CS447: Natural Language Processing

http://courses.engr.illinois.edu/cs447

Lecture 25: Compositional Semantics

Julia Hockenmaier

juliahmr@illinois.edu

3324 Siebel Center

Semantics

In order to understand language, we need to know its meaning.

- What is the meaning of a word? (**Lexical semantics**)
- What is the meaning of a sentence? **([Compositional] semantics)**
- What is the meaning of a longer piece of text? **(Discourse semantics)**

NB:There are a lot of different approaches for each of these different aspects of semantics; since each approach typically focuses on one or two of these aspects (or particular phenomena within them), we don't really have a "unified theory" of semantics.

Natural language conveys information about the world

We can compare statements about the world with the actual state of the world:

Champaign is in California. (false)

We can learn new facts about the world from natural language statements:

The earth turns around the sun.

We can answer questions about the world:

Where can I eat Korean food on campus?

We draw inferences from natural language statements

Some inferences are purely linguistic:

All blips are foos.
Blop is a blip.
Blop is a foo (whatever that is).

John ate the cake.
The cake was eaten by John.

We draw inferences from natural language statements

Some inferences require world knowledge.

Mozart was born in Salzburg.
Mozart was born in Vienna.
———————————————
No, that can't be — these are different cities.

Mozart was born in Salzburg.
Mozart was born in Austria.
———————————————
Yes, that is correct — Salzburg is a city in Austria.

Today's lecture

Our initial question:

What is the meaning of (declarative) sentences?

Declarative sentences: *"John likes coffee"*.

(We won't deal with questions (*"Who likes coffee?"*) and imperative sentences (commands: *"Drink up!"*))

Follow-on question 1:

How can we represent the meaning of sentences?

Follow-on question 2:

How can we map a sentence to its meaning representation?

What we won't discuss here

How do we represent world knowledge?
(e.g. that Salzburg and Vienna are cities, that Austria is a country, etc.)

There is a lot of work on knowledge representation and ontologies, including old research on "semantic networks", and recent work on so-called knowledge graphs that is becoming increasingly important for NLP and other areas of AI.

How do we draw logical inferences once we have translated natural language into formal logic?

There is a lot of work on purely logic-based theorem proving, as well as probabilistic logic (e.g Markov Logic Networks) that is relevant here.

What do nouns and verbs mean?

In the simplest case, an NP is just a name: *John*
Names refer to entities in the world.

Verbs define n-ary predicates: depending on the arguments they take (and the state of the world), the result can be true or false.

What do sentences mean?

Declarative sentences (statements) can be true or false, depending on the state of the world:

John sleeps.

In the simplest case, the consist of a verb and one or more noun phrase arguments.

Principle of compositionality (Frege):
The meaning of an expression depends on the meaning of its parts and how they are put together.

First-order predicate logic (FOL) as a meaning representation language

Predicate logic expressions

Terms: refer to entities

- Variables: x, y, z
- Constants: John’, Urbana’
- Functions applied to terms (fatherOf(John’)’)

Predicates: refer to properties of, or relations between, entities

- tall’(x), eat’(x,y), …

Formulas: can be true or false

- Atomic formulas: predicates, applied to terms: tall’(John’)
- Complex formulas: constructed recursively via logical connectives and quantifiers

Formulas

Atomic formulas are predicates, applied to terms:

book(x), *eat*(x,y)

Complex formulas are constructed recursively by

...**negation** ($\neg$): $\neg$*book*(*John'*)

...**connectives** ($\wedge, \vee, \rightarrow$): *book*($y$) $\wedge$ *read*(x,y)

conjunction (and): $\varphi \wedge \psi$ disjunction (or): $\varphi \vee \psi$ implication (if): $\varphi \rightarrow \psi$

...**quantifiers** ($\forall x$, $\exists x$)

universal (typically with implication) $\forall x[\varphi(x) \rightarrow \psi(x)]$

existential (typically with conjunction) $\exists x[\varphi(x)]$, $\exists x[\varphi(x) \wedge \psi(x)]$

Interpretation: formulas are either **true or false**.

The syntax of FOL expressions

Term $\Rightarrow$ Constant |
Variable |
Function(Term,...,Term)

Formula $\Rightarrow$ Predicate(Term, ...Term) |
$\neg$ Formula |
$\forall$ Variable Formula |
$\exists$ Variable Formula |
Formula $\wedge$ Formula |
Formula $\vee$ Formula |
Formula $\rightarrow$ Formula

Some examples

John is a student:
student(john)

All students take at least one class:
$\forall x\ student(x) \longrightarrow \exists y(class(y) \wedge takes(x,y))$

There is a class that all students take:
$\exists y(class(y) \wedge \forall x\ (student(x) \longrightarrow takes(x,y))$

FOL is sufficient for many Natural Language inferences

All blips are foos.	$\forall x\ \text{blip}(x) \rightarrow \text{foo}(x)$
Blop is a blip.	$\text{blip}(\text{blop})$
Blop is a foo	$\text{foo}(\text{blop})$

Some inferences require world knowledge.

Mozart was born in Salzburg.	$\text{bornIn}(\text{Mozart}, \text{Salzburg})$
Mozart was born in Vienna.	$\text{bornIn}(\text{Mozart}, \text{Vienna})$
No, that can't be- these are different cities	$\text{bornIn}(\text{Mozart}, \text{Salzburg}) \wedge \neg \text{bornIn}(\text{Mozart}, \text{Salzburg})$

Not all of natural language can be expressed in FOL:

Tense:

It <u>was</u> hot <u>yesterday</u>.

I <u>will</u> go to Chicago <u>tomorrow</u>.

Aspect:

I <u>am going</u> to Chicago.

Modals:

You <u>can</u> go to Chicago from here.

You <u>must</u> go to Chicago tomorrow.

Other kinds of quantifiers:

<u>Most</u> students hate 8:00am lectures.

λ-Expressions

We often use **λ-expressions**
to construct complex logical formulas:

- $\lambda x.\varphi(..x...)$ is a **function** where x is a variable, and φ some FOL expression.

- **β-reduction** (called λ-reduction in textbook):
  Apply $\lambda x.\varphi(..x...)$ to some argument a:
  $(\lambda x.\varphi(..\mathrm{x}...)\ a) \Rightarrow \varphi(..\mathrm{a}...)$
  Replace all occurrences of x in $\varphi(..\mathrm{x}...)$ with a

- **n-ary functions** contain embedded λ-expressions:
  $\lambda x.\lambda y.\lambda z.give(x,y,z)$

Using Combinatory Categorial Grammar (CCG) to map sentences to predicate logic

Function application

Combines a function X/Y or X\Y with its argument Y to yield the result X:

(S\NP)/NP	NP	->	S\NP
eats	tapas		eats tapas
NP	S\NP	->	S
John	eats tapas		John eats tapas

Type-raising and composition

Type-raising: X → T/(T\X)

Turns an argument into a function.

NP → S/(S\NP) (subject)

NP → (S\NP)\((S\NP)/NP) (object)

Harmonic composition: X/Y Y/Z → X/Z

Composes two functions (complex categories)

(S\NP)/PP PP/NP → (S\NP)/NP

S/(S\NP) (S\NP)/NP → S/NP

Crossing function composition: X/Y Y\Z → X\Z

Composes two functions (complex categories)

(S\NP)/S S\NP → (S\NP)\NP

Type-raising and composition

Wh-movement (relative clause):

$$
\begin{array}{cccc}
\text{the tapas} & \text{which} & \text{Mary} & \text{ordered} \\
\hline NP & (NP\backslash NP)/(S/NP) & NP & (S\backslash NP)/NP \\
 & & \overline{S/(S\backslash NP)}^{>T} & \\
 & & \multicolumn{2}{c}{\overline{\qquad S/NP \qquad}^{>B}} \\
 & \multicolumn{3}{c}{\overline{\qquad\qquad NP\backslash NP \qquad\qquad}^{>}} \\
\multicolumn{4}{c}{\overline{\qquad\qquad\qquad NP \qquad\qquad\qquad}^{<}}
\end{array}
$$

Right-node raising:

$$
\begin{array}{cccccc}
\text{Mary} & \text{ordered} & \text{and} & \text{John} & \text{ate} & \text{the tapas} \\
\hline NP & (S\backslash NP)/NP & conj & NP & (S\backslash NP)/NP & NP \\
\overline{S/(S\backslash NP)}^{>T} & & & \overline{S/(S\backslash NP)}^{>T} & & \\
\multicolumn{2}{c}{\overline{\qquad S/NP \qquad}^{>B}} & & \multicolumn{2}{c}{\overline{\qquad S/NP \qquad}^{>B}} & \\
\multicolumn{5}{c}{\overline{\qquad\qquad S/NP \qquad\qquad}^{<\Phi>}} & \\
\multicolumn{6}{c}{\overline{\qquad\qquad\qquad S \qquad\qquad\qquad}^{>}}
\end{array}
$$

An example

$$
\begin{array}{ccc}
\textit{John} & \textit{sees} & \textit{Mary} \\
\hline
\mathbf{NP} & (\mathbf{S}\backslash\mathbf{NP})/\mathbf{NP} & \mathbf{NP} \\
\cline{2-3}
 & \multicolumn{2}{c}{\mathbf{S}\backslash\mathbf{NP} \quad >} \\
\hline
\multicolumn{3}{c}{\mathbf{S} \quad <}
\end{array}
$$

CCG semantics

Every syntactic constituent has a semantic interpretation:

Every **lexical entry** maps a word to a syntactic category and a corresponding semantic type:

John=(**NP**, john') Mary= (**NP**, mary')
loves: (**(S\NP)/NP** $\lambda x.\lambda y.loves(x,y)$)

Every **combinatory rule** has a syntactic and a semantic part:

Function application: **X/Y**:$\lambda x.f(x)$ **Y:**a $\rightarrow$ **X**:f(a)

Function composition: **X/Y**:$\lambda x.f(x)$ **Y/Z:**$\lambda y.g(y)$ $\rightarrow$ **X/Z**:$\lambda z.f(\lambda y.g(y).z)$

Type raising: **X**:a $\rightarrow$ **T/(T\X)** $\lambda f.f(a)$

An example with semantics

$$
\begin{array}{ccc}
\mathit{John} & \mathit{sees} & \mathit{Mary} \\
\cline{1-1}\cline{2-2}\cline{3-3}
\mathbf{NP} : \mathit{John} & (\mathbf{S}\backslash\mathbf{NP})/\mathbf{NP} : \lambda x.\lambda y.\mathit{sees}(x,y) & \mathbf{NP} : \mathit{Mary} \\
\cline{2-3}
 & \multicolumn{2}{c}{\mathbf{S}\backslash\mathbf{NP} : \lambda y.\mathit{sees}(\mathit{Mary},y) \quad >} \\
\cline{1-3}
\multicolumn{3}{c}{\mathbf{S} : \mathit{sees}(\mathit{Mary},\mathit{John}) \quad <}
\end{array}
$$

Quantifier scope ambiguity

"Every chef cooks a meal"

- **Interpretation A:**
  For every chef, there is a meal which he cooks.

$$\forall x[chef(x) \rightarrow \exists y[meal(y) \wedge cooks(y,x)]]$$

- **Interpretation B:**
  There is some meal which every chef cooks.

$$\exists y[meal(y) \wedge \forall x[chef(x) \rightarrow cooks(y,x)]]$$

Interpretation A

$$
\begin{array}{ccccc}
\text{Every} & \text{chef} & \text{cooks} & \text{a} & \text{meal} \\
\hline
(\mathbf{S}/(\mathbf{S}\backslash\mathbf{NP}))/\mathbf{N} & \mathbf{N} & (\mathbf{S}\backslash\mathbf{NP})/\mathbf{NP} & ((\mathbf{S}\backslash\mathbf{NP})\backslash((\mathbf{S}\backslash\mathbf{NP})/\mathbf{NP}))/\mathbf{N} & \mathbf{N} \\
\lambda P\lambda Q.\forall x[Px \rightarrow Qx] & \lambda z.chef(z) & \lambda u.\lambda v.cooks(u,v) & \lambda P\lambda Q\exists y[Py \wedge Qy] & \lambda z.meal(z)
\end{array}
$$

Every chef $\Rightarrow$ (>):

$$
\begin{gathered}
\mathbf{S}/(\mathbf{S}\backslash\mathbf{NP}) \\
\lambda Q.\forall x[\lambda z.chef(z)x \rightarrow Qx] \\
\equiv \lambda Q.\forall x[chef(x) \rightarrow Qx]
\end{gathered}
$$

a meal $\Rightarrow$ (>):

$$
\begin{gathered}
(\mathbf{S}\backslash\mathbf{NP})\backslash((\mathbf{S}\backslash\mathbf{NP})/\mathbf{NP}) \\
\lambda Q\exists y[\lambda z.meal(z)y \wedge Qy] \\
\equiv \lambda Q\lambda w.\exists y[meal(y) \wedge Qyw]
\end{gathered}
$$

cooks a meal $\Rightarrow$ (<):

$$
\begin{gathered}
\mathbf{S}\backslash\mathbf{NP} \\
\lambda w.\exists y[meal(y) \wedge \lambda u\lambda v.cooks(u,v)yw] \\
\equiv \lambda w.\exists y[meal(y) \wedge cooks(y,w)]
\end{gathered}
$$

Every chef cooks a meal $\Rightarrow$ (>):

$$
\begin{gathered}
\mathbf{S} : \forall x[chef(x) \rightarrow \lambda w.\exists y[meal(y) \wedge cooks(y,w)]x] \\
\equiv \forall x[chef(x) \rightarrow \exists y[meal(y) \wedge cooks(y,x)]]
\end{gathered}
$$

Interpretation B

$$
\begin{array}{ccccc}
\text{Every} & \text{chef} & \text{cooks} & \text{a} & \text{meal}\\
\hline
(\mathbf{S}/(\mathbf{S}\backslash\mathbf{NP}))/\mathbf{N} & \mathbf{N} & (\mathbf{S}\backslash\mathbf{NP})/\mathbf{NP} & (\mathbf{S}\backslash(\mathbf{S}/\mathbf{NP}))/\mathbf{N} & \mathbf{N}\\
\lambda P \lambda Q.\forall x[Px \rightarrow Qx] & \lambda z.chef(z) & \lambda u.\lambda v.cooks(u,v) & \lambda P \lambda Q \exists y[Py \wedge Qy] & \lambda z.meal(z)
\end{array}
$$

Every chef $\xrightarrow{\;>\;}$

$$
\begin{array}{c}
\mathbf{S}/(\mathbf{S}\backslash\mathbf{NP})\\
\lambda Q \forall x[\lambda z.chef(z)x \rightarrow Qx]\\
\equiv \lambda Q \forall x[chef(x) \rightarrow Qx]
\end{array}
$$

a meal $\xrightarrow{\;>\;}$

$$
\begin{array}{c}
\mathbf{S}\backslash(\mathbf{S}/\mathbf{NP})\\
\lambda Q \exists y[\lambda z.meal(z)y \wedge Qy]\\
\equiv \lambda Q \exists y[meal(y) \wedge Qy]
\end{array}
$$

Every chef cooks $\xrightarrow{\;>\mathbf{B}\;}$

$$
\begin{array}{c}
\mathbf{S}/\mathbf{NP}\\
\lambda w.\forall x[chef(x) \rightarrow \lambda u \lambda v.cooks(u,v)wx]\\
\equiv \lambda w.\forall x[chef(x) \rightarrow cooks(w,x)]
\end{array}
$$

Every chef cooks a meal $\xrightarrow{\;<\;}$

$$
\begin{array}{c}
\mathbf{S} \exists y[meal(y) \wedge \lambda w.\forall x[chef(x) \rightarrow cooks(y,w)]x]\\
\equiv \exists y[meal(y) \wedge \forall x[chef(x) \rightarrow cooks(y,x)]]
\end{array}
$$

Additional topics

Representing events and temporal relations:

- Add event variables *e* to represent the events described by verbs, and temporal variables *t* to represent the time at which an event happens.

Other quantifiers:

- What about "*most / at least two / … chefs*"?

Underspecified representations:

- Which interpretation of *"Every chef cooks a meal"* is correct? This might depend on context. Let the parser generate an underspecified representation from which both readings can be computed.

Going beyond single sentences:

- How do we combine the interpretations of single sentences?

Today’s key concepts

Why do we need to represent meaning?

- Inference
- Interactions with (general) world knowledge and situational contex

How do we represent meaning?

- First order predicate logic
- Semantics in CCG

Today's reading

Textbook:

Chapter 17, sections 1-3

Chapter 18, section 2 (for a slightly different treatment of computational semantics)

Optional: Chapter 18, section 3 (underspecified representations)