

Logika i teoria typów

Wykład 12

18 stycznia 2020

Rachunek konstrukcji: podsumowanie

▶ Dwa *sorty*: $*$ oraz \square ;

▶ Aksjomat $*$: \square ;

▶ Dozwolone są produkty postaci

$$(x : A^s) \rightarrow B(x)^t : t,$$

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 $(s, t, u) \in \mathcal{R}$

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

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

Dozwolone produkty

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

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

- ▶ Inaczej: dozwolone są „typy” postaci

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

Przykłady PTS

- ▶ Cały rachunek konstrukcji.

Przykłady PTS

- ▶ Cały rachunek konstrukcji.
- ▶ Wierzchołki kostki, np λP ma reguły $(*, *)$, $(\square, *)$, a system **F** ma reguły $(*, *)$, $(*, \square)$.

Przykłady PTS

- ▶ Cały rachunek konstrukcji.
- ▶ Wierzchołki kostki, np λP ma reguły $(*, *)$, $(\square, *)$, a system \mathbf{F} ma reguły $(*, *)$, $(*, \square)$.
- ▶ System “Type is a type” ma jeden sort $*$, aksjomat $(* : *)$ i regułę $(*, *)$.

Przykłady PTS

- ▶ Cały rachunek konstrukcji.
- ▶ Wierzchołki kostki, np λP ma reguły $(*, *)$, $(\square, *)$, a system \mathbf{F} ma reguły $(*, *)$, $(*, \square)$.
- ▶ System “Type is a type” ma jeden sort $*$, aksjomat $(* : *)$ i regułę $(*, *)$.
- ▶ System $\lambda 2\text{-ML}$ ma trzy sorty: $*$, \square i Δ (schemat typu). Aksjomat: $(* : \square)$. Reguły: $(*, *)$, $(\square, *, \Delta)$, (\square, Δ) .

Produkty to zwykłe typy $(x : \tau^*) \rightarrow \sigma^* : *$,
i schematy typów $\forall \alpha_1 \dots \alpha_n. \sigma$, czyli
 $(\alpha_1 : *^{\square}) \rightarrow \dots \rightarrow (\alpha_n : *^{\square}) \rightarrow \sigma^* : \Delta$

Przykłady PTS

- ▶ System λ PRED (Berardi-Terlouw):
sorty $*^t, *^p, \square^p, \square^t$ i aksjomaty $(*^t : \square^t), (*^p : \square^p)$.
(„propositions \neq sets”)

Przykłady PTS

- ▶ System λPRED (Berardi-Terlouw):
sorty $*^t, *^p, \square^p, \square^t$ i aksjomaty $(*^t : \square^t), (*^p : \square^p)$.
(„propositions \neq sets”)
- ▶ System λC^ω : rachunek konstrukcji z nieskończoną hierarchią sortów (uniwersów): $* : \square_0 : \square_1 : \square_2 \dots$ ma regułę $(\square_i, \square_j, \square_{\max\{i,j\}})$.

Przykłady PTS

- ▶ System λ PRED (Berardi-Terlouw):
sorty $*^t, *^p, \square^p, \square^t$ i aksjomaty $(*^t : \square^t), (*^p : \square^p)$.
(„propositions \neq sets”)
- ▶ System λC^ω : rachunek konstrukcji z nieskończoną hierarchią sortów (uniwersów): $* : \square_0 : \square_1 : \square_2 \dots$ ma regułę $(\square_i, \square_j, \square_{\max\{i,j\}})$.
- ▶ Podobnie w Coq:
 $\text{Set} : \text{Type}(0), \text{Prop} : \text{Type}(0), \text{Type}(n) : \text{Type}(n+1)$.

„Typical ambiguity”

„Typical ambiguity”

Coq < Check Set.

„Typical ambiguity”

Coq < Check Set.

Set

: Type

„Typical ambiguity”

Coq < Check Set.

Set

: Type

Coq < Check Prop.

„Typical ambiguity”

Coq < Check Set.

Set
: Type

Coq < Check Prop.

Prop
: Type

„Typical ambiguity”

Coq < Check Set.

Set

: Type

Coq < Check Prop.

Prop

: Type

Coq < Check Type.

„Typical ambiguity”

```
Coq < Check Set.
```

```
Set  
  : Type
```

```
Coq < Check Prop.
```

```
Prop  
  : Type
```

```
Coq < Check Type.
```

```
Type  
  : Type
```

Naiwna Teoria Typów

- ▶ Slogan: Zbiór to predykat.

Naiwna Teoria Typów

- ▶ Slogan: Zbiór to predykat.
- ▶ Reguła $(*, \square, *)$ daje typ potęgowy $\mathbf{P}(\tau) = \tau \rightarrow * : *$.
Ale ten system jest sprzeczny.

Naiwna Teoria Typów

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

Paradoksy

Paradoksy w teorii zbiorów

Paradoks Russella:

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

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 $e_l : \mathbf{P}(A) \xrightarrow{1-1} A$.*

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 $set \circ el = \text{id}_{\mathbf{P}(A)}$ (lewa odwrotna).

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 $set \circ el = \text{id}_{\mathbf{P}(A)}$ (lewa odwrotna).

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

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 $set \circ el = \text{id}_{\mathbf{P}(A)}$ (lewa odwrotna).

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

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

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 $set \circ el = \text{id}_{\mathbf{P}(A)}$ (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 $e_l : \mathbf{P}(A) \xrightarrow{1-1} A$.*

Paradoks Cantora

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

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

Paradoks Cantora

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

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

Inaczej: *Nie istnieją takie funkcje $set : A \rightarrow \mathbf{P}(A)$,
oraz $el : \mathbf{P}(A) \rightarrow A$, że $set \circ el = id_{\mathbf{P}(A)}$ oraz $el \circ set = id_A$.*

Paradoks Cantora

Twierdzenie: Nie istnieją takie funkcje $set : A \rightarrow \mathbf{P}(A)$,
oraz $el : \mathbf{P}(A) \rightarrow A$, że $set \circ el = id_{\mathbf{P}(A)}$ oraz $el \circ set = id_A$.

Paradoks Cantora

Twierdzenie: Nie istnieją takie funkcje $set : A \rightarrow \mathbf{P}(A)$,
oraz $el : \mathbf{P}(A) \rightarrow A$, że $set \circ el = id_{\mathbf{P}(A)}$ oraz $el \circ set = id_A$.

Uwaga: Gdyby tak było, to niech

$$El : \mathbf{P}(\mathbf{P}(A)) \rightarrow \mathbf{P}(A)$$

$$Set : \mathbf{P}(A) \rightarrow \mathbf{P}(\mathbf{P}(A))$$

$$El(\mathcal{X}) = \{el(Y) \mid Y \in \mathcal{X}\}$$

$$Set(X) = \{set(a) \mid a \in X\}.$$

Paradoks Cantora

Twierdzenie: Nie istnieją takie funkcje $set : A \rightarrow \mathbf{P}(A)$,
oraz $el : \mathbf{P}(A) \rightarrow A$, że $set \circ el = id_{\mathbf{P}(A)}$ oraz $el \circ set = id_A$.

Uwaga: Gdyby tak było, to niech

$$El : \mathbf{P}(\mathbf{P}(A)) \rightarrow \mathbf{P}(A) \qquad Set : \mathbf{P}(A) \rightarrow \mathbf{P}(\mathbf{P}(A))$$
$$El(\mathcal{X}) = \{el(Y) \mid Y \in \mathcal{X}\} \qquad Set(X) = \{set(a) \mid a \in X\}.$$

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

Czyli $El \circ Set = id_{\mathbf{P}(A)}$.

Paradoks Cantora

Twierdzenie: Nie istnieją takie funkcje $set : A \rightarrow \mathbf{P}(A)$,
oraz $el : \mathbf{P}(A) \rightarrow A$, że $set \circ el = id_{\mathbf{P}(A)}$ oraz $el \circ set = id_A$.

Uwaga: Gdyby tak było, to niech

$$El : \mathbf{P}(\mathbf{P}(A)) \rightarrow \mathbf{P}(A) \qquad Set : \mathbf{P}(A) \rightarrow \mathbf{P}(\mathbf{P}(A))$$

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

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

$$\text{Czyli } El \circ Set = id_{\mathbf{P}(A)}.$$

$$\text{W szczególności } El \circ Set = set \circ el.$$

Paradoks Girarda-Hurkensa

Twierdzenie: Nie istnieją takie funkcje $set : A \rightarrow \mathbf{P}(A)$,
oraz $el : \mathbf{P}(A) \rightarrow A$, 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: $set : A \rightarrow \mathbf{P}(A)$, $el : \mathbf{P}(A) \rightarrow A$, $set \circ el = El \circ Set$.

Definicje:

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

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

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

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

Definicje:

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

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

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 $a \approx x \in set(b)$.

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

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 $c \in set(b)$.*

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

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:

$a' = el(set(a)) \in El(Set(set(b))) = set(el(set(b))) = set(b')$.

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

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

- ▶ 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 $a \varepsilon b \Leftrightarrow a \varepsilon b'$.*

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

- ▶ 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 $y \in set(b)$).

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

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

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

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

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

Niech $\Delta = \{a \in A \mid a \not\approx a\}$ oraz $\delta = el(\Delta)$.

Lemat: Dla $a \in A$ zachodzi $a \varepsilon \delta \Leftrightarrow a \not\approx a$.

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

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

Niech $\Delta = \{a \in A \mid a \not\approx a\}$ oraz $\delta = el(\Delta)$.

Lemat: Dla $a \in A$ zachodzi $a \varepsilon \delta \Leftrightarrow a \not\approx a$.

Dowód: (\Leftarrow) Jeśli $a \not\approx a$, to $a \in \Delta$. Wtedy:

$a \approx a' = el(set(a)) \in El(Set(\Delta)) = set(el(\Delta)) = set(\delta)$.

Czyli właśnie $a \varepsilon \delta$.

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

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

Niech $\Delta = \{a \in A \mid a \not\approx a\}$ oraz $\delta = el(\Delta)$.

Lemat: Dla $a \in A$ zachodzi $a \varepsilon \delta \Leftrightarrow a \not\approx a$.

Dowód: (\Leftarrow) Jeśli $a \not\approx a$, 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 $d \not\approx d$ oraz $a \approx d$, to $a \not\approx a$.

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

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

Niech $\Delta = \{a \in A \mid a \not\approx a\}$ oraz $\delta = el(\Delta)$.

Lemat: Dla $a \in A$ zachodzi $a \varepsilon \delta \Leftrightarrow a \not\approx a$.

Dowód: (\Leftarrow) Jeśli $a \not\approx a$, 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 $d \not\approx d$ oraz $a \approx d$, to $a \not\approx a$.

Morał: Sprzeczność, bo $\delta \varepsilon \delta \Leftrightarrow \delta \not\approx \delta$.

Paradoks Girarda-Hurkensa

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

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

Formalizacja

- ▶ Jeśli zbiór reprezentujemy w CC jako typ τ , to potęga $\mathbf{P}(\tau) = (\tau \Rightarrow *)$ jest rodzajem.

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.

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:

$$\alpha =^{\kappa} \beta \text{ oznacza } \forall \gamma^{\kappa \rightarrow *}. \gamma \alpha \rightarrow \gamma \beta.$$

Formalizacja

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

$$\alpha =^{\kappa} \beta \text{ oznacza } \forall \gamma^{\kappa \rightarrow *}. \gamma \alpha \rightarrow \gamma \beta.$$

- ▶ Jeśli $F : \kappa \rightarrow \varrho$, to $Lift F : \mathbf{P}(\kappa) \rightarrow \mathbf{P}(\varrho)$, 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 .

Sprzeczne otoczenie typowe:

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

$$\kappa : \square, \quad el : \mathbf{P}(\kappa) \rightarrow \kappa, \quad set : \kappa \rightarrow \mathbf{P}(\kappa),$$

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

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

Trzeba teraz zdefiniować odpowiedni rodzaj κ .

System sprzeczny

- ▶ System λU : rozszerzenie CC o sort Δ , aksjomat $\square : \Delta$ i reguły (Δ, \square) i $(\Delta, *)$.
- ▶ System λU^- : bez tej drugiej reguły.
- ▶ To jest „polimorfizm rodzajowy”: można definiować polimorficzne konstruktory, np.:

$$\lambda \kappa^{\square} \lambda \alpha^{\kappa \rightarrow \kappa} \lambda \beta^{\kappa} . \alpha(\alpha \beta) : \prod \kappa : \square ((\kappa \rightarrow \kappa) \rightarrow \kappa \rightarrow \kappa).$$

Rodzaj paradoksalny

$$\kappa = \mu k \square \mathbf{P}(k) = \forall k \square ((\mathbf{P}(k) \rightarrow k) \rightarrow k).$$

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

$$\text{Elim} = \lambda k \square \lambda f^{\mathbf{P}(k) \rightarrow k} \lambda z^{\kappa}. z k f$$

$$el = in = \lambda X^{\mathbf{P}(\kappa)} \wedge k \square \lambda y^{\mathbf{P}(k) \rightarrow k}. y(\text{Lift}(\text{Elim} k y) X),$$

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

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

$$\kappa = \mu k^{\square} \mathbf{P}(k) = \forall k^{\square} ((\mathbf{P}(k) \rightarrow k) \rightarrow k)$$

$$\text{Elim} = \lambda k^{\square} \lambda f^{\mathbf{P}(k) \rightarrow k} \lambda z^{\kappa}. z k f$$

$$el = in = \lambda X^{\mathbf{P}(\kappa)} \wedge k^{\square} \lambda y^{\mathbf{P}(k) \rightarrow k}. y(\text{Lift}(\text{Elim } k y) X),$$

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

Liczymy:

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

$$\kappa = \mu k^{\square} \mathbf{P}(k) = \forall k^{\square} ((\mathbf{P}(k) \rightarrow k) \rightarrow k)$$

$$\text{Elim} = \lambda k^{\square} \lambda f^{\mathbf{P}(k) \rightarrow k} \lambda z^{\kappa}. z k f$$

$$el = in = \lambda X^{\mathbf{P}(\kappa)} \wedge k^{\square} \lambda y^{\mathbf{P}(k) \rightarrow k}. y(\text{Lift}(\text{Elim } k y) X),$$

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

Liczymy:

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

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

$$\kappa = \mu k^{\square} \mathbf{P}(k) = \forall k^{\square} ((\mathbf{P}(k) \rightarrow k) \rightarrow k)$$

$$\text{Elim} = \lambda k^{\square} \lambda f^{\mathbf{P}(k) \rightarrow k} \lambda z^{\kappa}. z k f$$

$$el = in = \lambda X^{\mathbf{P}(\kappa)} \wedge k^{\square} \lambda y^{\mathbf{P}(k) \rightarrow k}. y(\text{Lift}(\text{Elim } k y) X),$$

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

Liczymy:

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

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

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

$$\kappa = \mu k^{\square} \mathbf{P}(k) = \forall k^{\square} ((\mathbf{P}(k) \rightarrow k) \rightarrow k)$$

$$\text{Elim} = \lambda k^{\square} \lambda f^{\mathbf{P}(k) \rightarrow k} \lambda z^{\kappa}. z k f$$

$$el = in = \lambda X^{\mathbf{P}(\kappa)} \wedge k^{\square} \lambda y^{\mathbf{P}(k) \rightarrow k}. y(\text{Lift}(\text{Elim } k y) X),$$

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

Liczymy:

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

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

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

$$(\text{Lift}(in)(\text{Lift}[out](X))) =$$

$$\kappa = \mu k \square \mathbf{P}(k) = \forall k \square ((\mathbf{P}(k) \rightarrow k) \rightarrow k)$$

$$\text{Elim} = \lambda k \square \lambda f^{\mathbf{P}(k) \rightarrow k} \lambda z^{\kappa}. z k f$$

$$el = in = \lambda X^{\mathbf{P}(\kappa)} \wedge k \square \lambda y^{\mathbf{P}(k) \rightarrow k}. y(\text{Lift}(\text{Elim } k y) X),$$

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

Liczmy:

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

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

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

$$(\text{Lift}(in)(\text{Lift}[\text{out}](X))) =$$

$$El(\text{Set}(X)).$$

Inne rozwiązanie

$$\kappa = \forall k^{\square} (\forall l^{\square} ((l \rightarrow k) \rightarrow \mathbf{P}(l) \rightarrow k) \rightarrow k).$$

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

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

Inne rozwiązanie: szczegóły dla dociekliwych

$$\begin{aligned}el &= \lambda X^{\mathbf{P}(\kappa)} \lambda k \square \lambda y^{\forall \iota: \square. (\iota \rightarrow k) \rightarrow \mathbf{P}(\iota) \rightarrow k}. y \kappa (\lambda \beta^\kappa. \beta k y) X; \\set &= \lambda \beta^\kappa. \beta(\mathbf{P}(\kappa)) \psi,\end{aligned}$$

gdzie

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

Rachunki:

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

Type is not a type

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

Zakręcone kombinatory (?)

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

$$YF \rightarrow F(YF)$$

Zakręcone kombinatory (?)

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

$$YF \rightarrow F(YF)$$

Wiadomo, że są tzw. looping combinators:

$$Y_0F \rightarrow F(Y_1F) \rightarrow F^2(Y_2F) \rightarrow F^3(Y_3F) \rightarrow \dots$$

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 $\pi_1 : \exists \alpha : *. \sigma \rightarrow *$ o własności $\pi_1[\tau, M] = \tau$;
- ▶ Rodzaje indukcyjne nie-ściśle pozytywne, np. rodzaj κ z konstruktorem typu $((\kappa \rightarrow *) \rightarrow *) \rightarrow \kappa$, czyli typu $\mathbf{P}(\mathbf{P}(\kappa)) \rightarrow \kappa$.

„Silna suma”

Do sprzeczności prowadzi konstrukcja

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

wraz z włożeniem $tam : * \rightarrow B$,

i rzutowaniem $nazad : B \rightarrow *$,

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

Zależny iloczyn kartezyjański 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)).  
  = R  
  : (fun _ : Be => Prop) (beintro R tt)
```

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

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
```

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

```
Coq < Ce is defined  
Ce_ind is defined
```


