Computational Linguistics II (Fall 2006, Exercise 3)

High-Level Goals

- Gain an understanding of how a recognizer is implemented in Lisp.
- Implement a full chart parser by modifying the recognizer.
- Employ recursion to present the output of the parser.

1 Obtaining the Starting Grammar

- Like last week, login to the Linux server '`mt.ifi.uio.no`' through X Windows.

```
cvs checkout grammar3
```

- Launch *emacs* from the shell ('`emacs &`'), and within emacs, start up the LKB ('`M-x lkb RET`'), then load the grammar using the LKB menu Load | Complete grammar.

2 Exploring the CFG Grammar

- We have converted the CFG sample grammar into LKB format. See '`lexicon.tdl`' and '`rules.tdl`'. The internal encoding of the grammar is as follows:

```
SSP(77): (pprint *grammar*)
(#S(RULE :LHS P :RHS (NEAR)) #S(RULE :LHS N :RHS (AARDVARK))
 #S(RULE :LHS DET :RHS (THAT)) #S(RULE :LHS V :RHS (GAVE))
 #S(RULE :LHS P :RHS (TO)) #S(RULE :LHS N :RHS (DOG))
 #S(RULE :LHS N :RHS (DOGS)) #S(RULE :LHS V :RHS (BARKS))
 #S(RULE :LHS N :RHS (CAT)) #S(RULE :LHS DET :RHS (THE))
 #S(RULE :LHS V :RHS (CHASED)) #S(RULE :LHS DET :RHS (THOSE))
 #S(RULE :LHS NP :RHS (KIM)) #S(RULE :LHS PP :RHS (P NP))
 #S(RULE :LHS VP :RHS (V NP)) #S(RULE :LHS VP :RHS (VP PP))
 #S(RULE :LHS VP :RHS (V)) #S(RULE :LHS NP :RHS (DET N))
 #S(RULE :LHS VP :RHS (V NP NP)) #S(RULE :LHS S :RHS (NP VP))
 #S(RULE :LHS NP :RHS (NP PP)) #S(RULE :LHS VP :RHS (V NP PP)))
```

- Review the file '`parse.lsp`' which contains the code for the parser as discussed in the lecture on Thursday. The current implementation is a recognizer only, i.e. will not try to return full parse trees or enumerate all possible analyses for each input. Try parsing something, Use the `*common-lisp*` buffer in emacs(1) to interact with the Lisp interpreter:

```
SSP(80): (parse '(the dog chased the cat))
(#S(EDGE :FROM 0 :TO 5 :CATEGORY S :DAUGHTERS (NP VP) :UNANALYZED NIL))
```

- Consider inspecting what is in the chart after parsing:

```
SSP(81): (pprint *chart*)
```

— A note on using pprint() in the above examples: just evaluating the symbols `*grammar*` or `*chart*` will return their values, i.e. it will be printed by the Lisp interpreter too. However, by default, Lisp usually truncates printing of objects, i.e. may suppress deeply embedded parts and output just '`...`' instead. The use of pprint() requests that the full object be printed.

3 Changing the Recognizer into a Parser (50 points)

- Allow multiple analyses for a (sub-)string and record the necessary information in each edge that unambiguously characterizes the full parse tree.

4 Presenting Parse Results (50 points)

- Write a recursive function edge-to-tree() that operates on one edge (typically from the return value of parse()) and returns the parse tree as a bracketed list. e.g.

```
SSP(76): (loop
             for edge in (parse '(kim chased the cat near that dog))
             do (pprint (edge-to-tree edge)))
(S (NP KIM)
 (VP (V CHASED)
  (NP (NP (DET THE) (N CAT)) (PP (P NEAR) (NP (DET THAT) (N DOG))))))
(S (NP KIM)
 (VP (VP (V CHASED) (NP (DET THE) (N CAT)))
  (PP (P NEAR) (NP (DET THAT) (N DOG)))))
(S (NP KIM)
 (VP (V CHASED)
  (NP (DET THE) (N CAT)) (PP (P NEAR) (NP (DET THAT) (N DOG)))))
```

- Use the slide copies of our Common-Lisp overview and maybe experiment a little with evaluation, list manipulation, simple recursive functions, and the almighty loop() facility. The `*common-lisp*` buffer that comes in emacs(1) as part of our LKB installation is a full copy of Allegro Common-Lisp (ACL, the most common commercial Lisp implementation).

- After making changes to our '`parse.lsp`' it may be easiest to reload the '`script`' file into the LKB, which will also re-compile your code; there are many ways of sending s-expressions directly from an emacs(1) buffer (like the one editing '`parse.lsp`') to the LKB for evaluation too, i.e. you could just redefine individual function definitions as you change things. If you feel like it, try out the ACL emacs(1) interface.

5 Background Reading and Resources

- An online copy of the Common Lisp reference manual can be found at
  `http://man.emmtee.net/man/cltl/node1.html`

- The online manual for Allegro Common Lisp is at
  `http://man.emmtee.net/man/acl/introduction.htm`

- Note that the LKB packages available for public download also include a full Lisp system, though not the compiler. All of the above will work, but your code will run interpreted, i.e. somewhat slower.

Submit your results in email to Dan and Stephan by noon on Monday, October 9.