Semantics & Discourse

— Mahtematical Preliminaries —

Philippe de Groote

Outline

1. First-order logic
2. λ-calculus
3. Typed λ-calculus and higher-order logic

Outline

1. First-order logic
   - Model-theoretic semantics
   - First-order language
   - Model and interpretation
   - Propositional logic
   - Quantification
   - Interpretation

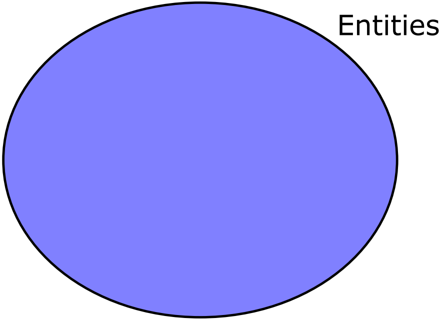
Model-theoretic semantics

Model-theoretic semantics

John is the brother of Jean

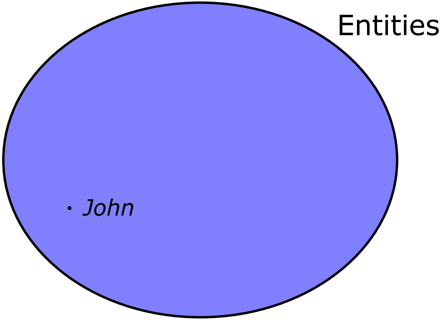
Model-theoretic semantics

John is the brother of Jean



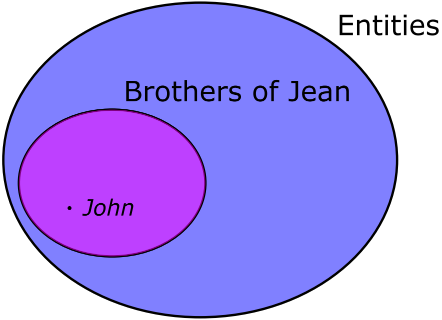
Model-theoretic semantics

John is the brother of Jean



Model-theoretic semantics

John is the brother of Jean



First-order language

Definition

A first-order langage consists in two sets of symbols:

- A set $\mathscr{F}$, together with an arity function $\mathrm{ar}_{\mathscr{F}} \in \mathbb{N}^{\mathscr{F}}$, whose elements are called *function symbols*.
- A set $\mathscr{R}$, together with an arity function $\mathrm{ar}_{\mathscr{R}} \in \mathbb{N}^{\mathscr{R}}$, whose elements are called *relation symbols*.

First-order language

Definition

A first-order langage consists in two sets of symbols:

- A set $\mathscr{F}$, together with an arity function $\mathrm{ar}_{\mathscr{F}} \in \mathbb{N}^{\mathscr{F}}$, whose elements are called *function symbols*.
- A set $\mathscr{R}$, together with an arity function $\mathrm{ar}_{\mathscr{R}} \in \mathbb{N}^{\mathscr{R}}$, whose elements are called *relation symbols*.

Example

- $\mathscr{F} = \{\mathbf{e}, \mathbf{j}, \mathbf{r}, \mathbf{father}\}$;
- $\mathrm{ar}_{\mathscr{F}}(\mathbf{e}) = 0, \mathrm{ar}_{\mathscr{F}}(\mathbf{j}) = 0, \mathrm{ar}_{\mathscr{F}}(\mathbf{r}) = 0, \mathrm{ar}_{\mathscr{F}}(\mathbf{father}) = 1$;
- $\mathscr{R} = \{\mathbf{Is}, \mathbf{Husband}\}$;
- $\mathrm{ar}_{\mathscr{R}}(\mathbf{Is}) = 2, \mathrm{ar}_{\mathscr{R}}(\mathbf{Husband}) = 2$.

First-order language

Terms

Let $\mathscr{X}$ be a countably infinite set of symbols whose elements are called *variables*. The set of terms is inductively defined as follows:

- every $x \in \mathscr{X}$ is a term;
- every $a \in \mathscr{F}$ such that $\mathrm{ar}_{\mathscr{F}}(a) = 0$ is a term,
- if $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$ and $n > 0$, and if $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

First-order language

Terms

Let $\mathscr{X}$ be a countably infinite set of symbols whose elements are called *variables*. The set of terms is inductively defined as follows:

- every $x \in \mathscr{X}$ is a term;
- every $a \in \mathscr{F}$ such that $\mathrm{ar}_{\mathscr{F}}(a) = 0$ is a term,
- if $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$ and $n > 0$, and if $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

Proposition

- if $R \in \mathscr{R}$ with $\mathrm{ar}_{\mathscr{R}}(R) = n$, and if $t_1, \ldots, t_n$ are terms, then $R(t_1, \ldots, t_n)$ is an atomic proposition.

First-order language

Terms

Let $\mathscr{X}$ be a countably infinite set of symbols whose elements are called *variables*. The set of terms is inductively defined as follows:

- every $x \in \mathscr{X}$ is a term;
- every $a \in \mathscr{F}$ such that $\mathrm{ar}_{\mathscr{F}}(a) = 0$ is a term,
- if $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$ and $n > 0$, and if $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

Proposition

- if $R \in \mathscr{R}$ with $\mathrm{ar}_{\mathscr{R}}(R) = n$, and if $t_1, \ldots, t_n$ are terms, then $R(t_1, \ldots, t_n)$ is an atomic proposition.

Example

- Terms: $\mathbf{e}; \mathbf{father}(\mathbf{j}); \mathbf{father}(\mathbf{father}(\mathbf{r})); \mathbf{father}(\mathbf{x})$.
- Proposition: $\mathbf{Is}(\mathbf{e}, \mathbf{father}(\mathbf{j})); \mathbf{Husband}(\mathbf{e}, \mathbf{r})$.

Model and interpretation

Model

Given a first-order langage, a model consists of a set D and an interpretation function $\mathcal{I}$ defined on $\mathscr{F} \cup \mathscr{R}$ such that:

- for every $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$, $\mathcal{I}(f) \in D^{D^n}$;
- for every $R \in \mathscr{R}$ with $\mathrm{ar}_{\mathscr{R}}(R) = n$, $\mathcal{I}(R) \in 2^{D^n}$.

Model and interpretation

Model

Given a first-order langage, a model consists of a set D and an interpretation function $\mathcal{I}$ defined on $\mathscr{F} \cup \mathscr{R}$ such that:

- for every $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$, $\mathcal{I}(f) \in D^{D^n}$;
- for every $R \in \mathscr{R}$ with $\mathrm{ar}_{\mathscr{R}}(R) = n$, $\mathcal{I}(R) \in 2^{D^n}$.

Example

- $D = \mathbb{N}_0$
- $\mathcal{I}(\mathbf{e}) = 6$
- $\mathcal{I}(\mathbf{j}) = 3$
- $\mathcal{I}(\mathbf{r}) = 7$
- $\mathcal{I}(\mathbf{father}) = f \in D^D$ such that $f(n) = 2n$
- $\mathcal{I}(\mathbf{Is}) = \{(a, b) \in D^2 : a = b\}$
- $\mathcal{I}(\mathbf{Husband}) = \{(a, b) \in D^2 : a = 2n \text{ and } b = a+1 \text{ for some } n \in D\}$

Model and interpretation

Interpretation of the ground terms

Given a first-order langage, and a model, the interpretation of the ground terms is inductively defined as follows:

- $[\![a]\!] = \mathcal{I}(a)$, for $a \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(a) = 0$;
- $[\![f(t_1, \ldots, t_n)]\!] = \mathcal{I}(f)([\![t_1]\!], \ldots, [\![t_n]\!])$, for $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$ and $n > 0$.

Model and interpretation

Interpretation of the ground terms

Given a first-order langage, and a model, the interpretation of the ground terms is inductively defined as follows:

- $[\![a]\!] = \mathcal{I}(a)$, for $a \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(a) = 0$;
- $[\![f(t_1, \ldots, t_n)]\!] = \mathcal{I}(f)([\![t_1]\!], \ldots, [\![t_n]\!])$, for $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$ and $n > 0$.

Example

$$\begin{aligned}
[\![\mathbf{father}(\mathbf{father}(\mathbf{r}))]\!] &= \mathcal{I}(\mathbf{father})([\![\mathbf{father}(\mathbf{r})]\!]) \\
&= 2 \cdot ([\![\mathbf{father}(\mathbf{r})]\!]) \\
&= 2 \cdot (\mathcal{I}(\mathbf{father})([\![\mathbf{r}]\!])) \\
&= 2 \cdot (2 \cdot [\![\mathbf{r}]\!]) \\
&= 2 \cdot (2 \cdot 7) \\
&= 28
\end{aligned}$$

Model and interpretation

Interpretation of the closed atomic propositions

Given a first-order langage, and a model, the interpretation of the closed atomic propositions is defined as follows:

- $[\![R(t_1,\ldots,t_n)]\!] = \mathcal{I}(R)([\![t_1]\!],\ldots,[\![t_n]\!])$, for $R \in \mathscr{R}$ with $\mathrm{ar}_{\mathscr{R}}(R) = n$.

Model and interpretation

Interpretation of the closed atomic propositions

Given a first-order langage, and a model, the interpretation of the closed atomic propositions is defined as follows:

- $[\![R(t_1, \ldots, t_n)]\!] = \mathcal{I}(R)([\![t_1]\!], \ldots, [\![t_n]\!])$, for $R \in \mathscr{R}$ with $\mathrm{ar}_{\mathscr{R}}(R) = n$.

Example

$$\begin{aligned}
[\![\mathbf{Is}(\mathbf{e}, \mathbf{father}(\mathbf{j}))]\!] &= \mathcal{I}(\mathbf{Is})([\![\mathbf{e}]\!], [\![\mathbf{father}(\mathbf{j})]\!]) \\
&= \mathcal{I}(\mathbf{Is})([\![\mathbf{e}]\!], \mathcal{I}(\mathbf{father})([\![\mathbf{j}]\!])) \\
&= \mathcal{I}(\mathbf{Is})(6, 2 \cdot 3) \\
&= \mathcal{I}(\mathbf{Is})(6, 6) \\
&= 1
\end{aligned}$$

Model and interpretation

Valuation

Given a first-order langage, and a model, a valuation is a a function $\xi \in D^{\mathscr{X}}$.

Model and interpretation

Valuation

Given a first-order langage, and a model, a valuation is a a function $\xi \in D^{\mathscr{X}}$.

Interpretation of the terms

Given a first-order langage, and a model, the interpretation of the terms is inductively defined as follows:

- $[\![x]\!]_\xi = \xi(x)$, for $x \in \mathscr{X}$;
- $[\![a]\!]_\xi = \mathcal{I}(a)$, for $a \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(a) = 0$;
- $[\![f(t_1, \ldots, t_n)]\!]_\xi = \mathcal{I}(f)([\![t_1]\!]_\xi, \ldots, [\![t_n]\!]_\xi)$, for $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$ and $n > 0$.

Model and interpretation

Interpretation of the atomic propositions

Given a first-order langage, and a model, the interpretation of the closed atomic propositions is defined as follows:

- $[\![R(t_1, \ldots, t_n)]\!]_\xi = \mathcal{I}(R)([\![t_1]\!]_\xi, \ldots, [\![t_n]\!]_\xi)$, for $R \in \mathscr{R}$ with $\mathrm{ar}_{\mathscr{R}}(R) = n$.

Propositional logic

propositions

Given a first-order language, the set of proposition is inductively defined as follows:

- every atomic proposition is a proposition;
- if α is a proposition then $\neg\alpha$ is a proposition;
- if α and β are propositions then $(\alpha \wedge \beta)$ is a proposition;
- if α and β are propositions then $(\alpha \vee \beta)$ is a proposition;
- if α and β are propositions then $(\alpha \to \beta)$ is a proposition.

Propositional logic

propositions

Given a first-order language, the set of proposition is inductively defined as follows:

- every atomic proposition is a proposition;
- if α is a proposition then $\neg\alpha$ is a proposition;
- if α and β are propositions then $(\alpha \wedge \beta)$ is a proposition;
- if α and β are propositions then $(\alpha \vee \beta)$ is a proposition;
- if α and β are propositions then $(\alpha \rightarrow \beta)$ is a proposition.

Example

$$\mathbf{Husband}(\mathbf{e}, \mathbf{r}) \wedge \mathbf{Is}(\mathbf{e}, \mathbf{father}(\mathbf{j}))$$

Propositional logic

Negation

$\neg\alpha$

- *not* α.
- $[\![\neg\alpha]\!]_\xi = 1$ iff $[\![\alpha]\!]_\xi = 0$.

α	$\neg\alpha$
0	1
1	0

Propositional logic

Conjunction

$\alpha \wedge \beta$

- α *and* β.
- $[\![\alpha \wedge \beta]\!]_\xi = 1$ iff $[\![\alpha]\!]_\xi = 1$ and $[\![\beta]\!]_\xi = 1$.

α	β	$\alpha \wedge \beta$
0	0	0
0	1	0
1	0	0
1	1	1

Propositional logic

Disjunction

$\alpha \vee \beta$

- α *or* β.
- $[\![\alpha \vee \beta]\!]_\xi = 1$ iff $[\![\alpha]\!]_\xi = 1$ or $[\![\beta]\!]_\xi = 1$ (or both).

α	β	$\alpha \vee \beta$
0	0	0
0	1	1
1	0	1
1	1	1

Propositional logic

Implication

$\alpha \rightarrow \beta$

- *If* α *then* β; α *implies* β.
- $[\![\alpha \rightarrow \beta]\!]_\xi = 1$ iff $[\![\beta]\!]_\xi = 1$ whenever $[\![\alpha]\!]_\xi = 1$.

α	β	$\alpha \rightarrow \beta$
0	0	1
0	1	1
1	0	0
1	1	1

Quantification

First-order formulas

Given a first-order language, the set of formulas is inductively defined as follows:

- every atomic proposition is a formula;
- if α is a formula then $\neg\alpha$ is a formula;
- if α and β are formulas then $(\alpha \wedge \beta)$ is a formula;
- if α and β are formulas then $(\alpha \vee \beta)$ is a formula;
- if α and β are formulas then $(\alpha \rightarrow \beta)$ is a formula;
- if α is a formulas and x a variable then $(\forall x.\,\alpha)$ is a formula;
- if α is a formulas and x a variable then $(\exists x.\,\alpha)$ is a formula.

Quantification

First-order formulas

Given a first-order language, the set of formulas is inductively defined as follows:

- every atomic proposition is a formula;
- if α is a formula then $\neg\alpha$ is a formula;
- if α and β are formulas then $(\alpha \wedge \beta)$ is a formula;
- if α and β are formulas then $(\alpha \vee \beta)$ is a formula;
- if α and β are formulas then $(\alpha \rightarrow \beta)$ is a formula;
- if α is a formulas and x a variable then $(\forall x.\, \alpha)$ is a formula;
- if α is a formulas and x a variable then $(\exists x.\, \alpha)$ is a formula.

Example

$$\forall x.\, \exists y.\, \mathbf{Is}(\mathbf{y}, \mathbf{father}(\mathbf{x}))$$

Quantification

Universal quantification

$\forall x.\, \alpha$

- *every entity* x is such that α.
- $[\![\forall x.\, \alpha]\!]_{\xi} = 1$ iff $[\![\alpha]\!]_{\xi[x:=d]} = 1$ for every $d \in D$.

Quantification

Existential quantification

$\exists x.\, \alpha$

- *There is some entity* x such that α.
- $[\![\exists x.\, \alpha]\!]_\xi = 1$ iff $[\![\alpha]\!]_{\xi[x:=d]} = 1$ for some $d \in D$.

Interpretation

Let a first-order language be given, and let ϕ, $\mathscr{M}$, and ξ be respectively a first-order formula, a model, and a valuation.

$\mathscr{M}, \xi \models \phi$

- $\mathscr{M}$ and ξ *satisfy* ϕ.
- ϕ *is valid in* $\mathscr{M}$ *according to* ξ.
- $[\![\phi]\!]_\xi = 1$.

$\mathscr{M} \models \phi$

- $\mathscr{M}$ *satisfies* ϕ.
- ϕ *is valid in* $\mathscr{M}$.
- $\mathscr{M}, \xi \models \phi$ for every possible valuation ξ.

$\models \phi$

- ϕ *is valid*.
- $\mathscr{M} \models \phi$ for every possible model $\mathscr{M}$.

Outline

2. λ-calculus
   - λ-Notatation
   - λ-Terms
   - β-Reduction

λ-Notatation

$$\text{“}2x + y\text{”}$$

λ-Notatation

$$\text{“}2x + y\text{”}$$

$f(x) = 2x + y$

λ-Notatation

$$\text{“}2x + y\text{”}$$

$f(x) = 2x + y$

$$\lambda x.\, 2x + y$$

λ-Notatation

$$\text{“}2x + y\text{”}$$

$$f(x) = 2x + y$$

$$\lambda x.\, 2x + y$$

$$f(y) = 2x + y$$

λ-Notatation

$$\text{“}2x + y\text{”}$$

$$f(x) = 2x + y$$

$$\lambda x.\, 2x + y$$

$$f(y) = 2x + y$$

$$\lambda y.\, 2x + y$$

λ-Notatation

$$\text{“}2x + y\text{”}$$

$$f(x) = 2x + y$$

$$\lambda x.\, 2x + y$$

$$f(y) = 2x + y$$

$$\lambda y.\, 2x + y$$

$$f(x, y) = 2x + y$$

λ-Notatation

$$\text{“}2x + y\text{”}$$

$$f(x) = 2x + y$$

$$\lambda x.\, 2x + y$$

$$f(y) = 2x + y$$

$$\lambda y.\, 2x + y$$

$$f(x, y) = 2x + y$$

$$\lambda xy.\, 2x + y$$

λ-Terms

Definition

Let $\mathscr{C}$ be a set of symbols whose elements are called *constants*, and let $\mathscr{X}$ be a countably infinite set of symbols, disjoint from $\mathscr{C}$, whose elements are called λ-*variables*. The set of λ-terms is inductively defined as follows:

- every $c \in \mathscr{C}$ is a λ-term;
- every $x \in \mathscr{X}$ is a λ-term;
- if t is a λ-term and x is a λ-variable then $(\lambda x.\, t)$ is a λ-term;
- if t and u are λ-terms then $(t\, u)$ is a λ-term.

λ-Terms

Abstraction

$$(\lambda x.\, t)$$

λ-Terms

Abstraction

$(\lambda x.\, t)$

- The function that maps x to t.
- t is called the *body* of the abstraction.
- The free occurences of x in t are bound in $(\lambda x.\, t)$.

λ-Terms

Abstraction

$(\lambda x.\, t)$

- The function that maps x to t.
- t is called the *body* of the abstraction.
- The free occurences of x in t are bound in $(\lambda x.\, t)$.

Curryfication

$g(x, y) = x + y$

λ-Terms

Abstraction

$(\lambda x.\, t)$

- The function that maps x to t.
- t is called the *body* of the abstraction.
- The free occurences of x in t are bound in $(\lambda x.\, t)$.

Curryfication

$$g(x, y) = x + y$$
$$f_x(y) = x + y$$
$$g'(x) = f_x$$

λ-Terms

Abstraction

$(\lambda x.\, t)$

- The function that maps x to t.
- t is called the *body* of the abstraction.
- The free occurences of x in t are bound in $(\lambda x.\, t)$.

Curryfication

$$g(x, y) = x + y$$
$$f_x(y) = x + y$$
$$g'(x) = f_x$$
$$g'(x)(y) = f_x(y) = x + y = g(x, y)$$

λ-Terms

Application

$(t\,u)$

λ-Terms

Application

$(t\,u)$

- The function t applied to the argument u.
- t is called the operator, and u the operand.

λ-Terms

Application

$(t\,u)$

- The function t applied to the argument u.
- t is called the operator, and u the operand.

Usual notations:

$f : x \mapsto x + 1$
$f(3)$

λ-Terms

Application

$(t\,u)$

- The function t applied to the argument u.
- t is called the operator, and u the operand.

Usual notations:

$$f : x \mapsto x + 1$$
$$f(3)$$

λ-calculus notations:

$$\lambda x.\,\mathbf{add}\,x\,1$$
$$(\lambda x.\,\mathbf{add}\,x\,1)\,3$$

β-Reduction

Substitution

Let t and u be λ-terms, and x be a λ-variable. $t[x := u]$ denotes the λ-term obtained by substituting u for the free occurrences of x in t. It is inductively defined as follows:

$c[x := u] = c$, for $c \in \mathscr{C}$.

$y[x := u] = y$, for $y \in \mathscr{X}$, and $y \neq x$.

$x[x := u] = u$

$(\lambda y.\, t_0)[x := u] = (\lambda y.\, t_0[x := u])$, where $y \neq x$ and y not free in u.

$(t_0\, t_1)[x := u] = (t_0[x := u]\, t_1[x := u])$

β-Reduction

Notion of β-reduction

$$(\lambda x.\, t)\, u \rightarrow_\beta t[x := u]$$

β-Reduction

Notion of β-reduction

$$(\lambda x.\, t)\, u \rightarrow_\beta t[x := u]$$

Relation of β-contraction

$$C[(\lambda x.\, t)\, u] \rightarrow_\beta C[t[x := u]]$$

β-Reduction

Notion of β-reduction

$$(\lambda x.\, t)\, u \rightarrow_\beta t[x := u]$$

Relation of β-contraction

$$C[(\lambda x.\, t)\, u] \rightarrow_\beta C[t[x := u]]$$

Relation of β-reduction

The *reflexive, transitive* closure of the relation of β-contraction.

$$t \twoheadrightarrow_\beta u$$

β-Reduction

Notion of β-reduction

$$(\lambda x.\, t)\, u \rightarrow_\beta t[x := u]$$

Relation of β-contraction

$$C[(\lambda x.\, t)\, u] \rightarrow_\beta C[t[x := u]]$$

Relation of β-reduction

The *reflexive, transitive* closure of the relation of β-contraction.

$t \twoheadrightarrow_\beta u$

Relation of β-equivalence

The *reflexive, transitive, symmetric* closure of the relation of β-contraction.

$t =_\beta u$

β-Reduction

Church-Rosser property

Let t_0, t_1, and t_2 be λ-terms such that

$$t_0 \twoheadrightarrow_\beta t_1$$
$$t_0 \twoheadrightarrow_\beta t_2$$

Then, there exists a λ-term t_3 such that

$$t_1 \twoheadrightarrow_\beta t_3$$
$$t_2 \twoheadrightarrow_\beta t_3$$

β-Reduction

Church-Rosser property

Let t_0, t_1, and t_2 be λ-terms such that

$$t_0 \twoheadrightarrow_\beta t_1$$
$$t_0 \twoheadrightarrow_\beta t_2$$

Then, there exists a λ-term t_3 such that

$$t_1 \twoheadrightarrow_\beta t_3$$
$$t_2 \twoheadrightarrow_\beta t_3$$

Corollary: unicity of the normal forms.

Outline

3. Typed λ-calculus and higher-order logic
   - Simple types
   - interpretation
   - Logical constants

Simple types

Definition

Let $\mathscr{A}$ be a set of symbols whose elements are called *atomic types* The set of simple types is inductively defined as follows:

- every $a \in \mathscr{A}$ is a simple type;
- if α and β are simple types then $(\alpha \to \beta)$ is a simple type.

Simple types

Definition

Let $\mathscr{A}$ be a set of symbols whose elements are called *atomic types* The set of simple types is inductively defined as follows:

- every $a \in \mathscr{A}$ is a simple type;
- if α and β are simple types then $(\alpha \to \beta)$ is a simple type.

The intended meaning is that $(\alpha \to \beta)$ is the type of the λ-terms that stand for functions whose domain is α, and range β.

Simple types

Definition

Let $\mathscr{A}$ be a set of symbols whose elements are called *atomic types* The set of simple types is inductively defined as follows:

- every $a \in \mathscr{A}$ is a simple type;
- if α and β are simple types then $(\alpha \rightarrow \beta)$ is a simple type.

The intended meaning is that $(\alpha \rightarrow \beta)$ is the type of the λ-terms that stand for functions whose domain is α, and range β.

Given a set of atomic type $\mathscr{A}$, we write $\mathscr{T}(\mathscr{A})$ for the set of simple types built upon $\mathscr{A}$.

Simple types

Signature

A higher-order signature is a triple $\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$, where:

- $\mathscr{A}$ is a set of atomic types;
- $\mathscr{C}$ is a set of constants;
- $\tau \in \mathscr{T}(\mathscr{A})^{\mathscr{C}}$ is a function that assigns each constant in $\mathscr{C}$ with a simple type built on $\mathscr{A}$.

Simple types

Typing environment

Given a signature a typing environment Γ is a finite set of ordered pairs $(x, \alpha) \in \mathscr{X} \times \mathscr{T}(\mathscr{A})$ such that $(x, \alpha), (x, \beta) \in \Gamma$ implies $\alpha = \beta$.

Given a typing environment Γ such that for every $(y, \beta) \in \Gamma$ $y \neq x$, we write "$\Gamma, x{:}\alpha$" for the typing environment "$\Gamma \cup \{(x, \alpha)\}$".

Simple types

Typing environment

Given a signature a typing environment Γ is a finite set of ordered pairs $(x, \alpha) \in \mathscr{X} \times \mathscr{T}(\mathscr{A})$ such that $(x, \alpha), (x, \beta) \in \Gamma$ implies $\alpha = \beta$.

Given a typing environment Γ such that for every $(y, \beta) \in \Gamma$ $y \neq x$, we write "$\Gamma, x{:}\alpha$" for the typing environment "$\Gamma \cup \{(x, \alpha)\}$".

Typing judgement

A typing judgement is an expression of the form

$$\Gamma \vdash t : \alpha$$

where Γ is a typing environment, t a λ-term, and α a simple type.

Simple types

Typing rules

$$\Gamma \vdash c : \tau(c)$$

$$\Gamma, x : \alpha \vdash x : \alpha$$

$$\frac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash (\lambda x.\, t) : (\alpha \to \beta)}$$

$$\frac{\Gamma \vdash t : (\alpha \to \beta) \quad \Gamma \vdash u : \alpha}{\Gamma \vdash (t\, u) : \beta}$$

Simple types

Typable terms

A λ-term t is typable if and only if there exist a typing environment Γ and a simple type α such that

$$\Gamma \vdash t : \alpha$$

Simple types

Typable terms

A λ-term t is typable if and only if there exist a typing environment Γ and a simple type α such that

$$\Gamma \vdash t : \alpha$$

Normalization

Every typable term has a normal form.

Interpretation

Definition à la Church

Let $\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$ be a signature, and let $(\mathscr{X}_\alpha)_{\alpha \in \mathscr{T}(\mathscr{A})}$ be a family of countably infinite disjoint sets, disjoint from $\mathscr{C}$, whose elements are called *typed λ-variables*. The set of typed λ-terms is inductively defined as follows:

- every $c \in \mathscr{C}$ is a λ-term of type $\tau(c)$;
- every $x \in \mathscr{X}_\alpha$ is a λ-term of type α;
- if x is a λ-variable of type α and t is a λ-term of type β then $(\lambda x.\, t)$ is a λ-term of type $(\alpha \rightarrow \beta)$;
- if t is a λ-term of type $(\alpha \rightarrow \beta)$ and u is a λ-term of type α then $(t\, u)$ is a λ-term is a λ-term of type β.

Interpretation

Model

A model consists of a family of sets $(D_\alpha)_{\alpha \in \mathscr{T}(\mathscr{A})}$ and an interpretation function $\mathcal{I}$ defined on $\mathscr{C}$ such that:

- $D_{\alpha\to\beta} = {D_\beta}^{D_\alpha}$;
- for every $c \in \mathscr{C}$, $\mathcal{I}(c) \in D_{\tau(c)}$.

Valuation

A typed valuation ξ is a function from $\bigcup_{\alpha \in \mathscr{T}(\mathscr{A})} \mathscr{X}_\alpha$ into $\bigcup_{\alpha \in \mathscr{T}(\mathscr{A})} \mathscr{D}_\alpha$ such that:

- if $x \in \mathscr{X}_\alpha$ then $\xi(x) \in \mathscr{D}_\alpha$.

Interpretation

Interpretation

- $[\![c]\!]_\xi = \mathcal{I}(c)$
- $[\![x]\!]_\xi = \xi(x)$
- $[\![\lambda x.\, t]\!]_\xi = a \mapsto [\![t]\!]_{\xi[x:=a]}$
- $[\![t\, u]\!]_\xi = [\![t]\!]_\xi([\![u]\!]_\xi)$

Logical constants

Signature with logical constants

Let $\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$ be such that:

- $\mathscr{A} = \{\mathrm{e}, \mathrm{t}\}$;
- $\mathbf{not}, \mathbf{and}, \mathbf{or}, \mathbf{implies}, \mathbf{all}, \mathbf{exists} \in \mathscr{C}$;
- $\tau(\mathbf{not}) = \mathrm{t} \to \mathrm{t}$;
- $\tau(\mathbf{and}) = \mathrm{t} \to \mathrm{t} \to \mathrm{t}$;
- $\tau(\mathbf{or}) = \mathrm{t} \to \mathrm{t} \to \mathrm{t}$;
- $\tau(\mathbf{implies}) = \mathrm{t} \to \mathrm{t} \to \mathrm{t}$;
- $\tau(\mathbf{all}) = (\mathrm{e} \to \mathrm{t}) \to \mathrm{t}$;
- $\tau(\mathbf{exists}) = (\mathrm{e} \to \mathrm{t}) \to \mathrm{t}$.

Logical constants

Interpretation

Let $\mathscr{M} = ((D_\alpha)_{\alpha \in \mathscr{T}(\mathscr{A})}, \mathcal{I})$ be such that:

- $D_{\mathrm{t}} = 2$;
- $\mathcal{I}(\mathbf{not}) = \{(0,1),(1,0)\}$;
- $\mathcal{I}(\mathbf{and}) = \{(0,\{(0,0),(1,0)\}),(1,\{(0,0),(1,1)\})\}$;
- $\mathcal{I}(\mathbf{or}) = \{(0,\{(0,0),(1,1)\}),(1,\{(0,1),(1,1)\})\}$;
- $\mathcal{I}(\mathbf{implies}) = \{(0,\{(0,1),(1,1)\}),(1,\{(0,0),(1,1)\})\}$;
- $\mathcal{I}(\mathbf{all})(f) = 1$ iff $f(a) = 1$ for every $a \in D_{\mathrm{e}}$;
- $\mathcal{I}(\mathbf{exists})(f) = 1$ iff $f(a) = 1$ for some $a \in D_{\mathrm{e}}$.

Logical constants

Notations

We write:

- $\neg a$ for $(\mathbf{not}\, a)$;
- $(a \wedge b)$ for $((\mathbf{and}\, a)\, b)$;
- $(a \vee b)$ for $((\mathbf{or}\, a)\, b)$;
- $(a \rightarrow b)$ for $((\mathbf{implies}\, a)\, b)$;
- $(\forall x.\, a)$ for $(\mathbf{all}\, (\lambda x.\, a))$;
- $(\exists x.\, a)$ for $(\mathbf{exits}\, (\lambda x.\, a))$.