

ECE 537 - Foundations of Computing

Prof. Sen

Homework #4

Solutions

1. Design a Turing machine for decrementing an unsigned integer. Assume that decrementing zero yields zero. Write out the states or give the state graph.

Answer: We assume that the input is provided on the infinite tape such that the LSB of the binary number is stored in the rightmost nonblank cell of the tape. The head of the tape is pointing to the MSB on the left which is the first “1” of the number. The rest of the tape contains blank spaces which we denote with “ \sqcup .” So for example, the number 7 would be stored on the tape as “ $\sqcup 111 \sqcup$ ”, the number 2 as “ $\sqcup 10 \sqcup$ ”, etc. We now list the transition table for the TM which would decrement this number:

	State			h
state	0	1	\sqcup	Comment
q_0	$(q_1, 0, R)$	$(q_1, 1, R)$	-	check the 1st bit
q_1	$(q_3, 0, R)$	$(q_3, 1, R)$	(q_2, \sqcup, L)	only 1 bit in string?
q_2	$(q_F, 0, -)$	$(q_F, 0, -)$	-	handle case of only 1-bit string
q_3	$(q_3, 0, R)$	$(q_3, 1, R)$	(q_4, \sqcup, L)	move to the right end of the string
q_4	$(q_5, 1, L)$	$(q_F, 0, L)$	-	decr LSB: $1 \rightarrow 0$ (and we're finished) or $0 \rightarrow 1$
q_5	$(q_5, 1, L)$	$(q_6, 0, L)$	-	propagate the $0 \rightarrow 1$ decrement
q_6	$(q_F, 0, -)$	$(q_F, 1, -)$	(q_7, \sqcup, R)	is the MSB now a 0?
q_7	$(q_F, \sqcup, -)$	-	-	replace leading 0 with a blank character “ \sqcup ”
q_F	-	-	-	accepting state

As you can see, we first move the head to the end of the number to the LSB. If that symbol is a 1, we can decrement it to 0 and halt. If not, we then start scanning leftward along the tape, propagating the decrement operation by changing 0's to 1's until we hit a 1, which is the point at which the propagation will stop. At this point, we change the 1 to a 0 and then check if we are now at the front of the string. If this is the case, we replace the leading 0 with a # so that we get our new final answer.

2. Give implementation-level descriptions of Turing machines that decide the following languages over alphabet 0,1. This means that you can describe what the machine should do e.g. move left to right crossing off zeros, etc.

- (a) $\{w \mid w \text{ contains an equal number of 0s and 1s}\}$

Answer: This solution comes straight out of Sipser p. 163:

- i. Scan the tape and mark the first 0 which has not been marked. If no unmarked 0 is found, go to 3. Otherwise go back to the front of the tape.
- ii. Scan the tape and mark the first 1 which has not been marked. If no unmarked 1 is found, reject. Otherwise go back to the front of the tape and go to 1.

iii. move head to the back of tape and scan to see if any unmarked 1's remain. If none are found, accept, else reject.

(b) $\{w \mid w \text{ contains twice as many 0s as 1s}\}$

Answer:

i. Scan the tape and mark the first 0 which has not been marked. If no unmarked 0 is found, go to 4.

ii. Continue scanning and mark the next unmarked 0. If there are none, reject. Otherwise move to back to the front of the tape.

iii. Scan the tape and mark the first 1 which has not been marked. If no unmarked 1 is found, reject. Otherwise go back to the front of the tape and go to 1.

iv. move head to the back of tape and scan to see if any unmarked 1's remain. If none are found, accept, else reject.

(c) $\{w \mid w \text{ does not contain twice as many 0s as 1s}\}$

Answer: Same as the previous, except you switch the accept and reject states.

3. Show that decidable languages are closed under:

(a) union

(b) concatenation

(c) Kleene star

(d) complement

(e) intersection

Answer: For all of these answers, let L , L_1 , and L_2 be decidable languages, and M , M_1 , and M_2 be the TM's that decide them.

(a) union: Construct TM M' which decides the union of L_1 and L_2 :

On input w :

1. Run M_1 on w . If it accepts, then accept.

2. Run M_2 on w . If it accepts, then accept. Otherwise, reject.

(b) concatenation: Construct NTM M' which decides the concatenation of L_1 and L_2 :

On input w :

1. Guess a way to cut w into two parts $w = w_1w_2$

2. Run M_1 on w_1 .

3. Run M_2 on w_2 .

4. If both accept, accept.

5. IF all cuts have been tried without success, reject.

(c) Kleene star: (similar to the last one). Construct NTM M' which decides the L^* :

On input w :

1. Cut w into parts $w = w_1w_2\dots w_n$

2. Run M on w_i , for $i = 1, 2, \dots, n$.

3. If M accepts all of these strings w_i then accept.
4. IF all cuts have been tried without success, reject.

(d) complement: Easy, just flip the accept, reject states on M .

(e) intersection: Construct TM M' which decides the intersection of L_1 and L_2 :

On input w :

1. Run M_1 on w . If it rejects, then reject.
2. Run M_2 on w . If it accepts, then accept. Otherwise, reject.

4. A *useless state* in a pushdown automaton is one that is never entered on any input string. Consider the problem of determining whether a pushdown automaton has any useless states. Formulate this problem as a language and show that it is decidable.

Answer: Let $U = \{\langle P \rangle \mid P \text{ is a PDA that has useless states}\}$. The following TM T decides U :

$T =$ "On input $\langle P \rangle$, where P is a PDA:

- (a) For each state q of P :
- (b) Modify P so that q is the only accept state
- (c) The problem of determining whether a PDA has an empty language is decidable (the book calls it E_{PDA}), so use E_{DFA} to decide if the language is empty. If it is, then accept, this is a PDA with a useless state. If not, continue
- (d) If we pass all states of P and do not find any that are empty then reject, the PDA does not have any useless states.

5. True or false: every subset of a decidable language is decidable. If true, give a proof. If false, give a counter example.

Answer: False. This is simple to show, eg. L is the set of all strings with 0's and 1's, which is certainly decidable. Let L' be a subset of L which encode the halting Turing machines which is not decidable.

6. Let $E = \{\langle M \rangle \mid M \text{ is a DFA that accepts some string with more 1s than 0s}\}$. Show that E is decidable. Hint: use the properties of CFLs.

Answer: False. The language of all strings with more 1's than 0's is a context-free language, recognized by a PDA P that keeps a (positive or negative) unary count on its stack of the number of 1's minus the number of 0's that have been seen so far on the input. Build a TM M for deciding E as follows. When given a DFA M as input, use M and P to construct a new PDA R that recognizes the intersection of B and P (remember that the intersection between a CFL and a regular language is CFL!). Now build TM to test whether the language of R is empty, which is decidable. If the language is empty, then reject because this DFA does not accept some string with more 1's than 0's. If it is not empty, then accept. Hence E is decidable.

7. Let $A = \{\langle G \rangle \mid G \text{ is a connected undirected graph}\}$. Recall that a graph is connected if every node can be reached from every other node by traveling along the edges of the graph. Show that A is in P .

Answer: The algorithm to decide whether a graph is connected is given in Example 3.23 on p. 157 of Sipser. Here we do a high-level analysis of the algorithm to show that it runs in $O(n^4)$ time, hence is in P . Stage 1 takes at most $O(n)$ steps to locate and mark the start node. Stage 2 needs at most $n + 1$ repetitions, since each repetition except for the last one marks at least one node. Stage 3 uses $O(n^3)$ steps because G has at most n nodes and checking each node uses at most $O(n^2)$ steps. Together, stages 2 and 3 take $O(n^4)$ steps. Hence the algorithm runs in $O(n^4)$ time and is therefore in P .