Compiler Construction (List 3)

Hans de Nivelle

18 October 2011

1. (a) Write a regular expression of form $(\Sigma|N)^*$, where N is your first name, with non-ASCII (Polish) characters replaced by ASCII characteres. If your name is too long, you can choose another word between 4 and 8 characters.

   (b) Transform the previous regular expression into an NDFA, using the translation on the slides.

   (c) Transform the NDFA into a DFA, using the algorithm on the slides.

   (d) Simply the DFA from the previous task, using the algorithm on the slides.

   (e) At this point, everyone should have the same DFA, do you understand why?

2. Consider $\Sigma = \{'(','')'\}$, and the language $\mathcal{L}$ defined by $\sigma \in \mathcal{L}$ if σ consists of a sequence of parentheses that could occur in a legal C-program. (So "()" is OK, "(()())" also, but ")(" or "())" is not)

   Is there a regular expression that recognizes this language? Give an expression, or an explanation why such regular expression does not exist.

3. Consider the language (Σ, R, S), defined by $\Sigma = \{\ '(','')'\ \}$, $R = \{S \to SS,\ \ S \to (S),\ \ S \to \epsilon$.

   - Give derivations for

     $$'(()()())',\ '((()))',\ '(((()())))'.$$

   - Show that the language is ambiguous. (There are words that have more than one derivation.)
   - Repair the grammar, so that it is not ambiguous anymore, but still accepts the same set of words.

4. (a) In the programming language Lisp, everything is a list. The empty list has form ( ) or **nil**. Non-empty lists have form (L_1), $(L_1\ L_2), (L_1\ L_2\ L_3)$, etc. Give a complete grammar for Lisp. The elements of a list can be atoms, numbers, or by themselves lists. (You may ignore the existence of dotted pairs and arrays.)

(b) Give a grammar for the language of Prolog-style lists. Lists have form

$$[\,],\ [L],\ [L_1, L_2],\ [L_1, L_2, L_3],\ \text{etc.}$$

The elements of the lists can by themselves be lists again.

(c) Consider the language consisting of functional expressions of form c, and $f(t_1, \ldots, t_n)$, with $n > 0$, and $t_1, \ldots, t_n$ functional expressions by themselves. Give a grammar for this language.

5. (a) Give a attribute functions for the grammars of Task 4. Take into account that $(L_1 \cdots L_n)$ denotes $\mathbf{cons}(L_1, \mathbf{cons}(L_2, \ldots \mathbf{nil}))$. Give a derivation for the list

(**car** (**quote** (1 2 3))).