COMP 314 Homework Assignment E

This is version 3 of the assignment, published 2/18/15. Questions E7 and E14 were replaced with completely different questions. Some point values were changed.

Chapter 8

Question E1. (15 points) Consider the decision problem HasMultipleOfK defined as follows:

Problem HasMultipleOfK

- **Input:** A single ASCII string containing a positive integer K, followed by a semicolon, followed by a list of positive integers $m_1, m_2, \ldots$. Each m_i and K is represented in decimal notation, and the m_i are separated by whitespace. Example: "`823;18910 5235 3422`"
- **Solution:** "`yes`" if any of the m_i is a multiple of K; and "`no`" otherwise. Example: for the input given above, the solution is "`no`", because none of 18910, 5235, 3422 is divisible by 823.

Write a Python program that solves HasMultipleOfK nondeterministically, using a separate thread for each integer m_i in the input.

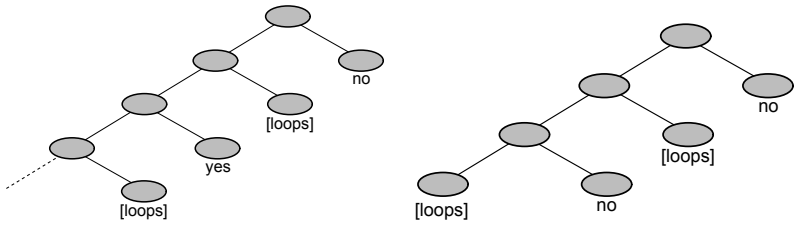
Question E2. (10 points) The problem FindMultipleOfK, defined below, is very similar to HasMultipleOfK. However, it is a general computational problem rather than a decision problem:

Problem FindMultipleOfK

- **Input:** A single ASCII string containing a positive integer K, followed by a semicolon, followed by a list of positive integers $m_1, m_2, \ldots$. Each m_i and K represented in decimal notation, and the m_i are separated by whitespace. Example: "`823;18910 5235 3422`"
- **Solution:** Any value m_i that is a multiple of K is a solution. If none of the m_i is a multiple of K, the solution is "`no`". Examples: for the input given above, the solution is "`no`", because none of 18910, 5235, 3422 is divisible by 823. For the input "`10;720 342 90`", the solution set is {"`720`", "`90`"}.

Write a Python program that solves FindMultipleOfK nondeterministically, using a separate thread for each integer m_i in the input.

Question E3. (6 points) For each of the following computation trees, give a valid output of the computation, or alternatively state that the computation is undefined.



(a) (b)

Question E4. (15 points) Create a nondeterministic Turing machine that searches genetic strings for one of the following three patterns: "`CG...GC`", "`CA...AC`", or "`CT...TC`". Here, the "`...`" represents any string of zero or more characters. You may assume the input is flanked by "`x`" characters. So, your machine should accept inputs like "`xGGTTCGAAAGGCAAx`" or "`xCTTCx`" and reject inputs like "`xCCCAGCx`" or "`xCTCx`".

It is reasonably easy to construct a deterministic Turing machine solving the above problem, but the intention of this question is to practice using nondeterminism. Therefore, your machine should spawn a separate clone for each of the three search patterns.

Question E5. (Ungraded) Would a multicore quantum computer be Turing-equivalent to a standard Turing machine? Justify your answer.

Chapter 9

Question E6. (10 points) Construct a dfa that accepts ASCII strings containing exactly two or five `Z` characters.

Question E7. (10 points) Construct an nfa that accepts the language $\{\texttt{C}^n : n \text{ is a multiple 3, 5, or 7}\}$.

Question E8. (Ungraded) Construct a dfa that recognizes the language

$$L = \{\texttt{abc}, \texttt{abcde}\} \cup \{(\texttt{xy})^n : n \text{ is divisible by 3}\}$$

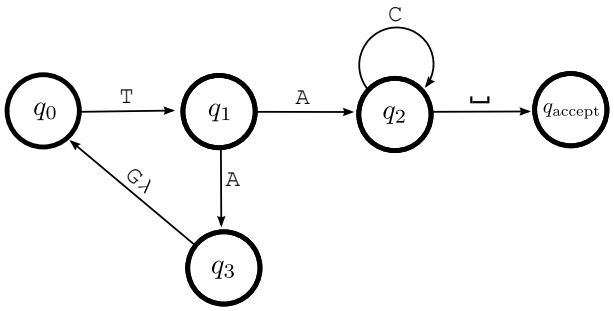
Question E9. (Ungraded) Let's agree to describe nfas and dfas using ASCII strings in a format similar to that of Figure 5.18 (page 116). Specifically, our format will be the same as Figure 5.18, but on every line we omit the semicolon and everything following it. (This works because we don't need a new tape symbol or head direction.) Assuming these conventions, give a plain-English description of the language accepted by the following dfa:

```
q0->q0: 0369
q0->q1: 147
q0->q2: 258
q0->qA: _
q1->q0: 258
q1->q1: 369
q1->q2: 147
q2->q0: 147
q2->q1: 258
q2->q2: 369
```

Question E10. (Ungraded) Using the provided `simulateTM.py` as a starting point, create a Python program `simulateDFA.py`. This program takes as input two parameters P, I, where: P is the ASCII description of a dfa (formatted as in the previous question), and I is an ASCII input string. The output is "`yes`" if the dfa accepts I, and "`no`" if it rejects I.

Question E11. (Ungraded) Construct two distinct, but equivalent, nfas that accept the language of genetic strings containing `GAGA` or `GTGT`.

Question E12. (15 points) Use the algorithm described in Section 9.3 to convert the following nfa into a dfa. You must label your dfa states using the convention of Figure 9.6 (page 185).

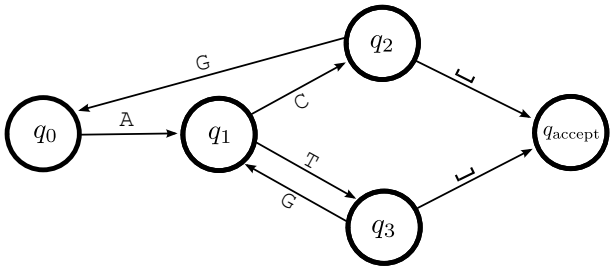


Question E13. (Ungraded) Take $\Sigma = \{\texttt{C}, \texttt{A}, \texttt{G}, \texttt{T}\}$. Prove that the language of all strings on Σ containing between 3 and 5 `G`'s (inclusive) is regular.

Question E14. (5 points) Again take $\Sigma = \{\texttt{C}, \texttt{A}, \texttt{G}, \texttt{T}\}$. Find a regular expression for the language of all strings on Σ containing an even number of occurrences of the substring "`TCA`", each separated by zero or more `G` characters. For example, the following strings are in this language: λ, "`TCATCA`", "`GGGTCAGGTCAGGGG`", "`TCAGGGTCAGGTCATCAGGGG`".

Question E15. (10 points) Find an nfa equivalent to the regexp `(T(GGA)*|C)*`.

Question E16. (Ungraded) Convert the following dfa into an equivalent regexp.



Question E17. (Ungraded) Read the documentation for the standard regular expression library in Python. (The library is called `re`.) Use this library to implement one or more useful Python programs. For example, produce programs that decide whether the ASCII input: (i) is alphanumeric; (ii) is a genetic string; (iii) contains the substring `GAGA` and the substring `CTTCC`, in that order; (iv) matches the regular expression `(AA(CG)*(T*))*|GGT`; (v) contains the definition of a Python function.

Question E18. (15 points) Use the Pumping Lemma to prove that the following language is not regular: $\{\texttt{C}^n\texttt{TTC}^m : m > n + 5\}$.

Question E19. (15 points) Use the Pumping Lemma to prove that the following language is not regular: the set of all ASCII strings that mention US states more often than countries in the EU. For example, "`Alabama Georgia Alabama Spain`" is in the language (3 US states, 1 EU country), but "`SpainAlabama 5GeorgiaAlabama23452cc Spain Portugal ItalyPennsylvania`" is not in the language (4 US states, 4 EU countries). Important hint: you'll probably need to use the fact that the intersection of two regular languages is also regular. You may use this fact without proving it.

Total points on this assignment: 126