

Grammar Engineering — Winter 2005 (Week 6)

Goals

1. Complete our implementation of Esperanto morphology.
2. Enrich the types for phrase structure rules in Esperanto.
3. Add semantics to the lexical entries and rules.
4. Use the LKB generator to identify overgeneration in the grammar.

Exercises

1. If you have not already done so, finish up the exercises from last week, so your grammar can provide at least one analysis for each of the examples in the file 'hard.items'.
2. You will notice that a familiar kind of spurious ambiguity arises in the Esperanto grammar, which allows prenominal adjectives and postnominal prepositional phrases. Modify your grammar to exclude one of the three analyses for *Knaboj lernis malfacilajn lingvojn en la lernejo*. Try developing your solution using the boolean feature --PM (for 'post-modified') on phrases. **(10 points)**
3. Add the types we'll need for semantics, modeled on the improvements we made for **grammar4**. First, add the feature SEM to *expression*, with the value constrained to be of type *semantics*. Then add the relevant semantic types to the file 'types.tdl', copying them from what you did for **grammar4**, including the types and subtypes for *index*, *relation*, and *semantics*. **(10 points)**
4. The syntactic rules in your Esperanto grammar look much like the very first version of the rules that you saw in the English grammars. In order to add semantics, we will need to upgrade the rule types for Esperanto. Add the types *unary-rule* and *binary-rule* in the 'types.tdl' file, modeling the definitions on the ones we used for **grammar4**, but ignoring the ORTH feature for this exercise. Then add the three types crossing the *head-initial/head-final* dimension with the *unary-rule/binary-rule* dimension, where we'll continue to treat unary phrases as always head-initial. Now using these additional types, revise the rule definitions in 'rules.tdl', and while you're at it, eliminate *head-complement-rule-2* by generalizing *head-complement-rule-1* as we did earlier for English. (You will notice that we don't yet have any lexical entries with two or more complements, but there may well be some.) **(10 points)**
5. Now enrich the lexical types and lexical entries with semantics, as you did for English for **grammar4**. Begin with the basic lexeme types for nouns, verbs, determiners, adjectives, and prepositions, and work to provide the desired semantics for sentences that do not require use of the lexeme-to-lexeme rules. To simplify debugging for this exercise, use English names for the semantic predicates for your Esperanto lexemes, even though this would not be appropriate for a real lexicon of another language. **(20 points)**
6. Remember to carry up the semantics from daughter to mother in each of the types for stems, themes, and words. Build up a file of these simple examples to test your initial semantics. **(10 points)**

7. Expand the LKB menu so you can use the generator. Index your lexicon for generation (using the “Generate” menu), and then try to generate from your simple test items. (Recall that you can invoke the generator by first parsing a sentence, and then clicking on one of the resulting little parse trees in the tree summary window, and choosing the “Generate” command.) If you encounter a warning about lexical rules, check to see that each of the lexeme-to-lexeme rules is constrained so that the output cannot serve as the input to the same rule. (If a rule can feed itself, then the generator will never know when to stop applying the rule over and over again, and will run out of edges.) If you see other opportunities to improve your grammar to reduce overgeneration, make the relevant changes. Write a brief discussion of the behavior of the generator for your grammar of Esperanto. **(10 points)**

8. The lexeme-to-lexeme rules not only change the syntactic properties of their inputs, but can also add semantic information. For this purpose, we will need an additional attribute on *expression* called **C-SEM** whose value is of type *semantics*. To use this feature, modify the definitions of the lexeme-to-lexeme rules so that the value of **RELS** on the (output) type is the result of appending the **RELS** value of the daughter and the **RELS** value of **C-SEM** of the mother. You will need to exercise once more your mastery of difference lists, and you might want to look at the definition of the type *binary-rule* to remind yourself. Now add an appropriate relation to the **C-SEM.RELS** value of each of the lexeme-to-lexeme rules, and link up the **ARGO** of that relation to the appropriate argument value in the daughter’s relation. Construct a file of test items to show off the splendid behavior of your grammar. **(30 points)**

Submit your results in email to Dan and Stephan by noon on Tuesday, February 22.