Introduction to CL I (684.01) OSU Linguistics
Detmar Meurers Winter 2006

Exercise sheet 4

(Submit as a plain text email message with subject "Homework 4" to dm@ling.osu.edu on Sunday, 19. Feb)

1. Compare the three encodings in `dcg/append_encoding1.pl`, `dcg/append_encoding2.pl`, and `dcg/dcg_encoding.pl`: After starting a new prolog and loading one of the files, try parsing a string that is well-formed according to this grammar, and one that is not well-formed. Specifically, try parsing `s([a,clown,loves,a,clown])` and `s([paul,laughs])` (where it is intentional that `paul` is not in the lexicon) and trace the execution.

   (a) Report how the results of the two calls differ for the three grammars, and

   (b) explain why, i.e., characterize the steps taken by the system in trying to prove these goals. (For each of the six characterizations, a couple of sentences should suffice.)

   Remember that Control-C interrupts Prolog execution.

2. Consider the following small DCG grammar (on the web as file `ex4.pl`):

```
top_s --> s([],[]).
s(G1,G3) --> np(G1,G2), vp(G2,G3).

np([gap],[]) --> [].
np(G,G) --> [the, man], postmod.

postmod --> [].
postmod --> [who], s([gap],[]).

vp(G,G) --> [slept].
vp(G1,G2) --> [saw], np(G1,G2).
```

   Explain why this DCG accepts the string "the man who the man saw slept" but does not accept "the man who saw slept" as a `top_s`. (Less than ten sentences should be sufficient to explain this.)

3. The idea behind this exercise is to think about and try out the three basic types of parsers we saw on the slides (which you can download from the course web page).

   (a) Top-down parsing can lead to termination problems with certain kinds of grammar rules. Explain this problem, using a tiny grammar to exemplify it. Then try the top-down parser we implemented in Prolog and explain whether the problem arises or not, based on an example. (In total, you should not write more than six sentences.)

   (b) Another kind of grammar rules is problematic for bottom-up parsing. As in the previous case, explain this problem in two sentences, using a tiny grammar to exemplify it. Then try the shift-reduce parser we implemented in Prolog and explain whether the problem arises or not, based on an example. (In total, you should not write more than six sentences.)

   (c) Where does a shift-reduce parser differ from a left-corner parser? Give a small example grammar suited for highlighting the difference and explain the difference by commenting on the tree traversal they perform for a small example string licensed by this grammar.

4. Read Chapter 6 “Computability and Complexity” and Chapter 7 “Introduction to Parsing” from the lecture notes.