Logika i teoria typów

Wykład 10

21 grudnia 2020

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.

Idea dowodu: Silna normalizacja implikuje całkowitość funkcji uniwersalnej $f(n,m) = f_n(m)$, gdzie f_n to wszystkie funkcje definiowalne w **F**. Ale funkcja f nie jest definiowalna, bo jeśli $f(n,n)+1 = f_k(n)$, to $f(k,k)+1 = f(k,k)$.

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.

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

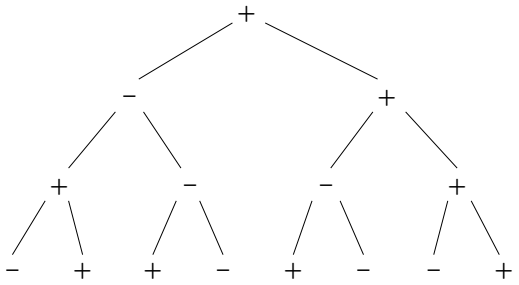
$$\alpha = \mu p.\, q \to p \quad \text{with} \quad q \to \alpha.$$

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

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

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 $\tau = \tau \to \ldots$)
  - **22K** typable.
    (Take $\alpha = \mu p.\, q \to p$. Then $\mathbf{K} : \alpha \to \alpha$.)

Positive recursive types

- Decidable inhabitation
  (reduction to simple types).
- Decidable type assignment
  (unification with infinite regular trees).
- Uwaga: ostrożnie z drzewami: typy
  $\mu p.\, p \to q$ i $\mu p.\, (p \to q) \to q$
  rozwijają się w to samo drzewo.

Typy rekurencyjne „w sensie Churcha"?

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

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

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

Jaki eliminator jest odpowiedni?

Pierwszy pomysł:

Oznaczenie $\mu = \mu p.\, \sigma(p)$. Mamy $in : \sigma(\mu) \to \mu$

Wprowadzamy operator

$$out : \mu \to \sigma(\mu)$$

i redukcję $out(in\, M) \Rightarrow M$.

To jest umiarkowanie ciekawe. (Niewiele można zdefiniować.)

Jaki eliminator jest odpowiedni?

Per Martin-Löf: Reguły wprowadzania dla spójnika logicznego (typu danych) są pierwotne. Reguła eliminacji i zasada beta-redukcji są wtórne.

Reguły wprowadzania określają jakie są kanoniczne formy elementów danego typu. Na przykład, dla alternatywy czyli koproduktu mamy dwie postaci kanoniczne:

$$\frac{\Gamma \vdash M : \alpha}{\Gamma \vdash \mathrm{inl}(M) : \alpha \vee \beta} \qquad \frac{\Gamma \vdash M : \beta}{\Gamma \vdash \mathrm{inr}(M) : \alpha \vee \beta}$$

Alternatywa/koprodukt

Dwie postaci kanoniczne = dwie reguły wprowadzania:

$$\frac{\Gamma \vdash M : \alpha}{\Gamma \vdash \mathrm{inl}(M) : \alpha \vee \beta} \qquad \frac{\Gamma \vdash M : \beta}{\Gamma \vdash \mathrm{inr}(M) : \alpha \vee \beta}$$

Reguła eliminacji podaje sposób użycia elementów kanonicznych. Ma po jednej przesłance niegłównej dla każdej postaci kanonicznej. Dla alternatywy dwie.

$$\frac{\Gamma \vdash M : \alpha \vee \beta \quad \Gamma, u{:}\alpha \vdash R : \tau \quad \Gamma, v{:}\beta \vdash Q : \tau}{\Gamma \vdash \mathrm{case}\ M\ \mathrm{of}\ [u]R\ \mathrm{or}\ [v]Q : \tau}$$

Alternatywa/koprodukt

$$\frac{\Gamma \vdash M : \alpha}{\Gamma \vdash \mathrm{inl}(M) : \alpha \vee \beta} \qquad \frac{\Gamma \vdash M : \beta}{\Gamma \vdash \mathrm{inr}(M) : \alpha \vee \beta}$$

$$\frac{\Gamma \vdash M : \alpha \vee \beta \quad \Gamma, u{:}\alpha \vdash R : \tau \quad \Gamma, v{:}\beta \vdash Q : \tau}{\Gamma \vdash \mathrm{case}\ M\ \mathrm{of}\ [u]R\ \mathrm{or}\ [v]Q : \tau}$$

Każda przesłanka niegłówna to sposób użycia innej postaci kanonicznej. To determinuje też zasadę beta-redukcji:

$\mathrm{case\ inl}\,(P^\alpha)\ \mathrm{of}\ [x^\alpha]M^\rho, [y^\beta]N^\rho \implies M[x := P] : \rho$
$\mathrm{case\ inr}\,(Q^\beta)\ \mathrm{of}\ [x^\alpha]M^\rho, [y^\beta]N^\rho \implies N[y := Q] : \rho$

Alternatywa/koprodukt

$$\frac{\Gamma \vdash M : \alpha}{\Gamma \vdash \mathrm{inl}(M) : \alpha \oplus \beta} \qquad \frac{\Gamma \vdash M : \beta}{\Gamma \vdash \mathrm{inr}(M) : \alpha \oplus \beta}$$

$$\frac{\Gamma \vdash M : \alpha \oplus \beta \quad \Gamma, u{:}\alpha \vdash R : \tau \quad \Gamma, v{:}\beta \vdash Q : \tau}{\Gamma \vdash \mathrm{case}\ M\ \mathrm{of}\ [u]R\ \mathrm{or}\ [v]Q : \tau}$$

Każda przesłanka niegłówna to sposób użycia innej postaci kanonicznej. To determinuje też zasadę beta-redukcji:

$\mathrm{case\ inl}\,(P^\alpha)\ \mathrm{of}\ [x^\alpha]M^\rho, [y^\beta]N^\rho \implies M[x := P] : \rho$
$\mathrm{case\ inr}\,(Q^\beta)\ \mathrm{of}\ [x^\alpha]M^\rho, [y^\beta]N^\rho \implies N[y := Q] : \rho$

To są niewłaściwe eliminacje

Dla koniunkcji dostaniemy coś takiego:

$$\frac{\Gamma \vdash M : A \wedge B \qquad \Gamma, x : A, y : B \vdash N : C}{\Gamma \vdash \mathrm{let}\ M\ \mathrm{be}\ \langle x, y\rangle\ \mathrm{in}\ N : C}$$

Beta-redukcja:

$$\mathrm{let}\ \langle P, Q\rangle\ \mathrm{be}\ \langle x, y\rangle\ \mathrm{in}\ N \Rightarrow N[x := P][y := Q]$$

Ćwiczenie: jaka koniunkcja jest w Coqu?

Studium przypadku: liczby naturalne

Typ $\mathbf{1}$ ma jeden element $\bullet : \mathbf{1}$.

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

czyli

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

czyli

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

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

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

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

```
Inductive nat : Set :=  0 : nat | S : nat -> nat.
```

Przykład: liczby naturalne

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

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

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

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

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

Trochę inaczej to samo

Zamiast jednej reguły wprowadzania $\dfrac{\Gamma \vdash M : \mathbf{1} \oplus \mathbf{int}}{\Gamma \vdash in(M) : \mathbf{int}}$

dwie reguły dla dwóch przypadków:

$$\frac{\Gamma \vdash \bullet : \mathbf{1}}{\Gamma \vdash in(\mathsf{inl}(\bullet)) : \mathbf{int}} \qquad \frac{\Gamma \vdash M : \mathbf{int}}{\Gamma \vdash in(\mathsf{inr}(M)) : \mathbf{int}}$$

Albo trochę przyjaźniej:

$$\Gamma \vdash 0 : \mathbf{int} \qquad \frac{\Gamma \vdash M : \mathbf{int}}{\Gamma \vdash \mathsf{s}(M) : \mathbf{int}}$$

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

Jaki eliminator jest odpowiedni dla liczb naturalnych?

Reguła wprowadzania $\dfrac{\Gamma \vdash M : \mathbf{1} \oplus \mathbf{int}}{\Gamma \vdash in(M) : \mathbf{int}}$

sugeruje taką (naiwną) eliminację:

$$\frac{\Gamma \vdash M : \mathbf{int} \quad \Gamma, x : \mathbf{1} \oplus \mathbf{int} \vdash R : \tau}{\Gamma \vdash \mathsf{Elim}\ M\ \mathsf{with}\ [x]R : \tau}$$

Naiwna eliminacja

$$\frac{\Gamma \vdash M : \mathbf{int} \quad \Gamma, x : \mathbf{1} \oplus \mathbf{int} \vdash R : \tau}{\Gamma \vdash \mathsf{Elim}\ M\ \mathsf{with}\ [x]R : \tau}$$

Naiwna eliminacja rozbita na przypadki:

$$\frac{\Gamma \vdash M : \mathbf{int} \quad \Gamma \vdash P : \tau \quad \Gamma, v{:}\mathbf{int} \vdash Q : \tau}{\Gamma \vdash \mathsf{Elim}\ M\ \mathsf{with}\ P\ \mathsf{or}\ [v]Q : \tau}$$

Z tego wychodzi taka beta-redukcja:

$\mathsf{Elim}\ 0\ \mathsf{with}\ P\ \mathsf{or}\ [v]Q \Rightarrow P$

$\mathsf{Elim}\ s(n)\ \mathsf{with}\ P\ \mathsf{or}\ [v]Q \Rightarrow Q[v := n]$

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

$$\frac{\Gamma \vdash M : \mathbf{int} \quad \Gamma \vdash P : \tau \quad \Gamma, v{:}\mathbf{int} \vdash Q : \tau}{\Gamma \vdash \mathsf{Elim}\ M\ \mathsf{with}\ P\ \mathsf{or}\ [v]Q : \tau}$$

$$\mathsf{Elim}\ s(n)\ \mathsf{with}\ P\ \mathsf{or}\ [v]Q \Rightarrow Q[v := n]$$

Sposób użycia postaci kanonicznej jest zdeterminowany przez sposób użycia jej „składowych" (cokolwiek to znaczy). Liczba n jest składową swojego następnika $s(n)$. Zatem sposób użycia $s(n)$ powinien zależeć od sposobu użycia n.

Ale jeśli $n = s(k)$, to sposób użycia n powinien zależeć od sposobu użycia k. Naiwna reguła na to nie pozwala.

Jaka eliminacja dla $\mathbf{int} = \mu p.\, \mathbf{1} \oplus p$?

Do przesłanek reguły

$$\frac{\Gamma \vdash M : \mathbf{int} \quad \Gamma \vdash P : \tau \quad \Gamma, v{:}\mathbf{int} \vdash Q : \tau}{\Gamma \vdash \mathsf{Elim}\ M\ \mathsf{with}\ P\ \mathsf{or}\ [v]Q : \tau}$$

trzeba dodać sposób eliminacji $V : \tau$ dla $v : \mathbf{int}$:

$$\frac{\Gamma \vdash M : \mathbf{int} \quad \Gamma \vdash P : \tau \quad \Gamma, v{:}\mathbf{int}, V : \tau \vdash Q : \tau}{\Gamma \vdash \mathsf{Elim}\ M\ \mathsf{with}\ P\ \mathsf{or}\ [v, V]Q : \tau}$$

Sposób użycia $s(n)$ określamy znając sposób użycia V dla n:

$$\mathsf{Elim}\ s(n)\ \mathsf{with}\ P\ \mathsf{or}\ [v, V]Q \Rightarrow Q[v := n, V :=?]$$

Jaka eliminacja dla $\mathbf{int} = \mu p.\, \mathbf{1} \oplus p$?

$$\frac{\Gamma \vdash M : \mathbf{int} \quad \Gamma \vdash P : \tau \quad \Gamma, v{:}\mathbf{int}, V : \tau \vdash Q : \tau}{\Gamma \vdash \text{Elim } M \text{ with } P \text{ or } [v, V]Q : \tau}$$

Sposób użycia $s(n)$ określamy znając sposób użycia V dla n:

$$\text{Elim } s(n) \text{ with } P \text{ or } [v, V]Q \Rightarrow Q[v := n, V :=?]$$

$$\text{Elim } s(n) \text{ with } P \text{ or } [v, V]Q \Rightarrow Q[v := n, V := \text{Elim } n \text{ with } P \text{ or } [v, V]Q]$$

Prostsza wersja rekursji: iteracja

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

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

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

Jak to można uogólnić?

Reguła dla iteratora:

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

przepisana z użyciem koproduktu:

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

Redukcja:

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

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

To define $F(4) = D(\text{inr } F(3))$ we use the defined part of F, applying it at the “right channel”:

$$\mathbf{int} \approx 1 \oplus \mathbf{int} \qquad 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus \cdots$$

$$\downarrow_{\mathsf{id}} \quad \downarrow_{\mathsf{F}} \qquad \downarrow_{\mathsf{id}} \quad \downarrow_{\mathsf{F}}$$

$$1 \oplus \tau \qquad 1 \oplus \tau$$

$$\downarrow_{\mathsf{D}} \qquad \downarrow_{\mathsf{D}}$$

$$\tau \qquad \tau$$

$F : \mathbf{int} \to \tau$ is “lifted” to $F^+ : 1 \oplus \mathbf{int} \to 1 \oplus \tau$

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

$$F \circ in = D \circ F^+$$

User-friendly variant

$$\frac{\Gamma \vdash M : \mathbf{int} \quad \Gamma \vdash P : \tau \quad \Gamma, v{:}\mathbf{int}, V : \tau \vdash Q : \tau}{\Gamma \vdash \text{Elim } M \text{ with } P \text{ or } [v, V]Q : \tau}$$

Instead of $\Gamma, v{:}\mathbf{int}, V : \tau \vdash Q : \tau$ and Elim take $\Gamma \vdash Q : \mathbf{int} \to \tau \to \tau$ and *recursor* R_τ:

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

Ciekawostka historyczna

Typ rekursora

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

jest wytarciem schematu indukcji Peana w wersji z 1889 roku:

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

Typ iteratora

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

jest wytarciem schematu indukcji Peana w wersji z 1891 roku:

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

What is going on here?

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

Given $D : \mathbf{1} \oplus \tau \to \tau$, we define by induction a function $F = \text{Elim}_\tau\, D : \mathbf{int} \to \tau$:

$$F(in(\text{inl}(\bullet))) = D(\text{inl}(\bullet)) \qquad F(in(\text{inr}(n))) = D(\text{inr}(F(n)))$$

We imagine that $\mathbf{int} = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus \cdots$

We want to define $F(4)$ (number 4 is the red component) when F is defined for the first four components. Then:

$$F(4) = F(in(\text{inr } 3)) = D(\text{inr } F(3))$$

Podsumowanie

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

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

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

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

Inny przykład: długość listy

$\mathbf{lista} \approx \mathbf{1} \oplus (\mathbf{int} \times \mathbf{lista})$, czyli:
$\mathbf{lista} \approx T(\mathbf{lista})$, gdzie $T(\alpha) = \mathbf{1} \oplus (\mathbf{int} \times \alpha)$

$$suma : \mathbf{lista} \to \mathbf{int}$$
$$suma(nil) = 0, \qquad suma(n :: \ell) = n + suma(\ell)$$

Mamy tu operację $D : T(\mathbf{int}) \to \mathbf{int}$:
$$D(\mathrm{inl}(\bullet)) = 0, \qquad D(\mathrm{inr}(\langle p, d\rangle)) = p + d.$$

Wtedy $suma(nil) = D(suma^+(\mathrm{inl}(\bullet)))$
oraz $suma(n :: \ell) = D(suma^+(\mathrm{inr}(\langle n, \ell\rangle)))$,

gdzie $suma^+ : T(\mathbf{lista}) \to T(\mathbf{int})$, to „podniesiona" $suma$:

$suma^+(\mathrm{inl}(\bullet)) = \mathrm{inl}(\bullet)$,
$suma^+(\mathrm{inr}(\langle n, \ell\rangle)) = \mathrm{inr}(\langle n, suma(\ell)\rangle)$

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

Powrót do pierwszego pomysłu?

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

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 $out \circ in \neq \mathrm{id}$:

$$out(in\ N) \to Lift(in)(Lift(out)(N))$$
$$out \circ in = In \circ Out$$

A generalization

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

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

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

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

Reduction rule:

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

$$\mathrm{Elim}_\tau\ D\ (in\ N) \Rightarrow D(Lift_\sigma(\mathrm{Elim}_\tau\ D)N)$$
$$Lift_\sigma : (\mu \to \tau) \to \sigma(\mu) \to \sigma(\tau).$$

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 $a = \inf B$. Pokażemy, że a jest najmniejszym punktem stałym funkcji f.

Dla dowolnego $x \in B$ mamy $a \leq x$, więc $f(a) \leq f(x) \leq x$. Zatem $f(a)$ jest ograniczeniem dolnym zbioru B, skąd $f(a) \leq a$, bo a jest kresem dolnym.

Ale skoro $f(a) \leq a$, to także $f(f(a)) \leq f(a)$, więc $f(a) \in B$. Zatem $a \leq f(a)$ i mamy równość.

Ponieważ wszystkie punkty stałe funkcji f muszą należeć do B, więc a jest najmniejszym punktem stałym. □

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 $\mu r.\, \sigma(r)$.

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

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

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

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

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

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

Problem otwarty

Ta konstrukcja nie przenosi się na (uogólnione) rekursory.

Wiadomo, że w systemie **F** nie można definiować rekursorów używając beta-redukcji.

Nie wiadomo, czy w systemie **F** można zdefiniować dowolny rekursor używając beta-eta-redukcji.

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

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

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

Część trudna: w modelu teoriomnogościowym:

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

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

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

Zatem nie istnieje model teoriomnogościowy.

Wada kodowania impredykatywnego

Typy kodowane w systemie **F** mają za dużo elementów.

Najprostszy przykład: $\alpha \times \beta = \forall p((\alpha \to \beta \to p) \to p)$.

Niech $\alpha = \forall r(r \to \forall p(p \to r) \to r) \quad \beta = \forall r(r \to r)$.

Wtedy $\mathbf{K} : \alpha$ oraz $\mathbf{I} : \beta$.

Dla $x : \alpha \to \beta \to p$ wyrażenie $x\mathbf{KI}$ ma typ p.

Zatem taki inhabitant typu $\alpha \times \beta$

$$\Lambda p \lambda x^{\alpha \to \beta \to p}.\, x(\Lambda r \lambda y^{r} \lambda z^{\forall p(p \to r)}.\, zp(x\mathbf{KI}))\mathbf{I}$$

nie jest parą $\langle M, N\rangle = \Lambda p \lambda x M N$, bo $x \in FV(M)$

Wada kodowania impredykatywnego

Nie można udowodnić zasady indukcji, nawet dla typu bool.

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:

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

Liczby Parigota:

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

Przykład 2: ten typ jest ściśle pozytywny:

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

Zalety ścisłej pozytywności

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

Typy indukcyjne

Typowa sytuacja: typ rekurencyjny jest zadany przez kilka konstruktorów (kilka schematów postaci kanonicznych), np:

- Zero $0 : \mathbf{int}$ i następnik $s : \mathbf{int} \to \mathbf{int}$,
  dla $\mathbf{int} \approx \mathbf{1} \oplus \mathbf{int}$;
- $nil : list$ i $cons : \mathbf{int} \times list \to list$,
  dla $list \approx \mathbf{1} \oplus \mathbf{int} \times list$;
- $leaf : \omega\text{-}tree$ oraz $node : (\mathbf{int} \to \omega\text{-}tree) \to \omega\text{-}tree$,
  dla $\omega\text{-}tree \approx \mathbf{1} \oplus (\mathbf{int} \to \omega\text{-}tree)$.

Przykłady

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

Inductive list (A : Type) : Type :=
    nil : list A | cons : A -> list A -> list A.
```

Typy indukcyjne nie muszą być rekurencyjne:

```
Inductive bool : Set :=  true : bool | false : bool.

Inductive unit : Set :=  tt : unit.

Inductive empty_set : Set := .
```

Spójniki zdaniowe:

```
Inductive and (A B : Prop) : Prop :=
    conj : A -> B -> A /\ B.
Inductive or (A B : Prop) : Prop :=
    or_introl : A -> A \/ B | or_intror : B -> A \/ B.
Inductive False : Prop := .
```

Kwantyfikator szczegółowy:

```
Inductive ex (A : Type) (P : A -> Prop) : Prop :=
    ex_intro : forall x : A, P x -> ex P.
```

Piszemy „exists x:A,Px" zamiast „ex(fun x:A => P x)".

Predykaty indukcyjne

```
Inductive even : nat -> Prop :=
 zeroparz : even 0 |
 nastparz : forall n : nat, even n -> even (n + 2)
```

```
Inductive eq (A : Type) (x : A) : A -> Prop :=
    eq_refl : x = x
```

Przyjęta notacja: $x =_A y$.

Proof by induction

```
Coq < Goal forall n : nat, even (n+n).
1 subgoal

  ============================
  forall n : nat, even (n + n)

Unnamed_thm < induction n.
2 subgoals

  ============================
  even (0 + 0)

subgoal 2 is:
 even (S n + S n)
```