Written Assignment 3

This assignment asks you to prepare written answers to questions on context-free grammars and Earley parsers. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work.

Please print your name and UVA ID on your homework! We need this information so that we can give you credit for the assignment and so that we can return it to you.

1. Use left-factoring and/or elimination of left recursion to convert the following grammars into LL(1) grammars. You may assume that these grammars are unambiguous.

   (a)

$$\begin{array}{lcl} E & \rightarrow & E+T \mid T \mid E! \\ T & \rightarrow & int \mid (E) \end{array}$$

   (b)

$$\begin{array}{lcl} L & \rightarrow & X \mid L, X \\ X & \rightarrow & int \mid string \mid (L) \end{array}$$

   (c)

$$\begin{array}{lcl} P & \rightarrow & P\ H\ 4\ U \mid p \\ H & \rightarrow & h \\ U & \rightarrow & u \mid u\ P \end{array}$$

2. Consider the following grammar:

$$\begin{array}{lcl} S & \rightarrow & A \\ A & \rightarrow & A + A \mid B + + \\ B & \rightarrow & y \end{array}$$

(Each '+' is a separate token.)

Draw the full Earley chart associated with parsing the input string `y + + + y + + +` using the above grammar and indicate whether or not the Earley parser accepts the string.

3. In class we discussed how how to use Earley's algorithm to *recognize* if a string is in the language of a grammar. However, modern parsers can do more than recognize: they can also produce parse trees, produce derivations, or otherwise execute user-defined actions (as in `yacc` or PA3) whenever a rewrite rule is applied.

   Explain in English prose how you would modify an Earley recognizer (such as the one developed in the class notes and handouts) into a full-fledged parser that executes user actions. Do not provide code diffs or deltas. Instead, describe what additional information must be maintained and how that information should be updated. Remember that user actions should be executed bottom-up even if the parsing algorithm is top-down. Remember also that user actions can read values associated with previously-executed rewrite rules (e.g., `$$ = new PlusNode($1,$3);` or `return new PlusNode(val[1], val[3]);` or the like). Be precise: make it clear to the reader what data structures you are adding or changing, how those data structures are updated, what invariants are maintained, and when and in what order the user actions are executed.

4. Consider an Earley parser, such as the one we discussed in class, with the *closure* (or *predict*) operation removed. That is, this new type of Parser only uses *shift* (or *scan*) and *reduction* (or *completion*) operations to fill in the chart of parsing states.

   - Give a grammar G_1 and a string S_1 such that $S_1 \in L(G_1)$ but this new closure-less parser fails to recognize that S_1 is in $L(G_1)$. Explain why the parse fails in one sentence.

- Give a grammar G_2 and a string S_2 such that $S_2 \in L(G_1)$ and this new closure-less parser still successfully recognizes that S_2 is in $L(G_2)$.