Programozási nyelvek, mint algebrai elméletek

Habilitációs tézisfüzet

Kaposi Ambrus
akaposi@inf.elte.hu

2024. október 16.

Tartalomjegyzék

1. Bevezetés

Egy matematikai vagy programozási probléma megoldását azon az absztrakciós szinten kívánatos végezni, azon a nyelven érdemes tárgyalni, ahol nem adminisztrációval, boilerplate-tel, hanem a tényleges tartalommal töltjük az időnket. A szintetikus matematika illetve a magas szintű programozási nyelvek adnak erre lehetőséget. Például:

– A geometria axiomatikus (euklidészi) tárgyalása szintetikus, míg a koordinátageometria analitikus.

– A természetes számok Zermelo-féle $(\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\{\{\emptyset\}\}\}, \dots)$ illetve von Neumann-féle $(\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}, \dots)$ elkódolása analitikus, míg Peano-aritmetikában való tárgyalása szintetikus. Benacerraf a következőképp kritizálja a halmazelméletet [Ben65; MK23]. A reprezentáció-függő kódolás patologikus tulajdonságokhoz vezet: például $0 \in$ $\in 2$ teljesül von Neumann kódolására, de Zermelo-éra nem. Honnan tudhatjuk, hogy ha az egyik reprezentációról bizonyítunk valamit, az teljesülni fog a másikra is? Lásd még Reynolds fabuláját [MR91].

– Az alábbi témakörök szokásos tárgyalása analitikus, de tárgyalhatók szintetikusan is: homotópia-elmélet [Bru19], számításelmélet [Bau06], valószínűségszámítás [Sim20], domainelmélet [OS98], relativitáselmélet [MNS07] stb.

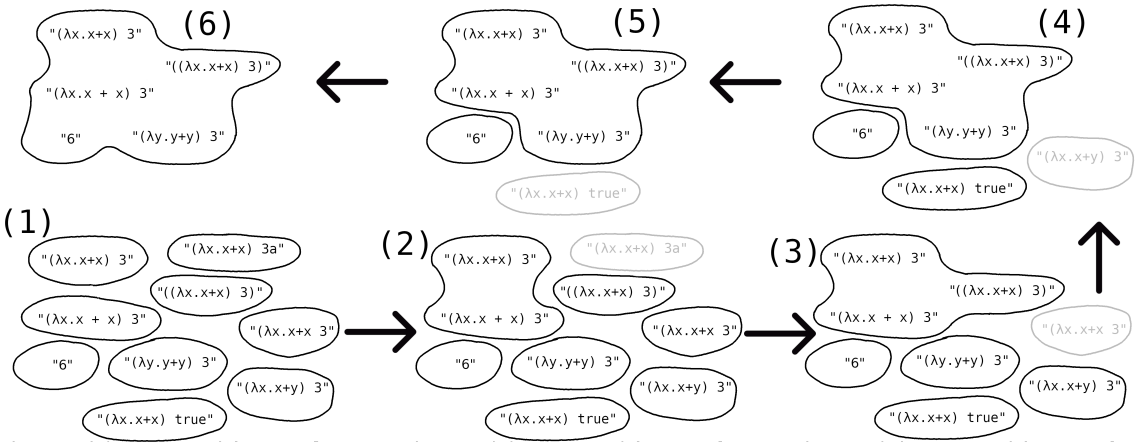
– A kategóriaelmélet [Awo10] célja meghatározni azt a lehető legkevesebb strukturával rendelkező univerzumot (kategóriát), melyben egy adott probléma reprezentációfüggetlenül tanulmányozható; gyakran kiderül, hogy a matematika különböző területein máshogy nevezett fogalmak ugyanannak az absztrakt fogalomnak speciális esetei. Szép példa erre Löb tételének, Gödel második nemteljességi tételének, a Kripke szemantikának és az őrzött (guarded) rekurziónak a közös általánosítása [Ram23].

– A funkcionális programozásban a programokat a szokásosnál absztraktabban (és így rövidebben) tudjuk megadni, a pontos reprezentációt vagy a végrehajtás sorrendjét a fordítóprogramra bízzuk [Hug89].

– A programozási nyelvek tudományában a szintetikus tárgyalás általában sekély beágyazásnak felel meg, míg az analitikus tárgyalás mély beágyazásnak [KKK19]. A sekély beágyazásban a beágyazott nyelv közel van a metanyelvhez, a mély beágyazásban a beágyazott nyelv szintaxisának alacsony szintű reprezentációs tulajdonságaihoz is hozzáférünk.

– A programozási és logikai nyelvek sok különböző absztrakciós szinten megadhatók, ezt alább kifejtük.

Az 1. ábra egy programozási nyelv néhány példaprogramját ábrázolja különböző absztrakciós szinteken. A nyelv legkonkrétabb és legkevésbé pontos megadása, ha azt mondjuk, hogy (1) egy program egy tetszőleges sztring. Azért konkrét, mert a számítógépbe ténylegesen sztringeket gépelünk be, amikor programozunk, és azért nem pontos, mert túl sok értelmetlen sztring van, melyek nem tartoznak a nyelvbe. Esetleg megadhatnánk a sztringeken egy predikátumot, mely csak azokra a sztringekre teljesül, melyekben csak a nyelvben megengedett betűk szerepelnek, de általában ehelyett azt mondjuk, hogy (2) a nyelv egy eleme egy adott ábécéből vett lexikális elemek sorozata. Vannak azonban olyan sorozatok, melyekben több kezdő-, mint záró-zárójel van, ezeket ki szeretnénk szűrni. Ezt is meg lehetne adni a sorozatokon egy predikátummal (ahogy például [VV03, 4.1.1. definíció.] teszi), ehelyett azt mondjuk, hogy (3) a nyelv egy eleme egy absztrakt szintaxisfa [Kap17], vagy még inkább (4) jól hatókörözött szintaxisfa (melyben



1. ábra. Egy lambda kalkulusra épülő nyelv néhány példaprogramja különböző szinteken megadva, (1)-es a legalacsonyabb, (6)-os a legmagasabb szintű. Minden buborék egy külön programnak felel meg. Alacsonyabb szinteken vannak olyan programok, melyek magasabb szinten értelmetlenek. Bizonyos programok, melyek alacsony szinten különbönek, magas szinten megegyeznek.

nincsenek sehova sem mutató változók, ezen a szinten dolgozik például [Har16]), (5) jól típusozott szintaxisfa [AR99; WKS22], (6) jól típusozott szintaxisfa a szemantikus egyenlőséggel faktorizálva [AK16]. A magasabb szinten kevesebb eleme van a nyelvnek, illetve bizonyos elemek, melyek alacsonyabb szinten különböznek, magasabb szinten megegyeznek. A magasabb szintű leírással megadott nyelven kevesebb konstrukció adható meg, például absztrakt szintaxisfákon nem tudjuk a zárójelek számát megszámolni, vagy jól hatókörözött szintaxisfán nem tudjuk a $\lambda x.x$ és a $\lambda y.y$ programokat megkülönböztetni. Ugyanakkor a magasabb szintű megadás rövidebb, érthetőbb, és bizonyos kívánatos tulajdonságok automatikusan teljesülnek az így megadott nyelvre. Például szükségtelen bebizonyítanunk, hogy a nyelv végrehajtása megőrzi a jólzárójelezettséget a (3)-as szinten, vagy a típushelyességet az (5)-ös szinten. A magasabb szintű megadást intrinzikusnak, míg az alacsony szintűt extrinzikusnak nevezik. A magas szintű megadás különösen bonyolult, sok szabállyal rendelkező nyelvek esetén előnyös, amikor alacsony szinten nem látjuk a fától az erdőt. A komplexitás növekedését így absztrakcióval ellensúlyozzuk.

A magasabb szintű leíráshoz erősebb metanyelvi eszközökre van szükség, például általános esetben (3)-hoz szükségünk van induktív halmazokra, (4)-hez indexelt induktív családokra, (5)-höz induktív-induktív családokra, (6)-hoz ezeknek a kvóciensekkel kombinált változataira. Ezek az eszközök mind elkódolhatók például halmazelméletben, de a fenti példákat is figyelembe véve talán még jobb, ha az adott eszközöket közvetlenül támogató, megfelelő szintetikus nyelvben dolgozunk. Martin-Löf típuselmélete [Mar98] egy sokoldalú programozási nyelv, mely többféle struktúra szintetikus tárgyalására használható (például homotópia-elmélet [Bru19], magasabb groupoidok [ALR14], kvantumtérelmélet [SS12], revision control rendszerek elmélete[Ang+16], process algebra [Gon23], univerzális algebra [Kov22] stb). A matematika általános megalapozására is alkalmas az elsőrendű logika és a halmazelmélet helyett, sőt, ha számítógéppel akarjuk a matematikát elkódolni, akkor a halmazelméletnél sokkal alkalmasabbnak tűnik [Pro13]. Az is mutatja ezt, hogy típuselméletre épülő bizonyítórendszerekben formalizálták nagyjából a teljes egyetemi matematika-tananyagot [Com20], illetve számos magasabb matematikai eredményt [BCM20; Sch21; Gon13].

A típuselmélet a típusok szintetikus elmélete (a típusokra lehet úgy gondolni, mint a halmazokra a strukturális halmazelméletekben). A természetes számok például az univerzális tulajdonságukkal [Awo10] vannak megadva: a természetes számok az iniciális *pontozott típus endofüggvénnyel*-struktúra (tehát egy N típus egy $z : N$ elemmel és egy $s : N \to N$ függvény típusú elemmel; az inicialitás azt jelenti, hogy minden más pontozott típus endofüggvénnyel

struktúrába pontosan egy homomorfizmus megy – tehát ha van egy másik (A, a, f) pontozott típus endofüggvénnyel, akkor pontosan egy olyan $\alpha : N \to A$ függvény van, melyre $\alpha\, z = a$ és $\alpha\,(s\, n) = f\,(\alpha\, n)$ minden $n : N$-re). A típuselmélet teljesíti Benacerraf kritériumát, és alkalmas a matematika strukturalista megalapozására.

A típuselméletben közvetlenül reprezentálhatók egy nyelv (1)–(6) szintű leírásai. Mindegyik szinten a nyelv szintaxisa valamilyen induktív típussal adható meg. A 2. fejezetben az induktív típusok különböző osztályait ismertetjük. Ezután a 3. fejezetben rátérünk az induktív típusokat támogató típuselméletre épülő metanyelvekre, végül a 4. fejezetben az ismertetett módszereket alkalmazzuk különböző programozási nyelvek magnyelveinek leírására, és azok tulajdonságainak bizonyítására. Minden alfejezet végén röviden ismertetjük a kapcsolódó munkákat. A téziseket az 5. fejezetben listázzuk. A szerző PhD fokozatának megszerzése óta megjelent saját közleményeit külön listázzuk [AK17]–[KX24b], az egyéb hivatkozott közlemények ezután következnek.

1.1. Jelölések

Ezen tézisfüzet metanyelve a típuselmélet, Agda-szerű jelöléseket használunk. A $t : A$ úgy olvasható, hogy t termnek (programnak) A a típusa, például $1 + 1 : \mathbb{N}$. Függvényeket λ-val adunk meg, például $(\lambda x.x * 2) : \mathbb{N} \to \mathbb{N}$ az a függvény, melynek kimenete a bemenet kettővel szorozva. A függvénytér jobbra zárójeleződik, tehát $A \to B \to C = A \to (B \to C)$. A típusok típusát Type-pal jelöljük. Függő függvényeket $(x : A) \to B$ jelöléssel adunk meg, például a polimorf identitás függvény típusa $(A : \mathsf{Type}) \to A \to A$, az adott méretű egységmátrixot visszaadó függvény típusa $(n : \mathbb{N}) \to \mathsf{Matrix}\, n\, n$. Az egyelemű típus $* : \mathbb{1}$, az egyenlőség típust $=$-vel jelöljük, a konverzió relációt (definicionális egyenlőséget) $\equiv$-vel. A függő szorzat típusokat $\times$-al jelöljük, például a tetszőleges méretű mátrixok típusát $(m : \mathbb{N}) \times (n : \mathbb{N}) \times \mathsf{Matrix}\, m\, n$-el jelöljük.

1.2. Köszönetnyilvánítás

Köszönjük szépen az ELTE Informatikai Kar támogatását a `HAB_23` pályázaton keresztül.

2. Szignatúrák elméletei

Az induktív típusok algebrai elméletek iniciális algebráinak felelnek meg.

1. Példa (Természetes számok)**.** *A természetes számok a legegyszerűbb algebrai elmélet-osztályba tartoznak, szignatúrájuk a következő:* $Nat : \mathsf{Type}, zero : Nat, suc : Nat \to Nat$*. Mint algebrai elméletet ezt úgy olvassuk, hogy egy fajtánk van, egy konstansunk és egy operátorunk, melynek aritása egy (nincs egyenlőségünk). Az elmélet egy algebrája (modellje) egy típus, amelynek van egy eleme és van rajta egy endofüggvény. Két algebra között van egy evidens homomorfizmus-fogalom. Mint induktív típust ezt úgy olvassuk, hogy a* Nat *egy induktív típus, két konstruktorral. Az első konstruktornak nincs paramétere, a másodiknak egy darab* Nat *paramétere van. A konstruktorok meghatározzák egy induktív típus eliminátorát, melynek itt két változatát ismertetjük.*

- *Az iterátor azt mondja, hogy tetszőleges algebrába megy egy homomorfizmus a* $Nat, zero, suc$ *algebrából.*
- *Egy függő algebra egy* $P : Nat \to \mathsf{Type}$ *típuscsaládból, egy* $z : P\, zero$ *elemből és egy* $s : (n : Nat) \to P\, n \to P\,(suc\, n)$ *függvényből áll. Egy* $(Nat, zero, suc)$*-ból (P,z,s)-be menő függő homomorfizmus (szekció) egy* $\alpha : (n : Nat) \to P\, n$ *függő függvényből áll, melyre teljesül, hogy* $\alpha\, zero = z$ *és minden* n*-re* $\alpha\,(suc\, n) = s\, n\,(\alpha\, n)$*. A természetes számok függő eliminátora (indukciós elve) azt mondja, hogy tetszőleges függő algebrába megy egy függő homomorfizmus* $(Nat, zero, suc)$*-ból.*

Ha az iterátort kiegészítjük azzal, hogy pontosan *egy homomorfizmus megy tetszőleges algebrába, az ekvivalens a függő eliminátorral. A különböző induktív típusokra ezzel teljesen analóg módon adjuk meg az iterátort és eliminátort.*

Az alábbi táblázat listáz néhány algebrai elmélet-osztályt és az ezekkel ekvivalens induktív típus-osztályt.

Egyszortú algebrai elmélet egyenlőségek nélkül	Egyszerű induktív típus
Többszortú algebrai elmélet egyenlőségek nélkül	Kölcsönösen megadott induktív típusok
Egyszortú algebrai elmélet	Kvóciens-induktív típus (QIT)
Általánosított algebrai elmélet egyenlőségek nélkül	Induktív-induktív típus (IIT)
Általánosított algebrai elmélet (GAT)	Kvóciens induktív-induktív típus (QIIT)
Magasabb általánosított algebrai elmélet	Magasabb induktív-induktív típus (HIIT)

Minden algebrai elmélet-osztályhoz tartozik egy univerzális algebra, mely meghatározza a következő fogalmakat: szignatúra, algebra (modell), homomorfizmus, algebrák szorzata, szabad algebrák stb. Szignatúrába mi nem csak az operátorok aritásait értjük bele, hanem a fajtákat és az egyenlőségeket is, mert általánosított algebrai elméletek esetén ezek nem különíthetők el (például lehetnek olyan operátorok, melyek egyenlőségektől függenek).

Ebben a fejezetben a *szignatúrák elmélete* (theory of signatures) módszert ismertetjük: egy adott induktív típus/algebrai elmélet-osztály egy alkalmazásspecifikus típuselméletnek felel meg. Ez egy kisméretű típuselmélet, amely kizárólag a szignatúrák leírására szolgál. A szignatúrák elméletének szemantikája adja meg az univerzális algebra további komponenseit. Egy típuselmélet (még általánosabban egy kötésekkel rendelkező formális nyelv) egy másodrendű általánosított algebrai elmélet (lásd 2.7. alfejezet), a nyelv szintaxisa ennek iniciális algebrája. A legegyszerűbb induktív típus-osztályt részletesebben ismertetjük, hogy bemutassuk az alapötletet, a többi osztály esetében csak a szignatúra-definíciót magát és az eredményeket ismertetjük.

Mi értelme van külön osztályokat vizsgálni, miért nem csak a legáltalánosabb osztállyal foglalkozunk? Bizonyos programozási nyelvekben, típuselmélet-modellekben csak bizonyos osztályba tartozó induktív típusok vannak; kevésbé általános osztályok létezéséhez egy modellben kevesebb feltételre van szükség; néha szeretnénk tudni, hogy egy adott algebrai elmélet milyen komplex (melyik a legegyszerűbb osztály, melybe befér).

2.1. Egyszerű induktív típusok

A szignatúrák elmélete módszert először a legegyszerűbb induktív típus-osztályon mutatjuk be.

2. Definíció (Egyszerű zárt végesen elágazó induktív típusok, más néven egyszortú egyenlőségek nélküli algebrai elméletek)**.** *Egy szignatúra a következő induktív típus egy eleme.*

$$
\begin{aligned}
&\mathsf{Ty} && : \mathsf{Type}\\
&\top && : \mathsf{Ty}\\
&\mathsf{C} && : \mathsf{Ty}\\
&- \times - && : \mathsf{Ty} \to \mathsf{Ty} \to \mathsf{Ty}\\
&\mathsf{C} \Rightarrow - && : \mathsf{Ty} \to \mathsf{Ty}
\end{aligned}
$$

A fenti induktív definíció egy nagyon egyszerű nyelvet (elméletet, típuselméletet, szignatúrák elméletét) ad meg: ebben a nyelvben csak típusok (Ty) vannak, termek nincsenek. Ne tévessze meg az olvasót, hogy vannak metanyelvi típusaink (Type elemei) és tárgynyelvi típusaink is (Ty elemei). Ty-nak van egy fix eleme, a C, ami az induktív típusunk egyetlen fajtáját (carrier) adja meg. Ty zárt a szorzatokra és egy olyan függvénytérre, amelynek értelmezési tartománya C (így

zárjuk ki a nem szigorúan pozitív konstruktorokat). Ty egy eleme egy szignatúra. Listázunk néhány példa szignatúrát.

Üres típus	$\top$
Egyelemű típus	C
Bool	$\mathsf{C} \times \mathsf{C}$
Négyelemű típus	$\mathsf{C} \times \mathsf{C} \times \mathsf{C} \times \mathsf{C}$
Peano természetes számok	$\mathsf{C} \times (\mathsf{C} \Rightarrow \mathsf{C})$
Bináris fák	$\mathsf{C} \times (\mathsf{C} \Rightarrow \mathsf{C} \Rightarrow \mathsf{C})$
Egyszerű zárt végesen elágazó induktív típusok	$\mathsf{C} \times \mathsf{C} \times (\mathsf{C} \Rightarrow \mathsf{C} \Rightarrow \mathsf{C}) \times (\mathsf{C} \Rightarrow \mathsf{C})$

Az utolsó példa mutatja, hogy ezek az induktív típusok le tudják önmagukat is írni.

Egy $A : \mathsf{Ty}$ szignatúrához tartozó algebra-fogalom az $(X : \mathsf{Type}) \times [\![A]\!]\, X$, ahol $[\![-]\!]$ a következőképp van megadva Ty szerinti rekurzióval.

$$
\begin{aligned}
&[\![-]\!] : \mathsf{Ty} \to \mathsf{Type} \to \mathsf{Type} \\
&[\![\top]\!] \quad X := \mathbb{1} \\
&[\![\mathsf{C}]\!] \quad X := X \\
&[\![A \times B]\!] \quad X := [\![A]\!]\, X \times [\![B]\!]\, X \\
&[\![\mathsf{C} \Rightarrow A]\!]\, X := X \to [\![A]\!]\, X
\end{aligned}
$$

Például a következő módon számítjuk ki a Bool-ra illetve természetes számokra az algebra-fogalmat:

$$(X : \mathsf{Type}) \times [\![\mathsf{C} \times \mathsf{C}]\!]\, X = (X : \mathsf{Type}) \times X \times X,$$

$$(X : \mathsf{Type}) \times [\![\mathsf{C} \times (\mathsf{C} \Rightarrow \mathsf{C})]\!]\, X = (X : \mathsf{Type}) \times X \times (X \to X).$$

Két A-algebra (X_0, a_1) és (X_1, a_1) közötti homomorfizmus-fogalmat az $(f : X_0 \to X_1) \times [\![A]\!]^\bullet \bullet f\, a_0\, a_1$ adja meg, ahol $[\![-]\!]^\bullet$ a következő.

$$
\begin{aligned}
&[\![-]\!]^\bullet : (A : \mathsf{Ty}) \to (f : X_0 \to X_1) \to [\![A]\!]\, X_0 \to [\![A]\!]\, X_1 \to \mathsf{Type} \\
&[\![\top]\!]^\bullet \quad f\, * \quad * \quad := \mathbb{1} \\
&[\![\mathsf{C}]\!]^\bullet \quad f\, x_0 \quad x_1 \quad := (f\, x_0 = x_1) \\
&[\![A \times B]\!]^\bullet \quad f\, (a_0, b_0)\, (a_1, b_1) := [\![A]\!]^\bullet\, f\, a_0\, a_1 \times [\![B]\!]^\bullet\, f\, b_0\, b_1 \\
&[\![\mathsf{C} \Rightarrow A]\!]^\bullet\, f\, \alpha_0 \quad \alpha_1 \quad := (x_0 : X_0) \to [\![A]\!]^\bullet\, f\, (\alpha_0\, x_0)\, (\alpha_1\, (f\, x_0))
\end{aligned}
$$

Például vegyük az (X_0, z_0, s_0) ill. (X_1, z_1, s_1) természetes szám-algebrákat, és nézzük meg, mi lesz közöttük egy homomorfizmus (egy függvény a két típus között, mely megőrzi a zero és a suc műveleteket).

$$
\begin{aligned}
&(f : X_0 \to X_1) \times [\![\mathsf{C} \times (\mathsf{C} \Rightarrow \mathsf{C})]\!]^\bullet\, f\, (z_0, s_0)\, (z_1, s_1) \quad = \\
&(f : X_0 \to X_1) \times [\![\mathsf{C}]\!]^\bullet\, f\, z_0\, z_1 \times [\![\mathsf{C} \Rightarrow \mathsf{C}]\!]^\bullet\, f\, s_0\, s_1 \quad = \\
&(f : X_0 \to X_1) \times (f\, z_0 = z_1) \times \Big((x_0 : X_0) \to [\![\mathsf{C}]\!]^\bullet\, f\, (s_0\, x_0)\, \big(s_1\, (f\, x_0)\big) \Big) = \\
&(f : X_0 \to X_1) \times (f\, z_0 = z_1) \times \Big((x_0 : X_0) \to f\, (s_0\, x_0) = \big(s_1\, (f\, x_0)\big) \Big)
\end{aligned}
$$

Hasonló módszerrel (szignatúra szerinti indukcióval) megadható a homomorfizmusok kompozíciója, az identitás homomorfizmus, a homomorfizmusok kompozíciójának asszociativitása, a

függő algebra fogalma, egy algebrából egy fölötte levő függő algebrába menő függő homomorfizmus (szekció) fogalma, az iniciális algebra stb. Összefoglalva: minden szignatúrához kapunk egy családos kategóriát véges limitekkel és iniciális objektummal (finite limit category with families with initial object). A családos kategória [CCD21] a típuselmélet egy modell-fogalma, a véges limitek azt jelentik, hogy a modellben van egyelemű típus, függő Descartes-szorzat és egyenlőség típus. Így minden szignatúrára kapunk egy típuselmélet modellt, melyben a típuselmélet belső nyelvén dolgozhatunk, a típusok (függő) algebráknak, a függvények homomorfizmusoknak felelnek meg, a típusok szorzata az algebrák szorzata, és például ezen az absztrakt módon bebizonyítható, hogy egy algebrára az indukció fogalma ekvivalens az inicialitással.

Kapcsolódó munkák. Ebben az alfejezetben a legegyszerűbb induktív típus-osztályt (egyenlőségek nélküli algebrai elméletek osztályát) ismertettük, és az itt közölt eredmények nem újak. Egyszerű induktív típusok leírására sokféle módszer létezik (pl. [AAG05; Har16; Hug20]). A bemutatott módszer azért érdekes, mert skálázódik általánosabb induktív típus-osztályokra, melyekre már a hasonló eredmények újak. A szignatúrák elmélete módszer egy független előzménye [CO12], melyben egy szignatúra egy függő típuselmélet-beli környezetként van megadva.

2.2. Kölcsönös induktív családok

A 2.1. alfejezetben bemutatott induktív típusok osztályát ebben a fejezetben kibővítjük indexelt induktív családokkal és kölcsönesen megadott induktív típusokkal. Előbbivel lehet az utóbbiakat is modellezni, de a mi szignatúráink lehetőséget adnak az utóbbiakat közvetlenül is megadni. A szignatúrák nyelve a következő.

3. Definíció (Kölcsönös induktív típusok, más néven többszortú egyenlőségek nélküli algebrai elméletek)**.** *Tekintsük az alábbi három induktív típust.*

$$
\begin{array}{lll}
\mathsf{Ty_S} & : \mathsf{Type} & \\
\mathsf{U} & : \mathsf{Ty_S} & \textit{rövidítések:} \\
\hat{\Pi}_\mathsf{S} & : (T : \mathsf{Type}) \to (T \to \mathsf{Ty_S}) \to \mathsf{Ty_S} & T \mathrel{\hat{\Rightarrow}_\mathsf{S}} A := \hat{\Pi}_\mathsf{S}\, T\, (\lambda\_ .A) \\
- \times_\mathsf{S} - & : \mathsf{Ty_S} \to \mathsf{Ty_S} \to \mathsf{Ty_S} & \\
\mathsf{Tm_S} & : \mathsf{Ty_S} \to \mathsf{Type} & \\
- \cdot - & : \mathsf{Tm_S}\, (\hat{\Pi}_\mathsf{S}\, T\, A) \to (t : T) \to \mathsf{Tm_S}\, (A\, t) & \\
\mathsf{fst} & : \mathsf{Tm_S}\, (A \times_\mathsf{S} B) \to \mathsf{Tm_S}\, A & \\
\mathsf{snd} & : \mathsf{Tm_S}\, (A \times_\mathsf{S} B) \to \mathsf{Tm_S}\, B & \\
\mathsf{Ty_P} & : \mathsf{Type} & \\
\mathsf{El} & : \mathsf{Tm_S}\, \mathsf{U} \to \mathsf{Ty_P} & \\
\hat{\Pi}_\mathsf{P} & : (T : \mathsf{Type}) \to (T \to \mathsf{Ty_P}) \to \mathsf{Ty_P} & T \mathrel{\hat{\Rightarrow}_\mathsf{P}} B := \hat{\Pi}_\mathsf{P}\, T\, (\lambda\_ .B) \\
- \Rightarrow_\mathsf{P} - & : \mathsf{Tm_S}\, \mathsf{U} \to \mathsf{Ty_P} \to \mathsf{Ty_P} & \\
- \times_\mathsf{P} - & : \mathsf{Ty_P} \to \mathsf{Ty_P} \to \mathsf{Ty_P} &
\end{array}
$$

Egy szignatúra a $(A : \mathsf{Ty_S}) \times (\mathsf{Tm_S}\, A \to \mathsf{Ty_P})$ *típus egy eleme.*

A fenti szignatúrák definíciója bonyolultabb, mint a náluk általánosabb, induktív-induktív típusok szignatúráié, ez azért van, mert több megkötésünk van: például el kell különíteni a fajtákat (sort-ok, $\mathsf{Ty_S}$ elemei) a konstruktoroktól (pont-konstruktorok, $\mathsf{Ty_P}$ elemei). Kizárólag fajta-termekre ($\mathsf{Tm_S}$) van szükségünk a szignatúrákban. A $\hat{\Pi}$ függvény-típusoknak metanyelvi az értelmezési tartomány és tárgynyelvi az értékkészlete, ezekkel lehet indexelni a fajtákat és a konstruktorokat.

Alább leírjuk a természetes számokat, vektorokat, és a páros-páratlan kölcsönösen megadott predikátumokat a fenti szignatúrákkal (bal oldalon) és informálisan is (jobb oldalon; az informális szignatúra megegyezik az algebra-fogalommal). Utóbbi két példához feltételezünk a metanyelvben természetes számokat illetve egy rögzített $A : \mathsf{Type}$ típust.

$(\mathsf{U}, \lambda N.$	$N : \mathsf{Type},$
$\mathsf{El}\, N \times_{\mathsf{P}} (N \Rightarrow_{\mathsf{P}} \mathsf{El}\, N))$	$zero : N,\ suc : N \to N$
$\Big(\mathbb{N} \hat{\Rightarrow}_{\mathsf{S}} \mathsf{U}, \lambda Vec.$	$Vec : \mathbb{N} \to \mathsf{Type},$
$\mathsf{El}\,(Vec \cdot 0) \times_{\mathsf{P}}$	$nil : Vec\, 0,$
$A \hat{\Rightarrow}_{\mathsf{P}} \hat{\Pi}_{\mathsf{P}}\, \mathbb{N}\, \lambda n. Vec \cdot n \Rightarrow_{\mathsf{P}} \mathsf{El}\,(Vec \cdot (1+n))\Big)$	$cons : A \to (n : \mathbb{N}) \to Vec\, n \to Vec\,(1+n)$
$\Big((\mathbb{N} \hat{\Rightarrow}_{\mathsf{S}} \mathsf{U}) \times_{\mathsf{S}} (\mathbb{N} \hat{\Rightarrow}_{\mathsf{S}} \mathsf{U}), \lambda X.$	$isEven : \mathbb{N} \to \mathsf{Type},\ isOdd : \mathbb{N} \to \mathsf{Type},$
$\mathsf{El}\,(\mathsf{fst}\, X \cdot 0) \times_{\mathsf{P}}$	$even0 : isEven\, 0,$
$\big(\hat{\Pi}_{\mathsf{P}}\, \mathbb{N}\, \lambda n. \mathsf{snd}\, X \cdot n \Rightarrow_{\mathsf{P}} \mathsf{fst}\, X \cdot (1+n)\big) \times_{\mathsf{P}}$	$even+ : (n : \mathbb{N}) \to isOdd\, n \to isEven\,(1+n),$
$\big(\hat{\Pi}_{\mathsf{P}}\, \mathbb{N}\, \lambda n. \mathsf{fst}\, X \cdot n \Rightarrow_{\mathsf{P}} \mathsf{snd}\, X \cdot (1+n)\big)\Big)$	$odd+ : (n : \mathbb{N}) \to isEven\, n \to isOdd\,(1+n)$

A kölcsönös induktív típusok szignatúrái leírhatók több, egymás után megadott induktív családdal, de közvetlenül egy darab kölcsönösen megadott induktív típussal nem.

A 3. definíciónak megadtuk az explicit környezetekkel rendelkező változatát, annak szemantikáját (algebrák, homomorfizmusok, függő algebrák, függő homomorfizmusok), és megmutattuk, hogy a szignatúrák induktív típusának a feltételezésével minden kölcsönös induktív típus megadható. Tehát a kölcsönös induktív típusok szignatúrája valójában az *univerzális kölcsönös induktív típus*. A szignatúrák típusát visszavezettük W-típusokra [AAG05], ezzel megmutatva, hogy az összes indexelt típus megadható W-típusokkal. Az eredményeket Jakob von Raumer-rel közösen publikáltuk [KR20].

Kapcsolódó munkák. A kölcsönös induktív típusok visszavezethetők indexelt induktív típusokra, de a mi leírásunk volt az első, amely közvetlenül megadott egy nyelvet kölcsönös induktív típusokra. Az indexelt típusokat korábban külső sémákkal [Dyb94; Pau93; Ana+18], belső kombinatorikus sémákkal [BDJ03; Cha+10; All+21] és belső szemantikus sémákkal adták meg [PS89; Alt+15]

2.3. Induktív-induktív típusok

4. Definíció (Zárt induktív-induktív típusok, más néven zárt egyenlőségek nélküli általánosított algebrai elméletek)**.** *Tekintsük az alábbi másodrendű általánosított algebrai elméletet. Egy szignatúra az alábbi* Ty *egy eleme.*

$$
\begin{array}{lll}
\mathsf{Ty} & : \mathsf{Type} & \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Type} & \\
\Sigma & : (A : \mathsf{Ty}) \to (\mathsf{Tm}\, A \to \mathsf{Ty}) \to \mathsf{Ty} & \\
\mathsf{U} & : \mathsf{Ty} & \\
\mathsf{El} & : \mathsf{Tm}\, \mathsf{U} \to \mathsf{Ty} & \textit{rövidítés:} \\
\Pi & : (a : \mathsf{Tm}\, \mathsf{U}) \to (\mathsf{Tm}\,(\mathsf{El}\, a) \to \mathsf{Ty}) \to \mathsf{Ty} & a \Rightarrow B := \Pi\, a\, (\lambda \_ . B) \\
\_ \cdot \_ & : \mathsf{Tm}\,(\Pi\, a\, B) \to (u : \mathsf{Tm}\,(\mathsf{El}\, a)) \to \mathsf{Tm}\,(B\, u) &
\end{array}
$$

Úgy is mondhatjuk, hogy egy szignatúra egy típus egy kisméretű típuselméletben, melyben csak az alábbi típusok vannak: Σ, egy üres univerzum és U-beli értelmezési tartományú függvénytér; U-nak és Π-nek csak eliminátorai vannak, konstruktorai nincsenek, Σ-nak meg egyik sincs. Tehát ez egy egyenlőségek (számítási szabályok) nélküli típuselmélet.

Míg a korábbi szignatúrák induktív típusként voltak megadva, az induktív-induktív típusok szignatúráit könnyebb másodrendű algebrai elméletként megadni. A másodrendű algebrai elméleteket le lehet fordítani elsőrendűre (lásd 2.7. fejezet), és azok már induktív típusoknak felelnek meg. Tehát valójában egy szignatúra a fenti algebrai elmélet elsőrendűre való lefordítása eredményeként kapott elmélet iniciális algebrájában Ty egy eleme. A fenti definíció is értelmes azonban: ha csak feltételezzük a fenti másodrendű algebrai elmélet egy (másodrendű) modelljét, és annak az elemeivel hozunk létre egy Ty-t, az (a típuselmélet egy előkéve modelljének belső nyelvében) pontosan meg fog egyezni az elsőrendű fordítás után megadott Ty egy elemének.

Induktív-induktív típusok az (üres) gráfok, az alábbi formális ill. informális szignatúrával.

$$\Sigma\,\mathsf{U}\,\lambda Vertex.Vertex \Rightarrow Vertex \Rightarrow \mathsf{U} \qquad\qquad Vertex : \mathsf{Type},\ Edge : Vertex \to Vertex \to \mathsf{Type}$$

Egy bonyolultabb példa a típuselmélet elsőrendű szintaxisának alábbi részlete. Csak környezetek (Con) és típusok (Ty) vannak, a környezet típusok függő listája, van egy univerzumunk (U, El) és függő függvény típusunk. Bal oldalt a formális, jobb oldalt az informális szignatúra.

$$\begin{aligned}
&\Sigma\,\mathsf{U}\,\lambda Con. && Con &&: \mathsf{Type},\\
&\Sigma\,(Con \Rightarrow \mathsf{U})\,\lambda Ty. && Ty &&: Con \to \mathsf{Type},\\
&\Sigma\,(\mathsf{El}\,Con)\,\lambda empty. && \diamond &&: Con,\\
&\Sigma\,(\Pi\,Con\,\lambda\Gamma.Ty\cdot\Gamma \Rightarrow \mathsf{El}\,Con)\,\lambda ext. && -\triangleright- &&: (\Gamma : Con) \to Ty\,\Gamma \to Con,\\
&\Sigma\,(\Pi\,Con\,\lambda\Gamma.\mathsf{El}\,(Ty\cdot\Gamma))\,\lambda U. && U &&: (\Gamma : Con) \to Ty\,\Gamma,\\
&\Sigma\,(\Pi\,Con\,\lambda\Gamma.\mathsf{El}\,(Ty\cdot(ext\cdot\Gamma\cdot(U\cdot\Gamma))))\,\lambda El. && El &&: (\Gamma : Con) \to Ty\,(\Gamma\triangleright U\,\Gamma),\\
&\quad \Pi\,Con\,\lambda\Gamma.\Pi\,(Ty\cdot\Gamma)\,\lambda A.Ty\cdot(ext\cdot\Gamma\cdot A) \Rightarrow && \Pi &&: (\Gamma : Con)(A : Ty\,\Gamma) \to Ty\,(\Gamma\triangleright A) \to\\
&\qquad\qquad \mathsf{El}\,(Ty\cdot\Gamma) && && \qquad\qquad Ty\,\Gamma.
\end{aligned}$$

A fenti szignatúra-fogalomnak megadtuk a metaelméleti függvénytérrel kiegészített elsőrendű változatát, mely támogatja a nyílt induktív-induktív típusokat is (például a 2.2. alfejezetben megadott vektorok és páros-páratlan predikátumok nyíltak, mert hivatkoznak a természetes számokra). Megmutattuk, hogy a szignatúrák elmélete létrehozható egyszerű induktív típusokkal (Streicher módszerével [Str91]), az összes végesen elágazódó induktív-induktív típus visszavezethető a szignatúrák típusára (mely ezáltal egy univerzális induktív-induktív típus), így a végesen elágazódó induktív-induktív típusok visszavezethetők egyszerű induktív típusokra. Az eredményeket Ambroise Lafont-tal és Kovács Andrással közösen publikáltuk [KKL20].

Kapcsolódó munkák. Az induktív-induktív típusokat az Agda bizonyítórendszer már azelőtt támogatta [Cha08], mielőtt a kifejezés megszületett. Nordvall Forsberg és Setzer írták le őket először [FS10], és Hugunin [Hug19] vizsgálta meg, hogy milyen feltételek mellett konstruálhatók meg homotópia-típuselméletben. A mi leírásunk volt az első közvetlen (elkódolás nélküli) leírás, és az első általános konstrukció, mely az összes induktív-induktív típus létezését visszavezette induktív típusokra (Nordvall Forsberg és Hugunin nem bizonyította az általános indukciós elvet az általuk konstruált induktív-induktív típusokra). Azóta Sestini [Ses23] általánosította az eredményeinket: a konstrukció működik végtelenül elágazó induktív-induktív típusokra is, és mindez egy intenzionális metanyelvben is működik (a mi eredményeink extenzionális metanyelvre vonatkoznak).

2.4. Kvóciens induktív típusok

5. Definíció (Zárt kvóciens induktív típusok, más néven zárt egyszortú algebrai elméletek)**.** *Egy szignatúra az alábbi* Ty *egy eleme.*

$$\mathsf{Ty} \quad : \mathsf{Type}$$

$$
\begin{array}{ll}
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Type} \\
\Sigma & : (A : \mathsf{Ty}) \to (\mathsf{Tm}\, A \to \mathsf{Ty}) \to \mathsf{Ty} \\
\mathsf{C} & : \mathsf{Ty} \\
\Pi\mathsf{C} & : (\mathsf{Tm}\, \mathsf{C} \to \mathsf{Ty}) \to \mathsf{Ty} \\
- \cdot - & : \mathsf{Tm}\, (\Pi\mathsf{C}\, B) \to (x : \mathsf{Tm}\, \mathsf{C}) \to \mathsf{Tm}\, (B\, x) \\
\mathsf{Eq} & : \mathsf{Tm}\, \mathsf{C} \to \mathsf{Tm}\, \mathsf{C} \to \mathsf{Ty}
\end{array}
\qquad \textit{rövidítés:} \quad \mathsf{C} \Rightarrow A := \Pi\mathsf{C}\, (\lambda\_ .A)
$$

Ezek a szignatúrák is másodrendű algebrai elméletként vannak megadva a kényelem kedvéért. Egy univerzális algebra tankönyvben általában elsőrendű induktív definíciókat találunk, de használhatjuk a 2.7. fejezetben megadott módszert is arra, hogy elsőrendű szignatúra-fogalmat kapjunk.

A csoport szignatúrája a következő.

$$
\begin{array}{ll}
 & C \quad : \mathsf{Type} \\
\Sigma\, (\mathsf{C} \Rightarrow \mathsf{C} \Rightarrow \mathsf{C})\, \lambda op. & - \otimes - \; : C \to C \to C \\
\Sigma\, \big(\Pi\mathsf{C}\, \lambda x.\Pi\mathsf{C}\, \lambda y.\Pi\mathsf{C}\, \lambda z. & ass \quad : (x\, y\, z : C) \to \\
\quad \mathsf{Eq}\, (op \cdot (op \cdot x \cdot y) \cdot z)\, (op \cdot x \cdot (op \cdot x \cdot y))\big)\, \lambda ass. & \qquad (x \otimes y) \otimes z = x \otimes (y \otimes z) \\
\Sigma\, \mathsf{C}\, \lambda u. & 1 \quad : C \\
\Sigma\, (\Pi\mathsf{C}\, \lambda x.\mathsf{Eq}\, (op \cdot u \cdot x)\, x)\, \lambda idl. & idl \quad : (x : C) \to 1 \otimes x = x \\
\Sigma\, (\Pi\mathsf{C}\, \lambda x.\mathsf{Eq}\, (op \cdot x \cdot u)\, x)\, \lambda idr. & idr \quad : (x : C) \to x \otimes 1 = x \\
\Sigma\, (\mathsf{C} \Rightarrow \mathsf{C})\, \lambda inv. & -^{-1} \quad : C \to C \\
\Sigma\, (\Pi\mathsf{C}\, \lambda x.\mathsf{Eq}\, (op \cdot (inv \cdot x) \cdot x)\, u)\, \lambda invl. & invl \quad : (x : C) \to x^{-1} \otimes x = 1 \\
\quad \Pi\mathsf{C}\, \lambda x.\mathsf{Eq}\, (op \cdot x \cdot (inv \cdot x))\, u & invl \quad : (x : C) \to x \otimes x^{-1} = 1
\end{array}
$$

Kapcsolódó munkák. A fenti szignatúrákat más módszerrel leírta Fiore, Pitts és Steenkamp [FPS22]. Az ő változatuk támogatja a nyílt illetve végtelenül elágazó kvóciens induktív típusokat is.

2.5. Kvóciens induktív-induktív típusok

6. Definíció (Zárt kvóciens induktív-induktív típusok, más néven zárt általánosított algebrai elméletek)**.** *Tekintsük az alábbi másodrendű általánosított algebrai elméletet. Egy szignatúra az alábbi* Ty *egy eleme.*

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Type} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Type} \\
\Sigma & : (A : \mathsf{Ty}) \to (\mathsf{Tm}\, A \to \mathsf{Ty}) \to \mathsf{Ty} \\
\mathsf{U} & : \mathsf{Ty} \\
\mathsf{El} & : \mathsf{Tm}\, \mathsf{U} \to \mathsf{Ty} \\
\Pi & : (a : \mathsf{Tm}\, \mathsf{U}) \to (\mathsf{Tm}\, (\mathsf{El}\, a) \to \mathsf{Ty}) \to \mathsf{Ty} \\
- \cdot - & : \mathsf{Tm}\, (\Pi\, a\, B) \to (u : \mathsf{Tm}\, (\mathsf{El}\, a)) \to \mathsf{Tm}\, (B\, u) \\
\mathsf{Eq} & : \mathsf{Tm}\, A \to \mathsf{Tm}\, A \to \mathsf{Ty} \\
\mathsf{reflect} & : \mathsf{Tm}\, (\mathsf{Eq}\, u\, v) \to u = v
\end{array}
\qquad \textit{rövidítés:} \quad a \Rightarrow B := \Pi\, a\, (\lambda\_ .B)
$$

A kategóriák szignatúrája kvóciens induktív-induktív típus. A jobb oldali informális szignatúrában a kompozíció első három paraméterét impliciten írjuk; a formális szignatúrákban nincsenek implicit paraméterek.

$$
\Sigma\, \mathsf{U}\, \lambda Ob. \qquad Ob \quad : \mathsf{Type},
$$

$$
\begin{array}{ll}
\Sigma\,(Ob \Rightarrow Ob \Rightarrow \mathsf{U})\,\lambda Hom. & Hom\ : Ob \to Ob \to \mathsf{Type},\\
\Sigma\,\big(\Pi\,Ob\,\lambda I.\Pi\,Ob\,\lambda J.\Pi\,Ob\,\lambda K.Hom\cdot J\cdot I \Rightarrow & -\circ-\ : \{I\,J\,K : Ob\} \to Hom\,J\,I \to\\
\quad Hom\cdot K\cdot J \Rightarrow \mathsf{El}\,(Hom\cdot K\cdot I)\big)\,\lambda comp. & \qquad Hom\,K\,J \to Hom\,K\,I\\
\Sigma\,\Big(\Pi\,Ob\,\lambda I.\Pi\,Ob\,\lambda J.\Pi\,Ob\,\lambda K.\Pi\,\lambda L.\Pi\,(Hom\cdot J\cdot I)\,\lambda f. & ass\ : (I\,J\,K\,L : Ob)(f : Hom\,J\,I) \to\\
\quad \Pi\,(Hom\cdot K\cdot J)\,\lambda g.\Pi\,(Hom\cdot L\cdot K)\,\lambda h. & \qquad (g : Hom\,K\,J)(h : Hom\,L\,K) \to\\
\quad \mathsf{Eq}\,\big(comp\cdot I\cdot K\cdot L\cdot(comp\cdot I\cdot J\cdot K\cdot f\cdot g)\cdot h\big) & \qquad (f\circ g)\circ h =\\
\qquad \big(comp\cdot I\cdot J\cdot L\cdot f\cdot(comp\cdot J\cdot K\cdot L\cdot g\cdot h)\big)\Big)\,\lambda ass. & \qquad f\circ(g\circ h)\\
\Sigma\,(\Pi\,Ob\,\lambda I.\mathsf{El}\,(Hom\cdot I\cdot I))\,\lambda id. & id\ : (I : Ob) \to Hom\,I\,I\\
\Sigma\,\big(\Pi\,Ob\,\lambda I.\Pi\,Ob\,\lambda J.\Pi\,(Hom\cdot J\cdot I)\,\lambda f. & idl\ : (I\,J : Ob)(f : Hom\,J\,I) \to\\
\quad \mathsf{Eq}\,(comp\cdot I\cdot I\cdot J\cdot(id\cdot I)\cdot f)\,f\big)\,\lambda idl. & \qquad id\,I\circ f = f\\
\quad \Pi\,Ob\,\lambda I.\Pi\,Ob\,\lambda J.\Pi\,(Hom\cdot J\cdot I)\,\lambda f. & idr\ : (I\,J : Ob)(f : Hom\,J\,I) \to\\
\quad \mathsf{Eq}\,(comp\cdot I\cdot J\cdot J\cdot f\cdot(id\cdot I))\,f & \qquad f\circ id\,I = f
\end{array}
$$

További példák különböző programozási nyelvek szintaxisai, lásd a 4. fejezetben. Az egyenlőségre eliminátorára (reflect) azért van szükség, mert kvóciens induktív-induktív típusoknál gyakran előfordul, hogy egy egyenlőség (vagy egy konstruktor) csak korábbi egyenlőségek miatt értelmes. Egy egyszerű példa egy típuselmélet szignatúrájának következő részlete, informálisan:

$$
\begin{array}{ll}
Con & : \mathsf{Type},\\
Ty & : Con \to \mathsf{Type}\\
-\triangleright- & : (\Gamma : Con) \to Ty\,\Gamma \to Con\\
U & : (\Gamma : Con) \to Ty\,\Gamma\\
El & : (\Gamma : Con) \to Ty\,(\Gamma \triangleright U\,\Gamma)\\
wk_0 & : (A : Ty\,\Gamma) \to Ty\,\Gamma \to Ty\,(\Gamma \triangleright A)\\
wk_1 & : (A : Ty\,\Gamma) \to Ty\,(\Gamma \triangleright B) \to Ty\,(\Gamma \triangleright A \triangleright wk_0\,A\,B)\\
wkU & : wk_0\,A\,(U\,\Gamma) = U\,(\Gamma \triangleright A)\\
wkEl & : wk_1\,A\,(El\,\Gamma) = El\,(\Gamma \triangleright A)
\end{array}
$$

A 2.3. fejezetben megadott típuselmélet-részletet kiegészítettük két gyengítés (weakening) művelettel, az egyik a környezet (Con) legvégére, a másik az utolsó előtti helyre tesz be egy újabb típust. A két egyenlőség (wkU és $wkEl$) elmondják, mi történik, ha egy U illetve El típust gyengítünk. Utóbbi egyenlőség két oldalán azonban különböző (meta-)típusú kifejezések vannak:

$$wk_1\,A\,(El\,\Gamma) : Ty\,(\Gamma \triangleright A \triangleright wk_0\,A\,(U\,\Gamma)) \qquad El\,(\Gamma \triangleright A) : Ty\,(\Gamma \triangleright A \triangleright U\,(\Gamma \triangleright A)).$$

Ez azért nem probléma, mert a wkU egyenlőség pont azt mondja, hogy ez a két (meta-)típus megegyezik. Ha teljesen formálisan írjuk ezt le, akkor a $wkEl$ egyenlőség kimondásakor fel kell használni wkU-t.

A fenti szignatúrák önleíróak, tehát a szignatúrák elmélete is megadható szignatúrával.

A fenti szignatúra-fogalomnak megadtuk a metaelméleti függvénytérrel kiegészített elsőrendű változatát, mely támogatja a nyílt kvóciens induktív-induktív típusokat is. Megmutattuk, hogy minden szignatúra meghatároz egy véges limites családos kategóriát, és hogy a szignatúrák elméletének szintaxisából le tudjuk vezetni bármely kvóciens-induktív-induktív típus iniciális algebráját. Az eredményeket Thorsten Altenkirch-el és Kovács Andrással publikáltuk [KKA19]. Kovács Andrással közösen mindezen eredményeket általánosítottuk végtelenül elágazódó kvóciens-induktív-induktív típusokra és egyenlőség-paraméterekre (kvázi-varietásokra) [KK20b].

Kapcsolódó munkák. Az első példa kvóciens induktív-induktív típusra (bár nem ezen a néven) a Cauchy-valós számok definíciója volt [Pro13]. A kvóciens induktív-induktív típus kifejezést Altenkirch és Kaposi vezette be a típuselmélet szintaxisának példáján keresztül [AK16]. Az első szemantikus leírást Altenkirch és mtsai. adták meg [Alt+18], ők még nem tudták körülírni a használható szignatúrákat, és nem bizonyították iniciális algebrák létezését. Kovács András doktori disszertációja [Kov22] megmutatja, hogyan adhatók meg általános szabad algebrák, míg Hugo Moeneclaey disszertációja [Moe22] megad feltételeket a ko-szabad algebrák létezésére. További alkalmazások a parcialitás monad [ADK17] és a hibrid szemantika [DG20]. Az algebrai oldalon az általánosított algebrai elméleteket Cartmell adta meg [Car86; Bez+21], ezekkel ekvivalensek az esszenciális algebrai elméletek [Fre72].

2.6. Magasabb induktív-induktív típusok

7. Definíció (Zárt magasabb induktív-induktív típusok, más néven zárt magasabb általánosított algebrai elméletek)**.** *Tekintsük az alábbi másodrendű általánosított algebrai elméletet. Egy szignatúra az alábbi* Ty *egy eleme.*

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Type} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Type} \\
\Sigma & : (A : \mathsf{Ty}) \to (\mathsf{Tm}\, A \to \mathsf{Ty}) \to \mathsf{Ty} \\
\mathsf{U} & : \mathsf{Ty} \\
\mathsf{El} & : \mathsf{Tm}\, \mathsf{U} \to \mathsf{Ty} \\
\Pi & : (a : \mathsf{Tm}\, \mathsf{U}) \to (\mathsf{Tm}\, (\mathsf{El}\, a) \to \mathsf{Ty}) \to \mathsf{Ty} \\
- \cdot - & : \mathsf{Tm}\, (\Pi\, a\, B) \to (u : \mathsf{Tm}\, (\mathsf{El}\, a)) \to \mathsf{Tm}\, (B\, u) \\
\mathsf{ID} & : \mathsf{Tm}\, A \to \mathsf{Tm}\, A \to \mathsf{Ty} \\
\mathsf{refl} & : (t : \mathsf{Tm}\, A) \to \mathsf{Tm}\, (\mathsf{ID}\, t\, t) \\
\mathsf{J} & : (P : (y : \mathsf{Tm}\, A) \to \mathsf{Tm}\, (\mathsf{ID}\, x\, y) \to \mathsf{Ty}) \to \mathsf{Tm}\, (P\, x\, (\mathsf{refl}\, x)) \to \\
 & \quad (y : \mathsf{Tm}\, A)(e : \mathsf{Tm}\, (\mathsf{ID}\, x\, y)) \to \mathsf{Tm}\, (P\, y\, e) \\
\mathsf{J}\beta & : \mathsf{J}\, P\, pr\, (\mathsf{refl}\, x) = pr \\
\mathsf{Id} & : \mathsf{Tm}\, (\mathsf{El}\, a) \to \mathsf{Tm}\, (\mathsf{El}\, a) \to \mathsf{Tm}\, \mathsf{U} \\
\mathsf{refl} & : (t : \mathsf{Tm}\, (\mathsf{El}\, a)) \to \mathsf{Tm}\, (\mathsf{El}\, (\mathsf{Id}\, t\, t)) \\
\mathsf{J} & : (P : (y : \mathsf{Tm}\, (\mathsf{El}\, a)) \to \mathsf{Tm}\, (\mathsf{El}\, (\mathsf{Id}\, x\, y)) \to \mathsf{Ty}) \to \mathsf{Tm}\, (P\, x\, (\mathsf{refl}\, x)) \to \\
 & \quad (y : \mathsf{Tm}\, (\mathsf{El}\, a))(e : \mathsf{Tm}\, (\mathsf{El}\, (\mathsf{Id}\, x\, y))) \to \mathsf{Tm}\, (P\, y\, e) \\
\mathsf{J}\beta & : \mathsf{Tm}\, (\mathsf{ID}\, (\mathsf{J}\, P\, pr\, (\mathsf{refl}\, x))\, pr)
\end{array}
$$

rövidítés:
$a \Rightarrow B := \Pi\, a\, (\lambda \_ .B)$

A 6. definícióhoz képest az egyenlőség típus megadása változott: intenzionális egyenlőségünk van (nincs egyenlőség reflexió, equality reflection, hanem az egyenlőség egy induktív típus), ráadásul kétféle: ID tetszőleges típusok elemei között felírható, és szigorú β szabálya van, míg Id csak a fajták elemei között írható fel, és gyenge a β szabálya, mely az előbbi egyenlőséggel van kifejezve. Egy másik fontos különbség, hogy míg az összes korábbi szignatúra értelmes extenzionális típuselméletben, tehát, ha a metaelméletben az egyenlőség típus és a definicionális egyenlőség megegyeznek, a magasabb induktív típusok csak intenzionális típuselméletben értelmesek.

A magasabb induktív típusok a homotópia-típuselméletből származnak [Pro13], a legegyszerűbb példák a topológiából származnak, ilyen a kör és a tórusz. Alább megadjuk a szignatúráikat formálisan és informálisan. A $- \bullet -$ tranzitivitás operátort J segítségével adjuk meg, itt most nem fejtjük ki.

$$\Sigma\, \mathsf{U}\, \lambda S^1. \qquad\qquad S^1 \quad : \mathsf{Type}$$

$$
\begin{array}{ll}
\Sigma\,(\mathsf{El}\,S^1)\,\lambda base. & base : S^1 \\
\quad \mathsf{El}\,(\mathsf{Id}\,base\,base) & loop : base = base \\
\hline
\Sigma\,\mathsf{U}\,\lambda T^2. & T^2 : \mathsf{Type} \\
\Sigma\,(\mathsf{El}\,T^2)\,\lambda b. & b : T^2 \\
\Sigma\,(\mathsf{El}\,(\mathsf{Id}\,b\,b))\,\lambda p. & p : b = b \\
\Sigma\,(\mathsf{El}\,(\mathsf{Id}\,b\,b))\,\lambda q. & q : b = b \\
\quad \mathsf{El}\,(\mathsf{Id}\,(p \bullet q)\,(q \bullet p)) & t : p \bullet q = q \bullet p
\end{array}
$$

A tórusz érdekessége, hogy utolsó konstruktora egy iterált egyenlőséget ad, két különböző $b = b$ közötti egyenlőség egyenlőségét adja meg.

Kovács Andrással közösen a fenti szignatúráknak csak egy nagyon egyszerű szemantikáját adtuk meg, mely kizárólag az algebra, homomorfizmus, függő algebra és függő homomorfizmus fogalmakat tartalmazza [KK18; KK20a].

Kapcsolódó munkák. Christian Sattler megmutatta, hogyan kell a teljes kategória-szemantikát megadni, és megmutatni, hogy a függő elimináció ekvivalens az inicialitással [Sat19]. Az iniciális algebrák konstrukciója még nyitott. Magasabb induktív-induktív típusokra más leírás illetve szemantika nem létezik. Az első példákat magasabb induktív típusokra a homotópia-típuselmélet könyben találjuk [Pro13]. Magasabb induktív típusok szemantikus leírását adta meg Lumsdaine és Shulman [LS20], a homotópia-típuselmélet kubikális modellje támogatja őket [CHM18], és egy külső szintaktikus leírásukat operációs szemantikával megadta Cavallo és Harper [CH19]. Ezek a szemantikák nem kezelik az induktív-induktív aspektussal rendelkező magasabb induktív típusokat.

2.7. Másodrendű általánosított algebrai elméletek

8. Definíció (Zárt másodrendű általánosított algebrai elméletek)**.** *Tekintsük az alábbi másodrendű általánosított algebrai elméletet. Egy szignatúra az alábbi* Ty *egy eleme.*

$$
\begin{array}{lll}
\mathsf{Ty} & : \mathsf{Type} & \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Type} & \\
\Sigma & : (A : \mathsf{Ty}) \to (\mathsf{Tm}\,A \to \mathsf{Ty}) \to \mathsf{Ty} & \\
\mathsf{U} & : \mathsf{Ty} & \\
\mathsf{El} & : \mathsf{Tm}\,\mathsf{U} \to \mathsf{Ty} & \textit{rövidítés:} \\
\Pi & : (a : \mathsf{Tm}\,\mathsf{U}) \to (\mathsf{Tm}\,(\mathsf{El}\,a) \to \mathsf{Ty}) \to \mathsf{Ty} & a \Rightarrow B := \Pi\,a\,(\lambda\_.B) \\
\_ \cdot \_ & : \mathsf{Tm}\,(\Pi\,a\,B) \to (u : \mathsf{Tm}\,(\mathsf{El}\,a)) \to \mathsf{Tm}\,(B\,u) & \\
\mathsf{Eq} & : \mathsf{Tm}\,A \to \mathsf{Tm}\,A \to \mathsf{Ty} & \\
\mathsf{reflect} & : \mathsf{Tm}\,(\mathsf{Eq}\,u\,v) \to u = v & \\
\mathsf{U}^+ & : \mathsf{Ty} & \\
\mathsf{el}^+ & : \mathsf{Tm}\,\mathsf{U}^+ \to \mathsf{Tm}\,\mathsf{U} & \\
\pi^+ & : (a^+ : \mathsf{Tm}\,\mathsf{U}^+) \to \Big(\mathsf{Tm}\,\big(\mathsf{El}\,(\mathsf{el}^+\,a^+)\big) \to \mathsf{Tm}\,\mathsf{U}\Big) \to \mathsf{Tm}\,\mathsf{U} & a \Rightarrow^+ b := \pi^+\,a^+\,(\lambda\_.b) \\
\mathsf{lam}^+ & : \big(x : \mathsf{El}\,(\mathsf{el}^+\,a^+)\big) \to \mathsf{Tm}\,(\mathsf{El}\,(b\,x)) \cong \mathsf{Tm}\,\big(\mathsf{El}\,(\pi^+\,a^+\,b)\big) : \_ \cdot^+ \_ &
\end{array}
$$

Az utolsó sorban az $f : X \cong Y : g$ *jelölés izomorfizmust jelent, tehát* $f : X \to Y$ *és* $g : Y \to X$ *függvényeket, melyekre* $f \circ g = \mathsf{id}$ *és* $g \circ f = \mathsf{id}$*.*

A fenti definíció röviden: típuselmélet Σ típusokkal és extenzionális egyenlőséggel, egy Tarski-univerzum U-val, a típusok zártak U-beli értelmezési tartományú függvényekre, U^+ egy

aluniverzuma U-nak, és U zárt az U^+-beli értelmezési tartományú függvényekre. U elemei a normál fajták, U^+ elemei azon fajták, melyek szerepelhetnek másodrendű függvényekben.

A fenti definícióval megadhatók a 2, 4, 5, 6, 7, 8. definíciók elméletei (tehát ez is önleíró). A 3. definíció elmélete is megadható, ha kiegészítjük paraméter-típusokkal.

Ha a programozási nyelveket másodrendű általánosított algebrai elméletként adjuk meg, akkor a legegyszerűbb programozási nyelv Church lambda kalkulusa (ha csak elsőrendű nyelveket engedünk meg, akkor ennél egyszerűbb Schönfinkel kombinátor-kalkulusa). A lambda kalkulus a következő szignatúrával adható meg.

$$
\begin{array}{ll}
\Sigma\,\mathsf{U}^+\,\lambda Tm & Tm \;\; : \mathsf{Type} \\
\Sigma\,\big((Tm \Rightarrow^+ \mathsf{el}^+\,Tm) \Rightarrow \mathsf{El}\,(\mathsf{el}^+\,Tm)\big)\,\lambda lam. & lam \;\; : (Tm \to Tm) \to Tm \\
\Sigma\,\big(\mathsf{el}^+\,Tm \Rightarrow \mathsf{el}^+\,Tm \Rightarrow \mathsf{El}\,(\mathsf{el}^+\,Tm)\big)\,\lambda app. & app : Tm \to Tm \to Tm \\
\quad \Pi\,(Tm \Rightarrow^+ \mathsf{el}^+\,Tm)\,\lambda b.\Pi\,(\mathsf{el}^+\,Tm)\,\lambda a. & \beta \quad\; : (b : Tm \to Tm)(a : Tm) \to \\
\quad \mathsf{Eq}\,(app \cdot (lam \cdot b) \cdot a)\,(b \cdot^+ a) & \qquad app\,(lam\,b)\,a = b\,a
\end{array}
$$

A másodrendű algebrai elméletekre nincsen értelmes homomorfizmus-fogalom: az világos, mi egy másodrendű lambda-kalkulus algebra (a fenti informális jobb oldali definíció összes komponense): tekintsünk egy $f : Tm_M \to Tm_N$ függvényt, mely megőrzi az app műveletet ($(t\,a : Tm_M) \to f\,(app_M\,t\,a) = app_N\,(f\,t)\,(f\,a)$). Mit jelent az, hogy megőrzi a lam műveletet? Valami ilyesmire lenne szükség: $(b : Tm_M \to Tm_M) \to f\,(lam_M\,b) = lam_N\,(\lambda a.f\,(b\,?))$, ahol $?$ típusa Tm_M, a bemenetünk $a : Tm_N$, tehát f inverzére lenne szükség (izomorfizmus-fogalom van másodrendű algebrai elméletekre, de homomorfizmus-fogalom nincs).

Ezért a megoldás, hogy a másodrendű elméletet lefordítjuk elsőrendűre: bevezetjük a környezeteket (context) és helyettesítéseket (substitution), a környezetek összegyűjtik a kötő operátorok (amilyen a lam) kötött paramétereit. A lamdba kalkulus elsőrendű szignatúrája, amelyet egy ilyen fordítás eredményeként kapunk, a következő.

$$
\begin{array}{ll}
\mathsf{Con} & : \mathsf{Type} \\
\mathsf{Sub} & : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Type} \\
- \circ - & : \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Sub}\,\Theta\,\Delta \to \mathsf{Sub}\,\Theta\,\Gamma \\
\mathsf{ass} & : (\gamma \circ \delta) \circ \theta = \gamma \circ (\delta \circ \theta) \\
\mathsf{id} & : \mathsf{Sub}\,\Gamma\,\Gamma \\
\mathsf{idl} & : \mathsf{id} \circ \gamma = \gamma \\
\mathsf{idr} & : \gamma \circ \mathsf{id} = \gamma \\
\diamond & : \mathsf{Con} \\
\epsilon & : \mathsf{Sub}\,\Gamma\,\diamond \\
\diamond\eta & : (\sigma : \mathsf{Sub}\,\Gamma\,\diamond) \to \sigma = \epsilon \\
\mathsf{Tm} & : \mathsf{Con} \to \mathsf{Type} \\
-[-] & : \mathsf{Tm}\,\Gamma \to \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Tm}\,\Delta \\
[\circ] & : t[\gamma \circ \delta] = t[\gamma][\delta]
\end{array}
\qquad
\begin{array}{ll}
[\mathsf{id}] & : t[\mathsf{id}] = t \\
- \triangleright & : \mathsf{Con} \to \mathsf{Con} \\
-, - & : \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Tm}\,\Delta \to \mathsf{Sub}\,\Delta\,(\Gamma \triangleright) \\
\mathsf{p} & : \mathsf{Sub}\,(\Gamma \triangleright)\,\Gamma \\
\mathsf{q} & : \mathsf{Tm}\,(\Gamma \triangleright) \\
\triangleright\beta_1 & : \mathsf{p} \circ (\gamma, t) = \gamma \\
\triangleright\beta_2 & : \mathsf{q}[\gamma, t] = t \\
\triangleright\eta & : \sigma = (\mathsf{p} \circ \sigma, \mathsf{q}[\sigma]) \\
\mathsf{lam} & : \mathsf{Tm}\,(\Gamma \triangleright) \to \mathsf{Tm}\,\Gamma \\
\mathsf{lam}[] & : (\mathsf{lam}\,t)[\gamma] = \mathsf{lam}\,(t[\gamma \circ \mathsf{p}, \mathsf{q}]) \\
\mathsf{app} & : \mathsf{Tm}\,\Gamma \to \mathsf{Tm}\,\Gamma \to \mathsf{Tm}\,\Gamma \\
\mathsf{app}[] & : (\mathsf{app}\,t\,u)[\gamma] = \mathsf{app}\,(t[\gamma])\,(u[\gamma]) \\
\beta & : \mathsf{app}\,(\mathsf{lam}\,t)\,u = t[\mathsf{id}, u]
\end{array}
$$

Tehát a környezetek és a helyettesítések egy kategóriát alkotnak ($\mathsf{Con}, \ldots, \mathsf{idr}$), ennek van terminális objektuma ($\diamond$, ϵ, $\diamond\eta$), a termek ezen előképét (Type-ba képező funktort) alkotnak ($\mathsf{Tm}, \ldots, [\mathsf{id}]$) alkotnak, mely lokálisan reprezentálható ($- \triangleright, \ldots, \triangleright\eta$), és a lam művelet függvény-bemenet helyett egy eggyel nagyobb környezetben levő termet kap. Az applikáció csak a korábbi app környezetekkel indexelt változata, a β szabályban pedig jobb oldalon metaelmélet függvény-alkalmazás helyett helyettesítés szerepel (a Tm funktor morfizmusokon értelmezett művelete). Ezen kívül minden művelet természetes transzformáció ($\mathsf{lam}[]$, $\mathsf{app}[]$).

Szumi Xie-vel közösen a másodrendű → elsőrendű fordítást általánosan megadtuk [KX24a]. A fordítás működik nyílt és végtelenül elágazó elméletekre is, és kétféle változata van: a párhuzamos helyettesítési kalkulusra és az egyszeri helyettesítési kalkulusra épülő (a fenti példa fordítás-végeredmény a párhuzamos fordítás eredménye). Bebizonyítottuk a párhuzamos fordítás helyességét: ez azt jelenti, hogy a kapott szignatúra modelljei megegyeznek a naív előkéve modellekkel, amit a magasabbrendű absztrakt szintaxis eredeti szemantikájában [Hof99] kapunk.

Közvetlenül másodrendű algebrai elméletekkel dolgozni nehéz, mert még nem ismerjük, hogy mi a megfelelő metaelmélet, melyben lehet velük dolgozni (úgy tűnik, hogy valamilyen multimodális típuselméletre [Gra+21] lesz szükség). Rafaël Bocquet-val és Christian Sattler-rel közösen megmutattuk, hogy a típuselmélet kanonicitás, normalizálás, parametricitás bizonyítása közvetlenül másodrendű elmélettel dolgozva is megadható [BKS23].

Kapcsolódó munkák. A másodrendű általánosított algebrai elméleteket Uemura kezdte el használni típuselméletek leírására [Uem21]. Uemura megadott egy szintaktikus definíciót a szignatúrákra, magukat az elméleteket prezentáció-független módon, representable map category-ként adta meg. A mi leírásunk a kettő között van: absztrakt, de közel a szintaxishoz. Másodrendű egyszerű algebrai elméleteket használtak programozási nyelvek leírására, ilyen a magasabbrendű absztrakt szintaxis [Hof99; FPT99], logikai keretrendszerek [HHP93; Har21], kétszintű típuselmélet [Ann+23],

3. Kvóciens-induktív típusokat támogató metanyelvek

Az előző fejezet metanyelve az extenzionális típuselmélet [Mar84], melyben nincs különbség az egyenlőség típus és az konverzió reláció (operációs szemantika, definicionális egyenlőség) között. Ha számítógépen formalizálni szeretnénk az eredményeket, azt valamilyen intenzionális típuselméletben kell végezni, mert extenzionális típuselméletben a konverzió reláció eldönthetetlen [CCD17]. A konverzió eldöntetősége alapvető követelmény jól használható típusellenőrző implementálásához.

A típuselmélet legtöbb elérhető implementációja (Agda, Coq, Idris, Lean) intenzionális egyenlőség típusra épül, és nem támogatja a kvóciens típusokat, kizárólag posztulálni tudjuk őket. Ekkor azonban elveszik a kanonicitás [Coq19] tulajdonság: lesznek olyan zárt Bool típusú termek, melyeket futtatva (normalizálva) nem kapunk sem true-t, sem false-ot, hanem valamelyik posztulátumon elakad a kiértékelés. A kubikális Agda [VMA21] a típuselmélet végtelen groupoid modelljére [Coh+17] épül, és támogatja a kvóciens-induktív típusokat. Azonban a típuselméletnek a jóval egyszerűbb setoid modellje [Alt99] is támogatja a kvóciens-induktív típusokat, és először ezt ismertetjük. Utána kitérünk a magasabb dimenziós változatra.

3.1. Setoid típuselmélet

A setoid modellben egy zárt típust egy meta-típussal és egy azon a típuson értelmezett homogén bináris ekvivalenciarelációval adunk meg. Ezt a relációt a típus egyenlőségének tekintjük. Mivel szabad kezünk van az egyenlőség reláció megadásánál, tetszőleges kvócienst megvalósíthatunk így. Például egy adott típust úgy tudunk állítássá csonkítani, hogy relációnak a teljes relációt adjuk meg: így tetszőleges két eleme egyenlő lesz, és csak annyi információt látunk a típusból, hogy van-e eleme vagy nincs (egy állítás egy olyan típus, melynek minden eleme egyenlő). Vagy megadhatjuk a Cauchy valós számokat Bishop [Bri99] stílusában, de magasabb absztrakciós szinten, kvóciensként [Pro13] (ez egy kvóciens induktív-induktív típus). A setoid modellben a kvócienseken túl a koinduktív típusok is jól működnek, például két pontonként egyenlő végtelen folyam egyenlő, vagy két pontonként egyenlő függvény is egyenlő. Ezenkívül igaz az állításokra vonatkozó extenzionalitás szabály, tehát két logikailag ekvivalens állítás egyenlő.

A setoid modellre építve megadtunk egy setoid típuselméletet, mely a Martin-Löf típuselmélet kiegészítése újabb konverziós szabályokkal, melyek leírják, hogy hogyan kell kiszámítani egyes típusok egyenlőségét, valamint az egyenlőség eliminátorával hogyan kell számításokat végezni. Ez volt az első típuselmélet, melyben igaz a függvény-extenzionalitás és egy tényleges bővítése Martin-Löf típuselméletének, anélkül, hogy pl. elvennénk azt a szabályt, hogy az egyenlőség eliminátora a reflexivitásra alkalmazva identitás. A setoid típuselmélet szemantikáját nem egyszerűen egy modellel, hanem egy fordítással adtuk meg, a setoid típuselméletet egyszerű Martin-Löf típuselméletre fordítottuk le. Az eredményeket Thorsten Altenkirch-el, Simon Boulier-rel és Nicolas Tabareau-val közösen publikáltuk [Alt+19]. Nyitott volt még a setoid modellben a függő univerzum (típusok típusa) konstrukciója, ezt szintén fordítással meg tudtuk adni: ha a fordítás célnyelvében vannak induktív-induktív típusok, akkor a tárgynyelvben megadható egy univerzum. Ezt az eredményt Thorsten Altenkirch-el, Simon Boulier-rel, Christian Sattler-rel és Filippo Sestini-vel közösen publikáltuk [Alt+21]. A setoid típuselméletnek elkészült egy kísérleti implementációja is [Kov23]. Szumi Xie-vel közösen megmutattuk, hogy a setoid modell támogatja a 6. definícióban megadott univerzális kvóciens induktív-induktív típust, melyre az összes kvóciens induktív-induktív típus visszavezethető [KX21].

A setoid típuselmélet problémája, hogy nem teljesül benne az egyedi kiválasztási axióma. Ez az axióma szükséges például ahhoz, hogy egy függvény gráfjából megkapjunk egy konkrét, végrehajtható függvényt. A setoid típuselmélet magasabb változatában az egyedi kiválasztási axióma teljesül, ezt a következő alfejezetben ismertetjük.

Kapcsolódó munkák. A setoid típuselmélet egy korai változata az observational type theory [AMS07]. Loïc Pujet és Nicolas Tabareau továbbfejlesztették a setoid típuselméletet, egyszerűsítették a szintaxisát, bebizonyították a normalizálás tulajdonságot [PT22], modellezték az impredikatív függvényteret [PT24], és végül Pujet konstruált egy univerzumot, mely nem használ induktív-induktív típusokat (még nincs publikálva). A setoid típuselmélethez hasonló rendszer Maietti és munkatársainak a minimalista rendszere a konstruktív matematika megalapozására [Mai09].

3.2. Higher observational type theory

A Martin-Löf típuselméletben az egyenlőség egy induktív típussal van megadva, mint a legkisebb reflexív reláció. A setoid típuselméletben minden típusra külön számítjuk ki annak egyenlőség típusát, például párok egyenlősége egyenlőségek párja ($=_A$ az A típusra vonatkozó egyenlőség típus, $\equiv$ a konverzió reláció):

$$\big((a, b) =_{A\times B} (a', b')\big) \equiv \big((a =_A a') \times (b =_B b')\big).$$

A setoid típuselméletben az univerzum (Type, típusok típusa) egyenlősége a típusok struktúrája szerinti indukcióval van megadva, például

$$\big((A \times B) =_{\mathsf{Type}} (A' \times B')\big) \equiv \big((A =_{\mathsf{Type}} A') \times (B =_{\mathsf{Type}} B')\big).$$

A homotópia-típuselméletben [Pro13] Voevodsky univalence axiómája azt mondja, hogy a típusok egyenlősége ugyanaz, mint a típusok közötti bijekciók (izomorfizmusok, ekvivalenciák) halmaza. Ez a következő számítási szabálynak felelne meg:

$$(A =_{\mathsf{Type}} B) \equiv (A \cong B).$$

A higher observational type theory (HOTT) egy olyan, egyelőre képzeletbeli típuselmélet, melyben teljesül a fenti számítási szabály.

Ennek megadása egy fontos jövőbeli cél, amit eddig sikerült elérni, az a HOTT-nek egy gyenge változata, melyben a típusok egyenlősége a típusok közötti relációk típusával egyezik meg, tehát

$$(A =_{\mathsf{Type}} B) \text{ "}\equiv\text{" } (A \to B \to \mathsf{Type}).$$

A konverzió relációt azért tettük idézőjelbe, mert a jobb és bal oldal között csak szekció-retrakció kapcsolat van, és nem konverzió. Azonban már ennek a rendszernek is végtelen-dimenziós kubikális halmazok a modelljei, mert a fenti megfeleltetés iterálható: a típusok egyenlősége reláció, a relációk egyenlősége még bonyolultabb reláció, és így tovább. Az így kapott típuselmélet belső parametricitással (internal parametricity) rendelkezik, például be lehet bizonyítani benne, hogy az $(A : \mathsf{Type}) \to A \to A$ típusnak csak egyetlen eleme van, míg az $(A : \mathsf{Type}) \to A \to A$ típusnak két eleme van. Ezt a típuselméletet parametric observational type theory-nak (POTT) nevezzük. A POTT szabályait, a kubikális halmazokra épülő modelljét és a POTT-re vonatkozó kanonicitás bizonyítást Thorsten Altenkirch-el, Michael Shulman-nal és Yorgo Chamoun-nal közösen publikáltuk [Alt+24].

Kapcsolódó munkák. A homotópia-típuselmélet első modellje szimpliciális halmazokra épült [KL21], az első konstruktív modellje kubikális halmazokra [BCH19]. Egy másik kubikális halmaz modellre épül a kubikális típuselmélet nyelve [Coh+17], melyet a kubikális Agda bizonyítórendszer implementál [VMA21]. Azóta többféle kubikális modellt és típuselméletet dolgoztak ki [AFH18; Ang+21; Awo+24], ezek közül mindegyik beépíti a geometriát (egy absztrakt intervallum típust és magasabb dimenziós kocka-kitöltő operátorokat) a szintaxisba. A HOTT előnye az lenne, hogy a szabályai sokkal egyszerűbbek, több konverziós szabály teljesül, és potenciálisan hatékonyabban lehet implementálni, mint az intervallumra építő típuselméleteket.

4. Programozási nyelvek algebrai leírása

Ebben a fejezetben különböző (programozási) nyelveket írunk le az algebrai módszerrel, és ezek tulajdonságait bizonyítjuk. Sokáig nem volt világos, hogy algebrai módon le lehet-e írni a Russell-univerzumokat, be lehet-e bizonyítani a kanonicitást vagy a normalizálást, lehet-e írni típusellenőrzőt.

Doktori disszertációmban megmutattam, hogy a normalizálás bizonyítható algebrai módon is, és nincs szükség típus nélküli pretermekre vagy konverzió relációra. Ez azért jelentős, mert az algebrai szintaxison minden függvénynek meg kell őriznie a konverzió relációt: tehát például nem különböztethetjük meg semilyen formában az $1+1$ és a 2 termeket: a normalizálás során viszont sehol sincs szükség ezen kvóciens megtörésére. A kapott normálformák egyenlősége nem volt triviálisan eldönthető, ezt javítottuk ki Thorsten Altenkirch-el közösen [AK17]. A normalizálás bizonyítás jelentős részét formalizáltuk az Agda bizonyítórendszerben. Ehhez kapcsolódóan megmutattuk, hogy típusellenőrzés során kizárólag jól típusozott (algebrai) termek egyenlőségét kell eldönteni [Kap18]. Megmutattuk, hogy az algebrai normalizálás-bizonyítás működik nagyméretű eliminációval rendelkező Bool-okkal kiegészített típuselméletre is [AKK17].

A normalizálás egy speciális esete a kanonicitás, megmutattuk, hogy a kanonicitás pedig a kategorikus ragasztásnak egy speciális esete, mely tetszőleges típuselmélet-modellre elvégezhető. Típuselmélet modellek kategorikus ragasztásából ezen kívül megkapjuk a szintaktikus és szemantikus parametricitást is. A kategorikus ragasztásra vonatkozó eredményeket Christian Sattler-rel és Simon Huber-rel közösen publikáltuk [KHS19].

Régóta ismert, hogy Schönfinkel kombinátor kalkulusának és Church lambda kalkulusának szintaxisa valamilyen értelemben ekvivalens [HS08]. Megmutattuk, hogy ez az ekvivalencia algebrai módon is bebizonyítható: a lambda kalkulust is, mint algebrai elméletet adjuk meg (típus nélküli illetve egyszerűen típusos családos kategóriákkal), és az ekvivalencia bizonyítása során nem kell megtörni a kvócienst. A típusozás a bizonyításunk paramétere, ezért egyszerre láttuk be ugyanazt a típus nélküli és az egyszerű típusokkal rendelkező lambda kalkulusra illetve kombinátor kalkulusra. Ezeket az eredményeket Thorsten Altenkirch-el, Artjoms Sinkarovs-val és Végh Tamással közösen publikáltuk [Alt+23], a cikk összes állítását kubikális Agdában is formalizáltuk.

A Russell-univerzumok azt jelentik, hogy valójában nem különítjük el a típusokat és a termeket, hanem a típusok csak olyan termek, amelyek típusa valamilyen univerzum. Közvetlenül így megadva a termeket azonban nagyon függő (very dependent) típusokat kapunk, melyek algebrailag értelmezhetetlenek: a term megjelenik saját típusában. Thorsten Altenkirch-el, Artjoms Sinkarovs-val és Végh Tamással közösen megadtunk egy módszert, mellyel az ilyen nagyon függő típusok is értelmezhetők algebrai módon, ezt Münchhausen módszernek neveztük [Alt+22]. A cikkben a Münchhausen módszert alkalmaztuk a Russell-univerzumok, a végtelen sorozatok, többdimenziós tömbök, a környezetek nélküli típuselmélet és a függő kombinátor kalkulus megadására.

A típuselmélet legnépszerűbb algebrai leírása családos kategóriákra [CCD21], más néven párhuzamos helyettesítésekre épül. Luksa Norberttel közösen megmutattuk, hogy az egyszerű típuselmélet [KL20] megadható egyes helyettesítési kalkulussal. Szumi Xie-vel közösen megmutattuk, hogy ugyanez megtehető a függő típuselméletre is [KX24b]. Az egyes helyettesítési kalkulus közelebb van a szokásos informális leíráshoz és kevesebb műveletet illetve egyenlőséget tartalmaz, mint a párhuzamos helyettesítési kalkulus. A jövőben lehet, hogy alkalmazható a típuselmélet szintaxisának halmaz-csonkítás nélküli megadására. Jelenleg nem ismert, hogy homotópia-típuselméletben megadható-e a típuselmélet végtelen-dimenziós szintaxisa, ennek a kérdésnek az eldöntésében segíthet az egyes helyettesítési kalkulus.

Az algebrai típuselmélet-leírás nagyon jól működik papíron, de a számítógépes formalizálás nehéz. Ennek az az oka, hogy bizonyos műveletek és egyenlőségek csak korábbi egyenlőségek miatt típushelyesek, így a korábbi egyenlőséggel transzportálni kell bizonyos termeket (lásd 2.5. alfejezet). A transzport az egyenlőség típus eliminátora, ha reflexivitásra alkalmazzuk, akkor az identitás függvény. Az ebből keletkező helyzetet transzport pokolként szokás jellemezni. A transzport pokol kezelésére kétféle módszer ismert: (i) elkerüljük az indexelt típusok alkalmazását, más szóval esszenciális algebrai elméletekkel dolgozunk általánosított algebrai elméletek helyett; (ii) az egyenlőségeket szigorúvá (definicionálissá, konverzióvá) alakítjuk, így a transzportok eltűnnek (mert a definicionális egyenlőség reflexivitással bizonyítható). Az (i) módszer problémája, hogy sokkal kevésbé olvasható, és távolabb van a típuselmélet gyakorlati implementációitól. A (ii) lehetőséget használjuk ki, amikor a típuselméletet sekélyen beágyazzuk a saját implementációjába (ezt nevezik standard modellnek, metacirkuláris modellnek, halmaz/típus modellnek is). Ilyenkor a metanyelv eldönti helyettünk az egyenlőségeket, így az algebrai elmélet összes egyenlősége definíció szerint teljesülni fog, és az összes transzport eltűnik. Így a típuselmélet sekély beágyazása fölött nagyon könnyű függő modelleket megadni, és így a típuselmélet szintaxisa szerinti indukciós bizonyítások helyességét számítógéppel ellenőrizni. Megmutattuk, hogy feltételezve a metanyelvi implementáció helyességét, ez a módszer pontosan akkor működik, amikor mély beágyazásra is működik a bizonyítás. Az ilyen módszerrel csak ellenőrizni lehet a bizonyításokat, de nem tudjuk futtatni a bizonyításokat, hiszen a sekély beágyazásra nincs indukciós elvünk. Az eredményeket Kovács Andrással és Nicolai Kraus-szal közösen publikáltuk [KKK19].

Míg extenzionális típuselméletben vagy setoid típuselméletben egy (elsőrendű) algebrai elméletnek egyféle modell-fogalma van, intenzionális típuselméletben (ahol nem teljesül a függvény-extenzionalitás), kétféle módon adhatók meg az egyenlőségek: pontonként illetve pontmentes módon. Például a félcsoportok asszociativitás tulajdonsága a következő pontonként (bal oldal) illetve pontfüggetlen módon (jobb oldal), C a félcsoport alaphalmaza, $\otimes$ a művelet:

$$(x\ y\ z : C) \to (x \otimes y) \otimes z =_C x \otimes (y \otimes z) \qquad (\lambda x\ y\ z.(x \otimes y) \otimes z) =_{C\to C\to C\to C} (\lambda x\ y\ z.x \otimes (y \otimes z))$$

Ha van függvény-extenzionalitásunk, akkor a bal oldaliból következik a jobb oldali egyenlőség, ha nincs, akkor a jobb oldali erősebb. Hasonlóképpen az állítások algebrai struktúrája is megadható pontfüggetlen módon: egy állítás egy P halmaz, melynek bármely két eleme egyenlő, a

bal oldali ennek a pontonkénti, a jobb oldali a pontfüggetlen megfogalmazása:

$$(x\ y : P) \to x =_P y \qquad\qquad (\lambda x\ y.x) =_{P\to P\to P} (\lambda x\ y.y)$$

Ha a metaelméletünkben nincs függvény-extenzionalitás, akkor a bal oldai állítás-fogalom nem zárt a függvény típusokra, míg a jobb oldali igen. Ezt a tulajdonságot kihasználva megadtuk a setoid modellnek egy változatát, mely intenzionális metaelméletben, az állítások szigorú univerzuma nélkül is működik. Sajnos ez a modell nem támogat induktív típusokat, emiatt korlátozott a használata, de illusztrálja, hogy pontfüggetlen egyenlőségekkel teljesen intenzionális környezetben is lehet dolgozni. Az eredményeket Donkó Istvánnal közösen publikáltuk [DK21].

Az eddig felsorolt eredmények mind elsőrendű algebrai elméletekre vonatkoztak. A 2.7. fejezetben leírt másodrendű algebrai leírás magasabb szintű, kevesebb adminisztrációt (boilerplate-et) tartalmaz, mint az elsőrendű. Christian Sattler-ral és Rafaël Bocquet-val közösen megmutattuk, hogy a típuselmélet kanonicitás, normalizálás és parametricitás bizonyításai megadhatók közvetlenül másodrendű függő modellekkel [BKS23]. Az alapötlet az, hogy az elsőrendű reprezentációra [KX24a] szintén szükség van, a másodrendű függő modell egy elsőrendű modell fölött adható meg, az elsőrendű modell viszont mindig generálható másodrendű modellből úgynevezett kontextualizációval (ami szintén a standard/metacirkuláris/halmaz/típus modell egy változata). Az ehhez szükséges metanyelv egy előkéve modell belső nyelve, és még nem világos, hogy tetszőleges másodrendű algebrai elmélethez mi a legmegfelelőbb előkéve modell. Egy adott nyelv megadása és a nyelvben való programozás azonban mindenképpen kényelmesebb másodrendű módon [Kap24].

Kapcsolódó munkák. Továbbra sincs formalizálva algebrai normalizálás-bizonyítás. A legteljesebb típuselméletekre vonatkozó normalizálás-bizonyítások mind alacsony szinten dogloznak [AÖV18; Adj+24]. Coquand mutatta meg, hogy a normalizálás egy speciális esete a kanonicitás, mely jóval egyszerűbben, előkéve struktúrák nélkül is bizonyítható [Coq19]. Az egyszeres helyettesítési kalkulus egy kategóriaelméleti modellje a B-rendszer [Ahr+23]. A mi egyszeres helyettesítési kalkulusunk kevesebb egyenlőséget tartalmaz, ezért több modellje van, mint a B-rendszereknek, a szintaxisok viszont ekvivalensek. A transzport pokol elkerülésének esszenciális algebrai módját választotta Brunerie és de Boer [BB20], amikor felépítették a típuselmélet iniciális modelljét kvóciens típusok segítségével. A sekély beágyazás módszerét mások is alkalmazták a típuselmélet szintaxisán megadott indukció formalizálására [BPT17], de a módszer helyességét nem bizonyították. Pontfüggetlen egyenlőségeket Hugunin is használt induktív típusok W-típusokra való visszavezetésére [Hug20], és mi is használtuk a setoid modell szigorú transzport szabályának formalizálására [Alt+19]. A közvetlen másodrendű modellekkel való érvelés alternatív, kategorikus módszerét Jon Sterling fejlesztette ki [Ste22], ez esszenciális algebrai leírásra épül, míg a mi módszerünk általánosított algebrai leírásra épül.

5. Tézisek

Az egyes téziseknél listázzuk a vonatkozó saját publikációkat.

1. A szignatúrák elmélete módszer kidolgozása, és alkalmazása a következő induktív típusosztályokra.

   a) kölcsönösen megadott induktív típusok [KR20],

   b) induktív-induktív típusok [KKL20],

   c) kvóciens induktív-induktív típusok [KKA19; KK20b],

   d) magasabb induktív-induktív típusok [KK18; KK20a],

   e) másodrendű általánosított algebrai elméletek [KX24a].

   Az a)–c) osztályok minden tagjára a kategorikus szemantika megadása és az iniciális algebrák létezésének bizonyítása. A d) osztály tagjaira a (függő) algebra és (függő) homomorfizmus-fogalom megadása. Az e) osztály tagjainak elsőrendű algebrai elméletre (kvóciens induktív-induktív típusra) fordítása és a fordítás helyességének bizonyítása.

2. A setoid típuselmélet kidolgozása a következő szolgáltatásokkal:

   a) függvény-extenzionalitás, propozicionális extenzionalitás, kvóciens típusok, szigorú számítási szabály az egyenlőség eliminátorára [Alt+19],

   b) univerzum, amely zárt az összes típusra [Alt+21],

   c) univerzális kvóciens induktív-induktív típus, ezen keresztül az összes kvóciens induktív-induktív típus [KX21]

3. A parametric observational type theory kidolgozása, szemantikával és kanonicitás bizonyítással [Alt+24].

4. Programozási nyelvek algebrai leírásának alkalmazásai.

   a) Függő típuselméletre normalizálás bizonyítás és a konverzió reláció eldönthetősége [AK17], a típuselmélet bővítése nagyméretű eliminációval rendelkező Bool típussal [AKK17].

   b) Kategorikus ragasztás típuselméletre [KHS19].

   c) Kombinátor kalkulus és lambda kalkulus ekvivalenciája [Alt+23].

   d) Münchhausen módszer a nagyon függő típusok algebrai megadására [Alt+22].

   e) Egyes helyettesítési kalkulus egyszerű [KL20] és függő [KX24b] típuselméletre.

   f) Sekély beágyazás típuselméletre vonatkozó metaelméleti bizonyítások számítógépes ellenőrzésére [KKK19].

   g) Pontfüggetlen egyenlőségek a setoid modell állítás-univerzum nélküli megadására [DK21].

   h) Metaelméleti bizonyítások függő típuselméletre másodrendű függő modellként megadva [BKS23].

A doktori fokozat óta megjelent saját publikációk

[AK17] Thorsten Altenkirch és Ambrus Kaposi. "Normalisation by Evaluation for Type Theory, in Type Theory". *Logical Methods in Computer Science* Volume 13, Issue 4 (2017. okt.). **10 független, 1 függő hivatkozás. MTMT link**. DOI: 10.23638/LMCS-13(4:1)2017.

[AKK17] Thorsten Altenkirch, Ambrus Kaposi és András Kovács. "Normalisation by Evaluation for a Type Theory with Large Elimination". *23rd International Conference on Types for Proofs and Programs, TYPES 2017*. Szerk. Ambrus Kaposi. **MTMT link**. Eötvös Loránd University, 2017. URL: http://types2017.elte.hu/proc.pdf#page=26.

[Kap17] Ambrus Kaposi. "A fröccs szintaxisa és operációs szemantikája". https://akaposi.github.io/froccs.pdf. 2017.

[KK18] Ambrus Kaposi és András Kovács. "A Syntax for Higher Inductive-Inductive Types". *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*. Szerk. Hélène Kirchner. 108. köt. Leibniz International Proceedings in Informatics (LIPIcs). **11 független, 4 függő hivatkozás. MTMT link**. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 20:1–20:18. ISBN: 978-3-95977-077-4. DOI: 10.4230/LIPIcs.FSCD.2018.20.

[Alt+19] Thorsten Altenkirch, Simon Boulier, Ambrus Kaposi és Nicolas Tabareau. "Setoid Type Theory—A Syntactic Translation". *Mathematics of Program Construction*. Szerk. Graham Hutton. **4 független, 4 függő hivatkozás. MTMT link**. Cham: Springer International Publishing, 2019, 155–196. old. ISBN: 978-3-030-33636-3.

[KHS19] Ambrus Kaposi, Simon Huber és Christian Sattler. "Gluing for Type Theory". *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*. Szerk. Herman Geuvers. 131. köt. Leibniz International Proceedings in Informatics (LIPIcs). **7 független hivatkozás. MTMT link**. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 25:1–25:19. ISBN: 978-3-95977-107-8. DOI: 10.4230/LIPIcs.FSCD.2019.25.

[KKA19] Ambrus Kaposi, András Kovács és Thorsten Altenkirch. "Constructing Quotient Inductive-inductive Types". *Proc. ACM Program. Lang.* 3.POPL (2019. jan.). **21 független, 6 függő hivatkozás. MTMT link**, 2:1–2:24. ISSN: 2475-1421. DOI: 10.1145/3290315.

[KKK19] Ambrus Kaposi, András Kovács és Nicolai Kraus. "Shallow Embedding of Type Theory is Morally Correct". *Mathematics of Program Construction*. Szerk. Graham Hutton. **2 független hivatkozás. MTMT link**. Cham: Springer International Publishing, 2019, 329–365. old. ISBN: 978-3-030-33636-3.

[KK20a] Ambrus Kaposi és András Kovács. "Signatures and Induction Principles for Higher Inductive-Inductive Types". *Logical Methods in Computer Science* Volume 16, Issue 1 (2020. febr.). **7 független hivatkozás. MTMT link**. URL: https://lmcs.episciences.org/6100.

[KKL20] Ambrus Kaposi, András Kovács és Ambroise Lafont. "For Finitary Induction-Induction, Induction Is Enough". *25th International Conference on Types for Proofs and Programs (TYPES 2019)*. Szerk. Marc Bezem és Assia Mahboubi. 175. köt. Leibniz International Proceedings in Informatics (LIPIcs). **MTMT link**. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 6:1–6:30. ISBN: 978-3-95977-158-0. DOI: 10.4230/LIPIcs.TYPES.2019.6.

[KL20] Ambrus Kaposi és Norbert Luksa. "A calculus of single substitutions for simple type theory". https://bitbucket.org/akaposi/tel_stt/raw/master/paper.pdf. **MTMT link**. 2020.

[KR20] Ambrus Kaposi és Jakob von Raumer. "A Syntax for Mutual Inductive Families". *5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference)*. Szerk. Zena M. Ariola. 167. köt. LIPIcs. **MTMT link**. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 23:1–23:21. ISBN: 978-3-95977-155-9. DOI: 10.4230/LIPIcs.FSCD.2020.23.

[KK20b] András Kovács és Ambrus Kaposi. "Large and Infinitary Quotient Inductive-Inductive Types". *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*. Szerk. Holger Hermanns, Lijun Zhang, Naoki Kobayashi és Dale Miller. **2 független, 1 függő hivatkozás. MTMT link**. ACM, 2020, 648–661. old. ISBN: 978-1-4503-7104-9. DOI: 10.1145/3373718.3394770.

[Alt+21] Thorsten Altenkirch, Simon Boulier, Ambrus Kaposi, Christian Sattler és Filippo Sestini. "Constructing a universe for the setoid model". *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*. Szerk. Stefan Kiefer és Christine Tasson. 12650. köt. Lecture Notes in Computer Science. **1 független hivatkozás. MTMT link**. Springer, 2021, 1–21. old. DOI: 10.1007/978-3-030-71995-1_1.

[DK21] István Donkó és Ambrus Kaposi. "Internal Strict Propositions Using Point-Free Equations". *27th International Conference on Types for Proofs and Programs, TYPES 2021, June 14-18, 2021, Leiden, The Netherlands (Virtual Conference)*. Szerk. Henning Basold, Jesper Cockx és Silvia Ghilezan. 239. köt. LIPIcs. **MTMT link**. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 6:1–6:21. DOI: 10.4230/LIPIcs.TYPES.2021.6.

[KX21] Ambrus Kaposi és Zongpu Xie. "Quotient inductive-inductive types in the setoid model". *27th International Conference on Types for Proofs and Programs, TYPES 2021*. Szerk. Henning Basold. **MTMT link**. Universiteit Leiden, 2021. URL: https://types21.liacs.nl/download/quotient-inductive-inductive-types-in-the-setoid-model/.

[Alt+22] Thorsten Altenkirch, Ambrus Kaposi, Artjoms Sinkarovs és Tamás Végh. "The Münchhausen Method in Type Theory". *28th International Conference on Types for Proofs and Programs, TYPES 2022, June 20-25, 2022, LS2N, University of Nantes, France*. Szerk. Delia Kesner és Pierre-Marie Pédrot. 269. köt. LIPIcs. **MTMT link**. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 10:1–10:20. DOI: 10.4230/LIPICS.TYPES.2022.10.

[Alt+23] Thorsten Altenkirch, Ambrus Kaposi, Artjoms Sinkarovs és Tamás Végh. "Combinatory Logic and Lambda Calculus Are Equal, Algebraically". *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, July 3-6, 2023, Rome, Italy*. Szerk. Marco Gaboardi és Femke van Raamsdonk. 260. köt. LIPIcs. **MTMT link**. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 24:1–24:19. DOI: 10.4230/LIPICS.FSCD.2023.24.

[BKS23] Rafaël Bocquet, Ambrus Kaposi és Christian Sattler. “For the Metatheory of Type Theory, Internal Sconing Is Enough”. *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, July 3-6, 2023, Rome, Italy*. Szerk. Marco Gaboardi és Femke van Raamsdonk. 260. köt. LIPIcs. **3 független hivatkozás. MTMT link**. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 18:1–18:23. DOI: 10.4230/LIPICS.FSCD.2023.18.

[MK23] Zoltán Gábor Molnár és Ambrus Kaposi. “A filozófia csodálatos hatékonysága a formális tudományokban”. *ÉRINTŐ : ELEKTRONIKUS MATEMATIKAI LAPOK* 29. szám 2023. szeptember (2023). **MTMT link**, 1–1. old. URL: https://m2.mtmt.hu/api/publication/34399561.

[Alt+24] Thorsten Altenkirch, Yorgo Chamoun, Ambrus Kaposi és Michael Shulman. “Internal Parametricity, without an Interval”. *Proc. ACM Program. Lang.* 8.POPL (2024). **1 független hivatkozás. MTMT link**, 2340–2369. old. DOI: 10.1145/3632920.

[Kap24] Ambrus Kaposi. “Formális nyelv = másodrendű algebrai elmélet”. *Húszéves az ELTE Eötvös József Collegium Informatikai Műhelye*. **MTMT link**. 2024, 183–218. old. URL: https://eotvos.elte.hu/dstore/document/10844/im20-v20240712.pdf#page=183.

[KX24a] Ambrus Kaposi és Szumi Xie. “Second-Order Generalised Algebraic Theories: Signatures and First-Order Semantics”. *9th International Conference on Formal Structures for Computation and Deduction, FSCD 2024, July 10-13, 2024, Tallinn, Estonia*. Szerk. Jakob Rehof. 299. köt. LIPIcs. **MTMT link**. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 10:1–10:24. DOI: 10.4230/LIPICS.FSCD.2024.10.

[KX24b] Ambrus Kaposi és Szumi Xie. “Type theory in type theory using single substitutions”. *30th International Conference on Types for Proofs and Programs (TYPES 2024)*. Szerk. Patrick Bahr és Rasmus Ejlers Møgelberg. **MTMT link**. Copenhagen, 2024. URL: https://types2024.itu.dk/abstracts.pdf#page=80.

Hivatkozások

[Ben65] Paul Benacerraf. "What Numbers Could not Be". *The Philosophical Review* 74.1 (1965), 47–73. old. ISSN: 00318108, 15581470. URL: http://www.jstor.org/stable/2183530 (elérés dátuma 2023. 11. 24.).

[Fre72] Peter Freyd. "Aspects of topoi". *Bulletin of the Australian Mathematical Society* 7.1 (1972), 1–76. old. DOI: 10.1017/S0004972700044828.

[Mar84] Per Martin-Löf. *Intuitionistic type theory*. 1. köt. Studies in proof theory. Bibliopolis, 1984. ISBN: 978-88-7088-228-5.

[Car86] John Cartmell. "Generalised algebraic theories and contextual categories". *Ann. Pure Appl. Log.* 32 (1986), 209–243. old. DOI: 10.1016/0168-0072(86)90053-9. URL: https://doi.org/10.1016/0168-0072(86)90053-9.

[Hug89] John Hughes. "Why Functional Programming Matters". *Comput. J.* 32.2 (1989), 98–107. old. DOI: 10.1093/COMJNL/32.2.98.

[PS89] Kent Petersson és Dan Synek. "A Set Constructor for Inductive Sets in Martin-Löf's Type Theory". *Category Theory and Computer Science, Manchester, UK, September 5-8, 1989, Proceedings*. Szerk. David H. Pitt, David E. Rydeheard, Peter Dybjer, Andrew M. Pitts és Axel Poigné. 389. köt. Lecture Notes in Computer Science. Springer, 1989, 128–140. old. DOI: 10.1007/BFB0018349. URL: https://doi.org/10.1007/BFb0018349.

[MR91] QingMing Ma és John C. Reynolds. "Types, Abstractions, and Parametric Polymorphism, Part 2". *Mathematical Foundations of Programming Semantics, 7th International Conference, Pittsburgh, PA, USA, March 25-28, 1991, Proceedings*. Szerk. Stephen D. Brookes, Michael G. Main, Austin Melton, Michael W. Mislove és David A. Schmidt. 598. köt. Lecture Notes in Computer Science. Springer, 1991, 1–40. old. DOI: 10.1007/3-540-55511-0_1.

[Str91] Thomas Streicher. *Semantics of Type Theory: Correctness, Completeness, and Independence Results*. Cambridge, MA, USA: Birkhauser Boston Inc., 1991. ISBN: 0-8176-3594-7.

[HHP93] Robert Harper, Furio Honsell és Gordon D. Plotkin. "A Framework for Defining Logics". *J. ACM* 40.1 (1993), 143–184. old. DOI: 10.1145/138027.138060. URL: https://doi.org/10.1145/138027.138060.

[Pau93] Christine Paulin-Mohring. "Inductive Definitions in the system Coq - Rules and Properties". *Typed Lambda Calculi and Applications, International Conference on Typed Lambda Calculi and Applications, TLCA '93, Utrecht, The Netherlands, March 16-18, 1993, Proceedings*. Szerk. Marc Bezem és Jan Friso Groote. 664. köt. Lecture Notes in Computer Science. Springer, 1993, 328–345. old. DOI: 10.1007/BFB0037116. URL: https://doi.org/10.1007/BFb0037116.

[Dyb94] Peter Dybjer. "Inductive Families". *Formal Aspects Comput.* 6.4 (1994), 440–465. old. DOI: 10.1007/BF01211308. URL: https://doi.org/10.1007/BF01211308.

[Mar98] Per Martin-Löf. "An intuitionistic theory of types". *Twenty-five years of constructive type theory (Venice, 1995)*. 36. köt. Oxford Logic Guides. Oxford Univ. Press, New York, 1998, 127–172. old.

[OS98] Jaap van Oosten és Alex K. Simpson. “Some axiomatic results in synthetic domain theory”. *Workshop on Domains IV 1998, Haus Humboldtstein, Remagen-Rolandseck, Germany, October 2-4, 1998*. Szerk. Dieter Spreen, Ralf Greb, Holger Schulz és Michel P. Schellekens. 35. köt. Electronic Notes in Theoretical Computer Science. Elsevier, 1998, 175. old. DOI: 10.1016/S1571-0661(05)80740-6.

[Alt99] Thorsten Altenkirch. “Extensional Equality in Intensional Type Theory”. *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. IEEE Computer Society, 1999, 412–420. old. DOI: 10.1109/LICS.1999.782636. URL: https://doi.org/10.1109/LICS.1999.782636.

[AR99] Thorsten Altenkirch és Bernhard Reus. “Monadic Presentations of Lambda Terms Using Generalized Inductive Types”. *Computer Science Logic, 13th International Workshop, CSL '99, 8th Annual Conference of the EACSL, Madrid, Spain, September 20-25, 1999, Proceedings*. Szerk. Jörg Flum és Mario Rodríguez-Artalejo. 1683. köt. Lecture Notes in Computer Science. Springer, 1999, 453–468. old. DOI: 10.1007/3-540-48168-0_32.

[Bri99] Douglas S. Bridges. “Constructive mathematics: a foundation for computable analysis”. *Theoretical Computer Science* 219.1 (1999), 95–109. old. ISSN: 0304-3975. DOI: https://doi.org/10.1016/S0304-3975(98)00285-0. URL: https://www.sciencedirect.com/science/article/pii/S0304397598002850.

[FPT99] Marcelo P. Fiore, Gordon D. Plotkin és Daniele Turi. “Abstract Syntax and Variable Binding”. *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. IEEE Computer Society, 1999, 193–202. old. DOI: 10.1109/LICS.1999.782615. URL: https://doi.org/10.1109/LICS.1999.782615.

[Hof99] Martin Hofmann. “Semantical Analysis of Higher-Order Abstract Syntax”. *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. IEEE Computer Society, 1999, 204–213. old. DOI: 10.1109/LICS.1999.782616. URL: https://doi.org/10.1109/LICS.1999.782616.

[BDJ03] Marcin Benke, Peter Dybjer és Patrik Jansson. “Universes for Generic Programs and Proofs in Dependent Type Theory”. *Nord. J. Comput.* 10.4 (2003), 265–289. old.

[VV03] Katalin Pásztorné Varga és Magda Várterész. *A matematikai logika alkalmazás-szemléletű tárgyalása*. Panem, 2003. ISBN: 9635453647. URL: https://dtk.tankonyvtar.hu/xmlui/handle/123456789/8365.

[AAG05] Michael Abbott, Thorsten Altenkirch és Neil Ghani. “Containers — Constructing Strictly Positive Types”. *Theoretical Computer Science* 342 (2005. szept.). Applied Semantics: Selected Topics, 3–27. old.

[Bau06] Andrej Bauer. “First Steps in Synthetic Computability Theory”. *Electron. Notes Theor. Comput. Sci.* 155 (2006. máj.), 5–31. old. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2005.11.049.

[AMS07] Thorsten Altenkirch, Conor McBride és Wouter Swierstra. “Observational equality, now!”: *Proceedings of the ACM Workshop Programming Languages meets Program Verification, PLPV 2007, Freiburg, Germany, October 5, 2007*. Szerk. Aaron Stump és Hongwei Xi. ACM, 2007, 57–68. old. DOI: 10.1145/1292597.1292608. URL: https://doi.org/10.1145/1292597.1292608.

[MNS07] Judit X. Madarász, István Németi és Gergely Székely. “First-Order Logic Foundation of Relativity Theories”. *Mathematical Problems from Applied Logic II: Logics for the XXIst Century*. Szerk. Dov M. Gabbay, Michael Zakharyaschev és Sergei S. Goncharov. New York, NY: Springer New York, 2007, 217–252. old. ISBN: 978-0-387-69245-6. URL: https://doi.org/10.1007/978-0-387-69245-6_4.

[Cha08] James Chapman. “Type Theory Should Eat Itself”. *Proceedings of the International Workshop on Logical Frameworks and Metalanguages: Theory and Practice, LFMTP@LICS 2008, Pittsburgh, PA, USA, June 23, 2008*. Szerk. Andreas Abel és Christian Urban. 228. köt. Electronic Notes in Theoretical Computer Science. Elsevier, 2008, 21–36. old. DOI: 10.1016/J.ENTCS.2008.12.114. URL: https://doi.org/10.1016/j.entcs.2008.12.114.

[HS08] J. Roger Hindley és Jonathan P. Seldin. *Lambda-Calculus and Combinators: An Introduction*. 2. kiad. USA: Cambridge University Press, 2008. ISBN: 0521898854.

[Mai09] Maria Emilia Maietti. “A minimalist two-level foundation for constructive mathematics”. *Ann. Pure Appl. Log.* 160.3 (2009), 319–354. old. DOI: 10.1016/J.APAL.2009.01.006. URL: https://doi.org/10.1016/j.apal.2009.01.006.

[Awo10] S. Awodey. *Category Theory*. Oxford Logic Guides. OUP Oxford, 2010. ISBN: 9780199587360. URL: http://books.google.co.uk/books?id=-MCJ6x2lC7oC.

[Cha+10] James Chapman, Pierre-Évariste Dagand, Conor McBride és Peter Morris. “The gentle art of levitation”. *Proceeding of the 15th ACM SIGPLAN international conference on Functional programming, ICFP 2010, Baltimore, Maryland, USA, September 27-29, 2010*. Szerk. Paul Hudak és Stephanie Weirich. ACM, 2010, 3–14. old. DOI: 10.1145/1863543.1863547. URL: https://doi.org/10.1145/1863543.1863547.

[FS10] Fredrik Nordvall Forsberg és Anton Setzer. “Inductive-Inductive Definitions”. *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*. Szerk. Anuj Dawar és Helmut Veith. 6247. köt. Lecture Notes in Computer Science. Springer, 2010, 454–468. old. DOI: 10.1007/978-3-642-15205-4_35. URL: https://doi.org/10.1007/978-3-642-15205-4%5C_35.

[CO12] Jacques Carette és Russell O’Connor. “Theory Presentation Combinators”. *Intelligent Computer Mathematics - 11th International Conference, AISC 2012, 19th Symposium, Calculemus 2012, 5th International Workshop, DML 2012, 11th International Conference, MKM 2012, Systems and Projects, Held as Part of CICM 2012, Bremen, Germany, July 8-13, 2012. Proceedings*. Szerk. Johan Jeuring, John A. Campbell, Jacques Carette, Gabriel Dos Reis, Petr Sojka, Makarius Wenzel és Volker Sorge. 7362. köt. Lecture Notes in Computer Science. Springer, 2012, 202–215. old. DOI: 10.1007/978-3-642-31374-5_14. URL: https://doi.org/10.1007/978-3-642-31374-5%5C_14.

[SS12] Urs Schreiber és Michael Shulman. “Quantum Gauge Field Theory in Cohesive Homotopy Type Theory”. *Proceedings 9th Workshop on Quantum Physics and Logic, QPL 2012, Brussels, Belgium, 10-12 October 2012*. Szerk. Ross Duncan és Prakash Panangaden. 158. köt. EPTCS. 2012, 109–126. old. DOI: 10.4204/EPTCS.158.8.

[Gon13] Georges Gonthier. “Engineering mathematics: the odd order theorem proof”. *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL ’13, Rome, Italy - January 23 - 25, 2013*. Szerk. Roberto Giacobazzi és Radhia Cousot. ACM, 2013, 1–2. old. DOI: 10.1145/2429069.2429071.

[Pro13] The Univalent Foundations Program. *Homotopy type theory: Univalent foundations of mathematics*. Techn. jel. Institute for Advanced Study, 2013.

[ALR14] Thorsten Altenkirch, Nuo Li és Ondrej Rypacek. "Some constructions on ω-groupoids". *Proceedings of the 2014 International Workshop on Logical Frameworks and Meta-languages: Theory and Practice, LFMTP '14, Vienna, Austria, July 17, 2014*. Szerk. Amy P. Felty és Brigitte Pientka. ACM, 2014, 4: 1–4: 8. DOI: 10.1145/2631172.2631176.

[Alt+15] Thorsten Altenkirch, Neil Ghani, Peter G. Hancock, Conor McBride és Peter Morris. "Indexed containers". *J. Funct. Program.* 25 (2015). DOI: 10.1017/S095679681500009X. URL: https://doi.org/10.1017/S095679681500009X.

[AK16] Thorsten Altenkirch és Ambrus Kaposi. "Type Theory in Type Theory Using Quotient Inductive Types". *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '16. St. Petersburg, FL, USA: ACM, 2016, 18–29. old. ISBN: 978-1-4503-3549-2. DOI: 10.1145/2837614.2837638.

[Ang+16] Carlo Angiuli, Edward Morehouse, Daniel R. Licata és Robert Harper. "Homotopical patch theory". *J. Funct. Program.* 26 (2016), e18. DOI: 10.1017/S0956796816000198.

[Har16] Robert Harper. *Practical Foundations for Programming Languages*. 2nd. New York, NY, USA: Cambridge University Press, 2016.

[ADK17] Thorsten Altenkirch, Nils Anders Danielsson és Nicolai Kraus. "Partiality, Revisited - The Partiality Monad as a Quotient Inductive-Inductive Type". *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*. Szerk. Javier Esparza és Andrzej S. Murawski. 10203. köt. Lecture Notes in Computer Science. 2017, 534–549. old. DOI: 10.1007/978-3-662-54458-7_31. URL: https://doi.org/10.1007/978-3-662-54458-7%5C_31.

[BPT17] Simon Boulier, Pierre-Marie Pédrot és Nicolas Tabareau. "The Next 700 Syntactical Models of Type Theory". *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*. CPP 2017. Paris, France: ACM, 2017, 182–194. old. ISBN: 978-1-4503-4705-1. DOI: 10.1145/3018610.3018620.

[CCD17] Simon Castellan, Pierre Clairambault és Peter Dybjer. "Undecidability of Equality in the Free Locally Cartesian Closed Category (Extended version)". *Log. Methods Comput. Sci.* 13.4 (2017). DOI: 10.23638/LMCS-13(4:22)2017. URL: https://doi.org/10.23638/LMCS-13(4:22)2017.

[Coh+17] Cyril Cohen, Thierry Coquand, Simon Huber és Anders Mörtberg. "Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom". *FLAP* 4.10 (2017), 3127–3170. old. URL: http://collegepublications.co.uk/ifcolog/?00019.

[AÖV18] Andreas Abel, Joakim Öhman és Andrea Vezzosi. "Decidability of conversion for type theory in type theory". *Proc. ACM Program. Lang.* 2.POPL (2018), 23:1–23:29. DOI: 10.1145/3158111. URL: https://doi.org/10.1145/3158111.

[Alt+18] Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus és Fredrik Nordvall Forsberg. "Quotient Inductive-Inductive Types". *Foundations of Software Science and Computation Structures - 21st International Conference, FOSSACS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*. Szerk. Christel Baier és Ugo Dal Lago. 10803. köt. Lecture Notes in Computer Science. Springer, 2018, 293–310. old. DOI: 10.1007/978-3-319-89366-2_16. URL: https://doi.org/10.1007/978-3-319-89366-2%5C_16.

[Ana+18] Abhishek Anand, Simon Boulier, Cyril Cohen, Matthieu Sozeau és Nicolas Tabareau. "Towards Certified Meta-Programming with Typed Template-Coq". *Interactive Theorem Proving - 9th International Conference, ITP 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*. Szerk. Jeremy Avigad és Assia Mahboubi. 10895. köt. Lecture Notes in Computer Science. Springer, 2018, 20–39. old. DOI: 10.1007/978-3-319-94821-8_2. URL: https://doi.org/10.1007/978-3-319-94821-8%5C_2.

[AFH18] Carlo Angiuli, Kuen-Bang Hou (Favonia) és Robert Harper. "Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities". *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*. Szerk. Dan R. Ghica és Achim Jung. 119. köt. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 6:1–6:17. DOI: 10.4230/LIPICS.CSL.2018.6. URL: https://doi.org/10.4230/LIPIcs.CSL.2018.6.

[CHM18] Thierry Coquand, Simon Huber és Anders Mörtberg. "On Higher Inductive Types in Cubical Type Theory". *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*. Szerk. Anuj Dawar és Erich Grädel. ACM, 2018, 255–264. old. DOI: 10.1145/3209108.3209197. URL: https://doi.org/10.1145/3209108.3209197.

[Kap18] Ambrus Kaposi. *Formalisation of type checking into algebraic syntax*. https://bitbucket.org/akaposi/tt-in-tt/src/master/Typecheck.agda. 2018.

[BCH19] Marc Bezem, Thierry Coquand és Simon Huber. "The Univalence Axiom in Cubical Sets". *J. Autom. Reason.* 63.2 (2019), 159–171. old. DOI: 10.1007/S10817-018-9472-6.

[Bru19] Guillaume Brunerie. "The James Construction and $\pi_4(\mathrm{S}^3)$ in Homotopy Type Theory". *J. Autom. Reason.* 63.2 (2019), 255–284. old. DOI: 10.1007/S10817-018-9468-2.

[CH19] Evan Cavallo és Robert Harper. "Higher inductive types in cubical computational type theory". *Proc. ACM Program. Lang.* 3.POPL (2019), 1:1–1:27. DOI: 10.1145/3290314. URL: https://doi.org/10.1145/3290314.

[Coq19] Thierry Coquand. "Canonicity and normalization for dependent type theory". *Theor. Comput. Sci.* 777 (2019), 184–191. old. DOI: 10.1016/J.TCS.2019.01.015. URL: https://doi.org/10.1016/j.tcs.2019.01.015.

[Hug19] Jasper Hugunin. "Constructing Inductive-Inductive Types in Cubical Type Theory". *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*. Szerk. Mikolaj Bojanczyk és Alex Simpson. 11425. köt. Lecture Notes in Computer Science. Springer, 2019, 295–312. old. DOI: 10.1007/978-3-030-17127-8_17. URL: https://doi.org/10.1007/978-3-030-17127-8%5C_17.

[Sat19] Christian Sattler. “Semantics of signatures for higher inductive-inductive types (HIITs) in complete Segal types”. https://www.cse.chalmers.se/~sattler/. 2019.

[BB20] Guillaume Brunerie és Menno de Boer. *Formalization of the initiality conjecture*. 2020. URL: https://github.com/guillaumebrunerie/initiality (elérés dátuma 2021. 07. 31.).

[BCM20] Kevin Buzzard, Johan Commelin és Patrick Massot. “Formalising perfectoid spaces”. *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*. Szerk. Jasmin Blanchette és Catalin Hritcu. ACM, 2020, 299–312. old. DOI: 10.1145/3372885.3373830.

[Com20] The mathlib Community. “The Lean Mathematical Library”. *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. CPP 2020. New Orleans, LA, USA: Association for Computing Machinery, 2020, 367–381. old. ISBN: 9781450370974. DOI: 10.1145/3372885.3373824.

[DG20] Tim Lukas Diezel és Sergey Goncharov. “Towards Constructive Hybrid Semantics”. *5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference)*. Szerk. Zena M. Ariola. 167. köt. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 24:1–24:19. DOI: 10.4230/LIPICS.FSCD.2020.24. URL: https://doi.org/10.4230/LIPIcs.FSCD.2020.24.

[Hug20] Jasper Hugunin. “Why Not W?”: *26th International Conference on Types for Proofs and Programs, TYPES 2020, March 2-5, 2020, University of Turin, Italy*. Szerk. Ugo de’Liguoro, Stefano Berardi és Thorsten Altenkirch. 188. köt. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 8:1–8:9. DOI: 10.4230/LIPICS.TYPES.2020.8. URL: https://doi.org/10.4230/LIPIcs.TYPES.2020.8.

[LS20] PETER LEFANU LUMSDAINE és MICHAEL SHULMAN. “Semantics of higher inductive types”. *Mathematical Proceedings of the Cambridge Philosophical Society* 169.1 (2020), 159–208. old. DOI: 10.1017/S030500411900015X.

[Sim20] Alex Simpson. “Synthetic Probability Theory”. Talk given at the Categorical Probability and Statistics workshop. 2020. jún. URL: http://tobiasfritz.science/2019/cps_workshop/slides/simpson.pdf.

[All+21] Guillaume Allais, Robert Atkey, James Chapman, Conor McBride és James McKinna. “A type- and scope-safe universe of syntaxes with binding: their semantics and proofs”. *J. Funct. Program.* 31 (2021), e22. DOI: 10.1017/S0956796820000076. URL: https://doi.org/10.1017/S0956796820000076.

[Ang+21] Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Robert Harper, Kuen-Bang Hou (Favonia) és Daniel R. Licata. “Syntax and models of Cartesian cubical type theory”. *Math. Struct. Comput. Sci.* 31.4 (2021), 424–468. old. DOI: 10.1017/S0960129521000347. URL: https://doi.org/10.1017/S0960129521000347.

[Bez+21] Marc Bezem, Thierry Coquand, Peter Dybjer és Martín Escardó. “On generalized algebraic theories and categories with families”. *Math. Struct. Comput. Sci.* 31.9 (2021), 1006–1023. old. DOI: 10.1017/S0960129521000268. URL: https://doi.org/10.1017/S0960129521000268.

[CCD21] Simon Castellan, Pierre Clairambault és Peter Dybjer. "Categories with Families: Unityped, Simply Typed, and Dependently Typed". *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*. Szerk. Claudia Casadio és Philip J. Scott. Cham: Springer International Publishing, 2021, 135–180. old. ISBN: 978-3-030-66545-6. DOI: 10.1007/978-3-030-66545-6_5. URL: https://doi.org/10.1007/978-3-030-66545-6_5.

[Gra+21] Daniel Gratzer, G. A. Kavvos, Andreas Nuyts és Lars Birkedal. "Multimodal Dependent Type Theory". *Log. Methods Comput. Sci.* 17.3 (2021). DOI: 10.46298/LMCS-17(3:11)2021. URL: https://doi.org/10.46298/lmcs-17(3:11)2021.

[Har21] Robert Harper. "An Equational Logical Framework for Type Theories". *CoRR* abs/2106.01484 (2021). arXiv: 2106.01484. URL: https://arxiv.org/abs/2106.01484.

[KL21] Krzysztof Kapulkin és Peter LeFanu Lumsdaine. "The simplicial model of Univalent Foundations (after Voevodsky)". *J. Eur. Math. Soc.* 23.6 (2021), 2071–2126. old.

[Sch21] Peter Scholze. *Half a year of the Liquid Tensor Experiment: Amazing developments. Blog post on the Xena project's website*. https://xenaproject.wordpress.com/2021/06/05/half-a-year-of-the-liquid-tensor-experiment-amazing-developments/. 2021.

[Uem21] Taichi Uemura. "Abstract and Concrete Type Theories". Dissz. University of Amsterdam, 2021.

[VMA21] Andrea Vezzosi, Anders Mörtberg és Andreas Abel. "Cubical Agda: A dependently typed programming language with univalence and higher inductive types". *J. Funct. Program.* 31 (2021), e8. DOI: 10.1017/S0956796821000034. URL: https://doi.org/10.1017/S0956796821000034.

[FPS22] Marcelo P. Fiore, Andrew M. Pitts és S. C. Steenkamp. "Quotients, inductive types, and quotient inductive types". *Log. Methods Comput. Sci.* 18.2 (2022). DOI: 10.46298/LMCS-18(2:15)2022. URL: https://doi.org/10.46298/lmcs-18(2:15)2022.

[Kov22] András Kovács. "Type-Theoretic Signatures for Algebraic Theories and Inductive Types". Dissz. Eötvös Loránd University, Hungary, 2022. URL: https://arxiv.org/pdf/2302.08837.pdf.

[Moe22] Hugo Moeneclaey. "Cubical models are cofreely parametric. (Les modèles cubiques sont colibrement paramétriques)". Dissz. Paris Cité University, France, 2022. URL: https://tel.archives-ouvertes.fr/tel-04435596.

[PT22] Loïc Pujet és Nicolas Tabareau. "Observational equality: now for good". *Proc. ACM Program. Lang.* 6.POPL (2022), 1–27. old. DOI: 10.1145/3498693. URL: https://doi.org/10.1145/3498693.

[Ste22] Jonathan Sterling. "First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory". Dissz. Carnegie Mellon University, USA, 2022. DOI: 10.1184/R1/19632681.V1. URL: https://doi.org/10.1184/r1/19632681.v1.

[WKS22] Philip Wadler, Wen Kokke és Jeremy G. Siek. *Programming Language Foundations in Agda*. 2022. aug. URL: https://plfa.inf.ed.ac.uk/22.08/.

[Ahr+23] Benedikt Ahrens, Jacopo Emmenegger, Paige Randall North és Egbert Rijke. "B-SYSTEMS AND C-SYSTEMS ARE EQUIVALENT". *The Journal of Symbolic Logic* (2023), 1–9. old. DOI: 10.1017/jsl.2023.41.

[Ann+23] Danil Annenkov, Paolo Capriotti, Nicolai Kraus és Christian Sattler. "Two-level type theory and applications". *Mathematical Structures in Computer Science* 33.8 (2023), 688–743. old. DOI: 10.1017/S0960129523000130.

[Gon23] Sergey Goncharov. "Representing Guardedness in Call-By-Value". *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, July 3-6, 2023, Rome, Italy*. Szerk. Marco Gaboardi és Femke van Raamsdonk. 260. köt. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 34:1–34:21. DOI: 10.4230/LIPICS.FSCD.2023.34.

[Kov23] András Kovács. *An experimental implementation of setoid type theory*. https://github.com/andrasKovacs/sett. 2023.

[Ram23] Sridhar Ramesh. "Introspective Theories and Geminal Categories". Dissz. Berkeley, 2023. URL: https://escholarship.org/uc/item/3mn0c475.

[Ses23] Filippo Sestini. "Bootstrapping Extensionality". Dissz. University of Nottingham, UK, 2023. URL: https://fsestini.github.io/pdfs/phd-draft.pdf.

[Adj+24] Arthur Adjedj, Meven Lennon-Bertrand, Kenji Maillard, Pierre-Marie Pédrot és Loïc Pujet. "Martin-Löf à la Coq". *Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2024, London, UK, January 15-16, 2024*. Szerk. Amin Timany, Dmitriy Traytel, Brigitte Pientka és Sandrine Blazy. ACM, 2024, 230–245. old. DOI: 10.1145/3636501.3636951. URL: https://doi.org/10.1145/3636501.3636951.

[Awo+24] Steve Awodey, Evan Cavallo, Thierry Coquand, Emily Riehl és Christian Sattler. "The equivariant model structure on cartesian cubical sets". *CoRR* abs/2406.18497 (2024). DOI: 10.48550/ARXIV.2406.18497. arXiv: 2406.18497. URL: https://doi.org/10.48550/arXiv.2406.18497.

[PT24] Loïc Pujet és Nicolas Tabareau. "Observational Equality Meets CIC". *Programming Languages and Systems - 33rd European Symposium on Programming, ESOP 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part I*. Szerk. Stephanie Weirich. 14576. köt. Lecture Notes in Computer Science. Springer, 2024, 275–301. old. DOI: 10.1007/978-3-031-57262-3_12. URL: https://doi.org/10.1007/978-3-031-57262-3%5C_12.