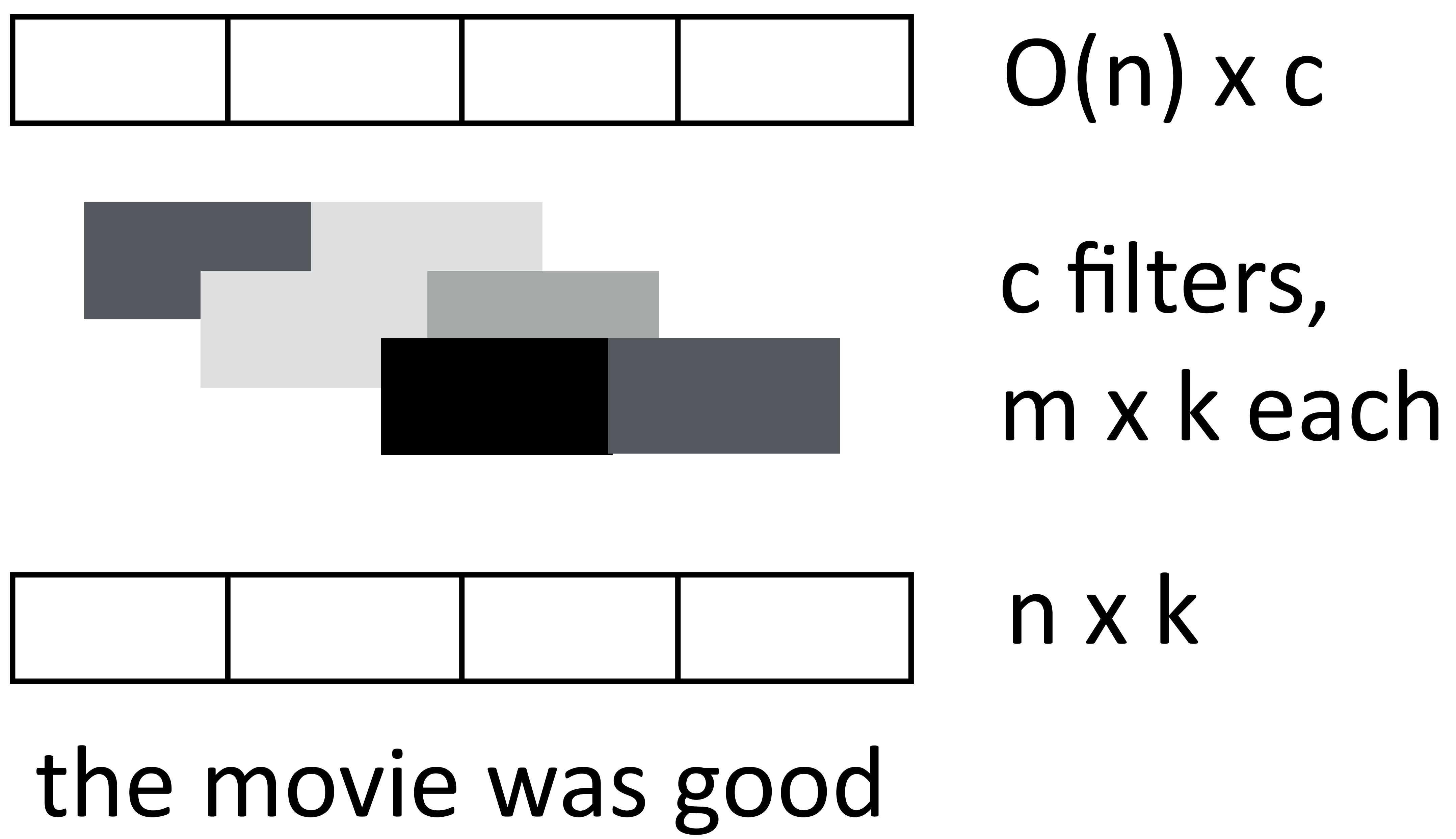
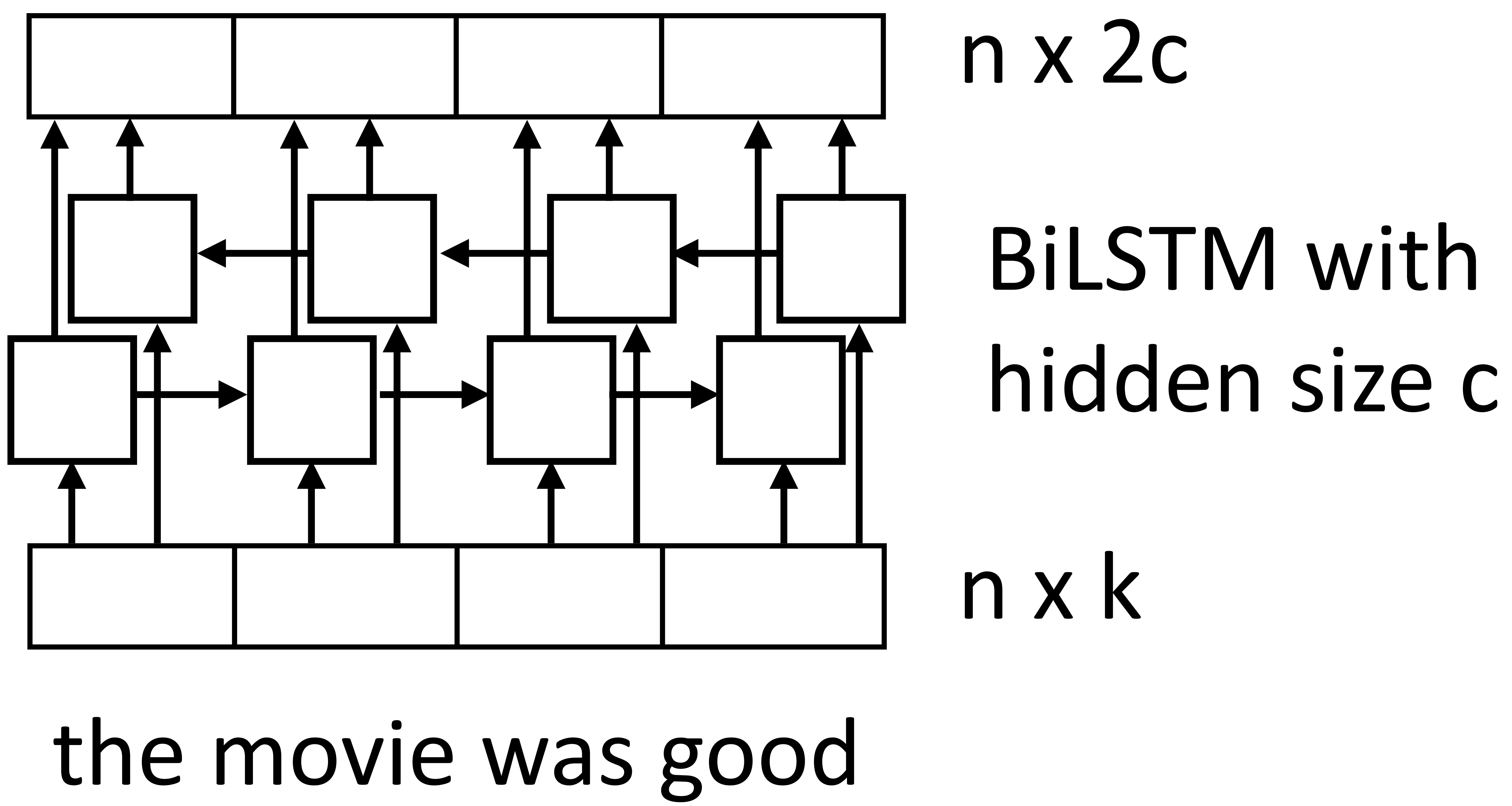
CS388: Natural Language Processing
Lecture 10: Syntax I

Greg Durrett

Slides adapted from Dan Klein, UC Berkeley

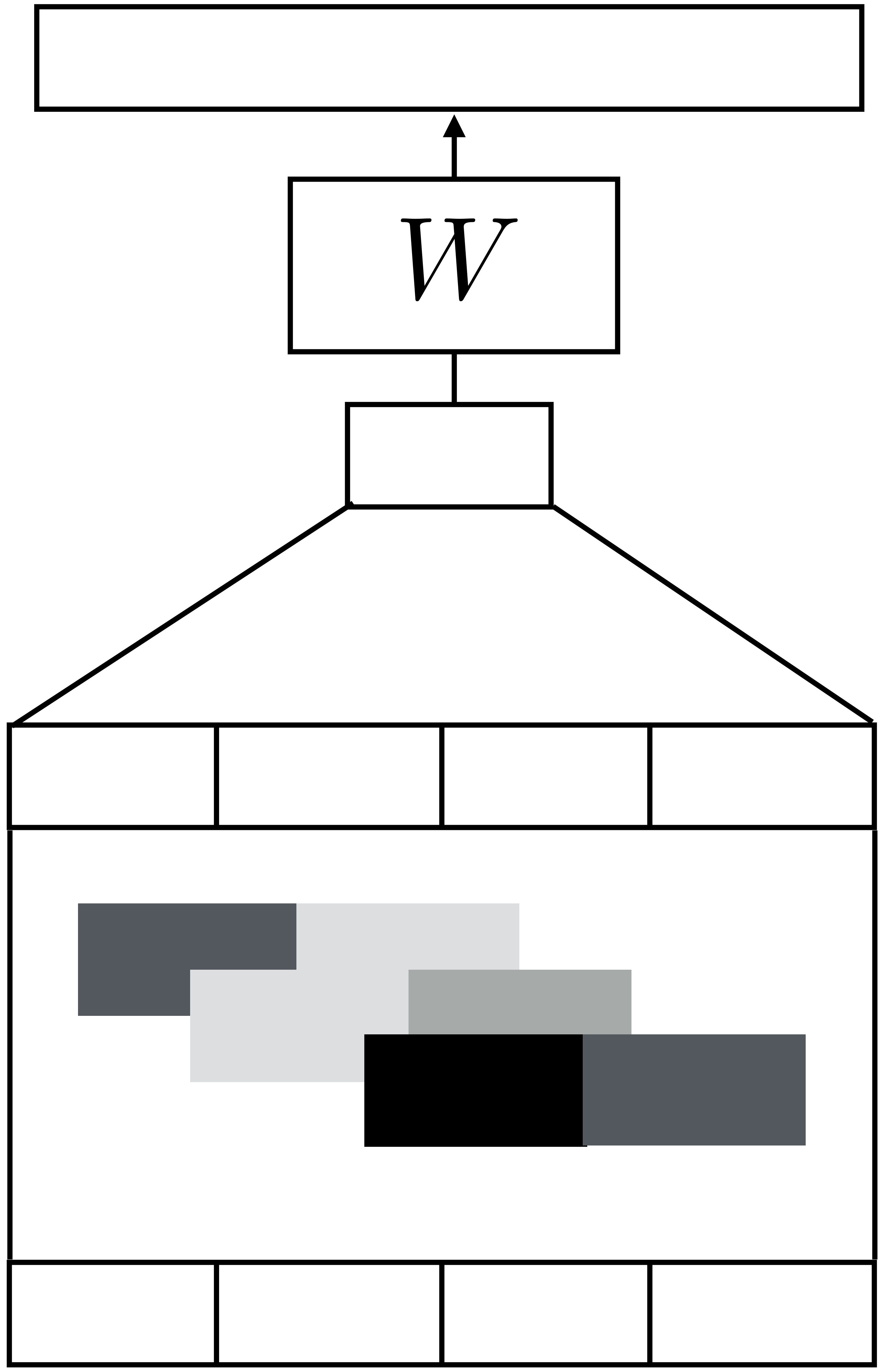
Recall: CNNs vs. LSTMs





- Both LSTMs and convolutional layers transform the input using context
- LSTM: “globally” looks at the entire sentence (but local for many problems)
- CNN: local depending on filter width + number of layers

Recall: CNNs



the movie was good

$P(y|\mathbf{x})$

projection + softmax

c-dimensional vector

max pooling over the sentence

n x c

c filters,
m x k each

n x k

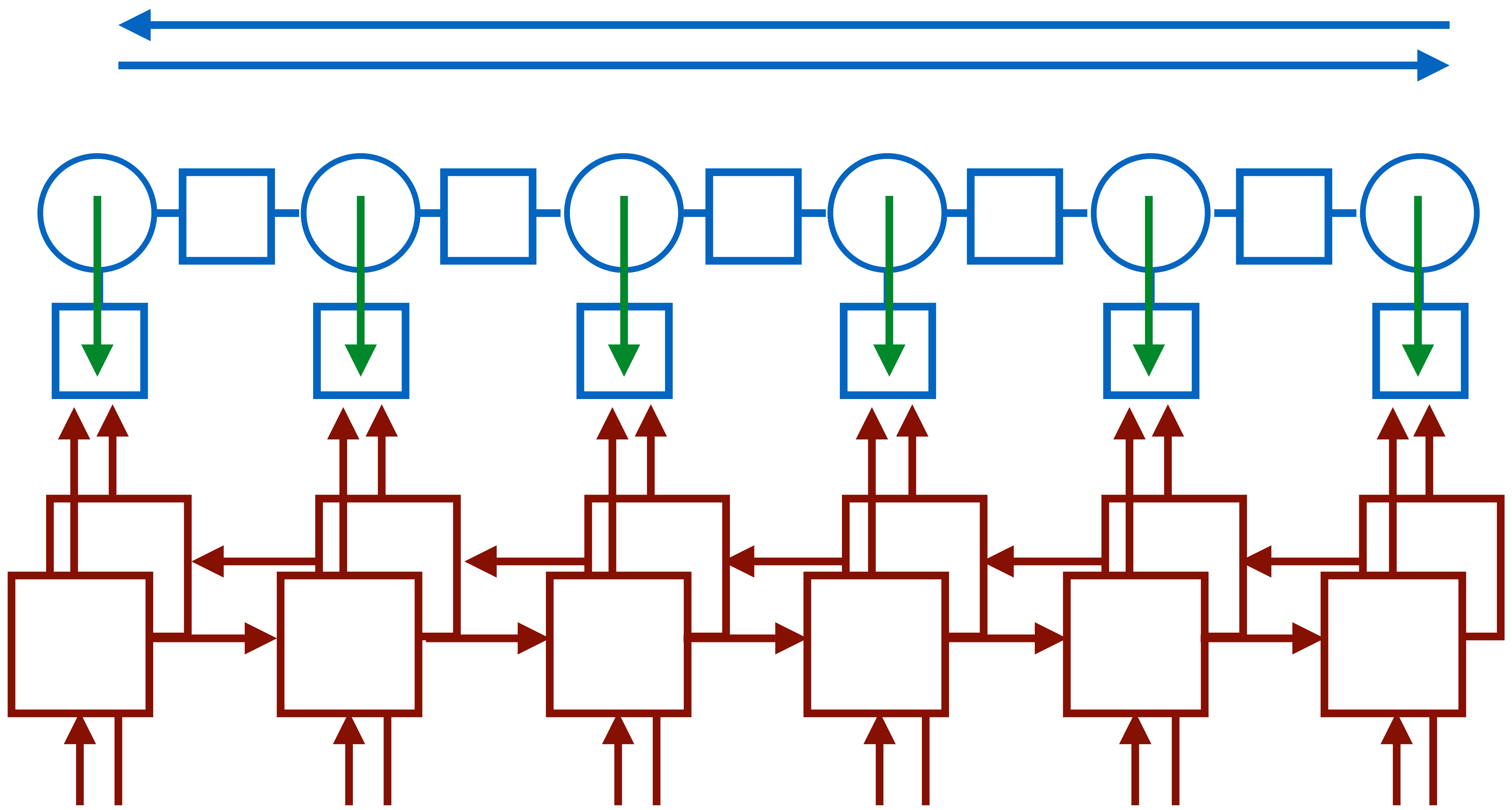
- Max pooling: return the max activation of a given filter over the entire sentence; like a logical OR (sum pooling is like logical AND)

Recall: Neural CRFs

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG



2) Run forward-backward

3) Compute error signal

1) Compute f(**x**)

4) Backprop (no knowledge of sequential structure required)

This Lecture

- Constituency formalism
- Context-free grammars and the CKY algorithm
- Refining grammars
- Discriminative parsers

Constituency

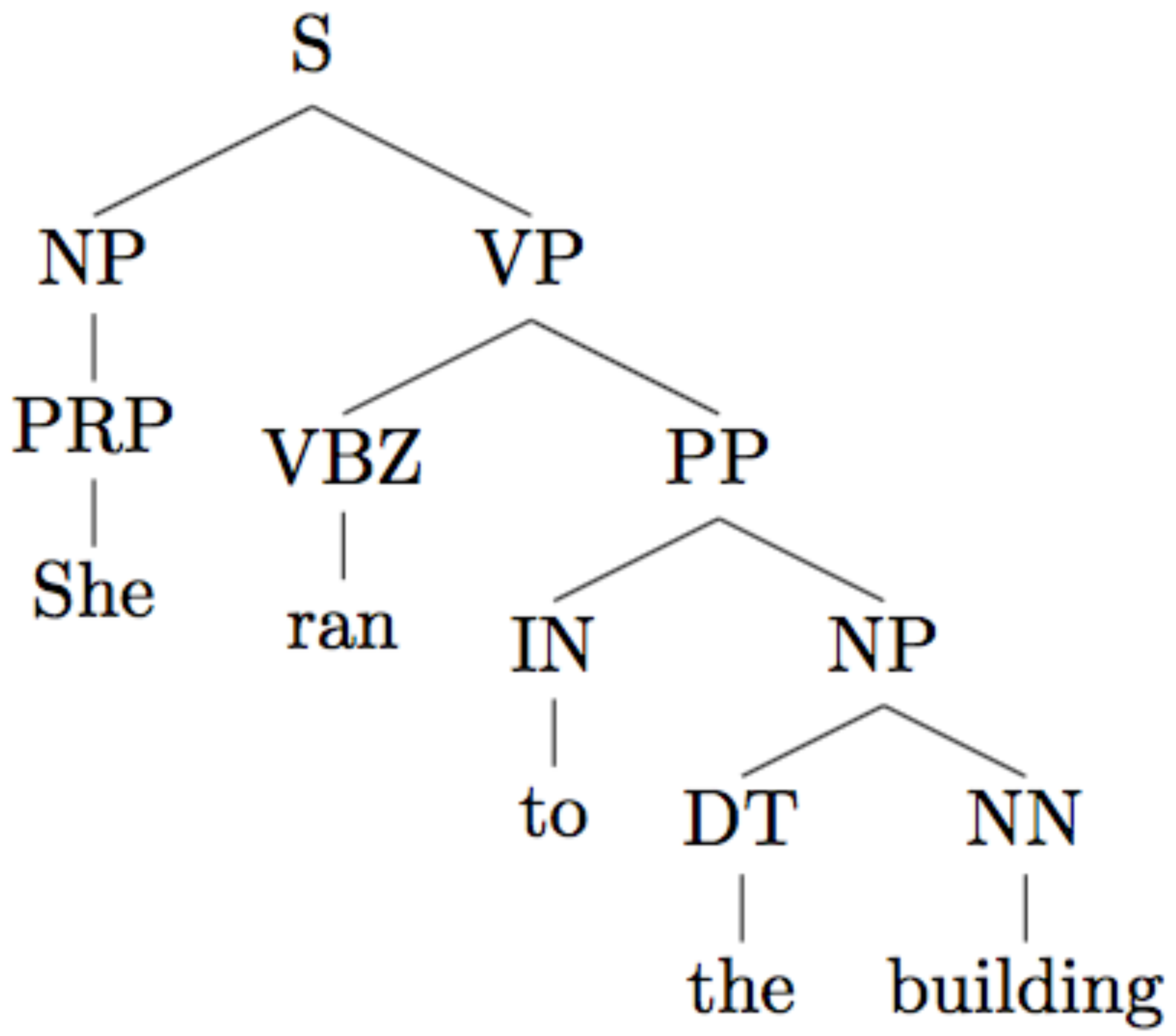
Syntax

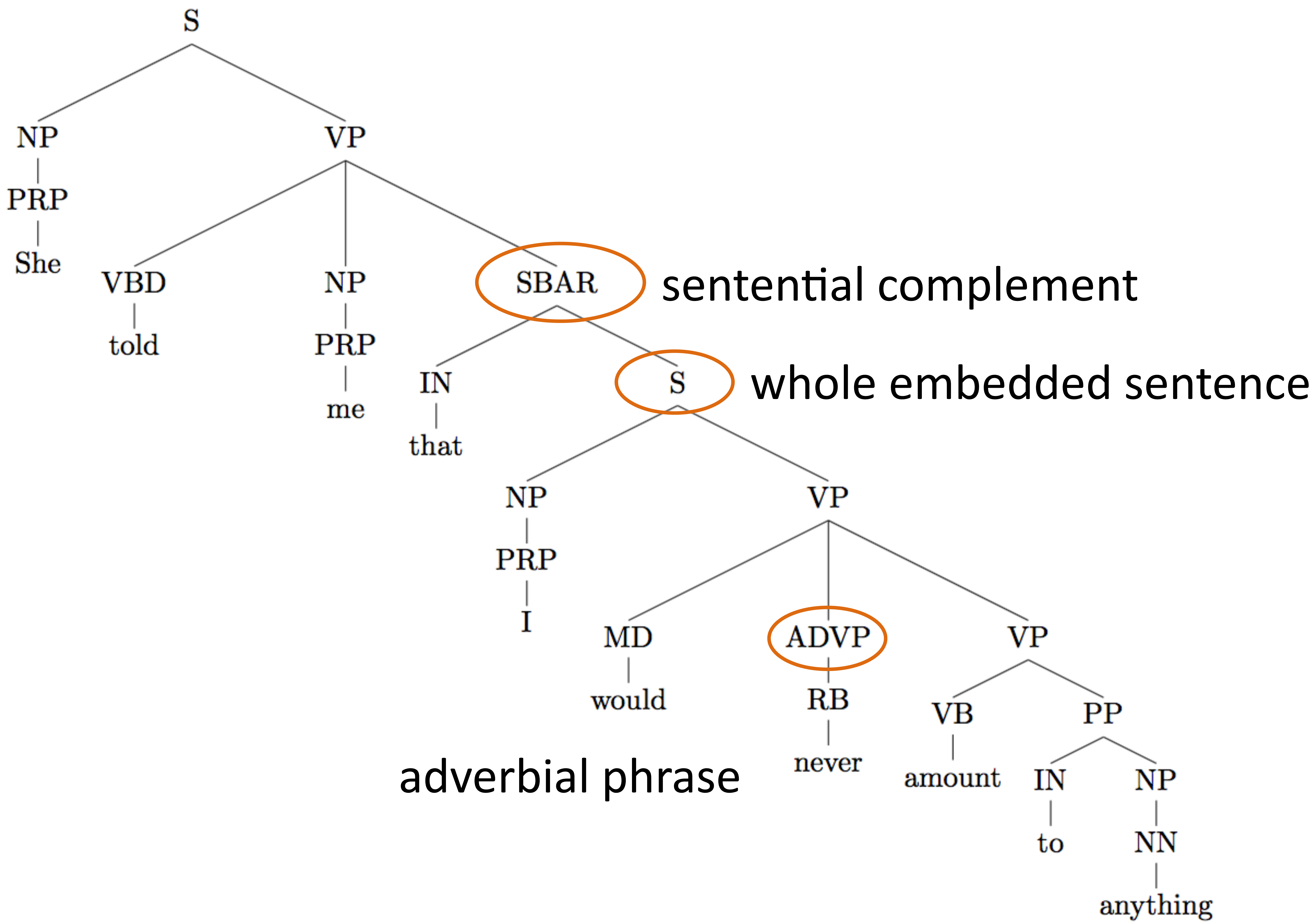
- Study of word order and how words form sentences

- Why do we care about syntax?
  - Multiple interpretations of words (noun or verb? *Fed raises*... example)
  - Recognize verb-argument structures (who is doing what to whom?)
  - Higher level of abstraction beyond words: some languages are SVO, some are VSO, some are SOV, parsing can canonicalize

Constituency Parsing

- Tree-structured syntactic analyses of sentences
- Common things: noun phrases, verb phrases, prepositional phrases
- Bottom layer is POS tags
- Examples will be in English. Constituency makes sense for a lot of languages but not all




S
NP
VP
PRP
She
VBD
told
NP
PRP
me
SBAR
sentential complement
IN
that
S
whole embedded sentence
NP
PRP
I
VP
MD
would
ADVP
RB
never
VP
VB
amount
PP
IN
to
NP
NN
anything
adverbial phrase

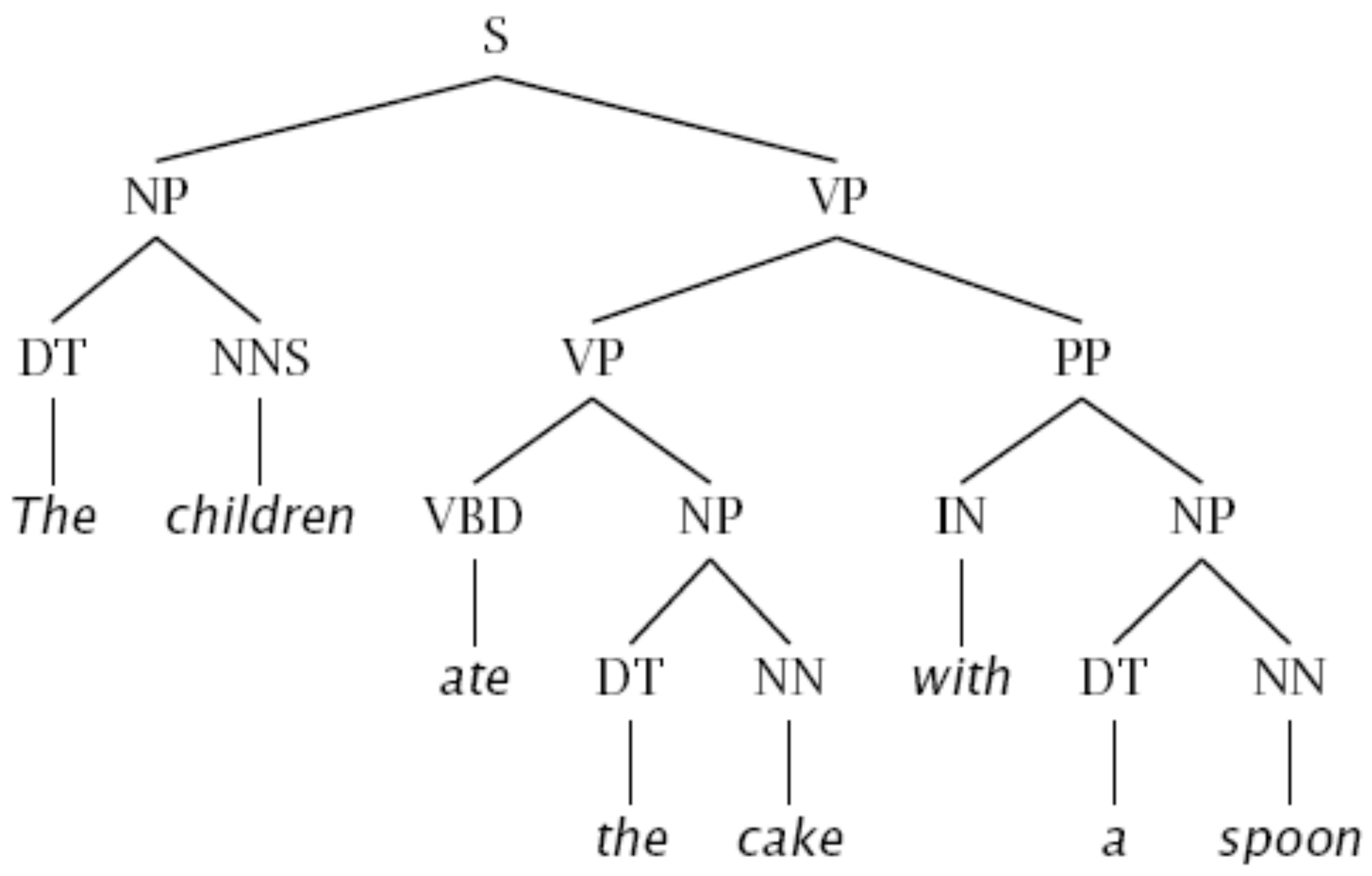
Constituency Parsing

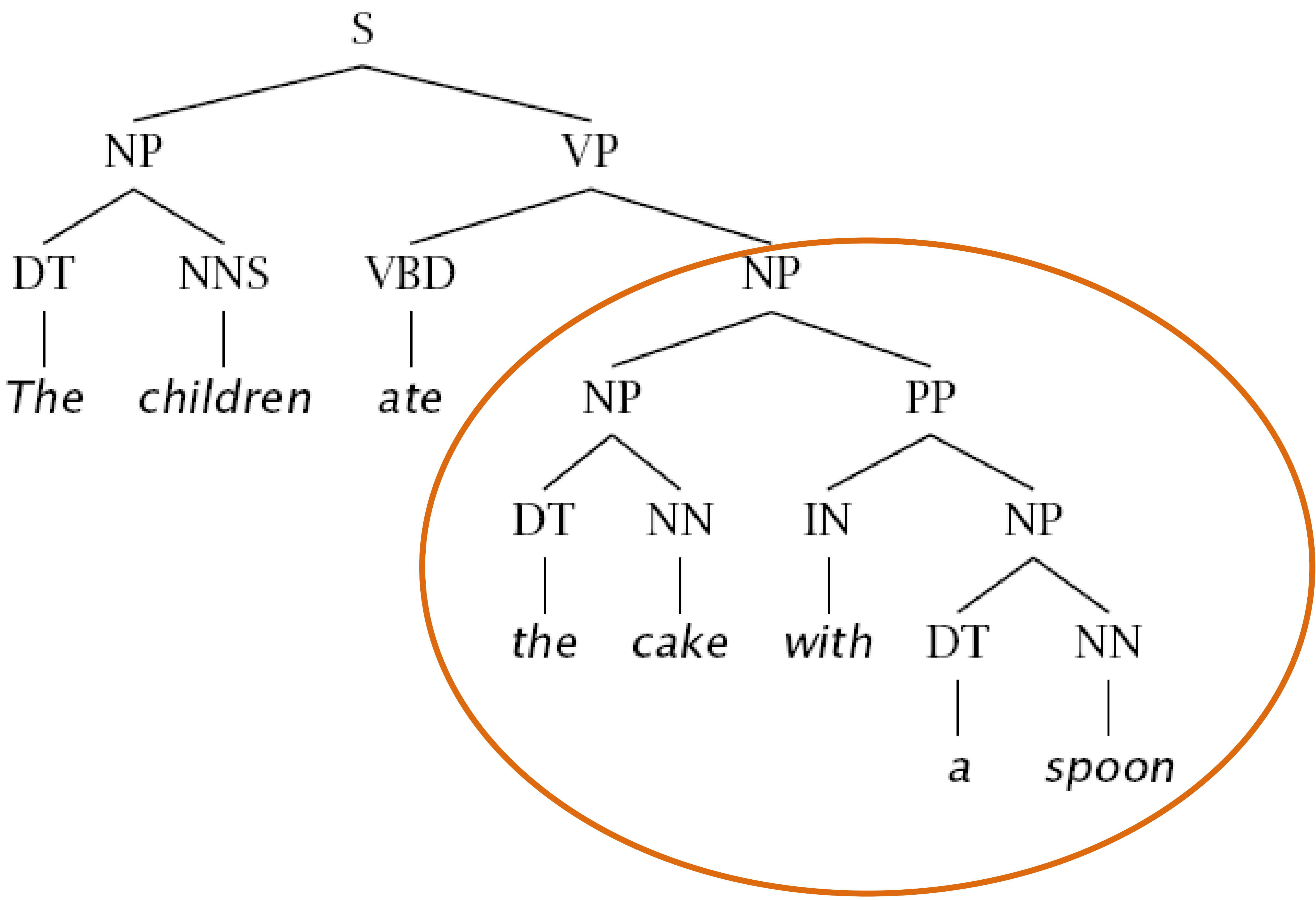
The rat the cat chased squeaked

I raced to Indianapolis , unimpeded by traffic

Challenges

- PP attachment

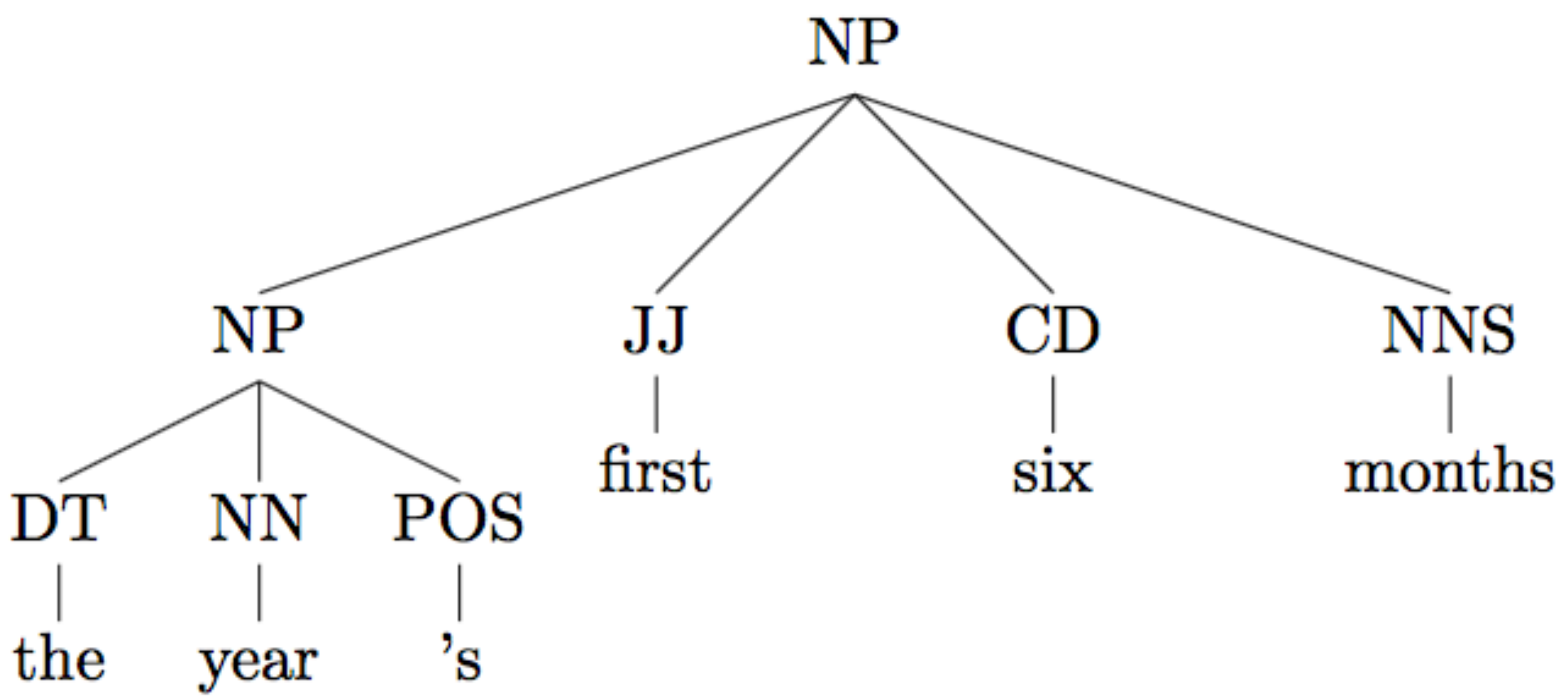


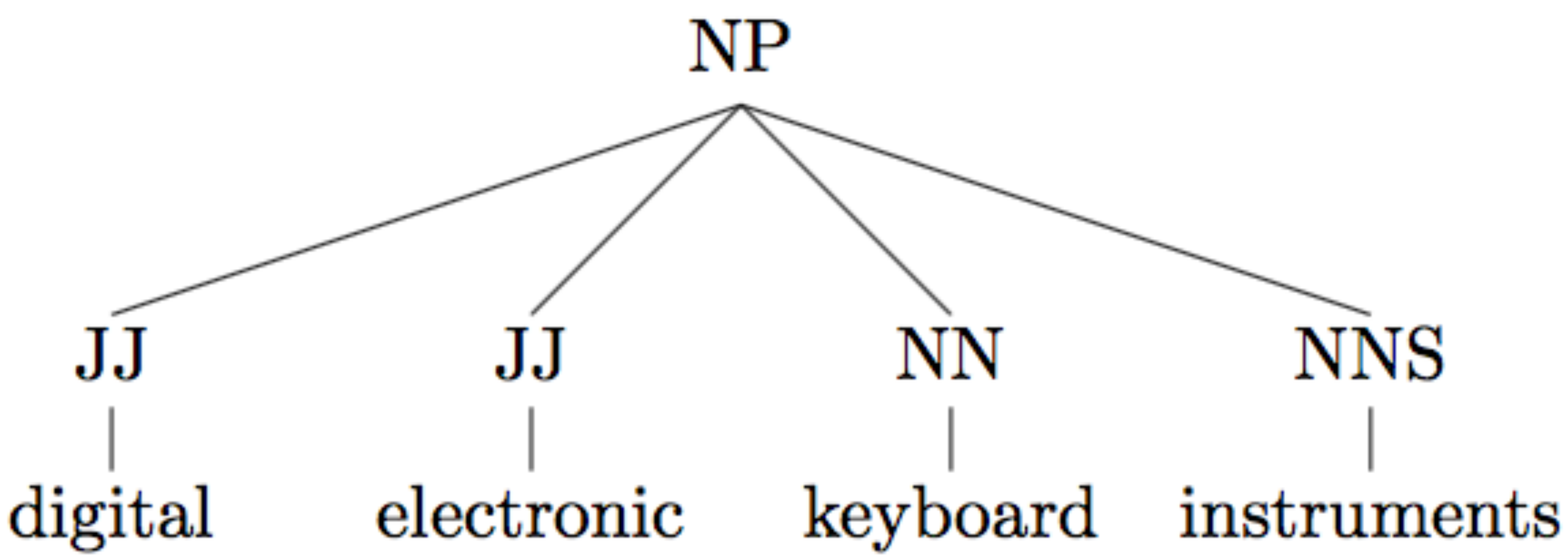


same parse as “the cake with some icing”

Challenges

- NP internal structure: tags + depth of analysis

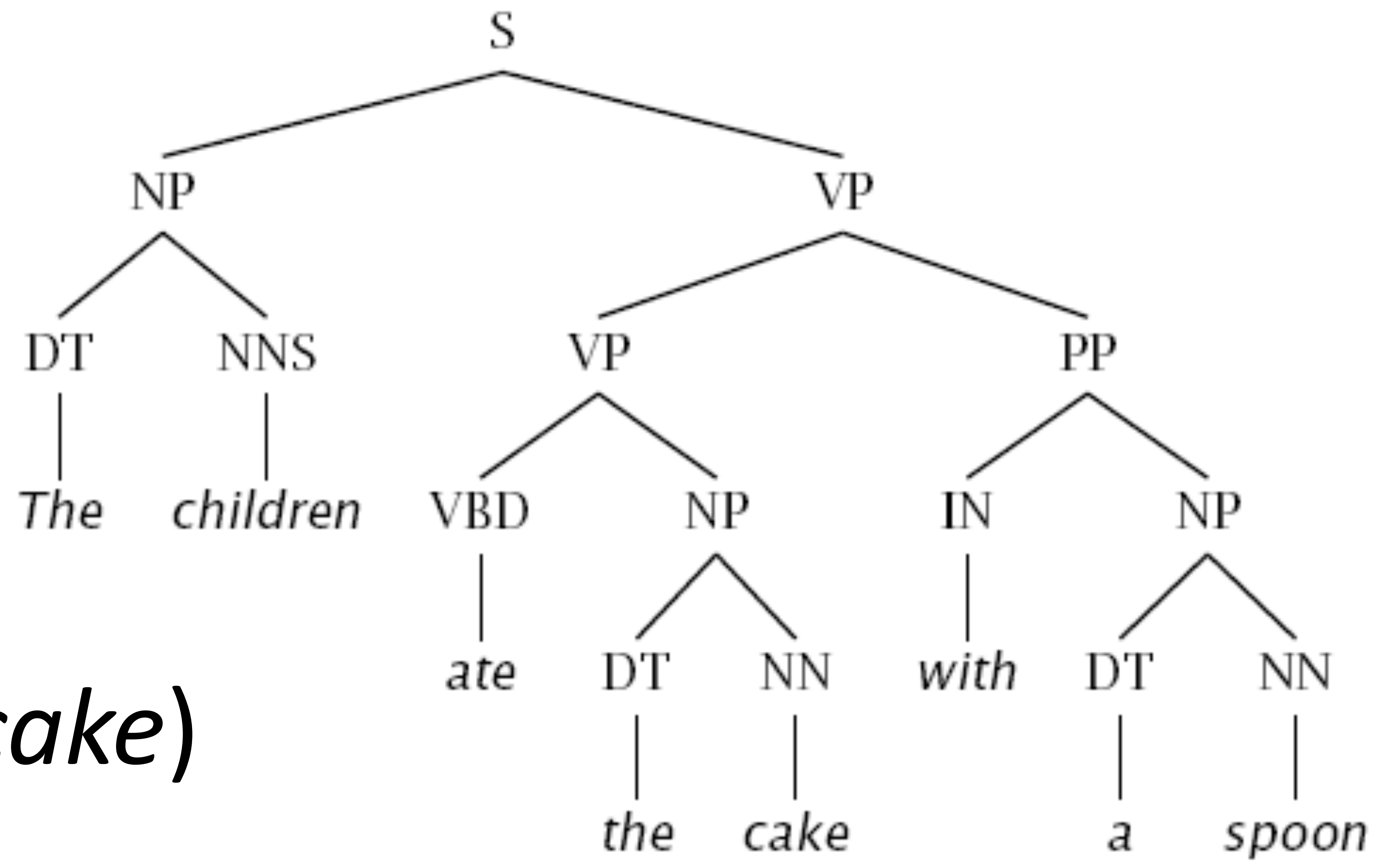




Constituency

- How do we know what the constituents are?

- Constituency tests:
  - Substitution by *proform* (e.g., pronoun)
  - Clefting (*It was with a spoon that…*)
  - Answer ellipsis (What did they eat? *the cake*)
    (How? *with a spoon*)



- Sometimes constituency is not clear, e.g., coordination: *she went to and bought food at the store*

Context-Free Grammars, CKY

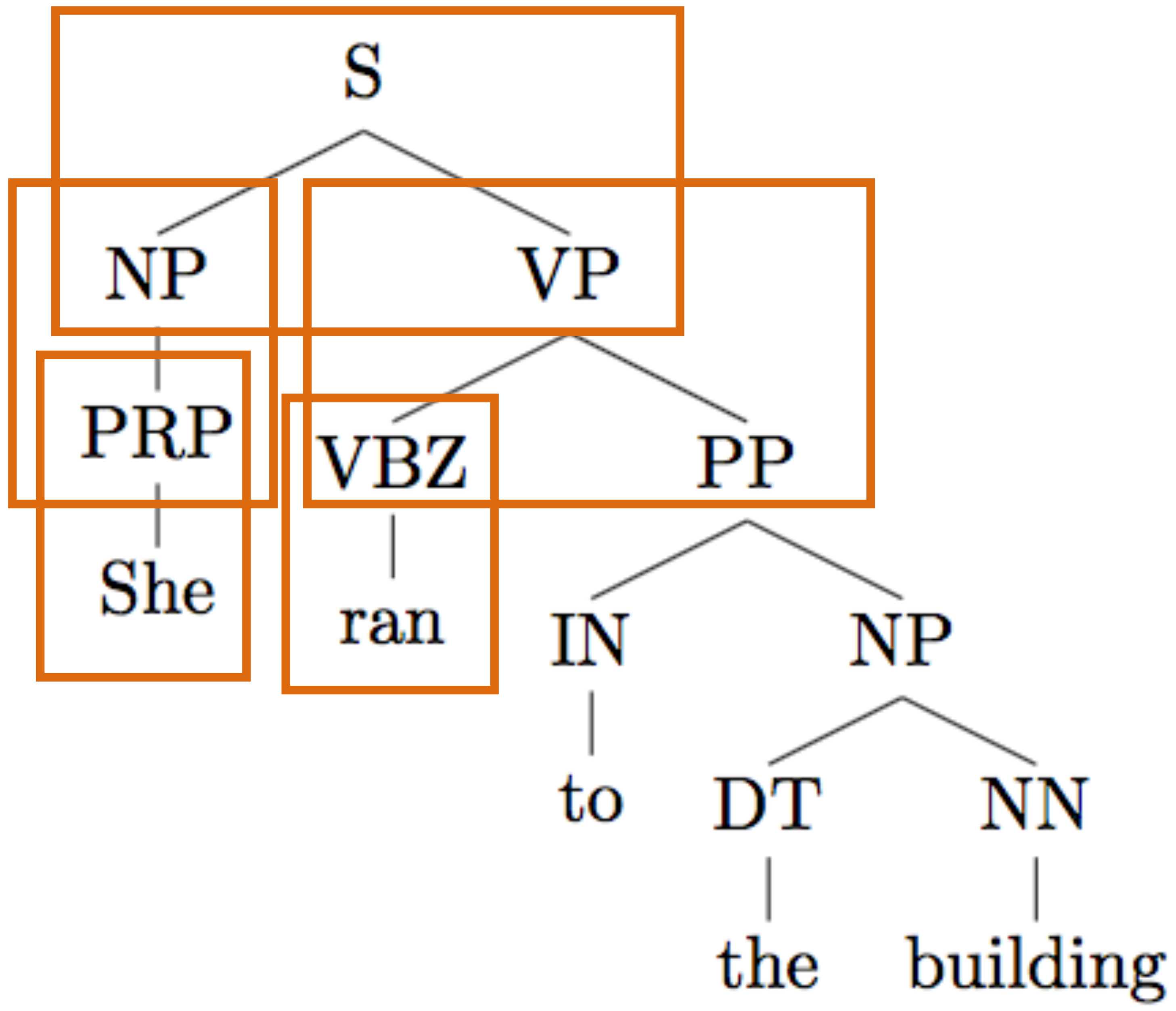
CFGs and PCFGs

Grammar (CFG)				Lexicon	
ROOT → S	1.0	NP → NP PP	0.3	NN → interest	1.0
S → NP VP	1.0	VP → VBP NP	0.7	NNS → raises	1.0
NP → DT NN	0.2	VP → VBP NP PP	0.3	VBP → interest	1.0
NP → NN NNS	0.5	PP → IN NP	1.0	VBZ → raises	1.0

- Context-free grammar: symbols which rewrite as one or more symbols
- Lexicon consists of "preterminals" (POS tags) rewriting as terminals (words)
- CFG is a tuple (N, T, S, R): N = nonterminals, T = terminals, S = start symbol (generally a special ROOT symbol), R = rules
- PCFG: probabilities associated with rewrites, normalize by source symbol

Estimating PCFGs

- Tree *T* is a series of rule applications *r*. $P(T) = \prod_{r \in T} P(r|\text{parent}(r))$

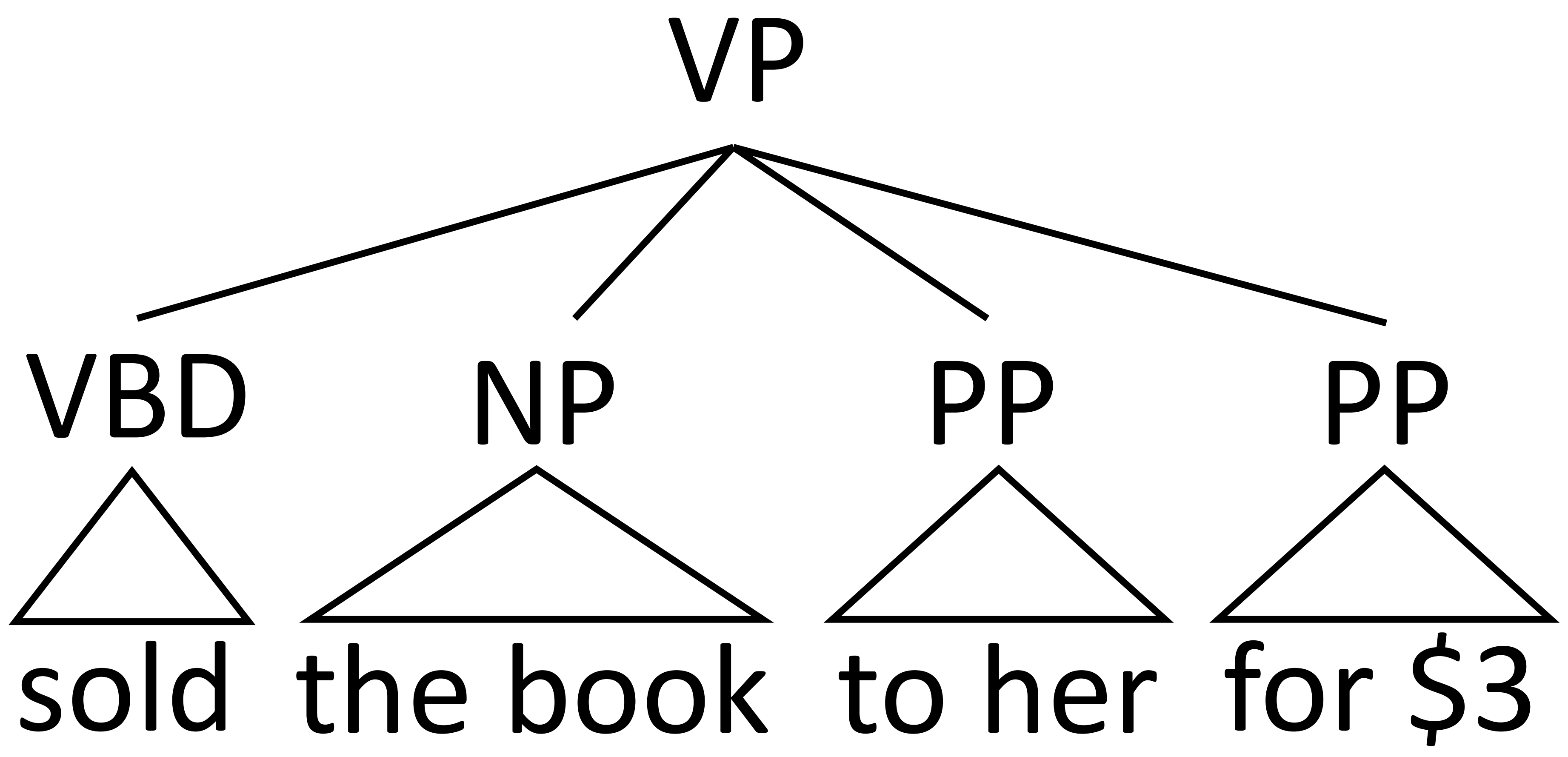


S $\rightarrow$ NP VP	1.0
NP $\rightarrow$ PRP	0.5
NP $\rightarrow$ DT NN	0.5
…	

- Maximum likelihood PCFG: count and normalize! Same as HMMs / Naive Bayes

Binarization

- To parse efficiently, we need our PCFGs to be at most binary (not CNF)

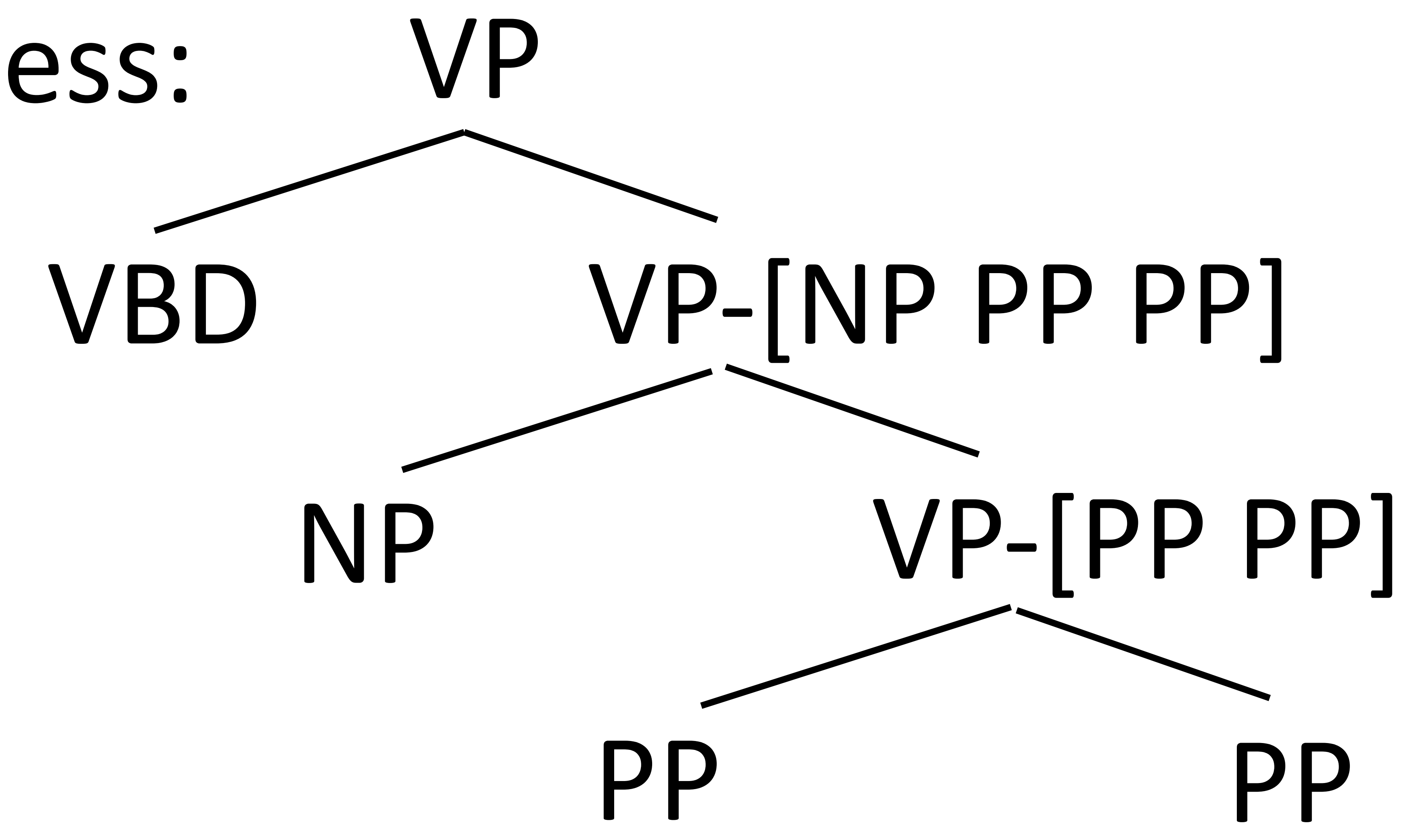


$P(VP \rightarrow VBD\ NP\ PP\ PP) = 0.2$

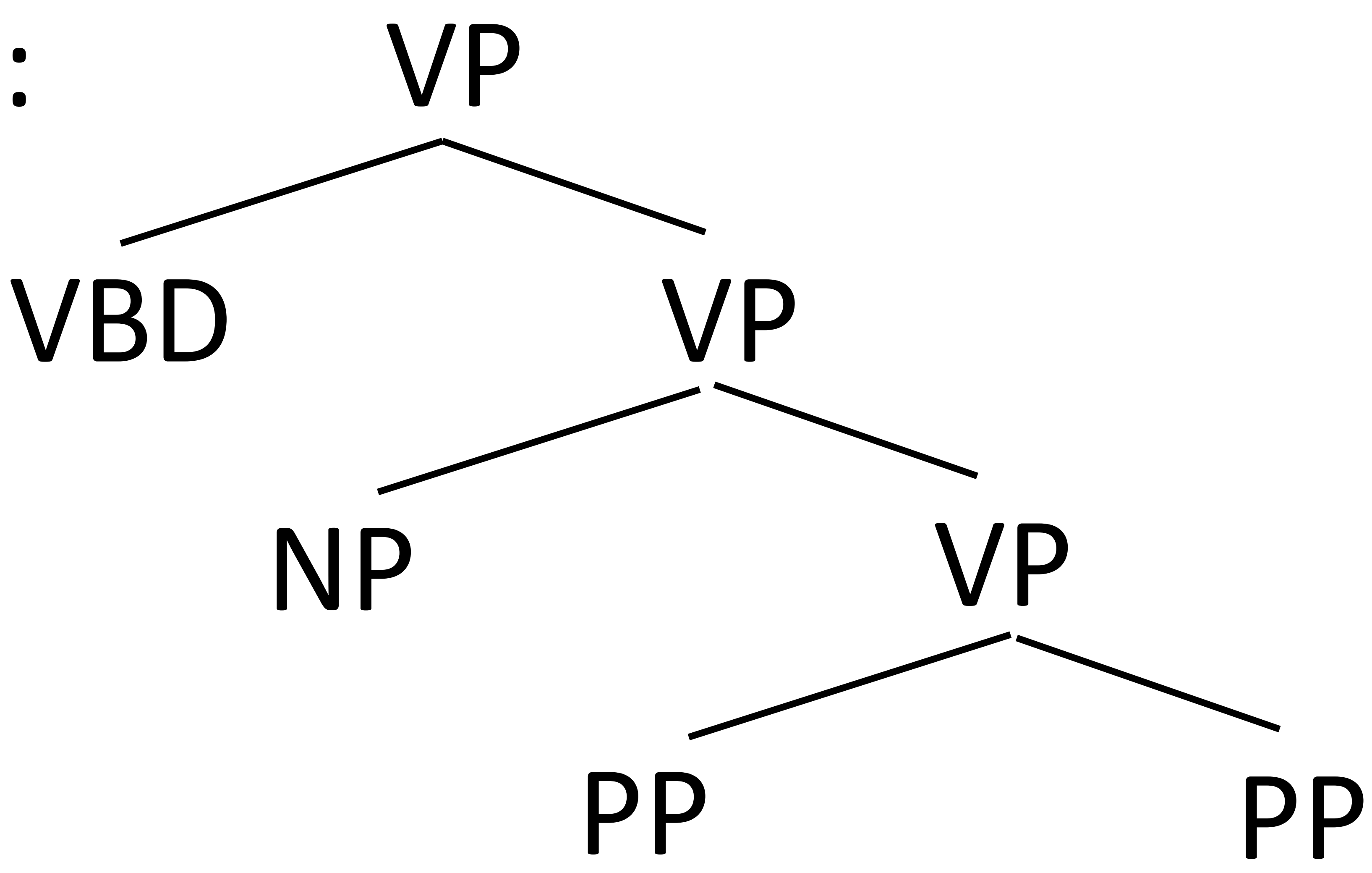
$P(VP \rightarrow VBZ\ PP) = 0.1$

…

- Lossless:

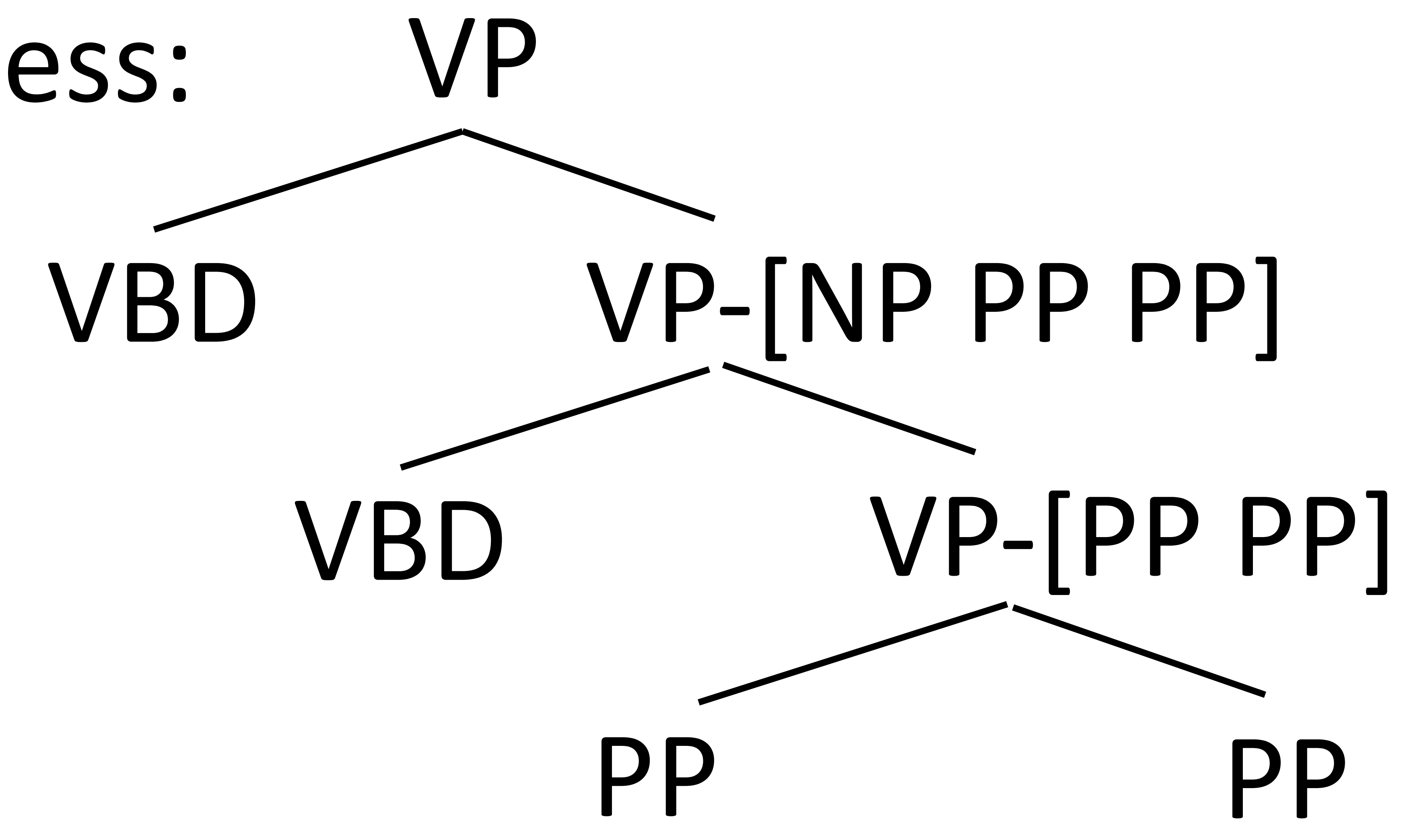


- Lossy:



Chomsky Normal Form

- Lossless:



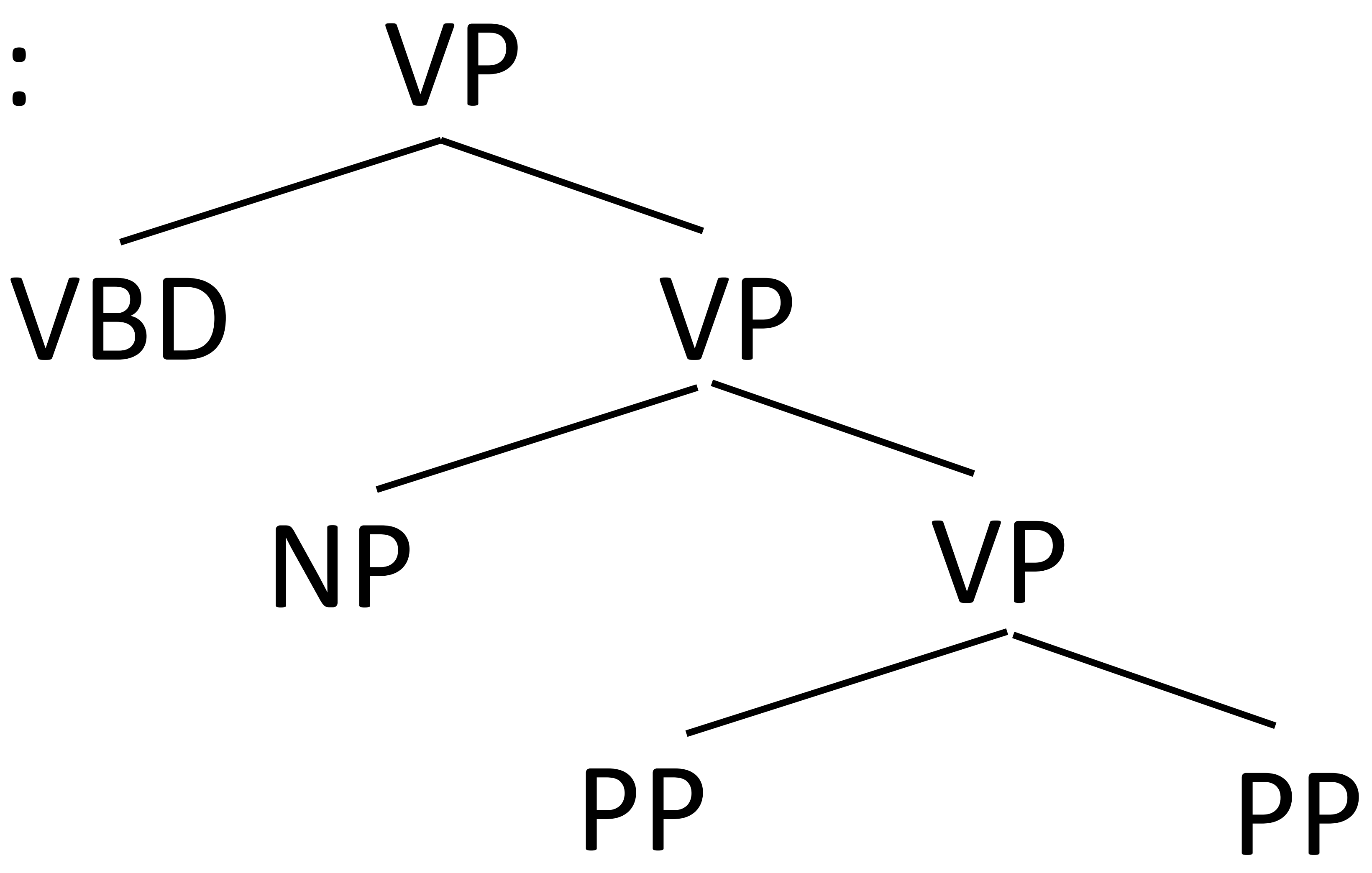
$P(\text{VP} \rightarrow \text{VBD VP-[NP PP PP]}) = 0.2$

$P(\text{VP-[NP PP PP]} \rightarrow \text{NP VP-[PP PP]}) = 1.0$

$P(\text{VP-[PP PP]} \rightarrow \text{PP PP}) = 1.0$

- Deterministic symbols make this the same as before

- Lossy:



$P(\text{VP} \rightarrow \text{VBD VP}) = 0.2$

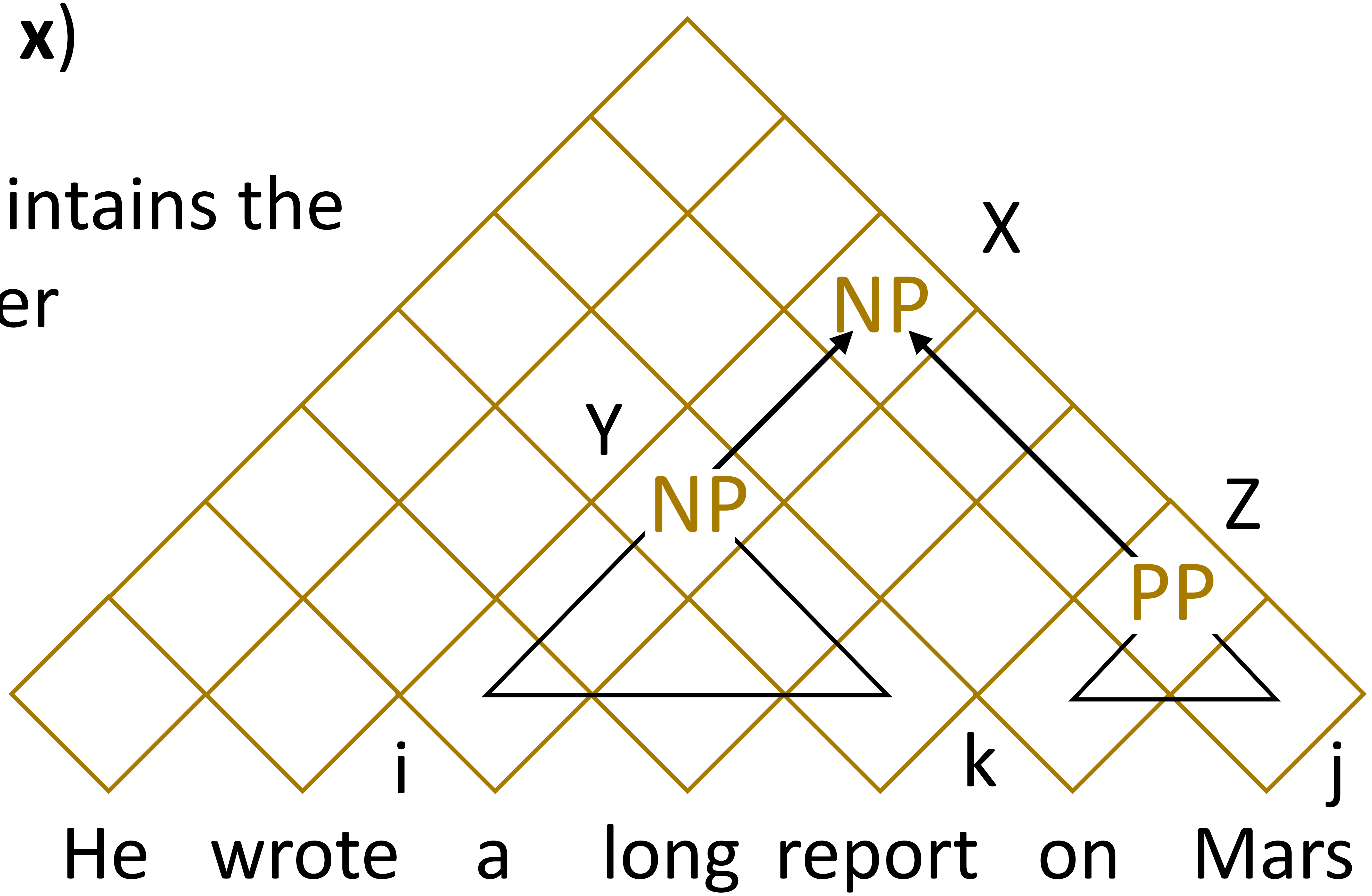
$P(\text{VP} \rightarrow \text{NP VP}) = 0.03$

$P(\text{VP} \rightarrow \text{PP PP}) = 0.001$

- Makes different independent assumptions, not the same PCFG

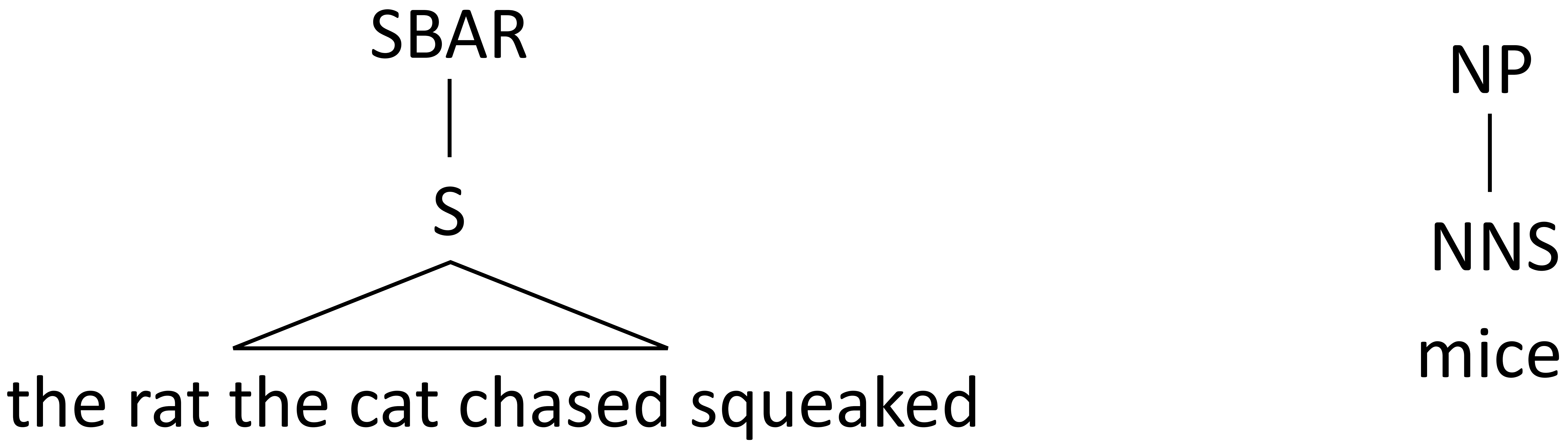
CKY

- Find argmax $P(T|x)$ = argmax $P(T, \mathbf{x})$
- Dynamic programming: chart maintains the best way of building symbol X over span (i, j)
- Loop over all split points k, apply rules X -> Y Z to build X in every possible way
- CKY = Viterbi, also an algorithm called inside-outside = forward-backward



Cocke-Kasami-Younger

Unary Rules



- Unary productions in treebank need to be dealt with by parsers
- Binary trees over n words have at most n-1 nodes, but you can have unlimited numbers of nodes with unaries (S → SBAR → NP → S → …)
- In practice: enforce at most one unary over each span, modify CKY accordingly

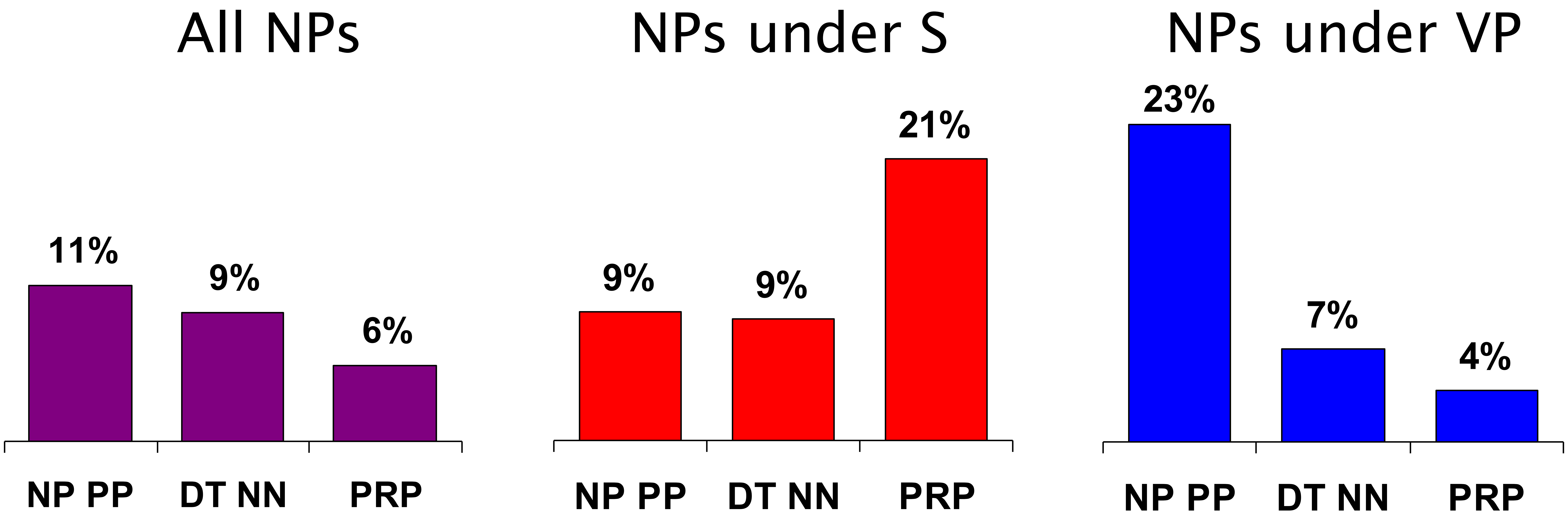
Results

- Standard dataset for English: Penn Treebank (Marcus et al., 1993)
  - Evaluation: F1 over labeled constituents of the sentence
- Vanilla PCFG: ~75 F1
- Best PCFGs for English: ~90 F1
- SOTA: 95 F1
- Other languages: results vary widely depending on annotation + complexity of the grammar

Klein and Manning (2003)

Refining Generative Grammars

PCFG Independence Assumptions

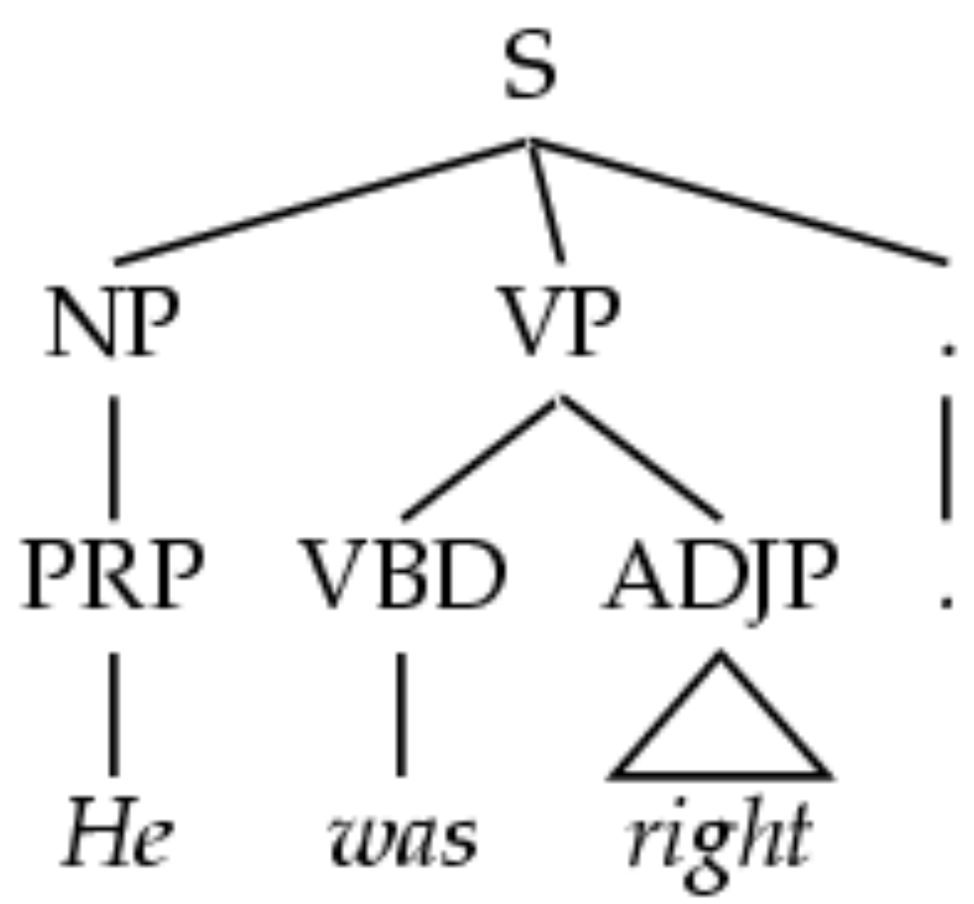


- Language is not context-free: NPs in different contexts rewrite differently
- Can we make the grammar "less context-free"?

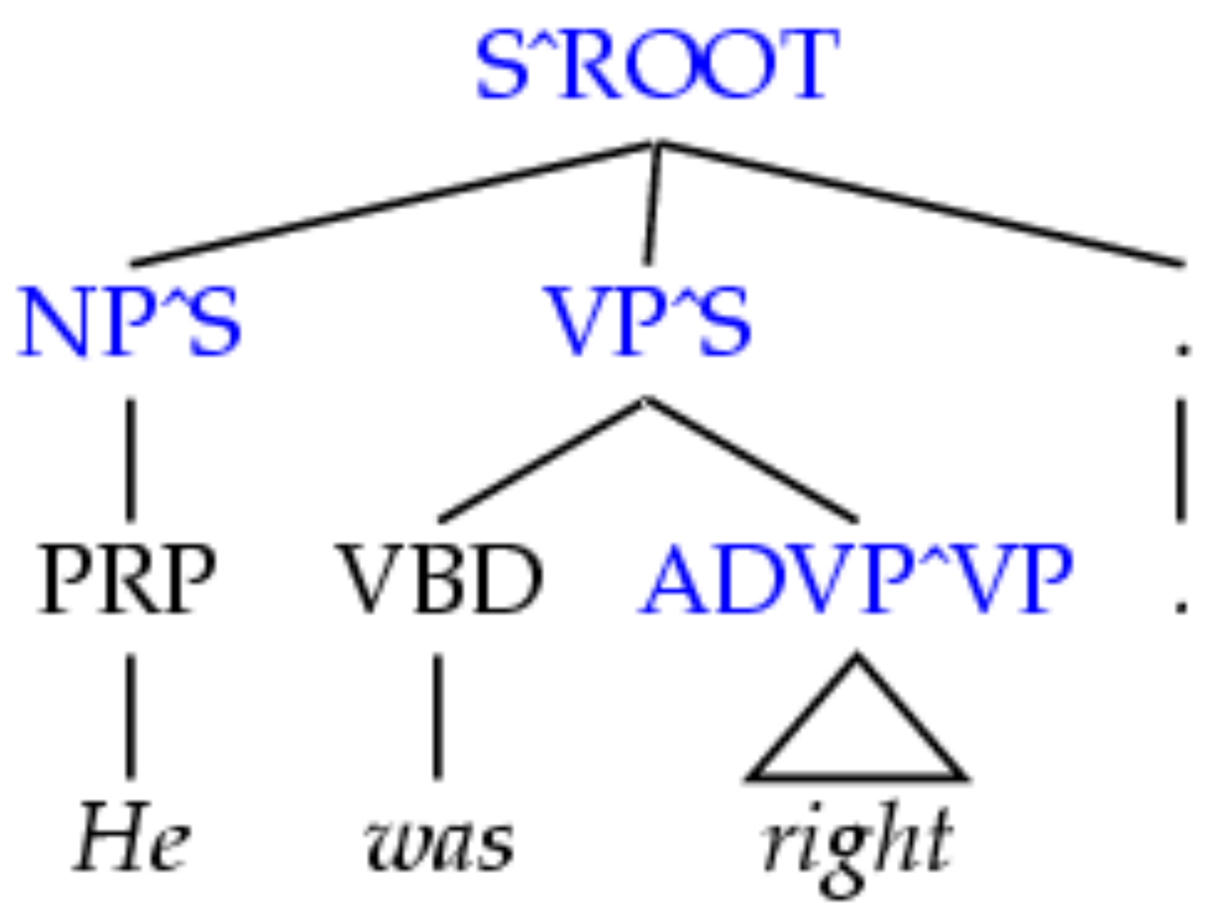
Rule Annotation

- Like a trigram HMM tagger, incorporates more context
- Vertical (parent) annotation: add the parent symbol to each node, can do grandparents too
- Horizontal annotation: remember the states of multi-arity rules during binarization

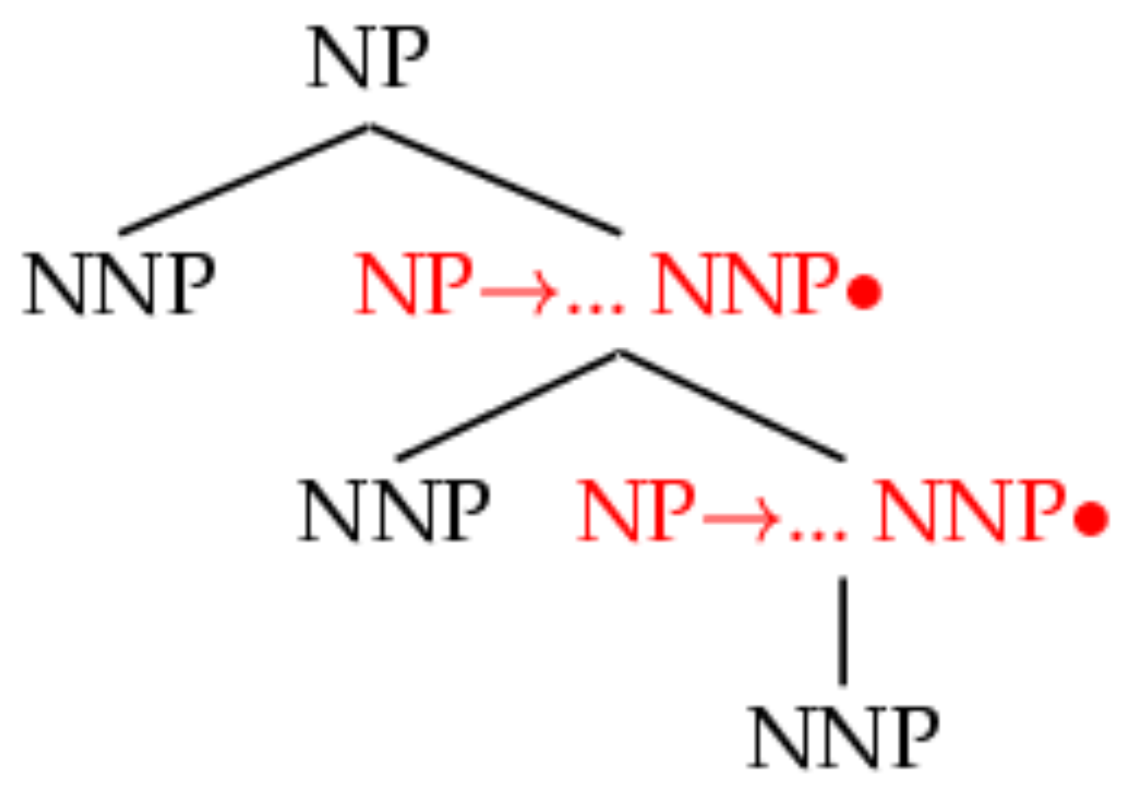
Order 1



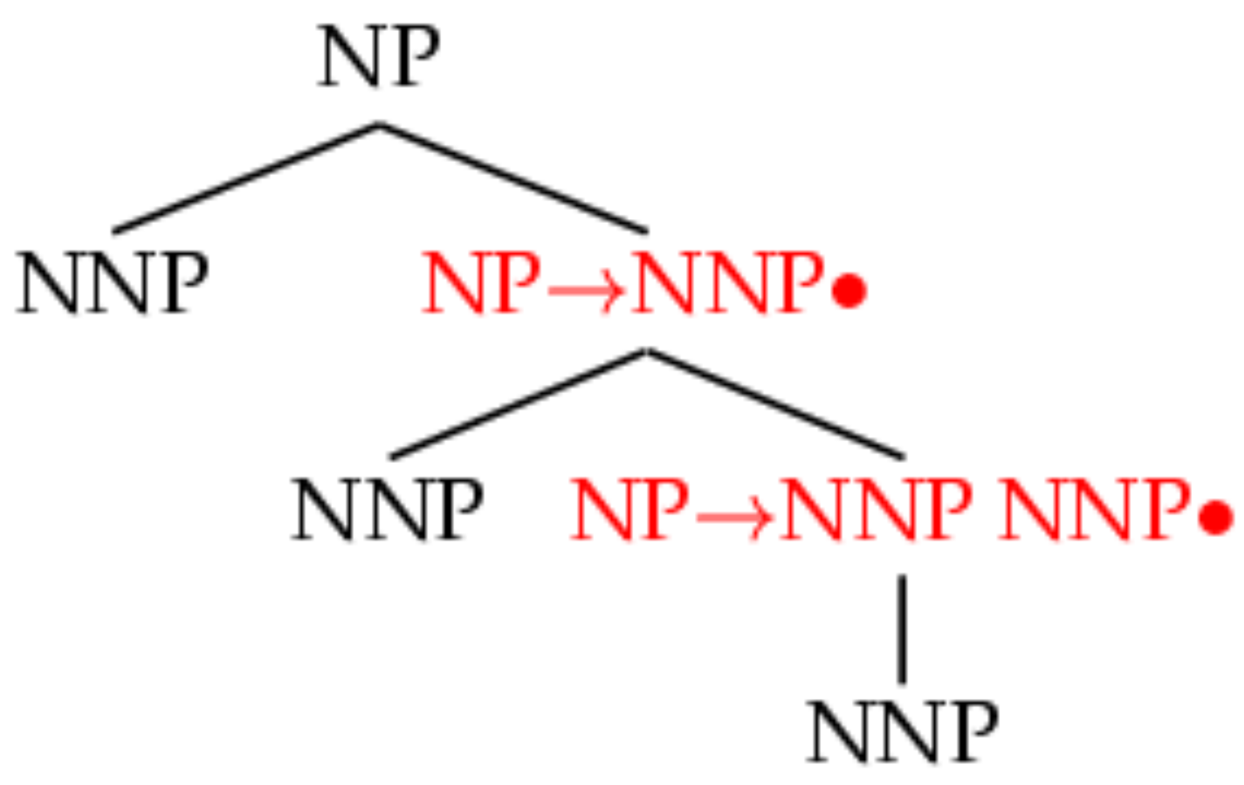
Order 2



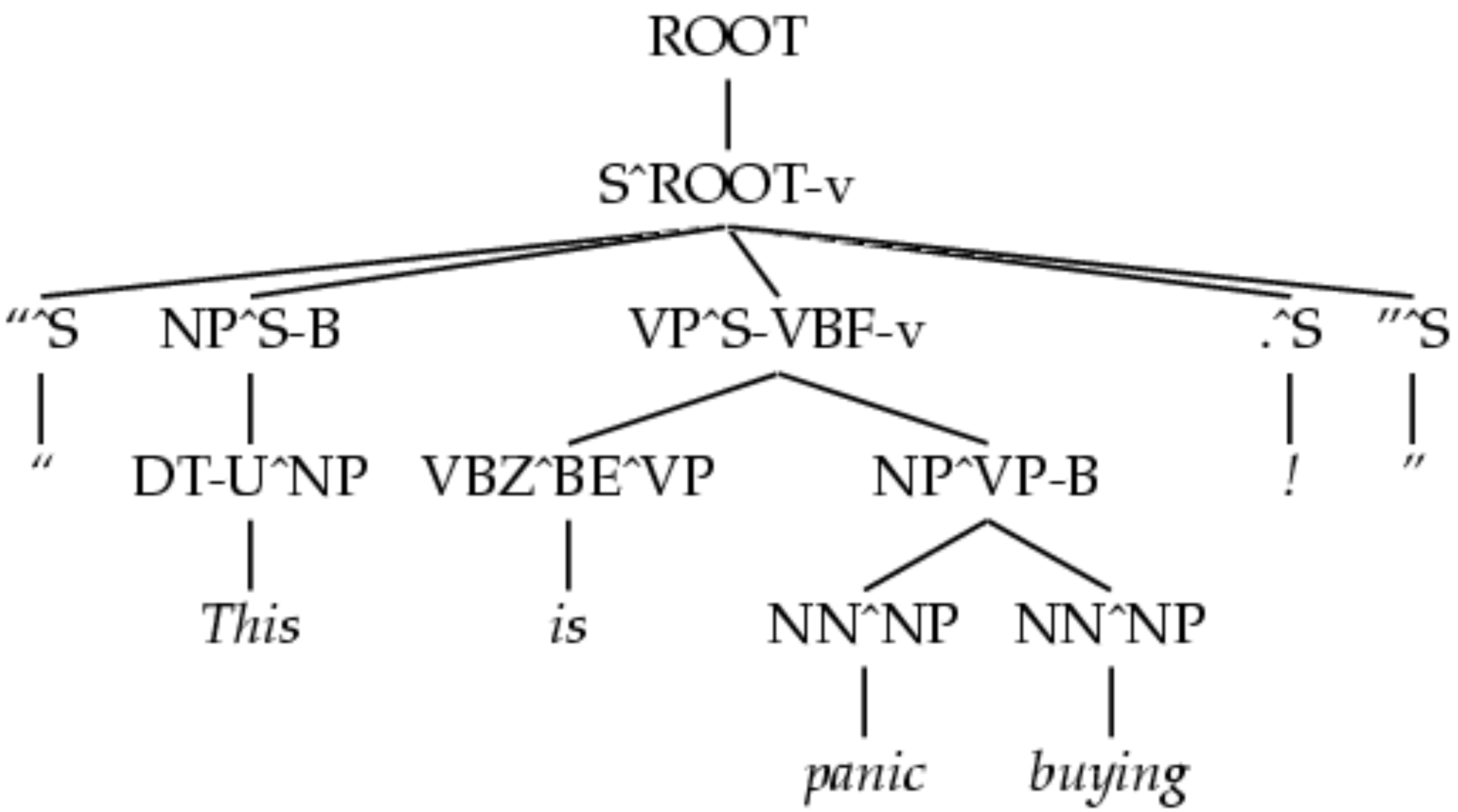
Order 1



Order ∞



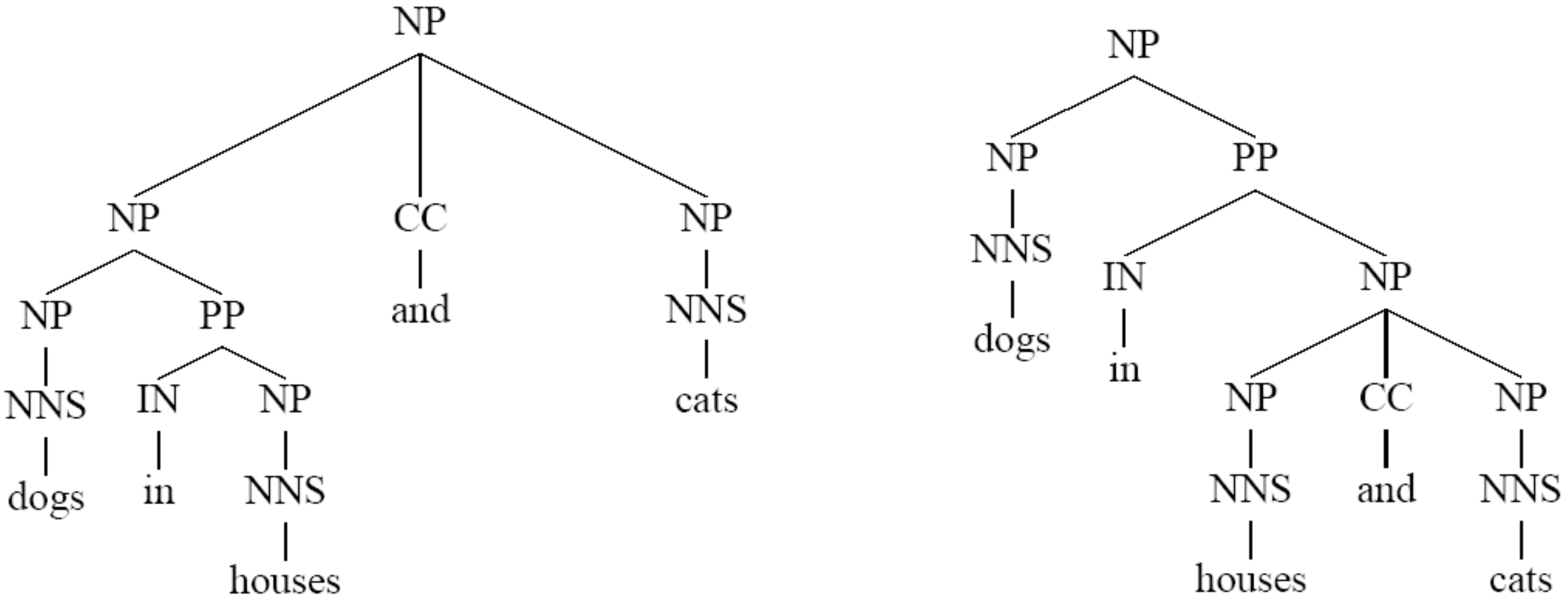
Annotated Tree



- 75 F1 with basic PCFG => 86.3 F1 with this highly customized PCFG (SOTA was 90 F1 at the time, but with more complex methods)

Klein and Manning (2003)

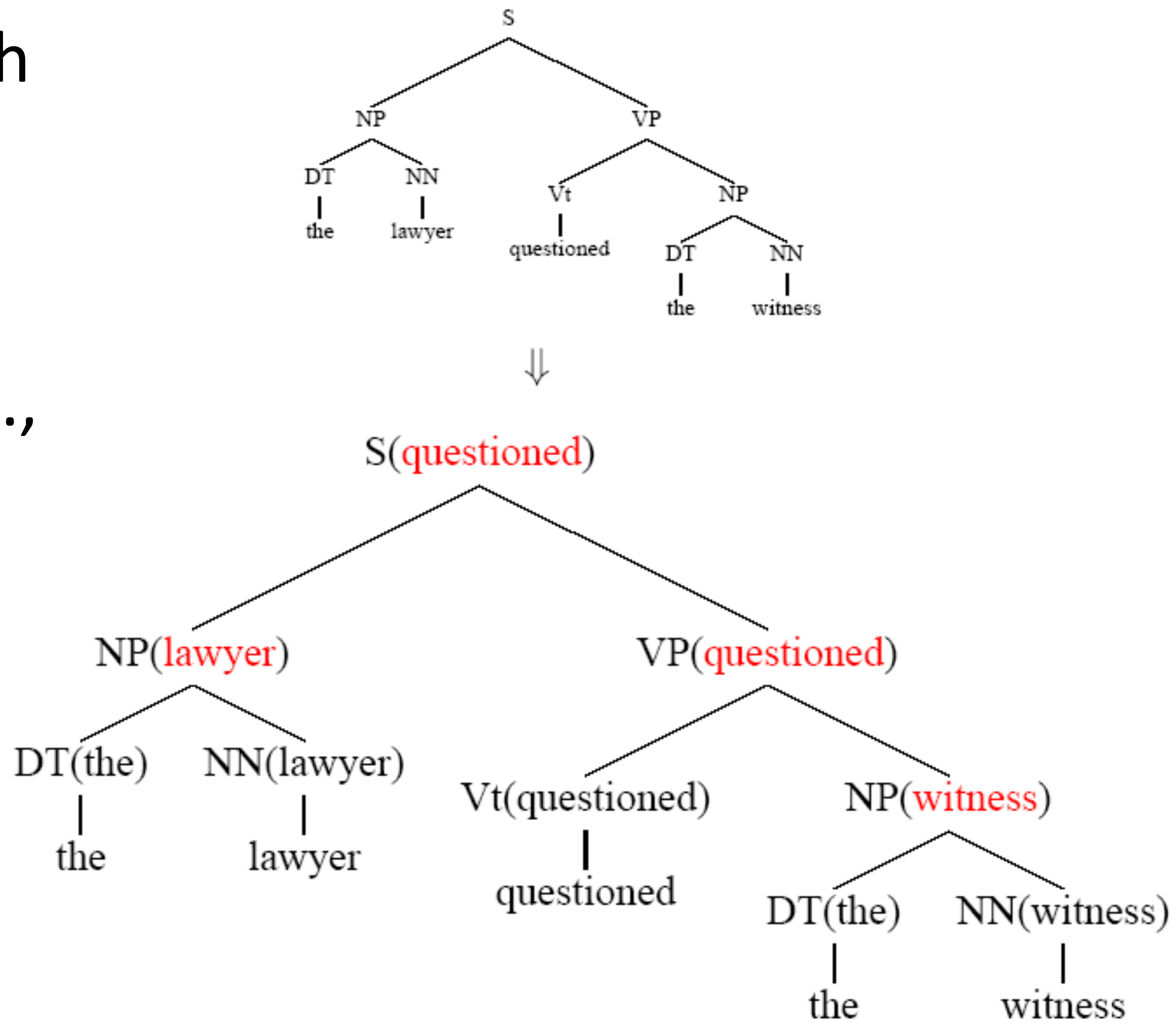
Lexicalized Parsers



- Even with parent annotation, these trees have the same rules. Need to use the words

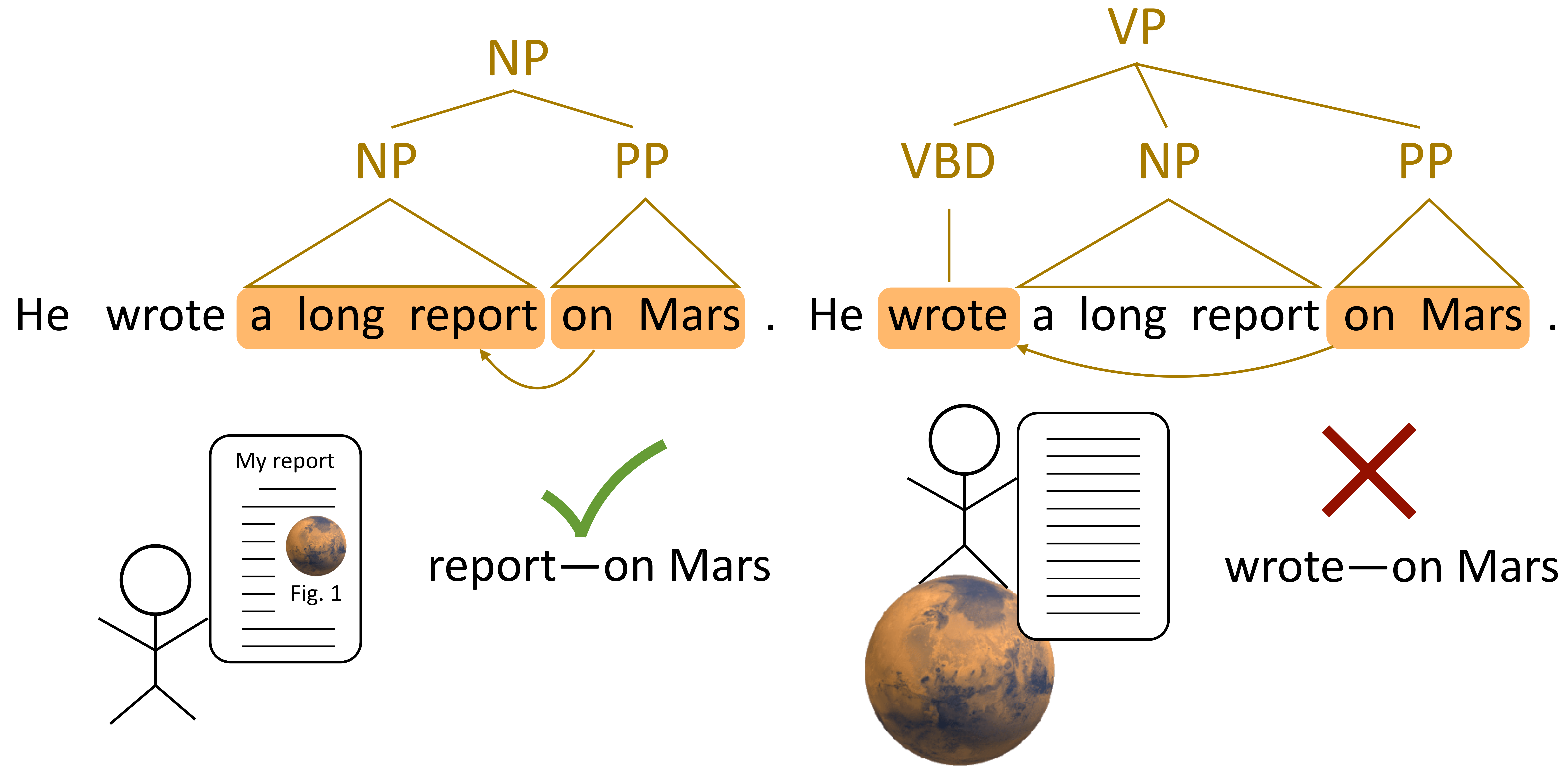
Lexicalized Parsers

- Annotate each grammar symbol with its “head word”: most important word of that constituent
- Rules for identifying headwords (e.g., the last word of an NP before a preposition is typically the head)
- Collins and Charniak (late 90s): ~89 F1 with these



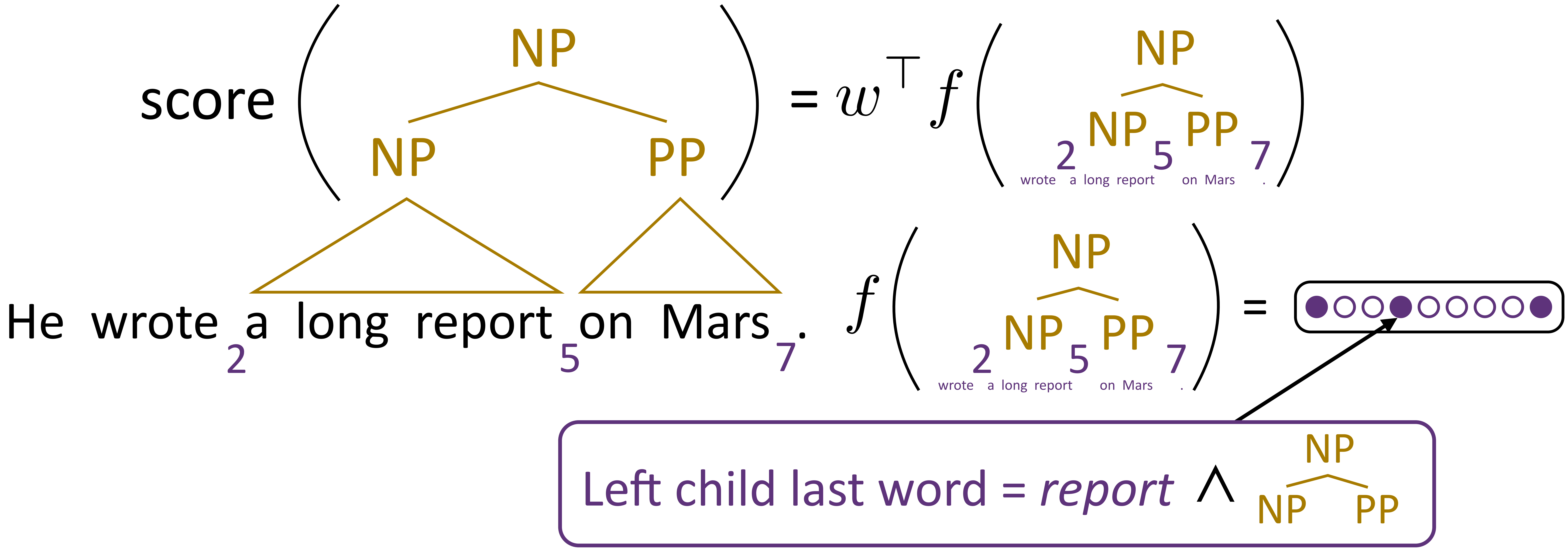
Discriminative Parsers

CRF Parsing



NP
NP
PP
He wrote a long report on Mars .
VP
VBD
NP
PP
He wrote a long report on Mars .
My report
Fig. 1
report—on Mars
wrote—on Mars

CRF Parsing

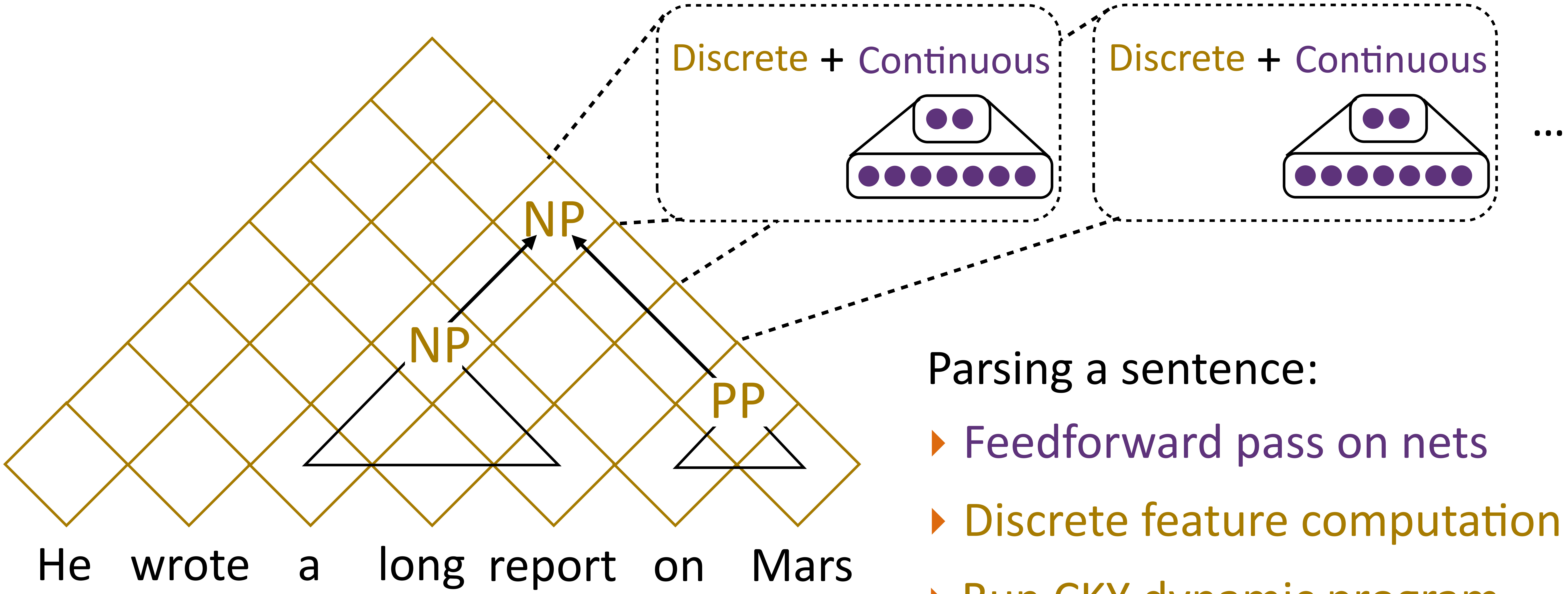


- Can learn that we *report* [PP], which is common due to *reporting on* things
- Can “neuralize” this as well like neural CRFs for NER

Taskar et al. (2004)
Hall, Durrett, and Klein (2014)
Durrett and Klein (2015)

Joint Discrete and Continuous Parsing

- Chart remains discrete!



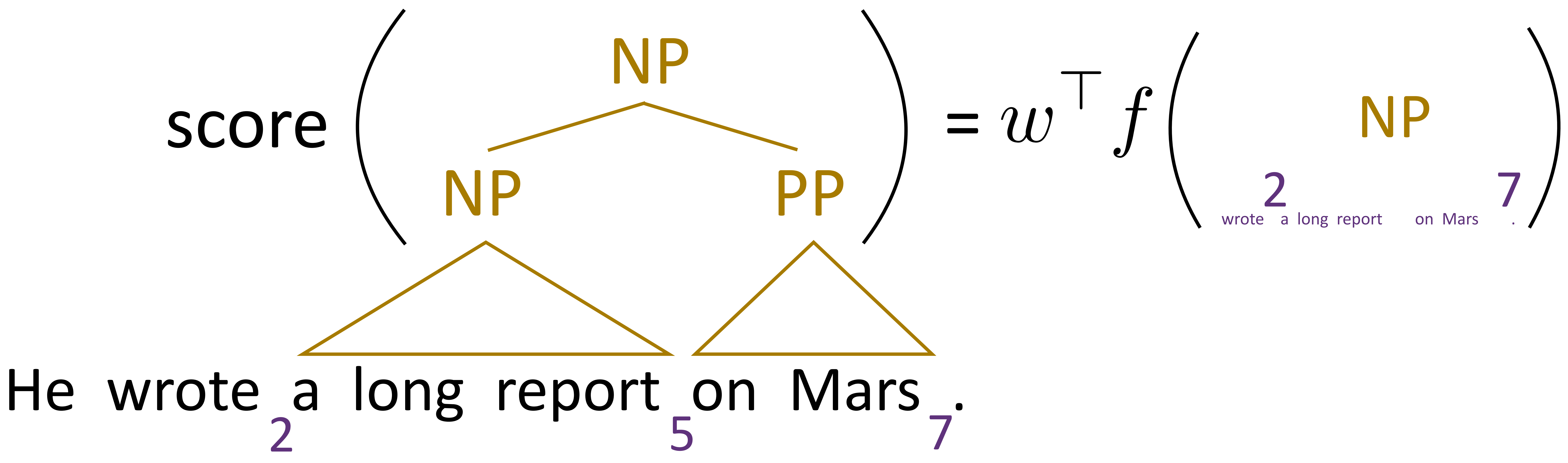
Parsing a sentence:

- Feedforward pass on nets
- Discrete feature computation
- Run CKY dynamic program

Durrett and Klein (ACL 2015)

Neural CRF Parsing

- Simpler version: score *constituents* rather than rule applications



- Use BiLSTMs (Stern) or self-attention (Kitaev) to compute span embeddings
- 91-93 F1, 95 F1 with ELMo (SOTA). Great on other langs too!

Stern et al. (2017), Kitaev et al. (2018)

Takeaways

- PCFGs estimated generatively can perform well if sufficiently engineered
- Neural CRFs work well for constituency parsing
- Next time: revisit lexicalized parsing as *dependency parsing*

Survey

- Write one thing you like about the class
- Write one thing you don't like about the class