Exercise Compiler Construction (8)

Hans de Nivelle

Due: 27.11.2013

In this task, you will construct two parsers for LISP-style lists, both of which will not only parse, but also return the list. The first is a recursive descend parser, derived from your top-down parser of exercise 6 (part 3), the second is a bottom-up parser, which uses Maphoon, and which you can derive from your solution of exercise 7.

1. First download **lisp.tar.gz** from the course homepage. It contains a class **list** for Lisp-style lists. A list is either a string (atom), a big integer, or a pair of lists.

   In task 6, you have made a top-down parser based on a DFA-based grammar for lists. Translate this top-down grammar into a recursive descend parser that returns the list. You can use the tokenizer of task 6 or task 7.

   In order to get this task accepted, I want to see **(1)** a DFA-based grammar, its implementation of task 6, and the present implementation which must clearly correspond to the solution of task 6 and the DFA-based grammar.

2. Starting with the calculator of task 7, it should be easy to modify it into a program that accepts lisp-style lists, and which constructs the parsed list as attribute. You have to replace `%attribute value double` by `%attribute value list`, (or you can add another attribute), define the constraints, and modify the grammar rules.