CS 164 Programming Languages and Compilers **Spring 2019**

Discussion Week of 3/11: Midterm Prepreparation

Instructor: Paul N. Hilfinger GSIs: Nikhil Athreya, Vivant Sakore

1 Regular Expressions

1. (a) Write a regex that matches binary strings divisible by 8.

   (b) Provide a *regular* grammar for the regex from part (a).

2 Finite State Automata

1. (a) Write the corresponding NFA for the regular expression `(0|1)?(10|01)+`;

   (b) Convert the NFA from part (a) into a DFA.

3 Grammar Rewriting and LL(k) Parsing

Consider the simple ambiguous grammar (which we've seen before):

$$
\begin{array}{rcl}
S & : & E \dashv \\
E & : & E + E \\
 & | & E * E \\
 & | & \texttt{ID}
\end{array}
$$

1. Show that the grammar is ambiguous with two different leftmost derivations of the string `a+b*c`;

2. Rewrite this grammar so that it preserves the standard order of operations, is LL(1), and is unambiguous. Draw the resulting tree for the string `a+b*c`;

3. Write down the equivalent unambiguous grammar that enforces both **left** associativity and correct precedence. Why can't this be achieved with an LL(1) grammar?

4 Earley's Algorithm

Consider the following grammar:

$$
\begin{array}{lcl}
P & : & E \dashv \\
E & : & ID \\
 & | & \lambda ID \,.\, E \\
 & | & E\ E
\end{array}
$$

1. Use it to parse the following expression with Earley's algorithm: λ`ID . ID ID` $\dashv$

	λ	`ID`	`.`	`ID`	`ID`	
	0	1	2	3	4	5
a						
b						
c						
d						
e						
f						
g						
h						
i						
j						

2. Is the expression in the language? Is the grammar ambiguous?