

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, ...).

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: | $a \in \{x \mid W(x)\}$ | \Leftrightarrow | $W(a)$ | $(\lambda x W(x))(a) = W(a)$ |

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

²separation

Ekstensjonalność dla zbiorów

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

$$A = \{x \mid x \in A\}. \quad (*)$$

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

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

What's wrong? The variable x in $(*)$ must be "fresh", so that A nie zależy od x .

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 . |

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.

► 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.

► 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

Przykłady

Lambda-wyrażenia:

- Zmienne x, y, z, \dots
- Aplikacje (MN);
- Abstrakcje ($\lambda x. M$).

Konwencje:

- Opuszczamy zewnętrzne nawiasy;
- Aplikacja wiąże w lewo: MNP oznacza $(MN)P$
- Skrót z kropką: $\lambda x_1 \dots x_n. M$ oznacza $\lambda x_1 (\dots (\lambda x_n M) \dots)$.

$$I = \lambda x. x$$

$$K = \lambda xy. x$$

$$S = \lambda xyz. xz(yz)$$

$$2 = \lambda fx. f(fx)$$

$$\omega = \lambda x. xx$$

$$\Omega = \omega\omega$$

$$Y = \lambda f. (\lambda x. f(xx))(\lambda x. f(xx))$$

Zmienne wolne (globalne)

$$\begin{aligned} \text{FV}(x) &= \{x\}; \\ \text{FV}(MN) &= \text{FV}(M) \cup \text{FV}(N); \\ \text{FV}(\lambda x M) &= \text{FV}(M) - \{x\}. \end{aligned}$$

Na przykład:

$$\begin{aligned} \text{FV}(\lambda x x) &= \emptyset; \\ \text{FV}(\lambda x. xy) &= \{y\}; \\ \text{FV}((\lambda x. xy)(\lambda y. xy)) &= \{x, y\}. \end{aligned}$$

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: Wyrażenia różniące się tylko wyborem zmiennych związanych utożsamiamy.

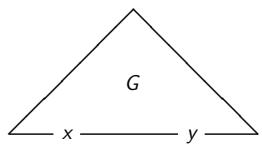
Lambda-termy to klasy abstrakcji tego utożsamienia.

Łatwiej powiedzieć, niż zrobić...

17

18

Termy jako grafy:



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

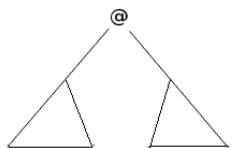
Termy jako grafy: zmienna



19

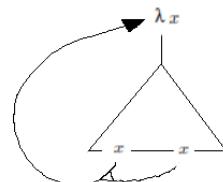
20

Termy jako grafy: aplikacja



21

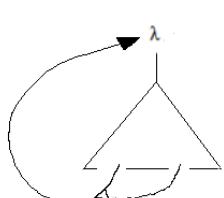
Termy jako grafy: abstrakcja



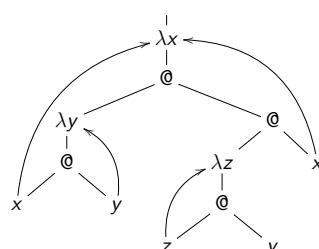
22

Termy jako grafy: abstrakcja

Zmienne związane są niepotrzebne.



Przykład:

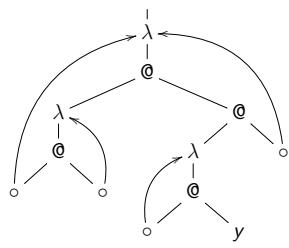


To jest graf wyrażenia $\lambda x.(\lambda y. xy)((\lambda z. zy)x)$

23

24

Przykład

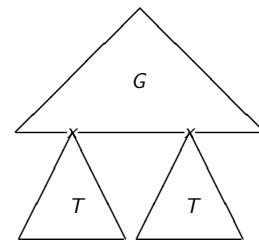


To jest graf termu $\lambda x.(\lambda y. xy)((\lambda z. zy)x)$

25

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 $y \notin FV(N)$.

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”.

25

26

Lemat o podstawieniu

Lemat

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

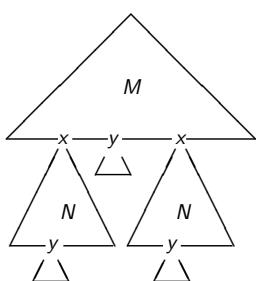
27

28

Lemat o podstawieniu

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

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



29

Beta-redukacja

Najmniejsza relacja \rightarrow_β , spełniająca warunki:

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

Term postaci $(\lambda x P)Q$ to β -redeks.

Relacja \rightarrow_β to zredukowanie jednego dowolnego redeksu.

30

Relacje pochodne:

Przykład: SKK $\rightarrow_\beta \mid$

Dowolna liczba kroków: \rightarrow_β lub \rightarrow_β^* ;

Niezerowa liczba kroków: \rightarrow_β^+ ;

Co najwyżej jeden krok: \rightarrow_β^\equiv ;

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

$$\begin{aligned} SKK &= (\lambda \textcolor{red}{x} y z. \textcolor{red}{x} z(yz))(\lambda x y. x)(\lambda x y. x) \\ &\rightarrow_\beta (\lambda \textcolor{red}{y} z. (\lambda x y. x)z(yz))(\lambda x y. x) \\ &\rightarrow_\beta \lambda z. (\lambda x y. \textcolor{red}{x})z((\lambda x y. x)z) \\ &\rightarrow_\beta \lambda z. (\lambda y. z)((\lambda x y. \textcolor{red}{x})z) \\ &\rightarrow_\beta \lambda z. (\lambda y. z)(\lambda y. z) \\ &\rightarrow_\beta \lambda z. z = \textcolor{blue}{I} \end{aligned}$$

31

32

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

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

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

$$\frac{M = N}{MP = NP} \quad \frac{M = N}{PM = PN} \quad (\epsilon) \quad \frac{M = N}{\lambda x M = \lambda x N}$$

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

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.

33

34

Kompozycjonalność

Lemat

- (1) Jeśli $M \rightarrow_{\beta} M'$, to $M[x := N] \rightarrow_{\beta} M'[x := N]$;
- (2) Jeśli $N \rightarrow_{\beta} N'$, to $M[x := N] \rightarrow_{\beta} M[x := N']$.

Dowód: Indukcja ze względu na długość M . □

Wniosek

Jeśli $M \rightarrow_{\beta} M'$ i $N \rightarrow_{\beta} N'$, to $M[x := N] \rightarrow_{\beta} M'[x := N']$.

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 ($M \in SN$), gdy nie istnieje nieskończony ciąg

$$M = M_0 \rightarrow_{\beta} M_1 \rightarrow_{\beta} M_2 \rightarrow_{\beta} \dots$$

Inaczej: każdy ciąg redukcji prowadzi do postaci normalnej.

35

36

Przykłady

Jaja aligatorów

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

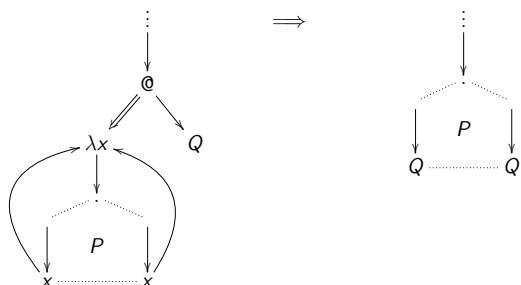
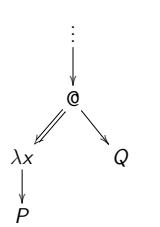
<http://worrydream.com/AlligatorEggs/>

37

38

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

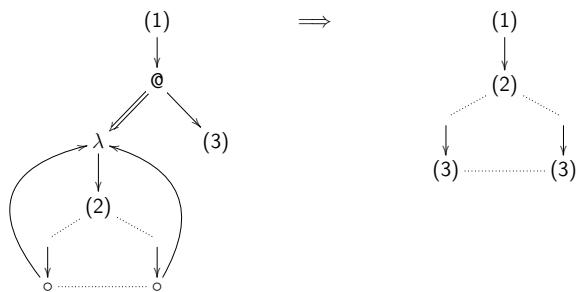
Beta-redukcja $(\lambda x P)Q \rightarrow P[x := Q]$



39

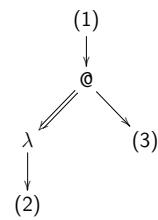
40

Beta-redukacja w grafie



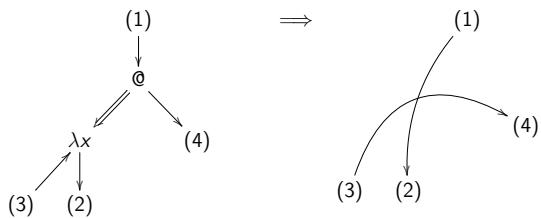
41

Beta-redeks w grafie



42

Beta-redukacja nieco wyidealizowana



Własności redukcji

43

44

Abstrakcyjne systemy redukcyjne

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

Oznaczenia:

- \rightarrow^+ domknięcie przechodnie;
- \rightarrow domknięcie przechodnio-zwrotne;
- $\rightarrow^=$ domknięcie zwrotne.

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

Inne definicje stosują się odpowiednio.

Normalizacja

Normalizacja (WN): Każdy element ma postać normalną.

Silna normalizacja (SN):

Każdy ciąg redukcji $a_0 \rightarrow a_1 \rightarrow a_2 \rightarrow \dots$ jest skończony.

(Wtedy relacja \leftrightarrow jest dobrym ufundowaniem.)

Fakt: Beta-redukacja nie ma własności normalizacji:

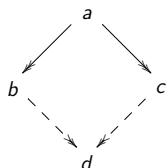
$$\Omega \xrightarrow{\beta} \Omega \xrightarrow{\beta} \Omega \xrightarrow{\beta} \dots$$

45

46

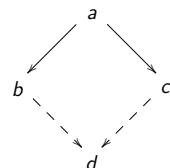
Własność Churcha-Rossera (CR)

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



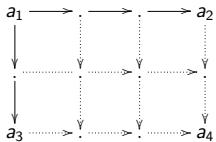
Własność rombu (diamond property)

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



47

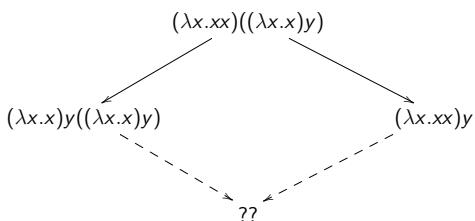
48



49

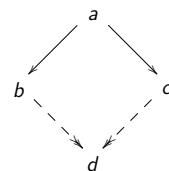
50

Własność rombu?

Fakt: Beta-redukacja nie ma własności rombu.

51

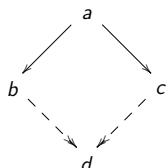
Słaba własność Churcha-Rossera (WCR)

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

Czy beta-redukacja ma słabą własność Churcha-Rossera?

52

Słaba własność Churcha-Rossera (WCR)

Jeśli $a \rightarrow b$ i $a \rightarrow c$, to istnieje takie d , że $b \rightarrow d$ i $c \rightarrow d$.**Fakt:** Beta-redukacja ma słabą własność Churcha-Rossera.

WCR nie implikuje CR

Przykład:

$$a \leftarrow b \leftarrow c \rightarrow d$$

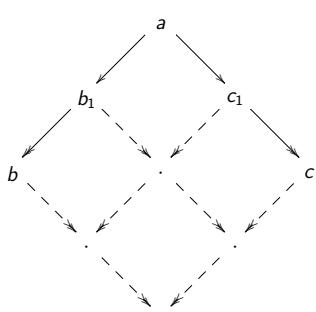
53

54

Lemat Newmana:

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

Dowód:

Indukcja ze względu na \leftarrow .

55

Twierdzenie Churcha-Rossera: Beta ma własność CR

Definicja (complete development):

Term M^* to pełne rozwiniecie termu M .

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

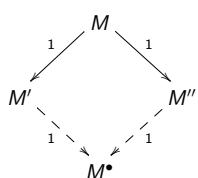
Sens: jednoczesna redukcja wszystkich istniejących redeksów.

56

- $x \xrightarrow{1} x$, 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
 $(\lambda x M)N \xrightarrow{1} M'[x := N']$.

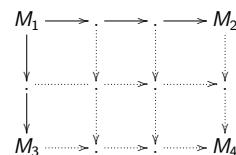
Sens: jednoczesna redukcja kilku redeksów
już obecnych w termie.

- (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

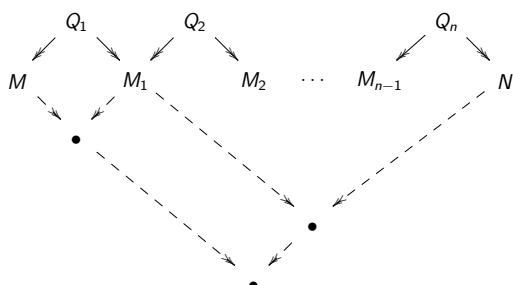
- 1) Ponieważ relacja $\xrightarrow{1}$ ma własność rombu, więc tym bardziej jej domknięcie przechodnio-zwrotne $\xrightarrow{1}$ ma własność rombu.



- 2) Ponieważ $\xrightarrow{\beta} \subseteq \xrightarrow{1} \subseteq \xrightarrow{\Rightarrow}$, więc $\xrightarrow{1}$ i $\xrightarrow{\Rightarrow}$ są równe.
- 3) Własność rombu dla $\xrightarrow{\Rightarrow}$ to własność CR dla $\xrightarrow{1}$.

Wnioski z twierdzenia Churcha-Rossera

- (1) Jeśli $M =_\beta N$, to $M \xrightarrow{\beta} Q_\beta \leftarrow N$, 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 $M \xrightarrow{\beta} N$

Dowód (3):
Jeśli $M =_\beta N$, to $M \xrightarrow{\beta} Q_\beta \leftarrow N$.
Skoro N normalne, to $N = Q$.

Wnioski z twierdzenia Churcha-Rossera

- (4) Beta-konwersja jest niesprzeczna teorią równościową

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

Eta-redukcja

Eta-reduction

The least relation \rightarrow_{η} , satisfying the conditions:

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

Symbol $\rightarrow_{\beta\eta}$ stands for the union of relations \rightarrow_{β} and \rightarrow_{η} .

Other definitions and notation are applicable respectively.

Eta-conversion

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

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

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). \square

65

66

Ekstensjonalność i słaba ekstensjonalność

$$(\text{ext}) \frac{Mx = Nx}{M = N} \quad (\text{when } x \notin FV(M) \cup FV(N))$$

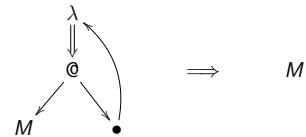
Jeśli $M = \lambda x P$ i $N = \lambda x Q$, to wystarczy:

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

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.

67

68

Eta-reduction

Fakt

Eta-reduction is strongly normalizing.

Proof.

Because terms shrink under eta. \square

Fakt

Eta-reduction is Church-Rosser.

Proof.

Because it is WCR (easy), Newman's lemma applies. \square

Beta-eta-redukcja

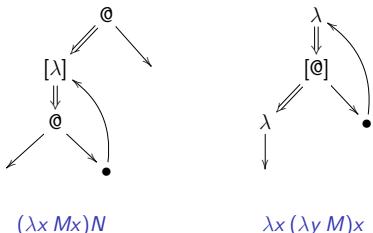
Relacja $\rightarrow_{\beta\eta}$ to suma \rightarrow_{β} i \rightarrow_{η} .

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

69

70

Beta-eta critical pairs



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

Beta-eta critical pairs

$$(\lambda x. Mx)N \rightarrow MN$$

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

Mora!

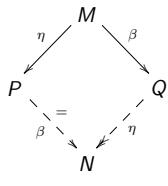
Beta-eta reduction is weakly Church-Rosser.

71

72

Lemma

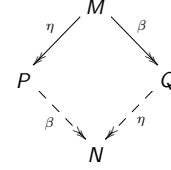
If $P \xrightarrow{\eta} M \xrightarrow{\beta} Q$ then there is N such that $P \xrightarrow{\beta} N \xleftarrow{\eta} Q$.



73

Lemma

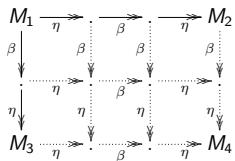
Relations $\xrightarrow{\beta}$ and $\xrightarrow{\eta}$ commute:
if $P \xrightarrow{\eta} M \xrightarrow{\beta} Q$ then there is N such that $P \xrightarrow{\beta} N \xleftarrow{\eta} Q$.



74

Beta-eta is Church-Rosser

Beta-SN versus beta-eta-SN



75

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

Proof: (\Rightarrow) Obvious.

(\Leftarrow) Ćwiczenia?

76

Eta-postponement (odkładanie eta-redukcji)

Standaryzacja

Theorem 1: If $M \xrightarrow{\beta\eta} N$ then $M \xrightarrow{\beta} P \xrightarrow{\eta} N$, 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.

77

78

Standard reduction:

Standardization

Never reduce to the left of something already reduced.

Theorem:

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

Standard reduction:

$$\begin{aligned} (\lambda x.xx)((\lambda zy.z(zy))(\lambda u.u)(\lambda w.w)) &\rightarrow \\ (\lambda x.xx)((\lambda y.(\lambda u.u)((\lambda u.u)y))(\lambda w.w)) &\rightarrow \\ (\lambda x.xx)((\lambda u.u)((\lambda u.u)(\lambda w.w))) &\rightarrow \\ (\lambda x.xx)((\lambda u.u)(\lambda w.w)) &\rightarrow (\lambda x.xx)(\lambda w.w). \end{aligned}$$

Non-standard reduction:

$$\begin{aligned} (\lambda x.xx)((\lambda zy.z(zy))(\lambda u.u)(\lambda w.w)) &\rightarrow \\ (\lambda x.xx)((\lambda y.(\lambda u.u)((\lambda u.u)y))(\lambda w.w)) &\rightarrow \\ (\lambda x.xx)((\lambda y.(\lambda u.u)y)(\lambda w.w)) &\rightarrow \\ (\lambda x.xx)((\lambda y.y)(\lambda w.w)) &\rightarrow (\lambda x.xx)(\lambda w.w). \end{aligned}$$

Slogan:

The leftmost *reduction strategy* is normalizing.

79

80

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

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

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

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

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

81

Main Lemma

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

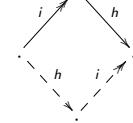
Warning: A naive proof attempt fails. The assumption:

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

does not imply

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

This diagram is wrong.



82

Dygresja

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

Lemma: If $M \twoheadrightarrow_{\beta} N$ then $M \xrightarrow{h} P \xrightarrow{i} N$, 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 wewnętrzne termów M_1, M_2, \dots, M_k . Można je łatwo przepermutować tak, aby najpierw redukować wewnętrz M_1 , potem wewnętrz M_2 i tak dalej. Do tych redukcji można zastosować indukcję... ze względu na długość N .

83

84

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.)

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

Infinitely many leftmost steps are within the same P_i .

By induction, P_i cannot normalize.

But then M cannot normalize.

85

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.)

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

Infinitely many leftmost steps are within the same P_i .

By induction, P_i cannot normalize.

But then M cannot normalize.

86

Praca domowa

1. Podać przykład takich P i Q , że ani P ani Q nie ma postaci normalnej, ale PQ ma postać normalną.
2. Udowodnić, że jeśli $M \xrightarrow{i} M'$ i $N \xrightarrow{i} N'$, to $M[x := N] \xrightarrow{i} M'[x := N']$.

85

The λI -calculus

The λI -terms:

- variables;
- applications (MN), where M and N are λI -terms;
- abstractions ($\lambda x M$), where M is a λI -term, and $x \in FV(M)$.

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

87

88

Question

Theorem: If a λI -term has a normal form then it is SN.

(Write \rightarrow^ℓ for leftmost beta-reduction.)

Are λI -terms closed under reductions?

Yes, if $M \rightarrow_{\beta\eta} N$, and M is a λI -term, then so is N .

Ale właściwie dlaczego? Jak to udowodnić?

Zróbmy to na tablicy.

89

90

Theorem: If a λI -term has a normal form then it is SN.

Question

Lemma: If $M \in \lambda I$ and $M \rightarrow^\ell N \in \text{SN}$ then $M \in \text{SN}$.

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

Case 2:

$$M = \lambda \vec{x}. (\lambda y P) Q R_1 \dots R_k \xrightarrow{\beta} \lambda \vec{x}. P[y := Q] R_1 \dots R_k = N.$$

Rozpatrzmy dowolną redukcję terminu M .

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

$$M \xrightarrow{i} \lambda \vec{x}. (\lambda y P') Q' R'_1 \dots R'_k \xrightarrow{h} \lambda \vec{x}. P'[y := Q'] R'_1 \dots R'_k \rightsquigarrow \dots$$

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

91

Does the proof use the assumption that M is a λI -term?
Where?

Yes, otherwise Q is not necessarily normal.

W istocie mamy taką obserwację:

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

92

Theorem: If a λI -term has a normal form then it is SN.

Jeszcze jedno proste ćwiczenie

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.

93

94

Curry's fixed point combinator \mathbf{Y}

Siła wyrazu rachunku lambda

$$\mathbf{Y} = \lambda F. (\lambda x. f(xx))(\lambda x. f(xx))$$

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

Proof: $\mathbf{Y}F \rightarrow_{\beta} (\lambda x. F(xx))(\lambda x. F(xx)) \xrightarrow{\beta} F((\lambda x. F(xx))(\lambda x. F(xx))) \xrightarrow{\beta} F(\mathbf{Y}F)$

95

96

Comment

$$YF =_{\beta} F(YF)$$

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

Example: Find an M such that $Mxy =_{\beta} Mxxym$.

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

Solution: No problem, $M = Y(\lambda m. \lambda xy. mxyym)$.

The price we pay for it is non-termination.

97

98

Turing's fixed-point combinator

$$\Theta = (\lambda xf. f(xxf))(\lambda xf. f(xxf))$$

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

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

Note the \rightarrow_{β} 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 $\{P = QSP, Q = PKQP\}$?

99

100

Example

Consider a combinator **day** = hour hour ··· hour, where each **hour** is defined as $\lambda abcde\text{fghijklmnpstuwyzr}. r$ (it is a fixpoint combinator).

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

Do domu: dobry przykład po polsku.

Ordered pair

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

It works:

$$\Pi_1 \langle M, N \rangle \rightarrow_{\beta} \langle M, N \rangle \pi_1 \rightarrow_{\beta} M.$$

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

$$\begin{aligned} \text{true} &= \lambda xy. x & \text{false} &= \lambda xy. y \\ \text{if } P \text{ then } Q \text{ else } R &= PQR. \end{aligned}$$

It works:

$$\begin{aligned} \text{if true then } Q \text{ else } R &\rightarrow_{\beta} Q \\ \text{if false then } Q \text{ else } R &\rightarrow_{\beta} R. \end{aligned}$$

103

104

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

$a_i := \lambda x_1 \dots x_n. x_i$

Define Boolean logic conjectives: $\vee, \wedge, \rightarrow, \neg$ as λ -terms.For example, define a term C so that

$C \text{ true false} =_{\beta} C \text{ false true} =_{\beta} C \text{ false false} = \text{false}$

$C \text{ true true} =_{\beta} \text{true}$

Selection operator:

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

It works:

$\text{case } a_i \text{ of } (F_1, \dots, F_n) \rightarrow_{\beta} F_i$

105

106

Ordered tuples

Tuple = Selector:

$$\langle M_1, \dots, M_n \rangle = \lambda x. xM_1 \dots M_n;$$

$$\begin{aligned} \pi_i &= \lambda x_1 \dots x_n. x_i & (i = 1, \dots, n); \\ \Pi_i &= \lambda t. t\pi_i & (i = 1, \dots, n). \end{aligned}$$

It works:

$\Pi_i \langle M_1, \dots, M_n \rangle \rightarrow_{\beta} M_i$

Church's numerals

$$c_n = n = \lambda f x. f^n(x),$$

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

107

108

Definable functions

Comment

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

- If $f(n_1, \dots, n_k) = m$, then $Fn_1 \dots n_k =_{\beta} m$;
- If $f(n_1, \dots, n_k)$ is undefined then $Fn_1 \dots n_k$ does not normalize.

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

- If $f(n_1, \dots, n_k)$ is defined then $Fn_1 \dots n_k \in \text{SN}$.

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.

109

110

Some well-definable functions

- Successor: $\text{succ} = \lambda n f x. f(nfx);$
We have $\text{succ } n \rightarrow_{\beta} n + 1$, please check.
Now try to define addition!

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

- Addition: $\text{add} = \lambda mnfx. m(mf(nfx));$
- Multiplication: $\text{mult} = \lambda mnfx. m(mf(nfx))x;$
- Exponentiation: $\text{exp} = \lambda mnfx. m(mf(nfx));$
- Test for zero: $\text{zero} = \lambda m. m(\lambda y. \text{false})\text{true};$
- Constant zero (with k arguments): $Z_k = \lambda m_1 \dots m_k. 0;$
- Projection on the i -th coordinate: $\Pi_k^i = \lambda m_1 \dots m_k. m_i.$

A challenge: predecessor

Digression: 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.

111

112

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

$$\begin{aligned} \text{Step} &= \lambda p. (\text{succ}(p\pi_1), p\pi_1) \\ \text{pred} &= \lambda n. (n\text{Step}(0, 0))\pi_2 \end{aligned}$$

How it works:

$$\begin{aligned} \text{Step}\langle 0, 0 \rangle &\rightarrow_{\beta} \langle 1, 0 \rangle \\ \text{Step}\langle 1, 0 \rangle &\rightarrow_{\beta} \langle 2, 1 \rangle \\ \text{Step}\langle 2, 1 \rangle &\rightarrow_{\beta} \langle 3, 2 \rangle, \end{aligned}$$

and so on.

(Evaluate pred 4.)

Some of the problems have solutions in the file zadlam.

1. Define the function $n \mapsto 2n^2 + 2n + 4$.
2. Let $f(0, x) = g(x)$ and $f(n+1, x) = h(n, x, f(n, x))$. Assume h and g are lambda-definable, show how to lambda-define f .
3. Define the function $n \mapsto 2n^2 - 2n + 4$.

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

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 " $wvBB\dots$ " (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 $q \in \text{Final}$. The result is ℓ .

Initial configuration: $C_n = \langle \varepsilon, q_0, \bullet^n, 0 \rangle$.

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

Encoding a Turing Machine

Encoding a Turing Machine

The set of states $\{q_0, q_1, \dots, q_r\}$ is finite.

States are encoded as projections: $q_i = \lambda x_0 \dots x_r. x_i$.

The alphabet $\{a_0, a_1, \dots, a_m\}$ is finite.

Symbols are encoded as projections: $a_i = \lambda x_0 \dots x_m. x_i$.

Empty word ε is encoded as $nil = \langle B, \mathbf{l} \rangle$.

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

List manipulation: $head = \Pi_1$, $tail = \Pi_2$.

(Check how it works)

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

Initial configuration: $Init = \lambda x. (nil, q_0, x(\lambda y. \langle \bullet, y \rangle) nil, 0)$.
Then $Init n \rightarrow \langle nil, q_0, \bullet^n, 0 \rangle$.

Stop test: $Halt = \lambda c. \Pi_2^4 cd_0 \dots d_r = \text{case } \Pi_2^4 c \text{ of } (d_0, \dots, d_r)$, where $d_i = \text{true}$, for $q_i \in \text{Final}$, and $d_i = \text{false}$, otherwise.

Result extraction: $Out = \lambda c. \Pi_4^4 c$.
(Check how it works)

Encoding a machine step

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

We need a term Next s.t. $\text{Next}(\text{code of } C_1) \rightarrow_{\beta} (\text{code of } C_2)$ whenever the machine can move from C_1 to C_2 . We take:

$\text{Next} = \lambda c. \text{case } \Pi_2^4 c \text{ of } (R^0, \dots, R^r) = \lambda c. \Pi_2^4 c R^0 \dots R^r$,
where R^i codes the configuration to be obtained if the machine state in the given configuration c is q_i .

121

Encoding a machine step

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

We take $\text{Next} = \lambda c. \Pi_2^4 c R^0 \dots R^r$, 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:

$$R^i = \lambda c. \text{head}(\Pi_3^4 c) R_1^i \dots R_m^i,$$

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

122

Encoding transitions

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

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

$$R_j^i = \langle \Pi_1^4 c, p, \langle b, \text{tail}(\Pi_3^4 c), \Pi_4^4 c \rangle \rangle$$

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

$$R_j^i = \langle \langle b, (\Pi_1^4 c) \rangle, p, \text{tail}(\Pi_3^4 c), \text{succ}(\Pi_4^4 c) \rangle,$$

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

$$R_j^i = \langle \text{tail}(\Pi_1^4 c), p, \langle \text{head}(\Pi_3^4 c), \langle b, (\text{tail}(\Pi_3^4 c)) \rangle \rangle, \text{pred}(\Pi_4^4 c) \rangle.$$

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

123

Special case: $a_j = B$

Let $C = \langle w^R, q_i, a_j v, n \rangle$ oraz $a_j = B$.

Then $a_j v$ is "empty", i.e. $\Pi_3^4 c = \text{nil} = \langle B, \text{I} \rangle$.

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

$$R_j^i = \langle \Pi_1^4 c, p, \langle b, \text{nil} \rangle, \Pi_4^4 c \rangle$$

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

$$R_j^i = \langle \langle b, (\Pi_1^4 c) \rangle, p, \text{nil}, \text{succ}(\Pi_4^4 c) \rangle,$$

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

$$R_j^i = \langle \text{tail}(\Pi_1^4 c), p, \langle \text{head}(\Pi_1^4 c), \langle b, \text{nil} \rangle \rangle, \text{pred}(\Pi_4^4 c) \rangle.$$

124

Encoding a Turing Machine

We have a term Next s.t.

$$\text{Next}(\text{code of } C_1) \rightarrow_{\beta} (\text{code of } C_2),$$

whenever the machine can move from C_1 to C_2 .

Teraz trzeba powtarzać Next aż do skutku.

Rozwiązań 1: Definiujemy taką funkcję F , że:
 $F(c) = \text{if } \text{Halt } c \text{ then } \text{Out } c \text{ else } F(\text{Next } c)$

The function computed by the machine is λ -definable as
 $\lambda x. F(\text{Init } x)$.

125

Rozwiązań 2

We have a term Next s.t.

$$\text{Next}(\text{code of } C_1) \rightarrow_{\beta} (\text{code of } C_2),$$

whenever the machine can move from C_1 to C_2 .

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

$$\lambda x. W(\text{Init } x)W,$$

where

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

126

Computing: $C_0 \Rightarrow C_1 \Rightarrow C_2 \Rightarrow \dots \Rightarrow C_m$

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

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

$$\begin{aligned} Fn &\rightarrow W(\text{Init } n)W \rightarrow WC_0W \rightarrow \\ &[\text{if } \text{Halt } C_0 \text{ then } \lambda w. \text{Out } C_0 \text{ else } \lambda w. w(\text{Next } C_0)w]W \rightarrow \\ &[\lambda w. w(\text{Next } C_0)w]W \rightarrow [\lambda w. wC_1w]W \rightarrow WC_1W \rightarrow \\ &[\text{if } \text{Halt } C_1 \text{ then } \lambda w. \text{Out } C_1 \text{ else } \lambda w. w(\text{Next } C_1)w]W \rightarrow \\ &[\lambda w. w(\text{Next } C_1)w]W \rightarrow [\lambda w. wC_2w]W \rightarrow WC_2W \rightarrow \\ &\dots \dots \dots \rightarrow WC_mW \rightarrow \\ &[\text{if } \text{Halt } C_m \text{ then } \lambda w. \text{Out } C_m \text{ else } \lambda w. w(\text{Next } C_m)w]W \rightarrow \\ &[\lambda w. w(\text{Next } C_m)w]W \rightarrow Out C_m \rightarrow f(n). \end{aligned}$$

127

Question

We have defined $F = \lambda x. W(\text{Init } x)W$, and

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

Suppose we replace this definition by:

$$W = \lambda cw. \text{if } \text{Halt } c \text{ then } \text{Out } c \text{ else } w(\text{Next } c)w.$$

Will anything change?

Answer: yes, the term WC_0W would never be SN.

128

Twierdzenie

A partial function $f : \mathbb{N} \rightarrow \mathbb{N}$ 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 Fn to normal form.

129

Twierdzenie

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

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

- If $f(n)$ is defined then Fn reduces to $f(n)$.
- Otherwise we have an infinite quasi-leftmost reduction:

$$Fn \rightarrow WC_0 W \rightarrow [\lambda w.wC_1w]W \rightarrow WC_1 W \rightarrow [\lambda w.wC_2w]W \rightarrow WC_2 W \rightarrow \dots$$

Therefore Fn does not normalize.

130

The following are undecidable problems:

- Given M and N , does $M \rightarrow_\beta N$ 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.,

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

Let H lambda-define the function χ_A .

To decide if $n \in A$, ask any of the questions:

- does $Hn \rightarrow_\beta 0$ hold?
- does $Hn =_\beta 0$ hold?
- does Hn have a normal form?

131

Undecidability

The following problem is undecidable:

- Given M , does $M \in SN$ hold?

Proof as before. To decide if $\chi_A(n)$ is defined, ask if Hn is strongly normalizable.

But that requires a stronger result:

Every partial computable function is well-definable.

Our coding has this property (proof omitted).

132

Some numbering tricks

Let n be the number of a term M . Write \bar{M} for n .

This is the Church numeral for the number of M .

There are computable functions app and num such that

$$\begin{aligned} app(\text{number of } M)(\text{number of } N) &= \text{number of } MN \\ num(n) &= \text{number of the numeral } n. \end{aligned}$$

There are terms App and Num such that

$$App \bar{M} \bar{N} =_\beta \bar{MN} \quad \text{and} \quad Num n =_\beta \bar{n}.$$

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

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

So \bar{M} is the Church numeral for...

the number of the numeral for the number of M .

133

A fixed-point theorem**Theorem**

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

Proof

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

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

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

(Compare $X = Z\bar{Z}$ to $Yf = (\lambda x. f(xx))(\lambda x. f(xx))$.)

134

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.

135

Proof idea

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

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

(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 $N = YG$. 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.

136

Proof of Rice's theorem

Suppose A is decidable, nontrivial, and closed under $=_\beta$.

Let F define the total characteristic function of A :

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

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 $G(\bar{N}) =_\beta N$.

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

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

Równość w rachunku lambda

137

138

Adding equational axioms

Example 1: Recall that $1 = \lambda f x. f x$. Add the axiom $I = 1$ to the equational theory of λ -calculus.

Then, for every M , one proves:

$$M = I M = 1 M = \lambda x. M x.$$

Example 2: Now add the axiom $K = S$.

Then, for every M , one proves:

$$M = S I (K M) I = K I (K M) I = I.$$

This extension is inconsistent.

Böhm's Theorem

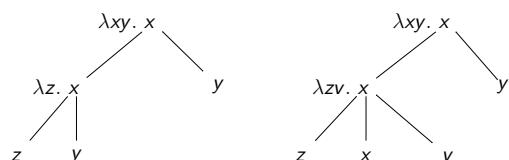
Let M, N be β -normal combinators with $M \neq_{\beta\eta} N$. Then $M\bar{P} =_\beta \text{true}$ and $N\bar{P} =_\beta \text{false}$, for some \bar{P} .

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

139

140

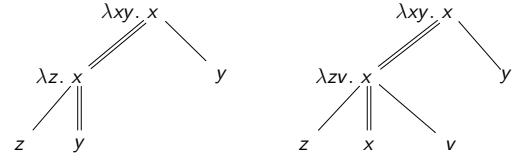
Böhm Trees (finite case)



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

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

Böhm Trees: the difference



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

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

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

141

142

Discriminating terms

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

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

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

$$\langle \lambda z. \langle z, Q \rangle, Q \rangle$$

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

Next apply both to **true**, **I**, **false** to obtain:

$$\begin{array}{ll} \lambda z. \langle z, Q \rangle & \lambda zv. \langle z, P \rangle v \\ (\text{I}, Q) & \lambda v. \langle \text{I}, P \rangle v \\ Q & P = \lambda u v. \langle u, v \rangle \end{array}$$

Choose $Q = \lambda u v v. \text{true}$ and apply both sides to **false**, **I**, **true**:

true

false.

Twierdzenie Böhma: szkic dowodu

► Tuple creation operator

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

► Substitution $S = [x_i := T_{p_i}]_{i=1, \dots, n}$ is an **m-substitution**, when $p_1, \dots, p_n > m$, and all numbers p_i are different.

► Terms M i N are **m-discriminable**, when for every m -substitution S there are combinators \bar{L} with:

$$M[S]\bar{L} \rightarrow_\beta \text{true} \quad \text{and} \quad N[S]\bar{L} \rightarrow_\beta \text{false}.$$

143

144

If M i N are normal and $M \neq_{\beta\eta} N$,
then M i N are m -discriminable for almost every m .

Czyli:

for every m -substitution S , there are \vec{L} , s.t.

$$M[S]\vec{L} \xrightarrow{\beta} \text{true} \quad \text{oraz} \quad N[S]\vec{L} \xrightarrow{\beta} \text{false}.$$

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 $M = xP_1 \dots P_k$ i $N = yQ_1 \dots Q_\ell$, bo:

- ▶ rozróżnić $\lambda x M$ i $\lambda x N$ to samo, co rozróżnić M i N ;
- ▶ jak zabraknie lambdy, to zamiast M bierzemy $\lambda x Mx$.

□

145

146

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 $M = xP_1 \dots P_k$ i $N = yQ_1 \dots Q_\ell$. Wtedy:

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

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

Przypadki bazowe: $x \neq y$ lub $x = y$ ale $k \neq \ell$: ćwiczenia.

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.

□

The standard theory

147

148

The Meaning of "Value" and "Undefined"

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

Can we identify all such terms?

No: for instance $\lambda x.xK\Omega = \lambda x.xS\Omega$ implies $K = S$
(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 (rozwiążalność)

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

If $\text{FV}(M) = \vec{x}$ 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 $M\vec{P} =_{\beta} I$ then $M\vec{P} \xrightarrow{\beta} I$. If $M\vec{P}$ head normalizes
then also M must head normalize.

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

149

150

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 $\text{FV}(M) \cup \text{FV}(N) = \vec{x}$, 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" $C[]$:

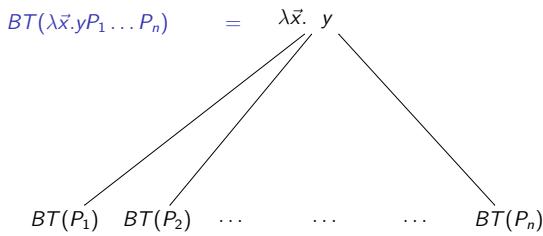
$$C[M] \text{ is solvable} \iff C[N] \text{ is solvable}$$

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

151

152

Böhm Trees



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

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

Example 1: any fixed-point combinator Z

Since $Zf =_{\beta} f(Zf)$ for a fresh variable f , it must be the case that $Z =_{\beta} \lambda f. fT$, for some T . Therefore:
 $fT =_{\beta} Zf =_{\beta} f(Zf) = f((\lambda f. fT)f) =_{\beta} f(fT)$.
Hence $fT =_{\beta} f(fT) =_{\beta} f(f(fT)) =_{\beta} f(f(f(fT))) \dots$
The term Z unfolds into an “infinite term” $\lambda f. f(f(f(\dots$

Its Böhm tree has one infinite branch:

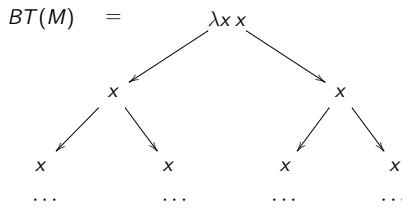
$$\lambda f. f \longrightarrow f \longrightarrow f \longrightarrow \dots$$

153

154

Example 2: $M = Y(\lambda mx. x(mx)(mx))$

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



155

Example 3: $J = Y(\lambda fxy. x(fy))$

Write Φ for $\lambda fxy. x(fy)$. Then:

$$\begin{aligned} J &= Y\Phi =_{\beta} \Phi J =_{\beta} \lambda xy_0. x(Jy_0) =_{\beta} \lambda xy_0. x(\Phi Jy_0) \\ &=_{\beta} \lambda xy_0. x(\lambda y_1. y_0(Jy_1)) =_{\beta} \lambda xy_0. x(\lambda y_1. y_0(\Phi Jy_1)) =_{\beta} \dots \end{aligned}$$

The tree $BT(J)$ consists of one infinite path:

$$\lambda xy_0. x \longrightarrow \lambda y_1. y_0 \longrightarrow \lambda y_2. y_1 \longrightarrow \lambda y_3. y_2 \longrightarrow \dots$$

156

Example 3: $J = Y(\lambda fxy. x(fy))$

The tree $BT(J)$ consists of one infinite path:

$$\lambda xy_0. x \longrightarrow \lambda y_1. y_0 \longrightarrow \lambda y_2. y_1 \longrightarrow \lambda y_3. y_2 \longrightarrow \dots$$

The tree $BT(I)$ 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:

$$\lambda x x \unders{\eta}{\leftarrow} \lambda xy_0. x \longrightarrow y_0 \unders{\eta}{\leftarrow} \lambda xy_0. x \longrightarrow \lambda y_1. y_0 \longrightarrow y_1$$

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:

$$B = B_0 \unders{\eta}{\leftarrow} B_1 \unders{\eta}{\leftarrow} B_2 \unders{\eta}{\leftarrow} B_3 \unders{\eta}{\leftarrow} \dots$$

$$B' = B'_0 \unders{\eta}{\leftarrow} B'_1 \unders{\eta}{\leftarrow} B'_2 \unders{\eta}{\leftarrow} B'_3 \unders{\eta}{\leftarrow} \dots$$

converging to the same (possibly infinite) tree.

Theorem (Wadsworth)

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

157

158

Combinatory Logic, or the calculus of combinators

Logika kombinatoryczna Combinatory Logic

Terms:

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

159

160

- $KFG \rightarrow_w F$;
- $SFGH \rightarrow_w FH(GH)$;
- If $F \rightarrow_w G$, then $FH \rightarrow_w GH$ and $HF \rightarrow_w HG$.

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

161

- $I = SKK$
 $IF \rightarrow_w KF(KF) \rightarrow_w F$
- $\Omega = SII(SII)$
 $\Omega \rightarrow_w I(SII)(I(SII)) \rightarrow_w \Omega$
- $W = SS(KI)$
 $WFG \rightarrow_w SF(KIF)G \rightarrow_w SFIG \rightarrow_w FG(IG) \rightarrow_w FGG$

162

Examples of combinators

- $B = S(KS)K$
 $BFGH \rightarrow_w KSF(KF)GH \rightarrow_w S(KF)GH \rightarrow_w KFH(GH) \rightarrow_w F(GH)$
- $C = S(BBS)(KK)$
 $CFGH \rightarrow_w FHG$
- $B' = CB$
 $B'FGH \rightarrow_w G(FH)$

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

163

Examples of combinators

- $0 = KI$
 $0FG \rightarrow_w G$
- $1 = SB(KI)$
 $1FG \rightarrow_w FG$
- $2 = SB(SB(KI))$
 $2FG \rightarrow_w F(FG)$

164

Fixed point combinators

$$\begin{aligned} Y &= WS(BWB) \\ \Theta &= WI(B(SI)(WI)) \end{aligned}$$

165

Good and bad properties

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

166

From CL to lambda

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

Properties:

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

167

From CL to lambda

If $F =_w G$, then $(F)_\Lambda =_\beta (G)_\Lambda$.
This is a “morphism” from $CL/_=w$ to $\Lambda/_=_\beta$.

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

168

Twierdzenie (Jarosław Tworek, 2010)

Istnieje efektywna redukcja $\varphi : \text{CL} \rightarrow \Lambda$ o własności:

$$F =_w G, \quad \text{wtedy i tylko wtedy, gdy} \quad \varphi(F) =_\beta \varphi(G).$$

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

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

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

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

169

170

Combinatory completeness

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

Case 1: If $x \notin \text{FV}(F)$ then

$$(\lambda^*x.F)G = KFG \rightarrow_w F = F[x := G].$$

Case 2: If $F = x$ then $(\lambda^*x.F)G = IG \rightarrow_w G = x[x := G]$.

Case 3: If $F = F'F''$ then

$$\begin{aligned} (\lambda^*x.F)G &= S(\lambda^*x.F')(\lambda^*x.F'')G \rightarrow_w \\ &\rightarrow_w (\lambda^*x.F')G((\lambda^*x.F'')G) \rightarrow_w F'[x := G]F''[x := G]. \end{aligned}$$

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

Theorem (Combinatory completeness)

Given any x and F , there is H such that, for all G ,

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

171

172

Translation from CL to lambda is surjective

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 Lemma 2:

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

Moral: Translation from CL to lambda is “surjective”.

- ▶ Złożenie $(())_\Lambda$ jest identycznością.
- ▶ Złożenie $(()_\Lambda)_c$ nie jest identycznością. Na przykład
 $((K)_\Lambda)_c = S(KK)I \neq_w K$.

173

174

Bad property

Translation from CL to lambda is surjective

- ▶ $M =_\beta N$ does not imply $(M)_c =_w (N)_c$. For example:
 $(\lambda x.KIx)_c =_w S(K(KI))I \neq_w KI = (\lambda x.I)_c$.

W rzeczy samej, bo $\lambda x.KIx \rightarrow_\beta \lambda x.I$, ale:

$$(\lambda x.KIx)_c = S(\lambda^*x.(\mathbf{K}I)_c)(\lambda^*x.x) = S(K((\mathbf{K})_c(I)_c))I = S(K(S(KK)I))I =_w S(K(KI))I \neq_w KI = (\lambda x.I)_c.$$

Moral:

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

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: $M =_\beta ((M)_c)_\Lambda$.

175

176

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

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

Indeed, $\lambda^*x.KIx = S(KI)I \neq_w KI = \lambda^*x.I$.

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

177

178

Open problem

NCL - "Naive Combinatory Logic"

Define a computable translation T from lambda to CL with:

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

Terms: If F, G are terms then PFG is a term.
Write $F \Rightarrow G$ for PFG .

Formulas:

- Every term F is a formula;
- Equations $F = G$ are formulas.

179

180

NCL - Axioms and rules

Axioms:

$$\begin{array}{lll} F = F & KFG = F & SFGH = FH(GH) \\ F \Rightarrow F & (F \Rightarrow (F \Rightarrow G)) \Rightarrow (F \Rightarrow G) & \end{array}$$

Rules:

$$\begin{array}{ll} \frac{M = N}{MQ = NQ} & \frac{M = N}{QM = QN} \\ \frac{M = N}{N = M} & \frac{M = N, N = Q}{M = Q} \\ \\ \frac{M, \quad M = N}{N} & \frac{M, \quad M \Rightarrow N}{N} \end{array}$$

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 $N \Rightarrow N$ it follows 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.

181

182

Modele

Semantics

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

More precisely, $\llbracket M \rrbracket$ may depend on a *valuation*:

$$v : Var \rightarrow A.$$

Write $\llbracket M \rrbracket_v$ for the value of M under v .

183

184

Application \cdot is a binary operation in A ;

$$[\] : \Lambda \times A^{\text{Var}} \rightarrow 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 $v|_{\text{FV}(P)} = u|_{\text{FV}(P)}$, then $[\![P]\!]_v = [\![P]\!]_u$.

185

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.

Należy wymagać $[\![P]\!]_{v[x \mapsto a]} = [\![Q]\!]_{v[x \mapsto a]}$, dla $a \in A$.

186

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 .

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:

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

Lambda-model: Weakly extensional lambda-interpretation:

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

187

188

“Trivial” examples

$$\mathfrak{M}(\lambda) = \langle \Lambda / _{=_\beta}, \cdot, [\] \rangle,$$

where $[\![M]\!]_v = [M[\vec{x} := v(\vec{x})]]_{\beta}$ 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.

“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)

189

190

Very Important Lemma

Lemma

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

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

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

$$\begin{aligned} & [\![\lambda y. P[x := N]]\!]_{v[x \mapsto [\![N]\!]_v]} \cdot a = [\![\lambda y. P[x := N]]\!]_v \cdot a \\ & = [\![P[x := N]]\!]_{v[y \mapsto a]} = [\![P]\!]_{v[y \mapsto a][x \mapsto [\![N]\!]_{v[y \mapsto a]}]} \\ & = [\![P]\!]_{v[y \mapsto a][x \mapsto [\![N]\!]_v]} = [\![\lambda y. P]\!]_{v[x \mapsto [\![N]\!]_v]} \cdot a, \text{ for all } a. \end{aligned}$$

Therefore $[\![\lambda y. P[x := N]]\!]_v = [\![\lambda y. P]\!]_{v[x \mapsto [\![N]\!]_v]}$.

Very Important Lemma

Lemma

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

Proof.

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

$$\begin{aligned} & [\![M[x := N]]\!]_v = [\![M[x := z][z := N]]\!]_v = [\![M[x := z]]\!]_w = \\ & [\![M]\!]_{w[x \mapsto z]} = [\![M]\!]_{v[x \mapsto z]} = [\![M]\!]_{v[x \mapsto [\![N]\!]_v]}. \end{aligned} \quad \square$$

191

192

Proposition

Every lambda-model is a lambda-algebra:

$$M =_{\beta} N \text{ implies } \llbracket M \rrbracket_v = \llbracket N \rrbracket_v$$

Proof.

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

(Beta)

$$\llbracket (\lambda x. P) Q \rrbracket_v = \llbracket \lambda x. P \rrbracket_v \cdot \llbracket Q \rrbracket_v = \llbracket P \rrbracket_{v[x \mapsto \llbracket Q \rrbracket_v]} = \llbracket P[x := Q] \rrbracket_v.$$

(Xi) Let $P =_{\beta} Q$ and let $M = \lambda x. P$, $N = \lambda x. Q$. Then

$$\llbracket M \rrbracket_v \cdot a = \llbracket P \rrbracket_{v[x \mapsto a]} = \llbracket Q \rrbracket_{v[x \mapsto a]} = \llbracket N \rrbracket_v \cdot a, \text{ for all } a. \quad \square$$

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) \Rightarrow (2) Z definicji lambda-algebry.

(2) \Rightarrow (3) Bo każdy lambda-model jest lambda-algebrą.

(3) \Rightarrow (1) Bo $\mathfrak{M}(\lambda)$ jest lambda-modelem. \square

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 $\perp = \sup \emptyset$.

Complete partial orders**Complete partial orders**

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

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.

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

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

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

Continuous functions**Continuous functions****Lemma**

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

Lemma

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

Proof hint: Use the previous lemma.

Lemma

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

Reflexive cpo

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

Then F must be onto and G is injective.

The following are equivalent conditions:

" $G \circ F = \text{id}_D$ ", "G onto", " F injective".

Reflexive cpo

$$F : D \rightarrow [D \rightarrow D], \quad G : [D \rightarrow D] \rightarrow D, \quad F \circ G = \text{id}.$$

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

Define interpretation as

- $\llbracket x \rrbracket_v = v(x);$
- $\llbracket PQ \rrbracket_v = \llbracket P \rrbracket_v \cdot \llbracket Q \rrbracket_v;$
- $\llbracket \lambda x. P \rrbracket_v = G(\lambda a. \llbracket P \rrbracket_{v[x \mapsto a]}).$

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

201

202

Reflexive cpo

Theorem

A reflexive cpo is a lambda-model.

Proof.

Należy udowodnić słabą ekstensjonalność.

Załóżmy, że $\llbracket \lambda x. M \rrbracket_v \cdot a = \llbracket \lambda x. N \rrbracket_v \cdot a$, dla każdego a .

Inaczej, $\lambda a. \llbracket M \rrbracket_{v[x \mapsto a]} = \lambda a. \llbracket N \rrbracket_{v[x \mapsto a]}$.

Chcemy $\llbracket \lambda x. M \rrbracket_v = \llbracket \lambda x. N \rrbracket_v$. Ale:

$$\begin{aligned} \llbracket \lambda x. M \rrbracket_v &= G(\lambda a. \llbracket M \rrbracket_{v[x \mapsto a]}) \\ \llbracket \lambda x. N \rrbracket_v &= G(\lambda a. \llbracket N \rrbracket_{v[x \mapsto a]}). \end{aligned}$$

□

Reflexive cpo

The cpo D is *reflexive* iff there are continuous functions

$$F : D \rightarrow [D \rightarrow D] \text{ and } G : [D \rightarrow D] \rightarrow D, \quad \text{with } F \circ G = \text{id}_{[D \rightarrow D]}.$$

Theorem

A reflexive cpo is a lambda-model.

Ale jak takie zrobić?

203

204

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

Notation $\mathcal{P}_\omega = \mathcal{P}(\mathbb{N})$.

Every set is a directed union of its finite subsets.

Lemma

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

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

Encodings in \mathcal{P}_ω

Pairs:

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

Finite sets: $e_0 = \emptyset$, and

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

205

206

\mathcal{P}_ω is reflexive

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

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

$$\text{fun} \circ \text{graph} = \text{id}_{[\mathcal{P}_\omega \rightarrow \mathcal{P}_\omega]}.$$

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

Przykłady w \mathcal{P}_ω (ćwiczenie)

- $\llbracket I \rrbracket = \text{graph}(\text{id}) = \{(n, m) \mid m \in e_n\};$
- $\llbracket K \rrbracket = \{(m, (k, \ell)) \mid \ell \in e_m\};$
- $\llbracket \omega \rrbracket = \{(x, m) \mid \exists n (e_n \subseteq e_x \wedge (n, m) \in e_x)\};$
- $\llbracket \Omega \rrbracket = \emptyset = \perp.$

207

208

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

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

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

Proof: Let $v(x) = a$, where $a \neq \emptyset$ is finite.
 Then $\llbracket x \rrbracket_v = a$ is finite. But $\llbracket \lambda y. xy \rrbracket_v = \text{graph}(\dots)$ is infinite.

209

Theorem (Hyland)

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

210

Trzy własności modeli denotacyjnych

- ▶ **Poprawność:** Jeśli $M =_\beta N$, to $\llbracket M \rrbracket = \llbracket N \rrbracket$.
- ▶ **Adekwatność:** Jeśli $\llbracket M \rrbracket = \llbracket N \rrbracket$, to $M \equiv N$.
- ▶ **Pełna abstrakcyjność:** Jeśli $M \equiv N$, to $\llbracket M \rrbracket = \llbracket N \rrbracket$.

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

Trzy własności modeli denotacyjnych

- ▶ **Poprawność:** Jeśli $M =_\beta N$, to $\llbracket M \rrbracket = \llbracket N \rrbracket$.
- ▶ **Adekwatność:** Jeśli $\llbracket M \rrbracket = \llbracket N \rrbracket$, to $M \equiv N$.
- ▶ **Pełna abstrakcyjność:** Jeśli $M \equiv N$, to $\llbracket M \rrbracket = \llbracket N \rrbracket$.

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

211

212

Towards a fully abstract model

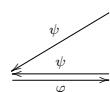
A *projection* of B onto A is a pair of continuous functions

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

such that

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

Then $\varphi(\perp_A) = \perp_B$, because $\varphi(\perp_A) \leq \varphi(\psi(\perp_B)) \leq \perp_B$.



213

Example

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

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

given by

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

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

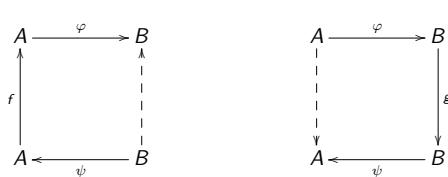
214

Raising a projection

Let (φ, ψ) be a projection of B onto A .

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

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



215

Towards D_∞

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

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

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:

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

216

$$\varphi_0(a) = \lambda x. a \quad \psi_0(f) = f(\perp)$$

$$\varphi_{n+1}(f) = \varphi_n \circ f \circ \psi_n \quad \psi_{n+1}(g) = \psi_n \circ g \circ \varphi_n.$$

$$\varphi_1(f)(g) = \lambda x. f(g(\perp)) \quad \psi_1(F)(a) = F(\lambda x. a)(\perp)$$

Jak wyglądają D_0 , D_1 , D_2 ?

217

218

Scott's D_∞

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

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

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 :

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

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

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

Convention:

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

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

219

220

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 monotone: $x_0 \leq x_1 \leq x_2 \leq \dots$ and $x = \sup x_n$.
- ▶ The bottom is unique: $\perp_{D_0} = \perp_{D_n} = \perp_{D_\infty}$.
- ▶ 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 $(x \cdot \perp)_0 = x_0$.

Proofs happily omitted.

221

222

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 $x_0 = (x \cdot \perp)_0 \leq (y \cdot \perp)_0 = y_0$.

Then $x_{n+1}(z) = (x \cdot z)_n \leq (y \cdot z)_n = y_{n+1}(z)$, for $z \in D_n$. \square

Scott's D_∞ model

Theorem

The cpo D_∞ is reflexive.

Proof.

Define $F : D_\infty \rightarrow [D_\infty \rightarrow D_\infty]$ by $F(x)(y) = x \cdot y$.

We know that F is continuous and injective.

Take any $f \in [D_\infty \rightarrow D_\infty]$.

Define $f^{(n)} : D_n \rightarrow D_n$ 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. \square

223

224

Corollary

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

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 $e_{n+1} = \text{id}_{D_n}$, czyli $e = (\perp, \text{id}, \text{id}, \dots)$

225

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

$[\text{Y}] \cdot a$ jest najmniejszym elementem b o własności $a \cdot b = b$.

Wieloznaczność w D_∞

Funkcje ciągłe

$F : D_\infty \rightarrow [D_\infty \rightarrow D_\infty]$ i $G : [D_\infty \rightarrow D_\infty] \rightarrow D_\infty$

są wzajemnie odwrotnymi izomorfizmami.

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

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

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

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

227

226

Scott's D_∞ model

Twierdzenie (Hyland, Wadsworth)

The model D_∞ is "adequate" and "fully abstract":

Terms M i N are observationally equivalent iff $D_\infty \models M = N$.

Ścislej:

Twierdzenie (Hyland, Wadsworth)

Następujące warunki są równoważne:

1. Termy M i N są obserwacyjnie równoważne.
2. $BT(M) \approx_\eta BT(N)$.
3. $D_\infty \models M = N$.

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

227

228

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 \perp .

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

Przyjmujemy, że $[\perp] = \perp_{D_\infty}$

Zbiór aproksymantów drzewa/termu:

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

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

229

230

Twierdzenie o aproksymacji

Tw. o aproksymacji: $\llbracket M \rrbracket_\rho = \sup \{\llbracket A \rrbracket_\rho \mid A \in A(M)\}$.

$$\llbracket M \rrbracket_\rho = \sup \{\llbracket A \rrbracket_\rho \mid A \in A(M)\}.$$

Intuicje (czyli mechaniki rękami):

Znaczenie $\llbracket M \rrbracket_\rho$ to nić $x \in D_\infty$, czyli w istocie ciąg funkcji $x_{n+1} : D_n \rightarrow D_n$, opisujących zachowanie $\llbracket M \rrbracket_\rho$ na skończonym zbiorze D_n elementów rzędu n .

To zachowanie jest zdeterminowane przez skończony fragment początkowy drzewa Böhma $BT(M)$.

231

Dowód: (\Rightarrow) Jeśli $M =_\beta \lambda x_1 \dots x_n. y \vec{N}$, gdzie y wolne, to $\llbracket M \rrbracket_\rho \neq \perp$, dla $\rho(y) = \lambda \vec{a}. d$, gdzie $d \neq \perp$.

Jeśli $M =_\beta \lambda x_1 \dots x_n. x_i \vec{N}$, to $\llbracket M \rrbracket_\rho \neq \perp$, dla każdego ρ . Należy użyć $\lambda \vec{a}. d$ jako i -tego argumentu dla $\llbracket M \rrbracket_\rho$.

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

232

Twierdzenie:

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

Dowód: Jeśli $\llbracket M \rrbracket_\rho = \llbracket N \rrbracket_\rho$ to także $\llbracket C[M] \rrbracket_\rho = \llbracket C[N] \rrbracket_\rho$.

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

Twierdzenie: *Jeśli $M \equiv N$, to $\llbracket M \rrbracket = \llbracket N \rrbracket$*

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

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

Zatem $\llbracket M \rrbracket = \llbracket N \rrbracket$ wynika z twierdzenia o aproksymacji.

Typy: motywacja semantyczna

Typy

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:

$$\sigma \Rightarrow \tau = \{a \in D \mid \forall b \in D (b \in \sigma \rightarrow a \cdot b \in \tau)\}.$$

Formal type assignment

Subtyping in a model

$$\sigma \cap \tau \subseteq \sigma \quad \sigma \cap \tau \subseteq \tau \quad \sigma \subseteq \sigma \cap \sigma$$

$$\sigma \subseteq \omega \quad \omega \subseteq \omega \Rightarrow \omega$$

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

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

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

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.

Subtyping (BCD)

Intersection types (formal system)

Types:

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

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

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

$$\begin{aligned} \sigma \cap \tau \leq \sigma & \quad \sigma \cap \tau \leq \tau & \sigma \leq \sigma \cap \sigma \\ \sigma \leq \omega & \quad \omega \leq \omega \rightarrow \omega \\ (\sigma \rightarrow \tau) \cap (\sigma \rightarrow \rho) \leq \sigma \rightarrow \tau \cap \rho & \end{aligned}$$

and closed under the rules:

$$\frac{\sigma \leq \sigma' \quad \tau \leq \tau'}{\sigma \cap \tau \leq \sigma' \cap \tau'} \quad \frac{\sigma \leq \sigma' \quad \tau \leq \tau'}{\sigma' \rightarrow \tau \leq \sigma \rightarrow \tau'}$$

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

Napis $\sigma \equiv \tau$ oznacza $\sigma \leq \tau \leq \sigma$.

Takie typy czasem się utożsamia.

Environments

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

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

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

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

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

Intersection type assignment (BCD)

$$\Gamma(x : \sigma) \vdash x : \sigma \text{ (Var)} \quad \Gamma \vdash M : \omega \text{ (\omega)}$$

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

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

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

BCD = Barendregt, Coppo, Dezani

241

242

Example

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

$$\frac{\begin{array}{c} z : \tau, x : \tau, y : \sigma \vdash x : \tau \\ \hline z : \tau, x : \tau \vdash \lambda y x : \sigma \rightarrow \tau \end{array}}{\begin{array}{c} z : \tau \vdash \lambda xy. x : \tau \rightarrow \sigma \rightarrow \tau \quad z : \tau \vdash z : \tau \\ \hline z : \tau \vdash (\lambda xy. x)z : \sigma \rightarrow \tau \end{array}}$$

243

Example

$$\frac{\Gamma(y : \sigma) \vdash M : \tau}{\Gamma \vdash \lambda y M : \sigma \rightarrow \tau} \text{ (Abs)} \quad \frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \text{ (App)}$$

$$\frac{\begin{array}{c} x : \tau, y : \sigma \vdash x : \tau \\ \hline x : \tau, y : \tau \vdash \lambda y x : \sigma \rightarrow \tau \end{array}}{\begin{array}{c} y : \tau \vdash \lambda xy. x : \tau \rightarrow \sigma \rightarrow \tau \quad y : \tau \vdash y : \tau \\ \hline y : \tau \vdash (\lambda xy. x)y : \sigma \rightarrow \tau \end{array}}$$

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

244

Example

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

$$\frac{\begin{array}{c} x : p \cap (p \rightarrow q) \vdash x : p \cap (p \rightarrow q) \quad x : p \cap (p \rightarrow q) \vdash x : p \cap (p \rightarrow q) \\ \hline x : p \cap (p \rightarrow q) \vdash x : p \quad x : p \cap (p \rightarrow q) \vdash x : p \rightarrow q \end{array}}{\begin{array}{c} x : p \cap (p \rightarrow q) \vdash xx : q \\ \hline \vdash \lambda x. xx : p \cap (p \rightarrow q) \rightarrow q \end{array}}$$

245

Example

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

$$\frac{\vdots}{\vdash K : p \rightarrow \omega \rightarrow p} \quad \frac{x : p \vdash x : p}{\vdash Kx : \omega \rightarrow p} \quad \frac{x : p \vdash x : p}{\vdash Kx\Omega : p}$$

246

Example

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

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

$$\frac{\Gamma \vdash y : (p \cap q \rightarrow r) \rightarrow r \quad \Gamma \vdash x : p \rightarrow r \cap s}{\Gamma \vdash yx : r} \quad \frac{\vdots}{\vdash K : (p \rightarrow q \rightarrow p) \rightarrow r \rightarrow p \rightarrow q \rightarrow p} \quad \frac{\vdots}{\vdash K : p \rightarrow q \rightarrow p}$$

247

Example

$$\frac{\vdots}{\vdash KK : r \rightarrow p \rightarrow q \rightarrow p}$$

248

Examples

- $\vdash I : t \cap s \rightarrow t;$
- $\vdash I : t \rightarrow t;$
- $\vdash 1 : (t \rightarrow s) \cap (s \rightarrow r) \rightarrow (t \rightarrow r);$
- $\vdash 2 : (t \rightarrow t) \rightarrow t \rightarrow t;$
- $\vdash K : t \rightarrow s \rightarrow t;$
- $\vdash S : (t' \rightarrow s \rightarrow r) \rightarrow (t'' \rightarrow s) \rightarrow (t' \cap t'') \rightarrow r;$
- $\vdash S : (t \rightarrow s \rightarrow r) \rightarrow (t \rightarrow s) \rightarrow t \rightarrow r.$

“Beta-soundness”

Lemma

Jeśli $\sigma \rightarrow \tau \not\equiv \omega$ oraz $\bigcap_{j \in J} p_j \cap \bigcap_{i \in I} (\sigma_i \rightarrow \tau_i) \leq \sigma \rightarrow \tau$
to $\{i \mid \sigma \leq \sigma_i\} \neq \emptyset$ oraz $\bigcap_{\sigma \leq \sigma_i} \tau_i \leq \tau$.

Proof.

Indukcja ze względu na definicję \leq . □

249

250

Generation (Inversion) Lemma

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

Proof: Induction wrt type derivation.

Example induction step: Last rule used is $(\cap l)$.
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:

$$\begin{array}{ll} \Gamma \vdash M : \tau_1 \rightarrow \sigma_1, & \Gamma \vdash N : \tau_1 \\ \Gamma \vdash M : \tau_2 \rightarrow \sigma_2, & \Gamma \vdash N : \tau_2. \end{array}$$

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

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

251

252

Correctness of substitution

Przypadek abstrakcji?

- If $\Gamma \vdash \lambda x. M : \sigma \not\equiv \omega$ then $\sigma = \bigcap_i (\sigma_i \rightarrow \tau_i)$.
(If $\sigma \not\equiv \omega$ then $\tau_i \not\equiv \omega$.)
- If $\Gamma \vdash \lambda x. M : \sigma \rightarrow \tau$ then $\Gamma(x : \sigma) \vdash M : \tau$.

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 $\tau = \mu \rightarrow \rho$ 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 $\Gamma \vdash \lambda y P[x := N] : \mu \rightarrow \rho$. □

253

254

Subject Reduction

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

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

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

By inversion $\Gamma \vdash \lambda x. P : \sigma \rightarrow \tau$ and $\Gamma \vdash Q : \sigma$.

Then $\Gamma, x : \sigma \vdash P : \tau$, whence $\Gamma \vdash P[x := Q] : \tau$

$$M = \lambda x. Nx \rightarrow_{\eta} N.$$

Then $\tau = \bigcap_i (\sigma_i \rightarrow \tau_i)$ and $\Gamma, x : \sigma_i \vdash Nx : \tau_i$, all i .

It follows that $\Gamma, x : \sigma_i \vdash N : \zeta_i \rightarrow \tau_i$ with $\sigma_i \leq \zeta_i$.

But $x \notin FV(N)$, so $\Gamma \vdash N : \bigcap_i (\zeta_i \rightarrow \tau_i) \leq \tau$.

Subject Conversion, czyli na odwrót tez

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 .

255

256

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 $M = M_1 M_2$,
z lematu o generowaniu istnieje takie ρ , że:

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

Z zał. ind. mamy takie σ_1, σ_2 , że:

$$\begin{aligned}\Gamma \vdash N : \sigma_1, & \quad \Gamma(x : \sigma_1) \vdash M_1 : \rho \rightarrow \tau, \\ \Gamma \vdash N : \sigma_2, & \quad \Gamma(x : \sigma_2) \vdash M_2 : \rho.\end{aligned}$$

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

257

258

Subject Conversion

Theorem: If $M \rightarrow_\beta N$ then in system BCD:

$$\Gamma \vdash M : \tau \quad \text{iff} \quad \Gamma \vdash N : \tau.$$

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. That won't work for eta: $x : p \not\vdash \lambda y. xy : p$

259

260

Interpreting types in a model

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

Interpretation of types:

- $\llbracket p \rrbracket_\xi = \xi(p)$, for type variable p ;
- $\llbracket \omega \rrbracket_\xi = \mathcal{D}$;
- $\llbracket \sigma \cap \tau \rrbracket_\xi = \llbracket \sigma \rrbracket_\xi \cap \llbracket \tau \rrbracket_\xi$;
- $\llbracket \sigma \rightarrow \tau \rrbracket_\xi = \llbracket \sigma \rrbracket_\xi \Rightarrow \llbracket \tau \rrbracket_\xi$.

Easy lemma: If $\sigma \leq \tau$, then $\llbracket \sigma \rrbracket_\xi \subseteq \llbracket \tau \rrbracket_\xi$.

Semantics of type assignment

Write $\mathcal{D}, v, \xi \models M : \sigma$ when $\llbracket M \rrbracket_v \in \llbracket \sigma \rrbracket_\xi$.

Write $\mathcal{D}, v, \xi \models \Gamma$ when $v(x) \in \llbracket \Gamma(x) \rrbracket_\xi$, for all $x \in \text{Dom}(\Gamma)$.

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

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

Theorem (Soundness)

$$\text{If } \Gamma \vdash M : \sigma \text{ then } \Gamma \models M : \sigma.$$

Proof: Easy induction wrt type derivation.

261

262

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

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

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

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

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

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

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

263

Model z filtrów: znaczenie terminu

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

Ale term ma zmienne wolne.

Jego znaczenie zależy od wartościowania.

Otoczenie Γ : zmienne obiektywne \longrightarrow typy
Wartościowanie v : zmienne obiektywne \longrightarrow filtry

An environment Γ is *consistent* with object valuation v when
 $\Gamma(x) \in v(x)$, for all $x \in \text{Dom}(\Gamma)$.

Interpretation:

$$\llbracket M \rrbracket_v = \{\sigma \mid \Gamma \vdash M : \sigma, \text{ for some } \Gamma \text{ consistent with } v\}.$$

264

Filter model

$F_1 \cdot F_2 = \{\tau \mid \sigma \rightarrow \tau \in F_1 \text{ for some } \sigma \in F_2\}$.
 $\llbracket M \rrbracket_v = \{\sigma \mid \Gamma \vdash M : \sigma, \text{ for some } \Gamma \text{ consistent with } v\}$.

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

- (a) $\llbracket x \rrbracket_v = v(x)$;
- (b) $\llbracket PQ \rrbracket_v = \llbracket P \rrbracket_v \cdot \llbracket Q \rrbracket_v$;
- (c) $\llbracket \lambda x. P \rrbracket_v \cdot F = \llbracket P \rrbracket_{v[x \mapsto F]}$, for any $F \in \mathcal{F}$;
- (d) If $v|_{\text{FV}(P)} = u|_{\text{FV}(P)}$, then $\llbracket P \rrbracket_v = \llbracket P \rrbracket_u$.
- (e) If $\llbracket \lambda x. M \rrbracket_v \approx \llbracket \lambda x. N \rrbracket_v$ then $\llbracket \lambda x. M \rrbracket_v = \llbracket \lambda x. N \rrbracket_v$.

Dowód opuszczamy.

265

Model z filtrów: znaczenie typu

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

$$\llbracket \tau \rrbracket_\xi = \{F \mid F \text{ filter, and } \tau \in F\}.$$

Inaczej: $\tau \in F \Leftrightarrow F \in \llbracket \tau \rrbracket_\xi$.

Dowód opuszczamy

266

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 $v(x) = \Gamma(x)\uparrow$.

Then $\Gamma(x) \in v(x)$, thus $v(x) \in \llbracket \Gamma(x) \rrbracket_\xi$.

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

Therefore $\llbracket M \rrbracket_v \in \llbracket \sigma \rrbracket_\xi$, i.e., $\sigma \in \llbracket M \rrbracket_v$.

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

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

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

267

Jeszcze dwa twierdzenia

Twierdzenie:

Następujące warunki są równoważne (dla systemu BCD):

- $\text{BT}(M) = \text{BT}(N)$;
- 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 .

268

Intersection types without omega

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

$$\begin{array}{c} \frac{\Gamma(x:\sigma) \vdash M : \tau \quad (\text{Abs})}{\Gamma \vdash \lambda x M : \sigma \rightarrow \tau} \quad \frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \text{ (App)} \\ \\ \frac{\Gamma \vdash M : \tau_1 \quad \Gamma \vdash M : \tau_2 \quad (\cap I)}{\Gamma \vdash M : \tau_1 \cap \tau_2} \quad \frac{\Gamma \vdash M : \tau_1 \cap \tau_2 \quad (\cap E)}{\Gamma \vdash M : \tau_i} \\ \\ \frac{\Gamma \vdash M : \sigma \quad [\sigma \leq \tau]}{\Gamma \vdash M : \tau} \text{ } (\leq) \end{array}$$

269

Intersection types without subtyping

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

$$\begin{array}{c} \frac{\Gamma(x:\sigma) \vdash M : \tau \quad (\text{Abs})}{\Gamma \vdash \lambda x M : \sigma \rightarrow \tau} \quad \frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \text{ (App)} \\ \\ \frac{\Gamma \vdash M : \tau_1 \quad \Gamma \vdash M : \tau_2 \quad (\cap I)}{\Gamma \vdash M : \tau_1 \cap \tau_2} \quad \frac{\Gamma \vdash M : \tau_1 \cap \tau_2 \quad (\cap E)}{\Gamma \vdash M : \tau_i} \end{array}$$

270

With or without subtyping

Theorem

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

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

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

271

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 $\Gamma \vdash M : \bigcap_i (\tau_i \rightarrow \sigma_i)$ and $\Gamma \vdash N : \bigcap_i \tau_i$.
- If $\Gamma \vdash \lambda x. M : \sigma$ then $\sigma = \bigcap_i (\sigma_i \rightarrow \tau_i)$, where $\Gamma(x : \sigma_i) \vdash M : \tau_i$.

Lemma:

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

272

Subject Reduction (bez ω i \leq)

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

Remark: No more subject conversion, e.g., $K\Omega \rightarrow I$.

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

$$\begin{array}{ll} x : (\alpha \rightarrow \beta) \cap (\alpha \rightarrow \gamma) & \vdash \lambda y. xy : \alpha \rightarrow (\beta \cap \gamma) \\ x : (\alpha \rightarrow \beta) \cap (\alpha \rightarrow \gamma) & \not\vdash x : \alpha \rightarrow (\beta \cap \gamma) \end{array}$$

Silna normalizacja

Termy typowalne nie mają nieskończonych redukcji

273

274

Strong normalization: Tait's proof method

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

- $\llbracket p \rrbracket := \text{SN}$;
- $\llbracket \tau \rightarrow \sigma \rrbracket := \{M \mid \forall N (N \in \llbracket \tau \rrbracket \Rightarrow MN \in \llbracket \sigma \rrbracket)\}$;
- $\llbracket \tau \cap \sigma \rrbracket := \llbracket \tau \rrbracket \cap \llbracket \sigma \rrbracket$.

Lemma: If $\tau \leq \sigma$, then $\llbracket \tau \rrbracket \subseteq \llbracket \sigma \rrbracket$.

Lemma:

- 1) $\llbracket \tau \rrbracket \subseteq \text{SN}$;
- 2) If $N_1, \dots, N_k \in \text{SN}$ then $xN_1 \dots N_k \in \llbracket \tau \rrbracket$.
In particular, variables are stable for every type.

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

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

- (1) Take $M \in \llbracket \sigma \rightarrow \rho \rrbracket$. By the ind. hyp. (2), we have $x \in \llbracket \sigma \rrbracket$, so that $Mx \in \llbracket \rho \rrbracket$. Thus $Mx \in \text{SN}$, by the ind. hyp. (1). Hence $M \in \text{SN}$.
- (2) We want $xN_1 \dots N_k P \in \llbracket \rho \rrbracket$, whenever $P \in \llbracket \sigma \rrbracket$.
But $P \in \text{SN}$ by the ind. hyp. (1), and we can apply ind. hyp. (2) to P .

275

276

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

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

Proof: Induction wrt τ . Base case above.

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

Let $M[x := N_0]N_1 \dots N_k \in \llbracket \sigma \rightarrow \rho \rrbracket$ and $P \in \llbracket \sigma \rrbracket$. We want $(\lambda x. M)N_0 N_1 \dots N_k P \in \llbracket \rho \rrbracket$.

Then $M[x := N_0]N_1 \dots N_k P \in \llbracket \rho \rrbracket$.

Apply induction to P with $N_{k+1} = P$.

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

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

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

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

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

That's what we need.

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

277

278

Strong Normalization

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

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

$$M = M[\vec{x} := \vec{x}] \in \llbracket \tau \rrbracket \subseteq \text{SN}.$$

Dygresja: sens moralny numeracji gödłowskiej

Natural number – prototype of finite object.

Peano Arithmetic – prototype of finitary reasoning.

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

279

280

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:

- $\llbracket p \rrbracket = \text{SN};$
- $\llbracket \tau \rightarrow \sigma \rrbracket = \{M \mid \forall N(N \in \llbracket \tau \rrbracket \Rightarrow MN \in \llbracket \sigma \rrbracket)\};$
- $\llbracket \tau \cap \sigma \rrbracket = \llbracket \tau \rrbracket \cap \llbracket \sigma \rrbracket.$

Formalizing this definition requires quantification over sets.

(Definiujemy relację „ $M \in \llbracket \tau \rrbracket$ ” 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).

281

282

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.

SN = typability

Theorem (Garrel Pottinger)

A term is typable in intersection types (without ω, \leq)
if and only if it is strongly normalizing.

283

284

Beta-expansion without ω .

SN = typability

Dowód (Betti Venneri):

Dla $M \in \text{SN}$, niech $\delta(M)$ oznacza maksymalną długość redukcji terminu M do postaci normalnej i niech $|M|$ będzie długością terminu M .

Dowodzimy, że M typowalny przez indukcję ze względu na dwa parametry $(\delta(M), |M|)$.

For $\delta(M) = 0$ we use the lemma on normal forms.

Let $\delta(M) = n > 0$.

Case 1: $M = \lambda x.N$. Ponieważ $\delta(N) \leq \delta(M)$ oraz $|N| < |M|$, więc N typowalny, zatem M też.

285

286

Proof continued

Proof continued

Case 3: $M = (\lambda x.P)QN_1 \dots N_k \rightarrow_L P[x := Q]N_1 \dots N_k = M'$.

Then $Q \in \text{SN}$ and $\delta(Q) < \delta(M)$, so Q is typable in some Γ_1 .

Also M' typable in some Γ_2 , so we have

$\Gamma_2 \vdash P[x := Q] : \rho_1 \rightarrow \dots \rightarrow \rho_k \rightarrow \tau$ and $\Gamma_2 \vdash N_i : \rho_i$, all i .

Therefore:

$\Gamma_1 \& \Gamma_2 \vdash P[x := Q] : \rho_1 \rightarrow \dots \rightarrow \rho_k \rightarrow \tau$ and $\Gamma_1 \& \Gamma_2 \vdash Q : \rho$.

By the “beta-expansion” lemma:

$\Gamma_1 \& \Gamma_2 \vdash (\lambda x.P)Q : \rho_1 \rightarrow \dots \rightarrow \rho_k \rightarrow \tau$,

whence $\Gamma_1 \& \Gamma_2 \vdash M : \tau$.

287

288

Case 2: $M = xN_1 \dots N_k$

Since $\delta(N_i) \leq \delta(M)$ and $|N_i| < |M|$,
we have $\Gamma_i \vdash N_i : \tau_i$, by induction.

Then $\Gamma_1 \& \dots \& \Gamma_k \& \{x : \tau_1 \rightarrow \dots \rightarrow \tau_k \rightarrow p\} \vdash M : p$.

Twierdzenie: Term zamknięty M jest:

1. SN wtedy i tylko wtedy, gdy ma typ w systemie bez ω ;
2. normalizowalny wtedy i tylko wtedy, ma w systemie BCD typ nie zawierający ω ;
3. rozwiązalny wtedy i tylko wtedy, gdy ma w systemie BCD typ różny od ω .

289

Typability problem:

Given M , are there Γ, τ such that $\Gamma \vdash M : \tau$?

Moral: Typability in intersection types is undecidable.

Type-checking problem:

Given Γ, M, τ , does $\Gamma \vdash M : \tau$ hold?

Corollary:

Type-checking in intersection types is undecidable.

Proof: Reduction typability \leq type-checking.

Let $\text{FV}(M) = \vec{x}$. To find out if M typable, ask if

$$x : p \vdash Kx(\lambda \vec{x}. M) : p.$$

290

A dual decision problem

Inhabitation problem:

Given Γ, τ , is there M such that $\Gamma \vdash M : \tau$?

Fact: Inhabitation in intersection types is undecidable.
(Proof postponed)

291

Typy proste

Simple types (Curry style)

Types:

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

Konwencja:

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

Każdy typ ma postać $\tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \text{atom}$.

293

Simple type assignment

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

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

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

Note: This is a syntax-oriented system.

294

Generation (Inversion) Lemma

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

295

Subject Reduction

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

Proof: Easy induction or appeal to intersection types.

296

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 $M \in T_{\sigma \rightarrow \tau}$ and $N \in T_\sigma$ then $(MN) \in T_\tau$;
- ▶ If $M \in T_\tau$ and $x \in V_\sigma$ then $(\lambda x M) \in T_{\sigma \rightarrow \tau}$.

Write M^σ for $M \in T_\sigma$ and define beta-reduction by

$$(\lambda x^\sigma. M^\sigma) N^\sigma \Rightarrow M[x^\sigma := N] \in T_\tau.$$

297

298

Non-orthodox Church

Type-assignment with type annotations on bound variables.

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

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

Fact: If $\Gamma \vdash M : \tau$ and $\Gamma \vdash M : \sigma$ then $\tau = \sigma$.

299

300

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 $\Gamma \vdash M : \tau$ then $\Gamma \vdash |M| : \tau$.
2. If $\Gamma \vdash M : \tau$ then there exists M such that $|M| = M$ and $\Gamma \vdash M : \tau$.
3. If $M \rightarrow_\beta N$ then $|M| \rightarrow_\beta |N|$.
4. If $|M| \rightarrow_\beta N$ then there exists N such that $|N| = N$ and $M \rightarrow_\beta N$.

Proof: easy induction.

Wniosek: If $M = |M|$ then $M \in SN$ iff $M \in SN$.

301

302

Informal type annotations

Well-typed Curry style terms can be informally annotated by types, e.g. $((\lambda x^\sigma. N^\tau)^{\sigma \rightarrow \tau} P^\sigma)^\tau$. 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.

Hindley-Milner

(Robinson)

303

304

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:

$$S(f t_1 \dots t_n) = f S(t_1) \dots S(t_n).$$

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*.

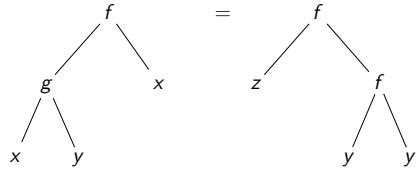
A substitution S is a *solution* of an equation “ $t = u$ ” iff $S(t) = S(u)$ and $S(u)$ is the same term. It is a solution of a system E of equations iff it is a solution of all equations in E .

303

304

Example 1

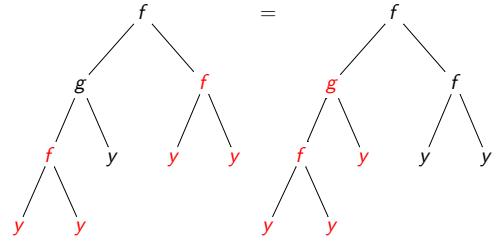
The equation $f(gxy)x = fz(fyy)$ has a solution S , such that $S(x) = fyy$, $S(y) = y$, $S(z) = g(fyy)y$, and $S(v) = v$ otherwise.



305

Example 1

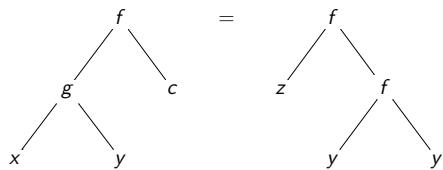
The equation $f(gxy)x = fz(fyy)$ has a solution S , such that $S(x) = fyy$, $S(y) = y$, $S(z) = g(fyy)y$, and $S(v) = v$ otherwise.



306

Example 2

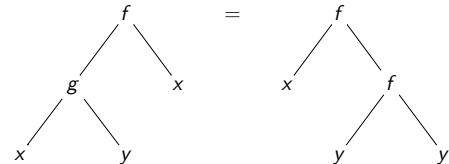
The equation $f(gxy)c = fz(fyy)$, where c is a constant, has no solution.



307

Example 2

The equation $f(gxy)c = fz(fyy)$ has no solution.



308

Principal solutions (rozwiązań 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.

309

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.

310

Type reconstruction

Fix a signature of one binary function symbol \rightarrow . 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 .

311

Type reconstruction

► If M is a variable x , then $E_M = \emptyset$ 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;
- $E_M = E_P \cup E_Q \cup \{\tau_P = \tau_Q \rightarrow p\}$,

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 $\tau_M = p \rightarrow \tau_P[p_x := p]$, where p is fresh.

312

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 $x \in \text{FV}(M)$.
2. Let S be a solution of E_M , and let Γ be such that $\Gamma(x) = S(p_x)$, for all $x \in \text{FV}(M)$. Then $\Gamma \vdash M : S(\tau_M)$.

Proof: Induction with respect to the length of M .

313

314

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.

315

316

Examples

- Principal type of S is $(p \rightarrow q \rightarrow r) \rightarrow (p \rightarrow q) \rightarrow p \rightarrow r$.
Another, non-principal, type of S :
 $(p \rightarrow q \rightarrow p) \rightarrow (p \rightarrow q) \rightarrow p \rightarrow p$
- Type $\omega = (p \rightarrow p) \rightarrow p \rightarrow p$ is the principal type of Church numerals n , for $n \geq 2$. For $\mathbf{0}$ and $\mathbf{1}$ the principal types are respectively $p \rightarrow q \rightarrow q$ and $(p \rightarrow q) \rightarrow p \rightarrow q$. Every Church numeral can also be assigned the type $\omega_{p \rightarrow q} = ((p \rightarrow q) \rightarrow p \rightarrow q) \rightarrow (p \rightarrow q) \rightarrow p \rightarrow q$.

Definable functions

Liczebniki Churcha $n = \lambda f x. f^n(x)$ mają każdy typ postaci $\omega_\sigma = (\sigma \rightarrow \sigma) \rightarrow (\sigma \rightarrow \sigma)$.

A function $f : \mathbb{N}^k \rightarrow \mathbb{N}$ is β -definable in type ω_σ if there is a closed term F such that

- $\vdash F : \omega_\sigma \rightarrow \dots \rightarrow \omega_\sigma \rightarrow \omega_\sigma$;
- If $f(n_1, \dots, n_k) = m$ then $F n_1 \dots n_k =_\beta m$.

Analogicznie można mówić o $\beta\eta$ -definiowalności.

317

318

Examples

- Addition: $\lambda n^{\omega_\sigma} \lambda m^{\omega_\sigma} \lambda f^{\sigma \rightarrow \sigma} \lambda x^\sigma. nf(mfx)$;
- Multiplication: $\lambda n^{\omega_\sigma} \lambda m^{\omega_\sigma} \lambda k^{\omega_\sigma} \lambda f^{\sigma \rightarrow \sigma} \lambda x^\sigma. n(mf)x$;
- Test for zero (if $n = 0$ then m else k):
 $\lambda n^{\omega_\sigma} \lambda m^{\omega_\sigma} \lambda k^{\omega_\sigma} \lambda f^{\sigma \rightarrow \sigma} \lambda x^\sigma. n(\lambda y^\sigma. kfx)(mfx)$.
- 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

$$f(x, y) = \text{if } x = 0 \text{ then if } y = 0 \text{ then } p_1(x, y) \text{ else } p_2(x, y) \text{ else if } y = 0 \text{ then } p_3(x, y) \text{ else } p_4(x, y).$$

Przykład:

$$f(x, y) = \text{if } x = 0 \text{ then if } y = 0 \text{ then } 3 \text{ else } y^2 + 1 \text{ else if } y = 0 \text{ then } x^3 \text{ else } x^2 + y.$$

Potęgowanie, poprzednik, równość itp.
nie są wielomianami warunkowymi.

319

320

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 :

$$\text{ifeven}(p, q, r) = \begin{cases} q, & \text{if } p \text{ is even;} \\ r, & \text{otherwise.} \end{cases}$$

More definable functions

A function f is *skewly* (skośnie) definable if there is a closed term F such that

- $\vdash F : \omega_{\sigma_1} \rightarrow \dots \rightarrow \omega_{\sigma_k} \rightarrow \omega_\sigma$;
- If $f(n_1, \dots, n_k) = m$ then $Fn_1 \dots n_k =_\beta m$.

Examples:

- The predecessor function $p(n) = n - 1$ and the exponentiation function $\exp(m, n) = m^n$ are skewly definable. (Easy)
- The subtraction $\text{minus}(m, n) = m - n$ and equality test $\text{Eq}(m, n) = \text{if } m = n \text{ then } 0 \text{ else } 1$ are not definable skewly. (Hard)

321

322

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

$$2^{\dots^{2^n}} \Big\}^k$$

Exercise: How long is the normal form of $2 \dots 2xy$?

Powtórzenie z teorii złożoności:
alternacja

323

324

Maszyna alternująca

Maszyna ma stany (konfiguracje, pozycje):
egzystencjalne i *uniwersalne*.

Konfiguracja akceptująca (końcowa) jest *wygrywająca*.

Konfiguracja niekońcowa C o następcach C_1, \dots, C_n jest *wygrywająca*, gdy:

- C jest egzystencjalna i któraś z C_1, \dots, C_n jest wygrywająca;
- C jest uniwersalna i wszystkie C_1, \dots, C_n są wygrywające.

Maszyna akceptuje, gdy konfiguracja początkowa jest wygrywająca.

Obliczenie akceptujące = drzewo.

Maszyna alternująca to gra

Pozycje w grze, to konfiguracje maszyny.
Zaczynamy w konfiguracji początkowej.

W pozycji niekońcowej C o następcach C_1, \dots, C_n , następną pozycję spośród C_1, \dots, C_n wybiera:

- gracz egzystencjalny \exists , jeśli C egzystencjalna;
- gracz uniwersalny \forall , jeśli C uniwersalna.

W pozycji końcowej gracz \exists wygrywa.

Obliczenie akceptujące = strategia gracza egzystencjalnego.
(Konfiguracje wygrywające, to te, w których \exists ma strategię.)

325

326

Złożoność alternująca

$\text{ ALOGSPACE} = \text{PSPACE}$

$\text{ APTIME} = \text{PSPACE}$

$\text{APSPACE} = \text{EXPTIME}$

$\text{AEXPSPACE} = \text{EXPSPACE}$

i tak dalej.

Powtórzenie z logiki:
minimalny rachunek zdań

(implikacyjny)

327

328

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?

$$\Gamma, \sigma \vdash \sigma \quad (\text{Ax})$$

$$\frac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \rightarrow \tau} \quad (\text{I } \rightarrow)$$

$$\frac{\Gamma \vdash \sigma \rightarrow \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau} \quad (\text{E } \rightarrow)$$

329

330

Naturalna dedukcja: notacja dla dowodów

Curry-Howard isomorphism

$$\begin{array}{c} \Gamma, x:\sigma \vdash x:\sigma \\ \frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash \lambda x M : \sigma \rightarrow \tau} \\ \frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \end{array}$$

Types = Formulas

Terms = Proofs

Computation = Proof normalization

331

332

Naturalna dedukcja: notacja dla dowodów

Hindley-Milner

$$\Gamma, x:\sigma \vdash x:\sigma$$

(Robinson)

$$\frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash \lambda x M : \sigma \rightarrow \tau}$$

$$\frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

333

334

Curry-Howard isomorphism

Normalizacja

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). \square

(Twierdzenie nadal zachodzi, jeśli zamiast „minimalnej implikacyjnej” napiszemy „intuicjonistycznej”.)

$$\frac{\Gamma, \sigma \vdash \sigma \quad \dots \quad \Gamma, \sigma \vdash \tau \quad \dots \quad \frac{\Gamma \vdash \sigma \rightarrow \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau}}{\Gamma \vdash \tau} \Rightarrow \frac{\Gamma \vdash \sigma \quad \dots \quad \Gamma \vdash \tau \quad \dots \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau}$$

$$(\lambda x:\sigma M^\tau)N^\sigma \Rightarrow M^\tau[x := N^\sigma]$$

335

336

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 terminu normalnego o typie atomowym.

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.

337

338

Long normal forms (Church style)

- If $x : \alpha_1 \rightarrow \dots \rightarrow \alpha_n \rightarrow p$ and $M_1 : \alpha_1, \dots, M_k : \alpha_k$ are long normal forms then $xM_1 \dots M_k : p$ 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 $\alpha \rightarrow \beta$.

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 $\Gamma(x) = p$ and $\Gamma \cup \{y : p\} \vdash M : \tau$ then $\Gamma \vdash M[y := x] : \tau$.

339

340

Searching for long normal forms

The Wajsberg/Ben-Yelles algorithm:

To answer $\Gamma \vdash ? : \alpha$, apply one of the following tactics:

- For $\alpha = \beta \rightarrow \gamma$, ask $\Gamma \cup \{x : \beta\} \vdash ? : \gamma$ (fresh x).
(Solution $M = \lambda x : \beta. N^\gamma$.)
- For $\alpha = p$, choose $x : \beta_1 \rightarrow \dots \rightarrow \beta_k \rightarrow p$ from Γ , then ask $\Gamma \vdash ? : \beta_i$, for all i . Accept if $k = 0$.
(Solution $M = xN_1^{\beta_1} \dots N_k^{\beta_k}$.)

Ben-Yelles algorithm

To answer $\Gamma \vdash ? : \alpha$, apply one of the following tactics:

- For $\alpha = \beta \rightarrow \gamma$, ask $\Gamma \cup \{x : \beta\} \vdash ? : \gamma$ (fresh x).
(Solution $M = \lambda x : \beta. N^\gamma$.)
- For $\alpha = p$, choose $x : \beta_1 \rightarrow \dots \rightarrow \beta_k \rightarrow p$ from Γ , then ask $\Gamma \vdash ? : \beta_i$, for all i . Success if $k = 0$.
(Solution $M = xN_1^{\beta_1} \dots N_k^{\beta_k}$.)

Ten alternujący proces można objaśniać jako grę między graczem \exists (który dowodzi tezy) i graczem \forall (który poszukuje błędu w dowodzie). Rozwiążanie istnieje wtedy i tylko wtedy, gdy gracz \exists ma strategię zwycięską.

341

342

Przykład (nieformalny): $(p \rightarrow \neg q) \rightarrow (\neg p \rightarrow \neg q) \rightarrow \neg q$

(Negacja $\neg\alpha$ oznacza $\alpha \rightarrow \perp$)

(Assumptions: $x : p \rightarrow \neg q, y : \neg p \rightarrow \neg q, z : q$. Goal: \perp)

Opponent (V): Can you prove \perp ?

Player (E): I will use assumption y !

Opponent (V): Can you prove the 1st assumption $\neg p$?
That is, can you prove \perp using new assumption $v : p$?

Player (E): Yes, I will use assumption x !

Opponent (V): Can you prove the 2nd assumption q ?

Player (E): Sure, take z !

Proof = strategy = inhabitant: $\lambda xyz. y(\lambda v. xvz)z$

Termination

- The question $\Gamma \vdash ? : \beta \rightarrow \gamma$ 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

343

344

Konstrukcja dowodu jako obliczenie

Ośad $\Gamma \vdash q$ (gdzie q – atom), można interpretować jako konfigurację automatu:

- ▶ Cel atomowy q to stan maszyny;
- ▶ Otoczenie Γ to zawartość pamięci, w tym program.
- ▶ Założenie $p \rightarrow q$ umożliwia zmianę stanu z q na p .
- ▶ Założenie $(r \rightarrow p) \rightarrow q$ umożliwia zmianę stanu z q na p , z jednokrotnym zapisaniem r w pamięci.
- ▶ Założenie $p \rightarrow r \rightarrow q$ to krok uniwersalny do p i r na raz.
- ▶ Założenie $p \rightarrow r \rightarrow q$ 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.

345

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) $q : \text{check } S_1; \text{set } S_2; \text{jmp } p$,
 - (2) $q : \text{jmp } p_1 \text{ and } p_2$,
 where $p, p_1, p_2 \in Q$ and $S_1, S_2 \subseteq R$.

346

Monotonic automata

Configuration: $\langle q, S \rangle$, where $q \in Q$ and $S \subseteq R$.
Final configurations are winning.

Transitions:

- ▶ for $I = q : \text{check } S_1; \text{set } S_2; \text{jmp } p$:
 $\langle q, S \rangle \rightarrow_I \langle p, S \cup S_2 \rangle$, provided $S_1 \subseteq S$;
 If $\langle p, S \cup S_2 \rangle$ winning then $\langle q, S \rangle$ is winning
- ▶ for $I = q : \text{jmp } p_1 \text{ and } p_2$:
 $\langle q, S \rangle \rightarrow_I \langle p_1, S \rangle \text{ and } \langle q, S \rangle \rightarrow_I \langle p_2, S \rangle$
 If $\langle p_1, S \rangle$ and $\langle p_2, S \rangle$ are winning then so is $\langle q, S \rangle$.

347

Monotonic automata

Lemma

The halting problem for monotonic automata
Is a given configuration winning?
is PSPACE-hard.

348

Monotonic encoding of an ATM

Alternating time $p(n)$.

Registers: $stan_s^t, lit_{ia}^t, poz_i^t$,

Initial configuration:

$$C_0 = \langle loop_1^0, \{lit_{1a_1}^0, \dots, lit_{na_n}^0, lit_{n+1B}^0, \dots, lit_{p(n)B}^0, stan_0^0, poz_1^0\} \rangle.$$

represents initial ID of TM for input $w = a_1 \dots a_n$.

Later:

$C = \langle loop_1^t, S \rangle$ encodes the first t steps of TM computation,
e.g., $lit_{6a}^5 \in S$ iff after 5 steps, the 6th cell contains a .

Zakończenie: $loop_1^t : \text{check } stan_s^t; \text{set } \emptyset; \text{jmp } halt$.

349

Instructions for $(s, a) \Rightarrow (u, b, +\varepsilon)$

Pierwszy pomysł: w stanie $loop_1^t$ powinniśmy wykonać:

check $stan_s^t, poz_i^t, lit_{ia}^t; \text{set } stan_u^{t+1}, poz_{i+\varepsilon}^{t+1}, lit_{ib}^{t+1}; \text{jmp } loop_1^{t+1}$;

Błąd: rejestr lit_{jx}^{t+1} są puste, a mogą być potrzebne.

Trzeba je też ustawić: w stanie $loop_i^t$ ustalamy zawartość pozycji i w czasie $t+1$.

350

Instructions for $(s, a) \Rightarrow (u, b, +\varepsilon)$

When $i < p(n)$:

$loop_i^t :$
check $stan_s^t, poz_i^t, lit_{ia}^t; \text{set } stan_u^{t+1}, poz_{i+\varepsilon}^{t+1}, lit_{ib}^{t+1}; \text{jmp } loop_{i+1}^t$;
 $check stan_s^t, poz_j^t, lit_{ia}^t; \text{set } lit_{ia}^{t+1}; \text{jmp } loop_{i+1}^t$;

When $i = p(n)$:

$loop_i^t :$
check $stan_s^t, poz_i^t, lit_{ia}^t; \text{set } stan_u^{t+1}, poz_{i+\varepsilon}^{t+1}, lit_{ib}^{t+1}; \text{jmp } loop_{i+1}^t$;
 $check stan_s^t, poz_j^t, lit_{ia}^t; \text{set } lit_{ia}^{t+1}; \text{jmp } loop_1^{t+1}$.

351

Example instructions: universal split

Assume that TM does not write nor move in universal states.

When $i < p(n)$:

$loop_i^t : \text{check } stan_s^t, lit_{ia}^t, poz_j^t; \text{set } lit_{ia}^{t+1}, poz_j^{t+1}; \text{jmp } loop_{i+1}^t$;

When $i = p(n)$:

$loop_i^t : \text{check } stan_s^t, lit_{ia}^t, poz_j^t; \text{set } lit_{ia}^{t+1}, poz_j^{t+1}; \text{jmp } split_{s1s2}^{t+1}$;
 $split_{s1s2}^{t+1} : \text{jmp } go_{s1}^{t+1} \text{ and } go_{s2}^{t+1}$;
 $go_{s1}^{t+1} : \text{check } \emptyset; \text{set } stan_{s1}^{t+1}; \text{jmp } loop_1^{t+1}$;

352

Programs into proofs!

Given $\mathcal{M} = \langle Q, R, f, I \rangle$ and $C_0 = \langle q_0, S_0 \rangle$,
define a set of axioms Γ so that

$\Gamma \vdash q_0$ iff C_0 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 $S_1 = \{s_1^1, \dots, s_1^k\}$ and $S_2 = \{s_2^1, \dots, s_2^\ell\}$:
(1) $s_1^1 \rightarrow \dots \rightarrow s_1^k \rightarrow (s_2^1 \rightarrow \dots \rightarrow s_2^\ell \rightarrow p) \rightarrow q$.

Axiom for q : jmp p_1 and p_2 :
(2) $p_1 \rightarrow p_2 \rightarrow q$.

353

354

The technical lemma we need

Proof construction as computation

To prove $\Gamma, S \vdash q$, one can:

- ▶ Note that $q = f$ and observe $f \in \Gamma$;
- ▶ Use axiom $p_1 \rightarrow p_2 \rightarrow q$ and prove in parallel:
 $\Gamma, S \vdash p_1$ and $\Gamma, S \vdash p_2$
- ▶ Use axiom $s_1^1 \rightarrow \dots \rightarrow s_1^k \rightarrow (s_2^1 \rightarrow \dots \rightarrow s_2^\ell \rightarrow p) \rightarrow q$,
provided $S_1 = \{s_1^1, \dots, s_1^k\} \subseteq S$, and prove that
 $\Gamma, S, s_2^1, \dots, s_2^\ell \vdash p$

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.

□

355

356

Hilbert style proofs

Morał

Twierdzenie:

Minimalny rachunek zdań jest PSPACE-zupełny.

A klasyczny?

Tylko co-NP zupełny.

Hilbert style propositional axioms for implication:

- ▶ $\alpha \rightarrow \beta \rightarrow \alpha$;
- ▶ $(\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \gamma$.

Hilbert style proof rule: *modus ponens* (application).
Dowód to po prostu ciąg formuł.

Deduction theorem:

If $\Gamma, \sigma \vdash \tau$ then $\Gamma \vdash \sigma \rightarrow \tau$.

357

358

Proofs as combinators

Example proof

1. $(\varphi \rightarrow (\psi \rightarrow \varphi) \rightarrow \varphi) \rightarrow (\varphi \rightarrow \psi \rightarrow \varphi) \rightarrow \varphi \rightarrow \varphi$;
2. $\varphi \rightarrow (\psi \rightarrow \varphi) \rightarrow \varphi$;
3. $(\varphi \rightarrow \psi \rightarrow \varphi) \rightarrow \varphi \rightarrow \varphi$ (detach 2 from 1);
4. $\varphi \rightarrow \psi \rightarrow \varphi$;
5. $\varphi \rightarrow \varphi$ (detach 4 from 3).

Hilbert style propositional axioms for implication:

- ▶ $K : \alpha \rightarrow \beta \rightarrow \alpha$;
- ▶ $S : (\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \gamma$.

Deduction theorem:

If $\Gamma, x:\sigma \vdash M:\tau$ then $\Gamma \vdash \lambda^*x.M : \sigma \rightarrow \tau$.

359

360

Logical connectives

$$\begin{array}{c}
 \frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi} (I\wedge) \qquad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi} (E\wedge) \quad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi} \\
 \\
 \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi} (I\vee) \quad \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi} \\
 \\
 \frac{\Gamma \vdash \varphi \vee \psi \quad \Gamma, \varphi \vdash \vartheta \quad \Gamma, \psi \vdash \vartheta}{\Gamma \vdash \vartheta} (E\vee) \\
 \\
 \frac{\Gamma \vdash \perp}{\Gamma \vdash \varphi} (E\perp)
 \end{array}$$

361

Conjunction is the product

$$\begin{array}{c}
 \frac{\Gamma \vdash M : \varphi \quad \Gamma \vdash N : \psi}{\Gamma \vdash \langle M, N \rangle : \varphi \wedge \psi} (I\wedge) \\
 \\
 \frac{\Gamma \vdash M : \varphi \wedge \psi}{\Gamma \vdash \pi_1(M) : \varphi} (E\wedge) \quad \frac{\Gamma \vdash M : \varphi \wedge \psi}{\Gamma \vdash \pi_2(M) : \psi} \\
 \\
 \pi_1(\langle M, N \rangle) \Rightarrow M, \quad \pi_2(\langle M, N \rangle) \Rightarrow N
 \end{array}$$

362

Disjunction is the coproduct (suma prosta)

$$\begin{array}{c}
 \frac{\Gamma \vdash M : \varphi}{\Gamma \vdash \text{inl}(M) : \varphi \vee \psi} (W\vee) \quad \frac{\Gamma \vdash M : \psi}{\Gamma \vdash \text{inr}(M) : \varphi \vee \psi} \\
 \\
 \frac{\Gamma \vdash M : \varphi \vee \psi \quad \Gamma(x:\varphi) \vdash P : \vartheta \quad \Gamma(y:\psi) \vdash Q : \vartheta}{\Gamma \vdash \text{case } M \text{ of } [x]P, [y]Q : \vartheta} (E\vee) \\
 \\
 \text{case inl}(P) \text{ of } [x]M, [y]N \Rightarrow M[x := P] \\
 \text{case inr}(Q) \text{ of } [x]M, [y]N \Rightarrow N[y := Q].
 \end{array}$$

363

Generalized eta-reductions

If we believe everything is canonical...

$$\langle \pi_1(M), \pi_2(M) \rangle \Rightarrow M$$

$$(\text{case } M \text{ of } [x]\text{inl } x, [y]\text{inr } y) \Rightarrow M.$$

364

Absurd is the empty type (*the thing which is not*)

Ex falso quodlibet:

$$\frac{\Gamma \vdash M : \perp}{\Gamma \vdash \varepsilon_\varphi(M) : \varphi} (E\perp)$$

No introduction rule, no beta- or eta-reduction.

Konserwatywność

Twierdzenie

Intuicjonistyczny rachunek zdań IPC (ze wszystkimi spójnikami $\vee, \wedge, \rightarrow, \perp, \neg$ 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 abstrakcją i aplikacją.

Classical (unnatural) deduction

$$\frac{\Gamma, \neg\sigma \vdash \perp}{\Gamma \vdash \sigma} (\text{Cheat})$$

Example: Peirce's Law

$$((p \rightarrow q) \rightarrow p) \rightarrow p$$

is classically valid, but unprovable without rule (Cheat).

Indeed, there is no normal M with

$$x : ((p \rightarrow q) \rightarrow p) \vdash M : p$$

365

Representing data types

- ▶ Natural numbers are generated by
 - ▶ Constant $0 : \text{int}$;
 - ▶ Successor $s : \text{int} \rightarrow \text{int}$.

They correspond to long normal forms of type
 $\omega = (0 \rightarrow 0) \rightarrow 0 \rightarrow 0$

- ▶ Words over $\{a, b\}$ are generated by
 - ▶ Constant $\varepsilon : \text{word}$;
 - ▶ Two successors $\lambda w.(a \cdot w)$ and $\lambda w.(b \cdot w)$ of type $\text{word} \rightarrow \text{word}$.

They correspond to long normal forms of type
 $\text{word} = (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow 0 \rightarrow 0$

366

367

368

- ▶ Booleans are two constants $true, false$. They correspond to terms of type $\text{bool} = 0 \rightarrow 0 \rightarrow 0$.
- ▶ Binary trees are generated by
 - Constant $\text{nil} : \text{tree}$;
 - Constructor $\text{cons} : \text{tree} \rightarrow \text{tree} \rightarrow \text{tree}$.
 They correspond to long normal forms of type

$$\text{tree} = (0 \rightarrow 0 \rightarrow 0) \rightarrow 0 \rightarrow 0$$

Generalization:

Term algebras correspond to types of order two, i.e., of the form

$$(0^{n_1} \rightarrow 0) \rightarrow \dots \rightarrow (0^{n_k} \rightarrow 0) \rightarrow 0$$

369

Types of order three and up define more complex structures.

For instance, long normal forms of type

$$((0 \rightarrow 0) \rightarrow 0) \rightarrow 0$$

are as follows:

$$\lambda F. F(\lambda x_1. F(\lambda x_2. F(\lambda x_3. F(\dots \lambda x_k. x_\ell) \dots))).$$

370

Type reducibility

Definition: Type τ is *reducible* to type σ iff there exists a closed term $\Phi : \tau \rightarrow \sigma$ such that the operator $\lambda M : \tau. \Phi M$ is injective on closed terms, i.e.,

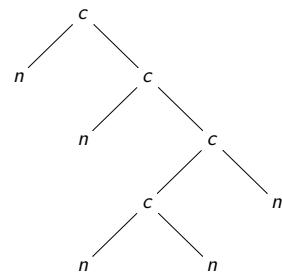
$$\Phi M_1 =_{\beta\eta} \Phi M_2 \text{ implies } M_1 =_{\beta\eta} M_2$$

for closed $M_1, M_2 : \tau$.

Examples:

- ▶ Type ω is reducible to type word using $\Phi = \lambda nfgx.nfx$.
- ▶ Types $\alpha \rightarrow \beta \rightarrow 0$ and $\beta \rightarrow \alpha \rightarrow 0$ are reducible to each other using $\Phi = \lambda xyz. xzy$.
- ▶ Type word is reducible to type tree using
 $\Phi = \lambda wcn.w(\lambda x. cnx)(\lambda x. cxn)(cnn)$.

371



372

Ważny przykład: redukcja $\text{bool} \rightarrow \text{word}$ do tree .

A nice theorem

Typ $\text{bool} \rightarrow \text{word}$, czyli:

$$(0 \rightarrow 0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow 0 \rightarrow 0$$

odpowiada algebrze termów nad sygnaturą

$$t : 0 \rightarrow 0 \rightarrow 0, \quad f, g : 0 \rightarrow 0, \quad c : 0.$$

Trzeba je zakodować jako zwykłe drzewa nad sygnaturą

$$t : 0 \rightarrow 0 \rightarrow 0, \quad c : 0.$$

Można to zrobić używając np. $\Theta : (\text{bool} \rightarrow \text{word}) \rightarrow \text{tree}$:

$$\Theta = \lambda Z. \lambda t^{0 \rightarrow 0 \rightarrow 0} c^0. Zt(\lambda x. tcx)(\lambda x. txc)(tcc)$$

373

Theorem (R. Statman):

Every type over a single type constant 0 is reducible to tree .

Idea dowodu wg Thierry'ego Joly w kilku krokach.

Krok pierwszy

Lemat: Każdy typ nad 0 można zredukować do typu

$$\mathcal{J} = (0 \rightarrow 0 \rightarrow 0) \rightarrow ((0 \rightarrow 0) \rightarrow 0) \rightarrow 0.$$

Pomysł na dowód:

Kombinatory typu \mathcal{J} są postaci $\lambda @ \lambda \Delta. W$, gdzie:

$$@ : 0 \rightarrow 0 \rightarrow 0, \quad \Delta : (0 \rightarrow 0) \rightarrow 0 \vdash W : 0.$$

Operatory $@$ i Δ mogą naśladować aplikację i abstrakcję.

374

Redukcja do typu \mathcal{J}

Mając „stały” $@ : 0 \rightarrow 0 \rightarrow 0$ i $\Delta : (0 \rightarrow 0) \rightarrow 0$, 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.

375

376

Krok drugi

Zredukowaliśmy dowolny typ do typu:

$$\mathcal{J} = (0 \rightarrow 0 \rightarrow 0) \rightarrow ((0 \rightarrow 0) \rightarrow 0) \rightarrow 0$$

czyli do $\mathcal{J} = \text{bool} \rightarrow ((0 \rightarrow 0) \rightarrow 0) \rightarrow 0$.

Teraz zredukujemy końcówkę $((0 \rightarrow 0) \rightarrow 0) \rightarrow 0$
do typu $\text{word} = (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow 0 \rightarrow 0$.
(Potem zredukujemy \mathcal{J} do typu $\text{bool} \rightarrow \text{word}$.)

$$\Phi : [((0 \rightarrow 0) \rightarrow 0) \rightarrow 0] \rightarrow [(0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow 0 \rightarrow 0]$$

$$\Phi = \lambda Z \lambda f^{0 \rightarrow 0} g^{0 \rightarrow 0} x^0. Z(\lambda h^{0 \rightarrow 0}. f(h(g(hx))))$$

Kombinatory typu $[(0 \rightarrow 0) \rightarrow 0] \rightarrow 0$ są takie:

$$M = \lambda \varphi^{(0 \rightarrow 0) \rightarrow 0}. \varphi(\lambda y_1. \varphi(\lambda y_2. \dots \varphi(\lambda y_k. y_j))).$$

Term M jest zdeterminowany przez liczby k i j .

Przykład: $M = \lambda \varphi^{(0 \rightarrow 0) \rightarrow 0}. \varphi(\lambda y_1. \varphi(\lambda y_2. \varphi(\lambda y_3. \varphi(\lambda y_4. y_2))))$.
Wtedy $\Phi(M) = \lambda f g x. f^4(g(f^2(x)))$.

Opuszczamy szczegóły.

377

378

Krok trzeci

Mamy $\mathcal{J} = \text{bool} \rightarrow \mathcal{J}_2 \rightarrow 0$, gdzie:

$$\text{bool} = 0 \rightarrow 0 \rightarrow 0 \quad \mathcal{J}_2 = (0 \rightarrow 0) \rightarrow 0.$$

W kroku drugim zredukowaliśmy typ $\mathcal{J}_2 \rightarrow 0$ do typu

$$\text{word} = (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow 0 \rightarrow 0$$

używając $\Phi = \lambda Z^{\mathcal{J}_2 \rightarrow 0} \lambda f^{0 \rightarrow 0} g^{0 \rightarrow 0} x^0. Z(\lambda h^{0 \rightarrow 0}. f(h(g(hx))))$

Krok czwarty: redukcja $\text{bool} \rightarrow \text{word}$ do tree .

To już było.

Teraz redukujemy $\text{bool} \rightarrow \mathcal{J}_2 \rightarrow 0$ do $\text{bool} \rightarrow \text{word}$ używając

$$\Psi = \lambda Z^{\text{bool} \rightarrow \mathcal{J}_2 \rightarrow 0} \lambda t^{\text{bool}} \lambda f^{0 \rightarrow 0} g^{0 \rightarrow 0} x^0. Z t (\lambda h^{0 \rightarrow 0}. f(h(g(hx))))$$

czyli $\Psi = \lambda Z^{\text{bool} \rightarrow \mathcal{J}_2 \rightarrow 0} \lambda t^{\text{bool}}. \Phi(Zt)$.

Uwaga: to nie jest oczywiste!

379

380

Semantics for finite types

Typy proste: semantyka

Assumptions:

- ▶ Orthodox Church style;
- ▶ Only one atomic type 0 ;
- ▶ Extensional equality $=_{\beta\eta}$.

381

382

Standard model $\mathfrak{M}(A)$

Completeness

- ▶ Basic domain $D_0 = A$;
- ▶ Function domains: $D_{\sigma \rightarrow \tau} = D_\sigma \rightarrow D_\tau$;
- ▶ Obvious semantics:
 - ▶ $\llbracket x \rrbracket_v = v(x)$;
 - ▶ $\llbracket MN \rrbracket_v = \llbracket M \rrbracket_v(\llbracket N \rrbracket_v)$;
 - ▶ $\llbracket \lambda x^\tau. M \rrbracket_v = \lambda d \in D_\tau. \llbracket M \rrbracket_{v[x \mapsto d]}$.

Theorem (Harvey Friedman):

Terms are $\beta\eta$ -equal iff they are equal in $\mathfrak{M}(\mathbb{N})$.

Szkic dowodu: Definiujemy częściowe surjekcje

$$\varphi_\sigma : D_\sigma \multimap T_\sigma / =_{\beta\eta}$$

$\varphi_0 : \mathbb{N} \rightarrow T_0 / =_{\beta\eta}$ – jakkolwiek.

$$\varphi_{\sigma \rightarrow \tau}(d) = [M]_{\beta\eta}, \text{ gdy dla każdego } N : \sigma$$

jeśli $\varphi_\sigma(e) = [N]_{\beta\eta}$, to $\varphi_\tau(d \cdot e) = [MN]_{\beta\eta}$

383

384

Completeness proof

Dla $\varphi_{\sigma \rightarrow \tau}(f) = [M]_{\beta\eta}$ ten diagram ma być przemienny:

$$\begin{array}{ccc} D_\tau & \xrightarrow{f} & D_\sigma \\ \varphi_\tau \downarrow & & \downarrow \varphi_\sigma \\ T_\tau /_{=_{\beta\eta}} & \xrightarrow{\lambda t. Mt} & T_\sigma /_{=_{\beta\eta}} \end{array}$$

For any M , there exists such an f (not unique).
For a given f , such an M (if exists) is unique up to $\beta\eta$.

385

Completeness proof

Nadużycie notacji:

Dla $e \in D_\sigma$ piszemy $\bar{e} = M$, gdy $\varphi_\sigma(e) = [M]_{\beta\eta}$.

Wtedy $\bar{e} \bar{a} =_{\beta\eta} \bar{e} \cdot \bar{a}$.

Lemma:

Take v so that $\overline{v(x)} = x$, for all x . Then $M =_{\beta\eta} \overline{[M]_v}$, all M .

Main Proof: Let $\mathfrak{M}(\mathbb{N}) \models M = N$. Then $\overline{[M]_v} = \overline{[N]_v}$, for all v , in particular for v as above. Therefore

$$M =_{\beta\eta} \overline{[M]_v} =_{\beta\eta} \overline{[N]_v} =_{\beta\eta} N.$$

386

Finite completeness

„Łatwa” obserwacja:

For every M and N there is k such that:

$$M =_{\beta\eta} N \quad \text{iff} \quad \mathfrak{M}(k) \models M = N.$$

Corollary:

Terms are $\beta\eta$ -equal iff they are equal in all finite models.

(Konwencja: $k = \{0, 1, \dots, k-1\}$)

Finite completeness

Theorem (R. Statman):

For every M there is k such that, for all N :

$$M =_{\beta\eta} N \quad \text{iff} \quad \mathfrak{M}(k) \models M = N.$$

Corollary:

Terms are $\beta\eta$ -equal iff they are equal in all finite models.

(Konwencja: $k = \{0, 1, \dots, k-1\}$)

387

388

Finite completeness proof

It suffices to prove that

for every closed $M : \text{tree}$ there is k such that, for all $N : \text{tree}$:

$$M =_{\beta\eta} N \quad \text{iff} \quad \mathfrak{M}(k) \models M = N.$$

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 $p \in D_{0 \rightarrow 0 \rightarrow 0}$ 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 $\overline{[M](p)(0)}$ można uważać za numer tego drzewa.

389

390

Ćwiczenie

$$p(m)(n) = 2^m(2n+1)$$

Jaka liczba jest numerem drzewa $\lambda px. px(p(pxx)x)$?

For $M : \text{tree}$, define $k = 2 + \overline{[M](p)(0)}$, i.e. $2 + \text{numer}(M)$.

Let $p' : k \rightarrow k \rightarrow k$ be p „truncated” to values less than k .
Then $p' \in D_{0 \rightarrow 0 \rightarrow 0}$ in $\mathfrak{M}(k)$.

Suppose $\mathfrak{M}(k) \models M = N$. Then in the model $\mathfrak{M}(k)$:

$$\overline{[M](p')(0)} = \overline{[N](p')(0)} \tag{*}$$

All numbers needed to verify (*) are at most $k-2$. Hence

$$\overline{[M](p')(0)} = \overline{[N](p')(0)} = k-2$$

and $\overline{[M](p)(0)} = \overline{[N](p)(0)}$ also holds in $\mathfrak{M}(\mathbb{N})$.
It follows that $M =_{\beta\eta} N$.

391

392

Equality is not definable in simple types

There is no $E : \omega_\tau \rightarrow \omega_\sigma \rightarrow \omega_\rho$, such that for all $p, q \in \mathbb{N}$:

$$E p^{\omega_\tau} q^{\omega_\sigma} =_{\beta\eta} 0^{\omega_\rho} \quad \text{iff} \quad p = q.$$

Proof: By Statman's thm., take k such that for all $N : \omega_p$:

$$\mathfrak{M}(k) \models 0^{\omega_p} = N \quad \text{iff} \quad 0^{\omega_p} =_{\beta\eta} N.$$

There are $p \neq q$ with $\llbracket p^{\omega_\tau} \rrbracket = \llbracket q^{\omega_\tau} \rrbracket$ in $\mathfrak{M}(k)$. So in $\mathfrak{M}(k)$:

$$\begin{aligned} \llbracket E p^{\omega_\tau} q^{\omega_\sigma} \rrbracket &= \llbracket E \rrbracket \llbracket p^{\omega_\tau} \rrbracket \llbracket q^{\omega_\sigma} \rrbracket = \llbracket E \rrbracket \llbracket q^{\omega_\tau} \rrbracket \llbracket q^{\omega_\sigma} \rrbracket \\ &\quad \llbracket E q^{\omega_\tau} q^{\omega_\sigma} \rrbracket = \llbracket 0^{\omega_\rho} \rrbracket \end{aligned}$$

Thus $\mathfrak{M}(k) \models E p^{\omega_\tau} q^{\omega_\sigma} = 0^{\omega_\rho}$, whence $p = q$.

393

394

Dygresja: rachunek λY

Do rachunku $\lambda \rightarrow$ dodajemy stałe $Y_\sigma : (\sigma \rightarrow \sigma) \rightarrow \sigma$ z regułą redukcji $Y_\sigma \Rightarrow \lambda f^{\sigma \rightarrow \sigma}. f(Y_\sigma f)$.

Czy teraz można zdefiniować więcej funkcji z $\mathbb{N} \rightarrow \mathbb{N}$?

Odpowiedź (Plotkin'82): Nie można.

Przyczyna: Poprawność semantyki skończonej.

System PCF: Rachunek λY plus typ **int** z operacjami poprzednika i następnika i stałą zero.

System T: Jak PCF, ale zamiast Y_σ są *rekursory* (uogólnienie rekursji prostej na wyższe typy.)

Twierdzenie Parysa

Definicja: Funkcje $f, g : A \rightarrow \mathbb{N}$ 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 $\tau \rightarrow \omega$ tworzą tylko skończonie wiele klas dominacyjnej równoważności.

Uwaga: To działa dla λY i nie zależy od definicji ω .

Wniosek: Dla każdego typu τ istnieje takie n , że dla $k \geq n$ zbiór \mathbb{N}^k nie można reprezentować w typie τ .

Przyczyna: Dla dużych k rzutowania Π_i^k są dominacyjnie parami nierównoważne.

393

394

Plotkin's problem

Given $d \in D_\tau$ in a finite model $\mathfrak{M}(X)$.
Is there a closed term $M : \tau$ with $\llbracket M \rrbracket = d$?

More generally:

Let $v(x_1) = e_1 \in D_{\sigma_1}, \dots, v(x_n) = e_n \in D_{\sigma_n}$.
Is there M such that $\llbracket M \rrbracket_v = d$?
(Is d definable from e_1, \dots, e_n ?)

Fact: These decision problems are reducible to each other.

395

396

Undecidability 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.

Szczegóły opuszczamy z braku czasu.

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 $d \in D_{\alpha \rightarrow \beta}$ then $\tau_d = \bigcap_{e \in D_\alpha} (\tau_e \rightarrow \tau_{de})$.

For a valuation v in $\mathfrak{M}(X)$, take $\Gamma_v(x) = \tau_{v(x)}$.

Main Lemma: $\llbracket M \rrbracket_v = d$ iff $\Gamma_v \vdash M : \tau_d$.

397

398

Higher-order unification

Problem: Given Church-style (normal) terms M, N of the same type with some variables $x_1^{\sigma_1}, \dots, x_n^{\sigma_n}$ called *unknowns*. (There may be other variables called *constants* or *parameters*.) Are there terms $P_1^{\sigma_1}, \dots, P_n^{\sigma_n}$ such that

$$M[x_1 := P_1, \dots, x_n := P_n] =_{\beta\eta} N[x_1 := P_1, \dots, x_n := P_n]?$$

Non-essential generalization to finite systems of equations.

Rząd (order)

Rząd typu:

- $\text{order}(p) = 0$;
- $\text{order}(\sigma \rightarrow \tau) = \max\{\text{order}(\sigma) + 1, \text{order}(\tau)\}$.

Uwaga:

$$\text{order}(\sigma_1 \rightarrow \dots \rightarrow \sigma_n \rightarrow p) = 1 + \max\{\text{order}(\sigma_n) \mid n = 1, \dots, n\}$$

Unification of order n has unknowns of order $n - 1$.

399

400

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 zagnieździone.

401

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?

402

Higher-order matching

(dopasowanie wyższego 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 $\omega^k \rightarrow \omega$.

Rozwiązywanie 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

$$\omega = (p \rightarrow p) \rightarrow p \rightarrow p.$$

403

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.

404

Logika pierwszego i drugiego rzędu

Polimorfizm

Logika pierwszego rzędu: kwantyfikatory wiążą tylko zmienne indywidualne. Symbole relacyjne są stałe.

Logika drugiego rzędu: kwantyfikatory mogą wiązać zmienne relacyjne.

405

406

Logika drugiego rzędu

“Dependency erasure”

Przykład formuły: $\forall P(P(x) \rightarrow Q(x)) \rightarrow Q(x)$

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).

Example 1: $\forall P(P(x) \rightarrow Q(x)) \rightarrow Q(x)$

This is the same principle as $\forall p(p \rightarrow q) \rightarrow q$.

Example 2: Definition of natural numbers:

$\text{Nat}(n) : \forall R(\forall x(R(x) \rightarrow R(sx)) \rightarrow R(0) \rightarrow R(n))$

Jak można udowodnić formułę $\text{Nat}(5)$?

Deleting first-order layer: $\forall r((r \rightarrow r) \rightarrow r \rightarrow r)$.

407

408

(Ax) $\Gamma, \varphi \vdash \varphi$

$$(\rightarrow I) \frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \rightarrow \psi}$$

$$(\rightarrow E) \frac{\Gamma \vdash \varphi \rightarrow \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$

$$(\forall I) \frac{\Gamma \vdash \varphi \quad (p \notin FV(\Gamma))}{\Gamma \vdash \forall p \varphi}$$

$$(\forall E) \frac{\Gamma \vdash \forall p \varphi}{\Gamma \vdash \varphi[p := \vartheta]}$$

$$(\exists I) \frac{\Gamma \vdash \varphi[p := \vartheta]}{\Gamma \vdash \exists p \varphi}$$

$$(\exists E) \frac{\Gamma \vdash \exists p \varphi, \quad \Gamma, \varphi \vdash \psi \quad (p \notin FV(\Gamma, \psi))}{\Gamma \vdash \psi}$$

- **Full comprehension:** Propositional variables range over all formulas:

$$\exists p (\varphi \leftrightarrow p)$$

$$\forall p \varphi(p) \rightarrow \varphi(\psi)$$

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

Girard's System F (Church style)

- Basic idea: $\forall p. p \rightarrow p$ is a type of a generic identity.

$$\Gamma(x : \varphi) \vdash x : \varphi$$

- Church style (explicit polymorphism):

- Generic identity I admits a type argument.
- The application $I\tau$ is of type $\tau \rightarrow \tau$.

- Curry style (implicit polymorphism):

- Generic identity I has all types $\tau \rightarrow \tau$ at once.

$$\frac{\Gamma(x : \varphi) \vdash M : \psi}{\Gamma \vdash (\lambda x : \varphi M) : \varphi \rightarrow \psi}$$

$$\frac{\Gamma \vdash M : \varphi \rightarrow \psi \quad \Gamma \vdash N : \varphi}{\Gamma \vdash (MN) : \psi}$$

$$\frac{\Gamma \vdash M : \varphi \quad (p \notin FV(\Gamma))}{\Gamma \vdash (\Lambda p M) : \forall p \varphi}$$

$$\frac{\Gamma \vdash M : \forall p \varphi}{\Gamma \vdash M\vartheta : \varphi[p := \vartheta]}$$

Girard's System F (Church style)

Reduction rules

Beta:

$$\frac{}{(\lambda x : \tau. M)N \Rightarrow_{\beta} M[x := N];}$$

$$\frac{}{(\Lambda p. M)\tau \Rightarrow_{\beta} M[p := \tau];}$$

Eta:

$$\frac{}{\lambda x : \tau. Mx \Rightarrow_{\eta} M \quad (x \notin FV(M));}$$

$$\frac{}{\Lambda p. Mp \Rightarrow_{\eta} M \quad (p \notin FV(M)).}$$

Subject reduction:

If $\Gamma \vdash M : \tau$ and $M \Rightarrow_{\beta\eta} N$ then $\Gamma \vdash N : \tau$

$$\Gamma(x : \varphi) \vdash x : \varphi$$

$$\frac{\Gamma(x : \varphi) \vdash M : \psi}{\Gamma \vdash (\lambda x : \varphi M) : \varphi \rightarrow \psi}$$

$$\frac{\Gamma \vdash M : \varphi \rightarrow \psi \quad \Gamma \vdash N : \varphi}{\Gamma \vdash (MN) : \psi}$$

$$\frac{\Gamma \vdash M : \varphi \quad (p \notin FV(\Gamma))}{\Gamma \vdash (\Lambda p M) : \forall p \varphi}$$

$$\frac{\Gamma \vdash M : \forall p \varphi}{\Gamma \vdash M\vartheta : \varphi[p := \vartheta]}$$

A few examples (zamiast $\lambda x : \tau$ piszemy λx^{τ})

Wycieranie typów

- Term $\Lambda q. \lambda x^{\forall p(p \rightarrow p)}. x(q \rightarrow q)(xq)$

has type $\forall q(\forall p(p \rightarrow p) \rightarrow q \rightarrow q);$

$$|x| = x$$

- Term $2 = \Lambda p. \lambda f^{p \rightarrow p} \lambda x^p. f(fx)$

has type $\forall p((p \rightarrow p) \rightarrow (p \rightarrow p));$

$$|MN| = |M||N|$$

- Term $\lambda f^{\forall p(p \rightarrow q \rightarrow p)} \Lambda p \lambda x^p. f(q \rightarrow p)(fpx)$

has type $\forall p(p \rightarrow q \rightarrow p) \rightarrow \forall p(p \rightarrow q \rightarrow q \rightarrow p).$

$$|\lambda x : \sigma. M| = \lambda x. |M|$$

$$|\Lambda p. M| = |M|$$

$$|M\tau| = |M|$$

$$\begin{array}{c}
 \Gamma(x:\varphi) \vdash x:\varphi & \Gamma(x:\tau) \vdash x:\tau \\
 \frac{\Gamma(x:\varphi) \vdash M:\psi}{\Gamma \vdash (\lambda x:\varphi M):\varphi \rightarrow \psi} & \frac{\Gamma \vdash M:\varphi \rightarrow \psi \quad \Gamma \vdash N:\varphi}{\Gamma \vdash (MN):\psi} \\
 \\
 \frac{\Gamma \vdash M:\varphi}{\Gamma \vdash (\Lambda p M):\forall p \varphi} \quad (p \notin \text{FV}(\Gamma)) & \frac{\Gamma \vdash M:\forall p \varphi}{\Gamma \vdash M\vartheta:\varphi[p:=\vartheta]} \\
 \\
 \frac{\Gamma \vdash M:\sigma}{\Gamma \vdash \lambda x.M:\tau \rightarrow \sigma} & \frac{\Gamma \vdash M:\tau \rightarrow \sigma \quad \Gamma \vdash N:\tau}{\Gamma \vdash NM:\sigma} \\
 \\
 \frac{\Gamma \vdash M:\sigma \quad (p \notin \text{FV}(\Gamma))}{\Gamma \vdash M:\forall p \sigma} & \frac{\Gamma \vdash M:\forall p \sigma}{\Gamma \vdash M:\sigma[p:=\tau]}
 \end{array}$$

417

418

Wycieranie redukcji

Properties of type erasure

$$\begin{array}{l}
 (\lambda x:\tau.M)N \xrightarrow{\beta} M[x:=N]; \\
 (\lambda x.|M|)|N| \xrightarrow{\beta} |M|[x:=|N|] = |M[x:=N]| \\
 \\
 (\Lambda p.M)\tau \xrightarrow{\beta} M[p:=\tau], \\
 |M| = |M| \\
 \\
 \lambda x:\tau.Mx \xrightarrow{\eta} M; \\
 \lambda x.|M|x \xrightarrow{\eta} |M| \\
 \\
 \Lambda p.Mp \xrightarrow{\eta} M; \\
 |M| = |M|
 \end{array}$$

(Church is blue, Curry is red.)

1. If $\Gamma \vdash M:\tau$ then $\Gamma \vdash |M|:\tau$.
2. If $\Gamma \vdash M:\tau$ then there exists M such that $|M| = M$ oraz $\Gamma \vdash M:\tau$.
3. If $M \xrightarrow{\beta} N$ then $|M| \xrightarrow{\beta} |N|$.
4. If $|M| \xrightarrow{\beta} N$ then there exists N such that $|N| = N$ and $M \xrightarrow{\beta} N$.

Uwaga: Tym razem część 4 nie jest oczywista, a w części 3 jest tylko $\xrightarrow{\beta}$, a nie \rightarrow .

419

420

Poprawność redukcji (Curry)

Subject reduction... fails for η **Twierdzenie:**Jeśli $\Gamma \vdash M:\tau$, oraz $M \xrightarrow{\beta} M'$ to $\Gamma \vdash M':\tau$.**Dowód:** Nie jest oczywisty.**Example:**

$$\begin{aligned}
 x:p \rightarrow \forall q(q \rightarrow q) &\vdash \lambda y.xy:p \rightarrow q \rightarrow q. \\
 x:p \rightarrow \forall q(q \rightarrow q) &\not\vdash x:p \rightarrow q \rightarrow q.
 \end{aligned}$$

421

422

Silna normalizacja

Silna normalizacja dla typów prostych

Twierdzenie (Jean-Yves Girard, 1972)
System F ma własność SN.

Wniosek
 Zdaniowa logika intuicjonistyczna drugiego rzędu jest niesprzeczna.

Termy stabilne (obliczalne):

- $\llbracket p \rrbracket := \text{SN};$
- $\llbracket \tau \rightarrow \sigma \rrbracket := \{M \mid \forall N(N \in \llbracket \tau \rrbracket \Rightarrow MN \in \llbracket \sigma \rrbracket)\};$

Naiwne uogólnienie:

- $\llbracket \forall p \tau \rrbracket = \bigcap \{\llbracket \tau[p:=\sigma] \rrbracket \mid \sigma \text{ — dowolny typ}\}.$

To nie jest dobrze ufundowana definicja.

423

424

Zbiór nasycony (Candidat de reductibilité)

to taki zbiór termów (Curry'ego), który ma trzy własności:

- 1) $X \subseteq \text{SN}$.
- 2) Jeśli $N_1, \dots, N_k \in \text{SN}$, to $xN_1 \dots N_k \in X$.
- 3) Jeśli $M[x := N_0]N_1 \dots N_k \in X$, oraz $N_0 \in \text{SN}$,
to $(\lambda x.M)N_0N_1 \dots N_k \in X$.

\mathcal{G} – rodzina wszystkich kandydatów (uwaga: $\text{SN} \in \mathcal{G}$).

Wartościowanie $\xi : TV \rightarrow \mathcal{G}$ wyznacza interpretację typu:

$$\begin{aligned}\llbracket p \rrbracket_\xi &= \xi(p); \\ \llbracket \sigma \rightarrow \tau \rrbracket_\xi &= \{M \mid \forall N (N \in \llbracket \sigma \rrbracket_\xi \Rightarrow MN \in \llbracket \tau \rrbracket_\xi)\}; \\ \llbracket \forall p.\sigma \rrbracket_\xi &= \bigcap_{X \in \mathcal{G}} \llbracket \sigma \rrbracket_{\xi(p \rightarrow X)}.\end{aligned}$$

Fakt: Zawsze $\llbracket \tau \rrbracket_\xi \in \mathcal{G}$.

425

426

Silna normalizacja (Curry)

Główny lemat: Niech $\Gamma \vdash M : \tau$, $\text{FV}(M) = \{x_1, \dots, x_k\}$.

Jeśli $N_i \in \llbracket \Gamma(x_i) \rrbracket_\xi$, dla $i = 1, \dots, k$,
to $M[x_1 := N_1, \dots, x_k := N_k] \in \llbracket \tau \rrbracket_\xi$.

Dowód twierdzenia:

Jeśli $\Gamma \vdash M : \tau$, to $M \in \llbracket \tau \rrbracket_\xi$, bo $x \in \llbracket \Gamma(x) \rrbracket_\xi$.
Teza wynika więc stąd, że $\llbracket \tau \rrbracket_\xi \subseteq \text{SN}$.

Silna normalizacja (Church)

Jeśli

$$M = M_0 \rightarrow M_1 \rightarrow M_2 \rightarrow \dots$$

to

$$|M| = |M_0| \succ |M_1| \succ |M_2| \succ \dots$$

gdzie \succ oznacza \rightarrow lub $=$.

Liczba wystąpień \rightarrow jest skończona, bo mamy SN (Curry).

Liczba wystąpień $=$ też, bo redukcje typowe usuwają Λ .

427

428

Zła wiadomość

The typability problem

Dowód Girarda nie jest wyrażalny nawet w języku arytmetyki drugiego rzędu.

Examples: The following terms are strongly normalizable, but untypable in system F:

Jeszcze gorsza wiadomość:

Własność SN dla systemu F jest NIEZALEŻNA od arytmetyki drugiego rzędu.

$$\blacktriangleright (\lambda zy.y(zl)(zK))(\lambda x.xx);$$

$$\blacktriangleright 22K.$$

429

430

Nierozstrzygalność (1)

Nierozstrzygalność (2)

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.

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.

Dowód: Redukcja logiki 1. rzędu:

Formuła φ jest twierdzeniem logiki 1. rzędu
wtedy i tylko wtedy, gdy
jej translacja $\bar{\varphi}$ jest niepustym typem systemu F.

431

432

Numerals $n = \lambda p \lambda f^{p \rightarrow p} \lambda x^p. f(f(\dots f(x)))$
 are of type $\omega = \forall p((p \rightarrow p) \rightarrow (p \rightarrow p))$.

A function $f : \mathbb{N}^k \rightarrow \mathbb{N}$ is *definable in system F*
 iff there is a term $F : \omega \rightarrow \dots \rightarrow \omega \rightarrow \omega$ such that

$$F n_1 \dots n_k =_{\beta} m \quad \text{iff} \quad f(n_1, \dots, n_k) = m.$$

- $Add = \lambda mn. \lambda p. \lambda fx. mpf(npfx)$;
- $Mult = \lambda mn. \lambda p. \lambda fx. mp(npfx)x$;
- $Exp = \lambda mn. \lambda p. \lambda fx. m(p \rightarrow p)(np)fx$.

433

434

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) Siłą normalizację dla termów $F n_1 \dots n_k : \omega$ można udowodnić w arytmetyce drugiego rzędu.
- (\Leftarrow) Dowód formuły $\forall n \exists m P(n, m)$ „wyciera się” do termu typu $\omega \rightarrow \omega$.

435

434