Assignment 2

due November 30, 2012

Compiler Design 1 (Kompilatorteknik I) 2012

1 Context-Free Grammars

Give the definition of a context free grammar over the alphabet $\Sigma = \{a, b, c\}$ where the amount of a's is double the amount of b's. The amount of c's is of no interest.

2 Parsing and Semantic Actions

The following grammar roughly resembles the syntax of some complex builtin datatypes in Python. The terminals are $\{\alpha, (,), [,], :, , \}$ and the initial symbol is A

(for more information see `http://docs.python.org/tutorial/datastructures.html`)

$$
\begin{array}{rcl}
A & \rightarrow & D \mid L \mid T \mid \alpha \\
D & \rightarrow & \{K\} \mid \{\} \\
K & \rightarrow & A : A \mid K, A : A \\
T & \rightarrow & (A, I) \mid () \\
L & \rightarrow & [I] \\
I & \rightarrow & J \mid \epsilon \\
J & \rightarrow & J, A \mid A
\end{array}
$$

and the string $\{((), (\alpha,)) : [\alpha, \alpha], [] : \{\alpha : [\alpha, (\alpha, \alpha)]\}\}$

1. Give a leftmost derivation for the string.
2. Give a parse tree for the rightmost derivation of the string
3. Is the grammar ambiguous? Justify your answer.
4. Let us assume, that we use such a nested datastructure for calculations in the following way:
   - α has a value of 1
   - Lists (L) shall evaluate as the sum of their elements
   - Tuples (T) shall evaluate as the negated sum of their elements
   - Dictionaries (D) shall evaluate as the product of differences of key-value pairs (difference between a key and a value of a key-value-pair)
     (Examples: $\{\} = 0$, $X = \{[\alpha, \alpha, \alpha] : \alpha\} = 2$, $\{X : \{\}, \alpha : X\} = -2$

   Write semantic actions to calculate the value of such a nested datastructure. You can associate a synthesized attribute `val` to each non-terminal symbol to store their value and you can read the values of the α's from α`.val`. The final value should be returned in the top-level `A.val`.

3 LL(1)

Consider the following grammar, which describes lists of words. Terminal symbols in this grammar are $\{word, and, ,\}$.

$$\begin{array}{rcl} S & \rightarrow & word \\ & | & word\ and\ word \\ & | & M,\ word\ and\ word \\ M & \rightarrow & M, word \\ & | & word \end{array}$$

Examples:

- `word`
- `word and word`
- `word, word, word and word`

Tasks:

1. Identify and explain all the reasons why this grammar is not LL(1).
2. Rewrite the grammar so that it is LL(1).
3. Give the FIRST and FOLLOW sets for the non-terminals in the new grammar.
4. To prove that your grammar is LL(1), construct an LL(1) parsing table for it.

4 LR(1)

Again, consider the following grammar, where $\{a, b, c\}$ are terminal symbols:

$$\begin{array}{rcll} S & \rightarrow & aXab & (1) \\ & | & Y & (2) \\ X & \rightarrow & bYa & (3) \\ & | & \epsilon & (4) \\ Y & \rightarrow & Sc & (5) \end{array}$$

1. Construct the full LR(1) DFA, showing all items in each state.
2. Construct the LR(1) parsing table using the DFA. For the reduce actions, please use the provided enumeration of the productions in the grammar.
3. Show all steps required to parse the following string: `abaabccaab`

Instructions

As in Assignment 1, you are allowed to work in pairs. There are two ways to submit this assignment:

1. Submit a physical copy of your answers in my mailbox (Andreas Löscher, 99) on the 4th floor of building 1, opposite the 'fika' room.

2. Send an email with an electronic copy of your answers to `andreas.loscher@it.uu.se`.

Good luck!