

# Semantics for Natural Languages

## Informatics 2A: Lecture 23

Shay Cohen

13 November 2015

## 1 Introduction

- Syntax and Semantics
- Compositionality
- Desiderata for Meaning Representation

## 2 Logical Representations

- Propositional Logic
- Predicate Logic

## 3 Semantic Composition

- Compositionality
- Lambda Expressions

**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.)

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:** 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.

**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.

**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).

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.

**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** (won't bother with these here).

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 ( $=$ )

# Predicate Logic Example

## Constant symbols:

- **Each constant symbol denotes one and only one entity:**  
Scotland, Aviemore, EU, Barack Obama, 2007
- **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

## 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.

Variable symbols:  $x$ ,  $y$ ,  $z$ :

- Variable symbols range over entities.
- An atomic sentence with a variable among its arguments, e.g.,  $\text{Part\_of}(x, \text{EU})$ , 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*
- $\forall x. \text{Cat}(x) \Rightarrow \text{Mammal}(x)$

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

$\text{Cat}(\text{sam}) \Rightarrow \text{Mammal}(\text{sam}) \wedge \text{Cat}(\text{zoot}) \Rightarrow \text{Mammal}(\text{zoot})$   
 $\wedge \text{Cat}(\text{fritz}) \Rightarrow \text{Mammal}(\text{fritz}) \wedge \dots$

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*
- $\exists x. \text{Cat}(x) \wedge \text{Own}(i, x)$

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

$(\text{Cat}(\text{Josephine}) \wedge \text{Own}(I, \text{Josephine})) \vee$   
 $(\text{Cat}(\text{Zoot}) \wedge \text{Own}(I, \text{Zoot})) \vee$   
 $(\text{Cat}(\text{Malcolm}) \wedge \text{Own}(I, \text{Malcolm})) \vee$   
 $(\text{Cat}(\text{John}) \wedge \text{Own}(I, \text{John})) \vee \dots$

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.

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.

$$\begin{aligned} F &\rightarrow AF \mid F \wedge F \mid F \vee F \mid F \Rightarrow F \mid \neg F \\ &\quad \mid \forall v.F \mid \exists v.F \\ AF &\rightarrow t=t \mid \text{UnaryPred}(t) \mid \text{BinaryPred}(t,t) \mid \dots \\ t &\rightarrow v \mid c \end{aligned}$$

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

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

(N.B. The logical form looks structurally quite different from the parse tree for the original sentence. So there's some real work to be done!)

**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.

*has a bone*     $\langle e, t \rangle$

*every dog*     $\langle \langle e, t \rangle, t \rangle$

- 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  **$\lambda$  expressions** ( $\lambda =$  lambda; written as `\` in Haskell).

# Lambda ( $\lambda$ ) Expressions

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

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

$\lambda x.sleep(x) : \langle e, t \rangle$

'The function that takes an entity  $x$  to the statement  $sleep(x)$ '

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

A  $\lambda$ -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.

$\lambda y. \lambda x. \text{love}(x,y) \quad : \quad \langle e, \langle e, t \rangle \rangle$

'The function that takes y to  
(the function that takes x to the statement  $\text{love}(x,y)$ )'

$\lambda z. \lambda y. \lambda x. \text{give}(x,y,z) \quad : \quad \langle e, \langle e, \langle e, t \rangle \rangle \rangle$

'The function that takes z to  
(the function that takes y to  
(the function that takes x to the statement  $\text{give}(x,y,z)$ ))'



# Beta Reduction

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

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

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

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

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

- 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.
- $\lambda$ -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  $\lambda$ -expressions.