

# Grammar Engineering — Winter 2005 (Exercise 7)

## High-Level Goals

- Extend the grammar to include long-distance dependencies.
- Explore an analysis of relative clauses.

## 1 Background

Like many languages, English has constructions where at least one of the syntactic arguments of a predicate does not show up right beside its predicate, but somewhere farther away. One class of these constructions is called *long-distance dependencies*, illustrated by the following examples:

- (1) *That cat, the dog wanted me to give to the aardvark.*
- (2) *Examples like this, I'm sure nobody ever really says.*

Long-distance dependencies are discussed in more detail in Chapter 14 of (Sag, Wasow, & Bender, 2003); for the current exercise, it will be enough to remember that a sentence-initial phrase like *that cat* can be analyzed as providing the *filler* for a *gap* (a missing argument) somewhere later in the sentence.

## 2 Obtaining the Starting Grammar

For this exercise, please get a fresh start and use our initial grammar, essentially the model solution for the last non-Esperanto exercise:

```
cvs checkout grammar7
```

## 3 Topicalization (40 Points)

- Prepare the basic machinery for handling long-distance dependencies. Our approach will be to remove a complement from the **COMPS** list of a predicate using a new syntactic rule, store it for awhile in a **GAP** attribute, and then retrieve it at the top of the phrase structure tree using a new syntactic rule.
- First, in the file ‘**types.tdl**’ add the feature **GAP** to *expression*, and make its value be of type *\*dlist\**, for reasons that will become clear in a moment. Then consider the various sub-types of *expressions*; we will assume that all lexical items have an empty **GAP** list. Make this so.
- Phrases may contain a gap, and that gap might come from any of the daughters of a phrase, so we use the same difference-list mechanism to collect up these values that we used for **ORTH** and **RELS** earlier already. Add a subtype of the type *unary-rule* called *unary-rule-pg* (short for ‘pass gap’) which makes the phrase’s **GAP** be unified with the **GAP** of its only daughter, and add a subtype of the type *binary-rule* called *binary-rule-pg* which appends the **GAP** values of its two daughters. Modify the sub-types of *unary-rule* and *binary-rule* to inherit from these new types.
- Still in ‘**types.tdl**’, add a new rule type *unary-head-initial-sg* (for ‘start gap’) which has a non-empty **GAP** whose single element is a *syn-struct*. Since this is where we introduce the missing argument, and we assume that only one gap per sentence is needed, the daughter in this construction should have an empty **GAP** value.
- Now move to the file ‘**rules.tdl**’ and modify the existing rules to inherit from the new ‘gap passing’ types.
- Still in ‘**rules.tdl**’, add a new unary rule which inherits from *unary-head-initial-sg*, and which extracts one complement of its daughter, moving it into the **GAP** list of the mother.

- Finally, add a new rule called *filler-head-rule* of type *binary-head-final* whose first daughter is identified with the single value of the **GAP** attribute on the second daughter. This is the rule that will combine, for example, *that cat* with *the dog chased* to build *that cat the dog chased*. Save your files, then reload the grammar, and check that you can parse the example *that cat the dog chased*. If not, make the required repairs, until you can parse this example.
- Now consider the examples *that dog the cat gave to the aardvark* and *to that cat the dog gave the aardvark*. Once these are parsing, try testing your analysis on the batch file ‘`gap.items`’. Use the earlier test items file to make sure a that there is no ‘leakage’ of gaps.

## 4 Relative Clauses (60 Points)

- Now that we have an analysis of gaps, we can introduce significant additional coverage to the grammar by adding an analysis of relative clauses, as in *the dog that the cat chased barks*. Some of the modifications to the grammar will look similar to what you did for gaps, and some will be new. Our analysis will assume that relative pronouns like *that* introduce a feature which propagates up to the clause containing the relative pronoun, and that there is a separate unary rule which discharges this feature and constructs a phrase that is a postmodifier of nouns, with the semantic index of the relative pronoun unified with that of the modified noun.

You will have considerable freedom in implementing this analysis, but here are some initial steps and a few guidelines:

- Add the feature **REL** to *expression*, and make its value of type *\*dlist\** for reasons similar to those for the feature **GAP** (we use the name **REL** for consistency with standard conventions in HPSG, but don’t confuse it with the unrelated semantics feature **RELS**).
- Make sure that all existing syntactic rules (both *unary-rule* and *binary-rule* subtypes) preserve the values of **REL** from their daughters.
- Add a new lexical entry for the relative pronoun *that* to the lexicon, with most of the properties of an ordinary pronoun, but with its **REL** value a one-element difference list where that one element is of type *index*, and that element is unified with the entry’s **SEM.INDEX** value. You may want to consider making this entry an instance of *word* rather than *lexeme*. However you decide, be sure that this entry has a value specified for the semantics feature **SEM.RELS**.
- Make sure that all other lexical entries have an empty value for **REL**, modifying the lexical type hierarchy appropriately.
- Add the unusual non-branching rule that converts a sentence containing a relative pronoun into a noun-modifying phrase. Notice that this rule will have a **HEAD** value on the mother that is not identified with the **HEAD** value of its (only) daughter, since verbs have an empty **MOD** value, but we want to build a phrase whose **MOD** value is non-empty. Unify the **SEM.INDEX** of the phrase with the index that is the single element of the **REL** value of the daughter, and also unify it with the index in the **MOD** value of the mother. This little bit of stitching will ensure that your grammar produces an elegant semantics for sentences with relative clauses.
- Not all relative clauses in English contain the relative pronoun *that*, of course, and in fact, some relative clauses contain no pronoun at all, as in *the dog the cat chased barks*. Extend your grammar to provide an analysis (including the appropriate semantics) for sentences containing these so-called ‘that-less’ relative clauses. As always, include a file of example items to showcase the coverage of your analysis.

**Submit your results in email to Dan and Stephan by noon on Tuesday, March 1.**