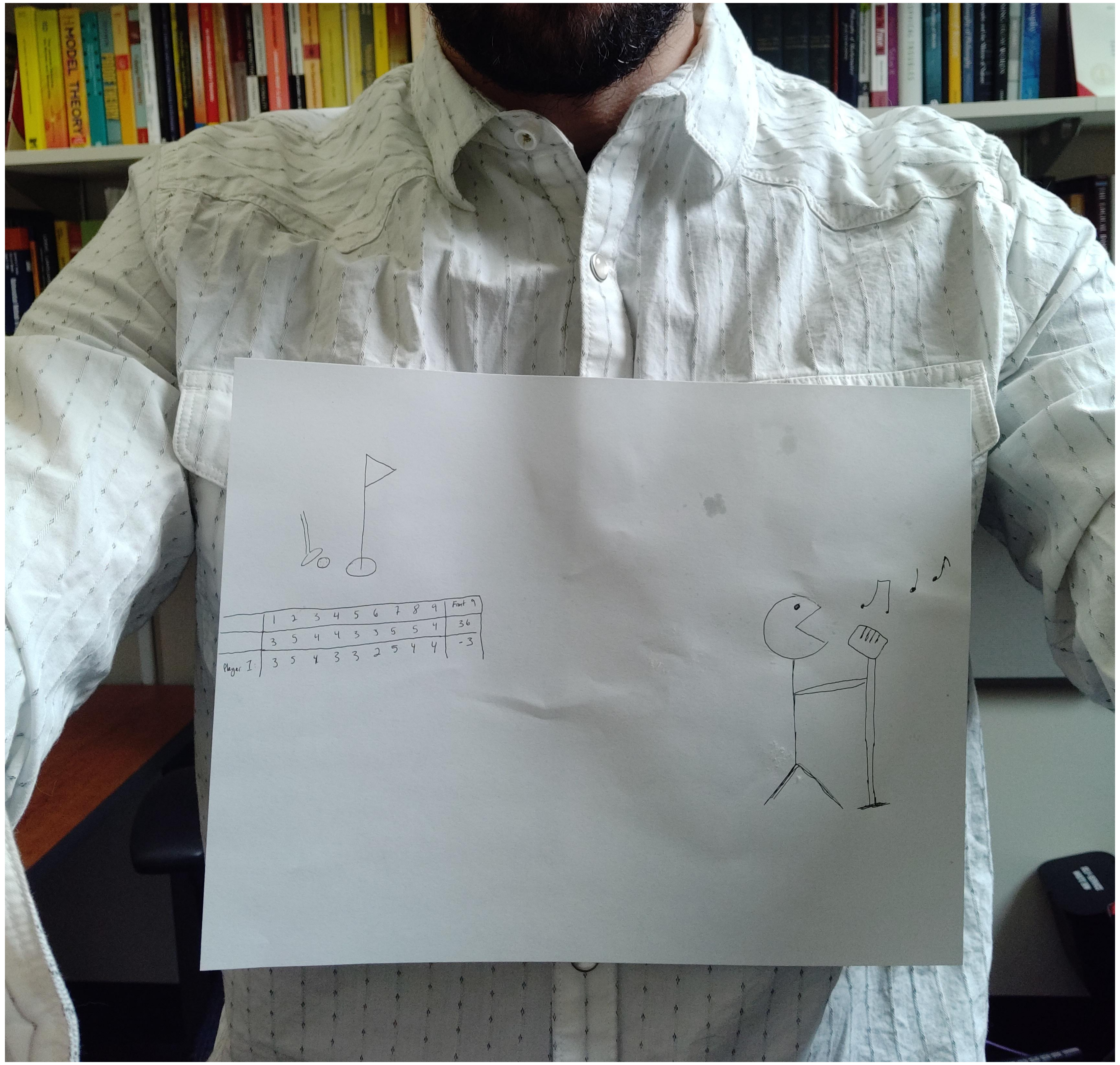
More λ-Calculus
Lexical Semantics

LING 571 — Deep Processing Techniques for NLP

Shane Steinert-Threlkeld

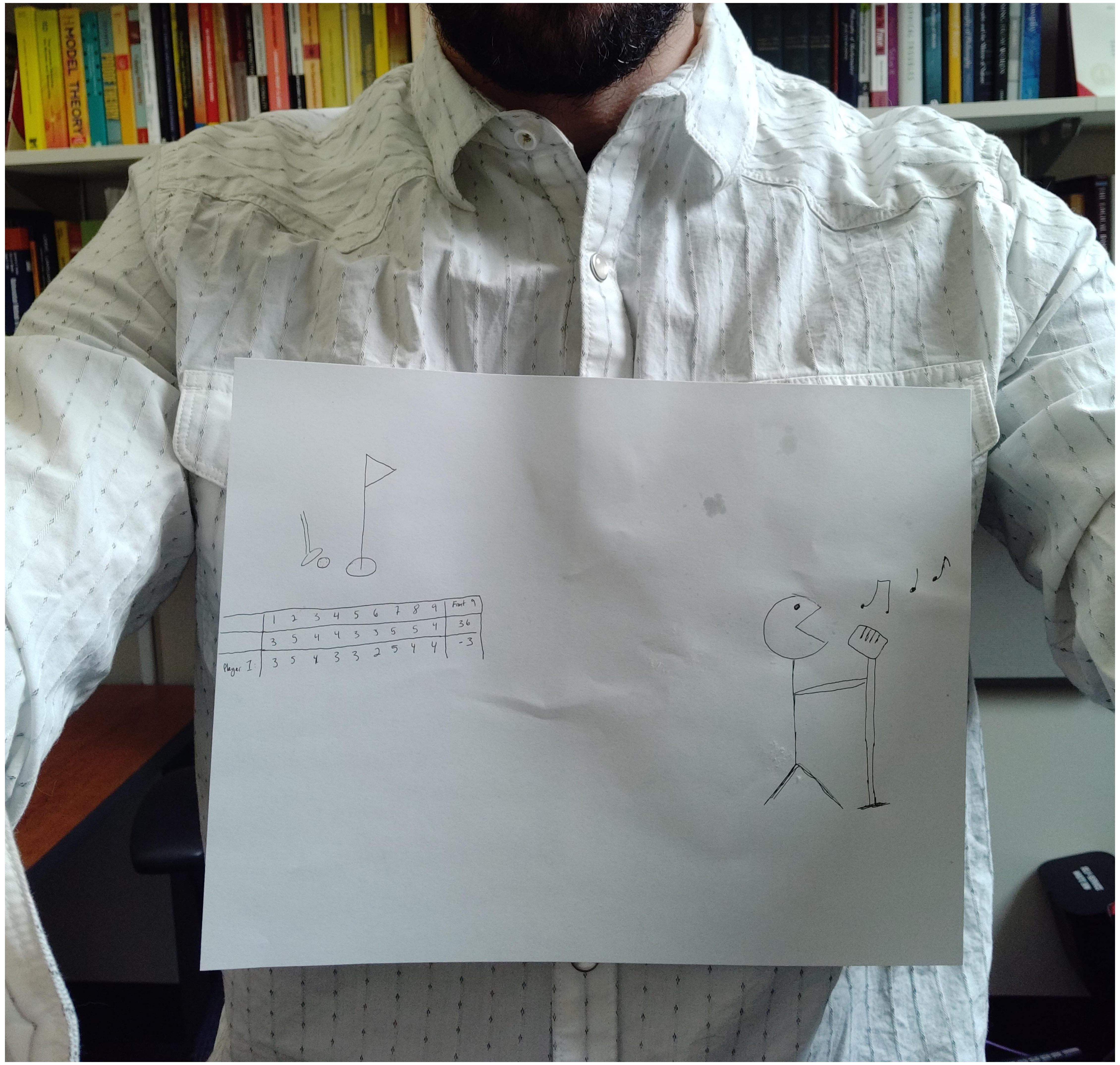
Happy Halloween!

2021:



Happy Halloween!

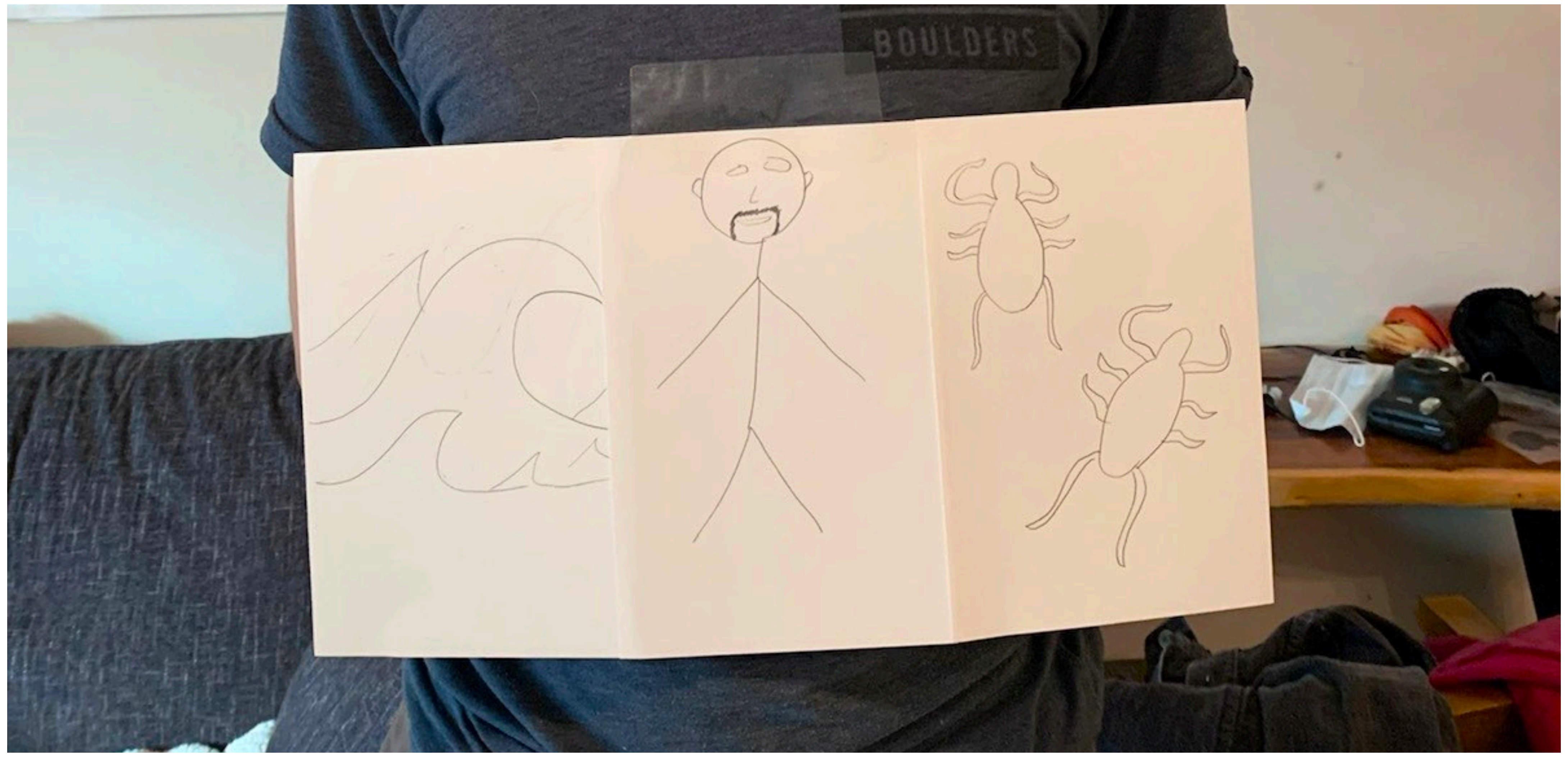
2021:



par + sing = parsing

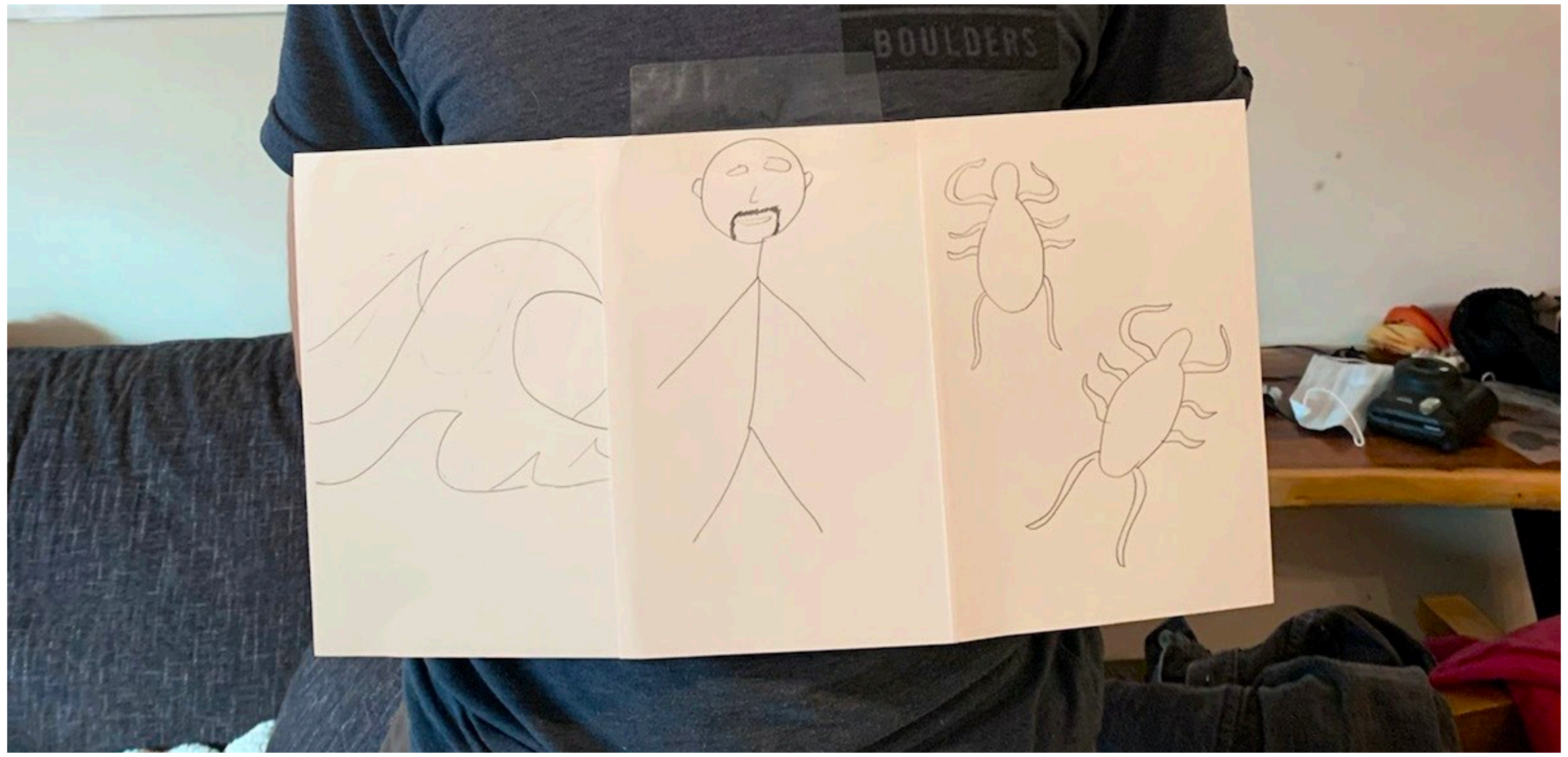
Happy Halloween!

2020:



Happy Halloween!

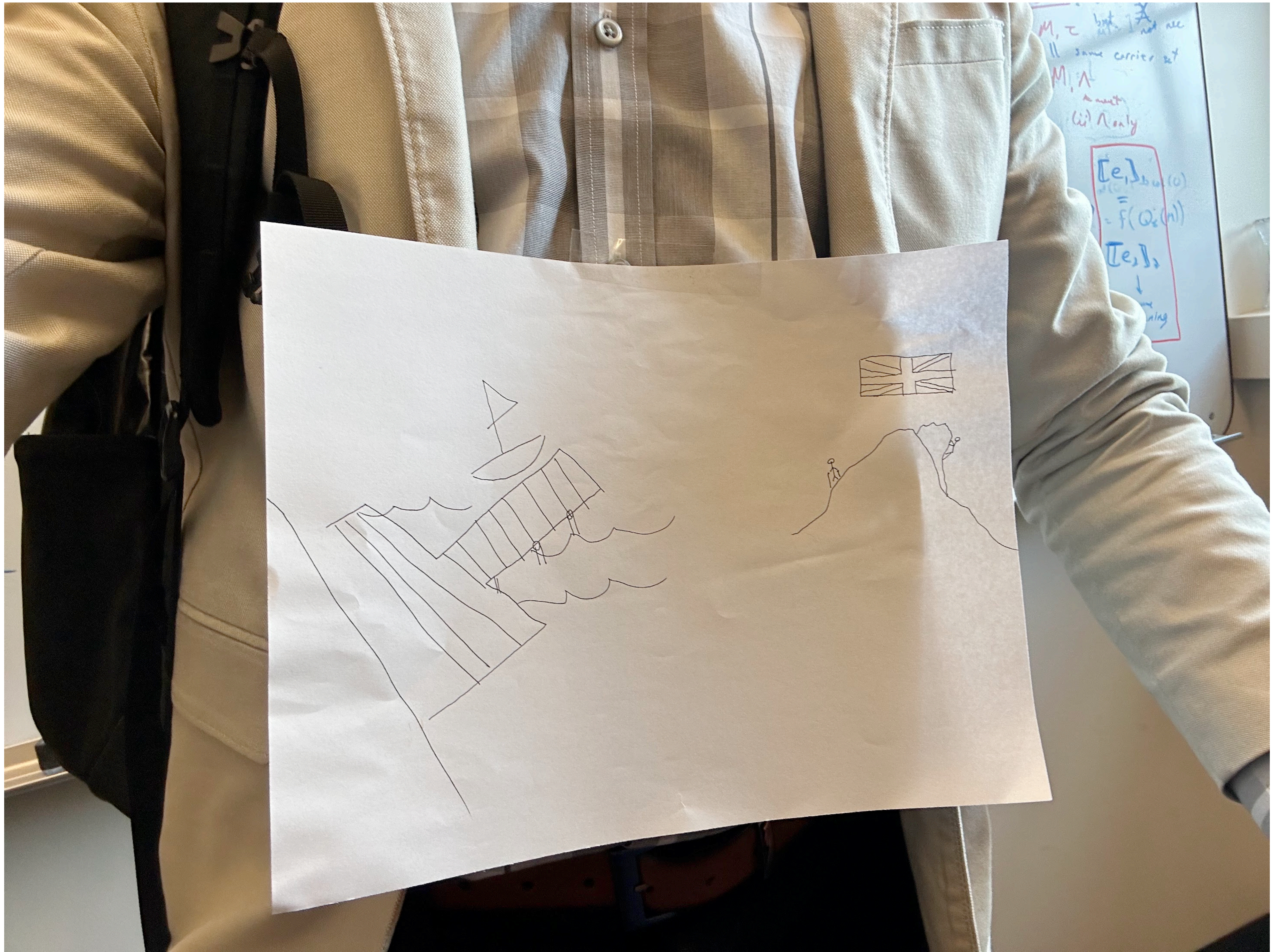
2020:



Sea + Man + Ticks = Semantics

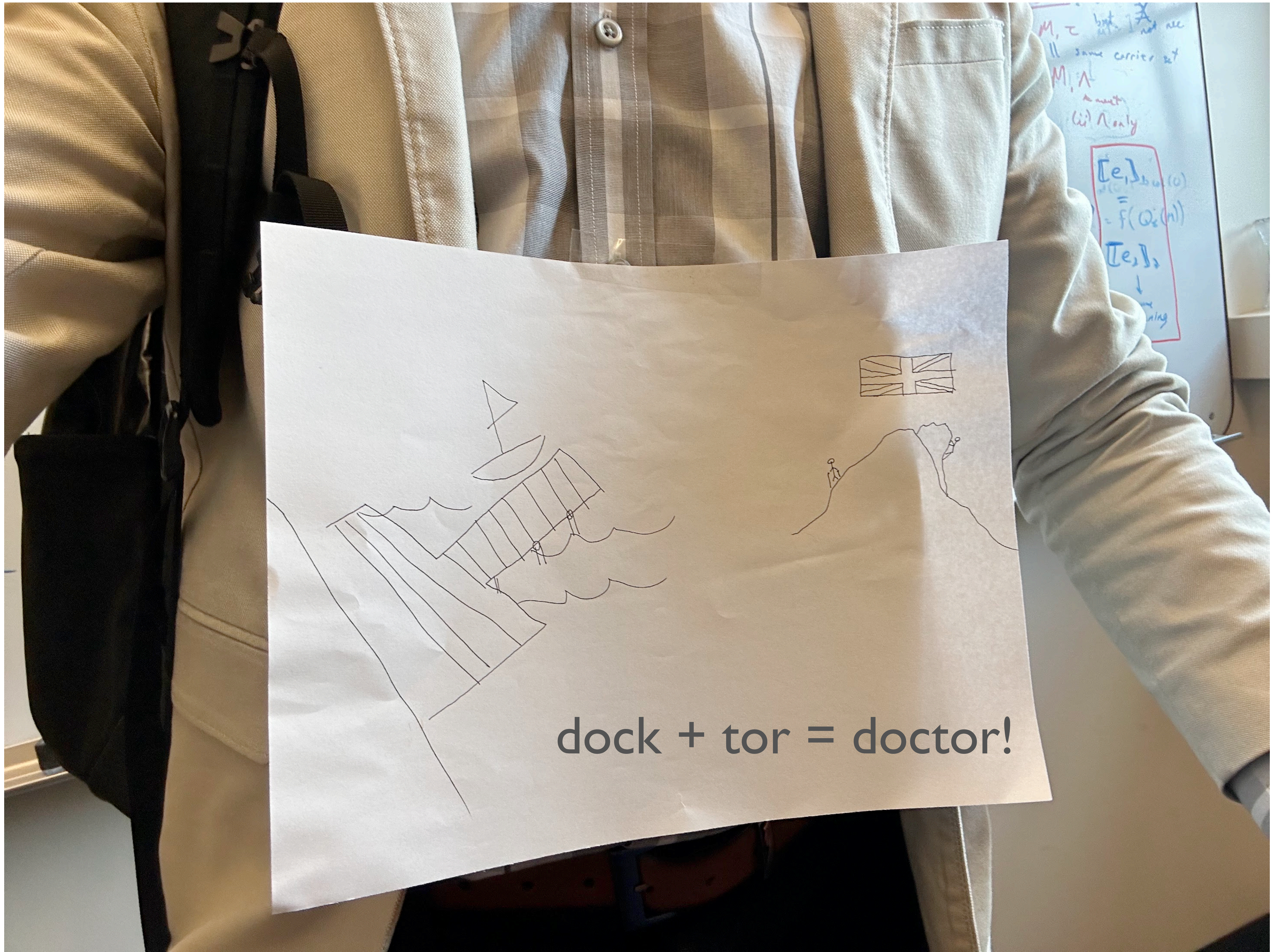
Happy Halloween!

2023: ???



Happy Halloween!

2023: ???



Roadmap

- FOL Semantics
- More Lambdas
- Learning Semantic Parsers
- Lexical Semantics
  - Motivation & Definitions
  - Word Senses
  - Tasks:
    - Word sense disambiguation
    - Word sense similarity
  - **Distributional Similarity**

NP → ***Det.sem*(*NP.sem*)**

S

$\{NP.sem(VP.sem)\}$

NP

$\{Det.sem(NP.sem)\}$

VP

Det

$\{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$

Noun

$\{\lambda y.Flight(y)\}$

V

$\{\lambda z.\exists e Arrived(e) \wedge ArrivedThing(e, z)\}$

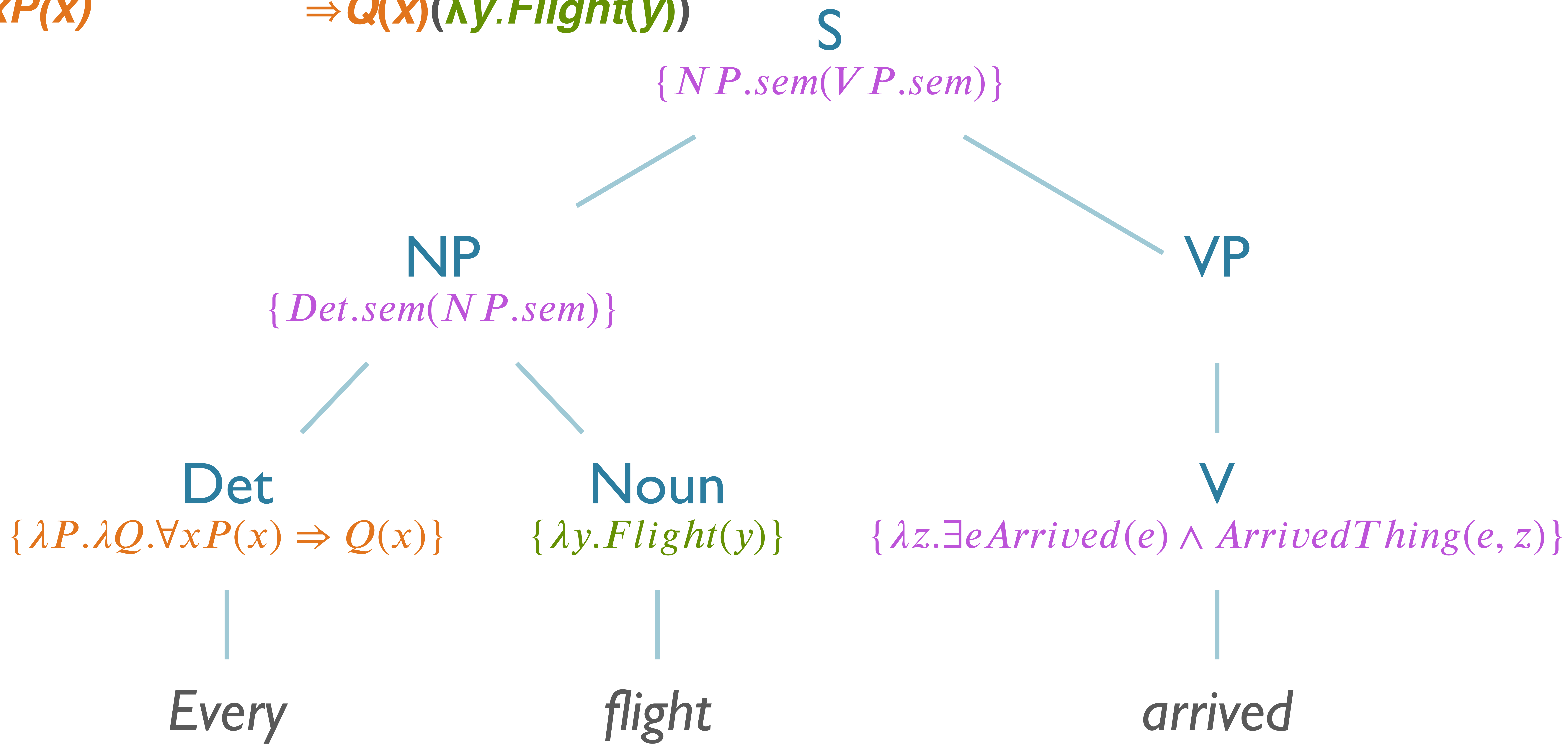
Every

flight

arrived

$NP \rightarrow Det.sem(NP.sem)$

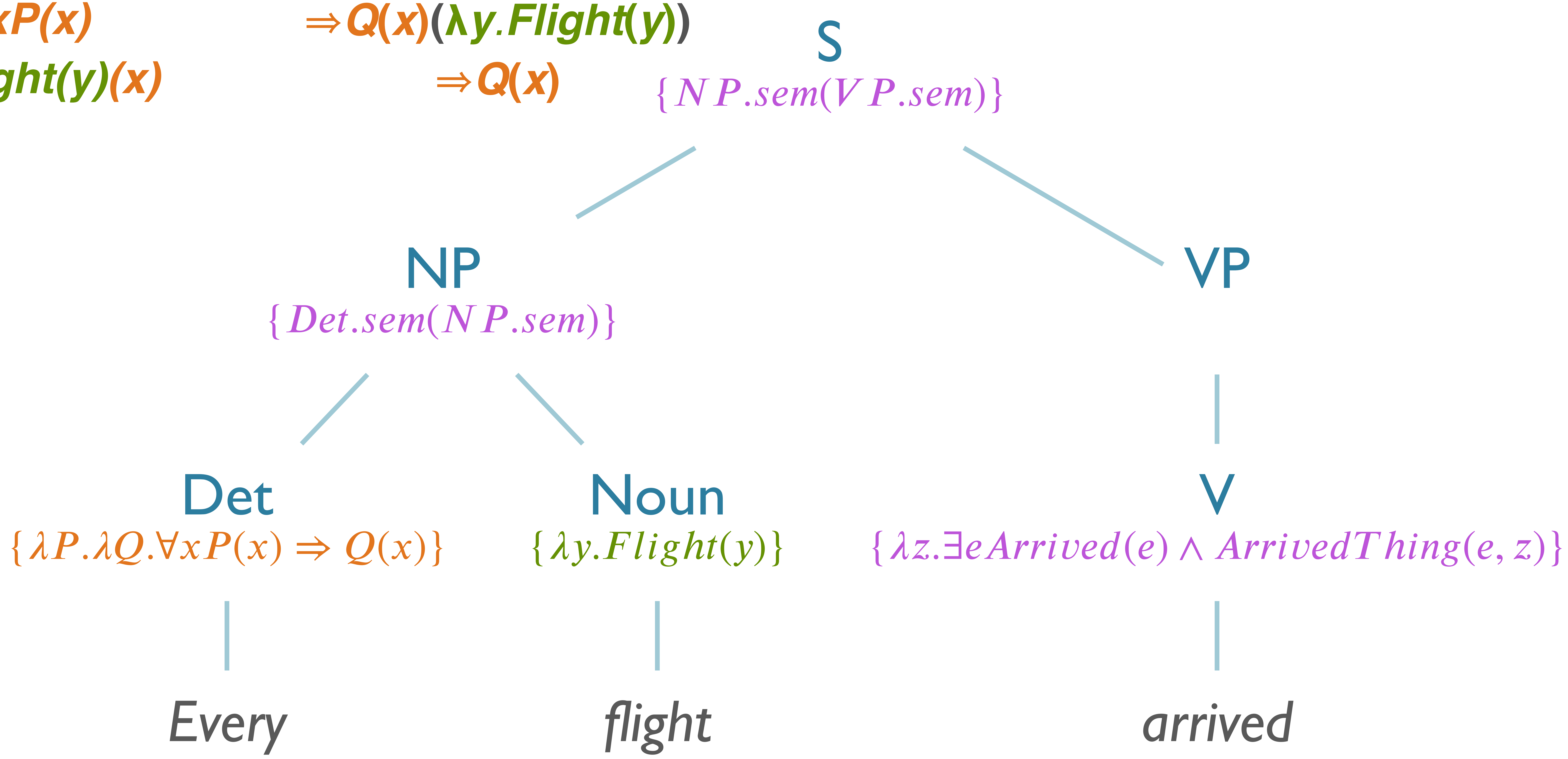
$\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)(\lambda y.Flight(y))$



$$NP \rightarrow Det.sem(NP.sem)$$

$$\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)(\lambda y.Flight(y))$$

$$\lambda Q.\forall x \lambda y.Flight(y)(x) \Rightarrow Q(x)$$

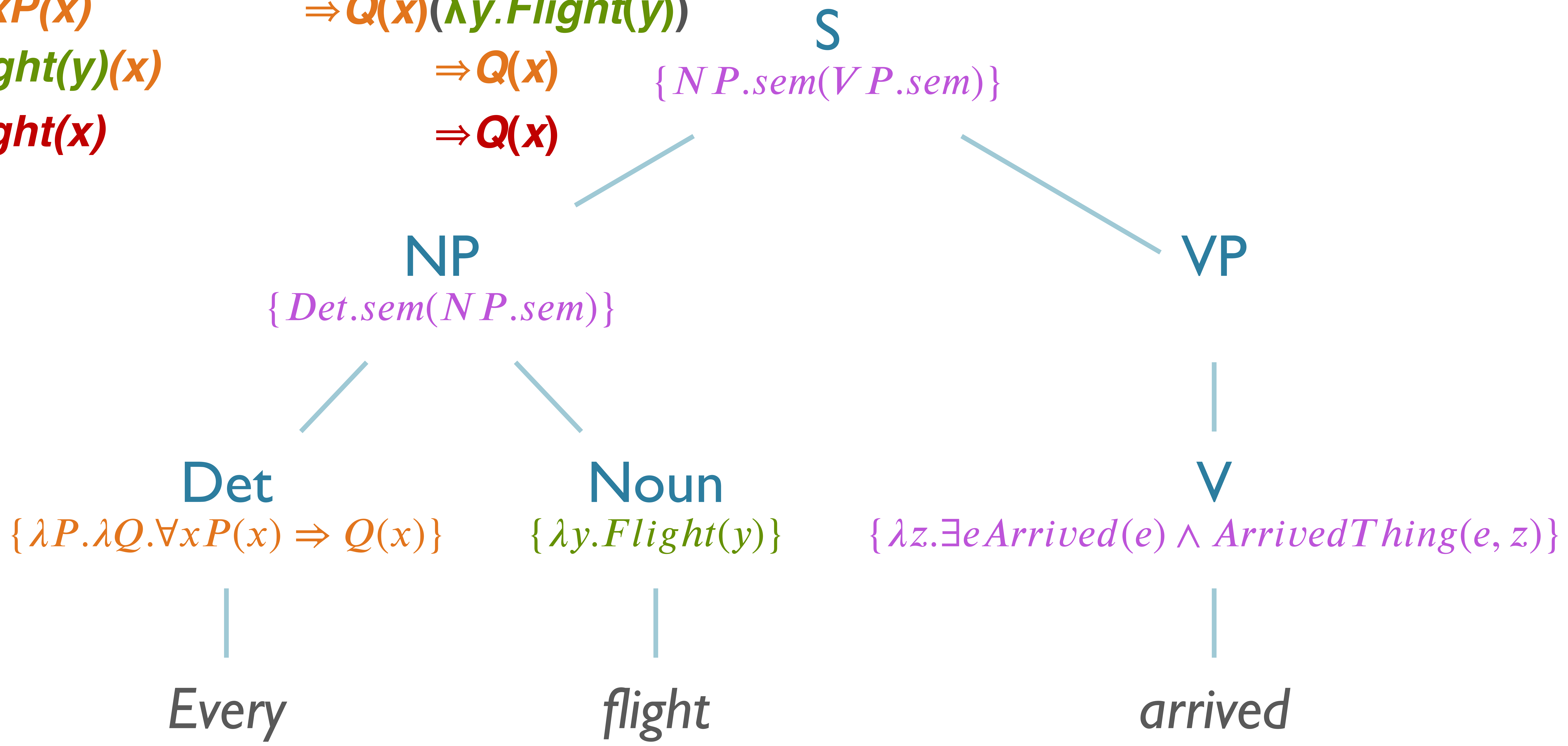


$$NP \rightarrow Det.sem(NP.sem)$$

$$\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)(\lambda y.Flight(y))$$

$$\lambda Q.\forall x \lambda y.Flight(y)(x) \Rightarrow Q(x)$$

$$\lambda Q.\forall x Flight(x) \Rightarrow Q(x)$$

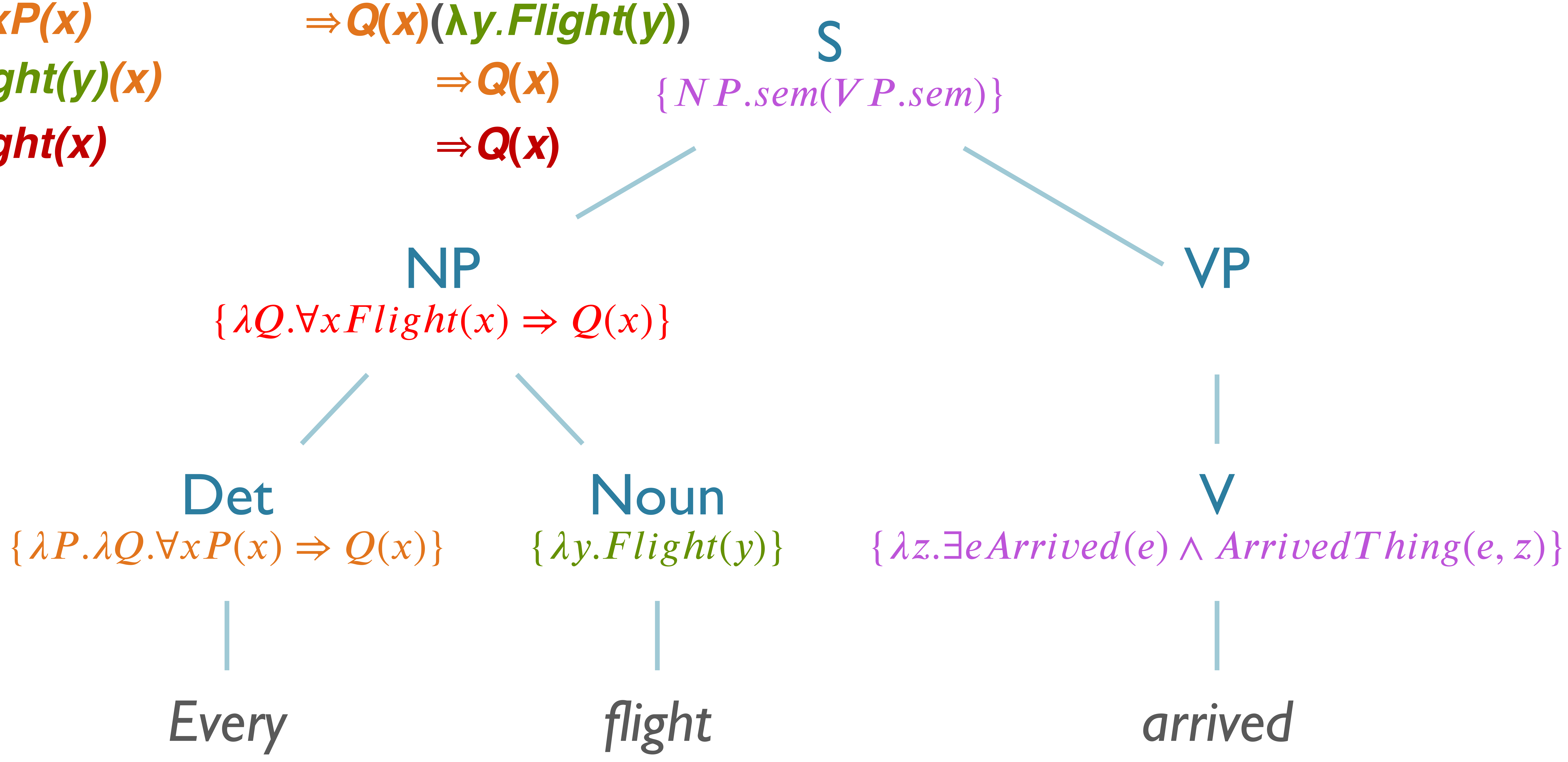


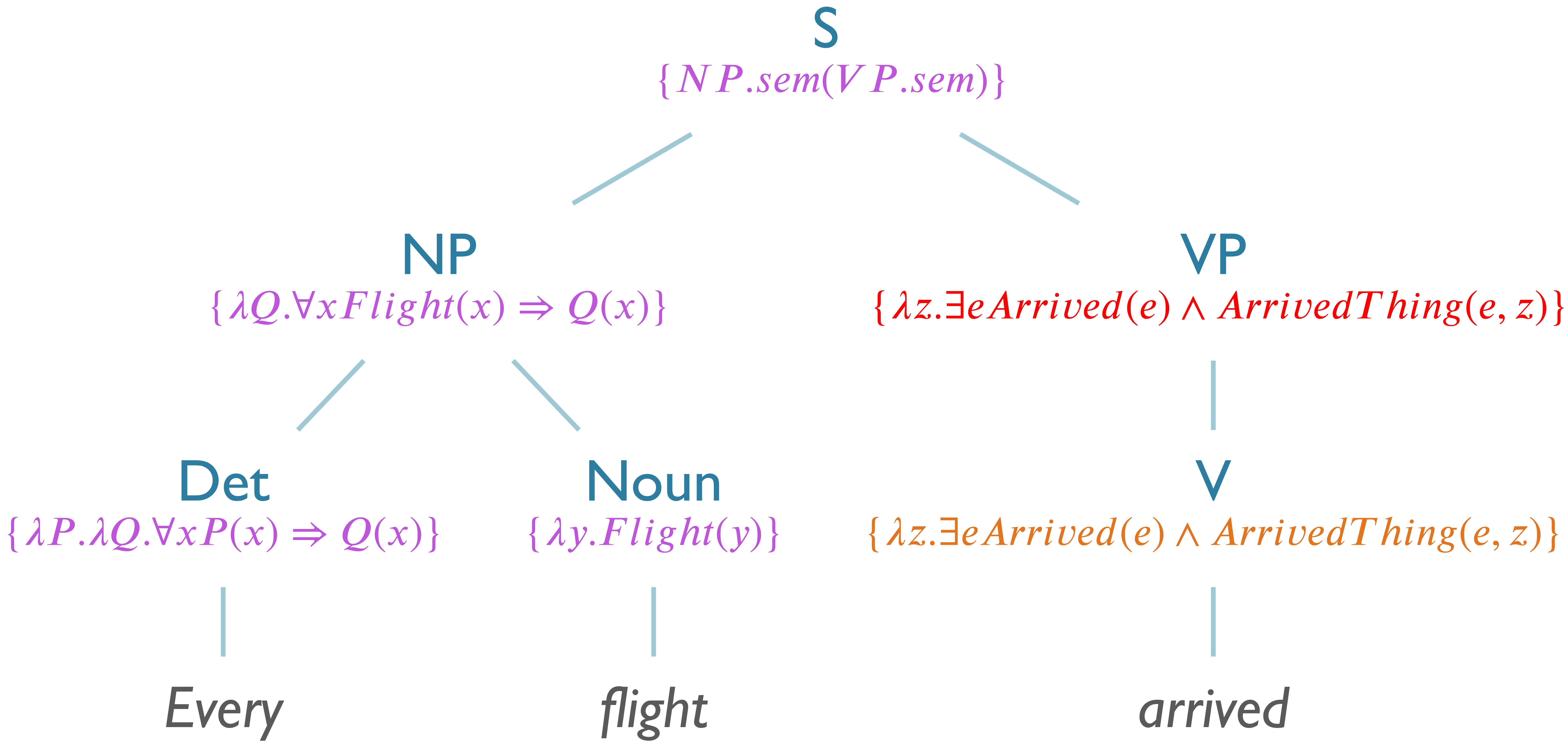
$$NP \rightarrow Det.sem(NP.sem)$$

$$\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)(\lambda y.Flight(y))$$

$$\lambda Q.\forall x \lambda y.Flight(y)(x) \Rightarrow Q(x)$$

$$\lambda Q.\forall x Flight(x) \Rightarrow Q(x)$$




S
$\{NP.sem(VP.sem)\}$
NP
$\{\lambda Q.\forall x Flight(x) \Rightarrow Q(x)\}$
VP
$\{\lambda z.\exists e Arrived(e) \wedge ArrivedThing(e,z)\}$
Det
$\{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$
Noun
$\{\lambda y.Flight(y)\}$
V
$\{\lambda z.\exists e Arrived(e) \wedge ArrivedThing(e,z)\}$
Every
flight
arrived

S
$\{NP.sem(VP.sem)\}$

NP
$\{\lambda Q.\forall x Flight(x) \Rightarrow Q(x)\}$

VP
$\{\lambda z.\exists e Arrived(e) \wedge ArrivedThing(e, z)\}$

Det
$\{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$

Noun
$\{\lambda y.Flight(y)\}$

V
$\{\lambda z.\exists e Arrived(e) \wedge ArrivedThing(e, z)\}$

Every *flight* *arrived*

S

$\{NP.sem(VP.sem)\}$

NP

$\{\lambda Q.\forall x Flight(x) \Rightarrow Q(x)\}$

VP

$\{\lambda z.\exists e Arrived(e) \wedge ArrivedThing(e, z)\}$

S

$$\{\forall x Flight(x) \Rightarrow \exists e Arrived(e) \wedge ArrivedThing(e, x)\}$$

NP

$$\{\lambda Q.\forall x Flight(x) \Rightarrow Q(x)\}$$

VP

$$\{\lambda z.\exists e Arrived(e) \wedge ArrivedThing(e, z)\}$$

S

$$\{\forall x Flight(x) \Rightarrow \exists e Arrived(e) \wedge ArrivedThing(e, x)\}$$

NP

$$\{\lambda Q.\forall x Flight(x) \Rightarrow Q(x)\}$$

VP

$$\{\lambda z.\exists e Arrived(e) \wedge ArrivedThing(e, z)\}$$

$$\lambda Q.\forall x Flight(x) \quad \Rightarrow Q(x)(\lambda z.\exists e Arrived(e) \wedge ArrivedThing(e, z))$$

S

$$\{\forall x Flight(x) \Rightarrow \exists e Arrived(e) \wedge ArrivedThing(e,x)\}$$

NP

$$\{\lambda Q.\forall x Flight(x) \Rightarrow Q(x)\}$$

VP

$$\{\lambda z.\exists e Arrived(e) \wedge ArrivedThing(e,z)\}$$

$$\lambda Q.\forall x Flight(x) \Rightarrow Q(x)(\lambda z.\exists e Arrived(e) \wedge ArrivedThing(e, z))$$

$$\forall x Flight(x) \Rightarrow \lambda z.\exists e Arrived(e) \wedge ArrivedThing(e, z)(x)$$

S

$$\{\forall x Flight(x) \Rightarrow \exists e Arrived(e) \land ArrivedThing(e,x)\}$$

NP

$$\{\lambda Q.\forall x Flight(x) \Rightarrow Q(x)\}$$

VP

$$\{\lambda z.\exists e Arrived(e) \land ArrivedThing(e,z)\}$$

$$\lambda Q.\forall x Flight(x) \Rightarrow Q(x)(\lambda z.\exists e Arrived(e) \land ArrivedThing(e,z))$$

$$\forall x Flight(x) \Rightarrow \lambda z.\exists e Arrived(e) \land ArrivedThing(e,z)(x)$$

$$\forall x Flight(x) \Rightarrow \exists e Arrived(e) \land ArrivedThing(e,x)$$

S
$\{\forall x Flight(x) \Rightarrow \exists e Arrived(e) \wedge ArrivedThing(e, x)\}$

NP
$\{\lambda Q.\forall x Flight(x) \Rightarrow Q(x)\}$

VP
$\{\lambda y.\exists e Arrived(e) \wedge ArrivedThing(e, y)\}$

Det
$\{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$

Noun
$\{\lambda x.Flight(x)\}$

V
$\{\lambda y.\exists e Arrived(e) \wedge ArrivedThing(e, y)\}$

Every

flight

arrived

More λ-Calculus

Common Nouns

- Noun -> ‘restaurant’ **{λ*x*.*Restaurant*(*x*)}**
  - Somewhat similar to the NNP construction
    - *λ var.Predicate(var)*

Common Nouns

- Noun -> 'restaurant' $\{\lambda x.Restaurant(x)\}$
  - Somewhat similar to the NNP construction
    - λ *var.Predicate(var)*
- But common nouns represent *properties*, rather than ***constants***
  - Meaning of the noun encoded in the predicate
  - Relate the concept of the noun to a particular instance of variable

Negation

- "No vegetarian restaurant serves meat."
  - $\neg(\exists x\, \mathit{VegetarianRestaurant}(x) \wedge \mathit{Serves}(x, \mathit{Meat}))$

Negation

- "No vegetarian restaurant serves meat."
  - $\neg(\exists x\, VegetarianRestaurant(x) \wedge Serves(x, Meat))$
- "All vegetarian restaurants do not serve meat."
  - $\forall x\ VegetarianRestaurant(x) \Rightarrow \neg Serves(x, Meat)$

Negation

- "No vegetarian restaurant serves meat."
  - $\neg(\exists x\, VegetarianRestaurant(x) \wedge Serves(x, Meat))$
- "All vegetarian restaurants do not serve meat."
  - $\forall x\ VegetarianRestaurant(x) \Rightarrow \neg Serves(x, Meat)$

Negation

- "No vegetarian restaurant serves meat."
  - $\neg(\exists x\, VegetarianRestaurant(x) \wedge Serves(x, Meat))$
- "All vegetarian restaurants do not serve meat."
  - $\forall x\ VegetarianRestaurant(x) \Rightarrow \neg Serves(x, Meat)$

- *These are semantically equivalent!*
  - [IF P, THEN ¬Q] ⇔ ¬[P AND Q]
  - $\neg\exists x P(x) \Leftrightarrow \forall x \neg P(x)$

Negation

- "No vegetarian restaurant serves meat."
  - $\neg(\exists x\, VegetarianRestaurant(x) \wedge Serves(x, Meat))$
- "All vegetarian restaurants do not serve meat."
  - $\forall x\ VegetarianRestaurant(x) \Rightarrow \neg Serves(x, Meat)$

- *These are semantically equivalent!*
  - [IF P, THEN ¬Q] ⇔ ¬[P AND Q]
  - $\neg\exists x P(x) \Leftrightarrow \forall x \neg P(x)$

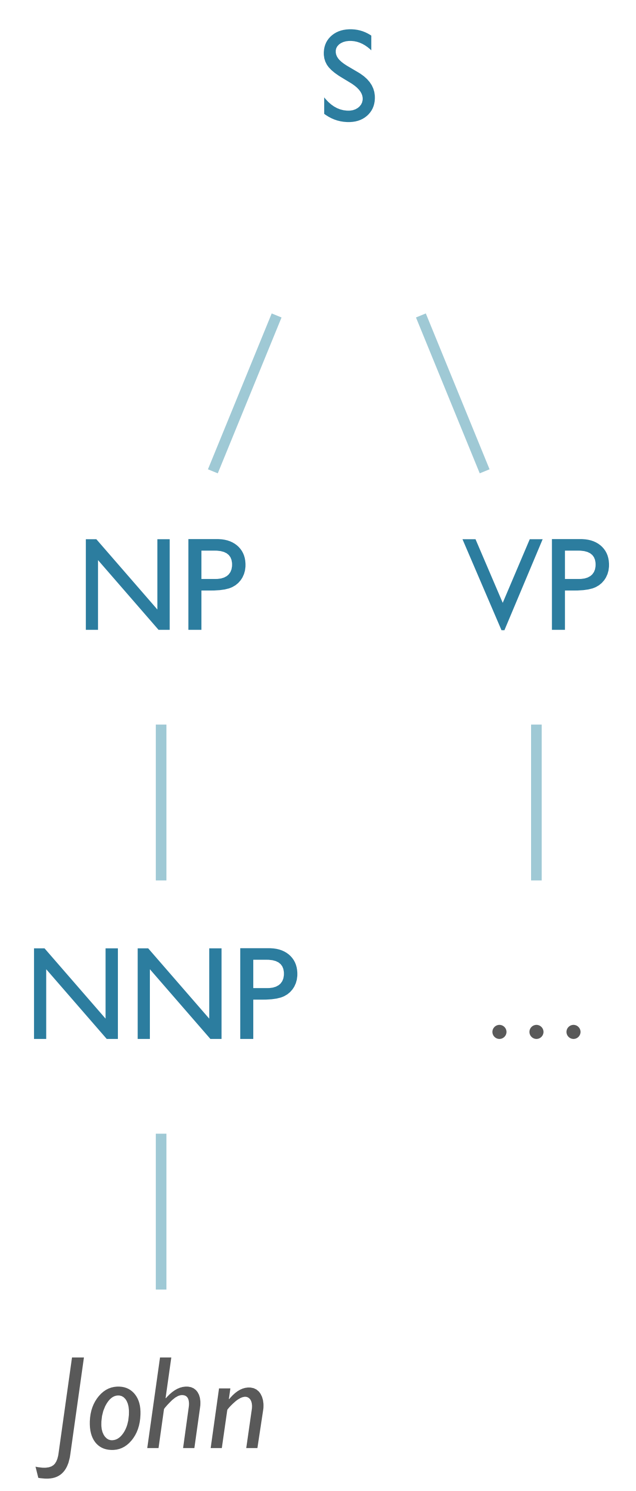
- For NLTK, use the hyphen/minus character: '–'

‘John booked a flight’

- Target representation:
  - $\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e, John) \wedge BookedThing(e, x))$

‘John booked a flight’

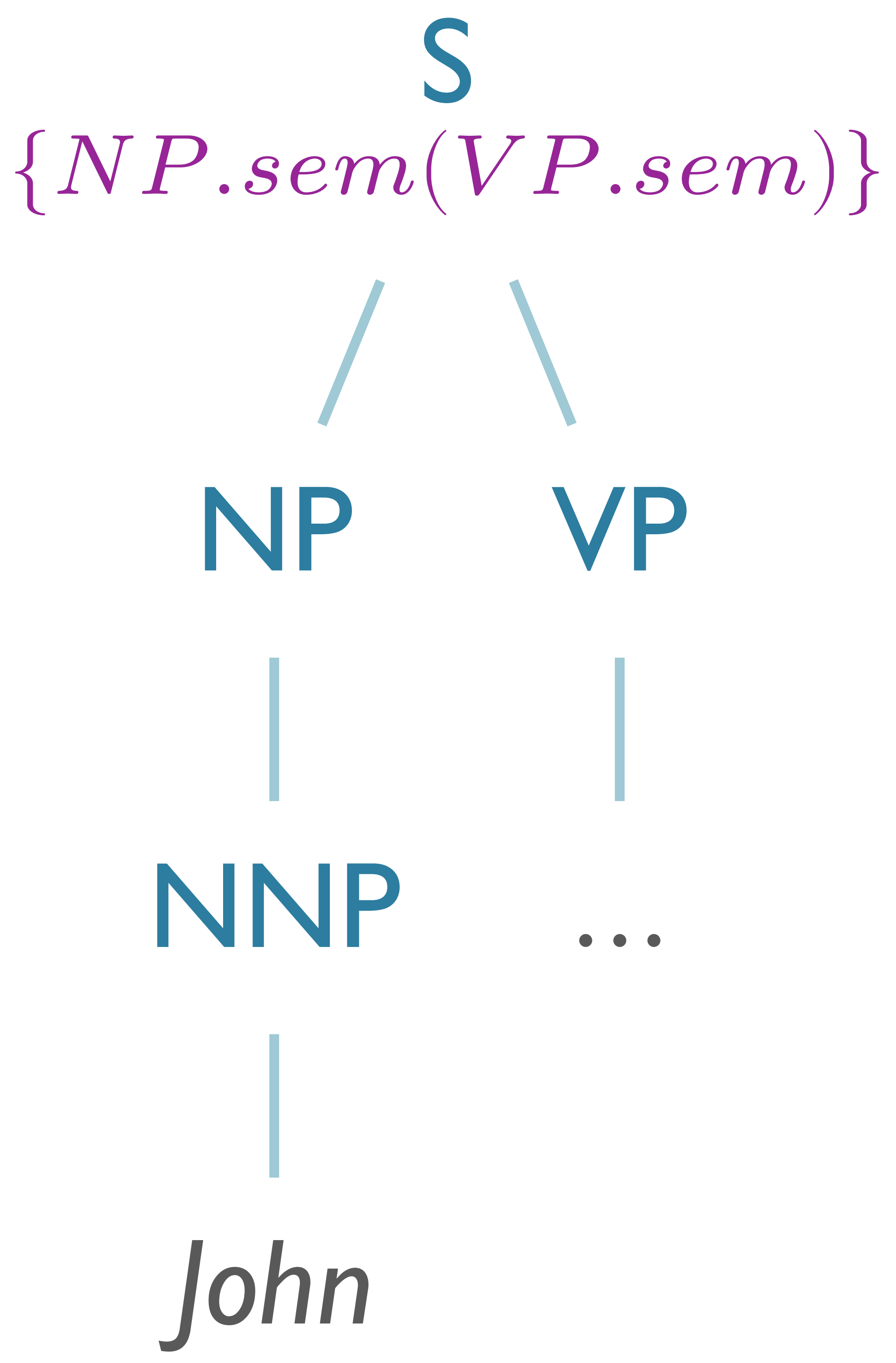
- $\exists x\, Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e, John) \wedge BookedThing(e, x))$



$S \rightarrow NP\ VP$ $\{NP.sem(VP.sem)\}$

'John booked a flight'

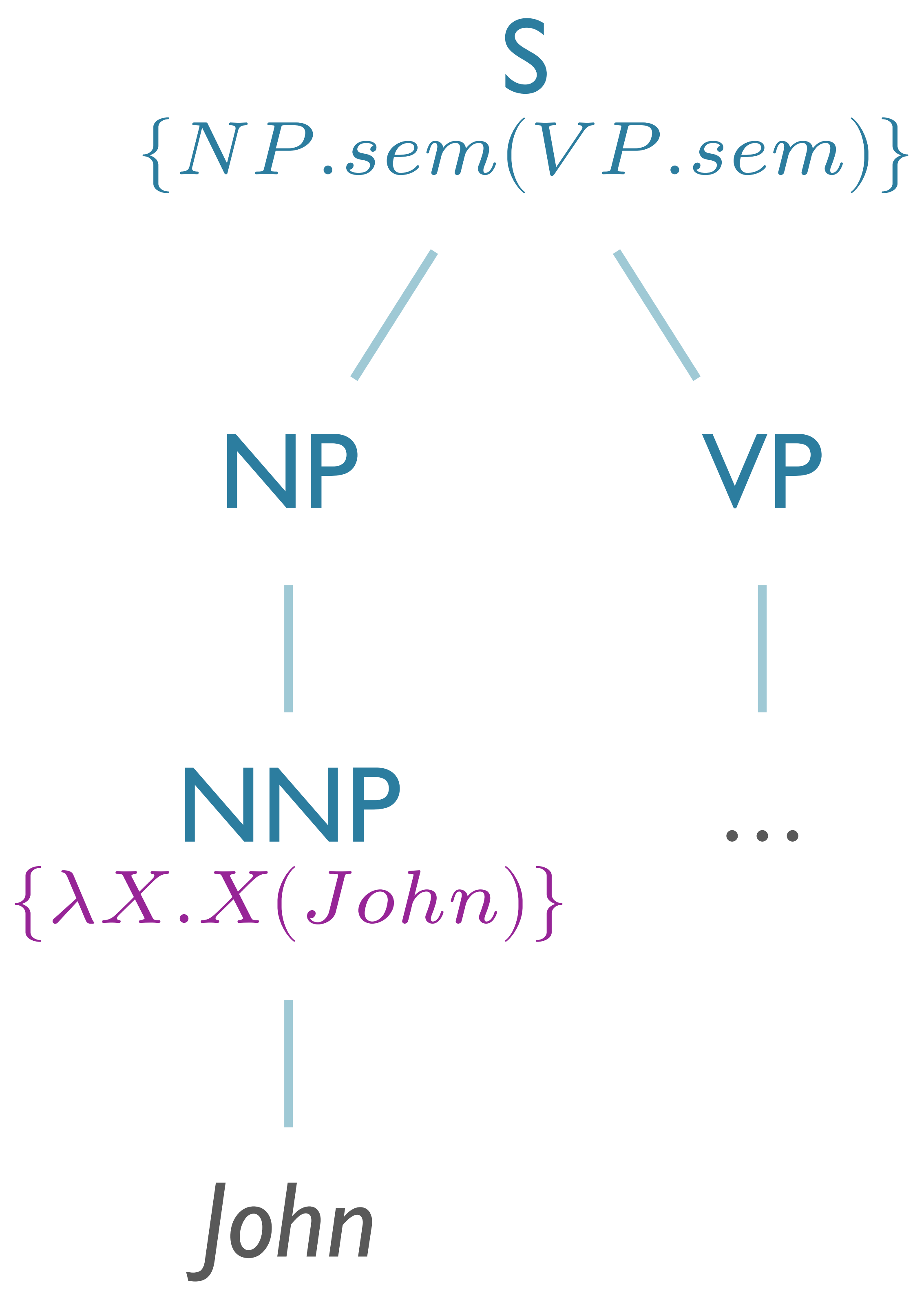
- $\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$



$S \rightarrow NP\ VP \qquad \{NP.sem(VP.sem)\}$

'John booked a flight'

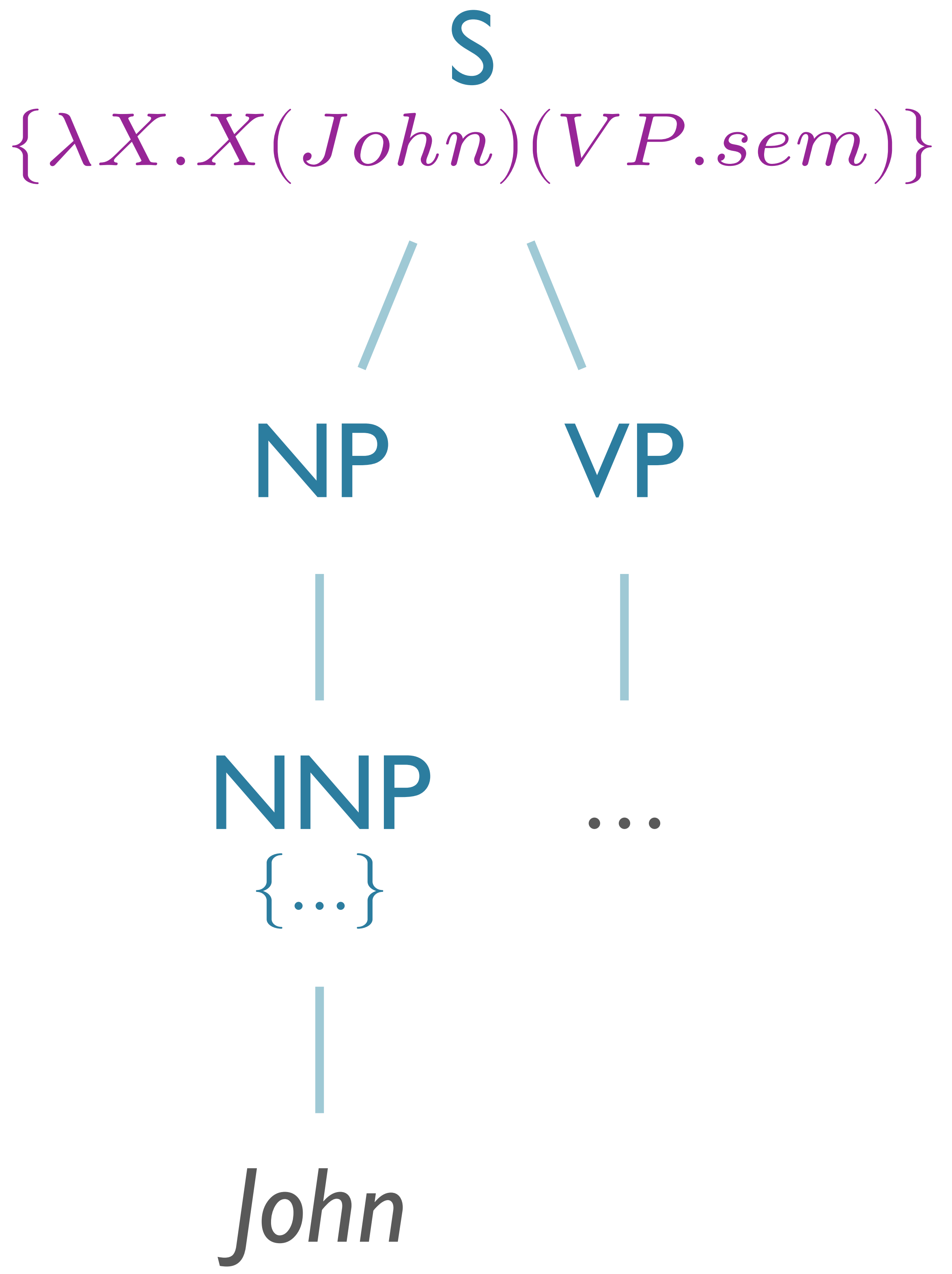
- $\exists x\, Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$



$S \rightarrow NP\ VP$	$\{NP.sem(VP.sem)\}$
$NNP \rightarrow$ 'John'	$\{\lambda X.X(John)\}$
$NP \rightarrow NNP$	$\{NNP.sem\}$
$VP \rightarrow Verb\ NP$	$\{Verb.sem(NP.sem)\}$

‘John booked a flight’

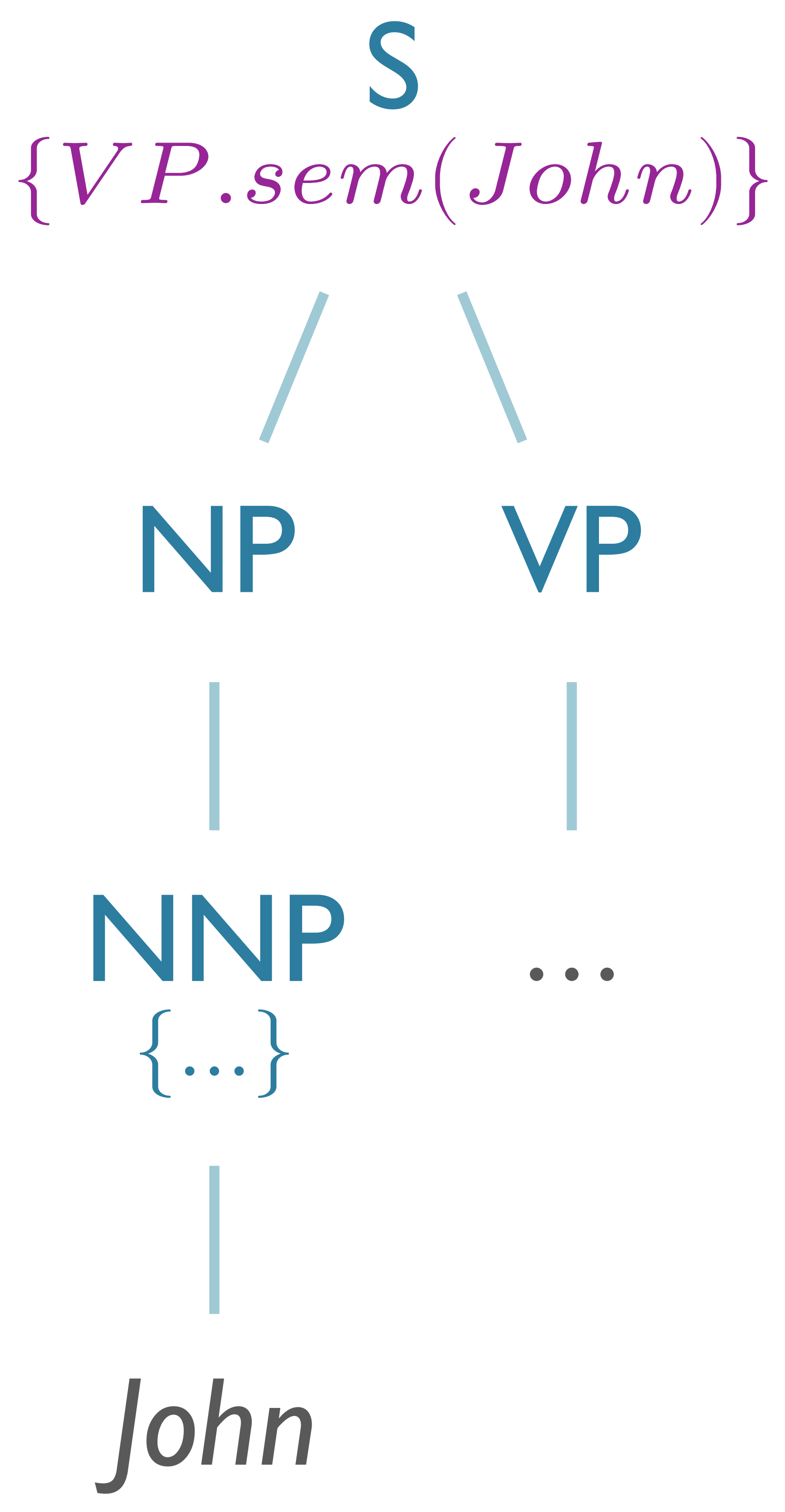
- $\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$



S → NP VP	*{NP.sem(VP.sem)}*
NNP → ‘John’	*{λX.X(John)}*
NP → NNP	*{NNP.sem}*
VP → Verb NP	*{Verb.sem(NP.sem)}*

'John booked a flight'

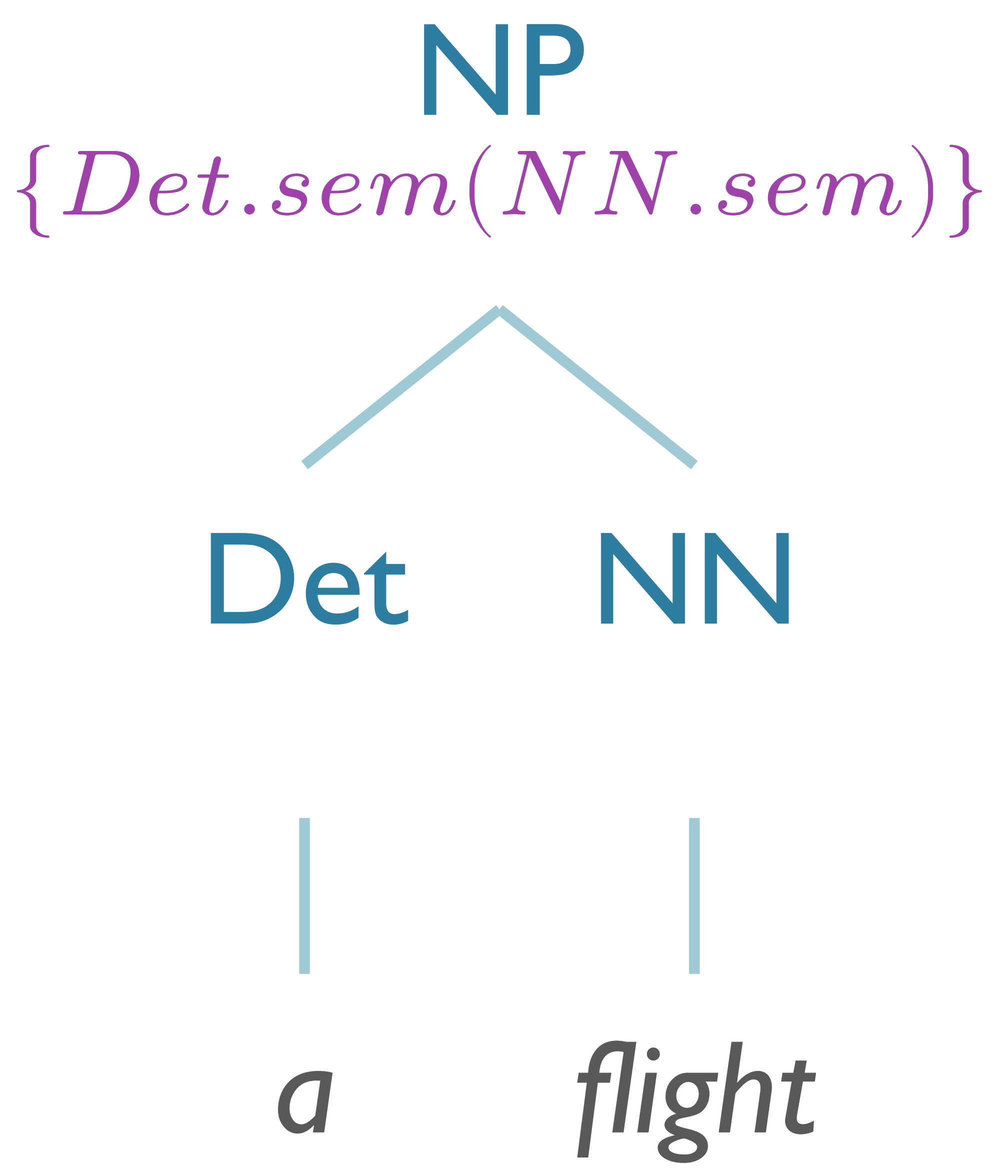
- $\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$



$S \rightarrow NP\ VP$	$\{NP.sem(VP.sem)\}$
$NNP \rightarrow$ 'John'	$\{\lambda X.X(John)\}$
$NP \rightarrow NNP$	$\{NNP.sem\}$
$VP \rightarrow Verb\ NP$	$\{Verb.sem(NP.sem)\}$

'John booked a flight'

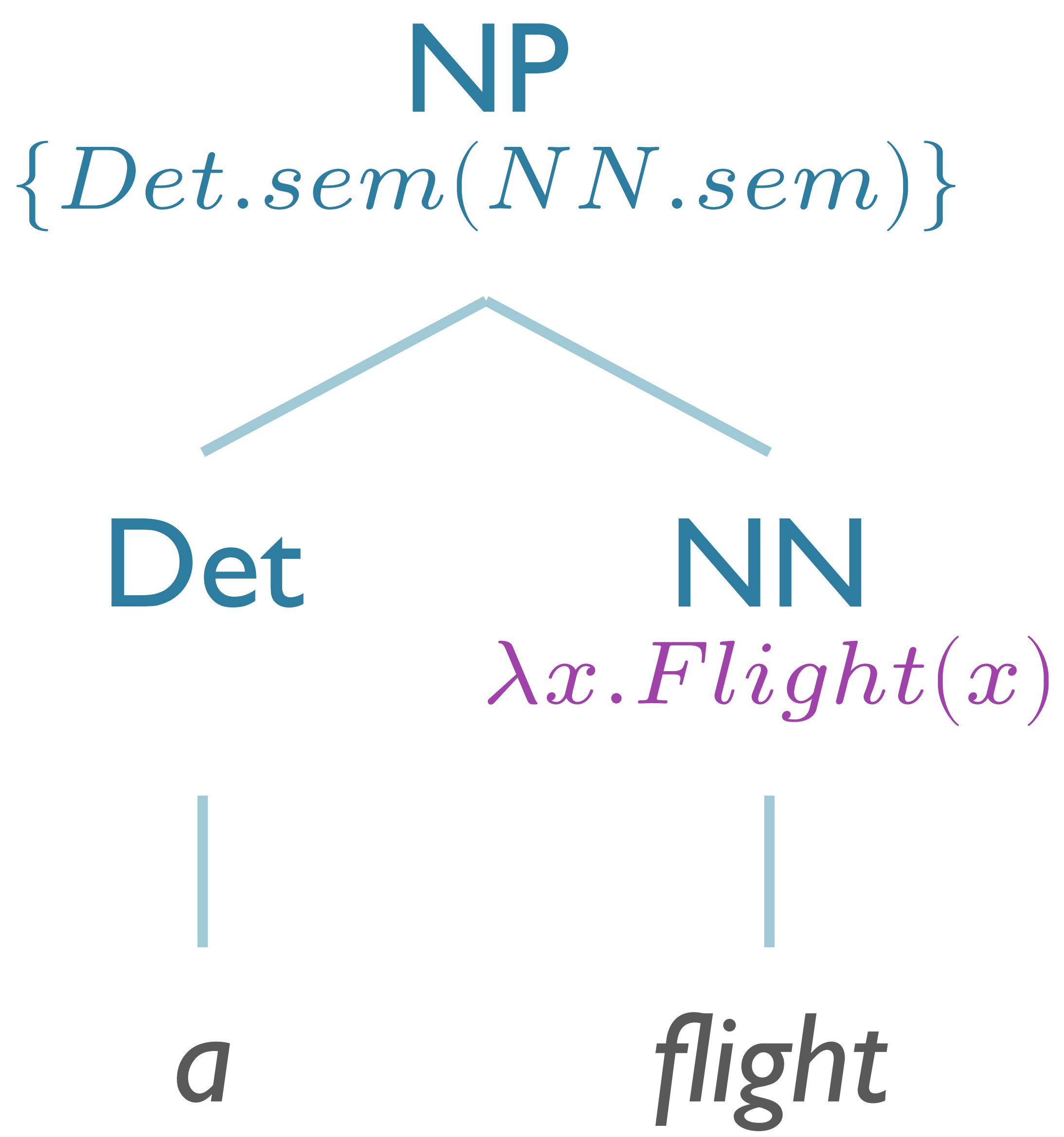
- $\exists x\, Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$



$NP \rightarrow Det\ NN \qquad \{Det.sem(NN.sem)\}$

‘John booked a flight’

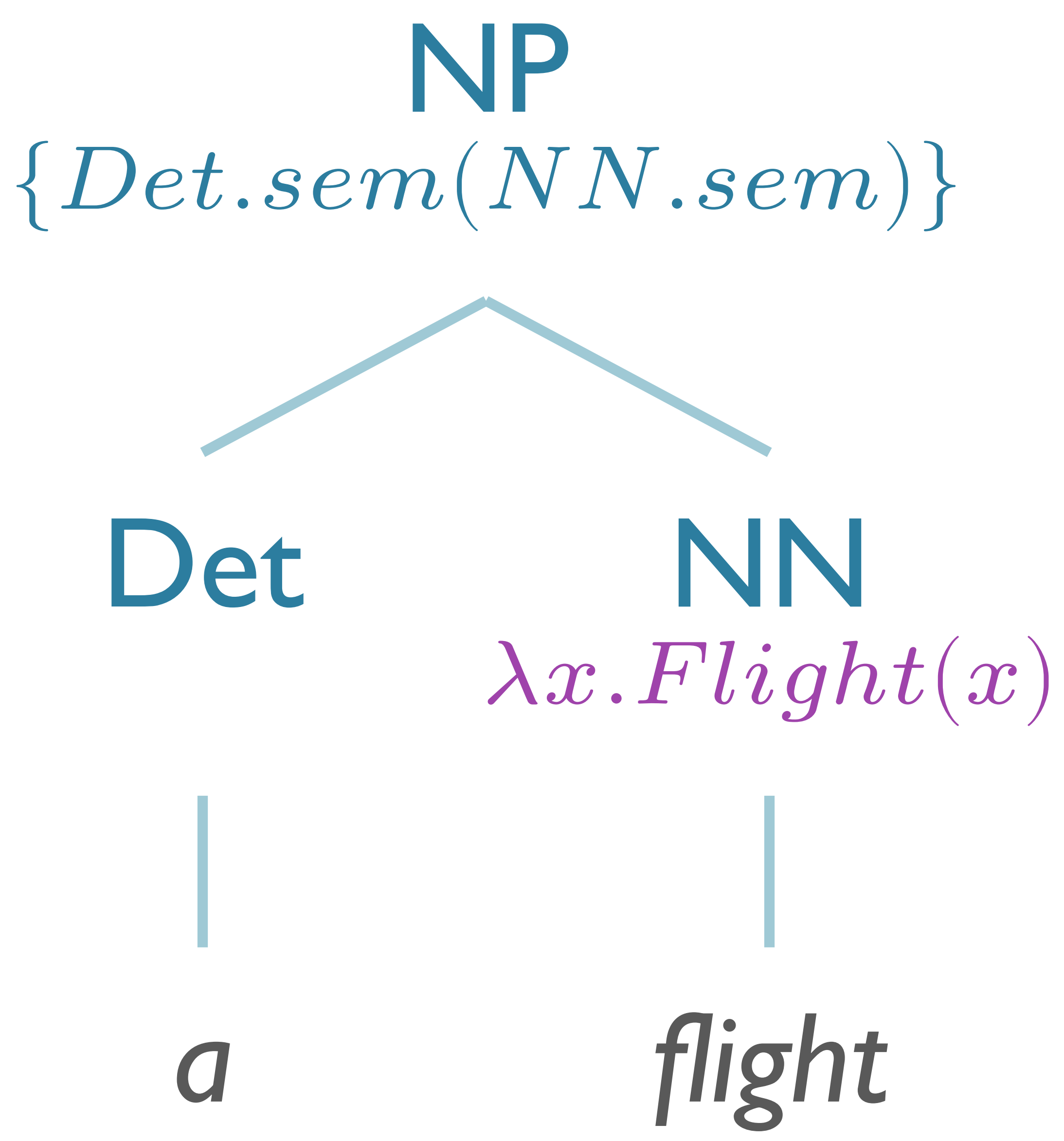
- $\exists x\, Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$



NP → *Det NN*	{*Det.sem*(*NN.sem*)}
NN → *‘flight’*	{λ*x*.*Flight*(*x*)}

'John booked a flight'

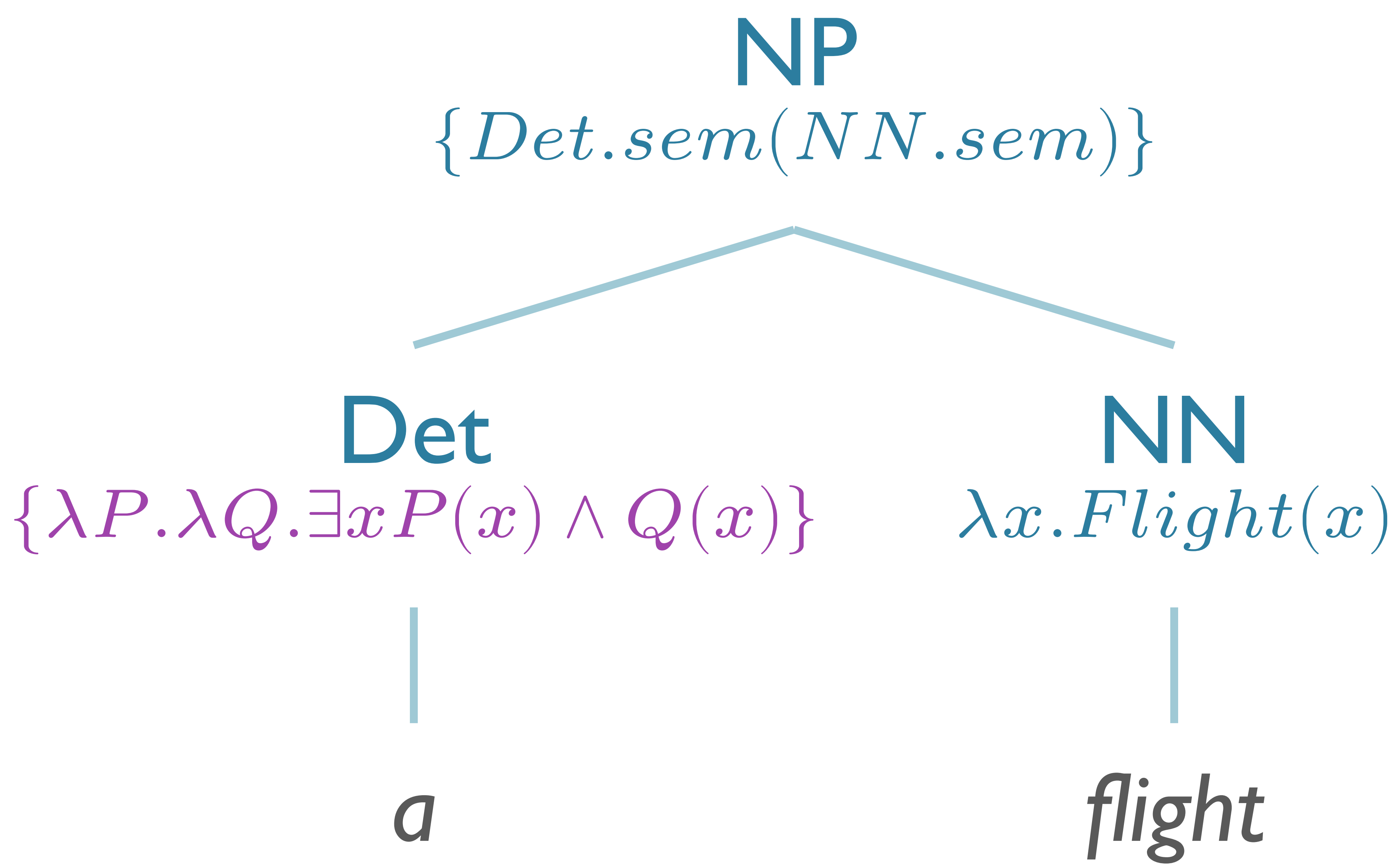
- $\exists x\, Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$



NP → Det NN	$\{Det.sem(NN.sem)\}$
NN → 'flight'	$\{\lambda x.Flight(x)\}$
Det → 'a'	$\{ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x) \}$

‘John booked a flight’

- ∃x *Flight*(*x*) ∧ (∃e*Booked*(e) ∧ *Booker*(*e*, *John*) ∧ *BookedThing*(*e*, *x*))



NP → Det NN	*{Det.sem(NN.sem)}*
NN → ‘flight’	*{λx.Flight(x)}*
Det → ‘a’	*{ λP.λQ.∃x P(x) ∧ Q(x) }*

'John booked a flight'

- $\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$

NP

$\{Det.sem(NN.sem)\}$

$NP \rightarrow Det\ NN$	$\{Det.sem(NN.sem)\}$
$NN \rightarrow$ 'flight'	$\{\lambda x.Flight(x)\}$
$Det \rightarrow$ 'a'	$\{\lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x)\}$

'John booked a flight'

- $\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$

NP

$\{Det.sem(NN.sem)\}$

$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)(\lambda x.Flight(x))\}$

NP → Det NN	$\{Det.sem(NN.sem)\}$
NN → 'flight'	$\{\lambda x.Flight(x)\}$
Det → 'a'	$\{ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x) \}$

'John booked a flight'

- $\exists x\, Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$

NP

$\{Det.sem(NN.sem)\}$

$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)(\lambda x.Flight(x))\}$

$\{\lambda Q.\exists x(\lambda x.Flight(x))(x) \wedge Q(x)\}$

NP → Det NN	{*Det.sem(NN.sem)*}
NN → 'flight'	{*λx.Flight(x)*}
Det → 'a'	{ *λP.λQ.∃x P(x) ∧ Q(x)* }

'John booked a flight'

- $\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$

NP

$$\{Det.sem(NN.sem)\}$$
$$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)(\lambda x.Flight(x))\}$$
$$\{\lambda Q.\exists x(\lambda x.Flight(x))(x) \wedge Q(x)\}$$
$$\{\lambda Q.\exists x Flight(x) \wedge Q(x)\}$$

NP → *Det NN*	{*Det.sem*(*NN.sem*)}
NN → *'flight'*	{$\lambda x.Flight(x)$}
Det → *'a'*	{ $\lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x)$ }

'John booked a flight'

- $\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$

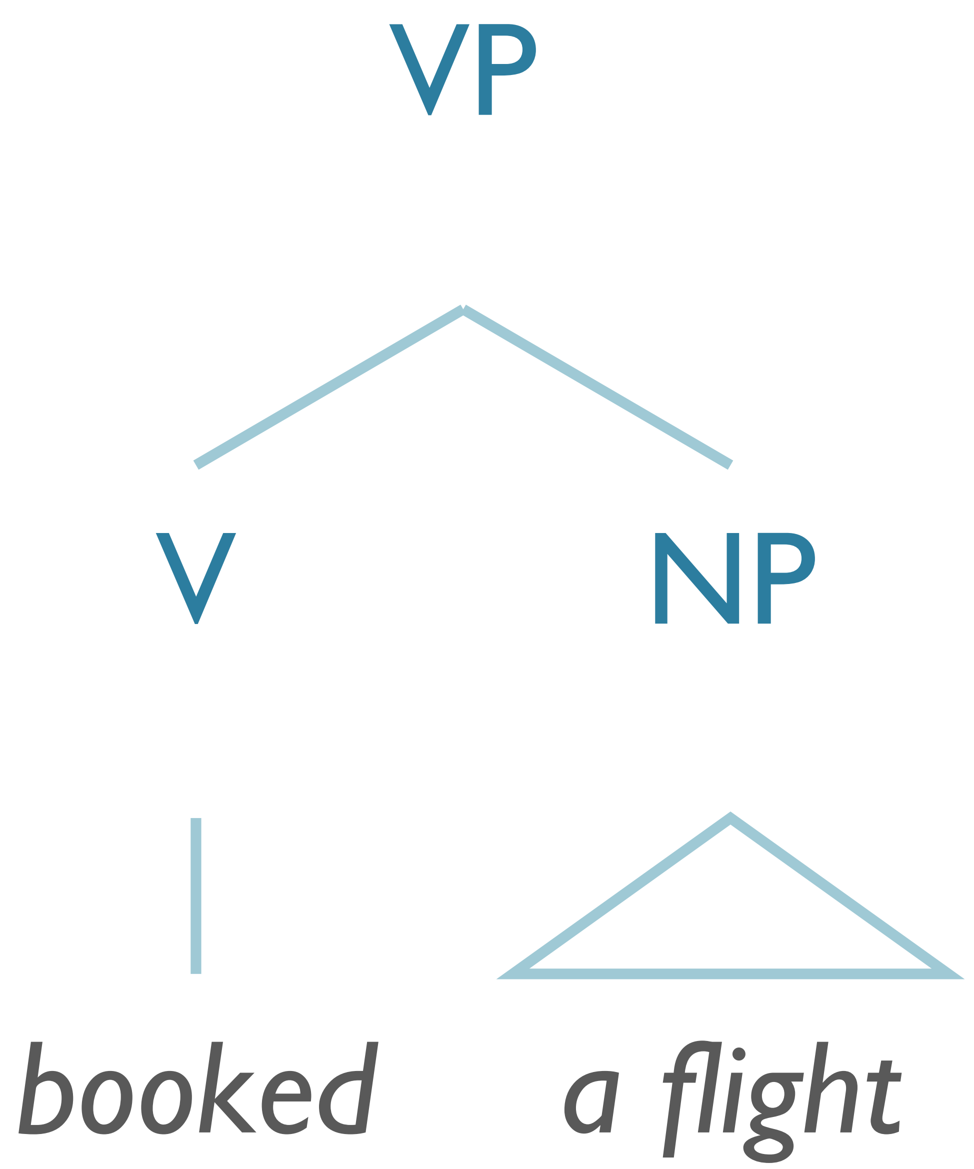
NP

$$\{Det.sem(NN.sem)\}$$
$$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)(\lambda x.Flight(x))\}$$
$$\{\lambda Q.\exists x(\lambda x.Flight(x))(x) \wedge Q(x)\}$$
$$\{\lambda Q.\exists x Flight(x) \wedge Q(x)\}$$

NP → Det NN	$\{Det.sem(NN.sem)\}$
NN → 'flight'	$\{\lambda x.Flight(x)\}$
Det → 'a'	$\{\lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x)\}$
'a flight'	$\{\lambda Q.\exists x\ Flight(x) \wedge Q(x)\}$

‘John booked a flight’

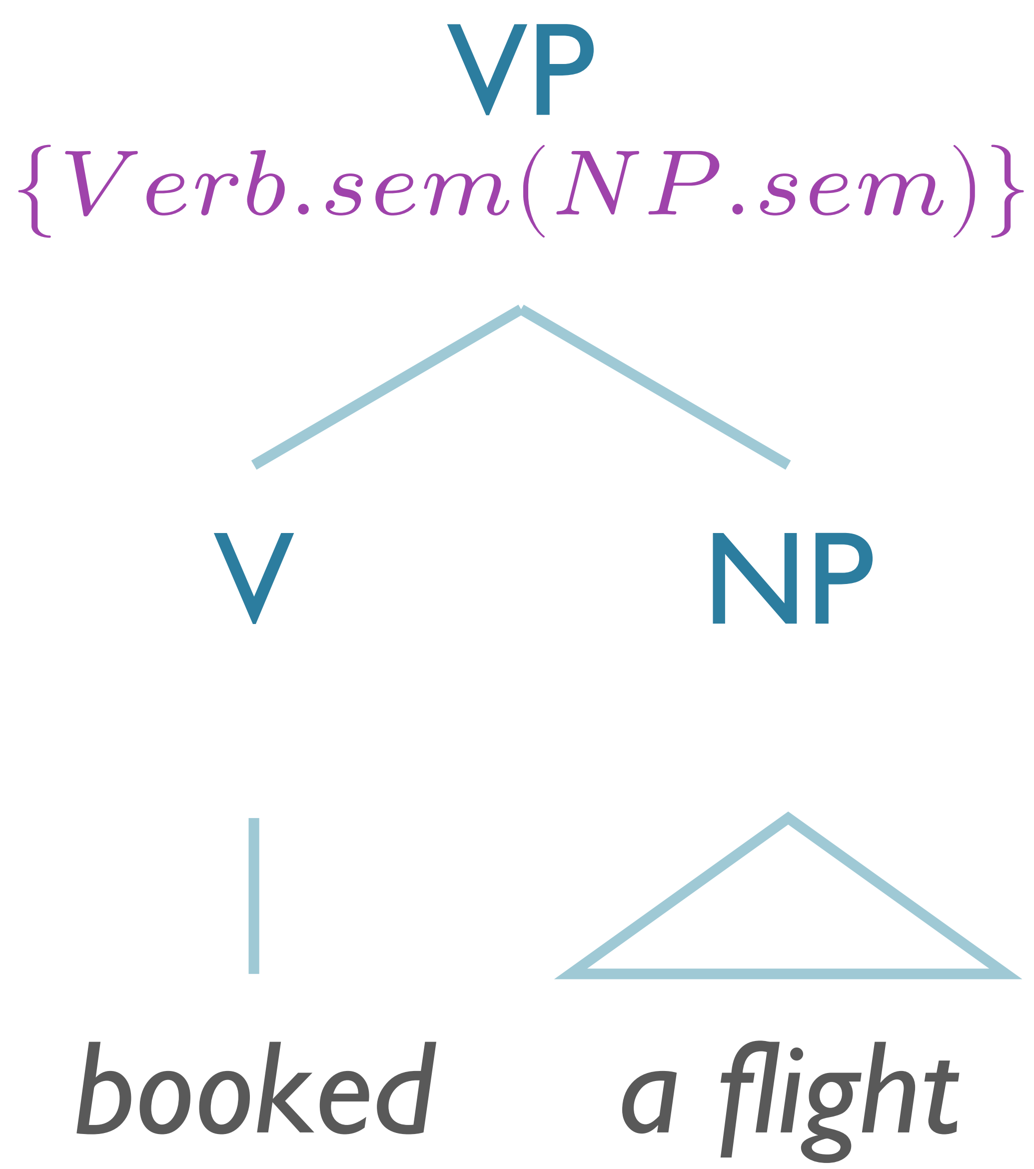
- $\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e, John) \wedge BookedThing(e, x))$



VP → Verb NP	*{Verb.sem(NP.sem)}*
‘a flight’	$\{\ \lambda Q.\exists x\ Flight(x) \wedge Q(x)\ \}$

'John booked a flight'

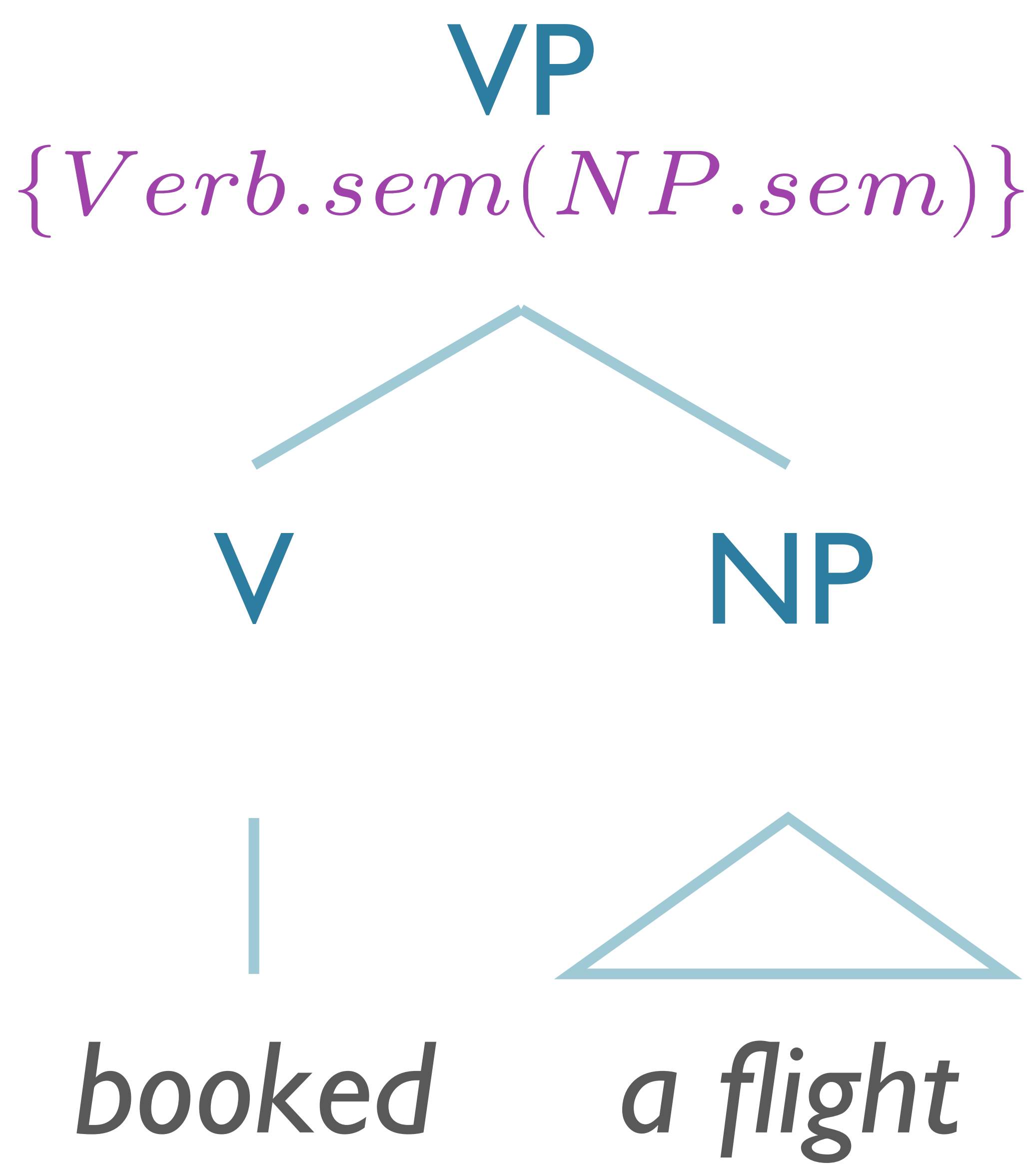
- $\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$



VP → Verb NP	$\{Verb.sem(NP.sem)\}$
'a flight'	$\{ \lambda Q.\exists x\ Flight(x) \wedge Q(x) \}$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$



Verb → *‘booked’*

$\{\lambda W.\lambda z.W(\lambda y.\exists e Booked(e) \wedge Booker(e,z) \wedge BookedThing(e,y))\}$

VP → *Verb NP*	$\{Verb.sem(NP.sem)\}$
‘a flight’	$\{\ \lambda Q.\exists x\ Flight(x) \wedge Q(x)\ \}$

'John booked a flight'

- $\exists x\, Flight(x) \wedge (\exists e Booked(\text{e}) \wedge Booker(e, John) \wedge BookedThing(e, x))$

Verb.sem(*NP.sem*)

'John booked a flight'

- $\exists x\ Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e, John) \wedge BookedThing(e, x))$

$Verb.sem(NP.sem)$

$\lambda W.\lambda z.W(\lambda y.\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,y))(\lambda Q.\exists x\ Flight(x) \wedge Q(x))$

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$

$Verb.sem(NP.sem)$

$\lambda W.\lambda z.W(\lambda y.\exists e Booked(e) \wedge Booker(e,z) \wedge BookedThing(e,y))(\lambda Q.\exists x\ Flight(x) \wedge Q(x))$

$\lambda z.(\lambda Q.\exists x\ Flight(x) \wedge Q(x))(\lambda y.\exists e Booked(e) \wedge Booker(e,z) \wedge BookedThing(e,y))$

'John booked a flight'

- ∃*x Flight*(*x*) ∧ (∃*eBooked*(e) ∧ *Booker*(*e, John*) ∧ *BookedThing*(*e, x*))

Verb.sem(*NP.sem*)

λ*W*.λ*z*.*W*(λ*y*.∃*eBooked*(e) ∧ *Booker*(*e,z*) ∧ *BookedThing*(*e,y*))(λ*Q*.∃*x Flight*(*x*) ∧ *Q*(*x*))

λ*z*.(λ*Q*.∃*x Flight*(*x*) ∧ *Q*(*x*))(λ*y*.∃*eBooked*(e) ∧ *Booker*(*e,z*) ∧ *BookedThing*(*e,y*))

λ*z*.∃*x Flight*(*x*) ∧ (λ*y*.∃*eBooked*(e) ∧ *Booker*(*e,z*) ∧ *BookedThing*(*e,y*))(*x*)

'John booked a flight'

- $\exists x\, Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$

$Verb.sem(NP.sem)$

$\lambda W.\lambda z.W(\lambda y.\exists e Booked(e) \wedge Booker(e,z) \wedge BookedThing(e,y))(\lambda Q.\exists x\, Flight(x) \wedge Q(x))$

$\lambda z.(\lambda Q.\exists x\, Flight(x) \wedge Q(x))(\lambda y.\exists e Booked(e) \wedge Booker(e,z) \wedge BookedThing(e,y))$

$\lambda z.\exists x\, Flight(x) \wedge (\lambda y.\exists e Booked(e) \wedge Booker(e,z) \wedge BookedThing(e,y))(x)$

$\lambda z.\exists x\, Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e,z) \wedge BookedThing(e,\boldsymbol{x})$

'John booked a flight'

- $\exists x\, Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e, John) \wedge BookedThing(e, x))$

$Verb.sem(NP.sem)$

$\lambda W.\lambda z.W(\lambda y.\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,y))(\lambda Q.\exists x\, Flight(x) \wedge Q(x))$

$\lambda z.(\lambda Q.\exists x\, Flight(x) \wedge Q(x))(\lambda y.\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,y))$

$\lambda z.\exists x\, Flight(x) \wedge (\lambda y.\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,y))(x)$

$\lambda z.\exists x\, Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e,z) \wedge BookedThing(e,\mathbf{x})$

‘John booked a flight’

- $\exists x$ *Flight*(x) $\wedge$ ($\exists e$*Booked*(e) $\wedge$ *Booker*(e, *John*) $\wedge$ *BookedThing*(e, x))

S
$\{VP.sem(John)\}$

NP VP

NNP ...
$\{...\}$

John

S

‘booked a flight'

VP.sem(*John*)

$\lambda z.\exists x$ *Flight*(x) $\wedge$ ($\exists e$*Booked*(e) $\wedge$ *Booker*(e,z) $\wedge$ *BookedThing*($e,\boldsymbol{x}$)

‘John booked a flight’

- $\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e, John) \wedge BookedThing(e, x))$

S — $VP.sem(John)$

‘booked a flight' — $\lambda z.\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e,z) \wedge BookedThing(e,\mathbf{x})$

$\boldsymbol{\lambda z}.\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e,z) \wedge BookedThing(e,x)(\mathbf{John})$

$\exists x\ Flight(x) \wedge (\exists e Booked(e) \wedge Booker(e,\mathbf{John}) \wedge BookedThing(e,x)$

'John booked a flight'

$Det \rightarrow$ 'a'	$\{ \lambda P.\lambda Q.\exists x\ P(x) \wedge Q(x) \}$
$Det \rightarrow$ 'every'	$\{ \lambda P.\lambda Q.\forall x\ P(x) \Rightarrow Q(x) \}$
$NN \rightarrow$ 'flight'	$\{\lambda x.Flight(x)\}$
$Verb \rightarrow$ 'booked'	$\{\lambda W.\lambda z.W(\lambda y.\exists e Booked(e) \wedge Booker(e,z) \wedge BookedThing(e,y))\}$
$NNP \rightarrow$ 'John'	$\{\lambda X.X(John)\}$
$NP \rightarrow NNP$	$\{NNP.sem\}$
$NP \rightarrow Det\ NN$	$\{Det.sem(NN.sem)\}$
$S \rightarrow NP\ VP$	$\{NP.sem(VP.sem)\}$
$VP \rightarrow Verb\ NP$	$\{Verb.sem(NP.sem)\}$

‘John booked no flight’

- $\neg(\exists x\, Flight(x) \wedge (\exists e Booked(\mathrm{e}) \wedge Booker(e, John) \wedge BookedThing(e, x)))$
- $\forall x Flight(x) \Rightarrow \neg(\exists e Booked(\mathrm{e}) \wedge Booker(e, John) \wedge BookedThing(e, x)$

'John booked no flight'

Det → 'no'	{ $\lambda P.\lambda Q.\neg\exists x\, P(x) \wedge Q(x)$}
Det → 'a'	{ $\lambda P.\lambda Q.\exists x\, P(x) \wedge Q(x)$ }
Det → 'every'	{ $\lambda P.\lambda Q.\forall x\, P(x) \Rightarrow Q(x)$ }
NN → 'flight'	{$\lambda x.Flight(x)$}
Verb → 'booked'	{$\lambda W.\lambda z.W(\lambda y.\exists e Booked(e) \wedge Booker(e,z) \wedge BookedThing(e,y))$}
NNP → 'John'	{$\lambda X.X(John)$}
NP → *NNP*	{*NNP.sem*}
NP → *Det NN*	{*Det.sem(NN.sem)*}
S → *NP VP*	{*NP.sem(VP.sem)*}
VP → *Verb NP*	{*Verb.sem(NP.sem)*}

Other Lambda Calculus

Adjectives

Adjectives

- Similar to nouns, but with an extra conjunction and dummy predicate:

Adjectives

- Similar to nouns, but with an extra conjunction and dummy predicate:
  - "red" = $\lambda P\, \lambda x(red(x) \wedge P(x))$

Adjectives

- Similar to nouns, but with an extra conjunction and dummy predicate:
  - “red” = $\lambda P\, \lambda x(red(x) \wedge P(x))$
- Any issues?

Adjectives

- Similar to nouns, but with an extra conjunction and dummy predicate:
  - "red" = $\lambda P\, \lambda x(red(x) \wedge P(x))$
- Any issues?
  - Non-intersective adjectives (e.g. 'skillful', 'alleged', 'fake')

Definite Article

- $a = \lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))$
- $the = \lambda P.\lambda Q.\exists x(P(x) \wedge \forall y(\mathrm{P}(y) \Leftrightarrow x{=}y) \wedge \mathrm{Q}(x)))$
- Roughly: “The P Q”: there is a *unique* P, which is also Q
  - Unique: *x* is P, and anything else that is also P is equal to *x*

Definite Article

- *the* = $\lambda P.\lambda Q.\exists x(P(x) \wedge \forall y(P(y) \Leftrightarrow x{=}y) \wedge Q(x)))$
  - Bertrand Russel, "*On Denoting*" (1905).
  - The definite article isn't exactly the same as a constant (like "*John*")
  - Rather, it picks out a set of items from a set (the generic NN), and makes a strong assertion:
    A) The book arrived.
    B) A book arrived.
    - A $\vDash$ B, but B $\nvDash$ A

Definite Article + Presupposition

- “The slides for Monday are amazing.”
  - ~> there are slides for Monday.
- “The slides for Monday are not amazing.”
  - ~> there are slides for Monday.
- The P Q: *presupposes* that there is a unique P, does not *assert* it [Strawson 1950, …]
  - If there is no P, “The P Q” is *neither true nor false*

Learning Semantic Parsers

Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars

Luke S. Zettlemoyer and Michael Collins
MIT CSAIL
lsz@csail.mit.edu, mcollins@csail.mit.edu

Abstract

This paper addresses the problem of mapping natural language sentences to lambda–calculus encodings of their meaning. We describe a learning algorithm that takes as input a training set of sentences labeled with expressions in the lambda calculus. The algorithm induces a grammar for the problem, along with a log-linear model that represents a distribution over syntactic and semantic analyses conditioned on the input sentence. We apply the method to the task of learning natural language interfaces to databases and show that the learned parsers outperform previous methods in two benchmark database domains.

Supervised learning:

- Sentences labeled with logical forms
- Induce grammar
- Plus semantic attachments
- Score analyses of ambiguous sentences with log-linear model

Zettlemoyer and Collins 2005

Learning from Denotations

Liang, Jordan, and Klein   Learning Dependency-Based Compositional Semantics

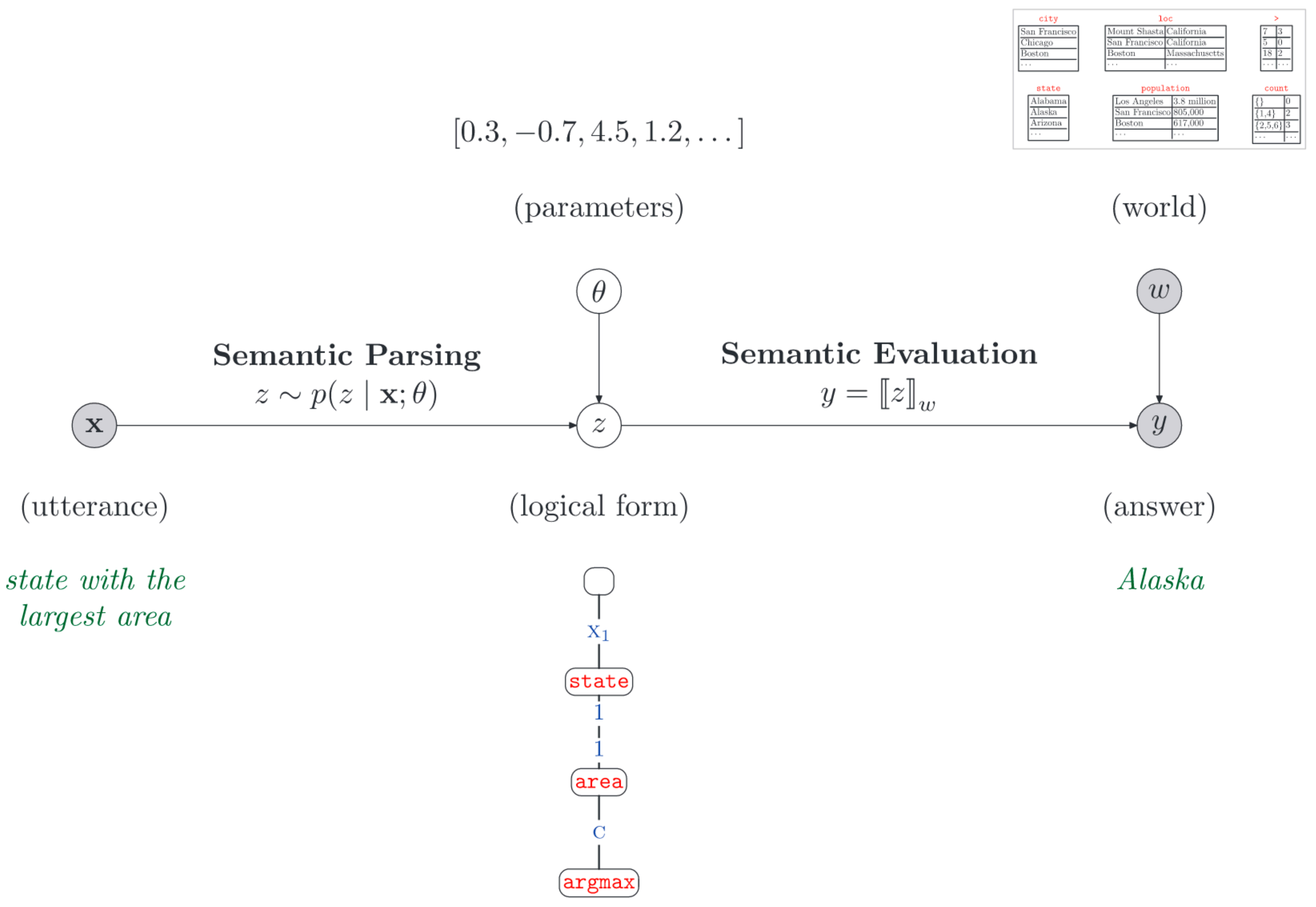


Figure 2
Our statistical methodology consists of two steps: (i) semantic parsing ($p(z \mid \mathbf{x}; \theta)$): an utterance $\mathbf{x}$ is mapped to a logical form z by drawing from a log-linear distribution parametrized by a vector θ; and (ii) evaluation ($[[z]]_w$): the logical form z is evaluated with respect to the world w (database of facts) to deterministically produce an answer y. The figure also shows an example configuration of the variables around the graphical model. Logical forms z are represented as labeled trees. During learning, we are given w and (x, y) pairs (shaded nodes) and try to infer the latent logical forms z and parameters θ.

Learn semantic representations as *latent variables* for downstream task (QA, conversation, …)

[Liang et al 2011]

Resources

- Datasets
  - General:
    - Abstract Meaning Representations: LDC2017T10
    - Minimal Recursion Semantics: DeepBank
  - SQL:
    - Spider: https://yale-lily.github.io/spider
    - SParC: https://yale-lily.github.io/sparc

Resources: Knowledge Graphs

- R.I.P. Freebase
  - Used by Google Knowledge Graph, then bought and killed
  - [they have an API with 100,000 queries/day for free]
- BUT: data moved to Wikidata

Lexical Semantics

Title logo

All these lambdas have me feeling...

0

Nobody has responded yet.

Hang tight! Responses are coming in.

Compositional vs Lexical Semantics

Foreword

In the spring of 1976, Terry Parsons and Barbara Partee taught a course on Montague grammar, which I attended. On the second to the final day of class, Terry went around the room asking the students if there were any questions at all that remained unanswered, and promised to answer them on the last day of class. I asked if he really meant ANY question at all, which he emphatically said that he meant. As I had encountered a few questions in my lifetime that remained at least partially unresolved, I decided to ask one of them. What is life? What is the meaning of life? After all, Barbara and Terry had promised to provide answers to any question at all.

On the final day of class Barbara wore her Montague grammar T-shirt, and she and Terry busied themselves answering our questions. At long last, they came to my question. I anticipated a protracted and involved answer, but their reply was crisp and succinct. First Barbara, chalk in hand, showed me the meaning of life.

^life'

Terry then stepped up and showed me what life really is.

ˇ^life'

As we were asked to show on a homework assignment earlier in the year, this is equivalent to: life'.

Leaving me astounded that I had been living in such darkness for all these years, the class then turned to the much stickier problem of pronouns.

Carlson 1980

Lexical Semantics

- Thus far: POS → *Word* {sem}
  - Can compose larger semantic formulae bottom-up this way
  - …but we haven't really discussed what a "word" is, semantically.

Lexical Semantics

- Thus far: POS → *Word* {sem}
  - Can compose larger semantic formulae bottom-up this way
  - …but we haven't really discussed what a "word" is, semantically.
- Lexical semantics:
  - How do we formally discuss what a "word" is?
  - How do we relate words to one another?
  - How do we differentiate/relate linked senses?

What is a Plant?

What is a Plant?

- There are more kinds of **plants** and animals in the rainforests than anywhere else on Earth. Over half of the millions of known species of **plants** and animals live in the rainforest. Many are found nowhere else. There are even **plants** and animals in the rainforest that we have not yet discovered.

What is a Plant?

- There are more kinds of **plants** and animals in the rainforests than anywhere else on Earth. Over half of the millions of known species of **plants** and animals live in the rainforest. Many are found nowhere else. There are even **plants** and animals in the rainforest that we have not yet discovered.
- The Paulus company was founded in 1938. Since those days the product range has been the subject of constant expansions and is brought up continuously to correspond with the state of the art. We're engineering, manufacturing, and commissioning world-wide ready-to-run **plants** packed with our comprehensive know-how.

Lexical Semantics

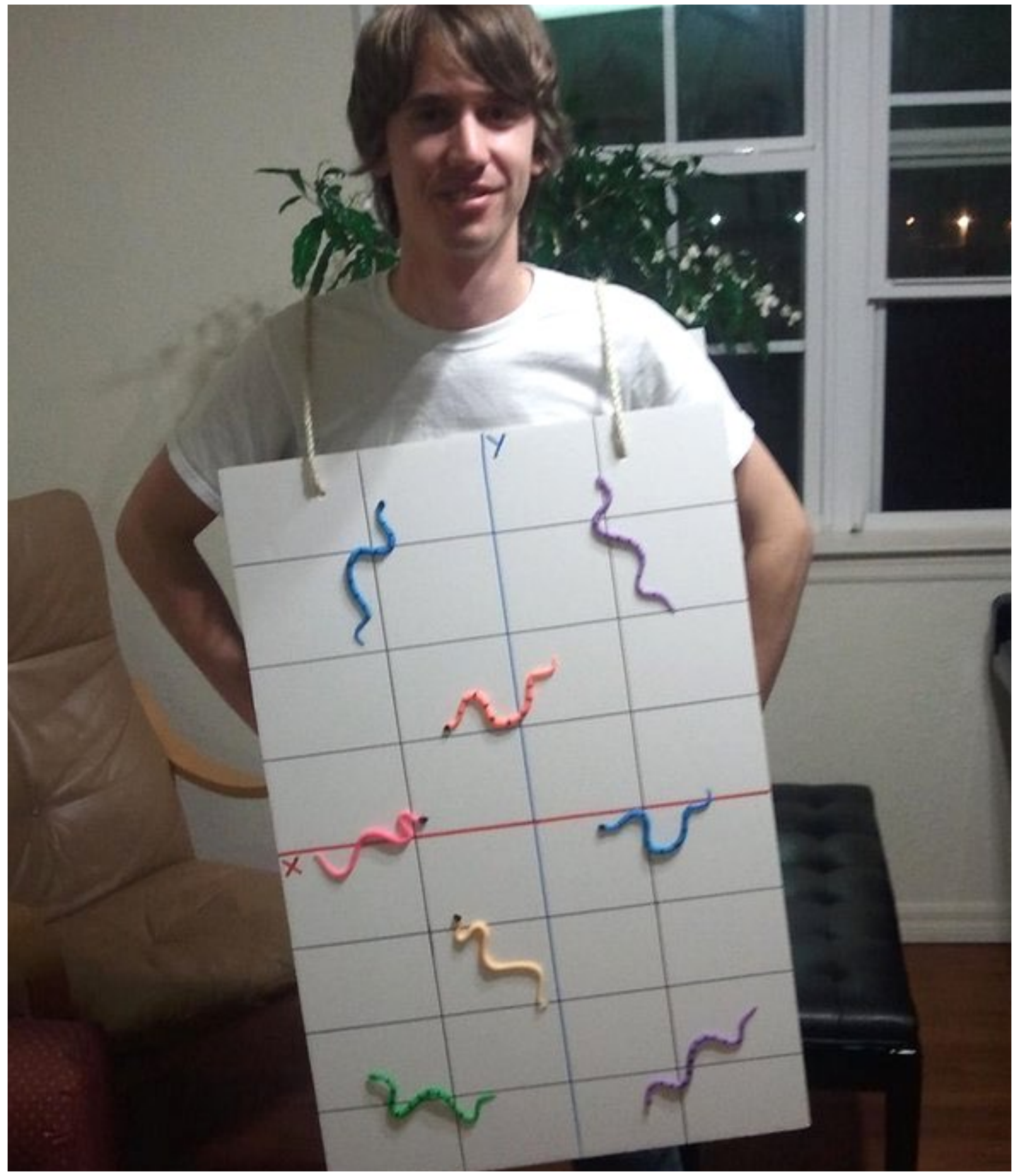
Lexical Semantics

…by way of dad-joke Halloween costumes. 🎃

Lexical Semantics

…by way of dad-joke Halloween costumes. 🎃

A Ceiling Fan

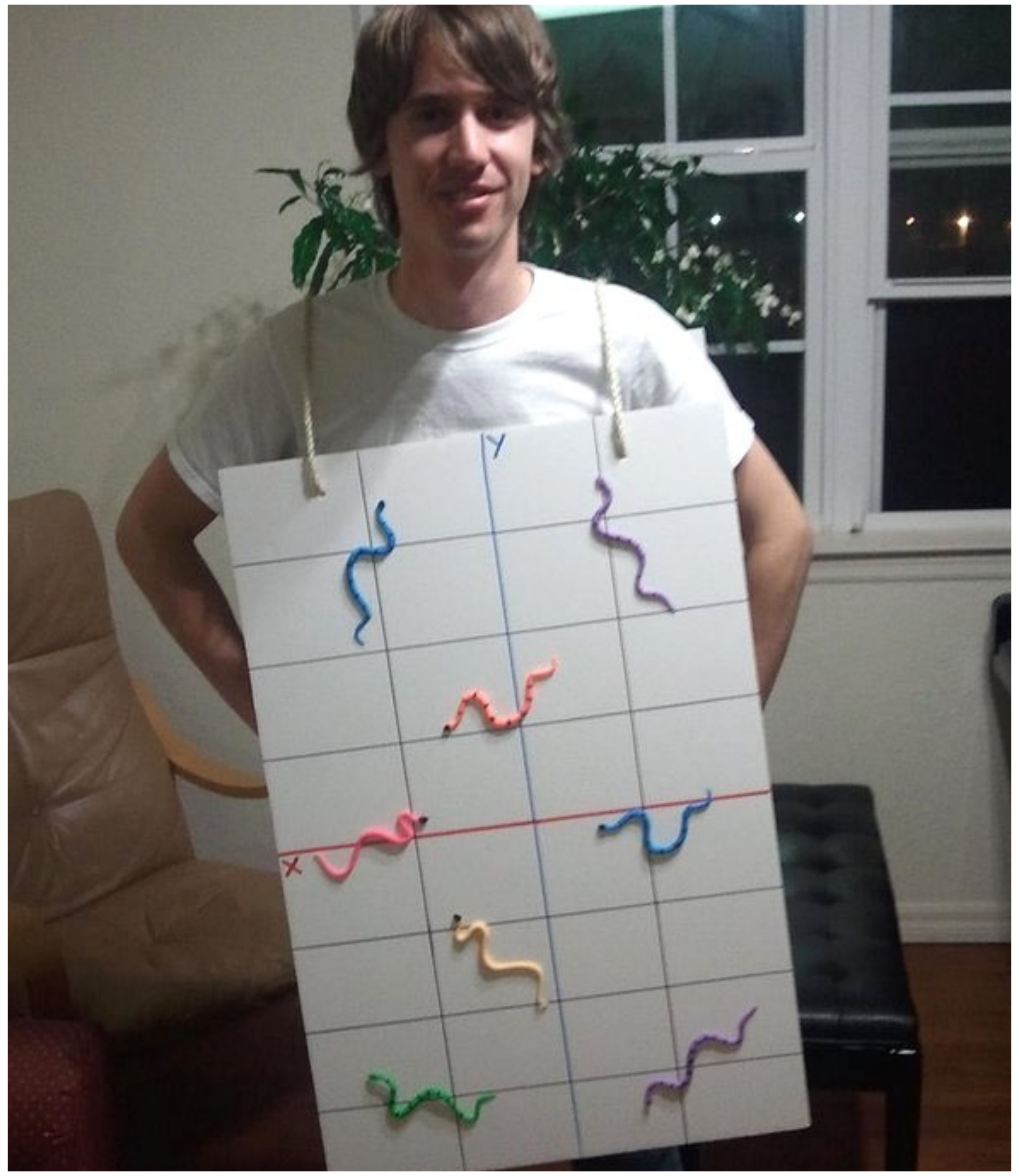


Snakes on a Plane

Lexical Semantics

…by way of dad-joke Halloween costumes. 🎃

A Ceiling Fan



Snakes on a Plane

Sources of Confusion

Homonymy

Sources of Confusion

Homonymy
Polysemy

Sources of Confusion

Homonymy
Polysemy
Synonymy

Sources of Confusion

Homonymy
Polysemy
Synonymy
Antonymy

Sources of Confusion

Homonymy
Polysemy
Synonymy
Antonymy
[Hypo/Hyper]-nymy

Sources of Confusion: Homonymy

- Words have same form but different meanings
  - Generally same POS, but unrelated meaning
    - bank_1 (side of river)
    - bank_2 (financial institution)

Sources of Confusion: Homonymy

- Different types of Homonymy:
  - ***Homophones***: same phonology, different orthographic form
    - two
    - to
    - too
  - ***Homographs***: Same orthography, different phonology:
    - "*lead*" (metal)
    - "*lead*" (take somewhere)

Sources of Confusion: Homonymy

- Different types of Homonymy:
  - ***Homophones***: same phonology, different orthographic form
    - two
    - to
    - too
  - ***Homographs***: Same orthography, different phonology:
    - "*lead*" (metal)
    - "*lead*" (take somewhere)
- Why do we care?
  - Problem for applications: TTS, ASR transcription, IR

Sources of Confusion: Polysemy

- Multiple RELATED senses
  - e.g. bank: money, organ, blood

Sources of Confusion: Polysemy

- Multiple RELATED senses
  - e.g. bank: money, organ, blood
- Big issue in lexicography
  - Number of senses
  - Relations between senses
  - Differentiation

Sources of Confusion: Polysemy

- Example: [[*serve*]]
  - *serve breakfast*
  - *serve Philadelphia*
  - *serve time*

Sources of Confusion: Synonymy

- (near) identical meaning
- Substitutability
  - Maintains propositional meaning

Sources of Confusion: Synonymy

- Issues:
  - Also has polysemy!
    - Shades of meaning - other associations
      - *price* vs. *fare*
      - *big* vs. *large*
      - *water* vs. *H_20*
- Collocational constraints
  - e.g. *babbling brook* vs. **babbling river*
- Register:
  - social factors: e.g. politeness, formality

Sources of Confusion: Antonymy

- Opposition
- Typically ends of a scale
  - *fast* vs. *slow*
  - *big* vs. *little*
- Can be hard to distinguish automatically from synonyms

Sources of Confusion: Hyponomy

- `instanceOf(x, y)` relations:
- More General (***hypernym***) vs. more specific (***hyponym***)
  - *dog* vs. *golden retriever*
  - *fruit* vs. *mango*
- Organize as ontology/taxonomy

Word Sense Disambiguation

- Application of lexical semantics
- Goal: given a word ***in context***, identify the appropriate sense
  - e.g. <u>plants</u> and animals in the rainforest
- Crucial for real syntactic & semantic analysis
  - Correct sense can determine
    - Available syntactic structure
    - Available thematic roles, correct meaning…

Robust Disambiguation

- More to semantics than predicate-argument structure
  - Select sense where predicates underconstrain
- Learning approaches
  - Supervised, bootstrapped, unsupervised
- Knowledge-based approaches
  - Dictionaries, taxonomies
- Contexts for sense selection

There are more kinds of **plants** and animals in the rainforests than anywhere else on Earth. Over half of the millions of known species of **plants** and animals live in the rainforest. Many are found nowhere else. There are even **plants** and animals in the rainforest that we have not yet discovered.

Biological Example

The Paulus company was founded in 1938. Since those days the product range has been the subject of constant expansions and is brought up continuously to correspond with the state of the art. We're engineering, manufacturing and commissioning world-wide ready-to-run **plants** packed with our comprehensive know-how. Our Product Range includes pneumatic conveying systems for carbon, carbide, sand, lime and many others. We use reagent injection in molten metal for the…

Industrial Example

Label the First Use of "Plant"

Roadmap

- Lexical Semantics
  - Motivation & Definitions
  - Word Senses
  - Tasks:
    - **Word sense disambiguation**
    - Word sense similarity
  - Distributional Similarity

Disambiguation: Features

- Part of Speech
  - Of word and neighbors

Disambiguation: Features

- Part of Speech
  - Of word and neighbors
- Morphologically simplified form

Disambiguation: Features

- Part of Speech
  - Of word and neighbors
- Morphologically simplified form
- Words in neighborhood
  - How big is “neighborhood?”
  - Is there a single optimal size? Why?

Disambiguation: Features

- (Possibly shallow) Syntactic analysis
  - predicate-argument relations
  - modification (complements)
  - phrases

Disambiguation: Features

- (Possibly shallow) Syntactic analysis
  - predicate-argument relations
  - modification (complements)
  - phrases
- Collocation
  - words in specific relation
    - Predicate-Argument, or (+/–)1 word index

Disambiguation: Features

- (Possibly shallow) Syntactic analysis
  - predicate-argument relations
  - modification (complements)
  - phrases
- Collocation
  - words in specific relation
    - Predicate-Argument, or (+/–)1 word index
- Co-occurrence
  - bag of words

Disambiguation: Evaluation

- Ideally, end-to-end evaluation with WSD component
  - Demonstrate real impact of technique in system
  - Difficult, expensive, still application specific

Disambiguation: Evaluation

- Ideally, end-to-end evaluation with WSD component
  - Demonstrate real impact of technique in system
  - Difficult, expensive, still application specific
- Typically intrinsic, sense-based
  - Accuracy, precision, recall
  - SENSEVAL/SEMEVAL: all words, lexical sample

WSD Evaluation

- Baseline:
  - Most frequent sense

WSD Evaluation

- Baseline:
  - Most frequent sense
- Ceiling:
  - Human inter-rater agreement
    - 75-80% fine
    - 90% coarse

Roadmap

- Lexical Semantics
  - Motivation & Definitions
  - Word Senses
  - Tasks:
    - Word sense disambiguation
    - **Word sense similarity**
  - Distributional Similarity

Word Sense Similarity

Word Sense Similarity

- Synonymy:
  - True propositional substitutability is rare, slippery

Word Sense Similarity

- Synonymy:
  - True propositional substitutability is rare, slippery

- Word similarity (semantic distance)
  - Looser notion, more flexible

Word Sense Similarity

- Appropriate to applications:
  - IR, summarization, MT, essay scoring
    - Don’t need binary +/– synonym decision
    - Want terms/documents that have high ***similarity***

Word Sense Similarity

- Appropriate to applications:
  - IR, summarization, MT, essay scoring
    - Don’t need binary +/– synonym decision
    - Want terms/documents that have high ***similarity***
- Approaches:
  - Distributional
  - Thesaurus-based

Similarity vs. Relatedness

Similarity vs. Relatedness

- Similarity:
  - *car, bicycle*
  - *nickel < coin < currency*

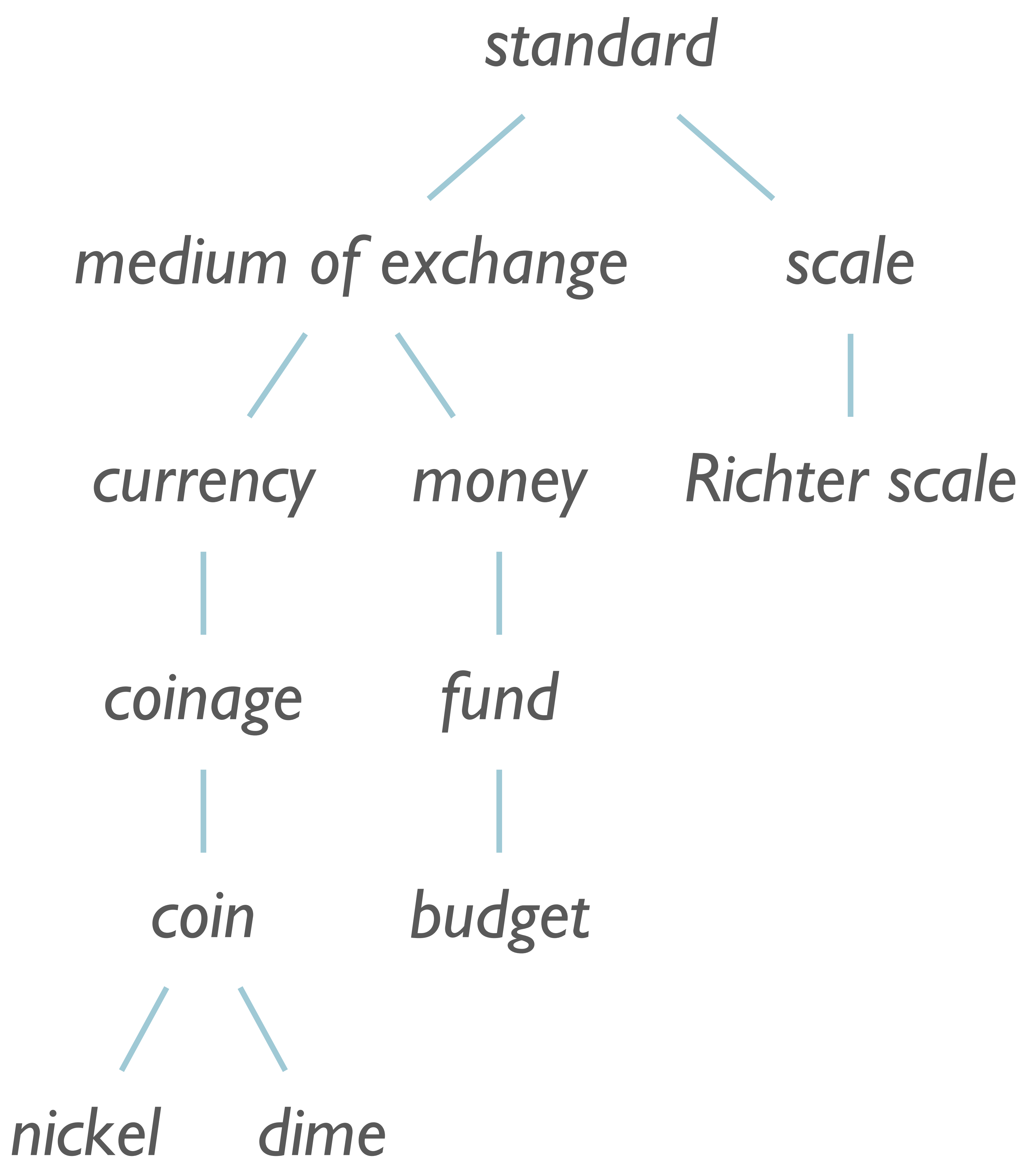
Similarity vs. Relatedness

- Similarity:
  - *car, bicycle*
  - *nickel < coin < currency*

- Related:
  - *car, gasoline*
  - *coin, budget*

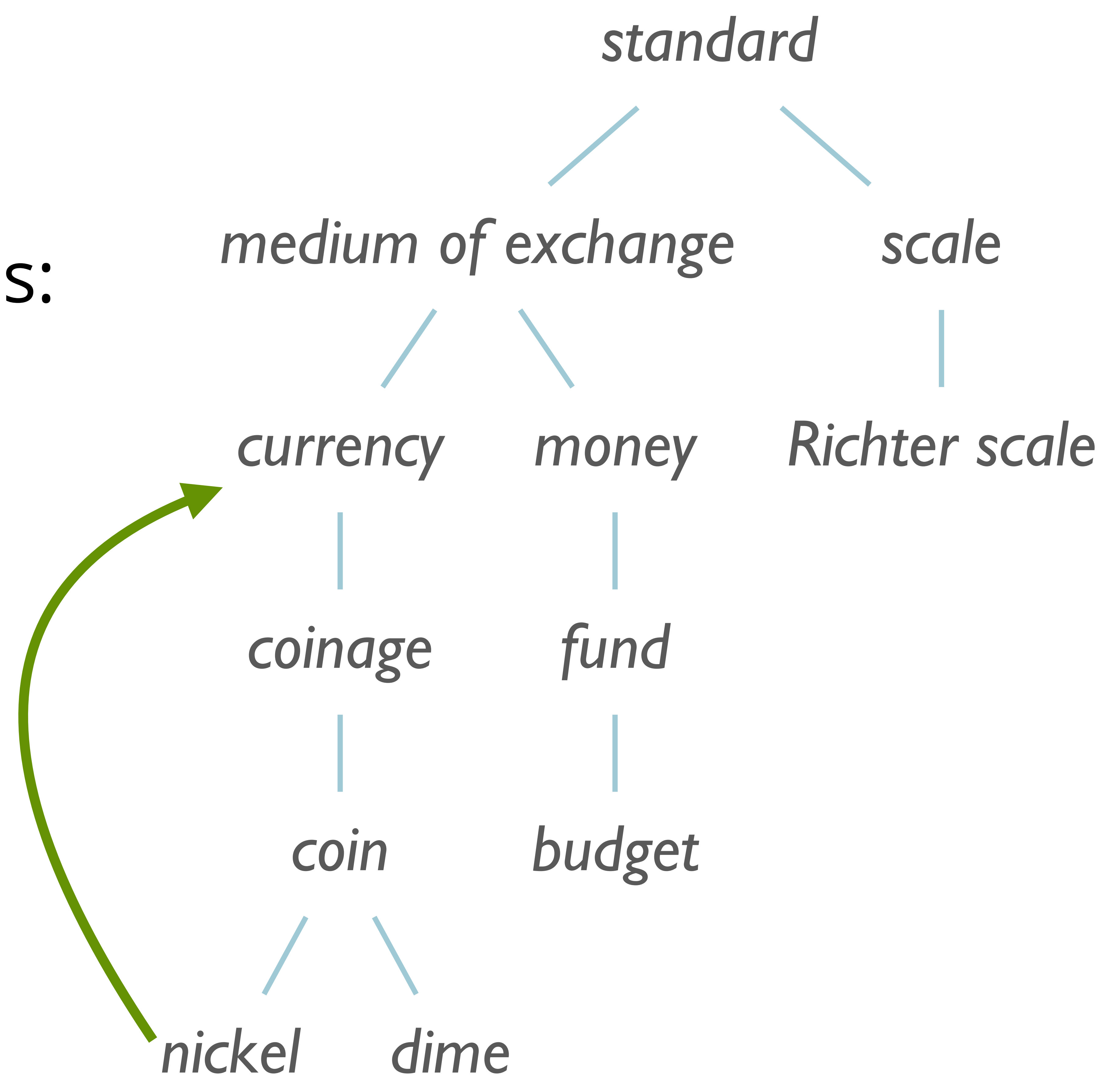
Thesaurus-Based:

- Build ontology of senses
  - e.g. WordNet
  - Use distance to infer similarity/relatedness:



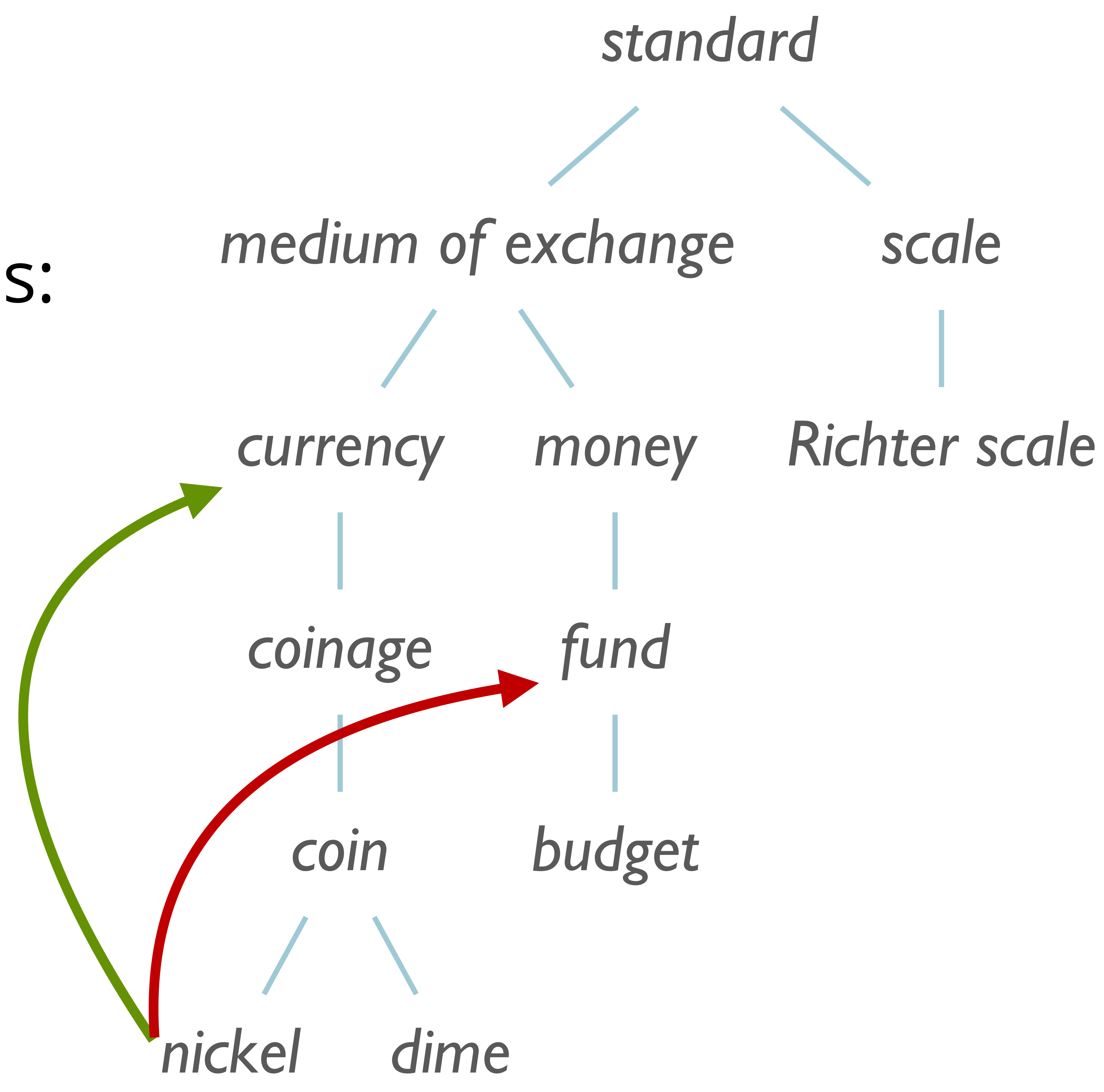
Thesaurus-Based:

- Build ontology of senses
  - e.g. [WordNet]
  - Use distance to infer similarity/relatedness:



Thesaurus-Based:

- Build ontology of senses
  - e.g. [WordNet]
  - Use distance to infer similarity/relatedness:



Roadmap

- Lexical Semantics
  - Motivation & Definitions
  - Word Senses
  - Tasks:
    - Word sense disambiguation
    - Word sense similarity
  - **Distributional Similarity**

Distributional Similarity

- "You shall know a word by the company it keeps!" (*Firth, 1957*)

Distributional Similarity

- "You shall know a word by the company it keeps!" (Firth, 1957)
  - A bottle of *tezgüino* is on the table.

Distributional Similarity

- "You shall know a word by the company it keeps!" (*Firth, 1957*)
  - A bottle of *tezgüino* is on the table.
  - Everybody likes *tezgüino*.

Distributional Similarity

- "You shall know a word by the company it keeps!" ([Firth, 1957])
  - A bottle of *tezgüino* is on the table.
  - Everybody likes *tezgüino*.
  - *Tezgüino* makes you drunk.

Distributional Similarity

- "You shall know a word by the company it keeps!" (*Firth, 1957*)
  - A bottle of *tezgüino* is on the table.
  - Everybody likes *tezgüino*.
  - *Tezgüino* makes you drunk.
  - We make *tezgüino* from corn.

Distributional Similarity

- "You shall know a word by the company it keeps!" (*Firth, 1957*)
  - A bottle of *tezgüino* is on the table.
  - Everybody likes *tezgüino*.
  - *Tezgüino* makes you drunk.
  - We make *tezgüino* from corn.
- Tezguino: corn-based alcoholic beverage. (From *Lin, 1998a*)

Distributional Similarity

- Represent ‘company’ of word such that similar words will have similar representations
  - ‘Company’ = context

Distributional Similarity

- Represent ‘company’ of word such that similar words will have similar representations
  - ‘Company’ = context
- Word represented by context feature vector
  - Many alternatives for vector

Distributional Similarity

- Represent 'company' of word such that similar words will have similar representations
  - 'Company' = context
- Word represented by context feature vector
  - Many alternatives for vector
- Initial representation:
  - 'Bag of words' binary feature vector
  - Feature vector length N, where N is size of vocabulary
    - f_i=1 if $word_i$ within window size w of $word_0$

Context Feature Vector

	arts	boil	data	function	large	sugar	summarized	water
Apricot	0	1	0	0	1	1	0	1
Pineapple	0	1	0	0	1	1	0	1
Digital	0	0	1	1	1	0	1	0
Information	0	0	1	1	1	0	1	0

Distributional Similarity Questions

- What is the right neighborhood?
  - What is the context?
- How should we weight the features?
- How can we compute similarity between vectors?