

# PCFGs: Parsing & Evaluation

LING 571 — Deep Processing Techniques for NLP  
Shane Steinert-Threlkeld

# Announcements

- HW2 due tonight at 11:59pm
  - readme.{txt|pdf}
    - Separate upload to Canvas
    - NOT in `hw2.tar.gz`
  - Run `check_hw2.sh` before submitting! (Also: `tar -tvf hw2.tar.gz` to preview.)
    - Flat structure; just files, no directories, inside tar-ball
    - Include *only* the files we ask for, not more
- Start symbol: either “%start S” or first nonterminal
  - NB: needs to be readable by nltk’s grammar loading methods
- **Use `nltk.data.load`:** best to use “**file**:path/to/grammar.cfg” as argument
  - Docs: <https://www.nltk.org/api/nltk.data.html#nltk.data.load>
- See hw2 slides as well on website for above points

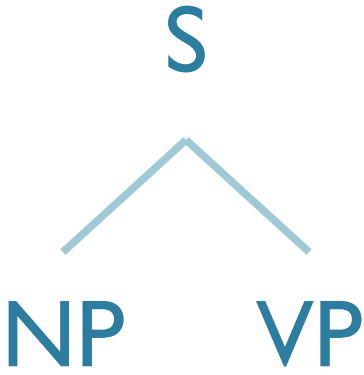
# Roadmap

- CKY + back-pointers example
- PCFGs
- PCFG Parsing (PCKY)
- Inducing a PCFG
- Evaluation
- [Earley parsing]
- HW3 + collaboration

# CKY + Back-pointers Example

cky\_table[0,6][S] = {(NP, (0,1)),  
VP, (1,6))}

<b>NP, Pronoun</b> [0,1]	<b>S</b> [0,2]	[0,3]	<b>S</b> [0,4]	[0,5]	<b>S</b> [0,6]
	<b>Verb, VP, S</b> [1,2]	[1,3]	<b>VP, X2, S</b> [1,4]	[1,5]	<b>VP, X2, S</b> [1,6]
		<b>Det</b> [2,3]	<b>NP</b> [2,4]	[2,5]	<b>NP</b> [2,6]
			<b>Noun, Nom</b> [3,4]	[3,5]	<b>Nom</b> [3,6]
				<b>Prep</b> [4,5]	<b>PP</b> [4,6]
					<b>NNP, NP</b> [5,6]



I prefer a flight on TWA

$S[S] = \{ (NP, ($   
VP, (

S

NP VP

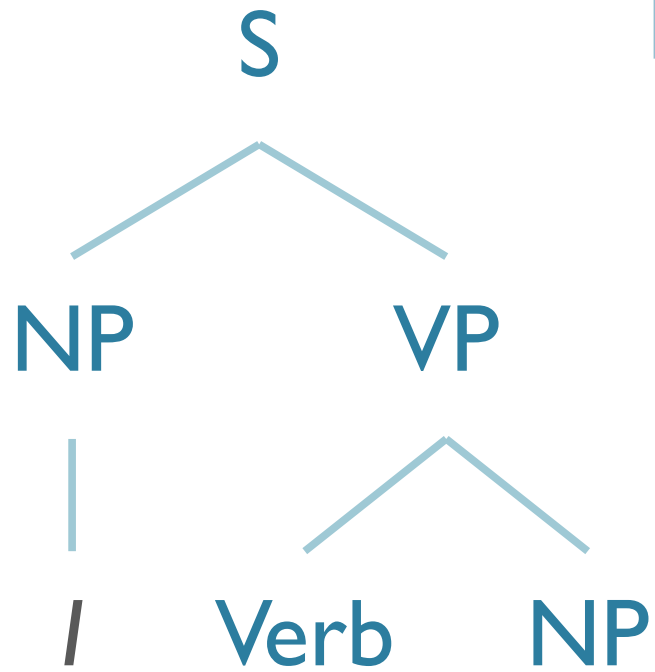
|

/

*I prefer a flight on TWA*

cky\_table[0,6][S] = {(NP, (0,1),  
VP, (1,6))}  
cky\_table[0,1][NP] = {'I'}  
cky\_table[1,6][VP] = {(Verb, (1,2),  
NP, (2,6)),  
(X2, (1,4),  
PP, (4,6))}

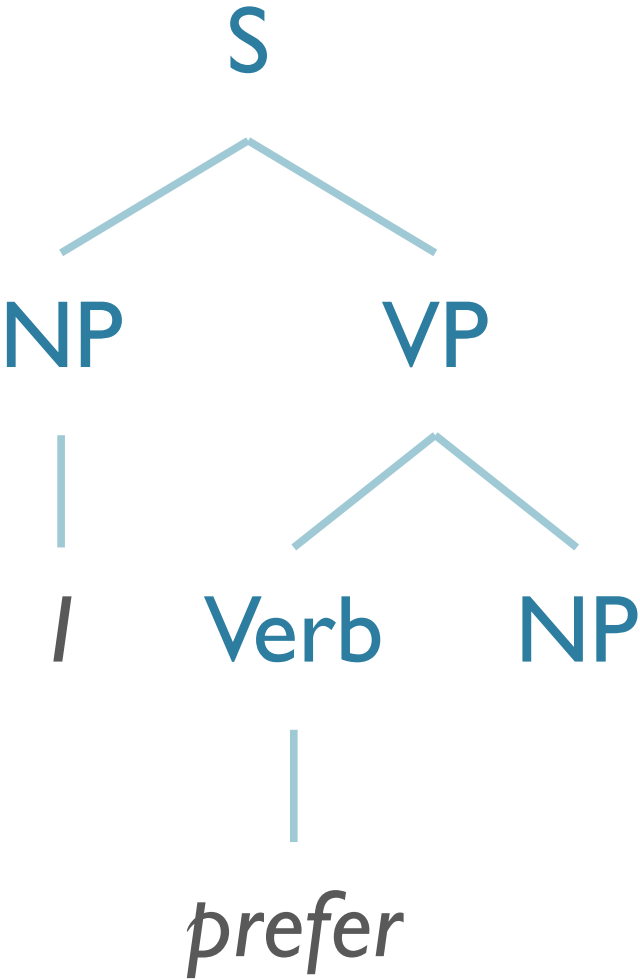
NP, Pronoun [0,1]	S [0,2]	[0,3]	S [0,4]	[0,5]	S [0,6]
	Verb, VP, S [1,2]	[1,3]	VP, X2, S [1,4]	[1,5]	VP, X2, S [1,6]
		Det [2,3]	NP [2,4]	[2,5]	NP [2,6]
			Noun, Nom [3,4]	[3,5]	Nom [3,6]
				Prep [4,5]	PP [4,6]
					NNP, NP [5,6]



I prefer a flight on TWA

cky\_table[0,6][S] = {(NP, (0,1),  
VP, (1,6))}  
cky\_table[0,1][NP] = {'I'}  
cky\_table[1,6][VP] = {(Verb, (1,2),  
NP, (2,6)),  
(X2, (1,4),  
PP, (4,6))}  
cky\_table[1,2][Verb] = {'prefer'}

NP, Pronoun [0,1]	S [0,2]	[0,3]	S [0,4]	[0,5]	S [0,6]
	Verb, VP, S [1,2]	[1,3]	VP, X2, S [1,4]	[1,5]	VP, X2, S [1,6]
		Det [2,3]	NP [2,4]	[2,5]	NP [2,6]
			Noun, Nom [3,4]	[3,5]	Nom [3,6]
				Prep [4,5]	PP [4,6]
					NNP, NP [5,6]



I prefer a flight on TWA



```
cky_table[0,6][S] = {(NP, (0,1),  
                      VP, (1,6))}
```

```
cky_table[0,1][NP] = {('I')}
```

```
cky_table[1,6][VP] = {(Verb, (1,2)),  
                        (NP, (2,6)),  
                        (X2, (1,4)),  
                        (PP, (4,6))}
```

```
cky_table[1,2][Verb] = {('prefer')}
```

```
cky_table[2,6][NP] = {(Det, (2,3)),  
                        (Nom, (3,6))}
```

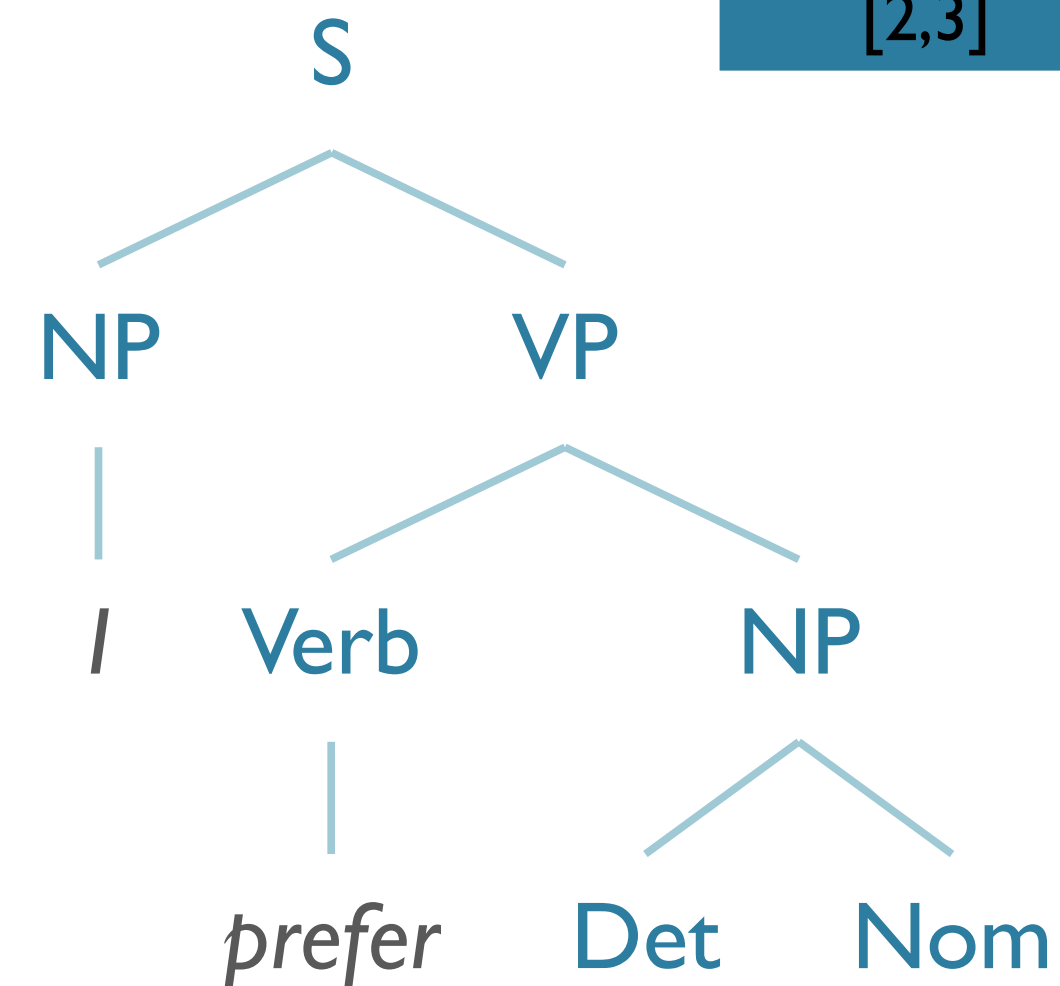
<b>NP, Pronoun</b> [0,1]	<b>S</b> [0,2]	[0,3]	<b>S</b> [0,4]	[0,5]	<b>S</b> [0,6]
	<b>Verb, VP, S</b> [1,2]	[1,3]	<b>VP, X2, S</b> [1,4]	[1,5]	<b>VP, X2, S</b> [1,6]
		<b>Det</b> [2,3]	<b>NP</b> [2,4]	[2,5]	<b>NP</b> [2,6]
			<b>Noun, Nom</b> [3,4]	[3,5]	<b>Nom</b> [3,6]
				<b>Prep</b> [4,5]	<b>PP</b> [4,6]
					<b>NNP, NP</b> [5,6]

Diagram illustrating the structure of a sentence (S) and its components:

```

graph TD
    S --> NP1[NP]
    S --> VP1[VP]
    NP1 --> I[/ /]
    VP1 --> Verb[Verb]
    VP1 --> NP2[NP]
    Verb --> V[ ]
    NP2 --> N1[N]
    NP2 --> N2[N]
    
```

The diagram shows a hierarchical structure for a sentence (S). The root node S branches into NP (Noun Phrase) and VP (Verb Phrase). The NP branches into a slash (/), representing a determiner or pronoun. The VP branches into Verb and NP. The Verb node branches into a vertical line, representing a verb. The NP node branches into two N (Noun) nodes, representing the subject and object of the verb.



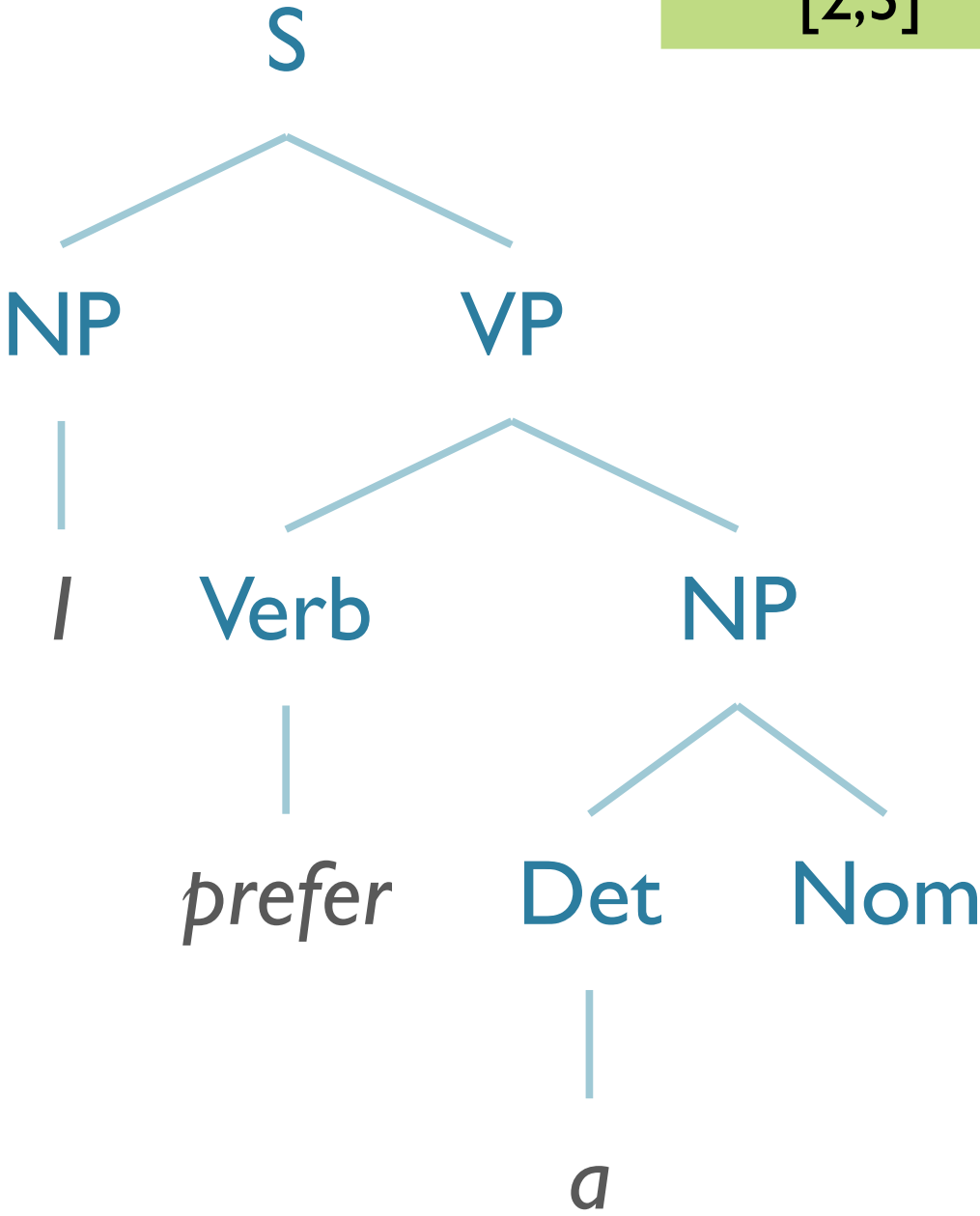
*I prefer a flight on TWA*

```

cky_table[0,6][S] = {(NP, (0,1),
                      VP, (1,6))}
cky_table[0,1][NP] = {'I'}
cky_table[1,6][VP] = {(Verb, (1,2),
                      NP, (2,6)),
                      (X2, (1,4),
                      PP, (4,6))}
cky_table[1,2][Verb] = {'prefer'}
cky_table[2,6][NP] = {(Det, (2,3),
                      Nom, (3,6))}
cky_table[2,3][Det] = {'a'}

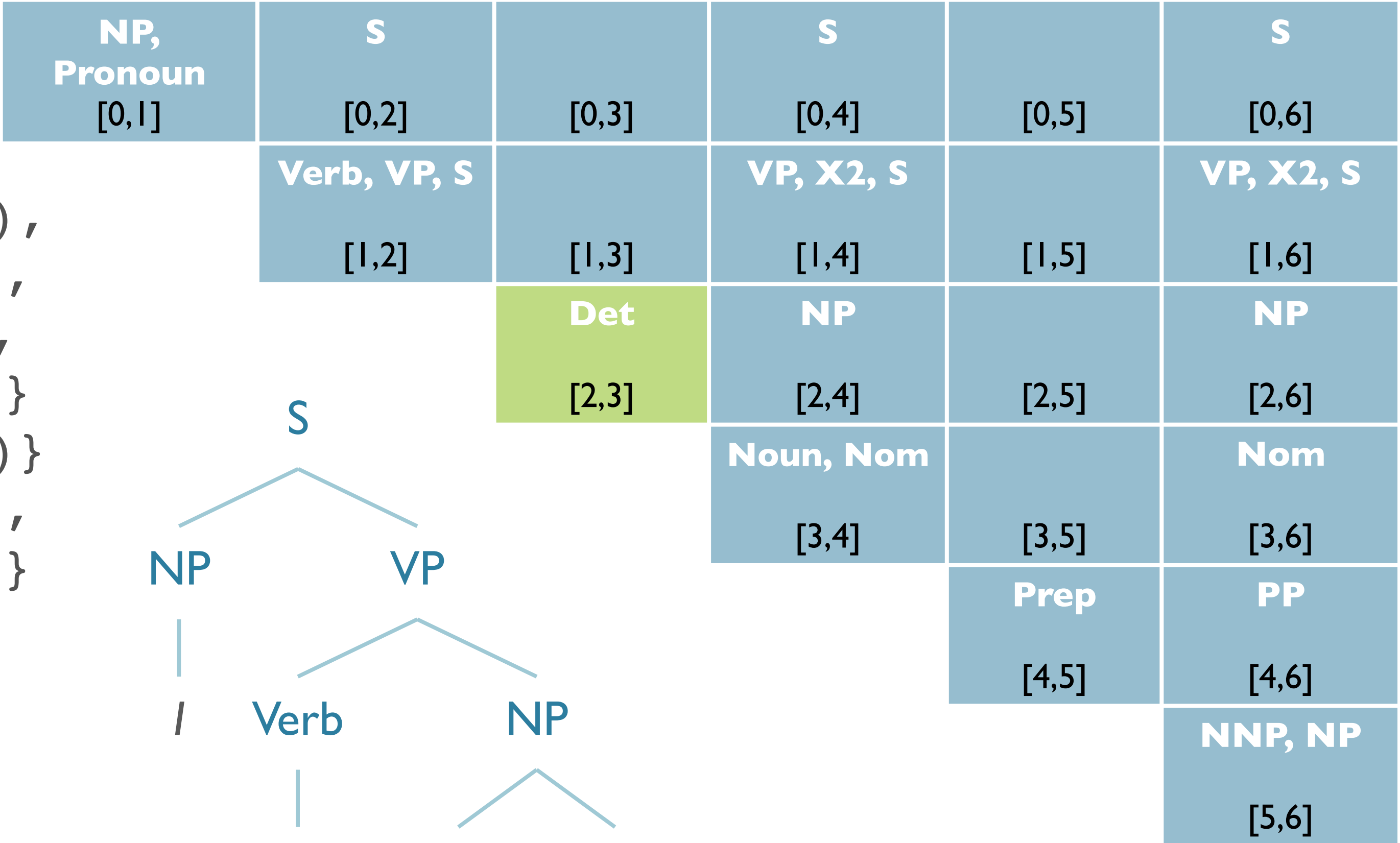
```

<b>NP, Pronoun</b> [0,1]	<b>S</b> [0,2]	[0,3]	<b>S</b> [0,4]	[0,5]	<b>S</b> [0,6]
	<b>Verb, VP, S</b> [1,2]	[1,3]	<b>VP, X2, S</b> [1,4]	[1,5]	<b>VP, X2, S</b> [1,6]
		<b>Det</b> [2,3]	<b>NP</b> [2,4]	[2,5]	<b>NP</b> [2,6]
			<b>Noun, Nom</b> [3,4]	[3,5]	<b>Nom</b> [3,6]
				<b>Prep</b> [4,5]	<b>PP</b> [4,6]
					<b>NNP, NP</b> [5,6]



*I          prefer          a          flight          on          TWA*

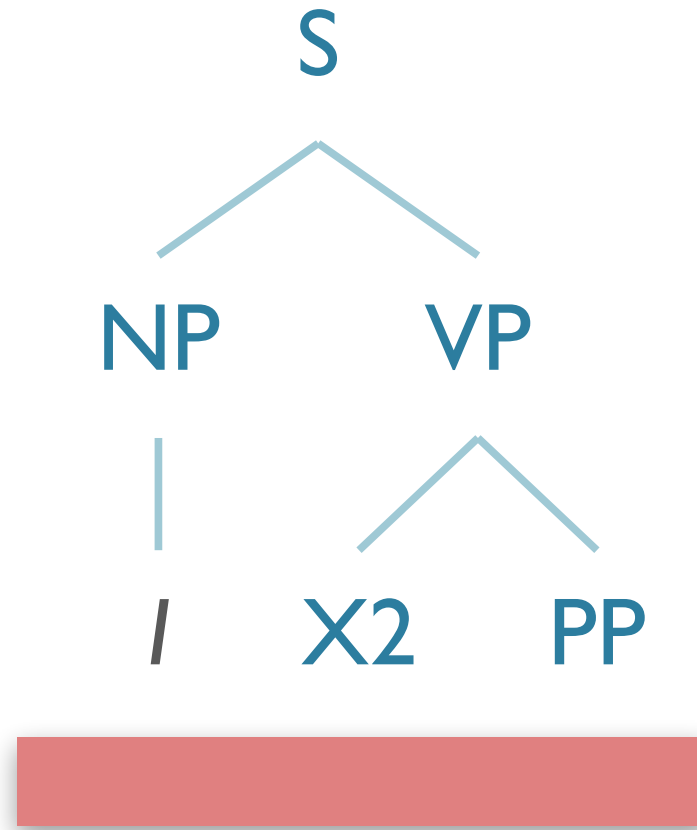
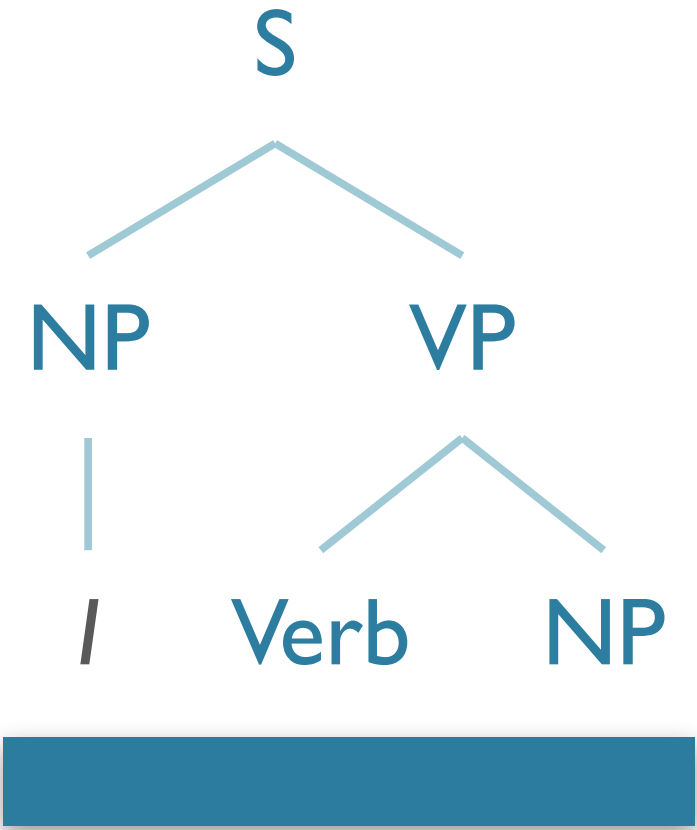
```
cky_table[0,6][S] = {(NP, (0,1),
                        VP, (1,6))}
cky_table[0,1][NP] = {'I'}
cky_table[1,6][VP] = {(Verb, (1,2),
                        NP, (2,6)),
                      (X2, (1,4),
                        PP, (4,6))}
cky_table[1,2][Verb] = {'prefer'}
cky_table[2,6][NP] = {(Det, (2,3),
                        Nom, (3,6))}
cky_table[2,3][Det] = {'a'}
```



*I prefer a flight on TWA*

```
cky_table[0,6][S] = {(NP, (0,1),  
                      VP, (1,6))}  
cky_table[0,1][NP] = {'I'}  
cky_table[1,6][VP] = {(Verb, (1,2),  
                      NP, (2,6))  
                      (X2, (1,4),  
                      PP, (4,6))}
```

<b>NP, Pronoun</b> [0,1]	<b>S</b> [0,2]	[0,3]	<b>S</b> [0,4]	[0,5]	<b>S</b> [0,6]
	<b>Verb, VP, S</b> [1,2]	[1,3]	<b>VP, X2, S</b> [1,4]	[1,5]	<b>VP, X2, S</b> [1,6]
		<b>Det</b> [2,3]	<b>NP</b> [2,4]	[2,5]	<b>NP</b> [2,6]
			<b>Noun, Nom</b> [3,4]	[3,5]	<b>Nom</b> [3,6]
				<b>Prep</b> [4,5]	<b>PP</b> [4,6]
					<b>NNP, NP</b> [5,6]



*I prefer a flight on TWA*

# Probabilistic Context-Free Grammars

# Probabilistic Context-free Grammars: Roadmap

Motivation: Ambiguity

Approach:

Definition

Disambiguation

Parsing

Evaluation

Enhancements

# Motivation

What about ambiguity?

Current algorithm can ***represent*** it

...can't resolve it.

# Probabilistic Parsing

- Provides strategy for solving disambiguation problem
  - Compute the probability of all analyses
  - Select the most probable
- Employed in language modeling for speech recognition
  - N-gram grammars predict words, constrain search
  - Also, constrain generation, translation



# PCFGs: Formal Definition

$N$

a set of **non-terminal symbols** (or **variables**)

---

# PCFGs: Formal Definition

$N$	a set of <b>non-terminal symbols</b> (or <b>variables</b> )
-----	---

$\Sigma$	a set of <b>terminal symbols</b> (disjoint from $N$ )
----------	---

# PCFGs: Formal Definition

$N$	a set of <b>non-terminal symbols</b> (or <b>variables</b> )
$\Sigma$	a set of <b>terminal symbols</b> (disjoint from $N$ )
$R$	a set of rules of productions, each of the form $A \rightarrow \beta[p]$ , where $A$ is a non-terminal where $A$ is a non-terminal, $\beta$ is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$ and $p$ is a number between 0 and 1 expressing $P(\beta A)$

# PCFGs: Formal Definition

$N$	a set of <b>non-terminal symbols</b> (or <b>variables</b> )
$\Sigma$	a set of <b>terminal symbols</b> (disjoint from $N$ )
$R$	a set of rules of productions, each of the form $A \rightarrow \beta[p]$ , where $A$ is a non-terminal where $A$ is a non-terminal, $\beta$ is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$ and $p$ is a number between 0 and 1 expressing $P(\beta A)$
$S$	a designated <b>start symbol</b>

# PCFGs

- Augment each production with probability that LHS will be expanded as RHS
  - $P(A \rightarrow \beta)$
  - $P(A \rightarrow \beta | A)$
  - $P(\beta | A)$
  - $P(RHS | LHS)$
- NB: the first is often used; but the latter are what's really meant.

# PCFGs

- Sum over all possible expansions is **1**

$$\sum_{\beta} P(A \rightarrow \beta) = 1$$

- A PCFG is **consistent** if sum of probabilities of all sentences in language is **1**
- Recursive rules often yield inconsistent grammars ([Booth & Thompson, 1973](#))

# Example PCFG: Augmented $\mathcal{L}_1$

Grammar		Lexicon
$S \rightarrow NP VP$	[.80]	$Det \rightarrow that [.10] \mid a [.30] \mid the [.60]$
$S \rightarrow Aux NP VP$	[.15]	$Noun \rightarrow book [.10] \mid flight [.30] \mid meal [.15] \mid money [0.5]$
$S \rightarrow VP$	[.05]	$\mid flights [0.40] \mid dinner [.10]$
$NP \rightarrow Pronoun$	[.35]	$Verb \rightarrow book [.30] \mid include [.30] \mid prefer [.40]$
$NP \rightarrow Proper-Noun$	[.30]	$Pronoun \rightarrow I [.40] \mid she [.05] \mid me [.15] \mid you [.40]$
$NP \rightarrow Det Nominal$	[.20]	$Proper-Noun \rightarrow Houston [.60] \mid NWA [.40]$
$NP \rightarrow Nominal$	[.15]	$Aux \rightarrow does [.60] \mid can [.40]$
$Nominal \rightarrow Noun$	[.75]	$Preposition \rightarrow from [.30] \mid to [.30] \mid on [.20] \mid near [.15]$
$Nominal \rightarrow Nominal Noun$	[.20]	$\mid through [.05]$
$Nominal \rightarrow Nominal PP$	[.05]	
$VP \rightarrow Verb$	[.35]	
$VP \rightarrow Verb NP$	[.20]	
$VP \rightarrow Verb NP PP$	[.10]	
$VP \rightarrow Verb PP$	[.15]	
$VP \rightarrow Verb NP NP$	[.05]	
$VP \rightarrow VP PP$	[.15]	
$PP \rightarrow Preposition NP$	[1.0]	

# Example PCFG: Augmented $\mathcal{L}_1$

Grammar		Lexicon	
$S \rightarrow NP VP$	[.80]	$Det \rightarrow that [.10] \mid a [.30] \mid the [.60]$	
$S \rightarrow Aux NP VP$	[.15]	$Noun \rightarrow book [.10] \mid flight [.30] \mid meal [.15] \mid money [0.5]$	
$S \rightarrow VP$	[.05]	$\mid flights [0.40] \mid dinner [.10]$	
$NP \rightarrow Pronoun$	[.35]	$Verb \rightarrow book [.30] \mid include [.30] \mid prefer [.40]$	
$NP \rightarrow Proper-Noun$	[.30]	$Pronoun \rightarrow I [.40] \mid she [.05] \mid me [.15] \mid you [.40]$	
$NP \rightarrow Det Nominal$	[.20]	$Proper-Noun \rightarrow Houston [.60] \mid NWA [.40]$	
$NP \rightarrow Nominal$	[.15]	$Aux \rightarrow does [.60] \mid can [.40]$	
$Nominal \rightarrow Noun$	[.75]	$Preposition \rightarrow from [.30] \mid to [.30] \mid on [.20] \mid near [.15]$	
$Nominal \rightarrow Nominal Noun$	[.20]	$\mid through [.05]$	
$Nominal \rightarrow Nominal PP$	[.05]		
$VP \rightarrow Verb$	[.35]		
$VP \rightarrow Verb NP$	[.20]		
$VP \rightarrow Verb NP PP$	[.10]		
$VP \rightarrow Verb PP$	[.15]		
$VP \rightarrow Verb NP NP$	[.05]		
$VP \rightarrow VP PP$	[.15]		
$PP \rightarrow Preposition NP$	[1.0]		

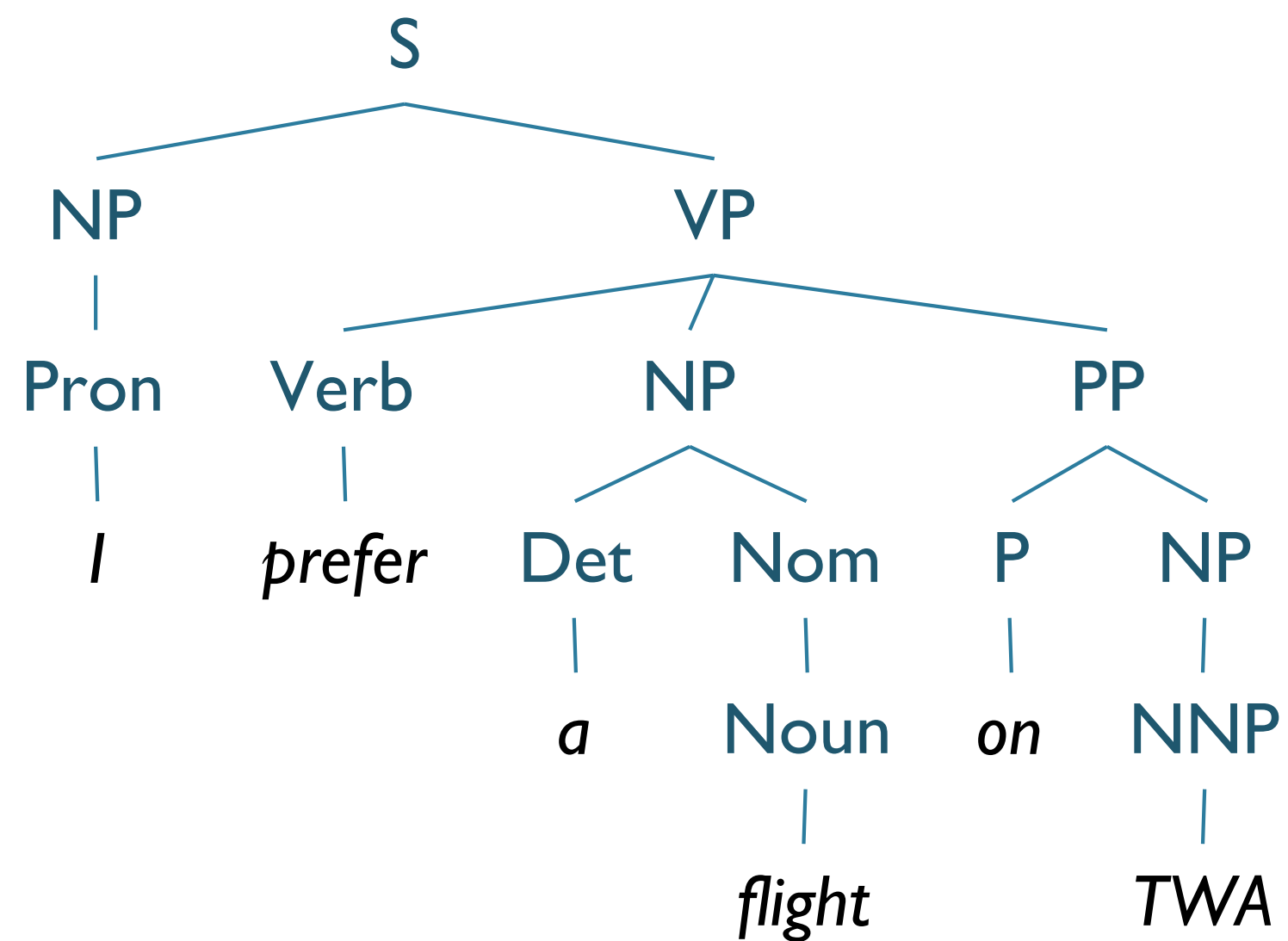


# Disambiguation

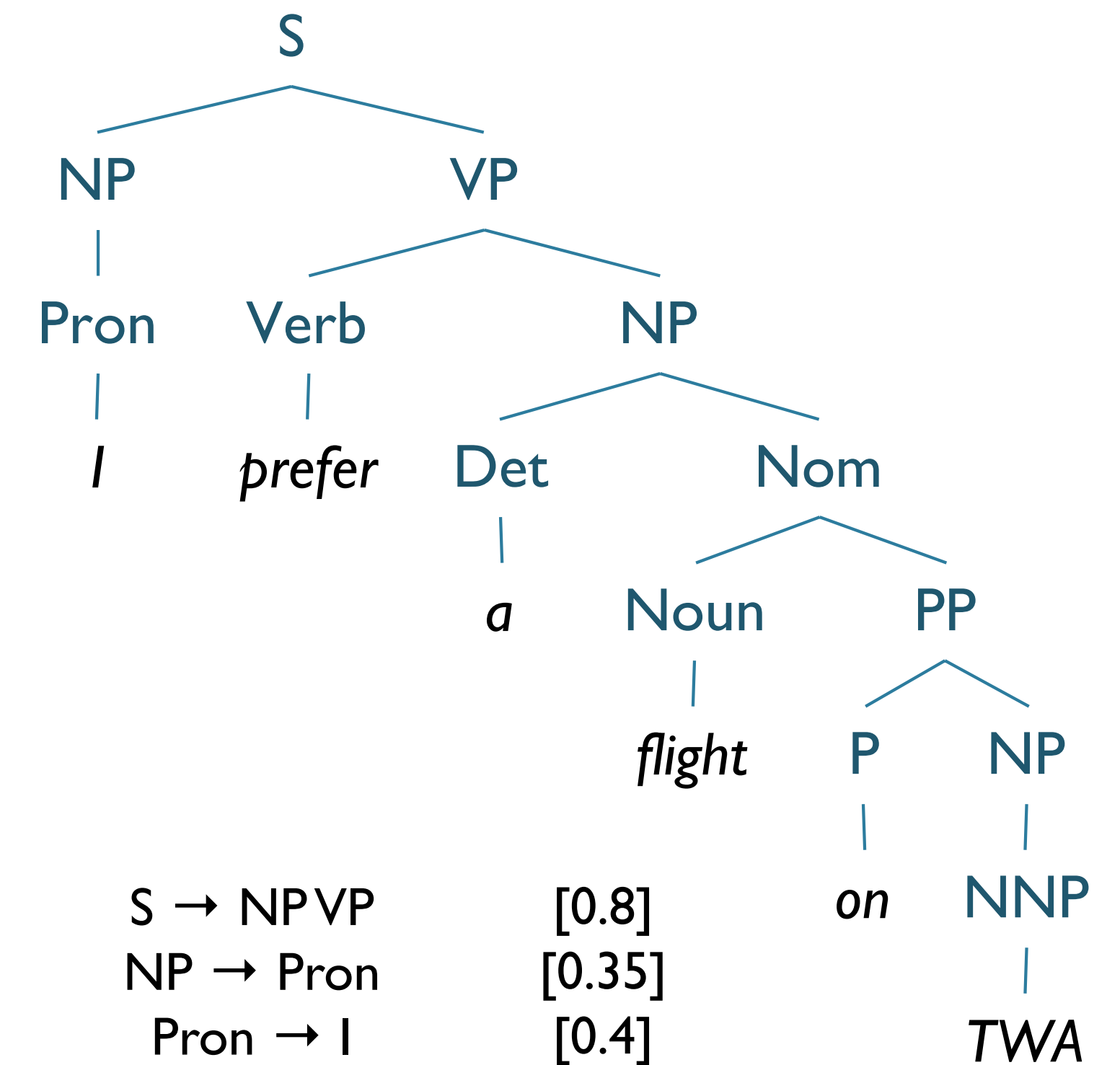
- A PCFG assigns probability to each parse tree  $T$  for input  $S$
- Probability of  $T$ : product of all rules used to derive  $T$

$$P(T, S) = \prod_{i=1}^n P(RHS_i | LHS_i)$$

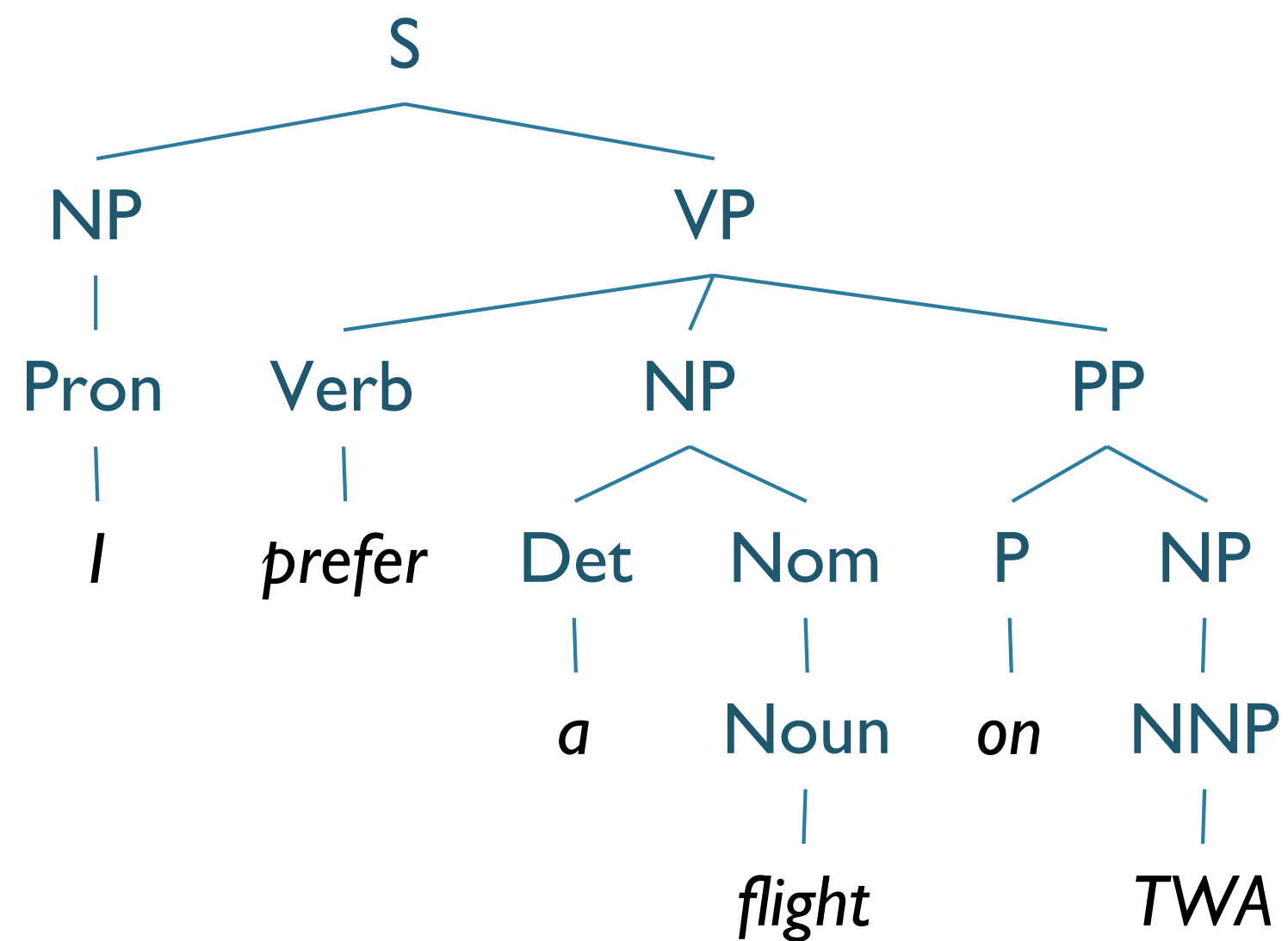
$$P(T, S) = P(T)P(S | T) = P(T)$$



$S \rightarrow NP VP$	[0.8]
$NP \rightarrow Pron$	[0.35]
$Pron \rightarrow I$	[0.4]
$VP \rightarrow V NP PP$	[0.1]
$V \rightarrow prefer$	[0.4]
$NP \rightarrow Det Nom$	[0.2]
$Det \rightarrow a$	[0.3]
$Nom \rightarrow N$	[0.75]
$N \rightarrow flight$	[0.3]
$PP \rightarrow P NP$	[1.0]
$P \rightarrow on$	[0.2]
$NP \rightarrow NNP$	[0.3]
$NNP \rightarrow NWA$	[0.4]

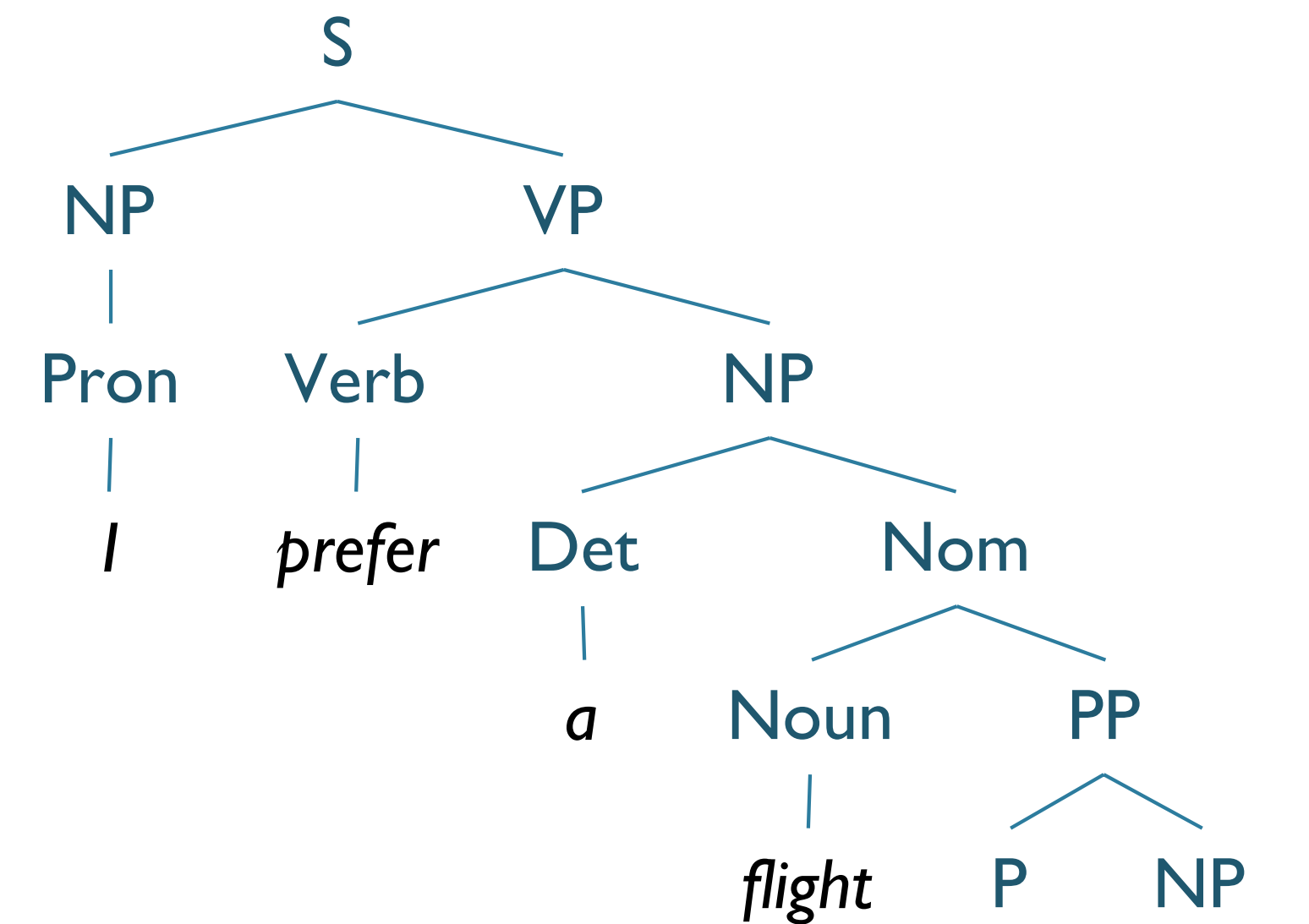


$S \rightarrow NP VP$	[0.8]
$NP \rightarrow Pron$	[0.35]
$Pron \rightarrow I$	[0.4]
$VP \rightarrow V NP$	[0.2]
$V \rightarrow prefer$	[0.4]
$NP \rightarrow Det Nom$	[0.2]
$Det \rightarrow a$	[0.3]
$Nom \rightarrow Nom PP$	[0.05]
$Nom \rightarrow N$	[0.75]
$N \rightarrow flight$	[0.3]
$PP \rightarrow P NP$	[1.0]
$P \rightarrow on$	[0.2]
$NP \rightarrow NNP$	[0.3]
$NNP \rightarrow NWA$	[0.4]



$S \rightarrow NP VP$	[0.8]
$NP \rightarrow Pron$	[0.35]
$Pron \rightarrow I$	[0.4]
$VP \rightarrow V NP PP$	[0.1]
$V \rightarrow prefer$	[0.4]
$NP \rightarrow Det Nom$	[0.2]
$Det \rightarrow a$	[0.3]
$Nom \rightarrow N$	[0.75]
$N \rightarrow flight$	[0.3]
$PP \rightarrow P NP$	[1.0]
$P \rightarrow on$	[0.2]
$NP \rightarrow NNP$	[0.3]
$NNP \rightarrow NWA$	[0.4]

$\sim 1.452 \times 10^{-6}$



$S \rightarrow NP VP$	[0.8]
$NP \rightarrow Pron$	[0.35]
$Pron \rightarrow I$	[0.4]
$VP \rightarrow V NP$	[0.2]
$V \rightarrow prefer$	[0.4]
$NP \rightarrow Det Nom$	[0.2]
$Det \rightarrow a$	[0.3]
$Nom \rightarrow Nom PP$	[0.05]
$Nom \rightarrow N$	[0.75]
$N \rightarrow flight$	[0.3]
$PP \rightarrow P NP$	[1.0]
$P \rightarrow on$	[0.2]
$NP \rightarrow NNP$	[0.3]
$NNP \rightarrow NWA$	[0.4]

$\sim 1.452 \times 10^{-7}$

# Parsing Problem for PCFGs

- Select  $T$  such that (*s.t.*)

$$\hat{T}(S) = \operatorname{argmax}_{T \text{ s.t. } S=\text{yield}(T)} P(T)$$

- String of words  $S$  is *yield* of parse tree
- Select the tree  $\hat{T}$  that maximizes the probability of the parse

# Application: Language Modeling

- $n$ -grams helpful for modeling the probability of a string

# Application: Language Modeling

- $n$ -grams helpful for modeling the probability of a string
- To model a whole sentence with  $n$ -grams either:

# Application: Language Modeling

- $n$ -grams helpful for modeling the probability of a string
- To model a whole sentence with  $n$ -grams either:
  - Must use 10+-grams... too sparse

# Application: Language Modeling

- $n$ -grams helpful for modeling the probability of a string
- To model a whole sentence with  $n$ -grams either:

- Must use 10+-grams... too sparse

- Approximate using conditioning on limited context:  $P(w_i | w_{i-1}) = \frac{P(w_{i-1}, w_i)}{P(w_{i-1})}$



# Application: Language Modeling

- $n$ -grams helpful for modeling the probability of a string
- To model a whole sentence with  $n$ -grams either:
  - Must use 10+-grams... too sparse
  - Approximate using conditioning on limited context:  $P(w_i | w_{i-1}) = \frac{P(w_{i-1}, w_i)}{P(w_{i-1})}$
- PCFGs are able to give probability of entire string without as bad sparsity

# Application: Language Modeling

- $n$ -grams helpful for modeling the probability of a string
- To model a whole sentence with  $n$ -grams either:
  - Must use 10+-grams... too sparse
  - Approximate using conditioning on limited context:  $P(w_i | w_{i-1}) = \frac{P(w_{i-1}, w_i)}{P(w_{i-1})}$
- PCFGs are able to give probability of entire string without as bad sparsity
- Model probability of *syntactically valid* sentences

# Application: Language Modeling

- $n$ -grams helpful for modeling the probability of a string
- To model a whole sentence with  $n$ -grams either:
  - Must use 10+-grams... too sparse
  - Approximate using conditioning on limited context:  $P(w_i | w_{i-1}) = \frac{P(w_{i-1}, w_i)}{P(w_{i-1})}$
- PCFGs are able to give probability of entire string without as bad sparsity
- Model probability of *syntactically valid* sentences
  - Not just probability of sequence of words

# PCFGs: Parsing

# Probabilistic CKY (PCKY)

- Like regular CKY
  - Assumes grammar in Chomsky Normal Form (CNF)
    - $A \rightarrow B C$
    - $A \rightarrow w$
  - Represent input with indices b/t words:
    - $_0 \text{ Book } _1 \text{ that } _2 \text{ flight } _3 \text{ through } _4 \text{ Houston } _5$

# Probabilistic CKY (PCKY)

- For input string length  $n$  and non-terminals  $V$ 
  - Cell  $[i, j, A]$  in  $(n+1) \times (n+1) \times V$  matrix
  - Contains probability that  $A$  spans  $[i, j]$

# PCKY Algorithm

```
function PROBABILISTIC-CKY-PARSE(words, grammar) returns most probable parse and its probability
for j  $\leftarrow$  from 1 to LENGTH(words) do
  for all { A |  $A \rightarrow words[j] \in grammar$  }
     $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$ 
  for i  $\leftarrow$  from j-2 downto 0 do
    for k  $\leftarrow$  i + 1 to j-1 do
      for all { A |  $A \rightarrow BC \in grammar$ ,
        and  $table[i, k, B] > 0$  and  $table[k, j, C] > 0$  }
        if ( $table[i, j, A] < P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ ) then
           $table[i, j, A] \leftarrow P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ 
           $back[i, j, A] \leftarrow \{ k, B, C \}$ 
  return BUILD_TREE( $back[1, LENGTH(words), S]$ ),  $table[1, LENGTH(words), S]$ 
```

# PCKY Algorithm

```
function PROBABILISTIC-CKY-PARSE(words, grammar) returns most probable parse and its probability
for j  $\leftarrow$  from 1 to LENGTH(words) do
  for all { A |  $A \rightarrow words[j] \in grammar$  }
     $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$ 
  for i  $\leftarrow$  from j-2 downto 0 do
    for k  $\leftarrow$  i + 1 to j-1 do
      for all { A |  $A \rightarrow BC \in grammar,$ 
        and  $table[i, k, B] > 0$  and  $table[k, j, C] > 0$  }
        if ( $table[i, j, A] < P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ ) then
           $table[i, j, A] \leftarrow P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ 
           $back[i, j, A] \leftarrow \{ k, B, C \}$ 
  return BUILD_TREE( $back[1, LENGTH(words), S]$ ),  $table[1, LENGTH(words), S]$ 
```



# PCKY Algorithm

```
function PROBABILISTIC-CKY-PARSE(words, grammar) returns most probable parse and its probability
for j  $\leftarrow$  from 1 to LENGTH(words) do
  for all { A |  $A \rightarrow words[j] \in grammar$  }
     $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$ 
  for i  $\leftarrow$  from j-2 downto 0 do
    for k  $\leftarrow$  i + 1 to j-1 do
      for all { A |  $A \rightarrow BC \in grammar$ ,
        and  $table[i, k, B] > 0$  and  $table[k, j, C] > 0$  }
        if ( $table[i, j, A] < P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ ) then
           $table[i, j, A] \leftarrow P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ 
           $back[i, j, A] \leftarrow \{ k, B, C \}$ 
  return BUILD_TREE( $back[1, LENGTH(words), S]$ ),  $table[1, LENGTH(words), S]$ 
```

# PCKY Algorithm

```
function PROBABILISTIC-CKY-PARSE(words, grammar) returns most probable parse and its probability
for j  $\leftarrow$  from 1 to LENGTH(words) do
  for all { A |  $A \rightarrow \text{words}[j] \in \text{grammar}$  }
     $\text{table}[j-1, j, A] \leftarrow P(A \rightarrow \text{words}[j])$ 
  for i  $\leftarrow$  from j-2 downto 0 do
    for k  $\leftarrow$  i + 1 to j-1 do
      for all { A |  $A \rightarrow BC \in \text{grammar}$ ,
        and  $\text{table}[i, k, B] > 0$  and  $\text{table}[k, j, C] > 0$  }
        if ( $\text{table}[i, j, A] < P(A \rightarrow BC) \times \text{table}[i, k, B] \times \text{table}[k, j, C]$ ) then
           $\text{table}[i, j, A] \leftarrow P(A \rightarrow BC) \times \text{table}[i, k, B] \times \text{table}[k, j, C]$ 
           $\text{back}[i, j, A] \leftarrow \{k, B, C\}$ 
  return BUILD_TREE( $\text{back}[1, \text{LENGTH}(\text{words}), S]$ ),  $\text{table}[1, \text{LENGTH}(\text{words}), S]$ 
```

# PCKY Algorithm

```
function PROBABILISTIC-CKY-PARSE(words, grammar) returns most probable parse and its probability
for j  $\leftarrow$  from 1 to LENGTH(words) do
  for all { A |  $A \rightarrow \text{words}[j] \in \text{grammar}$  }
     $\text{table}[j-1, j, A] \leftarrow P(A \rightarrow \text{words}[j])$ 
  for i  $\leftarrow$  from j-2 downto 0 do
    for k  $\leftarrow$  i + 1 to j-1 do
      for all { A |  $A \rightarrow BC \in \text{grammar}$ ,
        and  $\text{table}[i, k, B] > 0$  and  $\text{table}[k, j, C] > 0$  }
        if ( $\text{table}[i, j, A] < P(A \rightarrow BC) \times \text{table}[i, k, B] \times \text{table}[k, j, C]$ ) then
           $\text{table}[i, j, A] \leftarrow P(A \rightarrow BC) \times \text{table}[i, k, B] \times \text{table}[k, j, C]$ 
           $\text{back}[i, j, A] \leftarrow \{k, B, C\}$ 
  return BUILD_TREE( $\text{back}[1, \text{LENGTH}(\text{words}), S]$ ),  $\text{table}[1, \text{LENGTH}(\text{words}), S]$ 
```

# PCKY Grammar Segment

$S \rightarrow NP VP$  [0.80]

$NP \rightarrow Det N$  [0.30]

$VP \rightarrow V NP$  [0.20]

$Det \rightarrow \text{the}$  [0.40]

$Det \rightarrow \text{a}$  [0.40]

$V \rightarrow \text{includes}$  [0.05]

$N \rightarrow \text{meal}$  [0.01]

$N \rightarrow \text{flight}$  [0.02]

# PCKY Matrix

$S \rightarrow NP VP$  [0.80]

$NP \rightarrow Det N$  [0.30]

$VP \rightarrow V NP$  [0.20]

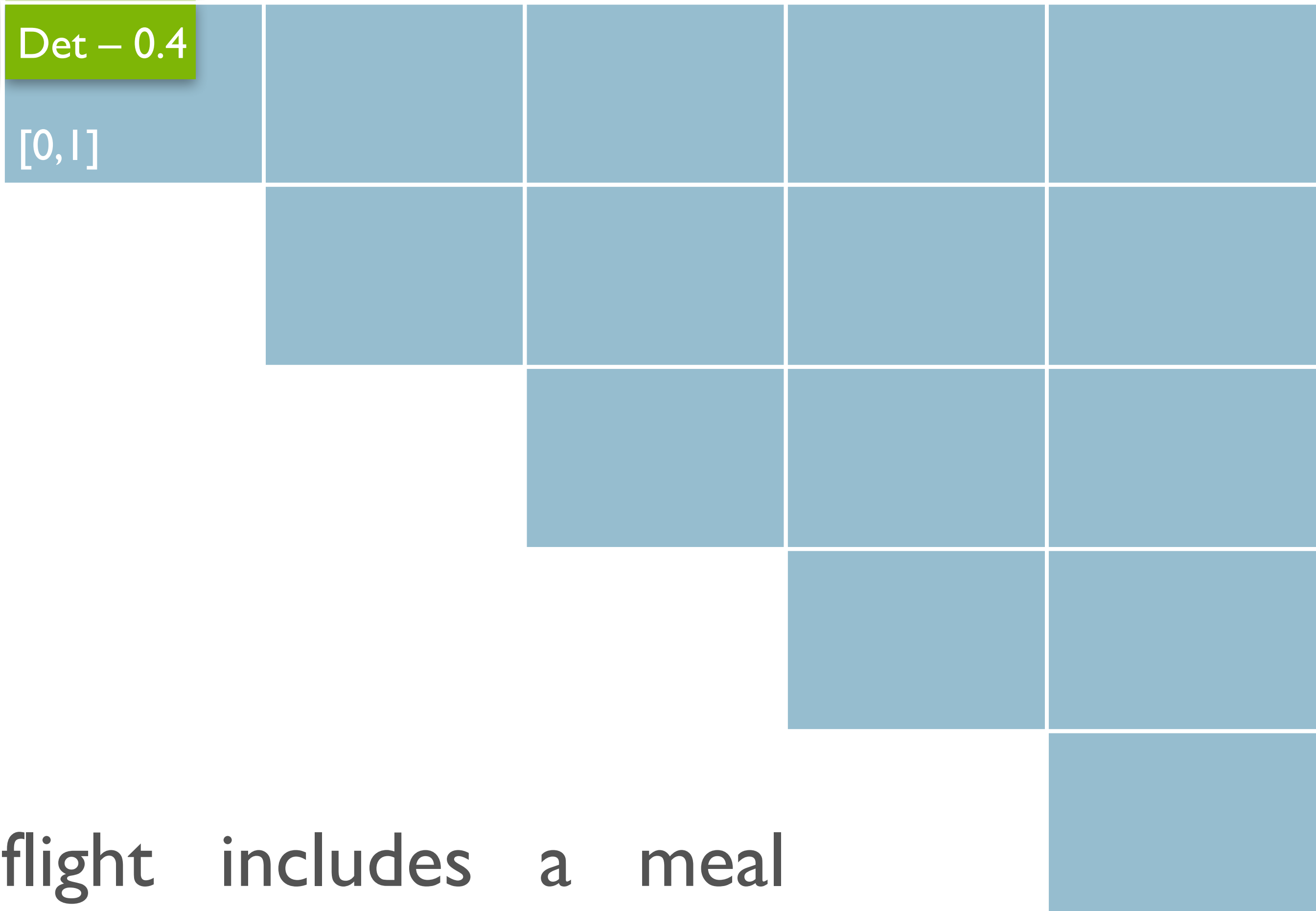
$Det \rightarrow the$  [0.40]

$Det \rightarrow a$  [0.40]

$V \rightarrow includes$  [0.05]

$N \rightarrow meal$  [0.01]

$N \rightarrow flight$  [0.02]



0      1      2      3      4      5

The flight includes a meal

# PCKY Matrix

$S \rightarrow NP VP$  [0.80]

$NP \rightarrow Det N$  [0.30]

$VP \rightarrow V NP$  [0.20]

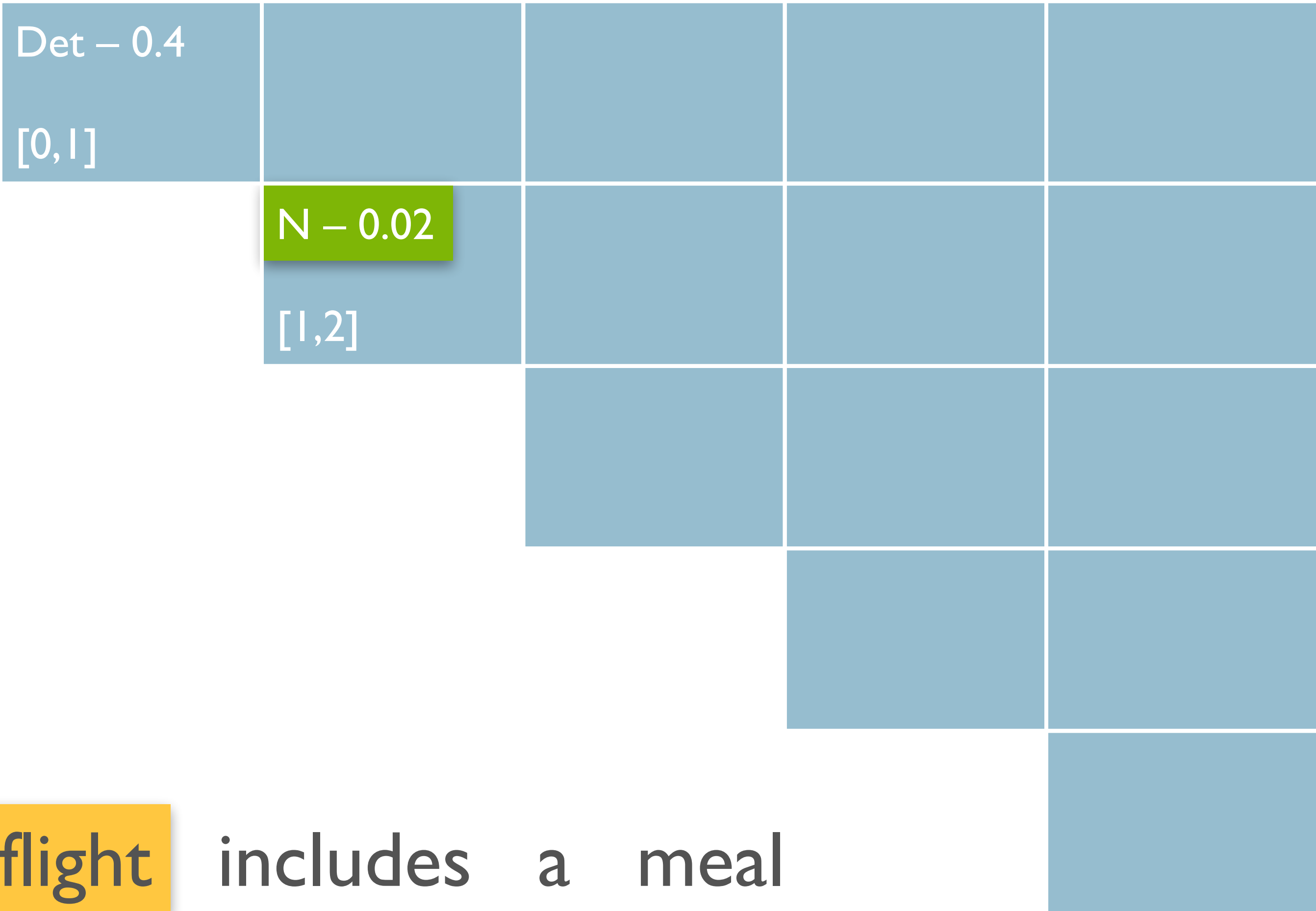
$Det \rightarrow the$  [0.40]

$Det \rightarrow a$  [0.40]

$V \rightarrow includes$  [0.05]

$N \rightarrow meal$  [0.01]

$N \rightarrow flight$  [0.02]



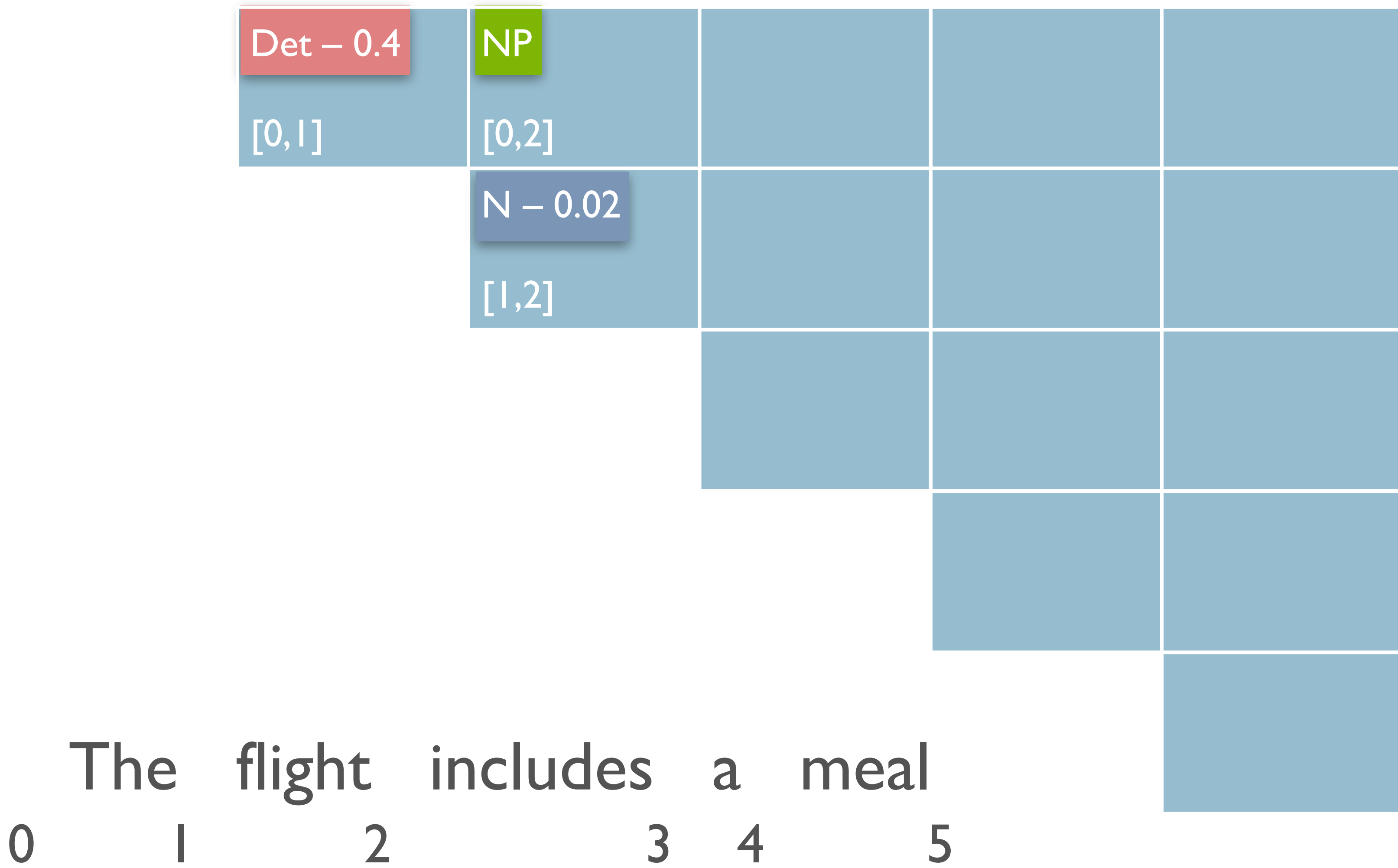
0      1      2      3      4      5

The flight includes a meal

# PCKY Matrix

$S \rightarrow NP VP$  [0.80]  
 $NP \rightarrow Det N$  [0.30]  
 $VP \rightarrow V NP$  [0.20]

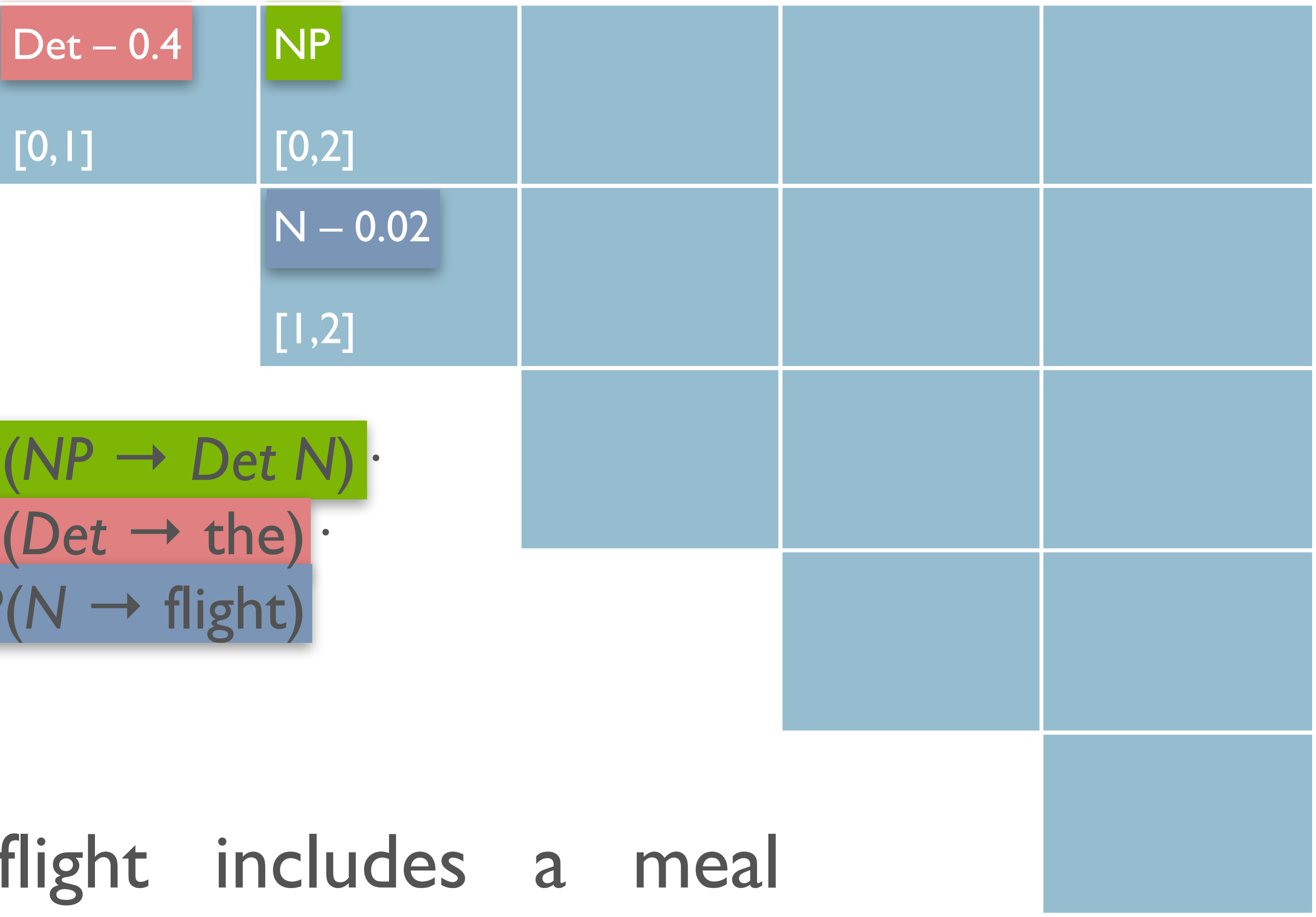
$Det \rightarrow the$  [0.40]  
 $Det \rightarrow a$  [0.40]  
 $V \rightarrow includes$  [0.05]  
 $N \rightarrow meal$  [0.01]  
 $N \rightarrow flight$  [0.02]



# PCKY Matrix

$S \rightarrow NP VP$  [0.80]  
 $NP \rightarrow Det N$  [0.30]  
 $VP \rightarrow V NP$  [0.20]

$Det \rightarrow the$  [0.40]  
 $Det \rightarrow a$  [0.40]  
 $V \rightarrow includes$  [0.05]  
 $N \rightarrow meal$  [0.01]  
 $N \rightarrow flight$  [0.02]



$$P = P(NP \rightarrow Det N) \cdot P(Det \rightarrow the) \cdot P(N \rightarrow flight)$$

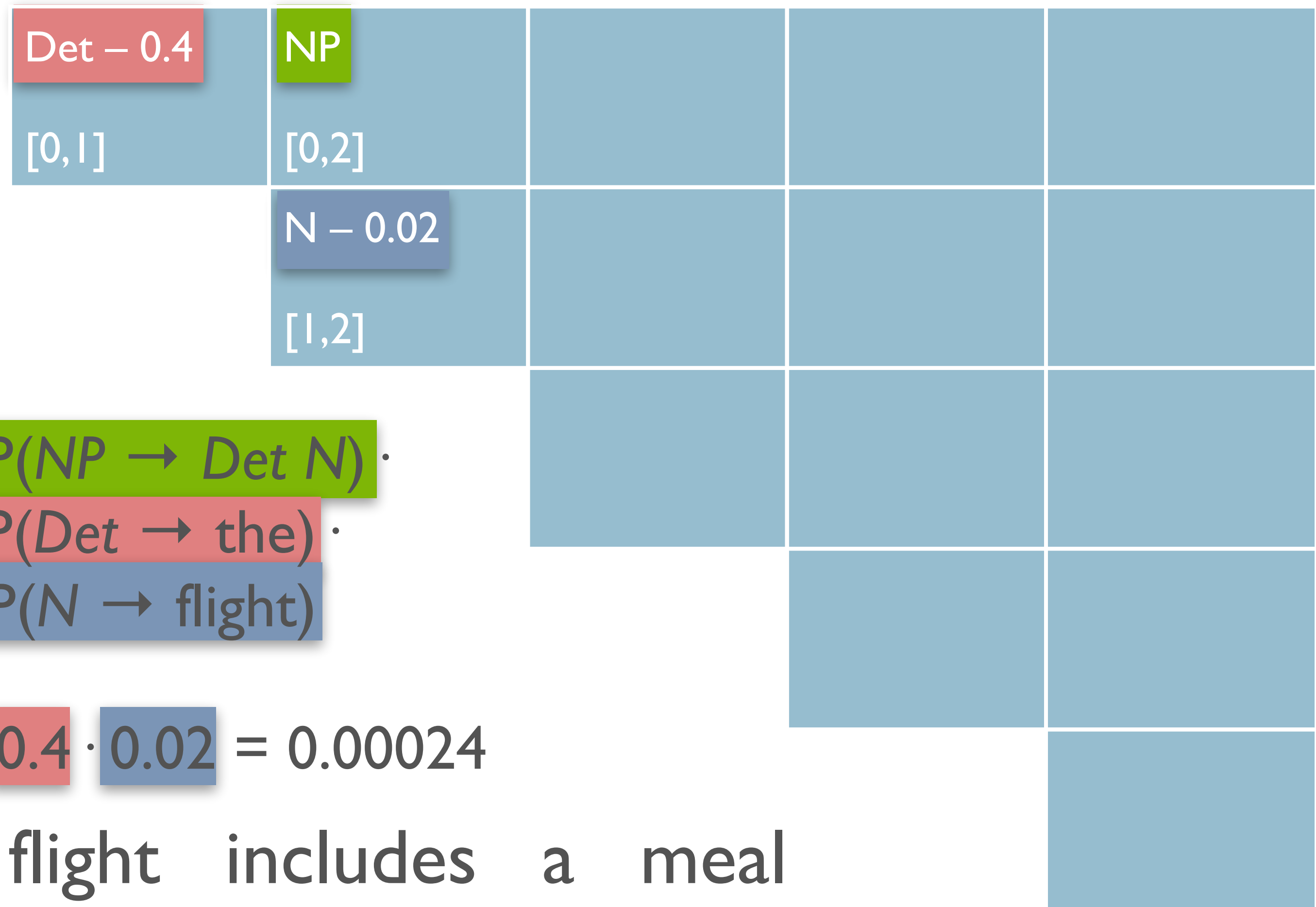
0      1      2      3      4      5  
 The   flight   includes   a   meal



# PCKY Matrix

$S \rightarrow NP VP$  [0.80]  
 $NP \rightarrow Det N$  [0.30]  
 $VP \rightarrow V NP$  [0.20]

$Det \rightarrow the$	[0.40]
$Det \rightarrow a$	[0.40]
$V \rightarrow includes$	[0.05]
$N \rightarrow meal$	[0.01]
$N \rightarrow flight$	[0.02]



The flight includes a meal

# PCKY Matrix

$S \rightarrow NP VP$  [0.80]  
 $NP \rightarrow Det N$  [0.30]  
 $VP \rightarrow V NP$  [0.20]

$Det \rightarrow the$  [0.40]  
 $Det \rightarrow a$  [0.40]  
 $V \rightarrow includes$  [0.05]  
 $N \rightarrow meal$  [0.01]  
 $N \rightarrow flight$  [0.02]

Det – 0.4	NP – 0.0024				
[0,1]	[0,2]				
	N – 0.02				
	[1,2]				

$$P = P(NP \rightarrow Det N) \cdot P(Det \rightarrow a) \cdot P(N \rightarrow flight)$$

$$P = 0.3 \cdot 0.4 \cdot 0.02 = 0.00024$$

0      1      2      3      4      5  
 The   flight   includes   a   meal

# PCKY Matrix

$S \rightarrow NP VP$  [0.80]  
 $NP \rightarrow Det N$  [0.30]  
 $VP \rightarrow V NP$  [0.20]

$Det \rightarrow the$  [0.40]  
 $Det \rightarrow a$  [0.40]  
 $V \rightarrow includes$  [0.05]  
 $N \rightarrow meal$  [0.01]  
 $N \rightarrow flight$  [0.02]

Det – 0.4 [0,1]	NP – 0.0024 [0,2]			S – 2.304×10 <sup>-8</sup> [0,5]
	N – 0.02 [1,2]			
		V – 0.05 [2,3]		VP – 1.2×10 <sup>-5</sup> [2,5]
			Det – 0.4 [3,4]	NP – 0.0012 [3,5]
				N – 0.01 [4,5]

0      1      2      3      4      5  
The flight includes a meal

# Inducing a PCFG

# Learning Probabilities

- Simplest way:
  - Use treebank of parsed sentences

# Learning Probabilities

- Simplest way:
  - Use treebank of parsed sentences
  - To compute probability of a rule, count:

# Learning Probabilities

- Simplest way:
  - Use treebank of parsed sentences
  - To compute probability of a rule, count:
    - Number of times a nonterminal is expanded:

$$\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)$$

# Learning Probabilities

- Simplest way:
  - Use treebank of parsed sentences
  - To compute probability of a rule, count:
    - Number of times a nonterminal is expanded:  $\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)$
    - Number of times a nonterminal is expanded by a given rule:  $\text{Count}(\alpha \rightarrow \beta)$



# Learning Probabilities

- Simplest way:
  - Use treebank of parsed sentences
  - To compute probability of a rule, count:
    - Number of times a nonterminal is expanded:  $\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)$
    - Number of times a nonterminal is expanded by a given rule:  $\text{Count}(\alpha \rightarrow \beta)$

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)} = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

# Learning Probabilities

- Simplest way:
  - Use treebank of parsed sentences
  - To compute probability of a rule, count:
    - Number of times a nonterminal is expanded:  $\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)$
    - Number of times a nonterminal is expanded by a given rule:  $\text{Count}(\alpha \rightarrow \beta)$

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)} = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

- Alternative: Learn probabilities by re-estimating
  - (Later)

# Probabilistic Parser Development Paradigm

	Train	Dev	Test
Size	Large  (eg. WSJ 2–21, 39,830 sentences)	Small  (e.g. WSJ 22)	Small/Med  (e.g. WSJ, 23, 2,416 sentences)
Usage	Estimate rule probabilities	Tuning/Verification, Check for Overfit	Held Out, Final Evaluation

# Parser Evaluation

# Parser Evaluation

- Assume a 'gold standard' set of parses for test set

# Parser Evaluation

- Assume a 'gold standard' set of parses for test set
- How can we tell how good the parser is?

# Parser Evaluation

- Assume a 'gold standard' set of parses for test set
- How can we tell how good the parser is?
- How can we tell how good a parse is?

# Parser Evaluation

- Assume a 'gold standard' set of parses for test set
- How can we tell how good the parser is?
- How can we tell how good a parse is?
  - Maximally strict: identical to 'gold standard'



# Parser Evaluation

- Assume a 'gold standard' set of parses for test set
- How can we tell how good the parser is?
- How can we tell how good a parse is?
  - Maximally strict: identical to 'gold standard'
  - Partial credit:

# Parser Evaluation

- Assume a 'gold standard' set of parses for test set
- How can we tell how good the parser is?
- How can we tell how good a parse is?
  - Maximally strict: identical to 'gold standard'
  - Partial credit:
    - Constituents in output match those in reference

# Parser Evaluation

- Assume a 'gold standard' set of parses for test set
- How can we tell how good the parser is?
- How can we tell how good a parse is?
  - Maximally strict: identical to 'gold standard'
  - Partial credit:
    - Constituents in output match those in reference
      - Same start point, end point, non-terminal symbol

# Parseval

- How can we compute parse score from constituents?
- Multiple Measures:

$$\text{Labeled Recall (LR)} = \frac{\# \text{ of } \mathbf{correct} \text{ constituents in } \mathbf{hypothetical} \text{ parse}}{\# \text{ of } \mathbf{total} \text{ constituents in } \mathbf{reference} \text{ parse}}$$

$$\text{Labeled Precision (LP)} = \frac{\# \text{ of } \mathbf{correct} \text{ constituents in } \mathbf{hypothetical} \text{ parse}}{\# \text{ of } \mathbf{total} \text{ constituents in } \mathbf{hypothetical} \text{ parse}}$$

# Parseval

- **F-measure:**

- Combines precision and recall

- Let  $\beta \in \mathbb{R}$  ,  $\beta > 0$  that adjusts  $P$  vs.  $R$  s.t.  $\beta \propto \frac{R}{P}$

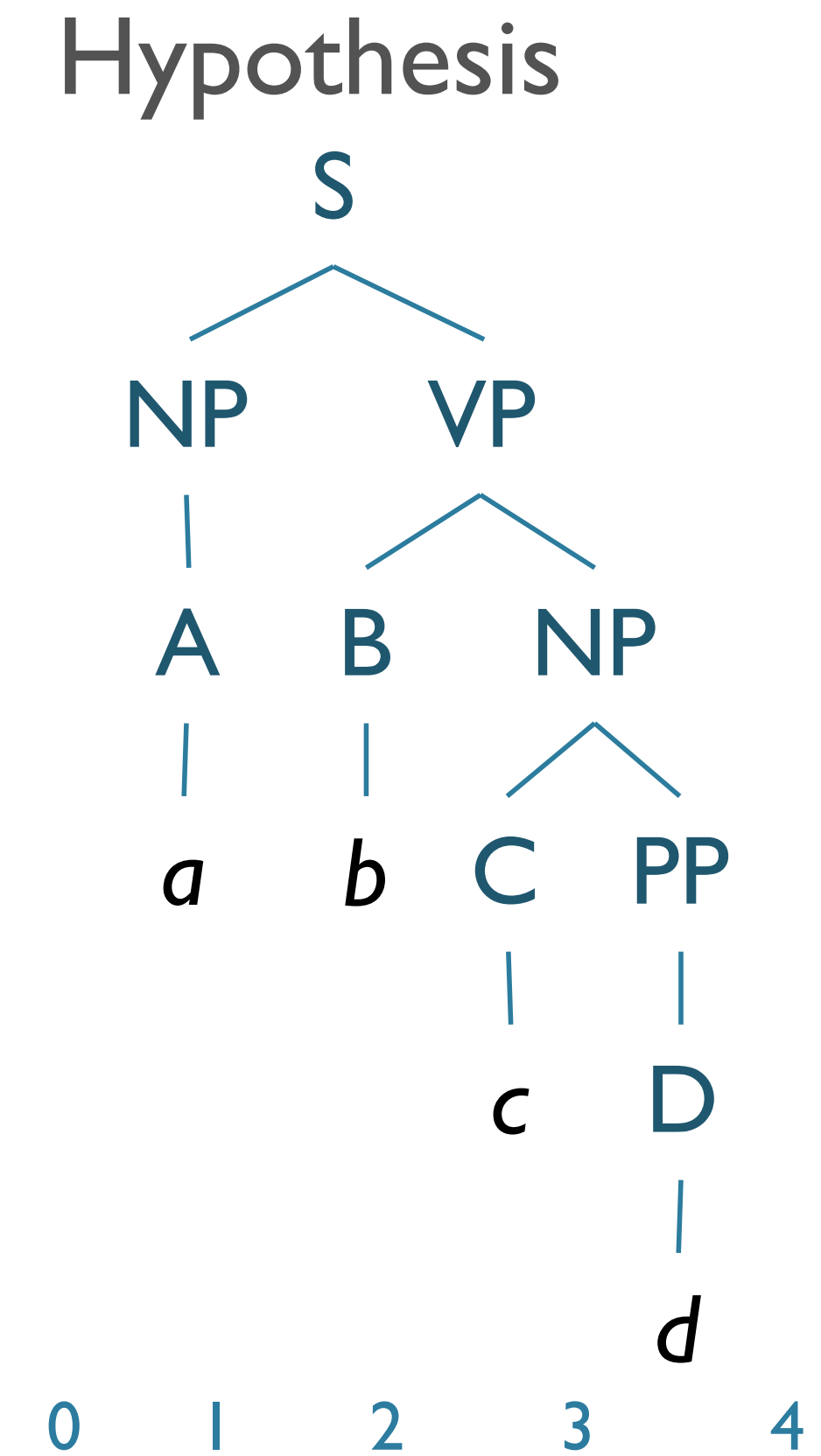
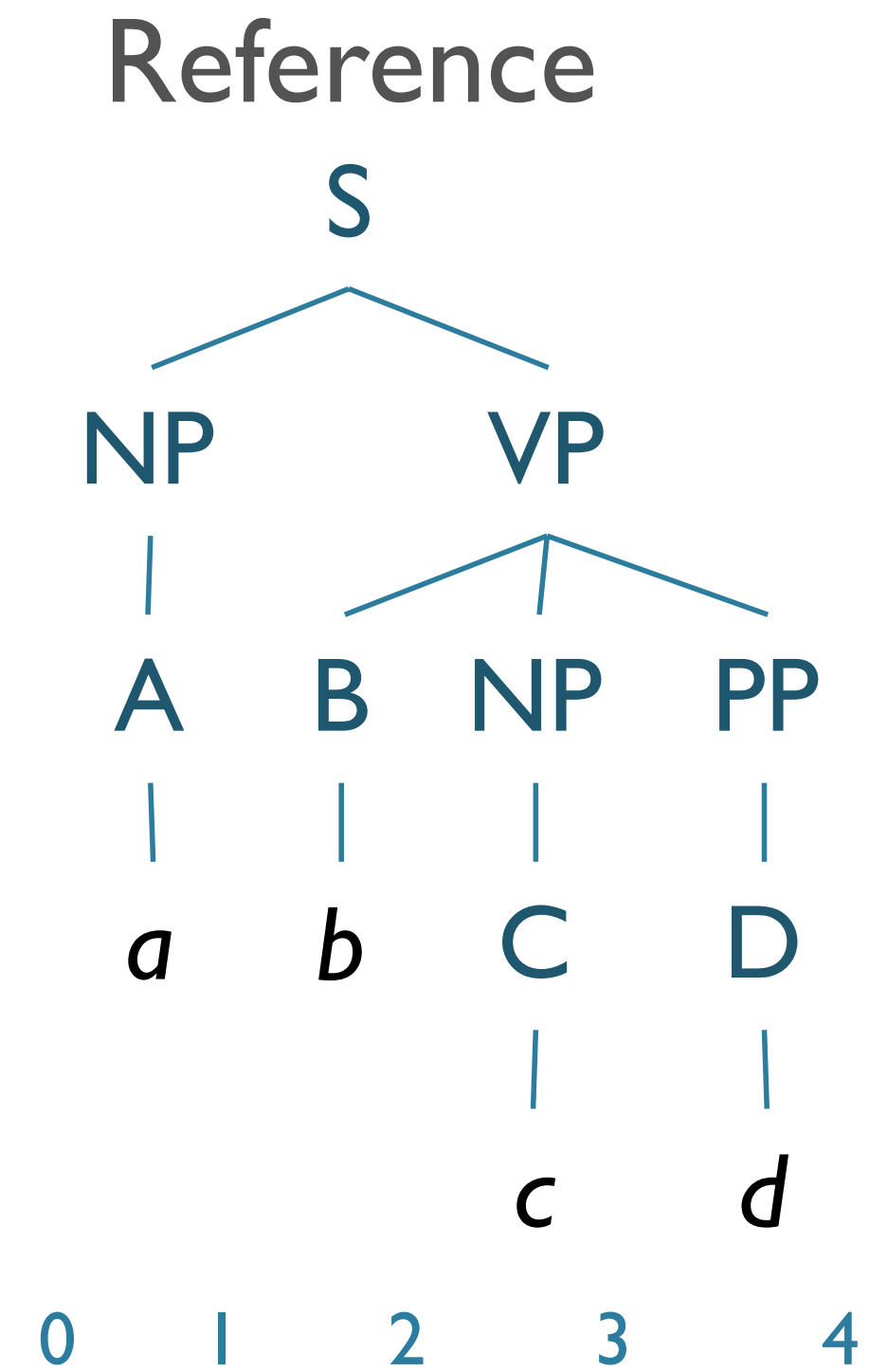
- $F_\beta$ -measure is then:

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{\beta^2 \cdot P + R}$$

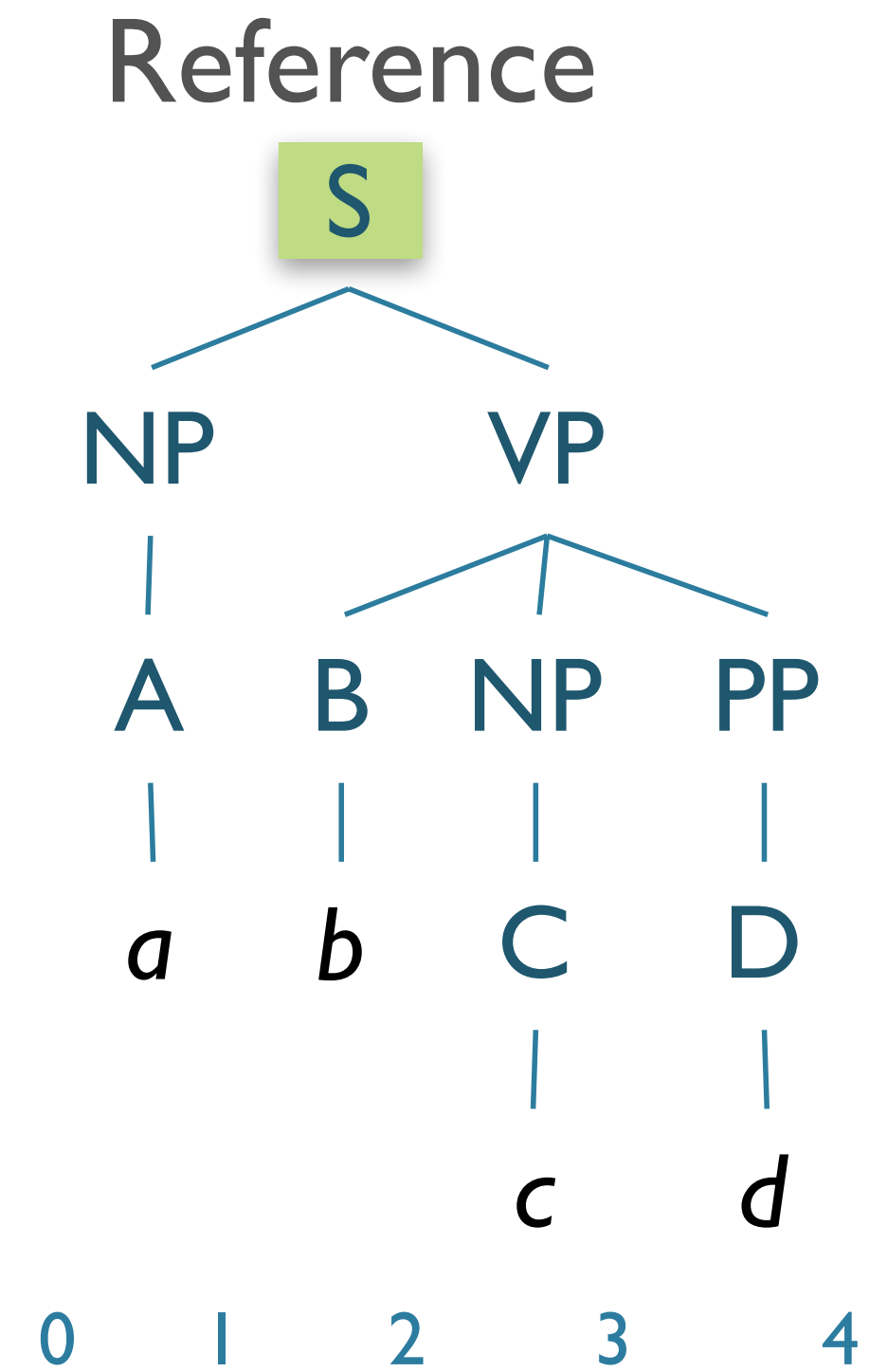
- With F1-measure as

$$F_1 = \frac{2PR}{P + R}$$

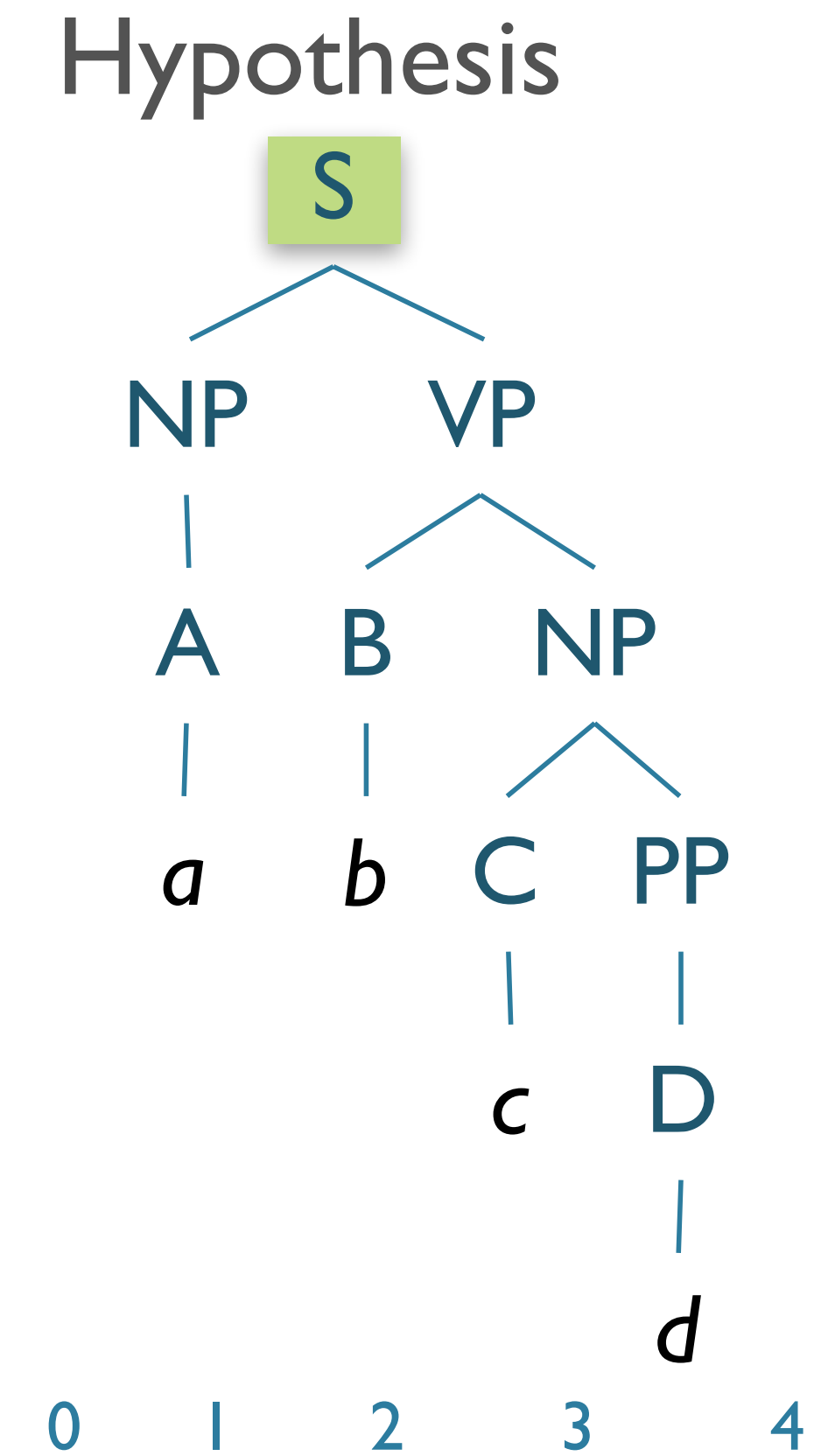
# Evaluation: Example



# Evaluation: Example

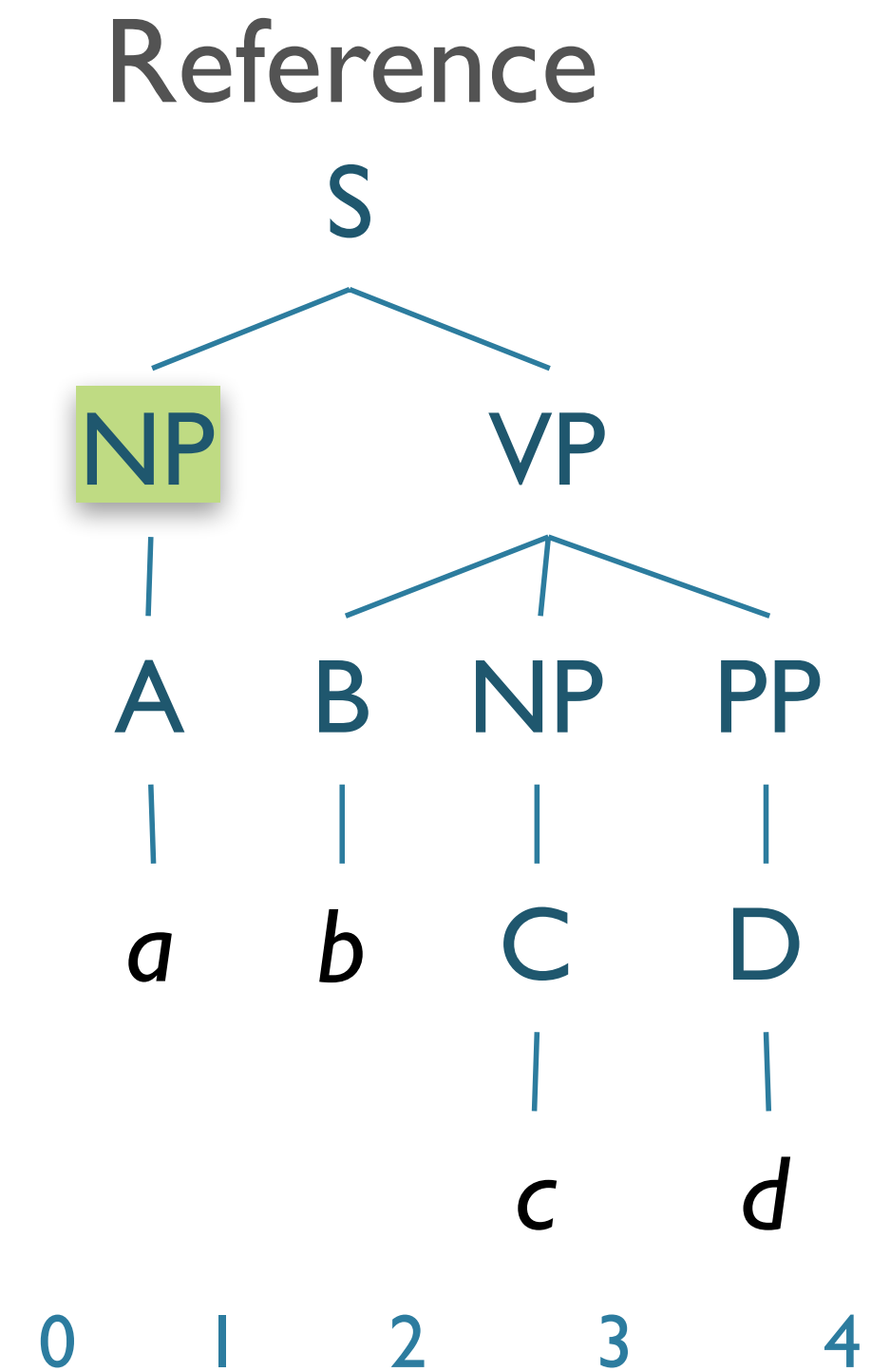


S(0,4)

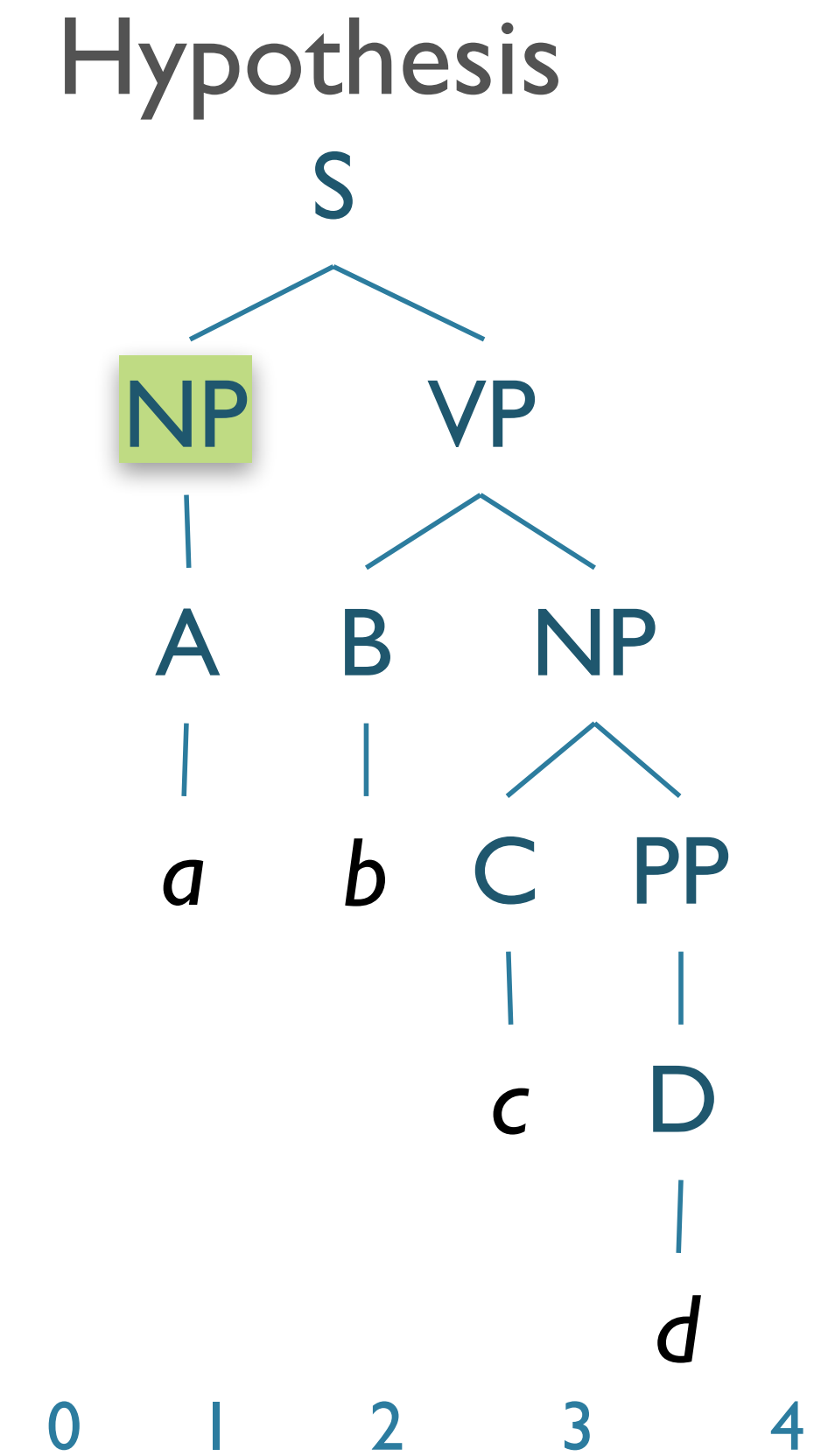


S(0,4)

# Evaluation: Example



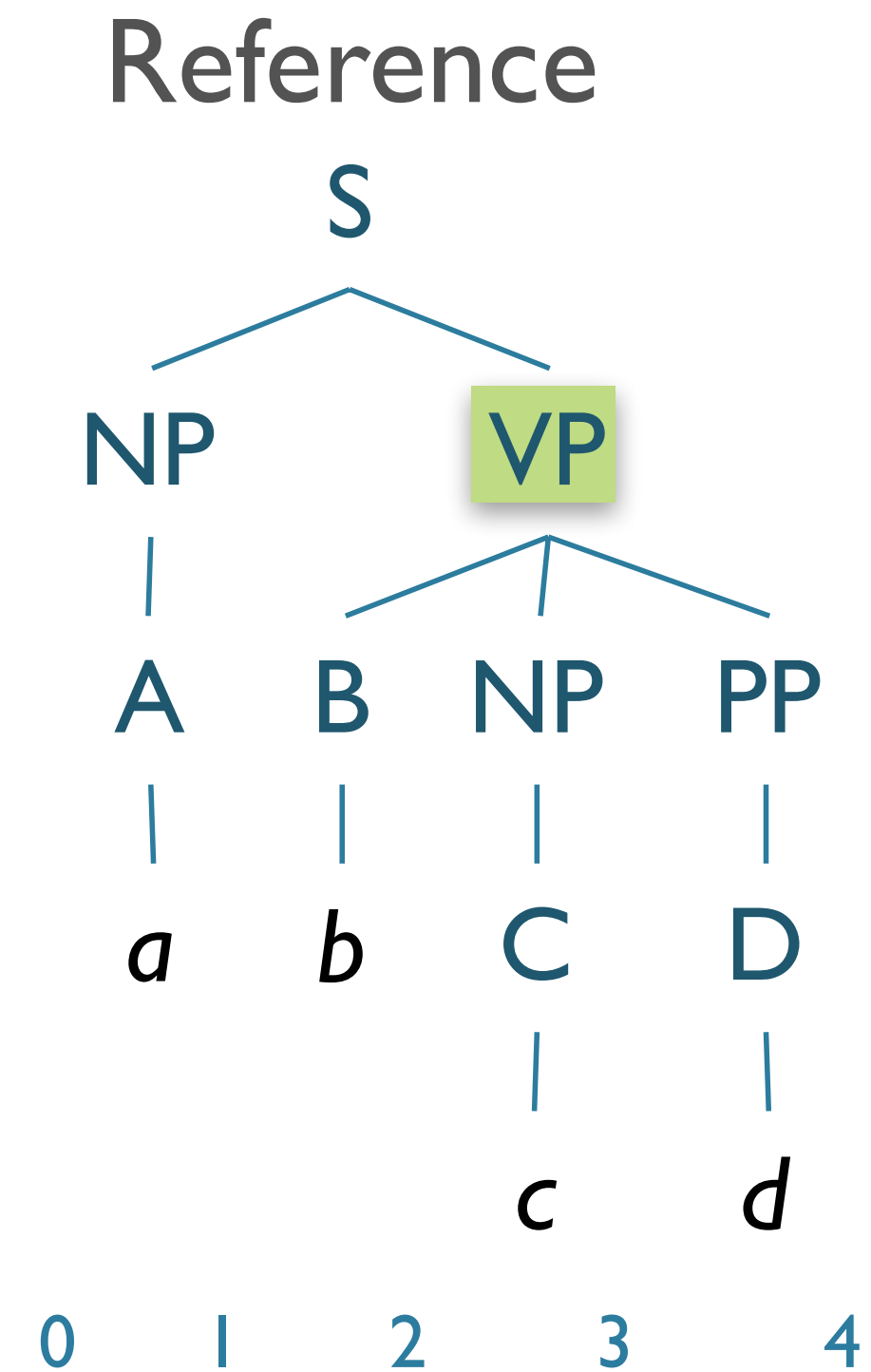
S(0,4)  
NP(0,1)



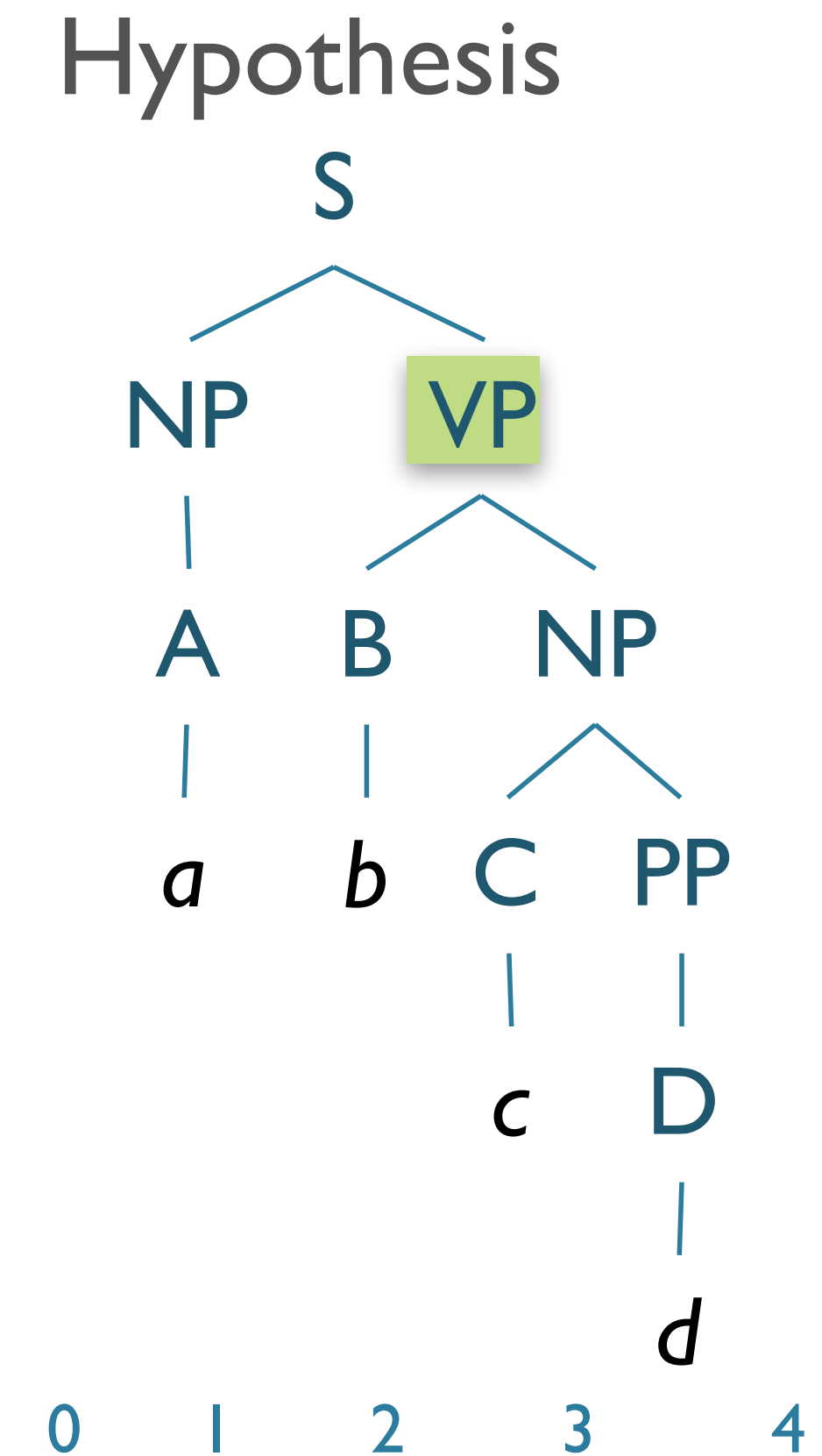
S(0,4)  
NP(0,1)



# Evaluation: Example

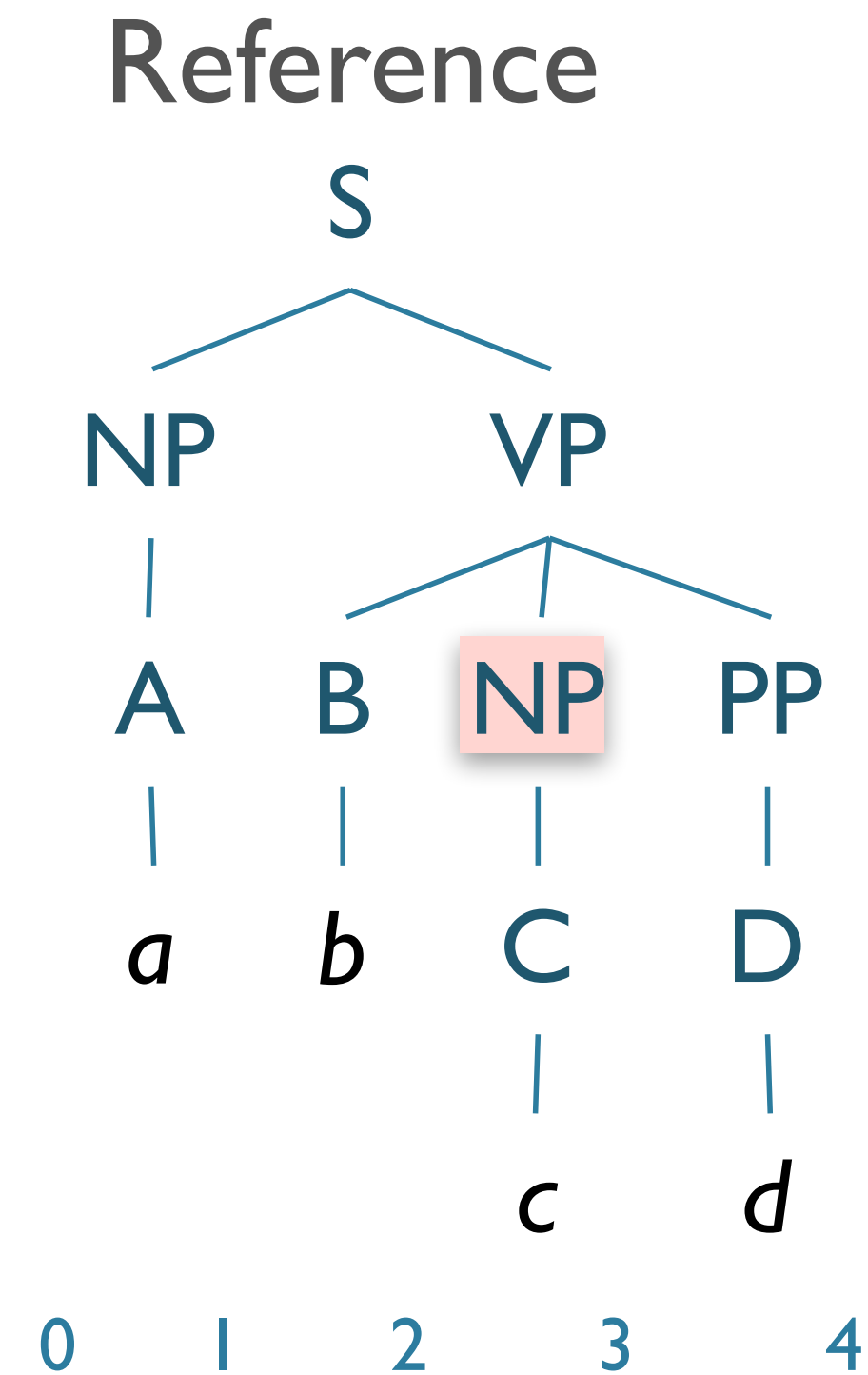


S(0,4)  
NP(0,1)  
VP(1,4)

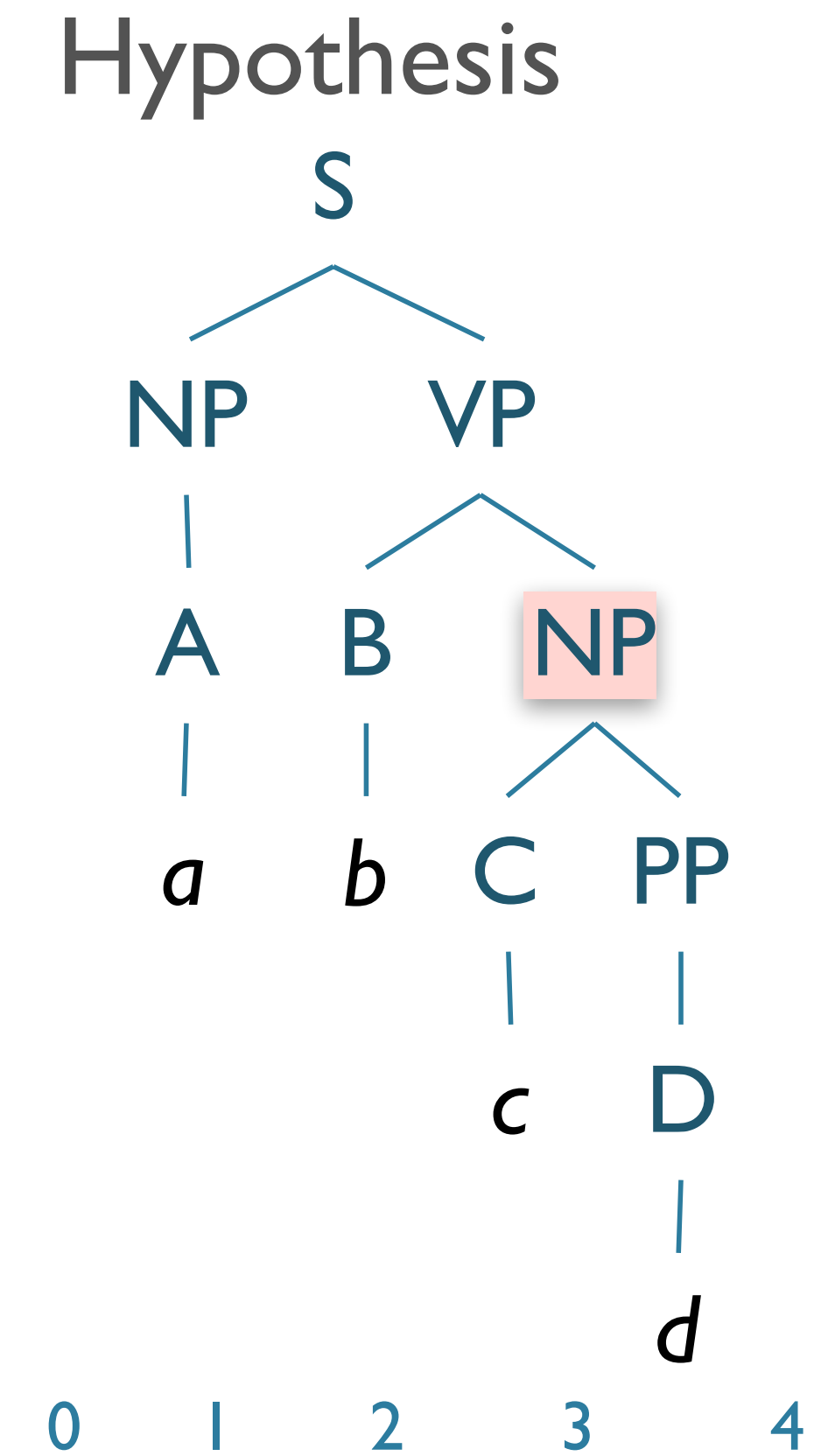


S(0,4)  
NP(0,1)  
VP(1,4)

# Evaluation: Example

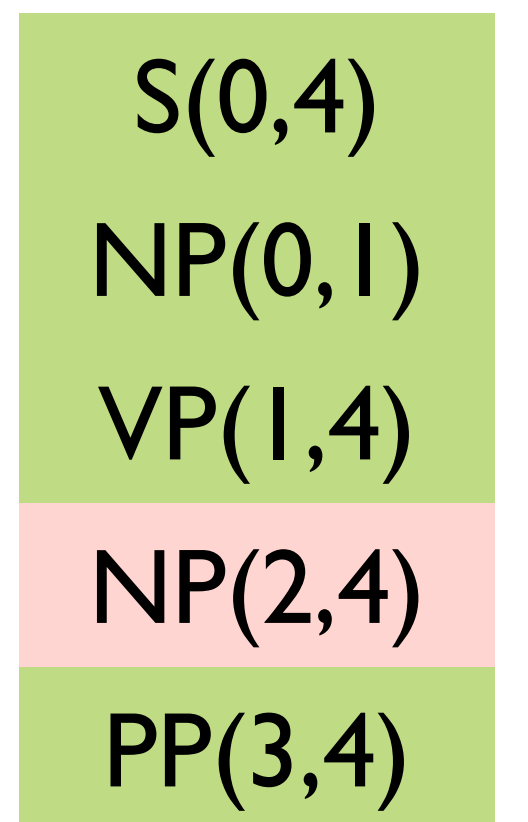
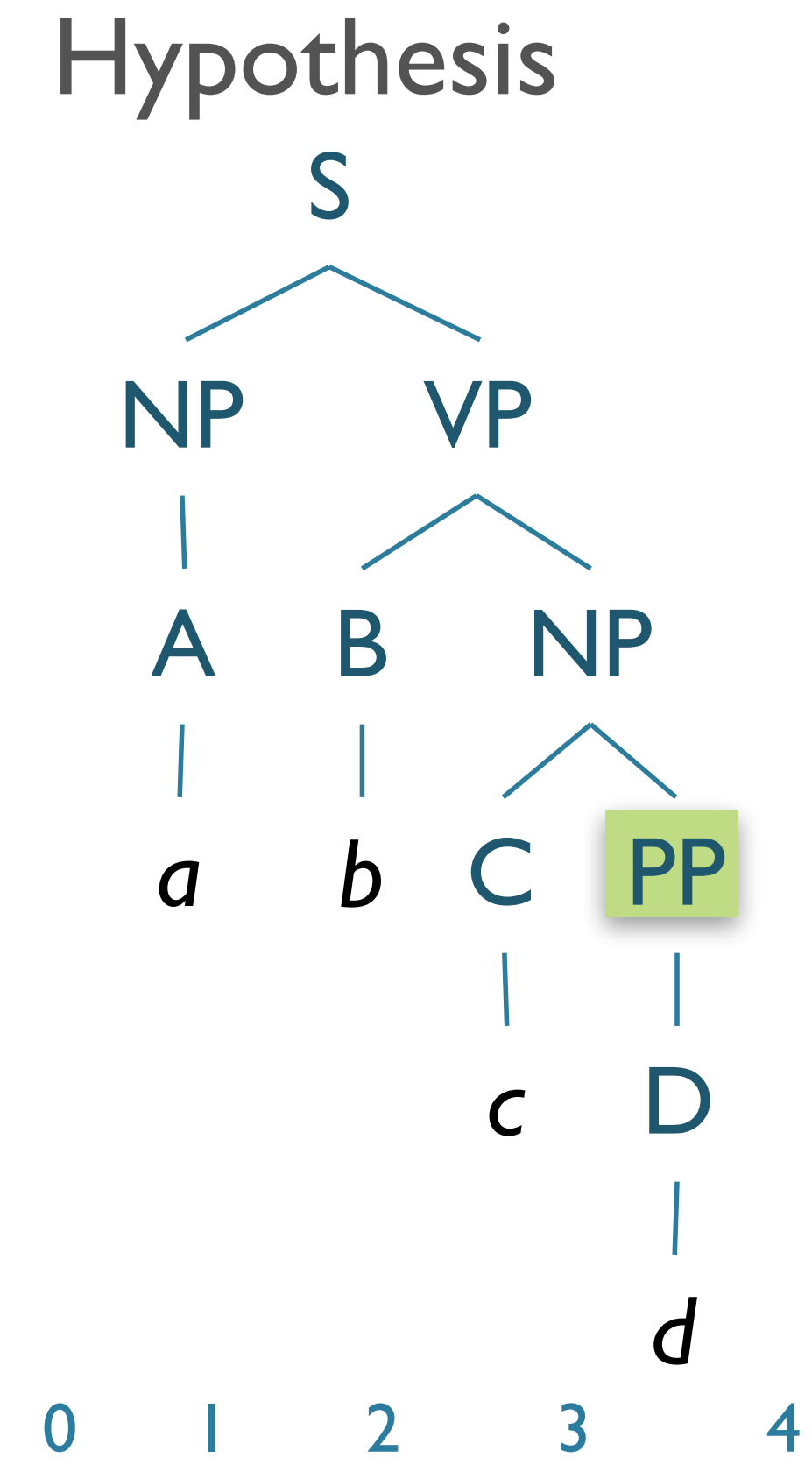
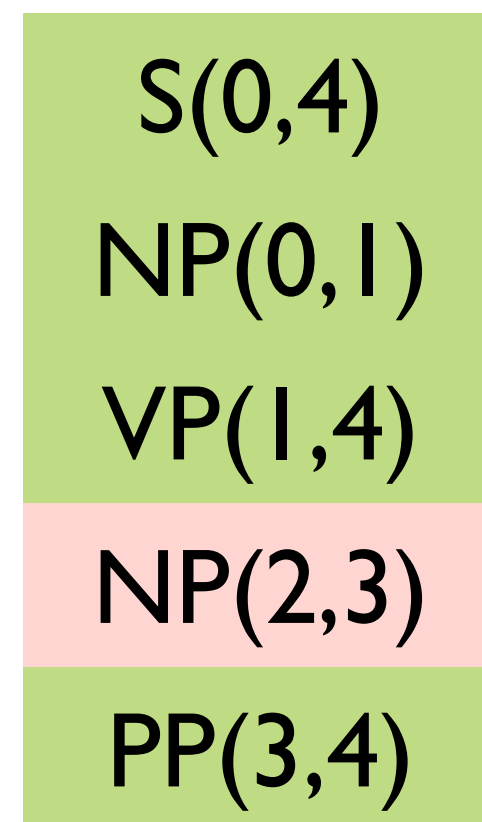
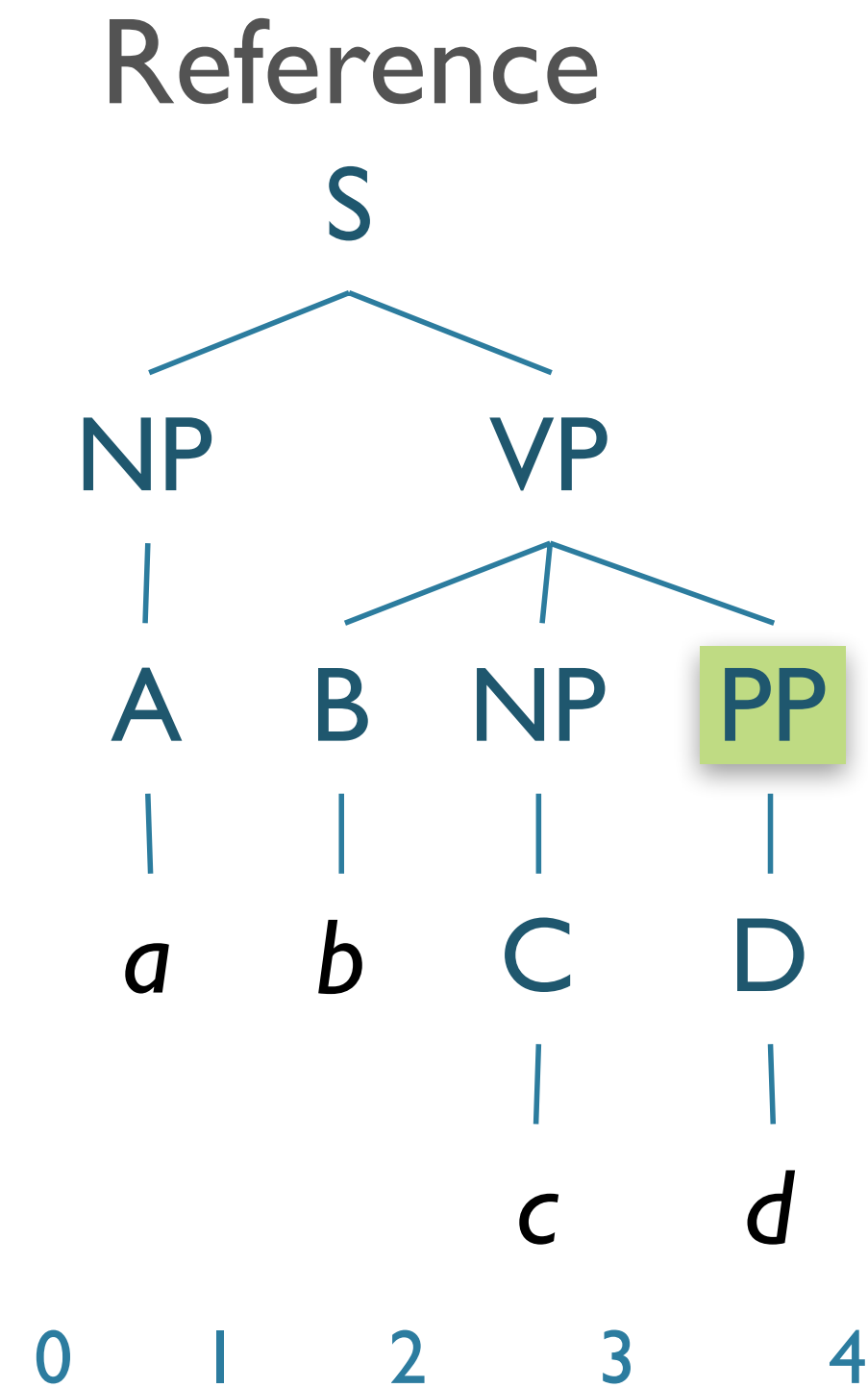


S(0,4)  
NP(0,1)  
VP(1,4)  
NP(2,3)

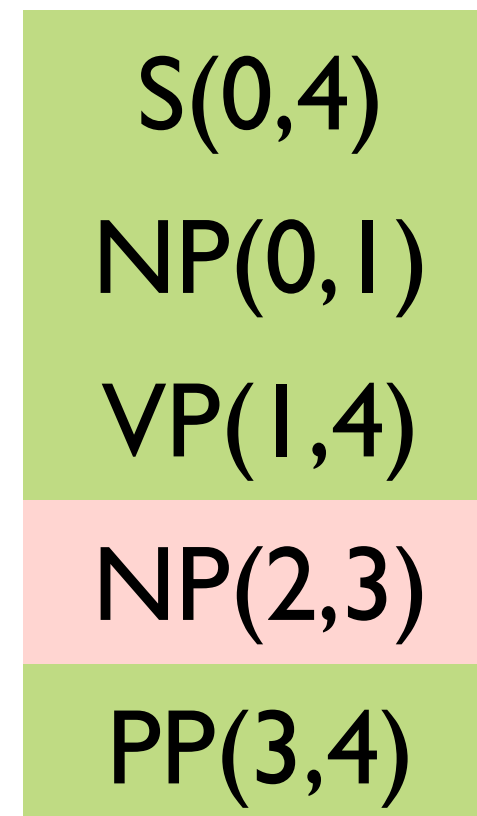
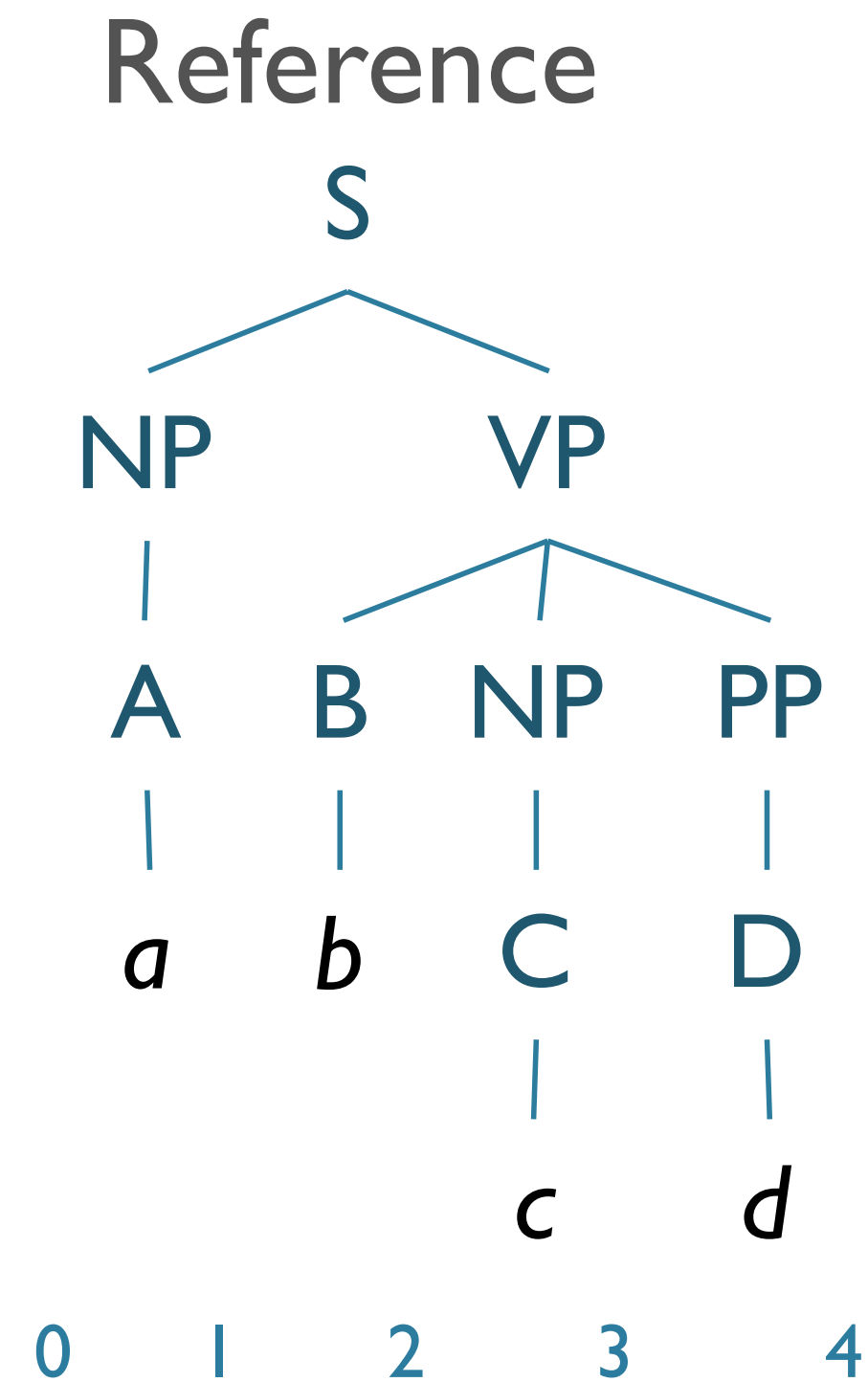


S(0,4)  
NP(0,1)  
VP(1,4)  
NP(2,4)

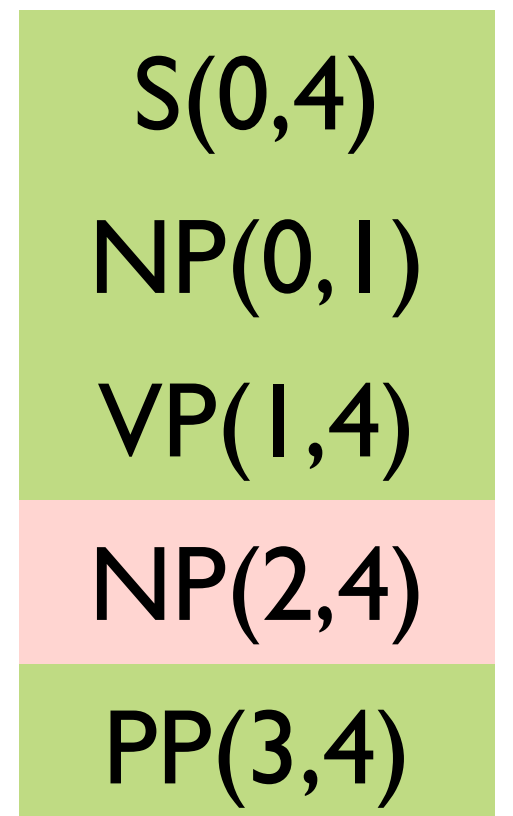
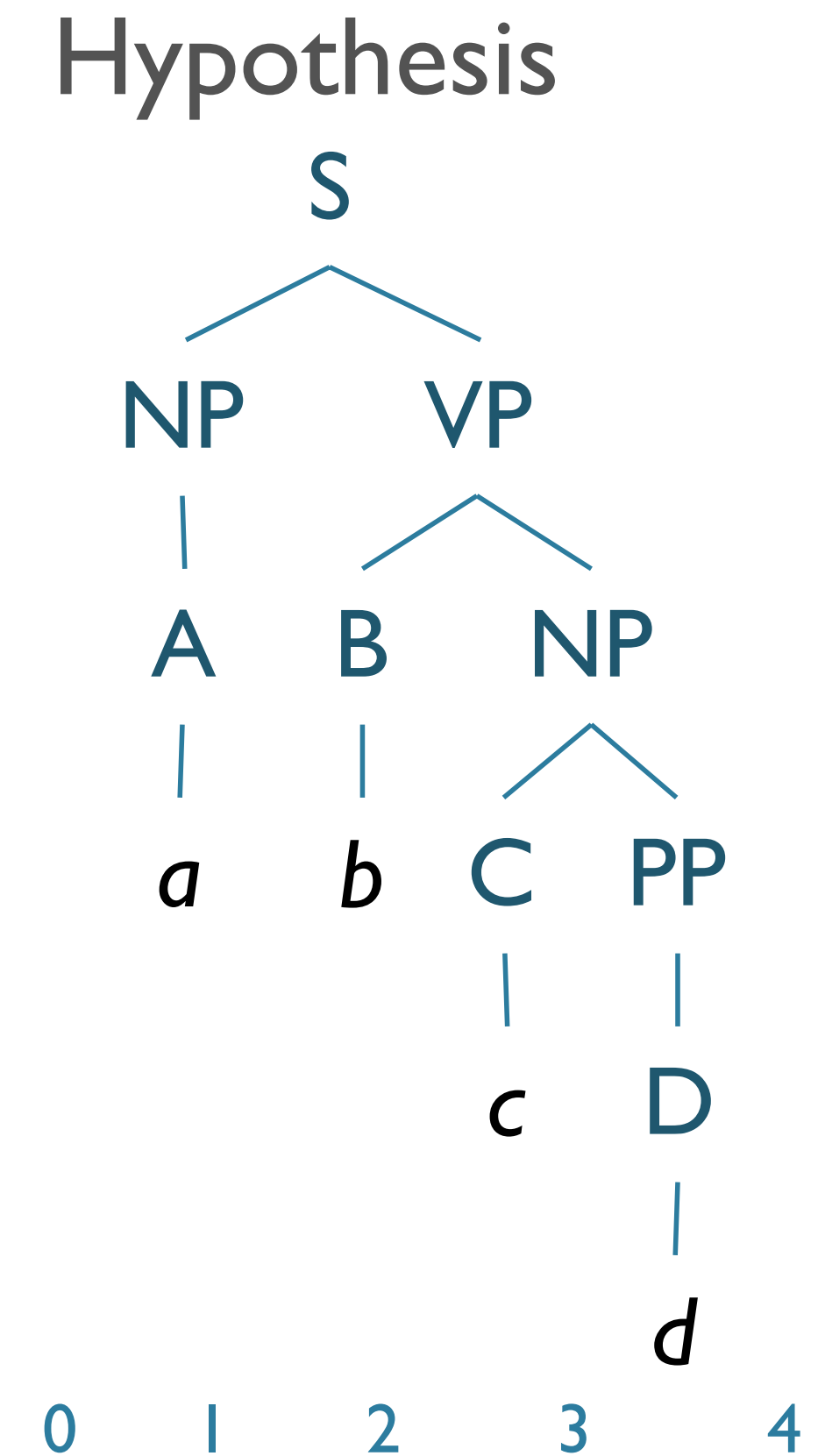
# Evaluation: Example



# Evaluation: Example

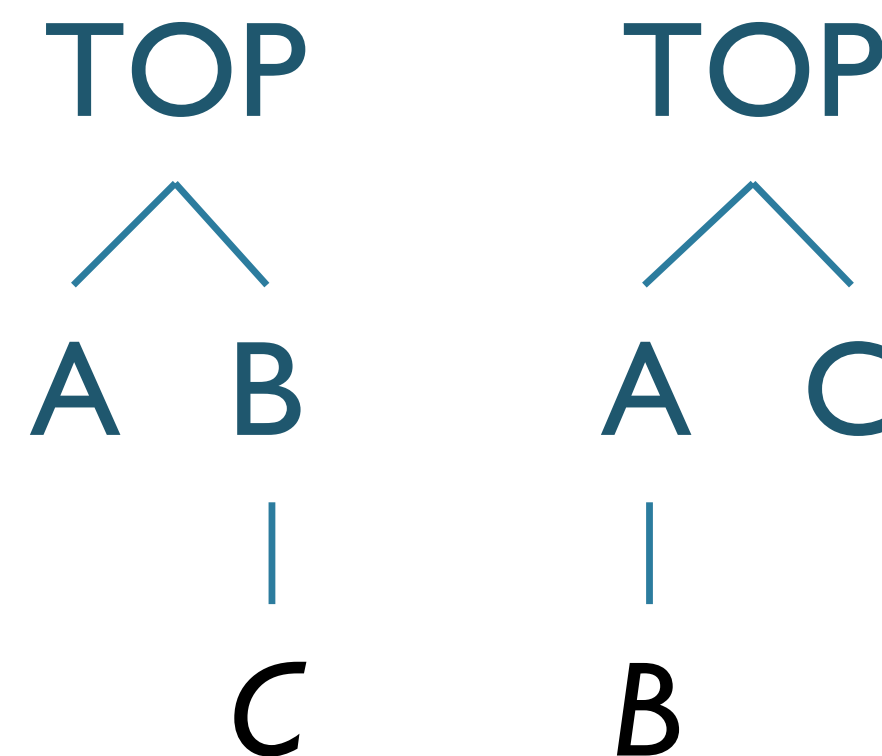


LP:	4/5
LR:	4/5
F <sub>1</sub> :	4/5



# Parser Evaluation

- Crossing Brackets:
  - # of constituents where produced parse has bracketings that overlap for the siblings:
  - ((A B) C) — { (0,2), (2,3) }  
and hyp. has  
(A (B C)) — { (0,1), (1, 3) }



```
/* crossing is counted based on the brackets */
/* in test rather than gold file (by Mike) */
for(j=0;j<bn2;j++){
  for(i=0;i<bn1;i++){
    if(bracket1[i].result != 5 &&
       bracket2[j].result != 5 &&
       ((bracket1[i].start < bracket2[j].start &&
          bracket1[i].end > bracket2[j].start &&
          bracket1[i].end < bracket2[j].end) ||
        (bracket1[i].start > bracket2[j].start &&
          bracket1[i].start < bracket2[j].end &&
          bracket1[i].end > bracket2[j].end))){
```

from evalb.c

# State-of-the-Art Parsing

- Parsers trained/tested on Wall Street Journal PTB
  - LR: 94%+;
  - LP: 94%+;
  - Crossing brackets: 1%
- Standard implementation of Parseval:
  - **evalb**

# Evaluation Issues

- Only evaluating constituency
- There are other grammar formalisms:
  - LFG (Constraint-based)
  - Dependency Structure
- **Extrinsic** evaluation
  - How well does getting the correct parse match the semantics, etc?

# Earley Parsing



# Earley vs. CKY

- CKY doesn't capture full original structure
  - Can back-convert binarization, terminal conversion
  - Unit non-terminals require change in CKY

# Earley vs. CKY

- CKY doesn't capture full original structure
  - Can back-convert binarization, terminal conversion
  - Unit non-terminals require change in CKY
- Earley algorithm
  - Supports parsing efficiently with arbitrary grammars
  - Top-down search
  - Dynamic programming
    - Tabulated partial solutions
  - Some bottom-up constraints

# Earley Algorithm

- Another dynamic programming solution
  - Partial parses stored in “chart”
  - Compactly encodes ambiguity
  - $O(N^3)$
- Chart entries contain:
  - Subtree for a single grammar rule
  - Progress in completing subtree
  - Position of subtree w.r.t. input

# Earley Algorithm

- First, left-to-right pass fills out a chart with  $N+1$  states
  - Chart entries — sit between words in the input string
  - Keep track of states of the parse at those positions
  - For each word position, chart contains set of states representing all partial parse trees generated so far
  - e.g. `chart[0]` contains all partial parse trees generated at the beginning of sentence

# Chart Entries

- Three types of constituents:
  - Predicted constituents
  - In-progress constituents
  - Completed constituents

# Parse Progress

- Represented by Dotted Rules
  - Position of • indicates type of constituent
- $_0$  Book  $_1$  that  $_2$  flight  $_3$ 
  - $S \rightarrow \cdot VP$  [0,0] (predicted)
  - $NP \rightarrow Det \cdot Nom$  [1,2] (in progress)
  - $VP \rightarrow V NP \cdot$  [0,3] (completed)
- [x,y] tells us what portion of the input is spanned so far by rule
- Each state  $s_i$ :  $\langle dotted\ rule \rangle, [\langle back\ pointer \rangle, \langle current\ position \rangle]$

0 Book<sub>1</sub> that<sub>2</sub> flight<sub>3</sub>

- $S \rightarrow \cdot VP, [0,0]$ 
  - First 0 means S constituent begins at the start of input
  - Second 0 means the dot is here too
  - So, this is a top-down prediction

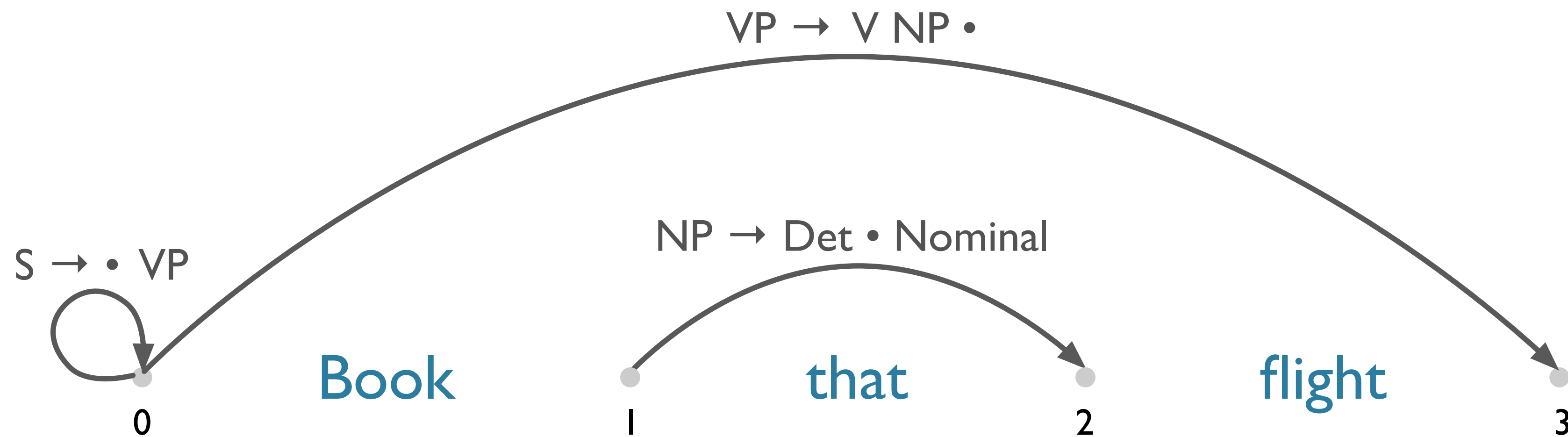
## 0 Book<sub>1</sub> that<sub>2</sub> flight<sub>3</sub>

- $S \rightarrow \cdot VP, [0,0]$ 
  - First 0 means S constituent begins at the start of input
  - Second 0 means the dot is here too
  - So, this is a top-down prediction
- $NP \rightarrow Det \cdot Nom, [1,2]$ 
  - the NP begins at position 1
  - the dot is at position 2
  - so, Det has been successfully parsed
  - Nom predicted next



# 0 Book<sub>1</sub> that<sub>2</sub> flight<sub>3</sub> (continued)

- $VP \rightarrow V NP \cdot [0,3]$
- Successful VP parse of entire input



# Successful Parse

- Final answer found by looking at last entry in chart
- If entry resembles  $S \rightarrow a \cdot [0,N]$  then input parsed successfully
- Chart will also contain record of all possible parses of input string, given the grammar

# Parsing Procedure for the Earley Algorithm

- Move through each set of states in order, applying one of three operations:
  - **predictor**: add predictions to the chart
  - **scanner**: read input and add corresponding state to chart
  - **completer**: move dot to right when new constituent found
- Results (new states) added to current or next set of states in chart
- No backtracking and no states removed: keep complete history of parse

# Earley Algorithm

```
function EARLEY-PARSE(words, grammar) returns chart
  ENQUEUE( $(\gamma \rightarrow \bullet S, [0,0])$ , chart[0])
  for  $i \leftarrow$  from 0 to LENGTH(words) do
    for each state in chart[i] do
      if INCOMPLETE?(state) and
        NEXT-CAT(state) is not a part of speech then
        PREDICTOR(state)
      elseif INCOMPLETE?(state) and
        NEXT-CAT(state) is a part of speech then
        SCANNER(state)
      else
        COMPLETER(state)
      end
    end
  return(chart)
```

# Earley Algorithm

```
procedure PREDICTOR( $(A \rightarrow \alpha \cdot B \beta, [i, j])$ )  
  for each  $(B \rightarrow \gamma)$  in GRAMMAR-RULES-FOR( $B, grammar$ ) do  
    ENQUEUE( $(B \rightarrow \cdot \gamma, [j, j])$ ,  $chart[j]$ )  
  end
```

```
procedure SCANNER( $(A \rightarrow \alpha \cdot B \beta, [i, j])$ )  
  if  $B \in \text{PARTS-OF-SPEECH}(word[j])$  then  
    ENQUEUE( $(B \rightarrow word[j] \cdot, [j, j+1])$ ,  $chart[j+1]$ )
```

```
procedure COMPLETER( $(B \rightarrow \gamma \cdot, [j, k])$ )  
  for each  $(A \rightarrow \alpha \cdot B \beta, [i, j])$  in  $chart[j]$  do  
    ENQUEUE( $(A \rightarrow \alpha B \cdot \beta, [i, k])$ ,  $chart[k]$ )  
  end
```

# 3 Main Subroutines of Earley

- Predictor
  - Adds predictions into the chart
- Scanner
  - Reads the input words and enters states representing those words into the chart
- Completer
  - Moves the dot to the right when new constituents are found

# Predictor

- Intuition:
  - Create new state for top-down prediction of new phrase
- Applied when non part-of-speech non-terminals are to the right of a dot:
  - $S \rightarrow \cdot VP$  [0,0]
- Adds new states to *current* chart
  - One new state for each expansion of the non-terminal in the grammar
    - $VP \rightarrow \cdot V$  [0,0]
    - $VP \rightarrow \cdot V NP$  [0,0]

# Chart[0]

S0	$\gamma \rightarrow \cdot S$	[0,0]	Dummy start state
S1	$S \rightarrow \cdot NP VP$	[0,0]	Predictor
S2	$S \rightarrow \cdot Aux NP VP$	[0,0]	Predictor
S3	$S \rightarrow \cdot VP$	[0,0]	Predictor
S4	$NP \rightarrow \cdot Pronoun$	[0,0]	Predictor
S5	$NP \rightarrow \cdot Proper-Noun$	[0,0]	Predictor
S6	$NP \rightarrow \cdot Det Nominal$	[0,0]	Predictor
S7	$VP \rightarrow \cdot Verb$	[0,0]	Predictor
S8	$VP \rightarrow \cdot Verb NP$	[0,0]	Predictor
S9	$VP \rightarrow \cdot Verb NP PP$	[0,0]	Predictor
S10	$VP \rightarrow \cdot Verb PP$	[0,0]	Predictor
S11	$VP \rightarrow \cdot VP PP$	[0,0]	Predictor



# Chart[1]

S12	$Verb \rightarrow book \cdot$	[0,1]	Scanner
S13	$VP \rightarrow Verb \cdot$	[0,1]	Completer
S14	$VP \rightarrow Verb \cdot NP$	[0,1]	Completer
S15	$VP \rightarrow Verb \cdot NP PP$	[0,1]	Completer
S16	$VP \rightarrow Verb \cdot PP$	[0,1]	Completer
S17	$S \rightarrow VP \cdot$	[0,1]	Completer
S18	$VP \rightarrow VP \cdot PP$	[0,1]	Completer
S19	$NP \rightarrow \cdot Pronoun$	[1,1]	Predictor
S20	$NP \rightarrow \cdot Proper-Noun$	[1,1]	Predictor
S21	$NP \rightarrow \cdot Det Nominal$	[1,1]	Predictor
S22	$PP \rightarrow \cdot Prep NP$	[1,1]	Predictor

# *Book that flight*

S0:  $\gamma \rightarrow \bullet S [0,0]$

$\gamma$   
|  
 $\bullet S$

# *Book that flight*

S0:  $\gamma \rightarrow \bullet S$  [0,0]

S3:  $S \rightarrow \bullet VP$  [0,0]

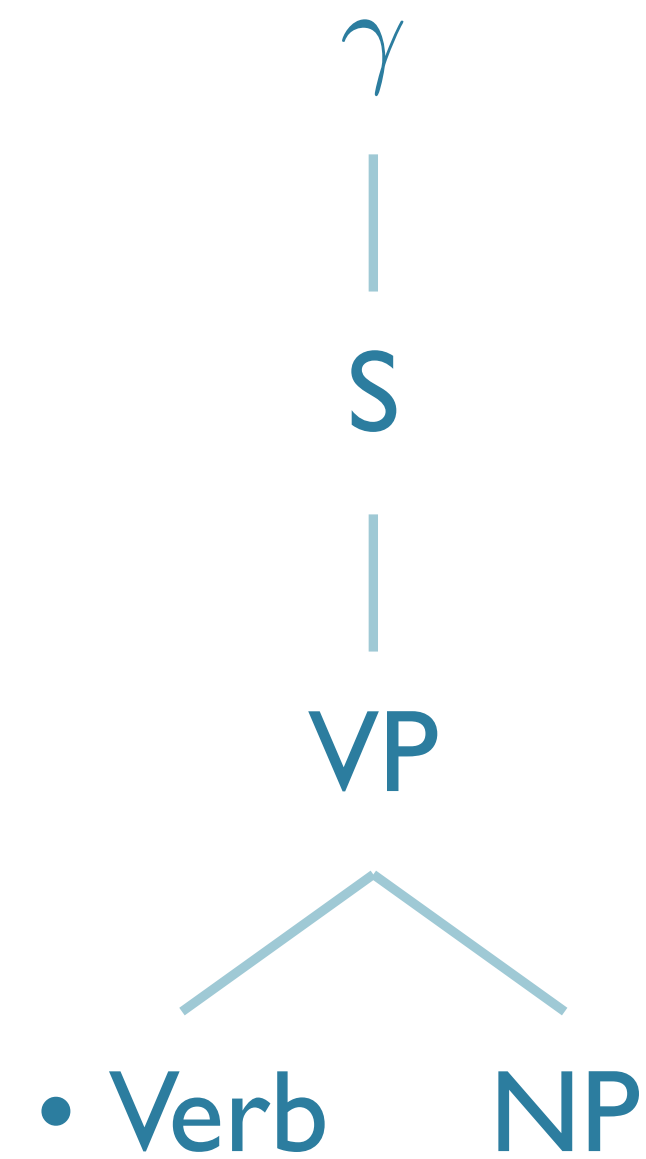
$\gamma$   
|  
S  
|  
 $\bullet VP$

# Book that flight

S0:  $\gamma \rightarrow \bullet S$  [0,0]

S3:  $S \rightarrow \bullet VP$  [0,0]

S8:  $VP \rightarrow \bullet Verb\ NP$  [0,0]



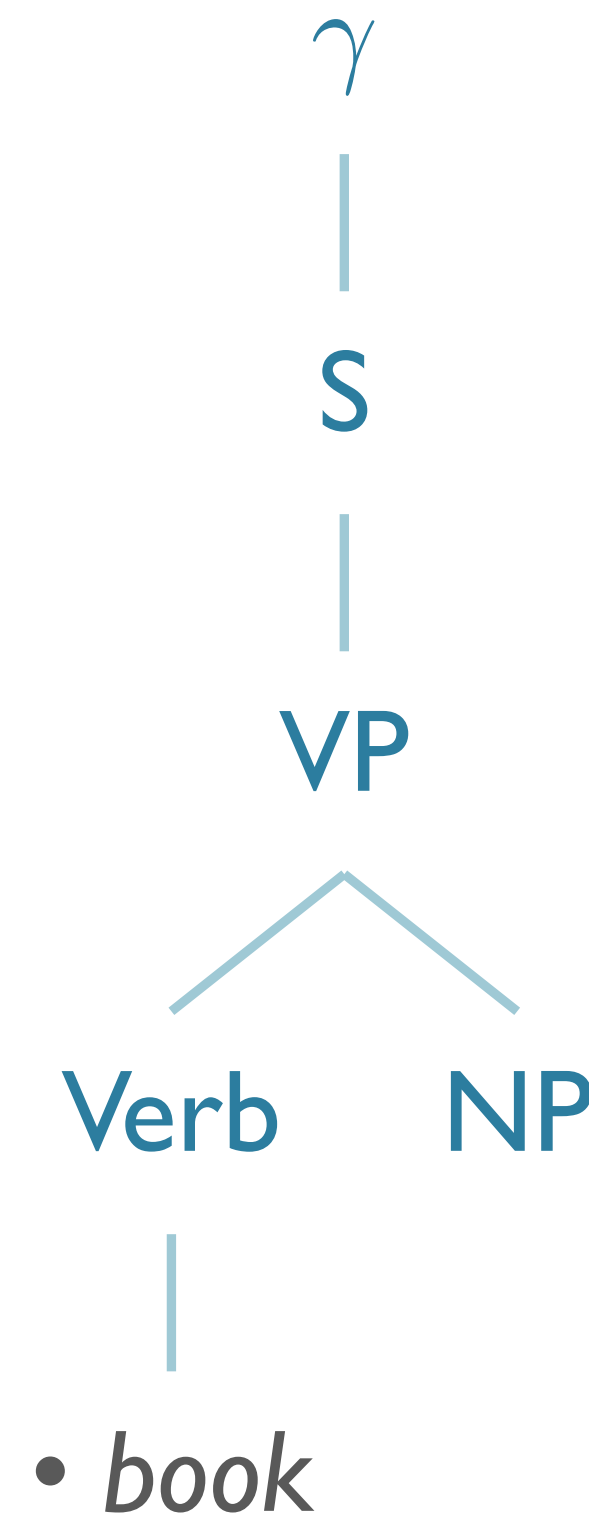
# Book that flight

S0:  $\gamma \rightarrow \bullet S$  [0,0]

S3:  $S \rightarrow \bullet VP$  [0,0]

S8:  $VP \rightarrow \bullet Verb NP$  [0,0]

S12:  $Verb \rightarrow \bullet book$  [0,0]



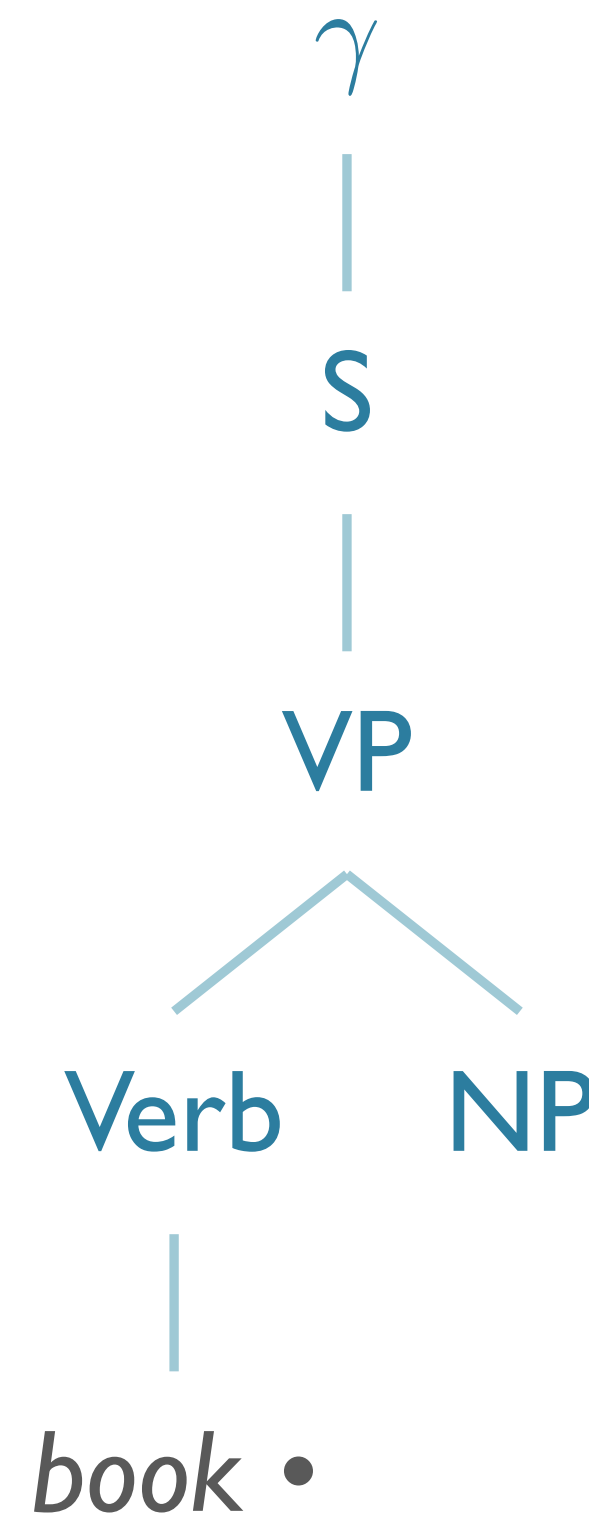
# Book that flight

S0:  $\gamma \rightarrow \cdot S$  [0,0]

S3:  $S \rightarrow \cdot VP$  [0,0]

S8:  $VP \rightarrow \cdot Verb\ NP$  [0,0]

S12:  $Verb \rightarrow book \cdot$  [0,1]

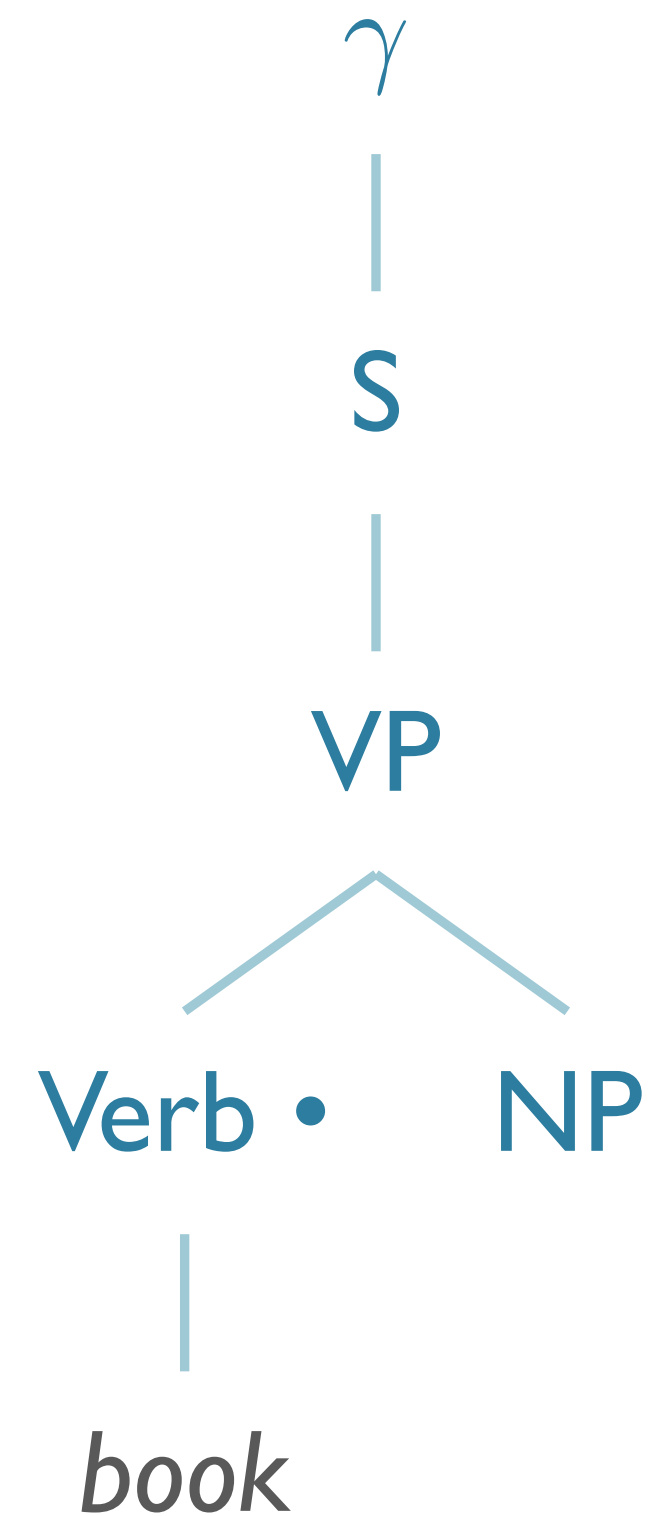


# *Book that flight*

S0:  $\gamma \rightarrow \cdot S$  [0,0]

S3:  $S \rightarrow \cdot VP$  [0,0]

S8:  $VP \rightarrow Verb \cdot NP$  [0,1]

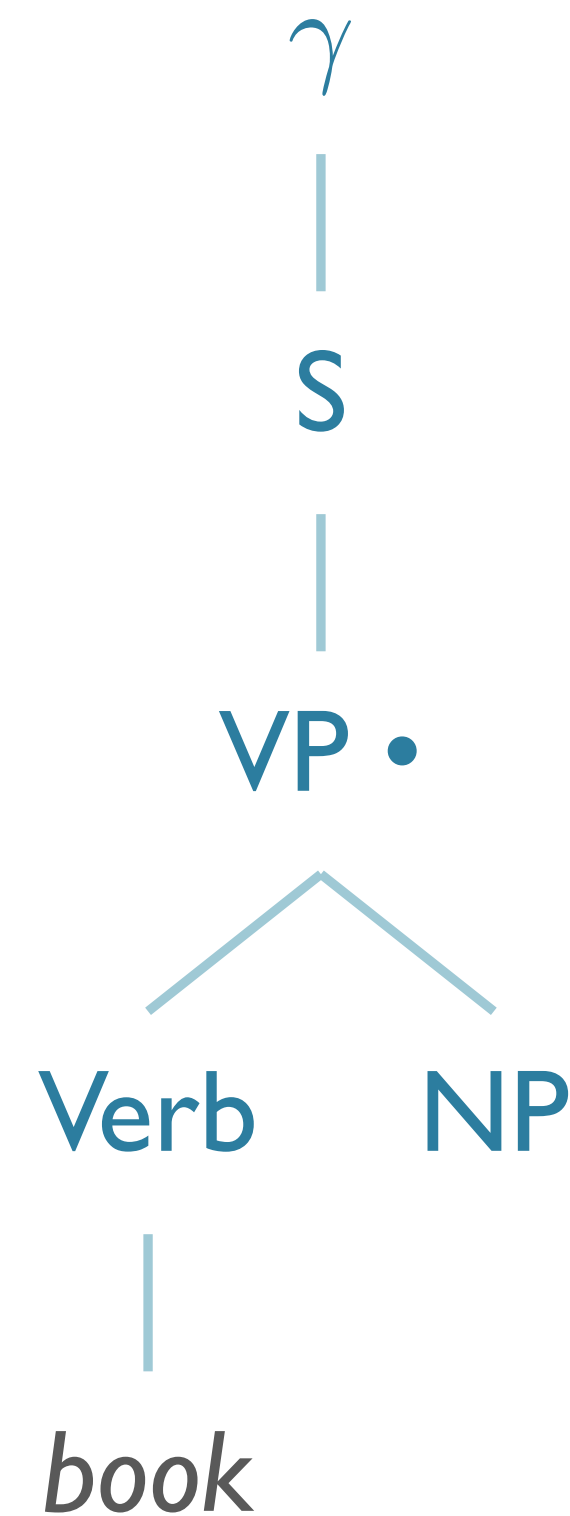


# Book that flight

S0:  $\gamma \rightarrow \cdot S$  [0,0]

S3:  $S \rightarrow VP \cdot$  [0,1]

S8:  $VP \rightarrow Verb \cdot NP$  [0,1]





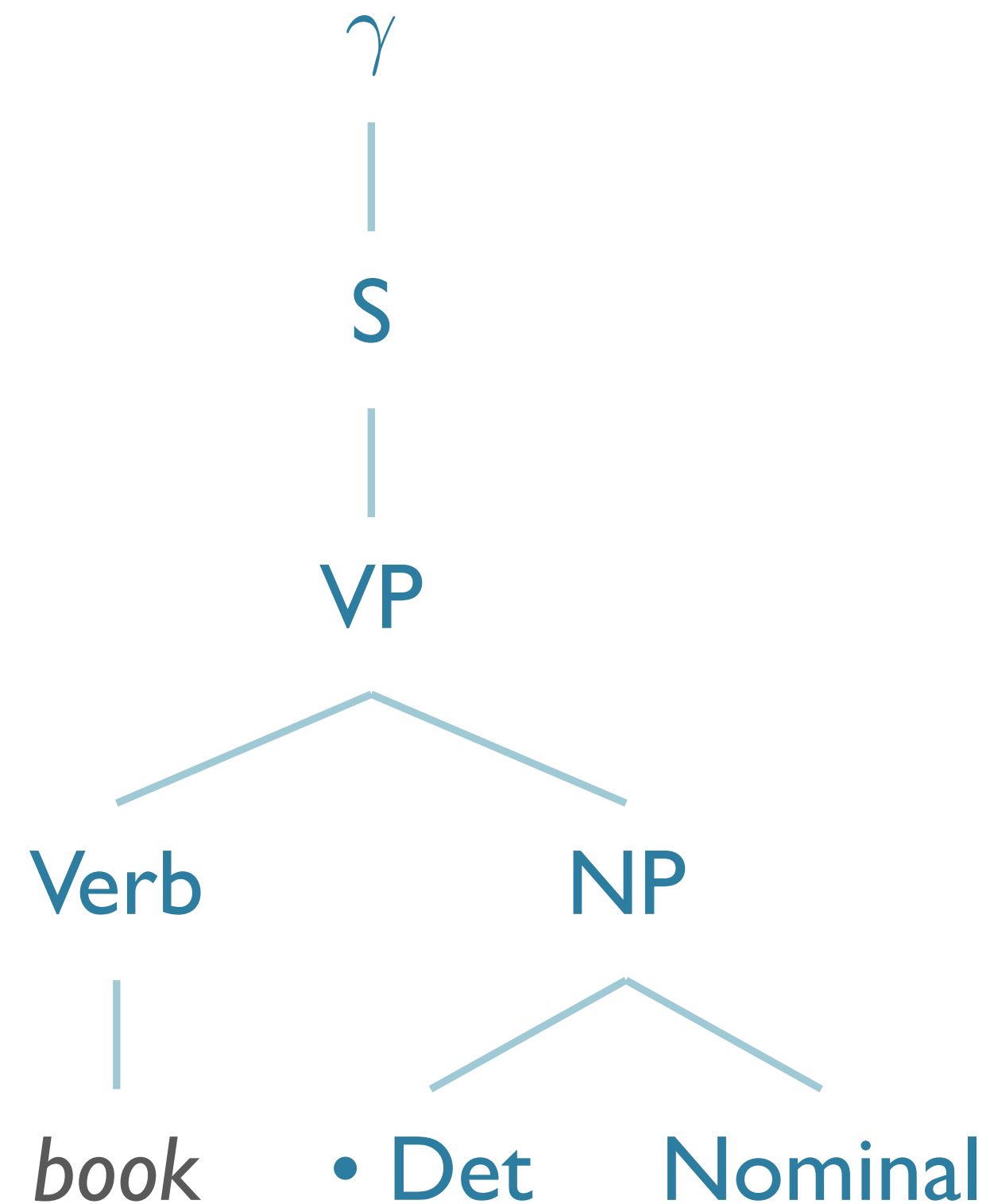
# Book that flight

S0:  $\gamma \rightarrow \cdot S$  [0,0]

S3:  $S \rightarrow VP \cdot$  [0,1]

S8:  $VP \rightarrow Verb \cdot NP$  [0,1]

S21:  $NP \rightarrow \cdot Det Nominal$  [1,1]



# Book that flight

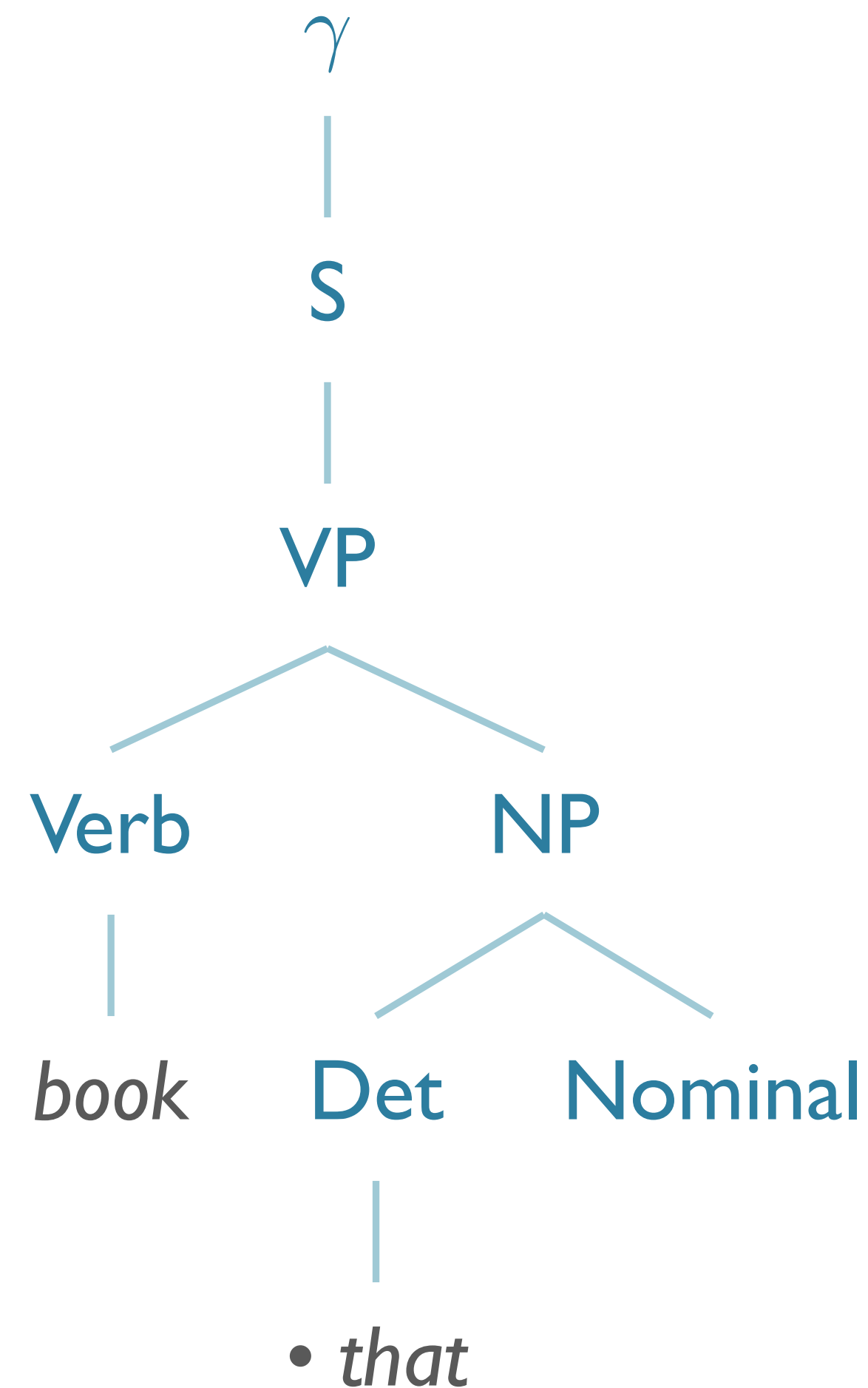
S0:  $\gamma \rightarrow \bullet S$  [0,0]

S3:  $S \rightarrow VP \bullet$  [0,1]

S8:  $VP \rightarrow Verb \bullet NP$  [0,1]

S21:  $NP \rightarrow \bullet Det Nominal$  [1,1]

S23:  $Det \rightarrow \bullet \text{"that"}$  [1,1]



# Book that flight

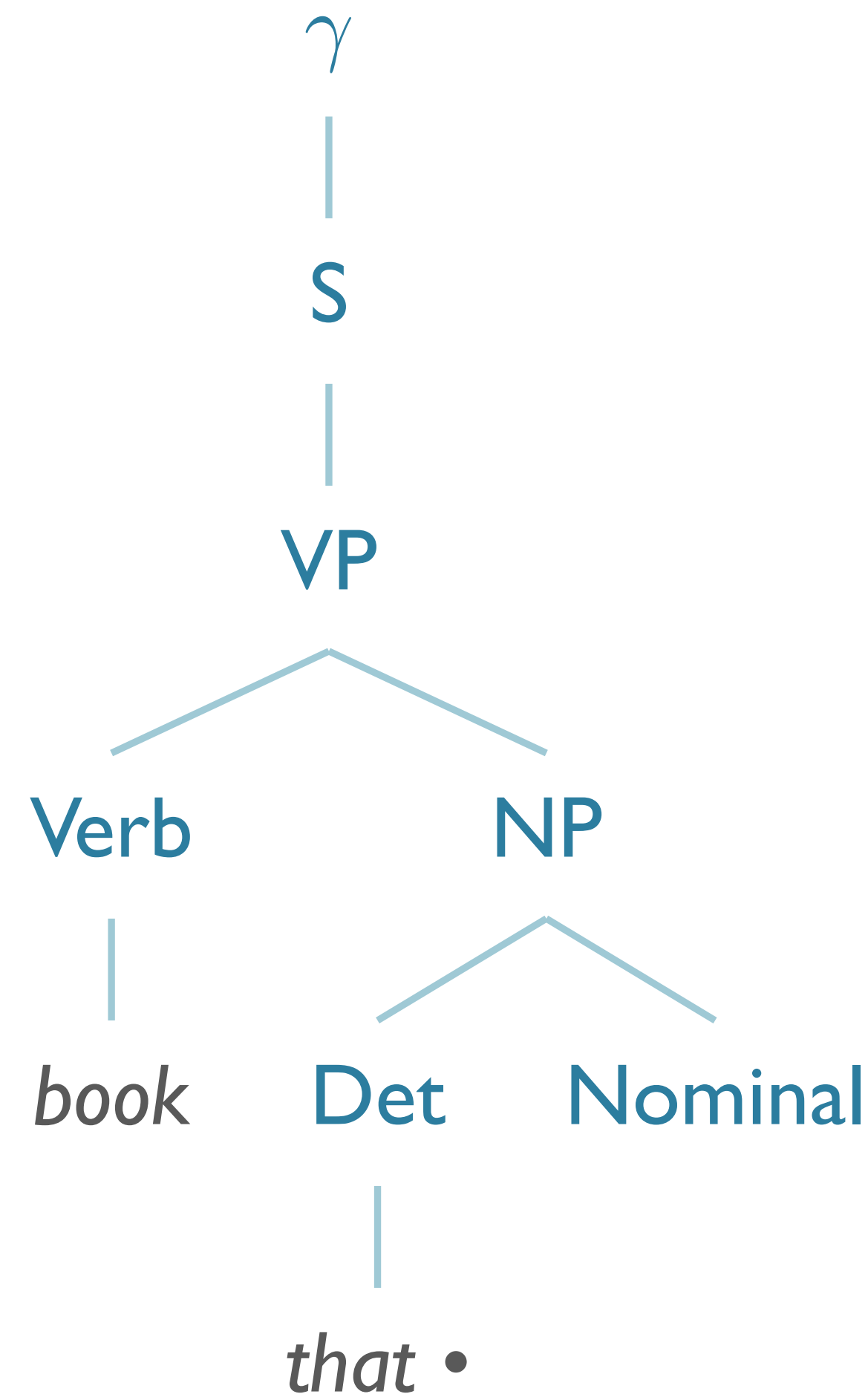
S0:  $\gamma \rightarrow \cdot S$  [0,0]

S3:  $S \rightarrow VP \cdot$  [0,1]

S8:  $VP \rightarrow Verb \cdot NP$  [0,1]

S21:  $NP \rightarrow \cdot Det Nominal$  [1,1]

S23:  $Det \rightarrow "that" \cdot$  [1,2]



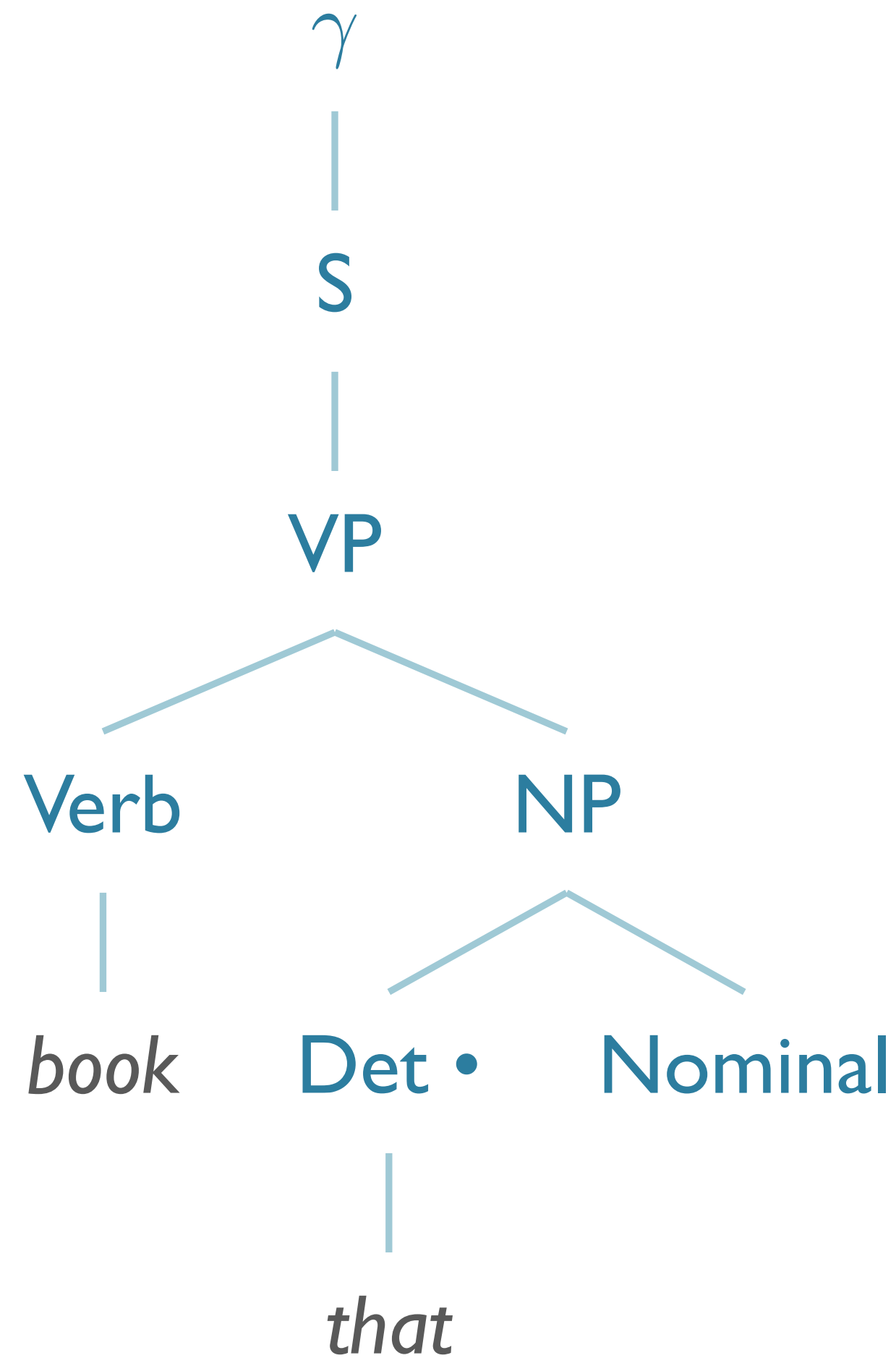
# Book that flight

S0:  $\gamma \rightarrow \cdot S$  [0,0]

S3:  $S \rightarrow VP \cdot$  [0,1]

S8:  $VP \rightarrow Verb \cdot NP$  [0,1]

S21:  $NP \rightarrow Det \cdot Nominal$  [1,2]



# Book that flight

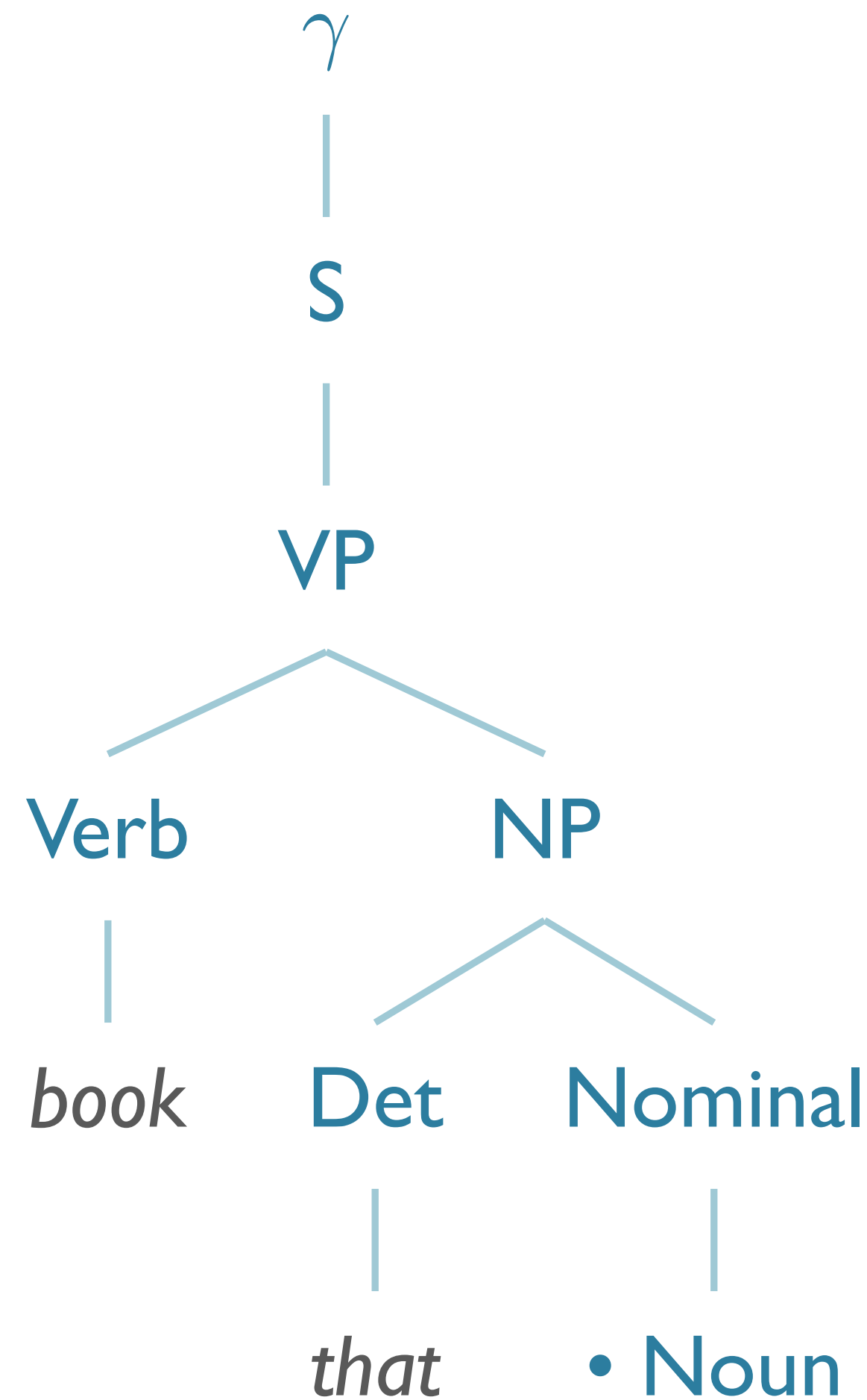
S0:  $\gamma \rightarrow \cdot S$  [0,0]

S3:  $S \rightarrow VP \cdot$  [0,1]

S8:  $VP \rightarrow Verb \cdot NP$  [0,1]

S21:  $NP \rightarrow Det \cdot Nominal$  [1,2]

S25:  $Nominal \rightarrow \cdot Noun$  [2,2]



# Book that flight

S0:  $\gamma \rightarrow \bullet S$  [0,0]

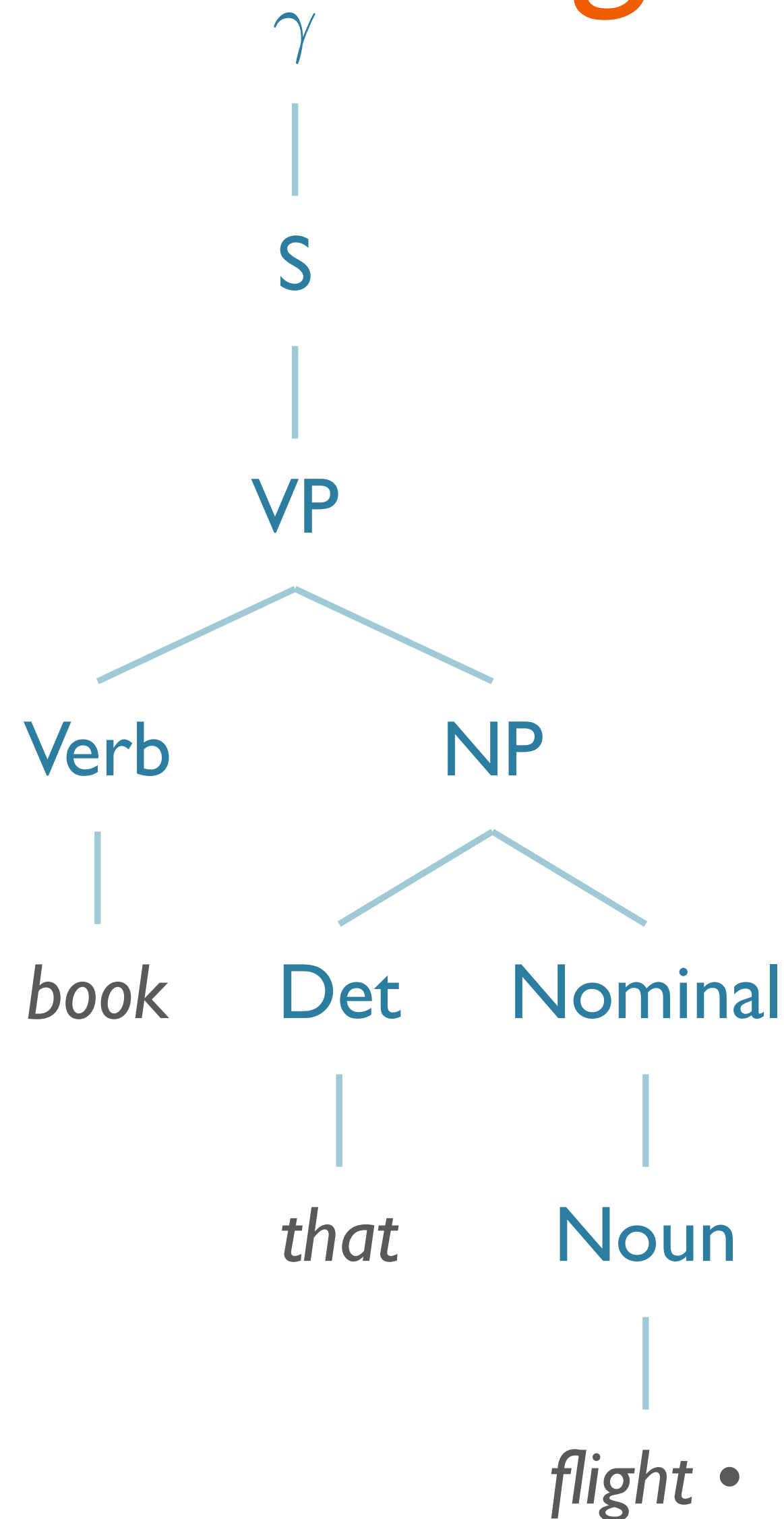
S3:  $S \rightarrow VP \bullet$  [0,1]

S8:  $VP \rightarrow Verb \bullet NP$  [0,1]

S21:  $NP \rightarrow Det \bullet Nominal$  [1,2]

S25:  $Nominal \rightarrow \bullet Noun$  [2,2]

S28:  $Noun \rightarrow \text{"flight"} \bullet$  [2,3]



# Book that flight

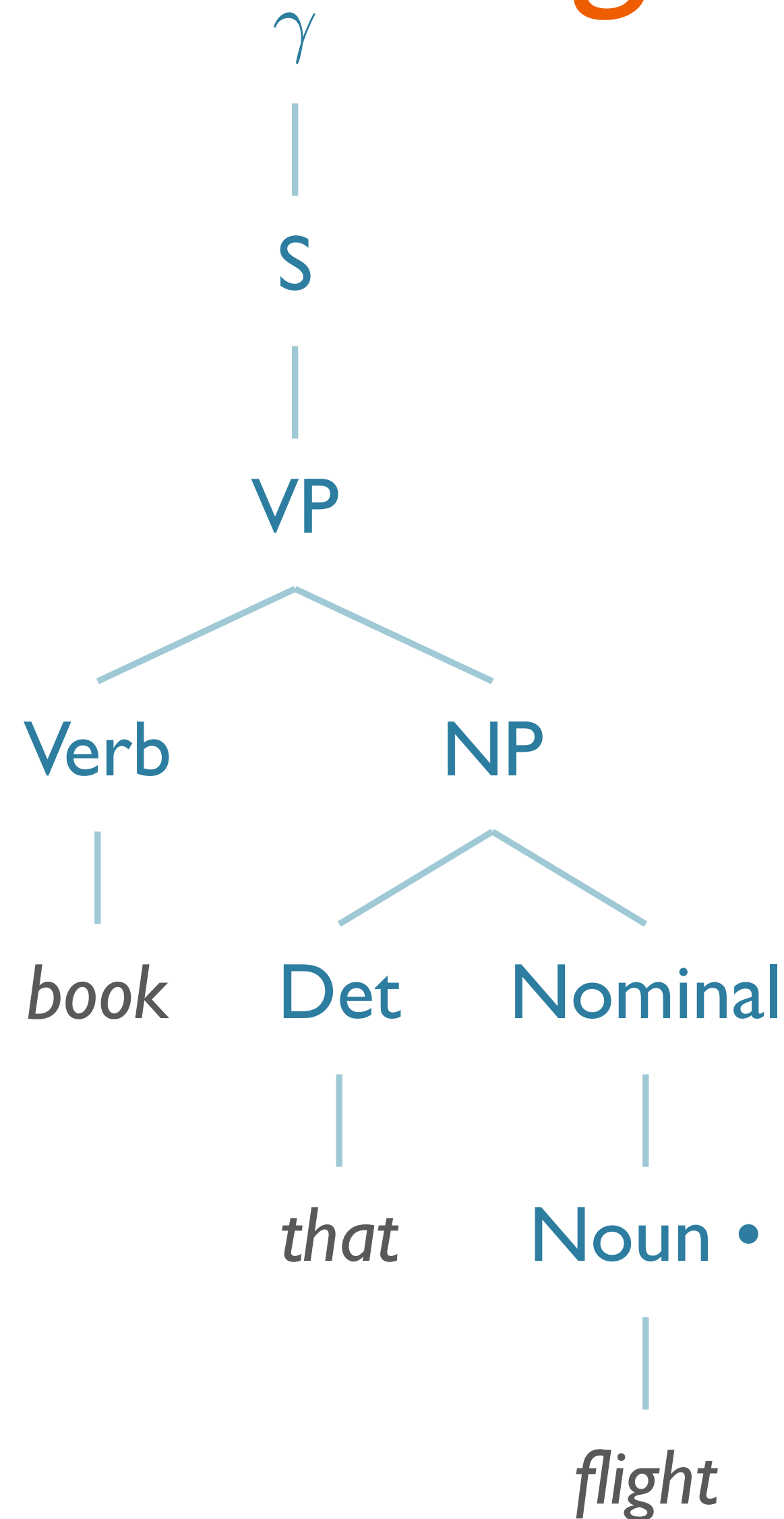
S0:  $\gamma \rightarrow \bullet S$  [0,0]

S3:  $S \rightarrow VP \bullet$  [0,1]

S8:  $VP \rightarrow Verb \bullet NP$  [0,1]

S21:  $NP \rightarrow Det \bullet Nominal$  [1,2]

S25:  $Nominal \rightarrow Noun \bullet$  [2,3]



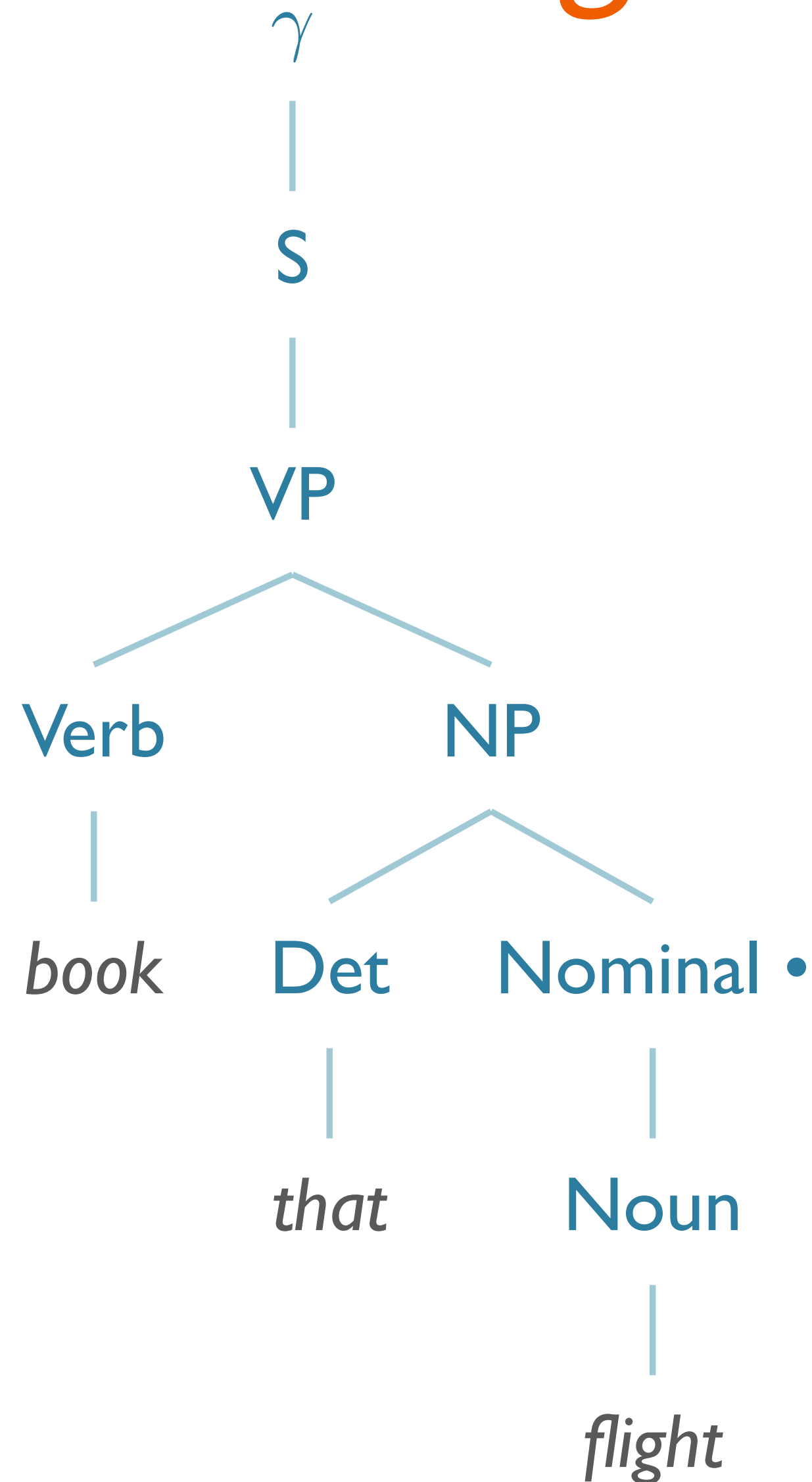
# Book that flight

S0:  $\gamma \rightarrow \bullet S$  [0,0]

S3:  $S \rightarrow VP \bullet$  [0,1]

S8:  $VP \rightarrow Verb \bullet NP$  [0,1]

S21:  $NP \rightarrow Det Nominal \bullet$  [1,3]



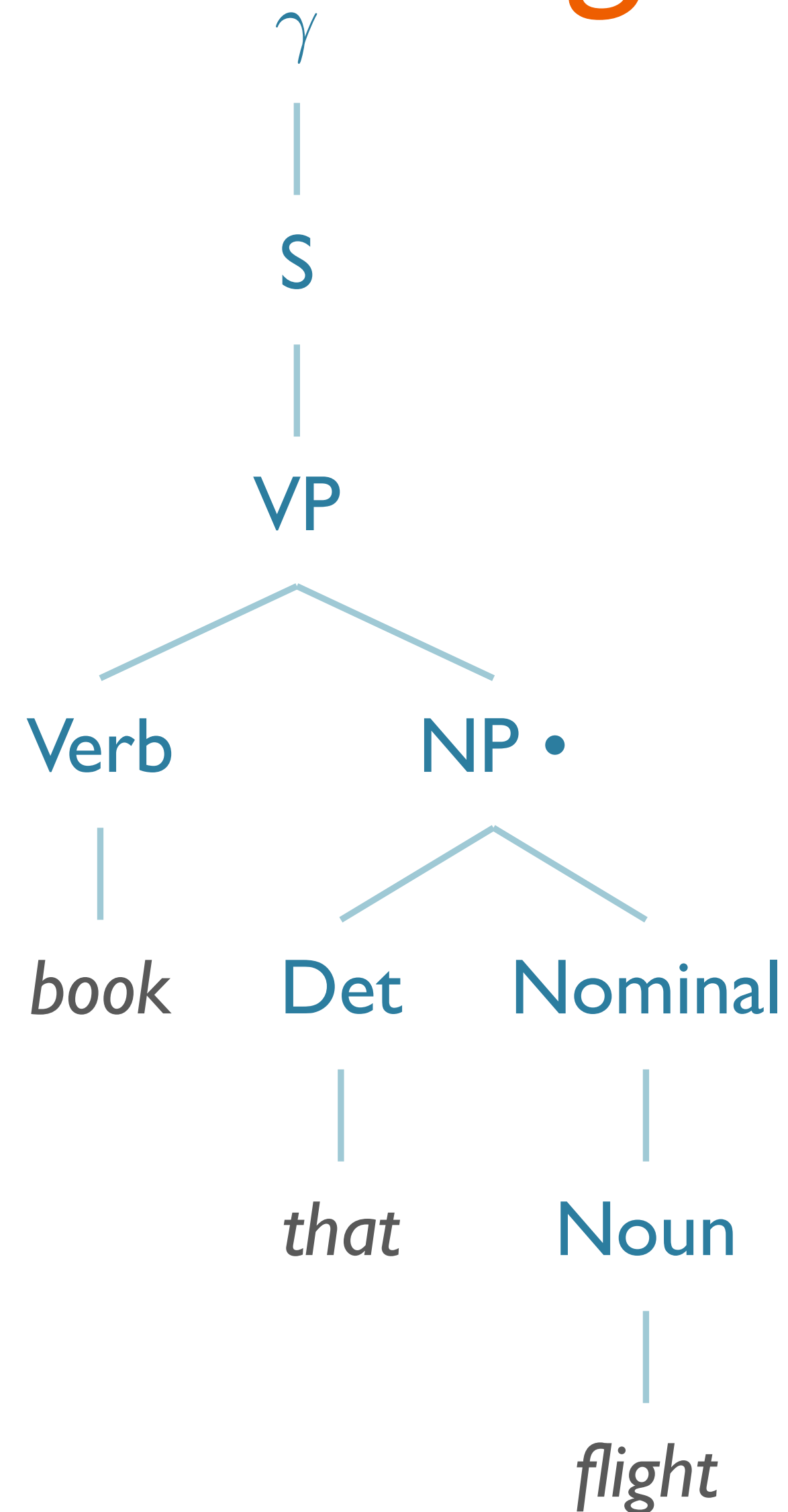


# Book that flight

S0:  $\gamma \rightarrow \bullet S$  [0,0]

S3:  $S \rightarrow VP \bullet$  [0,1]

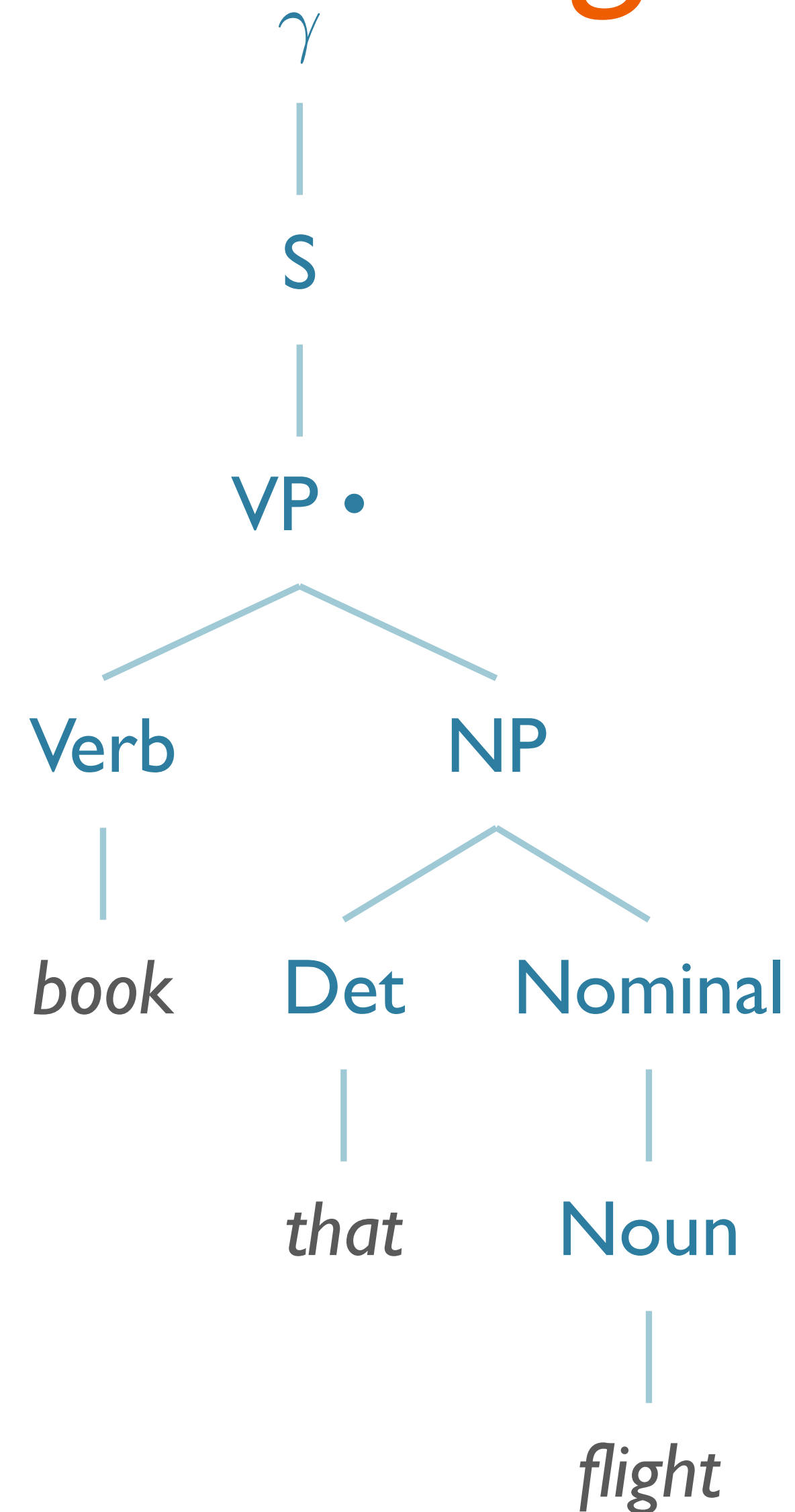
S8:  $VP \rightarrow Verb NP \bullet$  [0,3]



# Book that flight

S0:  $\gamma \rightarrow \bullet S$  [0,0]

S3:  $S \rightarrow VP \bullet$  [0,3]



# What About Dead Ends?

# Book that flight

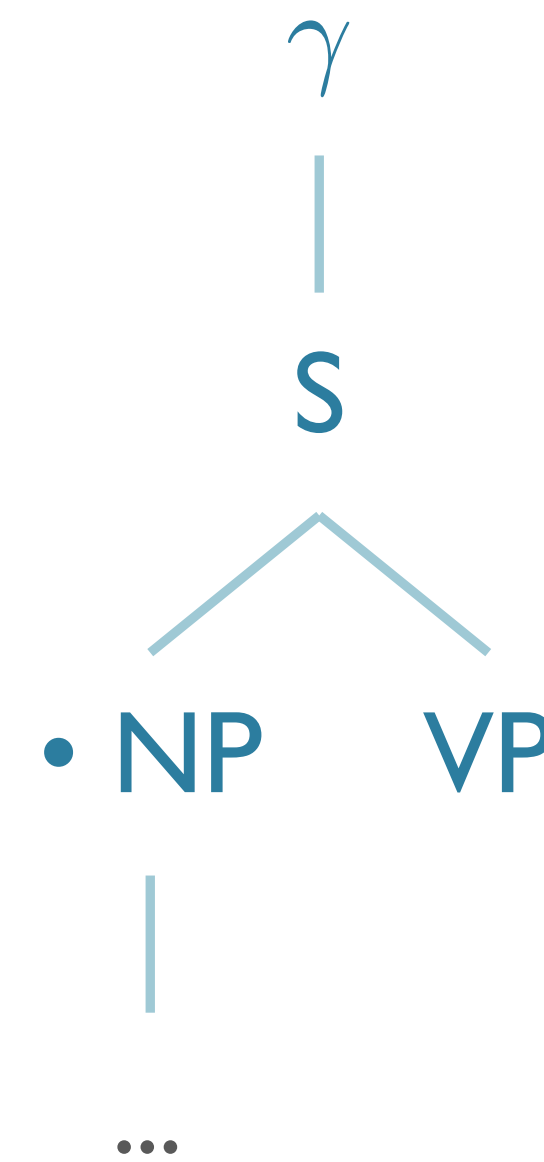
$S_0: \gamma \rightarrow \bullet S [0,0]$

$S_1: S \rightarrow \bullet NP VP [0,0]$

$NP \rightarrow \bullet Pronoun$

$NP \rightarrow \bullet Proper-Noun$

$NP \rightarrow \bullet Det Nominal$



*book*

# Book that flight

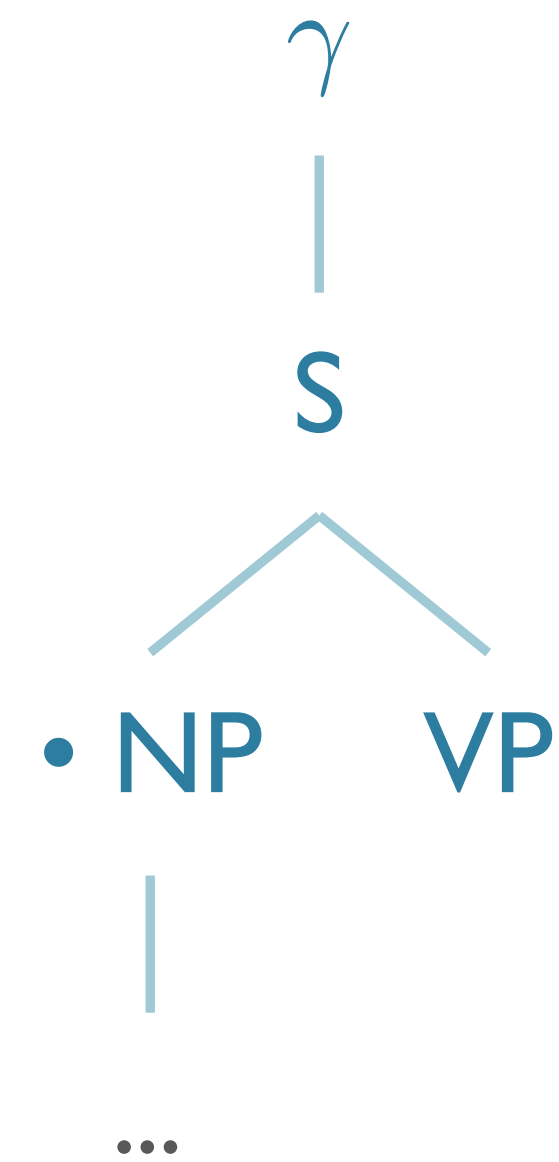
S0:  $\gamma \rightarrow \bullet S$  [0,0]

S1:  $S \rightarrow \bullet NP VP$  [0,0]

~~$NP \rightarrow \bullet \textit{Pronoun}$~~

~~$NP \rightarrow \bullet \textit{Proper-Noun}$~~

~~$NP \rightarrow \bullet \textit{Det Nominal}$~~



*book*

# What About Recursion?

# What about recursion?

# What about recursion?

- We now have a top-down parser in hand. Does it enter infinite loops on rules like  $S \rightarrow S \text{ 'and' } S$ ?



# What about recursion?

- We now have a top-down parser in hand. Does it enter infinite loops on rules like  $S \rightarrow S \text{ 'and' } S$ ?
- No!

```
procedure ENQUEUE(state, chart-entry)  
  if state is not already in chart-entry then  
    PUSH(state, chart-entry)  
  end
```

# What about recursion?

- We now have a top-down parser in hand. Does it enter infinite loops on rules like  $S \rightarrow S \text{ 'and' } S$ ?
- No!

```
procedure ENQUEUE(state, chart-entry)  
  if state is not already in chart-entry then  
    PUSH(state, chart-entry)  
  end
```

**Exercise:** parse 'table and chair' using the very simple grammar  
 $\text{Nom} \rightarrow \text{Nom 'and' Nom} \mid \text{'table'} \mid \text{'chair'}$