L = {a1b1a2b2...anbn | a1..an  is in A, b1..bn is in B} A and B are regular. L is also regular.

There exists FSA MA and MB that decide A and B.

Create a machine M with states 2 x QA x QB

State qk,j,even  will be machine MA is in it’s state k and machine MB is in its state j, and we are reading an even index on input to new machine M.

delta(qk,j,even, 1) = qk,x,odd where M\_B’s transition function goes from state j to state x when it reads a 1.

delta(qk,j,odd, 1) = qy,j,even where M\_A’s transition function goes from state k to state y when it reads a 1.

Accept states are any qx,y,odd where x is an accepting state for A’s machine and y is an accepting state for B’s machine

2. L = {a a-complement}

a) Show L is not regular.

The pumping lemma only applies to strings larger than p symbols for some p we don’t get to decide.

W = 0p1p . Let w be split into xyz. We don’t to decide how to split the string w. The proof must work for all ways to split.

From pumping lemma |xy| <= p. We know that y must be all 0’s and at least one 0, from the pumping lemma |y| >= 1.

Applying the lemma, consider xy2z, this string has more 0’s than 1’s. Any string in L must have the same number of 0’s as 1’s.

b) Show L is context-free. But it is not.

Let w = 0p1p0p1p0p1p

Let w be divided into uvxyz with |vxy| <= p, and |vy| >= 1.

Suppose vxy all occurs in the first half of the string. If this happens then a starts with a 0, but uv2xy2zhas a-complement also starting with a 0.

The same holds if vxy is all in the second half of the string.

Suppose vxy sits across the middle of the string. Consider uxz. The a part starts with 0p but the a-complement will not start with 1p.

3) Show L = {0k | k is a power of 2} is not context free.

Let w = 0(2^p)

Let w be split into uvxyz where |vxy| <= p. So uv2xy2z has 2p + k symbols where 1 <= k <= p. The next string in L must have 2p +1 symbols but k < 2p.

Oracle Turing machines:

A Turing machine that has an “oracle” tape. The machine at any time can place a string onto the oracle tape, then in one step, query an oracle for some language, and the oracle will respond with Y/N, if the string is in the language for the oracle.

Ex: Suppose we have an oracle for the 3-SAT problem.

We can use this to decide the complement of 3-SAT is polynomial time.

How? Given a 3-SAT instance, ask the oracle if the instance is in 3-SAT, then flip the answer of the oracle.

Given a 3-SAT instance, use an oracle for 3-SAT to, in polynomial time, find an assignment that satisfies the instance.

For each variable, adjust the formula to figure out if it has to be set true or false: For x, add (x, a, b), (x, a, b), (x, a, b), (x, a, b) with a and b being new variables, to the formula and see if it is still satisfiable. If it is, you know that x can be set true. Leave these new clauses in and go to the next variable. If it is not, you know that x must be set false, remove these clauses and go to the next variable.

MA to represent a machine that has an oracle tape for language A.

P3-SAT represents all languages that can be decided by polynomial time oracle machines that use an oracle to the 3-SAT problem.

PNP all languages decided in polynomial time oracle machine that uses an oracle that can decide any NP languages

NPPSPACE languages decided in nondeterministic polynomial time using an oracle to any PSPACE problem.

All of our reductions are basically oracle machines. Polynomial time reduction: this is a polynomial time machine that makes a single query to an oracle machine and returns with the answer provided by the oracle.

(Ex: 3-SAT <=P L: Took an instance to 3-SAT, converted it to an instance to L, if f(x) is in L if and only if x is in 3-SAT.)

All of our proofs by contradiction use oracle machines. We ask a “black box” (oracle) how it decided an instance and then we had our machine do the opposite.

Theorem: There exists an oracle A where PA = NPA and also an oracle B where PB <> NPB. This implies that we can’t use a simple contradiction proof based on oracles to separate P from NP.

Proof Wednesday