Parsing Context-free grammars – Part 1

ICS 482 Natural Language Processing

Lecture 12: Parsing Context-free grammars – Part 1

Husni Al-Muhtaseb

بسم الله الرحمن الرحيم
ICS 482 Natural Language Processing

Lecture 12: Parsing Context-free grammars – Part 1

Husni Al-Muhtaseb

NLP Credits and Acknowledgment

These slides were adapted from presentations of the Authors of the book

SPEECH and LANGUAGE PROCESSING:

An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition

and some modifications from presentations found in the WEB by several scholars including the following

NLP Credits and Acknowledgment

If your name is missing please contact me
muhtaseb
At
Kfupm.
Edu.
sa

NLP Credits and Acknowledgment

Husni Al-Muhtaseb
James Martin
Jim Martin
Dan Jurafsky
Sandiway Fong
Song young in
Paula Matuszek
Mary-Angela Papalaskari
Dick Crouch
Tracy Kin
L. Venkata Subramaniam
Martin Volk
Bruce R. Maxim
Jan Hajič
Srinath Srinivasa
Simeon Ntafos
Paolo Pirjanian
Ricardo Vilalta
Tom Lenaerts

Heshaam Feili
Björn Gambäck
Christian Korthals
Thomas G. Dietterich
Devika Subramanian
Duminda Wijesekera
Lee McCluskey
David J. Kriegman
Kathleen McKeown
Michael J. Ciaraldi
David Finkel
Min-Yen Kan
Andreas Geyer-Schulz
Franz J. Kurfess
Tim Finin
Nadjet Bouayad
Kathy McCoy
Hans Uszkoreit
Azadeh Maghsoodi

Khurshid Ahmad
Staffan Larsson
Robert Wilensky
Feiyu Xu
Jakub Piskorski
Rohini Srihari
Mark Sanderson
Andrew Elks
Marc Davis
Ray Larson
Jimmy Lin
Marti Hearst
Andrew McCallum
Nick Kushmerick
Mark Craven
Chia-Hui Chang
Diana Maynard
James Allan

Martha Palmer
julia hirschberg
Elaine Rich
Christof Monz
Bonnie J. Dorr
Nizar Habash
Massimo Poesio
David Goss-Grubbs
Thomas K Harris
John Hutchins
Alexandros Potamianos
Mike Rosner
Latifa Al-Sulaiti
Giorgio Satta
Jerry R. Hobbs
Christopher Manning
Hinrich Schütze
Alexander Gelbukh
Gina-Anne Levow
Guitao Gao
Qing Ma
Zeynep Altan

Previous Lectures

- Introduction and Phases of an NLP system
- NLP Applications - Chatting with Alice
- Finite State Automata & Regular Expressions & languages
- Morphology: Inflectional & Derivational
- Parsing and Finite State Transducers
- Stemming & Porter Stemmer
- Statistical NLP – Language Modeling
- N Grams
- Smoothing and NGram: Add-one & Witten-Bell
- Parts of Speech
- Arabic Parts of Speech
- Syntax: Context Free Grammar (CFG): Derivation, Parsing, Recursion, Agreement, Subcategorization

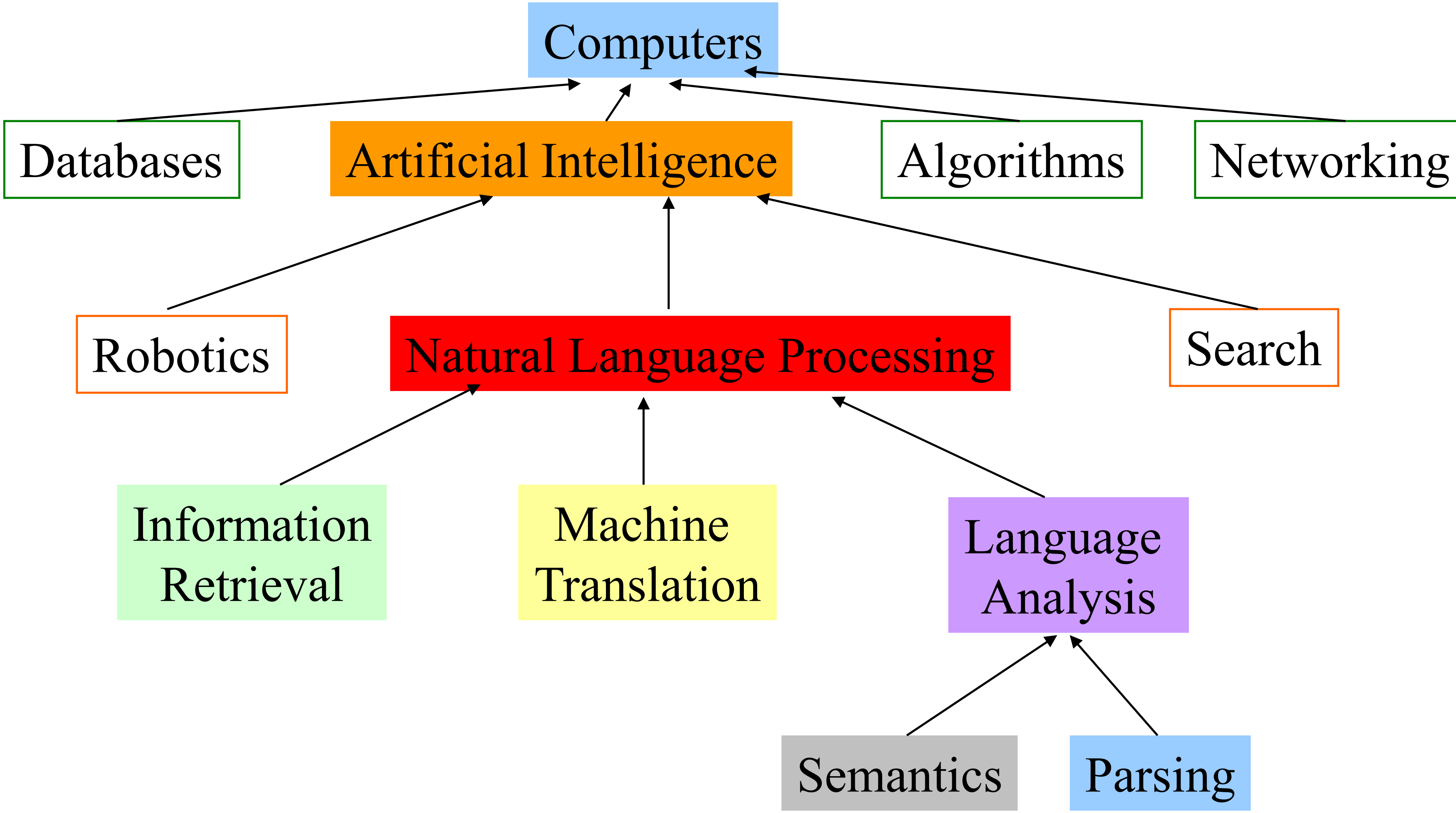
Today's Lecture

- Parsing Context Free Grammars
  - **T**op-**D**own (**TD**)
  - **B**ottom-**U**p (**BU**)
  - Top-Down Depth-First Left-to-Right (TD DF L2R)
  - Top-down parsing with bottom-up filtering
  - Dynamic
    - Earley’s Algorithm

Reminder

- Quiz 2
  - Tuesday 3rd April 2007?
  - Class time
  - Covered material
  - Textbook: Ch 6, 8, 9, covered part of 10
  - We are not covering Speech related material

Where does NLP fit in the CS taxonomy?



The Steps in NLP

Discourse

Pragmatics

Semantics

Syntax

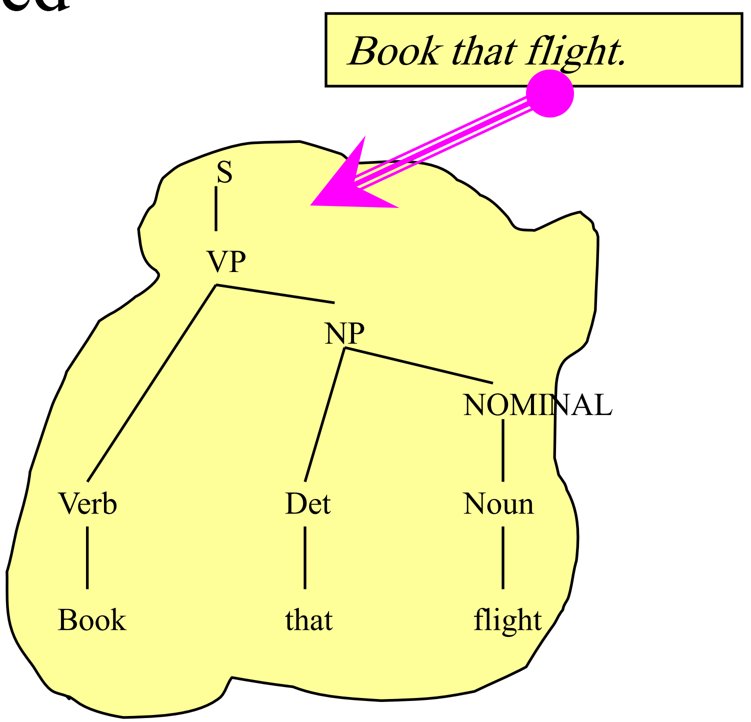
Morphology

**we can go up, down and up and down and combine steps too!!

**every step is equally complex

Introduction

- Parsing = associating a structure (parse tree) to an input string using a grammar
- CFG are declarative, they don't specify how the parse tree will be constructed
- Parse trees are used in
  - Grammar checking
  - Semantic analysis
  - Machine translation
  - Question answering
  - Information extraction



Parsing

- Parsing with CFGs refers to the task of assigning correct trees to input strings
- Correct here means a tree that covers all and only the elements of the input and has an S at the top
- It doesn't actually mean that the system can select the correct tree from among the possible trees

Parsing

- Parsing involves a search which involves the making of choices
- Some Parsing techniques:
  - Top-down parsing
  - Bottom-up parsing

For Now

- Assume…
  - You have all the words already in some buffer
  - The input isn't POS tagged
  - We won't worry about morphological analysis
  - All the words are known

Parsing as search

A Grammar to be used in our example

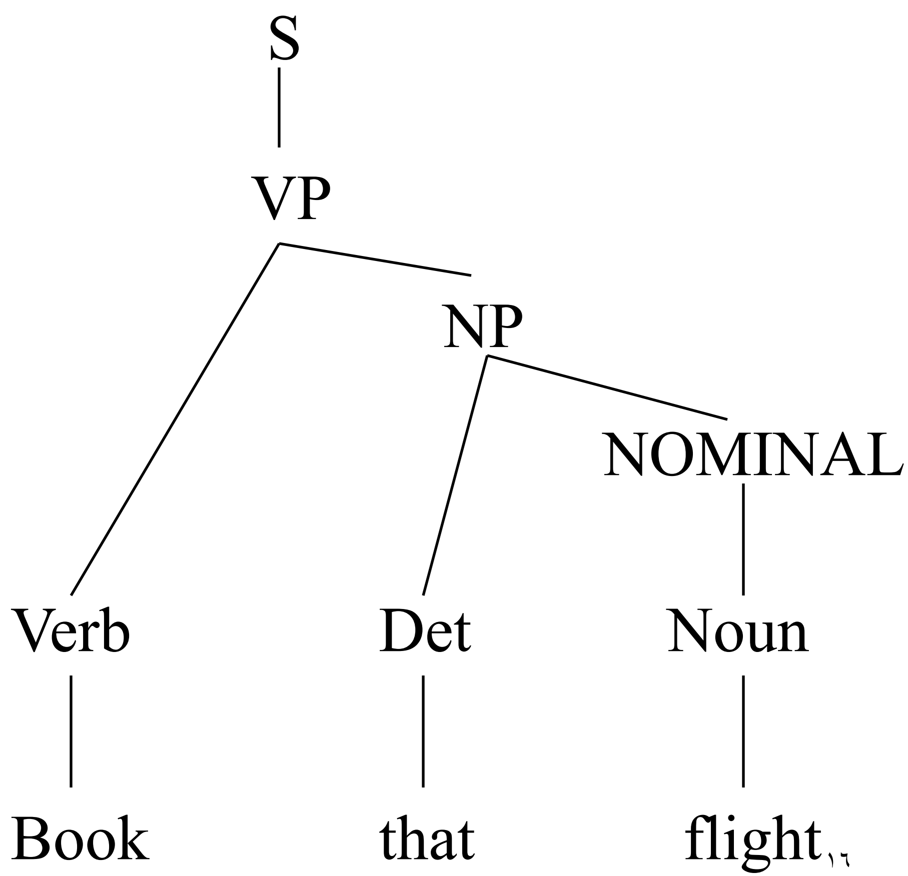
S → NP VP	VP → Verb NP PP
S → Aux NP VP	VP → Verb PP
S → VP	VP → VP PP
NP → Pronoun	PP → Preposition NP
NP → Det NOMINAL	Det → that | this |a
NP → Proper-Noun	Noun → book | flight | meal | money
NOMINAL → Noun	Verb → book | include | prefer
NOMINAL → NOMINAL Noun	Pronoun → I | she | me
NOMINAL → NOMINAL PP	Aux → does
VP → Verb	Proper-Noun → Houston | TWA
VP → Verb NP	Preposition → from | to | on | near | through

Parsing as search

S → NP VP	VP → Verb NP PP
S → Aux NP VP	VP → Verb PP
S → VP	VP → VP PP
NP → Pronoun	PP → Preposition NP
NP → Det NOMINAL	Det → that | this |a
NP → Proper-Noun	Noun → book | flight | meal | money
NOMINAL → Noun	Verb → book | include | prefer
NOMINAL → NOMINAL Noun	Pronoun → I | she | me
NOMINAL → NOMINAL PP	Aux → does
VP → Verb	Proper-Noun → Houston | TWA
VP → Verb NP	Preposition → from | to | on | near | through

Book that flight.

- Two types of constraints on the parses:
  1. some that come from the input string
  2. others that come from the grammar



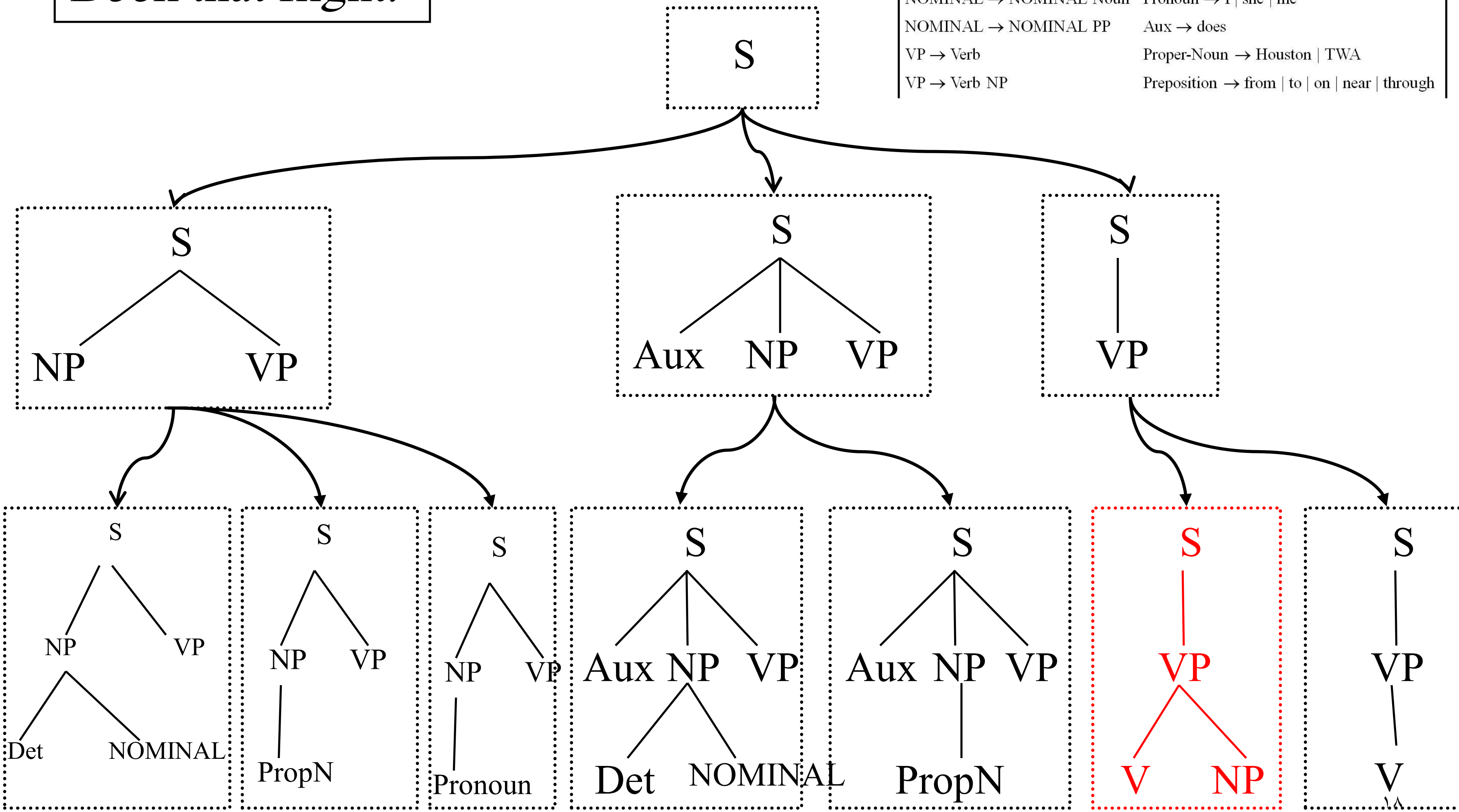
Top-Down Parsing (TD)

- Since we’re trying to find trees rooted with an S (Sentences) start with the rules that give us an S
- Then work your way down from there to the words

Top-down parsing (TD)

S → NP VP	VP → Verb NP PP
S → Aux NP VP	VP → Verb PP
S → VP	VP → VP PP
NP → Pronoun	PP → Preposition NP
NP → Det NOMINAL	Det → that | this |a
NP → Proper-Noun	Noun → book | flight | meal | money
NOMINAL → Noun	Verb → book | include | prefer
NOMINAL → NOMINAL Noun	Pronoun → I | she | me
NOMINAL → NOMINAL PP	Aux → does
VP → Verb	Proper-Noun → Houston | TWA
VP → Verb NP	Preposition → from | to | on | near | through

Book that flight.

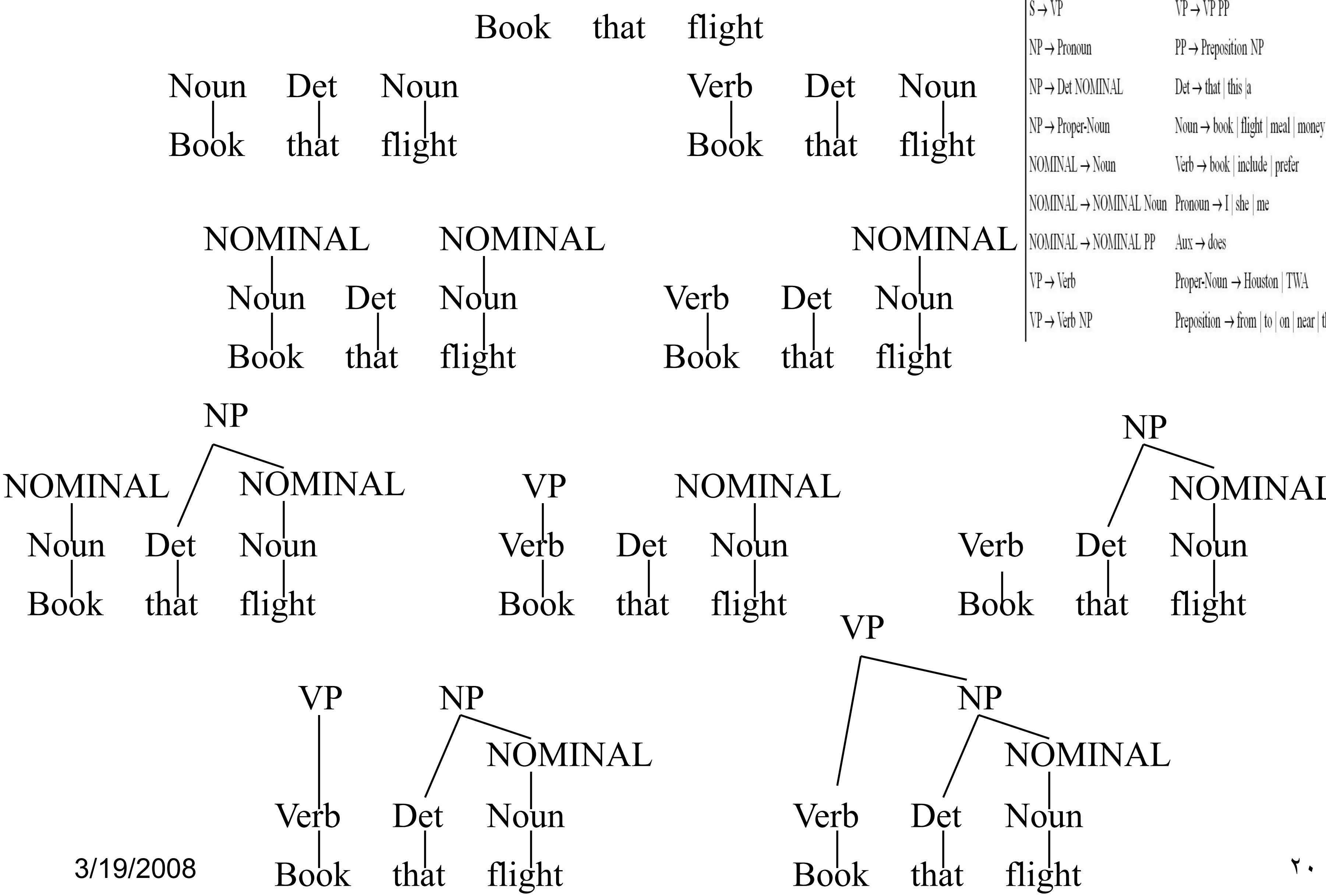


Bottom-Up Parsing

- Since we want trees that cover the input words start with trees that link up with the words in the right way.
- Then work your way up from there.

Bottom-up parsing (BU)

S → NP VP	VP → Verb NP PP
S → Aux NP VP	VP → Verb PP
S → VP	VP → VP PP
NP → Pronoun	PP → Preposition NP
NP → Det NOMINAL	Det → that | this | a
NP → Proper-Noun	Noun → book | flight | meal | money
NOMINAL → Noun	Verb → book | include | prefer
NOMINAL → NOMINAL Noun	Pronoun → I | she | me
NOMINAL → NOMINAL PP	Aux → does
VP → Verb	Proper-Noun → Houston | TWA
VP → Verb NP	Preposition → from | to | on | near | through

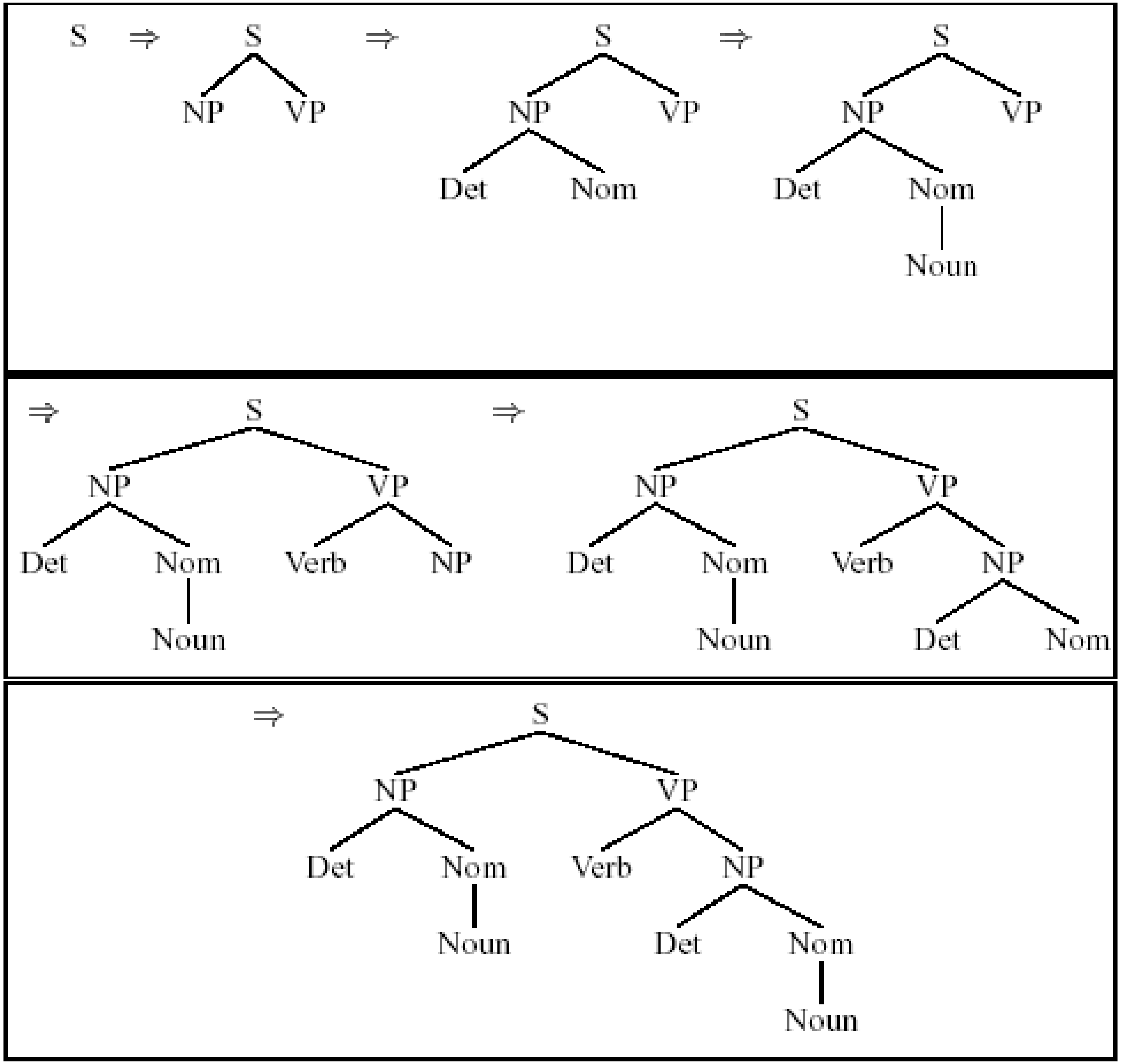


Comparing Top-Down and Bottom-Up

- Top-Down parsers never explore illegal parses (never explore trees that can't form an S) -- but waste time on trees that can never match the input
- Bottom-Up parsers never explore trees inconsistent with input -- but waste time exploring illegal parses (Explore trees with no S root)
- For both: how to explore the search space?
  - Pursuing all parses in parallel or …?
  - Which rule to apply next?
  - Which node to expand next?
- Needed: some middle ground.

Basic Top-Down (TD) parser

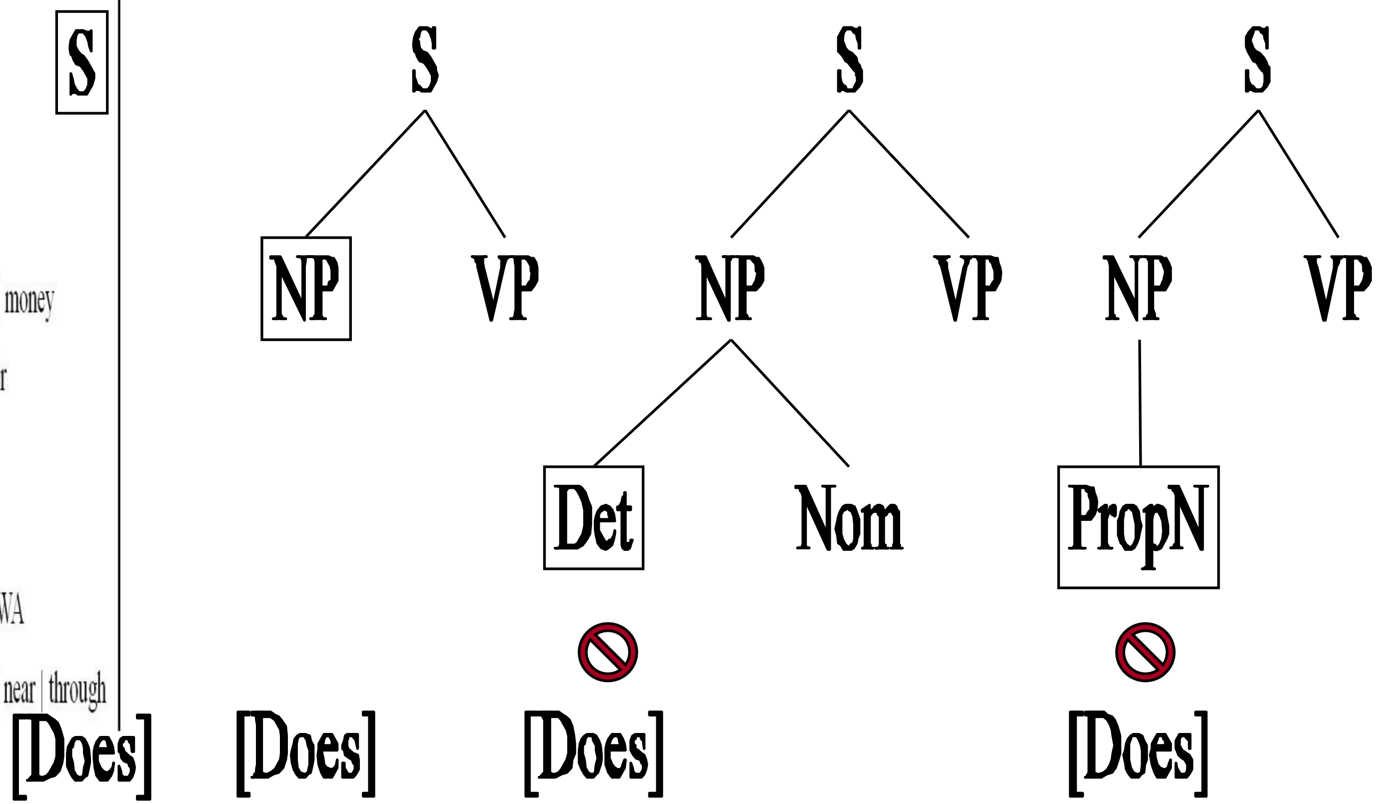
- Practically infeasible to generate all trees in parallel.
- Use depth-first strategy.
- When arriving at a tree that is inconsistent with the input, return to the most recently generated but still unexplored tree.


S ⇒ S ⇒ S ⇒ S
NP VP
NP VP
Det Nom
NP VP
Det Nom
Noun
⇒ S ⇒ S
NP VP
Det Nom Verb NP
Noun
NP VP
Det Nom Verb NP
Noun Det Nom
⇒ S
NP VP
Det Nom Verb NP
Noun Det Nom
Noun

An example: TD, DF, L2R Search

Does this flight include a meal?

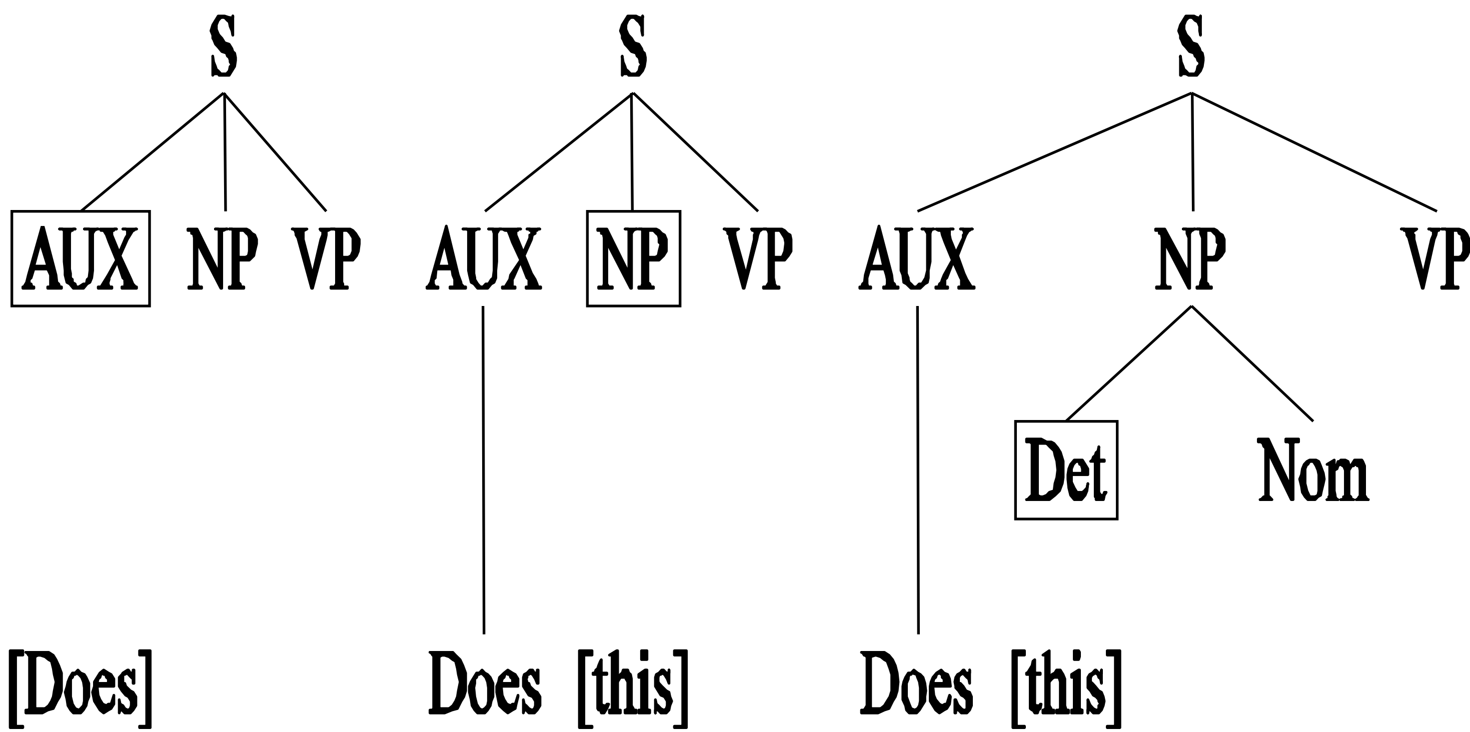
S → NP VP	VP → Verb NP PP
S → Aux NP VP	VP → Verb PP
S → VP	VP → VP PP
NP → Pronoun	PP → Preposition NP
NP → Det NOMINAL	Det → that | this | a
NP → Proper-Noun	Noun → book | flight | meal | money
NOMINAL → Noun	Verb → book | include | prefer
NOMINAL → NOMINAL Noun	Pronoun → I | she | me
NOMINAL → NOMINAL PP	Aux → does
VP → Verb	Proper-Noun → Houston | TWA
VP → Verb NP	Preposition → from | to | on | near | through



Example

S → NP VP	VP → Verb NP PP
S → Aux NP VP	VP → Verb PP
S → VP	VP → VP PP
NP → Pronoun	PP → Preposition NP
NP → Det NOMINAL	Det → that | this |a
NP → Proper-Noun	Noun → book | flight | meal | money
NOMINAL → Noun	Verb → book | include | prefer
NOMINAL → NOMINAL Noun	Pronoun → I | she | me
NOMINAL → NOMINAL PP	Aux → does
VP → Verb	Proper-Noun → Houston | TWA
VP → Verb NP	Preposition → from | to | on | near | through

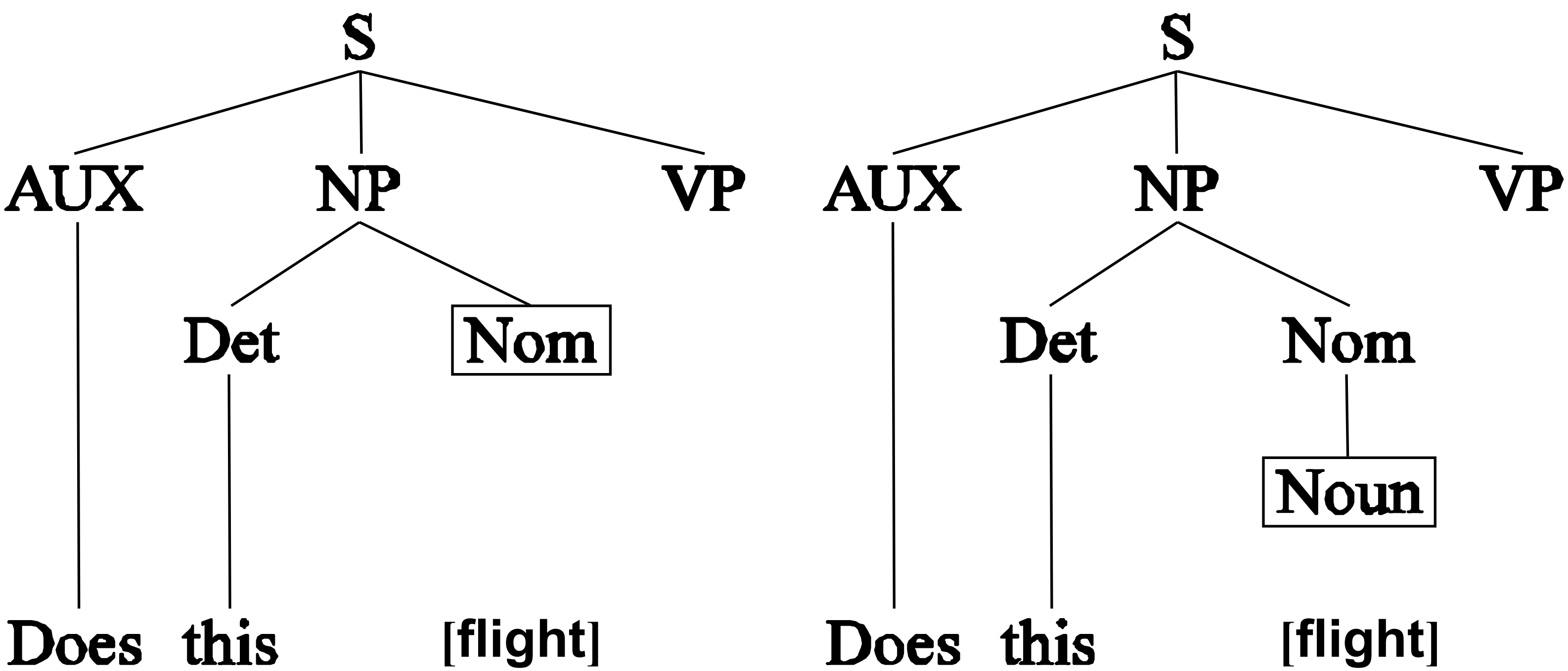
Does this flight include a meal?



Example

Does this flight include a meal?

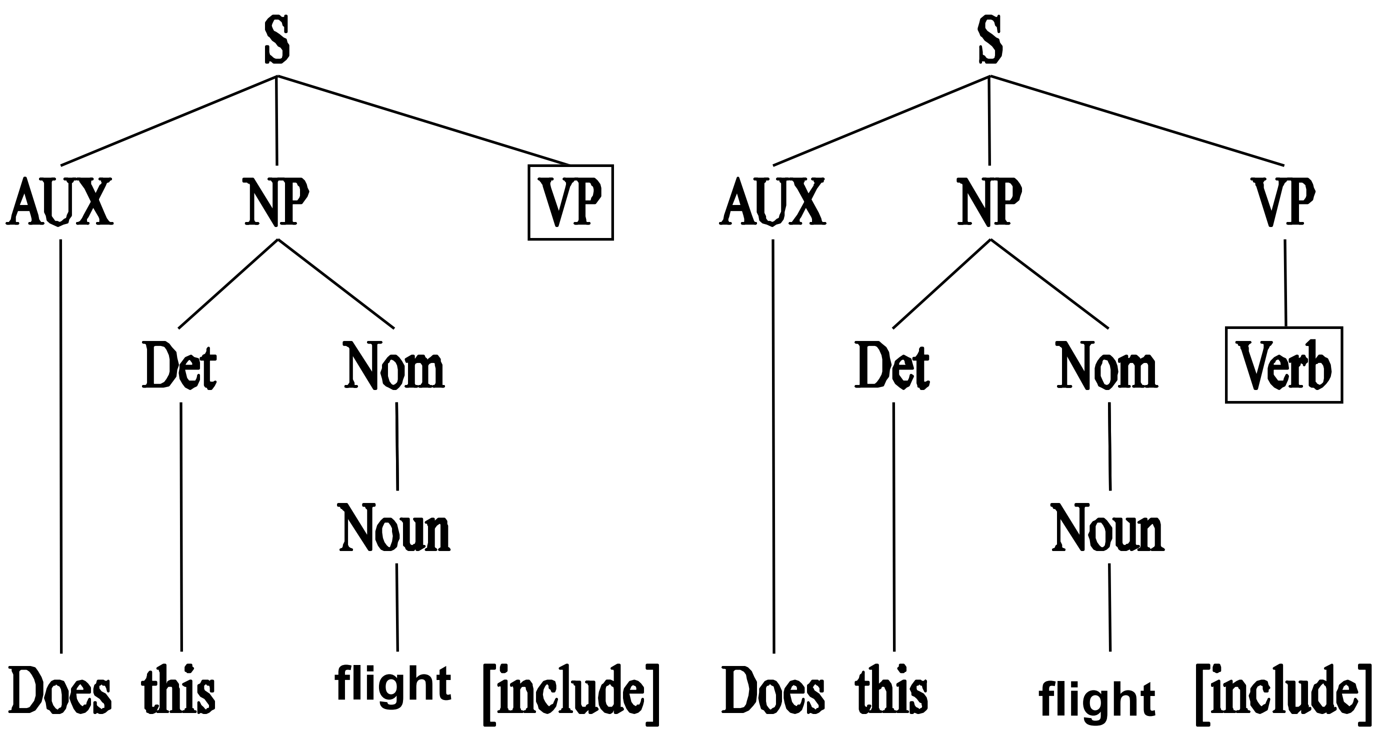
S → NP VP	VP → Verb NP PP
S → Aux NP VP	VP → Verb PP
S → VP	VP → VP PP
NP → Pronoun	PP → Preposition NP
NP → Det NOMINAL	Det → that	this	a
NP → Proper-Noun	Noun → book	flight	meal	money
NOMINAL → Noun	Verb → book	include	prefer
NOMINAL → NOMINAL Noun	Pronoun → I	she	me
NOMINAL → NOMINAL PP	Aux → does
VP → Verb	Proper-Noun → Houston	TWA
VP → Verb NP	Preposition → from	to	on	near	through



Example

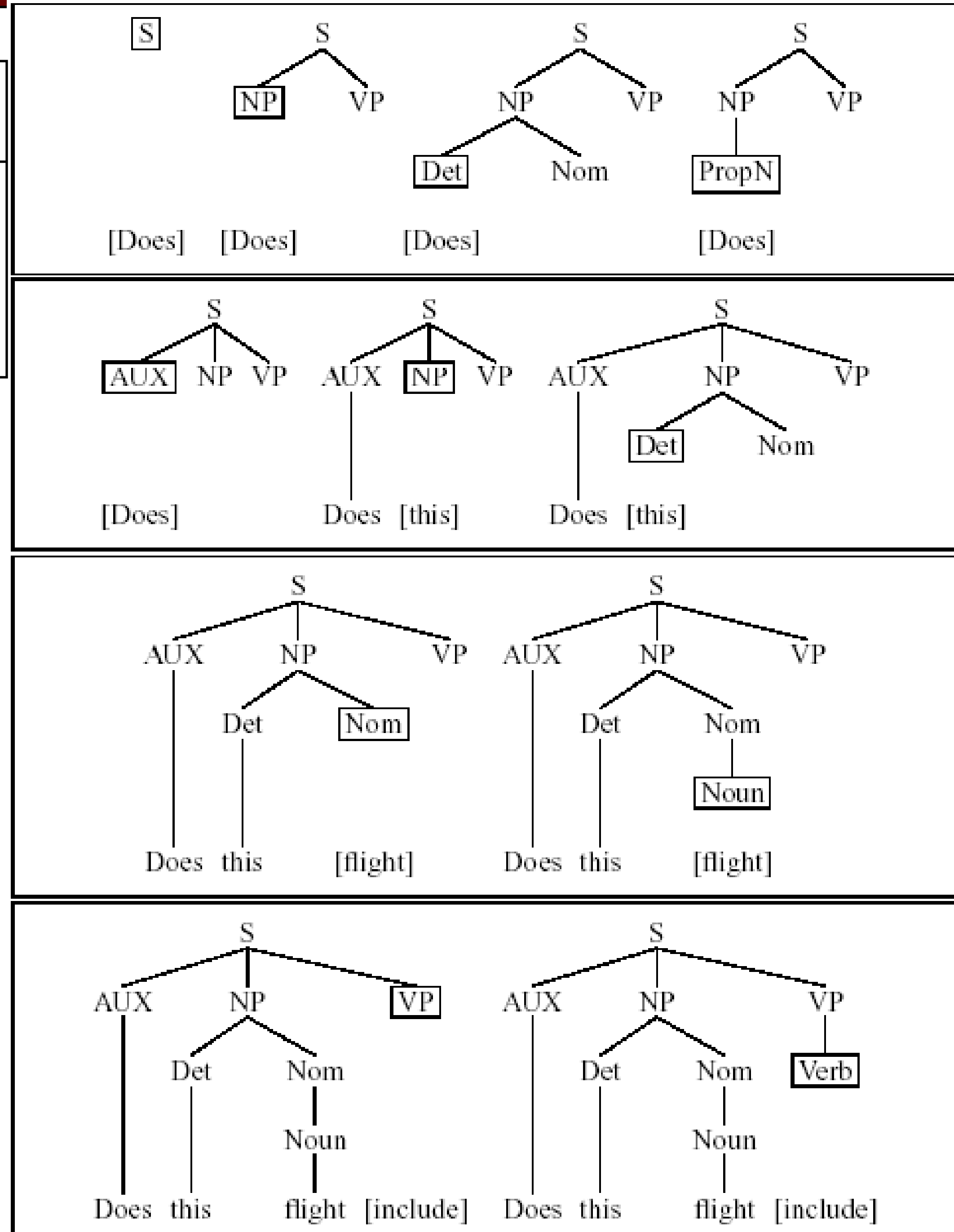
Does this flight include a meal?

S → NP VP	VP → Verb NP PP
S → Aux NP VP	VP → Verb PP
S → VP	VP → VP PP
NP → Pronoun	PP → Preposition NP
NP → Det NOMINAL	Det → that | this |a
NP → Proper-Noun	Noun → book | flight | meal | money
NOMINAL → Noun	Verb → book | include | prefer
NOMINAL → NOMINAL Noun	Pronoun → I | she | me
NOMINAL → NOMINAL PP	Aux → does
VP → Verb	Proper-Noun → Houston | TWA
VP → Verb NP	Preposition → from | to | on | near | through



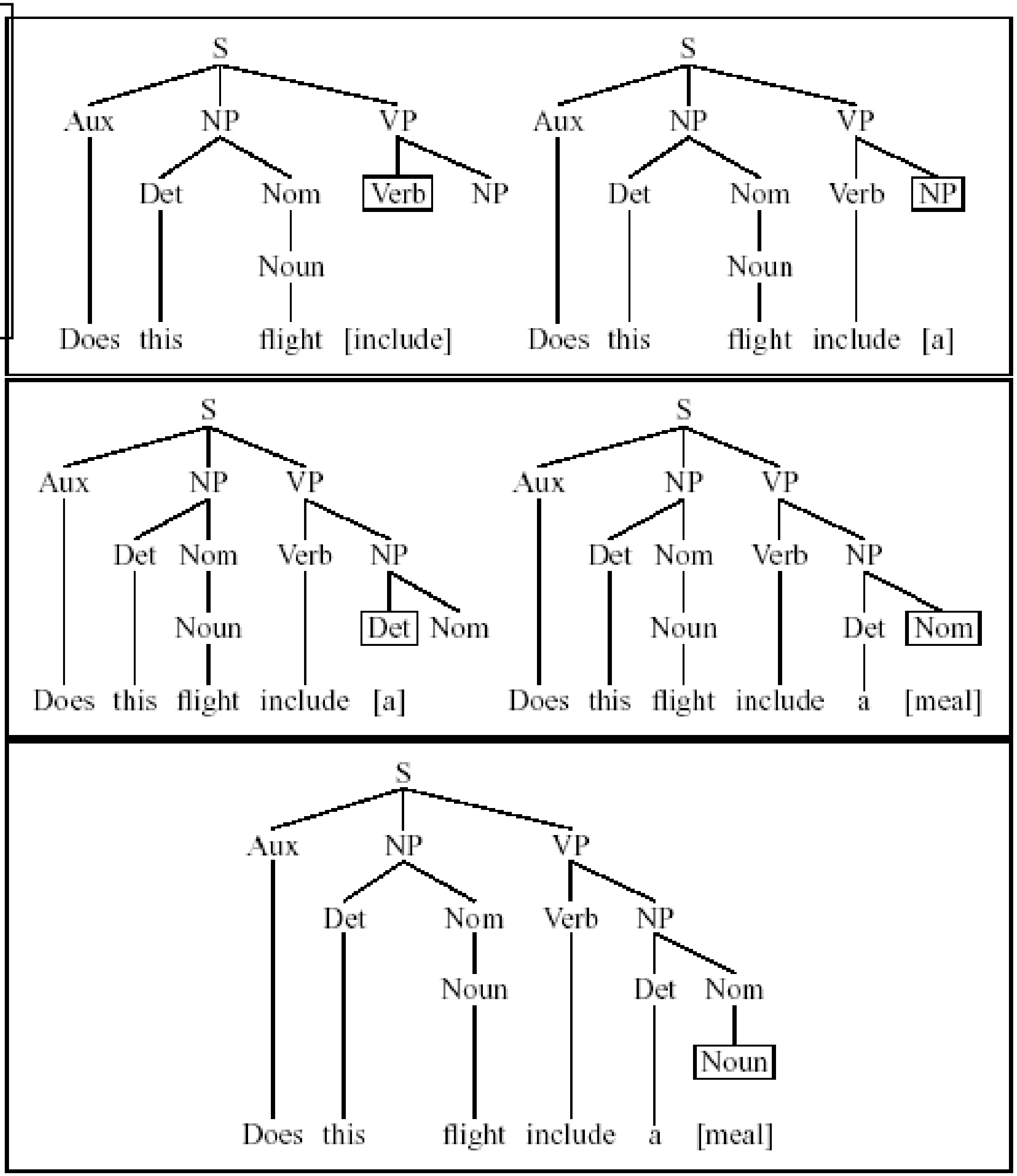
Previous 4 slides

Does this flight include a meal?



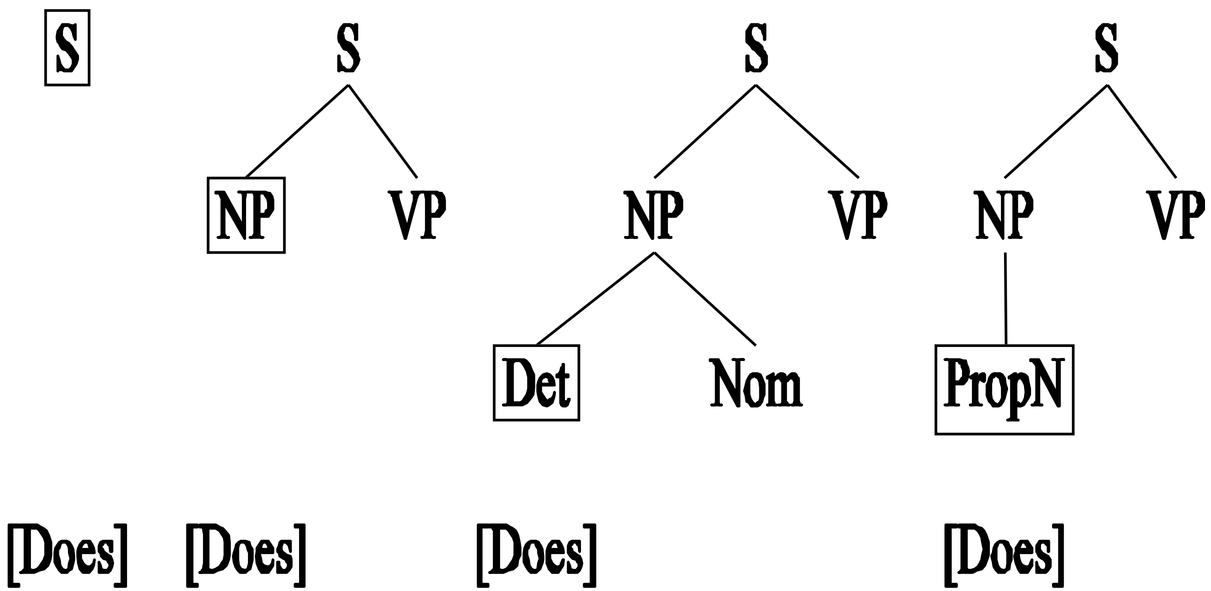
Example: (Cont.)

Does this flight include a meal?



Control

- Does this sequence make any sense?



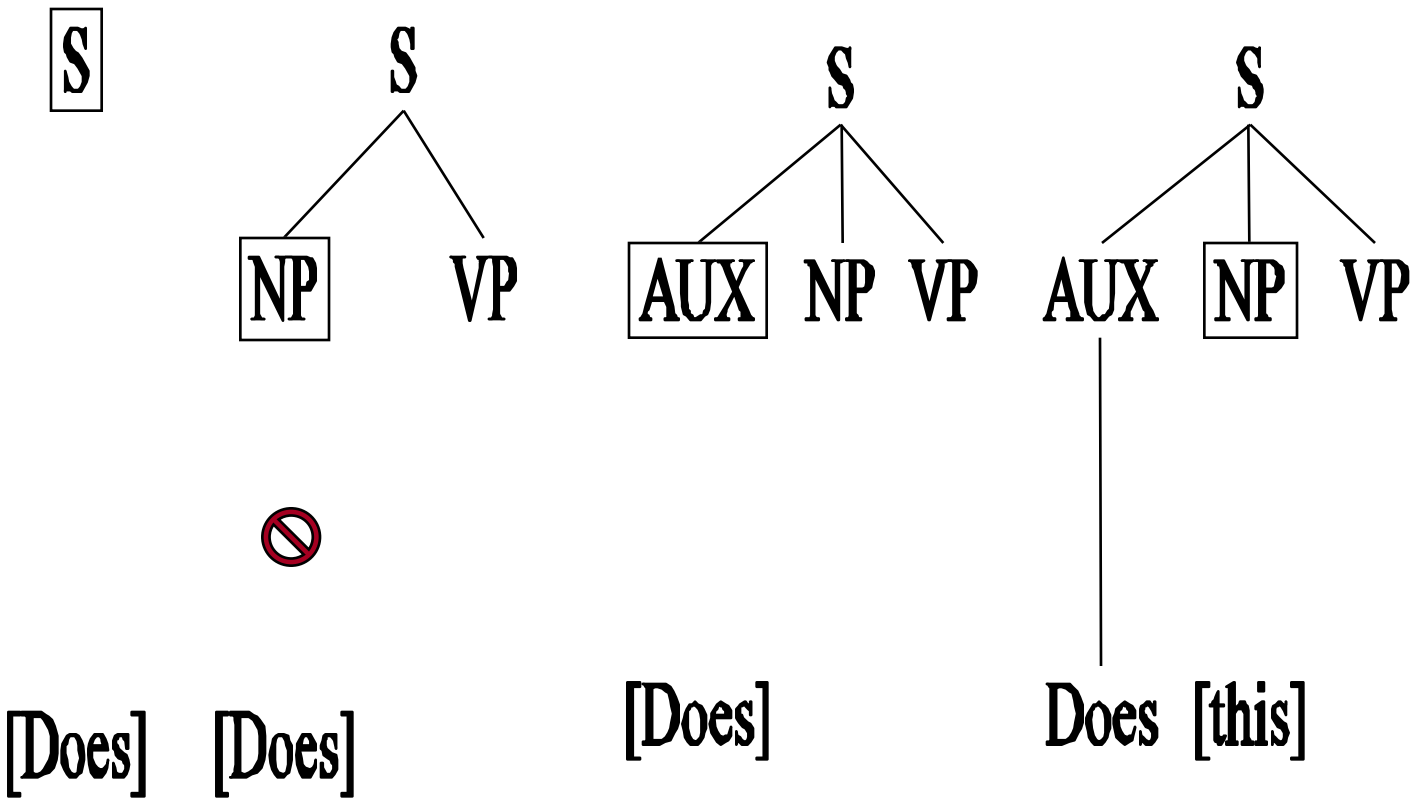
Top-Down and Bottom-Up

- Top-down
  - Only searches for trees that can be answers
  - But suggests trees that are not consistent with the words
- Bottom-up
  - Only forms trees consistent with the words
  - Suggest trees that make no sense globally

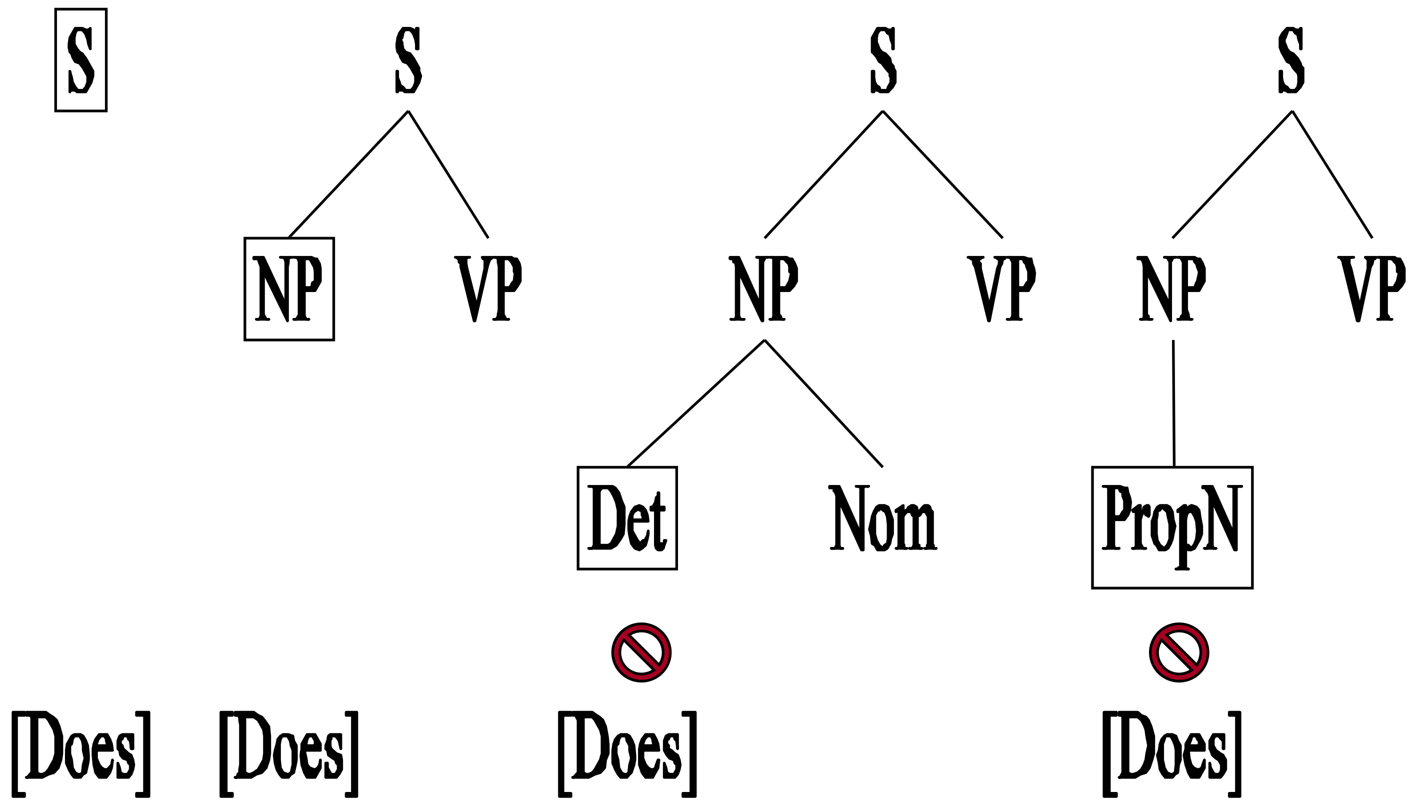
So Combine Them

- How to combine top-down expectations with bottom-up data to get more efficient searches?
- Use one kind as the control and the other as a filter
  - As in top-down parsing with bottom-up filtering

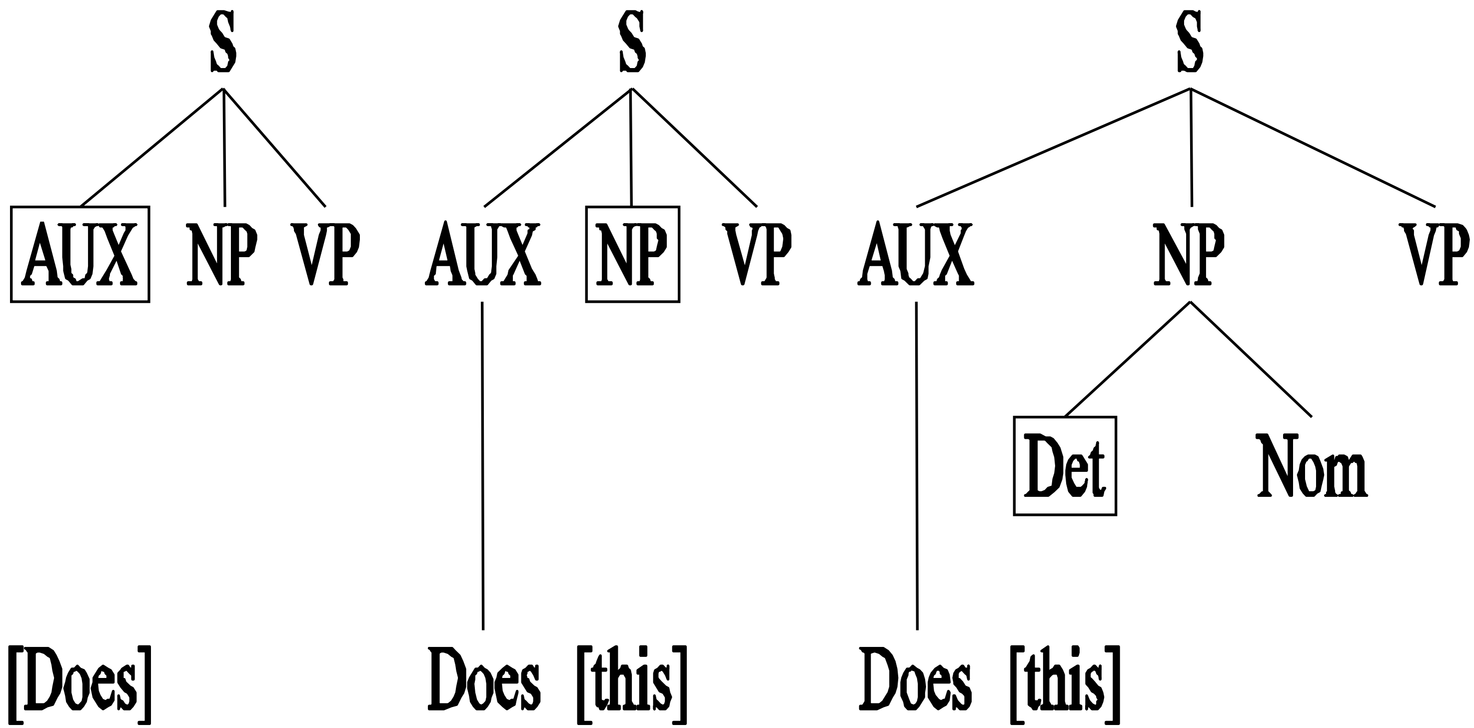
Bottom-Up Filtering



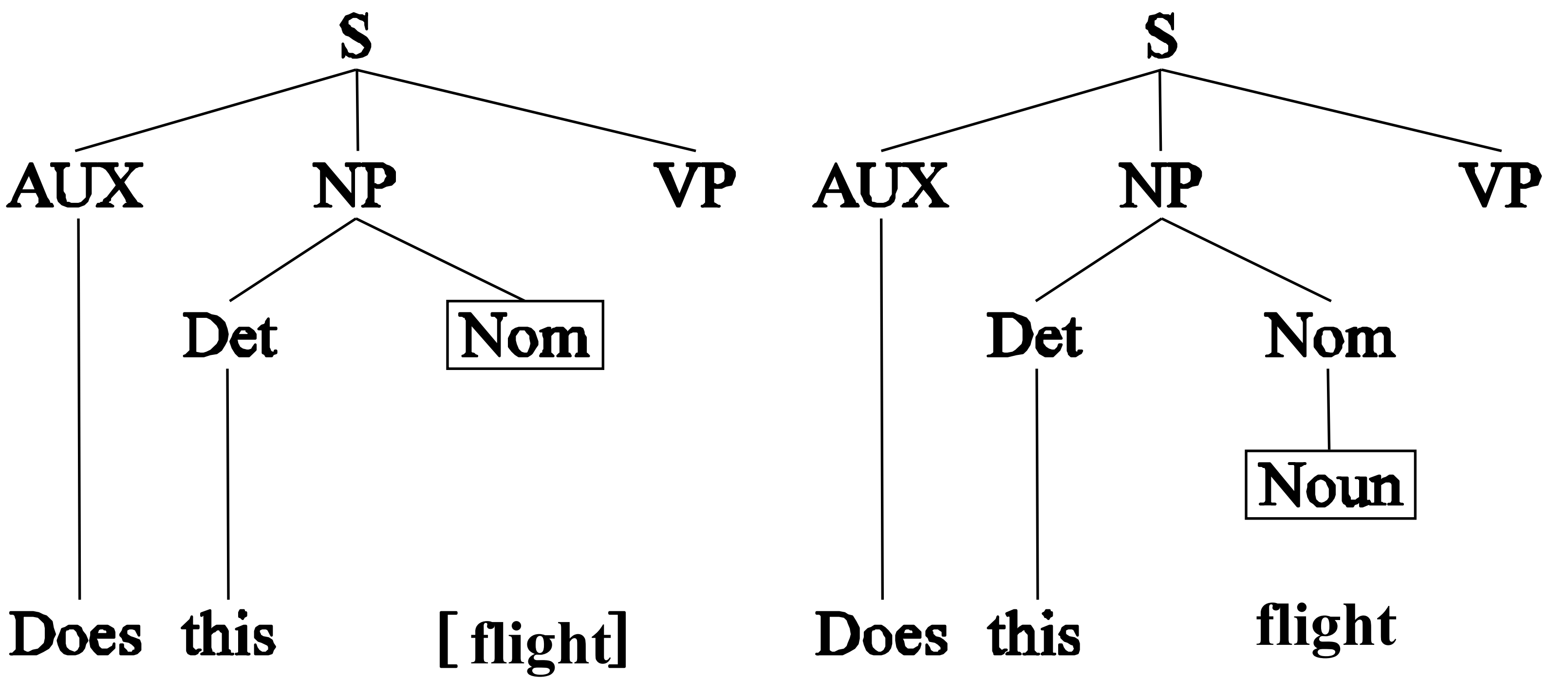
Top-Down, Depth-First, Left-to-Right Search



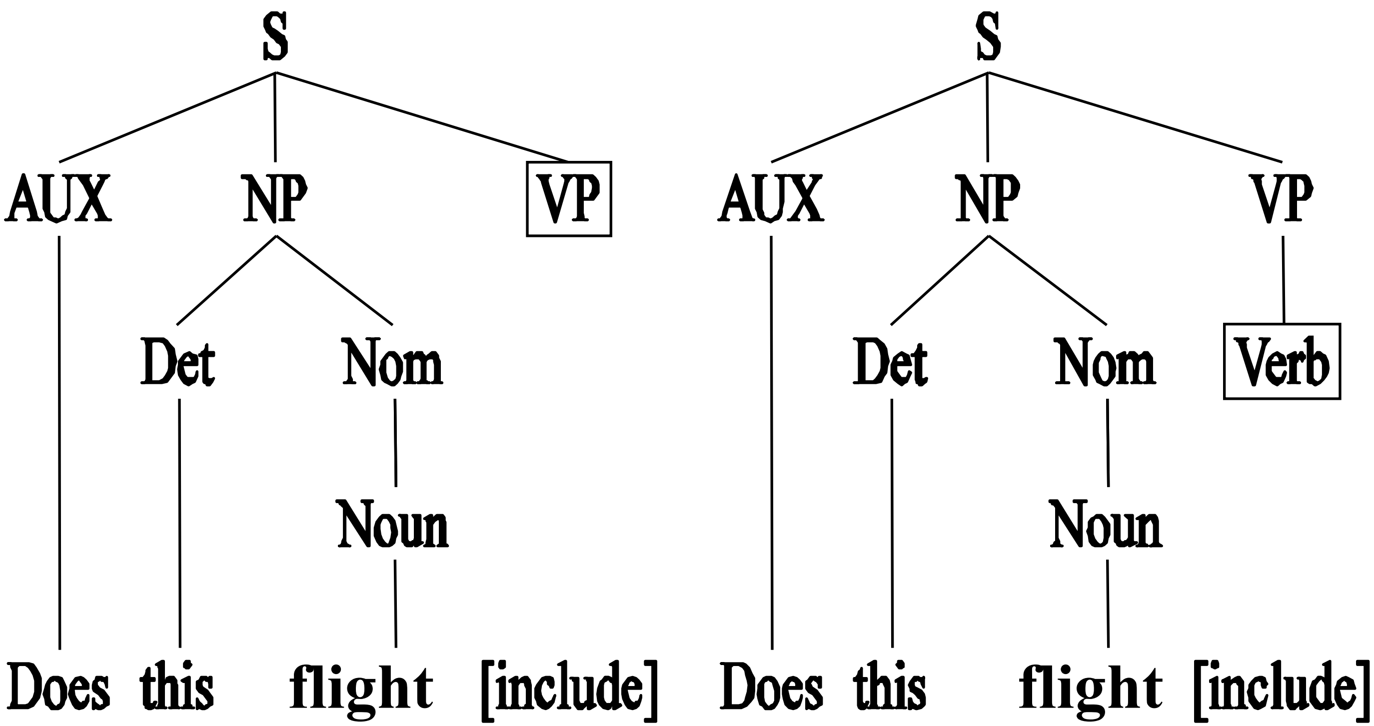
Example


S
AUX NP VP
[Does]
S
AUX NP VP
Does [this]
S
AUX NP VP
Det Nom
Does [this]

Example


S
AUX
NP
VP
Det
Nom
Does
this
[flight]
S
AUX
NP
VP
Det
Nom
Noun
Does
this
flight

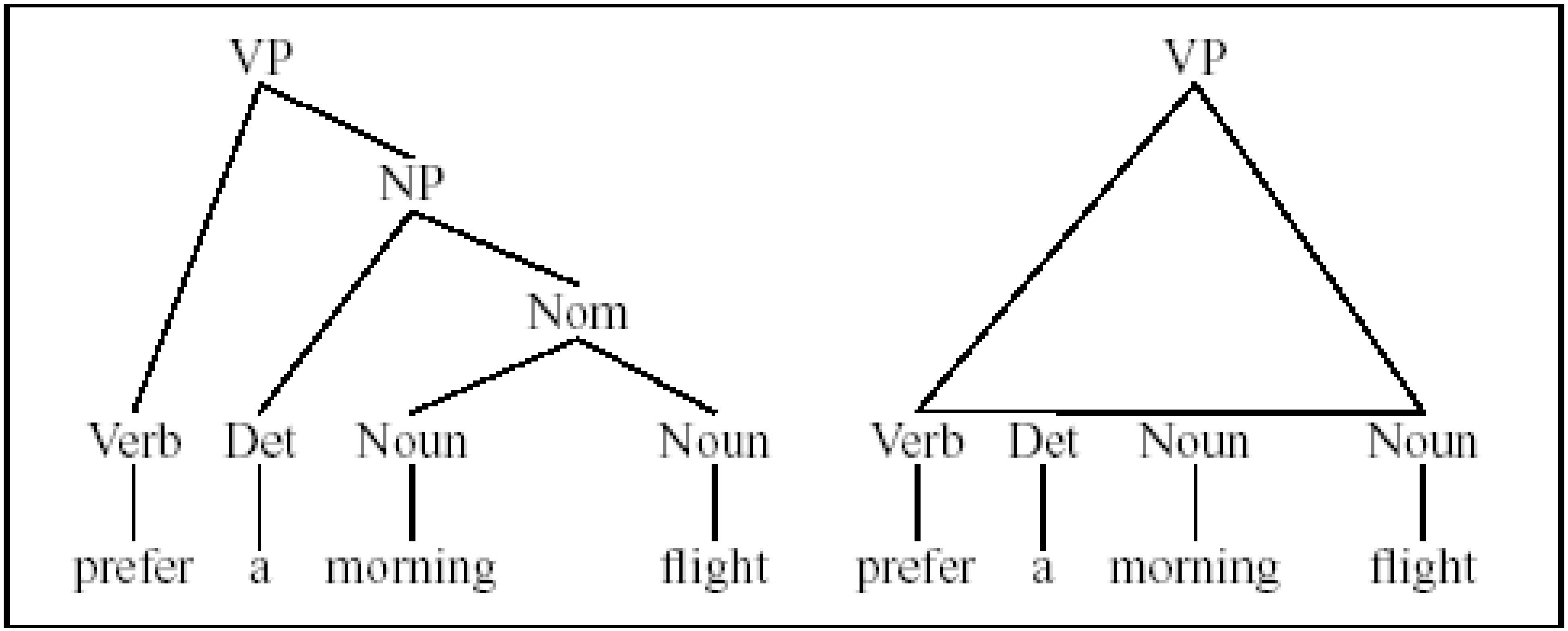
Example


S
AUX
NP
VP
Det
Nom
Noun
Does
this
flight
[include]
S
AUX
NP
VP
Det
Nom
Verb
Noun
Does
this
flight
[include]

Augmenting Top-Down Parsing with Bottom-Up Filtering

- Top-Down, depth-first, Left to Right (L2R) parsing
  - Expands non-terminals along the tree's left edge down to leftmost leaf of tree
  - Moves on to expand down to next leftmost leaf…
  - In a successful parse, current input word will be the first word in derivation of the unexpanded node that the parser is currently processing
  - So….lookahead to left-corner of the tree
    - B is a left-corner of A if A =*=> Bα
    - Build table with left-corners of all non-terminals in grammar and consult before applying rule

Left Corner


VP
NP
Nom
Verb
Det
Noun
Noun
prefer
a
morning
flight
VP
Verb
Det
Noun
Noun
prefer
a
morning
flight

Left-Corner Table for CFG

Category	Left Corners
S	Det, PropN, Aux, V
NP	Det, PropN
Nom	N
VP	V

S → NP VP	VP → Verb NP PP
S → Aux NP VP	VP → Verb PP
S → VP	VP → VP PP
NP → Pronoun	PP → Preposition NP
NP → Det NOMINAL	Det → that | this |a
NP → Proper-Noun	Noun → book | flight | meal | money
NOMINAL → Noun	Verb → book | include | prefer
NOMINAL → NOMINAL Noun	Pronoun → I | she | me
NOMINAL → NOMINAL PP	Aux → does
VP → Verb	Proper-Noun → Houston | TWA
VP → Verb NP	Preposition → from | to | on | near | through

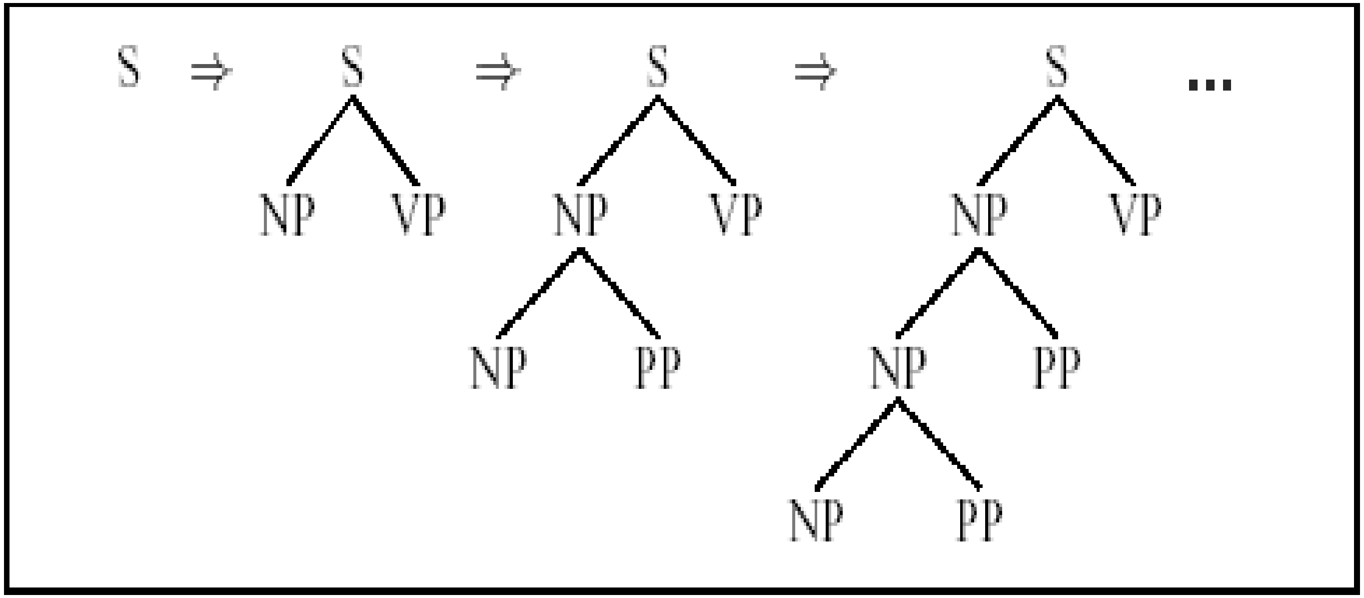
Summing Up

- Parsing is a search problem which may be implemented with many search strategies
- Top-Down vs. Bottom-Up Parsers
  - Both generate too many useless trees
  - Combine the two to avoid over-generation: Top-Down Parsing with Bottom-Up look-ahead
- Left-corner table provides more efficient look-ahead
  - Pre-compute all POS that can serve as the leftmost POS in the derivations of each non-terminal category

Have all the problems been solved?: Left Recursion

- Depth-first search will never terminate if grammar is left recursive (e.g. NP → NP PP)

$$(A \xrightarrow{*} \alpha AB, \alpha \xrightarrow{*} \varepsilon)$$

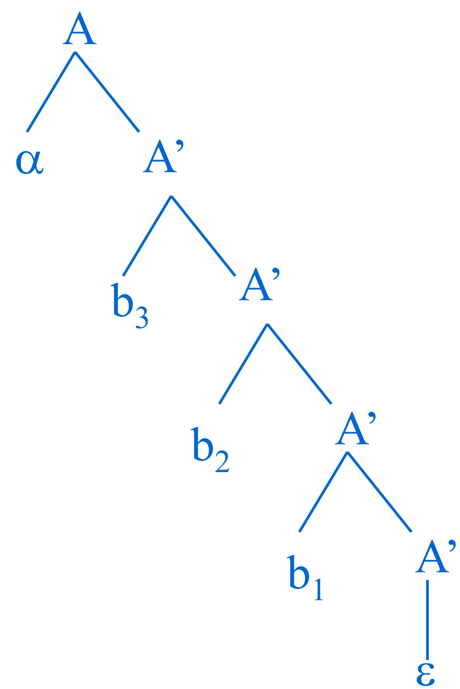


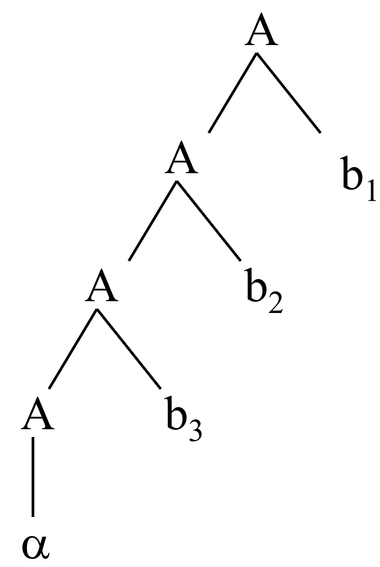
Problems with the basic parser

- Left-recursion: rules of the type:
  $NP \rightarrow NP\ PP$
  solution: rewrite each rule of the form $A \rightarrow A\beta \mid \alpha$ using a new symbol: A' as
- $A \rightarrow \alpha A'$ $A' \rightarrow \beta A' \mid \varepsilon$

Left-recursion Problem

- Rewrite
  - $A \to A\ b_1$
  - $A \to A\ b_2$
  - $A \to A\ b_3$
  - $A \to \alpha$
- as
  - $A \to \alpha\ A'$
  - $A' \to b_1\ A'$
  - $A' \to b_2\ A'$
  - $A' \to b_3\ A'$
  - $A' \to \varepsilon$





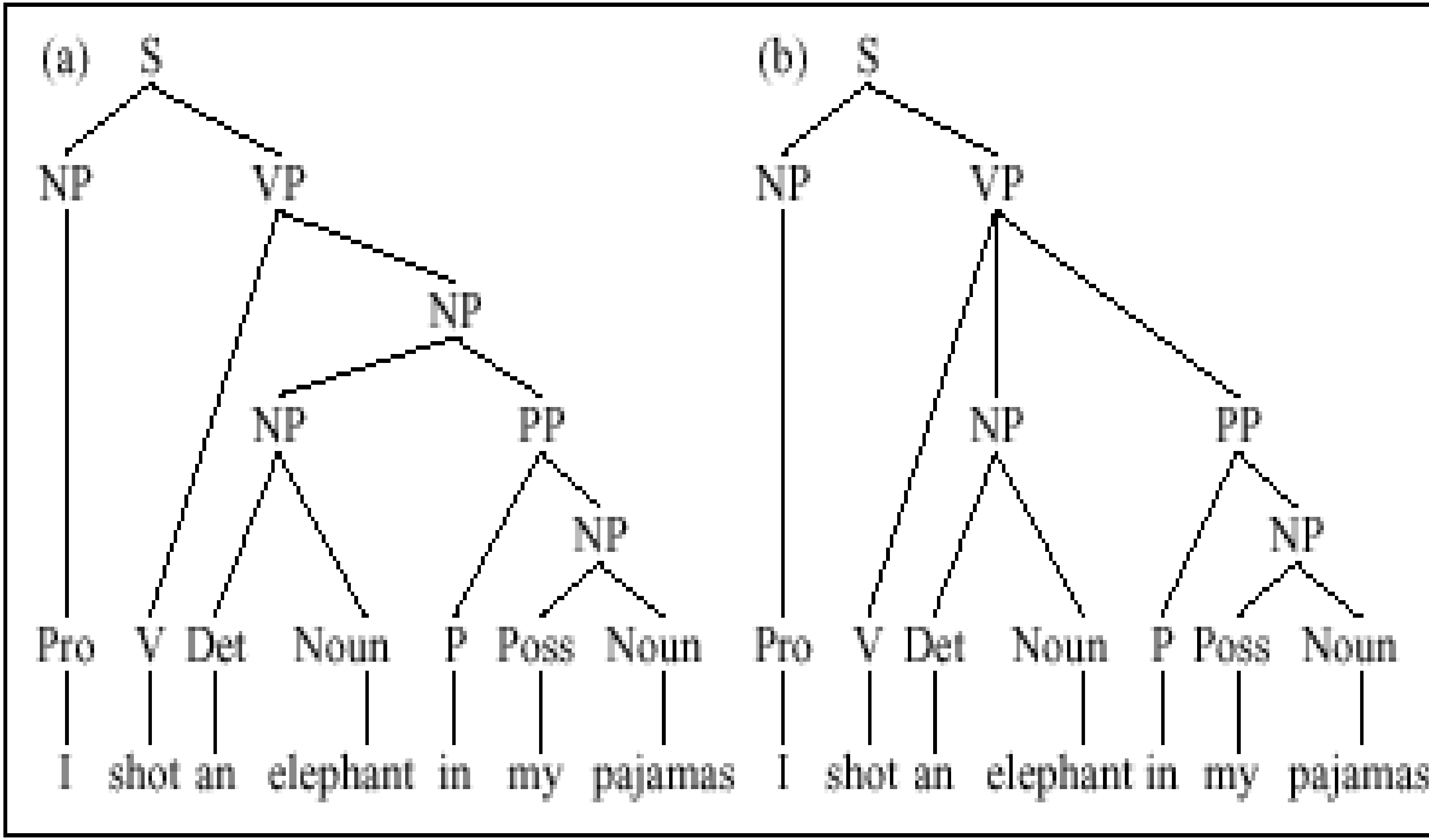
Structural ambiguity:

- Multiple legal structures
  - Attachment (e.g. I saw a man on a hill with a telescope)
  - Coordination (e.g. younger cats and dogs)
  - NP bracketing (e.g. Spanish language teachers)

Ambiguity

- Structural ambiguity occurs when a grammar assigns more than one possible parse trees to a sentence..
- **Attachment ambiguity** is when a particular constituent can be attached to the parse tree in more that one ways. E.g
  - I shot an elephant in my pajamas.
- **Coordination ambiguity** is when there are different sets of phrases that can be joined by a conjunction such as *and.*
  - [old [men and women]] *or* [old men] and [women]
- **Noun phrase bracketing ambiguity**.
  - [Dead [poets' society]] *or* [[Dead poets'] society]

Ambiguity


(a) S
NP VP
NP
NP PP
NP
Pro V Det Noun P Poss Noun
I shot an elephant in my pajamas
(b) S
NP VP
NP PP
NP
Pro V Det Noun P Poss Noun
I shot an elephant in my pajamas

Ambiguity

- Choosing the correct parse of a sentence among the possible parses is a task that requires additional semantic and statistical information. A parser without such information should return all possible parses.
- A sentence may lead to a huge number of parses. Sentences with many PP attachments like

  Show me the meal on Flight UA 386 from San Francisco to Denver.

  lead to an exponational number of parses.

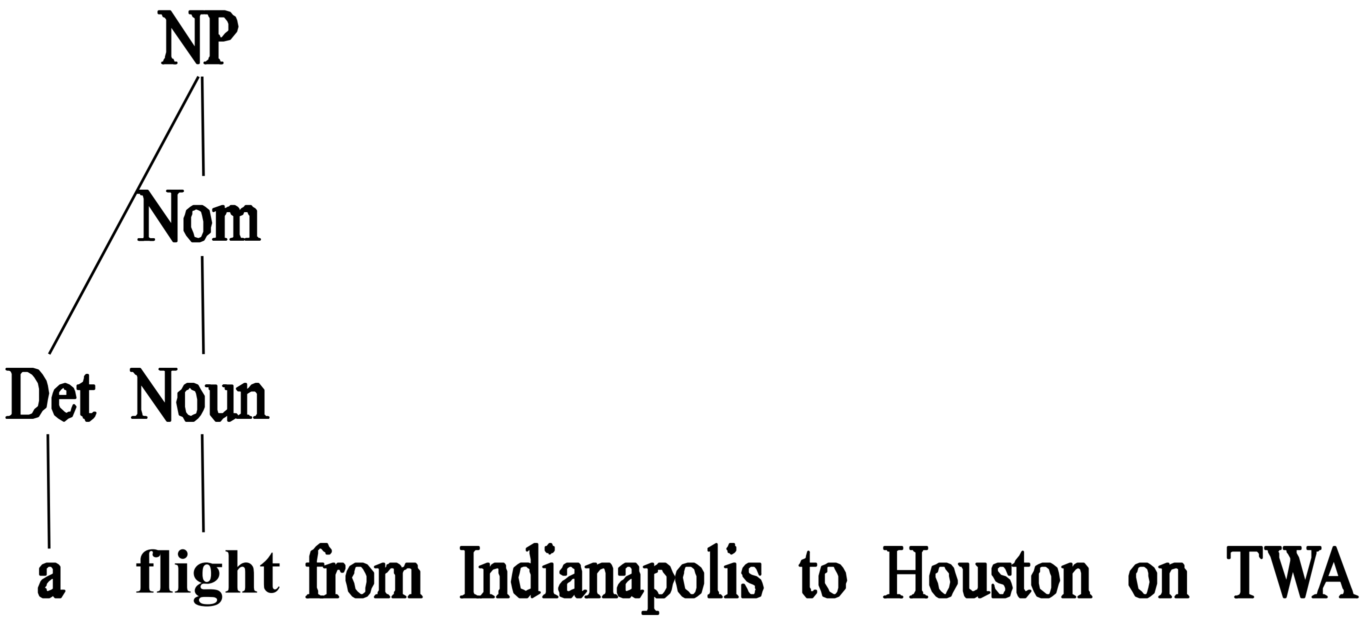
PPs	2	3	4	5	6	7	8
Parses	2	5	14	132	469	1430	4867

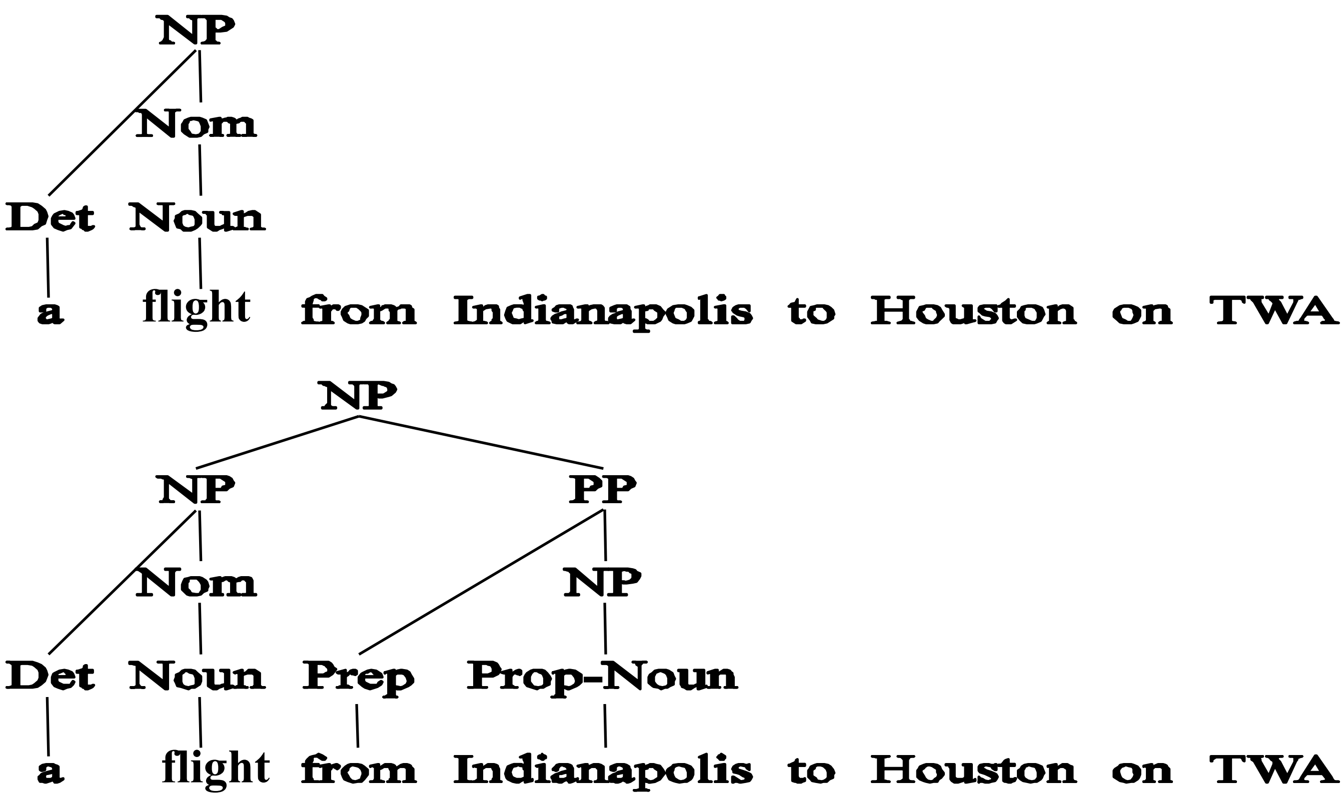
Avoiding Repeated Work

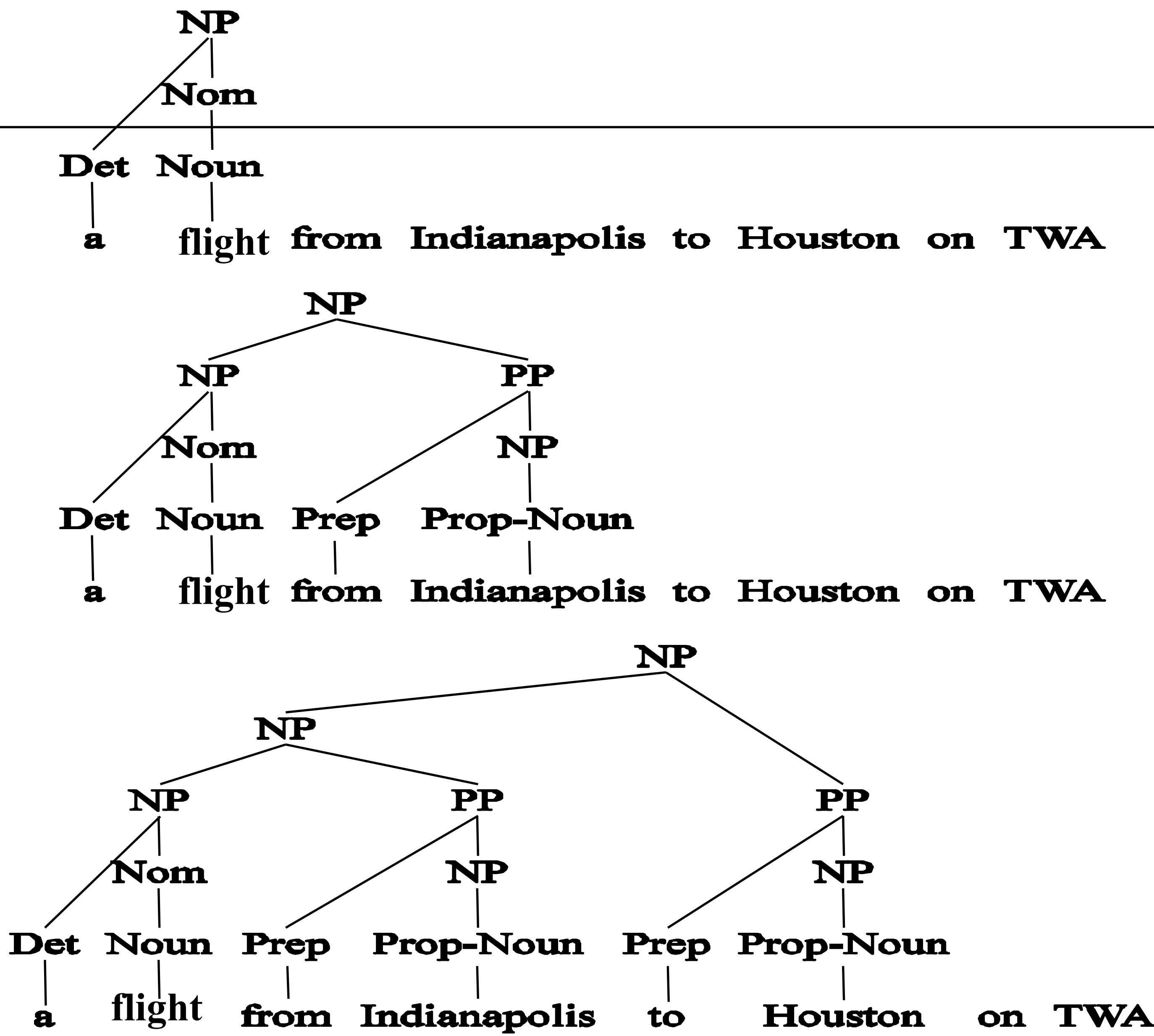
- Parsing is hard, and slow. It's wasteful to redo stuff over and over and over.

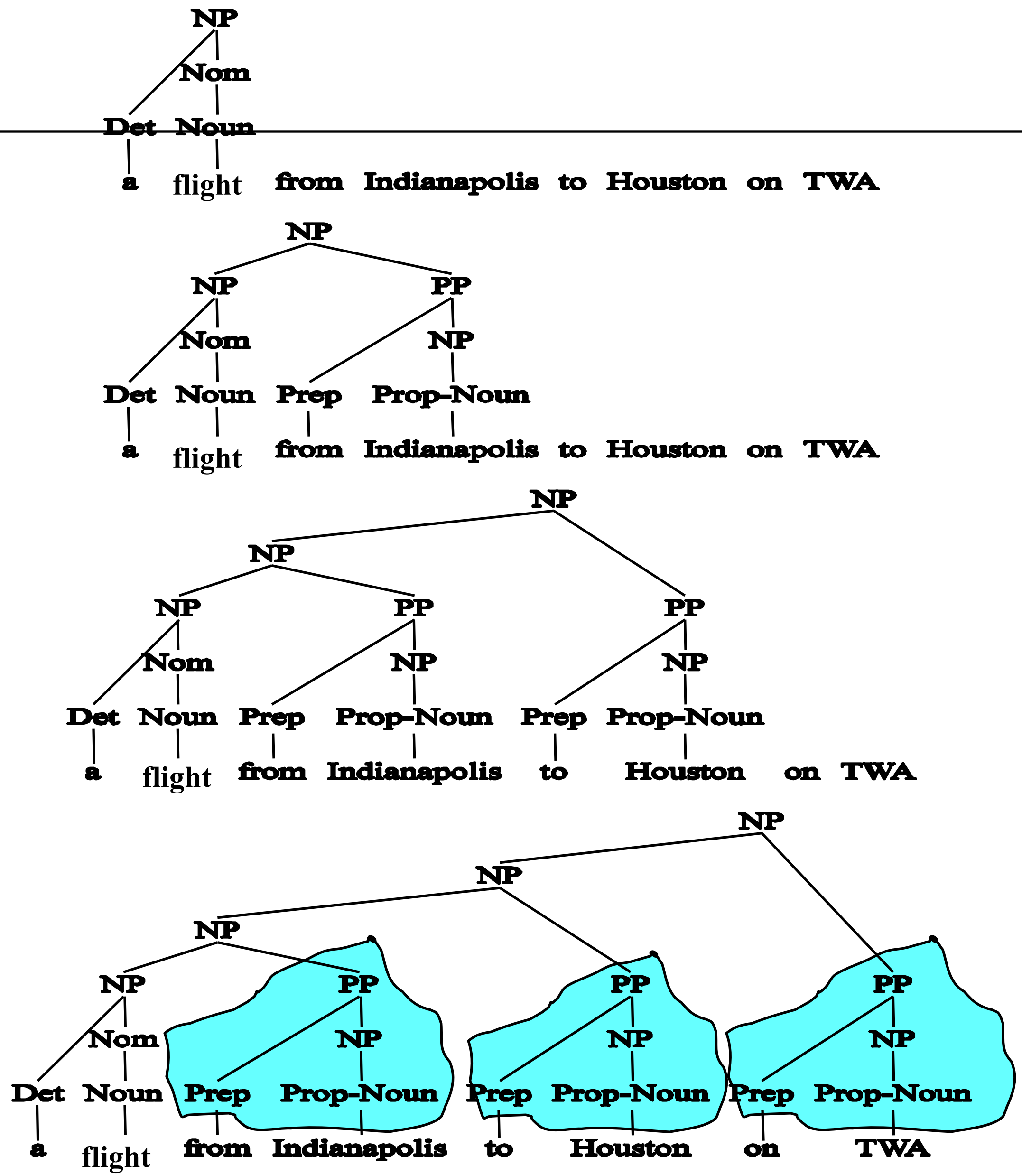
- Consider an attempt to top-down parse the following as an NP

A flight from Indianapolis to Houston on TWA


NP
Nom
Det Noun
a flight from Indianapolis to Houston on TWA


NP
Nom
Det Noun
a flight from Indianapolis to Houston on TWA
NP
NP PP
Nom NP
Det Noun Prep Prop-Noun
a flight from Indianapolis to Houston on TWA


NP
Nom
Det Noun
a flight from Indianapolis to Houston on TWA
NP
NP PP
Nom NP
Det Noun Prep Prop-Noun
a flight from Indianapolis to Houston on TWA
NP
NP
NP PP PP
Nom NP NP
Det Noun Prep Prop-Noun Prep Prop-Noun
a flight from Indianapolis to Houston on TWA


NP
Nom
Det Noun
a flight from Indianapolis to Houston on TWA
NP
NP PP
Nom NP
Det Noun Prep Prop-Noun
a flight from Indianapolis to Houston on TWA
NP
NP
NP PP PP
Nom NP NP
Det Noun Prep Prop-Noun Prep Prop-Noun
a flight from Indianapolis to Houston on TWA
NP
NP
NP
NP PP PP PP
Nom NP NP NP
Det Noun Prep Prop-Noun Prep Prop-Noun Prep Prop-Noun
a flight from Indianapolis to Houston on TWA

Dynamic Programming

- We need a method that fills a table with partial results that
  - Does not do (avoidable) repeated work
  - Does not hang in left-recursion
  - Solves an exponential problem in polynomial time (sort of)

Repeated Parsing of Subtrees

- The parser often builds valid trees for a portion of the input and then discards them during backtracking because they fail to cover all of the input. Later, the parser has to rebuild the same trees again in the search.

Repeated Parsing of Subtrees

- How many times each constituent in the Previous example sentence "A flight from Indianapolis to Houston on TWA" is built.

A flight	4
from Indianapolis	3
Houston	2
on TWA	1
A flight from Indianapolis	3
A flight from Indianapolis to Houston	2
A flight from Indianapolis to Houston on TWA	1

Dynamic Programming

- Create table of solutions to sub-problems (e.g. subtrees) as parse proceeds
- Look up subtrees for each constituent rather than re-parsing
- Since all parses implicitly stored, all available for later disambiguation
- Method:
  - Cocke-Younger-Kasami (CYK) (1960)
  - Graham-Harrison-Ruzzo (GHR) (1980)
  - Earley's (1970) algorithm
    - Next Class

Thank you

- السلام عليكم ورحمة الله