Assignement 4
COSC 4P61, Theory of Computation
Fall, 2017

Due: Dec. 5, Tuesday, 5:00 PM.

1. (30) Construct a Turing machine for $L = \{a^n b^n c^n | n \geq 1\}$. Please first describe the idea behind your construction in English.

δ	a	b	c	X	Y	Z	B
q_0	(q_1, X, R)				(q_4, Y, R)		
q_1	(q_1, a, R)	(q_2, b, R)			(q_1, Y, R)		
q_2		(q_2, b, R)	(q_3, Z, L)			(q_2, Z, R)	
q_3	(q_3, a, L)	(q_3, b, L)		(q_0, X, R)	(q_3, Y, L)	(q_3, Z, L)	
q_4					(q_4, Y, R)	(q_4, Z, R)	(q_f, R, B)
q_f							

2. (bonus 20) Construct a Turing machine for adding 1 to a binary number n with no leading 0's (i.e. $n \in \{0\} \bigcup \{1\}\{0+1\}^*$). Initially, the binary number is on the tape enclosed by a pair of #'s with state q_0 and head pointing to the left #. For convenience, we can assume that the tape is a two way infinite tape. For examples, for input #11111#, the output should be #100000#; for input #10111#, the output should be #11000#; while for input #100#, the output should be #101#.

$$
\begin{aligned}
\delta(q_0, \#) &= (q_1, \#, R) \\
\delta(q_1, 0) &= (q_1, 0, R),\ \delta(q_1, 1, R) = (q_1, 1, R),\ \delta(q_1, \#) = (q_2, L) \\
\delta(q_2, 0) &= (q_f, 1, \_),\ \delta(q_2, 1) = (q_2, 0, L),\ \delta(q_2, \#) = (q_3, 1, L) \\
\delta(q_3, B) &= (q_f, \#)
\end{aligned}
$$

3. (25) Let G be an unrestricted grammar. Then the problem of determing whether or not $L(G) = \emptyset$ is undecidable. Let M_1 and M_2 be two arbitrary Turing machines. Show that the problem $L(M_1) \subseteq L(M_2)$? is undecidable.

   We reduce the first problem (known to be undecidable) to the second problem.

   Since G is an unrestricted grammar, there is a Turing machine M_1 that recognizes G. We can easily build a Turing machines M_2 such that $L(M_2) = \emptyset$ (for example, by having no final states, i.e., $F = \emptyset$). If the problem whether the problem $L(M_1) \subseteq L(M_2)$ is decidable, then we can also decide whether $L(G) = \emptyset$.

4. (15) Given $A = \{< M > | M$ is a finite automaton and $L(M) = \emptyset\}$ where $< M >$ is some encoding of the machine M. Is A Turing-decidable? Prove your answer.

   We know (from the decision algorithms for regular languages) that $L(M)$ is nonempty if and only M accepts a string of length less than n, where n is the number of states in the finite state machine. Therefore, all we have to do to decide this language is create a Turing machine that does the following: generate all strings of lengths less than n (there are finite of them) and check to see whether any one of them is accepted. This process is guaranteed to halt. Therefore, the language is Turing-decidable.

5. (30) Let M be a deterministic Turing machine that accepts a nonrecursive language. Prove that the halting problem for M is undecidable. That is, there is no TM that takes input w and determines whether the computation of M halts with input w.

The proof idea is as follows:

Assume that the halting problem for this particular Turing machine is decidable, argue that we can construct another Turing machine that actually decides L (the non-recursive language), thus L becomes recursive, a contradiction.

Proof: Let the non-recursive language be L. If the halting problem for M is TM-decidable (recursive), then by definition, there exists a Turing machine M_1 such that for any string w, M_1 is able to decide (in finite amount of time) whether M running on w will halt, i.e., M_1 will always halt on input $< M, w >$, $\forall w$, such that

- M_1 halts in final state if M halts on w (M may accept or reject w);
- M_1 halts in non-final state if M does not halt on w.

Now we can design a TM M_2 that decides L as follows:

- $\forall w$, run M_1 on $< M, w >$ (this will always halt)
- if M_1 halts on final state (meaning M halts on w)
  - run M on w (because M halts on w)
    - if M accepts w, M_2 halts and accepts w
    - if M rejects w, M_2 halts and rejects w
- if M_1 halts on a non-final state (meaning M does not halt on w)
  - M_2 halts and rejects w.

In short, $\forall w$, M_2 will always halt such that
if $w \in L$, M_2 accepts, and
if $w \notin L$, M_2 rejects it

Therefore, by definition, L is TM-decidable, i.e., recursive, which is a contradiction.

Note: Since L is non-recursive, and M accepts L, L is TM-recognizable, which means that if w is in L, M will always halt on w, and if w is not in L, M may halt on a non-final state or simply loop forever.