# Statistical Parsing

## JURAFSKY AND MARTIN CHAPTER 12

*Based on slides from Sameer Singh, Dan Jurafsky, Noah Smith, Slav Petrov, and everyone else they copied from.*

# Outline

Parsing: CKY Algorithm

**Extensions: Probabilistic and Lexicalized**

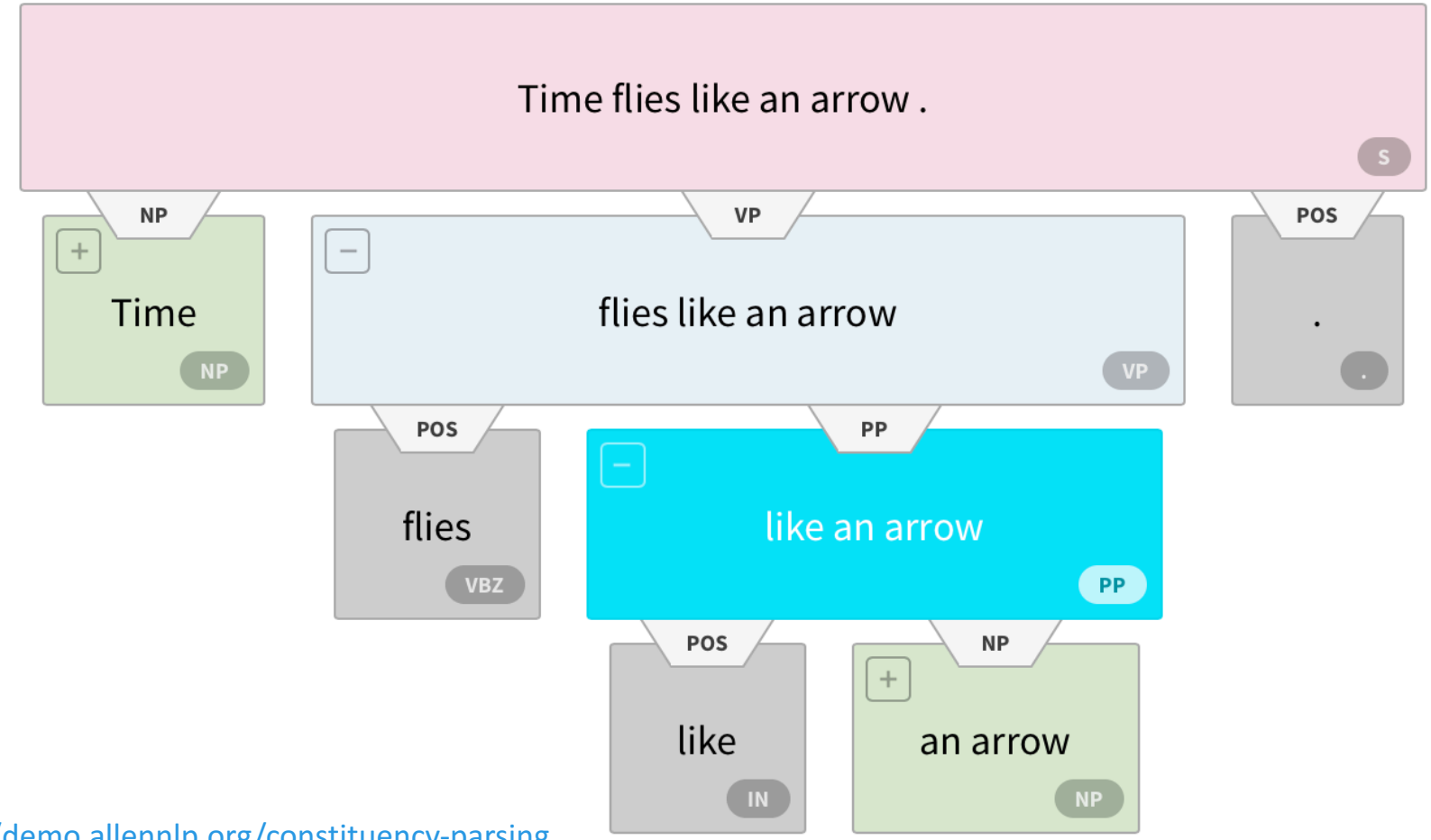
Dependency Parsing

# Ambiguity: Which parse?

Time flies like an arrow.

# Ambiguity: Which parse?

Time flies like an arrow.



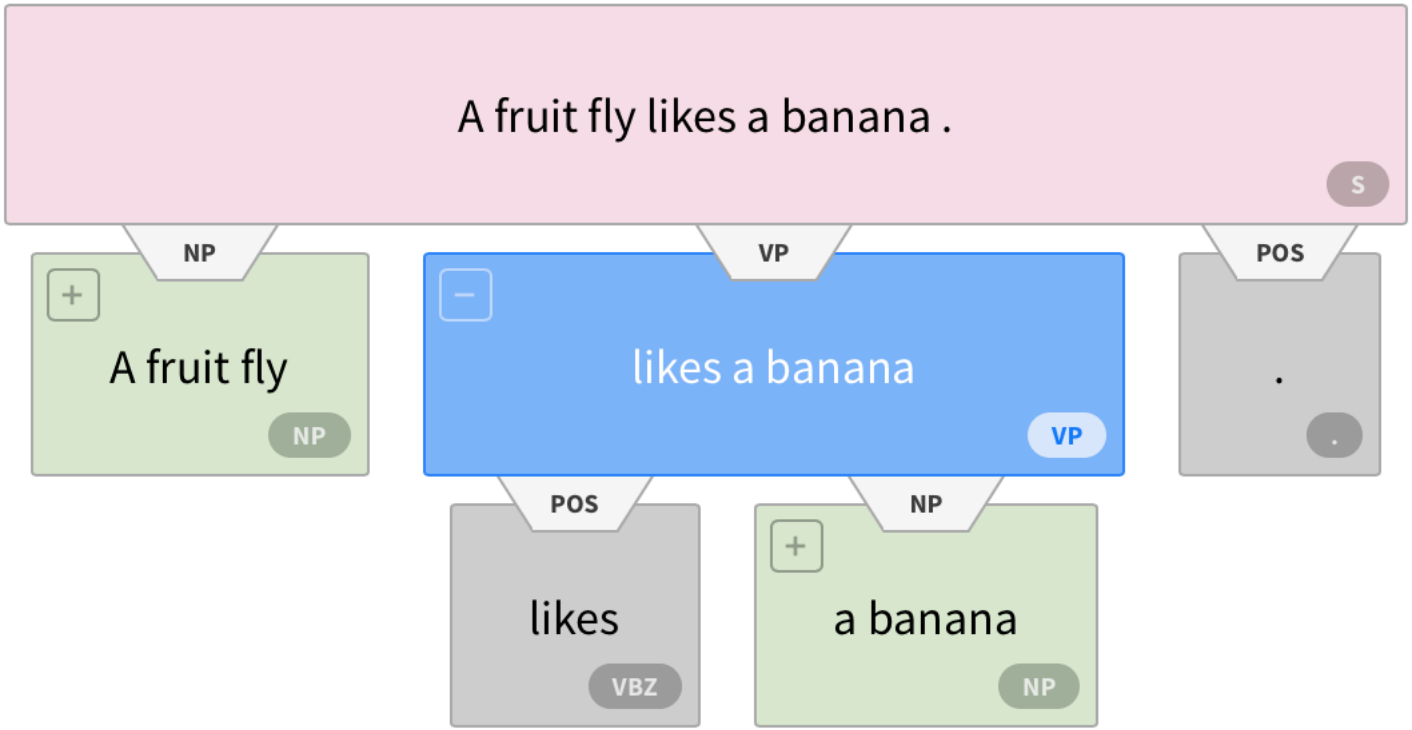

https://demo.allennlp.org/constituency-parsing

# Ambiguity: Which parse?

Fruit flies like a banana.

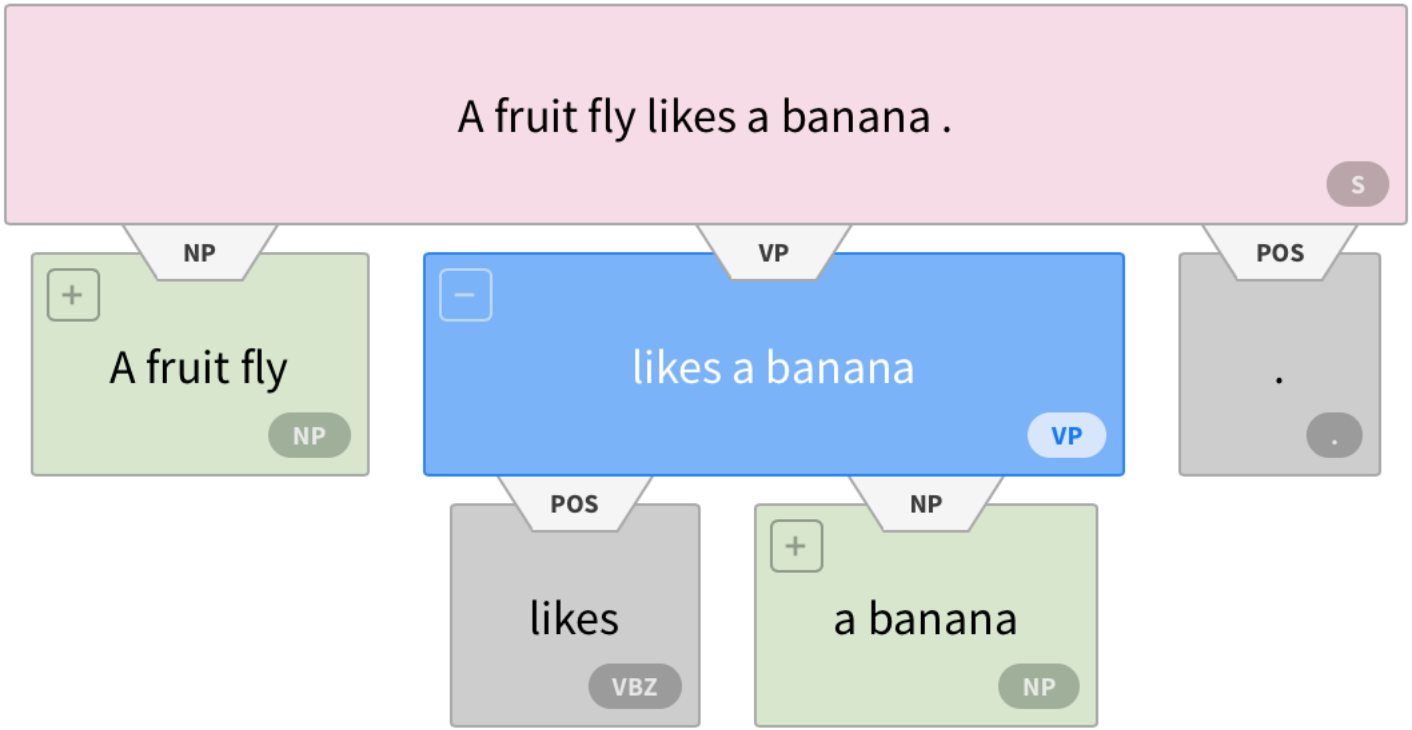
# Ambiguity: Which parse?

Fruit flies like a banana.



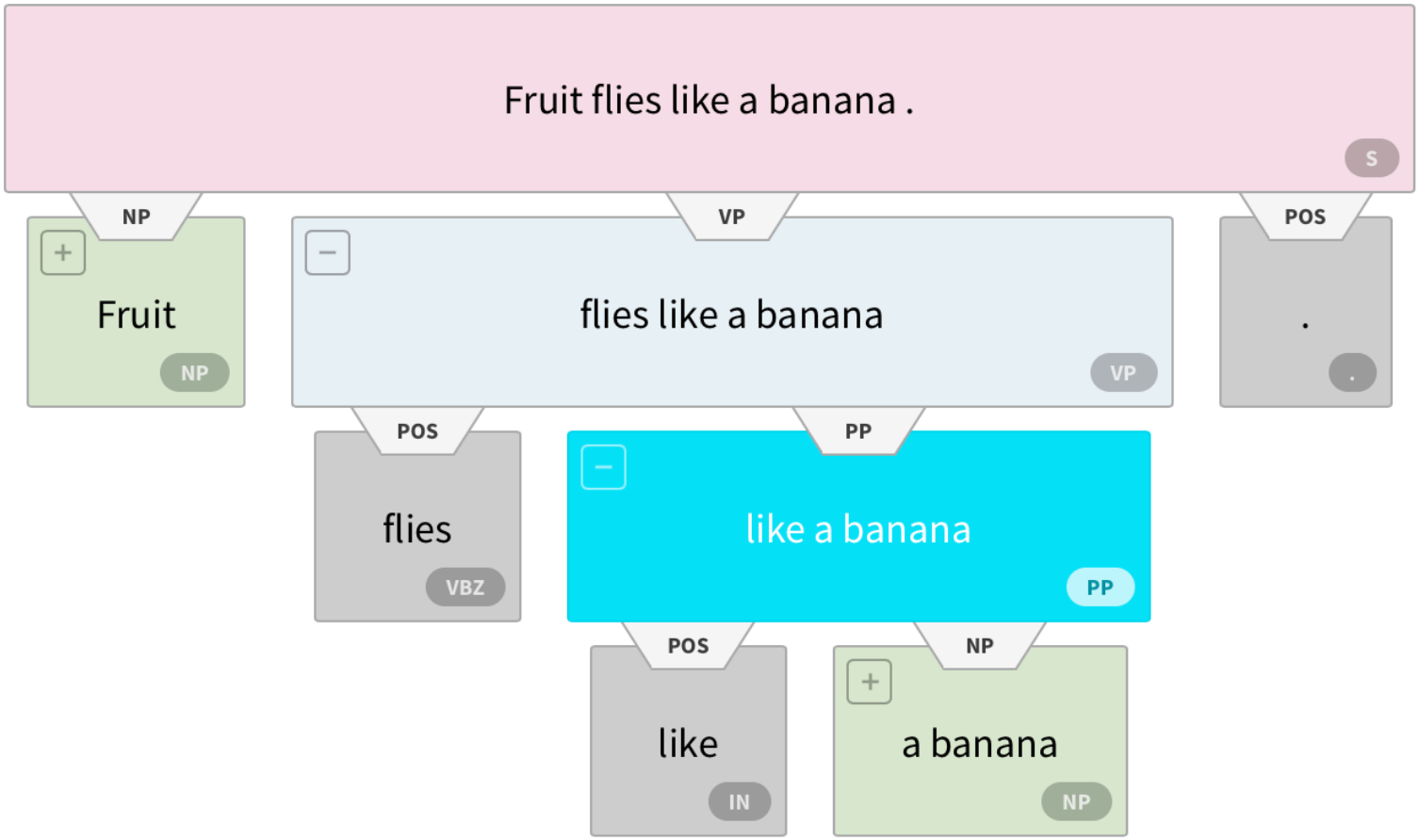
# Ambiguity: Which parse?

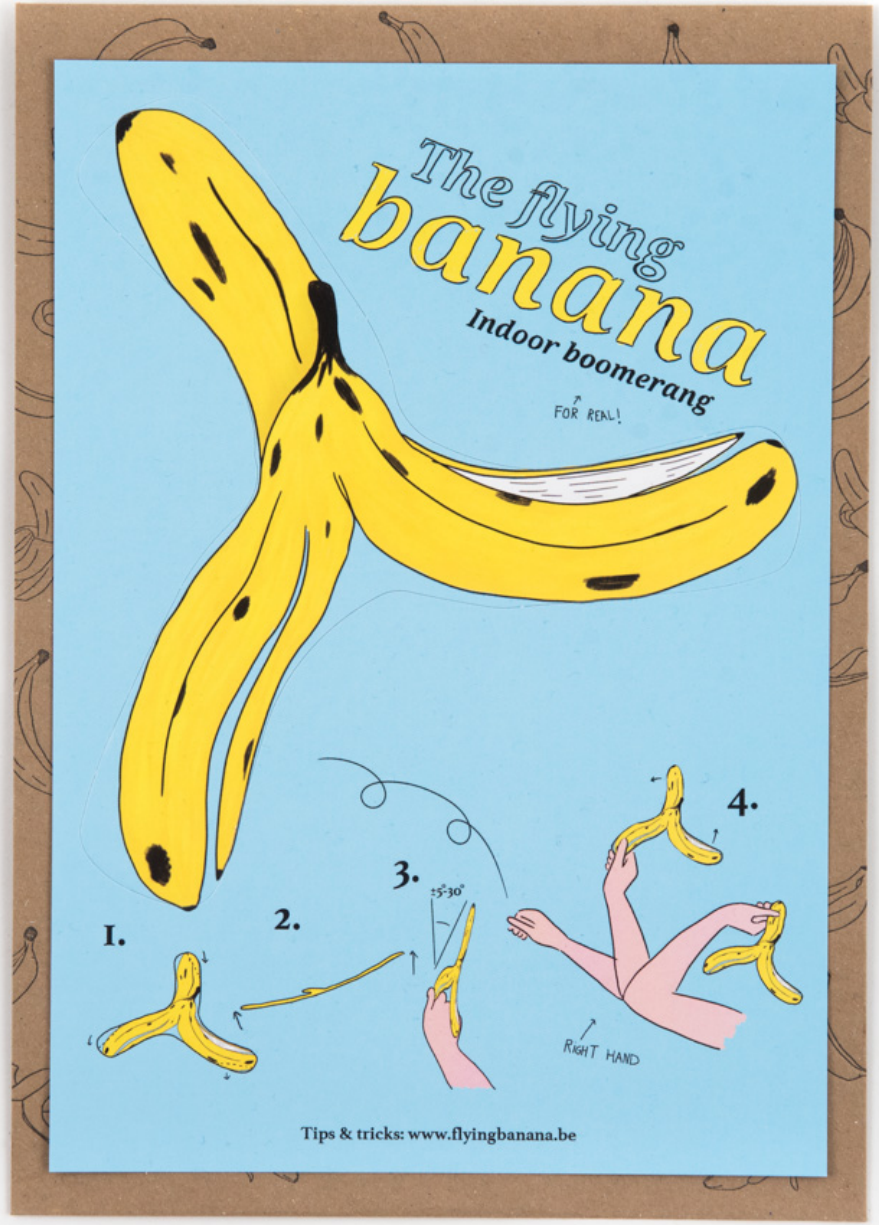
Fruit flies like a banana.



# Ambiguity: Which parse?

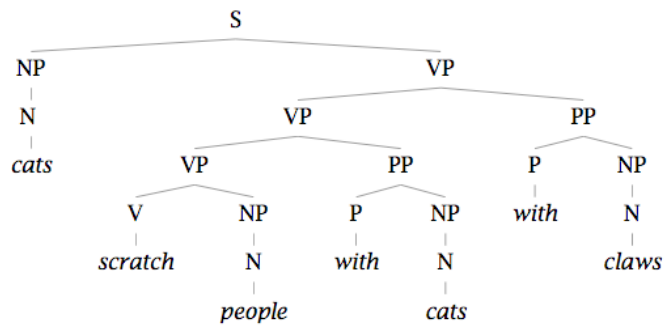
Fruit flies like a banana.

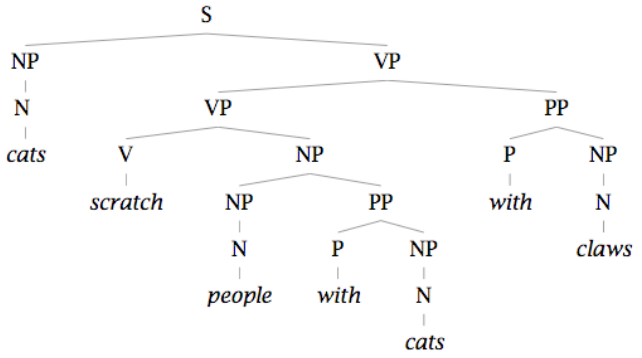


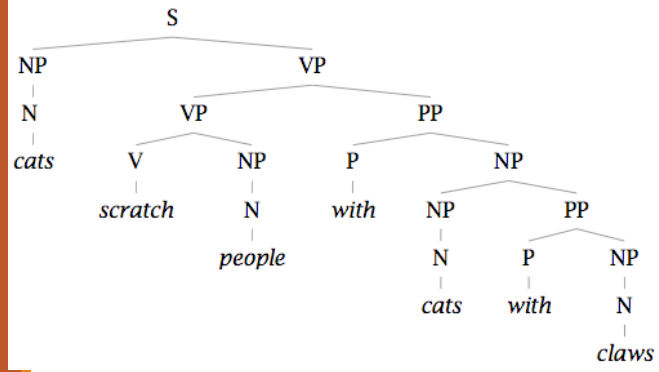

The flying banana
Indoor boomerang
FOR REAL!
1.
2.
3.
75°-30°
4.
RIGHT HAND
Tips & tricks: www.flyingbanana.be

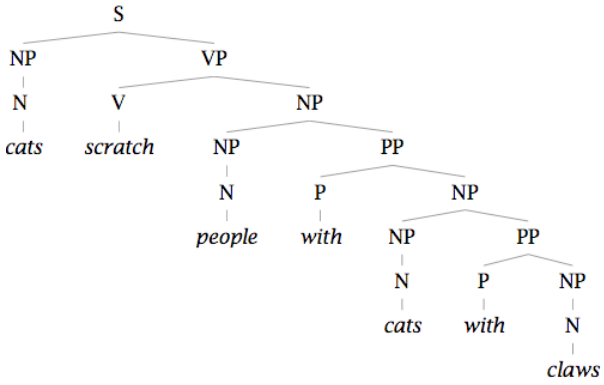
# Finding the *Best* Parse Tree

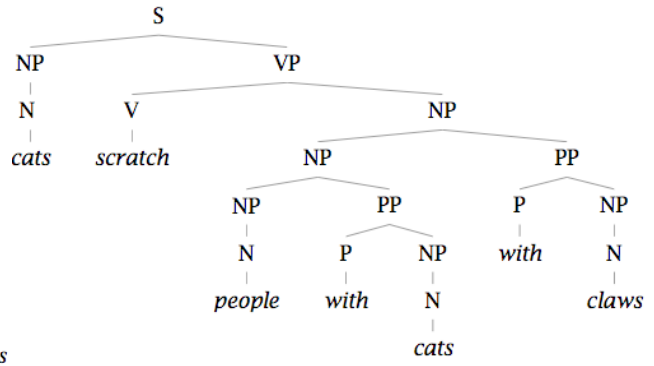
Cats scratch people with cats with claws.



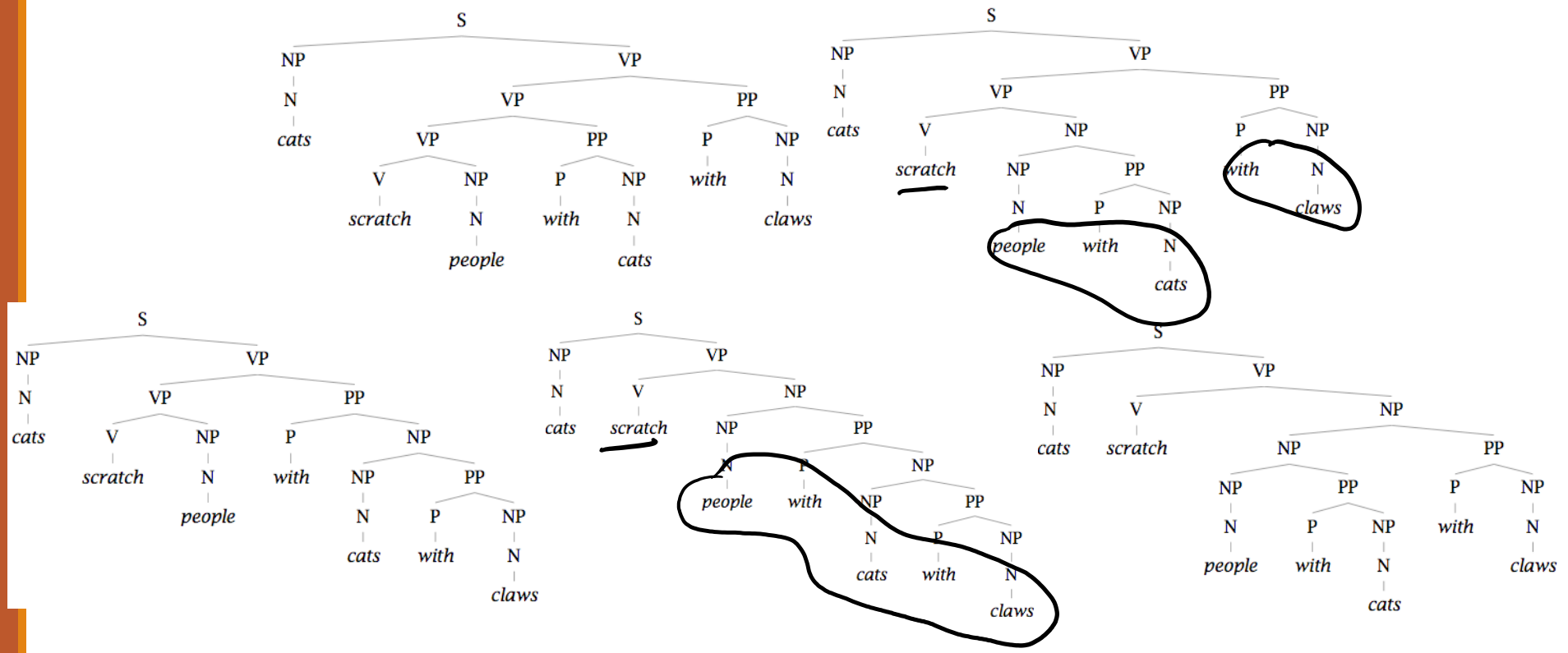



# Finding the *Best* Parse Tree

Cats scratch people with cats with claws.



# Probabilistic CFGs

Same as a regular context-free grammar:

- Terminal, non-terminals, and rules
- Additionally, attach a probability to each rule!

Rule: $A \rightarrow B\ C$

Probability: P($A \rightarrow B\ C \mid A$)

Compute the probability of a parse tree:

# Probabilistic CFGs

Same as a regular context-free grammar:

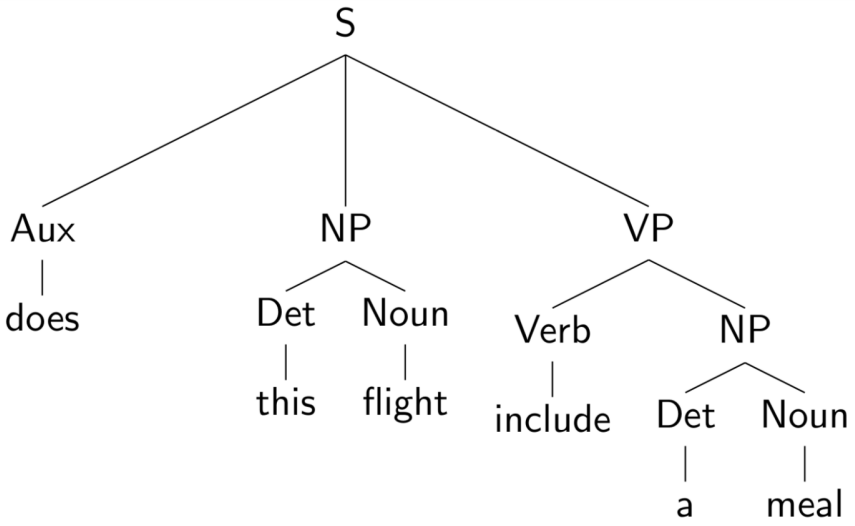
- Terminal, non-terminals, and rules
- Additionally, attach a probability to each rule!

Rule: $A \rightarrow B\ C$

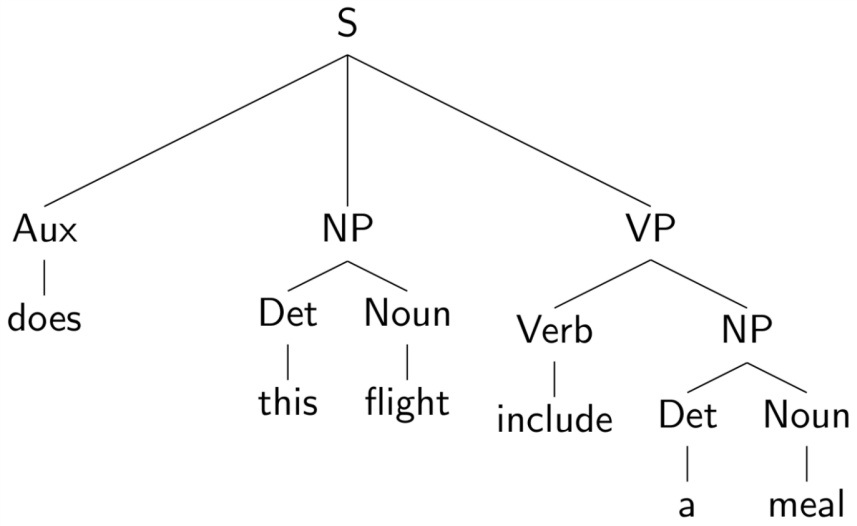
Probability: P($A \rightarrow B\ C \mid A$)

Compute the probability of a parse tree: $\prod_{A \rightarrow BC \in T} P(A \rightarrow BC \mid A)$

# Example of a PCFG



# Example of a PCFG




$P(\text{Aux NP VP} \mid S)$

$P(\text{does} \mid \text{AUX})$

$P(\text{Det Noun} \mid \text{NP})$

$P(\text{this} \mid \text{DET})\ P(\text{flight} \mid \text{Noun})$

$\vdots$

# Estimating the probabilities

# Estimating the probabilities

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\#\, \alpha \rightarrow \beta}{\#\, \alpha}$$

# The Parsing Problem

Given sentence **x** and grammar **G**,

Recognition

Is sentence **x** in the grammar? If so, prove it.

"Proof" is a deduction, valid parse tree.

Parsing

Show one or more derivations for **x** in **G**.

$$\underset{t \in \mathcal{T}_x}{\operatorname{argmax}}\, p(t \mid x)$$

Even with small grammars, grows exponentially!

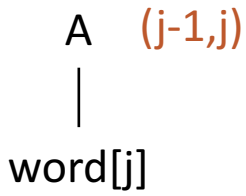
# Probabilistic CKY Algorithm

T[i,j,A] = Probability of the best parse with root A for the span (i,j)

**Base case**

Rule: P( $A \rightarrow word[j]$ )

T[j-1,j,A] = P(word[j] | A)

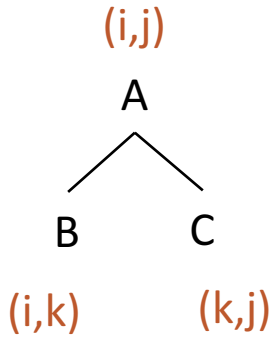



**Recursion**

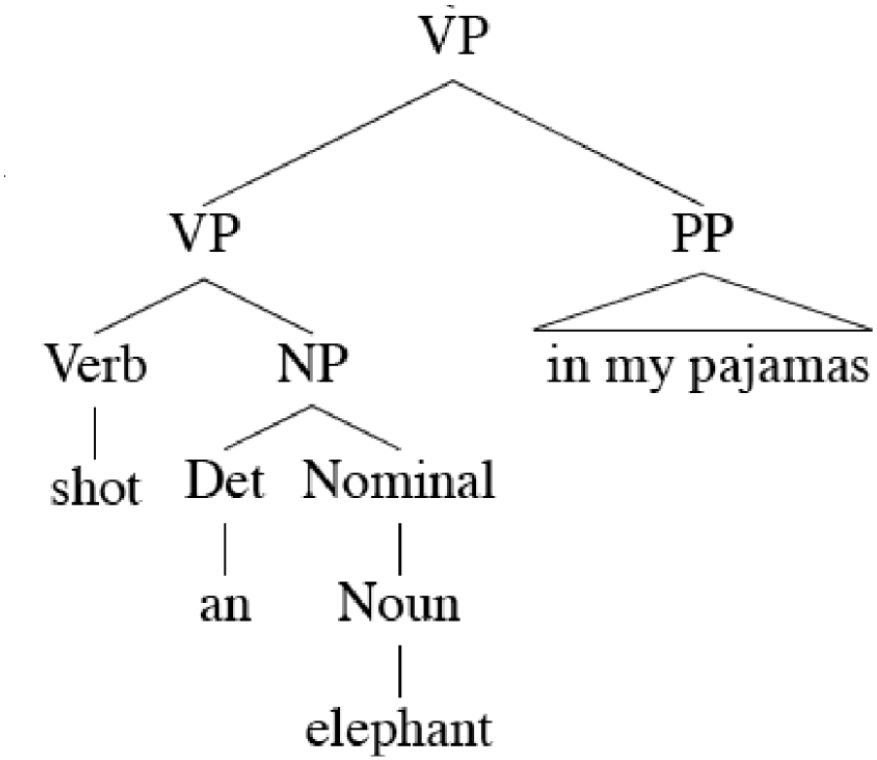
Rule: P( $A \rightarrow B\ C$ )

Try every position k, and every non-terminal pair:

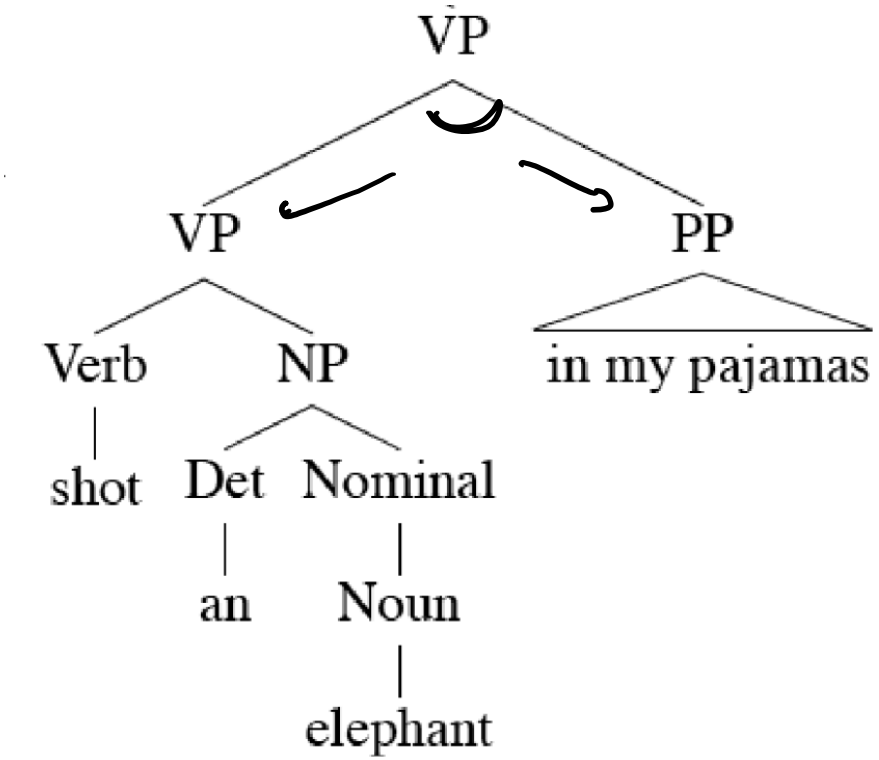
$$T[i,j,A] = \max_{k} P(B\ C \mid A)\ T[i,k,B]\ T[k,j,C]$$

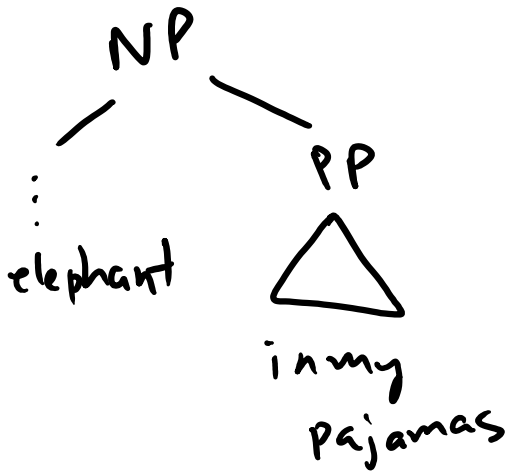


# Lexicalized PCFGs

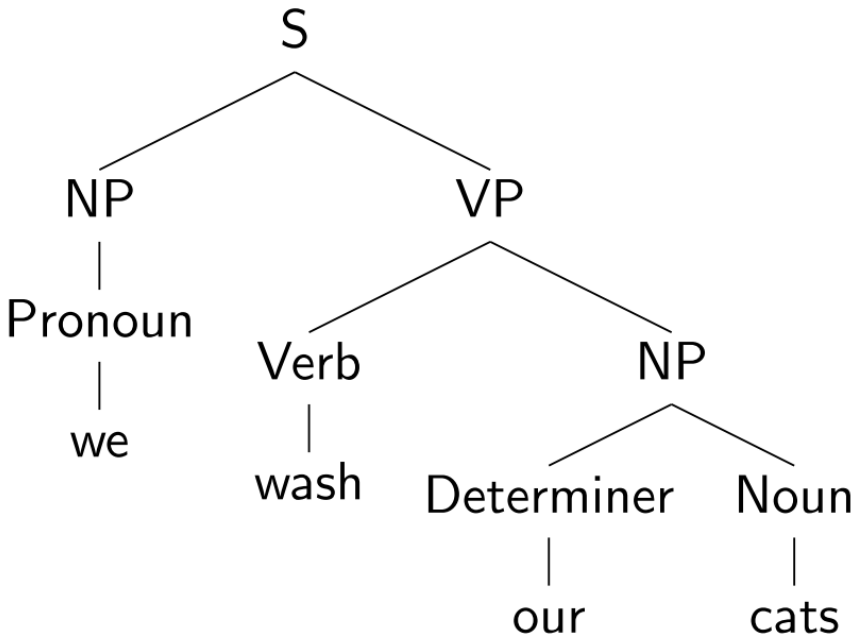


# Lexicalized PCFGs

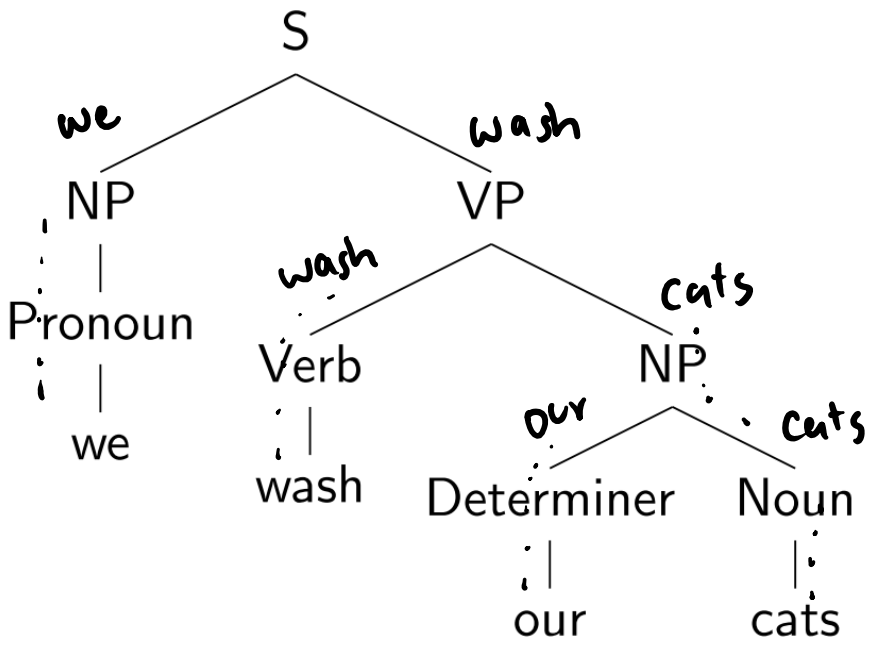


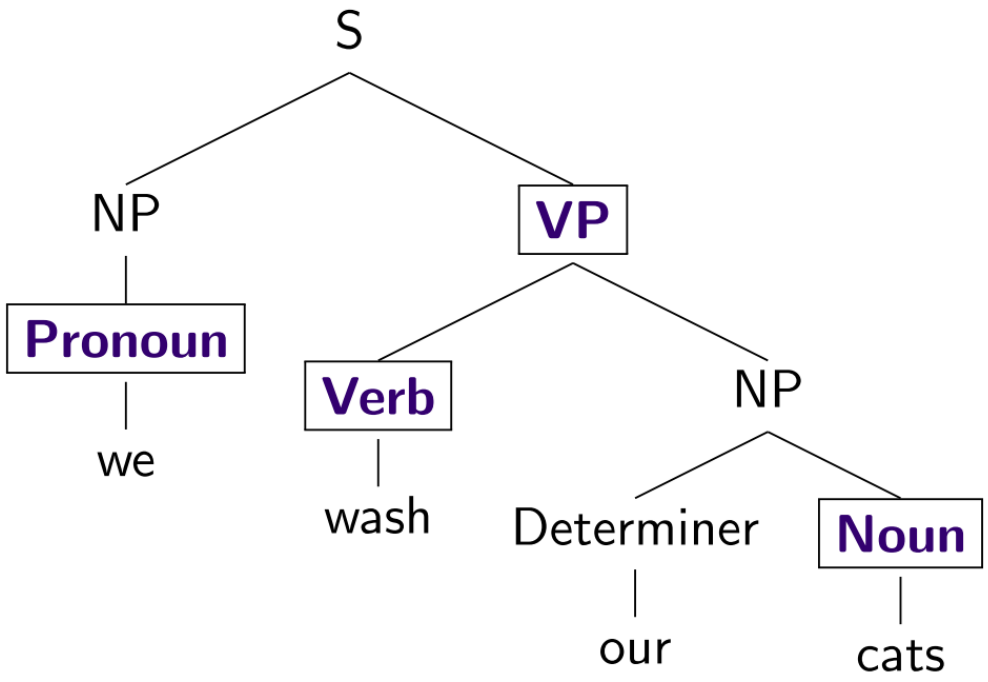
# Lexicalizing a CFG



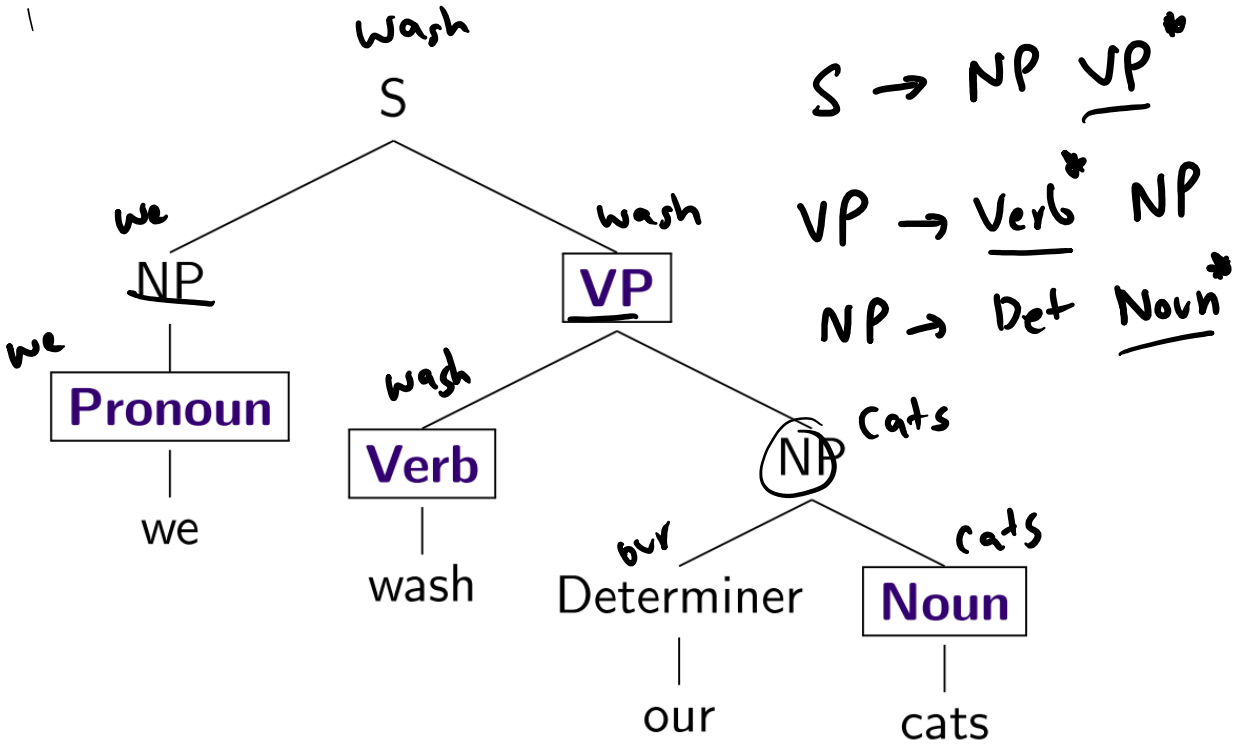
# Lexicalizing a CFG



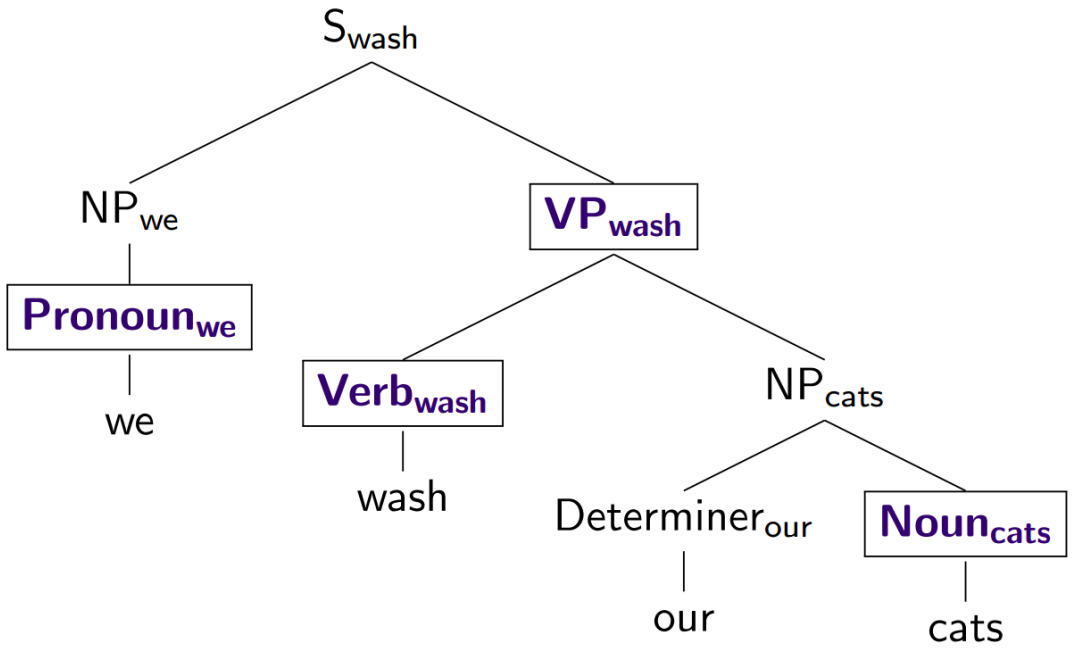
# Lexicalizing a CFG



# Lexicalizing a CFG



# Lexicalizing a CFG



# Outline

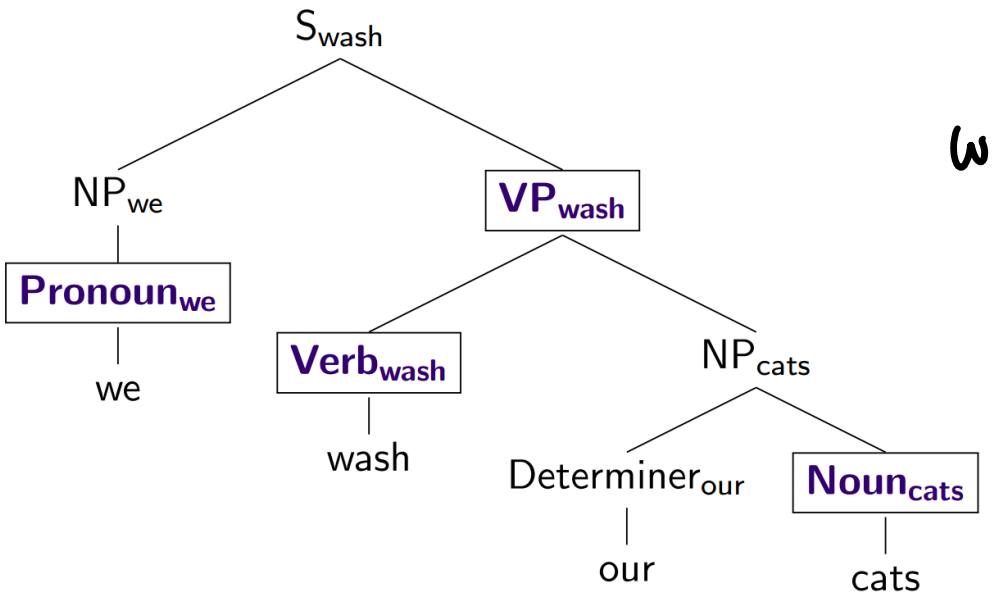
Parsing: CKY Algorithm

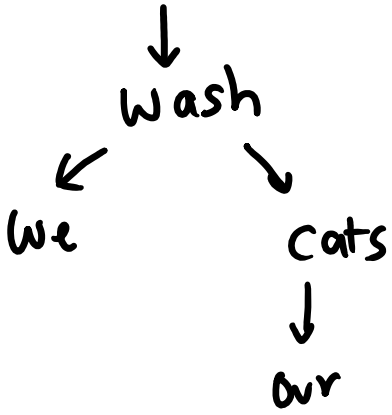
Extensions: Probabilistic and Lexicalized

Dependency Parsing

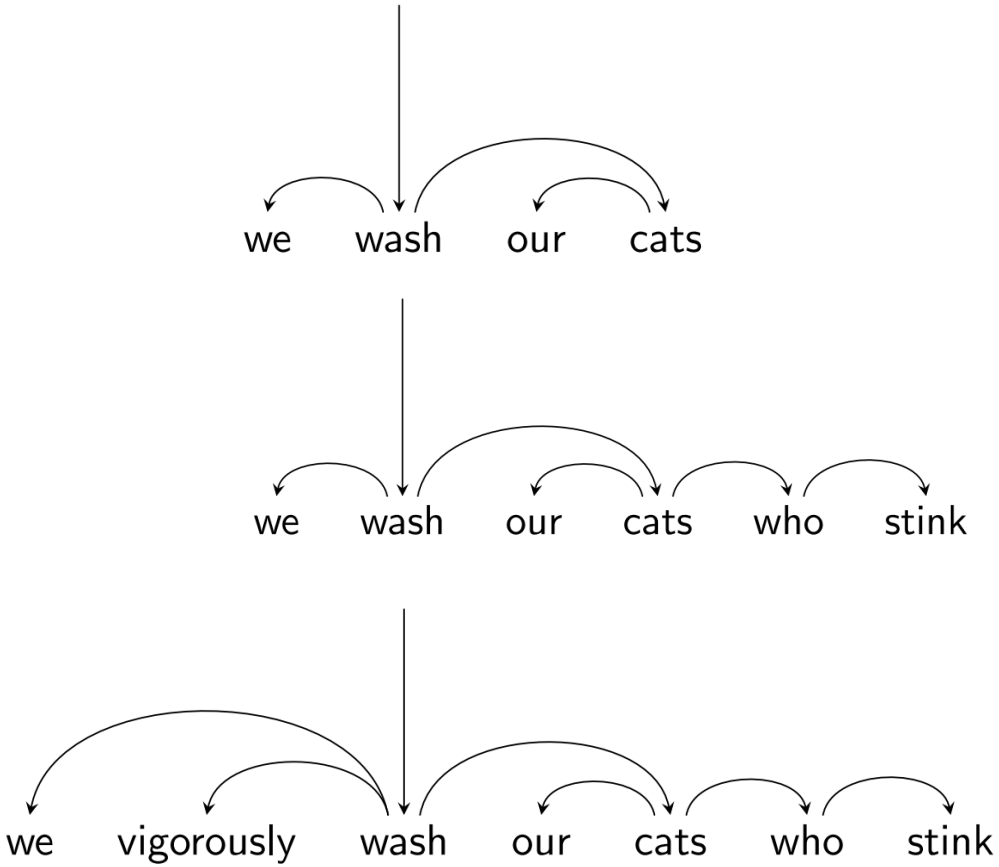
# Dependencies

Represent only the syntactic dependencies…

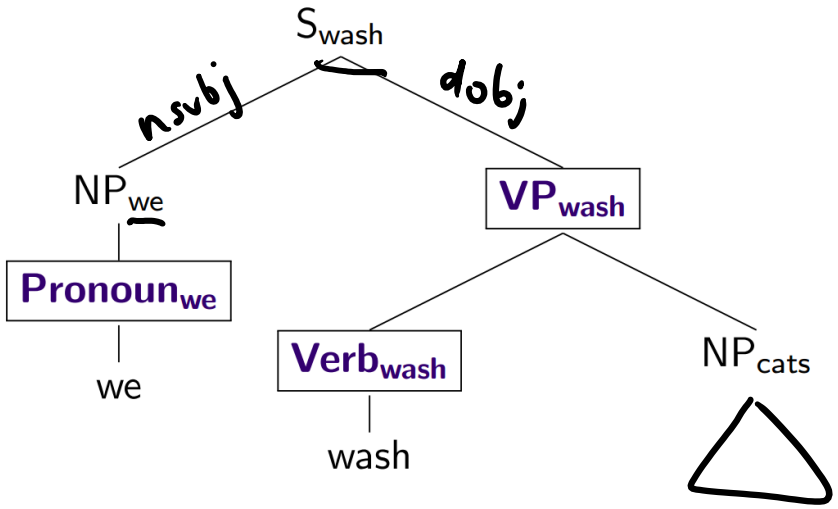


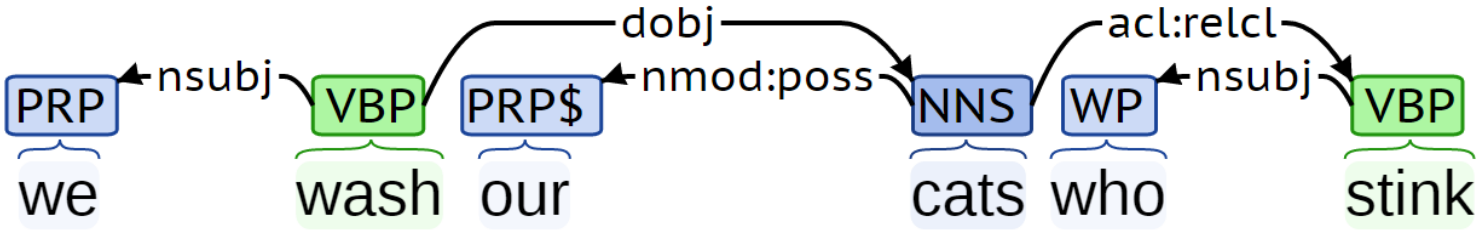



# Nested Structure = Subtrees



# Dependency Labels



# Dependency Labels

| Clausal Argument Relations | Description |
|---|---|
| NSUBJ | Nominal subject |
| DOBJ | Direct object |
| IOBJ | Indirect object |
| CCOMP | Clausal complement |
| XCOMP | Open clausal complement |
| **Nominal Modifier Relations** | **Description** |
| NMOD | Nominal modifier |
| AMOD | Adjectival modifier |
| NUMMOD | Numeric modifier |
| APPOS | Appositional modifier |
| DET | Determiner |
| CASE | Prepositions, postpositions and other case markers |
| **Other Notable Relations** | **Description** |
| CONJ | Conjunct |
| CC | Coordinating conjunction |

# Dependency Trees

# Dependency Trees

$$\{ \langle c_i, p_i, l_i \rangle \}_{i=1 \ldots n}$$

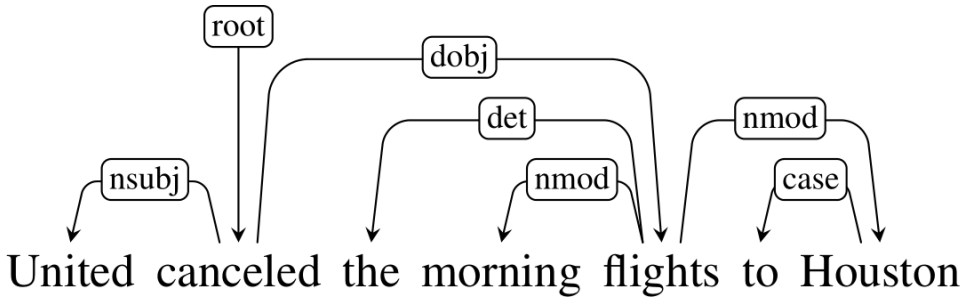
$c_i$: $1 : n$

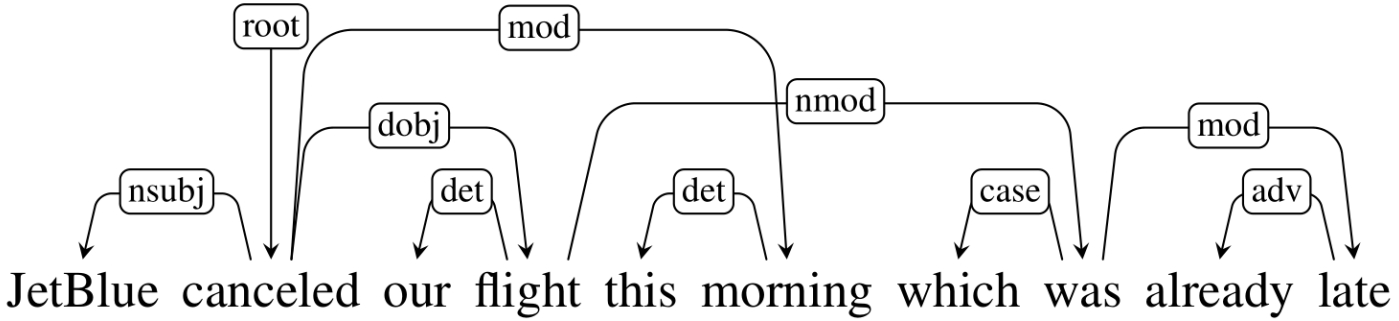
$p_i$: $0 : n$ (root)

$l_i$: label

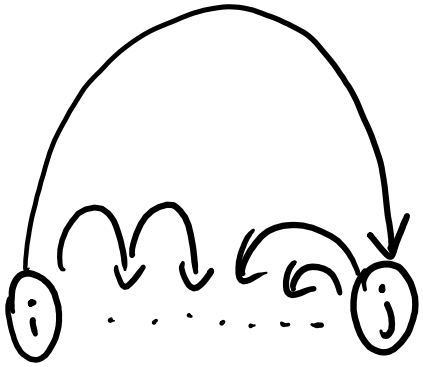
- single headed $\exists i$ only one $p_i = 0$
- connected
- acyclic
- Projective vs non-projective

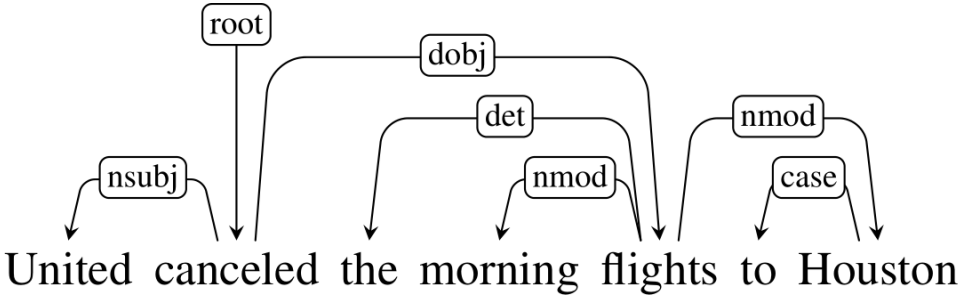
# Projective vs Non-projective

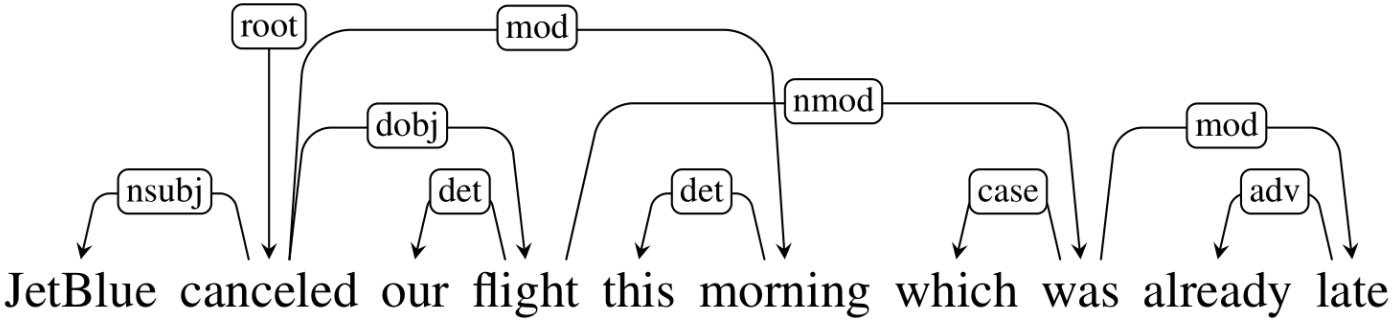


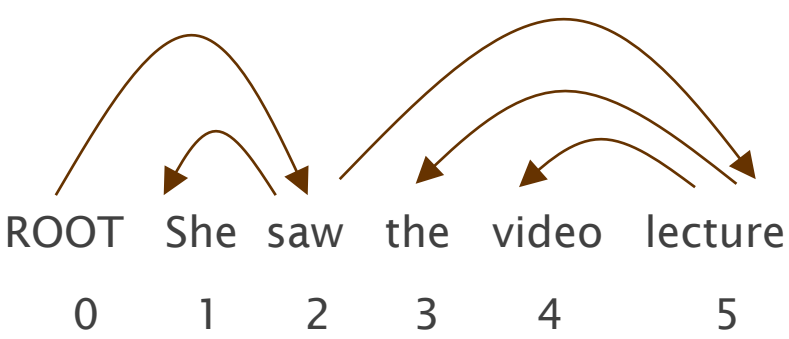
# Projective vs Non-projective

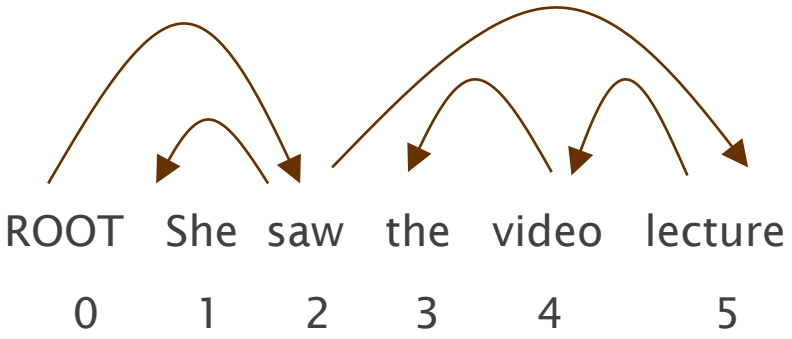


# Evaluating Dependency Parses



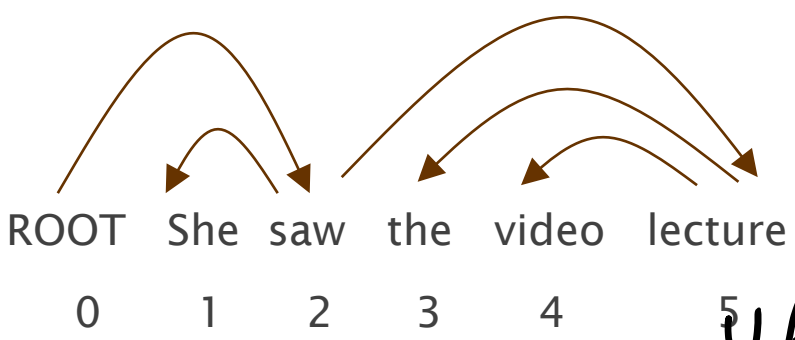

Gold

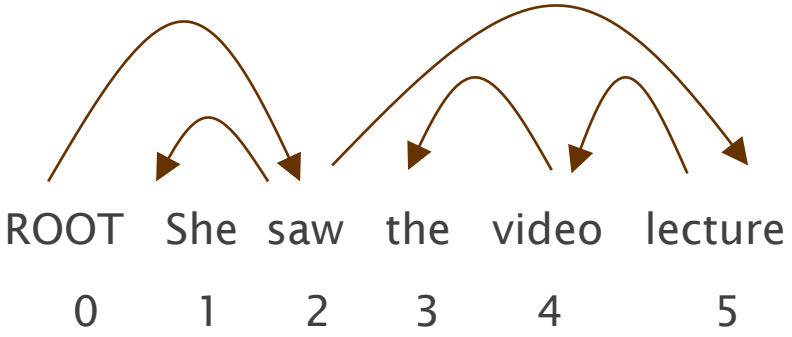
| | | | |
|---|---|---|---|
| 1 | 2 | She | nsubj |
| 2 | 0 | saw | root |
| 3 | 5 | the | det |
| 4 | 5 | video | nn |
| 5 | 2 | lecture | dobj |

Parsed

| | | | |
|---|---|---|---|
| 1 | 2 | She | nsubj |
| 2 | 0 | saw | root |
| 3 | 4 | the | det |
| 4 | 5 | video | nsubj |
| 5 | 2 | lecture | ccomp |

# Evaluating Dependency Parses




UAS

$\frac{4}{5} = 80\%$

LAS

$\frac{2}{5} = 40\%$

| Gold | | | |
|---|---|---|---|
| 1 | 2 | She | nsubj |
| 2 | 0 | saw | root |
| 3 | 5 | the | det |
| 4 | 5 | video | nn |
| 5 | 2 | lecture | dobj |

| Parsed | | | |
|---|---|---|---|
| 1 | 2 | She | nsubj |
| 2 | 0 | saw | root |
| 3 | 4 | the | det |
| 4 | 5 | video | nsubj |
| 5 | 2 | lecture | ccomp |

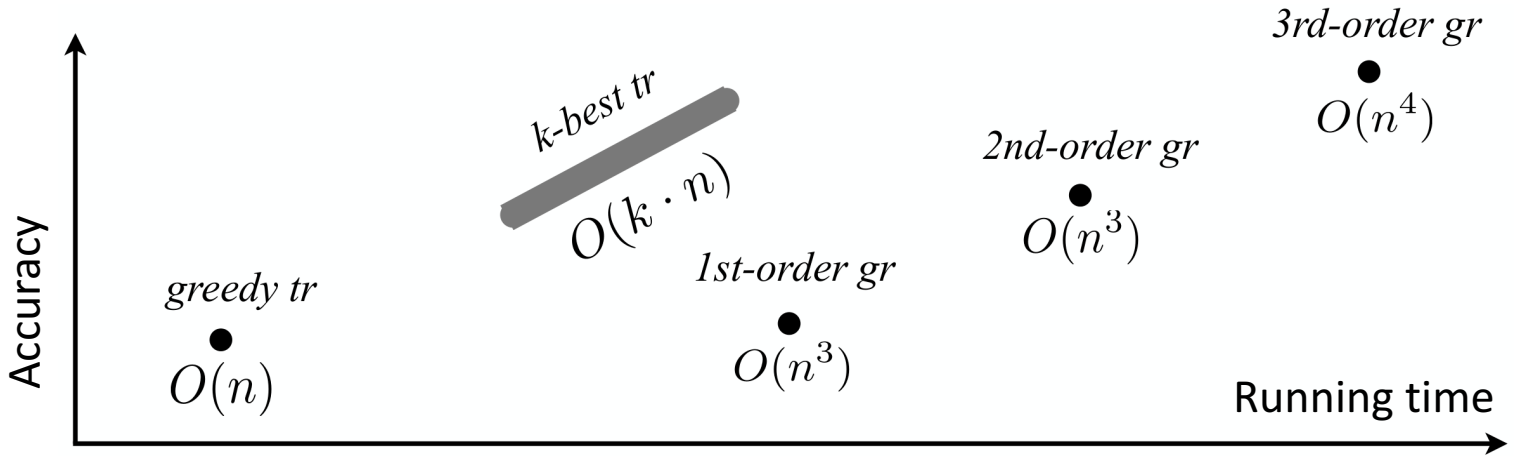
# Parsing Algorithms

**Transition-based**

- Fast, greedy, linear-time
- Trained for greedy search
- Features decide what to do next
- Beam search, i.e. *k*-best

**Graph-based**

- Slower, exhaustive algorithms
- Dynamic programming, inference
- Features used to score whole trees



# Graph-based Parsing

$$\operatorname*{argmax}_{t \in T} \; \text{score}(t, \theta)$$

1st order / fully factored

$$\sum_i \theta_i \, \phi(e_i) \quad (c_i, p_i, l_i)$$

2nd order $\phi(e_i, e_j)$

3rd order $\phi(e_i, e_j, e_k)$

Proj: Dynamic Programming

NonProj: Maximum Spanning Tree

# Transition-based Parsing

| Step | Stack | Word List | Action | Relation Added |
|---|---|---|---|---|
| 0 | [root] | [book, me, the, morning, flight] | SHIFT | |

# Transition-based Parsing

| Step | Stack | Word List | Action | Relation Added |
|---|---|---|---|---|
| 0 | [root] | [book, me, the, morning, flight] | SHIFT | |
| 1 | [root, book] | [me, the, morning, flight] | SHIFT | |

# Transition-based Parsing

| Step | Stack | Word List | Action | Relation Added |
|---|---|---|---|---|
| 0 | [root] | [book, me, the, morning, flight] | SHIFT | |
| 1 | [root, book] | [me, the, morning, flight] | SHIFT | |
| 2 | [root, book, me] | [the, morning, flight] | RIGHTARC | (book → me) |

# Transition-based Parsing

| Step | Stack | Word List | Action | Relation Added |
|---|---|---|---|---|
| 0 | [root] | [book, me, the, morning, flight] | SHIFT | |
| 1 | [root, book] | [me, the, morning, flight] | SHIFT | |
| 2 | [root, book, me] | [the, morning, flight] | RIGHTARC | (book → me) |
| 3 | [root, book] | [the, morning, flight] | SHIFT | |
| 4 | [root, book, the] | [morning, flight] | SHIFT | |
| 5 | [root, book, the, morning] | [flight] | SHIFT | |

# Transition-based Parsing

| Step | Stack | Word List | Action | Relation Added |
|---|---|---|---|---|
| 0 | [root] | [book, me, the, morning, flight] | SHIFT | |
| 1 | [root, book] | [me, the, morning, flight] | SHIFT | |
| 2 | [root, book, me] | [the, morning, flight] | RIGHTARC | (book → me) |
| 3 | [root, book] | [the, morning, flight] | SHIFT | |
| 4 | [root, book, the] | [morning, flight] | SHIFT | |
| 5 | [root, book, the, morning] | [flight] | SHIFT | |
| 6 | [root, book, the, morning, flight] | [] | LEFTARC | (morning ← flight) |
| 7 | [root, book, the, flight] | [] | LEFTARC | (the ← flight) |

# Transition-based Parsing

| Step | Stack | Word List | Action | Relation Added |
|---|---|---|---|---|
| 0 | [root] | [book, me, the, morning, flight] | SHIFT | |
| 1 | [root, book] | [me, the, morning, flight] | SHIFT | |
| 2 | [root, book, me] | [the, morning, flight] | RIGHTARC | (book → me) |
| 3 | [root, book] | [the, morning, flight] | SHIFT | |
| 4 | [root, book, the] | [morning, flight] | SHIFT | |
| 5 | [root, book, the, morning] | [flight] | SHIFT | |
| 6 | [root, book, the, morning, flight] | [] | LEFTARC | (morning ← flight) |
| 7 | [root, book, the, flight] | [] | LEFTARC | (the ← flight) |
| 8 | [root, book, flight] | [] | RIGHTARC | (book → flight) |
| 9 | [root, book] | [] | RIGHTARC | (root → book) |
| 10 | [root] | [] | Done | |

$\theta \cdot \phi$(stack, wlist, action)