Típuselmélet
beszámoló

Alexy Marcell

2021. december 10.

Témavezető: Kaposi Ambrus

1. Típuselmélet

A félév során a típuselmélettel és típuselmélettel foglalkoztam, és a típuselmélet alkalmazásával az Agda programozási nyelven keresztül. A típuselmélethez a Homotopy type theory könyvnek az első fejezetét és A Bevezetés a homotópia-típuselméletbe jegyzetet dolgoztam fel, amely a Martin-Löf típuselméletet mutatja be.

A típuselmélet alapvetően kifejezésekkel és azok típusaival foglalkozik. Ez egy formális matematikai rendszer, amellyel a konstruktív matematikai bizonyításokat lehet formalizálni.

A típuselmélet alapfogalma a típus, amely elemei egy előre meghatározott szabályok szerint viselkednek. Ezeknek a szabályoknak a felépítését az alábbi példákon keresztül lehet jól követni.

1.1. Példák típusokra

Egy új fajta típus bevezetése során általában öt lényeges szempont szerint történik:

- Először le kell írni, hogy hogyan képezhetjük ezt a fajta típust az előzőekből, a képzési szabályok. (Formation rules)
- Majd le kell írni, hogy hogyan vezethetjük be a típus elemeit, a bevezetési szabályok. (Introduction rules)

- Hogyan használhatjuk fel a típus elemeit, mik az elimináció szabályai. (Elimination rules)
- Az elemek bevezetési szabályai és felhasználási szabályai közt mi az összefüggés, ezek a számítási szabályok. (Computation rules)
- Az elemek közti egyediségre vonatkozó szabályok. Általában ha egy elemet eliminálunk, majd az ebből kapott értékekből újra bevezetjük, akkor az eredetivel egyforma értéket kell kapnunk. (Uniqueness principle)

Egyszerűbb típusoknál általában nem mindegyik szabálycsoport van meg, lehet hogy néhány szabálycsoport nem szükséges vagy triviális. A szabályokat következő példán keresztül be lehet jól mutatni:

1.1.1. Szorzat típusok

Ha van két különböző típusunk, akkor vehetjük az ezeknek az elemeiből álló párok típusát. Ehhez a következő jelöléseket használhatjuk:

Legyen adva az A és B típus. Ekkor ezekből képezhetjük az $A \times B$ típust, az A és B szorzatának a típusát.

Ha van adva egy $a : A$ és $b : B$, akkor bevezethetjük az $(a, b) : A \times B$ párt.

Ha van egy $p : A \times B$ párunk, akkor ezt kétféleképp eliminálhatjuk. Vagy vesszük az első elemet $(\pi_1 p : A)$, vagy vegyük a második elemét $(\pi_2 p : B)$ a párnak.

Természetesen az eliminációk értékét a képzésekből ki lehet számolni. Legyen $\pi_1(a, b) \equiv a$ és $\pi_2(a, b) \equiv b$.

Továbbá a ha két pár értéke megegyezik, akkor egyformának tekintjük őket. Vagyis $p = (\pi_1 p, \pi_2 p)$

1.1.2. Egyszerűbb típusok

Üres típus: Jelölés $\bot$. Ennek a típusnak nincs eleme. Viszont bármely C típus esetén ha $t : \bot$, akkor exfalso $t : C$ -vel eliminálni lehet.

Unit / egy elemű típus: Jelölés $\top$. Ennek a típusnak egy eleme van, amit *triv* -vel szoktak jelölni.

Függvény típus: Jelölés $A \to B$, ahol A a függvény alaphalmaza és B a függvény értékkészlete. Függvényeket a $\lambda x \mapsto t$ kifejezéssel szokták bevezetni, az eliminációs szabály a függvény kiértékelése. Két függvény egyforma, ha minden pontban ugyanaz az értéke.

Természetes számok Jelölés $\mathbb{N}$. Ebbe a típusba elemeket két módon lehet bevezetni: Vesszük a $0 : \mathbb{N}$ elemet, vagy ha van egy $n : \mathbb{N}$ számunk, akkor létrehozhatjuk az $\text{suc}\ n : \mathbb{N}$, az n-re rákövetkező számot.

A természetes számokon eliminálni lehet rekurzív függvényeket az rec $c_0 c_s n$ eliminátorral. Ha adva van egy $c_0 : A$ kezdőérték, és egy $c_s : A \to A$ függvény, akkor a *rec* ezt függvényt n -szer alkalmazza önmagára a c_0 kezdőértékkel. A bevezetési szabályok miatt n vagy 0 vagy $\text{suc}\ m$ alakú, ahol $m : \mathbb{N}$. Ez alapján rec $c_0\ c_s\ 0 = c_0$ és rec $c_0\ c_s(\text{suc}\ m) = c_s(\text{rec}\ c_0\ c_s\ m)$.

Két típus (diszjunkt) uniója / Coproduct: A típuselméletben általában a különböző típusokhoz tartozó elemek különbözőek, ezért típusok uniója csak a diszjunkt unió lehet. Ezt $A \uplus B$ -vel jelöljük. Az elemeket két módon képezhetjük: Vagy egy $a : A$-beli elemből az $\iota_1 a : A \uplus B$ konstruktorral, vagy $b : B$ -ből $\iota_2 b : A \uplus B$-vel. Szétbontani a case $t\ u\ v$ eliminátorral lehet, ahol $t : A \uplus B$, $u : A \to C$ és $v : B \to C$. Ennek a kiszámolását esetszétválasztással lehet elvégezni, ha $t = \iota_1\ a$ alakú, akkor $case(\iota_1\ a)\ u\ v = u\ a$, a másik esetben pedig$case(\iota_2\ b)\ u\ v = v\ b$.

1.1.3. Összetett típusok

Univerzum: A típusoknak is van típusa, ezek különböző U_i univerzumokba sorolhatóak. A legelső univerzum az U_0, ennek a típusa az U_1 univerzum, ennek a típusa az U_2, és így tovább. Ezek segítségével lehet beszélni, például $f : A \to U_0$ függvényekről, amely egy $a : A$ elemhez egy C típust rendel. Ha nem függ a kifejezés az univerzum indexétől, akkor nem szükséges kiírni.

Σ típus, függő párok típusa: Legyen adva egy $A : U$ típus, és egy $B : A \to U$ függvény. Ekkor képezhetjük a $\Sigma_{a:A} B(a)$ típust. Ennek az elemei az olyan (a, b) párok lesznek, ahol $a : A$ és $b : B(a)$, vagyis b típusa függ az a értékétől.

Π típus, függő függvények típusa: Egy $A : U$ típussal és $B : A \to U$ függvénnyel képezhetjük a $\Pi_{a:A} B(a)$ típust, a függő függvények típusát. Ez a típus hasonló a (nem függő) szorzat típushoz, sőt, ha B egy konstans függvény, akkor az $A \to B$ függvénytípust kapjuk. Az elemei függő függvények lesznek, amelyek értékkészlete függ a kiértékelés helyétől, ha f ennek a típusnak az eleme, akkor $a : A$ -ban $f(a) : B(a)$ típusú lesz az eredmény.

1.2. Összefüggés a típuselmélet és a logika között

A típuselmélet és logika közti összefüggésre mutat rá a Curry-Howard izomorfizmussal. A logikai állítások megfeleltethetőek típusoknak, egy állítása bizonyítása pedig megfelel egy elemnek. Így ha adott egy olyan program, melynek a típusa épp a bizonyítandó tételnek felel meg, és a program sikeresen lefut, vagyis visszaad egy értéket a tétel típusával, akkor ezzel a tételt is bebizonyítottuk. Az összefüggés részletesebben (típusok nagy, elemek kis betűvel):

típus	logika
$A : U$	állítások halmaza
$a : A$	az A állítás bizonyítása
$\bot$	hamis állítás
$\top$	igaz állítás
$A \uplus B$	A vagy B
$A \times B$	A és B
$A \to B$	A-ból következik B
$A \to \bot$	az A állítás tagadása
$B(x) : A \to U$	az A-n értelmezett állítások
$\Sigma_{a:A} B(a)$	$\exists x : A$ hogy $B(x)$ teljesüljön
$\Pi_{a:A} B(a)$	$\forall x : A$ teljesül $B(x)$

2. Agda

A félév során az Agda programnyelvvel is foglalkoztam. Ez a nyelv képes függő típusok kezelésére és ellenőrzésére, ezáltal bizonyításokat tud ellenőrizni. A nyelvben be lehet vezetni a fent jelzett egyszerűbb típusok közül néhányat az alábbi módon:

```
data Bot : Set where

exfalso : {C : Set} -> Bot -> C
exfalso ()

data Top : Set where
  triv : Top

data N : Set where
  zero : N
  suc : N -> N

rec : {C : Set} -> C -> (C -> C) -> N -> C
rec c0 cs zero = c0
rec c0 cs (suc n) = cs (rec c0 cs n)
```

Források

Agda (https://wiki.portal.chalmers.se/agda/pmwiki.php?n=Main.HomePage)

The Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*, első fejezet.

Kaposi Ambrus, *Bevezetés a homotópia-típuselméletbe*

Diviánszky Péter, *Agda Tutorial*

Jan Malakhovski, *Brutal [Meta]Introduction to Dependent Types in Agda*