Semantic Representations and Parsing

CSCI 1460: Computational Linguistics

Lecture 18

Ellie Pavlick
Fall 2022

Announcements

- Final Project!
- Questions?

Topics

- “Semantic” Tasks: Executable Forms and NLI
- Formal Semantics
- Syntax-Semantics Interface and CCG
- Semantics and Deep Learning

Topics

- **“Semantic” Tasks: Executable Forms and NLI**
- Formal Semantics
- Syntax-Semantics Interface and CCG
- Semantics and Deep Learning

Semantics

- Syntax = the study of ***form***
  - Why is “the cat ran past the dog” a good English sentence but “ran cat dog the the past” not?
- Semantics = the study of ***meaning***
  - Why can’t I say “the cat ran past the dog” in order to mean that the dog ran past the cat?
- General goal is to describe precisely how we get from form to meaning

Semantics

- Generally:
  - Semantics = “sentence meaning” = tied to form/grammar
  - Pragmatics = “speaker meaning” = tied more generally to context
    - A: “Wanna go for a run?” B: “I’m tired.”
- In linguistics, these are kept separate
- In NLP, we tend to blur the distinction, and focus on specific tasks (e.g., “Alexa, I can’t hear the music” should be received as an instruction)

Tasks

- Arguably, (basically) all NLP tasks require representing “meaning”
  - BOW classifiers, pretrained language models, machine translation
- But there are two types of tasks which we generally focus on when we talk about “semantics”
  - Tasks that require **executable (or logical) form**
  - Reasoning about **entailment**

Executable (aka Logical) Form

- Explicit representation of natural language in formal language
- Question Answering over Databases:
  - “Show flights to Denver on Monday” -> `SELECT * FROM flights WHERE city = “Denver” and day = “Monday”`
- Robotics:
  - “Move forward past the sofa” -> `ƛmove(a) ∧ dir(a, forward) ∧ past(a, Iy.sofa(y))`
- Digital Personal Assistants:
  - “wake me up at 7” -> `set_alarm(07:00, GMT-5)`

Natural Language Inference (NLI)
(aka Recognizing Textual Entailment, or RTE)

- Given a premise p and a hypothesis h, does p *entail* h
- E.g.,
  - Between March and June, scientific observers say, up to 300,000 seals are killed. In Canada, seal-hunting means jobs, but opponents say it is vicious and endangers the species, also threatened by global warming -> Hunting endangers seal species. (Dagan, 2006)
  - At 8:34, the Boston Center controller received a third transmission from American 11 -> At 8:34, the Boston Center controller received a third transmission from American 11 (Williams et al, 2018)

Natural Language Inference (NLI)

(aka Recognizing Textual Entailment, or RTE)

- Assumed to be a subtask required for many other tasks
  - Question answering, summarization, information retrieval
- Now widely used as a general-purpose evaluation task for systems of “understanding”
- The field has…mixed feelings…about its usefulness as a task

Topics

- “Semantic” Tasks: Executable Forms and NLI
- **Formal Semantics**
- Syntax-Semantics Interface and CCG
- Semantics and Deep Learning

NLP: A brief history

NLP: A brief history

Formal Semantics

The basic aim of semantics is to characterize *the notion of a true sentence* (under a given interpretation) and of *entailment*.

(Richard Montague)

Formal Semantics

There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed I consider it possible to comprehend the syntax and semantics of both kinds of languages with a single natural and mathematically precise theory.

(Richard Montague)

Formal Semantics

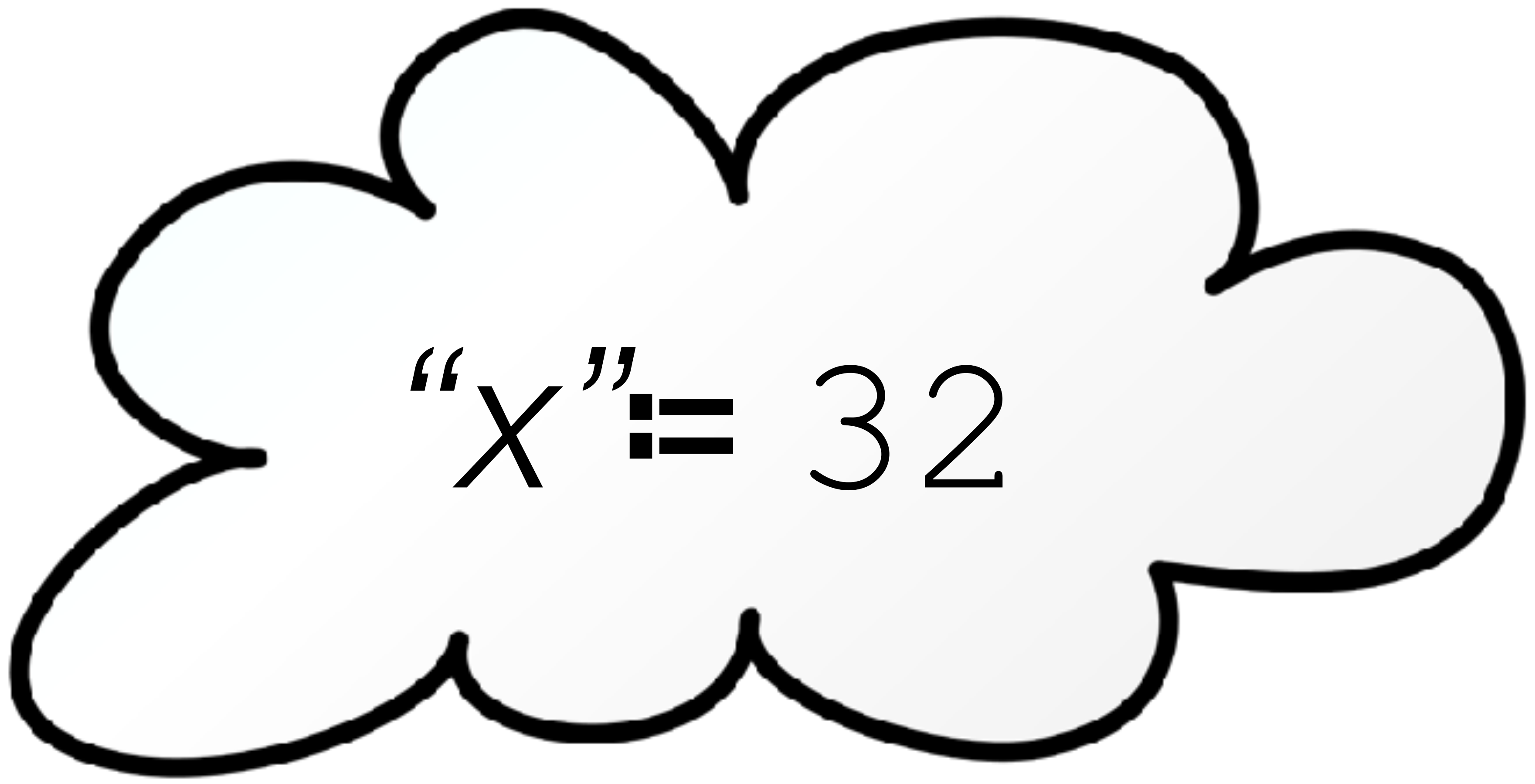
Model Theory 101

$$\text{“}x > 17\text{”}$$

Formal Semantics

Model Theory 101

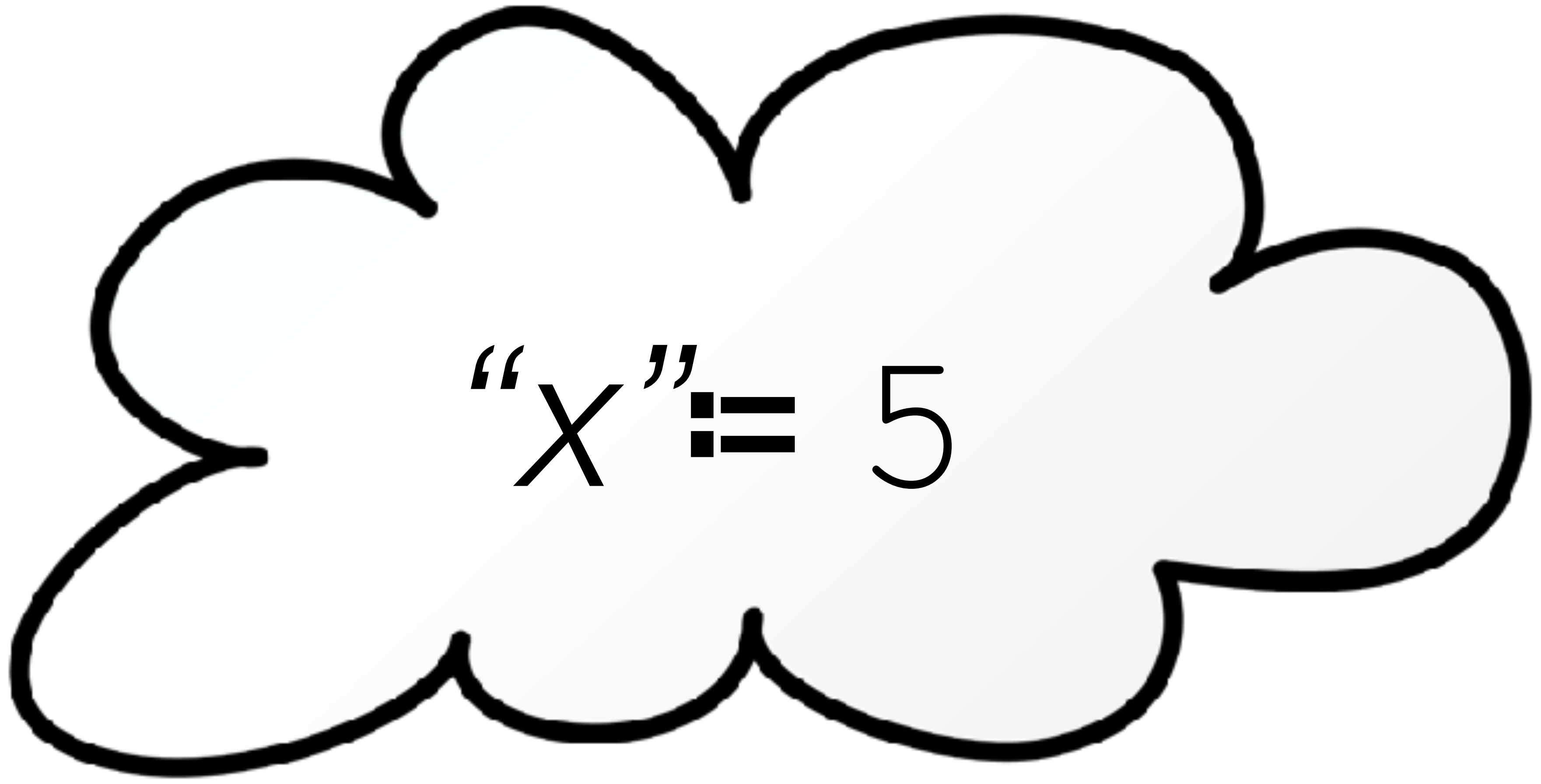
“$x > 17$” ✔



Formal Semantics

Model Theory 101

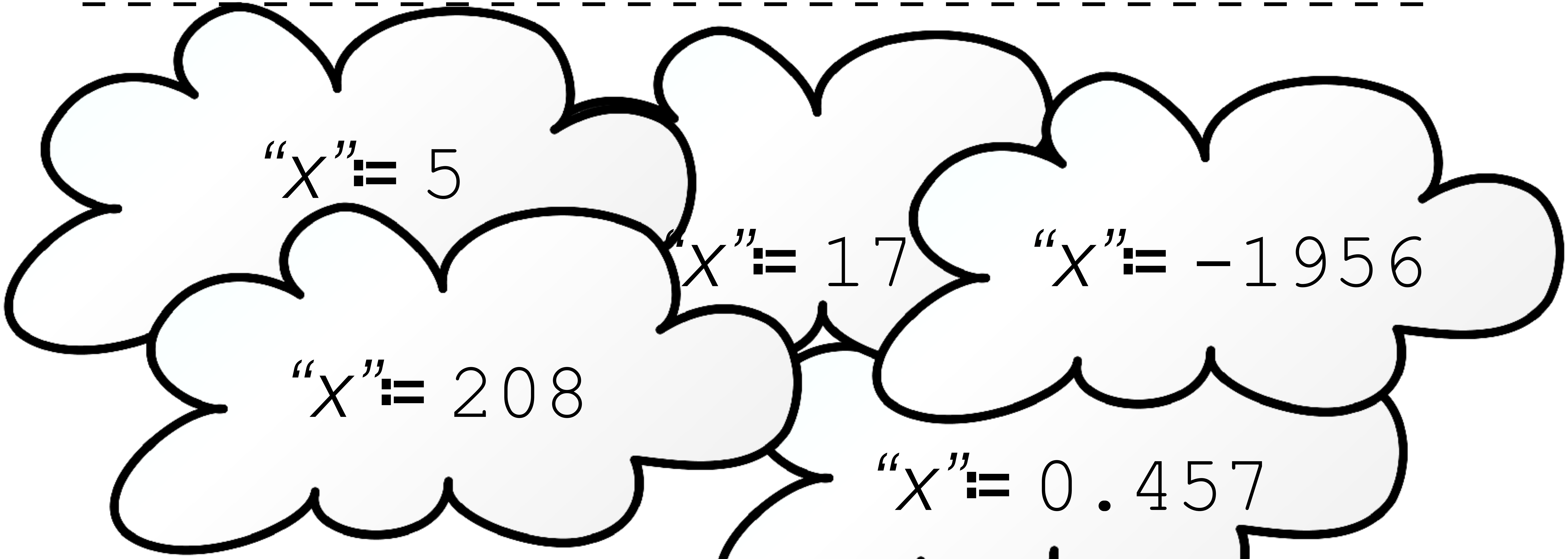
“$x > 17$” ✗



Formal Semantics

Model Theory 101

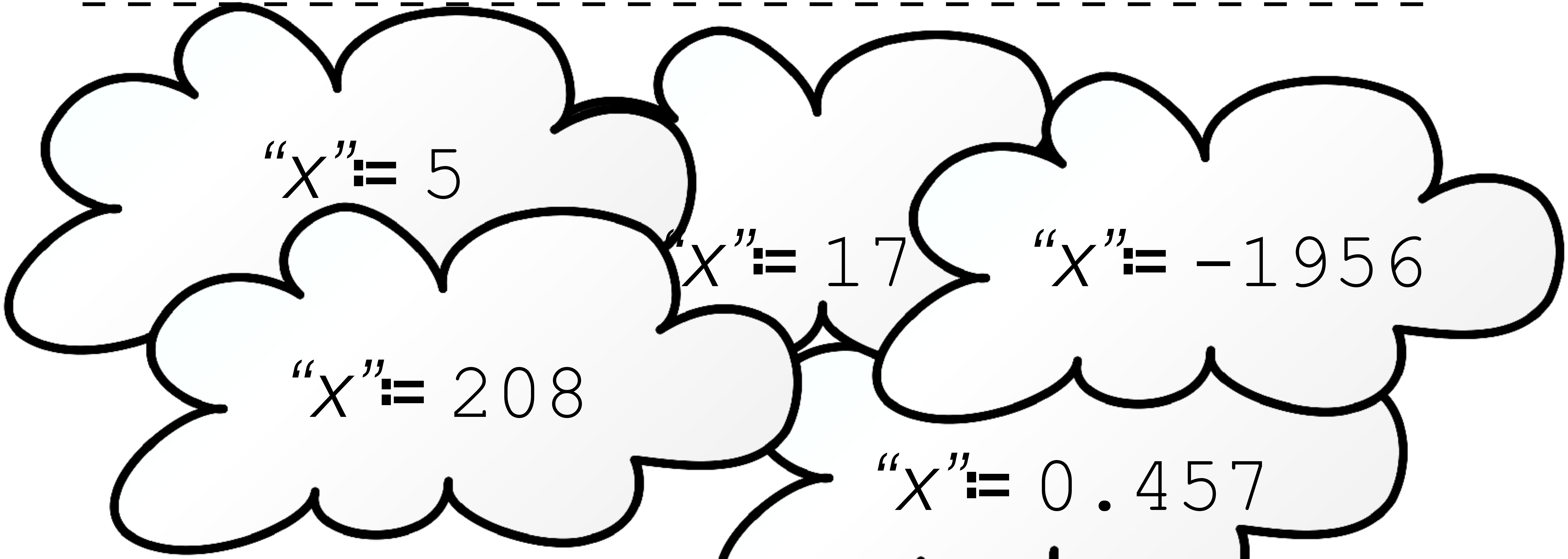
“$x > 17$”



Formal Semantics

Model Theory 101

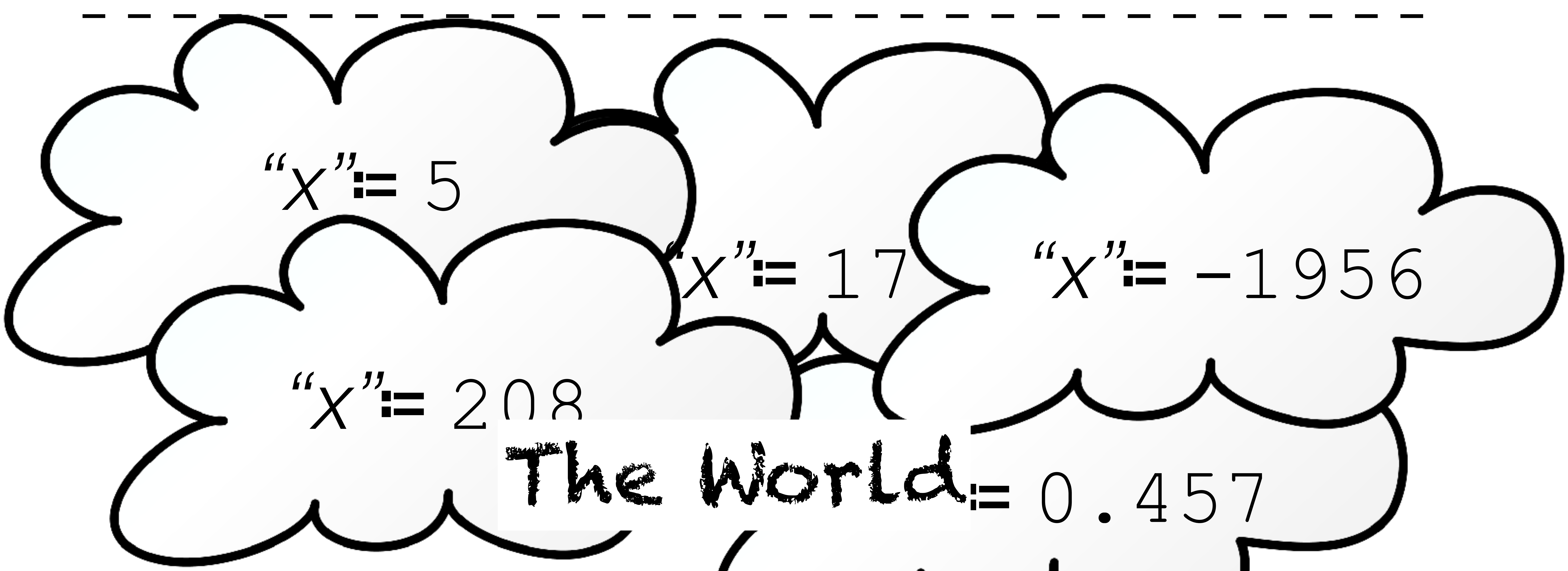
Language “x > 17”



Formal Semantics

Model Theory 101

Language “x > 17”



Formal Semantics

Model Theory 101

Language “x > 17”

Formal Semantics

Model Theory 101

Language

$$\frac{\begin{array}{c} x > y \\ y > z \end{array}}{x > z}$$

The World (TBD)

Formal Semantics

Model Theory 101

Language

Variables
(to be grounded)

$$\frac{x > y \qquad y > z}{x > z}$$

The World (TBD)

Formal Semantics

Model Theory 101

Language

$$\frac{x > y \qquad y > z}{x > z}$$

Relations (defined)

The World (TBD)

Formal Semantics

Model Theory 101

A premise (p) **entails** a hypothesis (h) iff,
in every possible world in which p is true,
h is also true.

$$\forall \mathcal{I}((\mathcal{I} \models p) \Rightarrow (\mathcal{I} \models h))$$

Formal Semantics

Model Theory 101

$$\frac{\begin{array}{c} x > y \\ z > w \end{array}}{x > w}$$

$x = 10 \quad y = 5 \quad z = 11 \quad w = 8$

Formal Semantics

Model Theory 101

$$\frac{\begin{array}{c} x > y \\ z > w \end{array}}{x > w}$$

x = 10   y = 5   z = 11   w = 8

Formal Semantics

Model Theory 101

$$\frac{\begin{array}{c} x > y \\ z > w \end{array}}{x > w}$$

$x = 10$ $y = 5$ $z = 11$ $w = 8$

Formal Semantics

Model Theory 101

$$\frac{\begin{array}{c} x > y \\ z > w \end{array}}{x > w}$$

✔

- -

$x = 10 \quad y = 5 \quad z = 11 \quad w = 8$

Formal Semantics

Model Theory 101

$$\frac{\begin{array}{c} x > y \\ z > w \end{array}}{x > w}$$

✘

- -

$x = 6 \quad y = 5 \quad z = 11 \quad w = 8$

Formal Semantics

Model Theory 101

the notion of a true sentence
(under a given interpretation)

"Broca is a bird"

Formal Semantics

Model Theory 101

the notion of a true sentence
(under a given interpretation)

"Broca is a bird" ✔

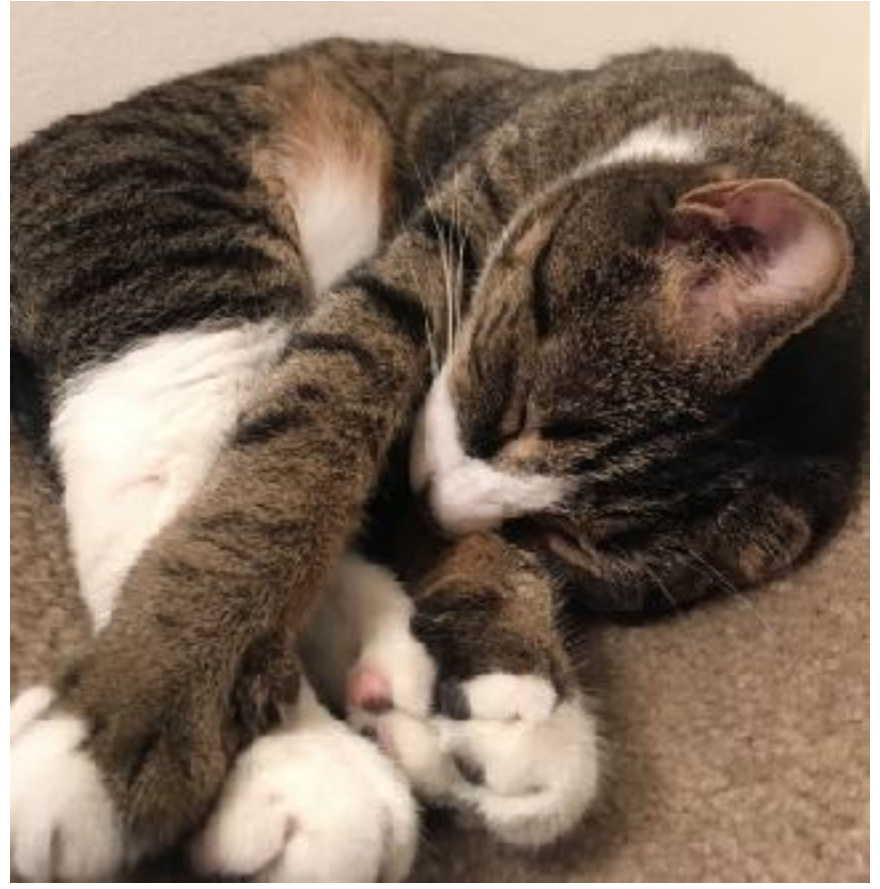
Broca

Formal Semantics

Model Theory 101

the notion of a true sentence
(under a given interpretation)

"Broca is a bird" ✘



Broca

Formal Semantics

Model Theory 101

the notion of entailment

"All birds are gray"

"Broca is a bird"

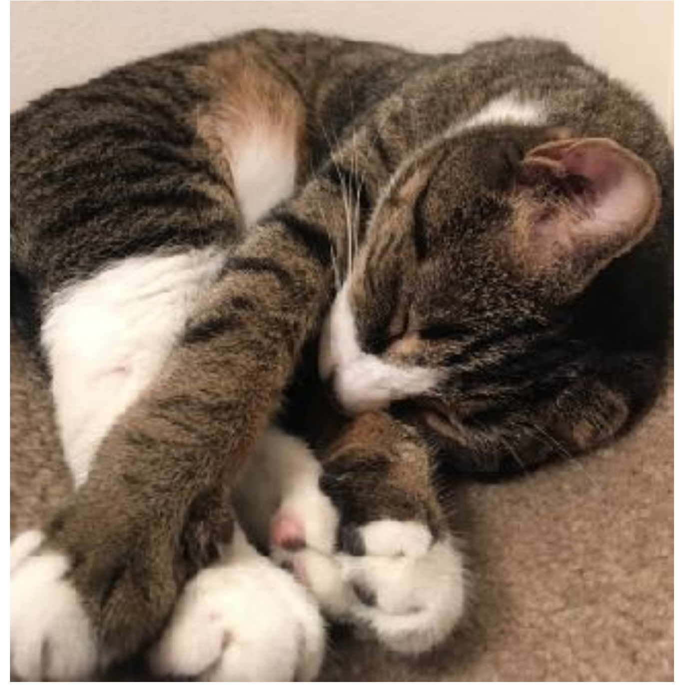
"Broca is gray" ✔

Formal Semantics

Truth Conditions and Truth Values

“Broca is a bird”

✘



Broca

✘

Broca

✔

Broca

✔

Broca

✘

Broca

Formal Semantics

Truth Conditions and Truth Values

Truth conditions specify what the world needs to be like for the sentence to be true

"Broca is a bird"

✗

Broca

✗

Broca

✓

Broca

✓

Broca

✗

Broca

Formal Semantics

Truth Conditions and Truth Values

Truth value says whether or not a sentence is true (give some specific state of the world)

“Broca is a bird”

Broca

Formal Semantics

Truth Conditions and Truth Values

- Truth Conditional (or “Intentional”) Semantics: the meaning of a sentence is its truth *conditions*
  - understanding the meaning of “I have a bag of potatoes in my cupboard” does not require knowing whether I have potatoes in my cupboard
- Contrast with Denotational Semantics: the meaning of a sentence is its truth *value*
  - i.e., “My mom’s name is Karin” and “3 is half of 6” mean the same thing
  - …?
- Formal semantics uses *truth conditional*

Formal Semantics

“The Fregean Program”

- Goal is to give an unambiguous account of the mapping for form to meaning
  - Input: A (syntactically parsed) string of words
  - Output: A context-independent logical form (e.g., lambda calculus, first order logic, etc)
- This program is complex and very rich—take CLPS 1342 (and 1341!)

Formal Semantics

Semantic Types

Formal Semantics

Semantic Types

- Want language to be a (formal) abstraction over “the world”, so only two primitive types:
  - e is the semantic type of entities
  - t is the semantic type of truth values
- Other types are defined recursively:
  - If a is a semantic type and b is a semantic type, then <a, b> is a semantic type

The Fregean Program

Semantic Types

- Referring expressions are type e: “Broca”, “Eddy”, “Ty”, “the cat on the mat”* …
- Propositional Sentences are type t: “Eddy is a cat”
- 1-place Predicates (adjectives, common nouns) are type <e, t>:
- $[\![\text{cat}]\!] =$

The Fregean Program

Semantic Types

- Referring expressions are type e: “Broca”, “Eddy”, “Ty”, “the cat on the mat”* …
- Propositional Sentences are type t: “Eddy is a cat”
- 1-place Predicates (adjectives, common nouns) are type <e, t>:
- $[\![\text{cat}]\!] = \begin{bmatrix} \text{Broca} \rightarrow 0 \\ \text{Eddy} \rightarrow 1 \\ \text{Ty} \rightarrow 1 \end{bmatrix}$

The Fregean Program

Semantic Types

- Referring expressions are type e: “Broca”, “Eddy”, “Ty”, “the cat on the mat”* …
- Propositional Sentences are type t: “Eddy is a cat”
- 1-place Predicates (adjectives, common nouns) are type <e, t>:
- $[\![\text{cat}]\!] = \begin{bmatrix} \text{Broca} \rightarrow 0 \\ \text{Eddy} \rightarrow 1 \\ \text{Ty} \rightarrow 1 \end{bmatrix}$

(Often thought of as sets,
i.e. the “characteristic
function of a set”)

The Fregean Program

Semantic Types

- 2-place Predicates (transitive verbs) are ???

The Fregean Program

Semantic Types

- 2-place Predicates (transitive verbs) are type <e,<e,t>>:

takes entity as
input, returns
function (/set)

The Fregean Program

Semantic Types

- 2-place Predicates (transitive verbs) are type <e,<e,t>>:
- ⟦likes⟧ =

$$\left[\begin{array}{rl}
\text{Broca} \rightarrow & \left[\begin{array}{r}\text{Broca} \rightarrow 1\\ \text{Eddy} \rightarrow 0\\ \text{Ty} \rightarrow 0\end{array}\right]\\[2ex]
\text{Ty} \rightarrow & \left[\begin{array}{r}\text{Broca} \rightarrow 1\\ \text{Eddy} \rightarrow 1\\ \text{Ty} \rightarrow 1\end{array}\right]\\[2ex]
\text{Eddy} \rightarrow & \left[\begin{array}{r}\text{Broca} \rightarrow 1\\ \text{Eddy} \rightarrow 1\\ \text{Ty} \rightarrow 0\end{array}\right]
\end{array}\right]$$

takes entity as input, returns function (/set)

The Fregean Program

Semantic Types

- Quantifiers (e.g. all, every) are ???

The Fregean Program

Semantic Types

- Quantifiers (e.g. all, every) are type <<e,t>, t>

takes a function
(/set) as input,
returns a truth value.
i.e., sets of sets

The Fregean Program

Semantic Types

- Quantifiers (e.g. all, every) are type <<e,t>, t>
- every cat sleeps: 1 if ∀ e (cat(e) -> sleeps(e)); else 0

The Fregean Program

Semantic Types

- Quantifiers (e.g. all, every) are type <<e,t>, t>
- ⟦every⟧(x)(y) = y → 1 if ∀ e (x(e) -> y(e)); else 0

The Fregean Program

Semantic Types

- Quantifiers (e.g. all, every) are type <<e,t>, t>
- $[\![\text{every}]\!] = \lambda f\ \lambda g.\ \forall\ x\ f(x) \to g(x)$

The Fregean Program

Semantic Types

- Quantifiers (e.g. all, every) are type <<e,t>, t>
- ⟦every⟧ = $\lambda f\ \lambda g.\ \forall\ x\ f(x) \rightarrow g(x)$

bound variable

The Fregean Program

Compositionality

“Principle of Compositionality”:
The meaning of the whole is a function of the meaning of the parts and the way in which they are combined.

The Fregean Program

Compositionality

Lexical Semantics

"Principle of Compositionality":
The meaning of the whole is a function of the **meaning of the parts** and the way in which they are combined.

The Fregean Program

Compositionality

Lexical Semantics

“Principle of Compositionality”: The meaning of the whole is a function of the **meaning of the parts** and **the way in which they are combined**.

Syntax

The Fregean Program

Compositionality

“Eddy likes Broca”

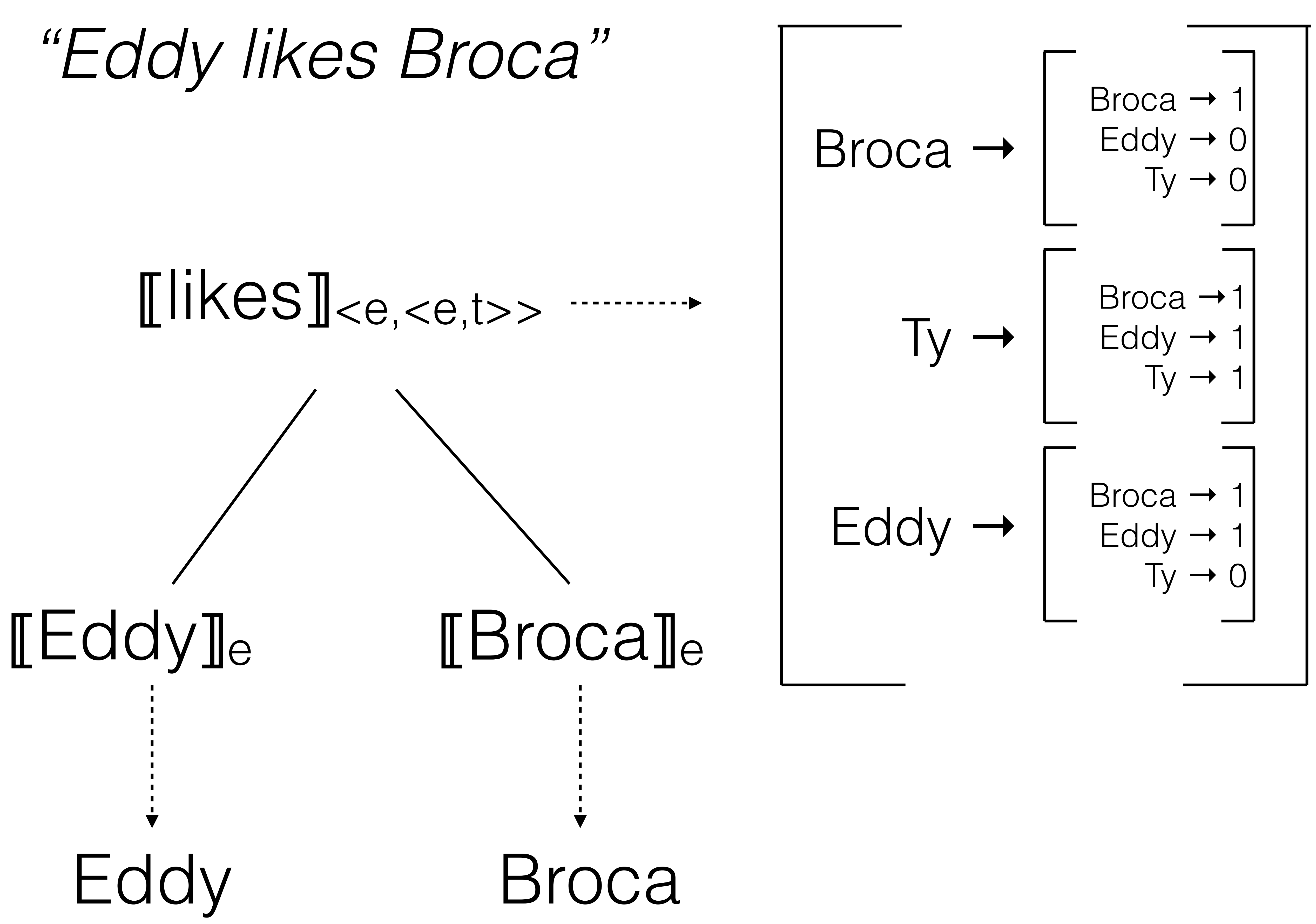
The Fregean Program
Compositionality

“Eddy likes Broca”

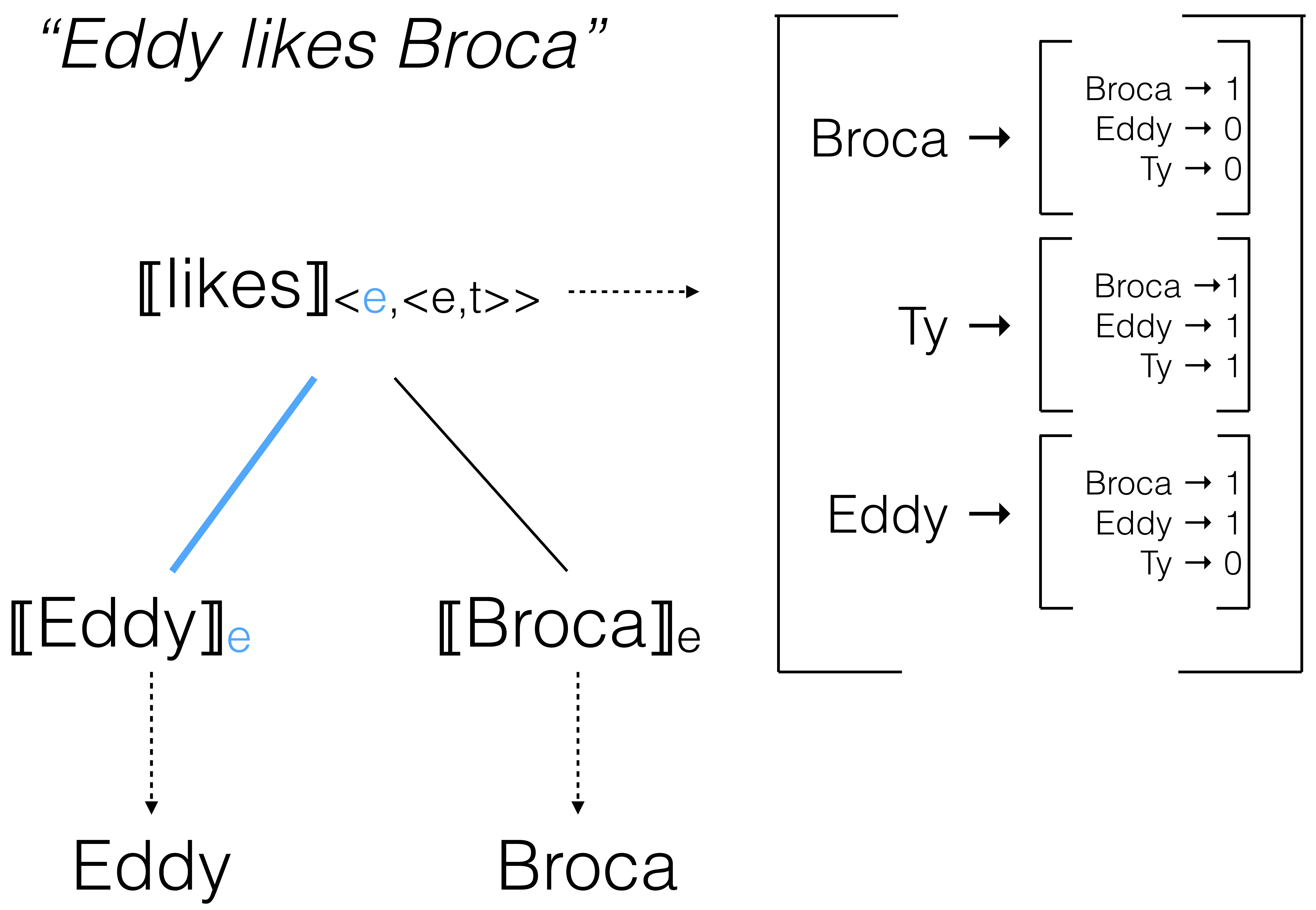
likes

Eddy Broca

The Fregean Program
Compositionality

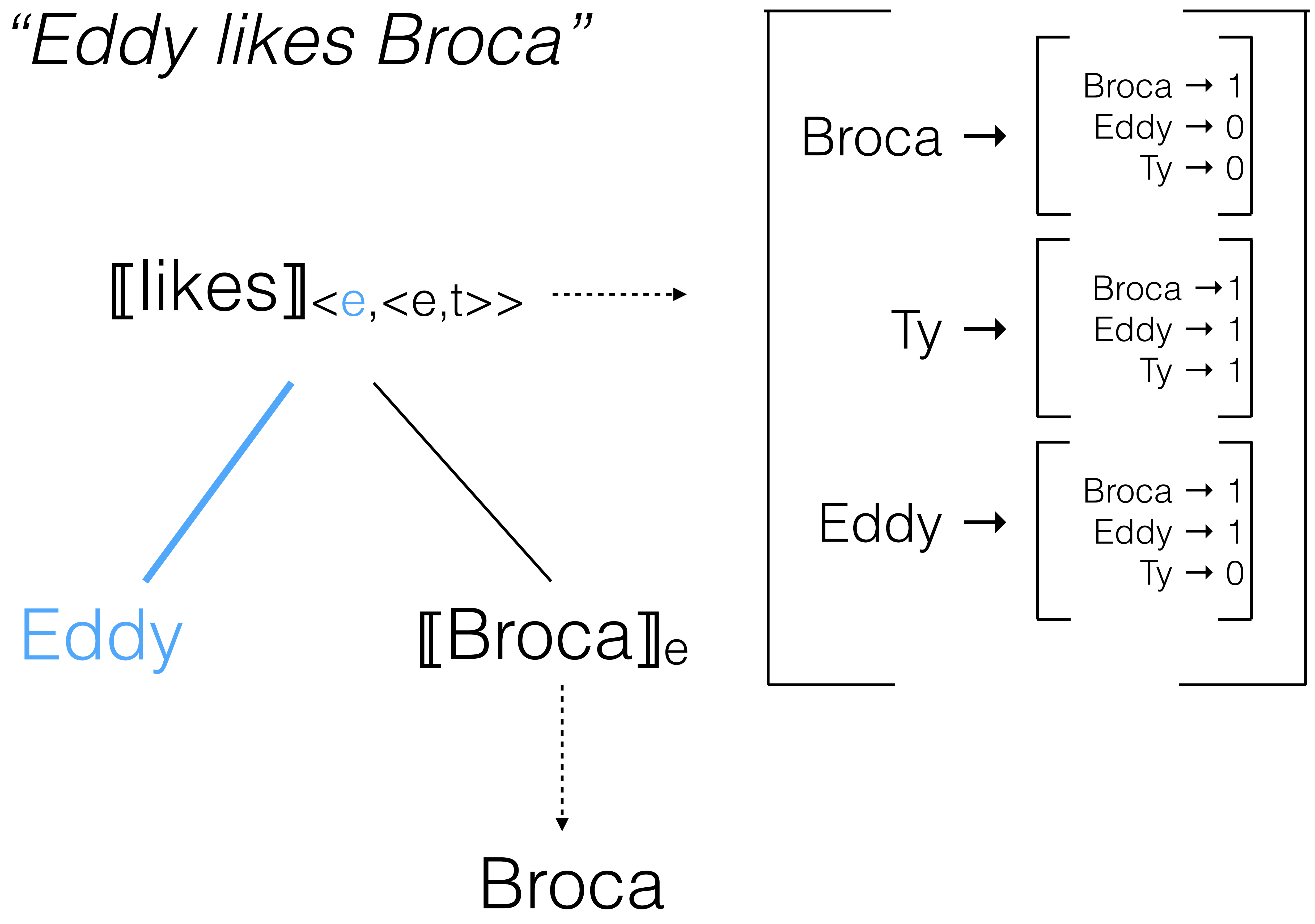


The Fregean Program
Compositionality



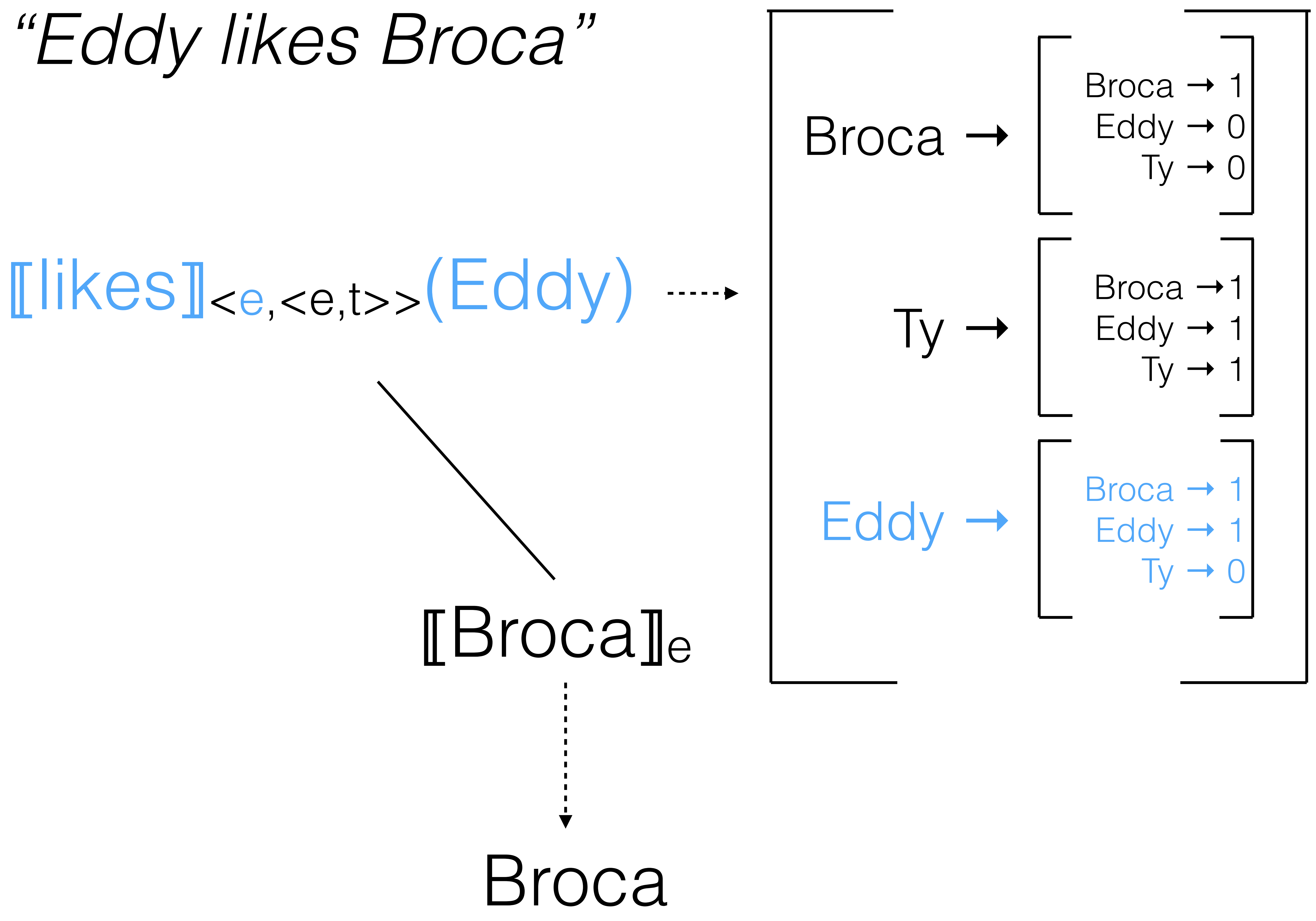
The Fregean Program

Compositionality



The Fregean Program

Compositionality



The Fregean Program

Compositionality

“Eddy likes Broca”

$$[\![\text{Eddy likes}]\!]_{<e,t>} \dashrightarrow \begin{bmatrix} \text{Broca} \rightarrow 1 \\ \text{Eddy} \rightarrow 1 \\ \text{Ty} \rightarrow 0 \end{bmatrix}$$

$$[\![\text{Broca}]\!]_{e}$$

$$\Big\downarrow$$

Broca

The Fregean Program

Compositionality

"Eddy likes Broca"

$$[\![\text{Eddy likes}]\!]_{<e,t>} \dashrightarrow \begin{bmatrix} \text{Broca} \to 1 \\ \text{Eddy} \to 1 \\ \text{Ty} \to 0 \end{bmatrix}$$

$$[\![\text{Broca}]\!]_{e}$$

$$\downarrow$$

Broca

The Fregean Program

Compositionality

“Eddy likes Broca”

$$[\![\text{Eddy likes}]\!]_{<e,t>} \dashrightarrow \begin{bmatrix} \text{Broca} \to 1 \\ \text{Eddy} \to 1 \\ \text{Ty} \to 0 \end{bmatrix}$$

Broca

The Fregean Program

Compositionality

"Eddy likes Broca"

$$[\![\text{Eddy likes}]\!]_{<e,t>}(\text{Broca}) \dashrightarrow \begin{bmatrix} \text{Broca} \rightarrow 1 \\ \text{Eddy} \rightarrow 1 \\ \text{Ty} \rightarrow 0 \end{bmatrix}$$

The Fregean Program

Compositionality

“Eddy likes Broca”

$[\![\text{Eddy likes Broca}]\!]_t \dashrightarrow 1$

Topics

- “Semantic” Tasks: Executable Forms and NLI
- Formal Semantics
- **Syntax-Semantics Interface and CCG**
- Semantics and Deep Learning

Combinatory Categorial Grammar (CCG)

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

CFG Phrase Structure Grammar

Combinatory Categorial Grammar (CCG)

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

CFG Phrase Structure Grammar

Combinatory Categorial Grammar (CCG)

"Eddy" `:=  NP : entity`

Combinatory Categorial Grammar (CCG)

"Eddy" `:=  NP : entity`

Syntax

Combinatory Categorial Grammar (CCG)

"Eddy" := `NP : entity`

Semantics

Combinatory Categorial Grammar (CCG)

“likes” := `(S\NP)/NP: λx.λy.likes(y,x)`

Combinatory Categorial Grammar (CCG)

"likes" $:=$ `(S\NP)/NP: λx.λy.likes(y,x)`

If I get an NP on my right

Combinatory Categorial Grammar (CCG)

"likes" $:= (\mathtt{S}\backslash\mathtt{NP})/\mathtt{NP}: \ \lambda x.\lambda y.\mathtt{likes}(y,x)$

If I get an NP on my right and another one on my left

Combinatory Categorial Grammar (CCG)

"*likes*" := $(S\backslash NP)/NP$: $\lambda x.\lambda y.\mathrm{likes}(y,x)$

If I get an NP on my right and another one on my left, I'll make a sentence.

Combinatory Categorial Grammar (CCG)

"*likes*" $:= (S\backslash NP)/NP: \ \lambda x.\lambda y.\mathtt{likes}(y,x)$

If I get an NP on my right and another one on my left, I'll make a sentence.

If I get an argument x and another argument y, I'll return some meaningful value.

Combinatory Categorial Grammar (CCG)

Principle of Combinatory Transparency

"likes" := (S\NP)/NP: λx.λy.likes(y,x)

If I get an NP on my right and another one on my left, I'll make a sentence.

If I get an argument x and another argument y, I'll return some meaningful value.

Combina… CCG)

Eddy → `NP: Eddy`

Broca → `NP: Broca`

likes → `(S\NP)/NP : λx.λy.likes(y,x)`

Eddy *likes* *Broca*

Combina… CCG)

Eddy → `NP: Eddy`

Broca → `NP: Broca`

likes → `(S\NP)/NP : λx.λy.likes(y,x)`

Eddy	*likes*	*Broca*
`NP`	`(S\NP)/NP`	`NP`
`Eddy`	`λx.λy.likes(y,x)`	`Broca`

Combina… CCG)

Eddy → `NP: Eddy`

Broca → `NP: Broca`

likes → `(S\NP)/NP : λx.λy.likes(y,x)`

Eddy	*likes*	*Broca*
`NP`	`(S\NP)/NP`	`NP`
`Eddy`	`λx.λy.likes(y,x)`	`Broca`

Combina … CCG)

Eddy → `NP: Eddy`

Broca → `NP: Broca`

likes → `(S\NP)/NP : λx.λy.likes(y,x)`

Eddy	*likes*	*Broca*
`NP`	`(S\NP)/NP`	`NP`
`Eddy`	`λx.λy.likes(y,x)`	`Broca`

`(S\NP)`

`λy.likes(y,Broca)`

Combina CCG)

Eddy → `NP: Eddy`

Broca → `NP: Broca`

likes → `(S\NP)/NP : λx.λy.likes(y,x)`

Eddy	*likes*	*Broca*
NP	(S\NP)/NP	NP
Eddy	λx.λy.likes(y,x)	Broca

(S\NP)

λy.likes(y,Broca)

Combina CCG)

Eddy → `NP: Eddy`

Broca → `NP: Broca`

likes → `(S\NP)/NP : λx.λy.likes(y,x)`

Eddy	*likes*	*Broca*
NP	(S\NP)/NP	NP
Eddy	λx.λy.likes(y,x)	Broca

(S\NP)

λy.likes(y,Broca)

S

likes(Eddy,Broca)

Topics

- “Semantic” Tasks: Executable Forms and NLI
- Formal Semantics
- Syntax-Semantics Interface and CCG
- **Semantics and Deep Learning**

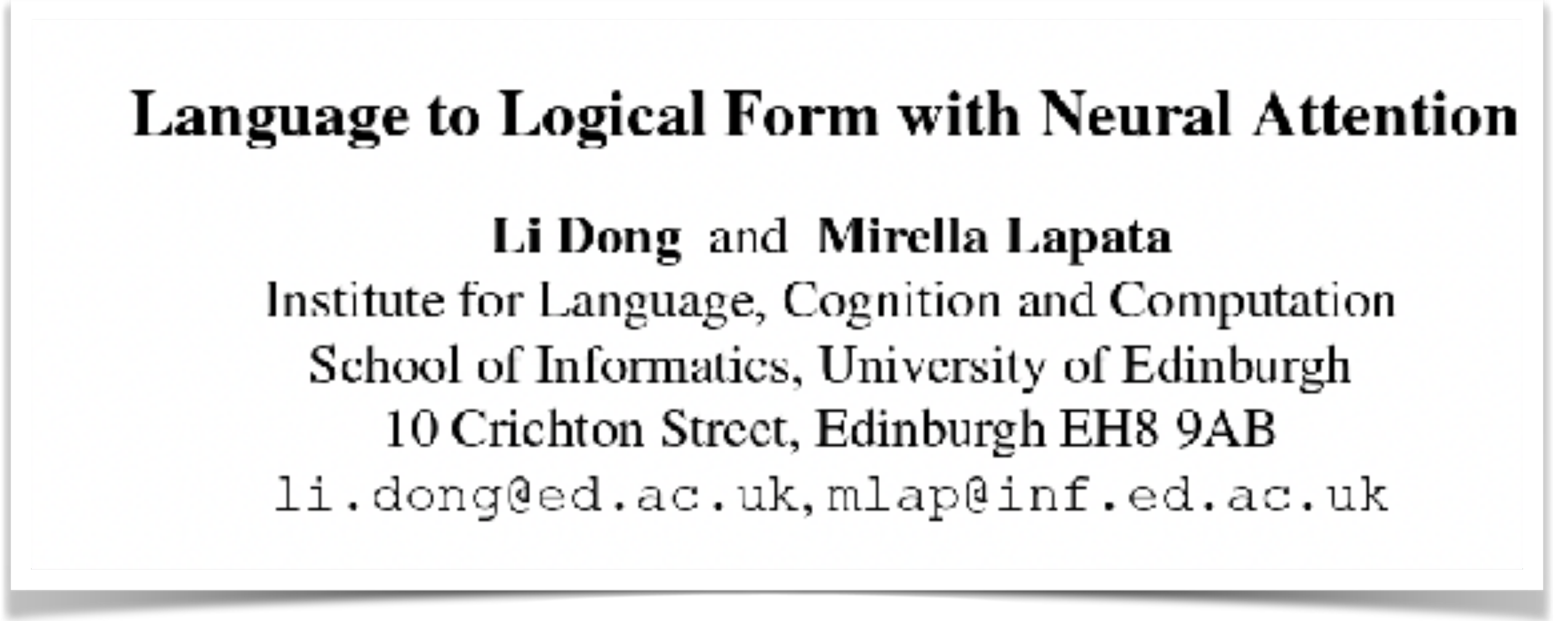
Semantics and Deep Learning

- Broadly, two types of work happening
  1. Use deep learning to make better semantic parsers
  2. Skip semantic parsing entirely, assume it will be captured "latently" by the network

Semantics and Deep Learning

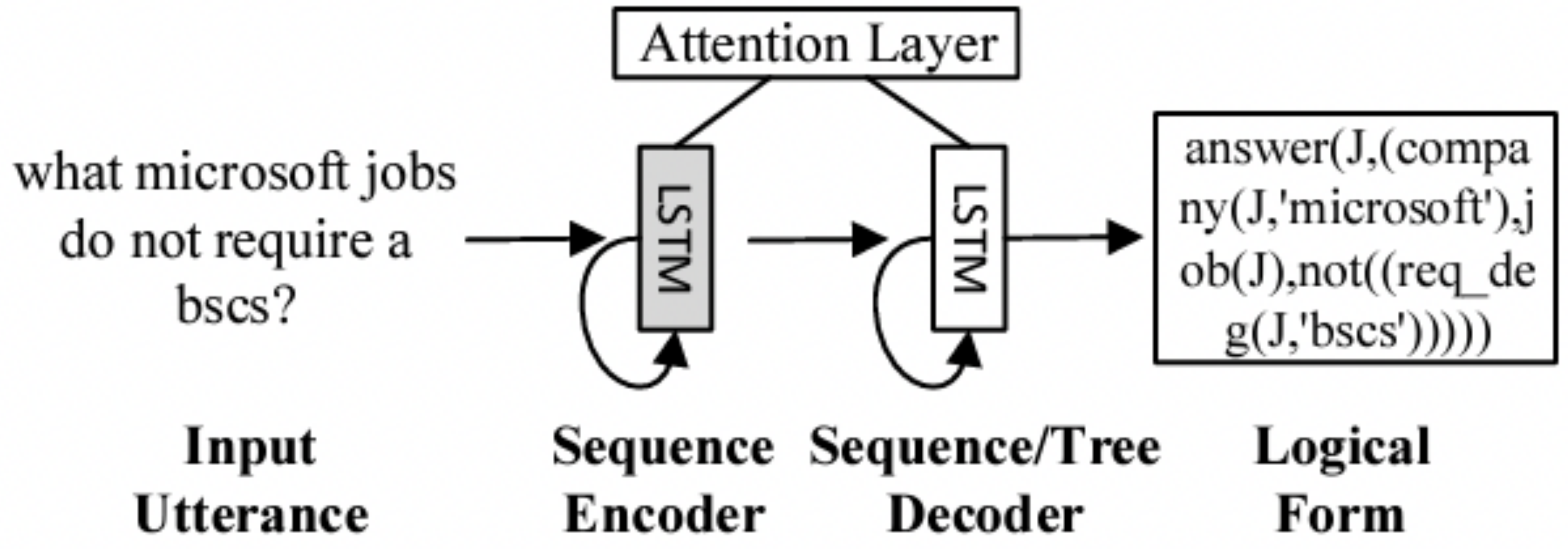
DL for better parsers

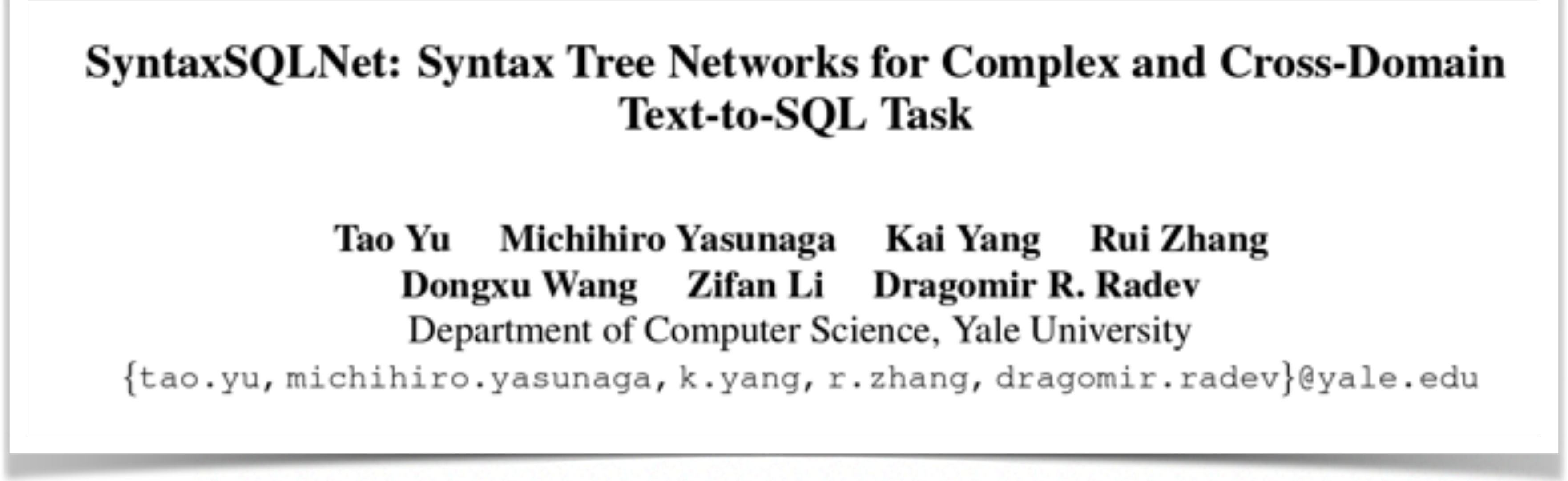
- E.g., treat English->Logical Form as a machine translation problem
- You can do this for your final project!



Language to Logical Form with Neural Attention

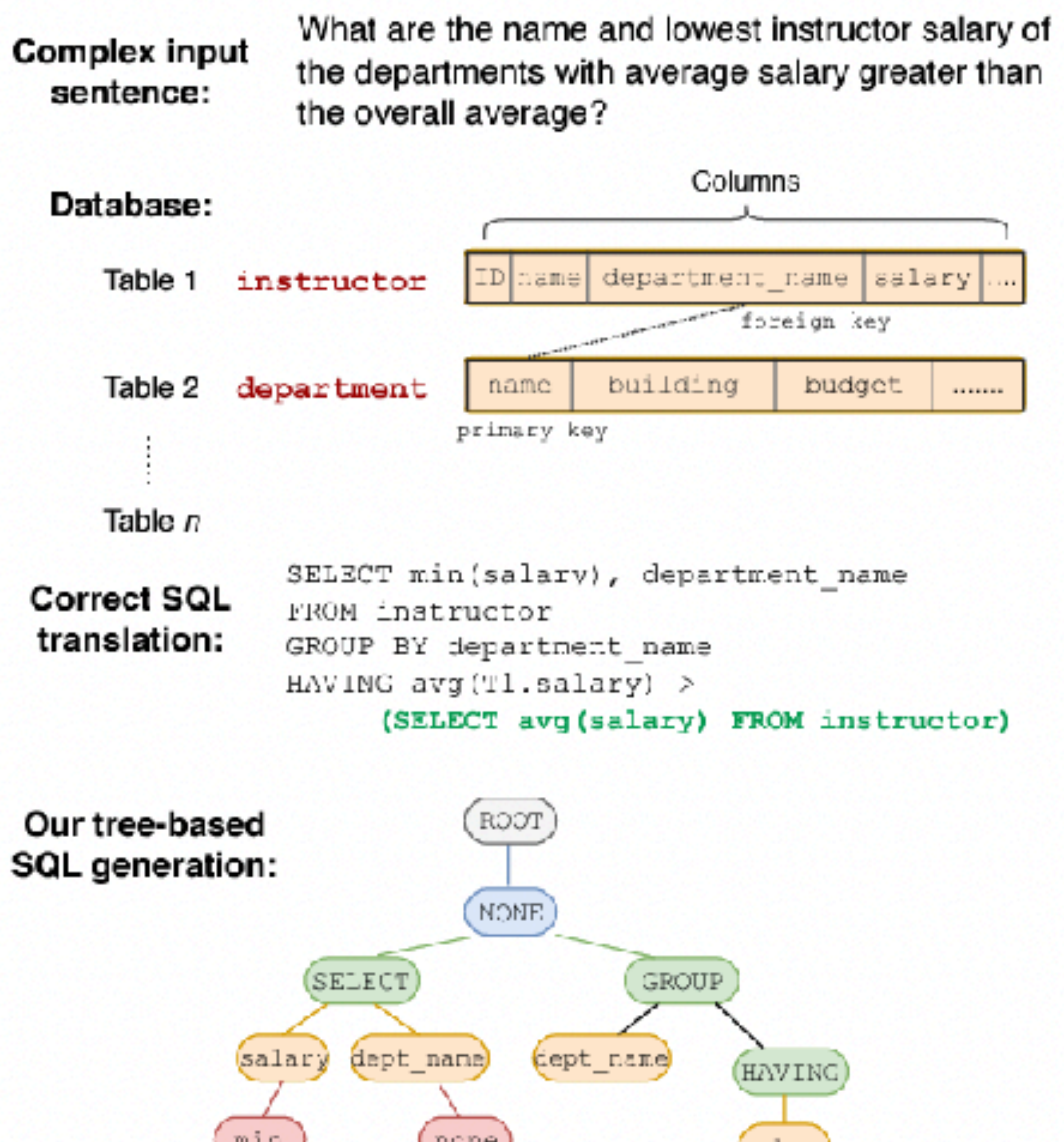
Li Dong and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
li.dong@ed.ac.uk, mlap@inf.ed.ac.uk





SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task

Tao Yu **Michihiro Yasunaga** **Kai Yang** **Rui Zhang**
Dongxu Wang **Zifan Li** **Dragomir R. Radev**
Department of Computer Science, Yale University
{tao.yu, michihiro.yasunaga, k.yang, r.zhang, dragomir.radev}@yale.edu



Semantics and Deep Learning

Treating Semantics as “Latent”

- Can we just train a model end-to-end to e.g., retrieve from a database or train a robot to navigate in an environment?
  - Raw tokens/perception in -> correct behavior out?
- Lots of debate! (E.g., https://compositionalintelligence.github.io/)
- Are NN’s “compositional”? Do they need to be to capture semantics?
- More next lecture!