CS 341 Automata Theory
Geoffrey Parker - grp352
Homework 11
Due: Tuesday, April 3

This assignment reviews Turing machine construction and covers Sections 17.3 - 17.6 and Chapter 18 and 19.

1) Define a Turing Machine M that computes the function $f: \{a, b\}^* \rightarrow N$, where:

$$f(x) = \text{the unary encoding of } max(\#a(x), \#b(x)).$$

For example, on input `aaaabb`, M should output `1111`. M may use more than one tape. It is not necessary to write the exact transition function for M. Describe it in clear English.

Solution. M will have three tapes. In the first phase, M will move from left to right over the input string. For each a, write a 1 to tape 2. For each b, write a 1 to tape 3. Overwrite the input on tape 1 with $\square$'s. Now move right to left on all three tapes in parallel. If there is a 1 on either tape 2 or 3, write a 1 to tape 1. If tapes 2 and 3 are both blank, halt. $\square$

2) Construct a Turing machine M that converts binary numbers to their unary representations. So, specifically, on input $\langle w \rangle$, where w is the binary encoding of a natural number n, M will output 1^n. (Hint: use more than one tape.)

Solution. Move left to right, copying the input from tape 2 to tape 1 and overwriting tape 1 with blanks. Write 1 on tape three. Now move tape 2 back to the left. For each character on tape 2, if the character is 1, copy the contents of tape 3 onto tape 1. Then, no matter what tape 2 is, double the contents of tape 3. $\square$

3) In Example 17.9, we showed a Turing machine that decides the language WcW. If we remove the middle marker c, we get the language WW. Construct a Turing machine M that decides WW. You may exploit nondeterminism and/or multiple tapes. It is not necessary to write the exact transition function for M. Describe it in clear English.

Solution. Move left to right over the input on tape 1, copying the input onto tape 2 as you go. At a nondeterministically chosen point, switch to writing the input to tape 3 instead of tape 2. When you reach the end of the input, compare tapes 2 and 3. If they match, accept, if not, reject. $\square$

4) In Example 4.9, we described the Boolean satisfiability problem and we sketched a nondeterministic program that solves it using the function choose. Now define the language SAT $= \{\langle w \rangle : w$ is a wff in Boolean logic and w is satisfiable$\}$. Describe in clear English the operation of a nondeterministic (and possibly n-tape) Turing machine that decides SAT.

Solution.

1. Simplify the expression. Replace all instances of $\neg true$ with $false$, $\neg false$ with $true$, $(false)$ with $false$, $true \wedge true$ with $true$, and so on. Keep doing this until there are no such expressions left in the string.
2. If the string as a whole is simply $true$, accept. If the string is $false$, reject.
3. Move to the start of the string. Move right until you find a predicate. Store the predicate as P and, nondeterministically, either true or false as x. Move over the string and replace each instance of P with x.
4. Return to the beginning of the string and repeat.

$\square$

5) What is the minimum number of tapes required to implement a universal Turing machine?

Solution. One. All n tape machines can be simulated with a 1 tape machine. $\square$

6) Encode the following Turing Machine as an input to the universal Turing machine:

$$M = (K, \Sigma, \Gamma, \delta, q_0, \{h\}), \text{ where: } K = \{q_0, q_1, h\},\ \Sigma = \{a, b\},\ \Gamma = \{a, b, c, \square\}, \text{ and } \delta =$$

q	σ	$\delta(q,\sigma)$
q_0	a	$(q_1, b, \rightarrow)$
q_0	b	$(q_1, a, \rightarrow)$
q_0	$\square$	$(h, \square, \rightarrow)$
q_0	c	$(q_0, c, \rightarrow)$
q_1	a	$(q_0, c, \rightarrow)$
q_1	b	$(q_0, b, \leftarrow)$
q_1	$\square$	$(q_0, c, \rightarrow)$
q_1	c	$(q_1, c, \rightarrow)$

Solution.

state/symbol	representation
q_0	q00
q_1	q01
h	q10
$\square$	a00
a	a01
b	a10
c	a11

(q00, a01, q01, a10, →), (q00, a10, q01, a01, →), (q00, a00, q10, a00, →), (q00, a11, q00, a11, →), (q01, a01, q00, a11, →), (q01, a10, q00, a10, ←), (q01, a00, q00, a11, →), (q01, a11, q01, a11, →) $\square$

7) Churchs Thesis makes the claim that all reasonable formal models of computation are equivalent. And we showed in, Section 17.4, a construction that proved that a simple accumulator/register machine can be implemented as a Turing machine. By extending that construction, we can show that any computer can be implemented as a Turing machine. So the existence of a decision procedure (stated in any notation that makes the algorithm clear) to answer a question means that the question is decidable by a Turing machine.

Now suppose that we take an arbitrary question for which a decision procedure exists. If the question can be reformulated as a language, then the language will be in D iff there exists a decision procedure to answer the question. For each of the following problems, your answers should be a precise description of an algorithm. It need not be the description of a Turing Machine:

(a) * Let $L = \{\langle M \rangle : M$ is a DFSM that doesnt accept any string containing an odd number of 1s$\}$. Show that L is in D.

Solution. □

(b) Consider the problem of testing whether a DFSM and a regular expression are equivalent. Express this problem as a language and show that it is in D.

Solution. Let $L = \{\langle M \rangle \#\beta : \langle M \rangle$ is a representation of a DFSM and β is a regular expression.$\}$. To recognize L, use the following algorithm:

1. From $\langle M \rangle$, construct α, a regular expression such that $L(M) = L(\alpha)$.
2. Create α' and β', minimized versions of α and β respectively.
3. If α' equals β', accept. Otherwise reject.

□

8) Consider the language $L = \{w = xy : x, y \in \{a, b\}^*$ and y is identical to x except that each character is duplicated$\}$. For example `ababaabbaabb` $\in L$.

(a) * Show that L is not context-free.

Solution. □

(b) Show a Post system that generates L.

Solution.

$$\begin{aligned} S &\to L\# \\ L &\to aL \mid bL \mid \epsilon \\ AaB\#R &\to AaB\#Raa \\ A\#B &\to AB \end{aligned}$$

□

9) Consider the language $L = \{\langle M\rangle : M \text{ accepts at least two strings}\}$.

(a) Describe in clear English a Turing machine M that semidecides L.

Solution. Simulate 2 instances of the universal turing machine, one with input $\langle M', w_0\rangle$ and the other with input $\langle M', w_1\rangle$ where $\langle M'\rangle$ is input string and w_0 and w_1 are nondeterminstically chosen elements of $\Sigma^*_{M'}$. If both instances accept, accept. If one of them rejects, reject. □

(b) Suppose we changed the definition of L just a bit. We now consider:

$$L' = \{\langle M\rangle : M \text{ accepts exactly 2 strings}\}.$$

Can you tweak the Turing machine you described in part a to semidecide L'?

Solution. No, you could never exaust all possible input strings to the machine, so you wouldn't be able to say yes once you found two strings that were accepted. There could always be another. □