Algorithmic Parsing

Ling 571
Deep Processing Techniques for NLP
January 10, 2011

Roadmap

- Motivation:
  - Recognition and Analysis
- Parsing as Search
  - Search algorithms
  - Top-down parsing
  - Bottom-up parsing
  - Issues: Ambiguity, recursion, garden paths

Parsing

- CFG parsing is the task of assigning proper trees to input strings
  - For any input A and a grammar G, assign (zero or more) parse-trees T that represent its syntactic structure, and
    - Cover all and only the elements of A
    - Have, as root, the start symbol S of G

Parsing

- CFG parsing is the task of assigning proper trees to input strings
  - For any input A and a grammar G, assign (zero or more) parse-trees T that represent its syntactic structure, and
    - Cover all and only the elements of A
    - Have, as root, the start symbol S of G
      - Do not necessarily pick one (or correct) analysis

Parsing

- CFG parsing is the task of assigning proper trees to input strings
  - For any input A and a grammar G, assign (zero or more) parse-trees T that represent its syntactic structure, and
    - Cover all and only the elements of A
    - Have, as root, the start symbol S of G
      - Do not necessarily pick one (or correct) analysis
  - Recognition:
    - Subtask of parsing
    - Given input A and grammar G, is A in the language defined by G or not

Motivation

- Parsing goals:
  - Is this sentence in the language – is it grammatical?
    I prefer United has the earliest flight.
    - FSAs accept the regular languages defined by automaton
    - Parsers accept language defined by CFG

Motivation

- Parsing goals:
  - Is this sentence in the language – is it grammatical?
    I prefer United has the earliest flight.
    - FSAs accept the regular languages defined by automaton
    - Parsers accept language defined by CFG
  - What is the syntactic structure of this sentence?
    - *What airline has the cheapest flight?*
    - *What airport does Southwest fly from near Boston?*
    - Syntactic parse provides framework for semantic analysis
      - What is the subject?

Parsing as Search

- Syntactic parsing searches through possible parse trees to find one or more trees that derive input
- Formally, search problems are defined by:

Parsing as Search

- Syntactic parsing searches through possible parse trees to find one or more trees that derive input
- Formally, search problems are defined by:
  - A start state S,

Parsing as Search

- Syntactic parsing searches through possible parse trees to find one or more trees that derive input
- Formally, search problems are defined by:
  - A start state S,
  - A goal state G,

Parsing as Search

- Syntactic parsing searches through possible parse trees to find one or more trees that derive input
- Formally, search problems are defined by:
  - A start state S,
  - A goal state G,
  - A set of actions, that transition from one state to another
    - Successor function

Parsing as Search

- Syntactic parsing searches through possible parse trees to find one or more trees that derive input
- Formally, search problems are defined by:
  - A start state S,
  - A goal state G,
  - A set of actions, that transition from one state to another
    - Successor function
  - A path cost function

Parsing as Search

- The parsing search problem (one model):
  - Start State S:

Parsing as Search

- The parsing search problem (one model):
  - Start State S: Start Symbol
  - Goal test:

Parsing as Search

- The parsing search problem (one model):
  - Start State S: Start Symbol
  - Goal test:
    - Does parse tree cover all and only input?
  - Successor function:

Parsing as Search

- The parsing search problem (one model):
  - Start State S: Start Symbol
  - Goal test:
    - Does parse tree cover all and only input?
  - Successor function:
    - Expand a non-terminal using production in grammar where non-terminal is LHS of grammar

Parsing as Search

- The parsing search problem (one model):
  - Start State S: Start Symbol
  - Goal test:
    - Does parse tree cover all and only input?
  - Successor function:
    - Expand a non-terminal using production in grammar where non-terminal is LHS of grammar
  - Path cost:
    - We'll ignore here

Parsing as Search

- Node:

Parsing as Search

- Node:
  - Partial solution to search problem:
    - Partial parse
- Search start node:
  - Initial state:

Parsing as Search

- Node:
  - Partial solution to search problem:
    - Partial parse
- Search start node:
  - Initial state:
    - Input string
    - Start symbol of CFG
- Goal node:

Parsing as Search

- Node:
  - Partial solution to search problem:
    - Partial parse
- Search start node:
  - Initial state:
    - Input string
    - Start symbol of CFG
- Goal node:
  - Full parse tree: covering all and only input, rooted at S

Search Algorithms

- Many search algorithms
  - Depth first

Search Algorithms

- Many search algorithms
  - Depth first
    - Keep expanding non-terminal until reach words
      - If no more expansions, back up
  - Breadth first

Search Algorithms

- Many search algorithms
  - Depth first
    - Keep expanding non-terminal until reach words
      - If no more expansions, back up
  - Breadth first
    - Consider all parses with a single non-terminal expanded
      - Then all with two expanded and so
  - Other alternatives if have associated path costs

Parse Search Strategies

- Two constraints on parsing:
  - Must start with the start symbol
  - Must cover exactly the input string

- Correspond to main parsing search strategies
  - Top-down search (Goal-directed search)

  - Bottom-up search (Data-driven search)

A Grammar

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Top-down Search

- All valid parse trees must start with start symbol
  - Begin search with productions with S on LHS
    - E.g., S -> NP VP

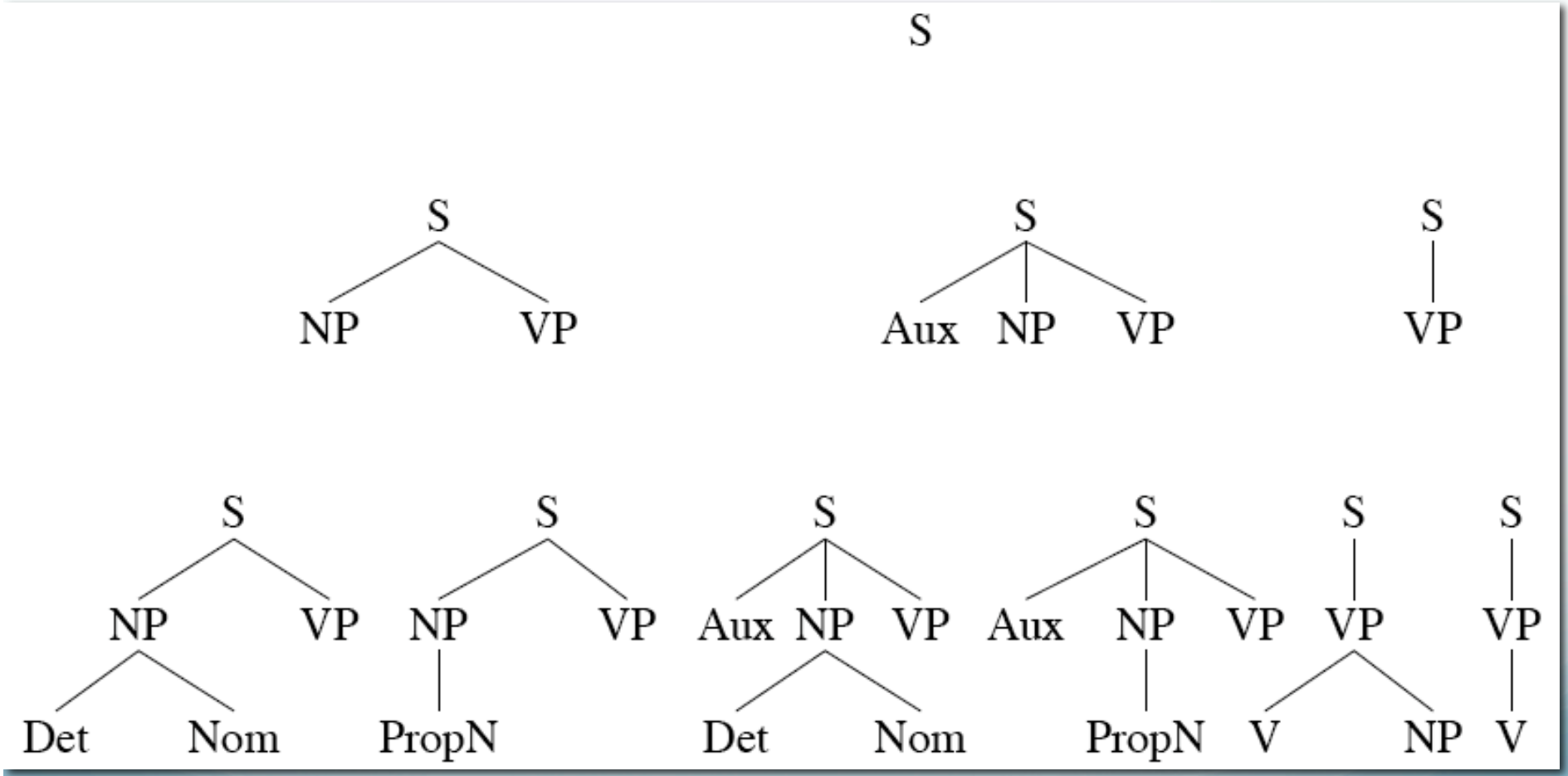
Top-down Search

- All valid parse trees must start with start symbol
  - Begin search with productions with S on LHS
    - E.g., S -> NP VP
  - Successively expand non-terminals
    - E.g., NP – Det Nominal; VP -> V NP

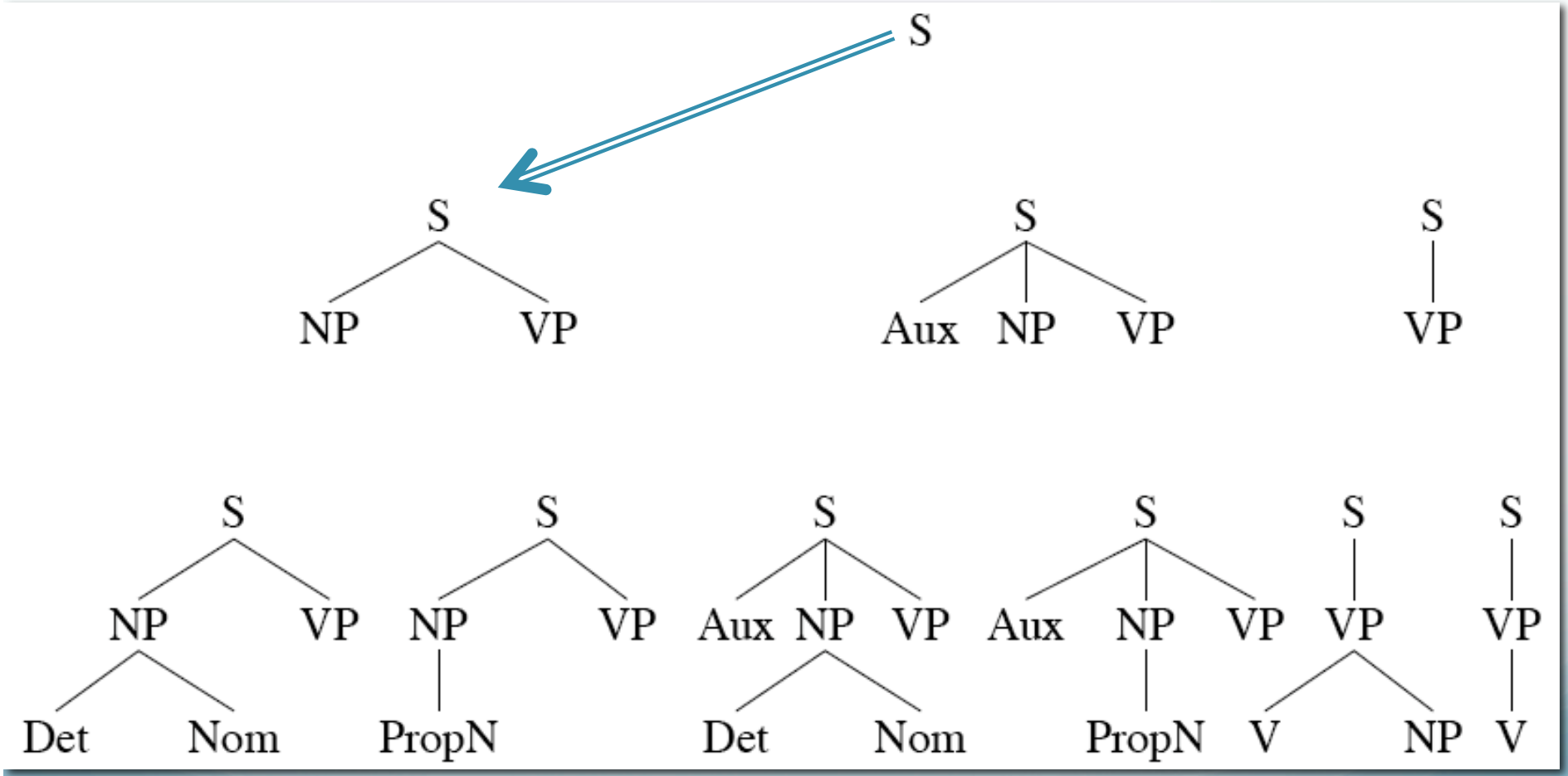
Top-down Search

- All valid parse trees must start with start symbol
  - Begin search with productions with S on LHS
    - E.g., S -> NP VP
  - Successively expand non-terminals
    - E.g., NP – Det Nominal; VP -> V NP
  - Terminate when all leaves are terminals
    - *Book that flight*

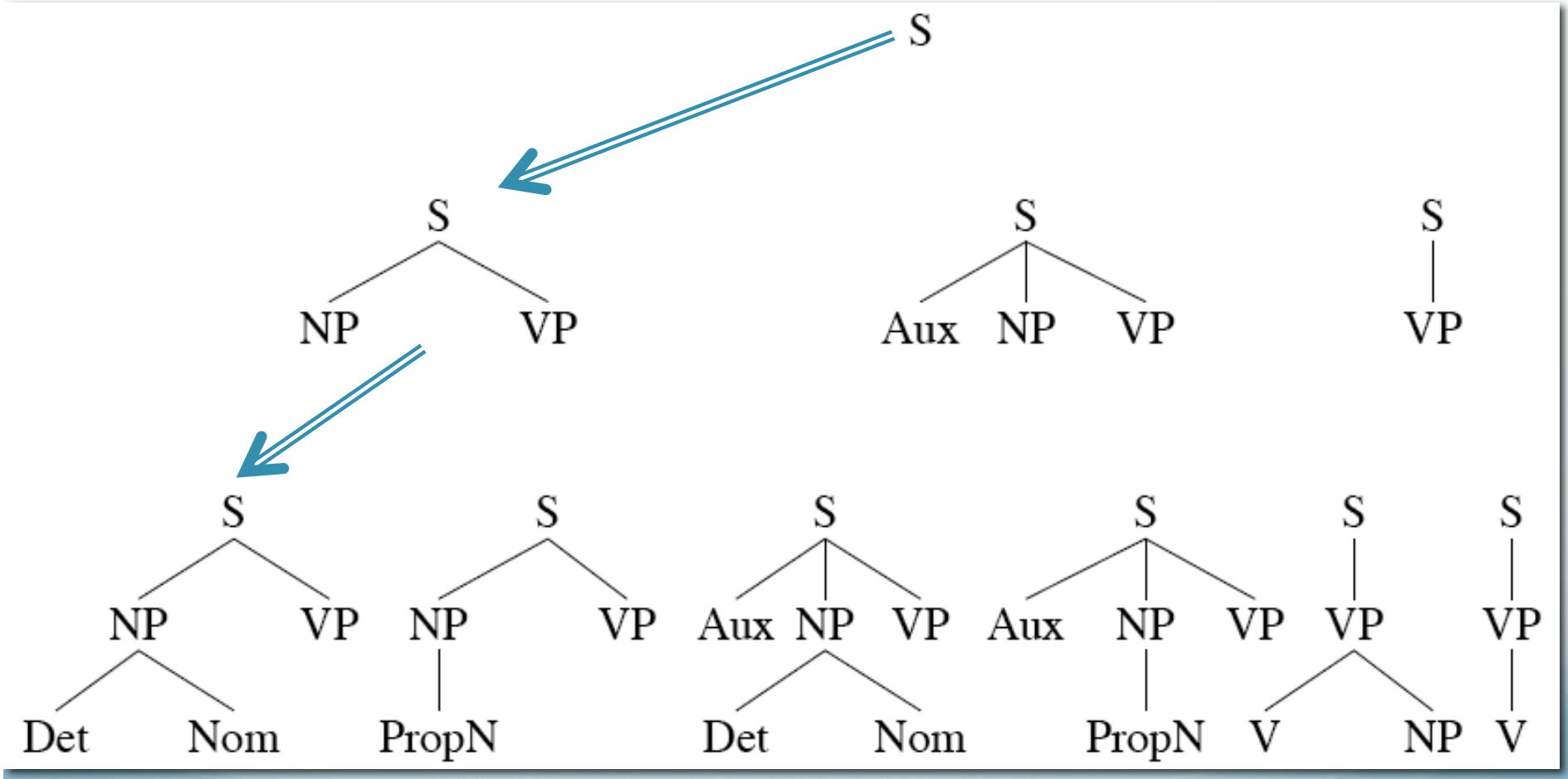
Top-down Search



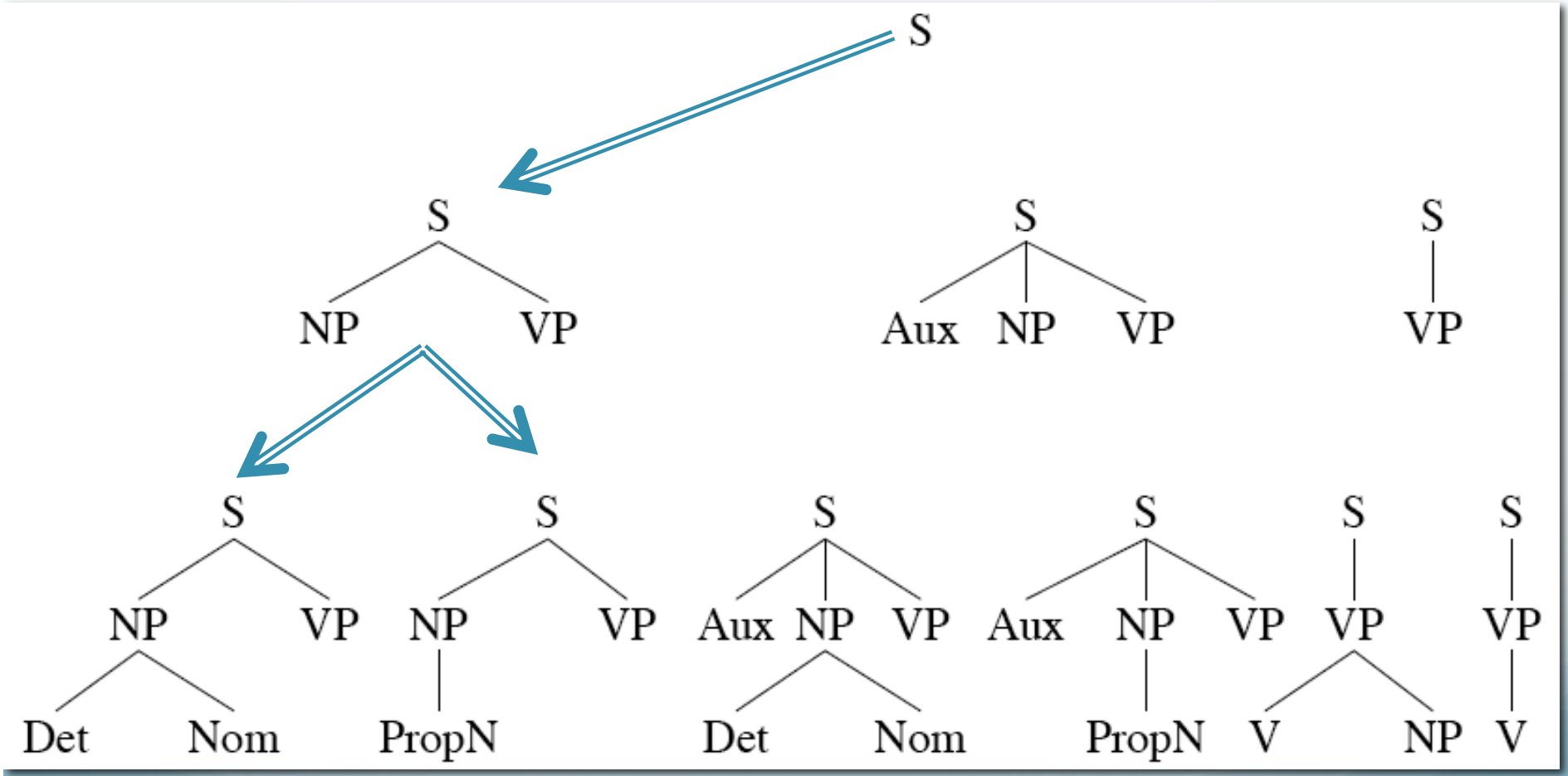
Depth-first Search



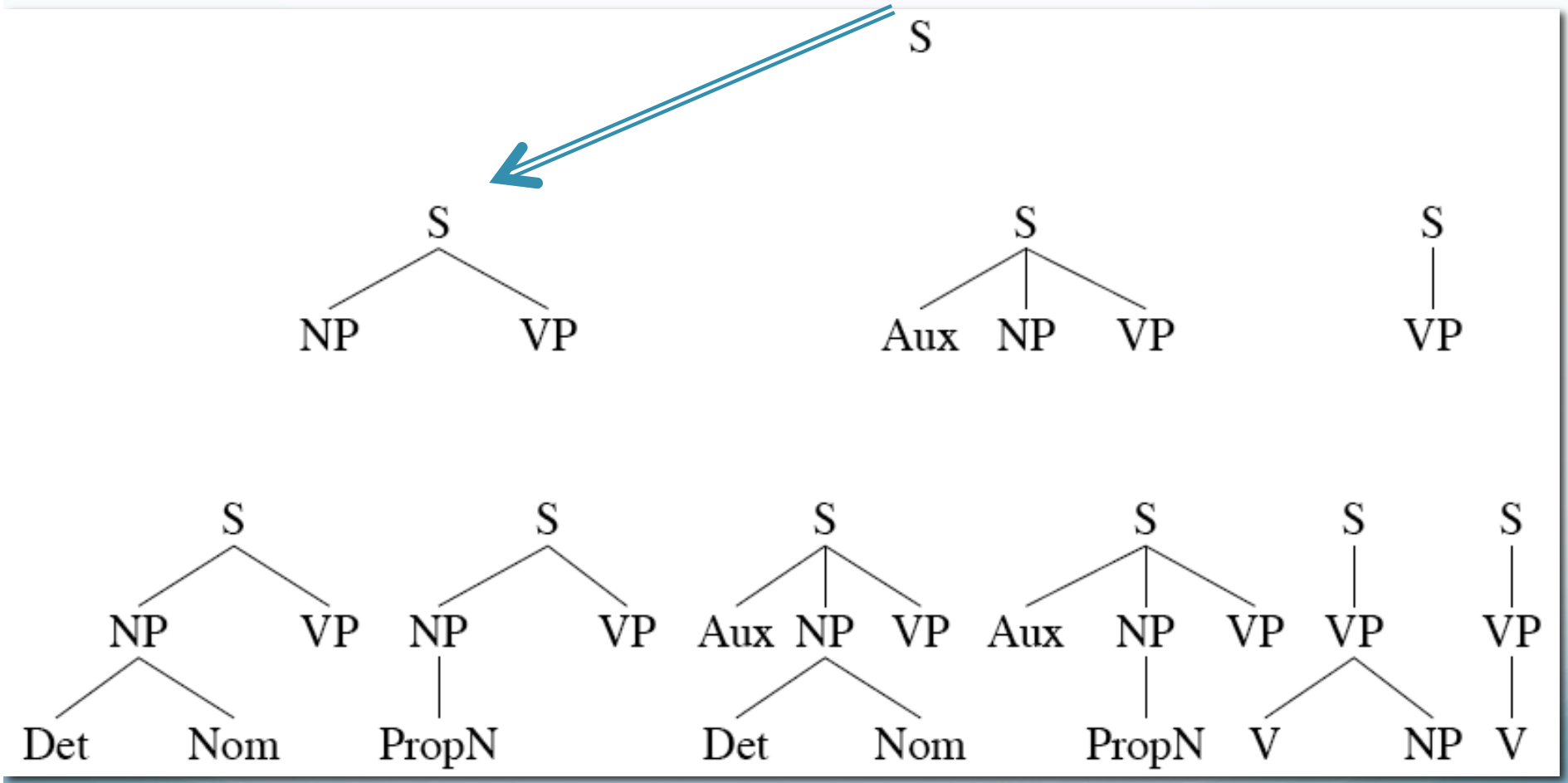
Depth-first Search



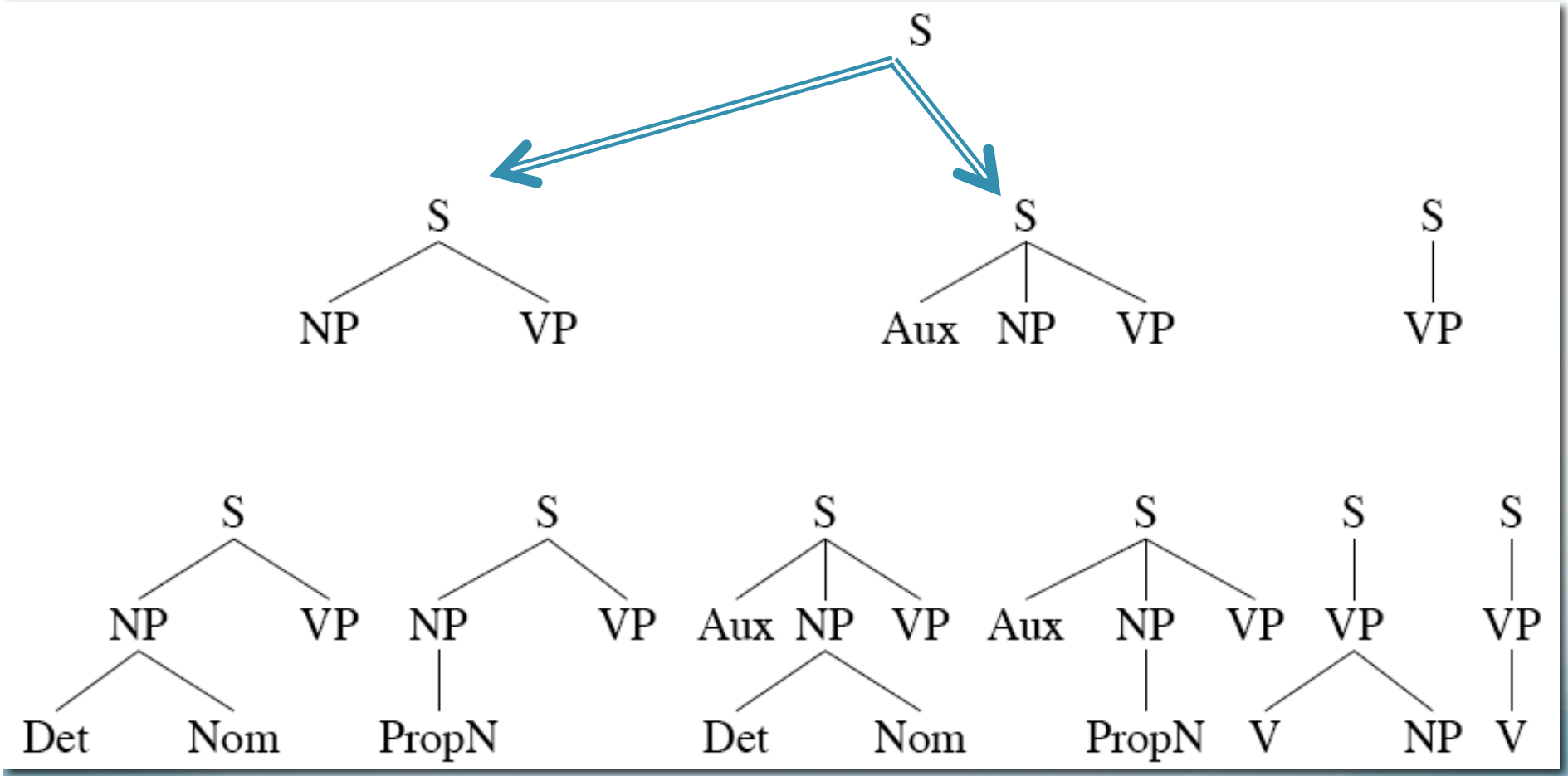
Depth-first Search



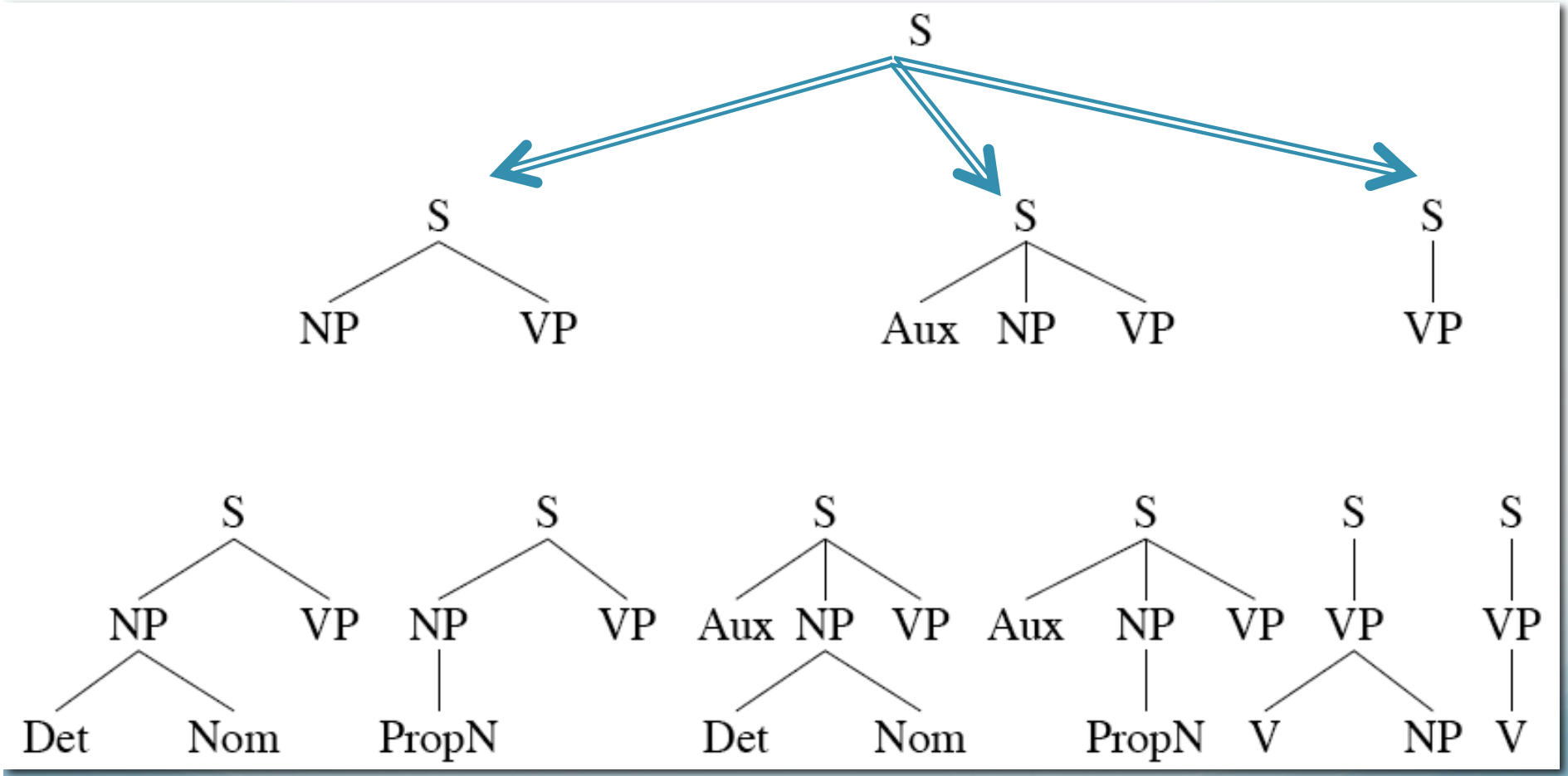
Breadth-first Search



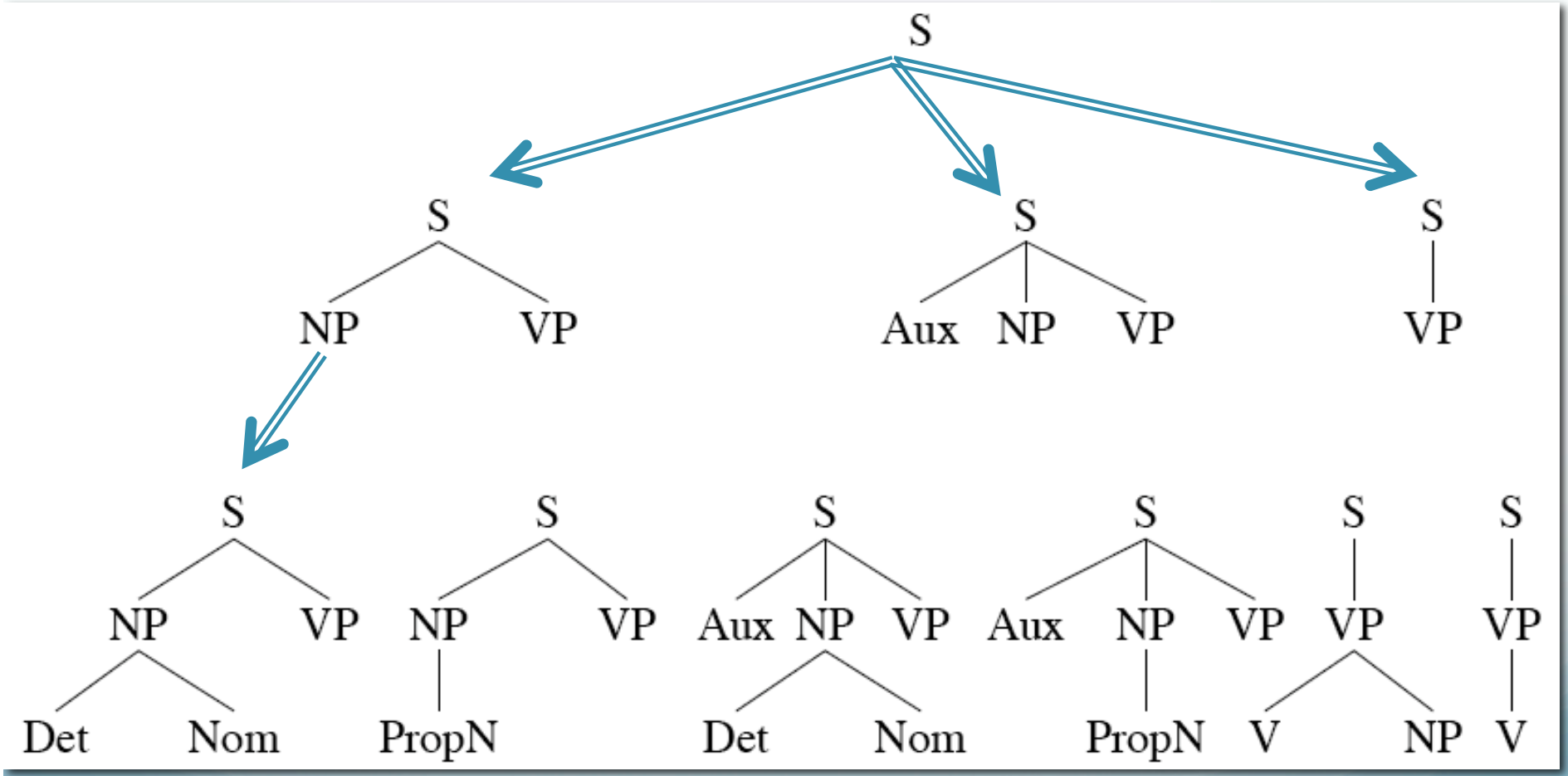
Breadth-first Search



Breadth-first Search



Breadth-first Search



Pros and Cons of Top-down Parsing

- Pros:

Pros and Cons of Top-down Parsing

- Pros:
  - Doesn't explore trees not rooted at S

Pros and Cons of Top-down Parsing

- Pros:
  - Doesn't explore trees not rooted at S
  - Doesn't explore subtrees that don't fit valid trees

Pros and Cons of Top-down Parsing

- Pros:
  - Doesn't explore trees not rooted at S
  - Doesn't explore subtrees that don't fit valid trees
- Cons:
  - Produces trees that may not match input

Pros and Cons of Top-down Parsing

- Pros:
  - Doesn't explore trees not rooted at S
  - Doesn't explore subtrees that don't fit valid trees
- Cons:
  - Produces trees that may not match input
  - May not terminate in presence of recursive rules

Pros and Cons of Top-down Parsing

- Pros:
  - Doesn't explore trees not rooted at S
  - Doesn't explore subtrees that don't fit valid trees
- Cons:
  - Produces trees that may not match input
  - May not terminate in presence of recursive rules
  - May rederive subtrees as part of search

Bottom-Up Parsing

- Try to find all trees that span the input
  - Start with input string
    - Book that flight.

Bottom-Up Parsing

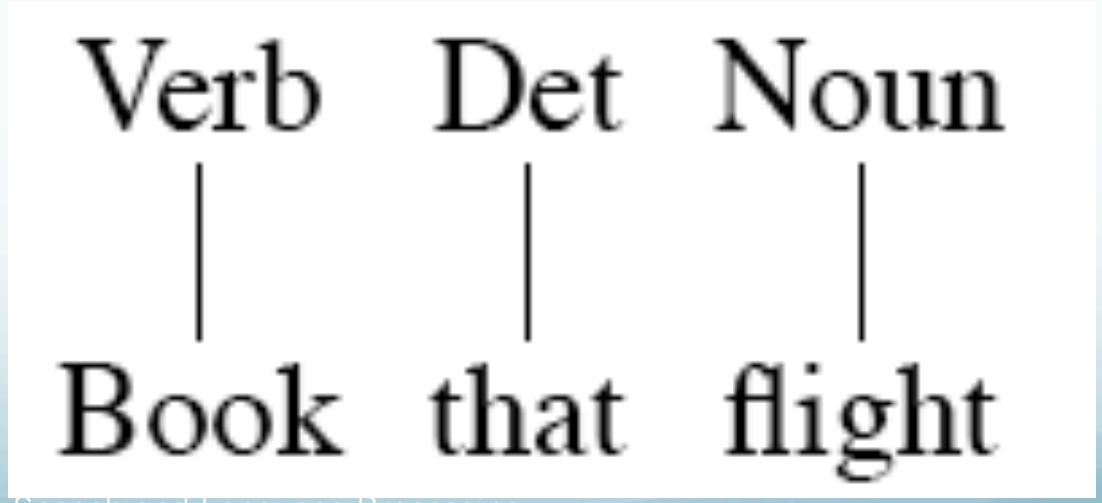
- Try to find all trees that span the input
  - Start with input string
    - Book that flight.
  - Use all productions with current subtree(s) on RHS
    - E.g., N -> Book; V -> Book

Bottom-Up Parsing

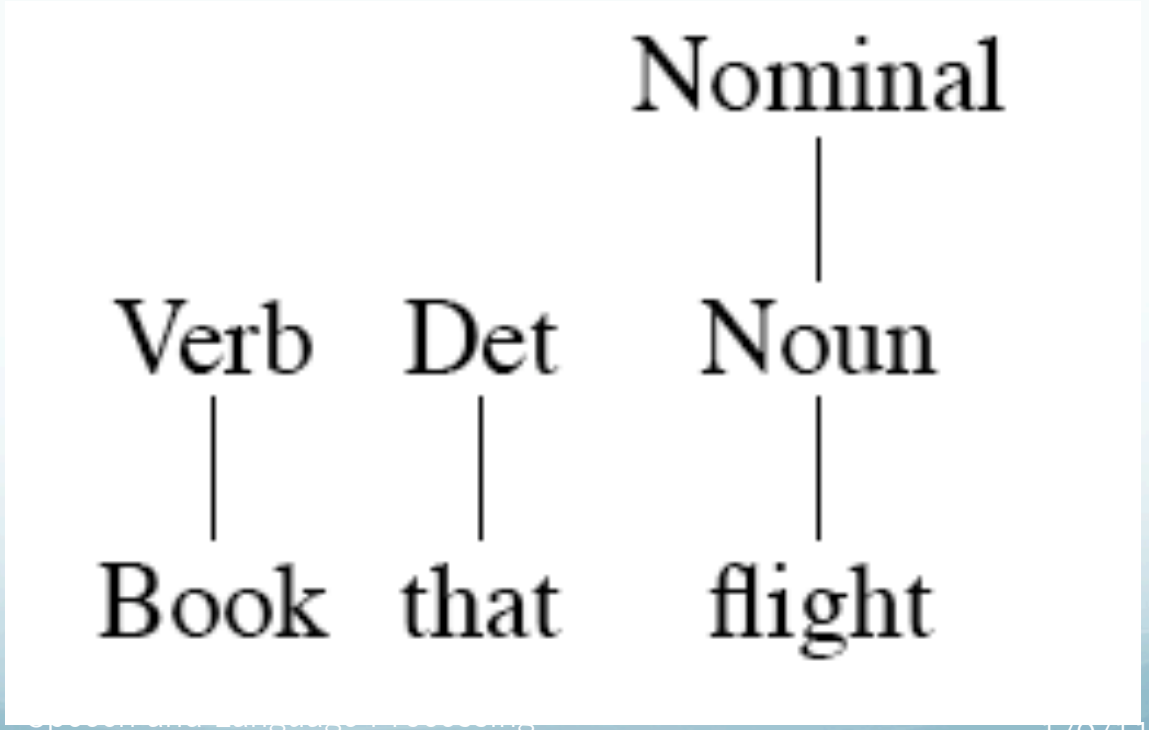
- Try to find all trees that span the input
  - Start with input string
    - Book that flight.
  - Use all productions with current subtree(s) on RHS
    - E.g., N -> Book; V -> Book
  - Stop when spanned by S (or no more rules apply)

Bottom-Up Search

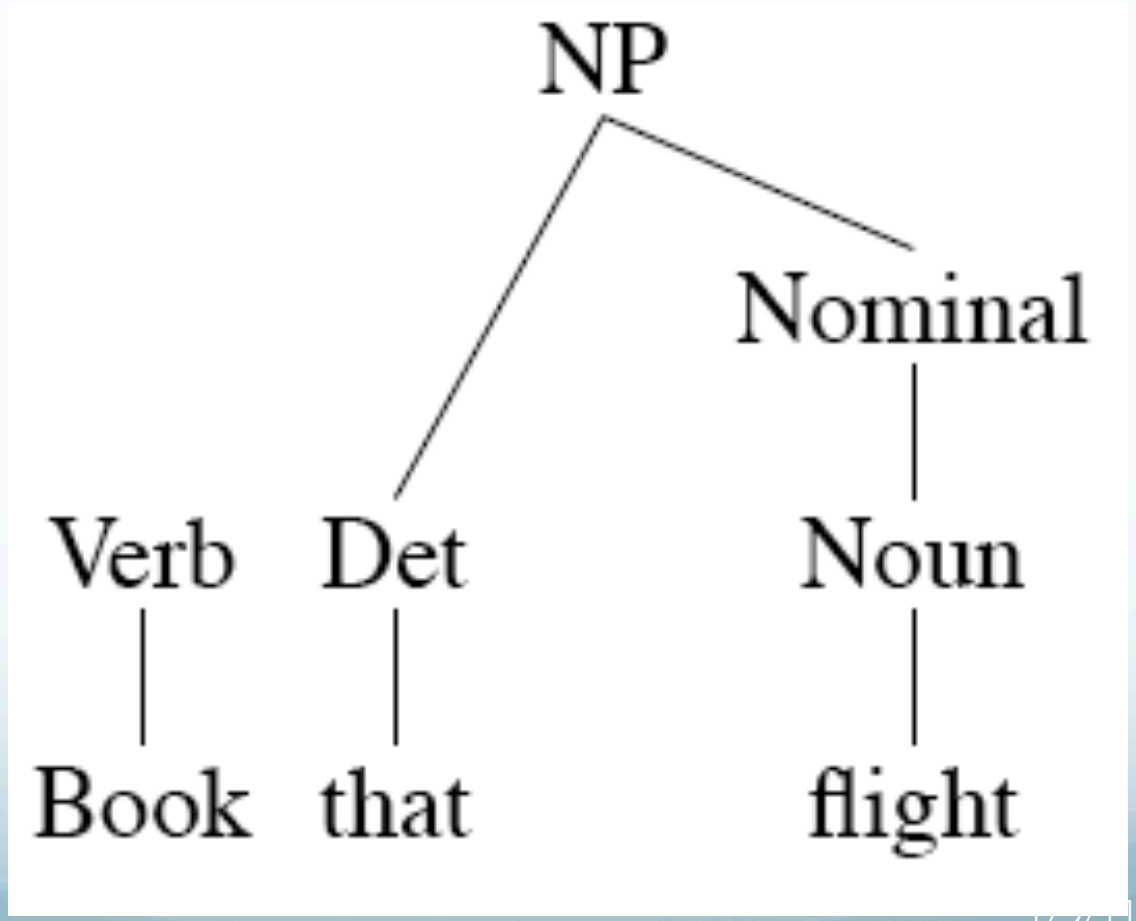
Bottom-Up Search



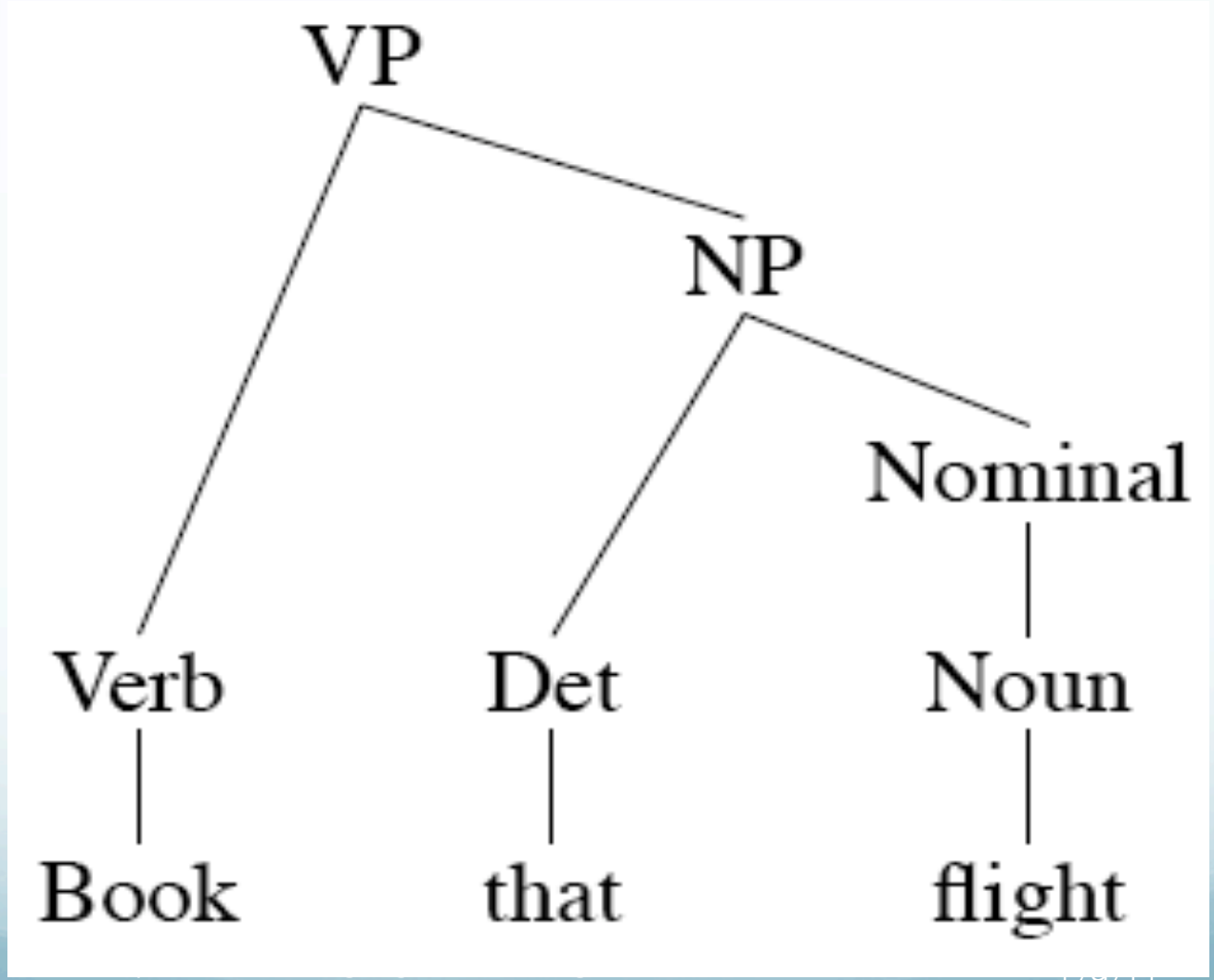
Bottom-Up Search



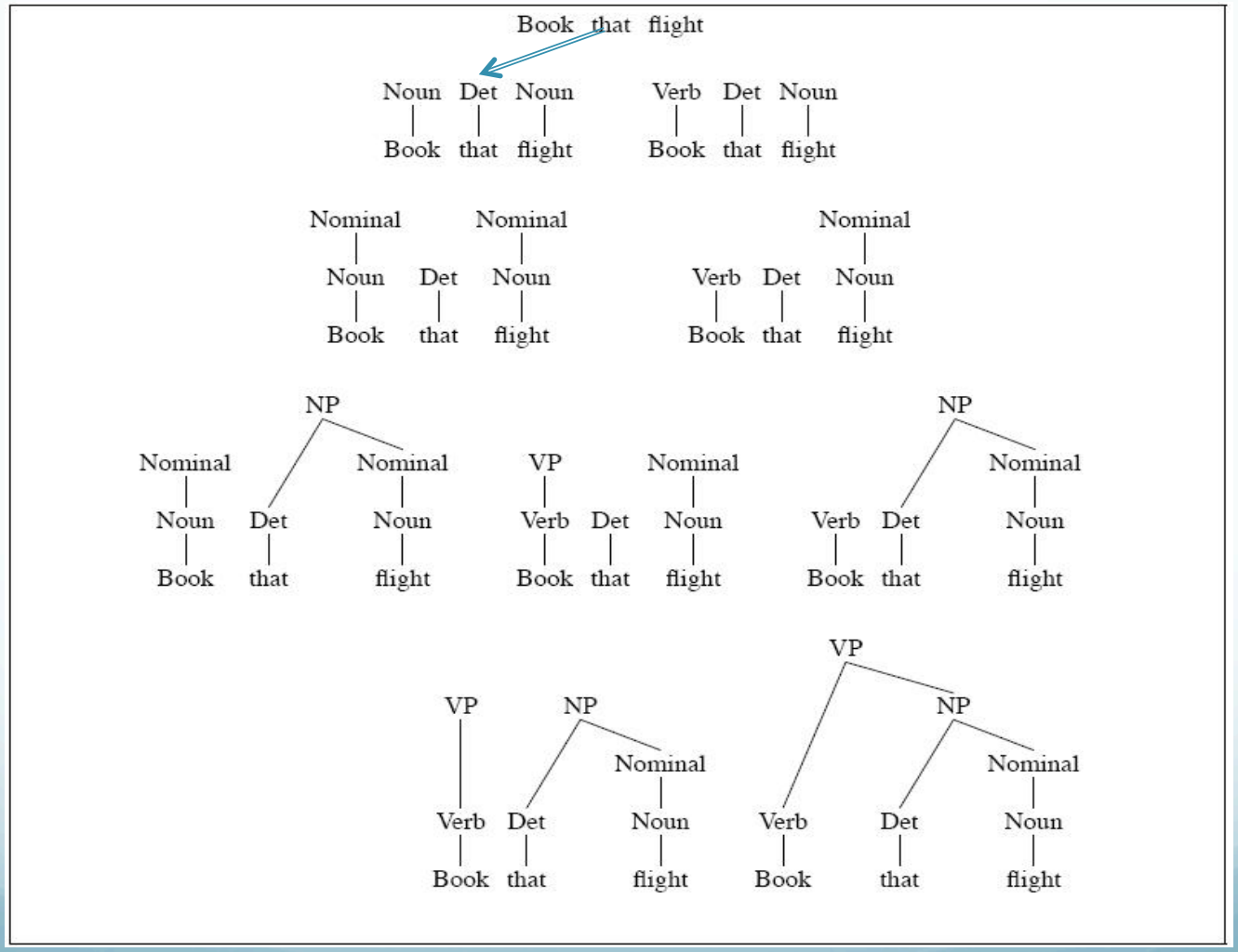
Bottom-Up Search



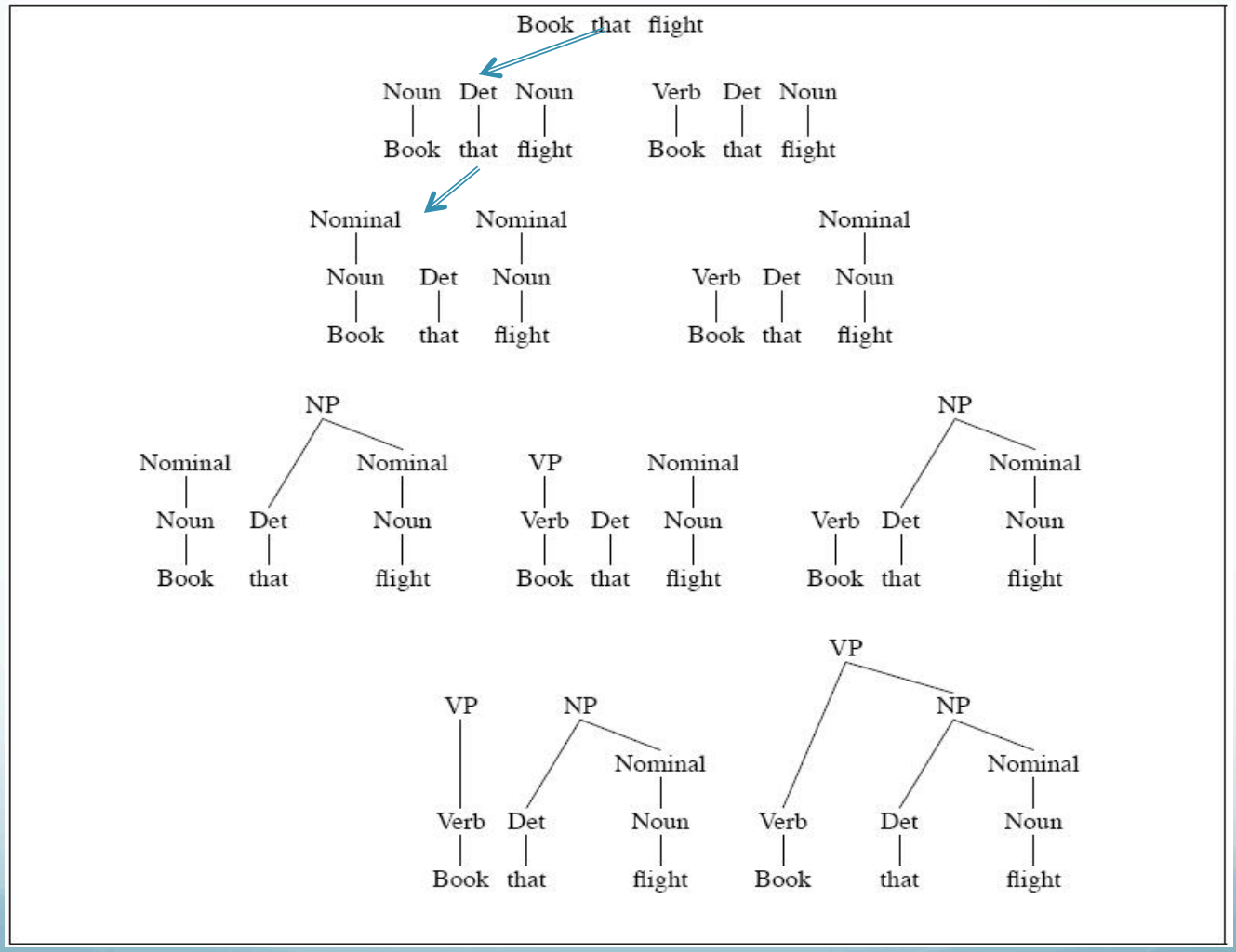
Bottom-Up Search



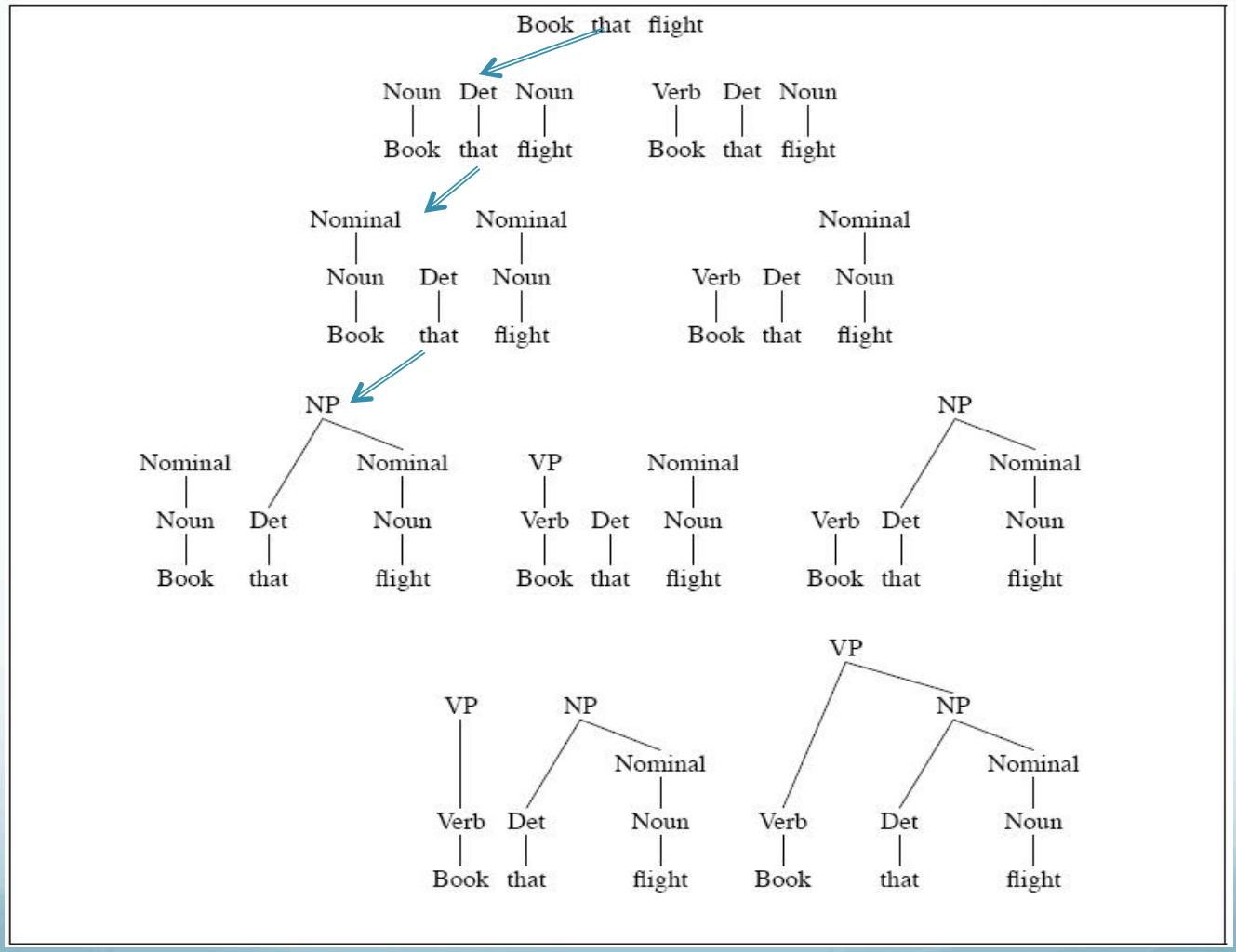
Bottom-Up Search



Bottom-Up Search



Bottom-Up Search



Pros and Cons of Bottom-Up Search

- Pros:

Pros and Cons of Bottom-Up Search

- Pros:
  - Will not explore trees that don't match input

Pros and Cons of Bottom-Up Search

- Pros:
  - Will not explore trees that don't match input
  - Recursive rules less problematic

Pros and Cons of Bottom-Up Search

- Pros:
  - Will not explore trees that don't match input
  - Recursive rules less problematic
  - Useful for incremental/ fragment parsing

Pros and Cons of Bottom-Up Search

- Pros:
  - Will not explore trees that don't match input
  - Recursive rules less problematic
  - Useful for incremental/ fragment parsing
- Cons:
  - Explore subtrees that will not fit full sentences

Parsing Challenges

- Ambiguity
- Repeated substructure
- Recursion

Parsing Ambiguity

- Many sources of parse ambiguity
  - Lexical ambiguity
    - Book/N; Book/V

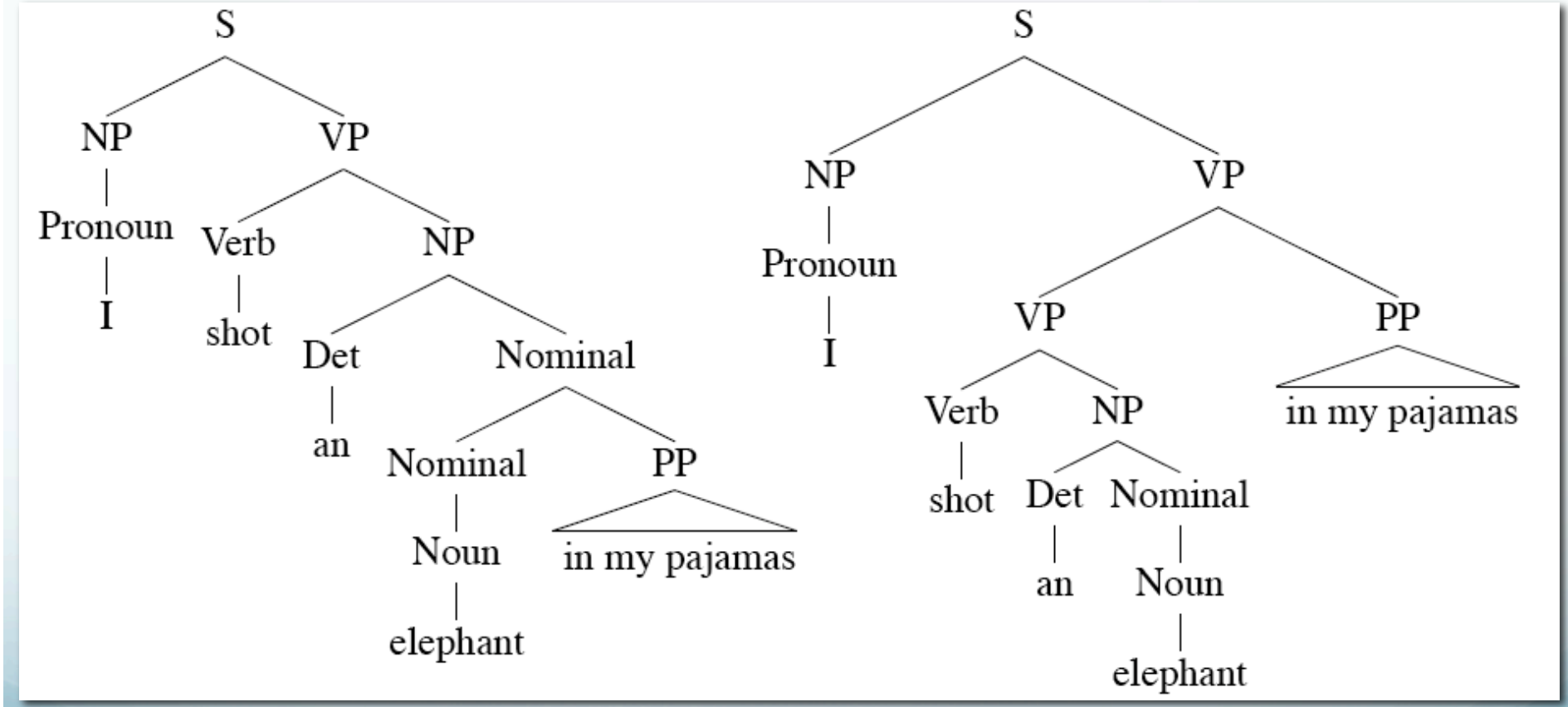
Parsing Ambiguity

- Many sources of parse ambiguity
  - Lexical ambiguity
    - Book/N; Book/V
  - Structural ambiguity: Main types:
    - Attachment ambiguity
      - Constituent can attach in multiple places
        - *I shot an elephant in my pyjamas*.

Parsing Ambiguity

- Many sources of parse ambiguity
  - Lexical ambiguity
    - Book/N; Book/V
  - Structural ambiguity: Main types:
    - Attachment ambiguity
      - Constituent can attach in multiple places
        - *I shot an elephant in my pyjamas*.
    - Coordination ambiguity
      - Different constituents can be conjoined
        - *Old men and women*

Ambiguity


S
NP
VP
Pronoun
Verb
NP
I
shot
Det
Nominal
an
Nominal
PP
Noun
in my pajamas
elephant
S
NP
VP
Pronoun
VP
PP
I
Verb
NP
in my pajamas
shot
Det
Nominal
an
Noun
elephant

Disambiguation

- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
    - Approaches exploit other information

Disambiguation

- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
    - Approaches exploit other information
      - Statistical
        - Some prepositional structs more likely to attach high/low
        - Some phrases more likely, e.g., (old (men and women))

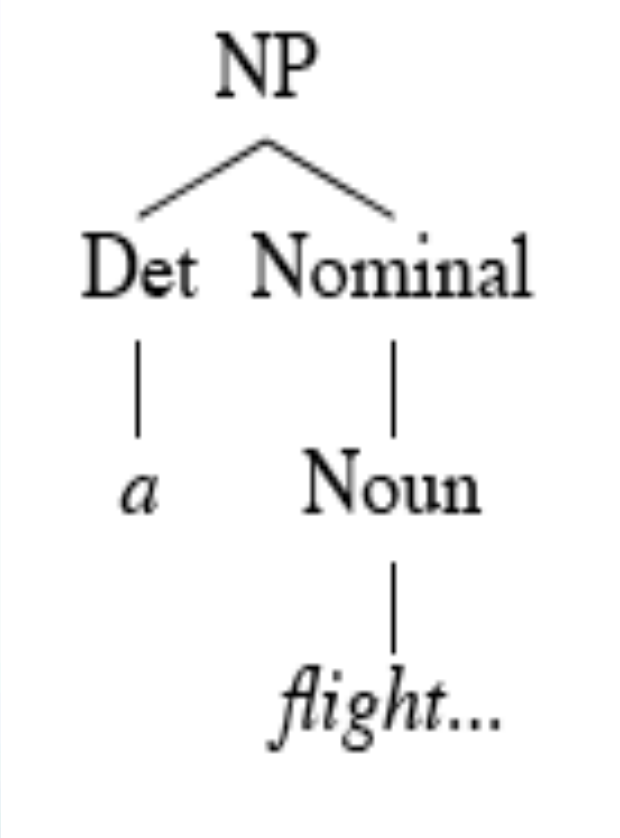
Disambiguation

- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
    - Approaches exploit other information
      - Statistical
        - Some prepositional structs more likely to attach high/low
        - Some phrases more likely, e.g., (old (men and women))
      - Semantic
      - Pragmatic
        - E.g., elephants and pyjamas
  - Alternatively, keep all
- Local ambiguity:
  - Ambiguity in subtree, resolved globally

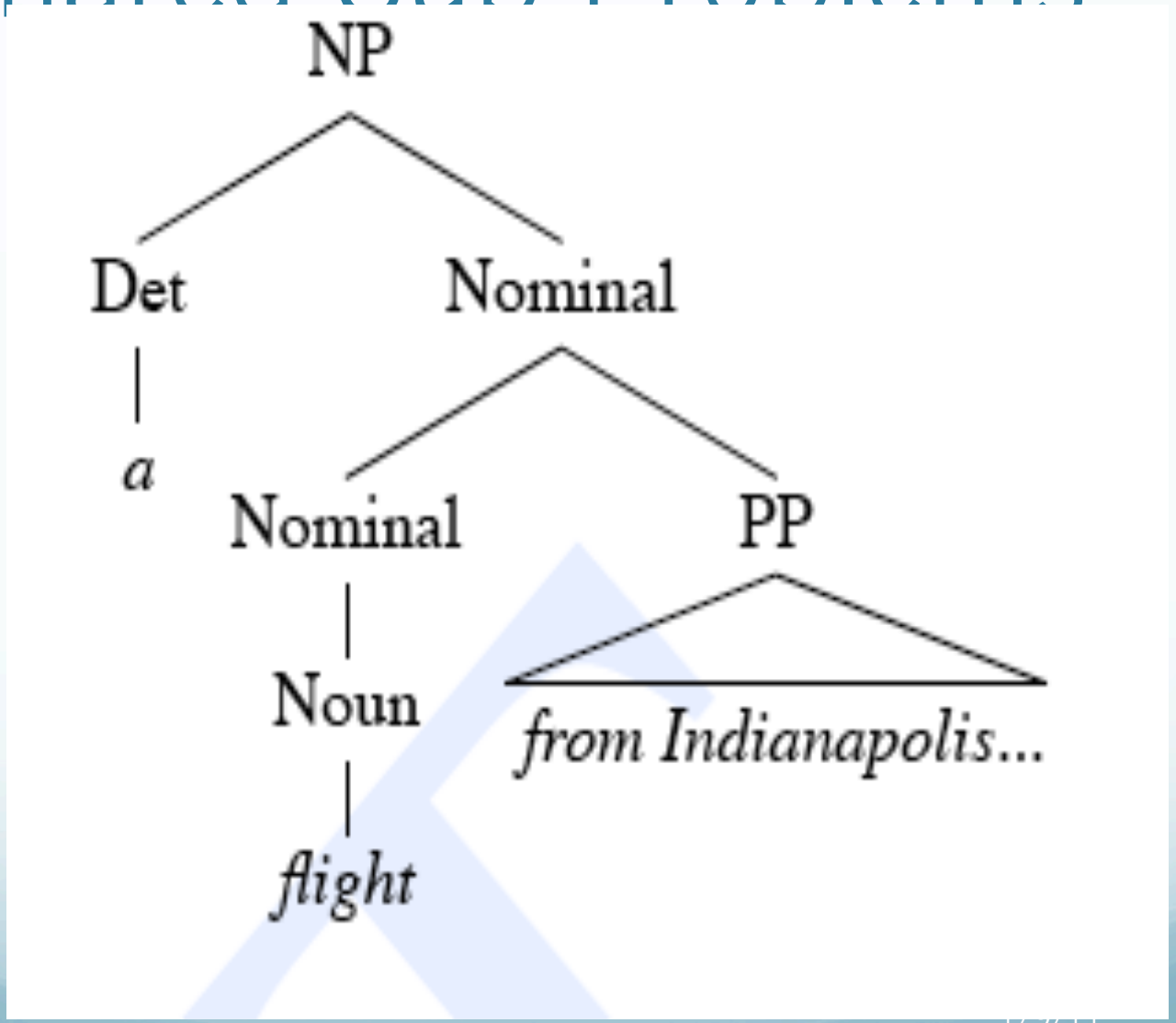
Repeated Work

- Top-down and bottom-up parsing both lead to repeated substructures
  - Globally bad parses can construct good subtrees
    - But overall parse will fail
    - Require reconstruction on other branch
  - No static backtracking strategy can avoid
- Efficient parsing techniques require storage of shared substructure
  - Typically with dynamic programming

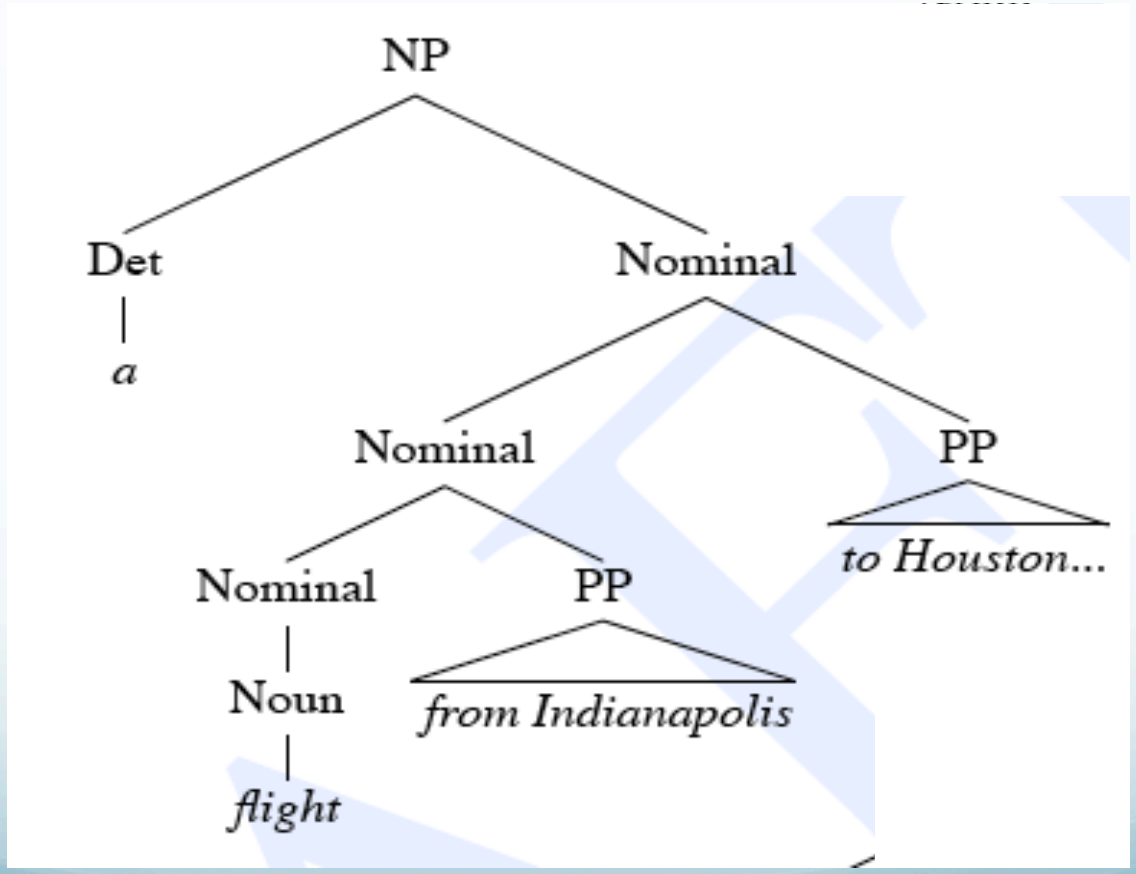
Shared Sub-Problems



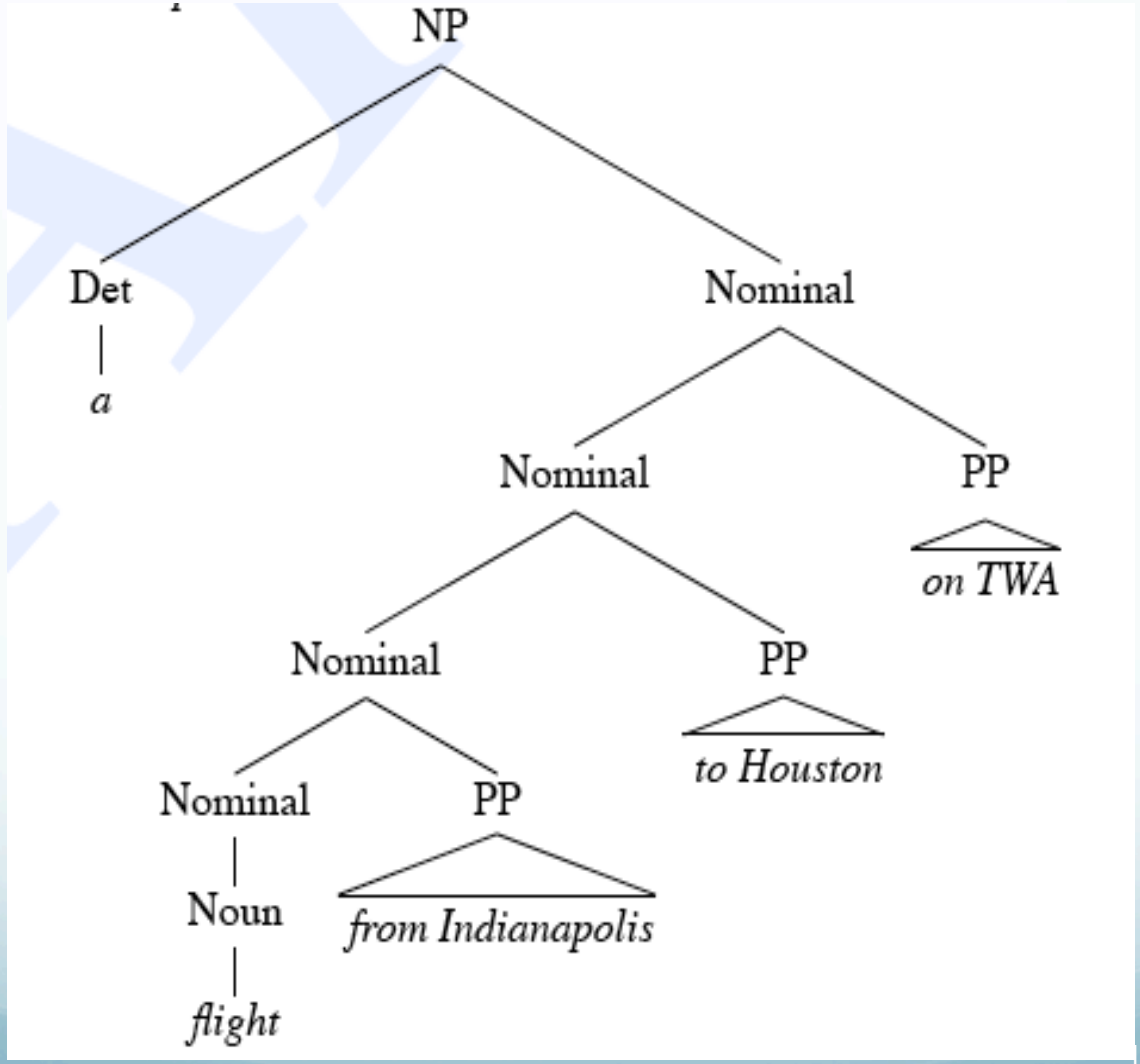
Shared Sub-Problems



Shared Sub-Problems



Shared Sub-Problems



Recursion

- Many grammars have recursive rules
  - E.g., S -> S Conj S
- In search approaches, recursion is problematic
  - Can yield infinite searches
    - Esp., top-down

Garden Paths

- Misleading partial analysis
  - Leads to backtracking, failure of initial analysis
  - The horse raced past the barn fell =>

Garden Paths

- Misleading partial analysis
  - Leads to backtracking, failure of initial analysis

  - The horse raced past the barn fell =>
  - The horse, raced past the barn, fell =>

Garden Paths

- Misleading partial analysis
  - Leads to backtracking, failure of initial analysis

  - The horse raced past the barn fell =>
  - The horse, raced past the barn, fell =>
  - The horse which was raced past the barn fell.

Dynamic Programming

- Challenge: Repeated substructure -> Repeated work
- Insight:
  - Global parse composed of parse substructures
  - Can record parses of substructures
- Dynamic programming avoids repeated work by tabulating solutions to subproblems
  - Here, stores subtrees

Parsing w/Dynamic Programming

- Avoids repeated work
- Allows implementation of (relatively) efficient parsing algorithms
  - Polynomial time in input length
    - Typically cubic (n^3) or less
- Several different implementations
  - Cocke-Kasami-Younger (CKY) algorithm
  - Earley algorithm
  - Chart parsing

Chomsky Normal Form (CNF)

- CKY parsing requires grammars in CNF
- Chomsky Normal Form
  - All productions of the form:
    - A -> B C, or
    - A -> a
- However, most of our grammars are not of this form
  - E.g., S -> Wh-NP Aux NP VP
- Need a general conversion procedue
  - Any arbitrary grammar can be converted to CNF

CNF Conversion

- Three main conditions:
  - Hybrid rules:
    - INF-VP -> to VP
  - Unit productions:
    - A -> B
  - Long productions:
    - A -> B C D

CNF Conversion

- Hybrid rule conversion:
  - Replace all terminals with dummy non-terminals
  - E.g., INF-VP -> to VP
    - INF-VP -> TO VP; TO -> to

- Unit productions:
  - Rewrite RHS with RHS of all derivable non-unit productions
    - If $A \overset{*}{\Rightarrow} B$ and B -> w, then add A -> w

CNF Conversion

- Long productions:
  - Introduce new non-terminals and spread over rules
  - S -> Aux NP VP
    - S -> X1 VP; X1 -> Aux NP
- For all non-conforming rules,
  - Convert terminals to dummy non-terminals
  - Convert unit productions
  - Binarize all resulting rules

$\mathscr{L}_1$ Grammar	$\mathscr{L}_1$ in CNF
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$