CSCC63 TUT 0002

Tutorial 6

richard.hong@mail.utoronto.ca

Proving decision problems are in NP/co-NP

Show that L is in NP

1. Show that L is a decision problem
2. Provide a certificate for the verifier
3. Justify why the certificate is polynomial in terms of the size of the input
4. Provide a Verifier that runs on the input/certificate
5. Justify the Verifier halts and verifies solutions in polynomial time

Prove that NP is closed under the concatenation operation

Prove that if L1, L2 are in NP, then L1 + L2 is in NP
L1 + L2 = { x | x = yz, y in L1, z in L2 }

Assume that L1, and L2 are in NP
L1 = { x | exists c (certificate), |c| polynomial in terms of |x|, V1(x, c) accepts in O(|x|^k) }
L2 = { x | exists c (certificate), |c| polynomial in terms of |x|, V2(x, c) accepts in O(|x|^j) }

1. L1 + L2 is a Decision problem, either elements belong to the set or it doesn't
2. Let x be the input, C = (i, c1, c2) such that c1 verifies x[0:i], c2 verifies x[i:]
3. |C| = |c1| + |c2| , by assumptions, C is polynomial in terms of |x|.
4. define V on input <x, C = (i, c1, c2)>:
   assert i in [0, |x|] # O(|x|)
   assert V1(x[0:i], c1) and V2(x[i:], c2) both accept # O(|x|^max(k, j))
   if both all assertions pass accept, otherwise reject
5. All assertions are done in polynomial time, since V1 and V2 run in polynomial time.

Prove that co-NP is closed under the concatenation operation

Prove that if L1, L2 are in co-NP, then L1 + L2 is in co-NP
L1 + L2 = { x | exists y in L1, z in L2, such that x = y + z }
co-(L1 + L2) = { x | forall y in L1, z in L2, such that x != y + z }
co-(L1 + L2) = { x | for i in [0, |x|], x[0:i] in co-L1 or x[i:] in co-L2 }

Assume that L1, and L2 are in co-NP
co-L1 = { x | exists c (certificate), |c| polynomial in terms of |x|, V1(x, c) accepts in O(|x|^k) }
co-L2 = { x | exists c (certificate), |c| polynomial in terms of |x|, V2(x, c) accepts in O(|x|^j) }

1. This is a decision problem, either an element is in L1 + L2 or its not.
2. Let x be the input, let C = [C_0, C_1, ... C_|x|], each C_i = (i, c1_i, c2_i), i in [0, |x|]
3. |C| = |x|, we know each |C_i| is polynomial from the previou question say O(|x|^m), then we have O(|x| * |x|^m)
4. define V on input <x, C = [C_0, C_1, ... C_|x|]>:

```
    for each C_i = (i, c1, c2) in C: # O(|C|) = O(|x|)
        assert that i in [0, |x|] # O(|x|)
        assert that atleast one of V1(x[0:i], c1) or V2(x[i:], c2) accepts # O(n^c)
        if any assertions fail, reject
    accept # we accept within O(|x| * (|x| + n^c))
```