

## Sample Questions for Midterm 2 (CS 421 Fall 2008)

On the actual midterm, you will have plenty of space to put your answers.  
Some of these questions may be reused for the exam.

- Using the rules provided in class, derive a valid type judgment for  
**let rec fact = fun n -> if n = 0 then 1 else let r = fact (n - 1) in n \* r in fact;;**  
(The rules will be provided for you on the exam, if this kind of question is asked.)
- Give a (most general) unifier for the following unification instance. Capital letters denote variables of unification. Show your work by listing the operation performed in each step of the unification and the result of that step.  
 $\{X = f(g(x),W), h(y) = Y, f(Z,x) = f(Y,W)\}$

- For each of the regular expressions below (over the alphabet  $\{a,b,c\}$ ), draw a non-deterministic finite state automaton that accepts exactly the same set of strings as the given regular expression.
  - $a^* \vee b^* \vee c^*$
  - $((aba \vee bab) c (aa \vee bb))^*$
  - $(a^*b^*)^*(c \vee \epsilon) (b^*a^*)^*$

- Consider the following grammar:

$$\begin{aligned} \langle S \rangle &::= \langle A \rangle \mid \langle A \rangle \langle S \rangle \\ \langle A \rangle &::= \langle Id \rangle \mid ( \langle B \rangle \\ \langle B \rangle &::= \langle Id \rangle ] \mid \langle Id \rangle \langle B \rangle \mid ( \langle B \rangle \\ \langle Id \rangle &::= 0 \mid 1 \end{aligned}$$

For each of the following strings, give a parse tree for the following expression as an  $\langle S \rangle$ , if one exists, or write “No parse” otherwise:

- $(0 \ 1 \ ( \ 1 \ ] \ ( \ ( \ 1 \ 0 \ ] \ 1$
  - $0 \ ( \ 1 \ 0 \ ( \ 1 \ ]$
  - $( \ 0 \ ( \ 1 \ 0 \ 1 \ ] \ 0 \ ]$
- Consider the following ambiguous grammar (Capitals are non-terminals, lowercase are terminals):

$$\begin{aligned} S &\rightarrow A \ a \ B \mid B \ a \ A \\ A &\rightarrow b \mid c \\ B &\rightarrow a \mid b \end{aligned}$$

Give an example of a string for which this grammar has two different parse trees, and give its parse trees.

- Write an unambiguous grammar generating the set of all strings over the alphabet  $\{0, 1, +, -\}$ , where  $+$  and  $-$  are infix operators which both associate to the left and such that  $+$  binds more tightly than  $-$ .
- Write a recursive descent parser for the following grammar:  
 $\langle S \rangle ::= \langle N \rangle \% \langle S \rangle \mid \langle N \rangle$   
 $\langle N \rangle ::= g \langle N \rangle \mid a \mid b$

You should include a datatype **token** of tokens input into the parser, one or more datatypes representing the parse trees produced by parsing (the abstract syntax trees), and the function(s) to produce the abstract syntax trees. Your parser should take a list of tokens as input and generate an abstract syntax tree corresponding to the parse of the input token list..

8. Why don't we ever get shift/shift conflicts in LR parsing?

Consider the following grammar with terminals **\***, **f**, **x**, and **y**, and **<eol>** for “end of line”, and non-terminals **S**, **E** and **N** and productions

- (P1) S => E <eol>
- (P2) E => E \* N
- (P3) E => N
- (P4) N => f N
- (P5) N => x
- (P6) N => y

The following are the Action and Goto tables generated by YACC for the above grammar:

| STATE | ACTION |    |    |    |     |     | GOTO |   |    |
|-------|--------|----|----|----|-----|-----|------|---|----|
|       | *      | f  | x  | y  | eol |     | S    | E | N  |
| 1     |        | s5 | s3 | s4 |     |     | 2    | 6 | 7  |
| 2     |        |    |    |    |     | acc |      |   |    |
| 3     | r5     | r5 | r5 | r5 | r5  | r5  |      |   |    |
| 4     | r6     | r6 | r6 | r6 | r6  | r6  |      |   |    |
| 5     |        | s5 | s3 | s4 |     |     |      |   | 8  |
| 6     | S9     |    |    |    | s10 |     |      |   |    |
| 7     | r3     | r3 | r3 | r3 | r3  | r3  |      |   |    |
| 8     | r4     | r4 | r4 | r4 | r4  | r4  |      |   |    |
| 9     |        | s5 | s3 | s4 |     |     |      |   | 11 |
| 10    | r1     | r1 | r1 | r1 | r1  | r1  |      |   |    |
| 11    | r2     | r2 | r2 | r2 | r2  | r2  |      |   |    |

where  $si$  is shift and stack state  $i$ ,  $rj$  is reduce using production  $Pj$ , and  $acc$  is accept. The blank cells should be considered as labeled with error. The empty “character” represents end of input.

Describe how the sentence **fx\*y<eol>** would be parsed with an LR parser using this table. For each step of the process, give the parser action (shift/reduce), input and stack state