Semantic Theory

Semantic Construction 3

Manfred Pinkal/ Stefan Thater
Saarland University

Compositionality and Inference

- Higher-order logic provides a framework for a unified treatment of all types of NPs in semantic construction.
- However, by representing quantified NPs as basic second-order predicates, without FOL quantifiers and connectives, we lose one of the main motivations for the logical modelling of meaning information: The availability of deduction systems and theorem provers which support correct and efficient reasoning.
- Challenge 3:
  A semantic representation that preserves compositionality of quantified expressions and at the same time provides access to the deductive power of FOL.
- The solution: Extending higher-order logic to typed λ-calculus.

Another coverage problem

John drinks and drives
Drinking is unwise
Drinking and driving is unwise (?)
Swimming is healthy
Not smoking is healthy (?)

λ-Expressions

- λx[drive(x) ∧ drink(x)]
- … a term of type (e,t)
- … denotes the property of being “an x such that x drives and drinks”.
- λ-abstraction is an operation that takes an expression and “opens” specific argument positions. The result of abstraction over individual variable x in the formula drive(x) ∧ drink(x) results in the complex predicate λx[drive(x) ∧ drink(x)].

Typed λ-Calculus, Syntax

- Syntax, like basic type theory, plus:
  - If $\alpha \in WE_\tau$ and $v \in VAR_\sigma$, then $\lambda v \alpha \in WE_{(\sigma,\tau)}$.
- Notational convention:
  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" λx. … indicating that the λ-operator takes widest possible scope.

λ-Expressions: Example

John drives and drinks.

$$\frac{\dfrac{\dfrac{\text{drive: (e,t)}\quad \text{x:e}}{\text{drive(x): t}}\qquad \dfrac{\text{drink: (e,t)}\quad \text{x:e}}{\text{drink(x): t}}}{\text{drive(x)}\wedge\text{drink(x): t}}}{\dfrac{\text{john : e}\qquad \lambda x[\text{drive(x)}\wedge\text{drink(x)}]\text{: (e,t)}}{\lambda x[\text{drive(x)}\wedge\text{drink(x)}](\text{john}) : t}}$$

λ-Expressions: More examples

John drinks and drives

λx[drive(x)∧drink(x)])(j*)

Drinking and driving is unwise

¬*wise*(λx.drink(x) ∧ drive(x))

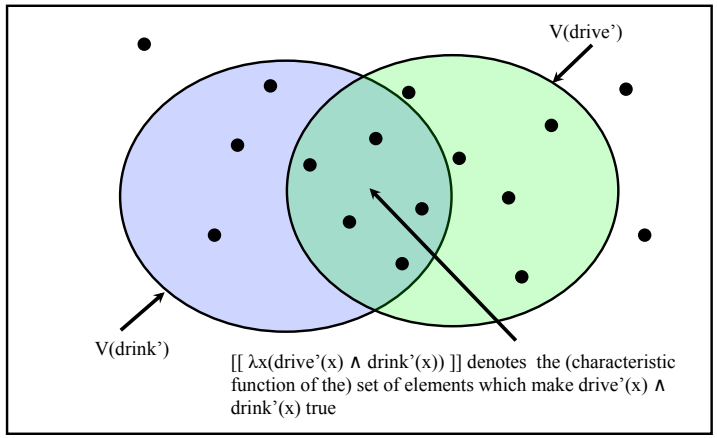
Not smoking is healthy

healthy(λx.¬smoke(x))

λ-Abstraction: Semantics

- $[\![\lambda v \alpha]\!]^{M,g}$ is that function $f : D_\sigma \rightarrow D_\tau$ such that for all $a \in D_\sigma$, $f(a) = [\![\alpha]\!]^{M,g[v/a]}$ (for $\alpha \in WE_\tau$, $v \in VAR_\sigma$)
- By the modified variable assignment, the argument value is passed down through the interpretation of the λ-expression and becomes the value of all occurrences of variables which are bound by the λ-operator.
- If the λ-expression is applied to an appropriate argument, we can simplify the interpretation:
  $[\![\lambda v \alpha(\beta)]\!]^{M,g} = [\![\alpha]\!]^{M,g[v\,/\,[\![\beta]\!]^{M,g}]}$
  The value of λvα(β) is the value of α, where the variable assignment function is set to the value of β for variable v.

λx(drive'(x) ∧ drink'(x))



λ-Conversion

- The interpretation of λvα(β) guarantees that all occurrences of the λ-variable in α get the interpretation of argument β as value.
- Can't we get the same result by directly substituting the occurrences of the λ-variable in α by β?
  Yes, we can:

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

- This means that the following equivalence relation holds:
  - λvα(β) ⇔ [β/v]α
- [β/v]α is the result of replacing all free occurrences of v in β with α.
- Attention: The equivalence relation is not unconditionally valid, but requires satisfaction of an additional constraint (see below).

Example

John drives and drinks.

$$\frac{\underline{\text{drive: (e,t)}} \quad \underline{\text{x:e}} \qquad \underline{\text{drink: (e,t)}} \quad \underline{\text{x:e}}}{\underline{\text{drive(x): t}} \qquad \underline{\text{drink(x): t}}}$$

$$\frac{\underline{\text{drive(x)}\wedge\text{drink(x): t}}}{\underline{\text{john : e}} \qquad \underline{\lambda x[\text{drive(x)}\wedge\text{drink(x)}]\text{: (e,t)}}}$$

$$\lambda x[\text{drive(x)}\wedge\text{drink(x)}](\text{john}) : t$$

⇔ drive(john) ∧ drink(john) : t

λ-Conversion

- Are λvα(β) and [β/v]α always equivalent?
  - λx[drive'(x) ∧ drink'(x)](j*) ⇒ drive'(j*) ∧ drink'(j*)
  - λx[drive'(x) ∧ drink'(x)](y) ⇒ drive'(y) ∧ drink'(y)
  - λx[∀y know'(x)(y)](j*) ⇒ ∀y know(j*)(y)
  - λx[∀y know'(x)(y)](y) \⇒ ∀y know(y)(y)

- What is going wrong here? Variable y which was unbound before is substituted for x in a position where it becomes bound ("is captured") by the universal quantifier: y "is not free for x" in the body of the λ-expression.
- 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 in the λ-calculus

- β-conversion:
  λvα(β) ⇔ [β/v]α if all free variables in β are free for v in α.
- α-conversion:
  λvα ⇔ λv'[v'/v]α if v' is free for v in α.
- η-conversion:
  λv(α(v)) ⇔ α
- The rule which we will use most in semantics construction is β-conversion in the left-to-right direction (β-reduction), which allows us to simplify representations.

Back to noun phrases

- Interpretation of "John:" applies to property P if John has property P:
  - λP(P(j*))
- Interpretation of "every student:" applies to property P if every student has property P:
  - λP(∀x(student'(x) → P(x)))
- Interpretation of "a student:" applies to property P if a student has property P:
  - λP(∃x(student'(x) ∧ P(x)))

Determiners

- a, some⇒ λFλG ∃x(F(x) ∧ G(x))
- every⇒ λFλG ∀x(F(x) → G(x))
- no⇒ λFλG ¬∃x(F(x) ∧ G(x))

Example: Every student works