Context Free Grammars

Ewan Klein
ewan@inf.ed.ac.uk

ICL — 31 October 2005

What is a Context Free Grammar?
- Some Definitions
- Trees

Example CFG for English
- Constituency
- Recursion
- Ambiguity

Challenges for CFGs
- Agreement
- Subcategorization
- Unbounded Dependencies

Summary

Syntax

- How words are combined to form phrases; and
- how phrases are combined to form sentences.

- New concept: Constituency
- Groups of words may behave as a single unit or constituent,
  - *They ate pizza* **at 8 pm**.
  - *They ate pizza* **then**. [substitution by pro-form]
  - **At 8 pm**, *they ate pizza.* [preposing]
  - *When did they eat pizza?* **At 8 pm**. [constituent answer]
  - *They ate pizza at 6 pm and* **at 8 pm**. [coordinate conjunct]

Syntax in CL

Syntactic analysis used to varying degrees in applications such as:

- Grammar Checkers
- Spoken Language Understanding
- Question Answering systems
- Information Extraction
- Automatic Text Generation
- Machine Translation

Typically, fine-grained syntactic analysis is a prerequisite for fine-grained semantic interpretation.

Context Free Grammars (CFGs)

- Capture constituency and ordering;
- formalise descriptive linguistic work of the 1940s and '50s;
- are widely used in linguistics.

- CFGs are somewhat biased towards languages like English which have relatively fixed word order.
- Most modern linguistic theories of grammar incorporate some notions from context free grammar.

Context Free Grammars (CFGs)

Formally, a CFG is a 4-tuple $\langle N, \Sigma, P, S \rangle$, where

- N is a set of non-terminal symbols (e.g., syntactic categories)
- Σ a set of terminal symbols (e.g., words)
- P a set of productions (rules) of the form $A \rightarrow \alpha$, where
  - A is a non-terminal, and
  - α is a string of symbols from the set $(\Sigma \cup N)^\star$ (i.e., both terminals and non-terminals)
- a designated start symbol S

Example CFG

Let $G = \langle N, \Sigma, P, S \rangle$, where

- N = {S, NP, VP, Det, Nom, V, N}
- Σ = {*a*, *flight*, *left*}
- P = { S $\rightarrow$ NP VP,
  NP $\rightarrow$ Det Nom,
  Nom $\rightarrow$ N,
  VP $\rightarrow$ V,
  Det $\rightarrow$ *a*,
  N $\rightarrow$ *flight*,
  V $\rightarrow$ *left* }
- S = S.

NP = 'noun phrase', VP = 'verb phrase', Det = 'determiner', Nom = 'Nominal', N = 'noun', V = 'verb'.

Derivations

- A derivation of a string from non-terminal A is the result of successively applying productions (from G) to A:

NP	
Det Nom	by NP$\longrightarrow$ Det Nom
a Nom	by Det$\longrightarrow$ *a*
a N	by Nom$\longrightarrow$ N
a *flight*	by N$\longrightarrow$ *flight*

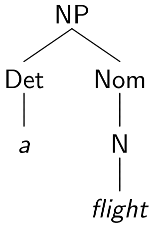
- Can also write: NP$\Rightarrow$ Det Nom$\Rightarrow$ *a* Nom$\Rightarrow$ *a* N$\Rightarrow$ *a* *flight*, where $\Rightarrow$ means "directly derives" or "yields in one rule application".
- G generates *a flight* (as a string of category NP).

Grammars and Languages

- CFG is an abstract model for associating structures with strings;
- **not** intended as model of how humans produce sentences.
- Sentences that can be derived by a grammar G belong to the formal language defined by G, and are called Grammatical Sentences with respect to G.
- Sentences that cannot be derived by G are Ungrammatical Sentences with respect to G..
- The language L_G defined by grammar G is the set of strings composed of terminal symbols that are derivable from the start symbol: $L_G = \{w | w \in \Sigma^\star \text{ and } S \text{ derives } w\}$

Parse Trees

- Derivations can also be visualized as parse trees (or constituent structure trees), e.g.



- Trees express:
  - hierarchical grouping into constituents
  - grammatical category of constituents
  - left-to-right order of constituents

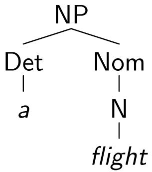
Parse Trees, cont.

- Trees can also be written as labeled bracketings:

```
[NP
    [Det a]
    [Nom [N flight]]]
```

- **Dominance**: node *x* dominates node *y* if there's a connected sequence of branches descending from *x* to *y*. E.g.
  - NP dominates non-terminals Det, Nom and N
- **Immediate Dominance**: node *x* immediately dominates node *y* if *x* dominates *y* and there's no distinct node between *x* and *y*. E.g.
  - NP immediately dominates Det and Nom.

Parse Trees, cont.



- A node is called the daughter of the node which immediately dominates it.
- Distinct nodes immediately dominated by the same node are called sisters.
- A node which is not dominated by any other node is called the root node.
- Nodes which do not dominate any other nodes are called leaves.

CFG: As opposed to what?

- Regular Grammars:
  - All rules of the form $A \rightarrow xB$ or $A \rightarrow x$.
  - Equivalent to Regular Expressions.
  - Regarded as too weak to capture lingistic generalizations.
- Context Sensitive Grammars:
  - Allows rules of the form $XAY \rightarrow X\alpha Y$; i.e., the way in which A is expanded can depend on the context X_Y.
  - Regarded as 'too strong' — can describe languages that aren't possible human languages.
  - Regular languages $\subset$ Context Free languages $\subset$ Context Sensitive languages

Grammars and Constituency

- A huge amount of skilled effort goes into the development of grammars for human languages — can only scratch the surface here.
- There's lot's of research into English syntactic structure — but also lots of disagreement.
- Various criteria for determining constituency:
  - substitution by pro-forms
  - preposing
  - constituent answers
  - coordination
- Some clear-cut decisions, but quite a lot of unclear ones too.

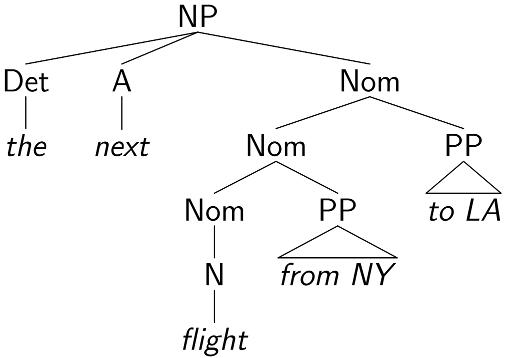
A Tiny Lexicon

N	→	*flight* | *passenger* | *trip* | *morning* | ...
V	→	*is* | *prefers* | *like* | *need* | *depend* | *fly*
A	→	*cheapest* | *non-stop* | *first* | *latest*
		other | *direct* | ...
Pro	→	*me* | *I* | *you* | *it* | ...
PropN	→	*Alaska* | *Baltimore* | *Los Angeles*
		Chicago | *United* | *American* | ...
Det	→	*the* | *a* | *an* | *this* | *these* | *that* | ...
P	→	*from* | *to* | *on* | *near* | ...
Conj	→	*and* | *or* | *but* | ...

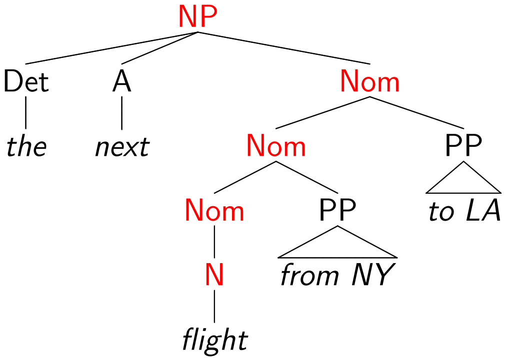
A Tiny Grammar

S	→	NP VP	*I + want a morning flight*
NP	→	Pro	*I*
	|	PropN	*Los Angeles*
	|	Det A Nom	*the + next + passenger*
	|	Det Nom	*a + flight*
Nom	→	Nom PP	*flight + to Los Angeles*
	|	N Nom	*morning + flight*
	|	N	*trip*
VP	→	VP PP	*leave + in the morning*
	|	V NP	*want + a flight*
	|	V NP PP	*sell + a ticket + to me*
	|	V PP	*depend + on the weather*
PP	→	P NP	*from + Los Angeles*

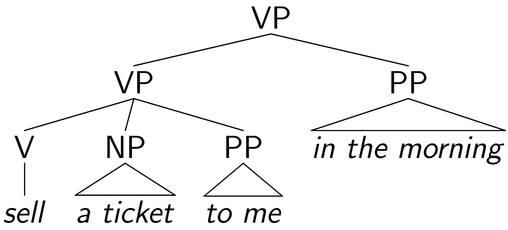
Example Noun Phrase



Example Noun Phrase: Heads



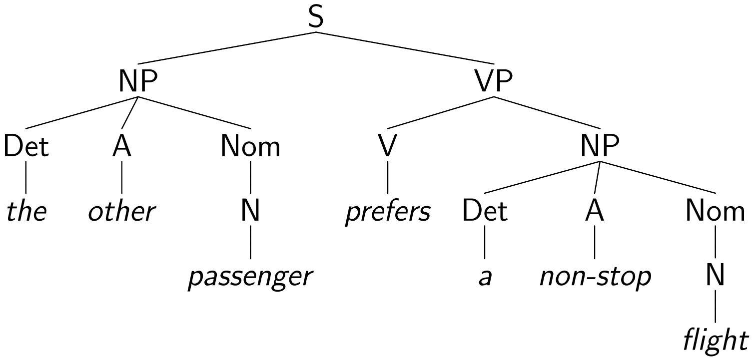
Example Verb Phrase



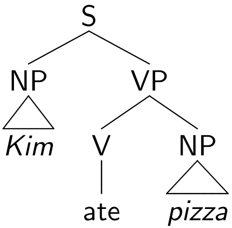
Arguments vs. Modifiers

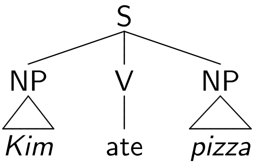
- **Arguments**: 'essential participants' in an event
- **Modifiers**: optional additional information about an event
- As with other linguistic distinctions, some clear cases and some unclear ones.
- We've chosen to reflect the distinction in the parse trees:
  - arguments are sisters of V (or N)
  - modifiers are sisters of VP (or Nom)

Example Sentence



Are VPs Constituents?





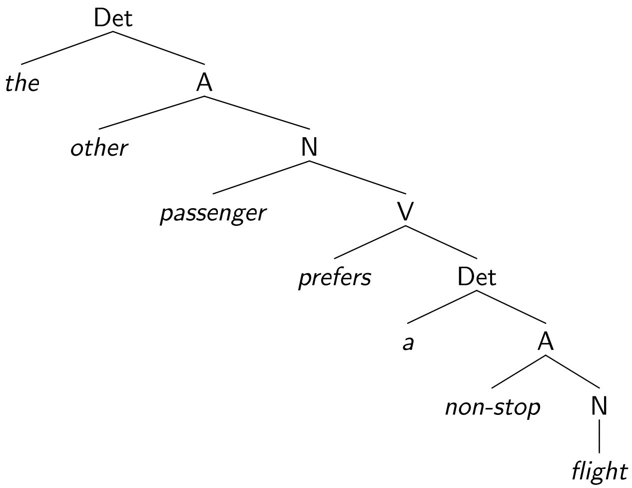
- *Kim ate pizza and Lee did too.*
- *What did Kim do? Ate pizza.*
- *Kim said she would eat pizza, and eat pizza she did.*

Constituency in REs?

- Regular Expression:

  `(the|a)(other|non-stop)?(passenger|flight)prefers`
  `(the|a)(other|non-stop)?(passenger|flight)`
- No explicit representation of NP which can be 're-used' in different positions in a sentence.

Constituency in Regular Grammars?



Recursive Structures

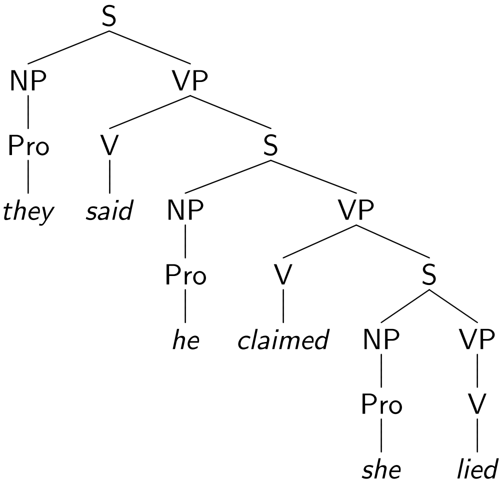
- There is no upper bound on the length of a grammatical English sentence.
  - Therefore the set of English sentences is infinite.
- A grammar is a finite statement about well-formedness.
  - To account for an infinite set, it has to allow iteration (e.g., X^+) or recursion.
- Recursive rules: where the non-terminal on the left-hand side of the arrow in a rule also appears on the right-hand side of a rule.

Recursive Structures, cont.

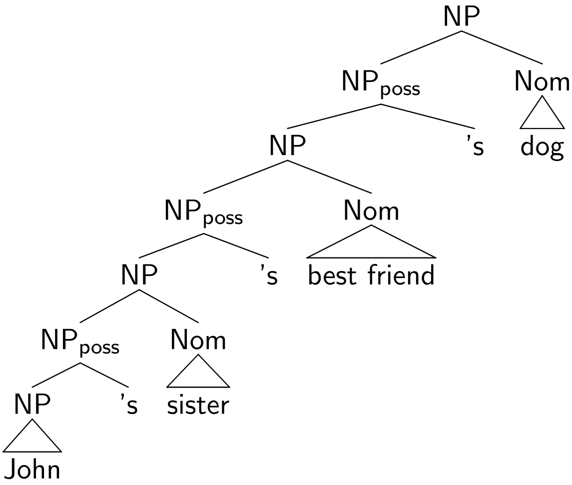
Direct recursion:	
Nom → Nom PP	*flight to Boston*
VP → VP PP	*departed Miami at noon*

Indirect recursion:	
S → NP VP	
VP → V S	*said that the flight was late*

Recursion Example: Sentential Complements



Recursion Example: Possessives



Coordination

NP → NP *and* NP
VP → VP *and* VP
S → S *and* S

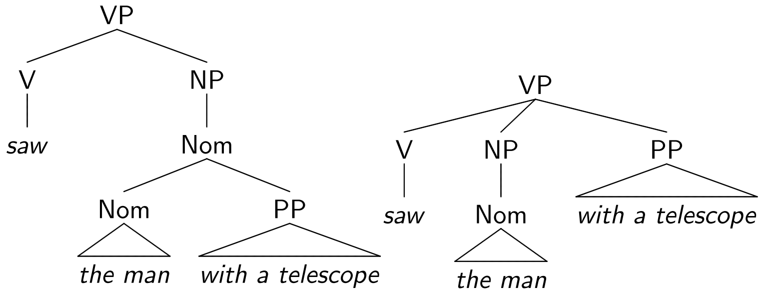
- *I need* [[$_{NP}$ *the times*] *and* [$_{NP}$ *the fares*]].
- *a flight* [[$_{VP}$ *departing at 9a.m.*] *and* [$_{VP}$ *returning at 5p.m.*]]
- [[$_{S}$ *I depart on Wednesday*] *and* [$_{S}$ *I'll return on Friday*]].

Any phrasal constituent XP can be conjoined with a constituent of the same type —XP to form a new constituent of type XP.
General schema:

XP → XP *and* XP

Syntactic Ambiguity

- Many kinds of syntactic (structural) ambiguity.
- PP attachment has received much attention:



PP Ambiguity

- Different structures naturally correspond to different semantic interpretations ('readings')
- Arises from independently motivated syntactic rules:
  VP → V . . . PP
  Nom → Nom PP
- However, also strong, lexically influenced, preferences:
  - *I bought* [*a book* [*on linguistics*]]
  - *I bought* [*a book*] [*on sunday*]

Problem Areas for CFGs

- Agreement
- Subcategorization
- 'Movement' or unbounded dependencies

Number Agreement

In English, some determiners agree in number with the head noun:

- *This dog*
- *Those dogs*
- **Those dog*
- **This dogs*

And verbs agree in number with their subjects:

- *What flights leave in the morning?*
- **What flight leave in the morning?*

Number Agreement, cont.

Expand our grammar with multiple sets of rules?

$NP_{sg} \rightarrow Det_{sg}\ N_{sg}$
$NP_{pl} \rightarrow Det_{pl}\ N_{pl}$
$S_{sg} \rightarrow NP_{sg}\ VP_{sg}$
$S_{pl} \rightarrow NP_{pl}\ VP_{pl}$
$VP_{sg} \rightarrow V_{sg}$ (NP) (NP) (PP)
$VP_{pl} \rightarrow V_{pl}$ (NP) (NP) (PP)

- worse when we add person and even worse in languages with richer agreement (e.g., three genders).
- lose generalizations about nouns and verbs — can't say property P is true of all words of category V.

Subcategorization

Verbs have preferences for the kinds of constituents (cf. arguments) they co-occur with.

- *I found the cat.*
- **I disappeared the cat.*

- *It depends* [$_{\text{PP}}$ *on the question*].
- **It depends* [$_{\text{PP}}$ {*to/from/by*} *the question*].

A traditional subcategorization of verbs:

- transitive (takes a direct object NP)
- intransitive

In more recent approaches, there might be as many as a hundred subcategorizations of verb.

Subcategorization, cont.

More examples:

- *find* is subcategorized for an NP (can take an NP complement)
- *want* is subcategorized for an NP or an infinitival VP
- *bet* is subcategorized for NP NP S

A listing of the possible sequences of complements is called the subcategorization frame for the verb.
As with agreement, the obvious CFG solution yields rule explosion:

$\text{VP} \rightarrow \text{V}_{\text{intr}}$
$\text{VP} \rightarrow \text{V}_{\text{tr}}\ \text{NP}$
$\text{VP} \rightarrow \text{V}_{\text{ditr}}\ \text{NP NP}$

Example Subcategorization Frames

Frame	Verb	Example
__	*eat, sleep*	*I want to eat*
NP	*prefer, find, leave,*	*Find [$_{NP}$ the flight from Pittsburgh to Boston]*
NP NP	*show, give*	*Show [$_{NP}$ me] [$_{NP}$ airlines with flights from Pittsburgh]*
NP PP	*help, load,*	*Can you help [$_{NP}$ me] [$_{PP}$ with a flight]*
VP_{inf}	*prefer, want, need*	*I would prefer [$_{VP_{inf}}$ to go by United airlines]*
S	*mean*	*Does this mean [$_{S}$ AA has a hub in Boston]?*

Unbounded Dependency (or Movement) Constructions

- **I gave __to the driver.*
- *I gave some money to the driver.*

- $5 [*I gave __to the driver*], (*and $1 I gave to the porter*).
- *He asked how much* [*I gave __to the driver*].
- *I forgot about the money which* [*I gave __to the driver*].

- *How much did you think* [*I gave __to the driver*]?
- *How much did you think he claimed* [*I gave __to the driver*]?
- *How much did you think he claimed that I said* [*I gave __to the drive*
- . . .

Summary

- CFGs capture hierarchical structure of constituents in natural language.
- More powerful than REs, and can express recursive structure.
- Hard to get a variety of linguistic generalizations in 'vanilla' CFGs, though this can be mitigated with use of features (not covered here).
- Building a CFG for a reasonably large set of English constructions is a lot of work!

Reading

- Jurafsky & Martin, Chapter 9
- **Parsing** tutorial in NLTK-Lite