CSE 34151 Theory of Computing: Homework 8
Reducibility

Version 1: April 11, 2018

Instructions

- Unless otherwise specified, all problems from "the book" are from Version 3. When a problem in the International Edition is different from Version 3, the problem will be listed as V3:x.yy/IE:x.zz, where x.zz is the equivalent number. When Version 2 is different, it will be listed as V3:x.yy/V2:x.zz. If either IE or V2 do not have a matching number, the problem text will be duplicated.
- You can prepare your solutions however you like (handwriting, LaTeX, etc.), but you must submit them in **legible PDF**. You can scan written solutions on the printer in the back of the classroom, or using a smartphone (with a scanner app like CamScanner). It is up to you to ensure that submissions are legible. REMEMBER THAT IF WE CAN"T READ IT OR SCAN IS CUT OFF, YOU DON"T GET A GRADE FOR IT.
- The problems marked as "TEAM" may be solved in a collaborative fashion with up to 2 other students. In such cases, your submission should have the word "TEAM" at the start of the problem, followed by the names of your collaborators (must be other students in this class). When such problems are graded, the first submission encountered by the grader for the team will be used for a common grade for all identified team members.
- Please give every PDF file a unique filename.
  - If you're making a complete submission (all problems), name your PDF file `netid-hw#.pdf`, where `netid` is replaced with your `NetID` and # is the homework number.
  - If you're submitting some problems now and other problems later, name your file `netid-hw#-1-2-3.pdf`, where `1-2-3` is replaced with just the problems you are submitting now. This may be useful for team submissions.
  - If you use the same filename twice, only the most recent version will be graded.
  - The time of submission is the time the most recent file was uploaded.
- If you use LaTeX and want to draw something like a state diagram, consider using the `tikz` package. A reference document is on the website under "Assignments".
- You may also find the website http://madebyevan.com/fsm/ a useful tool for drawing state diagrams via drop and drag. It will output both .png image files and latex in the tikz format.
- Submit your PDF file in Sakai to the appropriate directory. Don't forget to click the Submit (or Resubmit) button!

Practice Problems

These problems are from the book, and most have solutions listed for them. They are listed here for you to practice on as needed and any answers you generate **should not** be submitted. You are free to discuss these with others, but you are not allowed to post solutions to any public forum.

1. V3:4.1 Membership in decidable languages.
2. V3:4.5 Complement of E_{TM}
3. V3:4.ba,d function properties
4. V3:4.10, 4.12, 4.14, 4.25 prove language decidability

5. V3:5.5 Reducibility
6. V3:5.7
7. V3:5.10 and 5.11 properties of TMs
8. V3:7.1c,d V3:7.2 c,d Big O notation
9. V3:7.8 Problems in P

Book Exercises

These problems are found in the text book and are to be answered and submitted by each student. **If they are not marked as "TEAM," you are to solve them individually.** If they are marked as "TEAM" you may submit the same answer as the others in your team. In any case, use of solution manuals from any source or shared solutions is a violation of the ND Honor Code. You are also not allowed to show your solutions to another student not part of your TEAM.

1. (5 points) 4.2 Show decidability
   Solution: Define $EQ_{DFA,REX} = \{< A, R > |$A a DFA and R a regex such that $L(A) = L(R)\}$
   To show decidability describe a TM E that accepts a $< A, R >$ and always halts:
   (a) Verify A is a properly formatted DFA and R a validly formatted regex, and reject if not.
   (b) Convert R into a DFA M using the algorithm in proof of Lemma 1.55
   (c) Use the TM from EQ_{DFA} on $< A, M >$. If it accepts, accept and if it rejects, reject.
2. (10 points - TEAM) V3:4.11, V2:4.10, IE:4.31. Show decidability. Hint: consider what the pumping lemma tells you about the size of languages and when they are infinite.
   Solution:
   $INFINITE_{PDA} = \{< M > |$ M is a PDA and L(M) is infinite$\}$
   The pumping lemma tells us that if there is a string with length greater than the pumping length, then that string alone has an infinite number of pumped strings that are also in the language. So the answer is to compute the pumping length and then see if there are any strings of length greater than p by enumerating them one at a time in length order. If we find one greater than p, stop and accept. If no strings greater than p than stop and reject.
   (a) Verify N is a valid PDA description, and reject if not.
   (b) Convert P to a CFG using the algorithm from Lemma 2.27
   (c) Compute a bound on its pumping length from $p = b^{|V|+1}$
   (d) Let T be a regex that generates all strings of length greater than p. Use TM from Theorem 4.8 (E_{CFG}) to decide if L(G) $\cap$ T is empty.
   (e) If so, reject; else accept.
3. (5 points) 5.1 Show undecidability. Include a description of the language.
   Solution:
   $EQ_{CFG} = \{< A, B > |$ and and B are CFGs such that L(A)=L(B)$\}$
   Assume R is a decider for EQ_{CFG}. Now construct a decider S for ALL_{CFG} as follows:
   (a) Verify N is a valid PDA description, and reject if not.
   (b) Construct a CFG G2 such that L(G2)=Σ^*
   (c) Run R on $< G, G2 >$
   (d) Have S accept and reject as R does.

   If R exists, then we have just built a decider for S. But that is impossible, so R cannot exist.
4. (10 points) 5.2. Show co-Turing Recognizability. Remember co-Turing recognizable means that a TM can be built to recognize (not decide) the complement of a language. Include a description of the language being recognized.
   Solution:
   $NEQ_{CFG} = \{< G1, G2 > |$G1 and and G2 are CFGs such that L(G1) is not equal to L(G2)$\}$

Remember that a TM decider M_A exists for A_{CFG} (p. 198, a.k.a Theorem 4.7), i.e. Given a $< G, w >$ it accepts if w is in G and rejects if not. Being a decider, it never loops.
A TM M that recognizes the complement of EQ_{CFG} takes as input $< G1, G2 >$, and accepts if the two grammars generate different languages. We have no idea what happens if they are the same. Following are the steps that M performs.

(a) Verify G1 and G2 are properly formatted CFGs, and reject if not.

(b) Using an enumerator, generate each string one at a time from Σ^* in lexicographical order. Call it x.

(c) Run M_A on $< G1, x >$ (Remember its a decider so it always stops.)

(d) Run M_A on $< G2, x >$ (Remember its a decider so it always stops.)

(e) If one accepts and the other rejects, accept (as we have found a string in one but not the other)

(f) Repeat otherwise

If the two grammars generate different languages, then eventually we will find a string in one and not the other. If they are the same, this will loop forever.
Alternative solution: use one grammar to generate strings and use theorem 4.7 to check if other grammar accepts it. I took this (because I didn't see the error until I finished grading), but it really is not a valid test, as what if one language is a superset of the other and the grammar you chose for the generator is the smaller one.

5. (5 points) 5.3. Post Correspondence Problem. Note this does not need a proof, just solve the PCP puzzle using any (and only) number of tiles from the specified set.
Solution: 7 tiles that spell out "abababbbaaaa" on top and bottom
Second solution: 4 tiles "aaaabab"
Third Solution: 10 tiles "abababbbaaabaaabab"
4th solution: 13 tiles: "aababababaabaababababababab"

6. (10 points - TEAM) V3:5.12, V2:5.12, IE:5.28 Undecidability. Show the language. Hint: study the solutions to 5.10 and 5.11 (IE:5.26,27).
Solution:
Define $W = \{< M > |$M a 1-tape TM that writes a blank over a non-blank$\}$.
Remember that $A_{TM} = \{< M, w > |$ M accepts w$\}$ is undecidable.
Assume TM R decides W. Construct TM S that decides A_{TM} as follows:

(a) Verify input M is properly formatted TM, and reject if not.

(b) Use the input $< M, w >$ to construct a new TM M' that does the following

i. Simulates M on w, except when M uses a blank "$\sqcup$" in writing, M' uses a different $\sqcup'$. M' also treats $\sqcup'$ just like a $\sqcup$ when reading.

ii. If the simulated M accepts, M' writes $\sqcup'$ by $\sqcup$ over a non-blank before accepting.

(c) Run the decider R on M'.

(d) If R accepts (i.e. it knows M accepted w and wrote a non-blank over a blank), then accept. Else reject.

If M' accepts, then M accepted and R knew this because we had M' write a blank over a non blank. Thus S knows if M accepts w or not, and thus is a decider for A_{TM}. But no such decide can exist, thus R cannot exist. Thus W is not decidable.

7. (5 point) 7.9 membership in P
Solution:
$TRIANGLE = \{G|$ G a graph with 3 vertices with edges that form a triangle$\}$. Construct M that accepts $< G >$ as follows:

(a) Verify input G is properly formatted graph, and reject if not.

(b) Assume V vertices in G

(c) Generate all possible triples of vertices (v_1, v_2, v_3). There are $V * (V - 1) * (V - 2)$ of these for $O(V^3)$ triples.

(d) if (v_1, v_2), (v_1, v_3), and (v_2, v_3) are all edges in G, halt and accept. Each of the scans of the vertexes may tape O(n) to scan the tape.

(e) If no triangle found, reject

Since the test for edges is O(V), and there are $O(V^3)$ triples, the algorithm is a decider that runs in polynomial time.