MA/CSSE 474 **Homework #14** 60 points **Updated for Summer, 2018.**

There are lots of Q&A from previous term's Piazza at the end of this document.

1. 13.13
2. (t-6) 13.13d
3. (t-6-9) 13.14 Examples: If L is the language denoted by r.e. (ab)*, then alt(L) is denoted by r.e. a*.
   If L is the language denoted by r.e. (abcdefg)*,
   then alt(L) is denoted by r.e. (aceg)* .
   If L is the language denoted by r.e. a or the language denoted by r.e. a*,
   then alt(L) is L.
4. (t-6) 14.1a
5. (t-9) 14.1c
6. 14.1d
7. 14.1e
8. (t-6) 17.1a Don't just describe in English the *individual steps* the machine makes. Your answer should be a global one: in particular, describe how the final tape content is different from the original tape content.
9. 17.1b
10. (t-9) 17.3a It is worth the time it will take to learn the author's macro language for TMs. Use that language to express your answer.
11. Other parts of 17.3
12. (t-9) 17.4
13. (t-3) 17.6

Some questions and answers from previous term's Piazza discussions:

Problem 2 hint

A student came to my office to ask about this problem, and I could not immediately see how to do it. I knew it would involve adding lots of epsilon transitions for the "skipped-over" parts y and z, but what happens to the stack when you skip transitions that would have put things on the stack?

Here is the trick. Don't just work with the original machine. Make two copies of the original PDA M,
In each transition of each copy, replace the input symbol of that transition (if there is one) by epsilon.
Leave the stack parts as they are. The first copy machine deals with the y part, simulating what M would have done to the stack while reading y, without actually reading any input. the second copy does the same thing for z.

So now I have told you almost everything you need for this problem. The remaining question is how to connect the three machines into one machine that accepts exactly middle(L(M)).

Two more hints for Problem 2 (the "closed under middle" problem

1. What you need to do: Come up with a construction which, given any PDA M, produces a PDA M' such that L(M') = middle(L(M)).

2. **Suggestion:** As a warm-up, show that the regular languages are closed under *middle* by doing the same thing as above for DFSMs. This will show you how to handle states in your PDA construction. Then use my hint from yesterday as a starting point for how to handle the stack.

#3 (134) :

Problem 3: Proving that CFLs are closed under alt

Q: I'm pretty sure that CFGs are closed under alt, but I'm having some trouble proving that it is. At first I was thinking to combine pairs of transitions into one transitions, but logic for that gets incredibly messy for complicated machines...

Am I on the right track of "combining" transitions to produce a machine that accepts alt(L)? Or perhaps I'm completely wrong and it's actually not context free (I don't think this is the case though...) Any hints would be appreciated.

A: I don't think you can "combine" states, but you can have a similar effect by adding epsilon transitions that "skip" states.

#5 (14.1c) :

Create decision procedure that, given a CFG, generates at least 3 strings

Q: Are there any "free" functions we may use? (kinda like the minDFSM() and ndfsmtodfsm() algorithms for state machines)

A: You can assume and use any algorithm or decision procedure that we have done in class or homework or that is in the book.

#10 (17.3a) : can we assume there will be no 0 entry in the most significant bit?

A: If you wish, you may assume that the representation of a positive integer does not begin with 0. But the representation of the number zero has to begin with 0.

General question about Turing machines: Is the input to a Turing machine a finite tape? That is, if we reach the end of the input, does it halt, or do we assume that the input is nested between infinite blank squares?

A: The tape is infinite in both directions. When the computation starts, there are finitely many non-blank symbols on the tape (which implies that there are always finitely many non-blank symbols on the tape).

Unless specified otherwise for a particular TM, the read/write head is initially positioned at the blank symbol before the leftmost non-blank symbol in the input.

For the special case where the input is the empty string, the entire tape is blank, so all tape squares are initially equivalent and indistinguishable, so it does not matter where the head begins.

Follow-up to the previous question: so in a loop that starts with a >, does that mean we go back to the blank before the input string? #12 (17.4) : Constructing TM for unary encoding of max(#a(x), #b(x))

A: The > in the macro diagrams is simply a "this is where we start" marker, so I don't think there ever is a "loop containing >". For example, in the machine in example 17.7 (page 375), the > tells where to start, but it is not part of the loop. R is the first thing that gets done each time through the loop.

If you want your TM to "go back to the last blank before the input", you must place an explicit L_{blank} (or in some cases more than one of them, if there are blanks in the middle of the current string on the tape) in your diagram.