CSE547T Class 19

Jeremy Buhler

April 3, 2017

1 Polynomial Time Reduction

Reduction arguments are also useful for complexity theory!

- Let L_1, L_2 be recursive languages.
- We say that L_1 *polynomially reduces to* L_2, denoted $L_1 \leq_p L_2$, if there exists a function $f : \Sigma^* \to \Sigma^*$ such that
  1. $x \in L_1$ iff $f(x) \in L_2$.
  2. f is computable by a TM in time polynomial in $|x|$.
- The transducer function f turns instances of the membership problem for L_1 into instances for L_2, without doing too much work.
- **Lemma**: if $L_2 \in P$ and $L_1 \leq_p L_2$, then $L_1 \in P$.
- **Pf**: Let M be a TM to decide L_2 in polynomial time.
- Construct TM N to decide L_1 as follows.
- On input x, N computes $f(x)$, then passes it to M and does whatever M does.
- M runs in time $O(\text{poly}(|f(x)|)) = O(\text{poly}(|x|))$, and f is also computable in time $O(\text{poly}(|x|))$.
- Hence, N decides L_1 in polynomial time. QED
- Consider the contrapositive statement too:
- If $L_1 \notin P$, and $L_1 \leq_p L_2$, then $L_2 \notin P$.

How about an example?

- Consider problem CLIQUE(G,k): given an undirected graph G and integer k, does G contain a complete subgraph on at least k vertices?
- (Language for this problem is set of all (G, k) such that G contains a complete subgraph on at least k vertices.)
- A related problem: ISET(G, k): given an undirected graph G and integer k, does G contain an *independent set* on at least k vertices?

- (An independent set is a subgraph in which *no* two vertices have an edge between them.)

- **Lemma**: ISET $\leq_p$ CLIQUE.
- **Pf**: Let (G, k) be an input to ISET. Define f as follows.
- Let G' be a graph with the same vertex set as G, such that for each pair of vertices u, v, edge (u, v) is in G' iff it is *not* in G.
- Set $f(\langle G, k\rangle) = \langle G', k\rangle$.
- Now f runs in worst-case time $\Theta(|G|^2)$, so it is clearly polynomial in its input size.
- **Claim 1**: If $(G, k) \in$ ISET, then $(G', k) \in$ CLIQUE.
- **Pf**: If G contains an independent set of size k, then no pair of vertices in this subgraph is joined by an edge.
- By construction of G', all pairs in the set will be joined by edges in G', yielding a k-clique. QED
- **Claim 2**: If $(G', k) \in$ CLIQUE, then $(G, k) \in$ ISET.
- **Pf**: if G' contains a clique of size k, then every pair of vertices in this subgraph is joined by an edge.
- By construction of G', no pair in the set was joined by an edge in G, so the subgraph forms an iset in G. QED

2 NP-Completeness

We will use the contrapositive form of polytime reduction in another way that leverages the difficulty of determining whether P = NP.

- **Defn**: a language L is said to be *NP-complete* if
  1. $L \in NP$.
  2. For every $L' \in NP$, $L' \leq_p L$.
- Can read second condition as "L is as hard to decide as *any* language in NP."
- Hence, if L satisfies only condition 2, we say that L is *NP-hard*.
- (Condition 1 is also necessary – an NP-hard language need not be in NP!)

What can we say about NP-complete languages?

- **Fact 1**: let L be a language, and let L' be an NP-complete language.
- If $L' \leq_p L$, then L is NP-hard.
- If we also know that $L \in NP$, then L is NP-complete.
- (Proof follows by transitivity of polytime reduction.)
- **Fact 2**: if any NP-complete language is in P, then P = NP.
- **Pf**: Suppose L is NP-complete. For any $L' \in NP$,

$$L' \leq_p L.$$

- Hence, if $L \in P$, then $L' \in P$. QED
- Contrapositive says: *if $P \neq NP$, then no NP-complete language is in P.*
- We don't know whether P = NP, however:
  1. It's really hard to answer this question, so you are not likely to do so by finding a polynomial-time, deterministic TM to decide an NP-complete language.
  2. Same goes for finding an algorithm in *any* model of computation polynomially equivalent to deterministic TMs!
  3. Most people conjecture that $P \neq NP$.
- In conclusion, NP-complete languages are *practically impossible* to decide in polynomial time.
- Note that, w/r to our previous discussion of optimization, if the canonical decision problem $DEC_Q(x, y)$ is NP-complete, then we cannot find a polynomial time algorithm to compute optimum $Q(x)$ unless P = NP.

3 Our First NP-Complete Language

Is there such a thing as an NP-complete language?

- How can you prove that a language L is as hard to decide as *any* language in NP?
- Must show a polytime reduction from an arbitrary language in NP to L.
- *Idea*: as we did for PCP, we will identify a language L^* for which membership testing can be used to decide if a TM M accepts a string w.
- Previously, M was an arbitrary (deterministic) TM, so our reduction proved that PCP was undecidable.
- This time, M is an arbitrary *nondeterministic polytime* decider; that is, $L(M)$ is in NP.
- Hence, a polytime reduction will show that L^* is NP-hard.

On to the problem definition.

- Consider the set of all propositional Boolean formulas ϕ over the connectives $\wedge$, $\vee$, and $\neg$.
- **Example**:
$$\phi = (x \wedge y) \vee (\neg x \wedge z)$$
- Each propositional variable may be assigned a value of true or false.
- Depending on values assigned to vars, formula may be *true* or *false*.
- If assignment A of values to variable makes formula ϕ true, we say that A *satisfies* ϕ.
- **Example**: if $x =$ false, $y =$ false, and $z =$ true, then ϕ is true.
- Not every formula has a satisfying assignment!
- **Example**:
$$\psi = ((x \wedge y) \vee (\neg x \wedge z)) \wedge \neg(y \vee z)$$
is unsatisfiable.
- **Problem** (SAT): given a Boolean formula ϕ on variable set $X = \{x_1 \ldots x_n\}$, does there exist an assignment to X that satisfies ϕ?

4 SAT is NP-Complete

Thm (Cook, Levin): SAT is NP-complete.

- First, let's check that SAT is in NP.
- A certificate for formula ϕ is a satisfying assignment A to ϕ's variables!
- If ϕ uses all its variables, it certainly includes $\Omega(|A|)$ symbols, so A has size polynomial in ϕ.
- Moreover, we can verify ϕ by plugging in the assignment and evaluating the formula!
- Can be done in time proportional to size of ϕ: evaluate logical expressions from the inside out, taking constant time per logical operation in ϕ. QED

OK, but why is SAT complete for NP?

- **Claim**: let L be any language in NP. Then $L \leq_p$ SAT.
- **Pf**: L can be decided by some nondeterministic TM N in time polynomial in its input size.
- In particular, say that N decides L in time $t(n) \leq n^b$, for some constant b.
- If $w \in L$, there exists an accepting computation C of N on w that takes at most $w|^b$ steps.

- (For simplicity, assume it takes exactly $|w|^b$ steps, or that we replicate the last configuration as needed if N halts early.)
- Note that N cannot touch more than $|w|^b$ tape cells during computation C, so $|C| = O(|w|^{2b})$.
- We will develop a Boolean formula $\phi(|w|)$ such that ϕ is satisfiable iff accepting computation C exists.

And now, the Cook Tableau!

- Let's imagine writing down an accepting computation C of N on input w as a 2D table of size $t(n) \times t(n)$.
- Row i of the table contains the ith configuration c_i of C.
- Three key observations about this table:
  1. First row contains the initial configuration c_1 of N for w
  2. Last row contains a configuration with state q_a, the accepting state.
  3. For each two successive configurations c_i and c_{i+1}, $c_i \vdash c_{i+1}$.
- Our formula Φ will check the three conditions above.

Let's get started.

- Define the Boolean variable $x_{i,j,s}$ to be 1 if tape cell j of configuration c_i contains symbol s, or 0 otherwise.
- Every cell of the table contains some symbol. To express this, define

$$\phi_c = \bigwedge_{1 \le i,j \le t(n)} \left(\bigvee_s x_{i,j,s} \right) \wedge \left(\bigwedge_{s \ne t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right).$$

- ϕ_c is satisfied only by truth assignments to the variables that meet the criterion that each cell contains a unique symbol.
- Next, consider the first condition: c_1 had better be q_0, followed by w, followed by blanks.
- We can express this constraint as the following formula ϕ_s:

$$\phi_s = x_{1,1,q_0} \wedge x_{1,2,w[1]} \wedge x_{1,3,w[2]} \wedge \ldots \wedge x_{1,|w|+1,w[|w|]} \wedge \wedge x_{1,|w|+2,\Delta} \wedge \ldots \wedge x_{1,t(|w|),\Delta}.$$

- ϕ_s is satisfied precisely when all its variables are 1, and hence, when every cell in c_1 is as required for the initial configuration "$q_0 w$".
- Similarly, consider the second condition: $c_t(n)$ must contain the accepting state q_a.
- We can express this constraint as the formula ϕ_a:

$$\phi_a = \bigvee_{1 \le j \le t(n)} x_{t(n),j,q_a}.$$

What about condition 3?

- We need a formula that is satisfied precisely when each c_i of the table entails c_{i+1} (or c_{i+1} just replicates c_i, since we are allowed to repeat a configuration to pad out the table to n^b rows).
- As we saw with PCP, two successive configurations cannot differ except in a small area around the TM's head.
- In this area, the difference must reflect a legal move of the TM.
- As before, judging the legality of a move requires checking what happens to up to three symbols of each configuration: the head state, and the symbols to its left and right.
- Hence, we can check that $c_i \vdash c_{i+1}$ by looking at "windows" of 3×2 adjacent symbols within the table:

- Let $A_1 \ldots A_z$ be all legal lists of six symbols that can appear a window.
- (There are only $O((|Q| \times |\Gamma|)^6)$ such sets, independent of the input size.)
- For each $i < n^b$, $j \leq n^b - 2$, and $k \leq z$, define a formula $\psi_{i,j,k}$ as follows:

  $\psi_{i,j,k} = x_{i,j,A_k[1]} \wedge x_{i,j+1,A_k[2]} \wedge x_{i,j+2,A_k[3]} \wedge x_{i+1,j,A_k[4]} \wedge x_{i+1,j+1,A_k[5]} \wedge x_{i+1,j+2,A_k[6]}$.

- Now define formula ϕ_e as follows:

$$\phi_e = \bigwedge_{1 \leq i < n^b, 1 \leq j \leq n^b - 2} \left(\bigvee_{1 \leq k \leq z} \psi_{i,j,k} \right)$$

- Formula ϕ_e is satisfied only if every window of the table is legal, or equivalently if each configuration in the table entails the next (or is identical to the next).
- In other words, ϕ_e is satisfied only by truth assignments corresponding to valid computations of N.

Almost there!

- Finally, let $\Phi = \phi_c \wedge \phi_i \wedge \phi_a \wedge \phi_e$.

- We've argued that Φ can be satisfied precisely when there is an accepting computation of N on w.
- So how big is Φ?
- Each of the component formulas has constant size *per* table cell i, j.
- This is obvious for ϕ_a and ϕ_i.
- Let A be the number of distinct symbols that can occur at any position of a configuration.
- Then ϕ_c has $O(A^2)$ variables per i, j, while ϕ_e has $O(A^6)$ per (i, j) (since $z = O(A^6)$).
- But A is a constant for N independent of its input size.
- Conclude that Φ has constant variables per i, j, and hence is of size $O(|w|^{2b})$ overall, which is polynomial in $|w|$.
- Hence, if we can decide SAT in polynomial time, then given N and w, we can form $\Phi(w)$ in time polynomial in $|w|$ and use it to decide acceptance for an arbitrary nondeterministic TM in polynomial time.
- Therefore, SAT is NP-hard. QED