Grammar Engineering — ESSLLI 2016 (Exercise 3)

High-Level Goals

- Complete, debug, and refine the analysis of modification.
- Eliminate redundancy in the rules and the lexicon.

1 Grammar for Today

- Bring up our development environment by opening a terminal ('`Control-Alt-T`') and running the '`lkb` command from the shell command line.
- Continue with the grammar that you worked on yesterday, in the directory `grammar2`.

2 Complete Modification and Other Obligatory Parts of Excercise 2

- Picking up where you left off yesterday, finish making the additions to the grammar for modification with prepositional phrases, adjectives, and adverbs, as well as government of specific prepositions on complements and the head feature principle. Consider leaving the sub-excercise on phrase structure recursion for later tonight or when you return home after ESSLLI.

3 Eliminating Redundancy in the Lexicon

- Using the strategy of introducing additional types for common feature structure configurations that we have just applied to the rules, find and eliminate more redundant specifications in the grammar. Improve the organization of the type hierarchy to make it easier to add new words that are similar to words already in the lexicon. As a place to start, take a look at the lexical entries for nouns, and note that the same information is stated again and again in each entry. Recast those generalizations as constraints on a new type, *noun-word*, which every noun lexical entry inherits from. As you work, use the batch parse facility now and again to make sure none of your modifications have damaged the coverage of the grammar.
- Further eliminating redundancy from the lexicon, introduce subtypes of the type *word* for determiners, verbs, and other parts of speech, adding any constraints which are true for all instances of each word class, e.g.

  ```
  det-word := word & [ ... ].
  ```

- Introduce subtypes of the *noun-word* type for singular and plural nouns, and do the same for determiners.
- Introduce subtypes of the *verb-word* type whose instances select for third-singular or non-third-singular subjects for present-tense verbs, and an additional subtype of the verb-word type for past-tense verbs.
- Introduce subtypes of the *verb-word* type to distinguish intransitives, transitives, and the two types of ditransitive verbs.
- Take advantage of the notion of multiple inheritance to introduce lexical types in the file '`types.tdl`' for the verbs in the file '`lexicon.tdl`', making use of types from each of these two sets of subtypes of the verb-word type. Remember that the syntax for defining multiple inheritance in TDL is as follows:

  ```
  x := y & z & [A b].
  ```

  This definition says that the type x is a subtype of both y and z, and x introduces the feature `A` with value b. Note that lexical entries in the file '`lexicon.tdl`' can only inherit from a single type, so any multiple inheritance that you introduce must be defined in the types file.
- Modify your entries in the file '`lexicon.tdl`' to make use of these new types. When you are finished with this exercise, each of the definitions in your '`lexicon.tdl`' file should consist of the name of the lexical entry, the name of its lexical type, and the orthography. Everything else should be defined in the file '`types.tdl`' file. Here is a sample ideal entry:

  ```
  dog := noun-word-3sing &
  [ ORTH "dog" ].
  ```