

1. Here is a grammar for part of C expressions:

$$E \rightarrow id \mid E ? E : E \mid E \ll E$$

This grammar is ambiguous.

(a) Give a sentence that has two distinct parse trees, and show those parse trees.

(b) Explain how you might use ocaml yacc associativity declarations to resolve the conflict caused by this ambiguity. Be specific about what declarations you would use and what their effect would be. (The actual rules for resolving this ambiguity in C or Java do not matter; you just need to use the rules to resolve it in some way.)

2. Consider this grammar:

$$\begin{aligned} F &\rightarrow id (A) \\ A &\rightarrow \varepsilon \mid B \\ B &\rightarrow id id \mid B, id id \end{aligned}$$

(a) Show a parse tree for the following sentence: $f(C x, D y)$

(b) Write down all the sentential forms in that parse tree.

(c) Calculate $FIRST(F)$, $FIRST(A)$, and $FIRST(B)$.

(d) Calculate $FOLLOW(F)$, $FOLLOW(A)$, and $FOLLOW(B)$.

(e) Why is this not an LL(1) grammar?

(f) Transform it to an equivalent LL(1) grammar, and write a recursive descent parser for it. Assume the token type is

$$\text{type token} = \text{Id} \mid \text{LParen} \mid \text{RParen} \mid \text{Comma}$$

and the `parseF` function has type

$$\text{parseF: token list} \rightarrow (\text{token list}) \text{ option}$$

where `option` is the type defined as: `type 'a option = Some 'a | None.`

3. Consider this grammar:

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * id \mid id \end{aligned}$$

Show a complete shift-reduce parse for the sentence $x+y*z$. Include columns Action, Stack, and Input, as we did in class. (Warning: this may take more than one piece of paper, depending upon how big you write. There are 11 steps, counting the final "Accept" step.)

4. This grammar is not LR(1). It has a shift-reduce conflict:

A \rightarrow B , \$
B \rightarrow id | id , B

Show parse trees for two sentences with the following property: they lead to the same stack configuration (meaning, the roots of the trees on the stack or the same), with the same lookahead symbol, but in one case the correct action is the shift and in the other it is to reduce. Show that stack configuration.