Semantics for Natural Languages

Informatics 2A: Lecture 24

Adam Lopez

11 November 2016

Syntax and Semantics

Syntax is concerned with which expressions in a language are well-formed or grammatically correct. This can largely be described by rules that make no reference to *meaning*.
Semantics is concerned with the meaning of expressions: i.e. how they relate to 'the world'. This includes both their

- denotation (literal meaning)
- connotation (other associations)

When we say a sentence is ambiguous, we usually mean it has more than one 'meaning'. (So what exactly are meanings?)

We've already encountered word sense ambiguity and structural ambiguity. We'll also meet another kind of semantic ambiguity, called scope ambiguity. (This already shows that the meaning of a sentence can't be equated with its parse tree.)

Formal and natural language semantics

Providing a semantics for a language (natural or formal) involves giving a systematic mapping from the structure underlying a string (e.g. syntax tree) to its ‘meaning’.

Whilst the kinds of meaning conveyed by NL are much more complex than those conveyed by FLs, they both broadly adhere to a principle called compositionality.

Compositionality

Compositionality: The meaning of a complex expression is a function of the meaning of its parts and of the rules by which they are combined.

While formal languages are designed for compositionality, the meaning of NL utterances can often (not always) be derived compositionally as well.

Compare:

purple armadillo   hot dog

Other desiderata for Meaning Representation

Verifiability: One must be able to use the meaning representation of a sentence to determine whether the sentence is true with respect to some given model of the world.

Example: given an exhaustive table of ‘who loves whom’ relations (a world model), the meaning of a sentence like *everybody loves Mary* can be established by checking it against this model.

Other desiderata for Meaning Representation

Verifiability: One must be able to use the meaning representation of a sentence to determine whether the sentence is true with respect to some given model of the world.

Example: given an exhaustive table of ‘who loves whom’ relations (a world model), the meaning of a sentence like *everybody loves Mary* can be established by checking it against this model.

Question. Suppose we could do this perfectly. Why might it be insufficient for completely understanding natural language?

Desiderata for Meaning Representation

Unambiguity: a meaning representation should be unambiguous, with one and only one interpretation. If a sentence is ambiguous, there should be a different meaning representation for each sense.

Example: each interpretation of *I made her duck* or *time flies like an arrow* should have a distinct meaning representation.

Desiderata for Meaning Representation

Canonical form: the meaning representations for sentences with the same meaning should (ideally) both be convertible into the same canonical form, that shows their equivalence.

Example: the sentence *I filled the room with balloons* should ideally have the same canonical form with *I put enough balloons in the room to fill it from floor to ceiling*.

(The kind of formal semantics we discuss won't achieve this particularly well!)

Desiderata for Meaning Representation

Logical inference: A good meaning representation should come with a set of rules for logical inference or deduction, showing which truths imply which other truths.

E.g. from

Zoot is an armadillo.
Zoot is purple.
Every purple armadillo sneezes.

we should be able to deduce

Zoot sneezes.

Desiderata for Meaning Representation

Expressivity: a meaning representation should allow a wide range of meanings to be expressed in a natural and revealing way, including relationships between the words in a sentence.

Example: we want to express predicate-argument relations, i.e., the participants in the event associated with the head of a phrase:

- *Fred eats lentils* (NP V NP): an *eating* event, with Fred doing the eating (*agent*), and lentils being eaten (*theme*);
- *Fred eats lentils with a fork* (NP V NP with NP): the same, but with a fork as the *instrument* used for eating the lentils.

Propositional Logic

Propositional logic is a very simple system for meaning representation and reasoning in which expressions comprise:

- atomic sentences (P, Q, etc.);
- complex sentences built up from atomic sentences and logical connectives (and, or, not, implies).

Propositional Logic

Why not use propositional logic as a meaning representation system for NL? E.g.

Fred ate lentils or he ate rice. ($P \vee Q$)
Fred ate lentils or John ate lentils ($P \vee R$)

- We're unable to represent the internal structure of the proposition 'Fred ate lentils' (e.g. how its meaning is derived from that of 'Fred', 'ate', 'lentils').
- We're unable to express e.g.

Everyone ate lentils.
Someone ate lentils.

Predicate Logic

First-order predicate logic (FOPL) let us do a lot more (though still only accounts for a tiny part of NL).

Sentences in FOPL are built up from terms made from:

- constant and variable symbols that represent entities;
- predicate symbols that represent properties of entities and relations that hold between entities;
- function symbols (we won't discuss these in this simple introduction).

which are combined into simple sentences (predicate-argument structures) and complex sentences through:

quantifiers ($\forall$, $\exists$)	disjunction ($\vee$)
negation ($\neg$)	implication ($\Rightarrow$)
conjunction ($\wedge$)	equality ($=$)

Constants

Constant symbols:

- **Each constant symbol denotes one and only one entity:** Scotland, Edinburgh, Nicola Sturgeon, 2016
- **Not all entities have a constant that denotes them:** Barack Obama's right knee, this piece of chalk
- **Several constant symbols may denote the same entity:** The Morning Star $\equiv$ The Evening Star $\equiv$ Venus
  National Insurance number, Student ID, your name

Predicates

Predicate symbols:

- Every predicate has a specific arity. E.g. brown/1, country/1, live_in/2, give/3.
- A predicate symbol of arity n is interpreted as a set of n-tuples of entities that satisfy it.
- Predicates of arity 1 denote properties: brown/1.
- Predicates of arity > 1 denote relations: live_in/2, give/3.

Variables

Variable symbols: x, y, z:

- Variable symbols range over entities.
- An atomic sentence with a variable among its arguments, e.g., Part_of(x, EU), only has a truth value if that variable is bound by a quantifier.

Universal Quantifier ($\forall$)

Universal quantifiers can be used to express general truths:

- *Cats are mammals*
- $\forall$x.Cat(x) $\Rightarrow$ Mammal(x)

Intuitively, a universally quantified sentence corresponds to a (possibly infinite) conjunction of sentences:

Cat(sam) $\Rightarrow$ Mammal(sam) $\wedge$ Cat(zoot) $\Rightarrow$ Mammal(zoot) $\wedge$ Cat(fritz) $\Rightarrow$ Mammal(fritz) $\wedge \ldots$

A quantifier has a scope, analogous to scope of PL variables.

Existential Quantifier ($\exists$)

Existential quantifiers are used to express the existence of an entity with a given property, without specifying which entity:

- *I have a cat*
- $\exists x.\mathrm{Cat}(x) \wedge \mathrm{Own}(i, x)$

An existentially quantified sentence corresponds intuitively to a disjunction of sentences:

$$
\begin{aligned}
&(\mathrm{Cat}(\mathrm{Josephine}) \wedge \mathrm{Own}(\mathrm{I}, \mathrm{Josephine})) \vee \\
&(\mathrm{Cat}(\mathrm{Zoot}) \wedge \mathrm{Own}(\mathrm{I}, \mathrm{Zoot})) \vee \\
&(\mathrm{Cat}(\mathrm{Malcolm}) \wedge \mathrm{Own}(\mathrm{I}, \mathrm{Malcolm})) \vee \\
&(\mathrm{Cat}(\mathrm{John}) \wedge \mathrm{Own}(\mathrm{I}, \mathrm{John})) \vee \ldots
\end{aligned}
$$

Existential Quantifier ($\exists$)

Why do we use "$\wedge$" rather than "$\Rightarrow$" with the existential quantifier? What would the following correspond to?

$$\exists x.\text{Cat}(x) \Rightarrow \text{Own}(i, x)$$

(a) I own a cat

(b) There's something such that if it's a cat, I own it.

What if that something isn't a cat?

- The proposition formed by connecting two propositions with $\Rightarrow$ is true if the antecedent (the left of the $\Rightarrow$) is false.
- So this proposition is true if there is something that's e.g. a laptop. But "I own a cat" shouldn't be true simply for this reason.

Abstract syntax of FOPL

The language of first-order predicate logic can be defined by the following CFG (think of it as a grammar for abstract syntax trees). We write F for formulae, AF for atomic formulae, t for terms, v for variables, c for constants.

$$
\begin{array}{rcl}
\mathsf{F} & \rightarrow & \mathsf{AF} \mid \ \mathsf{F} \wedge \mathsf{F} \mid \ \mathsf{F} \vee \mathsf{F} \mid \ \mathsf{F} \Rightarrow \mathsf{F} \mid \ \neg\, \mathsf{F} \\
 & & \mid \ \forall\, \mathsf{v.F} \mid \ \exists\, \mathsf{v.F} \\
\mathsf{AF} & \rightarrow & \mathsf{t{=}t} \mid \ \mathsf{UnaryPred(t)} \mid \ \mathsf{BinaryPred(t,t)} \mid \ \ldots \\
\mathsf{t} & \rightarrow & \mathsf{v} \mid \ \mathsf{c}
\end{array}
$$

Question

Which captures the meaning of *Every dog has a bone*?

1. $\forall x.\exists y.(\text{dog}(x) \land \text{bone}(y) \land \text{has}(x,y))$
2. $\forall x.(\text{dog}(x) \Rightarrow \exists y.(\text{bone}(y) \land \text{has}(x,y)))$
3. $\forall x.\exists y.\text{bone}(y) \land (\text{dog}(x) \Rightarrow \text{has}(x,y))$
4. $\exists y.\forall x.(\text{dog}(x) \Rightarrow (\text{bone}(y) \land \text{has}(x,y)))$

Compositionality and Types

We've asserted that language is compositional. If each word of the sentence *Every dog has a bone* invokes a smaller unit of meaning, and those smaller units are composed to yield

$$\forall x.(\text{dog}(x) \Rightarrow \exists y.(\text{bone}(y) \wedge \text{has}(x, y)))$$

... then we need some way to formally specify how those smaller units (e.g. the predicates dog, bone, and has) are composed to form the complete logical form. We will consider two basic types.

Basic Types

1. e — the type of real-world *entities* (i.e. constants).
2. t — the type of *facts with truth value* (i.e. predicates over bound variables) like 'Inf2a is amusing'.

From these two basic types, we may construct more complex types via the function type constructor.

From basic to complex formal types

Where PL people write $\sigma \to \tau$, NL people often write $< \sigma, \tau >$. E.g.:

- *<e,t>*: **unary predicates** – functions from entities to facts.
- *<e, <e,t>>*: **binary predicates** – functions from entities to unary predicates.
- *<<e,t>, t>*: **type-raised entities** – functions from unary predicates to truth values.

- blue, mr_peanut_butter, gromit **:** *e*
- has **:** *<e, <e,t>>*
- has a bone, is excited **:** *<e,t>*
- blue has a bone, mr_peanut_butter is excited **:** *t*
- every dog **:** *<<e,t>, t>*

This simple system of types will be enough to be going on with (see next lecture). But for more precise semantic modelling, a much richer type system might be required.

Compositionality

Compositionality: The meaning of a complex expression is a function of the meaning of its parts and of the rules by which they are combined.

Do we have sufficient tools to systematically compute meaning representations according to this principle?

- The meaning of a complete sentence will hopefully be a FOPL formula, which we consider as having type t (truth values).
- But the meaning of smaller fragments of the sentence will have other types. E.g.

$$\begin{array}{ll} \textit{has a bone} & < e, t > \\ \textit{every dog} & << e, t >, t > \end{array}$$

- The idea is to show how to associate a meaning with such fragments, and how these meanings combine.
- To do this, we need to extend our language of FOPL with λ expressions ($\lambda =$ lambda; written as \ in Haskell).

Lambda (λ) Expressions

λ-**expressions** are an extension to FOPL that allows us to work with 'partially constructed' formulae. A λ-expression consists of:

- the Greek letter λ, followed by a variable (formal parameter);
- a FOPL expression that may involve that variable.

$\lambda x.sleep(x) \ : \ < e, t >$
'The function that takes an entity x to the statement sleep(x)'

$$\underbrace{(\lambda x.sleep(x))}_{function}\underbrace{(Mary)}_{argument} \ : \ t$$

A λ-expression can be applied to a term.
(The above has the same truth value as $sleep(Mary)$.)

Lambda expressions can be nested. We can use nesting to create functions of several arguments that accept their arguments one at a time.

$\lambda y.\lambda x.$ love(x,y) : $< e, < e, t >>$
'The function that takes y to
(the function that takes x to the statement love(x,y))'

$\lambda z.\lambda y.\lambda x.$ give(x,y,z) : $< e, < e, < e, t >>>$
'The function that takes z to
(the function that takes y to
(the function that takes x to the statement give(x,y,z)))'

Beta Reduction

When a lambda expression applies to a term, a reduction operation (beta (β) reduction) can be used to replace its formal parameter with the term and simplify the result. In general:

$$(\lambda x.M)N \Rightarrow_\beta M[x \mapsto N] \quad (M \text{ with } N \text{ substituted for } x)$$

$$\underbrace{(\lambda x.sleep(x))}_{function}\underbrace{(Mary)}_{argument} \Rightarrow_\beta sleep(Mary)$$

$$\underbrace{(\lambda y.\lambda x.love(x,y))}_{function}\underbrace{(crabapples)}_{argument} \Rightarrow_\beta \lambda x.love(x, crabapples)$$

$$\underbrace{(\lambda x.love(x, crabapples))}_{function}\underbrace{(Mary)}_{argument} \Rightarrow_\beta love(Mary, crabapples)$$

Summary

- Principle of compositionality: the meaning of an complex expression is a function of the meaning of its parts.
- Predicate logic can be used to give meaning representations for a certain portion of natural language.
- λ-expressions can be used to represent meanings for *fragments of* sentences.
- In the next lecture, we'll see how the meaning of sentences can be systematically derived in a compositional way using such λ-expressions.