FORMAL GRAMMARS

Ing. R. Tedesco. PhD, AA 20-21

(mostly from: Speech and Language Processing - Jurafsky and Martin)

Today

- Context-free grammar
- Grammars for English
- Treebanks

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** → *constituent*
- And, in a given language, these units form **coherent classes** that can be 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
  - Constituents have internal structures
  - We can describe such internal structures
- External behavior
  - For example, we can say that noun phrases can come before verbs

Constituency

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

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	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
- Non-terminals
  - The *constituents*
    - Like noun phrase, verb phrase, etc.
  - The *pre-terminals* (e.g., POS tags)
  - A special symbol *S* to start from
- Rules
  - a single non-terminal → any number of terminals and non-terminals

Some NP Rules

- Here are some rules for our noun phrases

$$
\begin{aligned}
NP &\rightarrow Det\ Nominal \\
NP &\rightarrow ProperNoun \\
Nominal &\rightarrow Noun \mid Nominal\ Noun
\end{aligned}
$$

- 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

The L_0 Grammar

GRAMMAR RULES			EXAMPLES
S	$\rightarrow$	$NP\ VP$	I + want a morning flight
NP	$\rightarrow$	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	$\rightarrow$	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	$\rightarrow$	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	$\rightarrow$	*Preposition NP*	from + Los Angeles

The L_0 Grammar

GRAMMAR LEXICON

$$
\begin{array}{rcl}
\textit{Noun} & \rightarrow & \textit{flights} \mid \textit{breeze} \mid \textit{trip} \mid \textit{morning} \mid \ldots \\
\textit{Verb} & \rightarrow & \textit{is} \mid \textit{prefer} \mid \textit{like} \mid \textit{need} \mid \textit{want} \mid \textit{fly} \\
\textit{Adjective} & \rightarrow & \textit{cheapest} \mid \textit{non-stop} \mid \textit{first} \mid \textit{latest} \\
 & & \mid \textit{other} \mid \textit{direct} \mid \ldots \\
\textit{Pronoun} & \rightarrow & \textit{me} \mid \textit{I} \mid \textit{you} \mid \textit{it} \mid \ldots \\
\textit{Proper-Noun} & \rightarrow & \textit{Alaska} \mid \textit{Baltimore} \mid \textit{Los Angeles} \\
 & & \mid \textit{Chicago} \mid \textit{United} \mid \textit{American} \mid \ldots \\
\textit{Determiner} & \rightarrow & \textit{the} \mid \textit{a} \mid \textit{an} \mid \textit{this} \mid \textit{these} \mid \textit{that} \mid \ldots \\
\textit{Preposition} & \rightarrow & \textit{from} \mid \textit{to} \mid \textit{on} \mid \textit{near} \mid \ldots \\
\textit{Conjunction} & \rightarrow & \textit{and} \mid \textit{or} \mid \textit{but} \mid \ldots
\end{array}
$$

Pre-terminals (POS tags)

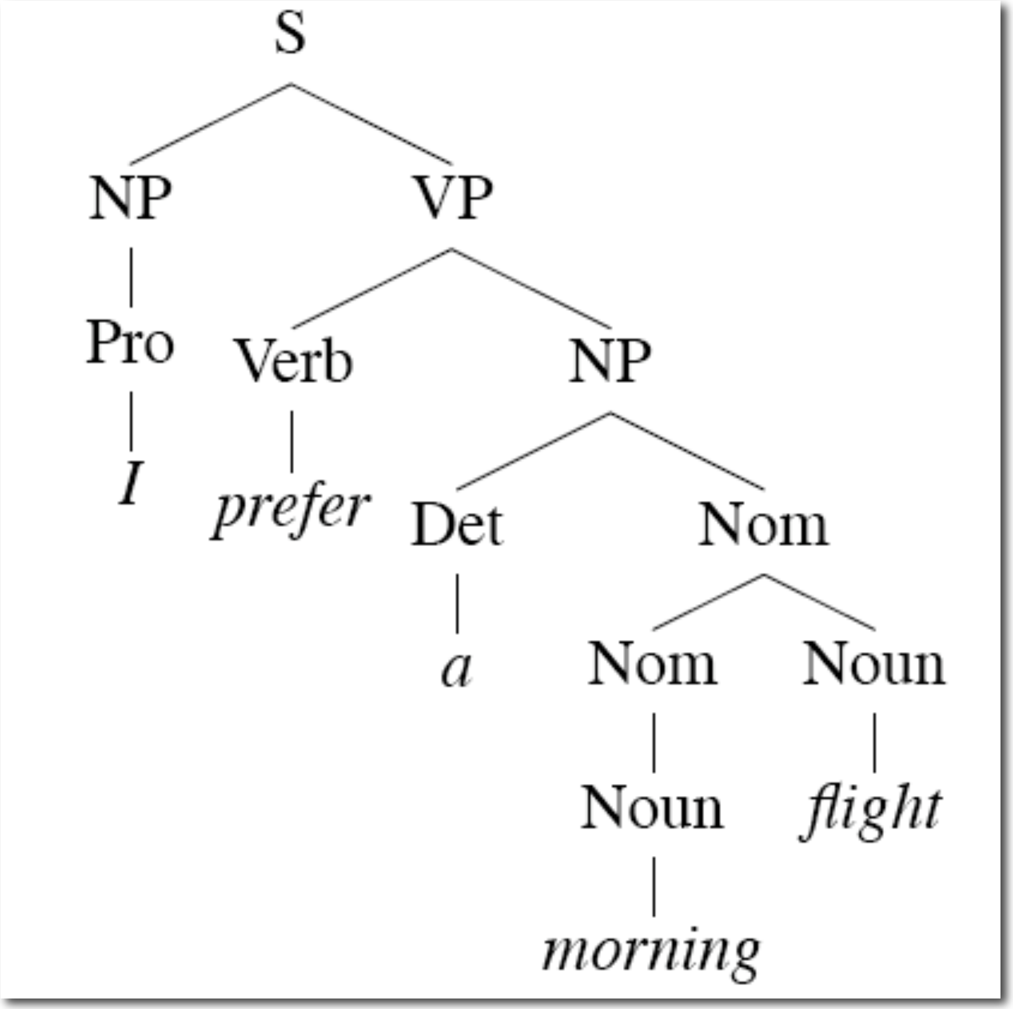
Terminals (words)

Generativity

- 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** is the process of taking a string and a grammar and returning a (multiple?) parse tree(s) for that string

Noun Phrases

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

  NP → Det Nominal

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	*This flights
Those flights	*Those flight

*=wrong sentence

Problem

- Our 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 zero or more following constituents which we'll call *arguments*.

VP	$\rightarrow$	$Verb$	disappear
VP	$\rightarrow$	$Verb\ NP$	prefer a morning flight
VP	$\rightarrow$	$Verb\ NP\ PP$	leave Boston in the morning
VP	$\rightarrow$	$Verb\ PP$	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 and complements.
- 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]$_{TO\text{-}VP}$
- …

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

Possible CFG Solution

- Possible solution for agreement and subcategorization: create ad-hoc sub-classes

- *S → NP VP*
  - *SG_S → SG_NP SG_VP*
  - *PL_S → PL_NP PL_VP*
- *NP → Det Nom*
  - *SG_NP → SG_Det SG_Nom*
  - *PL_NP → PL_Det PL_Nom*
- *VP → V NP*
  - *PL_VP → PL_V NP*
  - *SL_VP → SG_V NP*
- …
  - …

CFG Solution for Agreement

- It works and stays within the power of CFGs
- But it’s ugly and it doesn’t scale because of the interaction among the various constraints explodes the number of rules in our grammar
- CFGs appear to be just about what we need to account for a lot of basic syntactic structure in
- Other solutions:
  - Feature-based grammars

    S→NP VP

    <NP AGREEMENT>=<VP AGREEMENT>

    A feature associated to NP and VP
    Words carry features
  - LFG, HPSG, Construction grammar, XTAG, etc.
  - Do not try to find a solution!

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.

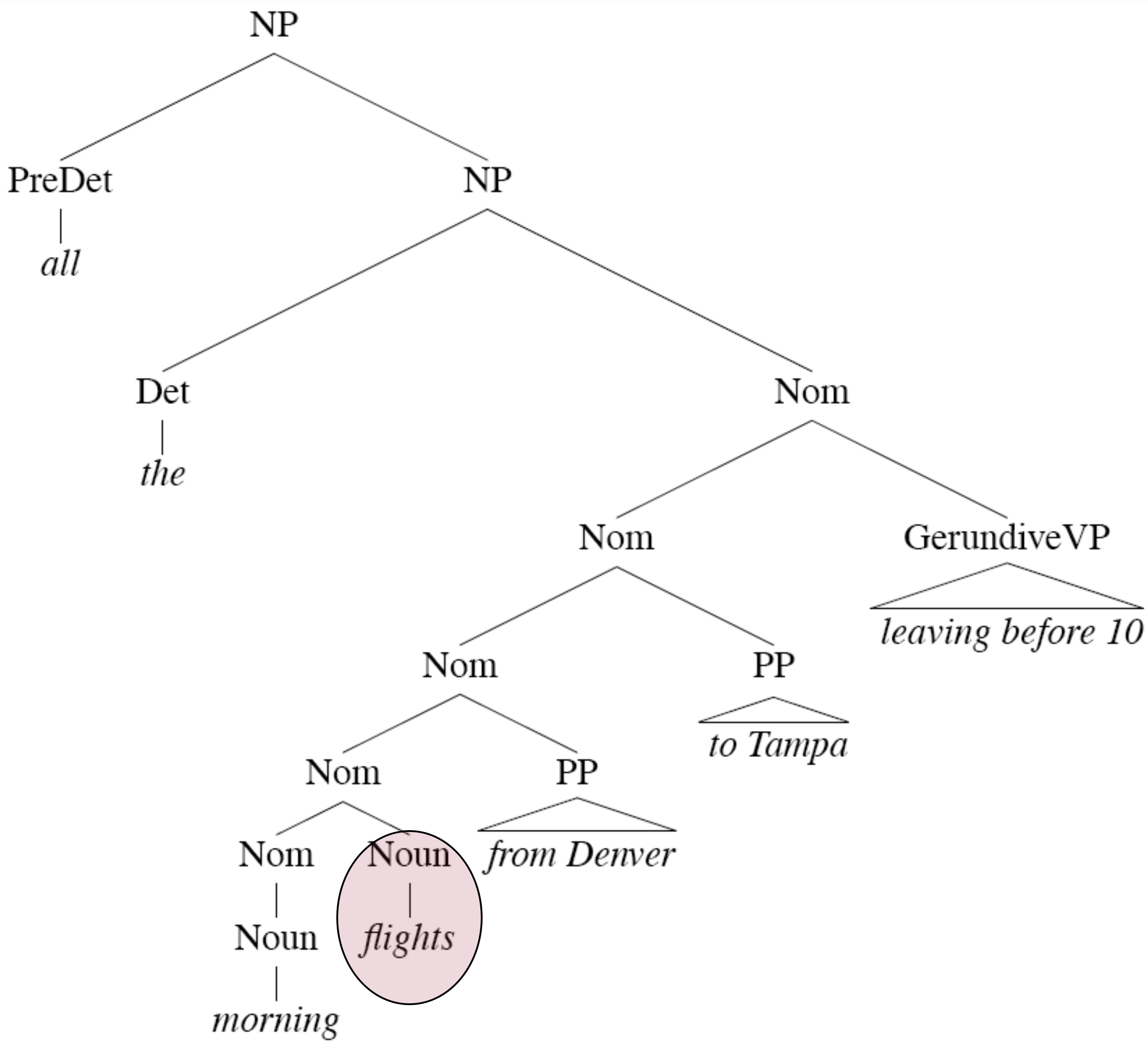
Treebank Grammars

- 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...

$$
\begin{aligned}
\mathtt{VP} &\rightarrow \mathtt{VBD\ PP} \\
\mathtt{VP} &\rightarrow \mathtt{VBD\ PP\ PP} \\
\mathtt{VP} &\rightarrow \mathtt{VBD\ PP\ PP\ PP} \\
\mathtt{VP} &\rightarrow \mathtt{VBD\ PP\ PP\ PP\ PP}
\end{aligned}
$$

- Instead of *VP → VBD PP2*
  PP2 → PP2 PP

Head


NP
PreDet
all
NP
Det
the
Nom
Nom
GerundiveVP
leaving before 10
Nom
PP
to Tampa
Nom
PP
from Denver
Nom
Noun
Noun
flights
morning

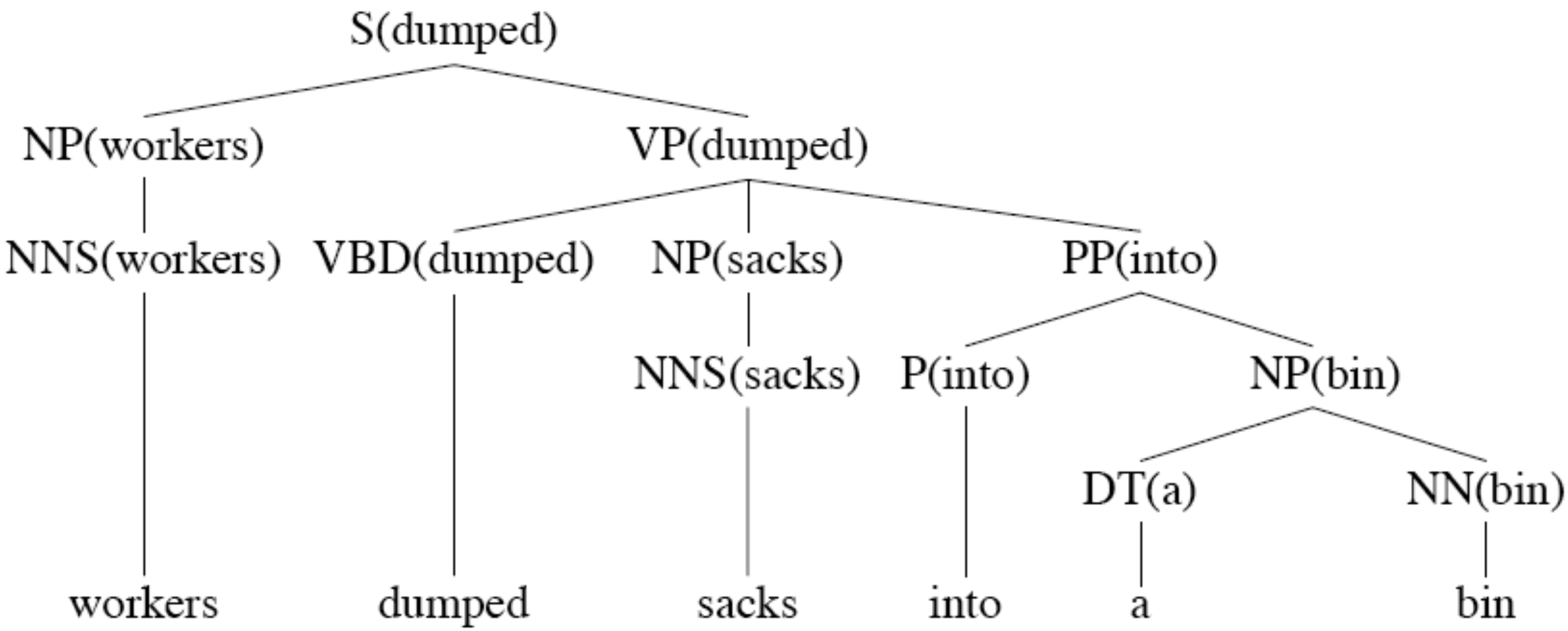
Head

- 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.
- Not only for NP…

Heads in Trees

- Finding heads in treebank trees is a task that arises frequently in many applications.
  - Particularly important in statistical parsing
  - More on that later…
- We can visualize this task by annotating the nodes of a parse tree with the heads of each corresponding node.

Lexically Decorated Tree



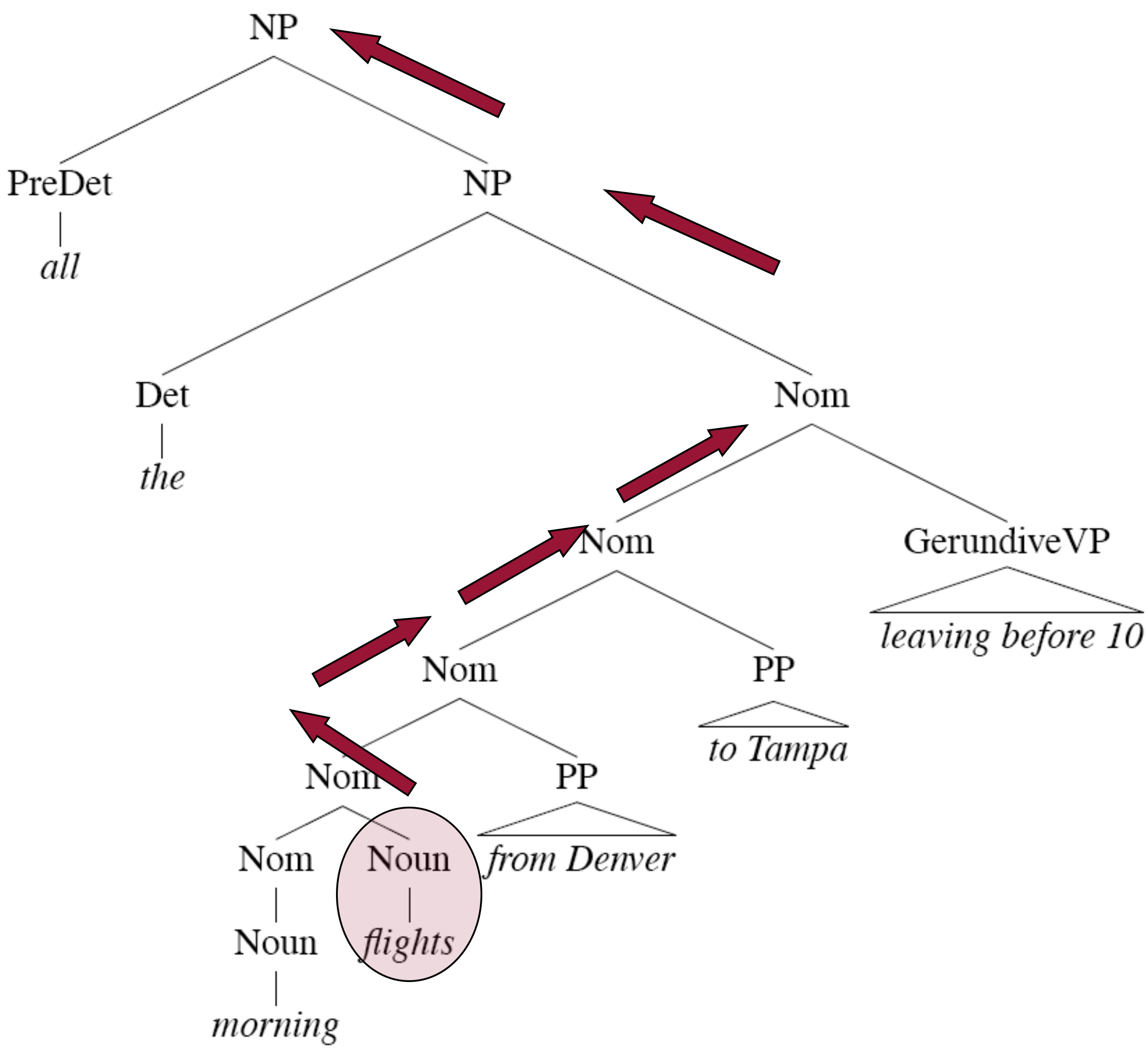
Head Finding

- The standard way to do head finding is to use a simple set of tree traversal rules specific to each non-terminal in the grammar.
  - such traversal rules are “encoded” into *lexicalized grammar rules* (more on that later…)
- Derived from lexicalized treebanks

VP (dumped) → *VDB* (dumped) *NP* (sacks) *PP* (into)

Means that in *VP* → *VDB NP PP*, the head of *VDB* goes to *VP*

Noun Phrases



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.