A Left-Corner Parser

The *left corner* of a context-free rule is the first symbol on the right hand side; for example, Det is the left corner of the rule NP $\rightarrow$ Det Adj N. A *left corner parser* can be characterized as follows, using our conventions for stack-based parsers. (For your reference, the top-down and bottom-up (shift-reduce) parsers are described using those conventions in an appendix.)

- Start Configuration: $\langle\, \epsilon, 0 \,\rangle$
- "Plain English" description of operations:
  - Shift.
    If the next word in the input is a, and $A \rightarrow a$ is a rule in the grammar, then read in a, and push $A \rightarrow a\,\bullet$ onto the top of the stack.
  - LC-Predict (left-corner prediction, LCP for short).
    If $A \rightarrow \beta\,\bullet$ is on top of the stack, and the grammar contains a rule X $\rightarrow A\beta$, then replace the top item on the stack with X $\rightarrow A \;\bullet\; \beta$
  - Complete.
    If the top stack item is Z $\rightarrow \beta\,\bullet$ and the next item down is X $\rightarrow \alpha \;\bullet\;$ Zγ, then replace those top two items with X $\rightarrow \alpha$Z $\bullet\; \gamma$.
- Successful Configuration: See question (b), below.

For purposes of this question, you can assume that in any grammar, terminal symbols are *only* introduced by rules of the form $A \rightarrow a$. (This condition can be met for any CFG with only trivial modifications.)

(a) Express each of the Shift, LC-Predict, and Complete operations as an inference rule of the form

$$\frac{\text{configuration before this rule has applied}}{\text{configuration after this rule has applied}}$$

(along with any conditions on the application of the rule).

(b) What should be filled in above for "Successful Configuration"?

(c) In the LC-Predict operation, why is the new dotted rule X $\rightarrow A \;\bullet\; \beta$ and not X $\rightarrow \;\bullet\, A\beta$?

(d) Consider the following grammar.

```
S    -> S Adv           Det  -> a | the
S    -> Aux NP VP       N    -> students | assignment | lectures
S    -> NP VP           Pron -> he | she | they
NP   -> Det N           V    -> enjoy | enjoyed | detested
NP   -> Pron            Adv  -> immensely | amazingly | recently
VP   -> V NP            Aux  -> could | would
VP   -> V
VP   -> Adv VP
```

Show the sequence of left-corner parser configurations for a successful parse of *they immensely enjoyed lectures* that assigns the structure

$$[_S\,[_{NP}\,[_{Pron}\textit{they}]]\,[_{VP}\,[_{Adv}\textit{immensely}]\,[_{VP}\,[_V\textit{enjoyed}]\,[_{NP}\,[_N\textit{lectures}]]]]]$$

Only show the configurations along the successful search path. Use this format: a table with two columns, where the first column is the Operation, and the second column is the Current Configuration. The first row in the table is:

```
Operation                 Current Configuration
-----------------------------------------------
Initial configuration     <epsilon,0>
```

(e) Compare this parser to the top-down and bottom-up (shift-reduce) parsers. How is it similar to each and how is it different? (Hint: if you look at each parser operation as updating an incomplete derivation of the final parse tree, like Jurafsky and Martin did when illustrating top-down and bottom-up parsing, then it's easier to see, and to illustrate, what the left corner parser is doing.)

(f) Identify one disadvantage of the top-down parser and how the left-corner parser avoids it.

(g) **Extra Credit (10%)** Implement the above left-corner parser in the "parsing as search" framework, i.e. using an agenda. For consistency assume depth first search. Turn in output from running the parser as your answer to part (d) instead of doing part (d) by hand. Turn in the code separately (electronically, not hardcopy).

Appendix

1 Conventions

- We number the positions for input `a b c ... x` as follows: ${}_0\, a\, {}_1\, b\, {}_2\, c \ldots x\, {}_n$.
- Stacks are represented with the top of a stack at the left.
- Greek letters stand for sequences of zero or more symbols. For example, a stack `VP` Γ has a VP at the top and we don't care what's below it. NP $\rightarrow$ Det β could represent any NP rule in which Det is the first symbol on the right hand side.
- Epsilon (ϵ) represents an empty stack.
- A *configuration* (or search state) is an ordered pair $\langle\ \Gamma,\ i\ \rangle$, where Γ is a stack and i is a position in the input.

2 Top-down (recursive descent) parsing

Start Configuration $\langle$ S, 0 $\rangle$

Operations:

- Apply-Rule (sometimes also called Top-Down Prediction)

$$\frac{\langle\ \mathrm{A}\ \Gamma,\ i\ \rangle}{\langle\ X_1 X_2 \cdots X_k\ \Gamma,\ i\ \rangle}$$

  where $A \rightarrow X_1 X_2 \cdots X_k$ is a rule of the grammar.

- Match

$$\frac{\langle\ \mathrm{A}\ \Gamma,\ i\ \rangle}{\langle\ \Gamma,\ i+1\ \rangle}$$

  if the word to the right of position i has part of speech `A`.

Successful Configuration: $\langle\ \epsilon,\ n\ \rangle$

3 Bottom-up (shift-reduce) parsing

Start Configuration $\langle\ \epsilon,\ 0\ \rangle$

Operations:

- Shift

$$\frac{\langle\ \Gamma,\ i\ \rangle}{\langle\ \mathrm{A}\ \Gamma,\ i+1\ \rangle}$$

  if the word to the right of position i has part of speech `A`.

- Reduce

$$\frac{\langle\ X_k \dots X_2 X_1 \Gamma,\ i\ \rangle}{\langle\ \mathrm{A}\ \Gamma,\ i\ \rangle}$$

  if $A \rightarrow X_1 X_2 \dots X_k$ is a rule of the grammar.

Successful Configuration: $\langle\ \mathrm{S},\ n\ \rangle$