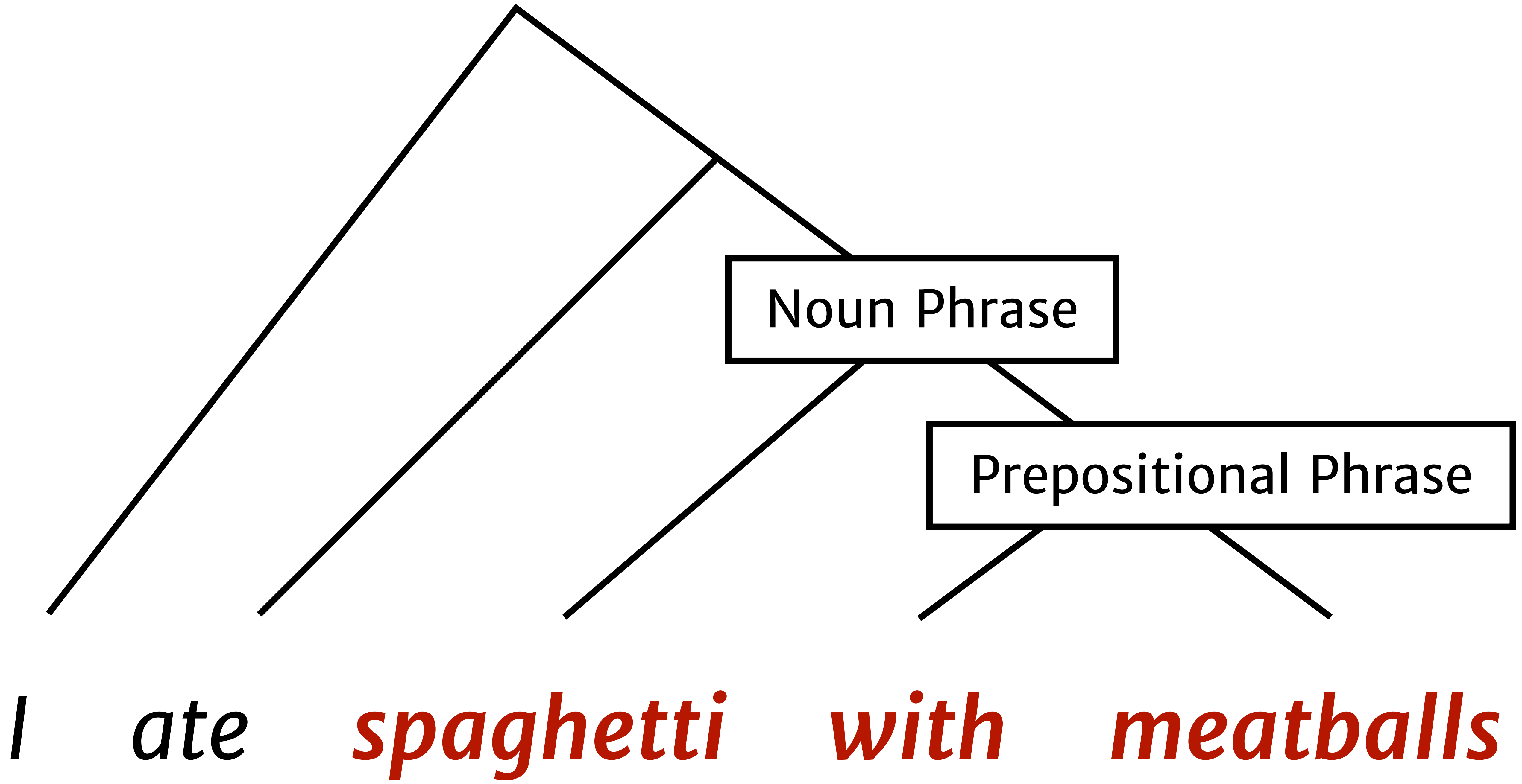
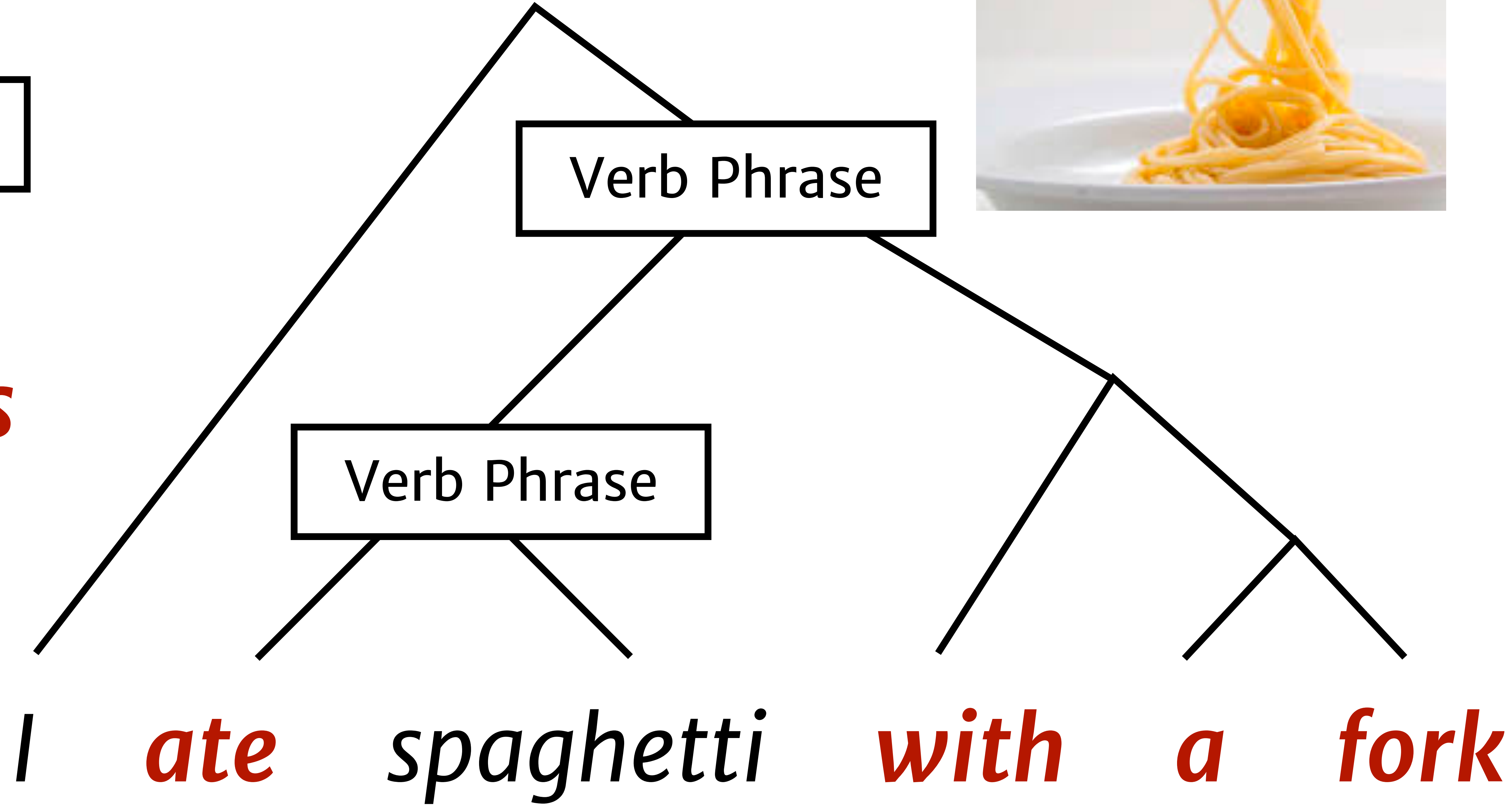
Formal Semantics

Jacob Andreas / MIT 6.804-6.864 / Spring 2020

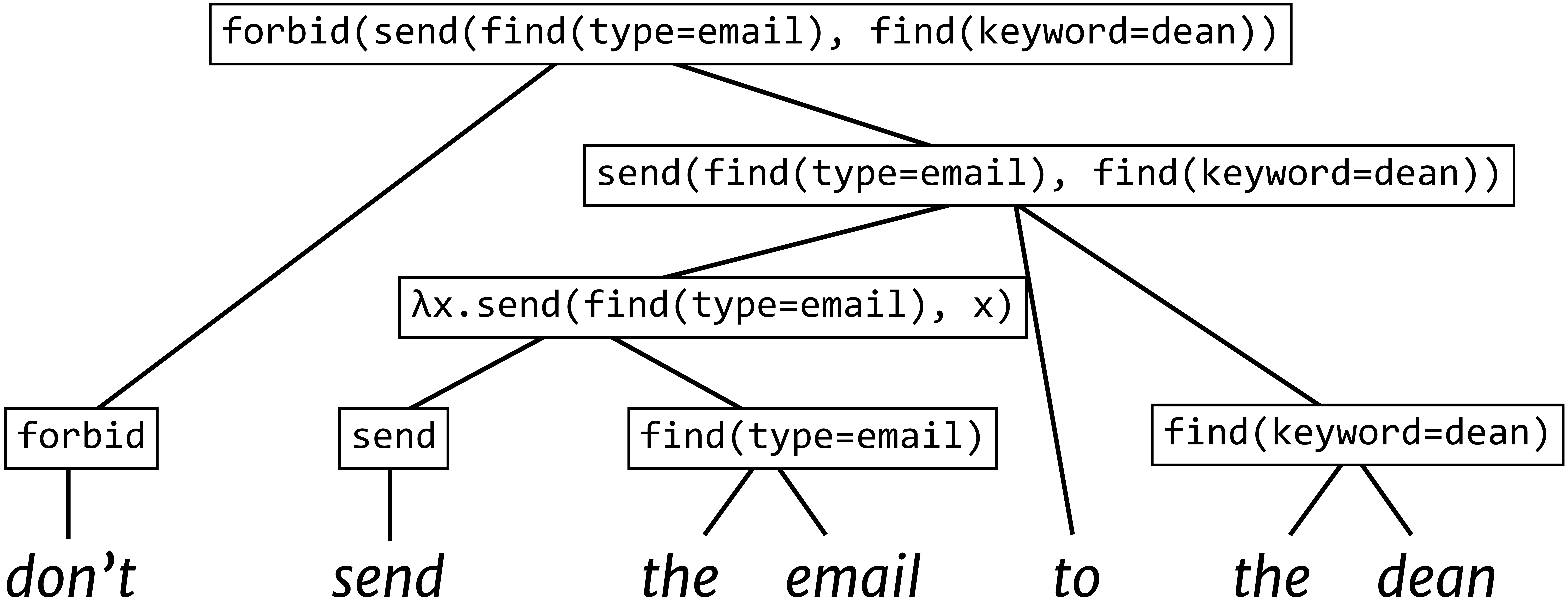
Recap: trees

Syntax


Noun Phrase
Prepositional Phrase
I ate spaghetti with meatballs

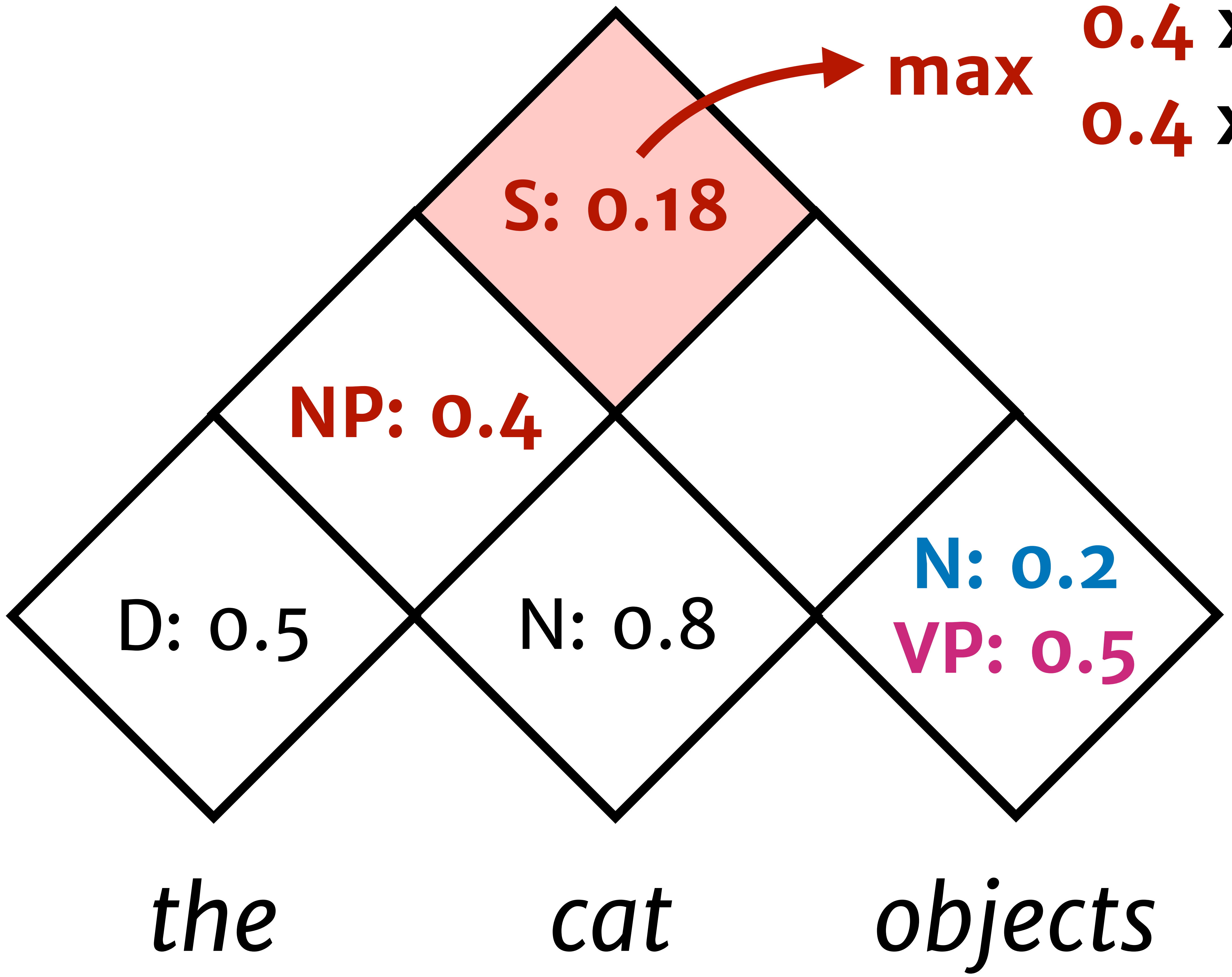

Verb Phrase
Verb Phrase
I ate spaghetti with a fork

Types & semantics



Highest-scoring parse

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



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

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

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

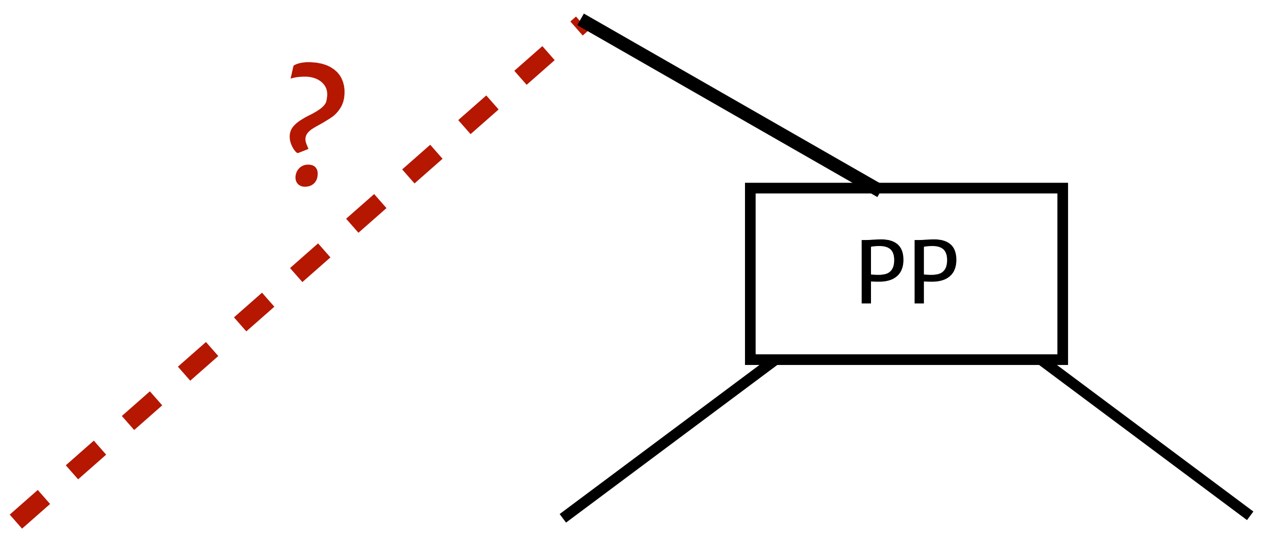
$$N \rightarrow \overset{0.8}{\text{cat}} \mid \overset{0.2}{\text{objects}}$$

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

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

Supervised learning

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



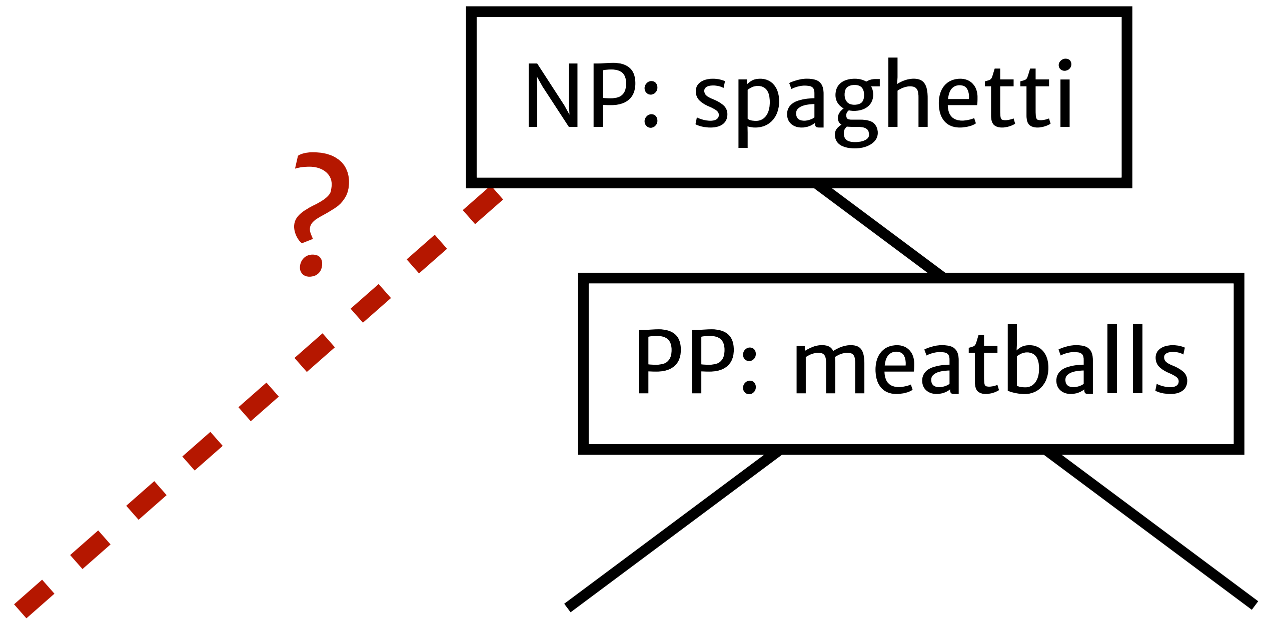
I ate ***spaghetti*** *with meatballs*

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

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:

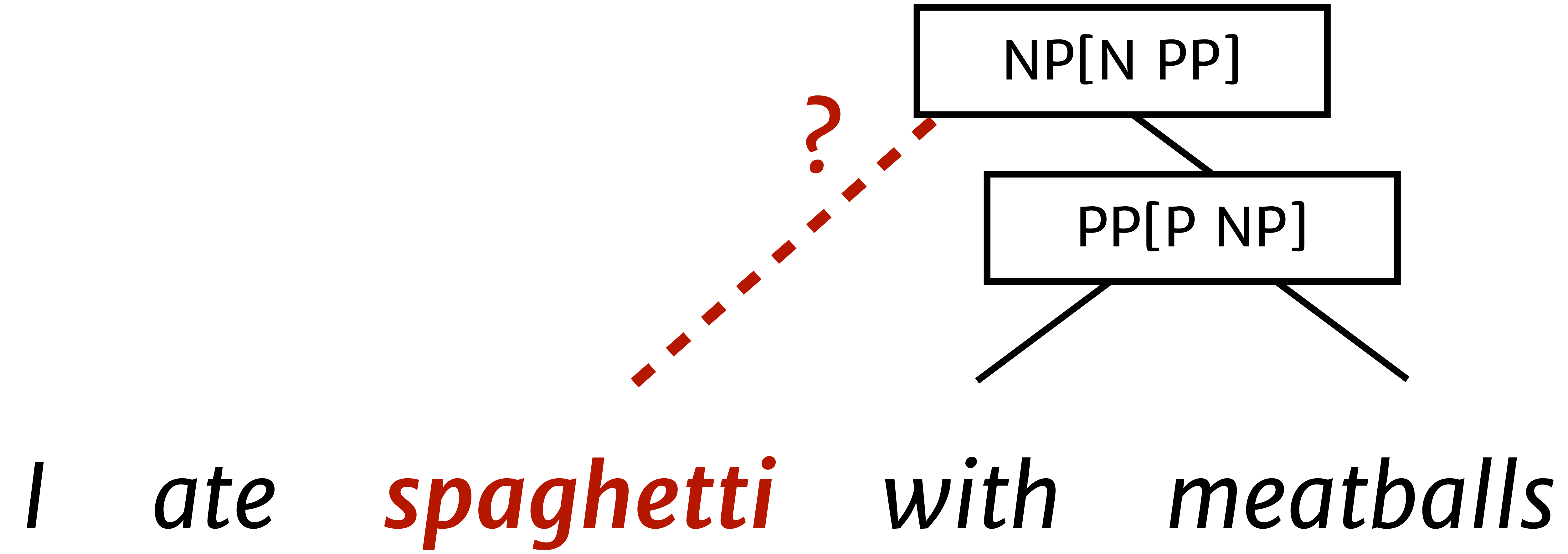


I ate spaghetti with meatballs

[e.g. Collins 97]

Supervised learning: Markovization

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



[e.g. Klein 03]

Supervised learning: features & NNs

Idea: Use the CRF version

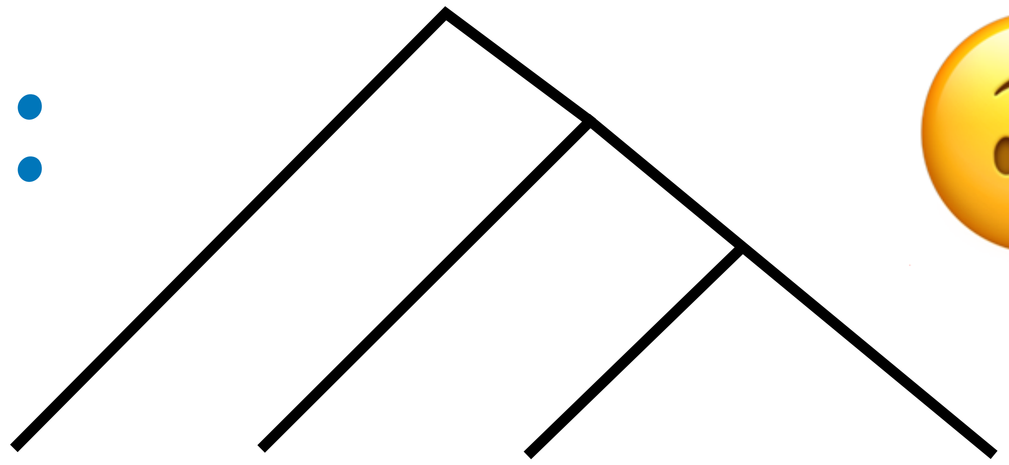
$$P(T) \propto \exp\left\{ \sum_{(A \to B\ C, i, k, j)} w^\top \phi(A, B, C, i, k, j) \right\}$$

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

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

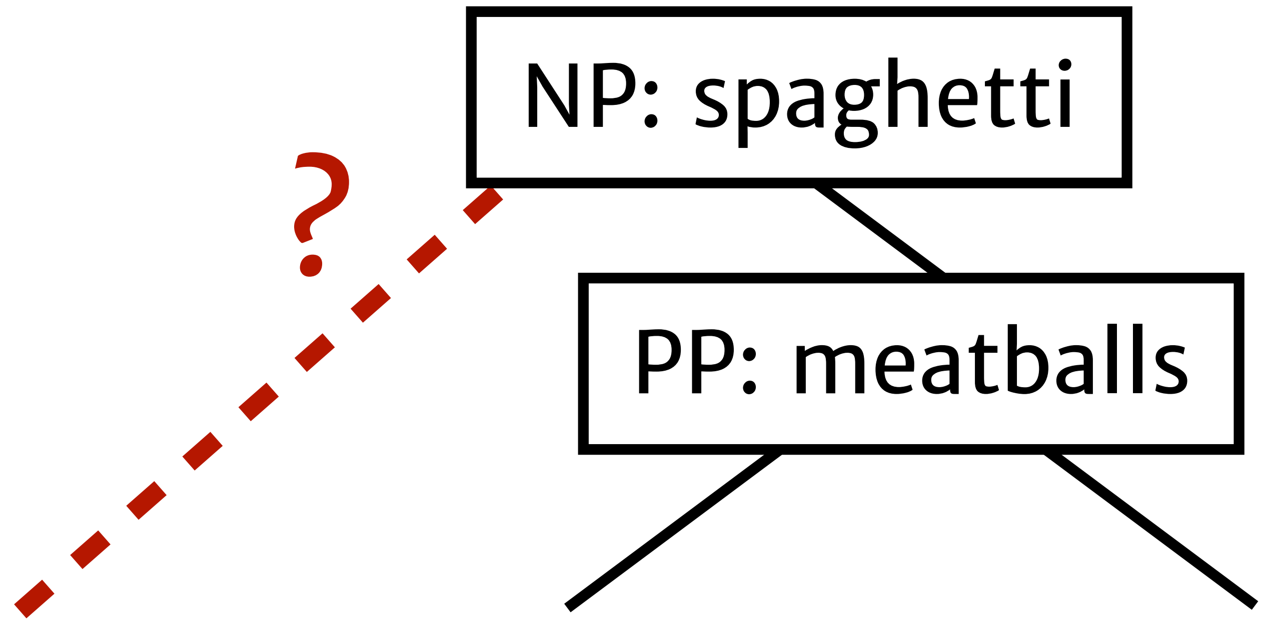
"Grammar embeddings": $p(A \rightarrow B\ C) \propto \exp\{v_A^\top f(v_B, v_C)\}$

[Kim et al. 2018]

Better learning for CFGs

Supervised learning: lexicalization

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



I ate spaghetti with meatballs

[e.g. Collins 97]

Problem 1

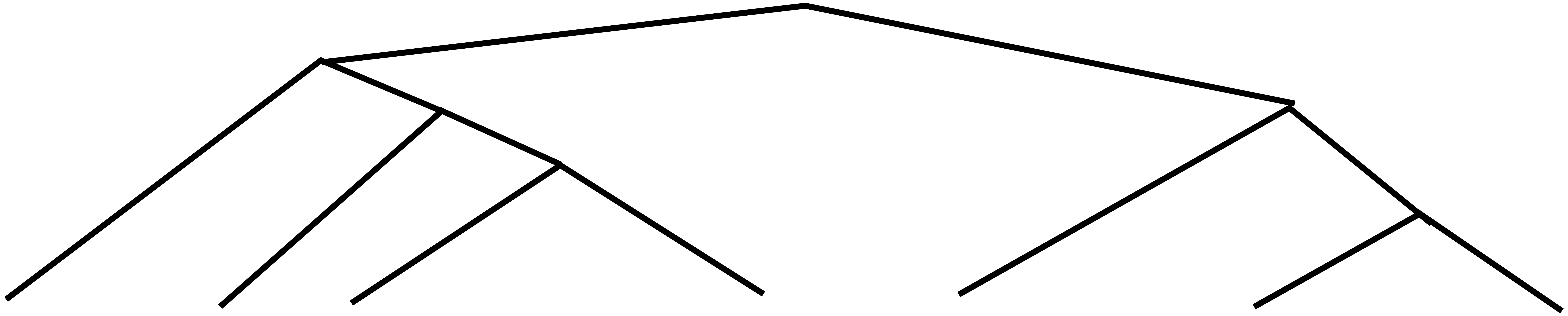
Each of the three girls has a platypus.

Each of the three girls climbed the mountain.

How many platypuses?

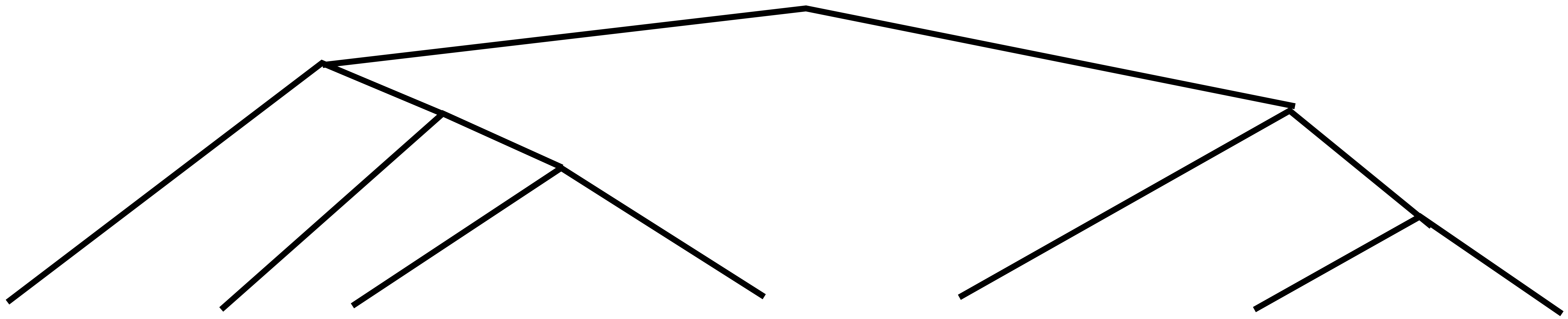
How many mountains?

Problem 1



Each of the three girls has a platypus.

Problem 1



Each of the three girls climbed the mountain.

Problem 2

There are 128 cities in South Carolina.

name	type	coastal
Columbia	city	no
Cooper	river	yes
Charleston	city	yes

Problem 3

Barack Obama was the 44th President of the United States. Obama was born on August 4 in Honolulu, Hawaii. In late August 1961, Obama's mother moved with him to the University of Washington in Seattle for a year…

Is Barack Obama from the United States?

Compositional semantics

It's not enough to have structured representations of syntax:
We also need structured representations of **meaning**.

Compositional semantics

It's not enough to have structured representations of syntax:
We also need structured representations of **meaning**.

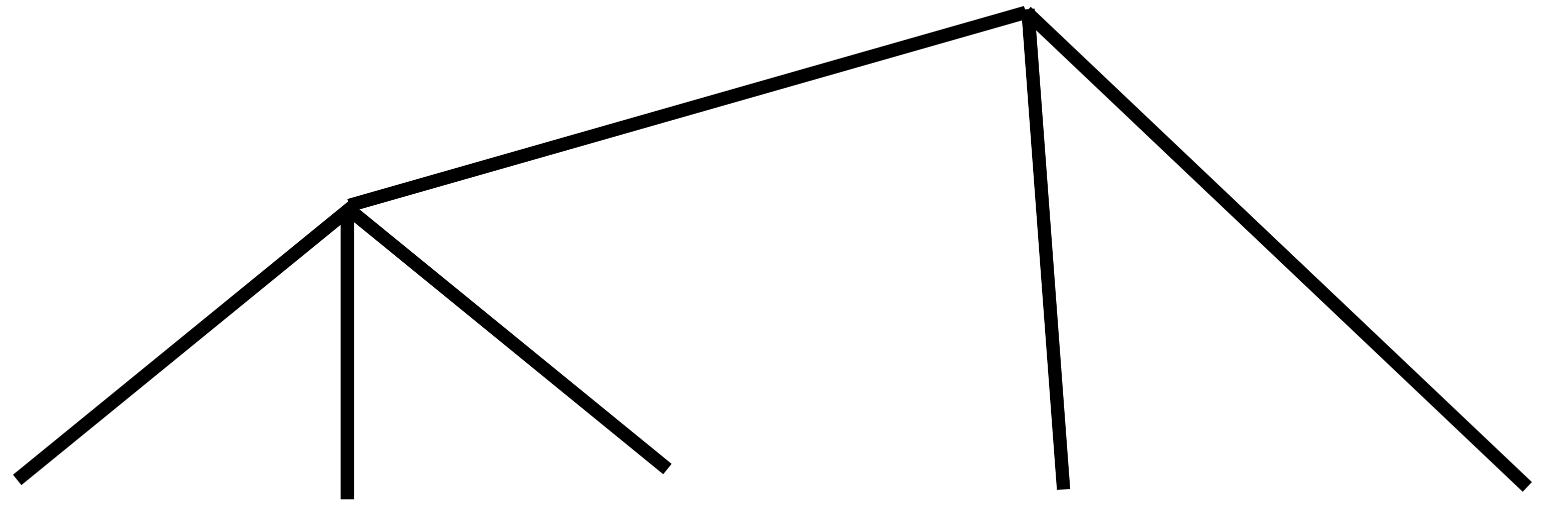
Today:
How do we get from **language** to **meaning?**

Representing meaning

Meaning in formal languages

$$a + b = 17$$

Meaning in formal languages



$$a + b = 17$$

Meaning in formal languages

$$a + b = 17$$

$$a = ?$$

$$b = ?$$

Meanings are sets of valid assignments

$$a + b = 17$$

{a=0, b=0} {a=17, b=0}

{a=3, b=10} {a=10, b=7}

{a=5, b=12} {a=5, b=5}

Meanings are sets of valid assignments

$$a + b = 17$$

$\{a=0, b=0\}$ ✘

$\{a=17, b=0\}$ ✔

$\{a=3, b=10\}$ ✘

$\{a=10, b=7\}$ ✔

$\{a=5, b=12\}$ ✔

$\{a=5, b=5\}$ ✘

Meanings are sets of valid assignments

$$a + 3 = 20 - b$$

{a=0, b=0} ✘	{a=17, b=0} ✔
{a=3, b=10} ✘	{a=10, b=7} ✔
{a=5, b=12} ✔	{a=5, b=5} ✘

Meanings are *functions* that judge validity

$$[\![a + b = 17]\!]$$

$\{a=5, b=12\}$ ✔

Meanings are *functions* that judge validity

$$[\![a + b = 17]\!]$$

$\{a{=}3,\ b{=}10\}$

✘

Lessons from math

$$[\![a + b = 17]\!]$$

The meaning of a statement is the **set** of possible worlds consistent with that statement.

Here, a "possible world" is an assignment of values to variables.

$$\{a=3,\ b=10\}$$

Meaning in natural languages

Pat likes Sal.

Representing possible worlds

Individuals	`Pat`	`Sal`
Properties	`whale ●—`	`sad ●—`
Relations	`—loves→`	`—contains→`

Example world

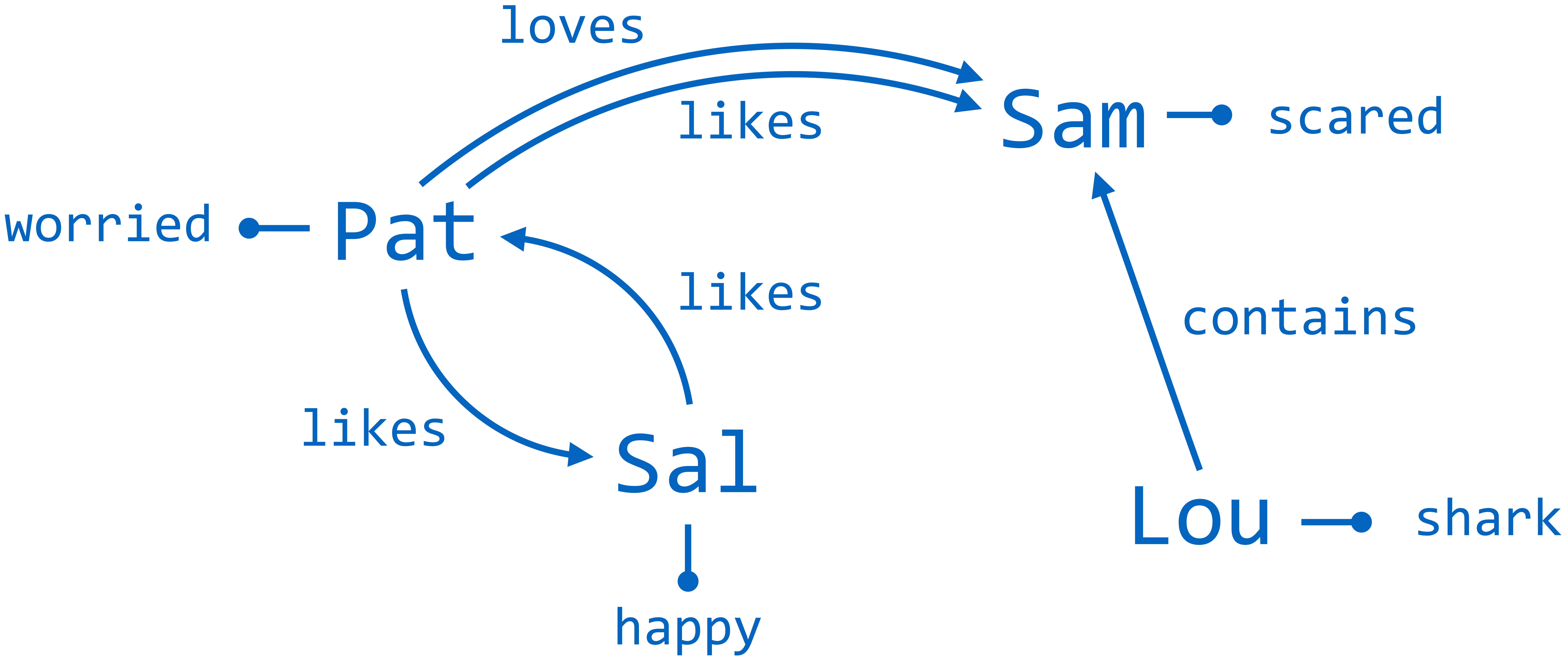
`Sam`

`Pat`

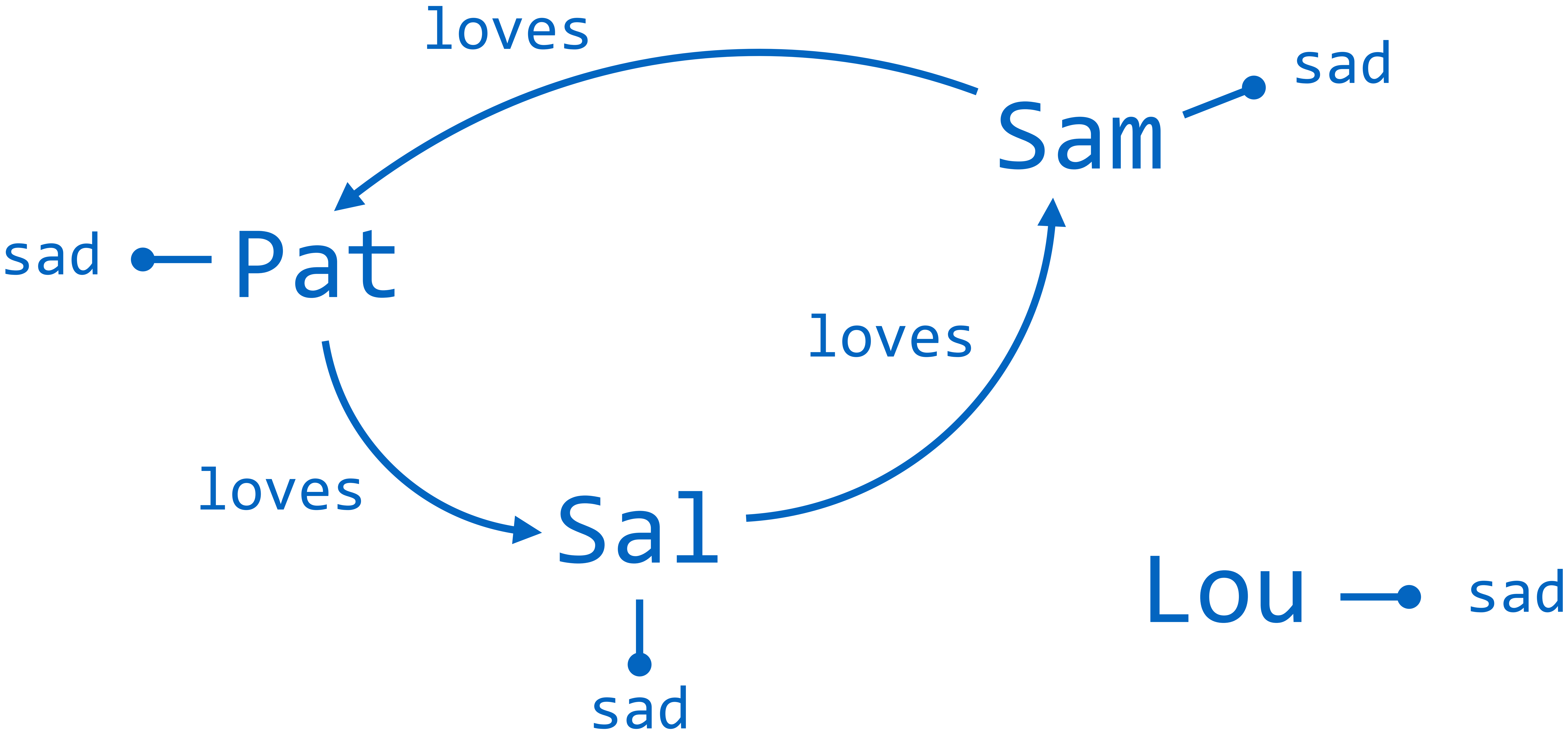
`Sal`

`Lou`

Example world



Different example world

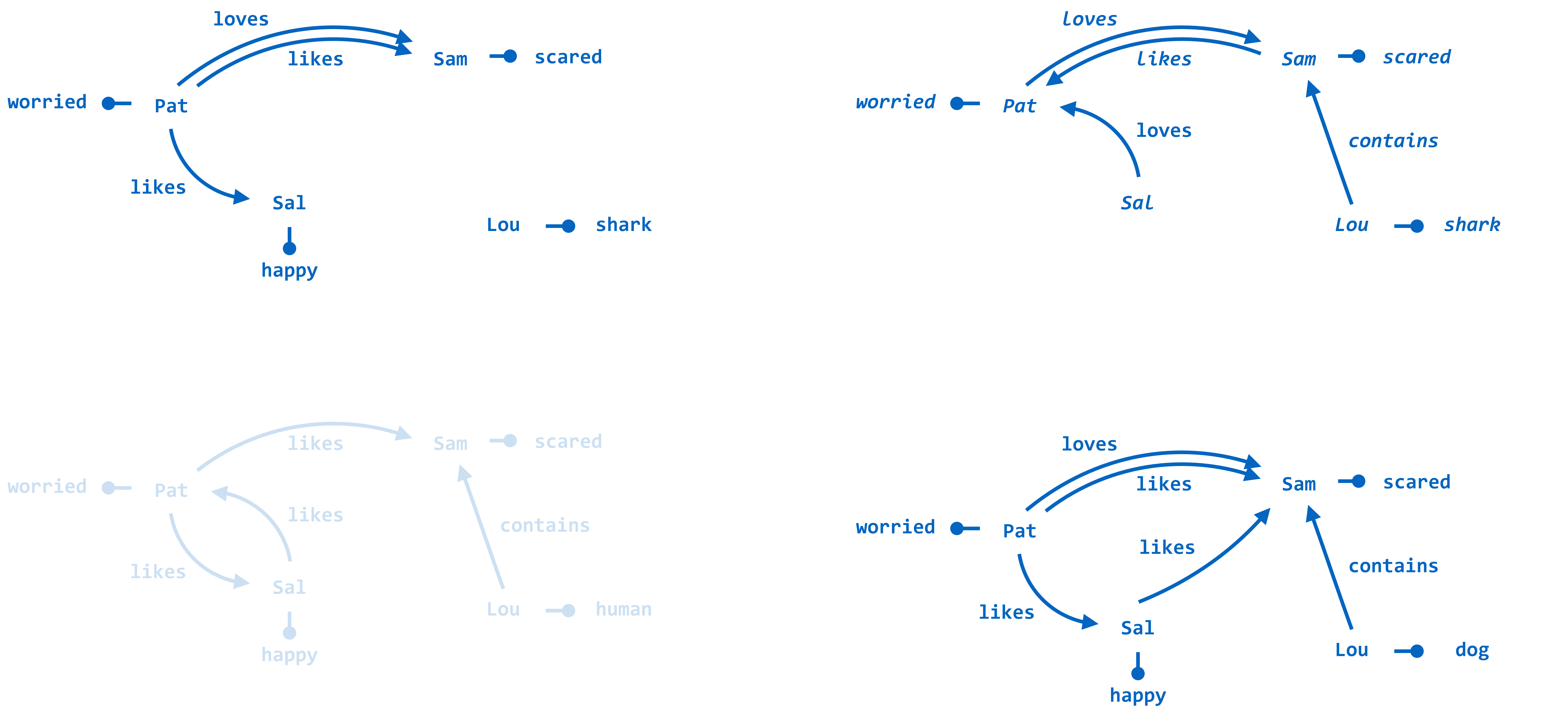


Representing possible worlds

Individuals	`Pat`	`Sal`
Properties	`whale={Lou}, sad={Pat,Sal}`	
Relations	`likes={(Pat,Sal),(Sal,Sam)}`	

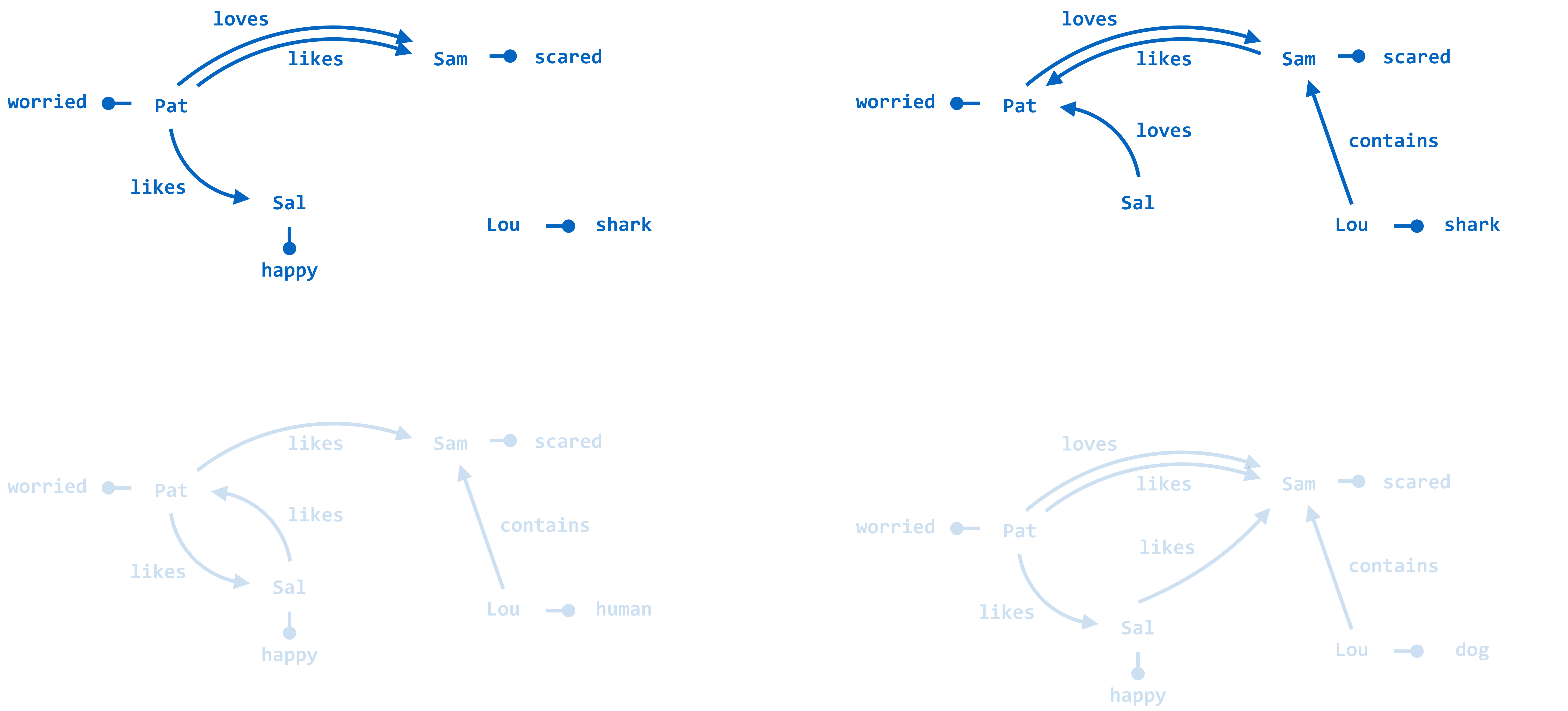
Interpretations of sentences

Pat likes Sal.



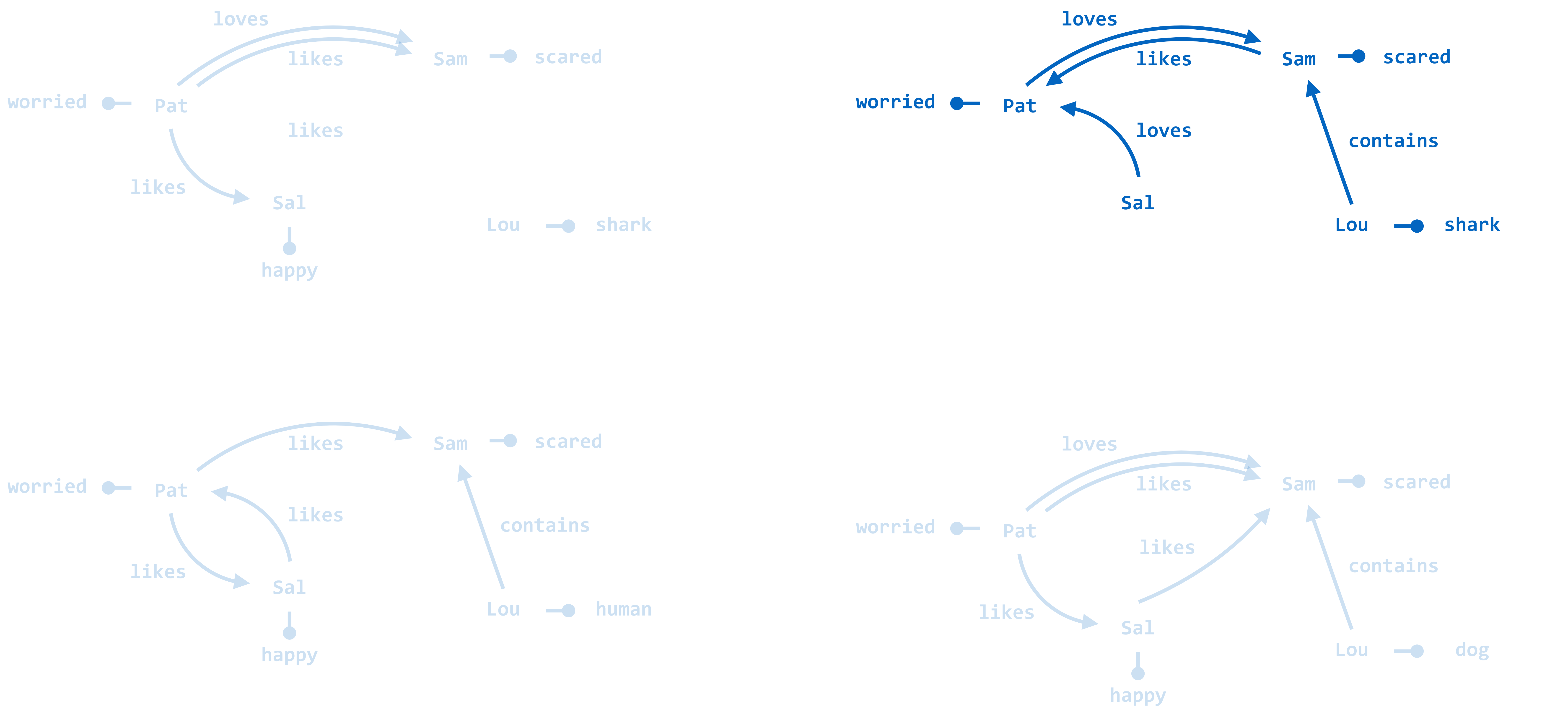
Interpretations of sentences

Lou is a shark.



Interpretations of sentences

Sam is inside Lou, a shark.



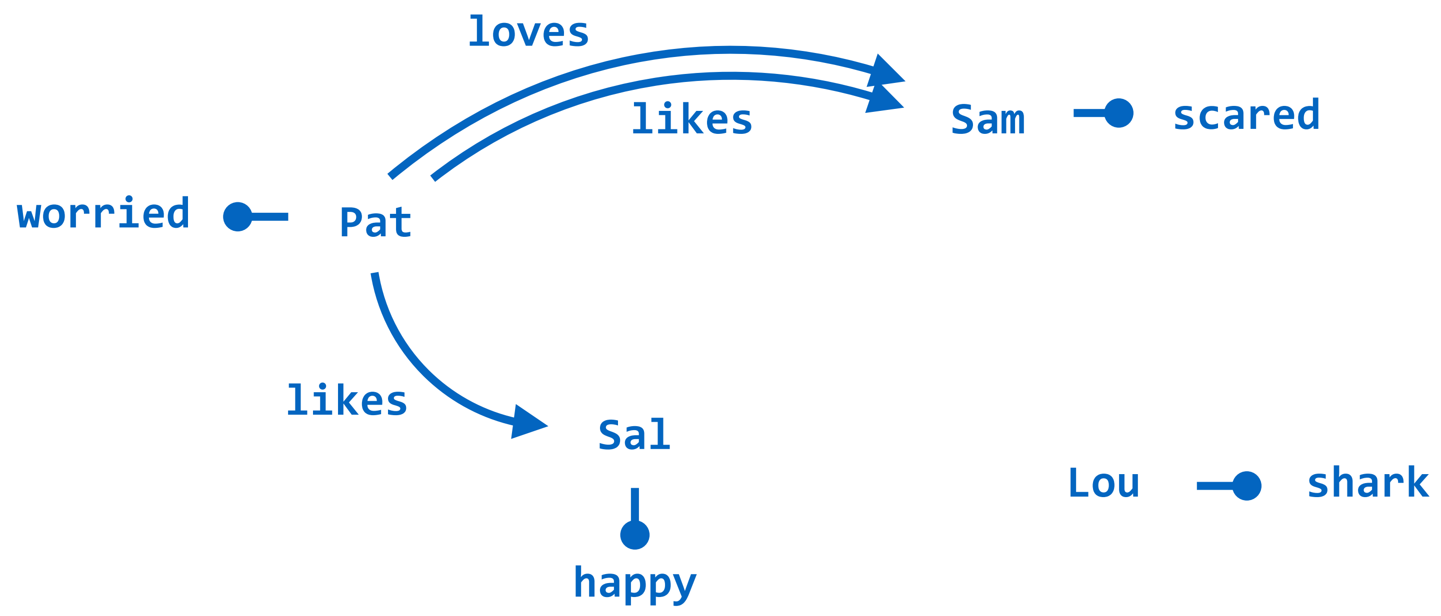
Key idea

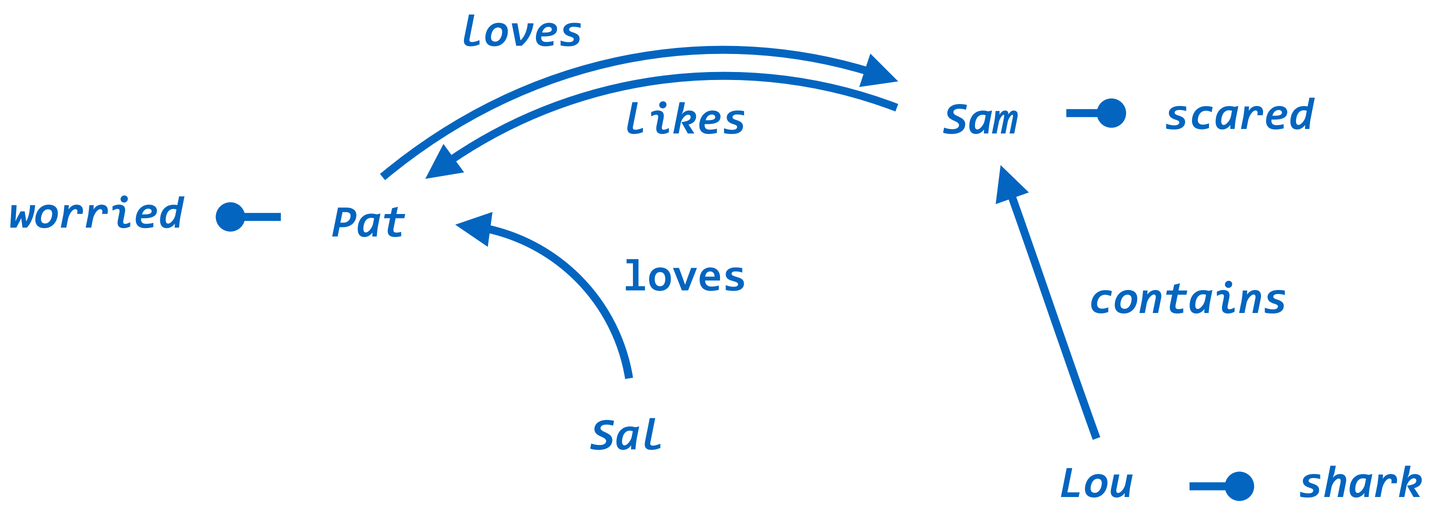
The meaning of a sentence is the set of possible worlds it picks out.

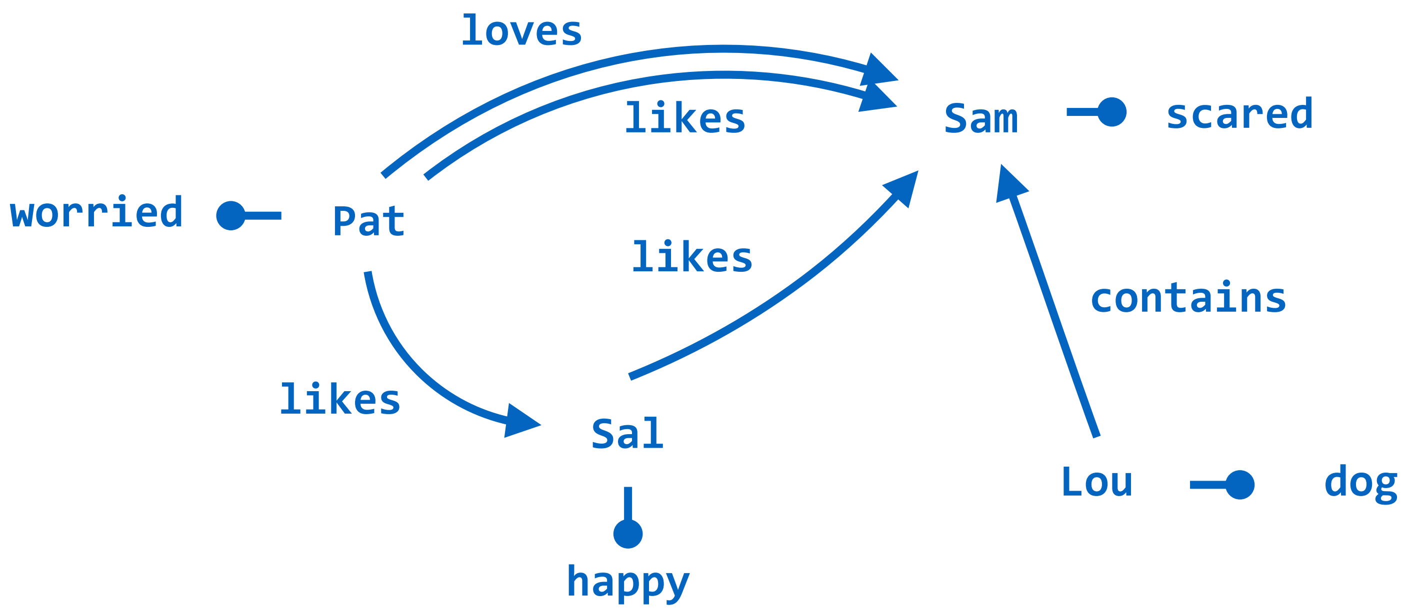
Possible worlds and logical forms

Explicit representation is too hard

Pat likes Sal.

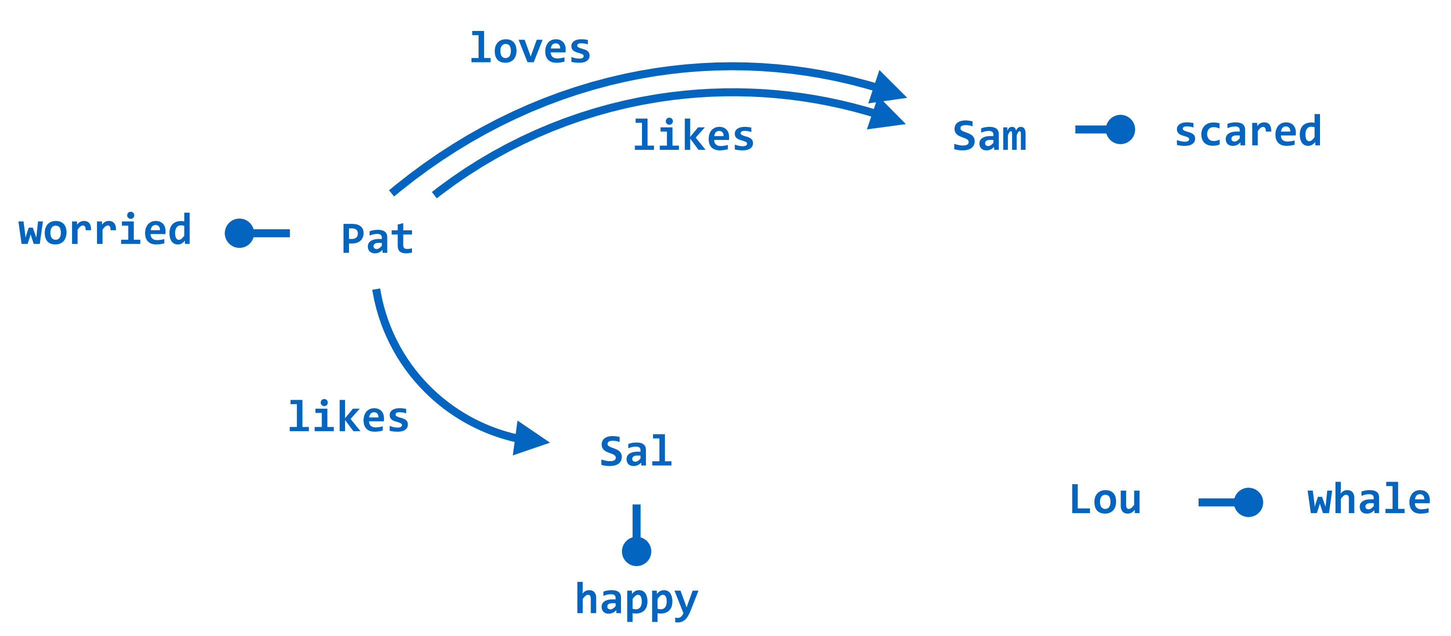






Meanings as functions

⟦*Pat likes Sal*⟧

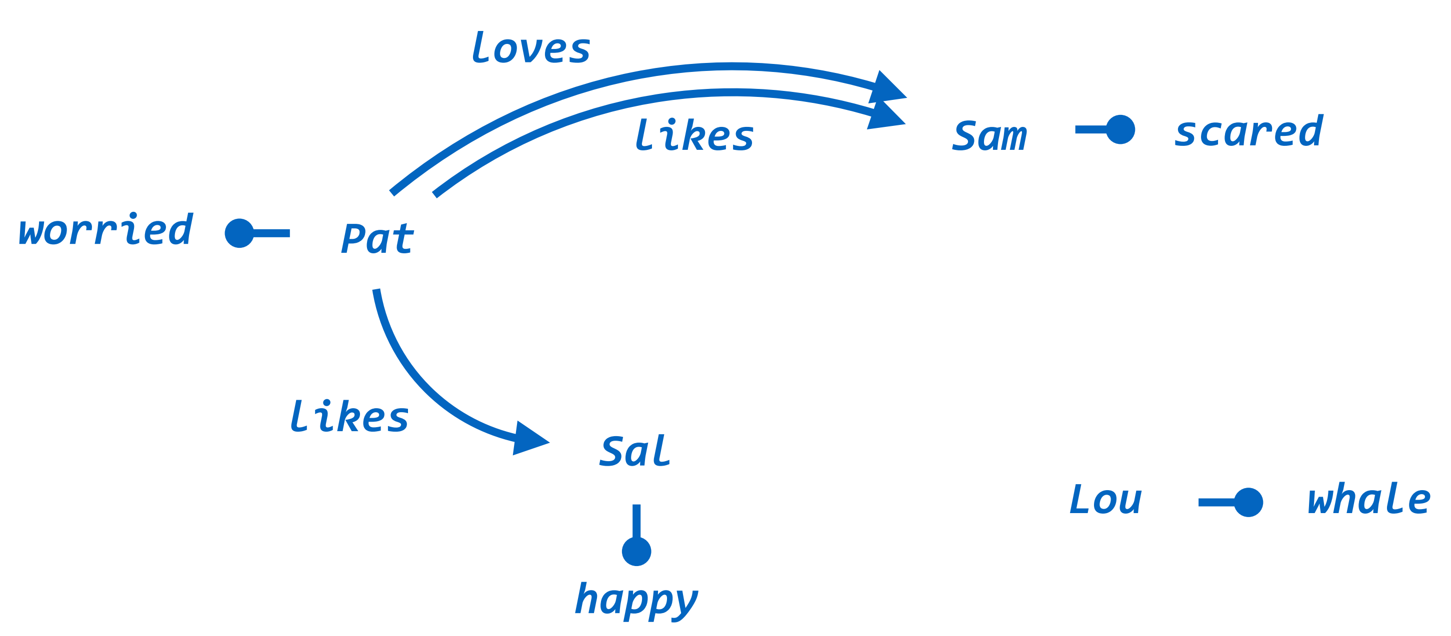


✔

Meanings as logical statements

⟦*Pat likes Sal*⟧

`likes(Pat, Sal)`



✔

Expressing functions with logic

Pat likes Sal

`likes(Pat, Sal)`

Meanings as logical statements

Lou is a shark

`shark(Lou)`

Meanings as logical statements

Sam is inside Lou, a shark

Meanings as logical statements

Sam is inside Lou, a shark

`shark(Lou) ∧ contains(Lou, Sam)`

Meanings as logical statements

Nobody likes Lou

Meanings as logical statements

Nobody likes Lou

$\forall x.\ \neg \mathrm{likes}(x, \mathrm{Lou})$

Meanings as logical statements

Everyone who knows Sal is happy

Meanings as logical statements

Everyone who knows Sal is happy

$$\forall x.\ \mathtt{knows}(x, \mathtt{Sal}) \rightarrow \mathtt{happy}(x)$$

Key idea

Collections of possible worlds can be compactly represented with logical forms.

Compositionality of meaning

Pat likes Sal	`likes(Pat, Sal)`
Lou is a shark	`shark(Lou)`
Sam is inside Lou, a shark	`shark(Lou) ∧ contains(Lou, Sam)`
Nobody likes Lou	`∀x.¬likes(x, Lou)`

Compositionality of meaning

Pat likes Sal	`likes(Pat, Sal)`
Lou is a shark	`shark(Lou)`
Sam is inside Lou, a shark	`shark(Lou) ∧ contains(Lou, Sam)`
Nobody likes Lou	`∀x.¬likes(x, Lou)`

Compositionality of meaning

Pat likes Sal	`likes(Pat, Sal)`
Lou is a shark	`shark(Lou)`
Sam is inside Lou, a shark	`shark(Lou) ∧ contains(Lou, Sam)`
Nobody likes Lou	`∀x.¬likes(x, Lou)`

Compositionality of meaning

A Sal le gusta Pat	`likes(Pat, Sal)`
Lou es un tiburón	`shark(Lou)`
Sam está dentro de Lou, un tiburón	`shark(Lou) ∧ contains(Lou, Sam)`
A nadie le gusta Lou	`∀x.¬likes(x, Lou)`

Compositionality of meaning

a12 b5 c67 a8	`likes(Pat, Sal)`
a12 b5 c0 a0	`shark(Lou)`
a12 b16 c12 c12	`shark(Lou) ∧ contains(Lou, Sam)`
a53	`∀x.¬likes(x, Lou)`

Key idea

Pieces of logical forms correspond to pieces of language

Building a lexicon

Sam is inside Lou, a shark `shark(Lou) ∧ contains(Lou, Sam)`

Pat: `Pat`

Sal: `Sal`

Sam: `Sam`

Lou: `Lou`

Building a lexicon

Sam is inside Lou, a shark `shark(Lou) ∧ contains(Lou, Sam)`

Pat: `Pat`

Sal: `Sal`

Sam: `Sam`

Lou: `Lou`

shark:

Building a lexicon

Sam is inside Lou, a shark `shark(Lou) ∧ contains(Lou, Sam)`

Pat: `Pat`

Sal: `Sal`

Sam: `Sam`

Lou: `Lou`

shark: `λx.shark(x)`

Building a lexicon

Sam is inside Lou, a shark `shark(Lou) ∧ contains(Lou, Sam)`

Pat: `Pat`

Sal: `Sal`

Sam: `Sam`

Lou: `Lou`

shark: `λx.shark(x)`

likes: `λyx.likes(x, y)`

nobody: `λf.∀x.¬f(x)`

...

Learning semantic parsers

Seq-to-seq semantic parsing

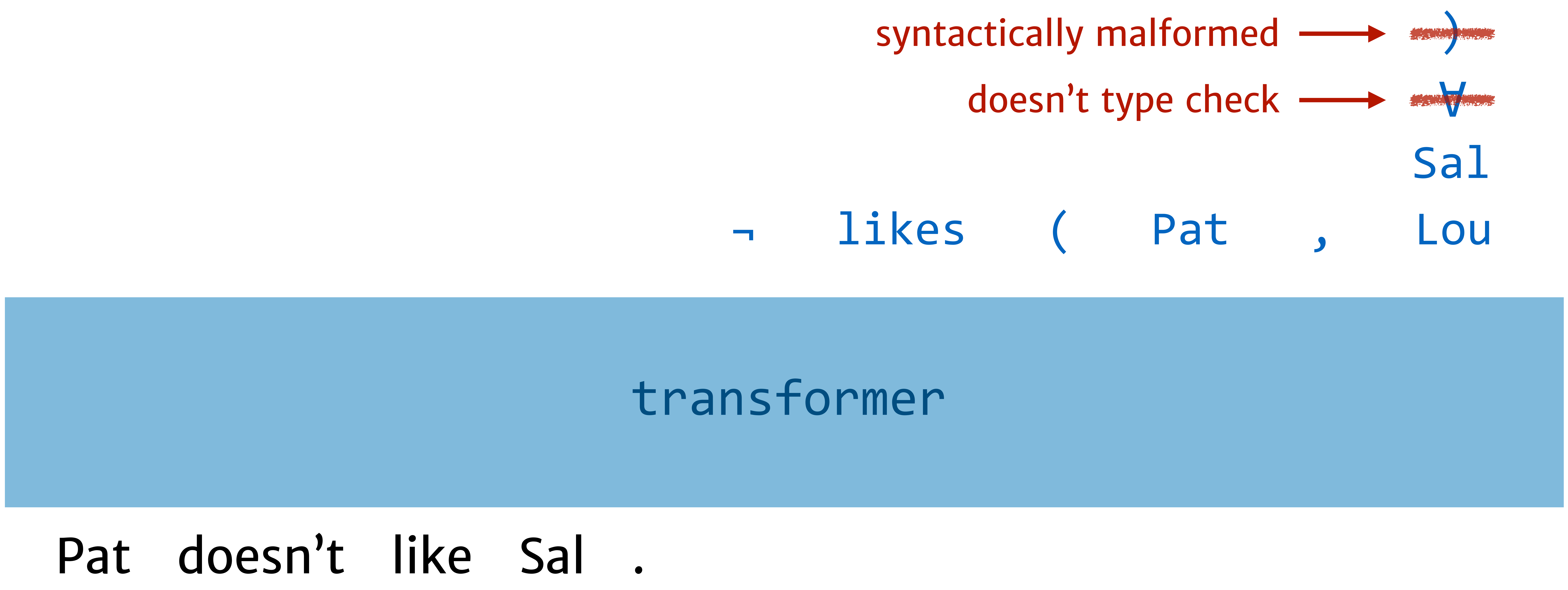
`¬ likes ( Pat , Sal )`

`transformer`

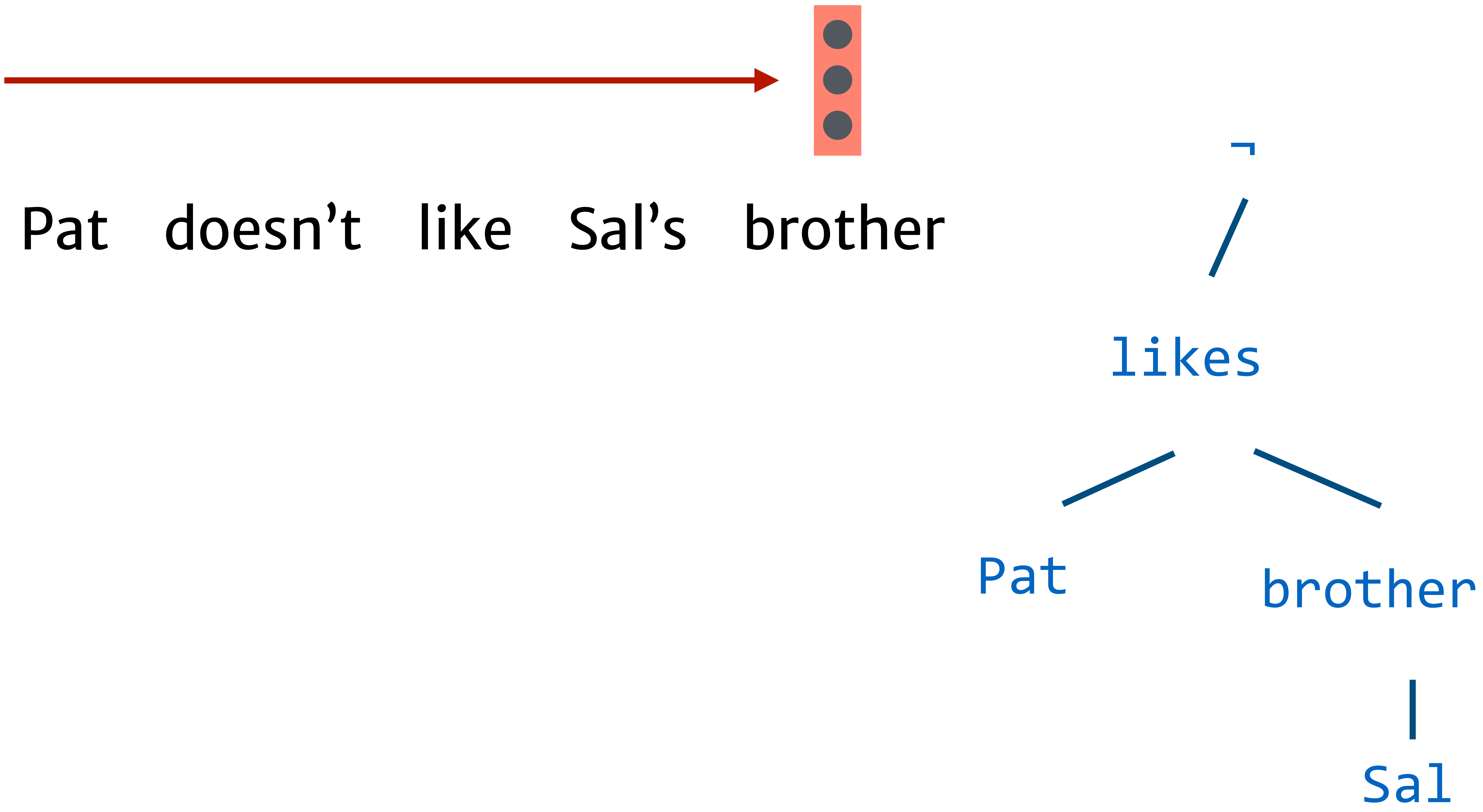
Pat doesn’t like Sal .

Decoder constraints

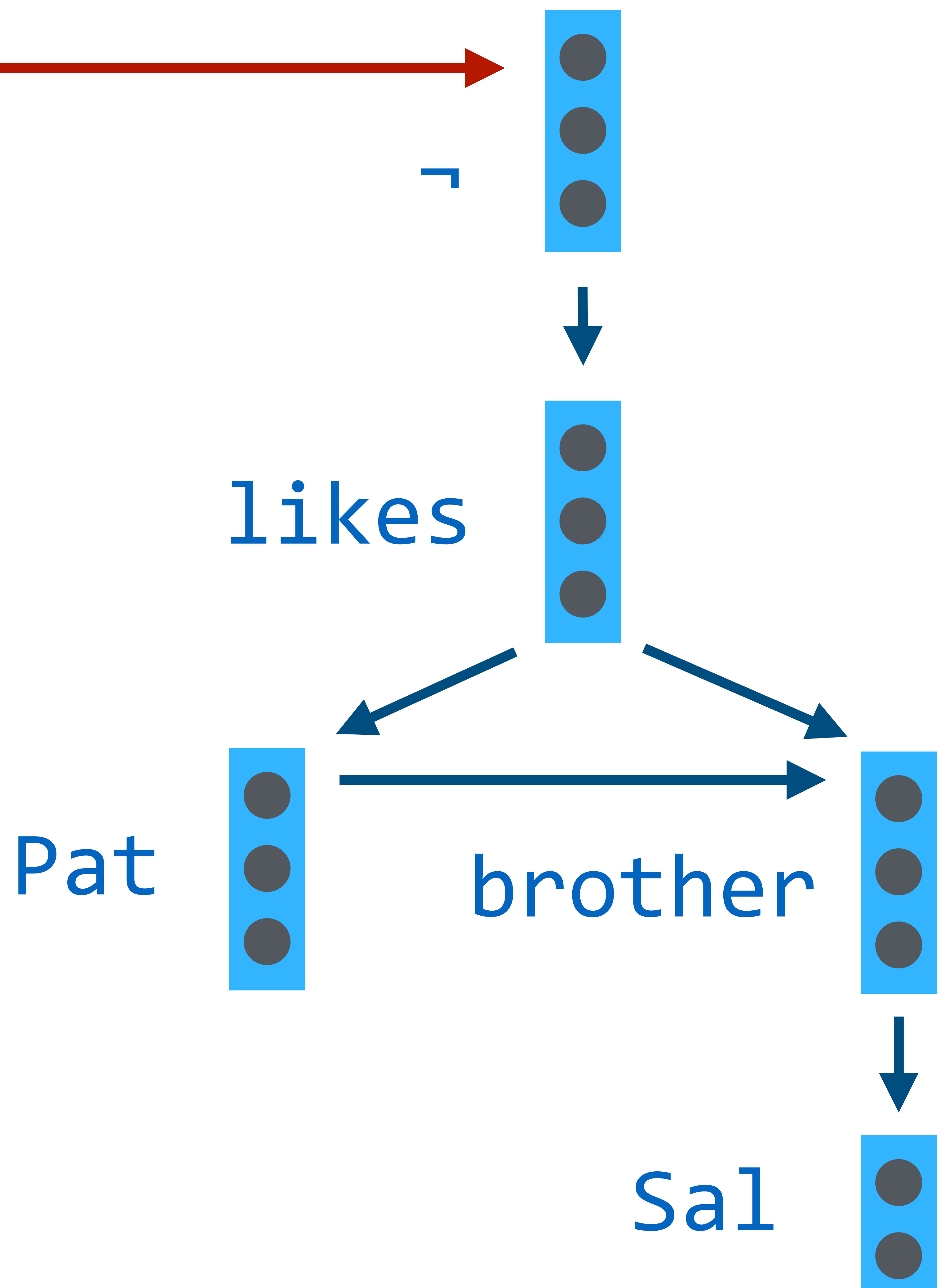
Decoder constraints



Tree-shaped decoders



Tree-shaped decoders



RNN states are updated based on parents and siblings, not arbitrary neighbors.

[e.g. Dong and Lapata 2016]

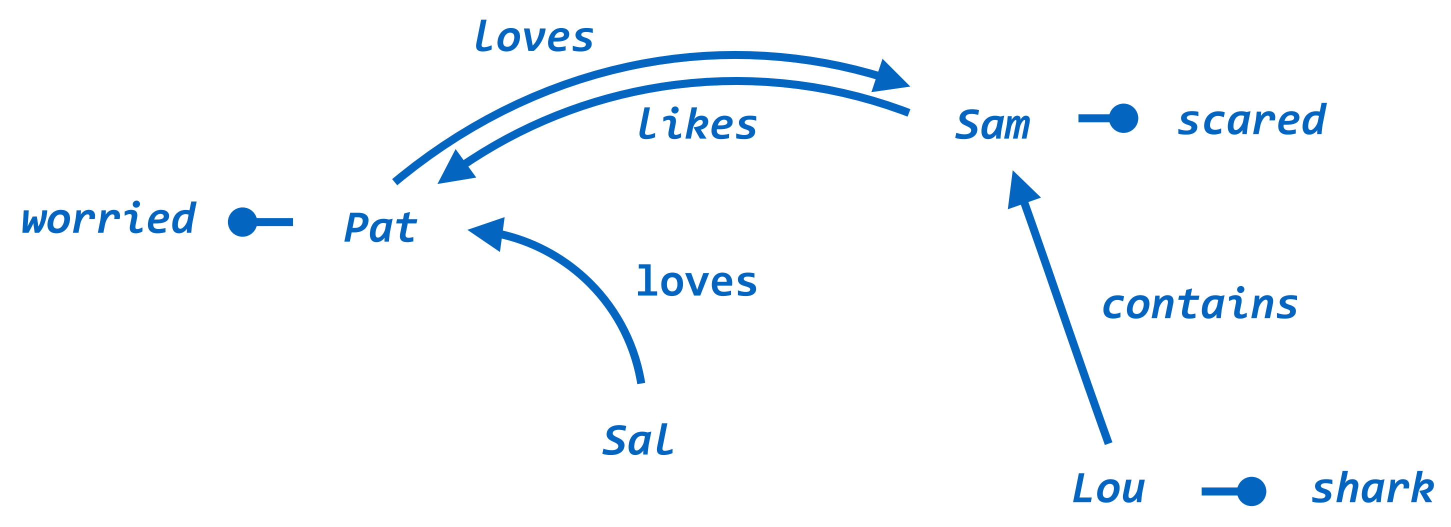
Learning from denotations

Logical form supervision:

Pat doesn't like Lou. ¬likes(Pat, Lou)

Answer supervision:
learn from (question, world, answer) triples without LFs!

Who does Pat like?



Sal

Maximum likelihood estimation

deterministic logical evaluation

semantic parser

$$p(\text{answer} \mid \text{question}) = \sum_{\text{LF}} p(\text{answer} \mid \text{LF})\; p(\text{LF} \mid \text{question})$$

Maximum likelihood estimation

deterministic logical evaluation

semantic parser

$$p(\text{answer} \mid \text{question}) = \sum_{\text{LF}} \underbrace{p(\text{answer} \mid \text{LF})}_{\text{deterministic logical evaluation}} \; \underbrace{p(\text{LF} \mid \text{question})}_{\text{semantic parser}}$$

compare:

syntactic parser

$$p(\text{sentence}) = \sum_{\text{tree}} p(\text{sentence} \mid \text{tree}) \; \underbrace{p(\text{tree})}_{\text{syntactic parser}}$$

Computational challenges

Can't efficiently compute this sum: no way to factor scoring fn over pieces of LFs.

no dynamic program!

$$p(\text{answer} \mid \text{question}) = \sum_{\text{LF}} p(\text{answer} \mid \text{LF})\, p(\text{LF} \mid \text{question})$$

dynamic program (CKY)

$$p(\text{sentence}) = \sum_{\text{tree}} p(\text{sentence} \mid \text{tree})\, p(\text{tree})$$

Computational challenges

Hard search problem!

This is 0 for almost all LFs

$$p(\text{answer} \mid \text{question}) = \sum_{\text{LF}} p(\text{answer} \mid \text{LF})\; p(\text{LF} \mid \text{question})$$

“Hard EM”

Alternate between:

$$\text{LF}^* = \operatorname{argmax}_{\text{LF}}\ p(\text{answer} \mid \text{LF})\ p(\text{LF} \mid \text{question}; \theta)$$

$$\theta^* = \operatorname{argmax}_{\theta}\ p(\text{answer} \mid \text{LF})\ p(\text{LF} \mid \text{question}; \theta)$$

(pick a “pseudo-gold”, treat it as gold, update params)

Lexicon-based semantic parsing

p(`λy. likes(Pat, y)` | *who does Pat like?*)

∝ exp { f(*like*, `λxy. likes(x, y)`) + f(*Pat*, `Pat`) + … }

`λy. likes(Pat, y)`

∅	∅	`Pat`	`λxy. likes(x, y)`
who	*does*	*Pat*	*like*

[Zettlemoyer & Collins 05, Artzi & Zettlemoyer 2013]

Semantic parsing via paraphrasing

1. Write a rule-based procedure for turning logical forms into sentences

`λy. likes(y, brother(Sal))` ⟶ *what likes brother of Sal*

2. Score LF based on similarity between the input sentence and fake one

$p(\text{LF} \mid \text{question}) \propto$ *f(who is it that likes Sal's brother,*
what likes brother of Sal)

use paraphrase features

[Berant and Liang 2014]

Aside: program synthesis

$$\max_{\boxed{\mathsf{LF}:\ p(\mathsf{answer}|\mathsf{LF})>0}} f(\mathbf{LF} \mid \mathbf{question})$$

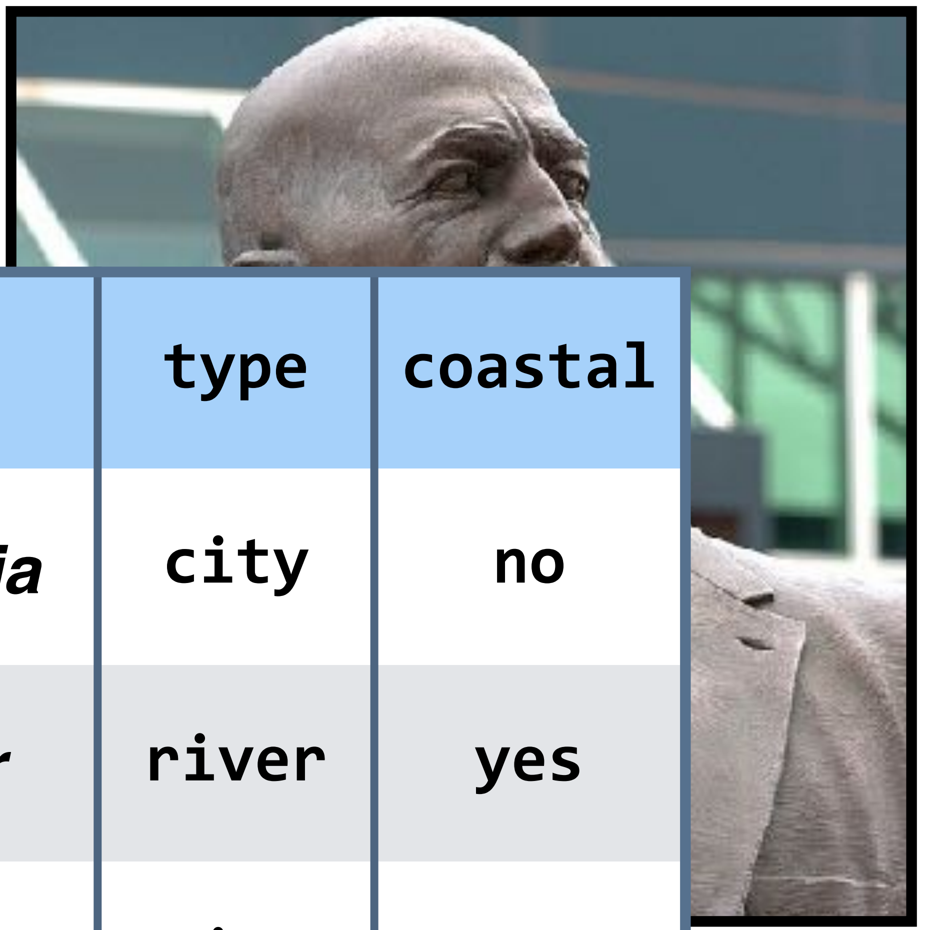
Huge amount of work on solving this problem in the programming languages literature!

(not widely used in NLP yet)

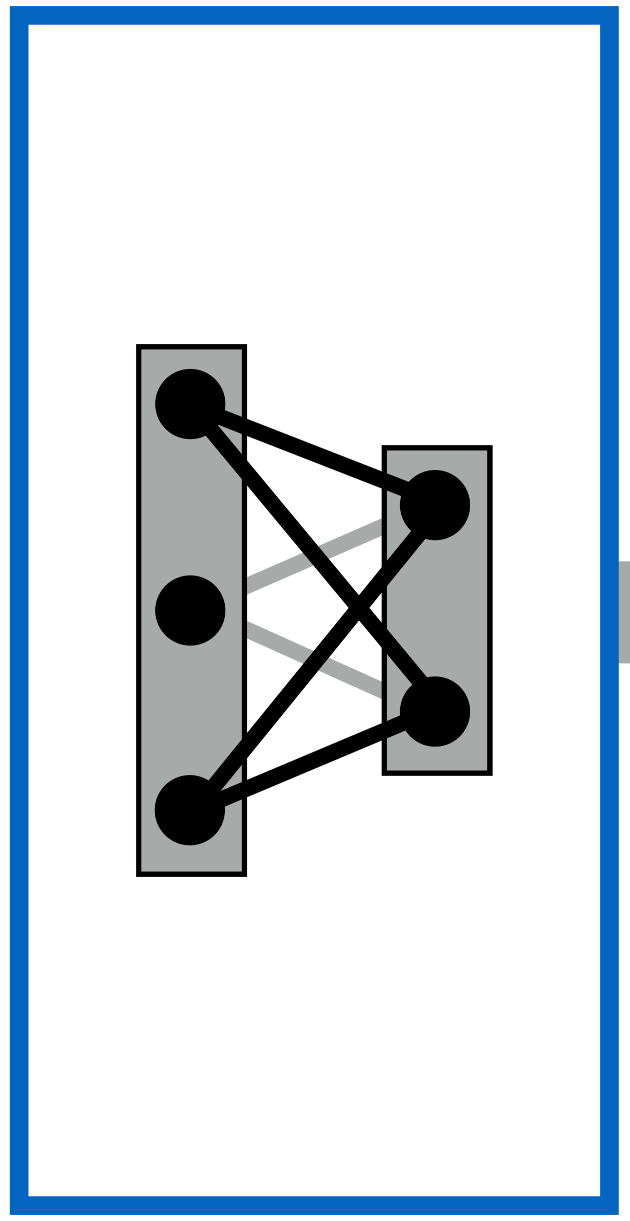
Why not just predict answers directly?

Still hard for "unstructured" neural models!

What color is the necktie?

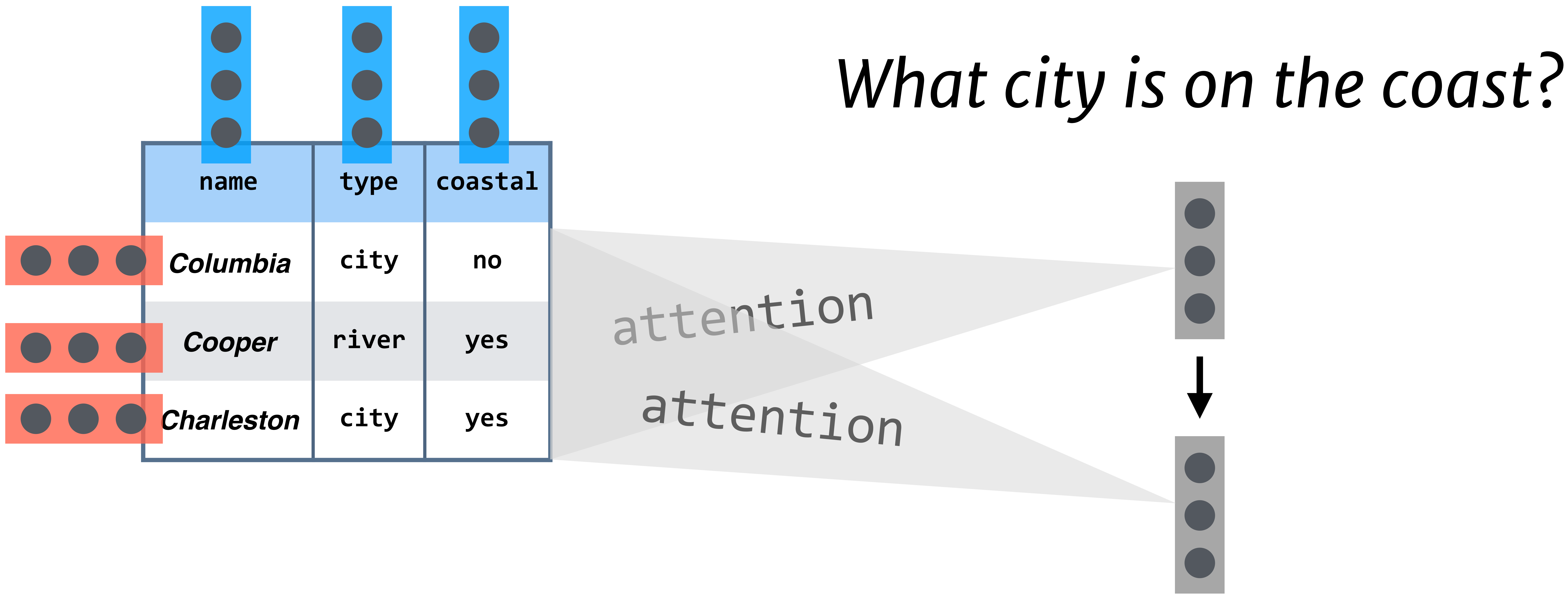


name	type	coastal
Columbia	city	no
Cooper	river	yes
Charleston	city	yes



yellow

Structured attention mechanisms

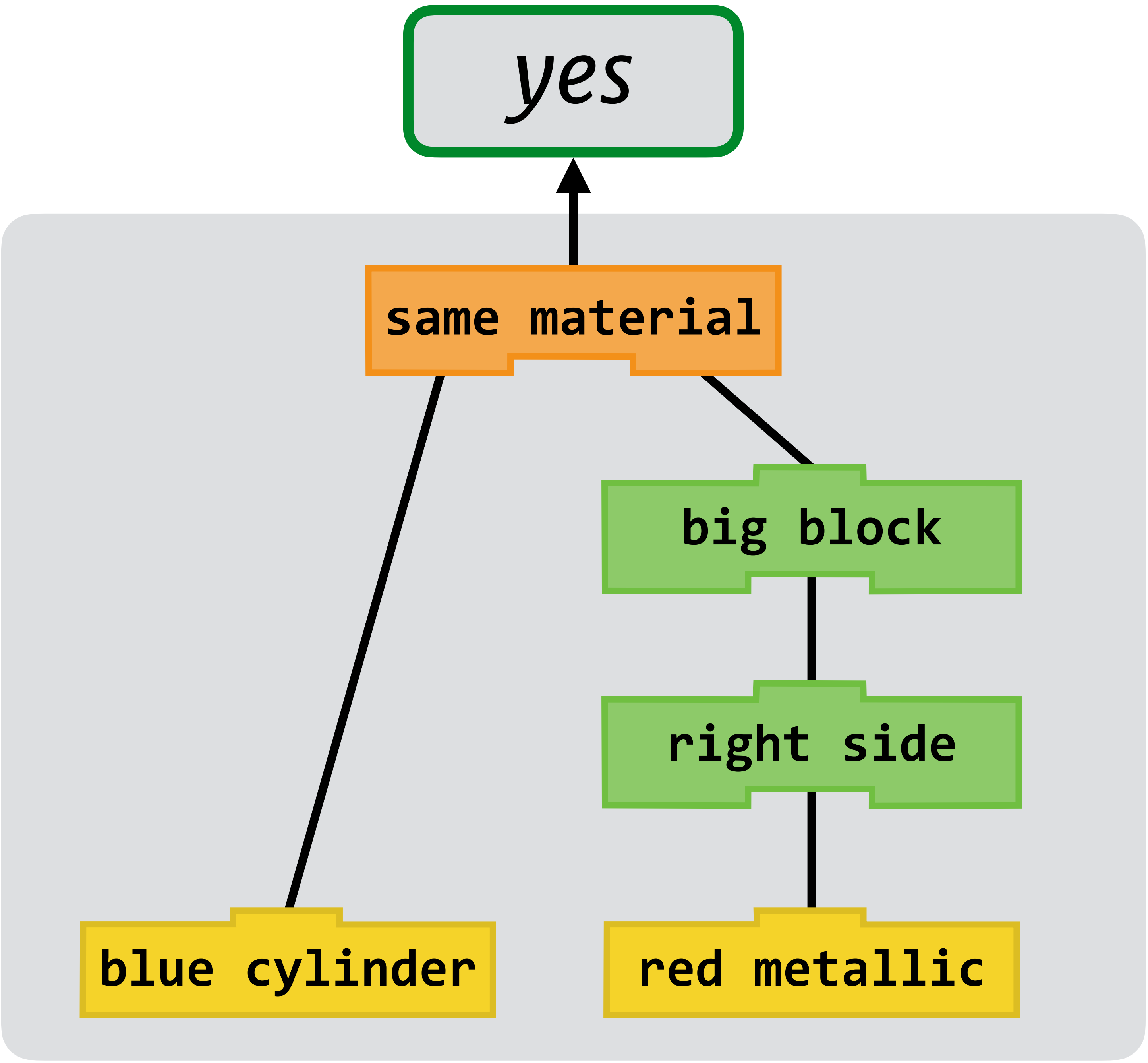


Key-value attention tailored
for tabular world representations

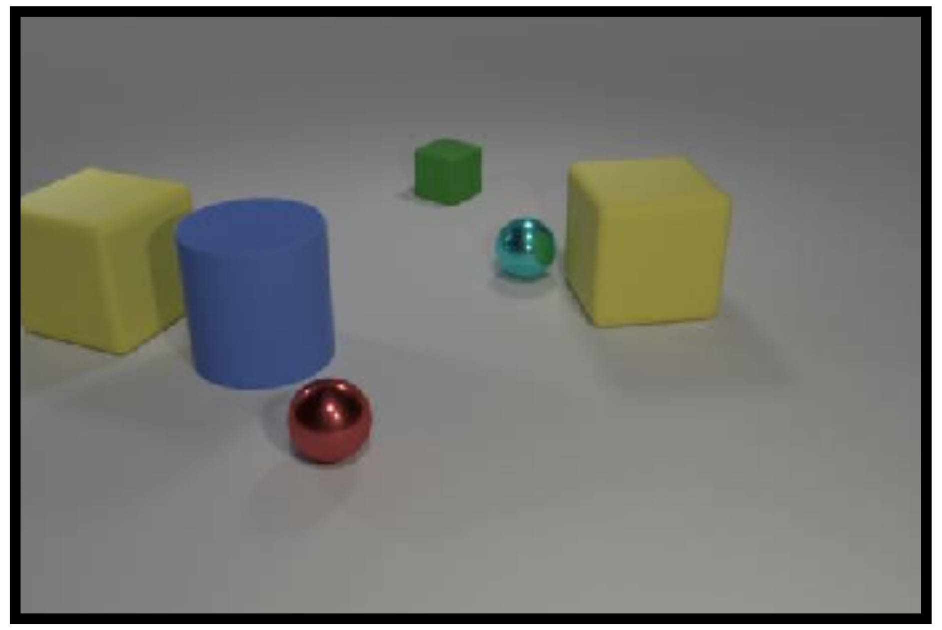
Charleston

[Yin et al. 2016]

Module networks



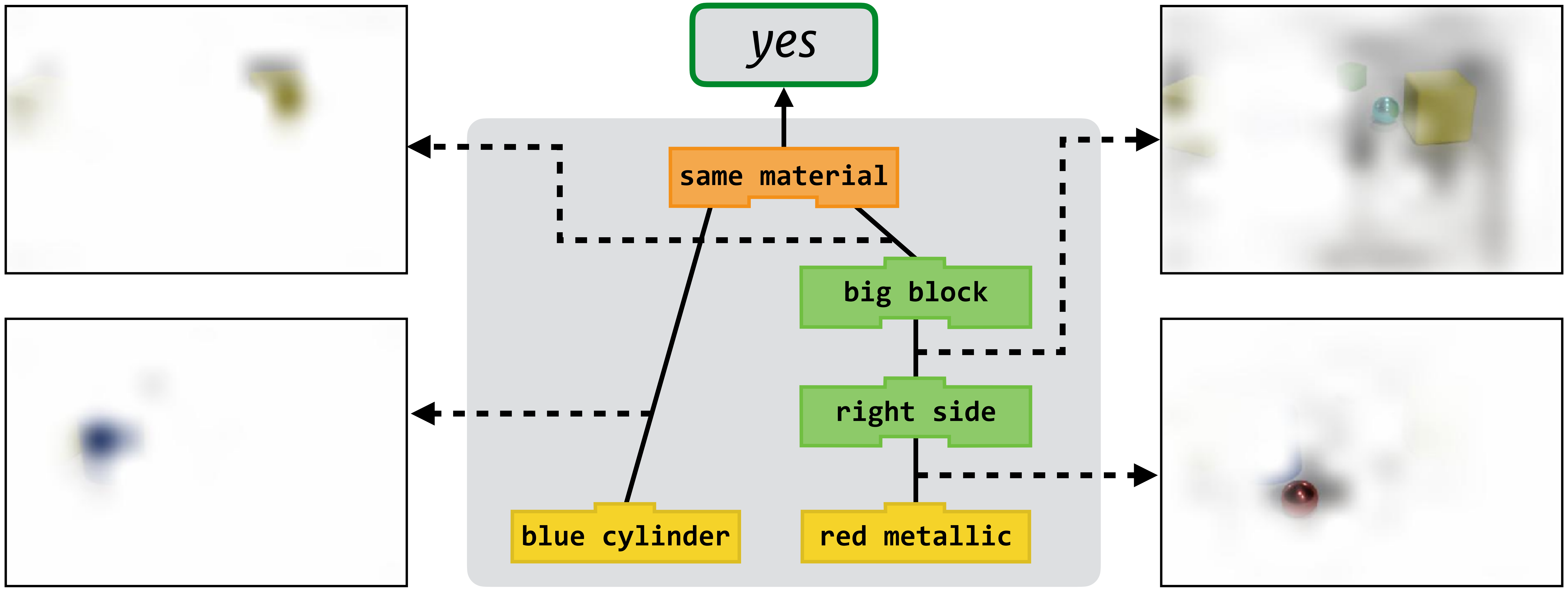

```
λw ∃xyz.
  eq(w,
     eq(material(x),
        material(y))
  blue_cylinder(x)
  big_block(y)
  red_metallic(z)
  right_side(y, z)
```



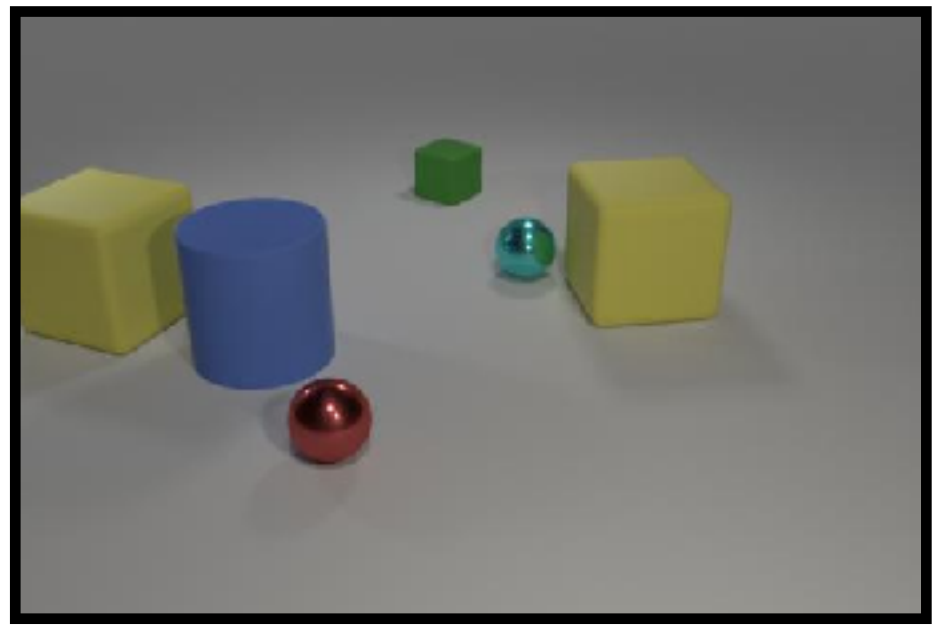
Does the blue cylinder have the same material as the big block on the right side of the red metallic thing?

[e.g. Andreas et al. 2016, Mao et al. 2019]

Module networks



No need to hand-write "logical" primitives!



Does the blue cylinder have the same material as the big block on the right side of the red metallic thing?

[e.g. Andreas et al. 2016, Mao et al. 2019]

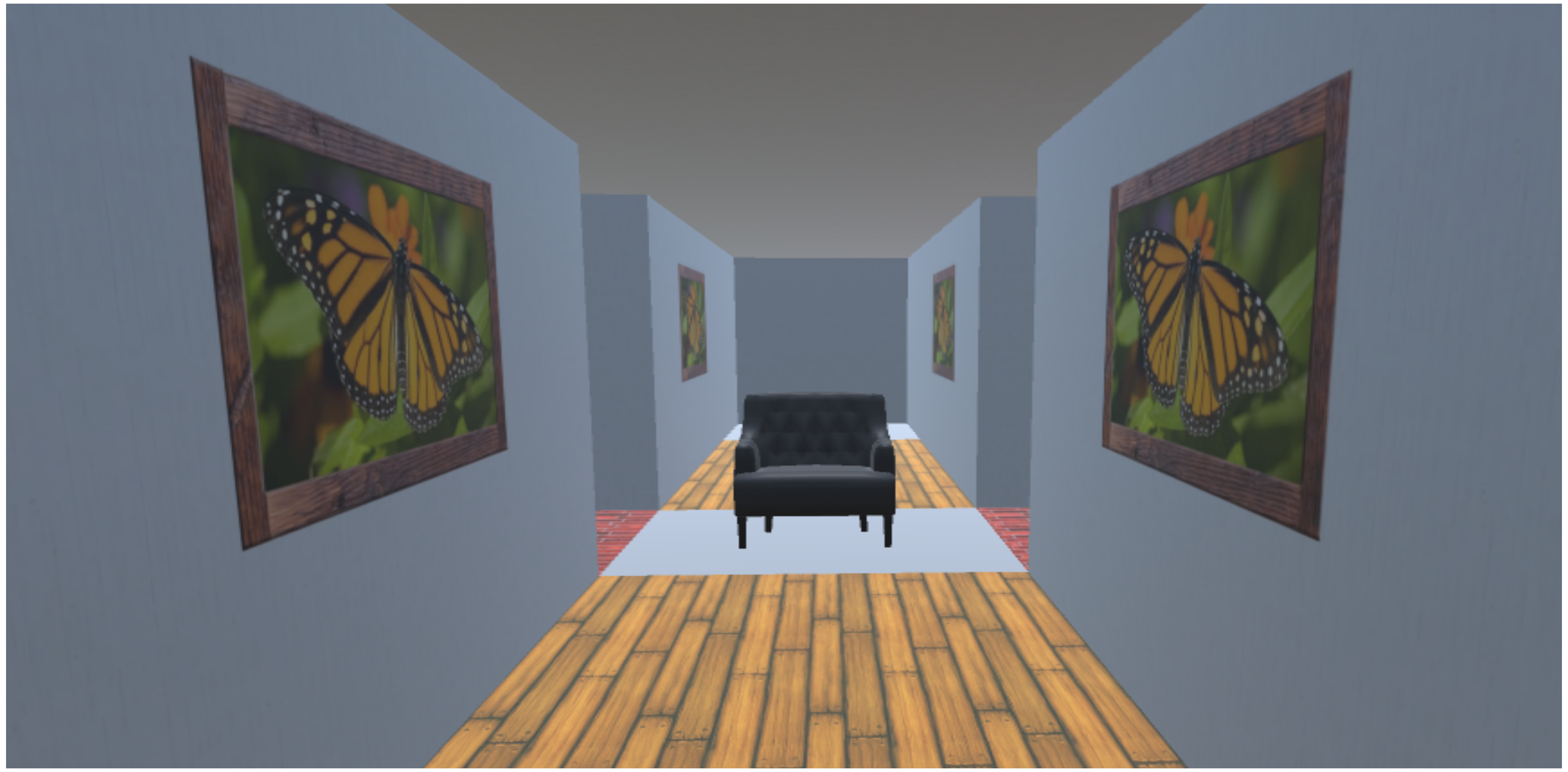
Question answering

Year	City	Country	Nations
1896	Athens	Greece	14
1900	Paris	France	24
1904	St. Louis	USA	12
...	...	...	...
2004	Athens	Greece	201
2008	Beijing	China	204
2012	London	UK	204

Greece last hosted the summer Olympics in which year?

[Pasupat & Liang 2015]

Instruction following



move forward twice to the chair

$$\lambda a.move(a) \wedge dir(a, forward) \wedge len(a, 2) \wedge \\ to(a, \iota x.chair(x))$$

at the corner turn left to face the blue hall

$$\lambda a.pre(a, \iota x.corner(x)) \wedge turn(a) \wedge dir(a, left) \wedge \\ post(a, front(you, \iota x.blue(x) \wedge hall(x)))$$

[Artzi & Zettlemoyer 2015]

Next class: ???