Test 1) Show L = {<a,b,c> | there is an x < c, x2 = a mod b} a,b,c positive integers. Show L is in NP.

Create a NTM M:

Step 1) Verify that the input is three positive integers.

Step 2) Guess an x < c

Step 3) Verify x2 = a mod b in time polynomial in <a,b,c>.

Computing x2 can be done in n2 steps where n is the number of digits of x.

Guess a d, check that bd + a = x2.

Addition is linear in the number of digits.

2) Thrice satisfiable cnf formula. Given a formula does it have at least 3 solutions.

a) Show thrice-sat is in NP. Guess a certificate is three solutions to the formula. Verify (the input is a cnf formula) that each certificate is different (linear time) and that each piece of the certificate is a solution to the formula (linear time).

b) SAT <= thrice-cnf

Given a boolean formula w in cnf. Create the formula w’ = w AND (x OR y). This new formula is in cnf. X and y are two new variables. This adds a constant amount of clauses and variables so the reduction is polynomial time.

If w has at least one solution, then w’ has at least 3, (x = T, y =T), (x = T, y = F) , (x=F, y=T). If w does not have a solution neither does w’. So w is satisfiable if and only if w’ has at least 3 solutions.

3) 4-player game is PSPACE-complete. 4 players on a graph, each player moves the token, a token can never return to a visited vertex, and has to follow the directed edges. The last one to move the token wins.

a) Show 4-player game is in PSPACE.

Recursive process. Given G,s choose one edge for player 1, create a G’ = G-s, and the “starting vertex” is now u where (s,u) is the edge. (Repeat to try all such edges to see if any leads to a win.) Consider all possible 3 moves. (n3 such possibilities) Remove those vertices and recurse from the new starting vertex and each of these possiblities must lead to a win for the original player 1 move to be a win. The height of this recursion is O(n).

TQBF <= 4-player game. Do exactly the same reduction as for TQBF <= 2-player graph game with the following change. Replace each 2nd edge of the reduction graph (player 2’s move) with a path of 3 edges. That means player 2 makes a move, and then players 3 and 4 have no choice in their move. Then player 1 and player 2 have the same choices as with the 2-player game reduction.

Interactive Proofs:

We have a random verifier that can make queries of a prover. The verifier has to run in polynomial time. The prover is all powerful. (The prover just needs to be at least PSPACE.)

For a language L, if x is in L, there is a prover such that our verifier answers yes with probability >= 2/3.

If x is not in L, then for all “provers” the verifier will answer “yes” with probability <= 1/3.

The class IP is all languages for which we can have such a protocol.

Ex: #k-sat. Given a formula and a number k and ask if the formula has exactly k solutions (can adjust this to also be at least k).

#k-sat is in PH and not known to be in NP.

Idea:

Step 1: Ask the prover for how many solutions the formula has.

Reject if the answer is not k.

Step 2: Give the prover an adjusted formula where the first variable is held as a variable. f(x) = # of solutions of the formula where x is a variable representing the first variable of the formula. Check that f(“false”) + f(“true”) = k.

To do this, turn the boolean formula into a polynomial.

Each variable will now take 0 or 1 instead of true or false.

Build up the polynomial in the same way we create the formula.

1) Turn (NOT x) into 1-x

2) Turn (x AND y) into xy

3) Turn (x OR y) into 1-(1-x)(1-y)

We have this polynomial f.

Step 1: Ask the prover for the number of solutions to f with all variables set to 0 or 1. Reject if the answer is not k.

Step 2: Create f(x1) = the of solutions as a function of the first variable. This is a polynomial of degree at most m. Ask the prover for f(x1) and it will return the polynomial. Check f(0) + f(1) = k.

(Wrong step 3: Create f(x1, x2) = the number of solutions as a function of x1 and x2. As the prover for this polynomial. Check that f(0,0) + f(0,1) + f(1,0) + f(1,1) = k)

Trick is to make the variables over a larger number of values that {0,1}.

Find a large prime number p > 2n. We can do this randomly in polynomial time. Now all the variables can have value 0,…,p-1 and the math is done mod p.

A polynomial of degree n has only n roots. Given n points, we can uniquely define a polynomial of degree n.

Step 3: Choose a random number between 0 and p-1; z1. Ask the prover for the number of solutions, f(z1, x2). This is a polynomial in x2. Check that f(z1, 0) + f(z1, 1) = f(z1).

Step 4: Choose a random number z2 : 0, .. p-1, and ask the prover for f(z1, z2, x3). Check that f(z1,z2,0) + f(z1, z2, 1) = f(z1, z2).

Repeat until step n.

Theorem: IP = PSPACE.

Zero-knowledge proofs. Interactive proofs. The prover needs to convince the verifier but not reveal any information about the solution.

We need to convince the verifier that x is in L but not reveal the certificate.

MIP: Like IP but the verfier can ask multiple provers. The provers are not allowed to communicate with each other.

Theorem: MIP = NEXPTIME

Theorem: MIP\* (we add quantum to MIP) = the set of recognizable languages.