

GDE – Handout 4

This handout continues with the HPSG-like implementation within ProFIT. It deals with a broader-coverage grammar for English, modelled on Sag & Wasow (1999), originally written by Graham Wilcock and extended by Paul Bennett. One of the extensions incorporates a treatment of verb classes along the lines of Dixon (1991). The intention is that you should use this grammar to explore various aspects of (an HPSG description of) English. In particular, you will be able to see how a small set of schemata are responsible for licensing a wide range of structures, how various kinds of generalization are captured, and how the lexicon can be structured so as to make maximum use of inheritance and to ease the task of adding new lexical entries. To aid in all this, there are frequent references to relevant discussion in the Sag–Wasow text.

1

Start by copying the grammar file into the directory containing your earlier ProFIT files. Assuming this is a sub-directory called `GramKit2` inside the `GDE` directory, this involves:

```
cd GDE/GramKit2
cp /home/rhum/paul/GFL/GramKit2/english.fit .
```

Include the final `.` ('dot'), which indicates the current directory. Then edit the `load.pl` file already in this directory so that it contains the line

```
:- fit('english').
```

and ensure that all other lines beginning with `fit` are commented out (by putting `%` at the start of them).

I have suggested throughout some words to add to the lexicon, largely so that you can test your understanding of the grammar. I hope I no longer need to tell you to edit a copy of the grammar and to alter which grammar is to be loaded by the `load.pl` file.

Don't forget the `tidy (X)` command to delete unwanted files once you've finished a session. If you want to delete all the automatically-generated files in `GramKit2`, just type `tidy` with no argument.

2

Now load the grammar in the standard way:

```
sicstus -l load
```

Get into parse mode by typing `parse.`, and then you could start by just parsing a small number of sentences to get an idea of the grammar's coverage and the structures it outputs. E.g., try parsing *the woman follows me* and *the nice man keeps laughing* and *it was eaten by the boy*. As before, both a syntactic tree and a semantic representation are shown. Have a look at these, and in particular get used to reading the semantic representations. Parse *Kim walked*: there are two analyses, one being the wanted one with a finite verb, the other with *walked* interpreted as a past participle, which is not wanted but cannot be excluded.

My suggestion as to how to proceed once you've examined a few sentences is that you look at a part of the grammar and lexicon and at the same time at how expressions containing words you've been examining are handled. It will sometimes help to switch on `show(signs)`. and then to parse individual words in order to look at the full specification for these.

Maybe you could start with pronouns (search through the grammar file `english.fit` for the word 'pronoun'). The properties of each pronoun are described in a very simple way, e.g.:

```
pron(he, nom, 3&sg&m, male, -).
```

The case of *he* is `nom(inative)`, its agreement properties are 3rd-singular masculine, it restricts its referent to being male, and it is non-anaphoric. This is converted into a fuller lexical entry by the following Prolog rule:

```
lex(Pron, @pron_lxm &
    syn!head!(case!Case & agr!Agr & ana!Ana) &
    sem!(index!I & restr![reln!Reln & inst!I]))
:-
    pron(Pron, Case, Agr, Reln, Ana).
```

The features of the `pron_lxm` template are inherited, and the information specified in the abbreviated entry is matched with the body of this rule and copied over as values of appropriate features (e.g. for *he, nom* becomes the value of `syn!head!case`). This copying is achieved by the use of Prolog variables (words with initial capitals); the same variable-name, such as `Case`, is given an identical interpretation at each occurrence. Then the following applies:

```
lex(Pron, <word & syn!Syn & sem!Sem)
:-
    lex(Pron, @pron_lxm & syn!Syn & sem!Sem).
```

The lexical entry for the pronoun is mapped into an expression of sort `word`, and its syntactic and semantic properties are copied over. A comparable approach is taken to determiners and common nouns: their properties are specified in a simple format and this is automatically converted to a form used by the grammar. This makes it much easier to add new lexical items in these categories. Now you can parse words like *me, he* and *nobody*. Add your own entry for *us* or *herself*.

Next look at the description of determiners and nouns. The indefinite article *a* has the value `3&sg` for its `syn!head!agr` feature, while plural nouns are subcategorized for a specifier with `3&pl`. The `agr` value of *a* will not unify with the subcategorization requirement of a plural noun, hence the ill-formedness of **a books*. Equally, *a* is `syn!head!count!+`, and a mass noun requires its specifier to be `count!-`, which is why **a furniture* is excluded. Try parsing a determiner and a noun separately, then put them together (e.g. *a brick* and *the food*). Add entries to enable the grammar to handle *that student* and *those pigs*.

3

Now is a good time to look at verbs. Dixon (1991) distinguishes a number of semantically-defined classes of verbs, which assign their own semantic roles and appear in specific alternations. For instance, *present* is a verb of giving, and so occurs with the roles of donor, gift and recipient; it appears in the *to-* and *with-*variants (*present X to Y, present Y with X*). The grammar's treatment of verb classes, based on an implementation of Dixon's work, differs from that found in orthodox HPSG, but is perfectly compatible with the general theory.

The lexical entry for *present* simply says:

```
give_verb(present, (to or with)).
```

For each semantic class of verbs, there are one or more lexical rules which map it into a more fully-specified lexeme, e.g.:

```

lex(Verb, @give_lxm &
    arg_st![_ , @np(J), @pp(to,K)] &
    sem!restr![reln!Verb & gift!J & recipient!K])
:-
    give_verb(Verb,to).

```

This is responsible for the *to*-alternant of a verb of giving. Assigning a semantic role to the subject is the same for all alternants, and so is done by a separate template (called by the above rule):

```

give_lxm := <give_verb & @tv_lxm &
    arg_st![@np(I)|_] &
    sem!restr![donor!I].

```

This links the subject to the donor role, and itself invokes the template for transitive verbs.

Other lexical rules map a lexeme to its various inflectional forms. Get the system to parse the word *presenting*, and then try the sentence *Kim presents the boy with some food*. Check how the output relates to the semantics specified in the above lexical rule. Add an entry for *donate* – note that this only occurs in the *to*-alternant.

An implementation of the Argument Realization Principle (Sag & Wasow 1999, p. 151) links the `spr` and `comps` values to the `arg-st` (argument-structure) list, the latter being a concatenation of the first two lists. The ARP template is called by a large number of lexical rules. An implementation of the Case Constraint (Sag & Wasow 1999, p. 184) assigns accusative case to all non-subject NPs in a verb's argument structure, thereby avoiding the need to state this again and again for different verb classes or even individual verbs.

You can now explore the various verb classes employed by the grammar – try parsing examples such as *Pat hits me with a stick* or *Dana wraps the cake in some paper*. If you like, add some new lexical entries for members of existing verb classes.

Verbs that take a sentential complement include the verbs of thinking (such as *think* and *believe*). These subcategorize for a finite clause, via the `@finclause(_)` template, which covers both presence and absence of a complementizer (Sag & Wasow 1999, p. 259). Try parsing both *Kim thinks John walked* and *Kim thinks that John walked*.

The semantic representations contain information about tense, as in the following for *Kim walks*:

```

[name!Kim&
    named!B&
    reln!name
    ,
    moving!B&
    reln!walk&
    sit!K
    ,
    reln!t_overlap&
    ref_time!K&
    speech_time!now
]

```

The second `reln` in this representation involves two arguments, since tense relates a reference time to the time of speech. This is a present-tense sentence, so the temporal specification

involves overlap between the reference time and the time of speech (**now**) – the time of walking includes the speech time. The argument is co-indexed with the situation described (the **E** variable). So we might paraphrase this as: there is a situation of walking, and the walker is named Kim, and this situation overlaps the time of speech. (This analysis of tense is based on fairly standard linguistic approaches – see Comrie (1985).)

The next section carries on looking at the treatment of verbs.

4

Auxiliary verbs invoke the `srv_lxm` template, which is for subject-raising verbs, and which in turn invokes a more general raising template, which applies also to adjectives such as *likely* and is here given in a simplified form:

```
srv_lxm := <srv_lxm & @verb_lxm & @ssr.
```

```
ssr := arg_st!([Subj, syn!val!spr![Subj]]).
```

The argument-structure list contains two items, a subject and a complement; the specifier of this complement is the same as the subject of the raising verb (shown by the shared `Subj` variable). So it is really very easy to capture this kind of structure-sharing. For *Kim may walk*, the semantics is:

```
[name!Kim&
  named!B&
  reln!name
  ,
  arg!J&
  reln!may&
  sit!L
  ,
  reln!t_overlap&
  ref_time!L&
  speech_time!now
  ,
  moving!B&
  reln!walk&
  sit!J
]
```

Here, *may* takes a single argument, that of the situation of walking, which itself has an entity named Kim as its argument. This kind of representation is appropriate for the epistemic modality reading of *may*, as in ‘it is possible that Kim walks’. Add an entry for the modal *might*.

For *Kim has walked*:

```

[name!Kim&
  named!B&
  reln!name
  ,
  reln!t_precede&
  event_time!M&
  ref_time!N
  ,
  reln!t_overlap&
  ref_time!N&
  speech_time!now
  ,
  moving!B&
  reln!walk&
  sit!M
]

```

Look at the tense information here: the event time precedes the reference time, which in turn overlaps the speech time. The English present perfect is often described as having a meaning of ‘current relevance’, and this is shown by having the reference time at the time of speech. You can now try parsing examples like *Kim had walked*, *Kim may have walked* or *John has been following him*.

Other raising verbs are handled in a similar way, as verbs that take sentential arguments, e.g.:

```
sent_verb(seem, to, extrap).
```

The lexical rule for subject-to-subject raising verbs applies:

```

lex(Verb, @srv_lxm & @main_verb_lxm &
    arg_st![@'NP', syn!head!form!Form & sem!index!I] &
    sem!(index!Sit & restr![reln!Verb & sit!Sit & arg!I]))
:-
    sent_verb(Verb, Form, _).

```

The template for raising verbs was mentioned above. *To* is treated as a complementizer (do a search on ‘Complementizer’ to locate its entry). Parse *Kim seems to walk* to look at the analysis that is made. This analysis of *to* is set out in Sag & Wasow (1999, pp. 271–3).

Control is dealt with along the lines of Pollard & Sag (1994, ch. 7). E.g. *persuade* is described as an **influence_verb**, which means that the influenced argument is the controller. Parse *Kim will persuade Pat to walk*, and then try examples like *Kim wishes to walk* and *Pat has promised Kim to walk*, to see the control relations. With control verbs, only the **index** value is shared between controller and controllee, whereas with raising verbs, the entire structure is shared/unified (Sag & Wasow 1999, p. 281). You could add entries for other raising and control verbs, such as *appear* or *try*.

Passives are handled by means of two lexical rules, one for short passives (without a *by*-phrase) and one for long passives (with a *by*-phrase), each of which maps a transitive verb lexeme into a passive verb. The **arg_st** list of the passive verb differs from that of an active in that the second NP on the **arg_st** list of the active (i.e. the object) is made into the first NP on the **arg_st** list of the passive (i.e. the subject). In a long passive, the active subject

is appended to the `arg_st` list as a *by*-phrase. Maybe the best way to appreciate this is to parse a participle such as *seen* (ignoring the parse of it as a perfect participle), and then look at *Kim was seen by John*, and compare it to *John saw Kim*. It is worth looking at a selection of examples with passives, so you can see how a single rule copes with a variety of structures: e.g. *Pat was hit with a stick*, *the man was given a book* and *the food seems to have been eaten*. The interaction of passive and control works fine: try *Kim was persuaded to walk by Bob*; nothing specific needs to be said about the passive of control examples. However, the system does not parse *Kim was persuaded by Bob to walk* (with the *by*-phrase before the *to*-phrase), and it does parse the ungrammatical **Kim was promised to walk*.

The verb *rumour*, which only occurs in the passive (*Kim is rumoured to be walking*), simply has a specification for its passive participle, and there is no entry for the lexeme *rumour*, thereby excluding examples like **they rumour Kim to be walking*.

5

Now we've finished what we have to say about verbs, and you can start looking at the treatment of adjectives. Simple adjectives such as *happy* are described very straightforwardly, and lexical rules create two separate entries for adjective words, one for adnominal (modifying, or attributive) use and one for predicative use. The analysis of premodifying adjectives is based on that in Sadler & Arnold (1994): words modify words, and an adjective modifies a noun. This is to avoid examples such as **a proud of his son man*, with a premodifying AP. The Head-Modifier Rule states that an expression with a feature `mod!X` modifies `X` (see the slightly different rule in Sag & Wasow (1999, pp. 114-5)). Try parsing *the tall man*. Modifying adjectives are not properly integrated into the semantics of their NP, however.

The lexical rule for predicative adjectives states that they do not modify anything and have the feature `syn!head!pred!+`. *Be* is required to occur with a complement bearing this feature, which is also found on present and passive participles. All other verb forms, and empty prepositions, are `pred!-`, which rules out **Kim is walks* and **Kim is to Pat*. So a single lexical entry for *be* is possible, in which a variety of different subcategorization possibilities are captured very simply. (For a brief discussion of this way of handling *be*, see Sag & Wasow (1999, pp. 251-3).) Now you can parse *Kim is happy*:

```
[name!Kim&
  named!B&
  reln!name
  ,
  reln!t_overlap&
  ref_time!N&
  speech_time!now
  ,
  theme!B&
  reln!happy&
  sit!N
]
```

There is no reflex of *be* here: *happy* is simply predicated directly of *Kim*.

Adjectives which occur only predicatively or only attributively are covered, though rather less elegantly, as is a single example of an adjective that takes two arguments – parse *Kim is aware that Bob followed him*. Try adding entries for an intransitive adjective such as *pretty*, and then for *familiar* (as in *familiar with X*).

This section looks at some miscellaneous aspects of the grammar.

Not is treated as an adverb with the feature **reln!not**. The negation lexical rule applies to auxiliary verbs, and creates a word which is subcategorized not just for its ‘ordinary’ complements but also for a negative word (Sag & Wasow 1999, pp. 305–7). For *Kim did not walk*, the syntax and semantics are:

```

      S(fin)
      |
NP-----VP(fin)
 |         |
  N V(fin)-ADVP-VP(inf)
 |   |   |   |
 |   |   |   |
 |   |   |   |
 |   |   |   |
Kim did not walk

[name!Kim&
  named!B&
  reln!name
  ,
  reln!t_precede&
  ref_time!N&
  speech_time!now
  ,
  reln!not&
  sit!N
  ,
  moving!B&
  reln!walk&
  sit!N
]
```

The ‘supportive’ *do* makes no contribution to the meaning, and so has no reflex in the above semantic representation. Try parsing *Kim has not walked* and *Bob is not happy*.

The grammar also handles other adverbs, such as *quickly*: try parsing *Kim walks quickly*. Add an entry for *carefully*.

A partial treatment of coordination is included (Sag & Wasow 1999, pp. 120–2). Look at the analyses of examples such as *Kim is tall and happy* or *I saw Bob and Kim*. Unfortunately, the semantics is not always quite what is wanted; and in other cases the coordination rules lead to massive overgeneration (try parsing *Kim and Bob saw me*).

Yes–no questions are handled by means of the Head-Specifier-Complement Rule, which sanctions a word (an auxiliary verb) followed by a subject and a complement. The only difference between the analysis of a question like *has John eaten the food* and the corresponding declarative is the value of the **sem!mode** feature, which is **ques** for the question but **prop** for the statement (Sag & Wasow 1999, pp. 106–7). This can be seen informally in the tree printed for the question, and more clearly if you turn on **show(signs)**. and examine the full representation.

The grammar contains a partial implementation of the binding of pronouns (Sag & Wasow 1999, p. 152). This is commented out in the grammar as supplied to you, as it slows down its operation. If you want to explore the rules for binding, you will need to remove the comments before the two 'BIND' templates, and also before the call to them in the `tv_lxm` template and in the lexical rules for 'smallclause' and 'sor' thinking verbs. The templates make use of the `ana` feature on pronouns, which states whether they are anaphoric (reflexive) or not. Then try parsing an example like *John saw himself*, and look at the indexing in the semantics. It's best to comment these parts out again once you've finished looking at the treatment of binding.

References

- Comrie, B. (1985), *Tense*, Cambridge University Press.
- Dixon, R. (1991), *A New Approach to English Grammar, on Semantic Principles*, Clarendon Press.
- Pollard, C. & Sag, I. A. (1994), *Head-Driven Phrase Structure Grammar*, University of Chicago Press.
- Sadler, L. & Arnold, D. (1994), Prenominal adjectives and the phrasal/lexical distinction, *Journal of Linguistics* **30**, 187–226.
- Sag, I. & Wasow, T. (1999), *Syntactic Theory: a Formal Introduction*, CSLI.