Rule-Based and Statistical Constituency Parsing

Natalie Parde

UIC CS 421

What is syntactic parsing?

The process of automatically recognizing and assigning syntactic (grammatical) roles to the constituents within sentences

Why is syntactic parsing useful?

- Lots of reasons!
  - Grammar checking
    - Sentences that can’t be parsed may be grammatically incorrect (or at least hard to read)
  - Semantic analysis
  - Downstream applications
    - Question answering
    - Information extraction

What courses were **taught** **by UIC CS assistant professors** **in 2022**?

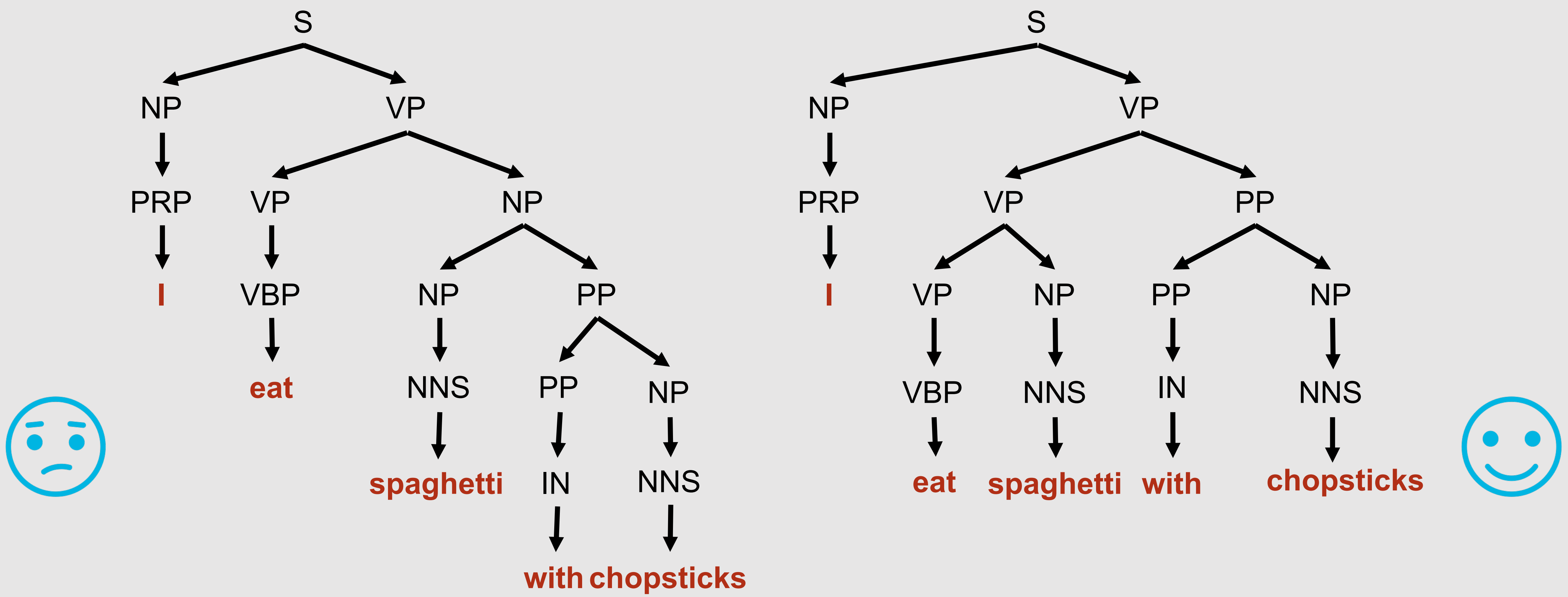
Subject = courses …don’t return a list of UIC CS assistant professors!

Parsing algorithms are one of the core tools for analyzing natural language.

- Parsing algorithms automatically describe the syntactic structure of sentences in terms of **context-free grammars**
- This can be viewed as a **search problem**:
  - Given the set of all possible parse trees, find the correct parse tree for this sentence.

Recognition vs. Parsing

- **Recognition:** Deciding whether a sentence belongs to the language specified by a formal grammar.
- **Parsing:** Producing a parse tree for the sentence based on that formal grammar.
- Both tasks are necessary for generating correct syntactic parses!
  - Failure to accurately recognize whether a sentence can be parsed will lead to **misparses**, which will in turn lead to additional errors in downstream applications.
- Parsing is more “difficult” (greater time complexity) than recognition



Remember, language is ambiguous!

Input sentences may have many possible parses

There are many ways to generate parse trees.

Top-Down Parsing:

- Goal-driven
- Builds parse tree from the start symbol down to the terminal nodes

Bottom-Up Parsing:

- Data-driven
- Builds parse tree from the terminal nodes up to the start symbol

These approaches can be implemented naïvely, or using more advanced techniques.

Naïve approach: Enumerate all possible solutions

Dynamic programming approach: Save partial solutions in a table, and use this information to reduce search time

Top-Down Parsing

- Assume that the input can be derived by the designated start symbol ***S***
- Find the tops of all trees that can start with ***S***
  - Look for all production rules with ***S*** on the left-hand side
- Find the tops of all trees that can start with those constituents
- (Repeat recursively until terminal nodes are reached)
- Trees whose leaves fail to match all words in the input sentence can be rejected, leaving behind trees that represent successful parses

Top-Down Parsing: Example

Input Sentence:

Book that flight.

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

Lexicon:

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

Top-Down Parsing: Example

Book that flight.

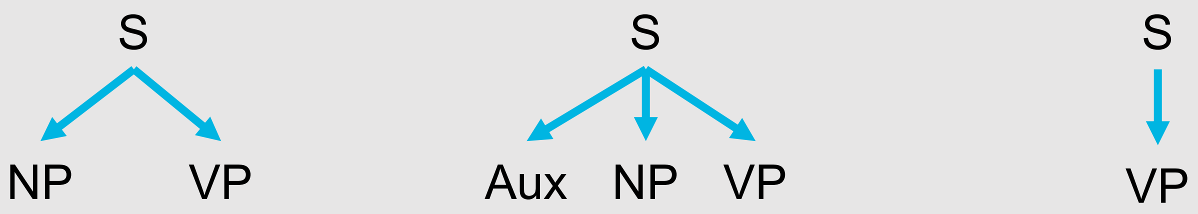
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

S S S

Top-Down Parsing: Example

Book that flight.

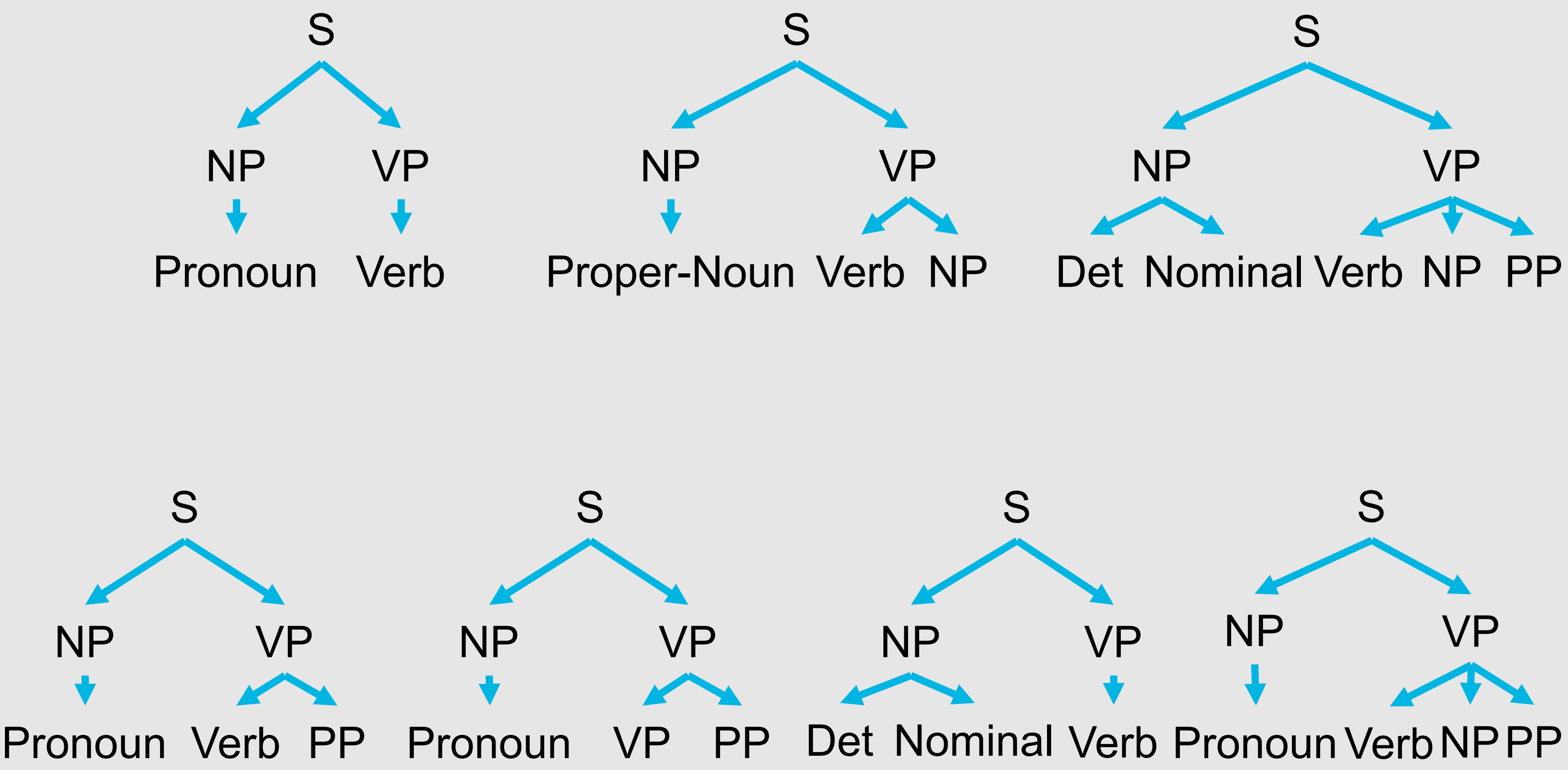
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

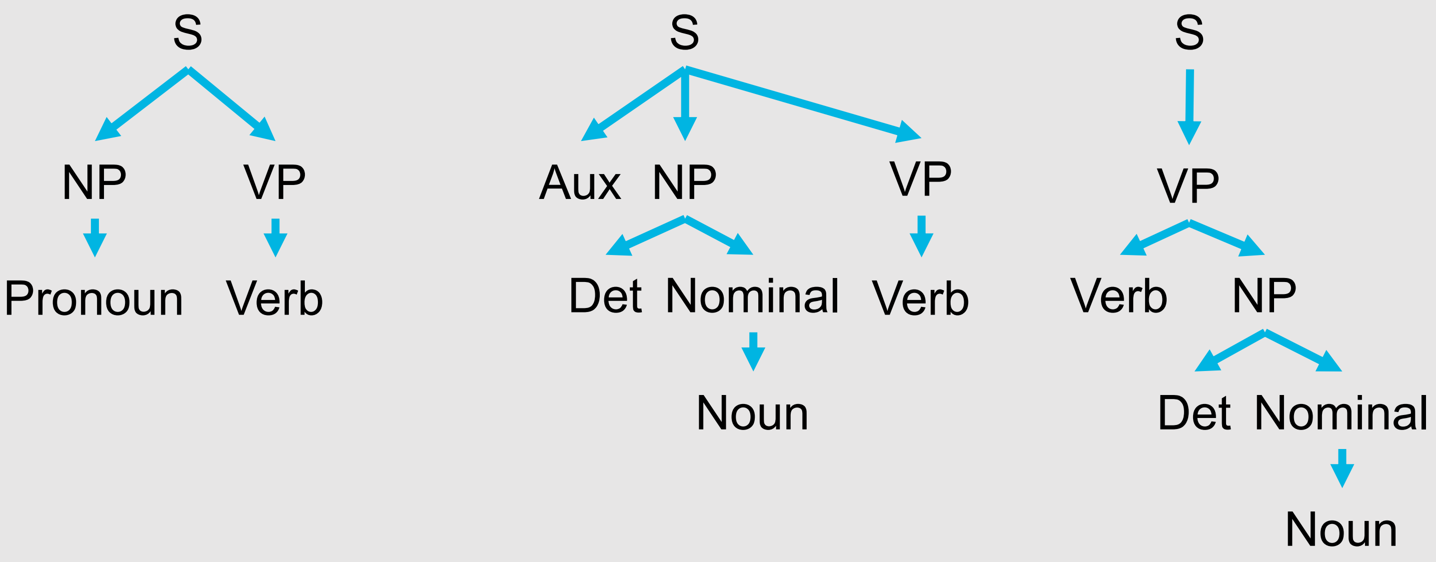


…and many more!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

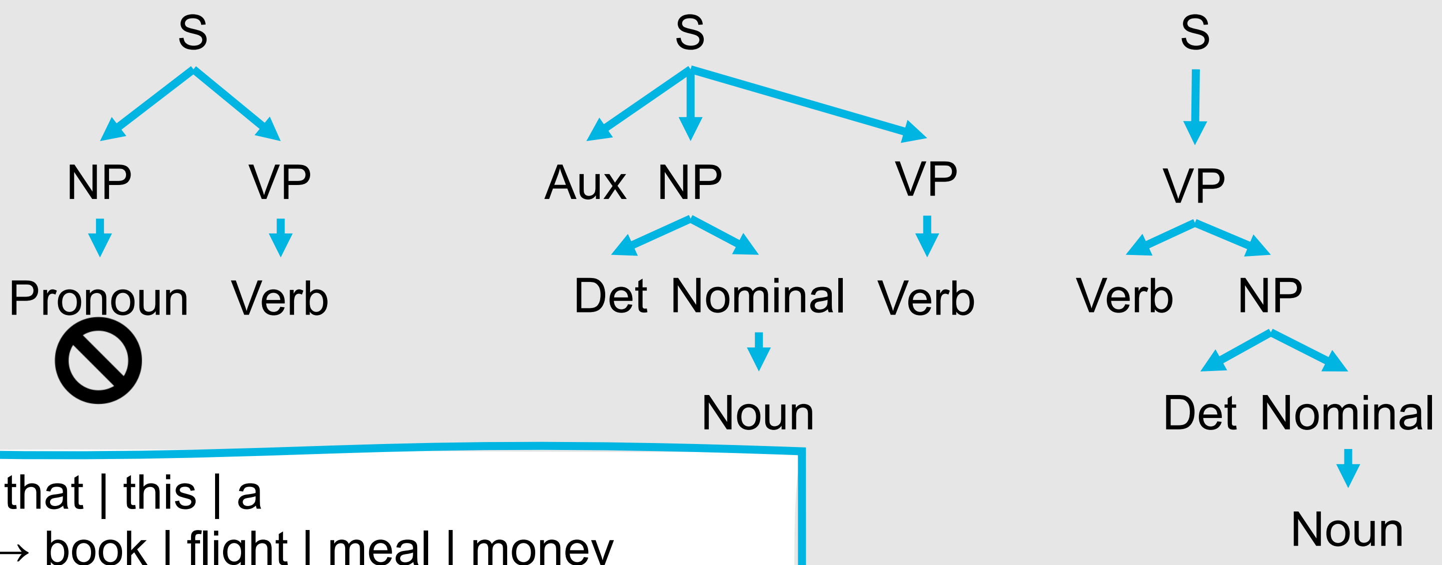


…and many, many more not shown!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



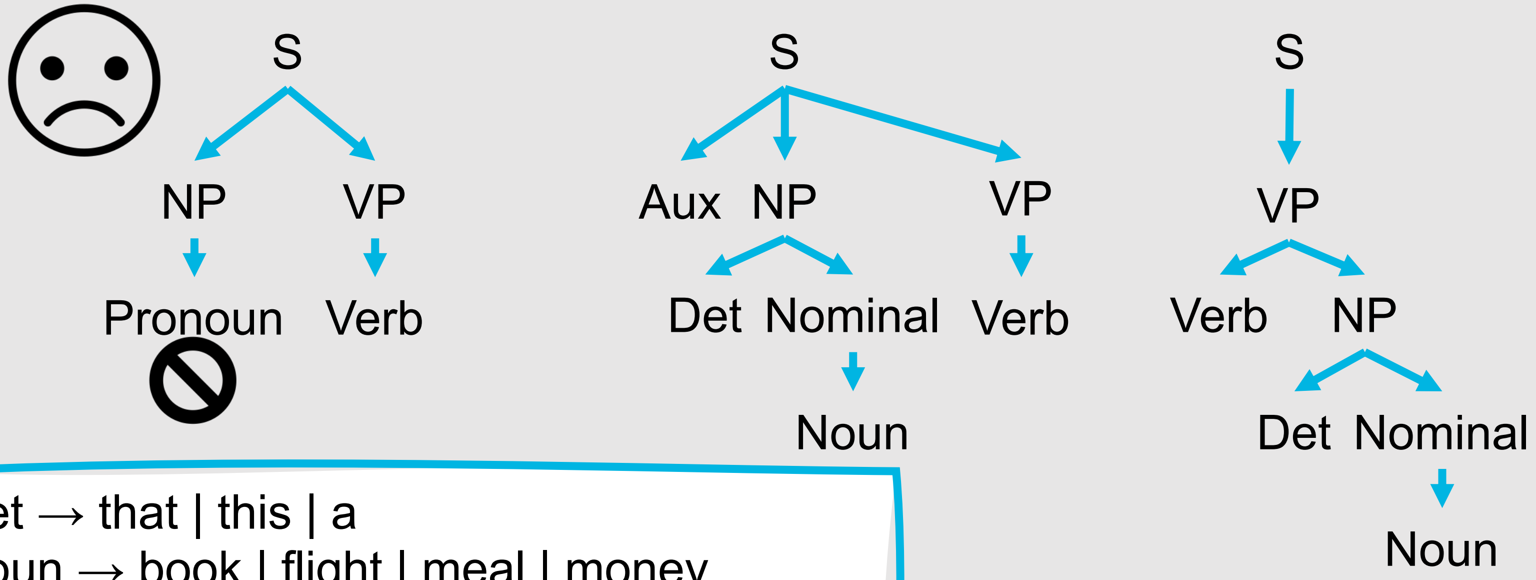
Det → that | this | a
Noun → book | flight | meal | money
Verb → **book** | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

…and many, many more not shown!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



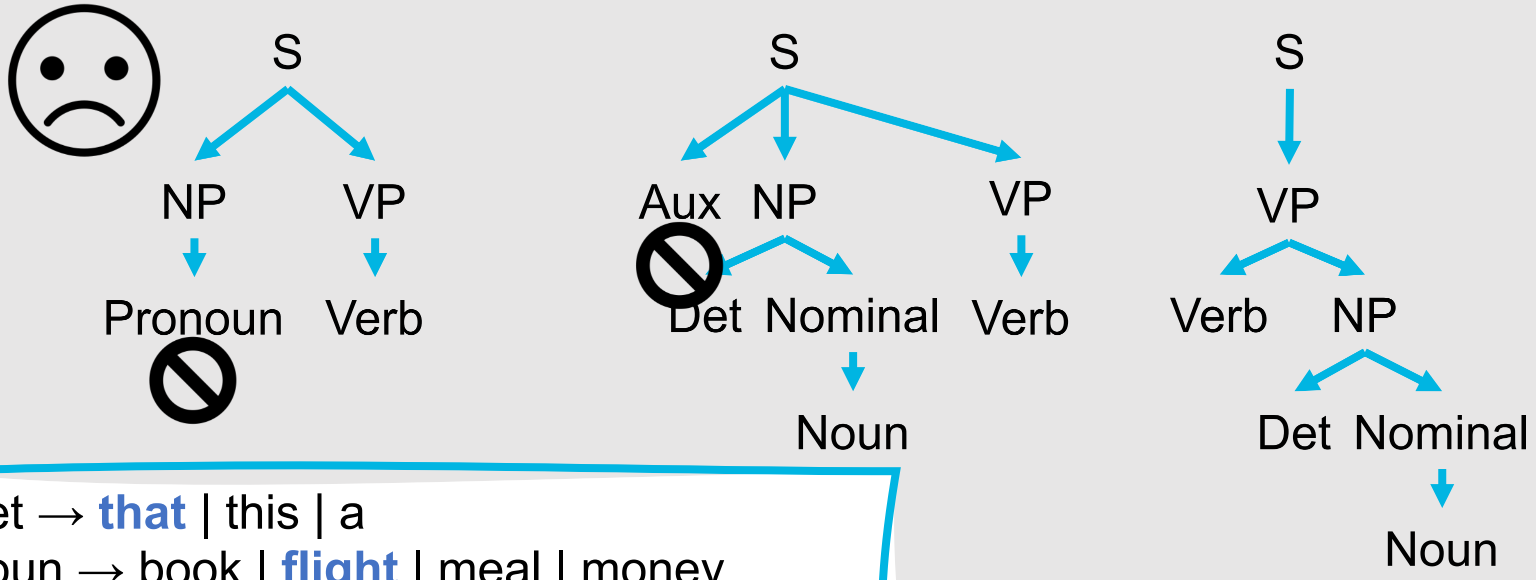
Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

…and many, many more not shown!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



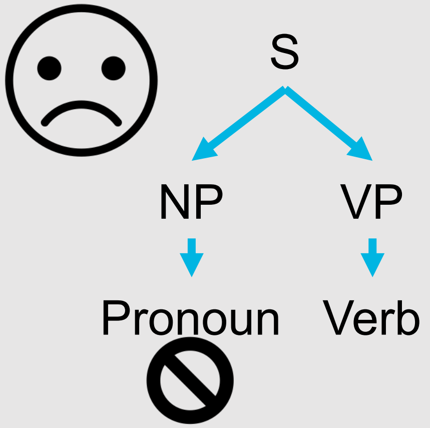
Det → **that** | this | a
Noun → book | **flight** | meal | money
Verb → **book** | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

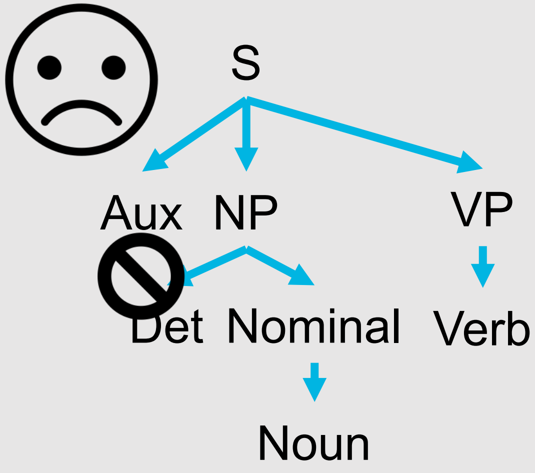
…and many, many more not shown!

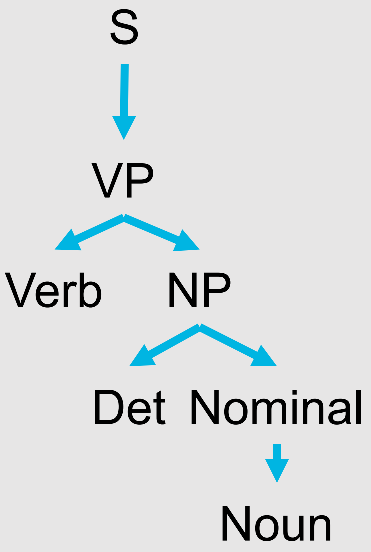
Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP







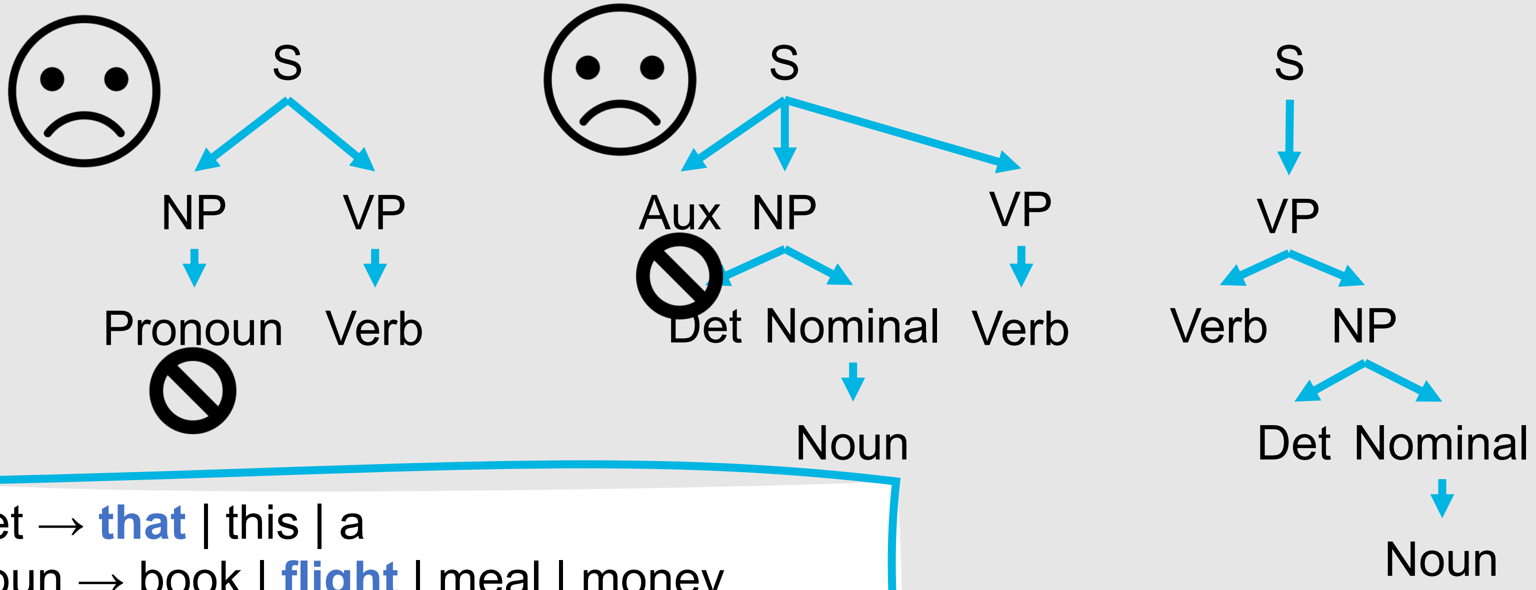
Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

…and many, many more not shown!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



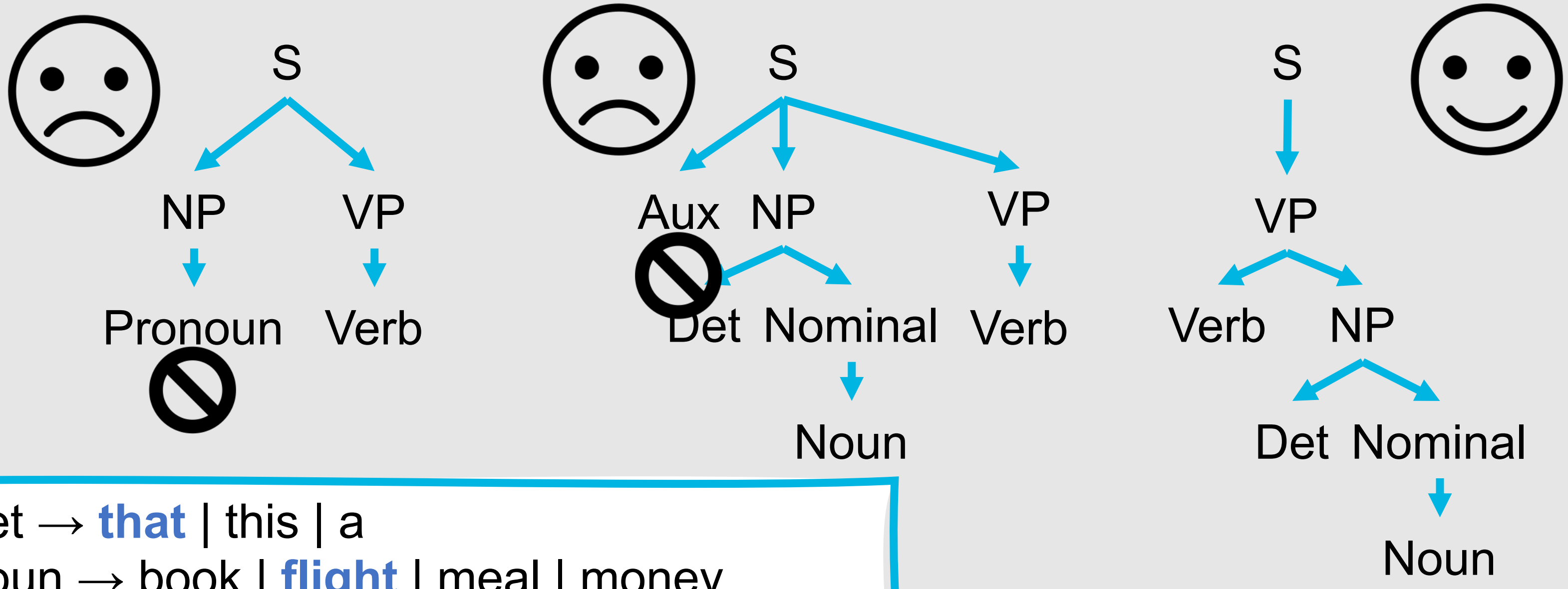
Det → **that** | this | a
Noun → book | **flight** | meal | money
Verb → **book** | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

…and many, many more not shown!

Top-Down Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Det → **that** | this | a
Noun → book | **flight** | meal | money
Verb → **book** | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

…and many, many more not shown!

Bottom-Up Parsing

- Earliest known parsing algorithm!
- Starts with the words in the input sentence, and tries to build trees from those words up by applying rules from the grammar one at a time
  - Looks for places in the in-progress parse where the righthand side of a production rule might fit
- Success = parser builds a tree rooted in the start symbol ***S*** that covers all of the input words

Bottom-Up Parsing: Example

Input Sentence:

Book that flight.

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

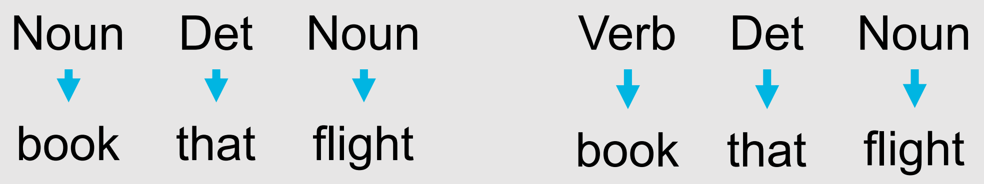
Lexicon:

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

Bottom-Up Parsing: Example

Book that flight.

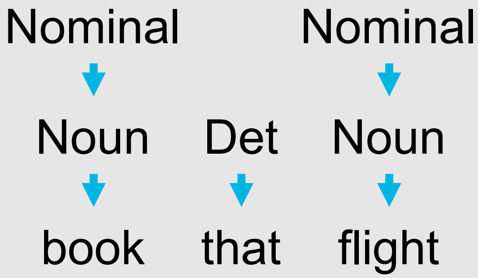
Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

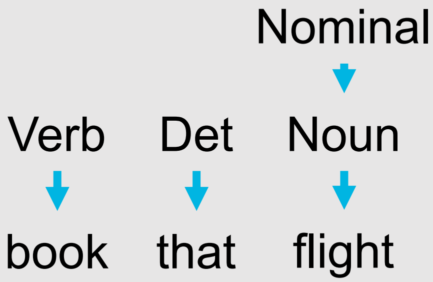


Bottom-Up Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP

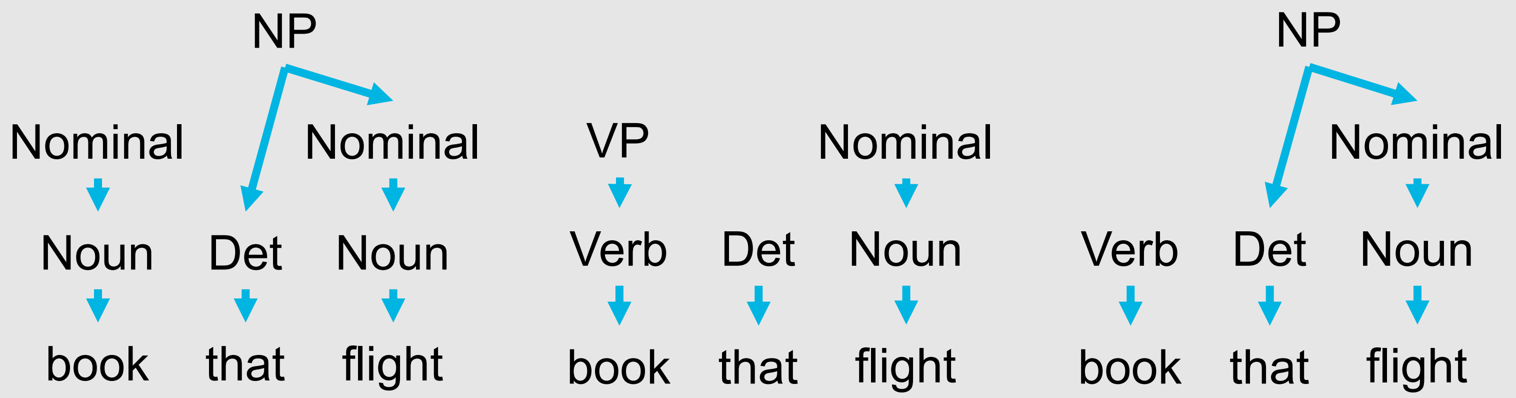




Bottom-Up Parsing: Example

Book that flight.

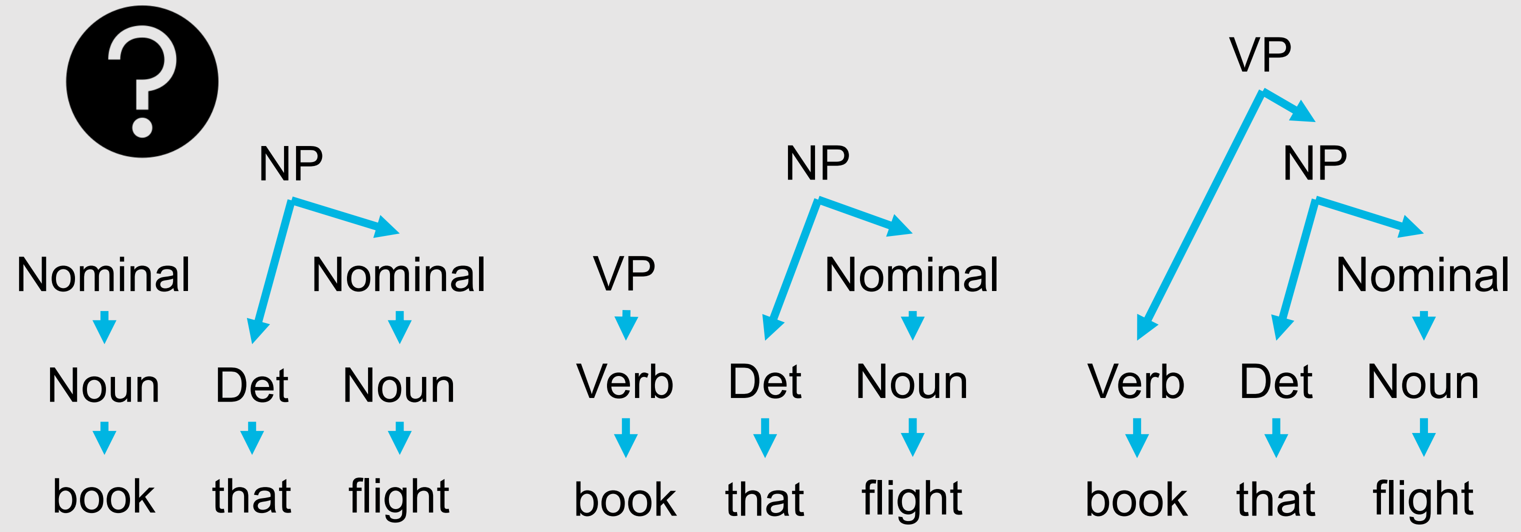
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Bottom-Up Parsing: Example

Book that flight.

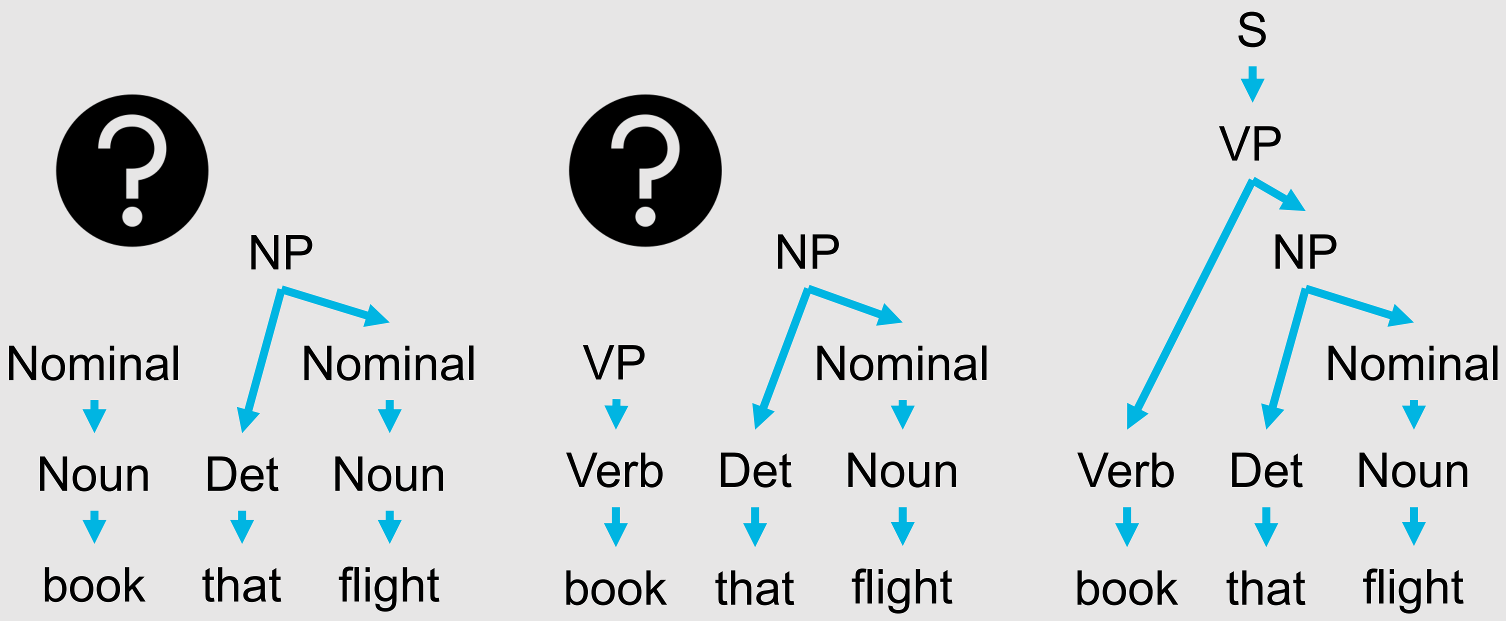
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Bottom-Up Parsing: Example

Book that flight.

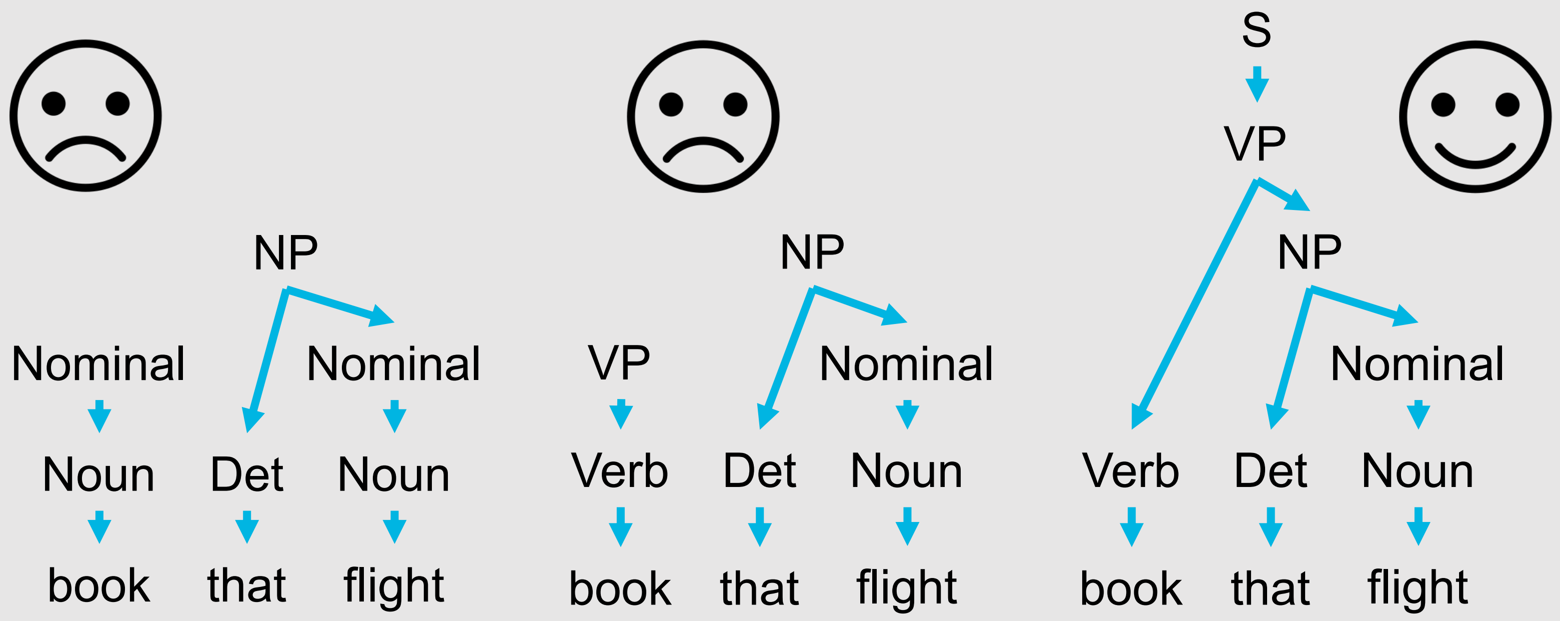
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Bottom-Up Parsing: Example

Book that flight.

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Preposition NP



Top-Down vs. Bottom-Up Parsing

Top-Down Parsing

- Pros:
  - Never wastes time exploring trees that cannot result in a sentence
  - Never explores subtrees that cannot fit into a larger valid (i.e., results in a sentence) tree
- Cons:
  - Spends considerable effort on trees that are not consistent with the input

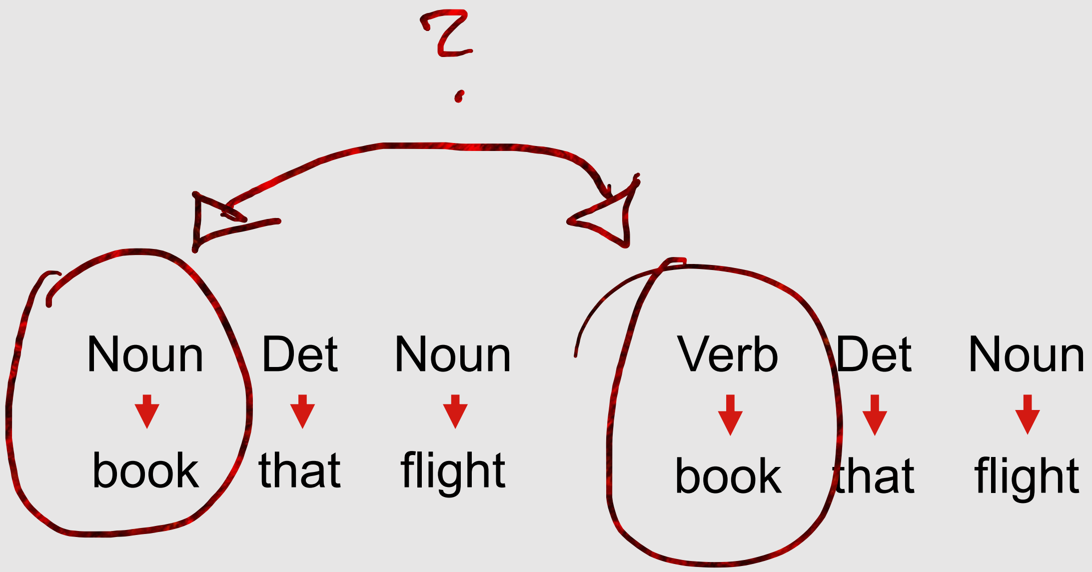
Bottom-Up Parsing

- Pros:
  - Never suggests trees that are inconsistent with the input
- Cons:
  - Generates many trees and subtrees that cannot result in a valid sentence (according to production rules specified by the grammar)

Many forms of ambiguity can arise during syntactic parsing!

- **Structural Ambiguity:** Occurs when a grammar allows for more than one possible parse for a given sentence
- Two Forms:
  - **Attachment Ambiguity:** Occurs when a constituent can be attached to a parse tree at more than one place
    - I eat spaghetti **with chopsticks**.
  - **Coordination Ambiguity:** Occurs when different sets of phrases can be conjoined by a conjunction
    - I grabbed a muffin from the table marked “nut-free scones **and** muffins,” hoping I’d parsed the sign correctly.

Local Ambiguity



- Det → that | this | a
- Noun → **book** | flight | meal | money
- Verb → **book** | include | prefer
- Pronoun → I | she | me
- Proper-Noun → Houston | NWA
- Aux → does
- Preposition → from | to | on | near | through

All of this ambiguity can lead to really complex search spaces!

- **Backtracking** approaches expand the search space incrementally, systematically exploring one state at a time
  - When they arrive at trees inconsistent with the input, they return to an unexplored alternative
  - However, in doing so they tend to discard valid subtrees …this means that time-consuming work needs to be repeated
- More efficient approach?
  - **Dynamic programming**

Dynamic Programming Parsing Methods

- **Tables store subtrees for constituents as they are discovered**
- Solves:
  - Re-parsing problem
  - (Partially) ambiguity problem, since the table implicitly stores all possible parses

Dynamic Programming Parsing Methods

- Widely used methods:
  - Cocke-Kasami-Younger (**CKY**) algorithm
  - **Earley** algorithm

CKY Algorithm

- One of the earliest recognition and parsing algorithms
- **Bottom-up dynamic programming**
- Standard version can only recognize CFGs in **Chomsky Normal Form** (CNF)

Chomsky Normal Form

- Grammars are restricted to production rules of the form:
  - A → B C
  - A → w
- This means that the righthand side of each rule must expand to either two non-terminals or a single terminal
- Any CFG can be converted to a corresponding CNF grammar that accepts exactly the same set of strings as the original grammar!

How does this conversion work?

- Three situations we need to address:
    1. Production rules that mix terminals and non-terminals on the righthand side
    2. Production rules that have a single non-terminal on the righthand side (**unit productions**)
    3. Production rules that have more than two non-terminals on the righthand side
- Situation #1: **Introduce a dummy non-terminal that covers only the original terminal**
    - INF-VP → to VP could be replaced with INF-VP → TO VP and TO → to
- Situation #2: **Replace the non-terminals with the non-unit production rules to which they eventually lead**
    - A → B and B → w could be replaced with A → w
- Situation #3: **Introduce new non-terminals that spread longer sequences over multiple rules**
    - A → B C D could be replaced with A → B X1 and X1 → C D

CNF Conversion: Example

- S → NP VP
- S → AdjP NP VP
- S → VP
- NP → Pronoun
- NP → Proper-Noun
- NP → Det Nominal
- Nominal → Noun
- Nominal → Nominal Noun
- Nominal → Nominal PP
- VP → Verb
- VP → Verb NP
- VP → Verb NP PP
- VP → Verb PP
- VP → VP PP
- PP → Preposition NP

Original	CNF
S → NP VP	S → NP VP
S → AdjP NP VP	S → X1 VP
	X1 → AdjP NP
S → VP	S → book | include | prefer

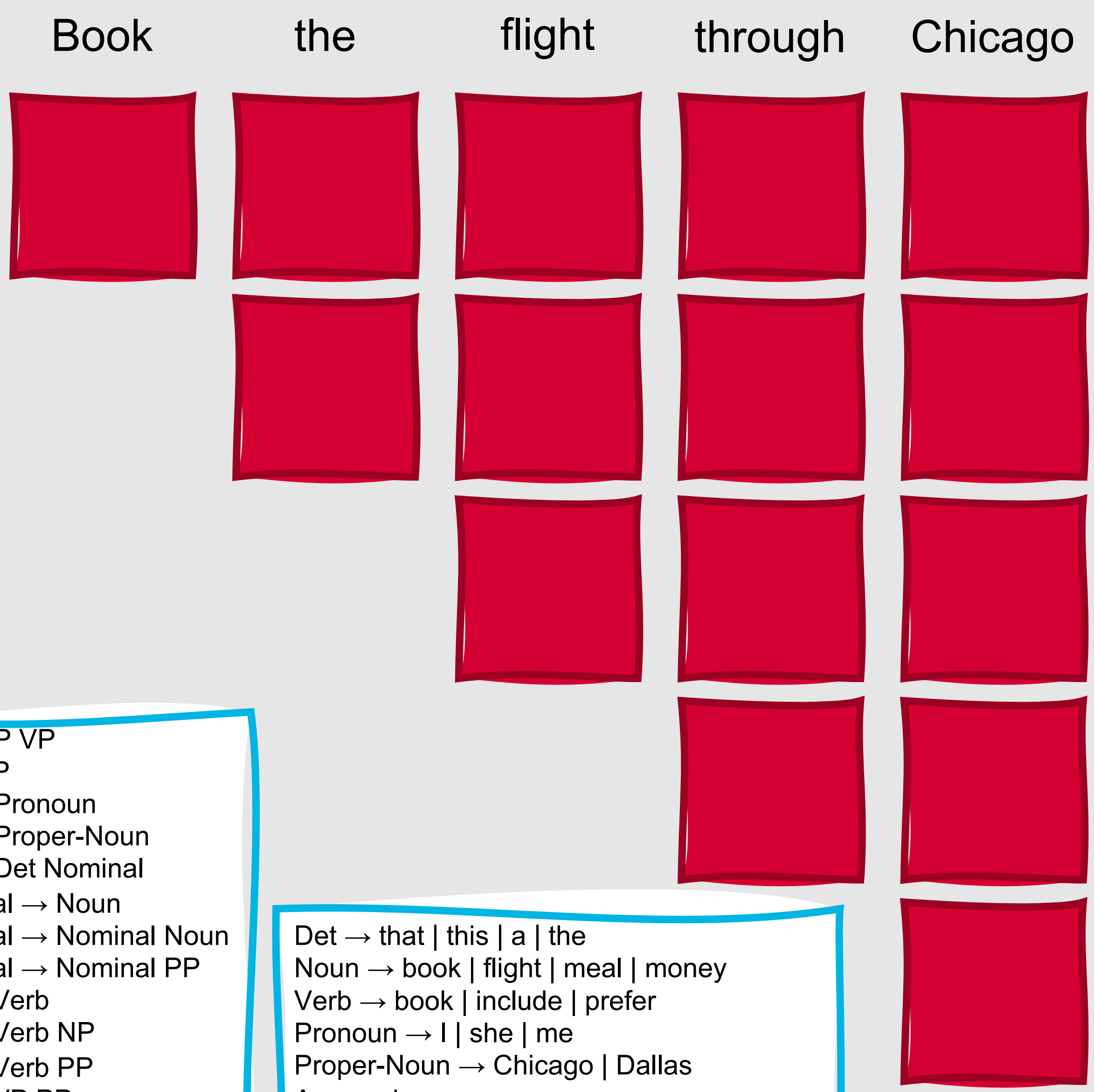
CKY Algorithm

- With the grammar in CNF, each non-terminal node above the POS level of the parse tree will have exactly two children
- Thus, a two-dimensional matrix can be used to encode the tree structure
- For sentence of length n, work with upper-triangular portion of $(n+1)$ x $(n+1)$ matrix
- Each cell $[i,j]$ contains a set of non-terminals that represent all constituents spanning positions i through j of the input
  - Cell that represents the entire input resides in position $[0,n]$

CKY Algorithm

- Non-terminal entries: For each constituent [*i*,*j*], there is a position, *k*, where the constituent can be split into two parts such that $i < k < j$
  - [*i*,*k*] must lie to the left of [*i*,*j*] somewhere along row *i*, and [*k*,*j*] must lie beneath it along column *j*
- To fill in the parse table, we proceed in a bottom-up fashion so when we fill a cell [*i*,*j*], the cells containing the parts that could contribute to this entry have already been filled

CKY Algorithm: Example



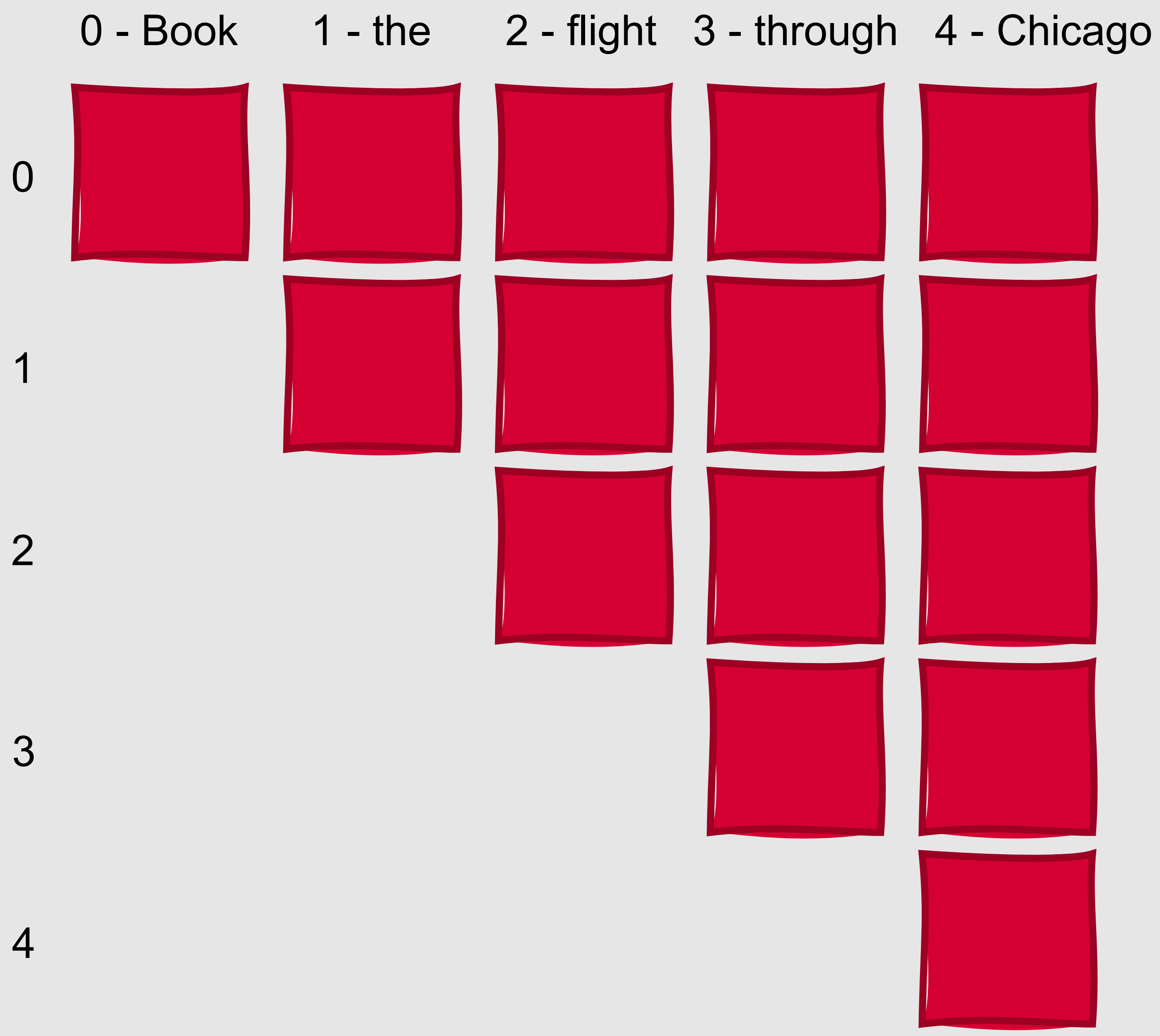
S → NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → **book** | flight | meal | money
Verb → **book** | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb				
1					
2					
3					
4					

CKY Algorithm: Example

Det → that | this | a | **the**
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb				
1		Det			
2					
3					
4					

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | **flight** | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb				
1		Det			
2			Noun		
3					
4					

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | **through**

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb				
1		Det			
2			Noun		
3				Prep.	
4					

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → **Chicago** | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb				
1		Det			
2			Noun		
3				Prep.	
4					PropN

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → **book** | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → **book** | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → **book** | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det			
2			Noun		
3				Prep.	
4					PropN

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | **flight** | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det			
2			Noun, Nominal		
3				Prep.	
4					PropN

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

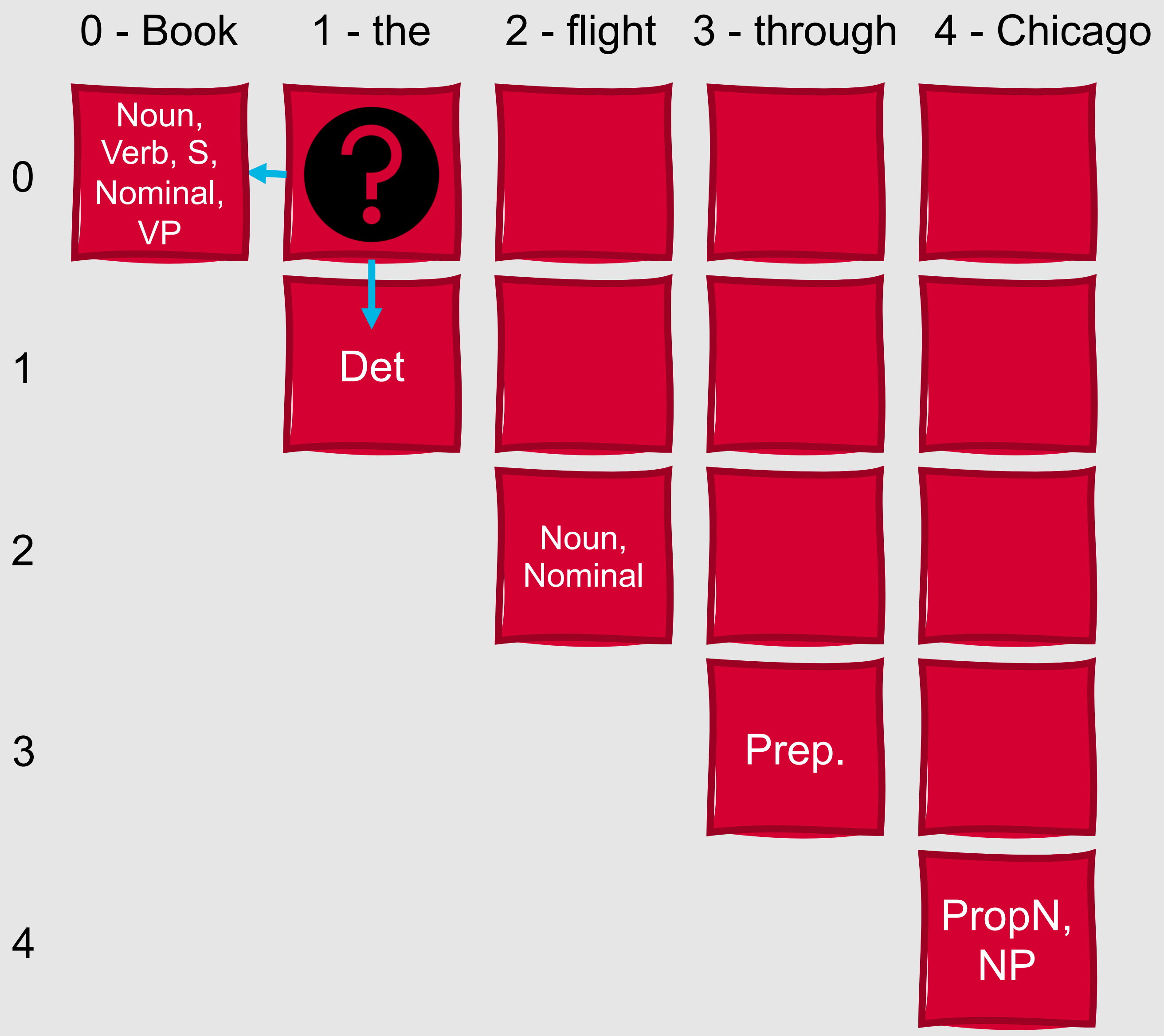
S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → **Chicago** | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det			
2			Noun, Nominal		
3				Prep.	
4					PropN, NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det	?		
2			Noun, Nominal		
3				Prep.	
4					PropN, NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → **Det Nominal**
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det	NP		
2			Noun, Nominal		
3				Prep.	
4					PropN, NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

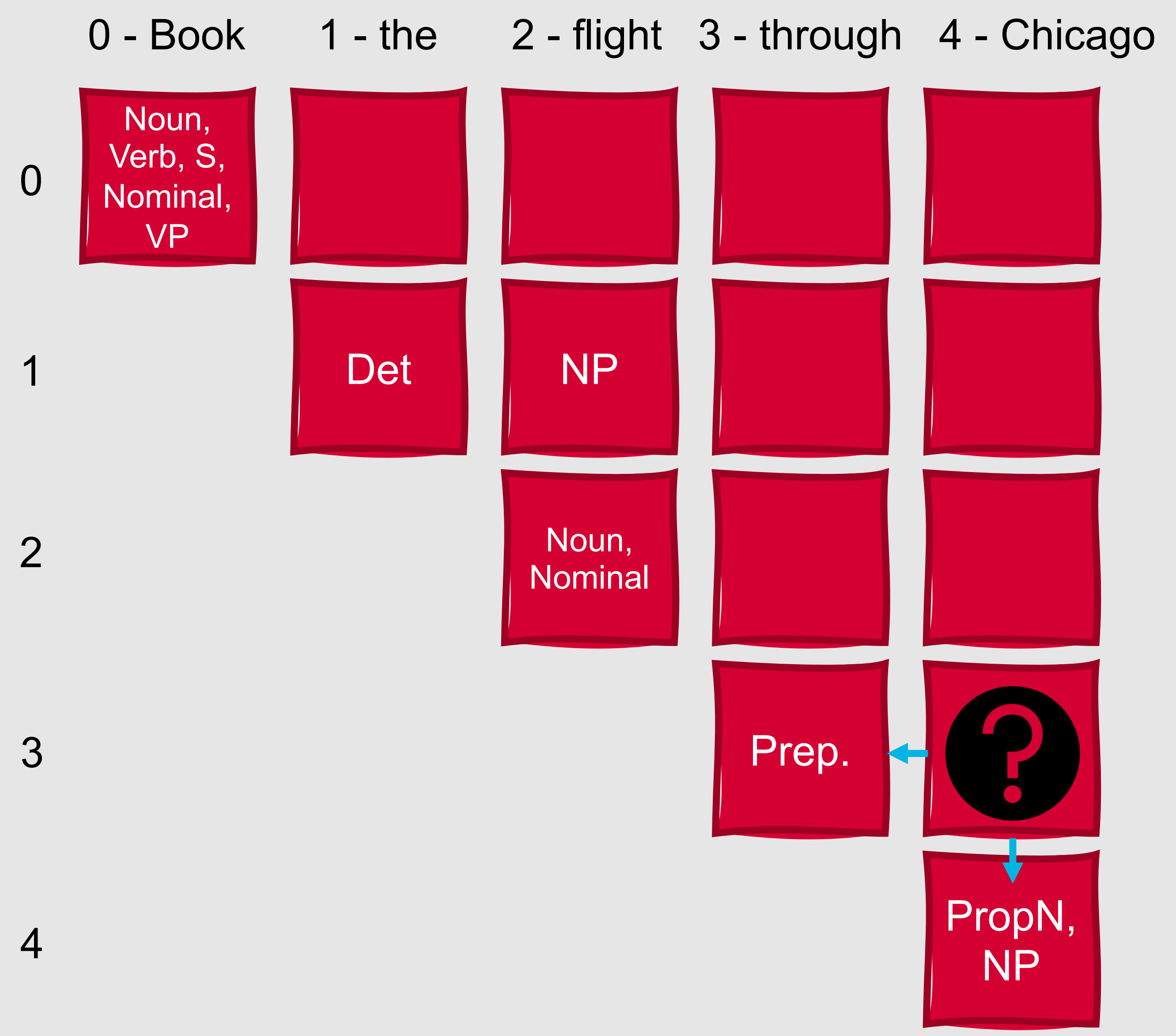
S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det	NP		
2			Noun, Nominal	?	
3				Prep.	
4					PropN, NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

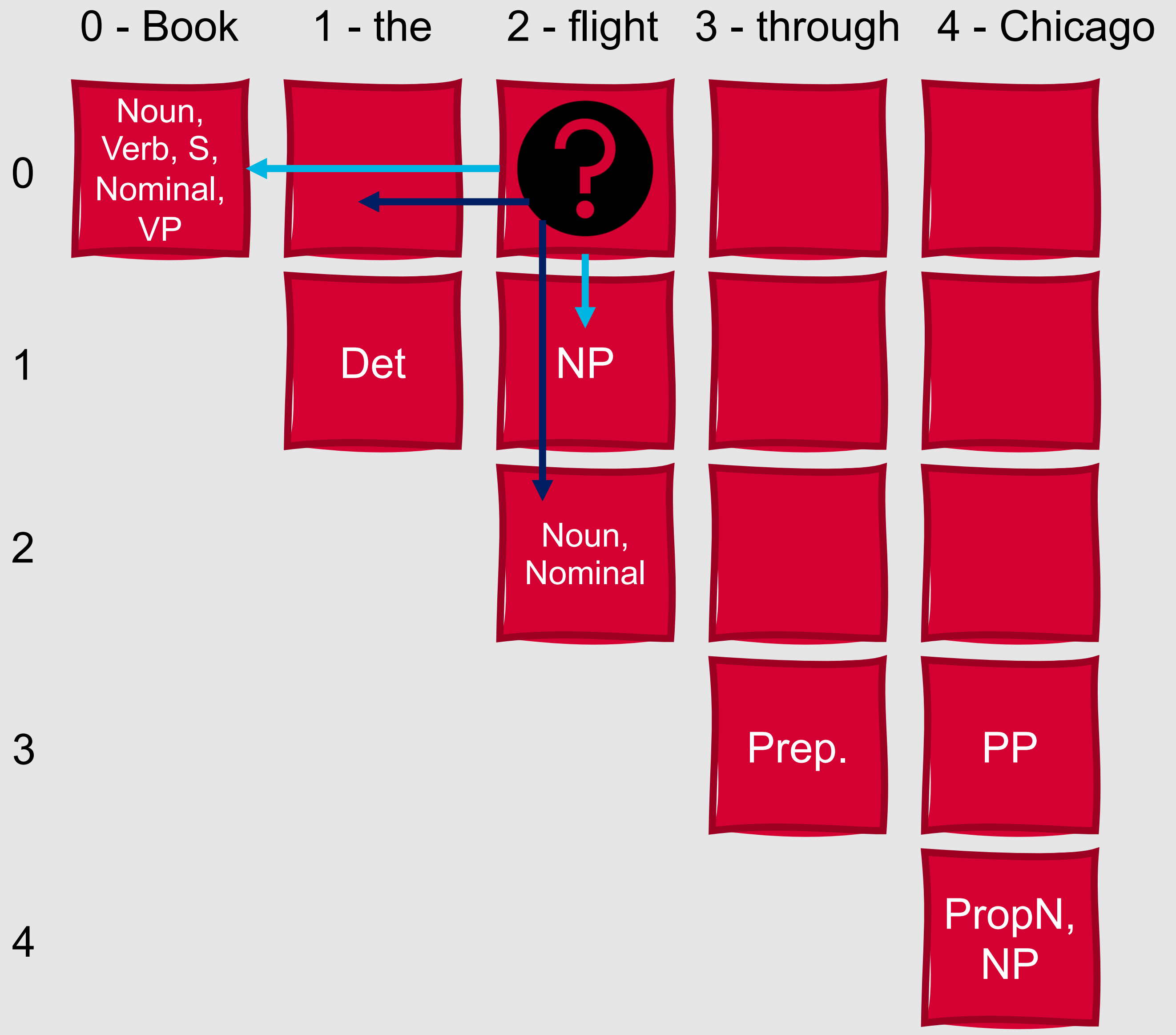
S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → **Preposition NP**

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP				
1		Det	NP		
2			Noun, Nominal		
3				Prep.	PP
4					PropN, NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → **Verb NP**
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → **Verb PP**
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

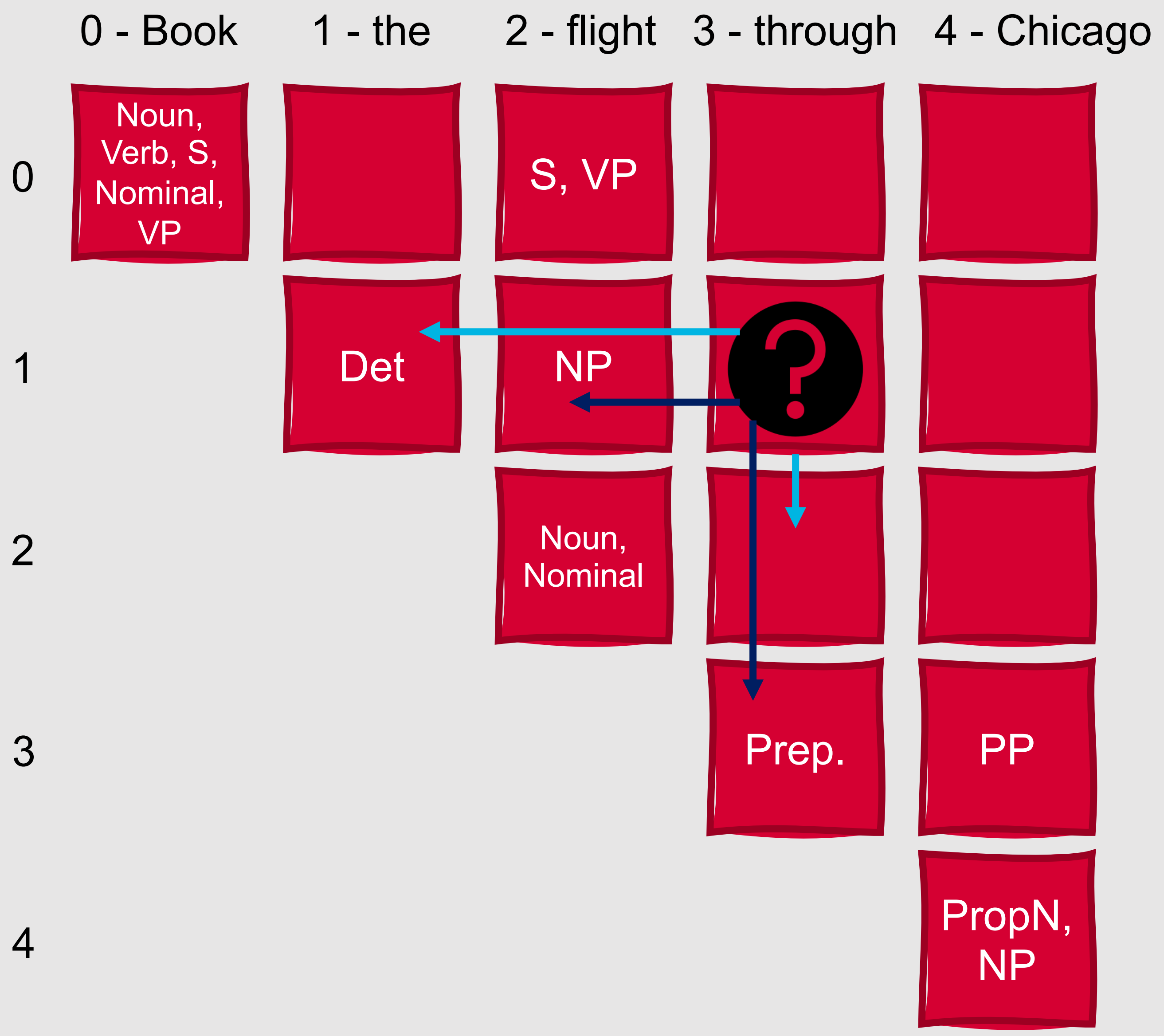
S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → **Verb NP**
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP		S, VP		
1		Det	NP		
2			Noun, Nominal		
3				Prep.	PP
4					PropN, NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP		S, VP		
1		Det	NP		
2			Noun, Nominal		?
3				Prep.	PP
4					PropN, NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

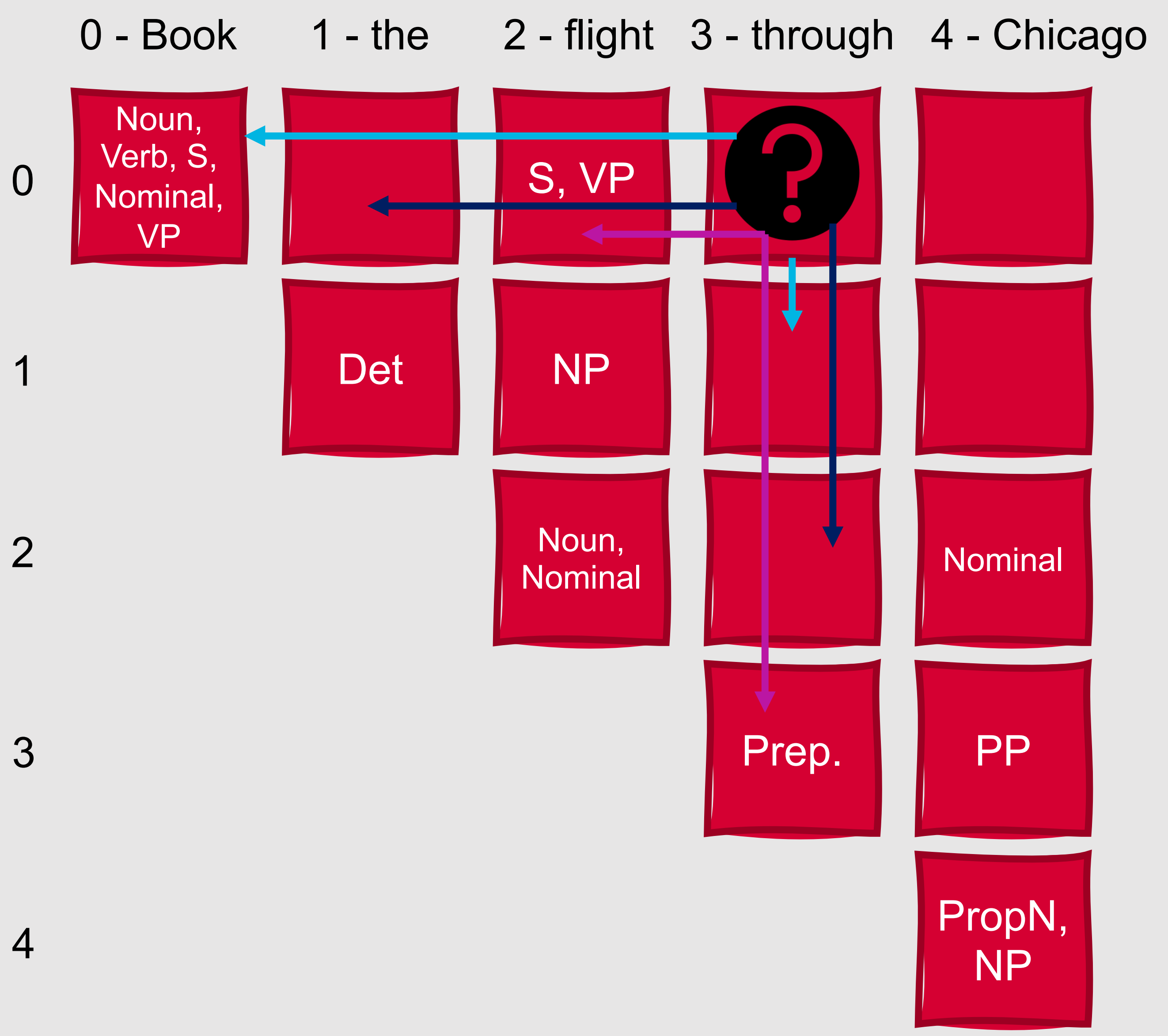
S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → **Nominal PP**
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP		S, VP		
1		Det	NP		
2			Noun, Nominal		Nominal
3				Prep.	PP
4					PropN, NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

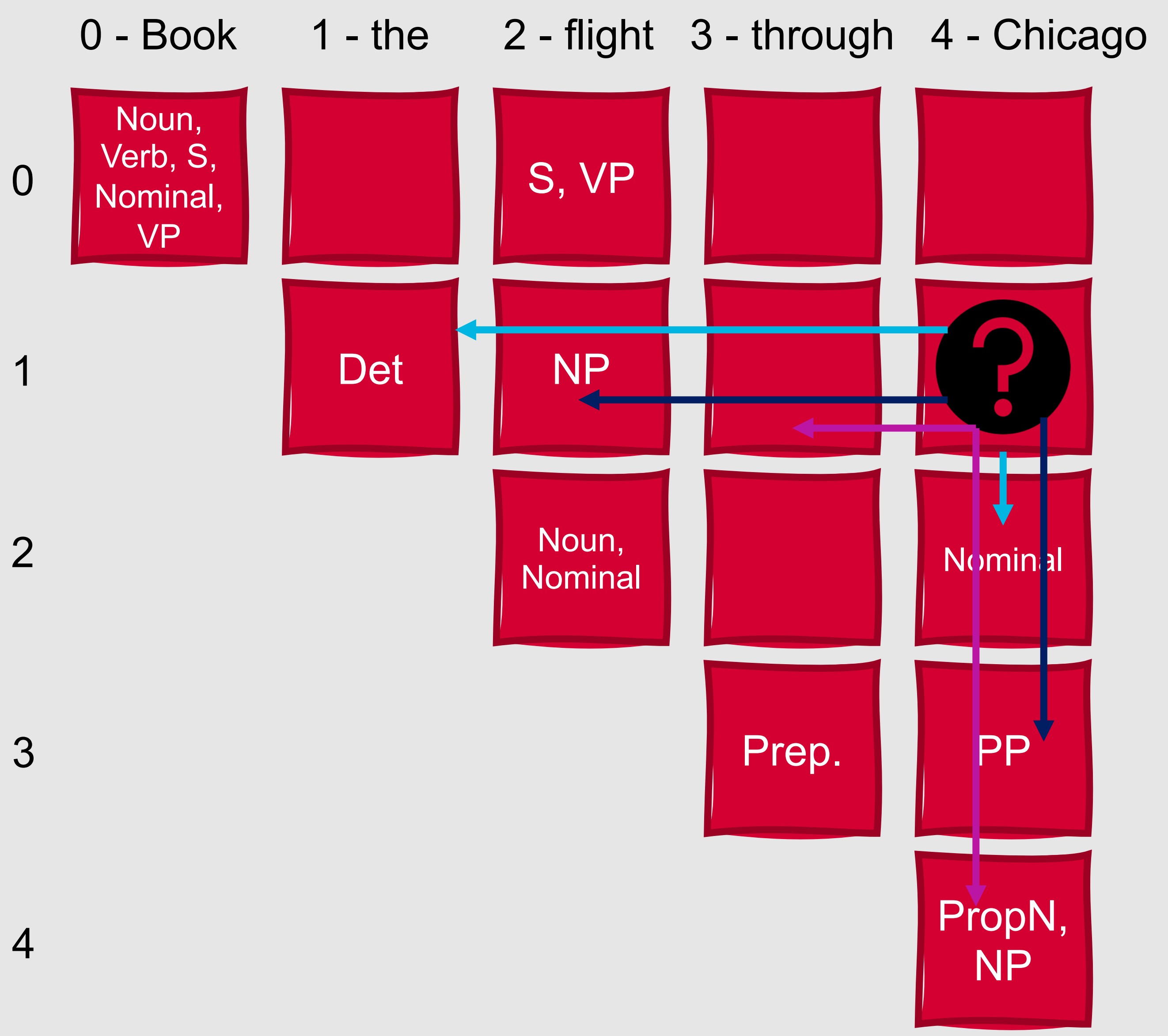
S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

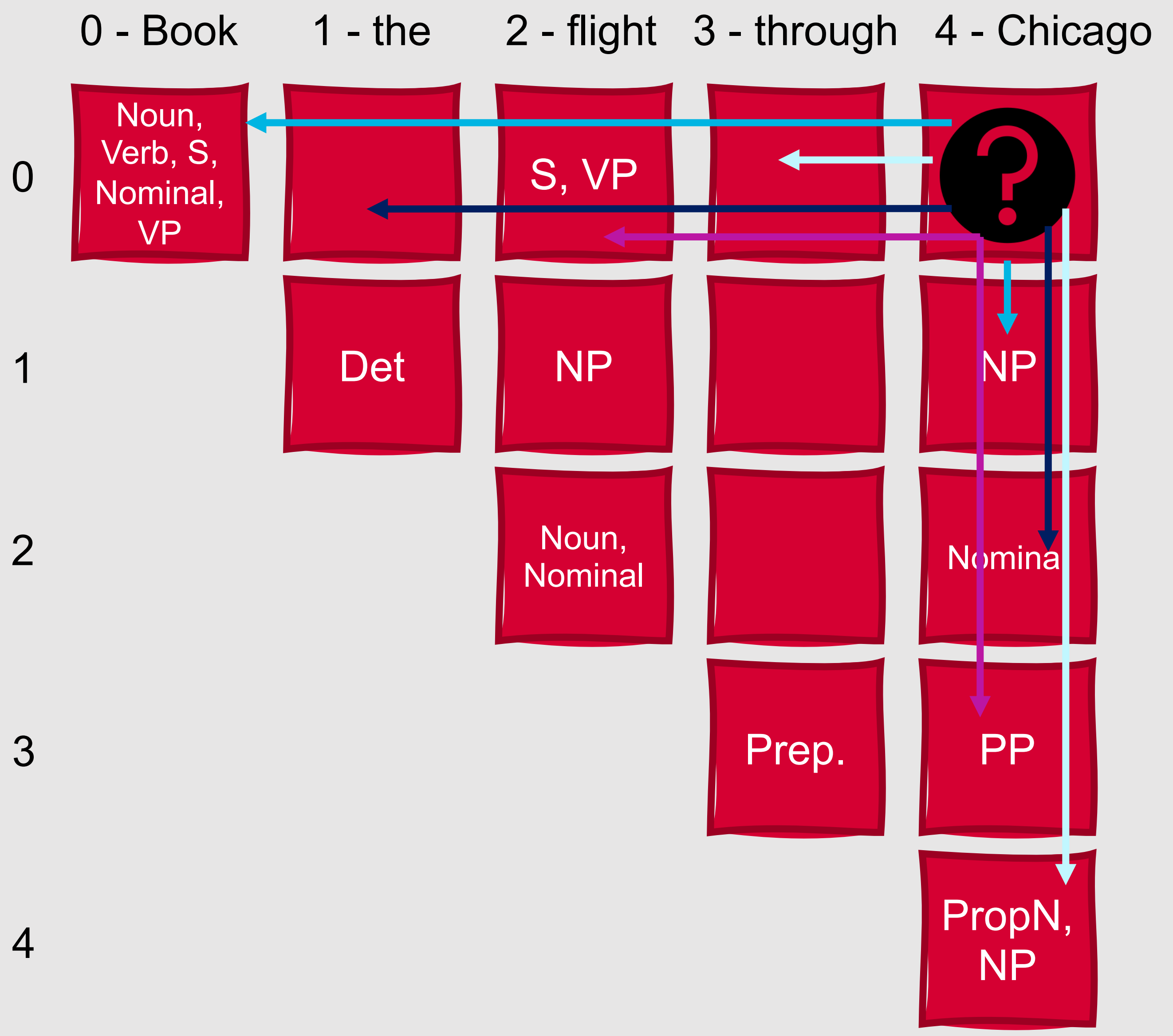
S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → **Det Nominal**
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP		S, VP		
1		Det	NP		NP
2			Noun, Nominal		Nominal
3				Prep.	PP
4					PropN, NP

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP



CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → **Verb NP**
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → **Verb NP**
VP → Verb PP
VP → **VP PP**
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP		S, VP		S, VP
1		Det	NP		NP
2			Noun, Nominal		Nominal
3				Prep.	PP
4					PropN, NP

If we arrive at S in multiple ways, each way is an alternative parse (S_1, S_2, …, S_n).

CKY Algorithm

- The example we just saw functions as a **recognizer** …for it to succeed (i.e., find a valid sentence according to this grammar), is simply needs to find an S in cell [0,n]
- To return all possible parses, we need to make two changes to the algorithm:
  - Pair each non-terminal with pointers to the table entries from which it was derived
  - Permit multiple versions of the same non-terminal to be entered into the table
- Then, we can choose an S from cell [0,n] and recursively retrieve its component constituents from the table

CKY Algorithm: Example

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Chicago | Dallas
Aux → does
Preposition → from | to | on | near | through

S → NP VP
S → VP → Verb → book | include | prefer
S → Verb NP
NP → Pronoun → I | she | me
NP → Proper-Noun → Chicago | Dallas
NP → Det Nominal
Nominal → Noun → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb → book | include | prefer
VP → Verb NP
VP → Verb PP
VP → VP PP
PP → Preposition NP

	0 - Book	1 - the	2 - flight	3 - through	4 - Chicago
0	Noun, Verb, S, Nominal, VP		S, VP		S, VP
1		Det	NP		NP
2			Noun, Nominal		Nominal
3				Prep.	PP
4					PropN, NP

CKY Complexity

Time Complexity: $O(n^3)$

Space Complexity: $O(n^2)$

Earley Parsing

- Top-down dynamic parsing approach
- Table is length *n*+1, where *n* is equivalent to the number of words
- Table entries contain three types of information:
  - A subtree corresponding to a single grammar rule
  - Information about the progress made in completing the subtree
  - The position of the subtree with respect to the input

In Earley parsing, table entries are known as states.

- States include structures called **dotted rules**
- A **•** within the righthand side of a state's grammar rule indicates the progress made towards recognizing it
- A state's **position with respect to the input is represented by two numbers**, indicating (1) where the state begins, and (2) where its dot lies

Example States

- Input: Book that flight.
- S → • VP, [0,0]
  - Top-down prediction for this particular kind of S
  - First 0: Constituent predicted by this state should begin at the start of the input
  - Second 0: Dot lies at the start of the input as well
- NP → Det • Nominal, [1,2]
  - NP begins at position 1
  - Det has been successfully parsed
  - Nominal is expected next
- VP → V NP •, [0,3]
  - Successful discovery of a tree corresponding to a VP that spans the entire input

Earley Algorithm

- An Earley parser moves through the n+1 sets of states in a chart in order
- At each step, one of three operators is applied to each state depending on its status
  - Predictor
  - Scanner
  - Completer
- States can be added to the chart, but are never removed
- The algorithm never backtracks
- The presence of S $\rightarrow \alpha \bullet$, [0,$n$] indicates a successful parse

Earley Operators: Predictor

Predictor

- Creates new states
- Applied to any state that has a non-terminal immediately to the right of its dot (as long as the non-terminal is not a POS category)
- New states are placed into the same chart entry as the generating state
- They begin and end at the same point in the input where the generating state ends

S → • VP, [0,0]

- VP → • Verb, [0,0]
- VP → • Verb NP, [0,0]
- VP → • Verb NP PP, [0,0]
- VP → • Verb PP, [0,0]
- VP → • VP PP, [0,0]

Earley Operators: Scanner

- Used when a state has a POS category to the right of the dot
- Examines input and incorporates a state corresponding to the prediction of a word with a particular POS into the chart
- VP → • Verb NP, [0,0]
    - Since category following the dot is a part of speech (Verb)….
    - Verb → book •, [0,1]

Earley Operators: Completer

- Applied to a state when its dot has reached the right end of the rule
- Indicates that the parser has successfully discovered a particular grammatical category over some span of input
- Finds all previously created states that were searching for this grammatical category, and creates new states that are copies with their dots advanced past the grammatical category
- NP → Det Nominal •, [1,3]
  - What incomplete states end at position 1 and expect an NP?
  - VP → Verb • NP, [0,1]
  - VP → Verb • NP PP, [0,1]
  - So, add VP → Verb NP •, [0,3] and the new incomplete VP → Verb NP • PP, [0,3] to the chart

Earley Algorithm: Example

Chart	State	Rule	Start, End	Added By
0	S0	γ → • S	0, 0	Start State
0	S1	S → • NP VP	0, 0	Predictor
0	S2	S → • VP	0, 0	Predictor
0	S3	NP → • Det Nominal	0, 0	Predictor
0	S4	VP → • Verb	0, 0	Predictor
0	S5	VP → • Verb NP	0, 0	Predictor

- Book that flight.

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer

S → NP VP
S → VP
NP → Det Nominal
Nominal → Noun
VP → Verb
VP → Verb NP

Earley Algorithm: Example

Book • that flight.

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer

S → NP VP
S → VP
NP → Det Nominal
Nominal → Noun
VP → Verb
VP → Verb NP

Chart	State	Rule	Start, End	Added By
0	S0	γ → • S	0, 0	Start State
0	S1	S → • NP VP	0, 0	Predictor
0	S2	S → • VP	0, 0	Predictor
0	S3	NP → • Det Nominal	0, 0	Predictor
0	S4	VP → • Verb	0, 0	Predictor
0	S5	VP → • Verb NP	0, 0	Predictor
1	S6	Verb → book •	0, 1	Scanner
1	S7	VP → Verb •	0, 1	Completer
1	S8	VP → Verb • NP	0, 1	Completer
1	S9	S → VP •	0, 1	Completer
1	S10	NP → • Det Nominal	1, 1	Predictor

Earley Algorithm: Example

Book that • flight.

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer

S → NP VP
S → VP
NP → Det Nominal
Nominal → Noun
VP → Verb
VP → Verb NP

Chart	State	Rule	Start, End	Added By
0	S0	γ → • S	0, 0	Start State
0	S1	S → • NP VP	0, 0	Predictor
0	S2	S → • VP	0, 0	Predictor
0	S3	NP → • Det Nominal	0, 0	Predictor
0	S4	VP → • Verb	0, 0	Predictor
0	S5	VP → • Verb NP	0, 0	Predictor
1	S6	Verb → book •	0, 1	Scanner
1	S7	VP → Verb •	0, 1	Completer
1	S8	VP → Verb • NP	0, 1	Completer
1	S9	S → VP •	0, 1	Completer
1	S10	NP → • Det Nominal	1, 1	Predictor
2	S11	Det → that •	1, 2	Scanner
2	S12	NP → Det • Nominal	1, 2	Completer
2	S13	Nominal → • Noun	2, 2	Predictor

Earley Algorithm: Example

Book that flight. •

Det → that | this | a | the
Noun → book | flight | meal | money
Verb → book | include | prefer

S → NP VP
S → VP
NP → Det Nominal
Nominal → Noun
VP → Verb
VP → Verb NP

Chart	State	Rule	Start, End	Added By
0	S0	γ → • S	0, 0	Start State
0	S1	S → • NP VP	0, 0	Predictor
0	S2	S → • VP	0, 0	Predictor
0	S3	NP → • Det Nominal	0, 0	Predictor
0	S4	VP → • Verb	0, 0	Predictor
0	S5	VP → • Verb NP	0, 0	Predictor
1	S6	Verb → book •	0, 1	Scanner
1	S7	VP → Verb •	0, 1	Completer
1	S8	VP → Verb • NP	0, 1	Completer
1	S9	S → VP •	0, 1	Completer
1	S10	NP → • Det Nominal	1, 1	Predictor
2	S11	Det → that •	1, 2	Scanner
2	S12	NP → Det • Nominal	1, 2	Completer
2	S13	Nominal → • Noun	2, 2	Predictor
3	S14	Noun → flight •	2, 3	Scanner
3	S15	Nominal → Noun •	2, 3	Completer
3	S16	NP → Det Nominal •	1, 3	Completer
3	S17	VP → Verb NP •	0, 3	Completer
3	S18	S → VP •	0, 3	Completer

Which states participate in the final parse?

Chart	State	Rule	Start, End	Added By
0	S0	$\gamma \rightarrow$ • S	0, 0	Start State
0	S1	S → • NP VP	0, 0	Predictor
0	S2	S → • VP	0, 0	Predictor
0	S3	NP → • Det Nominal	0, 0	Predictor
0	S4	VP → • Verb	0, 0	Predictor
0	S5	VP → • Verb NP	0, 0	Predictor
1	S6	Verb → book •	0, 1	Scanner
1	S7	VP → Verb •	0, 1	Completer
1	S8	VP → Verb • NP	0, 1	Completer
1	S9	S → VP •	0, 1	Completer
1	S10	NP → • Det Nominal	1, 1	Predictor
2	S11	Det → that •	1, 2	Scanner
2	S12	NP → Det • Nominal	1, 2	Completer
2	S13	Nominal → • Noun	2, 2	Predictor
3	S14	Noun → flight •	2, 3	Scanner
3	S15	Nominal → Noun •	2, 3	Completer
3	S16	NP → Det Nominal •	1, 3	Completer
3	S17	VP → Verb NP •	0, 3	Completer
3	S18	S → VP •	0, 3	Completer

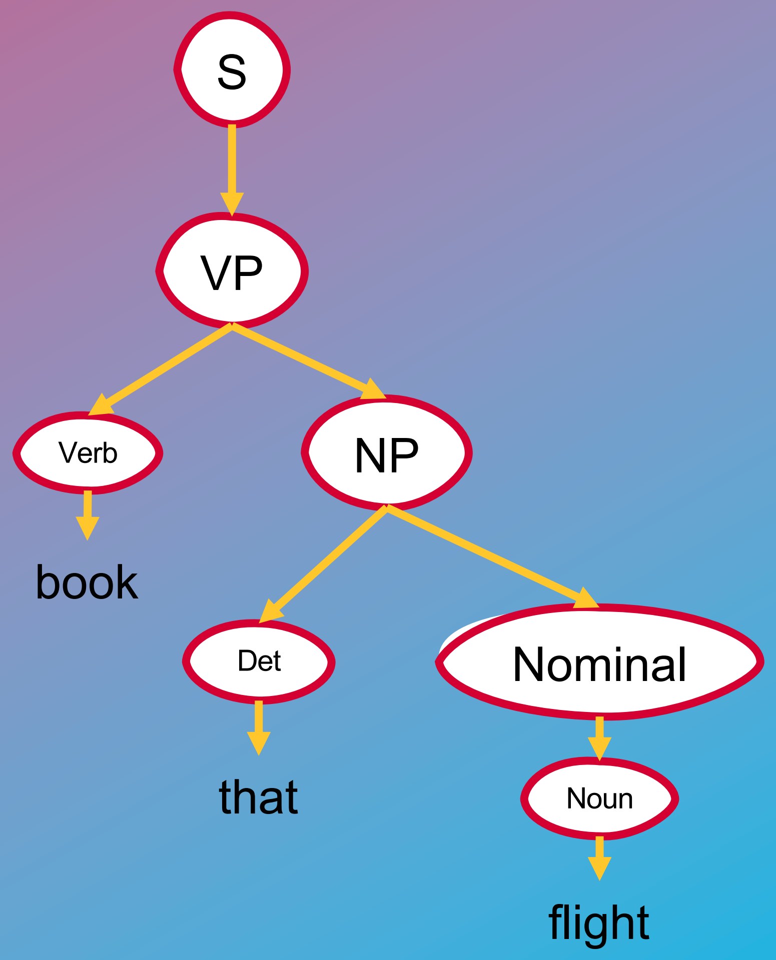
As with CKY, the example algorithm acted as a recognizer.

- We can retrieve parse trees by adding a field to store information about the completed states that generated constituents
- How to do this?
  - Have the Completer operator add a pointer to the previous state onto a list of constituent states for the new state
  - When an S is found in the final chart, just follow pointers backward

Which states participate in the final parse?

Chart	State	Rule	Start, End	Added By (Backward Pointer)
0	S0	γ → • S	0, 0	Start State
0	S1	S → • NP VP	0, 0	Predictor
0	S2	S → • VP	0, 0	Predictor
0	S3	NP → • Det Nominal	0, 0	Predictor
0	S4	VP → • Verb	0, 0	Predictor
0	S5	VP → • Verb NP	0, 0	Predictor
1	S6	Verb → book •	0, 1	Scanner
1	S7	VP → Verb •	0, 1	Completer
1	S8	VP → Verb • NP	0, 1	Completer
1	S9	S → VP •	0, 1	Completer
1	S10	NP → • Det Nominal	1, 1	Predictor
2	S11	Det → that •	1, 2	Scanner
2	S12	NP → Det • Nominal	1, 2	Completer
2	S13	Nominal → • Noun	2, 2	Predictor
3	S14	Noun → flight •	2, 3	Scanner
3	S15	Nominal → Noun •	2, 3	Completer (S14)
3	S16	NP → Det Nominal •	1, 3	Completer (S11, S15)
3	S17	VP → Verb NP •	0, 3	Completer (S6, S16)
3	S18	S → VP •	0, 3	Completer (S17)

Successful Earley Parse

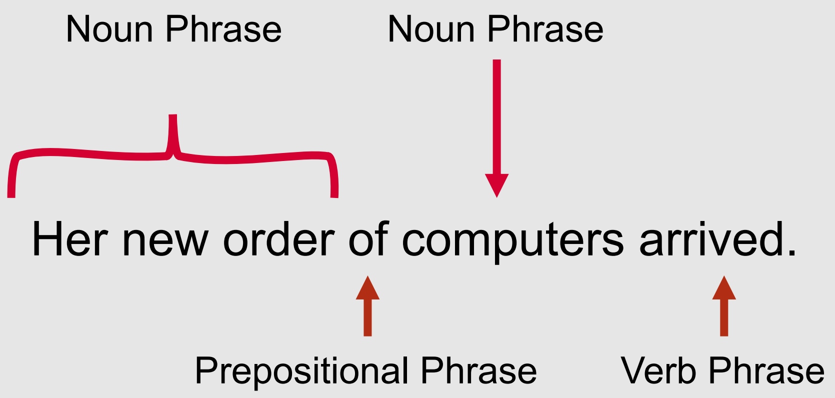


Summary: Syntactic Parsing

- **Syntactic parsing** is the process of automatically determining the grammatical structure of an input sentence
- Language is ambiguous, so **sentences can have multiple grammatically-correct parses**
- Parsing can be performed using either a **top-down** or **bottom-up** approach
- Common algorithms for syntactic parsing include:
  - **CKY**
  - **Earley**

What if we don't need a full parse tree?

- Full parse trees can be complex and time-consuming to build
- Many NLP tasks don't require full hierarchical parses

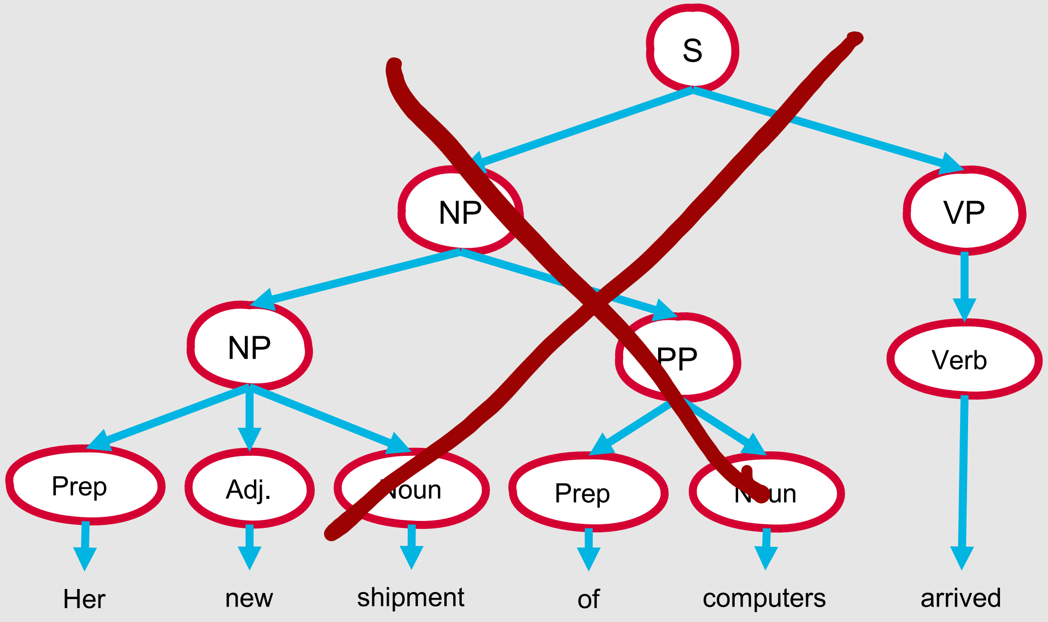


Easier solution?

- **Partial parsing**, or **shallow parsing**
- How to generate a partial parse?
  - Cascades of finite state transducers
  - **Chunking**

Chunking

- Process of finding the non-overlapping, non-recursive constituents in an input text
  - Noun phrases
  - Verb phrases
  - Prepositional phrases
  - Adjective phrases



[Her new shipment]$_{NP}$ [of]$_{PP}$ [computers]$_{NP}$ [arrived]$_{VP}$

Chunking: Fundamental Tasks

Segmentation: Identify the non-overlapping, fundamental phrases

[Her new order] [of] [computers] [arrived]

Labeling: Assign labels to those phrases

[Her new order]$_{NP}$ [of]$_{PP}$ [computers]$_{NP}$ [arrived]$_{VP}$

What is, and is not, a chunk?

- Depends on the task!
- General guidelines:
  - Non-recursive
  - When chunking phrases that would otherwise be parsed recursively:
    - Keep head word
    - Keep all material belonging to constituent that occurs before the head word

[Her new shipment of computers]$_{NP}$ [arrived]$_{VP}$

[Her new shipment]$_{NP}$ [of]$_{PP}$ [computers]$_{NP}$ [arrived]$_{VP}$

How do we segment text into chunks?

- **IOB tagging**
  - **I:** Tokens **inside** a chunk
  - **O:** Tokens **outside** any chunk
  - **B:** Tokens **beginning** a chunk
- Generally framed as a **sequence labeling** task

Task: IOB Tagging (All Constituent Types)

B_NP I_NP B_NP

Her new order of computers arrived.

I_NP B_PP B_VP

Task: IOB Tagging (Noun Phrases)

B_NP I_NP B_NP

Her new order of computers arrived.

I_NP O O

How do we evaluate chunking systems?

- Standard text classification metrics, comparing predictions with a gold standard
  - Precision
  - Recall
  - F-measure

With the parsing techniques so far, we've seen a variety of ways to represent ambiguity.

- CKY algorithm
- Earley algorithm
- Partial parsing
- However …what's an effective way to resolve ambiguities?
  - **Probabilistic context-free grammars (PCFGs)**

Probabilistic Context-Free Grammars

- Can be used to determine which parse out of multiple valid parses should be selected, based on how likely the parse tree is to occur in a large corpus
- Same core components as regular CFGs:
  - A set of non-terminals, N
  - A set of terminal symbols, Σ
  - A set of rules or productions, R
  - A designated start symbol, S
- Each rule in R is of the form $A \rightarrow \beta$, where A is a non-terminal and β is a string of symbols from the set $\Sigma \cup N$

How do PCFGs differ from CFGs?

- R is augmented with a probability, [p], learned from a corpus
- The sum of all probabilities for a given non-terminal is 1.0
- For example, if the following three expansions for S were possible, they might have the probabilities:
  - S → NP VP [0.80]
  - S → Aux NP VP [0.15]
  - S → VP [0.05]

Probabilistic Context-Free Grammars

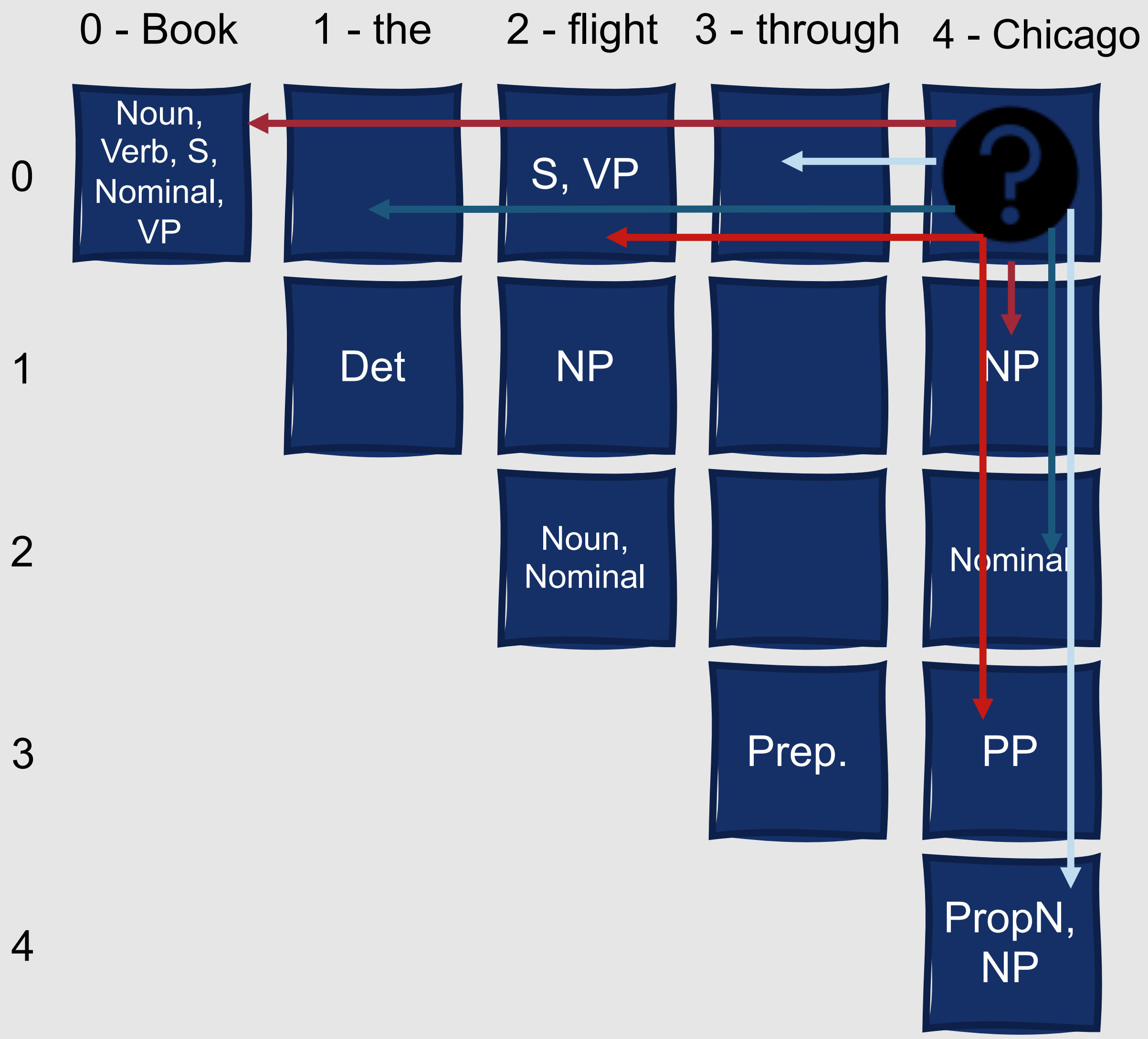
- The probability of sentence S having a parse tree T is the product of the individual probabilities associated with its constituent rules
  - $P(T,S) = \prod_{i=1}^{n} P(\beta_i|A_i)$
- To disambiguate between multiple valid parses, we find the parse tree T that results in the highest probability for the sentence S
  - $\hat{T}(S) = \underset{T\ s.t.\ S=\text{yield}(T)}{\text{argmax}} P(T)$

How do we compute the probability of a parse tree?

Simple solution: Extend the classic parsing algorithms we already have

Probabilistic CKY

- Still assume grammar is in Chomsky Normal Form
    - Right-hand side of production rule expands to two non-terminals or one terminal node
        - A → B C
        - A → *w*
- Still work with the upper triangular portion of a matrix



Probabilistic CKY

- Let *n* be the length of an input sentence, and *V* be the number of non-terminals in a grammar
- Consider the constituents *inside* the matrix cells to be part of a third dimension, of maximum length *V*
- Then, each cell $[i, j, A]$ in the $(n+1) \times (n+1) \times V$ matrix corresponds to the probability of constituent *A* spanning positions *i* through *j* of the input

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)				
1					
2					
3					
4					

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)				
1		N (0.01)			
2					
3					
4					

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)				
1		N (0.01)			
2			V (0.05)		
3				Det (0.40)	
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)			
2			V (0.05)		
3				Det (0.40)	
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)	☹️		
2			V (0.05)		
3				Det (0.40)	
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)	☹️		
2			V (0.05)	☹️	
3				Det (0.40)	
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)	☹		
2			V (0.05)	☹	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

> The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)	☹️		
2			V (0.05)	☹️	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)			
1		N (0.01)	☹️		
2			V (0.05)	☹️	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹		
1		N (0.01)	☹		
2			V (0.05)	☹	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	🙁		
1		N (0.01)	🙁		
2			V (0.05)	🙁	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹		
1		N (0.01)	☹		
2			V (0.05)	☹	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹️		
1		N (0.01)	☹️	☹️	
2			V (0.05)	☹️	
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹️		
1		N (0.01)	☹️	☹️	
2			V (0.05)	☹️	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.02 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹️		
1		N (0.01)	☹️	☹️	
2			V (0.05)	☹️	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹️		
1		N (0.01)	☹️	☹️	
2			V (0.05)	☹️	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹️		
1		N (0.01)	☹️	☹️	
2			V (0.05)	☹️	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹️		
1		N (0.01)	☹️	☹️	
2			V (0.05)	☹️	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹		
1		N (0.01)	☹	☹	
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹️	☹️	
1		N (0.01)	☹️	☹️	
2			V (0.05)	☹️	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	
1		N (0.01)	☹	☹	
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	🙁	🙁	
1		N (0.01)	🙁	🙁	
2			V (0.05)	🙁	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

> The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹️	☹️	
1		N (0.01)	☹️	☹️	
2			V (0.05)	☹️	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	
1		N (0.01)	☹	☹	☹
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	
1		N (0.01)	☹	☹	☹
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	S (0.8 * 0.0012 * 0.000024 = $2.304*10^{-8}$)
1		N (0.01)	☹	☹	☹
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	🙁	🙁	S (0.8 * 0.0012 * 0.000024 = $2.304*10^{-8}$)
1		N (0.01)	🙁	🙁	🙁
2			V (0.05)	🙁	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	S (0.8 * 0.0012 * 0.000024 = 2.304*10^{-8})
1		N (0.01)	☹	☹	☹
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Case Example: Probabilistic CKY

The price includes a computer

Production Rule	Probability
S → NP VP	0.80
NP → Det N	0.30
VP → V NP	0.20
V → includes	0.05
Det → the	0.40
Det → a	0.40
N → price	0.01
N → computer	0.02

	0 - The	1 - price	2 - includes	3 - a	4 - computer
0	Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0012)	☹	☹	S (0.8 * 0.0012 * 0.000024 = $2.304*10^{-8}$)
1		N (0.01)	☹	☹	☹
2			V (0.05)	☹	VP (0.2 * 0.05 * 0.0024 = 0.000024)
3				Det (0.40)	NP (0.3 * 0.4 * 0.01 = 0.0024)
4					N (0.02)

Where did these probabilities come from?

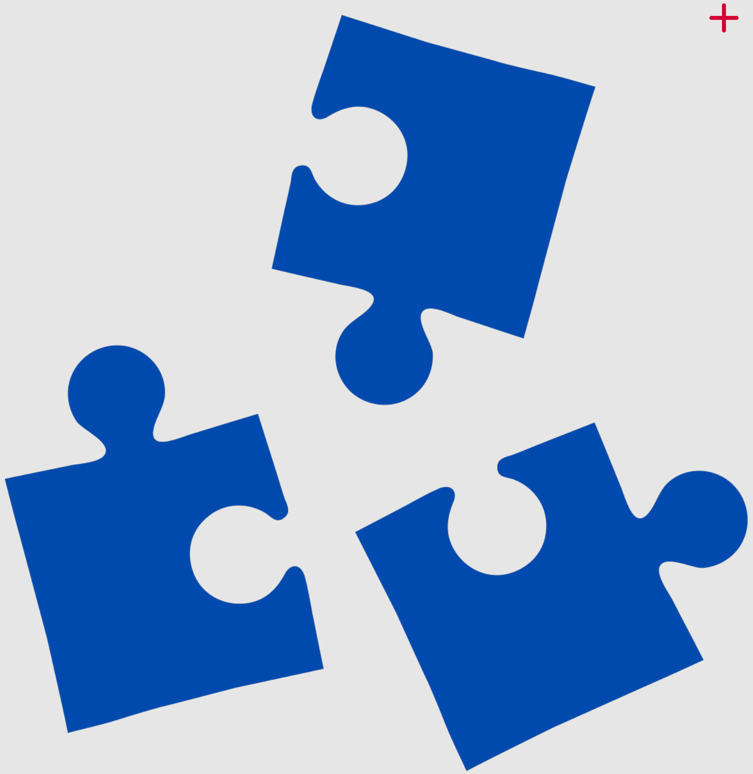
- Often, a corpus
  - $P(\alpha \rightarrow \beta|\alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_{\gamma} Count(\alpha \rightarrow \gamma)} = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$
- Or, if we don't have a labeled corpus, we can apply a generalization of the forward-backward algorithm called the **inside-out algorithm**
  - Start with equal probabilities for each rule
  - Parse the input
  - Compute a probability for each parse
  - Weight the counts based on these probabilities
  - Re-estimate the probabilities accordingly
  - Repeat until convergence

Challenges Associated with PCFGs

- PCFGs solve many issues associated with resolving ambiguities, but they're not perfect!
- A few problems associated with PCFGs:
  - **Poor independence assumptions**, which may make it difficult to model important **structural dependencies** in the parse tree
  - **Lack of lexical conditioning**, which may allow **lexical dependency issues** (e.g., those dealing with preposition attachment or other syntactic properties) to arise

How can we address these issues?

- More sophisticated techniques are needed, such as:
  - Adding extra constraints to rules by splitting them based on their parents or their syntactic positions
  - Using slightly different grammatical paradigms, such as **probabilistic lexicalized CFGs**



Parsing Methods

Probabilistic CKY

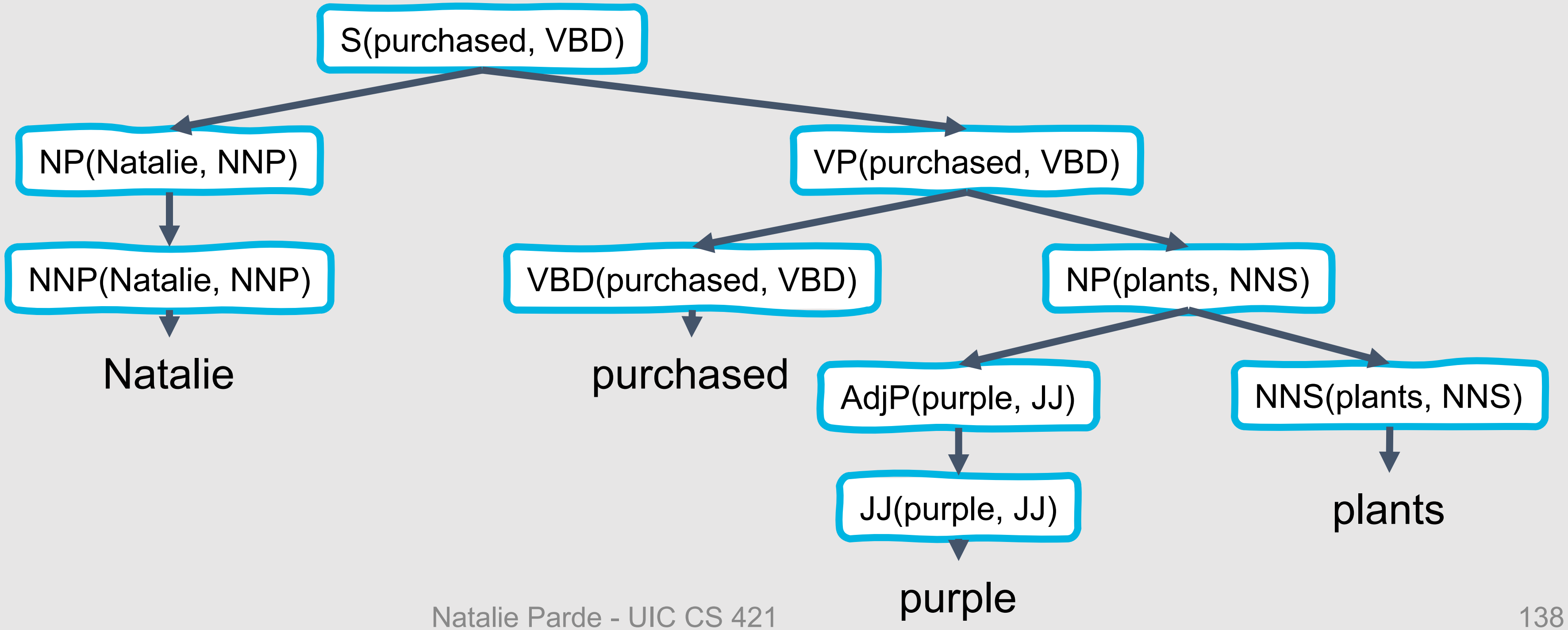
- Especially when employing automated splits and merges!

Lexicalized Parsers

- Collins Parser
- Charniak Parser

Lexicalized Parsers

- Allow lexicalized rules
  - Non-terminals specify lexical heads and associated POS tags
  - NP(plants, NNS) → AdjP(purple, JJ) NNS(plants, NNS)

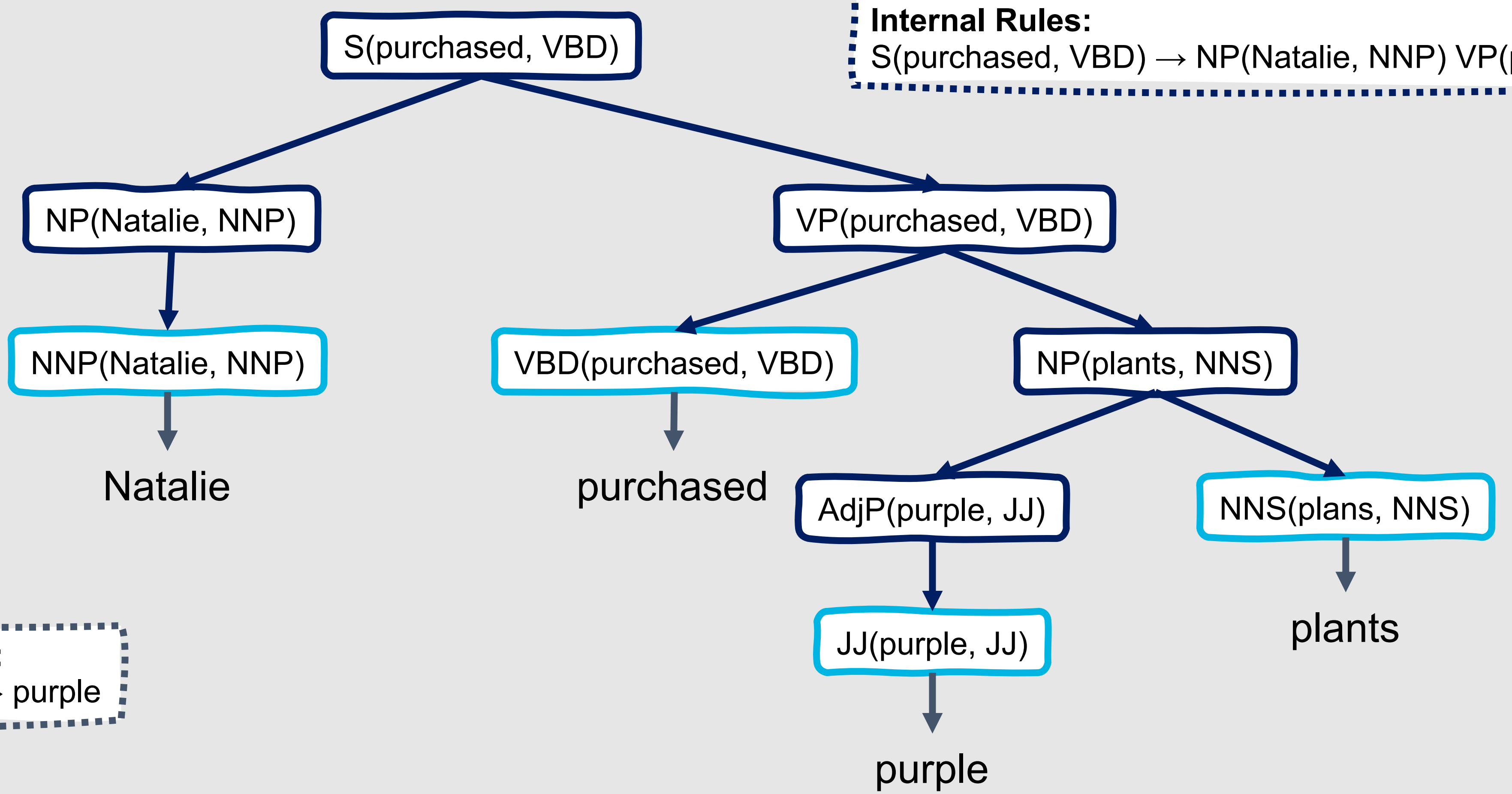


Lexicalized Grammars

- Intuitively, much like having many copies of the same production rule
  - NP(plants, NNS) → AdjP(purple, JJ) NNS(plants, NNS)
  - NP(plants, NNS) → AdjP(green, JJ) NNS(plants, NNS)
  - NP(computers, NNS) → AdjP(purple, JJ) NNS(computers, NNS)
- Two types of rules:
  - **Lexical Rules:** Generate a terminal word
  - **Internal Rules:** Generate a non-terminal constituent

Lexical vs. Internal Rules

Internal Rules:
S(purchased, VBD) → NP(Natalie, NNP) VP(purchased, VBD)



External Rules:
JJ(purple, JJ) → purple

Lexical vs. Internal Rules

- **Lexical Rules**
  - Deterministic
    - JJ(purple, JJ) → purple
- **Internal Rules**
  - Require estimated probabilities
    - Normal maximum likelihood estimation won't work well because the counts will be too sparse
    - Instead, estimate the probability of an internal rule based on the product of the smaller, more reliable probability estimates comprising it

The Collins Parser

- Consider the following generic production rule:
  - $LHS \rightarrow L_n \ L_{n-1} \ldots L_1 \ H \ R_1 \ldots R_{n-1} \ R_n$

Non-terminal LHS
NP(plants, NNS)

Non-terminal left of head
AdjP(purple, JJ)

Head
NNS(plants, NNS)

Non-terminal right of head
PP(under, P)

The Collins Parser

- Consider the following generic production rule:
  - $LHS \rightarrow L_n \; L_{n-1} \ldots L_1 \; H \; R_1 \ldots R_{n-1} \; R_n$

Non-terminal LHS
NP(plants, NNS)

Non-terminal left of head
AdjP(purple, JJ)

Head
NNS(plants, NNS)

Non-terminal right of head
PP(under, P)

P_L: Probability for generating dependents on the left

P_H: Probability for generating heads

P_R: Probability for generating dependents on the right

The Collins Parser

- Goal: Use P_H, P_L, and P_R to estimate the overall probability for the production rule
- Method:
  - Surround the righthand side of the rule with STOP non-terminals
    - NP(plants, NNS) → STOP AdjP(purple, JJ) NNS(plants, NNS) PP(under, IN) STOP
  - Compute the individual P_H, P_L, and P_R values for the head and the non-terminals to its left and right (including STOP non-terminals)
  - Multiply these together

Grab the **purple plants under the bookcase**.

The Collins Parser

- Consider the following generic production rule:
  - $LHS \rightarrow L_n \; L_{n-1} \ldots L_1 \; H \; R_1 \ldots R_{n-1} \; R_n$

Non-terminal LHS
NP(plants, NNS)

Non-terminal left of head
AdjP(purple, JJ)

Head
NNS(plants, NNS)

Non-terminal right of head
PP(under, IN)

P_L: Probability for generating dependents on the left

P_H: Probability for generating heads

P_R: Probability for generating dependents on the right

Grab the **purple plants under the bookcase**.

NP(plants, NNS) → STOP AdjP(purple, JJ) NNS(plants, NNS) PP(under, IN) STOP

The Collins Parser

- Consider the following generic production rule:
  - $LHS \rightarrow L_n \; L_{n-1} \ldots L_1 \; H \; R_1 \ldots R_{n-1} \; R_n$

Non-terminal LHS
NP(plants, NNS)

Non-terminal left of head
AdjP(purple, JJ)

Head
NNS(plants, NNS)

Non-terminal right of head
PP(under, IN)

P_L: Probability for generating dependents on the left

P_H: Probability for generating heads

P_R: Probability for generating dependents on the right

$P_H(H|LHS)$ = P(NNS(plants, NNS) | NP(plants, NNS))

Grab the **purple plants under the bookcase**.

NP(plants, NNS) → STOP AdjP(purple, JJ) **NNS(plants, NNS)** PP(under, IN) STOP

The Collins Parser

- Consider the following generic production rule:
  - $LHS \rightarrow L_n\ L_{n-1} \ldots L_1\ H\ R_1 \ldots R_{n-1}\ R_n$

Non-terminal LHS
NP(plants, NNS)

Non-terminal left of head
AdjP(purple, JJ)

Head
NNS(plants, NNS)

Non-terminal right of head
PP(under, IN)

P_L: Probability for generating dependents on the left

P_H: Probability for generating heads

P_R: Probability for generating dependents on the right

P_H(H|LHS) = P(NNS(plants, NNS) | NP(plants, NNS))

P_L(STOP|LHS H) = P(STOP | NP(plants, NNS) NNS(plants, NNS))

P_L(L_1|LHS H) = P(AdjP(purple, JJ) | NP(plants, NNS) NNS(plants, NNS))

Grab the **purple plants under the bookcase.**

NP(plants, NNS) → **STOP AdjP(purple, JJ)** NNS(plants, NNS) PP(under, IN) STOP

The Collins Parser

- Consider the following generic production rule:
  - $LHS \rightarrow L_n \; L_{n-1} \ldots L_1 \; H \; R_1 \ldots R_{n-1} \; R_n$

Non-terminal LHS
NP(plants, NNS)

Non-terminal left of head
AdjP(purple, JJ)

Head
NNS(plants, NNS)

Non-terminal right of head
PP(under, IN)

P_L: Probability for generating dependents on the left

P_H: Probability for generating heads

P_R: Probability for generating dependents on the right

Grab the **purple plants under the bookcase.**

NP(plants, NNS) → STOP AdjP(purple, JJ) NNS(plants, NNS) **PP(under, IN) STOP**

P_H(H|LHS) = P(NNS(plants, NNS) | NP(plants, NNS))

P_L(STOP|LHS H) = P(STOP | NP(plants, NNS) NNS(plants, NNS))

P_L(L_1|LHS H) = P(AdjP(purple, JJ) | NP(plants, NNS) NNS(plants, NNS))

P_R(R_1|LHS H) = P(PP(under, IN) | NP(plants, NNS) NNS(plants, NNS))

P_R(STOP|LHS H) = P(STOP | NP(plants, NNS) NNS(plants, NNS))

The Collins Parser

- Consider the following generic production rule:
  - $LHS \rightarrow L_n\ L_{n-1} \dots L_1\ H\ R_1 \dots R_{n-1}\ R_n$

Non-terminal LHS
NP(plants, NNS)

Non-terminal left of head
AdjP(purple, JJ)

Head
NNS(plants, NNS)

Non-terminal right of head
PP(under, IN)

P_L: Probability for generating dependents on the left

P_H: Probability for generating heads

P_R: Probability for generating dependents on the right

Grab the **purple plants under the bookcase.**

NP(facemasks, NNS) → STOP AdjP(purple, JJ) NNS(plants, NNS) **PP(under, IN) STOP**

= P_H(H|LHS) * P_L(STOP|LHS H) * P_L(L_1|LHS H) * P_R(R_1|LHS H) * P_R(STOP|LHS H)

P_H(H|LHS) = P(NNS(plants, NNS) | NP(plants, NNS))

P_L(STOP|LHS H) = P(STOP | NP(plants, NNS) NNS(plants, NNS))

P_L(L_1|LHS H) = P(AdjP(purple, JJ) | NP(plants, NNS) NNS(plants, NNS))

P_R(R_1|LHS H) = P(PP(under, IN) | NP(plants, NNS) NNS(plants, NNS))

P_R(STOP|LHS H) = P(STOP | NP(plants, NNS) NNS(plants, NNS))

Then, it's relatively easy to estimate the individual probabilities.

- Maximum likelihood estimate
- Much less subject to sparsity problems!

$P_R(R_1|\text{LHS H}) = P(\text{PP(under, IN)} \mid \text{NP(plants, NNS) NNS(plants, NNS)})$

$$\frac{\text{Count}(\text{NP}(\text{plants}, \text{NNS})\ \textit{with}\ \text{PP}(\text{under}, \text{IN})\ \textit{as a child to the right})}{\text{Count}(\text{NP}(\text{plants}, \text{NNS}))}$$

Combinatory Categorial Grammars (CCGs)

- *Heavily* lexicalized approach that groups words into categories and defines ways that those categories may be combined
- Three major parts:
  - Categories
  - Lexicon
  - Rules

CCG Categories

- **Atomic elements**
  - $\mathcal{A} \subseteq \mathcal{C}$, where $\mathcal{A}$ is a set of atomic elements, and $\mathcal{C}$ is the set of categories for the grammar
  - Sentences and noun phrases
- **Single-argument functions**
  - (X/Y), (X\Y) $\in \mathcal{C}$, if $\mathrm{X}, \mathrm{Y} \in \mathcal{C}$
    - (X/Y): Seeks a constituent of type Y to the right, and returns X
    - (X\Y): Seeks a constituent of type Y to the left, and returns X
  - Verb phrases, more complex noun phrases, etc.

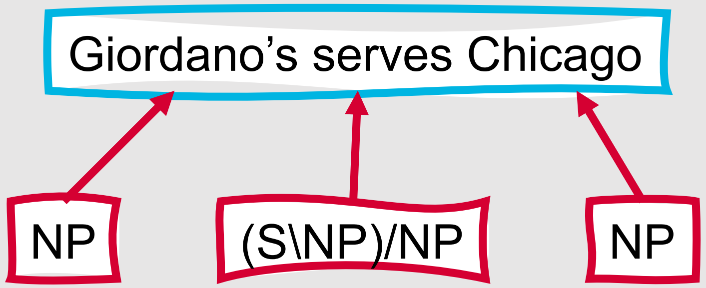
CCG Lexicon

- Assigns CCG categories to words
  - Chicago: NP
    - Atomic category
  - cancel: (S\NP)/NP
    - Functional category
    - Seeks an NP to the right, returning (S\NP), which seeks an NP to the left, returning S

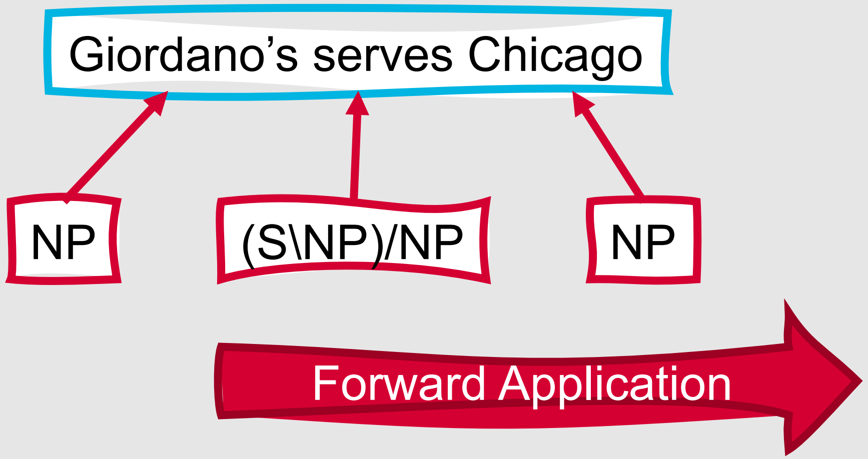
CCG Rules

- Specify how functions and their arguments may be combined
- **Forward function application:** Applies the function to its argument on the right, resulting in the specified category
  - X/Y Y ⇒ X
- **Backward function application:** Applies the function to its argument on the left, resulting in the specified category
  - Y X\Y ⇒ X
- A coordination rule can also be applied
  - X CONJ X ⇒ X

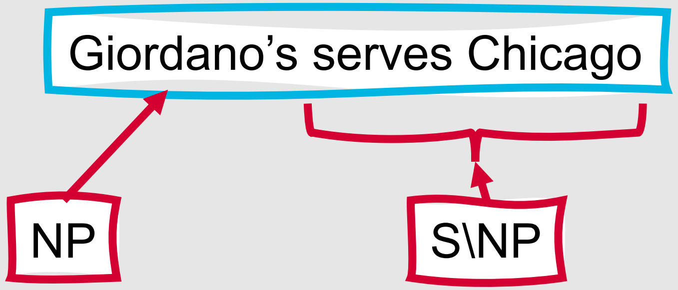
CCG Rules: Example



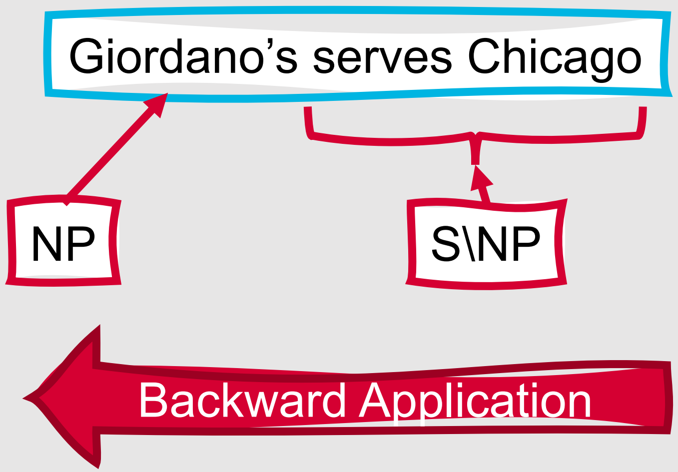
CCG Rules: Example



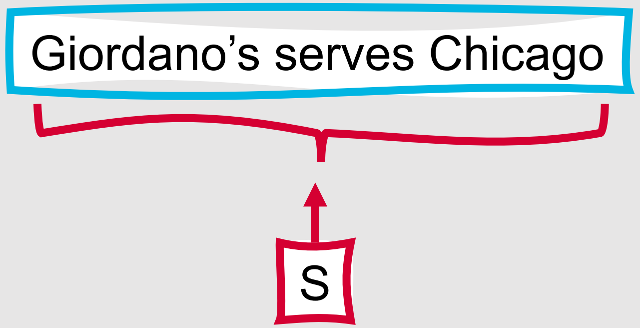
CCG Rules: Example



CCG Rules: Example



CCG Rules: Example



CCG Operations

- **Forward composition**
  - Can be applied when, given two functions, the first seeks a constituent of type Y to the right and the second provides a constituent of type Y as its result
    - $X/Y \; Y/Z \Rightarrow X/Z$
- **Backward composition**
  - Can be applied when, given two functions, the first seeks a constituent of type Y to the left and the second provides a constituent of type Y as its result
    - $Y\backslash Z \; X\backslash Y \Rightarrow X\backslash Z$

CCG Operations

- **Type raising**
  - Converts atomic categories to functional categories, or simple functional categories to more complex functional categories
    - $X \Rightarrow T/(T\backslash X)$, where T can be any existing atomic or functional category
    - $X \Rightarrow T\backslash(T/X)$
  - Facilitates the creation of intermediate elements that do not directly map to traditional constituents in the language
- Type raising and function composition can be employed together to parse **long-range dependencies**

CCGBank

- Largest and most popular CCG treebank
- Based on the Penn Treebank
- 44,000-word lexicon with 1200+ categories
- More details: https://catalog.ldc.upenn.edu/LDC2005T13

Ambiguity in CCGs

CCG lexicons allow words to be associated with numerous categories, depending on how they interact with other words in the sentence

This can create ambiguity when parsing!

CCG Parsing Frameworks

- Probabilistic CKY
  - Okay, but needs to be adapted a bit due to the large number of categories available for each word (otherwise, lots of unnecessary constituents would be added to the table)
  - The solution: **Supertagging**
- Supertags are also used in other CCG parsing frameworks

Supertagging

- Trained using CCG treebanks (e.g., CCGBank)
- Predict allowable category assignments (supertags) for each word in a lexicon, given an input context
- Commonly framed as a supervised sequence labeling problem

After extracting supertags, probabilistic CKY can be employed as a CCG parser.

- Another popular CCG parsing technique: **A* Algorithm**
- **A*:** Heuristic search algorithm that finds the lowest-cost path to an end state, by exploring the lowest-cost partial solution at each iteration until a full solution is identified
- Search states = edges representing completed constituents
- Cost is based on the probability of the CCG derivation
- A* results in fewer unnecessary constituents being explored than probabilistic CKY

Evaluating Parsers

- **PARSEVAL measures:** Seek to determine how close a predicted parse is to a gold standard parse for the same text, based on its individual constituents
  - Constituent is correct if it matches a constituent in the gold standard in terms of its:
    - Starting point
    - Ending point
    - Non-terminal symbol

Once constituent correctness is defined….

- We can apply the same metrics we use for other NLP problems!
  - $\text{Recall} = \frac{\#\text{ correct constituents in predicted parse}}{\#\text{ constituents in gold standard parse}}$
  - $\text{Precision} = \frac{\#\text{ correct constituents in predicted parse}}{\#\text{ constituents in predicted parse}}$
- It is also common to count the number of **cross-brackets**, or constituents for which the gold standard parse is formatted as ((A B) C) while the predicted parse is formatted as (A (B C))

Summary: Statistical Constituency Parsing

We can select the best parse for a sentence using **probabilistic context-free grammars**

The **CKY algorithm** can be updated to incorporate these probabilities for use with PCFG parsing

An alternative parsing paradigm uses **lexicalized grammar frameworks**

We can evaluate parsers using standard NLP metrics applied based on the number of **correctly identified constituents** in a predicted parse