Nyelvek típusrendszere (jegyzet)

Kaposi Ambrus
Eötvös Loránd Tudományegyetem
akaposi@inf.elte.hu

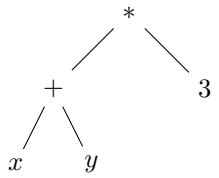
2016. szeptember 15.

1. Bevezető

Típusrendszerek helye a fordítóprogamokban:

1. Lexikális elemző (lexer): karakterek listájából tokenek listáját készíti, pl. `"(x + y) * 3"`-ból $\mathsf{brOpen}\,\mathsf{var}[x]\,\mathsf{op}[+]\,\mathsf{var}[y]\,\mathsf{brClose}\,\mathsf{op}[*]\,\mathsf{const}[3]$.

2. Szintaktikus elemző (parser): a tokenek listájából absztrakt szintaxisfát épít, az előbbi példából a következőt.



3. Típusellenőrzés: megnézi, hogy a szintaxisfa megfelel -e bizonyos szabályoknak (a típusrendszernek), pl. szükséges, hogy x és y numerikus típusú legyen. Általánosságban, nem enged át értelmetlen programokat. Mi ezzel a résszel foglalkozunk.

4. Kódgenerálás: ha a típusellenőrző átengedte a szintaxisfát, akkor ezt lefordítjuk a gép számára érthető kóddá.

Típusrendszerek és logika:

– Mi az, hogy számítás? Erre egy válasz a lambda kalkulus (alternatíva a Turing-gépekre), amely a Lisp programozási nyelv alapja. A különböző típusrendszereket azért fejlesztették ki, hogy megszorítsák a programokat a jó programokra.

– Mi az, hogy bizonyítás? Erre válaszol a logika az ítéletlogikával, predikátumkalkulussal.

– Típuselmélet: egy olyan rendszer, mely a két nézőpontot egyesíti. Ez egy típusrendszer, melynek speciális eseteit fogjuk tanulni. Megmutatja, hogy az intuitívan hasonlóan viselkedő fogalmak (pl. matematikai függvények,

logikai következtetés, függvény típusú program, univerzális kvantor) valójában ugyanazok. Elvezet az egyenlőség egy új nézőpontjára, ami egy új kapcsolatot nyit a homotópia-elmélet és a típuselmélet között.

A magas kifejezőerejű típusrendszerekben már nem az a kérdés, hogy egy megírt program típusozható -e. A nézőpont megfordul: először adjuk meg a típust (specifikáljuk, hogy mit csinálhat a program), majd a típus alapján megírjuk a programot (ez akár automatikus is lehet, ha a típus eléggé megszorítja a megírható programok halmazát).

Záróvizsga tematika:

- Az elsőrendű Church- és Curry-típusrendszer; szintaktika, operációs szemantika, következtetési szabályok, alaptípusok,
- az altípus és tulajdonságai.
- Típuskikövetkeztetési módszerek,
- letpolimorfizmus, Hindley-Milner és Milner-Mycroft algoritmusok, korlátozások generálása és egyesítés.
- Polimorfikus típusos és magasabbrendű típusrendszerek és tulajdonságaik;
- az F_2 típusrendszer és általánosításai,
- az $\mathsf{F}_{\omega<}$ típusrendszer.

Két könyv alapján haladunk, de nem szükséges egyik sem a tananyag elsajátításához.

- Robert Harper. Practical Foundations for Programming Languages. Cambridge University Press, 2012.
- Csörnyei Zoltán. Bevezetés a típusrendszerek elméletébe. ELTE Eötvös Kiadó, 2012.

2. Induktív definíciók

2.1. Absztrakt szintaxisfák

Absztrakt szintaxisfának (AST, abstract syntax tree) nevezzük az olyan fákat, melyek leveleinél *változók* vannak, közbenső pontjaikon pedig *operárotok*. Például a természetes szám kifejezések és az ezekből és összeadásból álló kifejezések AST-it az alábbi definíciókkal adhatjuk meg.

$$\begin{aligned} n \in \mathsf{Nat} &::= i \mid \mathsf{zero} \mid \mathsf{suc}\, n \\ e, e' \in \mathsf{Exp} &::= x \mid \mathsf{num}\, n \mid e + e' \end{aligned}$$

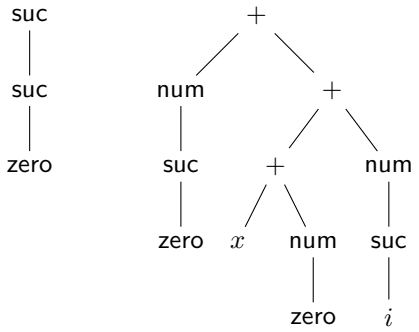
Nat-ot és Exp-et *fajtának* nevezzük. A Nat fajtájú AST-ket n-el jelöljük. Nat fajtájú AST lehet egy i változó, vagy létre tudjuk hozni a nulláris zero operátorral (*aritása* ()Nat) vagy az unáris suc operátorral (aritása (Nat)Nat). n egy tetszőleges Nat fajtájú AST-t jelöl, míg i maga egy Nat fajtájú AST, mely egy darab változóból áll.

Az Exp fajtájú AST-ket e-vel és ennek vesszőzött változataival jelöljük, az Exp fajtájú változókat x-el jelöljük. Exp fajtájú AST-t egy unáris operátorral (num, aritása $(\mathsf{Nat})\mathsf{Exp}$) és egy bináris operátorral ($+$, aritása $(\mathsf{Exp}, \mathsf{Exp})\mathsf{Exp}$) tudunk létrehozni. A num operátorral Nat fajtájú AST-ket tudunk kifejezésekbe beágyazni.

Minden fajtához változóknak egy külön halmaza tartozik, ezért jelöljük őket különböző betűkkel. A változók halmaza végtelen (mindig tudunk *friss* változót kapni, olyat, amilyet még sehol nem használtunk) és eldönthető, hogy két változó egyenlő -e. Az előbbi két fajtához tartozó változók halmazát így adhatjuk meg.

$$i, i', i_1 ... \in \mathsf{Var}_{\mathsf{Nat}}$$
$$x, x', x_1, ... \in \mathsf{Var}_{\mathsf{Exp}}$$

Az AST-ket lerajzolhatjuk, pl. $\mathsf{suc}\,(\mathsf{suc}\,\mathsf{zero})$ és $\mathsf{num}\,(\mathsf{suc}\,\mathsf{zero})+\big((x+\mathsf{num}\,\mathsf{zero})+ + \mathsf{num}\,(\mathsf{suc}\,i)\big)$.



Az $x + \mathsf{num}\,\mathsf{zero}$ *részfája* az $(x + \mathsf{num}\,\mathsf{zero}) + \mathsf{num}\,(\mathsf{suc}\,i)$ AST-nek, ami pedig részfája a teljes AST-nek.

Az olyan AST-ket, melyek nem tartalmaznak változót, *zártnak* nevezzük, egyébként *nyíltnak*. A változók értékét *helyettesítéssel* (substitution) adhatjuk meg. Ha a egy A fajtájú AST, x pedig egy A fajtájú változó, t pedig tetszőleges fajtájú AST, akkor $t[x \mapsto a]$-t úgy kapjuk meg t-ből, hogy x összes előfordulása helyére a-t helyettesítünk. A helyettesítést az alábbi módon adjuk meg (f egy n-paraméteres operátor).

$$\begin{aligned}
x[x \mapsto a] &:= a \\
y[x \mapsto a] &:= y && \text{feltéve, hogy } x \neq y \\
(\mathsf{f}\, t_1 \ldots t_n)[x \mapsto a] &:= \mathsf{f}\,(t_1[x \mapsto a]) \ldots (t_n[x \mapsto a])
\end{aligned}$$

Néhány példa:

$$(x + \mathsf{num}\,\mathsf{zero})[x \mapsto \mathsf{num}\,(\mathsf{suc}\,\mathsf{zero})] = \mathsf{num}\,(\mathsf{suc}\,\mathsf{zero}) + \mathsf{num}\,\mathsf{zero}$$
$$(x + x)[x \mapsto x' + \mathsf{num}\,\mathsf{zero}] = (x' + \mathsf{num}\,\mathsf{zero}) + (x' + \mathsf{num}\,\mathsf{zero})$$
$$(x + \mathsf{num}\,(\mathsf{suc}\,i))[i \mapsto \mathsf{zero}] = x + \mathsf{num}\,(\mathsf{suc}\,\mathsf{zero})$$

Ha szeretnénk bizonyítani, hogy egy állítás az összes adott fajtájú AST-re igaz, elég megmutatni, hogy a változókra igaz az állítás, és az összes operátor megtartja az állítást. Pl. ha minden $n \in \mathsf{Nat}$-ra szeretnénk belátni, hogy $\mathcal{P}(n)$, akkor elég azt belátni, hogy $\mathcal{P}(i)$ minden i-re, hogy $\mathcal{P}(\mathsf{zero})$ és hogy minden m-re $\mathcal{P}(m)$-ből következik $\mathcal{P}(\mathsf{suc}\,m)$. Ha minden $e \in \mathsf{Exp}$-re szeretnénk $\mathcal{Q}(e)$-t belátni,

elég azt belátni, hogy minden x-re $\mathcal{Q}(x)$, hogy $n \in \mathsf{Nat}$-ra igaz $\mathcal{Q}(\mathsf{num}\, n)$ és ha igaz $\mathcal{Q}(e)$ és $\mathcal{Q}(e')$, akkor igaz $\mathcal{Q}(e + e')$ is. Ezt az érvelési formát strukturális *indukciónak* nevezzük.

2.1. Feladat. *Végezzük el a következő helyettesítéseket:*

$$\big((x + x')[x \mapsto x']\big)[x' \mapsto x]$$
$$(x + x)[x \mapsto x' + x]$$
$$\big(\mathsf{suc}\,(\mathsf{suc}\, i)\big)[i \mapsto \mathsf{suc}\, i']$$
$$\big(\mathsf{num}\,(\mathsf{suc}\, i)\big)[x \mapsto \mathsf{num}\,\mathsf{zero}]$$

2.2. Feladat. *Adjuk meg a* Nat*-ok listájának fajtáját az operátorok aritásával együtt.*

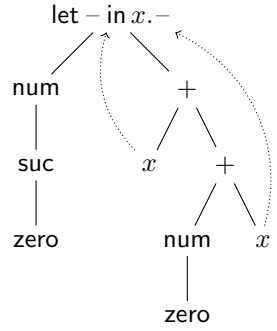
$$xs \in \mathsf{List}_{\mathsf{Nat}} ::= ?$$

További információ:

- A különböző fajtájú AST-k halmazait finomíthatjuk aszerint, hogy milyen változók vannak bennük. Ha egy Exp fajtájú e AST-ben x, x' a szabad változók, azt mondjuk, hogy $e \in \mathsf{Exp}_{x,x'}$ halmazban van. Ekkor pl. $e + x'' \in \in \mathsf{Exp}_{x,x',x''}$ és $e[x \mapsto \mathsf{num}\,\mathsf{zero}] \in \mathsf{Exp}_{x'}$. Általános esetben szükségünk van minden fajtához egy különböző változó-halmazra, így ha $a \in A_{X_A \ldots X_B}$ és $b \in B_{Y_A, y \ldots Y_B}$, akkor $b[y \mapsto a] \in B_{X_A \cup Y_A \ldots X_B \cup Y_B}$.
- Az ABT-ket polinomiális funktorok (polynomial functor, container) szabad monádjaiként (free monad) adjuk meg. Ezzel biztosítjuk, hogy pl. minden szintaxisfa véges.

2.2. Absztrakt kötéses fák

Az *absztrakt kötéses fák* (ABT, abstract binding tree) az AST-khez hasonlók, de változót *kötő* operátorok is szerepelhetnek benne. Ilyen például a $\mathsf{let}\, e\, \mathsf{in}\, x.e'$, mely azt fejezi ki, hogy e'-ben az x előfordulásai e-re hivatkoznak. Azt mondjuk, hogy az x változó kötve van az e' kifejezésben. A let operátor aritását $(\mathsf{Exp}, \mathsf{Exp}.\mathsf{Exp})\mathsf{Exp}$-el jelöljük, az operátor második paraméterében köt egy Exp fajtájú változót. Pl. $\mathsf{let}\,\mathsf{num}\,(\mathsf{suc}\,\mathsf{zero})\,\mathsf{in}\, x.x + (\mathsf{num}\,\mathsf{zero} + x)$ kifejezésben a $+$ operátor x paraméterei a kötött x-re vonatkoznak. Ezt a következőképp ábrázolhatjuk. A felfele mutató szaggatott nyilak mutatják, hogy az x változók melyik kötésre mutatnak. A pont után szereplő $x + (\mathsf{num}\,\mathsf{zero} + x)$ részkifejezést az x változó *hatáskörének* nevezzük.



A let-tel kiegészített Exp fajtájú ABT-ket a következő jelöléssel adjuk meg. A pont jelöli, hogy hol kötünk változókat.

$$e, e' \in \mathsf{Exp} ::= x \mid \mathsf{num}\, n \mid e + e' \mid \mathsf{let}\, e\, \mathsf{in}\, x.e'$$

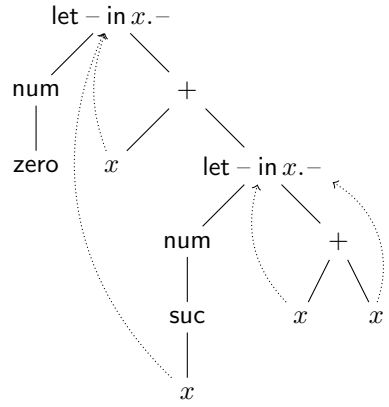
A $\mathsf{let\,num\,zero\,in}\, x.x + x$ ABT $x + x$ részfája tartalmaz egy *szabad változót*, az x-et, emiatt az $x + x$ ABT-t *nyíltnak*, míg a $\mathsf{let\,zero\,in}\, x.x + x$ ABT-t *zártnak* nevezzük. Az alábbi Exp fajtájú ABT-k kötött változóit aláhúztuk, szabad változóit felülhúztuk.

$$\mathsf{let\,num\,zero\,in}\, x.\underline{x} + \underline{x}$$
$$\mathsf{let}\, \overline{y}\, \mathsf{in}\, x.\underline{x} + \underline{x}$$
$$\mathsf{let}\, \overline{y}\, \mathsf{in}\, x.\underline{x} + \mathsf{let}\, \underline{x}\, \mathsf{in}\, z.\underline{z}$$
$$\mathsf{let}\, \overline{y}\, \mathsf{in}\, x.\underline{x} + \mathsf{let}\, \overline{z}\, \mathsf{in}\, z.\underline{z}$$

A kötött változók csak pozíciókra mutatnak, a nevük nem érdekes. Például a $\mathsf{let\,num\,zero\,in}\, x.x + x$ és a $\mathsf{let\,num\,zero\,in}\, y.y + y$ ABT-k megegyeznek (α-konvertálhatónak vagy α-ekvivalensnek szokás őket nevezni). A szabad változókra ez nem igaz, pl. $x + x \neq y + y$.

Megjegyezzük, hogy bár a jelentése (szemantikája) a $\mathsf{let}\, e\, \mathsf{in}\, x.x + x$-nek az, hogy x helyére e-t írunk, ez a kifejezés nem egyenlő az $e + e$ kifejezéssel.

Ha többször ugyanazt a változót kötjük egy ABT-ben, az újabb kötés *elfedi* az előzőt. Pl. $\mathsf{let\,num\,zero\,in}\, x.x + \big(\mathsf{let\,num}\, (\mathsf{suc}\, x)\, \mathsf{in}\, x.x + x\big)$-ben az $x + x$-ben levő x-ek a második kötésre (ahol $\mathsf{num}\, (\mathsf{suc}\, x)$-et adtunk meg) mutat (a $\mathsf{num}\, (\mathsf{suc}\, x)$-ben levő x viszont az első kötésre mutat).



Az elfedés megszüntethető a változónevek átnevezésével: $\mathsf{let\,num\,zero\,in}\, y.y + \\ + \big(\mathsf{let\,num}\, (\mathsf{suc}\, y)\, \mathsf{in}\, x.e\big)$. Ebben az ABT-ben már hivatkozhatunk az e részfában az x-re is meg a külső y-ra is.

Helyettesíteni tudunk ABT-kben is, pl. szeretnénk a következő egyenlőségeket.

$$(\mathsf{let}\, x\, \mathsf{in}\, x'.x' + x'')[x'' \mapsto \mathsf{num\,zero}] = \mathsf{let}\, x\, \mathsf{in}\, x'.x' + \mathsf{num\,zero}$$
$$(\mathsf{let}\, x\, \mathsf{in}\, x'.x' + x'')[x' \mapsto \mathsf{num\,zero}] = \mathsf{let}\, x\, \mathsf{in}\, x'.x' + x''$$
$$(\mathsf{let}\, x\, \mathsf{in}\, x'.x' + x'')[x'' \mapsto x'] = \mathsf{let}\, x\, \mathsf{in}\, x'''.x''' + x'$$

Az első esetben egyszerűen behelyettesítünk az x'-t kötő művelet alatt. A második esetben, mivel a kötés elfedi az x' változót, a kötés hatáskörében levő x'-k

mind a kötésre vonatkoznak, ezért nem történik semmi. A harmadik eset érdekesebb: itt azáltal, hogy a kötés alá mentünk, naivan csak lecserélnénk az x''-t x'-re, de ezzel az x' kötötté válna, és nem a „külső" x'-re, hanem a kötöttre vonatkozna, megváltoztatva ezzel az ABT jelentését. Emiatt a kötésre egy másik, még nem használt változót, x'''-t használjuk.

Általánosságban a következőképp tudjuk megadni a helyettesítést egy f operátorra, mely az első paraméterében köt.

$$(\mathsf{f}\, y.t\, t_1 \dots t_n)[x \mapsto a] := \mathsf{f}\left(\big(z.(t[y \mapsto z])\big)[x \mapsto a]\right) (t_1[x \mapsto a]) \dots (t_n[x \mapsto a])$$
$$\text{(ahol } z \text{ friss változónév)}$$

A biztonság kedvéért (lásd a fenti harmadik példát), a kötött változót átnevezzük egy friss z változóra, és csak ezután a helyettesítés után helyettesítjük a megmaradt x-eket a-val.

Ha szeretnénk strukturális indukcióval egy $\mathcal{Q}$ állítást a let-tel kiegészített Exp ABT-kről bizonyítani, a következőket kell belátnunk:

- minden x változóra $\mathcal{Q}(x)$,
- minden $n \in \mathsf{Nat}$-ra $\mathcal{Q}(n)$,
- minden olyan $e, e' \in \mathsf{Exp}$-re, melyekre $\mathcal{Q}(e)$ és $\mathcal{Q}(e')$, igaz, hogy $\mathcal{Q}(e + e')$,
- minden $e, e' \in \mathsf{Exp}$-re és x változóra, ha $\mathcal{Q}(e)$ és $\mathcal{Q}(e')$, akkor $\mathcal{Q}(\mathsf{let}\, e\, \mathsf{in}\, x.e')$.

2.3. Feladat. *Húzd alá a kötött változókat és fölé a szabad változókat! A kötött változóknál rajzolj egy nyilat, hogy melyik kötésre mutatnak!*

$$x + \mathsf{num}\, i$$
$$\mathsf{let}\, x\, \mathsf{in}\, x.\mathsf{num}\, i + x$$
$$\mathsf{let}\, x\, \mathsf{in}\, x.x' + \mathsf{let}\, x\, \mathsf{in}\, x'.x' + x$$
$$\mathsf{let}\, x\, \mathsf{in}\, x.x + \mathsf{let}\, x\, \mathsf{in}\, x.x + x$$
$$\mathsf{let}\, x\, \mathsf{in}\, x.(\mathsf{let}\, x\, \mathsf{in}\, x'.x'' + x') + \mathsf{let}\, x'\, \mathsf{in}\, x'.x' + x''$$

2.4. Feladat. *Lehet -e egy $i \in \mathsf{Var}_{\mathsf{Nat}}$ egy Exp fajtájú ABT-ben kötött?*

2.5. Feladat. *Adj algoritmust arra, hogy két ABT mikor egyenlő általános esetben. Az érdekes rész annak eldöntése, hogy egy $\mathsf{f}\, x.t\, t_1 \dots t_n$ alakú és egy $\mathsf{f}\, x'.t'\, t'_1 \dots t'_n$ alakú ABT egyenlő -e.*

2.6. Feladat. *Végezd el a következő helyettesítéseket!*

$$(\mathsf{let}\, x\, \mathsf{in}\, x.x + x)[x \mapsto \mathsf{num}\, \mathsf{zero}]$$
$$(\mathsf{let}\, x'\, \mathsf{in}\, x.x' + x)[x' \mapsto \mathsf{num}\, \mathsf{zero}]$$
$$(\mathsf{let}\, x'\, \mathsf{in}\, x.x' + x')[x' \mapsto x]$$

2.7. Feladat. *Döntsd el, hogy a következő* Exp *fajtájú ABT-k megegyeznek -e!*

$$x + \mathsf{num}\, i \stackrel{?}{=} x + \mathsf{num}\, i'$$

$$\mathsf{let}\, x\, \mathsf{in}\, x.\mathsf{num}\, i + x \stackrel{?}{=} \mathsf{let}\, x\, \mathsf{in}\, x'.\mathsf{num}\, i + x'$$

$$\mathsf{let}\, x\, \mathsf{in}\, x.\mathsf{num}\, i + x \stackrel{?}{=} \mathsf{let}\, x\, \mathsf{in}\, x.\mathsf{num}\, i + x'$$

$$\mathsf{let}\, x\, \mathsf{in}\, x.x' + \mathsf{let}\, x\, \mathsf{in}\, x'.x' + x \stackrel{?}{=} \mathsf{let}\, x\, \mathsf{in}\, x'.x + \mathsf{let}\, x'\, \mathsf{in}\, x.x + x'$$

$$\mathsf{let}\, x\, \mathsf{in}\, x.x' + \mathsf{let}\, x\, \mathsf{in}\, x'.x' + x \stackrel{?}{=} \mathsf{let}\, x\, \mathsf{in}\, x''.x' + \mathsf{let}\, x''\, \mathsf{in}\, x.x + x''$$

$$\mathsf{let}\, x\, \mathsf{in}\, x.x + \mathsf{let}\, x\, \mathsf{in}\, x.x + x \stackrel{?}{=} \mathsf{let}\, x\, \mathsf{in}\, x'.x' + \mathsf{let}\, x'\, \mathsf{in}\, x'.x' + x'$$

2.8. Feladat. *Írj minél több zárt ABT-t az alább megadott fajtában.* d *aritása* (A.A)A*,* g *aritása* (A, A)A*.*

$$a \in \mathsf{A} ::= y \mid \mathsf{d}\, y.a \mid \mathsf{g}\, a\, a$$
$$y, y', ... \in \mathsf{Var_A}$$

További információ:

– Az α-ekvivalencia legegyszerűbb implementációja a De Bruijn indexek használata. Változónevek helyett természetes számokat használunk, melyek azt mutatják, hogy hányadik kötésre mutat a változó. Pl. $\mathsf{d}\, y.\mathsf{d}\, y'.y$ helyett $\mathsf{d}\, (\mathsf{d}\, 1)$-et írunk (0 mutatna a legközelebbi kötésre).

2.3. Levezetési fák

Ítéleteket ABT-kről mondunk. Néhány példa ítéletekre és a lehetséges jelentésükre:

n nat	n egy természetes szám
$n + n'$ is n''	az n és n' természetes számok összege n''
e hasHeight n	az e bináris fának n a magassága
$e : \tau$	az e kifejezésnek τ a típusa
$e \longmapsto e'$	az e kifejezés e'-re redukálódik

Az ítéletek *levezetési szabályokkal* vezethetők le. A levezetési szabályok általános formája az alábbi. $J_1, ..., J_n$-t feltételeknek, J-t következménynek nevezzük.

$$\frac{J_1 \qquad ... \qquad J_n}{J}$$

Az $n + n'$ is n'' ítélethez például az alábbi kettő levezetési szabályt adjuk meg.

$$\frac{}{\mathsf{zero} + n' \text{ is } n'} \tag{2.1}$$

$$\frac{n + n' \text{ is } n''}{\mathsf{suc}\, n + n' \text{ is } \mathsf{suc}\, n''} \tag{2.2}$$

Az n, n', n'' metaváltozók bármilyen Nat fajtájú ABT-t jelenthetnek. A 2.1 szabályban fontos, hogy a két n' mindig ugyanaz kell, hogy legyen. Megjegyezzük,

hogy ebben az ítéletben a +-nak semmi köze nincsen az Exp fajtájú ABT-kben szereplő + operátorhoz, csak véletlenül ugyanazzal a karakterrel jelöljük.

A levezetési szabályok *levezetési fává* (levezetéssé) kombinálhatók. Pl. azt, hogy $2 + 1 = 3$, a következőképp tudjuk levezetni.

$$\dfrac{\dfrac{\dfrac{}{\mathsf{zero} + \mathsf{suc}\,\mathsf{zero}\ \mathsf{is}\ \mathsf{suc}\,\mathsf{zero}}\ (2.1)}{(\mathsf{suc}\,\mathsf{zero}) + \mathsf{suc}\,\mathsf{zero}\ \mathsf{is}\ \mathsf{suc}\,(\mathsf{suc}\,\mathsf{zero})}\ (2.2)}{\mathsf{suc}\,(\mathsf{suc}\,\mathsf{zero}) + \mathsf{suc}\,\mathsf{zero}\ \mathsf{is}\ \mathsf{suc}\,\big(\mathsf{suc}\,(\mathsf{suc}\,\mathsf{zero})\big)}\ (2.2)$$

A levezetési fa gyökerénél van, amit levezettünk, és mindegyik lépésben valamelyik szabályt alkalmaztuk: az alkalmazott szabály a vízszintes vonal mellé van írva.

Az olyan levezetési szabályokat, melyeknek nincs feltétele, *axiómának* nevezzük. A levezetési fák leveleinél mindig axiómák vannak.

A természetes számokat az alábbi levezetési szabályokkal adjuk meg.

$$\frac{}{\mathsf{zero}\ \mathsf{nat}} \tag{2.3}$$

$$\frac{n\ \mathsf{nat}}{\mathsf{suc}\,n\ \mathsf{nat}} \tag{2.4}$$

Ezek azt fejezik ki, hogy a 0 természetes szám, és bármely természetes szám rákövetkezője is az. N.b. azt nem tudjuk levezetni, hogy i nat egy i változóra. A természetes számokra gondolhatunk úgy, mint a zárt Nat fajtájú ABT-kre.

A következő levezetési szabályok azt fejezik ki, hogy az Exp fajtájú AST kiegyensúlyozott, és magassága n. Az ítélet általános alakja $\mathsf{isBalanced}\ n$.

$$\frac{}{x\ \mathsf{isBalanced}\ \mathsf{zero}} \tag{2.5}$$

$$\frac{}{\mathsf{num}\,n\ \mathsf{isBalanced}\ \mathsf{zero}} \tag{2.6}$$

$$\frac{e\ \mathsf{isBalanced}\ n \qquad e'\ \mathsf{isBalanced}\ n}{e + e'\ \mathsf{isBalanced}\ \mathsf{suc}\,n} \tag{2.7}$$

Egy példa levezetés.

$$\dfrac{\dfrac{\dfrac{}{x'\ \mathsf{isBalanced}\ \mathsf{zero}} \quad \dfrac{}{x\ \mathsf{isBalanced}\ \mathsf{zero}}}{x + x\ \mathsf{isBalanced}\ \mathsf{suc}\,\mathsf{zero}} \quad \dfrac{\dfrac{}{x'\ \mathsf{isBalanced}\ \mathsf{zero}} \quad \dfrac{}{\mathsf{num}\,\mathsf{zero}\ \mathsf{isBalanced}\ \mathsf{zero}}}{x' + \mathsf{num}\,i\ \mathsf{isBalanced}\ \mathsf{suc}\,\mathsf{zero}}}{(x + x) + (x' + \mathsf{num}\,i)\ \mathsf{isBalanced}\ \mathsf{suc}\,(\mathsf{suc}\,\mathsf{zero})}$$

Ha valamit be szeretnénk bizonyítani minden levezethető ítéletről, ehhez a szabályok szerinti strukturális indukciót használhatjuk. Azt szeretnénk belátni, hogy ha J levezethető, akkor $\mathcal{P}(J)$ igaz. Ebben az esetben elég belátnunk azt, hogy minden szabályra, melynek feltételei $J_1, ..., J_n$ és következménye J, ha $\mathcal{P}(J_1), ..., \mathcal{P}(J_n)$ mind teljesül, akkor $\mathcal{P}(J)$ is.

A strukturális indukció konkrét használatára mutatunk két példát.

A fenti 2.1 szabály alapján bármely $n \in \mathsf{Nat}$-ra le tudjuk vezetni, hogy $\mathsf{zero} + n\ \mathsf{is}\ n$. Megmutatjuk a másik irányt.

2.9. Lemma. *Ha n egy természetes szám, akkor $n + \mathsf{zero}\ \mathsf{is}\ n$.*

Bizonyítás. A természetes számok levezetése alapján szabály indukció. A fenti P-t úgy választjuk meg, hogy $P(n\ \mathsf{nat}) := n + \mathsf{zero}\ \mathsf{is}\ n$. Ha a 2.3 szabályt használtuk, akkor $\mathsf{zero} + \mathsf{zero}\ \mathsf{is}\ \mathsf{zero}$-t kell bizonyítanunk, ezt megtesszük a 2.1 szabállyal. Ha a 2.4 szabályt használtuk, akkor az indukciós hipotézis azt mondja, hogy $P(n\ \mathsf{nat}) = n + \mathsf{zero}\ \mathsf{is}\ n$, nekünk pedig azt kell bizonyítani, hogy $P(\mathit{suc}\, n\ \mathsf{nat}) = \mathsf{suc}\, n + \mathsf{zero}\ \mathsf{is}\ \mathsf{suc}\, n$. A 2.2 szabályt használjuk, a feltételét az indukciós feltevésünk adja meg. □

Második példaként bebizonyítjuk, hogy ha van egy kiegyensúlyott fánk, és ebbe behelyettesítünk egy 0 magasságú fát, akkor a kapott fa is kiegyensúlyozott lesz.

2.10. Lemma. *Ha* $e\ \mathsf{isBalanced}\ n$ *és* $e_1\ \mathsf{isBalanced}\ \mathsf{zero}$*, akkor bármely* x_1*-re* $e[x_1 \mapsto e_1]\ \mathsf{isBalanced}\ n$*.*

Bizonyítás. A következő eseteket kell ellenőriznünk.

- Ha a 2.5 szabályt használtuk $e\ \mathsf{isBalanced}\ n$ levezetésére, akkor azt kell belátnunk, hogy $x[x_1 \mapsto e_1]\ \mathsf{isBalanced}\ \mathsf{zero}$. Ha $x = x_1$, akkor ez azzal egyezik meg, hogy $e_1'\ \mathsf{isBalanced}\ \mathsf{zero}$, ezt pedig tudjuk. Ha $x \neq x_1$, akkor a 2.5 szabályt használjuk újra.
- Ha a 2.6 szabályt használtuk, akkor azt kell belátnunk, hogy $(\mathsf{num}\, n)[x_1 \mapsto e_1]\ \mathsf{isBalanced}\ \mathsf{zero}$, viszont a helyettesítés itt nem végez semmit, tehát a 2.6 szabályt újra alkalmazva megkapjuk a kívánt eredményt.
- Ha a 2.7 szabályt alkalmaztuk, akkor az indukciós feltevésekből tudjuk, hogy $e[x_1 \mapsto e_1]\ \mathsf{isBalanced}\ n$ és $e'[x_1 \mapsto e_1]\ \mathsf{isBalanced}\ n$, és azt szeretnénk belátni, hogy $(e + e')[x_1 \mapsto e_1]\ \mathsf{isBalanced}\ \mathsf{suc}\, n$, de a helyettesítés definíciója alapján ez megegyezik $e[x_1 \mapsto e_1] + e'[x_1 \mapsto e_1]\ \mathsf{isBalanced}\ \mathsf{suc}\, n$-al, amit pedig a 2.7 szabály alapján látunk.

□

2.11. Feladat. *Adjuk meg a* $\mathsf{max}\, n\, n' = n''$ *ítélet levezetési szabályait. Az ítélet azt fejezi ki, hogy* n *és* n' Nat*-beli ABT-k maximuma* n''*.*

2.12. Feladat. *Ennek segítségével adjuk meg a* $\mathsf{e}\ \mathsf{hasHeight}\ \mathsf{n}$ *ítélet levezetési szabályait.*

2.13. Feladat. *Adjuk meg a* $\mathsf{isEven}\, n$ *és* $\mathsf{isOdd}\, n$ *ítéletek levezetési szabályait, melyek azt fejezik ki, hogy* n *páros ill. páratlan szám. A levezetési szabályok hivatkozhatnak egymásra.*

2.14. Feladat. *Igaz -e, hogy bármely két zárt természetes számra levezethető, hogy mennyi azok összege a természetes számok összegének két levezetési szabályával?*

További információ:

- Az AST-k, ABT-k és a levezetési fák mind induktív definíciók, típuselméletben induktív családokként formalizálhatók (inductive families). A strukturális indukció elvét ebben az esetben az eliminátor fejezi ki.

3. Sztringek és természetes számok

3.1. Szintaxis

3.2. Operációs szemantika

4. Függvények

F_1 (Church)

5. Véges adattípusok

Szorzat és összeg típusok.

6. Végtelen adattípusok

Természetes számok, generikus programozás.

7. Parcialitás, rekurzív típusok

8. Polimorfizmus

Curry, típuskikövetkeztetés (korlátozások generálása és egységesítés), letpolimorfizmus, Hindley-Milner, Milner-Mycroft

9. Magasabbrendű polimorfizmus

F_2, F_ω

10. Altípus

$\mathsf{F}_{\omega<}$.

11. Típuselmélet