EÖTVÖS LORÁND TUDOMÁNYEGYETEM
Természettudományi kar

Program ekvivalencia és logikai relációk

Diplomamunka

Szalai Zsolt
matematikus hallgató

Konzulens:
Csörnyei Zoltán, egyetemi docens
Dr., PhD. matematikus
Programozási nyelvek és Fordítóprogramok Tanszék

Budapest, 2009

Tartalomjegyzék

1. fejezet

Bevezetés

Egy problémára sokféle algoritmus és megtöbb fajta implementáció születhet. Így jogos a kérdés, vajon tudjuk–e particionálni a programokat aszerint, hogy „ugyanazt csinálják–e”? Már precízen definiálni sem könnyű ezt a fogalmat, több megközelítés lehetséges, melyek közül ebben a dolgozatban a környezeti ekvivalenciát fogjuk bevezetni és a hozzá kifejlesztett módszerek egyikét, a logikai relációk elvét fogjuk bemutatni.

Ez a módszer teljes mértékben a típusrendszer és a nyelv operációs szemantikájára épít, az első lépcsőfok az ekvivalencia megértése felé. A vizsgált nyelv típusai döntő szerepet kapnak a vizsgálatok során, többféleképpen fogjuk tudni jellemezni az ekvivalenciát az egyes típusokon, sőt az elméletet alkalmazni fogjuk egy példán keresztül nemcsak kifejezések, de absztrakt adattípusok közti ekvivalenciára is. Mivel az ekvivalencia konstrukciójával nehézkes dolgozni, többféle karakterizációt foguk mutatni, például a ciu–ekvivalenciát és ilyen lesz a dolgozat tárgya, a logikai reláció is. A vizsgált nyelv determinisztikus szemantikájú, de azért elég erős lesz. Vizsgálni fogjuk az ekvivalenciát univerzálisan és egzisztenciálisan kvantifikált típusukra, rekurzív függvényekre is, azonban rekurzív típusokra a probléma a mai napig nem megoldott. A módszer erejét mutatja, hogy igen erős tételeket tudunk belátni segítségével, mint például a 3.5.1. és 3.5.2., de rekurzív típusoknál bár lehet logikai relációt definiálni, az gyengén közelíti a környezeti ekvivalenciát, ezért úgy néz ki, itt gyökeresen új ötlet kell majd. Nemdeterminisztikus szemantikával rendelkező nyelvek esetén a logikai relációk módszere sokat veszít erejéből, erre az esetre más konstrukciót érdemes használni, a „hasonlóságot”, de ezzel is a rekurzív adattípusok esete nyitott marad.

A környezeti ekvivalencia fogalmát először Morris [7] használta bizonyítási módszerként, innentől többen tanulmányozták külön a témakört. A logikai relációk módszere pedig egészen Plotkin [15] és Stateman [17] 70–es és 80–as évekbeli munkájáig nyúlik vissza. Mitchell [6] denotációs szemantikával leírt nyelvekre vizsgálta az ekvivalencia fogalmát, a szimuláció segítségével. 1993–ban Plotkin és Abadi [1] a polimorfikus λ–kalkulusra is alkalmazta az elméletet, fő eredményük, hogy bizonyítási principátumokat adtak, bár a rekurziót nélkülözte a mintanyelvük. Pitts [12][13] az ezredfordulón kiterjeszti mindezt rekurzív függvényekre is, amihez a megengedhetőségi feltételre van szükség.

A 2. fejezetben bemutatjuk a logikai relációk alkalmazását egy egyszerű esetben, definiáljuk az egyszerű típusos λ-kalkulust, Unit típussal és a nyelv egy definiciós ekvivalenciáját. Sikerül algoritmust is adni az ekvivalencia eldöntésére, melynek teljességét, azaz hogy az ekvivalenciából következik az algoritmus által meghatározott ekvivalencia, a logikai relációk módszerével fogjuk belátni. Definiálunk egy speciális relációt, és bebizonyítjuk, hogy az megegyezik mindkét ekvivalencia relációval. Ezt a két állítást a fő lemma és a fő tétel fogja kimondani.

A 3. fejezetben általánosabban vizsgáljuk a kérdést és bevezetjük a környezeti ekvivalencia fogalmát, ami a leginkább elfogadott definíciója λ kifejezések közti ekvivalenciának. Lényegében akkor tekintjük ekvivalensnek két kifejezést, ha minden környezetben ugyanazt a megfigyelhető eredményt adjak. Ez a megfogalmazás persze még finomításra szorul. Definiáljuk a vizsgált nyelvet, F_{ML} –t, majd a környezeti ekvivalenciára jellemzést adunk operációs szabályok segítségével. Keretrendszereket használva a ciu–ekvivalenciával is jellemzést adunk. Bevezetjük a típusok hatását a kifejezés–relációkon, ami egy lezárási operátor a relációk hálóján. Ez az absztrakt fogalom vezet el minket a Δ logikai relációhoz. Belátjuk, hogy valóban a környezeti ekvivalenciával egyezik meg, és erejét felhasználjuk további jellemzések, kiterjesztési lemmák bizonyítására, mint például függvények esetén, mely azt állítja, hogy két függvény pontosan akkor ekvivalens, ha bármely ekvivalens argumentumra is applikálva őket, ugyancsak ekvivalens eredményeket kapunk. A 3.5.3. és 3.5.4. fejezetekben konkrét példákon keresztül mutatjuk meg a logikai reláció alkalmazását, rávilágítunk a típushatás és kiterjesztési lemma lényegére egzisztenciális típus esetén, illetve típusokra is alkalmazzuk majd az elméletet: Belátjuk két polimorfikus típus környezeti egyelőségét, amik listákat reprezentálnak.

2. fejezet

Egy egyszerű logikai reláció

Jelen fejezetben bemutatjuk, hogyan működik egyszerű esetben a logikai relációk módszere, ezért sokszor apróbb részletek bizonyításától eltekintünk, illetve ahol lehet és egyértelmű, egyszerűsítéseket alkalmazunk.

2.1. A probléma

Az egyszerű típusos λ-kalkulust fogjuk használni, hogy eldöntsük két adott nyelvbeli kifejezésről, hogy azok ekvivalensek–e. Továbbá a nyelvhez hozzáadunk egy bázis típust is, azaz egy előre definiált konstanshalmazt. E típus elemeit k-val fogjuk jelölni.

Az ekvivalenciát $\Gamma \vdash s \equiv t : T$ fogja jelölni, ami azt jelenti, hogy a Γ típuskörnyezetben s és t ekvivalensek miközben típusuk T. Ezt a fajta ekvivalenciát szokás „definíciós” ekvivalenciának is hívni, hiszen közvetlenül szabályokkal definiáljuk. A szabályok nagy része egyszerű, azaz, hogy a szóban forgó reláció egy ekvivalencia, és egyben kongruencia is az absztrakcióra és az applikációra nézve. A nehézséget az utolsó szabály fogja okozni, mely szerint két függvény ekvivalens, ha ugyanazzal az argumentummal vett applikációjuk ekvivalens. Ez utóbbit kiterjesztési szabálynak is nevezik, melynek jelentését és jelentőségét a későbbiekben fogjuk látni.

Szinataxis

$$t := \; x \mid \lambda x : T.t \mid t\,t \mid k \qquad \text{kifejezés}$$

$$T := \; B \mid T \to T \qquad \text{típus}$$

$$\Gamma := \; \emptyset \mid \Gamma, x : T \qquad \text{kontextus}$$

Típusozás $\Gamma \vdash t : T$

$$\frac{x : T \in \Gamma}{\Gamma \vdash t : T} \qquad \text{(T-VAR)}$$

$$\frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1.t_2 : T_1 \to T_2} \qquad \text{(T-ABS)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \quad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1 t_2 : T_{12}} \qquad \text{(T-APP)}$$

$$\Gamma \vdash k : B \qquad \text{(T-CONST)}$$

Redukció $s \Rightarrow s$

$$t \Rightarrow t \qquad \text{(QR-REFL)}$$

$$\frac{s_2 \Rightarrow t_2}{\lambda x : T_1.s_2 \Rightarrow \lambda x : T_1.t_2} \qquad \text{(QR-ABS)}$$

$$\frac{s_1 \Rightarrow t_1 \quad s_2 \Rightarrow t_2}{s_1 s_2 \Rightarrow t_1 t_2} \qquad \text{(QR-APP)}$$

$$\frac{s_1 \Rightarrow t_1 \quad s_2 \Rightarrow t_2}{(\lambda x : T_1.s_1)s_2 \Rightarrow t_1[x := t_2]} \qquad \text{(QR-BETA)}$$

$$\frac{s \Rightarrow t \quad x \notin ft(s)}{\lambda x : T.sx \Rightarrow t} \qquad \text{(QR-ETA)}$$

Ekvivalencia $\Gamma \vdash s \equiv t : T$

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash t \equiv t : T} \qquad \text{(Q-REFL)}$$

$$\frac{\Gamma \vdash t \equiv s : T}{\Gamma \vdash s \equiv t : T} \qquad \text{(Q-SYMM)}$$

$$\frac{\Gamma \vdash s \equiv t : T \quad \Gamma \vdash t \equiv u : T}{\Gamma \vdash s \equiv u : T} \qquad \text{(Q-TRANS)}$$

$$\frac{\Gamma, x : T_1 \vdash s_2 \equiv t_2 : T_2}{\Gamma \vdash \lambda x : T_1.s_2 \equiv \lambda x : t_1.t_2 : T_1 \to T_2} \qquad \text{(Q-ABS)}$$

$$\frac{\Gamma \vdash s_1 \equiv t_1 : T_1 \to T_2 \quad \Gamma \vdash s_2 \equiv t_2 : T_1}{\Gamma \vdash s_1 s_2 \equiv t_1 t_2 : T_2} \qquad \text{(Q-APP)}$$

$$\frac{\Gamma, x : T_1 \vdash s_{12} \equiv t_{12} : T_2 \quad \Gamma \vdash s_2 \equiv t_2 : T_1}{\Gamma \vdash (\lambda x : T_1.s_{12})s_2 \equiv t_{12}[x := t_2] : T_2 : T_2} \qquad \text{(Q-BETA)}$$

$$\frac{\Gamma, x : T_1 \vdash sx \equiv tx : T_2}{\Gamma \vdash s \equiv t : T_1 \to T_2} \qquad \text{(Q-EXT)}$$

2.1. táblázat. Egyszerű típusos λ-kalkulus

2.2. Nem típusvezérelt ekvivalencia vizsgálat

A hagyományos, legegyszerűbb módszer az ekvivalencia eldöntésére az, ha normalizáljuk a kifejezéseket, majd összehasonlítjuk őket. Tehát s és t pontosan akkor ekvivalensek, ha normálformáik ugyanazok(α-konverziótól eltekintve). A módszer alkalmazásához a következő három állításnak kell teljesülnie:

- Az ekvivalencia szabályaiból egy redukciós relációt kell tudni készíteni, amire igaz, hogy $\Gamma \vdash s \equiv t : T$ pontosan akkor, ha $s \Leftrightarrow^* t$, ahol $\Leftrightarrow^*$ a redukciós reláció szimmetrikus és tranzitív lezárása.
- A redukciónak egyesítőnek kell lennie, azaz ha $r \Rightarrow^* s$ és $r \Rightarrow^* t$, akkor $\exists u\, s \Rightarrow^* u$ és $t \Rightarrow^* u$.
- A redukciónak normalizálónak kell lennie, azaz minden kifejezésnek kell létezzen hatékonyan kiszámítható normálformája.

Ezekből bizonyítható a következő lemma:

2.2.1. Lemma. *Legyen $\Rightarrow$ egyesítő, s és t normál formái s' és t'. Ekkor $s \Leftrightarrow^* t$ akkor és csak akkor, ha $s' = t'$.*

Ezek sokszor, bonyolultabb illetve kifejezőbb nyelveknél nagy elvárások, így a módszer ezekben az esetekben nem alkalmazható.

2.3. Típusvezérelt ekvivalencia

Ezen felül vannak esetek, amikor a fenti módszer azért sem alkalmazható, mert az ekvivalencia definíciója típusérzékeny. Adjuk hozzá a nyelvhez a `Unit` típust, aminek egyetlen eleme van, a `unit`.

Ezért evidens ennek ekvivalencia szabálya is, (Q-UNIT), ami azt fejezi ki, hogy bármely két elem a `Unit` típusból ekvivalensek egymással. Például a

$$\Gamma \vdash \texttt{unit} \equiv \texttt{unit} : \texttt{Unit}$$

Szintaxis		*Típusozás*	$\Gamma \vdash t : T$
$t := \dots \mid \texttt{unit}$	(unit)	$\Gamma \vdash \texttt{unit} : \texttt{Unit}$	(T-UNIT)
$T := \dots \mid \texttt{Unit}$	(Unit)	*Ekvivalencia*	$\Gamma \vdash s \equiv t : T$
		$\dfrac{\Gamma \vdash s : \texttt{Unit} \quad \Gamma \vdash t : \texttt{Unit}}{\Gamma \vdash s \equiv t : \texttt{Unit}}$	(Q-UNIT)

2.2. táblázat. A Unit típus

szabály bár helyes, mégsem lehet segítségével Unit-ra vonatkozó ekvivalenciát levezetni. Ha x és y normálformában vannak, nem biztos, hogy tudunk dönteni, csak a típusuk alapján és ahhoz kell az erősebbik, (Q-UNIT) szabály.

2.4. Az algoritmus

Algoritmust szeretnénk adni $\Gamma \vdash s \equiv t : T$ eldöntésére, ha s és t kifejezések adottak ugyanabból a típusból. Típusvezérelt algoritmusunkat a következő megfigyelésekből építjük fel:

1. Ha $T = \texttt{Unit}$, akkor azonnal igent mondhatunk az erős szabály alapján.
2. Ha T $T_1 \to T_2$ alakú, visszavezetjük a problémát egy egyszerűbbre, ahol T már csak T_2 alakú. Az eredetivel ekvivalens átírás a következő:

$$\Gamma, x : T_1 \vdash sz \equiv tx : T_2$$

Ez helyes következtetés, hisz az utóbbiból (Q-EXT)-et felhasználva azonnal látszik az előbbi. Megfordítva, a (Q-APP) szabályt kell felhasználni a gyengítő lemmával együtt:

2.4.1. Lemma (Gyengitő lemma)**.** *Ha* $\Gamma \vdash s \equiv t : T$*, akkor* $\Gamma, x : S \vdash s \equiv t : T$

A feladatot, bármilyen típusról, így sikerült leredukálni egyedül a bázis típusra, tehát már csak ott kell eldönteni az ekvivalenciát.

Ebben a helyzetben működhet a hagyományos normalizációs módszer, hiszen a típusérzékenységet most már nem kell számításba venni. A normalizációs fázis az alábbi öt forma egyikéhez vezet:

1. $\Gamma \vdash xs_1 \ldots s_n \equiv xt_1 \ldots t_n : B$
2. $\Gamma \vdash k \equiv k : B$
3. $\Gamma \vdash xs_1 \ldots s_m \equiv yt_1 \ldots t_n : B$, ahol $x \neq y$
4. $\Gamma \vdash xs_1 \ldots s_n \equiv k : B$
5. $\Gamma \vdash k \equiv k' : B$, ahol $k \neq k'$

Világos, hogy a 2. esetben azonnal visszaterhet az algoritmus pozitív válasszal és az 5.–ben negatívval. A 3-as és a 4-es esetben szintén leállhat az algoritmus, és az is igazolható, hogy ekkor sem lehetnek relációban, így egyetlen út marad a további vizsgálódásnak, az első pont. Vegyünk egy egyszerűbb esetét, $xs \equiv xt : B$-t, ahol $x\ T \rightarrow B$ típusú. Ekkor elég eldöntenünk, hogy $s \equiv t : T$ (Q-APP) és (Q-REFL) miatt. Azaz most ismét egy általános ekvivalencia kérdéssel kell folytatni az algoritmust, a T típusra, tehát a szintaktikai ellenőrzés nem segíthet, az algoritmus típusvezérelt részére van szükség.

Vegyük észre, hogy az előbb előírt forma nem is teljesen normalizált formája a kifejezésnek. Például vegyük az $xs \equiv xt : T$-t, ahol $x : \texttt{Unit} \rightarrow B$ és $s, t : \texttt{Unit}$. Ekkor teljesen értelmetlen, hogy a részkifejezéseket normalizáljuk. Elég tehát gyenge fej normalizálni, azaz mindig a legbaloldalibb, legkülső redexet választjuk redukcióra, majd azonnal készen vagyunk, ha a kifejezés mással kezdődik mint egy λ-absztrakció.

2.4.2. Definíció. Azokat a $\texttt{B}$ típusú kifejezéseket, melyek k konstansok, vagy a fenti $xt_1 \ldots t_n$ alakban állnak elő, útvonalaknak nevezzük.

Az előbbi megfigyelésekre alapozva a 2.3. ábra írja le az algoritmust és definiálja a felhasznált relációkat.

Útvonal

$$p, q := x \mid pt \mid k$$

Gyengе fej redukció $\qquad s \rightsquigarrow t$

$$(\lambda x : T_{11}.t_{12})t_2 \rightsquigarrow t_{12}[x := t_2] \qquad \text{(QAR-BETA)}$$

$$\frac{t_1 \rightsquigarrow t_1'}{t_1 t_2 \rightsquigarrow t_1' t_2} \qquad \text{(QAR-APP)}$$

Gyenge fej normalizáció $\qquad s \Downarrow t$

$$\frac{s \rightsquigarrow t \quad t \Downarrow u}{s \Downarrow u} \qquad \text{(QAN-REDUCE)}$$

$$\frac{t \not\rightsquigarrow}{t \Downarrow t} \qquad \text{(QAN-NORMAL)}$$

Algoritmikus kifejezés ekvivalencia $\qquad \Gamma \vdash s \Leftrightarrow t : T$

$$\frac{s \Downarrow p \quad t \Downarrow q \quad \Gamma \vdash p \leftrightarrow q : B}{\Gamma \vdash s \Leftrightarrow t : B} \qquad \text{(QAT-BASE)}$$

$$\frac{\Gamma, x : T_1 \vdash sx \Leftrightarrow tx : T_2}{\Gamma \vdash s \Leftrightarrow t : T_1 \to T_2} \qquad \text{(QAT-ARROW)}$$

$$\Gamma \vdash s \Leftrightarrow t : Unit \qquad \text{(QAT-ONE)}$$

Algoritmikus útvonal ekvivalencia $\qquad \Gamma \vdash p \leftrightarrow q : T$

$$\frac{x : T \in \Gamma}{\Gamma \vdash x \leftrightarrow x : T} \qquad \text{(QAP-VAR)}$$

$$\frac{\Gamma \vdash p \leftrightarrow q : T_1 \to T_2 \quad \Gamma \vdash s \Leftrightarrow t : T_1}{\Gamma \vdash ps \leftrightarrow qt : T_2} \qquad \text{(QAP-APP)}$$

$$\Gamma \vdash k \leftrightarrow k : B \qquad \text{(QAP-CONST)}$$

2.3. táblázat. Ekvivalencia algoritmus λ_b–hez

1. Algoritmikus kifejezés ekvivalencia, $\Gamma \vdash s \Leftrightarrow t : T$. Ez a reláció vezet vissza egy általános problémát a bázis típusra.

2. Algoritmikus útvonal ekvivalencia, $\Gamma \vdash p \leftrightarrow q : T$. Ez az útvonalak struktúrája szerint vezérelt része az algoritmusnak. Ellenőrizzük p és q „fejeit”, hogy azok azonosak–e, majd ha kell, a megfelelő részkifejezéseket vizsgáljuk algoritmikus kifejezés ekvivalenciára.

3. Gyenge fej redukció, $s \leadsto t$. A gyenge fej normalizáció alaplépése, amely s fejében egy redexet redukál.

4. Gyenge fej normalizáció, $s \Downarrow t$. s gyenge fej normalizációját végzi, gyenge fej redukciók sorozatával.

2.5. Teljesség

Következő feladatunk, hogy belássuk, az algoritmus jól működik, azaz helyes és teljes. Az algoritmus helyes, ha csakis ekvivalens kifejezésekre ad pozitív választ, vagyis $\Gamma \vdash s \Leftrightarrow t : T$, akkor $\Gamma \vdash s \equiv t : T$. Megfordítva, az összes ekvivalens kifejezéspárra pozitív választ ad, tehát,

2.5.1. Tétel. *Ha* $\Gamma \vdash s \equiv t : T$*, akkor* $\Gamma \vdash s \Leftrightarrow t : T$

A helyességet közvetlenül indukcióval be lehet bizonyítani. A teljesség ellenben nehézségeket rejt magában, amit a logikai relációk módszerével fogunk tudni leküzdeni.

A 2.5.1. tétel azaz a szabályok nagyrésze szintúgy bizonyítható indukcióval. (Q-ABS) esetében például $\Gamma \vdash \lambda x : T_1.s_2 \Leftrightarrow \lambda x : T_1.t_2 : T_1 \to T_2$–t, amihez elég látni, hogy

$$\Gamma, x : T_1 \vdash (\lambda x : T_1.s_2)x \Leftrightarrow (\lambda x : T_1.t_2)x : T_2$$

Ezzel az a gond, hogy az indukciós hipotézisből,

$$\Gamma, x : T_1 \vdash s_2 \Leftrightarrow t_2 : T_2$$

de ezt nem lehet közvetlenül bizonyítani. Szükség van a gyenge fej lezárási lemmára:

2.5.2. Lemma (Gyenge fej lezárási lemma)**.** *Ha* $\Gamma \vdash s \Leftrightarrow t : T$*,* $s' \rightsquigarrow^* s$ *és* $t' \rightsquigarrow^* t$*, akkor* $\Gamma \vdash s' \Leftrightarrow t' : T$*.*

2.5.3. Lemma (Algoritmikus szimmetria)**.** *Ha* $\Gamma \vdash s \Leftrightarrow t : T$*, akkor* $\Gamma \vdash t \Leftrightarrow s : T$

2.5.4. Lemma (Algoritmikus tranzitivitás)**.** *Ha* $\Gamma \vdash s \Leftrightarrow t : T$ *és* $\Gamma \vdash t \Leftrightarrow u : T$*, akkor* $\Gamma \vdash s \Leftrightarrow u : T$

Ezek a lemmák szintén indukcióval bizonyíthatóak.

Az applikációs szabályok bizonyításánál viszont nehézségekbe ütközünk, aminek az az oka, hogy ebben az esetben az indukciós feltevés nem tud semmilyen információval szolgálni azzal kapcsolatban, mi történik, amikor egy kifejezést applikálunk egy argumentumra. (Q-APP) esetében a $\Gamma \vdash s_1 \Leftrightarrow t_1 : T_1 \to T_2$ és $\Gamma \vdash s_2 \Leftrightarrow t_2 : T_2$ indukciós hipotézisből $\Gamma \vdash s_1 s_2 \Leftrightarrow t_1 t_2 : T_2$–t kéne levezetni, de ez nem megy. A legközelebbi forma amihez így eljuthatunk, az a $\Gamma, x : T_1 \vdash s_1 x \Leftrightarrow t_1 x : T_2$, és nem tovább, hiszen az algoritmus azon része, ami $s_1 x$–et $t_1 x$–el hasonlítja össze, teljesen másképp működik mint ami $s_1 s_2$–t $t_1 t_2$–vel hasonlítja össze.

2.5.5. Egy logikai reláció

A problémát az okozza, hogy az algoritmikus ekvivalenciáról nem látszik azonnal, hogy logikai, a következő értelemben:

2.5.6. Definíció. Tegyük fel, hogy $R(s, t, T)$ olyan reláció, melyben s és t típusa is T. Ekkor R logikai, ha valamikor is $R(s_1, t_1, T_1 \to T_2)$ és $R(s_2, t_2, T_1)$ fennáll, akkor $R(s_1 s_2, t_1 t_2, T_2)$ is fennáll.

Magyarán, hogy zárt az applikációra nézve. A logikai név onnan ered, hogy zártságot követel meg valamilyen, a típusok közötti logikai operátorra. A bizonyítás stratégiája, felhasználva ezt az új fogalmat, úgy módosul, hogy a definíciós és algoritmikus ekvivalencia közé beépítünk egy logikai relációt is. Ez lesz a logikai ekvivalencia két kifejezésre nézve. Megmutatjuk, hogy a logikai ekvivalenciából következik az algoritmikus, és hogy a definíciós pedig implikálja a logikait.

A relációt a típus szerinti indukcióval definiáljuk, pont úgy ahogy nekünk szükségünk lesz rá, vagyis függvénytípusokra pont a zártságot fogjuk megkövetelni:

$\Gamma \vdash s|t : T$ pontosan akkor, ha
$T = \texttt{Unit}$, vagy
$T = B$ és $\Gamma \vdash s \Leftrightarrow t : B$, vagy
$T = T_1 \to T_2$ és $\forall s', t'$ és $\forall \Gamma' \supseteq \Gamma$
$\Gamma' \vdash s'|t' : T_1$ akkor $\Gamma' \vdash ss'|tt' : T_2$

Vegyük észre, hogy a függvénytípusnál azt is megköveteljük, hogy egy tetszőleges, bővebb környezetben teljesüljön a logikai reláció. Ez valóban szükséges és lényeges kritériuma a definíciónak, ugyanis amikor függvénytípusra szeretnénk igazolni, hogy a logikai relációból következik az algoritmikus is, ahhoz elengedhetetlen a következő formula is: $\Gamma, x : T_1 \vdash sx \Leftrightarrow tx : T_2$. Ez az indukcióból nem jön ki, de ha be tudnánk látni, hogy $\Gamma, x : T_1 \vdash sx|tx : T_2$ is teljesül, már készen lennénk. Ez utóbbi könnyen be is látható, de csak akkor, ha megvan a relációnak a monotonitási tulajdonsága, azaz pont az amit meg is követeltünk tőle a definíciójában.

A következő lemmák a mononitást mondjak ki a relációkra, indukcióval lehet őket belátni:

2.5.7. Lemma (Algoritmikus monotonicitás)**.** *Tegyük fel, hogy* $\Gamma' \supseteq \Gamma$. *Ekkor*

1. *Ha* $\Gamma \vdash s \Leftrightarrow t : T$, *akkor* $\Gamma' \vdash s \Leftrightarrow t : T$
2. *Ha* $\Gamma \vdash s \leftrightarrow t : T$, *akkor* $\Gamma' \vdash s \leftrightarrow t : T$

2.5.8. Lemma (Logikai monotonicitás)**.** *Ha* $\Gamma \vdash s|t : T$ *és* $\Gamma' \supseteq \Gamma$, *akkor* $\Gamma' \vdash s|t : T$

2.5.9. A fő Lemma

Összegezve, mostmár úgy tűnik megvan a szükséges közbülső reláció, és készek vagyunk, hogy belássuk az algoritmus teljességet, amihez az kell, hogy a definíciós ekvivalencia implikálja a logikait és a logikaiból következzen az algoritmikus. Ezt az utóbbi állítást fogjuk úgy hívni, hogy a Fő lemma, ami igazából két dolgot állít. Először is a fent említett implikációt, másrészről, ahhoz hogy ezt lássuk, tudnunk kell, hogy a változók saját magukkal logikailag ekvivalensek–e. Ennél egy erősebb állítást fogalmaz meg a lemma:

2.5.10. Lemma (Fő lemma)**.**

1. *Ha* $\Gamma \vdash s|t : T$ *akkor* $\Gamma \vdash s \Leftrightarrow t : T$.
2. *Ha* $\Gamma \vdash s \leftrightarrow t : T$ *akkor* $\Gamma \vdash s|t : T$.

Bizonyítás. T struktújá szerinti indukcióval.

$T = B$

1. Definició szerint.
2. Mivel p és q útvonalak, (QAN-NORMAL) szerint $p \Downarrow p$ és $q \Downarrow q$. (QAT-BASE)–ből $\Gamma \vdash p \Leftrightarrow q : B$, így definició szerint készen vagyunk.

$T = Unit$

1. (QAT-ONE)–ból azonnal következik.
2. Definicióból azonnal következik.

$T = T_1 \to T_2$

1. Elég megmutatni, hogy $\Gamma, x : T_1 \vdash sx \Leftrightarrow tx : T_2$. Ha $\Gamma, x : T_1 \vdash sx|tx : T_2$, akkor indukcióval ez megmutatható: (QAP-VAR) szerint $\Gamma, x : T_1 \vdash x \leftrightarrow x : T_1$, és így $\Gamma, x : T_1 \vdash x|x : T_1$. Mivel $\Gamma \subseteq (\Gamma, x : T_1)$, $\Gamma, x : T_1 \vdash sx|tx : T_2$ is teljesül.

2. Tegyük fel, hogy $\Gamma \subseteq \Gamma'$ és $\Gamma' \vdash s|t : T_1$. Azt kell megmutatni, hogy $\Gamma' \vdash ps|qt : T_2$. Ez indukcióval bizonyítható, ha tudjuk hogy $\Gamma' \vdash ps \leftrightarrow qt : T_2$ teljesül.

   Az előző pontot és indukciót használva, $\Gamma' \vdash s \Leftrightarrow t : T_1$. Az algoritmikus monotonicitási lemmát(2.5.7.) használva $\Gamma' \vdash p \leftrightarrow q : T_1 \rightarrow T_2$, és így (QAP-APP)–ból $\Gamma' \vdash ps \Leftrightarrow qt : T_2$.

$\square$

2.5.11. A fő tétel

Már csak egy dolog maradt hátra, nevezetesen, hogy belássuk, a definíciós ekvivalenciából következik a logikai. Az egyszerűbb szabályok, (Q-SYM), (Q-TRANS), (Q-ABS) és (T-ANS) most is, mint mindig indukcióval bizonyítható, de azért mondjuk ki őket:

2.5.12. Lemma (Logikai szimmetria)**.** *Ha* $\Gamma \vdash s|t : T$*, akkor* $\Gamma \vdash t|s : T$.

2.5.13. Lemma (Logikai tranzitivitás)**.** *Ha* $\Gamma \vdash s|t : T$*, és* $\Gamma \vdash t|u : T$*, akkor* $\Gamma \vdash s|u : T$.

2.5.14. Lemma (Logikai gyenge fej lezárás)**.** *Ha* $\Gamma \vdash s|t : T$*,* $s' \leadsto^* s$*, és* $t' \leadsto^* t$*, akkor* $\Gamma \vdash s'|t' : T$.

Ezzel az összes szabály esetére beláttuk a teljességet is, kivéve kettőt, (T-VAR) és (Q-BETA) eseteit. Ez előbbi kijönne rögtön a fő lemma második részéből is, de a (Q-BETA)–ra mutatott következő bizonyításból is kipottyan majd:

Az indukciós feltevést használva azt kell megmutatni, hogy

$$\Gamma \vdash (\lambda x : T_1.s_{12})s_2|t_{12}[x := t_2] : T_2$$

A 2.5.14.–es lemma szerint ehhez elég

$$\Gamma \vdash [x := s_2]s_{12}|t_{12}[x := t_2] : T_2$$

is. Az indukció viszont csak a következőket adja:

$$\Gamma, x : T_1 \vdash s_{12} \equiv t_{12} : T_2 \text{ és } \Gamma \vdash s_2 \equiv t_2 : T_1$$

Emiatt a bizonyítást be tudjuk fejezni, ha igazolni látjuk, hogy a logikai ekvivalencia zárt a behelyettesítésre nézve. Magából a definícióból ez elég nehéz volna, így egy kicsit tovább bonyolítjuk, kiterjesztjük a logikai ekvivalencia definícióját.

2.5.15. Definíció (Helyettesítés és hatása)**.** Egy függvény, amely változók halmazát a kifejezésekre képzi, helyettesítésnek nevezzük.
Legyen γ egy helyettesítés, hogy $dom(\gamma)$ tartalmazza t szabad változóit. Ekkor $\gamma(t)$ az a kifejezés, amit γ–ból és t–ből nyerünk, annak szabad változóinak szimultán helyettesítésével γ szerint.

2.5.16. Definíció (Helyettesítés kiterjesztése)**.** Tegyük fel, hogy $x \notin dom(\gamma)$, ekkor $\gamma[x := t]$–vel jelöljük azt a helyettesítést, aminek domainje $dom(\gamma) \cup x$, minden pontban γ–val megegyezik, és x–ben t–t veszi fel.

2.5.17. Definíció (Logikailag ekvivalens helyettesítések)**.** $\Gamma' \vdash \gamma|\delta : \Gamma$ pontosan akkor, ha $dom(\gamma) = dom(\delta) = dom(\Gamma)$ és minden $x : T \in \Gamma$–ra $\Gamma' \vdash \gamma(x)|\delta(x) : T$.

Mostmár bizonyíthatjuk a fő tételt, amit két külön részből teszünk össze:

2.5.18. Tétel (1. Fő tétel)**.** *Ha* $\Gamma \vdash t : T$ *és* $\Gamma' \vdash \gamma \,|\, \delta : T$*, akkor* $\Gamma' \vdash \gamma(t) \,|\, \delta(t) : T$

Bizonyítás. Indukcióval.

T-VAR $t = x$–ből és a logikailag ekvivalens helyettesítések definíciójából azonnal adódik.

T-ABS Legyen $t = \lambda x : T_1.t_2$, $T = T_1 \to T_2$. Feltehető, hogy $\Gamma'' \supseteq \Gamma'$ és $\Gamma'' \vdash s' \,|\, t' : T_1$. Azt kéne látni, hogy $\Gamma'' \vdash (\lambda x : T_1.\gamma(t_2))s' \,|\, (\lambda x : T_1.\delta(t_2))t' : T_2$. A logikai gyenge fej zártságból ehhez elegendő, hogy $\Gamma'' \vdash \gamma(t_2)[x := s'] \,|\, \delta(t_2)[x := t'] : T_2$. A logikai monotonitás szerint a Γ típusú γ és δ a Γ'' környezetben is logikai ekvivalenciában állnak, így tehát $\Gamma'' \vdash \gamma[x := s'] \,|\, \delta[x := t'] : (\Gamma, x : T_1)$. Az indukcióból innen adódik, hogy $\Gamma'' \vdash \gamma[x := s'](t_2) \,|\, \delta[x := t'](t_2) : T_2$, ami már ekvivalens a kívánt formulával.

T-APP Most $t = t_1t_2, T = T_{12}$.

Indukciós feltevésünk szerint $\Gamma' \vdash \gamma(t_1) \,|\, \delta(t_1) : T_1 \to T_2$ és $\Gamma' \vdash \gamma(t_2) \,|\, \delta(t_2) : T_1$. A logikai reláció definícióját alkalmazva, $\Gamma' \vdash \gamma(t_1)\gamma(t_2) \,|\, \delta(t_1)\delta(t_2) : T_2$ következik, azaz $\Gamma' \vdash \gamma(t_1t_2) \,|\, \delta(t_1t_2) : T_2$.

T-CONST $\Gamma' \vdash k \,|\, k : B$, így azonnal következik a tétel állítása is, hisz k–nak nincsenek szabad változói.

T-UNIT Azonnal következik a definícióból.

□

2.5.19. Tétel (2. Fő tétel)**.** *Ha* $\Gamma \vdash s \equiv t : T$*, és* $\Gamma' \vdash \gamma \,|\, \delta : \Gamma$*, akkor* $\Gamma' \vdash \gamma(s) \,|\, \delta(t) : T$.

Bizonyítás. Indukcióval.

Q-REF, Q-SYMM Azonnal következnek.

Q-TRANS A logikai szimmetriából és tranzitivitásból kapjuk, hogy $\Gamma' \vdash \delta \,|\, \delta : \Gamma$, így az indukciót γ–ra és δ–ra alkalmazva $\Gamma' \vdash \gamma(s) \,|\, \delta(t) : T$ adódik. Most δ–ra és δ–ra alkalmazva az indukciót, $\Gamma' \vdash \delta(t) \,|\, \delta(u) : T$ –t kapjuk, ahonnan a kellő állítás a logikai tranzitivitásból azonnal látszik.

Q-ABS Legyen $s = \lambda x : T_1.s_2$, $t = \lambda x : T_1.t_2$, és $T = T_1 \to T_2$. Tegyük fel, hogy $\Gamma'' \supseteq \Gamma'$ és $\Gamma'' \vdash s' \,|\, t' : T_1$. Azt szeretnénk megmutatni, hogy $\Gamma'' \vdash (\lambda x : T_1.\gamma(s_2))s' \,|\, (\lambda x : T_1.\delta(t_2))t' : T_2$. A logikai gyenge fej lezárás szerint elég látni $\Gamma'' \vdash \gamma(s_2)[x := s'] \,|\, \delta(t_2)[x := t'] : T_2$.

A logikai monotonitás szerint $\Gamma'' \vdash \gamma \,|\, \delta : \Gamma$, így $\Gamma'' \vdash \gamma[x := s'] \,|\, \delta[x := t'] : (\Gamma, x : T_1)$. Indukcióból $\Gamma'' \vdash \gamma[x := s'](s_2) \,|\, \delta[x := t'](t_2) : T_2$, ami már a kívánt állítással ekvivalens.

Q-APP Legyen $s = s_1s_2$, $t = t_1t_2$ és $T = T_{12}$. Indukcióból, a logikai reláció definíciójából és $\Gamma' \supseteq \Gamma$–t felhasználva $\Gamma' \vdash \gamma(s_1)\gamma(s_2) \,|\, \delta(t_1)\delta(t_2) : T_2$ adódik, amiből $\Gamma' \vdash \gamma(s_1s_2) \,|\, \delta(t_1t_2) : T_2$.

Q-EXT Itt a változók kiosztása a következő: $s = s$, $t = t$ és $T = T_1 \to T_2$. Tegyük fel, hogy $\Gamma'' \supseteq \Gamma'$ és $\Gamma'' \vdash s' \,|\, t' : T_1$. Azt kell megmutatni, hogy $\Gamma'' \vdash \gamma(s)s' \,|\, \delta(t)t' : T_2$.

Logikai monotonitásból $\Gamma'' \vdash \gamma \,|\, \delta : \Gamma$, így $\Gamma'' \vdash \gamma[x := s'] \,|\, \delta[x := t'] : (\Gamma, x : T_1)$. Indukcióból $\Gamma'' \vdash \gamma[x := s'](sx) \,|\, \delta[x := t'](tx) : (\Gamma, x : T_2)$, ami $\Gamma'' \vdash \gamma(s)s' \,|\, \delta(t)t' : T_2$, ami a kívánt állítás.

Q-BETA Legyen $s = (\lambda x : T_1.s_{12})s_2$, $t = t_{12}[x := t_2] : T_2$ és $T = T_2$. Indukcióból jön, hogy $\Gamma' \vdash \gamma[x := \gamma(s_2)] \,|\, \delta[x := \delta(t_2)] : (\Gamma, x : T_1)$. Innen $\Gamma' \vdash \gamma(s_{12})[x := \gamma(s_2)] \,|\, \delta(t_{12})[x := t_2] : T_2$. Végül a logikai gyenge fej lezárásból $\Gamma' \vdash (\lambda x : T_1.\gamma(s_{12}))\gamma(s_2) \,|\, \delta(t_{12}[x := t_2]) : T_2$, azaz $\Gamma' \vdash \gamma((\lambda x : T_1.\gamma(s_{12})s_2) \,|\, \delta(t_{12}[x := t_2]) : T_2$

Q-UNIT A logikai reláció definíciójából azonnal, hiszen itt $T = \texttt{Unit}$.

□

2.5.20. Tétel (Teljesség). *Ha* $\Gamma \vdash s \equiv t : T$, *akkor* $\Gamma \vdash s \Leftrightarrow t : T$.

Bizonyítás. Legyen γ az identikus behelyettesítés $\texttt{dom}(\Gamma)$–n. Minden $x : T$–re Γ–ból igaz, hogy $\Gamma \vdash x \,|\, x : T$ a fő lemma értelmében. Ezért $\Gamma \vdash \gamma \,|\, \gamma : \Gamma$ is fennáll. A fő tétel miatt $\Gamma \vdash \gamma(s) \,|\, \gamma(t) : T$, ami nem más mint $\Gamma \vdash s \,|\, t : T$. Ekkor ismét a fő lemma szerint $\Gamma \vdash s \Leftrightarrow t : T$. □

3. fejezet

Környezeti ekvivalencia

3.1. Bevezető

Az előbbiekben a vizsgált nyelv egyszerűsége miatt még algoritmust is tudtuk adni a típusos ekvivalencia vizsgálatára, igaz, az ráadásul definíciós ekvivalencia volt, bizonyos operációs szabályok felálllításával volt elkészítve, ráadásul úgy, hogy az nem mindig fedte az intuíciónkat. Például észszerű volna két függvényt ekvivalensnek tekinteni akkor is, ha ekvivalens kifejezésekre applikálva őket, szintén ekvivalens eredményeket kapunk. Ekkor azt is lehetne mondani, hogy a két függvény ugyanazt csinálja, ugyanazt a feladatot látja el. Az előző fejezet konstrukciójában ilyesmi nincs benne, és könnyen lehet ellenpéldát is mutatni. Tehát valamilyen más megközelítést kell alkalmazni, valamit aminek van naiv megfogalmazása, érthető és világos, mégis benne rejlik minden típusra illetve kifejezésre az ekvivalencia.

Gondoljunk most két, akár különböző nyelveken írt programra. Mikor mondanánk azt, hogy ekvivalensek, hogy ugyanazt csinálják? Kissé finomabb megfogalmazásban, akkor, ha ugyanazt a megfigyelhető eredményt kapjuk. Ez akár a definíciónk is lehetne, ha tisztáznánk, hogy mit jelentenek a program és a megfigyelés terminusok. Valójában is ezt fogjuk tenni, és maga a definíciónk is ez lesz. Programnak beillenek a zárt kifejezések – visszatérve a λ–kalkulusok világába – a megfigyelés pedig nem más, mint az, hogy bármely kontextusba belehelyezve a kifejezéseket, a kapott kifejezés „végrehajtása" terminál. Ezen fogalmak pedig a nyelv sajátjai, tehát teljesnek mondható a definíciónk. Természetesen mást is megfigyelhetnénk, például nem kötelező a terminálást. Ha azt várjuk el, hogy például mindkét kapott kifejezés eredménye 42(ha szám típusúak) legyen, az is épp olyan jó, általában ekvivalens fogalmat kapunk.

A következőkben a fő feladat a logikai relációk módszerének, illetve magának a logikai relációnak a továbbfejlesztése, kiterjesztése bonyolultabb nyelvek esetére. Természetesen mindezen elmélet, pontosabban a definíciók, nagyban függenek a nyelvtől amit éppen tekintünk. Érdemes inkább úgy gondolni a nyelvre, mint bizonyos tulajdonságokkal rendelkező domain–re, mert tulajdonképpen mindig annak egy jól meghatározott tulajdonságán vagy típusán fognak múlni a tételeink. Inkább a típusokról fogunk állítani valamit, és nem a nyelvről magáról.

Habár az imperatív programozásból ismert lokális változók illetve állapot(state) is leírható, sőt vizsgálható volna az alábbi elmélet segítségével a környezeti ekvivalenciára nézve, ezt nem fogjuk megtenni, szigorúan maradunk a tiszta funkcionális nyelvekre jellemző típuskonstrukciók vizsgálatánál, ellenben az sem lesz teljeskörű, például nem vizsgáljuk meg a monádokat, mint típuskonstruktort, bár hasonló szerkezetet tartalmaz majd a vizsgált nyelv.

3.2. A nyelv

A vizsgált nyelv Girard F rendszerére épül, azaz a polimorfikus λ-kalkulusra. Ezt bővítjük még ki rekurzívan definiált függvényekkel, rekord, lista és bázis típusokkal. A szemantika szigorú, azaz érték szerinti függvényhívást fogunk alkalmazni. Végeredményben az ML családba tartozó nyelvet kapunk és F_{ML} –nek hívjuk. Szintaktikája és típusozása a 3.1. táblázatban található meg.

A `Gnd` alaptípus alatt vagy `Bool` vagy `Int` típusokat értjük, így a c konstansok értékei $true$, $false$ vagy számok lehetnek. `op` alatt pedig valamilyen elemi operatort értünk, összeadás, szorzás, logikai és stb. További egyszerűsítés, hogy a λ–absztrakciót és a `fix` rekurzív operátort összevonásra került `fun` $x(x : T) = t : T$ formába. Tehát az így magadott függvényeket automatikusan rekurzívnak

Szintaxis

$t :=$	kifejezés
v	érték
$\texttt{if}\ v\ \texttt{then}\ t\ \texttt{else}\ t$	kondíció
$\texttt{op}(v_i^{i\in 1\ldots n})$	operáció
vv	applikáció
$v.l$	projekció
vT	típus applikáció
$\texttt{let}\ \{{}^*X, x\} = v\ \texttt{in}\ t$	kicsomagolás
$\texttt{let}\ x = t\ \texttt{in}\ t$	sorbafűzés
$\texttt{case}\ t\ \texttt{of}\ \{nil \Rightarrow t \mid x :: x \Rightarrow t\}$	listavizsgálat
$v :=$	értek
x	változó
$\texttt{fun}\ x(x : T) = t : T$	rekurzív függvény
$\{l_i = v_i^{i\in\ldots n}\}$	rekord érték
$\lambda X.v$	típus absztrakció
$\{{}^*T, v\}\ \texttt{as}\ \{\exists X, T\}$	csomag érték
nil_T	üres lista
$t :: t$	lista
$T :=$	típus
X	típus változó
$\texttt{Gnd}$	alaptípus
$T \to T$	függvénytípus
$\{l_i : T_i^{i\in\ldots n}\}$	rekord típus
$\forall X.T$	univerzális típus
$\{\exists X, T\}$	egzisztenciális típus
$list_T$	lista típus

Típusozás $\qquad \Gamma \vdash t : T$

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \quad \text{(T-VAR)}$$

$$\Gamma \vdash c : \texttt{typeof(c)} \quad \text{(T-CONST)}$$

$$\frac{\Gamma, f : T_1 \to T_2, x : T_1 \vdash t : T_2}{\Gamma \vdash \text{fun}\ f(x : T_1) = t : T_2 : T_1 \to T_2} \quad \text{(T-FUN)}$$

$$\frac{(\Gamma \vdash v_i : T)^{i\in 1\ldots n}}{\Gamma \vdash \{l_i = v_i^{i\in\ldots n}\} : \{l_i : T_i^{i\in\ldots n}\}} \quad \text{(T-RCD)}$$

$$\frac{\Gamma, X \vdash v : T \quad X \notin ftv(\Gamma)}{\Gamma \vdash \lambda X.v : \forall X.T} \quad \text{(T-TABS)}$$

$$\frac{\Gamma \vdash v_1 : T[X := T_1] \quad T' = \{\exists X, T\}}{\Gamma \vdash \{{}^*T_1, v_1\}\ \texttt{as}\ T' : T'} \quad \text{(T-PACK)}$$

$$\frac{\Gamma \vdash v : \texttt{Bool} \quad \Gamma \vdash t_1 : T \quad \Gamma \vdash t_2 : T}{\Gamma \vdash \texttt{if}\ v\ \texttt{then}\ t_1\ \texttt{else}\ t_2 : T} \quad \text{(T-IF)}$$

$$\begin{array}{l}\texttt{op} : \texttt{Gnd}_1, \ldots, \texttt{Gnd}_n \to \texttt{Gnd} \\ \dfrac{(\Gamma \vdash v_i : \texttt{Gnd}_i)^{i\in 1\ldots n}}{\Gamma \vdash \texttt{op}(v_i^{i\in 1\ldots n}) : \texttt{Gnd}}\end{array} \quad \text{(T-OP)}$$

$$\frac{\Gamma \vdash v_1 : T_1 \to T_2 \quad \Gamma \vdash v_2 : T_1}{\Gamma \vdash v_1 v_2 : T_2} \quad \text{(T-APP)}$$

$$\frac{\Gamma \vdash v : \{l_i : T_i^{i\in\ldots n}\}}{\Gamma \vdash v.l_j : T_j} \quad \text{(T-PROJ)}$$

$$\frac{\Gamma \vdash v : \forall X.T}{\Gamma \vdash vT_1 : T[X := T_1]} \quad \text{(T-TAPP)}$$

$$\frac{\begin{array}{l}\Gamma, X, x : T \vdash t : T_1 \\ X \notin \texttt{fvt}(\Gamma, T_1) \quad \Gamma \vdash v : \{\exists X, T\}\end{array}}{\Gamma \vdash \texttt{let}\ \{{}^*X, x\} = v\ \texttt{in}\ t : T_1} \quad \text{(T-UNPACK)}$$

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \texttt{let}\ x = t_1\ \texttt{in}\ t_2 : T_2} \quad \text{(T-SEQ)}$$

$$\Gamma \vdash nil_T : list_T \quad \text{(T-NIL)}$$

$$\frac{\Gamma \vdash t_1 : T \quad \Gamma \vdash t_2 : list_T}{\Gamma \vdash t_1 :: t_2 : list_T} \quad \text{(T-LIST)}$$

$$\frac{\Gamma \vdash t_1 : list_{T_1} \quad \Gamma \vdash t_2 : T_2 \quad \Gamma, t_3 : T_1, t_4 : list_{T_1} \vdash t_5 : T_2}{\Gamma \vdash \texttt{case}\ t_1\ \texttt{of}\ \{nil \Rightarrow t_2 \mid t_3 :: t_4 \Rightarrow t_5\} : T_2} \quad \text{(T-CASE)}$$

3.1. táblázat. F_{ML} szintaxis és típusozás tekintjük, és operációs szemantikája is ennek megfelelően van definiálva a 3.2. táblázatban.

Mivel főleg a környezeti ekvivalencia szempontjából szeretnénk a nyelvet vizsgálni, kissé speciálisan definiáltuk a szemantikát. Csak az elemi redukciót adtuk meg, illetve a termináció fogalmát. A keretek is pont ezt a célt szolgálják, a hajtómechanizmusa a terminációnak és a környezet egyfajta reprezentációja is egyben, a következő módon: A nyelv egy hasznos tulajdonsága, hogy az applikáció átfogalmazható:

$$t_1 t_2 = \texttt{let}\ x_1 = t_1\ \texttt{in}\ (\texttt{let}\ x_2 = t_2\ \texttt{in}\ x_1 x_2)$$

Emiatt egy tetszőleges kifejezés, vagy méginkább környezet is átfogalmazható egymásba ágyazott `let` konstrukciókká.

$$E[-] = \texttt{let}\ x_1 = (\ldots(\texttt{let}\ x_n = (-)\ \texttt{in}\ t_n)\ldots)\ \texttt{in}\ t_1$$

Az ennek megfelelő keretrendszer

$$S = Id \circ (x_1.t_1)\ldots(x_n.t_n)$$

ami $(x_i.t_i)$ elemi redukciós keretek sorozata. Ebből meg lehet mutatni, hogy az $E[t]$ a hagyományos(nagy–lépés) operációs szemantikában is egy értékre redukálódik pontosan akkor, ha $\langle S, t\rangle \downarrow$ fennáll.

Ugyancsak szükség lesz még a rekurzív függvények egy alaptulajdonságára, az unwinding–re. A tétel a szintaktikus analogonja annak a ténynek, hogy egy rekurzív függvény a legkisebb felső korlátja a véges approximációjának.

3.2.1. Tétel (Unwinding)**.** *Legyen F zárt, rekurzív függvény a következő formában:* $\texttt{fun}\ x(x : T_1) = u : T_2$. *Legyen*

$$F_0 = \texttt{fun}\ x(x : T_1) = (fx) : T_2$$

$$F_{n+1} = \texttt{fun}\ x(x : T_1) = u[f := F_n] : T_2$$

Ekkor minden t kifejezésre, melynek legföljebb f a szabad változója, $t[f := F] \downarrow$ pontosan akkor, ha $\exists n\, t[f := F_n] \downarrow$

Primitív redukció $t_1 \rightsquigarrow t_2$

$$\texttt{if } true \texttt{ then } t_1 \texttt{ else } t_2 \rightsquigarrow t_1 \qquad \text{(R-IFTRUE)}$$

$$\texttt{if } false \texttt{ then } t_1 \texttt{ else } t_2 \rightsquigarrow t_2 \qquad \text{(R-IFFALSE)}$$

$$\frac{\texttt{op}(c_i^{i\in 1\ldots n}) \texttt{ értéke } c}{\texttt{op}(c_i^{i\in 1\ldots n}) \rightsquigarrow c} \qquad \text{(R-OP)}$$

$$\frac{v_1 = \texttt{fun } x(x : T_1) = t : T_2}{v_1 v_2 \rightsquigarrow t[f := v_1][x := v_2]} \qquad \text{(R-APPABS)}$$

$$\{l_i = v_i^{i\in\ldots n}\} \rightsquigarrow v_j \qquad \text{(R-PROJRCD)}$$

$$(\lambda X.v)T \rightsquigarrow v[X := T] \qquad \text{(R-TAPPTABS)}$$

$$\frac{v = \{{}^*T, v\} \texttt{ as } \{\exists X, T\}}{\texttt{let } \{{}^*X, x\} = v \texttt{ in } t} \qquad \text{(R-PACK)}$$
$$\rightsquigarrow t[X := T_1][x := v_1]$$

$$\texttt{case } nil_T \texttt{ of } \{nil \Rightarrow t_1 | h :: t \Rightarrow t_2\} \qquad \text{(R-CASENIL)}$$
$$\rightsquigarrow t_1$$

$$\texttt{case } t_1 :: t_2 \texttt{ of } \{nil \Rightarrow t_3 | h :: t \Rightarrow t_4\} \qquad \text{(R-CASE)}$$
$$\rightsquigarrow t_4[t_1 := h][t_2 := t]$$

Keret rendszer szintaxis

$S :=$	keret rendszer
Id	üres keret
$S \circ (x.t)$	keret összefűzés

Keret rendszer típusozása $\Gamma \vdash S : T_1 \multimap T_2$

$$\Gamma \vdash Id : T \multimap T \qquad \text{(S-NIL)}$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2 \quad \Gamma \vdash S : T_2 \multimap T_3}{\Gamma \vdash S \circ (x.t) : T_1 \multimap T_3} \qquad \text{(S-CONS)}$$

Termináció $\langle S, t\rangle \downarrow$ és $t \downarrow$

$$\langle Id, v\rangle \downarrow \qquad \text{(S-NILVAL)}$$

$$\frac{\langle S, t[x := v]\rangle \downarrow}{\langle S \circ (x.t), v\rangle \downarrow} \qquad \text{(S-CONSVAL)}$$

$$\frac{\langle S \circ (x.t_2), t_1\rangle \downarrow}{\langle S, \texttt{let } x = t_1 \texttt{ in } t_2\rangle \downarrow} \qquad \text{(S-SEQ)}$$

$$\frac{t_1 := t_2 \quad \langle S, t_2\rangle \downarrow}{\langle S, t_1\rangle \downarrow} \qquad \text{(S-RED)}$$

$$\frac{\langle Id, t\rangle \downarrow}{t \downarrow} \qquad \text{(TERM)}$$

3.2. táblázat. F_{ML} operációs szemantikája

Innentől kezdve bármilyen felmerülő kifejezéstől megköveteljük a jóltípusozottságot. Ha bármilyen nem jól típusozott kifejezésről esne szó, azt külön kiemeljük.

3.3. A környezeti ekvivalencia

Érzékelhető, hogy a környezeti ekvivalencia jelenlegi, környezetekkel megfogalmazott definícióját nehéz a gyakorlatban használni, hiszen iterál az összes lehetséges környezeten. Gordon [3] és Lassen [4] nyomán egy sokkal inkább használhatóbb, szabályközeli eljárás is kívnálkozik az ekvivalencia definiálására, mely jobban összpontosít az ekvivalencia fő tulajdonságára, azaz hogy kongruencia és adekvát a termináció megfigyelésével.

3.3.1. Definíció. F_{ML} kifejezései közötti R reláció típushű, ha (Γ, t, t', T) négyesekből áll és minden ilyenre igaz, hogy $\Gamma \vdash t : T$ és $\Gamma \vdash t' : T$. Egy ilyen reláció fennállását $\Gamma \vdash tRt' : T$ vel jelöljük. $\Gamma = \emptyset$ esetén a $tRt' : T$ rövidítés is ezt jelöli. Továbbá egy ilyen reláció

- ekvivalencia, ha reflexív, szimmetrikus és tranzitív,
- kongruencia, ha ekvivalencia és rendelkezik a behelyettesíthetőségi és kompatibilitási tulajdonságokkal, azaz zárt az ezeket definiáló szabályokra, továbbá
- adekvát(a $\downarrow$–ra nézve) ha $\emptyset \vdash tRt' : T$ akkor $t \downarrow$ pontosan akkor, ha $t' \downarrow$.

A fenti tulajdonságok szabályai a 3.3. táblázatban találhatóak meg.

3.3.2. Definíció.

- Az identikus reláció, $Id = \{(\Gamma, t, t, T) | \Gamma \vdash t : T\}$
- R reciproka, $R^{op} = \{(\Gamma, t', t, T) | \Gamma \vdash tRt' : T\}$
- R_1 és R_2 kompozíciója, $R_1 \circ R_2 = \{(\Gamma, t, t'', T) | \exists t' \Gamma \vdash tR_1t' : T \wedge \Gamma \vdash t'R_2t'' : T\}$

Ekvivalencia

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash tRt : T}$$

$$\frac{\Gamma \vdash tRt' : T}{\Gamma \vdash t'Rt : T}$$

$$\frac{\Gamma \vdash tRt' : T \quad \Gamma \vdash t'Rt'' : T}{\Gamma \vdash tRt'' : T}$$

Behelyettesíthetőség

$$\frac{\Gamma \vdash vRv' : T \quad \Gamma, x : T_1 \vdash tRt' : T_2}{\Gamma \vdash t[x := v]Rt'[x := v'] : T_2}$$

$$\frac{\Gamma, X \vdash tRt' : T}{\Gamma \vdash t[X := T_1]Rt'[X := T_1] : T[X := T_1]}$$

Kompatibilitás

$$\frac{(x : T) \in \Gamma}{\Gamma \vdash xRx : T}$$

$$\Gamma \vdash cRc : \mathtt{typeof}(c)$$

$$\frac{\Gamma, f : T_1 \to T_2, x : T_1 \vdash tRt' : T_2}{\Gamma \vdash \mathtt{fun}\ f(x : T_1) = t : T_2 R \ \mathtt{fun}\ f(x : T_1) = t' : T_2 : T_1 \to T_2}$$

$$\frac{(\Gamma \vdash v_i R v_i' : T_i)^{i \in 1 \ldots n}}{\Gamma \vdash \{l_i = v_i^{i \in 1 \ldots n}\} R \{l_i = v_i'^{i \in 1 \ldots n}\} : \{l_i : T_i^{i \in 1 \ldots n}\}}$$

$$\frac{\Gamma, X \vdash vRv' : T \quad X \notin ftv(\Gamma)}{\Gamma \vdash \lambda X.vR\lambda X.v' : \forall X.T}$$

$$\frac{\Gamma \vdash v_1 R v_1' : T[X := T_1]}{\Gamma \vdash \{{}^*T_1, v_1\}\ \mathtt{as}\ \{\exists X, T\} R \ \{{}^*T_1, v_1'\}\ \mathtt{as}\ \{\exists X, T\} : \{\exists X, T\}}$$

$$\frac{\Gamma \vdash vRv' : \mathtt{Bool} \quad \Gamma \vdash t_1 R t_1' : T \quad \Gamma \vdash t_2 R t_2' : T}{\Gamma \vdash \mathtt{if}\ v\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2 R \ \mathtt{if}\ v'\ \mathtt{then}\ t_1'\ \mathtt{else}\ t_2'}$$

$$\frac{(op) : \mathtt{Gnd}_1, \ldots, \mathtt{Gnd}_n \to \mathtt{Gnd} \quad (\Gamma \vdash v_i R v_i' : \mathtt{Gnd}_i)^{i \in 1 \ldots n}}{\Gamma \vdash v_i^{i \in 1 \ldots n} R v_i'^{i \in 1 \ldots n} : \mathtt{Gnd}}$$

$$\frac{\Gamma \vdash v_1 R v_1' : T_1 \to T_2 \quad \Gamma \vdash v_2 R v_2' : T_1}{\Gamma \vdash v_1 v_2 R v_1' v_2' : T_2}$$

$$\frac{\Gamma \vdash vRv' : \{l_i : T_i^{i \in 1 \ldots n}\}}{\Gamma \vdash v.l_j Rv.'.l_j : T_j}$$

$$\frac{\Gamma \vdash vRv' : \forall X.T}{\Gamma \vdash vT_1 Rv'T_1 : T[X := T_1]}$$

$$\frac{\Gamma, X, x : T \vdash tRt' : T_1 \quad X \notin ftv(\Gamma, T_1) \quad \Gamma \vdash vRv' : \{\exists X, T\}}{\Gamma \vdash \mathtt{let}\ \{{}^*X, x\} = v\ \mathtt{in}\ tR \ \mathtt{let}\ \{{}^*X, x\} = v'\ \mathtt{in}\ t' : T_1}$$

$$\frac{\Gamma \vdash t_1 R t_1' : T_1 \quad \Gamma, x : T_1 \vdash t_2 R t_2' : T_2}{\Gamma \vdash \mathtt{let}\ x = t_1\ \mathtt{in}\ t_2 R \mathtt{let}\ x = t_1'\ \mathtt{in}\ t_2' : T_2}$$

$$\Gamma \vdash nil_T R nil_T : list_T$$

$$\frac{\Gamma \vdash hRh' : T \quad \Gamma \vdash tRt' : list_T}{\Gamma \vdash (h :: t)Rh' :: t' : list_T}$$

$$\frac{\Gamma \vdash lRl' : list_{T_1} \quad \Gamma \vdash t_1 R t_1' : T_2 \quad \Gamma, x : T_1, t : list_{T_1} \vdash t_2 R t_2' : T_2}{\Gamma \vdash (\mathtt{case}\ l\ \mathtt{of}\ \{nil \Rightarrow t | t_1 :: t \Rightarrow t_2\}) R (\mathtt{case}\ l'\ \mathtt{of}\ \{nil \Rightarrow t_1' | h :: t \Rightarrow t_2'\}) : T_2}$$

3.3. táblázat. F_{ML} szintaxis és típusozás

- R tranzitív lezárása, $R^+ = \bigcup_{i \in \mathbb{N}} R_i$, ahol $R_0 = R$ és $R_{i+1} = R \circ R_i$
- R nyílt kiterjesztését, R° a következőféleképpen definiáljuk: Legyen $\Gamma = X_1, \ldots, X_m, x_1 : T_1, \ldots, x_n : T_n$, ekkor egy Γ–zárt helyettesítés egy olyan leképezés, mely az X_i változókat zárt T_i típusokra, míg az x_j változókat megfelelő típusú zárt v_j értékekre képezi le. Ekkor R° mindazokat a (Γ, t, t', T) elemeket tartalmazza, melyekre minden σ Γ–zárt behelyettesítésre $\emptyset \vdash \sigma(t) R \sigma(t') : \sigma(T)$

3.3.3. Tétel (F_{ML} környezeti ekvivalenciája, $=_{ctx}$)**.** *F_{ML} kifejezésein létezik legnagyobb kongruencia ami adekvát.*

Ezzel a naivabb környezeti ekvivalencia definíciót sikerült átalakítani a $=_{ctx}$ relációra, ami már sokkal technikai jellegűbb, de sajnos még ezzel is elég nehéz dolgozni. Ha be kéne bizonyítani egy ekvivalenciát, bajba kerülhetnénk. Egy újabb jellemzést adott Mason és Talcott [5]:

3.3.4. Definíció ($=_{ciu}$)**.** Legyen $=_{ciu}$ R°, ahol
$R = \{(\emptyset, t, t', T) | \emptyset \vdash t : T,\ \emptyset \vdash t' : T,\ \forall S. \langle S, t \rangle \downarrow \Leftrightarrow \langle S, t' \rangle \downarrow\}$ Azaz két zárt kifejezés ciu–ekvivalens, ha ugyanazt a terminációs viselkedést mutatják, bármilyen S keretrendszerrel is párosítjuk őket.

3.3.5. Lemma. *Valamely S keretrendszerre és t kifejezésre, definiáljuk $S[T]$–t a következő indukcióval:*

$$Id[t] = t$$
$$S \circ (x.t')[t] = S[\mathtt{let}\ \ x = t\ \ \mathtt{in}\ \ t']$$

Ekkor $\langle S, t \rangle \downarrow$ pontosan akkor, ha $S[t] \downarrow$, azaz $\langle Id, S[t] \rangle \downarrow$

3.3.6. Tétel (CIU tétel)**.** *A környezeti ekvivalencia és a ciu–ekvivalencia megegyezik.*

Azt viszonylag egyszerű bizonyítani, hogy amik környezetileg ekvivalensek, azok CIU–ekvivalenciában állnak, ellenben sokkal nehezebb a másik irány, amihez csakúgy mint az előző fejezetben, a logikai relációk módszerét fogjuk felhasználni.

3.4. A logikai reláció

A továbbiakban újabb, mégjobban használható jellemzést adunk a környezeti ekvivalenciára, logikai reláció segítségével. Több fontos eredmény is ebből egyszerűen bizonyítható majd, például az ekvivalencia kiterjesztési, extenzionális jellemzései.

3.4.1. Definíció. Legyen Typ F_{ML} típusainak halmaza. Ekkor adott $T \in Typ$–ra

- $\texttt{Term}(T)$ legyen a T típusú zárt kifejezések halmaza,
- $\texttt{Val}(T)$ $\texttt{Term}(T)$ azon részhalmaza, mely az értékeket tartalmazza,
- $\texttt{Stack}(T)$ zárt keretrendszerek halmaza, melyre $\exists T' \in Typ : \emptyset \vdash S : T \multimap T'$.

Ha $T' \in Typ$ is adott típus, akkor,

- $\texttt{TRel}(T, T')$ a kifejezés relációk halmaza, azaz $P(\texttt{Term}(T) \times \texttt{Term}(T'))$,
- $\texttt{VRel}(T) = P(\texttt{Val}(T) \times \texttt{Val}(T'))$, az érték relációk,
- $\texttt{SRel}(T) = P(\texttt{Stack}(T) \times \texttt{Stack}(T'))$, a keret relációk halmaza.

3.4.2. Definíció. Legyen $T, T' \in Typ$ zárt típusok, $r \in \texttt{TRel}(T, T')$. Ekkor $r^v = \{(v, v') \in \texttt{Val}(T) \times \texttt{Val}(T') | (v, v') \in r\}$, az r szűkítése értékekre. Jelölje továbbá $r^s \in \texttt{SRel}(T, T')$ azon keretrendszer párokat, melyekre az r relációban lévő kifejezéseket párosítva, ugyanaz a terminációs viselkedés, tehát

$$r^s = \{S, S') | \forall (t, t') \in r \langle S, t \rangle \downarrow \Leftrightarrow \langle S', t' \rangle \downarrow \}$$

Megfordítva, $s \in \texttt{SRel}(T, T')$ esetén

$$s^t = \{t, t') | \forall (S, S') \in s \langle S, t \rangle \downarrow \Leftrightarrow \langle S', t' \rangle \downarrow \} \in \texttt{TRel}(T, T')$$

Egy r relációt zártnak nevezünk, ha $r = r^{st}$ és értékesnek, ha $r = r^{vst}$

3.4.3. Lemma. *Az ^{s} és ^{t} operátorok Galois kapcsolatot definiálnak a keret és kifejezés relációk között.*

3.4.4. Lemma. *Zárt, $r \in \mathtt{TRel}(T, T')$ relációra teljsülnek az alábbi tulajdonságok:*

Ekvivalenciahűség: $(t, t') \in r,\ \emptyset \vdash t =_{ciu} t_1 : T,\ \emptyset \vdash t' =_{ciu} t'_1 : T \Rightarrow (t_1, t'_1) \in r$

Megengedhetőség rekurzív típusra: *Legyen $F = \mathtt{fun}\ \ f(x : T_1) = u : T_2$ és $F' = \mathtt{fun}\ \ f(x : T_1) = u' : T_2$ két rekurzív függvény, F_n és F'_n pedig az approximációjuk. Ha $\forall n\ (t[x := F_n], t'[x := F'_n]) \in r$, akkor $(t[x := F], t'[x := F']) \in r$*

3.4.5. Definíció (Típus hatása kifejezés reláción)**.** Legyen $T(\overline{X})$ típus adott, melynek szabad változói az $\overline{X} = X_1, \ldots, X_n$ között vannak. A típusnak megfelelően adott továbbá $r_1 \in \mathtt{TRel}(T_1, T'_1), \ldots, r_n \in \mathtt{TRel}(T_n, T'_n)$. Ekkor T hatása r_i relációkon az a $T[\overline{r}] \in \mathtt{TRel}(T[\overline{X} := \overline{T}], T[\overline{X} := \overline{T'}])$, melyet a következő T struktúrája szerinti indukció definiál:

$$
\begin{aligned}
X_i[\overline{r}] &= (t_i)^{vst} \\
\mathtt{Gnd}[\overline{r}] &= (Id_{\mathtt{Gnd}})^{st} \\
(T_1 \to T_2)[\overline{r}] &= \mathtt{fun}(T_1[\overline{r}], T_2[\overline{r}])^{st} \\
\{l_i : T_i^{i \in 1 \ldots n}\}[\overline{r}] &= \{l_i : T_i[\overline{r}]^{i \in 1 \ldots n}\}^{st} \\
(\forall X.T)[\overline{r}] &= (\lambda r.T[r, \overline{r}])^{st} \\
\{\exists X, T\}[\overline{r}] &= \{\exists r, T[r, \overline{r}]\}^{st} \\
list_T[\overline{r}] &= (\overline{r} \times list_T[\overline{r}])^{+st}
\end{aligned}
$$

ahol

$Id_{\mathtt{Gnd}} = \{(c, c) \mid typeof(c) = \mathtt{Gnd}\} \in \mathtt{VRel}(\mathtt{Gnd}, \mathtt{Gnd})$

$fun(r_1, r_2) = \{(v, v') \mid \forall (v_1, v'_1) \in (r_1)^v\ (vv_1, v'v'_1) \in r_2\}$
$\in \mathtt{VRel}(T_1 \to T_2, T'_1 \to T'_2)$ ahol $r_1 \in \mathtt{TRel}(T_1, T'_1)$ és $r_2 \in \mathtt{TRel}(T_2, T'_2)$

$\{l_i = r_i^{i\in 1\ldots n}\} = \{(v, v') \mid (v.l_i, v'.l_i) \in r_i \quad i \in 1\ldots n\}$
$\in \mathtt{VRel}(\{l_i : T_i^{i\in 1\ldots n}\}, \{l_i : T_i'^{i\in 1\ldots n}\})$ ahol $r_i \in \mathtt{TRel}(T_i, T_i')^{i\in 1\ldots n}$

$\lambda r.R(r) = \{(v, v') \mid \forall T_1, T_1' \in Typ\ \forall r \in \mathtt{VRel}(T_1, T_1')\ (vT_1, v'T_1') \in R(r)\} \in \mathtt{VRel}(\forall X.T, \forall X.T')$ ahol $R(r) \in \mathtt{VRel}(T[X := T_1], T'[X := T_1'])$, $r \in \mathtt{TRel}(T_1, T_1')$ adottak.

$\{\exists r, R(r)\} = \{(v, v') \mid \exists T_1, T_1' \in Typ\ \exists r \in \mathtt{TRel}(T_1, T_1')\ \exists (v_1, v_1') \in R(r)$
$v = \{{}^*T_1, v_1\}\ \mathtt{as}\ \{\exists X, T\} \wedge v = \{{}^*T_1', v_1'\}\ \mathtt{as}\ \{\exists X, T'\}\} \in \mathtt{VRel}(\{\exists X, T\}, \{\exists X, T'\})$ ahol $R(r) \in \mathtt{VRel}(T[X := T_1], T'[X := T_1'])$, $r \in \mathtt{TRel}(T_1, T_1')$ adottak.

$(\overline{r} \times list_T[\overline{r}])^+ = \mu x(\overline{r} \times x)^+$, ahol a μ operátor a típusrekurzió operátornak adaptációja kifejezés-relációkra. Egészen pontosan, ha $T, T' \in Typ$, $r_1 \in \mathtt{TRel}(T, T')$ és $r_2 \in \mathtt{TRel}(list_T, list_{T'})$ adottak, akkor $(r_1 \times r_2)^+ \in \mathtt{TRel}(list_T, list_{T'})$,

$$(r_1 \times r_2)^+ = \{(nil_T, nil_{T'})\} \cup \{(h :: t, h' :: t') \mid (h, h') \in r_1\ (t, t') \in r_2\}$$

Ekkor minden r_1–re, az $r_2 := (r_1 \times r_2)^{+st}$ monoton a $\mathtt{TRel}(list_T, list_{T'})$ teljes hálóban és így vehetjük a legnagyobb fixpontját. Ez a fixpont pont a definiálni kívánt hatás eredménye.

A definíció akkor is érvényben marad, ha T zárt, azaz $n = 0$. Hamarosan látni fogjuk, hogy $T[\,]$ nem más mint a környezeti ekvivalencia a T típuson, tehát

$$v =_{ctx} v' : T \Leftrightarrow (v, v') \in T[\,]$$

Most ezt a konstrukciót, a típusok hatását használjuk arra, hogy szabad változókat is tartalmazó kifejezéseken is definiáljuk olyan típushű relációt, melyre igaz, hogyha relációban lévő kifejezéseket helyettesítünk a kifejezések szabad változóiba, akkor relációban lévő kifejezéseket kapunk. Az ilyen relációkat nevezzük logikai relációknak Plotkin óta, mint amilyen a most következő Δ:

3.4.6. Definíció (A Δ logikai reláció)**.** Legyen $\Gamma \vdash t : T$ és $\Gamma \vdash t' : T$, ahol $\Gamma = X_1, \ldots, X_m, x_1 : T_1, \ldots, x_n : T_n$. Ekkor $\Gamma \vdash t\Delta t' : T$ jelentse azt, hogy minden

Γ–zárt σ, σ' behelyettesítésre és minden $\overline{r} = (r_i \in \texttt{TRel}(\sigma(X_i), \sigma'(X_i))^{i \in 1...n})$ reláció–halmazra, ha $\forall j \in 1 \dots n$ $(\sigma(x_j), \sigma'(x_j)) \in T_j[\overline{r}]^v$ akkor $(\sigma(t), \sigma'(t')) \in T[\overline{r}]$.

Tehát maga a konstrukció alapötlete egyszerű és független a program ekvivalenciától, mégis ami megadja Δ legfőbb tulajdonságát, hogy megegyezik a környezeti ekvivalenciával, az a típus hatásának körülményes definíciója, ami majd az elkövetkező kiterjesztési és parametrizációs tulajdonságokat hivatott biztosítani. Most nem is látszik igazán miért is hasznos ez a megközelítés, de látni fogunk egy–két konkrét példát, bizonyítást, ami egyébként rendkívül nehéz lenne és megvilágítja a típushatás értelmét.

Egypár egyéb tulajdonsága a hatásnak:

3.4.7. Lemma. *A* $\mathtt{fun}(-,-)$ *operációra,*

$$\mathtt{fun}(r_1, (r_2)^{st})^{stv} = \mathtt{fun}(r_1, (r_2)^{st})$$
$$\mathtt{fun}((r_1)^{vst}, (r_2)^{st}) = \mathtt{fun}(r_1, (r_2)^{st})$$

3.4.8. Lemma. *Fennállnak a következő egyenlőségek:*

$$\{l_i = (r_i)^{st\ i \in 1...n}\}^{stv} = \{l_i = (r_i)^{st}\ i \in 1 \dots n\} \quad (3.1)$$
$$(\lambda r.R(r)^{st})^{stv} = \lambda r.R(r)^{st} \quad (3.2)$$

3.4.9. Lemma. *Legyen* $T, T' \in Typ$*, hogy* $ftv(T) \subseteq X, \overline{X}$ *és* $ftv(T') \subseteq \overline{X}$*. Ekkor* $(T[X := T'])[\overline{r}] = T[T'[\overline{r}], \overline{r}]$*.*

3.4.10. Lemma. *Minden* $T[\overline{r}]$ *relációra* $T[\overline{r}]^{vst}$ *tehát zártak is.*

Innentől a célunk, hogy belássuk a definiált ekvivalencia relációk egyezőségét.

3.4.11. Lemma (Δ Fő lemma)**.** *A behelyettesítési és kompatibilitási tulajdonságokat* Δ *teljesíti.*

Bizonyítás. Az első behelyettesítési tulajdonság rögtön következik Δ definíciójából, ahogy helyettesítésekkel volt definiálva a reláció.

A második tulajdonság a 3.4.9. egyeneságú következménye.

A kompatibilitást úgy fogjuk belátni, hogy sorravesszük az egyes nyelvi elemeket, és azokra látjuk be a kellő tulajdonságot.

Változók: Azonnal adódik Δ definíciójából.

Konstansok: Azt kell látnunk, hogy $(c, c) \in \texttt{Gnd}[\overline{r}]$, ami egyenlő $(Id_{\texttt{Gnd}})^{st}$. $Id_{\texttt{Gnd}}$ definíciója szerint $(c, c) \in Id_{\texttt{Gnd}}$ és mivel $Id_{\texttt{Gnd}} \subseteq (Id_{\texttt{Gnd}})^{st}$, készen vagyunk.

Rekurzív függvények Megmutatjuk, hogy ha $F = \texttt{fun}\ \ f(x : T_1) = t : T_2 \in \texttt{Val}(T_1 \to T_2)$ és $F' = \texttt{fun}\ \ f(x : T_1') = t' : T_2 \in \texttt{Val}(T_1' \to T_2')$ függvényértékek, r_1, r_2 relációk, r_2 zárt és $\forall(v, v') \in r_1 \to r_2$ és $(v_1, v_1') \in r_1^v$ $(t[x := v_1][f := v], t[x := v_1'][f := v']) \in r_2$, akkor $(F, F') \in r_1 \to r_2$

Ehhez elég ha minden $n \in 0, 1, \ldots$ indexre belátjuk, hogy a függvények unwindelt változata teljesíti, ugyanis ekkor a megengedhetőség miatt $(F, F') \in r_1 \to r_2$ is teljesül. Index szerinti indukciót alkalmazunk:

$n = 0$ esetén definíció szerint $\langle S, F_0 v_1\rangle \downarrow$ minden S keretrendszerre és v_1 értékre. Hasonlóan F_0'–re is teljsül ugyanez. Minden s keret relációra és $(v_1, v_1') \in r_1^v$ re $(F_0 v_1, F_0' v_1') \in s^t$, így $s = r_2^s$–re méginkább. Azaz $(F_0 v_1, F_0' v_1') \in r_2$, azaz $(F_0, F_0') \in r_1 \to r_2$

Általános esetben, F_{n+1} definíciója miatt $F_{n+1} v_1 =_{ctx} t[x := v_1][f := F_n]$ és hasonlóan F_{n+1}'–re. Mivel r_2 zárt, használhatjuk a ekvivalencia–hűséget: $(F_{n+1} v_1, F_{n+1}' v_1') \in r_2$. Ez minden értékpárra fennáll, így készen vagyunk.

Rekord értékek: A 3.4.5. definíciójában használt jelöléssel, azt kell bebizonyítani, hogy ha $(v_i, v_i') \in r_i$, akkor $(v, v') \in \{l_i = r_i^{iin1\ldots n}\}$ szintén teljesül. Elég: $(v.l_i, v'.l_i) \in r_i$, de mivel zárt a reláció, elég ha r_I^{st}–ben benne vannak. Ez ekvivalens azzal, hogy $\langle S, v.l_i\rangle \downarrow \Leftrightarrow \langle S', v'.l_i\rangle \downarrow$ minden $(S, S') \in r_i^s$–re. v definíciója szerint $\langle S, v.l_i\rangle \downarrow \Leftrightarrow \langle S, v_i\rangle \downarrow$ és ez hasonlóan v'–re is. Feltétel szerint $(v_i, v_i') \in r_i$ és $(S, S') \in r_i^s$, így $\langle S, v_i\rangle \downarrow \Leftrightarrow \langle S', v_i'\rangle \downarrow$, ami kellett.

Típus absztrakciók: Most is a hatás definíciójában lévő jelöléseket használva, $(\lambda X.v, \lambda X.v') \in \lambda r.R(r)$ a cél, azaz tetszőleges T_1 és T_1' típusokkal $((\lambda X.v)T, (\lambda X.v')T') \in R(r)$, ahol $r \in \mathtt{TRel}(T_1, T_1')$. Feltétel szerint $(v[X := T_1], v[X := T_1] \in R(r)$, így $(S, S') \in R(r)^s$–el $\langle S, v[X := T_1]\rangle \downarrow \Leftrightarrow \langle S, v'[X := T_1']\rangle \downarrow]$. Ebből $\langle S, (\lambda X.v)T\rangle \downarrow \Leftrightarrow \langle S, v[X := T_1]\rangle \downarrow$, amiből $\langle S, (\lambda X.v)T\rangle \downarrow \Leftrightarrow \langle S', (\lambda X.v')T'\rangle \downarrow$, ami ugyanaz, mint a bizonyítandó $((\lambda X.v)T, (\lambda X.v')T') \in R)(r)^{st}$.

Csomagok: Ez az eset egyszerűen következik a 3.4.9. lemmából és a hatás definíciójából.

Feltételes utasítás: Egyszerűen látható, hogy $(S \circ (x\ \mathtt{if}\ x\ \mathtt{then}\ t_1\ \text{else}\ t_2), S' \circ (x\ \mathtt{if}\ x\ \mathtt{then}\ t_1'\ \mathtt{else}\ t_2')) \in (Id_{\mathtt{Bool}})^s$. Innen
$\langle S \circ (x\ \mathtt{if}\ x\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2), v\rangle \downarrow \Leftrightarrow$
$\langle S' \circ (x\ \mathtt{if}\ x\ \mathtt{then}\ t_1'\ \mathtt{else}\ t_2'), v'\rangle \downarrow$, ami ekvivalens a következővel:
$\langle S,\ \mathtt{if}\ v\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2\rangle \downarrow \Leftrightarrow \langle S',\ \mathtt{if}\ v'\ \mathtt{then}\ t_1'\ \mathtt{else}\ t_2'\rangle \downarrow$
minden $(S, S') \in r^s$, ami a kívánt eredményt adja.

Elemi operációk: Egyszerűen bizonyítható, hogy minden Gnd típusra $(Id_{\mathtt{Gnd}})^{stv} = Id_{\mathtt{Gnd}}$, így elég látni, hogy minden op operátorra és megfelelő típusú c_i konstansokra $(\mathtt{op}(c_i^{i\in 1\ldots n}), \mathtt{op}(c_i^{i\in 1\ldots n})) \in (Id_{\mathtt{Gnd}})^{st}$. Ha $\mathtt{op}(c_i^{i\in 1\ldots n}) = c$, akkor minden $(S, S') \in (Id_{\mathtt{Gnd'}})^s$ esetén $\langle S, \mathtt{op}(c_i^{i\in 1\ldots n})\rangle \downarrow \Leftrightarrow \langle S, c\rangle \downarrow \Leftrightarrow \langle S', c\rangle \downarrow \Leftrightarrow \langle S', \mathtt{op}(c_i^{i\in 1\ldots n})\rangle \downarrow$. Azaz $(\mathtt{op}(c_i^{i\in 1\ldots n}), \mathtt{op}(c_i^{i\in 1\ldots n})) \in (Id_{\mathtt{Gnd}})^{st}$.

Applikáció: Ez az eset a 3.4.7. lemma első részéből következik:
$(v, v') \in \mathtt{fun}(r_1, r_2)^{stv} = \mathtt{fun}(r_1, (r_2)^{st})^{stv} = \mathtt{fun}(r_1, (r_2)^{st}) = \mathtt{fun}(r_1, r_2)$.

Projekciók és típusapplikációk: Hasonlóan az előzőhöz, csak a 3.4.8. lemmát kell használni.

Kicsomagolás: Minden $(S, S') \in r_2^s$–re és $\{\exists r_1, R(r_1)\}$ definiciójából, $(S \circ (y\ \mathtt{let}\ \{^*X, x\} = y\ \mathtt{in}\ t), S' \circ (y\ \mathtt{let}\ \{^*X, x\} = y\ \mathtt{in}\ t')) \in \{\exists r_1, R(r_1)\}^s$. Így ha $(v, v') \in \{\exists r_1, R(r_1)\}^{stv} \subseteq (\{\exists r_1, R(r_1)\}^s)^t$, akkor $\langle S \circ (y\ \mathtt{let}\ \{^*X, x\} = y\ \mathtt{in}\ t), v\rangle \downarrow \Leftrightarrow \langle S' \circ (y\ \mathtt{let}\ \{^*X, x\} =$

$y \texttt{ in } t'), v'\rangle \downarrow$, és így $\langle S, \texttt{let } \{^*X, x\} = v \texttt{ in } t\rangle \downarrow \Leftrightarrow \langle S', \texttt{ let } \{^*X, x\} = v' \texttt{ in } t')\rangle \downarrow$. Mivel ez minden $(S, S') \in r_2^s$–re igaz, tudunk a következtetni a következőre: $(\texttt{ let } \{^*X, x\} = v \texttt{ in } t, \texttt{ let } \{^*X, x\} = v' \texttt{ in } t') \in r_2^{st} = r_2$

Sorbafűzés: Minden $(S, S') \in r_2^s$–re a t, t'–re tett feltételek miatt $(S \circ (xt_2), S' \circ (st_2')) \in r_1^{vs}$. Mivel $r_1^{vst} = r_1$, ha $(t_1, t_1') \in r_1$, akkor $\langle S \circ (xt_2), t_1\rangle \downarrow \Leftrightarrow \langle S' \circ (xt_2'), t_1'\rangle \downarrow$, azaz $\langle S, \texttt{let } x = t_1 \texttt{ in } t_2\rangle \downarrow \Leftrightarrow \langle S', \texttt{let } x = t_1' \texttt{ in } t_2'\rangle \downarrow$. Mivel ez minden $(S, S') \in r_2^s$–re teljesül, a következő is igaz: $(\texttt{let } x = t_1 \texttt{ in } t_2, \texttt{let } x = t_1' \texttt{ in } t_2') \in r_2^{st} = r_2$.

□

3.4.12. Lemma (Adekvátság). Δ *adekvát a* $\downarrow$*–ra nézve.*

Bizonyítás. Legyen $t\Delta t' : T$. Azt kell megmutatni, hogy $\langle Id, t\rangle \downarrow \Leftrightarrow \langle Id, t'\rangle \downarrow$.A Δ reláció definíciója szerint ekkor $(t, t') \in T[\,]$, ami egy értékes reláció, azaz $(t, t') \in T[\,]^{vst}$. Így elég, ha megmutatjuk, hogy $(Id, Id) \in (T[\,]^v)^s$. Minden $(v, v') \in T[\,]^v$ esetén $\langle Id, t\rangle \downarrow \Leftrightarrow \langle Id, t'\rangle \downarrow$ az erre vonatkozó axiómából rögtön, ezért készen vagyunk. □

3.4.13. Tétel (Ekvivalenciák). *$\Gamma \vdash t =_{ctx} t' : T$ pontosan akkor, ha $\Gamma \vdash t\Delta t' : T$, pontosan akkor, ha $\Gamma \vdash t =_{ciu} t' : T$.*

Bizonyítás. Elég a következő tartalmazásokat látni:

$$=_{ctx} \overset{(1)}{\subseteq} =_{ciu} \overset{(2)}{\subseteq} \Delta \overset{(3)}{\subseteq} =_{ctx}$$

Az elsőt már beláttuk a 3.3.6. tételben.

A második esetében, ha felhasználjuk a 3.4.10. és 3.4.4. lemmákat, a következőt állíthatjuk:

$$\Gamma \vdash t =_{ciu} t' : T \wedge \Gamma \vdash t'\Delta t'' : T \Rightarrow \Gamma \vdash t\Delta t'' : T$$

Mivel Δ kompatibilis, ezért reflexív is, tehát az előző állításból $\Gamma \vdash t\Delta t' : T$ jön ki, ami bizonyítandó volt.

A harmadik esetben, Δ–ról már tudjuk, hogy kompatibilis, helyettesíthető és adekvát. $=_{ctx}$ pedig pont ilyenek uniójaként volt definiálva, és nem számított hogy az kongruencia–e vagy sem, tehát Δ így része $=_{ctx}$–nek. $\square$

3.5. Kiterjeszthetőség, alkalmazások

A következő tétel a 3.4.13. tétel következményeként tekinthető, amely bizonyos típusú zárt kifejezések ekvivalenciáját jellemzi. Elég zárt kifejezésekre megadni, hiszen a hasonló állítások megkaphatók helyettesítések alkalmazásával.

3.5.1. Tétel (Értékek extenzionalitása)**.**

1. *Nyílt kifejezések: Legyen* $\Gamma \vdash M : T$, $\Gamma \vdash M' : T$, *ahol* $\Gamma = X_1, \dots, X_m, x_1 : T_1, \dots, x_n : T_n$. *Ekkor* $\Gamma \vdash M =_{ctx} M' : T$ *pontosan akkor, ha minden* $G_i \in Typ$ *és minden* $N_j \in \mathtt{Term}(T_j[\overline{\alpha} := \overline{G}])$*–re* $M[\overline{\alpha} := \overline{G}, \overline{x} := \overline{N}] =_{ctx} M'[\overline{\alpha} := \overline{G}, \overline{x} := \overline{N}] : T[\overline{\alpha} := \overline{G}]$

2. *Konstansok:* $\emptyset \vdash c =_{ctx} c'$ *pontosan akkor, ha* $c = c'$

3. *Függvények: Jelölje* v *és* v' *a fun* $f(x : T_1) = t : T_2$ *és fun* $f(x : T_1) = t' : T_2$ *rekurzív függvényeket. Ekkor* $\emptyset \vdash v =_{ctx} v' : T_1 \to T_2$ *pontosan akkor, ha minden* $\emptyset \vdash v_1 : T_1$*–re,* $\emptyset \vdash t[f := v][x := v_1] =_{ctx} t'[f := v'][x := v_1] : T_2$

4. *Rekordok: Legyenek* $\emptyset \vdash v_i : T_i$ *és* $\emptyset \vdash v_i' : T_i$ *értékek halmaza, minden* $i \in 1..n$ *esetén. Ekkor* $\emptyset \vdash \{l_i = v_i^{i \in 1 \dots n}\} =_{ctx} \{l_i = v_i'^{i \in 1 \dots n}\} : \{l_i : T_i^{i \in 1 \dots n}\}$ *pontosan akkor, ha minden* $i \in 1..n$ *esetén* $\emptyset \vdash v_i =_{ctx} v_i' : T_i$

5. *Típus absztrakciók: Ha* $X \vdash v : T$ *és* $X \vdash v' : T$*, akkor* $\emptyset \vdash \lambda X.v =_{ctx} \lambda X.v' : \forall X.T$ *pontosan akkor, ha minden zárt* T'*–re* $\emptyset \vdash v[X := T'] =_{ctx} v'[X := T'] : T[X := T']$

6. *Csomagok: Minden zárt egzisztenciális* $\{\exists X, T\}$ *és* T_1*,* T_2 *típusokra és* $\emptyset \vdash v_i : T[X := T_i](i = 1, 2)$

$$\emptyset \vdash \{^*T_1, v_1\} \ \mathtt{as} \ \{\exists X, T\} =_{ctx} \{^*T_2, v_2\} \ \mathtt{as} \ \{\exists X, T\} : \{\exists X, T\}$$

fennáll, ha létezik $r \in \mathtt{TRel}(T_1, T_2)$*, melyre* $(v_1, v_2) \in T[r]$

7. *Lista: Minden zárt* $v, v' \in list_T$ *értékekre,* $\emptyset \vdash v =_{ctx} v' : list_T$ *pontosan akkor, ha* $\exists r \in \mathtt{TRel}(list_T, list_T)$*, melyre* $(v, v') \in List_T[r]$

Bizonyítás.

2.

$$\begin{aligned} v =_{ctx} v' &\Leftrightarrow (v, v') \in (T_1 \to T_2)[\,] && \text{(3.4.13. tétel)} \\ &\Leftrightarrow \forall (v_1, v_1') \in T_1[\,](vv_1, v'v_1') \in T_2[\,] && \text{(3.4.5. definíció)} \\ &\Leftrightarrow \forall (v_1, v_1') \in \mathtt{Term}(T_1) v_1 =_{ctx} v_1' : T_1 \Rightarrow vv_1 =_{ctx} v'v_1' : T_2 \\ & && \text{(3.4.13. tétel)} \end{aligned}$$

Már csak annyit kell megjegyezni, hogy $=_{ctx}$ reflexív, tranzitív és továbbá

$$vv_1 =_{ctx} t[f := v][x := v_1] : T_2 \tag{3.3}$$

ugyanis a primitív β–redukció a ciu–ekvivalencia része, tehát a 3.4.13. tétel miatt környezeti ekvivalencia is fennáll.

5. A 3.4.5. hatás definíciójából, ha

$$\begin{aligned} (\{^*T_1, v_1\} \ \mathtt{as} \ \{\exists X, T\}, \{^*T_2, v_2\} \ \mathtt{as} \ \{\exists X, T\}) &\in \{\exists r, T[r]\} \\ &\subseteq \{\exists r, T[r]\}^{st} \\ &= \{\exists X, T\}[\,] \end{aligned}$$

tehát $\{^*T_1, v_1\} \ \mathtt{as} \ \{\exists X, T\} \Delta \{^*T_2, v_2\} \ \mathtt{as} \ \{\exists X, T\} : \{\exists X, T\}$. Ezután a 3.4.13. tétel adja a kívánt állítást.

□

3.5.2. Tétel. *Adott* $X \vdash v : T$ *és* $X \vdash v' : T$*, akkor* $\lambda X.v =_{ctx} \lambda X.v' : \forall X.T$ *pontosan akkor, ha minden zárt* T_1, T_1' *típusra és* $r \in \mathtt{TRel}(T_1, T_1')$*–re* $(v[X := T_1], v'[X := T_1']) \in T[r]$.

Így értelmet nyer végre a típus hatásának fogalma is, illetve jobban látszik hogyan is segíti az ekvivalenciával való munkát. Elsődlegesen arra hivatott, hogy argumentumainak tulajdonságait átmentse a céltípus elemeire, ezáltal a bonyolultabb típusú értékek ekvivalenciájának problémája redukálható egyszerűbb típusokra.

Egy gyakorlati példát is mutatunk a logikai reláció használatára, két, egzisztenciális típushoz tartozó kifejezés ekvivalenciáját fogjuk belátni.

3.5.3. Csomagok ekvivalenciája

Legyen adott a következő típus, és értékek benne:

$$\text{type Semaphore } \{\exists X, \{\text{bit} : X,\ \text{flip} : X \to X,\ \text{read} : X \to \texttt{Bool}\}\}$$

$$\begin{aligned} v_1 = \{{}^*\texttt{Int}, \{&\text{bit} = 1, \\ &\text{flip} = \lambda x : \texttt{Int}\ \ -x, \\ &\text{read} = \lambda x : \texttt{Int}\ \ x \geq 0\}\} \text{ as Semaphore} \end{aligned}$$

$$\begin{aligned} v_2 = \{{}^*\texttt{Int}, \{&\text{bit} = 1, \\ &\text{flip} = \lambda x : \texttt{Int}\ \ x, \\ &\text{read} = \lambda x : \texttt{Int}\ \ x <= 0\}\} \text{ as Semaphore} \end{aligned}$$

Világos, hogy ezek nem ugyanazt az eredményt adják ugyanabban a környezetben, például értékük -1 és 1 a $(-.flip)1$ környezetet tekintve. De mégis ekvivalensek egymással, mert ha a második kifejezésben olyan számtípust használnánk amiben a negatív és pozitív számok szerepe fel van cserélve, ugyanahhoz a kifejezéshez jutnánk. Tehát amire szükségünk van, az egy–egy megfelelő bijekció az `Int` és a `Bool` típusokon. Ilyen persze van, az $i = \lambda x : \texttt{Int} - x$ és a $k = \lambda x : \texttt{Bool}(\texttt{if}\ \ x\ \ \texttt{then}\ \ false\ \ \texttt{else}\ \ true)$.

Most nézzük azt a bonyolultabb esetet, amikor

$$\begin{aligned} v_1 = \{{}^*\texttt{Bool}, \{&\text{bit} = true, \\ &\text{flip} = \lambda x : \texttt{Bool} \;\; not\, x, \\ &\text{read} = \lambda x : \texttt{Bool} \;\; x\}\} \text{ as Semaphore} \end{aligned}$$

$$\begin{aligned} v_2 = \{{}^*Int, \{&\text{bit} = 1, \\ &\text{flip} = \lambda x : \texttt{Int} \;\; -x, \\ &\text{read} = \lambda x : \texttt{Int} \;\; x \geq 0\}\} \text{ as Semaphore} \end{aligned}$$

Itt az előző gondolatmenetből következően egy `Int` $\leftrightarrow$ `Bool` bijekciót kéne találnunk, de az persze nincs, de a környezeti ekvivalencia teljesül rájuk. Van ugyanis olyan reláció `Int` és `Bool` között, amely megteremti a megfelelő kapcsolatot a típusok között, amivel látható az ekvivalencia is:

$$r = \{(true, 1), (false, -1)\}$$

Ezzel a relációval teljesülnek az alábbiak:

- $(s_1.\text{bit}, s_2.\text{bit}) \in r$
- Ha $(t_1, t_2) \in r$, akkor $((s_1.\text{flip})t_1, (s_2.\text{flip})t_2) \in r$
- Ha $(t_1, t_2) \in r$, akkor $(s_1.\text{read})t_1 = (s_2.\text{flip})t_2$

Tehát bár nem ugyanazt az eredményt adják vissza ugyanabban a környezetben, hisz absztrakt polimorfikus típus elemei, mégis az r reláció erejéig megkülönböztethetetlenek, és a így a megfigyelt viselkedésük is ugyanaz, nevezetes pontosan akkor terminál az egyik, ha a másik is.

Nem véletlen, hogy a reláció ilyen kicsi most, ami azt is jelenti, hogy bármely kifejezés ami Semaphore típusú elemet tartalmaz az csak olyan lehet amely a bit, flip és read mezők tetszőleges kombinációit alkalmazza az adott elemre, így kimenete is csak a relációban lévő elemek egyike lehet. Mindez a szigorú jóltípusozottság következménye.

Összefoglalva, az egzisztenciális típust tekintve, hogy ha $\exists r : T_1 \leftrightarrow T_2$ reláció, melyre $(v_1, v_2) \in R$, ahol R valamilyen T–től és r–től függő reláció, akkor $\emptyset \vdash \{^*T_1, v_1\}$ `as` $\{\exists X, T\} =_{ctx} \{^*T_2, v_2\}$ `as` $\{\exists X, T\} : \{\exists X, T\}$. Egyedül R–ről nem mondtunk semmit, illetve R pont a típus hatásával egyezik meg, amit úgy definiáltunk, hogy a fenti állítás, azaz a 3.5.1. tétel 5. pontja teljesüljön.

Bizonyítás. ($v_1 =_{ctx} v_2$: Semaphore).
A 3.5.1. tétel 5. pontja szerint elég megmutatni, hogy $(v_1, v_2) \in T[r]$, ahol $T = \{\text{bit} : X,\ \text{flip} : X \to X,\ \text{read} : X \to \texttt{Bool}\}$ a fenti v_1, v_2 értékekre és r relációval.

Mivel r értékes, használhatjuk a 3.4.7. lemmát, hogy egyszerűsítsük $T[r]$–t:

$$T[r] = \{\text{bit} = r^{st},\ \text{flip} = \texttt{fun}(r^{st}, r^{st})^{st},\ \text{read} = \texttt{fun}(r^{st}, Id^{st}_{\texttt{Bool}})\}^{st} =$$

$$\{\text{bit} = r^{st},\ \text{flip} = \texttt{fun}(r^{st}, r^{st})^{st},\ \text{read} = \texttt{fun}(r^{st}, Id^{st}_{\texttt{Bool}})\}^{st}$$

Innen azt kell látni, hogy

$$(true, 1) \in r$$
$$(\lambda x : \texttt{Bool.not}\, x, \lambda x : \texttt{Int.} - x) \in \texttt{fun}(r, r^{st})$$
$$(\lambda x : \texttt{Int.}x, \lambda x : \texttt{Int.}x \geq 0) \in \texttt{fun}(r, Id^{st}_{\texttt{Bool}})$$

Ebből az első rögtön következik r definíciójából, míg a másik kettő azért teljesül, mert a 3.4.4. lemma szerint r^{st} és $Id^{st}_{\texttt{Bool}}$ relációk megtartják a ciu–ekvivalenciát: Ha $(h_1, h_1') \in r$, akkor $(\lambda x : \texttt{Bool.not}\, x)h_1$ és $(\lambda x : \texttt{Int.}{-x})h_1'$ ciu–ekvivalensek h_2 és h_2' r relációban álló értékekkel. Innen mivel $(h_2, h_2') \in r \subseteq r^{st}$ és utóbbi zárt, $((\lambda x : \texttt{Bool.not}\, x)h_1, (\lambda x : \texttt{Int.} - x)h_1') \in r^{st}$. Mivel ez minden $(h_1, h_1') \in r$ párra fennáll, $((\lambda x : \texttt{Bool.not}\, x)h_1, (\lambda x : \texttt{Int.} - x)h_1') \in \texttt{fun}(r, r^{st})$ is teljesül. □

Vegyük észre, hogy a 3.5.1. extenzionalitási tétel csak elégséges feltételt tud adni az ekvivalenciára egzisztenciális típus esetén. És valóban, mutatható két olyan csomag érték, melyek környezeti ekvivalenciában állnak, mégsem létezik r reláció, ami mutatná ezt.

3.5.4. Típusekvivalencia

Innentől kivételesen érdemes lesz görög kisbetűt használni típusváltozók és típusok jelölésére, hogy jobban elkülönüljenek a kifejezésekben. A környezeti ekvivalencia segítségével típusok közti ekvivalenciát vagy inkább izomorfizmust is tudunk definiálni. A következő példán keresztül megmutatjuk, hogyan lehet leváltani vagy visszavezetni a nyelv lista típusát univerzálisan kvantifikált típusra. Legyen

$$L(\alpha) = \forall\alpha'(\alpha \to (\alpha \to \alpha' \to \alpha) \to \alpha)$$

Ugyanakkor definiáljunk köztük lévő polimorfikus függvényeket:

$$\begin{aligned}
I &= \Lambda\alpha(\texttt{fun}\ \ g(l : list_\alpha) = \\
&\qquad (\Lambda\alpha'(\lambda x' : \alpha'(\lambda f : \alpha \to \alpha' \to \alpha' \\
&\qquad\qquad (\texttt{case}\ \ l\ \ \texttt{of}\ \ \{nil \Rightarrow x' \mid h :: t \Rightarrow fh(g\,t\,\alpha' x' f)\}))))) \\
J &= \Lambda\alpha(\lambda p : L(\alpha)(p(list_\alpha)(N\alpha)(C\alpha))) \quad \text{ahol} \\
N &= \Lambda\alpha(nil_\alpha) \\
C &= \Lambda\alpha(\lambda h : \alpha(\lambda t : list_\alpha(h :: t)))
\end{aligned}$$

I és J zárt elemei a $\forall\alpha(list_\alpha \to L(\alpha))$ és $\forall\alpha(L(\alpha) \to list_\alpha)$ típusoknak.

3.5.5. Lemma (Keretrendszer gráfja zárt). *$\forall\tau, \tau' \in Typ$, $\forall S \in Stack(\tau, \tau')$ $graph_S = \{(M, M') \mid S[M] =_{ctx} M' : \tau'\}$ zárt.*

3.5.6. Lemma. *Azt állítjuk, hogy a fenti I és J kifejezések izomorfizmust adnak $list_\alpha$ és $L(\alpha)$ típusok közt környezeti ekvivalencia erejéig, a következő értelemben:*

$$\alpha, l : list_\alpha \vdash J\alpha(I\alpha\, l) =_{ctx} l : list_\alpha$$
$$\alpha, g : L(\alpha) \vdash I\alpha(J\alpha g) =_{ctx} g : L(\alpha)$$

Bizonyítás. A 3.5.1. tétel 1. pontja szerint ha $\tau \in Typ$, $L \in \text{Term}(list_\tau)$ és $G \in \texttt{Term}(L(\tau))$ akkor azt kell megmutatnunk, hogy $J\tau(I\tau L) =_{ctx} L : list_\tau$ és $I\tau(J\tau G) =_{ctx} G : list_\tau$.

Először tekintsük az előbbit. A (3.3) β–szabályt alkalmazva I és J definíciójára,

$$I\tau L\,\tau' M' F =_{ctx} \texttt{case}\ \ L\ \ \texttt{of}\ \ \{nil \Rightarrow M' \mid h :: t \Rightarrow Fh(I\,\tau\,t\,\tau' M' F)\} : list_\tau \tag{3.4}$$

és

$$J\tau(I\,\tau L) =_{ctx} \texttt{case}\ \ L\ \ \texttt{of}\ \ \{nil \Rightarrow nil_\tau \mid h :: t \Rightarrow h :: (J\tau(I\,\tau L))\} : list_\tau$$

minden L, M' és F–re a megfelelő típusokból. Ebből az látszik, hogy az $r = \{(L, L') \mid L =_{ctx} J\tau(I\,\tau L') : list_\tau\}$ relációval teljesül a 3.5.1. listára vonatkozó pontja, így azon tétel szerint része $=_{ctx}$–nek. Mivel $(J\tau(I\,\tau L), L) \in r$ így a kívánt állítás adódik.

Most nézzük a második részét a lemmának. Tekintsük a következő $S \in \texttt{Stack}(list_\tau, \tau')$ keretrendszert:

$$S = Id \circ (\texttt{case}\ \ -\ \ \texttt{of}\ \ \{nil \Rightarrow M' \mid h :: t \Rightarrow (Fh)(I\,\tau\,t\,\tau' M' F)\})$$

Figyelembevéve a (3.4)–es állítást, $SL =_{ctx} I\,\tau\,L\,\tau' M' F : list_\tau$ adódik, így az $r_{M',F} = \{(L, M'') \mid I\,\tau\,L\,\tau' M' F =_{ctx} M''\tau'\} \in \texttt{TRel}(list_\tau, \tau')$ zárt reláció a 3.5.5. lemma miatt. Tehát minden G kifejezésre $(G, G) \in L(\tau)[\,] = \forall r(r^{st} \to (\tau[\,] \to r^{st} \to r^{st}) \to r^{st})$, ahonnan $(G\,list_\tau, G\tau') \in r_{M',F} \to (\tau[\,] \to r_{M',F} \to r_{M',F}) \to r_{M',F}$. Ismét (3.4)–ből és $r_{M',F}$ definíciójából $(N\tau, M') \in r_{M',F}$ és $(C\tau, F) \in \tau[\,] \to r_{M',F} \to r_{M',F}$, tehát $(Glist_\tau(N\tau)(C\tau), G\tau' M' F) \in r_{M',F}$. Ez pedig a reláció definíciójából $I\tau(Glist_\tau(N\tau)(C\tau))\tau' M' F =_{ctx} G\tau' M' F : \tau'$ adja, ami már a kívánt állítás J definícióját figyelembe véve. $\square$

3.6. Összegzés

Programok automatizált transzformációjánál a legalapvetőbb követelmény, hogy a szemantikán ne változtassunk. Ilyenkor a kérdés inkább az, hogyan változtathatjuk meg a kódot, hogy az előbbi megkötésünk érvényben maradjon. Most viszont inkább elméleti bizonyítási módszert fejlesztettünk ki. Felmerül persze a kérdés, hogyan lehet alkalmazni ezt az elméletet a gyakorlatban, ha egyáltalán lehet. Például David Sands [16] kísérli meg felhasználni az itt leírtakat, de a programtranszformációs eljárásokban nem alkalmazhatók igazán a fentiek.

Az elméleti kutatások ellenben még mindig aktívan folynak a témában, hiszen vannak még típusok, amelyeken nem sikerült még relációs jellemzést adni. Illetve sok érdekes kérdést lehet fölvetni, mint például, hogy mi az a nyelvi tulajdonság ami megakadályozza a teljes karakterizációt az egzisztenciális típusokon(3.5.1. tétel), azaz miért nem teljes a logikai reláció ezen a típuson?

Irodalomjegyzék

[1] Martín Abadi, Luca Cardelli, and Pierre louis Curien. A logic for parametric polymorphism. In *Theoretical Computer Science*, pages 361–375. Springer-Verlag, 1993.

[2] G. M. Bierman, A. M. Pitts, and C. V. Russo. Operational properties of lily, a polymorphic linear lambda calculus with recursion. In *In Fourth International Workshop on Higher Order Operational Techniques in Semantics, Montréal, volume 41 of Electronic Notes in Theoretical Computer Science*. Elsevier, 2000.

[3] A. D. Gordon. Operational equivalences for untyped and polymorphic object calculi. pages 9–54, 1998.

[4] Søren Bøgh Lassen and Søren Bøgh Lassen. Relational reasoning about functions and nondeterminism. Technical report, 1998.

[5] I. A. Mason and C. L. Talcott. Equivalence in functional languages with effects. *Journal of Functional Programming*, 1(3):297–327, 1991.

[6] John C. Mitchell. On the equivalence of data representations. pages 305–329, 1991.

[7] James H. Morris. *Lambda–calculus models of programming languages.* PhD thesis, 1968.

[8] Andrew Pitts. Operationally-based theories of program equivalence. In *Semantics and Logics of Computation*, pages 241–298. Cambridge University Press, 1997.

[9] Andrew Pitts. Typed operational reasoning. In *Advanced Topics in Types and Programming Languages, chapter 7*, pages 245–289. MIT Press, 2005.

[10] Andrew Pitts and Ian Stark. Observable properties of higher order functions that dynamically create local names, or: What's new. In *In Mathematical Foundations of Computer Science, Proc. 18th Int. Symp*, pages 122–141. Springer-Verlag, 1993.

[11] Andrew Pitts and Ian Stark. Operational reasoning for functions with local state. In *In Higher Order Operational Techniques in Semantics*, pages 227–273. Cambridge University Press, 1998.

[12] Andrew M. Pitts. Existential types: Logical relations and operational equivalence. In *In Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, pages 309–326. Springer-Verlag, 1998.

[13] Andrew M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical. Structures in Computer Science*, 10(3):321–359, June 2000.

[14] Andrew M. Pitts. Operational semantics and program equivalence. In *INRIA Sophia Antipolis, 2000. Lectures at the International Summer School On Applied Semantics, APPSEM 2000, Caminha, Minho*, pages 378–412. Springer-Verlag, 2002.

[15] Gordon D. Plotkin. Lambda–definability and logical relations. 1973.

[16] David Sands. Improvement theory and its applications. In *Higher Order Operational Techniques in Semantics, Publications of the Newton Institute*, pages 275–306. Cambridge University Press, 1998.

[17] Richard Statman. Logical relations and the typed lambda calculus. *Information and Control*, 65:85–97, 1985.