Parsing

1. Grammars and parsing
2. Top-down and bottom-up parsing
3. Chart parsers
4. Bottom-up chart parsing
5. The Earley Algorithm

Syntax

syntax: from the Greek *syntaxis*, meaning “setting out together or arrangement.”

Refers to the way words are arranged together.

Why worry about syntax?

- The boy ate the frog.
- The frog was eaten by the boy.
- The frog that the boy ate died.
- The boy whom the frog was eaten by died.

Syntactic Analysis

Key ideas:

- **constituency**: groups of words may behave as a single unit or phrase
- **grammatical relations**: refer to the SUBJECT, OBJECT, INDIRECT OBJECT, etc.
- **subcategorization and dependencies**: refer to certain kinds of relations between words and phrases, e.g. *want* can be followed by an infinitive, but *find* and *work* cannot.

All can be modeled by various kinds of grammars that are based on context-free grammars.

Grammars and Parsing

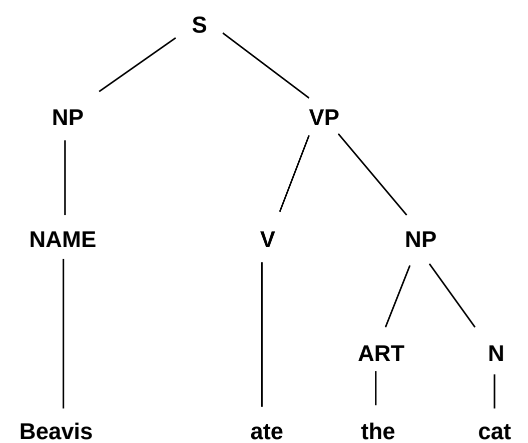
Need a **grammar:** a formal specification of the structures allowable in the language.

Need a **parser**: algorithm for assigning syntactic structure to an input sentence.

Sentence

Beavis ate the cat.

Parse Tree



CFG example

CFG's are also called phrase-structure grammars.
Equivalent to Backus-Naur Form (BNF).

1. S $\rightarrow$ NP VP
2. VP $\rightarrow$ V NP
3. NP $\rightarrow$ NAME
4. NP $\rightarrow$ ART N
5. NAME $\rightarrow$ Beavis
6. V $\rightarrow$ ate
7. ART $\rightarrow$ the
8. N $\rightarrow$ cat

- CFG's are *powerful* enough to describe most of the structure in natural languages.
- CFG's are *restricted* enough so that efficient parsers can be built.

CFG's

A context free grammar consists of:

1. a set of non-terminal symbols N
2. a set of terminal symbols Σ (disjoint from N)
3. a set of productions, P, each of the form $A \rightarrow \alpha$, where A is a non-terminal and α is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$
4. a designated start symbol S

Derivations

- If the rule $A \rightarrow \beta \in P$, and α and γ are strings in the set $(\Sigma \cup N)^*$, then we say that $\alpha A \gamma$ **directly derives** $\alpha\beta\gamma$, or $\alpha A \gamma \Rightarrow \alpha\beta\gamma$
- Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be strings in $(\Sigma \cup N)^*$, $m > 1$, such that

$$\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{m-1} \Rightarrow \alpha_m,$$

then we say that α_1 **derives** α_m or $\alpha_1 \stackrel{*}{\Rightarrow} \alpha_m$

L_G

The language L_G generated by a grammar G is the set of strings composed of terminal symbols that can be derived from the designated start symbol S.

$L_G = \{w | w \in \Sigma^*, S \stackrel{*}{\Rightarrow} w\}$

Parsing: the problem of mapping from a string of words to its parse tree according to a grammar G.

General Parsing Strategies

Grammar	Top-Down	Bottom-Up
1. S $\rightarrow$ NP VP	S $\rightarrow$ NP VP	$\rightarrow$ NAME ate the cat
2. VP $\rightarrow$ V NP	$\rightarrow$ NAME VP	$\rightarrow$ NAME V the cat
3. NP $\rightarrow$ NAME	$\rightarrow$ Beav VP	$\rightarrow$ NAME V ART cat
4. NP $\rightarrow$ ART N	$\rightarrow$ Beav V NP	$\rightarrow$ NAME V ART N
5. NAME $\rightarrow$ Beavis	$\rightarrow$ Beav ate NP	$\rightarrow$ NP V ART N
6. V $\rightarrow$ ate	$\rightarrow$ Beav ate ART N	$\rightarrow$ NP V NP
7. ART $\rightarrow$ the	$\rightarrow$ Beav ate the N	$\rightarrow$ NP VP
8. N $\rightarrow$ cat	$\rightarrow$ Beav ate the cat	$\rightarrow$ S