

050.334/634 — Computational Models of Cognition

Lab 3: Due 10/28/05

To do this assignment, you will need to retrieve the following files from the course website: lab3-bottom-up.pl and grammar-bottom-up.pl.

Problem 1:

Shieber (1983) proposes to model human parsing preferences using a bottom-up parser via the following preferences among parsing operations (when both yield a possible parse):

- (1) Prefer Shift over Reduce.
- (2) Prefer Longer Reductions over Shorter Reductions.

The bottom-up parser I demonstrated in class does not exhibit either of these preferences, and as a result produces the parses for a sentence in the opposite order from the one Shieber suggests it should:

```
?- parse(X, [], [john, sees, the, man, with, the, telescope]).
```

```
X = s(np(pn(john)), vp(v(sees), np(np(det(the), n(man)), pp(p(with), np(det(the), n(telescope)))))) ;
```

```
X = s(np(pn(john)), vp(v(sees), np(det(the), n(man)), pp(p(with), np(det(the), n(telescope)))) ;
```

Modify the prolog code for the bottom-up parser so that it abides by both of these preferences. Indicate clearly the changes you need to make in order to represent each one. (Hint: you may assume that the longest grammar rule has at most three elements on its right hand side.)

Problem 2:

In class, we discussed how to enforce agreement between determiners and nouns, by utilizing an additional argument of the relevant predicates. This can be done in grammars of the sort that are used by the bottom-up and top-down parsers in the same way:

```
np(np(DET,N)) ---> [det(DET,Agr), n(N,Agr)].
det(det(a),sing) ---> [a].
det(det(most),plur) ---> [most].
det(det(the),_) ---> [the].
n(n(book),sing) ---> [book].
n(n(books),plur) ---> [books].
```

In this problem, you will use this idea to capture a number of other regularities of natural language.

Part A: Modify the necessary grammar rules to ensure that subjects and verbs agree in features. That is, make sure that your grammar generates only grammatical examples like those in (3), and not examples like those in (4).

- (3) a. The man opens the books.
- b. Most men open the books.
- (4) a. *A man open the books.
- b. *The men opens the books.

(Hint: think about which categories will need to encode agreement features, and which rule you will use to enforce agreement between the subject NP and the verb.) Note that in this part and in the rest of the assignment what your grammar does *not* generate is just as important (if not more so) as what it does.

Part B: Verb phrases may take a variety of forms in English: they may include just a verb (called an intransitive VP), or a verb with a noun phrase object (a transitive VP), or a verb with a noun phrase and a prepositional phrase (a ditransitive VP), or any one of a number of other possibilities.

- (5) a. Karl laughs.
- b. Valerie discovers the telescope.
- c. Libby reveals the book to the woman.

A given verb need not be able to appear in each of these types of verb phrases, as the following examples show (* indicates an impossible sentence):

- (6) a. *Karl reveals.
- b. *Valerie laughs the book.
- c. *Libby discovers the book to the man

This restriction of a verb to a certain type (or set of types) of verb phrase is called the verb's *subcategorization*: some verbs are marked as intransitive, others as transitive, and still others as ditransitive (and some may be marked as more than one of these). Your task in this part is to modify the grammar in such a way that it only generates sentences that respect subcategorization. Do this by first modifying the rules which introduce the verbs, adding a property reflecting into the predicate associated with each verb (intransitive, transitive or ditransitive), just as we did for agreement. Then, modify the VP rules so that they make use of this information and generate only sentences that respect subcategorization requirements. Make sure to test your grammar on both grammatical and ungrammatical sentences! Your grammar should include subcategorization information for all of the verbs in the original grammar file, but you may also add other verbs as well.

Part C: In the previous part of this question, subcategorization was conceived of as an atomic property associated with a verb, either intransitive, transitive or ditransitive. An alternative way

of representing subcategorization is in terms of a list of arguments. Thus, the verb *opened* would be associated with the list [np], while the verb *put* would be associated with the list [np, pp]. Modify your grammar so that subcategorization information in lexical entries is represented in this list form, and adjust the grammar rules to make use of this new representation. (Optional: try to simplify your grammar so that it makes use of a single VP rule by using the subcategorization list to fill in part of the right hand side of the rule.)

Part D: Add additional rules and/or subcategorizations to generate sentences involving clausal complements, such as the following:

- (7) a. Mary says that the books exist.
- b. The men wonder whether John laughs.
- c. John knows that the women see the hills.

Assume that clausal complements are labelled CP (or complementizer phrase), and consist of a complementizer (which may be either the words *that* or *whether*) followed by a sentence. Make sure to represent subcategorization properties appropriately on the verbs you introduce, following what you've done in part D.

Part E: Representing subcategorization in terms of a category or list of categories is too coarse-grained to encode subtler distinctions that exist among verbs which select for phrases of the same category. For instance, even though *give* and *remove* are both ditransitive in the sense that each may form a VP with NP and PP arguments, the PP argument of *give* must be headed by the preposition *to*, while the PP argument of *remove* must be headed by *from*.

- (8) a. Karl gives a book to/*from Valerie.
- b. Libby removes the telescope *to/from the hill.

Similarly, the CP arguments of *say* must always begin with the complementizer *that*, while those of *wonder* must begin with *whether* (*know* may take either).

- (9) a. Karl says that/*whether Valerie laughs.
- b. Valerie wonders *that/whether Karl exists.
- c. Valerie knows that/whether Karl exists.

Modify your grammar so that subcategorization features may refer to the identity of the lexical head of a phrasal complement, and use this new encoding to enforce the requirements in (8) and (9).

Problem 3:

Another kind of CP complement involves sentences of the following sort:

- (10) John knows who Mary sees.

In this case, the complementizer is filled by the wh-pronoun *who*. To generate such a sentence, it is not sufficient to add *who* to the set of possible complementizers for *know*: the presence of *who* prevents the appearance of the np object, as the complementizer *who* plays the role of the object.

(11) *John knows who Mary sees the telescope.

This dependency can be represented by adding to each grammar rule an additional feature that encodes information about the “fronted” wh-object. The CP rule will add the fronted whnp as the value of the gap feature for the S, and the whnp rules will introduce the wh-phrases *who* and *what* (without introducing a gap feature, like other rules that introduce lexical items), as follows:

```
cp (cp (NP, S), nogap) ---> [whnp (NP, nogap), s (S, gap (NP))] .  
whnp (np (who), nogap) ---> [who] .  
whnp (np (what), nogap) ---> [what] .
```

Other rules of the grammar should pass this gap feature downward without change. Finally, a noun phrase with a gap feature may be realized as the empty list, as follows:

```
np (gap (NP), gap (NP)) --> [] .
```

Following this strategy, make the necessary changes to your grammar to generate examples like the following:

- (12) a. John knows who Mary sees.
b. John knows who sees the the hill.
c. John knows what Mary gives to the men.
d. John knows who the men give the books to.
e. John knows what the woman says that Mary sees.

Make sure that your grammar continues to enforce subcategorization requirements and does not accept examples like (11).

Note that in doing this problem, you may find it easier not to use the simplest grammar hinted at in part D of question 2 (with a single VP rule), but rather one with separate VP rules for each subcategorization possibility.