Reductions

CS254

Chris Pollett

Oct 16, 2006.

Outline

- Reductions
- Completeness
- Closure Under Reductions

Introductions to Reductions

- The class NP contains an infinite number of languages.
- We have already mentioned that TSP(D) is in NP. Similarly, SAT is in NP, and so is REACHABILITY.
- How do we compare how hard one problem X is versus another problem Y?
- One way would be to show instances of X map efficiently to instances of Y. Then we could say, if we can do Y efficiently, we can do X efficiently. Or another way, Y is at least as hard as X.
- This is the idea behind the notion of reduction.

Definition

We say L_1 ***is reducible*** to language L_2 with respect to reduction computable in the functions class FC, if there is some function f in FC such that for all strings x,

$x \in L_1$ ***iff*** $f(x) \in L_2$.

- For example, we showed in the practice midterm, every r.e. language is reducible to the halting problem via a computable function.
- For this class, we will mainly be interested in the case where FC is either the deterministic logspace computable functions or the deterministic polynomial time computable functions. Both of these classes are closed under composition (We'll see for logspace in amoment). So logspace or p-time is what we'll mean by efficient.

Proposition

If R is a logspace reduction computed by a Turing Machine M, then for all inputs x, M halts after a polynomial number of steps.

Proof. There are $O(nc^{\log n}) = O(n^{1+\log c})$ possible configurations for M on input x, where $n=|x|$. Since M is deterministic and doesn’t run forever, no configuration can repeat. So the machine must halt within time $O(n^{1+\log c})$.

Examples

- Hamiltonian Path is the problem given the adjacency matrix I for a graph G=(V,E) written as a string is there a path in this graph which visits each node exactly once?
- One can show this problem reduces to SAT. Given I we compute an instance of SAT as follows
  1. We use variables x_{ij} to represent that j is the ith vertex used in the Hamiltonian path.
  2. For each $1 \leq j \leq n = |V|$, we have a clause $(x_{1j} \vee \ldots \vee x_{nj})$ (excluding x_{jj}) to say each vertex must appear in the path.
  3. We have clauses $(\neg x_{ij} \vee \neg x_{ik})$ to say that node i cannot be two different vertices.
  4. Finally, for each pair (i,j) which is not in G and for k =1, .. n-1 we have the clause $(\neg x_{ki} \vee \neg x_{k+1j})$ to say that i is not followed in the path by j.
- Notice (2) and (3) can be computed only knowing the number of vertices and given the adjacency graph we only need to remember the row i and the column j (logarithmic amount of info) to determine if we should spit out a given clause or not.

Composition of Reductions

Prop. Assume FC is close under composition. If S is a reduction in FC of L_1 to L_2 and R is a reduction in FC of L_2 to L_3. Then R°S is a reduction in FC of L_1 to L_3.

Proof. Notice $x \in L_1$ iff $S(x) \in L_2$ iff $R(S(x)) \in L_3$. Q.E.D.

- To see FL (the class of functions in logspace) is closed under composition, we have to be careful, since the output tape of S is write-only and typically the output takes polynomial many squares to store. Instead, we simulate the machine for R by keeping on a counter which tape square i it is reading from the output of S(x). To figure out the value of this tape square we start simulating S(x), ignoring any writes it does, but keeping track of the number of times it was about to write. When S it is about to do it's ith write, we jump back to our simulation of R and use this value.

Completeness

Defn. Let C be a complexity class, and let L be a language in C. We say that L is C-complete with respect to reductions in FC´ if any language L´ in C can be reduced to L via a reduction computable in FC´.

Example. The Halting Problem is complete for the r.e. languages with respect to Turing computable reductions.

- One can view complete problems as the hardest problems in a given complexity class.

Closure under Reductions

- We say a complexity class C is ***closed under reductions*** if whenever L is in C and L´ is reducible to L then L´ is in C.
- It is not too hard to show P, NP, coNP, L, NL, PSPACE, and EXP are all closed under logspace computable reductions.
- TIME(n) is not closed under logspace reductions. Therefore, TIME(n) cannot equal any of these classes.

Prop. If two classes C and C´ are both closed under reductions then if there is a complete language L in C which is also in C´ then $C \subseteq C´$.