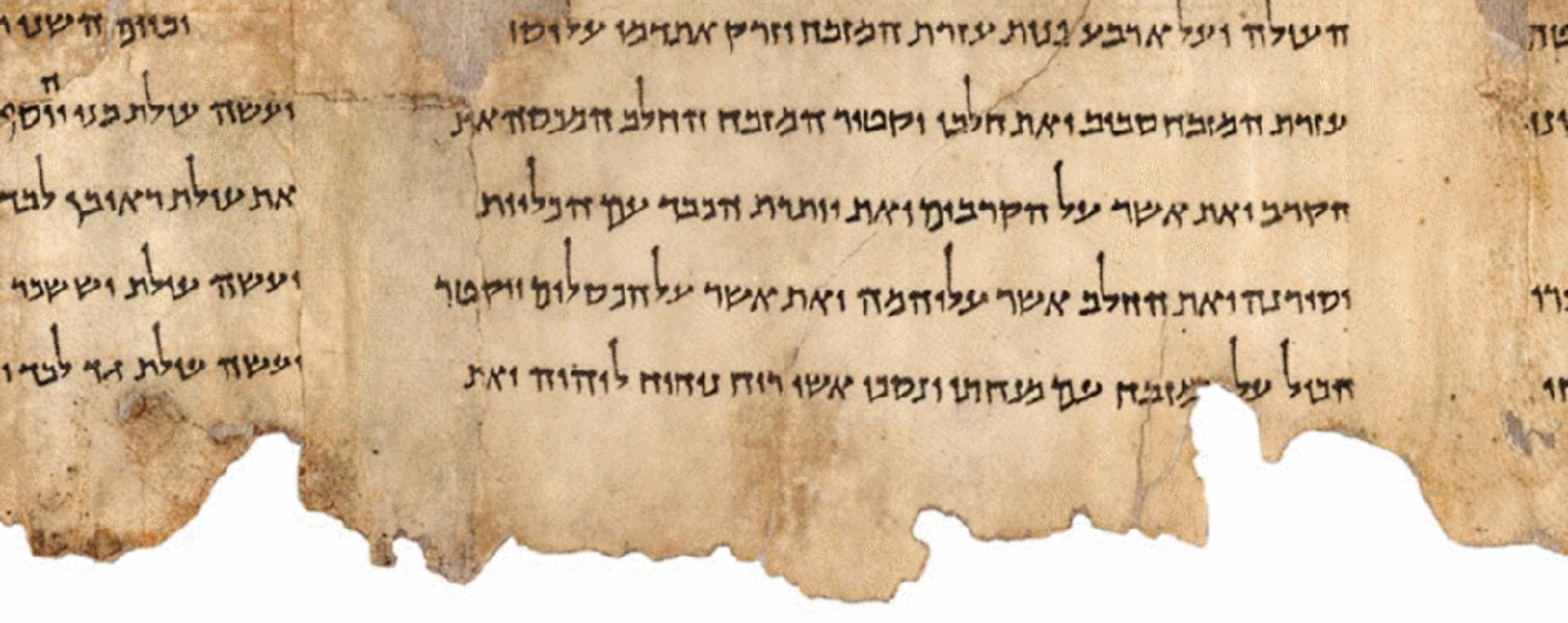


Natural Language Processing

Info 159/259

Lecture 18: Semantics (Oct 25, 2018)

David Bamman, UC Berkeley

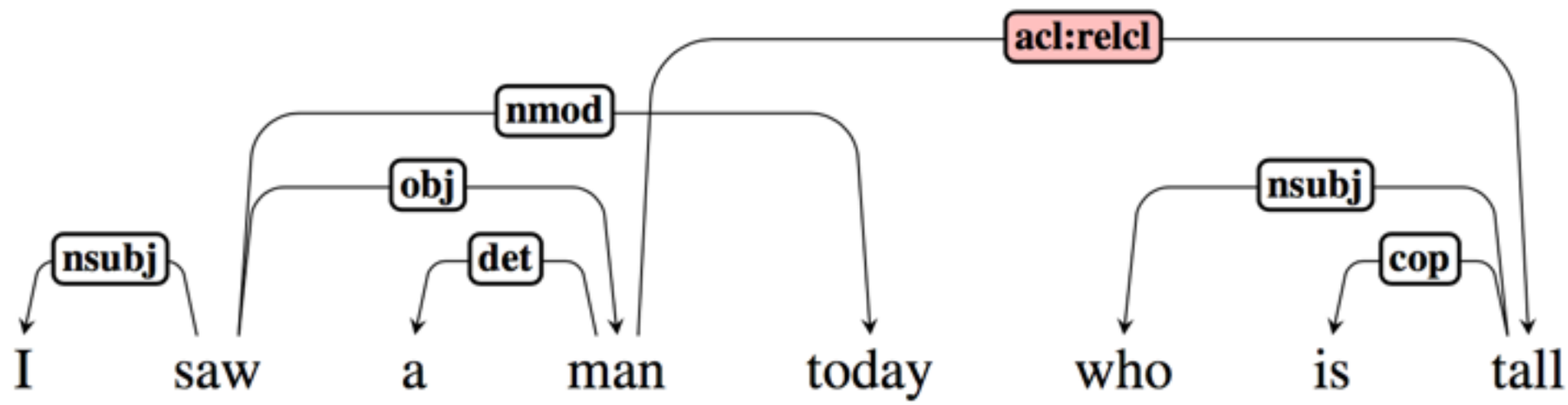
Graph-based parsing

- For a given sentence S, we want to find the highest-scoring tree among all possible trees for that sentence $\mathcal{G}_S$

$$\hat{T}(S) = \arg\max_{t \in \mathcal{G}_S} \text{score}(t, S)$$

- Edge-factored scoring: the total score of a tree is the sum of the scores for all of its edges (arcs):

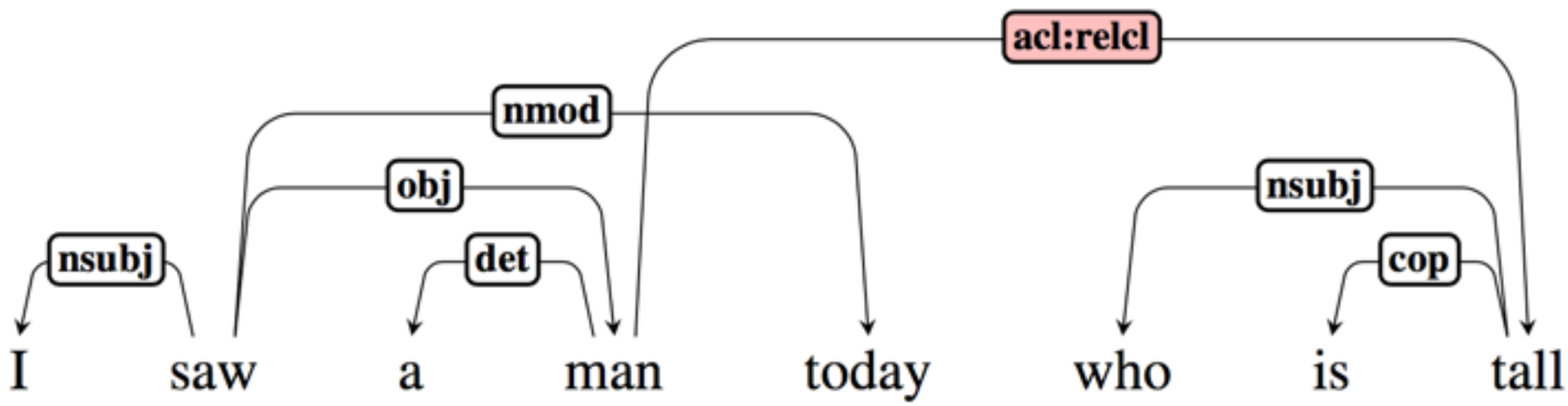
$$\text{score}(t, S) = \sum_{e \in t} \text{score}(e)$$



Edge-factored features

- Word form of head/dependent
- POS tag of head/dependent
- Distributed representation of h/d
- Distance between h/d
- POS tags between h/d
- Head to left of dependent?

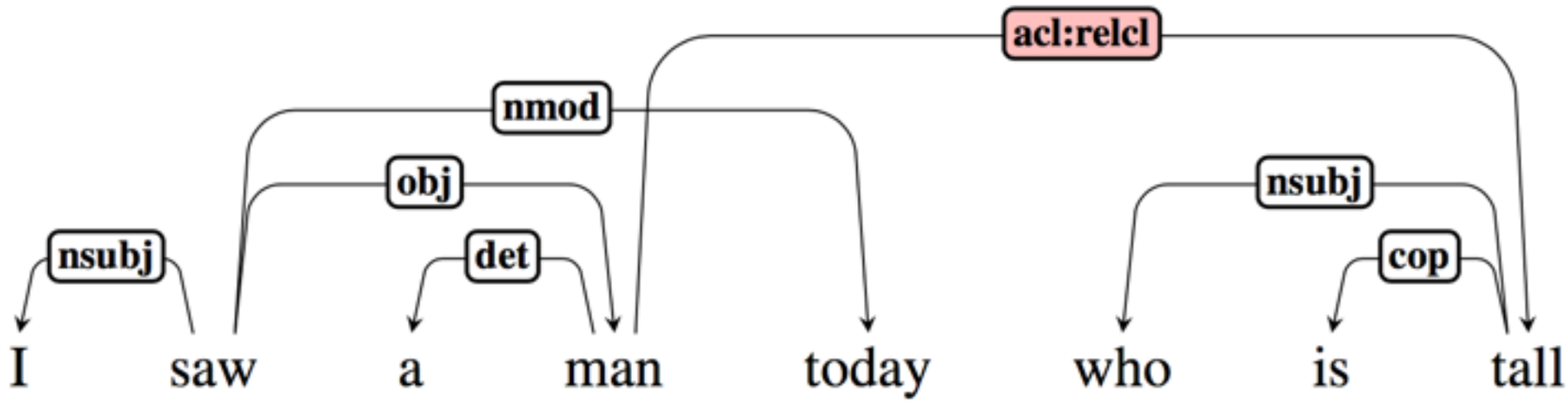
$head_t$ = man	1
$head_{pos}$ = NN	1
distance	4
$child_{pos}$ = JJ and $head_{pos}$ = NN	1
$child_{pos}$ = NN and $head_{pos}$ = JJ	0



$$\text{score}(e) = \sum_{i=1}^{F} x_i \beta_i$$

Feature value

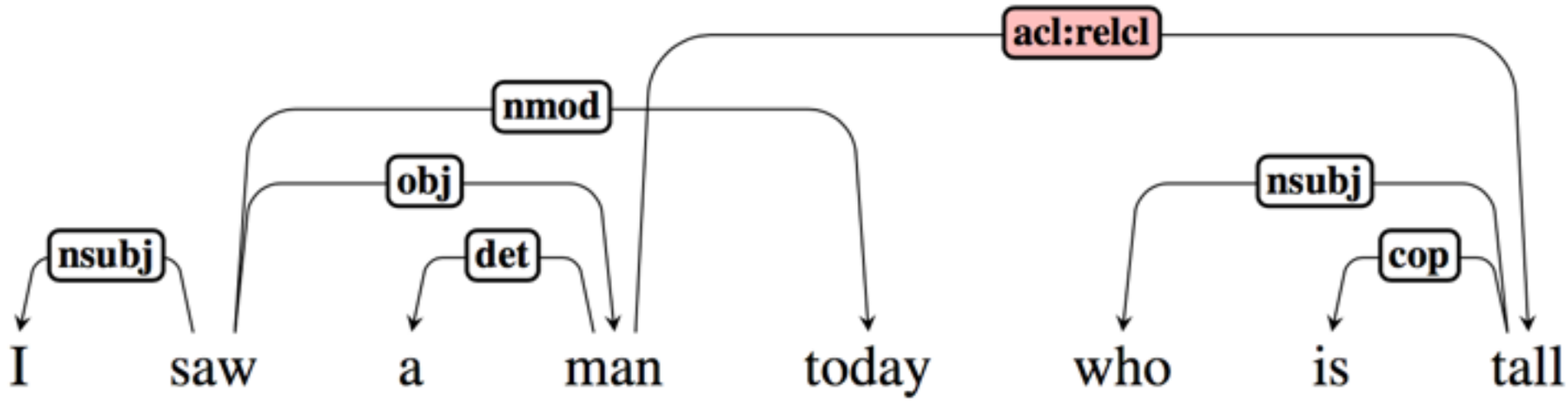
Learned coefficient for that feature

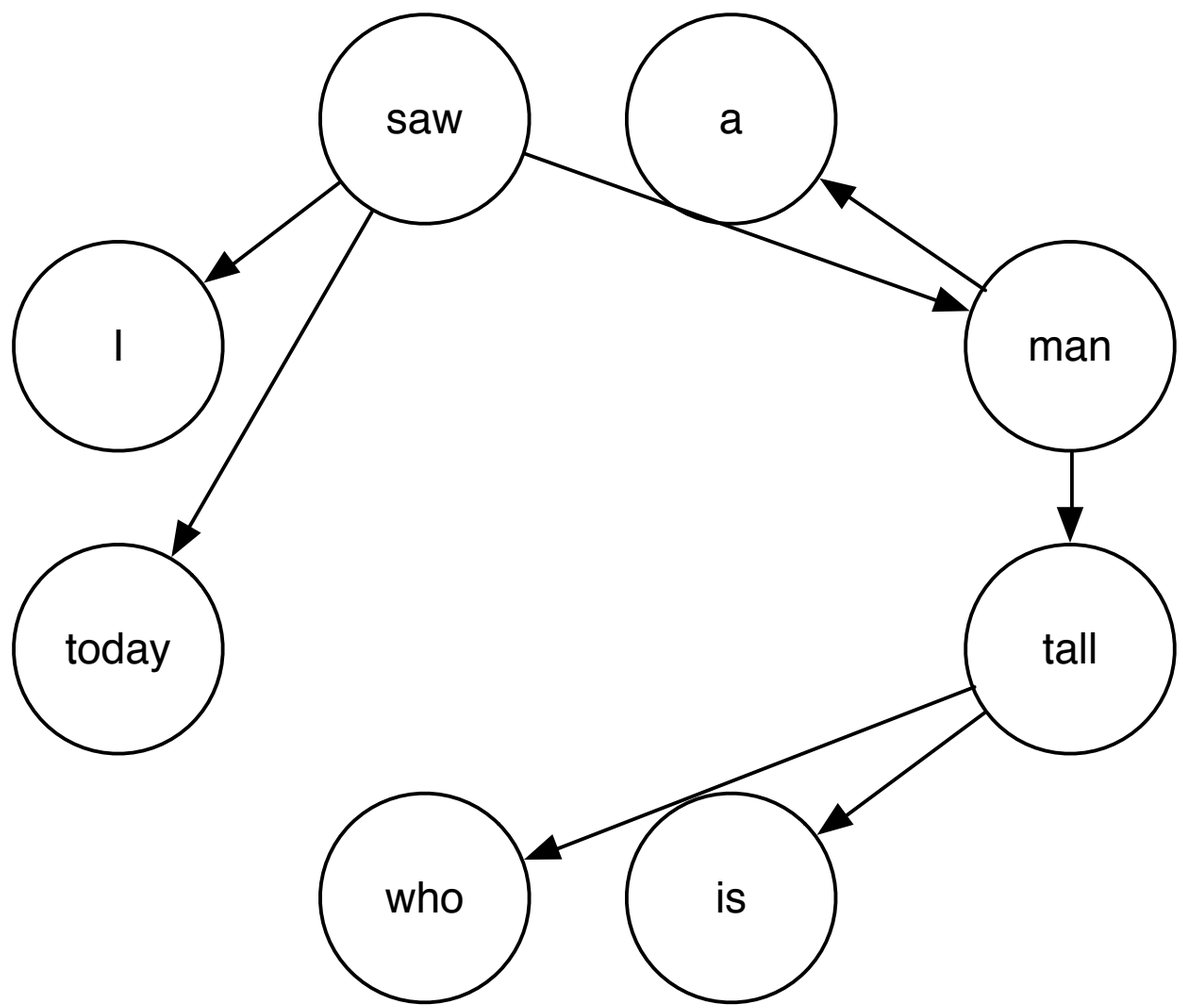


$$\text{score}(e) = \sum_{i=1}^{F} x_i \beta_i$$

	x	β
head_t = man	1	3.7
head_t = man	1	1.3
distance	4	0.7
child_{pos} = JJ and	1	0.3
child_{pos} = NN and	0	-2.7

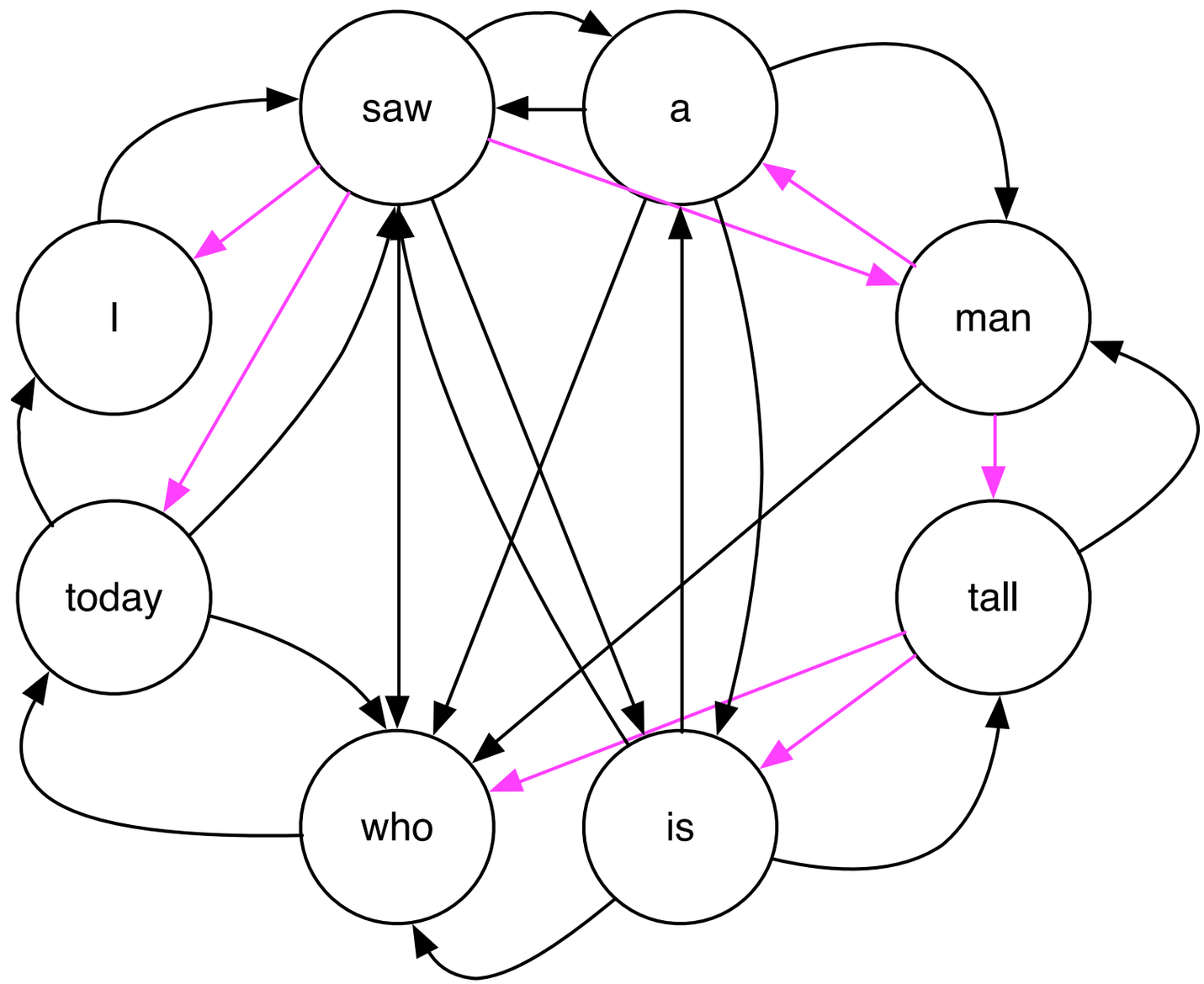
$$\text{score}(e) = 8.1$$


acl:relcl
nmod
obj
nsubj
det
nsubj
cop
I
saw
a
man
today
who
is
tall


saw
a
I
man
today
tall
who
is

MST Parsing

- We start out with a fully connected graph with a score for each edge
- N^2 edges total

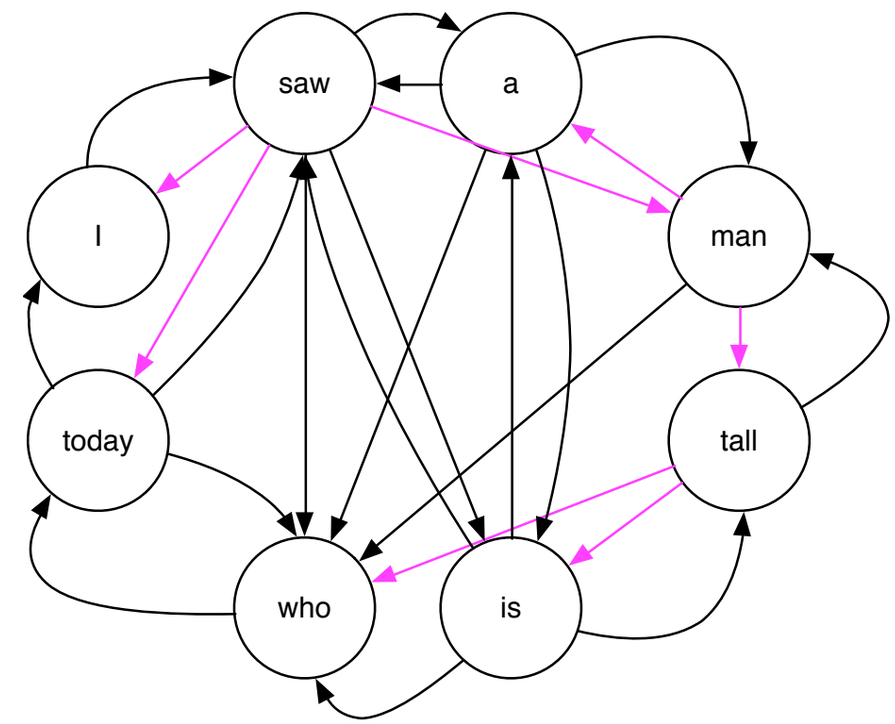


(Assume one edge connects each node as dependent and node as head, N^2 total)

MST Parsing

- From this graph G, we want to find a spanning tree (tree that spans G [includes all the vertices in G])

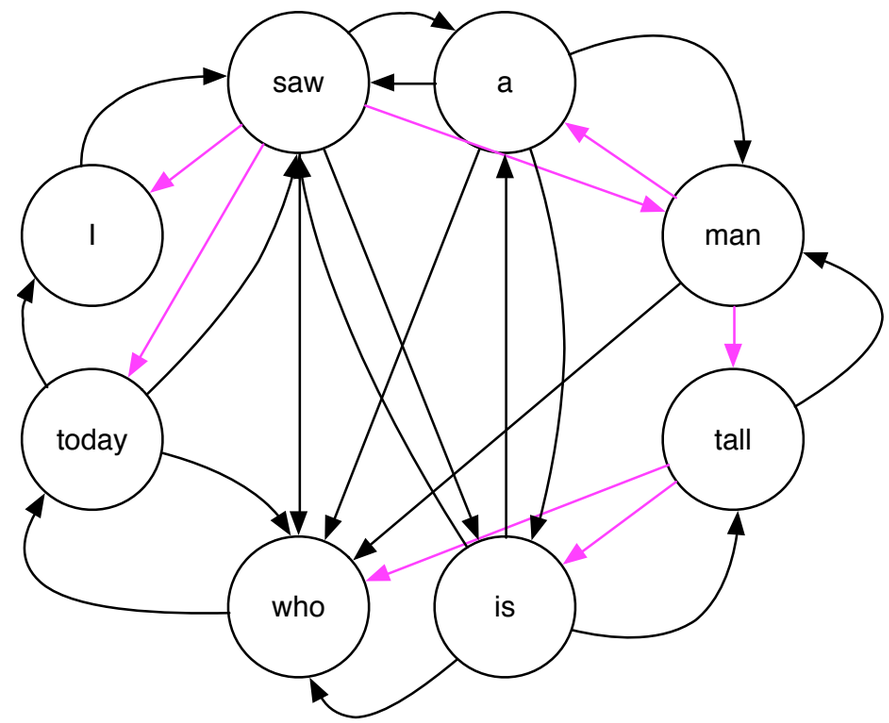
- If the edges have weights, the best parse is the maximal spanning tree (the spanning tree with the highest total weight).



MST Parsing

- To find the MST of any graph, we can use the Chu-Liu-Edmonds algorithm in $O(n^3)$ time.

- More efficient Gabow et al. find the MST in $O(n^2+n \log n)$



Learning

$$\hat{T}(S) = \arg\max_{t \in \mathcal{G}_S} \text{score}(t, S)$$

ϕ is our feature vector scoped over the source dependent, target head and entire sentence x

both are vectors

$$\hat{T}(S) = \arg\max_{t \in \mathcal{G}_S} \sum_{e \in E} \phi(e, x)^\top \beta$$

$$\hat{T}(S) = \arg\max_{t \in \mathcal{G}_S} \left[\sum_{e \in E} \phi(e, x) \right]^\top \beta$$

Learning

$$\hat{T}(S) = \arg\max_{t \in \mathcal{G}_S} \text{score}(t, S)$$

- Given this formulation, we want to learn weights for β that make the score for the gold tree higher than for all other possible incorrect trees.

- That's expensive, so let's just try to make the score for the gold tree higher than the highest scoring tree we predict (if it's wrong)

Learning

$$\left[\sum_{e \in E} \phi(e, x)\right]^\top \beta = \Phi_{gold}(E, x)^\top \beta$$

score for gold tree in treebank

$$\left[\sum_{e \in \hat{E}} \phi(e, x)\right]^\top \beta = \hat{\Phi}_{gold}(\hat{E}, x)^\top \beta$$

score for argmax tree in our model

Learning

- We can optimize this using SGD by taking the derivative with respect to the difference in scores (which we want to maximize):

$$\Phi_{gold}(E, x)^\top \beta - \hat{\Phi}_{pred}(\hat{E}, x)^\top \beta$$

$$= \left[\Phi_{gold}(E, x) - \hat{\Phi}_{pred}(\hat{E}, x)\right]^\top \beta$$

$$\frac{\partial}{\partial \beta} \left[\Phi_{gold}(E, x) - \hat{\Phi}_{pred}(\hat{E}, x)\right]^\top \beta = \Phi_{gold}(E, x) - \hat{\Phi}_{pred}(\hat{E}, x)$$

Perceptron

Data: training data $x \in R^F$, $y \in \{-1, +1\}$, i = 1 ... N;
initialize $\beta_0 = 0^F$;
k=0;
while *not converged* **do**
  k = k + 1;
  i = k mod N;
  if $\hat{y}_i \neq y_i$ **then**
    $\beta_{t+1} = \beta_t + y_i x_i$
  else
    do nothing;
  end
end

Perceptron update for binary classification = adding the feature values to the current estimate of β

Structured Perceptron

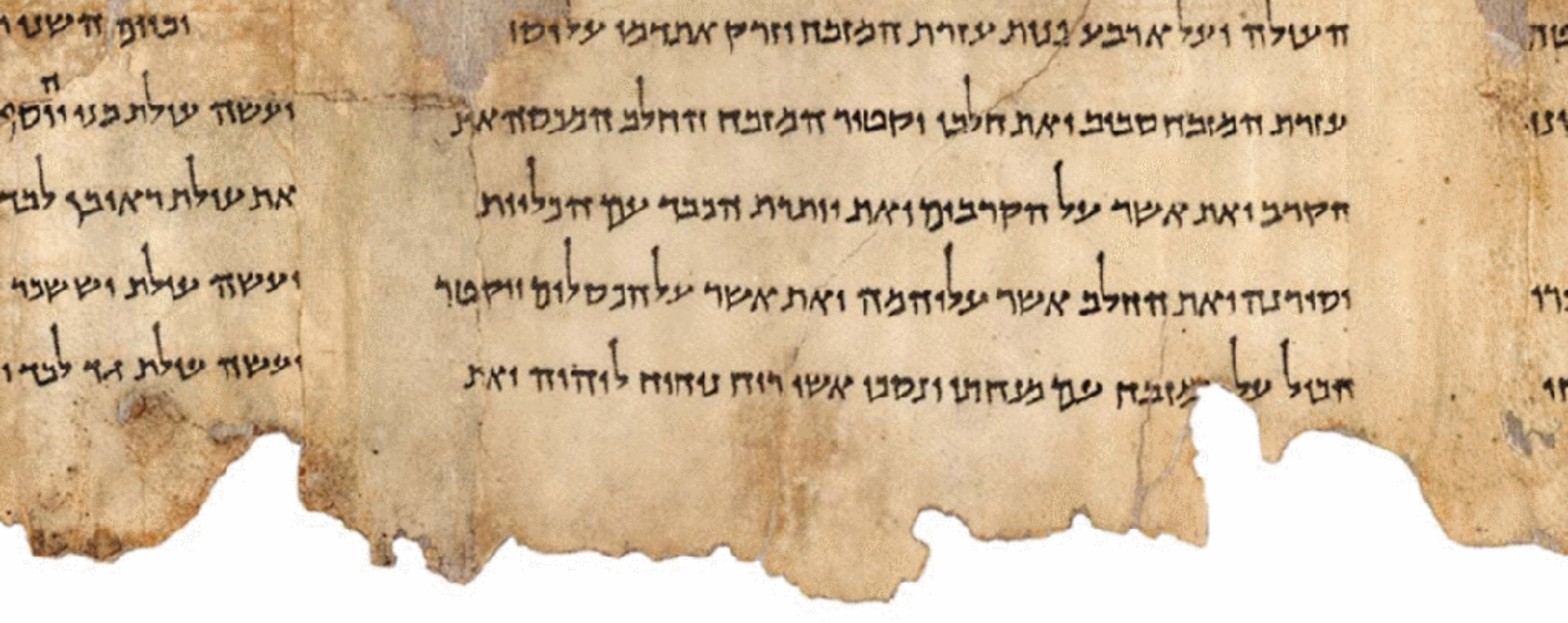
Algorithm 1 Structured perceptron

```
function PerceptronUpdate(x, E, β)
    Φ_gold(E, x) ← createFeatures(x, E)
    Φ̂_pred(Ê, x) ← createFeatures(x, Ê)
    Ê ← CLU(x, β)
    β ← β + Φ_gold(E, x) − Φ̂_pred(Ê, x)
end function
```

Create feature vector from true tree

Use CLU to find best tree given scores from current β

Update β with the different between the feature vectors



Natural Language Processing

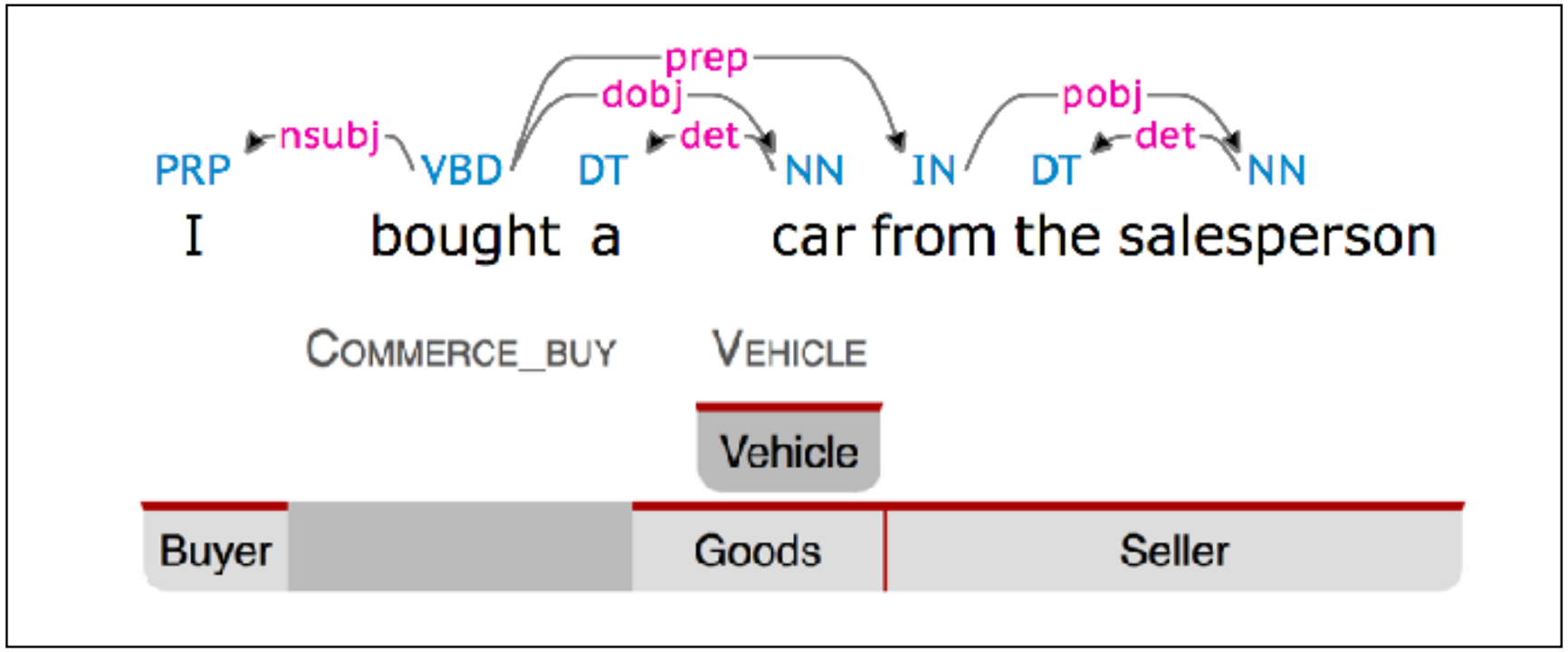
Info 159/259

Lecture 18: Semantics (Oct 25, 2018)

David Bamman, UC Berkeley

Why is syntax important?

- Foundation for semantic analysis (on many levels of representation: semantic roles, compositional semantics, frame semantics)

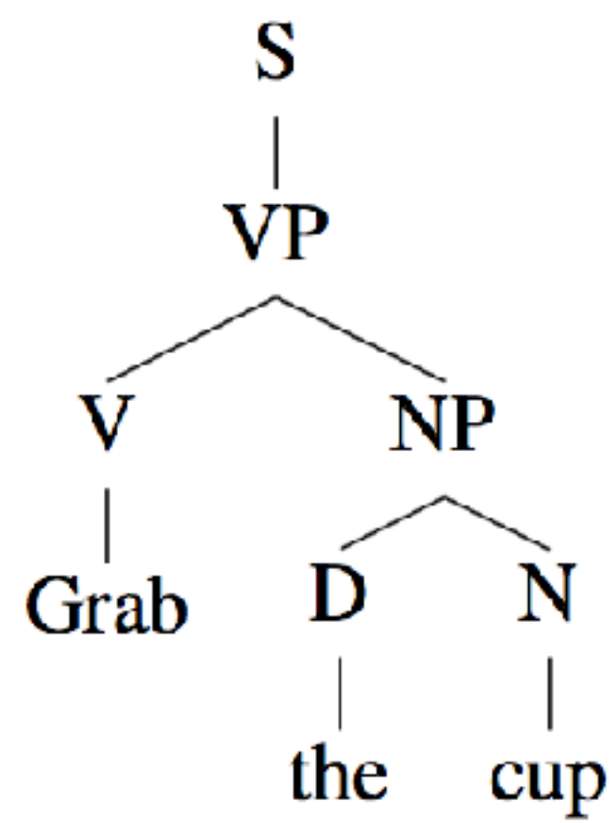


http://demo.ark.cs.cmu.edu

From 10/5

Why is syntax insufficient?

- Syntax encodes the structure of language but doesn't directly address meaning.



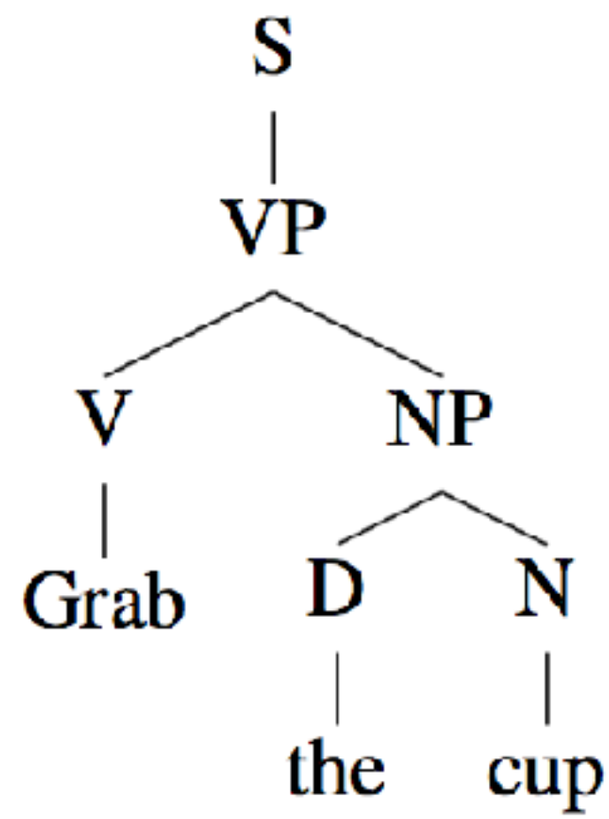
- Syntax alone doesn't ground "grab" in an action to take in the world.

Lexical semantics

- Vector representation that encodes information about the distribution of contexts a word appears in
- Words that appear in similar contexts have similar representations (and similar meanings, by the distributional hypothesis).
- We can represent what individual words “mean” as a function of what other words they’re related to (but that’s still not grounding).

Lexical semantics

grab	1
throw	0.824
pull	0.818
knock	0.799
grabbing	0.789
steal	0.787
pulling	0.764
grabs	0.756
away	0.746
catch	0.74

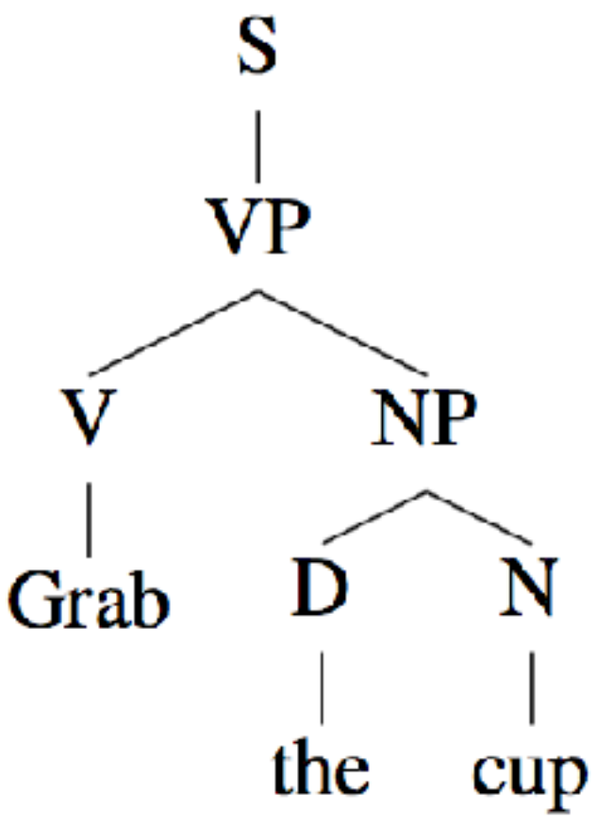


- “Grab” = execute GrabbingFunction()
- “the cup” = object ID 9AF1948A81CD22

Why is syntax insufficient?

- Even if we have a reference model for each word in a sentence, syntax doesn't tell us how those referents changes as a function of their compositionally.

- What is the meaning of a VP?



Semantics

Lexical semantics is concerned with representing the meaning of words (and their relations)

Logical semantics is concerned with representing the meaning of sentences.

Meaning representation

- A representation of the meaning of a sentence needs to bridge linguistic aspects of the sentence with non-linguistic knowledge about the world.

Following directions

Linguistic	Non-linguistic
Verbs like *grab*, *release*	Actions a robot can execute
Nouns like *cup*, *block*	Specific entities in the world
transitive VP (V NP)	An action executed upon an object

Question answering

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...

Was Barack Obama born in the United States?

Non-linguistic
Hawaii is a place
Hawaii is a place within the United States
“born in X” entails “from x”

[example due to J. Andreas]

Meaning representation

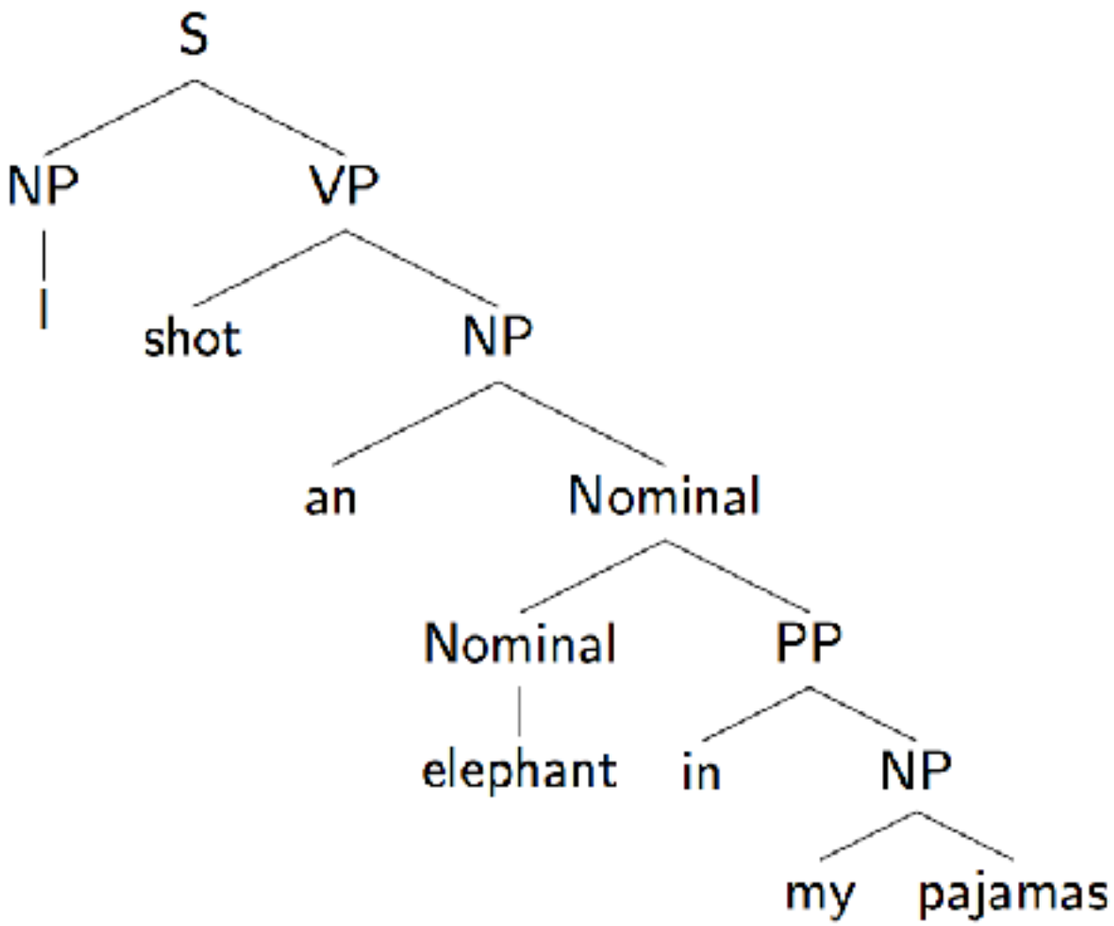
Geo	which state has the most rivers running through it?
	(argmax $0 (state:t $0) (count $1 (and (river:t $1) (loc:t $1 $0))))
ATIS	all flights from dallas before 10am
	(lambda $0 e (and (flight $0) (from $0 dallas:ci) (< (departure time $0) 1000:ti)))
SQL	What record company did conductor Mikhail Snitko record for after 1996?
	SELECT Record Company WHERE (Year of Recording > 1996) AND (Conductor = Mikhail Snitko)
Django	if length of bits is lesser than integer 3 or second element of bits is not equal to string 'as'
	if len(bits) < 3 or bits[1] != 'as':

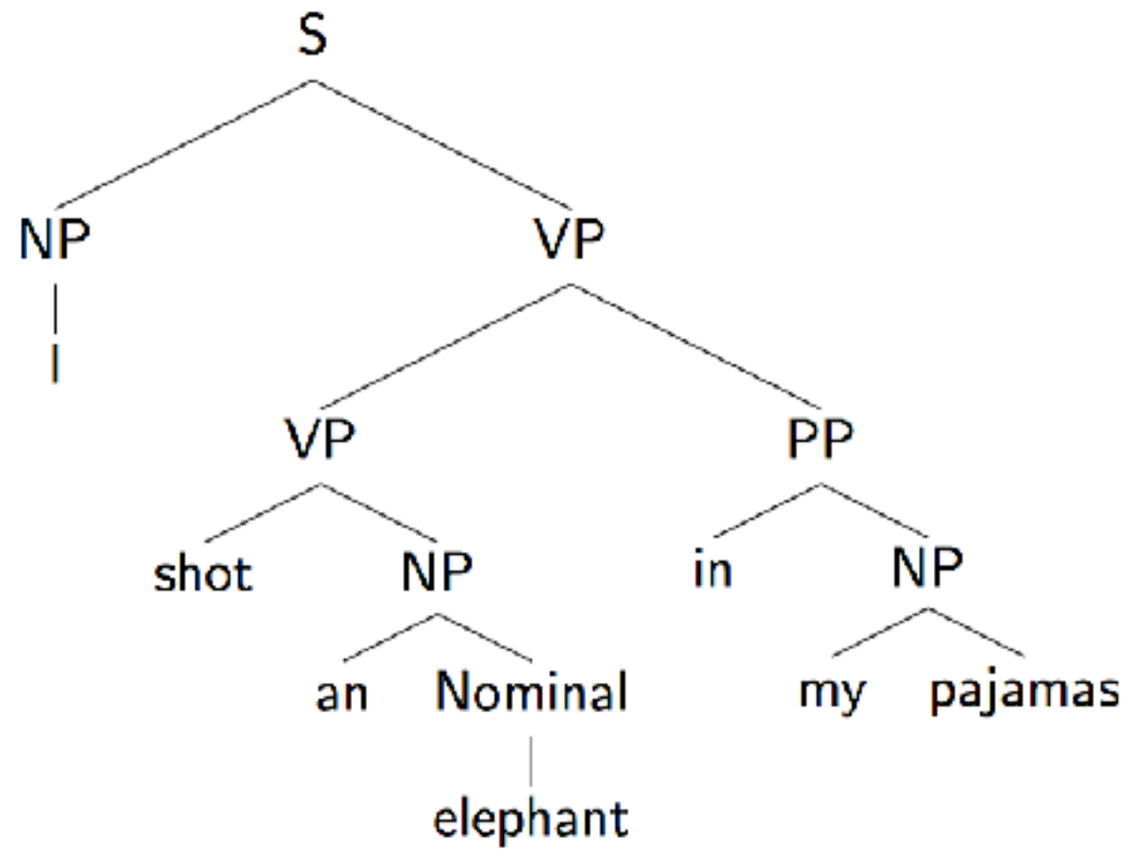
Dong and Lapata (2018), "Coarse-to-Fine Decoding for Neural Semantic Parsing"

Meaning representation

A meaning representation should be unambiguous; each statement in a meaning representation should have one meaning.

Meaning representation



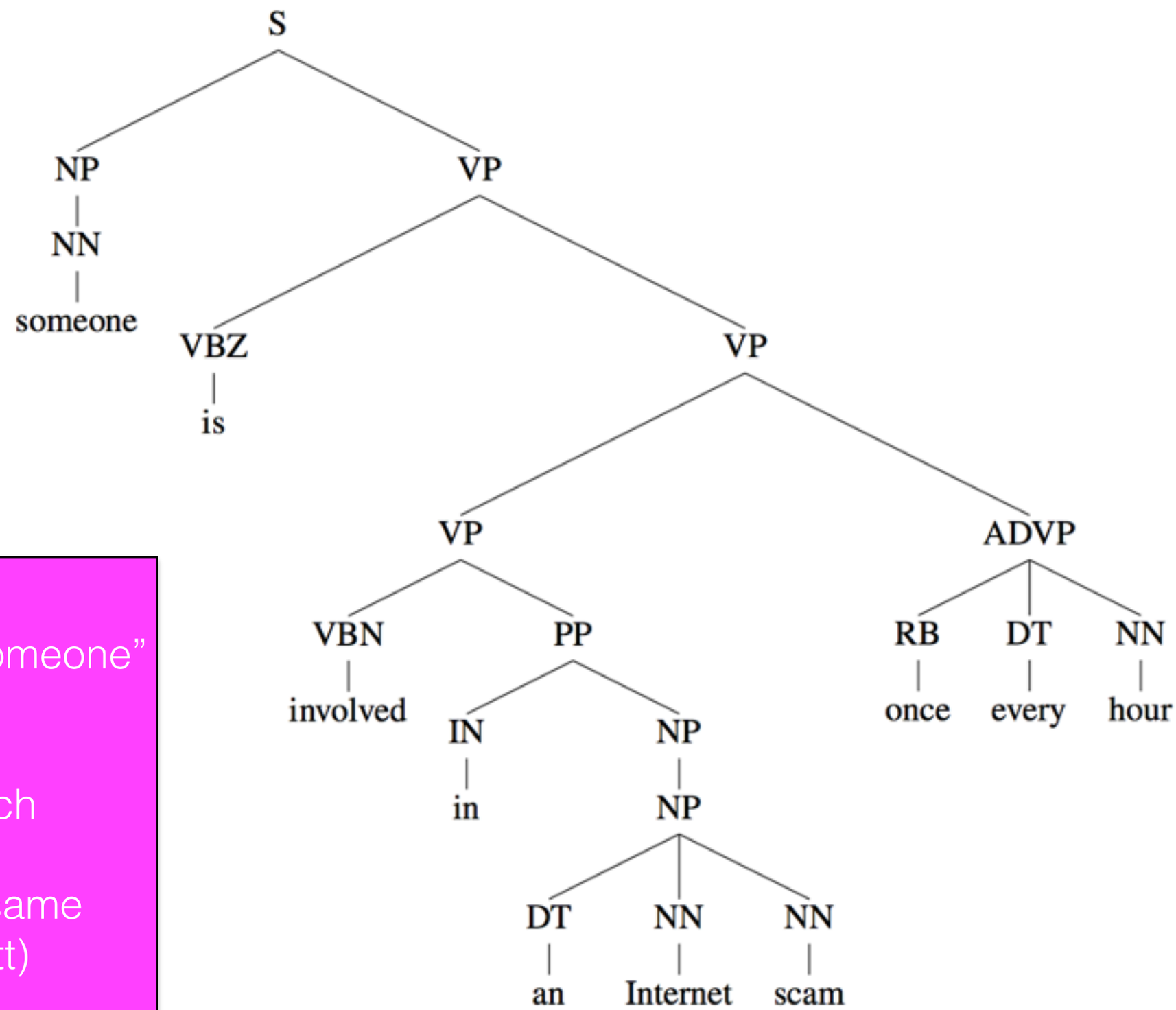


Syntax resolves some ambiguity

The Office (2005)

“Once every hour, someone is involved in an Internet scam”

“That person is Michael Scott”



Same structure for “someone” meaning:

- Each person for each scam
- One person in the same scam (Michael Scott)

First-order logic (FOL)

- We want to represent every sentence as an unambiguous proposition in FOL

Sentence	*Luke was fighting with Darth Vader*
FOL	FIGHT(LUKE, VADER)

FIGHT(LUKE, VADER)

This is a *relation*; we define what it means

These are *constants*; we know who they uniquely identify

WIKIDATA

Item Discussion

Star Wars Episode IV: A New Hope (Q17738)

director	George Lucas 11 references	edit + add value
screenwriter	George Lucas 1 reference	edit + add value
cast member	Harrison Ford character role: Han Solo 12 references	edit
	Alec Guinness character role: Obi-Wan Kenobi 8 references	edit
	Carrie Fisher character role: Princess Leia	edit

First-order logic (FOL)

- How we map a natural language sentence to FOL is the task of semantic parsing; but we define the FOL relations and entities to be sensitive to what matters in our model.

Sentence	*Luke battled Vader*
Sentence	*Luke fought with Vader*
Sentence	*Skywalker dueled with Darth Vader*
Sentence	*Luke was fighting with Darth Vader*
FOL	FIGHT(LUKE, VADER)

First-order logic (FOL)

- How we map a natural language sentence to FOL is the task of semantic parsing; but we define the FOL relations and entities to be sensitive to what matters in our world model.

Maybe in our star wars model we want to preserve the difference between *fighting* and *dueling*

Sentence	*Luke battled Vader*
Sentence	*Luke fought with Vader*
Sentence	*Luke was fighting with Darth Vader*
FOL	FIGHT(LUKE, VADER)

Sentence	*Skywalker dueled with Darth Vader*
FOL	DUEL(LUKE, VADER)

First-order logic (FOL)

- How we map a natural language sentence to FOL is the task of semantic parsing; but we define the FOL relations and entities to be sensitive to what matters in our world model.

For a robot model, there is only one possible “grabbing” kind of action, so subtle differences don’t matter.

Sentence	*Grab the cup*
Sentence	*Snatch the cup!*
Sentence	*Take the cup*
FOL	GRAB(ROBOT, CUP)

First-order logic (FOL)

- Constants
- Relations (including properties)
- Variables
- Quantifiers
- Functions
- Logical connectives

Constants

- Constants refer to specific entities in the world being described.
- Refer to exactly one object in that world.
- Dependent on the model; people, things, places, events, etc.

Constants

- DarthVader
- DarthMaul
- Emperor
- Luke
- HanSolo
- Leia
- Obi-Wan
- Chewbacca
- R2-D2
- C-3PO
- Yoda

- Luke_Lightsaber_1
- Luke_Lightsaber_2
- Vader_Lightsaber_1

- Tattoine
- Hoth
- Endor

the domain simply *enumerates* the objects; we know nothing else about them yet

Relations

- N-ary relations hold among FOL terms (constants, variables, functions).

Unary (property)	HUMAN(LUKE), ROBOT(C-3PO)
binary relation	FIGHTS(LUKE, VADER)
ternary	GIVES(OBI-WAN, LUKE, LUKE_LIGHTSABER_1)
…	*…*

Properties (unary relations)

- Informally, properties describe attributes of entities in the model (e.g. Luke being human).
- Formally, properties denote *sets of elements* in the domain

Properties

- [[HUMAN]] = {Darth Vader, Emperor, Luke, Han Solo, Leia, Obi-Wan}
- [[ROBOT]] = {R2-D2, C-3PO}

- [[LIGHTSABER]] = {Luke_Lightsaber_1, Luke_Lightsaber_2, Vader_Lightsaber_1}

- [[RED]] = {Vader_Lightsaber_1}
- [[BLUE]] = {Luke_Lightsaber_1}
- [[GREEN]] = {Luke_Lightsaber_2}

- [[PLANET]] = {Tattoine, Hoth, Endor}

Relations

- Relations denote sets of tuples in the domain.

Relations

- [[FIGHT]] = {<Darth Vader, Luke>, <Darth Vader, Obi-Wan>, <Luke, Emperor>}
- [[LIVES]] = {<Luke, Tattoine>}
- [[FATHER]] = {<Darth Vader, Luke>, <Darth Vader, Leia>}
- [[GIVES]] = {<Obi-Wan, Luke, Luke_Lightsaber_1>}

Extensional definition

- The denotation of RED is the set of objects that are lightsabers.
- The denotation of FIGHT is the set of pairs of entities that fight.

Logical connectives

- Formal means for composing expressions in a meaning representation language (symbols, quantifiers)

- Boolean operators on connectives yield known truth values

Negation	$\neg\phi$	*True if ϕ is false*
Conjunction	$\phi \wedge \psi$	*True if ϕ and ψ are both true*
Disjunction	$\phi \vee \psi$	*True if ϕ or ψ is true*
Implication	$\phi \Rightarrow \psi$	*True unless ϕ is true and ψ is false*
Equivalence	$\phi \Leftrightarrow \psi$	*True if ϕ and ψ are both true or both false*

$\mathcal{M}$ star_wars

- Darth Vader
- Emperor
- Luke
- Han Solo
- Leia
- Obi-Wan
- Chewbacca
- R2-D2
- C-3PO

- Luke_Lightsaber_1
- Luke_Lightsaber_2
- Vader_Lightsaber

- Tattoine
- Hoth
- Endor

- [[HUMAN]] = {Darth Vader, Emperor, Luke, Han Solo, Leia, Obi-Wan}
- [[ROBOT]] = {R2-D2, C-3PO}

- [[LIGHTSABER]] = {Luke_Lightsaber_1, Luke_Lightsaber_2, Vader_Lightsaber_1}

- [[RED]] = {Vader_Lightsaber_1}
- [[BLUE]] = {Luke_Lightsaber_1}
- [[GREEN]] = {Luke_Lightsaber_2}

- [[PLANET]] = {Tattoine, Hoth, Endor}

- [[FIGHT]] = {<Darth Vader, Luke>, <Darth Vader, Obi-Wan>, <Luke, Emperor>}

- [[LIVES]] = {<Luke, Tattoine>}

- [[FATHER]] = {<Darth Vader, Luke>, <Darth Vader, Leia>}

- [[GIVES]] {<Obi-Wan, Luke, Luke_Lightsaber_1>}

Meaning

- Model-theoretic semantics; the truth of a proposition is determined with respect to some model $\mathcal{M}$ of the world.

- Separate from verification (does not need to be true of the real world); what would the world need to be like for the sentence to be true?

$$[[\text{LUKE}]]_{\mathcal{M}}$$

$$[[\text{FIGHTS}]]_{\mathcal{M}}$$

We'll use [[*]] to describe the denotation of a term in a specific model $\mathcal{M}$.

Truth

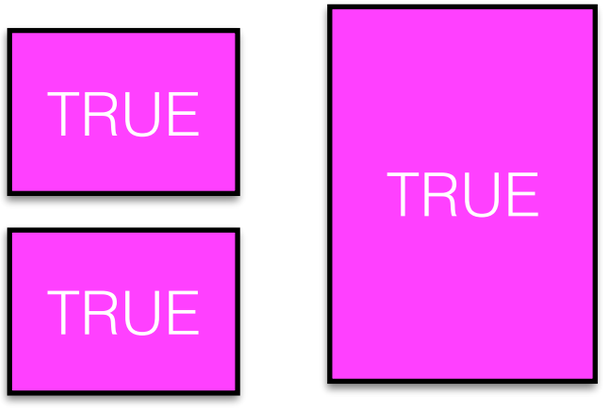
- The truth of a proposition under a model depends on whether the denotation of the constants is in the denotation of the relation.

Luke lives on Tattoine $([\![\text{Luke}]\!]_{\mathcal{M}}, [\![\text{Tattoine}]\!]_{\mathcal{M}}) \in [\![\textsc{Lives}]\!]_{\mathcal{M}}$ TRUE

Luke fights Han $([\![\text{Luke}]\!]_{\mathcal{M}}, [\![\text{Han}]\!]_{\mathcal{M}}) \in [\![\textsc{Fights}]\!]_{\mathcal{M}}$ FALSE

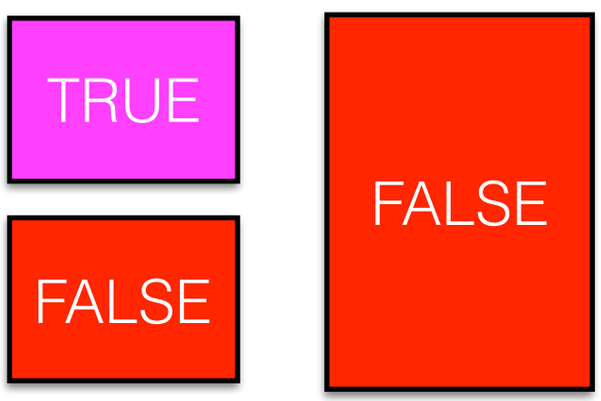
Luke lives on Tattoine and fights Darth Vader

$$([\![\text{Luke}]\!]_{\mathcal{M}}, [\![\text{Tattoine}]\!]_{\mathcal{M}}) \in [\![\textsc{Lives}]\!]_{\mathcal{M}}$$
$$\wedge$$
$$([\![\text{Luke}]\!]_{\mathcal{M}}, [\![\text{Darth Vader}]\!]_{\mathcal{M}}) \in [\![\textsc{Fights}]\!]_{\mathcal{M}}$$



Luke lives on Tattoine and fights Han

$$([\![\text{Luke}]\!]_{\mathcal{M}}, [\![\text{Tattoine}]\!]_{\mathcal{M}}) \in [\![\textsc{Lives}]\!]_{\mathcal{M}}$$
$$\wedge$$
$$([\![\text{Luke}]\!]_{\mathcal{M}}, [\![\text{Han}]\!]_{\mathcal{M}}) \in [\![\textsc{Fights}]\!]_{\mathcal{M}}$$



First-order logic

- This machinery lets us evaluate the truth conditions about specific entities and relations in $\mathcal{M}$.

First-order logic

Does C-3PO fight anyone?

- Does C-3PO fight Luke?
- Does C-3PO fight Han?
- Does C-3PO fight Darth Vader?
- Does C-3PO fight Leia?
- Does C-3PO fight R2D2?
- Does C-3PO fight the Emperor?

First-order logic

C-3PO fights someone

Variable	fights(C-3PO, x)	x is a free variable with no committed assignment
Quantifier	∃x fights(C-3PO, x)	x is now bound and has a truth value in $\mathcal{M}$

First-order logic

$\exists x.\text{fights}(\text{C-3PO}, x)$

$\forall x.\text{fights}(\text{C-3PO}, x)$

Semantic parsing

- How do we get from a natural language statement to a proposition whose truth content we can evaluate in a model?

"Luke fights Han"

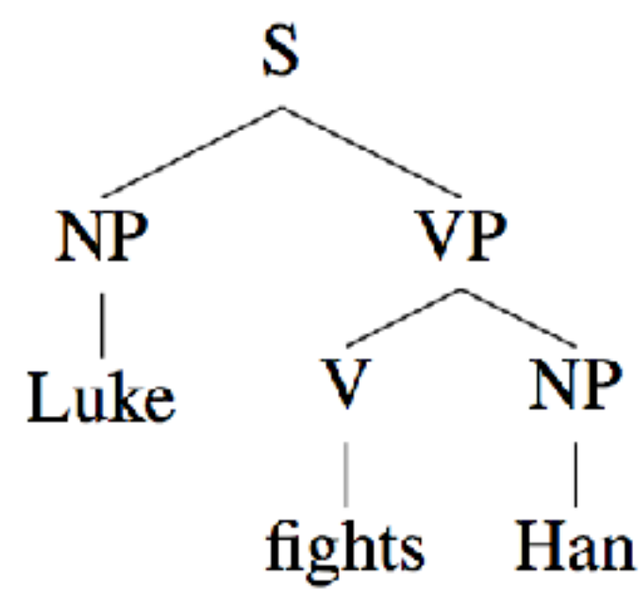
$$([\![\text{Luke}]\!]_{\mathcal{M}}, [\![\text{Han}]\!]_{\mathcal{M}}) \in [\![\textsc{Fights}]\!]_{\mathcal{M}}$$

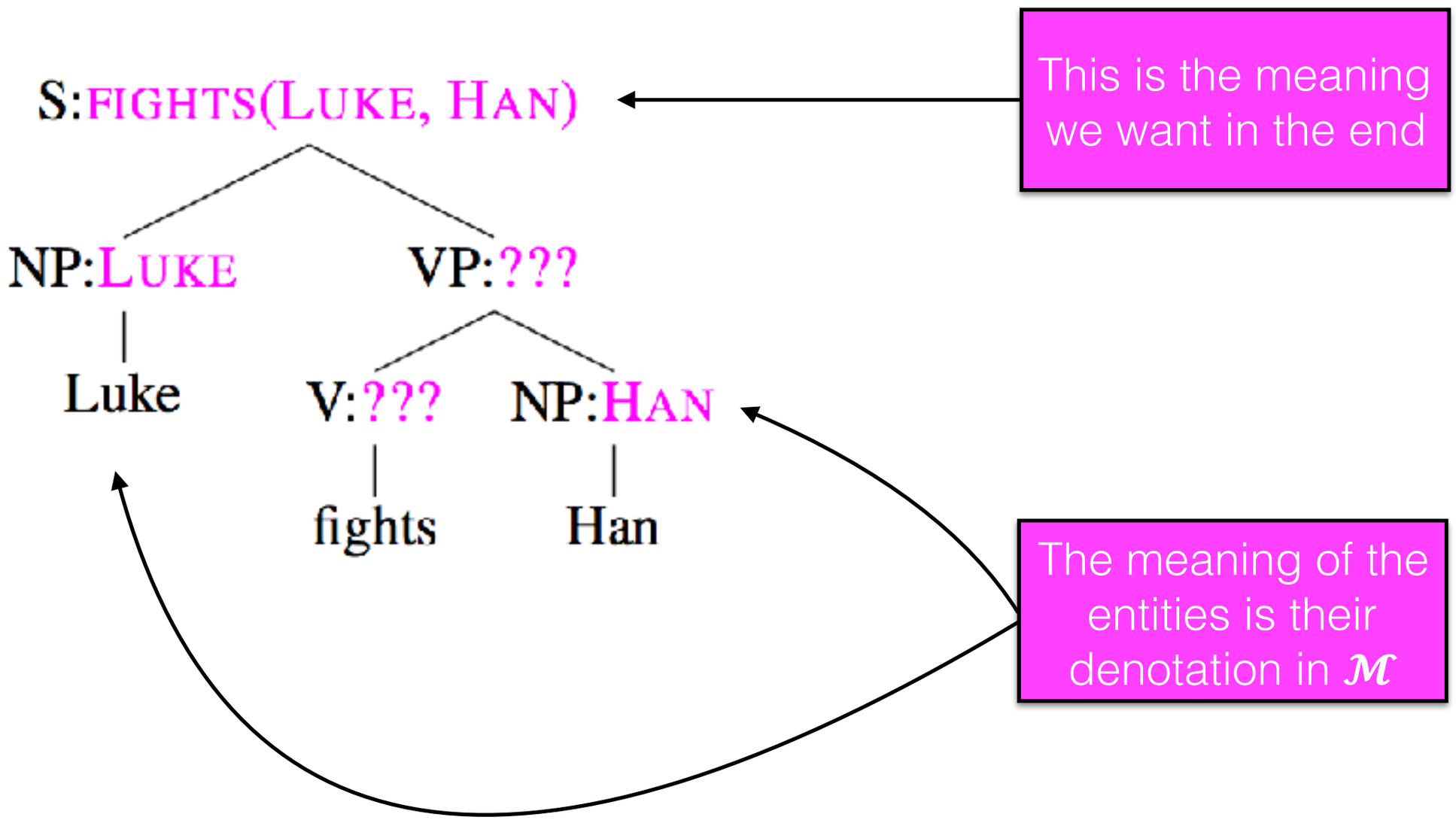
Compositionality

- Principle of compositionally: the meaning of a complex expression is function of the meaning of its constituent parts (Frege).

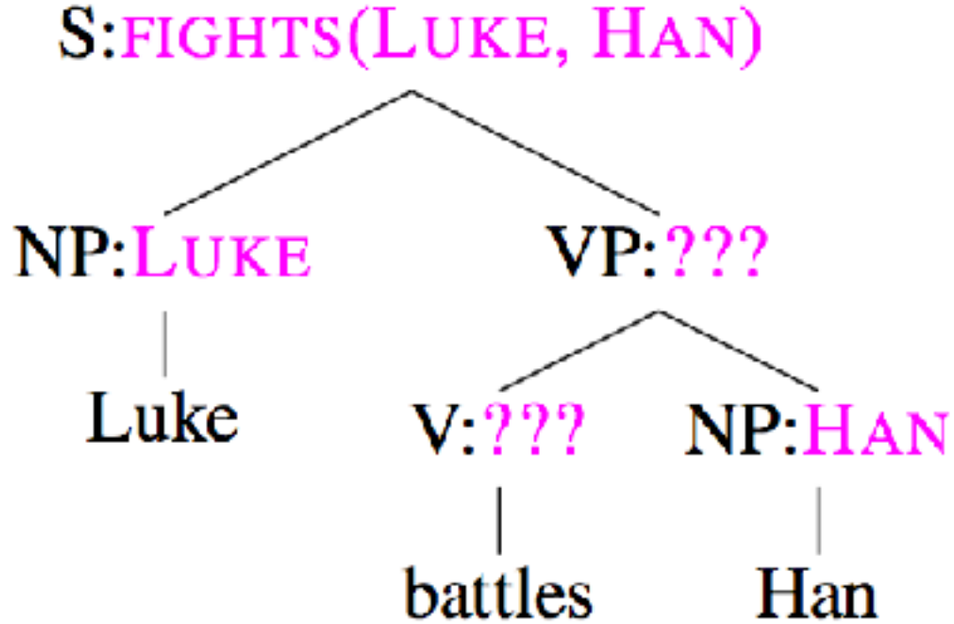
- Constituent parts here = syntactic constituents

S.meaning = function(NP.meaning, VP.meaning)



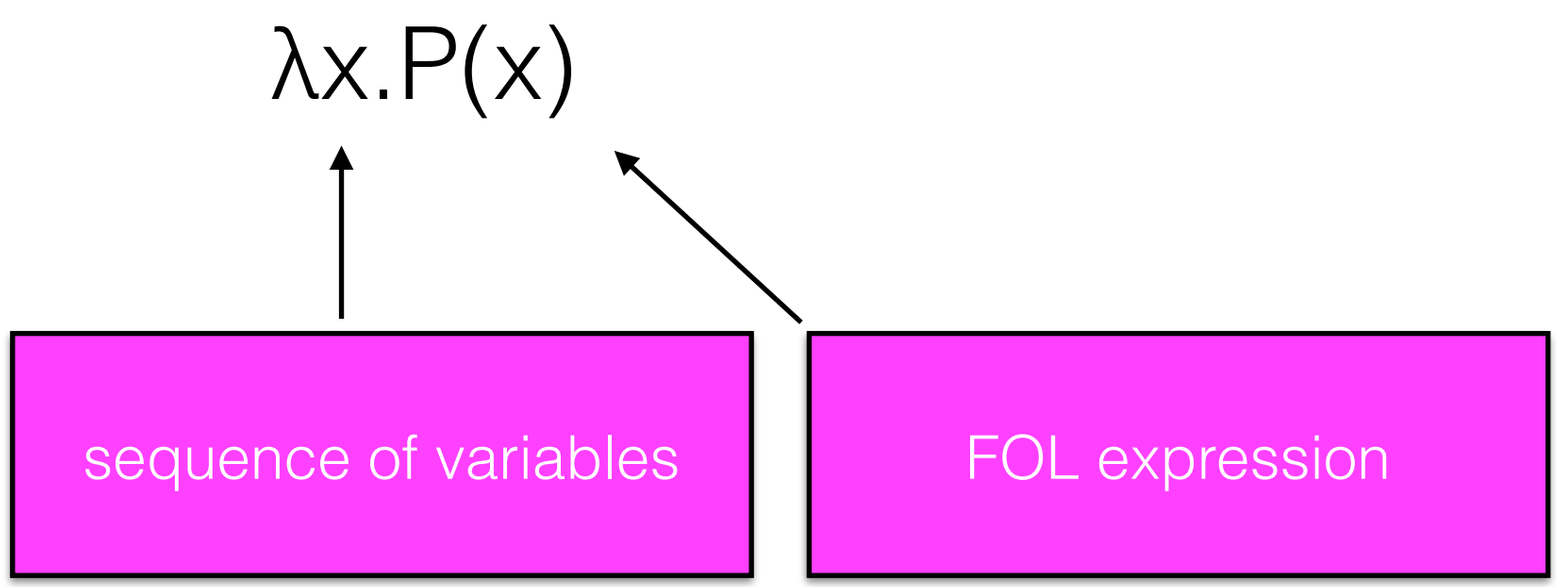

S:FIGHTS(LUKE, HAN)
This is the meaning we want in the end
NP:LUKE
VP:???
Luke
V:???
NP:HAN
fights
Han
The meaning of the entities is their denotation in $\mathcal{M}$

How do we represent the meaning of this partial structure?



Lambda calculus

- Lambda expressions: descriptions of anonymous functions.



Lambda calculus

- Lambda expressions: descriptions of anonymous functions.

$$\lambda x.P(x)$$

- A lambda expression is a function that can be applied to FOL terms as arguments to yield new FOL expressions where the variables are bound to the argument terms

Lambda calculus

- Lambda expressions: descriptions of anonymous functions.

$$\lambda x.P(x)(A)$$

Argument

Yields:

$$P(A)$$

Lambda calculus

- Lambda expression can take multiple arguments (where subsequent reduction yields new lambda expressions)

$$\lambda x.\lambda y.P(x, y)$$

Lambda calculus

$\lambda y.\lambda x.P(x, y)(A)$

Yields:

$\lambda x.P(x, A)$

$\lambda x.P(x, A)(B)$

Yields:

$P(B, A)$

Lambda calculus

The state of one thing being near something else

$\lambda y.\lambda x.\textsc{Near}(x, y)$

The state of one thing being near Cafe Strada

$\lambda y.\lambda x.\textsc{Near}(x, y)(\textsc{CafeStrada})$

$\lambda x.\textsc{Near}(x, \textsc{CafeStrada})$

$\lambda x.\textsc{Near}(x, \textsc{CafeStrada})(\text{Boalt})$

Fully specified FOL formula: Boalt is near Cafe Strada

$\textsc{Near}(\text{Boalt}, \textsc{CafeStrada})$

Lambda calculus

Order matters!

$\lambda y.\lambda x.\text{LOVES}(x, y)(\text{HAN})$

$\lambda x.\lambda y.\text{LOVES}(x, y)(\text{HAN})$

Currying

- Converting a predicate with multiple arguments into a sequence of single-argument predicates

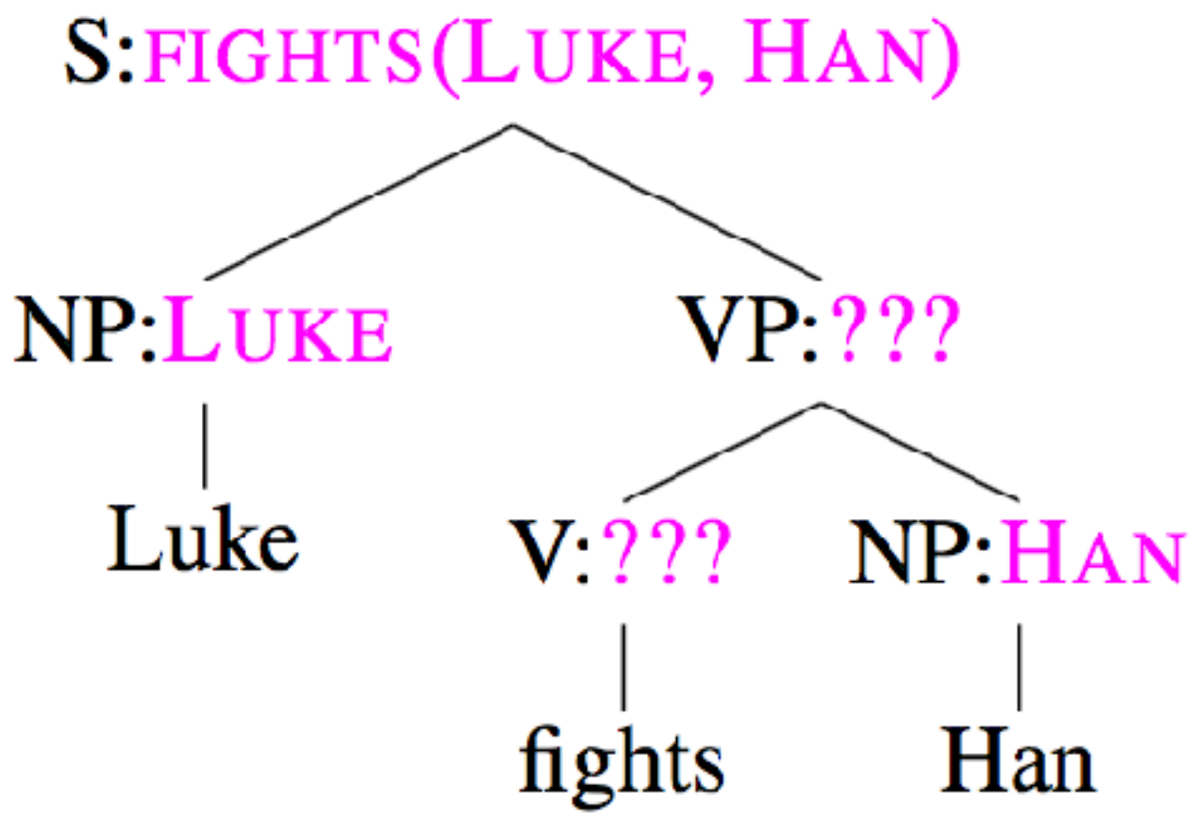
$\lambda z.\lambda y.\lambda x.\textsc{Give}(x, y, z)$

$\lambda z.\lambda y.\lambda x.\textsc{Give}(x, y, z)(A)$ — One argument (A)

$\lambda y.\lambda x.\textsc{Give}(x, y, A)(B)$ — One argument (B)

$\lambda x.\textsc{Give}(x, B, A)(C)$ — One argument (C)

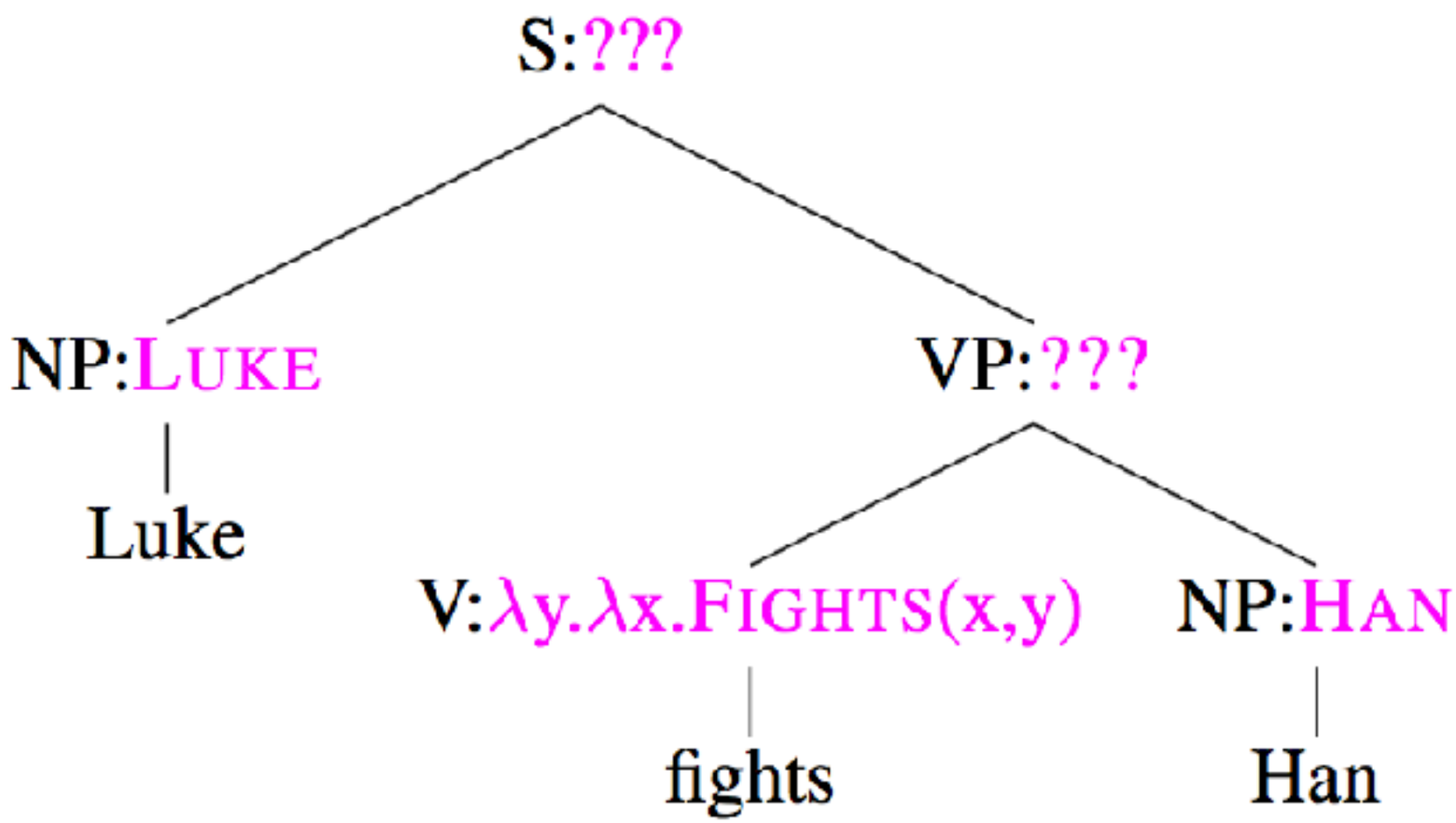
$\textsc{Give}(C, B, A)$


S:FIGHTS(LUKE, HAN)
NP:LUKE
VP:???
Luke
V:???
NP:HAN
fights
Han

Lambda calculus

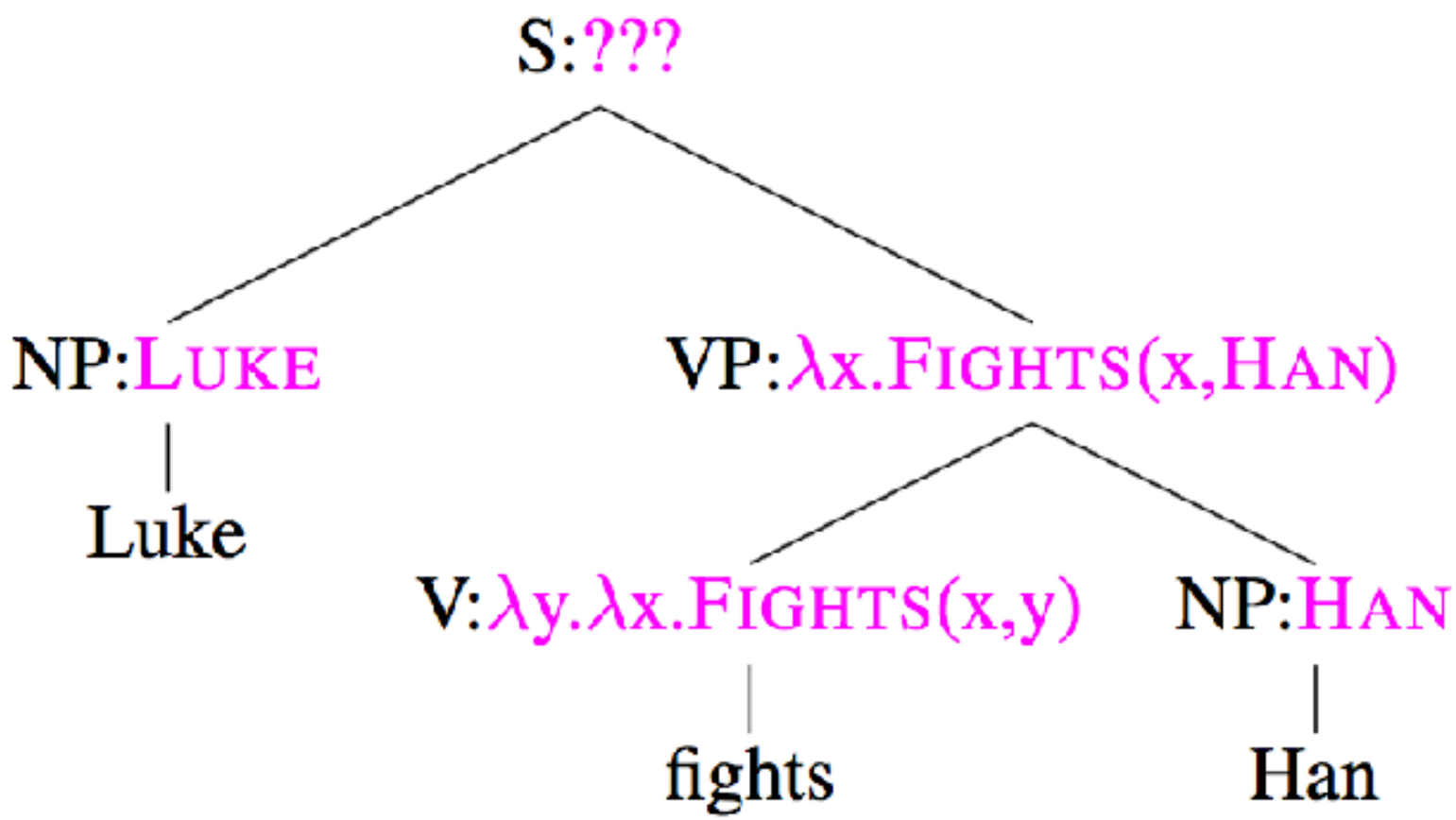
- With lambda calculus, pieces of logical forms correspond to pieces of linguistic structure.
- We can solve our problem by pairing each terminal with a entity (Han) or a lambda expression
- Fights is a verb that expects subject argument (the fighter) and an object argument (the thing fought).

fights $\lambda y.\lambda x.\textsc{Fights}(x, y)$



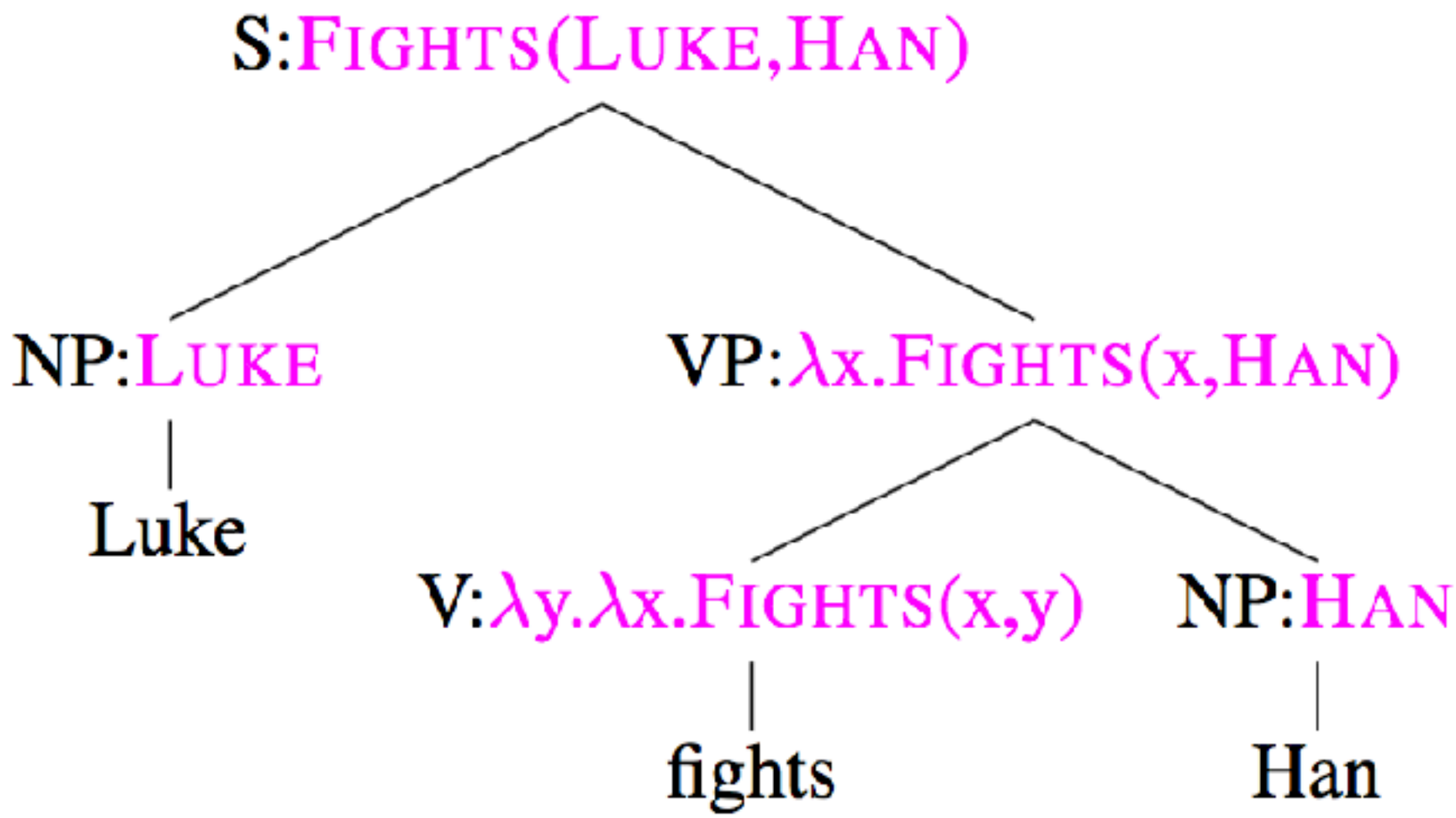
Here we can let naturally the NP to the right be the argument to this lambda expression to yield a new lambda expression

$\lambda y.\lambda x.\text{FIGHTS}(x, y)(\text{HAN}) \rightarrow \lambda x.\text{FIGHTS}(x, \text{HAN})$



Here we can state the NP to the left should be the argument to this lambda expression to yield a new lambda expression

λx.FIGHTS(x, HAN)(LUKE) → FIGHTS(LUKE, HAN)


S:FIGHTS(LUKE,HAN)
NP:LUKE
VP:λx.FIGHTS(x,HAN)
Luke
V:λy.λx.FIGHTS(x,y)
NP:HAN
fights
Han

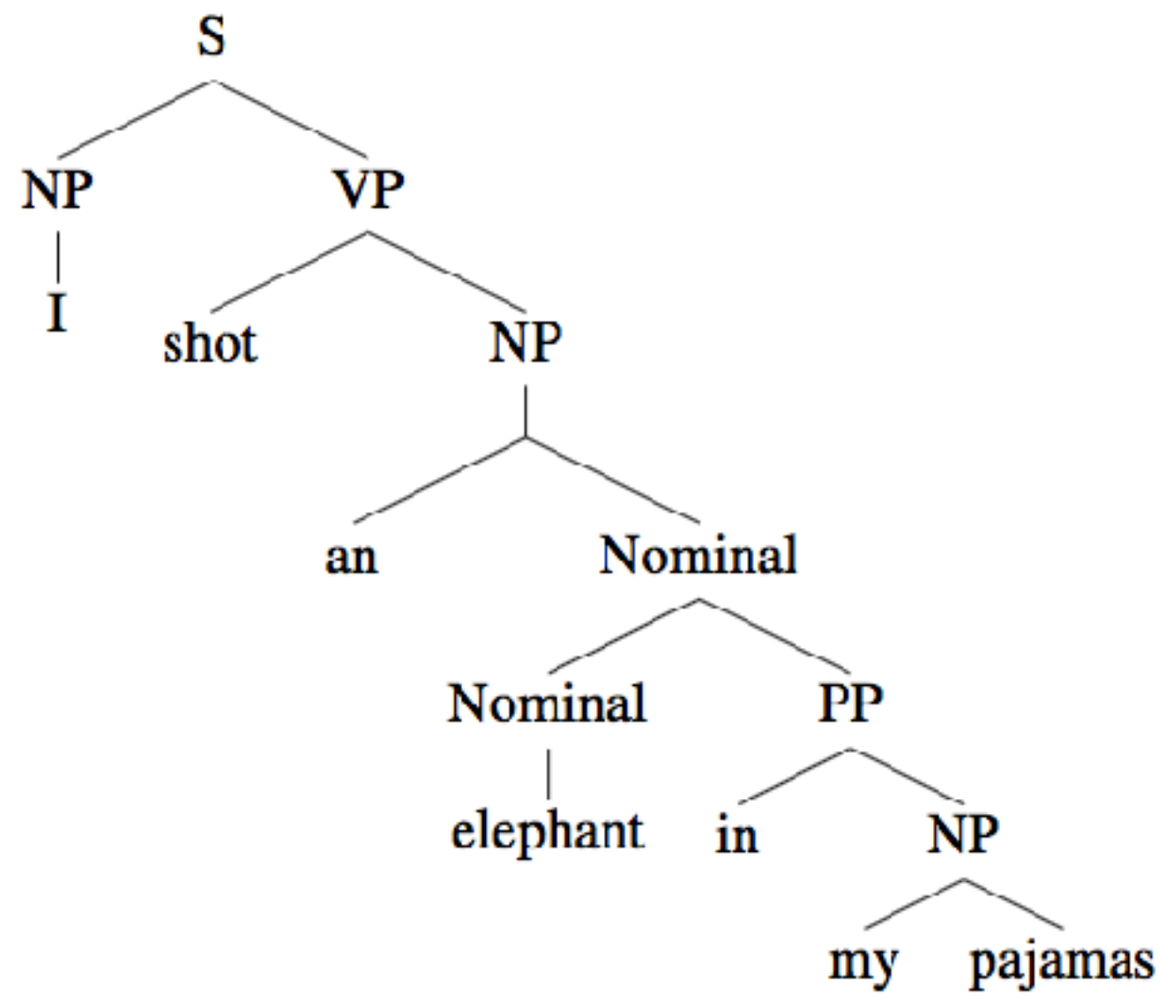
S → NP VP	VP.sem@NP.sem
VP → V NP	V.sem@NP.sem
V → fights	λy.λx.fights(x,y)
V → battles	λy.λx.fights(x,y)
NP → Han	Han
NP → Luke	Luke

In general, we can let the non-terminals determine how the semantics of their constituents are combined

Syntax

- We could represent the relationship between syntax and semantics in a CFG.

- But what we want is fine-grained control over the mapping between words and semantic primitives.



Tuesday

- Semantic parsing