Curry-Howard Correspondence

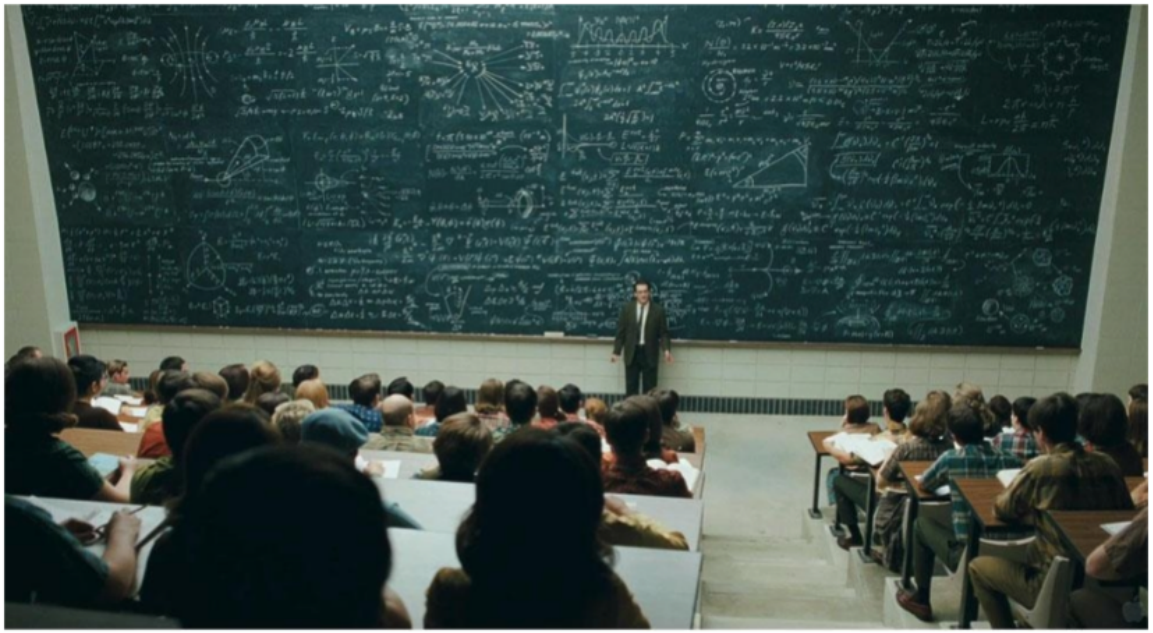
Federico Aschieri

Institut für Logic and Computation
Technische Universität Wien

7-8 February 2018

Curry-Howard Correspondence

Logical Proofs



Curry-Howard Correspondence

Computer Programs

Specification of the Program Behaviour

Coq Proof Assistant

Coq is based on Curry-Howard Correspondence

Summary

1. Lambda calculus
2. Curry-Howard for Propositional Intuitionistic Logic
3. Curry-Howard for Classical and Gödel Logic
4. Curry-Howard for First-Order Classical Logic Arithmetic

LAMBDA CALCULUS

Church (1936)

Lambda calculus

λ-calculus is a programming language whose basic objects are functions

Lambda calculus

λ-calculus is a programming language whose basic objects are functions

Definition of functions:

$$f = x \mapsto x^2 + 1$$

Lambda calculus

λ-calculus is a programming language whose basic objects are functions

Definition of functions:

$$f = x \mapsto x^2 + 1$$

Application of a function to an argument:

$$f(a) = a^2 + 1$$

Lambda calculus

λ-calculus is a programming language whose basic objects are functions

Definition of functions:

$$f = x \mapsto x^2 + 1$$

Application of a function to an argument:

$$f(a) = a^2 + 1$$

In λ-calculus:

$$\lambda x\ x^2 + 1$$

$$(\lambda x\ x^2 + 1)\ a \to a^2 + 1$$

Lambda Terms

1. Variables $x, y, z, \ldots$ are λ-terms

Lambda Terms

1. Variables $x, y, z, \ldots$ are λ-terms
2. If u is a λ-term, then $\lambda x\, u$ is a λ-term

Lambda Terms

1. Variables $x, y, z, \ldots$ are λ-terms
2. If u is a λ-term, then $\lambda x\, u$ is a λ-term
3. If t and u are λ-terms, then $t\, u$ is a λ-term

Lambda Terms

1. Variables $x, y, z, \ldots$ are λ-terms
2. If u is a λ-term, then $\lambda x\, u$ is a λ-term
3. If t and u are λ-terms, then $t\, u$ is a λ-term

(x is said to be *bound* in $\lambda x\, u$. Variables which are not bound are said to be *free*)

Lambda Terms

1. Variables $x, y, z, \ldots$ are λ-terms
2. If u is a λ-term, then $\lambda x\, u$ is a λ-term
3. If t and u are λ-terms, then $t\, u$ is a λ-term

(x is said to be *bound* in $\lambda x\, u$. Variables which are not bound are said to be *free*)

Notation: $t\, u\, u_1\ \ldots u_n$ stands for $(((t\, u)\, u_1)\ \ldots u_n)$

Lambda Terms

1. Variables $x, y, z, \ldots$ are λ-terms
2. If u is a λ-term, then $\lambda x\, u$ is a λ-term
3. If t and u are λ-terms, then $t\, u$ is a λ-term

(x is said to be *bound* in $\lambda x\, u$. Variables which are not bound are said to be *free*)

Notation: $t\, u\, u_1\ \ldots u_n$ stands for $(((t\, u)\, u_1)\ \ldots u_n)$

$$\lambda x\, \lambda y\, x$$

$$\lambda x\, \lambda y\, z$$

$$\lambda f\, \lambda x\, f(f(x))$$

$$= \text{(By renaming of bound variable)}$$

$$\lambda g\, \lambda x\, g(g(x))$$

Reduction Rules

Redex :

$$(\lambda x\, u)\, t \mapsto u[t/x]$$

(t must not contain as free variables bound variables of u)

Reduction Rules

Redex :

$$(\lambda x\, u)\, t \mapsto u[t/x]$$

(t must not contain as free variables bound variables of u)

$$(\lambda x\, \lambda y\, x)\, t_1\, t_2$$

$$\mapsto$$

$$(\lambda y\, t_1)\, t_2$$

$$\mapsto$$

$$t_1$$

Reduction Rules

$$(\lambda x\, u)\, t \mapsto u[t/x]$$

(t must not contain as free variables bound variables of u)

$$(\lambda x\, \lambda y\, x)\, y\, t_2$$

$$\mapsto$$

$$(\lambda y\, y)\, t_2$$

$$\mapsto$$

$$t_2$$

Examples: Pairs

$$\langle t, u \rangle,\ \pi_0,\ \pi_1$$

Examples: Pairs

$$\langle t, u \rangle,\ \pi_0,\ \pi_1$$

$$\langle t, u \rangle \pi_0 \mapsto t$$

$$\langle t, u \rangle \pi_1 \mapsto u$$

Examples: Pairs

$$\langle t, u \rangle,\ \pi_0,\ \pi_1$$

$$\langle t, u \rangle \pi_0 \mapsto t$$
$$\langle t, u \rangle \pi_1 \mapsto u$$

$$\langle t, u \rangle := \lambda x\, x\, t\, u$$
$$\pi_0 := \lambda x\, \lambda y\, x$$
$$\pi_1 := \lambda x\, \lambda y\, y$$

Examples: Pairs

$$\langle t, u\rangle,\ \pi_0,\ \pi_1$$

$$\langle t, u\rangle \pi_0 \mapsto t$$
$$\langle t, u\rangle \pi_1 \mapsto u$$

$$\langle t, u\rangle := \lambda x\, x\, t\, u$$
$$\pi_0 := \lambda x\, \lambda y\, x$$
$$\pi_1 := \lambda x\, \lambda y\, y$$

$$\langle t, u\rangle \pi_0 = (\lambda x\, x\, t\, u)(\lambda x\, \lambda y\, x) \mapsto (\lambda x\, \lambda y\, x)\, t\, u \mapsto (\lambda y\, t) u \mapsto t$$

Examples: Pairs

$$\langle t, u\rangle,\ \pi_0,\ \pi_1$$

$$\langle t, u\rangle \pi_0 \mapsto t$$
$$\langle t, u\rangle \pi_1 \mapsto u$$

$$\langle t, u\rangle := \lambda x\, x\, t\, u$$
$$\pi_0 := \lambda x\, \lambda y\, x$$
$$\pi_1 := \lambda x\, \lambda y\, y$$

$$\langle t, u\rangle \pi_0 = (\lambda x\, x\, t\, u)(\lambda x\, \lambda y\, x) \mapsto (\lambda x\, \lambda y\, x)\, t\, u \mapsto (\lambda y\, t)u \mapsto t$$

$$\langle t, u\rangle \pi_1 = (\lambda x\, x\, t\, u)(\lambda x\, \lambda y\, y) \mapsto (\lambda x\, \lambda y\, y)\, t\, u \mapsto (\lambda y\, y)u \mapsto u$$

Examples: Booleans

$$\mathit{if}\ w\ \mathit{then}\ t\ \mathit{else}\ u,\quad \mathit{True}, \mathit{False}$$

Examples: Booleans

$$\mathit{if}\ w\ \mathit{then}\ t\ \mathit{else}\ u,\ \ \mathit{True}, \mathit{False}$$

$$\mathit{if}\ \mathit{True}\ \mathit{then}\ t\ \mathit{else}\ u \mapsto t$$

$$\mathit{if}\ \mathit{False}\ \mathit{then}\ t\ \mathit{else}\ u \mapsto u$$

Examples: Booleans

$$\mathit{if}\ w\ \mathit{then}\ t\ \mathit{else}\ u,\ \ \mathit{True}, \mathit{False}$$

$$\mathit{if}\ \mathit{True}\ \mathit{then}\ t\ \mathit{else}\ u \mapsto t$$

$$\mathit{if}\ \mathit{False}\ \mathit{then}\ t\ \mathit{else}\ u \mapsto u$$

$$\mathit{if}\ w\ \mathit{then}\ t\ \mathit{else}\ u := \langle t, u\rangle\ w$$

$$\mathit{True} := \pi_0 := \lambda x\ \lambda y\ x$$

$$\mathit{False} := \pi_1 := \lambda x\ \lambda y\ y$$

Examples: Booleans

$$\mathit{if}\ w\ \mathit{then}\ t\ \mathit{else}\ u,\ \ \mathit{True}, \mathit{False}$$

$$\mathit{if}\ \mathit{True}\ \mathit{then}\ t\ \mathit{else}\ u \mapsto t$$

$$\mathit{if}\ \mathit{False}\ \mathit{then}\ t\ \mathit{else}\ u \mapsto u$$

$$\mathit{if}\ w\ \mathit{then}\ t\ \mathit{else}\ u := \langle t, u\rangle\ w$$

$$\mathit{True} := \pi_0 := \lambda x\ \lambda y\ x$$

$$\mathit{False} := \pi_1 := \lambda x\ \lambda y\ y$$

$$\mathit{if}\ \mathit{True}\ \mathit{then}\ t\ \mathit{else}\ u = \langle t, u\rangle \pi_0 \mapsto t$$

Examples: Numbers

$$\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f\, x)\ldots)}^{n \text{ times}}$$

Examples: Numbers

$$\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$$

$$\mathit{IsZero}\, \overline{0} \mapsto \mathit{True}$$

$$\mathit{isZero}\, \overline{n+1} \mapsto \mathit{False}$$

Examples: Numbers

$$\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$$

$$\mathit{IsZero}\, \overline{0} \mapsto \mathit{True}$$

$$\mathit{isZero}\, \overline{n+1} \mapsto \mathit{False}$$

$$\overline{0} = \lambda f \lambda x\, x$$

Examples: Numbers

$$\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$$

$$\mathit{IsZero}\, \overline{0} \mapsto \mathit{True}$$

$$\mathit{isZero}\, \overline{n+1} \mapsto \mathit{False}$$

$$\overline{0} = \lambda f \lambda x\, x$$

$$(\lambda f \lambda x\, x)\, ?\, \mathit{True} \mapsto \mathit{True}$$

Examples: Numbers

$$\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f\, x)\ldots)}^{n \text{ times}}$$

$$\mathit{IsZero}\, \overline{0} \mapsto \mathit{True}$$

$$\mathit{isZero}\, \overline{n+1} \mapsto \mathit{False}$$

$$\overline{0} = \lambda f \lambda x\, x$$

$$(\lambda f \lambda x\, x)\, ?\, \mathit{True} \mapsto \mathit{True}$$

$$(\lambda f \lambda x\, fx)\, (\lambda z\, \mathit{False})? \mapsto (\lambda z\, \mathit{False})? \mapsto \mathit{False}$$

Examples: Numbers

$$\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$$

$$\mathit{IsZero}\, \overline{0} \mapsto \mathit{True}$$

$$\mathit{isZero}\, \overline{n+1} \mapsto \mathit{False}$$

$$\overline{0} = \lambda f \lambda x\, x$$

$$(\lambda f \lambda x\, x)\, ?\, \mathit{True} \mapsto \mathit{True}$$

$$(\lambda f \lambda x\, fx)\, (\lambda z\, \mathit{False})? \mapsto (\lambda z\, \mathit{False})? \mapsto \mathit{False}$$

$$\mathit{isZero} := \lambda n\, n\, (\lambda z\, \mathit{False})\, \mathit{True}$$

Examples: Numbers

$$\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$$

Examples: Numbers

$$\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$$

$$\mathit{Add}\, \overline{n}\, \overline{m} \mapsto \overline{n+m}$$

Examples: Numbers

$$\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$$

$$\mathit{Add}\, \overline{n}\, \overline{m} \mapsto \overline{n+m}$$

$$\mathit{Add} := \lambda m\, \lambda n\, \lambda f\, \lambda x\; n\, f\, (m\, f\, x)$$

Examples: Numbers

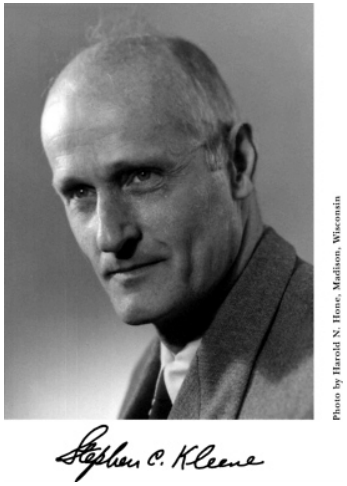
$$\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$$

$$\mathit{Add}\, \overline{n}\, \overline{m} \mapsto \overline{n+m}$$

$$\mathit{Add} := \lambda m\, \lambda n\, \lambda f\, \lambda x\; n\, f\, (m\, f\, x)$$

$$\mathit{Add}\, \overline{2}\, \overline{3} \mapsto \lambda f\, \lambda x\, \overline{2}\, f\, (\overline{3}\, f\, x) \mapsto \lambda f\, \lambda x\, \overline{2}\, f\, (f(f(f\, x)))$$
$$\mapsto \lambda f\, \lambda x\; f(f(f(f(f\, x)))) = \overline{5}$$

N-1?



Uhm...

N-1?

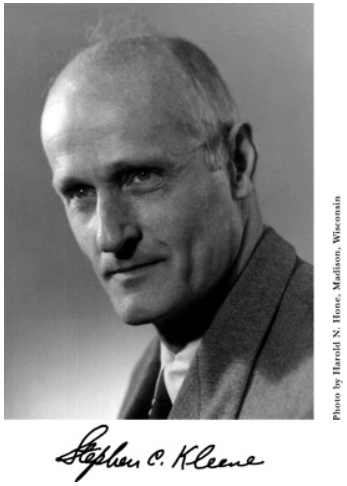


Photo by Harold N. Hone, Madison, Wisconsin
Stephen C. Kleene

Uhm...


DENTIST

Kleene's Predecessor Function

Given $\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$, we want to the iterate successor function from 0 exactly $n - 1$ times.

Kleene's Predecessor Function

Given $\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$, we want to the iterate successor function from 0 exactly $n - 1$ times.

$$\langle \mathit{True}, n \rangle \to \langle \mathit{False}, n \rangle$$

Kleene's Predecessor Function

Given $\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$, we want to the iterate successor function from 0 exactly $n-1$ times.

$$\langle \mathit{True}, n \rangle \to \langle \mathit{False}, n \rangle$$

$$\langle \mathit{False}, n \rangle \to \langle \mathit{False}, n+1 \rangle$$

Kleene's Predecessor Function

Given $\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$, we want to the iterate successor function from 0 exactly $n-1$ times.

$$\langle \mathit{True}, n\rangle \to \langle \mathit{False}, n\rangle$$

$$\langle \mathit{False}, n\rangle \to \langle \mathit{False}, n+1\rangle$$

$$\lambda p\, \mathit{if}\ p\,\pi_0\ \mathit{then}\ \langle \mathit{False}, p\,\pi_1\rangle\ \mathit{else}\ \langle \mathit{False}, p\,\pi_1 + 1\rangle$$

Kleene's Predecessor Function

Given $\overline{n} := \lambda f\, \lambda x\, \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$, we want to the iterate successor function from 0 exactly $n-1$ times.

$$\langle \mathit{True}, n\rangle \to \langle \mathit{False}, n\rangle$$

$$\langle \mathit{False}, n\rangle \to \langle \mathit{False}, n+1\rangle$$

$$\lambda p\, \mathit{if}\ p\,\pi_0\ \mathit{then}\ \langle \mathit{False}, p\,\pi_1\rangle\ \mathit{else}\ \langle \mathit{False}, p\,\pi_1 + 1\rangle$$

$$\mathit{Pred} :=$$

$$\lambda n.\ n(\lambda p\, \mathit{if}\ p\,\pi_0\ \mathit{then}\ \langle \mathit{False}, p\,\pi_1\rangle\ \mathit{else}\ \langle \mathit{False}, p\,\pi_1 + 1\rangle)\ \langle \mathit{False}, 0\rangle\ \pi_1$$

Higher Order Recursion

$$R$$

Higher Order Recursion

$$R$$

$$R\,u\,v\,\overline{0} \mapsto u$$

$$R\,u\,v\,\overline{n+1} \mapsto v\overline{n}(R\,u\,v\,\overline{n})$$

Higher Order Recursion

$$R$$

$$R\,u\,v\,\overline{0} \mapsto u$$

$$R\,u\,v\,\overline{n+1} \mapsto v\overline{n}(R\,u\,v\,\overline{n})$$

$$\langle \overline{n}, R\,u\,v\,\overline{n} \rangle \to \langle \overline{n+1}, v\overline{n}(R\,u\,v\,\overline{n}) \rangle$$

Higher Order Recursion

$$R$$

$$R\,u\,v\,\overline{0} \mapsto u$$

$$R\,u\,v\,\overline{n+1} \mapsto v\overline{n}(R\,u\,v\,\overline{n})$$

$$\langle \overline{n}, R\,u\,v\,\overline{n}\rangle \to \langle \overline{n+1}, v\overline{n}(R\,u\,v\,\overline{n})\rangle$$

$$\lambda p\,\langle p\,\pi_0 + 1, v(p\,\pi_0)(p\,\pi_1)\rangle$$

Higher Order Recursion

$$R$$

$$R\,u\,v\,\overline{0} \mapsto u$$

$$R\,u\,v\,\overline{n+1} \mapsto v\overline{n}(R\,u\,v\,\overline{n})$$

$$\langle \overline{n}, R\,u\,v\,\overline{n}\rangle \to \langle \overline{n+1}, v\overline{n}(R\,u\,v\,\overline{n})\rangle$$

$$\lambda p\,\langle p\,\pi_0 + 1, v(p\,\pi_0)(p\,\pi_1)\rangle$$

$$R := \lambda u \lambda v \lambda n\; n\;(\lambda p\,\langle p\,\pi_0 + 1, v(p\,\pi_0)(p\,\pi_1)\rangle)\;\langle \overline{0}, u\rangle$$

Expressivity

Theorem

All Turing–computable functions are representable in λ-calculus.

Expressivity

Theorem

All Turing–computable functions are representable in λ-calculus.

Also weird programs, as:

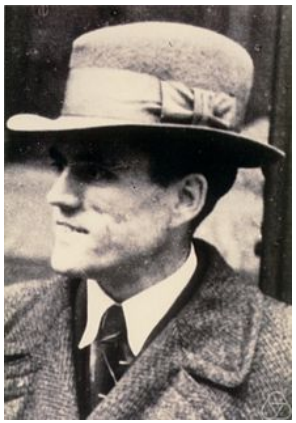
$$(\lambda x\, x\, x)(\lambda x\, x\, x)$$

Properties

Theorem (Confluence)

Assume $t \mapsto^* u$ and $t \mapsto^* v$, where u and v are normal forms, i.e. no reduction rule can be applied to them. Then u and v are the same term.

INTUITIONISTIC PROPOSITIONAL LOGIC



Natural Deduction (Gentzen 1934, Prawitz 1965)

Assumptions, Introduction and Elimination Rules

Assumptions, Introduction and Elimination Rules

Reasoning starts from assumptions

Assumptions, Introduction and Elimination Rules

Reasoning starts from assumptions

To every logical symbol belongs precisely one inference figure which introduces the symbol and one which eliminates it (Gentzen, 1934).

Assumptions, Introduction and Elimination Rules

Reasoning starts from assumptions

To every logical symbol belongs precisely one inference figure which introduces the symbol and one which eliminates it (Gentzen, 1934).

The introductions represent, so to speak, the definitions of the symbols concerned

Assumptions, Introduction and Elimination Rules

Reasoning starts from assumptions

To every logical symbol belongs precisely one inference figure which introduces the symbol and one which eliminates it (Gentzen, 1934).

The introductions represent, so to speak, the definitions of the symbols concerned

and the eliminations are no more, in the final analysis, than consequences of these definitions (ibid.)

Natural Deduction Trees

A

Natural Deduction Trees

$$A$$

$$\dfrac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \to B}$$

Natural Deduction Trees

$$A$$

$$\begin{array}{c} [A] \\ \vdots \\ B \end{array}$$
$$\overline{A \to B}$$

$$\frac{A \to B \qquad A}{B}$$

Example

$$\dfrac{\dfrac{\dfrac{\dfrac{}{}}{}}{}}{(A \to B) \to ((B \to C) \to (A \to C))}$$

Example

$$\dfrac{\dfrac{\dfrac{[A \to B]}{\dfrac{}{\dfrac{}{\dfrac{}{(B \to C) \to (A \to C)}}}}}{(A \to B) \to ((B \to C) \to (A \to C))}}$$

Example

$$\dfrac{\dfrac{[A \to B] \qquad [B \to C]}{A \to C}}{(B \to C) \to (A \to C)}$$
$$\overline{(A \to B) \to ((B \to C) \to (A \to C))}$$

Example

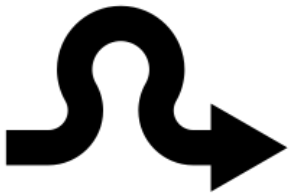
$$\dfrac{\dfrac{\dfrac{\dfrac{[A \to B] \qquad [A]}{\dfrac{[B \to C]}{C}}}{A \to C}}{(B \to C) \to (A \to C)}}{(A \to B) \to ((B \to C) \to (A \to C))}$$

Example

$$\dfrac{\dfrac{\dfrac{[B \to C] \qquad \dfrac{[A \to B] \qquad [A]}{B}}{C}}{A \to C}}{(B \to C) \to (A \to C)}$$
$$(A \to B) \to ((B \to C) \to (A \to C))$$

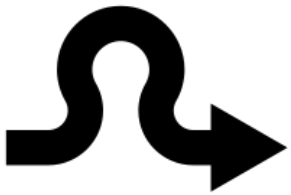
Detour Removal

$$\dfrac{\dfrac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \to B} \qquad \begin{array}{c}\vdots \\ A\end{array}}{B}$$



Detour Removal

$$\dfrac{\dfrac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \to B} \quad \begin{array}{c}\vdots\\ A\end{array}}{B}$$



$$\begin{array}{c}\vdots\\ A\\ \vdots\\ B\end{array}$$

Simply Typed λ-Terms

$$\frac{}{x^A : A}$$

Simply Typed λ-Terms

$$\dfrac{\dfrac{}{x^A : A}}{\lambda x^A\, u : A \to B}$$

$$\dfrac{\begin{array}{c} x^A : A \\ \vdots \\ u : B \end{array}}{\lambda x^A\, u : A \to B}$$

Simply Typed λ-Terms

$$\frac{}{x^A : A}$$

$$\frac{\begin{array}{c} x^A : A \\ \vdots \\ u : B \end{array}}{\lambda x^A\, u : A \to B}$$

$$\frac{t : A \to B \qquad u : A}{tu : B}$$

Simply Typed λ-Terms

$$\frac{}{x^A : A}$$

$$\frac{\begin{array}{c} x^A : A \\ \vdots \\ u : B \end{array}}{\lambda x^A\, u : A \to B}$$

$$\frac{t : A \to B \qquad u : A}{tu : B}$$

CURRY-HOWARD ISOMORPHISM

λ-TERM of TYPE $A \implies$ PROOF of A

Simply Typed λ-Terms

$$\frac{}{x^A : A}$$

$$\frac{\begin{array}{c} x^A : A \\ \vdots \\ u : B \end{array}}{\lambda x^A\, u : A \to B}$$

$$\frac{t : A \to B \qquad u : A}{tu : B}$$

CURRY-HOWARD ISOMORPHISM

λ-TERM of TYPE $A \implies$ PROOF of A

TYPES of its free variables $\implies$ HYPOTHESES of the Proof

Example

$$\lambda x^{A \to B} \lambda y^{B \to C} \lambda z^{A}\ y(x\,z) : (A \to B) \to ((B \to C) \to (A \to C))$$

Properties

Theorem (Subject Reduction)

If $s \mapsto s'$ and $s : A$, then $s' : A$. Moreover, the free variables of s' are among those of s.

Properties

Theorem (Subject Reduction)

If $s \mapsto s'$ and $s : A$, then $s' : A$. Moreover, the free variables of s' are among those of s.

Computation as Elimination of Detours

$$\dfrac{\dfrac{\begin{array}{c} x^A : A \\ \vdots \\ u : B \end{array}}{\lambda x^A u : A \to B} \qquad t : A}{(\lambda x^A u) t : B}$$

Properties

Theorem (Subject Reduction)

If $s \mapsto s'$ and $s : A$, then $s' : A$. Moreover, the free variables of s' are among those of s.

Computation as Elimination of Detours

$$\dfrac{\dfrac{\begin{array}{c} x^A : A \\ \vdots \\ u : B \end{array}}{\lambda x^A u : A \to B} \qquad t : A}{(\lambda x^A u) t : B}$$

$$u[t/x^A] : B$$

Properties

Theorem (Subformula Property)

Assume $t : A$ is in normal form. Then the type of any subterm of t is contained in A or in the type of some free variable of t.

Properties

Theorem (Subformula Property)

Assume $t : A$ is in normal form. Then the type of any subterm of t is contained in A or in the type of some free variable of t.

No logical detours

Properties

Theorem (Strong Normalization)

Assume $t : A$. Then t is strongly normalizable (SN), i.e. there is no infinite sequence of terms

$$t \mapsto t_1 \mapsto t_2 \ldots \mapsto t_n \mapsto t_{n+1} \mapsto \ldots$$

Key Lemma

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Key Lemma

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(t))$

Key Lemma

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(t))$

Lemma 1: u is SN $\Longrightarrow$ $longestred(u)$ exists by König Lemma!

Key Lemma

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(t))$

Lemma 1: u is SN $\implies$ $longestred(u)$ exists by König Lemma!

Lemma 2: t is SN and $u[t/x]$ is SN $\implies$ $(\lambda x\, u)t$ is SN

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 1:

Suppose $u = \lambda z^B w$.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 1:

Suppose $u = \lambda z^B w$.

$$size(w) < size(u)$$

$$longestred(w) = longestred(u)$$

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple: $(size(A), longestred(u), size(u))$

CASE 1:

Suppose $u = \lambda z^B w$.

$$size(w) < size(u)$$

$$longestred(w) = longestred(u)$$

By IH, $w[t/x]$ is SN.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple: $(size(A), longestred(u), size(u))$

CASE 1:

Suppose $u = \lambda z^B w$.

$$size(w) < size(u)$$

$$longestred(w) = longestred(u)$$

By IH, $w[t/x]$ is SN.

Therefore $u[t/x] = \lambda z^B\, w[t/x]$ is SN

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 2:

$u = y\, w_1 \ldots w_n$, with $y \neq x$.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 2:

$u = y\, w_1 \ldots w_n$, with $y \neq x$.

$$size(w_i) < size(u)$$

$$longestred(w_i) \leq longestred(u)$$

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 2:

$u = y\, w_1 \ldots w_n$, with $y \neq x$.

$$size(w_i) < size(u)$$

$$longestred(w_i) \leq longestred(u)$$

Thus $w_i[t/x] \in$ SN by IH.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 2:

$u = y\, w_1 \ldots w_n$, with $y \neq x$.

$$size(w_i) < size(u)$$

$$longestred(w_i) \leq longestred(u)$$

Thus $w_i[t/x] \in$ SN by IH.

Therefore $u[t/x] = y(w_1[t/x]) \ldots (w_n[t/x]) \in$ SN.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 3:

$u = (\lambda y^B\, v)\, w\, w_1 \ldots w_n$.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 3:

$u = (\lambda y^B\, v)\, w\, w_1 \ldots w_n$.

Let $v' = v[t/x]$, $w' = w[t/x]$, $w'_i = w_i[t/x]$, $(i = 1 \ldots n)$.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 3:

$u = (\lambda y^B\, v)\, w\, w_1 \ldots w_n$.

Let $v' = v[t/x]$, $w' = w[t/x]$, $w'_i = w_i[t/x]$, $(i = 1 \ldots n)$.

By IH, v', w', w'_i are SN, since the size of v, w, w_i is smaller than $size(u)$ and their longestred is at most that of $longestred(u)$.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 3:

$u = (\lambda y^B\, v)\, w\, w_1 \ldots w_n$.

Let $v' = v[t/x]$, $w' = w[t/x]$, $w'_i = w_i[t/x]$, $(i = 1 \ldots n)$.

By IH, v', w', w'_i are SN, since the size of v, w, w_i is smaller than $size(u)$ and their longestred is at most that of $longestred(u)$.

$longestred(v[w/y^B]w_1 \ldots w_n) < longestred(u)$

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 3:

$u = (\lambda y^B\, v)\, w\, w_1 \dots w_n$.

Let $v' = v[t/x]$, $w' = w[t/x]$, $w'_i = w_i[t/x]$, $(i = 1 \dots n)$.

By IH, v', w', w'_i are SN, since the size of v, w, w_i is smaller than $size(u)$ and their longestred is at most that of $longestred(u)$.

$longestred(v[w/y^B]w_1 \dots w_n) < longestred(u)$

By IH, $v'[w'/y^B]w'_1 \dots w'_n$ is SN, thus
$u[t/x] = (\lambda y^B.v')w'w'_1 \dots w'_n$ is SN by Lemma 2.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 4:

$u = x\, w_1 \ldots w_n$, with $n > 0$.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 4:

$u = x\, w_1 \ldots w_n$, with $n > 0$.

$w_i' = w_i[t/x]$ is SN by IH.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 4:

$u = x\, w_1 \ldots w_n$, with $n > 0$.

$w'_i = w_i[t/x]$ is SN by IH.

$(x\, w_1 \ldots w_{n-1})\,[t/x] = t w'_1 \ldots w'_{n-1}$ is in SN by IH.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple: $(size(A), longestred(u), size(u))$

CASE 4:

$u = x\, w_1 \ldots w_n$, with $n > 0$.

$w'_i = w_i[t/x]$ is SN by IH.

$(x\, w_1 \ldots w_{n-1})\,[t/x] = t w'_1 \ldots w'_{n-1}$ is in SN by IH.

$u[t/x] = (t\, w'_1 \ldots w'_{n-1}\, z)\,[w'_n/z]$, with z fresh.

Lemma (Van Dalen, Levy, David)

If u is SN and $t : A$ is SN, then $u[t/x]$ is SN.

Proof by lexicographic induction on the triple:
$(size(A), longestred(u), size(u))$

CASE 4:

$u = x\, w_1 \ldots w_n$, with $n > 0$.

$w'_i = w_i[t/x]$ is SN by IH.

$(x\, w_1 \ldots w_{n-1})\,[t/x] = t w'_1 \ldots w'_{n-1}$ is in SN by IH.

$u[t/x] = (t\, w'_1 \ldots w'_{n-1}\, z)\,[w'_n/z]$, with z fresh.

$type(w'_n) < type(t) \implies u[t/x]$ *is* SN by IH.

Properties

Theorem (Strong Normalization)

Assume $t : A$. Then t is strongly normalizable, i.e. there is no infinite sequence of terms

$$t \mapsto t_1 \mapsto t_2 \ldots \mapsto t_n \mapsto t_{n+1} \mapsto \ldots$$

By induction on t.

Properties

Theorem (Strong Normalization)

Assume $t : A$. Then t is strongly normalizable, i.e. there is no infinite sequence of terms

$$t \mapsto t_1 \mapsto t_2 \ldots \mapsto t_n \mapsto t_{n+1} \mapsto \ldots$$

By induction on t.

$t = x^A$ is SN

Properties

Theorem (Strong Normalization)

Assume $t : A$. Then t is strongly normalizable, i.e. there is no infinite sequence of terms

$$t \mapsto t_1 \mapsto t_2 \ldots \mapsto t_n \mapsto t_{n+1} \mapsto \ldots$$

By induction on t.

$t = x^A$ is SN

$t = \lambda x^A u$. u is SN by IH, so t is SN.

Theorem (Strong Normalization)

Assume $t : A$. Then t is strongly normalizable, i.e. there is no infinite sequence of terms

$$t \mapsto t_1 \mapsto t_2 \ldots \mapsto t_n \mapsto t_{n+1} \mapsto \ldots$$

By induction on t.

$t = x^A$ is SN

$t = \lambda x^A u$. u is SN by IH, so t is SN.

$t = u\,v = x\,v[u/x]$, where u and v are SN by IH. By Lemma, t is SN.

Reducibility (Tait 1960s)

Reducibility (Tait 1960s)

$t \in Red_P$ iff $t : P$ is strongly normalizable

Reducibility (Tait 1960s)

$t \in Red_P$ iff $t : P$ is strongly normalizable

$t \in Red_{A \to B}$ iff forall $u \in Red_A$, $tu \in Red_B$

Reducibility (Tait 1960s)

$t \in Red_P$ iff $t : P$ is strongly normalizable

$t \in Red_{A \to B}$ iff forall $u \in Red_A$, $tu \in Red_B$

$$t \in Red_T \implies t \in SN$$

$$t : T \implies t \in Red_T$$

Conjunction

$$\frac{u : A \qquad v : B}{\langle u, v \rangle : A \wedge B}$$

Conjunction

$$\frac{u : A \qquad v : B}{\langle u, v \rangle : A \wedge B}$$

$$\frac{t : A \wedge B}{t\,\pi_0 : A}$$

$$\frac{t : A \wedge B}{t\,\pi_1 : B}$$

Conjunction

$$\frac{u : A \qquad v : B}{\langle u, v \rangle : A \wedge B}$$

$$\frac{t : A \wedge B}{t\,\pi_0 : A}$$

$$\frac{t : A \wedge B}{t\,\pi_1 : B}$$

$$\langle t_0, t_1 \rangle\,\pi_i \mapsto t_i \qquad (i \in \{0, 1\})$$

Disjunction

$$\frac{u : A}{\iota_0(u) : A \vee B}$$

$$\frac{u : B}{\iota_1(u) : A \vee B}$$

Disjunction

$$\frac{u : A}{\iota_0(u) : A \vee B}$$

$$\frac{u : B}{\iota_1(u) : A \vee B}$$

$$\frac{\begin{array}{ccc} & \begin{array}{c} x^A : A \\ \vdots \\ u : C \end{array} & \begin{array}{c} y^B : B \\ \vdots \\ v : C \end{array} \\ t : A \vee B & & \end{array}}{t\,[x^A.u, y^B.v] : C}$$

Disjunction

$$\frac{u : A}{\iota_0(u) : A \vee B}$$

$$\frac{u : B}{\iota_1(u) : A \vee B}$$

$$\frac{\begin{array}{ccc} & x^A : A & y^B : B \\ & \vdots & \vdots \\ t : A \vee B & u : C & v : C \end{array}}{t\,[x^A.u, y^B.v] : C}$$

$$\iota_0(u)[x.v, y.w] \mapsto v[u/x]$$

$$\iota_1(u)[x.v, y.w] \mapsto w[u/y]$$

Logical Detours

$$\dfrac{\dfrac{u : A \qquad v : B}{\langle u, v \rangle : A \wedge B}}{\langle u, v \rangle \pi_0 : A} \quad \mapsto \quad u : A$$

Natural Deduction

$$\dfrac{\dfrac{u : A}{\iota_0(u) : A \vee B} \qquad \begin{array}{c} x^A : A \\ \vdots \\ v : C \end{array} \qquad \begin{array}{c} y^B : B \\ \vdots \\ w : C \end{array}}{\iota_0(u)[x^A.v, y^B.w] : C}$$

$\mapsto$

$$v[u/x^A] : C$$

Properties

Theorem (Disjunction Property)

Suppose $t : A \vee B$ is in normal form, i.e. no reduction rule can be applied to t, and t does not contain free variables. Then $t = \iota_0(u)$ and $u : A$ or $t = \iota_1(u)$ and $u : B$.

Expressivity

Church Numerals cannot be given a useful type!

$$\lambda f^{Y \to Y} \lambda x^{Y} f(f(x)) : (Y \to Y) \to Y \to Y := NAT$$

Expressivity

Church Numerals cannot be given a useful type!

$$\lambda f^{Y \to Y} \lambda x^{Y} f(f(x)) : (Y \to Y) \to Y \to Y := NAT$$

$$\langle \overline{n}, R\, u\, v\, \overline{n} \rangle \to \langle \overline{n+1}, v \overline{n} (R\, u\, v\, \overline{n}) \rangle$$

Expressivity

Church Numerals cannot be given a useful type!

$$\lambda f^{Y \to Y} \lambda x^{Y} f(f(x)) : (Y \to Y) \to Y \to Y := NAT$$

$$\langle \overline{n}, R\, u\, v\, \overline{n} \rangle \to \langle \overline{n+1}, v \overline{n} (R\, u\, v\, \overline{n}) \rangle$$

$$\lambda p \, \langle p\, \pi_0 + 1, v (p\, \pi_0)(p\, \pi_1) \rangle : (NAT \wedge NAT) \to (NAT \wedge NAT)$$

Expressivity

Church Numerals cannot be given a useful type!

$$\lambda f^{Y \to Y} \lambda x^{Y} f(f(x)) : (Y \to Y) \to Y \to Y := NAT$$

$$\langle \overline{n}, R\, u\, v\, \overline{n} \rangle \to \langle \overline{n+1}, v\overline{n}(R\, u\, v\, \overline{n}) \rangle$$

$$\lambda p \, \langle p\, \pi_0 + 1, v(p\, \pi_0)(p\, \pi_1) \rangle : (NAT \wedge NAT) \to (NAT \wedge NAT)$$

$$R := \lambda u \lambda v \lambda n^{NAT} \; n \, (\lambda p \, \langle p\, \pi_0 + 1, v(p\, \pi_0)(p\, \pi_1) \rangle) \, \langle \overline{0}, u \rangle$$

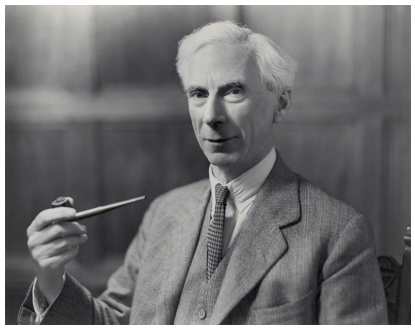
Theorem (Schwichtenberg)

Suppose that $t : A$, k be the size of t and r be the size of A. Then t reduces to a normal term in at most

$$\underbrace{2^{2^{\cdot^{\cdot^{2^k}}}}}_{r+1}$$

steps.

INTUITIONISTIC SECOND ORDER PROPOSITIONAL LOGIC



Polymorphic λ-calculus (Girard 1971, System F)

Types

1. Type Variables $A, B, C, X, Y, \ldots$ are types.

Types

1. Type Variables $A, B, C, X, Y, \ldots$ are types.
2. If A and B are types, $A \to B$ is a type.

Types

1. Type Variables $A, B, C, X, Y, \ldots$ are types.
2. If A and B are types, $A \to B$ is a type.
3. If A is a type and X a type variable, then $\forall X\, A$ is a type.

Polymorphic λ-Terms

$$\frac{}{x^A : A}$$

Polymorphic λ-Terms

$$\frac{}{x^A : A}$$

$$\frac{\begin{array}{c} x^A : A \\ \vdots \\ u : B \end{array}}{\lambda x^A\, u : A \to B}$$

$$\frac{t : A \to B \qquad u : A}{tu : B}$$

Polymorphic λ-Terms

$$\frac{t : A}{\Lambda X\, t : \forall X\, A}$$

(X is not in the types of the free variables of t)

Polymorphic λ-Terms

$$\frac{t : A}{\Lambda X\, t : \forall X\, A}$$

(X is not in the types of the free variables of t)

$$\frac{t : \forall X\, A}{t\, B : A[B/X]}$$

Polymorphic λ-Terms

$$\frac{t : A}{\Lambda X\, t : \forall X\, A}$$

(X is not in the types of the free variables of t)

$$\frac{t : \forall X\, A}{t\, B : A[B/X]}$$

$$(\Lambda X\, t)\, B \mapsto t[B/X]$$

Example

$$\dfrac{\dfrac{\dfrac{}{}\qquad\dfrac{}{}}{}}{(\forall X.\,(A \to B \to X) \to X) \to A}$$

Example

$$\dfrac{\dfrac{[\forall X.\,(A \to B \to X) \to X)]}{\dfrac{A}{(\forall X.\,(A \to B \to X) \to X) \to A}}}{}$$

Example

$$\dfrac{\dfrac{[\forall X.\,(A \to B \to X) \to X)]}{(A \to B \to A) \to A)} \qquad \dfrac{\qquad}{\qquad}}{\dfrac{A}{(\forall X.\,(A \to B \to X) \to X) \to A}}$$

Example

$$\dfrac{\dfrac{[\forall X.\,(A \to B \to X) \to X)]}{(A \to B \to A) \to A)} \qquad \dfrac{\qquad}{A \to B \to A}}{\dfrac{A}{(\forall X.\,(A \to B \to X) \to X) \to A}}$$

Example

$$\dfrac{\dfrac{[\forall X.\,(A \to B \to X) \to X)]}{(A \to B \to A) \to A)} \qquad \dfrac{\dfrac{[A]}{B \to A}}{A \to B \to A}}{\dfrac{A}{(\forall X.\,(A \to B \to X) \to X) \to A}}$$

Booleans

$$\mathsf{Bool} := \forall X.\ X \to X \to X$$

Booleans

$$\mathsf{Bool} := \forall X.\ X \to X \to X$$

$$\mathit{True} := \Lambda X \lambda x^X \lambda y^X\ x$$

$$\mathit{False} := \Lambda X \lambda x^X \lambda y^X\ y$$

Booleans

$$\mathsf{Bool} := \forall X.\ X \to X \to X$$

$$\mathit{True} := \Lambda X \lambda x^X \lambda y^X\ x$$

$$\mathit{False} := \Lambda X \lambda x^X \lambda y^X\ y$$

$$w : \mathsf{Bool}, t : C, u : C$$

$$\mathit{if}\ w\ \mathit{then}\ t\ \mathit{else}\ u := w\ C\ t\ u$$

Booleans

$$\mathsf{Bool} := \forall X.\ X \to X \to X$$

$$\mathit{True} := \Lambda X \lambda x^X \lambda y^X\ x$$

$$\mathit{False} := \Lambda X \lambda x^X \lambda y^X\ y$$

$$w : \mathsf{Bool}, t : C, u : C$$

$$\mathit{if}\ w\ \mathit{then}\ t\ \mathit{else}\ u := w\ C\ t\ u$$

$$\mathit{if}\ \mathit{True}\ \mathit{then}\ t\ \mathit{else}\ u = (\Lambda X \lambda x^X \lambda y^X\ x)\ C\ t\ u \mapsto (\lambda x^C \lambda y^C\ x)\ t\ u \mapsto t$$

Church Numerals

$$\mathsf{NAT} := \forall X.\ (X \to X) \to X \to X$$

Church Numerals

$$\mathsf{NAT} := \forall X.\ (X \to X) \to X \to X$$

Theorem

Every normal term of type NAT *is of the form:*

$$\overline{n} := \Lambda X \lambda f^{X \to X}\ \lambda x^{X}\ \overbrace{f(\ldots(f}^{n \text{ times}}\, x)\ldots)$$

Pairs

$$A \wedge B := \forall X.\, (A \to B \to X) \to X$$

Pairs

$$A \wedge B := \forall X.\, (A \to B \to X) \to X$$

$$u : A, v : B$$

$$\langle u, v \rangle := \Lambda X\, \lambda x^{A \to B \to X}\, x\, u\, v$$

Pairs

$$A \wedge B := \forall X.\, (A \to B \to X) \to X$$

$$u : A, v : B$$

$$\langle u, v \rangle := \Lambda X\, \lambda x^{A \to B \to X}\, x\, u\, v$$

$$u : A \wedge B$$

$$u\, \pi_0 := u\, A\, (\lambda x^A \lambda y^B x)$$

$$u\, \pi_1 := u\, B\, (\lambda x^A \lambda y^B y)$$

Expressivity

Theorem

The terms of type Nat $\to$ Nat *represent all computable functions that are provably total in second-order Arithmetic.*

Still λ-calculus

Let t^* the term obtained from t by erasing types and Λ. Then:

$$t \mapsto u$$

$$\Longrightarrow$$

$$t^* \mapsto u^*$$

Properties

Theorem (Subject Reduction)

If $t \mapsto t'$ and $t : A$, then $t' : A$. Moreover, the free variables of t' are among those of t.

Properties

Theorem (Subject Reduction)

If $t \mapsto t'$ and $t : A$, then $t' : A$. Moreover, the free variables of t' are among those of t.

Computation as Elimination of Detours

$$\dfrac{\dfrac{\vdots}{u : A}}{\Lambda X\, u : \forall X\, A}\qquad\Longrightarrow\qquad \dfrac{\vdots}{(\Lambda X\, u)B : A[B/X]}$$

Properties

Theorem (Subject Reduction)

If $t \mapsto t'$ and $t : A$, then $t' : A$. Moreover, the free variables of t' are among those of t.

Computation as Elimination of Detours

$$\dfrac{\dfrac{\vdots}{u : A}}{\Lambda X\, u : \forall X\, A}}{(\Lambda X\, u)B : A[B/X]}$$

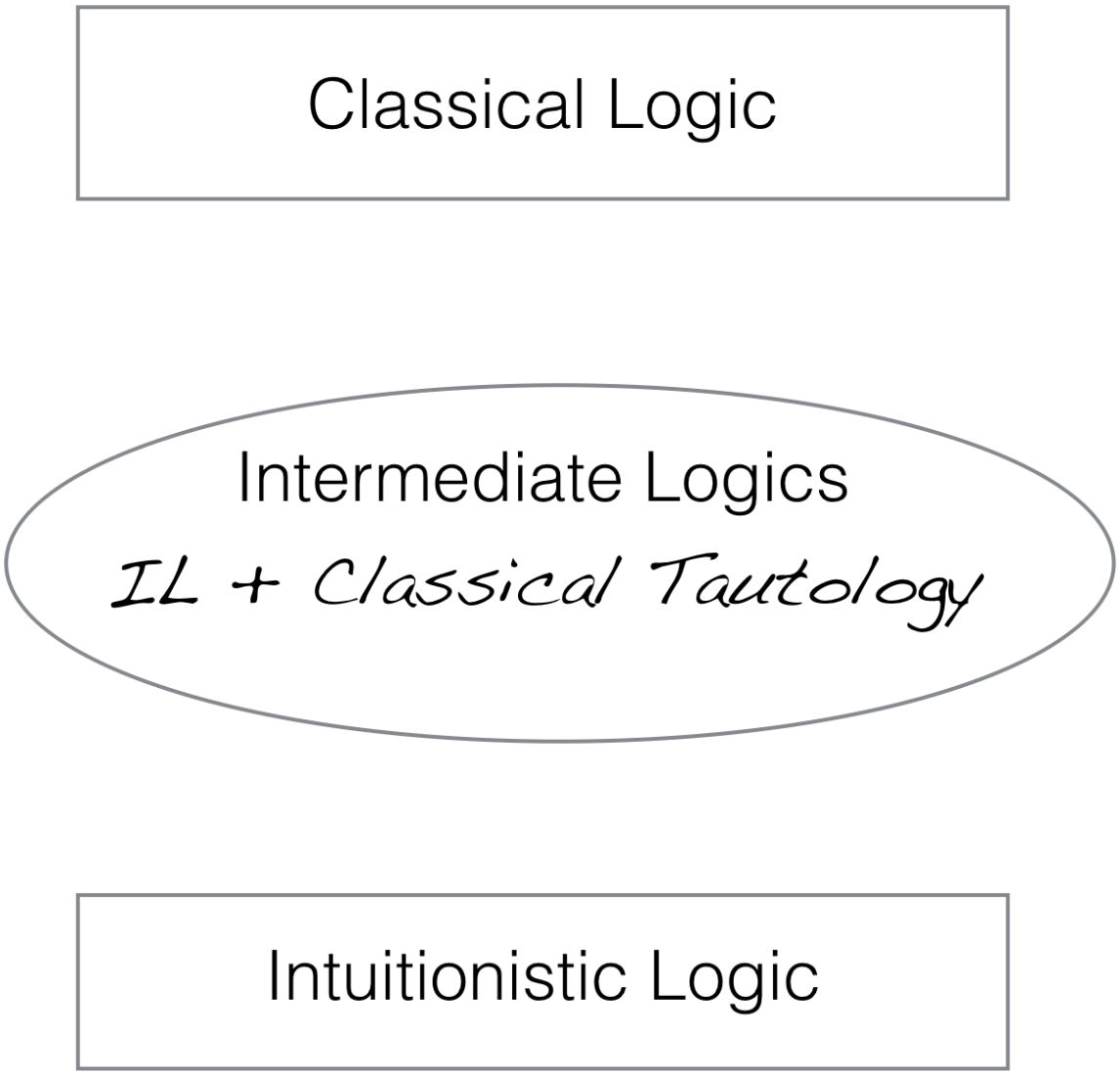
$$u[B/X] : A[B/X]$$

Strong Normalization

Theorem (Girard, 1971)

Every term of the polymorphic λ-calculus is strongly normalizable.

Intermediate Logics

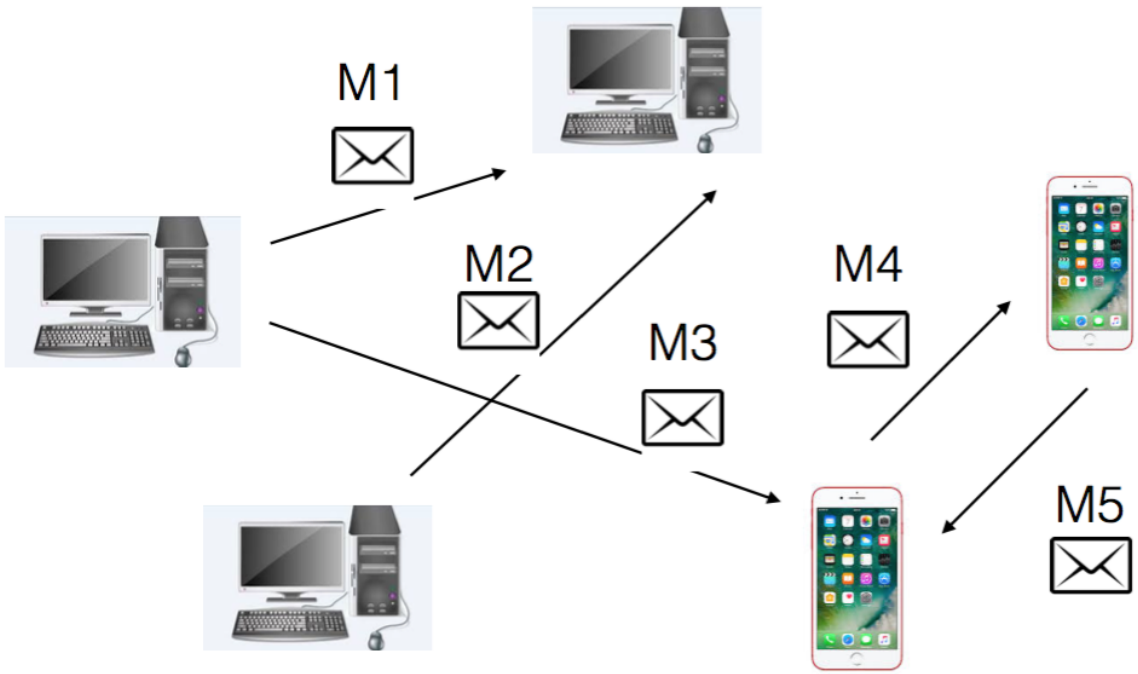


Intermediate Logics

$\sim$

Intuitionistic Logic on Steroids

Concurrency



CLASSICAL AND GÖDEL LOGIC

(Griffin 1991, Gödel 1933)

Classical λ-calculi

λ_C calculus: Griffin 1991, Krivine 2010–...

Classical λ-calculi

λ_C calculus: Griffin 1991, Krivine 2010–...

$\lambda\mu$ calculus: Parigot 1992

Classical λ-calculi

λ_C calculus: Griffin 1991, Krivine 2010–...

$\lambda\mu$ calculus: Parigot 1992

$\overline{\lambda}\mu\widetilde{\mu}$: Curien and Herbelin 2010

Classical Logic

Classical Logic: Intuitionistic Propositional Logic with Excluded Middle

$$\neg A \vee A$$

Classical Logic

Classical Logic: Intuitionistic Propositional Logic with Excluded Middle

$$\neg A \vee A$$

$$\neg A := A \to \bot$$

Ex-Falso

$$\frac{t : \bot}{t\,\mathsf{efq}_P : P}$$

Excluded Middle

$$\dfrac{\begin{array}{cc}[a^{A\to\bot} : A \to \bot] & [a^{A} : A] \\ \vdots & \vdots \\ u : C & v : C\end{array}}{u \parallel_a v : C}$$

3 Principles

1. Unidirectional message passing, exception

3 Principles

1. Unidirectional message passing, exception

2. If a process can communicate with several processes, it *will*

3 Principles

1. Unidirectional message passing, exception
2. If a process can communicate with several processes, it *will*
3. Messages can be open processes

Principle 1: Message Passing, Exception

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D} \qquad \mapsto \qquad \mathcal{D}[u/a]$$

Principle 1: Message Passing, Exception

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D} \qquad \mapsto \qquad \mathcal{D}[u/a]$$

the displayed occurrence of a is the rightmost in $\mathcal{C}$ and u is closed

Principle 1: Message Passing, Exception

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D} \qquad \mapsto \qquad \mathcal{D}[u/a]$$

the displayed occurrence of a is the rightmost in $\mathcal{C}$ and u is closed

$$u \parallel_a v \mapsto u, \text{ if } a \text{ does not occur in } u$$

$$u \parallel_a v \mapsto v, \text{ if } a \text{ does not occur in } v$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} O u T \mapsto^* T \\ O T u \mapsto^* T \\ OFF \mapsto^* F \end{cases}$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} O u \mathsf{T} \mapsto^* \mathsf{T} \\ O \mathsf{T} u \mapsto^* \mathsf{T} \\ O \mathsf{F}\mathsf{F} \mapsto^* \mathsf{F} \end{cases}$$

Add type `Bool`, booleans and if-then-else.

$$O :=$$

$$\lambda x^{\mathsf{Bool}}\, \lambda y^{\mathsf{Bool}}\, \text{if } x \text{ then } \mathsf{T} \text{ else } (a^{\mathit{Bool}\to\bot}\, x\, \mathsf{efq}_{\mathsf{Bool}})\ \|_a\ (\text{if } y \text{ then } \mathsf{T} \text{ else } a^{\mathsf{Bool}})$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} \mathrm{O}u\mathrm{T} & \mapsto^* & \mathrm{T} \\ \mathrm{OT}u & \mapsto^* & \mathrm{T} \\ \mathrm{OFF} & \mapsto^* & \mathrm{F} \end{cases}$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} \mathsf{O}u\mathsf{T} & \mapsto^* \mathsf{T} \\ \mathsf{OT}u & \mapsto^* \mathsf{T} \\ \mathsf{OFF} & \mapsto^* \mathsf{F} \end{cases}$$

$$\mathsf{OT}u := \mathsf{if}\,\mathsf{T}\,\mathsf{then}\,\mathsf{T}\,\mathsf{else}\,(a^{Bool\to\bot}\,\mathsf{T}\,\mathsf{efq}_{\mathsf{Bool}})\parallel_a(\mathsf{if}\,u\,\mathsf{then}\,\mathsf{T}\,\mathsf{else}\,a^{\mathsf{Bool}})$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} O u \mathsf{T} \mapsto^* \mathsf{T} \\ O \mathsf{T} u \mapsto^* \mathsf{T} \\ O \mathsf{F}\mathsf{F} \mapsto^* \mathsf{F} \end{cases}$$

$$O \mathsf{T} u := \text{if } \mathsf{T} \text{ then } \mathsf{T} \text{ else } (a^{Bool \to \bot} \mathsf{T}\, \text{efq}_{\text{Bool}})\ \|_a\ (\text{if } u \text{ then } \mathsf{T} \text{ else } a^{\text{Bool}})$$

$$\mapsto$$

$$\mathsf{T}\ \|_a\ (\text{if } u \text{ then } \mathsf{T} \text{ else } a^{\text{Bool}})$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} \mathsf{O}u\mathsf{T} & \mapsto^* \mathsf{T} \\ \mathsf{OT}u & \mapsto^* \mathsf{T} \\ \mathsf{OFF} & \mapsto^* \mathsf{F} \end{cases}$$

$$\mathsf{OT}u := \mathsf{if}\,\mathsf{T}\,\mathsf{then}\,\mathsf{T}\,\mathsf{else}\,(a^{Bool\to\bot}\,\mathsf{T}\,\mathsf{efq}_{\mathsf{Bool}})\parallel_a(\mathsf{if}\,u\,\mathsf{then}\,\mathsf{T}\,\mathsf{else}\,a^{\mathsf{Bool}})$$

$$\mapsto$$

$$\mathsf{T}\parallel_a(\mathsf{if}\,u\,\mathsf{then}\,\mathsf{T}\,\mathsf{else}\,a^{\mathsf{Bool}})$$

$$\mapsto$$

$$\mathsf{T}$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} \mathsf{O}u\mathsf{T} \mapsto^* \mathsf{T} \\ \mathsf{OT}u \mapsto^* \mathsf{T} \\ \mathsf{OFF} \mapsto^* \mathsf{F} \end{cases}$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} \mathsf{O}u\mathsf{T} & \mapsto^* & \mathsf{T} \\ \mathsf{OT}u & \mapsto^* & \mathsf{T} \\ \mathsf{OFF} & \mapsto^* & \mathsf{F} \end{cases}$$

$$\mathsf{OFF} := \mathsf{if\ F\ then\ T\ else}\ (a^{Bool \to \bot}\ \mathsf{F}\ \mathsf{efq}_{\mathsf{Bool}})\ \|_a\ (\mathsf{if\ F\ then\ T\ else}\ a^{\mathsf{Bool}})$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} O u T \mapsto^* T \\ O T u \mapsto^* T \\ O F F \mapsto^* F \end{cases}$$

$$\mathsf{OFF} := \mathsf{if\ F\ then\ T\ else}\ (a^{Bool \to \bot}\ \mathsf{F}\ \mathsf{efq}_{\mathsf{Bool}})\ \|_a\ (\mathsf{if\ F\ then\ T\ else}\ a^{\mathsf{Bool}})$$

$$\mapsto$$

$$a^{Bool \to \bot}\ \mathsf{F}\ \mathsf{efq}_{\mathsf{Bool}}\ \|_a\ a^{\mathsf{Bool}}$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} \mathsf{O}u\mathsf{T} & \mapsto^* \mathsf{T} \\ \mathsf{OT}u & \mapsto^* \mathsf{T} \\ \mathsf{OFF} & \mapsto^* \mathsf{F} \end{cases}$$

$$\mathsf{OFF} := \mathsf{if\ F\ then\ T\ else}\ (a^{Bool \to \bot}\ \mathsf{F}\ \mathsf{efq}_{\mathsf{Bool}})\ \|_a\ (\mathsf{if\ F\ then\ T\ else}\ a^{\mathsf{Bool}})$$

$$\mapsto$$

$$a^{Bool \to \bot}\ \mathsf{F}\ \mathsf{efq}_{\mathsf{Bool}}\ \|_a\ a^{\mathsf{Bool}}$$

$$\mapsto$$

$$a^{Bool \to \bot}\ \mathsf{F}\ \mathsf{efq}_{\mathsf{Bool}}\ \|_a\ \mathsf{F}$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} O u T \mapsto^* T \\ O T u \mapsto^* T \\ O F F \mapsto^* F \end{cases}$$

$$\mathsf{OFF} := \mathsf{if\ F\ then\ T\ else}\ (a^{Bool \to \bot}\ \mathsf{F}\ \mathsf{efq}_{\mathsf{Bool}})\ \|_a\ (\mathsf{if\ F\ then\ T\ else}\ a^{\mathsf{Bool}})$$

$$\mapsto$$

$$a^{Bool \to \bot}\ \mathsf{F}\ \mathsf{efq}_{\mathsf{Bool}}\ \|_a\ a^{\mathsf{Bool}}$$

$$\mapsto$$

$$a^{Bool \to \bot}\ \mathsf{F}\ \mathsf{efq}_{\mathsf{Bool}}\ \|_a\ \mathsf{F}$$

$$\mapsto$$

$$\mathsf{F}$$

Principle 2: Permutations and Scope Extrusion

$$w \parallel_b (u \parallel_a v) \mapsto (w \parallel_b u) \parallel_a (w \parallel_b v)$$

Principle 2: Permutations and Scope Extrusion

$$w \parallel_b (u \parallel_a v) \mapsto (w \parallel_b u) \parallel_a (w \parallel_b v)$$

$$(u \parallel_a v) \parallel_b w \mapsto (u \parallel_b w) \parallel_a (v \parallel_b w)$$

Principle 2: Permutations and Scope Extrusion

$$w \parallel_b (u \parallel_a v) \mapsto (w \parallel_b u) \parallel_a (w \parallel_b v)$$

$$(u \parallel_a v) \parallel_b w \mapsto (u \parallel_b w) \parallel_a (v \parallel_b w)$$

SCOPE EXTRUSION

$$(v \parallel_a \mathcal{C}[b\, a]) \parallel_b w$$

Principle 2: Permutations and Scope Extrusion

$$w \parallel_b (u \parallel_a v) \mapsto (w \parallel_b u) \parallel_a (w \parallel_b v)$$

$$(u \parallel_a v) \parallel_b w \mapsto (u \parallel_b w) \parallel_a (v \parallel_b w)$$

SCOPE EXTRUSION

$$(v \parallel_a \mathcal{C}[b\,a]) \parallel_b w$$

$$\nu a\,(v \mid \mathcal{C}[b\,a]) \mid w \equiv \nu a\,(v \mid \mathcal{C}[b\,a] \mid w)$$

Principle 2: Permutations and Scope Extrusion

$$w \parallel_b (u \parallel_a v) \mapsto (w \parallel_b u) \parallel_a (w \parallel_b v)$$

$$(u \parallel_a v) \parallel_b w \mapsto (u \parallel_b w) \parallel_a (v \parallel_b w)$$

SCOPE EXTRUSION

$$(v \parallel_a \mathcal{C}[b\,a]) \parallel_b w$$

$$\nu a\,(v \mid \mathcal{C}[b\,a]) \mid w \equiv \nu a\,(v \mid \mathcal{C}[b\,a] \mid w)$$

$$(v \parallel_a \mathcal{C}[b\,a]) \parallel_b w \mapsto (v \parallel_b w) \parallel_a (\mathcal{C}[b\,a] \parallel_b w)$$

Principle 3: Cross Reductions

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}$$

Principle 3: Cross Reductions

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}$$

$\boldsymbol{z}$ is the sequence of the free variables of u which are bound in $\mathcal{C}[a\,u]$;

Principle 3: Cross Reductions

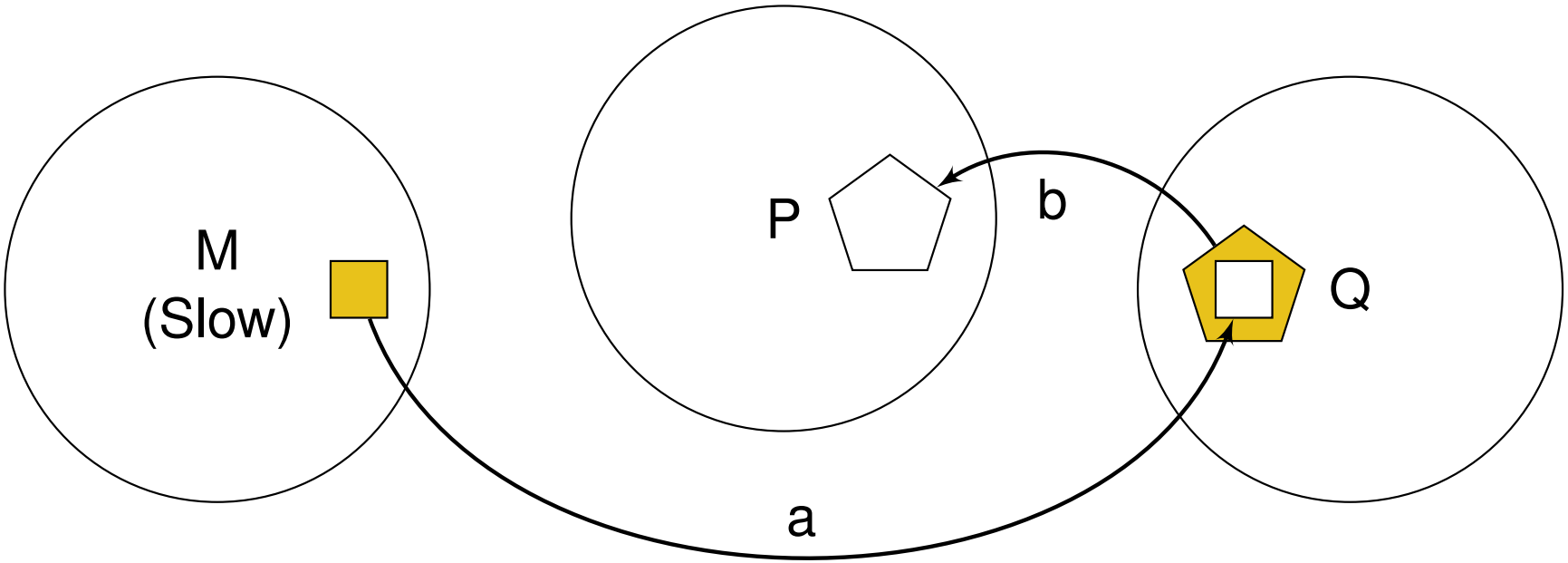
$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}$$

$\boldsymbol{z}$ is the sequence of the free variables of u which are bound in $\mathcal{C}[a\,u]$;

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D} \;\mapsto\; (\mathcal{C}[b\,\boldsymbol{z}] \parallel_a \mathcal{D}) \parallel_b \mathcal{D}[u^{b/\boldsymbol{z}}/a]$$

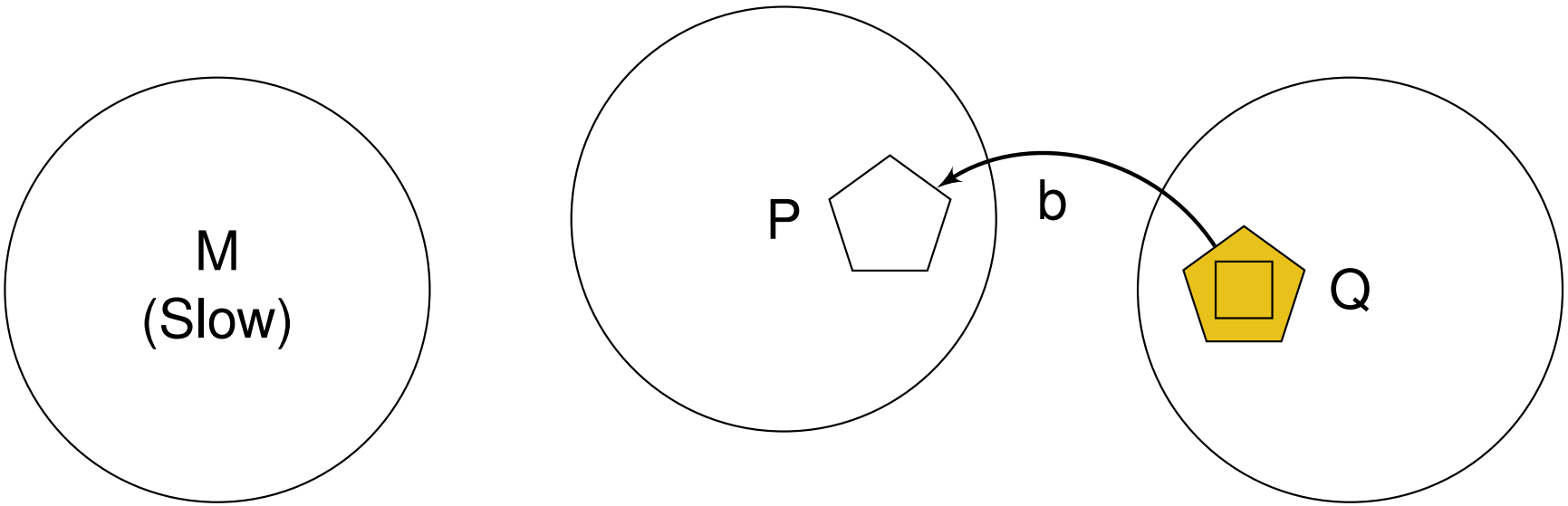
Optimisation via code mobility: the full power

Optimisation via code mobility: the full power



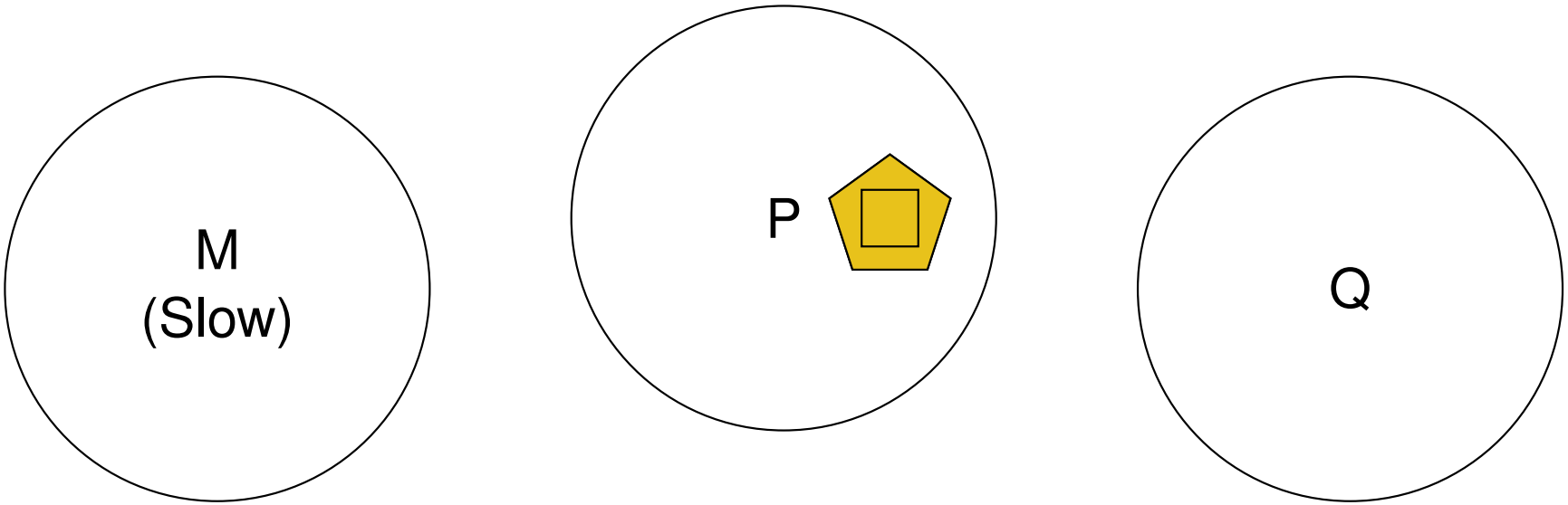
Without input redirection

Optimisation via code mobility: the full power



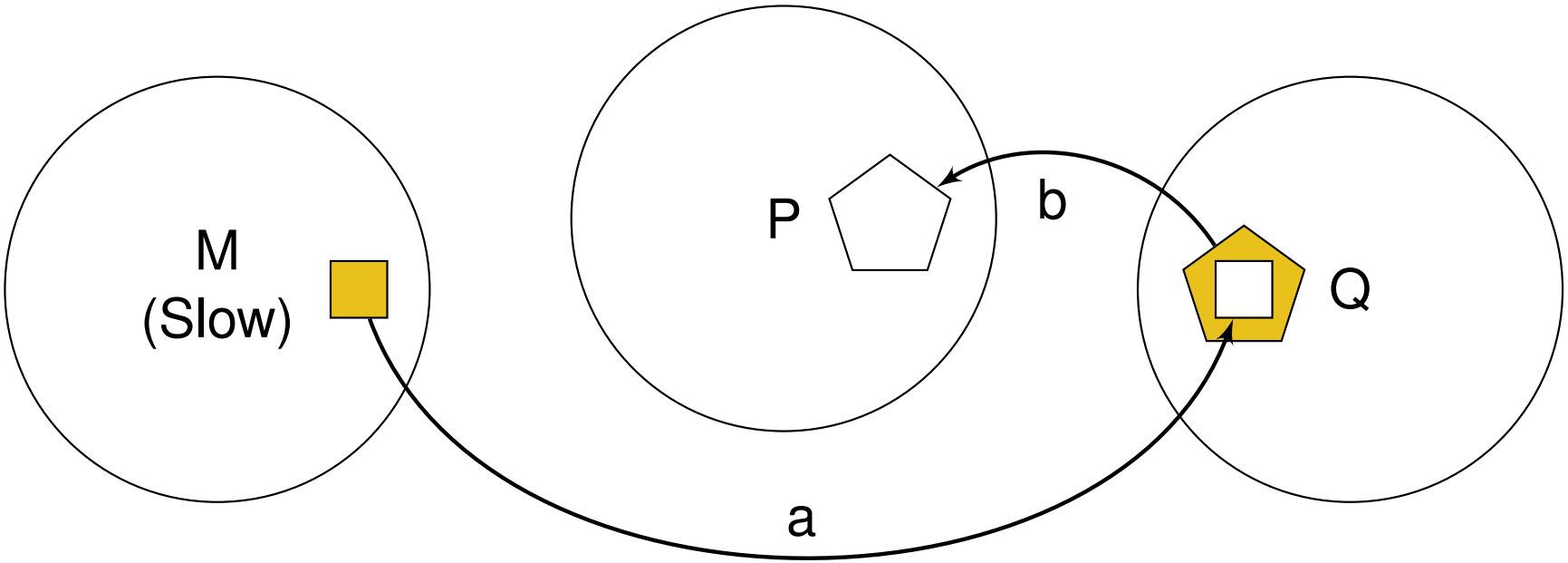
Without input redirection

Optimisation via code mobility: the full power



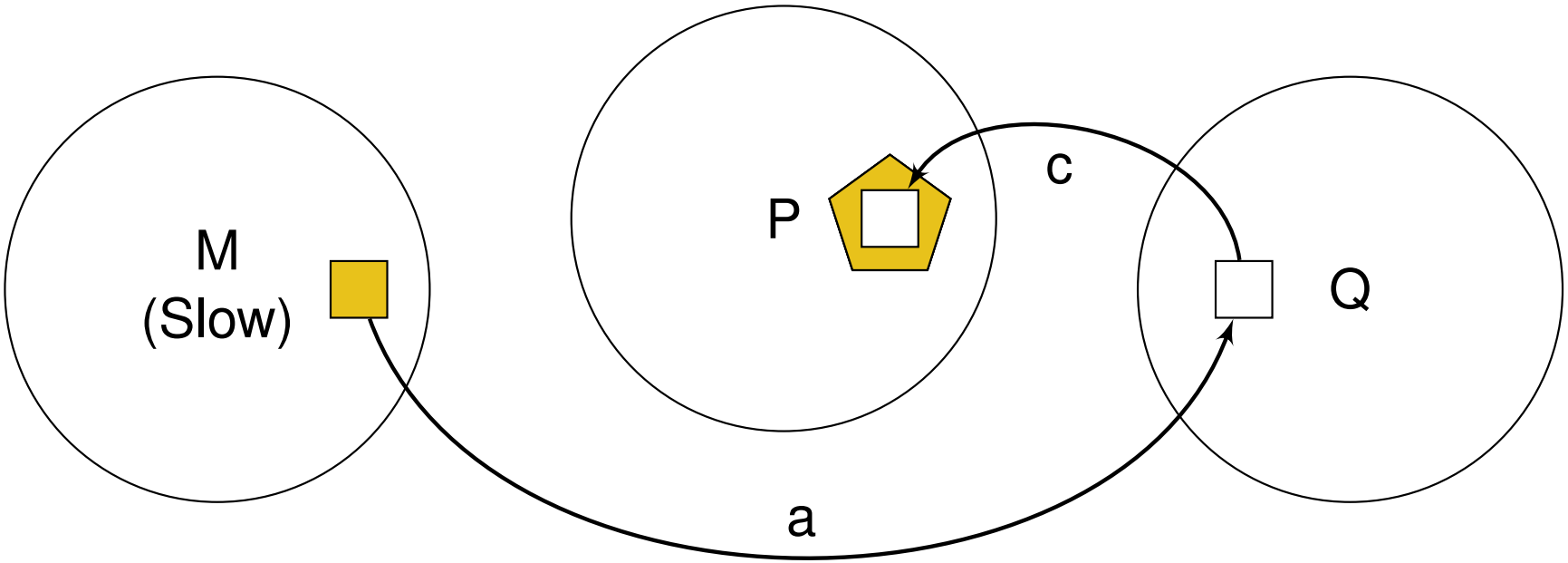
Without input redirection

Optimisation via code mobility: the full power



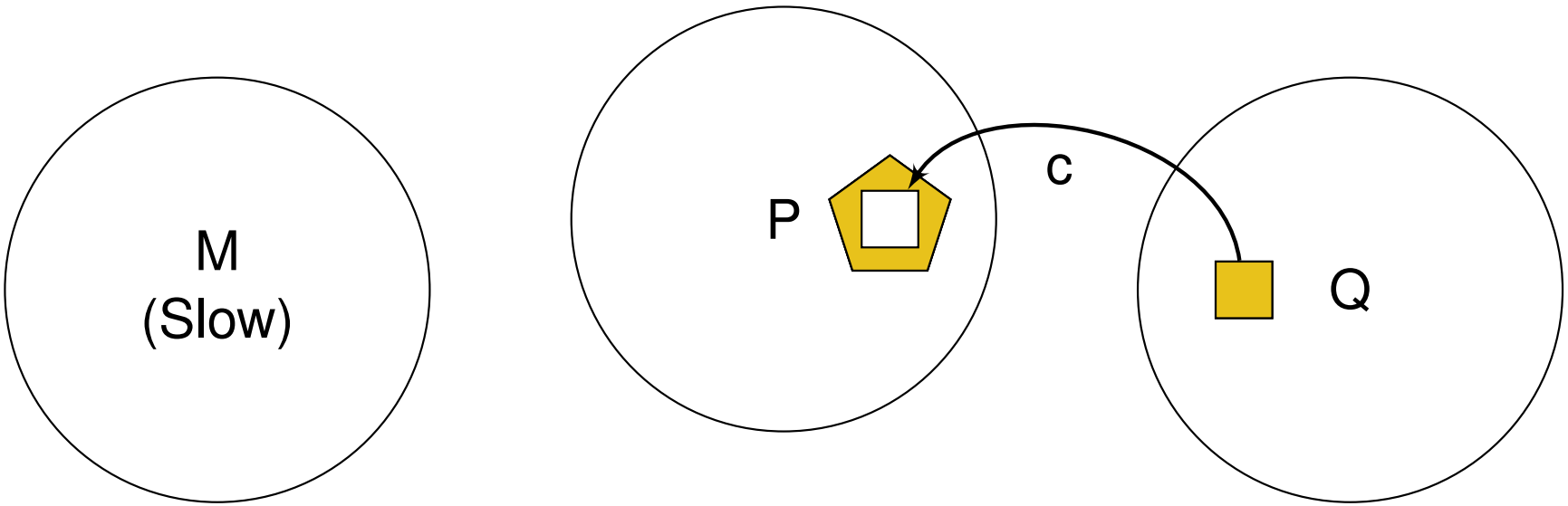
With input redirection

Optimisation via code mobility: the full power



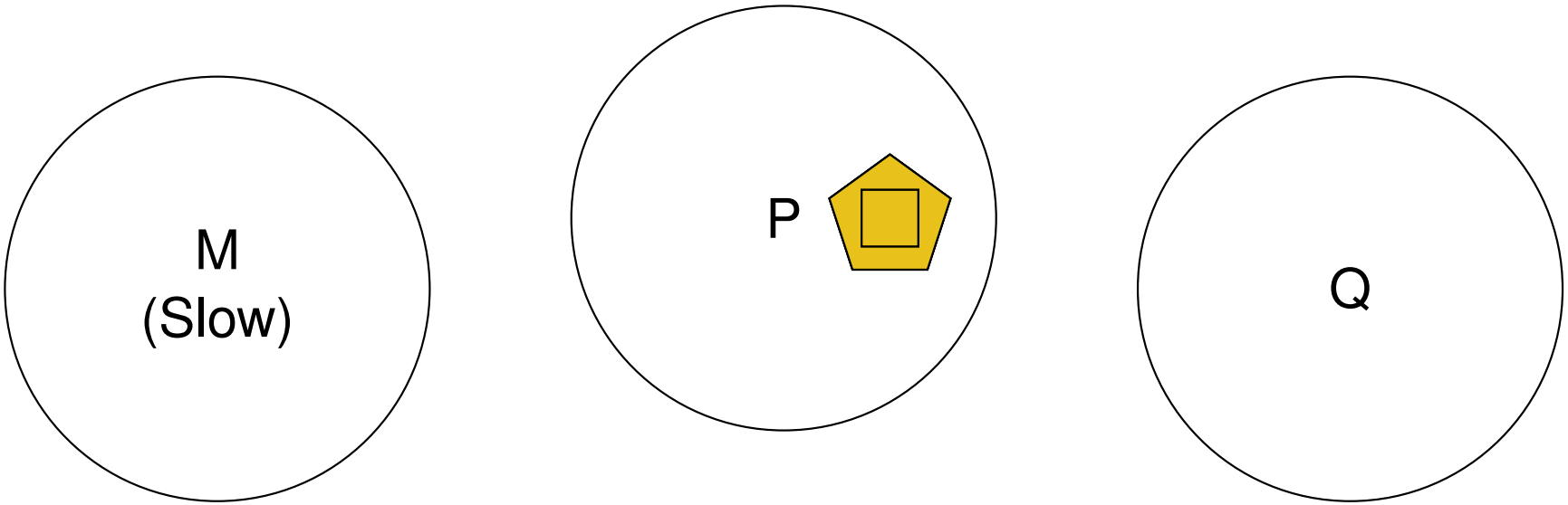
With input redirection

Optimisation via code mobility: the full power



With input redirection

Optimisation via code mobility: the full power



With input redirection

Parallel Form

$$t = t_1 \parallel_{a_1} t_2 \parallel_{a_2} \cdots \parallel_{a_n} t_{n+1}$$

where each t_i is a simply typed λ-term.

PARALLEL FORM

Parallel Form

$$t = t_1 \parallel_{a_1} t_2 \parallel_{a_2} \dots \parallel_{a_n} t_{n+1}$$

where each t_i is a simply typed λ-term.

PARALLEL FORM

$$\lambda x\,(u \parallel_a v) \mapsto \lambda x\,u \parallel_a \lambda x\,v$$

$$\langle u \parallel_a v,\ w\rangle \mapsto \langle u, w\rangle \parallel_a \langle v, w\rangle, \text{ if } a \text{ does not occur free in } w$$

$$\langle w,\ u \parallel_a v\rangle \mapsto \langle w, u\rangle \parallel_a \langle w, v\rangle, \text{ if } a \text{ does not occur free in } w$$

Normalization and Subformula Property

Theorem

*Every proof term reduces to a **normal** proof term, satisfying the **Subformula Property**.*

Gödel Logic

Gödel's G: Intuitionistic Propositional Logic with Dummett's Axiom

$$A \to B \quad \vee \quad B \to A$$

Curry-Howard Correspondence for Gödel Logic

$$\lambda_G$$

Dummett's Axiom

$$\dfrac{\begin{array}{cc}[a^{A\to B} : A \to B] & [a^{B\to A} : B \to A] \\ \vdots & \vdots \\ u : C & v : C\end{array}}{u \parallel_a v : C}$$

3 Principles

1. Non-deterministic message passing

3 Principles

1. Non-deterministic message passing

2. If a process can communicate with several processes, it *will*

3 Principles

1. Non-deterministic message passing
2. If a process can communicate with several processes, it *will*
3. Messages can be open processes

Principle 1: Non-deterministic Message Passing

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v] \qquad \mapsto \qquad \mathcal{C}[v] \parallel_a \mathcal{D}[a\,v]$$

Principle 1: Non-deterministic Message Passing

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v] \quad \mapsto \quad \mathcal{C}[v] \parallel_a \mathcal{D}[a\,v]$$

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v] \quad \mapsto \quad \mathcal{D}[u] \parallel_a \mathcal{C}[a\,u]$$

Principle 1: Non-deterministic Message Passing

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v] \quad \mapsto \quad \mathcal{C}[v] \parallel_a \mathcal{D}[a\,v]$$

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v] \quad \mapsto \quad \mathcal{D}[u] \parallel_a \mathcal{C}[a\,u]$$

the displayed occurrence of a is the rightmost in $\mathcal{C}$ and $\mathcal{D}$

Principle 1: Non-deterministic Message Passing

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v] \quad \mapsto \quad \mathcal{C}[v] \parallel_a \mathcal{D}[a\,v]$$

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v] \quad \mapsto \quad \mathcal{D}[u] \parallel_a \mathcal{C}[a\,u]$$

the displayed occurrence of a is the rightmost in $\mathcal{C}$ and $\mathcal{D}$

$$u \parallel_a v \mapsto u, \text{ if } a \text{ does not occur in } u$$

$$u \parallel_a v \mapsto v, \text{ if } a \text{ does not occur in } v$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} \mathsf{O}u\mathsf{T} \mapsto^* \mathsf{T} \\ \mathsf{OT}u \mapsto^* \mathsf{T} \\ \mathsf{OFF} \mapsto^* \mathsf{F} \end{cases}$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} \mathsf{O}u\mathsf{T} \mapsto^* \mathsf{T} \\ \mathsf{OT}u \mapsto^* \mathsf{T} \\ \mathsf{OFF} \mapsto^* \mathsf{F} \end{cases}$$

$$\mathsf{O} := \lambda x^{\mathsf{Bool}}\, \lambda y^{\mathsf{Bool}} (\mathsf{if}\, x\, \mathsf{then}\, (\lambda z\, \lambda k\, z)\, \mathsf{else}\, (\lambda z\, \lambda k\, k))\mathsf{T}(ax)$$

$$\|_a\, (\mathsf{if}\, y\, \mathsf{then}\, (\lambda z\, \lambda k\, z)\, \mathsf{else}\, (\lambda z\, \lambda k\, k))\mathsf{T}(ay)$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} \mathsf{O}u\mathsf{T} \mapsto^* \mathsf{T} \\ \mathsf{OT}u \mapsto^* \mathsf{T} \\ \mathsf{OFF} \mapsto^* \mathsf{F} \end{cases}$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} O u T \mapsto^* T \\ O T u \mapsto^* T \\ O F F \mapsto^* F \end{cases}$$

$$O u T := (\text{if } u \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k)) T (a u)$$

$$\|_a \; (\text{if } T \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k)) T (a T)$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} O u \mathsf{T} \mapsto^* \mathsf{T} \\ O \mathsf{T} u \mapsto^* \mathsf{T} \\ O \mathsf{F}\mathsf{F} \mapsto^* \mathsf{F} \end{cases}$$

$$O u \mathsf{T} := (\text{if } u \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k))\mathsf{T}(a u)$$

$$\|_a\ (\text{if } \mathsf{T} \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k))\mathsf{T}(a \mathsf{T})$$

$$\mapsto$$

$$(\text{if } u \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k))\mathsf{T}(a x)\ \|_a\ \mathsf{T}$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} O u T \mapsto^* T \\ O T u \mapsto^* T \\ O F F \mapsto^* F \end{cases}$$

$$O u T := (\text{if } u \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k)) T (a u)$$

$$\|_a \, (\text{if } T \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k)) T (a T)$$

$$\mapsto$$

$$(\text{if } u \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k)) T (a x) \, \|_a \, T$$

$$\mapsto$$

$$T$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} \mathsf{O}u\mathsf{T} \mapsto^* \mathsf{T} \\ \mathsf{OT}u \mapsto^* \mathsf{T} \\ \mathsf{OFF} \mapsto^* \mathsf{F} \end{cases}$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} O u T \mapsto^* T \\ O T u \mapsto^* T \\ O F F \mapsto^* F \end{cases}$$

$$\mathsf{OFF} := (\text{if } \mathsf{F} \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k))\mathsf{T}(a\,\mathsf{F})$$

$$\|_a\ (\text{if } \mathsf{F} \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k))\mathsf{T}(a\,\mathsf{F})$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} O u \mathsf{T} \mapsto^* \mathsf{T} \\ O \mathsf{T} u \mapsto^* \mathsf{T} \\ O \mathsf{F} \mathsf{F} \mapsto^* \mathsf{F} \end{cases}$$

$$\mathsf{OFF} := (\text{if } \mathsf{F} \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k))\mathsf{T}(a\,\mathsf{F})$$

$$\|_a\ (\text{if } \mathsf{F} \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k))\mathsf{T}(a\,\mathsf{F})$$

$$\mapsto$$

$$a\,\mathsf{F}\ \|_a\ a\,\mathsf{F}$$

Parallel disjunction

$$\text{Term O s.t.} \begin{cases} O u \mathsf{T} \mapsto^* \mathsf{T} \\ O \mathsf{T} u \mapsto^* \mathsf{T} \\ O \mathsf{F}\mathsf{F} \mapsto^* \mathsf{F} \end{cases}$$

$$\mathsf{OFF} := (\text{if } \mathsf{F} \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k))\mathsf{T}(a\,\mathsf{F})$$

$$\|_a\ (\text{if } \mathsf{F} \text{ then } (\lambda z\, \lambda k\, z) \text{ else } (\lambda z\, \lambda k\, k))\mathsf{T}(a\,\mathsf{F})$$

$$\mapsto$$

$$a\,\mathsf{F}\ \|_a\ a\,\mathsf{F}$$

$$\mapsto$$

$$\mathsf{F}$$

Buyer-Vendor

$$\mathcal{B} \xrightarrow{\textit{product name}} \mathcal{V}$$

$$\mathcal{B} \xleftarrow{\textit{price}} \mathcal{V}$$

$$\mathcal{B} \xrightarrow{\textit{credit card n.}} \mathcal{V}$$

Buyer-Vendor

$$\mathcal{B} \xrightarrow{\textit{product name}} \mathcal{V}$$

$$\mathcal{B} \xleftarrow{\textit{price}} \mathcal{V}$$

$$\mathcal{B} \xrightarrow{\textit{credit card n.}} \mathcal{V}$$

$$\mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \parallel_a \mathcal{V}[\text{use}(a(\text{cost}(a\,0)))] \mapsto$$

Buyer-Vendor

$$\mathcal{B} \xrightarrow{\textit{product name}} \mathcal{V}$$

$$\mathcal{B} \xleftarrow{\textit{price}} \mathcal{V}$$

$$\mathcal{B} \xrightarrow{\textit{credit card n.}} \mathcal{V}$$

$$\mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \parallel_a \mathcal{V}[\text{use}(a(\text{cost}(a\,0)))] \mapsto$$

$$\mathcal{V}[\text{use}(a(\text{cost}(\text{prod})))] \parallel_a \mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \mapsto$$

Buyer-Vendor

$$\mathcal{B} \xrightarrow{\textit{product name}} \mathcal{V}$$

$$\mathcal{B} \xleftarrow{\textit{price}} \mathcal{V}$$

$$\mathcal{B} \xrightarrow{\textit{credit card n.}} \mathcal{V}$$

$$\mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \parallel_a \mathcal{V}[\text{use}(a(\text{cost}(a\,0)))] \mapsto$$

$$\mathcal{V}[\text{use}(a(\text{cost}(\text{prod})))] \parallel_a \mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \mapsto$$

$$\mathcal{V}[\text{use}(a(\text{price}))] \parallel_a \mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \mapsto$$

Buyer-Vendor

$$\mathcal{B} \xrightarrow{\textit{product name}} \mathcal{V}$$

$$\mathcal{B} \xleftarrow{\textit{price}} \mathcal{V}$$

$$\mathcal{B} \xrightarrow{\textit{credit card n.}} \mathcal{V}$$

$$\mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \parallel_a \mathcal{V}[\text{use}(a(\text{cost}(a\,0)))] \mapsto$$

$$\mathcal{V}[\text{use}(a(\text{cost}(\text{prod})))] \parallel_a \mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \mapsto$$

$$\mathcal{V}[\text{use}(a(\text{price}))] \parallel_a \mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \mapsto$$

$$\mathcal{B}[a(\text{pay\_for}(\text{price}))] \parallel_a \mathcal{V}[\text{use}(a(\text{price}))] \mapsto$$

Buyer-Vendor

$$\mathcal{B} \xrightarrow{\textit{product name}} \mathcal{V}$$

$$\mathcal{B} \xleftarrow{\textit{price}} \mathcal{V}$$

$$\mathcal{B} \xrightarrow{\textit{credit card n.}} \mathcal{V}$$

$$\mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \parallel_a \mathcal{V}[\text{use}(a(\text{cost}(a\,0)))] \mapsto$$

$$\mathcal{V}[\text{use}(a(\text{cost}(\text{prod})))] \parallel_a \mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \mapsto$$

$$\mathcal{V}[\text{use}(a(\text{price}))] \parallel_a \mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \mapsto$$

$$\mathcal{B}[a(\text{pay\_for}(\text{price}))] \parallel_a \mathcal{V}[\text{use}(a(\text{price}))] \mapsto$$

$$\mathcal{B}[a(\text{card})] \parallel_a \mathcal{V}[\text{use}(a(\text{price}))] \mapsto^*$$

Buyer-Vendor

$$\mathcal{B} \xrightarrow{\textit{product name}} \mathcal{V}$$

$$\mathcal{B} \xleftarrow{\textit{price}} \mathcal{V}$$

$$\mathcal{B} \xrightarrow{\textit{credit card n.}} \mathcal{V}$$

$$\mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \parallel_a \mathcal{V}[\text{use}(a(\text{cost}(a\,0)))] \mapsto$$

$$\mathcal{V}[\text{use}(a(\text{cost}(\text{prod})))] \parallel_a \mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \mapsto$$

$$\mathcal{V}[\text{use}(a(\text{price}))] \parallel_a \mathcal{B}[a(\text{pay\_for}(a(\text{prod})))] \mapsto$$

$$\mathcal{B}[a(\text{pay\_for}(\text{price}))] \parallel_a \mathcal{V}[\text{use}(a(\text{price}))] \mapsto$$

$$\mathcal{B}[a(\text{card})] \parallel_a \mathcal{V}[\text{use}(a(\text{price}))] \mapsto^*$$

$$\mathcal{V}[\text{use}(\text{card})] \parallel_a \mathcal{B}[a(\text{card})]$$

Principle 2: Permutations and Scope Extrusion

$$w \parallel_b (u \parallel_a v) \mapsto (w \parallel_b u) \parallel_a (w \parallel_b v)$$

Principle 2: Permutations and Scope Extrusion

$$w \parallel_b (u \parallel_a v) \mapsto (w \parallel_b u) \parallel_a (w \parallel_b v)$$

$$(u \parallel_a v) \parallel_b w \mapsto (u \parallel_b w) \parallel_a (v \parallel_b w)$$

Principle 2: Permutations and Scope Extrusion

$$w \parallel_b (u \parallel_a v) \mapsto (w \parallel_b u) \parallel_a (w \parallel_b v)$$

$$(u \parallel_a v) \parallel_b w \mapsto (u \parallel_b w) \parallel_a (v \parallel_b w)$$

SCOPE EXTRUSION

$$(v \parallel_a \mathcal{C}[b\, a]) \parallel_b w$$

Principle 2: Permutations and Scope Extrusion

$$w \parallel_b (u \parallel_a v) \mapsto (w \parallel_b u) \parallel_a (w \parallel_b v)$$

$$(u \parallel_a v) \parallel_b w \mapsto (u \parallel_b w) \parallel_a (v \parallel_b w)$$

SCOPE EXTRUSION

$$(v \parallel_a \mathcal{C}[b\, a]) \parallel_b w$$

$$(v \parallel_a \mathcal{C}[b\, a]) \parallel_b w \mapsto (v \parallel_b w) \parallel_a (\mathcal{C}[b\, a] \parallel_b w)$$

Principle 3: Cross Reductions

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v]$$

Principle 3: Cross Reductions

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v]$$

$\boldsymbol{y}$ is the sequence of the free variables of u which are bound in $\mathcal{C}[a\,u]$;

$\boldsymbol{z}$ is the sequence of the free variables of v which are bound in $\mathcal{D}[a\,v]$;

Principle 3: Cross Reductions

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v]$$

$\boldsymbol{y}$ is the sequence of the free variables of u which are bound in $\mathcal{C}[a\,u]$;

$\boldsymbol{z}$ is the sequence of the free variables of v which are bound in $\mathcal{D}[a\,v]$;

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v] \;\mapsto\; (\mathcal{D}[u^{b\langle \boldsymbol{z}\rangle/\boldsymbol{y}}] \parallel_a \mathcal{C}[a\,u]) \parallel_b (\mathcal{C}[v^{b\langle \boldsymbol{y}\rangle/\boldsymbol{z}}] \parallel_a \mathcal{D}[a\,v])$$

Code Mobility

$$\mathcal{C}[a\,u] \parallel_a \mathcal{D}[a\,v] \;\mapsto\; (\mathcal{D}[u^{b\langle \boldsymbol{z}\rangle/\boldsymbol{y}}] \parallel_a \mathcal{C}[a\,u]) \parallel_b (\mathcal{C}[v^{b\langle \boldsymbol{y}\rangle/\boldsymbol{z}}] \parallel_a \mathcal{D}[a\,v])$$

CODE MOBILITY

Parallel Form

$$t = t_1 \parallel_{a_1} t_2 \parallel_{a_2} \cdots \parallel_{a_n} t_{n+1}$$

where each t_i is a simply typed λ-term.

PARALLEL FORM

Parallel Form

$$t = t_1 \parallel_{a_1} t_2 \parallel_{a_2} \dots \parallel_{a_n} t_{n+1}$$

where each t_i is a simply typed λ-term.

PARALLEL FORM

$$\lambda x\,(u \parallel_a v) \mapsto \lambda x\,u \parallel_a \lambda x\,v$$

$$\langle u \parallel_a v,\ w\rangle \mapsto \langle u, w\rangle \parallel_a \langle v, w\rangle, \text{ if } a \text{ does not occur free in } w$$

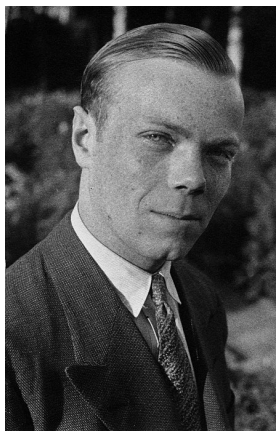
$$\langle w,\ u \parallel_a v\rangle \mapsto \langle w, u\rangle \parallel_a \langle w, v\rangle, \text{ if } a \text{ does not occur free in } w$$

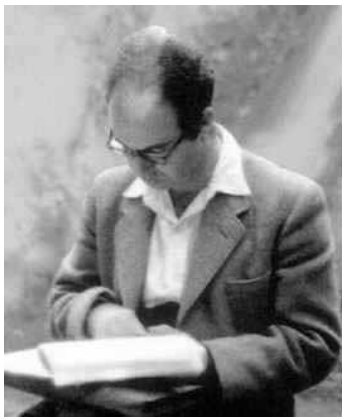
Normalization and Subformula Property

Theorem

*Every proof term reduces to a **normal** proof term, satisfying the **Subformula Property**.*

CLASSICAL FIRST-ORDER LOGIC AND ARITHMETIC





(Herbrand 1931, Kreisel 1952)

Herbrand's and Kreisel's Theorem

Theorem (Herbrand)

$$\mathsf{CL} \vdash \exists \alpha\, \mathsf{P} \implies \mathsf{PCL} \vdash \mathsf{P}[m_1/\alpha] \vee \cdots \vee \mathsf{P}[m_k/\alpha]$$

P quantifier-free

Theorem (Kreisel)

$$\mathsf{PA} \vdash \exists \alpha\, \mathsf{P} \implies \mathsf{HA} \vdash \exists \alpha \mathsf{P} \rightsquigarrow \mathsf{HA} \vdash P[n/\alpha]$$

P quantifier-free

Simply Typed λ-Terms

$$\frac{}{x^A : A}$$

Simply Typed λ-Terms

$$\frac{}{x^A : A}$$

$$\frac{\begin{array}{c} x^A : A \\ \vdots \\ u : B \end{array}}{\lambda x^A\, u : A \to B}$$

Simply Typed λ-Terms

$$\frac{}{x^A : A}$$

$$\frac{\begin{array}{c} x^A : A \\ \vdots \\ u : B \end{array}}{\lambda x^A\, u : A \to B}$$

$$\frac{t : A \to B \qquad u : A}{tu : B}$$

Conjunction

$$\frac{u : A \qquad v : B}{\langle u, v \rangle : A \wedge B}$$

Conjunction

$$\frac{u : A \qquad v : B}{\langle u, v \rangle : A \wedge B}$$

$$\frac{t : A \wedge B}{t\,\pi_0 : A}$$

$$\frac{t : A \wedge B}{t\,\pi_1 : B}$$

Conjunction

$$\frac{u : A \qquad v : B}{\langle u, v \rangle : A \wedge B}$$

$$\frac{t : A \wedge B}{t\,\pi_0 : A}$$

$$\frac{t : A \wedge B}{t\,\pi_1 : B}$$

$$\langle t_0, t_1 \rangle\,\pi_i \mapsto t_i \qquad (i \in \{0, 1\})$$

Disjunction

$$\frac{u : A}{\iota_0(u) : A \vee B}$$

$$\frac{u : B}{\iota_1(u) : A \vee B}$$

$$\frac{\begin{array}{ccc} & \begin{array}{c} x^A : A \\ \vdots \\ u : C \end{array} & \begin{array}{c} y^B : B \\ \vdots \\ v : C \end{array} \\ t : A \vee B & & \end{array}}{t\,[x^A.u, y^B.v] : C}$$

Disjunction

$$\frac{u : A}{\iota_0(u) : A \vee B}$$

$$\frac{u : B}{\iota_1(u) : A \vee B}$$

$$\frac{\begin{array}{ccc} & x^A : A & y^B : B \\ & \vdots & \vdots \\ t : A \vee B & u : C & v : C \end{array}}{t\,[x^A.u, y^B.v] : C}$$

$$\iota_0(u)[x.v, y.w] \mapsto v[u/x]$$

$$\iota_1(u)[x.v, y.w] \mapsto w[u/y]$$

Universal Quantifier

$$\frac{t : A}{\lambda\alpha\, t : \forall\alpha\, A}$$

provided α does not occur in the types of the free variables of t

Universal Quantifier

$$\frac{t : A}{\lambda\alpha\, t : \forall\alpha\, A}$$

provided α does not occur in the types of the free variables of t

$$\frac{t : \forall\alpha\, A}{t\, m : A[m/\alpha]}$$

Universal Quantifier

$$\frac{t : A}{\lambda\alpha\, t : \forall\alpha\, A}$$

provided α does not occur in the types of the free variables of t

$$\frac{t : \forall\alpha\, A}{t\, m : A[m/\alpha]}$$

$$(\lambda\alpha\, t)\, m \mapsto t[m/\alpha]$$

Existential Quantifier

$$\frac{t : A[m/\alpha]}{(m, t) : \exists \alpha\, A}$$

Existential Quantifier

$$\frac{t : A[m/\alpha]}{(m, t) : \exists\alpha\, A}$$

$$\frac{t : \exists\alpha\, A \qquad \begin{array}{c} x^A : A \\ \vdots \\ u : C \end{array}}{t\,[(\alpha, x^A).u] : C}$$

provided α does not occur in the types of the free variables of t differe

Existential Quantifier

$$\frac{t : A[m/\alpha]}{(m, t) : \exists \alpha\, A}$$

$$\frac{t : \exists \alpha\, A \qquad \begin{array}{c} x^A : A \\ \vdots \\ u : C \end{array}}{t\,[(\alpha, x^A).u] : C}$$

provided α does not occur in the types of the free variables of t differe

$$(m, t)[(\alpha, x).u] \mapsto u[m/\alpha][t/x]$$

Logical Interpretation

$$\dfrac{\dfrac{\pi}{A}}{\dfrac{\forall\alpha\, A}{A[m/\alpha]}}$$

converts to:

$$\begin{array}{c}\pi[m/\alpha] \\ \\ A[m/\alpha]\end{array}$$

Logical Interpretation

$$\dfrac{\dfrac{\vdots}{A[m/\alpha]} \quad \dfrac{[A]}{\pi}}{\dfrac{\exists\alpha\, A \qquad C}{C}}$$

converts to:

$$\begin{array}{c} \vdots \\ A[m/\alpha] \\ \pi[m/\alpha] \\ C \end{array}$$

Involutive Negation

$$P^{\perp} := \neg P$$

$$(\exists \alpha\, A)^{\perp} := \forall \alpha\, A^{\perp}$$

$$(\forall \alpha\, A)^{\perp} := \exists \alpha\, A^{\perp}$$

Excluded Middle

$$\dfrac{\begin{array}{cc}[H^{\neg P} : \neg P] & [H^{P} : P] \\ \vdots & \vdots \\ u : C & v : C\end{array}}{u \mid v : C}\ \mathrm{EM}_0$$

Excluded Middle

$$\dfrac{\begin{array}{cc}[a^{\forall\alpha A} : \forall\alpha A] & [W_a^{\exists\alpha A^\perp} : \exists\alpha A^\perp] \\ \vdots & \vdots \\ u : C & v : C\end{array}}{u \parallel_a v : C}\ \mathrm{EM}_n$$

$\forall\alpha A$ prenex formula with n alternating quantifiers

Reductions

$$(u \mid v)w \mapsto uw \mid vw$$

Reductions

$$(u \mid v)w \mapsto uw \mid vw$$

$$(u \mid v)\pi_i \mapsto u\pi_i \mid v\pi_i$$

Reductions

$$(u \mid v)w \mapsto uw \mid vw$$

$$(u \mid v)\pi_i \mapsto u\pi_i \mid v\pi_i$$

$$(u \mid v)[x.w_1, y.w_2] \mapsto u[x.w_1, y.w_2] \mid v[x.w_1, y.w_2]$$

Reductions

$$(u \mid v)w \mapsto uw \mid vw$$

$$(u \mid v)\pi_i \mapsto u\pi_i \mid v\pi_i$$

$$(u \mid v)[x.w_1, y.w_2] \mapsto u[x.w_1, y.w_2] \mid v[x.w_1, y.w_2]$$

$$(u \mid v)[(\alpha, x).w] \mapsto u[(\alpha, x).w] \mid v[(\alpha, x).w]$$

Reductions

$$(u \mid v)w \mapsto uw \mid vw$$

$$(u \mid v)\pi_i \mapsto u\pi_i \mid v\pi_i$$

$$(u \mid v)[x.w_1, y.w_2] \mapsto u[x.w_1, y.w_2] \mid v[x.w_1, y.w_2]$$

$$(u \mid v)[(\alpha, x).w] \mapsto u[(\alpha, x).w] \mid v[(\alpha, x).w]$$

$$(u \parallel_a v)w \mapsto uw \parallel_a vw$$

$$(u \parallel_a v)\pi_i \mapsto u\pi_i \parallel_a v\pi_i$$

$$(u \parallel_a v)[x.w_1, y.w_2] \mapsto u[x.w_1, y.w_2] \parallel_a v[x.w_1, y.w_2]$$

$$(u \parallel_a v)[(\alpha, x).w] \mapsto u[(\alpha, x).w] \parallel_a v[(\alpha, x).w]$$

Reductions

(EM):

$$u \parallel_a v$$

Reductions

(EM):

$$u \parallel_a v$$

$$a^{\forall \alpha P}$$

$$a^{\exists \alpha \neg P}$$

Reductions

(EM):

$$u \parallel_a v$$

$$a^{\forall \alpha P}$$

$$a^{\exists \alpha \neg P}$$

Raising Multiple Exceptions:

Reductions

(EM):

$$u \parallel_a v$$

$$a^{\forall \alpha P}$$

$$a^{\exists \alpha \neg P}$$

Raising Multiple Exceptions:

(if a does not occur free in u)

$$u \parallel_a v \mapsto u$$

Proof Terms

EM_1 (1 quantifier)
Raising Multiple Exceptions:

$$\mathcal{C}[\mathsf{a}^{\forall\alpha\mathsf{P}}\, m] \parallel_a v$$

$$\mapsto$$

$$v[(m, \mathsf{H}^{\neg\mathsf{P}[m/\alpha]})/a^{\exists\alpha\neg\mathsf{P}}] \mid (\mathcal{C}[\mathsf{H}^{\mathsf{P}[m/\alpha]}] \parallel_a v)$$

Proof Terms

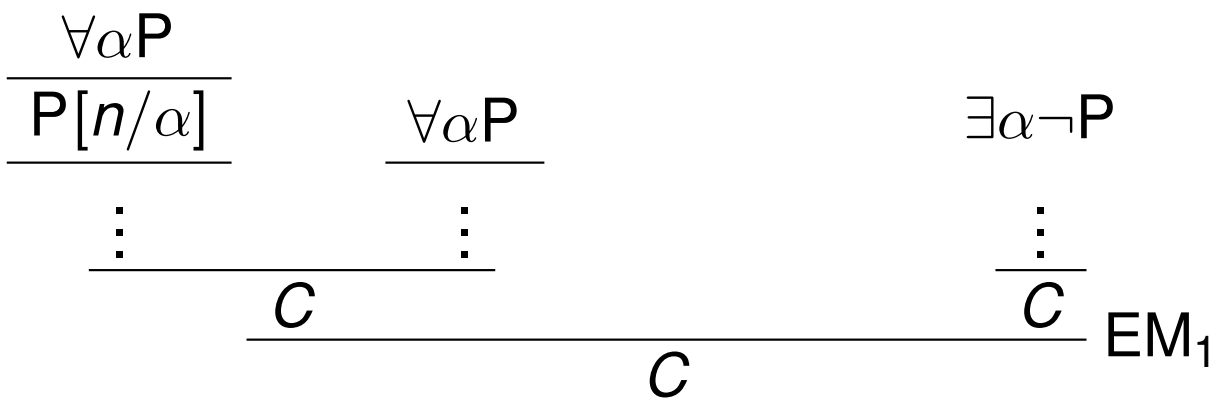
EM_n (n alternating quantifiers)
Raising Multiple Exceptions:

$$\mathcal{C}[\mathrm{a}^{\forall\alpha \mathrm{A}} m] \parallel_a v$$

$$\mapsto$$

$$v[(m, b^{A^{\perp}[m/\alpha]})]/a^{\exists\alpha A^{\perp}} \parallel_b (\mathcal{C}[b^{A[m/\alpha]}] \parallel_a v)$$

Logical Interpretation: EM_1



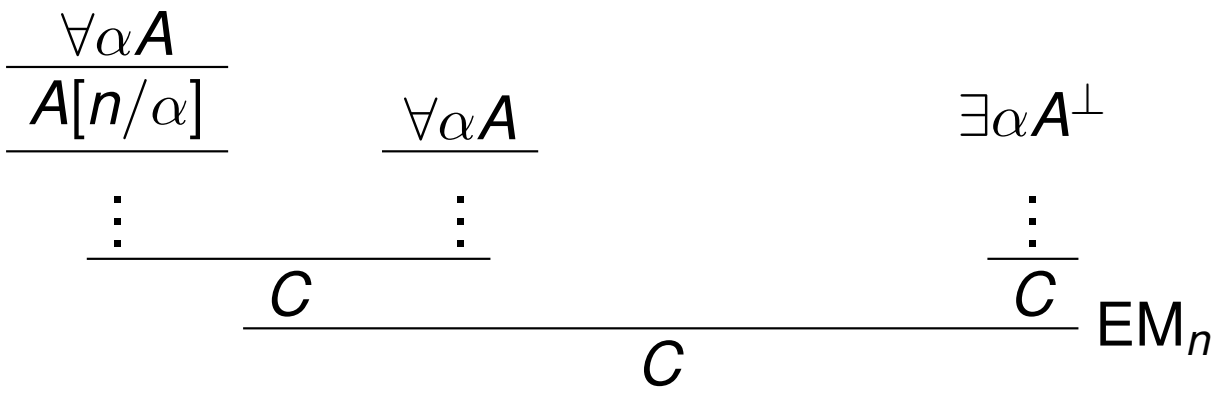
Logical Interpretation: EM_1

$$\dfrac{\dfrac{\dfrac{\forall\alpha P}{P[n/\alpha]}\quad\vdots}{C}\qquad\dfrac{\forall\alpha P}{\vdots}}{C}\qquad\dfrac{\exists\alpha\neg P}{\vdots}\;\dfrac{}{C}\;EM_1$$

converts to:

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg P[n/\alpha]}{\exists\alpha\neg P}}{\vdots}}{C}\qquad\dfrac{\dfrac{P[n/\alpha]}{\vdots}\qquad\dfrac{\forall\alpha P}{\vdots}}{C}}{C}\qquad\dfrac{\exists\alpha\neg P}{\vdots}\;\dfrac{}{C}\;EM_1\;EM_0$$

Logical Interpretation: EM_n



Logical Interpretation: EM_n

$$\dfrac{\dfrac{\dfrac{\forall\alpha A}{A[n/\alpha]}\quad\forall\alpha A}{C}\quad\exists\alpha A^{\perp}}{C}\ \mathrm{EM}_n$$

converts to:

$$\dfrac{\dfrac{\dfrac{A^{\perp}[n/\alpha]}{\exists\alpha\neg\mathrm{P}}}{C}\quad\dfrac{\dfrac{A[n/\alpha]\quad\forall\alpha A}{C}\quad\exists\alpha A^{\perp}}{C}\ \mathrm{EM}_n}{C}\ \mathrm{EM}_{n-1}$$

Logical Interpretation

$$\dfrac{\dfrac{\dfrac{[\forall\alpha A]}{\vdots}}{B \to C} \quad \dfrac{\dfrac{[\exists\alpha A^{\perp}]}{\vdots}}{B \to C}}{B \to C} \quad \dfrac{\vdots}{B}}{C}$$

converts to

$$\dfrac{\dfrac{\dfrac{[\forall\alpha A]}{\vdots}}{B \to C} \quad \dfrac{\vdots}{B}}{C} \quad \dfrac{\dfrac{\dfrac{[\exists\alpha A^{\perp}]}{\vdots}}{B \to C} \quad \dfrac{\vdots}{B}}{C}}{C}$$

Strong Normalization and Herbrand Disjunction Extraction

$$\mathsf{CL} \vdash t : A \implies t \in \mathsf{SN}$$

Strong Normalization and Herbrand Disjunction Extraction

$$\mathsf{CL} \vdash t : A \implies t \in \mathsf{SN}$$

$$\mathsf{CL} \vdash t : \exists \alpha \mathsf{P} \implies t \mapsto^* (m_0, u_0) \,|\, (m_1, u_1) \,|\, \ldots \,|\, (m_k, u_k)$$

Strong Normalization and Herbrand Disjunction Extraction

$$\mathsf{CL} \vdash t : A \implies t \in \mathsf{SN}$$

$$\mathsf{CL} \vdash t : \exists \alpha \mathsf{P} \implies t \mapsto^* (m_0, u_0) \,|\, (m_1, u_1) \,|\, \ldots \,|\, (m_k, u_k)$$

$$\mathsf{CL} \vdash \mathsf{P}[m_1/\alpha] \vee \cdots \vee \mathsf{P}[m_k/\alpha]$$

PA Arithmetic- Primitive Recursive Predicate Axioms

$$\frac{}{add(t, 0, t)} \qquad \frac{add(t_1, t_2, t_3)}{\mathrm{add}(t_1, S(t_2), S(t_3))}$$

$$\frac{}{mult(t, 0, 0)} \qquad \frac{mult(t_1, t_2, t_3) \;\; \mathrm{add}(t_3, t_1, t_4)}{mult(t_1, \mathrm{S}t_2, t_4)}$$

....

PA Arithmetic- Induction Rule

$$\frac{u : A[0] \qquad v : \forall \alpha\, A[\alpha] \to A[\mathsf{S}(\alpha)]}{R\, u\, v\, m : A[m]}$$

PA Arithmetic- Induction Rule

$$\frac{u : A[0] \qquad v : \forall \alpha\, A[\alpha] \to A[\mathsf{S}(\alpha)]}{R\, u\, v\, m : A[m]}$$

$$R\, u\, v\, 0 \mapsto u$$

$$R\, u\, v\, S(m) \mapsto v\, n\, (R\, u\, v\, m)$$

Excluded Middle - Revisited

$$\dfrac{\begin{array}{cc}[H^{\neg P} : \neg P] & [H^{P} : P] \\ \vdots & \vdots \\ u : C & v : C\end{array}}{u \mid v : C}\ \mathrm{EM}_0$$

Excluded Middle - Revisited

$$\dfrac{\begin{array}{cc}[H^{\neg P} : \neg P] & [H^{P} : P] \\ \vdots & \vdots \\ u : C & v : C\end{array}}{u \mid v : C}\ \mathrm{EM}_0$$

$u \mid v \mapsto u$, if P is false

$u \mid v \mapsto v$, if P is true

PA Arithmetic and Witness Extraction

$$\mathsf{PA} \vdash t : A \implies t \in \mathsf{SN}$$

PA Arithmetic and Witness Extraction

$$\mathsf{PA} \vdash t : A \implies t \in \mathsf{SN}$$

$$\mathsf{PA} \vdash t : \exists \alpha \mathsf{P}(\alpha) \implies t \mapsto^* (n, u) \text{ and } P(n) \text{ is true}$$

PA Arithmetic and Witness Extraction

$$\mathsf{PA} \vdash t : A \implies t \in \mathsf{SN}$$

$$\mathsf{PA} \vdash t : \exists \alpha \mathsf{P}(\alpha) \implies t \mapsto^* (n, u) \text{ and } P(n) \text{ is true}$$

$$\mathsf{PA} \vdash t : \forall \beta \exists \alpha \mathsf{P}(\beta, \alpha) \implies t\, m \mapsto^* (n, u) \text{ and } P(m, n) \text{ is true}$$

Bibliography

1. Girard, Proofs and Types, 1989.
2. Sorensen, Urzyczyn, Lectures on the Curry-Howard Isomorphism, 2006.
3. F. Aschieri, S. Berardi, G. Birolo, Realizability and Strong Normalization for a Curry-Howard Interpretation of HA + EM1, 2013.
4. F. Aschieri, M. Zorzi, On natural deduction in classical first-order logic: Curry-Howard correspondence, strong normalization and Herbrand's theorem, 2016.
5. F. Aschieri, A. Ciabattoni, F. A. Genco, Gödel Logic: from Natural Deduction to Parallel Computation, 2017.