CIS 511 Homework 4

Stephen Phillips, Dagaen Golomb

February 23, 2015

Problem 1

We want to show that finding if a Turing Machine has a *useless state* is Turing Decidable. We formulate this as a language:

$$L = \{\langle M, q\rangle \mid q \in Q(M),\ \forall s \in \Sigma^* M(q)\, M \text{ does not enter } q\}$$

Now we reduce this to A_{TM} to show that it is undecidable. First as usual, we suppose toward contradiction that there was a decider of this language, which we will denote D. We build a Turing Machine in the following manner:

function $N(\langle M, x\rangle)$
  Build the description for the following machine
  function A(y)
    Ignore input y
    Simulate $M(x)$, and output what it outputs
  end function
  Simulate $D(\langle A, q_A\rangle)$, where q_A is the accept state of D, and output what it outputs
end function

If $M(x)$ halts and accepts, then we eventually reach the accept state of A, by construction. Also by construction we never reach the accept state if $M(x)$ loops or if it rejects. Since D is a decider, and building the machine description takes finite time, N always halts. Therefore N accepts if and only if D accepts, which in turn accepts if and only if $\langle M, x\rangle \in A_{TM}$. Therefore, N decides A_{TM}, a contradiction.

Problem 2

Problem 3

We want to show that the intersection of two context free languages is undecidable using A_{TM}.

Problem 4

Problem 5

Show that the language

$$ISO = \{\langle G, H\rangle\} \mid G \text{ and } H \text{ are isomorphic graphs}$$

is in NP.

To do this we need to show a verifier V for ISO. As one might expect, the certificate for a member of the language $\langle G, H\rangle$ is the isomorphism $\phi : V_G \to V_H, v \to \phi(v)$ between them. The size of such an isomorphism would be about $2n$ where n is the number of nodes in G and H, so it polynomial in the size of G and H. We also need to check that the verifier V using this runs in polynomial time. A simple algorithm to use this is the following:

function $V(\langle G, H\rangle, \phi)$
  If the number of nodes or edges in G differ from the number of nodes or edges in H, reject
  for Nodes v in G **do**
    If the number of edges (v, u) in G differ from the number of edges $(\phi(v), u')$ in H, reject

for Edges (v, u) in G **do**
If $(\phi(v), \phi(u))$ is not in H, reject
end for
end for
Accept
end function

By assering that the size of the graphs are the same, that each node v has the same degree as $\phi(v)$ and maps to the corresponding vertices, we have shown that the isomorphism is correct. If there is no isomorphism then at least one of these tests will fail. Therefore $V(\langle G, H \rangle, y)$ will accept for some input y if and only if there is an isomorphism between G and H.

Problem 6

Problem 7

For a language A to be NP-complete, it needs to satisfy two conditions:

- $B \in \text{NP}$
- $\forall L \in \text{NP}, A \leq_p B$

We want to show that if $\text{P} = \text{NP}$, then every language $A \in P$ exept $A = \varnothing$ and $A = \Sigma^*$ is NP-complete.

The basic idea is that since we have polynomial time algorithms for all NP-complete problems like SAT, we map each language to SAT, solve it, then map that output to appropriate inputs for the original language, i.e. use $L \leq_p$ SAT then that $L \leq_p$ SAT (kind of a hack).

We know that SAT $\in NP$, and since by assumption $\text{P} = \text{NP}$, SAT $\in P$. That means there must be a polynomial time turing machine M_{SAT} that accepts SAT in polynomial time. We also know that SAT is NP-complete, so for every language $L \in \text{NP}$ there is a polynomial time function $f_L^{SAT} : \Sigma^* \to \Sigma^*$ that satisfies $w \in L \iff f(w) \in \text{SAT}$.

Given a language $A \in \text{P}$ besides the empty and full languages, we know there exists strings $s_{acc} \in A$ and $s_{rej} \notin A$. Since we assume $\text{P} = \text{NP}$, we know $A \in \text{NP}$. Using all of this we create the following map for a language $L \in \text{P}$:

function $f_L^A(x)$
Let $y = f_L^{SAT}(x)$
Simulate $M_{SAT}(y)$, and record its output
If it accepted, output s_{acc}
Otherwise, output s_{rej}
end function

This machine satisfies $x \in L \iff f_L^A(x) \in A$ since it outputs $s_{acc} \in A$ if M_{SAT} accepted and $s_{rej} \in A$ if M_{SAT} rejected, and we know M_{SAT} accepts if and only if $f_L^{SAT}(x) \in \text{SAT}$. We also know this runs in polynomial time, since $f_L^{SAT}(x)$ runs in polynomial time and $M_{SAT}(y)$ runs in polynomial time, and the rest are just a constant number of operations. Therefore this is a polynomial time map from aribtrary language L to arbitrary language A and so.

Since A satisfies both the conditions to be NP-complete, and A was any language in P (with the two exceptions), then any langauge in P is NP-complete if $\text{P} = \text{NP}$.