CS 585: Natural Language Processing
Fall 2004
Programming Assignment 1: A CKY Parser

Out: Thu, September 15, 2004
Due: Tue, September 28, 2004

The aim of this project is for you to gain experience writing a simple parser from scratch. You may use any programming language for this task (we provide no skeleton for you to build on). While the focus of this assignment is not on efficiency, you program should have a $O(n^3)$ running time. You may assume grammars are always given in Chomsky Normal Form.

1 Tasks

- Your program should take three filenames as arguments: the grammar, the lexicon, and a list of sentences. The grammar and lexicon are specified in separate files, one rule per line, in the following format:

```
S -> NP VP
NP -> NP PP
VP -> VP PP

N -> Homer
P -> to
DET -> the
```

The **S** symbol is always assumed to be the start symbol of the grammar. All other symbol names may be arbitrary.

The input sentences will be provided one per line in a file:

```
The dog ate the homework
Peter relies on Mary for help with his homework
```

Download the sample grammar, lexicon, and sentences from

`http://www.cs.umass.edu/~mccallum/courses/inlp2004//pa1-sample.tgz`

Make sure your program outputs all possible parses for each sentence based on the grammar. The output format of a parse should be similar to the labeled bracketing given on page 98 of Manning & Schutze.

- Come up with a grammar/lexicon on your own to parse at least three ambiguous sentences. Make sure to convert your grammar to Chomsky Normal Form before giving it to the parser. A search for "ambiguous headlines" on Google can be a good starting point.

- **Extra credit**. Make your parser support parsing general context-free grammars (i.e., not restricted to CFGs in Chomsky Normal Form). You don't have to worry about ϵ production rules. Note that you have two choices here: (1) Write code to convert the grammar to Chomsky Normal Form, then run your CYK implementation, or (2) Implement the Early Parser.

  Extra, extra credit. Choose option (1) above, and then print the output parse *in the original grammar*, not the Chomsky Normal Form grammar.

2 What to turn in

- **Code**: Print out all source code written for the project, and your own grammars and sentences along with the parser outputs on them.

- **Program outputs**: Print out your parser's outputs for the sample sentences we provided (based on the sample grammar) and for the ambiguous sentences you used (based on your own grammar).

- **Report**: Write a report (roughly) that includes your implementation experience and details. For example: Describe some interesting issues and questions that arose during your implementation. What kinds of bugs did you encounter along the way. If there were difficulties, how did you resolve them? What data structures did you use, and how did you decide which data structures to use? What can you say about the speed of your parser? Did you make up and try a lot of different grammars? Some more ambiguous than others? How does the ambiguity and sentence length change the running time or space required? What was your favorite part of the assignment? Any suggestions for future years' versions of this assignment?