Lambda Calculus

Week 4

Slides and materials based on the courses by
Noortje Venhuizen and Mareike Hartmann

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, 1993)

$$[\![A]\!] = \textbf{fa}([\![B]\!], [\![C]\!])$$

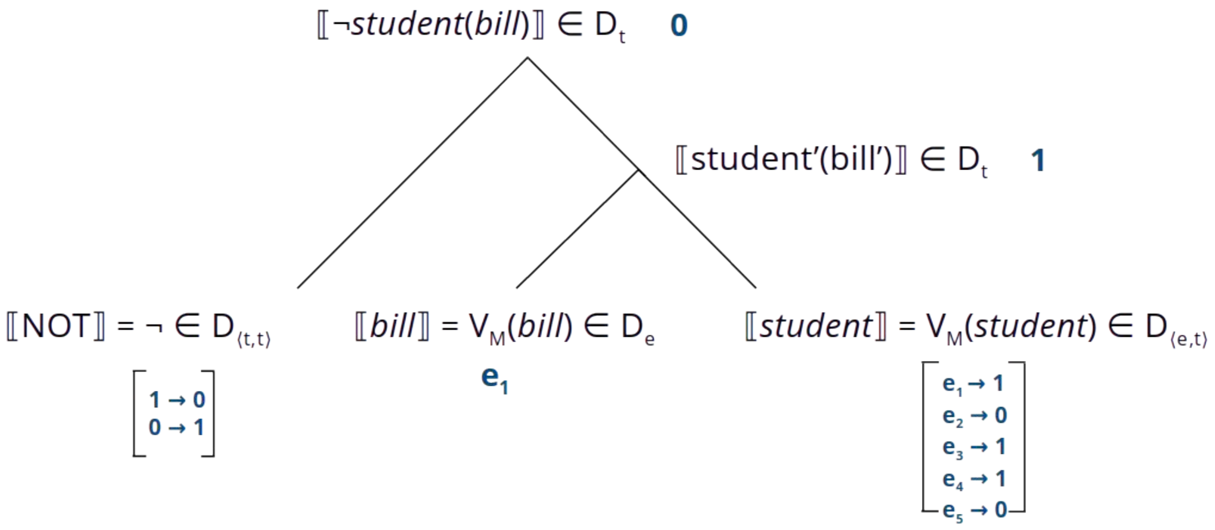
$$[\![B]\!] \qquad [\![C]\!] = \textbf{fa}([\![D]\!], [\![E]\!])$$

$$[\![D]\!] \qquad [\![E]\!]$$

- Compositional semantic construction:
  1. Define meaning representations for sub-expressions
  2. Combine them in a principled manner to obtain a meaning representation for a complex expression.

Compositionality: first try

"Bill is not a student" $\mapsto$ [NOT [[bill]$_{NP}$ [student]$_{VP}$]$_S$]$_S$



Functions and arguments

"Bill is not a student" $\mapsto [\text{NOT } [[\text{bill}]_{NP}\ [\text{student}]_{VP}]_S\ ]_S$

$[\![\ \lambda S.\neg S\ (student(bill))]\!] = [\![\neg student(bill)]\!] \in D_t$ $\quad \mathbf{0}$

$[\![\lambda x.student(x)\ (bill)]\!] = [\![student(bill)]\!] \in D_t$ $\quad \mathbf{1}$

$[\![NOT]\!] = [\![\lambda S.\neg S]\!] \in D_{\langle t,t\rangle}$

$$\begin{bmatrix} 1 \to 0 \\ 0 \to 1 \end{bmatrix}$$

$[\![bill]\!] = V_M(bill) \in D_e$

$\mathbf{e_1}$

$[\![student]\!] = [\![\lambda x.student(x)]\!] \in D_{\langle e,t\rangle}$

$$\begin{bmatrix} e_1 \to 1 \\ e_2 \to 0 \\ e_3 \to 1 \\ e_4 \to 1 \\ e_5 \to 0 \end{bmatrix}$$

I'm looking for an entity ...

... it goes here

Lambda expressions

- **Lambda expressions** are functions that consist of a set of lambda variables and a body
  - The **body** of a lambda expression is an *open** WFF:

    $[\text{Mary}_e\ [\text{sings and dances}]_{\langle e,\, t\rangle}] \mapsto [\![\lambda x(\mathit{sing}(x) \wedge \mathit{dance}(x))(\mathit{mary})]\!] \in D_t$

- Lambda expressions can themselves serve as arguments for functions (including other lambda expressions)

  $[[\text{Not smoking}]_{\langle e,\, t\rangle}\ [\text{is healthy}]_{\langle\langle e,\, t\rangle,\, t\rangle}] \mapsto [\![\mathit{healthy}(\lambda y.\neg(\mathit{smoke}(y)))]\!] \in D_t$

λ-abstraction

- Formal definition: if $\alpha \in WE_{\sigma}$ and $x \in VAR_{\pi}$, then $\lambda x(\alpha)$ is in $WE_{\langle \pi, \sigma \rangle}$

- **λ-abstraction**: the operation that transforms expressions of any type σ into a function π→σ (i.e. of the type ⟨π, σ⟩), where π is the type of the *λ-variable*
  - 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” $\lambda x.\varphi$ indicating that the λ-operator takes wide scope over φ

Interpretation of λ-expressions

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

- If the λ-expression is applied to an argument, we can simplify the interpretation:
  - $[\![\lambda v\alpha]\!]^{M,g}([\![x]\!]^{M,g}) = [\![\alpha]\!]^{M,g[v/[\![x]\!]^{M,g}]}$

- Example: *"Bill is a student"*

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

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

Interpretation of λ-expressions

- For $\varphi \in WE_t$, $x \in VAR_\sigma$:

  $V_M(\lambda x.\varphi) = \{d \in D_\sigma \mid [\![\varphi]\!]^{M,g[x/d]}\}$

- For example:
  - $V_M(\lambda x.student(x) \wedge happy(x)) = V_M(student) \cap V_M(happy)$
  - $V_M(\lambda x.blue^*(x) \vee green^*(x)) = V_M(blue^*) \cup V_M(green^*)$
  - $V_M(\lambda x.see(john)(x)) = \{d \in D_e \mid (d, john) \in V_M(see)\}$
  - $V_M(\lambda x.\forall y.eat(y)(x)) = \{d \in D_e \mid [\![\forall y.eat(y)(x)]\!]^{M,g[x/d]}\}$
  - $V_M(\lambda x_e.like(john)(mary))$ = ???

β-reduction: function application in λ-calculus

- $[\![\lambda v(\alpha)(\beta)]\!]^{M,g} = [\![\alpha]\!]^{M,g[v/[\![\beta]\!]^{M,g}]}$
  - all (free) occurrences of the λ-variable (v) in α get the interpretation of β as their value

- This operation is called **β-reduction**
  - $\lambda v(\alpha)(\beta) \Leftrightarrow \alpha[v/\beta]$
  - where: $\alpha[v/\beta]$ is the result of replacing all free occurrences of v in α with β
  - Warning: this equivalence is not unconditionally valid …

Variable capturing

- Are $\lambda v(\alpha)(\beta)$ and $\alpha[\beta/v]$ always equivalent?
  - $\lambda x(\mathit{sing}(x) \wedge \mathit{dance}(x))(\mathit{john}) \Leftrightarrow \mathit{sing}(\mathit{john}) \wedge \mathit{dance}(\mathit{john})$
  - $\lambda x(\mathit{sing}(x) \wedge \mathit{dance}(x))(y) \Leftrightarrow \mathit{sing}(y) \wedge \mathit{dance}(y)$ (where $y \in \mathit{VAR}_e$)
  - $\lambda x(\forall y.\mathit{know}(x)(y))(\mathit{john}) \Leftrightarrow \forall y.\mathit{know}(\mathit{john})(y)$
  - $\lambda x(\forall y.\mathit{know}(x)(y))(y) \not\Leftrightarrow \forall y.\mathit{know}(y)(y)$

    Problem: y is not "free for x" in $\forall y.\mathit{know}(x)(y)$

- Let x, y be variables of the same type, and let α be a WE of any type
  - **y is free for x** in α iff no free occurrence of x in α is in the scope of a quantifier or a λ-operator that binds y

Equivalence transformations in λ-calculus

- **β-conversion**: $\lambda v(\alpha)(\beta) \Leftrightarrow \alpha[v/\beta]$ (α with all instances of v replaced by β)
  - assuming all free variables in β are free for v in α

- **α-conversion**: $\lambda v.\alpha \Leftrightarrow \lambda w.\alpha[v/w]$ (α with all instances of v replaced by w)
  - assuming w is free for v in α

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

Quantifiers as λ-expressions

- *"a student works"* $\mapsto \exists x(\textit{student}(x) \land \textit{work}(x)) \quad :: t$
  - *"a student"* $\mapsto \lambda P \exists x(\text{student}'(x) \land P(x)) \quad :: \langle\langle e, t\rangle, t\rangle$
  - *"a", "some"* $\mapsto \lambda Q \lambda P \exists x(Q(x) \land P(x)) \quad :: \langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$

- *"every student"* $\mapsto \lambda P \forall x(\text{student}'(x) \rightarrow P(x)) \quad :: \langle\langle e, t\rangle, t\rangle$
  - *"every"* $\mapsto \lambda Q \lambda P \forall x(Q(x) \rightarrow P(x)) \quad :: \langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$

- *"no student"* $\mapsto \lambda P \neg \exists x(\text{student}(x) \land P(x)) \quad :: \langle\langle e, t\rangle, t\rangle$
  - *"no"* $\mapsto \lambda Q \lambda P \neg \exists x(Q(x) \land P(x)) \quad :: \langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$

- *"someone"* $\mapsto \lambda F \exists x(\textit{person}(x) \land F(x)) \quad :: \langle\langle e, t\rangle, t\rangle$
- *"something"* $\mapsto \lambda F \exists x.F(x) \quad :: \langle\langle e, t\rangle, t\rangle$

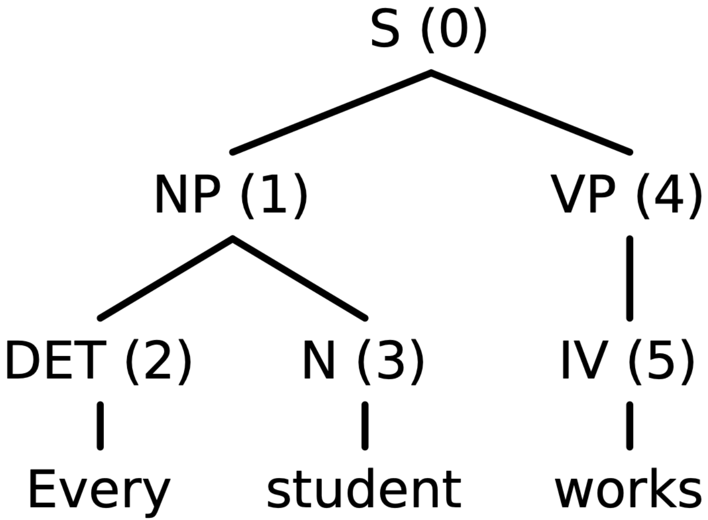
Interpretation of expressions of type $\langle\langle e, t\rangle, t\rangle$

- *something* $\in CON_{\langle\langle e, t\rangle, t\rangle}$, so V_M(*something*) $\in D_{\langle\langle e, t\rangle, t\rangle}$

- $D_{\langle\langle e, t\rangle, t\rangle}$ is the set of functions from $D_{\langle e, t\rangle}$ to D_t
  - i.e. the set of functions from $\mathbb{p}(U_M)$ (the **powerset** of U_M) to {0, 1}, which in turn is equivalent to $\mathbb{p}(\mathbb{p}(U_M))$

- From V_M(*something*) $\in \mathbb{p}(\mathbb{p}(U_M))$ it follows that V_M(*something*) $\subseteq \mathbb{p}(U_M)$
  - More specifically: V_M(*something*) = $\{S \subseteq U_M \mid S \neq \varnothing\}$, if U_M is a domain of individuals

Compositional construction



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

(3) $\lambda y.\mathit{student}(y) \Leftrightarrow_{\eta} \mathit{student} :: \langle e, t\rangle$

(1) $\lambda P \lambda Q \forall x(P(x) \rightarrow Q(x))(\mathit{student})$

$\Leftrightarrow_{\beta} \lambda Q \forall x(\mathit{student}(x) \rightarrow Q(x)) :: \langle\langle e, t\rangle, t\rangle$

(4)/(5) $\lambda z.\mathit{work}(z) \Leftrightarrow_{\eta} \mathit{work} :: \langle e, t\rangle$

(0) $\lambda Q \forall x(\mathit{student}(x) \rightarrow Q(x))(\mathit{work}) \Leftrightarrow_{\beta} \forall x(\mathit{student}(x) \rightarrow \mathit{work}(x)) :: t$

Compositional construction

"not smoking is healthy" $\mapsto$ [[not smoking] [is healthy]]

$[\![\text{healthy(not smoking)}]\!] \in D_t$

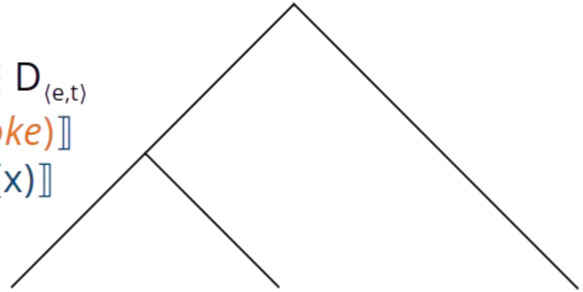
$[\![\lambda P.\mathit{healthy}(P)(\lambda x(\neg \mathit{smoke}(x)))]\!]$

$\Leftrightarrow^{\beta} [\![\mathit{healthy}(\lambda x(\neg \mathit{smoke}(x)))]\!]$

$[\![\text{not smoking}]\!] \in D_{\langle e,t\rangle}$

$[\![\lambda Q\lambda x.\neg Q(x)(\mathit{smoke})]\!]$

$\Leftrightarrow^{\beta} [\![\lambda x.\neg \mathit{smoke}(x)]\!]$



$[\![\text{not}]\!] \in D_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$

$[\![\lambda P\lambda x.\neg P(x)]\!] \Leftrightarrow^{\alpha} [\![\lambda Q\lambda x.\neg Q(x)]\!]$

$[\![\text{smoking}]\!] \in D_{\langle e,t\rangle}$

$[\![\lambda x.\mathit{smoke}(x)]\!] \Leftrightarrow^{\eta} [\![\mathit{smoke}]\!]$

$[\![\text{healthy}]\!] \in D_{\langle\langle e,t\rangle,t\rangle}$

$[\![\mathit{healthy}]\!] \Leftrightarrow [\![\lambda P.\mathit{healthy}(P)]\!]$

Type Clash

- Problem: in natural language, quantified expressions occur with transitive verbs in both subject and object position
  - Example: *"someone reads a book"*

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

- Solution: reverse functor-argument relation
  - Logical form: *someone*(*read*(*a book*))
  - Use **type raising** to adjust the type of the transitive verb: $read_{\langle\langle\langle e,\, t\rangle,\, t\rangle,\, \langle e,\, t\rangle\rangle}$

Type Raising

- What if we just change the type of the transitive verb?
  - “read” ➔ $read \in CON_{\langle\langle\langle e,\, t\rangle,\, t\rangle,\, \langle e,\, t\rangle\rangle}$

$$[\![\text{someone reads a book}]\!] =$$

$$[\![\lambda F \exists x(person(x) \land F(x))(read(\lambda P \exists y(book'(y) \land P(y))))]\!]$$

$$\Leftrightarrow_{\beta} [\![\exists x(person(x) \land read(\lambda P \exists y(book'(y) \land P(y)))(x))]\!]$$

- Stuck! We need a more explicit λ-term:
  - $read \leftarrow \lambda Q \lambda z.Q(\lambda x(read^*(x)(z))) \in WE_{\langle\langle\langle e,\, t\rangle,\, t\rangle,\, \langle e,\, t\rangle\rangle}$ *(type raising)*
  - where $read^* \in WE_{\langle e,\, \langle e,\, t\rangle\rangle}$ is the “underlying” first-order relation

Type Raising (Cont.)

- Specifically, given $f: \pi\rightarrow\sigma$, $g: \sigma\rightarrow\delta$, we can *type raise* f to get $\uparrow$(f): $(\sigma\rightarrow\delta)\rightarrow(\pi\rightarrow\delta)$, so that f takes g as an argument (this is basically just function composition):
  - $\uparrow(f) = \lambda h_{(\sigma\rightarrow\delta)}\lambda x_{\pi}.h(f(x))$
  - $\uparrow(f)(g) = \lambda x_{\pi}.g(f(x)): \pi\rightarrow\delta$

- Given $c: \sigma$, $g: \sigma\rightarrow\delta$, we can consider c as a function $c: \varnothing\rightarrow\sigma$
  - $\uparrow(c): (\sigma\rightarrow\delta)\rightarrow(\varnothing\rightarrow\delta)$
  - $\uparrow(c): \lambda h_{(\sigma\rightarrow\delta)}.h(c)$
  - $\uparrow(c)(g) = g(c): \varnothing\rightarrow\delta \Leftrightarrow \delta$

Type Raising

someone reads a book $\mapsto$ someone(reads(a book))

$$\lambda F \exists x(\mathit{person}(x) \wedge F(x))(\lambda Q \lambda z(Q(\lambda w(\mathit{read}^*(w)(z))))(\lambda P \exists y(\mathit{book}(y) \wedge P(y))))$$

$$\Leftrightarrow_\beta \lambda F \exists x(\mathit{person}(x) \wedge F(x))(\lambda z(\lambda P \exists y(\mathit{book}(y) \wedge P(y))(\lambda w(\mathit{read}^*(w)(z)))))$$

$$\Leftrightarrow_\beta \lambda F \exists x(\mathit{person}(x) \wedge F(x))(\lambda z(\exists y(\mathit{book}(y) \wedge \lambda w(\mathit{read}^*(w)(z))(y))))$$

$$\Leftrightarrow_\beta \lambda F \exists x(\mathit{person}(x) \wedge F(x))(\lambda z(\exists y(\mathit{book}(y) \wedge \mathit{read}^*(y)(z))))$$

$$\Leftrightarrow_\beta \exists x(\mathit{person}(x) \wedge \lambda z(\exists y(\mathit{book}(y) \wedge \mathit{read}^*(y)(z)))(x))$$

$$\Leftrightarrow_\beta \exists x(\mathit{person}(x) \wedge \exists y(\mathit{book}(y) \wedge \mathit{read}^*(y)(x)))$$