Michael Yoshitaka Erlewine & Hadas Kotek September 10, 2014

Review: Compositional semantics

Today: Some syntactic and semantic assumptions and background.

1 The principle of compositionality

The goal of the semantics we develop here: describe the meanings of sentences.
The meanings of sentences are described in terms of their ***truth conditions***: the situations under which the sentence is true and the situations under which the sentence is false.

The principle of compositionality: the meaning of a complex expression depends upon its constituent parts and the way they are combined.

→ Linguistic expressions contribute to the meaning of a sentence in systematic and predictable ways.

(1) The cat chased the rat.
(2) The grey cat chased the rat.
(3) The cat chased the dog.
(4) The cat licked the rat.
(5) A cat chased the rat.

2 A model theoretic approach to sentence meanings

Below is a simple model for a (small, cat-centric) fragment of the English language.

Syntax

Vocabulary (lexicon)
N → John, Mary, Tonia
V_i → smokes, purrs, is hungry
V_{tr} → hugs
Neg → not
Conj → and, or

Phrase structure rules
S → N VP
VP → V_i
VP → V_{tr} N
S → S Conj S
S → Neg S

Semantics

Semantic values of the basic expressions:

$$[\![\text{smokes}]\!] = \begin{bmatrix} \text{John} \mapsto 1 \\ \text{Mary} \mapsto 1 \\ \text{Mitzi} \mapsto 0 \end{bmatrix}$$

$$[\![\text{purrs}]\!] = \begin{bmatrix} \text{John} \mapsto 0 \\ \text{Mary} \mapsto 0 \\ \text{Mitzi} \mapsto 1 \end{bmatrix}$$

$$[\![\text{is hungry}]\!] = \begin{bmatrix} \text{John} \mapsto 0 \\ \text{Mary} \mapsto 1 \\ \text{Mitzi} \mapsto 1 \end{bmatrix}$$

$$[\![\text{likes}]\!] = \begin{bmatrix} \text{<John, Mitzi>} \mapsto 1 \\ \text{<Mary, Mitzi>} \mapsto 1 \\ \text{<Mitzi, Mary>} \mapsto 1 \end{bmatrix}$$
$$\text{0 otherwise}$$

$$[\![\text{John}]\!] = \text{John}$$
$$[\![\text{Mary}]\!] = \text{Mary}$$
$$[\![\text{Mitzi}]\!] = \text{Mitzi}$$

$$[\![\text{not}]\!] = \begin{bmatrix} 1 \mapsto 0 \\ 0 \mapsto 1 \end{bmatrix}$$

$$[\![\text{and}]\!] = \begin{bmatrix} <1,1> \mapsto 1 \\ <1,0> \mapsto 0 \\ <0,1> \mapsto 0 \\ <0,0> \mapsto 0 \end{bmatrix}$$

$$[\![\text{or}]\!] = \begin{bmatrix} <1,1> \mapsto 1 \\ <1,0> \mapsto 1 \\ <0,1> \mapsto 1 \\ <0,0> \mapsto 0 \end{bmatrix}$$

Recursive Truth Definition (Semantic rules):

1. $[\![[_X\ \alpha\ ]]\!] = [\![\alpha]\!]$, where X is a category, and α is a lexical item.
2. $[\![[_{VP}\ \text{V}_i\ ]]\!] = \{\ x : [\![\text{V}_i]\!](x) = 1\ \}$
3. $[\![[_{VP}\ \text{V}_{tr}\ \text{N}\ ]]\!] = \{\ x : [\![\text{V}_{tr}]\!](<x, [\![\text{N}]\!]>) = 1\ \}$
4. $[\![[_S\ \text{N VP}\ ]]\!] = 1$ *iff* $[\![\text{N}]\!] \in [\![\text{VP}]\!]$, 0 otherwise
5. $[\![[_S\ \text{neg}\ \text{S}_1\ ]]\!] = [\![\text{neg}]\!]([\![\text{S}_1]\!])$
6. $[\![[_S\ \text{S}_1\ \text{Conj}\ \text{S}_2\ ]]\!] = [\![\text{Conj}]\!](<[\![\text{S}_1]\!], [\![\text{S}_2]\!]>)$

Exercise: Draw a syntactic tree and derive the truth conditions of the following sentences:

(6) Mary is not hungry. (Model as: "not Mary is hungry")

(7) John smokes or Mary is hungry.

(8) Mitzi purrs and Mary likes John.

3 Types, sets and functions

The system above has the disadvantage of having to specify specific interpretation rules for different lexical categories. Instead we can use a more general procedure, where we describe the meaning of natural language expressions as denoting functions.[1] This allows us to use a general rule for composition.

We will use lambda calculus.

- **Basic types:**
  - e for individuals, in D_e
  - t for truth values, in $\{0, 1\}$
- **Definition:** If σ and τ are types, then $\langle\sigma, \tau\rangle$ is a type. An object of type $\langle\sigma, \tau\rangle$ is a function which takes an argument of type σ and returns an object of type τ

Exercise: Give examples for elements of the following types:

(9) e

(10) $\langle e, t\rangle$

(11) $\langle e, \langle e, t\rangle\rangle$

(12) $\langle e, \langle e, \langle e, t\rangle\rangle\rangle$

We will write functions (almost exclusively) using λ notation: "λx. <*something involving x*>" takes an argument, x, and returns the <*something involving x*>.

(Variables optionally have a subscript to indicate the expected type.)

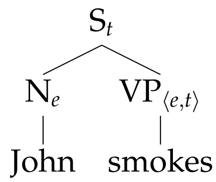
This allows us to represent more complex Common Noun Phrases, Adjective Phrases, and Verb Phrases. Examples:

(13) $\lambda x.\ x$ smokes : the function that maps every individual x in D_e to 1 if x smokes, and to 0 otherwise.

(14) $\lambda x.\ \lambda y.\ y$ likes x : the function that maps every individual x in D_e to the function that maps every individual y in D_e to 1, if 1 if y likes x, and to 0 otherwise.

Definition: Functional Application (FA)
If α has β and γ as its daughter constituents and $[\![\beta]\!] \in D_\sigma$ and $[\![\gamma]\!] \in D_{\langle\sigma,\tau\rangle}$,
then $[\![\alpha]\!] = [\![\gamma]\!]([\![\beta]\!])$

[1] In particular, functions that take one argument at a time, so no arguments that take pairs of arguments like we had before.

(15) 

$[\![S]\!] = [\![\text{smokes}]\!]([\![\text{John}]\!])$ by FA

$= \underbrace{[\lambda x.x \text{ smokes}]}_{\text{function}}\ \underbrace{(\text{John})}_{\text{argument}}$ Terminal Nodes

$= 1$ *if and only if* John smokes

Exercise: Does the function: $\lambda x.\ \lambda y.\ x$ likes y mean the same thing as the function in (14): $\lambda y.\ \lambda x.\ x$ likes y?

Exercise: Write lexical entries for:

(16) *give*

(17) *not*

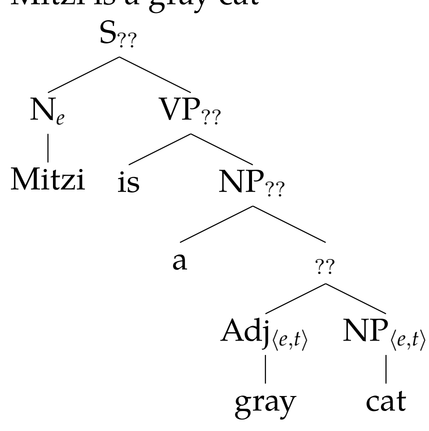
(18) *gray*

(19) *slowly*

(20) Passive *-en* (that takes a by-phrase)

How does a predicate such as 'gray' combine with a predicate like 'cat'?

(21) Mitzi is a gray cat



Definition: Predicate Modification
If α is a branching node that has β and γ as its daughter constituents and $[\![\beta]\!]$ and $[\![\gamma]\!]$ are both $\in D_{\langle e,t\rangle}$, then $[\![\alpha]\!] = \lambda x.[\![\beta]\!](x) = [\![\gamma]\!](x) = 1$

Exercise: Compute the meaning of (21).