Problem Set 7: CS103

Katherine Cheng, Richard Davis, Marty Keil

June 3, 2015

Problem 1: Closure Properties of RE

i. bool inL1nL2(string w) {
   return inL1(w) && inL2(w);
   }

   Will return true when true, returns false or loops infinitely otherwise.

ii. Evaluates left to right (confirm?), so if the first function is an infinite loop, will not evaluate the second function (which might be true)

iii. bool imConvincedIsInL1uL2(string w, string c) {
   return imConvincedIsInL1(w, c) || imConvincedIsInL2(w, c);
   }

   Now won't get caught in an infinite loop.

Problem 2: Password Checking

1. meta level - by contradiction that password checking language is RE/verifiable. Then we can create this. now (i). there is some certificate c where imConvinced will return true, we accept(). If we accept, the language of the machine is (we have 2 levels of machines, (1) candidate password checkers, (2) looks at machines and says if it's a password checker) higher machine has said you're only going to accept p. Now, reach a contradiction. If we are a passwordchecker, just accepts always regardless of the input. input of the machine is everything.

   contradiction: if it is a password checker then it accepts all inputs

2. it'll infinitely loop

   we're guaranteed the pass checker returns p and only p

   if we loop infinitely on every string,

3. input equal to p?

Problem 3: Equivalent TMs

m1 takes an input, it will accept/reject/loop m2 has some other language if imConvinced true, then language does not match what you have coded

1. REG
2. ALL
3. REG
4. REG
5. REG (regular languages closed under union)
6. R (M-N so not REG, but CFG so R)
7. RE (?)
8. ALL (there is a TM where L={$\emptyset$} will loop forever, can't be RE)
9. RE (n is the certificate) (what happens when it loops? don't worry about that)
10. ALL
11. ALL
12. RE

if in RE, if I give you a machine, deep down I know this is in the machine, machine will halt. R if not in language, will halt, which is not guaranteed. Not in RE, I can give you (M,n) that's in the language and you can't conclusively tell me it's in the language. ex) at most length 5, run on all strings on at most length 5, if in language, will accept it. I'm giving you something in the language, if you halt

Problem 4: The Big Picture

Problem 5: 4-Colorability

i. 3COLOR $\in$ NP because it is NP-complete and NP-complete is a subset of NP. Therefore if 3COLOR can be reduced to 4COLOR, then 4COLOR is also an element of NP. Take every 3COLOR graph, in which there exists a way to color each of the nodes one of 3 colors, such that no two nodes of the same color are connected by an edge. Each of these graphs is also by definition 4-colorable, just one of the

4 colors will go unused. Therefore, there is polynomial time reduction that shows that 4COLOR must also be in NP.

I'm am not sure about this explanation. I actually think they might be looking for:

We can guess different 4Colorings of a graph using a nondeterministic turing machine. We can then deterministically check whether the coloring option is a legal 4-coloring of G. This non-deterministic turing machine us in polynomial time, because it would only take the number of nodes, n, to check if this was a valid coloring for the graph.

ii. Take an arbitrary graph G and contrast Graph G' by adding a new node that is connected to all other nodes.

1. If the original graph G is 3-colorable, then the new graph G' must be 4-colorable because in the original graph G at most three colors were used so that no two nodes of the same color were connected by an edge. If a new node is added that connects to all other nodes, then this node must be of a new color. This new graph would require one additional color to prevent the same color from being connected by an edge, bringing the total to at most 4 colors, the definition of a 4-colorable graph.
2. If we take G' and complete the transformation that removes the node that connects to all other nodes, we have then reduced the colorable number by 1. This is because that node must have been a different color than all other nodes, by definition of colorability. Since G' was 4-colorable(had at most 4 colors needed), G is therefore 3-colorable(has at most 3 colors needed).
3. This reduction can be completed in polynomial time n+1. The reduction must add one node to graph G, then make n (the number of nodes in graph G) edges between the new node and all previous nodes in G.

Problem 6: Resolving $P \overset{?}{=} NP$

1. neither
2. neither
3. $P = NP$
4. $P = NP$
5. $P \neq NP$
6. $P \neq NP$
7. neither
8. neither
9. neither

10. neither
11. neither
12. neither
13. P = NP
14. P = NP
15. neither
16. neither

Problem 7: The Big Picture

i. you can make a DFA

ii. prove context free? provide grammar. Prove not regular? M-H

iii. give a regular language, since regular languages $\subseteq$ P

iv. traveling salesman, 3SAT, we don't know if it's in P because we don't know if P=NP

v. A_{TM}, can't be a decider

vi. looping, can't be a verifier