CS103 Handout 24

Spring 2012 June 4, 2012

Additional Practice Final #3

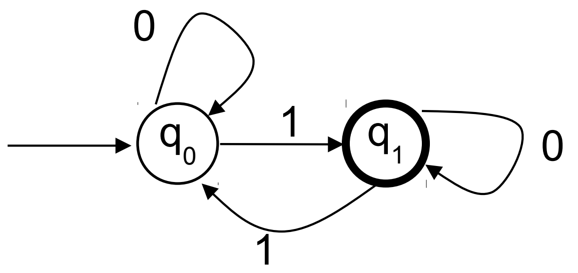
This additional practice final is an actual CS103 final exam from a previous quarter. The structure of this exam is not the same as the structure of the upcoming final exam, but the sorts of questions on it are similar to what you might expect to see on the actual exam.

This practice final is **not worth any extra credit points**, but should be a good resource to help you prepare for the exam.

Enjoy!

FINITE AUTOMATA

1. (16 points) Consider the following DFA:



(a) Describe the language accepted by this DFA. (The level of description we are looking for is something like "all strings of 0's and 1's with more 0's than 1's".)

(b) Prove <u>by induction</u> on the length of the input string that your description is correct. To be more precise, the proposition P_n for your induction will be that your description is correct for strings of length n. Show that P_n holds for $n \geq 0$.

2. (12 points) Prove that if L is a non-empty language such that every string in L has length at least n, then any DFA accepting L must have at least n + 1 states.

REGULAR EXPRESSIONS / REGULAR LANGUAGES / GRAMMARS

3. (8 points) State whether each of the following claims are true for all regular expressions r and s. The symbol ≡ stands for equivalence of regular expressions in the sense that both expressions denote the same language.

(a) $(r^*)^* \equiv r^*$ T or F: ____________

(b) $r^*(r \cup s)^* \equiv (r \cup s)^*$ T or F: ____________

(c) $(r \cup s)^* \equiv (r^*s^*)^*$ T or F: ____________

(d) $(rs)^* \equiv r^*s^*$ T or F: ____________

4. (10 points) The following grammar generates a regular language. Draw the state diagram for a DFA that accepts the language.

$S \rightarrow abA$
$A \rightarrow baB$
$B \rightarrow aA \mid bb$

5. (12 points) Use the Pumping Lemma to show that the following language L is not regular. We have laid out the proof for you, so use this form and fill in the blank spots.

The language definition uses the notation #a(w) to mean the number of a's in string w (for example, if w = abca, then #a(w) = 2 and #c(w) = 1).

$L = \{w \in \{a, b, c\}^* \mid 3\#a(w) = |w|\}$ (so one-third of the symbols in the strings are a's)

Examples of strings in L: abc, bca, aabcbc, bacabb.

The proof is by contradiction. Assume that L is ________________. Let p be the pumping length for L as given by the pumping lemma.

Consider the string w = _______________.

Clearly, w belongs to L, because ______________________________________.

By the Pumping Lemma, we can write w = xyz where for each $i \geq 0$,
$xy^iz \in L$, $|y| > 0$, ,and $|xy| \leq p$.

(the rest of the proof goes here)

Thus, we have a contradiction, and L is not regular.

TURING MACHINES/UNDECIDABILITY

6. (15 points) Prove that the following language is undecidable:

$L = \{ \langle M, w, a\rangle \mid$ M is a TM, $a \in \Gamma$, $w \in \Sigma^+$, and M writes the symbol a when given input w $\}$.

Γ is the tape alphabet for M. So the question is whether you can decide if a TM ever writes a particular symbol when given a particular input.

7. (12 points) Prove that the following language is undecidable:

L = { ⟨M⟩ | M is a TM and $\mathcal{L}$(M) contains two different strings of the same length}

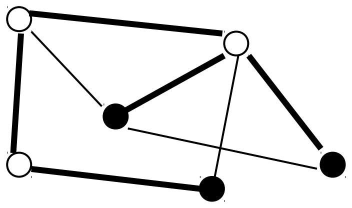
NP-COMPLETENESS

8. (15 points) In an undirected graph G, a spanning tree is a set T of edges such that

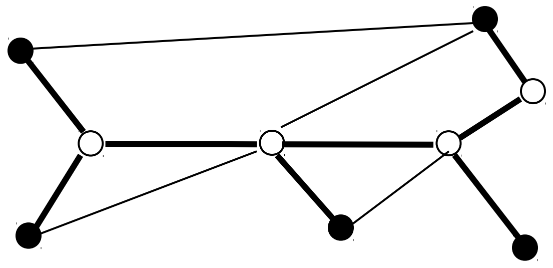
- T contains no cycles, and
- Every node in G is connected to every other node using only edges in T.

Every connected graph has at least one spanning tree.

For example, in this graph, the edges of a spanning tree are shown with heavy lines:



The three solid nodes in the spanning tree above are **leaf nodes**: only one edge in the spanning tree touches each of them (i.e., within the spanning tree, leaf nodes have degree one). As another example, here is a graph and a spanning tree with 5 leaf nodes:



Now consider the following language:

STK = { ⟨G, k⟩ | G is a graph with a spanning tree with no more than k leaf nodes}

Assume that STK is in the class NP, and prove that it is NP-complete. (As you may recall from lecture, what you have to prove is called showing that the language is "NP-hard".)

Hint: do a reduction from UHAMPATH.