

Seminarium: Programowanie w teorii typów

Teoria typów cd.

Wojciech Jedynek, Paweł Wieczorek

Instytut Informatyki Uniwersytetu Wrocławskiego

13 października 2011

Plan na dziś

- Co już umiemy:
 - ▶ posługiwać się systemem naturalnej dedukcji
 - ▶ przerbnięliśmy 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

$$\begin{array}{c}
 \frac{}{N_k \text{ set}} \quad \frac{\text{dla } m < k}{m_k \in N_k} \\
 \\
 \begin{array}{c}
 a \in N_k \quad C(v) \text{ set } [v \in N_k] \\
 c_0 \in C(0_k) \\
 \vdots \\
 c_{k-1} \in C((k-1)_k)
 \end{array} \\
 \hline
 \text{case}_k(a, c_0, \dots, c_{k-1}) \in C(a) \\
 \\
 \begin{array}{c}
 C(v) \text{ set } [v \in N_k] \\
 c_0 \in C(0_k) \\
 \vdots \\
 c_{k-1} \in C((k-1)_k)
 \end{array} \\
 \hline
 \text{case}_k(m_k, c_0, \dots, c_{k-1}) = c_m \in C(m_k)
 \end{array}$$

Typy wyliczeniowe

$$N_0 = \perp$$

$$\neg A = A \rightarrow N_0$$

Typy wyliczeniowe

$$N_0 = \perp$$

$$\neg A = A \rightarrow N_0$$

$$N_1 = \top$$

$$N_1 = \mathit{Unit}$$

$$0_1 = \mathit{tt}$$

Typy wyliczeniowe

$$N_0 = \perp$$

$$\neg A = A \rightarrow N_0$$

$$N_1 = \top$$

$$N_1 = \textit{Unit}$$

$$0_1 = \textit{tt}$$

$$N_2 = \textit{Bool}$$

$$0_2 = \textit{true}$$

$$1_2 = \textit{false}$$

$$\textit{case}_2(a, c_0, c_1) = \textit{if } a \textit{ then } c_0 \textit{ else } c_1$$

Typy wyliczeniowe

$$N_0 = \perp$$

$$\neg A = A \rightarrow N_0$$

$$N_1 = \top$$

$$N_1 = \textit{Unit}$$

$$0_1 = \textit{tt}$$

$$N_2 = \textit{Bool}$$

$$0_2 = \textit{true}$$

$$1_2 = \textit{false}$$

$$\textit{case}_2(a, c_0, c_1) = \textit{if } a \textit{ then } c_0 \textit{ else } c_1$$

$$\frac{a \in N_0 \quad C(v) \text{ set } [v \in N_0]}{\textit{case}_0(a) \in C(a)} \quad \frac{\perp \textit{ true} \quad C \textit{ prop}}{C \textit{ true}}$$

Wyrażenia

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

Wyrażenia

- język wyrażeń:
 - ▶ $e(e')$ - aplikacja ($e(e_1, \dots, e_n)$ skrócony zapis $e(e_1) \dots (e_n)$)
 - ▶ $(x)e$ - abstrakcja
 - ▶ stałe, np λ , Π , *apply*, 0, *succ*
- posługuję się prostym systemem typów (z jednym typem bazowym O) zwanym „arnościami”
- $e(e') : \beta$ gdy $e : \alpha \rightarrow \beta$, $e' : \alpha$
- konstruktory:
 - ▶ $\lambda : (O \rightarrow O) \rightarrow O$
 - ▶ $\Pi : O \rightarrow (O \rightarrow O) \rightarrow O$
 - ▶ *apply* : $O \rightarrow O \rightarrow O$
- 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 ...”:
 $\Gamma \vdash (t \in A) \rightarrow \varphi$
 - ▶ 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 ...”:
 $\Gamma \vdash (t \in A) \rightarrow \varphi$
 - ▶ 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 : $\forall x y. x = y \rightarrow P(x) \rightarrow P(y)$

Typ identycznościowy

- Sądy a formuły to nie to samo
 - ▶ nie możemy powiedzieć wewnątrz systemu „jeżeli t ma typ A to ...”:
 $\Gamma \vdash (t \in A) \rightarrow \varphi$
 - ▶ 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 : $\forall x y. x = y \rightarrow P(x) \rightarrow P(y)$
- Co znaczy, że funkcje są sobie równe?
 - ▶ Ile jest funkcji (jako zbiór par) sortujących? Ekstensjonalnie tylko jedna.
 - ▶ $(\forall xs \in List. qsort(xs) \equiv isort(xs)) \rightarrow qsort \equiv isort$

Typ identycznościowy

- Sądy a formuły to nie to samo
 - ▶ nie możemy powiedzieć wewnątrz systemu „jeżeli t ma typ A to ...”: $\Gamma \vdash (t \in A) \rightarrow \varphi$
 - ▶ 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 : $\forall x y. x = y \rightarrow P(x) \rightarrow P(y)$
- Co znaczy, że funkcje są sobie równe?
 - ▶ Ile jest funkcji (jako zbiór par) sortujących? Ekstensjonalnie tylko jedna.
 - ▶ $(\forall xs \in List. qsort(xs) \equiv isort(xs)) \rightarrow qsort \equiv isort$
- 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 $a = b \in A$
 - ▶ równość typów $A = B$
- chcemy jeszcze:
 - ▶ formuła logiczna $[a =_A b]$

Typ identycznościowy

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

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

Typ identycznościowy

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

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

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

Typ identycznościowy

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

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

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

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

Typ identycznościowy

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

- 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

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

$$\frac{\begin{array}{l} a \in A \\ C(x, y, z) \text{ set } [x \in A, y \in A, z \in [x =_A y]] \\ d(x) \in C(x, x, id(x)) [x \in A] \end{array}}{idpeel(id(a), d) = d(a) \in C(a, a, id(a))}$$

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

Typ identycznościowy

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

$$\frac{\begin{array}{l} a \in A \quad b \in A \quad c \in [a =_A b] \\ C(x, y, z) \text{ set } [x \in A, y \in A, z \in [x =_A y]] \\ id(x) \in C(x, x, id(x)) [x \in A] \end{array}}{idpeel(c, id) \in C(a, b, c)}$$
$$\frac{\begin{array}{l} a \in A \quad b \in A \quad c \in [a =_A b] \\ [y =_A x] \text{ set } [x \in A, y \in A, z \in [x =_A y]] \\ id(x) \in [x =_A x] [x \in A] \end{array}}{idpeel(c, id) \in [b =_A a]}$$

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

Typ identycznościowy

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

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

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

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

Typ identycznościowy

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

Typ identycznościowy

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

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

- równość 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 $C(x, y, z) \equiv P(x) \rightarrow P(y)$

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

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

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

Typ identycznościowy

- Nie potrafimy wydać sądu
 $if\ b\ then\ c\ else\ c = c \in A\ [b \in Bool, c \in A]$
- Potrafimy udowodnić $(\prod b \in Bool)(\prod c \in A)[if\ b\ then\ c\ else\ c =_A c]$

Uniwersum

$$\begin{array}{c} \frac{}{Set \ set} \qquad \frac{}{Nat \in Set} \\[1em] \frac{A \in Set \quad B \in Set}{A \rightarrow B \in Set} \qquad \frac{A \in Set \quad B \in Set}{A \times B \in Set} \\[1em] \dots \\[1em] \frac{}{Set \in Set} \end{array}$$

Uniwersum ala Tarski

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

Uniwersum ala Tarski

$$\frac{}{U \text{ set}} \quad \frac{}{\widehat{Nat} \in U}$$

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

...

$$\frac{a \in U \quad b \in U}{\widehat{a \rightarrow b} \in U} \quad \frac{a \in U \quad b \in U}{\widehat{a \times b} \in U}$$

Uniwersum ala Tarski

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

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

$$Set(\widehat{Nat}) = Nat$$

$$Set(\widehat{N_0}) = N_0$$

$$Set(\widehat{N_1}) = N_1$$

$$Set(\widehat{N_2}) = N_2$$

...

$$Set(\widehat{a \rightarrow b}) = Set(a) \rightarrow Set(b)$$

$$Set(\widehat{a \times b}) = Set(a) \times Set(b)$$

Uniwersum ala Tarski

- Co nam się nie podoba w tej funkcji?

$$g \in \text{Nat} \rightarrow \text{Bool} \cup \text{Nat}$$

$$g(2n) = \text{left}(\text{true})$$

$$g(2n + 1) = \text{right}(42)$$

Uniwersum ala Tarski

- Co nam się nie podoba w tej funkcji?

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

- Bezpieczniejsza wersja

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

Uniwersum ala Tarski

$$\begin{aligned} B(x) &\equiv \text{Set}(\text{case}_2(\text{apply}(\text{isEven}, x), \widehat{Bool}, \widehat{Nat})) \\ b &\equiv (x)\text{case}_2(\text{apply}(\text{isEven}, x), \text{true}, 42) \end{aligned}$$

Uniwersum ala Tarski

$$B(x) \equiv \text{Set}(\text{case}_2(\text{apply}(\text{isEven}, x), \widehat{\text{Bool}}, \widehat{\text{Nat}}))$$

$$b \equiv (x)\text{case}_2(\text{apply}(\text{isEven}, x), \text{true}, 42)$$

$$\lambda x.b \in (\prod n \in \text{Nat}) B(n)$$

$$\text{apply}(\lambda x.b, n) \in B(n) [n \in \text{Nat}]$$

Uniwersum ala Tarski

$$B(x) \equiv \text{Set}(\text{case}_2(\text{apply}(\text{isEven}, x), \widehat{\text{Bool}}, \widehat{\text{Nat}}))$$

$$b \equiv (x)\text{case}_2(\text{apply}(\text{isEven}, x), \text{true}, 42)$$

$$\lambda x.b \in (\prod n \in \text{Nat})B(n)$$

$$\text{apply}(\lambda x.b, n) \in B(n) [n \in \text{Nat}]$$

- niech $\text{negb} \in \text{Bool} \rightarrow \text{Bool}$.

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

Uniwersum ala Tarski

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

$$nary \equiv (a, x)natrec(x, a, (n, y)\widehat{a \rightarrow y})$$

$$\lambda x.nary(a, x) \in (\Pi x \in Nat)U [a \in U]$$

$$\lambda a.\lambda x.nary(a, x) \in (\Pi a \in U)(\Pi x \in Nat)U$$

$$nAry \equiv \lambda a.\lambda x.nary(a, x)$$

Uniwersum ala Tarski

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

$$nary \equiv (a, x)natrec(x, a, (n, y)\widehat{a \rightarrow y})$$

$$\lambda x.nary(a, x) \in (\Pi x \in Nat)U [a \in U]$$

$$\lambda a.\lambda x.nary(a, x) \in (\Pi a \in U)(\Pi x \in Nat)U$$

$$nAry \equiv \lambda a.\lambda x.nary(a, x)$$

$$boolfun \equiv (n)Set(apply(apply(nAry, \widehat{Bool}), n))$$

$$\lambda n.boolfun \in (\Pi n \in Nat)Set$$

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

$$fst(a, b) = a \quad fst3(a, b, c) = a \quad fst4(a, b, c, d) = a$$

Uniwersum ala Tarski

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

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

Uniwersum ala Tarski

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

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

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

Uniwersum ala Tarski

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

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

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

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

Uniwersum ala Tarski

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

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

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

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

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

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 $n \in Nat$, $e \in [0 =_{Nat} succ(n)]$

$$IsZ(m) \equiv Set(natrec(m, \widehat{N}_1, (x, y)\widehat{N}_0))$$

$$IsZ(0) \equiv N_1$$

$$IsZ(succ(n)) \equiv N_0$$

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

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

Universum ala Russel

$$\frac{}{U \text{ set}} \quad \frac{}{Nat \in U}$$

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

...

$$\frac{A \in U \quad B(x) \in U [x \in A]}{(\prod x \in A) B(x) \in U} \quad \frac{A \in U \quad B(x) \in U [x \in A]}{(\sum x \in A) B(x) \in U}$$

$$\frac{a \in U}{a \text{ set}} \quad \frac{a = b \in U}{a = b}$$

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

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

Universum ala Russel

$$U_0 = U$$

$$\frac{}{U_n \text{ set}}$$

$$\frac{A \in U_n \quad B(x) \in U_n [x \in A]}{(\prod x \in A) B(x) \in U_n}$$

$$\frac{A \in U_n \quad B(x) \in U_n [x \in A]}{(\sum x \in A) B(x) \in U_n}$$

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

W-typy

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

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

- dobrze ufundowane drzewa
- kodowanie typów indukcyjnych

$$B(x) \equiv \text{Set}(\text{case}_2(x, \hat{N}_0, \hat{N}_1))$$

$$\text{Nat} = (\forall x \in N_2) B(x)$$

$$\text{zero} = \text{sup}(0_2, \text{case}_0)$$

$$\text{succ}(n) = \text{sup}(1_2, (\lambda x) n)$$

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

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

$$wrec(\text{sup}(d, e), b) = b(d, e, (x)wrec(e(x), b)) \in C(\text{sup}(d, e))$$

W-typy

$$B(x) \equiv \text{Set}(\text{case}_2(x, \hat{N}_0, \hat{N}_1))$$

$$\text{Nat} = (Wx \in N_2)B(x)$$

$$\text{zero} = \text{sup}(0_2, \text{case}_0)$$

$$\text{succ}(n) = \text{sup}(1_2, (x)n)$$

$$\text{natrec}(a, b, c) = \text{wrec}(a, (y, z, u)\text{case}_2(y, b, c(z(tt), u(tt))))$$

- równość ekstensjonalna lub aksjomat ekstensjonalności funkcji