Polynomail time reductions

reducing language A to language B A <=P B. There exists a function f: A → B , f to be deterministic polynimial time computable x is in A if and only if f(x) in B.

Fact: A <= B and B is polynomial time computable, then A must be polynomial time computable.

Proof: Let x be the input to A. f(x) is computed in time |x|k for some k. B is polynomial time computable. B computes its answer in time (|f(x)|)j.  |f(x)| is O(|x|k): B computes in time (|x|k)j = |x|kj which is a polynomial in |x|.

Our reductions, these are called “mapping” reductions. We get a single call to B, and we must map “yes” to “yes” and “no” to “no”. This is also called “many-to-one” reductions. The conversion function f is not onto, and it does not need to be 1-1.

Turing reduction. We can call the “API” for B as many times as we want, and we can do post processing on the output.

We could not do a mapping reduction from ATM to the complement of ATM. But we can do a Turing reduction.

Definition: A language L is **complete** for a class (set) of languages C. If

a) L is in C

b) for every B in C, B <= L.

\* This does depend on the power we give to the conversion function f.

f can’t be more powerful than the class C.

Definition. A is complete for the class NP. If

a) A is in NP.

b) for every B in NP, B <= A.

Cook’s Theorem. The Satisfiability Problem is complete for NP. (NP-complete).

Satisfiability: A set of n variables, a logical propositional statement on those n variables:

a OR b AND (NOT c) → d Suppose we have a total of m terms in this compound statement.

Question: Is there an assignment of T/F to the variables that makes the compound proposition true?

We can solve this in 2n time. Just try all possible values.

Prove Satisfiability is in NP.

Guess an assignment to the n variables

Verify in O(m) time that the assignment makes the proposition true.

Consider an arbitrary NP language L. Since L is in NP, there exists a nondeterministic Turing machine M that decides L. Create a Satisfiability proposition that models how M runs on an input. Given an input x to machine M, if x is in L, then some execution / selection of choices, will lead M to the accept state.

Consider the entire trace of how M runs.

Create nk x nk table. Where nk is the running time of M on input x with |x| = n.

Each row is going to be the current state of the tape of M.

If we run for nk steps, then there will be nk rows in the table.

#q1x1x2x3….xn \_ \_ \_ \_ \_ \_ \_ (nk blanks ) #

#bq2x2x3 …. # (q1, x1) = (q2, b, R)

#q3bcx3 … xn # (q2, x2) = (q3, c, L)

If M accepts x, then the state qaccept will show up as a symbol in the table.

If L is in NP, there exists a polynomial time deterministic verifier that takes the table and verifies that it is a valid execution of M on input x, and that M accepts.

Create a giant propositional statement that checks that this is a valid table.

We need lots of variables. Ti,j,c = true if cell I in row j is storing symbol c.

OR (Ti,j,c) for all I, j

for each pair of c1 and c2, (not Ti,j,c1 or not Ti,h,c2)

→ Set it so that every cell has a valid symbol and no cell gets two symbols

We need to verify each transtion step.

Consider a 3 cells by 2 row region of the table, create a logical statement for each one.

[abc, abc] (Ti,j,a and Ti+1,j,b and Ti+2,j,c) → (Ti,j+1,a and Ti+1,j+1,b and Ti+2,j+1,c)

[q5bc, xdc] if (q5, b) = (q?, d, L)

We need to verifiy that qaccept is anywhere in the table.

(The Satisfiability statement is: Given some table, we can verify that it is a valid execution of M on x that leads to accept.)