Logika i teoria typów

Wykład 13

18 stycznia 2022

Polimorficzne liczby naturalne

Numerals:

$$\mathrm{n} = \Lambda p \lambda f^{p \to p} \lambda x^{p}.\, f(f(\cdots f(x)))$$

are of type

$$\omega = \forall p((p \to p) \to (p \to p)).$$

Dziwnym trafem, ten typ jest wytarciem formuły:

$$Nat(a) = \forall R(\forall b(R(b) \to R(sb)) \to R(0) \to R(a))$$

Natural numbers

$$Nat(a) = \forall R(\forall b(R(b) \to R(sb)) \to R(0) \to R(a))$$

Dowód formuły $Nat(0)$ można zapisać tak: $\lambda R \lambda X \lambda Y.\, Y$, gdzie $X : \forall b(R(b) \to R(sb))$, $Y : R(0)$

Dowodem $Nat(1)$, czyli $Nat(s0)$, jest $\lambda R \lambda X \lambda Y.\, X0Y$ (po wytarciu zostaje $\lambda R \lambda X \lambda Y.\, XY$).

Dowód formuły $Nat(s^n 0)$ wyciera się do liczebnika $\mathbf{n}$.

Natural numbers

Numerals:

$$\mathrm{n} = \Lambda p \lambda f^{p \to p} \lambda x^{p}.\, f(f(\cdots f(x)))$$

are of type

$$\omega = \forall p((p \to p) \to (p \to p)).$$

Examples of representable functions:

- $Add = \lambda mn.\Lambda p.\lambda fx.mpf(npfx)$;
- $Mult = \lambda mn.\Lambda p.\lambda fx.mp(npf)x$;
- $Exp = \lambda mn.\Lambda p.\lambda fx.m(p \to p)(np)fx$.

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):

($\Leftarrow$) Dowód formuły $\forall n \exists m\, P(n, m)$ wyciera się do termu typu $\omega \to \omega$.

($\Rightarrow$) Dowód silnej normalizacji dla ustalonej funkcji można przeprowadzić w arytmetyce drugiego rzędu.

Siła wyrazu polimorfizmu

Twierdzenie (Girard):

Funkcja jest definiowalna w systemie **F** wtedy i tylko wtedy, gdy jest dowodliwie rekurencyjna w arytmetyce drugiego rzędu.

Wniosek: Silna normalizacja dla systemu **F** jest niezależna od arytmetyki drugiego rzędu.

Recursive and inductive types

Recursive types (Curry style)

Principal idea: Type $\mu p.\, \tau(p)$ is a solution of $X = \tau(X)$. For example, one identifies

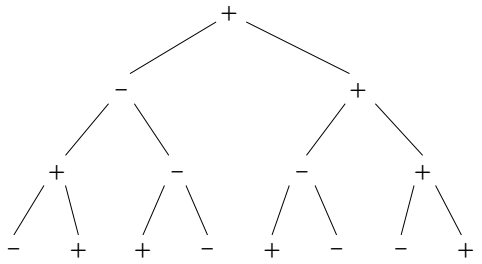
$$\alpha = \mu p.\, q \to p \quad \text{with} \quad q \to \alpha.$$

Therefore

$$y : q, \quad x : \alpha \quad \vdash xyy \ldots y : \alpha$$

Dangerous trick: Let $\iota = \mu p.\, p \to p$. Then every closed term has type ι (no more SN).

Positive and Negative



Monotoniczność: Jeśli p występuje w φ tylko pozytywnie, to

$$p \to p' \vdash \varphi(p) \to \varphi(p').$$

Tak jest na przykład gdy $\varphi(p) = (p \to q) \to p$.

Positivity

- $Pos(p) = \{p\}$;
- $Neg(p) = \varnothing$;
- $Pos(\tau \to \sigma) = Neg(\tau) \cup Pos(\sigma)$;
- $Neg(\tau \to \sigma) = Pos(\tau) \cup Neg(\sigma)$;
- $Pos(\mu p\, \tau) = Pos(\tau) - \{p\}$;
- $Neg(\mu p\, \tau) = Neg(\tau) - \{p\}$

Type $\mu p\, \tau$ is only permitted when $p \notin Neg(\tau)$.

Positive recursive types (Curry style)

- Strong normalization holds (even in PA).
  But add any negative fixpoint and SN is lost.
- Incomparable with F wrt typability:
  - $\lambda x.\, xx$ untypable.
    (Needs a negative fixpoint $\tau = \tau \to \dots$)
  - 22K typable.
    (Take $\alpha = \mu p.\, q \to p$. Then $\mathsf{K} : \alpha \to \alpha$.)

Typy rekurencyjne „w sensie Churcha”?

Oznaczenie $\mu = \mu p.\, \sigma(p)$.

Potrzebujemy wprowadzania i eliminacji.
To pierwsze to operator $in : \sigma(\mu) \to \mu$:

$$\frac{\Gamma \vdash M : \sigma(\mu)}{\Gamma \vdash in(M) : \mu}$$

Studium przypadku: liczby naturalne

Typ 1 ma jeden element $\bullet : 1$.

Definiujemy:

$$\mathsf{int} = \mu p.\, 1 \oplus p,$$

czyli

$$\mathsf{int} \approx 1 \oplus \mathsf{int},$$

czyli

$$\mathsf{int} \approx 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus \cdots$$

Zero to element $in(\mathsf{inl}(\bullet))$.

Liczba $n+1$ to element $in(\mathsf{inr}(n))$.

Przykład: liczby naturalne

Niech $\mathsf{int} = \mu p.\, 1 \oplus p$ oraz $in : 1 \oplus \mathsf{int} \to \mathsf{int}$.

Zero to element $in(\mathsf{inl}(\bullet))$.

Liczba $n+1$ to element $in(\mathsf{inr}(n))$.

Funkcja następnika: $succ = \lambda n^{\mathsf{int}}.\, in(\mathsf{inr}(n))$

Uwaga: włożenie $in : 1 \oplus \mathsf{int} \to \mathsf{int}$, to moralnie
to samo, co para $\langle 0, s \rangle : \mathsf{int} \times (\mathsf{int} \to \mathsf{int})$.

Włożenie $in : 1 \oplus \mathsf{int} \to \mathsf{int}$, to moralnie
to samo, co para $\langle 0, s \rangle : \mathsf{int} \times (\mathsf{int} \to \mathsf{int})$.

```
Inductive nat : Set :=  O : nat | S : nat -> nat.
```


$\mathsf{int} = \mu p.\, 1 \oplus p$

Jaki eliminator jest odpowiedni dla liczb naturalnych?

Już go mamy: to jest rekursor.

$$\frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash P : \tau \quad \Gamma \vdash Q : \mathsf{int} \to \tau \to \tau}{\Gamma \vdash \mathsf{R}_\tau\, M\, P\, Q : \tau}$$

$$\mathsf{R}_\tau : \mathsf{int} \to \tau \to (\mathsf{int} \to \tau \to \tau) \to \tau$$

$$\mathsf{R}_\tau 0\, P\, Q \Rightarrow P \qquad \mathsf{R}_\tau (\mathsf{s}\, n)\, P\, Q \Rightarrow Q\, n\, (\mathsf{R}_\tau n\, P\, Q)$$

(To jest wytarcie reguły dla aksjomatu indukcji)

Prostsza wersja rekursji: iteracja

$$\frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash P : \tau \quad \Gamma \vdash Q : \tau \to \tau}{\Gamma \vdash \mathsf{It}_\tau M P Q : \tau}$$

$$\mathsf{It}_\tau 0 P Q \to P \qquad \mathsf{It}_\tau (\mathsf{s} n) P Q \to Q(\mathsf{It}_\tau n P Q)$$

Operator Q nie ma bezpośredniego dostępu do „zmiennej sterującej" n.

Ciekawostka historyczna

Typ rekursora

$\mathsf{R}_\tau : \mathsf{int} \to \tau \to (\mathsf{int} \to \tau \to \tau) \to \tau$

jest wytarciem schematu indukcji Peana w wersji z 1889 roku:

$$\forall x\,(\mathsf{int}(x) \to \tau(0) \to \forall y(\mathsf{int}(y) \to \tau(y) \to \tau(\mathsf{s}y)) \to \tau(x)).$$

Typ iteratora

$$\mathsf{It}_\tau : \mathsf{int} \to \tau \to (\tau \to \tau) \to \tau$$

jest wytarciem schematu indukcji Peana w wersji z 1891 roku:

$$\forall x\,(\mathsf{int}(x) \to \tau(0) \to \forall y(\tau(y) \to \tau(\mathsf{s}y)) \to \tau(x)).$$

Jak to można uogólnić?

Reguła dla iteratora:

$$\frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash P : \tau \quad \Gamma \vdash Q : \tau \to \tau}{\Gamma \vdash \mathsf{It}_\tau\, M\, P\, Q : \tau}$$

przepisana z użyciem koproduktu:

$$\frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash D : 1 \oplus \tau \to \tau}{\Gamma \vdash \mathsf{Elim}_\tau\, D\, M : \tau}$$

Redukcja:

$\mathsf{Elim}_\tau\, D\,(in\ N^{1\oplus \mathsf{int}}) \Rightarrow$
$D(\mathsf{case}\ N\ \mathsf{of}\ [x^1]\mathsf{inl}(x)\ \mathsf{or}\ [y^{\mathsf{int}}]\mathsf{inr}(\mathsf{Elim}_\tau\, Dy))$

Definicja indukcyjna

$$\frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash D : 1 \oplus \tau \to \tau}{\Gamma \vdash \mathsf{Elim}_\tau\, D\, M : \tau}$$

Mamy $D : 1 \oplus \tau \to \tau$, definiujemy $F = \mathsf{Elim}_\tau\, D : \mathsf{int} \to \tau$.

$\mathsf{Elim}_\tau\, D\,(in\ N^{1\oplus \mathsf{int}}) \Rightarrow$
$D(\mathsf{case}\ N\ \mathsf{of}\ [x^1]\mathsf{inl}(x)\ \mathsf{or}\ [y^{\mathsf{int}}]\mathsf{inr}(\mathsf{Elim}_\tau\, Dy))$

$F(in\ N^{1\oplus \mathsf{int}}) \Rightarrow D(\mathsf{case}\ N\ \mathsf{of}\ [x^1]\mathsf{inl}(x)\ \mathsf{or}\ [y^{\mathsf{int}}]\mathsf{inr}(Fy))$

Albo: $F(in\ N) \Rightarrow D(F^+(N))$,

gdzie $F^+(Z) = \mathsf{case}\ Z\ \mathsf{of}\ [x^1]\mathsf{inl}(x)\ \mathsf{or}\ [y^{\mathsf{int}}]\mathsf{inr}(Fy)$

Podsumowanie

$$\frac{\Gamma \vdash M : \mathsf{int} \quad \Gamma \vdash D : 1 \oplus \tau \to \tau}{\Gamma \vdash F(M) : \tau}$$

$F(in\ N^{1\oplus \mathsf{int}}) \Rightarrow D(\mathsf{case}\ N\ \mathsf{of}\ [x^1]\mathsf{inl}(x)\ \mathsf{or}\ [y^{\mathsf{int}}]\mathsf{inr}(Fy))$

$F^+(Z) = \mathsf{case}\ Z\ \mathsf{of}\ [x^1]\mathsf{inl}(x)\ \mathsf{or}\ [y^{\mathsf{int}}]\mathsf{inr}(Fy)$

$F(in\ N) \Rightarrow D(F^+(N))$

$F : \mathsf{int} \to \tau \qquad F^+ : 1 \oplus \mathsf{int} \to 1 \oplus \tau$

Lifting

For $F : \mathsf{int} \to \tau$ one defines $Lift(F) : 1 \oplus \mathsf{int} \to 1 \oplus \tau$

$Lift(F)(N) = \mathsf{case}\ N\ \mathsf{of}\ [x^1]\mathsf{inl}(x)\ \mathsf{or}\ [y^{\mathsf{int}}]\mathsf{inr}(FN)$

This generalizes to any $\sigma(p)$, if p is only positive in σ.

If $F : \rho \to \delta$, then $Lift(F) : \sigma(\rho) \to \sigma(\delta)$

A generalization

Assume p only positive in σ. Write $\mu = \mu p.\, \sigma(p)$.

$in : \sigma(\mu) \to \mu \qquad \mathsf{Elim}_\tau : (\sigma(\tau) \to \tau) \to \mu \to \tau$

Let $D : \sigma(\tau) \to \tau$. To define $F = \mathsf{Elim}_\tau\, D$ of type $\mu \to \tau$, we recursively apply F in the form of

$$F^+ : \sigma(\mu) \to \sigma(\tau).$$

Reduction rule:

$$F(in\ N) \Rightarrow D(F^+(N))$$

$$\mathsf{Elim}_\tau\, D\,(in\ N) \Rightarrow D(Lift_\sigma(\mathsf{Elim}_\tau\, D)N)$$

$$Lift_\sigma : (\mu \to \tau) \to \sigma(\mu) \to \sigma(\tau).$$

Twierdzenie Tarskiego-Knast(e)ra

Twierdzenie
Jeśli $\langle A, \leq \rangle$ jest kratą zupełną, to każda monotoniczna funkcja $f : A \to A$ ma najmniejszy punkt stały.

Dowód: Niech $B = \{x \in A \mid f(x) \leq x\}$; niech $a = \inf B$. Wtedy a jest najmniejszym punktem stałym funkcji f. □

Jeśli $A = \mathrm{P}(X)$, a porządek jest przez inkluzję, to

$$\mathrm{lfp}(f) = \bigcap \{x \mid f(x) \subseteq x\}$$

Reprezentacja w systemie **F**

W logice drugiego rzędu można definiować punkty stałe.

$$\mathrm{LFP}_\Phi(a) := \forall R((\Phi(R) \subseteq R) \to Ra)$$

Po wytarciu to wygląda tak:

$$\mu r.\,\sigma(r) = \forall r((\sigma(r) \to r) \to r).$$

Okazuje się, że to jest dobra definicja $\mu r.\,\sigma(r)$.

$\mu = \mu r.\,\sigma(r) = \forall r((\sigma(r) \to r) \to r)$

Włożenie $in : \sigma(\mu) \to \mu$

Iterator $\mathrm{Elim}_\tau : (\sigma(\tau) \to \tau) \to \mu \to \tau$

Redukcja $\mathrm{Elim}_\tau\, D\,(in\ N) \Rightarrow D(Lift(\mathrm{Elim}_\tau\, D)N)$

Definiujemy to w systemie **F**:

$\mathrm{Elim} = \Lambda r \lambda y^{\sigma(r)\to r} \lambda z^{\mu}.\, zry$

$in = \lambda x^{\sigma(\mu)} \Lambda r \lambda y^{\sigma(r)\to r}.\, y(Lift(\mathrm{Elim}\, r\, y)x)$,

gdzie $Lift : (\mu \to q) \to \sigma(\mu) \to \sigma(q)$

Ćwiczenie: policzyć, że przy tej definicji

$$\mathrm{Elim}_\tau\, D\,(in\ N) \to_\beta D(Lift(\mathrm{Elim}_\tau\, D)N)$$

$\mu r.\,\sigma(r) = \forall r((\sigma(r) \to r) \to r)$

Niech $\sigma(r) = 1 \oplus r$. Co to jest $\mu r.\,\sigma(r)$?

Wychodzi takie coś: $\mu r.\,1 \oplus r = \forall r((1 \oplus r \to r) \to r)$

Typ $\forall r((1 \oplus r \to r) \to r)$ to z grubsza to samo, co

$\forall r(((1 \to r) \times (r \to r)) \to r)$, czyli

$\forall r(r \times (r \to r) \to r)$, czyli

$\forall r(r \to (r \to r) \to r)$, czyli

$\forall r((r \to r) \to r \to r) = \omega$.

Tam i nazad?

Mamy $in : \sigma(\mu) \to \mu$, $\quad \mathrm{Elim}_\tau : (\sigma(\tau) \to \tau) \to \mu \to \tau$
i redukcję $\quad \mathrm{Elim}_\tau\, D\,(in\ N) \Rightarrow D(Lift(\mathrm{Elim}_\tau\, D)N)$

Czy umiemy zdefiniować operację $out : \mu \to \sigma(\mu)$?

Można tak: $out = \mathrm{Elim}_{\sigma(\mu)}(Lift(in))$,
gdzie $Lift(in) : \sigma(\sigma(\mu)) \to \sigma(\mu)$. Ale $out \circ in \neq \mathrm{id}$:

$$out(in\, N) \to Lift(in)(Lift(out)(N))$$

$$out \circ in = In \circ Out$$

Paradoks Reynoldsa, czyli „Polymorphism is not set-theoretic"

Niech $T(\alpha) = (\alpha \to \mathrm{bool}) \to \mathrm{bool}$.

Można zdefiniować typ $\mu = \mu p.\, T(p) = \forall p\,(T(p) \to p) \to p)$
i operację $\mathrm{in} : T(\mu) \to \mu$.

Część trudna: w modelu teoriomnogościowym:

– typ $\alpha \to \mathrm{bool}$ musi być interpretowany jako $\mathrm{P}(\alpha)$;

– typ $T(\alpha)$ jako $\mathrm{P}(\mathrm{P}(\alpha))$;

– funkcja $\mathrm{in} : T(\mu) \to \mu$ musi być różnowartościowa.

Zatem nie istnieje model teoriomnogościowy.

Typy ściśle pozytywne

Nieścisła definicja ścisłej pozytywności:

Typ $\mu p.\,\sigma(p)$ jest dozwolony tylko wtedy, gdy p nie występuje w $\sigma(p)$ w argumencie implikacji.

Przykład 1: ten typ nie jest ściśle pozytywny:

$$\mu p.\,0 \to (p \to 0) \to 0$$

Liczebniki Parigota:

$\overline{0} = \lambda x f.\, x$, $\quad \overline{1} = \lambda x f.\, f\overline{0}$, $\quad \overline{2} = \lambda x f.\, f\overline{1}$, ...

Przykład 2: ten typ jest ściśle pozytywny:

$$\omega\text{-}tree = \mu p.\,1 \oplus (\mathrm{int} \to p).$$

Zalety ścisłej pozytywności

- Takie typy najczęściej występują w praktyce;
- Łatwiejsza metateoria (np. dowodzenie SN);
- Mniejsze ryzyko paradoksu:
  typ $\mu p\,((p \to \mathrm{bool}) \to \mathrm{bool})$
  jest niebezpiecznie podobny do $A \approx \mathrm{P}(\mathrm{P}(A))$.
- Naturalne uogólnienie na typy zależne.
- Typy indukcyjne są ściśle pozytywne.

Rachunek konstrukcji

Jeśli τ jest typem, to piszemy $\tau : *$.

Czyli $*$ to *rodzaj* (*kind*) wszystkich typów.

Innym rodzajem jest np. $\tau \Rightarrow *$.

Piszemy $* : \Box$, $\tau \Rightarrow * : \Box$.

Czyli $\Box$ to *sort* wszystkich rodzajów.

Dependencies in lambda-calculi

The origin: Objects (terms) depend on objects:

$$\lambda x{:}\tau\, M^\sigma : \tau \to \sigma$$

Pattern: $(x : \tau^*) \to \sigma^* : *$ and rule: $(*, *, *)$.

(Arrow is a *constructor* of *kind* $* \Rightarrow * \Rightarrow *$.)

Polymorphism: Objects depend on types (system **F**):

$$\Lambda\alpha{:}* \, M^\sigma : \forall\alpha\, \sigma$$

Pattern: $(\alpha : *^\Box) \Rightarrow \sigma^* : *$ and rule: $(\Box, *, *)$.

Dependencies in lambda-calculi

Dependent types: Types depend on objects (system λP):

$$\lambda x{:}\tau\, \sigma(x)^* : \tau \Rightarrow *$$

Pattern: $(x : \tau^*) \Rightarrow *^\Box : \Box$ and rule: $(*, \Box, \Box)$

What is missing? Types depend on types (system $\lambda\underline{\omega}$)

$$\lambda\alpha{:}* \, \sigma(\alpha) : * \Rightarrow *$$

Pattern: $(\alpha : *^\Box) \Rightarrow *^\Box : \Box$ and rule: $(\Box, \Box, \Box)$.

Zależności

Początek układu: Obiekty mogą zależeć od obiektów:

$\lambda x{:}\tau\, M^\sigma : \tau \to \sigma$ $\qquad$ $(x : \tau^*) \to \sigma^* : *$

Oś X: Typy mogą zależeć od obiektów:

$\lambda x{:}\tau\, \sigma(x)^* : \tau \Rightarrow *$ $\qquad$ $(x : \tau^*) \to *^\Box : \Box$

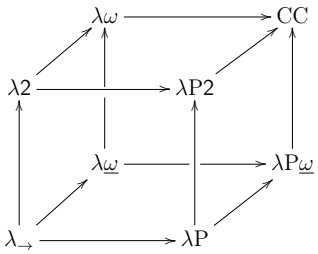
Oś Y: Obiekty mogą zależeć od typów:

$\Lambda\alpha{:}* \, M^\sigma : \forall\alpha\, \sigma$ $\qquad$ $(\alpha : *^\Box) \to \sigma^* : *$.

Oś Z: Typy mogą zależeć od typów:

$\lambda\alpha{:}* \, \sigma(\alpha) : * \Rightarrow *$ $\qquad$ $(\alpha : *^\Box) \to *^\Box : \Box$

Barendregt's Cube: combine all dependencies



$\lambda\!\to$ to typy proste, $\lambda 2$ to system **F**, $\lambda\omega$ to system $\mathbf{F}_\omega$.
CC to *rachunek konstrukcji*

Examples: dependent types

Niech $\Gamma = \{\tau : *, \alpha : \tau \Rightarrow *, \beta : \tau \Rightarrow *\}$.

- Term $\lambda x^{\forall z:\tau.\, \alpha z \to \beta z} \lambda y^{\forall z:\tau.\, \alpha z} \lambda z^\tau.\, xz(yz)$ ma w Γ typ $(\forall z{:}\tau.\, \alpha z \to \beta z) \to (\forall z{:}\tau.\, \alpha z) \to \forall z{:}\tau.\, \beta z$.
- Term $\lambda f^{\tau\to\tau} \lambda y^{\forall x:\tau.\, \alpha x \to \alpha(fx)} \lambda x^\tau \lambda z^{\alpha x}.\, y(fx)(yxz)$ ma typ $\forall f{:}\tau\to\tau.\, (\forall x{:}\tau.\, \alpha x \to \alpha(fx)) \to \forall x{:}\tau.\, \alpha x \to \alpha(f^2(x))$.
- Term $\lambda y^\tau \lambda f^{\forall(x:\tau)\alpha(x)} \lambda g^{\alpha(y)\to\tau}.\, f(g(fy))$ ma typ $\forall y^\tau \forall f^{\forall(x:\tau)\alpha(x)} \forall g^{\alpha(y)\to\tau}.\, \alpha(g(f(y)))$.

Polimorfizm wyższego rzędu: system F_ω

Example: Why is $(\lambda zy.\, y(zI)(zK))(\lambda x.\, xx)$ untypable in **F**?

Because types we could assign to I and K differ too much:

$$I : \forall p.\, p \to p \qquad K : \forall p.\, p \to (q \to p).$$

There is no common pattern for these two types...
unless we write it this way:

$$I, K : \forall p.\, p \to \alpha(p).$$

Here, α is a constructor of kind $* \Rightarrow *$.

Niech $\zeta = \forall\alpha^{*\Rightarrow*}.\, (\forall p.\, p \to \alpha(p)) \to (\forall p.\, \alpha(p \to \alpha(p)))$

i niech $\mathrm{omega} = \Lambda\alpha\lambda x^{\forall p.\, p\to\alpha(p))} \Lambda p.\, x(p \to \alpha(p))(xp) : \zeta$.

Teraz można napisać term

$(\lambda z^\zeta \lambda y^?.\, y(z(\lambda p.p)I)(z(\lambda p.q{\to}p)K))\,\mathrm{omega} : \zeta \to ? \to ?$

Calculus of Constructions: Three levels

- Terms, e.g., $\lambda x^{\forall z:q.\, \alpha(z)} \lambda y^{p\to q} v^p.\, x(yv)$;
- Types and constructors, e.g., $\lambda x^\tau.\, \alpha x \to p$, $\quad \lambda p^*.\, p \to p$;
- Kinds, e.g., $\tau \Rightarrow \sigma \Rightarrow *$, $\quad * \Rightarrow *$.

$$M : \tau : * : \Box$$
$$\phi : \kappa : \Box$$

No "absolute" notion of term, type, etc.

Is αx is a legal type or not?
It depends on the kind of α and the type of x.

Environment is a **sequence** of declarations, not a set.

Calculus of Constructions: Raw syntax

Sorts:
$s ::= * \mid \Box$;

Kinds:
$\kappa ::= * \mid (\Pi x{:}\phi\ \kappa) \mid (\Pi x{:}\kappa\ \kappa)$;

Constructors:
$\phi ::= \alpha \mid (\forall x{:}\phi\ \phi) \mid (\forall \alpha{:}\kappa\ \phi) \mid (\phi M) \mid (\phi\phi) \mid (\lambda x{:}\phi\ \phi) \mid (\lambda\alpha{:}\kappa\ \phi)$;

Terms:
$M ::= x \mid (MM) \mid (M\phi) \mid (\lambda x{:}\phi\ M) \mid (\lambda\alpha{:}\kappa\ M)$;

Environments:
$\Gamma ::= \varnothing \mid \Gamma, (x : \phi) \mid \Gamma, (\alpha : \kappa)$.

Rachunek konstrukcji: składnia uniwersalna

Sorty: $s ::= * \mid \Box$

Pseudo-termy:

$$A ::= x \mid s \mid (AA) \mid (\lambda x{:}A\,A) \mid (\Pi x{:}A\,A)$$

Redukcja:

$$(\lambda x{:}A.\,B)C \quad \to_\beta \quad B[x := C].$$

Rachunek konstrukcji: reguły

(Ax) $\quad \varnothing \vdash * : \Box$

(Var) $\quad \dfrac{\Gamma \vdash A : s}{\Gamma,\, x : A \vdash x : A} \quad (x \notin \mathrm{Dom}(\Gamma))$

(Prod) $\quad \dfrac{\Gamma \vdash A : s_1 \qquad \Gamma,\, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x{:}A.\,B) : s_2} \quad (s_1, s_2 \in \{*, \Box\})$

Rachunek konstrukcji: reguły

(Prod) $\quad \dfrac{\Gamma \vdash A : s_1 \qquad \Gamma,\, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x{:}A.\,B) : s_2} \quad (s_1, s_2 \in \{*, \Box\})$

(Abs) $\quad \dfrac{\Gamma,\, x : A \vdash B : C \qquad \Gamma \vdash (\Pi x{:}A.\,C) : s}{\Gamma \vdash (\lambda x{:}A.\,B) : (\Pi x{:}A.\,C)}$

(App) $\quad \dfrac{\Gamma \vdash A : (\Pi x{:}B.\,C) \qquad \Gamma \vdash D : B}{\Gamma \vdash (AD) : C[x := D]}$

Rachunek konstrukcji: reguły

(Weak) $\quad \dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma,\, x : C \vdash A : B} \quad (x \notin \mathrm{Dom}(\Gamma))$

(Conv) $\quad \dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'} \quad (B =_\beta B')$

Rachunek konstrukcji: własności

- Poprawność redukcji:
  Jeśli $\Gamma \vdash A : B$ i $A \twoheadrightarrow A'$ to $\Gamma \vdash A' : B$.
- "Wzmacnianie":
  Jeśli $\Gamma, x : A, \Delta \vdash C : B$,
  oraz x nie jest wolne w Δ, C ani w B,
  to $\Gamma, \Delta \vdash C : B$.
- Jednoznaczność typu:
  Jeśli $\Gamma \vdash A : B$ i $\Gamma \vdash A : B'$, to $B =_\beta B'$.
- Silna normalizacja: Jeśli $\Gamma \vdash A : B$, to $A \in SN$.

Klasyfikacja w rachunku konstrukcji

Fakt:
Jeśli $\Gamma \vdash A : B$ to zachodzi dokładnie jeden przypadek:

– albo $\Gamma \vdash B : *$ (czyli A jest obiektem typu B),
– albo $\Gamma \vdash B : \Box$ (czyli A jest konstruktorem rodzaju B),
– albo $B = \Box$ (czyli A jest rodzajem).

Rachunek konstrukcji: podsumowanie

- Dwa *sorty*: $*$ oraz $\Box$;
- Aksjomat $* : \Box$;
- Dozwolone są produkty postaci
  $\Pi x{:}A^s.\,B(x)^t$, czyli $(x : A^s) \to B(x)^t : t$,
  gdzie s i t są dowolnymi sortami;
- Te produkty są typami abstrakcji $\lambda x{:}A^s.\,M^{B(x)}$;
- Wierzchołki kostki: ograniczenie w tworzeniu produktów.

Uogólnienie: Pure Type Systems

- Zbiór sortów $\mathcal{S}$;
- Aksjomaty $\mathcal{A}$ postaci $(s : t)$, gdzie $s, t \in \mathcal{S}$;
- Reguły $\mathcal{R}$ postaci (s, t, u);
- Tworzenie produktów, gdy $(s, t, u) \in \mathcal{R}$

$$\frac{\Gamma \vdash A : s \qquad \Gamma, x : A \vdash B : t}{\Gamma \vdash (\Pi x{:}A.\, B) : u}$$

- Konwencja: zamiast (s, t, t) piszemy (s, t).
- Reszta jak w rachunku konstrukcji.

Dozwolone produkty

- Tworzenie produktów, gdy $(s, t, u) \in \mathcal{R}$

$$\frac{\Gamma \vdash A : s \qquad \Gamma, x : A \vdash B : t}{\Gamma \vdash (\Pi x{:}A.\, B) : u}$$

- Inaczej: dozwolone są „typy" postaci

$$(x : A^s) \to B(x)^t : u.$$

Przykłady PTS

- Cały rachunek konstrukcji.
- Wierzchołki kostki, np λP ma reguły $(*, *)$, $(*, \Box)$, a system **F** ma reguły $(*, *)$, $(\Box, *)$.
- System "Type is a type" ma jeden sort $*$, aksjomat $(* : *)$ i regułę $(*, *)$.
- System λ2-ML ma trzy sorty: $*$, $\Box$ i $\triangle$ (schemat typu). Aksjomat: $(* : \Box)$. Reguły: $(*, *)$, $(\Box, *, \triangle)$, $(\Box, \triangle)$.

  Produkty to zwykłe typy $(x : \tau^*) \to \sigma^* : *$, i schematy typów $\forall \alpha_1 \ldots \alpha_n.\, \sigma$, czyli $(\alpha_1 : *^{\Box}) \to \cdots \to (\alpha_n : *^{\Box}) \to \sigma^* : \triangle$

Przykłady PTS

- System λPRED (Berardi-Terlouw): sorty $*^t, *^p, \Box^p, \Box^t$ i aksjomaty $(*^t : \Box^t)$, $(*^p : \Box^p)$. („propositions $\neq$ sets")
- System λC^{ω}: rachunek konstrukcji z nieskończoną hierarchią sortów (uniwersów): $* : \Box_0 : \Box_1 : \Box_2 \ldots$ ma regułę $(\Box_i, \Box_j, \Box_{\max\{i,j\}})$.
- Podobnie w Coq: `Set:Type(0)`, `Prop:Type(0)`, `Type(n):Type(n+1)`.

„Typical ambiguity"

```
  Coq < Check Set.

Set
     : Type

Coq < Check Prop.

Prop
     : Type

Coq < Check Type.

Type
     : Type
```


Naiwna Teoria Typów

- Slogan: Zbiór to predykat.
- Reguła $(*, \Box, *)$ daje typ potęgowy $\mathrm{P}(\tau) = \tau \to * : *$. Ale ten system jest sprzeczny.
- Mniej Naiwna Teoria Typów Agnieszki Kozubek ma sorty $*^t, *^p, \Box^p, \Box^t$, aksjomaty $(*^t : \Box^t)$, $(*^p : \Box^p)$, i reguły: $(*^t, *^t)$, $(*^p, *^p)$, $(*^t, *^p)$, $(*^t, \Box^p, *^t)$, $(*^t, \Box^t)$.
- Zbiór $\{x : \tau \mid \varphi(x)\}$ obiektów typu τ, to predykat $\lambda x{:}\tau.\, \varphi(x)$ typu $(x : \tau) \to *^p : *^t$

Paradoksy

Paradoksy w teorii zbiorów

Paradoks Russella:

Nie istnieje zbiór A o własności $\mathrm{P}(A) \subseteq A$.

Dowód: Przypuśćmy, że $\mathrm{P}(A) \subseteq A$ i niech

$$\Delta = \{a \in A \mid a \notin a\}.$$

Wtedy $\Delta \subseteq A$, więc $\Delta \in A$.

Jeśli $\Delta \in \Delta$, to $\Delta \notin \Delta$ – sprzeczność.

Jeśli $\Delta \notin \Delta$, to $\Delta \in \Delta$ (bo $\Delta \in A$) – też sprzeczność.

Paradoks Cantora

Twierdzenie: *Nie istnieje funkcja* $el : \mathrm{P}(A) \xrightarrow{1-1} A$.

Dowód: Jeśli $el : \mathrm{P}(A) \xrightarrow{1-1} A$, to istnieje taka funkcja $set : A \xrightarrow{\mathrm{na}} \mathrm{P}(A)$, że $set \circ el = \mathrm{id}_{\mathrm{P}(A)}$ (lewa odwrotna).

Niech $\Delta = \{a \in A \mid a \notin set(a)\}$ oraz $\delta = el(\Delta)$.

Wtedy $\delta \in A$, oraz $set(\delta) = set(el(\Delta)) = \Delta$.

Jeśli $\delta \in \Delta$, to $\delta \notin set(\delta) = \Delta$.

Jeśli $\delta \notin \Delta$, to $\delta \in set(\delta) = \Delta$.

Paradoks Cantora

Twierdzenie: *Nie istnieje funkcja* $el : \mathrm{P}(A) \xrightarrow{1-1} A$.

Tym bardziej: *Nie istnieje funkcja* $el : \mathrm{P}(A) \xrightarrow[\mathrm{na}]{1-1} A$.

Inaczej: *Nie istnieją takie funkcje* $set : A \to \mathrm{P}(A)$, *oraz* $el : \mathrm{P}(A) \to A$, *że* $set \circ el = \mathrm{id}_{\mathrm{P}(A)}$ *oraz* $el \circ set = \mathrm{id}_A$.

Paradoks Cantora

Twierdzenie: *Nie istnieją takie funkcje* $set : A \to \mathrm{P}(A)$, *oraz* $el : \mathrm{P}(A) \to A$, *że* $set \circ el = \mathrm{id}_{\mathrm{P}(A)}$ *oraz* $el \circ set = \mathrm{id}_A$.

Uwaga: Gdyby tak było, to niech

$El : \mathrm{P}(\mathrm{P}(A)) \to \mathrm{P}(A)$ $\quad Set : \mathrm{P}(A) \to \mathrm{P}(\mathrm{P}(A))$

$El(\mathcal{X}) = \{el(Y) \mid Y \in \mathcal{X}\}$ $\quad Set(X) = \{set(a) \mid a \in X\}$.

Wtedy $El(Set(X)) = \{el(set(x)) \mid x \in X\} = X$.

Czyli $El \circ Set = \mathrm{id}_{\mathrm{P}(A)}$.

W szczególności $El \circ Set = set \circ el$.

Paradoks Girarda-Hurkensa

Twierdzenie: *Nie istnieją takie funkcje* $set : A \to \mathrm{P}(A)$, *oraz* $el : \mathrm{P}(A) \to A$, *że* $set \circ el = El \circ Set$, *gdzie:*

$El(\mathcal{X}) = \{el(Y) \mid Y \in \mathcal{X}\}$ $\quad Set(X) = \{set(a) \mid a \in X\}$.