Reading Assignments

- For lecture on 9/26/2013: Dragon-book 2.5 + 5.1-5.3 (28p)
- For lecture on 10/3/2013: Dragon-book 1.6 + 2.7 (15p)

Homework Assignments

5 questions, total 24 points.

1. Top-down (predictive) parsing tables (4 points).
   Consider the following grammar:
   
   \[
   expr \rightarrow expr \ast expr \mid expr \wedge expr \mid \text{number}
   \]
   
   Show the FIRST and FOLLOW sets for the \textit{expr} non-terminal.

2. Left-recursion elimination (4 points).
   Consider the following left-recursive grammar:
   
   \[
   lVal \rightarrow \text{identifier} \mid lVal \ [ lVal ] \mid lVal \ . \ \text{identifier}
   \]
   
   Rewrite the grammar so that it is not left-recursive anymore but still recognizes the same language.

3. Grammar-based associativity and precedence (5 points).
   Consider the following ambiguous grammar:
   
   \[
   expr \rightarrow expr \ast expr \mid expr \wedge expr \mid \text{number}
   \]
   
   Assume that operator operator \(*\) is left-associative, and operator \(^\) is right-associative. Further assume that operator \(*\) has lower precedence than \(^\), in other words, \(^\) binds stronger than \(*\). Rewrite the grammar to make it unambiguous, by enforcing the associativity and precedence rules. Hint: to enforce precedence, you will need to introduce a separate layer of non-terminals for each level; and to enforce associativity, your grammar rules need to recurse asymmetrically.
4. Translation scheme for type inference (5 points).
Consider the following grammar:

\[
E \rightarrow E + E \\
| F \\
| I
\]

In this language, \( F \) has type \texttt{float} and \( I \) has type \texttt{int}. If both operands of operator \( + \) have the same type, that becomes the type of the result. Otherwise, if at least one operand of \( + \) is of type \texttt{float}, then the result is also of type \texttt{float}. Write a syntax-directed translation scheme that synthesizes a \( t \) attribute with the correct type for every expression.

5. Translation scheme for code generation (6 points).
Consider the following grammar:

\[
E \rightarrow E * E \\
| E + E \\
| I
\]

Assuming that \( I \) can be any integer, an example expression in this language is \( 2 + 3 \times 4 + 2 \). Your task is to implement a translator from expressions to a stack-based intermediate representation (IR). For example, the IR for \( 2 + 3 \times 4 + 2 \) is:

\[
\begin{align*}
\text{push}(2); \\
\text{push}(3); \\
\text{push}(4); \\
\text{push}(\text{pop()} \times \text{pop()}); & \quad \text{\( (3 \times 4) \)} \\
\text{push}(\text{pop()} + \text{pop()}); & \quad \text{\( 2 + (3 \times 4) \)} \\
\text{push}(2); \\
\text{push}(\text{pop()} + \text{pop()}); & \quad \text{\( 2 + (3 \times 4) + 2 \)}
\end{align*}
\]

Your translator can assume that attribute \( I.v \) holds the value of the integer \( I \). Your translator should write the generated IR code into attribute \( i \) of each expression.