Probabilistic Parsing: Issues & Improvement

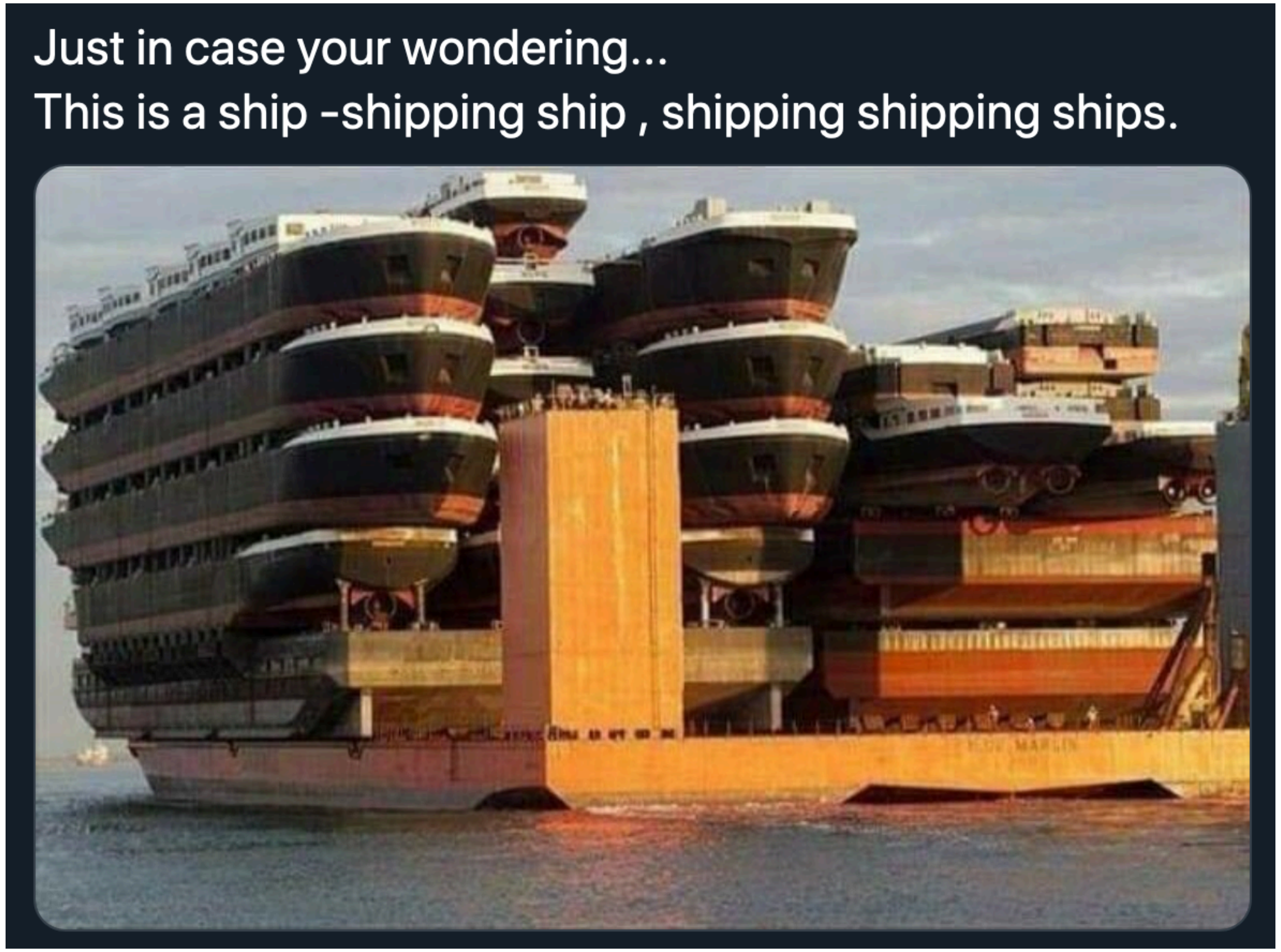
LING 571 — Deep Processing Techniques for NLP

Shane Steinert-Threlkeld

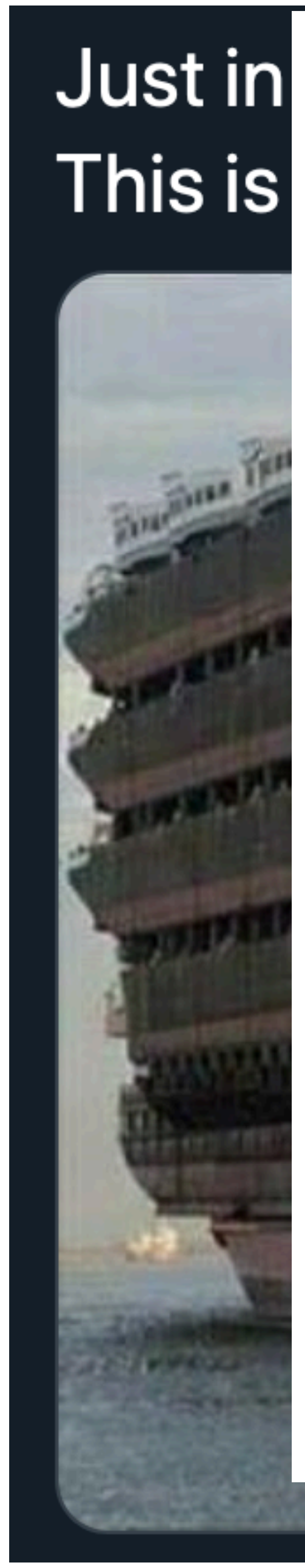
Notes on HW #3

- Python's `range` has many use cases by manipulating start/end, and step
  - `range(n)` is equivalent to `range(0, n, 1)`
- Reminder: the `rhs=` argument in NLTK's `grammar.productions()` method only matches the *first* symbol, not an entire string
  - You'll want to implement an efficient look-up based on RHS
- HW3: compare your output to running HW1 parser on the same grammar/sentences
  - order of output in ambiguous sentences could differ
- We will provide grammars in CNF; don't need to use your HW2 for that

Language Does the Darnedest Things



Language Does the Darnedest Things



Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo

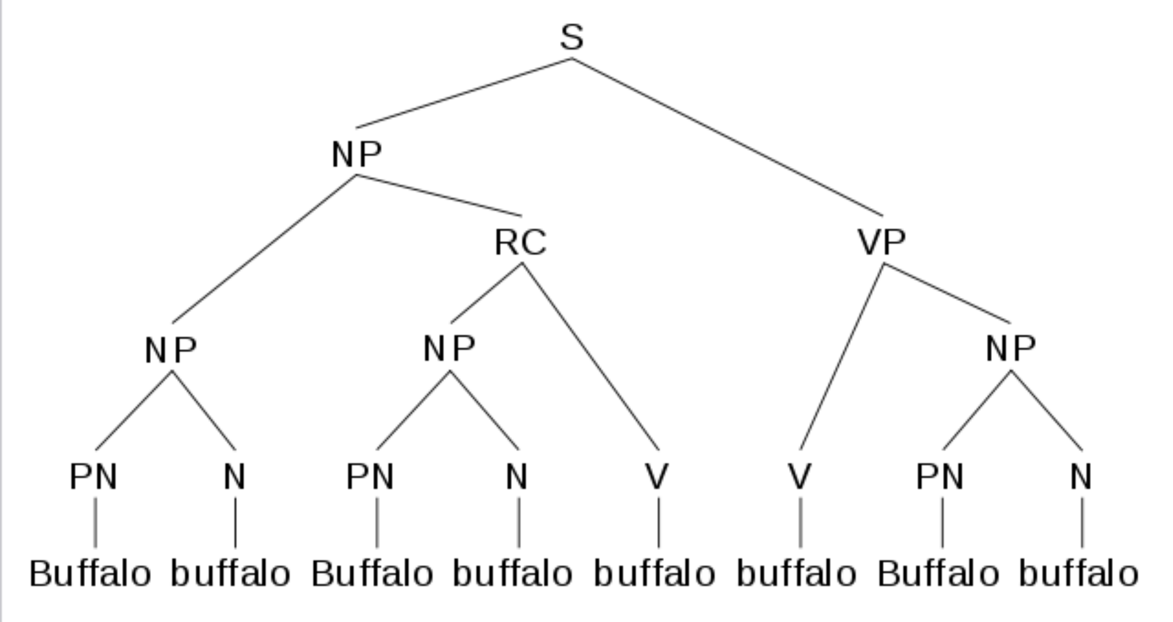
From Wikipedia, the free encyclopedia

"**Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo**" is a grammatically correct sentence in English, often presented as an example of how homonyms and homophones can be used to create complicated linguistic constructs through lexical ambiguity. It has been discussed in literature in various forms since 1967, when it appeared in Dmitri Borgmann's *Beyond Language: Adventures in Word and Thought*.

The sentence employs three distinct meanings of the word *buffalo*:

- as an adjectival proper noun to refer to a specific place named Buffalo, the city of Buffalo, New York, being the most notable;
- as a verb *to buffalo*, meaning (in American English[1]) "to bully, harass, or intimidate" or "to baffle"; and
- as a noun to refer to the animal, bison (often called *buffalo* in North America). The plural is also *buffalo*.

A semantically equivalent form preserving the original word order is: "Buffalo bison that other Buffalo bison bully also bully Buffalo bison."



Simplified parse tree

S = sentence
NP = noun phrase
RC = relative clause
VP = verb phrase
PN = proper noun
N = noun
V = verb

Unit Testing

Unit Testing

- Strategy of testing individual pieces of code in isolation
- Helps ensure:
  - Basic functionality in isolation
  - Complex functionality when individual components are combined
- In many industry jobs, you can't commit code without unit tests!

Unit Testing in Python

- Many good tutorials on the web
  - https://diveinto.org/python3/unit-testing.html
- In a nutshell:

```
from unittest import TestCase

class longTests(TestCase):
 def test_three(self):
   length_3_rule = parse_productions('A -> B C D')
   target_rules = parse_productions('''A -> B _X0_
                                      _X0_ -> C D''')
   self.assertSetEqual(set(target_rules),
                       set(fix_long_rules(length_3_rule)))
```

Unit Testing in Python

- Built-in unittest module/library:

```
python -m unittest hw2.py
```

```
................
----------------------------------------------------------------------
Ran 16 tests in 0.002s

OK
```

Unit Testing

- Good practice:
  - Save input that crashes your program for a unit test
- Other popular unit testing frameworks for python (e.g. in 574):
  - pytest: https://docs.pytest.org/
    - Nice auto-discovery of tests based on file, class, and method name
    - Works with native assert statements, not special ones
    - …

Today's Plan

- PCFG Induction example
- Problems with PCFGs
  - Independence
  - Lack of lexical conditioning
- Improving PCFGs
  - Coverage (3 methods)
  - Efficiency

PCFG Induction

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: $\Sigma_\gamma \, Count(\alpha \rightarrow \gamma)$

Learning Probabilities

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

Learning Probabilities

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

$$P(\alpha \rightarrow \beta \,|\, \alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_\gamma Count(\alpha \rightarrow \gamma)} = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

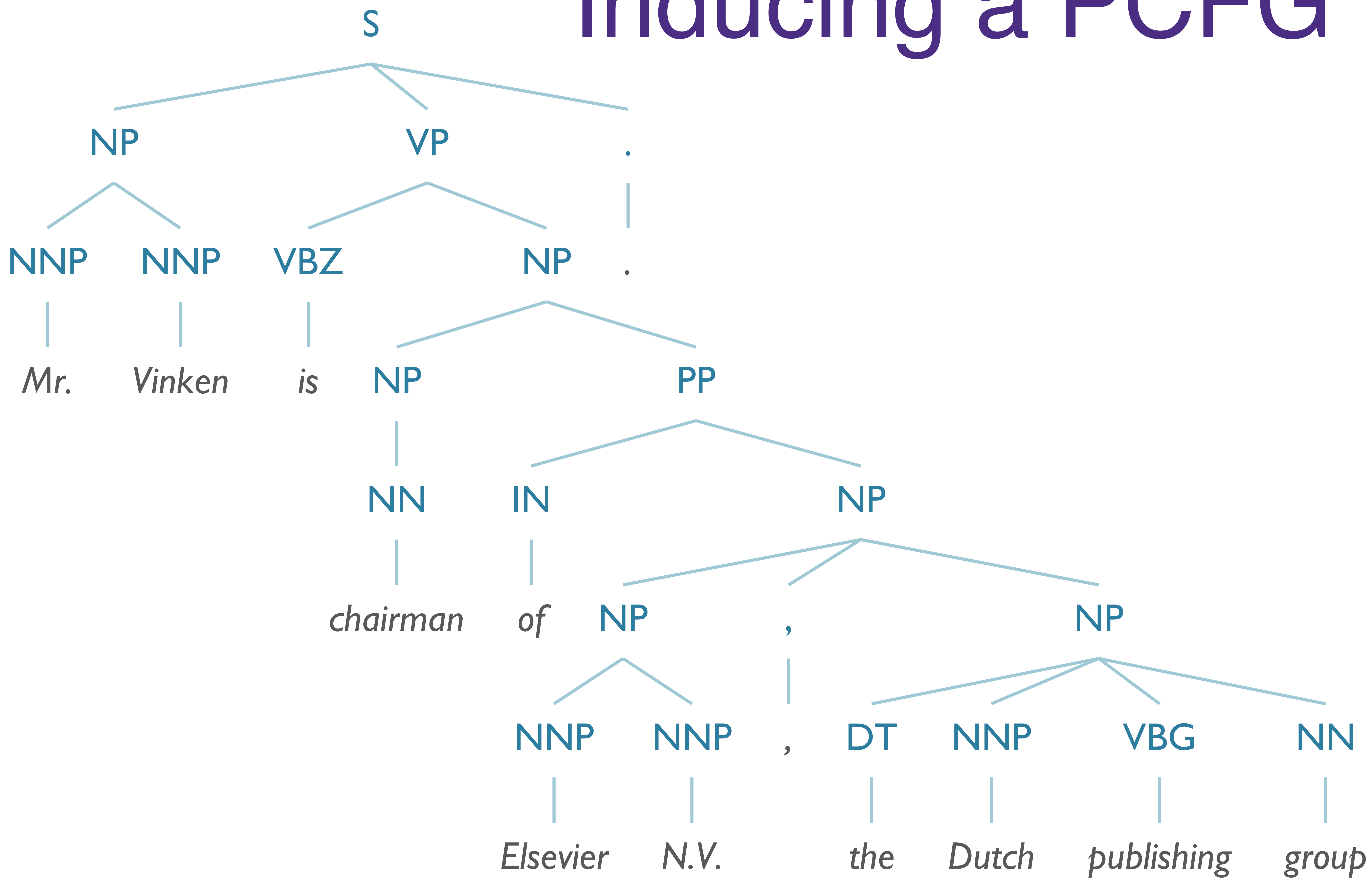
Learning Probabilities

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

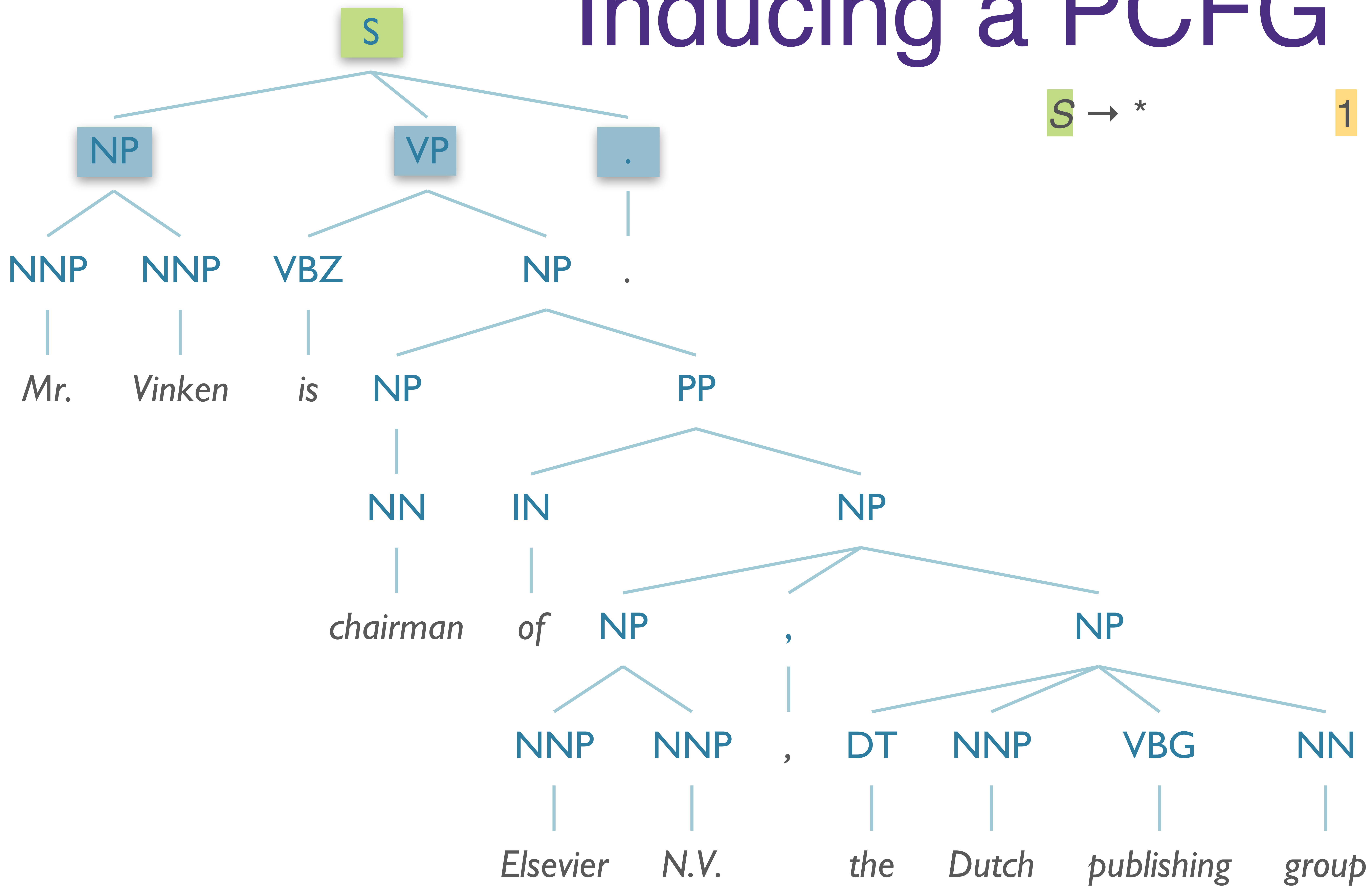
$$P(\alpha \to \beta \mid \alpha) = \frac{Count(\alpha \to \beta)}{\sum_\gamma Count(\alpha \to \gamma)} = \frac{Count(\alpha \to \beta)}{Count(\alpha)}$$

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

Inducing a PCFG

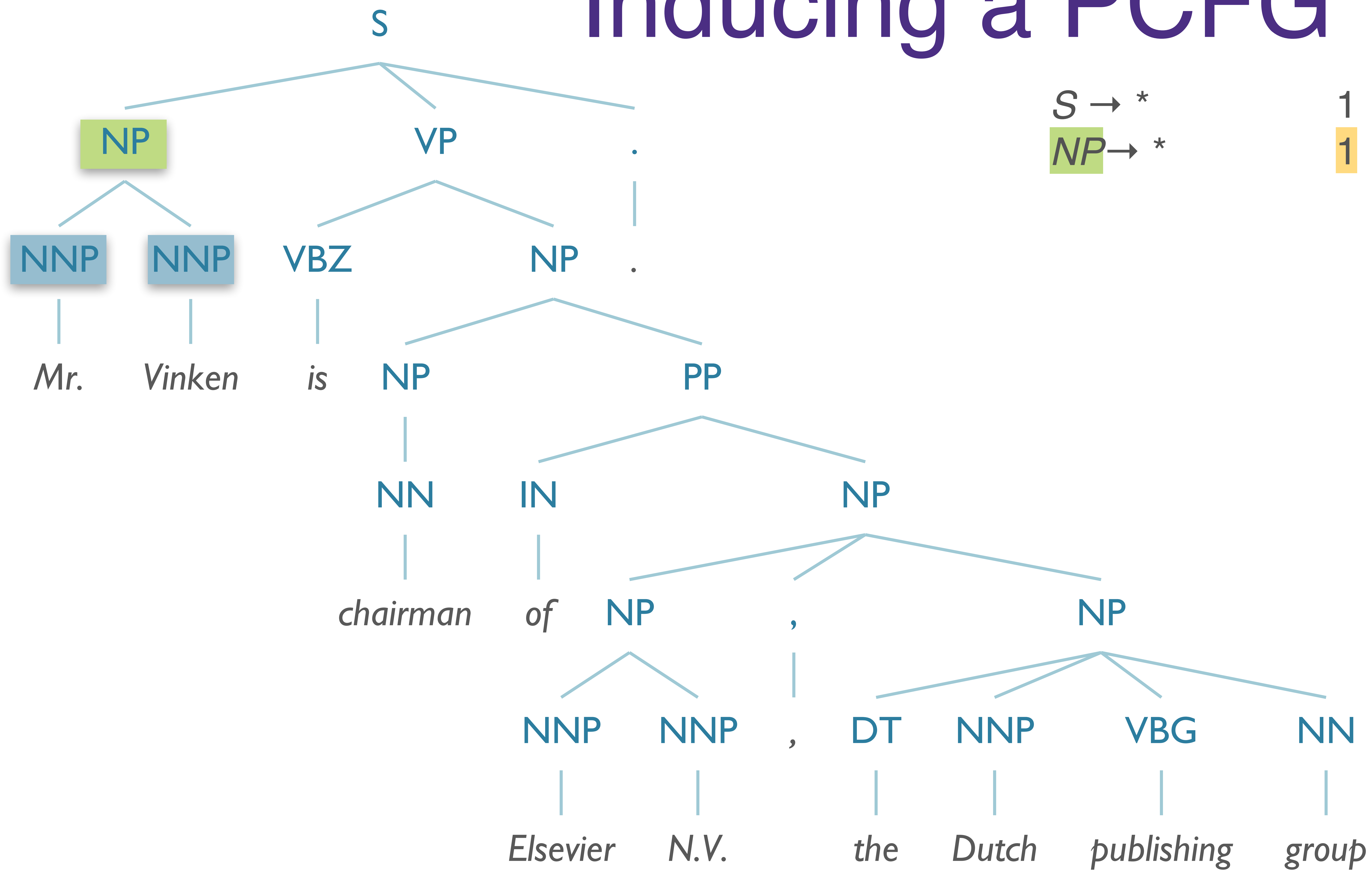


Inducing a PCFG



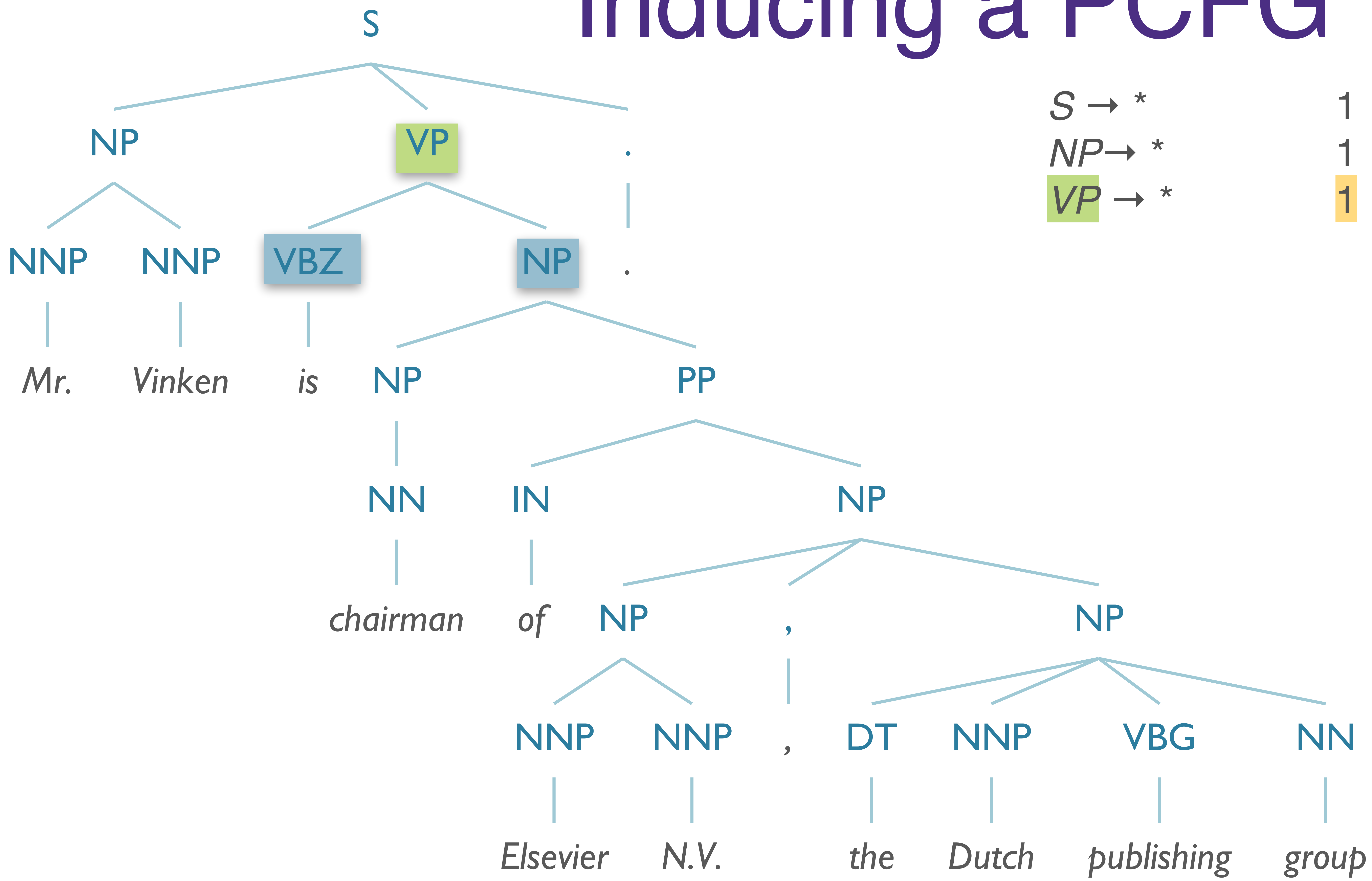
S → *	1	*S* → *NP VP .*	1

Inducing a PCFG



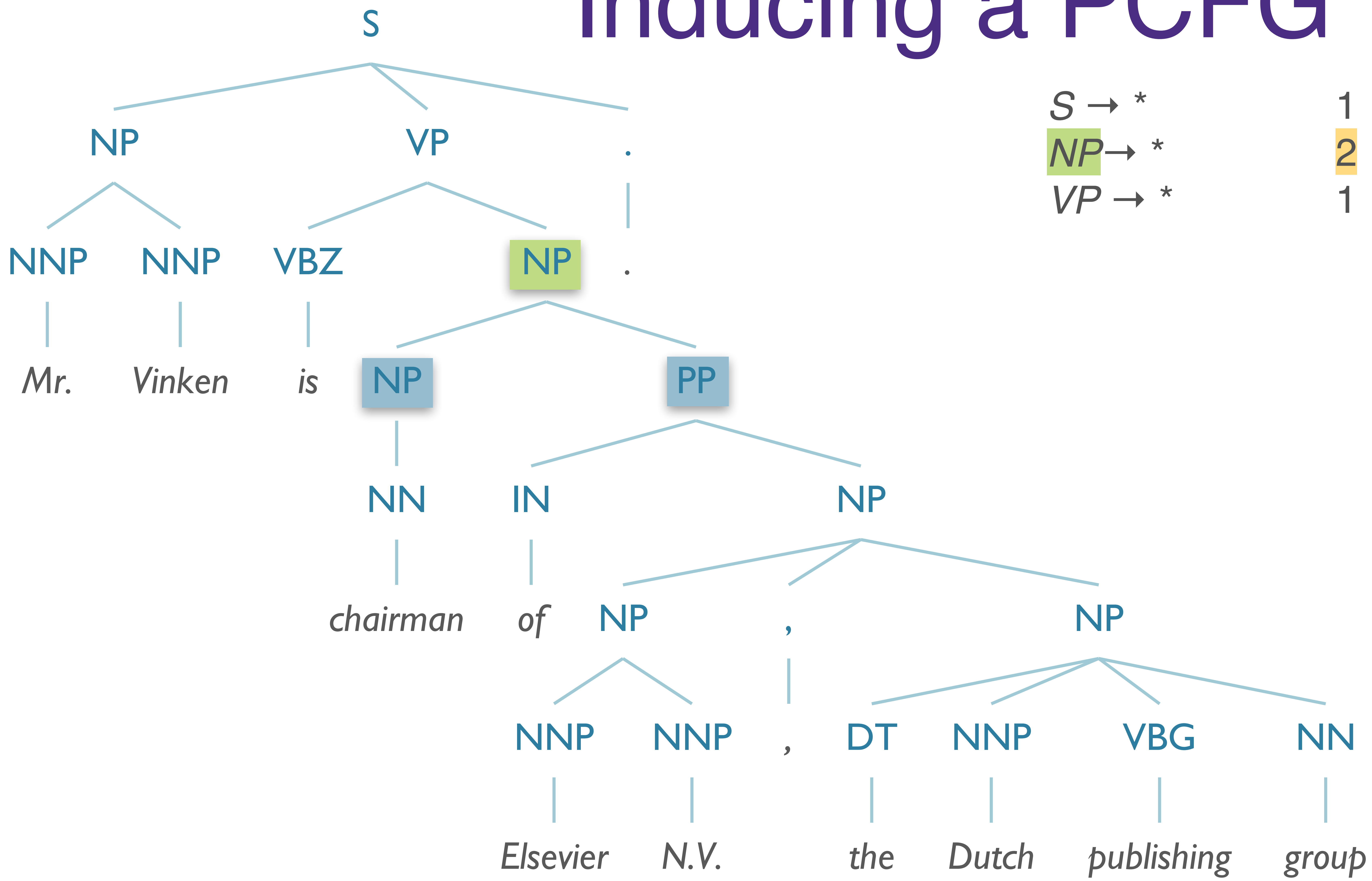
$S \rightarrow *$	1	$S \rightarrow NP\ VP\ .$	1
$NP \rightarrow *$	1	$NP \rightarrow NNP\ NNP$	1

Inducing a PCFG



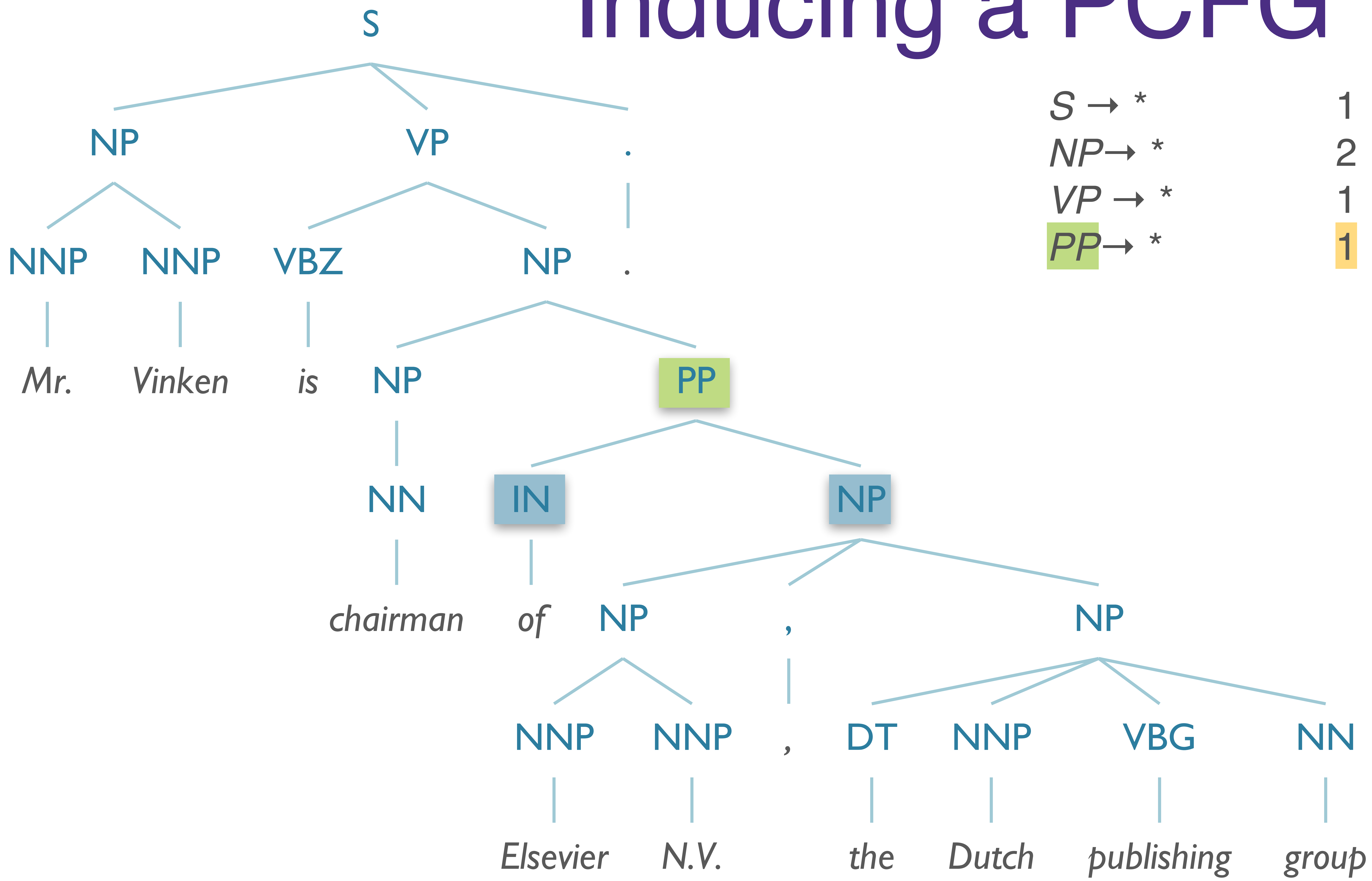
$S \rightarrow *$	1	$S \rightarrow NP\ VP\ .$	1
$NP \rightarrow *$	1	$NP \rightarrow NNP\ NNP$	1
$VP \rightarrow *$	1	$VP \rightarrow VBZ\ NP$	1

Inducing a PCFG



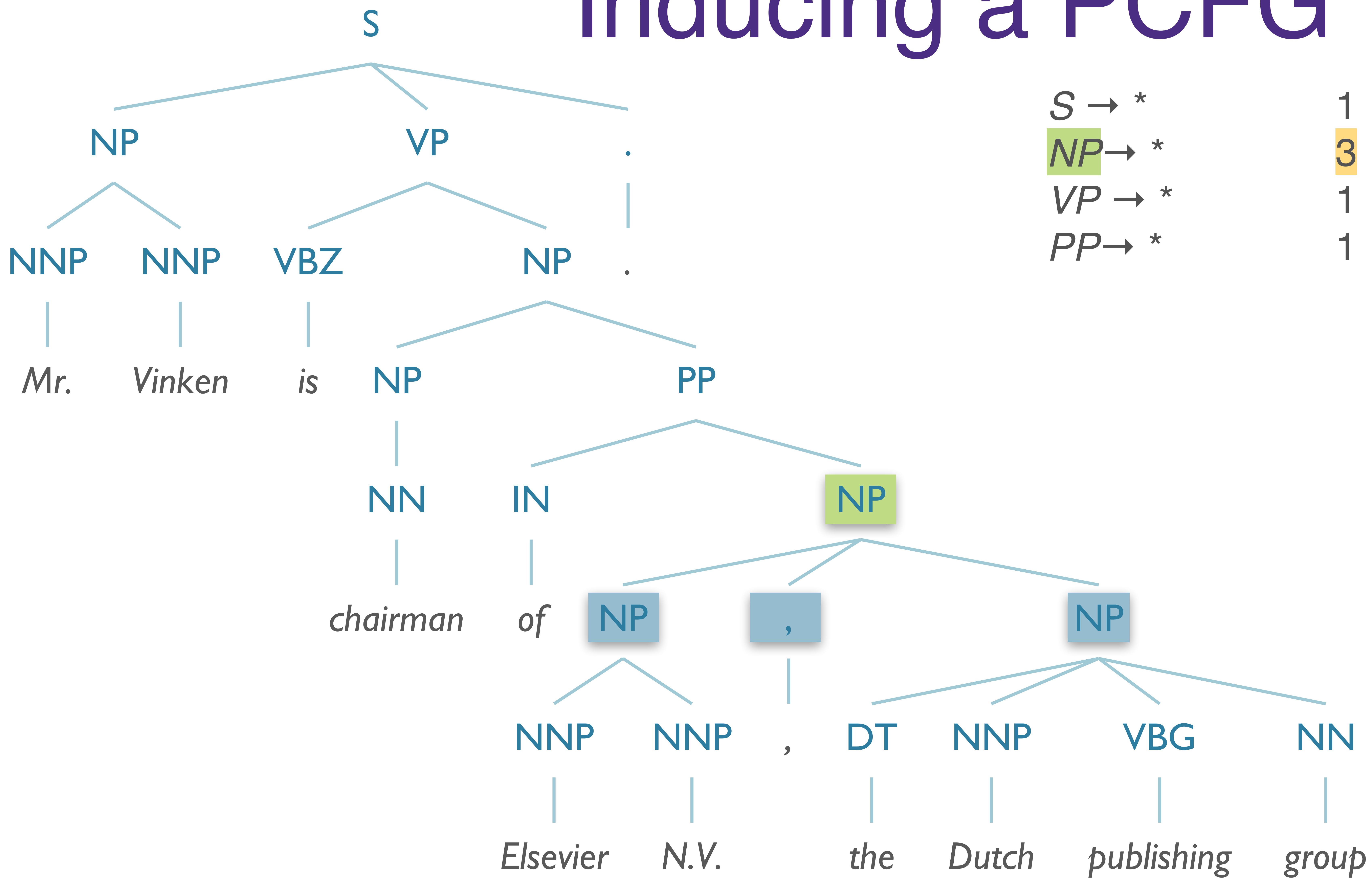
$S \rightarrow *$	1	$S \rightarrow NP\ VP\ .$	1
$NP \rightarrow *$	2	$NP \rightarrow NNP\ NNP$	1
$VP \rightarrow *$	1	$VP \rightarrow VBZ\ NP$	1
		$NP \rightarrow NP\ PP$	1

Inducing a PCFG



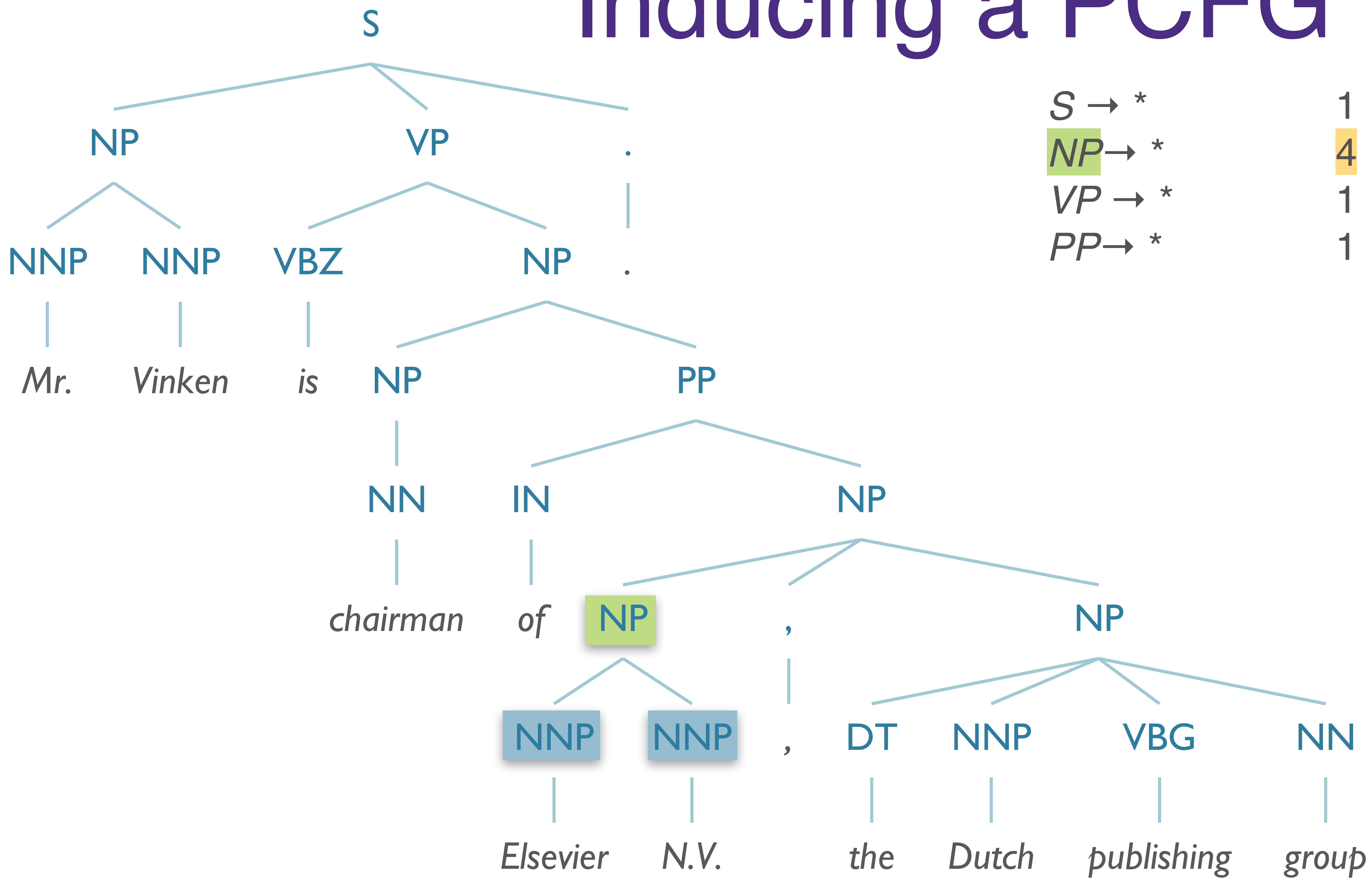
*S → **	1	*S → NP VP .*	1
*NP→ **	2	*NP → NNP NNP*	1
*VP → **	1	*VP → VBZ NP*	1
*PP→ **	1	*NP→ NP PP*	1
		PP→ IN NP	1

Inducing a PCFG



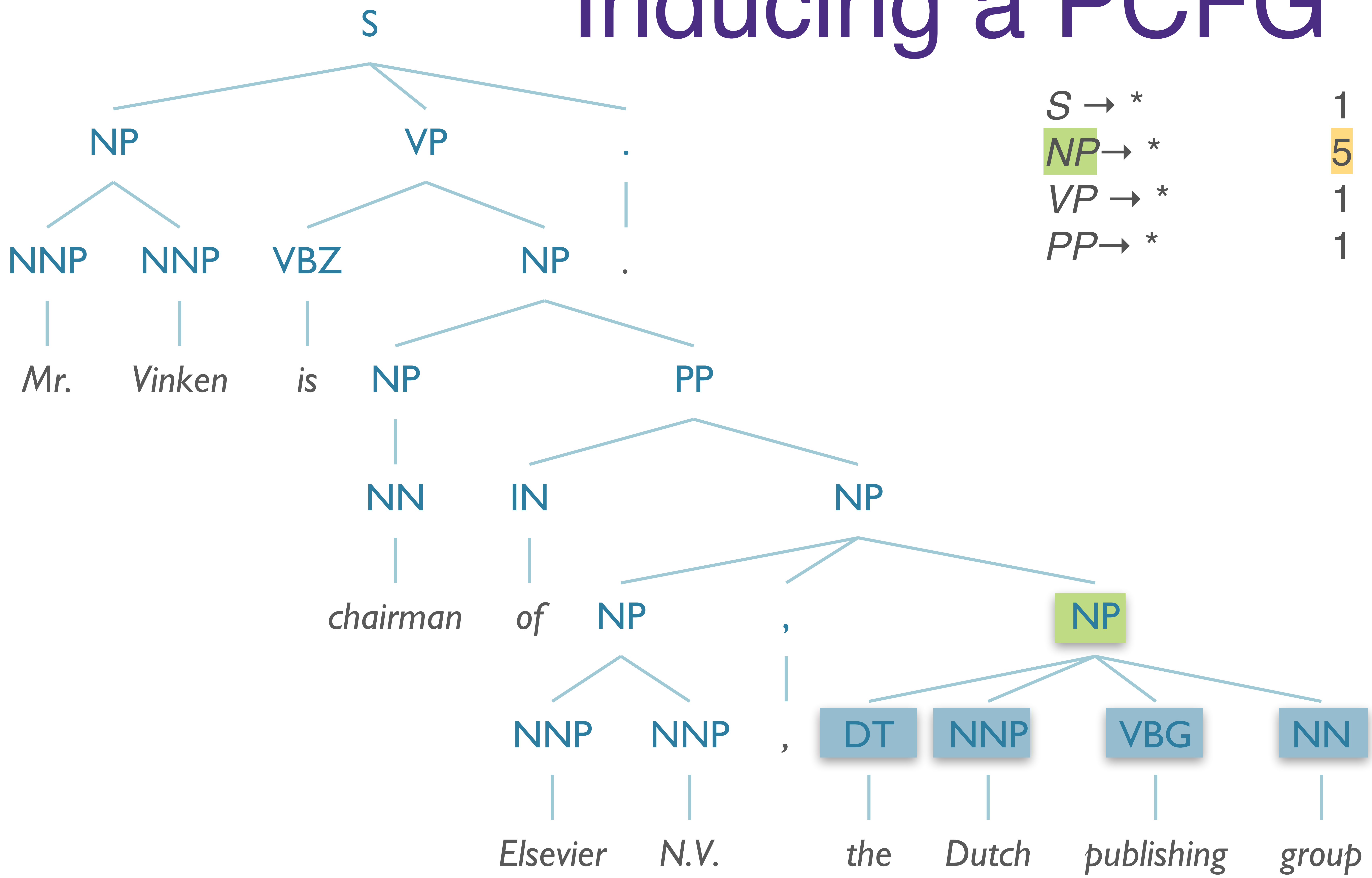
S → *	1	*S* → *NP VP .*	1
NP → *	3	*NP* → *NNP NNP*	1
VP → *	1	*VP* → *VBZ NP*	1
PP → *	1	*NP* → *NP PP*	1
		PP → *IN NP*	1
		NP → *NP , NP*	1

Inducing a PCFG



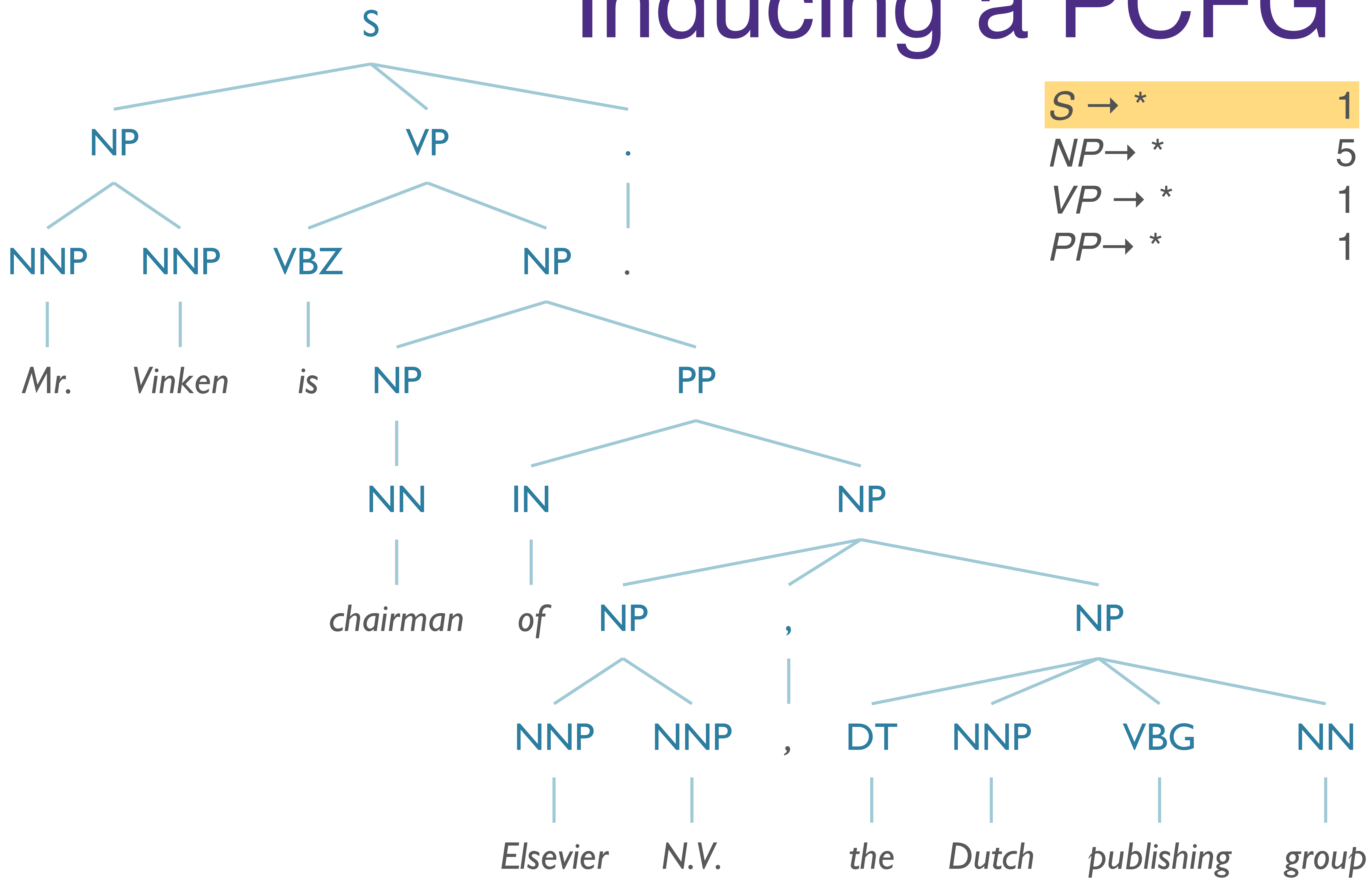
*S → **	1	*S → NP VP .*	1
*NP → **	4	*NP → NNP NNP*	2
*VP → **	1	*VP → VBZ NP*	1
*PP → **	1	*NP → NP PP*	1
		PP → IN NP	1
		NP → NP , NP	1

Inducing a PCFG



S → *	1	*S* → *NP VP .*	1
NP→ *	5	*NP*→ *NNP NNP*	2
VP → *	1	*VP* → *VBZ NP*	1
PP→ *	1	*NP*→ *NP PP*	1
		PP→ *IN NP*	1
		NP→ *NP , NP*	1
		NP→ *DT NNP VBG NN*	1

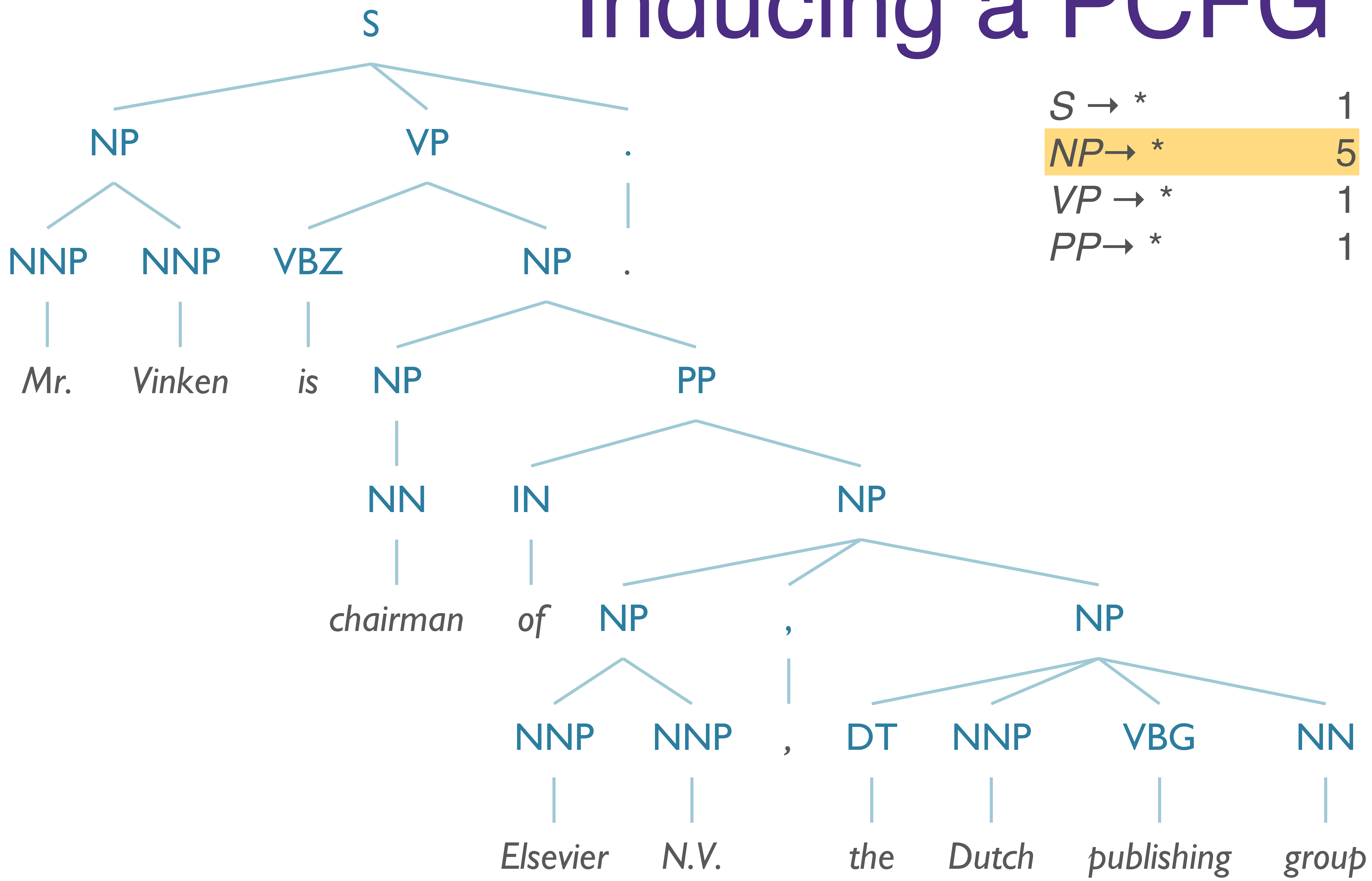
Inducing a PCFG



*S → **	1
*NP→ **	5
*VP → **	1
*PP→ **	1

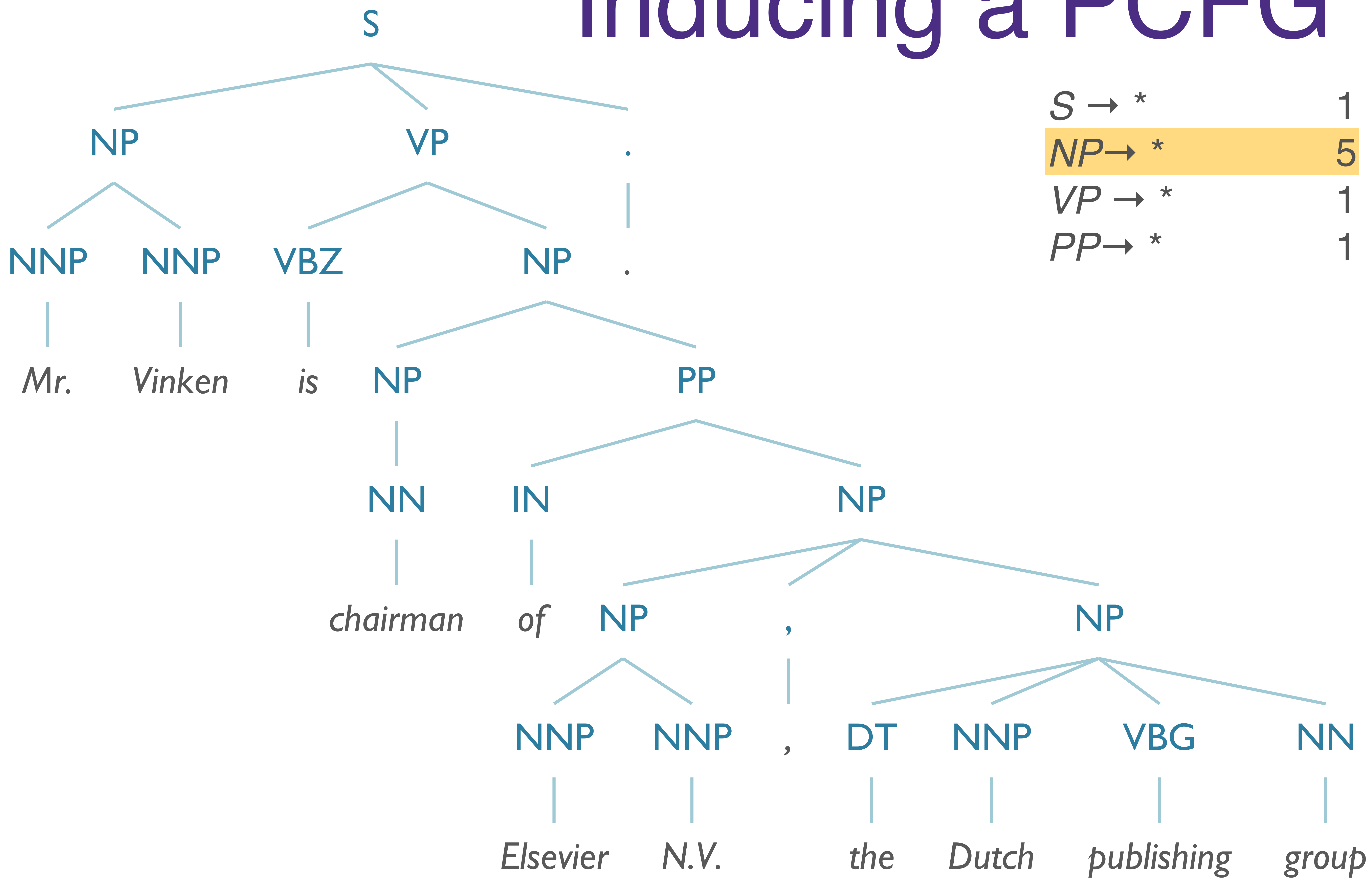
S → NP VP .	1
NP→ NNP NNP	2
VP → VBZ NP	1
NP→ NP PP	1
PP→ IN NP	1
NP→ NP , NP	1
NP→ DT NNP VBG NN	1

Inducing a PCFG



*S → **	1	*S → NP VP .*	1
*NP→ **	5	*NP→ NNP NNP*	2/5
*VP → **	1	*VP → VBZ NP*	1
*PP→ **	1	*NP→ NP PP*	1/5
		PP→ IN NP	1
		NP→ NP , NP	1/5
		NP→ DT NNP VBG NN	1/5

Inducing a PCFG



*S → **	1	*S → NP VP .*	1
*NP→ **	5	*NP→ NNP NNP*	0.4
*VP → **	1	*VP → VBZ NP*	1
*PP→ **	1	*NP→ NP PP*	0.2
		PP→ IN NP	1
		NP→ NP , NP	0.2
		NP→ DT NNP VBG NN	0.2

Problems with PCFGs

Problems with PCFGs

- Independence Assumption
  - Assume that rule probabilities are independent

Problems with PCFGs

- Independence Assumption
  - Assume that rule probabilities are independent

- Lack of Lexical Conditioning
  - Lexical items should influence the choice of analysis

Issues with PCFGs: Independence Assumption

- *Context Free ⇒ Independence Assumption*
  - Rule expansion is context-independent
  - Allows us to multiply probabilities

Issues with PCFGs: Independence Assumption

- *Context Free ⇒ Independence Assumption*
  - Rule expansion is context-independent
  - Allows us to multiply probabilities
- If we have two rules:
  - $NP \rightarrow DT\ NN \quad [0.28]$
  - $NP \rightarrow PRP \qquad [0.25]$

Issues with PCFGs: Independence Assumption

- *Context Free ⇒ Independence Assumption*
  - Rule expansion is context-independent
  - Allows us to multiply probabilities
- If we have two rules:
  - $NP \rightarrow DT\ NN \quad [0.28]$
  - $NP \rightarrow PRP \quad [0.25]$

Semantic Role of **NPs** in Switchboard Corpus

	Pronomial	Non-Pronomial
Subject	91%	9%
Object	34%	66%

Issues with PCFGs: Independence Assumption

- *Context Free ⇒ Independence Assumption*
  - Rule expansion is context-independent
  - Allows us to multiply probabilities
- If we have two rules:
  - $NP \rightarrow DT\ NN \quad [0.28]$
  - $NP \rightarrow PRP \quad [0.25]$
- What does this new data tell us?

Semantic Role of **NPs** in Switchboard Corpus

	Pronomial	Non-Pronomial
Subject	91%	9%
Object	34%	66%

Issues with PCFGs: Independence Assumption

- *Context Free ⇒ Independence Assumption*
  - Rule expansion is context-independent
  - Allows us to multiply probabilities
- If we have two rules:
  - $NP \rightarrow DT\ NN \quad [0.28]$
  - $NP \rightarrow PRP \quad [0.25]$
- What does this new data tell us?
  - $NP \rightarrow DT\ NN \quad [0.09 \textbf{ if } NP_{\Theta=subject} \textbf{ else } 0.66]$
  - $NP \rightarrow PRP \quad [0.91 \textbf{ if } NP_{\Theta=subject} \textbf{ else } 0.34]$

Semantic Role of **NPs** in Switchboard Corpus

	Pronomial	Non-Pronomial
Subject	91%	9%
Object	34%	66%

Issues with PCFGs: Independence Assumption

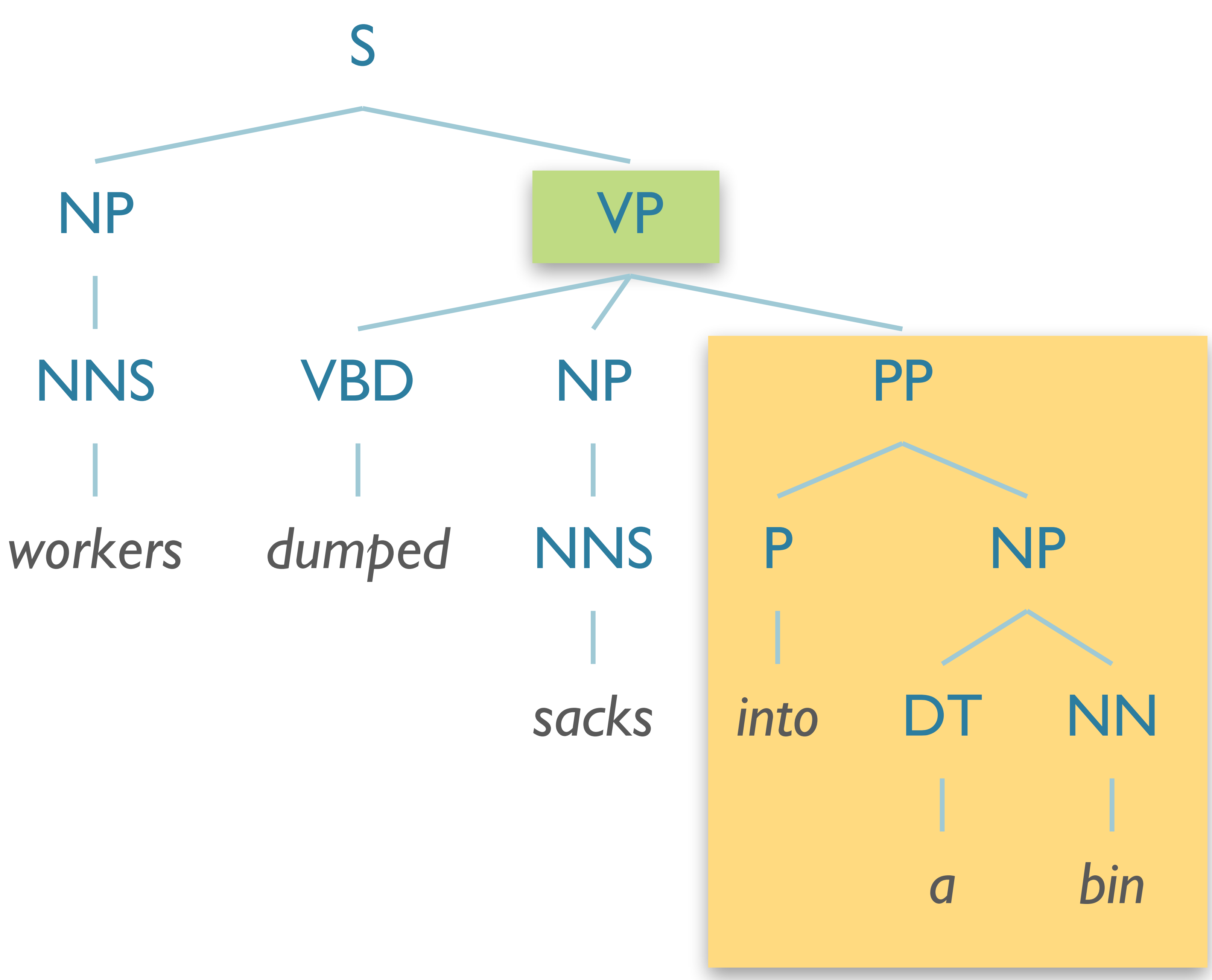
- *Context Free ⇒ Independence Assumption*
  - Rule expansion is context-independent
  - Allows us to multiply probabilities
- If we have two rules:
  - $NP \rightarrow DT\ NN \quad [0.28]$
  - $NP \rightarrow PRP \quad [0.25]$
- What does this new data tell us?
  - $NP \rightarrow DT\ NN \quad [0.09 \textbf{ if } NP_{\Theta=subject} \textbf{ else } 0.66]$
  - $NP \rightarrow PRP \quad [0.91 \textbf{ if } NP_{\Theta=subject} \textbf{ else } 0.34]$

Semantic Role of **NPs** in Switchboard Corpus

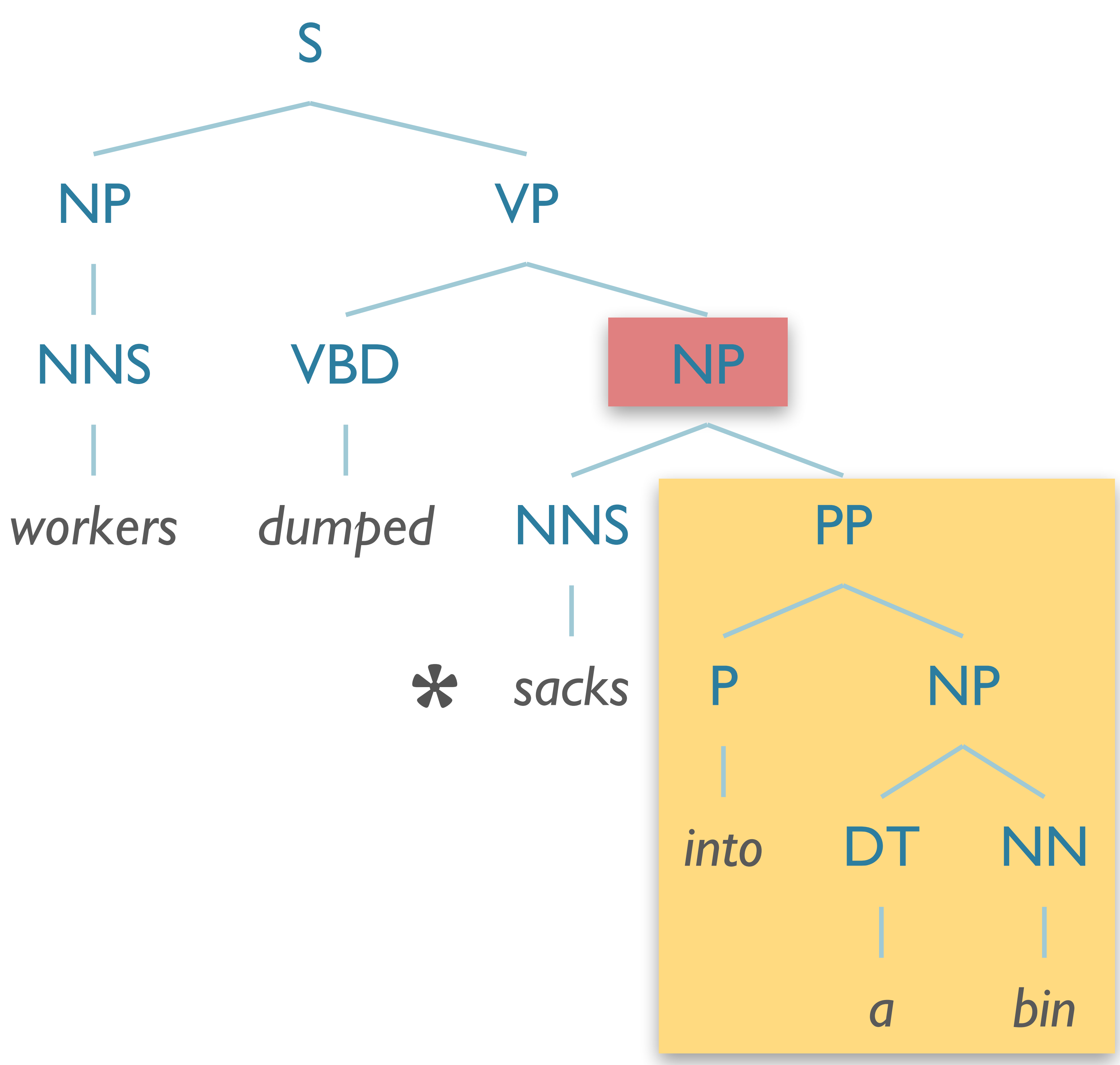
	Pronomial	Non-Pronomial
Subject	91%	9%
Object	34%	66%

…Can try **parent annotation**

Issues with PCFGs: Lexical Conditioning

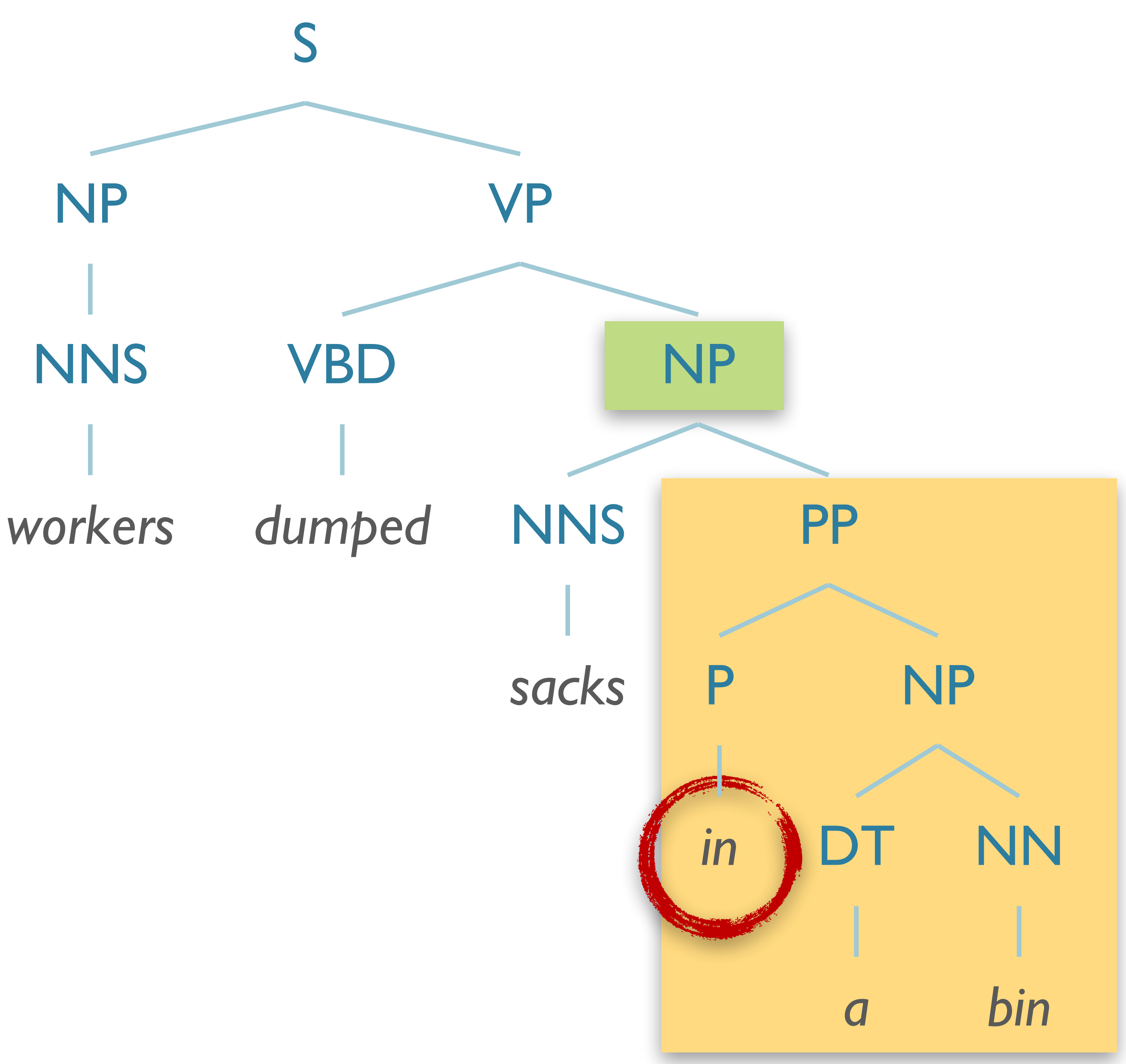


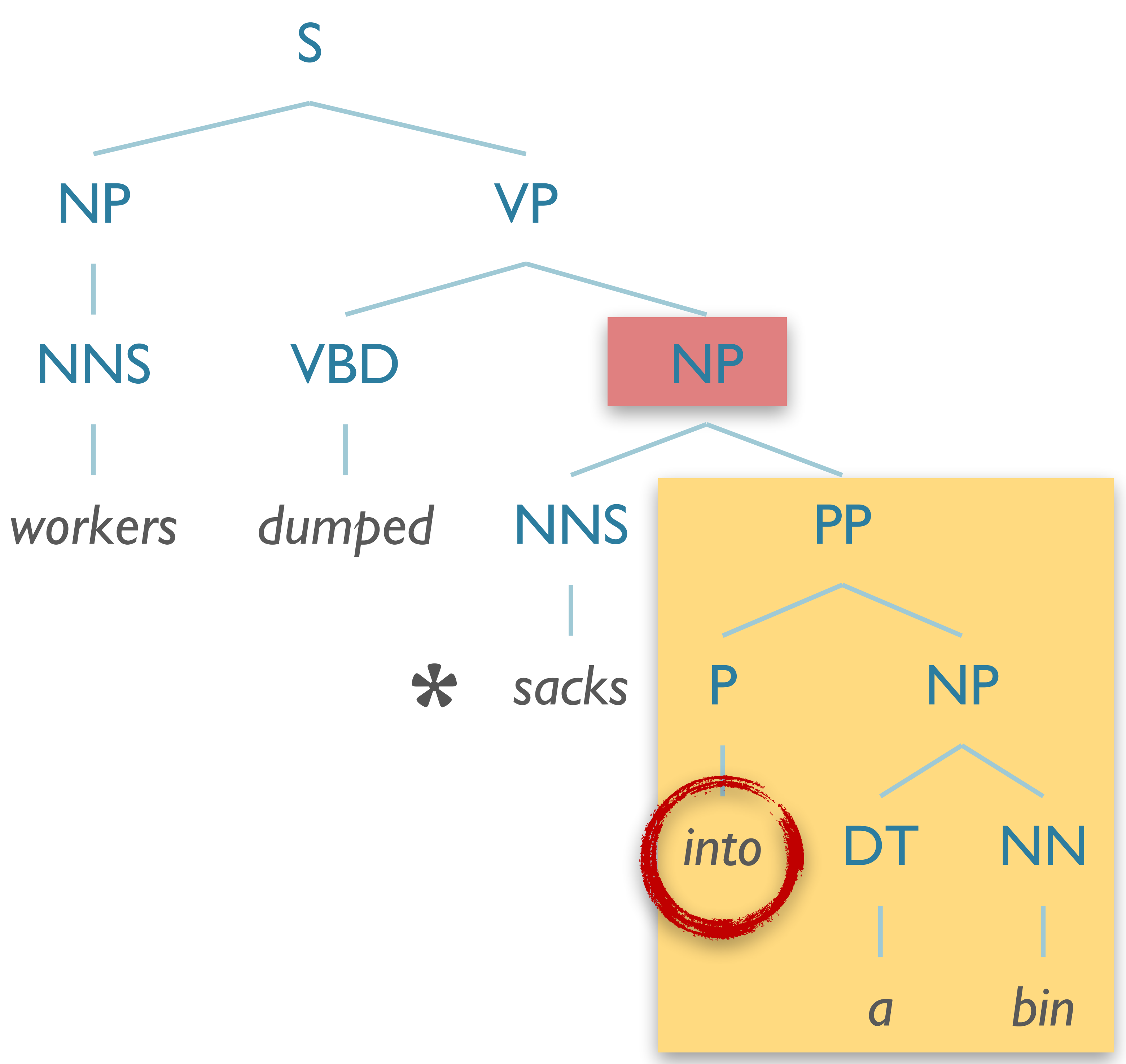
("into a bin" = location of sacks after dumping)
OK!



("into a bin" = *the sacks which were located ***in PP***)
not OK

Issues with PCFGs: Lexical Conditioning





("*in* a bin" = location of sacks ***before*** dumping)
OK!

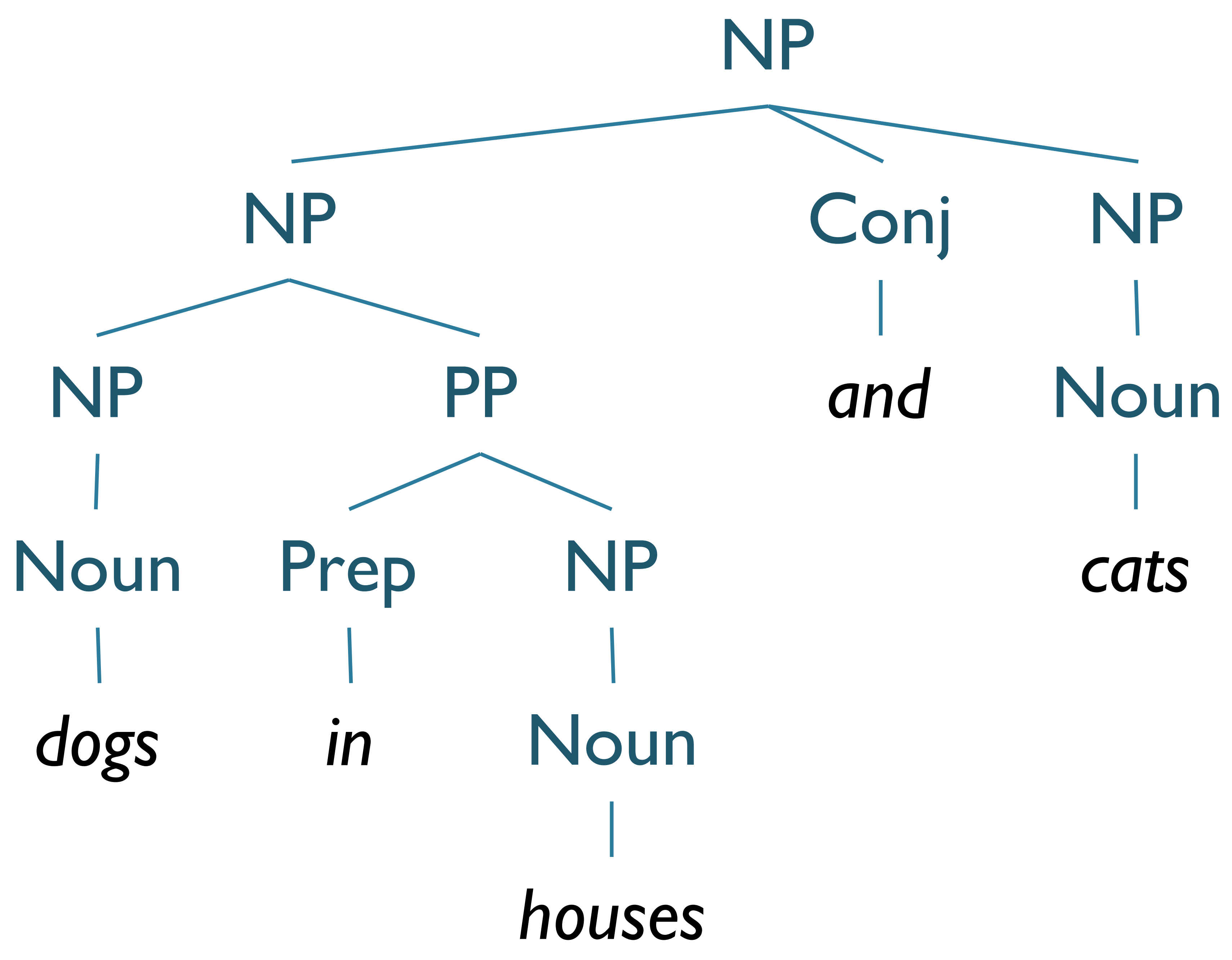
("***into*** a bin" = *the sacks which were located ***in PP***)
not OK

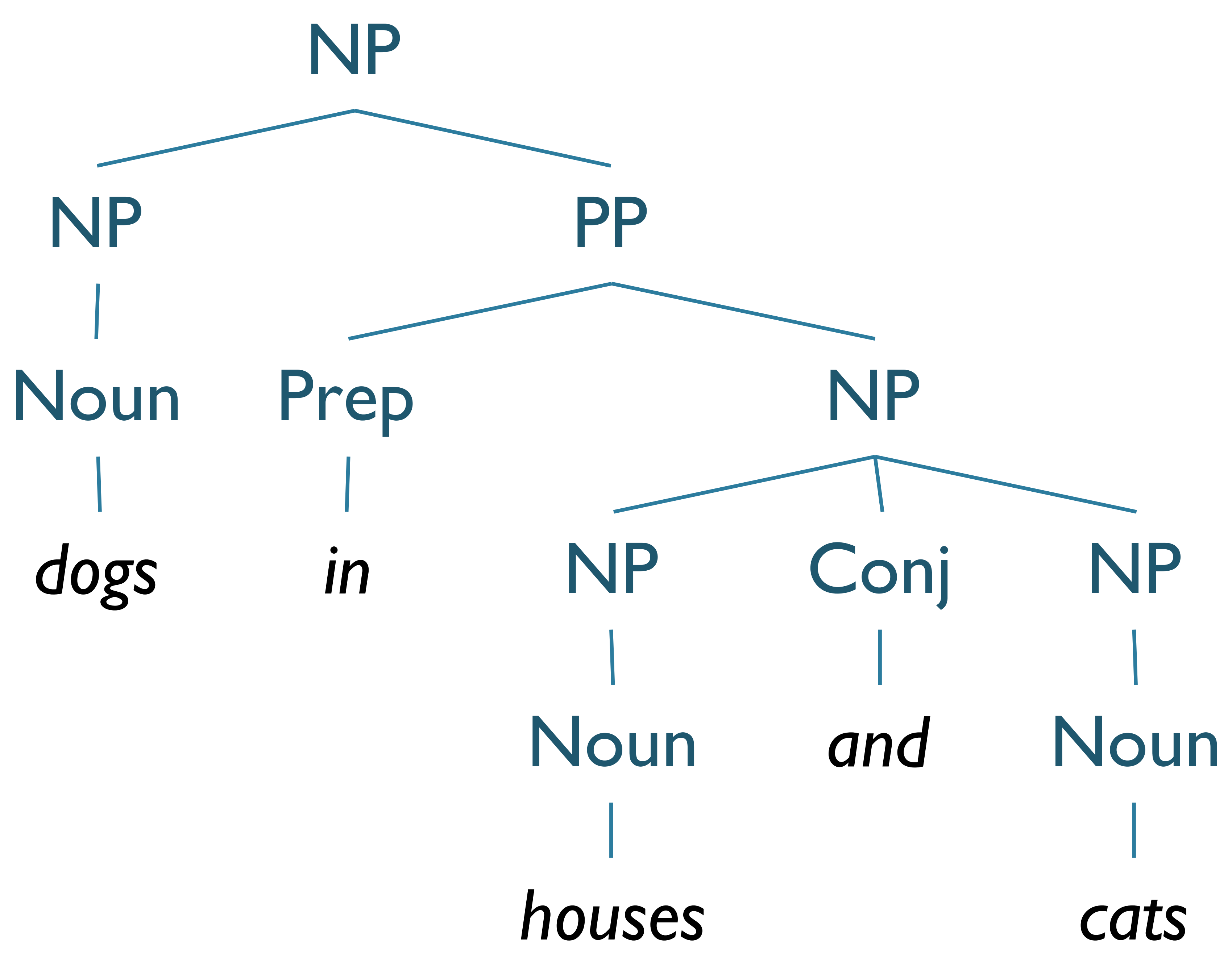
Issues with PCFGs: Lexical Conditioning

- *workers dumped sacks into a bin*
  - ***into*** should **prefer** modifying ***dumped***
  - ***into*** should **disprefer** modifying ***sacks***

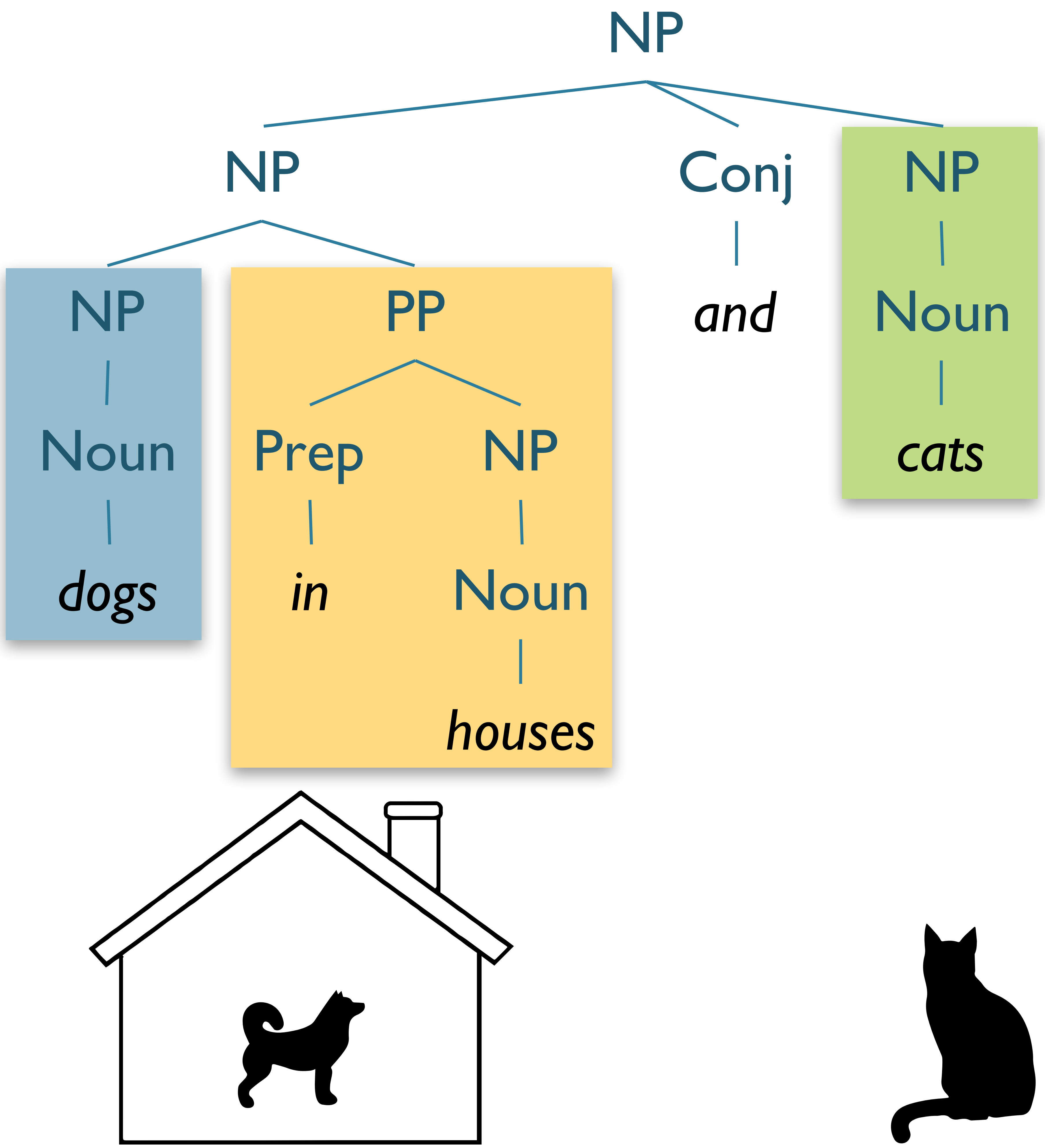
- *fishermen caught tons of herring*
  - ***of*** should **prefer** modifying ***tons***
  - ***of*** should **disprefer** modifying ***caught***

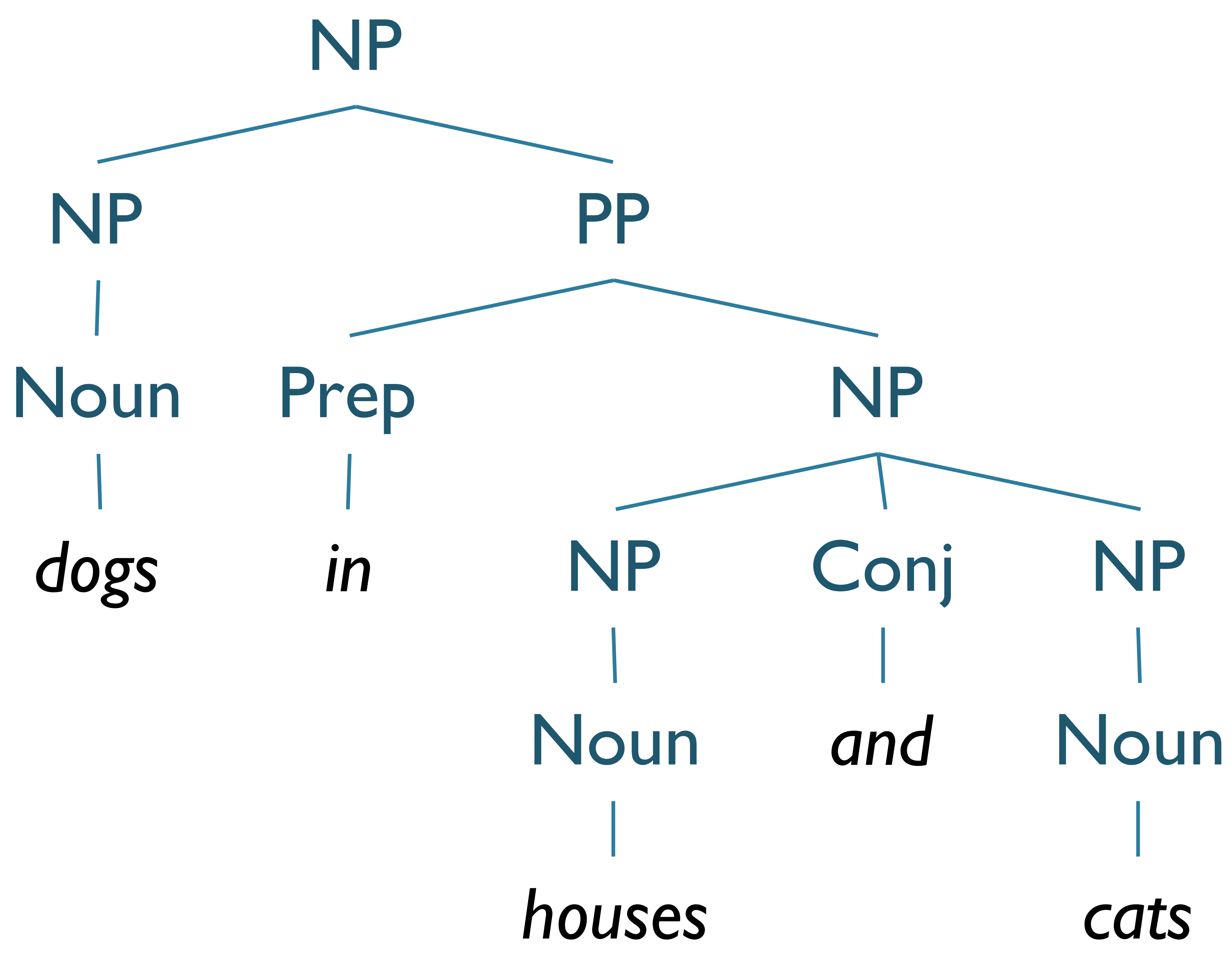
Issues with PCFGs: Coordination Ambiguity



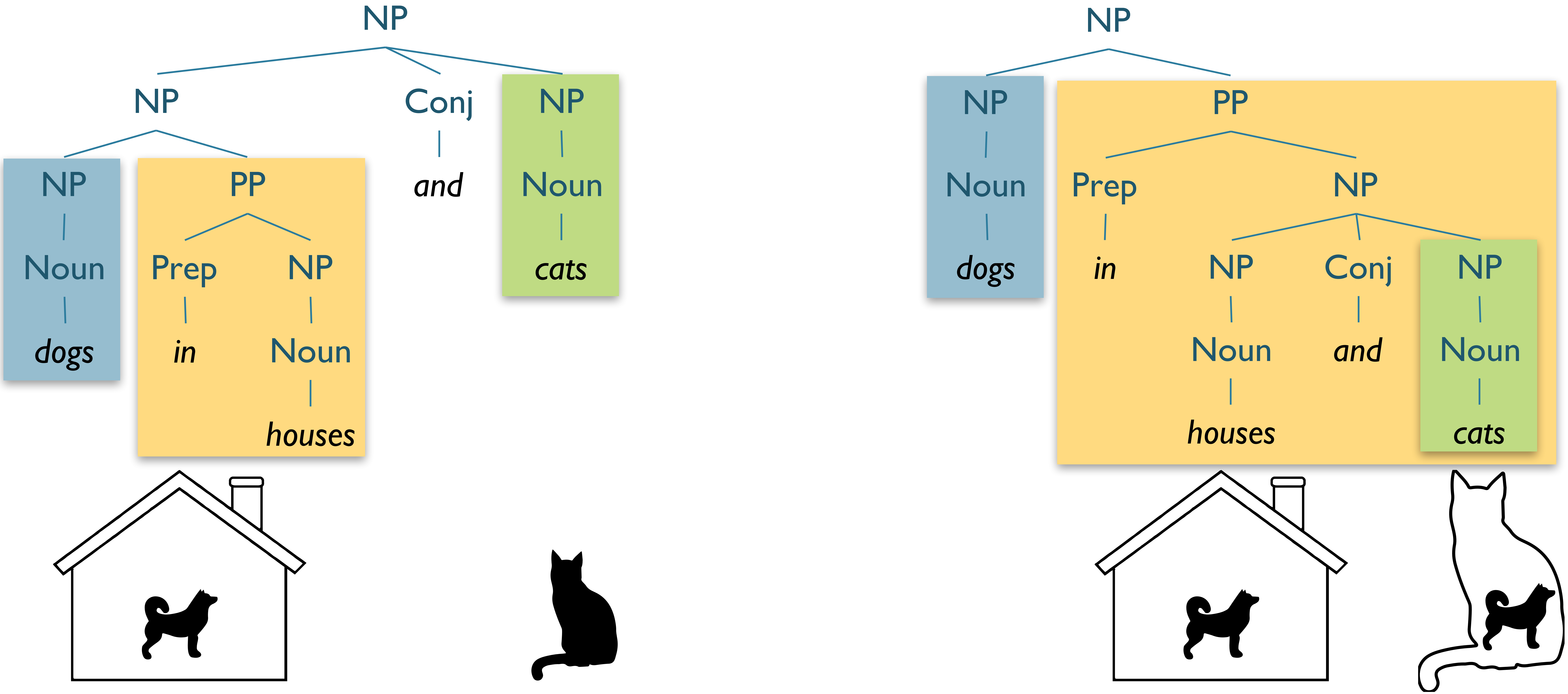


Issues with PCFGs: Coordination Ambiguity

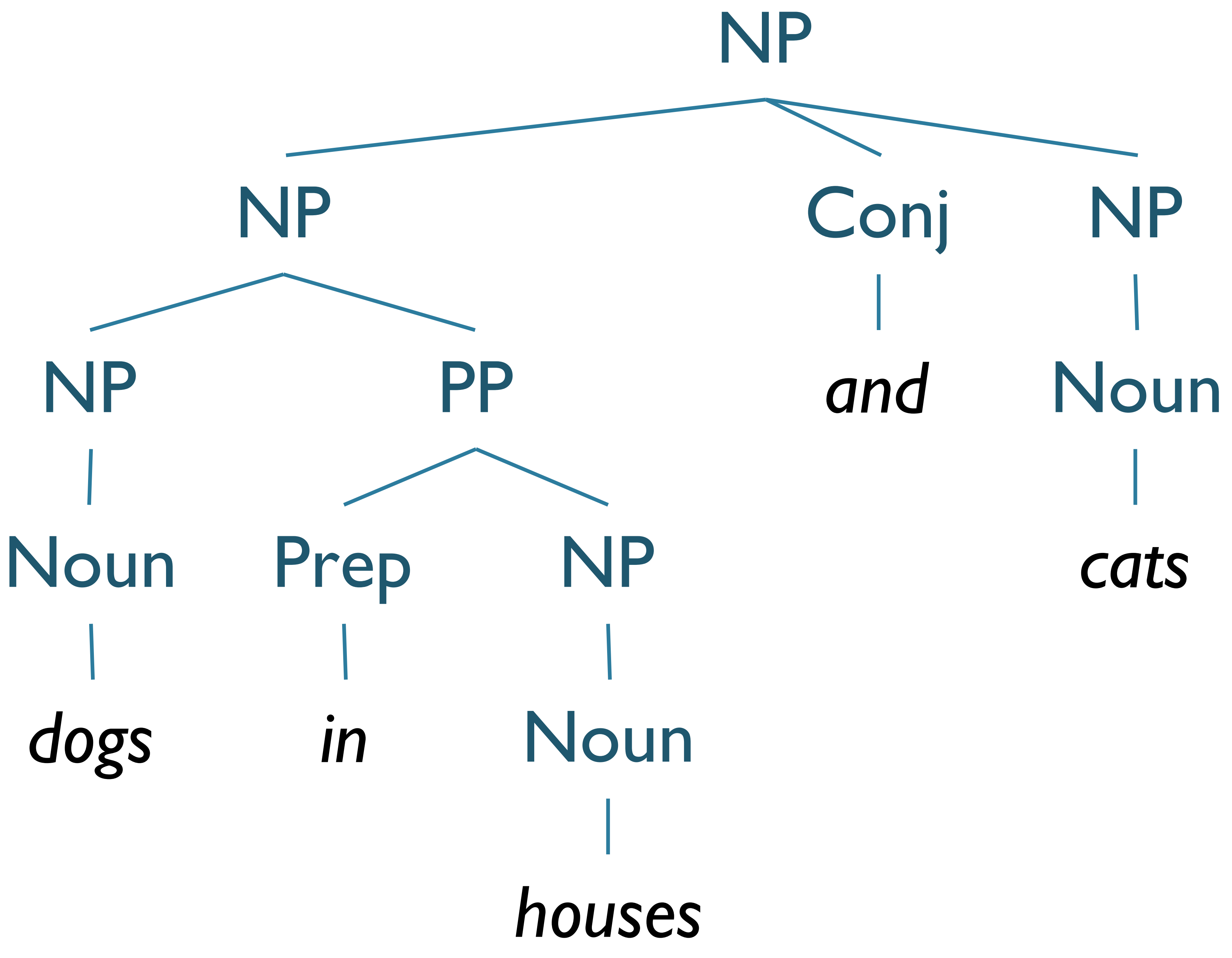


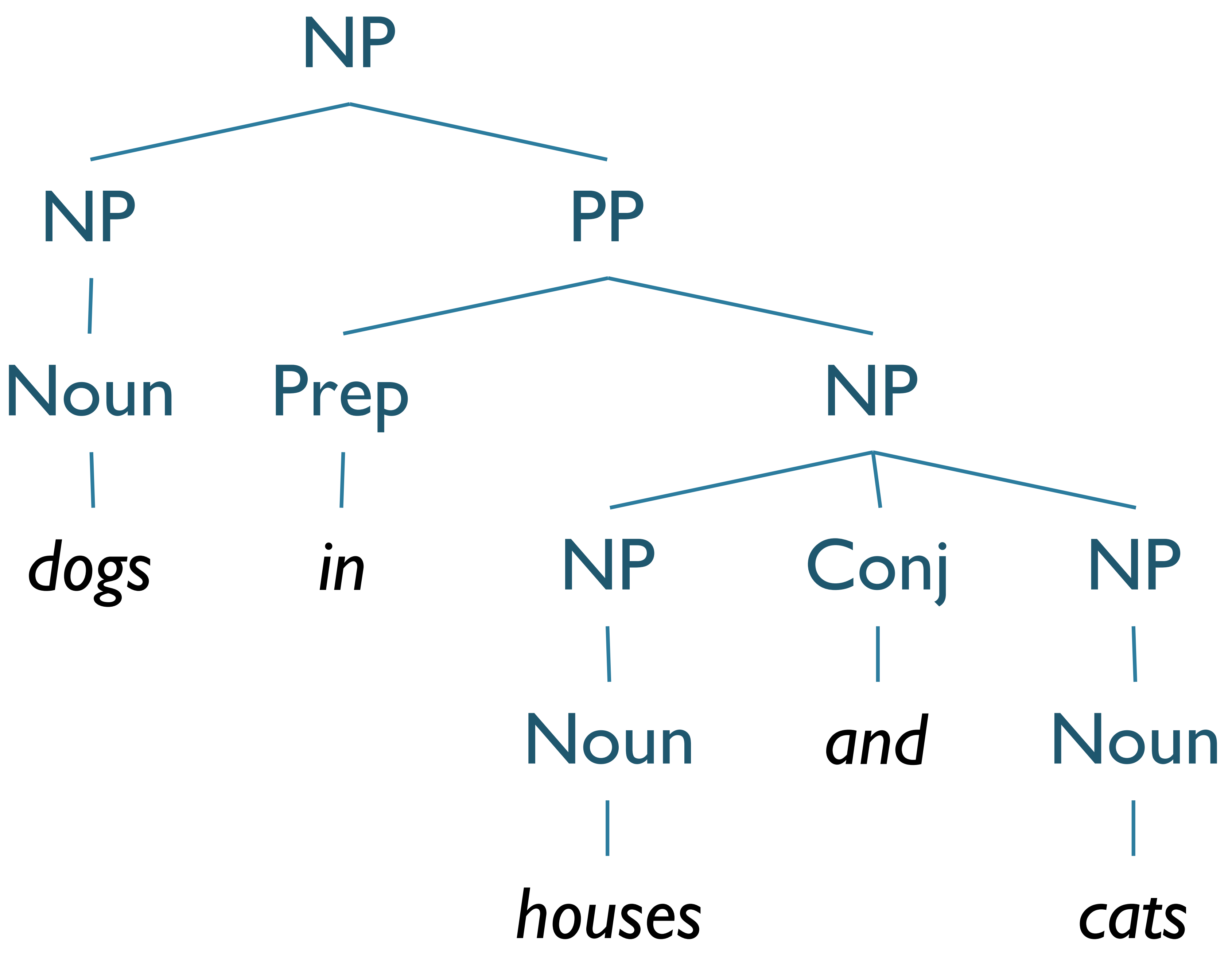


Issues with PCFGs: Coordination Ambiguity



Issues with PCFGs: Coordination Ambiguity



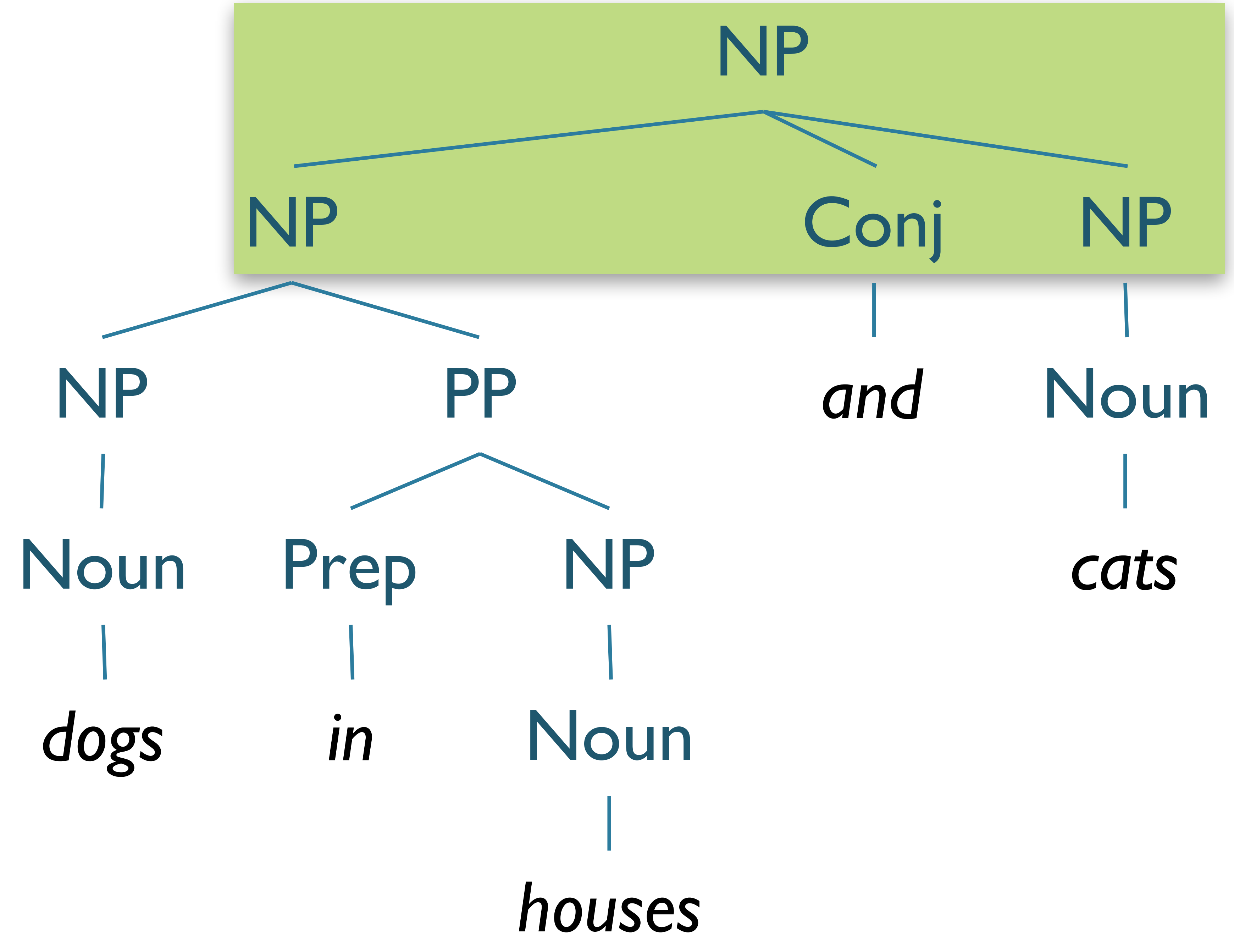


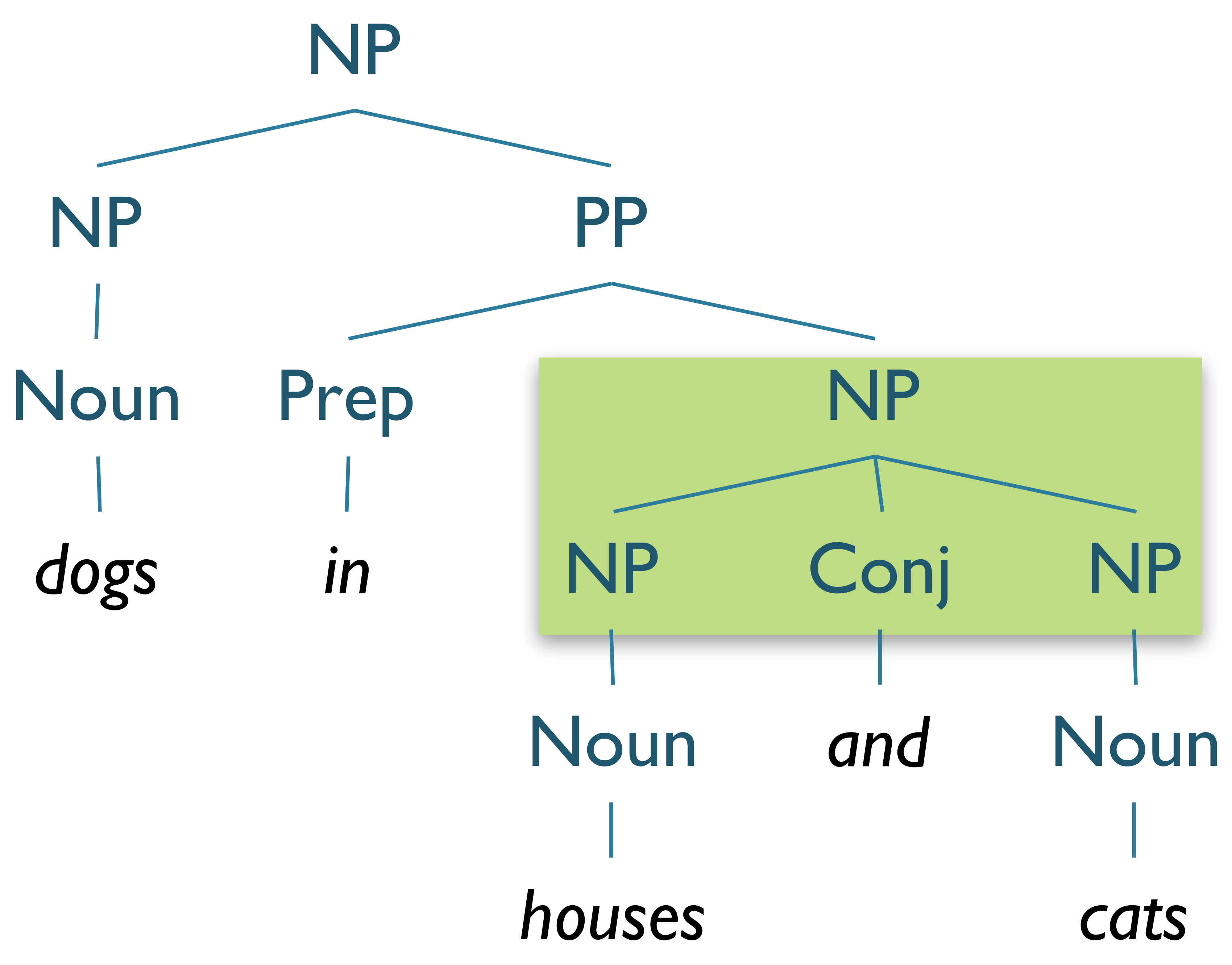
NP → NP Conj NP
NP → NP PP
Noun → "dogs"
PP → Prep NP
Prep → "in"
NP → Noun
Noun → "houses"
Conj → "and"
NP → Noun
Noun → "cats"

Same Rules!

NP → NP PP
Noun → "dogs"
PP → Prep NP
Prep → "in"
NP → NP Conj NP
NP → Noun
Noun → "houses"
Conj → "and"
NP → Noun
Noun → "cats"

Issues with PCFGs: Coordination Ambiguity



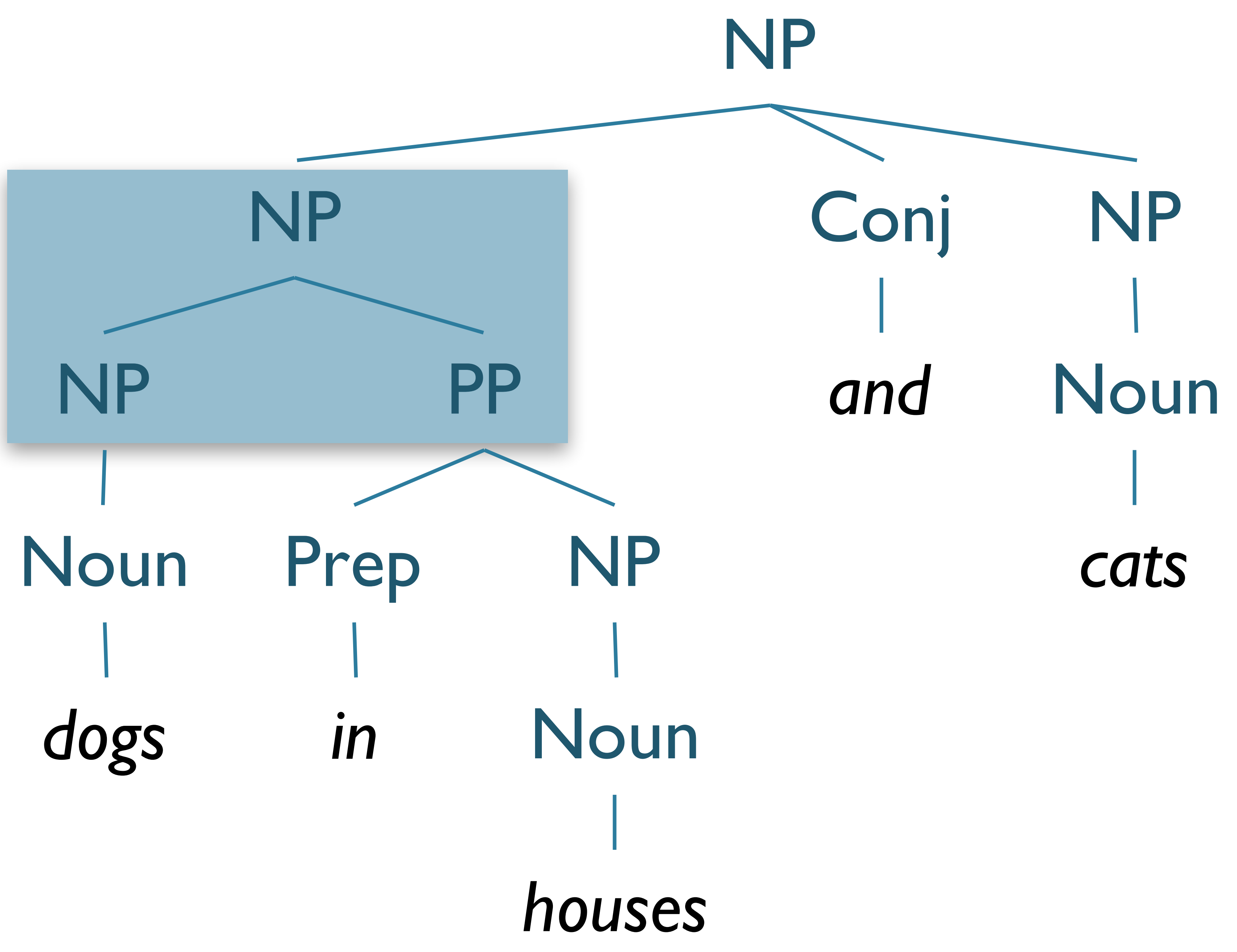


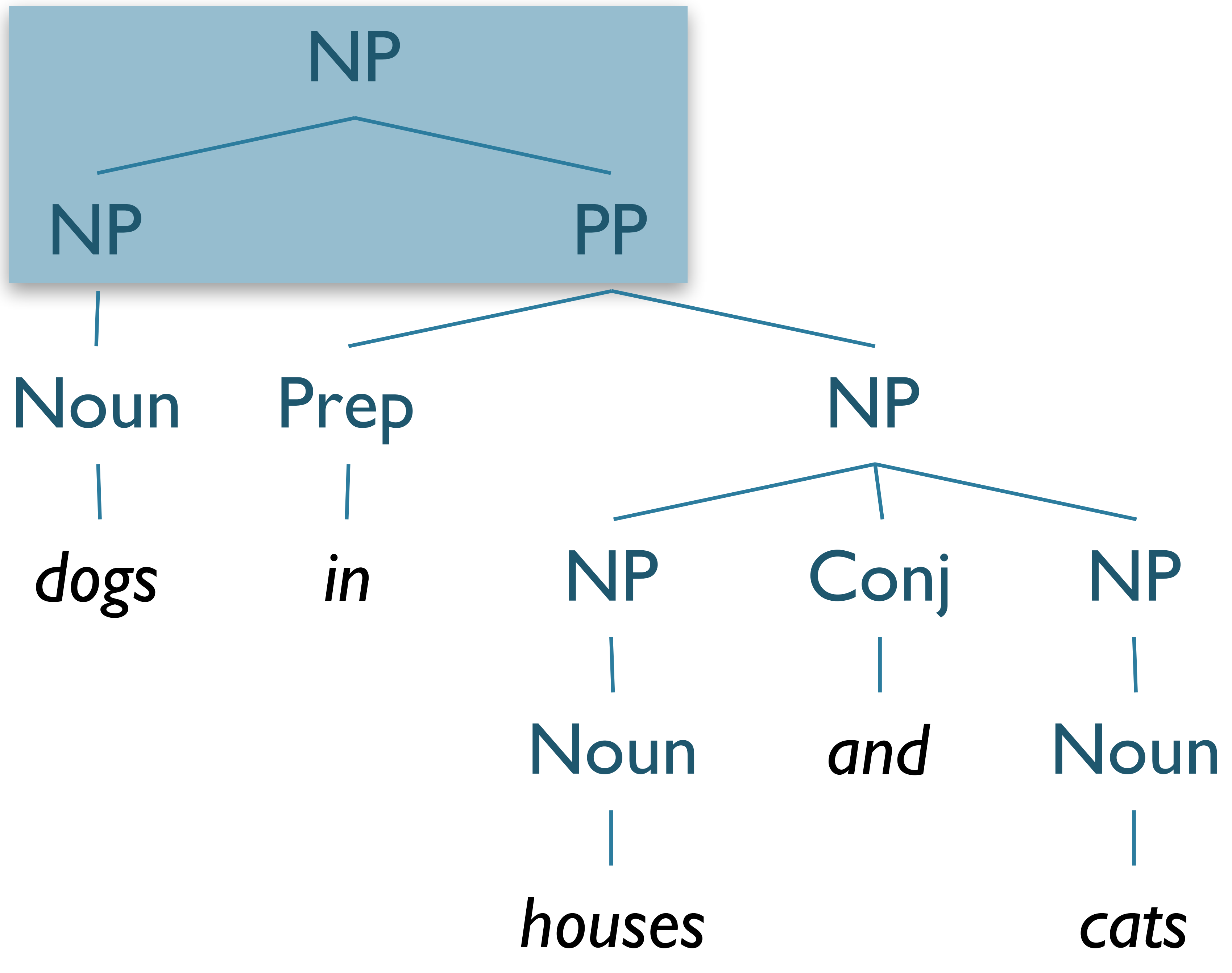
NP → NP Conj NP
NP → NP PP
Noun → "dogs"
PP → Prep NP
Prep → "in"
NP → Noun
Noun → "houses"
Conj → "and"
NP → Noun
Noun → "cats"

Same Rules!

NP → NP PP
Noun → "dogs"
PP → Prep NP
Prep → "in"
NP → NP Conj NP
NP → Noun
Noun → "houses"
Conj → "and"
NP → Noun
Noun → "cats"

Issues with PCFGs: Coordination Ambiguity





NP → NP Conj NP
NP → NP PP
Noun → "dogs"
PP → Prep NP
Prep → "in"
NP → Noun
Noun → "houses"
Conj → "and"
NP → Noun
Noun → "cats"

Same Rules!

NP → NP PP
Noun → "dogs"
PP → Prep NP
Prep → "in"
NP → NP Conj NP
NP → Noun
Noun → "houses"
Conj → "and"
NP → Noun
Noun → "cats"

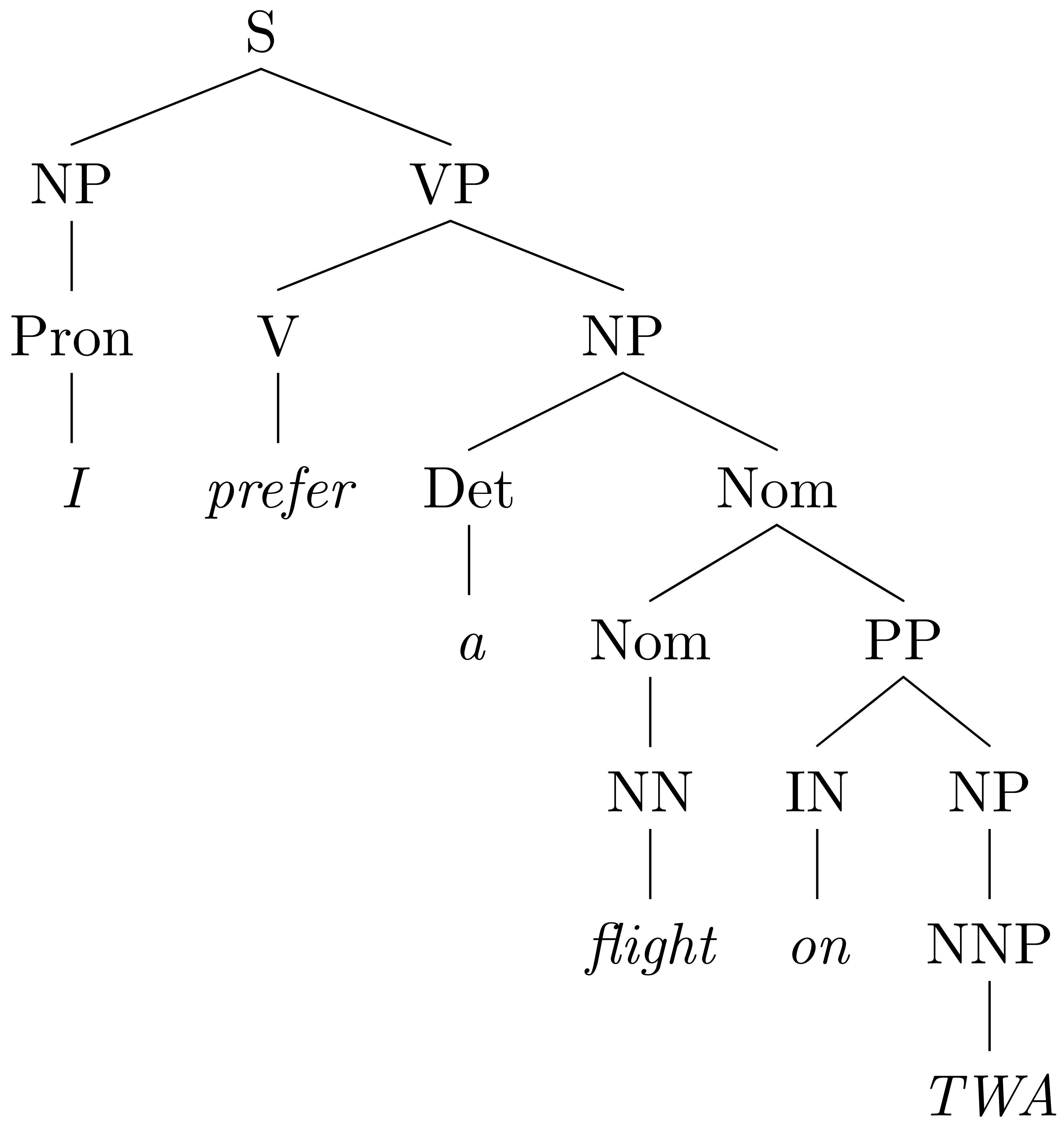
Improving PCFGs

Improving PCFGs

- **Parent Annotation**
- Lexicalization
- Reranking

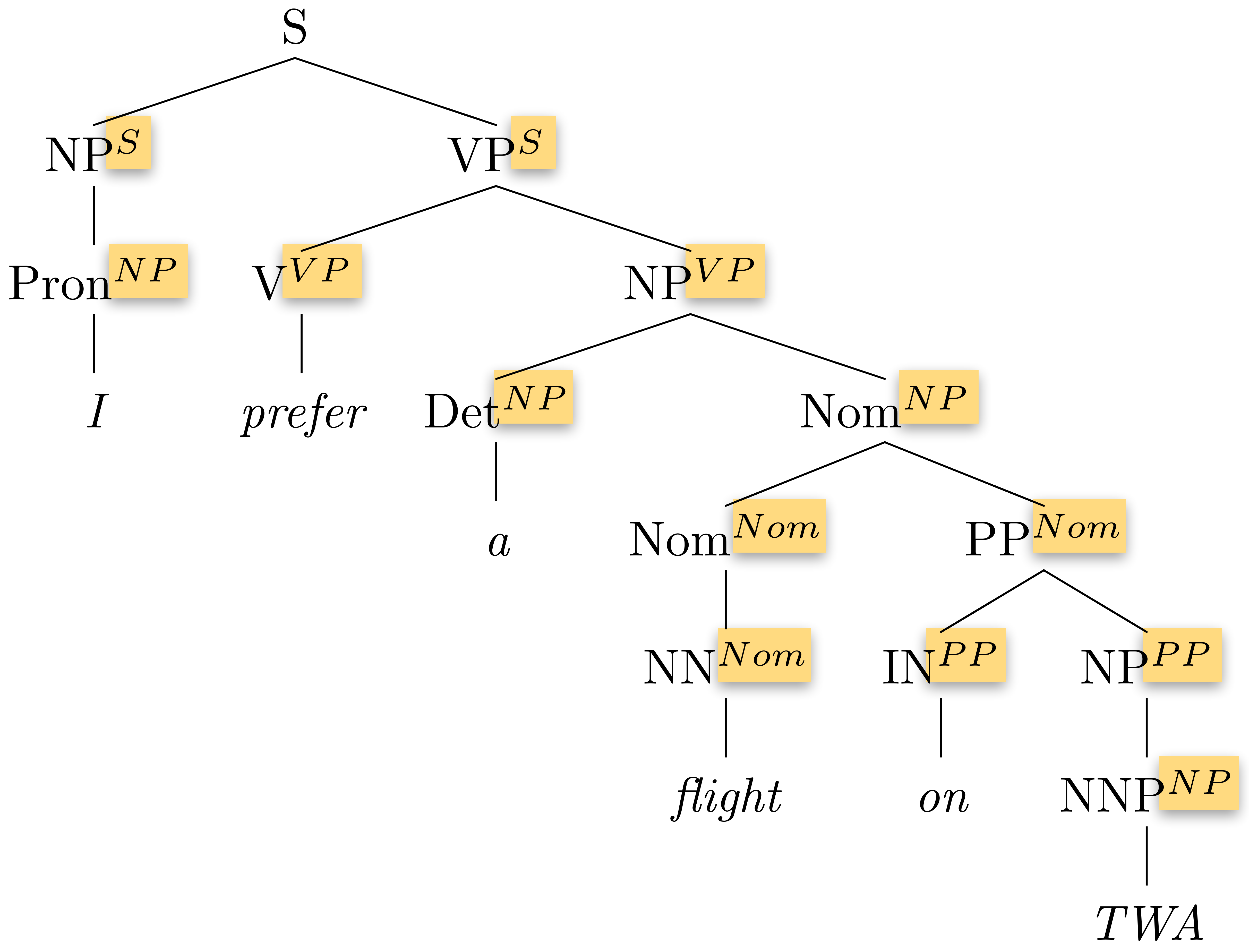
Improving PCFGs: Parent Annotation

- To handle the $NP \rightarrow PRP$ $[0.91 \textbf{ if } NP_{\Theta=subject} \textbf{ else } 0.34]$



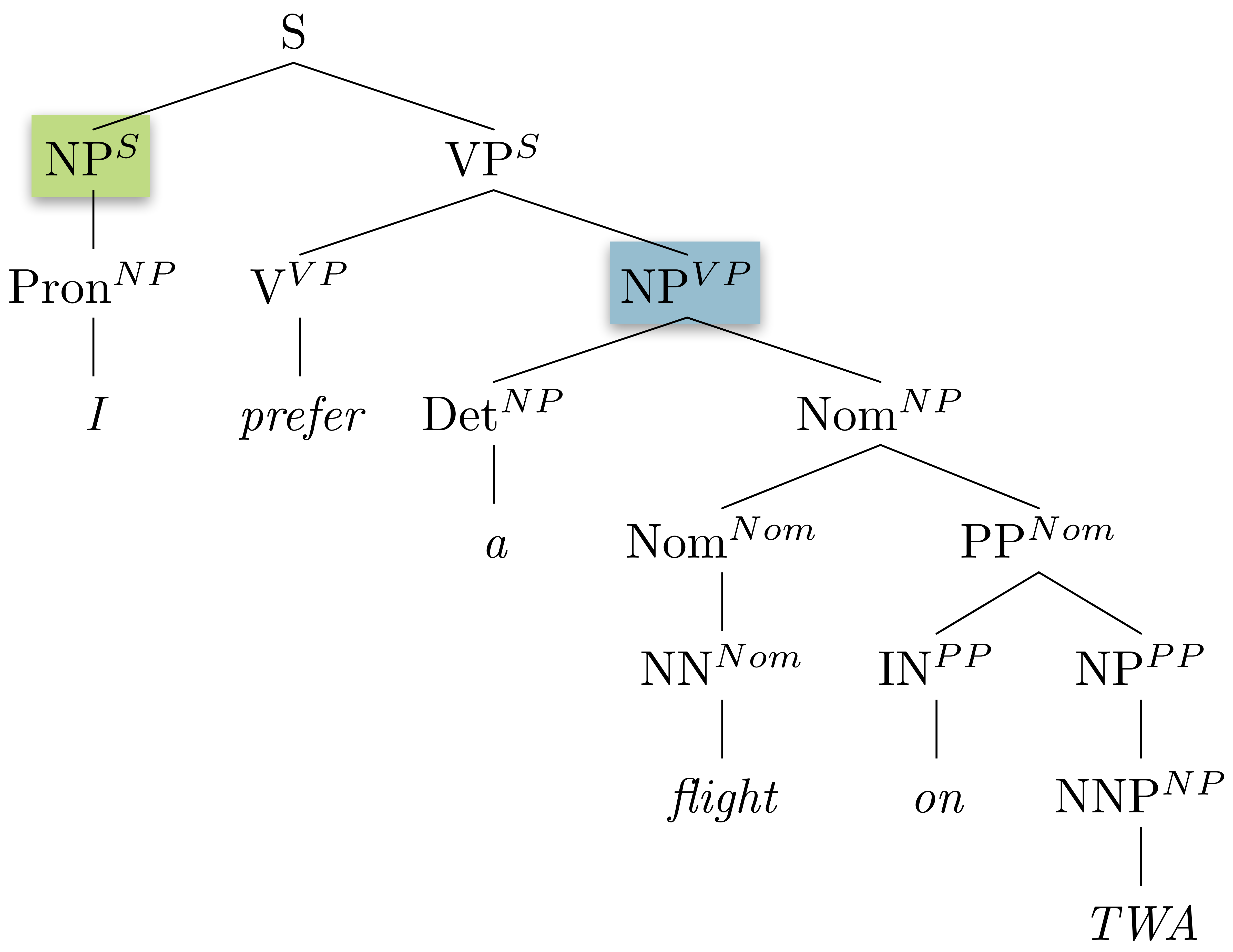
Improving PCFGs: Parent Annotation

- To handle the $NP \rightarrow PRP$ $[0.91$ **if** $NP_{\Theta=subject}$ **else** $0.34]$



Improving PCFGs: Parent Annotation

- To handle the $NP \rightarrow PRP$ $\color{red}{[0.91 \textbf{ if } NP_{\Theta=subject} \textbf{ else } 0.34]}$



Improving PCFGs: Parent Annotation

- Advantages:
  - Captures structural dependencies in grammar

Improving PCFGs: Parent Annotation

- Advantages:
  - Captures structural dependencies in grammar
- Disadvantages:
  - Explodes number of rules in grammar
    - Same problem with subcategorization
  - Results in sparsity problems

Improving PCFGs: Parent Annotation

- Advantages:
  - Captures structural dependencies in grammar
- Disadvantages:
  - Explodes number of rules in grammar
    - Same problem with subcategorization
  - Results in sparsity problems
- Strategies to find an optimal number of splits
  - Petrov et al (2006)

Improving PCFGs

- Parent Annotation
- **Lexicalization**
- Reranking

Improving PCFGs: Lexical "Heads"

- Remember back to syntax intro (Lecture #1)
  - Phrases are "headed" by key words
    - **VP** are headed by **V**
    - **NP** by **NN, NNS, PRON**
    - **PP** by **PREP**
- We can take advantage of this in our grammar!

Improving PCFGs: Lexical Dependencies

- As we've seen, some rules should be conditioned on certain words
- **Proposal**: annotate nonterminals with lexical head

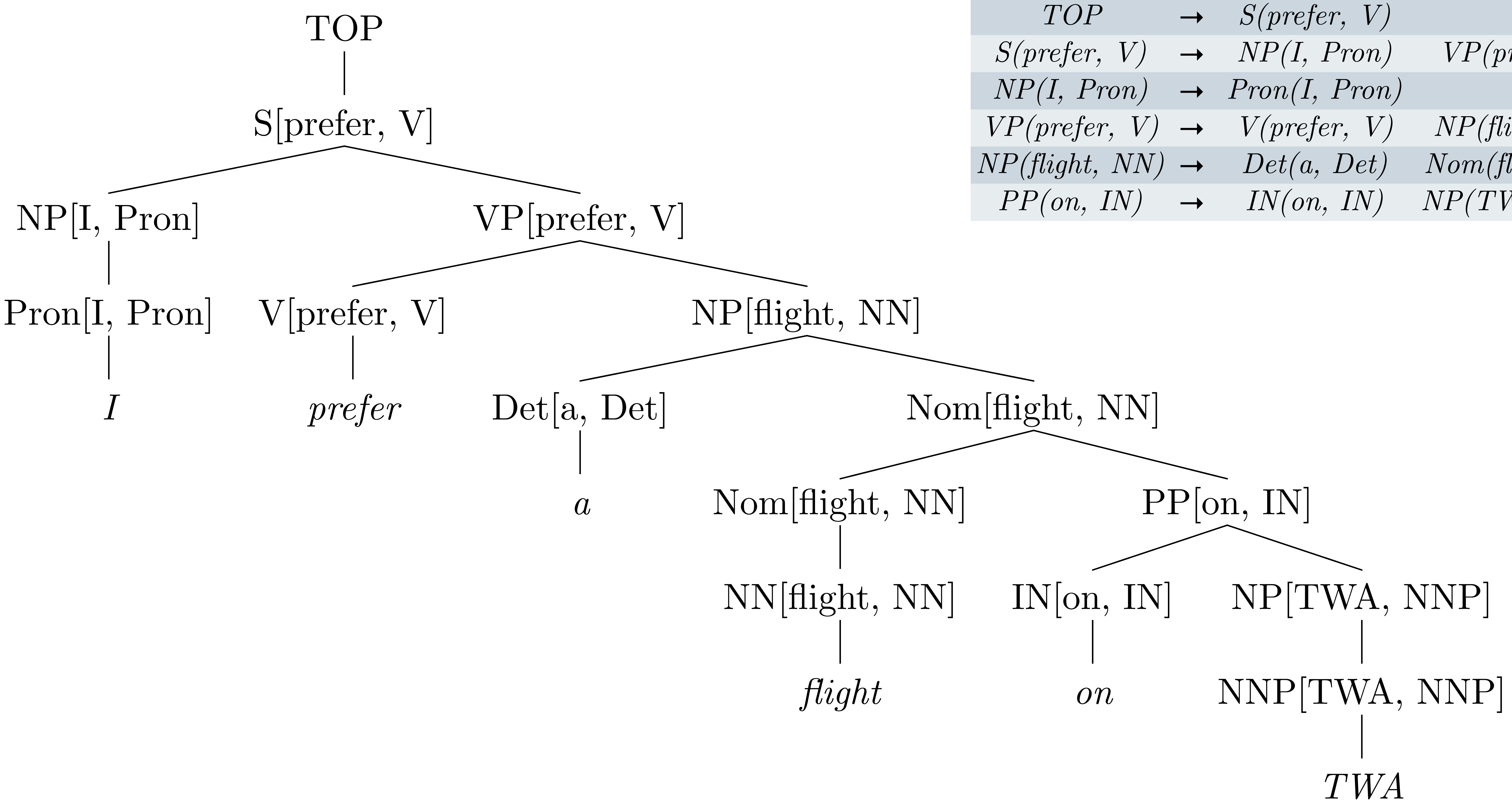
$$VP \rightarrow VBD\ NP\ PP$$

$$VP(\boldsymbol{dumped}) \rightarrow VBD(\boldsymbol{dumped})\ NP(\boldsymbol{sacks})\ PP(\boldsymbol{into})$$

- **Additionally**: annotate with lexical head + POS

$$VP(dumped,\ \boldsymbol{VBD}) \rightarrow VBD(dumped,\ \boldsymbol{VBD})\ NP(sacks,\ \boldsymbol{NNS})\ PP(into,\ \boldsymbol{IN})$$

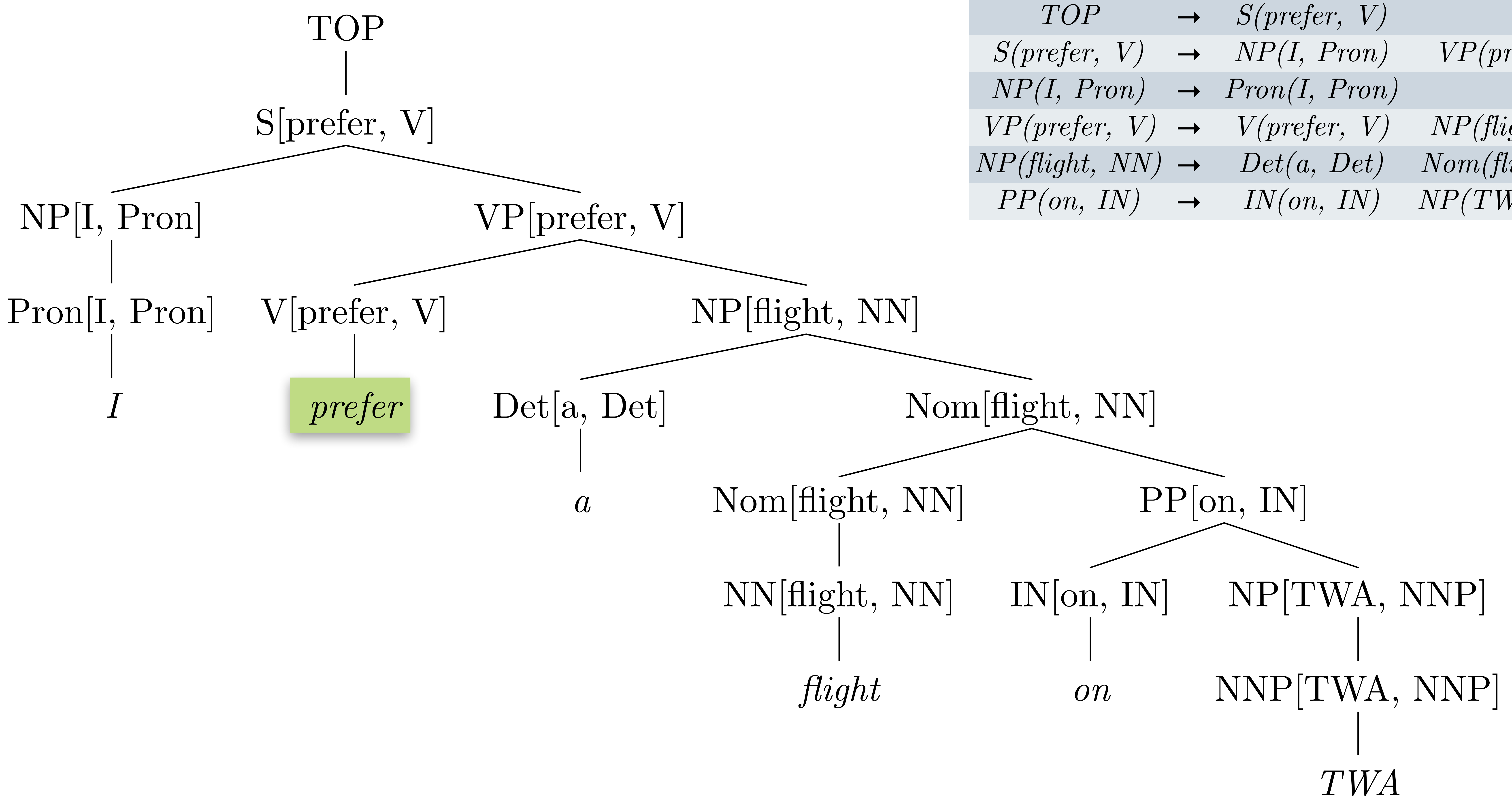
Lexicalized Parse Tree



Internal Rules			
TOP	→	*S(prefer, V)*	
S(prefer, V)	→	*NP(I, Pron)*	*VP(prefer, V)*
NP(I, Pron)	→	*Pron(I, Pron)*	
VP(prefer, V)	→	*V(prefer, V)*	*NP(flight, NN)*
NP(flight, NN)	→	*Det(a, Det)*	*Nom(flight, NN)*
PP(on, IN)	→	*IN(on, IN)*	*NP(TWA, NNP)*

Lexical Rules		
Pron(I, Pron)	→	I
V(prefer, V)	→	prefer
Det(a, Det)	→	a
NN(flight, NN)	→	flight
IN(on, IN)	→	on
NNP(NWA, NNP)	→	TWA

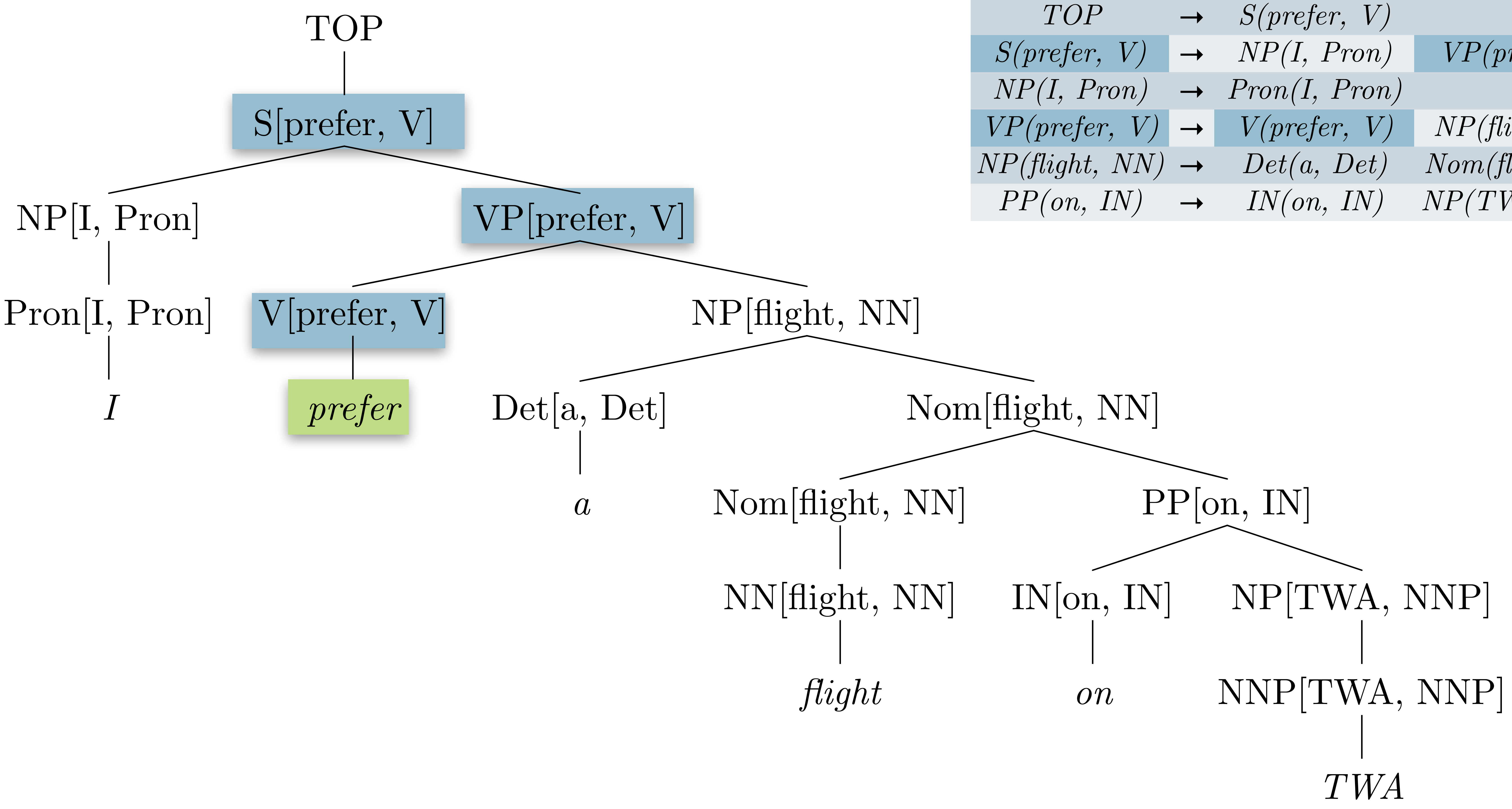
Lexicalized Parse Tree



Internal Rules			
TOP	→	*S(prefer, V)*	
S(prefer, V)	→	*NP(I, Pron)*	*VP(prefer, V)*
NP(I, Pron)	→	*Pron(I, Pron)*	
VP(prefer, V)	→	*V(prefer, V)*	*NP(flight, NN)*
NP(flight, NN)	→	*Det(a, Det)*	*Nom(flight, NN)*
PP(on, IN)	→	*IN(on, IN)*	*NP(TWA, NNP)*

Lexical Rules		
Pron(I, Pron)	→	I
V(prefer, V)	→	prefer
Det(a, Det)	→	a
NN(flight, NN)	→	flight
IN(on, IN)	→	on
NNP(NWA, NNP)	→	TWA

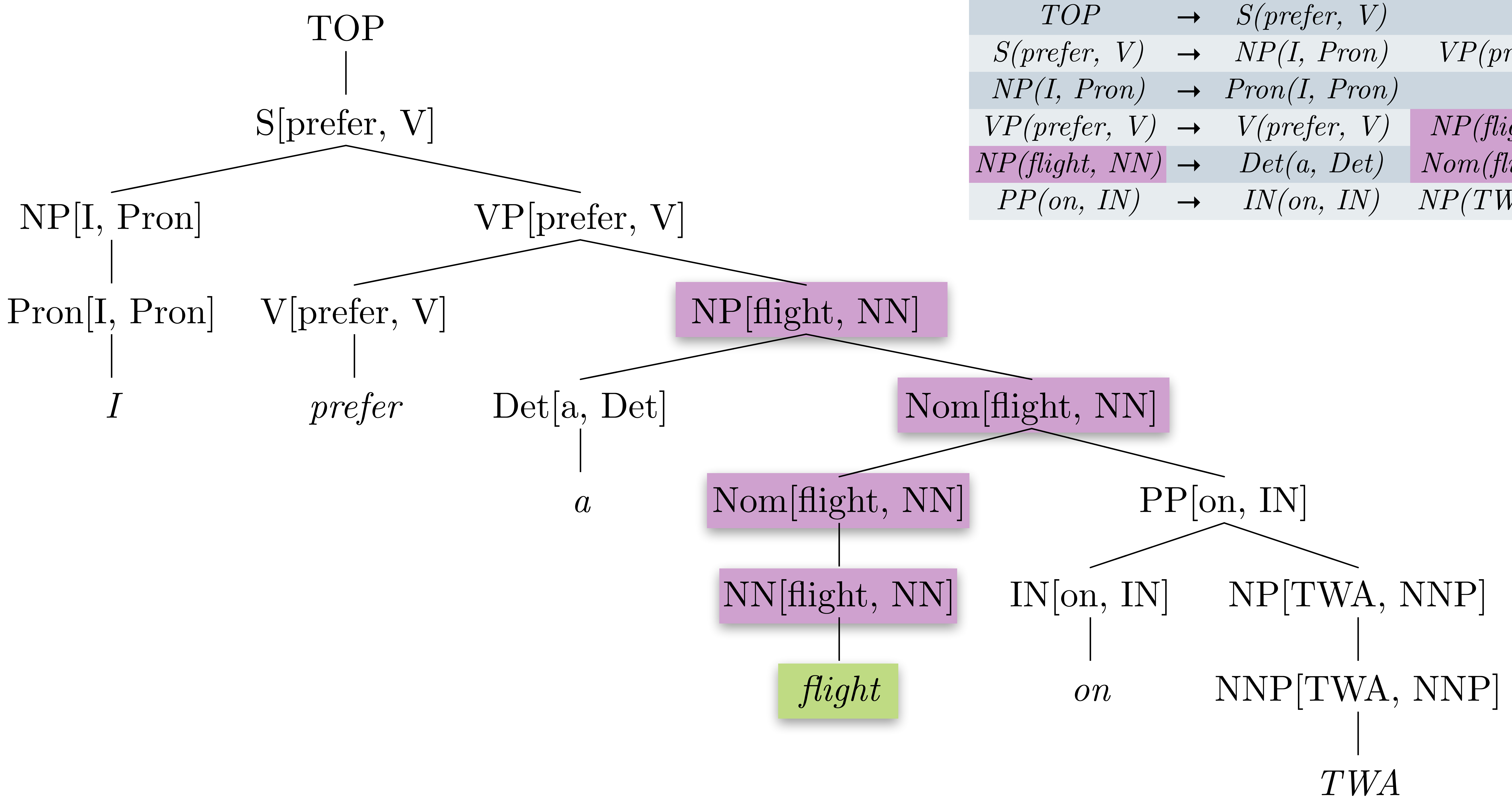
Lexicalized Parse Tree



Internal Rules			
TOP	→	*S(prefer, V)*	
S(prefer, V)	→	*NP(I, Pron)*	*VP(prefer, V)*
NP(I, Pron)	→	*Pron(I, Pron)*	
VP(prefer, V)	→	*V(prefer, V)*	*NP(flight, NN)*
NP(flight, NN)	→	*Det(a, Det)*	*Nom(flight, NN)*
PP(on, IN)	→	*IN(on, IN)*	*NP(TWA, NNP)*

Lexical Rules		
Pron(I, Pron)	→	I
V(prefer, V)	→	prefer
Det(a, Det)	→	a
NN(flight, NN)	→	flight
IN(on, IN)	→	on
NNP(NWA, NNP)	→	TWA

Lexicalized Parse Tree



Internal Rules			
TOP	→	*S(prefer, V)*	
S(prefer, V)	→	*NP(I, Pron)*	*VP(prefer, V)*
NP(I, Pron)	→	*Pron(I, Pron)*	
VP(prefer, V)	→	*V(prefer, V)*	*NP(flight, NN)*
NP(flight, NN)	→	*Det(a, Det)*	*Nom(flight, NN)*
PP(on, IN)	→	*IN(on, IN)*	*NP(TWA, NNP)*

Lexical Rules		
Pron(I, Pron)	→	I
V(prefer, V)	→	prefer
Det(a, Det)	→	a
NN(flight, NN)	→	flight
IN(on, IN)	→	on
NNP(NWA, NNP)	→	TWA

Improving PCFGs: Lexical Dependencies

- Upshot: heads propagate up tree:

Improving PCFGs: Lexical Dependencies

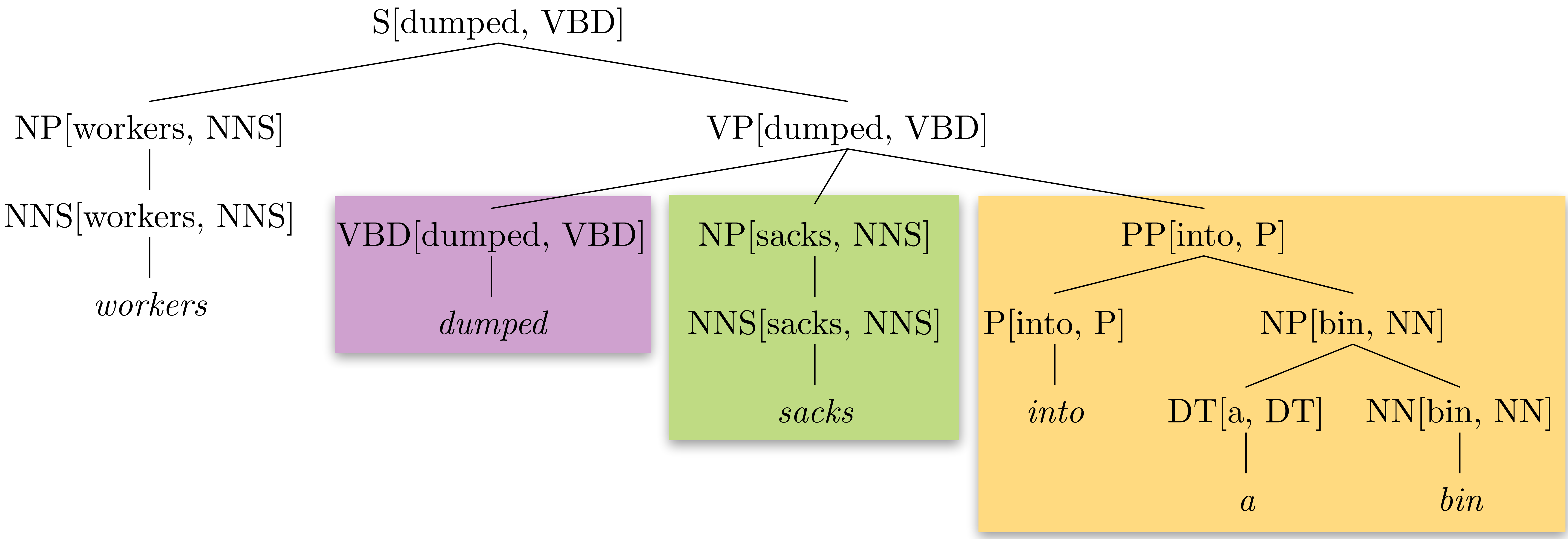
- Upshot: heads propagate up tree:
  - $VP \rightarrow VBD(dumped,\ VBD)\ NP(sacks,\ NNS)\ PP(into,\ P)$
  - $NP \rightarrow NNS(sacks,\ NNS)\ PP(into,\ P)$

Improving PCFGs: Lexical Dependencies

- Upshot: heads propagate up tree:
  - $VP \rightarrow VBD(dumped, VBD)\ NP(sacks, NNS)\ PP(into, P)$ ✔
  - $NP \rightarrow NNS(sacks, NNS)\ PP(into, P)$ ✘

Improving PCFGs: Lexical Dependencies

- Upshot: heads propagate up tree:
  - $VP \rightarrow VBD(dumped,\ VBD)\ NP(sacks,\ NNS)\ PP(into,\ P)$
  - $NP \rightarrow NNS(sacks,\ NNS)\ PP(into,\ P)$



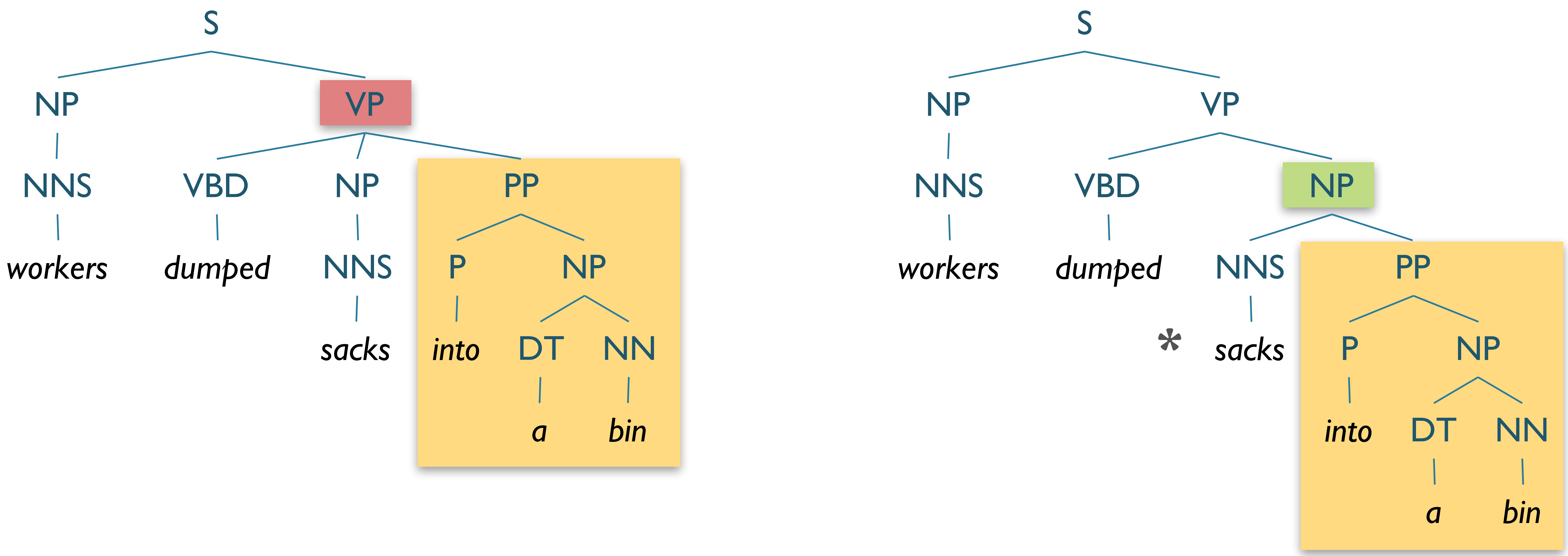
Improving PCFGs: Lexical Dependencies

- Downside:
  - Rules far too specialized — will be sparse
- Solution:
  - Assume ***conditional*** independence
  - Create more rules

Improving PCFGs: Collins Parser

- Proposal:
  - $\boldsymbol{LHS} \rightarrow \boldsymbol{LeftOfHead} \ldots \quad \boldsymbol{Head} \quad \ldots \boldsymbol{RightOfHead}$
  - Instead of calculating *P*(*EntireRule*), which is sparse:
  - Calculate:
    - Probability that $\boldsymbol{LHS}$ has nonterminal phrase $\boldsymbol{H}$ given head-word $hw \ldots$
    - × Probability of modifiers to the **left** given head-word $hw \ldots$
    - × Probability of modifiers to the **right** given head-word $hw \ldots$

Collins Parser Example



Collins Parser Example

$P(VP \rightarrow VBD\ NP\ PP \mid VP, dumped)$

Collins Parser Example

$$P(VP \rightarrow VBD\ NP\ PP \mid VP,\ dumped)$$

$$= \frac{Count\left(VP\left(dumped\right) \rightarrow VBD\ NP\ PP\right)}{\sum_{\beta} Count\left(VP\left(dumped\right) \rightarrow \beta\right)}$$

Collins Parser Example

$$P(VP \rightarrow VBD\ NP\ PP \mid VP,\ dumped)$$

$$= \frac{Count\left(VP\left(dumped\right) \rightarrow VBD\ NP\ PP\right)}{\sum_{\beta} Count\left(VP\left(dumped\right) \rightarrow \beta\right)}$$

$$= \frac{6}{9} = 0.67$$

Collins Parser Example

$$P(VP \rightarrow VBD\ NP\ PP \mid VP,\ dumped)$$

$$= \frac{Count\left(VP\left(dumped\right) \rightarrow VBD\ NP\ PP\right)}{\sum_{\beta} Count\left(VP\left(dumped\right) \rightarrow \beta\right)}$$

$$= \frac{6}{9} = 0.67$$

$$P_R(into \mid PP, dumped)$$

Collins Parser Example

$$P(VP \to VBD\ NP\ PP \mid VP, dumped)$$

$$= \frac{Count\left(VP\left(dumped\right) \to VBD\ NP\ PP\right)}{\sum_{\beta} Count\left(VP\left(dumped\right) \to \beta\right)}$$

$$= \frac{6}{9} = 0.67$$

$$P_R(into \mid PP, dumped)$$

$$= \frac{Count\left(X\left(dumped\right) \to \ldots\ PP\left(into\right)\ \ldots\right)}{\sum_{\beta} Count\left(X\left(dumped\right) \to \ldots\ PP\ \ldots\right)}$$

Collins Parser Example

$$P(VP \rightarrow VBD\ NP\ PP \mid VP,\ dumped)$$

$$= \frac{Count\left(VP\left(dumped\right) \rightarrow VBD\ NP\ PP\right)}{\sum_{\beta} Count\left(VP\left(dumped\right) \rightarrow \beta\right)}$$

$$= \frac{6}{9} = 0.67$$

$$P_R(into \mid PP, dumped)$$

$$= \frac{Count\left(X\left(dumped\right) \rightarrow \ldots\ PP\left(into\right)\ \ldots\right)}{\sum_{\beta} Count\left(X\left(dumped\right) \rightarrow \ldots\ PP\ \ldots\right)}$$

$$= \frac{2}{9} = 0.22$$

Collins Parser Example

$$P(VP \rightarrow VBD\ NP\ PP \,|\, VP, dumped)$$

$$= \frac{Count\left(VP\left(dumped\right) \rightarrow VBD\ NP\ PP\right)}{\sum_\beta Count\left(VP\left(dumped\right) \rightarrow \beta\right)}$$

$$= \frac{6}{9} = 0.67$$

$$P(VP \rightarrow VBD\ NP \,|\, VP, dumped)$$

$$= \frac{Count\left(VP\left(dumped\right) \rightarrow VBD\ NP\right)}{\sum_\beta Count\left(VP\left(dumped\right) \rightarrow \beta\right)}$$

$$= \frac{1}{9} = 0.11$$

$$P_R(into \,|\, PP, dumped)$$

$$= \frac{Count\left(X\left(dumped\right) \rightarrow \ldots\ PP\left(into\right)\ \ldots\right)}{\sum_\beta Count\left(X\left(dumped\right) \rightarrow \ldots\ PP\ \ldots\right)}$$

$$= \frac{2}{9} = 0.22$$

Collins Parser Example

$$P(VP \rightarrow VBD\ NP\ PP \mid VP, dumped)$$

$$= \frac{Count\left(VP\left(dumped\right) \rightarrow VBD\ NP\ PP\right)}{\sum_{\beta} Count\left(VP\left(dumped\right) \rightarrow \beta\right)}$$

$$= \frac{6}{9} = 0.67$$

$$P(VP \rightarrow VBD\ NP \mid VP, dumped)$$

$$= \frac{Count\left(VP\left(dumped\right) \rightarrow VBD\ NP\right)}{\sum_{\beta} Count\left(VP\left(dumped\right) \rightarrow \beta\right)}$$

$$= \frac{1}{9} = 0.11$$

$$P_R(into \mid PP, dumped)$$

$$= \frac{Count\left(X\left(dumped\right) \rightarrow \ldots\ PP\left(into\right)\ \ldots\right)}{\sum_{\beta} Count\left(X\left(dumped\right) \rightarrow \ldots\ PP\ \ldots\right)}$$

$$= \frac{2}{9} = 0.22$$

$$P_R(into \mid PP, sacks)$$

$$= \frac{Count\left(X\left(sacks\right) \rightarrow \ldots\ PP\left(into\right)\ \ldots\right)}{\sum_{\beta} Count\left(X\left(sacks\right) \rightarrow \ldots\ PP\ \ldots\right)}$$

$$= \frac{0}{0}$$

Improving PCFGs

- Parent Annotation
- Lexicalization
- **Reranking**

Reranking

- Issue: Locality
  - PCFG probabilities associated with rewrite rules
  - Context-free grammars are, well, context-free
  - Previous approaches create new rules to incorporate context
- Need approach that incorporates broader, global info

Discriminative Parse Reranking

- General approach:
  - Parse using (L)PCFG
  - Obtain top-N parses
  - Re-rank top-N using better features
- Use discriminative model (e.g. MaxEnt, NN) to rerank with features:
  - right-branching vs. left-branching
  - speaker identity
  - conjunctive parallelism
  - fragment frequency
  - …

Reranking Effectiveness

- How can reranking improve?
- Results from Collins and Koo (2005), with 50-best

System	Accuracy
Baseline	0.897
Oracle	0.968
Discriminative	0.917

- "Oracle" is to automatically choose the correct parse if in N-best

Improving PCFGs: Tradeoffs

- **Pros:**
  - Increased accuracy/specificity
  - e.g. Lexicalization, Parent annotation, Markovization, etc
- **Cons**:
  - Explode grammar size
  - Increased processing time
  - Increased data requirements
- *How can we balance?*

Improving PCFGs: Efficiency

- **Beam thresholding**
- Heuristic Filtering

Efficiency

- PCKY is $|G| \cdot n^3$
  - Grammar can be huge
  - Grammar can be extremely ambiguous
  - Hundreds of analyses not unusual
- …but only care about best parses
- Can we use this to improve efficiency?

Beam Thresholding

- Inspired by Beam Search
- Assume low probability parses unlikely to yield high probability overall
  - Keep only top k most probable partial parses
  - Retain only k choices per cell
    - For large grammars, maybe 50-100
    - For small grammars, 5 or 10

Heuristic Filtering

- **Intuition**: Some rules/partial parses unlikely to create best parse
- **Proposal**: Don't store these in table.
- Exclude:
  - Low frequency: e.g. singletons
  - Low probability: constituents $\boldsymbol{X}$ s.t. $P(\boldsymbol{X}) < 10^{-200}$
  - Low relative probability:
    - Exclude $\boldsymbol{X}$ if there exists $\boldsymbol{Y}$ s.t. $P(\boldsymbol{Y}) > 100 \times P(\boldsymbol{X})$