

Chapter 8

The Structure of the Lexicon

8.1 Lecture notes

Chapter 8, Lecture One

I. Capturing generalizations

- So far, the grammar has developed from having simple lexical entries and a complex set of rules, to have a simple set of rules and complex lexical entries.
- In Chapter 8, we discuss how we organize the lexicon to pull out as many generalization as possible.
- There are two techniques for doing this – the type hierarchy and lexical rules.

II. The type hierarchy

- ? Why do we classify feature structures into types?
 - To avoid pointless feature specifications, e.g. ANA on verbs
 - To let us impose constraints on whole types of elements, reducing stipulation
- ? Why do we arrange types hierarchically? (To allow generalizations over classes of elements of various sizes.)
 - ‘Maximal’ types:

- ? What is a ‘maximal type’? (One that has no subtypes.)
- ? Why do we call it ‘maximal’ (rather than, say, ‘minimal’)? (Its intension (its information content) is maximally spelled out; consequently, its extension is minimal. The ‘sub-’ and ‘super-’ designations are based on extension, whereas the ‘maximal’ designation is based on intension.)
- ? Why do we have default inheritance allowed? (Idiosyncratic exceptions and subregularities abound in natural language. Examples:)
 - Most nouns in English aren’t marked for CASE, but pronouns are.
 - Most verbs in English only distinguish two agreement categories (*3sing* and *non-3sing*), but *be* distinguishes more.
 - Most prepositions in English are transitive, but *here* and *there* are intransitive prepositions.
 - Most nominal words in English are 3rd person, but some (all pronouns) are 1st or 2nd person.
 - Most proper nouns in English are singular, but some, e.g. The San Francisco 49ers, are plural.

III. Some bits of linguistic analysis from Ch. 8

- The Case Constraint [**Slides:1**]
 - ? Is the Case Constraint adequate to describe the distribution of accusative case in English? (No, it doesn’t cover argument-marking prepositions or pronouns in isolation: *Who broke this? Him*. But it is a first approximation.)
 - ? What does the Case Constraint say about the distribution of nominative case in English? (Nothing.)
 - ? Would this kind of Case Constraint work for Icelandic or Wambaya? (No.)
- Case on nouns
 - ? Since CASE is really only relevant to pronouns and since we now have a type *pron-lxm* just for pronouns, why not limit CASE to the HEAD values of *pron-lxm* rather than of all nouns? [**Slides:2**]

- To actually limit the feature CASE to the HEAD values of *pron-lxm*, we'd have to make two subtypes of *noun*. [Slides:3]
- We could no longer simply say [CASE acc] for the COMPS value of a preposition, since that wouldn't unify with any non-pronouns. Instead, we'd have to introduce a messy disjunction, and repeat it everywhere a verb or preposition specifies a CASE value. [Slides:4]
- If it's not assigned as homework, go through Problem 8.1.

Chapter 8, Lecture Two

I. Lexemes and words

- ? What's the difference between a lexeme and a word?
 - A word is a pairing of form and meaning, e.g. *runs* and *ran* are different words.
 - A lexeme is an (abstract) entity that corresponds to a family of words, e.g. *runs*, *ran* and *run* are inflected forms that are realizations of one lexeme in the grammar.
- The big picture: [Slides:5]
- Word structures satisfy word descriptions, which are derived from basic (lexemic) lexical entries via (first) constraint inheritance and then (inflectional) lexical rule application.
- A lexeme description is of the form: $\langle xxx, [lexeme \dots] \rangle$
A word description is of the form: $\langle xxx, [word \dots] \rangle$ [Slides:6]
- Derivational lexical rules map lexeme descriptions to new lexeme descriptions, hence feeding inflection LRs.

II. Lexical rules and a streamlined lexicon

- Goal: Eliminate stipulation in the lexicon – each basic lexical entry should stipulate as little as possible, say the shape (e.g. love, dog), the semantics, and the lexeme type.
- The hierarchy does part of the work, and lexical rules take the next step.

- The lexicon in this book won't quite get there, as our basic lexical entries have to state which ARG-ST members are indexed to which semantic roles. This could be solved with a 'Linking Theory' that classified lexemes according to their semantic properties and stated generalizations about how the semantic arguments of certain classes of words are linked to ARG-ST members. For some recent proposals, see Wechsler 1995 and Davis 1996.
- Alternative conceptions of lexical rules:
 - Lexical rules are merely a bookkeeping device which reduces redundancy. In this case, if we take the perspective of someone building a parser, lexical rules are just a convenience. In the parser, the lexicon is precompiled, making run-time faster.
 - Lexical rules are productive devices for creating new lexical entries.

? What would require the second conception?

- Recursive lexical rules that permitted an infinite lexicon. No clear cases in English (though something like *anti-* or *post-* might be an example). Other languages have clearer examples. Agglutinative languages like Hungarian or Turkish are particularly striking examples.
- Production and comprehension of novel forms, e.g. *post-web*.

? What are some arguments for precompiling?

- Frequent forms are accessed more quickly (Oldfield and Wingfield (1965)).
- Go over how lexical rules work. [Slides:7]
 - A complete lexeme description (CLD) is one that has enriched the basic lexical entry through constraint inheritance. That is, a CLD has all the information from the lexical entry plus all the information that is unified in from constraints on its type and supertypes. A CLD is what serves as input to an LR.
 - To apply an LR, you unify the CLD input with the description on the left-hand side (LHS) of the rule. Only those CLDs that are compatible with the LHS can undergo the rule.

- The output is produced by then ‘unifying in’ the description on the right-hand side (RHS) of the rule. If anything specified there is incompatible, there is no output, i.e. the CLD fails to undergo the LR.
- There is one exception to the last statement: if both the LHS of the rule and RHS of the rule make explicit mention of a given feature, then the LR is to be interpreted as changing the value of that feature. That is, if the LHS says [FOO a] and the RHS says [FOO b], then the rule operates so as to ensure that the output of the rule is [FOO b].
- Similarly, the LHS of an LR may specify a type that is inconsistent with the type mentioned in the RHS. In this case, too, the LR’s output must have the type mentioned in the RHS. Inflectional rules are a good example of this, as they require that the input be of type *lxm* ensure that the output is of type *word*.

III. Examples

- Show the basic lexical entry for *dog* and the partial hierarchy above *cn-lxm*. [Slides:8–9].
- ? What does the lexeme description for *dog* look like after constraint inheritance? What are the values for PER, ANA, MODE? [Slides:10]
- This can then undergo the Plural Noun Lexical Rule. [Slides:11]
- The result is to change the type of the feature structure and to unify in a NUM specification. [Slides:12]
- Show the basic lexical entry for *love* and the partial hierarchy above *stv-lxm*. [Slides:13–14].
- ? What does the lexeme description for *love* look like after constraint inheritance? What are the values for HEAD, MODE? [Slides:15]
- This can then undergo the 3rd-Singular Verb Lexical Rule. [Slides:16]
- The result is to change the type of the feature structure and to unify in a NUM specification. [Slides:17]
- Show the basic lexical entry for *and* and the hierarchy over *conj-lxm*. [Slides:18–19]

- What does the lexeme description for *and* look like after constraint inheritance? (The same)
- This can only undergo the Constant Lexeme Lexical Rule. [Slides:20]
- The result is pretty much the same – only the type changes. [Slides:21]