

CKY Parsing & CNF Conversion

LING 571 — Deep Processing Techniques for NLP

October 5, 2022

Shane Steinert-Threlkeld

Announcements

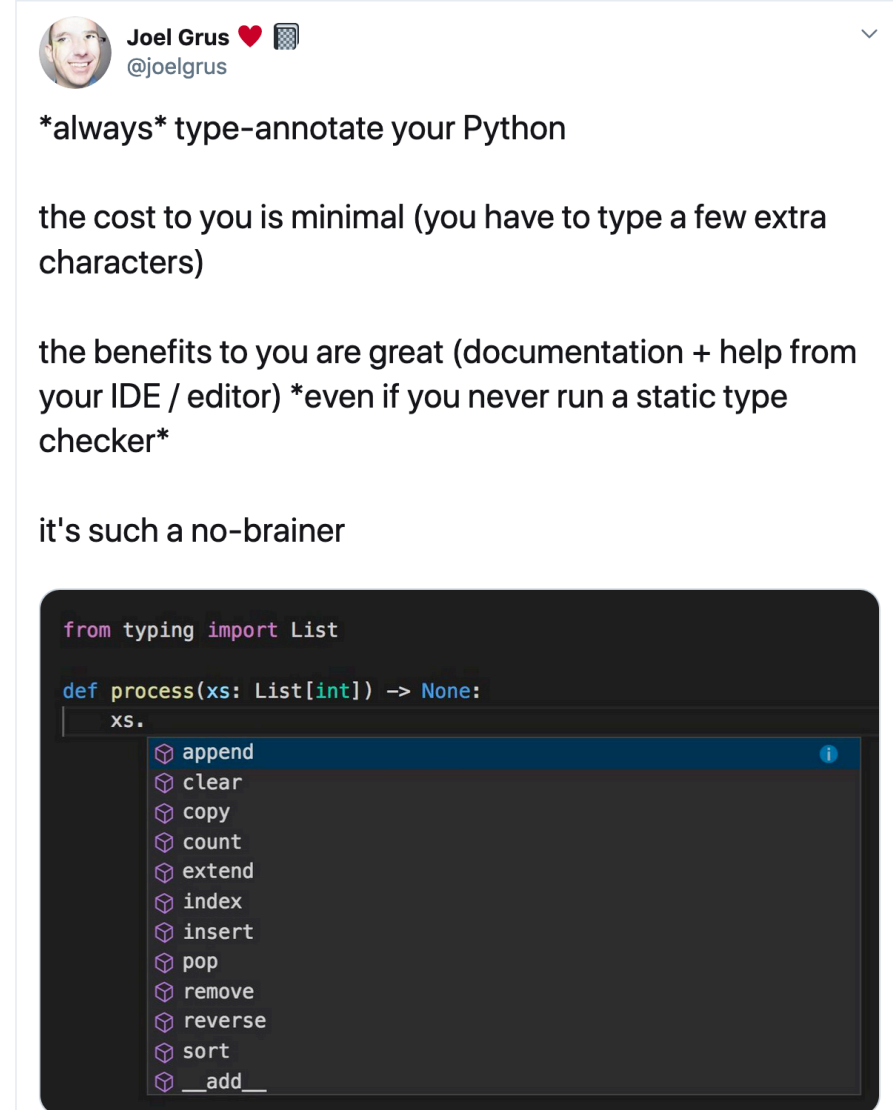
- **HW #1** due tonight at **11:59pm**.
- Python on Patas: installed versions ``ls /opt | grep python``. E.g., invoke by:
 - `/opt/python-3.6/bin/python3`
 - `nltk` is installed.
- [For personal projects, but not 571 HW, you can use the latest of everything via Anaconda (download with `wget`).]
- When in doubt, use *full paths* for everything (python binary, file names, etc)
- `check_hwX.sh`: invoke from your local directory (for permission reasons)

Type Hinting in Python

- Supported in ≥ 3.6 [[tutorial](#)]

```
from typing import List
from nltk.grammar import Production
```

```
def fix_hybrid_production(hybrid_prod: Production) -> List[Production]:
    ...
```



Type Hinting in Python

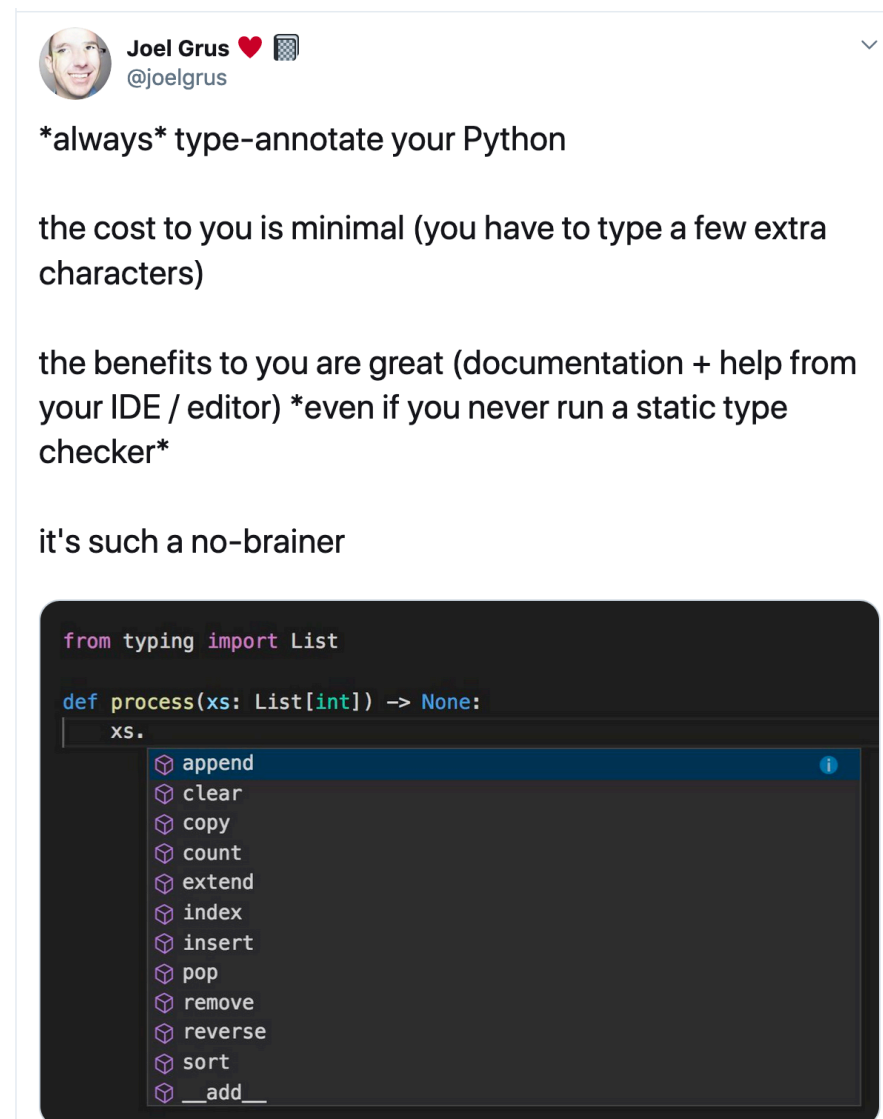
- Supported in ≥ 3.6 [tutorial]

```
from typing import List
from nltk.grammar import Production
```

```
def fix_hybrid_production(hybrid_prod: Production) -> List[Production]:
    ...
```

- Also available in PyCharm through docstrings and/or comments:

```
def fix_hybrid_productions(hybrid_prod):
    """
    This function takes a hybrid production and
    returns a list of new CNF productions
    :type hybrid_prod: Production
    :rtype: list[Production]
    """
```



Joke of the Week (PP Attachment Ambiguity)

tott @crazytott · Oct 5

A cop just knocked on my door and told me that my dogs were chasing people on bikes???? Wtf??? My dogs don't even own bikes tf

Roadmap

- **Parsing-as-Search**
- Parsing Challenges
- Strategy: Dynamic Programming
- Grammar Equivalence
- CKY parsing algorithm

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

What is Parsing?

- CFG parsing is the task of assigning trees to input strings
 - For any input A and grammar G
 - ...assign ≥ 0 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 single (or correct) analysis
- Subtask: Recognition
 - Given input A , G – is A in language defined by G or not?

Motivation

- Is this sentence in the language — i.e. is it “grammatical?”
 - ** I prefer United has the earliest flight.*
 - FSAs accept regular languages defined by finite-state automata.
 - Our parsers accept languages defined by CFG (equiv. pushdown automata).

Motivation

- Is this sentence in the language — i.e. is it “grammatical?”
 - ** I prefer United has the earliest flight.*
 - FSAs accept regular languages defined by finite-state automata.
 - Our parsers accept languages defined by CFG (equiv. pushdown automata).
- 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? Direct object?

Parsing as Search

- Syntactic parsing searches through possible trees to find one or more trees that derive input

Parsing as Search

- Syntactic parsing searches through possible trees to find one or more trees that derive input
- Formally, search problems are defined by:
 - Start state S
 - Goal state G (with a test)
 - Set of actions that transition from one state to another
 - “Successor function”
 - A path cost function

Parsing as Search: One Model

- Start State S : Start Symbol

Parsing as Search: One Model

- Start State S : Start Symbol
- Goal test:
 - Does the parse tree cover all of, and only, the input?

Parsing as Search: One Model

- Start State S : Start Symbol
- Goal test:
 - Does the parse tree cover all of, and only, the input?
- Successor function:
 - Expand a nonterminal using a production where nonterminal is the LHS of the production

Parsing as Search: One Model

- Start State S : Start Symbol
- Goal test:
 - Does the parse tree cover all of, and only, the input?
- Successor function:
 - Expand a nonterminal using a production where nonterminal is the LHS of the production
- Path cost:
 - ...ignored for now.

Parsing as Search: One Model

- Node:
 - Partial solution to search problem (partial parse)

Parsing as Search: One Model

- Node:
 - Partial solution to search problem (partial parse)
- Search start node (initial state):
 - Input string
 - Start symbol of CFG

Parsing as Search: One Model

- 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 of, and only the input, rooted at S

Search Algorithms

- Depth First
 - Keep expanding nonterminals until they reach words
 - If no more expansions available, back up

Search Algorithms

- Depth First
 - Keep expanding nonterminals until they reach words
 - If no more expansions available, back up
- Breadth First
 - Consider all parses that expand a single nonterminal...
 - ...then all with two expanded, etc...

Search Algorithms

- Depth First
 - Keep expanding nonterminals until they reach words
 - If no more expansions available, back up
- Breadth First
 - Consider all parses that expand a single nonterminal...
 - ...then all with two expanded, etc...
- 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

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)
 - Bottom-up search (Data-driven search)

A Grammar

Grammar	Lexicon
$S \rightarrow NP VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux NP VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$

Jurafsky & Martin, Speech and Language Processing, p.390

A Grammar

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

Jurafsky & Martin, Speech and Language Processing, p.390

A Grammar

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

Jurafsky & Martin, Speech and Language Processing, p.390

A Grammar

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

Jurafsky & Martin, Speech and Language Processing, p.390

Top-down Search

- All valid parse trees must be rooted with start symbol

Top-down Search

- All valid parse trees must be rooted with start symbol
- Begin search with productions where S is on LHS
 - e.g. $S \rightarrow NP VP$

Top-down Search

- All valid parse trees must be rooted with start symbol
- Begin search with productions where S is on LHS
 - e.g. $S \rightarrow NP VP$
- Successively expand nonterminals
 - e.g. $NP \rightarrow Det Nominal$; $VP \rightarrow V NP$

Top-down Search

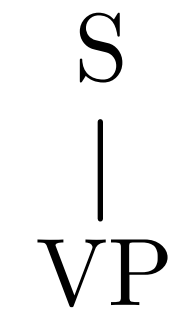
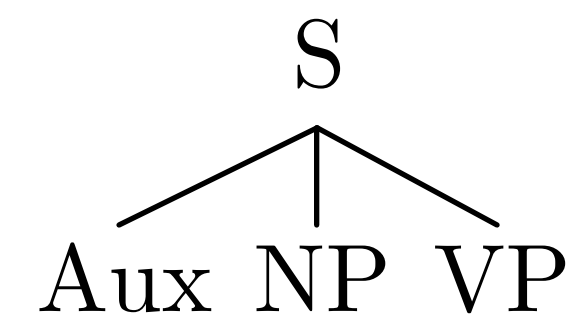
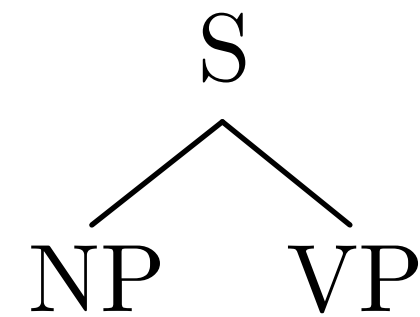
- All valid parse trees must be rooted with start symbol
- Begin search with productions where S is on LHS
 - e.g. $S \rightarrow NP VP$
- Successively expand nonterminals
 - e.g. $NP \rightarrow Det Nominal$; $VP \rightarrow V NP$
- Terminate when all leaves are terminals

Depth-First Search

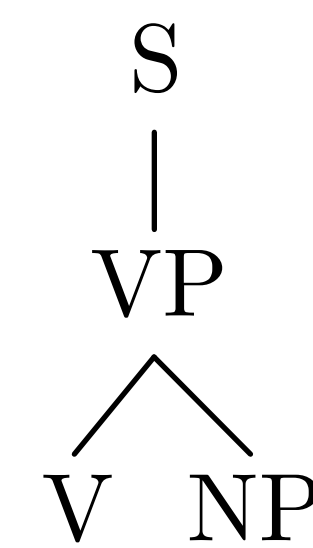
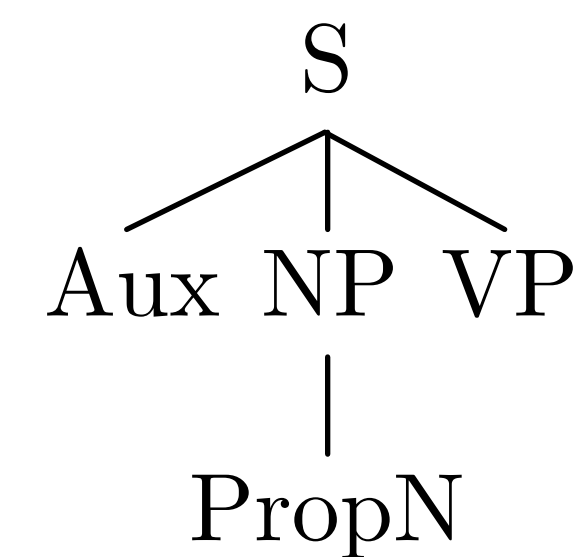
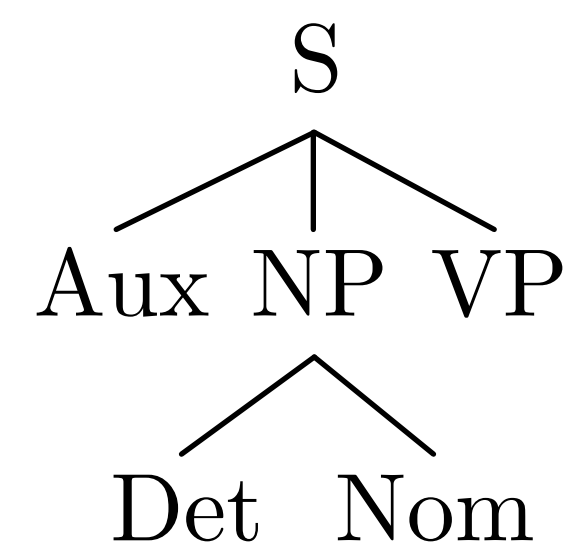
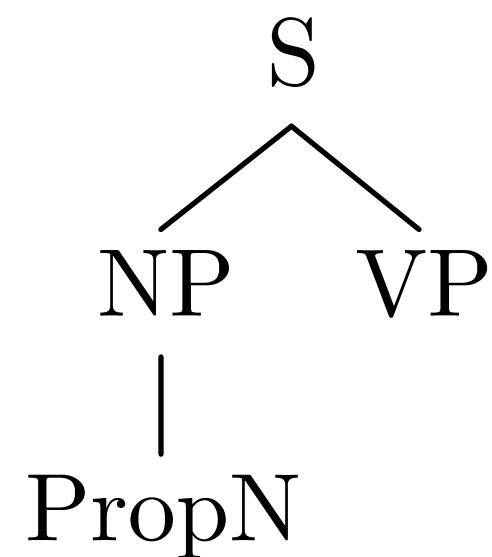
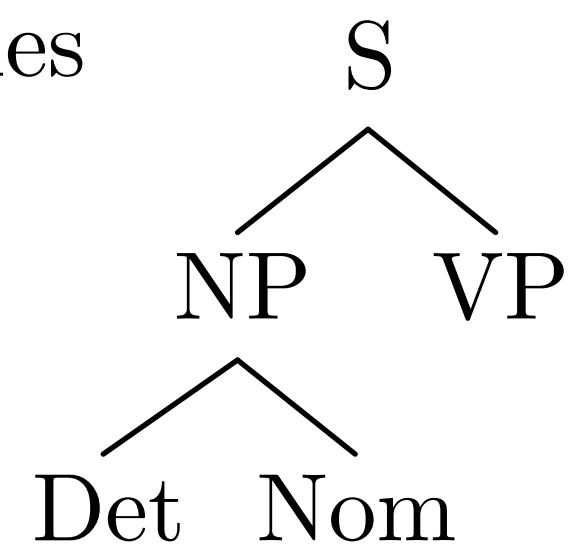
Start State

S

1 Rule



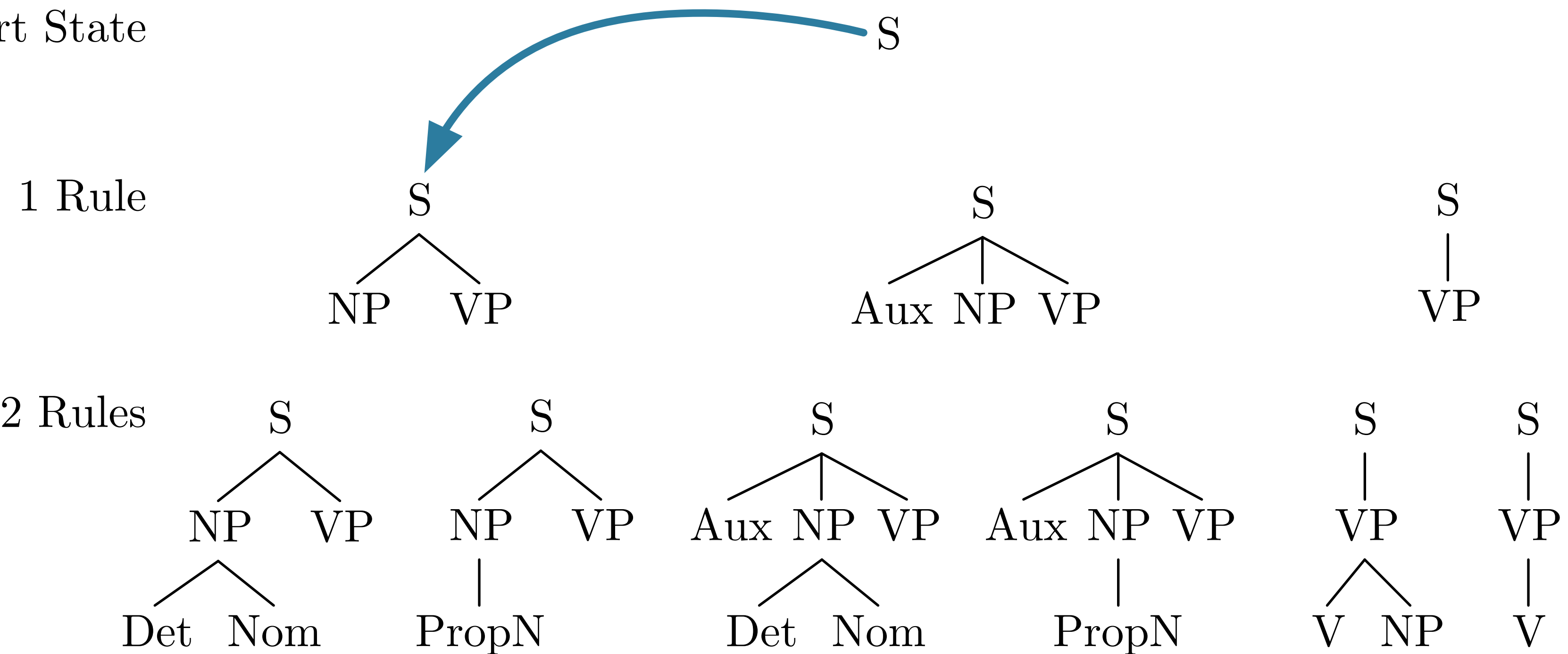
2 Rules



Book that flight.

Depth-First Search

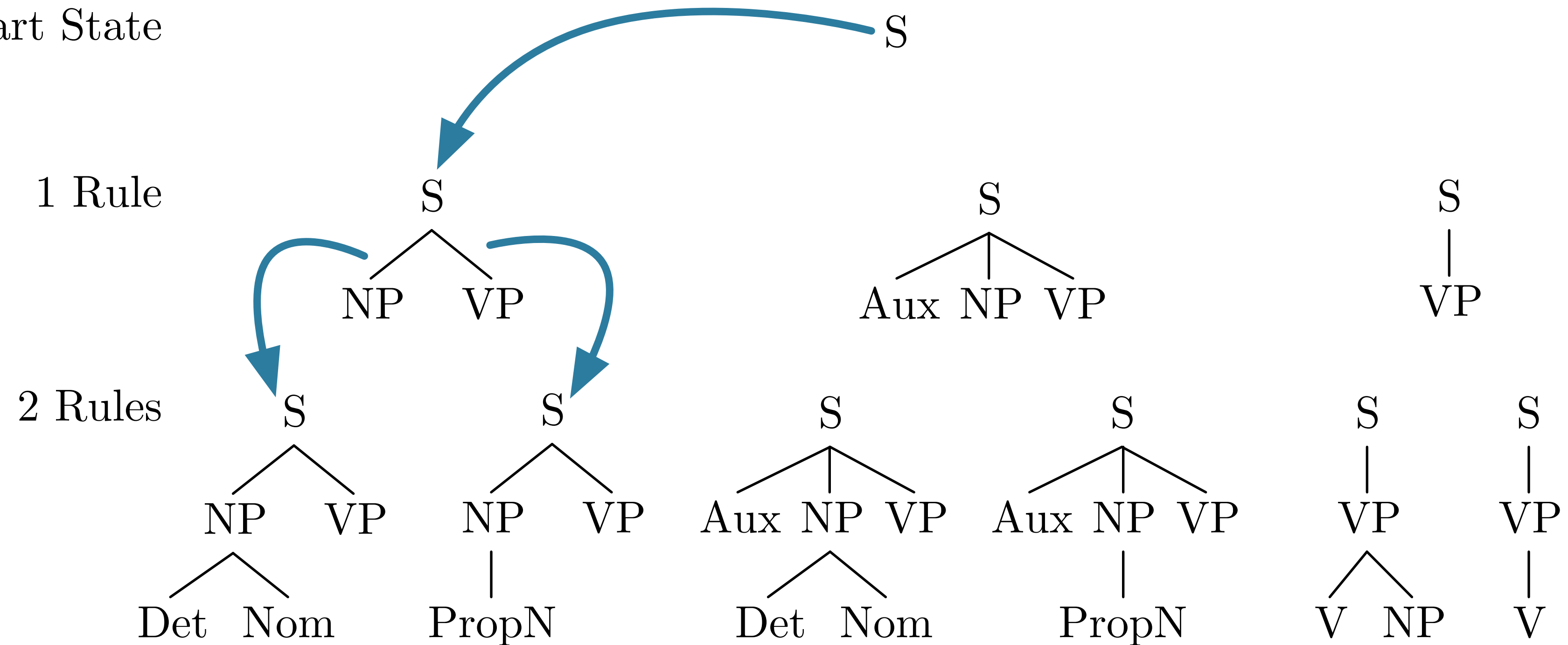
Start State



Book that flight.

Depth-First Search

Start State



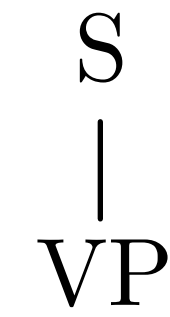
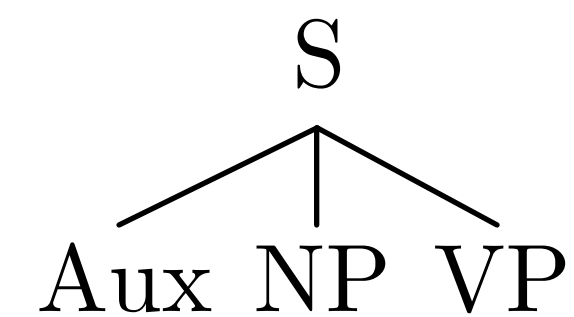
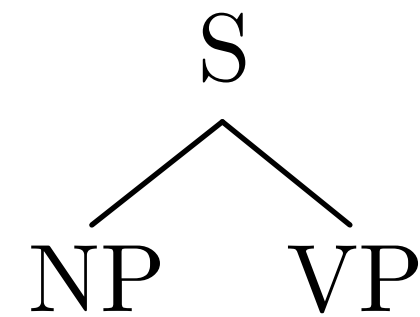
Book that flight.

Depth-First Search

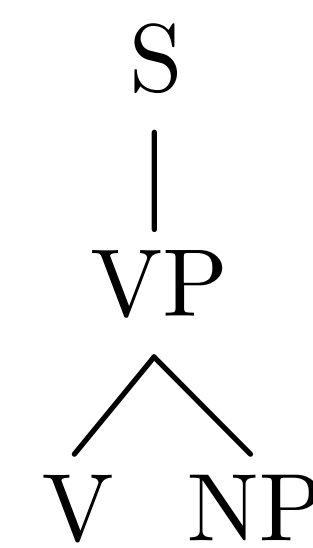
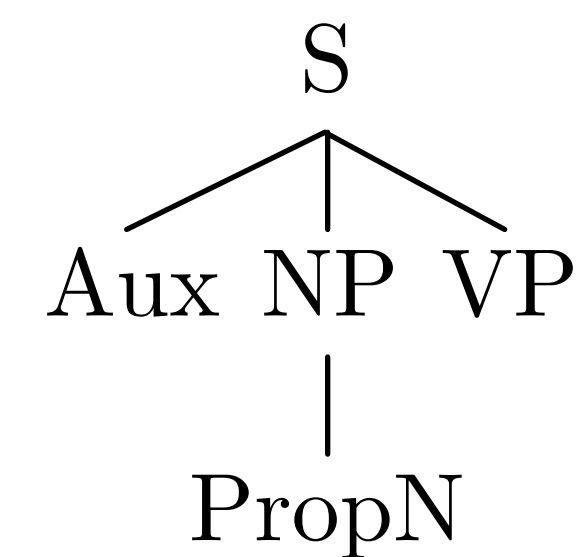
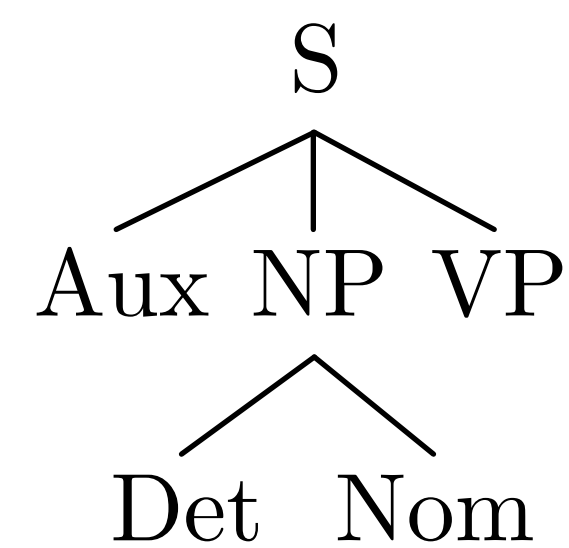
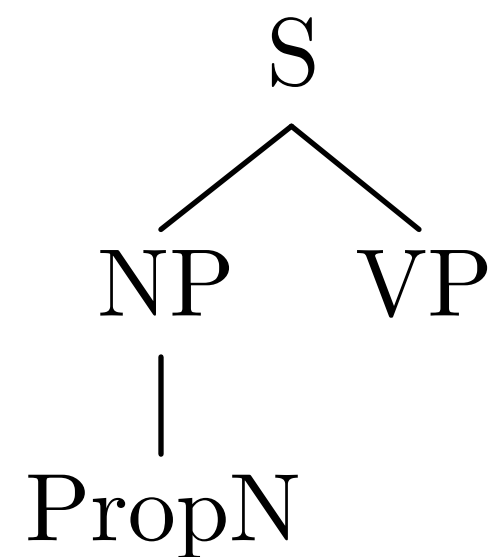
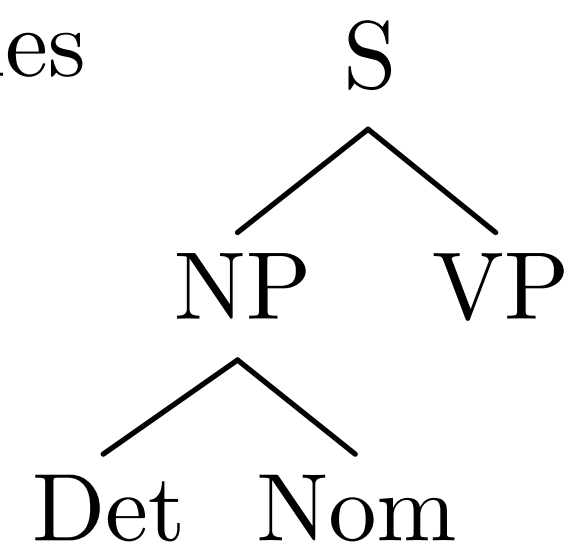
Start State

S

1 Rule



2 Rules

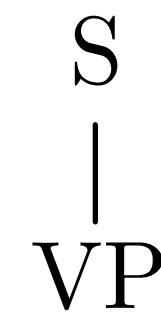
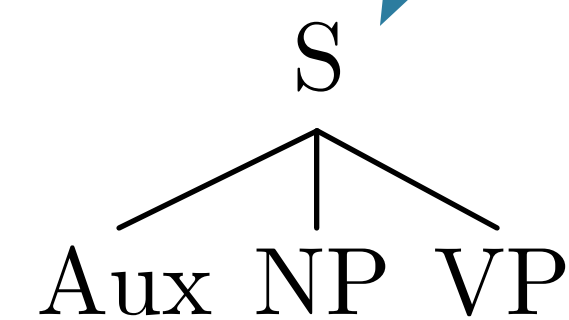
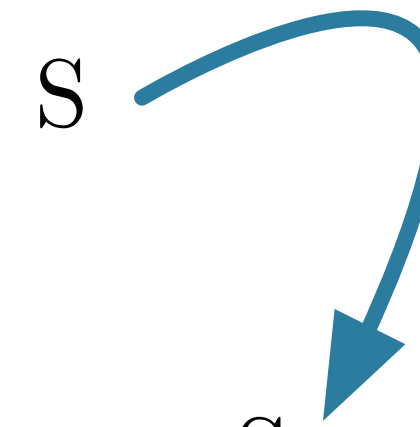
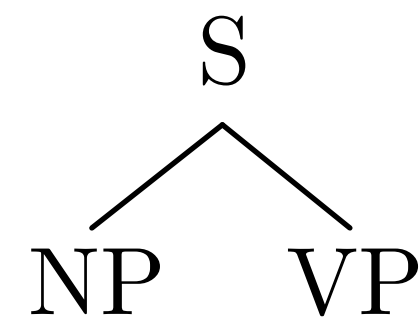


Book that flight.

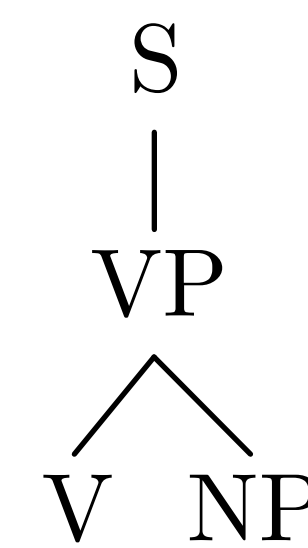
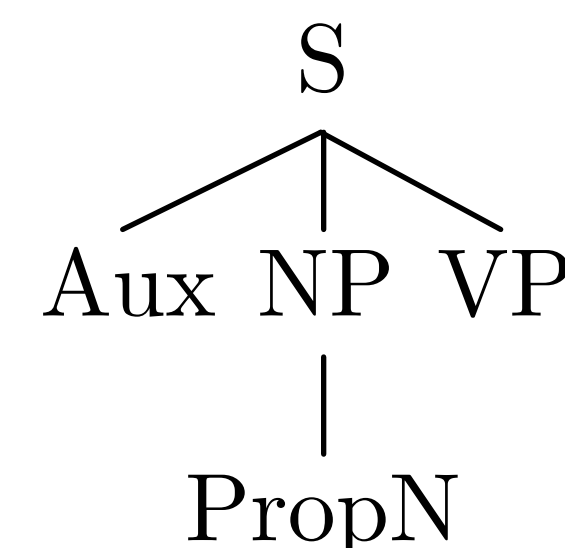
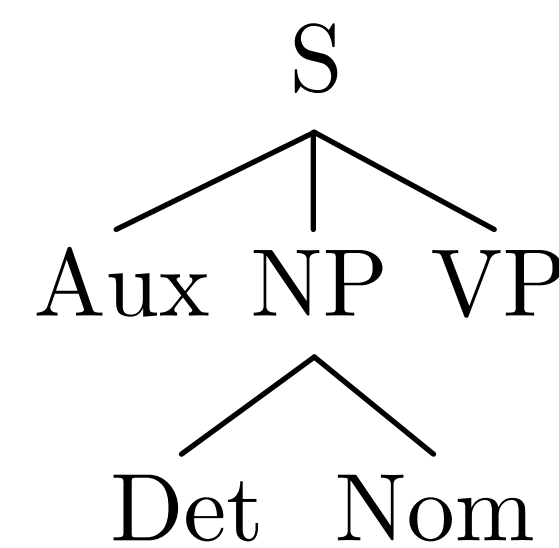
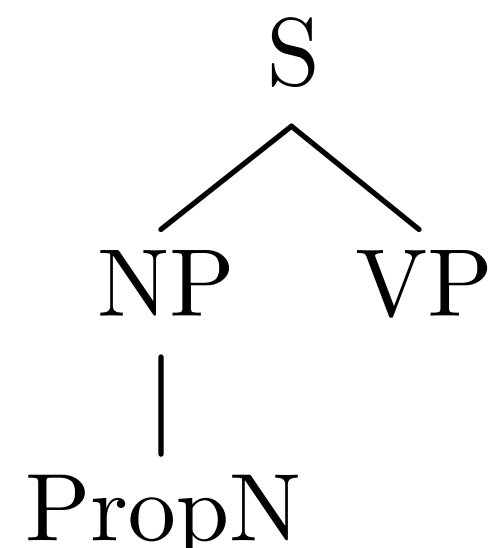
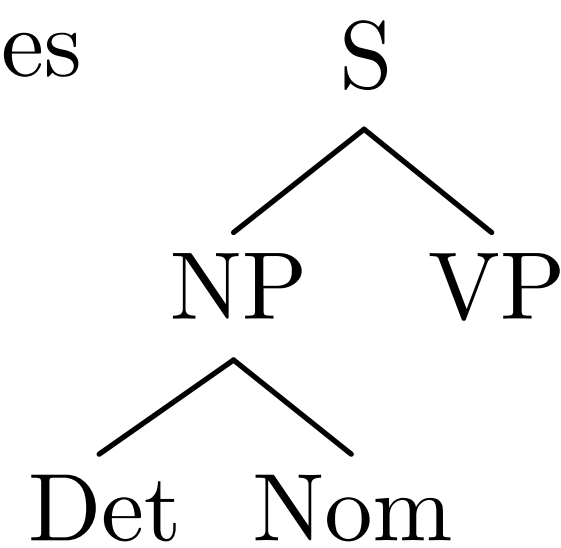
Depth-First Search

Start State

1 Rule



2 Rules

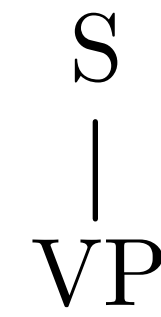
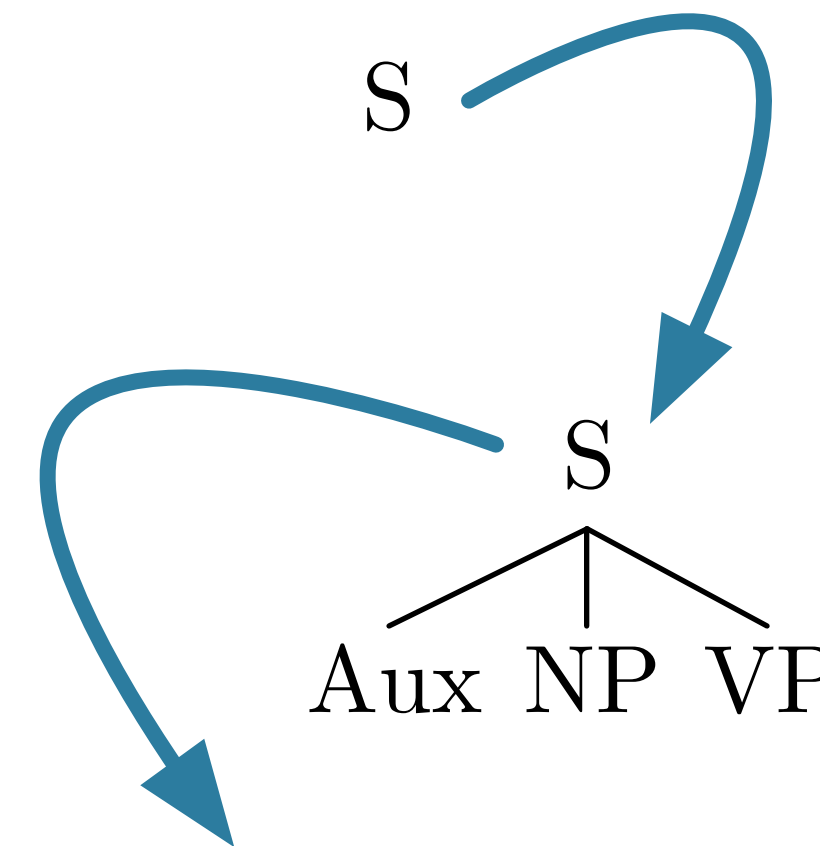
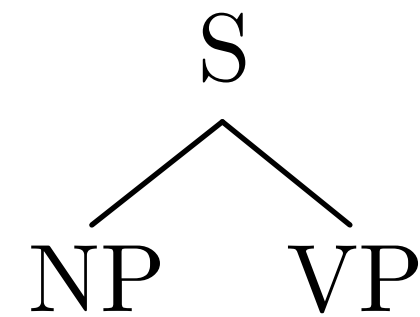


Book that flight.

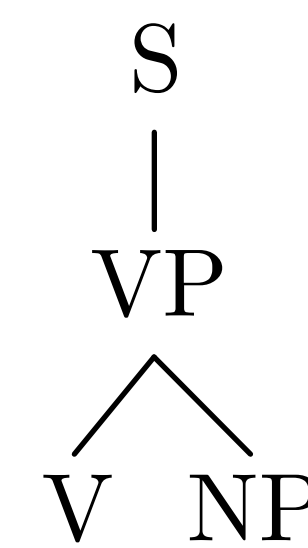
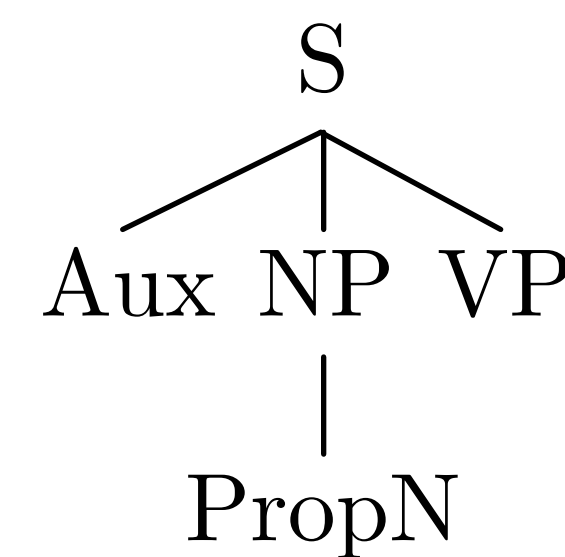
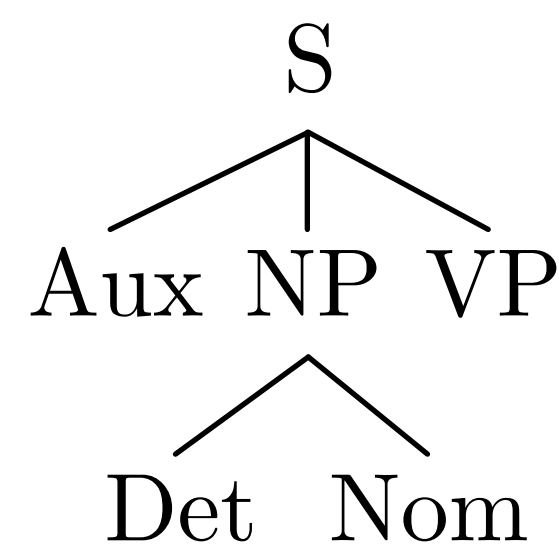
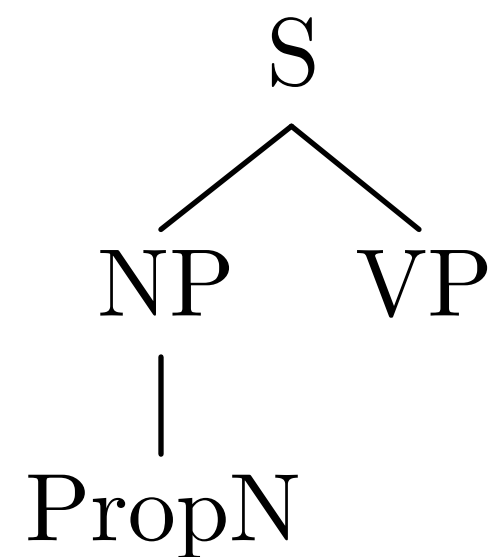
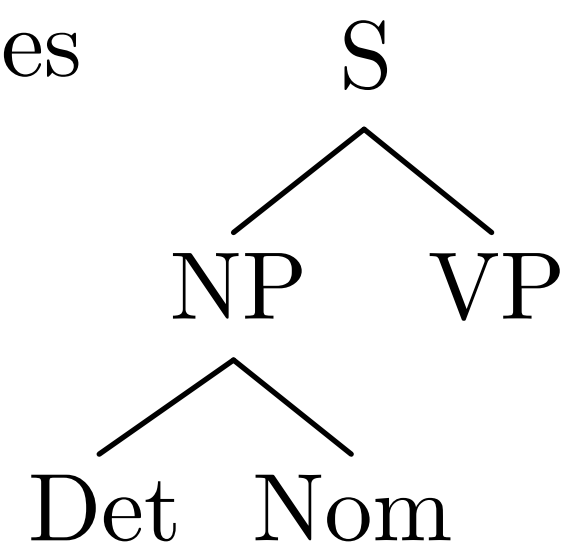
Depth-First Search

Start State

1 Rule



2 Rules

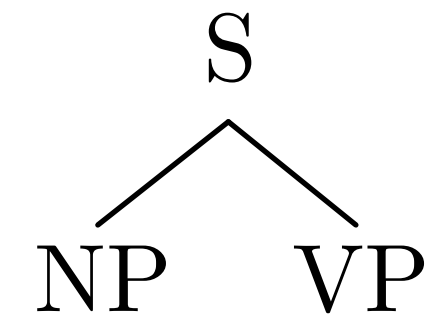


Book that flight.

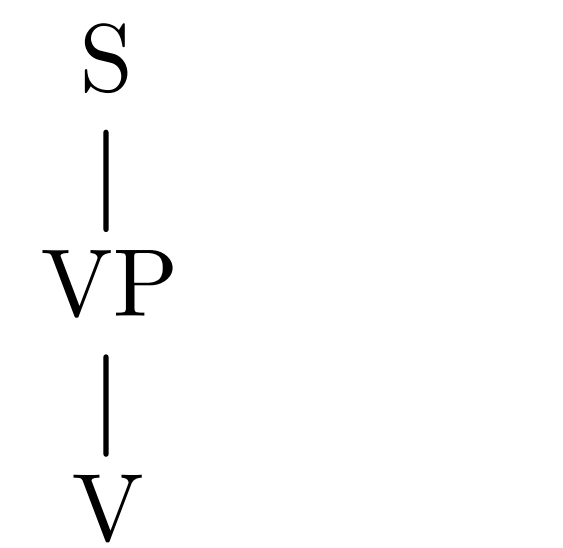
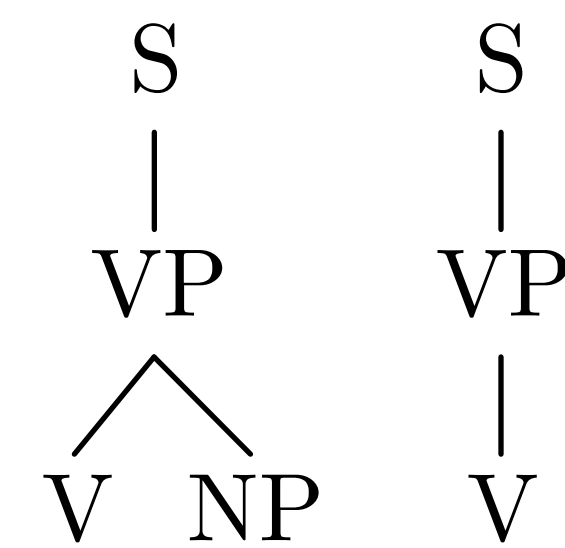
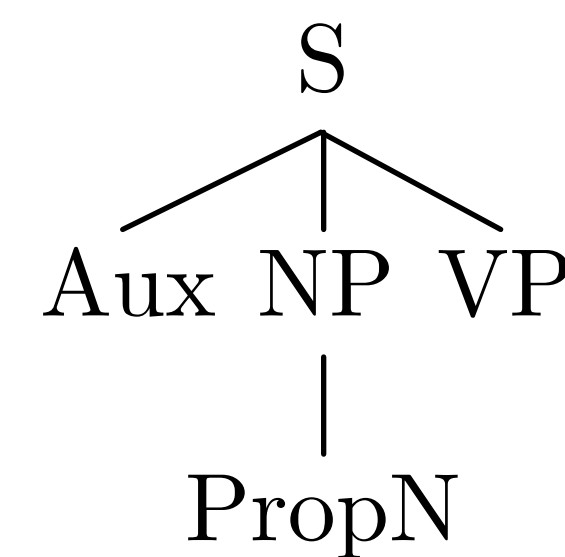
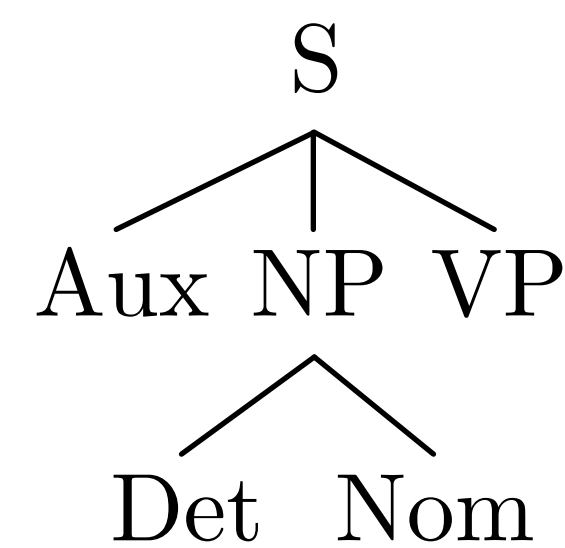
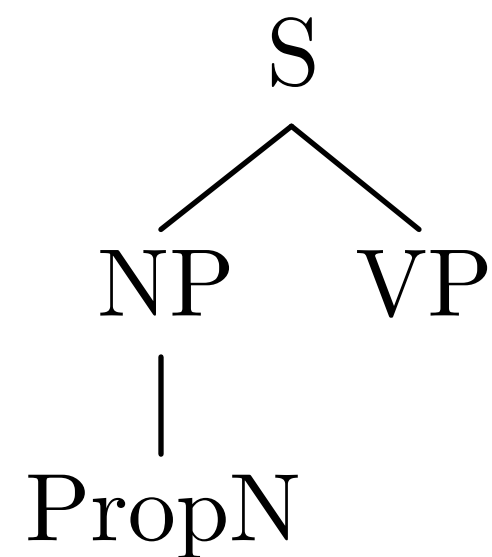
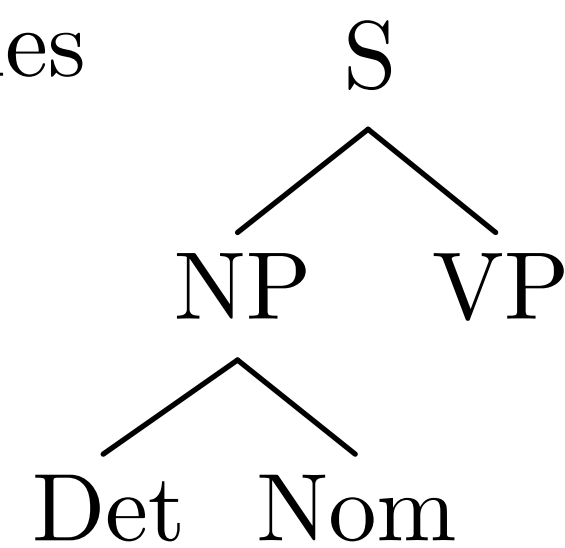
Depth-First Search

Start State

1 Rule



2 Rules



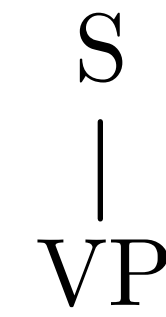
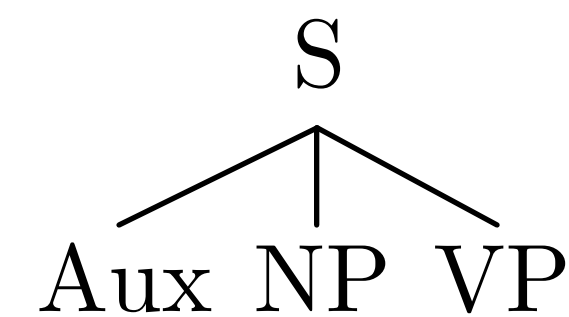
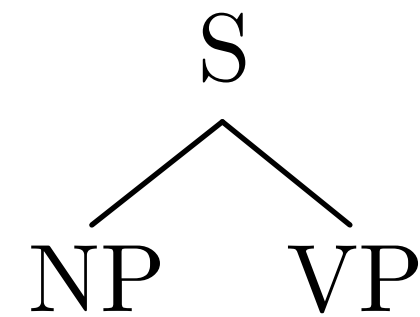
Book that flight.

Breadth-First Search

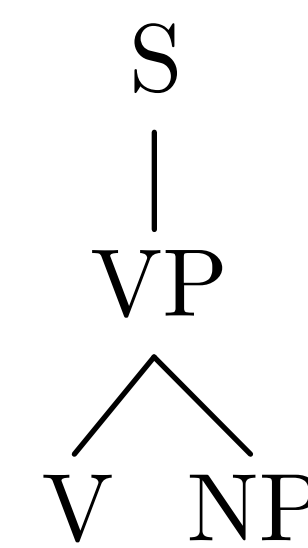
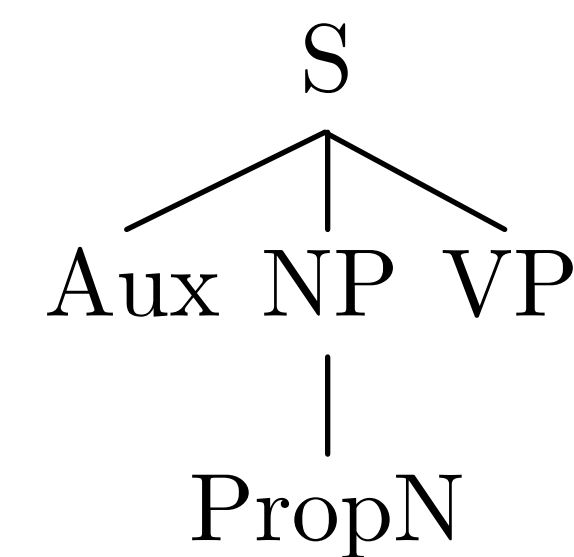
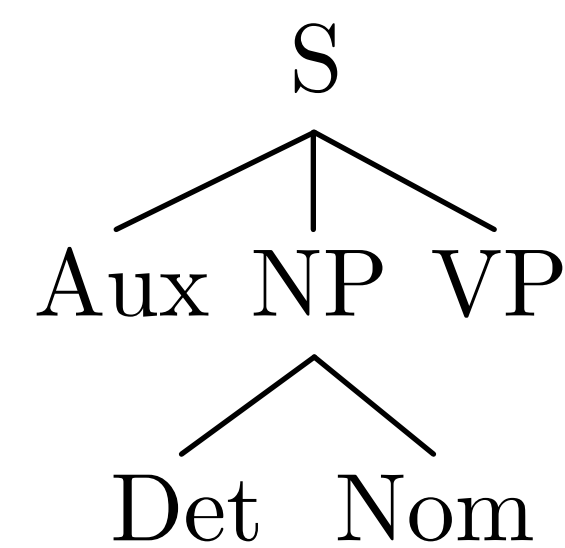
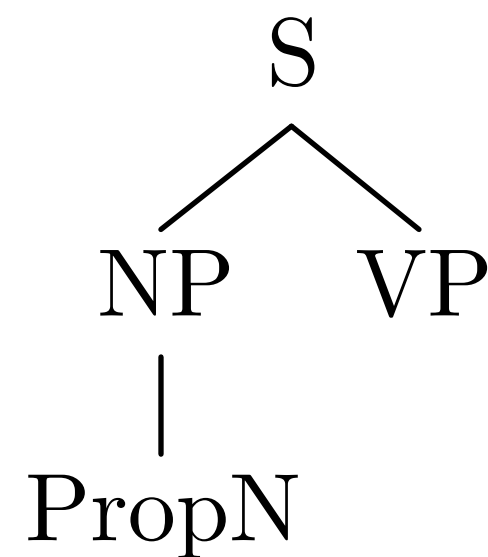
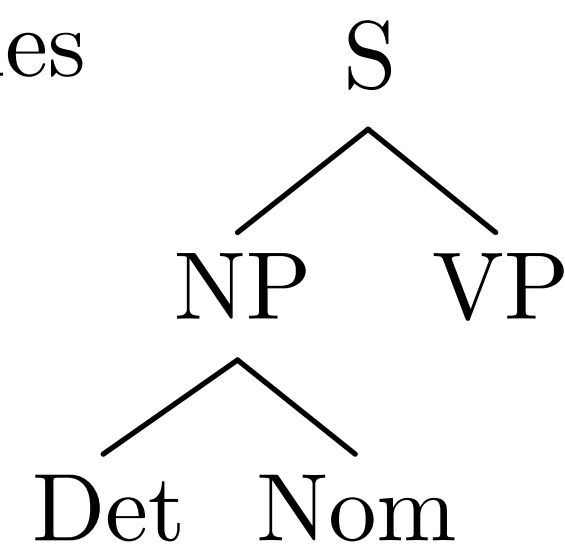
Start State

S

1 Rule



2 Rules

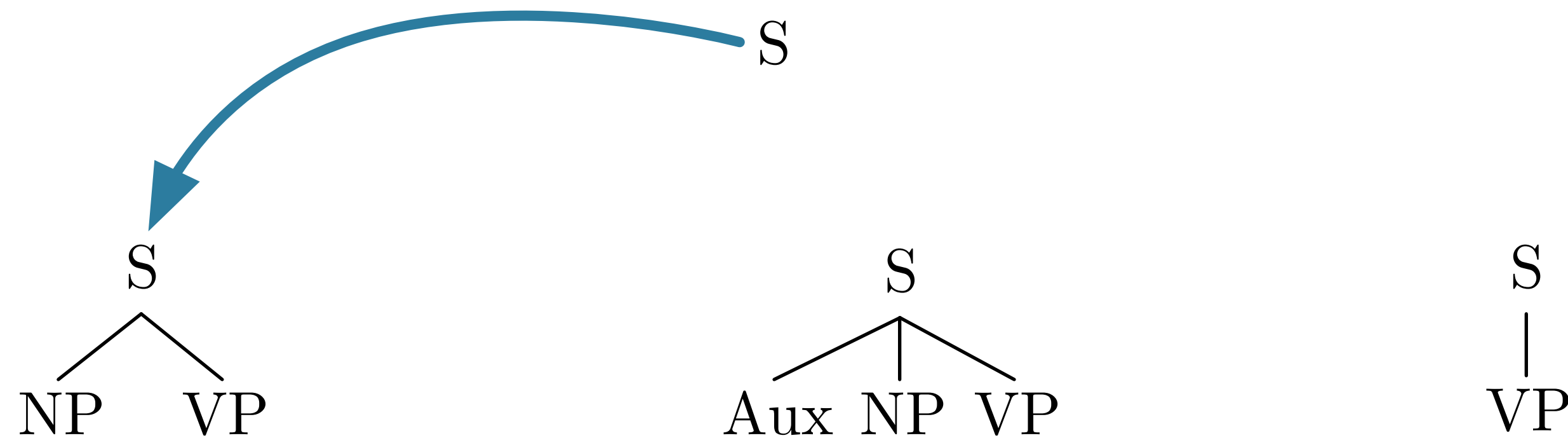


Book that flight.

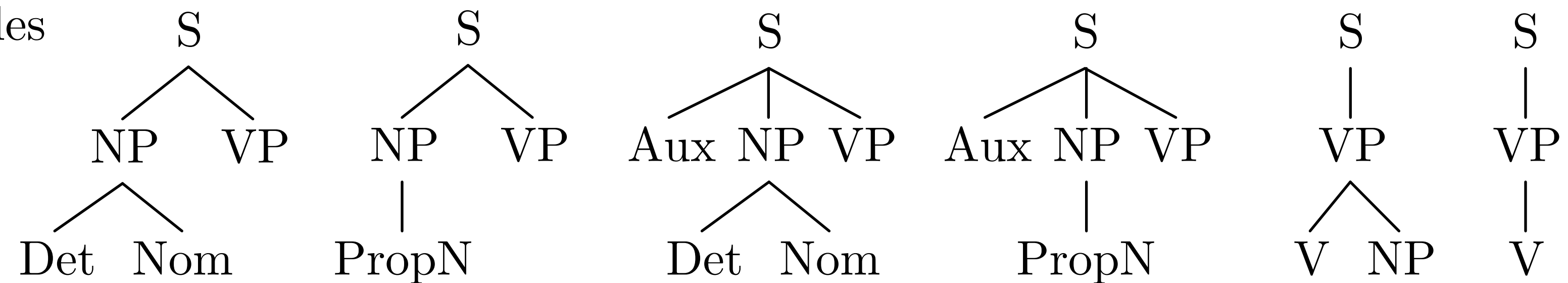
Breadth-First Search

Start State

1 Rule



2 Rules

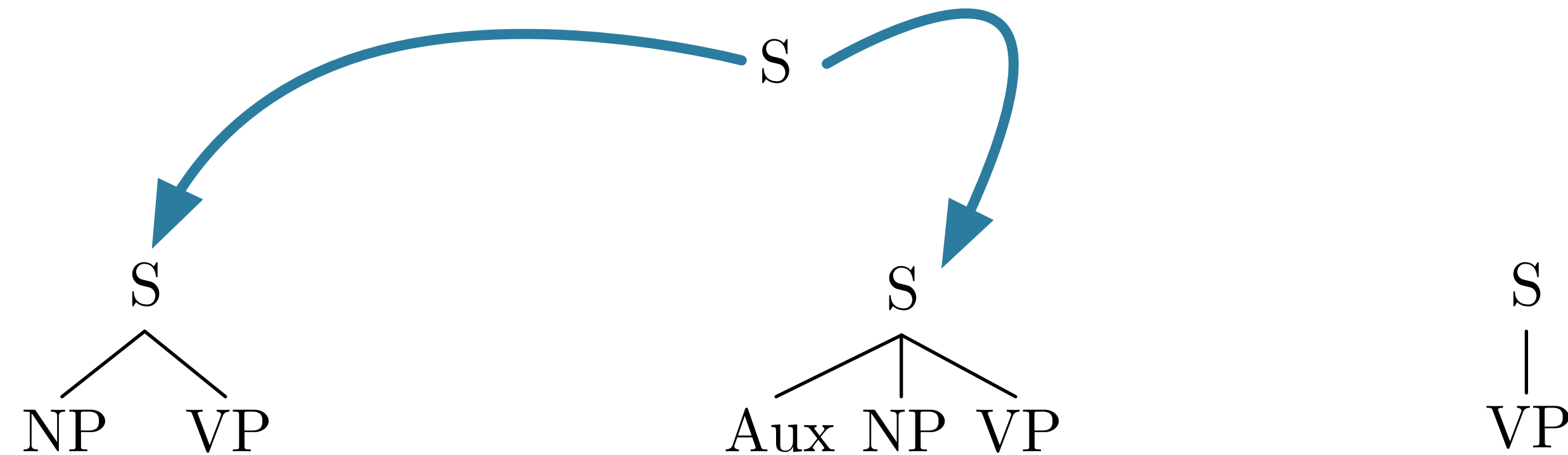


Book that flight.

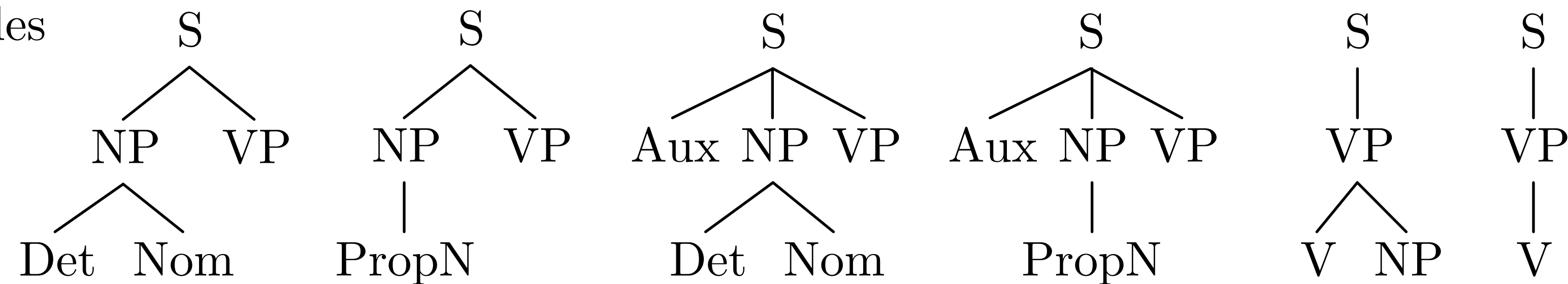
Breadth-First Search

Start State

1 Rule



2 Rules

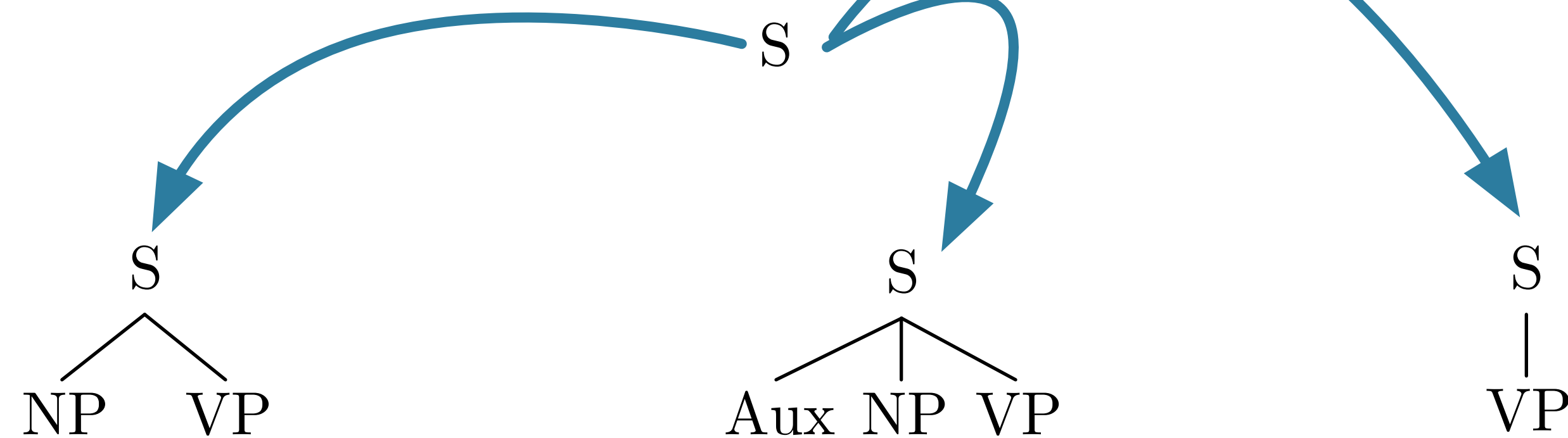


Book that flight.

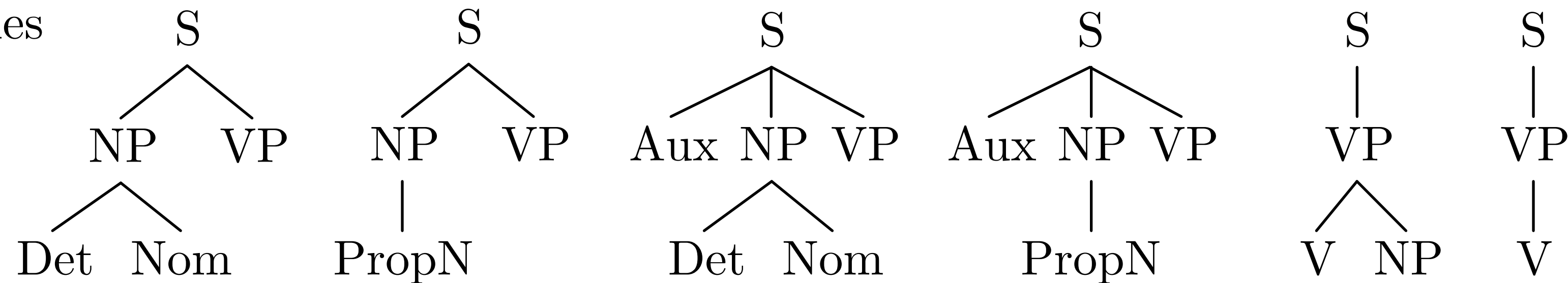
Breadth-First Search

Start State

1 Rule



2 Rules



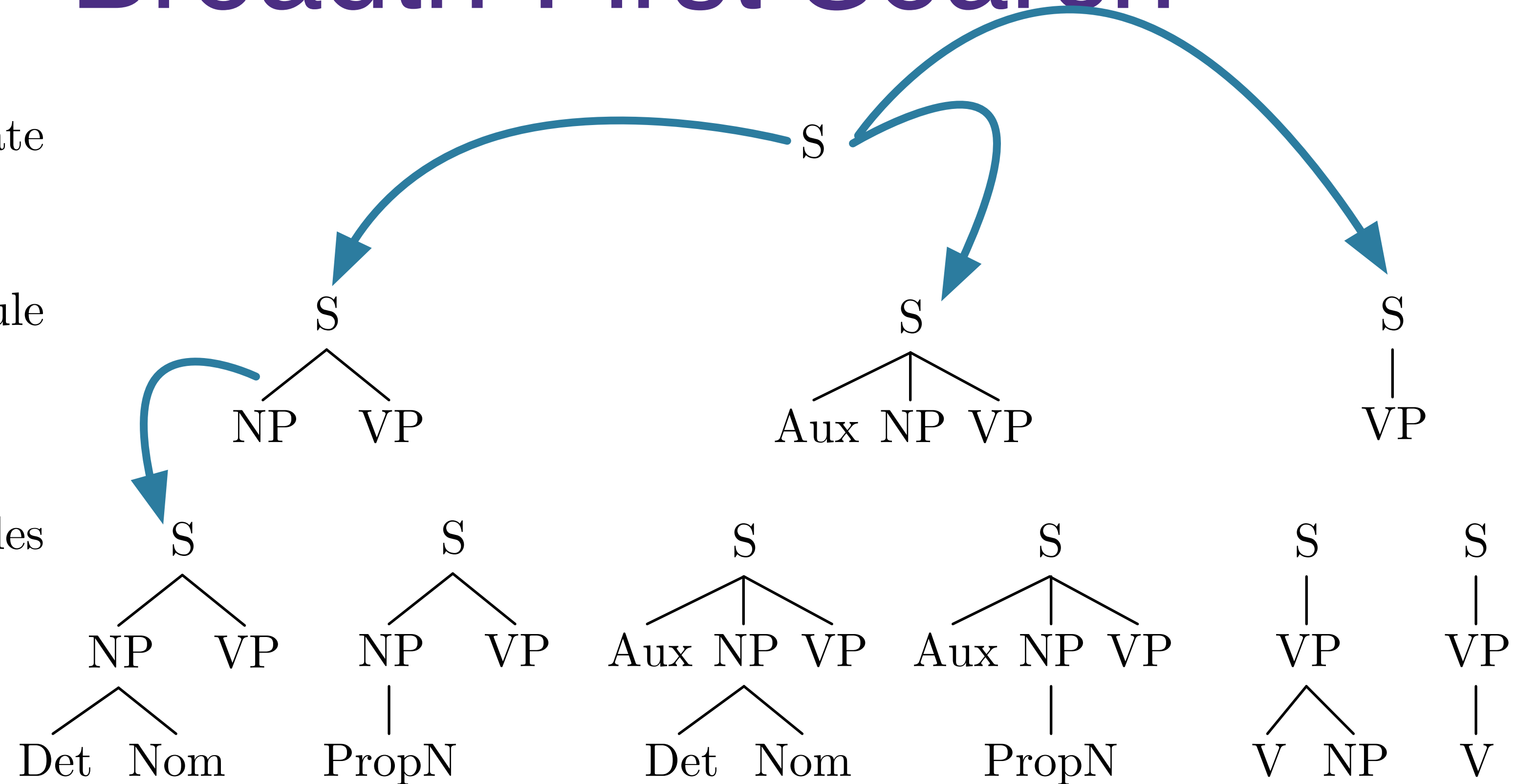
Book that flight.

Breadth-First Search

Start State

1 Rule

2 Rules



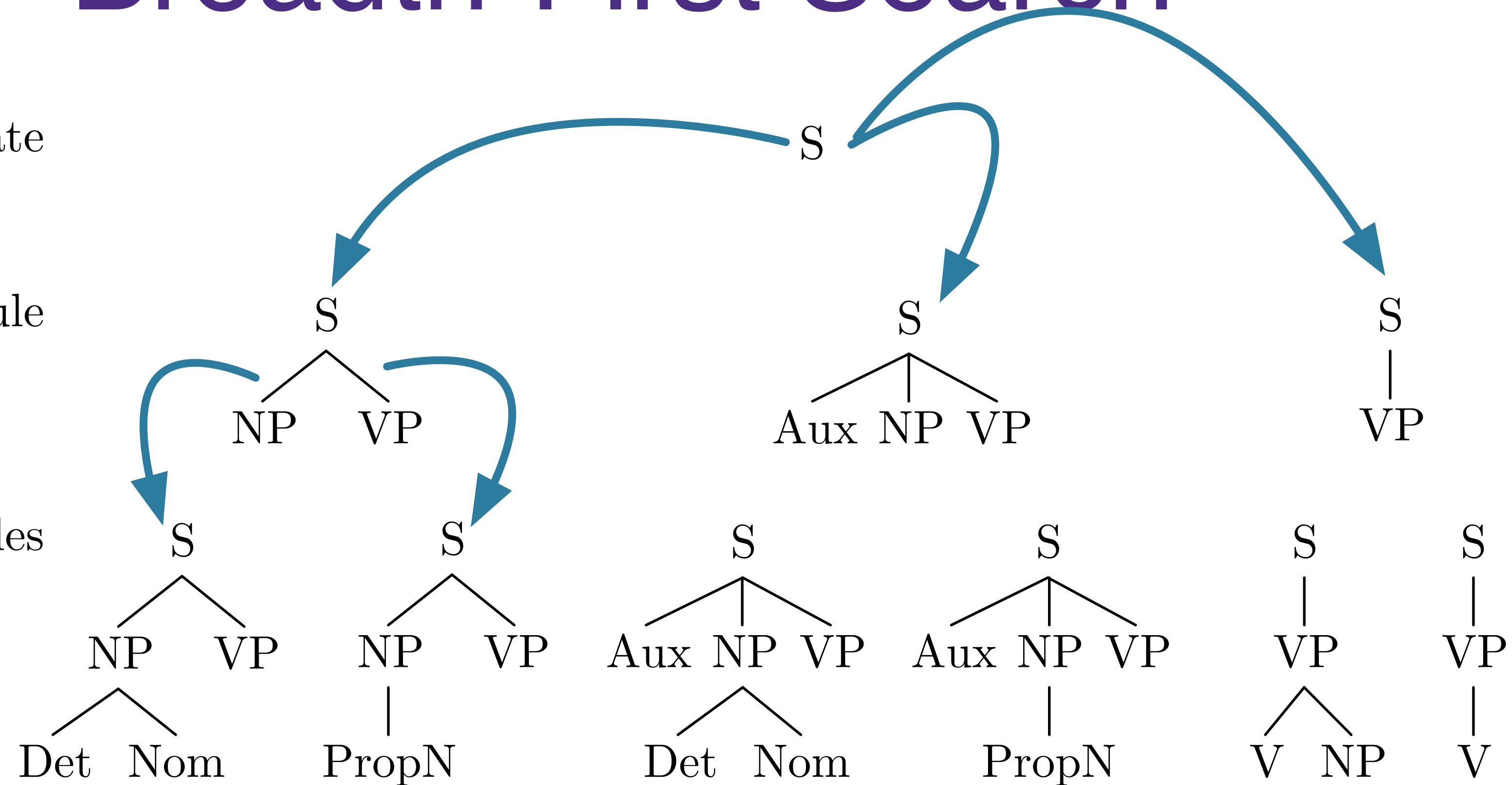
Book that flight.

Breadth-First Search

Start State

1 Rule

2 Rules



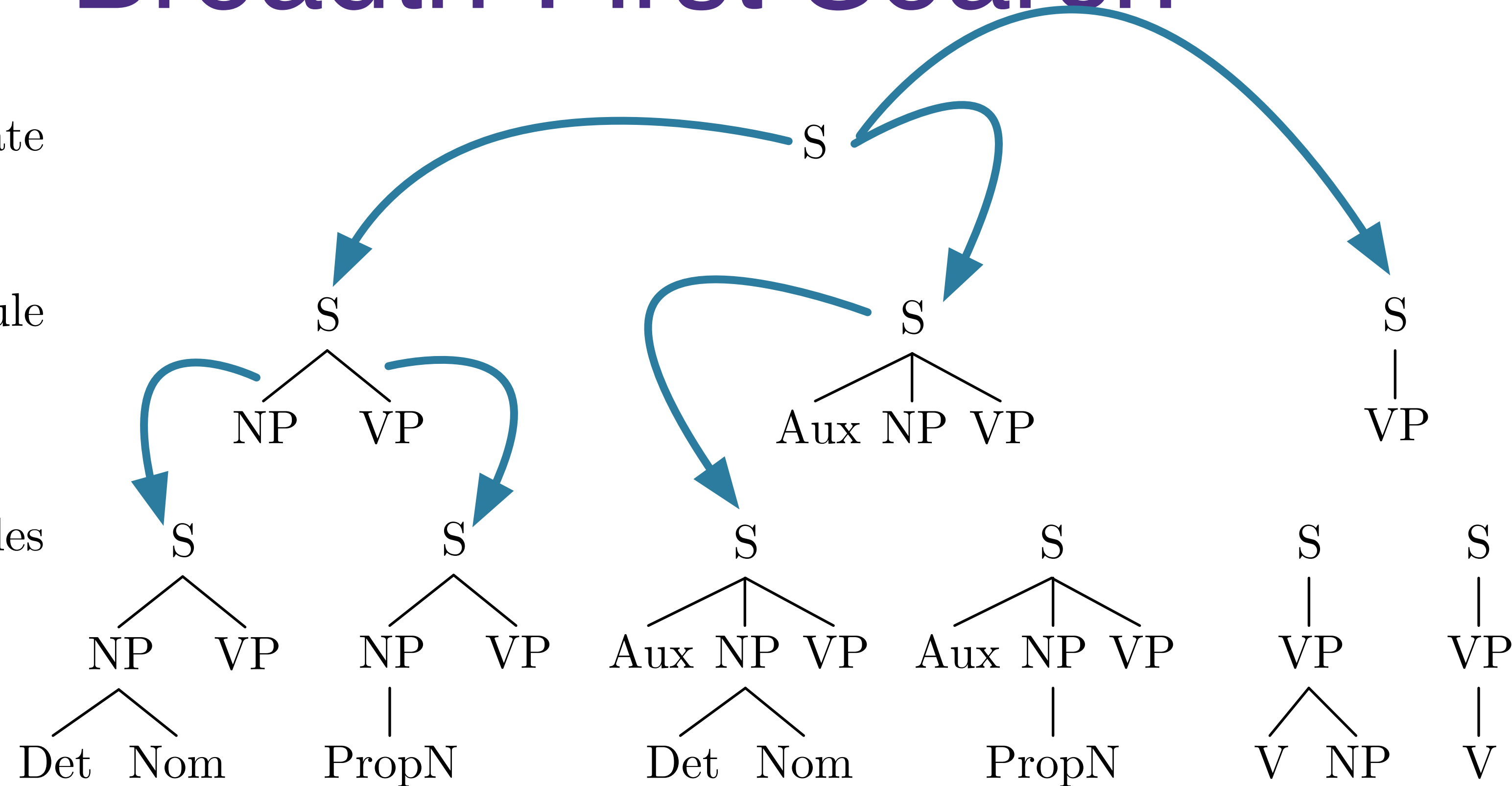
Book that flight.

Breadth-First Search

Start State

1 Rule

2 Rules



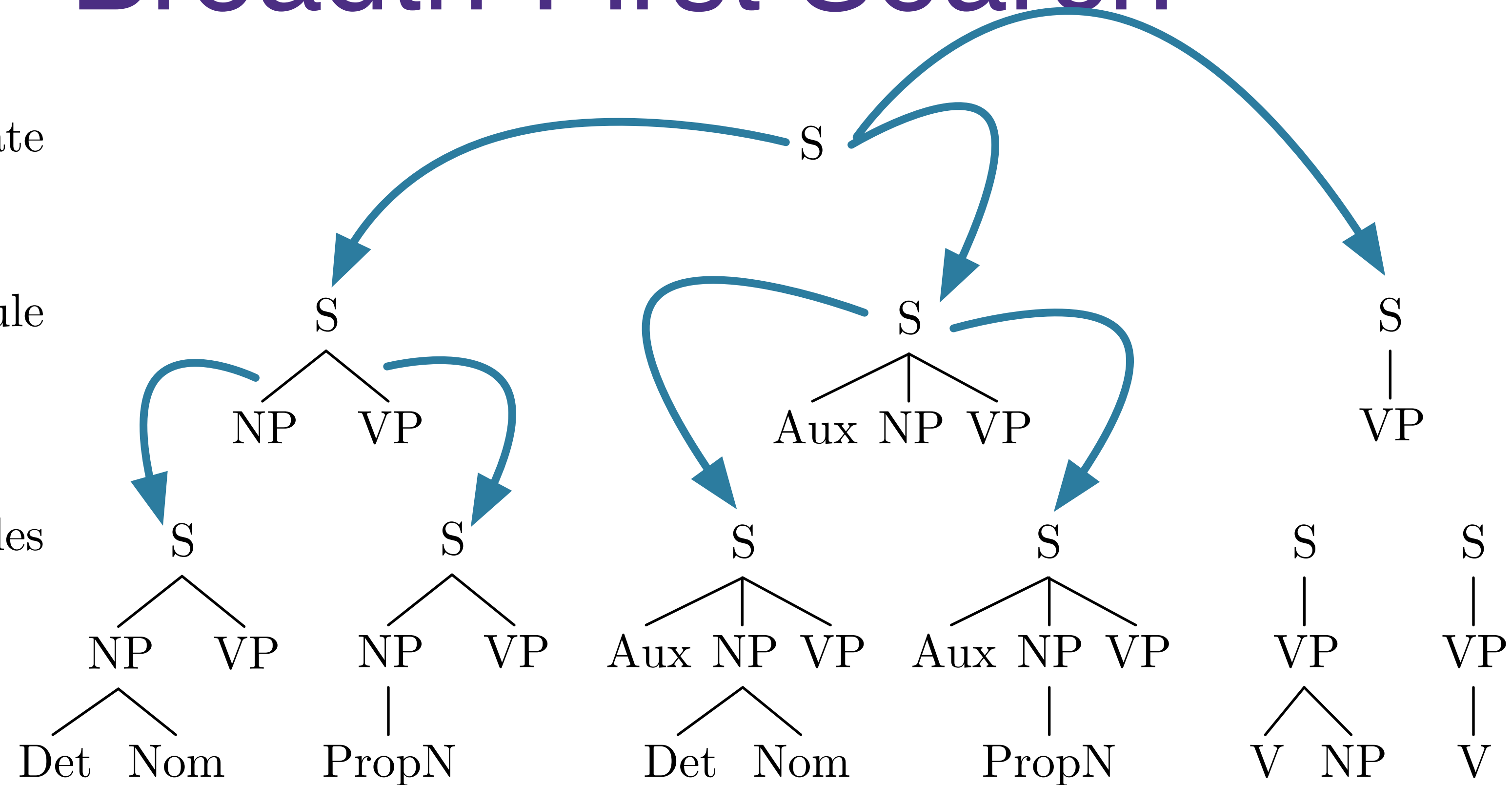
Book that flight.

Breadth-First Search

Start State

1 Rule

2 Rules



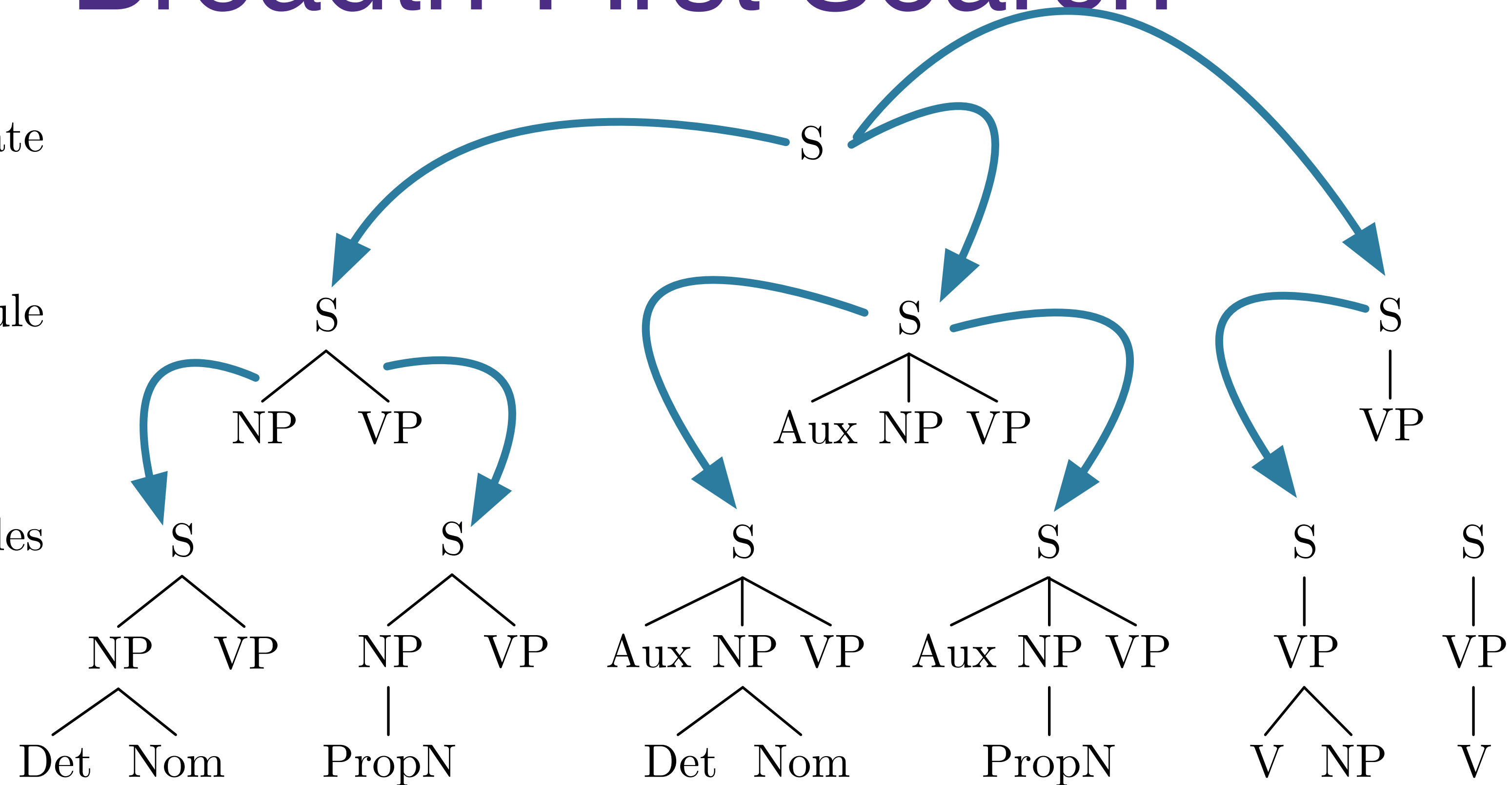
Book that flight.

Breadth-First Search

Start State

1 Rule

2 Rules



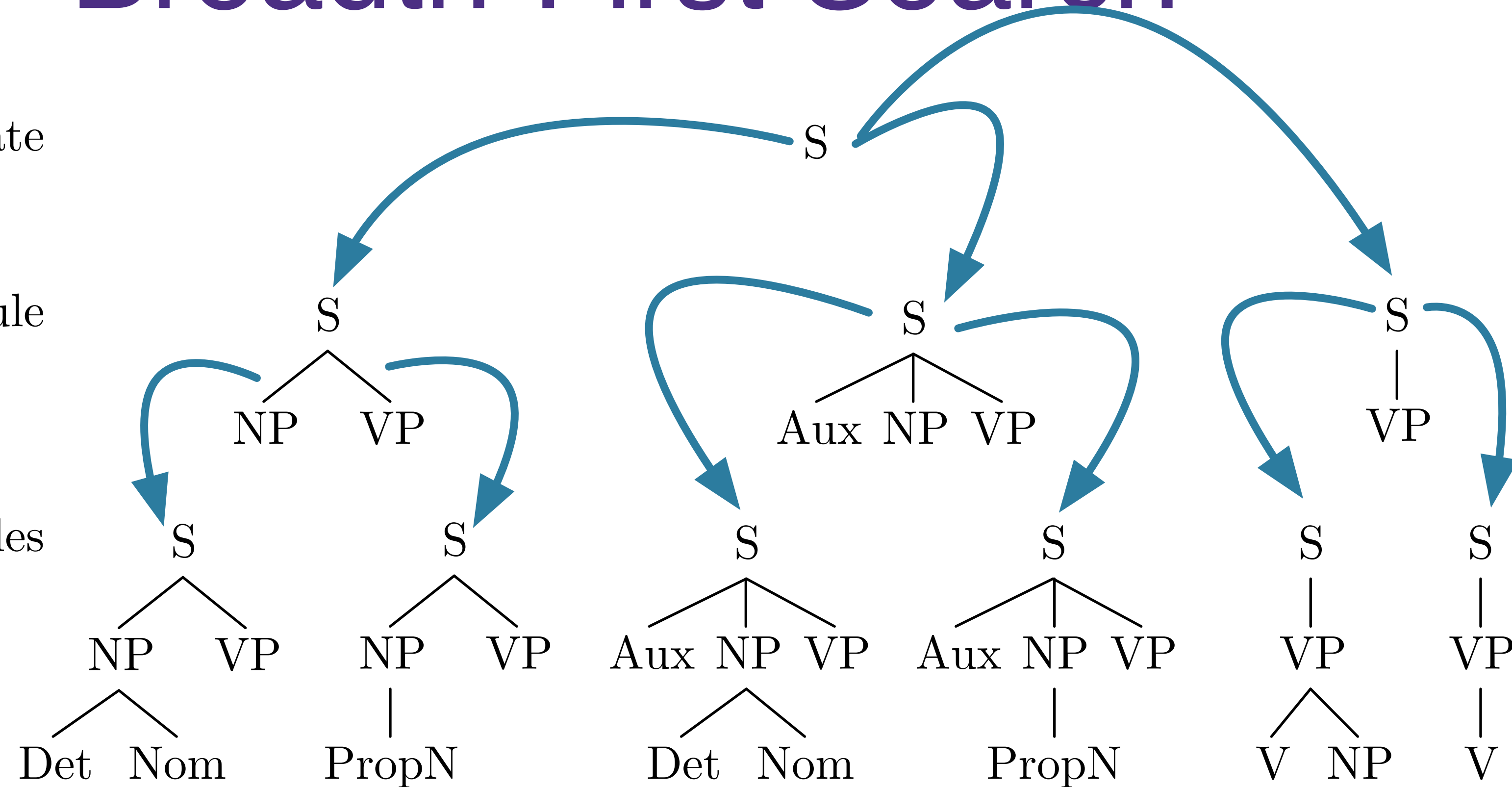
Book that flight.

Breadth-First Search

Start State

1 Rule

2 Rules



Book that flight.

Pros and Cons of Top-down Parsing

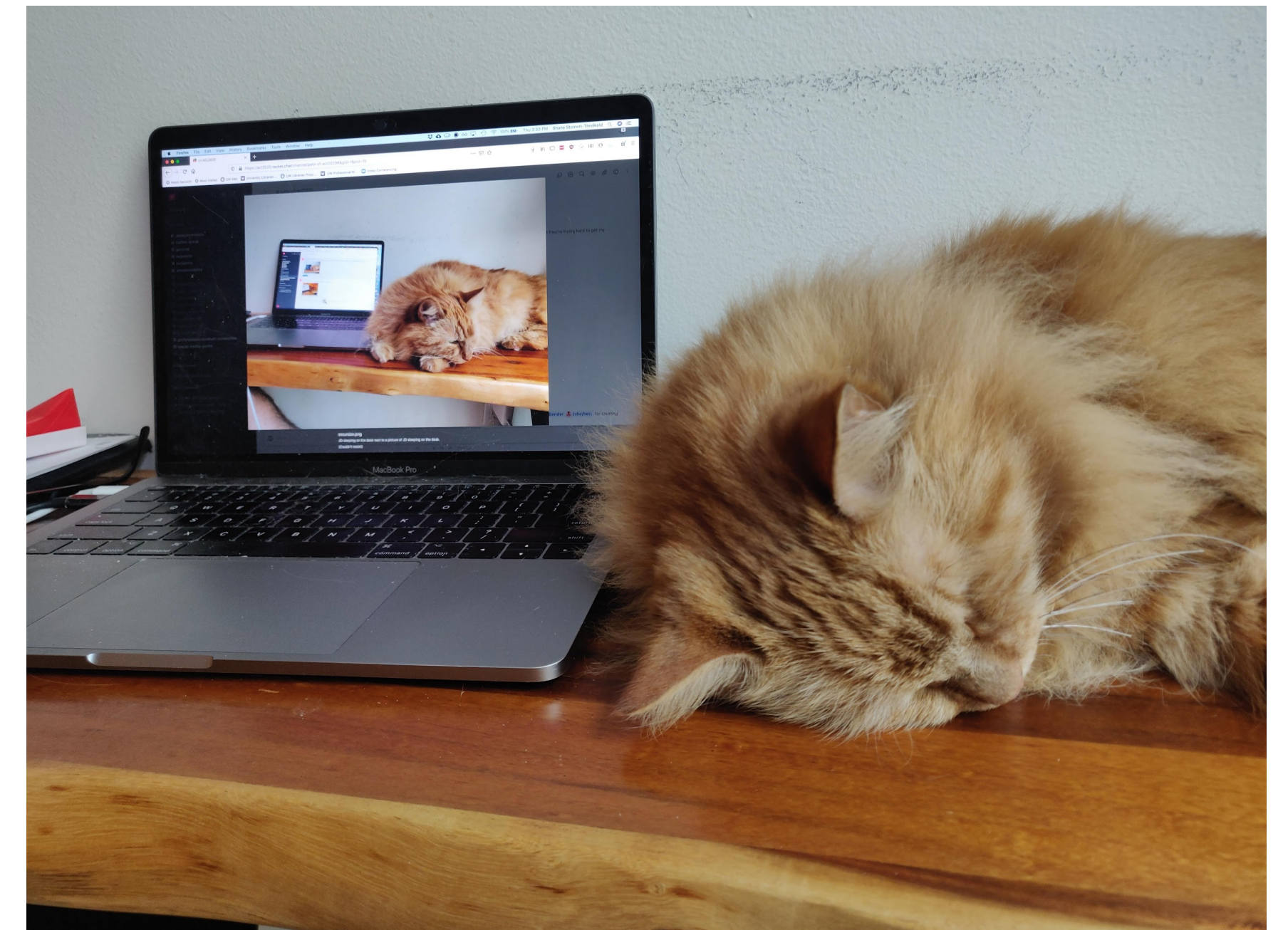
- Pros:
 - Doesn't explore trees not rooted at S
 - Doesn't explore subtrees that don't fit valid trees

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
 - May not terminate in presence of recursive rules
 - May re-derive subtrees as part of 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 re-derive subtrees as part of search



Bottom-Up Parsing

Bottom-Up Parsing

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

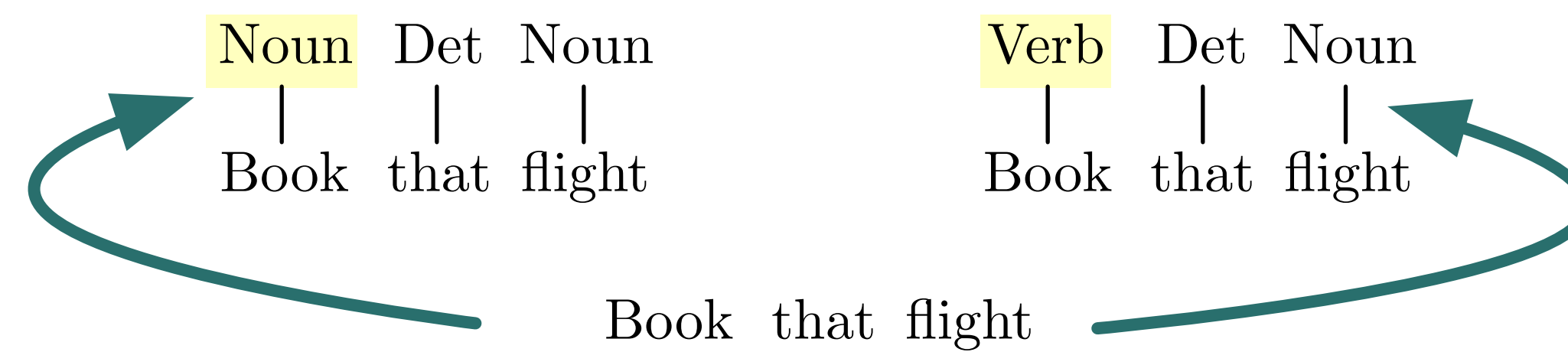
Bottom-Up Parsing

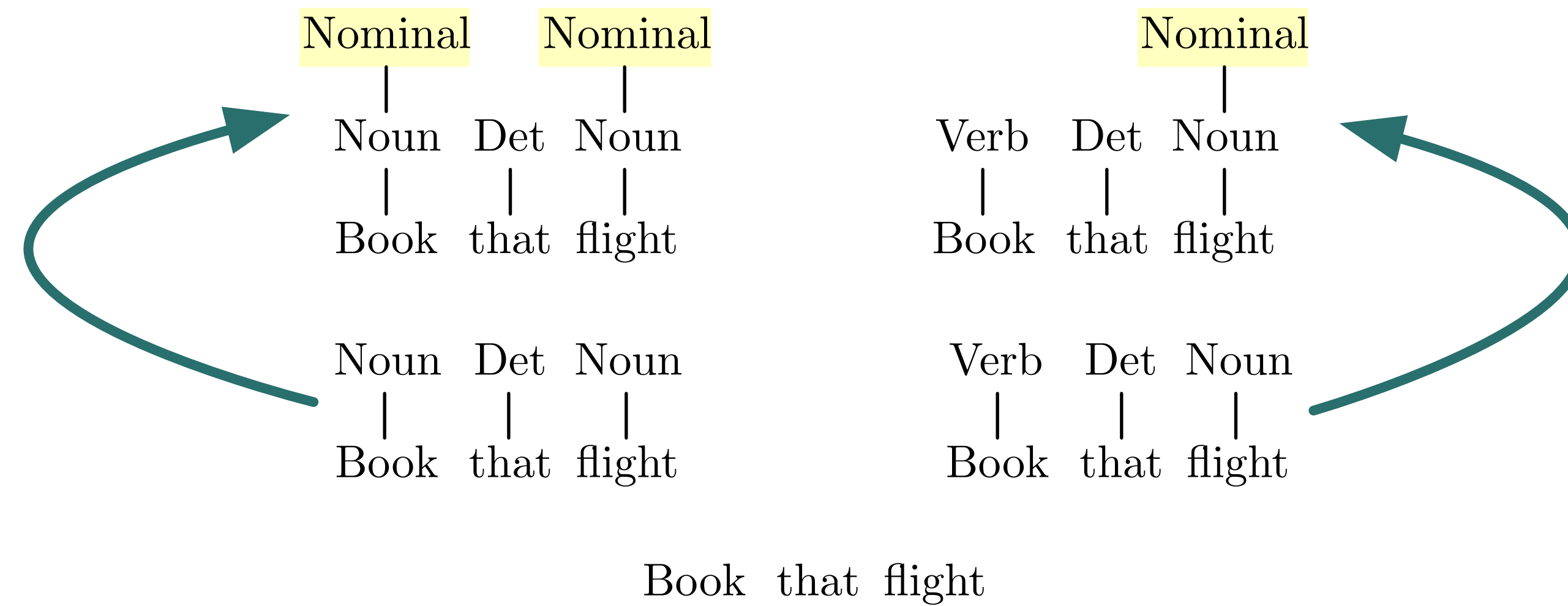
- Try to find all trees that span the input
 - Start with input string
 - Book that flight
- Use all productions with current subtree(s) on RHS
 - e.g. $N \rightarrow \text{Book}$; $V \rightarrow \text{Book}$

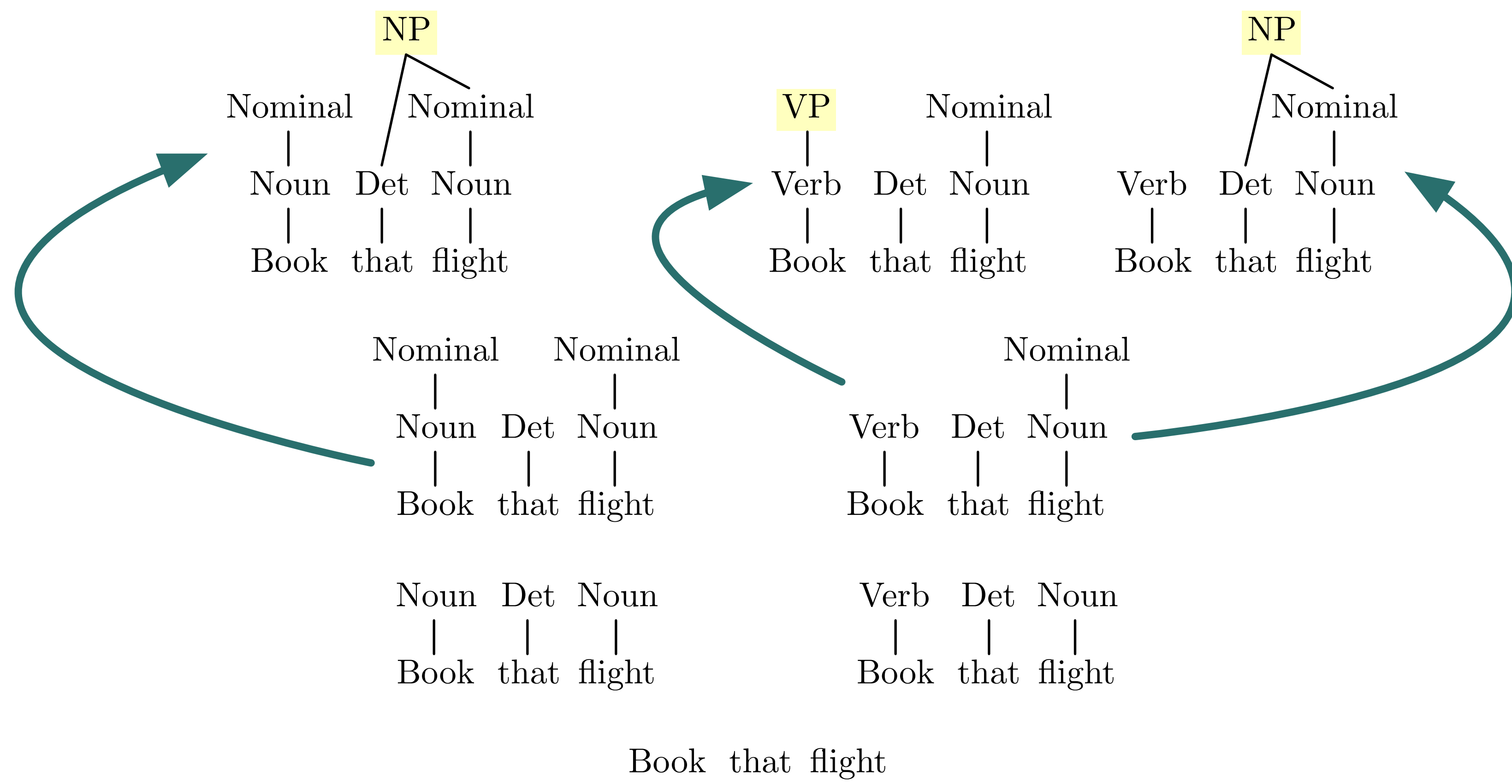
Bottom-Up Parsing

- 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. $N \rightarrow \text{Book}$; $V \rightarrow \text{Book}$
- Stop when spanned by S, or no more rules apply

Book that flight







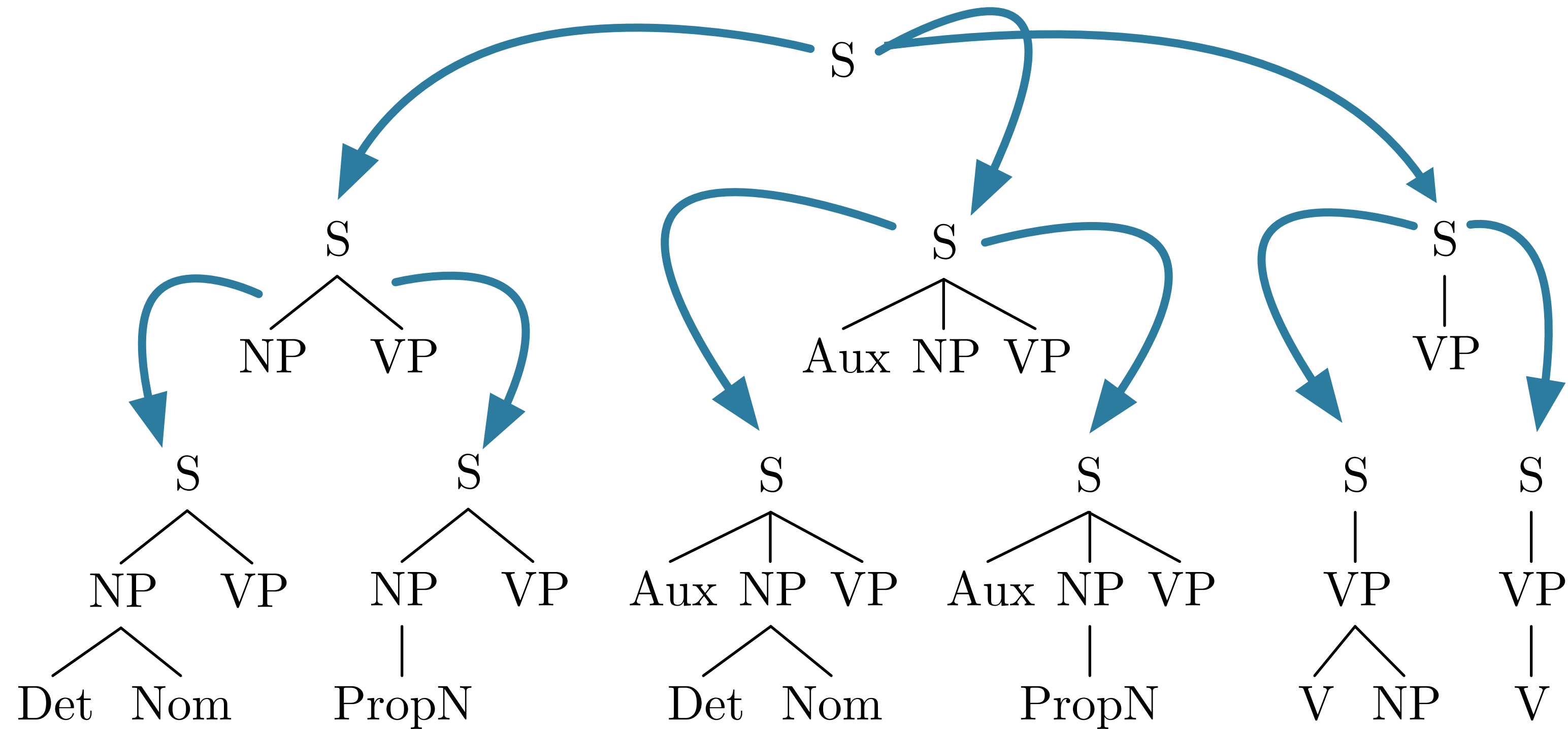
Pros and Cons of Bottom-Up Search

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

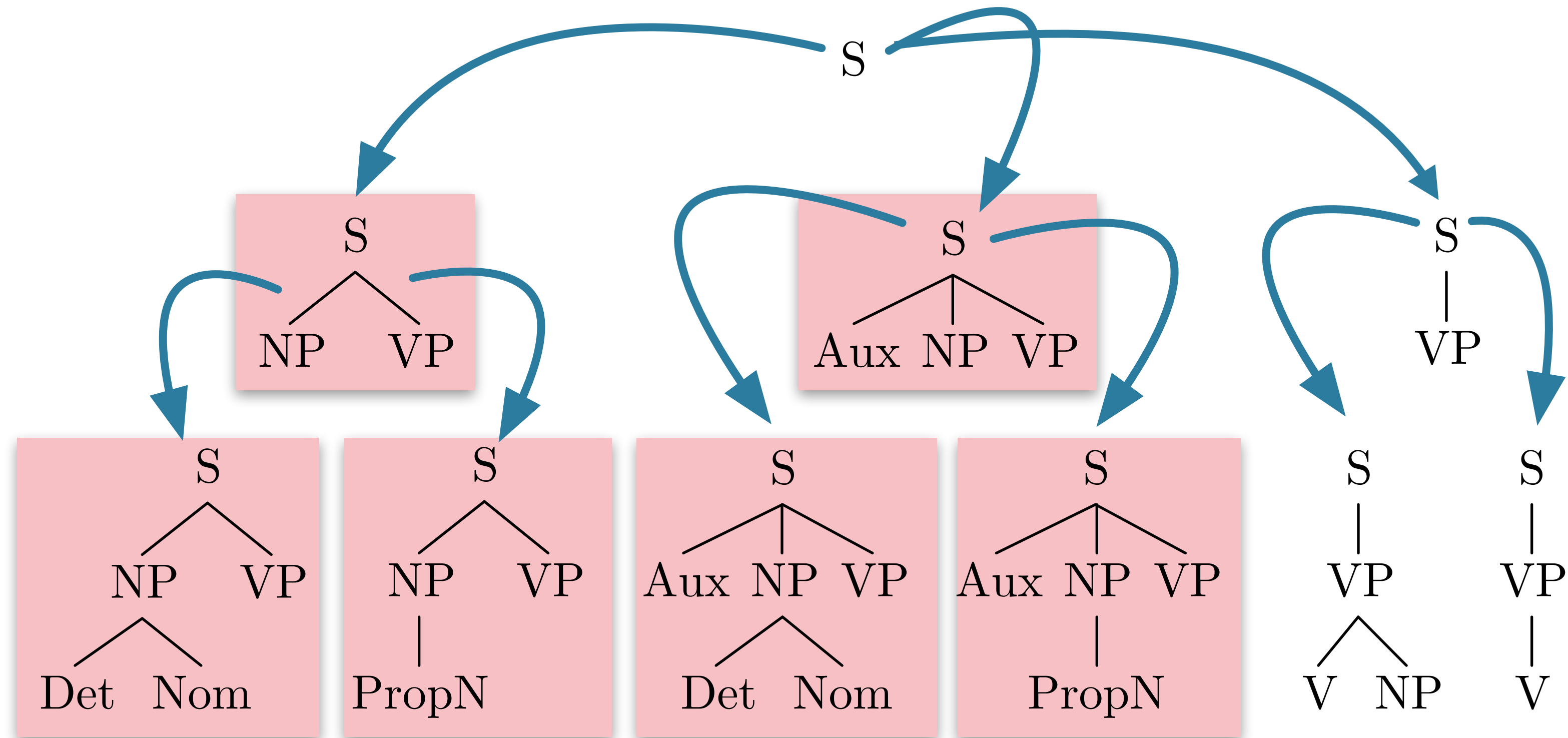
Pros and Cons of Bottom-Up Search

- 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 input

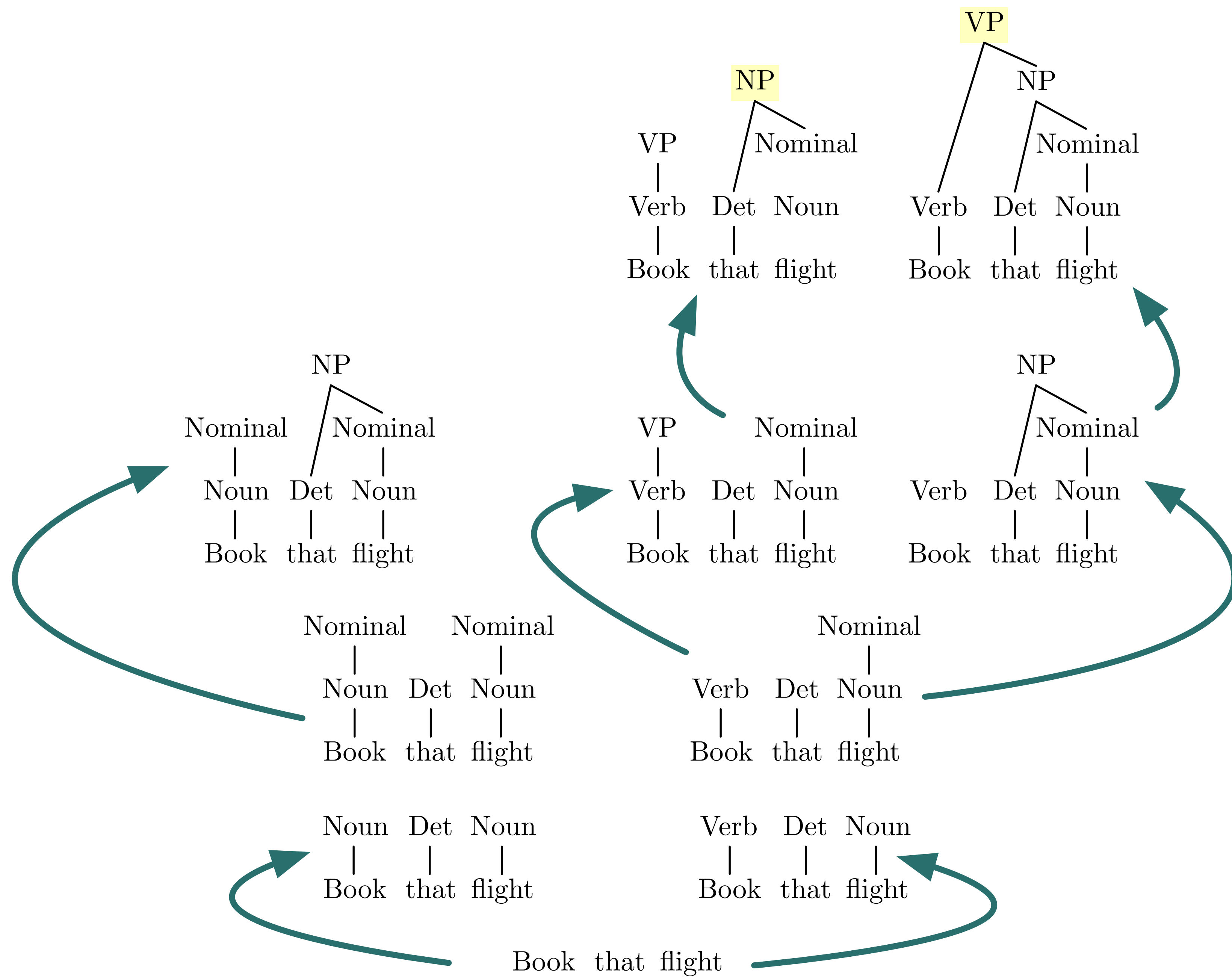
Recap: Parsing as Search



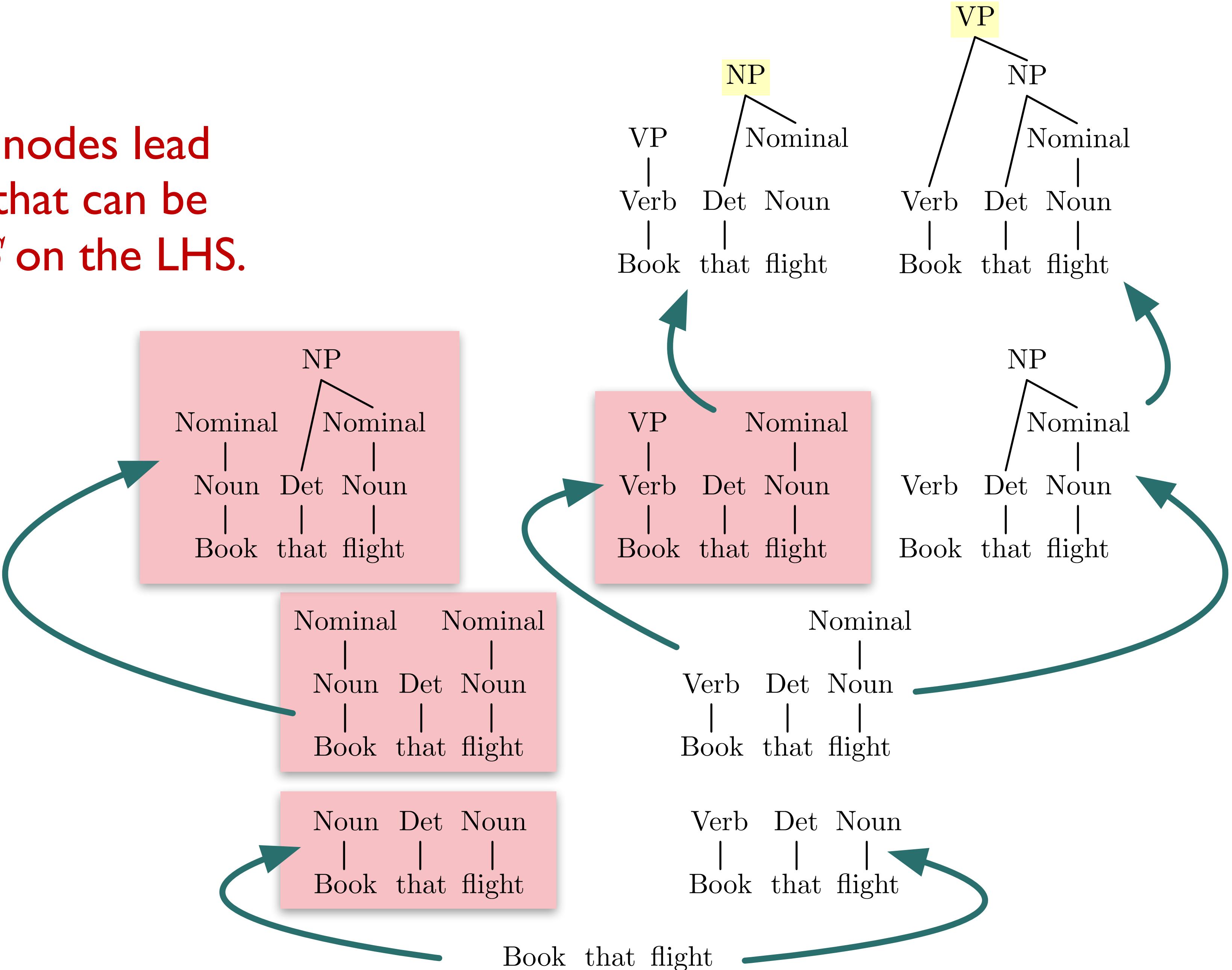
Recap: Parsing as Search



None of these nodes can produce *book* as first terminal



None of these nodes lead
lead to a RHS that can be
combined with *S* on the LHS.



Parsing Challenges

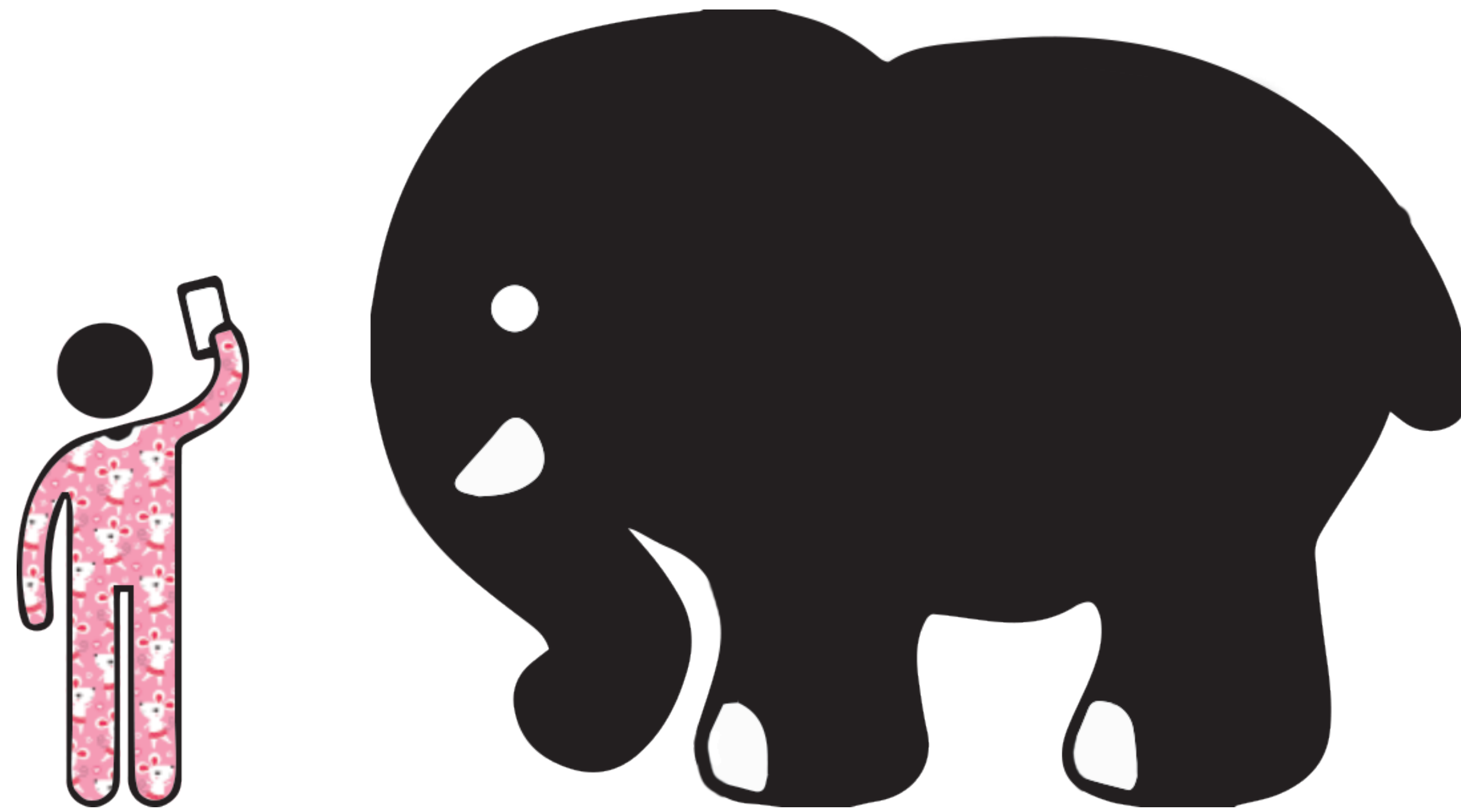
- Parsing-as-Search
- **Parsing Challenges**
 - **Ambiguity**
 - Repeated Substructure
 - Recursion
- Strategy: Dynamic Programming
- Grammar Equivalence
- CKY parsing algorithm

Parsing Ambiguity

- **Lexical Ambiguity:**
 - **Book/NN** → *I left a **book** on the table.*
 - **Book/VB** → ***Book** that flight.*
- **Structural Ambiguity**

Attachment Ambiguity

“One morning, I shot an elephant in my pajamas.

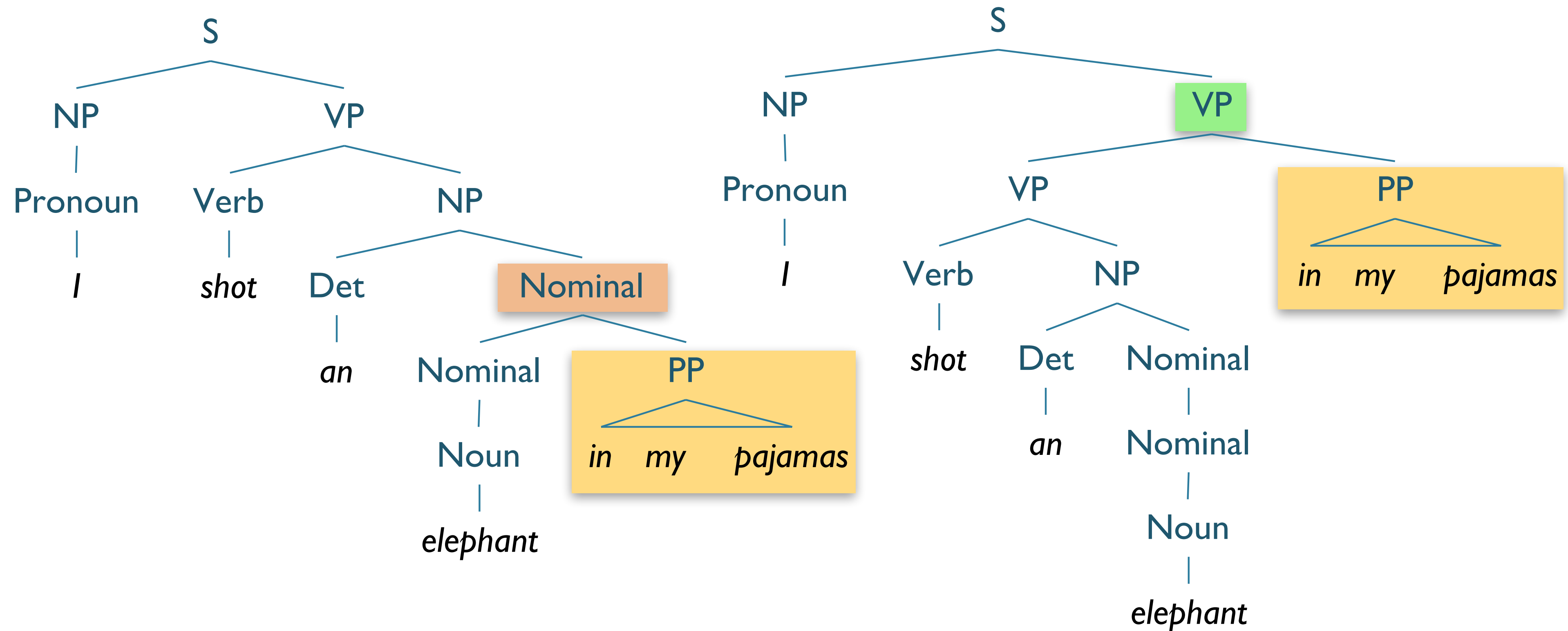


Attachment Ambiguity

“One morning, I shot an elephant in my pajamas.
How he got into my pajamas, I’ll never know.” — *Groucho Marx*



Attachment Ambiguity



“We saw the Eiffel Tower flying to Paris”



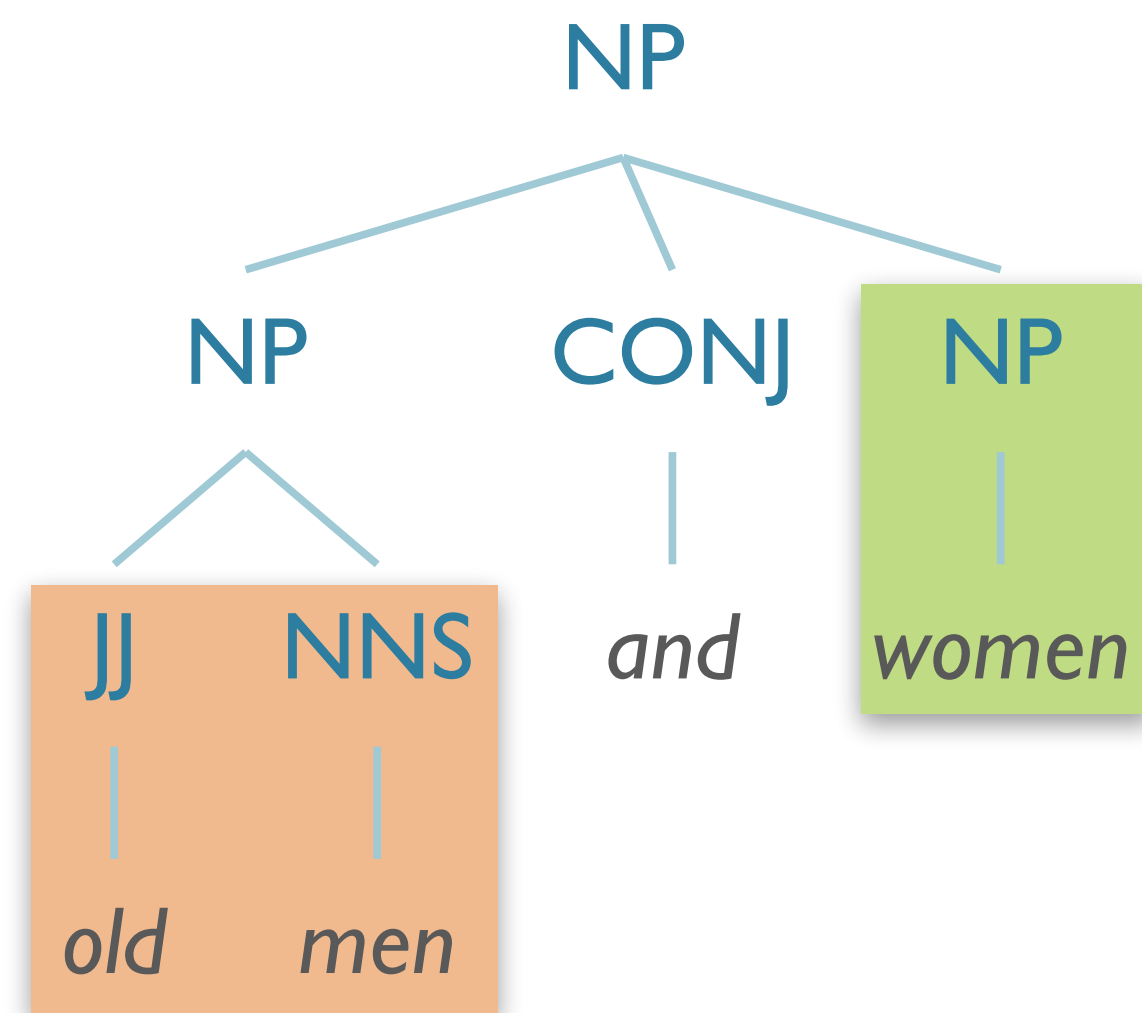
“We saw the Eiffel Tower flying to Paris”



Coordination Ambiguity:

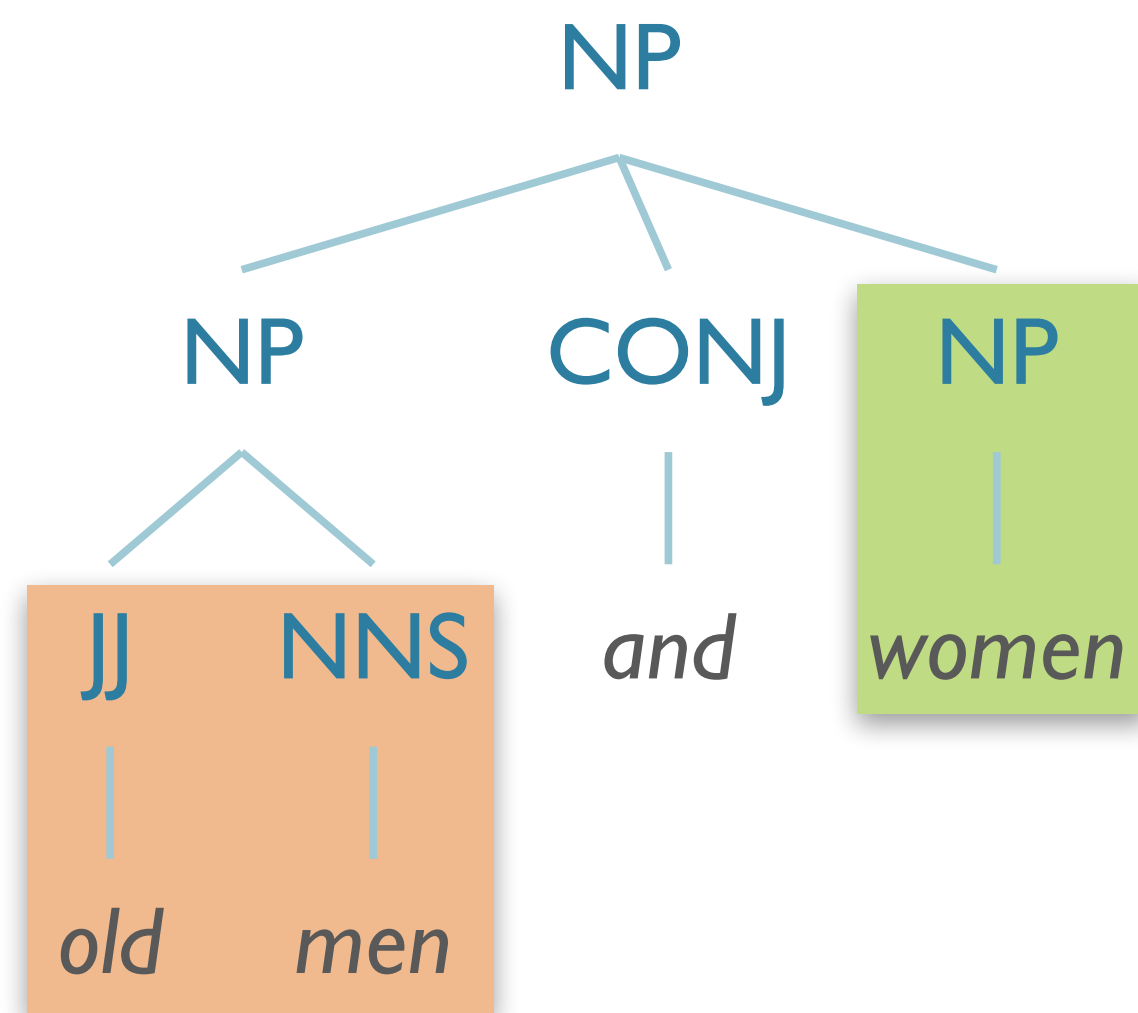
Coordination Ambiguity:

[old men] and **[women]**

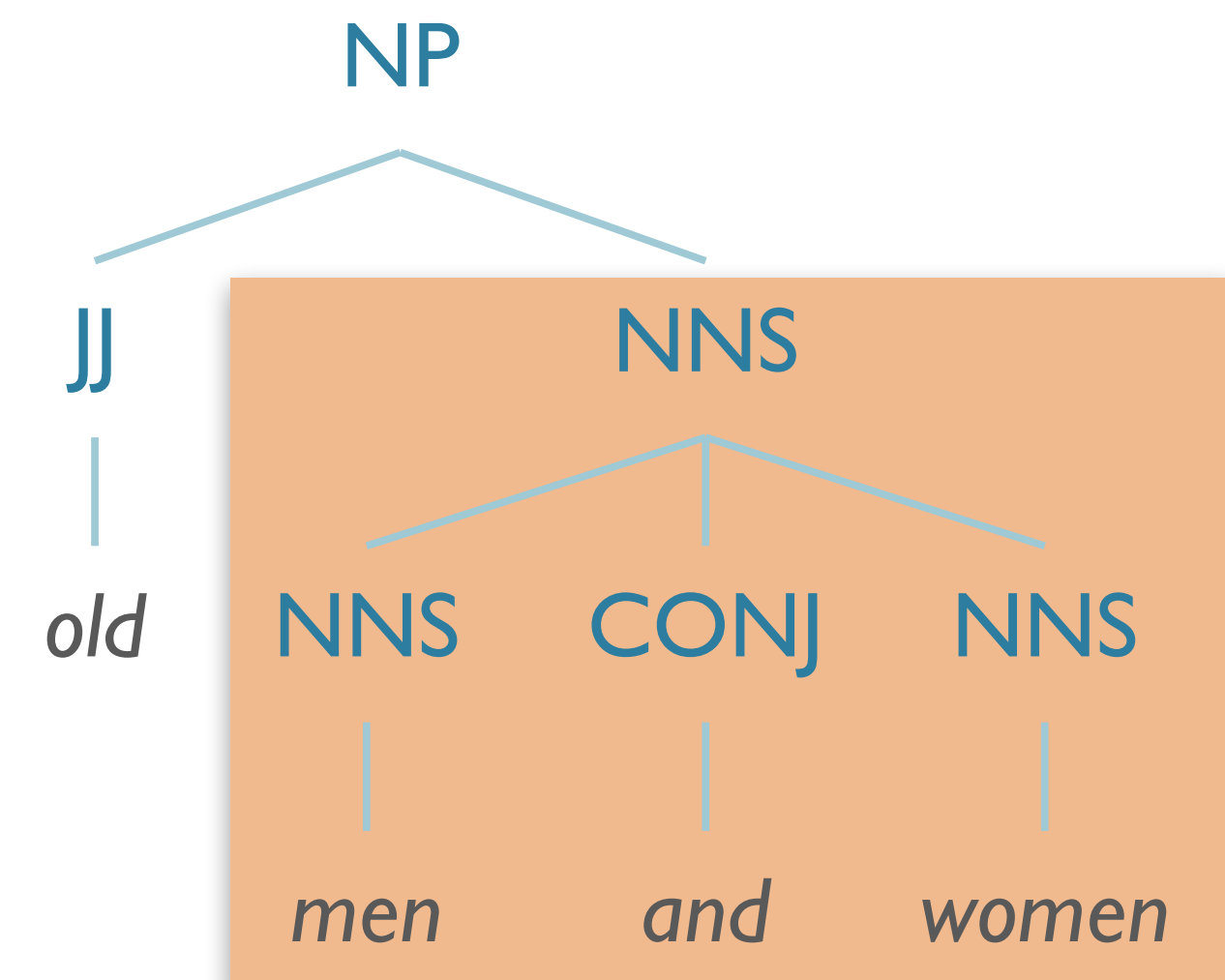


Coordination Ambiguity:

[old men] and **[women]**



[old **[men and women]**]



Local vs. Global Ambiguity

- **Local** ambiguity:
 - Ambiguity that cannot contribute to a full, valid parse
 - e.g. *Book/NN* in “*Book that flight*”

Local vs. Global Ambiguity

- **Local** ambiguity:
 - Ambiguity that cannot contribute to a full, valid parse
 - e.g. *Book/NN* in “*Book that flight*”
- **Global** ambiguity
 - Multiple valid parses

Why is Ambiguity a Problem?

- *Local* ambiguity:
 - increased processing time
- *Global* ambiguity:
 - Would like to yield only “reasonable” parses
 - Ideally, the one that was intended*

Solution to Ambiguity?

Solution to Ambiguity?

- ***Disambiguation!***

Solution to Ambiguity?

- ***Disambiguation!***
- Different possible strategies to select correct interpretation:

Disambiguation Strategy: Statistical

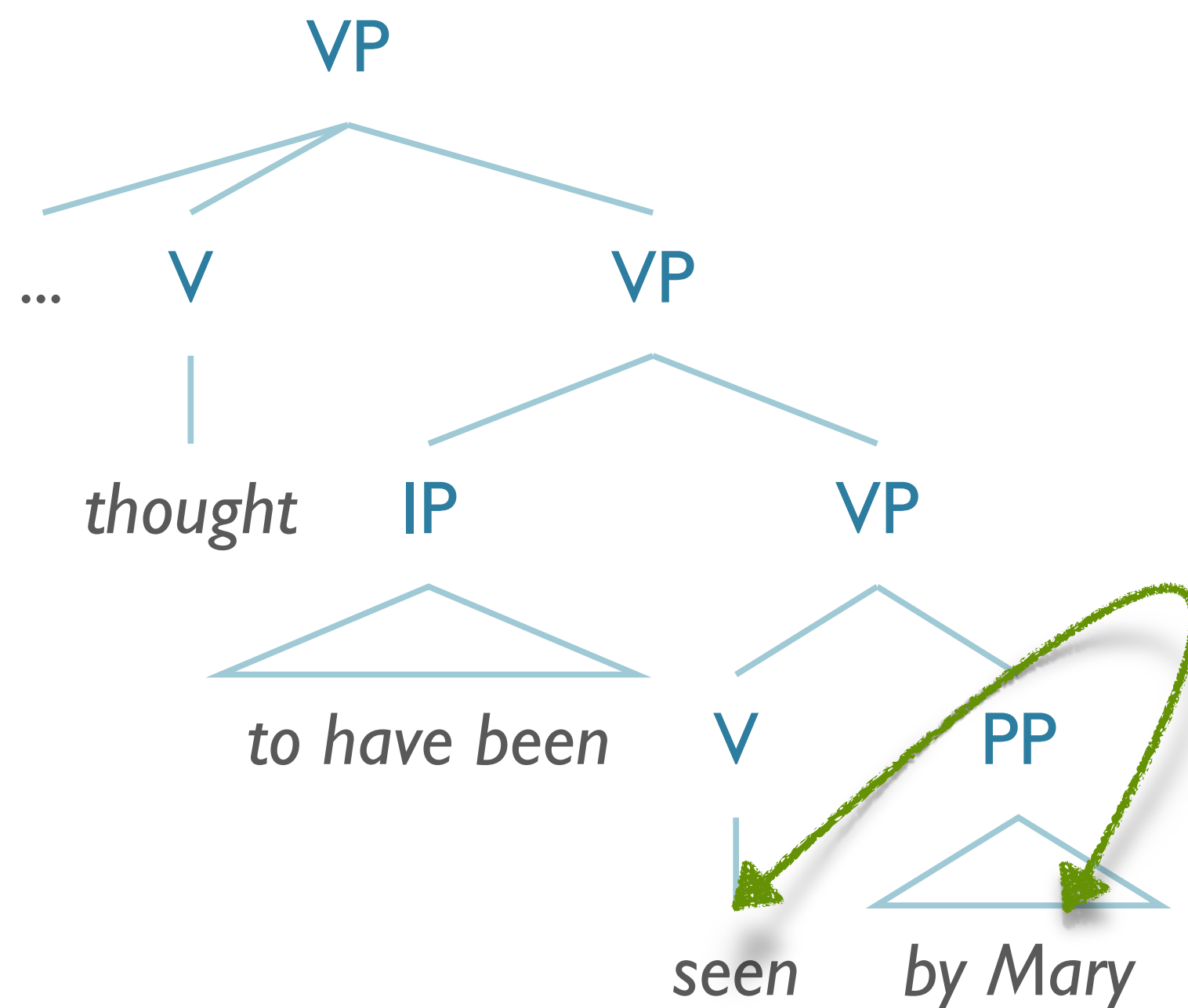
- Some prepositional structs more likely to attach high/low

Disambiguation Strategy: Statistical

- Some prepositional structs more likely to attach high/low
 - *John was thought to have been seen by Mary*
 - Mary could be doing the **seeing** or **thinking** — seeing more likely

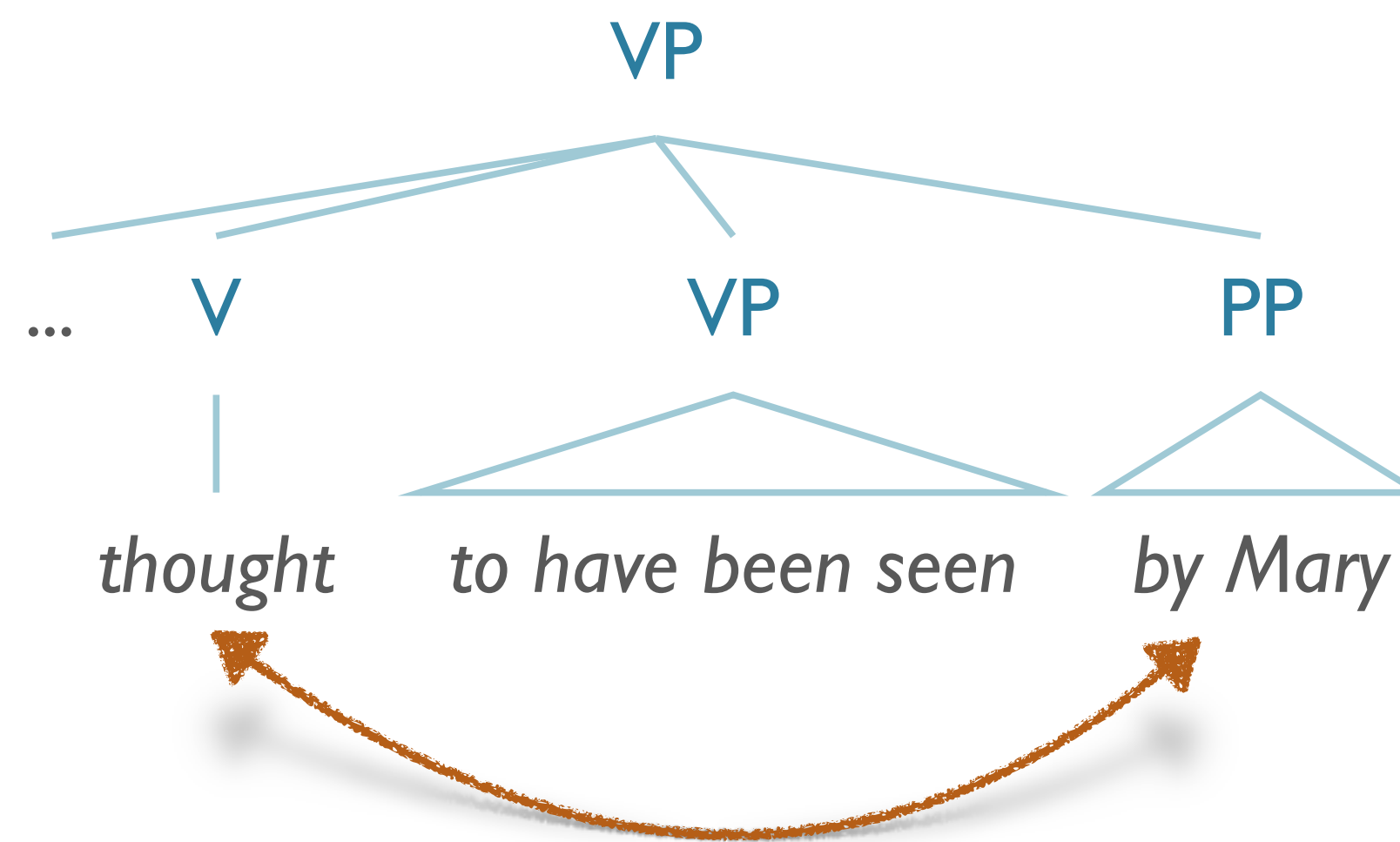
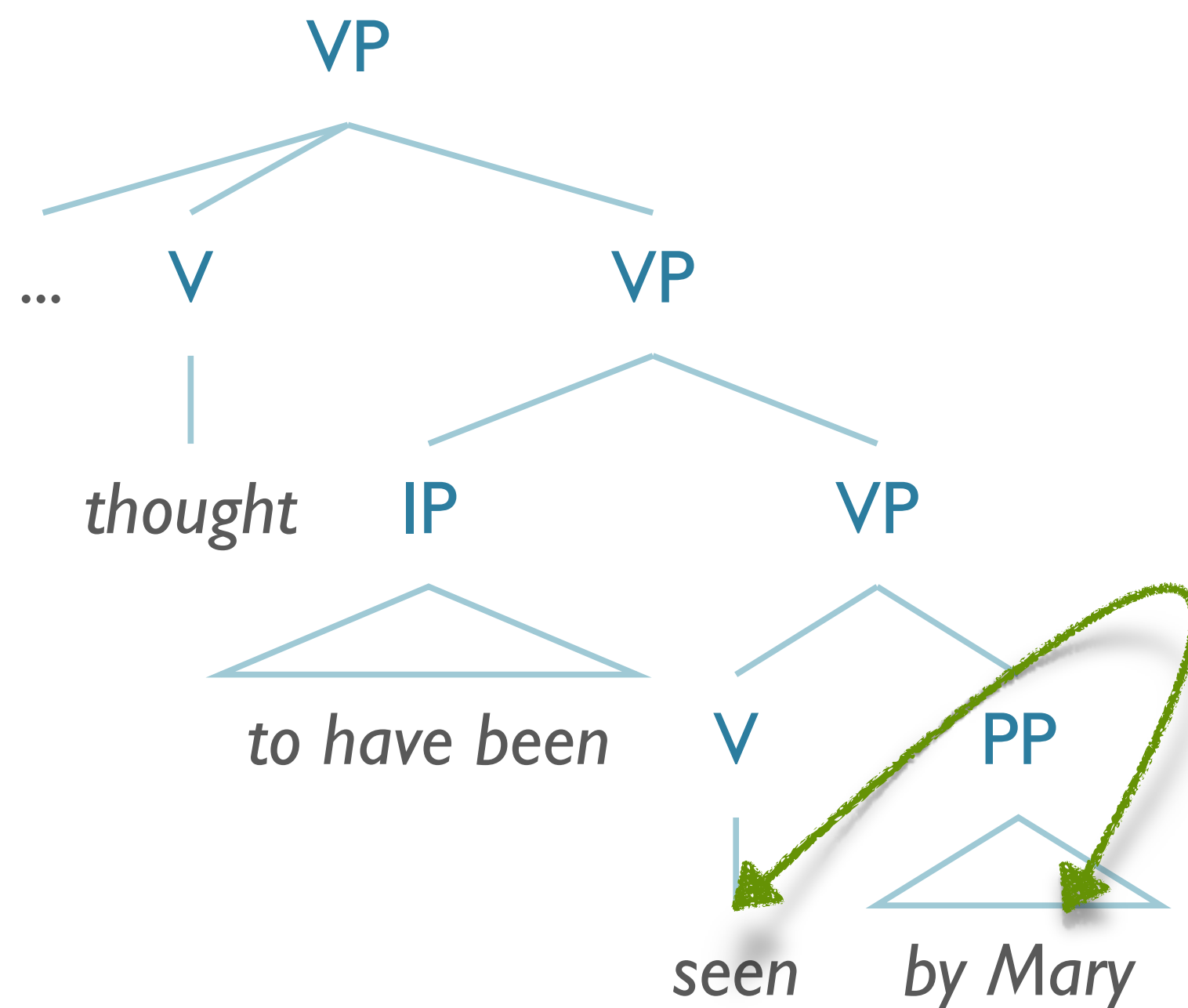
Disambiguation Strategy: Statistical

- Some prepositional structs more likely to attach high/low
- *John was thought to have been seen by Mary*
 - Mary could be doing the **seeing** or **thinking** — seeing more likely



Disambiguation Strategy: Statistical

- Some prepositional structs more likely to attach high/low
- *John was thought to have been seen by Mary*
 - Mary could be doing the **seeing** or **thinking** — seeing more likely

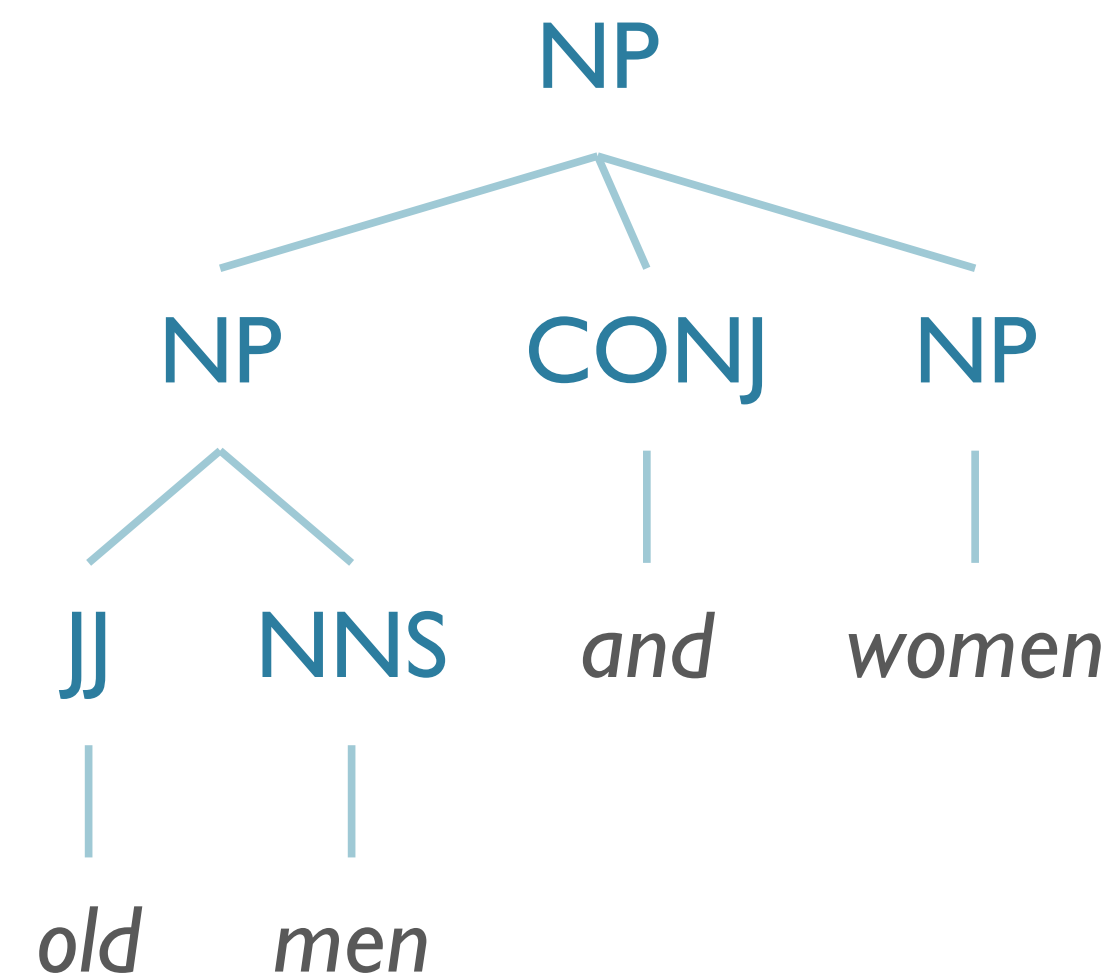
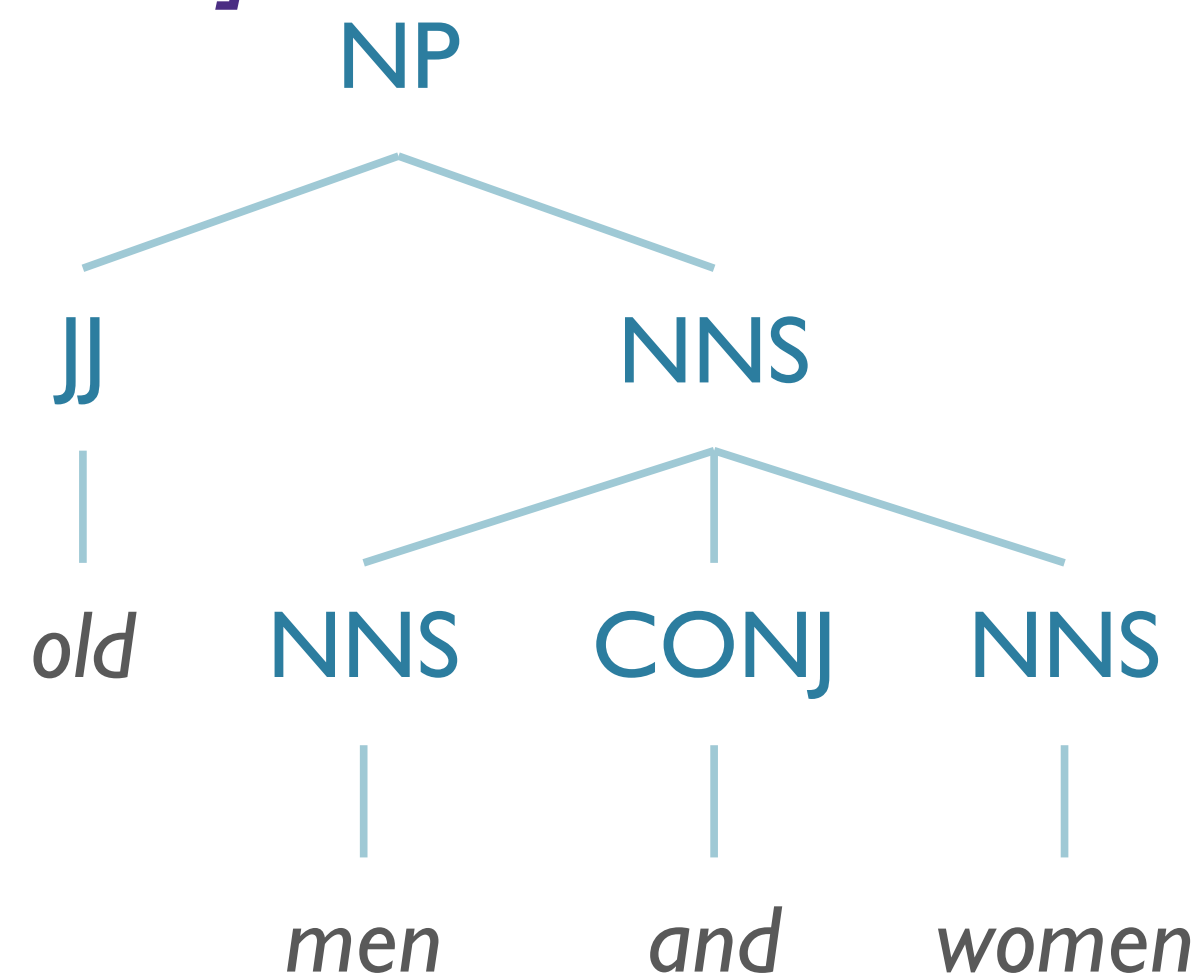


Disambiguation Strategy: Statistical

- Some phrases more likely overall

Disambiguation Strategy: Statistical

- Some phrases more likely overall
 - *[old [men and women]] is a more common construction than [old men] and [women]*



Disambiguation Strategy:

Semantic

- Some interpretations we know to be semantically impossible

Disambiguation Strategy:

Semantic

- Some interpretations we know to be semantically impossible
 - *Eiffel tower* as subject of *fly*

Disambiguation Strategy: Pragmatic

- Some interpretations are possible, unlikely given world knowledge

Disambiguation Strategy: Pragmatic

- Some interpretations are possible, unlikely given world knowledge
 - e.g. elephants and pajamas

Incremental Parsing and Garden Paths

- Idea: model *left-to-right* nature of (English) text
- Problem: “garden path” sentences

Incremental Parsing and Garden Paths

- Idea: model *left-to-right* nature of (English) text
- Problem: “garden path” sentences



Business

Markets

World

Politics

TV

More

SPORTS NEWS

SEPTEMBER 30, 2019 / 9:17 AM / A DAY AGO

California to let college athletes be paid in blow to NCAA rules

<https://www.reuters.com/article/us-sport-california-education/california-to-let-college-athletes-be-paid-in-blow-to-ncaa-rules-idUSKBNIWFISR>

Disambiguation Strategy:



- Alternatively, keep all parses

Disambiguation Strategy:



- Alternatively, keep all parses
 - *(Might even be the appropriate action for some jokes)*

Parsing Challenges

- Parsing-as-Search
- **Parsing Challenges**
 - Ambiguity
 - **Repeated Substructure**
 - Recursion
- Strategy: Dynamic Programming
- Grammar Equivalence
- CKY parsing algorithm

Repeated Work

- Search (top-down/bottom-up) both lead to repeated substructures
 - Globally bad parses can construct good subtrees
 - ...will reconstruct along another branch
 - No static backtracking can avoid

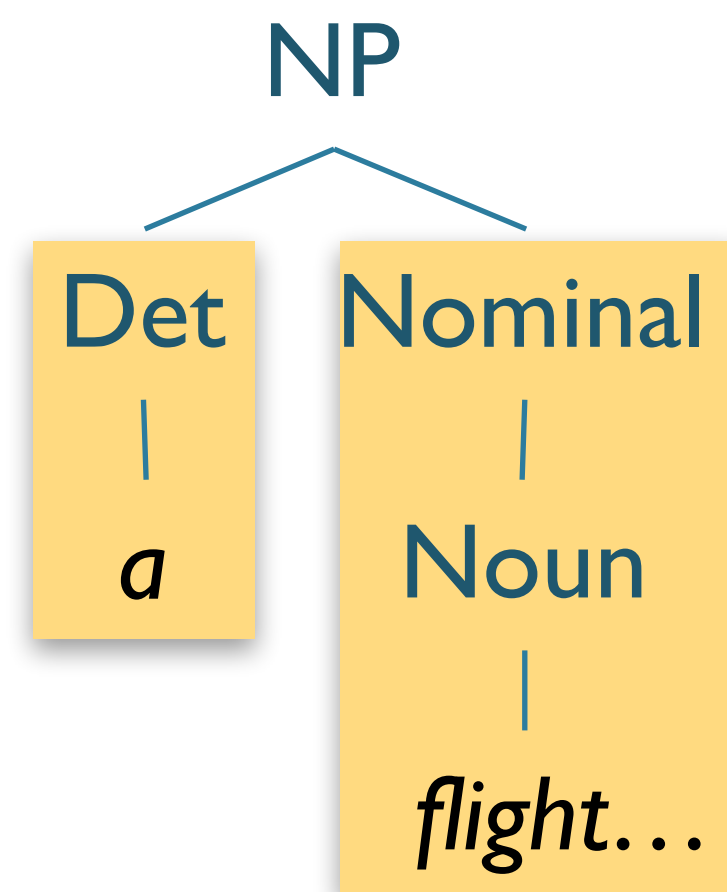
Repeated Work

- Search (top-down/bottom-up) both lead to repeated substructures
 - Globally bad parses can construct good subtrees
 - ...will reconstruct along another branch
 - No static backtracking can avoid
- Efficient parsing techniques require storage of partial solutions

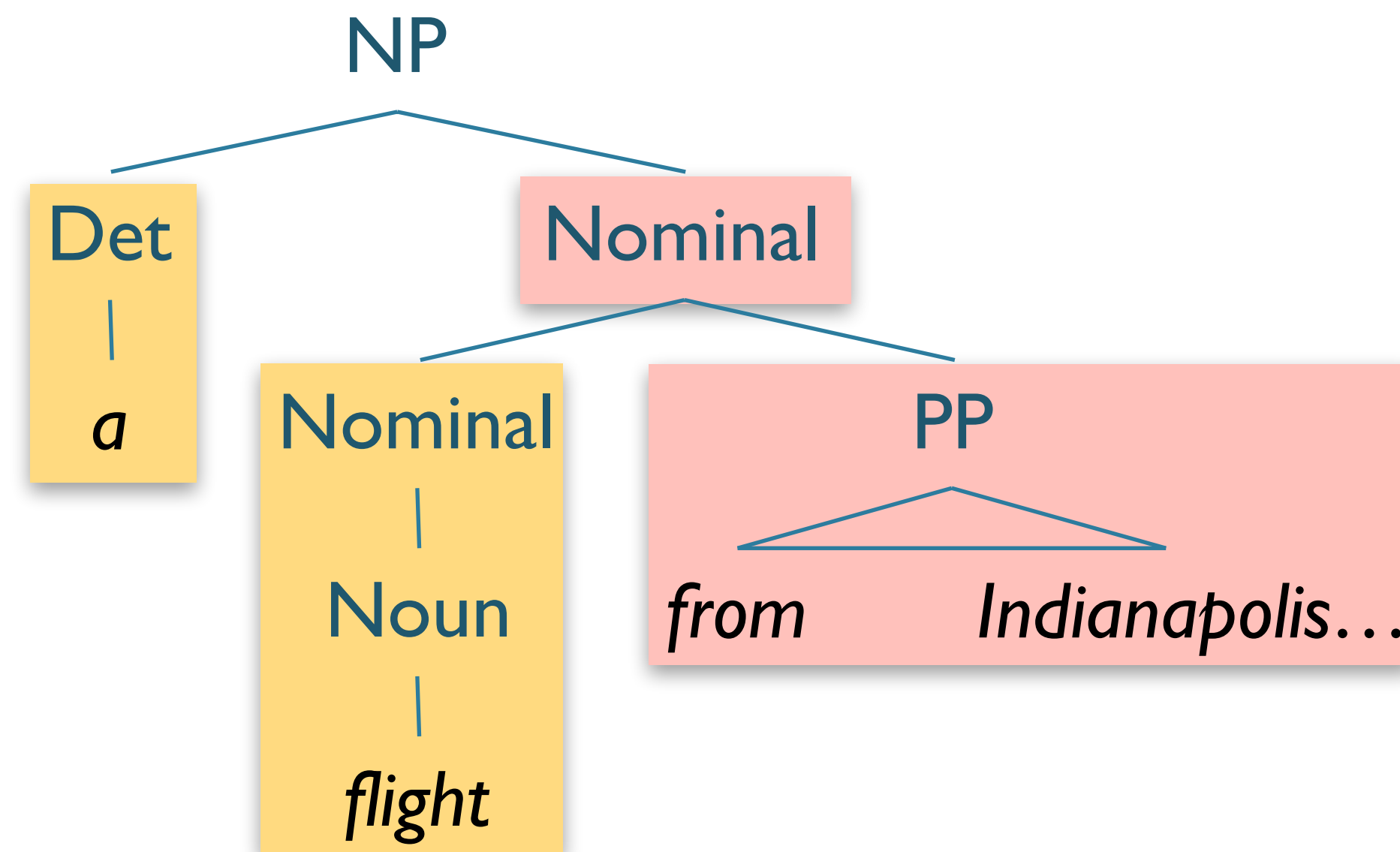
Repeated Work

- Search (top-down/bottom-up) both lead to repeated substructures
 - Globally bad parses can construct good subtrees
 - ...will reconstruct along another branch
 - No static backtracking can avoid
- Efficient parsing techniques require storage of partial solutions
- Example: *a flight from Indianapolis to Houston on TWA*

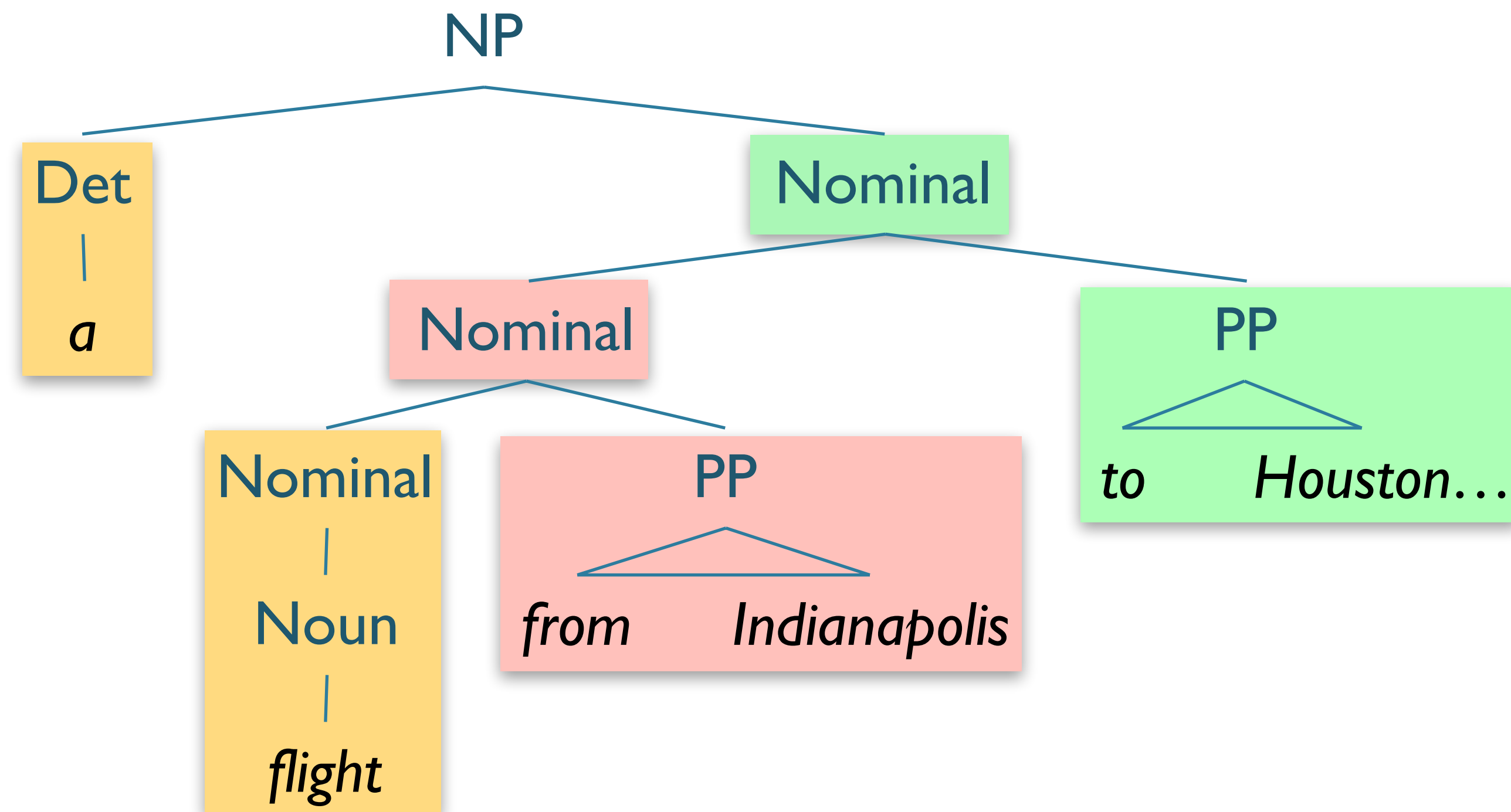
Shared Sub-Problems



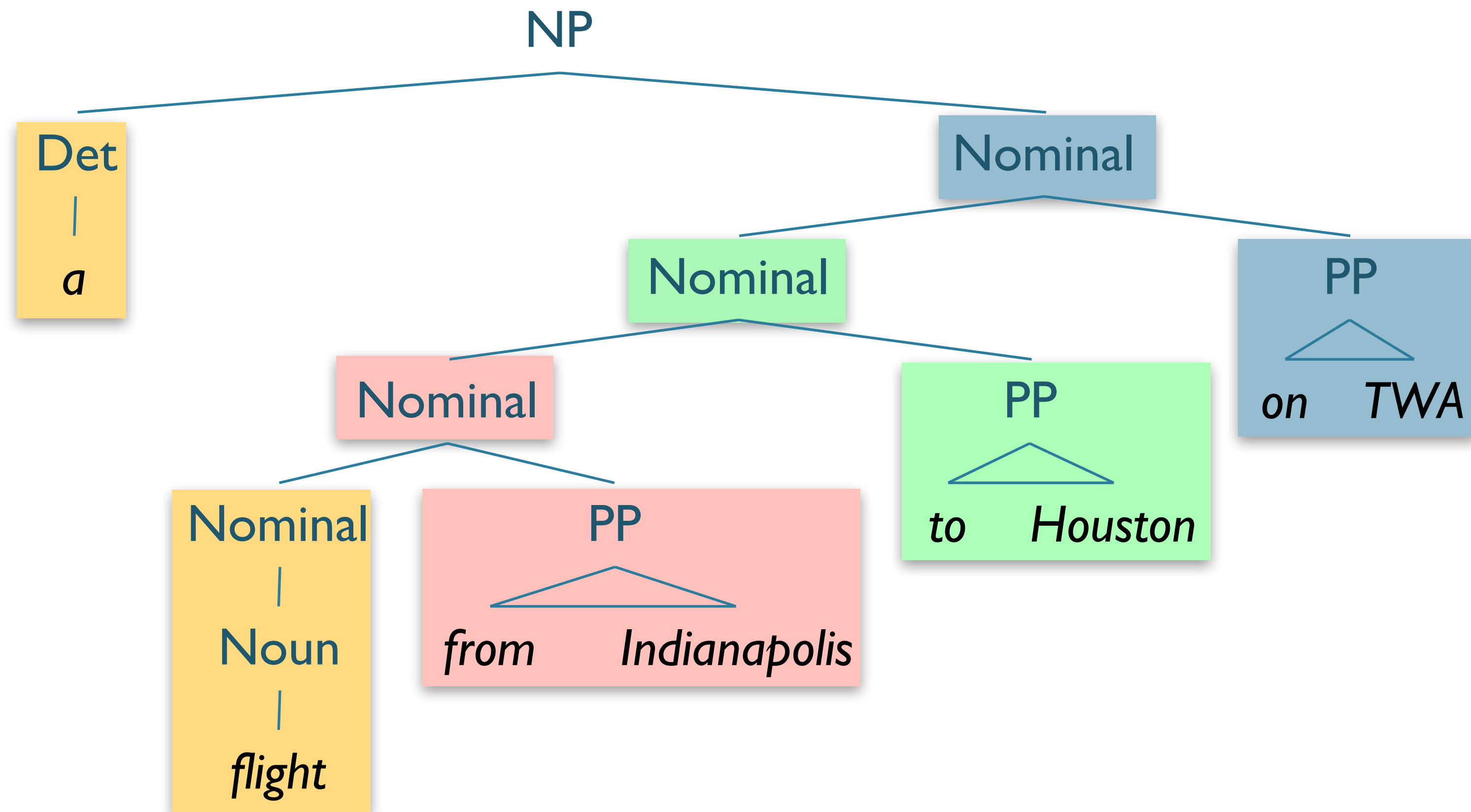
Shared Sub-Problems



Shared Sub-Problems



Shared Sub-Problems

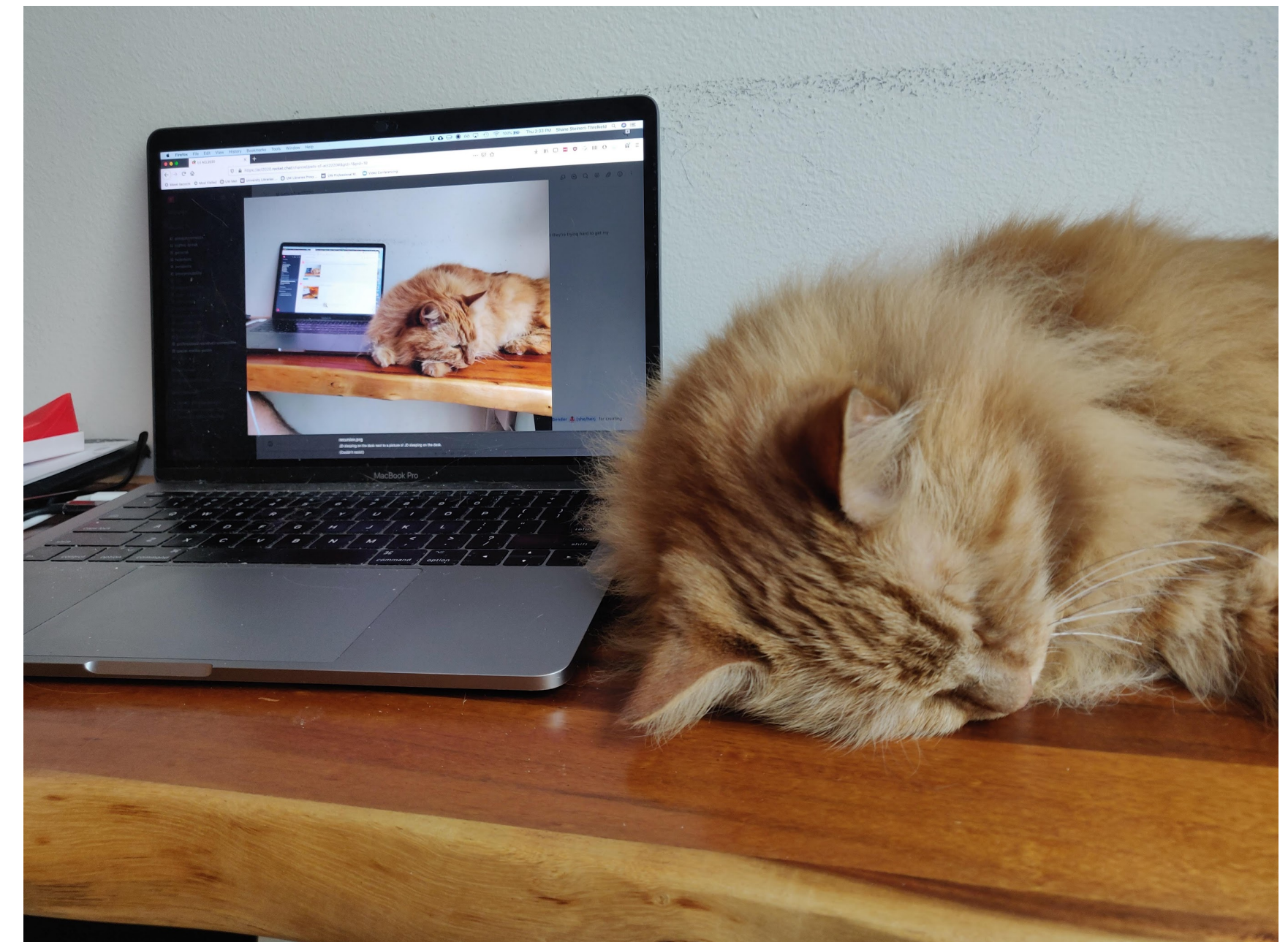


Parsing Challenges

- Parsing-as-Search
- **Parsing Challenges**
 - Ambiguity
 - Repeated Substructure
 - **Recursion**
- Strategy: Dynamic Programming
- Grammar Equivalence
- CKY parsing algorithm

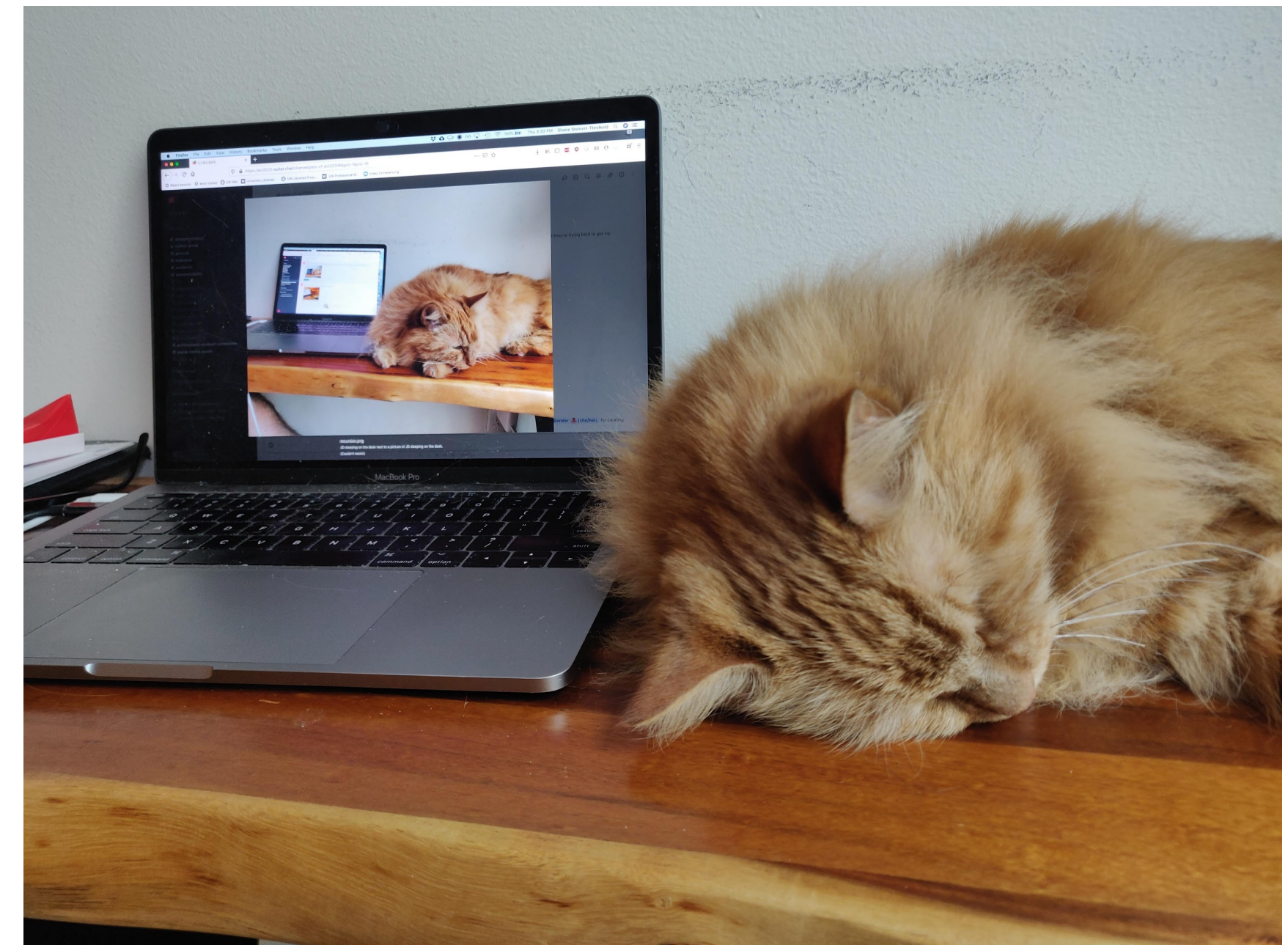
Recursion

- Many grammars have recursive rules
 - $S \rightarrow S \text{ Conj } S$



Recursion

- Many grammars have recursive rules
 - $S \rightarrow S \text{ Conj } S$
- In search approaches, recursion is problematic
 - Can yield infinite searches
 - Top-down especially vulnerable



Roadmap

- Parsing-as-Search
- Parsing Challenges
- **Strategy: Dynamic Programming**
- Grammar Equivalence
- CKY parsing algorithm

Dynamic Programming

- Challenge:
 - Repeated substructure → Repeated Work

Dynamic Programming

- Challenge:
 - Repeated substructure → Repeated Work
- Insight:
 - Global parse composed of sub-parses
 - Can record these sub-parses and re-use

Dynamic Programming

- Challenge:
 - Repeated substructure → Repeated Work
- Insight:
 - Global parse composed of sub-parses
 - Can record these sub-parses and re-use
- Dynamic programming avoids repeated work by recording the subproblems
 - Here, stores subtrees

Parsing with Dynamic Programming

- Avoids repeated work
- Allows implementation of (relatively) efficient parsing algorithms
 - Polynomial time in input length
 - Typically cubic (n^3) or less

Parsing with 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

Roadmap

- Parsing-as-Search
- Parsing Challenges
- Strategy: Dynamic Programming
- **Grammar Equivalence**
- CKY parsing algorithm

Grammar Equivalence and Form

- *Weak* Equivalence
 - **Accepts** same language
 - May produce **different** structures
- *Strong* Equivalence
 - Accepts same language
 - Produces **same** structures

Grammar Equivalence and Form

Grammar Equivalence and Form

- Reason?
 - We can create a weakly-equivalent grammar that allows for greater efficiency
 - This is required by the CKY algorithm

Chomsky Normal Form (CNF)

- Required by CKY Algorithm

Chomsky Normal Form (CNF)

- Required by CKY Algorithm
- All productions are of the form:
 - $A \rightarrow B C$
 - $A \rightarrow a$

Chomsky Normal Form (CNF)

- Required by CKY Algorithm
- All productions are of the form:
 - $A \rightarrow B C$
 - $A \rightarrow a$
- Most of our grammars are not of this form:
 - $S \rightarrow Wh-NP Aux NP VP$

Chomsky Normal Form (CNF)

- Required by CKY Algorithm
- All productions are of the form:
 - $A \rightarrow B C$
 - $A \rightarrow a$
- Most of our grammars are not of this form:
 - $S \rightarrow Wh-NP Aux NP VP$
- Need a general conversion procedure

CNF Conversion

Hybrid productions:

$INF-VP \rightarrow \text{to } VP$

Unit productions:

$A \rightarrow B$

Long productions:

$A \rightarrow B C D \dots$

CNF Conversion: Hybrid Productions

- Hybrid production:
 - Replace all terminals with dummy non-terminal
 - $INF-VP \rightarrow to\ VP$
 - $INF-VP \rightarrow TO\ VP$
 - $TO \rightarrow to$

CNF Conversion: Unit Productions

- Unit productions:
 - Rewrite RHS with RHS of all derivable, non-unit productions
 - If $A \Rightarrow^* B$ and $B \rightarrow \gamma$, **add** $A \rightarrow \gamma$ [where γ is any non-unit RHS]
 - [$A \Rightarrow^* B$: *B is reachable from A by a sequence of unit productions*]
- $Nominal \rightarrow Noun, Noun \rightarrow \mathbf{dog}$
 - $Nominal \rightarrow \mathbf{dog}$
 - $Noun \rightarrow \mathbf{dog}$
- NB: this example has γ as a single terminal, but the rule applies to all non-unit RHS.

CNF Conversion: Long Productions

CNF Conversion: Long Productions

- Long productions

$$S \rightarrow Aux\ NP\ VP$$

CNF Conversion: Long Productions

- Long productions

$$S \rightarrow Aux\ NP\ VP$$
$$S \rightarrow \textcolor{red}{X1}\ VP \qquad \textcolor{red}{X1} \rightarrow Aux\ NP$$

CNF Conversion: Long Productions

- Long productions

$$S \rightarrow Aux\ NP\ VP$$
$$S \rightarrow \textcolor{red}{X1}\ VP \qquad \textcolor{red}{X1} \rightarrow Aux\ NP$$

- Introduce unique nonterminals, and spread over rules

CNF Conversion

Convert terminals in hybrid rules to dummy non-terminals

Convert unit productions

Binarize long production rules

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

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

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

Roadmap

- Parsing-as-Search
- Parsing Challenges
- Strategy: Dynamic Programming
- Grammar Equivalence
- **CKY parsing algorithm**

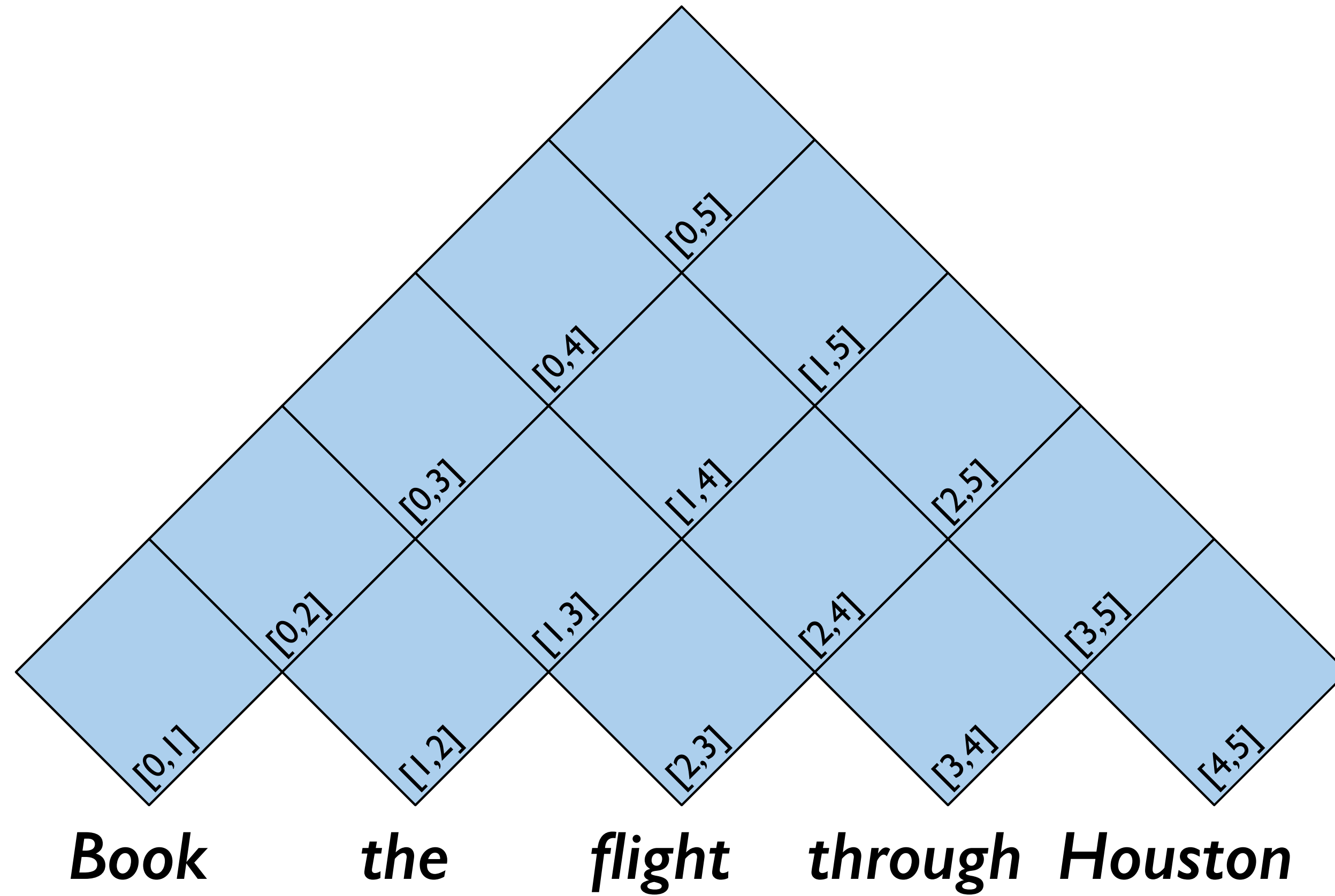
CKY Parsing

- (Relatively) efficient parsing algorithm
- Based on tabulating substring parses to avoid repeat work
- Approach:
 - Use CNF Grammar
 - Build an $(n + 1) \times (n + 1)$ matrix to store subtrees
 - Upper triangular portion
 - Incrementally build parse spanning whole input string

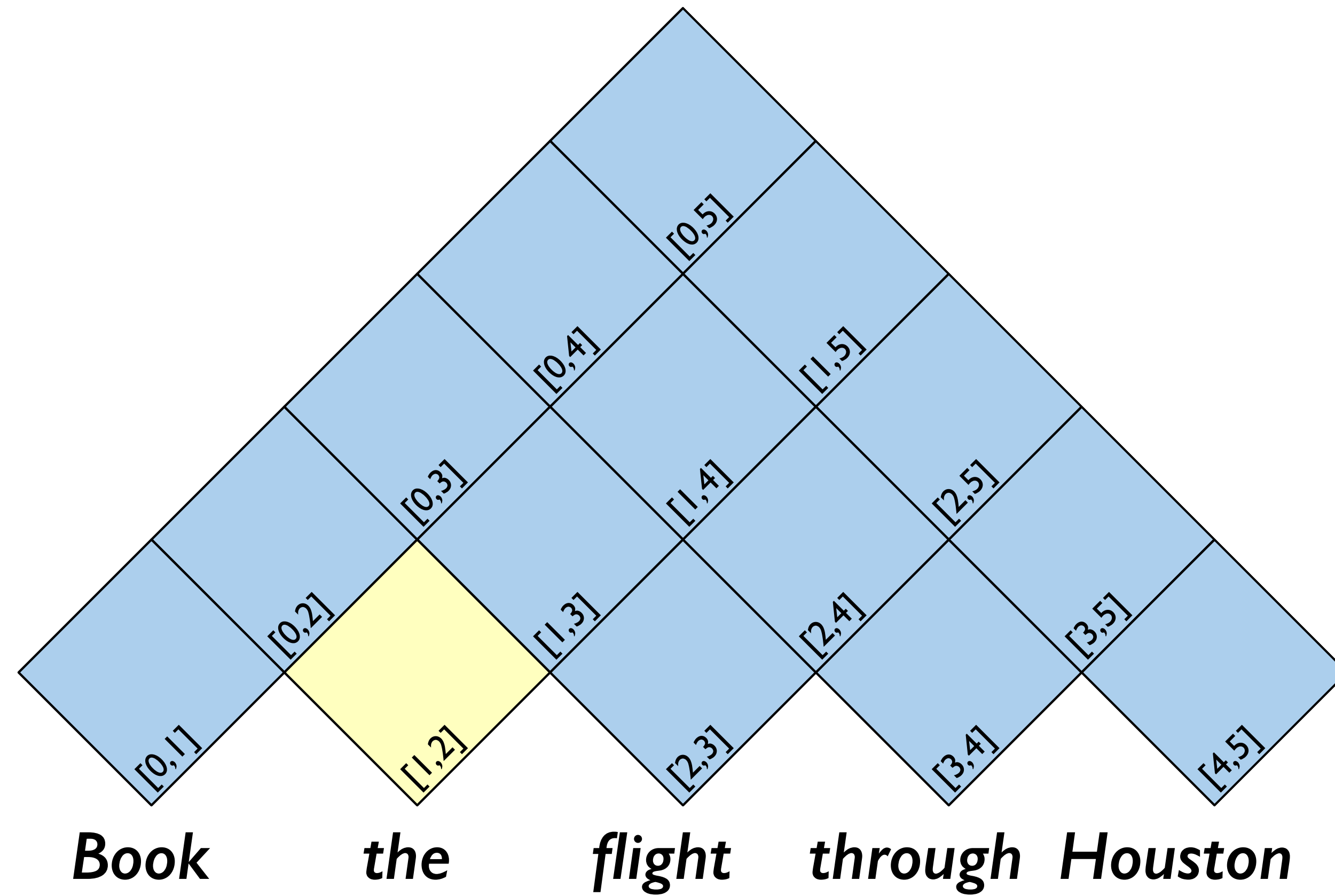
CKY Matrix

<i>Book</i>	<i>the</i>	<i>flight</i>	<i>through</i>	<i>Houston</i>
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	[1,2]	[1,3]	[1,4]	[1,5]
		[2,3]	[2,4]	[2,5]
			[3,4]	[3,5]
				[4,5]

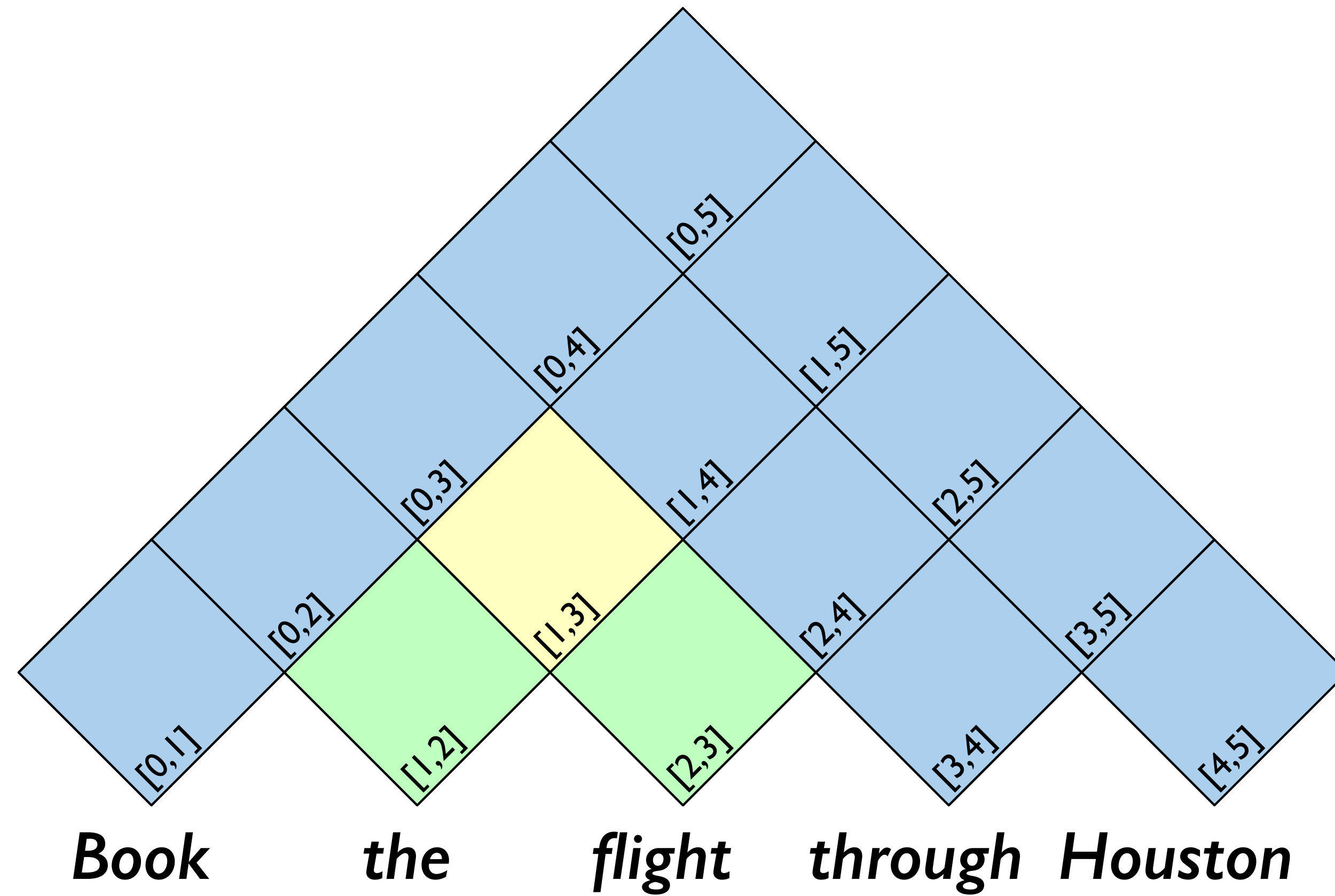
CKY Matrix



CKY Matrix



CKY Matrix



Dynamic Programming in CKY

- Key idea:
 - for $i < k < j$
 - ...and a parse spanning substring $[i, j]$
 - There is a k such that there are parses spanning $[i, k]$ and $[k, j]$
 - We can construct parses for whole sentences by building from these partial parses
- So to have a rule $A \rightarrow B \ C$ in $[i, j]$
 - Must have B in $[i, k]$ and C in $[k, j]$ for some $i < k < j$
 - CNF forces this for all $j > i + 1$