Trees!

Jacob Andreas / MIT 6.804-6.864 / Spring 2021

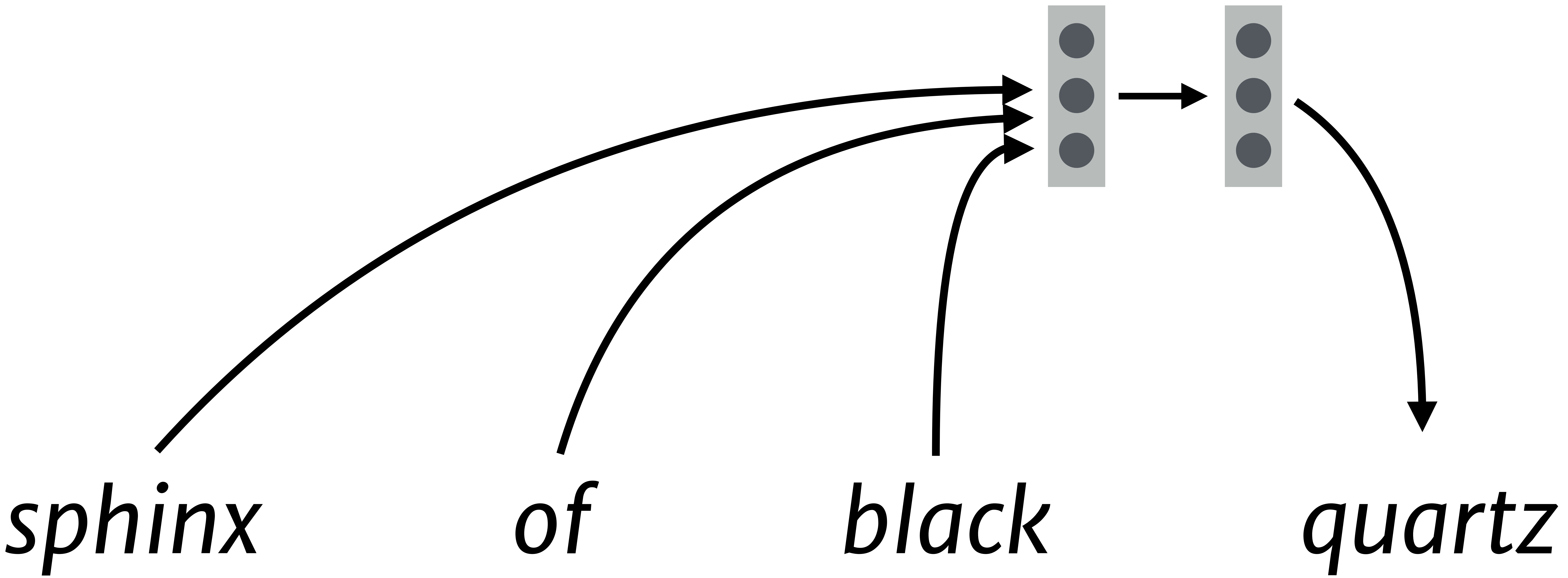
Recap: labels and sequences

Predicting labels

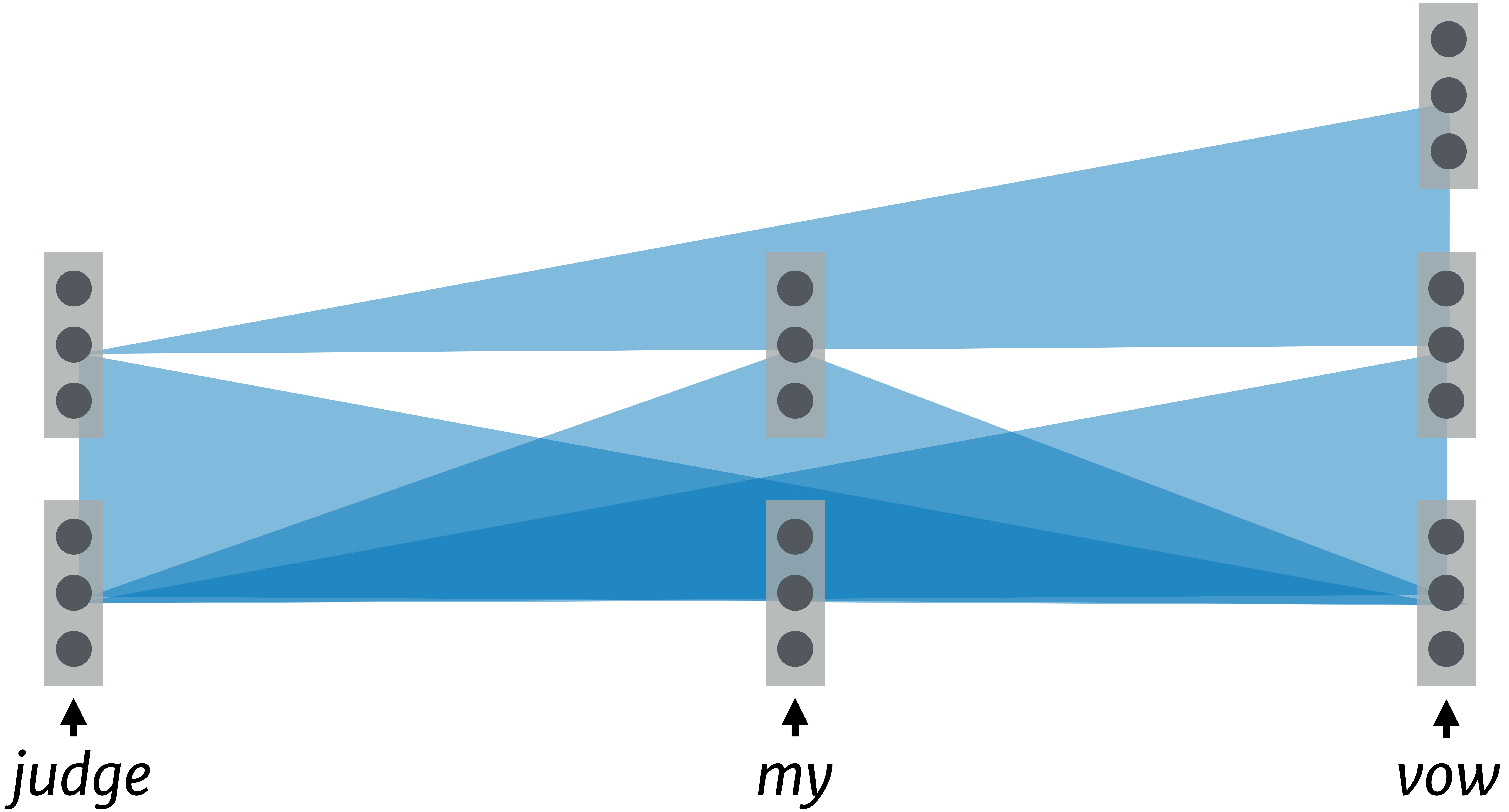
$$s = W_2^\top f(W_1^\top x)$$

x	$f(W_1^\top x) = h_1$	$W_2^\top h_1 = s$
$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ input	$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ "hidden layer"	$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ output

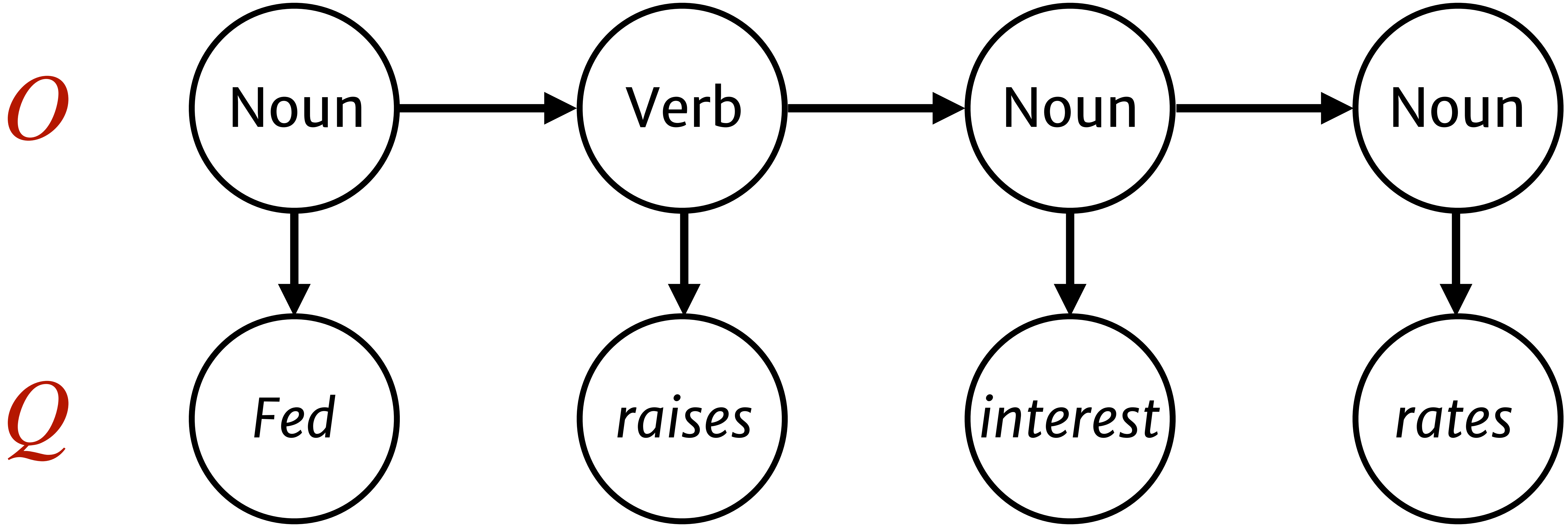
Predicting sequences: n-gram models



Predicting sequences: neural networks

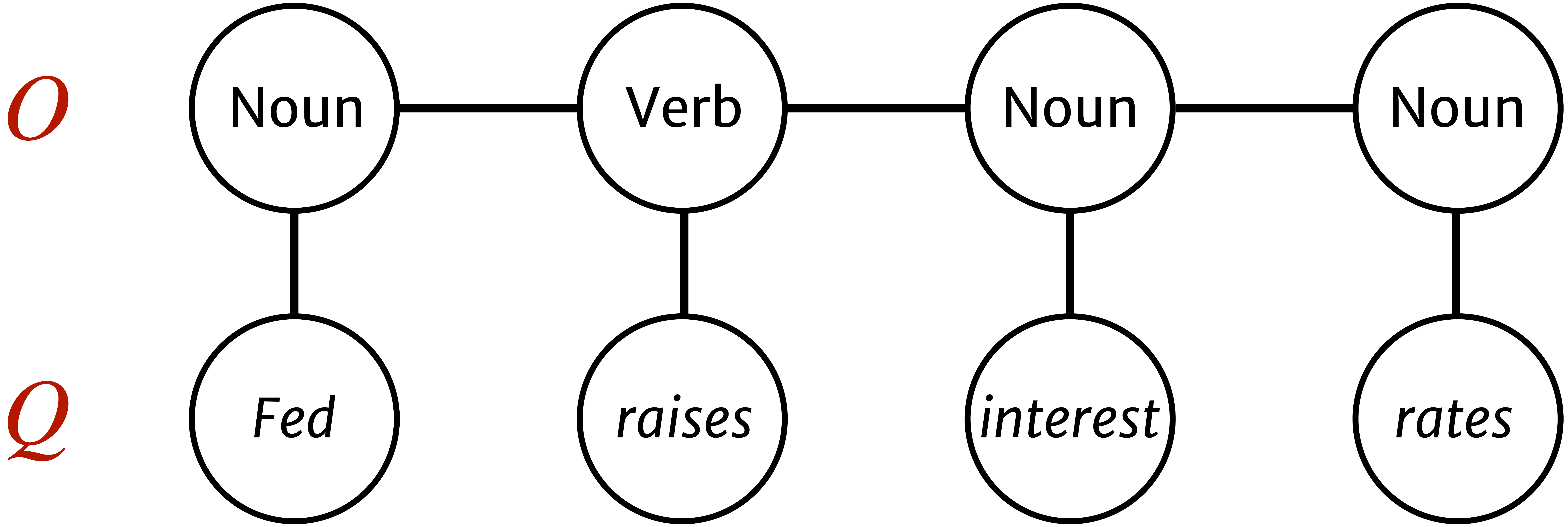


Labeling sequences: HMMs & CRFs



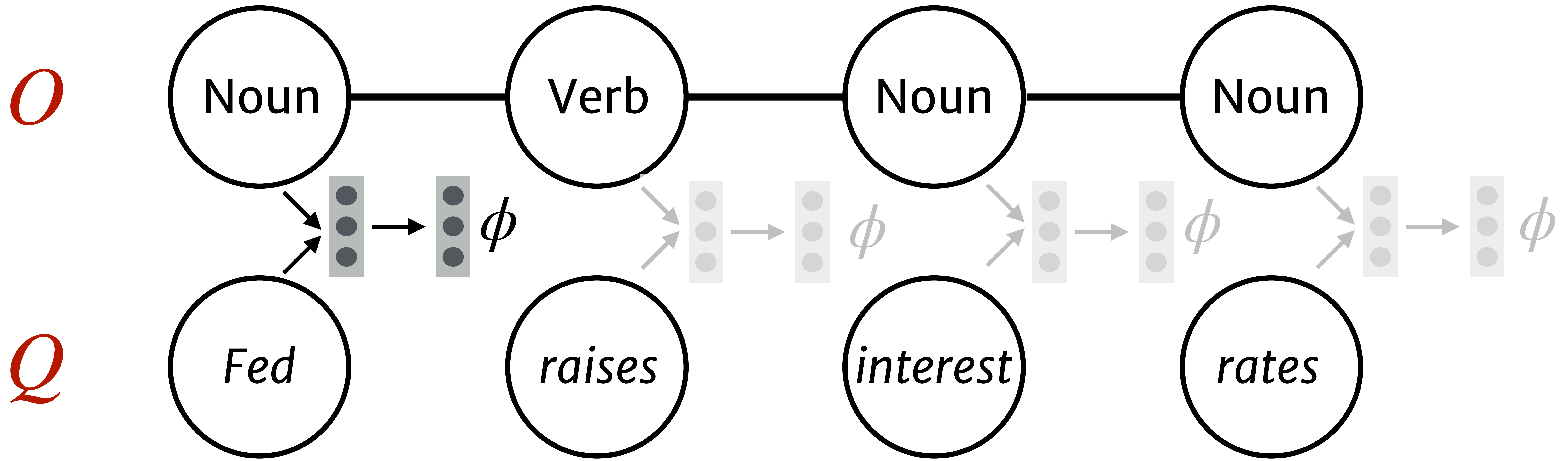
HMM: $p(O, Q) = \prod_t p(q_t \mid q_{:t-1})\, p(o_t \mid q_t)$

Labeling sequences: HMMs & CRFs



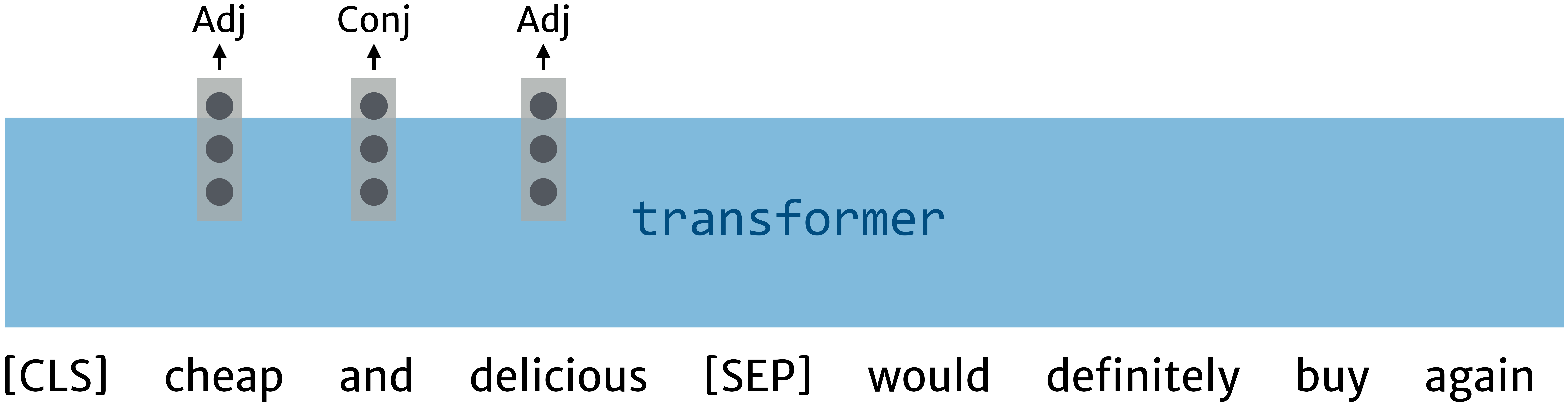
CRF: $$p(O,Q) = \frac{1}{Z}\exp\{\sum_t a^\top \phi(q_t, q_{:t-1}) + b^\top \phi(o_t \mid q_t)\}$$

Labeling sequences: HMMs & CRFs

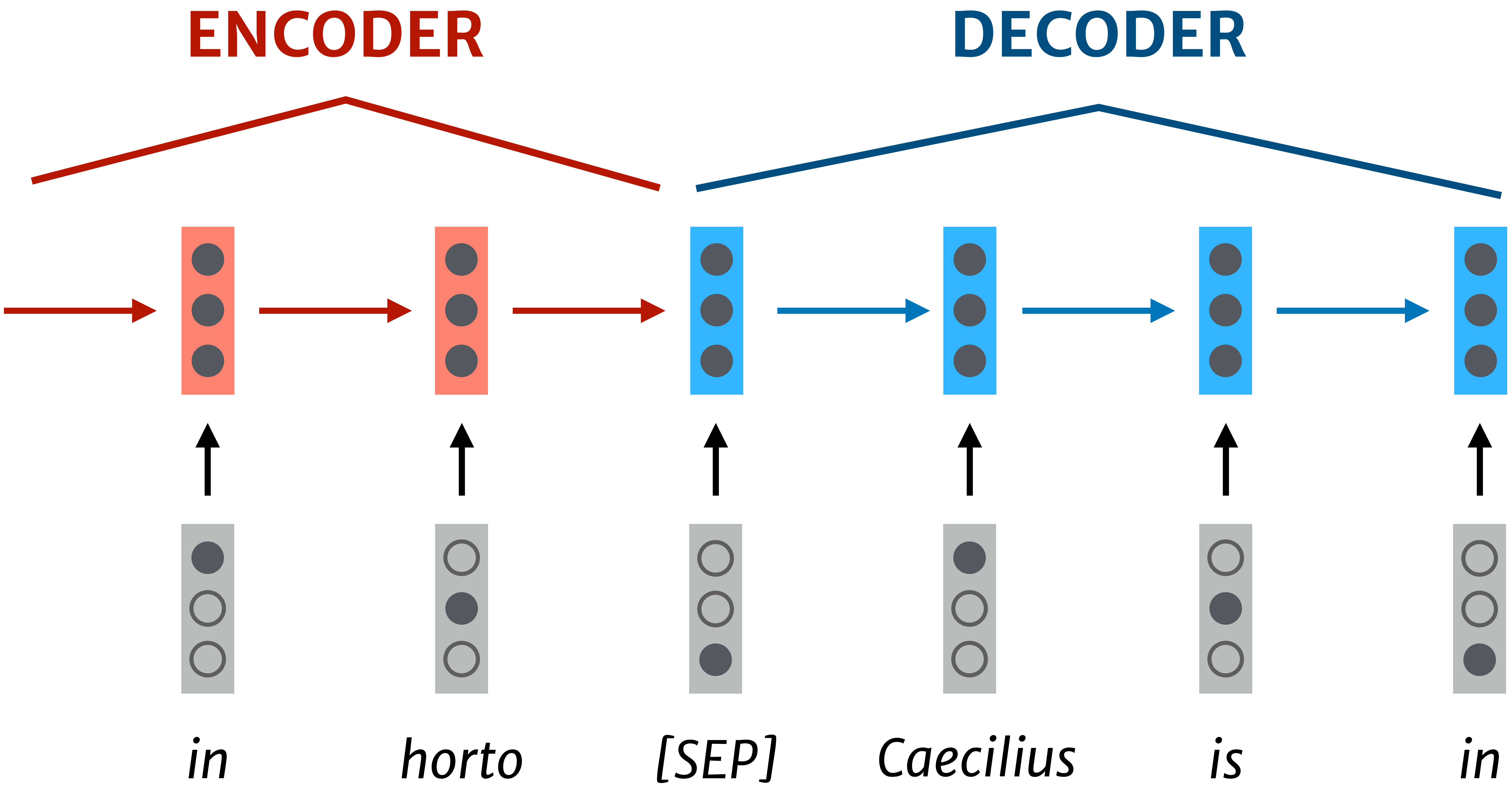


CRF: $p(O,Q) = \frac{1}{Z}\exp\{\sum_t a^\top \phi(q_t, q_{:t-1}) + b^\top \phi(o_t \mid q_t)\}$

Labeling sequences: neural networks

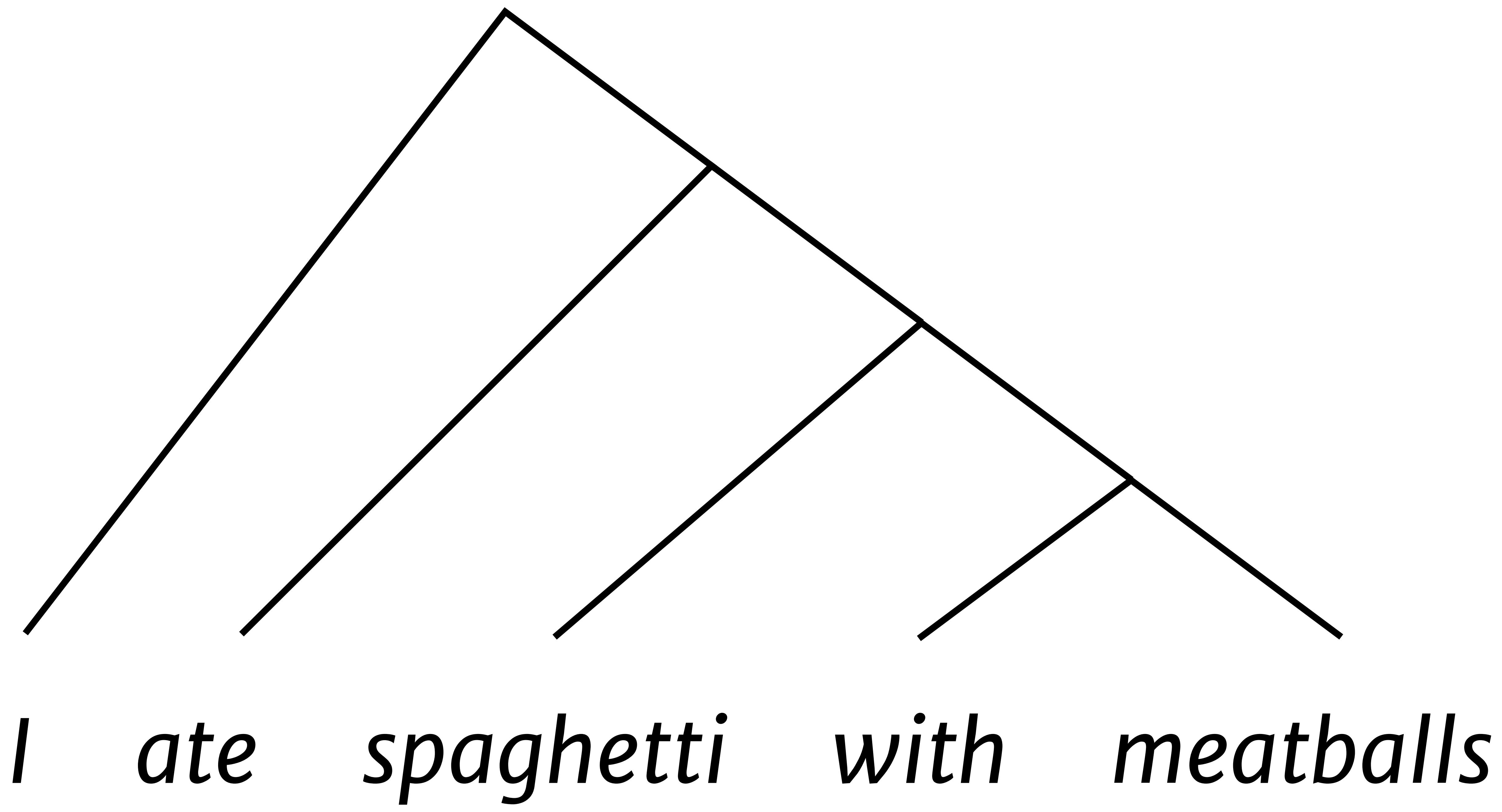


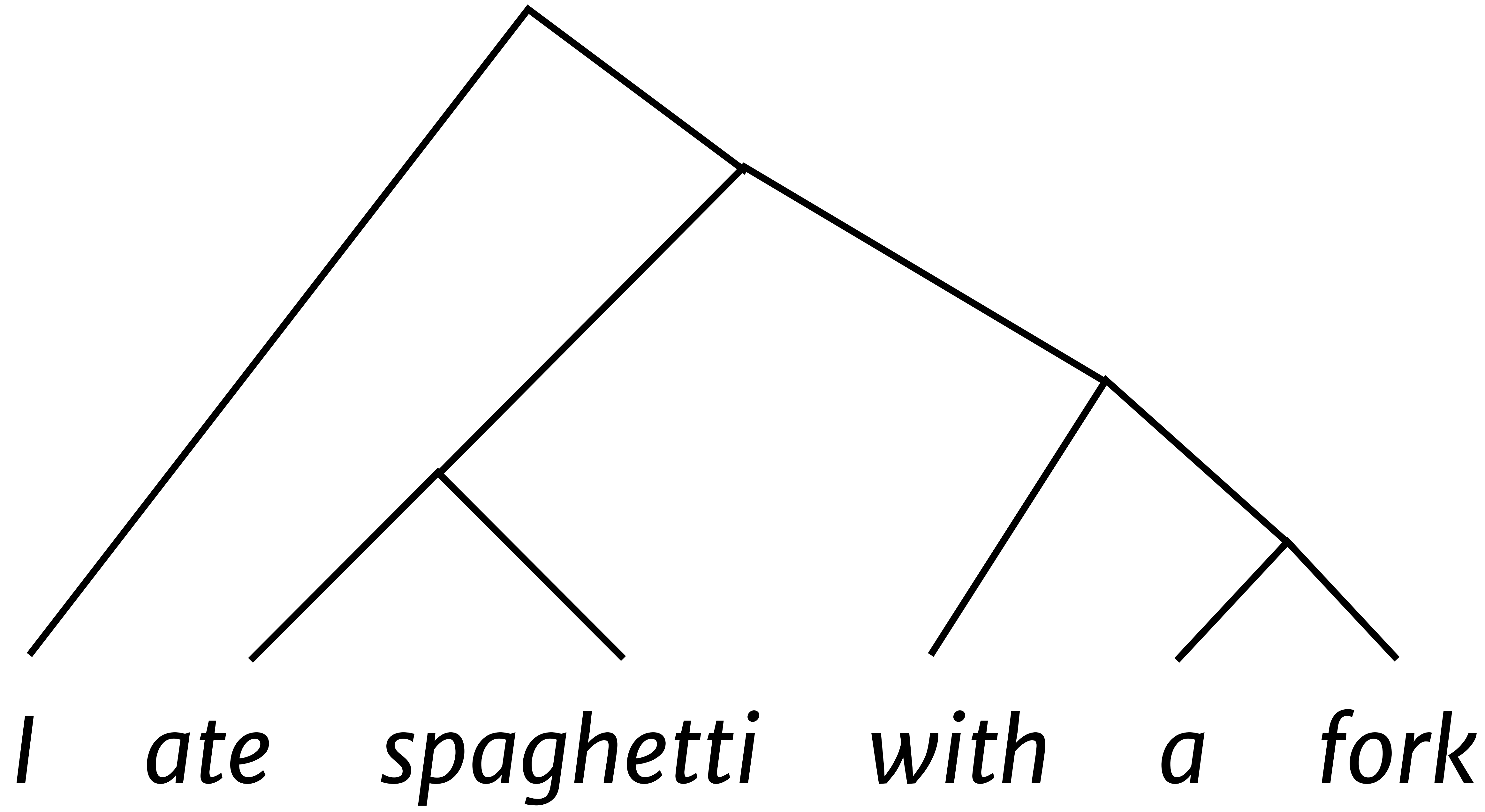
Sequence-to-sequence models



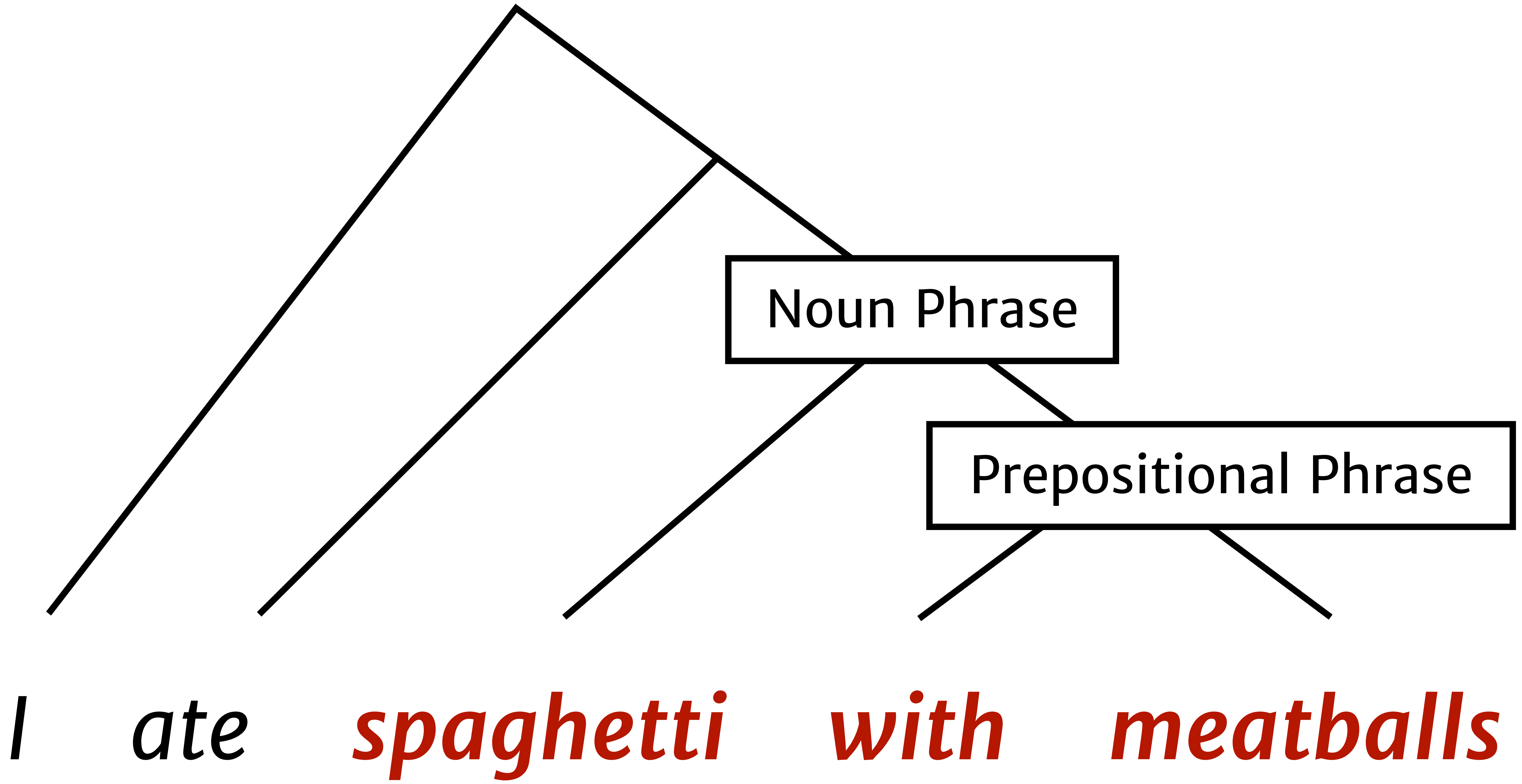
Other structures

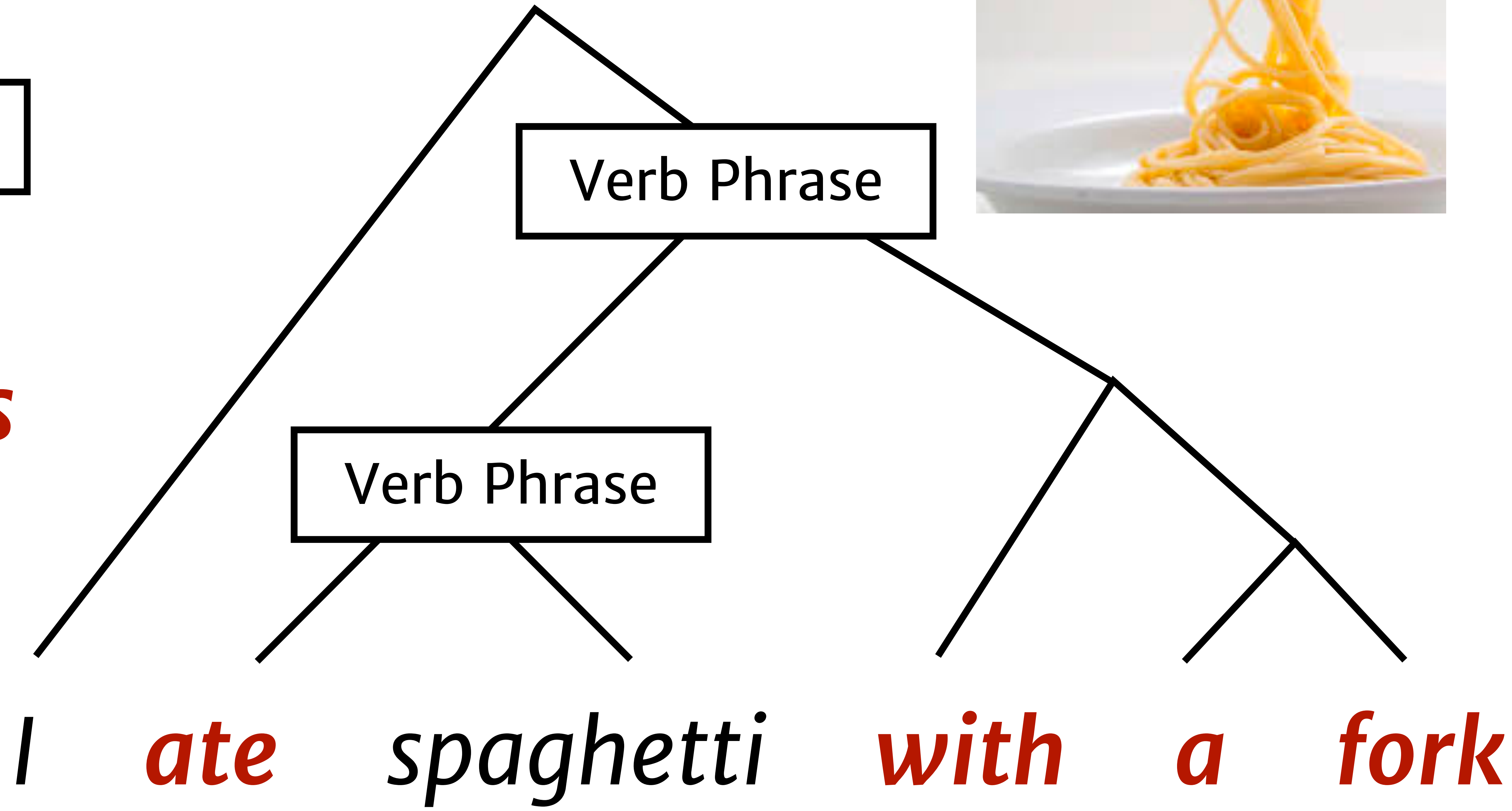
Syntax





Syntax

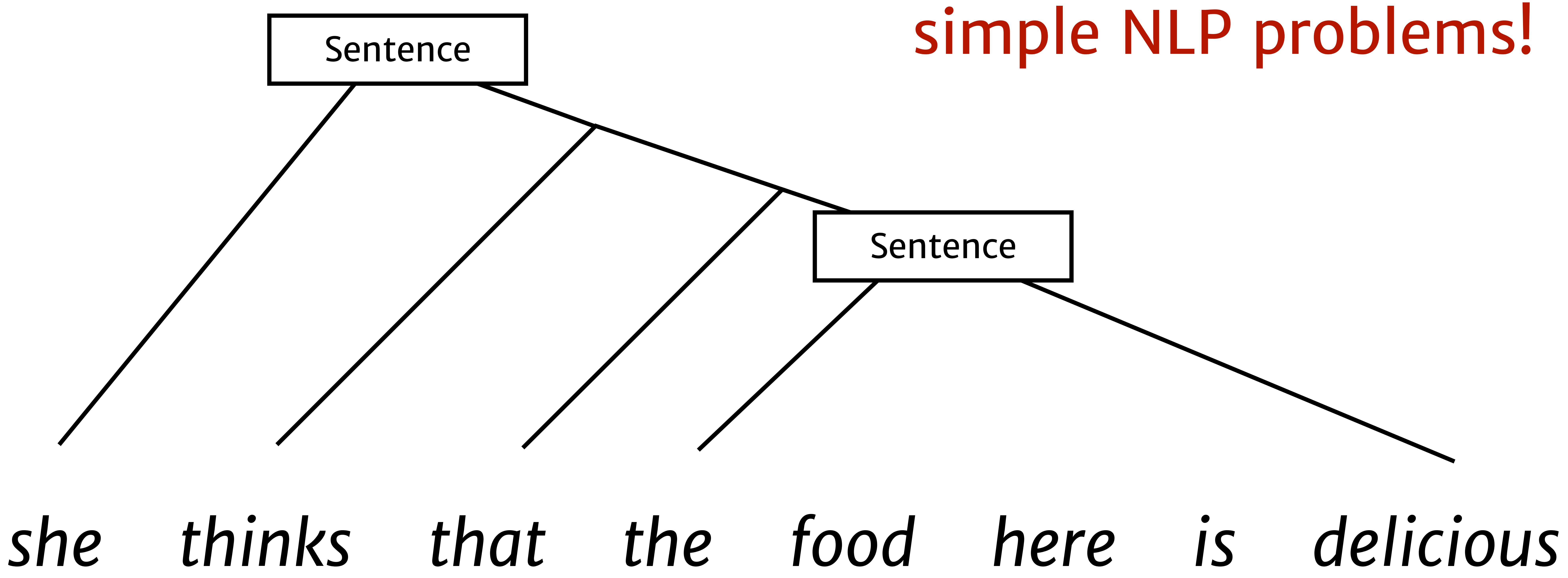




[Images: thespruceeats.com, freepik.com]

Syntax

Useful to distinguish between statements and beliefs, even in simple NLP problems!

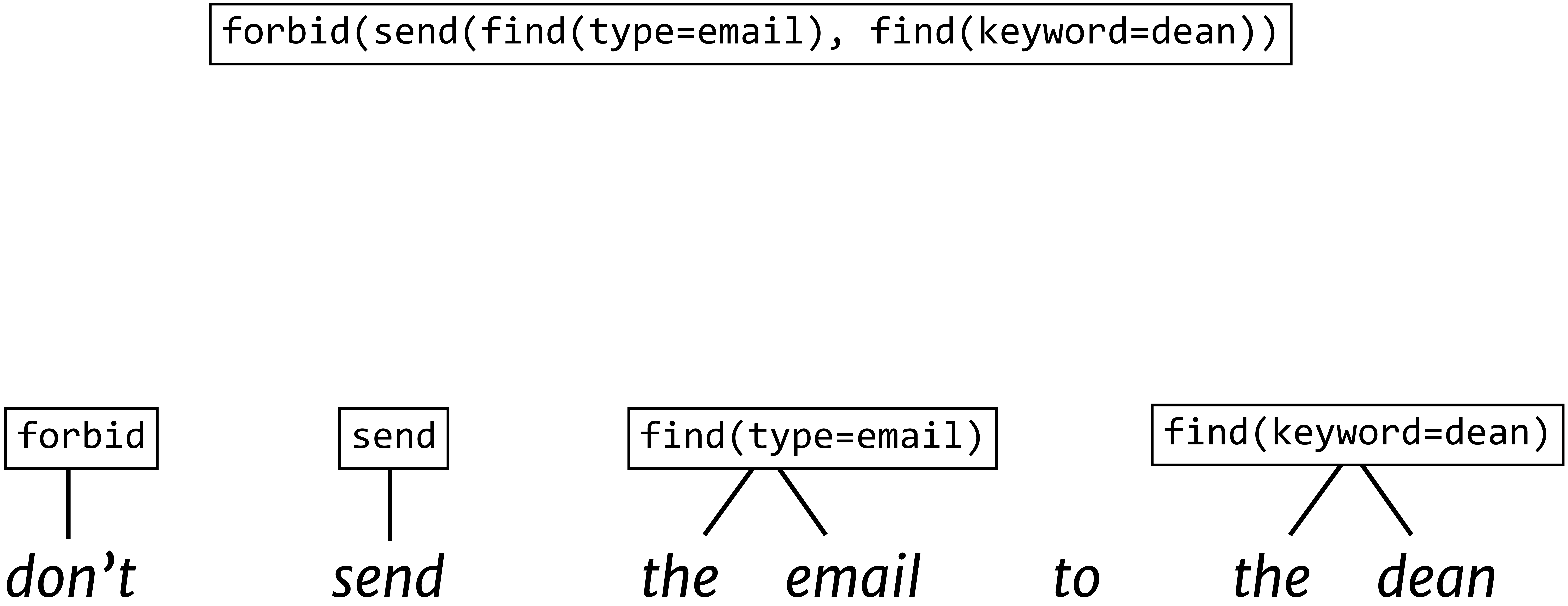


Semantics

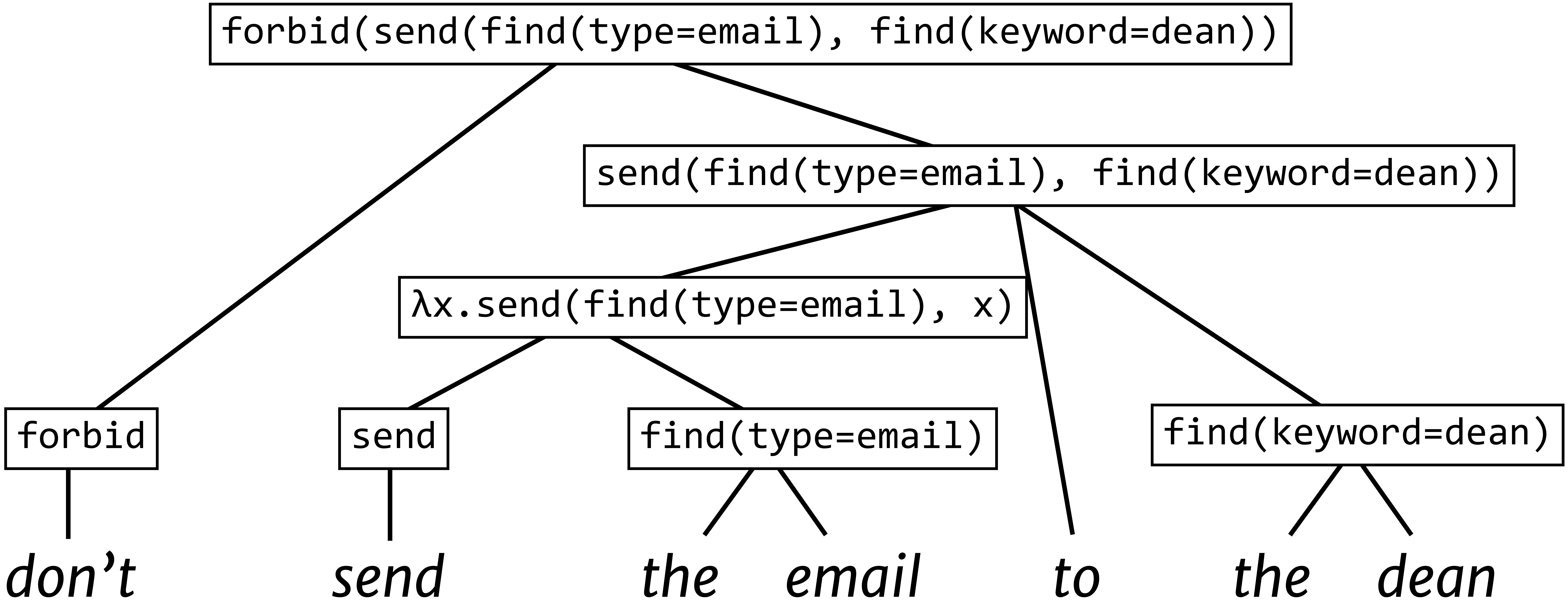
```
forbid(send(find(type=email), find(keyword=dean))
```

don't send the email to the dean

Semantics


forbid(send(find(type=email), find(keyword=dean))
forbid
send
find(type=email)
find(keyword=dean)
don't
send
the
email
to
the
dean

Semantics



forbid(send(find(type=email), find(keyword=dean))
send(find(type=email), find(keyword=dean))
λx.send(find(type=email), x)
forbid
send
find(type=email)
find(keyword=dean)
don't
send
the
email
to
the
dean

Discourse

COMPARISON

The projections are in the neighborhood of 50 cents a share to 75 cents,

CIRCUMSTANCE

compared with a restated $1.65 a share a year earlier,

when profit was $107.8 million on sales of $435.5 million.

[Ji and Eisenstein 2014]

Why trees?

"Simplest" formal generative process that provides **hierarchical relationships** and **long-distance dependencies:**

My relative gave me a microscope.

My aunt's sister gave me a microscope.

My aunt's sister, who works at the NIH, gave me a microscope.

My aunt's sister, who works at a little-known constituent institute of th

Syntax in ten minutes

Constituents

Key idea from previous examples: some sentence fragments "stick together"—can be moved around, replaced, and modified without affecting meaning / grammaticality:

I ate ***spaghetti with meatballs***

I ate

I ate ***it***

It was ***spaghetti with meatballs*** *that I ate*

Constituents

Some fragments are harder to manipulate:

I ***ate spaghetti with*** *meatballs*

I ***ate*** *meatballs* ✘ (meaning changed)

It was ***ate spaghetti with*** *that I meatballs* ✘ (not grammatical)

Constituents

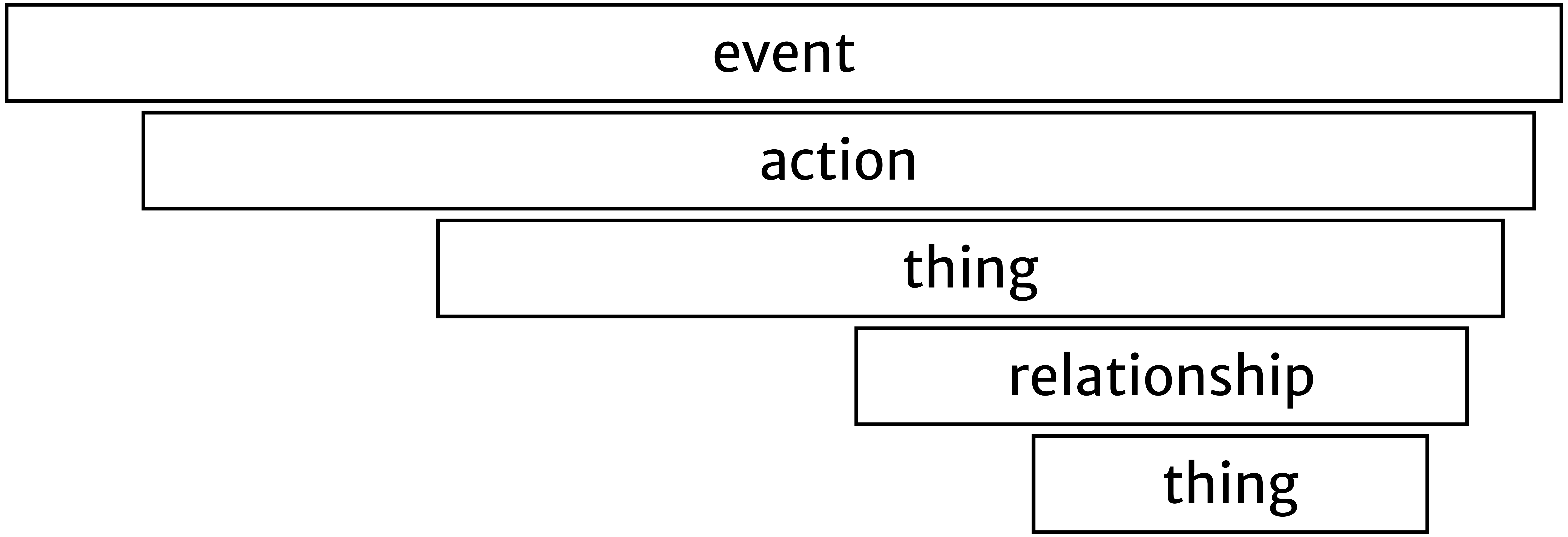
Not just things:

I ate spaghetti ***with a fork***

I ate spaghetti

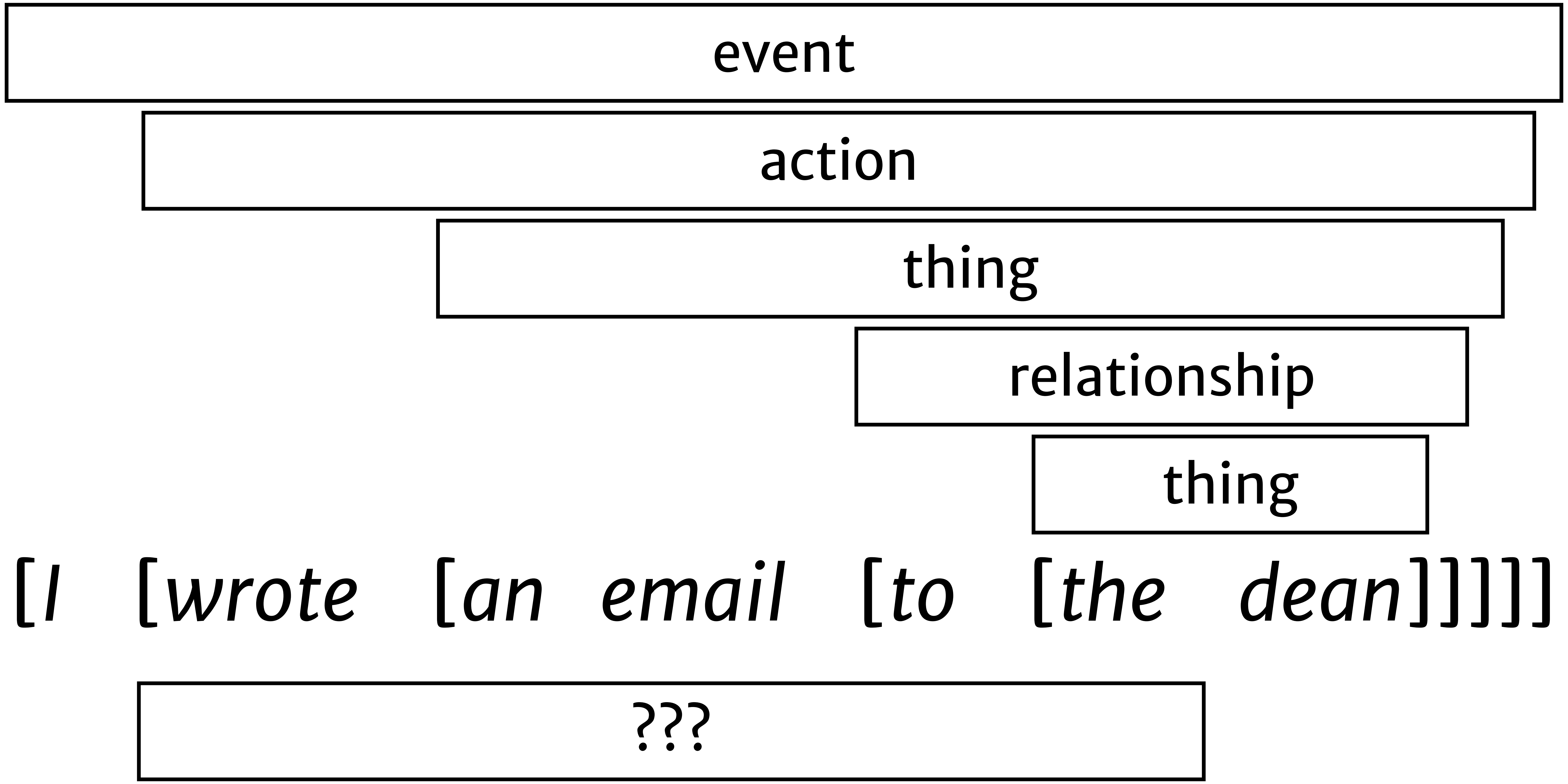
It was ***with a fork*** *that I ate spaghetti*

Constituents & Types

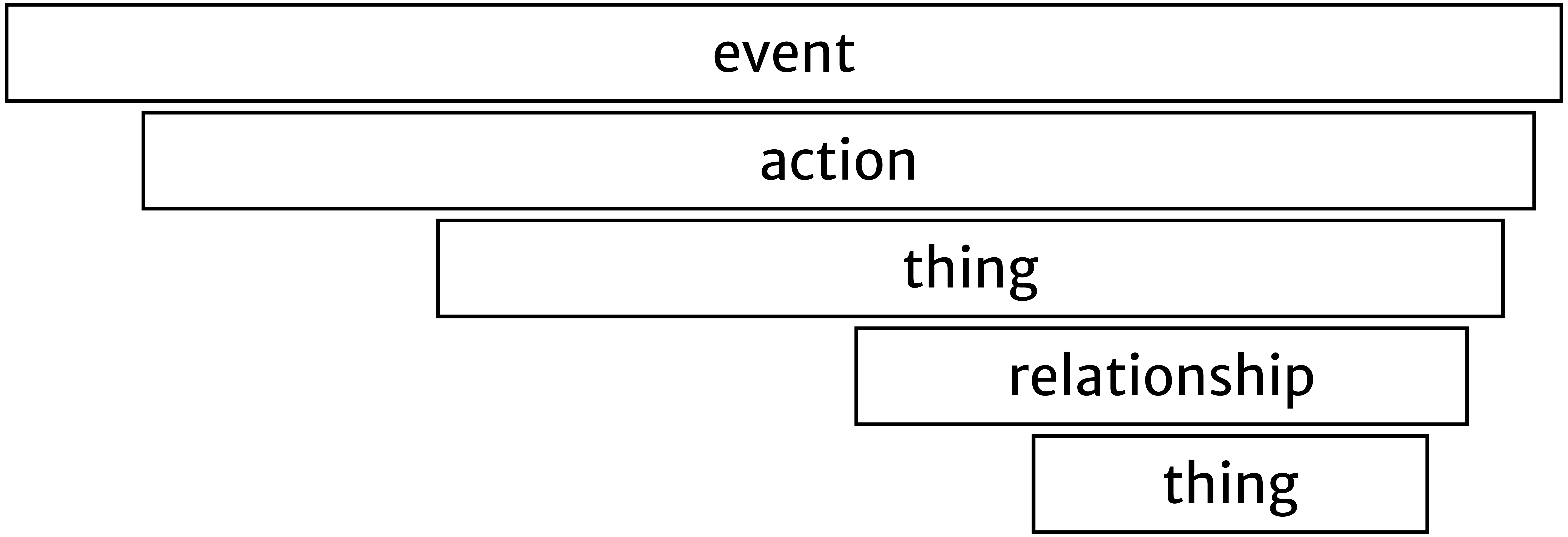


[*I* [*wrote* [*an email* [*to* [*the dean*]]]]]

Constituents & Types



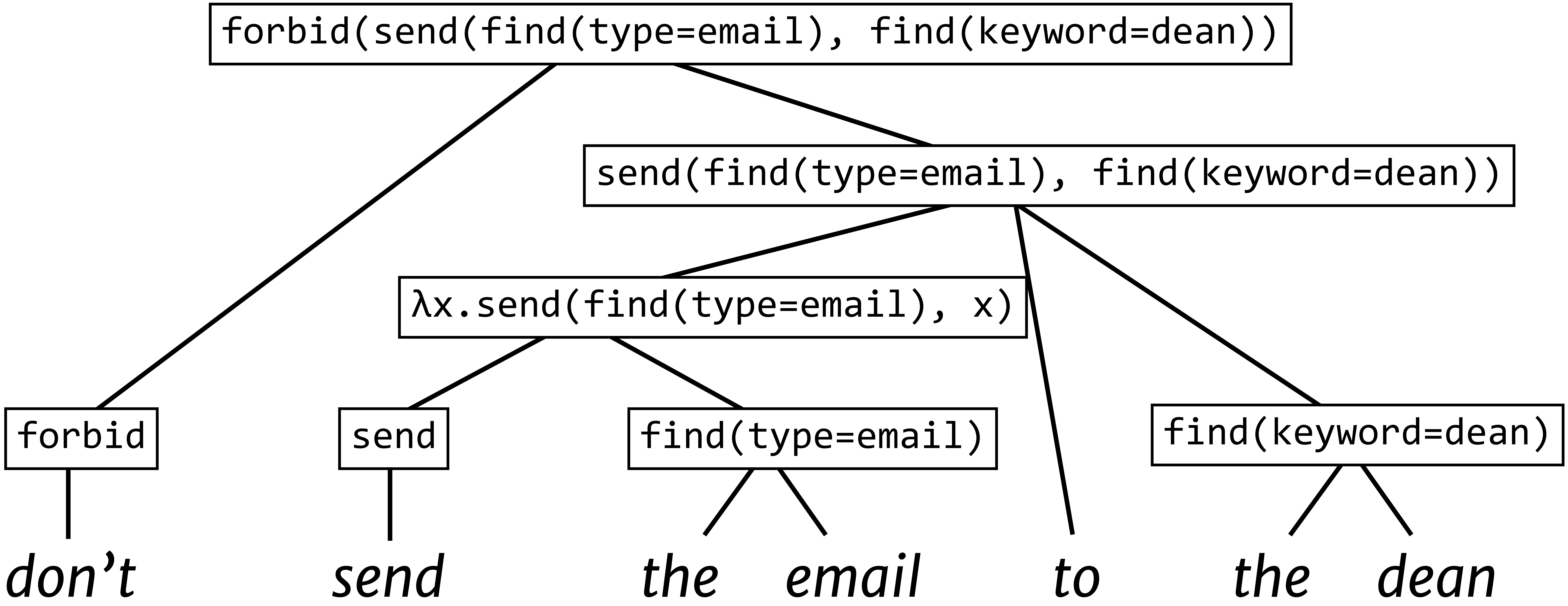
Constituents & Types



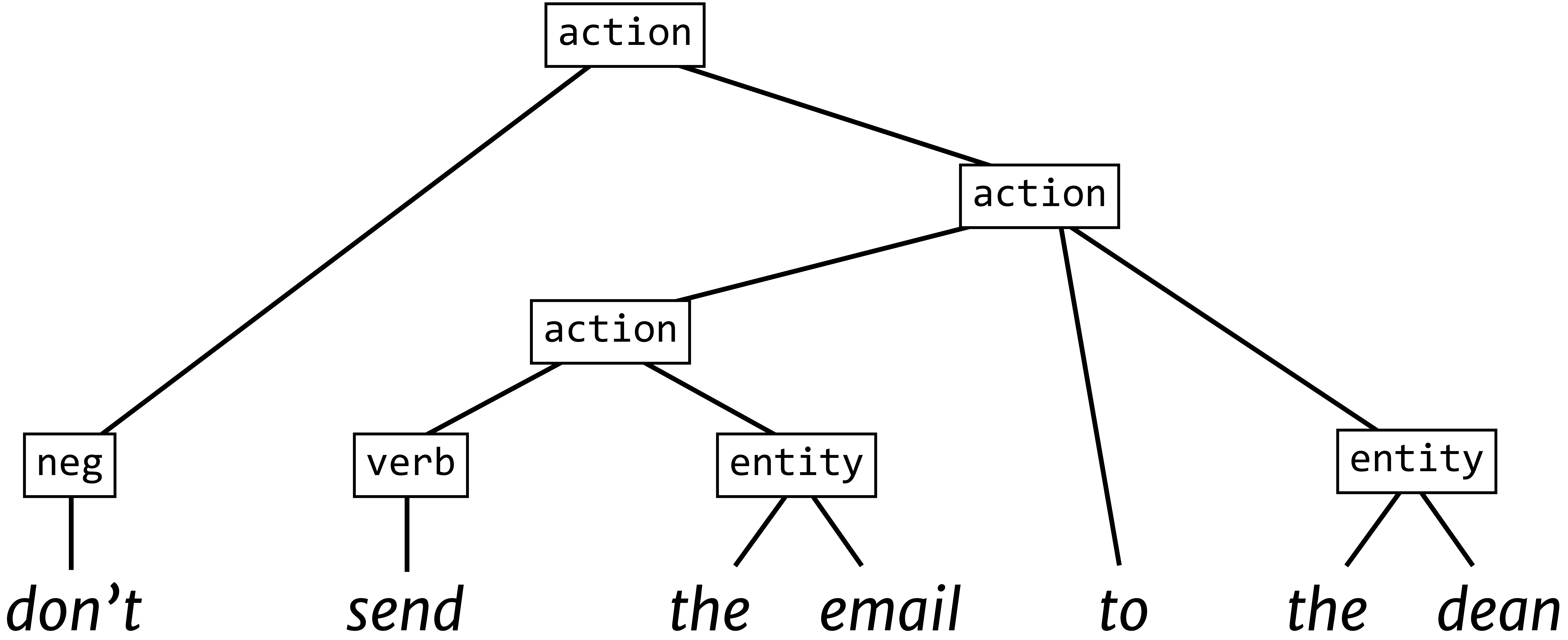
[*I* [*wrote* [*an* *email* [*to* [*the* *dean*]]]]]

Lots of research on the exact form of this hierarchy.
For most NLP applications: **entities, events, relations.**

Types & semantics



Types & semantics



Context free grammars

Just like in HMMs, we'd like to define some joint distribution over sentences and underlying structures, and reason about marginals and conditionals.

What's the right distribution over trees and sentences?

Context free grammars

A **sentence** might consist of an entity and an action.

[*I*] [*swallowed the spider*]

Context free grammars

A **sentence** might consist of an entity and an action.

[*I*] [*swallowed the spider*]

S → NP VP   a Noun Phrase followed by a Verb Phrase make a Sentence

A **sentence** might just consist of an action.

[*eat the spider*]

S → VP   a Verb Phrase makes a Sentence

Context free grammars

S → NP VP | VP

"or"

the followed by a noun makes an entity

NP → *the* N

N → *cat* | *dog* | *spider* | *cheesecake* | *democracy*

a verb and an optional entity make an action

VP → V | V NP

V → eat | eats | run | differentiate | …

A sample from our CFG

S	→	NP VP | VP
NP	→	*the* N
N	→	*cat* | *dog* | *spider* | *cheesecake* | *democracy*
VP	→	V | V NP
V	→	eat | eats | run | differentiate | …

A sample from our CFG

S → NP VP | VP
NP → *the* N
N → *cat* | *dog* | *spider* | *cheesecake* | *democracy*
VP → V | V NP
V → eat | eats | run | differentiate | ...

S

NP VP

the N VP

the cat VP

the cat V NP

the cat eats NP

the cat eat the N

the cat eats the democracy

A sample from our CFG

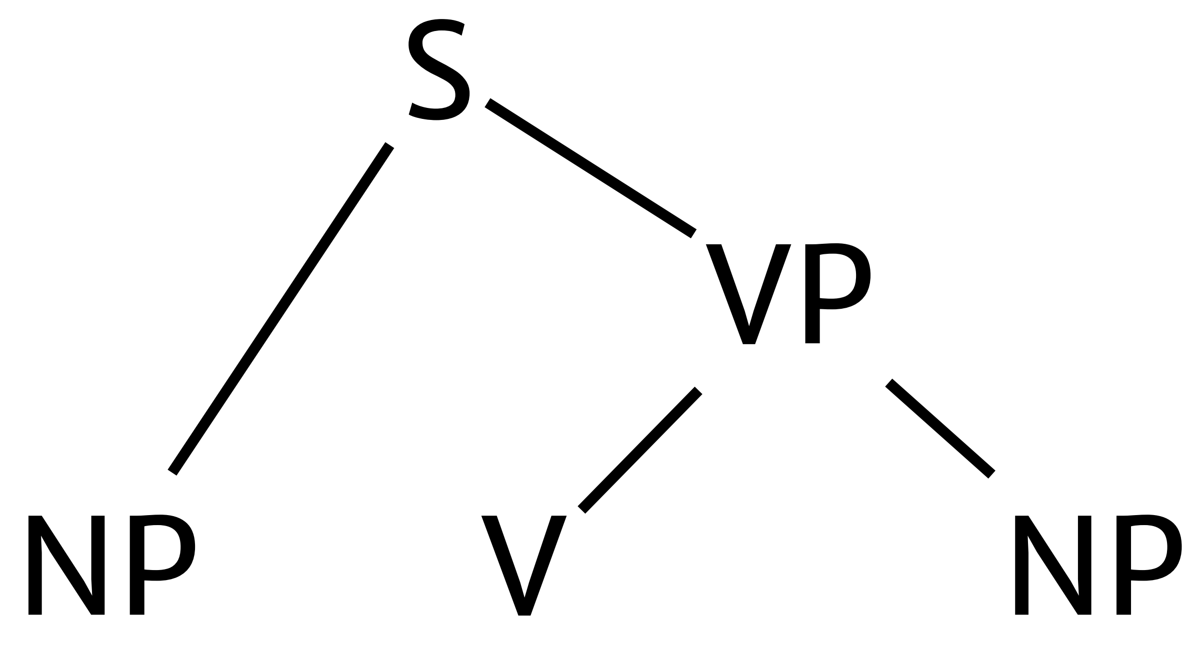
S → NP VP | VP

NP → *the* N

N → *cat* | *dog* | *spider* | *cheesecake* | *democracy*

VP → V | V NP

V → eat | eats | run | differentiate | ...



the cat eats the democracy

What about other languages?

a	*taky*	*na*	*to*	*většinou*	*nemá*	*peníze*
and	also	for	it	generally	hasn’t	money

and in most cases he has no money for it either

[McDonald et al 2005]

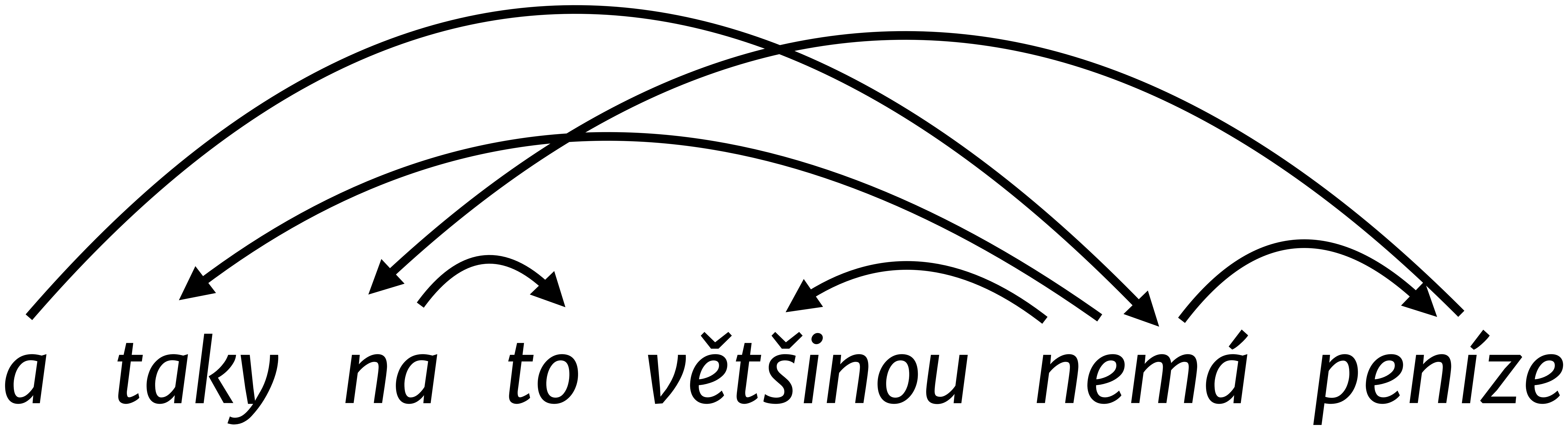
What about other languages?

a	*taky*	*na*	*to*	*většinou*	*nemá*	*peníze*
and	also	for	it	generally	hasn't	money

and in most cases he has no money for it either

can't draw a constituency tree!

Dependency grammar

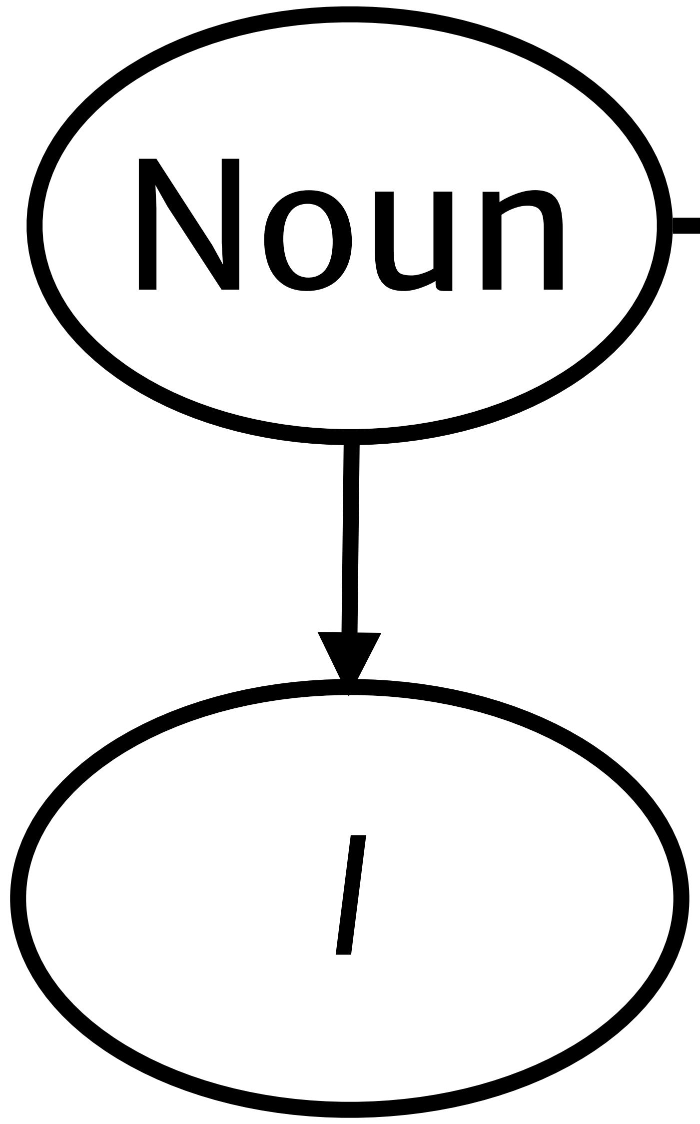


Probabilistic grammars

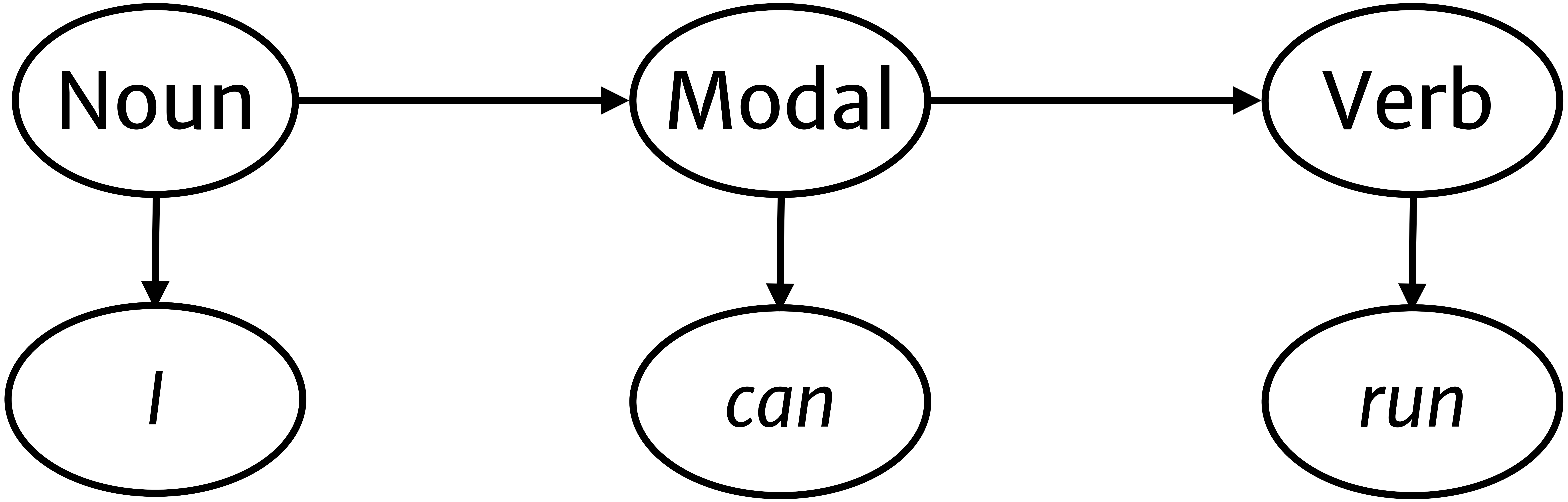
CFGs as generative models

Noun

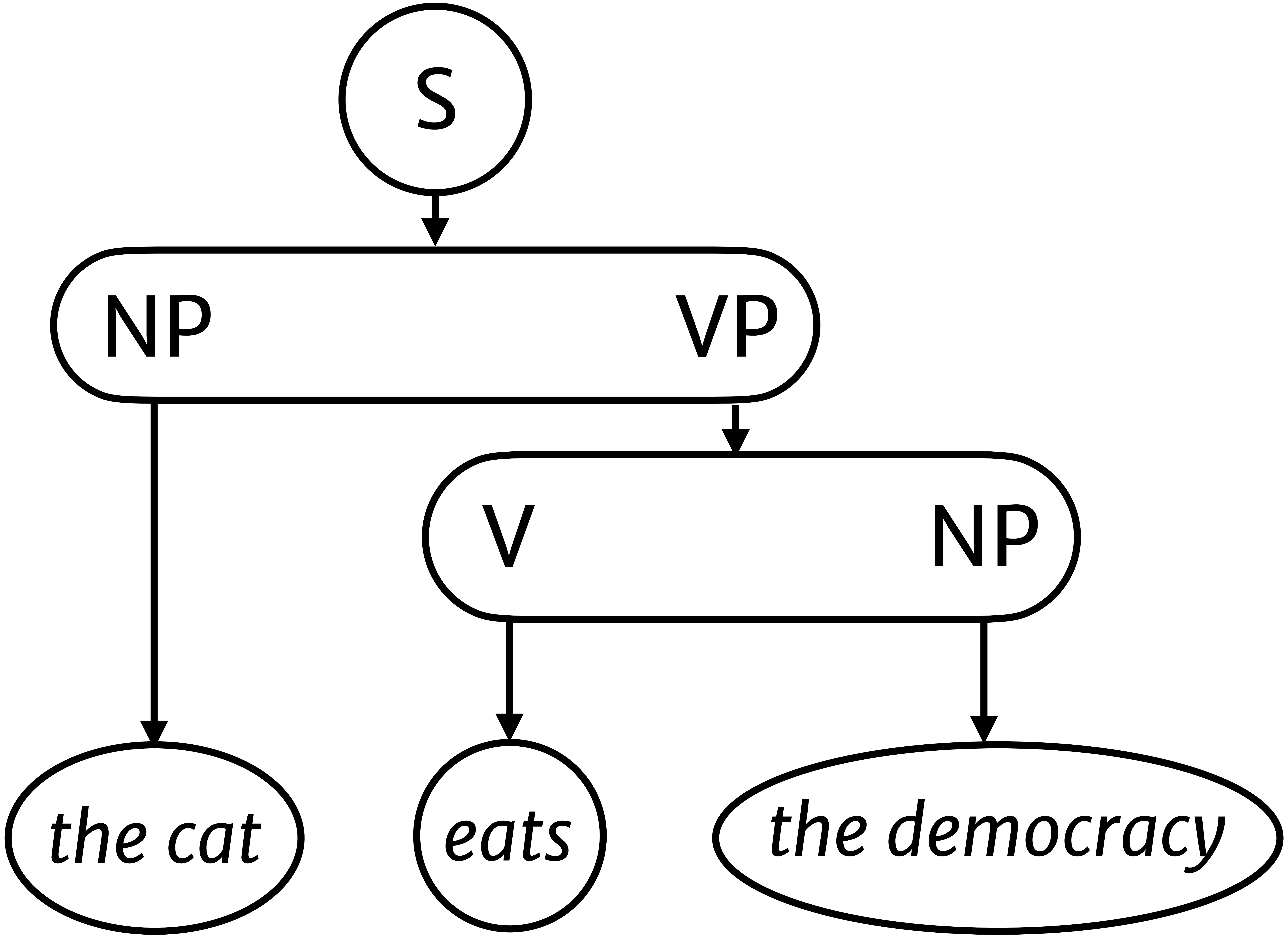
CFGs as generative models



CFGs as generative models



CFGs as generative models



Probabilistic CFGs

A probabilistic context free grammar (PCFG) consists of

1. A set of nonterminal symbols N
2. A set of terminal symbols T
3. A set of rules R
4. A set of rule probabilities $p(r \in R \mid n \in N)$

Probabilistic CFGs

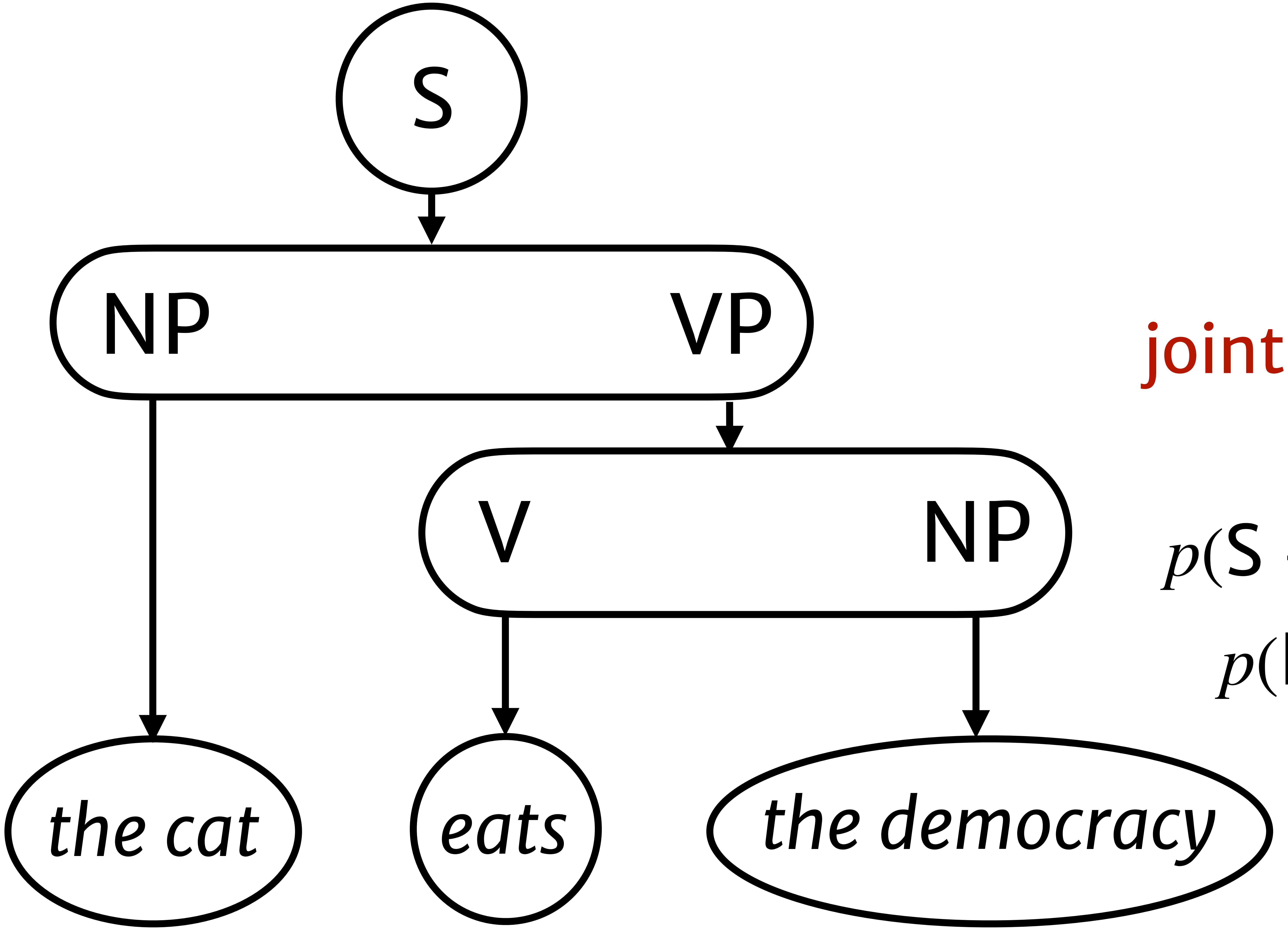
A rule consists of

1. A left hand symbol
2. A sequence of right-hand symbols

LHS		RHS	prob
S	→	NP VP	0.75

such that $\sum_{\text{rules with LHS symbol A}} p(\text{rule} \mid \text{LHS}) = 1$

Queries: joint probability

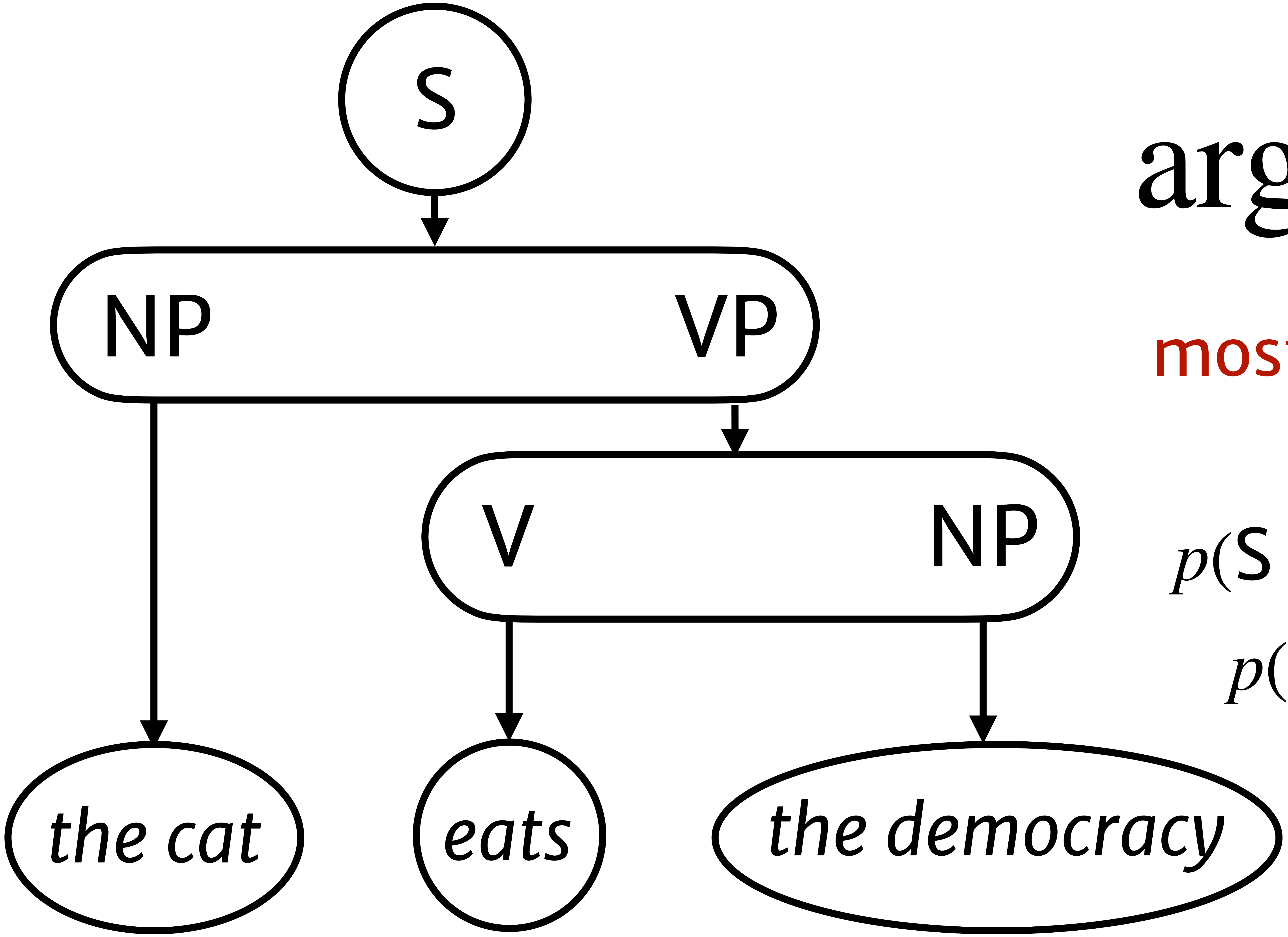


$$p(T, S)$$

joint prob. of tree and sentence

$$p(\mathsf{S} \rightarrow \mathsf{NP\ VP}) \cdot p(\mathsf{NP} \rightarrow \mathsf{D\ N})$$
$$p(\mathsf{D} \rightarrow \mathsf{the}) \cdot p(\mathsf{N} \rightarrow \mathsf{cat}) \;\; \cdots$$

Queries: best tree



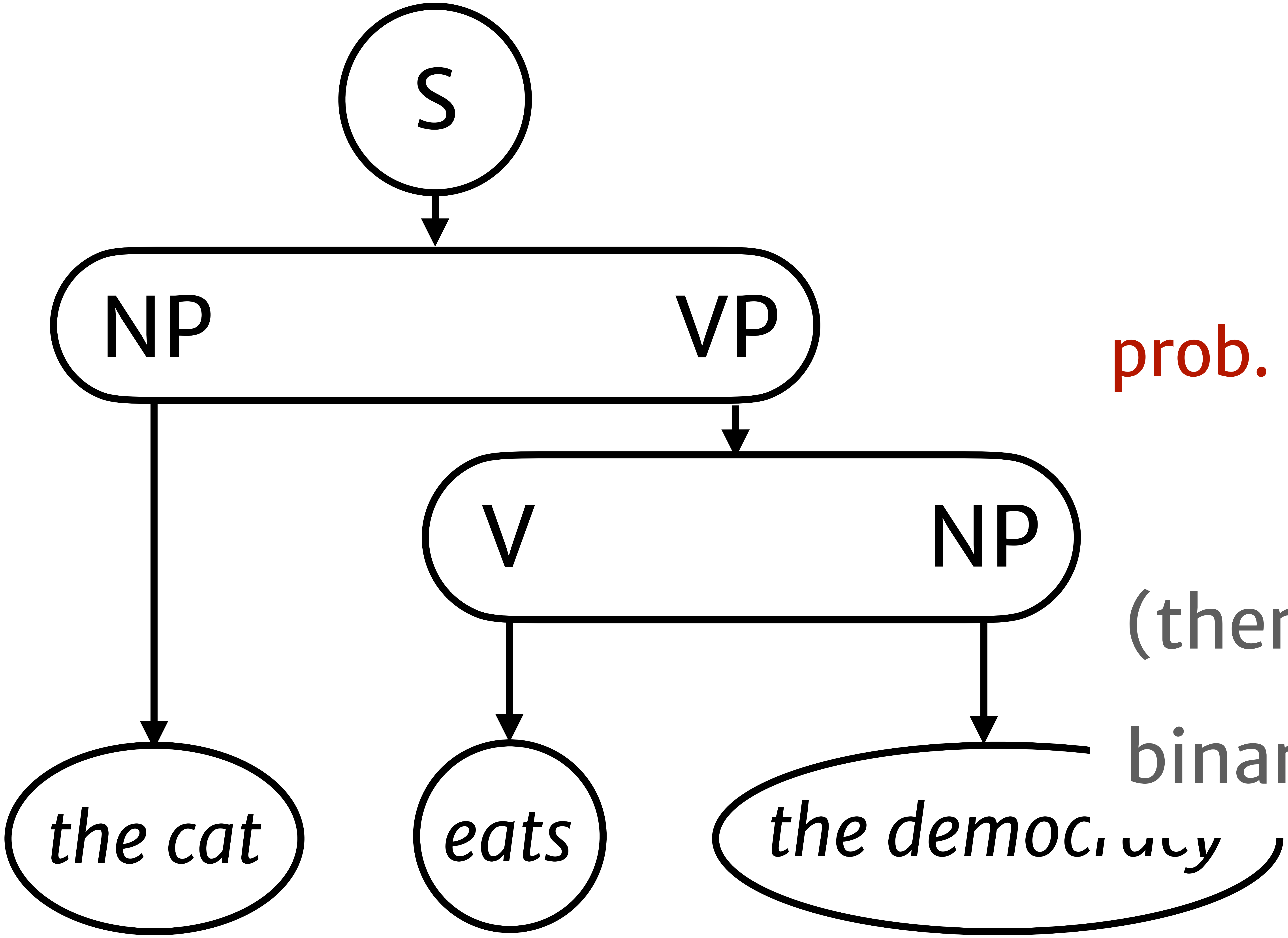
$$\operatorname{argmax}_T \; p(T \mid S)$$

most probable tree given sent.

$p(\text{S} \to \text{NP VP}) \cdot p(\text{NP} \to \text{D N})$

$p(\text{D} \to \text{the}) \cdot p(\text{N} \to \text{cat}) \; \cdots$

Queries: sentence marginal



$$p(S)$$

prob. of sentence under **any** tree

(there are $\frac{(2n)!}{(n+1)!n!}$ *unlabeled* binary trees over *n* words...)

Queries: sentence marginal

$$p(S)$$

prob. of sentence under **any** tree

(there are $\dfrac{(2n)!}{(n+1)!n!}$ *unlabeled* binary trees over n words...)

Parsing

Chomsky normal form

Notational convenience: only binary trees.

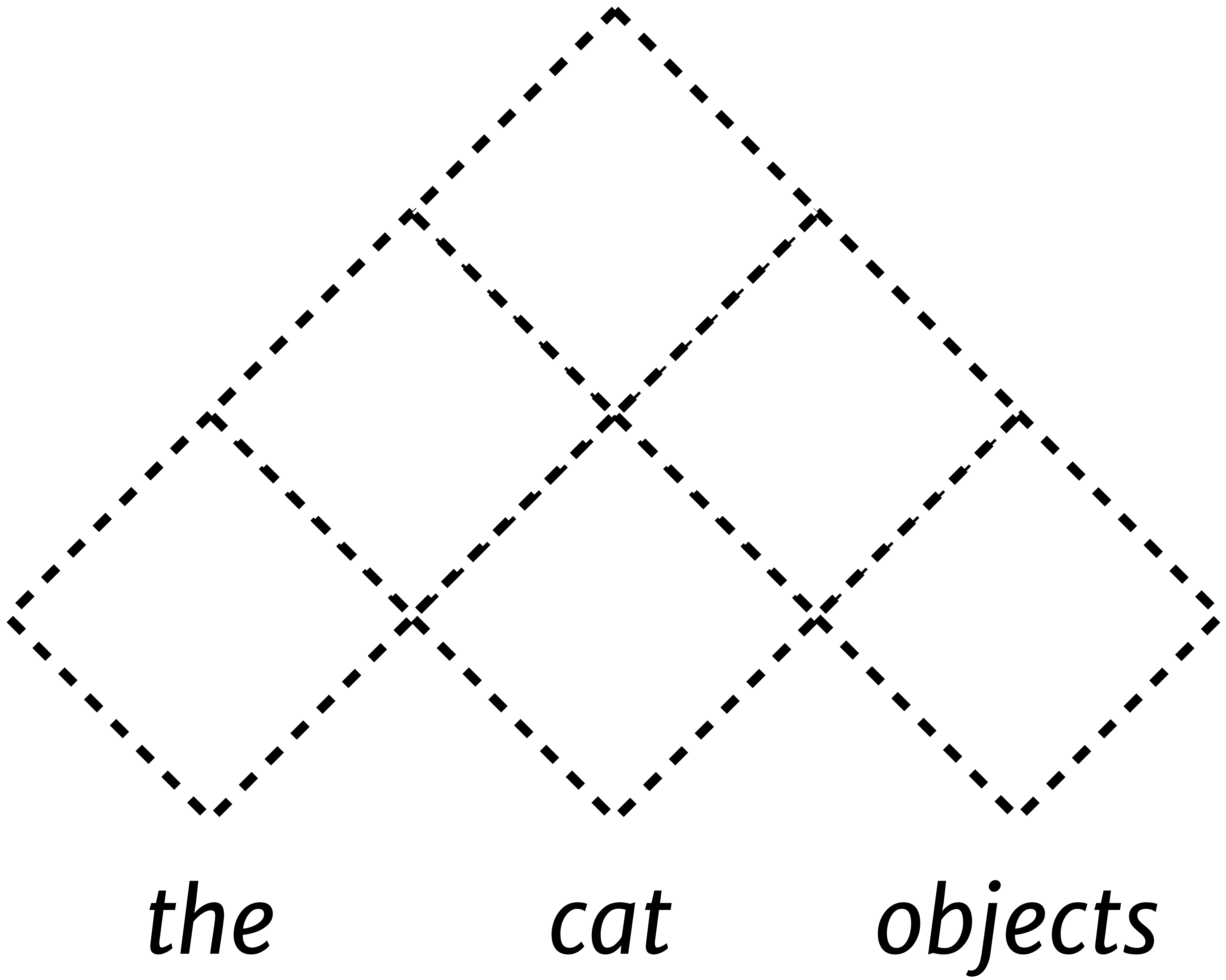
Every rule has one of these forms:

Nonterminal → Terminal
Nonterminal → Nonterminal Nonterminal

(Can always get rules into this form by introducing new NTs)

Warmup: the CKY algorithm

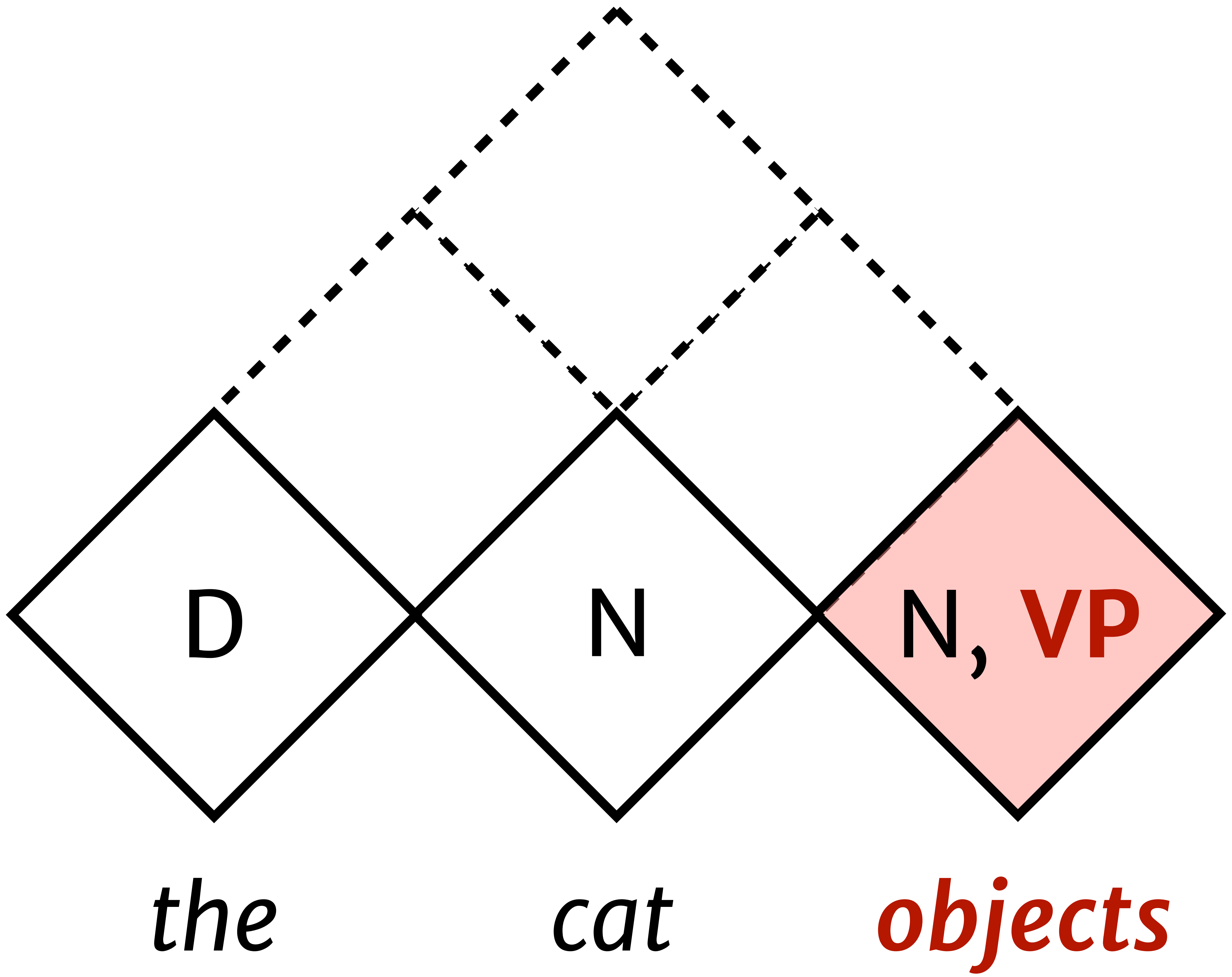
Is the string S generated by CFG G?



S	→	NP VP
NP	→	D N
N	→	cat | objects
VP	→	objects | sings
D	→	the | a

Warmup: the CKY algorithm

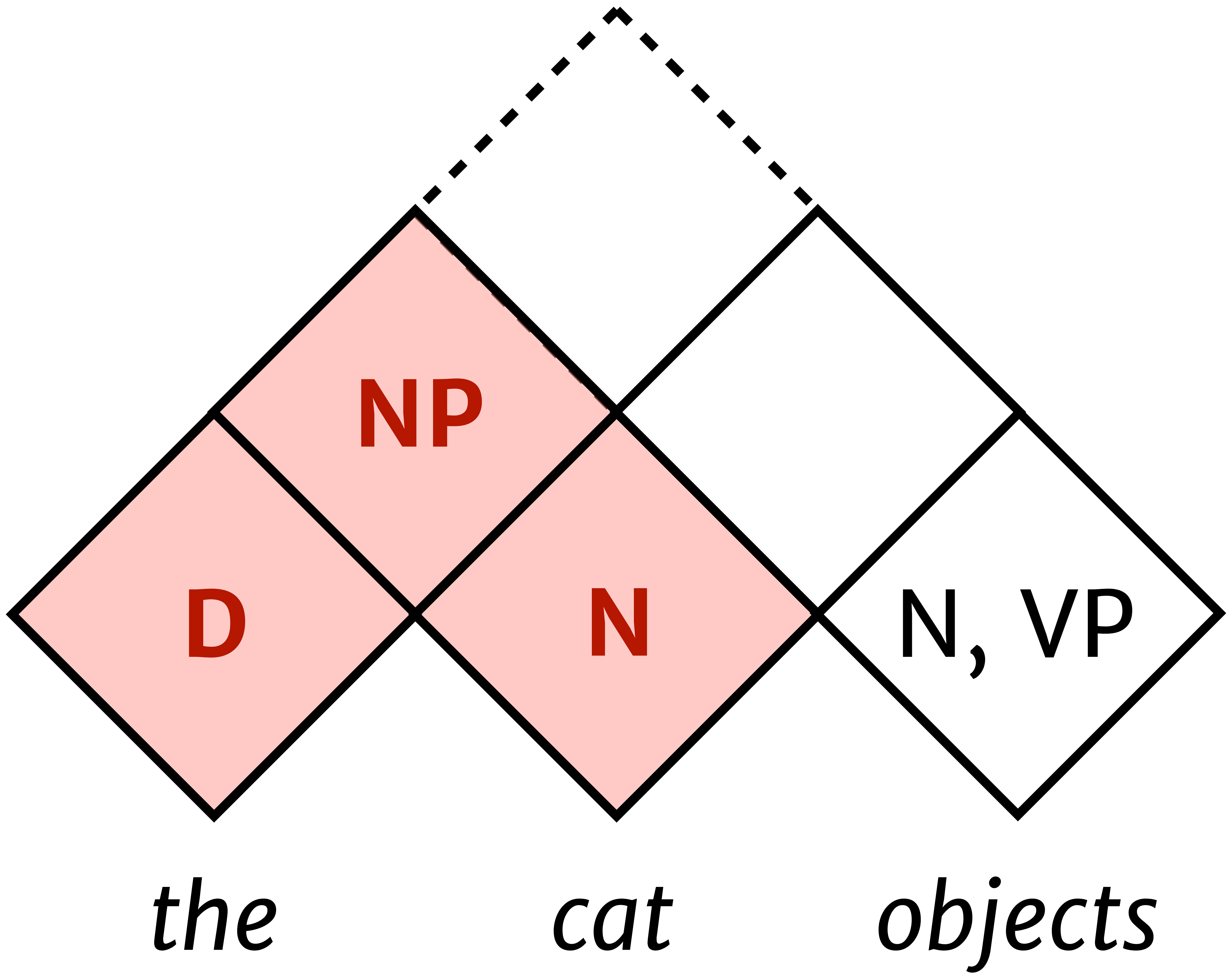
1. Fill in bottom row with NTs that can generate observed words



S	→	NP VP
NP	→	D N
N	→	cat | objects
VP	**→**	**objects | sings**
D	→	the | a

Warmup: the CKY algorithm

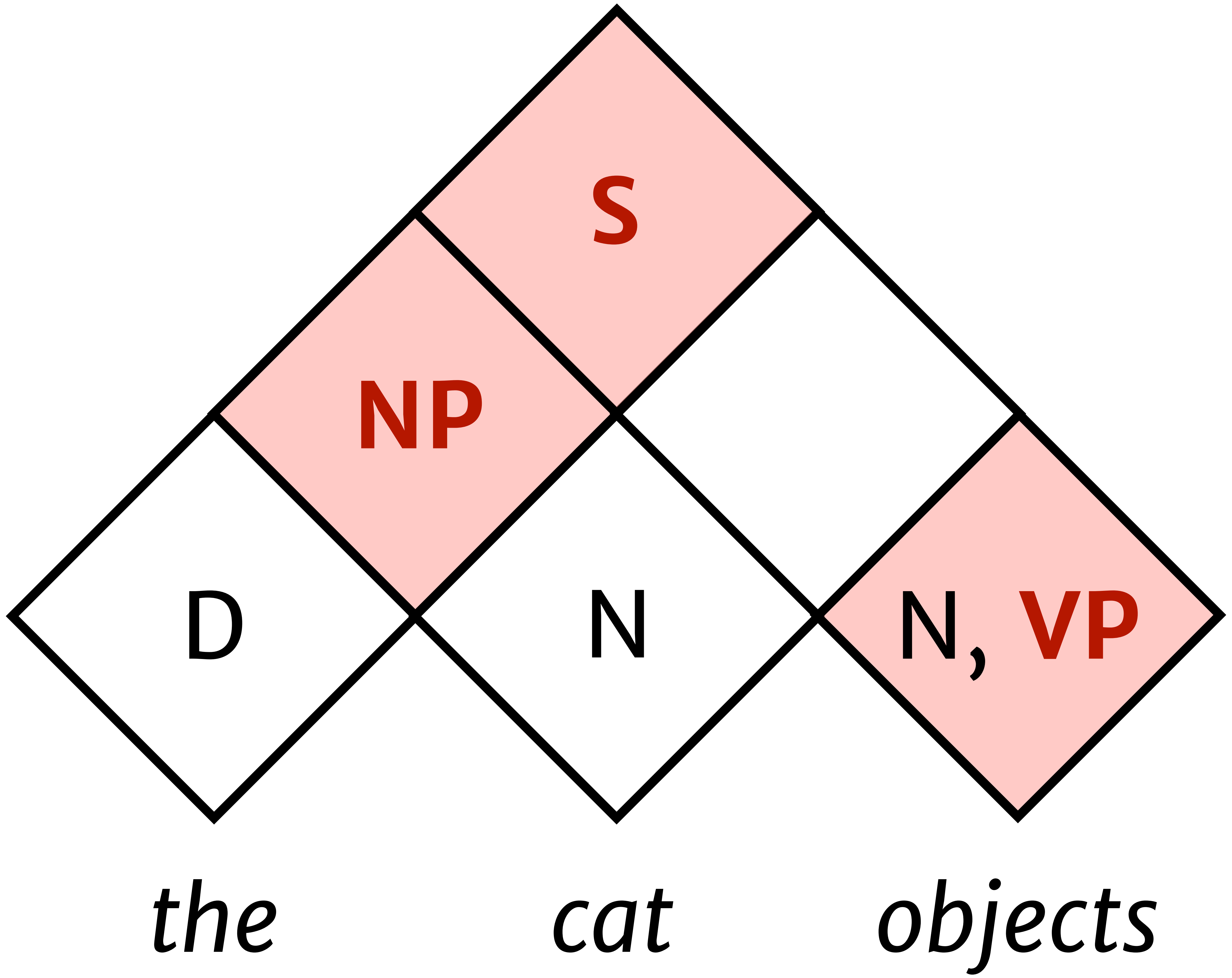
2. Fill in second row with NTs that generate a symbol in **each** child



S → NP VP
NP → D N
N → cat | objects
VP → objects | sings
D → the | a

Warmup: the CKY algorithm

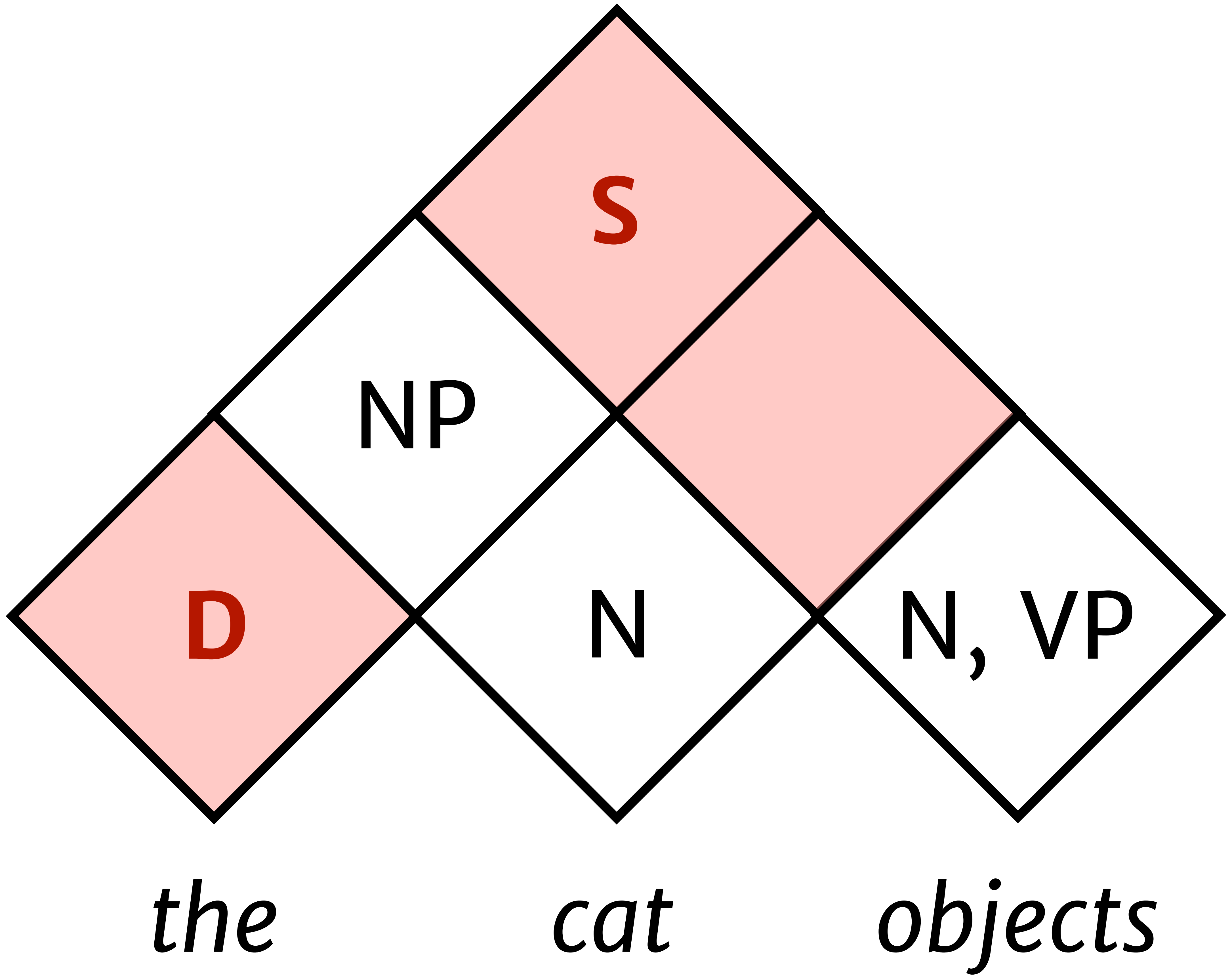
3. Fill in higher rows with NTs that generate a symbol **any pair of non-overlapping** children



S → NP VP
NP → D N
N → cat | objects
VP → objects | sings
D → the | a

Warmup: the CKY algorithm

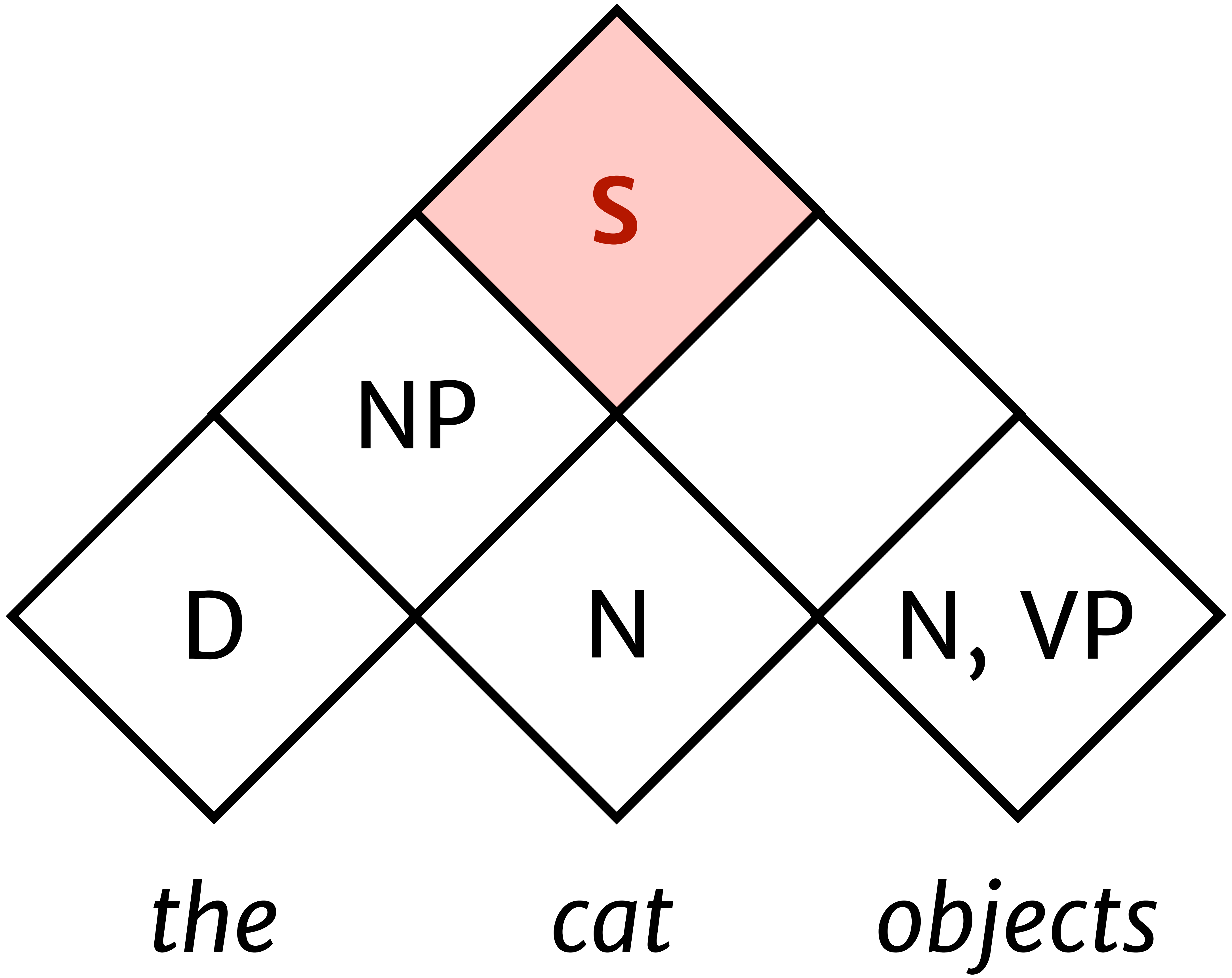
3. Fill in higher cells with NTs that generate symbols in **any pair of non-overlapping** children



S	→	**NP VP**
NP	→	D N
N	→	cat | objects
VP	→	objects | sings
D	→	the | a

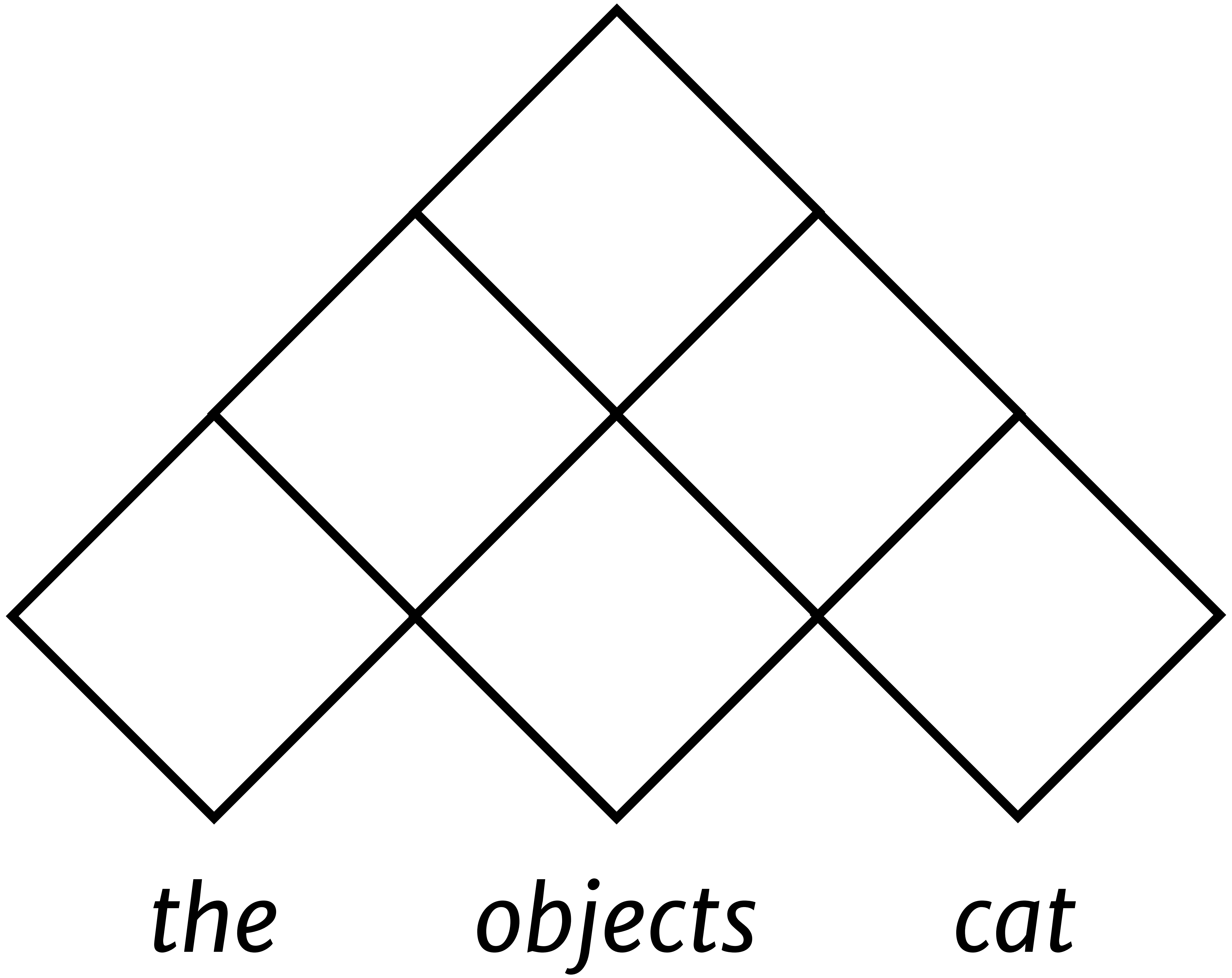
Warmup: the CKY algorithm

4. If the top cell contains the start symbol, the string is generated.



S → NP VP
NP → D N
N → cat | objects
VP → objects | sings
D → the | a

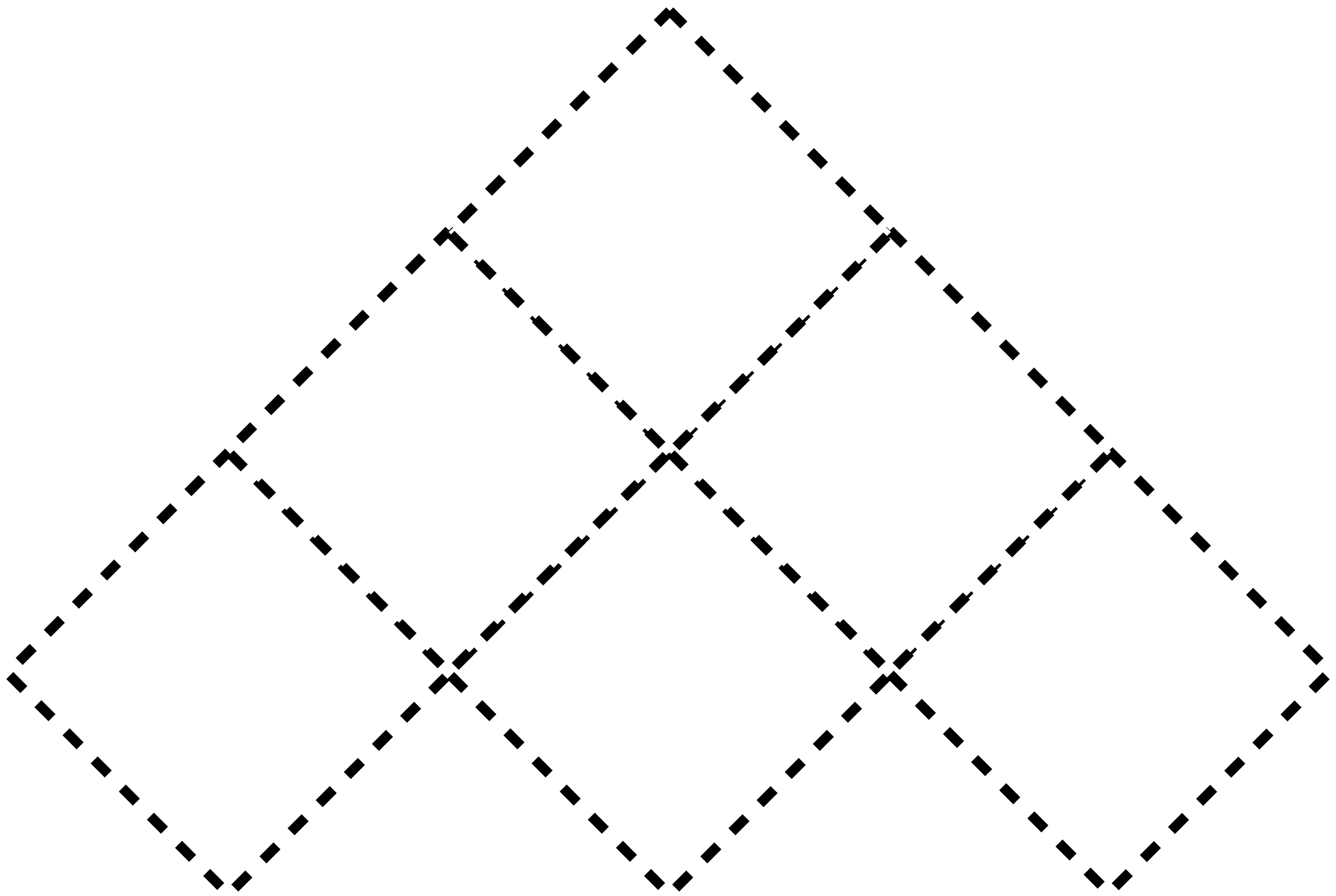
Warmup: the CKY algorithm



S	→	NP VP
NP	→	D N
N	→	cat | objects
VP	→	objects | sings
D	→	the | a

Highest-scoring parse

What parse assigns highest prob. to S under the PCFG G?

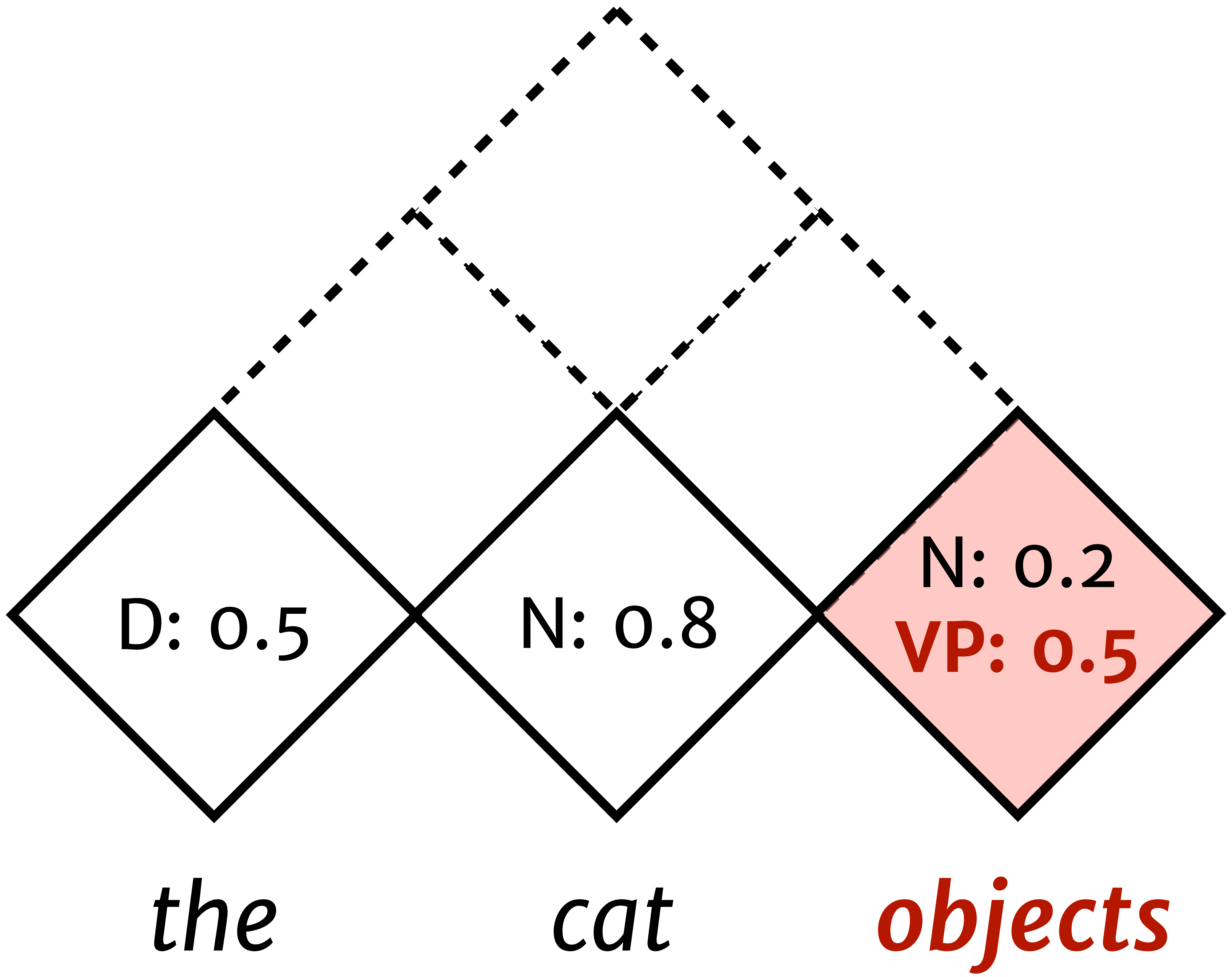


the *cat* *objects*

S	→	NP VP (0.9) | NP N (0.1)
NP	→	D N (1)
N	→	cat (0.8) | objects (0.2)
VP	→	objects (0.5) | sings (0.5)
D	→	the (0.5) | a (0.5)

Highest-scoring parse

1. Fill in bottom row with prob. that each NT generates word



$$S \rightarrow \overset{0.9}{NP\ VP} \mid \overset{0.1}{NP\ N}$$

$$NP \rightarrow \overset{1}{D\ N}$$

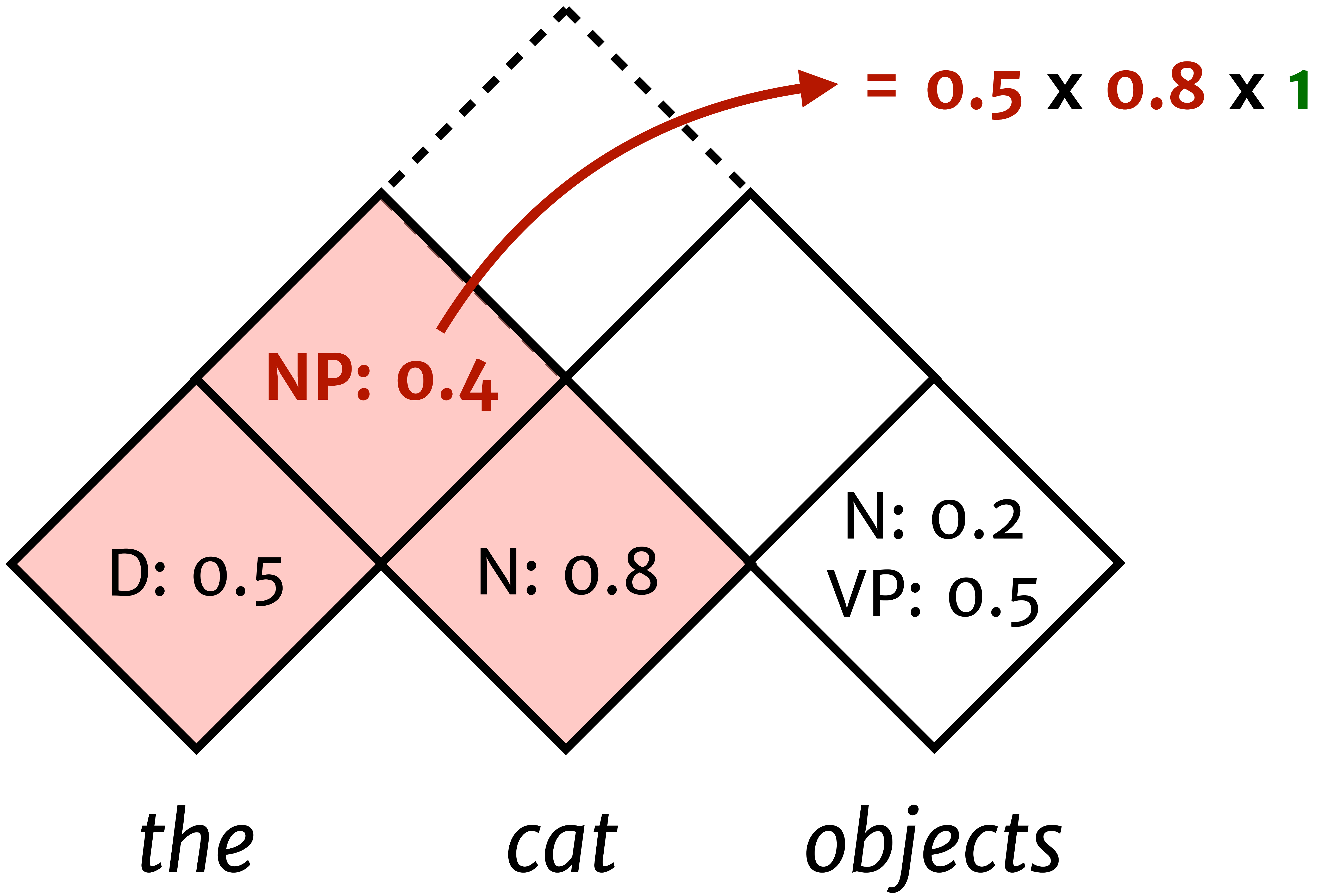
$$N \rightarrow \overset{0.8}{\text{cat}} \mid \overset{0.2}{\text{objects}}$$

$$\mathbf{VP \rightarrow \overset{0.5}{\textbf{objects}} \mid \overset{0.5}{\textbf{sings}}}$$

$$D \rightarrow \overset{0.5}{\text{the}} \mid \overset{0.5}{\text{a}}$$

Highest-scoring parse

2. Fill in higher rows with highest-scoring product of child probs. times rule prob.



0.9 0.1
S → NP VP | NP N

1
NP → D N

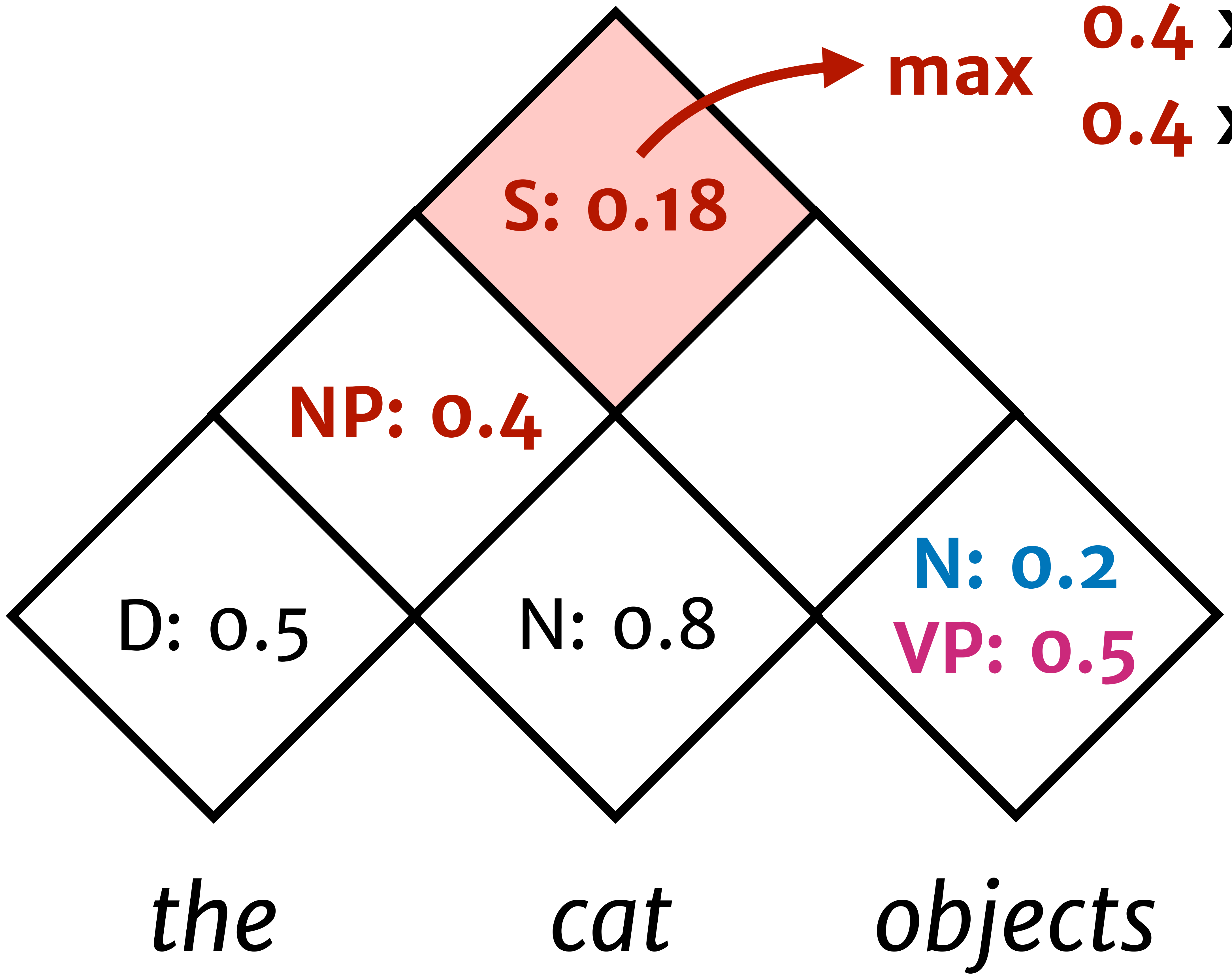
0.8 0.2
N → cat | objects

0.5 0.5
VP → objects | sings

0.5 0.5
D → the | a

Highest-scoring parse

2. Fill in higher rows with highest-scoring product of child probs. times rule prob.



max $0.4 \times 0.5 \times 0.9,$ $0.4 \times 0.2 \times 0.1$

$$S \rightarrow \overset{0.9}{NP\ VP} \mid \overset{0.1}{NP\ N}$$

$$NP \rightarrow \overset{1}{D\ N}$$

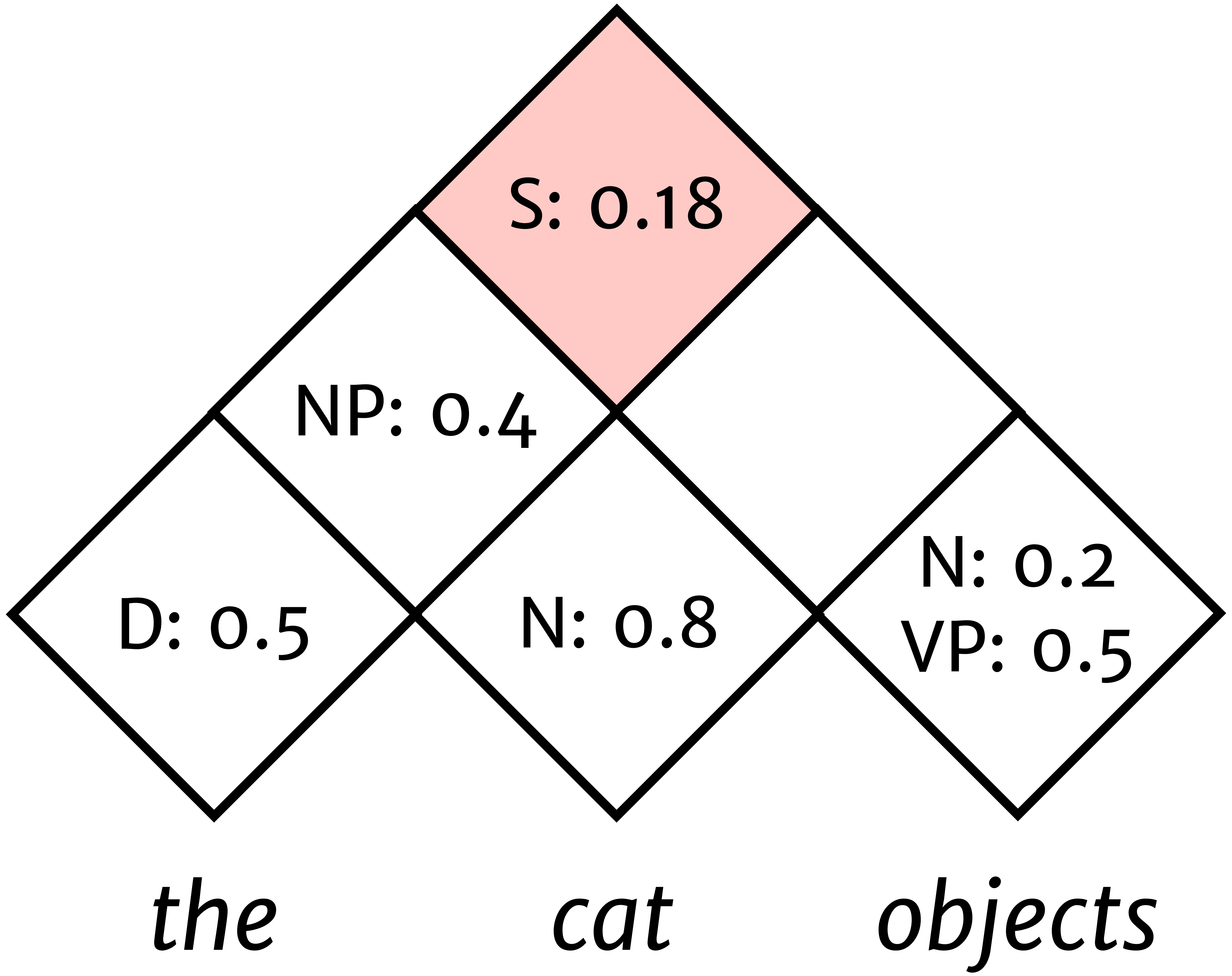
$$N \rightarrow \overset{0.8}{\text{cat}} \mid \overset{0.2}{\text{objects}}$$

$$VP \rightarrow \overset{0.5}{\text{objects}} \mid \overset{0.5}{\text{sings}}$$

$$D \rightarrow \overset{0.5}{\text{the}} \mid \overset{0.5}{\text{a}}$$

Highest-scoring parse

3. The score for S in the top cell is the score of the best parse.



$$S \rightarrow \overset{0.9}{\text{NP VP}} \mid \overset{0.1}{\text{NP N}}$$

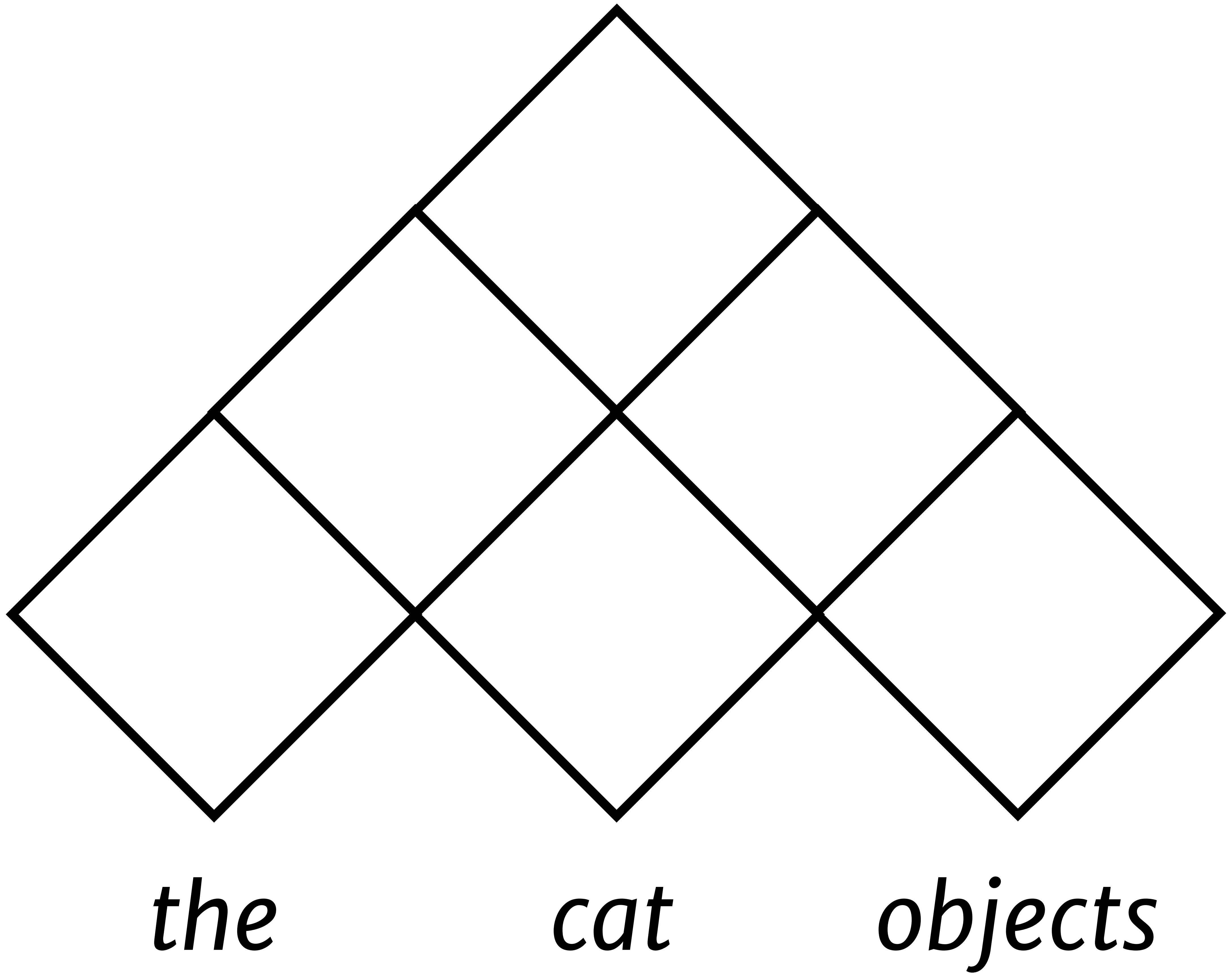
$$NP \rightarrow \overset{1}{\text{D N}}$$

$$N \rightarrow \overset{0.8}{\text{cat}} \mid \overset{0.2}{\text{objects}}$$

$$VP \rightarrow \overset{0.5}{\text{objects}} \mid \overset{0.5}{\text{sings}}$$

$$D \rightarrow \overset{0.5}{\text{the}} \mid \overset{0.5}{\text{a}}$$

Highest-scoring parse



$$S \rightarrow \overset{0.9}{NP\ VP} \mid \overset{0.1}{NP\ N}$$

$$NP \rightarrow \overset{1}{D\ N}$$

$$N \rightarrow \overset{0.8}{cat} \mid \overset{0.2}{objects}$$

$$VP \rightarrow \overset{0.5}{objects} \mid \overset{0.5}{sings}$$

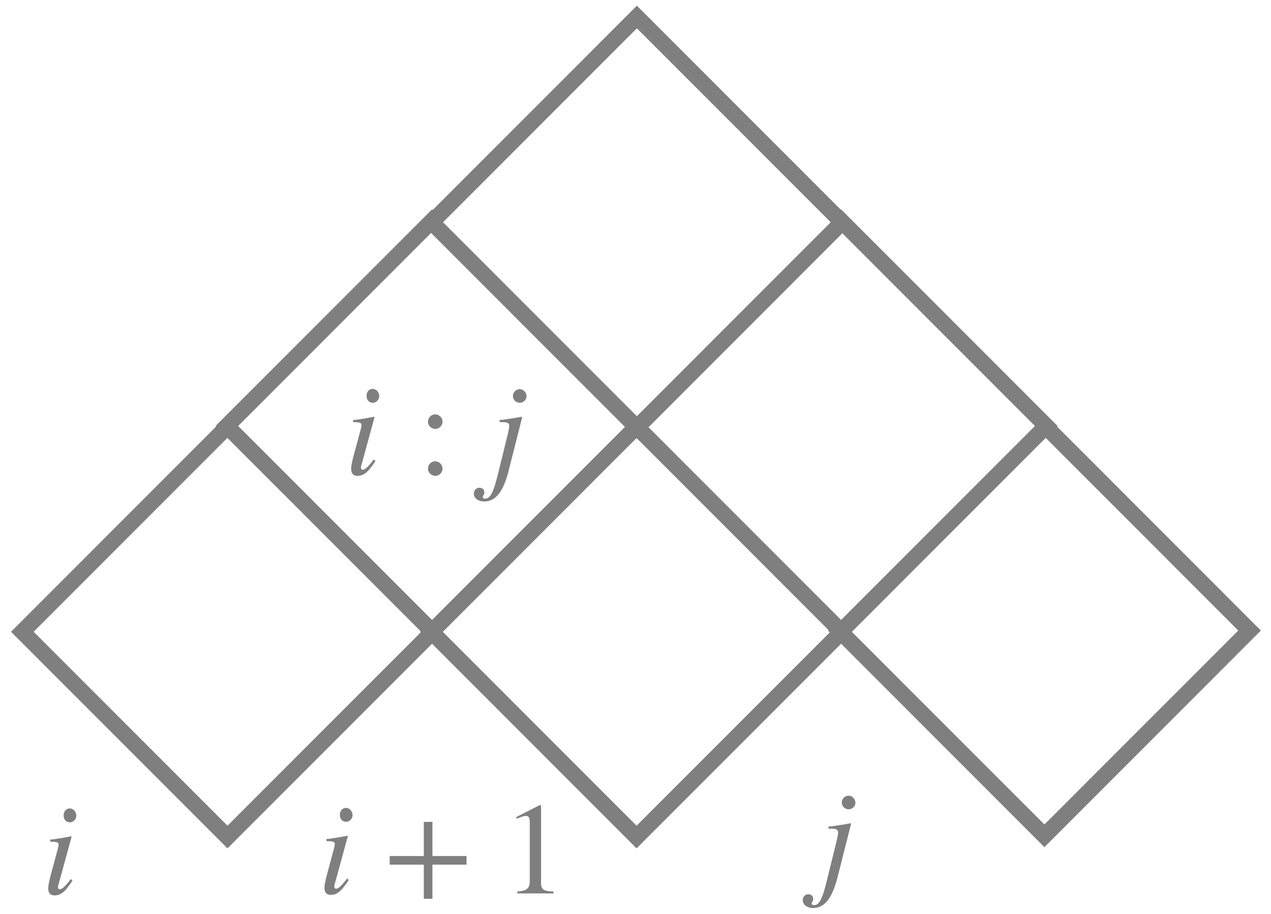
$$D \rightarrow \overset{0.5}{the} \mid \overset{0.5}{a}$$

The Viterbi algorithm for CFGs

Q: what is the **most probable** tree for a given sentence?

$$\max_T \; p(T, S)$$

$\delta(s, i, j)$ highest-scoring tree with root s covering words i:j



base case:

$$\delta(s, i, i+1) = p(s \to w_i)$$

inductive case:

$$\delta(s, i, j) = \max_{k \in [i+1, j-1]} \; \max_{s', s''} p(s \to s' s'') \; \delta(s', i, k) \; \delta(s'', k, j)$$

The Viterbi algorithm for CRFs

Q: what is the **most probable** assignment of tags to observations?

$$\text{argmax}_Q \; p(Q \mid O)$$

$$\boxed{\delta(t,j) = \max_i \delta(t-1,i)\; a_{ij}\; b_j(o_t) \qquad \delta(1,j) = \pi(j)\; b_j(o_1)}$$

The inside algorithm for CFGs

Q: what is the marginal probability of a sentence given a tree?

$$\sum_T p(T, S)$$

$\beta(s, i, j)$ probability of all parses with root s covering words i:j

base case:

$$\beta(s, i, i+1) = p(s \rightarrow w_i)$$

inductive case:

$$\beta(s, i, j) = \sum_{k \in [i+1, j-1]} \sum_{s', s''} p(s \rightarrow s' s'') \; \beta(s', i, k) \; \beta(s'', k, j)$$

Tree-structured CRFs

Instead of scores $p(A \to B\ C)$, use an arbitrary scorer

$$w^\top \phi(A, B, C, i, j)$$

(can be different in each cell & look at full sentence sentence)

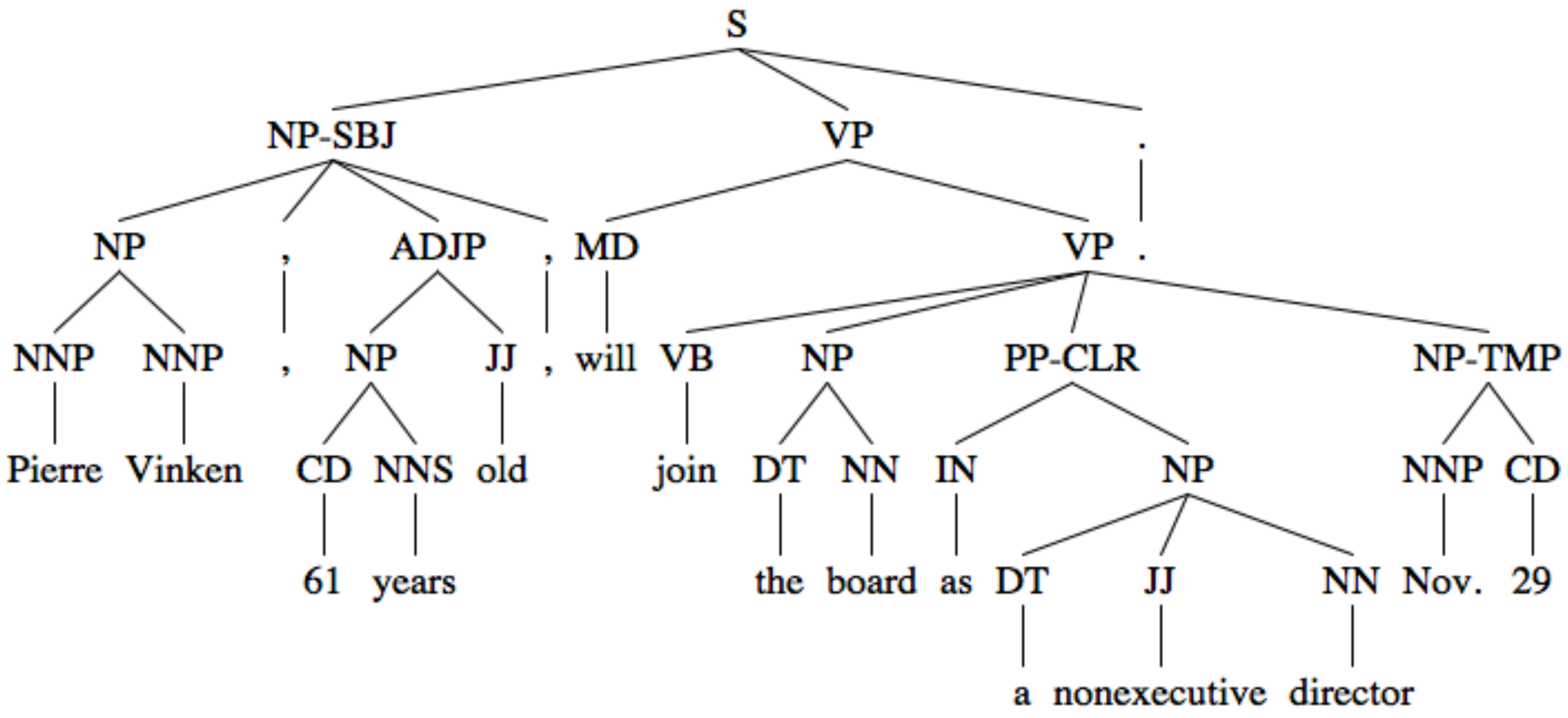
Works just like the HMM version!

$\beta(\mathsf{S}, 0, |S|)$ is the partition function

Learning

Supervised learning

For PCFGs—given a **treebank,** estimate by counting:

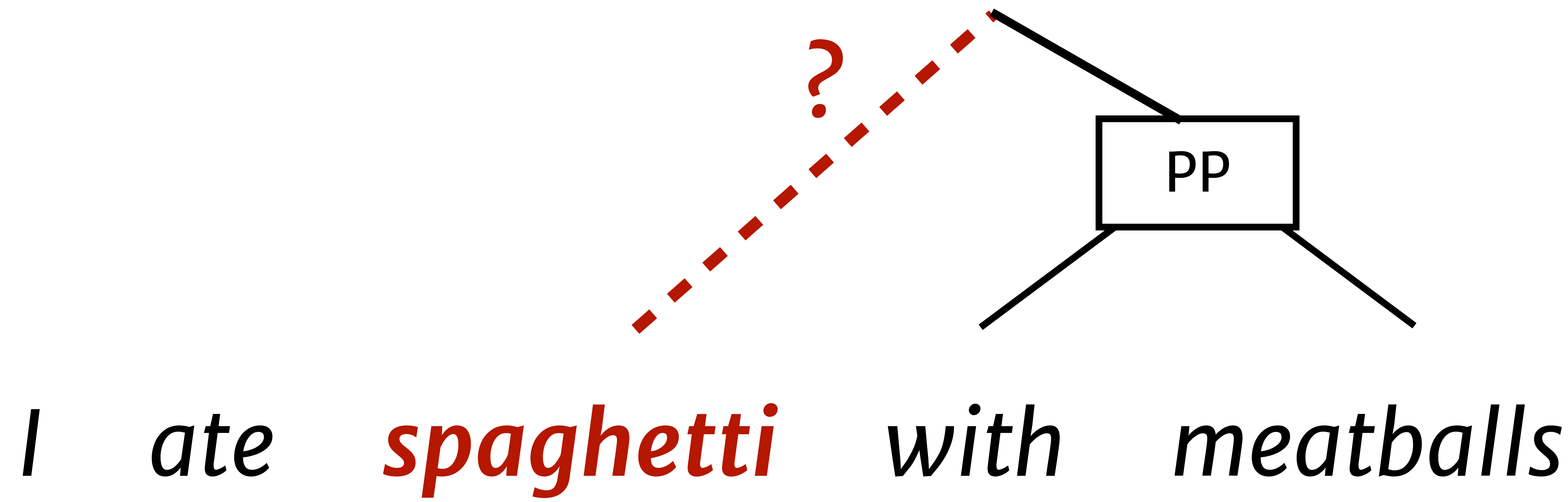


$$p(\mathrm{S} \to \mathrm{NP\ VP}) = \frac{\#(\mathrm{S} \to \mathrm{NP\ VP})}{\#(\mathrm{S})}$$

Supervised learning

For PCFGs—given a **treebank**, estimate by counting:

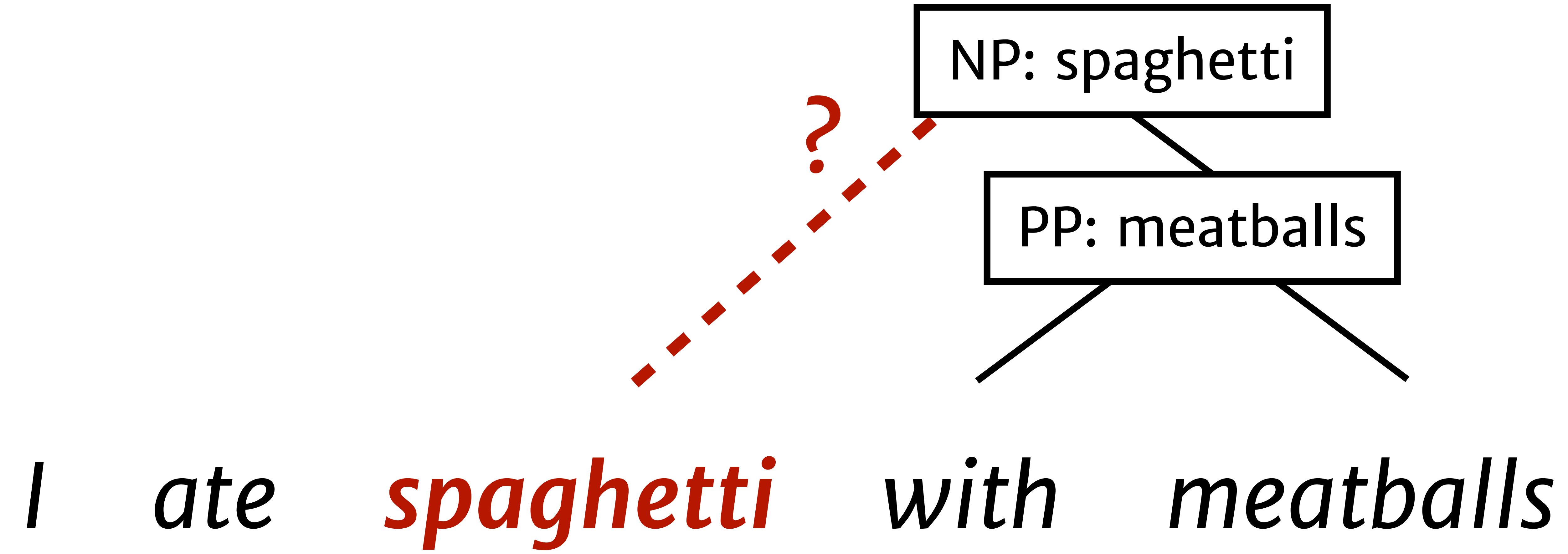
$$p(\mathsf{S} \to \mathsf{NP\ VP}) = \frac{\#(\mathsf{S} \to \mathsf{NP\ VP})}{\#(\mathsf{S})}$$



This doesn't work very well: basic syntactic categories are too coarse.

Supervised learning: lexicalization

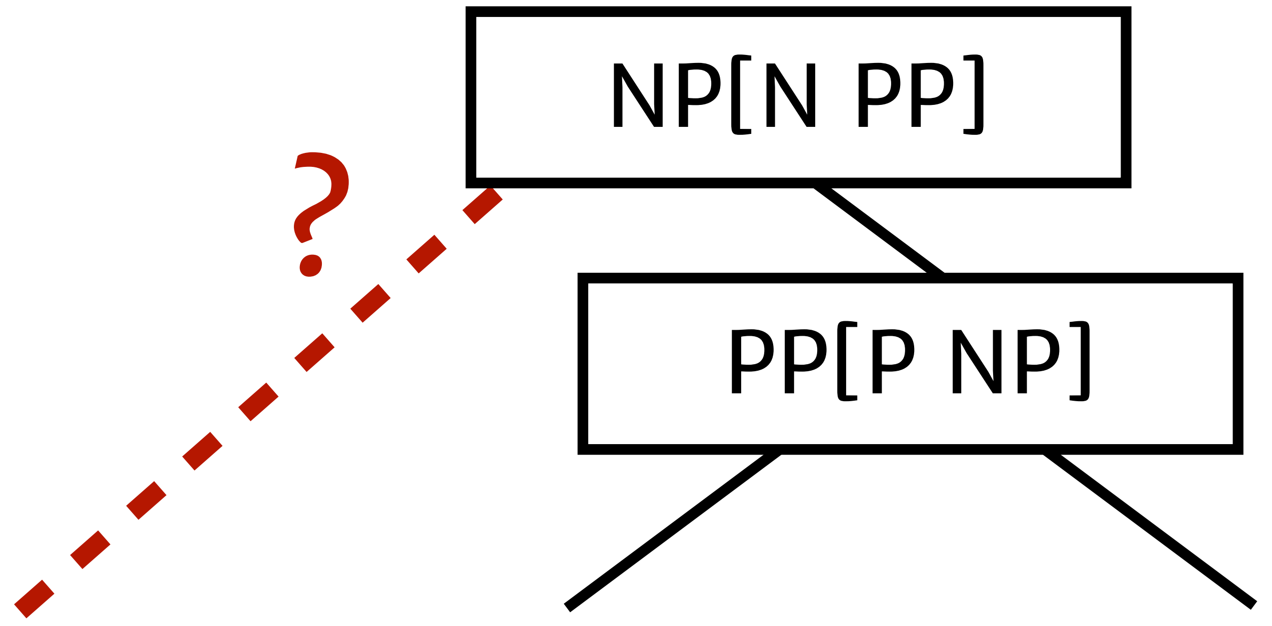
Idea: enrich nonterminal alphabet with information about the most important **word** underneath:



[e.g. Collins 97]

Supervised learning: Markovization

Idea: enrich nonterminal alphabet with more information about the local tree structure:



I ate ***spaghetti*** *with meatballs*

[e.g. Klein 03]

Supervised learning: features & NNs

Idea: Use the CRF version

$$P(T) \propto \exp\left\{ \sum_{(A \to B\ C, i, k, j)} w^\top \phi(A, B, C, i, k, j) \right\}$$

and give ϕ features like "A = NP and *j:k* contains *fork*"
(or make it a neural network)

Supervised parsing: what's still hard?

Error Type	Occurrences	Nodes Involved	Ratio
PP Attachment	846	1455	1.7
Single Word Phrase	490	490	1.0
Clause Attachment	385	913	2.4
Adverb and Adjective Modifier Attachment	383	599	1.6
Different Label	377	754	2.0
Unary	347	349	1.0
NP Attachment	321	597	1.9
NP Internal Structure	299	352	1.2
Coordination	209	557	2.7
Unary Clause Label	185	200	1.1
VP Attachment	64	159	2.5
Parenthetical Attachment	31	74	2.4
Missing Parenthetical	12	17	1.4
Unclassified	655	734	1.1

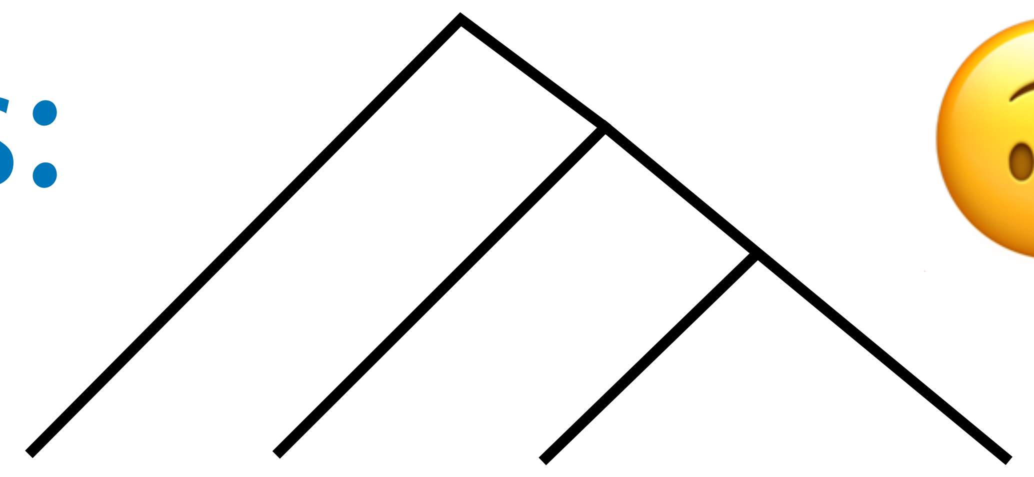
→ *spaghetti with a fork*

→ *[[world oil] prices]*

[Kummerfeld, 2016]

Unsupervised learning

Model	F_1	Training/Test PPL
Random Trees	19.5	–
Right Branching	39.5	–
Scalar PCFG (unsupervised)	< 35.0	> 350

worse than assuming every tree looks like this:  🙃

[Kim et al. 2018]

Unsupervised learning: embeddings

Model	F_1	Training/Test PPL
Random Trees	19.5	–
Right Branching	39.5	–
Scalar PCFG	< 35.0	> 350
Neural PCFG	52.6	≈ 250

"Grammar embeddings": $p(A \rightarrow B\ C) \propto \exp\{v_A^\top f(v_B, v_C)\}$

[Kim et al. 2018]

Next class: NNs and trees