Syntax: Context-free Grammars

Ling 571
Deep Processing Techniques for NLP
January 9, 2017

Roadmap

- Motivation: Applications
- Context-free grammars (CFGs)
  - Formalism
  - Grammars for English
  - Treebanks and CFGs
  - Speech and Text
  - Parsing

Applications

- Shallow techniques useful, but limited
- Deeper analysis supports:
  - Grammar-checking – and teaching
  - Question-answering
  - Information extraction
  - Dialogue understanding

Representing Syntax

- Context-free grammars
- CFGs: 4-tuple
  - A set of terminal symbols: Σ
  - A set of non-terminal symbols: N
  - A set of productions P: of the form $A \rightarrow \alpha$
    - Where A is a non-terminal and α in $(\Sigma \cup N)^*$
  - A designated start symbol S

CFG Components

- Terminals:
  - Only appear as leaves of parse tree
  - Right-hand side of productions (rules) (RHS)
  - Words of the language
    - Cat, dog, is, the, bark, chase

- Non-terminals
  - Do not appear as leaves of parse tree
  - Appear on left or right side of productions (rules)
  - Constituents of language
    - NP, VP, Sentence, etc

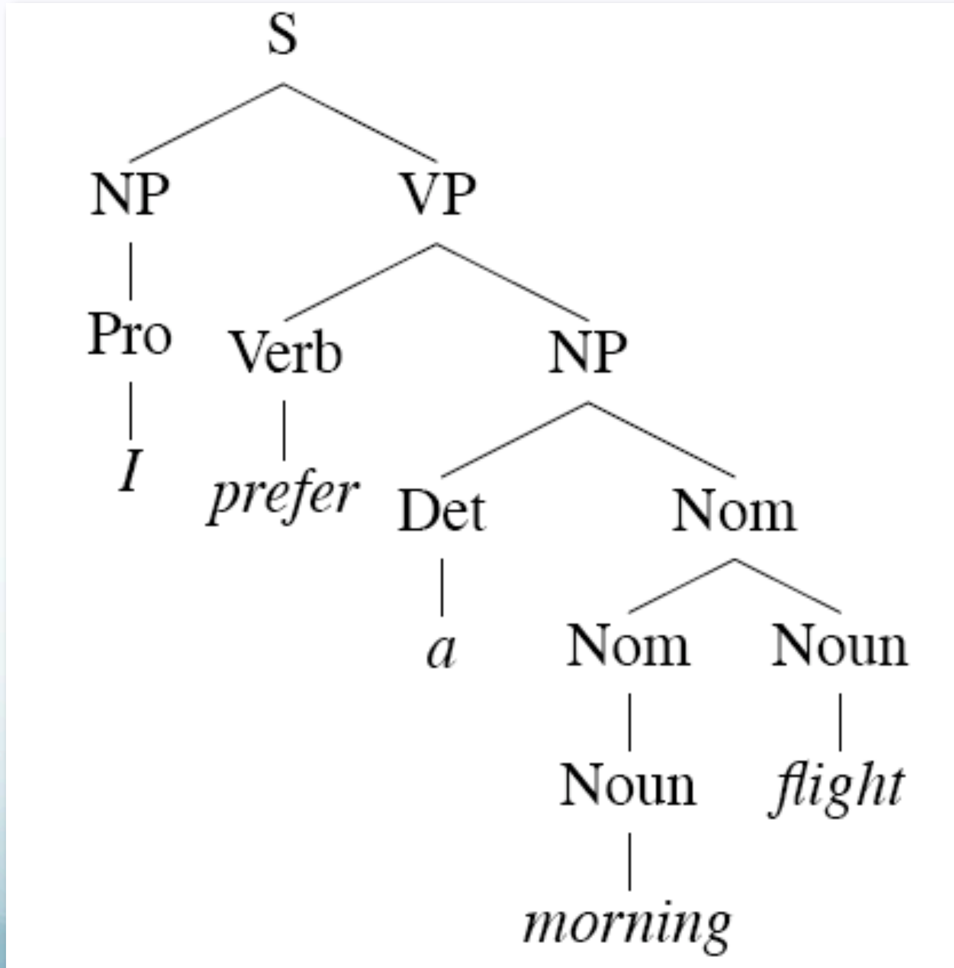
CFG Components

- Productions
  - Rules with one non-terminal on LHS and any number of terminals and non-terminals on RHS

  - S → NP VP
  - VP → V NP PP | V NP
  - Nominal → Noun | Nominal Noun
  - Noun → dog | cat | rat
  - Det → the

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

Parse Tree


S
NP
VP
Pro
Verb
NP
I
prefer
Det
Nom
a
Nom
Noun
Noun
flight
morning

Some English Grammar

- Sentences: Full sentence or clause; a complete thought
  - Declarative: S → NP VP
    - I want a flight from Sea-Tac to Denver.
  - Imperative: S → VP
    - Show me the cheapest flight from New York to Los Angeles.
  - S → Aux NP VP
    - Can you give me the non-stop flights to Boston?
  - S → Wh-NP VP
    - Which flights arrive in Pittsburgh before 10pm?
  - S → Wh-NP Aux NP VP
    - What flights do you have from Seattle to Orlando?

The Noun Phrase

- NP → Pronoun | Proper Noun (NNP) | Det Nominal
  - Head noun + pre-/post-modifiers
- Determiners:
  - Det → DT
    - the, this, a, those
  - Det → NP 's
    - United's flight, Chicago's airport

In and around the Noun

- Nominal → Noun
  - PTB POS: NN, NNS, NNP, NNPS
  - flight, dinner, airport
- NP → (Det) (Card) (Ord) (Quant) (AP) Nominal
  - The least expensive fare, one flight, the first route
- Nominal → Nominal PP
  - The flight from Chicago

Verb Phrase and Subcategorization

- Verb phrase includes Verb, other constituents
  - Subcategorization frame: what constituent arguments the verb requires

  - VP → Verb — disappear
  - VP → Verb NP — book a flight
  - VP → Verb PP PP — fly from Chicago to Seattle
  - VP → Verb S — think I want that flight
  - VP → Verb VP — want to arrange three flights

CFGs and Subcategorization

- Issues?
  - I prefer United has a flight.

- How can we solve this problem?
  - Create explicit subclasses of verb
    - Verb-with-NP
    - Verb-with-S-complement, etc...
- Is this a good solution?
  - No, explosive increase in number of rules
  - Similar problem with agreement

Treebanks

- Treebank:
  - Large corpus of sentences all of which are annotated syntactically with a parse
    - Built semi-automatically
      - Automatic parse with manual correction
  - Examples:
    - Penn Treebank (largest)
      - English: Brown (balanced); Switchboard (conversational speech); ATIS (human-computer dialogue); Wall Street Journal; Chinese; Arabic
    - Korean, Hindi,..
    - DeepBank, Prague dependency,...

Treebanks

- Include wealth of language information
  - Traces, grammatical function (subject, topic, etc), semantic function (temporal, location)

- Implicitly constitutes grammar of language
  - Can read off rewrite rules from bracketing
  - Not only presence of rules, but frequency
  - Will be crucial in building statistical parsers

Treebank WSJ Example

```
( (S (`` ``)
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets)))))))))))))
    (, ,) ('' '')
    (NP-SBJ (PRP he) )
    (VP (VBD said)
      (S (-NONE- *T*-2) ))
    (. .) ))
```

Treebanks & Corpora

- Many corpora on patas
- patas$ ls /corpora

birkbeck	enron_email_dataset	grammars	LEAP	TREC
Coconut	europarl	ICAME	med-data	treebanks
Conll	europarl-old	JRC-Acquis.3.0	nltk	
DUC	framenet	LDC	proj-gutenberg	

- Also, corpus search function on CLMS wiki

- Many large corpora from LDC
- Many corpus samples in nltk

Treebank Issues

- Large, expensive to produce
- Complex
  - Agreement among labelers can be an issue
- Labeling implicitly captures theoretical bias
  - Penn Treebank is 'bushy', long productions
- Enormous numbers of rules
  - 4,500 rules in PTB for VP
    - VP→ V PP PP PP
  - 1M rule tokens; 17,500 distinct types – and counting!

Spoken & Written

- Can we just use models for written language directly?
- No!
- Challenges of spoken language
  - Disfluency
    - Can I um uh can I g- get a flight to Boston on the 15th?
      - 37% of Switchboard utts > 2 wds
  - Short, fragmentary
    - Uh one way
  - More pronouns, ellipsis
    - That one

Computational Parsing

- Given a grammar, how can we derive the analysis of an input sentence?
  - Parsing as search
  - CKY parsing

- Given a body of (annotated) text, how can we derive the grammar rules of a language, and employ them in automatic parsing?

  - Treebanks & PCFGs

Algorithmic Parsing

Ling 571
Deep Processing Techniques for NLP
January 9, 2017

Roadmap

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

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
  - 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:
  - 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: 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:
  - 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
    - 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

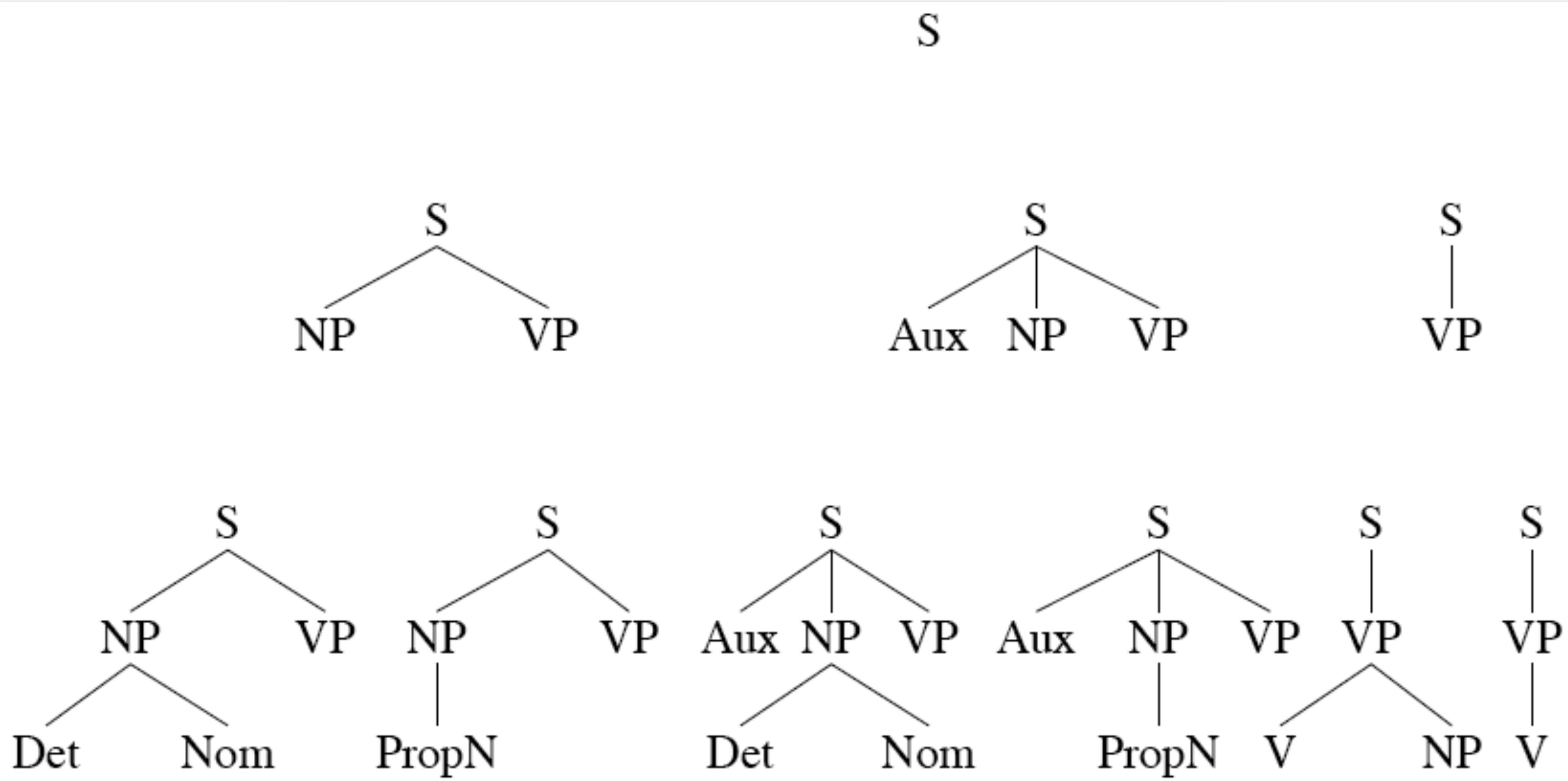
Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP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

Book that flight.

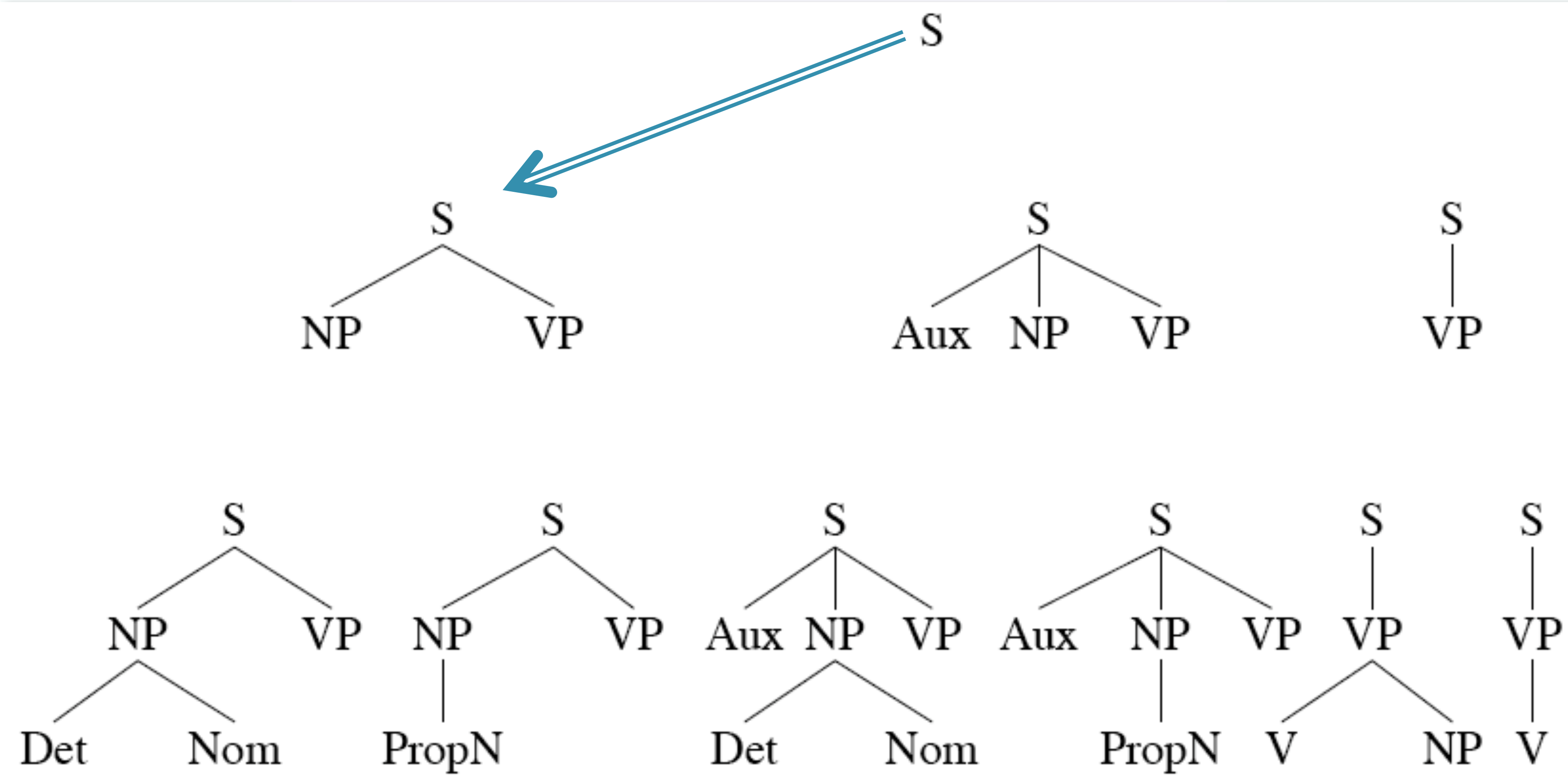
Top-down Search

- 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*

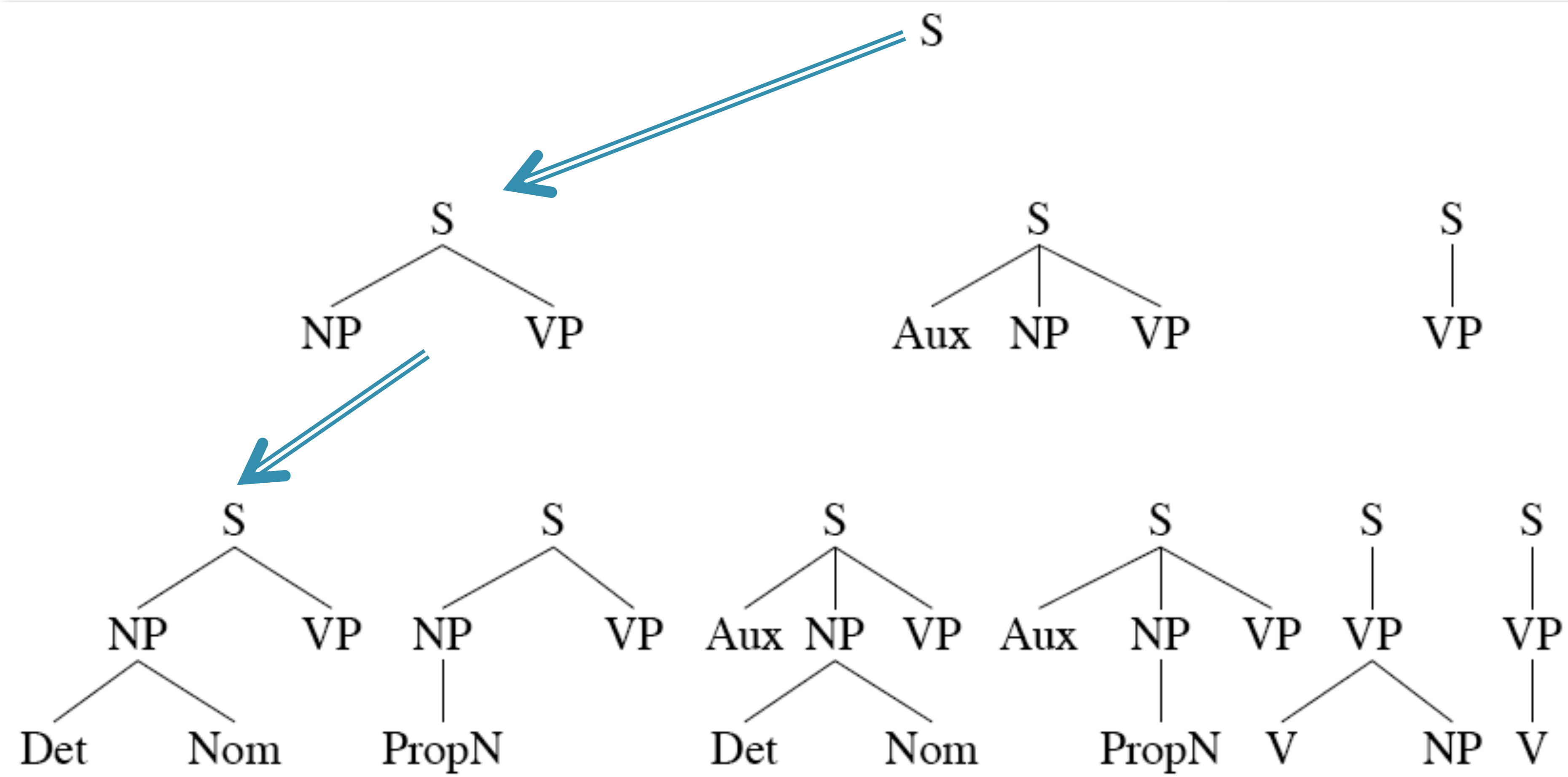
Top-down Search



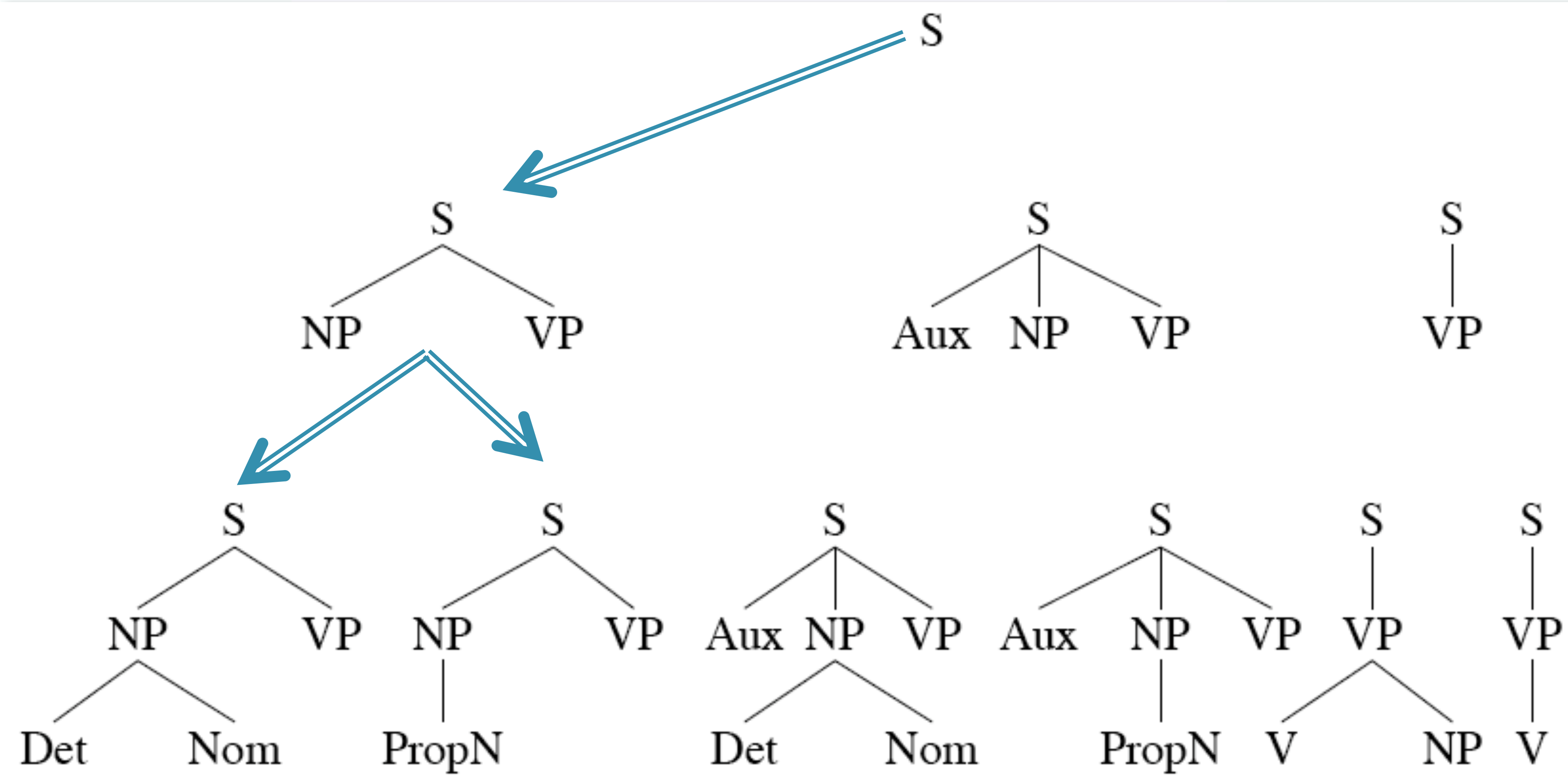
Depth-first Search



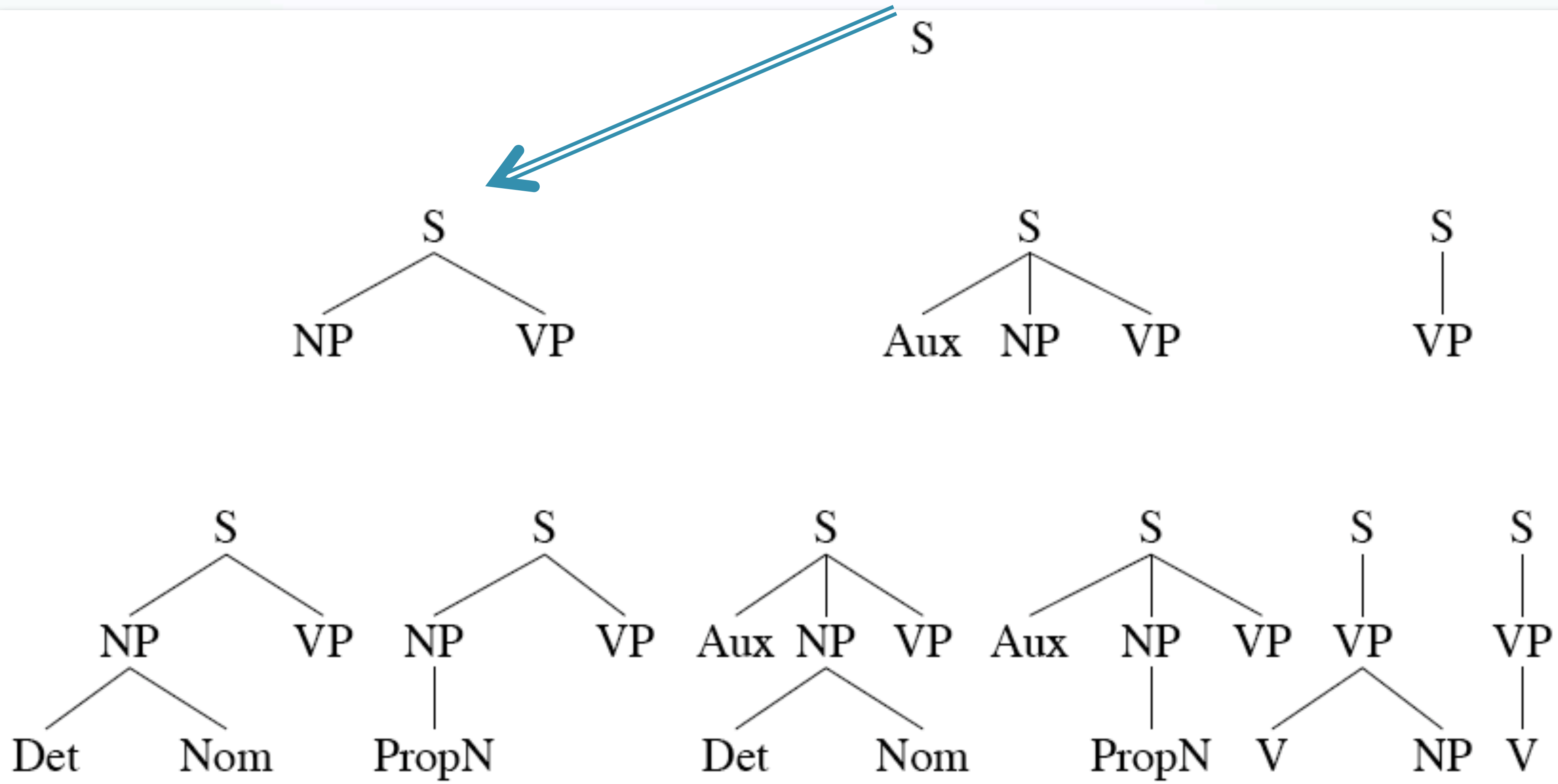
Depth-first Search



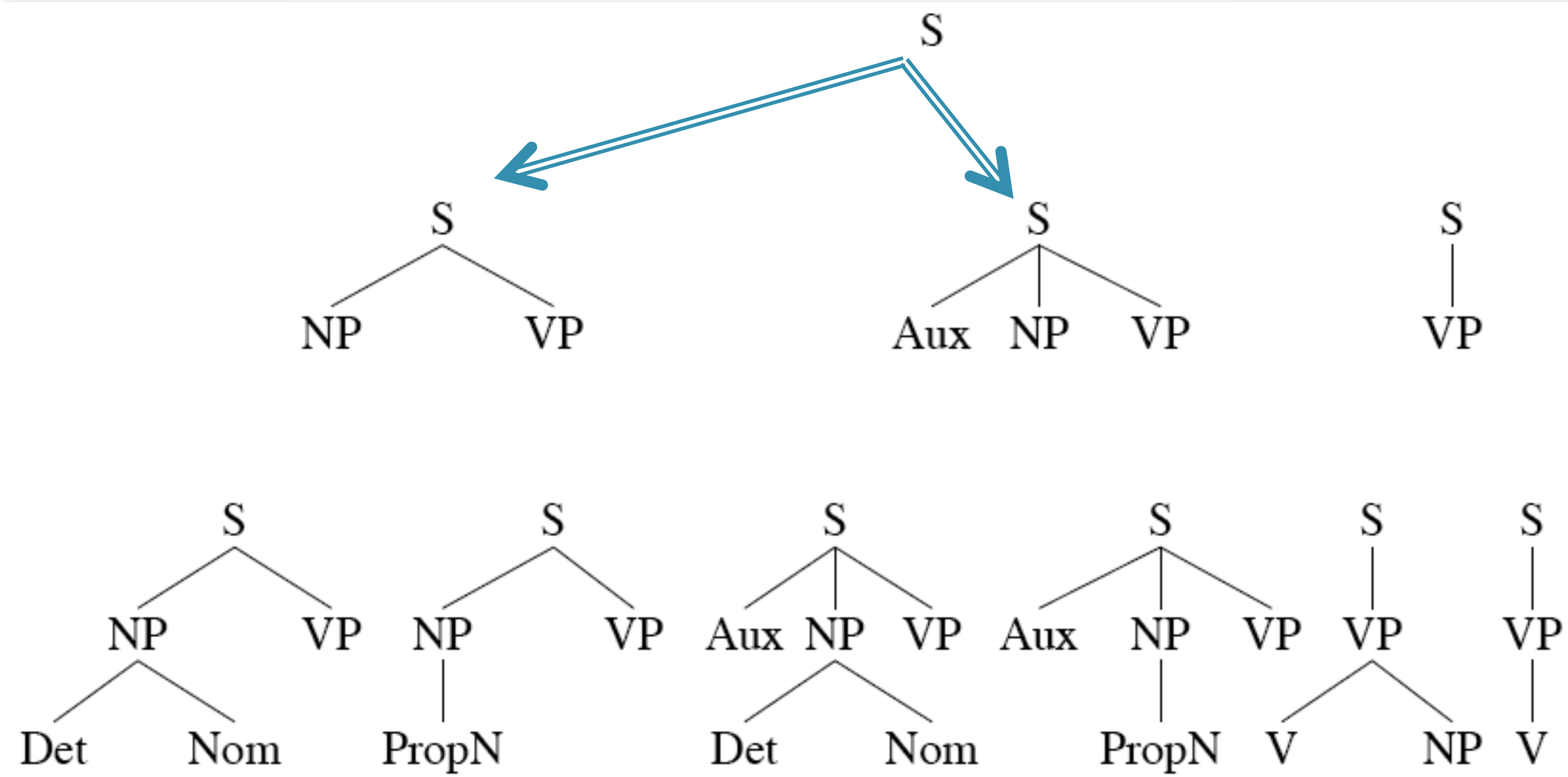
Depth-first Search



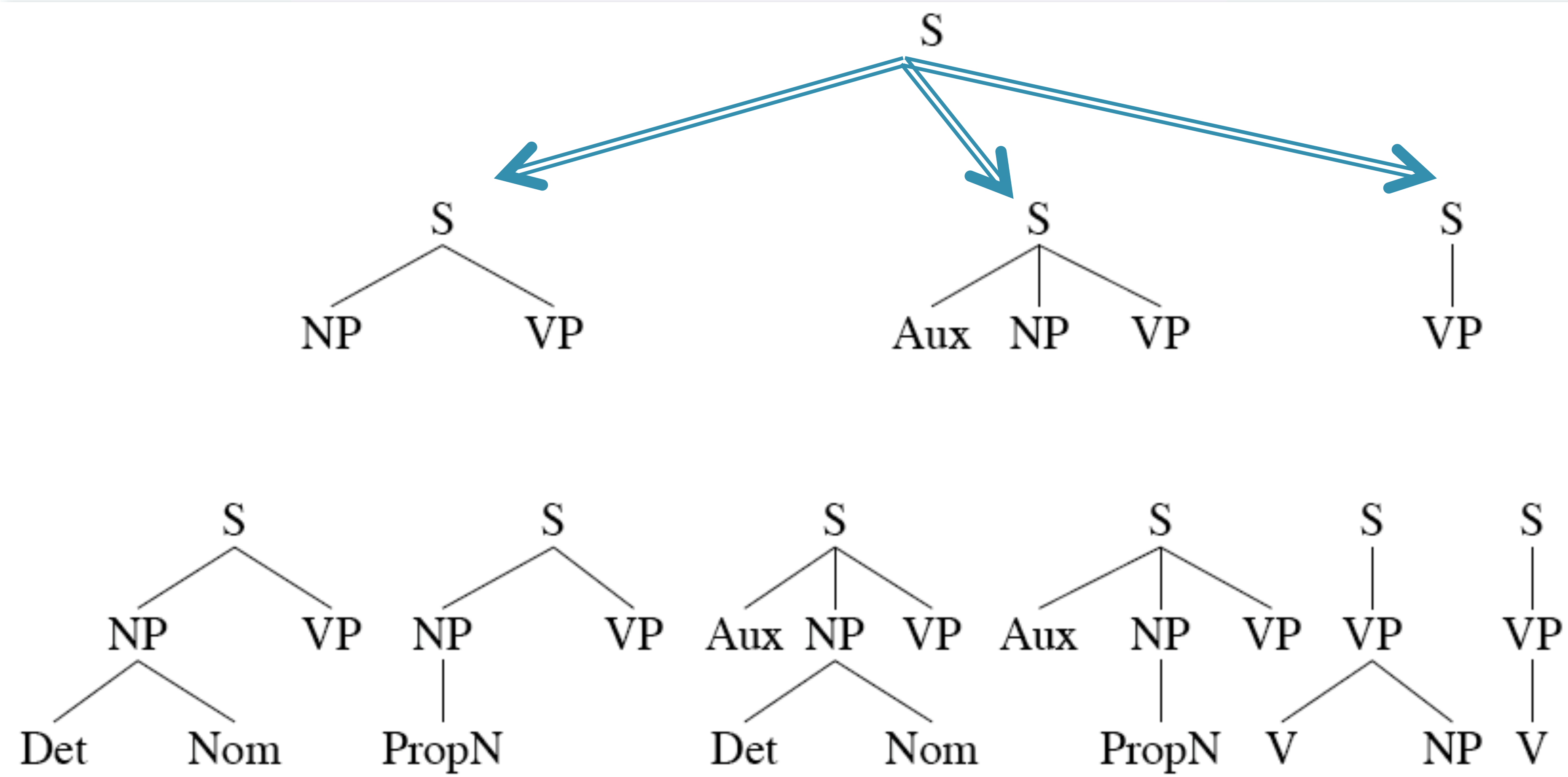
Breadth-first Search



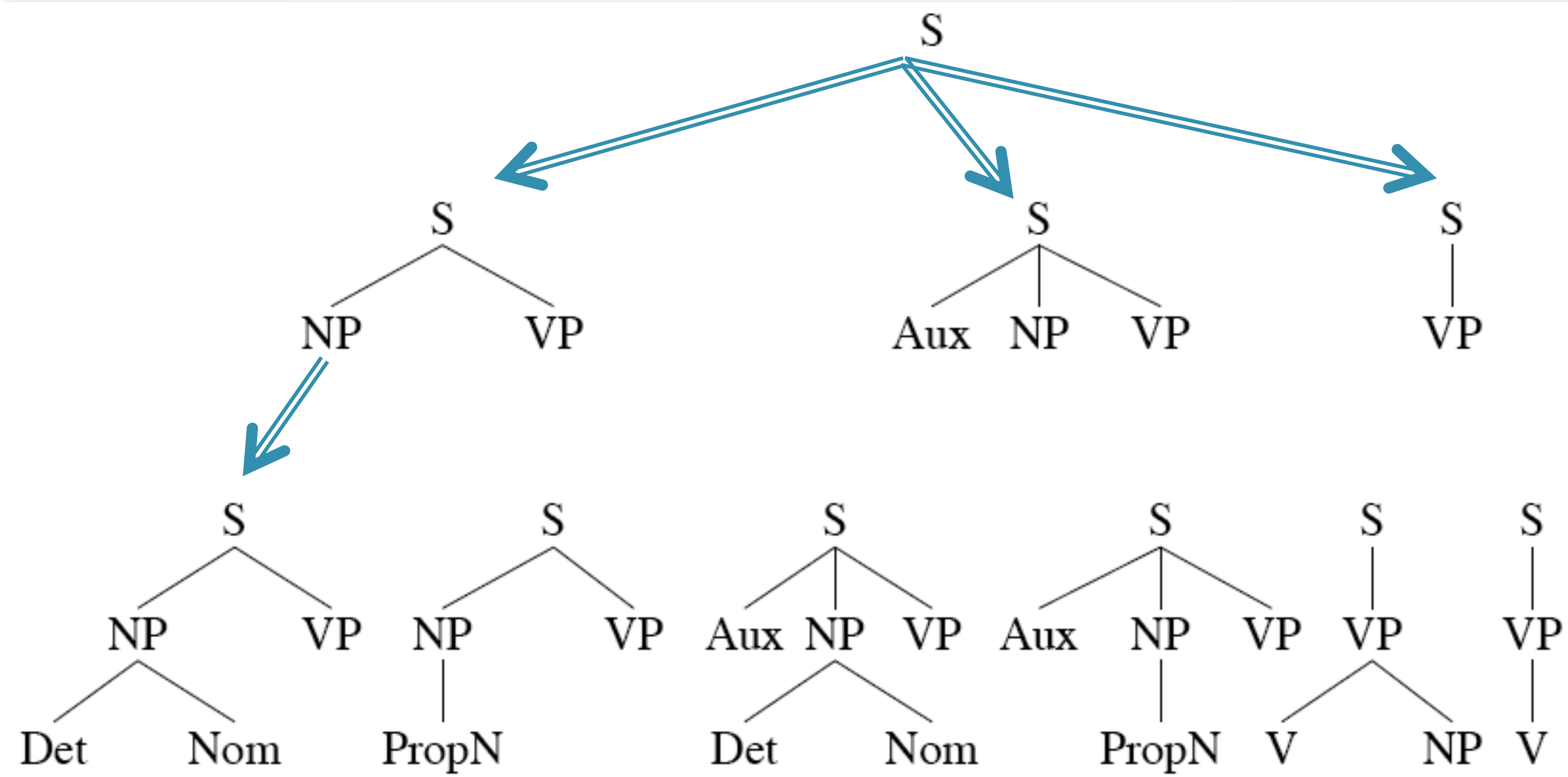
Breadth-first Search



Breadth-first Search



Breadth-first Search



Pros and Cons of Top-down Parsing

- Pros:
  - 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