

Speech and Language Processing

Formal Grammars, Parsing

Chapter 12, start 13

Today

- Formal Grammars
 - Context-free grammar
 - Grammars for English
- Treebanks
- Start Parsing

Syntax

- By grammar, or syntax, we have in mind the kind of implicit knowledge of your native language that you had mastered by the time you were 3 years old without explicit instruction
- Not the kind of stuff you were later taught in “grammar” school

Syntax

- Why should you care?
- Grammars (and parsing) are key components in many applications
 - Grammar checkers
 - Dialogue management
 - Question answering
 - Information extraction
 - Machine translation

Syntax

- Key notions that we'll cover
 - Constituency
 - Grammatical relations and Dependency
 - Heads
- Key formalism
 - Context-free grammars
- Resources
 - Treebanks

Constituency

- The basic idea here is that groups of words within utterances can be shown to act as single units.
- And in a given language, these units form coherent classes that can be shown to behave in similar ways
 - With respect to their internal structure
 - And with respect to other units in the language

Constituency

- **Internal structure**
 - We can describe an internal structure to the class (might have to use disjunctions of somewhat unlike sub-classes to do this).
- **External behavior**
 - For example, we can say that noun phrases can come before verbs

Constituency

- For example, it makes sense to say that the following are all *noun phrases* in English...

Harry the Horse
the Broadway coppers
they

a high-class spot such as Mindy's
the reason he comes into the Hot Box
three parties from Brooklyn

- Why? One piece of evidence is that they can all precede verbs.
 - This is external evidence

Grammars and Constituency

- Of course, there's nothing easy or obvious about how we come up with right set of constituents and the rules that govern how they combine...
- That's why there are so many different theories of grammar and competing analyses of the same data.
- The approach to grammar, and the analyses, adopted here are very generic (and don't correspond to any modern linguistic theory of grammar).

Context-Free Grammars

- Context-free grammars (CFGs)
 - Also known as
 - Phrase structure grammars
 - Backus-Naur form
- Consist of
 - Rules
 - Terminals
 - Non-terminals

Context-Free Grammars

- **Terminals**
 - We'll take these to be words (for now)
- **Non-Terminals**
 - The constituents in a language
 - Like noun phrase, verb phrase and sentence
- **Rules**
 - Rules are equations that consist of a single non-terminal on the left and any number of terminals and non-terminals on the right.

Some NP Rules

- Here are some rules for our noun phrases

NP → *Det Nominal*

NP → *ProperNoun*

Nominal → *Noun* | *Nominal Noun*

- Together, these describe two kinds of NPs.
 - One that consists of a determiner followed by a nominal
 - And another that says that proper names are NPs.
 - The third rule illustrates two things
 - An explicit disjunction
 - Two kinds of nominals
 - A recursive definition
 - Same non-terminal on the right and left-side of the rule

LO Grammar

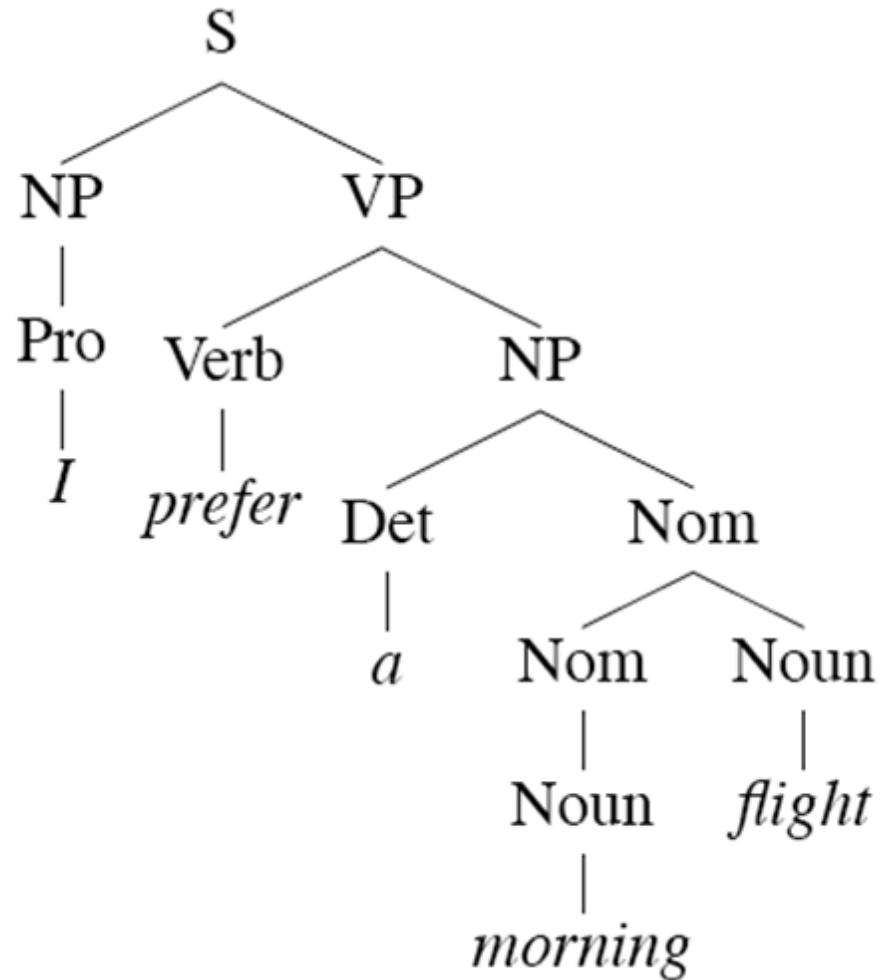
Grammar Rules	Examples
$S \rightarrow NP VP$	I + want a morning flight
$NP \rightarrow$ <i>Pronoun</i> <i>Proper-Noun</i> <i>Det Nominal</i>	I Los Angeles a + flight
$Nominal \rightarrow$ <i>Nominal Noun</i> <i>Noun</i>	morning + flight flights
$VP \rightarrow$ <i>Verb</i> <i>Verb NP</i> <i>Verb NP PP</i> <i>Verb PP</i>	do want + a flight leave + Boston + in the morning leaving + on Thursday
$PP \rightarrow$ <i>Preposition NP</i>	from + Los Angeles

Generativity

- As with FSAs and FSTs, you can view these rules as either analysis or synthesis machines
 - Generate strings in the language
 - Reject strings not in the language
 - Impose structures (trees) on strings in the language

Derivations

- A derivation is a sequence of rules applied to a string that *accounts* for that string
 - Covers all the elements in the string
 - Covers only the elements in the string



Definition

- More formally, a CFG consists of

N a set of **non-terminal symbols** (or **variables**)

Σ a set of **terminal symbols** (disjoint from N)

R a set of **rules** or productions, each of the form $A \rightarrow \beta$,
where A is a non-terminal,

β is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$

S a designated **start symbol**

Parsing

- Parsing is the process of taking a string and a grammar and returning a (multiple?) parse tree(s) for that string
- It is completely analogous to running a finite-state transducer with a tape
 - It's just more powerful
 - Remember this means that there are languages we can capture with CFGs that we can't capture with finite-state methods
 - More on this when we get to Ch. 13.

An English Grammar Fragment

- Sentences
- Noun phrases
 - Agreement
- Verb phrases
 - Subcategorization
- Prepositional Phrases

Sentence Types

- Declaratives: *A plane left.*

$S \rightarrow NP VP$

- Imperatives: *Leave!*

$S \rightarrow VP$

- Yes-No Questions: *Did the plane leave?*

$S \rightarrow Aux NP VP$

- WH Questions: *When did the plane leave?*

$S \rightarrow WH-NP Aux NP VP$

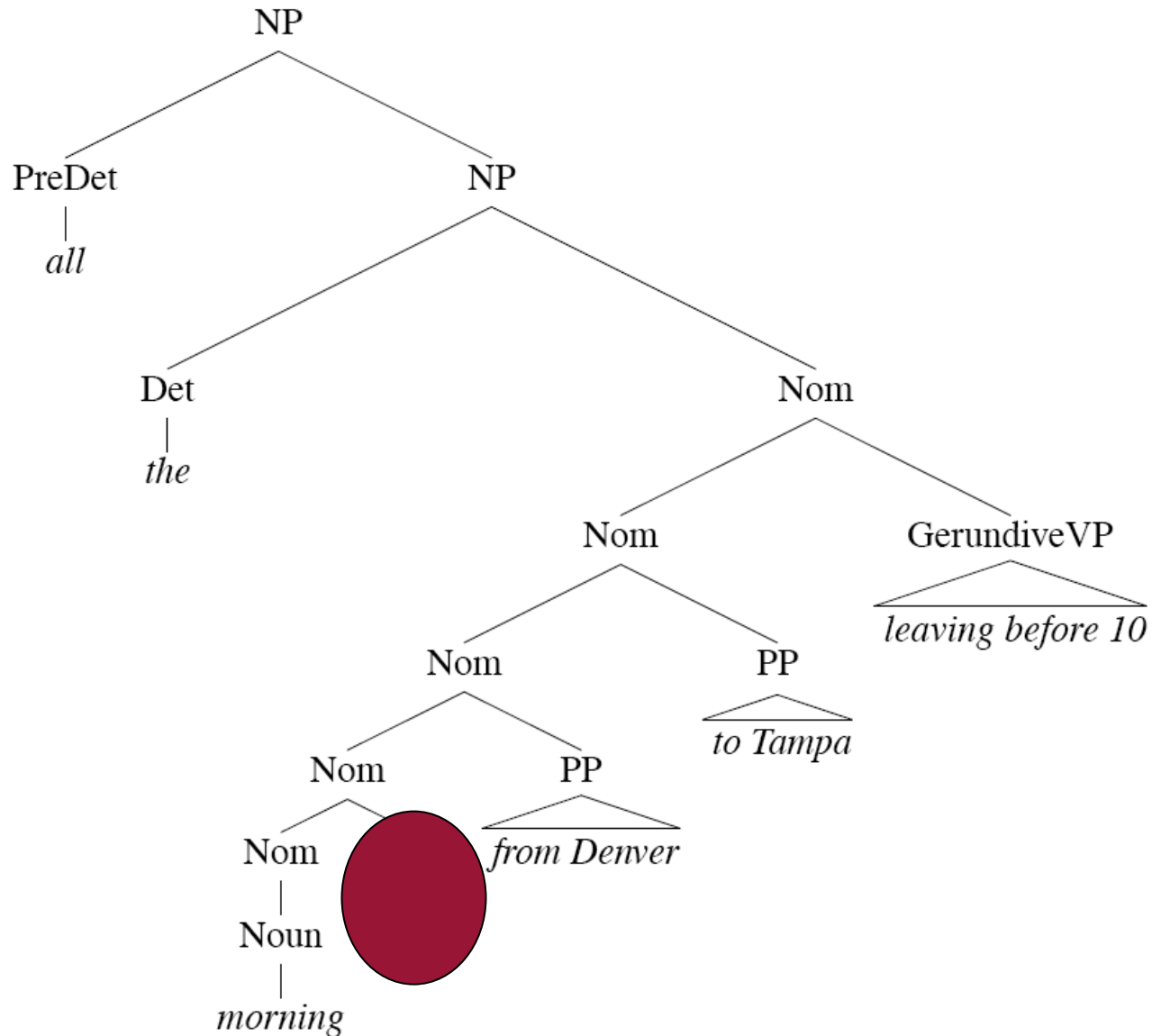
Noun Phrases

- Let's consider the following rule in more detail...

NP → Det Nominal

- Most of the complexity of English noun phrases is hidden in this rule.
- Consider the derivation for the following example
 - *All the morning flights from Denver to Tampa leaving before 10*

Noun Phrases



NP Structure

- Clearly this NP is really about *flights*. That's the central critical noun in this NP. Let's call that the *head*.
- We can dissect this kind of NP into the stuff that can come before the head, and the stuff that can come after it.

Determiners

- Noun phrases can start with determiners...
- Determiners can be
 - Simple lexical items: *the, this, a, an*, etc.
 - A car
 - Or simple possessives
 - John's car
 - Or complex recursive versions of that
 - John's sister's husband's son's car

Nominals

- Contains the head and any pre- and post-modifiers of the head.
 - Pre-
 - Quantifiers, cardinals, ordinals...
 - Three cars
 - Adjectives and Aps
 - large cars
 - Ordering constraints
 - Three large cars
 - ?large three cars

Postmodifiers

- Three kinds
 - Prepositional phrases
 - From Seattle
 - Non-finite clauses
 - Arriving before noon
 - Relative clauses
 - That serve breakfast
- Same general (recursive) rule to handle these
 - *Nominal* → *Nominal PP*
 - *Nominal* → *Nominal GerundVP*
 - *Nominal* → *Nominal RelClause*

Agreement

- By *agreement*, we have in mind constraints that hold among various constituents that take part in a rule or set of rules
- For example, in English, determiners and the head nouns in NPs have to agree in their number.

This flight

Those flights

*This flights

*Those flight

Problem

- Our earlier NP rules are clearly deficient since they don't capture this constraint
 - *NP* → *Det Nominal*
 - Accepts, and assigns correct structures, to grammatical examples (*this flight*)
 - But its also happy with incorrect examples (**these flight*)
 - Such a rule is said to *overgenerate*.
 - We'll come back to this in a bit

Verb Phrases

- English *VPs* consist of a head verb along with 0 or more following constituents which we'll call *arguments*.

VP → *Verb* disappear

VP → *Verb NP* prefer a morning flight

VP → *Verb NP PP* leave Boston in the morning

VP → *Verb PP* leaving on Thursday

Subcategorization

- But, even though there are many valid VP rules in English, not all verbs are allowed to participate in all those VP rules.
- We can subcategorize the verbs in a language according to the sets of VP rules that they participate in.
- This is a modern take on the traditional notion of transitive/intransitive.
- Modern grammars may have 100s or such classes.

Subcategorization

- Sneeze: John sneezed
- Find: Please find [a flight to NY]_{NP}
- Give: Give [me]_{NP}[a cheaper fare]_{NP}
- Help: Can you help [me]_{NP}[with a flight]_{PP}
- Prefer: I prefer [to leave earlier]_{TO-VP}
- Told: I was told [United has a flight]_S
- ...

Subcategorization

- *John sneezed the book
 - *I prefer United has a flight
 - *Give with a flight
-
- As with agreement phenomena, we need a way to formally express the constraints

Why?

- Right now, the various rules for VPs *overgenerate*.
 - They permit the presence of strings containing verbs and arguments that don't go together
 - For example
 - **VP** \rightarrow **V NP** therefore
Sneezed the book is a VP since "sneeze" is a verb and "the book" is a valid NP

Possible CFG Solution

- Possible solution for agreement.
- Can use the same trick for all the verb/VP classes.
- SgS -> SgNP SgVP
- PIS -> PINp PIVP
- SgNP -> SgDet SgNom
- PINP -> PIDet PINom
- PIVP -> PIV NP
- SgVP -> SgV Np
- ...

CFG Solution for Agreement

- It works and stays within the power of CFGs
- But its ugly
- And it doesn't scale all that well because of the interaction among the various constraints explodes the number of rules in our grammar.

The Point

- CFGs appear to be just about what we need to account for a lot of basic syntactic structure in English.
- But there are problems
 - That can be dealt with adequately, although not elegantly, by staying within the CFG framework.
- There are simpler, more elegant, solutions that take us out of the CFG framework (beyond its formal power)
 - LFG, HPSG, Construction grammar, XTAG, etc.
 - Chapter 15 explores the unification approach in more detail

Treebanks

- Treebanks are corpora in which each sentence has been paired with a parse tree (presumably the right one).
- These are generally created
 - By first parsing the collection with an automatic parser
 - And then having human annotators correct each parse as necessary.
- This generally requires detailed annotation guidelines that provide a POS tagset, a grammar and instructions for how to deal with particular grammatical constructions.

Penn Treebank

- Penn TreeBank is a widely used treebank.

■ Most well known is the Wall Street Journal section of the Penn TreeBank.

- 1 M words from the 1987-1989 Wall Street Journal.

```
( (S ( ' ' ' ' )
  (S-TPC-2
    (NP-SBJ-1 (PRP We) )
    (VP (MD would)
      (VP (VB have)
        (S
          (NP-SBJ (-NONE- *-1) )
          (VP (TO to)
            (VP (VB wait)
              (SBAR-TMP (IN until)
                (S
                  (NP-SBJ (PRP we) )
                  (VP (VBP have)
                    (VP (VBN collected)
                      (PP-CLR (IN on)
                        (NP (DT those)(NNS assets))))))))))
          ( , , ) ( ' ' ' ' )
          (NP-SBJ (PRP he) )
          (VP (VBD said)
            (S (-NONE- *T*-2) ))
          ( . . ) )
```

Treebank Grammars

- Treebanks implicitly define a grammar for the language covered in the treebank.
- Simply take the local rules that make up the sub-trees in all the trees in the collection and you have a grammar.
- Not complete, but if you have decent size corpus, you'll have a grammar with decent coverage.

Treebank Grammars

- Such grammars tend to be very flat due to the fact that they tend to avoid recursion.
 - To ease the annotators burden
- For example, the Penn Treebank has 4500 different rules for VPs. Among them...

VP → VBD PP

VP → VBD PP PP

VP → VBD PP PP PP

VP → VBD PP PP PP PP

Grammar Summary

- Context-free grammars can be used to model various facts about the syntax of a language.
- When paired with parsers, such grammars constitute a critical component in many applications.
- Constituency is a key phenomena easily captured with CFG rules.
 - But agreement and subcategorization do pose significant problems
- Treebanks pair sentences in corpus with their corresponding trees.