit{Circuit-SAT} \leq 3\text{-}SAT \leq \text{Clique} \leq \text{Vertex Cover} \leq \text{Subset Sum}.$$

4.2 3-$SAT \leq VC$

What is Vertex Cover (VC)? Let $G = (V, E)$ be an undirected graph. A subset V' of V is a vertex cover for G if every edge $(u, v) \in E$ has at least one endpoint (u or v) in V'. The Vertex Cover decision problem is defined as follows:

$$VC = \{(G, K) \mid G \text{ has a vertex cover of size } \leq K\}$$

To show that VC is NP-complete, we would like take an arbitrary 3-CNF formula F and transform it: $F \mapsto (G_F, n + 2m)$ so that F is satisfiable if and only if G_F has a vertex cover of size $n + 2m$ (where n is the number of propositional atoms, m is the number of clauses in F and the vertices of G_F are the literals of F). This transformation is depicted in figure 2.1 on p. 52 of the textbook. Although it is easier than the transformation to Hamiltonian Circuit, it is still not obvious, and is worth a look.

In the upcoming lectures, we will work with other natural problems, reducing them to each other. We should expect that each time we go to a new problem domain (for example, from graph theory to number theory), the transformation will entail new and interesting insights.

5 A concluding remark

It is amazing, how we sometimes adopt such firm views that it locks us into a position, until contrary results start exploding all around us. One specific result of this sort is the 1988 result on $\overline{NL}$. Our firm view at the moment is that $NP \neq \overline{NP}$ and of course the weaker assertion $P \neq \overline{NP}$.