Introduction to CL I (684.01) OSU Linguistics
Detmar Meurers Winter 2005

Exercise sheet 5
(Submit on a piece of paper at the beginning of class on Thursday, 10. Feb)

1. 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.

2. Consider the following small grammar (on the web as file `parser/cyk/cyk_grammar.pl`):

```
% lexicon:
vp  ---> [left].
vp  ---> [duck].
vs  ---> [thought].
vs  ---> [saw].
vt  ---> [saw].
vt  ---> [loved].
det ---> [the].
det ---> [a].
det ---> [her].
n   ---> [dragon].
n   ---> [telescope].
n   ---> [man].
n   ---> [cave].
n   ---> [duck].
pp  ---> [there].
p   ---> [in].
p   ---> [at].
p   ---> [with].
np  ---> [mary].
np  ---> [midnight].

% syntactic rules:
s  ---> [np, vp].
vp ---> [vp, pp].
vp ---> [vt, np].
vp ---> [vs, s].
np ---> [det, n].
n  ---> [n, pp].
pp ---> [p, np].
```

   Pick three ambiguous example sentences licensed by this grammar (or, if you prefer, a variation of this grammar, which you should provide then) and draw the CYK table for them. For each table, add a three sentence explanation pointing to the place where the ambiguities are encoded.

   In `parser/cyk/cyk.pl` (and with more basic output predicates in `parser/cyk/cyk_trace.pl`) you find a PROLOG implementation of the CYK algorithm. Try out your example sentences and see whether it fills the chart as you were expecting.

As preparation for the next exercise, try out the variants of Earley's parser we discussed in class with an example grammar and some sentences of your choice. Carefully look through some tracing output and make sure it corresponds to your understanding of the underlying algorithm.

3. Left-recursive rules are a serious problem for simple top-down parsers. The Earley algorithm has a strong top-down component. Explore with a small example grammar how the Earley algorithm (in the variant introducing preterminals as passive edges, i.e., `parser/earley/preterminals/earley.pl` and `earley_trace.pl`) deals with left-recursion. To do so, provide

   - a small grammar including direct and indirect left recursion and
   - one example run (in the form we saw in class) including use of a directly recursive rule and
   - one example run including use of an indirectly recursive set of rules, and
   - a short discussion (five sentences) of the issue of recursion in these example runs.
   - Will the Earley algorithm always terminate? Give a five sentence argumentation for or against.

4. Read chapter 6 of Gazdar and Mellish (1989)