

## Phrase Structure and Parsing as Search

### Informatics 2A: Lecture 13

Bonnie Webber

School of Informatics  
University of Edinburgh  
bonnie@inf.ed.ac.uk

23 October 2009

- 1 **Phrase Structure**
  - Heads and Phrases
  - Desirable Properties of a Grammar
  - A Fragment of English
- 2 **Grammars and Parsing**
  - Recursion
  - Structural Ambiguity
  - Recursive Descent Parsing
  - Search Strategies

#### Readings:

*J&M (2<sup>nd</sup> edition) Ch. 12 (intro – section 12.3)  
Ch. 13 (intro – section 13.3)*

## Heads and Phrases

In lectures 11 & 12, we looked at the terminal symbols in natural language.

Here, we'll look at the non-terminals, which are called **phrases**.

The **class** that a word belongs to is closely linked to the name of the **phrase** it customarily appears in:

Noun (N): Noun Phrase (NP)  
Adjective (A): Adjective Phrase (AP)  
Verb (V): Verb Phrase (VP)  
Preposition (P): Prepositional Phrase (PP)

In a X-phrase (eg **NP**), X (eg **N**) is called the **head**.

In English, the head tends to appear in the middle of a phrase.

## Heads and Phrases

English NPs are commonly of the form:

(Det) Adj\* **Noun** (PP | RelClause)\*

**NP**: *an angry duck that tried to bite me*, **head**: *duck*.

VPs are commonly of the form:

(Aux) Adv\* **Verb** Arg\* Adjunct\*

Arg → NP | PP

Adjunct → PP | AdvP | ...

**VP**: *would usually eat artichokes for dinner*, **head**: *eat*.

In Japanese, Korean, Hindi, Urdu, and other **head-final** languages, the head is at the end of its associated phrase.

In Irish, Welsh, Scots Gaelic and other **head-initial** languages, the head is at the beginning of its associated phrase.

## Desirable Properties of a Grammar

Recall that Chomsky specified two properties that make a grammar “interesting and satisfying”:

- It should be a **finite** specification of the strings of the language, rather than a list of its sentences.
- It should be **revealing**, in allowing strings to be associated with meaning (semantics) in a systematic way.

We can add another desirable property:

- It should capture **structural** and **distributional** properties of the language, including the location of the heads of phrases.

## Desirable Properties of a Grammar

With respect to these properties **context-free grammars** (CFGs) are a good approximation.

Recall that CFGs consist of rules of the form:

$$V \rightarrow a_1 a_2 \dots a_n$$

where  $V$  is a non-terminal symbol and  $a_i$  are either terminal or non-terminal symbols.

N.B. There are more modern grammar formalisms that better capture structural and distributional properties of human languages.

Some of these are described in the *Introduction to Cognitive Science* and in other courses we will mention in Lecture 31.

## A Tiny Fragment of English

Let's say we want to capture in a grammar the structural and distributional properties that give rise to sentences like:

A duck walked in the park.  
The man walked with a duck.  
You made a duck.  
You made her duck.  
A man with a telescope saw you.  
A man saw you with a telescope.  
You saw a man with a telescope.

We want to write **grammatical rules** that describe the phrases and **lexical rules** that describe what words appear in them.

## Patterns in the Tiny Fragment of English

Let's say we know that *walked*, *made* and *saw* are **verbs**.  
What appears to the left?

*a duck, the man, a man with a telescope, a man, you*

English is called an **SOV** language, because its standard sentence structure is **Subject-Verb-Object**.

All of these are Noun Phrases (NP), with all but *you* headed by a Noun *duck, man*.

Where else do we see NPs in these sentences?

*To the right of the prepositions with, in.*

We can use these observations to write a grammar for the tiny fragment.

## Grammar for the Tiny Fragment of English

Grammar G1 generates the sentences on the previous slide:

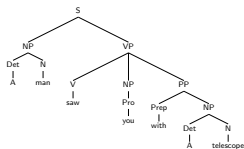
| Grammatical rules         | Lexical rules                                                 |
|---------------------------|---------------------------------------------------------------|
| $S \rightarrow NP VP$     | $Det \rightarrow a \mid the \mid her$ (determiner)            |
| $NP \rightarrow Det N$    | $N \rightarrow man \mid park \mid duck \mid telescope$ (noun) |
| $NP \rightarrow Det N PP$ | $Pro \rightarrow you$ (pronoun)                               |
| $NP \rightarrow Pro$      | $V \rightarrow saw \mid walked \mid made$ (verb)              |
| $VP \rightarrow V NP PP$  | $Prep \rightarrow in \mid with \mid for$ (preposition)        |
| $VP \rightarrow V NP$     |                                                               |
| $VP \rightarrow V$        |                                                               |
| $PP \rightarrow Prep NP$  |                                                               |

Does G1 produce a finite or an infinite number of sentences?

## Structural Ambiguity

Let's analyze the following two sentences with G1.

A man saw you with a telescope.  
You saw a man with a telescope.



## Recursion

**Recursion** in a grammar makes it possible to generate an **infinite** number of sentences.

In **direct recursion**, a non-terminal on the LHS of a rule also appears on its RHS. The following rules add direct recursion to G1:

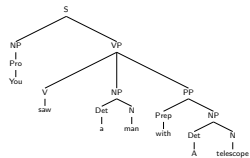
$VP \rightarrow VP Conj VP$   
 $Conj \rightarrow and \mid or$

In **indirect recursion**, a non-terminal on the LHS of a rule appears on the RHS of a derivation rooted in the non-terminal or substring that the original rule applied to. G1 contains indirect recursion:

$NP \rightarrow Det N PP$   
 $PP \rightarrow Prep NP$

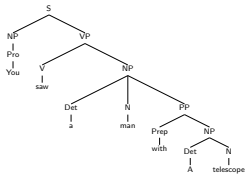
## Structural Ambiguity

You saw a man with a telescope.



## Structural Ambiguity

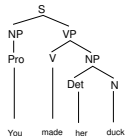
You saw a man with a telescope.



These two trees illustrate **attachment ambiguity**: the PP can be a part of the VP (*see with a telescope*) or of the NP (*a man with a telescope*). A PP **cannot** attach to an NP headed by a pronoun.

## Structural Ambiguity

Grammar G1 only gives us one analysis of *you made her duck*.



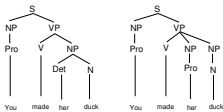
There is another, ditransitive (ie, two object) analysis of this sentence – one that underlies the pair:

What did you make for her?  
You made her duck.

## Structural Ambiguity

For this, G1 also needs rules like:

NP → N  
VP → V NP NP  
Pro → her



The resulting **structural ambiguity** (two analyses) is caused by **part of speech ambiguity**, (*her* is both a determiner and a pronoun), and the fact that *make* can take either **two arguments** or **three**.

## Structural Ambiguity

There is a third analysis as well, one that underlies the pair:

What did you make her do?  
You made her duck.

Here, the **small clause** (*her duck*) is the direct object of a verb.

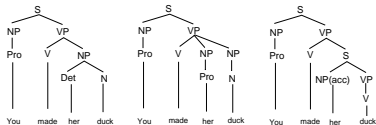
Similar **small clauses** are possible with verbs like *see*, *hear* and *notice*, but not *ask*, *want*, *persuade*, etc.

G1 needs a rule that requires accusative case-marking on the subject of a small clause and no tense on its verb.:

VP → V S1  
S1 → NP(acc) VP(untensed)  
NP(acc) → her | him | them

## Structural Ambiguity

Now we have three analyses for *you made her duck*:



How can we compute these analyses automatically?

## Recursive Descent Parsing

- 1 The top-level goal is to derive the start symbol (S).
- 2 Choose a **grammatical rule** with S as its LHS (e.g.,  $S \rightarrow NP VP$ ), and replace S with the RHS of the rule (the subgoals; e.g., NP and VP).
- 3 Choose a rule with the leftmost subgoal as its LHS (e.g.,  $NP \rightarrow Det N$ ). Replace the subgoal with the RHS of the rule.

## Recursive Descent Parsing

A **parser** is an algorithm that computes a structure for an input string given a grammar. All parsers have two fundamental properties:

- **Directionality:** the sequence in which the structures are constructed (e.g., top-down or bottom-up).
- **Search strategy:** the order in which the search space of possible analyses explored (e.g., depth-first, breadth-first).

A **recursive descent** parser treats a grammar as a specification of how to break down a top-level goal into subgoals. Therefore:

- Directionality = **top-down:** It starts from the start symbol of the grammar, and works its way down to the terminals.
- Search strategy = **depth-first:** It expands a given terminal as far as possible before proceeding to the next one.

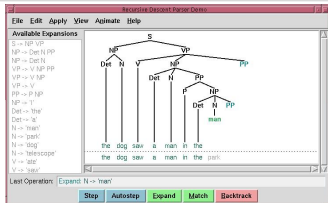
## Recursive Descent Parsing

- Whenever you reach a **lexical rule** (e.g.,  $Det \rightarrow the$ ), match its RHS against the current position in the input string.
  - If it matches, move on to the next position in the input.
  - If it doesn't, try the next lexical rule with the same LHS.
  - If there are no more lexical rules with the same LHS, backtrack to the most recent choice of grammatical rule and choose another rule with the same LHS.
  - If there are no more grammatical rules to choose from, back up to the previous subgoal.
- Iterate until the whole input string is consumed, or you fail to match one of the positions in the input. Backtrack on failure.

## Recursive Descent Parsing

We can see a recursive descent parser in action using NLTK:

```
>>> from nltk.app import rdparser
>>> rdparser()
```



## Shift-Reduce Parsing

**Search strategy** does not imply a particular **directionality**.

**Shift-reduce parsing** operates **bottom-up**, **shifting** terminal symbols from the input string onto a stack, and **reducing** them to the LHS side of a rule when the top elements in the stack match its RHS.

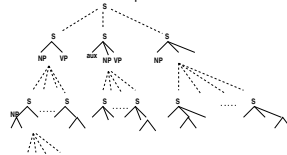
The LHS of the rule then gets pushed on the stack for later RHS-rule matching and subsequent reduction.

While its **directionality** is bottom-up, its **search strategy** is still depth-first.

```
>>> from nltk.app import srparser
>>> srparser()
```

## Search Strategies

Schematic view of the search space:



In **depth-first search**, the parser explores one branch of the search space at a time. If this branch is a dead-end, it needs to **backtrack**.

In **breadth-first search**, the parser explores all possible branches in parallel (often impossible due to memory requirements).

## Local and Global Ambiguity

A string can have more than one structural analysis (called **global ambiguity**) for one or both of two reasons:

- Grammatical rules allow for different attachment options;
- Lexical rules that allow a word to be in more than one word class.

Within a single analysis, some sub-strings can be analyzed in more than one way (called **local ambiguity**), even if not all these sub-string analyses are compatible with some global analysis of the entire string.

Local ambiguity is very common.

Recursive descent parsing is a wasteful way to deal with both types of ambiguity. We'll examine parsers that better handle ambiguity in Lectures 14–19.