Rachunek lambda, 2023-24

P. Urzyczyn

Wstęp

Rachunek lambda

- Abstrakcyjna teoria funkcji.
- Alternatywa dla teorii zbiorów w podstawach matematyki.[1]
- Aparat użyteczny w teorii obliczeń (teorii obliczalności).
- Model programowania funkcyjnego.
- Język dla semantyki programów.
- Model programowania z typami, model polimorfizmu.
- Język dowodów konstruktywnych i teorii typów.
- Język narzędzi wspomagających dowodzenie (Coq, ...).

[1] To się nie do końca udało, ale... może nie wszystko stracone?

Zbiory i funkcje

Sposób użycia:
$a \in A$ (należenie) $F(a)$ (aplikacja)

Tworzenie:
$\{x \mid W(x)\}$ (wycinanie[2]) $\lambda x\, W(x)$ (abstrakcja)

Ewaluacja:
$a \in \{x \mid W(x)\} \quad \Leftrightarrow \quad W(a)$ $\qquad (\lambda x\, W(x))(a) = W(a)$

[2] separation

Ekstensjonalność dla zbiorów

$A = B$ wtedy i tylko wtedy, gdy $\forall x\,(x \in A \Leftrightarrow x \in B)$

$A = \{x \mid x \in A\}$.

Example: let $A = \{y \mid y < x\}$. Then:

$A = \{x \mid x \in A\} = \{x \mid x \in \{y \mid y < x\}\} = \{x \mid x < x\} = \varnothing$

Błąd: zmienna x jest jednocześnie wolna i związana.

Ekstensjonalność dla funkcji

Dla zbiorów:

$A = B$ wtedy i tylko wtedy, gdy $\forall x\,(x \in A \Leftrightarrow x \in B)$

Dla funkcji:

$F = G$ wtedy i tylko wtedy, gdy $\forall x\,(F(x) = G(x))$

Dla zbiorów: $A = \{x \mid x \in A\}$

Dla funkcji: $F = \lambda x\, F(x)$

gdy x nie występuje wolno w F ani w A.

Ekstensjonalność (?)

Statyczne i dynamiczne rozumienie funkcji:

1. Jako przyporządkowanie, wykres, relację, zbiór par...
2. Jako regułę, przekształcenie, algorytm, metodę...

Teoria zbiorów nie nadaje się do opisu aspektu (2).

Zbiory i funkcje

- Teoria mnogości: teoria w logice 1. rzędu:
  - Należenie jest pojęciem pierwotnym;
  - Wycinanie (ograniczone) jest jednym z aksjomatów;
  - Ekstensjonalność jest pierwszym aksjomatem.
- Rachunek lambda: system formalny (a raczej rodzina systemów):
  - Aplikacja i abstrakcja są bazowymi operacjami;
  - Ekstensjonalność nie jest obowiązkowa.

Historia

- Logika kombinatoryczna:
  Abstrakcyjna teoria funkcji z aplikacją jako jedyną operacją. Eliminacja pojęcia zmiennej.
  - Moses Schönfinkel, 1924;
  - Haskell B. Curry, 1930.
- Rachunek lambda:
  - Alonzo Church, 1930;
  - S.C. Kleene, B. Rosser, 1935: sprzeczność logiczna;
  - Pojęcie funkcji obliczalnej, teza Churcha, 1936;
  - John McCarthy, 1958: Lisp;
  - Dana Scott, Ch. Strachey, 1969: semantyka denotacyjna.

Historia

- Typy:
  - Curry, Church, od początku;
  - N.G. de Bruijn, od 1967: system Automath;
  - William Howard, 1968: izomorfizm Curry'ego-Howarda;
  - J. Roger Hindley, 1969: algorytm Hindleya-Milnera;
  - Robin Milner, 1970: ML (polimorfizm);
  - Per Martin-Löf, od 1970: teoria typów Martina-Löfa;
  - Jean-Yves Girard, 1970: system F;
  - John Reynolds, 1974: polimorficzny rachunek lambda;
  - Około 1984: Coq (początki);
  - Około 1987: Haskell;
  - Około 2006: Homotopijna teoria typów.

O czym ma być ten wykład

- Rachunek lambda jako system redukcyjny:
  - Własność Churcha-Rossera, standaryzacja...
- Rachunek lambda jako teoria równościowa:
  - Drzewa Böhma, modele Scotta...
- Siła wyrazu:
  - Punkty stałe, reprezentowanie funkcji obliczalnych...
- Rachunek kombinatorów:
  - Jak sobie poradzić bez lambdy?
- Rachunki z typami:
  (typy proste, iloczynowe, polimorficzne, rekurencyjne)
  - Twierdzenia o normalizacji;
  - Formuły-typy (izomorfizm Curry'ego-Howarda);
  - Problemy decyzyjne.

Literatura

- Barendregt,
  The Lambda Calculus. Its Syntax and Semantics.
- Hindley, Seldin,
  Lambda-Calculus and Combinators, an Introduction.
- Girard, *Proofs and Types*
- Krivine, *Lambda-Calculus, Types and Models*
- Barendregt, Dekkers, Statman,
  Lambda Calculus with Types.
- Sørensen, Urzyczyn,
  Lectures on the Curry-Howard Isomorphism.
- Hindley, *Basic Simple Type Theory.*
- www.mimuw.edu.pl/~urzy/Lambda/erlambda.pdf

Beztypowy rachunek lambda

Beztypowy rachunek lambda

- Funkcja rozumiana jako działanie.
- Każdemu obiektowi można przypisać działanie, więc...
- ...nie ma innych obiektów niż funkcje.
- Przedmiotem działania może być cokolwiek, zatem...
- ...funkcja nie ma a priori ograniczonej dziedziny.
- Funkcja może być np. aplikowana sama do siebie.

Analogia: Każdy ciąg bitów można zinterpretować
– jako program;
– jako dane.

Składnia: lambda-wyrażenia

Lambda-wyrażenia:

– Zmienne x, y, z, ...

– Aplikacje (MN);

– Abstrakcje $(\lambda x\, M)$.

Konwencje:

– Opuszczamy zewnętrzne nawiasy;

– Aplikacja wiąże w lewo: MNP oznacza $(MN)P$

– Skrót z kropką: $\lambda x_1 \ldots x_n.M$ oznacza $\lambda x_1(\ldots(\lambda x_n M)\cdots)$.

Przykłady

$$
\begin{aligned}
\mathsf{I} &= \lambda x.x \\
\mathsf{K} &= \lambda xy.x \\
\mathsf{S} &= \lambda xyz.xz(yz) \\
2 &= \lambda fx.\, f(fx) \\
\omega &= \lambda x.xx \\
\Omega &= \omega\omega \\
\mathsf{Y} &= \lambda f.\,(\lambda x.f(xx))(\lambda x.f(xx))
\end{aligned}
$$

Zmienne *wolne* (globalne)

$\mathrm{FV}(x) = \{x\}$;
$\mathrm{FV}(MN) = \mathrm{FV}(M) \cup \mathrm{FV}(N)$;
$\mathrm{FV}(\lambda x\, M) = \mathrm{FV}(M) - \{x\}$.

Na przykład:

$\mathrm{FV}(\lambda x\, x) = \varnothing$;
$\mathrm{FV}(\lambda x.\, xy) = \{y\}$;
$\mathrm{FV}((\lambda x.\, xy)(\lambda y.\, xy)) = \{x, y\}$.

Alfa-konwersja

Wyrażenia $\lambda x.\, xy$ i $\lambda z.\, zy$ oznaczają tę samą operację („zaaplikuj dany argument do y").

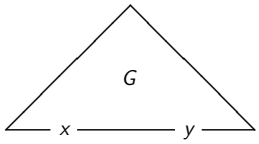
Należy je uważać za identyczne.

Alfa-konwersja: Utożsamiamy wyrażenia różniące się tylko wyborem zmiennych związanych.

Lambda-termy to klasy abstrakcji tego utożsamienia.

Łatwiej powiedzieć, niż zrobić...

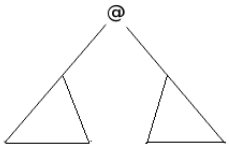
Termy jako grafy:



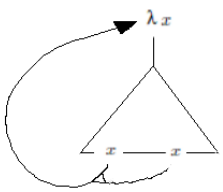
– Jeden wierzchołek początkowy;
– Zmienne wolne jako wierzchołki końcowe (liście).

Termy jako grafy: zmienna

Termy jako grafy: aplikacja

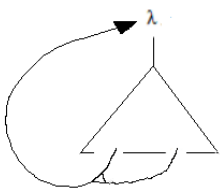


Termy jako grafy: abstrakcja

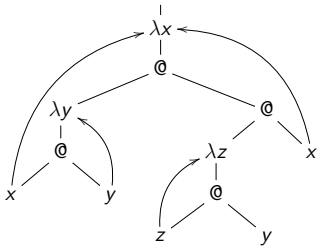


Termy jako grafy: abstrakcja

Zmienne związane sa niepotrzebne.

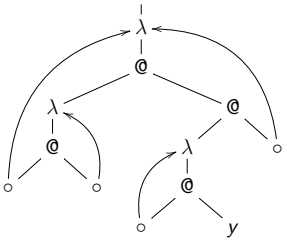


Przykład:



To jest graf wyrażenia $\lambda x.(\lambda y.\, xy)((\lambda z.\, zy)x)$

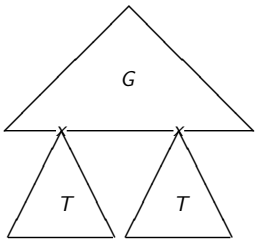
Przykład



To jest graf termu $\lambda x.(\lambda y.\, xy)((\lambda z.\, zy)x)$

Podstawienie $G[x := T]$

Podstawienie termu T do termu G w miejsce wolnych wystąpień zmiennej x.



Podstawienie

- $x[x := N] = N$;
- $y[x := N] = y$,
  gdy y jest zmienną różną od x;
- $(PQ)[x := N] = P[x := N]Q[x := N]$;
- $(\lambda y\, P)[x := N] = \lambda y.P[x := N]$,
  gdy $y \neq x$ oraz $y \notin \mathrm{FV}(N)$.

Wykonanie podstawienia na konkretnej reprezentacji termu może wymagać wymiany zmiennych:

$(\lambda y\, P)[x := N] = \lambda z\, P[y := z][x := N]$, gdzie z jest „nowe".

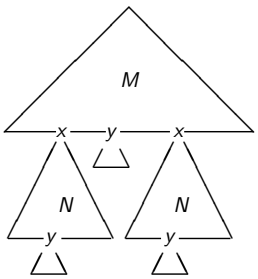
Lemat o podstawieniu

Lemat
Jeśli $x \neq y$ oraz albo $x \notin \mathrm{FV}(R)$ albo $y \notin \mathrm{FV}(M)$, to
$M[x := N][y := R] = M[y := R][x := N[y := R]]$.

Lemat o podstawieniu

$$M[x := N][y := R] = M[y := R][x := N[y := R]]$$

gdy $x \notin \mathrm{FV}(R)$ lub $y \notin \mathrm{FV}(M)$



Beta-redukcja

Najmniejsza relacja $\rightarrow_\beta$, spełniająca warunki:

- $(\lambda xP)Q \rightarrow_\beta P[x := Q]$;
- jeśli $M \rightarrow_\beta M'$, to:
  $MN \rightarrow_\beta M'N$, $NM \rightarrow_\beta NM'$ oraz $\lambda xM \rightarrow_\beta \lambda xM'$.

Term postaci $(\lambda xP)Q$ to *β-redeks*.

Relacja $\rightarrow_\beta$ to zredukowanie jednego dowolnego redeksu.

Wołanie przez nazwę

$$(\lambda xP)Q \rightarrow_\beta P[x := Q]$$

Ewaluacja procedury o parametrze formalnym x i treści P, gdy parametrem aktualnym jest Q:

Należy wstawić parametr aktualny do treści procedury, wymieniając, jeśli trzeba, lokalne identyfikatory na nowe.

Relacje pochodne:

Dowolna liczba kroków: $\twoheadrightarrow_\beta$ lub $\rightarrow_\beta^*$;

Niezerowa liczba kroków: $\rightarrow_\beta^+$;

Co najwyżej jeden krok: $\rightarrow_\beta^=$;

Równoważność (beta-konwersja): $=_\beta$.

Przykład: SKK $\twoheadrightarrow_\beta$ I

$$\begin{aligned}
\mathsf{SKK} &= (\lambda xyz.xz(yz))(\lambda xy.x)(\lambda xy.x) \\
&\rightarrow_\beta (\lambda yz.(\lambda xy.x)z(yz))(\lambda xy.x) \\
&\rightarrow_\beta \lambda z.(\lambda xy.x)z((\lambda xy.x)z) \\
&\rightarrow_\beta \lambda z.(\lambda y.z)((\lambda xy.x)z) \\
&\rightarrow_\beta \lambda z.(\lambda y.z)(\lambda y.z) \\
&\rightarrow_\beta \lambda z.z = \mathsf{I}
\end{aligned}$$

Ćwiczenia

- $(\lambda x.\, xx)(\lambda z.\, zyz)$
- 2(2(2(K))))
- 222K
- $(\lambda z.\, xxy)(\lambda z.\, xxy)$
- $(\lambda y.\, (\lambda x.\, z)(y\omega))\omega$
- $(\lambda x.\, xx)(\lambda zu.\, zu)$

Kompozycjonalność

Lemat
(1) Jeśli $M \rightarrow_\beta M'$, *to* $M[x := N] \rightarrow_\beta M'[x := N]$;
(2) Jeśli $N \rightarrow_\beta N'$, *to* $M[x := N] \twoheadrightarrow_\beta M[x := N']$,

Dowód: Indukcja ze względu na długość M. □

Wniosek
Jeśli $M \twoheadrightarrow_\beta M'$ *i* $N \twoheadrightarrow_\beta N'$, *to* $M[x := N] \twoheadrightarrow_\beta M'[x := N']$.

Normalizacja

Postać normalna to term bez redeksów.

Nie da się go redukować.

Term M *ma postać normalną* (*jest normalizowalny*),
gdy redukuje się do pewnej postaci normalnej.

Nazywamy ją *postacią normalną* termu M.

Term M jest *silnie normalizowalny* ($M \in \mathsf{SN}$),
gdy nie istnieje nieskończony ciąg

$$M = M_0 \rightarrow_\beta M_1 \rightarrow_\beta M_2 \rightarrow_\beta \cdots$$

Inaczej: każdy ciąg redukcji prowadzi do postaci normalnej.

Przykłady

- Term $\mathsf{S} = \lambda xyz.xz(yz)$ jest w postaci normalnej.
- Term SKK jest silnie normalizowalny i ma postać normalną **I**.
- Term $\mathbf{\Omega} = (\lambda x.\, xx)(\lambda x.\, xx)$ nie ma postaci normalnej.
- Term $(\lambda x.\, y)\mathbf{\Omega}$ ma postać normalną y, ale nie jest silnie normalizowalny.

Rachunek lambda jako teoria równościowa

Termy M i N są beta-równe ($M =_\beta N$) wtedy i tylko wtedy,
gdy równość „$M = N$" można udowodnić w systemie:

$$(\beta)\ \ (\lambda x\, M)N = M[x := N] \qquad x = x$$

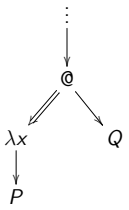
$$\frac{M = N}{MP = NP} \qquad \frac{M = N}{PM = PN} \qquad (\xi)\ \ \frac{M = N}{\lambda x\, M = \lambda x\, N}$$

$$\frac{M = N}{N = M} \qquad \frac{M = N, N = P}{M = P}$$

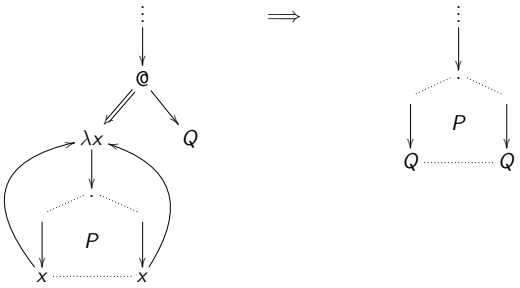
Jaja aligatorów

http://worrydream.com/AlligatorEggs/

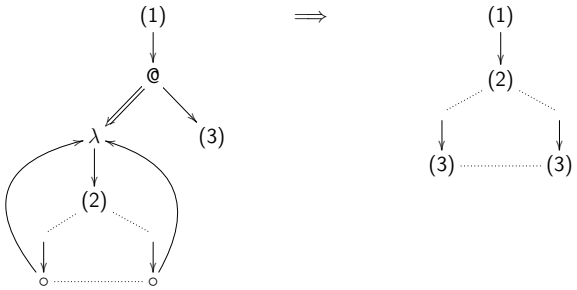
Beta-redeks $(\lambda x P)Q$



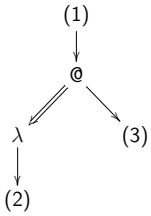
Beta-redukcja $(\lambda x P)Q \to P[x := Q]$



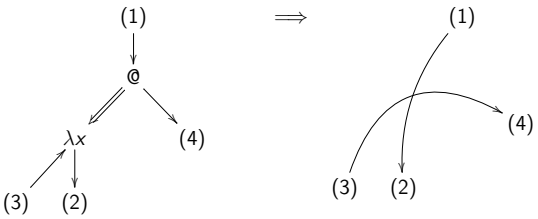
Beta-redukcja w grafie



Beta-redeks w grafie



Beta-redukcja nieco wyidealizowana



Pytanie:

Jak wyglądają postaci normalne?

- Zmienne;
- Abstrakcje $\lambda x\, M$, gdzie M normalne;
- Aplikacje? Tylko takie $x\vec{M}$, gdzie $\vec{M}$ normalne.

Ogólnie: $\lambda\vec{x}.\, y\vec{M}$, gdzie $\vec{M}$ normalne.

Ćwiczenia

Oznaczenia:

$P\uparrow$ czytamy „P nie ma postaci normalnej"

$P\downarrow$ czytamy „P ma postać normalną"

Zadanie: Które implikacje zachodzą dla dowolnych P i Q?

1. Jeśli $P\downarrow$ lub $Q\downarrow$, to $PQ\downarrow$?
2. Jeśli $P\downarrow$ i $Q\downarrow$, to $PQ\downarrow$?
3. Jeśli $PQ\downarrow$, to $P\downarrow$?
4. Jeśli $PQ\downarrow$, to $Q\downarrow$?
5. Jeśli $PQ\downarrow$, to $P\downarrow$ i $Q\downarrow$?
6. Jeśli $PQ\downarrow$, to $P\downarrow$ lub $Q\downarrow$?

Własności redukcji

Abstrakcyjne systemy redukcyjne

Abstrakcyjny system redukcyjny:
para $\langle A, \to \rangle$, gdzie $\to$ jest relacją binarną w A.

Oznaczenia:
$\to^+$ domknięcie przechodnie;
$\twoheadrightarrow$ domknięcie przechodnio-zwrotne (także $\to^*$);
$\to^=$ domknięcie zwrotne.

Postać normalna: takie $a \in A$, że $\forall b.\, a \not\to b$.

Normalizacja

Normalizacja (WN): Każdy element ma postać normalną.

Silna normalizacja (SN):
Każdy ciąg redukcji $a_0 \to a_1 \to a_2 \to \cdots$ jest skończony.

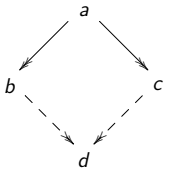
(Wtedy relacja $\twoheadleftarrow$ jest dobrym ufundowaniem. Dlaczego?)

Fakt: Beta-redukcja nie ma własności normalizacji:

$$\mathbf{\Omega} \to_\beta \mathbf{\Omega} \to_\beta \mathbf{\Omega} \to_\beta \cdots$$

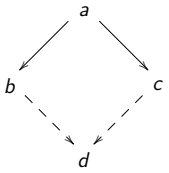
Własność Churcha-Rossera (CR)

Jeśli $a \twoheadrightarrow b$ i $a \twoheadrightarrow c$, to istnieje takie d, że $b \twoheadrightarrow d$ i $c \twoheadrightarrow d$.

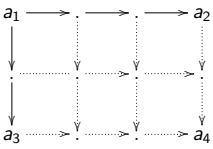


Własność rombu (diamond property)

Jeśli $a \to b$ i $a \to c$, to istnieje takie d, że $b \to d$ i $c \to d$.

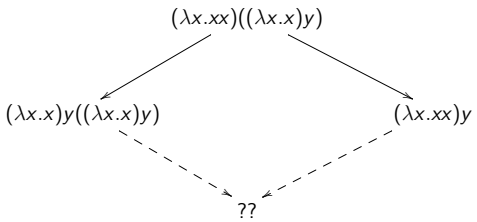


Własność rombu implikuje CR



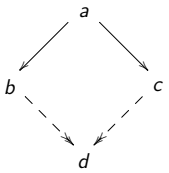
Własność rombu?

Fakt: Beta-redukcja nie ma własności rombu.



Słaba własność Churcha-Rossera (WCR)

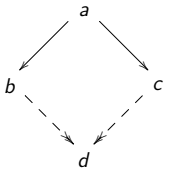
Jeśli $a \to b$ i $a \to c$, to istnieje takie d, że $b \twoheadrightarrow d$ i $c \twoheadrightarrow d$.



Czy beta-redukcja ma słabą własność Churcha-Rossera?

Słaba własność Churcha-Rossera (WCR)

Jeśli $a \to b$ i $a \to c$, to istnieje takie d, że $b \twoheadrightarrow d$ i $c \twoheadrightarrow d$.



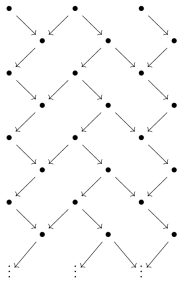
Fakt: Beta-redukcja ma słabą własność Churcha-Rossera.
Dlaczego?

WCR nie implikuje CR

Przykład:

$$a \longleftarrow b \longleftrightarrow c \longrightarrow d$$

Inny przykład

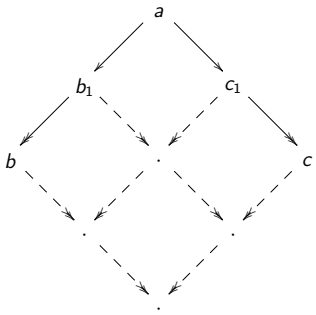


(Peter Selinger, *Lecture Notes on the Lambda Calculus*, Dalhousie University, 2013)

Lemat Newmana: WCR $\wedge$ SN $\Longrightarrow$ CR

Lemat Newmana:

$$\text{WCR} \wedge \text{SN} \Longrightarrow \text{CR}$$



Indukcja ze względu na $\twoheadleftarrow$.

Twierdzenie Churcha-Rossera: Beta ma własność CR

Definicja (complete development):

Term $M^\bullet$ to *pełne rozwinięcie* termu M.

– $x^\bullet = x$;
– $(\lambda x\, M)^\bullet = \lambda x\, M^\bullet$;
– $(MN)^\bullet = M^\bullet N^\bullet$, gdy M nie jest abstrakcją;
– $((\lambda x\, M)N)^\bullet = M^\bullet[x := N^\bullet]$.

Sens: jednoczesna redukcja wszystkich istniejących redeksów.
Czy $M^\bullet$ jest postacią normalną?

Relacja pomocnicza $\xrightarrow{1}$

– $x \xrightarrow{1} x$, gdy x jest zmienną;
– jeśli $M \xrightarrow{1} M'$, to $\lambda x M \xrightarrow{1} \lambda x M'$;
– jeśli $M \xrightarrow{1} M'$ i $N \xrightarrow{1} N'$, to:
$MN \xrightarrow{1} M'N'$, oraz
$(\lambda x M)N \xrightarrow{1} M'[x := N']$.

Sens: jednoczesna redukcja kilku redeksów już obecnych w termie.

Własności relacji $\xrightarrow{1}$

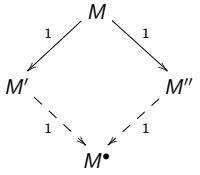
(1) Jeśli $M \xrightarrow{1} M'$, to $\text{FV}(M') \subseteq \text{FV}(M)$.

(2) Dla dowolnego M zachodzi $M \xrightarrow{1} M$ oraz $M \xrightarrow{1} M^\bullet$.

(3) Jeśli $M \xrightarrow{1} M'$ i $N \xrightarrow{1} N'$, to $M[x := N] \xrightarrow{1} M'[x := N']$.

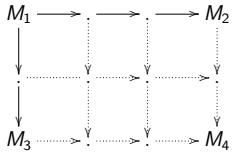
(4) Jeśli $M \xrightarrow{1} M'$, to $M' \xrightarrow{1} M^\bullet$.

Relacja $\xrightarrow{1}$ ma własność rombu



Dowód twierdzenia Churcha-Rossera

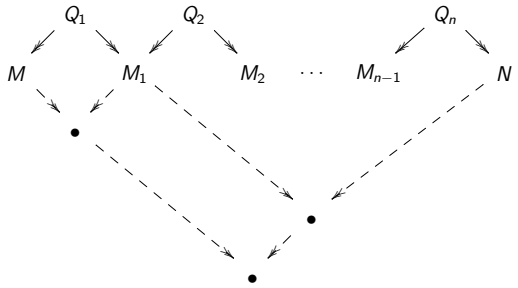
1) Ponieważ relacja $\overset{1}{\to}$ ma własność rombu, więc tym bardziej jej domknięcie przechodnio-zwrotne $\overset{1}{\twoheadrightarrow}$ ma własność rombu.



2) Ponieważ $\to_\beta \subseteq \overset{1}{\to} \subseteq \twoheadrightarrow_\beta$, więc $\overset{1}{\twoheadrightarrow}$ i $\twoheadrightarrow_\beta$ są równe.
3) Własność rombu dla $\twoheadrightarrow$ to własność CR dla $\to$.

Wnioski z twierdzenia Churcha-Rossera

(1) *Jeśli* $M =_\beta N$, *to* $M \twoheadrightarrow_\beta Q_\beta \twoheadleftarrow N$, *dla pewnego* Q.



Wnioski z twierdzenia Churcha-Rossera

(2) *Każdy term ma co najwyżej jedną postać normalną*

(3) *Jeśli* $M =_\beta N$ *i* N *normalny, to* $M \twoheadrightarrow_\beta N$

Dowód (3):
Jeśli $M =_\beta N$, to $M \twoheadrightarrow_\beta Q_\beta \twoheadleftarrow N$.
Skoro N normalne, to $N = Q$.

Wnioski z twierdzenia Churcha-Rossera

(4) *Beta-konwersja jest niesprzeczną teorią równościową*

Dowód: Na przykład $\not\vdash x = y$, ponieważ $x \neq_\beta y$.

Jak powstają nowe redeksy?

Typ 1: $(\lambda x \lambda y\, Q)NP \to_\beta (\lambda y\, Q[x := N])P$

Typ 2: $(\lambda x \ldots xP \ldots)(\lambda y\, Q) \to_\beta \ldots (\lambda y.\, Q)P \ldots$

Typ 3: $(\lambda x\, x)(\lambda y\, Q)P \to_\beta (\lambda y\, Q)P$

Eta-redukcja

Eta-reduction

The least relation $\to_\eta$, satisfying the conditions:

- $\lambda x.Mx \to_\eta M$, when $x \notin \mathrm{FV}(M)$;
- jeśli $M \to_\eta M'$, to $MN \to_\eta M'N$, $NM \to_\eta NM'$ oraz $\lambda xM \to_\eta \lambda xM'$.

Symbol $\to_{\beta\eta}$ stands for the union of relations $\to_\beta$ and $\to_\eta$.
Other definitions and notation are applicable respectively.

Eta-conversion

Axiom (η): $\lambda x.Mx = M$, when $x \notin \mathrm{FV}(M)$.

Rule (ext): $\dfrac{Mx = Nx}{M = N}$ (when $x \notin \mathrm{FV}(M) \cup \mathrm{FV}(N)$)

Fakt
Axiom (η) *and rule* (ext) *are equivalent.*

Proof.
(1) Assume (η) and let $Mx = Nx$. Then $\lambda x\, Mx = \lambda x\, Nx$, by rule (ξ), whence $M = N$.
(2) Since $(\lambda x.Mx)x = Mx$, one has $\lambda x.Mx = M$ by (ext). □

Ekstensjonalność i słaba ekstensjonalność

(ext) $\dfrac{Mx = Nx}{M = N}$ (when $x \notin \mathrm{FV}(M) \cup \mathrm{FV}(N)$)

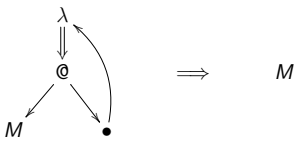
Jeśli $M = \lambda x\, P$ i $N = \lambda x\, Q$, to wystarczy:

$$(\xi)\ \frac{P = Q}{\lambda x\, P = \lambda x\, Q}$$

Słaba ekstensjonalność dla zbiorów to taka zasada

$$\frac{\vdash W(x) \leftrightarrow V(x)}{\{x \mid W(x)\} = \{x \mid V(x)\}}$$

Eta-reduction graphically



Eta-reduction amounts to removing a small loop in a graph.

Eta-reduction

Fakt
Eta-reduction is strongly normalizing.

Proof.
Because terms shrink under eta. □

Fakt
Eta-reduction is Church-Rosser.

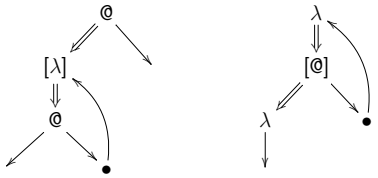
Proof.
Because it is WCR (easy), Newman's lemma applies. □

Beta-eta-redukcja

Relacja $\to_{\beta\eta}$ to suma $\to_\beta$ i $\to_\eta$.

Notacja $\twoheadrightarrow_{\beta\eta}$, $=_{\beta\eta}$ itd. stosuje się odpowiednio.

Beta-eta critical pairs



$(\lambda x\, Mx)N$ $\lambda x\, (\lambda y\, M)x$

Critical pair occurs when two redexes use the same resource.
Proving WCR amounts to resolving critical pairs.

Beta-eta critical pairs

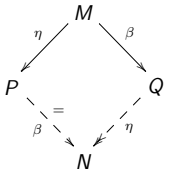
$$(\lambda x\, Mx)N \to MN$$

$$\lambda x\, (\lambda y\, M)x \to \lambda y\, M = \lambda x.\, M[y := x]$$

Morał
Beta-eta reduction is weakly Church-Rosser.

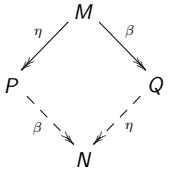
Towards Church-Rosser ($\beta\eta$)

Lemma
If $P \,{}_\eta\!\leftarrow M \to_\beta Q$ then there is N such that $P \to_\beta^= N \,{}_\eta\!\leftarrow Q$.

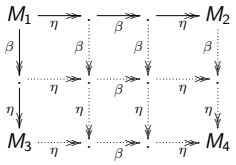


Towards Church-Rosser

Lemma
Relations $\twoheadrightarrow_\beta$ and $\twoheadrightarrow_\eta$ commute:
if $P \,{}_\eta\!\twoheadleftarrow M \twoheadrightarrow_\beta Q$ then there is N such that $P \twoheadrightarrow_\beta N \,{}_\eta\!\twoheadleftarrow Q$.



Beta-eta is Church-Rosser



Beta-SN versus beta-eta-SN

Theorem 0: *A term is $\beta\eta$-SN if and only if it is β-SN.*

Proof: ($\Rightarrow$) Obvious.
($\Leftarrow$) Ćwiczenia.

Eta-postponement (odkładanie eta-redukcji)

Theorem 1: *If $M \twoheadrightarrow_{\beta\eta} N$ then $M \twoheadrightarrow_{\beta} P \twoheadrightarrow_{\eta} N$, for some P.*

Theorem 2:
A term has a β-normal form iff it has a $\beta\eta$-normal form.

Proofs: Omitted.

Uwaga: Thm. 2 nie wynika *natychmiast* z Thm. 1.

Standaryzacja

Standard reduction:

Never reduce to the left of something already reduced.

Standard reduction:

$$(\lambda x.xx)((\lambda zy.z(zy))(\lambda u.u)(\lambda w.w)) \to$$
$$(\lambda x.xx)((\lambda y.(\lambda u.u)((\lambda u.u)y))(\lambda w.w)) \to$$
$$(\lambda x.xx)((\lambda u.u)((\lambda u.u)(\lambda w.w))) \to$$
$$(\lambda x.xx)((\lambda u.u)(\lambda w.w)) \to (\lambda x.xx)(\lambda w.w).$$

Non-standard reduction:

$$(\lambda x.xx)((\lambda zy.z(zy))(\lambda u.u)(\lambda w.w)) \to$$
$$(\lambda x.xx)((\lambda y.(\lambda u.u)((\lambda u.u)y))(\lambda w.w)) \to$$
$$(\lambda x.xx)((\lambda y.(\lambda u.u)y)(\lambda w.w)) \to$$
$$(\lambda x.xx)((\lambda y.y)(\lambda w.w)) \to (\lambda x.xx)(\lambda w.w).$$

Standardization

Theorem:

If $M \twoheadrightarrow_{\beta} N$ then there is a standard reduction from M to N.

Corollary:
If M has a β-normal form then the leftmost reduction leads to the normal form.

Slogan:
The leftmost *reduction strategy* is normalizing.

Redukcje czołowe i wewnętrzne

A term $\lambda\vec{x}.z\vec{R}$ is in *head normal form*.[3]

A term $\lambda\vec{x}.(\lambda y.P)Q\vec{R}$ has a *head redex* $(\lambda y.P)Q$.

A reduction step of the form

$$M = \lambda\vec{x}.(\lambda y.P)Q\vec{R} \to_{\beta} \lambda\vec{x}.P[y := Q]\vec{R} = N$$

is called *head reduction*. Write $M \xrightarrow{\mathrm{h}} N$.

Other reductions are *internal*. Write $M \xrightarrow{\mathrm{i}} N$.

[3]Postać normalna jest wtedy, gdy $\vec{R}$ są normalne.

Main Lemma

Lemma: If $M \twoheadrightarrow_{\beta} N$ then $M \xrightarrow{\mathrm{h}}\!\!\!\!\twoheadrightarrow P \xrightarrow{\mathrm{i}}\!\!\!\!\twoheadrightarrow N$, for some P.

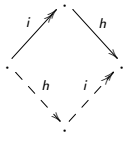
Warning: A naive proof attempt fails. The assumption:

$$\lambda\vec{x}.(\lambda y.P)Q\vec{R} \xrightarrow{\mathrm{i}}\!\!\!\!\twoheadrightarrow \lambda\vec{x}.(\lambda y.P')Q'\vec{R}' \xrightarrow{\mathrm{h}} \lambda\vec{x}.P'[y := Q']\vec{R}'$$

does not imply

$$\lambda\vec{x}.(\lambda y.P)Q\vec{R} \xrightarrow{\mathrm{h}} \lambda\vec{x}.P[y := Q]\vec{R} \xrightarrow{\mathrm{i}}\!\!\!\!\twoheadrightarrow \lambda\vec{x}.P'[y := Q']\vec{R}'.$$

This diagram is wrong.



Dygresja

Lemat łatwy:

Jeśli $M \xrightarrow{\mathrm{i}}\!\!\!\!\rightarrow N \xrightarrow{\mathrm{h}} P$, *to* $M \xrightarrow{\mathrm{h}} Q \twoheadrightarrow_\beta P$, *dla pewnego* Q.

Wniosek: *If* M *has a reduction with infinitely many head steps then it has an infinite head reduction.*

Standaryzacja: jeśli $M \twoheadrightarrow_\beta N$, to istnieje standardowa redukcja

Lemma: If $M \twoheadrightarrow_\beta N$ then $M \xrightarrow{\mathrm{h}}\!\!\!\!\rightarrow P \xrightarrow{\mathrm{i}}\!\!\!\!\rightarrow N$, for some P.

(Dowód lematu opuszczamy)

Dowód standaryzacji:
Najpierw wykonujemy same redukcje czołowe. Dostajemy term $P = \lambda\vec{x}.\, M_1 M_2 \dots M_k$, i dalej wszystkie redukcje są wewnątrz termów $M_1, M_2, \dots, M_k$. Można je łatwo przepermutować tak, aby najpierw redukować wewnątrz M_1, potem wewnątrz M_2 i tak dalej. Do tych redukcji można zastosować indukcję... ze względu na długość N.

Redukcje quasi-lewostronne

Lemma: *If* M *has a reduction with infinitely many head steps then it has an infinite head reduction.*

Uogólnienie:

A *quasi-leftmost reduction* is an infinite reduction sequence with infinitely many leftmost steps.

Redukcje quasi-lewostronne

Theorem: *A normalizing term has no quasi-leftmost reduction sequence.*

Proof: Ale co to za indukcja?

Suppose M has a quasi-leftmost reduction. If there is infinitely many head steps, then M has an infinite head (thus leftmost) reduction and cannot normalize.

Thus almost all reductions are internal. (W szczególności, od pewnego miejsca mamy czołowe postaci normalne.)

$$\lambda\vec{z}.yP_1 \dots P_k \xrightarrow{\mathrm{i}} \lambda\vec{z}.yP'_1 \dots P'_k \xrightarrow{\mathrm{i}} \lambda\vec{z}.yP''_1 \dots P''_k \xrightarrow{\mathrm{i}} \cdots$$

Infinitely many leftmost steps are within the same P_i.
By induction, P_i cannot normalize.
But then M cannot normalize.

Redukcje quasi-lewostronne

Theorem: *A normalizing term has no quasi-leftmost reduction sequence.*

Proof: Induction wrt the length of the normal form of M.

Suppose M has a quasi-leftmost reduction. If there is infinitely many head steps, then M has an infinite head (thus leftmost) reduction and cannot normalize.

Thus almost all reductions are internal. (W szczególności, od pewnego miejsca mamy czołowe postaci normalne.)

$$\lambda\vec{z}.yP_1 \dots P_k \xrightarrow{\mathrm{i}} \lambda\vec{z}.yP'_1 \dots P'_k \xrightarrow{\mathrm{i}} \lambda\vec{z}.yP''_1 \dots P''_k \xrightarrow{\mathrm{i}} \cdots$$

Infinitely many leftmost steps are within the same P_i.
By induction, P_i cannot normalize.
But then M cannot normalize.

The $\lambda\mathbf{I}$-calculus

The $\lambda\mathbf{I}$-*terms* (termy „relewantne"):

– variables;

– applications (MN), where M and N are $\lambda\mathbf{I}$-terms;

– abstractions $(\lambda x\, M)$, where M is a $\lambda\mathbf{I}$-term, and $x \in \mathrm{FV}(M)$.

Terms $\mathsf{S} = \lambda xyz.xz(yz)$ and $\mathsf{I} = \lambda x.x$ are $\lambda\mathbf{I}$-terms,
but $\mathsf{K} = \lambda xy.x$ is not.

Question

Are $\lambda\mathbf{I}$*-terms closed under reductions?*

Yes, if $M \to_{\beta\eta} N$, *and* M *is a* $\lambda\mathbf{I}$*-term, then so is* N.

Ale właściwie dlaczego? Jak to udowodnić?
Zróbmy to na tablicy.

Theorem: *If a* $\lambda\mathbf{I}$*-term has a normal form then it is SN.*

(Write $\to^\ell$ for leftmost beta-reduction.)

Lemma: *If* $M \in \lambda\mathrm{I}$ *and* $M \to^\ell N \in \mathrm{SN}$ *then* $M \in \mathrm{SN}$.

Proof: Induction wrt the length of the normal form of N.

Case 1: $M = \lambda\vec{x}.\, yR_1R_2 \dots R_k$. Then $N = \lambda\vec{x}.\, yR'_1 \dots R'_k$, with $R_i \to R'_i$, for some i, and $R_i = R'_i$, otherwise.

Since $N \in \mathrm{SN}$, also $R'_i \in \mathrm{SN}$, with shorter normal forms.
Thus all R_i are SN, and so is M.

Theorem: *If a $\lambda\mathbf{I}$-term has a normal form then it is SN.*

Lemma: *If $M \in \lambda\mathrm{I}$ and $M \to^{\ell} N \in \mathrm{SN}$ then $M \in \mathrm{SN}$.*

Proof: Induction wrt the length of the normal form of N.

Case 2:
$M = \lambda\vec{x}.\,(\lambda y\,P)QR_1 \dots R_k \to_\beta \lambda\vec{x}.\,P[y := Q]R_1 \dots R_k = N$.

Rozpatrzmy dowolną redukcję termu M.

All terms $P, Q, R_1, \dots, R_k$ are SN. Therefore *internal* reductions of M must terminate, but this can happen:

$M \overset{\mathrm{i}}{\twoheadrightarrow} \lambda\vec{x}.\,(\lambda y\,P')Q'R'_1 \dots R'_k \overset{\mathrm{h}}{\to} \lambda\vec{x}.\,P'[y := Q']R'_1 \dots R'_k \twoheadrightarrow \cdots$

But $N \twoheadrightarrow_\beta \lambda\vec{x}.\,P'[y := Q']R'_1 \dots R'_k$, and N is SN.

Question

Does the proof use the assumption that M is a $\lambda\mathbf{I}$-term? Where?

Yes, otherwise Q is not necessarily SN.

W istocie mamy taką obserwację:

Jeśli $\lambda\vec{x}.\,P[y := Q]\vec{R} \in$ SN, oraz $Q \in$ SN,
to $\lambda\vec{x}.\,(\lambda y\,P)Q\vec{R} \in$ SN.

Theorem: *If a $\lambda\mathbf{I}$-term has a normal form then it is SN.*

Proof: Let $M \in \lambda\mathrm{I}$ have normal form M'.

Then $M \twoheadrightarrow^{\ell} M' \in$ SN.

By induction wrt number of steps show that M is SN.

Siła wyrazu rachunku lambda

Curry's fixed point combinator $\mathbf{Y}$

$$\mathrm{Y} = \lambda f.\,(\lambda x.f(xx))(\lambda x.f(xx))$$

Fact: $\mathrm{Y}F =_\beta F(\mathrm{Y}F)$, for every F.

Proof: $\mathrm{Y}F \to_\beta (\lambda x.F(xx))(\lambda x.F(xx)) \to_\beta$
$F((\lambda x.F(xx))(\lambda x.F(xx))) \;{}_\beta\!\leftarrow F(\mathrm{Y}F)$

Comment

Lambda calculus is a fantastic world: every function has a fixpoint (every equation $X = Anything(X)$ has a solution)!

The moral sense is: every recursive definition is well-formed.

The price we pay for it is non-termination.

$\mathrm{Y}F =_\beta F(\mathrm{Y}F)$

Example: Find an M such that $Mxy =_\beta MxxyM$.

Solution: No problem, $M = \mathrm{Y}(\lambda m\,\lambda xy.\,mxxym)$.

Turing's fixed-point combinator

$$\mathbf{\Theta} = (\lambda xf.\,f(xxf))(\lambda xf.\,f(xxf))$$

Fact: $\mathbf{\Theta}F \twoheadrightarrow_\beta F(\mathbf{\Theta}F)$, for all F.

Proof: $\mathbf{\Theta}F = (\lambda xf.\,f(xxf))(\lambda xf.\,f(xxf))F \to_\beta$
$(\lambda f.\,f((\lambda xf.\,f(xxf))(\lambda xf.\,f(xxf))f)F \to_\beta$
$F((\lambda xf.\,f(xxf))(\lambda xf.\,f(xxf))F) = F(\mathbf{\Theta}F)$

Note the $\twoheadrightarrow_\beta$ rather than $=_\beta$.

Problems

1. Is there a term M such that $M =_\beta MSM$?
2. Can you solve non-fix-point equations like $MxxM =_\beta MxM$?
3. Can you solve any equation at all?
4. What are the fixpoints of the operator **SI** (where **S** and **I** are the standard combinators)?
5. Is there a fixpoint combinator in normal form?
6. Is **Y** beta-equal to Θ?
7. Can you solve systems of equations with 2 or more unknowns like $\{P = QSP, Q = PKQP\}$?

Example

Consider a combinator **day** $=$ hour hour $\cdots$ hour, where each hour is defined as $\lambda abcdefghijklmnopstuwxyzr.\, r(itisafixedpointcombinator)$.

How many hour*s should a* **day** *have to be a fixpoint combinator, i.e., to satisfy* **day** $F =_\beta F($**day** $F)$ *?*

Do domu: dobry przykład po polsku.

Ordered pair

$$\begin{aligned} \langle M, N\rangle &= \lambda x.xMN; \\ \pi_i &= \lambda x_1 x_2.x_i \qquad (i = 1, 2); \\ \Pi_i &= \lambda p.\, p\pi_i \qquad (i = 1, 2). \end{aligned}$$

It works:

$$\Pi_1\langle M, N\rangle \to_\beta \langle M, N\rangle\pi_1 \twoheadrightarrow_\beta M.$$

Question

$\langle M, N\rangle = \lambda x.xMN, \quad \pi_i = \lambda x_1 x_2.x_i, \quad \Pi_i = \lambda p.\, p\pi_i$

Will this work: $\langle \Pi_1 M, \Pi_2 M\rangle =_\beta M$*?*

Not when M *is not a pair; take e.g.* $M = x$.

Comment: This pairing is not *surjective*. And one cannot do better: Church-Rosser fails with surjective pairing!

Note the similarity with eta.

Conditional

$$\text{true} = \lambda xy.x \qquad \text{false} = \lambda xy.y$$

$$\text{if } P \text{ then } Q \text{ else } R = PQR.$$

It works:

if true then Q else $R \twoheadrightarrow_\beta Q$

if false then Q else $R \twoheadrightarrow_\beta R$.

A generalization: finite sets

Coding elements of a finite set $\{a_1, \ldots, a_n\}$:

$$a_i := \lambda x_1 \ldots x_n.\, x_i$$

Selection operator:

$$\text{case } a \text{ of } (F_1, \ldots, F_n) \;:=\; aF_1 \ldots F_n$$

It works:

$$\text{case } a_i \text{ of } (F_1, \ldots, F_n) \twoheadrightarrow_\beta F_i$$

Ordered tuples

Tuple = Selector:

$$\begin{aligned} \langle M_1, \ldots, M_n\rangle &= \lambda x.xM_1 \ldots M_n; \\ \pi_i^n &= \lambda x_1 \ldots x_n.x_i \qquad (i = 1, \ldots, n); \\ \Pi_i^n &= \lambda t.\, t\pi_i \qquad (i = 1, \ldots, n). \end{aligned}$$

It works:

$$\Pi_i^n\langle M_1, \ldots, M_n\rangle \twoheadrightarrow_\beta M_i$$

Church's numerals

$$c_n = \mathrm{n} = \lambda fx.f^n(x),$$

$$\begin{aligned} 0 &= \lambda fx.x; \\ 1 &= \lambda fx.fx; \\ 2 &= \lambda fx.f(fx); \\ 3 &= \lambda fx.f(f(fx)), \text{ etc.} \end{aligned}$$

Definable functions

A partial function $f : \mathbb{N}^k \multimap \mathbb{N}$ is *λ-definable*
if there is a term F such that for all $n_1, \ldots, n_k \in \mathbb{N}$:

- If $f(n_1, \ldots, n_k) = m$, then $F\mathrm{n}_1 \ldots \mathrm{n}_k =_\beta \mathrm{m}$;
- If $f(n_1, \ldots, n_k)$ is undefined
  then $F\mathrm{n}_1 \ldots \mathrm{n}_k$ does not normalize.

Function F is *well-definable* when, in addition:

- If $f(n_1, \ldots, n_k)$ is defined then $F\mathrm{n}_1 \ldots \mathrm{n}_k \in \mathrm{SN}$.

Comment

1. There are other definitions of numerals.
But the main results are the same.

2. There are other definitions of lambda-definability.
They may slightly differ in the undefined case.
But the main results are the same.

3. The notion of "well definable" is not as common,
but it proves technically useful.

Some well-definable functions

- Successor: $\mathrm{succ} = \lambda nfx.f(nfx)$;
  We have $\mathrm{succ}\,\mathrm{n} \twoheadrightarrow_\beta \mathrm{n}+1$, please check.
  Now try to define addition!
- Addition: $\mathrm{add} = \lambda mnfx.mf(nfx)$;
- Multiplication: $\mathrm{mult} = \lambda mnfx.m(nf)x$;
- Exponentiation: $\mathrm{exp} = \lambda mnfx.mnfx$;
- Test for zero: $\mathrm{zero} = \lambda m.m(\lambda y.\mathrm{false})\mathrm{true}$;
- Constant zero (with k arguments): $\mathrm{Z}_k = \lambda m_1 \ldots m_k.0$;
- Projection on the i-th coordinate: $\mathbf{\Pi}_i^k = \lambda m_1 \ldots m_k.m_i$.

A challenge: predecessor

$$p(n+1) = n, \quad p(0) = 0$$

Diggression: They say Kleene invented this definition
while his tooth was being extracted.

And that Church formulated Church Thesis
when Kleene came back from the dentist's.

Because this construction generalizes
to all primitive recursive functions.

Predecessor is definable

$$p(n+1) = n, \quad p(0) = 0$$

$$\mathrm{Step} = \lambda p.\langle \mathrm{succ}(p\pi_1), p\pi_1 \rangle$$
$$\mathrm{pred} = \lambda n.(n\,\mathrm{Step}\langle 0,0\rangle)\pi_2$$

How it works:

$$\mathrm{Step}\langle 0,0\rangle \twoheadrightarrow_\beta \langle 1,0\rangle$$
$$\mathrm{Step}\langle 1,0\rangle \twoheadrightarrow_\beta \langle 2,1\rangle$$
$$\mathrm{Step}\langle 2,1\rangle \twoheadrightarrow_\beta \langle 3,2\rangle,$$

and so on. (*Evaluate* pred 4.)

Nierozstrzygalność / Undecidability

Turing machine

We consider deterministic single-tape Turing Machines
computing functions from $\mathbb{N}$ to $\mathbb{N}$. We assume that:

- The machine is deterministic.
- There is one tape infinite to the right.
- The machine never writes a blank.
- The machine head never moves beyond the left tape end.

The input number is the length of input word and the output
number is represented by the head position.

Turing machine

It is convenient to represent machine IDs so that we can
have a direct access to head position,
and to the symbol at the left of tape head.

So if the tape contents is abrakadabra, with machine scanning
the letter d (at position 6) in state q, we represent it
as the quadruple $\langle akarba, q, dabra, 6\rangle$.

Turing machine

"Superkonfiguracja".

$$\langle w^R, q, v, \ell \rangle$$

(Notation: w^R is w written backwards, e.g. $011^R = 110$.)

- The tape contents is "wvBB..." (where B is blank);
- The head reads the first cell after w.
- The machine state is q;
- The length of w is ℓ.

Final configuration when $q \in \mathit{Final}$. The *result* is ℓ.

Initial configuration: $\mathcal{C}_n = \langle \varepsilon, q_0, \bullet^n, 0 \rangle$.

Machine computes a partial function from $\mathbb{N}$ to $\mathbb{N}$.

Encoding a Turing Machine

Superconfiguration $\langle w^R, q, v, n \rangle$ is encoded as a quadruple $\langle \mathrm{w}^R, \mathrm{q}, \mathrm{v}, \mathrm{n} \rangle$.

Initial configuration: $\mathit{Init} = \lambda x.\, \langle \mathit{nil}, \mathrm{q}_0, x(\lambda y.\, \langle \bullet, y \rangle)\mathit{nil}, 0 \rangle$.
Then $\mathit{Init}\ \mathrm{n} \twoheadrightarrow \langle \mathit{nil}, \mathrm{q}_0, \bullet^n, 0 \rangle$.

Stop test: $\mathit{Halt} = \lambda c.\, \Pi^4_2 c d_0 \ldots d_r = \mathsf{case}\ \Pi^4_2 c\ \mathsf{of}\ (d_0, \ldots, d_r)$,
where $d_i = \mathsf{true}$, for $q_i \in \mathit{Final}$, and $d_i = \mathsf{false}$, otherwise.

Result extraction: $\mathit{Out} = \lambda c.\, \Pi^4_4 c$.
(*Check how it works*)

Encoding a Turing Machine

The set of states $\{q_0, q_1, \ldots, q_r\}$ is finite.
States are encoded as projections: $\mathrm{q}_i = \lambda x_0 \ldots x_r . x_i$.

The alphabet $\{a_0, a_1, \ldots, a_m\}$ is finite.
Symbols are encoded as projections: $\mathrm{a}_i = \lambda x_0 \ldots x_m . x_i$.

Empty word ε is encoded as $\mathit{nil} = \langle \mathrm{B}, \mathrm{I} \rangle$.

Blank-free word aw (list $a :: w$) is encoded as $\langle \mathrm{a}, \mathrm{w} \rangle$.

List manipulation: $\mathit{head} = \Pi_1$, $\mathit{tail} = \Pi_2$.
(*Check how it works*)

Encoding a machine step

(Superconfiguration: $\langle \mathrm{w}^R, \mathrm{q}, \mathrm{v}, \mathrm{n} \rangle$.)

We take $\mathit{Next} = \lambda c.\, \Pi^4_2 c R^0 \ldots R^r$, where R^i codes the next configuration when the machine state in c is q_i.

How to define R^i? As a case selector on the scanned symbol:

$$R^i = \mathit{head}(\Pi^4_3 c) R^i_0 \ldots R^i_m,$$

where R^i_j is obtained according to the transition $\delta(q_i, a_j)$ determined by state q_i and symbol a_j.

Encoding a machine step

Superconfiguration: $\langle \mathrm{w}^R, \mathrm{q}, \mathrm{v}, \mathrm{n} \rangle$.

We need a term Next s.t. $\mathit{Next}(\text{code of } \mathcal{C}_1) \twoheadrightarrow_\beta (\text{code of } \mathcal{C}_2)$ whenever the machine can move from $\mathcal{C}_1$ to $\mathcal{C}_2$. We take:

$$\mathit{Next} = \lambda c.\, \mathsf{case}\ \Pi^4_2 c\ \mathsf{of}\ (R^0, \ldots, R^r) = \lambda c.\, \Pi^4_2 c R^0 \ldots R^r,$$

where R^i codes the configuration to be obtained if the machine state in the given configuration c is q_i.

Encoding transitions

Given code c of configuration $\mathcal{C} = \langle w^R, q_i, a_j v, n \rangle$, we need a term R^i_j representing the next configuration, depending on the transition table δ. (Na razie $a_j \neq \mathrm{B}$.)

If $\delta(q_i, a_j) = \langle b, 0, p \rangle$ (no move) then take

$$R^i_j = \langle \Pi^4_1 c, \mathrm{p}, \langle \mathrm{b}, \mathit{tail}(\Pi^4_3 c) \rangle, \Pi^4_4 c \rangle$$

If $\delta(q_i, a_j) = \langle b, +1, p \rangle$ (right move) then take

$$R^i_j = \langle \langle \mathrm{b}, (\Pi^4_1 c) \rangle, \mathrm{p}, \mathit{tail}(\Pi^4_3 c), \mathrm{succ}(\Pi^4_4 c) \rangle,$$

If $\delta(q_i, a_j) = \langle b, -1, p \rangle$ (left move) then take

$$R^i_j = \langle \mathit{tail}(\Pi^4_1 c), \mathrm{p}, \langle \mathit{head}(\Pi^4_1 c), \langle \mathrm{b}, (\mathit{tail}(\Pi^4_3 c)) \rangle \rangle, \mathrm{pred}(\Pi^4_4 c) \rangle.$$

(The head never goes beyond the left tape end.)

Special case: $a_j = \mathrm{B}$

Let $\mathcal{C} = \langle w^R, q_i, a_j v, n \rangle$ oraz $a_j = \mathrm{B}$.
Then $a_j v$ is "empty", i.e. $\Pi^4_3 c = \mathit{nil} = \langle \mathrm{B}, \mathrm{I} \rangle$.

If $\delta(q_i, a_j) = \langle b, 0, p \rangle$ (no move) then take

$$R^i_j = \langle \Pi^4_1 c, \mathrm{p}, \langle \mathrm{b}, \mathit{nil} \rangle, \Pi^4_4 c \rangle$$

If $\delta(q_i, a) = \langle b, +1, p \rangle$ (right move) then take

$$R^i_j = \langle \langle \mathrm{b}, (\Pi^4_1 c) \rangle, \mathrm{p}, \mathit{nil}, \mathrm{succ}(\Pi^4_4 c) \rangle,$$

If $\delta(q_i, a) = \langle b, -1, p \rangle$ (left move) then take

$$R^i_j = \langle \mathit{tail}(\Pi^4_1 c), \mathrm{p}, \langle \mathit{head}(\Pi^4_1 c), \langle \mathrm{b}, \mathit{nil} \rangle \rangle, \mathrm{pred}(\Pi^4_4 c) \rangle.$$

Encoding a Turing Machine

We have a term Next s.t.

$$\mathit{Next}(\text{code of } \mathcal{C}_1) \twoheadrightarrow_\beta (\text{code of } \mathcal{C}_2),$$

whenever the machine can move from $\mathcal{C}_1$ to $\mathcal{C}_2$.

Teraz trzeba powtarzać Next aż do skutku.

Rozwiązanie 1: Definiujemy taką funkcję F, że:
$F(c) = \mathsf{if}\ \mathit{Halt}\ c\ \mathsf{then}\ \mathit{Out}\ c\ \mathsf{else}\ F(\mathit{Next}\ c)$
Używamy kombinatora punktu stałego.

The function computed by the machine is λ-definable as

$$\lambda x.\, F(\mathit{Init}\ x).$$

Rozwiązanie 2

We have a term $Next$ s.t.

$$Next(\text{code of } \mathcal{C}_1) \twoheadrightarrow_\beta (\text{code of } \mathcal{C}_2),$$

whenever the machine can move from $\mathcal{C}_1$ to $\mathcal{C}_2$.

The function computed by the machine is λ-definable as:

$$\lambda x.W(Init\, x)W,$$

where

$W = \lambda c.$if $Halt\, c$ then $\lambda w.Out\, c$ else $\lambda w.w(Next\, c)w$.

Computing: $\mathcal{C}_0 \Rightarrow \mathcal{C}_1 \Rightarrow \mathcal{C}_2 \Rightarrow \cdots \Rightarrow \mathcal{C}_m$

$F = \lambda x.W(Init\, x)W$,

$W = \lambda c.$ if $Halt\, c$ then $\lambda w.Out\, c$ else $\lambda w.w(Next\, c)w$.

$F\mathbf{n} \to W(Init\, \mathbf{n})W \twoheadrightarrow W\mathrm{C}_0W \to$
[if $Halt\, \mathrm{C}_0$ then $\lambda w.Out\, \mathrm{C}_0$ else $\lambda w.w(Next\, \mathrm{C}_0)w]W \twoheadrightarrow$
$[\lambda w.w(Next\, \mathrm{C}_0)w]W \twoheadrightarrow [\lambda w.w\mathrm{C}_1w]W \to W\mathrm{C}_1W \to$
[if $Halt\, \mathrm{C}_1$ then $\lambda w.Out\, \mathrm{C}_1$ else $\lambda w.w(Next\, \mathrm{C}_1)w]W \twoheadrightarrow$
$[\lambda w.w(Next\, \mathrm{C}_1)w]W \twoheadrightarrow [\lambda w.w\mathrm{C}_2w]W \to W\mathrm{C}_2W \to$
$\cdots\cdots\cdots \twoheadrightarrow W\mathrm{C}_mW \to$
[if $Halt\, \mathrm{C}_m$ then $\lambda w.Out\, \mathrm{C}_m$ else $\lambda w.w(Next\, \mathrm{C}_m)w]W \twoheadrightarrow$
$[\lambda w.Out\, \mathrm{C}_m]W \to Out\, \mathrm{C}_m \twoheadrightarrow \mathbf{f(n)}$.

Question

We have defined $F = \lambda x.W(Init\, x)W$, and
$W = \lambda c.$ if $Halt\, c$ then $\lambda w.Out\, c$ else $\lambda w.w(Next\, c)w$.

Suppose we replace this definition by:
$W = \lambda cw.$ if $Halt\, c$ then $Out\, c$ else $w(Next\, c)w$.

Will anything change?

Answer: yes, the term $W\mathrm{C}_0W$ would never be SN.

Computability $\Leftrightarrow$ Definability

Twierdzenie

A partial function $f : \mathbb{N} \multimap \mathbb{N}$ is λ-definable if and only if it is partial computable.

Dowód ($\Rightarrow$):

Let f be λ-definable by F.
To compute $f(n)$ reduce $F\mathbf{n}$ to normal form.

Computability $\Leftrightarrow$ Definability

Twierdzenie

A partial function $f : \mathbb{N} \multimap \mathbb{N}$ is λ-definable if and only if it is partial computable.

Dowód ($\Leftarrow$): Zał. $f : \mathbb{N} \multimap \mathbb{N}$ jest częściowo obliczalna.
We encode the appropriate Turing Machine, and then:

- If $f(n)$ is defined then $F\mathbf{n}$ reduces to $\mathbf{f(n)}$.
- Otherwise we have an infinite quasi-leftmost reduction:

  $F\mathbf{n} \twoheadrightarrow W\mathrm{C}_0W \twoheadrightarrow [\lambda w.w\mathrm{C}_1w]W \to W\mathrm{C}_1W \twoheadrightarrow$
  $[\lambda w.w\mathrm{C}_2w]W \to W\mathrm{C}_2W \twoheadrightarrow \cdots$

  Therefore $F\mathrm{n}$ does not normalize.

The following are undecidable problems:

- Given M and N, does $M \twoheadrightarrow_\beta N$ hold?
- Given M and N, does $M =_\beta N$ hold?
- Given M, does M normalize?

Proof: Let $A \subseteq \mathbb{N}$ be recursively enumerable, not recursive.
Let χ_A be the partial characteristic function of A, i.e.,

$$\chi_A(x) = \texttt{if } x \in A \texttt{ then 0 else undefined}$$

Let H lambda-define the function χ_A.
To decide if $n \in A$, ask any of the questions:

- does $H\mathbf{n} \twoheadrightarrow_\beta \mathbf{0}$ hold?
- does $H\mathbf{n} =_\beta \mathbf{0}$ hold?
- does $H\mathbf{n}$ have a normal form?

Undecidability

The following problem is undecidable:

- Given M, does $M \in \mathrm{SN}$ hold?

Proof as before. To decide if $\chi_A(n)$ is defined,
ask if $H\mathbf{n}$ is strongly normalizable.

But that requires a stronger result:

Every partial computable function is well-definable.

Our coding has this property (proof omitted).

Ćwiczenie

Jaka jest różnica między „numerem PESEL" i „numerem NIP"?

(Które z tych określeń jest poprawne?)

Wskazówka:

PESEL = Powszechny Elektroniczny System Ewidencji Ludności
NIP = Numer Identyfikacji Podatkowej

Some numbering tricks

Let n be the number of a term M. Write $\overline{M}$ for $\mathbf{n}$.
This is the Church numeral for the number of M.

There are computable functions app and num such that

$$app(\text{number of } M)(\text{number of } N) = \text{number of } MN$$

$$num(n) = \text{number of the numeral } \mathbf{n}.$$

There are terms App and Num such that

$$App\,\overline{M}\,\overline{N} =_\beta \overline{MN} \qquad \text{and} \qquad Num\,\mathbf{n} =_\beta \overline{\mathbf{n}}.$$

Example: If the numeral $\mathbf{2} = \lambda fx.\, f(fx)$ has number 9, then $num(2) = 9$, $\overline{\mathbf{2}} = \mathbf{9}$, and $Num\,\mathbf{2} =_\beta \mathbf{9}$.

Now if 2 is the number of some M then $Num\,\overline{M} =_\beta \overline{\overline{M}} = \mathbf{9}$.

So $\overline{\overline{M}}$ is the Church numeral for...
the number of the numeral for the number of M.

Rice's theorem (Scott)

Theorem
Let A be a set of (numbers of) λ-terms, which is:

- *Nontrivial, i.e., not empty and not full;*
- *Closed under $=_\beta$.*

Then A is undecidable.

Warning: Not every interesting set is closed under $=_\beta$.
For instance, the set SN and sets of typable terms.

Proof of Rice's theorem

Suppose A is decidable, nontrivial, and closed under $=_\beta$.
Let F define the total characteristic function of A:

$$F\overline{M} =_\beta 0, \text{ for } M \in A \qquad \text{and} \qquad F\overline{M} =_\beta 1, \text{ for } M \notin A.$$

Take $M_1 \in A$, $M_2 \notin A$ and define

$$G = \lambda x.\, \text{if zero}(Fx) \text{ then } M_2 \text{ else } M_1.$$

Let N be such that $G(\overline{N}) =_\beta N$.

If $N \in A$ then $N =_\beta G(\overline{N}) =_\beta M_2 \notin A$.

If $N \notin A$ then $N =_\beta G(\overline{N}) =_\beta M_1 \in A$, contradiction.

Adding equational axioms

Example 1: Recall that $1 = \lambda fx.fx$. Add the axiom $\mathsf{I} = 1$ to the equational theory of λ-calculus.

Then, for every M, one proves:

$$M = \mathsf{I}M = 1M = \lambda x.\, Mx.$$

Example 2: Now add the axiom $\mathsf{K} = \mathsf{S}$.

Then, for every M, one proves:

$$M = \mathsf{SI}(\mathsf{K}M)\mathsf{I} = \mathsf{KI}(\mathsf{K}M)\mathsf{I} = \mathsf{I}.$$

This extension is inconsistent.

A fixed-point theorem

Theorem
For every term F, there is a term X such that $F(\overline{X}) =_\beta X$.

Proof

Take $X = Z\overline{Z}$, where $Z = \lambda x.\, F(App\,x(Num\,x))$. Then:

$$X = Z\overline{Z} =_\beta F(App\,\overline{Z}\,(Num\,\overline{Z})) =_\beta$$

$$F(App\,\overline{Z}\,\overline{\overline{Z}}) =_\beta F(\overline{Z\overline{Z}}) = F(\overline{X}).$$

(Compare $X = Z\overline{Z}$ to $\mathsf{Y}f = (\lambda x.\, f(xx))(\lambda x.\, f(xx))$.)

Proof idea

Suppose A is decidable, nontrivial, and closed under $=_\beta$.
We prove that there is no lambda-term F such that:

$$FM =_\beta 0, \text{ for } M \in A \qquad \text{and} \qquad FM =_\beta 1, \text{ for } M \notin A.$$

(If such an F existed, the set A would be decidable.)
(But otherwise it is not enough.)

Take $M_1 \in A$, $M_2 \notin A$ and define

$$G = \lambda x.\, \text{if zero}(Fx) \text{ then } M_2 \text{ else } M_1.$$

Let $N = \mathsf{Y}G$. Then $G(N) =_\beta N$.

If $N \in A$ then $N =_\beta G(N) =_\beta M_2 \notin A$.

If $N \notin A$ then $N =_\beta G(N) =_\beta M_1 \in A$, contradiction.

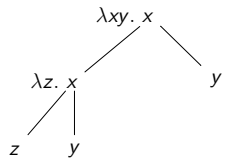
Równość w rachunku lambda

Böhm's Theorem

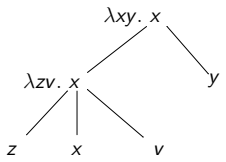
Let M, N be β-normal combinators with $M \neq_{\beta\eta} N$.
Then $M\vec{P} =_\beta \mathsf{true}$ and $N\vec{P} =_\beta \mathsf{false}$, for some $\vec{P}$.

Moral: Any consistent lambda-theory must discriminate between beta-eta normal forms.

Böhm Trees (finite case)

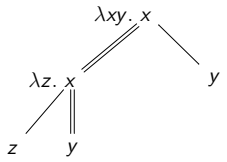


$M = \lambda xy.x(\lambda z.xzy)y$

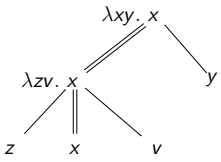


$N = \lambda xy.x(\lambda zv.xzxv)y$

Böhm Trees: the difference



$M = \lambda xy.x(\lambda z.xzy)y$



$N = \lambda xy.x(\lambda zv.xzxv)y$

Trick: Applying M to $\lambda uv.\langle u, v\rangle$ gives $\lambda y.\langle\lambda z.\langle z, y\rangle, y\rangle$.
And components can be extracted from a pair.

Discriminating terms

$M = \lambda xy.x(\lambda z.xzy)y$ $\qquad$ $N = \lambda xy.x(\lambda zv.xzxv)y$

Applying M and N to $P = \lambda uv.\langle u,\ v\rangle$, then to any Q yields:

$\langle\lambda z.\langle z, Q\rangle, Q\rangle$ $\qquad$ $\langle\lambda zv.\langle z, P\rangle v,\ Q\rangle$

Next appply both to true, I, false to obtain:

$\lambda z.\langle z, Q\rangle$	$\lambda zv.\langle z, P\rangle v$
$\langle \mathsf{I}, Q\rangle$	$\lambda v.\langle \mathsf{I}, P\rangle v$
Q	$P = \lambda uv.\langle u, v\rangle$

Choose $Q = \lambda uvw.\,\mathsf{true}$ and apply both sides to false, I, true:

true $\qquad$ false.

Twierdzenie Böhma: szkic dowodu

- Operator tworzenia krotki

  $T_p = \lambda x_1 \ldots x_p.\langle x_1, \ldots, x_p\rangle = \lambda x_1 \ldots x_p \lambda f.\, fx_1 \ldots x_p.$

- Podstawienie $S = [x_i := T_{p_i}]_{i=1,\ldots,n}$ jest *m-podstawieniem*, gdy $p_1, \ldots p_n$ są większe od m i parami różne.
- Termy M i N są *m-rozróżnialne*, gdy
  dla dowolnego m-podstawienia S
  istnieje taki ciąg kombinatorów $\vec{L}$, że:

  $M[S]\vec{L} \twoheadrightarrow_\beta \mathsf{true}$ $\qquad$ oraz $\qquad$ $N[S]\vec{L} \twoheadrightarrow_\beta \mathsf{false}$.

Twierdzenie Böhma: główny lemat

Jeśli M i N są w postaci normalnej oraz $M \neq_{\beta\eta} N$,
to M i N są m-rozróżnialne dla prawie wszystkich m.

Czyli: jeśli m jest dostatecznie duże, to
dla każdego m-podstawienia S, są takie $\vec{L}$, że

$M[S]\vec{L} \twoheadrightarrow_\beta \mathsf{true}$ $\qquad$ oraz $\qquad$ $N[S]\vec{L} \twoheadrightarrow_\beta \mathsf{false}$.

Twierdzenie Böhma: główny lemat

Jeśli M i N są w postaci normalnej oraz $M \neq_{\beta\eta} N$,
to M i N są m-rozróżnialne dla prawie wszystkich m.

Dowód: Indukcja ze względu na sumę długości termów.
Można założyć, że $M = xP_1 \ldots P_k$ i $N = yQ_1 \ldots Q_\ell$, bo:

- aby rozróżnić $\lambda x\, M$ i $\lambda x\, N$ wystarczy rozróżnić M i N;
- jak zabraknie lambdy, to zamiast M bierzemy $\lambda x\, Mx$.

□

Twierdzenie Böhma: główny lemat

Jeśli M i N są w postaci normalnej oraz $M \neq_{\beta\eta} N$,
to M i N są m-rozróżnialne dla prawie wszystkich m.

Dowód: Indukcja ze względu na sumę długości termów.
Można założyć, że $M = xP_1 \ldots P_k$ i $N = yQ_1 \ldots Q_\ell$. Wtedy:

$M[S] = T_p P_1[S] \ldots P_k[S] \twoheadrightarrow_\beta \lambda\vec{u}\langle P_1[S], \ldots, P_k[S], u_1, \ldots, u_{p-k}\rangle$

$N[S] = T_q Q_1[S] \ldots Q_\ell[S] \twoheadrightarrow_\beta \lambda\vec{v}\langle Q_1[S], \ldots, Q_\ell[S], v_1, \ldots, v_{q-\ell}\rangle$

Jeśli $x = y$ i $k = \ell$, to $P_i \neq_{\beta\eta} Q_i$ dla pewnego i.
Rzutujemy krotkę na i-tą współrzędną i stosujemy indukcję.
Przypadki bazowe: $x \neq y$, lub $x = y$ ale $k \neq \ell$: ćwiczenia. □

The standard theory

The Meaning of "Value" and "Undefined"

First idea: *Value = Normal form.*
Undefined = without normal form.

Can we identify all such terms?

No: for instance $\lambda x.x\mathsf{K}\Omega = \lambda x.x\mathsf{S}\Omega$ implies $\mathsf{K} = \mathsf{S}$ (apply both to K).

Moral: A term without normal form can still behave in a well-defined way. In a sense it has a "value".

Better idea: *Value = Head normal form.*
Undefined = without head normal form.

Solvability (rozwiązalność)

A closed term is *solvable* iff $M\vec{P} =_\beta \mathsf{I}$, for some closed $\vec{P}$.

If $\mathrm{FV}(M) = \vec{x}$ then M is *solvable* iff $\lambda\vec{x}.M$ is solvable.

Theorem
A term is solvable iff it has a head normal form.

Proof for closed terms:

($\Rightarrow$) If $M\vec{P} =_\beta \mathsf{I}$ then $M\vec{P} \twoheadrightarrow_\beta \mathsf{I}$. If $M\vec{P}$ head normalizes then also M must head normalize.

($\Leftarrow$) If $M =_\beta \lambda x_1 x_2 \ldots x_n.x_i R_1 \ldots R_m$ then $MP\ldots P =_\beta \mathsf{I}$, for $P = \lambda y_1 \ldots y_m.\mathsf{I}$.

The standard theory

We identify all unsolvable terms as "undefined".

Which solvable terms may be now be consistently identified?

We cannot classify terms by their head normal forms.
Too many of them!

We can only *observe* their behaviour.

Observational equivalence

Terms M, N with $\mathrm{FV}(M) \cup \mathrm{FV}(N) = \vec{x}$, are *observationally equivalent* ($M \equiv N$) when, for all closed P:

$$P(\lambda\vec{x}.M) \text{ is solvable} \iff P(\lambda\vec{x}.N) \text{ is solvable}$$

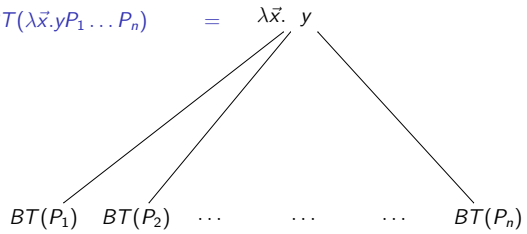
Put it differently: for any "context" $C[\,]$:

$$C[M] \text{ is solvable} \iff C[N] \text{ is solvable}$$

Note: If $M =_\eta N$ then $M \equiv N$.

Böhm Trees

$BT(\lambda\vec{x}.yP_1\ldots P_n)$ $=$



If M has a hnf N then $BT(M) = BT(N)$.

If M is unsolvable then $BT(M) = \bot$.

Example 1: any fixed-point combinator Z

Since $\mathsf{Z}f =_\beta f(\mathsf{Z}f)$ for a fresh variable f, it must be the case that $\mathsf{Z} =_\beta \lambda f.fT$, for some T. Therefore:

$fT =_\beta \mathsf{Z}f =_\beta f(\mathsf{Z}f) = f((\lambda f.fT)f) =_\beta f(fT)$.

Hence $fT =_\beta f(fT) =_\beta f(f(fT)) =_\beta f(f(f(fT)))\cdots$

The term Z unfolds into an "infinite term" $\lambda f.f(f(f(\ldots$

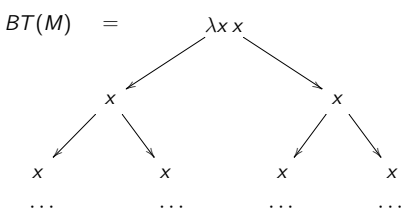
Its Böhm tree has one infinite branch:

$$\lambda f.f\text{——}f\text{——}f\text{——}\cdots$$

Example 2: $M = \mathsf{Y}(\lambda mx.\,x(mx)(mx))$

Let $M = \mathsf{Y}(\lambda mx.\,x(mx)(mx))$. Then $M =_\beta \lambda x.\,x(Mx)(Mx)$.

$BT(M)$ $=$



Example 3: $\mathsf{J} = \mathsf{Y}(\lambda fxy.\,x(fy)) = \mathsf{YB}'$

Write B' for $\lambda fxy.\,x(fy)$. Then:

$\mathsf{J} = \mathsf{YB}' =_\beta \mathsf{B}'\mathsf{J} =_\beta \lambda xy_0.\,x(\mathsf{J}y_0) =_\beta \lambda xy_0.\,x(\mathsf{B}'\mathsf{J}y_0)$
$=_\beta \lambda xy_0.\,x(\lambda y_1.\,y_0(\mathsf{J}y_1)) =_\beta \lambda xy_0.\,x(\lambda y_1.\,y_0(\mathsf{B}'\mathsf{J}y_1)) =_\beta \ldots$

The tree $BT(\mathsf{J})$ consists of one infinite path:

$$\lambda xy_0.\,x\text{——}\lambda y_1.\,y_0\text{——}\lambda y_2.\,y_1\text{——}\lambda y_3.\,y_2\text{——}\cdots$$

Example 3: $\mathsf{J} = \mathsf{Y}(\lambda fxy.\, x(fy))$

The tree $BT(\mathsf{J})$ consists of one infinite path:

$\lambda xy_0.\, x$——$\lambda y_1.\, y_0$——$\lambda y_2.\, y_1$——$\lambda y_3.\, y_2$—— $\cdots$

The tree $BT(\mathsf{I})$ consists of a single node: $\lambda x\, x$

The first can be obtained from the second by means of an infinite sequence of *η-expansions*:

$\lambda x\, x \quad {}_\eta\!\leftarrow \quad \lambda xy_0.\, x$——$y_0 \quad {}_\eta\!\leftarrow \quad \lambda xy_0.\, x$——$\lambda y_1.\, y_0$——$y_1$

When are terms observationally equivalent?

Böhm trees B i B' are *η-equivalent* ($B \approx_\eta B'$), if there are two (possibly infinite) sequences of η-expansions:

$$B = B_0 \;{}_\eta\!\leftarrow B_1 \;{}_\eta\!\leftarrow B_2 \;{}_\eta\!\leftarrow B_3 \;{}_\eta\!\leftarrow \cdots$$
$$B' = B'_0 \;{}_\eta\!\leftarrow B'_1 \;{}_\eta\!\leftarrow B'_2 \;{}_\eta\!\leftarrow B'_3 \;{}_\eta\!\leftarrow \cdots$$

converging to the same (possibly infinite) tree.

Theorem (Wadsworth)

Terms M and N are observationally equivalent if and only if $BT(M) \approx_\eta BT(N)$.

Logika kombinatoryczna
Combinatory Logic

Combinatory Logic, or *the calculus of combinators*

Terms:

- Variables;
- Constants K and S;
- Applications (FG).

The "weak" reduction

- $\mathsf{K}FG \to_w F$;
- $\mathsf{S}FGH \to_w FH(GH)$;
- If $F \to_w G$, then $FH \to_w GH$ and $HF \to_w HG$.

Notation $\twoheadrightarrow_w$, $=_w$ etc. used as usual.

Examples of combinators

- $\mathsf{I} = \mathsf{SKK}$
  $\mathsf{I}F \to_w \mathsf{K}F(\mathsf{K}F) \to_w F$
- $\Omega = \mathsf{SII}(\mathsf{SII})$
  $\Omega \to_w \mathsf{I}(\mathsf{SII})(\mathsf{I}(\mathsf{SII})) \twoheadrightarrow_w \Omega$
- $\mathsf{W} = \mathsf{SS}(\mathsf{KI})$
  $\mathsf{W}FG \to_w \mathsf{S}F(\mathsf{KI}F)G \to_w \mathsf{SFI}G \to_w FG(\mathsf{I}G) \twoheadrightarrow_w FGG$

Examples of combinators

- $\mathsf{B} = \mathsf{S}(\mathsf{KS})\mathsf{K}$
  $\mathsf{B}FGH \to_w \mathsf{KS}F(\mathsf{K}F)GH \to_w \mathsf{S}(\mathsf{K}F)GH \to_w$
  $\to_w \mathsf{K}FH(GH) \to_w F(GH)$
- $\mathsf{C} = \mathsf{S}(\mathsf{BBS})(\mathsf{KK})$
  $\mathsf{C}FGH \to_w FHG$
- $\mathsf{B}' = \mathsf{CB}$
  $\mathsf{B}'FGH \twoheadrightarrow_w G(FH)$

Remark: Each of those can be added as a new constant.

Examples of combinators

- $0 = \mathsf{KI}$ — $0FG \twoheadrightarrow_w G$
- $1 = \mathsf{SB}(\mathsf{KI})$ — $1FG \twoheadrightarrow_w FG$
- $2 = \mathsf{SB}(\mathsf{SB}(\mathsf{KI}))$ — $2FG \twoheadrightarrow_w F(FG)$

Fixed point combinators

$$\mathsf{Y} = \mathsf{WS}(\mathsf{BWB})$$

$$\Theta = \mathsf{WI}(\mathsf{B}(\mathsf{SI})(\mathsf{WI}))$$

From CL to lambda

- $(x)_\Lambda = x$;
- $(\mathsf{K})_\Lambda = \lambda xy.x$;
- $(\mathsf{S})_\Lambda = \lambda xyz.xz(yz)$;
- $(FG)_\Lambda = (F)_\Lambda(G)_\Lambda$.

Properties:

- If $F \twoheadrightarrow_w G$, then $(F)_\Lambda \twoheadrightarrow_\beta (G)_\Lambda$.
- If $F =_w G$, then $(F)_\Lambda =_\beta (G)_\Lambda$.

Good and bad properties

- Church-Rosser;
- Standardization;
- Definability of computable functions;
- Undecidability.

From CL to lambda

If $F =_w G$, then $(F)_\Lambda =_\beta (G)_\Lambda$.
This is a "morphism" from $\mathsf{CL}/_{=_w}$ to $\Lambda/_{=_\beta}$.

Ale nie monomorfizm: $(F)_\Lambda =_\beta (G)_\Lambda$ nie implikuje $F =_w G$.
Na przykład $(\mathsf{KI})_\Lambda =_\beta \lambda xy.\, y =_\beta (\mathsf{S}(\mathsf{K}(\mathsf{KI}))\mathsf{I})_\Lambda$

From CL to lambda

Twierdzenie (Jarosław Tworek, 2010)
Istnieje efektywna redukcja $\varphi : \mathsf{CL} \to \Lambda$ o własności:

$F =_w G$, *wtedy i tylko wtedy, gdy* $\varphi(F) =_\beta \varphi(G)$.

Sens: Relacja $=_w$ redukuje się do relacji $=_\beta$.

From lambda to CL: combinatory abstraction

- $\lambda^* x.\, F = \mathsf{K}F$, when $x \notin \mathsf{FV}(F)$;
- $\lambda^* x.\, x = \mathsf{I}$;
- $\lambda^* x.\, FG = \mathsf{S}(\lambda^* x.F)(\lambda^* x.G)$, otherwise.

Lemma 1: $(\lambda^* x.F)G \twoheadrightarrow_w F[x := G]$.

Lemma 2: $(\lambda^* x.F)_\Lambda =_\beta \lambda x.\, (F)_\Lambda$.

Lemma 1: $(\lambda^* x.F)G \twoheadrightarrow_w F[x := G]$

Case 1: If $x \notin \mathsf{FV}(F)$ then
$(\lambda^* x.F)G = \mathsf{K}FG \to_w F = F[x := G]$.

Case 2: If $F = x$ then $(\lambda^* x.F)G = \mathsf{I}G \to_w G = x[x := G]$.

Case 3: If $F = F'F''$ then
$(\lambda^* x.F)G = \mathsf{S}(\lambda^* x.F')(\lambda^* x.F'')G \to_w$
$\to_w (\lambda^* x.F')G((\lambda^* x.F'')G) \twoheadrightarrow_w F'[x := G]F''[x := G]$.

Combinatory completeness

Lemma 1: $(\lambda^* x.F)G \twoheadrightarrow_w F[x := G]$.

Theorem (Combinatory completeness)
Given any x and F, there is H such that, for all G,

$$HG \twoheadrightarrow_w F[x := G]$$

From lambda to CL

- $(x)_C = x$;
- $(MN)_C = (M)_C(N)_C$;
- $(\lambda x.M)_C = \lambda^* x.(M)_C$.

Good property: $((M)_C)_\Lambda =_\beta M$.

Proof: Induction with respect to M, using:

$$(\lambda^* x.F)_\Lambda =_\beta \lambda x.(F)_\Lambda.$$

Moral: Translation from CL to lambda is "surjective".

Translation from CL to lambda is surjective

Corollary: *Terms* K *and* S *form a basis of lambda-calculus: every term (up to β-equality) can be obtained from* K, S, *and variables, by means of application.*

Proof: $M =_\beta ((M)_C)_\Lambda$.

Translation from CL to lambda is surjective

- Złożenie $((\ )_C)_\Lambda$ jest identycznością.
- Złożenie $((\ )_\Lambda)_C$ nie jest identycznością. Na przykład
$$((\mathsf{K})_\Lambda)_C = \mathsf{S}(\mathsf{KK})\mathsf{I} \neq_w \mathsf{K}.$$

Bad property

- $M =_\beta N$ does **not** imply $(M)_C =_w (N)_C$. For example:
$$(\lambda x.\mathbf{K}\mathbf{I}x)_C =_w \mathsf{S}(\mathsf{K}(\mathsf{KI}))\mathsf{I} \neq_w \mathsf{KI} = (\lambda x\,\mathbf{I})_C.$$

W rzeczy samej, bo $\lambda x.\mathbf{K}\mathbf{I}x \to_\beta \lambda x\,\mathbf{I}$, ale:

$(\lambda x.\mathbf{K}\mathbf{I}x)_C = \mathsf{S}(\lambda^* x.(\mathbf{K}\mathbf{I})_C)(\lambda^* x.x) = \mathsf{S}(\mathsf{K}((\mathbf{K})_C(\mathbf{I})_C))\mathsf{I} = \mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{KK})\mathsf{II}))\mathsf{I} =_w \mathsf{S}(\mathsf{K}(\mathsf{KI}))\mathsf{I} \neq_w \mathsf{KI} = (\lambda x\,\mathbf{I})_C$.

Moral:

- This is *not* a morphism from $\Lambda/_{=_\beta}$ to $\mathsf{CL}/_{=_w}$.
- „Weak" equality is *stronger* than beta-equality.

The guilty one

The combinatory abstraction λ^* is not weakly extensional. Rule (ξ) is not valid for λ^* and $=_w$:

$$\frac{M = N}{\lambda x.M = \lambda x.N,} \qquad (\xi)$$

Indeed, $\lambda^* x.\mathsf{KI}x = \mathsf{S}(\mathsf{K}(\mathsf{KI}))\mathsf{I} \neq_w \mathsf{KI} = \lambda^* x.\mathsf{I}$.

Extensional CL

- If $G =_w H$ then $G =_{ext} H$;
- If $Gx =_{ext} Hx$ and $x \notin \mathsf{FV}(G) \cup \mathsf{FV}(H)$ then $G =_{ext} H$;
- If $G =_{ext} G'$ then $GH =_{ext} G'H$ and $HG =_{ext} HG'$.

Properties:

- $G =_{ext} H$ if and only if $(G)_\Lambda =_{\beta\eta} (H)_\Lambda$;
- $M =_{\beta\eta} N$ if and only if $(M)_C =_{ext} (N)_C$.
- $((G)_\Lambda)_C =_{ext} G$, for all G.

Open problem

Define a computable translation T from lambda to CL with:

$$M =_\beta N \qquad \Longleftrightarrow \qquad T(M) =_w T(N).$$

NCL - "Naive Combinatory Logic"

Terms: If F, G are terms then $\mathsf{P}FG$ is a term.
Write $F \Rightarrow G$ for $\mathsf{P}FG$.

Formulas:
– Every term F is a formula;
– Equations $F = G$ are formulas.

NCL - Axioms and rules

Axioms:

$F = F \qquad KFG = F \qquad SFGH = FH(GH)$

$F \Rightarrow F \qquad (F \Rightarrow (F \Rightarrow G)) \Rightarrow (F \Rightarrow G)$

Rules:

$$\frac{M = N}{MQ = NQ} \qquad \frac{M = N}{QM = QN}$$

$$\frac{M = N}{N = M} \qquad \frac{M = N,\ N = Q}{M = Q}$$

$$\frac{M, \quad M = N}{N} \qquad \frac{M, \quad M \Rightarrow N}{N}$$

Modele

Curry's Paradox

Take any term F and define $N = Y(\lambda^* x.\, x \Rightarrow F)$.

Then $N = N \Rightarrow F$ in NCL.

Using axiom $N \Rightarrow N$ we obtain that $N \Rightarrow (N \Rightarrow F)$.

Thus $N \Rightarrow F$, using $(N \Rightarrow (N \Rightarrow F)) \Rightarrow (N \Rightarrow F)$.

This proves N, because $N = N \Rightarrow F$.

Eventually, F follows by modus ponens.

Moral: System NCL is logically inconsistent.

Semantics

Goal: Interpret any term M as an element $[\![M]\!]$ of some structure A, so that $M =_\beta N$ implies $[\![M]\!] = [\![N]\!]$.

More precisely, $[\![M]\!]$ may depend on a *valuation*:

$$v : Var \to A.$$

Write $[\![M]\!]_v$ for the value of M under v.

Lambda-interpretation: $\mathcal{A} = \langle A, \cdot, [\![\ ]\!] \rangle$

Application $\cdot$ is a binary operation in A;

$[\![\ ]\!] : \Lambda \times A^{Var} \to A$.

Write $[\![M]\!]_v$ instead of $[\![\ ]\!](M, v)$.

Postulates:

(a) $[\![x]\!]_v = v(x)$;

(b) $[\![PQ]\!]_v = [\![P]\!]_v \cdot [\![Q]\!]_v$;

(c) $[\![\lambda x.P]\!]_v \cdot a = [\![P]\!]_{v[x \mapsto a]}$, for any $a \in A$;

(d) If $v|_{\mathrm{FV}(P)} = u|_{\mathrm{FV}(P)}$, then $[\![P]\!]_v = [\![P]\!]_u$.

Extensionality

Write $a \approx b$ when $a \cdot c = b \cdot c$, for all c.

Extensional interpretation: $a \approx b$ implies $a = b$, for all a, b.

Dygresja

Jeśli znamy $[\![P]\!]_v$ i $[\![Q]\!]_v$, to znamy $[\![PQ]\!]_v = [\![P]\!]_v \cdot [\![Q]\!]_v$.
Znaczenie aplikacji zależy tylko od znaczenia jej składowych.
To samo chcemy mieć dla abstrakcji.

Kiedy powinno zachodzić $[\![\lambda x\, P]\!]_v = [\![\lambda x\, Q]\!]_v$?

Równość $[\![P]\!]_v = [\![Q]\!]_v$ to trochę za mało.

Warunek $\forall v\, ([\![P]\!]_v = [\![Q]\!]_v)$ to trochę za dużo.

Ale $\forall a \in A.\, [\![P]\!]_{v[x \mapsto a]} = [\![Q]\!]_{v[x \mapsto a]}$ jest w sam raz.

Czyli $[\![\lambda x\, P]\!]_v \approx [\![\lambda x\, Q]\!]_v$.

Słaba ekstensjonalność

Weakly extensional interpretation:

$[\![\lambda x.M]\!]_v \approx [\![\lambda x.N]\!]_v$ implies $[\![\lambda x.M]\!]_v = [\![\lambda x.N]\!]_v$, for all N, v.

Meaning: Abstraction makes sense algebraically.

(N.B. $[\![\lambda x.M]\!]_v \approx [\![\lambda x.N]\!]_v$ iff $[\![M]\!]_{v[x \mapsto a]} = [\![N]\!]_{v[x \mapsto a]}$, all a.)

Lambda-algebra and lambda-model

Lambda-algebra: a lambda-interpretation satisfying beta:

$$M =_\beta N \quad \text{implies} \quad [\![M]\!]_\nu = [\![N]\!]_\nu$$

Lambda-model: Weakly extensional lambda-interpretation:

$$[\![\lambda x.M]\!]_\nu \approx [\![\lambda x.N]\!]_\nu \quad \text{implies} \quad [\![\lambda x.M]\!]_\nu = [\![\lambda x.N]\!]_\nu$$

"Trivial" examples

Fact

Structures $\mathfrak{M}(\lambda)$ and $\mathfrak{M}(\lambda\eta)$ are lambda-models.
Structures $\mathfrak{M}^0(\lambda)$ and $\mathfrak{M}^0(\lambda\eta)$ are lambda-algebras, but not lambda-models.

It may happen that $MP =_\beta NP$ for all closed P,
(i.e. $[\lambda x.\, Mx] \approx [\lambda x.\, Nx]$),

but $Mx \neq_\beta Nx$
(i.e. $[\lambda x.\, Mx] \neq [\lambda x.\, Nx]$).

(Gordon Plotkin)

"Trivial" examples

$\mathfrak{M}(\lambda) = \langle \Lambda/_{=_\beta}, \cdot, [\![\ ]\!] \rangle$,
where $[\![M]\!]_\nu = [M[\vec{x} := \nu(\vec{x})]]_\beta$ and $[M]_\beta \cdot [N]_\beta = [MN]_\beta$

Similarly:

$\mathfrak{M}(\lambda\eta) = \langle \Lambda/_{=_{\beta\eta}}, \cdot, [\![\ ]\!] \rangle$;
$\mathfrak{M}^0(\lambda) = \langle \Lambda^0/_{=_\beta}, \cdot, [\![\ ]\!] \rangle$;
$\mathfrak{M}^0(\lambda\eta) = \langle \Lambda^0/_{=_{\beta\eta}}, \cdot, [\![\ ]\!] \rangle$,
gdzie Λ^0 oznacza zbiór wszystkich kombinatorów.

Very Important Lemma

Lemma

In every lambda-model, $[\![M[x := N]]\!]_\nu = [\![M]\!]_{\nu[x \mapsto [\![N]\!]_\nu]}$.

Proof: Induction wrt M. First assume that $x \notin \mathrm{FV}(N)$.

Krok indukcyjny dla abstrakcji $M = \lambda y\, P$:

$[\![(\lambda y\, P)[x := N]]\!]_{\nu[x \mapsto [\![N]\!]_\nu]} \cdot a = [\![\lambda y.P[x := N]]\!]_\nu \cdot a$
$= [\![P[x := N]]\!]_{\nu[y \mapsto a]} = [\![P]\!]_{\nu[y \mapsto a][x \mapsto [\![N]\!]_{\nu[y \mapsto a]}]}$
$= [\![P]\!]_{\nu[y \mapsto a][x \mapsto [\![N]\!]_\nu]} = [\![\lambda y.P]\!]_{\nu[x \mapsto [\![N]\!]_\nu]} \cdot a$, for all a.
Therefore $[\![(\lambda y\, P)[x := N]]\!]_\nu = [\![(\lambda y.P)]\!]_{\nu[x \mapsto [\![N]\!]_\nu]}$.

Very Important Lemma

Lemma

In every lambda-model, $[\![M[x := N]]\!]_\nu = [\![M]\!]_{\nu[x \mapsto [\![N]\!]_\nu]}$

Proof.

In case $x \in \mathrm{FV}(N)$ take fresh z and let $w = \nu[z \mapsto [\![N]\!]_\nu]$.

$[\![M[x := N]]\!]_\nu = [\![M[x := z][z := N]]\!]_\nu = [\![M[x := z]]\!]_w =$
$[\![M]\!]_{w[x \mapsto [\![z]\!]_w]} = [\![M]\!]_{\nu[x \mapsto [\![z]\!]_w]} = [\![M]\!]_{\nu[x \mapsto [\![N]\!]_\nu]}$. □

Soundness

Proposition

Every lambda-model is a lambda-algebra:

$$M =_\beta N \quad \textit{implies} \quad [\![M]\!]_\nu = [\![N]\!]_\nu$$

Proof.

Induction wrt $M =_\beta N$. Non-immediate cases are two:

(Beta)
$[\![(\lambda x.P)Q]\!]_\nu = [\![\lambda x.P]\!]_\nu \cdot [\![Q]\!]_\nu = [\![P]\!]_{\nu[x \mapsto [\![Q]\!]_\nu]} = [\![P[x := Q]]\!]_\nu$.

(Xi) Let $P =_\beta Q$ and let $M = \lambda x.P$, $N = \lambda x.Q$. Then
$[\![M]\!]_\nu \cdot a = [\![P]\!]_{\nu[x \mapsto a]} = [\![Q]\!]_{\nu[x \mapsto a]} = [\![N]\!]_\nu \cdot a$, for all a. □

Completeness

Theorem

The following are equivalent:

1) $M =_\beta N$;
2) $\mathcal{A} \models M = N$, *for every lambda-algebra* $\mathcal{A}$;
3) $\mathcal{A} \models M = N$, *for every lambda-model* $\mathcal{A}$.

Proof.

(1)⇒(2) Z definicji lambda-algebry.
(2)⇒(3) Bo każdy lambda-model jest lambda-algebrą.
(3)⇒(1) Bo $\mathfrak{M}(\lambda)$ jest lambda-modelem. □

Modele Scotta

Semantyka w zbiorach częściowo uporządkowanych

Complete partial orders

Let $\langle A, \leq \rangle$ be a partial order.

A subset $B \subseteq A$ is *directed (skierowany)* when
for every $a, b \in B$ there is $c \in B$ with $a, b \leq c$.

The set A is a *complete partial order (cpo)* when
every directed subset has a supremum.

It follows that every cpo has a least element $\bot = \sup \varnothing$.

Complete partial orders

Let $\langle A, \leq \rangle$ and $\langle B, \leq \rangle$ be cpos, and $f : A \to B$.

Then f is *monotone* if $a \leq a'$ implies $f(a) \leq f(a')$.

And f is *continuous* if $\sup f(C) = f(\sup C)$
for every **nonempty** directed $C \subseteq A$.

Fact: *Every continuous function is monotone.*

$[A \to B]$ is the set of all continuous functions from A to B

Complete partial orders

If $\langle A, \leq \rangle$ and $\langle B, \leq \rangle$ are cpos then:

- The product $A \times B$ is a cpo with
  $\langle a, b \rangle \leq \langle a', b' \rangle$ iff $a \leq a'$ and $b \leq b'$.
- The function space $[A \to B]$ is a cpo with
  $f \leq g$ iff $\forall a.\, f(a) \leq g(a)$.

Continuous functions

Lemma
A function $f : A \times B \to C$ is continuous iff it is continuous wrt both arguments, i.e. all functions of the form $\lambda\!\!\lambda a.\, f(a, b)$ and $\lambda\!\!\lambda b.\, f(a, b)$ are continuous.

Continuous functions

Lemma
The application $App : [A \to B] \times A \to B$ is continuous.

Proof hint: Use the previous lemma.

Lemma
The abstraction $Abs : [(A \times B) \to C] \to [A \to [B \to C]]$, given by $Abs(F)(a)(b) = F(a, b)$, is continuous.

Reflexive cpo

The cpo D is *reflexive* iff there are continuous functions
$F : D \to [D \to D]$ and $G : [D \to D] \to D$,
with $F \circ G = \mathrm{id}_{[D \to D]}$.

Then F must be onto and G is injective.

The following are equivalent conditions:
"$G \circ F = \mathrm{id}_D$", "$G$ onto", "F injective".

Reflexive cpo

$F : D \to [D \to D]$, $G : [D \to D] \to D$, $F \circ G = \mathrm{id}$.

Define application as $a \cdot b = F(a)(b)$ so that $G(f) \cdot b = f(b)$.

Define interpretation as

- $[\![x]\!]_\nu = \nu(x)$;
- $[\![PQ]\!]_\nu = [\![P]\!]_\nu \cdot [\![Q]\!]_\nu$;
- $[\![\lambda x.P]\!]_\nu = G(\lambda\!\!\lambda a.[\![P]\!]_{\nu[x \mapsto a]})$.

Fact: This is a (well-defined) lambda interpretation.
(Use continuity of App and Abs.)

Reflexive cpo

Theorem
A reflexive cpo is a lambda-model.

Proof.
Należy udowodnić słabą ekstensjonalność.
Załóżmy, że $[\![\lambda x.M]\!]_\nu \cdot a = [\![\lambda x.N]\!]_\nu \cdot a$, dla każdego a.
Inaczej, $\lambda\!\!\lambda a.[\![M]\!]_{\nu[x \mapsto a]} = \lambda\!\!\lambda a.[\![N]\!]_{\nu[x \mapsto a]}$.
Chcemy $[\![\lambda x.M]\!]_\nu = [\![\lambda x.N]\!]_\nu$.Ale:
$[\![\lambda x.M]\!]_\nu = G(\lambda\!\!\lambda a.[\![M]\!]_{\nu[x \mapsto a]})$
$[\![\lambda x.N]\!]_\nu = G(\lambda\!\!\lambda a.[\![N]\!]_{\nu[x \mapsto a]})$.

□

Reflexive cpo

The cpo D is *reflexive* iff there are continuous functions $F : D \to [D \to D]$ and $G : [D \to D] \to D$, with $F \circ G = \mathrm{id}_{[D \to D]}$.

Theorem
A reflexive cpo is a lambda-model.

Ale jak takie zrobić?

A reflexive cpo: Model $\mathcal{P}_\omega = \langle \mathrm{P}(\mathbb{N}), \subseteq \rangle$

Notation $P_\omega = \mathrm{P}(\mathbb{N})$.

Every set is a directed union of its finite subsets.

Lemma
A function $f : P_\omega \to P_\omega$ is continuous iff
$f(a) = \bigcup\{f(e) \mid e \text{ finite and } e \subseteq a\}$,
for all $a \in P_\omega$.

Moral: A continuous function is fully determined by its values on finite arguments.

Encodings in $\mathcal{P}_\omega$

Pairs:

$$(m, n) = \frac{(n+m)(n+m+1)}{2} + m,$$

Finite sets: $e_0 = \varnothing$, and

$$e_n = \{k_0, k_1, \ldots, k_{r-1}\}, \quad \text{for } n = \sum_{i<r} 2^{k_i}.$$

$\mathcal{P}_\omega$ is reflexive

$$\mathit{graph}(f) = \{(n, m) \mid m \in f(e_n)\};$$
$$\mathit{fun}(a)(x) = \{m \mid \exists n \in \mathbb{N}(e_n \subseteq x \wedge (n, m) \in a)\}.$$

Lemma: Functions *graph* and *fun* are continuous, and

$$\mathit{fun} \circ \mathit{graph} = \mathrm{id}_{[P_\omega \to P_\omega]}.$$

Proof: $\mathit{fun}(\mathit{graph}(f))(x) =$
$= \{m \mid \exists n \in \mathbb{N}(e_n \subseteq x \wedge (n, m) \in \mathit{graph}(f))\}$
$= \{m \mid \exists n \in \mathbb{N}(e_n \subseteq x \wedge m \in f(e_n))\}$
$= \bigcup\{f(e_n) \mid e_n \subseteq x\} = f(x)$.

Przykłady w $\mathcal{P}_\omega$ (ćwiczenie)

- $[\![\mathsf{I}]\!] = \mathit{graph}(\mathrm{id}) = \{(n, m) \mid m \in e_n\}$;
- $[\![\mathsf{K}]\!] = \{(m, (k, \ell)) \mid \ell \in e_m\}$;
- $[\![\omega]\!] = \{(x, m) \mid \exists n(e_n \subseteq e_x \wedge (n, m) \in e_x)\}$;
- $[\![\Omega]\!] = \varnothing = \bot$.

$\mathcal{P}_\omega$ is not a model of η-conversion

$$\mathit{graph}(f) = \{(n, m) \mid m \in f(e_n)\};$$
$$\mathit{fun}(a)(x) = \{m \mid \exists n \in \mathbb{N}(e_n \subseteq x \wedge (n, m) \in a)\}.$$

Every nonempty $\mathit{graph}(f)$ is infinite:
If $(n, m) \in \mathit{graph}(f)$ then also $(k, m) \in \mathit{graph}(f)$, for $e_n \subseteq e_k$.
(Thus $\mathit{graph} \circ \mathit{fun} \neq \mathrm{id}_{P_\omega}$.)

Fact: $\mathcal{P}_\omega$ is not a model of η-conversion: $\mathcal{P}_\omega \not\models x = \lambda y.xy$.
In particular, $\mathcal{P}_\omega$ is not extensional.

Proof: Let $v(x) = a$, where $a \neq \varnothing$ is finite.
Then $[\![x]\!]_v = a$ is finite. But $[\![\lambda y.xy]\!]_v = \mathit{graph}(\ldots)$ is infinite.

Theory of $\mathcal{P}_\omega$

Theorem (Hyland)
$\mathcal{P}_\omega \models M = N \iff BT(M) = BT(N)$

Trzy własności modeli denotacyjnych

- **Poprawność:** Jeśli $M =_\beta N$, to $[\![M]\!] = [\![N]\!]$.
- **Adekwatność:** Jeśli $[\![M]\!] = [\![N]\!]$, to $M \equiv N$.
- **Pełna abstrakcyjność:** Jeśli $M \equiv N$, to $[\![M]\!] = [\![N]\!]$.

Uwaga: Adekwatność to „słaba pełność",
a pełna abstrakcyjność to „silna poprawność".

Trzy własności modeli denotacyjnych

- **Poprawność:** Jeśli $M =_\beta N$, to $[\![M]\!] = [\![N]\!]$.
- **Adekwatność:** Jeśli $[\![M]\!] = [\![N]\!]$, to $M \equiv N$.
- **Pełna abstrakcyjność:** Jeśli $M \equiv N$, to $[\![M]\!] = [\![N]\!]$.

Uwaga: Model $\mathcal{P}_\omega$ jest poprawny i adekwatny, ale nie jest w pełni abstrakcyjny.

Towards a fully abstract model

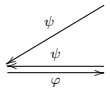
A *projection* of B onto A is a pair of continuous functions

$$\varphi : A \to B \quad \text{and} \quad \psi : B \to A,$$

such that

$$\psi \circ \varphi = \mathrm{id}_A \quad \text{and} \quad \varphi \circ \psi \leq \mathrm{id}_B.$$

Then $\varphi(\bot_A) = \bot_B$, because $\varphi(\bot_A) \leq \varphi(\psi(\bot_B)) \leq \bot_B$.



Example

Let D be any cpo. For example $D = \{\bot, \top\}$. Functions

$$\varphi_0 : D \to [D \to D] \quad \text{i} \quad \psi_0 : [D \to D] \to D,$$

given by

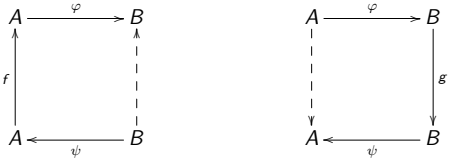
$$\varphi_0(d)(a) = d, \quad \text{oraz} \quad \psi_0(f) = f(\bot)$$

make a projection of $[D \to D]$ onto D.

Raising a projection

Let (φ, ψ) be a projection of B onto A.
Then (φ^*, ψ^*) is a projection of $[B \to B]$ onto $[A \to A]$:

$$\varphi^*(f) = \varphi \circ f \circ \psi \quad \text{and} \quad \psi^*(g) = \psi \circ g \circ \varphi,$$



Towards D_∞

Take any fixed D_0, for instance $D_0 = \{\bot, \top\}$.

Define by induction $D_{n+1} = [D_n \to D_n]$.

Define projections (φ_n, ψ_n) of D_{n+1} onto D_n by induction:

$$(\varphi_{n+1}, \psi_{n+1}) = (\varphi_n^*, \psi_n^*).$$

Two-way transmission:

$$D_0 \xrightarrow{\varphi_0} D_1 \xrightarrow{\varphi_1} D_2 \xrightarrow{\varphi_2} \cdots$$
$$D_0 \xleftarrow{\psi_0} D_1 \xleftarrow{\psi_1} D_2 \xleftarrow{\psi_2} \cdots$$

Ćwiczenie

Jak wyglądają D_0, D_1, D_2?

Scott's D_∞

Thread (nić): ciąg $(x_n)_{n\in\mathbb{N}}$, gdzie $x_n \in D_n$ oraz $x_n = \psi_n(x_{n+1})$.

$$x_0 \xleftarrow{\psi_0} x_1 \xleftarrow{\psi_1} x_2 \xleftarrow{\psi_2} \cdots$$

Denote the set of all threads by D_∞. Ordering:

$$x \leq y \quad \text{iff} \quad \forall n \in \mathbb{N}(x_n \leq y_n).$$

Fact: The set D_∞ is a cpo.

Proof: For directed $X \subseteq D_\infty$ take $X_n = \{x_n \mid x \in X\}$.
Then $(\sup X_n)_n$ is a thread and $(\sup X_n)_n = \sup X$.

Scott's D_∞

For $n < m$ define projections $(\varphi_{nm}, \psi_{nm})$ of D_m onto D_n:

$$\varphi_{nm} = \varphi_{m-1} \circ \cdots \circ \varphi_n, \quad \psi_{nm} = \psi_n \circ \cdots \circ \psi_{m-1}.$$

Define projections $(\varphi_{n\infty}, \psi_{n\infty})$ of D_∞ onto D_n:

$$(\varphi_{n\infty}(x))_i = \begin{cases} \psi_{in}(x), & \text{gdy } i < n; \\ x, & \text{gdy } i = n; \\ \varphi_{ni}(x), & \text{gdy } i > n. \end{cases} \qquad \psi_{n\infty}(y) = y_n.$$

Convention:

$$D_0 \subseteq D_1 \subseteq D_2 \subseteq \cdots \subseteq D_\infty,$$

Element $x \in D_n$ identified with an almost constant thread.

Application

$$x \cdot y = \sup\{x_{n+1}(y_n) \mid n \in \mathbb{N}\}$$

Fact: *Application is a continuous function.*

Proof: One shows continuity wrt both arguments.

N.B. The sequence $x_{n+1}(y_n)$ does not have to form a thread.
But it is monotone: $x_n(y_{n-1}) \leq x_{n+1}(y_n)$, and has a supremum.

Some properties

- Every thread is mononotone: $x_0 \leq x_1 \leq x_2 \leq \ldots$ and $x = \sup x_n$.
- The bottom is unique: $\bot_{D_0} = \bot_{D_n} = \bot_{D_\infty}$.
- If $x \in D_{n+1}$ then $x \cdot y = x(y_n)$.
  If also $y \in D_n$, then $x \cdot y = x(y)$.
- If $y \in D_n$ then $(x \cdot y)_n = x_{n+1}(y)$.
- Always $(x \cdot \bot)_0 = x_0$.

Proofs happily omitted.

Extensionality

Lemma
If $x \cdot z = y \cdot z$, for all z, then $x = y$.

Proof.
One shows that
if $\forall z \in D_\infty (x \cdot z \leq y \cdot z)$ *then* $x_n \leq y_n$, *for all* n.
Begin with $x_0 = (x \cdot \bot)_0 \leq (y \cdot \bot)_0 = y_0$.
Then $x_{n+1}(z) = (x \cdot z)_n \leq (y \cdot z)_n = y_{n+1}(z)$, for $z \in D_n$. □

Scott's D_∞ model

Theorem
The cpo D_∞ is reflexive.

Proof.
Define $F : D_\infty \to [D_\infty \to D_\infty]$ by $F(x)(y) = x \cdot y$.
We know that F is continuous and injective.
Take any $f \in [D_\infty \to D_\infty]$.
Define $f^{(n)} : D_n \to D_n$ by $f^{(n)}(y) = f(y)_n$, for $y \in D_n$.
The sequence $f^{(n)}$ is monotone. Define $G(f) = \sup_n f^{(n)}$.
Then $F(G(f)) = f$. Details omitted. □

Scott's D_∞ model

Corollary
The cpo D_∞ is an extensional lambda-model.
It is isomorphic to $[D_\infty \to D_\infty]$.

Przykład: Co to jest $[\![\lambda x\, x]\!]$?

To taki element e, że $a = e \cdot a$ dla $a \in D_\infty$.

W szczególności dla $a \in D_n$ jest $a = a_n = (e \cdot a)_n = e_{n+1}(a)$.

No to $e_{n+1} = \mathrm{id}_{D_n}$, czyli $e = (\bot, \mathrm{id}, \mathrm{id}, \ldots$

Przykład

Kombinator Y jest interpretowany jako operator najmniejszego punktu stałego, tj.:

$[\![Y]\!] \cdot a$ jest najmniejszym elementem b o własności $a \cdot b = b$.

Wieloznaczność w D_∞

Funkcje ciągłe

$$F : D_\infty \to [D_\infty \to D_\infty] \quad \text{i} \quad G : [D_\infty \to D_\infty] \to D_\infty$$

są wzajemnie odwrotnymi izomorfizmami.

To znaczy, że każdy element $a \in D_\infty$ można utożsamiać z funkcją ciągłą $F(a) \in [D_\infty \to D_\infty]$, a nawet z funkcją

$$F \circ F(a) \circ G \in [[D_\infty \to D_\infty] \to [D_\infty \to D_\infty]]$$

Aplikacja $a \cdot b$ to to samo co $F(a)(b)$ i można pisać $a(b)$.

I to samo co $(F \circ F(a) \circ G)(F(b)) = F(F(a)(G(F(b)))) = F(F(a)(b)) = F(a \cdot b)$.

Twierdzenie (Hyland, Wadsworth)

Następujące warunki są równoważne:

1. *Termy M i N są obserwacyjnie równoważne.*
2. $BT(M) \approx_\eta BT(N)$.
3. $\mathcal{D}_\infty \models M = N$.

Implikacja (1) ⇒ (2) to w istocie twierdzenie Böhma.
Naszkicujemy (3) ⇒ (1) (adekwatność) i (2) ⇒ (3).

Definicje

Niech B, B' będą drzewami Böhma. Napis $B \sqsubseteq B'$ oznacza, że B' powstaje z B przez wstawienie jakichś poddrzew w miejsca, w których w B występuje $\bot$.

Aproksymant to skończone drzewo Böhma (term w postaci normalnej), w którym może występować stała $\bot$).

Przyjmujemy, że $[\![\bot]\!] = \bot_{D_\infty}$

Zbiór *aproksymantów drzewa/termu*:

$A(T) = \{A \mid A \text{ jest aproksymantem oraz } A \sqsubseteq T\}$

$A(M) = A(BT(M))$.

Twierdzenie o aproksymacji

$$[\![M]\!]_\rho = \sup\{[\![A]\!]_\rho \mid A \in A(M)\}.$$

Intuicje (czyli machanie rękami):

Znaczenie $[\![M]\!]_\rho$ to nić $x \in D_\infty$, czyli w istocie ciąg funkcji $x_{n+1} : D_n \to D_n$, opisujących zachowanie $[\![M]\!]_\rho$ na skończonych zbiorach D_n elementów rzędu n.

Dla każdego n to zachowanie jest zdeterminowane przez skończony fragment początkowy drzewa Böhma $BT(M)$.

Tw. o aproksymacji: $[\![M]\!]_\rho = \sup\{[\![A]\!]_\rho \mid A \in A(M)\}$.

Wniosek: *Term M jest rozwiązalny wtedy i tylko wtedy, gdy $[\![M]\!]_\rho \neq \bot$, dla pewnego ρ.*

Dowód: ($\Rightarrow$) Jeśli $M =_\beta \lambda x_1 \ldots x_n.\, y\vec{N}$, gdzie y wolne, to $[\![M]\!]_\varrho \neq \bot$, dla $\rho(y) = \lambda\!\!\lambda \vec{a}.d$, gdzie $d \neq \bot$.

Jeśli $M =_\beta \lambda x_1 \ldots x_n.\, x_i\vec{N}$, to $[\![M]\!]_\rho \neq \bot$, dla każdego ρ. Należy użyć $\lambda\!\!\lambda \vec{a}.d$ jako i-tego argumentu dla $[\![M]\!]_\rho$.

($\Leftarrow$) Wtedy z tw. o aproksymacji istnieje nietrywialny aproksymant, czyli jest czołowa postać normalna.

Adekwatność

Twierdzenie:

Jeśli $\mathcal{D}_\infty \models M = N$, to $M \equiv N$.

Dowód: Jeśli $[\![M]\!]_\rho = [\![N]\!]_\rho$ to także $[\![C[M]]\!]_\rho = [\![C[N]]\!]_\rho$.

Jeśli jedno jest różne od $\bot$, to i drugie. Zatem jeśli jedno rozwiązalne to i drugie.

Pełna abstrakcyjność

Twierdzenie: *Jeśli $M \equiv N$, to $[\![M]\!] = [\![N]\!]$*

Szkic dowodu: Skoro $M \equiv N$, to $BT(M) \approx_\eta BT(N)$. (To już wiemy.)

Te drzewa są różne ale ich aproksymanty są z grubsza takie same (z dokładnością do η-konwersji).

Zatem $[\![M]\!] = [\![N]\!]$ wynika z twierdzenia o aproksymacji.

Typy

Typy: motywacja semantyczna

Typ = podzbiór modelu (własność jego elementów)

- Trivial property – the whole domain $\mathcal{D}$, denoted by ω;
- Intersection $\sigma \cap \tau$ of properties σ and τ;
- Function space:
  $\sigma \Rightarrow \tau = \{a \in D \mid \forall b \in D(b \in \sigma \to a \cdot b \in \tau)\}$.

Subtyping in a model

$$\sigma \cap \tau \subseteq \sigma \qquad \sigma \cap \tau \subseteq \tau \qquad \sigma \subseteq \sigma \cap \sigma$$

$$\sigma \subseteq \omega \qquad \omega \subseteq \omega \Rightarrow \omega$$

$$(\sigma \Rightarrow \tau) \cap (\sigma \Rightarrow \rho) \subseteq \sigma \Rightarrow \tau \cap \rho$$

If $\sigma \subseteq \sigma'$ and $\tau \subseteq \tau'$ then

$$\sigma \cap \tau \subseteq \sigma' \cap \tau' \qquad \sigma' \Rightarrow \tau \subseteq \sigma \Rightarrow \tau'$$

Formal type assignment

A *type assignment system* derives judgements of the form

$$\Gamma \vdash M : \tau,$$

where M is a λ-term, τ is a type,
and Γ is a set of type declarations for free variables.

Intersection types (formal system)

Types:

- The constant ω is a type.
- Type variables $p, q, \cdots \in TV$ are types. (*)
- If σ and τ are types then $(\sigma \to \tau)$ is a type. (*)
- If σ and τ are types then $(\sigma \cap \tau)$ is a type.

Klauzule (*) definiują *typy proste*.

Assumption:
Intersection is commutative, associative and idempotent.

Convention: Arrow is "right-associative", that is,

$\tau \to (\sigma \to \rho)$ is written $\tau \to \sigma \to \rho$.

Subtyping (BCD)

Define $\leq$ as the least quasi-order satisfying the axioms:

$$\sigma \cap \tau \leq \sigma \qquad \sigma \cap \tau \leq \tau \qquad \sigma \leq \sigma \cap \sigma$$

$$\sigma \leq \omega \qquad \omega \leq \omega \to \omega$$

$$(\sigma \to \tau) \cap (\sigma \to \rho) \leq \sigma \to \tau \cap \rho$$

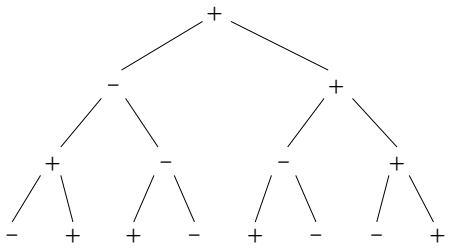
and closed under the rules:

$$\frac{\sigma \leq \sigma' \qquad \tau \leq \tau'}{\sigma \cap \tau \leq \sigma' \cap \tau'} \qquad \frac{\sigma \leq \sigma' \qquad \tau \leq \tau'}{\sigma' \to \tau \leq \sigma \to \tau'}$$

Przykład: $p \cap (a \to b) \cap (c \to d) \leq a \cap c \cap p \to b$.

Napis $\sigma \equiv \tau$ oznacza $\sigma \leq \tau \leq \sigma$.
Takie typy czasem się utożsamia.

Positive and Negative



Monotoniczność: Jeśli p występuje w φ tylko pozytywnie, oraz $\tau \leq \sigma$, to $\varphi(\tau) \leq \varphi(\sigma)$. A jak tylko negatywnie?

Environments

An *environment* (also called "context" or "base" or...)
is a finite partial function $\Gamma : Var \to Type$, identified
with the set of pairs $x : \Gamma(x)$, for $x \in \mathrm{Dom}(\Gamma)$.

Assume $\Gamma(x) = \omega$, for $x \notin \mathrm{Dom}(\Gamma)$.

Write $\Gamma(x : \tau)$ for the environment Γ' such that

$\Gamma'(x) = \tau$ and otherwise $\Gamma'(y) = \Gamma(y)$.

Write $\Gamma_1 \& \Gamma_2$ for the environment Γ'' such that

$$\Gamma''(x) = \Gamma_1(x) \cap \Gamma_2(x).$$

Przykład: $\Gamma_1 = \{(x : \tau), (y : \sigma)\}$, $\Gamma_2 = \{(x : \rho), (z : \tau)\}$.
Wtedy $\Gamma_1(x : \sigma) \& \Gamma_2 = \{(x : \sigma \cap \rho), (y : \sigma), (z : \tau)\}$.

Intersection type assignment (BCD)

$$\Gamma(x : \sigma) \vdash x : \sigma \ (\mathrm{Var}) \qquad \Gamma \vdash M : \omega \ (\omega)$$

$$\frac{\Gamma(x : \sigma) \vdash M : \tau}{\Gamma \vdash \lambda x\, M : \sigma \to \tau} \ (\mathrm{Abs}) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \ (\mathrm{App})$$

$$\frac{\Gamma \vdash M : \tau_1 \quad \Gamma \vdash M : \tau_2}{\Gamma \vdash M : \tau_1 \cap \tau_2} \ (\cap \mathrm{I}) \qquad \frac{\Gamma \vdash M : \tau_1 \cap \tau_2}{\Gamma \vdash M : \tau_i} \ (\cap \mathrm{E})$$

$$\frac{\Gamma \vdash M : \sigma \quad [\sigma \leq \tau]}{\Gamma \vdash M : \tau} \ (\leq)$$

BCD = Barendregt, Coppo, Dezani

Example

$$\frac{\Gamma(x : \sigma) \vdash M : \tau}{\Gamma \vdash \lambda x\, M : \sigma \to \tau} \ (\mathrm{Abs}) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \ (\mathrm{App})$$

$$\dfrac{\dfrac{\dfrac{z : \tau, x : \tau, y : \sigma \vdash x : \tau}{z : \tau, x : \tau \vdash \lambda y\, x : \sigma \to \tau}}{z : \tau \vdash \lambda xy.\, x : \tau \to \sigma \to \tau} \qquad z : \tau \vdash z : \tau}{z : \tau \vdash (\lambda xy.\, x) z : \sigma \to \tau}$$

Example

$$\frac{\Gamma(y : \sigma) \vdash M : \tau}{\Gamma \vdash \lambda y\, M : \sigma \to \tau} \ (\mathrm{Abs}) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \ (\mathrm{App})$$

$$\dfrac{\dfrac{\dfrac{x : \tau, y : \sigma \vdash x : \tau}{x : \tau, y : \tau \vdash \lambda y\, x : \sigma \to \tau}}{y : \tau \vdash \lambda xy.\, x : \tau \to \sigma \to \tau} \qquad y : \tau \vdash y : \tau}{y : \tau \vdash (\lambda xy.\, x) y : \sigma \to \tau}$$

Explanation: $\{(x : \tau, y : \tau)\}(y : \sigma) = \{(x : \tau, y : \sigma)\}$

Example

$$\frac{\Gamma \vdash M : \tau_1 \quad \Gamma \vdash M : \tau_2}{\Gamma \vdash M : \tau_1 \cap \tau_2}\ (\cap\mathrm{I}) \qquad \frac{\Gamma \vdash M : \tau_1 \cap \tau_2}{\Gamma \vdash M : \tau_i}\ (\cap\mathrm{E})$$

$$\dfrac{\dfrac{\dfrac{x : p \cap (p \to q) \vdash x : p \cap (p \to q)}{x : p \cap (p \to q) \vdash x : p} \quad \dfrac{x : p \cap (p \to q) \vdash x : p \cap (p \to q)}{x : p \cap (p \to q) \vdash x : p \to q}}{x : p \cap (p \to q) \vdash xx : q}}{\vdash \lambda x.\, xx : p \cap (p \to q) \to q}$$

Example

Aksjomat: $\Gamma \vdash M : \omega$

$$\dfrac{\dfrac{\dfrac{\vdots}{x : p \vdash \mathsf{K} : p \to \omega \to p} \quad x : p \vdash x : p}{x : p \vdash \mathsf{K}x : \omega \to p} \quad x : p \vdash \mathbf{\Omega} : \omega}{x : p \vdash \mathsf{K}x\mathbf{\Omega} : p}$$

Example

$$\frac{\Gamma \vdash M : \sigma \quad [\sigma \le \tau]}{\Gamma \vdash M : \tau}\ (\le)$$

Let $\Gamma = \{y : (p \cap q \to r) \to r,\ \ x : p \to r \cap s\}$.

$$\dfrac{\dfrac{\Gamma \vdash y : (p \cap q \to r) \to r}{\Gamma \vdash y : (p \to r) \to r} \quad \dfrac{\Gamma \vdash x : p \to r \cap s}{\Gamma \vdash x : p \to r}}{\Gamma \vdash yx : r}$$

Example

$$\dfrac{\dfrac{\vdots}{\vdash \mathsf{K} : (p \to q \to p) \to r \to p \to q \to p} \quad \dfrac{\vdots}{\vdash \mathsf{K} : p \to q \to p}}{\vdash \mathsf{KK} : r \to p \to q \to p}$$

Examples

- $\vdash \mathsf{I} : t \cap s \to t$;
- $\vdash \mathsf{I} : t \to t$;
- $\vdash 2 : (t \to s) \cap (s \to r) \to (t \to r)$;
- $\vdash 2 : (t \to t) \to t \to t$;
- $\vdash \mathsf{K} : t \to s \to t$;
- $\vdash \mathsf{S} : (t' \to s \to r) \to (t'' \to s) \to (t' \cap t'') \to r$;
- $\vdash \mathsf{S} : (t \to s \to r) \to (t \to s) \to t \to r$.

"Beta-soundness"

Lemma
Jeśli $\sigma \to \tau \not\equiv \omega$ *oraz* $\bigcap_{j\in J} p_j \ \cap\ \bigcap_{i\in I}(\sigma_i \to \tau_i) \le \sigma \to \tau$ *to* $\{i \mid \sigma \le \sigma_i\} \neq \varnothing$ *oraz* $\bigcap_{\sigma\le\sigma_i} \tau_i \le \tau$.

Proof.
Indukcja ze względu na definicję $\le$. □

Intersection type assignment (BCD)

$$\Gamma, (x : \sigma) \vdash x : \sigma\ (\mathrm{Var}) \qquad \Gamma \vdash M : \omega\ (\omega)$$

$$\frac{\Gamma, (x : \sigma) \vdash M : \tau}{\Gamma \vdash \lambda x\, M : \sigma \to \tau}\ (\mathrm{Abs}) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ (\mathrm{App})$$

$$\frac{\Gamma \vdash M : \tau_1 \quad \Gamma \vdash M : \tau_2}{\Gamma \vdash M : \tau_1 \cap \tau_2}\ (\cap\mathrm{I}) \qquad \frac{\Gamma \vdash M : \tau_1 \cap \tau_2}{\Gamma \vdash M : \tau_i}\ (\cap\mathrm{E})$$

$$\frac{\Gamma \vdash M : \sigma \quad [\sigma \le \tau]}{\Gamma \vdash M : \tau}\ (\le)$$

Generation (Inversion) Lemma

- *If* $\Gamma \vdash x : \sigma$ *then* $\Gamma(x) \le \sigma$.
- *If* $\Gamma \vdash MN : \sigma$ *then* $\Gamma \vdash M : \tau \to \sigma$ *and* $\Gamma \vdash N : \tau$.
- *If* $\Gamma \vdash \lambda x.M : \sigma$ *then* $\sigma = \bigcap_i(\sigma_i \to \tau_i)$.
  (If $\sigma \not\equiv \omega$ *then* $\tau_i \not\equiv \omega$.*)*
- *If* $\Gamma \vdash \lambda x.M : \sigma \to \tau$ *then* $\Gamma, (x : \sigma) \vdash M : \tau$.

Proof: Induction wrt the number of nodes in the type derivation. By cases depending on the last rule used.

Przypadek aplikacji

- *If* $\Gamma \vdash MN : \sigma$ *then* $\Gamma \vdash M : \tau \to \sigma$ *and* $\Gamma \vdash N : \tau$.

Proof: Induction wrt type derivation.

Example induction step: Last rule used is ($\cap$I).
We have $\sigma = \sigma_1 \cap \sigma_2$ and $\Gamma \vdash MN : \sigma_1$, and $\Gamma \vdash MN : \sigma_2$, with smaller derivations. Apply the induction hypothesis:

$$\Gamma \vdash M : \tau_1 \to \sigma_1, \quad \Gamma \vdash N : \tau_1$$
$$\Gamma \vdash M : \tau_2 \to \sigma_2, \quad \Gamma \vdash N : \tau_2.$$

Then $\Gamma \vdash M : (\tau_1 \to \sigma_1) \cap (\tau_2 \to \sigma_2)$ and $\Gamma \vdash N : \tau_1 \cap \tau_2$.

But $(\tau_1 \to \sigma_1) \cap (\tau_2 \to \sigma_2) \leq (\tau_1 \cap \tau_2 \to \sigma_1) \cap (\tau_1 \cap \tau_2 \to \sigma_2)$
$\leq \tau_1 \cap \tau_2 \to \sigma_1 \cap \sigma_2$, whence $\Gamma \vdash M : \tau_1 \cap \tau_2 \to \sigma_1 \cap \sigma_2$.

Correctness of substitution

Lemma
If $\Gamma(x : \sigma) \vdash M : \tau$ *and* $\Gamma \vdash N : \sigma$, *then* $\Gamma \vdash M[x := N] : \tau$.

Proof.
Induction wrt M. For example, if $M = \lambda y\, P$ and $\tau = \mu \to \rho$ then $\Gamma(x : \sigma)(y : \mu) \vdash P : \rho$ by the inversion lemma.
By the induction hypothesis $\Gamma(y : \mu) \vdash P[x := N] : \rho$,
whence $\Gamma \vdash \lambda y\, P[x := N] : \mu \to \rho$. □

Subject Reduction

Theorem: *If* $\Gamma \vdash M : \tau$ *and* $M \to_{\beta\eta} N$ *then* $\Gamma \vdash N : \tau$.

Proof: Induction wrt definition of $\to_{\beta\eta}$. Main cases:

$M = (\lambda x.P)Q \to_\beta P[x := Q] = N$.

By inversion $\Gamma \vdash \lambda x.P : \sigma \to \tau$ and $\Gamma \vdash Q : \sigma$.
Then $\Gamma, x : \sigma \vdash P : \tau$, whence $\Gamma \vdash P[x := Q] : \tau$

$M = \lambda x.Nx \to_\eta N$.

Then $\tau = \bigcap_i(\sigma_i \to \tau_i)$ and $\Gamma, x : \sigma_i \vdash Nx : \tau_i$, all i.
It follows that $\Gamma, x : \sigma_i \vdash N : \zeta_i \to \tau_i$ with $\sigma_i \leq \zeta_i$.
But $x \notin FV(N)$, so $\Gamma \vdash N : \bigcap_i(\zeta_i \to \tau_i) \leq \tau$.

Subject Conversion, czyli na odwrót też

Lemma: *Let* $\Gamma \vdash M[x := N] : \tau$. *Then there is* σ *with*
$\Gamma \vdash N : \sigma$ *and* $\Gamma(x : \sigma) \vdash M : \tau$.

Proof: Induction wrt M. In case $M = y \neq x$ take $\sigma = \omega$.

Wtedy $\Gamma \vdash N : \omega$ za darmo, oraz $\Gamma(x : \omega) \vdash y : \tau$.

Jeśli $M = x$, to $x[x := N] = N$ i jako σ bierzemy τ.

Lemma: *Let* $\Gamma \vdash M[x := N] : \tau$. *Then there is* σ *with*
$\Gamma \vdash N : \sigma$ *and* $\Gamma(x : \sigma) \vdash M : \tau$.

Proof: Induction wrt M. W przypadku $M = M_1M_2$,
z lematu o generowaniu istnieje takie ρ, że:

$\Gamma \vdash M_1[x := N] : \rho \to \tau$ oraz $\Gamma \vdash M_2[x := N] : \rho$.

Z zał. ind. mamy takie σ_1, σ_2, że:

$$\Gamma \vdash N : \sigma_1, \quad \Gamma(x : \sigma_1) \vdash M_1 : \rho \to \tau,$$
$$\Gamma \vdash N : \sigma_2, \quad \Gamma(x : \sigma_2) \vdash M_2 : \rho.$$

Można więc przyjąć $\sigma = \sigma_1 \cap \sigma_2$.

Lemma: *Let* $\Gamma \vdash M[x := N] : \tau$. *Then there is* σ *with*
$\Gamma \vdash N : \sigma$ *and* $\Gamma(x : \sigma) \vdash M : \tau$.

Proof: Induction wrt M. Niech $M = \lambda y\, P$.

Z lematu o generowaniu: $\tau = \bigcap_{i \in I}(\alpha_i \to \beta_i)$,
gdzie $\Gamma(y : \alpha_i) \vdash P[x := N] : \beta_i$, dla $i \in I$.

Są σ_i, że $\Gamma(y : \alpha_i) \vdash N : \sigma_i$ oraz $\Gamma(y : \alpha_i)(x : \sigma_i) \vdash P : \beta_i$.

Niech $\sigma = \bigcap_{i \in I} \sigma_i$. Wtedy $\Gamma(y : \alpha_i)(x : \sigma) \vdash P : \beta_i$,
skąd $\Gamma(x : \sigma) \vdash \lambda y\, P : \alpha_i \to \beta_i$ i dobrze.

No zaraz, ale dlaczego $\Gamma \vdash N : \sigma$*?*

Subject Conversion

Theorem: *If* $M \to_\beta N$ *then in system* BCD*:*
$\Gamma \vdash M : \tau$ *iff* $\Gamma \vdash N : \tau$.

Dowód: Najważniejszy przypadek wynika z poprzedniego lematu: jeśli $\Gamma \vdash M[x := N] : \tau$, to istnieje takie σ, że $\Gamma \vdash N : \sigma$ oraz $\Gamma(x : \sigma) \vdash M : \tau$. Wtedy $\Gamma \vdash (\lambda x\, M)N : \tau$.

N.B. To nie działa dla eta-redukcji: $x : p \not\vdash \lambda y.\, xy : p$

Interpreting types in a model

Type valuation $\xi : TV \to \mathrm{P}(\mathcal{D})$ assigns sets to type variables.

Interpretation of types:

- $[\![p]\!]_\xi = \xi(p)$, for type variable p;
- $[\![\omega]\!]_\xi = \mathcal{D}$;
- $[\![\sigma \cap \tau]\!]_\xi = [\![\sigma]\!]_\xi \cap [\![\tau]\!]_\xi$;
- $[\![\sigma \to \tau]\!]_\xi = [\![\sigma]\!]_\xi \Rightarrow [\![\tau]\!]_\xi$.

Easy lemma: If $\sigma \leq \tau$, then $[\![\sigma]\!]_\xi \subseteq [\![\tau]\!]_\xi$.

Semantics of type assignment

Write $\mathcal{D}, \nu, \xi \models M : \sigma$ when $[\![M]\!]_\nu \in [\![\sigma]\!]_\xi$.

Write $\mathcal{D}, \nu, \xi \models \Gamma$ when $\nu(x) \in [\![\Gamma(x)]\!]_\xi$, for all $x \in \mathrm{Dom}(\Gamma)$.

Write $\Gamma \models M : \sigma$ when, for all $\mathcal{D}, \nu, \xi$,

$\mathcal{D}, \nu, \xi \models \Gamma$ implies $\mathcal{D}, \nu, \xi \models M : \sigma$,

Theorem (Soundness)

If $\Gamma \vdash M : \sigma$ *then* $\Gamma \models M : \sigma$.

Proof: Easy induction wrt type derivation.

Filter model

Definition: A set F of types is a *filter* when:

- F is not empty;
- If $\sigma, \tau \in F$ then $\sigma \cap \tau \in F$;
- If $\sigma \in F$ and $\sigma \leq \tau$ then $\tau \in F$.

Example: *filtr główny (principal filter)*

$$\sigma\uparrow = \{\tau \mid \sigma \leq \tau\}.$$

Application: $F_1 \cdot F_2 = \{\tau \mid \sigma \to \tau \in F_1, \text{ for some } \sigma \in F_2\}$

Lemma: *If F_1 and F_2 are filters then $F_1 \cdot F_2$ is a filter.*

Proof: (1) Not empty, because $\omega \leq \omega \to \omega$, and $\omega \in F_1, F_2$.

(2) Let $\tau_1, \tau_2 \in F_1 \cdot F_2$. There are $\sigma_1 \to \tau_1, \sigma_2 \to \tau_2 \in F_1$ and $\sigma_1, \sigma_2 \in F_2$. Then $\sigma_1 \cap \sigma_2 \in F_2$ and

$$(\sigma_1 \to \tau_1) \cap (\sigma_2 \to \tau_2) \leq \sigma_1 \cap \sigma_2 \to \tau_1 \cap \tau_2 \in F_1.$$

Hence $\tau_1 \cap \tau_2 \in F_1 \cdot F_2$.

(3) If $\sigma \to \tau \in F_1$ and $\tau \leq \tau'$ then $\sigma \to \tau' \in F_1$.

Model z filtrów: znaczenie termu

Idea: Znaczenie termu, to zbiór wszystkich typów tego termu.

Ale term ma zmienne wolne.
Jego znaczenie zależy od wartościowania.

Otoczenie Γ: zmienne obiektowe $\longrightarrow$ typy
Wartościowanie ν: zmienne obiektowe $\longrightarrow$ filtry

An environment Γ is *consistent* with object valuation ν when

$$\Gamma(x) \in \nu(x), \text{ for all } x \in \mathrm{Dom}(\Gamma).$$

Interpretation:

$$[\![M]\!]_\nu = \{\sigma \mid \Gamma \vdash M : \sigma, \text{ for some } \Gamma \text{ consistent with } \nu\}.$$

Filter model

$$F_1 \cdot F_2 = \{\tau \mid \sigma \to \tau \in F_1 \text{ for some } \sigma \in F_2\}.$$
$$[\![M]\!]_\nu = \{\sigma \mid \Gamma \vdash M : \sigma, \text{ for some } \Gamma \text{ consistent with } \nu\}.$$

Lemma: *The filter model $\mathcal{F}$ is a lambda-model:*

(a) $[\![x]\!]_\nu = \nu(x)$;
(b) $[\![PQ]\!]_\nu = [\![P]\!]_\nu \cdot [\![Q]\!]_\nu$;
(c) $[\![\lambda x.P]\!]_\nu \cdot F = [\![P]\!]_{\nu[x \mapsto F]}$, *for any* $F \in \mathcal{F}$;
(d) *If* $\nu|_{\mathrm{FV}(P)} = u|_{\mathrm{FV}(P)}$, *then* $[\![P]\!]_\nu = [\![P]\!]_u$.
(e) *If* $[\![\lambda x.M]\!]_\nu \approx [\![\lambda x.N]\!]_\nu$ *then* $[\![\lambda x.M]\!]_\nu = [\![\lambda x.N]\!]_\nu$.

Dowód opuszczamy.

Model z filtrów: znaczenie typu

Lemma: Let $\xi(p) = \{F \mid F \text{ filter, and } p \in F\}$. Then for all τ

$$[\![\tau]\!]_\xi = \{F \mid F \text{ filter, and } \tau \in F\}.$$

Inaczej: $\tau \in F \;\Leftrightarrow\; F \in [\![\tau]\!]_\xi$.

Dowód opuszczamy

Completeness of type assignment

Theorem: *If* $\Gamma \models M : \sigma$ *then* $\Gamma \vdash M : \sigma$.

Proof: Let $\xi(p) = \{F \mid p \in F\}$ and $\nu(x) = \Gamma(x)\uparrow$.

Then $\Gamma(x) \in \nu(x)$, thus $\nu(x) \in [\![\Gamma(x)]\!]_\xi$.

That is, $\mathcal{F}, \nu, \xi \models \Gamma$, whence $\mathcal{F}, \nu, \xi \models M : \sigma$.

Therefore $[\![M]\!]_\nu \in [\![\sigma]\!]_\xi$, i.e., $\sigma \in [\![M]\!]_\nu$.

There is Γ', consistent with ν, such that $\Gamma' \vdash M : \sigma$.

We have $\Gamma'(x) \in \nu(x) = \Gamma(x)\uparrow$, whence $\Gamma(x) \leq \Gamma'(x)$.

It follows that $\Gamma \vdash M : \sigma$.

Jeszcze dwa twierdzenia

Twierdzenie:
Następujące warunki są równoważne (dla systemu BCD):

- $\mathrm{BT}(M) = \mathrm{BT}(N)$;
- Termy M i N mają te same typy w każdym otoczeniu.

Twierdzenie: Term zamknięty ma ma typ τ w BCD wtedy i tylko wtedy, gdy jakiś jego aproksymant ma typ τ.

Intersection types without omega

$$\Gamma, (x:\sigma) \vdash x : \sigma \ \text{(Var)}$$

$$\frac{\Gamma, (x:\sigma) \vdash M : \tau}{\Gamma \vdash \lambda x\, M : \sigma \to \tau}\ \text{(Abs)} \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ \text{(App)}$$

$$\frac{\Gamma \vdash M : \tau_1 \quad \Gamma \vdash M : \tau_2}{\Gamma \vdash M : \tau_1 \cap \tau_2}\ (\cap\text{I}) \qquad \frac{\Gamma \vdash M : \tau_1 \cap \tau_2}{\Gamma \vdash M : \tau_i}\ (\cap\text{E})$$

$$\frac{\Gamma \vdash M : \sigma \quad [\sigma \le \tau]}{\Gamma \vdash M : \tau}\ (\le)$$

Intersection types without subtyping

$$\Gamma, (x:\sigma) \vdash x : \sigma \ \text{(Var)}$$

$$\frac{\Gamma, (x:\sigma) \vdash M : \tau}{\Gamma \vdash \lambda x\, M : \sigma \to \tau}\ \text{(Abs)} \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ \text{(App)}$$

$$\frac{\Gamma \vdash M : \tau_1 \quad \Gamma \vdash M : \tau_2}{\Gamma \vdash M : \tau_1 \cap \tau_2}\ (\cap\text{I}) \qquad \frac{\Gamma \vdash M : \tau_1 \cap \tau_2}{\Gamma \vdash M : \tau_i}\ (\cap\text{E})$$

With or without subtyping

Theorem
A type judgment is derivable in the system with rule $(\le)$
if and only if it is derivable in the system with rule

$$(\eta)\ \frac{\Gamma \vdash M : \sigma, \quad M \to_\eta N}{\Gamma \vdash N : \sigma}$$

Example: $x : (\alpha \to \beta) \cap (\alpha \to \gamma) \ \vdash \ \lambda y.\, xy : \alpha \to (\beta \cap \gamma)$.

No omega and no subtyping

Generation (Inversion) Lemma:

- *If* $\Gamma \vdash x : \sigma$ *then* $\Gamma(x) = \sigma \cap \tau$.
- *If* $\Gamma \vdash MN : \sigma$ *then* $\sigma = \bigcap_i \sigma_i$, *where* $\Gamma \vdash M : \bigcap_i (\tau_i \to \sigma_i)$ *and* $\Gamma \vdash N : \bigcap_i \tau_i$.
- *If* $\Gamma \vdash \lambda x.M : \sigma$ *then* $\sigma = \bigcap_i (\sigma_i \to \tau_i)$, *where* $\Gamma, (x : \sigma_i) \vdash M : \tau_i$.

Lemma:
If $\Gamma, x : \sigma \vdash M : \tau$ and $\Gamma \vdash N : \sigma$, then $\Gamma \vdash M[x := N] : \tau$.

Subject Reduction (bez ω i $\le$)

Theorem: *If* $\Gamma \vdash M : \tau$ *and* $M \to_\beta N$ *then* $\Gamma \vdash N : \tau$.

Remark: No more subject conversion, e.g., $\mathsf{KI}\Omega \to \mathsf{I}$.

Remark: No more subject reduction for η, e.g.,

$x : (\alpha \to \beta) \cap (\alpha \to \gamma) \ \vdash \ \lambda y.\, xy : \alpha \to (\beta \cap \gamma)$
$x : (\alpha \to \beta) \cap (\alpha \to \gamma) \ \nvdash \ x : \alpha \to (\beta \cap \gamma)$

Silna normalizacja

Termy typowalne nie mają nieskończonych redukcji

Strong normalization: Tait's proof method

Definition: *Stable (computable,reducible...)* terms:

- $[\![p]\!] := \mathrm{SN}$;
- $[\![\tau \to \sigma]\!] := \{M \mid \forall N (N \in [\![\tau]\!] \Rightarrow MN \in [\![\sigma]\!])\}$;
- $[\![\tau \cap \sigma]\!] := [\![\tau]\!] \cap [\![\sigma]\!]$.

Lemma: If $\tau \le \sigma$, then $[\![\tau]\!] \subseteq [\![\sigma]\!]$.

Lemma:

1) $[\![\tau]\!] \subseteq \mathrm{SN}$;
2) If $N_1, \ldots, N_k \in \mathrm{SN}$ then $xN_1 \ldots N_k \in [\![\tau]\!]$.
   In particular, variables are stable for every type.

Proof: Induction wrt τ. If $\tau = p$ then (1) is immediate. Also (2), because $xN_1 \ldots N_k \in \mathrm{SN}$.

For $\tau = \sigma \cap \rho$ apply induction. Let $\tau = \sigma \to \rho$.

(1) Take $M \in [\![\sigma \to \rho]\!]$. By the ind. hyp. (2), we have $x \in [\![\sigma]\!]$, so that $Mx \in [\![\rho]\!]$. Thus $Mx \in \mathrm{SN}$, by the ind. hyp. (1). Hence $M \in \mathrm{SN}$.

(2) We want $xN_1 \ldots N_k P \in [\![\rho]\!]$, whenever $P \in [\![\sigma]\!]$. But $P \in \mathrm{SN}$ by the ind. hyp. (1), and we can apply ind. hyp. (2) to ρ.

Lemma: Let $M[x{:=}N_0]N_1 \dots N_k \in \mathrm{SN}$ and $N_0 \in \mathrm{SN}$.
Then also $(\lambda x.M)N_0N_1 \dots N_k \in \mathrm{SN}$. (To już było.)

Lemma: Let $M[x := N_0]N_1 \dots N_k \in [\![\tau]\!]$ oraz $N_0 \in \mathrm{SN}$.
Then also $(\lambda x.M)N_0N_1 \dots N_k \in [\![\tau]\!]$.

Proof: Induction wrt τ. Base case above.

For $\tau = \sigma \cap \rho$ apply induction. Let $\tau = \sigma \to \rho$.

Let $M[x := N_0]N_1 \dots N_k \in [\![\sigma \to \rho]\!]$ and $P \in [\![\sigma]\!]$.
We want $(\lambda x.M)N_0N_1 \dots N_kP \in [\![\rho]\!]$.

Then $M[x := N_0]N_1 \dots N_kP \in [\![\rho]\!]$.
Apply induction to ρ with $N_{k+1} = P$.

Lemma: Let $\Gamma \vdash M : \tau$ and $N_i \in [\![\Gamma(x_i)]\!]$, for $i \leq n$.
Then $M[x_1 := N_1, \dots, x_n := N_n] \in [\![\tau]\!]$.

Proof: Induction wrt derivation of $\Gamma \vdash M : \tau$.

Example case: $\Gamma \vdash \lambda y\, P : \sigma \to \rho$, because $\Gamma(y : \sigma) \vdash P : \rho$.

We want $(\lambda y\, P)[\vec{x} := \vec{N}] \in [\![\sigma \to \rho]\!]$. Let $Q \in [\![\sigma]\!]$.

By ind. hyp., $P[\vec{x} := \vec{N}][y := Q] = P[\vec{x} := \vec{N}, y := Q] \in [\![\rho]\!]$,

so $(\lambda y.P[\vec{x} := \vec{N}])Q \in [\![\rho]\!]$, by the previous lemma.

That's what we need.

(Uwaga: $y \notin \mathrm{FV}(\vec{N})$.)

Strong Normalization

Theorem: If $\Gamma \vdash M : \tau$ then $M \in \mathrm{SN}$.

Proof: Let $\mathrm{FV}(M) = \vec{x}$. Variables are stable, so we have

$$M = M[\vec{x} := \vec{x}] \in [\![\tau]\!] \subseteq \mathrm{SN}.$$

Dygresja: sens moralny numeracji gödlowskiej

Natural number – prototype of finite object.

Peano Arithmetic – prototype of finitary reasoning.

Gödel's Incompleteness Theorem: There are arithmetical statements unprovable in Peano Arithmetic.

Strong Normalization

For every typable lambda-term M,
every reduction sequence of M is finite.

Using König's Lemma:

For every typable lambda-term M, there exists k such that every reduction sequence of M terminates in at most k steps.

Arithmetization:

For every number n of a type derivation for a lambda-term M there exists m such that every number of a finite reduction sequence of M is less than m.

Arithmetization of Tait's proof?

Bad News: The definition of stability is by induction:

- $[\![p]\!] = \mathrm{SN}$;
- $[\![\tau \to \sigma]\!] = \{M \mid \forall N(N \in [\![\tau]\!] \Rightarrow MN \in [\![\sigma]\!])\}$;
- $[\![\tau \cap \sigma]\!] = [\![\tau]\!] \cap [\![\sigma]\!]$.

Formalizing this definition requires quantification over sets.

(Definiujemy relację „$M \in [\![\tau]\!]$" jako najmniejszą relację o własnościach jw.)

Therefore Tait's proof is not arithmetical.

Good News: There are other proofs which are arithmetical (for the system of intersection types).

Typability

Definition
A term M is *typable* iff $\Gamma \vdash M : \tau$, for some Γ and τ.

Question: Which exactly terms are typable?

Lemma
Every normal form is typable in intersection types.

Proof: Ćwiczenie.

Typy proste

Simple types (Curry style)

Types:

- Zmienne i/lub stałe typowe, np. jedna stała 0.
- If σ and τ are types then $(\sigma \to \tau)$ is a type.

Konwencja:

- Zamiast $(\tau \to (\sigma \to \rho))$ piszemy $\tau \to \sigma \to \rho$.

Każdy typ ma postać $\tau_1 \to \cdots \to \tau_n \to atom$.

Simple type assignment

$$\Gamma\,(x:\sigma) \vdash x : \sigma \ \text{(Var)}$$

$$\frac{\Gamma(x:\sigma) \vdash M : \tau}{\Gamma \vdash \lambda x\, M : \sigma \to \tau}\ \text{(Abs)}$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ \text{(App)}$$

Note: This is a syntax-oriented system.

Generation (Inversion) Lemma

- *If* $\Gamma \vdash x : \sigma$ *then* $\Gamma(x) = \sigma$.
- *If* $\Gamma \vdash MN : \sigma$ *then* $\Gamma \vdash M : \tau \to \sigma$ *and* $\Gamma \vdash N : \tau$.
- *If* $\Gamma \vdash \lambda x.M : \sigma$ *then* $\sigma = \rho \to \tau$ *and* $\Gamma(x:\rho) \vdash M : \tau$.

Proof: Can't be simpler.

Note: In a derivation of $\Gamma \vdash M : \tau$,
every subterm of M is assigned a unique type.

Subject Reduction

Theorem: *If* $\Gamma \vdash M : \tau$ *and* $M \to_{\beta\eta} N$ *then* $\Gamma \vdash N : \tau$.

Proof: Easy induction or appeal to intersection types.

Church vs. Curry

Curry style (type-assignment systems):

- Ordinary untyped lambda-terms.
- Types are derivable properties of terms.
- System of type assignment rules.
- A term may have many types or none.
- Typability not obvious.

Church style (typed systems):

- New syntax, built-in types.
- Every term has exactly one type.
- No "untypable" terms.

Church style syntax (orthodox)

Assume infinite sets V_τ of variables of each type τ.

Define sets T_τ of terms of type τ:

- A variable of type τ is a term of type τ;
- If $M \in T_{\sigma\to\tau}$ and $N \in T_\sigma$ then $(MN) \in T_\tau$;
- If $M \in T_\tau$ and $x \in V_\sigma$ then $(\lambda x\, M) \in T_{\sigma\to\tau}$.

Write M^σ for $M \in T_\sigma$ and define beta-reduction by

$$(\lambda x^\sigma.\, M^\tau)N^\sigma \Rightarrow M[x^\sigma := N] \in T_\tau.$$

Non-orthodox Church

Type-assignment with type annotations on bound variables.

$$\Gamma\,(x:\sigma) \vdash x : \sigma \ \text{(Var)}$$

$$\frac{\Gamma(x:\sigma) \vdash M : \tau}{\Gamma \vdash \lambda x{:}\sigma\, M : \sigma \to \tau}\ \text{(Abs)}$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ \text{(App)}$$

Fact: If $\Gamma \vdash M : \tau$ and $\Gamma \vdash M : \sigma$ then $\tau = \sigma$.

Relating systems

Orthodox Church terms are morally the same as:

- Non-orthodox terms in a fixed infinite environment.
- Curry-style type derivations.

Wycieranie typów:

Erasing types from (non-orthodox) Church terms:

- $|x| = x$;
- $|MN| = |M||N|$;
- $|\lambda x{:}\sigma.\, M| = \lambda x\, |M|$.

Properties of type erasure

(Church is blue, Curry is green.)

1. If $\Gamma \vdash \mathrm{M} : \tau$ then $\Gamma \vdash |\mathrm{M}| : \tau$.
2. If $\Gamma \vdash M : \tau$ then there exists M such that $|\mathrm{M}| = M$ and $\Gamma \vdash \mathrm{M} : \tau$.
3. If $\mathrm{M} \to_\beta \mathrm{N}$ then $|\mathrm{M}| \to_\beta |\mathrm{N}|$.
4. If $|\mathrm{M}| \to_\beta N$ then there exists N such that $|\mathrm{N}| = N$ and $\mathrm{M} \to_\beta \mathrm{N}$.

Proof: easy induction.

Wniosek: If $M = |\mathrm{M}|$ then $M \in \mathrm{SN}$ iff $\mathrm{M} \in \mathrm{SN}$.

Hindley-Milner

(Robinson)

Informal type annotations

Well-typed Curry style terms can be informally annotated by types, e.g. $((\lambda x^\sigma . N^\tau)^{\sigma\to\tau} P^\sigma)^\tau$. Such annotations represent type derivations and can be identified with Church-style terms.

In many cases it does not matter if we consider Curry style or Church style, orthodox or not. We always choose what is most convenient.

Warning: Sometimes one has to be careful. For instance, Church-Rosser in Church style is not immediate.

Unification (first order)

Assume a fixed signature of function symbols and constants. At least one symbol is more than unary.

Substitution – a function from variables to terms which is identity almost everywhere. Extended to terms:

$$S(f\,t_1 \dots t_n) = f\,S(t_1) \cdots S(t_n).$$

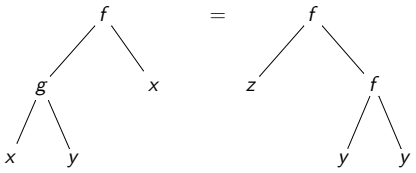
An *equation* is a pair of terms, written "$t = u$". A *system of equations* is a finite set of equations.

Variables in equations are called *unknowns*.

A substitution S is a *solution* of an equation "$t = u$" iff $S(t)$ and $S(u)$ is the same term. It is a solution of a system E of equations iff it is a solution of all equations in E.

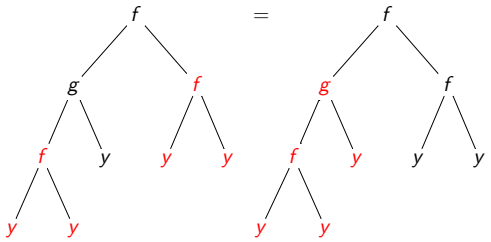
Example 1

The equation $f(gxy)x = fz(fyy)$ has a solution S, such that $S(x) = fyy$, $S(y) = y$, $S(z) = g(fyy)y$, and $S(v) = v$ otherwise.



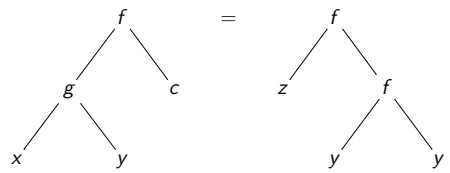
Example 1

The equation $f(gxy)x = fz(fyy)$ has a solution S, such that $S(x) = fyy$, $S(y) = y$, $S(z) = g(fyy)y$, and $S(v) = v$ otherwise.



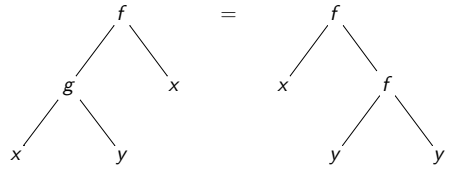
Example 2

The equation $f(gxy)c = fz(fyy)$, where c is a constant, has no solution.



Example 2

The equation $f(gxy)x = fx(fyy)$ has no solution.



Principal solutions (rozwiązania główne)

A substitution S is an *instance* of another substitution R (written $R \leq S$) iff $S = P \circ R$, for some substitution P.

A solution R of a system E is *principal* iff the following equivalence holds for all substitutions S:

$$S \text{ is a solution of } E \quad \text{iff} \quad R \leq S.$$

Example: The solution S of $f(gxy)x = fz(fyy)$, such that $S(x) = fyy$, $S(y) = y$, $S(z) = g(fyy)y$, $S(v) = v$ otherwise, is principal.

Theorem: *If a system of equations has a solution then it has a principal solution.*

Unification

The (first-order) *unification problem* is to decide if a given (system of) equation(s) has a solution.

Theorem: *The first-order unification problem is decidable in polynomial time. More precisely, it is* Ptime*-complete wrt* Logspace*-reductions.*

Type reconstruction

Fix a signature of one binary function symbol $\to$.
Terms over this signature are identified with simple types.

For every term M, define by induction

- a system of equations E_M;
- a type τ_M.

The goal: System E_M has a solution iff M is typable, and τ_M is the type sought for M.

Type reconstruction

- If M is a variable x, then $E_M = \varnothing$ and $\tau_M = p_x$, where p_x is a fixed type variable.
  Variables p_x are called *main variables*.
- If M is an application PQ then
  - $\tau_M = p$, where p is a fresh type variable;
  - $E_M = E_P \cup E_Q \cup \{\tau_P = \tau_Q \to p\}$,

  assuming non-main variables in E_P and E_Q are distinct.
- If M is an abstraction $\lambda x P$, then $E_M = E_P[p_x := p]$, and $\tau_M = p \to \tau_P[p_x := p]$, where p is fresh.

Type reconstruction

Lemma

1. *If $\Gamma \vdash M : \rho$, then there exists a solution S of E_M such that $\rho = S(\tau_M)$ and $S(p_x) = \Gamma(x)$, for $x \in \mathrm{FV}(M)$.*
2. *Let S be a solution of E_M, and let Γ be such that $\Gamma(x) = S(p_x)$, for all $x \in \mathrm{FV}(M)$. Then $\Gamma \vdash M : S(\tau_M)$.*

Proof: Induction with respect to the length of M.

Type reconstruction

Theorem: *Typability is decidable in* Ptime.

Proof: Reduction to unification.

Fact: *The same holds for type-checking.*

Proof: Similar.

Para główna i typ główny

A pair (Γ, τ) is a *principal pair* for M iff the following are equivalent for all Γ' and τ':

- $\Gamma' \vdash M : \tau'$;
- $S(\Gamma) \subseteq \Gamma'$ and $S(\tau) = \tau'$, for some substitution S.

Then τ is the *principal type* of M.

Corollary: If a term M is typable, then there exists a principal pair for M. This principal pair is unique up to renaming of type variables.

Examples

- Principal type of **S** is $(p \to q \to r) \to (p \to q) \to p \to r$.
  Another, non-principal, type of **S**:
  $(p \to q \to p) \to (p \to q) \to p \to p$
- Type $\omega = (p \to p) \to p \to p$ is the principal type of Church numerals $\mathbf{n}$, for $n \geq 2$. For $\mathbf{0}$ and $\mathbf{1}$ the principal types are respectively $p \to q \to q$ and $(p \to q) \to p \to q$. Every Church numeral can also be assigned the type $\omega_{p \to q} = ((p \to q) \to p \to q) \to (p \to q) \to p \to q$.

Definable functions

Liczebniki Churcha $\mathrm{n} = \lambda fx.\, f^n(x)$ mają każdy typ postaci

$$\omega_\sigma = (\sigma \to \sigma) \to (\sigma \to \sigma).$$

A function $f : \mathbb{N}^k \to \mathbb{N}$ is *β-definable in type ω_σ* if there is a closed term F such that

- $\vdash F : \omega_\sigma \to \cdots \to \omega_\sigma \to \omega_\sigma$;
- If $f(n_1, \ldots, n_k) = m$ then $F\mathrm{n}_1 \ldots \mathrm{n}_k =_\beta \mathrm{m}$.

Analogicznie można mówić o $\beta\eta$-definiowalności.

Examples

- Addition: $\lambda n^{\omega_\sigma} \lambda m^{\omega_\sigma} \lambda f^{\sigma\to\sigma} \lambda x^\sigma.\, nf(mfx)$;
- Multiplication: $\lambda n^{\omega_\sigma} \lambda m^{\omega_\sigma} \lambda f^{\sigma\to\sigma} \lambda x^\sigma.\, n(mf)x$;
- Test for zero (*if $n = 0$ then m else k*): $\lambda n^{\omega_\sigma} \lambda m^{\omega_\sigma} \lambda k^{\omega_\sigma} \lambda f^{\sigma\to\sigma} \lambda x^\sigma.\, n(\lambda y^\sigma.kfx)(mfx)$.
- Potęgowanie?
- Poprzednik?
- Odejmowanie? Równość?

Extended polynomials (wielomiany warunkowe)

The least class of functions containing:

- Addition;
- Multiplication;
- Test for zero;
- Constants zero and one;
- Projections,

and closed under compositions.

Wielomian warunkowy o 2 zmiennych

$f(x,y) = \texttt{if } x = 0 \texttt{ then if } y = 0 \texttt{ then } p_1(x,y) \texttt{ else } p_2(x,y) \texttt{ else if } y = 0 \texttt{ then } p_3(x,y) \texttt{ else } p_4(x,y)$.

Przykład:

$f(x,y) = \texttt{if } x = 0 \texttt{ then if } y = 0 \texttt{ then } 3 \texttt{ else } y^2 + 1 \texttt{ else if } y = 0 \texttt{ then } x^3 \texttt{ else } x^2 + y$.

Potęgowanie, poprzednik, równość itp.
nie są wielomianami warunkowymi.

Definable functions

Theorem (H. Schwichtenberg'76): *For every σ, the functions beta-definable in type ω_σ are exactly the extended polynomials.*

Example (M. Zakrzewski'07:) *The following function is $\beta\eta$-definable in type ω_σ for a certain σ:*

$$\mathit{ifeven}(p,q,r) = \begin{cases} q, & \text{if } p \text{ is even;} \\ r, & \text{otherwise.} \end{cases}$$

More definable functions

A function f is *skewly* (skośnie) definable if there is a closed term F such that

- $\vdash F : \omega_{\sigma_1} \to \cdots \to \omega_{\sigma_k} \to \omega_\sigma$;
- If $f(n_1, \ldots, n_k) = m$ then $F\mathrm{n}_1 \ldots \mathrm{n}_k =_\beta \mathrm{m}$.

Examples:

- The predecessor function $p(n) = n \dot{-} 1$ and the exponentiation function $\mathit{exp}(m,n) = m^n$ are skewly definable. (Easy)
- The subtraction $\mathit{minus}(m,n) = m \dot{-} n$ and equality test $\mathit{Eq}(m,n) = \texttt{if } m = n \texttt{ then } 0 \texttt{ else } 1$ are not definable skewly. (Hard)

Equality

Theorem (R. Statman'79): *The equality problem*

Are two well-typed terms beta-equal?

is non-elementary. That is, for no fixed k it is solvable in time

$$2^{2^{\cdot^{\cdot^{2^n}}}}\Big\}k$$

Exercise: How long is the normal form of $2 \ldots 2xy$?

Powtórzenie z logiki: minimalny rachunek zdań

(implikacyjny)

Naturalna dedukcja

Dowodzimy *osądów* postaci $\Gamma \vdash A$, gdzie Γ jest zbiorem formuł, a A jest formułą. Sens: A wynika z założeń Γ.

- Reguły *wprowadzania* spójników logicznych: jak można udowodnić formułę danej postaci?
- Reguły *eliminacji* spójników: jak można wykorzystać formułę tej postaci do udowodnienia innej?

Naturalna dedukcja: reguły dla $\to$

$$\Gamma, \sigma \vdash \sigma \quad \text{(Ax)}$$

$$\frac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \to \tau} \quad (\mathrm{I}\to)$$

$$\frac{\Gamma \vdash \sigma \to \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau} \quad (\mathrm{E}\to)$$

Naturalna dedukcja: notacja dla dowodów

$$\Gamma, x : \sigma \vdash x : \sigma$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x\, M : \sigma \to \tau}$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

Curry-Howard isomorphism

Twierdzenie
Dla dowolnego typu τ, kombinator typu τ istnieje wtedy i tylko wtedy, gdy τ jest twierdzeniem implikacyjnej logiki minimalnej.

Dowód: Inhabitant typu τ to nic innego jak dowód twierdzenia (ale z dodatkowymi adnotacjami). □

(Twierdzenie nadal zachodzi, jeśli zamiast „minimalnej implikacyjnej" napiszemy „intuicjonistycznej".)

Normalizacja

$$\dfrac{\dfrac{\begin{array}{c}\Gamma, \sigma \vdash \sigma \\ \vdots \\ \Gamma, \sigma \vdash \tau\end{array}}{\Gamma \vdash \sigma \to \tau} \quad \begin{array}{c}\vdots \\ \Gamma \vdash \sigma\end{array}}{\Gamma \vdash \tau} \quad \Rightarrow \quad \begin{array}{c}\vdots \\ \Gamma \vdash \sigma \\ \vdots \\ \Gamma \vdash \tau\end{array}$$

$$(\lambda x{:}\sigma\, M^{\tau})N^{\sigma} \quad \Rightarrow \quad M^{\tau}[x := N^{\sigma}]$$

Sens moralny normalizacji

Jeśli τ jest twierdzeniem, to istnieje taki term M w postaci normalnej, że $\vdash M : \tau$.

Czyli „dowód normalny".

Wniosek: *Logika minimalna jest niesprzeczna: istnieją formuły, których nie można udowodnić.*

Dowód: Na przykład $\not\vdash p$, bo w pustym otoczeniu nie ma termu normalnego o typie atomowym.

Curry-Howard isomorphism

Types	=	Formulas
Terms	=	Proofs
Computation	=	Proof normalization

The inhabitation problem

Inhabitation problem:

Given Γ, τ, is there M such that $\Gamma \vdash M : \tau$?

Twierdzenie (R. Statman): *Inhabitation in simple types is decidable and* PSPACE*-complete.*

Wniosek: *To samo dotyczy minimalnego rachunku zdań.*

Uwaga: Klasyczny rachunek zdań jest . . .
. . . „zaledwie" co-NP zupełny.

Long normal forms (Church style)

- If $x : \alpha_1 \to \cdots \alpha_n \to p$ and $M_1 : \alpha_1, \ldots M_k : \alpha_k$ are long normal forms then $xM_1 \ldots M_k : p$ is a long normal form.
- If $M : \beta$ is a long normal form, then $\lambda x : \alpha.\, M$ is a long normal form of type $\alpha \to \beta$.

Lemat 1: *If $\Gamma \vdash M : \tau$ then there exists a long normal form M' such that $M' =_{\beta\eta} M$ and $\Gamma \vdash M' : \tau$.*

Kontrakcja

Lemat 2: *If $\Gamma(x) = \rho$ and $\Gamma \cup \{y : \rho\} \vdash M : \tau$ then $\Gamma \vdash M[y := x] : \tau$.*

Searching for long normal forms

The Wajsberg/Ben-Yelles algorithm:

To answer $\Gamma \vdash ? : \alpha$, apply one of the following tactics:

- For $\alpha = \beta \to \gamma$, ask $\Gamma \cup \{x : \beta\} \vdash ? : \gamma$ (fresh x).

  (Solution $M = \lambda x{:}\beta.\, N^{\gamma}$.)

- For $\alpha = p$, choose $x : \beta_1 \to \cdots \to \beta_k \to p$ from Γ, then ask $\Gamma \vdash ? : \beta_i$, for all i. Accept if $k = 0$.

  (Solution $M = xN_1^{\beta_1} \ldots N_k^{\beta_k}$.)

Alternation

- A solution of $\Gamma \vdash ? : p$ is of the form $M = xN_1^{\beta_1} \ldots N_k^{\beta_k}$, for some $x : \beta_1 \to \cdots \to \beta_k \to p$ in Γ.
- Nondeterminism: because there can be more than one x.
- Branching (recursion): because $\Gamma \vdash N_i : \beta_i$ must be solved in parallel.

Ben-Yelles algorithm

To answer $\Gamma \vdash ? : \alpha$, apply one of the following tactics:

- For $\alpha = \beta \to \gamma$, ask $\Gamma \cup \{x : \beta\} \vdash ? : \gamma$ (fresh x).
  (Solution $M = \lambda x{:}\beta.\, N^{\gamma}$.)
- For $\alpha = p$, choose $x : \beta_1 \to \cdots \to \beta_k \to p$ from Γ, then ask $\Gamma \vdash ? : \beta_i$, for all i. Success if $k = 0$.
  (Solution $M = xN_1^{\beta_1} \ldots N_k^{\beta_k}$.)

Ten alternujący proces można objaśniać jako grę między graczem $\exists$ (który dowodzi tezy) i graczem $\forall$ (który poszukuje błędu w dowodzie). Rozwiązanie istnieje wtedy i tylko wtedy, gdy gracz $\exists$ ma strategię zwycięską.

Przykład (nieformalny): $(p \to \neg q) \to (\neg p \to \neg q) \to \neg q$

(Negacja $\neg\alpha$ oznacza $\alpha \to \bot$)

(*Assumptions:* $x : p \to \neg q$, $y : \neg p \to \neg q$, $z : q$. *Goal:* $\bot$)

Opponent ($\forall$): *Can you prove* $\bot$?

Player ($\exists$): *I will use assumption* y !

Opponent ($\forall$): *Can you prove the 1st assumption* $\neg p$? *That is, can you prove* $\bot$ *using new asumption* $v : p$?

Player ($\exists$): *Yes, I will use assumption* x !

Opponent ($\forall$): *Can you prove the 2nd assumption* q ?

Player ($\exists$): *Sure, take* z !

Proof = strategy = inhabitant: $\lambda xyz.\, y(\lambda v.\, xvz)z$

Termination

- The question $\Gamma \vdash ? : \beta \to \gamma$ leads to $\Gamma \cup \{x{:}\beta\} \vdash ? : \gamma$. The environment Γ is extended by $x : \beta$.
- Every such β is a subformula in the initial problem (there is only a linear number of such formulas).
- New variables can be replaced by old ones, i.e., Γ must in fact stabilize (in polynomial time).

Po wielomianowej liczbie kroków zadanie musi się powtórzyć.

Morał: Problem inhabitacji (czyli problem decyzyjny dla minimalnego rachunku zdań) jest w klasie Aptime = Pspace

Curry-Howard isomorphism

Types	=	Formulas
Terms	=	Proofs
Computation	=	Proof normalization
Computation	=	Proof construction

Konstrukcja dowodu jako obliczenie

Osąd $\Gamma \vdash q$ (gdzie q – atom),
można interpretować jako konfigurację automatu:

- Cel atomowy q to stan maszyny;
- Otoczenie Γ to zawartość pamięci, w tym program.
- Założenie $p \to q$ umożliwia zmianę stanu z q na p.
- Założenie $(r \to p) \to q$ umożliwia zmianę stanu z q na p, z jednoczesnym zapisaniem r w pamięci.
- Założenie $p \to r \to q$ to krok uniwersalny do p i r na raz.
- Założenie $p \to r \to q$ umożliwia zmianę stanu z q na r, gdy w pamięci jest zapisane p.

Uwaga: (1) nie można usunąć danej z pamięci;
(2) nie można sprawdzić, że danej w pamięci nie ma.

Monotonic automata

Monotonic automaton: $\mathcal{M} = \langle Q, R, f, \mathcal{I} \rangle$,

- Q is a finite set of states, $f \in Q$ the final state.
- R is a finite set of registers;
- $\mathcal{I}$ is a finite set of instructions of the form:
  (1) q : `check` S_1; `set` S_2; `jmp` p,
  (2) q : `jmp` p_1 `and` p_2,
  where $p, p_1, p_2 \in Q$ and $S_1, S_2 \subseteq R$.

Monotonic automata

Configuration: $\langle q, S \rangle$, where $q \in Q$ and $S \subseteq R$.
Final configurations are winning.

Transitions:

- for $I = q$: `check` S_1; `set` S_2; `jmp` p:
  $\langle q, S \rangle \to_I \langle p, S \cup S_2 \rangle$, provided $S_1 \subseteq S$;
  If $\langle p, S \cup S_2 \rangle$ winning then $\langle q, S \rangle$ is winning
- for $I = q$: `jmp` p_1 `and` p_2:
  $\langle q, S \rangle \to_I \langle p_1, S \rangle$ **and** $\langle q, S \rangle \to_I \langle p_2, S \rangle$
  If $\langle p_1, S \rangle$ and $\langle p_2, S \rangle$ are winning then so is $\langle q, S \rangle$.

Programs into proofs!

Given $\mathcal{M} = \langle Q, R, f, \mathcal{I} \rangle$ and $C_0 = \langle q_0, S_0 \rangle$,

define a set of axioms Γ so that

$$\Gamma \vdash q_0 \quad \text{iff} \quad C_0 \text{ is winning.}$$

Propositional variables: states and registers of $\mathcal{M}$.

Put all of S_0 into Γ, also $f \in \Gamma$.

Other axioms represent instructions of $\mathcal{M}$.

Instructions seen as axioms

Axiom for q : `check` S_1; `set` S_2; `jmp` p,
where $S_1 = \{s_1^1, \ldots, s_1^k\}$ and $S_2 = \{s_2^1, \ldots, s_2^\ell\}$:

(1) $\quad s_1^1 \to \cdots \to s_1^k \to (s_2^1 \to \cdots s_2^\ell \to p) \to q.$

Axiom for q : `jmp` p_1 `and` p_2:

(2) $\quad p_1 \to p_2 \to q.$

Proof construction as computation

To prove $\Gamma, S \vdash q$, one can:

- Note that $q = f$ and observe $f \in \Gamma$;
- Use axiom $p_1 \to p_2 \to q$ and prove in parallel:
  $\Gamma, S \vdash p_1$ and $\Gamma, S \vdash p_2$
- Use axiom $s_1^1 \to \cdots \to s_1^k \to (s_2^1 \to \cdots s_2^\ell \to p) \to q$,
  provided $S_1 = \{s_1^1, \ldots, s_1^k\} \subseteq S$, and prove that
  $\Gamma, S, s_2^1, \ldots, s_2^\ell \vdash p$

The technical lemma we need

Lemma
$\Gamma, S \vdash q$ *iff* $\langle q, S_0 \cup S \rangle$ *is winning.*

Proof.
($\Rightarrow$) Induction wrt the size of proof.
($\Leftarrow$) Induction wrt the definition of acceptance. □

Morał

Problem stopu dla automatów monotonicznych
redukuje się do minimalnej logiki implikacyjnej...

...czyli do inhabitacji w typach prostych.

I oczywiście (?) na odwrót.

A jak trudny jest ten problem stopu?

Monotonic automata

Lemma
The halting problem for monotonic automata
Is a given configuration winning?
is PSPACE*-hard.*

Trzeba to udowodnić.

Znany fakt

LBA = Linear Bounded Automaton
(maszyna liniowo ograniczona)

To jest maszyna Turinga, która pracuje wyłącznie na słowie wejściowym.

Wiadomo, że:

Problem stopu dla maszyn liniowo ograniczonych jest PSPACE*-zupełny*

Dowód: Padding (wypychanie).

Kodujemy LBA monotonicznie

Dany automat liniowo ograniczony A i słowo wejściowe w. Czas jest $2^{p(n)}$, gdzie $|w| = n$, a $p(n)$ jest wielomianem.

Idea: The monotonic automaton M simulates A by splitting the $2^{p(n)}$ steps of computation recursively into halves and executing the obtained fractions concurrently.

The „history" of each branch of the computation tree of M is recorded in its registers.

Rejestry

Trzy grupy rejestrów: B(egin), M(iddle), E(nd):

- $s_d^B(q)$, $s_d^M(q)$, $s_d^E(q)$ – "current state of A is q";
- $c_{id}^B(a)$, $c_{id}^M(a)$, $c_{id}^E(a)$ – "symbol at position i is a";
- h_{id}^B, h_{id}^M, h_{id}^E – "machine head scans position i".

Kodowanie

Zbiór rejestrów S *reprezentuje* konfigurację maszyny A *w pozycji* B, d, gdy do S należy

- dokładnie jeden rejestr postaci $s_d^B(q)$;

- dokładnie jeden rejestr postaci h_{id}^B;

- dokładnie jeden rejestr postaci $c_{id}^B(a)$, dla każdego $i \leq n$,

i nie ma w S żadnych rejestrów dla „czasu" $e < d$.

Analogicznie dla M, E.

Symulacja

Automat monotoniczny M na początku ustawia sobie rejestry reprezentujące konfigurację początkową A w pozycji $B, p(n)$ i konfigurację końcową A w pozycji $E, p(n)$.

Następnie wchodzi w stan $q_{p(n)}$.

Zadaniem automatu M w takiej konfiguracji $\langle q_{p(n)}, S\rangle$

jest sprawdzenie, że maszyna A może przejść z konfiguracji początkowej do końcowej w czasie $2^{p(n)}$.

Symulacja

Teraz $d \leq p(n)$.

W konfiguracji $\langle q_d, S\rangle$ zbiór S reprezentuje konfigurację C_0 maszyny A w pozycji B, d i konfigurację C_2 w pozycji E, d. W zbiorze S nie ma innych rejestrów typu $e \leq d$.

Zadaniem automatu M w takiej konfiguracji $\langle q_d, S\rangle$ jest sprawdzenie, że maszyna A może przejść z konfiguracji C_0 do konfiguracji C_2 w czasie 2^d.

Teraz M „zgaduje" konfigurację C_1 i ustawia rejestry tak, żeby ją reprezentować w pozycji M, d

Symulacja

Mamy rejestry reprezentujące trzy konfiguracje maszyny A:

- C_0 w pozycji B, d;

- C_1 w pozycji M, d;

- C_2 w pozycji E, d.

Teraz obliczenie rozdziela się na dwie gałęzie.
W pierwszej gałęzi wykonujemy (nieformalnie)
$B_{d-1} := B_d$; $E_{d-1} := M_d$; jmp Q_{d-1},

W drugiej gałęzi $B_{d-1} := M_d$; $E_{d-1} := E_d$; jmp Q_{d-1}.

Jak to się kończy?

W konfiguracji $\langle q_d, S\rangle$ sprawdzamy, że maszyna A może przejść z C_0 do C_2 w czasie 2^d.

Dla $d = 0$ należy sprawdzić, że C_2 jest osiągalna z C_0 w jednym kroku. To łatwe.

Ale niedeterministyczne, bo to jest maszyna monotoniczna. Trzeba zgadywać, które rejestry są ustawione.

Co trzeba udowodnić

Automat A potrafi przejść z C_0 do C_2 w czasie 2^d
wtedy i tylko wtedy, gdy
konfiguracja $\langle q_d, S\rangle$ (dla odpowiedniego S) wygrywa w M.

W szczególności:

Automat A akceptuje w (w czasie $2^{p(n)}$)
wtedy i tylko wtedy, gdy
konfiguracja początkowa wygrywa w M.

Morał

Twierdzenie:
Minimalny rachunek zdań jest PSPACE*-zupełny.*

A klasyczny?
Tylko *co-NP zupełny*.

Hilbert style proofs

Hilbert style propositional axioms for implication:

- $\alpha \to \beta \to \alpha$;
- $(\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma$.

Hilbert style proof rule: *modus ponens* (application).
Dowód to po prostu ciąg formuł.

Deduction theorem:
If $\Gamma, \sigma \vdash \tau$ *then* $\Gamma \vdash \sigma \to \tau$.

Example proof

1. $(\varphi \to (\psi \to \varphi) \to \varphi) \to (\varphi \to \psi \to \varphi) \to \varphi \to \varphi$;
2. $\varphi \to (\psi \to \varphi) \to \varphi$;
3. $(\varphi \to \psi \to \varphi) \to \varphi \to \varphi$ (detach 2 from 1);
4. $\varphi \to \psi \to \varphi$;
5. $\varphi \to \varphi$ (detach 4 from 3).

Proofs as combinators

Hilbert style propositional axioms for implication:

- $\mathsf{K} : \alpha \to \beta \to \alpha$;
- $\mathsf{S} : (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma$.

Deduction theorem:
If $\Gamma, x{:}\sigma \vdash M : \tau$ *then* $\Gamma \vdash \lambda^* x.\, M : \sigma \to \tau$.

Logical connectives

$$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi}\,(I\wedge) \qquad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi}\,(E\wedge)\ \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi}$$

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi}\,(I\vee)\ \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi}$$

$$\frac{\Gamma \vdash \varphi \vee \psi \quad \Gamma, \varphi \vdash \vartheta \quad \Gamma, \psi \vdash \vartheta}{\Gamma \vdash \vartheta}\,(E\vee)$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash \varphi}\,(E\bot)$$

Conjunction is the product

$$\frac{\Gamma \vdash M : \varphi \quad \Gamma \vdash N : \psi}{\Gamma \vdash \langle M, N\rangle : \varphi \wedge \psi}\,(I\wedge)$$

$$\frac{\Gamma \vdash M : \varphi \wedge \psi}{\Gamma \vdash \pi_1(M) : \varphi}\,(E\wedge) \qquad \frac{\Gamma \vdash M : \varphi \wedge \psi}{\Gamma \vdash \pi_2(M) : \psi}$$

$$\pi_1(\langle M, N\rangle) \Rightarrow M, \qquad \pi_2(\langle M, N\rangle) \Rightarrow N$$

Disjunction is the coproduct (suma prosta)

$$\frac{\Gamma \vdash M : \varphi}{\Gamma \vdash \mathsf{inl}(M) : \varphi \vee \psi}\,(W\vee) \qquad \frac{\Gamma \vdash M : \psi}{\Gamma \vdash \mathsf{inr}(M) : \varphi \vee \psi}$$

$$\frac{\Gamma \vdash M : \varphi \vee \psi \quad \Gamma(x : \varphi) \vdash P : \vartheta \quad \Gamma(y : \psi) \vdash Q : \vartheta}{\Gamma \vdash \mathsf{case}\ M\ \mathsf{of}\ [x]P, [y]Q : \vartheta}\,(E\vee)$$

$$\begin{aligned} \mathsf{case}\ \mathsf{inl}(P)\ \mathsf{of}\ [x]M, [y]N &\Rightarrow M[x := P] \\ \mathsf{case}\ \mathsf{inr}(Q)\ \mathsf{of}\ [x]M, [y]N &\Rightarrow N[y := Q]. \end{aligned}$$

Generalized eta-reductions

If we believe everything is canonical...

$$\langle \pi_1(M), \pi_2(M) \rangle \Rightarrow M$$

$$(\text{case } M \text{ of } [x]\text{inl } x, [y]\text{inr } y) \Rightarrow M.$$

Absurd is the empty type (*the thing which is not*[4])

Ex falso quodlibet:

$$\frac{\Gamma \vdash M : \bot}{\Gamma \vdash \varepsilon_\varphi(M) : \varphi} \; (E\bot)$$

No introduction rule, no beta- or eta-reduction.

[4] Rzecz, która nie jest. (J. Swift)

Konserwatywność

Twierdzenie

Intuicjonistyczny rachunek zdań IPC (ze wszystkimi spójnikami $\vee, \wedge, \to, \bot, \neg$ i regułą ex falso) jest konserwatywny nad logiką minimalną: jeśli formuła implikacyjna τ jest twierdzeniem IPC, to jest twierdzeniem logiki minimalnej.

Dowód: Trzeba udowodnić normalizację rozszerzonego rachunku lambda.

Postać normalna o typie implikacyjnym jest zbudowana tylko z abstrakcji i aplikacji.

Classical (unnatural) deduction

$$\frac{\Gamma, \neg\sigma \;\vdash\; \bot}{\Gamma \;\vdash\; \sigma} \; (\text{Cheat})$$

Example: Peirce's Law

$$((p \to q) \to p) \to p$$

is classically valid, but unprovable without rule (Cheat).

Indeed, there is no normal M with

$$x : (p \to q) \to p \;\vdash\; M : p$$

Representing data types

- Natural numbers are generated by
  - Constant $0 : \text{int}$;
  - Successor $s : \text{int} \to \text{int}$.

  They correspond to long normal forms of type
  $$\omega = (0 \to 0) \to 0 \to 0$$
- Words over $\{a, b\}$ are generated by
  - Constant $\varepsilon : \text{word}$;
  - Two successors $\lambda\!\!\lambda w\,(a \cdot w)$ and $\lambda\!\!\lambda w\,(b \cdot w)$ of type $\text{word} \to \text{word}$.

  They correspond to long normal forms of type
  $$\text{word} = (0 \to 0) \to (0 \to 0) \to 0 \to 0$$

Representing data types

- Booleans are two constants *true*, *false*.
  They correspond to terms of type $\text{bool} = 0 \to 0 \to 0$.
- Binary trees are generated by
  - Constant $\mathit{nil} : \text{tree}$;
  - Constructor $\mathit{cons} : \text{tree} \to \text{tree} \to \text{tree}$.

  They correspond to long normal forms of type
  $$\text{tree} = (0 \to 0 \to 0) \to 0 \to 0$$

Generalization:
Term algebras correspond to types of order two, i.e, of the form

$$(0^{n_1} \to 0) \to \cdots \to (0^{n_k} \to 0) \to 0$$

Representing data types

Types of order three and up define more complex structures.

For instance, long normal forms of type

$$((0 \to 0) \to 0) \to 0$$

are as follows:

$$\lambda F.\, F(\lambda x_1.\, F(\lambda x_2.\, F(\lambda x_3\, F \ldots \lambda x_k.\, x_\ell) \ldots)).$$

Type reducibility

Definition: Type τ is *reducible* to type σ iff there exists a closed term $\Phi : \tau \to \sigma$ such that the operator $\lambda\!\!\lambda M{:}\tau.\, \Phi M$ is injective on closed terms, i.e.,

$$\Phi M_1 =_{\beta\eta} \Phi M_2 \quad \text{implies} \quad M_1 =_{\beta\eta} M_2$$

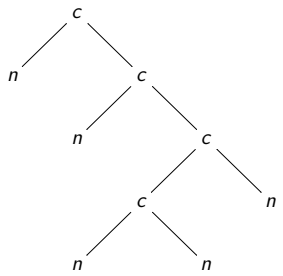
for closed $M_1, M_2 : \tau$.

Examples:

- Type ω is reducible to type word using $\Phi = \lambda nfgx.nfx$.
- Types $\alpha \to \beta \to 0$ and $\beta \to \alpha \to 0$ are reducible to each other using $\Phi = \lambda xyz.\, xzy$.
- Type word is reducible to type tree using $\Phi = \lambda wcn.w(\lambda x.\, cnx)(\lambda x.\, cxn)(cnn)$.

Example

Encoding word $w = aab = a \cdot a \cdot b \cdot \varepsilon$ as a tree,
corresponding to $\Phi w = \lambda cn.w(\lambda x.\, cnx)(\lambda x.\, cxn)(cnn)$.



A nice theorem

Theorem (R. Statman):

Every type over a single type constant 0 *is reducible to* tree.

Idea dowodu wg Thierry'ego Joly w kilku krokach.

Redukcja do typu $\mathcal{J}$

Mając „stałe" $@ : 0 \to 0 \to 0$ i $\Delta : (0 \to 0) \to 0$,
możemy zakodować dowolny term jako term typu 0.

Na przykład term $2 = \lambda fx.\, f(fx)$ zakodujemy jako

$$\text{Dwa} = \Delta(\lambda f^0.\, \Delta(\lambda x^0.\, @f(@fx)))$$

Bardziej sugestywna notacja:
— piszmy Λx zamiast $\Delta(\lambda x$ oraz $M \cdot N$ zamiast $@MN$.
Wtedy term **Dwa** napiszemy tak: $\Lambda f^0\, \Lambda x^0\, (f \cdot (f \cdot x))$.

Opuszczamy szczegóły.

$\Phi : [((0 \to 0) \to 0) \to 0] \;\to\; [(0 \to 0) \to (0 \to 0) \to 0 \to 0]$

$$\Phi = \lambda Z \lambda f^{0\to 0} g^{0\to 0} x^0.\, Z(\lambda h^{0\to 0}.\, f(h(g(hx))))$$

Kombinatory typu $[(0 \to 0) \to 0] \to 0$ są takie:

$$M = \lambda\varphi^{(0\to 0)\to 0}.\, \varphi(\lambda y_1.\, \varphi(\lambda y_2.\, \ldots \varphi(\lambda y_k.\, y_j))).$$

Term M jest zdeterminowany przez liczby k i j.

Przykład: $M = \lambda\varphi^{(0\to 0)\to 0}.\, \varphi(\lambda y_1.\, \varphi(\lambda y_2.\, \varphi(\lambda y_3.\, \varphi(\lambda y_4.\, y_2))))$.
Wtedy $\Phi(M) = \lambda fgx.\, f^4(g(f^2(x)))$.

Opuszczamy szczegóły.

Ważny przykład: redukcja **bool** $\to$ **word** do **tree**.

Typ bool $\to$ word, czyli:

$$(0 \to 0 \to 0) \to (0 \to 0) \to (0 \to 0) \to 0 \to 0$$

odpowiada algebrze termów nad sygnaturą

$$t : 0 \to 0 \to 0, \quad f, g : 0 \to 0, \quad c : 0.$$

Trzeba je zakodować jako zwykłe drzewa nad sygnaturą

$$t : 0 \to 0 \to 0, \quad c : 0.$$

Można to zrobić używając np. $\Theta : (\text{bool} \to \text{word}) \to \text{tree}$:

$$\Theta = \lambda Z\, \lambda t^{0\to 0\to 0} c^0.\, Zt(\lambda x\, tcx)(\lambda x\, txc)(tcc)$$

Krok pierwszy

Lemat: *Każdy typ nad* 0 *można zredukować do typu*

$$\mathcal{J} = (0 \to 0 \to 0) \to ((0 \to 0) \to 0) \to 0.$$

Pomysł na dowód:

Kombinatory typu $\mathcal{J}$ są postaci $\lambda @\, \lambda\Delta.\, W$, gdzie:

$$@ : 0 \to 0 \to 0, \quad \Delta : (0 \to 0) \to 0 \;\vdash\; W : 0.$$

Operatory $@$ i Δ mogą naśladować aplikację i abstrakcję.

Krok drugi

Zredukowaliśmy dowolny typ do typu:

$$\mathcal{J} = (0 \to 0 \to 0) \to ((0 \to 0) \to 0) \to 0$$

czyli do $\quad \mathcal{J} = \text{bool} \to ((0 \to 0) \to 0) \to 0$.

Teraz zredukujemy końcówkę $((0 \to 0) \to 0) \to 0$
do typu $\quad \text{word} = (0 \to 0) \to (0 \to 0) \to 0 \to 0$.
(Potem zredukujemy $\mathcal{J}$ do typu bool $\to$ word.)

Krok trzeci

Mamy $\mathcal{J} = \text{bool} \to \mathcal{J}_2 \to 0$, gdzie:

$$\text{bool} = 0 \to 0 \to 0 \qquad \mathcal{J}_2 = (0 \to 0) \to 0.$$

W kroku drugim zredukowaliśmy typ $\mathcal{J}_2 \to 0$ do typu

$$\text{word} = (0 \to 0) \to (0 \to 0) \to 0 \to 0$$

używając $\Phi = \lambda Z^{\mathcal{J}_2\to 0} \lambda f^{0\to 0} g^{0\to 0} x^0.\, Z(\lambda h^{0\to 0}\, f(h(g(hx))))$

Teraz redukujemy bool $\to \mathcal{J}_2 \to 0$ do bool $\to$ word używając
$\Psi = \lambda Z^{\text{bool}\to\mathcal{J}_2\to 0} \lambda t^{\text{bool}} \lambda f^{0\to 0} g^{0\to 0} x^0.\, Zt(\lambda h^{0\to 0}\, f(h(g(hx))))$
czyli $\Psi = \lambda Z^{\text{bool}\to\mathcal{J}_2\to 0} \lambda t^{\text{bool}}.\, \Phi(Zt)$.

Uwaga: to nie jest oczywiste!

Krok czwarty: redukcja bool $\to$ word do tree.

Typy proste: semantyka

To już było.

Semantics for finite types

Assumptions:

- Orthodox Church style;
- Only one atomic type 0;
- Extensional equality $=_{\beta\eta}$.

Standard model $\mathfrak{M}(A)$

- Basic domain $D_0 = A$;
- Function domains: $D_{\sigma\to\tau} = D_\sigma \to D_\tau$;
- Obvious semantics:
  - $[\![x]\!]_v = v(x)$;
  - $[\![MN]\!]_v = [\![M]\!]_v([\![N]\!]_v)$;
  - $[\![\lambda x^\tau.\, M]\!]_v = \lambda\!\!\lambda d{\in}D_\tau.\, [\![M]\!]_{v[x\mapsto d]}$.

Wykonatowane szczegóły po angielsku

Completeness

Theorem (Harvey Friedman):

Terms are $\beta\eta$-equal iff they are equal in $\mathfrak{M}(\mathbb{N})$.

Szkic dowodu: „Zanurzenie" $T_\sigma/_{=_{\beta\eta}}$ w D_σ.

Finite completeness

„Łatwa" obserwacja:

For every M and N there is k such that:

$$M =_{\beta\eta} N \quad \textit{iff} \quad \mathfrak{M}(k) \models M = N.$$

Corollary:

Terms are $\beta\eta$-equal iff they are equal in all finite models.

(Konwencja: $k = \{0, 1, \ldots, k-1\}$)

Finite completeness

Theorem (R. Statman):

For every M there is k such that, for all N:

$$M =_{\beta\eta} N \quad \textit{iff} \quad \mathfrak{M}(k) \models M = N.$$

Corollary:

Terms are $\beta\eta$-equal iff they are equal in all finite models.

(Konwencja: $k = \{0, 1, \ldots, k-1\}$)

Finite completeness proof

It suffices to prove that

for every closed $M : \text{tree}$ *there is* k *such that, for all* $N : \text{tree}$*:*

$$M =_{\beta\eta} N \qquad \textit{iff} \qquad \mathfrak{M}(k) \models M = N.$$

Indeed, for closed $M : \tau$, consider $\Phi(M)$,
where Φ is a reduction of τ to tree.

For non-closed terms, consider appropriate lambda-closures.

Finite completeness for **tree**

Let $p(m)(n) = 2^m(2n+1)$. Then $p \in D_{0\to 0\to 0}$ in $\mathfrak{M}(\mathbb{N})$.
Observe that $p(m)(n) > m, n$, for all m, n.

Term zamknięty typu tree, to w istocie drzewo.

Wartość $[\![M]\!](p)(0)$ można uważać za numer tego drzewa.

Ćwiczenie

$p(m)(n) = 2^m(2n+1)$
Jaka liczba jest numerem drzewa $\lambda px.\, px(p(pxx)x)$?

For $M : \text{tree}$, define $k = 2 + [\![M]\!](p)(0)$, i.e. $2 + \mathit{numer}(M)$.

Let $p' : k \to k \to k$ be p „truncated" to values less than k.
Then $p' \in D_{0\to 0\to 0}$ in $\mathfrak{M}(k)$.

Suppose $\mathfrak{M}(k) \models M = N$. Then in the model $\mathfrak{M}(k)$:

$$[\![M]\!](p')(0) = [\![N]\!](p')(0) \qquad (*)$$

All numbers needed to verify (*) are at most $k-2$. Hence

$$[\![M]\!](p')(0) = [\![N]\!](p')(0) = k-2$$

and $[\![M]\!](p)(0) = [\![N]\!](p)(0)$ also holds in $\mathfrak{M}(\mathbb{N})$.
It follows that $M =_{\beta\eta} N$.

Equality is not definable in simple types

There is no $E : \omega_\tau \to \omega_\sigma \to \omega_\rho$*, such that for all* $p, q \in \mathbb{N}$*:*

$$E\,\mathsf{p}^{\omega_\tau}\mathsf{q}^{\omega_\sigma} =_{\beta\eta} 0^{\omega_\rho} \qquad \text{iff} \qquad p = q.$$

Proof: By Statman's thm., take k such that for all $N : \omega_\rho$:

$$\mathfrak{M}(k) \models 0^{\omega_\rho} = N \qquad \text{iff} \qquad 0^{\omega_\rho} =_{\beta\eta} N.$$

There are $p \neq q$ with $[\![\mathsf{p}^{\omega_\tau}]\!] = [\![\mathsf{q}^{\omega_\tau}]\!]$ in $\mathfrak{M}(k)$. So in $\mathfrak{M}(k)$:

$$[\![E\mathsf{p}^{\omega_\tau}\mathsf{q}^{\omega_\sigma}]\!] = [\![E]\!][\![\mathsf{p}^{\omega_\tau}]\!][\![\mathsf{q}^{\omega_\sigma}]\!] = [\![E]\!][\![\mathsf{q}^{\omega_\tau}]\!][\![\mathsf{q}^{\omega_\sigma}]\!] = [\![E\mathsf{q}^{\omega_\tau}\mathsf{q}^{\omega_\sigma}]\!] = [\![0^{\omega_\rho}]\!]$$

Thus $\mathfrak{M}(k) \models E\mathsf{p}^{\omega_\tau}\mathsf{q}^{\omega_\sigma} = 0^{\omega_\rho}$, whence $p = q$.

Twierdzenie Parysa

Definicja: Funkcje $f, g : A \to \mathbb{N}$ są *dominacyjnie równoważne* wtw, gdy dla dowolnego $X \subseteq A$:

zbiór $f(X)$ jest ograniczony $\Leftrightarrow$ zbiór $g(X)$ jest ograniczony.

Twierdzenie (Paweł Parys, 2016): Dla każdego typu τ, wszystkie funkcje definiowalne typu $\tau \to \omega$ tworzą tylko skończenie wiele klas dominacyjnej równoważności.

Wniosek: Dla każdego typu τ istnieje takie n, że dla $k \geq n$ zbioru $\mathbb{N}^k$ nie można reprezentować w typie τ.

Przyczyna: Dla dużych k rzutowania Π^k_i są dominacyjnie parami nierównoważne.

Rachunek λY

Do rachunku $\lambda_\to$ dodajemy stałe $\mathsf{Y}_\sigma : (\sigma \to \sigma) \to \sigma$
z regułą redukcji $\mathsf{Y}_\sigma \Rightarrow \lambda f^{\sigma\to\sigma}.\, f(\mathsf{Y}_\sigma f)$.

Czy teraz można zdefiniować więcej funkcji z $\mathbb{N} \to \mathbb{N}$?

Odpowiedź (Plotkin'82): Nie można.

Dowód: Analiza rzetelności (strictness analysis)

Analiza rzetelności (strictness analysis)

Język $\lambda\Omega$ – rachunek $\lambda_\to$ plus stałe Ω_σ.

Model $\mathbb{O}$ rozróżnia tylko określone i nieokreślone:

$\mathcal{O}_0 = \{\bot, \top\}$, gdzie $\bot < \top$

$\mathcal{O}_{\sigma\to\tau} = [\mathcal{O}_\sigma \to \mathcal{O}_\tau]$

$[\![\Omega_\sigma]\!] = \bot_\sigma$, $[\![\mathsf{Y}_\sigma]\!] = \mathrm{lfp}$

Fakt: $[\![\mathsf{Y}_\sigma]\!] = [\![\mathsf{Y}^\bullet_\sigma]\!]$, gdzie $\mathsf{Y}^\bullet_\sigma = \lambda f.\, f^{h(\sigma)}(\Omega_\sigma)$,
dla odpowiednich stałych $h(\sigma)$.

Ogólnie $[\![M]\!]_\rho = [\![M^\bullet]\!]_\rho$

Tester rzetelności

Dla $\sigma = \sigma_1 \to \cdots \to \sigma_n \to 0$ definiujemy:

$s_\sigma \in \mathcal{O}_\sigma$ oraz $t_\sigma : \mathcal{O}_\sigma \to \{\bot, \top\}$.

$t_\sigma(d) = d s_{\sigma_1} \dots s_{\sigma_n}$, dla $d \in \mathcal{O}_\sigma$;

$s_\sigma d_1 \dots d_n = \inf_i t_{\sigma_i}(d_i)$, dla $d_i \in \sigma_i$.

Główny lemat:

Term zamknięty $M \in \lambda\Omega$ ma postać normalną bez omegi wtedy i tylko wtedy, gdy $t_\sigma([\![M]\!]) = \top$.

Szkic dowodu

Jeśli $F\mathsf{n} \twoheadrightarrow \mathsf{m}$, to $[\![F^\bullet\mathsf{n}]\!] = [\![\mathsf{m}^\bullet]\!] = [\![\mathsf{m}]\!]$ w modelu $\mathbb{O}$.

Zatem $t_\omega([\![F^\bullet\mathsf{n}]\!]) = t_\omega([\![\mathsf{m}]\!]) = \top$.

Postać normalna termu $F^\bullet\mathsf{n}$ nie zawiera omegi, więc musi być liczebnikiem. Pozostaje sprawdzić, że

– ten liczebnik to jest m;

– omegi w termie $F^\bullet$ są niepotrzebne.

Morał: Istnieje term $\overline{F}^\bullet$, bez omeg i ygreków, który definiuje tę samą funkcję, co F.

Rozszerzenia rachunku typów prostych

System PCF: Rachunek λY plus typ **int** z operacjami poprzednika i następnika i stałą zero.

System T: Jak PCF, ale zamiast $\mathbf{Y}_\sigma$ są *rekursory* (uogólnienie rekursji prostej na wyższe typy.)

Twierdzenie 1: System T ma własność SN.

Twierdzenie 2: Twierdzenie 1 jest niezależne od PA.

Tydzień temu: λY

Twierdzenie (Plotkin)

Dodanie kombinatorów punktu stałego nie zmienia klasy funkcji definiowalnych w typach prostych.

System PCF: Rachunek λY plus typ **int** z operacjami poprzednika i następnika i stałą zero.

System T: Jak PCF, ale zamiast $\mathbf{Y}_\sigma$ są *rekursory* (uogólnienie rekursji prostej na wyższe typy.)

System T Gödla

Typy: Typy proste zbudowane z jednej stałej typowej **int**. (Czasem dodaje się **bool**, produkty itp.)

Termy: Jak w rachunku lambda z typami prostymi plus stałe:

$$0 : \mathsf{int} \qquad \mathsf{s} : \mathsf{int} \to \mathsf{int}$$
$$\mathsf{R}_\tau : \mathsf{int} \to \tau \to (\mathsf{int} \to \tau \to \tau) \to \tau,$$

Redukcja: Zwykła beta-redukcja oraz:

$$\mathsf{R}_\tau\, 0\, P\, Q \Rightarrow P \qquad \mathsf{R}_\tau(\mathsf{s}\, n)\, P\, Q \Rightarrow Q\, n\, (\mathsf{R}_\tau\, n\, P\, Q)$$

Przykład: Funkcja poprzednika

$$pred = \lambda n^{\mathsf{int}}.\, \mathsf{R}_{\mathsf{int}}\, n\, 0\, (\lambda xy.\, x)$$

Skąd się wziął system T

Weźmy aksjomat indukcji Peana z 1889 roku:

$$\tau(0) \to \forall y(\mathsf{int}(y) \to \tau(y) \to \tau(\mathsf{s}y)) \to \forall x(\mathsf{int}(x) \to \tau(x)).$$

Napiszmy go trochę inaczej:

$$\forall x(\mathsf{int}(x) \to \tau(0) \to \forall y(\mathsf{int}(y) \to \tau(y) \to \tau(\mathsf{s}y)) \to \tau(x)).$$

I wytrzyjmy wszystko, co indywiduowe:

$$\mathsf{int} \to \tau \to (\mathsf{int} \to \tau \to \tau) \to \tau.$$

To jest typ rekursora.

Własności systemu T

- System **T** ma własność silnej normalizacji (dowód „metodą Taita").
- Definiowalne są wszystkie funkcje dowodliwie rekurencyjne w PA.
- Własność SN dla systemu **T** jest niezależna od PA.

Problemy decyzyjne

Plotkin's problem

Given $d \in D_\tau$ in a finite model $\mathfrak{M}(X)$.
Is there a closed term $M : \tau$ with $[\![M]\!] = d$?

More generally:

Let $v(x_1) = e_1 \in D_{\sigma_1}, \ldots, v(x_n) = e_n \in D_{\sigma_n}$.
Is there M such that $[\![M]\!]_v = d$?

(Is d definable from $e_1, \ldots, e_n$?)

Fact: These decision problems are reducible to each other.

Undecidablility of lambda-definability

Theorem (Ralph Loader, 1993):

Plotkin's problem is undecidable.

Dowód: Redukcja z problemu przepisywania słów.

Kodujemy słowa w i v i reguły systemu jako elementy modelu.
Pytamy, czy v jest definiowalne z w i reguł.

Proof

Take $X = \{a, b, L, R, *, 1, 0\}$. Encode any word $w = o_1 \ldots o_n$ as a function $\overline{w} : D_0^n \to D_0$, such that

- $\overline{w}(* \cdots * o_i * \cdots *) = 1$, if the i-th symbol in w is o_i;
- $\overline{w}(* \cdots * LR * \cdots *) = 1$;
- $\overline{w}(\ldots) = 0$, otherwise.

How does it work?

For $w = w_1 C w_2$ we have $\overline{w} = \lambda\!\!\lambda \vec{x} \lambda\!\!\lambda \vec{y} \lambda\!\!\lambda \vec{z}.\, \overline{w}(\vec{x})(\vec{y})(\vec{z})$.

Fix $\vec{x}, \vec{z}$ and consider the function $g = \lambda\!\!\lambda \vec{y}.\, \overline{w}(\vec{x})(\vec{y})(\vec{z})$.

It "accepts" the following strings (depending on $\vec{x}, \vec{z}$):

$\vec{x}$	$\vec{y}$	$\vec{z}$
$* \cdots * o_i * \cdots *$	$* \cdots\cdots *$	$* \cdots\cdots *$
$* \cdots\cdots *$	same as $\overline{C}$	$* \cdots\cdots *$
$* \cdots\cdots *$	$* \cdots\cdots *$	$* \cdots * o_i * \cdots *$
$* \cdots * LR * \cdots *$	$* \cdots\cdots *$	$* \cdots\cdots *$
$* \cdots\cdots * L$	$R * \cdots\cdots *$	$* \cdots\cdots *$
$* \cdots\cdots *$	$* \cdots\cdots * L$	$R * \cdots\cdots *$
$* \cdots\cdots *$	$* \cdots\cdots *$	$* \cdots * LR * \cdots *$

How does it work?

Fix $\vec{x}, \vec{z}$ and consider the function $g = \lambda\!\!\lambda \vec{y}.\, \overline{w}(\vec{x})(\vec{y})(\vec{z})$.

Depending on $\vec{x}, \vec{z}$, the function g is as follows:

$\vec{x}$	g	$\vec{z}$
$* \cdots * o_i * \cdots *$	$\chi_{\{*\cdots*\}}$	$* \cdots\cdots *$
$* \cdots\cdots *$	$\overline{C}$	$* \cdots\cdots *$
$* \cdots\cdots *$	$\chi_{\{*\cdots*\}}$	$* \cdots * o_i * \cdots *$
$* \cdots * LR * \cdots *$	$\chi_{\{*\cdots*\}}$	$* \cdots\cdots *$
$* \cdots\cdots * L$	$\chi_{\{R*\cdots*\}}$	$* \cdots\cdots *$
$* \cdots\cdots *$	$\chi_{\{*\cdots*L\}}$	$R * \cdots\cdots *$
$* \cdots\cdots *$	$\chi_{\{*\cdots*\}}$	$* \cdots * LR * \cdots *$

Otherwise $g = \chi_\varnothing$.

How to encode a rule $F = (C \Rightarrow D)$?

Fix $\vec{x}, \vec{z}$ and consider the function $g = \lambda\!\!\lambda \vec{y}.\, \overline{w}(\vec{x})(\vec{y})(\vec{z})$.

What will change in this table if we replace $w_1 C w_2$ by $w_1 D w_2$?

$\vec{x}$	g	$\vec{z}$
$* \cdots * o_i * \cdots *$	$\chi_{\{*\cdots*\}}$	$* \cdots\cdots *$
$* \cdots\cdots *$	$\overline{C}$	$* \cdots\cdots *$
$* \cdots\cdots *$	$\chi_{\{*\cdots*\}}$	$* \cdots * o_i * \cdots *$
$* \cdots * LR * \cdots *$	$\chi_{\{*\cdots*\}}$	$* \cdots\cdots *$
$* \cdots\cdots * L$	$\chi_{\{R*\cdots*\}}$	$* \cdots\cdots *$
$* \cdots\cdots *$	$\chi_{\{*\cdots*L\}}$	$R * \cdots\cdots *$
$* \cdots\cdots *$	$\chi_{\{*\cdots*\}}$	$* \cdots * LR * \cdots *$
other	$\chi_\varnothing$	other

How to encode a rule $F = (C \Rightarrow D)$

Every rule $F = (C \Rightarrow D)$ is encoded as a function

$$\overline{F} : (D_0^m \to D_0) \to (D_0^n \to D_0),$$

where $m = |C|$ and $n = |D|$. We take:

- $\overline{F}(\chi_{\{*\cdots*\}}) = \chi_{\{*\cdots*\}}$;
- $\overline{F}(\chi_{\{R*\cdots*\}}) = \chi_{\{R*\cdots*\}}$;
- $\overline{F}(\chi_{\{*\cdots*L\}}) = \chi_{\{*\cdots*L\}}$;
- $\overline{F}(\overline{C}) = \overline{D}$;
- $\overline{F}(g) = \chi_\varnothing$, for any other g.

Claim

A word w can be rewritten to v iff the element $\overline{v}$ of $\mathfrak{M}(X)$ is definable from $\overline{w}$ and the functions $\overline{F}$ encoding the rules.

The easy part: Let $w = w_1 C w_2$ rewrite to $v = w_1 D w_2$ using $F = (C \Rightarrow D)$. Assume that term W defines $\overline{w}$. Then $\overline{v}$ is definable by

$$V = \lambda \vec{x} \vec{u} \vec{z}.\, \overline{F}(\lambda \vec{y}.\, W \vec{x} \vec{y} \vec{z}) \vec{u}, \qquad (*)$$

It follows that codes of reachable words are definable.

The hard part: And conversely.

Inhabitation for intersection types

Theorem

The inhabitation problem for intersection types is undecidable.

Proof: Reduction from definability.

Encode elements d of $\mathfrak{M}(X)$ as intersection types τ_d:

- If $d \in D_0$ then $\tau_d = d$.
- If $d \in D_{\alpha\to\beta}$ then $\tau_d = \bigcap_{e\in D_\alpha}(\tau_e \to \tau_{de})$.

For a valuation v in $\mathfrak{M}(X)$, take $\Gamma_v(x) = \tau_{v(x)}$.

Main Lemma: $[\![M]\!]_v = d$ iff $\Gamma_v \vdash M : \tau_d$.

Higher-order unification

Problem: Given Church-style (normal) terms M, N of the same type with some variables $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$ called *unknowns*. (There may be other variables called *constants* or *parameters*.) Are there terms $P_1^{\sigma_1}, \ldots, P_n^{\sigma_n}$ such that

$$M[x_1 := P_1, \ldots x_n := P_n] =_{\beta\eta} N[x_1 := P_1, \ldots x_n := P_n]?$$

Non-essential generalization to finite systems of equations.

Rząd (order)

Rząd typu:

- $order(p) = 0$;
- $order(\sigma \to \tau) = \max\{order(\sigma) + 1, order(\tau)\}$.

Uwaga:

$order(\sigma_1 \to \cdots \to \sigma_n \to p) = 1 + \max\{order(\sigma_n) \mid n = 1, \ldots, n\}$

Unification *of order n* has unknowns of order $n - 1$.

Undecidability

Theorem 1 (C.L. Lucchesi'72, G. Huet'73):
The third-order unification is undecidable.

X problem Hilberta

Twierdzenie (Matjasiewicz)

Następujący problem jest nierozstrzygalny:
Dane wielomiany $w_1(\vec{x})$ i $w_2(\vec{x})$ o współczynnikach naturalnych. Czy równanie $w_1(\vec{x}) = w_2(\vec{x})$ ma rozwiązanie w liczbach naturalnych?

Nierozstrzygalność unifikacji 3. rzędu

Dane równanie $w_1(\vec{x}) = w_2(\vec{x})$ o współczynnikach naturalnych.

Wielomiany w_1 i w_2 są definiowalne w typach prostych termami W_1, W_2 typu $\omega^k \to \omega$.

Rozwiązanie równania istnieje wtedy i tylko wtedy, gdy istnieją liczebniki $\vec{n}$ spełniające równość $W_1\vec{n} =_{\beta\eta} W_2\vec{n}$.

To jest unifikacja 3. rzędu, bo niewiadome są typu

$$\omega = (p \to p) \to p \to p.$$

Undecidability

Theorem 1 (C.L. Lucchesi'72, G. Huet'73):
The third-order unification is undecidable.

Theorem 2 (W. Goldfarb'81):
The second-order unification is undecidable.

Theorem 3 (A. Schubert'97):
Nierozstrzygalność drugiego rzędu nawet wtedy, gdy niewiadome nie są zagnieżdżone.

Higher-order matching

(dopasowanie wyższego rzędu)

Higher-order matching is higher-order unification restricted to equations $M =_{\beta\eta} N$, where no unknown occurs in N.

Long known to be decidable up to order 4.

Theorem (Colin Stirling, 2007?)
Higher-order matching is decidable (over a single type atom).

Proof: Incomprehensible.

Complexity: Non-elementary (by Statman's theorem).

Theorem (Ralph Loader, 2003)
Higher-order β-matching is undecidable.

Polimorfizm

Logika pierwszego i drugiego rzędu

Logika pierwszego rzędu: kwantyfikatory wiążą tylko zmienne indywiduowe. Symbole relacyjne są stałe.

Logika drugiego rzędu: kwantyfikatory mogą wiązać zmienne relacyjne.

(Klasyczna) logika drugiego rzędu

Składnia: Zmienne relacyjne i kwantyfikatory $\forall R, \exists R$.

Przykład formuły: $\forall P(P(x) \to Q(x)) \to Q(x)$

Semantyka w stylu Tarskiego: Interpretujemy zmienne relacyjne jako relacje. Na przykład formuła

$$Nat(x) = \forall R(\forall y(R(y) \to R(sy)) \to R(0) \to R(x))$$

definiuje standardowe liczby naturalne.

Wniosek: *Aksjomaty Peana plus* $\forall x Nat(x)$ *definiują standardowy model arytmetyki z dokładnością do izomorfizmu.*

Wniosek: *Zbiór tautologii drugiego rzędu nie jest rekurencyjnie przeliczalny (bo* $\mathrm{Th}(\mathbb{N})$ *nie jest).*

Wniosek: *Nie ma pełnego systemu wnioskowania.*

Siła wyrazu języka drugiego rzędu

Przykład: Dodawanie liczb naturalnych ($m + n = k$):

$$\forall R(\forall a(Ra0a) \to \forall abc\,(Rabc \to Ra(sb)(sc)) \to Rmnk).$$

Dodawanie jest najmniejszym punktem stałym operatora:

$$R \mapsto \{\langle a, 0, a\rangle \mid a \in \mathbb{N}\} \cup \{\langle a, sb, sc\rangle \mid \langle a, b, c\rangle \in R\}.$$

Uogólnienie: Najmniejszy punkt stały operatora Φ:

$$\mathrm{LFP}_\Phi(a) := \forall R((\Phi(R) \subseteq R) \to Ra)$$

$$\mathrm{LFP}_\Phi(a) := \forall R(\forall b(\Phi(R)b \to Rb) \to Ra)$$

Tarski vs Henkin

Two ways to understand the quantifier $\forall P$:

1. Tarski: the variable P ranges over unary relations (subsets of a domain).
2. Henkin: The variable P ranges over definable *predicates*.

The set of tautologies wrt (1) is not recursively enumerable. There is no complete proof system.

We think in terms of (2).

"Dependency erasure"

Example 1: $\forall P(P(x) \to Q(x)) \to Q(x)$

This is the same principle as $\forall p(p \to q) \to q$.

Example 2: Definition of natural numbers:

$Nat(n)$: $\forall R(\forall x(R(x) \to R(sx)) \to R(0) \to R(n))$

Jak można udowodnić formułę $Nat(5)$?

Deleting first-order layer: $\forall r((r \to r) \to r \to r)$.

Second-order propositional logic

$$\text{(Ax)}\ \Gamma, \varphi \vdash \varphi$$

$$(\to \text{I})\ \frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi} \qquad (\to \text{E})\ \frac{\Gamma \vdash \varphi \to \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$

$$(\forall \text{I})\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall p\, \varphi}\ (p \notin \mathrm{FV}(\Gamma)) \qquad (\forall \text{E})\ \frac{\Gamma \vdash \forall p\, \varphi}{\Gamma \vdash \varphi[p := \vartheta]}$$

$$(\exists \text{I})\ \frac{\Gamma \vdash \varphi[p := \vartheta]}{\Gamma \vdash \exists p\, \varphi} \qquad (\exists \text{E})\ \frac{\Gamma \vdash \exists p\, \varphi, \quad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi}\ (p \notin \mathrm{FV}(\Gamma, \psi))$$

Two educated cliches

- *Full comprehension:* Propositional variables range over all formulas:

$$\exists p\,(\varphi \leftrightarrow p)$$

$$\forall p\, \varphi(p) \to \varphi(\psi)$$

  The meaning of p in $\forall p\, \varphi$ can be $\forall p\, \varphi$ itself.

- *Impredicativity:* The meaning of $\forall p\, \varphi$ is defined by a reference to a domain to which $\forall p\, \varphi$ itself belongs.

Polymorphic types

- Basic idea: $\forall p.\, p \to p$ is a type of a generic identity.
- Church style (explicit polymorphism):
  - Generic identity I admits a type argument.
  - The application $\mathsf{I}\tau$ is of type $\tau \to \tau$.
- Curry style (implicit polymorphism):
  - Generic identity I has all types $\tau \to \tau$ at once.

Girard's System F (Church style)

$$\Gamma(x : \varphi) \vdash x : \varphi$$

$$\frac{\Gamma(x:\varphi) \vdash M : \psi}{\Gamma \vdash (\lambda x{:}\varphi\, M) : \varphi \to \psi} \qquad \frac{\Gamma \vdash M : \varphi \to \psi \quad \Gamma \vdash N : \varphi}{\Gamma \vdash (MN) : \psi}$$

$$\frac{\Gamma \vdash M : \varphi}{\Gamma \vdash (\Lambda p\, M) : \forall p\, \varphi}\ (p \notin \mathrm{FV}(\Gamma)) \qquad \frac{\Gamma \vdash M : \forall p\, \varphi}{\Gamma \vdash M\vartheta : \varphi[p := \vartheta]}$$

Girard's System F (Church style)

$$\Gamma(x : \varphi) \vdash x : \varphi$$

$$\frac{\Gamma(x:\varphi) \vdash M : \psi}{\Gamma \vdash (\lambda x{:}\varphi\, M) : \varphi \to \psi} \qquad \frac{\Gamma \vdash M : \varphi \to \psi \quad \Gamma \vdash N : \varphi}{\Gamma \vdash (MN) : \psi}$$

$$\frac{\Gamma \vdash M : \varphi}{\Gamma \vdash (\Lambda p\, M) : \forall p\, \varphi}\ (p \notin \mathrm{FV}(\Gamma)) \qquad \frac{\Gamma \vdash M : \forall p\, \varphi}{\Gamma \vdash M\vartheta : \varphi[p := \vartheta]}$$

Reduction rules

Beta:

- $(\lambda x{:}\tau.M)N \Longrightarrow_\beta M[x := N]$;
- $(\Lambda p.\, M)\tau \Longrightarrow_\beta M[p := \tau]$,

A few examples (zamiast $\lambda x{:}\tau$ piszemy λx^τ)

- Term $\Lambda q \lambda x^{\forall p(p\to p)}.x(q \to q)(xq)$
  has type $\forall q(\forall p(p \to p) \to q \to q)$;
- Term $2 = \Lambda p.\lambda f^{p\to p}\lambda x^p.f(fx)$
  has type $\forall p((p \to p) \to (p \to p))$;
- Term $\lambda f^{\forall p(p\to q\to p)}\Lambda p\lambda x^p.f(q{\to}p)(fpx)$
  has type $\forall p(p \to q \to p) \to \forall p(p \to q \to q \to p)$.

Definability of integer functions

Numerals $\mathrm{n} = \Lambda p\lambda f^{p\to p}\lambda x^p.\, f(f(\cdots f(x)))$
are of type $\omega = \forall p((p \to p) \to (p \to p))$.

A function $f : \mathbb{N}^k \to \mathbb{N}$ is *definable in system* **F**
iff there is a term $F : \omega \to \cdots \to \omega \to \omega$ such that

$$F\mathrm{n}_1 \ldots \mathrm{n}_k =_\beta \mathrm{m} \quad \text{iff} \quad f(n_1, \ldots, n_k) = m.$$

Examples of representable functions:

- $Add = \lambda mn.\Lambda p.\lambda fx.mpf(npfx)$;
- $Mult = \lambda mn.\Lambda p.\lambda fx.mp(npf)x$;
- $Exp = \lambda mn.\Lambda p.\lambda fx.m(p \to p)(np)fx$.

Data types

Zbiór pusty: $x \in \varnothing \ \Leftrightarrow\ \forall P(x \in P)$.

Typ pusty (fałsz): $\bot = \forall p\, p$
Ex falso quodlibet:

If $\Gamma \vdash M : \bot$ *then* $\Gamma \vdash M\tau : \tau$.

Suma/alternatywa

$$x \in A \cup B \Leftrightarrow \forall P(A \subseteq P \to B \subseteq P \to x \in P),$$

$$A(x) \vee B(x) \Leftrightarrow \forall P(\forall y(A(y) \to P(y)) \to \forall y(B(y) \to P(y)) \to P(x)).$$

$$A \vee B = \forall p((A \to p) \to (B \to p) \to p).$$

Iloczyn/koniunkcja

$$x \in A \cap B \Leftrightarrow \forall P(\forall z(z \in A \to z \in B \to z \in P) \to x \in P).$$

$$A \wedge B = \forall p((A \to B \to p) \to p).$$

Iloczyn. . . kartezjański

$$\sigma \wedge \tau = \forall p((\sigma \to \tau \to p) \to p)$$

Pairs and projections:

- Term $\langle M^\sigma, N^\tau \rangle = \Lambda p \lambda y^{\sigma \to \tau \to p}.yMN$ has type $\sigma \wedge \tau$.
- Term $\pi_1(P^{\sigma \wedge \tau}) = P\sigma(\lambda x^\sigma \lambda y^\tau.x)$; has type σ.
- Term $\pi_2(P^{\sigma \wedge \tau}) = P\tau(\lambda x^\sigma \lambda y^\tau.y)$. has type τ.

Beta-reduction works: $\pi_1(\langle M, N \rangle) \twoheadrightarrow M$, because
$(\Lambda p \lambda y^{\sigma \to \tau \to p}.yMN)\sigma(\lambda x^\sigma \lambda y^\tau.x) \twoheadrightarrow (\lambda x^\sigma \lambda y^\tau.x)MN \twoheadrightarrow M$.

Eta-reduction does not work:

$$\langle \pi_1(P), \pi_2(P) \rangle = \Lambda p \lambda y.y(\pi_1(P))(\pi_2(P)) \not\twoheadrightarrow P$$

Coproduct

$$\sigma \vee \tau = \forall p((\sigma \to p) \to (\tau \to p) \to p)$$

Embeddings and cases:

- Term $\mathsf{inl}_{\sigma \vee \tau} L^\sigma = \Lambda p \lambda u^{\sigma \to p} \lambda v^{\tau \to p}.uL$ has type $\sigma \vee \tau$.
- Term $\mathsf{inr}_{\sigma \vee \tau} R^\tau = \Lambda p \lambda u^{\sigma \to p} \lambda v^{\tau \to p}.vR$ has type $\sigma \vee \tau$.
- And case M of $[x]P^\vartheta, [y]Q^\vartheta = M\vartheta(\lambda x^\sigma P)(\lambda y^\tau Q) : \vartheta$.

Beta-reduction:

$\mathsf{case}\ \mathsf{inl}(L)\ \mathsf{of}\ [x]P, [y]Q = (\mathsf{inl}(L))\vartheta(\lambda x^\sigma P)(\lambda y^\tau Q) =$
$(\Lambda p \lambda uv.uL)\vartheta(\lambda x^\sigma P)(\lambda y^\tau Q) \twoheadrightarrow (\lambda x^\sigma P)L \twoheadrightarrow P[x := L]$.

Jak zdefiniować kwantyfikator szczegółowy

$$x \in \bigcup_P S_P \;\Leftrightarrow\; \forall Q(\forall P(S_P \subseteq Q) \to x \in Q)$$

$$\exists p\, \sigma = \forall q(\forall p(\sigma \to q) \to q).$$

Wycieranie typów

$$\begin{aligned} |x| &= x \\ |MN| &= |M||N| \\ |\lambda x{:}\sigma.\, M| &= \lambda x.\, |M| \\ |\Lambda p.\, M| &= |M| \\ |M\tau| &= |M| \end{aligned}$$

Girard's System **F** (Church style)

$$\Gamma(x : \varphi) \vdash x : \varphi$$

$$\frac{\Gamma(x : \varphi) \vdash M : \psi}{\Gamma \vdash (\lambda x{:}\varphi\, M) : \varphi \to \psi} \qquad \frac{\Gamma \vdash M : \varphi \to \psi \quad \Gamma \vdash N : \varphi}{\Gamma \vdash (MN) : \psi}$$

$$\frac{\Gamma \vdash M : \varphi}{\Gamma \vdash (\Lambda p\, M) : \forall p\, \varphi}\ (p \notin \mathrm{FV}(\Gamma)) \qquad \frac{\Gamma \vdash M : \forall p\, \varphi}{\Gamma \vdash M\vartheta : \varphi[p := \vartheta]}$$

System F w stylu Curry'ego

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

$$\frac{\Gamma(x : \tau) \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \to \sigma} \qquad \frac{\Gamma \vdash M : \tau \to \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash NM : \sigma}$$

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M : \forall p\, \sigma}\ (p \notin \mathrm{FV}(\Gamma)) \qquad \frac{\Gamma \vdash M : \forall p\, \sigma}{\Gamma \vdash M : \sigma[p := \tau]}$$

Wycieranie redukcji

$(\lambda x{:}\tau.M)N \Longrightarrow_\beta M[x := N]$;
$(\lambda x.\,|M|)|N| \Longrightarrow_\beta |M|[x := |N|] = |M[x := N]|$

$(\Lambda p.\,M)\tau \Longrightarrow_\beta M[p := \tau]$,
$|M| = |M|$

$\lambda x{:}\tau.Mx \Longrightarrow_\eta M$;
$\lambda x.\,|M|x \Longrightarrow_\eta |M|$

$\Lambda p.Mp \Longrightarrow_\eta M$;
$|M| = |M|$

Properties of type erasure

(Church is blue, Curry is red.)

1. If $\Gamma \vdash \mathrm{M} : \tau$ then $\Gamma \vdash |\mathrm{M}| : \tau$.
2. If $\Gamma \vdash M : \tau$ then there exists M such that $|\mathrm{M}| = M$ oraz $\Gamma \vdash \mathrm{M} : \tau$.
3. If $\mathrm{M} \twoheadrightarrow_\beta \mathrm{N}$ then $|\mathrm{M}| \twoheadrightarrow_\beta |\mathrm{N}|$.
4. If $|\mathrm{M}| \to_\beta N$ then there exists N such that $|\mathrm{N}| = N$ and $\mathrm{M} \to_\beta \mathrm{N}$.

Uwaga: Tym razem część 4 nie jest oczywista, a w części 3 jest tylko $\twoheadrightarrow$, a nie $\to$.

Poprawność redukcji (Curry)

Twierdzenie:

Jeśli $\Gamma \vdash M : \tau$, *oraz* $M \to_\beta M'$ *to* $\Gamma \vdash M' : \tau$.

Dowód: Nie jest oczywisty.

Subject reduction. . . fails for η

Example:

$$x : p \to \forall q\,(q \to q) \quad \vdash \lambda y.\,xy : p \to q \to q.$$

$$x : p \to \forall q\,(q \to q) \quad \not\vdash \quad x : p \to q \to q.$$

Paradoks?

Silna normalizacja

Twierdzenie (Jean-Yves Girard, 1972)
System **F** *ma własność SN.*

Wniosek
Zdaniowa logika intuicjonistyczna drugiego rzędu jest niesprzeczna.

Silna normalizacja dla typów prostych

Termy stabilne (obliczalne):

- $[\![p]\!] := \mathrm{SN}$;
- $[\![\tau \to \sigma]\!] := \{M \mid \forall N(N \in [\![\tau]\!] \Rightarrow MN \in [\![\sigma]\!])\}$;

Naiwne uogólnienie:

- $[\![\forall p\,\tau]\!] = \bigcap\{[\![\tau[p := \sigma]]\!] \mid \sigma$ — dowolny typ$\}$.

To nie jest dobrze ufundowana definicja.

Girarda metoda kandydatów

Zbiór nasycony (*Candidat de reductibilité*)
to taki zbiór termów (Curry'ego), który ma trzy własności:

1) $X \subseteq \mathrm{SN}$.
2) Jeśli $N_1, \ldots, N_k \in \mathrm{SN}$, to $xN_1 \ldots N_k \in X$.
3) Jeśli $M[x := N_0]N_1 \ldots N_k \in X$, oraz $N_0 \in \mathrm{SN}$, to $(\lambda x.M)N_0N_1 \ldots N_k \in X$.

$\mathcal{G}$ – rodzina wszystkich kandydatów (uwaga: $\mathrm{SN} \in \mathcal{G}$).

Interpretacja Girarda

Wartościowanie $\xi : TV \to \mathcal{G}$ wyznacza interpretację typu:

$$\begin{aligned}
[\![p]\!]_\xi &= \xi(p);\\
[\![\sigma \to \tau]\!]_\xi &= \{M \mid \forall N(N \in [\![\sigma]\!]_\xi \Rightarrow MN \in [\![\tau]\!]_\xi)\};\\
[\![\forall p.\sigma]\!]_\xi &= \textstyle\bigcap_{X \in \mathcal{G}} [\![\sigma]\!]_{\xi(p \mapsto X)}.
\end{aligned}$$

Fakt: Zawsze $[\![\tau]\!]_\xi \in \mathcal{G}$.

Silna normalizacja (Curry)

Główny lemat: *Niech* $\Gamma \vdash M : \tau$, $\mathrm{FV}(M) = \{x_1, \ldots, x_k\}$.
Jeśli $N_i \in [\![\Gamma(x_i)]\!]_\xi$, *dla* $i = 1, \ldots, k$,
to $M[x_1 := N_1, \ldots, x_k := N_k] \in [\![\tau]\!]_\xi$.

Dowód twierdzenia:
Jeśli $\Gamma \vdash M : \tau$, to $M \in [\![\tau]\!]_\xi$, bo $x \in [\![\Gamma(x)]\!]_\xi$.
Teza wynika więc stąd, że $[\![\tau]\!]_\xi \subseteq \mathrm{SN}$.

Silna normalizacja (Church)

Jeśli

$$M = M_0 \to M_1 \to M_2 \to \cdots$$

to

$$|M| = |M_0| \succ |M_1| \succ |M_2| \succ \cdots$$

gdzie $\succ$ oznacza $\to$ lub $=$.

Liczba wystąpień $\to$ jest skończona, bo mamy SN (Curry).

Liczba wystąpień $=$ też, bo redukcje typowe usuwają Λ.

Zła wiadomość

Dowód Girarda nie jest wyrażalny nawet w języku arytmetyki drugiego rzędu.

Jeszcze gorsza wiadomość:

Własność SN dla systemu **F** jest NIEZALEŻNA od arytmetyki drugiego rzędu.

Siła wyrazu polimorfizmu

Twierdzenie (Girard):

Funkcja jest definiowalna w systemie **F** wtedy i tylko wtedy, gdy jest dowodliwie rekurencyjna w arytmetyce drugiego rzędu.

Idea dowodu (w uproszczeniu):

($\Rightarrow$) Silną normalizację dla termów $F\,\mathrm{n}_1 \ldots \mathrm{n}_k : \omega$ można udowodnić w arytmetyce drugiego rzędu.

($\Leftarrow$) Dowód formuły $\forall n \exists m\, P(n, m)$ „wyciera się" do termu typu $\omega \to \omega$.

The typability problem

Examples: The following terms are strongly normalizable, but untypable in system F:

- $(\lambda zy.\, y(z\mathsf{I})(z\mathsf{K}))(\lambda x.\, xx)$;
- $22\mathsf{K}$.

Nierozstrzygalność (1)

Twierdzenie (J.B. Wells, 1993)

Problem typowalności:
Dany term M, *czy istnieją takie* Γ *i* τ, *że* $\Gamma \vdash M : \tau$?
i problem sprawdzenia typu:
Dane są Γ, M *i* τ. *Czy* $\Gamma \vdash M : \tau$?
są dla systemu **F** *nierozstrzygalne.*

Nierozstrzygalność (2)

Twierdzenie (Löb, 1976, Gabbay, 1974, Sobolew, 1977)

Problem inhabitacji:
Dany typ τ, *czy istnieje term zamknięty* M *typu* τ?
jest dla systemu **F** *nierozstrzygalny.*

Uwaga: To tyle, co nierozstrzygalność intuicjonistycznej logiki zdaniowej drugiego rzędu.

Koniec