Curry-típusrendszer: szintaktika (ismétlés)

Definíció. ***A Curry-típusrendszer szintaktikája***

<típus>	*::=*	*<típusváltozó>*
	|	*(<típus> → <típus>)*
<λ-kifejezés>	*::=*	*<változó>*
	|	*(λ <változó> . <λ-kifejezés>)*
	|	*(<λ-kifejezés> <λ-kifejezés>)*

$\alpha, \beta, \ldots$ típusváltozók
τ típuskifejezés

$E : \tau$: Az E kifejezés típusa τ

F_1: $\lambda x : Nat.x : Nat \to Nat$ $\qquad \alpha \equiv \alpha$

Curry: $\lambda x.x : \alpha \to \alpha$ $\qquad \alpha \not\equiv \beta$

Curry: $\lambda x.x : (\alpha \to \beta) \to (\alpha \to \beta)$ $\qquad \alpha \to \alpha \not\equiv \beta \to \beta$

F_1: $\{\ y : Bool\ \} \vdash \lambda x : Bool \to Nat.x\ y : (Bool \to Nat) \to Nat$

Curry: $\{\ y : \beta\ \} \vdash \lambda x.x\ y : (\beta \to \alpha) \to \alpha$

Curry-típuskikövetkeztetés (ismétlés)

- Típusellenőrzés, a típus levezetése:
  Adott τ és E esetén $\varnothing \vdash E : \tau$ teljesül-e?
- Típusreprezentáció:
  Adott τ esetén, létezik-e E, hogy $\varnothing \vdash E : \tau$?
- Típuskikövetkeztetés, a típus rekonstrukciója:
  Adott E esetén létezik-e τ, hogy $\varnothing \vdash E : \tau$?

Típusozás korlátozásokkal: a korlátozások generálása, azok megoldása

- Korlátozás (*constraint*): típuskifejezések közti egyenlőség
- Egységesítés (*unification*): a korlátozások egyenletrendszerének megoldása

Curry-típuskikövetkeztetés ≡ Wand-algoritmus

Curry-típusrendszer + `let`-polimorfizmus ≡ Hindley-Milner típusrendszer

Hindley-Milner típusrendszer + polimorfikus rekurzió ≡ Milner-Mycroft típusrendszer

A Hindley-Milner típuskikövetkeztetés

Hindley-Milner: egyszerű `let`-kifejezés (szintaktika)

Definíció. ***Az egyszerű `let`-kifejezés szintaktikája***

<λ-kifejezés> ::= *<változó>*
| *(λ <változó> . <λ-kifejezés>)*
| *(<λ-kifejezés> <λ-kifejezés>)*
| `let` *<változó>* = *<λ-kifejezés>* `in` *<λ-kifejezés>*

`let` x = E `in` F

ahol

- E az x változó definíciója
- F a `let`-kifejezés törzse (hatásköre)
- $x \notin FV(E)$
- x típusa monomorf F-ben

A `let`-kifejezések egymásba skatulyázhatóak.

Hindley-Milner: egyszerű `let`-kifejezés (szemantika)

Definíció. ***Az egyszerű `let`-kifejezés operációs szemantikája***

$$\texttt{let}\ x = E\ \texttt{in}\ F \rightarrowtriangle F\,[x := E]$$

Definíció. ***Az egyszerű `let`-kifejezés típusszabálya***

$$\frac{\Gamma \vdash E : \tau, \Gamma, x : \tau \vdash F : \tau'}{\Gamma \vdash \texttt{let}\ x = E\ \texttt{in}\ F : \tau'}\ (\text{VAL LET})$$

vagy a $\texttt{let}\ x = E\ \texttt{in}\ F \equiv (\lambda x.F)\ E$ miatt:

$$\frac{\Gamma \vdash E : \tau \qquad \dfrac{\Gamma, x : \tau \vdash F : \tau'}{\Gamma \vdash \lambda x.F : \tau \to \tau'}\ (\text{VAL ABS})}{\Gamma \vdash (\lambda x.F)\ E : \tau'}\ (\text{VAL APPL})$$

Hindley-Milner: egyszerű `let`-kifejezés (korlátozások)

Definíció. ***Korlátozások generálása***

A korlátozások generálását a $\mathcal{T}(\Gamma, E, \tau)$ *leképezéssel végezzük, ahol* Γ *egy típuskörnyezet,* E *egy kifejezés, és* τ *egy típus.*

1. $\mathcal{T}(\Gamma, x, \tau) \quad \rightsquigarrow \quad \{\tau = \Gamma(x)\}$
2. $\mathcal{T}(\Gamma, \lambda x.E, \tau) \quad \rightsquigarrow \quad \mathcal{T}(\Gamma \cup \{x : \alpha\}, E, \beta) \cup \{\tau = \alpha \to \beta\}$
   ahol α *és* β *új típusváltozók*
3. $\mathcal{T}(\Gamma, E\ F, \tau) \quad \rightsquigarrow \quad \mathcal{T}(\Gamma, E, \alpha \to \tau) \cup \mathcal{T}(\Gamma, F, \alpha)$
   ahol α *új típusváltozó*

Az új szabály:

4. $\mathcal{T}(\Gamma, \texttt{let}\ x = E\ \texttt{in}\ F, \tau) \quad \rightsquigarrow \quad \mathcal{T}(\Gamma, E, \alpha) \cup \mathcal{T}(\Gamma \cup \{x : \alpha\}, F, \tau)$
   ahol α *egy új típusváltozó*

A principális típust továbbra is az egységesítés korábban megismert algoritmusával határozhatjuk meg.

Hindley-Milner: `let`-polimorfizmus

F-ben az x változó mindegyik előfordulása ugyanolyan típusú, tehát nem lehet polimorf. Mint például:

$\texttt{let}\ f = \lambda x.x\ \texttt{in}\ \texttt{pair}\ (f\ 0)\ (f\ \texttt{true})$

$f : \mathit{Nat} \to \mathit{Nat}$ és $f : \mathit{Bool} \to \mathit{Bool}$!

Megoldás:

F-ben x szerepelhessen n-szer: $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$, így külön-külön mindegyik előfordulásra meghatározható egy típus.

$\texttt{let}\ x = E\ \texttt{in}\ F \neq (\lambda \mathrm{x.F})\ \mathrm{E}$

$F\ [x^{(1)} := E, x^{(2)} := E, \ldots, x^{(n)} := E]$

$$\frac{\Gamma \vdash E : \tau \qquad \Gamma, x : \tau \vdash F\ [x^{(1)} := E, x^{(2)} := E, \ldots, x^{(n)} := E] : \tau'}{\Gamma \vdash \texttt{let}\ x = E\ \texttt{in}\ F : \tau'}$$

A Hindley-Milner-típusrendszer

HM-típusrendszer: típussémák

Definíció. ***A Hindley-Milner-típusrendszer típusai***

<típus>	*::=*	*<egyszerű típus>*
	|	*(∀ <típusváltozó> . <típus>)*
<egyszerű típus>	*::=*	*<típusváltozó>*
	|	*(<egyszerű típus> → <egyszerű típus>)*

τ: egyszerű típus (monotípus), σ: típusséma (politípus)

$\forall\overline{\alpha}.\sigma \equiv \forall\alpha_1.\forall\alpha_2.\ldots.\forall\alpha_n.\sigma \equiv \forall\alpha_1\alpha_2\ldots\alpha_n.\sigma$ $(n > 0)$

– kötött változók: generikus vagy polimorfikus típusváltozók

Definíció. ***Szabad típusváltozók***

$FTV(\tau)$	=	τ összes típusváltozója
$FTV(\sigma)$	=	$FTV(\tau) \setminus \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, ha $\sigma \equiv \forall\alpha_1\alpha_2\ldots\alpha_n.\tau$

Definíció. ***Típuskörnyezet szabad változói***

$FTV(\Gamma) = \{\alpha \mid \{x : \sigma\} \in \Gamma, \alpha \in FTV(\sigma)\}$

$\Gamma = \{x : \forall\alpha.\alpha \to \beta\}, FTV(\Gamma) = \{\beta\}$

$\Gamma = \{x : \forall\alpha.\alpha \to \beta, y : \gamma \to \delta\}, FTV(\Gamma) = \{\beta, \gamma, \delta\}$

HM-típusrendszer: helyettesítések típussémákban

Definíció. ***Típusváltozó helyettesítése típussémában***

$$(\forall\alpha.\sigma)\ [\beta := \tau] \equiv \begin{cases} \forall\alpha.\sigma\ [\beta := \tau] & \text{ha } \alpha \not\equiv \beta \text{ és } \alpha \notin \mathit{FTV}(\tau) \\ \forall\alpha.\sigma & \text{egyébként} \end{cases}$$

$$\begin{array}{lcl} (\forall\alpha.\beta \to \gamma)\ [\beta := \gamma \to \delta] & \equiv & \forall\alpha.(\gamma \to \delta) \to \gamma \\ (\forall\alpha.\beta \to \gamma)\ [\beta := \gamma \to \alpha] & \equiv & (\forall\alpha.\beta \to \gamma) \end{array}$$

Definíció. ***Típusséma α-konverziója***

Ha $\beta \notin \mathit{FTV}(\sigma)$*, akkor* $\forall\alpha.\sigma \leftrightarrow_\alpha \forall\beta.\sigma\ [\alpha := \beta]$*.*

Tétel. ***Kötött típusváltozó helyettesítése a típusban***

Ha $\forall\alpha.\sigma \leftrightarrow_\alpha \forall\beta.\sigma\ [\alpha := \beta]$*, akkor* $\Gamma \vdash E : \forall\alpha.\sigma$ *esetén* $\Gamma \vdash E : \forall\beta.\sigma\ [\alpha := \beta]$ *is fennáll.*

HM-típusrendszer: helyettesítések típussémákban

Tétel. ***Kötött típusváltozó helyettesítése a környezetben***
Ha $\forall\alpha.\sigma \leftrightarrow_\alpha \forall\beta.\sigma\,[\alpha := \beta]$, *akkor* $\Gamma \cup \{x : \forall\alpha.\sigma\} \vdash E : \sigma'$ *esetén* $\Gamma \cup \{x : \forall\beta.\sigma\,[\alpha := \beta]\} \vdash E : \sigma'$ *is teljesül.*

$\Gamma = \{x : \forall\alpha.\alpha \to \beta, y : \forall\beta.\alpha \to \beta\}$, $FTV(\Gamma) = \{\alpha, \beta\}$

$$\begin{array}{lll} \forall\alpha.\alpha \to \beta & \leftrightarrow_\alpha & \forall\gamma.\gamma \to \beta \\ \forall\beta.\alpha \to \beta & \leftrightarrow_\alpha & \forall\delta.\alpha \to \delta \end{array}$$

$\Gamma' = \{x : \forall\gamma.\gamma \to \beta, y : \forall\delta.\alpha \to \delta\}$, $FTV(\Gamma') = \{\alpha, \beta\}$

HM-típusrendszer: típus általánosítása

Definíció. ***Típus általánosítása (liftelés, lezárás)***
A τ típus Γ típuskörnyezetre vett általánosítása a következő típus:

$gen(\Gamma, \tau) = \forall \alpha_1 \alpha_2 \ldots \alpha_n . \tau$,

ahol $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} = FTV(\tau) \setminus FTV(\Gamma)$.

$$
\begin{array}{lcl}
gen(\varnothing, \alpha \to \alpha) & = & \forall \alpha . \alpha \to \alpha \\
gen(\{x : \alpha\}, \alpha \to \alpha) & = & \alpha \to \alpha \\
gen(\{x : \alpha\}, \alpha \to \beta) & = & \forall \beta . \alpha \to \beta \\
gen(\{x : \alpha, y : \beta\}, \alpha \to \beta) & = & \alpha \to \beta
\end{array}
$$

$\Gamma = \{x : \forall \alpha . \alpha \to \beta\}$, $FTV(\Gamma) = \{\beta\}$, $\tau = \alpha \to \beta \to \gamma$,
$FTV(\tau) = \{\alpha, \beta, \gamma\}$, $gen(\Gamma, \tau) = \forall \alpha \gamma . \alpha \to \beta \to \gamma$

$\Gamma = \{x : \alpha, y : \forall \beta . \beta \to \delta\}$, $FTV(\Gamma) = \{\alpha, \delta\}$, $\tau = \alpha \to \beta \to \gamma \to \delta$,
$FTV(\tau) = \{\alpha, \beta, \gamma, \delta\}$, $gen(\Gamma, \tau) = \forall \beta \gamma . \alpha \to \beta \to \gamma \to \delta$

HM-típusrendszer: típus példányosítása

Definíció. ***Típus példányosítása***
A σ típus $inst(\sigma)$ példánya a következő típus:

- $inst(\tau) = \tau$
- $inst(\sigma) = \tau\,[\alpha_1 := \tau_1, \alpha_2 := \tau_2, \ldots, \alpha_n := \tau_n]$
  ha $\sigma \equiv \forall\alpha_1\alpha_2 \ldots \alpha_n.\tau$ *és ahol* $\tau_1, \tau_2, \ldots, \tau_n$ *új típusváltozókat tartalmazó típusok.*

$$
\begin{aligned}
inst(\forall\alpha\forall\gamma.\alpha \to \beta \to \gamma) &= \delta \to \beta \to \varphi \\
inst(\forall\alpha.\alpha \to \beta \to \gamma) &= (\delta \to \delta) \to \beta \to \gamma
\end{aligned}
$$

HM-típusrendszer: szabályok (1/2)

$$\frac{}{\varnothing \vdash \mathsf{wf}}\ (\textsc{env}\ \varnothing) \qquad \frac{\Gamma \vdash \sigma, x \notin \mathit{dom}(\Gamma)}{\Gamma, x : \sigma \vdash \mathsf{wf}}\ (\textsc{env}\ x)$$

$$\frac{\Gamma \vdash \mathsf{wf}, \alpha \in \mathit{TV}}{\Gamma \vdash \alpha}\ (\textsc{type var}) \qquad \frac{\Gamma \vdash \tau', \Gamma \vdash \tau''}{\Gamma \vdash \tau' \to \tau''}\ (\textsc{type arrow})$$

$$\frac{\Gamma \vdash \sigma, \Gamma \vdash \alpha}{\Gamma \vdash \forall \alpha.\sigma}\ (\textsc{type forall}) \qquad \frac{\Gamma', x : \sigma, \Gamma'' \vdash \mathsf{wf}}{\Gamma', x : \sigma, \Gamma'' \vdash x : \sigma}\ (\textsc{val}\ x)$$

HM-típusrendszer: szabályok (2/2)

$$\frac{\Gamma, x : \tau' \vdash E : \tau''}{\Gamma \vdash \lambda x.E : \tau' \to \tau''} \text{ (VAL ABS)}$$

$$\frac{\Gamma \vdash E : \tau' \to \tau'', \Gamma \vdash F : \tau'}{\Gamma \vdash E\ F : \tau''} \text{ (VAL APPL)}$$

$$\frac{\Gamma \vdash E : \sigma}{\Gamma \vdash E : \mathit{gen}(\Gamma, \sigma)} \text{ (VAL GEN)} \qquad \frac{\Gamma \vdash E : \sigma}{\Gamma \vdash E : \mathit{inst}(\sigma)} \text{ (VAL INST)}$$

$$\frac{\Gamma \vdash E : \sigma \qquad \Gamma, x : \sigma \vdash F : \tau}{\Gamma \vdash \texttt{let}\ x = E\ \texttt{in}\ F : \tau} \text{ (VAL LET)}$$

Szintaktika-vezérelt Hindley-Milner-típusrendszer

A HM'-típusrendszer

Mivel a Hindley-Milner-típusrendszerben a szintaktikából nem derül ki, hogy mikor kell a VAL GEN és VAL INST szabályokat alkalmazni.

F típusának meghatározása:

- kiszámítjuk $E\ \tau$ (principális) típusát a hagyományos módon,
- ha τ tartalmaz szabad típusváltozókat, akkor a VAL GEN alapján megadjuk τ általánosítását, azaz $gen(\Gamma, \tau) = \forall \alpha_1 \ldots \alpha_n . \tau$,
- a VAL LET alapján a Γ típuskörnyezet bővül a $\{x : gen(\Gamma, \tau)\}$ hozzárendeléssel,
- F-ben az x i. előfordulásaira alkalmazzuk a VAL INST szabályt, példányosítsuk az általánosított τ típust: $\tau\ [\alpha_1 := \tau_1^{(i)}, \ldots, \alpha_n := \tau_m^{(i)}]$,

A HM'-típusrendszer (folytatás)

Módosított szabályok:

$$\frac{\Gamma \vdash E : \tau \qquad \Gamma \cup \{x : gen(\Gamma, \tau)\} \vdash F : \tau'}{\Gamma \vdash \texttt{let}\ x = E\ \texttt{in}\ F : \tau'}\ (\text{LET POLY}')$$

$$\frac{\Gamma', x : \sigma, \Gamma'' \vdash \mathsf{wf}}{\Gamma', x : \sigma, \Gamma'' \vdash x : inst(\sigma)}\ (\text{VAL INST } x')$$

Tétel. ***A HM és HM′ típusrendszerek egyenlősége***
Minden Γ *típuskörnyezetre és minden* E *kifejezésre*
$\Gamma \vdash_{HM} E : \tau \Leftrightarrow \Gamma \vdash_{HM'} E : \tau$

A `let`-polimorfizmus alkalmazása

A `let`-polimorfizmus alapján a következő kifejezés típushozható:

$E \equiv \text{let } f = \lambda x.x \text{ in pair}(f\ 0)\ (f\ \text{true})$

$\lambda x.x : \alpha \to \alpha$, liftelve: $\forall \alpha.\alpha \to \alpha$
$(\lambda x : \alpha.x)\ 0 : \mathit{Nat}$, $\lambda x.x : \mathit{Nat} \to \mathit{Nat}$ esetén,
$(\lambda x : \alpha.x)\ \text{true} : \mathit{Bool}$, $\lambda x.x : \mathit{Bool} \to \mathit{Bool}$ esetén,
$E : \mathit{Pair}_{\mathit{Nat} \times \mathit{Bool}}$.

viszont az alábbi nem (csak a törzsben lehet polimorfizmus):

$F \equiv \text{let } f = \lambda x.1 \text{ in let } g = \lambda y.(y\ 1) + (y\ \text{true}) \text{ in } g\ f$

A $\mathcal{W}$-algoritmus

$\mathcal{W}$: bevezetés

Definíció. ***A $\mathcal{W}$-algoritmus***
A $\mathcal{W}$-algoritmus egy adott Γ típuskörnyezethez és egy E kifejezéshez meghatároz egy s helyettesítést és egy τ típust, vagy `fail` hibajelzéssel megáll:

$$\mathcal{W} : \Gamma \times E \to \begin{cases} s \times \tau \\ \texttt{fail} \end{cases}$$

Mindig egy egyszerű típust ad eredményül, de a típuskörnyezetben lehetnek típussémák.

Tétel. ***A $\mathcal{W}$-algoritmus helyessége***
Ha $\mathcal{W}(\Gamma, E) = (s, \tau)$, akkor $\Gamma\ s \vdash E : \tau$.

Tétel. ***A $\mathcal{W}$-algoritmus teljessége***
Ha $\mathcal{W}(\Gamma, E) = (s, \tau)$ és egy s' helyettesítésre és egy τ' típusra $\Gamma\ s' \vdash E : \tau'$ is fennáll, akkor van olyan s'' helyettesítés, amelyre $\Gamma\ s' = \Gamma\ ss''$ és $\tau' = inst((gen(\Gamma\ s, \tau))\ s'')$.

$\mathcal{W}$: szabályok

1. $\mathcal{W}(\Gamma, x) \quad \rightsquigarrow \quad \begin{cases} (Id, inst(\sigma)), & \text{ha } \{x : \sigma\} \in \Gamma \\ \texttt{fail} & \text{egyébként} \end{cases}$

2. $\mathcal{W}(\Gamma, \lambda x.E) \quad \rightsquigarrow \quad (s, \alpha\ s \to \tau)$, ahol

2.1. $\mathcal{W}(\Gamma \cup \{x : \alpha\}, E) = (s, \tau)$ és α új változó

3. $\mathcal{W}(\Gamma, E\ F) \quad \rightsquigarrow \quad (s_1\ s_2\ s_3, \alpha\ s_3)$, ahol

3.1. $\mathcal{W}(\Gamma, E) = (s_1, \tau')$,

3.2. $\mathcal{W}(\Gamma\ s_1, F) = (s_2, \tau'')$,

3.3. $s_3 = \mathcal{U}(\{\tau'\ s_2 = \tau'' \to \alpha\}, \varepsilon)$ és α új változó

4. $\mathcal{W}(\Gamma, \texttt{let}\ x = E\ \texttt{in}\ F) \rightsquigarrow (s_1\ s_2, \tau'')$, ahol

4.1. $\mathcal{W}(\Gamma, E) = (s_1, \tau')$,

4.2. $\mathcal{W}(\Gamma\ s_1 \cup \{x : gen(\Gamma\ s_1, \tau')\}, F) = (s_2, \tau'')$

$\Gamma\ s$: Γ minden $x : \sigma$ elemére végrehajtjuk az $x : \sigma$ átalakítást.

$\mathcal{W}$: példa: $\mathcal{W}(\varnothing, \lambda x.x)$

$\mathcal{W}(\varnothing, \lambda x.x) \rightsquigarrow ?$

$\mathcal{W}$: példa: $\mathcal{W}(\varnothing, \lambda x.x)$

$$\mathcal{W}(\varnothing, \lambda x.x) \rightsquigarrow \mathcal{W}(\varnothing, \lambda x_1.x_2) \rightsquigarrow ?$$

$\mathcal{W}$: példa: $\mathcal{W}(\varnothing, \lambda x.x)$

$$\mathcal{W}(\varnothing, \lambda x.x) \rightsquigarrow \mathcal{W}(\varnothing, \lambda x_1.x_2) \rightsquigarrow (S, \alpha\ S \to \tau) \rightsquigarrow ?$$
$$\mathcal{W}(\varnothing \cup \{x_1 : \alpha_1\}, x_2) = (S, \tau)$$

$\mathcal{W}$: példa: $\mathcal{W}(\varnothing, \lambda x.x)$

$$\mathcal{W}(\varnothing, \lambda x.x) \rightsquigarrow \mathcal{W}(\varnothing, \lambda x_1.x_2) \rightsquigarrow (S, \alpha\ S \to \tau) \rightsquigarrow ?$$

$$\mathcal{W}(\varnothing \cup \{x_1 : \alpha_1\}, x_2) \rightsquigarrow \mathcal{W}(\{x_1 : \alpha_1\}, x_2) = (S, \tau)$$

$$\mathcal{W}(\{x_1 : \alpha_1\}, x_2) \rightsquigarrow ?$$

$\mathcal{W}$: példa: $\mathcal{W}(\varnothing, \lambda x.x)$

$$\mathcal{W}(\varnothing, \lambda x.x) \rightsquigarrow \mathcal{W}(\varnothing, \lambda x_1.x_2) \rightsquigarrow (s, \alpha\ s \to \tau) \rightsquigarrow ?$$
$$\mathcal{W}(\varnothing \cup \{x_1 : \alpha_1\}, x_2) \rightsquigarrow \mathcal{W}(\{x_1 : \alpha_1\}, x_2) = (s, \tau)$$
$$\mathcal{W}(\{x_1 : \alpha_1\}, x_2) \rightsquigarrow (\mathit{Id}, \mathit{inst}(\alpha)) \rightsquigarrow ?$$

$\mathcal{W}$: példa: $\mathcal{W}(\varnothing, \lambda x.x)$

$\mathcal{W}(\varnothing, \lambda x.x) \rightsquigarrow \mathcal{W}(\varnothing, \lambda x_1.x_2) \rightsquigarrow (s, \alpha\ s \to \tau) \rightsquigarrow$
$(Id, (\alpha\ Id) \to \alpha) \rightsquigarrow ?$

$\mathcal{W}(\varnothing \cup \{x_1 : \alpha_1\}, x_2) \rightsquigarrow \mathcal{W}(\{x_1 : \alpha_1\}, x_2) = (s, \tau)$

$\mathcal{W}(\{x_1 : \alpha_1\}, x_2) \rightsquigarrow (Id, inst(\alpha)) \rightsquigarrow (Id, \alpha) \rightsquigarrow s = Id, \tau = \alpha$

$\mathcal{W}$: példa: $\mathcal{W}(\varnothing, \lambda x.x)$

$\mathcal{W}(\varnothing, \lambda x.x) \rightsquigarrow \mathcal{W}(\varnothing, \lambda x_1.x_2) \rightsquigarrow (s, \alpha\ s \to \tau) \rightsquigarrow$
$(Id, (\alpha\ Id) \to \alpha) \rightsquigarrow (Id, \alpha \to \alpha) \rightsquigarrow ?$

$\mathcal{W}(\varnothing \cup \{x_1 : \alpha_1\}, x_2) \rightsquigarrow \mathcal{W}(\{x_1 : \alpha_1\}, x_2) = (s, \tau)$

$\mathcal{W}(\{x_1 : \alpha_1\}, x_2) \rightsquigarrow (Id, inst(\alpha)) \rightsquigarrow (Id, \alpha) \rightsquigarrow s = Id, \tau = \alpha$

$\mathcal{W}$: példa: $\mathcal{W}(\varnothing, \lambda x.x)$

$\mathcal{W}(\varnothing, \lambda x.x) \rightsquigarrow \mathcal{W}(\varnothing, \lambda x_1.x_2) \rightsquigarrow (s, \alpha\ s \to \tau) \rightsquigarrow$
$(Id, (\alpha\ Id) \to \alpha) \rightsquigarrow (Id, \alpha \to \alpha) \rightsquigarrow \varnothing \vdash \lambda x.x : \alpha \to \alpha$

$\mathcal{W}(\varnothing \cup \{x_1 : \alpha_1\}, x_2) \rightsquigarrow \mathcal{W}(\{x_1 : \alpha_1\}, x_2) = (s, \tau)$

$\mathcal{W}(\{x_1 : \alpha_1\}, x_2) \rightsquigarrow (Id, inst(\alpha)) \rightsquigarrow (Id, \alpha) \rightsquigarrow s = Id, \tau = \alpha$

$\mathcal{W}$: példa: $\mathcal{W}(\{x : \alpha\}, x\ x)$

$\mathcal{W}(\{x : \alpha\}, x\ x) \rightsquigarrow \mathcal{W}(\{x : \alpha\}, x_1\ x_2) \rightsquigarrow \texttt{fail}$

$\mathcal{W}(\{x : \alpha\}, x_1) = (s_1, \tau') \rightsquigarrow s_1 = Id_1, \tau' = \alpha$

$\mathcal{W}(\{x : \alpha\}, x_1) \rightsquigarrow (Id_1, inst(\alpha)) \rightsquigarrow (Id_1, \alpha)$

$\mathcal{W}(\{x : \alpha\}\ s_1, x_2) = (s_2, \tau'') \rightsquigarrow s_2 = Id_2, \tau'' = \alpha$

$\mathcal{W}(\{x : \alpha\}, x_2) \rightsquigarrow (Id_2, inst(\alpha)) \rightsquigarrow (Id_2, \alpha)$

$s_3 = \mathcal{U}(\{\tau'\ s_2 = \tau'' \to \alpha\}, \varepsilon) \rightsquigarrow \mathcal{U}(\{\alpha\ Id_2 = \alpha \to \beta\}, \varepsilon) \rightsquigarrow$

$\mathcal{U}(\{\alpha = \alpha \to \beta\}, \varepsilon) \rightsquigarrow \texttt{fail}$

$\mathcal{W}$: példa: $\mathcal{W}(\varnothing, \texttt{let}\ f = \lambda x.x\ \texttt{in}\ f\ f)$

$\mathcal{W}(\varnothing, \texttt{let}\ f = \lambda x.x\ \texttt{in}\ f\ f) = (Id_1\ s_2, \tau'') \rightsquigarrow ([\beta := \delta, \delta := \gamma \to \gamma], \gamma \to \gamma)$

$\mathcal{W}(\varnothing, \lambda x_1.x_2) = (s_1, \tau') \rightsquigarrow s_1 = Id_1, \tau' = \alpha \to \alpha$

$\mathcal{W}(\varnothing, \lambda x_1.x_2) = (s, \alpha\ s \to \tau) \rightsquigarrow (Id_1, \alpha\ Id_1 \to \alpha)$

$\mathcal{W}(\varnothing \cup \{x_1 : \alpha\}, x_2) \rightsquigarrow \mathcal{W}(\{x : \alpha\}, x) = (s, \tau) \rightsquigarrow s = Id_1, \tau = \alpha$

$\mathcal{W}(\{x : \alpha\}, x) \rightsquigarrow (Id_1, inst(\alpha)) \rightsquigarrow (Id_1, \alpha)$

$\mathcal{W}(\varnothing\ Id_1 \cup \{f : gen(\varnothing\ Id_1, \alpha \to \alpha)\}, f\ f) = (s_2, \tau'') \rightsquigarrow$

$s_2 = [\beta := \delta, \delta := \gamma \to \gamma], \tau'' = \gamma \to \gamma$

$\mathcal{W}(\{f : \forall\alpha.\alpha \to \alpha\}, f_1\ f_2) =$

$(Id_2\ Id_3\ [\beta := \delta, \delta := \gamma \to \gamma], \delta\ [\delta := \gamma \to \gamma]) \rightsquigarrow$

$([\beta := \delta, \delta := \gamma \to \gamma], \gamma \to \gamma)$

$\mathcal{W}(\{f : \forall\alpha.\alpha \to \alpha\}, f_1) = (s_1, \tau') \rightsquigarrow s_1 = Id_2, \tau' = \beta \to \beta$

$\mathcal{W}(\{f : \forall\alpha.\alpha \to \alpha\}, f_1) \rightsquigarrow (Id_2, inst(\forall\alpha.\alpha \to \alpha)) \rightsquigarrow (Id_2, \beta \to \beta)$

$\mathcal{W}(\{f : \forall\alpha.\alpha \to \alpha\}\ Id_2, f_2) = (s_2, \tau'') \rightsquigarrow s_2 = Id_3, \tau'' = \gamma \to \gamma$

$s_3 = \mathcal{U}(\{(\beta \to \beta)\ Id_3 = (\gamma \to \gamma) \to \delta\}, \varepsilon) \rightsquigarrow [\beta := \delta, \delta := \gamma \to \gamma]$

$\varnothing \vdash \texttt{let}\ f = \lambda x.x\ \texttt{in}\ f\ f : \gamma \to \gamma$