Nyelvek típusrendszere (IPM-08sztNYTRE, IPM-08EsztNYTRE)

http://people.inf.elte.hu/pgj/nytr_msc/

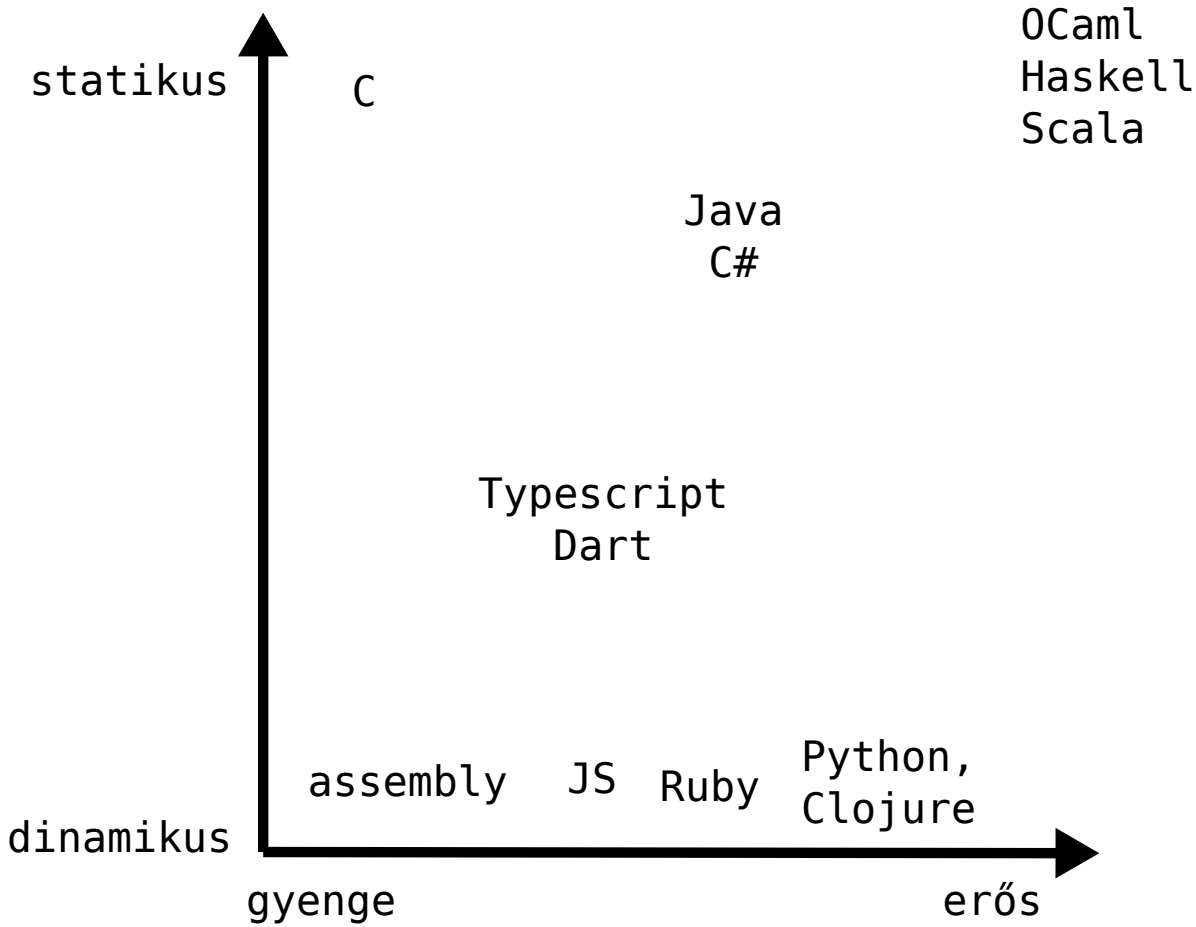
Páli Gábor János
pgj@elte.hu

Eötvös Loránd
Tudományegyetem, Informatikai Kar

Programozási Nyelvek és
Fordítóprogramok Tanszék

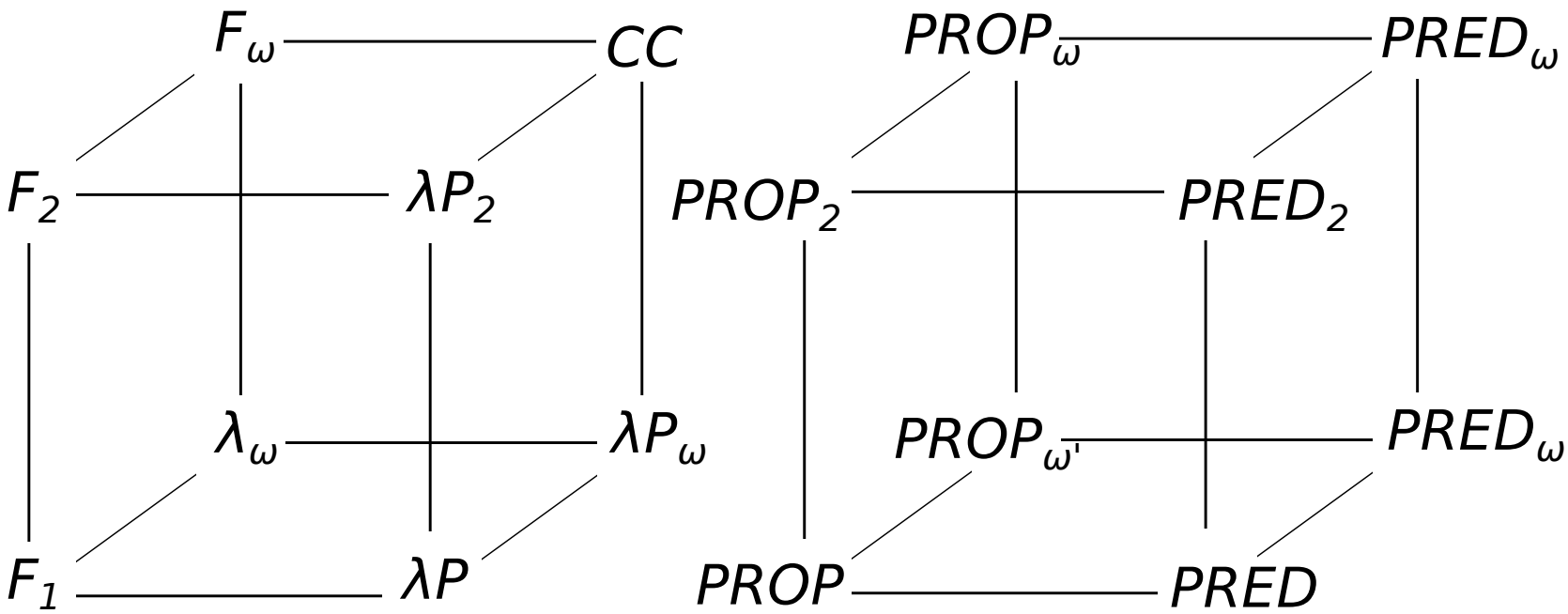
Miről szól ez a tantárgy?

Típusrendszerek



Szabályok gyűjteménye, amelyekkel tulajdonságokat rendelhetünk a program egyes részeihez, hogy így csökkentsük a hibák lehetőségét.

Típuselméletek



A halmazelmélet mint a matematikai alapjának alternatívájaként szolgáló formális rendszer, célja a paradoxonok elkerülése.

Halmazelmélet → típuselmélet: Russell-paradoxon

Nevezzünk egy halmazt *rendes* halmaznak, ha nem tartalmazza önmagát elemként. Legyen R az összes rendes halmaz halmaza:

$$R = \{x \mid x \notin x\}$$

- ha R rendes halmaz, akkor benne van R-ben → nem lehet rendes
- ha R nem rendes halmaz, akkor benne van R-ben → nem lehet rendes

$$R \in R \Leftrightarrow R \notin R$$

A naiv halmazelmélet *inkonzisztens*

Típuselmélet → típusrendszer (Haskell, F_2^+)

```
isEq :: forall a . forall b . (Eq a, Eq b) => a -> b -> Bool
isEq x y = x == y


IsEq.hs:2:17:
    Could not deduce (b ~ a)
    from the context (Eq a, Eq b)
      bound by the type signature for
                 isEq :: (Eq a, Eq b) => a -> b -> Bool
      at IsEq.hs:1:9-38
      ‘b’ is a rigid type variable bound by
          the type signature for isEq :: (Eq a, Eq b) => a -> b -> Bool
          at IsEq.hs:1:9
      ‘a’ is a rigid type variable bound by
          the type signature for isEq :: (Eq a, Eq b) => a -> b -> Bool
          at IsEq.hs:1:9
    Relevant bindings include
      y :: b (bound at IsEq.hs:2:8)
      x :: a (bound at IsEq.hs:2:6)
      isEq :: a -> b -> Bool (bound at IsEq.hs:2:1)
    In the second argument of ‘(==)’, namely ‘y’
    In the expression: x == y
```

Történeti áttekintés

Típustalan rendszerek (1928–1936):

- Kombinátorlogika (Schönfinkel)
- λ-kalkulus (Church)
- Turing-gép (Turing)

1941: Típusos λ-kalkulus (Church)

1958, 1969: Curry-Howard izomorfizmus

1967: Automath (de Bruijn)

1971: Intuicionista (konstruktív) típuselmélet (Martin-Löf)

1972, 1974: System F (Girard, Reynolds)

1991: λ-kocka (Barendregt)

A tematika rövid összefoglalása

- Típus nélküli λ-kalkulus
- Formális típusrendszerek
- Elsőrendű típusos λ-kalkulus – szintaktika, szemantika, kifejezések redukálása, altípus, levezetés, Church- és Curry-típusos kalkulusok
- Másodrendű típusos λ-kalkulus – szintaktika, szemantika, redukció, altípus, levezetés, egzisztenciális típus, rekurzív és függő típusok
- Magasabbrendű típusos λ-kalkulusok. A λ_ω és az $F^\omega_{<:}$ rendszer, típushelyesség, a rendszer elméleti tulajdonságai, Curry-Howard izomorfizmus
- Az $F^\omega_{<:}$ típusrendszer mint a programozási nyelvek alapja
- Az objektumorientált programozás formális leírása

λ-kalkulus

λ-kalkulus: szintaktika

Definíció. ***Egyszerű típusnélküli λ-kifejezések***
Adott egy nem feltétlenül véges, egymástól páronként különböző változókat, a λ jelet, a pontot, valamint a nyitó- és csukózárójeleket tartalmazó halmaz. Ezen mint ábécén értelmezett λ-kifejezések a következőek:

<λ-kifejezés>	*::=*	*<változó>*
	|	*<λ-absztrakció>*
	|	*<applikáció>*
<λ-absztrakció>	*::=*	*(λ<változó>.<λ-kifejezés>)*
<applikáció>	*::=*	*(<λ-kifejezés> <λ-kifejezés>)*

λ: λ-absztraktor, unáris operátor

$E \equiv F$: az E és F λ-kifejezések (szintaktikus) azonossága

Példák: x, $(\lambda x.x)$, $(\lambda x.(\lambda y.x))$, $((\lambda x.(\lambda y.x))z)$

λ-kalkulus: szintaktika

Definíció. *λ-absztrakció*
Ha E egy λ-kifejezés és x egy változó, akkor a

$\lambda x.E$

λ-kifejezést λ-absztrakciónak, az x változót az absztrakció változójának (formális paraméterének), az E kifejezést törzsnek (a paraméter hatókörének) nevezzük.

$\lambda x.(\lambda y.F) \equiv \lambda x.\lambda y.F \equiv \lambda xy.F$ (jobbasszociatív)

Példák:

$I \equiv \lambda x.x$
$K \equiv \lambda xy.x$
$S \equiv \lambda xyz.x\ z\ (y\ z)$

λ-kalkulus: szintaktika

Definíció. ***Applikáció (kombináció)***
Ha E és F λ-kifejezések, akkor az

$E\ F$

λ-kifejezést applikációnak nevezzük. Ha E egy λ-absztrakció, akkor az $E\ F$ neve függvényapplikáció, ahol az F az E λ-kifejezés aktuális paramétere.

$(E\ F)\ G \equiv E\ F\ G$ (balasszociatív)

Példák: $(\lambda xyz.z)\ w$, $(\lambda xy.x)\ z\ w$, $(\lambda x.x)\ (\lambda x.x)$

λ-kalkulus: szintaktika

Definíció. *Szabad változó (FV)*

- *Az x szabad az x kifejezésben.*
- *Az x szabad $\lambda y.E$-ben, ha $x \not\equiv y$ és x szabad E-ben.*
- *Az x szabad E F-ben, ha x szabad E-ben vagy F-ben.*

Definíció. *Kötött változó (BV)*

- *Az x kötött $\lambda y.E$-ben, ha $x \equiv y$ és x szabad E-ben.*
- *Az x kötött $\lambda y.E$-ben, ha x kötött E-ben.*
- *Az x kötött E F-ben, ha x kötött E-ben vagy F-ben.*

Példák:

$E \equiv \lambda x.(\lambda x.x\ y),\ FV(E) = \{y\}, BV(E) = \{x\}$
$F \equiv \lambda y.(\lambda x.x\ y),\ FV(F) = \varnothing, BV(F) = \{x, y\}$
$G \equiv \lambda x.(\lambda y.x\ y),\ FV(G) = \varnothing, BV(G) = \{x, y\}$
$H \equiv \lambda y.(\lambda y.x\ y),\ FV(H) = \{x\}, BV(H) = \{y\}$

λ-kalkulus: szintaktika

Definíció. *Zárt λ-kifejezés (kombinátor)*
Ha egy λ-kifejezésben nincs szabad változó, akkor a λ-kifejezést zártnak nevezzük. Ha egy λ-kifejezés zárt, akkor vagy egy zárt λ-absztrakció, vagy zárt λ-kifejezések applikációja.

Példák: $\lambda x.x$, $\lambda xx.x$, $\lambda xy.x\ y$, $(\lambda x.x)\ (\lambda xy.x)$

Definíció. *Nyitott λ-kifejezés*
A nem zárt λ-kifejezések a nyitott λ-kifejezések.

Példák: $\lambda x.y$, $x\ y$, $(\lambda x.x)\ y$

A zártság a kiértékelhetőség előfeltétele.

Definíció. *λ-kifejezés lezárása*
Ha $FV(E) = \{x_1, x_2, \ldots, x_n\}$, akkor a $\lambda x_1 x_2 \ldots x_n.E$ kifejezést az E egy lezárásának nevezzük.

λ-kalkulus: szintaktika

Definíció. ***Helyettesítés***

$$
\begin{array}{lcl}
x[y := G] & \equiv & \begin{cases} G, & \text{ha } x \equiv y \\ x & \text{egyébként} \end{cases} \\
(E\ F)[y := G] & \equiv & (E\,[\,y := G\,])\ (F\,[\,y := G\,]) \\
(\lambda x.E)[y := G] & \equiv & \begin{cases} \lambda x.E, & \text{ha } x \equiv y \\ \lambda x.E[y := G], & \text{ha } x \not\equiv y \text{ és } x \notin FV(G) \\ \lambda x.E & \text{egyébként} \end{cases}
\end{array}
$$

$(E[x := F])[y := G] \equiv E[x := F][y := G]$ (balasszociatív)

Példák:

$(\lambda x.y)[x := \lambda x.y] \equiv \lambda x.y$
$(\lambda x.y)[y := \lambda x.y] \equiv \lambda x.\lambda x.y \equiv \lambda xx.y$
$(\lambda x.y)[y := \lambda y.x] \equiv \lambda x.y$
$(\lambda x.y)[y := \lambda x.x] \equiv \lambda x.\lambda x.x \equiv \lambda xx.x$
$(\lambda x.y)[y := \lambda y.y] \equiv \lambda x.\lambda y.y \equiv \lambda xy.y$

λ-kalkulus: szintaktika

Lemma. ***Az*** $E[x := F]$ ***szabad változói***
Ha $x \in FV(E)$, *akkor* $FV(E[x := F]) = (FV(E) \setminus \{x\}) \cup FV(F)$.

Lemma. ***Egyszerű helyettesítések (1)***

$$\begin{array}{lcl} E[x := x] & \equiv & E \\ E[x := F] & \equiv & E, \text{ ha } x \notin FV(E) \end{array}$$

Lemma. ***Egyszerű helyettesítések (2)***
Ha $BV(E) \cap (FV(F) \cup FV(G) \cup \{y\}) = \varnothing$, *akkor*

$$\begin{array}{lcl} (E[x := y])[y := F] & \equiv & E[x := F], \text{ ha } y \notin FV(E) \\ (E[x := y])[y := x] & \equiv & E, \text{ ha } y \notin FV(E) \\ (E[x := F])[x := G] & \equiv & E[x := F[x := G]] \end{array}$$

Lemma. ***Helyettesítési lemma***
Ha $x \not\equiv y$ *és* $x \notin FV(G)$, *akkor*
$E[x := F][y := G] \equiv E[y := G][x := F[y := G]]$.

λ-kalkulus: operációs szemantika

Definíció. ***β-redukció***
Ha az $E[x := F]$ kifejezésben az F szabad változói nem válnak az E kötött változóivá, akkor $(\lambda x.E)F \to_\beta E[x := F]$.

Példák:

$(\lambda x.x)\ y \to_\beta y$
$(\lambda x.y)\ z \to_\beta y$
$(\lambda y.(\lambda x.y\ x))\ u\ y \to_\beta (\lambda x.u\ x)\ y \to_\beta u\ y$
$(\lambda f.f\ u)\ (\lambda x.x\ y) \to_\beta (\lambda x.x\ y)\ u \to_\beta u\ y$
$(\lambda x.(\lambda x.x\ y)\ x)\ u \to_\beta (\lambda x.x\ y)\ u \to_\beta u\ y$
$(\lambda xy.x\ y)\ y \not\to_\beta \lambda y.y\ y$ (névkonfliktus)
$(\lambda xy.x)\ y \not\to_\beta \lambda y.y$ (névkonfliktus)

λ-kalkulus: operációs szemantika

Definíció. α-konverzió
Ha az E kifejezésben y nem szabad változó, azaz $y \notin FV(E)$, akkor $\lambda x.E \leftrightarrow_\alpha \lambda y.E[x := y]$.

Módosított helyettesítés:

$$(\lambda x.E)[y := G] \quad \equiv \quad \begin{cases} \lambda x.E, \text{ha } x \equiv y \\ \lambda x.E[y := G], \text{ha } x \not\equiv y \text{ és } x \notin FV(G) \\ \leftrightarrow_\alpha \lambda z.E[x := z][y := G], \\ \quad \text{ha } x \not\equiv y \text{ és } x \in FV(G), z \text{ egy új változó} \end{cases}$$

Példák:

$(\lambda xy.x\ y)\ y \leftrightarrow_\alpha (\lambda xz.x\ z)\ y \rightarrow_\beta \lambda z.y\ z$
$(\lambda xy.x)\ y \leftrightarrow_\alpha (\lambda xz.x)\ y \rightarrow_\beta \lambda z.y$

λ-kalkulus: operációs szemantika

Definíció. η-*konverzió*
Ha az x az E kifejezésnek nem szabad változója, azaz $x \notin FV(E)$, akkor $\lambda x.E\ x \leftrightarrow_\eta E$.

Valamint:

$E\ F \leftrightarrow_\eta (\lambda x.E\ x)\ F$

Példák:

$\lambda x.y\ x \leftrightarrow_\eta y$
$\lambda x.(y\ z)\ x \leftrightarrow_\eta y\ z$
$\lambda x.(\lambda x.x\ y)\ x \leftrightarrow_\eta \lambda x.x\ y$
$\lambda xx.x\ y\ y\ z\ x \leftrightarrow_\eta \lambda x.x\ y\ y\ z$
$\lambda x.x\ x \not\leftrightarrow_\eta x$

λ-kalkulus: normálforma

Definíció. *Normálforma*
Ha egy λ-kifejezésben nincs elvégezhető redukció (redukálható kifejezés), akkor λ-kifejezés normálformában van.

Egy λ-kifejezés akkor és csak akkor van normálformában, ha

$\lambda x_1 x_2 \ldots . x_m . M_1 \ M_2 \ \ldots \ M_n$, $(m, n \geq 0)$

alakú, ahol $M_1, M_2, \ldots, M_n$ mindegyike normálformában van.

Példák:

x
$x \ y$
$\lambda x.y$
$\lambda x.(\lambda y.z)$
$\Omega \equiv (\lambda x.x \ x) \ (\lambda x.x \ x)$ (nincs)

λ-kalkulus: normálforma

Tétel. ***I. Church-Rosser-tétel (rombusztulajdonság)***
Ha $E_1 = E_2$, akkor létezik egy olyan F λ-kifejezés, amelyre $E_1 \to_* F$ és $E_2 \to_* F$.

Következmények.

- *Ha $E_1 = E_2$, és E_2 normálformában van, $E_1 \to_* E_2$.*
- *Minden λ-kifejezésnek legfeljebb egy normálformája van.*
- *Ha E és F mindegyike normálforma, és $E \not\equiv F$, akkor $E \neq F$.*

Példák:

$$\underline{(\lambda x.((\lambda y.x\ y)\ ((\lambda z.z)\ v)))\ u} \to_\beta \underline{(\lambda y.u\ y)\ ((\lambda z.z)\ v)} \to_\beta u\ \underline{((\lambda z.z)\ v)} \to_\beta u\ v$$

$$(\lambda x.((\lambda y.x\ y)\ \underline{((\lambda z.z)\ v)}))\ u \to_\beta \underline{(\lambda x.((\lambda y.x\ y)\ v)}\ u \to_\beta \underline{(\lambda x.(x\ v))\ u} \to_\beta u\ v$$

λ-kalkulus: normálforma

Redukálási stratégia: meghatározza, több redukció esetén melyiket kell elvégezni, redukálási sorrendet ad meg. Például:

- legbaloldalibb legkülső (normál, lusta):
  $\underline{((\lambda p.p)\ ((\lambda q.q)\ r))}\ s\ ((\lambda t.t)\ ((\lambda u.u)\ v))$
- legbaloldalibb legbelső (applikatív, mohó):
  $((\lambda p.p)\ (\underline{(\lambda q.q)\ r}))\ s\ ((\lambda t.t)\ ((\lambda u.u)\ v))$
- legjobboldalibb legkülső:
  $((\lambda p.p)\ ((\lambda q.q)\ r))\ s\ (\underline{(\lambda t.t)\ ((\lambda u.u)\ v)})$
- legjobboldalibb legbelső:
  $((\lambda p.p)\ ((\lambda q.q)\ r))\ s\ ((\lambda t.t)\ (\underline{(\lambda u.u)\ v}))$

Normalizáló stratégia: ha létezik normálforma, megkapjuk.

Tétel. ***II. Church-Rosser-tétel (normalizálás)***
A normál sorrendű redukálási stratégia normalizáló redukálási stratégia.

Típusrendszer

Célkitűzés: helyesség (soundness) és biztonságosság (safety)

Milner (1978): *"Well-typed programs cannot go wrong"*

- Típusmegmaradás tétele (type preservation)
- Haladás tétele (progression)

Church-típusos rendszerek: *explicit* típus*ellenőrzés*
(type checking)

Curry-típusos rendszerek: *implicit* típus*kikövetkeztetés*
(type inference)

Formális típusrendszerek

Formális matematikai rendszer

Definíció. ***Formális matematikai rendszer***
Legyen $\Phi = (\Sigma, \mathcal{F}, \mathcal{A}, \mathcal{R})$ *egy formális matematikai rendszer, ahol*

- Σ *egy ábécé,*
- $\mathcal{F}$ *a nyelvtani szabályok halmaza,*
- $\mathcal{A}$ *az axiómák halmaza,*
- $\mathcal{R}$ *a levezetési szabályok halmaza.*

Formális matematikai rendszer

Definíció. ***Formális matematikai rendszer***
Legyen $\Phi = (\Sigma, \mathcal{F}, \mathcal{A}, \mathcal{R})$ *egy formális matematikai rendszer, ahol*

- Σ *egy ábécé,*
  A Σ elemei az alapszimbólumok: változók, elválasztó jelek, operátorok. Az alapszimbólumokból képzett véges sorozatok a formulák.
- $\mathcal{F}$ *a nyelvtani szabályok halmaza,*
  A nyelvtani szabályokat kielégítő formulák jól formáltak.
- $\mathcal{A}$ *az axiómák halmaza,*
- $\mathcal{R}$ *a levezetési szabályok halmaza.*
  Az axiómák jelentéssel bíró (szemantikailag helyes) formulák, a többi formulát ezekből kell tudni levezetni a szabályok segítségével.

Formális típusrendszer: jelölések

$E, F, \ldots$	(λ-)kifejezés
$A, B, \ldots$	típus(kifejezés) (F_1)
$\tau, \sigma, \ldots$	típus(kifejezés)
$x, y, \ldots$	változó
$\alpha, \beta, \ldots$	típusváltozó
$\Gamma, \Delta, \ldots$	(típus)környezet, kontextus

$E : A$	E típusa A
$\Gamma \vdash A$	A egy jól formált típus
$\Gamma \vdash E : A$	E típusa A a Γ környezetben
$E \equiv F$	E és F egyenlőek

Formális típusrendszer: jól formáltság

Definíció. ***Jól formált λ-kifejezés***
Az E λ-kifejezés jól formált, ha nyelvtanilag helyes, azaz megfelel a λ-kifejezéseket leíró nyelvtan szabályainak.

Definíció. ***Jól formált típus***
Az A típus jól formált, ha nyelvtanilag helyes, azaz megfelel a típusokat leíró nyelvtan szabályainak.

Egy jól formált kifejezésre a típusok jól formáltságának megkövetelése viszont önmagában még nem elég.

Formális típusrendszer: típuskörnyezet

Definíció. ***Típuskörnyezet (szintaktika)***

<típuskörnyezet> ::= $\varnothing$
| *<környezet>, <változó> : <típus>*

valamint ha x_i egy A_i típusú változó $(1 \leq i \leq n)$ *és a típuskörnyezet egy*

$$\Gamma = \{x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n\}$$

halmaz, akkor $x_i \neq x_j$ $(i \neq j)$.

$$dom(\Gamma) = \{x_1, x_2, \ldots\}$$

Definíció. ***Jól formált típuskörnyezet***
Egy Γ *típuskörnyezet jól formált, ha megfelel a típuskörnyezet definíciójában megadott nyelvtannak.*

Formális típusrendszer: következtetés

A következtetések általános alakja:

$$\Gamma \vdash \mathcal{I}$$

ahol Γ egy (statikus) típuskörnyezet, $\mathcal{I}$ egy állítás.

Példák:

$\varnothing \vdash \mathsf{wf}$	$\varnothing$ jól formált
$\Gamma \vdash \mathsf{wf}$	Γ jól formált
$\Gamma \vdash A$	A egy jól formált típus
$\Gamma \vdash E : A$	E típusa A a Γ környezetben

Formális típusrendszer: következtetési szabály

A következtetési szabályok általános alakja:

$$\frac{\Gamma_1 \vdash \mathcal{I}_1, \ldots, \Gamma_n \vdash \mathcal{I}_n}{\Gamma \vdash \mathcal{I}} \text{ (SZABÁLYNÉV)}$$

ahol

- $\Gamma_1 \vdash \mathcal{I}_1, \ldots, \Gamma_n \vdash \mathcal{I}_n$: feltételek
- $\Gamma \vdash \mathcal{I}$: következmény

Axiómák: A feltételek halmaza üres.

$$\frac{}{\varnothing \vdash \mathsf{wf}} \text{ (ENV } \varnothing\text{)}$$

Ha a feltételek teljesülnek, akkor a következmény is teljesül.

Formális típusrendszer: levezetés

A következtetésekből levezetési fákat építhetünk.

$$\dfrac{\dfrac{K_{11}}{\dfrac{K_{12}}{\dfrac{K_{22}}{K_{32}}\,(S3)}\,(S2)}\,(S1)}{\dfrac{\dfrac{K_{41}}{\dfrac{K_{42}}{K_{52}}\,(S5)}\,(S4)\qquad\dfrac{K_{61}}{K_{62}}\,(S6)}{K_{72}}\,(S7)}$$

Formális típusrendszer: érvényes következtetés

Definíció. ***Érvényes következtetés***
Azt mondjuk, hogy a $\Gamma \vdash E : A$ *következtetés érvényes, ha létezik olyan (típus)levezetés, ahol a következtetés a levezetéshez tartozó (következtetési) fa gyökere.*

Definíció. ***Jól típusozott kifejezés***
Ha egy típusrendszerben a $\Gamma \vdash E : A$ *következtetés érvényes, akkor azt mondjuk, hogy az E kifejezés jól típusozott.*

Jól típusozott kifejezés = típusozható kifejezés