Parsing with Context Free Grammars

Syntactic Parsing

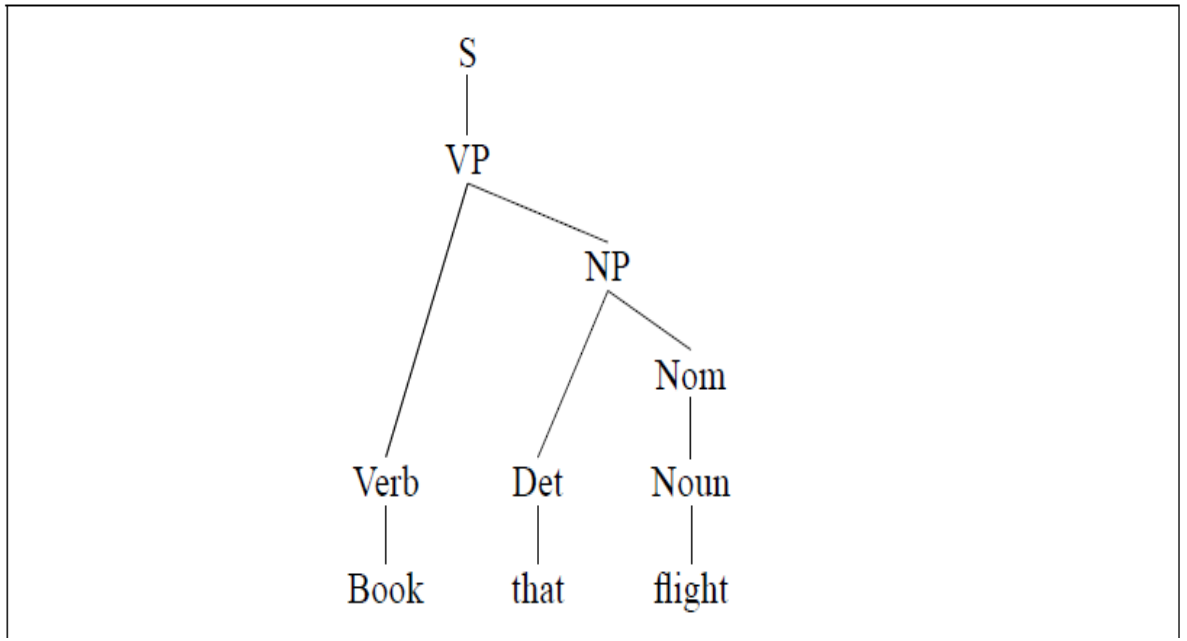
- Syntactic parsing is the task of recognizing a sentence and assigning a syntactic structure to it
- This lecture focuses on the kind of structures assigned by the context-free grammars
- In addition, parsing is an important intermediate stage of representation for **semantic analysis**, and thus plays an important role in applications like **machine translation**, **question answering**, and **information extraction**
- In syntactic parsing,the parser can be viewed as searching through the space of all possible parse trees to find the correct parse tree for the sentence

Book that flight

- A miniature English Grammar and Lexicon- Fig 1

$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	
$Nominal \rightarrow Noun\ Nominal$	$Prep \rightarrow from \mid to \mid on$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid TWA$
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	$Nominal \rightarrow Nominal\ PP$

- Correct Parse Tree- Fig 2



cntd

- The goal of a parsing search is to find all trees whose root is the start symbol *S*, which cover exactly the words in the input.
- The final parse tree, it must have one root, which must be the start symbol *S* and there must be three leaves, and they must be the words *book*, *that*, and *flight*
- There are clearly two kinds of constraints that should help guide the search
- One kind of constraint comes from the data, i.e. the input sentence itself
- The second kind of constraint comes from the grammar
- These two constraints give rise to the two search strategies underlying most parsers: **top-down** or **goal-directed search** and **bottom-up** or **data-directed search**.

Top Down Parsing

- A **top-down** parser searches for a parse tree by trying to build from the root node *S* down to the leaves.
- The algorithm starts by assuming the input can be derived by the designated start symbol *S*.
- The next step is to find the tops of all trees which can start with *S*, by looking for all the grammar rules with *S* on the left-hand side. In the grammar in Figure A, there are three rules that expand *S*, so the second **ply**, or level, of the search space in Figure 3 has three partial trees

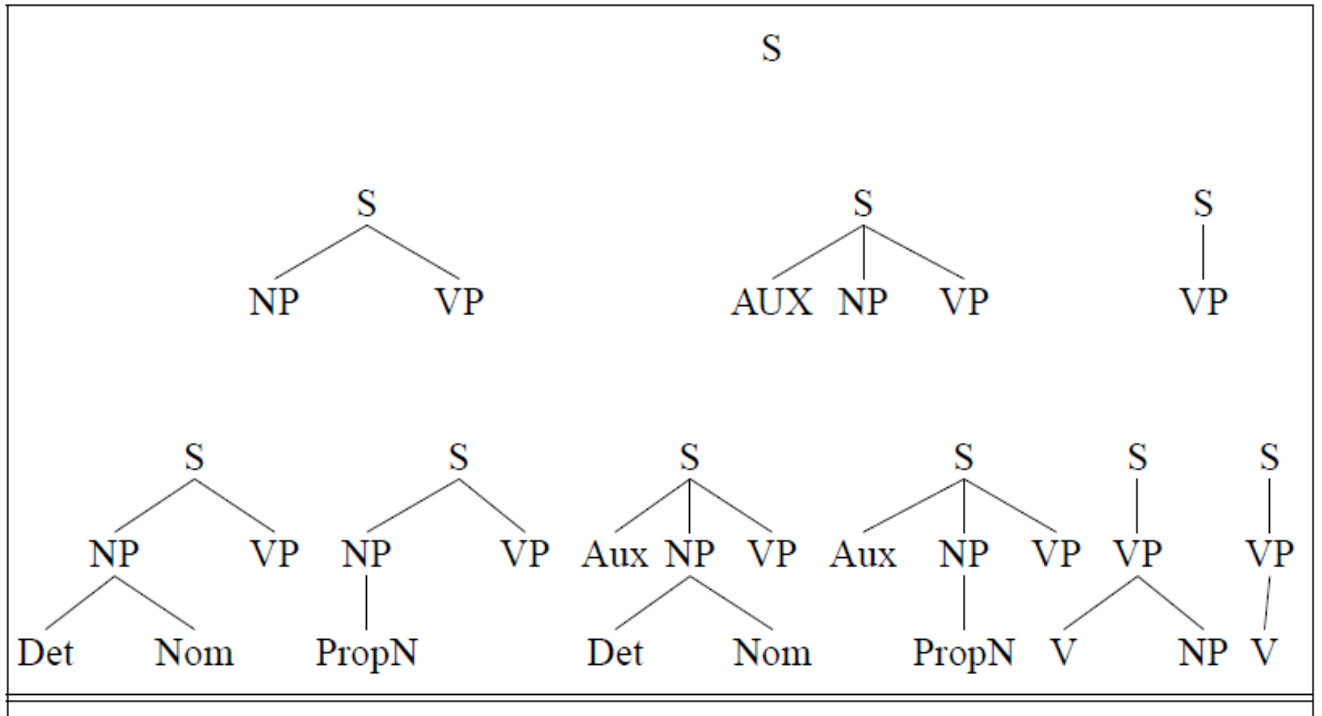


Figure 3

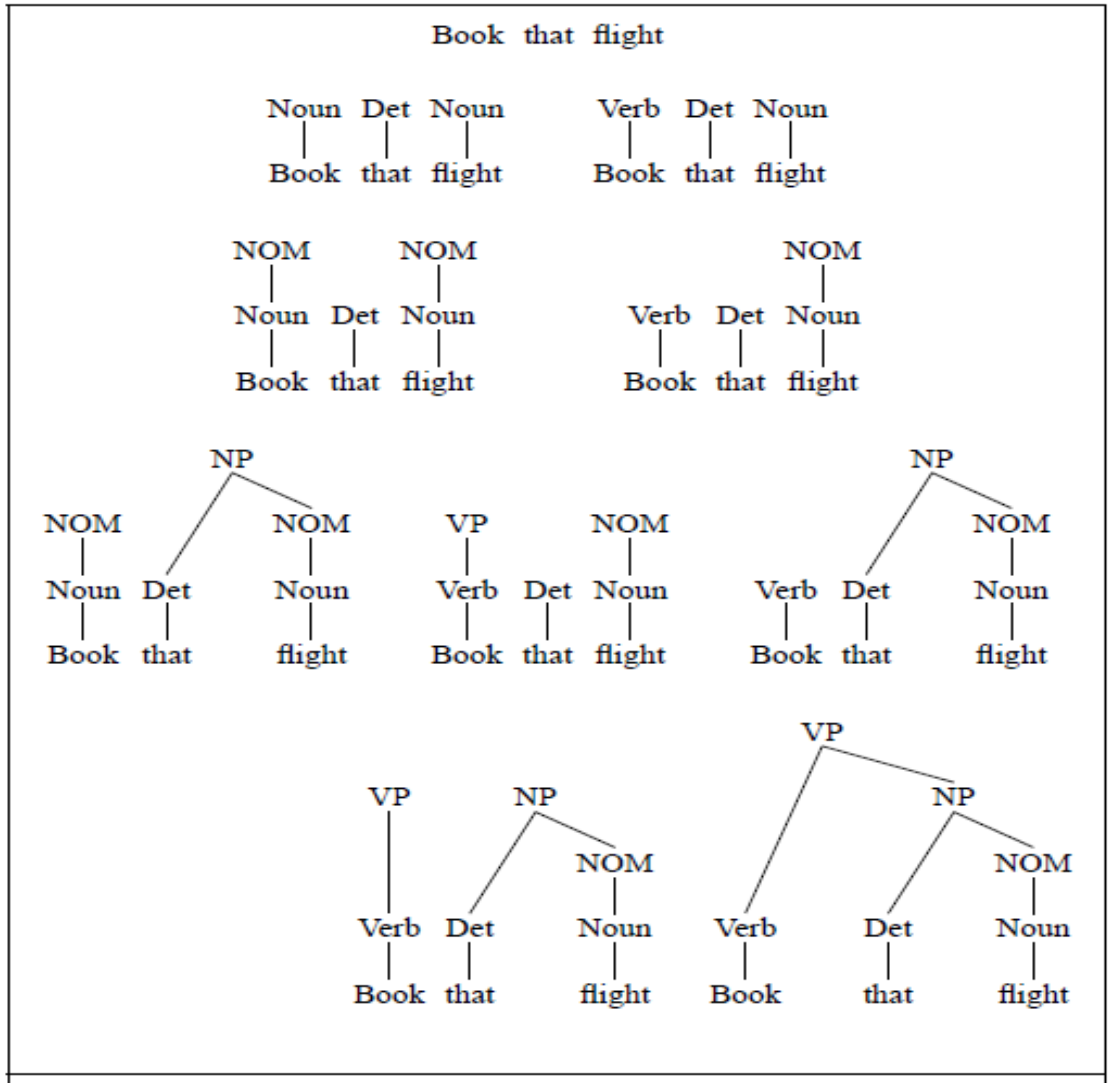
cntd

- We next expand the constituents in these three new trees, just as we originally expanded *S*.
- The first tree tells us to expect an *NP* followed by a *VP*, the second expects an *Aux* followed by an *NP* and a *VP*, and the third a *VP* by itself.
- Trees are grown downward until they eventually reach the part-of-speech categories at the bottom of the tree
- At this point, trees whose leaves fail to match all the words in the input can be rejected, leaving behind those trees that represent successful parses.
- In Figure 3, only the 5th parse tree (the one which has expanded the rule *VP* ->*Verb NP*) will eventually match the input sentence *Book that flight*

Bottom-Up Parsing

- In bottom-up parsing, the parser starts with the words of the input, and tries to build trees from the words up, again by applying rules from the grammar one at a time
- The parse is successful if the parser succeeds in building a tree rooted in the start symbol *S* that covers all of the input
- Figure 4 show the bottom-up search space, beginning with the sentence *Book that flight*
- The parser begins by looking up each word (*book*, *that*, and *flight*) in the lexicon and building three partial trees with the part of speech for each word
- But the word *book* is ambiguous; it can be a noun or a verb.
- Thus the parser must consider two possible sets of trees.
- The first two plies in Figure 10.4 show this initial bifurcation of the search space.

Bottom up Search Space for the sentence *Book that flight*



cntd

- Each of the trees in the second ply are then expanded.
- In the parse on the left (the one in which *book* is incorrectly considered a noun), the

  Nominal->Noun rule is applied to both of the *Noun*s (*book* and *flight*).
- This same rule is also applied to the sole *Noun* (*flight*) on the right, producing the trees on the third ply.
- In general, the parser extends one ply to the next by looking for places in the parse-in-progress where the right-hand-side of some rule might fit.
- This contrasts with the earlier top-down parser, which expanded trees by applying rules when their left-hand side matched an unexpanded nonterminal.
- Thus in the fourth ply, in the first and third parse, the sequence *Det Nominal* is recognized as the right-hand side of the *NP->Det Nominal* rule.

cntd

- In the fifth ply, the interpretation of *book* as a noun has been pruned from the search space.
- This is because this parse cannot be continued: there is no rule in the grammar with the right-hand side *Nominal NP*.
- The final ply of the search space is the correct parse tree
- Make sure you understand which of the two parses on the penultimate ply gave rise to this parse.

Features and unification

- We introduce the idea that grammatical categories like *VPto*, *Sthat*, *Non3sgAux*, or *3sgNP*, as well as the grammatical rules like *S->NP VP* that make use of them, should be thought of as *objects* that can have complex sets of *properties* associated with them
- The information in these properties is represented by **constraints**, and so these kinds of models are often called **constraint-based formalisms**
- Noun phrases like *this flight* and *those flights* can be distinguished based on whether they are singular or plural.
- This distinction can be captured if we associate a property called NUMBER that can have the value singular or plural, with appropriate members of the *NP* category.
- Given this ability, we can say that *this flight* is a member of the *NP* category and, in addition, has the value singular for its NUMBER property.
- This same property can be used in the same way to distinguish singular and plural members of the *VP* category such as *serves lunch* and *serve lunch*.

Feature Structures

- Feature structures are sets of feature-value pairs, where features are unanalyzable atomic symbols drawn from some finite set, and values are either atomic symbols or feature structures.
- Such feature structures are traditionally illustrated with the following kind of matrix-like diagram

$$\begin{bmatrix} \text{FEATURE}_1 & \text{VALUE}_1 \\ \text{FEATURE}_2 & \text{VALUE}_2 \\ \vdots & \\ \text{FEATURE}_n & \text{VALUE}_n \end{bmatrix}$$

cntd

- To be concrete, let us consider the number property discussed above.
- To capture the number property, we will use the symbol NUMBER to designate this grammatical attribute, and the symbols SG and PL to designate the possible values it can take on in English.
- A simple feature structure consisting of this single feature would then be illustrated as follows.

$$\begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix}$$

- Adding an additional feature-value pair to capture the grammatical notion of person leads to the following feature structure

$$\begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix}$$

cntd

- Next we can encode the grammatical category of the constituent that this structure corresponds to through the use of the CAT feature
- For example, we can indicate that these features are associated with a noun phrase by using the following structure

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix}$$

- This structure can be used to represent the *3sgNP* category to capture a restricted subcategory of noun phrases
- The corresponding plural version of this structure would be captured as follows

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUMBER} & \text{PL} \\ \text{PERSON} & 3 \end{bmatrix}$$

Cntd

- Features can also have other feature structures as their values
- This is to bundle a set of feature-value pairs together for similar treatment
- As an example the NUMBER and PERSON features are often lumped together since grammatical subjects must agree with their predicates in both their number and person properties
- This lumping together can be captured by introducing an AGREEMENT feature that takes a feature structure consisting of the NUMBER and PERSON feature-value pairs as its value
- Introducing this feature into our third person singular noun phrase yields the following kind of structure

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix} \end{bmatrix}$$

UNIFICATION OF FEATURE STRUCTURES

The two principal operations on feature structures are

- Merging the information content of two structures
- Rejecting the merger of structures that are incompatible
- A single computational technique, called **unification**, suffices for both of these purposes

Simple Application of the unification operator

$$\begin{bmatrix}\text{NUMBER} & \text{SG}\end{bmatrix} \sqcup \begin{bmatrix}\text{NUMBER} & \text{SG}\end{bmatrix} = \begin{bmatrix}\text{NUMBER} & \text{SG}\end{bmatrix}$$

$$\begin{bmatrix}\text{NUMBER} & \text{SG}\end{bmatrix} \sqcup \begin{bmatrix}\text{NUMBER} & \text{PL}\end{bmatrix} \textit{ Fails!}$$

$$\begin{bmatrix}\text{NUMBER} & \text{SG}\end{bmatrix} \sqcup \begin{bmatrix}\text{NUMBER} & [\,]\end{bmatrix} = \begin{bmatrix}\text{NUMBER} & \text{SG}\end{bmatrix}$$

$$\begin{bmatrix}\text{NUMBER} & \text{SG}\end{bmatrix} \sqcup \begin{bmatrix}\text{PERSON} & 3\end{bmatrix} = \begin{bmatrix}\text{NUMBER} & \text{SG} \\ \text{PERSON} & 3\end{bmatrix}$$

An equality check complicated by the presence of a reentrant structure in the first argument

$$\begin{bmatrix} \text{AGREEMENT} & \boxed{1}\begin{bmatrix}\text{NUMBER} & \text{SG}\\ \text{PERSON} & 3\end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix}\text{AGREEMENT} & \boxed{1}\end{bmatrix}\end{bmatrix}$$

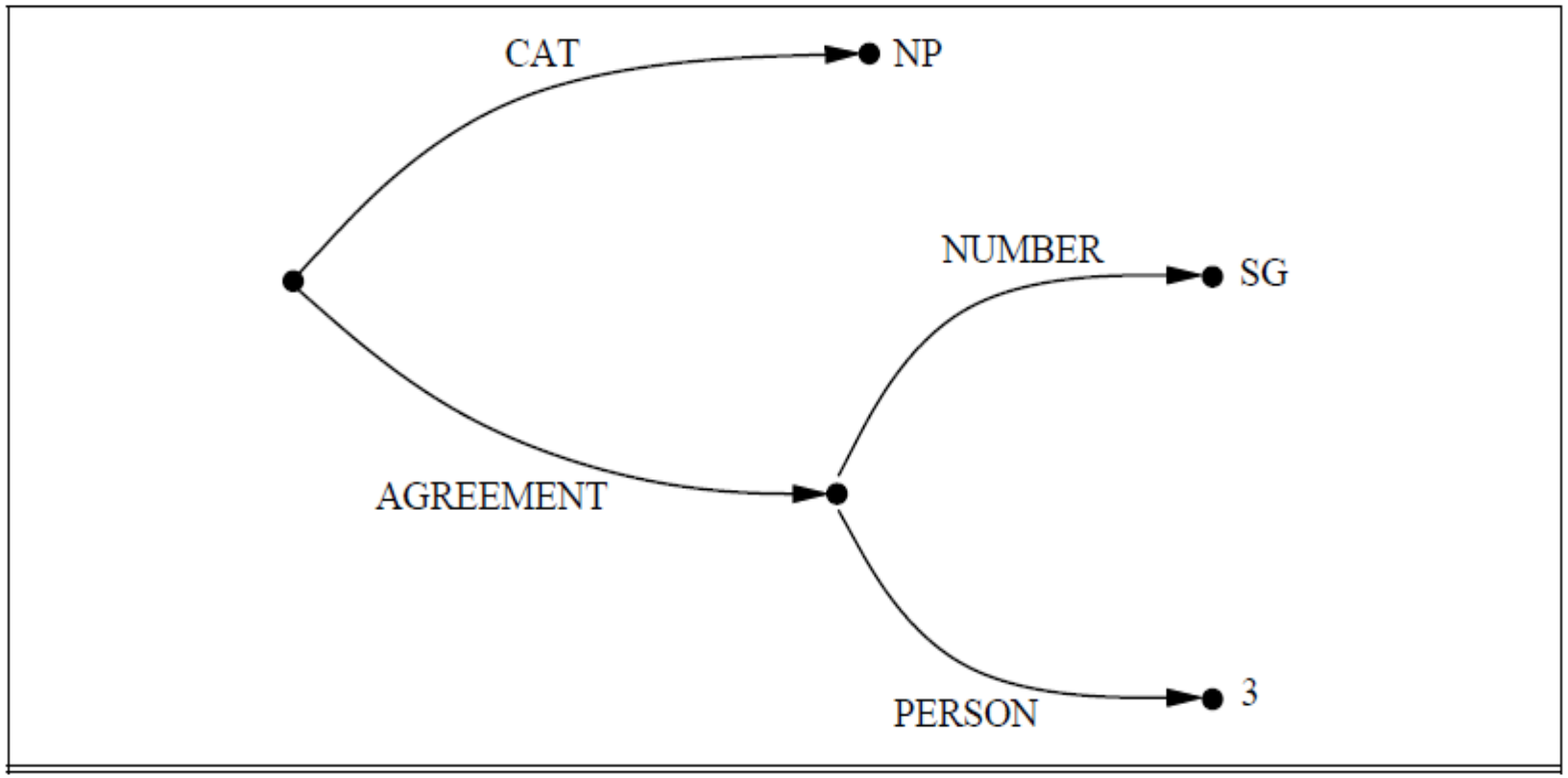
$$\sqcup \begin{bmatrix}\text{SUBJECT} & \begin{bmatrix}\text{AGREEMENT} & \begin{bmatrix}\text{PERSON} & 3\\ \text{NUMBER} & \text{SG}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

$$= \begin{bmatrix} \text{AGREEMENT} & \boxed{1}\begin{bmatrix}\text{NUMBER} & \text{SG}\\ \text{PERSON} & 3\end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix}\text{AGREEMENT} & \boxed{1}\end{bmatrix}\end{bmatrix}$$

Feature Path

- Feature path – a sequence of features through a feature structure leading to a particular value
- Eg: In previous slide <AGREEMENT NUMBER> leads to value sg
- Leads to an alternative representation of features using directed graphs
- Features appear as labeled edges and values as nodes

A directed graph notation for feature structure



Feature Structures in the Grammar

- Our next step, is to specify a way to integrate feature structures and unification operations into the specification of a grammar.
- This can be accomplished by *augmenting* the rules of ordinary context-free grammars with attachments that specify feature structures for the constituents of the rules, along with appropriate unification operations that express constraints on those constituents.
- From a grammatical point of view, these attachments will be used to accomplish the following goals:

- To associate complex feature structures with both lexical items and instances of grammatical categories.
- To guide the composition of feature structures for larger grammatical constituents based on the feature structures of their component parts.
- To enforce compatibility constraints between specified parts of grammatical constructions.

Notation to denote grammar augmentations

$$\beta_0 \rightarrow \beta_1 \cdots \beta_n$$
$$\{set\ of\ constraints\}$$

The specified constraints have one of the following forms.

$$\langle \beta_i\ feature\ path \rangle = Atomic\ value$$
$$\langle \beta_i\ feature\ path \rangle = \langle \beta_j\ feature\ path \rangle$$

- The notation <β*i feature path*>denotes a feature path through the feature structure associated with the β*i* component of the context-free part of the rule.
- The first style of constraint specifies that the value found at the end of the given path must unify with the specified atomic value.
- The second form specifies that the values found at the end of the two given paths must be unifiable.

cntd

$S \rightarrow NP\ VP$

Only if the number of the NP is equal to the number of the VP.

Using the new notation, this rule can now be expressed as follows.

$S \rightarrow NP\ VP$

$\langle NP\ \text{NUMBER} \rangle = \langle VP\ \text{NUMBER} \rangle$

Probabilistic Parsing

- Due to the prevalence of ambiguity, probabilistic parsers can play an important role in most parsing or natural language understanding task
- Another important use of probabilistic grammars is in **language modeling** for speech recognition or augmentative communication.
- The simplest augmentation of the context-free grammar is the **Probabilistic Context-Free Grammar** (**PCFG**), also known as the PCFG or Stochastic Context- SCFG Free Grammar (**SCFG**)

Recall that a context-free grammar G is defined by four parameters (N, Σ, P, S):

1. a set of nonterminal symbols (or 'variables') N
2. a set of terminal symbols Σ (disjoint from N)
3. a set of productions P, each of the form $A \rightarrow \beta$, where A is a non-terminal and β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$.
4. a designated start symbol S

A PCFG, a probabilistic augmentation of the miniature English Grammar and Lexicon

$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.05] | the [.80] | a [.15]	
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$	[.10]
$S \rightarrow VP$	[.05]	$Noun \rightarrow flights$	[.50]
$NP \rightarrow Det\ Nom$	[.20]	$Noun \rightarrow meal$	[.40]
$NP \rightarrow Proper\text{-}Noun$	[.35]	$Verb \rightarrow book$	[.30]
$NP \rightarrow Nom$	[.05]	$Verb \rightarrow include$	[.30]
$NP \rightarrow Pronoun$	[.40]	$Verb \rightarrow want$	[.40]
$Nom \rightarrow Noun$	[.75]	$Aux \rightarrow can$	[.40]
$Nom \rightarrow Noun\ Nom$	[.20]	$Aux \rightarrow does$	[.30]
$Nom \rightarrow Proper\text{-}Noun\ Nom$	[.05]	$Aux \rightarrow do$	[.30]
$VP \rightarrow Verb$	[.55]	$Proper\text{-}Noun \rightarrow TWA$	[.40]
$VP \rightarrow Verb\ NP$	[.40]	$Proper\text{-}Noun \rightarrow Denver$	[.40]
$VP \rightarrow Verb\ NP\ NP$	[.05]	$Pronoun \rightarrow you$ [.40] | I [.60]	

cntd

- A probabilistic context-free grammar augments each rule in *P* with a conditional probability:

$$A \rightarrow \beta [p]$$

- A PCFG is thus a 5-tuple $G = (N, \Sigma, P, S, D)$ where *D* is a function assigning probabilities to each rule in *P*.
- This function expresses the probability *p* that the given nonterminal *A* will be expanded to the sequence
- It is often referred to as

$$P(A \rightarrow \beta)$$

or as

$$P(A \rightarrow \beta | A)$$

cntd

- Formally this is conditional probability of a given expansion given the left-hand-side nonterminal *A*
- A PCFG can be used to estimate a number of useful probabilities concerning a sentence and its parse-tree(s).
- For example a PCFG assigns a probability to each parse-tree *T* (i.e. each derivation) of a sentence *S*.
- This attribute is useful in **disambiguation**.
- For example, consider the two parses of the sentence "Can you book TWA flights" (one meaning 'Can you book flights on behalf of TWA', and the other meaning 'Can you book flights run by TWA') shown in Figure
- The probability of a particular parse *T* is defined as the product of the probabilities of all the rules *r* used to expand each node *n* in the parse tree:

$$P(T,S) = \prod_{n \in T} p(r(n))$$

cntd

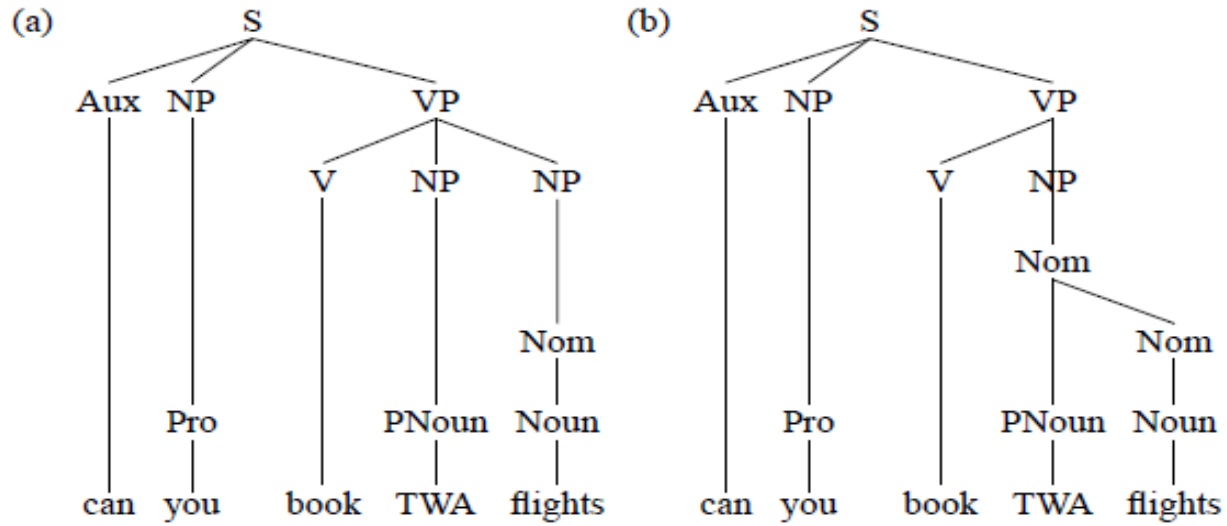
- The resulting probability P(T,S) is both the joint probability of the
- parse and the sentence, and also the probability of the parse P(T).
- First, by the definition of joint probability

$$P(T,S) = P(T)P(S|T)$$

- But since a parse tree includes all the words of the sentence, P(S|T) is 1.
- 1. Thus

$$P(T,S) = P(T)P(S|T) = P(T)$$

- The probability of each of the trees in the following figure can be computed by multiplying together each of the rules used in the derivation.
- For example, the probability of the left tree (call it *Tl*) and the right tree (12.2b or *Tr*) can be computed as follows



Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP NP	.05
NP	→	Nom	.05
NP	→	PNoun	.35
Nom	→	Noun	.75
Aux	→	Can	.40
NP	→	Pro	.40
Pro	→	you	.40
Verb	→	book	.30
PNoun	→	TWA	.40
Noun	→	flights	.50

Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP	.40
NP	→	Nom	.05
Nom	→	PNoun Nom	.05
Nom	→	Noun	.75
Aux	→	Can	.40
NP	→	Pro	.40
Pro	→	you	.40
Verb	→	book	.30
Pnoun	→	TWA	.40
Noun	→	flights	.50

Figure 12.2 Two parse trees for an ambiguous sentence. Parse (a) corresponds to the meaning 'Can you book flights on behalf of TWA?', parse (b) to 'Can you book flights which are run by TWA'.

cntd

$$
\begin{aligned}
P(T_l) &= .15 * .40 * .05 * .05 * .35 * .75 * .40 * .40 * .40 \\
&\quad * .30 * .40 * .50 \\
&= 1.5 \times 10^{-6}
\end{aligned}
$$

$$
\begin{aligned}
P(T_r) &= .15 * .40 * .40 * .05 * .05 * .75 * .40 * .40 * .40 \\
&\quad * .30 * .40 * .50 \\
&= 1.7 \times 10^{-6}
\end{aligned}
$$

cntd

- Thus this parse would correctly be chosen by a disambiguation algorithm which selects the parse with the highest PCFG probability.
- The disambiguation algorithm picks the best tree for a sentence *S* out of the set of parse trees for *S*
- We want the parse tree *T* which is most likely given the sentence *S*.

$$\hat{T}(S) = \underset{T \in \tau(S)}{\operatorname{argmax}} P(T|S)$$

- Furthermore, since we showed above that P(T,S)=P(T) the final equation for choosing the most likely parse simplifies to choosing the parse with the highest probability:

$$\hat{T}(S) = \underset{T \in \tau(S)}{\operatorname{argmax}} P(T)$$

Lexicalized PCFG

- Lexicalization of a tree bank
- Lexicalized Context free grammars
- Parameterization in lexicalized context free grammars

Heads in Context Free Rules

Add annotations specifying the "head" of each rule:

S	⇒	NP	VP
VP	⇒	Vi	
VP	⇒	Vt	NP
VP	⇒	VP	PP
NP	⇒	DT	NN
NP	⇒	NP	PP
PP	⇒	IN	NP

Vi	⇒	sleeps
Vt	⇒	saw
NN	⇒	man
NN	⇒	woman
NN	⇒	telescope
DT	⇒	the
IN	⇒	with
IN	⇒	in

More about Heads

- Each context-free rule has one "special" child that is the head of the rule. e.g.,

S	$\Rightarrow$	NP	VP		(VP is the head)
VP	$\Rightarrow$	Vt	NP		(Vt is the head)
NP	$\Rightarrow$	DT	NN	NN	(NN is the head)

- A core idea in syntax
  (e.g., see X-bar Theory, Head-Driven Phrase Structure Grammar)
- Some intuitions:
  - The central sub-constituent of each rule.
  - The semantic predicate in each rule.

Rules which recover heads: An example for NPs

If the rule contains NN, NNS, or NNP:
Choose the rightmost NN, NNS, or NNP

Else If the rule contains an NP: Choose the leftmost NP

Else If the rule contains a JJ: Choose the rightmost JJ

Else If the rule contains a CD: Choose the rightmost CD

Else Choose the rightmost child

e.g.,

NP	⇒	DT	NNP	NN
NP	⇒	DT	NN	NNP
NP	⇒	NP	PP	
NP	⇒	DT	JJ	
NP	⇒	DT		

Rules which recover heads: An example for VPs

If the rule contains Vi or Vt: Choose the leftmost Vi or Vt

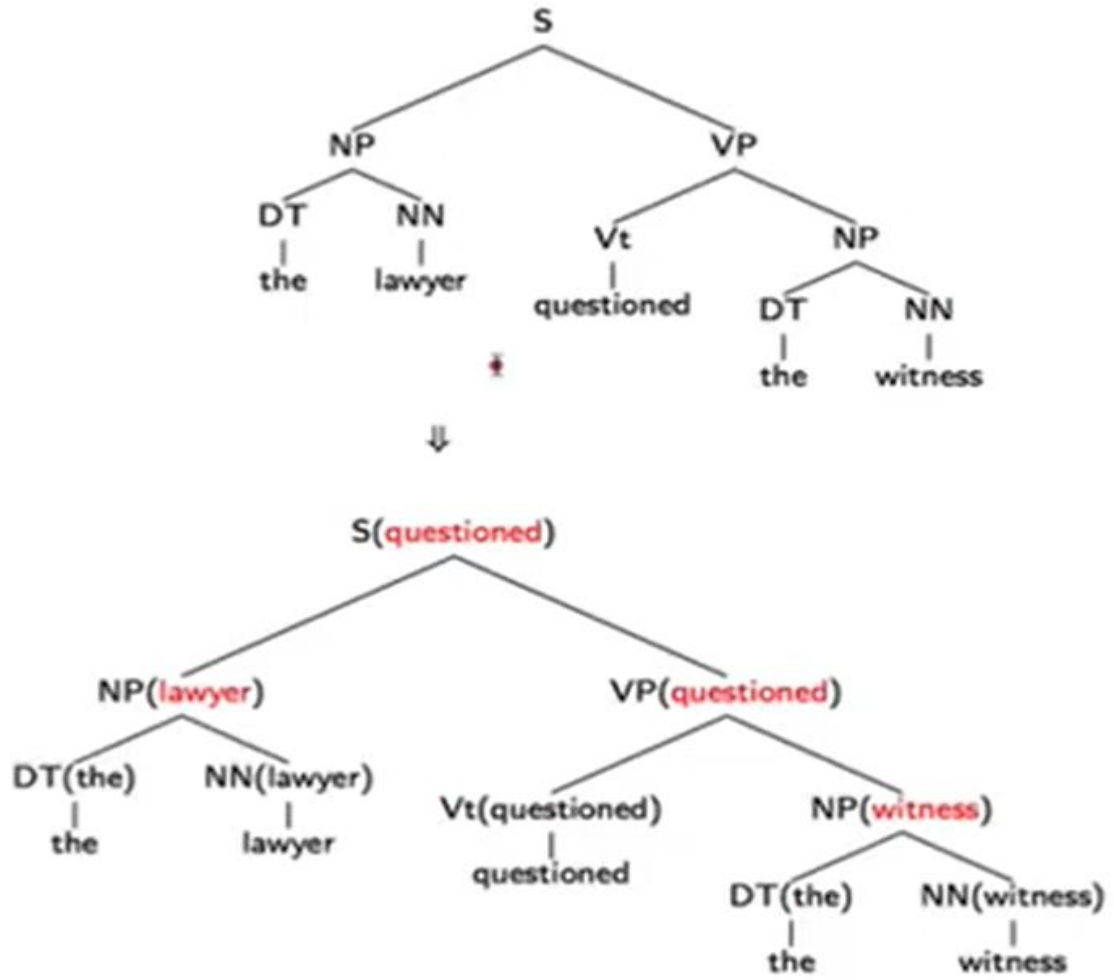
Else If the rule contains an VP: Choose the leftmost VP

Else Choose the leftmost child

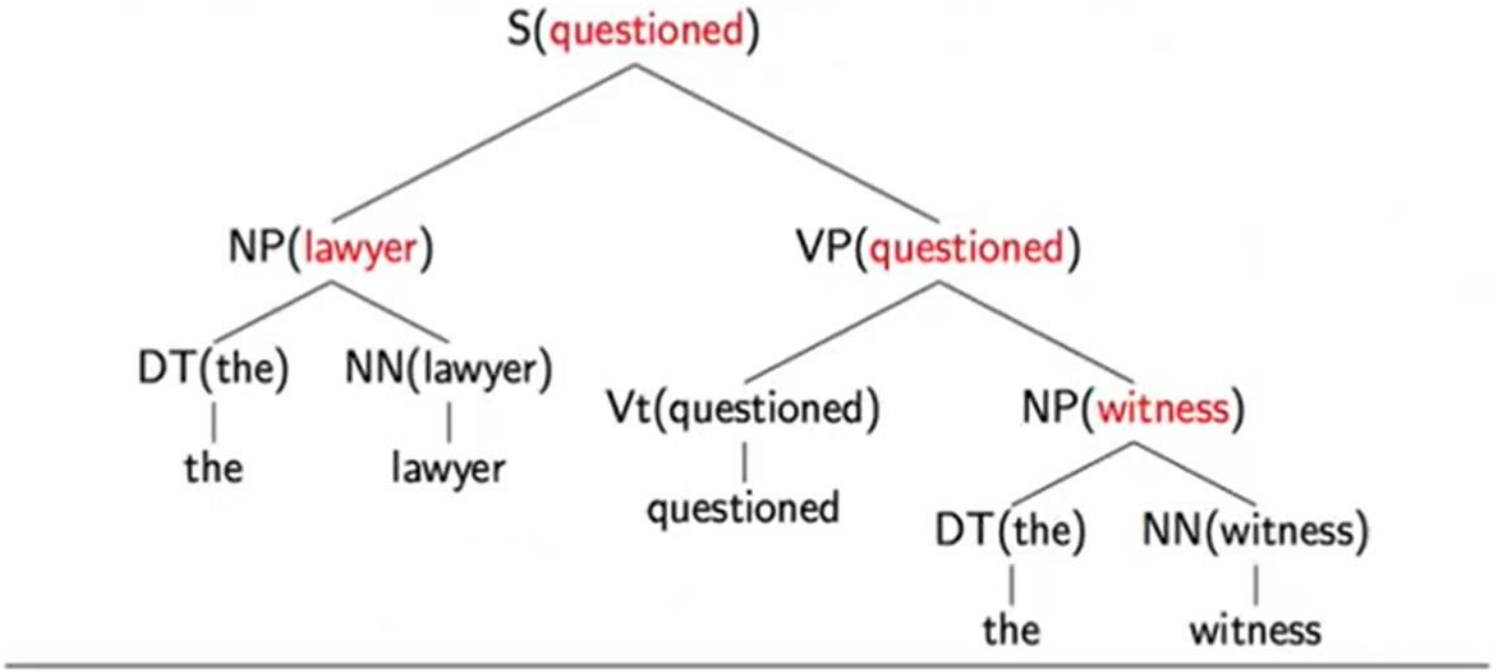
e.g.,

VP ⇒ Vt NP
VP ⇒ VP PP

Adding Headwords to Trees



Adding Headwords to Trees-cntd



- A constituent receives its headword from its **head child**.

S	⇒	NP	VP		(S receives headword from VP)
VP	⇒	Vt	NP		(VP receives headword from Vt)
NP	⇒	DT		NN	(NP receives headword from NN)

Chomsky Normal Form

A context free grammar $G = (N, \Sigma, R, S)$ in Chomsky Normal Form is as follows

- N is a set of non-terminal symbols
- Σ is a set of terminal symbols
- R is a set of rules which take one of two forms:
  - $X \rightarrow Y_1 Y_2$ for $X \in N$, and $Y_1, Y_2 \in N$
  - $X \rightarrow Y$ for $X \in N$, and $Y \in \Sigma$
- $S \in N$ is a distinguished start symbol

We can find the highest scoring parse under a PCFG in this form, in $O(n^3|N|^3)$ time where n is the length of the string being parsed.

Lexicalized Context Free Grammars in Chomsky Normal Form

- N is a set of non-terminal symbols
- Σ is a set of terminal symbols
- R is a set of rules which take one of three forms:
  - $X(h) \rightarrow_1 Y_1(h)\ Y_2(w)$ for $X \in N$, and $Y_1, Y_2 \in N$, and $h, w \in \Sigma$
  - $X(h) \rightarrow_2 Y_1(w)\ Y_2(h)$ for $X \in N$, and $Y_1, Y_2 \in N$, and $h, w \in \Sigma$
  - $X(h) \rightarrow h$ for $X \in N$, and $h \in \Sigma$
- $S \in N$ is a distinguished start symbol

An Example

S(saw)	$\rightarrow_2$	NP(man)	VP(saw)
VP(saw)	$\rightarrow_1$	Vt(saw)	NP(dog)
NP(man)	$\rightarrow_2$	DT(the)	NN(man)
NP(dog)	$\rightarrow_2$	DT(the)	NN(dog)
Vt(saw)	$\rightarrow$	saw	
DT(the)	$\rightarrow$	the	
NN(man)	$\rightarrow$	man	
NN(dog)	$\rightarrow$	dog	

Parameters in Lexicalized PCFG

- An example parameter in a PCFG:

$$q(\text{S} \rightarrow \text{NP VP})$$

- An example parameter in a Lexicalized PCFG:

$$q(\text{S(saw)} \rightarrow_2 \text{NP(man) VP(saw)})$$