Formális nyelv = másodrendű algebrai elmélet

Kaposi Ambrus*

akaposi@inf.elte.hu

Kivonat

A formális nyelvek közé tartoznak a logikai és a programozási nyelvek. Ebben a jegyzetben példákon keresztül megmutatjuk, hogyan lehet redundáns információ nélkül, magas szinten ilyen nyelveket megadni, és bennük programozni.

Tartalomjegyzék

1. Elsőrendű algebrai elméletek

Ebben a fejezetben a félcsoport és a kombinátor kalkulus példáin keresztül bemutatjuk az algebrai elméleteket. Az algebrai elméletek (algebraic theory, AT) nagyon egyszerű programozási nyelveknek felelnek meg: olyan nyelveknek, melyekben nincsenek változók és típusok.

A legegyszerűbb Turing-teljes programozási nyelv Moses Schönfinkel kombinátor kalkulusa. Az S és K kombinátorokat tudjuk egymás után írni zárójelezetten (például $(\mathsf{SK})\mathsf{K}$ illetve $\mathsf{S}(\mathsf{KK})$ két különböző kombinátor term), és két átírási szabályunk van: $(\mathsf{K}u)f \rightsquigarrow u$ illetve $((\mathsf{S}f)g)u \rightsquigarrow (fu)(gu)$, ahol u, f és g tetszőleges kombinátor termek.

* ELTE Informatikai Kar, Programozási Nyelvek és Fordítóprogramok Tanszék

Hogyan írjuk le ezt a nyelvet teljesen precízen? Először is, mik azok a termek? Tetszőleges sztringek (karaktersorozatok)? Sztringek, melyekben csak S, K, nyitó és záró zárójelek szerepelhetnek? Ezek közül csak azon sztringek, melyek jól zárójelezettek? Bináris fák, melyeknek kétféle levelük lehet? Kétféle levelű bináris fák ekvivalenciaosztályai, melyek a fenti két átírási szabályból képzett ekvivalencia-kongruencia reláció alapján vannak képezve?

Ebben a cikkben a legutolsó változatot fogjuk használni, de kicsit egyszerűbben fogalmazzuk meg ugyanezt: a kombinátor kalkulus egy algebrai elmélet, melyben egy bináris művelet van (applikáció, egymás mellé írás), két nulláris művelet van (S és K), és két egyenlőség $(\mathsf{K}u)f = u$ és $((\mathsf{S}f)g)u = (fu)(gu)$. Konvenció, hogy az applikáció balra zárójeleződik, tehát a két egyenlőség rövidebben így is írható: $\mathsf{K}uf = u$ és $\mathsf{S}fgu = fu(gu)$. Az ilyen, három művelettel ellátott halmazokat, melyekre teljesül a fenti két egyenlőség, kombinátor algebráknak nevezzük.

Mielőtt a kombinátor algebrákata tárgyalnánk, megnézünk egy jól ismert algebrai elméletet.

1. Definíció (félcsoport). *Egy félcsoport a következő komponensekből áll:*

$$\begin{aligned} \mathsf{C} &: \mathsf{Set} \\ - \cdot - &: \mathsf{C} \to \mathsf{C} \to \mathsf{C} \\ \mathsf{ass} &: (x \cdot y) \cdot z = x \cdot (y \cdot z) \end{aligned}$$

Egy félcsoport tehát egy halmaz, rajta egy bináris művelet (mely „curryzett" módon, $\mathsf{C} \to (\mathsf{C} \to \mathsf{C})$ függvénnyel van megadva $\mathsf{C} \times \mathsf{C} \to \mathsf{C}$ helyett), mely asszociatív. Például félcsoportot alkotnak az egész számok az összeadással ($\mathsf{C} := \mathbb{Z}, x \cdot y := x + y$), hiszen az összeadás asszociatív. További példák: egész számok a szorzással; logikai értékek (kételemű halmaz) az és/vagy/kizáró vagy művelettel; sztringek az összefűzéssel (konkatenáció); $n \times n$-es valós mátrixok a mátrix-szorzással; $A \to A$ függvények a függvénykompozícióval tetszőleges A halmazra; A részhalmazai a metszettel/unióval; az egyelemű halmaz (mi a művelet?); az üres halmaz (mi a művelet?). Nem példa: egész számok a kivonással vagy hatványozással.

A kombinátor algebrákat is megadjuk ugyanezzel a jelöléssel.

2. Definíció (kombinátor algebra).

$$\mathsf{Tm} \quad : \mathsf{Set}$$

$- \cdot - : \mathsf{Tm} \to \mathsf{Tm} \to \mathsf{Tm}$ *(balra zárójeleződik)*

$\mathsf{K} \quad : \mathsf{Tm}$

$\mathsf{S} \quad : \mathsf{Tm}$

$\mathsf{K}\beta \quad : \mathsf{K} \cdot u \cdot f = u$

$\mathsf{S}\beta \quad : \mathsf{S} \cdot f \cdot g \cdot u = f \cdot u \cdot (g \cdot u)$

A kombinátor algebrákat szokás a kombinátor kalkulus modelljeinek, a félcsoportokat a félcsoport-elmélet modelljeinek nevezni.

A triviális kombinátor algebrában Tm az egyelemű halmaz.

3. Feladat. *Van-e kételemű kombinátor algebra (melyben* Tm *a kételemű halmaz)?*

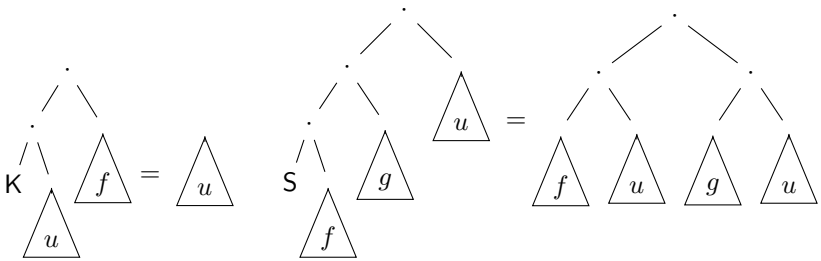
4. Gondolkodnivaló. *Van-e nemtriviális véges kombinátor algebra (melyben* Tm *véges halmaz)?*

Nemtriviális kombinátor algebrákat nem olyan könnyű találni, mint félcsoportokat. Ez nem meglepő, hiszen ez egy Turing-teljes nyelv – egy nemtriviális kombinátor algebrában elkódolható az összes Turing-gép.

5. Gondolkodnivaló. *Van-e olyan kombinátor algebra, melyben* Tm *a Turing-gépek kódjainak halmaza? Esetleg a Turing-gépek valamely kvóciens halmaza?*

Van azonban egy módszer, amellyel tetszőleges algebrai elméletnek megadható egy nemtriviális modellje, így a kombinátor kalkulusnak is: ezt szokás szabad algebrának vagy szintaxisnak nevezni. A kombinátor kalkulus szintaxisa a következő.

6. Definíció (Kombinátor kalkulus szintaxisa). *Vesszük a bináris fákat, melyek leveleinél a kételemű halmaz valamely eleme szerepel (ezt a két elemet* K*-val illetve* S*-el fogjuk jelölni), és ezek közül a következő fákat megegyezőnek tekintjük (itt a háromszögekkel tetszőleges fákat jelölünk, melyeket elnevezünk u-nak, f-nek illetve g-nek):*



A szintaxis „szabadsága” azt fejezi ki, hogy ez egy olyan algebra, mely csak annyit tud, amennyit az algebrai elmélet követel, sem többet (például nincs benne több egyenlőség), sem kevesebbet (hiszen az adott elmélet algebrája). A szintaxis elemeit szabadon generálja a három művelet, és az egyenlőségek szerint veszünk egy kvócienst. Ezt úgy is ki lehet fejezni, hogy a szintaxis elemei valójában ekvivalencia-osztályok: két term egy ekvivalenciaosztályban van, ha a két egyenlőség által generált ekvivalencia-kongruencia reláció szerint relációban állnak egymással. Ha két term a szintaxisban egyenlő, arra úgy gondolhatunk, hogy a két term, mint program, ugyanazt a végeredményt adja (vagy ugyanúgy nem adnak végeredményt, ha végtelen sokáig futó programok). Ha valamit meg tudunk csinálni a szintaxisban, azt meg tudjuk csinálni tetszőleges kombinátor-algebrában is: ha egy termet fel tudunk építeni a szintaxisban, azt fel tudjuk építeni tetszőleges algebrában. Ha két term egyenlő a szintaxisban, a megfelelő termek egyenlőek tetszőleges kombinátor algebrában is. A szintaxisban való munkát nevezzük programozásnak, ezt illusztráljuk most.

7. Feladat (Identitás). *Tetszőleges kombinátor algebrában megadható az algebra egy* I *eleme, melyre* $\mathsf{I} \cdot u = u$ *minden* u*-ra. Úgy is fogalmazhattunk volna, hogy a szintaxisban megadható egy* I *term, melyre* $\mathsf{I} \cdot u = u$ *minden* u*-ra.*

8. Feladat (kompozíció). *A szintaxisban megadható egy* B *term, melyre* $\mathsf{B} \cdot f \cdot g \cdot u = f \cdot (g \cdot u)$ *minden* f*,* g*,* u*-ra.*

9. Feladat (C kombinátor). *A szintaxisban megadható egy* C *term, melyre* $\mathsf{C} \cdot f \cdot u \cdot g = f \cdot g \cdot u$ *minden* f*,* g*,* u*-ra.*

Kombinátor termekkel reprezentálhatók a természetes számok: az alapötlet, hogy egy n szám egy olyan függvény (term), melyre $n \cdot u \cdot f = f \cdot (f \cdot (\dots (f \cdot u) \dots))$, ahol f-et n-szer alkalmazzuk u-ra. A nulla nem más, mint K, a rákövetkező, összeadás, szorzás stb. definiálható.

10. Feladat (Rákövetkező). *A szintaxisban megadható egy* suc *term, melyre* $\mathsf{suc} \cdot n \cdot z \cdot s = s \cdot (n \cdot z \cdot s)$ *minden* n*,* z*,* s*-re.*

11. Feladat (Összeadás). *A szintaxisban megadható egy* P *term, melyre* $\mathsf{P} \cdot a \cdot b \cdot z \cdot s = a \cdot (b \cdot z \cdot s) \cdot s$*.*

12. Feladat (Fixpont kombinátor). *A szintaxisban megadható egy* Y *term, melyre* $\mathsf{Y} \cdot f = f \cdot (\mathsf{Y} \cdot f)$*.*

Ennek segítségével tetszőleges rekurzív függvény implementálható.

13. Feladat (Fixpont kombinátor használata). *Az összeadás megadható a fixpont kombinátort használva.*

14. Gondolkodnivaló (Extenzionalitás). *Adjuk meg az* 1 *természetes szám két különböző implementációját! Tehát* t_0, t_1 *termeket, melyekre* $t_i \cdot u \cdot f = f \cdot u$, *de* $t_0 \neq t_1$. *Hogyan bizonyítjuk, hogy két term nem egyenlő?*

15. Gondolkodnivaló (Egyenlőségének eldönthetetlensége). *Feltételezve, hogy a metaelméletünk konstruktív (pl. Martin-Löf típuselmélet), a kombinátor kalkulus szintaxisában az egyenlőség eldönthetetlen. Tehát nem igaz, hogy bármely* t_0, t_1 *termekre* $t_0 = t_1$ *vagy* $t_0 \neq t_1$. *Ezt úgy tudjuk belátni, hogy megadjuk a metaelméletünk egy modelljét, amiben eldönthető (ez könnyű, tetszőleges klasszikus modell megfelel), és megadjuk egy modelljét, melyben ez az állítás hamis (ez már érdekesebb).*

16. Feladat. *Tetszőleges két kombinátor algebrának képezhető a szorzat algebrája, ahol a termek halmaza a két halmaz Descartes-szorzata (meg kell mutatnunk, hogy a termek Descartes-szorzatán megadható egy kombinátor algebra).*

17. Feladat. *Mutassuk meg, hogy van olyan kombinátor szintaktikus termek közötti egyenlőség, amely teljesül bizonyos kombinátor algebrában, de nem a szintaxisban.*

18. Feladat. *Mutassuk meg, hogy ha egy egyenlőség teljesül az összes kombinátor algebrában, akkor a szintaxisban is!*

19. Jelölés (Függvény-applikáció). *Ha van egy* $f : A \to B$ *függvényünk és hozzá egy* $a : A$ *bemenetünk, a függvény alkalmazását a bemenetre a matematikában legtöbbször* $f(a)$ *módon jelöljük. Ebben a jegyzetben a funkcionális programozásban szokásos módon ehelyett* f a*-t írunk.*

(Eddig csak infix függvényeink voltak, például $- \cdot -$, ezért nem kellett az applikációról külön beszélni. Ne tévessze meg az olvasót, hogy most már két különböző applikációról is beszélünk: a metanyelvi (metaelméleti) applikációt egyszerűen egymás mellé írással jelöljük, a tárgynyelvi (kombinátor kalkulusbeli) applikációt $- \cdot -$-tal.)

Néhány további, jól ismert algebrai elmélet: egységelemes félcsoport (angolul monoid), csoport, gyűrű, tetszőleges test feletti vektortér. Nagyon sok algebrai elmélet van: annyi, ahányféle művelet és egyenlőség adható meg. Egy algebrai elméletet a *szignatúrája*[1] határoz meg: szignatúra az operátorok aritását és az egyenlőségek oldalait tartalmazza. Egy egyszerű definíció ezekre a következő.

20. Definíció (Algebra szignatúra). *A szignatúra egyrészt egy* $\nu : \mathbb{N} \to \mathbb{N}$ *függvényből áll, ahol* $\nu\, n$ *megadja, hogy hány darab* n*-paraméteres műveletünk van. Másrészt tekintsük az alábbi szintaxist, amelynek segítségével az egyenlőségek két oldalát fogjuk elkódolni.*

$$\begin{aligned}
&\mathsf{tm} &&: \mathsf{Set}\\
&\mathsf{var} &&: \mathbb{N} \to \mathsf{tm}\\
&\mathsf{op} &&: (n : \mathbb{N}) \to \overline{\nu\, n} \to \underbrace{\mathsf{tm} \to \mathsf{tm} \to \ldots \to}_{n\ \mathit{darab}} \mathsf{tm}
\end{aligned}$$

Itt az $\bar{i}$ *az* i*-elemű halmazt jelöli. Ez egy olyan algebrai elmélet, melyben nincsenek egyenlőségek. Végtelen sok nulláris operátor van (minden természetes számhoz egy), ezek felelnek meg a változóknak, melyek az egyenlőségekben szerepelnek.* $\nu\, n$ *darab* n*-áris műveletünk van. A* ν *függvényen túl a szignatúra* tm*-párok egy halmazát tartalmazza.*

Például félcsoport szignatúrájában $\nu\, 2 = 1$ és minden $n \neq 2$-re $\nu\, n = 0$; a kombinátor kalkulus szignatúrájában $\nu\, 0 = 2$, $\nu\, 2 = 1$, és minden egyéb n-re $\nu\, n = 0$. A félcsoport szignatúrájában egy egyenlőség van, melyet a következő pár határoz meg (0_1-el jelöljük az egyelemű halmaz egyetlen elemét):

$$\Big(\mathsf{op}\, 2\, 0_1\, \big(\mathsf{op}\, 2\, 0_1\, (\mathsf{var}\, 0)\, (\mathsf{var}\, 1)\big)\, (\mathsf{var}\, 2),\\
\mathsf{op}\, 2\, 0_1\, (\mathsf{var}\, 0)\, \big(\mathsf{op}\, 2\, 0_1\, (\mathsf{var}\, 1)\, (\mathsf{var}\, 2)\big)\Big)$$

A kombinátor kalkulus szignatúrájában két egyenlőség van, a $\mathsf{K}\beta$ egyenlőséget az alábbi pár kódolja (a kételemű halmaz elemeit 0_2-vel és 1_2-vel jelöljük):

$$\Big(\mathsf{op}\, 2\, 0_1\, \big(\mathsf{op}\, 2\, 0_1\, (\mathsf{op}\, 0\, 0_2)\, (\mathsf{var}\, 0)\big)\, (\mathsf{var}\, 1), \mathsf{var}\, 0\Big)$$

[1] A szignatúrába beleértjük az egyenlőségeket is, mert az általánosított algebrai elméleteknél (2. fejezet) a műveletek és az egyenlőségek nem választhatók el: lesznek olyan műveleteink, melyek aritása csak bizonyos egyenlőségek megléte miatt értelmes.

Az $\mathsf{S}\beta$ egyenlőséget az alábbi pár kódolja:

$$\Big(\mathsf{op}\,2\,0_1\,\big(\mathsf{op}\,2\,0_1\,(\mathsf{op}\,2\,0_1\,(\mathsf{op}\,0\,1_2)\,(\mathsf{var}\,0))\,(\mathsf{var}\,1)\big)\,(\mathsf{var}\,2)$$
$$,\ \mathsf{op}\,2\,0_1\,\big(\mathsf{op}\,2\,0_1\,(\mathsf{var}\,0)\,(\mathsf{var}\,2)\big)\,\big(\mathsf{op}\,2\,0_1\,(\mathsf{var}\,1)\,(\mathsf{var}\,2)\big)\Big)$$

21. Feladat. *Kódoljuk el a csoport szignatúráját a fenti módon!*

22. Feladat. *Adjunk meg olyan algebrai elméletet, amelynek csak triviális algebrája van!*

23. Feladat. *Adjuk meg az algebra-fogalmat tetszőleges szignatúrára!*

24. Feladat. *Mutassuk meg, hogy minden algebrai elméletnek van triviális algebrája!*

25. Feladat. *Mutassuk meg, hogy minden algebrai elméletben tetszőleges két algebrának van szorzat-algebrája!*

26. Megjegyzés. *Az 5. fejezetben megadjuk egy szebb leírását az algebrai elméleteknek.*

Az algebra homomorfizmus egy függvény a két algebra alaphalmaza között, mely megtartja a műveleteket. Például A és B félcsoportok között ez egy $f : \mathsf{C}_A \to \mathsf{C}_B$ függvény, melyre teljesül, hogy $f\,(x \cdot_A y) = f\,x \cdot_B f\,y$ minden x, y-ra. Hasonlóképp, A és B kombinátor algebrákra ez egy $f : \mathsf{Tm}_A \to \mathsf{Tm}_B$ függvény, melyre $f\,(x \cdot_A y) = f\,x \cdot_B f\,y$, és $f\,\mathsf{K}_A = \mathsf{K}_B$ és $f\,\mathsf{S}_A = \mathsf{S}_B$.

27. Gondolkodnivaló. *Minden algebrai elmélethez megadható a homomorfizmus fogalom. A homomorfizmusok komponálhatók, és az identitás függvény mindig homomorfizmus.*

28. Definíció (Szintaxis). *A szintaxis az iniciális algebra: ez azt jelenti, hogy minden más algebrába pontosan egy homomorfizmus megy a szintaxisból.*

29. Feladat. *Mutassuk meg, hogy a szintaxis egyértelmű: bármely két iniciális algebra között van egy izomorfizmus (oda-vissza homomorfizmusok, melyek kompozíciói identitások).*

30. Feladat. *Mutassuk meg, hogy a félcsoportok szintaxisa az üres halmaz!*

31. Feladat. *Mutassuk meg, hogy a 6. definícióban megadott algebra valóban a kombinátor kalkulus szintaxisa!*

A szintaxisból egy adott algebrába menő homomorfizmust kiértékelésnek (interpretációnak) nevezzük.

32. Gondolkodnivaló (Üzemanyag szemantika).
A kombinátor-normálformák ilyen alakú termek: K, $\mathsf{K}{\cdot}n$, S, $\mathsf{S}{\cdot}n$, $\mathsf{S}{\cdot}n{\cdot}n'$, *ahol* n *és* n' *normálformák. Meg szeretnénk adni egy függvényt, ami szintaktikus kombinátor-termekről velük egyenlő normálformákra képez. Ez a függvény parciális lesz, hiszen például a fixpont-kombinátor nem egyenlő semmilyen normálformával. (Ezt hogyan bizonyítjuk?) Másrészt az nem eldönthető, hogy egy termnek van-e normálformája (Rice tétele). Emiatt egy olyan szemantikát tudunk megadni, ahol üzemanyag (egy természetes szám) mennyiségű lépést hajthatunk létre, és azt adjuk meg, hogy ennyi lépésben megkapjuk-e a normál formát. Minden, a szintaxison megadott függvénynek meg kell tartania az egyenlőségeket, ezért a végeredményt kvóciensezni kell a teljes relációval.*

33. Gondolkodnivaló. *Mutassuk meg, hogy a fixpont-kombinátornak nincs normálformája!*

34. Gondolkodnivaló. *Minden algebrai elméletnek van szintaxisa.*

Az induktívan megadott halmazok egyenlőségek nélküli algebrai elméletek szintaxisai: például a Peano természetes számokat az alábbi algebrai elmélet adja meg: $\mathsf{N} : \mathsf{Set}, \mathsf{zero} : \mathsf{N}, \mathsf{suc} : \mathsf{N} \to \mathsf{N}$.

Egy adott algebrai elmélet többféleképpen is prezentálható. Például a kombinátor kalkulushoz hozzávehetjük az I műveletet (lásd a 7. feladatot), és ezzel egy vele izomorf algebrai elméletet kapunk, melynek modelljei egy az egyben megfeleltethetők a kombinátor algebráknak. Vagy a félcsoportokhoz hozzávehetünk egy ternáris műveletet és egy egyenlőséget, mely kimondja, hogy a ternáris művelet egyenlő a bináris művelet kétszeri alkalmazásával. Általánosságban nehéz kérdés, hogy van-e egyszerűbb prezentációja egy adott algebrai elméletnek, és néha nem világos, hogy melyik a „jobb” prezentáció.

A testek nem adhatók meg közvetlenül a szokásos módon algebrai elméletként, mert az osztás csak a nem-nulla elemekre működik, de a fenti definíció szerint minden műveletnek működnie kell a teljes

alaphalmazra. Ez azonban még nem jelenti azt, hogy nincs valamilyen okosabb, izomorf megadása a testeknek. Ilyen azonban nincs.

35. Gondolkodnivaló. *A testek nem adhatók meg algebrai elméletként, mert tetszőleges algebrai elméletre vannak szorzat-algebrák, és van 2-elemű test és van 3-elemű test, de nincsen 6-elemű test.*

2. Elsőrendű általánosított algebrai elméletek

Ebben a fejezetben megadjuk a típusos kombinátor kalkulust. Ennek a természetes prezentációja már nem egyszerű algebrai elmélet, hanem általánosított algebrai elmélet (generalised algebraic theory, GAT).

Egy általánosított algebrai elméletnek az algebrájában nem csak egy alaphalmaz, hanem az alaphalmaz feletti indexelt család is van. Egy típusos kombinátor-algebrában van a típusoknak egy alaphalmaza, és minden típushoz tartozik egy külön halmaz, mely az olyan típusú termek halmaza. A típusokat dokumentációnak is gondolhatjuk: a kombinátor kalkulus ezen leírása intuitívabb, hiszen a típusok elmondják, hogy valójában mit csinál a K és az S kombinátor.

36. Definíció (Típusos kombinátor algebra).

Ty : Set

Tm : $\mathsf{Ty} \to \mathsf{Set}$

ι : Ty

$- \Rightarrow -$: $\mathsf{Ty} \to \mathsf{Ty} \to \mathsf{Ty}$

$- \cdot -$: $\mathsf{Tm}\,(A \Rightarrow B) \to \mathsf{Tm}\,A \to \mathsf{Tm}\,B$ *(balra zárójeleződik)*

K : $\mathsf{Tm}\,(A \Rightarrow B \Rightarrow A)$

S : $\mathsf{Tm}\,((A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C)$

$\mathsf{K}\beta$: $\mathsf{K} \cdot u \cdot f = u$

$\mathsf{S}\beta$: $\mathsf{S} \cdot f \cdot g \cdot u = f \cdot u \cdot (g \cdot u)$

$- \cdot -$ applikáció műveletnek van két implicit paramétere, ezek az A és a B. Az applikáció explicit leírása a következő lenne:

$$(A : \mathsf{Ty}) \to (B : \mathsf{Ty}) \to \mathsf{Tm}\,(A \Rightarrow B) \to \mathsf{Tm}\,A \to \mathsf{Tm}\,B$$

Nem akarjuk mindig kiírni az A-t és a B-t, ezért ezeket implicit módon mindig odaértjük. Az értékük általában kikövetkeztethető a másik két

paraméterből. Hasonlóan, a K *műveletnek két, az* S *műveletnek három implicit* Ty*-paramétere van. Az egyenlőségeknél is odaértjük az implicit* Ty/Tm*-paramétereket.*

Az ilyen, általánosított algebrai elméletekre is igaz minden, amit a közönséges algebrai elméletekről elmondtunk:

- Megadható a szignatúrájuk (lásd a 4. fejezetben).
- Minden szignatúrához megadhatók a következő fogalmak: algebra/modell, homomorfizmus.
- Minden általánosított algebrai elméletnek van szintaxisa.
- Algebráknak van Descartes-szorzata, van triviális algebra.

37. Feladat. *Mutassuk meg, hogy minden típus nélküli kombinátor algebra megad egy típusos kombinátor algebrát! A másik irány miért nem teljesül?*

38. Gondolkodnivaló. *Mutassuk meg, hogy minden típusos kombinátor termnek van normálformája! (A normálformák ugyanazok, mint a típus nélküli esetben; Tait logikai predikátumok módszerét kell használni a bizonyításra.)*

Megadunk néhány további általánosított algebrai elméletet.

39. Definíció (Gráf).

$$\begin{aligned}\mathsf{V} &: \mathsf{Set}\\ \mathsf{E} &: \mathsf{V} \to \mathsf{V} \to \mathsf{Set}\end{aligned}$$

A gráfban csúcsok (vertex) és élek (edge) vannak.

40. Definíció (Előrendezés (preorder)).

$$\begin{aligned}\mathsf{Ob} &: \mathsf{Set}\\ \mathsf{Mor} &: \mathsf{Ob} \to \mathsf{Mor} \to \mathsf{Set}\\ - \circ - &: \mathsf{Mor}\ J\ I \to \mathsf{Mor}\ K\ J \to \mathsf{Mor}\ K\ I\\ \mathsf{id} &: \mathsf{Mor}\ I\ I\\ \mathsf{irr} &: (f\ g : \mathsf{Mor}\ J\ I) \to f = g\end{aligned}$$

A preorder egy tranzitív és reflexív egyszerű gráf.

41. Definíció (Kategória).

$$
\begin{aligned}
&\mathsf{Ob} && : \mathsf{Set}\\
&\mathsf{Mor} && : \mathsf{Ob} \to \mathsf{Mor} \to \mathsf{Set}\\
&- \circ - && : \mathsf{Mor}\ J\ I \to \mathsf{Mor}\ K\ J \to \mathsf{Mor}\ K\ I\\
&\mathsf{id} && : \mathsf{Mor}\ I\ I\\
&\mathsf{ass} && : (f \circ g) \circ h = f \circ (g \circ h)\\
&\mathsf{idl} && : \mathsf{id} \circ f = f\\
&\mathsf{idr} && : f \circ \mathsf{id} = f
\end{aligned}
$$

A kategória egy tranzitív és reflexív gráf, melyben az élek kompozíciója asszociatív, és a reflexív élekkel való kompozíció az identitás.

42. Feladat. *Minden egységelemes félcsoport megad egy kategóriát (ahol egy darab objektum van).*

43. Feladat. *Adjuk meg a gráf-, preorder- és kategória-homomorfizmus fogalmát! Utóbbit funktornak szokás nevezni.*

44. Feladat. *Adjuk meg a halmazok kategóriáját, melyben az objektumok halmazok, a morfizmusok függvények az adott halmazok között.*

45. Megjegyzés. *Az egyenlőségek nélküli általánosított algebrai elméletek induktív-induktív típusoknak felelnek meg. Ha egyenlőségeket is tartalmaznak, ezeket kvóciens induktív-induktív típusoknak nevezzük [10]. Az általánosított algebrai elméleteket először Cartmell írta le [5].*

3. Másodrendű algebrai elméletek

Ebben a fejezetben másodrendű algebrai elméleteket (second-order algebraic theory, SOAT) vizsgálunk.

A legtöbb nyelvben vannak változók, vagy más néven kötéssel rendelkező operátorok. Például az $\int_a^b \frac{1}{x^2} dx$ kifejezésben az integrálás $\int - d -$ operátor köti az x változót az $\frac{1}{x^2}$ kifejezésben, ez utóbbit nevezzük a kötés hatáskörének. Ez a kifejezés megegyzik az $\int_a^b \frac{1}{y^2} dy$ kifejezéssel, hiszen a kötött változó neve nem számít, az csak egy mutató a kötés helyére. Hasonlóan, a $\sum_{x=1}^{10} x^2$ kifejezésben a $\sum$ a kötő operátor, a kötött változó az x, a kötés hatásköre az x^2 kifejezés. Az `int f(int i) { return(i+1); }` kifejezésben a függvénydefiníció a kötő operátor, a kötött név az `i`, a kötés hatásköre a kapcsos zárójelek (`{` és `}`) közötti rész.

46. Jelölés (Névnélküli függvény). *A „plusz egy” $\mathbb{N} \to \mathbb{N}$ függvényt megadhatjuk az alábbi két, ekvivalens módon:* $\mathsf{pluszegy}\, n := n + 1$ *illetve* $\lambda n.n + 1$*. Utóbbinak előnye, hogy nem kell nevet adnunk a függvénynek.*

Az integrálás operátor például a következő halmazban van (ahol $a, b : \mathbb{R}$): $\int_a^b : (\mathbb{R} \to \mathbb{R}) \to \mathbb{R}$. A fenti példa kifejezést az alábbi módon írhatjuk: $\int_a^b (\lambda x.\frac{1}{x^2})$, erre a szokásos jelölés a fenti, ahol $\lambda x.$ helyett a kifejezés végére írunk dx-et. Hasonlóan, az összegzés operátor precízen például mint $\sum : \mathbb{N} \to \mathbb{N} \to (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ adható meg, és a fenti kifejezés $\sum 1\, 10\, (\lambda x.x^2)$-et jelent.

A legegyszerűbb, kötésekkel rendelkező nyelv a lambda kalkulus, amely a típus nélküli kombinátor kalkulushoz hasonlóan Turing-teljes. Egy kötő operátor van, a $\mathsf{lam} : (\mathsf{Tm} \to \mathsf{Tm}) \to \mathsf{Tm}$, például az identitás függvényt a következőképp adjuk meg: $\mathsf{lam}\, (\lambda x.x)$. Ne tévessze meg az olvasót a két különböző „lambda”: a λ a metanyelvi (metaelméleti) lambda, hiszen a metanyelvünkben is tudunk függvényeket megadni, míg a lam a tárgynyelvi lambda.

47. Definíció (Lambda kalkulus). *Egy másodrendű lambda kalkulus modell (algebra) a következő komponensekből áll.*

$$\begin{aligned}
\mathsf{Tm} &: \mathsf{Set} \\
- \cdot - &: \mathsf{Tm} \to \mathsf{Tm} \to \mathsf{Tm} \\
\mathsf{lam} &: (\mathsf{Tm} \to \mathsf{Tm}) \to \mathsf{Tm} \\
\beta &: \mathsf{lam}\, f \cdot u = f\, u
\end{aligned}$$

A lambda kalkulus hasonló a kombinátor kalkulushoz, de a K és S helyett csak egy operátorunk van, a lam, és ennek megfelelően csak egy számítási szabályuk van. A lam azonban egy másodrendű operátor, hiszen a bemenete egy függvény. Ilyen másodrendű operátorok nem szerepelhetnek algebrai elméletekben, a 20. definícióval nem fejezhetők ki. A számítási szabály azt fejezi ki, hogy egy lam-mal megadott függvény tárgynyelvi applikációja ugyanaz, mint a metanyelvi applikáció.

Egy másodrendű lambda kalkulus modellből készíthető egy kombinátor algebra a következőképp.

$$\begin{aligned}
\mathsf{Tm} &:= \mathsf{Tm} \\
f \cdot u &:= f \cdot u \\
\mathsf{K} &:= \mathsf{lam}\left(\lambda u.\mathsf{lam}\,(\lambda f.u)\right)
\end{aligned}$$

$$\mathsf{S} \quad := \mathsf{lam}\left(\lambda f.\mathsf{lam}\left(\lambda g.\mathsf{lam}\left(\lambda u. f \cdot u \cdot (g \cdot u)\right)\right)\right)$$

Az egyenlőségek is teljesülnek:

$$\begin{aligned}
&\mathsf{K}\beta : \mathsf{K} \cdot u \cdot f && = (\mathsf{K}\text{ definíciója})\\
&\quad \left(\left(\mathsf{lam}\left(\lambda u.\mathsf{lam}\left(\lambda f.u\right)\right)\right) \cdot u\right) \cdot f && = (\beta)\\
&\quad \left(\left(\lambda u.\mathsf{lam}\left(\lambda f.u\right)\right) u\right) \cdot f && = (\text{metanyelvi függvényalkalmazás})\\
&\quad \left(\mathsf{lam}\left(\lambda f.u\right)\right) \cdot f && = (\beta)\\
&\quad (\lambda f.u)\, f && = (\text{metanyelvi függvényalkalmazás})\\
&\quad u
\end{aligned}$$

48. Feladat. *Mutassuk meg, hogy a fent megadott modellben az* $\mathsf{S}\beta$ *szabály is teljesül!*

49. Jelölés (Levezetési szabályok). *A lambda kalkulus operátorainak a megadására egy alternatív jelölés levezetési szabályokkal történik. Az elsőrendű függvénytér konklúziója helyett vízszintes vonalat írunk, a bemeneteket összegyűjtük (uncurry-zzük), a másodrendű függvénytér helyett* $\vdash$ *(turnstyle) szimbólumot.*

$$\frac{f : \mathsf{Tm} \qquad u : \mathsf{Tm}}{f \cdot u : \mathsf{Tm}} \qquad \frac{x : \mathsf{Tm} \vdash f : \mathsf{Tm}}{\mathsf{lam}\, x.f : \mathsf{Tm}} \qquad \frac{x : \mathsf{Tm} \vdash f : \mathsf{Tm} \qquad u : \mathsf{Tm}}{\beta : (\mathsf{lam}\, x.\, f) \cdot u = f[x \mapsto u]}$$

A vízszintes vonal feletti komponensek a bemenetek, az alatta levő rész a kimenet. Az első szabályt például a következőképp olvassuk: ha f *és* u *termek, akkor* $f \cdot u$ *is term. A levezetési fás jelölésben a másodrendű függvénytér* ($\vdash$) *használata esetén nevet adunk a másodrendű paraméternek, fent erre* x*-et használatunk. A választott név megjelenik a konklúzióban is. A metanyelvi applikációt* $f\, u$ *helyett* $f[x \mapsto u]$*-val jelöljük, amit úgy olvasunk, hogy* f*-ben az* x *értékét* u*-val helyettesítjük. A nevek nem számítanak, hiszen az* $x : \mathsf{Tm} \vdash f : \mathsf{Tm}$ *ugyanazt jelenti, mint hogy* $f : \mathsf{Tm} \to \mathsf{Tm}$*. A* β *szabály konklúzióját írhattuk volna például úgy is, hogy* $\beta : \mathsf{lam}\, y.\, f \cdot u = f[y \mapsto u]$*, a második szabály helyett írhattunk volna az alábbit is:*

$$\frac{y : \mathsf{Tm} \vdash f : \mathsf{Tm}}{\mathsf{lam}\, y.f : \mathsf{Tm}}$$

Ha csak egy szortunk van, akkor a sok $:\mathsf{Tm}$*-et nem szoktuk kiírni, hanem az alábbi, tömörebb módon is megadhatjuk a szabályokat.*

$$\frac{f \qquad u}{f \cdot u} \qquad \frac{x \vdash f}{\mathsf{lam}\, x.f} \qquad \frac{x \vdash f \qquad u}{\beta : (\mathsf{lam}\, x.f) \cdot u = f[x \mapsto u]}$$

A levezetési szabályos jelölés egyik előnye, hogy nem kell λ-kat írni a kötő operátorok számítási szabályainál, hátránya, hogy minden kötésnél meg kell adni egy nevet. Egy másik előny, hogy kikényszeríti a másodrendűséget, vagyis harmadrendű műveletek (például $((\mathsf{Tm} \to \mathsf{Tm}) \to \mathsf{Tm}) \to \mathsf{Tm}$) nem adhatók meg, mert a $\vdash$ bal oldalán nem szerepelhet függvény, vízszintes vonal vagy $\vdash$.

A levezetési szabályok között nincsenek változókra vonatkozóak, mint például az alábbi.

$$\frac{}{x : \mathsf{Tm} \vdash x : \mathsf{Tm}}$$

Ez a szabály a másodrendű algebrai leírásban egyszerűen a $\mathsf{Tm} \to \mathsf{Tm}$ identitás függvény lenne. Az ilyen és ehhez hasonló szabályokat implicit módon beleértjük a levezetési szabályos leírásba, hiszen az ilyen szabályok formális változata a másodrendű függvényekkel megadott. A levezetési szabályos változatot azért használjuk, mert a programozási nyelvek és bizonyításelméleti nyelvek ilyen jelölést szoktak használni.

Másodrendű algebrai elméleteknél nincs értelmes homomorfizmus-fogalom, például hogyan mondanánk ki azt, hogy egy $f : \mathsf{Tm}_A \to \mathsf{Tm}_B$ függvény megőrzi a lam operátort? Azt szeretnénk, hogy $f\,(\mathsf{lam}_A\, g) = \mathsf{lam}_B\,(f \circ g \circ ?)$, de nem tudunk a ? helyére mit írni, hiszen $\mathsf{Tm}_B \to \mathsf{Tm}_A$-ban nincs függvényünk ($\circ$ a szokásos függvénykompozíció). Ha f izomorfizmus, tehát egy olyan homomofizmus, melynek van inverzte, akkor ez működik, de az egy nagyon megszorított helyzet, ha csak izomorfizmusaink vannak. Ugyanez a probléma a szintaxis megadásánál. Emiatt a következő módszer [4] működik a homomorfizmus-fogalomra illetve a szintaxisra: a másodrendű elméletet lefordítjuk elsőrendűre, és ott tekintjük ezeket a fogalmakat. A fordítás alapötlete, hogy bevezetjük a környezetek és a helyettesítések szortját, és minden operátort és egyenlőséget ezekkel is indexelünk. A környezetek gyűjtik össze a másodrendű paramétereket. Ha van egy másodrendű modellünk, abból mindig képezhető egy modellje a megfelelő elsőrendű elméletnek. Ennél részletesebben ezt itt nem fejtjük ki, az érdeklődők utánanézhetnek [4, 14].

50. Megjegyzés. *A legtöbb programozási nyelvekről szóló tankönyv nem másodrendű nyelvként, hanem elsőrendű nyelvként adja meg a nyelveket, tehát lényegileg közvetlenül a másodrendű → elsőrendű fordítás eredményét adják meg. Sőt, legtöbbször még ennél is alacsonyabb szinten, típusozatlan pretermekként adják meg a szintaxist — nem világos azonban, hogy az általunk használt magas szinten tárgyalható-e minden fontos tulajdonsága a programozási nyelveknek. A következő tankönyveket javasoljuk az érdeklődőknek [8, 15, 12, 13, 11]. Bizonyos nyelvekben a változók használatára korlátozások vannak: például egy változó maximum egyszer vagy pontosan egyszer használható fel: ezeket szubstrukturális nyelveknek nevezzük. Ilyenek például a lineáris logika [6], lineáris típusrendszer [2], modális típuselméletek [7]. Ilyen nyelveket nem tárgyalunk ebben a jegyzetben, a másodrendű algebrai leíráshoz további kiegészítésekre lenne szükség.*

A nyelv fogalma illetve a szintaxis használata (a szintaxisban való programozás) azonban teljesen jól működik másodrendben is, ezt fogjuk itt illusztrálni. A programozás úgy működik, hogy feltételezünk egy másorendű modellt, és ennek a komponenseit használjuk.

Például (feltételezve egy tetszőleges másodrendű lambda kalkulus modellt, és annak komponenseit „levezetési szabályok" jelöléssel használva) az Y-kombinátor a következőképp adható meg.

$$\mathsf{Y} := \mathsf{lam}\, f.\big(\mathsf{lam}\, x.f \cdot (x \cdot x)\big) \cdot \big(\mathsf{lam}\, x.f \cdot (x \cdot x)\big)$$

Megmutatjuk, hogy ez egy fixpont-kombinátor:

$$\begin{aligned}
&\mathsf{Y} \cdot f && = (\mathsf{Y}\text{ definíciója})\\
&\Big(\mathsf{lam}\, f.\big(\mathsf{lam}\, x.f \cdot (x \cdot x)\big) \cdot \big(\mathsf{lam}\, x.f \cdot (x \cdot x)\big)\Big) \cdot f && = (\beta)\\
&\big(\mathsf{lam}\, x.f \cdot (x \cdot x)\big) \cdot \big(\mathsf{lam}\, x.f \cdot (x \cdot x)\big) && = (\beta)\\
&f \cdot \Big(\big(\mathsf{lam}\, x.f \cdot (x \cdot x)\big) \cdot \big(\mathsf{lam}\, x.f \cdot (x \cdot x)\big)\Big) && = (\mathsf{Y}\text{ definíciója})\\
&f \cdot (\mathsf{Y} \cdot f)
\end{aligned}$$

Feltételezve, hogy vannak $\mathsf{isZero} : \mathsf{Tm} \to \mathsf{Tm}$, kivonás $- - - : \mathsf{Tm} \to \mathsf{Tm} \to \mathsf{Tm}$, logikai elágazás $\mathsf{if}-\mathsf{then}-\mathsf{else}-$ műveleteink és számliteráljaink (ezek mind definiálhatók, csakúgy mint a típus nélküli kombinátor kalkulus esetén), szeretnénk megadni egy fac termet, mely a faktoriális függvényt implementálja, vagyis az alábbi egyenlőség igaz rá:

$$\mathsf{fac} \cdot n = \mathsf{if}\,\mathsf{isZero}\, n\,\mathsf{then}\, 1\,\mathsf{else}\,\mathsf{fac} \cdot (n-1)$$

Először is megadunk egy f termet, amely a rekurzív hívás struktúráját adja meg.

$$\mathsf{f} := \mathsf{lam}\, r.\mathsf{lam}\, n.\mathsf{if}\ \mathsf{isZero}\, n\ \mathsf{then}\ 1\ \mathsf{else}\, r \cdot (n-1)$$

Ennek segítségével a faktoriális függvény a következő.

$$\mathsf{fac} := \mathsf{Y} \cdot \mathsf{f}$$

És ez a definíció teljesíti a fenti egyenlőséget:

$$\begin{aligned}
&\mathsf{fac} \cdot n && = (\mathsf{fac}\ \text{definíciója}) \\
&\mathsf{Y} \cdot \mathsf{f} \cdot n && = (\mathsf{Y}\ \text{fixpont-kombinátor}) \\
&\mathsf{f} \cdot (\mathsf{Y} \cdot \mathsf{f}) \cdot n && = (\mathsf{f}\ \text{definíciója}) \\
&\big(\mathsf{lam}\, r.\mathsf{lam}\, n.\mathsf{if}\ \mathsf{isZero}\, n\ \mathsf{then}\ 1\ \mathsf{else}\, r \cdot (n-1)\big) \cdot (\mathsf{Y} \cdot \mathsf{f}) \cdot n \\
& && = (\beta\ \text{kétszer}) \\
&\mathsf{if}\ \mathsf{isZero}\, n\ \mathsf{then}\ 1\ \mathsf{else}\, \mathsf{Y} \cdot \mathsf{f} \cdot (n-1) && = (\mathsf{fac}\ \text{definíciója}) \\
&\mathsf{if}\ \mathsf{isZero}\, n\ \mathsf{then}\ 1\ \mathsf{else}\, \mathsf{fac} \cdot (n-1)
\end{aligned}$$

51. Megjegyzés. *A lambda kalkulus (elsőrendű) szintaxisa nem teljesen izomorf a kombinátor kalkulus szintaxisával, de ha mindkettőhöz hozzáveszünk néhány újabb egyenlőséget, akkor már izomorfak lesznek [1].*

4. Másodrendű általánosított algebrai elméletek

A legtöbb nyelv tartalmaz változókat, ezért a megadásához másodrendű algebrai elméletre van szükségünk. A legtöbb nyelvben van valamilyen típus-fogalom, ezért általánosított algebrai elméletre van szükségünk. Ebben a fejezetben ezt a két általánosítást kombinálva leírunk néhány alapvető logikai és programozási nyelvet másodrendű általánosított algebrai elméletként (second-order generalised algebraic theory, SOGAT).

Az alábbiakban megadjuk a minimális elsőrendű intuicionista logika nyelvét természetes levezetéses bizonyításelmélettel. A „minimális" azt jelenti, hogy csak implikációnk, univerzális kvantorunk van, és egy relációnk, az egyenlőség. Az egyszerűség kedvéért a bizonyítások szerkezetével nem foglalkozunk, ezt az az egyenlőség fejezi ki, amely

azt mondja, hogy tetszőleges két bizonyítás egyenlő. Emiatt a bizonyításokra vonatkozó hipotéziseknek nem adunk nevet, tehát nem írunk kettőspontot és elé nem írunk nevet (sem a $\vdash$ előtt, sem a feltételekben és a konklúziókban).

52. Definíció (Minimális intuicionista logika).

$$\frac{}{\mathsf{For} : \mathsf{Set}} \quad \frac{}{\mathsf{Tm} : \mathsf{Set}} \quad \frac{A : \mathsf{For} \qquad B : \mathsf{For}}{A \supset B : \mathsf{For}} \quad \frac{x : \mathsf{Tm} \vdash A : \mathsf{For}}{\forall x.A : \mathsf{For}}$$

$$\frac{t : \mathsf{Tm} \qquad t' : \mathsf{Tm}}{\mathsf{Eq}\, t\, t' : \mathsf{For}} \quad \frac{A : \mathsf{For}}{\mathsf{Pf}\, A : \mathsf{Set}} \quad \frac{p : \mathsf{Pf}\, A \qquad q : \mathsf{Pf}\, A}{p = q}$$

$$\frac{\mathsf{Pf}\, A \vdash \mathsf{Pf}\, B}{\mathsf{Pf}\, (A \supset B)} \qquad \frac{\mathsf{Pf}\, (A \supset B) \qquad \mathsf{Pf}\, A}{\mathsf{Pf}\, B}$$

$$\frac{x : \mathsf{Tm} \vdash \mathsf{Pf}\, A}{\mathsf{Pf}\, (\forall x.A)} \qquad \frac{\mathsf{Pf}\, (\forall x.A) \qquad t : \mathsf{Tm}}{\mathsf{Pf}\, (A[x \mapsto t])} \qquad \frac{t : \mathsf{Tm}}{\mathsf{Pf}\, (\mathsf{Eq}\, t\, t)} \qquad \frac{\mathsf{Pf}\, (\mathsf{Eq}\, t\, t')}{t = t'}$$

Példaként bebizonyítjuk, hogy $(A \supset B \supset C) \supset (B \supset A \supset C)$ minden A, B, C állításra. A leveleknél változók vannak.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{\Gamma \vdash \mathsf{Pf}\, (A \supset B \supset C)} \qquad \overline{\Gamma \vdash \mathsf{Pf}\, A}}{\Gamma \vdash \mathsf{Pf}\, (B \supset C)} \qquad \overline{\Gamma \vdash \mathsf{Pf}\, B}}{\underbrace{\mathsf{Pf}\, (A \supset B \supset C), \mathsf{Pf}\, B, \mathsf{Pf}\, A}_{=:\Gamma} \vdash \mathsf{Pf}\, C}}{\mathsf{Pf}\, (A \supset B \supset C), \mathsf{Pf}\, B \vdash \mathsf{Pf}\, (A \supset C)}}{\mathsf{Pf}\, (A \supset B \supset C) \vdash \mathsf{Pf}\, (B \supset A \supset C)}}{\mathsf{Pf}\, \big((A \supset B \supset C) \supset (B \supset A \supset C)\big)}$$

53. Feladat. *Adjuk meg az intuicionista logika másodrendű standard modelljét (Tarski-szemantikáját)! A formulák (metanyelvi) állításokkal, a termek valamilyen posztulált halmazzak, a bizonyítások metanyelvi bizonyításokkal vannak modellezve.*

54. Feladat. *Egészítsük ki a nyelvet általános reláció- és termszimbólumokkal, melyeket egy elsőrendű szignatúrából veszünk!*

55. Feladat. *Egészítsük ki a nyelvet a logikai konjunkció, diszjunkció, tagadás, igaz, hamis és egzisztenciális kvantor szabályaival!*

56. Feladat. *Mutasuk meg, hogy ha az így kiegészített nyelvhez hozzávesszük az alábbi szabályt, akkor a formulák Heyting-algebrát alkotnak!*

$$\frac{\mathsf{Pf}\,(A \supset B) \qquad \mathsf{Pf}\,(B \supset A)}{A = B}$$

57. Feladat. *Vezessük le tetszőleges A-ra, hogy* $\mathsf{Pf}\,(\neg\neg(A \vee \neg A))$*!*

58. Feladat. *Mutassuk meg, hogy akkor és csak akkor vezethető le minden A-ra* $\mathsf{Pf}\,(A \vee \neg A)$*, ha minden A-ra* $\mathsf{Pf}\,(\neg\neg A \supset A)$ *levezethető!*

59. Feladat. *Tegyük az előző feladat nyelvét klasszikussá: adjuk hozzá a kizárt harmadik elvét, tehát az*

$$\frac{}{\mathsf{Pf}\,(A \vee \neg A)}$$

szabályt! Mutasuk meg, hogy ha a klasszikus nyelvhez hozzávesszük az alábbi szabályt, akkor a formulák Boole-algebrát alkotnak!

$$\frac{\mathsf{Pf}\,(A \supset B) \qquad \mathsf{Pf}\,(B \supset A)}{A = B}$$

A következő nyelvet egyszerű típuselméletnek is szokás nevezni.

60. Definíció (Egyszerű típusos lambda kalkulus).

$$\frac{}{\mathsf{Ty} : \mathsf{Set}} \qquad \frac{A : \mathsf{Ty}}{\mathsf{Tm}\,A : \mathsf{Set}} \qquad \frac{}{\iota : \mathsf{Ty}} \qquad \frac{A : \mathsf{Ty} \qquad B : \mathsf{Ty}}{A \Rightarrow B : \mathsf{Ty}}$$

$$\frac{x : \mathsf{Tm}\,A \vdash b : \mathsf{Tm}\,B}{\mathsf{lam}\,x.b : \mathsf{Tm}\,(A \Rightarrow B)} \qquad \frac{f : \mathsf{Tm}\,(A \Rightarrow B) \qquad a : \mathsf{Tm}\,A}{f \cdot a : \mathsf{Tm}\,B}$$

$$\frac{}{\Rightarrow\beta : (\mathsf{lam}\,x.b) \cdot a = t[x \mapsto a]} \qquad \frac{f : \mathsf{Tm}\,(A \Rightarrow B)}{\Rightarrow\eta : f = \mathsf{lam}\,x.f \cdot x}$$

Az olyan nyelveknél, melyeknél ugyanez a két szort van, megadhatunk egy rövidebb jelölést is. Egyszerűen elhagyjuk a Ty-t és a Tm-et. Egyértelmű, hogy adott esetben melyikről van szó, mert a termeknek mindig oda van írva a típusa. Ezenkívül az egyenlőségeknek nem adunk nevet. Ezzel a spórolósabb jelöléssel ugyanez a kalkulus a következő.

$$\bar{\iota} \qquad \frac{A \qquad B}{A \Rightarrow B} \qquad \frac{x : A \vdash b : B}{\mathsf{lam}\,x.b : A \Rightarrow B} \qquad \frac{f : A \Rightarrow B \qquad a : A}{f \cdot a : B}$$

$$\frac{}{(\mathsf{lam}\, x.b) \cdot a = t[x \mapsto a]} \qquad \frac{f : A \Rightarrow B}{f = \mathsf{lam}\, x.f \cdot x}$$

Megadjuk az egyszerű típuselméletet algebrai jelöléssel is.

$$\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set} \\
\iota & : \mathsf{Ty} \\
- \Rightarrow - & : \mathsf{Ty} \to \mathsf{Ty} \to \mathsf{Ty} \\
\mathsf{lam} & : (\mathsf{Tm}\, A \to \mathsf{Tm}\, B) \to \mathsf{Tm}\, (A \Rightarrow B) \\
- \cdot - & : \mathsf{Tm}\, (A \Rightarrow B) \to \mathsf{Tm}\, A \to \mathsf{Tm}\, B \\
\Rightarrow\beta & : \mathsf{lam}\, b \cdot a = b\, a \\
\Rightarrow\eta & : f = \mathsf{lam}\, \lambda x.f \cdot x
\end{array}$$

61. Jelölés. *Az utolsó négy sort még rövidebben is meg tudjuk adni, a fentivel megegyezőt jelent az alábbi jelölés.*

$$\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set} \\
\iota & : \mathsf{Ty} \\
- \Rightarrow - & : \mathsf{Ty} \to \mathsf{Ty} \to \mathsf{Ty} \\
\mathsf{lam} & : (\mathsf{Tm}\, A \to \mathsf{Tm}\, B) \simeq \mathsf{Tm}\, (A \Rightarrow B) : - \cdot -
\end{array}$$

Azt mondjuk, hogy a $(\mathsf{Tm}\, A \to \mathsf{Tm}\, B)$ *és a* $\mathsf{Tm}\, (A \Rightarrow B)$ *halmazok ekvivalensek. Az ekvivalencia két oldalára írjuk az operátorokat, amelyek a két oldal között képeznek. Általánosságban egy ekvivalencia* $(f : X \simeq \simeq Y : g)$ *a következőt rövidíti:*

$$\begin{array}{l}
(f : X \to Y) \times (g : Y \to X) \times \\
\big((x : X) \to g\, (f\, x) = x\big) \times \big((y : Y) \to f\, (g\, y) = y\big)
\end{array}$$

62. Feladat. *Adjuk meg az egyszerű típusos lambda kalkulus standard modelljét! A típusok metanyelvi halmazokkal (metanyelvi típusokkal), a termek ezek elemeivel vannak modellezve.*

A következő nyelvnél is a spórolós jelölést használjuk.

63. Definíció (Gödel-féle System T).

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set} \\
- \Rightarrow - & : \mathsf{Ty} \to \mathsf{Ty} \to \mathsf{Ty} \\
\mathsf{lam} & : (\mathsf{Tm}\, A \to \mathsf{Tm}\, B) \simeq \mathsf{Tm}\, (A \Rightarrow B) : - \cdot - \\
\mathsf{Nat} & : \mathsf{Ty} \\
\mathsf{zero} & : \mathsf{Tm}\, \mathsf{Nat} \\
\mathsf{suc} & : \mathsf{Tm}\, \mathsf{Nat} \to \mathsf{Tm}\, \mathsf{Nat} \\
\mathsf{rec} & : \mathsf{Tm}\, A \to (\mathsf{Tm}\, \mathsf{Nat} \to \mathsf{Tm}\, A \to \mathsf{Tm}\, A) \to \mathsf{Tm}\, \mathsf{Nat} \to \mathsf{Tm}\, A \\
\mathsf{Nat}\beta_1 & : \mathsf{rec}\, z\, s\, \mathsf{zero} = z \\
\mathsf{Nat}\beta_1 & : \mathsf{rec}\, z\, s\, (\mathsf{suc}\, n) = s\, n\, (\mathsf{rec}\, z\, s\, n)
\end{array}
$$

A rec operátort leírjuk a levezetési szabályos jelöléssel is: ez azért érdekes, mert a rec a második paraméterében két változót is köt, ezeket ilyenkor vesszővel elválasztva soroljuk fel, a kötött változók nevét pedig az operátor jelölésekor ponttal választjuk el:

$$\frac{z : A \qquad i : \mathsf{Nat}, a : A \vdash s : A \qquad n : \mathsf{Nat}}{\mathsf{rec}\, z\, (i.a.s)\, n : A}$$

A számítási szabályok a levezetési szabályos jelöléssel a következők.

$$\frac{}{\mathsf{rec}\, z\, (i.a.s)\, \mathsf{zero} = z} \qquad \frac{}{\mathsf{rec}\, z\, (i.a.s)\, (\mathsf{suc}\, n) = s[i \mapsto n, a \mapsto \mathsf{rec}\, z\, (i.a.s)\, n]}$$

A suc-ra vonatkozó számítási szabályában ($\mathsf{Nat}\beta_2$) párhuzamosan helyettesítjük az i és az a változók értékét. Algebrai jelöléssel explicit változónevekkel ez a következőnek felel meg.

$$\mathsf{rec}\, z\, (\lambda i\, a.s\, i\, a)\, (\mathsf{suc}\, n) = s\, n\, (\mathsf{rec}\, z\, s\, n)$$

A természetes számok beágyazható System T-be: megadható egy $\ulcorner - \urcorner : \mathbb{N} \to \mathsf{Tm}\, \mathsf{Nat}$ függvény, mely n-hez $\mathsf{suc}^n\, \mathsf{zero}$-t rendeli, például $\ulcorner 3 \urcorner = \mathsf{suc}\, (\mathsf{suc}\, (\mathsf{suc}\, \mathsf{zero}))$. Egy $f : \mathbb{N} \to \mathbb{N}$ függvény definiálható System T-ben, ha létezik olyan $t : \mathsf{Tm}\, (\mathsf{Nat} \to \mathsf{Nat})$, melyre minden n-re $\ulcorner f\, n \urcorner = t \cdot \ulcorner n \urcorner$.

64. Feladat. *Az összeadás, szorzás, hatványozás és az Ackermann függvény definiálható System T-ben.*

65. Feladat. *Van olyan függvény, mely nem definiálható System T-ben.*

66. Feladat. *Adjuk meg a System T standard modelljét!*

67. Feladat. *Adjuk meg a* $\mathsf{pred} : \mathsf{Nat} \Rightarrow \mathsf{Nat}$ *függvényt, melyre teljesül, hogy* $\mathsf{pred}\,(\mathsf{suc}\,n) = n$ *minden* $n : \mathsf{Nat}$*-ra!*

68. Gondolkodnivaló. *Ha rekurzió helyett csak iteráció van, vagyis az* s *paraméter nem kapja meg az* $i : \mathsf{Nat}$*-ot, akkor is megadható a* pred *függvény, de az egyenlőség nem teljesül minden* $n : \mathsf{Nat}$*-ra, csak az* $n = \mathsf{suc}^n\,\mathsf{zero}$ *alakúakra.*

A következő nyelv ismét Turing-teljes, csakúgy, mint a típus nélküli kombinátor/lambda kalkulus, de vannak benne típusok. Lényegileg a System T kiegészítése egy fixpont-kombinátorral, de nincs η szabály a függvényekre. És mivel van fixpont-kombinátor, ezért nincs szükség rekurzorra a természetes számokhoz, elég egy ifZero C-stílusú if-then-else elágazás.

69. Definíció (PCF). *Két szortunk van, a típusok és a típusokkal indexelt termek.*

$$\frac{A \qquad B}{A \Rightarrow B} \qquad \frac{x : A \vdash b : B}{\mathsf{lam}\,x.b : A \Rightarrow B} \qquad \frac{f : A \Rightarrow B \qquad a : A}{f \cdot a : B}$$

$$\frac{}{(\mathsf{lam}\,x.b) \cdot a = t[x \mapsto a]} \qquad \frac{x : A \vdash t : A}{\mathsf{fix}\,x.t : A} \qquad \frac{}{\mathsf{fix}\,x.t = t[x \mapsto \mathsf{fix}\,x.t]}$$

$$\frac{}{\mathsf{Nat}} \qquad \frac{}{\mathsf{zero} : \mathsf{Nat}} \qquad \frac{n : \mathsf{Nat}}{\mathsf{suc}\,n : \mathsf{Nat}} \qquad \frac{b : \mathsf{Nat} \qquad t : A \qquad f : A}{\mathsf{ifZero}\,b\,t\,f : A}$$

$$\frac{}{\mathsf{ifZero}\,\mathsf{zero}\,t\,f = t} \qquad \frac{}{\mathsf{ifZero}\,(\mathsf{suc}\,n)\,t\,f = f}$$

Vegyük észre, hogy a függvényekre nincs η szabályunk.

70. Feladat. *Az összeadás, szorzás, hatványozás és az Ackermann függvény definiálható PCF-ben.*

71. Definíció (Összeg és szorzat típusok). *Az egyszerű típusos lambda kalkulust az alábbi szabályokkal egészítjük ki.*

$$\frac{}{\bot} \qquad \frac{b : \bot}{\mathsf{exfalso}\,b : A} \qquad \frac{x : \bot \vdash a : A}{a = \mathsf{exfalso}\,x}$$

$$\frac{A \qquad B}{A + B} \qquad \frac{a : A}{\mathsf{inl}\, a : A + B} \qquad \frac{b : B}{\mathsf{inr}\, b : A + B}$$

$$\frac{t : (A + B) \qquad x : A \vdash c_0 : C \qquad y : B \vdash c_1 : C}{\mathsf{case}\, t\, (x.c_0)\, (y.c_1) : C}$$

$$\frac{}{\mathsf{case}\, (\mathsf{inl}\, a)\, (x.c_0)\, (y.c_1) = c_0[x \mapsto a]} \qquad \frac{}{\mathsf{case}\, (\mathsf{inr}\, b)\, (x.c_0)\, (y.c_1) = c_1[y \mapsto b]}$$

$$\frac{z : A + B \vdash t : C}{t = \mathsf{case}\, z\, (x.t[z \mapsto \mathsf{inl}\, x])\, (y.t[z \mapsto \mathsf{inr}\, y])}$$

$$\frac{}{\top} \qquad \frac{}{\mathsf{tt} : \top} \qquad \frac{t : \top}{t = \mathsf{tt}}$$

$$\frac{A \qquad B}{A \times B} \qquad \frac{a : A \qquad b : B}{(a, b) : A \times B} \qquad \frac{t : A \times B}{\mathsf{fst}\, t : A} \qquad \frac{t : A \times B}{\mathsf{snd}\, t : B}$$

$$\frac{}{\mathsf{fst}\, (a, b) = a} \qquad \frac{}{\mathsf{snd}\, (a, b) = b} \qquad \frac{t : A \times B}{t = (\mathsf{fst}\, t, \mathsf{snd}\, t)}$$

72. Feladat. *Adjuk meg a logikai értékek típusát a következőképp:* $\mathsf{Bool} := \top + \top$*! Mutassuk meg, hogy az alábbi operátorok és egyenlőségek definiálhatóak!*

$$\frac{}{\mathsf{true} : \mathsf{Bool}} \qquad \frac{}{\mathsf{false} : \mathsf{Bool}} \qquad \frac{b : \mathsf{Bool} \qquad a_0 : A \qquad a_1 : A}{\mathsf{ite}\, b\, a_0\, a_1 : A}$$

$$\frac{}{\mathsf{ite}\, \mathsf{true}\, a_0\, a_1 = a_0} \qquad \frac{}{\mathsf{ite}\, \mathsf{false}\, a_0\, a_1 = a_1}$$

$$\frac{x : \mathsf{Bool} \vdash t : A}{t = \mathsf{ite}\, x\, (t[x \mapsto \mathsf{true}])\, (t[x \mapsto \mathsf{false}])}$$

73. Definíció (Izomorfizmus). *A és B típusok izomorfak, ha létezik $x : A \vdash t : B$ és $y : B \vdash t' : A$ term, melyekre $t[x \mapsto t'] = y$ és $t'[y \mapsto t] = x$. Ebben az esetben $A \cong B$-t írunk.*

74. Feladat. *Mutassuk, meg hogy $a \cong$ reláció reflexív, szimmetrikus és tranzitív, valamint kongruencia a $\times$-ra és a $+$-ra vontakozóan! (Tehát például $A \cong A'$-ből következik $A \times B \cong A' \times B$.)*

75. Feladat. *Mutassuk meg, hogy az összeg és szorzat típusokkal rendelkező nyelvben a $\cong$ relációval a típusok exponenciális semiring-et (félgyűrű, rig) alkotnak!*

76. Feladat. *Adjuk meg a fenti nyelv standard modelljét!*

Az alábbi nyelvben van egy generikus map függvényünk, ami nagyon sok $F : \mathsf{Ty} \to \mathsf{Ty}$ típusfüggvényre (a pozitív típusfüggvényekre) képes egy $(\mathsf{Tm}\, A \to \mathsf{Tm}\, B) \to \mathsf{Tm}\,(F\, A) \to \mathsf{Tm}\,(F\, B)$ függvényt megadni. A típusfüggvényekre megadjuk, hogy melyik pozitív és melyik negatív. Például pozitív az $F\, X = ((X + X) \Rightarrow \mathsf{Nat}) \Rightarrow X$, negatív az $F\, X = X \Rightarrow \mathsf{Nat}$, és az $F\, X = X \Rightarrow X$ se nem pozitív, se nem negatív. Pozitív és negatív helyett szokás kovariánsat és kontravariánsat mondani. Ebben és az induktív típusok nyelvében Robert Harpert követjük [8].

77. Definíció (Generikus programozás). *Az összeg és szorzat típusok nyelvét (71. definíció) kiegészítjük az alábbiakkal.*

$$
\begin{aligned}
&\mathsf{Pos} && : (\mathsf{Ty} \to \mathsf{Ty}) \to \mathsf{Set}\\
&\mathsf{Neg} && : (\mathsf{Ty} \to \mathsf{Ty}) \to \mathsf{Set}\\
&\mathsf{id}^{\mathsf{P}} && : \mathsf{Pos}\,(\lambda X.X)\\
&\mathsf{const}^{\mathsf{P}} && : (A : \mathsf{Ty}) \to \mathsf{Pos}\,(\lambda\_.A)\\
&\mathsf{const}^{\mathsf{N}} && : (A : \mathsf{Ty}) \to \mathsf{Neg}\,(\lambda\_.A)\\
&\_ +^{\mathsf{P}} \_ && : \mathsf{Pos}\, F \to \mathsf{Pos}\, G \to \mathsf{Pos}\ (\lambda X.F\, X + G\, X)\\
&\_ +^{\mathsf{N}} \_ && : \mathsf{Neg}\, F \to \mathsf{Neg}\, G \to \mathsf{Neg}\,(\lambda X.F\, X + G\, X)\\
&\_ \times^{\mathsf{P}} \_ && : \mathsf{Pos}\, F \to \mathsf{Pos}\, G \to \mathsf{Pos}\ (\lambda X.F\, X \times G\, X)\\
&\_ \times^{\mathsf{N}} \_ && : \mathsf{Neg}\, F \to \mathsf{Neg}\, G \to \mathsf{Neg}\,(\lambda X.F\, X \times G\, X)\\
&\_ \Rightarrow^{\mathsf{P}} \_ && : \mathsf{Neg}\, F \to \mathsf{Pos}\, G \to \mathsf{Pos}\ (\lambda X.F\, X \Rightarrow G\, X)\\
&\_ \Rightarrow^{\mathsf{N}} \_ && : \mathsf{Pos}\, F \to \mathsf{Neg}\, G \to \mathsf{Neg}\,(\lambda X.F\, X \Rightarrow G\, X)\\
&\mathsf{map}^{\mathsf{P}} && : \mathsf{Pos}\, F \to (f : \mathsf{Tm}\, A \to \mathsf{Tm}\, B) \to \mathsf{Tm}\,(F\, A) \to \mathsf{Tm}\,(F\, B)\\
&\mathsf{map}^{\mathsf{N}} && : \mathsf{Neg}\, F \to (f : \mathsf{Tm}\, A \to \mathsf{Tm}\, B) \to \mathsf{Tm}\,(F\, B) \to \mathsf{Tm}\,(F\, A)
\end{aligned}
$$

A számítási szabályok a következők, nem adunk nekik nevet.

$$
\begin{aligned}
&\mathsf{map}^{\mathsf{P}}\ \mathsf{id}^{\mathsf{P}} && f\, a && = f \cdot a\\
&\mathsf{map}^{\mathsf{P}}\ (\mathsf{const}^{\mathsf{P}}\ A) && f\, u && = u\\
&\mathsf{map}^{\mathsf{N}}\ (\mathsf{const}^{\mathsf{N}}\ A) && f\, u && = u\\
&\mathsf{map}^{\mathsf{P}}\ (F^{\mathsf{P}} +^{\mathsf{P}} G^{\mathsf{P}}) && f\,(\mathsf{inl}\, u) && = \mathsf{inl}\,(\mathsf{map}^{\mathsf{P}}\ F^{\mathsf{P}}\ f\, u)\\
&\mathsf{map}^{\mathsf{P}}\ (F^{\mathsf{P}} +^{\mathsf{P}} G^{\mathsf{P}}) && f\,(\mathsf{inr}\, v) && = \mathsf{inr}\,(\mathsf{map}^{\mathsf{P}}\ G^{\mathsf{P}}\ f\, v)\\
&\mathsf{map}^{\mathsf{N}}\ (F^{\mathsf{N}} +^{\mathsf{N}} G^{\mathsf{N}}) && f\,(\mathsf{inl}\, u) && = \mathsf{inl}\,(\mathsf{map}^{\mathsf{N}}\ F^{\mathsf{N}}\ f\, u)
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{map}^{\mathsf{N}}\,(F^{\mathsf{N}} +^{\mathsf{N}} G^{\mathsf{N}})\ f\,(\mathsf{inr}\,v) = \mathsf{inr}\,(\mathsf{map}^{\mathsf{N}}\,G^{\mathsf{N}}\,f\,v)\\
&\mathsf{map}^{\mathsf{P}}\,(F^{\mathsf{P}} \times^{\mathsf{P}} G^{\mathsf{P}})\ f\,(u,v) = (\mathsf{map}^{\mathsf{P}}\,F^{\mathsf{P}}\,f\,u, \mathsf{map}^{\mathsf{P}}\,G^{\mathsf{P}}\,f\,v)\\
&\mathsf{map}^{\mathsf{N}}\,(F^{\mathsf{N}} \times^{\mathsf{N}} G^{\mathsf{N}})\ f\,(u,v) = (\mathsf{map}^{\mathsf{N}}\,F^{\mathsf{N}}\,f\,u, \mathsf{map}^{\mathsf{N}}\,G^{\mathsf{N}}\,f\,v)\\
&\mathsf{map}^{\mathsf{P}}\,(F^{\mathsf{N}} \Rightarrow^{\mathsf{P}} G^{\mathsf{P}})\,f\,g \quad = \mathsf{lam}\,\lambda x.\mathsf{map}^{\mathsf{P}}\,G^{\mathsf{P}}\,f\,\big(g \cdot (\mathsf{map}^{\mathsf{N}}\,F^{\mathsf{N}}\,f\,x)\big)\\
&\mathsf{map}^{\mathsf{N}}\,(F^{\mathsf{P}} \Rightarrow^{\mathsf{N}} G^{\mathsf{N}})\,f\,g \quad = \mathsf{lam}\,\lambda x.\mathsf{map}^{\mathsf{N}}\,G^{\mathsf{N}}\,f\,\big(g \cdot (\mathsf{map}^{\mathsf{P}}\,F^{\mathsf{P}}\,f\,x)\big)
\end{aligned}
$$

A map függvény $\Rightarrow^{\mathsf{P}}$ esetét a következő diagram illusztrálja.

$$
\begin{array}{ccc}
A & & FA \xrightarrow{\quad g \quad} GA \\
\Big\downarrow{\scriptstyle f} & & \Big\uparrow{\scriptstyle \mathsf{map}^{\mathsf{N}}\,F^{\mathsf{N}}\,f} \quad\quad \Big\downarrow{\scriptstyle \mathsf{map}^{\mathsf{P}}\,F^{\mathsf{N}}\,f} \\
B & & FB \xrightarrow[\mathsf{map}^{\mathsf{P}}\,(F^{\mathsf{N}} \Rightarrow^{\mathsf{P}} G^{\mathsf{P}})\,f\,g]{} GB
\end{array}
$$

Egy induktív típust egy szigorúan pozitív típusfüggvény határoz meg (lásd az induktív típusok iniciális algebra szemantikáját). Először is map-pelnünk kell a szigorúan pozitív típusoperátorokon, majd megadhatjuk az induktív típusokat, mint szigorúan pozitív típusfüggvények legkisebb fixpontját.

78. Definíció (Induktív típusok). *Az összeg és szorzat típusok nyelvét (71. definíció) kiegészítjük az alábbiakkal.*

$$
\begin{aligned}
&\mathsf{SPos} && : (\mathsf{Ty} \to \mathsf{Ty}) \to \mathsf{Set}\\
&\mathsf{id} && : \mathsf{SPos}\,(\lambda X.X)\\
&\mathsf{const} && : (A : \mathsf{Ty}) \to \mathsf{SPos}\,(\lambda \_.A)\\
&- \bar{+} - && : \mathsf{SPos}\,F \to \mathsf{SPos}\,G \to \mathsf{SPos}\,(\lambda X.F\,X + G\,X)\\
&- \bar{\times} - && : \mathsf{SPos}\,F \to \mathsf{SPos}\,G \to \mathsf{SPos}\,(\lambda X.F\,X \times G\,X)\\
&- \bar{\Rightarrow} - && : (A : \mathsf{Ty}) \to \mathsf{SPos}\,F \to \mathsf{SPos}\,(\lambda X.A \Rightarrow F\,X)\\
&\mathsf{map} && : \mathsf{SPos}\,F \to (f : \mathsf{Tm}\,A \to \mathsf{Tm}\,B) \to \mathsf{Tm}\,(F\,A) \to \mathsf{Tm}\,(F\,B)\\
&&& : \mathsf{map}\ \ \mathsf{id}\ \ \ f\,a \quad = f \cdot a\\
&&& : \mathsf{map}\ \ (\mathsf{const}\,A)\,f\,u \quad = u\\
&&& : \mathsf{map}\ \ (\bar{F} \bar{+} \bar{G})\ \ f\,(\mathsf{inl}\,u) = \mathsf{inl}\,(\mathsf{map}\,\bar{F}\,f\,u)\\
&&& : \mathsf{map}\ \ (\bar{F} \bar{+} \bar{G})\ \ f\,(\mathsf{inr}\,v) = \mathsf{inr}\,(\mathsf{map}\,\bar{G}\,f\,v)\\
&&& : \mathsf{map}\ \ (\bar{F} \bar{\times} \bar{G}), f\,(u,v) = (\mathsf{map}\,\bar{F}\,f\,u, \mathsf{map}\,\bar{G}\,f\,v)\\
&&& : \mathsf{map}\ \ (A \bar{\Rightarrow} \bar{F})\ f\,g \quad = \mathsf{lam}\,\lambda a.\mathsf{map}\,\bar{F}\,f\,(g \cdot a)\\
&\mathsf{Ind} && : \mathsf{SPos}\,\mathsf{F} \to \mathsf{Ty}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{con} \quad & : (\bar{F} : \mathsf{SPos}\,\mathsf{F}) \to \mathsf{Tm}\left(F\,(\mathsf{Ind}\,\bar{F})\right) \to \mathsf{Tm}\,(\mathsf{Ind}\,\bar{F}) \\
\mathsf{ite} \quad & : (\bar{F} : \mathsf{SPos}\,\mathsf{F})(A : \mathsf{Ty}) \to \left(\mathsf{Tm}\,(F\,A) \to \mathsf{Tm}\,A\right) \to \mathsf{Tm}\,(\mathsf{Ind}\,\bar{F}) \to \\
& \quad\ \mathsf{Tm}\,A \\
\mathsf{Ind}\beta \quad & : \mathsf{ite}\,\bar{F}\,A\,f\,(\mathsf{con}\,\bar{F}\,x) = f\left(\mathsf{map}\,\bar{F}\,(\mathsf{ite}\,\bar{F}\,A\,f)\,x\right)
\end{aligned}
$$

79. Feladat. *A természetes számokat a $\lambda X.\top + X$ típusfüggvény segítségével a $\mathsf{Nat} := \mathsf{Ind}\,(\mathsf{const}\,\top \dot{+} \mathsf{id})$ típussal adjuk meg. Adjuk meg ehhöz a* zero *és* suc *konstruktorokat, valamint az iterátort és mutassuk meg, hogy a számítási szabályok teljesülnek!*

80. Feladat. *Kódoljuk el a következő induktív típusokat: természetes számok listái, bináris fák, háromfelé ágazó fák, végtelenfelé ágazó fák.*

A koinduktív típusokat terminális koalgebrák adják meg. Minden szigorúan pozitív típusfüggvény meghatároz egy koinduktív típust. A következőkben kiegészítjük az induktív típusok nyelvét koinduktív típusokkal is (igazából az induktív típusokra nincs szükségünk, csak SPos-ra és map-re).

81. Definíció ((Ko)induktív típusok). *Az induktív típusok nyelvét (78. definíció) kiegészítjük az alábbiakkal.*

$$
\begin{aligned}
\mathsf{Coind} \quad & : \mathsf{SPos}\,\mathsf{F} \to \mathsf{Ty} \\
\mathsf{des} \quad & : (\bar{F} : \mathsf{SPos}\,\mathsf{F}) \to \mathsf{Tm}\,(\mathsf{Coind}\,\bar{F}) \to \mathsf{Tm}\left(F\,(\mathsf{Coind}\,\bar{F})\right) \\
\mathsf{gen} \quad & : (\bar{F} : \mathsf{SPos}\,\mathsf{F})(A : \mathsf{Ty}) \to \left(\mathsf{Tm}\,A \to \mathsf{Tm}\,(F\,A)\right) \to \mathsf{Tm}\,A \to \\
& \quad\ \mathsf{Tm}\,(\mathsf{Coind}\,\bar{F}) \\
\mathsf{Coind}\beta \quad & : \mathsf{des}\,\bar{F}\,(\mathsf{gen}\,\bar{F}\,A\,f\,a) = \mathsf{map}\,\bar{F}\,(\mathsf{gen}\,\bar{F}\,A\,f)\,(f\,a)
\end{aligned}
$$

A koinduktív típusoknak destruktorai vannak, és generátoruk. A legalapvetőbb példa az A típusú elemeket tartalmazó végtelen lista (folyam, stream), melyet a $\lambda X.A \times X$ típusfüggvény határoz meg.

82. Feladat. *Adjuk meg a folyamokat koinduktív típusként, és adjuk meg az $[1, 2, 3, \dots]$ folyamot!*

83. Feladat. *Adjuk meg az olyan gépek koinduktív típusát, melyekbe be lehet írni egy (természetes) számot, ki lehet íratni velük egy számot, és meg lehet nyomni rajtuk egy gombot! Ennek a típusnak az elemeként készítsünk egy összeadó-gépet, mely a beírt számokat összeadja, az összeget kiírja, és a gomb megnyomásával nullázza az összeget.*

A következő nyelvben vannak polimorf típusok. Ezt úgy érjük el, hogy vannak típusváltozók, tehát a $\vdash$ bal oldalán típusok is szerepelhetnek. Például az identitás függvényt egyszerű típuselméletben különkülön meg kell adni az összes típusra, például $\mathsf{Nat} \Rightarrow \mathsf{Nat}$, $\mathsf{Bool} \Rightarrow \mathsf{Bool}$ stb., míg polimorf típusként meg tudjuk adni az összes típusra: $\forall A.A \Rightarrow A$. A következő típusrendszerben megkülönböztetjük a monomorf (MTy) és a polimorf típusokat (Ty), és az univerzális kvantor csak egy polimorf típus elején szerepelhet. Az alaptípusok (például Nat) monomorfak.

84. Definíció (Hindley–Milner-típusrendszer).

$$
\begin{array}{ll}
\mathsf{MTy} & : \mathsf{Set} \\
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set} \\
- \Rightarrow - & : \mathsf{MTy} \to \mathsf{MTy} \to \mathsf{MTy} \\
\forall & : (\mathsf{MTy} \to \mathsf{Ty}) \to \mathsf{Ty} \\
\mathsf{i} & : \mathsf{MTy} \to \mathsf{Ty} \\
\mathsf{lam} & : \big(\mathsf{Tm}\,(\mathsf{i}\,A) \to \mathsf{Tm}\,(\mathsf{i}\,B)\big) \simeq \mathsf{Tm}\,(\mathsf{i}\,(A \Rightarrow B)) \\
\mathsf{Lam} & : \big((A : \mathsf{MTy}) \to \mathsf{Tm}\,(B\,A)\big) \simeq \mathsf{Tm}\,(\forall\,B)
\end{array}
$$

A Hindley–Milner-típusrendszerben nincsenek induktív és koinduktív típusok, kézzel kell őket hozzáadni, csakúgy mint egyszerű típuselméletben. A szokásos módszer az, hogy minden típusra van egy if-then-else/case jellegű operátorunk, és van egy központi fixpontoperátorunk, mint PCF-ben.

85. Feladat. *Egészítsük ki a Hindley–Milner-típusrendszert fixpont-operátorral, egész számokkal (egész számokat egy $\mathbb{Z} \to \mathsf{Tm}\,\mathsf{Int}$ operátorral lehet bevezetni), melyekre van néhány primitív operátor: összeadás, szorzás, hatványozás, és $\leq$ összehasonlítás, mely C-stílusú logikai értéket ad vissza.*

86. Feladat. *Mutassuk meg, hogy az így megadott nyelven definiálható a faktoriális, a Fibonacci és az Ackermann függvény!*

87. Gondolkodnivaló. *A Hindley–Milner-nyelvre megadható egy típuskikövetkeztető, mely típusinformáció nélküli pretermekből szintaktikus termeket készít.*

A következő nyelvben a $\forall$ már bárhol szerepelhet, nem csak a típusok legelején.

88. Definíció (System F). *Két szortunk van, a típusok és a típusokkal indexelt termek.*

$$\frac{A \qquad B}{A \Rightarrow B} \qquad \frac{x : A \vdash b : B}{\mathsf{lam}\, x.b : A \Rightarrow B} \qquad \frac{f : A \Rightarrow B \qquad a : A}{f \cdot a : B}$$

$$\frac{}{(\mathsf{lam}\, x.b) \cdot a = t[x \mapsto a]} \qquad \frac{f : A \Rightarrow B}{f = \mathsf{lam}\, x.f \cdot x}$$

$$\frac{X \vdash A}{\forall X.A} \qquad \frac{X \vdash a : A}{\mathsf{Lam}\, X.a : \forall X.A} \qquad \frac{f : \forall X.A \qquad C}{f \bullet C : A[X \mapsto C]}$$

$$\frac{}{(\mathsf{Lam}\, X.a) \bullet C = a[X \mapsto C]} \qquad \frac{f : \forall X.A}{f = \mathsf{Lam}\, X.f \bullet X}$$

A System F rövidebb megadása:

$$\begin{array}{ll} \mathsf{Ty} & : \mathsf{Set} \\ \mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set} \\ - \Rightarrow - & : \mathsf{Ty} \to \mathsf{Ty} \to \mathsf{Ty} \\ \mathsf{lam} & : (\mathsf{Tm}\, A \vdash \mathsf{Tm}\, B) \simeq \mathsf{Tm}\, (A \Rightarrow B) : - \cdot - \\ \forall & : (\mathsf{Ty} \vdash \mathsf{Ty}) \to \mathsf{Ty} \\ \mathsf{Lam} & : (X : \mathsf{Ty} \vdash \mathsf{Tm}\, (A\, X)) \simeq \mathsf{Tm}\, (\forall\, A) : - \bullet - \end{array}$$

89. Gondolkodnivaló. *System F-ben az összes induktív és koinduktív típus elkódolható (Church-kódolás). Például* $Bool := \forall X.X \to X \to X$.

System F-ben vannak polimorf függvények, de nincsenek típus szintű függvények. Például a listák nem adhatók meg, ami egy típusról típusra képező függvény. A következő rendszer ezt kijavítja. A típusokon és a termeken túl van egy újabb szortunk, a fajták (Kind-ok) szortja, a típusok pedig indexelve vannak az ő „típusokkal" (fajtájukkal). A $*$ fajtájú típusok a tulajdonképpeni típusok, nekik lehetnek termei. A $* \Rrightarrow *$ fajtájú típusok a típus-függvények, melyek egy tulajdonképpeni típusból képeznek egy tulajdonképpeni típusba.

90. Definíció (System $\mathbf{F}_\omega$).

$$\begin{array}{ll} \mathsf{Kind} & : \mathsf{Set} \\ \mathsf{Ty} & : \mathsf{Kind} \to \mathsf{Set} \end{array}$$

$$
\begin{array}{ll}
- \Rrightarrow - & : \mathsf{Kind} \to \mathsf{Kind} \to \mathsf{Kind} \\
\mathsf{LAM} & : (\mathsf{Ty}\, K \to \mathsf{Ty}\, L) \simeq \mathsf{Ty}\, (K \Rrightarrow L) : - \bullet - \\
* & : \mathsf{Kind} \\
\mathsf{Tm} & : \mathsf{Ty} * \to \mathsf{Set} \\
\forall & : (\mathsf{Ty}\, K \to \mathsf{Ty} *) \to \mathsf{Ty} * \\
\mathsf{Lam} & : ((X : \mathsf{Ty}\, K) \to \mathsf{Tm}\, (A\, X)) \simeq \mathsf{Tm}\, (\forall\, A) : - \bullet - \\
- \Rightarrow - & : \mathsf{Ty} * \to \mathsf{Ty} * \to \mathsf{Ty} * \\
\mathsf{lam} & : (\mathsf{Tm}\, A \to \mathsf{Tm}\, B) \simeq \mathsf{Tm}\, (A \Rightarrow B) : - \cdot -
\end{array}
$$

A fajták és típusok a $\Rrightarrow$-val úgy viselkednek, mint egyszerű típuselméletben a típusok és a termek. A System F_ω érdekessége, hogy az egyik szort felhasznál egy operátort ($*$-ot), tehát a szortok nem adhatók meg az operátoroktól elkülönülten.

A lista típus a következőképp Church-kódolható:

$$
\begin{array}{l}
\mathsf{List} : \mathsf{Ty}\, (* \Rrightarrow *) \\
\mathsf{List} := \mathsf{LAM}\, \lambda A. \forall\, \lambda B. B \Rightarrow (A \Rightarrow B \Rightarrow B) \Rightarrow B
\end{array}
$$

A Haskell bind operátorának a következő a típusa:

$$\forall\, \lambda M. \forall\, \lambda A. \forall\, \lambda B. (A \Rightarrow M \bullet B) \Rightarrow M \bullet A \Rightarrow M \bullet B$$

A λ-kat kihagyva és a $\forall$ által kötött típusok fajtáit odaírva a következőt kapjuk:

$$\forall (M : * \Rrightarrow *). \forall (A : *). \forall (B : *). (A \Rightarrow M \bullet B) \Rightarrow M \bullet A \Rightarrow M \bullet B$$

91. Feladat. *Definiáljuk a Church-kódolt lista típus konstruktorait és iterátorát (fold operátorát)!*

92. Feladat. *Adjuk meg a Church-kódolt lista típus return és bind operátorait és vizsgáljuk meg, hogy teljesítik-e a monád törvényeket!*

93. Feladat. *Adjuk meg a lambda-kocka (lambda cube [3]) összes nyelvét SOGAT-ként! Adjuk meg a legáltalánosabb nyelvet (CC) és fejezzük ki, hogy a speciálisabb nyelvek milyen extra kritériumokkal adódnak ebből.*

A függő típuselméletben a típusok termektől is függhetnek (System F-ben a típusok csak típusoktól függhettek).

94. Definíció (Martin-Löf típuselmélete). *Két szortunk van, a típusok és a típusokkal indexelt termek. A típusok továbbá a szintjükkel vannak indexelve, ami egy természetes szám. A 61. jelölést használjuk.*

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathbb{N} \to \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty}\, i \to \mathsf{Set} \\
\Pi & : (A : \mathsf{Ty}\, i) \to (\mathsf{Tm}\, A \to \mathsf{Ty}\, i) \to \mathsf{Ty}\, i \\
\mathsf{lam} & : ((a : \mathsf{Tm}\, A) \to \mathsf{Tm}\, (B\, a)) \simeq \mathsf{Tm}\, (\Pi\, A\, B) : - \cdot - \\
\Sigma & : (A : \mathsf{Ty}\, i) \to (\mathsf{Tm}\, A \to \mathsf{Ty}\, i) \to \mathsf{Ty}\, i \\
(-, -) & : (a : \mathsf{Tm}\, A) \times \mathsf{Tm}\, (B\, a) \simeq \mathsf{Tm}\, (\Sigma\, A\, B) : \mathsf{fst}, \mathsf{snd} \\
\mathsf{U} & : (i : \mathbb{N}) \to \mathsf{Ty}\, (i + 1) \\
\mathsf{c} & : \mathsf{Ty}\, i \simeq \mathsf{Tm}\, (\mathsf{U}\, i) : \mathsf{El} \\
\mathsf{Lift} & : \mathsf{Ty}\, i \to \mathsf{Ty}\, (i + j) \\
\mathsf{mk} & : \mathsf{Tm}\, A \simeq \mathsf{Tm}\, (\mathsf{Lift}\, A) : \mathsf{un} \\
\bot & : \mathsf{Ty}\, 0 \\
\mathsf{exfalso} & : \mathsf{Tm}\, \bot \to \mathsf{Tm}\, A \\
\top & : \mathsf{Ty}\, 0 \\
\mathsf{tt} & : \top \simeq \mathsf{Tm}\, \top \\
\mathsf{Bool} & : \mathsf{Ty}\, 0 \\
\mathsf{true} & : \mathsf{Tm}\, \mathsf{Bool} \\
\mathsf{false} & : \mathsf{Tm}\, \mathsf{Bool} \\
\mathsf{indBool} & : (C : \mathsf{Tm}\, \mathsf{Bool} \to \mathsf{Ty}\, i) \to \mathsf{Tm}\, (C\, \mathsf{true}) \to \mathsf{Tm}\, (C\, \mathsf{false}) \to \\
 & \quad (b : \mathsf{Tm}\, \mathsf{Bool}) \to \mathsf{Tm}\, (C\, b) \\
\mathsf{Bool}\beta_1 & : \mathsf{indBool}\, t\, f\, \mathsf{true} = t \\
\mathsf{Bool}\beta_2 & : \mathsf{indBool}\, t\, f\, \mathsf{false} = f \\
\mathsf{Id} & : (A : \mathsf{Ty}\, i) \to \mathsf{Tm}\, A \to \mathsf{Tm}\, A \to \mathsf{Ty}\, i \\
\mathsf{refl} & : (a : \mathsf{Tm}\, A) \to \mathsf{Tm}\, (\mathsf{Id}\, a\, a) \\
\mathsf{J} & : \big(C : (x : \mathsf{Tm}\, A) \to \mathsf{Tm}\, (\mathsf{Id}\, A\, a\, x) \to \mathsf{Ty}\, i\big) \to \\
 & \quad \mathsf{Tm}\, \big(C\, a\, (\mathsf{refl}\, a)\big) \to (x : \mathsf{Tm}\, A)(e : \mathsf{Tm}\, (\mathsf{Id}\, A\, a\, x)) \to \mathsf{Tm}\, (C\, x\, e) \\
\mathsf{Id}\beta & : \mathsf{J}\, C\, w\, a\, (\mathsf{refl}\, a) = w \\
\mathsf{W} & : (S : \mathsf{Ty}\, i) \to (\mathsf{Tm}\, S \to \mathsf{Ty}\, i) \to \mathsf{Ty}\, i \\
\mathsf{sup} & : (s : \mathsf{Tm}\, S) \to (\mathsf{Tm}\, (P\, s) \to \mathsf{W}\, S\, P) \to \mathsf{W}\, S\, P \\
\mathsf{indW} & : (C : \mathsf{Tm}\, (\mathsf{W}\, S\, P) \to \mathsf{Ty}\, i) \to
\end{array}
$$

$$\Big(\big((p : \mathsf{Tm}\,(P\,s)) \to \mathsf{Tm}\,(C\,(f\,p))\big) \to \mathsf{Tm}\,\big(C\,(\mathsf{sup}\,s\,f)\big)\Big) \to$$
$$(w : \mathsf{Tm}\,(\mathsf{W}\,\mathsf{S}\,\mathsf{P})) \to \mathsf{Tm}\,\big(C\,w\big)$$
$$\mathsf{W}\beta \quad : \mathsf{indW}\,C\,h\,(\mathsf{sup}\,s\,f) = h\,(\lambda p.\mathsf{indW}\,C\,h\,(f\,p))$$

95. Feladat. *Adjuk meg a Martin-Löf-típuselmélet másodrendű standard modelljét!*

96. Jelölés. *Rövidítések:*

$$A \Rightarrow B := \Pi\,A\,(\lambda\_.B)$$
$$A \times\ B := \Sigma\,A\,(\lambda\_.B)$$

97. Feladat. *Adjunk meg* $\mathsf{Tm}\,(\mathsf{Id}\,\mathsf{Bool}\,\mathsf{true}\,\mathsf{false} \Rightarrow \bot)$ *egy elemét!*

98. Feladat. *Fogalmazzuk meg, majd bizonyítsuk be, hogy az* $\mathsf{Id}\,A$ *tetszőleges A-ra egy ekvivalencia-reláció!*

99. Feladat. *Adjuk meg a természetes számokat a* W *típus segítségével! Bizonyítsuk be a teljes indukció elvét! (Szükség van Hugunin trükkjére [9].)*

100. Feladat. *Adjuk meg a koinduktív típusok szabályait* M *típusok segítségével!*

101. Feladat. *Mutassuk meg, hogy ha az univerzum szabálya* $\mathsf{U}\,i : \mathsf{Ty}\,i$ *lenne, akkor van* $\mathsf{Tm}\,\bot$*-nak eleme.*

102. Feladat. *Fogalmazzuk meg, majd mutassuk meg, hogy az elsőrendű logika beágyazható Martin-Löf-típuselméletbe!*

103. Definíció (Russell). *A Martin-Löf-típuselmélet Russell típusú, ha a következő szort-egyenlőség teljesül:*

$$\mathsf{Ty}\,i = \mathsf{Tm}\,(\mathsf{U}\,i)$$

104. Feladat. *Mutassunk egy másodrendű modellt, amelyben ez teljesül! Mutassuk meg, hogy tetszőleges másodrendű modellből képezhető olyan nemtriviális másodrendű modell, mely Russell.*

105. Gondolkodnivaló. *A következő nyelvek (elsőrendű) szintaxisában minden termnek van normálformája: egyszerű típuselmélet, System F, Hindley–Milner, Martin-Löf-típuselmélet.*

5. Másodrendű általánosított algebra szignatúrák megadása másodrendű általánosított algebrai elmélettel

Ebben a fejezetben univerzális algebrával foglalkozunk. Ahogy a 20. definícióban megadtuk az algebrai elméletek szignatúráit, most megint megadjuk az AT szignatúrákat, de a szignatúrák nyelve egy konkrét SOGAT szintaxisa lesz. Látni fogjuk, hogy ez kényelmes definícióhoz vezet, ahol kevesebb kódolás költség van. A SOGAT-tal megadott szignatúra módszer továbbá nem csak AT szignatúrákra, hanem GAT és SOGAT szignatúrákra is működik.

A következő SOGAT szintaxisa írja le az algebrai elméletek szignatúráit, ez egy alternatívája a 20. definíciónak. Egy típus felel meg egy szignatúrának. Az algebrai szignatúrák nyelve Martin-Löf típuselméletének egy megszorítása: a Π típusnak például fix az értelmezési tartománya. Ezzel biztosítjuk, hogy ne lehessenek magasabbrendű függvények, és a függvények bemenete mindig az adott algebrai elmélet egyetlen szortja legyen. Egy alaptípusuk van, az algebrai elmélet szortja (Srt).

106. Definíció (AT).

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set} \\
\Sigma & : (A : \mathsf{Ty}) \to (\mathsf{Tm}\, A \to \mathsf{Ty}) \to \mathsf{Ty} \\
\mathsf{Srt} & : \mathsf{Ty} \\
\Pi\mathsf{Srt} & : (\mathsf{Tm}\, \mathsf{Srt} \to \mathsf{Ty}) \to \mathsf{Ty} \\
- \cdot - & : \mathsf{Tm}\, (\Pi\mathsf{Srt}\, B) \to (x : \mathsf{Tm}\, \mathsf{Srt}) \to \mathsf{Tm}\, (B\, x) \\
\mathsf{Id} & : \mathsf{Tm}\, \mathsf{Srt} \to \mathsf{Tm}\, \mathsf{Srt} \to \mathsf{Ty}
\end{array}
$$

Minden $B : \mathsf{Ty}$-ra bevezetjük a $\mathsf{Srt} \Rightarrow B := \Pi\mathsf{Srt}\, (\lambda\_.B)$ rövidítést.

Például a félcsoportok (1. definíció) a következőképp adhatók meg, a szignatúra Ty egy eleme.

$$
\begin{array}{l}
\Sigma\, (\mathsf{Srt} \Rightarrow \mathsf{Srt} \Rightarrow \mathsf{Srt}) \lambda op. \\
\quad \Pi\mathsf{Srt}\, \lambda x. \Pi\mathsf{Srt}\, \lambda y. \Pi\mathsf{Srt}\, \lambda z. \mathsf{Id}\, \big(op \cdot (op \cdot x \cdot y) \cdot z\big)\, \big(op \cdot x \cdot (op \cdot y \cdot z)\big)
\end{array}
$$

Összehasonlítva az 1. definícióval, a szortot nem kell megadnunk, hiszen algebrai elméletekben (AT-ben) csak egy szort van, a Σ típus első

komponense a bináris operátor lesz, melynek curry-zett típusát megadjuk és *op*-nak nevezzük el. A Σ típus második komponense kvantifikál három Srt-beli elem fölött, majd visszaadja az egyenlőség (identitás) típus megfelelő elemét. az 1. definícióban megadott verzió tekinthető ennek a formális definíciónak a kényelmesebb megadásának.

107. Feladat. *Adjuk meg az 1 fejezetben megadott összes AT szignatúráját a fenti módon!*

108. Megjegyzés. *A definíciónk szerint az üres algebrai elmélet (egy szort, nulla operátor) nem AT. Ez is bevehető, ha az AT elmélethez hozzávesszük az egyelemű $\top$ típust.*

A következő SOGAT szintaxisa írja le a zárt GAT-ok szignatúráit. Egy típus felel meg egy szignatúrának, a szignatúra szortjai U-ban vannak, az operátorait (és az indexelt szortokat) a Π függvénytípussal írjuk le.

109. Definíció (Zárt GAT).

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Set} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Set} \\
\Sigma & : (A : \mathsf{Ty}) \to (\mathsf{Tm}\, A \to \mathsf{Ty}) \to \mathsf{Ty} \\
(-,-) & : (a : \mathsf{Tm}\, A) \times \mathsf{Tm}\,(B\, a) \simeq \mathsf{Tm}\,(\Sigma\, A\, B) : \mathsf{fst}, \mathsf{snd} \\
\mathsf{U} & : \mathsf{Ty} \\
\mathsf{El} & : \mathsf{Tm}\, \mathsf{U} \to \mathsf{Ty} \\
\Pi & : (a : \mathsf{Tm}\, \mathsf{U}) \to (\mathsf{Tm}\,(\mathsf{El}\, a) \to \mathsf{Ty}) \to \mathsf{Ty} \\
- \cdot - & : \mathsf{Tm}\,(\Pi\, a\, B) \to (x : \mathsf{Tm}\,(\mathsf{El}\, a)) \to \mathsf{Tm}\,(B\, x) \\
\mathsf{Id} & : (a : \mathsf{Tm}\, \mathsf{U}) \to \mathsf{Tm}\,(\mathsf{El}\, a) \to \mathsf{Tm}\,(\mathsf{El}\, a) \to \mathsf{Ty} \\
\mathsf{reflect} & : \mathsf{Tm}\,(\mathsf{Id}\, a\, u\, v) \to u = v
\end{array}
$$

Itt is használjuk a Martin-Löf-típuselméletnél bevezetett $A \Rightarrow B := \Pi\, A\, (\lambda\_.B)$ és $A \times B := \Sigma\, A\, (\lambda\_.B)$ rövidítéseket.

Például az előrendezés (40. definíció) szignatúrája a fenti SOGAT szintaxisában a következő típus (Ty egy eleme):

$$
\begin{array}{l}
\Sigma\, \mathsf{U}\, \lambda Ob.\Sigma \\
\quad (Ob \Rightarrow Ob \Rightarrow \mathsf{U})\, \lambda Mor. \\
\quad \big(\Pi\, Ob\, \lambda I.\Pi\, Ob\, \lambda J.\Pi\, Ob\, \lambda K.Mor \cdot J \cdot I \Rightarrow Mor \cdot K \cdot J \Rightarrow
\end{array}
$$

$$
\begin{aligned}
&\mathsf{El}\,(Mor \cdot K \cdot I))\times\\
&\big(\Pi\, Ob\, \lambda I.\mathsf{El}\,(Mor \cdot I \cdot I)\big)\times\\
&\big(\Pi\, Ob\, \lambda I.\Pi\, Ob\, \lambda J.\Pi\,(Mor \cdot J \cdot I)\, \lambda f.\Pi\,(Mor \cdot J \cdot I)\, \lambda g.\\
&\mathsf{Id}\,(Mor \cdot J \cdot I)\, f\, g\big)
\end{aligned}
$$

Látjuk, hogy a felsorolás helyett Σ típusokat használunk, kötéseknél metanyelvi λ-t, nincsenek implicit paramétereink, és ki kell írnunk az El-t, amikor egy U típusú termből típust csinálunk. Mindez csak egy nagyon explicit leírás a 40. definícióra.

110. Feladat. *Adjuk meg a 2 fejezetben megadott összes GAT szignatúráját a fenti módon!*

111. Definíció (Nyílt GAT). *A zárt GAT szignatúrák elméletét az alábbiakkal egészítjük ki:*

$$
\begin{aligned}
\hat{\Pi} \quad &: (T : \mathsf{Set}) \to (T \to \mathsf{Ty}) \to \mathsf{Ty}\\
-\hat{\cdot}- &: \mathsf{Tm}\,(\hat{\Pi}\, T\, B) \to (\alpha : T) \to \mathsf{Tm}\,(B\,\alpha)
\end{aligned}
$$

A nyílt általánosított algebrai elméletek már nem adhatók meg nyílt másodrendű általánosított algebrai elméletként, csak végtelenül elágazódó (infinitary) másodrendű általánosított algebrai elméletként, de ezzel semmi gond nincs, lásd [10].

A másodrendű általánosított algebrai elméleteket a következő szignatúrával adjuk meg. Van egy új szortunk, U^+, az ebben levő szortokat lehet másodrendű függvénytérben használni, más néven az U^+-ban levő szortjainkhoz tartoznak változók. Persze tehetjük az összes szortot U^+-ba, ezzel a szemlélettel írtuk le a korábbi SOGAT-jainkat.

112. Definíció (SOGAT). *A zárt GAT szignatúrák elméletét az alábbiakkal egészítjük ki:*

$$
\begin{aligned}
\mathsf{U}^+ \quad &: \mathsf{Ty}\\
\mathsf{el}^+ \quad &: \mathsf{Tm}\,\mathsf{U}^+ \to \mathsf{Tm}\,\mathsf{U}\\
\pi^+ \quad &: (a^+ : \mathsf{Tm}\,\mathsf{U}^+) \to \Big(\mathsf{Tm}\,\big(\mathsf{El}\,(\mathsf{el}^+\, a^+)\big) \to \mathsf{Tm}\,\mathsf{U}\Big) \to \mathsf{Tm}\,\mathsf{U}\\
\mathsf{lam}^+ \; &: \Big(\big(x : \mathsf{El}\,(\mathsf{el}^+\, a^+)\big) \to \mathsf{Tm}\,(\mathsf{El}\,(b\, x))\Big) \simeq \mathsf{Tm}\,\big(\mathsf{El}\,(\pi^+\, a^+\, b)\big) : -\cdot^+-
\end{aligned}
$$

Például az egyszerű típuselmélet (60. definíció) szignatúrája a következő (mivel típusok nem szerepelnek nyíl bal oldalán, más szóval nincsenek

típusváltozóink, ezért a típusok szortja U-ban van).

$$
\begin{aligned}
&\Sigma\,\mathsf{U}\lambda Ty.\Sigma\\
&(Ty \Rightarrow \mathsf{U}^{+})\,\lambda Tm.\\
&\mathsf{El}\,Ty \times \Sigma\\
&(Ty \Rightarrow Ty \Rightarrow \mathsf{El}\,Ty).\lambda arr.\Sigma\\
&\Big(\Pi\,Ty\,\lambda A.\Pi\,Ty\,\lambda B.\\
&\quad \big(Tm\cdot A \Rightarrow^{+} \mathsf{el}^{+}\,(Tm\cdot B)\big) \Rightarrow \mathsf{El}\,\big(\mathsf{el}^{+}\,(Tm\cdot(arr\cdot A\cdot B))\big)\Big)\,\lambda lam.\Sigma\\
&\Big(\Pi\,Ty\,\lambda A.\Pi\,Ty\,\lambda B.\\
&\quad \mathsf{el}^{+}\,(Tm\cdot(arr\cdot A\cdot B)) \Rightarrow \mathsf{el}^{+}\,(Tm\cdot A) \Rightarrow \mathsf{El}\,\big(\mathsf{el}^{+}\,(Tm\cdot B)\big)\Big)\,\lambda app.\\
&\Big(\Pi\,Ty\,\lambda A.\Pi\,Ty\,\lambda B.\Pi\,\big(Tm\cdot A \Rightarrow^{+} \mathsf{el}^{+}\,(Tm\cdot B)\big)\,\lambda b.\\
&\quad \Pi\,\big(\mathsf{el}^{+}\,(Tm\cdot A)\big)\,\lambda a.\\
&\quad \mathsf{Id}\,\big(\mathsf{el}^{+}\,(Tm\cdot B)\big)\,\big(app\cdot A\cdot B\cdot(lam\cdot A\cdot B\cdot b)\cdot a\big)\,(b\cdot^{+} a)\Big)\times\\
&\Big(\Pi\,Ty\,\lambda A.\Pi\,Ty\,\lambda B.\Pi\,\big(Tm\cdot(arr\cdot A\cdot B)\big)\,\lambda f.\\
&\quad \mathsf{Id}\,\Big(\mathsf{el}^{+}\,\big(Tm\cdot(arr\cdot A\cdot B)\big)\Big)\\
&\qquad f\,\big(lam\cdot A\cdot B\cdot(\mathsf{lam}^{+}\,\lambda x.app\cdot A\cdot B\cdot f\cdot x)\big)\Big)
\end{aligned}
$$

113. Feladat. *Mutassuk meg, hogy a 4. fejezetben megadott összes nyelv leírható SOGAT szignatúrával. Csak azokat a szortokat tegyük U^{+}-ba, melyeket feltétlenül szükséges (például egyszerű típuselméletnél vagy Martin-Löf-típuselméletnél csak Tm van U^{+}-ban, míg System F-nél Ty is).*

114. Feladat. *Mutassuk meg, hogy a SOGAT-okat leíró SOGAT megadható SOGAT-ként (saját farkába harap a kígyó).*

115. Gondolkodnivaló. *Adjuk meg a SOAT szignatúrákat SOGAT-leírással!*

116. Gondolkodnivaló. *Tetszőleges másodrendű általánosított algebrai elmélet másodrendű modelljéből képezhető nemtriviális elsőrendű modell. Lásd [4].*

117. Gondolkodnivaló. *A Martin-Löf-típuselméletet egyszer szeretnénk SOGAT-okkal kiegészíteni: például olyan szabályokkal, melyek tetszőleges SOGAT szignatúrához megadják annak elsőrendű szintaxisát.*

Hivatkozások

[1] Thorsten Altenkirch, Ambrus Kaposi, Artjoms Sinkarovs, and Tamás Végh. Combinatory logic and lambda calculus are equal, algebraically. In Marco Gaboardi and Femke van Raamsdonk, editors, *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, July 3-6, 2023, Rome, Italy*, volume 260 of *LIPIcs*, pages 24:1–24:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[2] Robert Atkey. Syntax and semantics of quantitative type theory. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 56–65. ACM, 2018.

[3] Henk Barendregt. Introduction to generalized type systems. *J. Funct. Program.*, 1(2):125–154, 1991.

[4] Rafaël Bocquet, Ambrus Kaposi, and Christian Sattler. For the metatheory of type theory, internal sconing is enough. In Marco Gaboardi and Femke van Raamsdonk, editors, *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, July 3-6, 2023, Rome, Italy*, volume 260 of *LIPIcs*, pages 18:1–18:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[5] John Cartmell. Generalised algebraic theories and contextual categories. *Ann. Pure Appl. Log.*, 32:209–243, 1986.

[6] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

[7] Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal. Multimodal dependent type theory. *Log. Methods Comput. Sci.*, 17(3), 2021.

[8] Robert Harper. *Practical Foundations for Programming Languages*. Cambridge University Press, New York, NY, USA, 2nd edition, 2016.

[9] Jasper Hugunin. Why not W? In Ugo de'Liguoro, Stefano Berardi, and Thorsten Altenkirch, editors, *26th International Conference on Types for Proofs and Programs, TYPES 2020, March 2-5, 2020, University of Turin, Italy*, volume 188 of *LIPIcs*, pages 8:1–8:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[10] András Kovács. Type-theoretic signatures for algebraic theories and inductive types. *CoRR*, abs/2302.08837, 2023.

[11] John Longley and Dag Normann. *Higher-Order Computability*. Theory and Applications of Computability. Springer, 2015.

[12] Benjamin C. Pierce. *Types and programming languages*. MIT Press, 2002.

[13] Benjamin C. Pierce, Arthur Azevedo de Amorim, Chris Casinghino, Marco Gaboardi, Michael Greenberg, Cătălin Hriţcu, Vilhelm Sjöberg, and Brent Yorgey. *Logical Foundations*, volume 1 of *Software Foundations*. Electronic textbook, 2023. Version 6.5, `http://softwarefoundations.cis.upenn.edu`.

[14] Taichi Uemura. A general framework for the semantics of type theory. *CoRR*, abs/1904.04097, 2019.

[15] Philip Wadler, Wen Kokke, and Jeremy G. Siek. *Programming Language Foundations in Agda*. August 2022.