L90: Overview of Natural Language Processing
Lecture 9: Compositional Semantics

Weiwei Sun

Department of Computer Science and Technology
University of Cambridge

Michaelmas 2020/21
Lecture 9: Compositional Semantics

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

syntax provides scaffolding for semantic composition
Principle: Being Able to Transform
What is meaning of $3 + 5 \times 6$?

- First parse it into $3 + (5 \times 6)$
- Now give a meaning to each node in the tree (bottom-up)
What is meaning of $3 + 5 \times 6$?

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

How about $3 + 5 \times x$?

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

$$E = \text{add}(3, \text{mult}(5, x))$$

$$E = 3 \quad F = \text{add} \quad E = \text{mult}(5, x)$$

$$N = 3 \quad + \quad E = 5 \quad F = \text{mult} \quad E = x$$

$$3 \quad 5 \quad 5$$

$$N = x \quad x$$
Interpreting in an environment

How about $3 + 5 \times x$?

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

```
E = add(3, mult(5, x))
```

```
E = 3  F = add
  |    |  E = mult(5, x)
  |    +  |
N = 3  +  E = 5  F = mult  E = x
  |    |    |
3     E = 5  *  N = x
  |    |    |    |
3     N = 5  *  N = x
  |    |    |    |    |
5     x
```
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?

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

<table>
<thead>
<tr>
<th>a compiler is a translator</th>
</tr>
</thead>
<tbody>
<tr>
<td>(formal language to formal language)</td>
</tr>
</tbody>
</table>

Natural Language Understanding ▶ 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

natural language \rightarrow representation \rightarrow formal language

formal semantics (logic) \rightarrow models \rightarrow representation \rightarrow real world

sentences

formulas \rightarrow truth conditions \rightarrow properties \rightarrow follow \rightarrow property

facts

sentence

formula

entail

truth condition

from Yanjing Wang
The general picture

natural language
  representation
    formal language
      formal semantics (logic)
        models
          representation
            real world

sentences
    formula
      entail
        truth conditions
          properties
            follow
              property

facts
  fact

from Yanjing Wang
The general picture

natural language
  \arrow{representation} \rightarrow formal language
  \arrow{formal semantics (logic)} \rightarrow models
  \arrow{representation} \rightarrow real world

sentences

sentence

formulas \rightarrow entail \rightarrow formula

logic

truth conditions

properties \rightarrow follow \rightarrow property

facts

fact

from Yanjing Wang
The general picture

natural language
  ↓
representation
  ↓
formal language
  ↓
formal semantics (logic)
  ↓
models
  ↑
representation
  ↓
real world

sentences
  ↑
sentence
  ↓
formulas
    ↓
    entail
  →
formula
  ↓
truth conditions
  ↓
properties
    follow
  →
property
  ↓
facts

from Yanjing Wang
Logic as a formal language

Example

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

- 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.*

B. Partee
Modeling syntactico-semantic composition

The Principle of Compositionality

The meaning of an expression is a function of its parts and of the way they are syntactically combined.
Rule-to-rule translation

Syntactic parsing + **Lexical interpretation**

\[\downarrow\]

**Meaning representation**

Example

```
S
  /\  \\
 NP  VP
  /   |   |
 Bill V   NP
    /   |   |
   likes Mary
```
Rule-to-rule translation

Syntactic parsing + Lexical interpretation

⇓

Meaning representation

Example

```
S
  NP: bill’
    Bill
  VP
    V: like’(_, _)
    NP: mary’
      likes
      Mary
```
Rule-to-rule translation

Syntactic parsing + Lexical interpretation

⇓

Meaning representation

Example

```
S

NP: bill'     VP: like'( _, mary' )

Bill    V: like'( _, _ )    NP: mary'

likes    Mary
```
Rule-to-rule translation

Syntactic parsing + **Lexical interpretation**

⇓

Meaning representation

Example

S:like’(bill’,mary’)

NP:bill’

VP:like’(_,mary’)

Bill

V:like’(_,_,)

NP:mary’

likes

Mary
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:**
  - $\lambda x [\text{sleep'}(x)](\text{john'})$ becomes $\text{sleep'}(\text{john'})$
  - $\lambda y [\lambda x.\text{love'}(x, y)](\text{pizza'})$ becomes $\lambda x [\text{love'}(x, \text{pizza'})]$
  - $\lambda x [\text{love'}(x, \text{pizza'})](\text{john'})$ becomes $\text{love'}(\text{john'}, \text{pizza'})$

**Example**

- $f(x) = x^2$
  - $f(5) = 25$
- $g(x, y) = x^2 + y^2$
- $g(2, 1) = 5$
Rule-to-rule translation

Syntactic parsing + Lexical interpretation

⇒

Meaning representation

\[ \lambda x[\text{car}'(x) \land \text{in}'(\text{cambridge}', x) \land \text{red}'(x)] \]

\[ \lambda x[\text{car}'(x) \land \text{in}(\text{cambridge}', x)] \]

\[ \lambda P[\lambda x[P(x) \land \text{red}'(x)]] \]

\[ \lambda x[\text{car}'(x)] \]

\[ \lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]] \]

\[ \text{red} \]

\[ \text{N}' \]

\[ \text{N:} \lambda x[\text{car}'(x)] \]

\[ \lambda y[\lambda P[\lambda x[P(x) \land \text{in}'(y, x)]]] \]

\[ \text{P} \]

\[ \text{NP} \]

\[ \text{Cambridge} \]
Rule-to-rule translation

Syntactic parsing + Lexical interpretation

\[\downarrow\]

Meaning representation

\[\lambda x[\text{car}(x) \land \text{in}'(\text{cambridge}', x) \land \text{red}'(x)]\]

\[\lambda P[\lambda x[P(x) \land \text{red}'(x)]]\]

\[\text{Adj}\]

\[\lambda P[\lambda x[P(x) \land \text{red}'(x)]]\]

\[\lambda x[\text{car}(x) \land \text{red}'(x)]\]

\[\lambda x[\text{car}(x) \land \text{in}(\text{cambridge}', x)]\]

\[\lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]]\]

\[\lambda x[\text{car}(x)]\]

\[\text{PP}\]

\[\lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]]\]

\[\text{NP}\]

\[\lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]]\]

\[\text{cambridge}'\]

\[\text{NP}\]

\[\lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]]\]

\[\text{cambridge}'\]

\[\text{NP}\]

\[\lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]]\]

\[\text{NP}\]

\[\lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]]\]

\[\text{cambridge}'\]

\[\text{NP}\]

\[\lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]]\]

\[\text{NP}\]

\[\lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]]\]

\[\text{cambridge}'\]
Rule-to-rule translation

Syntactic parsing + **Lexical interpretation**

\[ \Downarrow \]

Meaning representation

\[ \lambda x[\text{car}'(x) \land \text{in}'(\text{cambridge}', x) \land \text{red}'(x)] \]

\[ \lambda P[\lambda x[P(x) \land \text{red}'(x)]] \]

\[ \lambda x[\text{car}'(x) \land \text{in}(\text{cambridge}', x)] \]

\[ \text{Adj} \]

\[ \lambda P[\lambda x[P(x) \land \text{red}'(x)]] \]

\[ \lambda x[\text{car}'(x)] \]

\[ \lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]] \]

\[ \text{NP} \]

\[ \lambda x[\text{car}'(x)] \]

\[ \lambda y[\lambda P[\lambda x[P(x) \land \text{in}'(y, x)]]] \]

\[ \text{PP} \]

\[ \lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]] \]

\[ \text{NP} \]

\[ \text{cambridge}' \]

\[ \text{car} \]

\[ \text{in} \]

\[ \text{Cambridge} \]
Rule-to-rule translation

Syntactic parsing + Lexical interpretation

\[ \Downarrow \]

Meaning representation

\[ \lambda x[ \text{car}'(x) \land \text{in}'(\text{cambridge}', x) \land \text{red}'(x)] \]

\[ \lambda P[\lambda x[P(x) \land \text{red}'(x)]] \]

\[ \lambda x[\text{car}'(x) \land \text{in}(\text{cambridge}', x)] \]

\[ \lambda P[\lambda x[P(x) \land \text{in}'(\text{cambridge}', x)]] \]

\[ \lambda x[\text{car}'(x)] \]

\[ \lambda y[\lambda P[\lambda x[P(x) \land \text{in}'(y, x)]]] \]

\[ \lambda P[\lambda x[P(x) \land \text{in}'(y, x)]] \]

\[ \lambda x[\text{car}'(x)] \]

\[ \lambda y[\lambda P[\lambda x[P(x) \land \text{in}'(y, x)]]] \]

\[ \text{car} \]

\[ \text{in} \]

\[ \text{Cambridge} \]
Rule-to-rule translation

Syntactic parsing + **Lexical interpretation**

\[ \downarrow \]

Meaning representation

\[ N' \lambda x [ \text{car'(x)} \land \text{in'(cambridge', x)} \land \text{red'(x)}] \]

\[ \text{Adj} \]

\[ \lambda P[\lambda x [P(x) \land \text{red'(x)}]] \]

\[ \text{red} \]

\[ N' \lambda x [\text{car'(x)}] \]

\[ N : \lambda x [\text{car'(x)}] \]

\[ \text{car} \]

\[ \lambda P[\lambda x [P(x) \land \text{in'(cambridge', x)}]] \]

\[ \text{PP} \]

\[ \lambda y [\lambda P[\lambda x [P(x) \land \text{in'(y, x)}]]] \]

\[ \text{in} \]

\[ \text{NP} \]

\[ \text{cambridge'} \]

\[ \text{Cambridge} \]
Semantic composition rules are non-trivial

Ordinary pronouns contribute to the semantics:

(1) a. *It barked.*
    
    b. $\exists x [\text{bark}'(x) \land \text{PRON}(x)]$

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

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

(6) a. *Rover barked loudly*
   b. $\exists e[\text{bark}'(e, r) \land \text{loud}'(e)]$

There was an event of Rover barking and that event was loud.
Graph-Based Meaning Representations
Non-tree dependency structures

John wants to go
Logical expression and semantic graph

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

bracketing $\Rightarrow$ tree

\[
\text{every}(x) \\
\text{dog}(x) \quad \text{some}(y) \\
\text{cat}(y) \quad \land \\
\text{happy}(e_2, e_1) \quad \text{chase}(e_1, x, y)
\]
Logical expression and semantic graph

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

bracketing \(\Rightarrow\) tree
Logical expression and semantic graph

- Every dog chases some cats.
  \[ \exists y (\text{cat}(y) \land \forall x (\text{dog}(x) \rightarrow \text{chase}(e, x, y))) \]

- \text{some}(y, \text{cat}(y), \text{every}(x, \text{dog}(x), \text{chase}(e, x, y)))

- \text{every}(x, \text{dog}(x) \rightarrow \exists y (\text{cat}(y) \land \text{chase}(e, x, y)))

- \text{every}(x, \text{dog}(x), \text{some}(y, \text{cat}(y), \text{chase}(e, x, y)))

separate predicates and variables
Logical expression and semantic graph

- Every dog chases some cats.
  - $\exists y (\text{cat}(y) \land \forall x (\text{dog}(x) \rightarrow \text{chase}(e, x, y)))$
- some$(y, \text{cat}(y), \text{every}(x, \text{dog}(x), \text{chase}(e, x, y)))$
- $\forall x (\text{dog}(x) \rightarrow \exists y (\text{cat}(y) \land \text{chase}(e, x, y)))$
- every$(x, \text{dog}(x), \text{some}(y, \text{cat}(y), \text{chase}(e, x, y)))$
Logical expression and semantic graph

- Every dog chases some cats.
- $\exists y (\text{cat}(y) \land \forall x (\text{dog}(x) \rightarrow \text{chase}(e, x, y)))$
- $\text{some}(y, \text{cat}(y), \text{every}(x, \text{dog}(x), \text{chase}(e, x, y)))$
- $\forall x (\text{dog}(x) \rightarrow \exists y (\text{cat}(y) \land \text{chase}(e, x, y)))$
- $\text{every}(x, \text{dog}(x), \text{some}(y, \text{cat}(y), \text{chase}(e, x, y)))$
Logical expression and semantic graph

- Every dog chases some cats.
  \[ \exists y (\text{cat}(y) \land \forall x (\text{dog}(x) \rightarrow \text{chase}(e, x, y))) \]
- Some(y, cat(y), every(x, dog(x), chase(e, x, y)))
- \[ \forall x (\text{dog}(x) \rightarrow \exists y (\text{cat}(y) \land \text{chase}(e, x, y))) \]
- Every(x, dog(x), some(y, cat(y), chase(e, x, y)))
Logical expression and semantic graph

- Every dog chases some cats.
- $\exists y (\text{cat}(y) \land \forall x (\text{dog}(x) \rightarrow \text{chase}(e, x, y)))$
- $\text{some}(y, \text{cat}(y), \text{every}(x, \text{dog}(x), \text{chase}(e, x, y)))$
- $\forall x (\text{dog}(x) \rightarrow \exists y (\text{cat}(y) \land \text{chase}(e, x, y)))$
- $\text{every}(x, \text{dog}(x), \text{some}(y, \text{cat}(y), \text{chase}(e, x, y)))$
Logical expression and semantic graph

• Every dog chases some cats.
  \[ \exists y (\text{cat}(y) \land \forall x (\text{dog}(x) \rightarrow \text{chase}(e, x, y))) \]

• some(y, cat(y), every(x, dog(x), chase(e, x, y)))

• \[ \forall x (\text{dog}(x) \rightarrow \exists y (\text{cat}(y) \land \text{chase}(e, x, y))) \]

• every(x, dog(x), some(y, cat(y), 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.,

$$\forall x [bachelor'(x) \rightarrow man'(x) \land unmarried'(x)]$$

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

  \[
  \text{Kim is a bachelor} \\
  \Downarrow \\
  bachelor'(kim') \\
  \Downarrow \\
  unmarried'(kim')
  \]

- 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 \texttt{true} (if follows) or \texttt{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.

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

$H$  The girl was discovered in Drummondville.

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

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

- So $T' \Rightarrow H'$ and we conclude true
More complex examples

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.

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

\text{T} Clinton’s book is not a big seller here.

\text{H} Clinton’s book is a big seller.

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

\text{H} After the war, the city was returned to the Dutch.

\text{T} Lyon is actually the gastronomic capital of France.

\text{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