Phrase Structures and Syntax

ANLP: Lecture 11

Shay Cohen

School of Informatics
University of Edinburgh

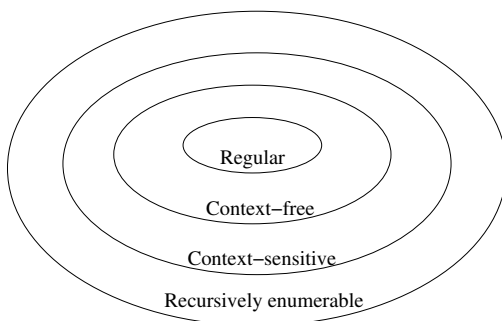8 October 2019

Until now...

- Focused mostly on *regular* languages
  - Finite state machines and transducers
  - n-gram models
  - Hidden Markov Models
  - Viterbi search and friends
- ... Next: going up one level in the Chomsky hierarchy

Recap: The Chomsky hierarchy

Regular

Context–free

Context–sensitive

Recursively enumerable

Side note: Is English Regular?

Centre-embedding

[The cat$_1$ likes tuna fish$_1$].
[The cat$_1$ [the dog$_2$ chased$_2$] likes tuna fish$_1$].
[The cat$_1$ [the dog$_2$ [the rat$_3$ bit$_3$] chased$_2$] likes tuna fish$_1$].

Consider $L = \{(\text{the N})^n \text{ TV}^m \text{ likes tuna fish} \mid n, m \geq 0\}$

where N = { cat, dog, rat, elephant, kangaroo ... }
      TV = { chased, bit, admired, ate, befriended ... }

Clearly L is regular. However, $L \cap$ English is the language

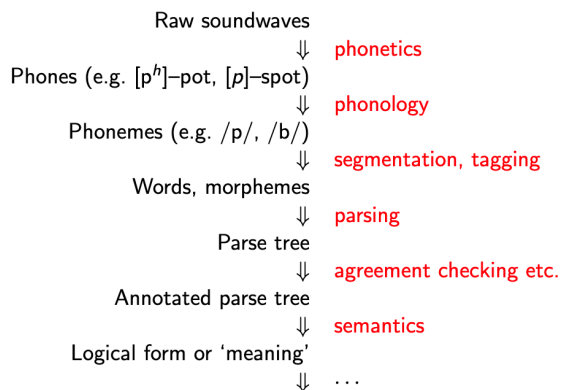$$\{(\text{the N})^n \text{ TV}^{n-1} \text{ likes tuna fish} \mid n \geq 1\}$$

Can use pumping lemma to show L is not regular.

Assumption 1. "(the N)^n TV^m likes tuna fish" is ungrammatical for $m \neq n - 1$.
Assumption 2. "(the N)^n TV^{n-1} likes tuna fish" is grammatical for all $n \geq 1$. (Is this reasonable? You decide!)

The NLP Pipeline

Raw soundwaves
⇓   phonetics
Phones (e.g. [p^h]–pot, [p]–spot)
⇓   phonology
Phonemes (e.g. /p/, /b/)
⇓   segmentation, tagging
Words, morphemes
⇓   parsing
Parse tree
⇓   agreement checking etc.
Annotated parse tree
⇓   semantics
Logical form or 'meaning'
⇓   . . .

Grammar Writing Exercise

Date: October 25 (Friday during lecture time)

You will write a grammar for the English language

There will be a competition between the grammars for "precision" and "recall"

You should be able to start working on your grammar by the end of this class

More details here:
`http://www.inf.ed.ac.uk/teaching/courses/anlp/cgw`
There will be prizes!

Computing meaning

A well-studied, difficult, and unsolved problem.

Fortunately, we know enough to have made partial progress (Watson won).

Over the next few weeks, we will work up to the study of systems that can assign **logical forms** that mathematically state the meaning of a sentence, so that they can be processed by machines.

Our first stop will be **natural language syntax**.

Natural language syntax

Syntax provides the scaffolding for semantic composition.

The brown dog on the mat saw the striped cat through the window.
The brown cat saw the striped dog through the window on the mat.

Do the two sentences above mean the same thing? What is the process by which you computed their meanings?

Constituents

Words in a sentence often form groupings that can combine with other units to produce meaning. These groupings, called **constituents** can often be identified by substitution tests (much like parts of speech!)

Kim [read a book], [gave it to Sandy], and [left]

You said I should read the book and [read it] I did.

Kim read [a very interesting book about grammar].

Heads and Phrases

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

▶ So far we have looked at terminals (words or POS tags).
▶ Today, we'll look at non-terminals, which correspond to phrases.
▶ The part of speech that a word belongs to is closely linked to the type of constituent that it is associated with.
▶ In a X-phrase (eg NP), the key occurrence of X (eg N) is called the head, and controls how the phrase interacts (both syntactically and semantically) with the rest of the sentence.
▶ In English, the head tends to appear in the middle of a phrase.

Constituents have structure

English NPs are commonly of the form:
(Det) Adj* **Noun** (PP | RelClause)*
NP: *the angry duck that tried to bite me*, head: *duck*.

VPs are commonly of the form:
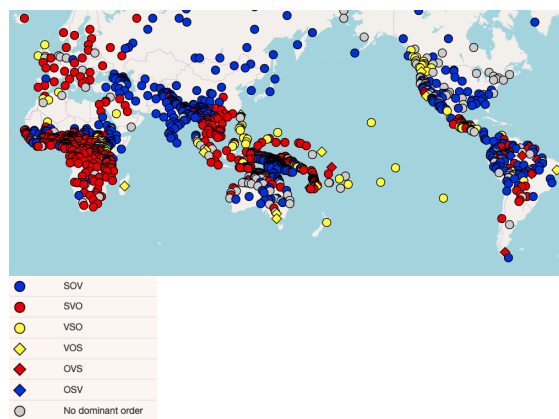(Aux) Adv* **Verb** Arg* Adjunct*
Arg → NP | PP
Adjunct → PP | AdvP | . . .
VP: *usually eats artichokes for dinner*, head: *eat*.

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.

WALS - Subject Verb Object order

●	SOV
●	SVO
○	VSO
◇	VOS
◆	OVS
◆	OSV
○	No dominant order

Taken from `https://wals.info/feature/81A#2/5.6/172.8`

Desirable Properties of a Grammar

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. (E.g. where heads of phrases are located; how a sentence transforms into a question; which phrases can float around the sentence.)

Desirable Properties of a Grammar

- Context-free grammars (CFGs) provide a pretty good approximation.
- Some features of NLs are more easily captured using mildly context-sensitive grammars, as well see later in the course.
- There are also more modern grammar formalisms that better capture structural and distributional properties of human languages. (E.g. combinatory categorial grammar.)
- Programming language grammars (such as the ones used with compilers, like LL(1)) aren't enough for NLs.

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.	NP,V,PP
The man walked with a duck.	NP,V,PP
You made a duck.	Pro,V,NP
You made her duck.	? Pro,V,NP
A man with a telescope saw you.	NP,PP,V,Pro
A man saw you with a telescope.	NP,V,Pro,PP
You saw a man with a telescope.	Pro,V,NP,PP

We want to write grammatical rules that generate these phrase structures, and lexical rules that generate the words appearing in them.

Grammar for the Tiny Fragment of English

Grammar G1 generates the sentences on the previous slide:

Grammatical rules	Lexical rules
S → NP VP	Det → a | the | her (determiners)
NP → Det N	N → man | park | duck | telescope (nouns)
NP → Det N PP	Pro → you (pronoun)
NP → Pro	V → saw | walked | made (verbs)
VP → V NP PP	Prep → in | with | for (prepositions)
VP → V NP	
VP → V	
PP → Prep NP	

Context-free grammars: formal definition

A context-free grammar (CFG) $\mathcal{G}$ consists of

- a finite set N of non-terminals,
- a finite set Σ of terminals, disjoint from N,
- a finite set P of productions of the form $X \to \alpha$, where $X \in N$, $\alpha \in (N \cup \Sigma)^*$,
- a choice of start symbol $S \in N$.

A sentential form is any sequence of terminals and nonterminals that can appear in a derivation starting from the start symbol.

Formal definition: The set of sentential forms derivable from $\mathcal{G}$ is the smallest set $\mathcal{S}(\mathcal{G}) \subseteq (N \cup \Sigma)^*$ such that

- $S \in \mathcal{S}(\mathcal{G})$
- if $\alpha X \beta \in \mathcal{S}(\mathcal{G})$ and $X \to \gamma \in P$, then $\alpha \gamma \beta \in \mathcal{S}(\mathcal{G})$.

The language associated with grammar is the set of sentential forms that contain only terminals.

Formal definition: The language associated with $\mathcal{G}$ is defined by $\mathcal{L}(\mathcal{G}) = \mathcal{S}(\mathcal{G}) \cap \Sigma^*$

A language $L \subseteq \Sigma^*$ is defined to be context-free if there exists some CFG $\mathcal{G}$ such that $L = \mathcal{L}(\mathcal{G})$.

Assorted remarks

- $X \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$ is simply an abbreviation for a bunch of productions $X \to \alpha_1$, $X \to \alpha_2$, ..., $X \to \alpha_n$.

- These grammars are called context-free because a rule $X \to \alpha$ says that an X can *always* be expanded to α, no matter where the X occurs.
  This contrasts with context-sensitive rules, which might allow us to expand X only in certain contexts, e.g. $bXc \to b\alpha c$.

- Broad intuition: context-free languages allow nesting of structures to arbitrary depth. E.g. brackets, begin-end blocks, if-then-else statements, subordinate clauses in English, . . .

Grammar for the Tiny Fragment of English

Grammar G1 generates the sentences on the previous slide:

Grammatical rules	Lexical rules
S → NP VP	Det → a | the | her (determiners)
NP → Det N	N → man | park | duck | telescope (nouns)
NP → Det N PP	Pro → you (pronoun)
NP → Pro	V → saw | walked | made (verbs)
VP → V NP PP	Prep → in | with | for (prepositions)
VP → V NP	
VP → V	
PP → Prep NP	

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

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:
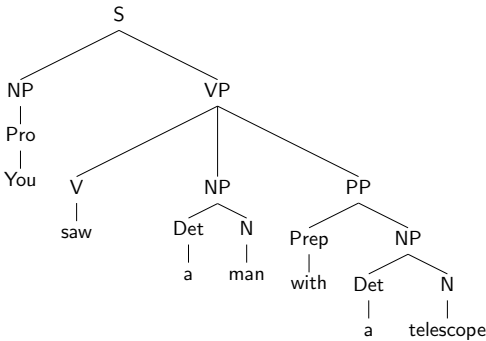
VP → VP Conj VP
Conj → and | or

In indirect recursion, some non-terminal can be expanded (via several steps) to a sequence of symbols containing that non-terminal:
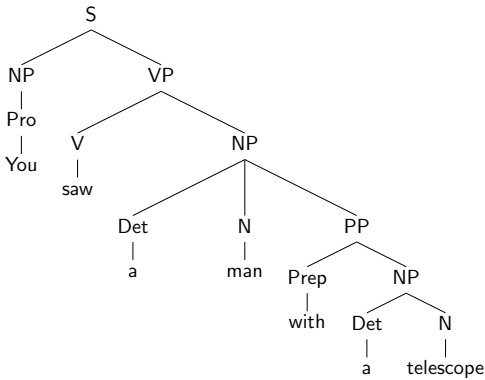
NP → Det N PP
PP → Prep NP

Structural Ambiguity
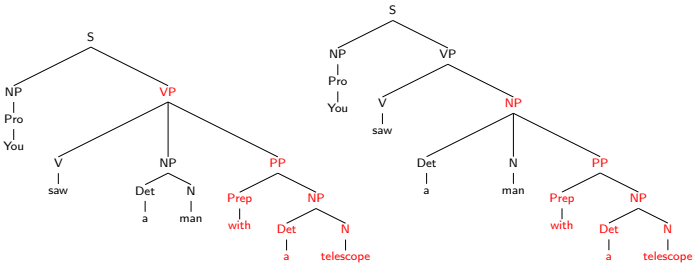
You saw a man with a telescope.

Structural Ambiguity

You saw a man with a telescope.
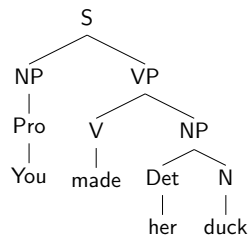
Structural Ambiguity

You saw a man with a telescope.

This illustrates attachment ambiguity: the PP can be a part of the VP or of the NP. Note that there's no POS ambiguity here.

Structural Ambiguity

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

```
              S
          ┌───┴────┐
         NP        VP
          │     ┌───┴───┐
         Pro    V       NP
          │     │     ┌──┴──┐
         You  made   Det    N
                      │      │
                     her   duck
```
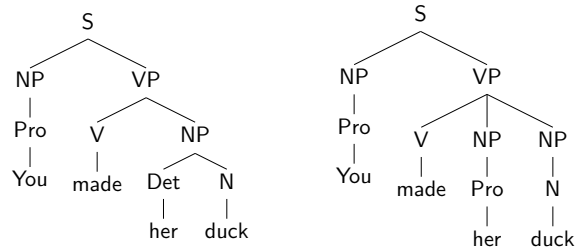
There is another, ditransitive (i.e., two-object) analysis of this sentence – one that underlies the pair:

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

Structural Ambiguity

For this alternative, G1 also needs rules like:

NP → N
VP → V NP NP
Pro → her

```
          S                            S
      ┌───┴────┐                   ┌───┴────┐
     NP        VP                 NP        VP
      │     ┌───┴───┐              │     ┌───┼───┐
     Pro    V       NP            Pro    V   NP  NP
      │     │     ┌──┴──┐          │     │   │    │
     You  made  Det     N         You  made Pro   N
                 │      │                    │     │
                her    duck                 her   duck
```

In this case, the structural ambiguity is rooted in POS ambiguity.

Structural Ambiguity

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

What did you make her do?
You made her duck. (move head or body quickly downwards)

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)
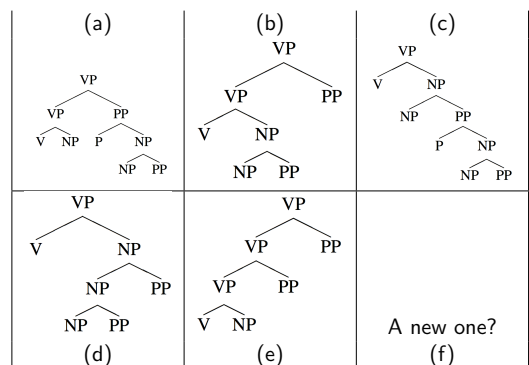NP(acc) → her | him | them

Structural Ambiguity

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

```
        S                      S                        S
    ┌───┴──┐              ┌────┴──┐               ┌─────┴──┐
   NP      VP            NP       VP             NP         VP
    │    ┌──┴──┐          │    ┌───┼──┐           │      ┌───┴──┐
   Pro   V    NP         Pro   V  NP  NP         Pro     V      S
    │         │               │             │            │   ┌──┴──┐
    │    ┌────┴──┐            Pro  N                    NP(acc)  VP
    │   Det     N                                              │
                                                               V

   You made her  duck     You made her duck     You  made her  duck
```
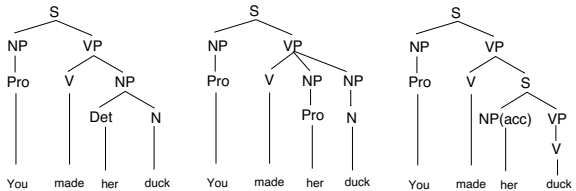
How can we compute these analyses automatically?

A Fun Exercise - Which is the VP?

(a)	(b)	(c)
VP	VP	VP
(d)	(e)	(f)
VP	VP	A new one?

saw the car from my house window with my telescope
E

Questions to Ask Yourself

- Can this context-free grammar formalism tackle all syntax phenomena?
- Where do we get the grammar from? How big would it be?
- How do we take a grammar and a sentence, and get a tree for the sentence from the grammar *efficiently*?
- How would we introduce probabilities into the use of a context-free grammar?

Summary

- We use CFGs to represent NL grammars
- Grammars need recursion to produce infinite sentences
- Most NL grammars have structural ambiguity
- A parser computes structure for an input automatically
- Recursive descent and shift-reduce parsing