Computational Linguistics (Spring 2010) — Exercise 3a

High-Level Goals

- Establish the *lexeme* vs. *word* type distinction.
- Add lexical rules to the grammar for inflection and derivation.
- Simplify the lexicon and extend grammar coverage using the new rules.

Background Reading

Read Sections 8.1, 8.2, and 8.4 through 8.8 from Sag, Wasow, & Bender (2003). Make sure to compare their examples and grammars to our slide copies and observe where in the lecture we have in some cases (further) simplified over the book; specifically, at this point, ignore everything they have to say about semantics, default inheritance, `CASE`, `ARG-ST`, `FORM`, and any other attributes not present in our own grammars. Also, observe that their notation of lexical and inflectional rules differs somewhat from the one used in the LKB (where lexical rules are much more similar to regular phrase structure rules, only working within the lexicon).

1 Obtaining the Starting Grammar (0 Points)

(a) Connect to the IFI Linux environment and launch the LKB. For this week, please obtain a fresh starting grammar, by executing the following command from the shell prompt.

```
exercise3a
```

For a change, we provide you with an initial grammar that produces error messages when loaded and does not parse any sentences; half-way into the exercise we will arrive at a working version of the grammar. This exercise involves a considerable amount of rearrangement of the grammar files. This is, unfortunately, something that grammar engineers end up doing quite often in real life. Take things step-by-step, and consider making backup copies of your files occasionally.

- Through this exercise, we will rework the grammar in order to make the distinction between lexemes and words and to add inflection in order to remove more redundancy in the lexicon. Following Sag, Wasow, & Bender (2003), we will make every entry in the lexicon a subtype of *lexeme*, but require that the syntactic rules continue to operate on structures which are subtypes of *word*. This means that every lexical entry will have to be converted from *lexeme* to *word* via a (lexical) rule. Some of these rules will have morphological effects, changing the 'surface' orthography using another facility of the LKB that we have not exercised so far.

2 Allow for Words to Have Internal Structure (15 Points)

(a) Move the ARGS attribute from *phrase* to a higher type which includes both words and phrases. Add the top-level lexeme vs. word distinction. We do this by distinguishing between two types *word* and *lexeme* which inherit from a new type *lex-item*. The feature ORTH is introduced on *lex-item*. The type *word* will be a supertype of all the lexeme-to-word rules (this may seem a little surprising at first, having these new unary rules be subtypes of *word*, but have patience). The general structures you need will be as follows:

```
lex-item := expression &
[ ORTH *string* ].

lexeme := lex-item.

word := lex-item &
[ HEAD #head,
  SPR #spr,
  COMPS #comps,
  ARGS < lexeme & [ HEAD #head, SPR #spr, COMPS #comps ] > ].
```

(b) Many of the existing lexical types (*verb-word* et al.) have constraints which should belong to lexemes under our new view of the world. Copying from old types where appropriate, create subtypes of *lexeme* for *verb-lxm*, *noun-lxm* et al., which use the constraints from *verb-word*, *noun-word* et al. that hold true across all inflected forms. Additionally, make *det-lxm*, *prep-lxm*, and the ones for adjectives and adverbs inherit from a new type *const-lxm*, defined as follows:

```
const-lxm := lexeme.
```

As the result of this revision, your grammar should now have a fair number of subtypes to *lexeme* and rather few subtypes to *word*—viz. only ones corresponding to inflectional properties: *3sing-word* and *non-3sing-word*. Conversely, most of the lexeme subtypes need *not* be subdivided for agreement properties, except in the case of determiners maybe (where the contrast between *that* vs. *those* can be viewed as 'lexicalized' inflection, i.e. specific AGR properties on lexemes).

3 Rework the Lexicon as a Repository of Lexemes (15 Points)

(a) Remove all lexical entries which are not morphologically equivalent to base forms (for example *dogs* and *barks*). If your '`lexicon.tdl`' file contains the form *gave* but not *give*, replace the existing entry with one whose orthography reflects the base form *give*.

(b) Change the types on the remaining entries so they are all subtypes of *lexeme* appropriate for the entry. Your file should now consist of base forms with just base orthography and with each entry instantiating a single type.

At this point, loading your grammar should no longer result in error messages printed to the Lkb Top window, and you should again be able to parse sentences. If there are still load-time errors, make sure that all required types (see above) are defined; if necessary, take a look at the (new) file '`irules.tdl`' to look up what the names for lexeme and word subtypes are expected to be. Test the functionality of the resulting grammar, using the batch parse machinery on the '`all.items`' file.

(c) You are likely to see spurious ambiguity, still, since we now have both lexemes and words in the grammar and need to make sure that only inflected forms (i.e. words) get to project into the syntax. Using knowledge about existing constraints on (the daughters in) your rules and a bit of experimentation with the LKB parser, find a sentence or two for which your grammar admits 'duplicate' analyses, where at least one tree is illegitimate because it allows an uninflected *lexeme* to directly build a *phrase*. Remark on your discoveries in a couple of sentences that you add as comments to the top of the types file. To eliminate this unwanted ambiguity, add an additional type *syn-struc* to include words and phrases, but exclude lexemes, and then use this type to ensure that phrases do not accept lexemes as their (head) daughters. Consider the use of a parameterized list type '*list of syn-struc*' (see earlier slide copies from the formalisms lecture) to encode this constraint as a condition on the ARGS value on the type *phrase*.

We have given you a new file '`irules.tdl`' which defines the actual inflectional rules. Go take a tour of these new rules, but please do not worry too much about the lines beginning with the '`%`' character; these are instructions to the LKB orthographemic component, relating the application of each rule to a specific variation in spelling. Looking at the file in emacs(1), convince yourself that all inflectional rules map arguments of type *lexeme* to signs of type *word* (which can then act as arguments to syntactic rules). Study carefully the parse tree for the sentence *the dog barks near the cat*, and observe how the different types of lexemes are mapped to words by the inflectional machinery.

4 Another Type of Lexical Rule (20 Points)

(a) Next, we will introduce another type of lexical rule, namely a lexeme-to-lexeme rule, to capture another generalization and further eliminate redundancy in our lexicon. In English, most ditransitive verbs with two NP complements (e.g. *give*, *send*, *sell*) can undergo the lexical process known as *dative shift*, resulting in a variant of the verb where the second NP argument has been promoted to the first argument position, and the original first NP argument turns into a second PP argument, headed by the preposition *to*. For example, dative shift captures the alternation in the two sentences *that cat gave the aardvark those dogs* and *that cat gave those dogs to the aardvark*. To account for this alternation in argument structure, we will add a new lexical rule, deriving one verbal lexeme from another.

Open the (new) file '`lrules.tdl`' and add a new rule which has the following rough structure:

```
dative-shift-lrule := ... &
[ ORTH #orth,
  ...
  ARGS < ... & [ ORTH #orth, ... ] > ].
```

Fill in any necessary constraints for each attribute, so that the rule takes as its single argument a ditransitive verb with two NP complements, and produces a ditransitive verb with an NP complement and a PP complement.

(b) Remove your hand-built lexical entry for the NP – PP version of *give* from the lexicon, since we now have a productive lexical rule which generates this entry for you. Your grammar should now account nicely for the dative alternation. Test appropriately and, as always, consider additions of additional test items and maybe lexical entries for testing purposes.

5 Agentive Nominalizations (30 Points)

For the remaining part of this assignment we will need to make use of the LKB orthographemic component—i.e. the facility to associate a change in orthography with a lexical rule. You should be able to just work from the examples provided in the new '`irules.tdl`' file, and refer to the lecture slides from last week.

(a) Add another derivational lexical rule to the file '`lrules.tdl`', this one deriving agentive nouns from verbs, to provide an analysis for sentences like *The barkers chased those cats* and *the chasers barked*. Adapt the orthography-changing machinery (taking inspiration from the examples of inflectional rules in '`irules.tdl`'), to add the *-er* suffix to the input orthography of this new rule. Add relevant test items (both grammatical and ungrammatical) to the file '`all.items`', and check your analysis for both overgeneration and undergeneration.

(b) Extend your analysis of agentive nouns to allow for a complement PP marked with *of*, as in *the chasers of the cats barked*. Be sure that your analysis still accepts *The chasers barked*. Assume for this exercise that we also want to accept *the givers of the cats to the dogs barked happily*. Add more test examples to your '`all.items`' to test your analysis.

6 Typed Feature Structures (5 Points)

(a) Draw the following feature structure in DAG notation, i.e. as a directed acyclic graph of labeled nodes and directed arcs:

$$\textit{phrase}\left[\begin{array}{ll} \texttt{HEAD}\ \boxed{1}\ \textit{verb} & \\ \texttt{ARGS} & \textit{*ne-list*}\left[\begin{array}{ll} \texttt{FIRST} & \textit{word}\left[\begin{array}{l}\texttt{ORTH}\ \textit{“chased”}\\ \texttt{HEAD}\ \boxed{1}\end{array}\right] \\ \texttt{REST} & \textit{*ne-list*}\left[\begin{array}{ll}\texttt{FIRST} & \textit{expression}\left[\texttt{HEAD}\ \textit{noun}\right]\\ \texttt{REST} & \textit{*null*}\end{array}\right]\end{array}\right]\end{array}\right]$$

(b) In no more than two sentences, comment on the correspondences between elements of the feature structure and elements of the DAG.

7 Linguistic Concepts (15 Points)

(a) Name at least two syntactic properties that characterize (syntactic) heads in the formation of phrases. Use one or two examples.

(b) Sketch a constituent tree for the sentence *the fierce dog chased the cat near the aardvark*. On each node, provide information about its general category (using abbreviatory notions like ‘Det’, ‘N’, ‘NP’, ‘VP’, et al.) and for each branch of the tree indicate whether the constituent dominated by it acts as a *head*, *specifier*, *complement*, or *modifier*.

(c) What is the difference between the function of the prepositional phrases (marked by square brackets) in the following two sentences:

(i) *that cat gave the aardvark [to the dogs]*

(ii) *the cat chased the aardvark [near the dogs]*

8 Submitting Your Results

To provide your results to us, please pack up the entire contents of your ‘`exercise3a/`’ directory when you are done—for example as one ‘`.tgz`’ or ‘`.zip` file. Email the archive file to both Arne and Stephan before the final deadline, midnight on Monday, May 3. In the IFI Linux environment, we provide the *submit* command-line tool for you to automate the process of packaging up and sending your results to us. For the theoretical questions, make sure to include comments in your grammar files and consider adding a file ‘`README`’, or the like. If you prefer submitting your results to questions (7) and (8) in more traditional form, say drawing on a sheet of paper, please make sure to get your answer to Arne or Stephan in time. For these parts, we will accept submissions until the start of the lecture on Thursday, May 6 (at 12:15).