Semantic Theory
Week 4 – Lambda Calculus

Noortje Venhuizen

University of Groningen/Universität des Saarlandes

Summer 2015

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$_{\langle\langle e,t \rangle,t\rangle}$”

!

Lambda abstraction

λ-abstraction is an operation that takes an expression and "opens" specific argument positions.

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

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

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

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

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

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

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

β-Reduction

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

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

This operation is called **β-reduction**

- $\lambda v(\alpha)(\beta) \Leftrightarrow [\beta/v]\alpha$
- $[\beta/v]\alpha$ 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 $\lambda v(\alpha)(\beta)$ and $[\beta/v]\alpha$ always equivalent?

- $\lambda x(\text{drive'}(x) \wedge \text{drink'}(x))(j^*) \Leftrightarrow \text{drive'}(j^*) \wedge \text{drink'}(j^*)$
- $\lambda x(\text{drive'}(x) \wedge \text{drink'}(x))(y) \Leftrightarrow \text{drive'}(y) \wedge \text{drink'}(y)$
- $\lambda x(\forall y\ \text{know'}(x)(y))(j^*) \Leftrightarrow \forall y\ \text{know}(j^*)(y)$
- NOT: $\lambda x(\forall y\ \text{know'}(x)(y))(y) \Leftrightarrow \forall y\ \text{know}(y)(y)$

Let v, v' be variables of the same type, α 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: $\lambda v(\alpha)(\beta) \Leftrightarrow [\beta/v]\alpha$

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

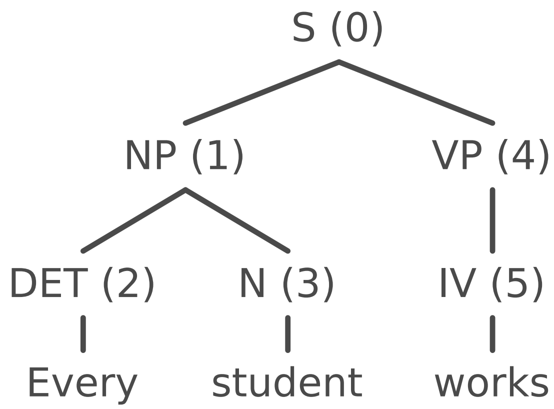
- α-conversion: $\lambda v\alpha \Leftrightarrow \lambda w[w/v]\alpha$

  (if w is free for v in α)

- η-conversion: $\lambda v(\alpha(v)) \Leftrightarrow \alpha$

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) student’ : $\langle e, t\rangle$

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

(4)/(5) work’ : $\langle e, t\rangle$

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

Type inferencing: revisited

5. Anakin$_e$ <u>believes</u>$_{\langle\langle e, t\rangle,\langle e, t\rangle\rangle}$ he will be a Jedi$_{\langle e, t\rangle}$.

   $(\lambda P_{\langle e, t\rangle}\lambda x_e(believes(P(x))(x))(J))(a^*) \Rightarrow^{\beta} \lambda x(believes(J(x))(x))(a^*) \Rightarrow^{\beta} believes(J(a^*))(a^*)$

7. Obi-Wan$_e$ <u>expects</u>$_{\langle\langle e,\langle e, t\rangle\rangle\langle e,\langle e, t\rangle\rangle\rangle}$ to pass$_{\langle e,\langle e, t\rangle\rangle}$.

   $(\lambda Q_{\langle e,\langle e, t\rangle\rangle}\lambda x_e\lambda y_e(expects(Q(y)(x))(x))(P))(o^*) \Rightarrow^{\beta^*} \lambda y.expects(P(y)(o^*))(o^*)$

8. Yoda$_e$ <u>encouraged</u>$_{\langle e,\langle\langle e, t\rangle,\langle e, t\rangle\rangle}$ Obi-Wan$_e$ to take$_{\langle e,\langle e, t\rangle\rangle}$ the exam$_e$.

   $((\lambda x_e\lambda P_{\langle e, t\rangle}\lambda y_e(encourage(x)(P(x))(y))(o^*))(\lambda x_e\lambda y_e(T(x)(y))(e^*)))(y^*)$
   $\Rightarrow^{(\beta,\alpha)^*} encourage(o^*)(T(e^*)(o^*))(y^*)$

Background reading material

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