Semantic Theory
Week 4 – Typed Lambda Calculus

Noortje Venhuizen
Harm Brouwer

Universität des Saarlandes

Summer 2019

Compositionality

The principle of compositionality: “The meaning of a complex expression is a function of the meanings of its parts and of the syntactic rules by which they are combined” (Partee et al.,1993)

Compositional semantics construction:

- compute meaning representations for sub-expressions
- combine them to obtain a meaning representation for a complex expression.

Problematic case: “Not smoking$_{\langle e,t \rangle}$ [is healthy]$_{\langle\langle e,t \rangle ,t \rangle}$”

$$\begin{tikzcd}\text{Not} \arrow[r, bend right, "!"'] & \text{smoking}\end{tikzcd}$$

Lambda abstraction

λ-abstraction is the operation that transforms expressions of any type τ into a function $\langle\sigma,\tau\rangle$, where σ is the type of the λ-variable.

Formal definition:

If α is in WE_{τ}, and x is in VAR_{σ} then $\lambda x(\alpha)$ is in $WE_{\langle\sigma, \tau\rangle}$

- The scope of the λ-operator is the smallest WE to its right. Wider scope must be indicated by brackets.

- We often use the “dot notation” $\lambda x.\phi$ indicating that the λ-operator takes widest possible scope (over ϕ).

Interpretation of Lambda-expressions

If $\boldsymbol{\alpha} \in WE_\tau$ and $v \in VAR_\sigma$, then $[\![\lambda v \boldsymbol{\alpha}]\!]^{M,g}$ is that function $f : D_\sigma \rightarrow D_\tau$ such that for all $a \in D_\sigma$, $f(a) = [\![\boldsymbol{\alpha}]\!]^{M,g[v/a]}$

If the λ-expression is applied to some argument, we can simplify the interpretation:

- $[\![\lambda v \boldsymbol{\alpha}]\!]^{M,g}(x) = [\![\boldsymbol{\alpha}]\!]^{M,g[v/x]}$

Example: *"Bill is a non-smoker"*

$[\![\lambda x(\neg S(x))(b')]\!]^{M,g} = 1$

iff $[\![\lambda x(\neg S(x))]\!]^{M,g}([\![b']\!]^{M,g}) = 1$

iff $[\![\neg S(x)]\!]^{M,g'} = 1$ where $g'=g[x/[\![b']\!]^{M,g}]$

iff $[\![S(x)]\!]^{M,g'} = 0$

iff $[\![S]\!]^{M,g'}([\![x]\!]^{M,g'}) = 0$

iff $V_M(S)(V_M(b')) = 0$

β-Reduction

$[\![\lambda v(\boldsymbol{\alpha})(\boldsymbol{\beta})]\!]^{M,g} = [\![\boldsymbol{\alpha}]\!]^{M,g[v/[\![\beta]\!]^{M,g}]}$

⇒ all (free) occurrences of the λ-variable in **α** get the interpretation of **β** as value.

This operation is called β-reduction

- **λv**(**α**)(**β**) ⇔ **α**[**β**/**v**]
- **α**[**β**/**v**] is the result of replacing all free occurrences of **v** in **α** with **β**

Achtung: The equivalence is not unconditionally valid!

Variable capturing

Q: Are $\lambda v(\boldsymbol{\alpha})(\boldsymbol{\beta})$ and $\boldsymbol{\alpha}[\boldsymbol{\beta}/v]$ always equivalent?

- $\lambda x(\text{drive'}(x) \wedge \text{drink'}(x))(j') \Leftrightarrow \text{drive'}(j') \wedge \text{drink'}(j')$

- $\lambda x(\text{drive'}(x) \wedge \text{drink'}(x))(y) \Leftrightarrow \text{drive'}(y) \wedge \text{drink'}(y)$

- $\lambda x(\forall y\ \text{know'}(x)(y))(j') \Leftrightarrow \forall y\ \text{know}(j')(y)$

- $\lambda x(\forall y\ \text{know'}(x)(y))(y) \not\Leftrightarrow \forall y\ \text{know}(y)(y)$

Let v, v' be variables of the same type, and let **α** be any well-formed expression.

- v is free for v' in **α** iff no free occurrence of v' in **α** is in the scope of a quantifier or a λ-operator that binds v.

Conversion rules

- β-conversion: $\lambda v(\boldsymbol{\alpha})(\boldsymbol{\beta}) \Leftrightarrow \boldsymbol{\alpha}[\boldsymbol{\beta}/v]$

  (if all free variables in **β** are free for v in **α**)

- α-conversion: $\lambda v.\boldsymbol{\alpha} \Leftrightarrow \lambda w.\boldsymbol{\alpha}[w/v]$

  (if w is free for v in **α**)

- η-conversion: $\lambda v.\boldsymbol{\alpha}(v) \Leftrightarrow \boldsymbol{\alpha}$

Determiners as lambda-expressions

- a student works ➔ $\exists x(\text{student}'(x) \wedge \text{work}'(x)) :: t$
  - a student ➔ $\lambda P \exists x(\text{student}'(x) \wedge P(x)) :: \langle\langle e,t\rangle,t\rangle$
  - a, some ➔ $\lambda Q \lambda P \exists x(Q(x) \wedge P(x)) :: \langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$
- every student ➔ $\lambda P \forall x(\text{student}'(x) \rightarrow P(x)) :: \langle\langle e,t\rangle,t\rangle$
  - every ➔ $\lambda Q \lambda P \forall x(Q(x) \rightarrow P(x)) :: \langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$
- no student ➔ $\lambda P \neg\exists x(\text{student}(x) \wedge P(x)) :: \langle\langle e,t\rangle,t\rangle$
  - no ➔ $\lambda Q \lambda P \neg\exists x(Q(x) \wedge P(x)) :: \langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$
- someone ➔ $\lambda F \exists x F(x) :: \langle\langle e,t\rangle,t\rangle$

NL Quantifier Expressions: Interpretation

- someone’$\in$ CON$_{\langle\langle e,t\rangle,t\rangle}$, so V_M(someone’) $\in D_{\langle\langle e,t\rangle,t\rangle}$

- $D_{\langle\langle e,t\rangle,t\rangle}$ is the set of functions from $D_{\langle e,t\rangle}$ to D_t , i.e.,

  the set of functions from $\mathcal{P}(U_M)$ to $\{0,1\}$,

  which in turn is equivalent to $\mathcal{P}(\mathcal{P}(U_M))$

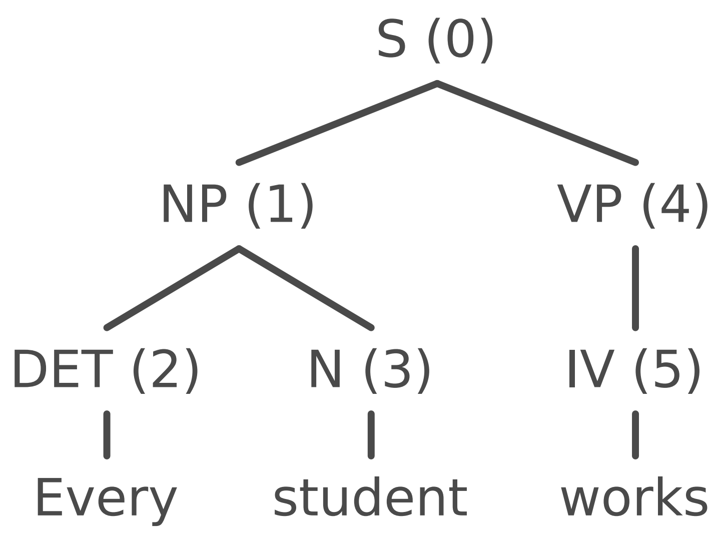
- Thus, V_M(someone’) $\subseteq \mathcal{P}(U_M)$. More specifically:

- V_M(someone’) = $\{S \subseteq U_M \mid S \neq \varnothing\}$, if U_M is a domain of persons

$\Rightarrow$ More on Natural Language Quantifiers next week!

β-Reduction Example

Every student works.



(2) $\lambda P \lambda Q \forall x(P(x) \rightarrow Q(x)) :: \langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle$

(3) $\lambda x.\text{student'}(x) \Leftrightarrow^{\eta} \text{student'} :: \langle e, t\rangle$

(1) $\lambda P \lambda Q \forall x(P(x) \rightarrow Q(x))(\text{student'})$
$\Leftrightarrow^{\beta} \lambda Q \forall x(\text{student'}(x) \rightarrow Q(x)) :: \langle\langle e, t\rangle, t\rangle$

(4)/(5) $\lambda x.\text{work'}(x) \Leftrightarrow^{\eta} \text{work'} :: \langle e, t\rangle$

(0) $\lambda Q \forall x(\text{student'}(x) \rightarrow Q(x))(\text{work'}) \Leftrightarrow^{\beta} \forall x(\text{student'}(x) \rightarrow \text{work'}(x)) :: t$

Transitive Verbs: Type Clash

- Someone reads a book

$$\dfrac{\text{someone} :: \langle\langle e, t\rangle,t\rangle \qquad \dfrac{\text{read} :: \langle e,\langle e, t\rangle\rangle \qquad \text{a book} :: \langle\langle e, t\rangle,t\rangle}{?? :: ??}}{?? :: t}$$

Solution: reverse functor-argument relation (again)

$\text{read}_{\langle\langle\langle e,\ t\rangle,t\rangle,\langle e,\ t\rangle\rangle}$ (*Type Raising*)

Type Raising

It's not enough to just change the type of the transitive verb:

- read ➔ read' $\in \mathrm{CON}_{\langle\langle\langle e,t\rangle, t\rangle, \langle e, t\rangle\rangle}$

  someone reads a book:
  $\lambda F \exists x F(x)(\mathrm{read}'(\lambda P \exists y(\mathrm{book}'(y) \wedge P(y)))$
  $\Leftrightarrow^{\beta} \exists x(\mathrm{read}'(\lambda P \exists y(\mathrm{book}'(y) \wedge P(y)))(x)$

  …but this does not support the following entailment:
  someone reads a book ⊨ *there exists a book*

We need a more explicit λ-term:

- read ➔ $\lambda Q \lambda z.Q(\lambda x(\mathrm{read}^*(x)(z))) \in \mathrm{WE}_{\langle\langle\langle e,t\rangle, t\rangle, \langle e, t\rangle\rangle}$
  where: $\mathrm{read}^* \in \mathrm{WE}_{\langle e, \langle e, t\rangle\rangle}$ is the "underlying" first-order relation

Transitive Verbs: example

someone reads a book

$\lambda F \exists x F(x)(\lambda Q \lambda z.Q(\lambda x(read^*(x)(z)))(\lambda R \lambda P.\exists y(R(y) \wedge P(y))\ (book')))$

$\Leftrightarrow_\beta\ \lambda F \exists x F(x)(\lambda Q \lambda z.Q(\lambda x(read^*(x)(z)))(\lambda P.\exists y(book'(y) \wedge P(y))))$

$\Leftrightarrow_\beta\ \lambda F \exists x F(x)(\lambda z.(\lambda P.\exists y(book'(y) \wedge P(y)))(\lambda x(read^*(x)(z))))$

$\Leftrightarrow_\beta\ \lambda F \exists x F(x)(\lambda z.\exists y(book'(y) \wedge \lambda x(read^*(x)(z))(y)))$

$\Leftrightarrow_\beta\ \lambda F \exists x F(x)(\lambda z.\exists y(book'(y) \wedge read^*(y)(z)))$

$\Leftrightarrow_\beta\ \exists x(\lambda z.\exists y(book'(y) \wedge read^*(y)(z)))(x)$

$\Leftrightarrow_\beta\ \exists x \exists y(book'(y) \wedge read^*(y)(x))$

Background reading material

- Gamut: Logic, Language, and Meaning Vol II
  (Chapter 4, minus 4.3)

- Winter: Elements of Formal Semantics (Chapter 3)
  http://www.phil.uu.nl/~yoad/efs/main.html