Logika i teoria typów

Wykład 12

18 stycznia 2020

Rachunek konstrukcji: podsumowanie

- Dwa *sorty*: $*$ oraz $\Box$;
- Aksjomat $* : \Box$;
- Dozwolone są produkty postaci
$$(x : A^s) \to B(x)^t : t,$$
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 $\lambda 2$-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 $\mathbf{P}(\tau) = \tau \to * : *$. 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: $(*^t, *^t)$, $(*^p, *^p)$, $(*^t, *^p)$, $(*^t, \Box^p, *^t)$, $(*^t, \Box^t, \Box^t)$.
- 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 $\mathbf{P}(A) \subseteq A$.

Dowód: Przypuśćmy, że $\mathbf{P}(A) \subseteq A$ 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* $el : \mathbf{P}(A) \xrightarrow{1-1} A$.

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

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

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

Paradoks Cantora

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

Uwaga: Gdyby tak było, to niech

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

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

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

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

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

Paradoks Girarda-Hurkensa

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

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

Założenia: $set : A \to \mathbf{P}(A)$, $el : \mathbf{P}(A) \to A$, $set \circ el = El \circ Set$.

Definicje:

- Dla $a \in A$ piszemy $a' = el(set(a))$.
- Relacja $\approx$ to najmniejsza relacja równoważności, spełniająca $a \approx a'$ dla każdego $a \in A$.

Główny pomysł:

Elementy zbioru $X \subseteq A$ wyznaczają te same klasy abstrakcji relacji $\approx$, co elementy zbioru

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

Zatem funkcja set jest „na" z dokładnością do $\approx$.

Ściślej: niech $\overline{X} = \{[a]_\approx \mid a \in X\}$ dla $X \subseteq A$ i niech $X \equiv Y$ oznacza $\overline{X} = \overline{Y}$. Wtedy $X \equiv set(el(X))$ dla dowolnego X.

Założenia: $set : A \to \mathbf{P}(A)$, $el : \mathbf{P}(A) \to A$, $set \circ el = El \circ Set$.

Definicje:

- Dla $a \in A$ piszemy $a' = el(set(a))$.
- Relacja $\approx$ to najmniejsza relacja równoważności, spełniająca $a \approx a'$ dla każdego $a \in A$.
- Piszemy $a \varepsilon b$, gdy $a \approx x \in set(b)$.

Lemat: (1) *Jeśli* $a \in set(b)$, *to* $a' \in set(b')$.
(2) *Jeśli* $a \in set(b')$, *to* $a = c'$ *gdzie* $c \in set(b)$.

Dowód: (1) Niech $a \in set(b)$. Wtedy:
$a' = el(set(a)) \in El(Set(set(b))) = set(el(set(b))) = set(b')$.

(2) Jeśli $a \in set(b') = set(el(set(b))) = El(Set(set(b)))$, to $a = el(set(c)) = c'$, gdzie $c \in set(b)$.

Założenia: $set : A \to \mathbf{P}(A)$, $el : \mathbf{P}(A) \to A$, $set \circ el = El \circ Set$.

- Dla $a \in A$ piszemy $a' = el(set(a))$.
- Relacja $\approx$ to najmniejsza relacja równoważności, spełniająca $a \approx a'$ dla każdego $a \in A$.
- Piszemy $a \varepsilon b$, gdy $a \approx x \in set(b)$.

Lemat: *Relacja $\approx$ jest kongruencją ze względu na ε. W szczególności $a \varepsilon b \Leftrightarrow a \varepsilon b'$.*

Dowód: Wystarczy sprawdzić, że:

1. Jeśli $c \approx a \varepsilon b$, to $c \varepsilon b$ (z definicji);
2. Jeśli $a \varepsilon b$, to $a \varepsilon b'$ (bo $x \in set(b) \Rightarrow x' \in set(b')$);
3. Jeśli $a \varepsilon b'$, to $a \varepsilon b$ (bo jeśli $x \in set(b')$, to $x = y'$, gdzie $y \in set(b)$).

Paradoks Girarda-Hurkensa

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

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

Założenia: $set : A \to \mathbf{P}(A)$, $el : \mathbf{P}(A) \to A$, $set \circ el = El \circ Set$

Oznaczenia: $a' = el(set(a))$, $\quad a \varepsilon b \Leftrightarrow a \approx x \in set(b)$

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

Lemat: *Dla $a \in A$ zachodzi $a \varepsilon \delta \Leftrightarrow a \notin a$.*

Dowód: ($\Leftarrow$) Jeśli $a \notin a$, to $a \in \Delta$. Wtedy: $a \approx a' = el(set(a)) \in El(Set(\Delta)) = set(el(\Delta)) = set(\delta)$. Czyli właśnie $a \varepsilon \delta$.

($\Rightarrow$) Wtedy: $a \approx x \in set(\delta) = set(el(\Delta)) = El(Set(\Delta))$. Inaczej, $a \approx x = el(set(d)) = d'$, gdzie $d \in \Delta$. Skoro $d \notin d$ oraz $a \approx d$, to $a \notin a$.

Morał: Sprzeczność, bo $\delta \varepsilon \delta \Leftrightarrow \delta \notin \delta$.

Sprzeczne otoczenie typowe:

W tym otoczeniu można sformalizować paradoks Hurkensa tj., skonstruować term typu $\bot = \forall p : *.\, p$:

$\kappa : \Box,\ el : \mathbf{P}(\kappa) \to \kappa,\ set : \kappa \to \mathbf{P}(\kappa)$,

$v : \forall X^{\mathbf{P}(\kappa)} \forall \alpha^{\kappa} (set(elX)\alpha \leftrightarrow \exists \beta^{\kappa} (X\beta \wedge \alpha =^{\kappa} el(set\beta)))$.

Inaczej: $v : set \circ el = El \circ Set$ (ekstensjonalnie).

Trzeba teraz zdefiniować odpowiedni rodzaj κ.

Formalizacja

- Jeśli zbiór reprezentujemy w CC jako typ τ, to potęga $\mathbf{P}(\tau) = (\tau \Rightarrow *)$ jest rodzajem.
- Jeśli zbiór reprezentujemy w CC jako rodzaj κ, to potęga $\mathbf{P}(\kappa) = (\kappa \Rightarrow *)$ też jest rodzajem.
- Równość Leibniza:

  $\alpha =^{\kappa} \beta$ oznacza $\forall \gamma^{\kappa \to *}.\, \gamma\alpha \to \gamma\beta$.
- Jeśli $F : \kappa \to \varrho$, to $Lift\, F : \mathbf{P}(\kappa) \to \mathbf{P}(\varrho)$, gdzie:

  $Lift\, F = \lambda X^{\mathbf{P}(\kappa)} \lambda y^{\varrho} (\exists x^{\kappa}.\, Fx =^{\varrho} y)$.

  Moralnie $Lift\, F\, X$ to obraz X przy F.

System sprzeczny

- System λU: rozszerzenie CC o sort $\bigtriangleup$, aksjomat $\Box : \bigtriangleup$ i reguły $(\bigtriangleup, \Box)$ i $(\bigtriangleup, *)$.
- System λU^{-}: bez tej drugiej reguły.
- To jest „polimorfizm rodzajowy": można definiować polimorficzne konstruktory, np.:

  $\lambda \kappa^{\Box} \lambda \alpha^{\kappa \to \kappa} \lambda \beta^{\kappa}.\, \alpha(\alpha\beta) : \Pi \kappa{:}\Box((\kappa \to \kappa) \to \kappa \to \kappa)$.

Rodzaj paradoksalny

$$\kappa = \mu k^{\Box} \mathbf{P}(k) = \forall k^{\Box}((\mathbf{P}(k) \to k) \to k).$$

Można tak zdefiniować funkcje *el* i *set*, że otrzymamy otoczenie paradoksalne:

$\mathrm{Elim} = \lambda k^{\Box} \lambda f^{\mathbf{P}(k) \to k} \lambda z^{\kappa}.\, zkf$

$el = in = \lambda X^{\mathbf{P}(\kappa)} \Lambda k^{\Box} \lambda y^{\mathbf{P}(k) \to k}.\, y(Lift(\mathrm{Elim}\, k\, y)X)$,

$set = out = \mathrm{Elim}_{\mathbf{P}(\kappa)}(Lift(in))$.

Wtedy $set \circ el = El \circ Set$.

$\kappa = \mu k^{\Box} \mathbf{P}(k) = \forall k^{\Box}((\mathbf{P}(k) \to k) \to k)$

$\mathrm{Elim} = \lambda k^{\Box} \lambda f^{\mathbf{P}(k) \to k} \lambda z^{\kappa}.\, zkf$

$el = in = \lambda X^{\mathbf{P}(\kappa)} \Lambda k^{\Box} \lambda y^{\mathbf{P}(k) \to k}.\, y(Lift(\mathrm{Elim}\, k\, y)X)$,

$set = out = \mathrm{Elim}(\mathbf{P}(\kappa))(Lift(in))$.

Liczymy:

$set(el(X^{\mathbf{P}(\kappa)})) = \mathrm{Elim}(\mathbf{P}(\kappa))(Lift(in))(in(X)) =$

$in(X)(\mathbf{P}(\kappa))(Lift(in)) =$

$(Lift(in)(Lift[\mathrm{Elim}(\mathbf{P}(\kappa))(Lift(in))](X)) =$

$(Lift(in)(Lift[out](X)) =$

$El(Set(X))$.

Inne rozwiązanie

$$\kappa = \forall k^{\Box}(\forall \iota^{\Box}((\iota \to k) \to \mathbf{P}(\iota) \to k) \to k).$$

Motywacja: to jeszcze inny sposób na definicję punktu stałego. Ten typ jest wytarciem schematu definiującego $x \in \mathrm{LFP}(P)$:

$$\forall K(\forall J(J \subseteq K \to P(J) \subseteq K) \to x \in K)$$

Inne rozwiązanie: szczegóły dla dociekliwych

$$el = \lambda X^{\mathbf{P}(\kappa))} \lambda k^{\Box} \lambda y^{\forall \iota:\Box.(\iota \to k) \to \mathbf{P}(\iota)) \to k}. y\boldsymbol{\kappa}(\lambda \beta^{\kappa}. \beta k y) X;$$
$$set = \lambda \beta^{\kappa}. \beta(\mathbf{P}(\boldsymbol{\kappa}))\psi,$$

gdzie

$$\psi = \lambda \iota^{\Box} \lambda f^{\iota \to \mathbf{P}(\kappa))} \lambda X^{\mathbf{P}(\iota)} \lambda \alpha^{\kappa}. \exists \beta^{\iota}(X\beta \wedge \alpha =^{\kappa} el(f\beta)).$$

Rachunki:

$$\begin{aligned} set(elX)\alpha &=_{\beta} (\lambda\beta.\, \beta(\mathbf{P}(\boldsymbol{\kappa}))\psi)(elX)\alpha =_{\beta} elX(\mathbf{P}(\boldsymbol{\kappa}))\psi\alpha \\ &=_{\beta} \psi\kappa(\lambda\beta.\, \beta(\mathbf{P}(\kappa))\psi)X\alpha = \psi\kappa\, set X\alpha \\ &=_{\beta} \exists\beta(X\beta \wedge \alpha =^{\kappa} el(set\,\beta)), \end{aligned}$$

Type is not a type

System "type is a type" jest sprzeczny. Tę samą konstrukcję można powtórzyć zamieniając wszędzie $\Box$ i ∇ na $*$.

Zakręcone kombinatory (?)

System "type is a type" jest sprzeczny, więc nie ma własności SN. Hipoteza: w tym systemie można zdefiniować kombinator punktu stałego:

$$YF \twoheadrightarrow F(YF)$$

Wiadomo, że są tzw. looping combinators:

$$Y_0F \twoheadrightarrow F(Y_1F) \twoheadrightarrow F^2(Y_2F) \twoheadrightarrow F^3(Y_3F) \twoheadrightarrow \cdots$$

Inne konstrukcje sprzeczne

Podobne paradoksy pojawiają się m.in. w systemach dopuszczających:

- "Silną sumę" tj. utożsamienie $\exists\alpha : *.\, \sigma$ z produktem zależnym $(\alpha : *) \times \sigma$, dopuszczającym rzutowanie $\pi_1 : \exists\alpha : *.\, \sigma \to *$ o własności $\pi_1[\tau, M] = \tau$;
- Rodzaje indukcyjne nie-ściśle pozytywne, np. rodzaj κ z konstruktorem typu $((\kappa \to *) \to *) \to \kappa$, czyli typu $\mathbf{P}(\mathbf{P}(\kappa)) \to \kappa$.

„Silna suma"

Do sprzeczności prowadzi konstrukcja

$$B = (\exists\alpha : *.\mathit{Unit}) : *,$$

wraz z włożeniem $tam : * \to B$,

i rzutowaniem $nazad : B \to *$,

spełniającymi równanie $nazad(tam(\alpha)) = \alpha$, dla $\alpha : *$

To jest w istocie włożenie $*$ w pewien typ, co daje taki sam efekt jak $* : *$.

Zależny iloczyn kartezjański w Coqu?

```
Inductive Be :=
beintro: forall (a:Prop), unit -> Be.

Coq < Be is defined
Be_rect is defined
Be_ind is defined
Be_rec is defined

Coq < Check Be.
Be
     : Type
```

Rzut na pierwszą współrzędną:

```
Check Be_rect.
Be_rect
  : forall P : Be -> Type,
  (forall (a : Prop) (u : unit), P (beintro a u))
             -> forall b : Be, P b


Definition Rzut := Be_rect(fun X: Be => Prop)
(fun a : Prop => fun u : unit => a).


Coq < Eval simpl in (Rzut (beintro R tt)):Prop.
     = R
     : Prop
```

```
Inductive Be :=
beintro: forall (a:Prop), unit -> Be.

Coq < Be is defined
Be_rect is defined
Be_ind is defined
Be_rec is defined


Inductive Ce : Prop :=
ceintro: forall (a:Prop), unit -> Ce.

Coq < Ce is defined
Ce_ind is defined
```