Homework 05

Due 2/27/2019

Purpose:

The purpose of this homework assignment is to help you deepen your understanding of context free languages and push-down automata.

Like last week I want you to turn in a file Hmwk5.pdf with all the answers to the homework. There are no Haskell programs due this week (though for the last problem you will be providing some answers).

Tasks:

**Problems from the texts are given in the form c.n(k) where c is the chapter and n is the problem number, and k is the sub-problem number. Thus problem 2.7(d) is problem 7(d) from Chapter 2.*

1. (0 points) **Academic Honesty Statement**
2. (15 points) **Context-free Language**

   Let L = $\{w \in \{a,b\}^* | w = w^{rev}\}$.

   (a) Design a cfg generating L.

   (b) Design a pda accepting L.

   (c) Design a cfg generating the complement of L

   Notice that this language is slightly different from the language L' = $\{ww^{rev} | w \in \{a,b\}^*\}$. In particular the new language also contains strings of odd length.
3. (10 points) **Ambiguity and Parse Trees**

   Rich 11.10
4. (10 points) **CFL to PDA**

   Use the construction in the text or lecture to create a pda that accepts the language generated by the grammar for arithmetic expressions with V = {Exp, Addop, Term, Factor, Mulop, num, *, +, -, /, (,) }, Σ = {num, *, +, -, /, (,) }, start symbol Exp, and the following rules:

   - Exp → Exp Addop Term | Term
   - Term → Term Mulop Factor | Factor
   - Factor → (Exp) | num
   - Addop → + | -
   - Mulop → * | /

   Be sure to use the given construction. We will see later that it is undecidable for two pda's M and M' whether L(M) = L(M'). The TA's get irritated if they have to solve undecidable problems in order to grade the homework!!
5. (20 points) **Regular expressions in Haskell**

   In this problem I would like you to explore our implementation of regular expressions in Haskell as presented in class.[1]

[1] In the code presented in class, the function star had an error making it accept too many strings. The on line version is now corrected.

(a) We begin by exploring the DFSM generated by simple regular expressions. The simplest way to experiment is to place the file NDFSMez.hs in the same directory as your own Haskell code, and include

```
import NDFSMez
```

early in your code.

Define simp = cat (once 'h') (once 'e'). That is simp is the regular expression that is the concatenation of regular expressions 'h' and 'e'. It describes the set {"*he*"}.

The value of simp is a term of type Regex Char. Because these terms are records, we can ask for the components by name. E.g., we can evaluate numberOfStates simp, startingStates simp, acceptingStates simp, and transitionFunction simp. The first three can be evaluated and printed. Unfortunately the transition function is a function and can't be printed. However, you can ask for the results of applying it to arguments. For example, if simpTf is the name of the transition function, then simpTf 'h' 0 would give the result `fromList [1,2]`.

For this first part of the problem, provide for simp the number of states, the starting states, the accepting states, and the results of applying the transition function on all 4 of the states for the inputs 'h', 'e', and 'a'. Once you have done this, please draw a picture of the DFSM given by simp.

(b) Do the same thing as above, but this time with the regular expression dble = cat simp simp. Again, draw the picture of the DFSM. Do your best to describe how the new DFSM was constructed from simp. Look up the construction we performed in class to get the concatenation of two regular languages by creating an NDFSM combining the DFSM's of the starting languages. Because we are going directly to a DFSM, there will be differences, but the ideas should be the same.

(c) Do the same thing again with starsimp = star simp. Again do your best to explain how the new DFSM was constructed from simp by looking at the construction we did in class of the Kleene * of a language.

*Note that the class constructions of concatenation and Kleene * created NDFSM's while we are creating DFSM's directly. Thus the analogy will not be exact!*

Criteria:

Your assignment will be graded based on the correctness and the completeness of your solutions. Please make sure that you use the correct terminologies when you write proofs or describe a structure, such as CFG, PDA, etc.

Submission Guideline:

Please edit your HW following the editing guideline, especially the instructions on the number of pages each question should occupy. The text in blue are instructions; they should be either removed, or replaced with proper contents and turned into black. Please submit your homework solutions online via gradescope.