Reminders

HW6 IS DUE TONIGHT BY 11:59PM

HW7 HAS BEEN RELEASED. IT IS DUE BEFORE SPRING BREAK

Grammars and Parsing

JURAFSKY AND MARTIN CHAPTERS 12, 13, AND 14

Based on slides from Sameer Singh, Dan Jurafsky, Noah Smith, Slav Petrov, and everyone else they copied from.

Prescriptive v. Descriptive Grammars

Concerned with establishing norms or rules for good writing.

For example:

1. It's important to never split infinitives
2. Prepositions are bad to end sentences with
3. Always have ten items or less

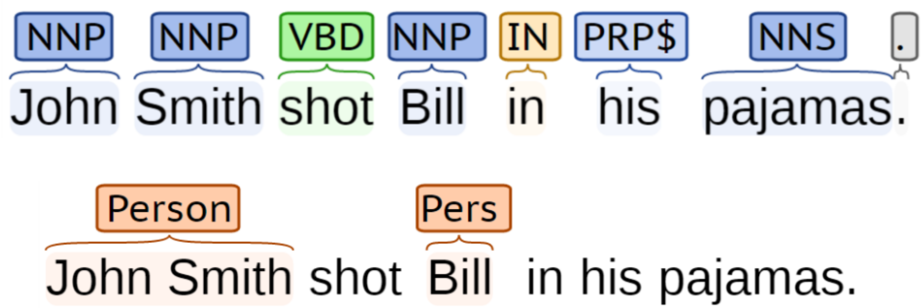
Describe how native speakers actually use a language.

Language is shaped by the speakers of a language and not dictated by central body.

Therefore rules should describe how language is sued.

Limitations of Sequence Tags

John Smith shot Bill in his pajamas.



What happened?
Who shot who?
Who was wearing the pajamas?

How do words group together in English?

A **noun phrase** is a sequence of words surrounding at least one noun.

What evidence do we have that these words group together?

Noun Phrases
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

Constituents

Constituent behave as a unit that can be rearranged:

John talked [to the children] [about drugs].

John talked [about drugs] [to the children].

John talked drugs to the children about

Or substituted/expanded:

John talked [to the children taking the drugs] [about alcohol].

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	X	arrive(s) attract(s) love(s) sit(s)

"Noun phrases appear before verbs in English."

Constituents and Grammars

Grammar

Tells you how the constituents can be arranged
Implicit knowledge for us (we often can't tell *why* something is wrong)
Generate all, and only, the possible sentences of the language
Different from meaning:

Colorless green ideas sleep furiously.

The words are in the right order,
And that ideas are green and colorless,
And that ideas sleep,
And that sleeping is done furiously,
As opposed to: "sleep green furiously ideas colorless"

Uses of Parsing

[send [the text message from James] [to Sharon]]

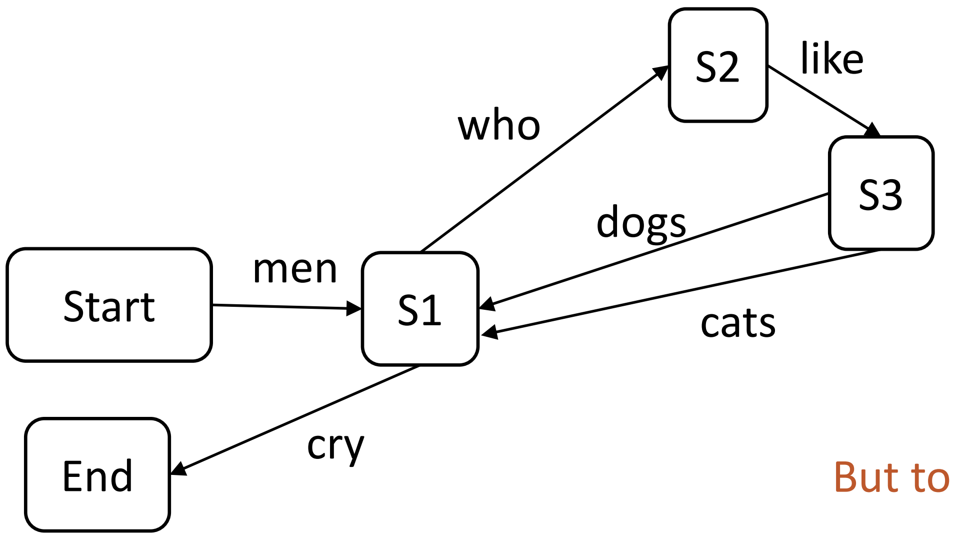
[translate [the message] [from Hindi] [to English]]

1. Grammar checkers
2. Dialog systems
3. High precision question answering
4. Named entity recognition
5. Sentence compression
6. Extracting opinions about products
7. Improved interaction in computer games
8. Helping linguists find data
9. Machine translation
10. Relation extraction systems

Basic Grammar: Regular Expr.

- You can capture individual words:
  - (men|dogs|cats)
- Simple sentences:
  - (men|dogs|cats)(eat|love|consumed)(.|food|lunch)
- Infinite length? Yes!
  - men (who like (cats|dogs))* cry.

Finite State Machine



But too weak for English.

Context Free Grammars

Context Free Grammars

$$G \equiv \langle T, N, R, S \rangle$$

- T — words
- N — constituents
- R — how to compose them
- S — sentence

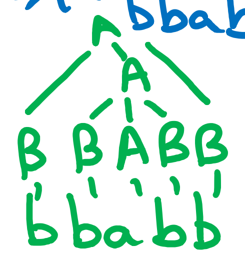
$$A \rightarrow A\,B\,A$$
$$A \rightarrow a$$
$$B \rightarrow b$$

$$G \equiv \langle \{a, b\}, \{A, B\}, R, A \rangle$$

a
bab
bbabb
⋮

$\longleftrightarrow$ $b^* a\, b^*$

$\not\longleftrightarrow$ bbab



Context-Free Grammars

Grammar, G

Terminal Symbols

Non-terminal Symbols

Rules

Grammar applies rules recursively..
If we can construct the input sentence, it is in the grammar, otherwise not.

Context-Free Grammars

Grammar, G

	T	N "Constituents"
	Terminal Symbols	Non-terminal Symbols
	words	
	{man, dog, food... }	S, NP, VP, ...
Rules	A → B C D	A → B A B
	A → w	A → a B → b
		"a"
		"bab" "baba" — x
		→ $b^* a b^*$ ←

Grammar applies rules recursively..

If we can construct the input sentence, it is in the grammar, otherwise not.

Example CFG

Noun	→	*flights* | *breeze* | *trip* | *morning*
Verb	→	*is* | *prefer* | *like* | *need* | *want* | *fly*
Adjective	→	*cheapest* | *non-stop* | *first* | *latest*
		| *other* | *direct*
Pronoun	→	*me* | *I* | *you* | *it*
Proper-Noun	→	*Alaska* | *Baltimore* | *Los Angeles*
		| *Chicago* | *United* | *American*
Determiner	→	*the* | *a* | *an* | *this* | *these* | *that*
Preposition	→	*from* | *to* | *on* | *near*
Conjunction	→	*and* | *or* | *but*

Grammar Rules			**Examples**
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

Example CFG

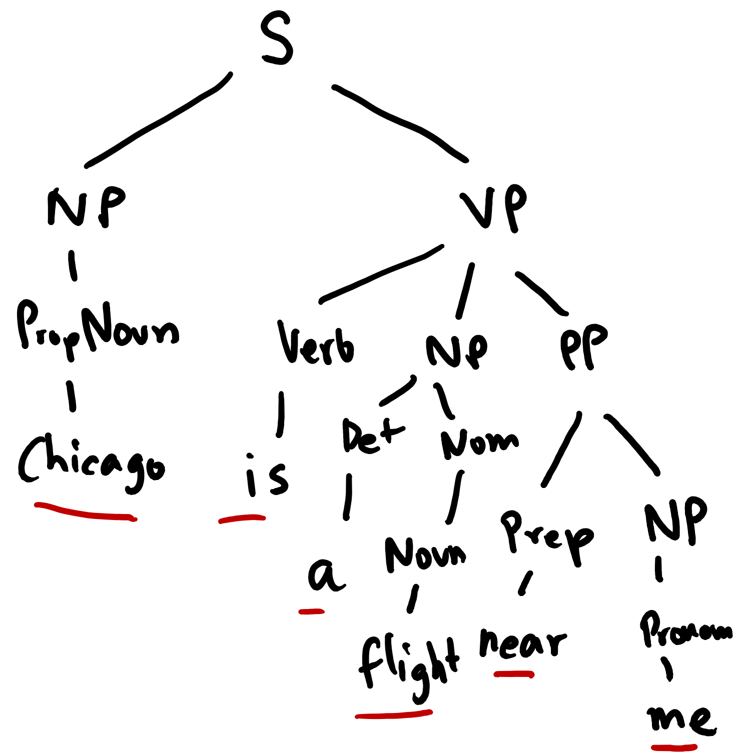
"Lexicon"

$A \to w1 \mid w2$

$A \to w_1 \quad A \to w_2$

Noun →	*flights* | *breeze* | *trip* | *morning*
Verb →	*is* | *prefer* | *like* | *need* | *want* | *fly*
Adjective →	*cheapest* | *non-stop* | *first* | *latest*
	| *other* | *direct*
Pronoun →	*me* | *I* | *you* | *it*
Proper-Noun →	*Alaska* | *Baltimore* | *Los Angeles*
	| *Chicago* | *United* | *American*
Determiner →	*the* | *a* | *an* | *this* | *these* | *that*
Preposition →	*from* | *to* | *on* | *near*
Conjunction →	*and* | *or* | *but*

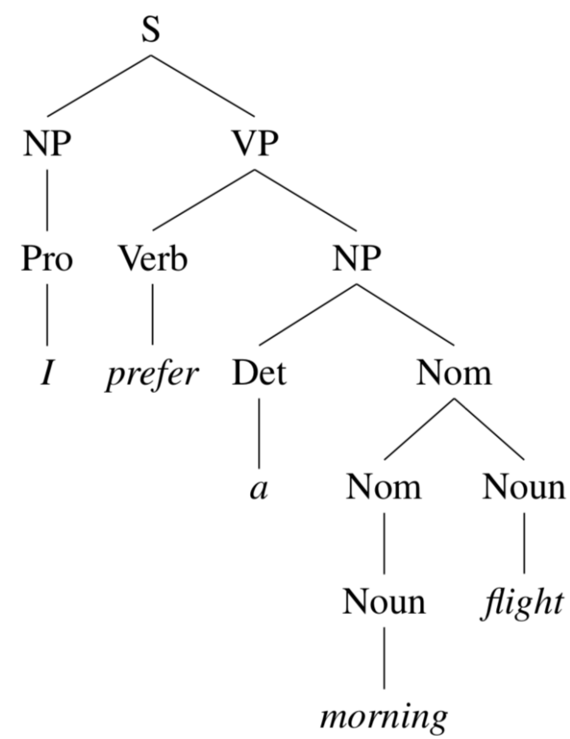
Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles



Example Parse Tree

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

I prefer a morning flight.



Example Parse Tree

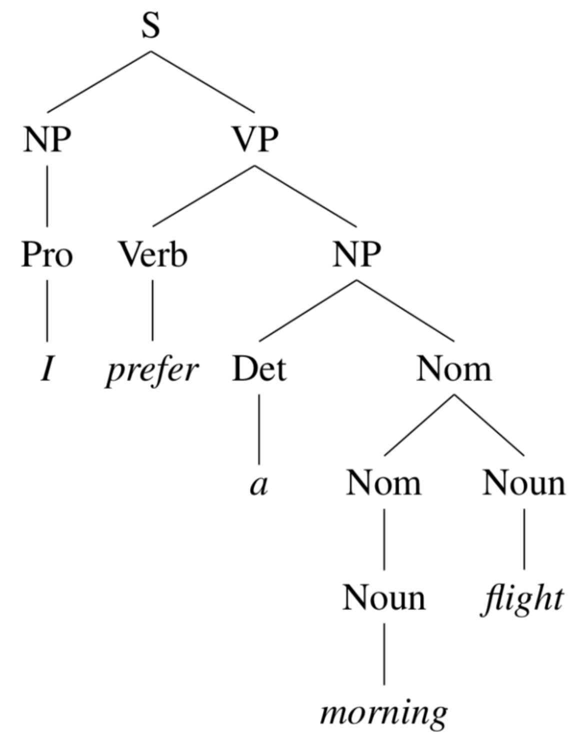
Nom → N_{time} F

F → flight

N_{time} → morning / night / evening

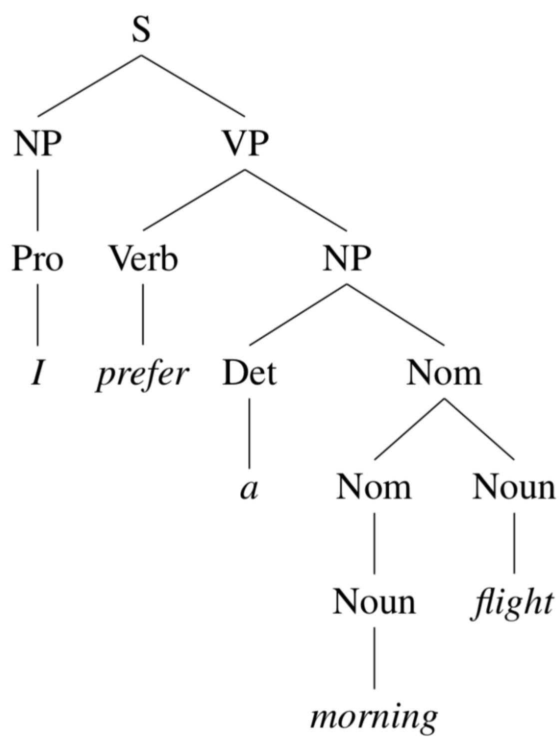
Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

I prefer a morning flight.



Example Parse Tree: Brackets

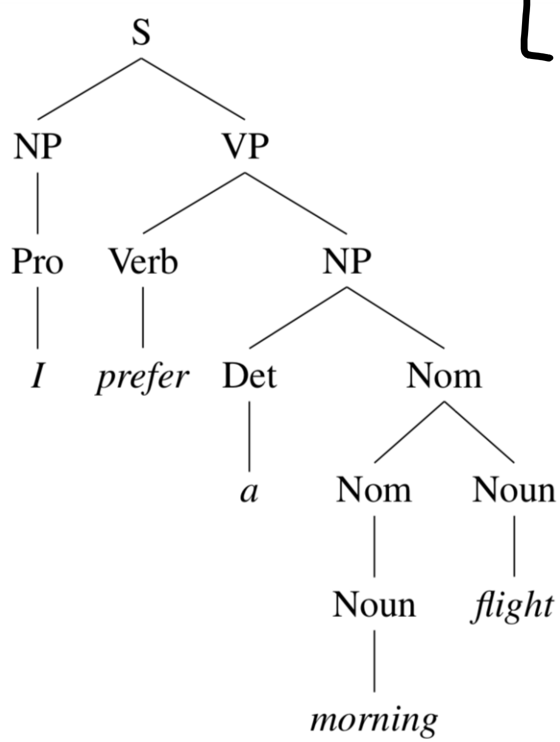
I prefer a morning flight.



Example Parse Tree: Brackets

I prefer a morning flight.

[S [NP [Pro I]] [VP [Verb prefer] [NP [Det ...] ..]]]



Types of Sentences

Declarative	$S \rightarrow NP\ VP$	A plane left.
Imperative	$S \rightarrow VP$	Show the plane.
Yes/no Questions	$S \rightarrow Aux\ NP\ VP$	Did the plane leave?
Wh-Questions	$S \rightarrow WhNP\ Aux\ NP\ VP$	When did the plane leave?

A Computer Model of a Grammar for English Questions

Chris Callison-Burch

June 2000

Contents

Clauses versus phrases

A clause expresses a “complete thought”. One example is a complete sentence, which has a main **verb** and all of its **arguments** filled.

*I **prefer** a morning flight*

Verb: Prefer

Arguments:

1. The subject (the holder of the preference)
2. The direct object (the thing preferred)

The Noun Phrase

Three of the most frequent types of noun phases in English are:

1. Pronouns
2. Proper nouns
3. The construction *NP → Det Nominal*

For the third type, often, we have simple determiner like:

a flight	the flights	this flight
those flights	any flights	some flight

Remember that mass nouns don't require a determiner.

Complex determiners

Sometimes instead of single words like **a, the, those, this, some, any** determiners can be complex phrases themselves.

In English this can have with the possessive marker **'s**.

CCB's laptop
CCB's laptop's broken keyboard
CCB's laptop's keyboard's butterfly keys

This is our first example of recursive rules.

$$NP \rightarrow Det\ Nominal$$

$$Det \rightarrow NP\ \textbf{'s}$$

Nominals

NP → Det Nominal

In English, a *nominal* follows a determiner. This can be as simple as a single noun

Nominal → Noun

But can also be more complex with **pre-** or **post-head noun modifiers**.

Before the noun	After the head noun
Cardinal and ordinal numbers	prepositional phrases
Quantifiers *(many, few, several*)	non-finite clauses
Adjectives (*first-class, earliest*)	relative clauses

Recursive Noun Phrases

this is the house

this is the house that Jack built

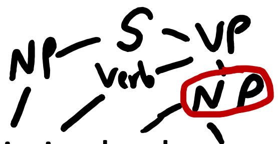
this is the cat that lives in the house that Jack built

this is the dog that chased the cat that lives in the house that Jack built

this is the flea that bit the dog that chased the cat that lives in the house the Jack built

this is the virus that infected the flea that bit the dog that chased the cat that lives in the house that Jack built

Recursive Noun Phrases



this is the house

this is the house that Jack built

this is the cat that lives in the house that Jack built

this is the dog that chased the cat that lives in the house that Jack built

this is the flea that bit the dog that chased the cat that lives in the house the Jack built

this is the virus that infected the flea that bit the dog that chased the cat that lives in the house that Jack built

Recursive Noun Phrases

Simple Noun Phrases

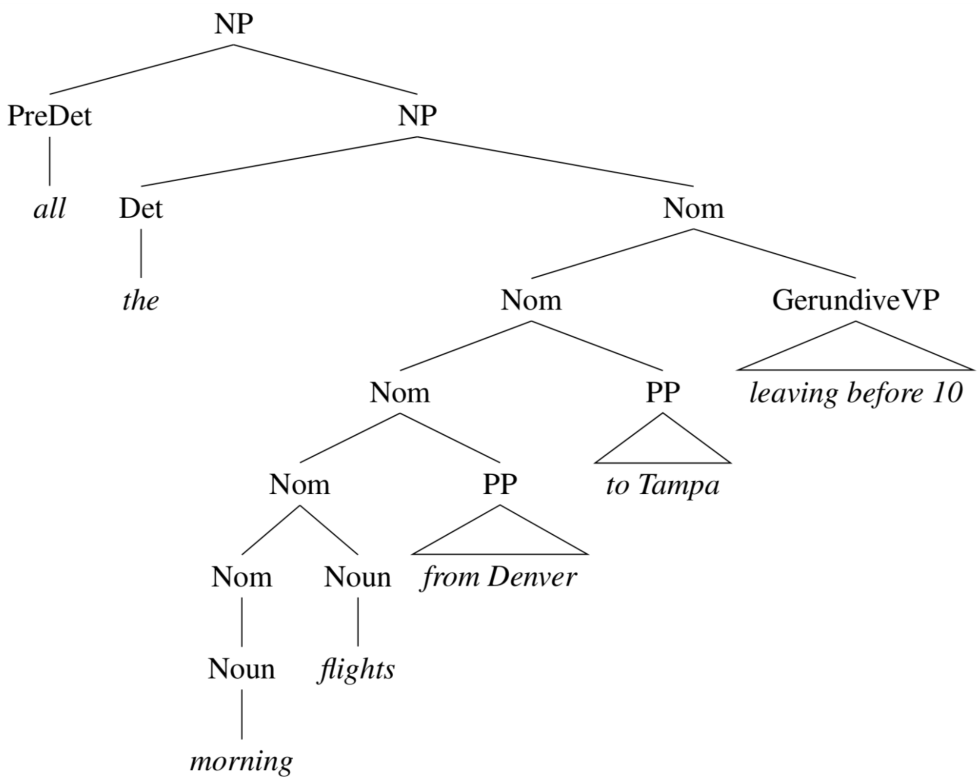
$NP \rightarrow ProperNoun$

$NP \rightarrow Det\ Nominal$

$Nominal \rightarrow Noun \mid Noun\ Nominal$

Complex Noun Phrases

“all the morning flights from Denver to Tampa leaving before 10”



Verb Phrases

Simple Verb Phrases

$VP \rightarrow Verb$	disappear
$VP \rightarrow Verb\ NP$	prefer a morning flight
$VP \rightarrow Verb\ NP\ PP$	leave Boston in the morning
$VP \rightarrow Verb\ PP$	leave in the morning

But all verbs are not the same!
(this grammar over-generates)

Solution: subcategorize!

Sneezed:	Skyler sneezed.
Read:	Gaurav read the book
Find:	Please find a flight to NY.
Give:	Give me a cheaper fare.
Help:	Can you help me with a flight?
Prefer:	I prefer to leave earlier.
Told:	I was told United has a flight.

Subcategorization frames

Frame	Verb	Example
–	eat, sleep	I ate
NP	prefer, find, leave	Find [NP the one ring]
NP NP	show, give	Give [NP Frodo] [NP the ring]
PP_{from} PP_{to}	fly, travel	Fly [PP from Philadelphia] [PP to Honolulu]
NP PP_{with}	help	Help [NP me] [PP with my homework]
$VP_{infinitival}$	prefer, want, need	I would prefer [VP to finish] early
VP_{bare}	can, would, might	I can [VP finish] my homework
S	mean	Does this mean [S we are done]?

Expanding our rule set

We could create subtypes of the class Verb capture the association between verbs and their complements:

Verb-with-NP-complement → *find*| *leave*| *repeat*| ...

Verb-with-S-complement → *think*| *believe*| *say*| ...

Verb-with-Inf-VP-complement → *want*| *try*| *need*| ...

Then modify each VP rule to give the right argument types:

VP → *Verb-with-no-complement*

VP → *Verb-with-NP-comp NP*

VP → *Verb-with-S-comp S*

Predicate-argument relations

Another way of talking about the relation between the verb and these other constituents is to think of the verb as a logical predicate and the constituents as logical arguments of the predicate.

FIND(BILBO, THE RING)
WANTS(SAURON, THE RING)
THROW(FRODO, THE RING, MOUNT DOOM)

Coordination

Conjunctions like **and, or,** and **but** can form larger constructions of the same type. For example, a coordinate noun phrase can consist of two other noun phrases separated by a conjunction:

Please repeat [NP [NP the flights] and [NP the costs]]

I need to know [NP [NP the aircraft] and [NP the flight number]]

Here's a rule that allows these structures:

NP → NP and NP

This also works with other types, like VP, S and Nominal. So some people create "Metarules" like

X → X and X

Source of Grammar?

Manual

Noam Chomsky

Write symbolic grammar (CFG or often richer) and lexicon

S → NP VP	NN → *interest*
NP → (DT) NN	NNS → *rates*
NP → NN NNS	NNS → *raises*
NP → NNP	VBP → *interest*
VP → V NP	VBZ → *rates*

Used grammar/proof systems to prove parses from words

Fed raises interest rates 0.5% in effort to control inflation

- Minimal grammar: 36 parses
- Simple 10 rule grammar: 592 parses
- Real-size broad-coverage grammar: millions of parses

Source of Grammar?

From data!

The Penn Treebank

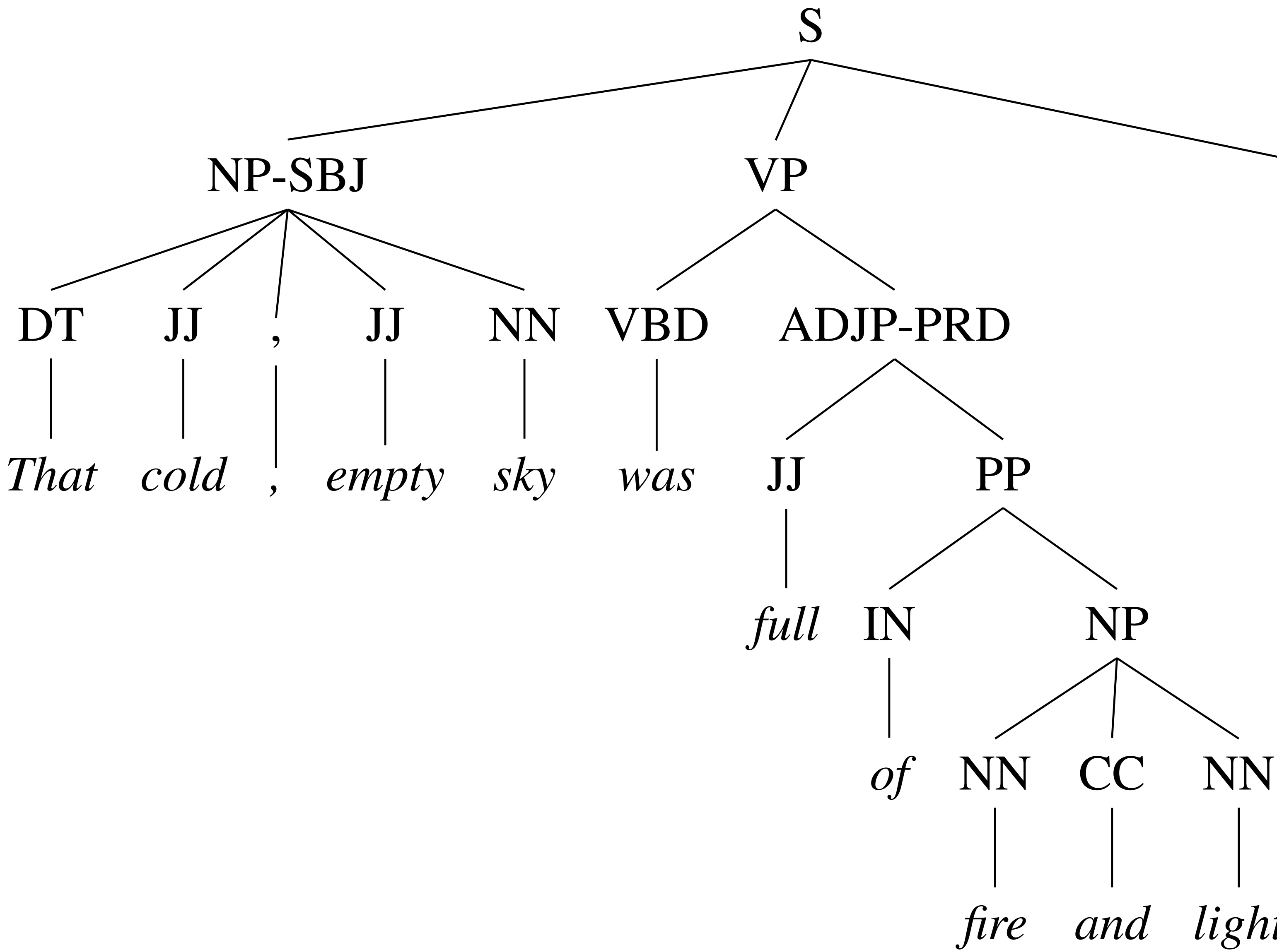
Building a treebank seems a lot slower and less useful than building a grammar

But a treebank gives us many things

- Reusability of the labor
  - Many parsers, POS taggers, etc.
  - Valuable resource for linguistics
- Broad coverage
- Frequencies and distributional information
- A way to evaluate systems

[Marcus et al. 1993, *Computational Linguistics*]

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```


S
NP-SBJ
VP
.
DT
JJ
,
JJ
NN
VBD
ADJP-PRD
.
That
cold
,
empty
sky
was
JJ
PP
full
IN
NP
of
NN
CC
NN
fire
and
light

```
( (S
    (NP-SBJ (DT The) (NN move))
    (VP (VBD followed)
      (NP
        (NP (DT a) (NN round))
        (PP (IN of)
          (NP
            (NP (JJ similar) (NNS increases))
            (PP (IN by)
              (NP (JJ other) (NNS lenders)))
            (PP (IN against)
              (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
      (, ,)
      (S-ADV
        (NP-SBJ (-NONE- *))
        (VP (VBG reflecting)
          (NP
            (NP (DT a) (VBG continuing) (NN decline))
            (PP-LOC (IN in)
              (NP (DT that) (NN market)))))))
    (. .)))
```

Some of the rules, with counts

40717 PP → IN NP
33803 S → NP-SBJ VP
22513 NP-SBJ → -NONE-
21877 NP → NP PP
20740 NP → DT NN
14153 S → NP-SBJ VP .
12922 VP → TO VP
11881 PP-LOC → IN NP
11467 NP-SBJ → PRP
11378 NP → -NONE-
11291 NP → NN
...
989 VP → VBG S
985 NP-SBJ → NN
983 PP-MNR → IN NP
983 NP-SBJ → DT
969 VP → VBN VP

100 VP → VBD PP-PRD
100 PRN → : NP :
100 NP → DT JJS
100 NP-CLR → NN
99 NP-SBJ-1 → DT NNP
98 VP → VBN NP PP-DIR
98 VP → VBD PP-TMP
98 PP-TMP → VBG NP
97 VP → VBD ADVP-TMP VP
...
10 WHNP-1 → WRB JJ
10 VP → VP CC VP PP-TMP
10 VP → VP CC VP ADVP-MNR
10 VP → VBZ S , SBAR-ADV
10 VP → VBZ S ADVP-TMP

4500 rules for VP!

NP rules

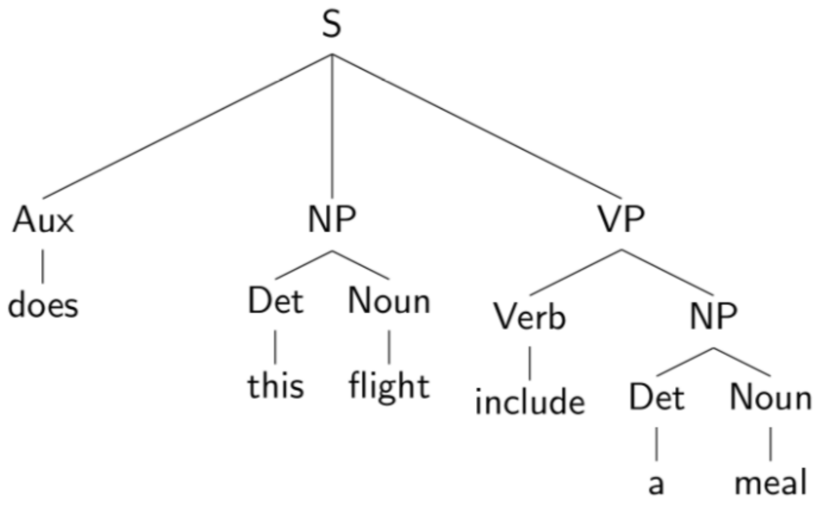
NP → DT JJ NN
NP → DT JJ NNS
NP → DT JJ NN NN
NP → DT JJ JJ NN
NP → DT JJ CD NNS
NP → RB DT JJ NN NN
NP → RB DT JJ JJ NNS
NP → DT JJ JJ NNP NNS
NP → DT NNP NNP NNP NNP JJ NN
NP → DT JJ NNP CC JJ JJ NN NNS
NP → RB DT JJS NN NN SBAR
NP → DT VBG JJ NNP NNP CC NNP
NP → DT JJ NNS , NNS CC NN NNS NN NP → DT JJ JJ VBG NN NNP NNP FW NNP NP → NP JJ , JJ ‘‘ SBAR ’’ NNS

[DT The] [JJ state-owned] [JJ industrial] [VBG holding] [NN company] [NNP Instituto] [NNP Nacional] [FW de] [NNP Industria]

[NP Shearson’s] [JJ easy-to-film], [JJ black-and-white] “[SBAR Where We Stand]” [NNS commercials]

Evaluating Parses

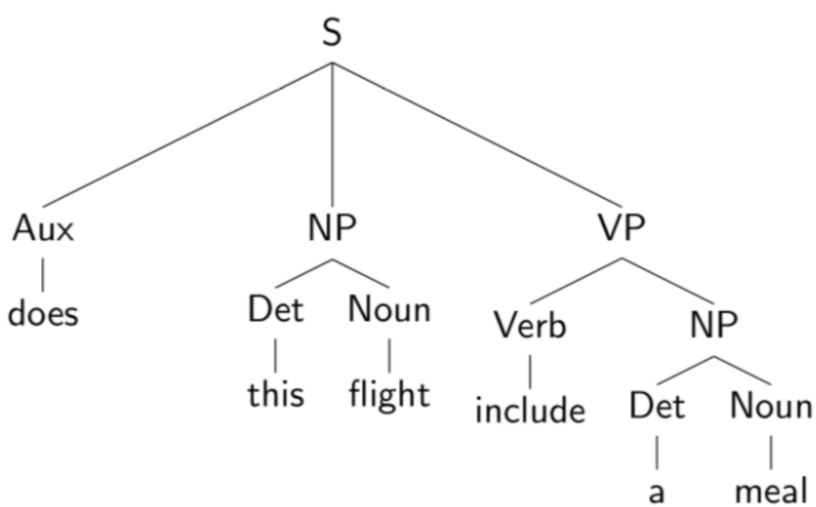
Each parse tree is represented by a list of tuples:



Use this to estimate precision/recall!

Evaluating Parses

Each parse tree is represented by a list of tuples: $\{\langle t_i, s_i, e_i \rangle\}$



$\langle S, 0, 6 \rangle$ $\langle Aux, 0, 1 \rangle$

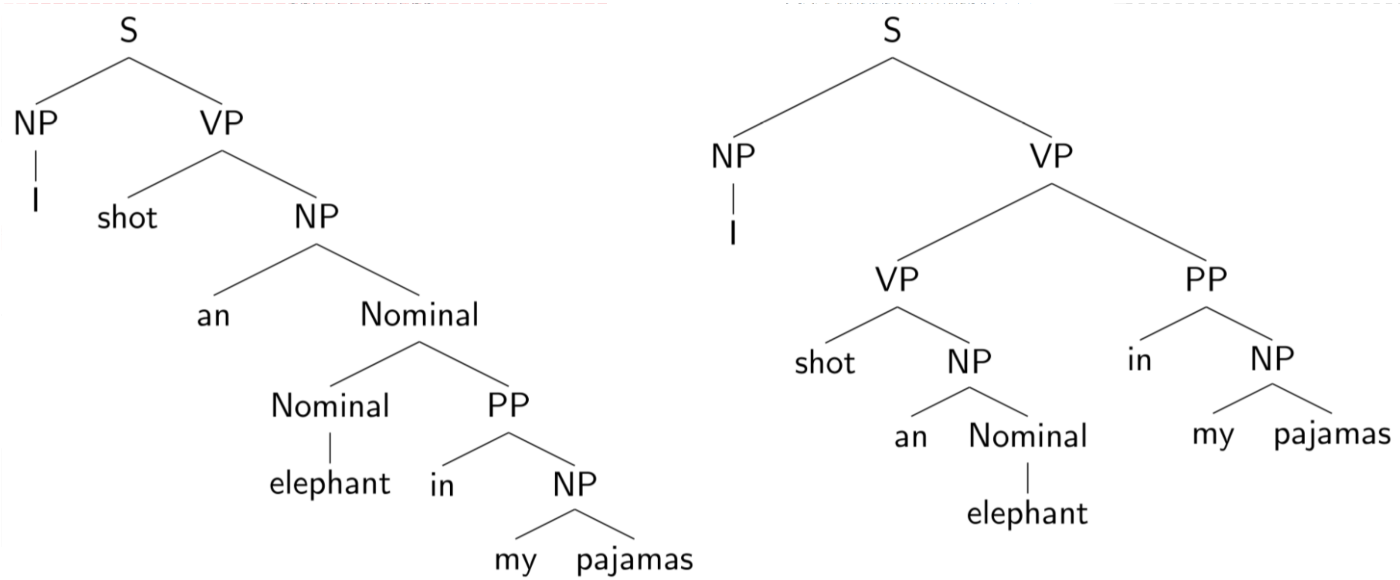
$\langle NP, 1, 3 \rangle$ $\langle DET, 1, 2 \rangle$

$\langle Noun, 2, 3 \rangle$ $\langle NP, 4, 6 \rangle$

$\langle VP, 3, 6 \rangle$ ...

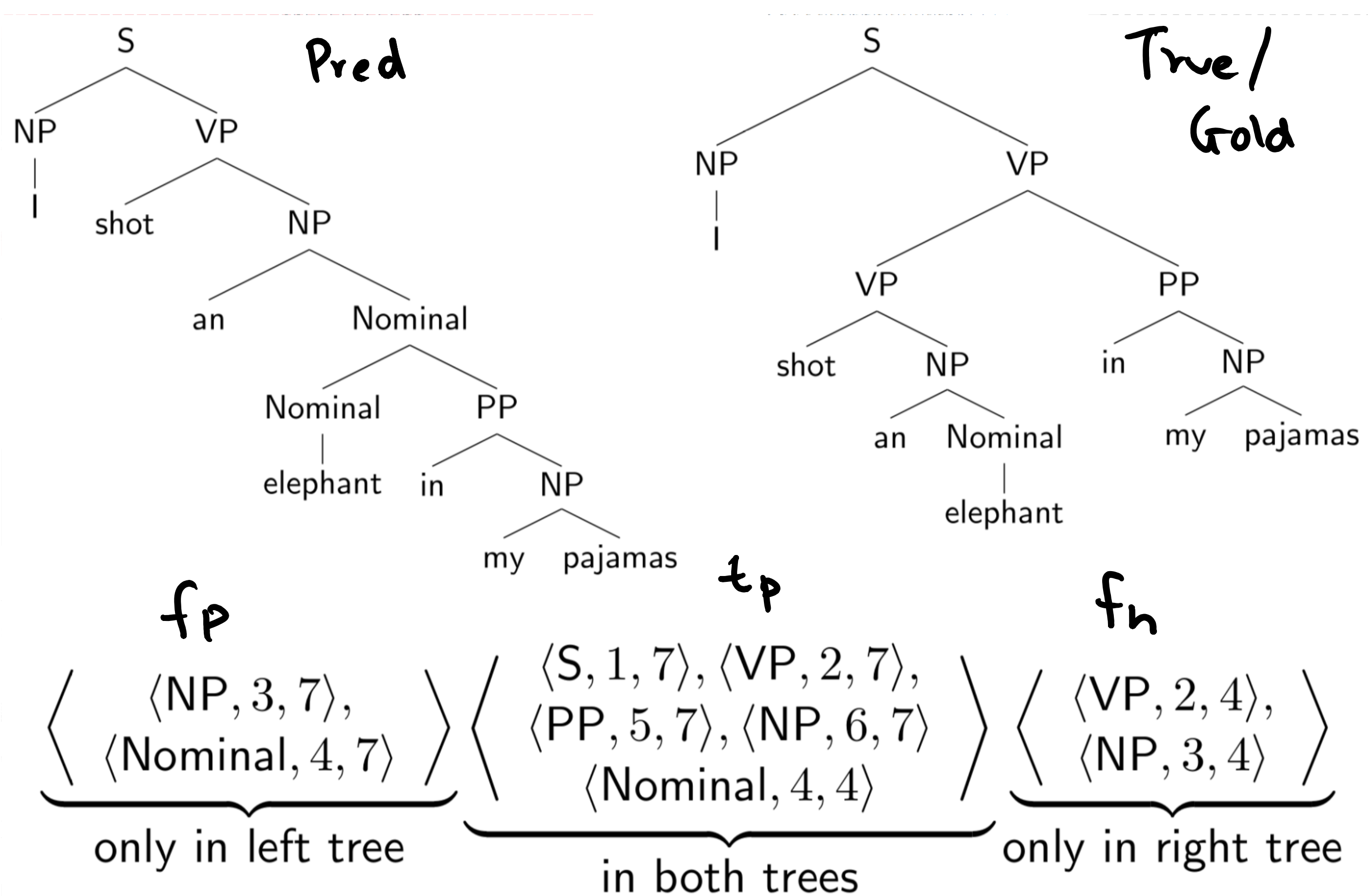
Use this to estimate precision/recall!

Evaluating Parses: Example



$$\underbrace{\left\langle \begin{matrix} \langle \text{NP}, 3, 7 \rangle, \\ \langle \text{Nominal}, 4, 7 \rangle \end{matrix} \right\rangle}_{\text{only in left tree}} \underbrace{\left\langle \begin{matrix} \langle \text{S}, 1, 7 \rangle, \langle \text{VP}, 2, 7 \rangle, \\ \langle \text{PP}, 5, 7 \rangle, \langle \text{NP}, 6, 7 \rangle \\ \langle \text{Nominal}, 4, 4 \rangle \end{matrix} \right\rangle}_{\text{in both trees}} \underbrace{\left\langle \begin{matrix} \langle \text{VP}, 2, 4 \rangle, \\ \langle \text{NP}, 3, 4 \rangle \end{matrix} \right\rangle}_{\text{only in right tree}}$$

Evaluating Parses: Example



Outline

- Context Free Grammars
- **Parsing: CKY Algorithm**
- Extensions: Probabilistic and Lexicalized
- Dependency Parsing

The Parsing Problem

Given sentence **x** and grammar **G**,

Recognition

Is sentence **x** in the grammar? If so, prove it.

"Proof" is a deduction, valid parse tree.

Parsing

Show one or more derivations for **x** in **G**.

Even with small grammars, brute force grows exponentially!

"Book that flight"

Left to Right?

The old man the boat.

The complex houses married and single soldiers and their families.

Garden Path Sentences

Left to Right?

The old man the boat.

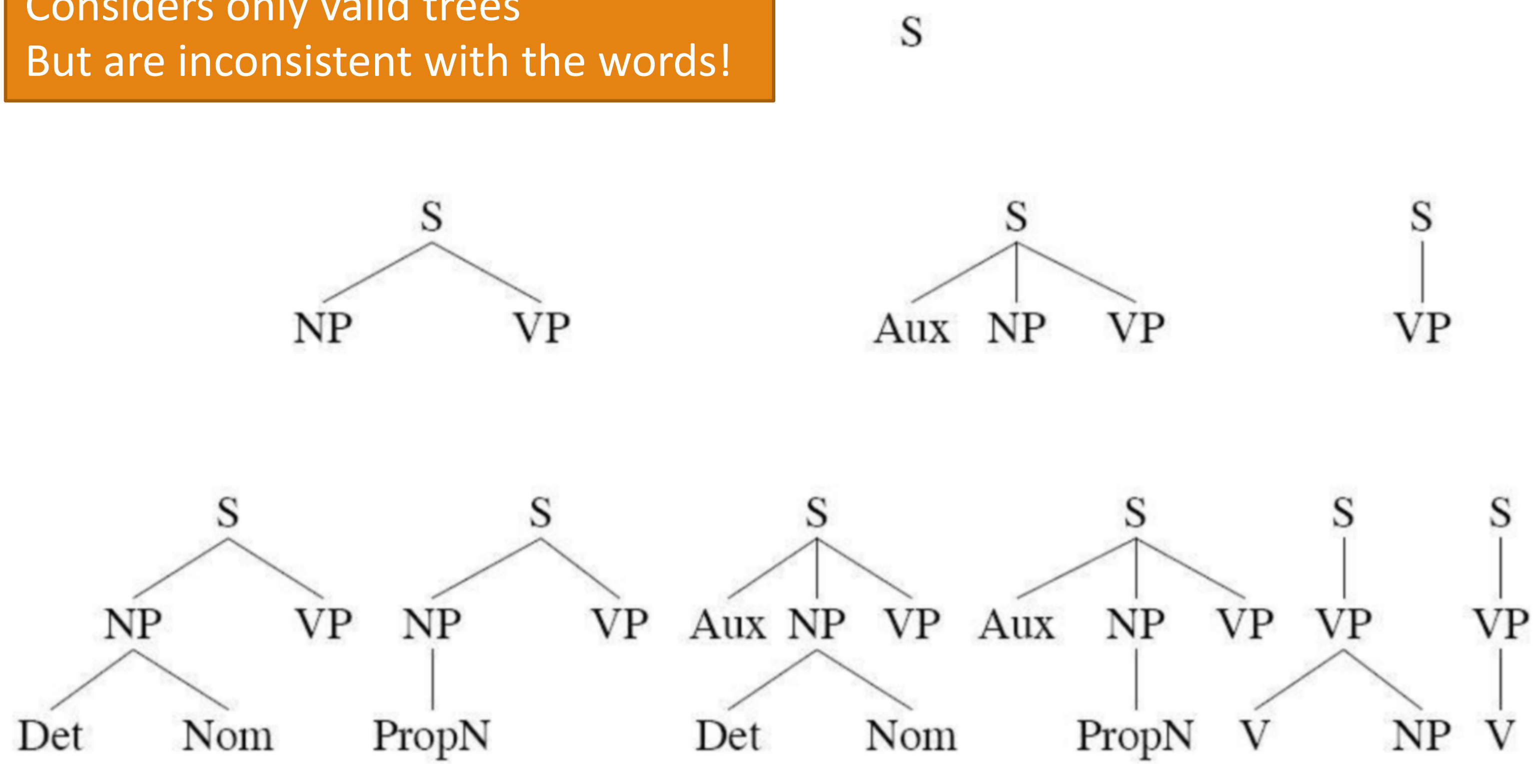
The complex houses married and single soldiers and their families.

Garden Path Sentences

Top Down Parsing

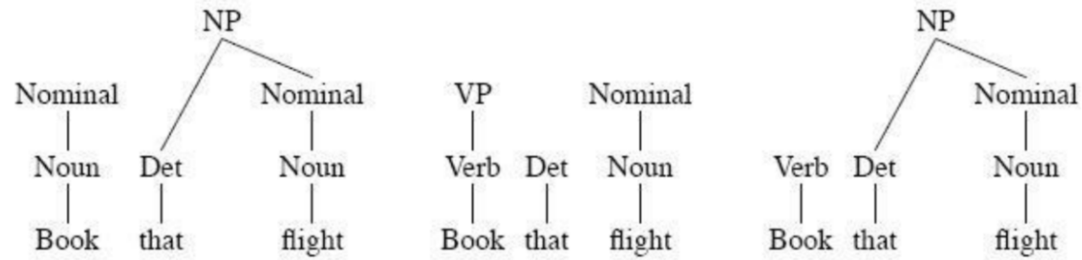
Considers only valid trees
But are inconsistent with the words!

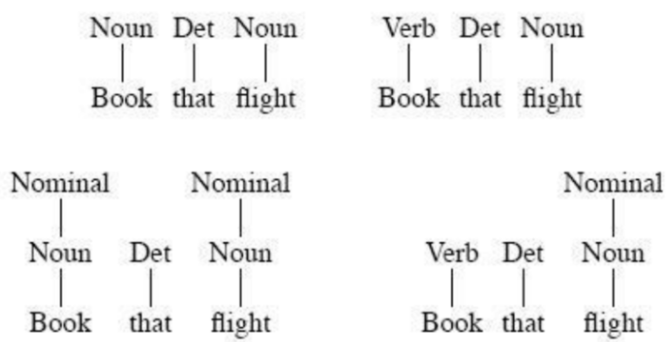
"Book that flight"

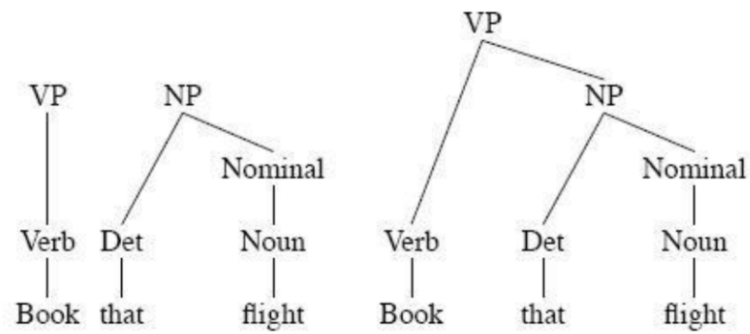


Bottom-up Parsing

"Book that flight"







Builds only consistent trees
But most of them are invalid (don't go anywhere)!

Chomsky Normal Form

Context free grammar where all non-terminals to go:

- 2 non-terminals, or
- A single terminal

$A \rightarrow B\ C$ $D \rightarrow w$

Converting to CNF

Case 1

$A \rightarrow B$
$B \rightarrow C\ D$
$B \rightarrow w$

⇒

$A \rightarrow C\ D$
$A \rightarrow w$

Case 2

$A \rightarrow B\ C\ D\ E$

⇒

$A \rightarrow X\ E$
$X \rightarrow Y\ D$
$Y \rightarrow B\ C$

Original Grammar	Chomsky Normal Form
S → NP VP	*S → NP VP*
S → Aux NP VP	*S → X1 VP*
	X1 → Aux NP
S → VP	$S \rightarrow book \mid include \mid prefer$
	S → Verb NP
	S → X2 PP
	S → Verb PP
	S → VP PP
NP → Pronoun	$NP \rightarrow I \mid she \mid me$
NP → Proper-Noun	$NP \rightarrow TWA \mid Houston$
NP → Det Nominal	*NP → Det Nominal*
Nominal → Noun	$Nominal \rightarrow book \mid flight \mid meal \mid money$
Nominal → Nominal Noun	*Nominal → Nominal Noun*
Nominal → Nominal PP	*Nominal → Nominal PP*
VP → Verb	$VP \rightarrow book \mid include \mid prefer$
VP → Verb NP	*VP → Verb NP*
VP → Verb NP PP	*VP → X2 PP*
	X2 → Verb NP
VP → Verb PP	*VP → Verb PP*
VP → VP PP	*VP → VP PP*
PP → Preposition NP	*PP → Preposition NP*

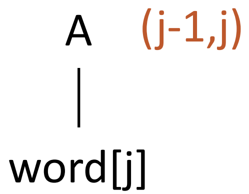
Dynamic Programming

table[i,j] = Set of all valid non-terminals for the constituent span (i,j)

Base case

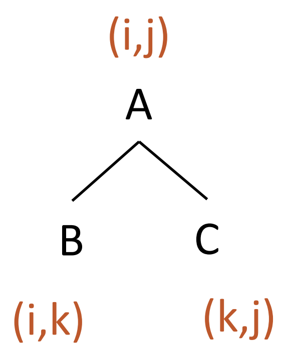
Rule: *A → word[j]*

A should be in table[j-1,j]



Recursion

Rule: *A → B C*



If you find a k such that

B is in table[i,k], and

C is in table[k,j], then A should be in table[i,j]

Dynamic Programming

S ∈ table[0,n]

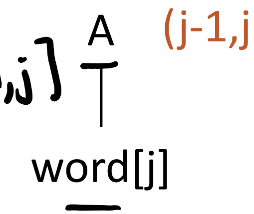
table[i,j] = Set of all valid non-terminals for the constituent span (i,j)

Base case

Rule: *A → word[j]* ←

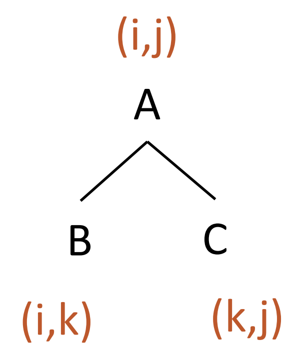
A should be in table[j-1,j]

A ∈ table[j-1,j]



Recursion

Rule: *A → B C*



If you find a k such that

B is in table[i,k], and

C is in table[k,j], then A should be in table[i,j]

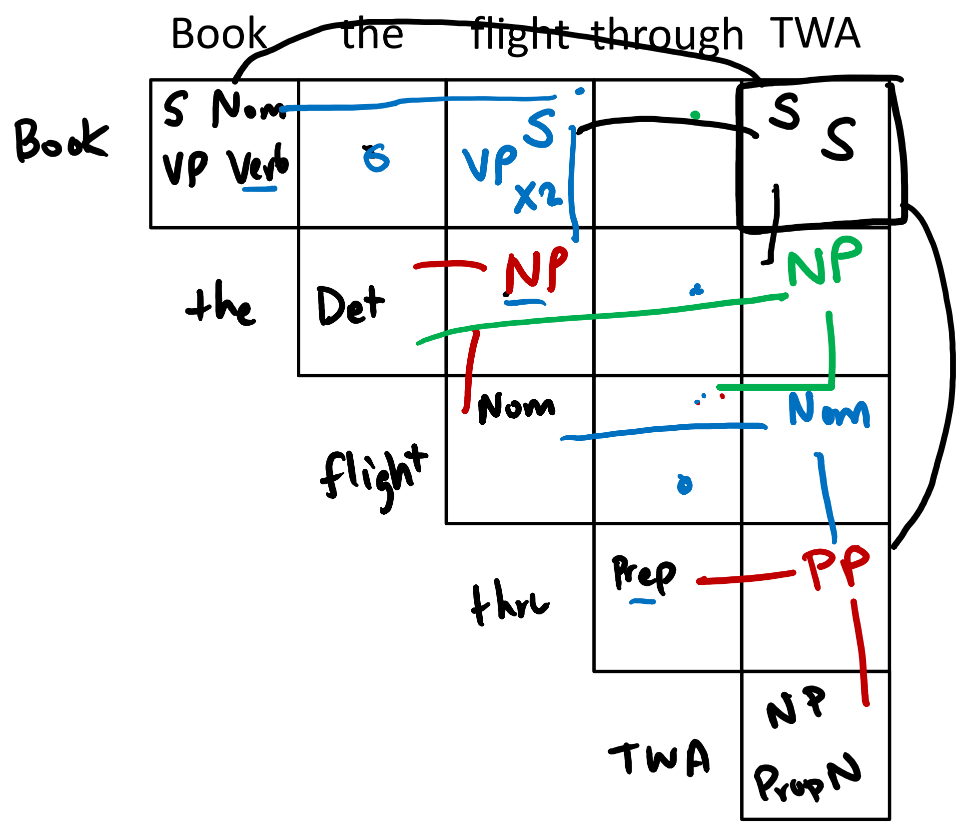
CKY Algorithm

$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

Book	the	flight	through	TWA

CKY Algorithm

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → X2 PP
S → Verb PP
S → VP PP
NP → I | she | me
NP → TWA | Houston
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → X2 PP
X2 → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm

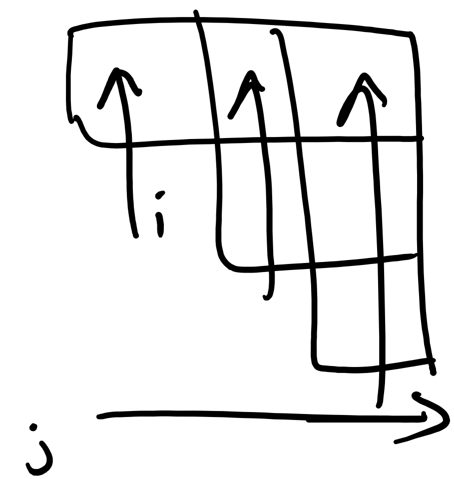
function CKY-PARSE(*words, grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{A \mid A \rightarrow words[j] \in grammar\}$
    $table[j-1, j] \leftarrow table[j-1, j] \cup A$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
        $table[i,j] \leftarrow table[i,j] \cup A$

CKY Algorithm

function CKY-PARSE(*words, grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
    for all $\{A \mid A \rightarrow words[j] \in grammar\}$
        $table[j-1, j] \leftarrow table[j-1, j] \cup A$
    for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
        for $k \leftarrow i+1$ **to** $j-1$ **do**
            for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
                $table[i,j] \leftarrow table[i,j] \cup A$



CKY Algorithm: Complexity

|N|: Number of non-terminals

|R|: Number of rules

n: Number of tokens in the sentence

Memory

Time

CKY Algorithm: Complexity

|N|: Number of non-terminals

|R|: Number of rules

n: Number of tokens in the sentence

Memory $O(n^2 |N|)$

Time $O(n^3 |R|)$

Outline

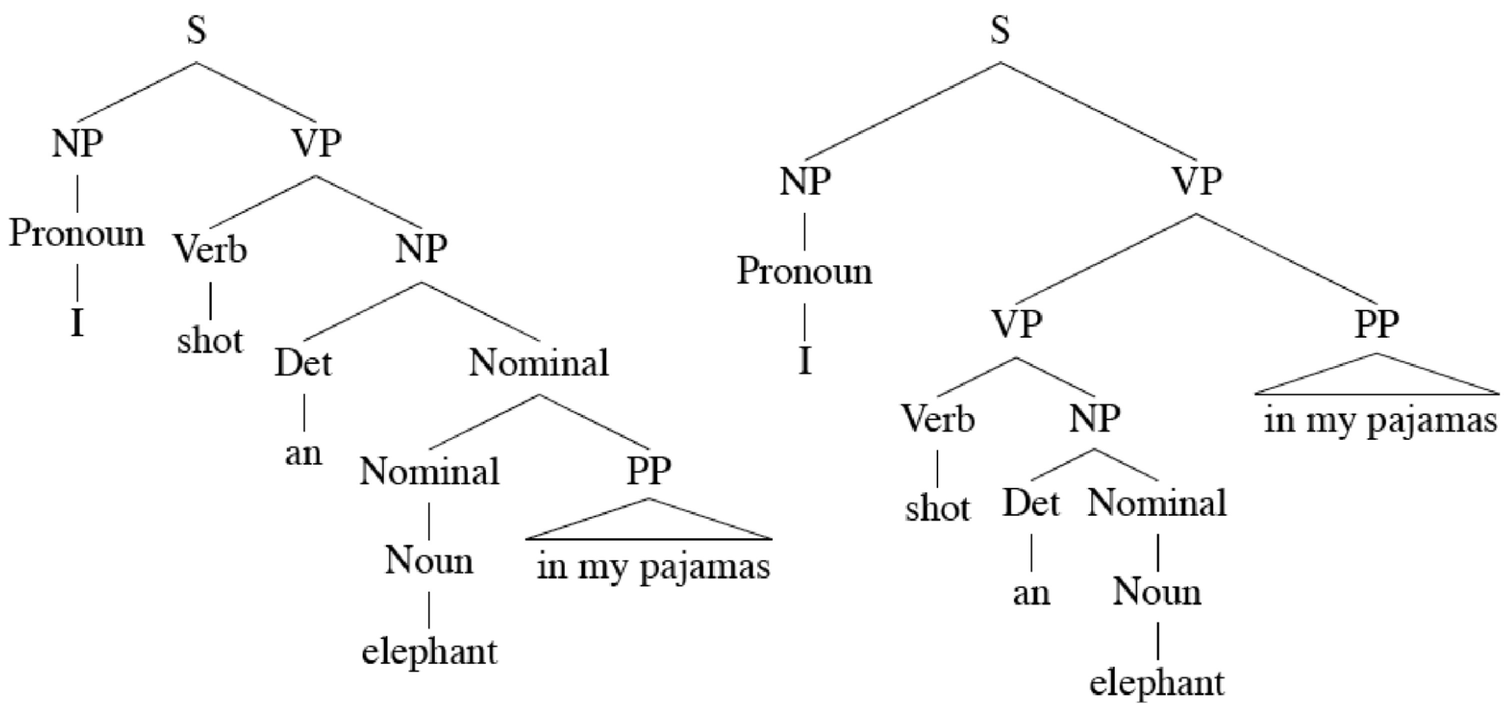
Parsing: CKY Algorithm

Extensions: Probabilistic and Lexicalized

Dependency Parsing

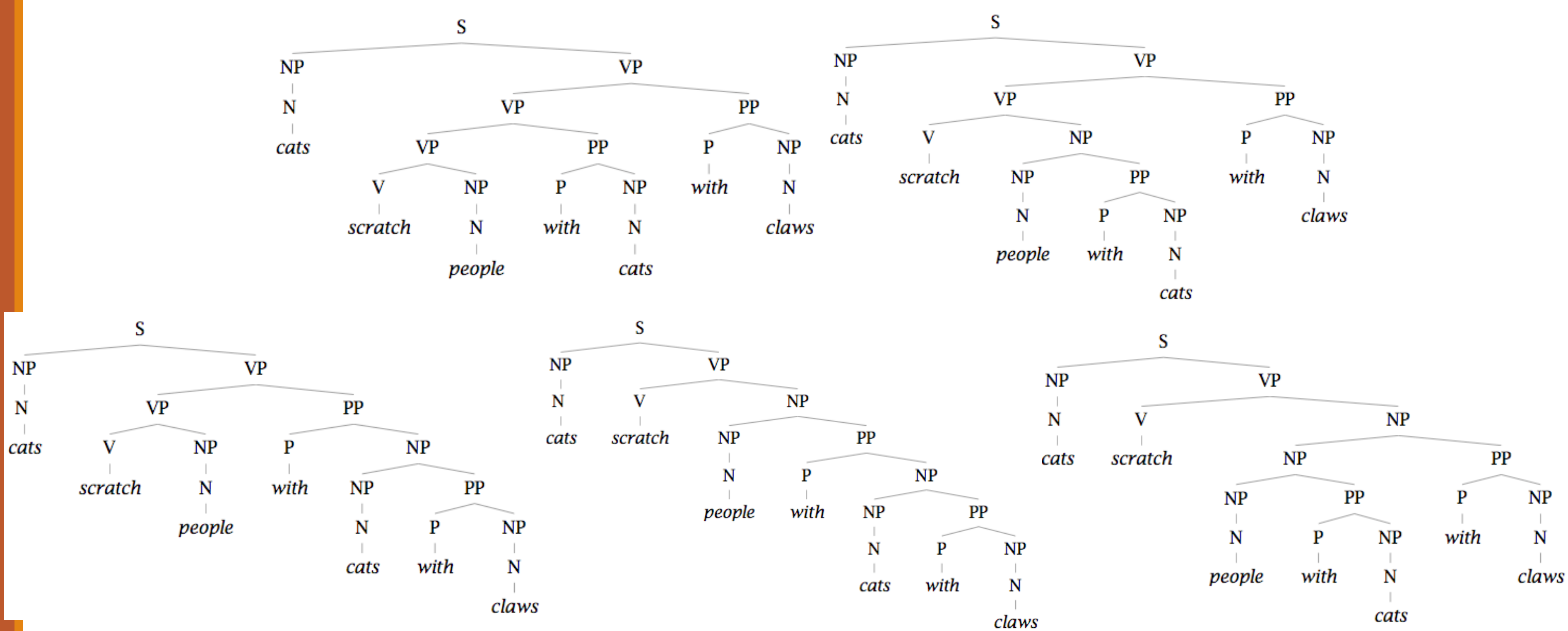
Ambiguity: Which parse?

I shot an elephant in my pajamas.



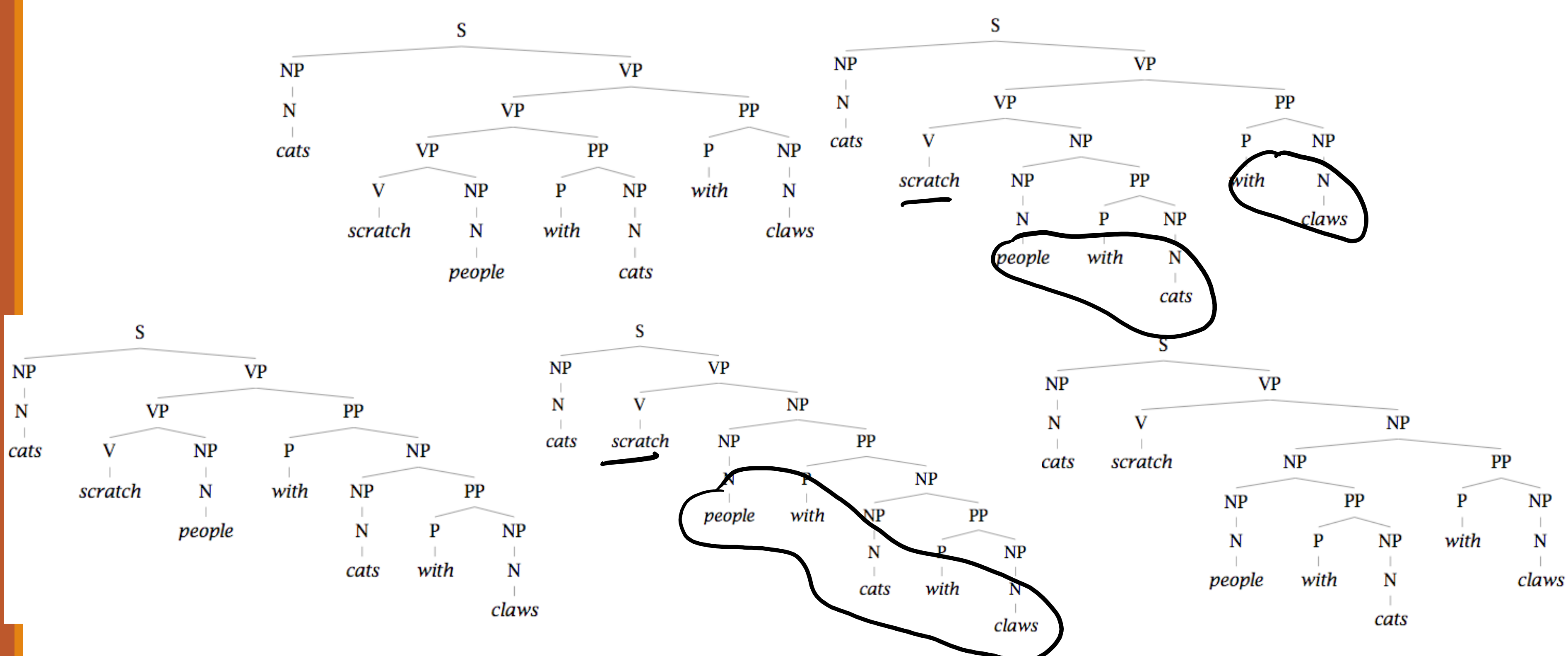
Finding the *Best* Parse Tree

Cats scratch people with cats with claws.



Finding the *Best* Parse Tree

Cats scratch people with cats with claws.



Probabilistic CFGs

Same as a regular context-free grammar:

- Terminal, non-terminals, and rules
- Additionally, attach a probability to each rule!

Rule: $A \rightarrow B\ C$

Probability: $P(A \rightarrow B\ C \mid A)$

Compute the probability of a parse tree:

Probabilistic CFGs

Same as a regular context-free grammar:

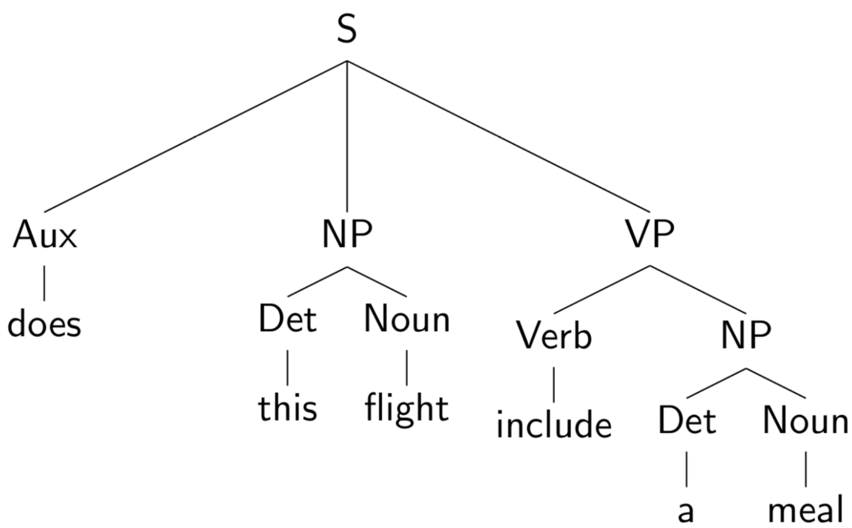
- Terminal, non-terminals, and rules
- Additionally, attach a probability to each rule!

Rule: $A \rightarrow B\ C$

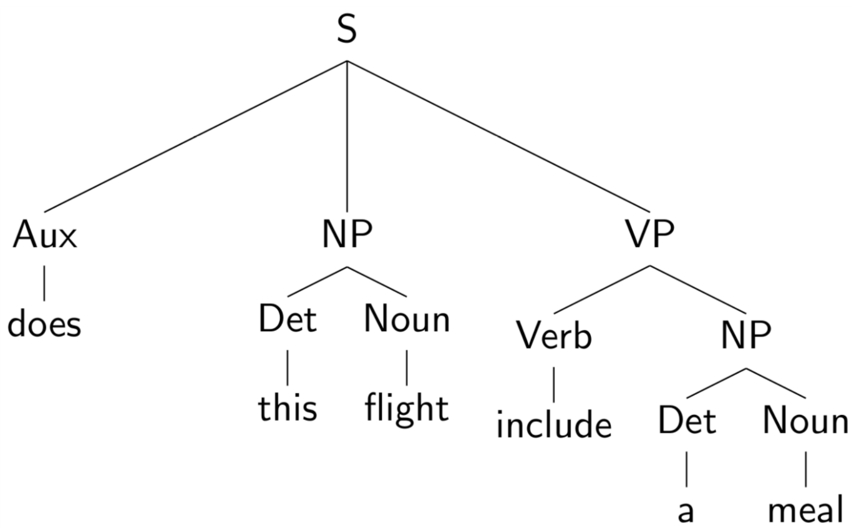
Probability: $P(A \rightarrow B\ C \mid A)$

Compute the probability of a parse tree: $\prod_{A \rightarrow BC \in T} P(A \rightarrow BC \mid A)$

Example of a PCFG



Example of a PCFG



$P(\text{Aux NP VP} \mid S)$

$P(\text{does} \mid \text{AUX})$

$P(\text{Det Noun} \mid NP)$

$P(\text{this} \mid \text{DET})$ $P(\text{flight} \mid \text{Noun})$

⋮

Estimating the probabilities

Estimating the probabilities

$$P(\alpha \to \beta \mid \alpha) = \frac{\# \alpha \to \beta}{\# \alpha}$$

The Parsing Problem

Given sentence **x** and grammar **G**,

Recognition

Is sentence **x** in the grammar? If so, prove it.

"Proof" is a deduction, valid parse tree.

Parsing

Show one or more derivations for **x** in **G**.

$$\operatorname*{argmax}_{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{x}}} p(\boldsymbol{t} \mid \boldsymbol{x})$$

Even with small grammars, grows exponentially!

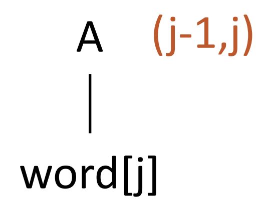
Probabilistic CKY Algorithm

$T[i,j,A]$ = Probability of the best parse with root A for the span (i,j)

Base case

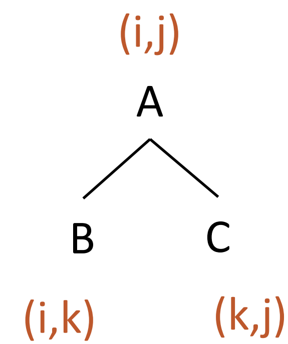
Rule: $P(A \rightarrow word[j])$

$T[j-1,j,A] = P(word[j] \mid A)$



Recursion

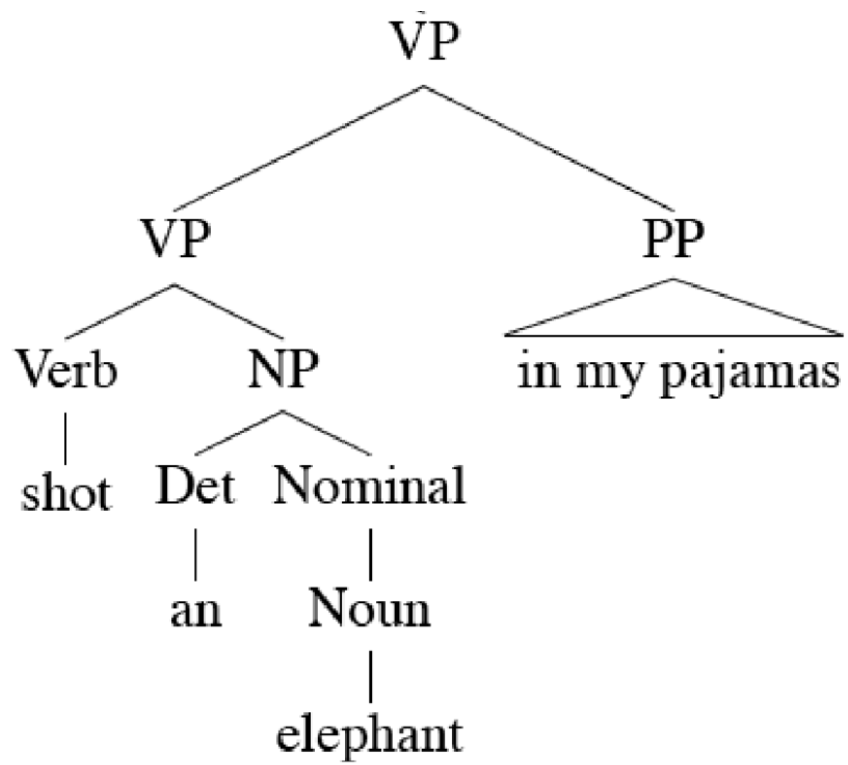
Rule: $P(A \rightarrow B\ C)$



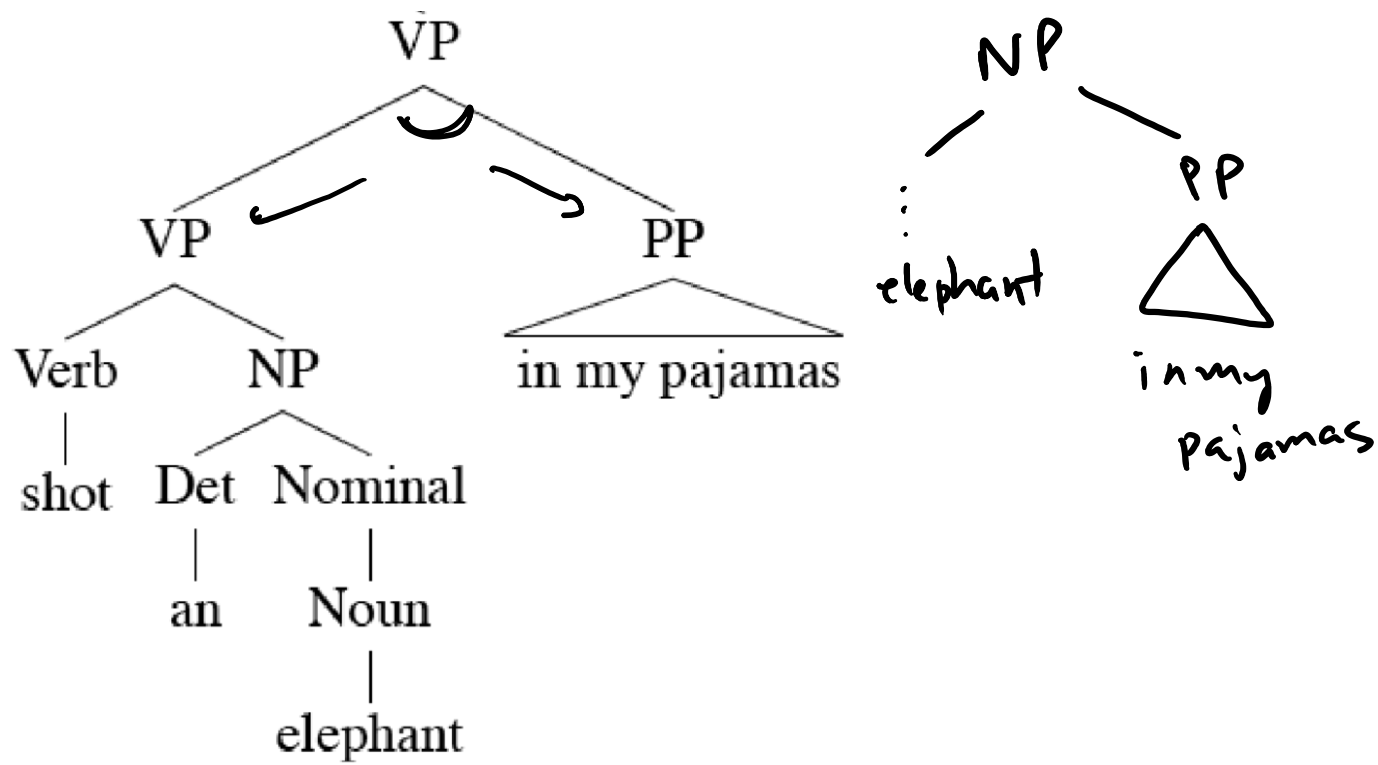
Try every position k, and every non-terminal pair:

$$T[i,j,A] = \max_{k} \; P(B\ C \mid A)\; T[i,k,B]\; T[k,j,C]$$

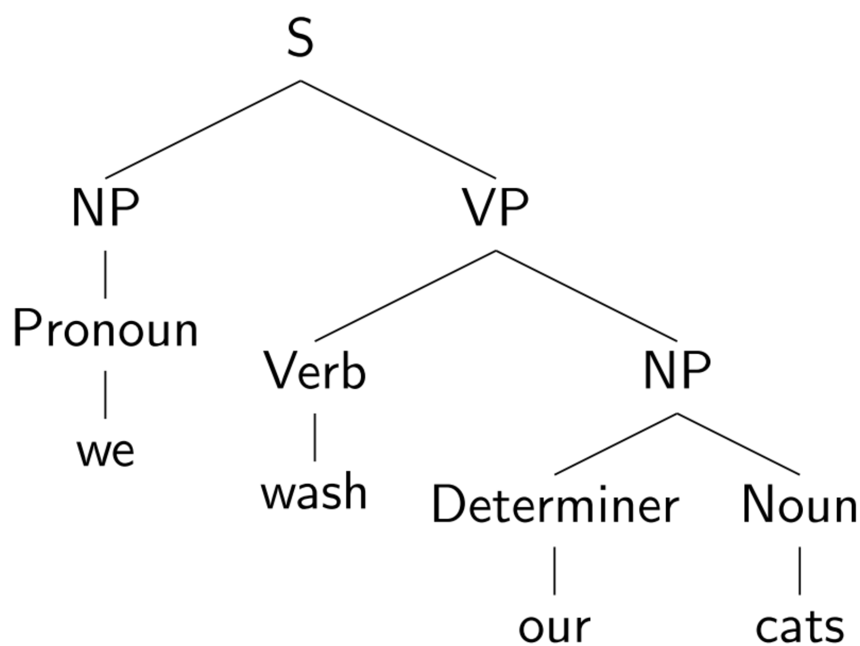
Lexicalized PCFGs



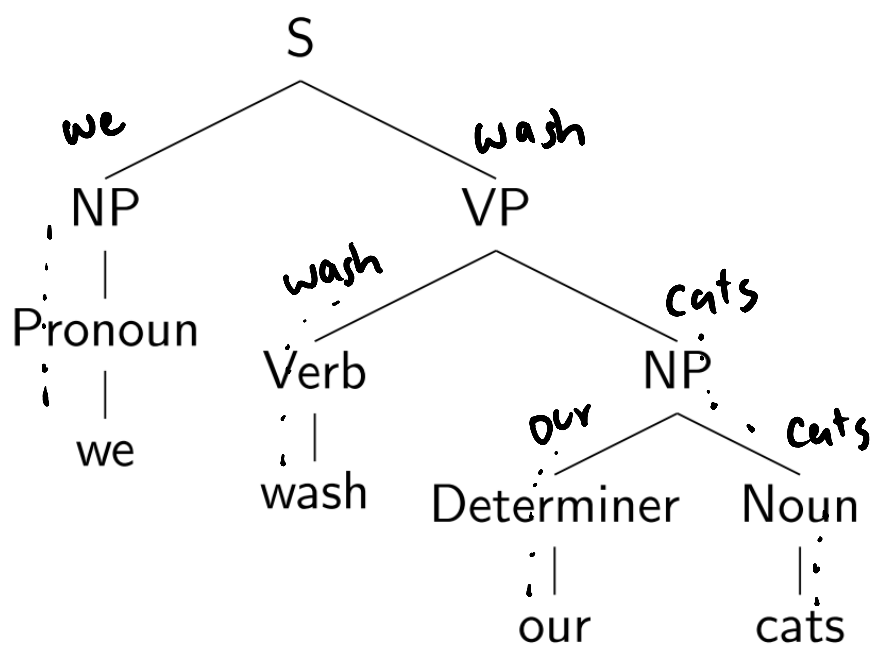
Lexicalized PCFGs



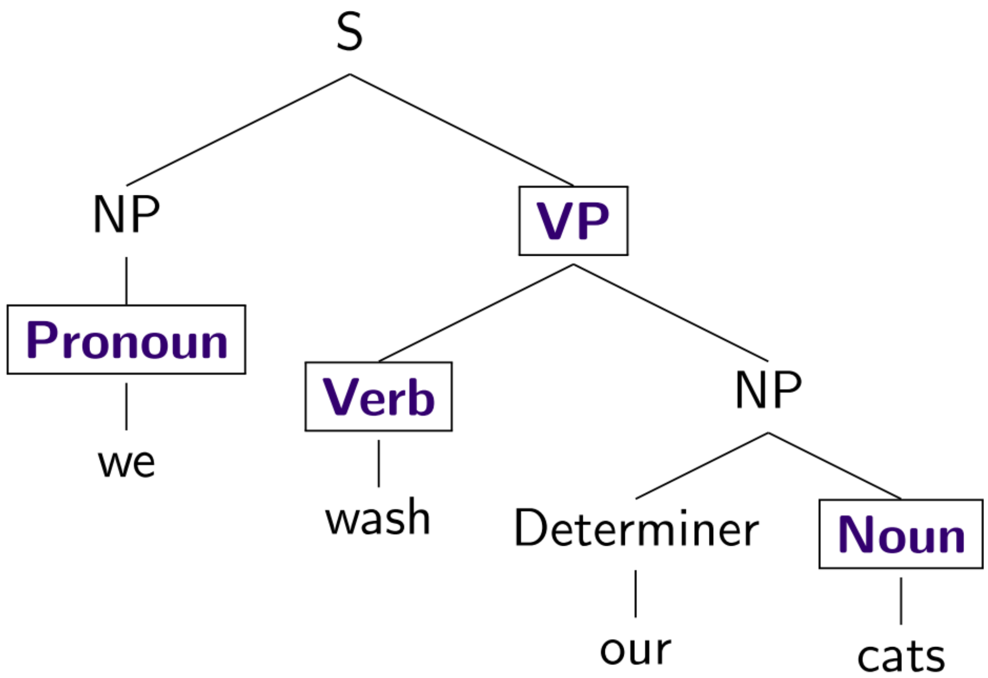
Lexicalizing a CFG



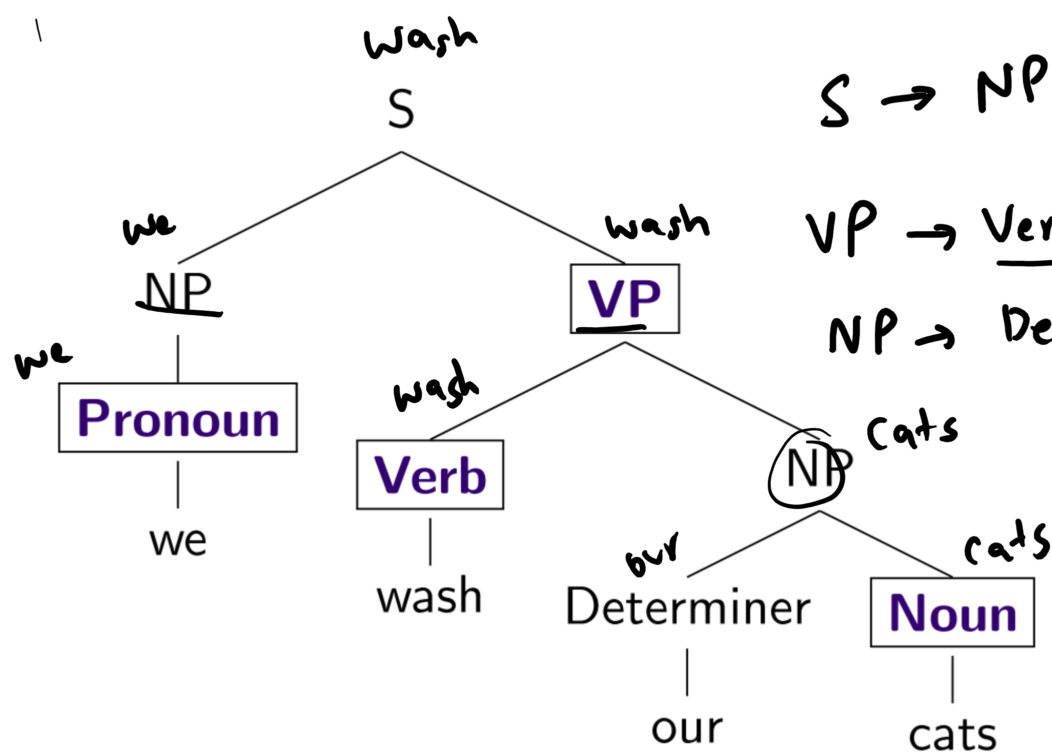
Lexicalizing a CFG



Lexicalizing a CFG



Lexicalizing a CFG

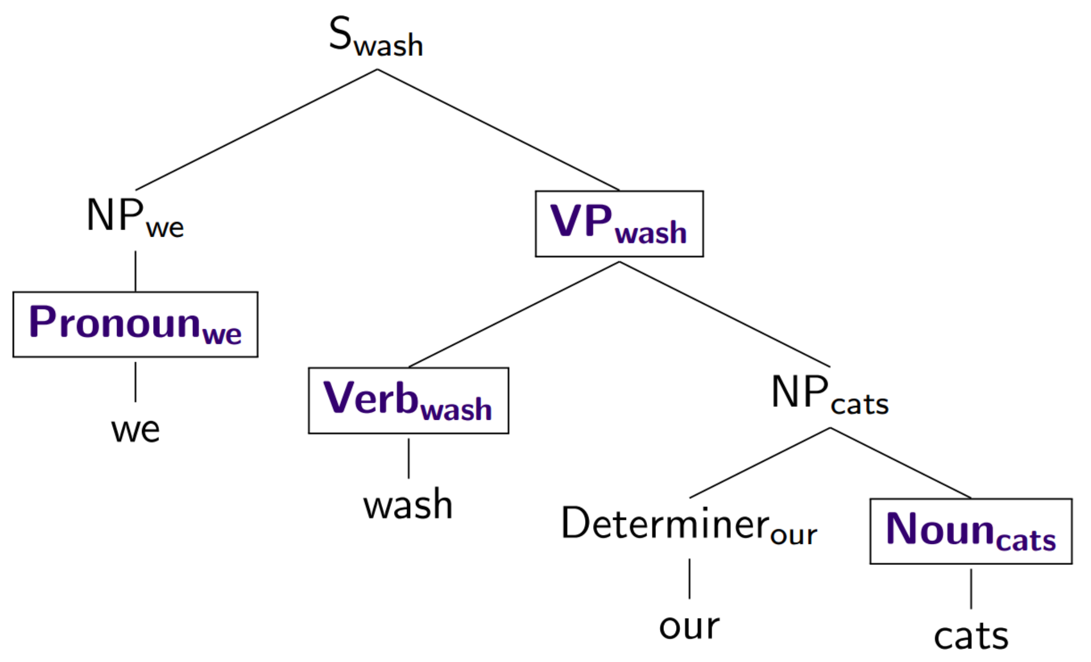


S → NP VP*

VP → Verb* NP

NP → Det Noun*

Lexicalizing a CFG



Outline

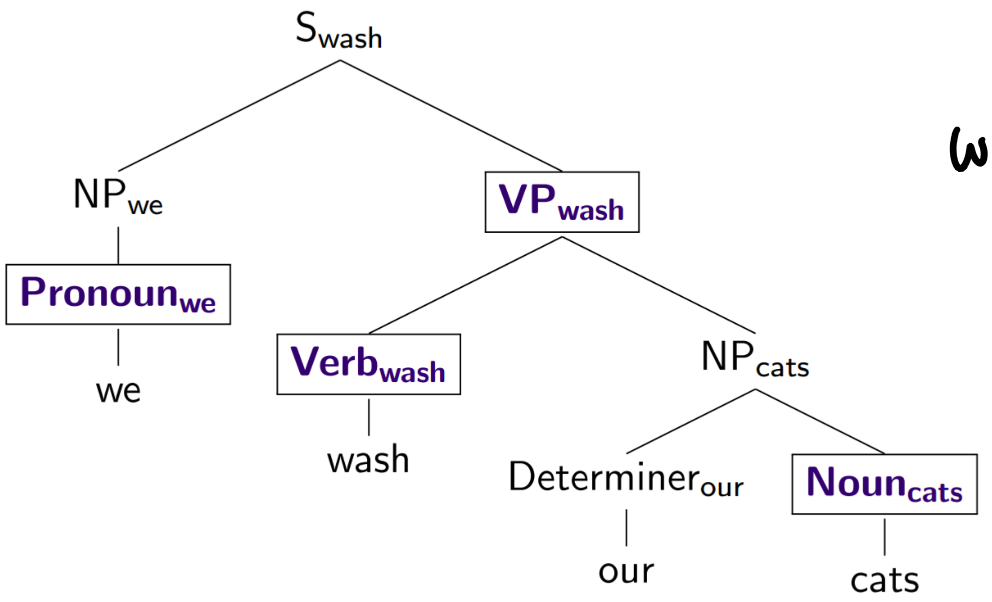
Parsing: CKY Algorithm

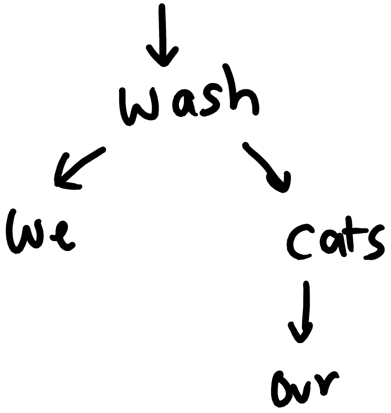
Extensions: Probabilistic and Lexicalized

Dependency Parsing

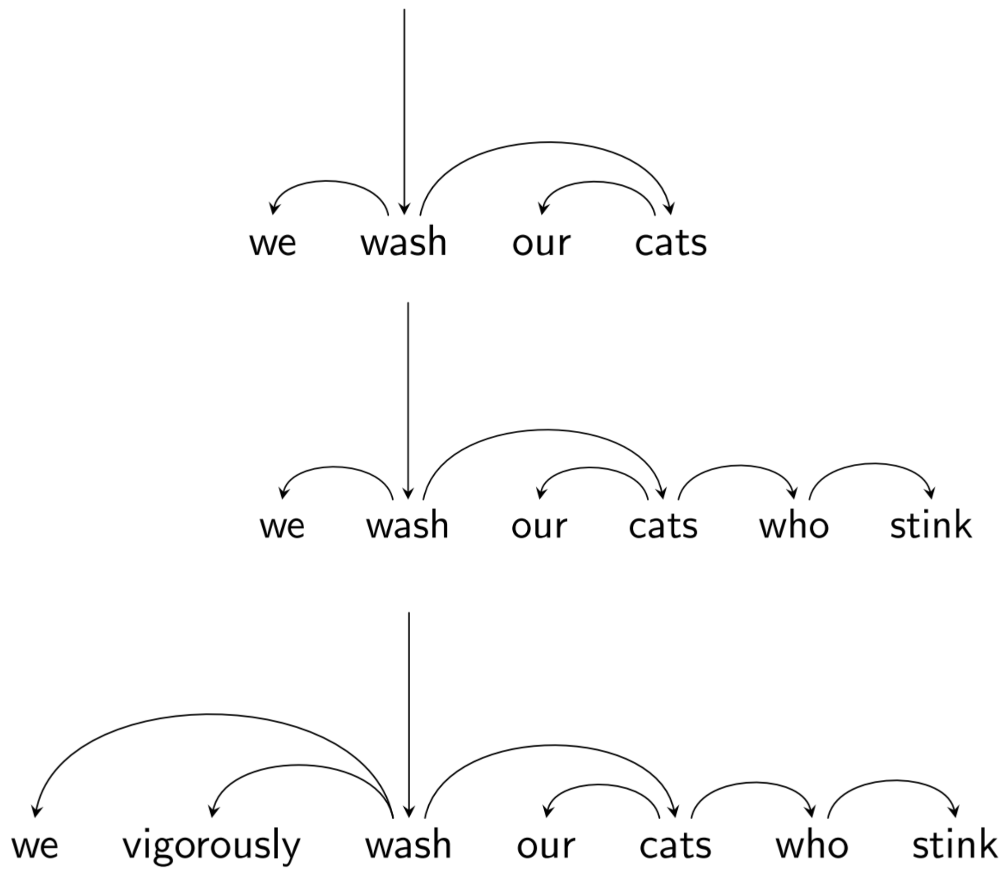
Dependencies

Represent only the syntactic dependencies…

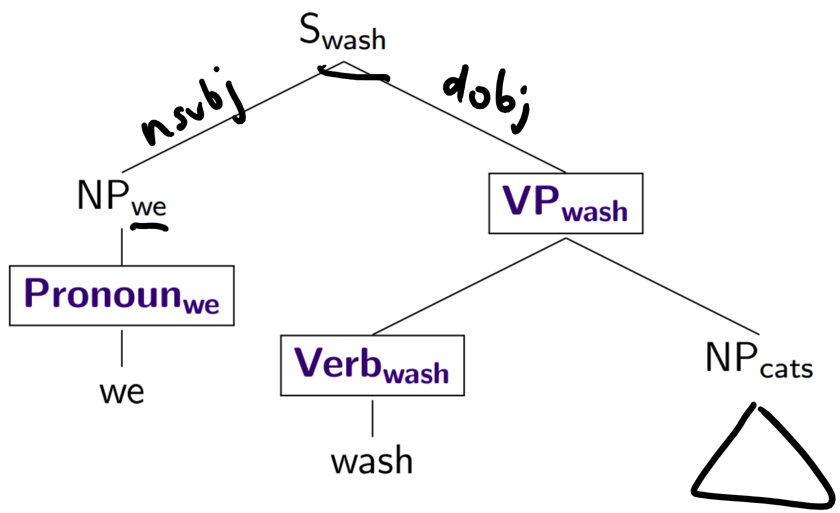


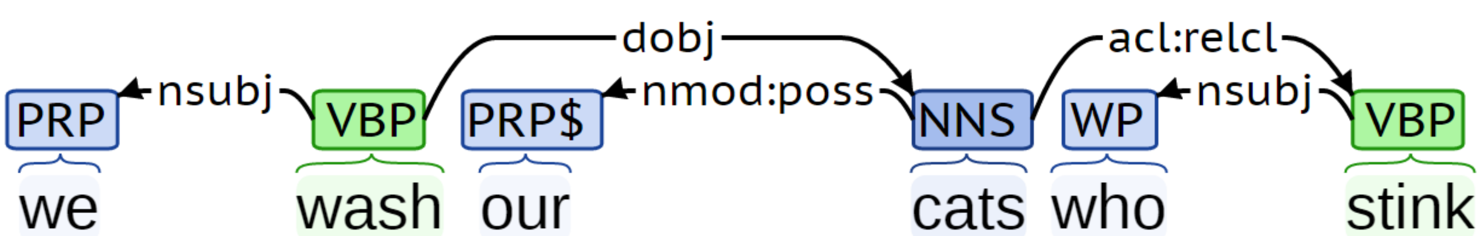


Nested Structure = Subtrees



Dependency Labels





Dependency Labels

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

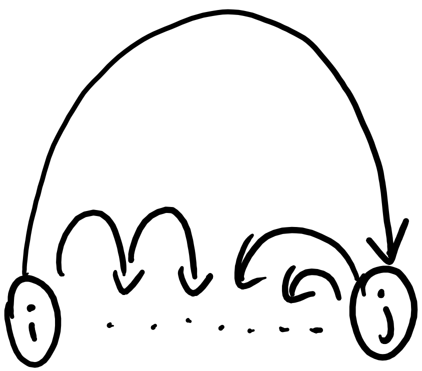
Dependency Trees

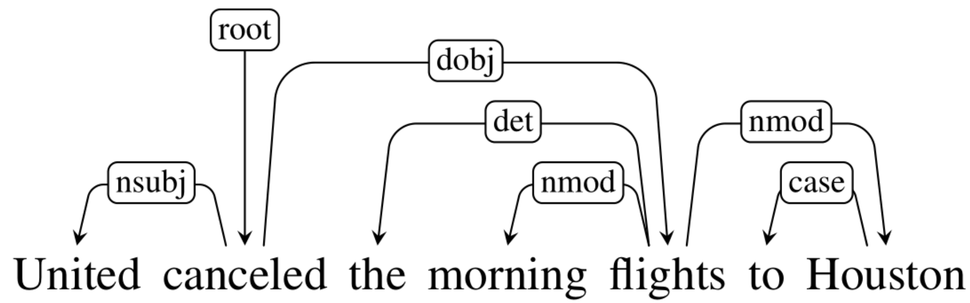
$$\{ \langle c_i, p_i, l_i \rangle \}_{i=1 \ldots n}$$

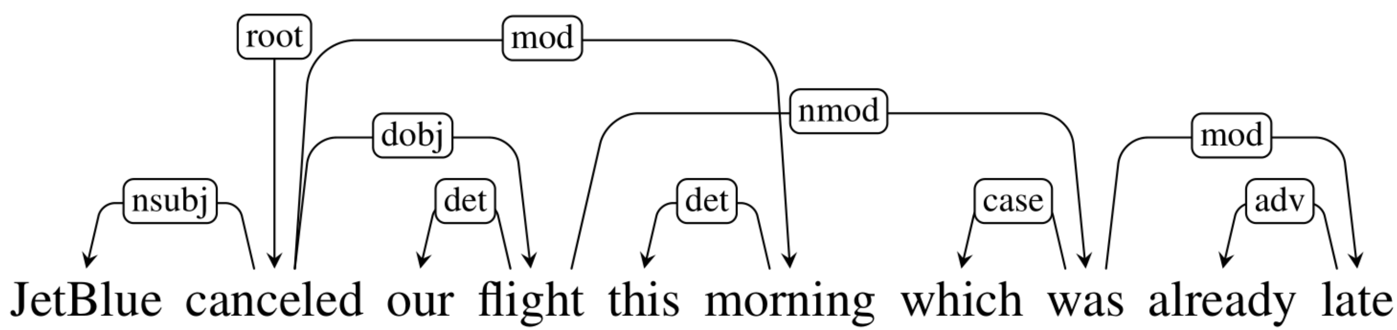
- c_i: $1:n$
- p_i: $0:n$ (0 ↑ root)
- l_i: label

- single headed $\exists i$ only one $p_i = 0$
- connected
- acyclic
- projective vs non-projective

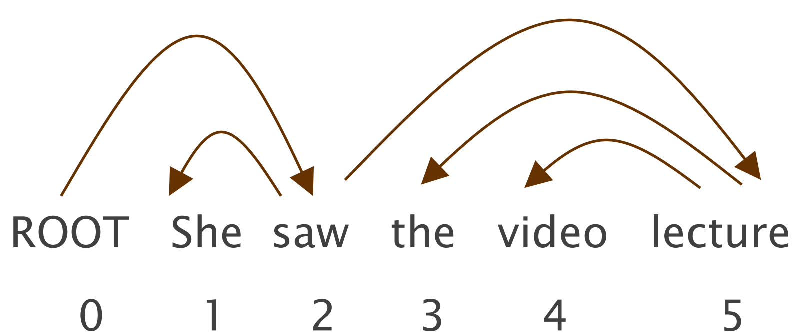
Projective vs Non-projective

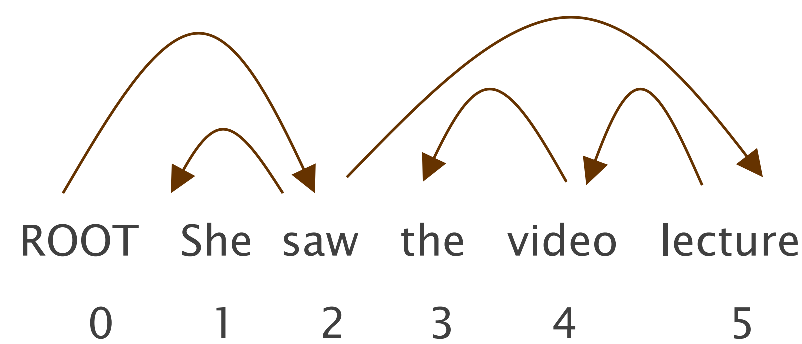






Evaluating Dependency Parses





Gold

1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn

UAS $= \frac{4}{5} = 80\%$

LAS $= \frac{2}{5} = 40\%$

Parsed

1	2 ✓	She	nsubj ✓
2	0 ✗	saw	root ✗
3	4 ✓	the	det ✗
4	5 ✓	video	nsubj ✗
5	2	lecture	

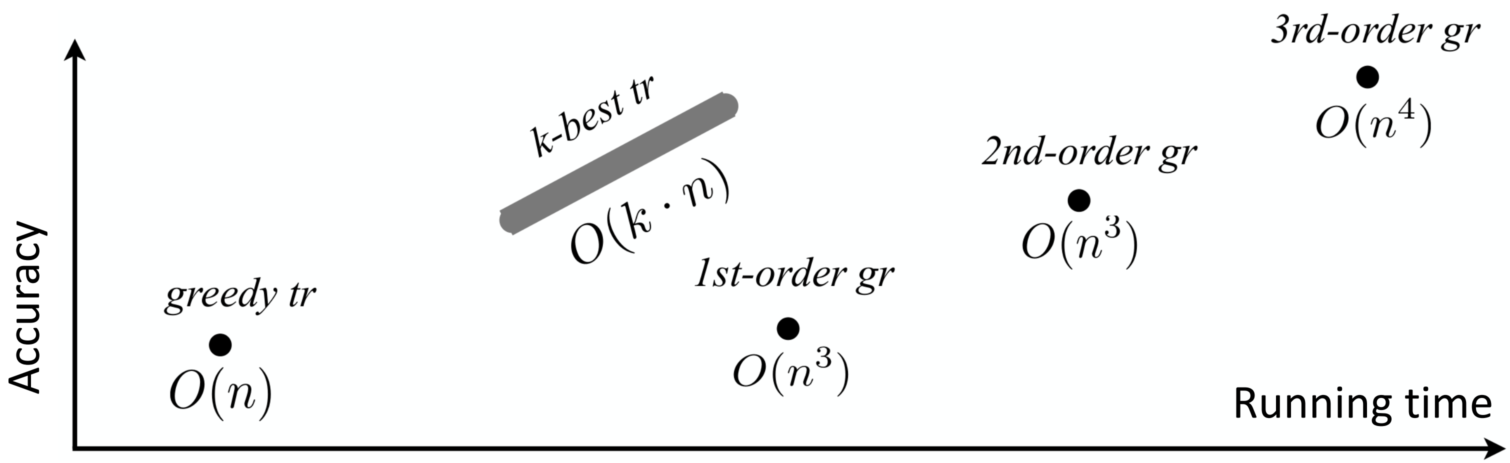
Parsing Algorithms

Transition-based

- Fast, greedy, linear-time
- Trained for greedy search
- Features decide what to do next
- Beam search, i.e. *k*-best

Graph-based

- Slower, exhaustive algorithms
- Dynamic programming, inference
- Features used to score whole trees



Graph-based Parsing

$$\underset{t \in T}{\operatorname{argmax}} \ \text{score}(t, \theta)$$

1st order / fully factored

$$\sum_i \theta_i \phi(e_i) \quad (c_i, p_i, l_i)$$

Proj: Dynamic Programming

NonProj: Maximum Spanning Tree

2nd order $\phi(e_i, e_j)$

3rd order $\phi(e_i, e_j, e_k)$

Transition-based Parsing

→ action

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	

Transition-based Parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	

Transition-based Parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book $\rightarrow$ me)

Transition-based Parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	

Transition-based Parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)

Transition-based Parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	→ [root, book, flight]	[]	RIGHTARC ←	(book → flight)
9	[root, book]	[]	RIGHTARC ←	(root → book)
10	[root]	[]	Done ←	

$\theta \cdot \phi(\text{stack, wlist, action})$