ToC: Homework 4 [150 points]

Due: Saturday, October 30th, by midnight

Reading

Please read Lectures 21, 23, 24, 25, and F of Kozen.

How to submit

Please put your solution in a folder whose name is your Hawkid. If you worked with a partner please include a file called `partner.txt` that lists the Hawkids for you and your partner. Only one partner should submit their assignment. You will both receive the same grade.

How to get help

Please post questions to ICON Discussions, if possible (as opposed to emailing us directly), so that our answers can benefit others who likely have similar questions. You may also come to our Zoom office hours, listed on ICON under Pages - Office Hours.

1 Pushdown automata in set theory [65 points]

The following problems concern the PDA whose transition diagram is shown in Figure 1. You will need to refer to the definitions for PDAs that I gave in class the week of Oct. 4th (not the definition in Kozen). There, I defined the set $\hat{\Gamma}$ as the set of lists of stack actions (each stack action is a push, pop, or noop over stack alphabet Γ) that normalize to a list of push actions. In other words, elements of $\hat{\Gamma}$ are sequences of stack actions where all the pops succeed, because there is always the required symbol on the top of the stack.

1. Write out the set-theoretic description of the PDA (again, as I gave in class, not from Kozen). [15 points]
2. There is exactly one element g of $\hat{\Gamma}$ such that $(0, g, 3) \in R^E[b \to b]$ (thus showing that the string $b \to b$ is in the language of the PDA, as 0 is the start state and 3 the only final state). Please find this g and show how it normalizes using the rewriting rules for $\rightsquigarrow$ that I gave:

$$+X - X \quad \rightsquigarrow \quad \cdot \tag{1}$$

$$\cdot x \rightsquigarrow \quad x \tag{2}$$

$$x \cdot \rightsquigarrow \quad x \tag{3}$$

[20 points]

3. The PDA corresponds (via the translation from CFGs to PDAs that I covered in class Oct. 12th) to a CFG with three productions, and nonterminals E and A. Write out the productions of this CFG. [15 points]

4. Give an example of a string that is accepted by this PDA with two different values of $\hat{\Gamma}$ for the sequence of stack actions. (Hint: this amounts to finding a string which has two different production trees in the grammar you identified for the previous problem.) Show those values of $\hat{\Gamma}$. [15 points]

2 Pushdown automata in Haskell [25 points]

Consider the following context-free grammar, where P is the start symbol:

$$E \rightarrow \texttt{(}\ L\ \texttt{)} \tag{4}$$
$$E \rightarrow \texttt{x} \tag{5}$$
$$L \rightarrow L\ E \tag{6}$$
$$L \rightarrow E\ E \tag{7}$$

For the following problems, you can put your answers in a file called `Applic.hs` (because the grammar describes an applicative syntax for function calls, where you write $x\ x$ instead of $x(x)$).

1. Give two examples strings in the language of this grammar. [5 points]

2. Using the construction described in class Oct. 12th, define a `Pda` to recognize the language of that grammar. My `Pda.hs` code is working now to test (in finite time) whether strings are accepted or rejected. So you should be able to test your PDA on some examples (like the ones you think of for the first problem), to make sure you have it right. [20 points]

3 Transforming grammars to Chomsky Normal Form, in Haskell [60 points]

In `CfgExamples.hs`, there is a grammar `lst`. The problems below make use of this. Please put your solution in a file called `Transform.hs`. The goal of this problem is to explore the transformation to Chomsky normal form defined in Kozen Lecture 21. [10 points for each problem below]

1. Write out the productions for the grammar in the usual form. For example, the first one would be written

   ```
   L -> [ S ]
   ```

   You can just write these out in a comment at the top of your `Transform.hs` file.

2. Define a grammar `lst1` where you remove the fifth production (`R ->` , an empty production) by carrying out the transformation described using (a) on page 141 of Kozen.

3. Now transform `lst1` by adding new nonterminals for all the terminals, as described at the top of page 143. To make grading easier, please use `C` as the nonterminal for comma, `LB` as the nonterminal for left square brace, and `RB` as the nonterminal for right square brace. Call the resulting grammar `lst2`.

4. The previous two steps have gotten you close to Chomsky Normal Form. The final transformation is to introduce new nonterminals so we can break up the right-hand sides of two productions that should, at this point, have three nonterminals. Please introduce the nonterminals `N1` and `N2` for those productions, where `N1` is introduced for the production with `L` on the left-hand side. Call the final grammar `lst3`.

5. Write out a legal production tree (`ProdTree`) with respect to `lst3` called `pt` for the string

   ```
   simple :: [String]
   simple = promote "[8,8]"
   ```

6. Use the function `pump` from `CfgPumping.hs` to apply my code for the pumping lemma for context-free languages to break up the production tree you wrote for the previous problem, and pump in 2 copies of the middle part of the tree. Use `writeIf` to save the resulting `ProdTree` to a file (`output.gv`), which you should render to `output.pdf` using `GraphViz`.

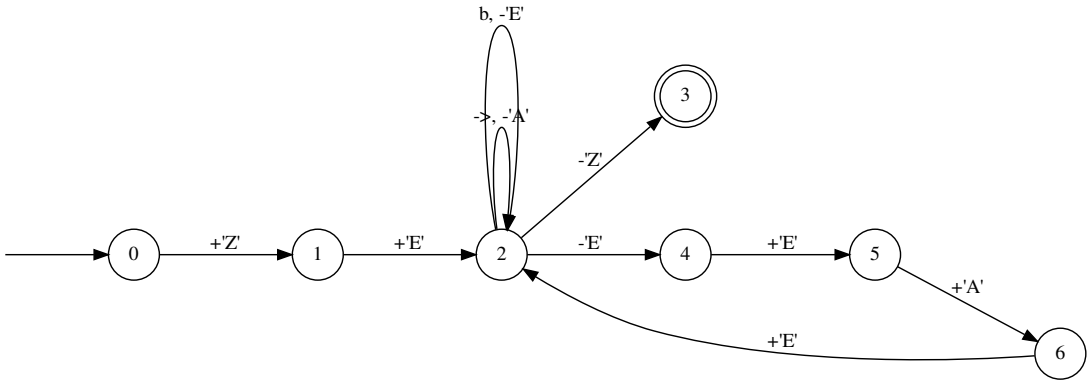

Figure 1: PDA for Section 2 (note that the arrow symbol $\rightarrow$ has been rendered as `->` due to limitations of GraphViz)