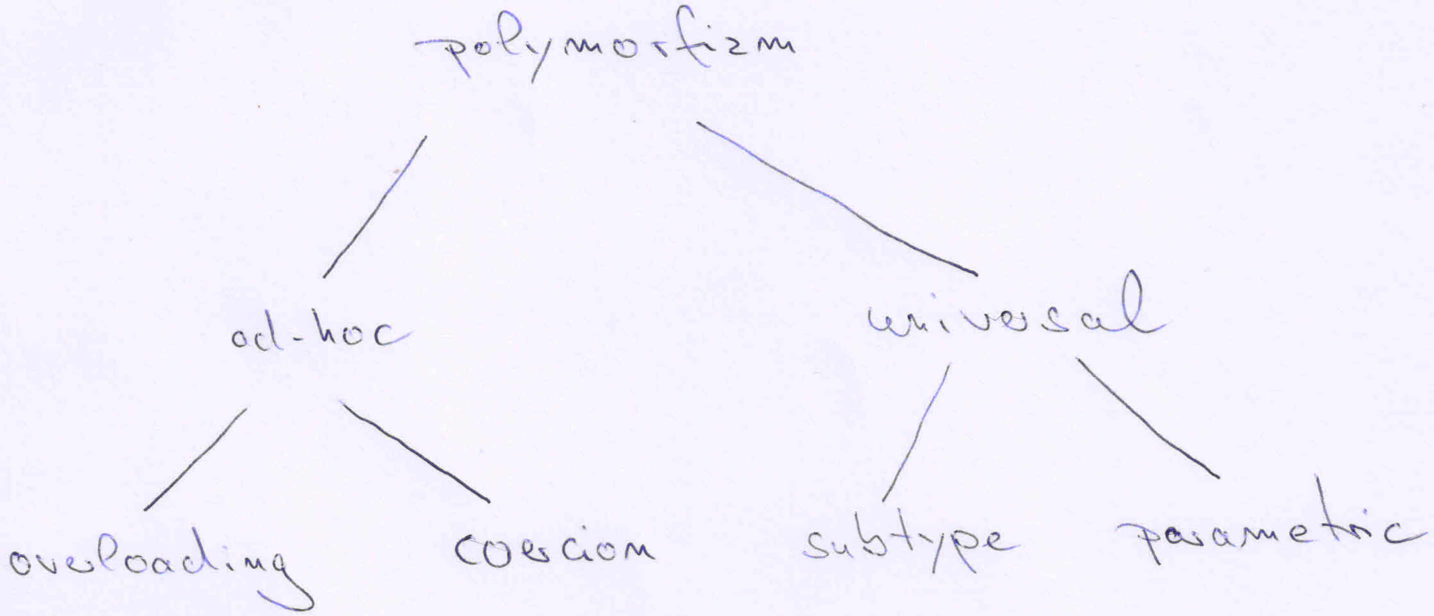
§ 4 Typy w Haskellu

silne statyczne typowanie

- każdy wyrażenie ma typ (który Haskell automatycznie obliczy)
- w czasie wykonania nie wolno zmienić typy

typy polymorficzne

wyrażenie może mieć więcej niż jeden typ – zmienne typów.



4.1 Obliczenie typów

Składnia typów (bez klas)

$$
\begin{aligned}
\langle \text{typ} \rangle ::= \; & \langle \text{typ bazowy} \rangle \\
| \; & \langle \text{zmienna typów} \rangle \\
| \; & \langle \text{konstruktor typów}_{(n)} \rangle \; \langle \text{typ} \rangle^{n}
\end{aligned}
$$

$$\langle \text{typ bazowy} \rangle ::= \text{Int} \mid \text{Char} \mid \text{Bool} \mid \ldots$$

konstruktory typów:

- zbudowane
  - listy $[\cdot]$
  - tuple $(\cdot, \ldots, \cdot)$
  - funkcja $\cdot \to \cdot$
- przez użytkownika
  - drzewa
  - ⋮

Przykład:

(i) $2 :: \text{Int}$

$(+) :: \text{Int} \to (\text{Int} \to \text{Int})$

(ii) $\text{length} :: [\alpha] \to \text{Int}$

typ polimorficzne (α)

instancje: $[\text{Char}] \to \text{Int}$

$[[\beta]] \to \text{Int}$

(iii) $(:) :: \alpha \to [\alpha] \to [\alpha]$

(iv) $\text{map} :: (\alpha \to \beta) \to [\alpha] \to [\beta]$

(v) $2 * a :: ?$

jeżeli $a :: Int$, to $2 * a :: Int$

$(\backslash x \to 2 * a + x) :: Int \to Int$,

ale tylko, jeżeli $a :: Int$

Typing judgement

$$\Gamma \vdash e :: \alpha$$

e wyrażenie

α typ

Γ kontekst, tzn. $\Gamma = \{ x_i :: \alpha_i \mid i = 1, \dots n \}$

gdzie x_i zmienna, α_i typ

Przykład:

(i) $\{x :: Int\} \vdash x + 2 :: Int$

(ii) $\{a :: Int, y :: Int\} \vdash (\backslash x \to a + 2 * x) :: Int \to Int$

(iii) $\{x :: \alpha \to \beta\} \vdash (\backslash y \to x) :: \gamma \to (\alpha \to \beta)$

(iv) $\emptyset \vdash length :: [\alpha] \to Int$

(v) $\{x :: Int\} \vdash [1, x] :: [Char]$?

system inferencji!

Systemy inferencji

zamiast $\models$, dedukcja na podstawie składni
do tego: aksjomaty i reguły.

Przykład:

(i) $$\frac{A,\ A \to B}{B}$$

(ii) $$\frac{A \vee B}{A}$$

↳ krok po kroku "idziemy" od aksjomatów do twierdzeń.

Dowód dla T, to

ciąg $P_1, \dots P_n$ taki że

(i) $P_n = T$

(ii) $\forall i \leq n$: P_i "=" Axiom lub

P_i "=" wynik zastosowania reguły

tzn.

$\exists$ reguła $\dfrac{A_1 \ \dots\dots \ A_m}{A}$: P_i "=" A $\wedge$

$\forall k = 1, \dots m \ \exists j < i : A_k$ "=" P_j

w tym przypadku T jest twierdzeniem (systemu) i piszemy $\vdash T$.

$\vdash$ poprawny, jeśli $\vdash T \Rightarrow \models T$

$\vdash$ zupełny, jeśli $\models T \Rightarrow \vdash T$

u nas:

$\models \Gamma \vdash e :: \alpha$, jeżeli

"pod warunkiem, że zmienne $x_i \in \Gamma$ mają podany typ α_i, α jest "prawdziwym typem" wyrażenia e."

Cel: system inferencji $\vdash$ taki że

$$\vdash \Gamma \vdash e :: \alpha \quad \text{wtw.} \quad \models \Gamma \vdash e :: \alpha$$

Aksjomaty

(i) $\emptyset \vdash f :: \alpha$

dla operacji bazowej z typem α

Przykłady:

$\emptyset \vdash 7 :: Int$

$\emptyset \vdash$ 'a' $:: Char$

$\emptyset \vdash (+) :: Int \to Int \to Int$

$\emptyset \vdash (:) :: \alpha \to [\alpha] \to [\alpha]$

(ii) $\{x :: \alpha\} \vdash x :: \alpha$

dla zmiennej x i typu α

Przykłady:

$\{x :: Int\} \vdash x :: Int$

$\{a :: Char\} \vdash a :: Char$

$\{l :: [\alpha]\} \vdash l :: [\alpha]$

Reguły

- dla abstrakcji (budowania funkcji)
- dla aplikacji (zastosowania funkcji)

(i)

$$\frac{\Gamma \cup \{x :: \alpha\} \vdash e :: \beta}{\Gamma \vdash (\backslash x \to e) :: \alpha \to \beta}$$

Przykłady:

$$\frac{\{x :: \text{Int}\} \vdash x * x :: \text{Int}}{\emptyset \vdash (\backslash x \to x * x) :: \text{Int} \to \text{Int}}$$

$$\cfrac{\cfrac{\{x :: \text{Int},\ a :: \text{Int}\} \vdash a + 2 * x :: \text{Int}}{\{x :: \text{Int}\} \vdash (\backslash a \to a + 2 * x) :: \text{Int} \to \text{Int}}}{\emptyset \vdash \underbrace{(\backslash x \to (\backslash a \to a + 2 * x))}_{\backslash x\ a\ \to\ a + 2 * x} :: \text{Int} \to (\text{Int} \to \text{Int})}$$

(ii)

$$\frac{\Gamma_1 \vdash f :: \alpha \to \beta \qquad \Gamma_2 \vdash e :: \alpha}{\Gamma_1 \cup \Gamma_2 \vdash (f\ e) :: \beta}$$

uwaga: Γ_1 i Γ_2 muszą być kompatybilne

tzn.

$$\forall \alpha : (\alpha :: e_1 \in \Gamma_1 \land \alpha :: e_2 \in \Gamma_2) \Rightarrow e_1 = e_2$$

Przykłady:

$$\frac{\emptyset \vdash (\backslash x \to x * x) :: \mathrm{Int} \to \mathrm{Int} \qquad \emptyset \vdash 1 :: \mathrm{Int}}{\emptyset \vdash (\backslash x \to x * x)\ 1 :: \mathrm{Int}}$$

$$\frac{\{1 :: \mathrm{Int}\} \vdash \mathrm{length} :: [\mathrm{Char}] \to \mathrm{Int} \qquad \emptyset \vdash [\text{'a'}] :: [\mathrm{Char}]}{\{1 :: \mathrm{Int}\} \vdash \mathrm{length}\ [\text{'a'}] :: \mathrm{Int}}$$

$$\frac{\begin{array}{l}\{a :: \mathrm{Int}\} \vdash \backslash x \to a + 2 * x :: \mathrm{Int} \to \mathrm{Int} \\ \{y :: \mathrm{Int}\} \vdash 5 * y :: \mathrm{Int}\end{array}}{\begin{array}{l}\{a :: \mathrm{Int}, y :: \mathrm{Int}\} \vdash \\ \qquad \underbrace{(\backslash x \to a + 2 * x)(5 * y)}_{\text{"="}\ \backslash x \to a + 10 * y} :: \mathrm{Int}\end{array}}$$

Dodatek:

$$\frac{\Gamma_1 \vdash e :: \alpha}{\Gamma_1 \cup \Gamma_2 \vdash e :: \alpha}$$

jeżeli Γ_1 i Γ_2 kompatybilne

Przykład:

$$\frac{\dfrac{\emptyset \vdash 3 :: Int}{\{x :: Char\} \vdash 3 :: Int}}{\emptyset \vdash \backslash x \to 3 :: Char \to Int}$$

Typy polymorficzne

$$\frac{\emptyset \vdash length :: [\alpha] \to Int \qquad \emptyset \vdash [1,2,3] :: [Int]}{?}$$

$$[\alpha] \neq [Int] \quad !$$

$$\frac{?}{\Gamma \vdash (\backslash f \to f\ 3) :: (Int \to \alpha) \to \alpha}$$

↳ uogólnić regułę aplikacji

↳ substitucja, unifikator (najbardziej ogólny)