Lecture 15: Compositional Semantics

Julia Hockenmaier
juliahmr@illinois.edu
3324 Siebel Center
Admin
Midterm results

- Undergrads (black)
- Grads (blue)

100% = 22 points
Ugrad median: 17.9 points ≈ B
(Same scale for grads)
Midterm results

Converting **points to percentages:**
- 22 out of 25 points = 100% (you will soon see both in Compass)

Converting **points/percentages to letter grades:**
The final conversion will be based on the **total percentage** at the end of the semester (MPs, midterm, final, (project))
I use the **undergrads’ performance** as yardstick for everybody

If I had to give letter grades for this midterm, here is a **rough** scale:
- You would need **19 points (~86%)** or more to get an A
- The **undergrad median (17.9 points = 81.4%)** would correspond to a **B** letter grade
- You would need **at least 40% (9 points)** to pass the class.
Regrade requests

We will post solutions on the class website (securely).

We will accept regrade requests until Nov 9.
How can you do better?

Come to class, and participate.

Spend time with the material after each lecture.

Read the textbook.

Use Piazza.

Come to office hours.

Let us know if you struggle.
4th Credit hour: Proposal

Upload a one-page PDF to Compass by this Friday
- written in LaTeX (not MS Word)
- with full bibliography of the papers you want to read or base your project on
  (ideally with links to online versions; add url-field to your bibtex file)

- include a motivation of why you have chosen those papers
- for a research project: tell me whether you have the data you need, what existing software you will be using, what you will have to implement yourself.
- mention any questions/concerns that you may have
- include your names and NetId/Illinois emails
- one proposal per project is fine.
Back to the material…
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)
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:

\textit{All blips are foos.}
\textit{Blop is a blip.}
\textit{Blop is a foo (whatever that is).}

Some inferences require world knowledge.

\textit{Mozart was born in Salzburg.}
\textit{Mozart was born in Vienna.}
\textit{No, that can’t be - these are different cities.}
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 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** (\(\land, \lor, \rightarrow\)):

**conjunction (and):** \(\varphi \land \psi\)  
**disjunction (or):** \(\varphi \lor \psi\)  
**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):** \(\exists x [\varphi(x)], \, \exists x [\varphi(x) \land \psi(x)]\)

**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 $\land$ Formula | Formula $\lor$ Formula | Formula $\rightarrow$ Formula
Some examples

John is a student:
student(john)

All students take at least one class:
\( \forall x \, \text{student}(x) \rightarrow \exists y (\text{class}(y) \land \text{takes}(x,y)) \)

There is a class that all students take:
\( \exists y (\text{class}(y) \land \forall x \, (\text{student}(x) \rightarrow \text{takes}(x,y))) \)
FOL is sufficient for many Natural Language inferences

All blips are foos.  
Blop is a blip.  
Blop is a foo

\[ \forall x \ blip(x) \rightarrow foo(x) \]
\[ blip(blop) \]
\[ foo(blop) \]

Some inferences require world knowledge.

Mozart was born in Salzburg.  
Mozart was born in Vienna.

No, that can’t be-
these are different cities

\[ bornIn(Mozart, Salzburg) \]
\[ bornIn(Mozart, Vienna) \]
\[ \neg bornIn(Mozart, Salzburg) \land \neg bornIn(Mozart, Salzburg) \]
Not all of natural language can be expressed in FOL:

**Tense:**
- It was hot yesterday.
- I will go to Chicago tomorrow.

**Modals:**
- You can go to Chicago from here.

**Other kinds of quantifiers:**
- Most students hate 8:00am lectures.
\(\lambda\)-Expressions

We often use \(\lambda\)-expressions to construct complex logical formulas:

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

- \(\beta\)-reduction (called \(\lambda\)-reduction in textbook):
  Apply \(\lambda x.\varphi(x)\) to some argument \(a\):
  \((\lambda x.\varphi(x)) a \Rightarrow \varphi(a)\)
  Replace all occurrences of \(x\) in \(\varphi(x)\) with \(a\)

- \(n\)-ary functions contain embedded \(\lambda\)-expressions:
  \(\lambda x.\lambda y.\lambda z.\text{give}(x,y,z)\)
CCG: the machinery

Categories:
   specify subcat lists of words/constituents.

Combinatory rules:
   specify how constituents can combine.

The lexicon:
   specifies which categories a word can have.

Derivations:
   spell out process of combining constituents.
(Combinatory) Categorial Grammar
## CCG categories

**Simple (atomic) categories:** NP, S, PP

**Complex categories** (functions):
Return a **result** when combined with an **argument**

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP, intransitive verb</td>
<td>S\NP</td>
</tr>
<tr>
<td>Transitive verb</td>
<td>(S\NP)/NP</td>
</tr>
<tr>
<td>Adverb</td>
<td>(S\NP)(S\NP)</td>
</tr>
</tbody>
</table>
| Prepositions              | ((S\NP)\(S\NP))/NP  
                          | (NP\NP)/NP      
                          | PP/NP             |
Function application

Forward application (>):

\[(S\NP)/NP\ NP \Rightarrow S\NP\]

\[\text{eats tapas} \Rightarrow \text{eats tapas}\]

Backward application (<):

\[\text{NP} \quad \text{S}\NP \quad \Rightarrow \quad \text{S}\]

\[\text{John eats tapas} \quad \Rightarrow \quad \text{John eats tapas}\]

Used in all variants of categorial grammar
A (C)CG derivation

\[
\begin{align*}
\text{John} & \quad \text{eats} & \quad \text{tapas} \\
\text{NP} & \quad \frac{(S \setminus \text{NP})/\text{NP}}{} & \quad \text{NP} \\
\frac{S \setminus \text{NP}}{} & \quad \frac{S}{S}
\end{align*}
\]
Function application

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

\[(S\ Beneficial Action) / \ Beneficial Action \rightarrow S\ Beneficial Action\]

\[\text{eats tapas \rightarrow eats tapas}\]

\[\text{NP \rightarrow S}\]

\[\text{John eats tapas \rightarrow John eats tapas}\]
Type-raising and composition

Type-raising: \( X \rightarrow T/(T\backslash X) \)

**Turns an argument into a function.**

- NP → S/(S\NP) (subject)
- NP → (S\NP)/(S\NP)/NP (object)

Harmonic composition: \( X/Y \ Y/Z \rightarrow X/Z \)

**Composes two functions (complex categories)**

- (S\NP)/PP PP/NP → (S\NP)/NP
- S/(S\NP) (S\NP)/NP → S/NP

Crossing function composition: \( X/Y \ Y\backslash Z \rightarrow X\backslash Z \)

**Composes two functions (complex categories)**

- (S\NP)/S S\NP → (S\NP)/NP

CS447: Natural Language Processing
Type-raising and composition

Wh-movement (relative clause):

Right-node raising:

CS447: Natural Language Processing 30
An example

\[
\begin{array}{c}
\text{John} \\
\text{NP} \\
(\text{S}\backslash\text{NP})/\text{NP} \\
\text{S}\backslash\text{NP} \\
\text{S}
\end{array}
\]
Using Combinatory Categorial Grammar (CCG) to map sentences to predicate logic
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 λx.λy.loves(x,y))

Every **combinatory rule** has a syntactic and a semantic part:
Function application:   X/Y:λx.f(x)    Y:a          → X:f(a)
Function composition: X/Y:λx.f(x)  Y/Z:λy.g(y) → X/Z:λz.f(λy.g(y).z)
Type raising:          X:a          → T/(T\X)  λf.f(a)
An example with semantics

\[
\begin{align*}
\text{John} & \quad \text{sees} & \quad \text{Mary} \\
\text{NP} : \text{John} & \quad (S\backslash\text{NP})/\text{NP} : \lambda x.\lambda y.\text{sees}(x, y) & \quad \text{NP} : \text{Mary} \\
& \quad S\backslash\text{NP} : \lambda y.\text{sees}(\text{Mary}, y) & > \\
& \quad S : \text{sees}(\text{Mary}, \text{John}) & <
\end{align*}
\]
Supplementary material: quantifier scope ambiguities in CCG
Quantifier scope ambiguity

“Every chef cooks a meal”

- **Interpretation A:**
  For every chef, there is a meal which he cooks.
  \[ \forall x [chef(x) \rightarrow \exists y [meal(y) \land cooks(y, x)]] \]

- **Interpretation B:**
  There is some meal which every chef cooks.
  \[ \exists y [meal(y) \land \forall x [chef(x) \rightarrow cooks(y, x)]] \]
Interpretation A
Interpretation B

\[
\begin{align*}
\text{Every chef cooks a meal} & \quad \text{Interpretation B} \\
\frac{(S/(S/NP))/N}{\text{Every}} & \quad \frac{(S/NP)/NP}{\text{chef}} & \quad \frac{(S/(S/NP))/N}{\text{meal}} \\
\lambda P \lambda Q. \forall x [P x \rightarrow Q x] & \quad \lambda z. \text{chef}(z) & \quad \lambda P \lambda Q \exists y [P y \land Q y] \\
\lambda Q \forall x [\lambda z. \text{chef}(z) x \rightarrow Q x] & \quad \lambda u. \lambda v. \text{cooks}(u, v) & \quad \lambda z. \text{meal}(z) \\
\lambda Q \forall x [\text{chef}(x) \rightarrow Q x] & \quad \lambda Q \exists y [\text{meal}(y) \land Q y] \\
\end{align*}
\]
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?