Syntactic Theory

Context-Free Grammars

Dr. Dan Flickinger & PD Dr. Valia Kordoni

Department of Computational Linguistics
Saarland University

November 4, 2011

Parsing: Assigning Structure to Sentences

We want to take in an input sentence and assign it a structure.

- **Input:** The man left the room.
- **Output:** (S (NP (DT The) (NN man)) (VP (VBD left) (NP (DT the) (NN room))))

But why this sort of representation?

- Why do we group words as we do?
- Where do we get these categories and what do they mean?

Syntax

Syntax = the study of the way that sentences are constructed from smaller units.
No "dictionary" for sentences → infinite number of possible sentences.

- The house is large.
- John believes that the house is large.
- Mary says that John believes that the house is large.

There are some basic principles of sentence organization:

- Linear order
- Hierarchical structure (Constituency)
- Subcategorization and Grammatical relations

Linear Order

Linear order = the order of words in a sentence.

A sentence has different meanings based on its linear order.

- John loves Mary.
- Mary loves John.

Languages vary as to what extent this is true, but linear order is still a guiding principle for organizing words into meaningful sentences.

Constituency

But we can't only use linear order to determine sentence organization. e.g. We can't simply say "The verb is the second word in the sentence."

- I eat at really fancy restaurants.
- Many executives eat at really fancy restaurants.

Constituency (cont.)

What are the “meaningful units” of the sentence *Many executives eat at really fancy restaurants*?

- Many executives
- really fancy
- really fancy restaurants
- at really fancy restaurants
- eat at really fancy restaurants

We refer to these meaningful groupings as **constituents** of a sentence.

There are many “tests” to determine what a constituent is.

Constituency tests

These tests are more like guidelines (i.e., they often work, but they sometimes don't)

- Preposed/Postposed constructions—i.e., can you move the grouping around?

(1) a. On September seventeenth, I'd like to fly from Atlanta to Denver.
b. I'd like to fly on September seventeenth from Atlanta to Denver.
c. I'd like to fly from Atlanta to Denver on September seventeenth.

- Pro-form substitution

(2) John has some very heavy books, but he didn't want them.
(3) I want to go home, and John wants to do so, too.

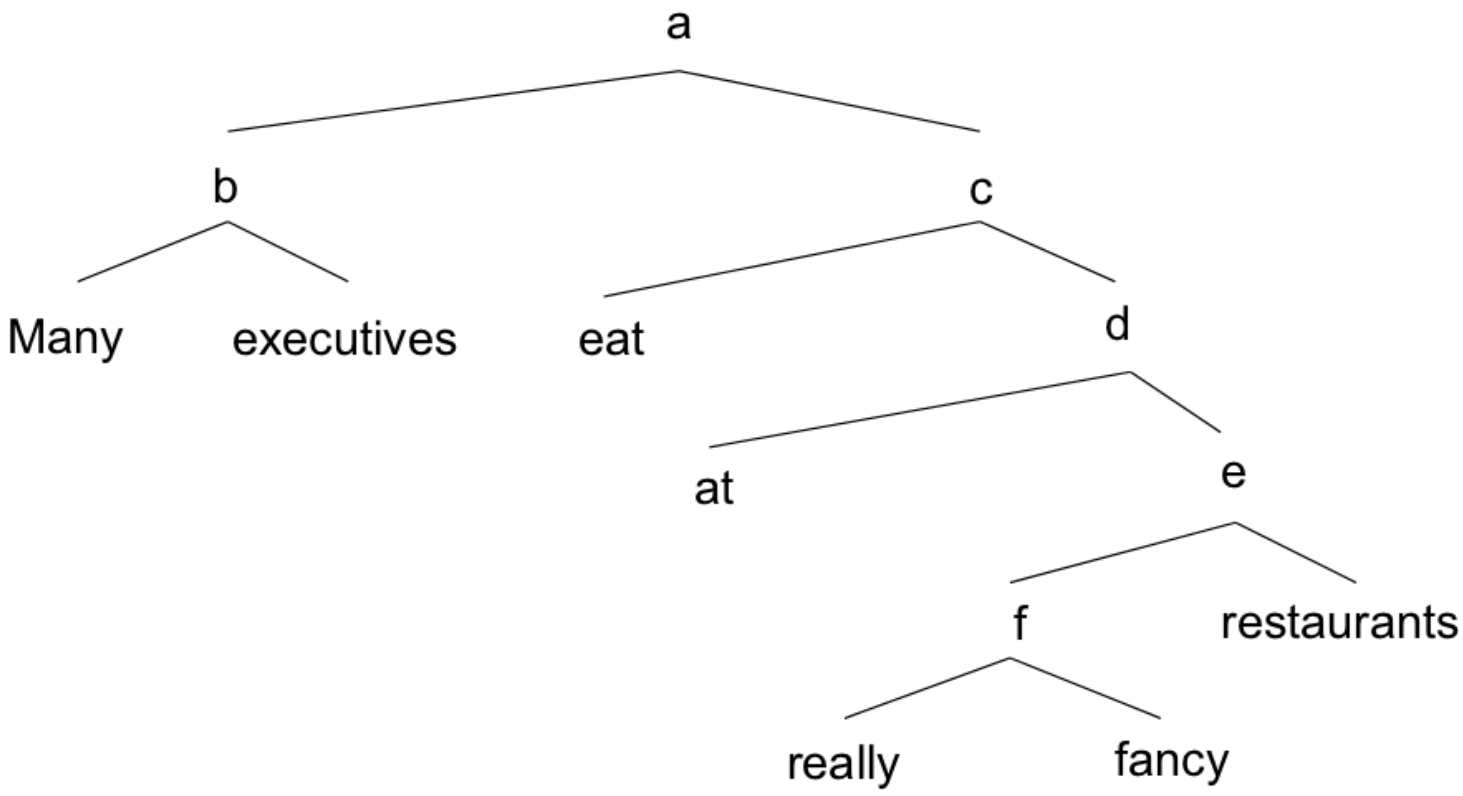
Hierarchical structure

Note that constituents appear within other constituents. We can represent this in a bracket form or in a **syntactic tree**

Bracket form:

[[Many executives] [eat [at [[really fancy] restaurants]]]]

Syntactic tree is on the next page ...


a
b
c
Many
executives
eat
d
at
e
f
restaurants
really
fancy

Categories

We would also like some way to say that *Many executives* and *really fancy restaurants* are the same type of grouping, or constituent, whereas at *really fancy restaurants* seems to be something else.

For this, we will talk about different **categories**

- Lexical
- Phrasal

Lexical categories

Lexical categories are simply word classes, or parts of speech. The main ones are:

- verbs: *eat*, *drink*, *sleep*, ...
- nouns: *gas*, *food*, *lodging*, ...
- adjectives: *quick*, *happy*, *brown*, ...
- adverbs: *quickly*, *happily*, *well*, *westward*
- prepositions: *on*, *in*, *at*, *to*, *into*, *of*, ...
- determiners/articles: *a*, *an*, *the*, *this*, *these*, *some*, *much*, ...
- conjunctions: *and*, *but*, *or*, *since*, *while*, ...

Determining lexical categories

How do we determine which category a word belongs to?

- **Distribution**: where these kinds of words can appear in a sentence.

  e.g. Nouns like *mouse* can appear after articles ("determiners") like *the*, while a verb like *eat* cannot.

- **Morphology**: what kinds of word prefixes/suffixes can a word take?

  e.g. Verbs like *walk* can take a *ed* ending to mark them as past tense. A noun like *mouse* cannot.

Closed & Open classes

We can add words to some classes, but not to others.

Open classes: new words can be easily added (tend to carry meaning):

- verbs
- nouns
- adjectives
- adverbs

Closed classes: new words cannot be easily added (tend to be function words):

- prepositions
- determiners
- conjunctions

Phrasal categories

We can also look at the distribution of phrases and see which ones behave in the same way, in order to assign them categories.

- The joggers ran through the park.

What other phrases can we put in place of *The joggers*?

Susan	students
you	most dogs
some children	a huge, lovable bear
my friends from Brazil	the people that we interviewed

Since all of these contain nouns, we consider these to be *noun phrases* (NPs).

Noun Phrases

Noun phrases, like other kinds of phrases, are **headed**: there is a designated item (the noun) which determines the properties of the whole phrase

- Before the noun, you can have determiners (and pre-determiners) and adjective phrases
- After the noun, you can have prepositional phrases, gerunds (and other verbal clauses), and relative clauses
- You can also have noun-noun compounds

➔ **General rule**: The category of the head word percolates up to the phrase level

Verb Phrases: Subcategorization

Verbs tend to drive the analysis of a sentence because they **subcategorize** for elements

We can say that verbs have **subcategorization frames**

- *sleep*: subject
- *find*: subject, object
- *show*: subject, object, second object
- *want*: subject, object, infinitive verb phrase
- *think*: subject, sentential complement

Grammatical relations

Grammatical relations are the basic relations between words in a sentence

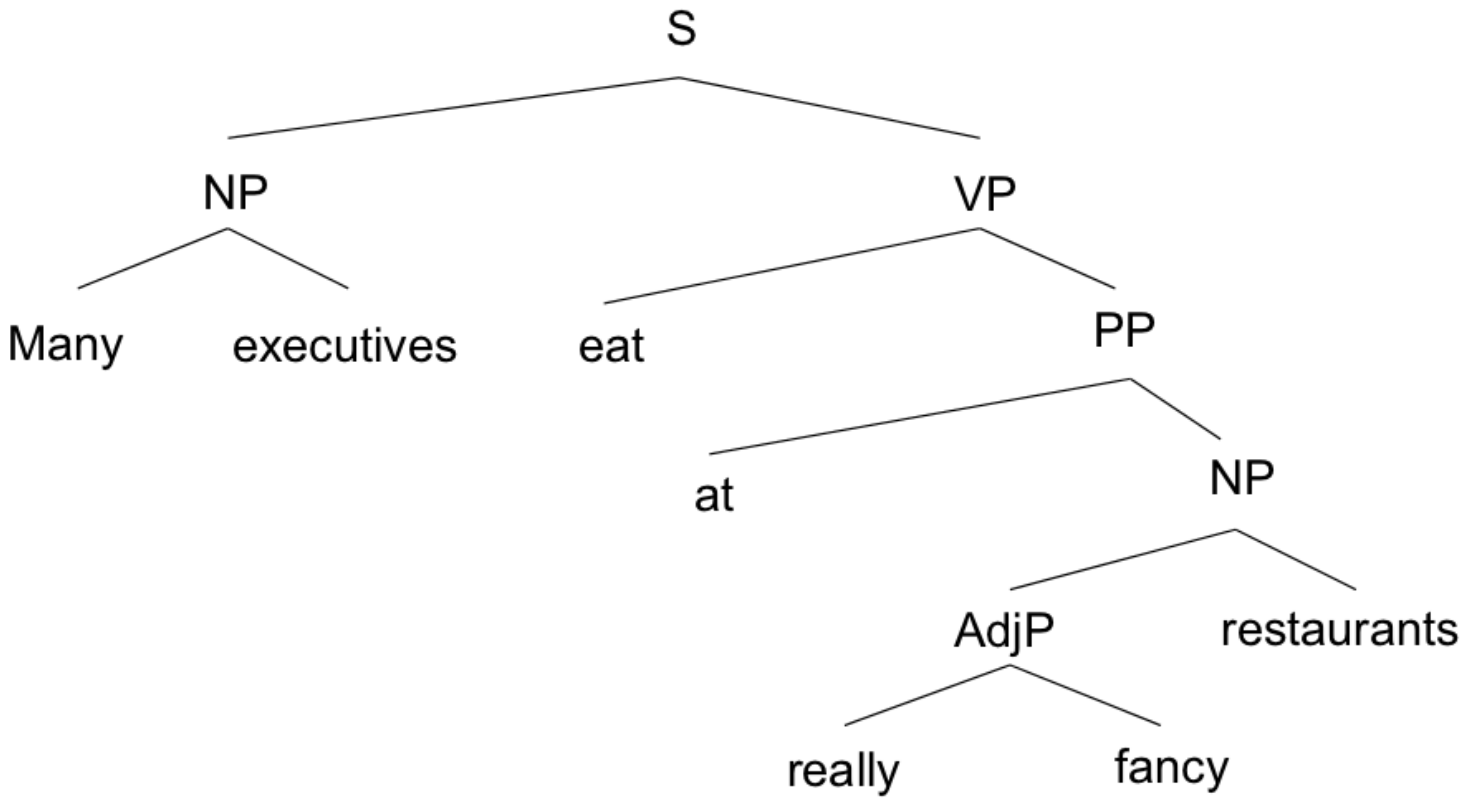
(4) She eats a mammoth breakfast.

- In this sentence, *She* is the SUBJECT, while *a mammoth breakfast* is the OBJECT
- In English, the SUBJECT must agree in person and number with the verb.

Building a tree

Other phrases work similarly, giving us the tree on the following page.

(S = sentence, VP = verb phrase, PP = prepositional phrase, AdjP = adjective phrase)


S
NP
VP
Many
executives
eat
PP
at
NP
AdjP
restaurants
really
fancy

Phrase Structure Rules (PSRs)

We can give rules for building these phrases. That is, we want a way to say that a determiner and a noun make up a noun phrase, but a verb and an adverb do not.

➔ **Phrase structure rules** (PSRs) are a way to build larger constituents from smaller ones.

e.g. S → NP VP

This says:

- A sentence (S) constituent is composed of a noun phrase (NP) constituent and a verb phrase (VP) constituent. (hierarchy)
- The NP must precede the VP. (linear order)

Some other English rules

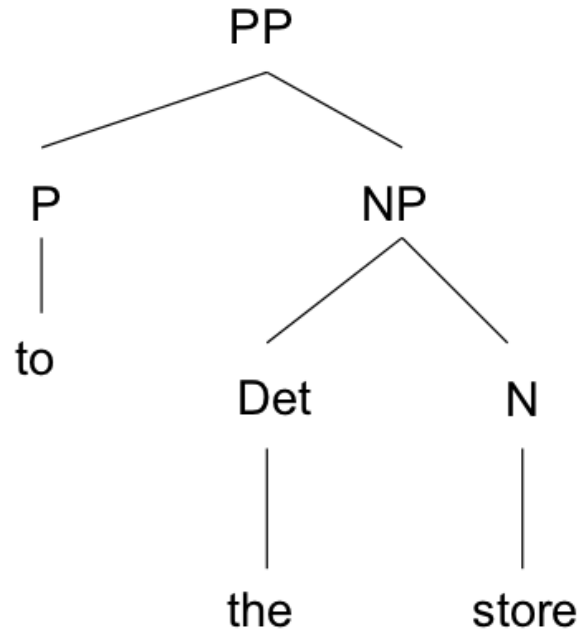
- NP → Det N (*the cat*, *a house*, *this computer*)
- NP → Det AdjP N (*the happy cat*, *a really happy house*)

  Can combine these by putting parentheses around AdjP, indicating that it is optional. (Note that this is a different use of parentheses than with regular expressions.)

  NP → Det (AdjP) N
- AdjP → (Adv) Adj (*really happy*)
- VP → V (*laugh*, *run*, *eat*)
- VP → V NP (*love John*, *hit the wall*, *eat cake*)
- VP → V NP NP (*give John the ball*)
- PP → P NP (*to the store*, *at John*, *in a New York minute*)
- NP → NP PP (*the cat on the stairs*)

PSRs and Trees

With every phrase structure rule (PSR), you can draw a tree for it.



PSR Practice

Try analyzing these sentences and drawing trees for them, based on the PSRs given above.

- The man in the kitchen drives a truck.
- That dang cat squeezed some fresh juice.
- The mouse in the corner by the stairs ate the cheese.

➔ **Important**: every part of the tree must correspond to a rule, and you have to use the rules you've been given

Properties of Phrase Structure Rules

- **generative** = a schematic strategy that describes a set of sentences completely.
- potentially **(structurally) ambiguous** = have more than one analysis

  (5) I [$_{V\ P}$ saw [$_{NP}$ [$_{NP}$ the man] [$_{PP}$ with the telescope]]]
  (6) I [$_{V\ P}$ saw [$_{NP}$ the man] [$_{PP}$ with the telescope]]
- **hierarchical** = categories have internal structure; they aren't just linearly ordered.
- **recursive** = property allowing for a rule to be reapplied (within its hierarchical structure).

  e.g. **NP** → NP PP
  PP → P **NP**

  The property of recursion means that the set of potential sentences in a language is **infinite**.

Coordination

One type of phrase we have not mentioned yet is the coordinate phrase, for example *John and Mary*

- Coordination can generally apply to any kinds of (identical) phrases
- This makes it ambiguous and cause problems for parsing

(7) I saw John and Mary left early.

➔ At some point, a parser has to decide between *and* joining NPs and joining Ss.

Context-free grammars

A **context-free grammar** (CFG) is essentially a collection of phrase structure rules, i.e., what we've been describing.

- It specifies that each rule must have:
  - a left-hand side (LHS): a single **non-terminal** element = (phrasal and lexical) categories
  - a right-hand side (RHS): a mixture of non-terminal and terminal elements **terminal** elements = actual words
- A CFG tries to capture a natural language completely.

Why "context-free"? Because these rules make no reference to any context surrounding them. i.e. you can't say "PP → P NP" *when* there is a verb phrase (VP) to the left.

Formal definition of CFGs

1. N: a set of non-terminal (phrasal) symbols, e.g., NP, VP, etc.
2. σ: a set of terminal (lexical) symbols

   N and σ are disjoint
3. P: a set of productions (rules) of the form A ! α, where A is a non-terminal and α is a collection of terminals and non-terminals
4. S: a designated start symbol

The language defined by a CFG

Derivation: A directly derives β if there is a rule of the form $A \rightarrow \beta$

- A chain of derivations can be established, such that A derives a string: $A \rightarrow \alpha_1 \rightarrow ... \rightarrow \alpha_m$

  – We abbreviate this by $A \rightarrow^* \alpha_m$

- We can thus define a language as the set of strings which are derivable from the designated start symbol S

  – $L = \{\alpha | S \rightarrow^* \alpha\}$

  – Sentences in language *L* are **grammatical**

Pushdown automata

Pushdown automaton = the computational implementation of a context-free grammar.

It uses a **stack** (its memory device) and has two operations:

- **push** = put an element onto the top of a stack.
- **pop** = take the topmost element from the stack.

This has the property of being **Last in first out (LIFO)**.

So, when you have a rule like "PP → P NP", you push NP onto the stack and then push P onto it. If you find a preposition (e.g. *on*), you pop P off of the stack and now you know that the next thing you need is an NP.

Automata

An **automaton** in general has three components:

- an **input tape**, divided into squares with a read-write head positioned over one of the squares
- an **auxiliary memory** characterized by two functions
  - fetch: memory configuration $\longrightarrow$ symbols
  - store: memory configuration $\times$ symbol $\longrightarrow$ memory configuration
- and a **finite-state control** relating the two components.

PDAs as FSAs with auxiliary memory

For the language $a^n b^n$:

Context-free grammar:

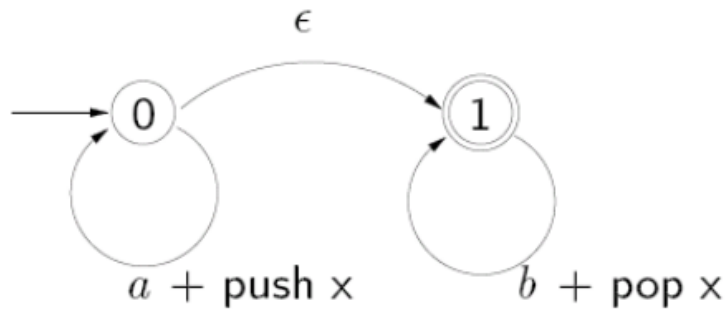
$N = \{\mathsf{S}\}$

$\Sigma = \{\mathsf{a}, \mathsf{b}\}$

$S = \mathsf{S}$

$$P = \left\{ \begin{array}{lcl} \mathsf{S} & \rightarrow & \mathsf{a\ S\ b} \\ \mathsf{S} & \rightarrow & \epsilon \end{array} \right\}$$

Push-down automaton:



The Chomsky Hierarchy

Type	Automaton		Grammar	
	Memory	Name	Rule	Name
0	Unbounded	TM	$\alpha \rightarrow \beta$	General rewrite
1	Bounded	LBA	$\beta\ A\ \gamma \rightarrow \beta\ \delta\ \gamma$	Context-sensitive
2	Stack	PDA	$A \rightarrow \beta$	Context-free
3	None	FSA	$A \rightarrow xB$, $A \rightarrow x$	Right linear (regular)

Abbreviations:

- TM: Turing Machine
- LBA: Linear-Bounded Automaton
- PDA: Push-Down Automaton
- FSA: Finite-State Automaton

Context-Sensitive Grammars

A rule of a **context-sensitive grammar**

- rewrites at most one non-terminal from the left-hand side.
- right-hand side of a rule required to be at least as long as the left-hand side, i.e. only contains rules of the form

$$\alpha \rightarrow \beta \text{ with } |\alpha| \leq |\beta|$$

and optionally $S \rightarrow \varepsilon$ with the start symbol S not occurring in any β.

A **linear-bounded automaton** is a

- finite state automaton, with an
- auxiliary memory which cannot exceed the length of the input string.

General Rewrite Grammars

- In a **general rewrite grammar** there are no restrictions on the form of a rewrite rule.
- A **turing machine** has an unbounded auxiliary memory.
- Any language for which there is a recognition procedure can be defined, but recognition problem is not decidable.

Context-Sensitive Example: Tree-Adjoining Grammar (TAG)

(Lexicalized) TAG has lexical entries which contain trees, and the subtrees are pasted together to form a complete tree