CSCC63 TUT 0002

Tutorial 4

X = { <M> | M accepts a string, then M loops on some other string with the same length }
X = { <M> | Forall x such that x in L(M), M loops on some other string of length |x| } (set notation)
X-c = { <M> | M accepts a string and M halts on all other strings with the same length }
X-c = { <M> | Exists x such that in L(M), M halts on all other strings of length |x| } (set notation)

```
let w_i be enumerations over Sigma*
R on input <M>:
  for i = 0 to infinity:
    for j = 0 to i:
      run M w_j for i steps
      if M accepts:
        flag = True
        forall x in Sigma* such that |x| = |w_j|:
            if M doesn't halt in i steps:
               set flag = False
      accept if flag
```

X = { <M> | if M accepts any string, then M loops on some other string with the same length }
Reduce from HALT-c = { <M, w> | M loops on w }

P on input <M, w>:
 define M' on input <x>:
  1. pass
  2. run M on w
  3. accept
 return M'

Show that <M, w> in HALT-c iff <M'> in X
Assume <M, w> in HALT-c, line 2 will loop then M' will loop. Thus M' will accept nothing.
If M' accepts no strings, then M' in X, since X is the set of all M such that if it accepts some string, it must loop on some other string of the same length. M' accepts no strings, therefore we don't care if it loops on some other string.

Assume that <M, w> in HALT/<M, w> not in HALT-c, line 2 finishes executing, and line 3 runs. M will accept all strings, it never loops. Therefore <M'> cannot be in X

Assume <M'> in X, then M must have looped on w because otherwise it will accept all strings
Since HALT-c is not recognizable, and HALT-c reduces to X, X not recognizable

WTS Atm = { <M, w> | M is a TM that accepts w } is not decidable

1. Assume that Atm is decidable and that it is decided by a TM M_A
2. define D on input <X>:
   run M_A on <X, X>
   if M_A accepts (meaning X accepts X):
     reject
   otherwise accept

Basically D takes in a TM and checks if it accepts itself, if so, D rejects otherwise D accepts

Suppose we run D on <D>
if M_A accepts <D, D> then D must have accepted itself. But we are defining D and we are telling it to reject the next line.
if M_A rejects <D, D> then D must have rejected itself. But we are defining D ad we just told it to accept the next line.

Question 8

ALL_TM^HALT = { <M^HALT> | M^HALT in ALL_TM }
ALL_TM^HALT-C = { <M^HALT> | M^HALT not in ALL_TM }

HALT^ALL_TM = { <M^ALL_TM, w> | M^ALL_TM halts on w }

Consider using enumeration over TMs, let M_0, M_1 be TMs

```
P on input <M^HALT>:
  define M' on input <x>: #Remember we have access to ALL_TM oracle
    # have to use ALL_TM oracle in here somewhere
return <M', s>
```

WTS That if M^HALT rejects some string, then M must halt on s AND
That if M^HALT accepts all strings, then M must loop on s

Show ALL TM not co-recognizable

Consider ALL_TM reduction from HALT-C

```
P on input <M, w>
  define M' on input <x>
    run M on w for |x| steps
    if M halts, loop otherwise accept
return M'
```

L_5 = { <M> | Exists w in Sigma* such that M accepts w within 2|w| and forall w' in Sigma* M doesn't accept in fewer than |w'| steps }
L_5-C = { <M> | Forall w in Sigma* M does not accept w within 2|w| or exists w' in Sigma* M accepts in fewer than |w'| steps }