Advanced Natural Language Processing
Lecture 22: Computational Semantics (I)

Frank Keller (based on slides by Bonnie Webber, Mirella Lapata, and others)

Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
http://homepages.inf.ed.ac.uk/keller/

11 November 2011

Syntax and Semantics

Semantics is concerned with how expressions in a language map to a world – both their

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

When we say (in everyday usage) that a sentence is ambiguous, we usually mean it has more than one (literal) meaning.

Some ambiguity comes from words having more than one sense, **some** from sentences having more than one parse tree (syntactic analysis) with respect to a grammar, and **some** from a property called *scope*.

Syntax and Semantics

A possible 'meaning' for a sentence should take account of both the intended senses of its words and its intended syntactic analysis. Take the example:

I made her duck

- I caused her to drop and avert her head. (*duck* as action)
- I created the duck that she owns. (*duck* as individual)
- I cooked a/some duck for her. (*duck* as mass)

Syntax and Semantics

Providing a semantics for a language (natural or formal) involves giving a systematic mapping from the structure underlying a string to its 'meaning'.

While the kinds of meaning conveyed by NL are generally much more complex than those conveyed formal languages, they both adhere to the principle of 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 literal meaning of NL utterances can often be derived compositionally as well.

Verifiability

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.

Unambiguous

Unambiguous: 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.

Canonical Form

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

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

Relationships other than identity should be derivable by entailment and other forms of inference.

Expressivity

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 restrictions on the concept denoted by the head of a phrase:

- *brown cow* (How is *brown* related to *cow*?)
- *man who came to dinner* (or *man* related to *came to dinner*?)
- *walk briskly* (or *walk* related to *briskly*?)

Expressivity

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 one system for 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, etc.).

Propositional Logic

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

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

We lose any obvious relationship between the clauses that make up these sentences.

Everyone ate lentils. $(P1 \land P2 \land P3 \land P4 \ldots)$
Someone ate lentils. $(P1 \lor P2 \lor P3 \lor P4 \ldots)$

We can't really express either sentence.

Predicate Logic

First-order predicate logic (FOPL) is closer to being expressive enough for NL semantics.

Sentences in FOPL are built up from terms made from:

- constant and variable symbols that represent entities;
- function symbols that allow us to indirectly specify entities;
- predicate symbols that represent properties of entities and relations that hold between entities;

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, Perth, EU, John, George W. Bush, Scotland, 2007
- **Not all entities have a constant that denotes them:**
  George W. Bush's right knee, this pen
- **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:** Brown/1, Country/1, Live_in/2, Give/3.
- Each 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)

Universally quantified sentence corresponds to a conjunction of sentences in which a constant substitutes for a variable.

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

A quantifier has a scope, defined as what depends on it.

Existential Quantifier (∃)

Existential quantifier is used to express that a property/relation holds of some entity, without specifying which one:

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

An existentially quantified sentence corresponds to disjunction of sentences in which a constant substitutes for a variable.

$$(Cat(Josephine) \wedge Own(I, Josephine)) \vee$$
$$(Cat(Zoot) \wedge Own(I, Zoot)) \vee$$
$$(Cat(Malcolm) \wedge Own(I, Malcolm)) \vee$$
$$(Cat(John) \wedge Own(I, John)) \vee \ldots$$

Existential Quantifier ($\exists$)

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

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

(a) I own a cat

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

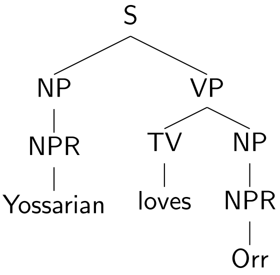
What if that something is not a cat?

- 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 a laptop, for example: "I own a cat" shouldn't be true simply for this reason.

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?



- If loves is the binary predicate love(x,y) and Orr is orr, how do we combine them to produce an interpretation loves Orr?
- To compute NL interpretations compositionally, we need lambda expressions (λ-expressions).

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.

$\lambda x.sleep(x)$	'The function that takes an entity x to the statement sleep(x)'
$\underbrace{(\lambda x.sleep(x))}_{functor} \underbrace{(orr)}_{argument}$	A λ-expression can be applied to a term has the same truth value as $sleep(orr)$

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)	'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)	'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.

$$\underbrace{(\lambda x.sleep(x))}_{functor}\ \underbrace{(orr)}_{argument} \text{ simplifies to } \Rightarrow_\beta sleep(orr)$$

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

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

Summary

- Principle of compositionality: the meaning of an complex expression is a function of the meaning of its parts;
- predicate logic can be used as a meaning representation language for natural language;
- λ-expressions can be used to compute meaning representations from syntactic trees based on the principle of compositionality;
- in the next lecture, we will see how a probabilistic model can be learned that automates this mapping.