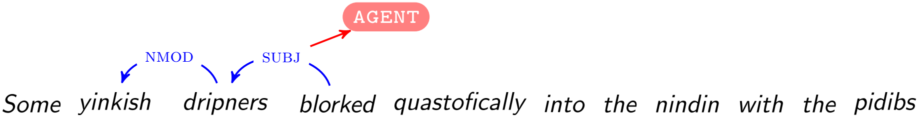
L90: Overview of Natural Language Processing

Lecture 9: Compositional Semantics

Weiwei Sun

Department of Computer Science and Technology
University of Cambridge

Michaelmas 2020/21



syntax provides scaffolding for semantic composition

Lecture 9: Compositional Semantics

1. Being able to transform
2. Semantic composition
3. Graph-based meaning representations
4. Inference and RTE

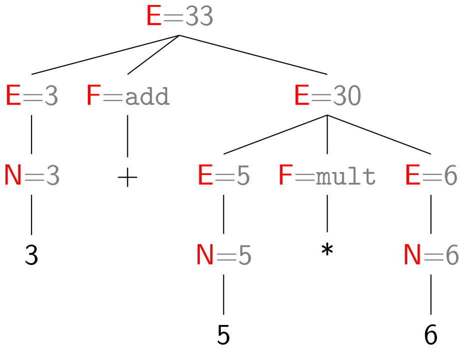
many slides are from Ann Copestake

Principle: Being Able to Transform

Programming language interpreter

What is meaning of $3 + 5 * 6$?

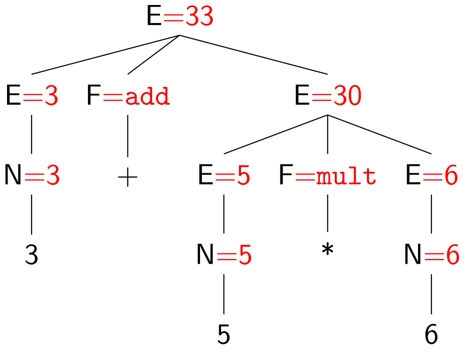
- First parse it into $3 + (5 * 6)$
- Now give a meaning to each node in the tree (bottom-up)



Programming language interpreter

What is meaning of $3 + 5 * 6$?

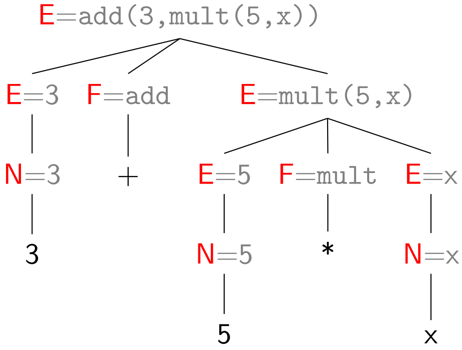
- First parse it into $3 + (5 * 6)$
- Now give a meaning to each node in the tree (bottom-up)



Interpreting in an environment

How about $3 + 5 * x$?

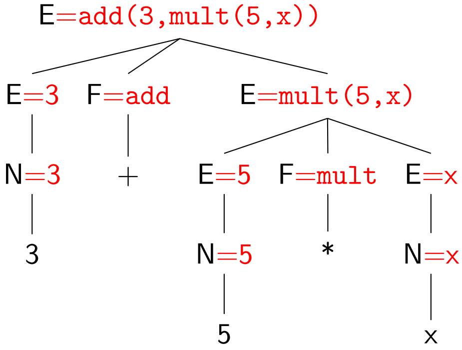
- Don't know x at compile time
- *Meaning* at a node is a piece of code, not a number



Interpreting in an environment

How about $3 + 5 * x$?

- Don't know x at compile time
- *Meaning* at a node is a piece of code, not a number



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 within a single natural and mathematically precise theory.

Richard Montague, 1930–1971

What counts as understanding?

Charaterizing what we mean by *meaning* is a difficult philosophical issue.

a compiler is a translator
(formal language to formal language)

What counts as understanding?

Charaterizing what we mean by *meaning* is a difficult philosophical issue.

a compiler is a translator (formal language to formal language) $\Downarrow$ Natural Language Understanding $\triangleright$ being able to translate

Natural language to natural language?

- Reasonable. Sometimes requires deeper understanding

Natural language to formal language (defined by logic)?

- Popular in NLP.

The general picture

$$
\begin{tikzcd}
\text{natural language:} \arrow[d, "\text{representation}"] & \text{sentences} & \text{sentence} \\
\text{formal language:} \arrow[d, dash, "\text{formal semantics}"] & \text{formulas} \arrow[r, "entail\ (\vdash)"] \arrow[d, "\text{truth conditions}"] & \text{formula} \arrow[d, "\text{truth conditions}"] \\
\text{models:} & \text{properties} \arrow[r, "follow\ (\vDash)"] & \text{property} \\
\text{real world:} \arrow[u, "\text{representation}"] & \text{facts} & \text{fact}
\end{tikzcd}
$$

from Yanjing Wang

The general picture

$$
\begin{tikzcd}
\text{natural language:} \arrow[d, "\text{representation}"] & \text{sentences} & & \text{sentence} \\
\text{formal language:} \arrow[d, dash, "\text{formal semantics}"] & \text{formulas} \arrow[rr, "entail\ (\vdash)"] \arrow[d, "\text{truth conditions}"] & & \text{formula} \arrow[d, "\text{truth conditions}"] \\
\text{models:} & \text{properties} \arrow[rr, "follow\ (\vDash)"] & & \text{property} \\
\text{real world:} \arrow[u, "\text{representation}"] & \text{facts} & & \text{fact}
\end{tikzcd}
$$

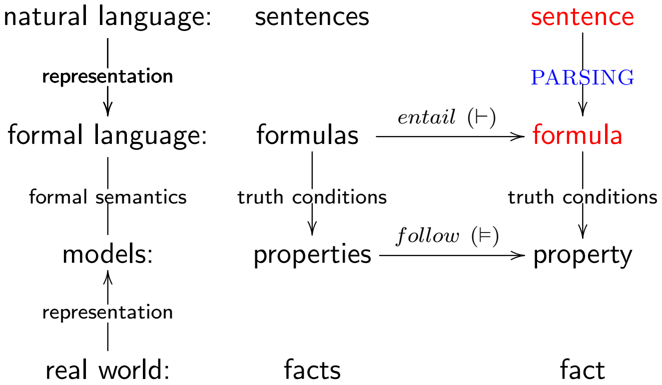
from Yanjing Wang

The general picture

$$\begin{tikzcd}
\text{natural language:} \arrow[d, "\text{representation}"] & \text{sentences} & \text{sentence} \\
\text{formal language:} \arrow[d, dash, "\text{formal semantics}"] & \text{formulas} \arrow[r, "entail\ (\vdash)"] \arrow[d, "\text{truth conditions}"] & \text{formula} \arrow[d, "\text{truth conditions}"] \\
\text{models:} & \text{properties} \arrow[r, "follow\ (\models)"] & \text{property} \\
\text{real world:} \arrow[u, "\text{representation}"] & \text{facts} & \text{fact}
\end{tikzcd}$$

from Yanjing Wang

The general picture

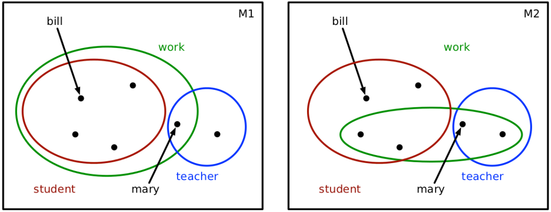


from Yanjing Wang

Logic as a formal language

Example

- $[\![$*every student works*$]\!]$ = $\texttt{true}$ iff. $\underline{\text{student}} \subseteq \underline{\text{work}}$
- *every student works* $\Rightarrow \forall x(\text{stud'}(x) \rightarrow \text{work'}(x))$



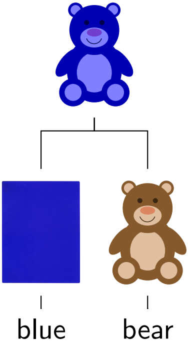
- Logic supports precise, consistent and controlled meaning representation via truth-conditional interpretation.
- Logic provides deduction systems to model inference processes, controlled through a formal entailment concept.
- Logic supports uniform modelling of the semantic composition process.

Semantic Composition

Modeling syntactico-semantic composition

The Principle of Compositionality

> *The meaning of an expression is a function of* **the meanings of its parts** *and of* **the way they are syntactically combined**.
>
> *B. Partee*

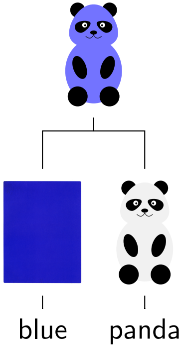


Modeling syntactico-semantic composition

The Principle of Compositionality

The meaning of an expression is a function of **the meanings of its parts** *and of* **the way they are syntactically combined**.

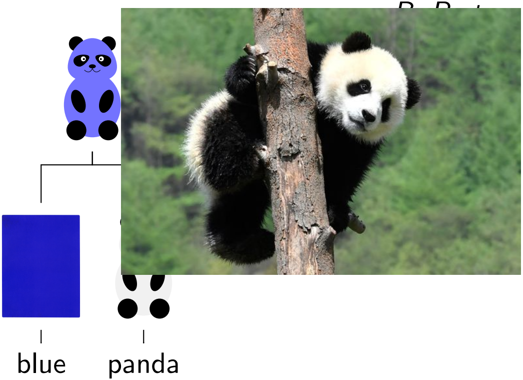
B. Partee



Modeling syntactico-semantic composition

The Principle of Compositionality

The meaning of an expression is a function of **the meanings of its parts** *and of* **the way they are syntactically combined**.

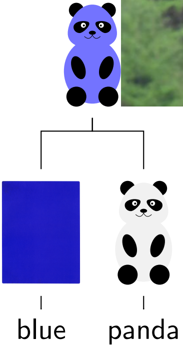


Modeling syntactico-semantic composition

The Principle of Compositionality

The meaning of an expression is

its parts *and of* **the way they are s**



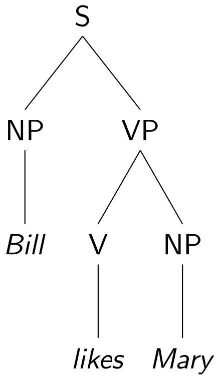
Rule-to-rule translation

Syntactic parsing + Lexical interpretation

$\Downarrow$

Meaning representation

Example



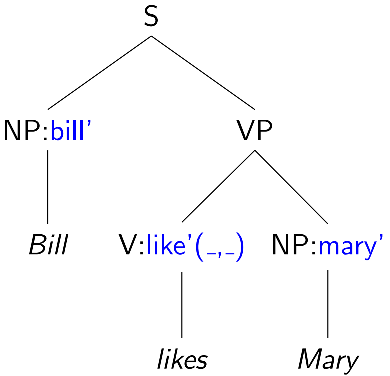
Rule-to-rule translation

Syntactic parsing + Lexical interpretation

$\Downarrow$

Meaning representation

Example



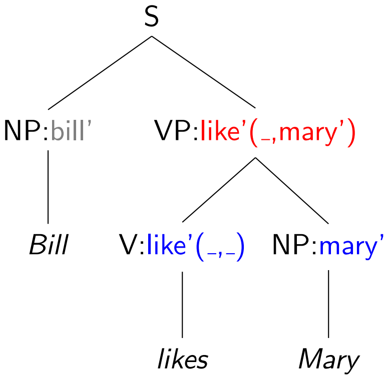
Rule-to-rule translation

Syntactic parsing + Lexical interpretation

$\Downarrow$

Meaning representation

Example



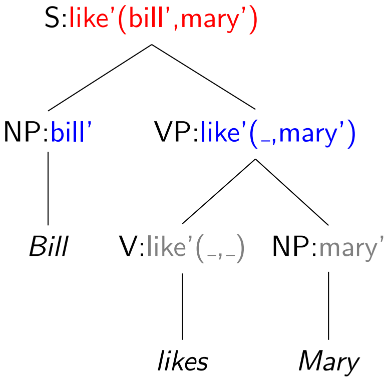
Rule-to-rule translation

Syntactic parsing + Lexical interpretation

$\Downarrow$

Meaning representation

Example



Using λ's

- **Church** defined an idealized programming language called the *λ-calculus*.
- A formal system in mathematical logic. A model of computation.
- λ-reduction:
  - λx[sleep'(x)](john') becomes sleep'(john')
  - $\lambda y[\lambda x.$love'$(x, y)]$(pizza') becomes λx[love'$(x,$ pizza')]
  - λx[love'$(x,$ pizza')](john') becomes love'(john', pizza')

Example

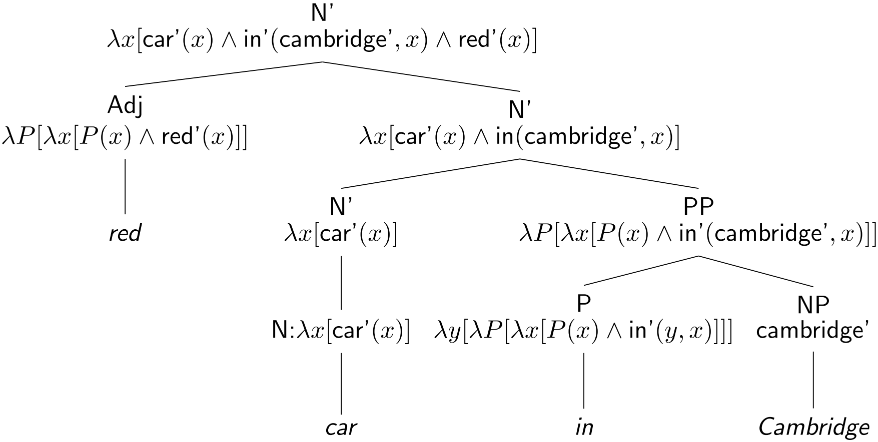
- $f(x) = x^2$ $\lambda x[x^2]$
- $f(5) = 25$ $\lambda x[x^2](5) = 25$
- $g(x, y) = x^2 + y^2$ $\lambda x[\lambda y[x^2 + y^2]]$
- $g(2, 1) = 5$ $(\lambda x[\lambda y[x^2 + y^2](2)](1) = 5$

Rule-to-rule translation

Syntactic parsing + Lexical interpretation

$\Downarrow$

Meaning representation

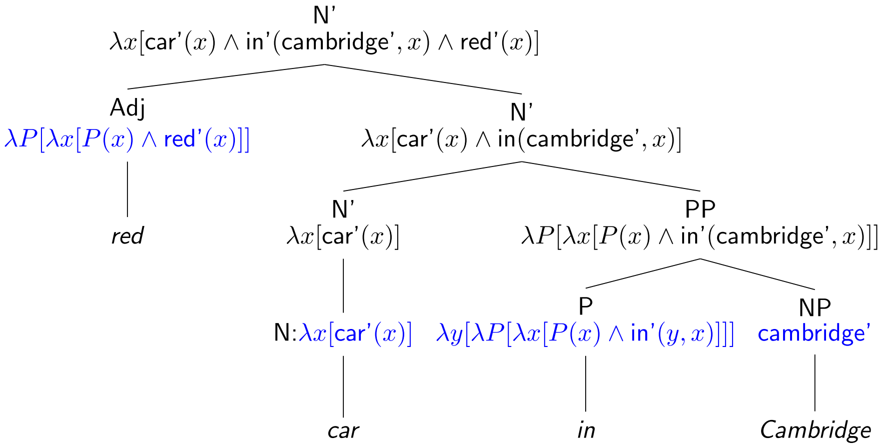


Rule-to-rule translation

Syntactic parsing + Lexical interpretation

$\Downarrow$

Meaning representation

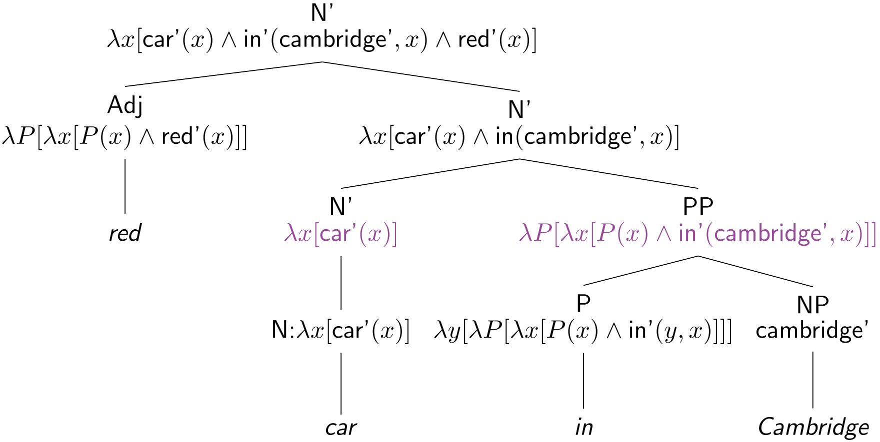


Rule-to-rule translation

Syntactic parsing + Lexical interpretation

$\Downarrow$

Meaning representation

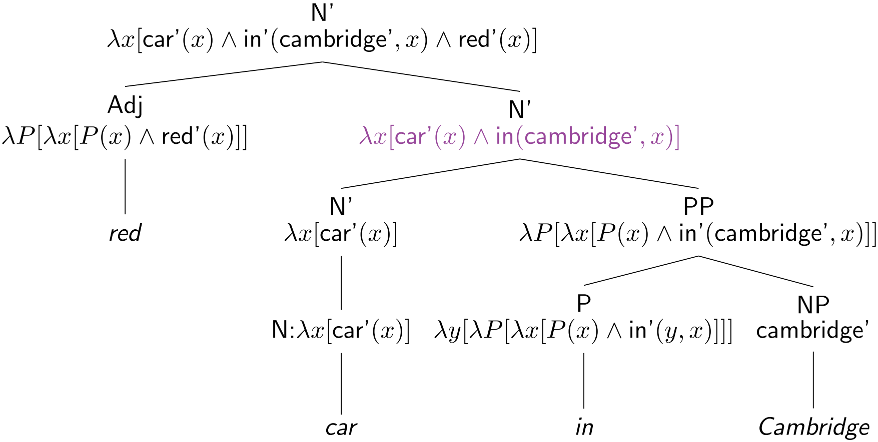


Rule-to-rule translation

Syntactic parsing + Lexical interpretation

$\Downarrow$

Meaning representation



Rule-to-rule translation

Syntactic parsing + Lexical interpretation

$\Downarrow$

Meaning representation

- N' $\lambda x[\text{car'}(x) \land \text{in'}(\text{cambridge'}, x) \land \text{red'}(x)]$
  - Adj $\lambda P[\lambda x[P(x) \land \text{red'}(x)]]$
    - *red*
  - N' $\lambda x[\text{car'}(x) \land \text{in}(\text{cambridge'}, x)]$
    - N' $\lambda x[\text{car'}(x)]$
      - N:$\lambda x[\text{car'}(x)]$
        - *car*
    - PP $\lambda P[\lambda x[P(x) \land \text{in'}(\text{cambridge'}, x)]]$
      - P $\lambda y[\lambda P[\lambda x[P(x) \land \text{in'}(y, x)]]]$
        - *in*
      - NP cambridge'
        - *Cambridge*

Semantic composition rules are non-trivial

Ordinary pronouns contribute to the semantics:

(1) a. *It barked.*

b. $\exists x[\text{bark'}(x) \wedge \textsc{pron}(x)]$

Pleonastic pronouns don't:

(2) a. *It rained.*

b. rain'

Similar syntactic structures may have different meanings. Different syntactic structures may have the same meaning:

(3) a. *Kim seems to sleep.*

b. *It seems that Kim sleeps.*

Differences in presentation but not in truth conditions.

Beyond toy examples ...

Use first order logic where possible (e.g., event variables, next slide).

However, First Order Predicate Calculus (FOPC) is sometimes inadequate: e.g., *most*, *may*, *believe*.

Quantifier scoping multiplies analyses:

(4) a. *Every cat chased some dog*

b. $\forall x[\text{cat'}(x) \rightarrow \exists y[\text{dog'}(y) \wedge \text{chase'}(x, y)]]$

c. $\exists y[\text{dog'}(y) \wedge \forall x[\text{cat'}(x) \rightarrow \text{chase'}(x, y)]]$

Often no straightforward logical analysis e.g., Bare plurals such as *Ducks lay eggs*.

Non-compositional phrases (multiword expressions): e.g., *red tape* meaning bureaucracy.

Event variables

Allow first order treatment of adverbs and PPs modifying verbs by *reifying* the event.

(5) a. *Rover barked*

b. bark'(r)

c. $\exists e[$bark'$(e, r)]$

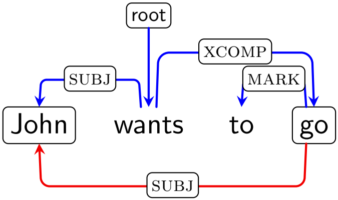
(6) a. *Rover barked loudly*

b. $\exists e[$bark'$(e, r) \wedge$ loud'$(e)]$

There was an event of Rover barking and that event was loud.

Graph-Based Meaning Representations

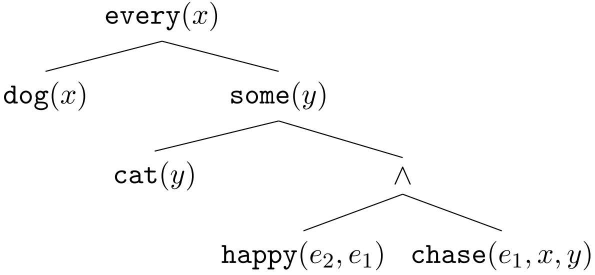
Non-tree dependency structures



Logical expression and semantic graph

- Every dog chases some cats.
- $\exists y(\mathtt{cat}(y) \wedge \forall x(\mathtt{dog}(x) \rightarrow \mathtt{chase}(e, x, y)))$
- $\mathtt{some}(y, \mathtt{cat}(y), \mathtt{every}(x, \mathtt{dog}(x), \mathtt{chase}(e, x, y)))$
- $\forall x(\mathtt{dog}(x) \rightarrow \exists y(\mathtt{cat}(y) \wedge \mathtt{chase}(e, x, y)))$
- $\mathtt{every}(x, \mathtt{dog}(x), \mathtt{some}(y, \mathtt{cat}(y), \mathtt{chase}(e, x, y)))$

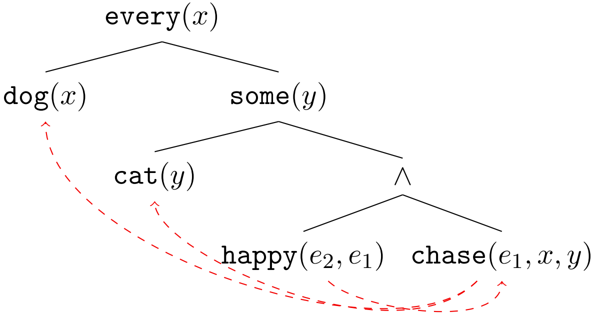
bracketing $\Rightarrow$ tree



Logical expression and semantic graph

- Every dog chases some cats.
- $\exists y(\texttt{cat}(y) \wedge \forall x(\texttt{dog}(x) \rightarrow \texttt{chase}(e, x, y)))$
- $\texttt{some}(y, \texttt{cat}(y), \texttt{every}(x, \texttt{dog}(x), \texttt{chase}(e, x, y)))$
- $\forall x(\texttt{dog}(x) \rightarrow \exists y(\texttt{cat}(y) \wedge \texttt{chase}(e, x, y)))$
- $\texttt{every}(x, \texttt{dog}(x), \texttt{some}(y, \texttt{cat}(y), \texttt{chase}(e, x, y)))$

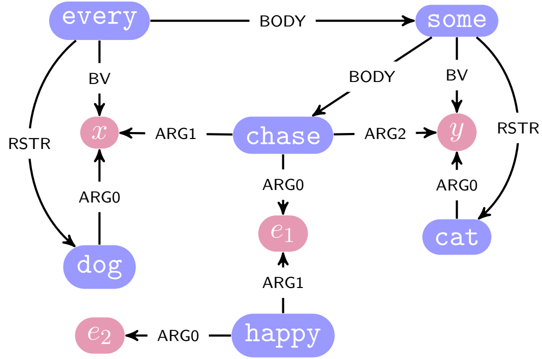
bracketing $\Rightarrow$ tree



Logical expression and semantic graph

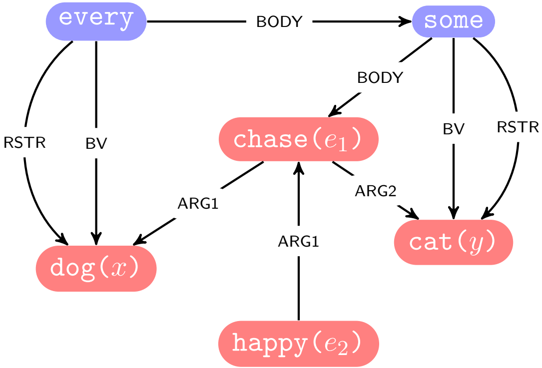
- Every dog chases some cats.
- $\exists y(\texttt{cat}(y) \land \forall x(\texttt{dog}(x) \rightarrow \texttt{chase}(e, x, y)))$
- $\texttt{some}(y, \texttt{cat}(y), \texttt{every}(x, \texttt{dog}(x), \texttt{chase}(e, x, y)))$
- $\forall x(\texttt{dog}(x) \rightarrow \exists y(\texttt{cat}(y) \land \texttt{chase}(e, x, y)))$
- $\texttt{every}(x, \texttt{dog}(x), \texttt{some}(y, \texttt{cat}(y), \texttt{chase}(e, x, y)))$

seperate predicates and variables



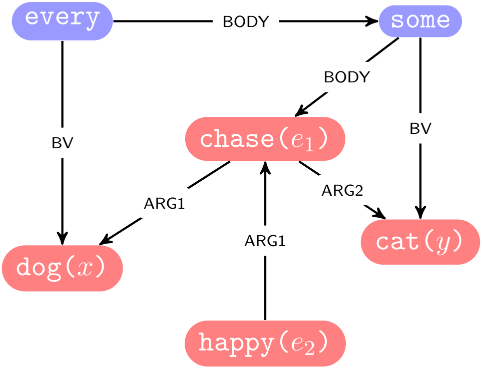
Logical expression and semantic graph

- Every dog chases some cats.
- $\exists y(\texttt{cat}(y) \land \forall x(\texttt{dog}(x) \rightarrow \texttt{chase}(e, x, y)))$
- $\texttt{some}(y, \texttt{cat}(y), \texttt{every}(x, \texttt{dog}(x), \texttt{chase}(e, x, y)))$
- $\forall x(\texttt{dog}(x) \rightarrow \exists y(\texttt{cat}(y) \land \texttt{chase}(e, x, y)))$
- $\texttt{every}(x, \texttt{dog}(x), \texttt{some}(y, \texttt{cat}(y), \texttt{chase}(e, x, y)))$



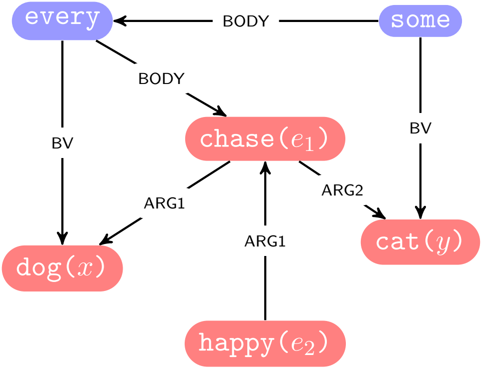
Logical expression and semantic graph

- Every dog chases some cats.
- $\exists y(\mathtt{cat}(y) \wedge \forall x(\mathtt{dog}(x) \rightarrow \mathtt{chase}(e, x, y)))$
- $\mathtt{some}(y, \mathtt{cat}(y), \mathtt{every}(x, \mathtt{dog}(x), \mathtt{chase}(e, x, y)))$
- $\forall x(\mathtt{dog}(x) \rightarrow \exists y(\mathtt{cat}(y) \wedge \mathtt{chase}(e, x, y)))$
- $\mathtt{every}(x, \mathtt{dog}(x), \mathtt{some}(y, \mathtt{cat}(y), \mathtt{chase}(e, x, y)))$



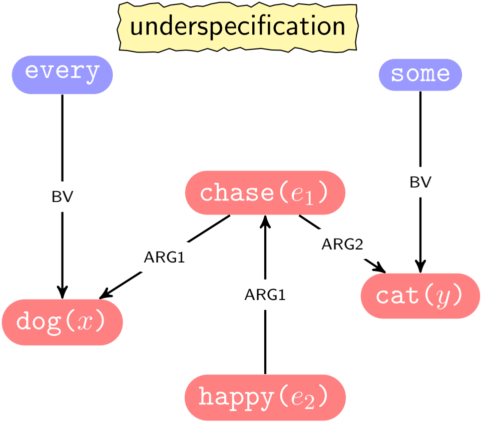
Logical expression and semantic graph

- Every dog chases some cats.
- $\exists y(\mathtt{cat}(y) \wedge \forall x(\mathtt{dog}(x) \rightarrow \mathtt{chase}(e,x,y)))$
- $\mathtt{some}(y, \mathtt{cat}(y), \mathtt{every}(x, \mathtt{dog}(x), \mathtt{chase}(e,x,y)))$
- $\forall x(\mathtt{dog}(x) \rightarrow \exists y(\mathtt{cat}(y) \wedge \mathtt{chase}(e,x,y)))$
- $\mathtt{every}(x, \mathtt{dog}(x), \mathtt{some}(y, \mathtt{cat}(y), \mathtt{chase}(e,x,y)))$



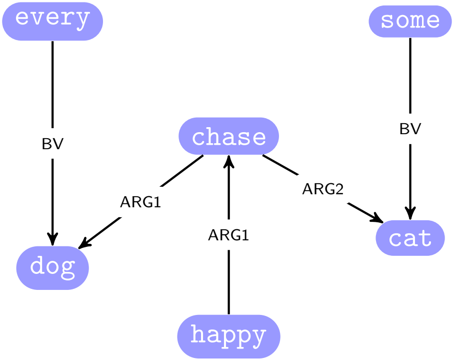
Logical expression and semantic graph

- Every dog chases some cats.
- $\exists y(\texttt{cat}(y) \wedge \forall x(\texttt{dog}(x) \rightarrow \texttt{chase}(e, x, y)))$
- $\texttt{some}(y, \texttt{cat}(y), \texttt{every}(x, \texttt{dog}(x), \texttt{chase}(e, x, y)))$
- $\forall x(\texttt{dog}(x) \rightarrow \exists y(\texttt{cat}(y) \wedge \texttt{chase}(e, x, y)))$
- $\texttt{every}(x, \texttt{dog}(x), \texttt{some}(y, \texttt{cat}(y), \texttt{chase}(e, x, y)))$



Logical expression and semantic graph

- Every dog chases some cats.
- $\exists y(\mathtt{cat}(y) \land \forall x(\mathtt{dog}(x) \rightarrow \mathtt{chase}(e, x, y)))$
- $\mathtt{some}(y, \mathtt{cat}(y), \mathtt{every}(x, \mathtt{dog}(x), \mathtt{chase}(e, x, y)))$
- $\forall x(\mathtt{dog}(x) \rightarrow \exists y(\mathtt{cat}(y) \land \mathtt{chase}(e, x, y)))$
- $\mathtt{every}(x, \mathtt{dog}(x), \mathtt{some}(y, \mathtt{cat}(y), \mathtt{chase}(e, x, y)))$



Inference and RTE

Natural language inference

Inference on a knowledge base: convert natural language expression to KB expression, valid inference according to KB.

- ☺ Precise
- ☺ Formally verifiable
- ☺ Disambiguation using KB state
- ☹ Limited domain, requires KB to be formally encodable

Language-based inference: does one utterance follow from another?

- ☺ Unlimited domain
- ☺/☹ Human judgement
- ☹/☺ Approximate/imprecise

Both approaches may use logical form of utterance.

Lexical meaning and meaning postulates

- Some inferences validated on logical representation directly, most require lexical meaning. What makes soup, soup?
- meaning postulates: e.g.,

$$\forall x[\textit{bachelor'}(x) \rightarrow \textit{man'}(x) \wedge \textit{unmarried'}(x)]$$

- usable with compositional semantics and theorem provers, e.g.

$$\textit{Kim is a bachelor}$$
$$\Downarrow$$
$$\text{bachelor'(kim')}$$
$$\Downarrow$$
$$\text{unmarried'(kim')}$$

- Problematic in general, OK for narrow domains or micro-worlds

Recognising Textual Entailment (RTE) shared tasks

T The girl was found in Drummondville earlier this month.

H The girl was discovered in Drummondville.

- *Data*: pairs of text (T) and hypothesis (H). H may or may not follow from T.
- *Task*: label `true` (if follows) or `false` (if doesn't follow), according to human judgements.

RTE using logical forms

- T sentence has logical form T', H sentence has logical form H'
- If $T' \Rightarrow H'$ conclude `true`, otherwise conclude `false`.

T The girl was found in Drummondville earlier this month.

T' $\exists x, u, e[\text{girl'}(x) \wedge \text{find'}(e, u, x) \wedge \text{in'}(e, \text{drummondville'}) \wedge \text{earlier-this-month'}(e)]$

H The girl was discovered in Drummondville.

H' $\exists x, u, e[\text{girl'}(x) \wedge \text{discover'}(e, u, x) \wedge \text{in'}(e, \text{drummondville'})]$

MP $\text{find'}(x, y, z) \Rightarrow \text{discover'}(x, y, z)$

- So $T' \Rightarrow$H$'$ and we conclude `true`

More complex examples

- T Four Venezuelan firefighters who were traveling to a training course in Texas were killed when their sport utility vehicle drifted onto the shoulder of a highway and struck a parked truck.
- H Four firefighters were killed in a car accident.

Systems using logical inference are not robust to missing information: simpler techniques can be effective (partly because of choice of hypotheses in RTE).

More examples

T Clinton's book is not a big seller here.

H Clinton's book is a big seller.

T After the war the city was briefly occupied by the Allies and then was returned to the Dutch.

H After the war, the city was returned to the Dutch.

T Lyon is actually the gastronomic capital of France.

H Lyon is the capital of France.

An example from a linguist

T The Commissioner doesn't regret that the President failed to make him leave Athens before May 2.

H The Commissioner was in Athens on May 2.

presupposition
negation
causation
event
semantic role
coreference
temporal expression

Reading

- Ann's lecture notes.
- ACL tutorial on graph-based meaning representations
  `https://github.com/cfmrp/tutorial`