Seminarium: Programowanie w teorii typów

Teoria typów cd.

Wojciech Jedynak, Paweł Wieczorek

Instytut Informatyki Uniwersytetu Wrocławskiego

12 października 2011

Plan na dziś

- Co już umiemy:
  - posługiwać się systemem naturalnej dedukcji
  - przerbnęliśmy przez regułki dla Π-typów oraz Σ-typów
  - zobaczyliśmy jak typy kodują kwantyfikatory z logiki pierwszego rzędu
- Co dziś zobaczymy:
  - typy wyliczeniowe
  - równość w teorii typów
  - W-typy
  - uniwersa

Typy wyliczeniowe

$$\frac{}{N_k \; set} \qquad \frac{\text{dla } \; m < k}{m_k \in N_k}$$

$$\frac{a \in N_k \qquad C(v) \; set \; [v \in N_k] \quad c_0 \in C(0_k) \quad \vdots \quad c_{k-1} \in C((k-1)_k)}{case_k(a, c_0, \cdots, c_{k-1}) \in C(a)}$$

$$\frac{C(v) \; set \; [v \in N_k] \quad c_0 \in C(0_k) \quad \vdots \quad c_{k-1} \in C((k-1)_k)}{case_k(m_k, c_0, \cdots, c_{k-1}) = c_m \in C(m_k)}$$

Typy wyliczeniowe

$$
\begin{aligned}
N_0 &= \bot \\
\neg A &= A \to N_0
\end{aligned}
$$

Typy wyliczeniowe

$$
\begin{aligned}
N_0 &= \bot \\
\neg A &= A \to N_0 \\
N_1 &= \top \\
N_1 &= \mathit{Unit} \\
0_1 &= tt
\end{aligned}
$$

Typy wyliczeniowe

$$
\begin{array}{rcl}
N_0 & = & \bot \\
\neg A & = & A \to N_0 \\
N_1 & = & \top \\
N_1 & = & \mathit{Unit} \\
0_1 & = & \mathit{tt} \\
N_2 & = & \mathit{Bool} \\
0_2 & = & \mathit{true} \\
1_2 & = & \mathit{false} \\
\mathit{case}_2(a, c_0, c_1) & = & \mathit{if}\ \ a\ \mathit{then}\ c_0\ \mathit{else}\ c_1
\end{array}
$$

Typy wyliczeniowe

$$
\begin{array}{rcl}
N_0 & = & \bot \\
\neg A & = & A \to N_0 \\
N_1 & = & \top \\
N_1 & = & \mathit{Unit} \\
0_1 & = & \mathit{tt} \\
N_2 & = & \mathit{Bool} \\
0_2 & = & \mathit{true} \\
1_2 & = & \mathit{false} \\
\mathit{case}_2(a, c_0, c_1) & = & \mathit{if}\ \ a\ \mathit{then}\ c_0\ \mathit{else}\ c_1
\end{array}
$$

$$
\frac{a \in N_0 \qquad C(v)\ \mathit{set}\ [v \in N_0]}{\mathit{case}_0(a) \in C(a)} \qquad \frac{\bot\ \mathit{true} \qquad C\ \mathit{prop}}{C\ \ \mathit{true}}
$$

Wyrażenia

- język wyrażeń:
  - $e(e')$ - aplikacja ($e(e_1, \cdots, e_n)$ skrócony zapis $e(e_1) \cdots (e_n)$)
  - $(x)e$ - abstrakcja
  - stałe, np λ, Π, $apply, 0, succ$

Wyrażenia

- język wyrażeń:
  - $e(e')$ - aplikacja ($e(e_1, \cdots, e_n)$ skrócony zapis $e(e_1)\cdots(e_n)$)
  - $(x)e$ - abstrakcja
  - stałe, np λ, Π, $apply, 0, succ$

- posługujem się prostym systemem typów (z jednym typem bazowym O) zwanym „arnościami"
- $e(e') : \beta$ gdy $e : \alpha \to \beta$, $e' : \alpha$
- konstruktory:
  - $\lambda : (O \to O) \to O$
  - $\Pi : O \to (O \to O) \to O$
  - $apply : O \to O \to O$
- otrzymujemy silną normalizację oraz rozstrzygalność dla równości definicyjnej

Typ identycznościowy

- Sądy a formuły to nie to samo
  - nie możemy powiedzieć wewnątrz systemu „jeżeli t ma typ A to ...”: $\Gamma \vdash (t \in A) \to \varphi$
  - nie możemy też powiedzieć wewnątrz systemu „dany sąd jest nieprawdziwy”: $\Gamma \vdash \neg(a = b \in A)$

Typ identycznościowy

- Sądy a formuły to nie to samo
  - nie możemy powiedzieć wewnątrz systemu „jeżeli t ma typ A to ...”: $\Gamma \vdash (t \in A) \to \varphi$
  - nie możemy też powiedzieć wewnątrz systemu „dany sąd jest nieprawdziwy”: $\Gamma \vdash \neg(a = b \in A)$
- Co ma oznaczać stwierdzenie, że elementy a oraz b są sobie równe?
  - nie poróżnia je żadna własność (równość Leibniza)
  - dla każdego predykatu P: $\forall x\ y.\ x = y \to P(x) \to P(y)$

Typ identycznościowy

- Sądy a formuły to nie to samo
  - nie możemy powiedzieć wewnątrz systemu „jeżeli t ma typ A to ...": $\Gamma \vdash (t \in A) \to \varphi$
  - nie możemy też powiedzieć wewnątrz systemu „dany sąd jest nieprawdziwy": $\Gamma \vdash \neg(a = b \in A)$
- Co ma oznaczać stwierdzenie, że elementy a oraz b są sobie równe?
  - nie poróżnia je żadna własność (równość Leibniza)
  - dla każdego predykatu P: $\forall x\ y.\ x = y \to P(x) \to P(y)$
- Co znaczy, że funkcje są sobie równe?
  - Ile jest funkcji (jako zbiór par) sortujących? Ekstensjonalnie tylko jedna.
  - $(\forall xs \in \mathit{List}.\ \mathit{qsort}(xs) = \mathit{isort}(xs)) \to \mathit{qsort} = \mathit{isort}$

Typ identycznościowy

- Sądy a formuły to nie to samo
  - nie możemy powiedzieć wewnątrz systemu „jeżeli t ma typ A to ...”: $\Gamma \vdash (t \in A) \to \varphi$
  - nie możemy też powiedzieć wewnątrz systemu „dany sąd jest nieprawdziwy”: $\Gamma \vdash \neg(a = b \in A)$
- Co ma oznaczać stwierdzenie, że elementy *a* oraz *b* są sobie równe?
  - nie poróżnia je żadna własność (równość Leibniza)
  - dla każdego predykatu P: $\forall x\ y.\ x = y \to P(x) \to P(y)$
- Co znaczy, że funkcje są sobie równe?
  - Ile jest funkcji (jako zbiór par) sortujących? Ekstensjonalnie tylko jedna.
  - $(\forall xs \in List.\ qsort(xs) = isort(xs)) \to qsort = isort$
- Równość jest formułą atomową, chcemy mieć dla niej typ.
  - który zgodnie z *formulas-as-types* jest zamieszkany wtedy i tylko wtedy, gdy elementy są sobie równe

Typ identycznościowy

- mamy:
  - równość definicyjna: $e \equiv e'$
  - równość elementów o tym samym typie $a = b \in A$
  - równosć typów $A = B$
- chcemy jeszcze:
  - formuła logiczna $[a =_A b]$

Typ identycznościowy

$$\frac{A\ set \qquad a \in A \qquad b \in A}{[a =_A b]\ set} \qquad \frac{a \in A}{id(a) \in [a =_A a]}$$

$$\frac{a = a' \in A \qquad b = b' \in A}{[a =_A b] = [a' =_A b']}$$

Typ identycznościowy

$$\frac{A\ set \qquad a \in A \qquad b \in A}{[a =_A b]\ set} \qquad \frac{a \in A}{id(a) \in [a =_A a]}$$

$$\frac{a = a' \in A \qquad b = b' \in A}{[a =_A b] = [a' =_A b']}$$

$$\frac{a = b \in A}{id(a) \in [a =_A b]}$$

Typ identycznościowy

$$\frac{A\ set \qquad a \in A \qquad b \in A}{[a =_A b]\ set} \qquad \frac{a \in A}{id(a) \in [a =_A a]}$$

$$\frac{a = a' \in A \qquad b = b' \in A}{[a =_A b] = [a' =_A b']}$$

$$\frac{a = b \in A}{id(a) \in [a =_A b]}$$

$$\frac{\dfrac{}{id(a) \in [a =_A a]} \qquad \dfrac{a = a \in A \quad a = b \in A}{[a =_A a] = [a =_A b]}}{id(a) \in [a =_A b]}$$

Typ identycznościowy

$$\frac{a \in A \qquad b \in A \qquad c \in [a =_A b]}{a = b \in A}$$

- silna eliminacja, równość ekstensjonalna
- chcemy by system nieodróżniał od siebie elementów o których możemy wydać sąd równościowy
- „gubimy dowód" $[a =_A b]$, a więc system chcąc sprawdzić $a = b \in A$ może być zmuszony by go odgadnąć
- nierozstrzygalność

Typ identycznościowy

$$\frac{a \in A \qquad b \in A \qquad c \in [a =_A b] \quad C(x,y,z)\ set\ [x \in A, y \in A, z \in [x =_A y]] \quad d(x) \in C(x,x,id(x))\ [x \in A]}{idpeel(c,d) \in C(a,b,c)}$$

$$\frac{a \in A \quad C(x,y,z)\ set\ [x \in A, y \in A, z \in [x =_A y]] \quad d(x) \in C(x,x,id(x))\ [x \in A]}{idpeel(id(a),d) = d(a) \in C(a,a,id(a))}$$

- słaba eliminacja, równość intensjonalna
- zasada indukcji jak dla innych typów
- zachowujemy rozstrzygalność dla równości definicyjnej wyrażeń oraz sądów równościowych

Typ identycznościowy

- niech $c \in [a =_A b]$.
- niech $C(x, y, z) \equiv [y =_A x]$

$$\frac{a \in A \qquad b \in A \qquad c \in [a =_A b] \\ C(x, y, z)\ set\ [x \in A, y \in A, z \in [x =_A y]] \\ id(x) \in C(x, x, id(x))\ [x \in A]}{idpeel(c, id) \in C(a, b, c)}$$

$$\frac{a \in A \qquad b \in A \qquad c \in [a =_A b] \\ [y =_A z]\ set\ [x \in A, y \in A, z \in [x =_A y]] \\ id(x) \in [x =_A x]\ [x \in A]}{idpeel(c, id) \in [b =_A a]}$$

- $symm(c) \equiv idpeel(c, id)$

Typ identycznościowy

- niech $e \in [a =_A b]$, $e' \in [b =_A c]$.
- niech $C(x, y, z) \equiv [y =_A c] \rightarrow [x =_A c]$

$$\frac{\begin{array}{c} a \in A \qquad b \in A \qquad e \in [a =_A b] \\ C(x, y, z) \; set \; [x \in A, y \in A, z \in [x =_A y]] \\ \lambda y.y \in C(x, x, id(x)) \; [x \in A] \end{array}}{idpeel(e, (x)\lambda y.y) \in C(a, b, c)}$$

$$\frac{\begin{array}{c} a \in A \qquad b \in A \qquad e \in [a =_A b] \\ [y =_A c] \rightarrow [x =_A c] \; set \; [x \in A, y \in A, z \in [x =_A y]] \\ \lambda y.y \in \; [x =_A c] \rightarrow [x =_A c][x \in A] \end{array}}{idpeel(e, (x)\lambda y.y) \in [b =_A c] \rightarrow [a =_A c]}$$

- $trans(e, e') \equiv apply(idpeel(d, (x)\lambda y.y), e')$

Typ identycznościowy

$$\frac{P(x)\ \mathit{prop}\ [x \in A] \qquad a \in A \qquad b \in A \qquad [a =_A b]\ \mathit{true} \qquad P(a)\ \mathit{true}}{P(b)\ \mathit{true}}$$

Typ identycznościowy

$$\frac{P(x)\ prop\ [x \in A] \qquad a \in A \qquad b \in A \qquad [a =_A b]\ true \qquad P(a)\ true}{P(b)\ true}$$

$$\frac{P(x)\ set\ [x \in A] \qquad a \in A \qquad b \in A \qquad c \in [a =_A b] \qquad p \in P(a)}{subst(c, p) \in P(b)}$$

- rowność Leibniza
- „kontrolowane” podstawienie

Typ identycznościowy

- niech $a \in A$, $b \in A$, $e \in [a =_A b]$
- niech $P(x)$ *set* $[x \in A]$, $p \in P(a)$
- niech $C(x, y, z) \equiv P(x) \to P(y)$

$$\frac{\begin{array}{c} a \in A \qquad b \in A \qquad e \in [a =_A b] \\ C(x, y, z)\ set\ [x \in A, y \in A, z \in [x =_A y]] \\ id(x) \in C(x, x, id(x))\ [x \in A] \end{array}}{idpeel(e, (x)\lambda y.y) \in C(a, b, c)}$$

$$\frac{\begin{array}{c} a \in A \qquad b \in A \qquad e \in [a =_A b] \\ P(x) \to P(y)\ set\ [x \in A, y \in A, z \in [x =_A y]] \\ \lambda y.y \in P(x) \to P(x)\ [x \in A] \end{array}}{idpeel(e, (x)\lambda y.y) \in P(a) \to P(b)}$$

- $subst(e, p) \equiv apply(idpeel(e, (x)\lambda y.y), p)$

Uniwersum

$$\frac{}{Set\ set} \qquad \frac{}{Nat \in Set}$$

$$\frac{A \in Set \qquad B \in Set}{A \to B \in Set} \qquad \frac{A \in Set \qquad B \in Set}{A \times B \in Set}$$

$$\cdots$$

$$\frac{}{Set \in Set}$$

Uniwersum ala Tarski

- posługujemy się nazwami(kodami, identyfikatorami, ...) typów, a nie typami bezpośrednio
- nazwy są zwykłymi danymi

Uniwersum ala Tarski

$$\frac{}{U \; set} \qquad \frac{}{\widehat{Nat} \in U}$$

$$\frac{}{\widehat{N_0} \in U} \qquad \frac{}{\widehat{N_1} \in U} \qquad \frac{}{\widehat{N_2} \in U}$$

$$\ldots$$

$$\frac{a \in U \qquad b \in U}{\widehat{a \to b} \in U} \qquad \frac{a \in U \qquad b \in U}{\widehat{a \times b} \in U}$$

Uniwersum ala Tarski

- Funkcja semantyczna *Set* bierze nazwę typu a zwraca typ

$$\frac{u \in U}{Set(u)\ set}$$

$$\begin{array}{rcl}
Set(\widehat{Nat}) & = & Nat \\
Set(\widehat{N_0}) & = & N_0 \\
Set(\widehat{N_1}) & = & N_1 \\
Set(\widehat{N_2}) & = & N_2 \\
& \ldots & \\
Set(\widehat{a \to b}) & = & Set(a) \to Set(b) \\
Set(\widehat{a \times b}) & = & Set(a) \times Set(b)
\end{array}$$

Uniwersum ala Tarski

- Co nam się nie podoba w tej funkcji?

$$\begin{array}{rcl} g & \in & \mathit{Nat} \rightarrow \mathit{Bool} \cup \mathit{Nat} \\ g(2n) & = & \mathit{left}(\mathit{true}) \\ g(2n+1) & = & \mathit{right}(42) \end{array}$$

Uniwersum ala Tarski

- Co nam się nie podoba w tej funkcji?

$$\begin{array}{rcl} g & \in & \mathit{Nat} \to \mathit{Bool} \cup \mathit{Nat} \\ g(2n) & = & \mathit{left}(\mathit{true}) \\ g(2n+1) & = & \mathit{right}(42) \end{array}$$

- Bezpieczniejsza wersja

$$\begin{array}{rcl} B_{2n} & = & \mathit{Bool} \\ B_{2n+1} & = & \mathit{Nat} \\ f & \in & \prod_{n \in N} B_n \\ f(2n) & = & \mathit{true} \\ f(2n+1) & = & 42 \end{array}$$

Uniwersum ala Tarski

$$\begin{aligned} B(x) &\equiv Set(case_2(apply(isEven, x), \widehat{Bool}, \widehat{Nat})) \\ b &\equiv (x)case_2(apply(isEven, x), true, 42) \end{aligned}$$

Uniwersum ala Tarski

$$
\begin{aligned}
B(x) &\equiv Set(case_2(apply(isEven, x), \widehat{Bool}, \widehat{Nat})) \\
b &\equiv (x)case_2(apply(isEven, x), true, 42) \\
\lambda x.b &\in (\Pi n \in Nat)B(n) \\
apply(\lambda x.b, n) &\in B(n)\ [n \in Nat]
\end{aligned}
$$

Uniwersum ala Tarski

$$
\begin{array}{rcl}
B(x) & \equiv & Set(case_2(apply(isEven, x), \widehat{Bool}, \widehat{Nat})) \\
b & \equiv & (x)case_2(apply(isEven, x), true, 42) \\
\lambda x.b & \in & (\Pi n \in Nat)B(n) \\
apply(\lambda x.b, n) & \in & B(n)\ [n \in Nat]
\end{array}
$$

- niech $negb \in Bool \rightarrow Bool$.

$$
\frac{\dfrac{}{negb \in Bool \rightarrow Bool} \quad \dfrac{apply(\lambda x.b, 2) \in B(2) \qquad B(2) = Bool}{apply(\lambda x.b, 2) \in Bool}}{apply(negb, apply(\lambda x.b, 2)) \in Bool}
$$

Uniwersum ala Tarski

- $nAry(A, n) = \underbrace{A \to A \to \cdots A}_{n+1}$

$$
\begin{array}{rcl}
nary & \equiv & (a, x)natrec(x, a, (n, y)\widehat{a \to y}) \\
\lambda x.nary(a, x) & \in & (\Pi x \in Nat)U \; [a \in U] \\
\lambda a.\lambda x.nary(a, x) & \in & (\Pi a \in U)(\Pi x \in Nat)U \\
nAry & \equiv & \lambda a.\lambda x.nary(a, x)
\end{array}
$$

Uniwersum ala Tarski

- $nAry(A, n) = \underbrace{A \to A \to \cdots A}_{n+1}$

$$
\begin{array}{rcl}
nary & \equiv & (a, x)natrec(x, a, (n, y)\widehat{a \to y}) \\
\lambda x.nary(a, x) & \in & (\Pi x \in Nat)U \; [a \in U] \\
\lambda a.\lambda x.nary(a, x) & \in & (\Pi a \in U)(\Pi x \in Nat)U \\
nAry & \equiv & \lambda a.\lambda x.nary(a, x)
\end{array}
$$

$$
\begin{array}{rcl}
boolfun & \equiv & (n)Set(apply(apply(nAry, \widehat{Bool}), n)) \\
\lambda n.boolfun & \in & (\Pi n \in Nat)Set
\end{array}
$$

- podobnie moglibyśmy generalizować proste schematy dla krotek A^2, A^3, itd:

```
fst (a,b) = a   fst3(a,b,c) = a   fst4(a,b,c,d) = a
```

Uniwersum ala Tarski

- Wyrażenie kodu dla $A \rightarrow B$ jest proste.

$$\frac{a \in U \qquad b \in U}{\widehat{a \rightarrow b} \in U} \qquad Set(\widehat{a \rightarrow b}) = Set(a) \rightarrow Set(b)$$

Uniwersum ala Tarski

- Wyrażenie kodu dla $A \rightarrow B$ jest proste.

$$\frac{a \in U \qquad b \in U}{\widehat{a \rightarrow b} \in U} \qquad Set(\widehat{a \rightarrow b}) = Set(a) \rightarrow Set(b)$$

- Co dla $(\Pi x \in A)B(x)$?

Uniwersum ala Tarski

- Wyrażenie kodu dla $A \to B$ jest proste.

$$\frac{a \in U \qquad b \in U}{\widehat{a \to b} \in U} \qquad Set(\widehat{a \to b}) = Set(a) \to Set(b)$$

- Co dla $(\Pi x \in A)B(x)$?

$$\frac{a \in U \qquad b \in Set(a) \to U}{\widehat{\Pi(a, b)} \in U}$$

Uniwersum ala Tarski

- Wyrażenie kodu dla $A \rightarrow B$ jest proste.

$$\frac{a \in U \qquad b \in U}{\widehat{a \rightarrow b} \in U} \qquad Set(\widehat{a \rightarrow b}) = Set(a) \rightarrow Set(b)$$

- Co dla $(\Pi x \in A)B(x)$?

$$\frac{a \in U \qquad b \in Set(a) \rightarrow U}{\widehat{\Pi(a, b)} \in U}$$

$$Set(\widehat{\Pi(a, b)}) = (\Pi x \in Set(a))\ Set(b(x))$$

Uniwersum ala Tarski

- Bez uniwersów nie umiemy udowodnić że konstruktory tworzą różne wartości!
- celujemy w sąd $\neg[0 =_{Nat} succ(n)]\ [n \in Nat]$
- niech $n \in Nat$, $e \in [0 =_{Nat} succ(n)]$

$$
\begin{array}{rcl}
IsZ(m) & \equiv & Set(natrec(m, \widehat{N_1}, (x, y)\widehat{N_0})) \\
IsZ(0) & \equiv & N_1 \\
IsZ(succ(n)) & \equiv & N_0
\end{array}
$$

$$
\frac{e \in [0 =_{Nat} succ(n)] \qquad 0_1 \in IsZ(0)}{subst(e, 0_1) \in IsZ(succ(n))}
$$

- $\lambda n.\lambda e.\ subst(e, 0_1) \in (\Pi n \in Nat)\neg[0 =_{Nat} succ(n)]$

Uniwersum ala Russel

$$\frac{}{U \; set} \qquad \frac{}{Nat \in U}$$

$$\frac{}{N_0 \in U} \qquad \frac{}{N_1 \in U} \qquad \frac{}{N_2 \in U}$$

$$\dots$$

$$\frac{A \in U \qquad B(x) \in U \; [x \in A]}{(\Pi x \in A)B(x) \in U} \qquad \frac{A \in U \qquad B(x) \in U \; [x \in A]}{(\Sigma x \in A)B(x) \in U}$$

$$\frac{a \in U}{a \; set} \qquad \frac{a = b \in U}{a = b}$$

Uniwersum ala Russel

- wygodne
- różnica w sensie: posługiwanie się nazwami typów a posługiwanie się typami
  - pokazanie $\neg[\widehat{N} =_U \widehat{Bool}]$ to jak pokazanie $\neg[0 =_{Nat} 1]$
  - pokazanie $\neg[N =_U Bool]$ nie jest takie proste

$$\forall b \in Bool.\ [b =_{Bool} true] \vee [b =_{Bool} false]$$

Uniwersum ala Russel

$$U_0 = U$$

$$\frac{}{U_n \; set}$$

$$\frac{A \in U_n \qquad B(x) \in U_n \; [x \in A]}{(\Pi x \in A)B(x) \in U_n} \qquad \frac{A \in U_n \qquad B(x) \in U_n \; [x \in A]}{(\Sigma x \in A)B(x) \in U_n}$$

$$\frac{}{U_n \in U_{n+1}}$$

W-typy

$$\frac{A\ set \qquad B(x)\ set\ [x \in A]}{(Wx \in A)B(x)\ set}$$

$$\frac{a \in A \qquad b(x) \in (Wx \in A)\ B(x)\ [x \in B(a)]}{sup(a, b) \in (Wx \in A)B(x)}$$

- dobrze ufundowane drzewa
- kodowanie typów indukcyjnych

W-typy

$$
\begin{array}{rcl}
B(x) & \equiv & Set(case_2(x, \hat{N}_0, \hat{N}_1)) \\
Nat & = & (Wx \in N_2)B(x) \\
zero & = & sup(0_2, case_0) \\
succ(n) & = & sup(1_2, (x)n)
\end{array}
$$

W-typy

$$\frac{\begin{array}{c} a \in (Wx \in A)B(x) \\ C(v)\ set\ [v \in (Wx \in A)B(x)] \\ b(y,z,u) \in C(sup(y,z)) \\ [\\ y \in A, \\ z(x) \in (Wx \in A)B(x)\ [x \in B(y)] \\ u(x) \in C(z(x))\ [x \in B(y)] \\ ] \end{array}}{wrec(a,b) \in C(a)}$$

W-typy

$$\frac{d \in A \qquad e(x) \in (Wx \in A)B(x)\ [x \in B(d)] \quad C(v)\ set\ [v \in (Wx \in A)B(x)] \quad b(y,z,u) \in C(sup(y,z))\ \left[\begin{array}{c} y \in A, \\ z(x) \in (Wx \in A)B(x)\ [x \in B(y)] \\ u(x) \in C(z(x))\ [x \in B(y)] \end{array}\right]}{wrec(sup(d,e), b) = b(d, e, (x)wrec(e(x), b)) \in C(sup(d,e))}$$

W-typy

$$
\begin{aligned}
B(x) &\equiv Set(case_2(x, \hat{N}_0, \hat{N}_1)) \\
Nat &= (Wx \in N_2)B(x) \\
zero &= sup(0_2, case_0) \\
succ(n) &= sup(1_2, (x)n) \\
natrec(a, b, c) &= wrec(a, (y, z, u)case_2(y, b, c(z(tt), u(tt))))
\end{aligned}
$$

- równość ekstensjonalna lub aksjomat ekstensjonalności funkcji