Massachusetts Institute of Technology

6.038 Artificial Intelligence in Practice

Problem Set 5B: Thursday, March 5, 1998
DUE: Tuesday, March 10, 1998

NB: For this assignment, we encourage (but do not require) you to work together in pairs and hand in a single solution per pair. Please select different partners than any from previous problem sets. You may retain the same partners for Part A and B of this assignment.

1 Background

In this assignment we try to get some experience in constructing grammars.

2 Files

The parser runs in Scheme. The files can be found in `/mit/6.038/ps5/*` (the `.scm` files are for MIT Scheme and the `.sc` files are for Scheme-to-C). To load all the files, load the file `load.scm` (or `load.sc`. If you are running Scheme-to-C, we have also provided `/mit/6.038/bin/ps5-sci` which has the key files compiled in, but you will need to load `grammar.sc`. The compiled `ps5-sci` is *much* faster than the interpreted files in MIT Scheme or Scheme-to-C.

Once all the files are loaded in (including `grammar`), then you can parse sentences by executing `(parse)`. This will go into a loop, reading a sentence — typed as a list — and printing out any assertions produced that indicate a sentence parse.

3 Assignment

Your goal is to write a grammar to handle a small domain of English sentences. The purpose of the grammar is to assign a structure (parse) to legal sentences and reject sentences that are not legal. When writing the grammar, try to avoid writing rules that accept illegal sentences. Also, make sure that you think about the potential `meaning` that will be assigned to the nodes in the parse tree.

- Pick some domain of discourse. For example, imagine that you are writing a system that will answer E-mail questions for an Internet data source, e.g. weather, stocks, buying cars, etc. Construct a list of example sentences in your domain that you want to be able to parse. Also, construct a list of (related) counter-example sentences that you do not want to accept. Try to get as varied a set of sentences as possible. Make sure you include some sentences involving `movement` as we talked about in class.
- For your domain, construct a lexicon that has enough words to cover the sentences you are interested in.
- Construct a grammar that can produce parses for the sentences in your examples list and which rejects the counter-examples.

Hand in your grammar rules and the output from a run where all the example and counterexamples are parsed (be sure that `*verbose?*` is false for this).