LAMBDA CALCULUS: A BRIEF INTRODUCTION

Language, Logic, and computation

Yanjing Wang, Department of Philosophy

Dec. 12th, 2017

Recap

Lambda-calculus

TWO PHILOSOPHICAL IDEAS ABOUT MEANING

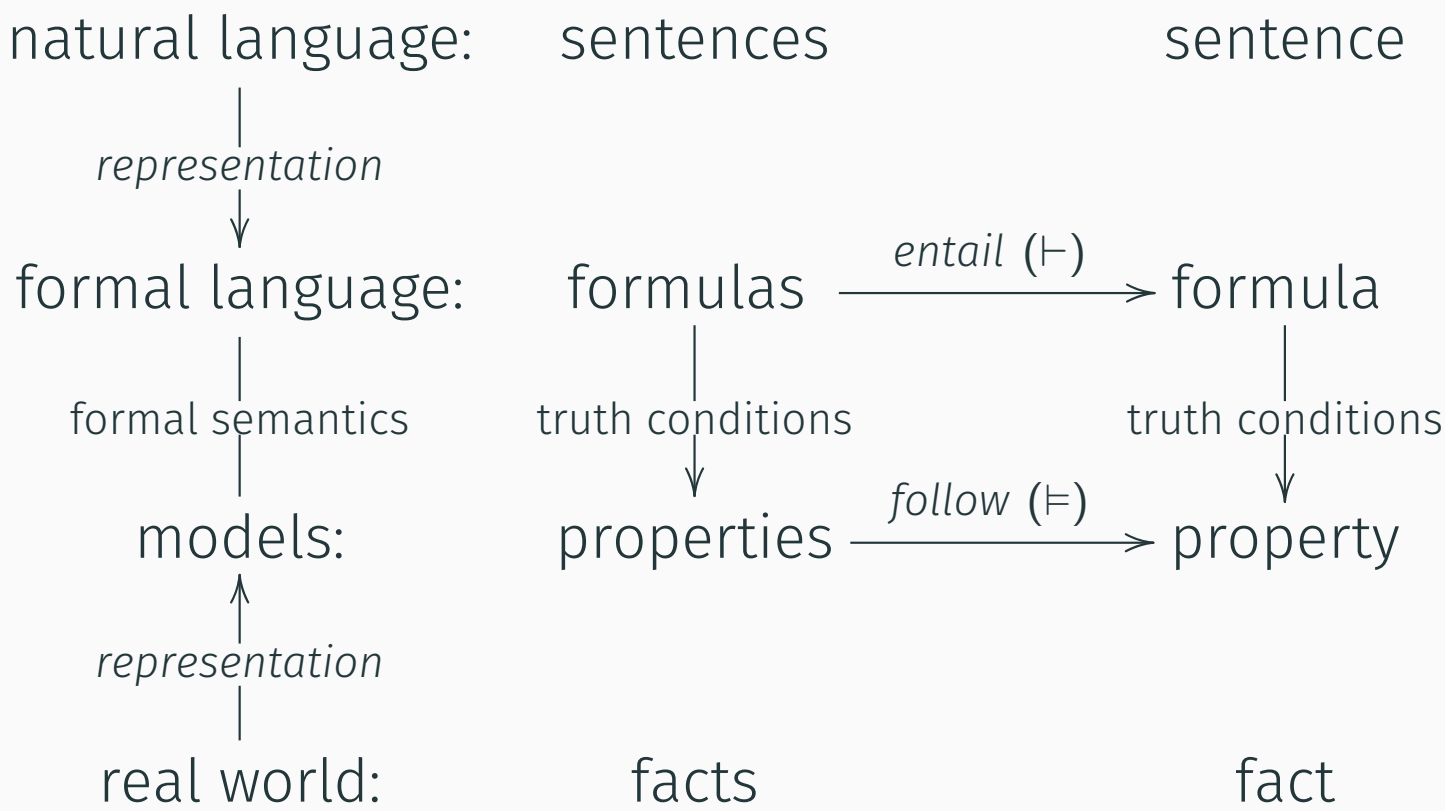
In philosophy of language:

- Meaning as reference and sense (Frege)
  - Formal semantics: care more about logical /grammatical words
- Meaning as use in contexts (Wittgenstein)
  - Pragmatics
  - Distributional semantics: care more about content words

The reality is mixed: how much is its meaning and how much is its use (in contexts)?

Recap

THE GENERAL PICTURE



FIRST-ORDER LANGUAGES

A *signature* (符号表) gives the *non-logical* symbols:

- variables (变元) $x, y \cdots \in Var$
- constants (常元) $c, d \cdots \in Con$
- predicate symbols (谓词符号) $P, Q \ldots$ each has an arity
- function symbols (函数符号) $f, g \ldots$ each has an arity

Given this signature the first-order language is defined as:

$$\begin{aligned} \text{term (项) } t &::= x \mid c \mid f(\vec{t}) \quad \text{where } x \in Var, c \in Con \\ \text{formula } \varphi &::= t \equiv t \mid P\vec{t} \mid \neg\varphi \mid (\varphi \to \varphi) \mid \forall x\varphi \end{aligned}$$

We can define $\exists x$ as $\neg\forall x\neg$.

EXAMPLE MODEL

Let $D = \{Alice, Bill, Charles\}$ and I be defined as follows:

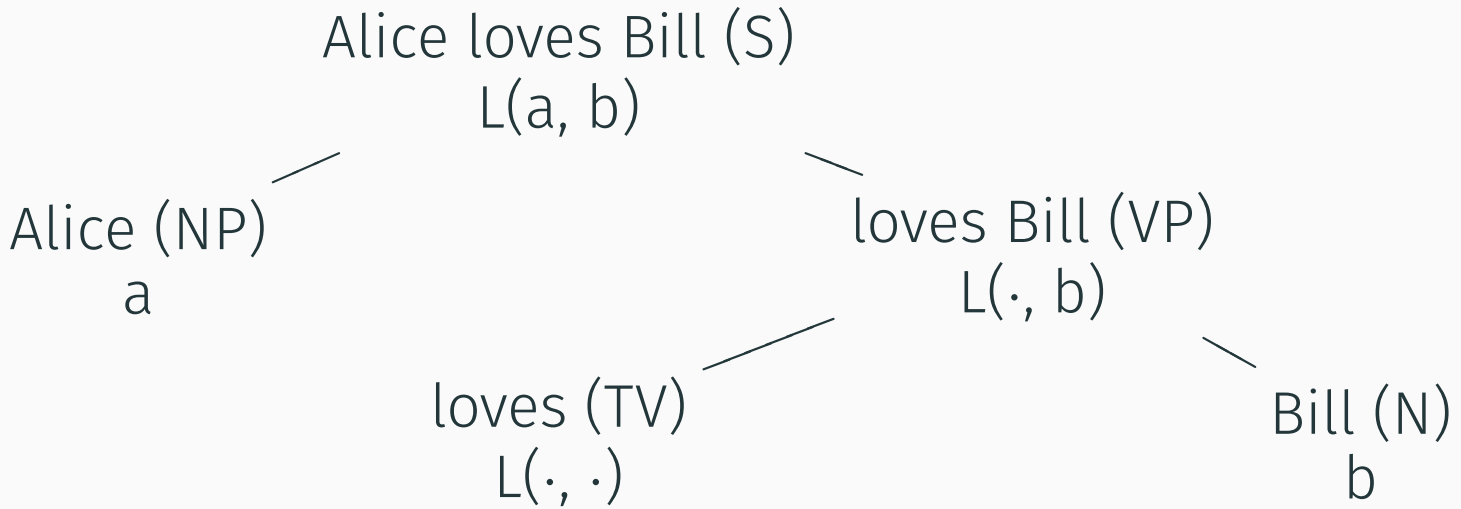
non-logical symbols	Interpretation
constant a, b, c	Alice, Bill, Charles
one-place predicate H, R	$\{Charles, Bill\}, \{Alice, Charles\}$
two-place predicate L	$\{\langle Alice, Bill\rangle, \langle Charles, Alice\rangle, \langle Bill, Bill\rangle\}$

We also use $[\![a]\!]$, $[\![P]\!]$ to denote the interpretations given a model. Check the truth value of following formulas:

- $L(a, b) \wedge \neg L(b, a)$.
- $\neg\forall x(R(x) \rightarrow H(x))$, equiv. $\exists x(R(x) \wedge \neg H(x))$.
- $L(a, b) \wedge \forall x(L(x, b) \rightarrow x \equiv a)$ (false)
- $\forall x(H(x) \rightarrow \exists y(L(x, y)))$

BACK TO NL: TROUBLES WITH FIRST-ORDER LANGUAGE

- How to represent the parts of an NL sentence?
- How to go from meaning of lexicon and the syntactic structure step by step to the logical form?



$L(\cdot, b)$ is not a formula or a term in first-order language. What are they exactly? How to compose them to form a formula?

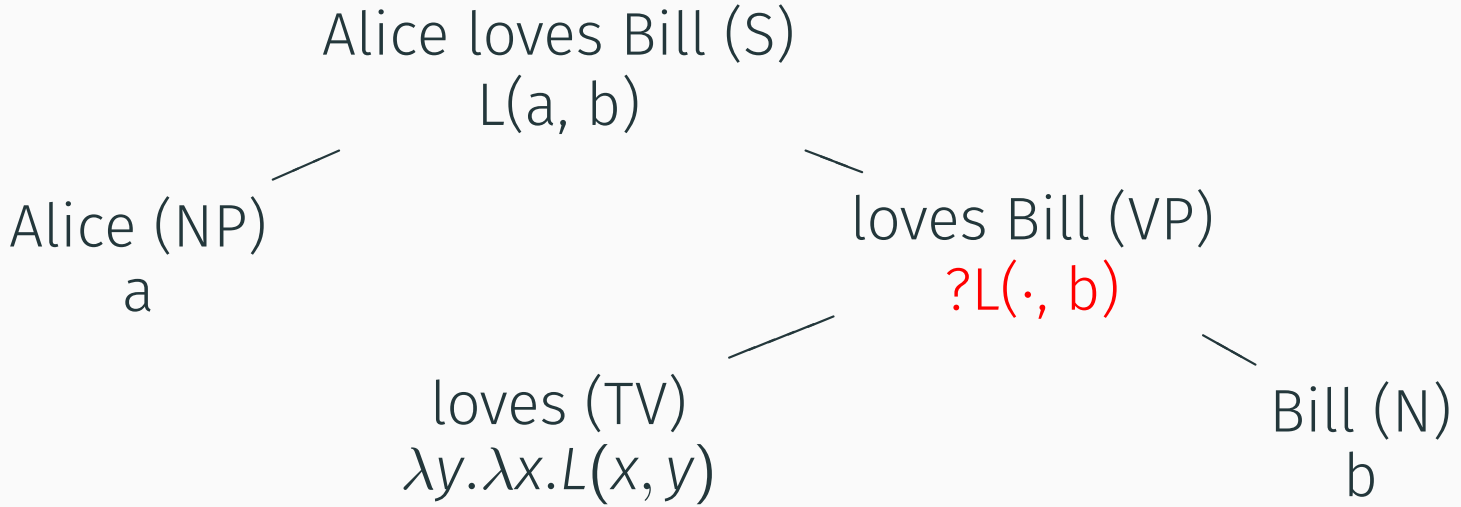
Lambda-calculus

SOME BASIC IDEAS

From a predicate to a function

- Recall the example model.
- $[\![R]\!] = \{Alice, Charles\}$ which is a subset of the domain D.
- A subset can be viewed as its *characteristic function* $f : D \to \{0, 1\}$ such that $f(x) = 1 \iff x \in [\![R]\!]$.
- R can be viewed as a function without a name: $\lambda x.R(x)$
- Occurrence of variables bound by λ are *place holders*.
- Similarly for $\lambda y.\lambda x.L(x, y)$
- In general we can build functions from terms with already defined functions: $\lambda x.(x + 7)$

BACK TO NL



- functional application: $\lambda y.\lambda x.L(x,y)(b)$
- after substitution (β-conversion): $\lambda x.L(x,b)$
- similarly $\lambda x.L(x,b)(a)$ gives $L(a,b)$ after β-conversion
- sometimes we need to be more careful: $\lambda y.\lambda x.L(x,y)(x)$? It gives $\lambda x.L(x,x)$. We can first rename the bound variables (α-conversion): $\lambda y.\lambda z.L(z,y)(x)$ gives $\lambda z.L(z,x)$

MORE FORMAL DEFINITION

GIven a set of variables, the lambda-terms are defined as:

- a variable, x, is itself a lambda term
- if t is a lambda term, and x is a variable, then $(\lambda x.t)$ is a λ-term (called a *lambda abstraction*);
- if t and s are lambda terms, then (ts) is a λ-term (called an application).
- only those constructed by the above rules are λ-terms.

Two main conversions

- α-conversion: $(\lambda x.t[x]) \implies (\lambda y.t[y])$
- β-conversion: $(\lambda x.t)(s) \implies t[s/x]$

ENCODE NUMBERS AND TRUTH VALUES

Lambda calculus is a model of computation (like Turing machine).

Encode natural numbers:

$$0 := \lambda f.\lambda x.x \qquad 1 := \lambda f.\lambda x.fx \quad 2 := \lambda f.\lambda x.f(fx) \quad \ldots$$

$$S := \lambda n.\lambda f.\lambda x.f(nfx) \quad S(n) = n + 1$$

Note: function applications are left associative: stx is $(st)x$
Encode truth values and boolean connectives:

$$TRUE := \lambda x.\lambda y.x \quad FALSE := \lambda x.\lambda y.y \quad NOT := \lambda p.p\ FALSE\ TRUE$$

Y-COMBINATOR AND RECURSION

The functions are anonymous, how can we define recursion?

$$Y = \lambda f.((\lambda x.f\ (x\ x))\ (\lambda x.f\ (x\ x)))$$

It is the way to do recursion in λ-calculus.

$$\begin{aligned} Y\ g &= (\lambda f.((\lambda x.f\ (x\ x))\ (\lambda x.f\ (x\ x))))\ g \\ &= (\lambda x.g\ (x\ x))\ (\lambda x.g\ (x\ x)) \\ &= g((\lambda x.g\ (x\ x))\ (\lambda x.g\ (x\ x))) \\ &= g\ (Y\ g) \end{aligned}$$

FOL WITH TYPED-LAMBDA TERMS

Recall the types: e (entities) and t (truth values), the corresponding domain $D_e = \{Alice, Bob, \dots\}$ and $D_t = \{0, 1\}$.

- sentences are of type t
- constants are of type e
- other types are built by using existing ones: e.g., $(e \to t)$

We can index every term with the corresponding type.

word	type	FO-lambda term	meaning
alice	e	a	Alice
walk	$(e \to t)$	$\lambda x_e.(W(x))_t$	{Alice, Charles}
love	$(e \to (e \to t))$	$\lambda y_e.\lambda x_e.(L(x, y))_t$	$\{\langle Alice, Bob\rangle, \langle Bob, Bob\rangle\}$

WHAT ABOUT QUANTIFIED NP?

- Every happy one is also rich: $\forall x(H(x) \rightarrow R(x))$.
- Some rich one is not happy: $\exists x(R(x) \wedge \neg H(x))$.

How do we compute the meaning from the lexicon? E.g., what is the meaning of *every happy one*?

$$\begin{aligned} [\![happy]\!] &= \{Bob, Charles\} \\ [\![rich]\!] &= \{Alice, Bob, Charles\} \\ [\![thin]\!] &= \{Alice\} \end{aligned}$$

The meaning of *every happy one*:

$$\{Y \mid [\![H]\!] \subseteq Y\} = \{\{Bob, Charles\}, \{Alice, Bob, Charles\}\}$$

The corresponding lambda term is $\lambda Y_{(e \rightarrow t)}.(\forall x(H(x) \rightarrow Y(x)))_t$

CONNECTIONS BETWEEN FREGE AND WITTGENSTEIN

An idea from structuralism: the meaning (sense) of a word comes from its relationship with other words in use.

Merriam-Webster Dictionary (2013):

- Oak (1a): “any of a genus (Quercus) of trees or shrubs of the beech family that produce **acorns**”
- Acorn: “the nut of the **oak tree**”

Distributional semantics!