EÖTVÖS LORÁND TUDOMÁNYEGYETEM
INFORMATIKAI KAR

CSÖRNYEI ZOLTÁN

FUNKCIONÁLIS PROGRAMNYELVEK IMPLEMENTÁCIÓJA

II. RÉSZ

A KOMBINÁTOR LOGIKA[1]

BUDAPEST, 2003

[1] Részben az OTKA T037742 támogatásával

Jelölések

A könyvben az ábécék betűivel és a betűtípusokkal a következőket jelöljük:

$x, y, z, \ldots$	*λ-kalkulus és kombinátor logika változó*
$E, F, G, \ldots$	*λ-kalkulus és kombinátor logika kifejezés*
$\mathcal{C}, \mathcal{D}$	*λ-kalkulus kontextus*
$X, Y, Z, \ldots$	*λ-kalkulus kontextus változó*
$\alpha, \beta, \gamma, \ldots$	*típusos λ-kalkulus típusváltozó*
$A, B, C, \ldots$	*típusos λ-kalkulus típuskifejezés*
kurzív	*matematikai függvény*
sans serif	*λ-kalkulus kifejezés neve*
sans serif	*kombinátor logika kifejezés neve*
$\equiv$	*ekvivalencia*
$=$	*egyenlőség*
$\rightarrow$	*típuskonstruktor*
$\twoheadrightarrow$	*redukció*
$\mapsto$	*kombinátor logika optimalizálás*

Tartalomjegyzék

2. FEJEZET

A kombinátor logika

Ebben a fejezetben a kombinátor logika kifejezéseit vizsgáljuk. Először a kombinátor logika kifejezéseit definiáljuk és megadjuk a kifejezések szintaktikus szabályait, majd a kifejezések szemantikájával foglalkozunk.

Két módszer van a kombinátor logika tárgyalására:

1. a λ-kalkuluson belül, mint a λ-kalkulus része, ennek előnye az, hogy felhasználhatók a λ-kalkulusból már megismert eredmények,
2. önálló elméletként, ilyen volt a kombinátor logika eredeti megfogalmazása.

Mi a második módszert követjük, de megmutatjuk a kombinátor logikának a λ-kalkulussal való kapcsolatát, például a kombinátor logika és a λ-kalkulus normál formái közötti különbséget, és módszereket adunk a kombinátor logika és a λ-kalkulus kifejezéseinek egymásba konvertálására is.

Az 1. fejezet felépítését követjük, gyakran — ha a kombinátor logikában is érvényes — szószerint megismételjük a λ-kalkulusban megadottakat.

2.1. A kombinátor logika szintaktikája

Minden funkcionális program egy kifejezésnek tekinthető, a program végrehajtása a kifejezés értékének meghatározását jelenti. Mint majd látni fogjuk, a funkcionális programokat alkotó kifejezések leírhatók a kombinátor logika kifejezéseivel is.

A funkcionális program végrehajtása ezeknek a kifejezéseknek a kiértékelését, azaz a legegyszerűbb formára hozását jelenti. A kombinátor logika ennek elvégzéséhez ad átalakítási szabályokat, azaz a kombinátor logika a kifejezések közötti olyan egyenlőségekből áll, amelyek levezethetők a konverziós szabályokból származtatott axiómákból.

Először a kombinátor logika kifejezéseinek szintaktikájával, majd az operációs szemantikával foglalkozunk, és csak ezután adjuk meg a kombinátor logika pontos definícióját.

2.1.1. Definíció. Az egyszerű kombinátor logika kifejezései:

Tegyük fel, hogy adott egy nem feltétlenül véges, egymástól páronként különböző változókat (szimbólumokat), a K *és* S *konstansokat, valamint a nyitó- és csukózárójelet tartalmazó halmaz. Ezen a halmazon, mint ábécén értelmezett kombinátor logikai kifejezések a következő szavak:*

$$
\begin{aligned}
&< \textit{kifejezés} > \quad ::= < \textit{változó} > \\
&\qquad\qquad\quad\ \mid \quad < \textit{konstans} > \\
&\qquad\qquad\quad\ \mid \quad < \textit{applikáció} > \\
&< \textit{konstans} > \quad ::= \mathsf{K} \mid \mathsf{S} \\
&< \textit{applikáció} > ::= (\ < \textit{kifejezés} > < \textit{kifejezés} >\ )
\end{aligned}
$$

A kombinátor logika kifejezéseit röviden *CL-kifejezéseknek* nevezzük, és a CL-kifejezések halmazát $\mathcal{C}$ -vel jelöljük.

A definícióban az „egyszerű” jelző arra utal, hogy a kifejezések halmaza csak a K és az S konstanst tartalmazza. A $\{\mathsf{K}, \mathsf{S}\}$ halmazt az egyszerű kombinátor logika *bázisának* nevezzük.

A továbbiakban a nem feltétlenül egy változóból álló kifejezéseket nagybetűkkel, a változókat kisbetűkkel jelöljük, és a kifejezésekben a legkülső zárójelpárt elhagyjuk.

Két kifejezés szintaktikus *azonosságára* az $\equiv$ jelet használjuk, és azt mondjuk, hogy E „szintaktikusan azonos”, vagy röviden „azonos” F-fel, azaz $E \equiv F$, ha az E és F CL-kifejezések pontosan megegyeznek. Természetesen feltesszük, hogy ha $EF \equiv GH$, akkor $E \equiv G$ és $F \equiv H$. Az $\equiv$ reláció *equivalencia reláció*, azaz *reflexív*, *szimmetrikus* és *tranzitív*.

Az azonosságot arra is használjuk, hogy a CL-kifejezéseknek nevet adjunk, és ezzel a CL-kifejezések leírását lerövidítsük. Ezeket az emlékeztető neveket, *mnemonikokat* ***dőlt sans serif*** betűtípussal írjuk.

A CL-kifejezések azonos átalakításával részletesen a 2.1.3. pontban foglalkozunk.

A CL-kifejezésekre nem definiáltunk absztrakciót, így itt nem beszélhetünk függvényekről. Két CL-kifejezés között egyetlen egy művelet van, az applikáció, és természetesen a függvényapplikáció fogalom sincs értelmezve.

2.1.1. Az applikáció

Mint a 2.1.1. definícióból látható, az applikáció bináris operátor, jele gyakran a $\cdot$, vagy mint a 2.1.2. pontban látni fogjuk, a CL-kifejezés gráfjában az applikációt a @ jel jelöli. Ebben a jegyzetben az applikációt a CL-kifejezések egymás mellé írásával, vagy ha a jobb olvashatóság miatt szükséges, akkor a CL-kifejezések közé írt szóköz karakterrel jelöljük.

2.1.2. Definíció. Az applikáció:

Ha E és F CL-kifejezések, akkor az

EF

CL-kifejezést applikációnak nevezzük.

Az applikáció *balasszociatív*, így például

$(EF)G \equiv EFG,$

azaz a redundáns zárójelpárok elhagyhatók.

2.1.3. Példa. *(Néhány CL-kifejezés)*
$\mathsf{K}EF,$
$\mathsf{S}xyz,$
$\mathsf{S}\mathsf{K}\mathsf{K},$
$x(\mathsf{K}x)\mathsf{S}y.$ □

2.1.2. A változók

Mivel a kombinátor logikában nincs absztrakció, egy CL-kifejezésben *minden változó szabad.* Ha $FV(E)$ jelöli az E kifejezés változóinak halmazát, akkor

- $FV(x) = \{x\}$,
- $FV(EF) = FV(E) \cup FV(F)$.

Mivel a CL-kifejezésekben minden változó szabad, a kombinátor logikában nincs értelme α-konverzióról beszélni.

2.1.4. Definíció. Zárt *CL*-kifejezés:

Ha egy CL-kifejezésben nincs változó, akkor a CL-kifejezést zártnak nevezzük.

2.1.5. Definíció. Kombinátor:

A zárt CL-kifejezéseket a kombinátor logika kombinátorainak nevezzük.

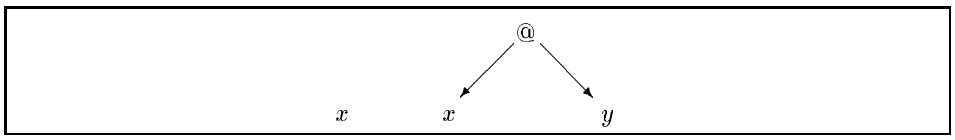
A kombinátorok halmazát $\mathcal{C}^0$ -val jelöljük.

2.1.6. Példa. *(Kombinátorok)*
A 2.1.1. definícióban szereplő K és S konstansok zárt CL-kifejezések, azaz kombinátorok voltak. Kombinátor volt a 2.1.3. példa harmadik sorában megadott CL-kifejezés is. □

Az K és az S kombinátort *primitív kombinátoroknak* is nevezzük. Megjegyezzük, hogy a λ-kalkulus kombinátorait sans serif betűtípussal írjuk, a kombinátor logikában a kombinátorok azonosító neveire a ***dőlt sans serif*** betűtípust használjuk. Tehát K és S λ-kifejezések, K és S pedig a kombinátor logika kifejezései.

2.1.3. Helyettesítés

Rendeljük hozzá az x és az xy CL-kifejezésekhez a 2.1. ábrán látható gráfokat. Ezekből az elemekből egy tetszőleges CL-kifejezés gráfja felépíthető.



2.1. ábra. Az x és az xy CL-kifejezések gráfja

A gráf felépítéséből látható, hogy egy CL-kifejezés gráfja egy olyan fa, amelynek levelein a kombinátor logika változói vagy konstansai vannak, a gráf csúcspontjai pedig az applikációnak felelnek meg. A gráfban a @ szimbólum az applikáció jele.

Egy CL-kifejezés gráfjából a CL-kifejezés részkifejezései is könnyen meghatározhatók. Egy CL-kifejezés *részkifejezését* a következőképpen definiáljuk:

2.1.7. Definíció. Egy *CL*-kifejezés részkifejezése:

Az E, F és G CL-kifejezésre

- *az E önmagának részkifejezése,*
- *ha F az E részkifejezése, akkor F az EG-nek és a GE-nek is részkifejezése.*

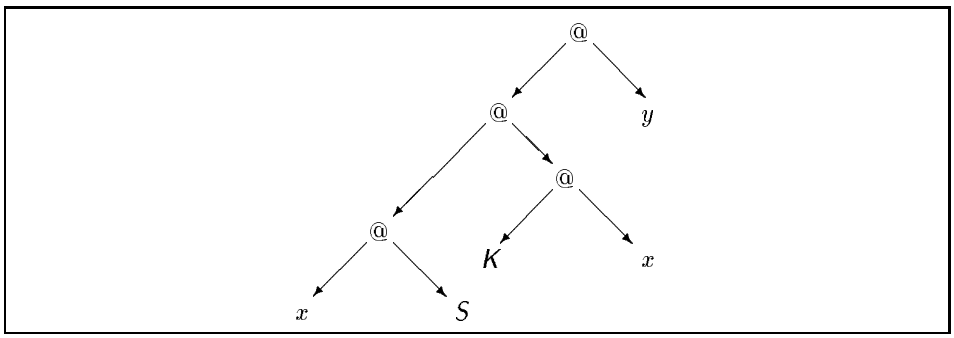
Látható, hogy a részkifejezések a CL-kifejezés gráfjában a „részfáknak" felelnek meg.

2.1.8. Példa. *(Részkifejezések)*
Az $x\mathsf{S}(\mathsf{K}x)y$ CL-kifejezésben balról jobbra haladva a következő részkifejezések vannak (2.2. ábra):

$$x, \mathsf{S}, x\mathsf{S}, \mathsf{K}, x, \mathsf{K}x, x\mathsf{S}(\mathsf{K}x), y, x\mathsf{S}(\mathsf{K}x)y.$$

Látható, hogy ugyanaz a CL-kifejezés többször is előfordulhat részkifejezésként. Az $\mathsf{S}(\mathsf{K}x)$, vagy például a $(\mathsf{K}x)y$ nem részkifejezés. □

Ha az E CL-kifejezésben az x *változókat* az F CL-kifejezéssel helyettesítjük, akkor a helyettesítéssel kapott CL-kifejezést $E[x := F]$-



2.2. ábra. Az $xS(Kx)y$ CL-kifejezés gráfja

fel jelöljük, ezt régebbi publikációkban gyakran $[F/x]E$-vel, $[x/F]E$-vel, $E[x/F]$-fel vagy $E[F/x]$-szel is jelölték. Megjegyezzük, hogy az $E[x := F]$ nem CL-kifejezés, hanem ezt a karaktersorozatot csak egy CL-kifejezés jelölésére használjuk.

Most azt vizsgáljuk meg, hogy az $E[x := F]$ milyen CL-kifejezéssel azonos.

2.1.9. Definíció. Helyettesítés:

$$1.\ x[y := G] \quad \equiv \begin{cases} G, \text{ ha } x \equiv y, \\ x, \text{ egyébként,} \end{cases}$$

$$2.\ (EF)[y := G] \equiv \ (E[y := G])(F[y := G]).$$

2.1.10. Példa. *(Kifejezések helyettesítése)*

$x[x := xy] \equiv xy$,

$x[y := xy] \equiv x$,

$xy[y := xy] \equiv xxy$,

$(xy[x := xy])[y := xy] \equiv ((xy)y)[y := xy] \equiv x(xy)(xy)$. □

A helyettesítés *balasszociatív*, így a redundáns zárójeleket elhagyva

$(E[x := F])[y := G] \equiv E[x := F][y := G]$.

A helyettesítésekre nyilvánvalóan igazak a következő egyszerű állítások:

2.1.11. Lemma. (Egyszerű helyettesítések)

1. $E[x := x] \equiv E$,
2. $E[x := F] \equiv E$, *ha* $x \notin FV(E)$,
3. $(E[x := y])[y := x] \equiv E$, *ha* $y \notin FV(E)$,
4. $(E[x := y])[y := F] \equiv E[x := F]$, *ha* $y \notin FV(E)$,
5. $(E[x := F])[x := G] \equiv E[x := F[x := G]]$.

A helyettesítés definíciójának következménye a következő állítás.

2.1.12. Lemma. (Az $E[x := F]$ változói)

Ha $x \in FV(E)$, *akkor*

$FV(E[x := F]) = (FV(E) \setminus \{x\}) \cup FV(F)$.

Az E szerkezete szerinti indukcióval bizonyítható a helyettesítési lemmának nevezett következő azonosság.

2.1.13. Lemma. (A helyettesítési lemma)

Ha $x \not\equiv y$ *és* $x \notin FV(G)$, *akkor*

$E[x := F][y := G] \equiv E[y := G][x := F[y := G]]$.

2.2. A kombinátor logika operációs szemantikája

A kombinátor logika szemantikáját *konverziós szabályokkal* írjuk le, amelyek megadják, hogy egy CL-kifejezést hogyan lehet egy másik CL-kifejezésbe transzformálni. A konverziós szabályok *reflexív*, *szimmetrikus* és *tranzitív* tulajdonsága biztosítja azt, hogy a konverziós szabályokkal átalakított CL-kifejezés az eredeti CL-kifejezésre visszakonvertálható.

A kombinátor logikában két ilyen konverziós szabály van, ezek a K és S kombinátorokra vonatkoznak. Ezekből a szabályokból látható, hogy a kombinátorok a kombinátor logikában a függvények szerepét játsszák.

2.2.1. Definíció. Gyenge redukció:

$\| \; \mathsf{K}EF \;\; \twoheadrightarrow_w E,$
$\| \; \mathsf{S}EFG \twoheadrightarrow_w EG(FG).$

A fenti definíció szerint a redukciók csak akkor hajthatók végre, ha a K-nak kettő, S-nek pedig három argumentuma van.

A definíció azt mondja ki, hogy $\mathsf{K}EF$ és $\mathsf{S}EFG$ *gyenge redukálható kifejezések*, *gyenge redexek*. A „gyenge" jelző arra utal, hogy ha egy CL-kifejezés nem redukálható, akkor előfordulhat, hogy — mint majd a 2.9.3. és 2.9.5. példákban látni fogjuk, — a neki megfeleltetett λ-kifejezésnek van redexe.

A gyenge redukció fordított irányú alkalmazását, azaz a

$E \;\; \twoheadleftarrow_w \;\; \mathsf{K}EF,$
$EF(EG) \;\; \twoheadleftarrow_w \;\; \mathsf{S}EFG$

átalakításokat *fordított gyenge redukcónak* nevezzük.

A továbbiakban, ha a könnyebb olvashatóság érdekében szükséges, a redukálandó redexet aláhúzással jelöljük.

2.2.2. Lemma. (Az I kombinátor)
$\|$ *Legyen* $\mathsf{I} \equiv \mathsf{SKK}$. *Ekkor* $\mathsf{I}E \;\; \twoheadrightarrow_w \;\; E$.

Bizonyítás. $\mathsf{I}E \equiv \underline{\mathsf{SKK}E} \;\; \twoheadrightarrow_w \;\; \underline{\mathsf{K}E(\mathsf{K}E)} \;\; \twoheadrightarrow_w \;\; E.$ □

Az I kombinátort *identifikáló* kombinátornak, vagy röviden *identifikátornak* nevezzük.

Megjegyezzük, hogy az I kombinátort az SKS kifejezéssel is megadhatjuk, hiszen

$\mathsf{I}E \equiv \underline{\mathsf{SKS}E} \;\; \twoheadrightarrow_w \;\; \underline{\mathsf{K}E(\mathsf{S}E)} \;\; \twoheadrightarrow_w \;\; E.$

2.2.3. Lemma. (A gyenge-redukció és a változók)
$\|$ *Ha* $E \;\; \twoheadrightarrow_w \;\; F$ *és* $x \notin FV(E)$, *akkor* $x \notin FV(F)$.

Bizonyítás. A lemma a gyenge-redukció definíciójából következik. □

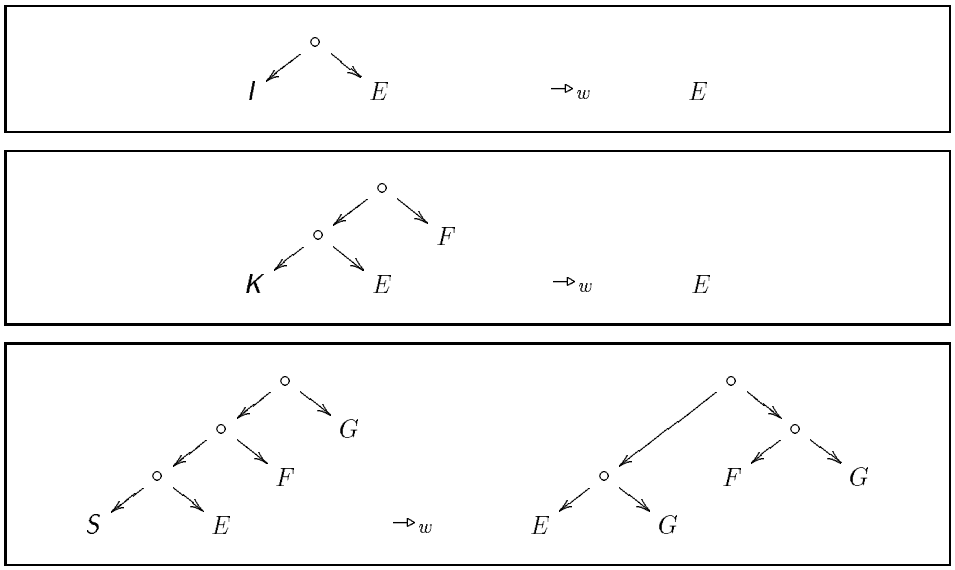
2.2.4. Lemma. (A gyenge-redukciók és helyettesítések)

Ha $E \twoheadrightarrow_w F$, *akkor tetszőleges* G *-re*

1.$G[x := E] \twoheadrightarrow_w G[x := F]$,
2.$E[x := G] \twoheadrightarrow_w F[x := G]$.

Bizonyítás. Az első állítás nyilvánvaló, a $G[x := E]$-ben levő E részkifejezésekre kell elvégezni az $E \twoheadrightarrow_\beta F$ redukciót. A második állítás bizonyítására vegyük észre, hogy ha R redex és T -re redukálható, akkor $R[x := G]$ is redex, és $T[x := G]$ -re redukálható. □

A gyenge redukció nagyon szemléletesen ábrázolható gráfokkal (2.7. ábra), így egy CL-kifejezés redukálása gráfújraírási módszerekkel könnyen megvalósítható.



2.3. ábra. A gráf-újraírási szabályok

2.2.1. Egyenlőség

Az 2.1.9. definíció két CL-kifejezés szintaktikus azonosságát adta meg. Az előző pontokban láttuk, hogy egy CL-kifejezés konverziókkal egy másik CL-

kifejezésbe alakítható át. Most két CL-kifejezés egyenlőségét definiáljuk. Az egyenlőséget az $=_w$ jellel jelöljük, és *gyenge egyenlőségnek* nevezzük.

2.2.5. Definíció. Két *CL*-kifejezés gyenge egyenlősége:

Az E és F CL-kifejezésekre $E =_w F$, ha

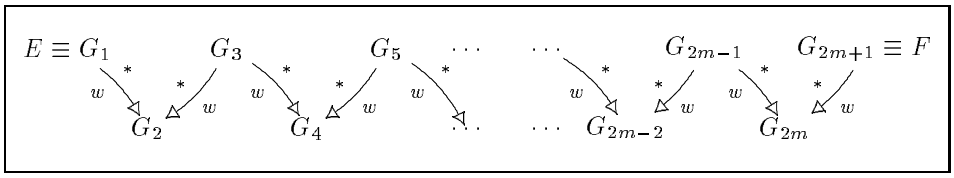
- $E \equiv F$, *vagy*
- $E \leftrightarrow_w^* F$.

A $\leftrightarrow_w$ a kétirányú redukálás lehetőségét, azaz a gyenge redukciót ($\rightarrow_w$) vagy a fordított gyenge redukciót ($\leftarrow_w$) jelöli.

Ha $E =_w F$, akkor az E és az F egymásba *konvertálhatók*. Ha $E \equiv F$, akkor $E =_w F$, de a fordított irányú következtetés természetesen nem áll fenn.

A definíció második pontja azt mondja ki, hogy ha $E =_w F$, akkor létezik olyan $F_1, F_2, \ldots, F_n$ sorozat $(n > 1)$, amelyre

$E \equiv F_1$, $F_i \leftrightarrow_w^+ F_{i+1}$ vagy $F_i \equiv F_{i+1}$ $(1 \leq i \leq n-1)$, és $F_n \equiv F$.



2.4. ábra. Az $E =_w F$ egyenlőség.

Speciálisan (2.4. ábra), létezik olyan $G_1, G_2, \ldots, G_{2m}, G_{2m+1}$ sorozat $(m \geq 0)$, amelyre

$E \equiv G_1$,
$G_{2j-1} \rightarrow_w^* G_{2j}$ vagy $G_{2j-1} \equiv G_{2j}$ $(1 \leq j \leq m)$,
$G_{2j} \leftarrow_w^* G_{2j+1}$ vagy $G_{2j} \equiv G_{2j+1}$ $(1 \leq j \leq m)$,
$G_{2m+1} \equiv F$.

A konverziós szabályok tulajdonságaiból azonnal látható, hogy az egyenlőség reláció *equivalencia reláció*, azaz az egyenlőség

- *reflexív*, $E =_w E$,
- *szimmetrikus*, ha $E =_w F$, akkor $F =_w E$,
- *tranzitív*, ha $E =_w F$ és $F =_w G$, akkor $E =_w G$.

Az egyenlőség relációval a Leibniz-szabály a következőképpen fogalmazható meg:

2.2.6. Tétel. (Leibniz-szabály:)

Ha $E_1 =_w F_1$, E_1 az E CL-kifejezés egy részkifejezése, és F az E-től csak abban különbözik, hogy benne az E_1 részkifejezés helyén az F_1 CL-kifejezés van, akkor $E =_w F$.

A Leibniz-szabály speciális esetei:

2.2.7. Következmény.

Ha $E =_w F$, akkor egy tetszőleges G CL-kifejezésre

1.$EG =_w FG$,
2.$GE =_w GF$.

Ennek a következménynek a „fordított irányú" állítása, azaz hogy például $EG =_w FG$-ből az $E =_w F$ következik, csak a $CL{+}ext$ kalkulusban (2.2.10. definíció) lesz érvényes.

2.2.2. A kombinátor logika axiómái

Miután áttekintettük a kombinátor logika kifejezéseinek szintaktikáját és szemantikáját, megismertük a CL-kifejezések átalakítási, konverziós szabályait, most megadjuk a kombinátor logika pontos definícióját.

2.2.8. Definíció. Az egyszerű típusnélküli kombinátor logika:

Az egyszerű típusnélküli kombinátor logika az egyszerű típusnélküli CL-kifejezések közötti olyan

$E =_w F \quad (E, F \in \mathcal{C})$

egyenlőségeket tartalmaz, amelyek a következő axiómák felhasználásával bizonyíthatók:

I.i.	$\mathsf{K}EF =_w E$	*gyenge konverzió*	(w_K)
I.ii.	$\mathsf{S}EFG =_w EG(FG)$	*gyenge konverzió*	(w_S)
II.i.	$E =_w E$	*reflexívitás*	(ρ)
II.ii.	$E =_w F \Rightarrow F =_w E$	*szimmetria*	(σ)
II.iii.	$E =_w F$ *és* $F =_w G \Rightarrow E =_w G$	*tranzitívitás*	(τ)
II.iv.	$E =_w F \Rightarrow EG =_w FG$	*Leibniz-szab. 1. köv.*	(μ)
II.v.	$E =_w F \Rightarrow GE =_w GF$	*Leibniz-szab. 2. köv.*	(ν)

A továbbiakban az egyszerű típusnélküli kombinátor logikát jelöljük *CL*-lel. Ezt a kombinátor logikát gyakran a gyenge redukciók miatt a *gyenge egyenlőség elméletének* is nevezik.

2.2.3. A kiterjesztés

A λ-kalkulusban láttuk, hogy a kiterjeszthetőség azon az észrevételen alapul, hogy ha két függvény értéke minden argumentumra azonos, akkor a két függvény megegyezik. A kiterjeszthetőséget definiálhatjuk a kombinátor logikában is, mivel a kéttagú applikációk első tagját függvénynek tekinthetjük.

2.2.9. Definíció. Kiterjeszthetőség:

Ha $Ex =_w Fx$, *és* $x \notin FV(E)$, $x \notin FV(F)$, *akkor* $E =_w F$.

2.2.10. Definíció. Az egyszerű típusnélküli kiterjesztett kombinátor logika:

A kiterjesztett kombinátor logikát úgy kapjuk meg, hogy a kombinátor logika axiómáit kiegészítjük a

$II.vi.Ex =_w Fx \Rightarrow E =_w F$ *kiterjeszthetőség* (ext)

axiómával.

A kiterjesztett kombinátor logikát $CL{+}ext$ -tel jelöljük.

Mivel a kombinátor logikában nincs explicit függvényfogalom, nem beszélhetünk sem η-, sem ξ-szabályról, és így a λ-kalkulussal ellentétben, a kombinátor logikának csak ez az egy bővítése létezik.

A $CL{+}ext$ kalkulusnak, mint majd a 2.9. pontban látni fogjuk, a λ-kalkulussal való összehasonlításban lesz nagy szerepe.

2.3. A CL-kifejezés gyenge normál formája

A CL-kifejezések átalakítására gyenge redukciók és fordított gyenge redukciók szolgálnak. A redukciók sorozatának minden lépésében egy *gyenge redukálható kifejezést*, *gyenge redexet* redukálunk. Ha az E_1 CL-kifejezést gyenge redukciók sorozatával az E_2 CL-kifejezésre alakítjuk át, azaz $E_1 \twoheadrightarrow_w^+ E_2$, akkor azt mondjuk, hogy E_2 az E_1 *gyenge redukáltja.*

2.3.1. Definíció. Gyenge normál forma:

Ha egy CL-kifejezésben nincs gyenge redukálható kifejezés, akkor a CL-kifejezés gyenge normál formában van.

Ha egy CL-kifejezés gyenge redukciók sorozatával gyenge normál formára hozható, akkor azt mondjuk, hogy a CL-kifejezésnek *van gyenge normál formája.*

Ha egy CL-kifejezés gyenge normál formában van, akkor nincs $\mathsf{K}EF$, vagy $\mathsf{S}EFG$ alakú részkifejezése.

2.3.2. Példa. *(A következő CL-kifejezések gyenge normál formában vannak)*

x,

xy,

Kx,
$Sx(Sxy)$. □

2.3.3. Példa. *(A következő CL-kifejezés nincs gyenge normál formában, de gyenge normál formára hozható:)*
$\underline{Sx(Kx)y} \rightarrow_w$
$xy(\underline{Kxy}) \rightarrow_w$
xyx □

Ha egy CL-kifejezés nincs gyenge normál formában, akkor nem feltétlenül hozható gyenge normál formára:

2.3.4. Példa. *(A következő CL-kifejezés nincs gyenge normál formában és a CL-kifejezésnek nincs gyenge normál formája)*
$\underline{SII(SII)} \rightarrow_w$
$\underline{I(SII)}(I(SII)) \rightarrow_w$
$SII\underline{(I(SII))} \rightarrow_w$
$SII(SII) \rightarrow_w$
...

A gyenge redukciók sorozata nem terminál, hiszen a harmadik lépésben az eredeti CL-kifejezést kapjuk vissza. A 2.3.6. példában majd látni fogjuk, hogy a harmadik sorban egy másik redex választása is ugyanezt az eredményt adja. □

2.3.1. Redukálási stratégiák

Ha egy CL-kifejezésnek több gyenge redexe is van, akkor a gyenge redexek különböző redukálási sorrendjei különböző gyenge redukálási sorozatokat határoznak meg. Több gyenge redex esetén lehetséges, hogy a különböző gyenge redukálási sorozatok ugyanahhoz a gyenge normál formához vezetnek, de az is lehetséges, hogy az egyik gyenge redukálási sorozattal nem kapjuk meg a CL-kifejezés gyenge normál formáját, egy másik sorozattal viszont a gyenge normál formához jutunk.

2.3.5. Példa. *(Különböző redukálási sorrenddel ugyanazt a gyenge normál formát kapjuk)*
Először redukáljunk a belső gyenge redexszel:

$I(\underline{Ix}) \rightarrow_w$
$\underline{Ix} \rightarrow_w$
$x.$

Most először redukáljunk a külső gyenge redexszel:

$\underline{I(Ix)} \rightarrow_w$
$\underline{Ix} \rightarrow_w$
$x.$ □

2.3.6. Példa. *(Egyik redukálási sorrenddel sem kapunk gyenge normál formát)*
A 2.3.4. példában láttuk, hogy

$SII(SII) \rightarrow_w^+$
$SII(SII),$

most egy másik redukálási sorozattal:

$\underline{SII(SII)} \rightarrow_w$
$\underline{I(SII)}(I(SII)) \rightarrow_w$
$\underline{SII(I(SII))} \rightarrow_w$
$\underline{I(I(SII))}(I(I(SII))) \rightarrow_w$
$\underline{(I(SII))}(I(I(SII))) \rightarrow_w$
$SII(I(I(SII))) \rightarrow_w$
...

ebből pedig azonnal látszik, hogy nincs olyan redukálási sorrend, ami egy gyenge normál formához vezetne. □

2.3.7. Példa. *(A K gyenge redukciójának a végrehajtásával gyenge*

normál formát kapunk, az S gyenge redukciójának végrehajtásakor az 2.3.4. példában is szereplő végtelen ciklust)

$\underline{Kx(SII(SII))} \rightarrow_w$

$x,$

$Kx(\underline{SII(SII)}) \rightarrow_w^+$

$Kx(\underline{SII(SII)}) \rightarrow_w^+$

$\ldots$ □

A kombinátor logikában is érvényesek a λ-kalkulusban megismert Church-Rosser tételek, és azok következményei.

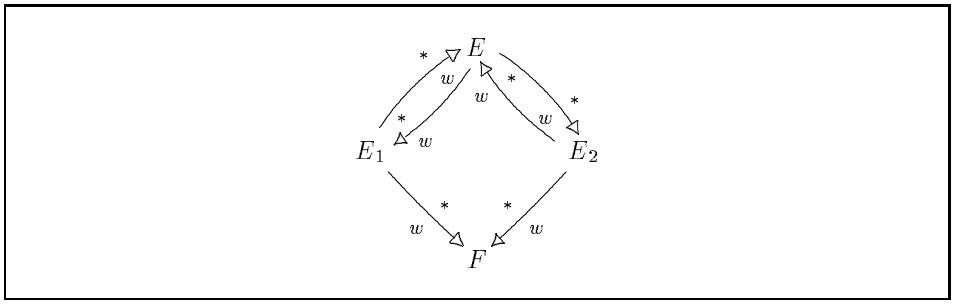
A következő tétel és a tétel következménye ennek az értéknek az egyértelműségét mondja ki.

2.3.8. Tétel. (Az I. Church–Rosser tétel)

Ha $E_1 =_w E_2$, akkor létezik egy olyan F CL-kifejezés, amelyre $E_1 \rightarrow_w^* F$, és $E_2 \rightarrow_w^* F$.

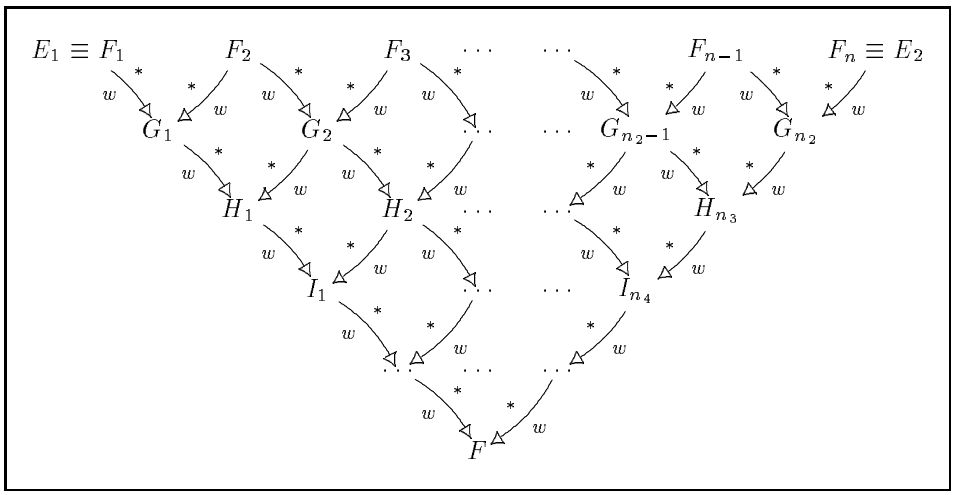
Bizonyítás. A tételt nem bizonyítjuk, a bizonyítás [Bar84] és [Hin86]-ban is megtalálható. A bizonyítás alapja a következő állítás:

Ha $E \leftrightarrow_w^* E_1$ és $E \leftrightarrow_w^* E_2$, akkor létezik egy olyan F CL-kifejezés, amelyre $E_1 \rightarrow_w^* F$, és $E_2 \rightarrow_w^* F$.



2.5. ábra. A rombusz tulajdonság

Ez az állítás az 2.5. ábrán látható, a gráf alapján ezt a tulajdonságot gyakran *rombusz-tulajdonságnak* is nevezik. Az $E_1 =_w E_2$ egyenlőségben



2.6. ábra. Az I. Church–Rosser tétel

szereplő konverziókra a rombusz-tulajdonságot alkalmazva (2.6. ábra) a tétel állítása bizonyítható. □

Az I. Church–Rosser tétel tehát azt mondja ki, hogy két egymásba transzformálható *CL*-kifejezéshez mindig létezik, esetleg az egyikkel meg is egyező *CL*-kifejezés, amelybe mindkét *CL*-kifejezés átkonvertálható. A tételből két egyszerűen bizonyítható állítás következik:

2.3.9. Következmény.

Ha $E_1 =_w E_2$, *és* E_2 *gyenge normál formában van, akkor* $E_1 \twoheadrightarrow_w^* E_2$.

A következmény tehát azt mondja ki, hogy ha egy *CL*-kifejezésnek létezik gyenge normál formája, akkor a gyenge normál forma a *CL*-kifejezésből gyenge redukciók sorozatával előállítható.

2.3.10. Következmény.

Minden CL-kifejezésnek legfeljebb egy gyenge normál formája van.

Bizonyítás. Tegyük fel, hogy az E *CL*-kifejezésnek az F_1 és F_2 két különböző gyenge normál formája. Ekkor az I. Church–Rosser tétel alapján létezik olyan F *CL*-kifejezés, amelyre $F_1 \twoheadrightarrow_w^* F$ és $F_2 \twoheadrightarrow_w^* F$. Az F_1 és

az F_2 gyenge normál formák, azaz tovább nem redukálhatók, így csak az $F_1 =_w F_2 =_w F$ lehetséges. □

A fenti két következményből azonnal következik a következő tulajdonság:

2.3.11. Következmény.

Ha E és F mindegyike gyenge normál forma, és $E \not\equiv F$, akkor $E \neq_w F$.

A fentiek szerint tehát, ha eljutunk a gyenge normál formához, akkor már közömbös, hogy milyen gyenge redukciók eredményeként kaptuk azt meg, hiszen minden más normál formára hozó redukció ugyanezt az eredményt adja. Ez a kombinátor logika *konzisztens* tulajdonságát mutatja, azaz azt, hogy minden olyan sorozat, amelyik gyenge normál formát állít elő, ugyanahhoz a CL-kifejezéshez vezet. Ugyanakkor a fentiekből az is következik, hogy a kombinátor logika *konfluens*, azaz nincs két olyan gyenge redukciós sorozat, amely különböző gyenge normál formákat állítana elő.

Egy CL-kifejezés legkülső gyenge redexe az a redukálható kifejezése, amelyik nincs más gyenge redexe belsejében. Azt a redukálási stratégiát, amelyik a legbaloldalibb legkülső gyenge redexet redukálja, *normál sorrendű redukálási stratégiának* nevezzük, és azt mondjuk, hogy alkalmazásával *normál sorrendű* redukálásokat hajtunk végre.

A normál sorrendű redukálás jelentőségét a következő tétel adja meg:

2.3.12. Tétel. (A II. Church–Rosser tétel, a normalizálás tétele)

A normál sorrendű redukálási stratégia mindig normalizáló redukálási stratégia.

A tétel szerint, ha egy CL-kifejezésnek van gyenge normál formája, akkor a gyenge normál forma normál sorrendű redukálási stratégiával előállítható.

2.4. A zárójeles absztrakció

Mint láttuk, a kombinátor logikában nincs olyan függvény jellegű absztrakció, mint a λ-kalkulusban. A kombinátor logikában absztrakción a CL-kifejezésben levő változók absztrahálását értjük, így az absztrahálást végző algoritmusok csak szintaktikus átalakítást végeznek, azaz az absztrakcióval a CL-kifejezést egy másik, de az eredetihez hasonló tulajdonságokkal rendelkező alakjában írjuk fel.

Mivel az x változónak az E CL-kifejezésre alkalmazott absztrakcióját régebben $[x]E$-vel jelölték, ezt az absztrakciót *zárójeles absztrakciónak* nevezzük. Mi az E kifejezés x szerinti absztrakciójára a ma használatos $\lambda^* x\,.\,E$ jelölést fogjuk használni.

Hangsúlyozzuk, hogy a kombinátor logika absztrakciója csak szintaktikus átalakítás, a λ^* ennek az átalakításnak a jele, és ellentétben a λ-kalkulussal, a λ^* nem része a kombinátor logika kifejezéseinek.

Többféle absztrakciós algoritmus létezik, a legegyszerűbb absztrakciós algoritmus a következő:

2.4.1. Definíció. A λ^* absztrakciós algoritmus:

A $\lambda^* x\,.\,E$ kifejezést az E szerkezete szerint definiáljuk:

$$\begin{aligned}
&\lambda^* x\,.\,x \quad \equiv I,\\
&\lambda^* x\,.\,E \quad \equiv K E, \qquad\qquad ha\ x \notin FV(E),\\
&\lambda^* x\,.\,EF \equiv S\,(\lambda^* x\,.\,E)(\lambda^* x\,.\,F), \quad egyébként.
\end{aligned}$$

2.4.2. Példa. *(Az $xy(yy)$ CL-kifejezés x szerinti absztrakciója)*

$$\begin{aligned}
&\lambda^* x\,.\,yx(yy) \equiv\\
&S\,(\lambda^* x\,.\,yx)(\lambda^* x\,.\,yy) \equiv\\
&S\,(S\,(\lambda^* x\,.\,y)(\lambda^* x\,.\,x))(\lambda^* x\,.\,yy) \equiv\\
&S\,(S\,(Ky)(\lambda^* x\,.\,x))(\lambda^* x\,.\,yy) \equiv\\
&S\,(S\,(Ky)I)(\lambda^* x\,.\,yy) \equiv\\
&S\,(S\,(Ky)I)(K(yy))
\end{aligned}$$

□

2.4.3. Példa. *(Az xy CL-kifejezés x és y szerinti absztrakciói)*

$\lambda^* yx\,.\,xy \equiv$
$\lambda^* y\,.\,(\lambda^* x\,.\,xy) \equiv$
$\lambda^* y\,.\,(\mathsf{S}\,(\lambda^* x\,.\,x)(\lambda^* x\,.\,y)) \equiv$
$\lambda^* y\,.\,(\mathsf{S}\,\mathsf{I}(\mathsf{K}y)) \equiv$
$\mathsf{S}\,(\lambda^* y\,.\,(\mathsf{S}\,\mathsf{I}))(\lambda^* y\,.\,(\mathsf{K}y)) \equiv$
$\mathsf{S}\,(\mathsf{K}\,(\mathsf{S}\,\mathsf{I}))(\lambda^* y\,.\,(\mathsf{K}y)) \equiv$
$\mathsf{S}\,(\mathsf{K}\,(\mathsf{S}\,\mathsf{I}))(\mathsf{S}\,(\lambda^* y\,.\,\mathsf{K})(\lambda^* y\,.\,y)) \equiv$
$\mathsf{S}\,(\mathsf{K}\,(\mathsf{S}\,\mathsf{I}))(\mathsf{S}\,(\mathsf{K}\mathsf{K})\mathsf{I})$

Ugyanakkor az absztrakciókat fordított sorrendben végezve:

$\lambda^* xy\,.\,xy \equiv \cdots \equiv$
$\mathsf{S}\,(\mathsf{S}\,(\mathsf{K}\mathsf{S})(\mathsf{S}\,(\mathsf{K}\mathsf{K})\mathsf{I}))(\mathsf{K}\mathsf{I})$.

Látható, hogy az eredményül kapott kifejezések nem tartalmaznak változókat. □

A zárójeles absztrakció definíciójából látható, hogy ha az E kifejezésre a λ^* absztrakciót alkalmazzuk, az eredményül kapott kifejezés már nem tartalmaz x változót. Ezt mondja ki a következő tétel.

2.4.4. Tétel. (Változók eliminálása)

Minden E CL-kifejezésre

$FV(\lambda^* x\,.\,E) = FV(E) \setminus \{x\}$.

A kombinátor logika zárójeles absztrakciója veszi át a λ-kalkulus függvényfogalmát, olyan értelemben, hogy egy $(\lambda^* x\,.\,E)F$ CL-kifejezés gyenge redukciójának eredménye az $E[x := F]$ helyettesítés lesz, azaz a $(\lambda^* x\,.\,E)F$ applikáció a λ-kalkulus függvényapplikációjához hasonló, és az eredmény olyan, mintha a λ-kalkulus β-redukcióját hajtottuk volna végre.

2.4.5. Tétel. (A zárójeles absztrakció és az applikáció)

Minden E és F CL-kifejezésre

$(\lambda^* x\,.\,E)F \;\twoheadrightarrow_w^+\; E[x := F]$.

Ha a fenti tételben speciálisan $F \equiv x$, akkor $(\lambda^* x\,.\,E)x \twoheadrightarrow_w^+ E$, azaz visszakapjuk azt az E kifejezést, amelyre a zárójeles absztrakció vonatkozik.

2.4.6. Példa. *(A $(\lambda^* yx\,.\,xy)EF$ és a $(\lambda^* xy\,.\,xy)FE$ applikációk)*

$(\lambda^* yx\,.\,xy)EF \equiv$
$\underline{(S(K(SI))(S(KK)I))EF} \twoheadrightarrow_w$
$\underline{K(SI)E}(S(KK)IE)F \twoheadrightarrow_w$
$\underline{SI(S(KK)IE)F} \twoheadrightarrow_w$
$\underline{IF}(S(KK)IEF) \twoheadrightarrow_w$
$F(\underline{S(KK)IE}F) \twoheadrightarrow_w$
$F(\underline{KKE}(IE)F) \twoheadrightarrow_w$
$F(K(\underline{IE})F) \twoheadrightarrow_w$
$F(\underline{KEF}) \twoheadrightarrow_w$
FE

és

$(\lambda^* xy\,.\,xy)FE \twoheadrightarrow_w$
$\underline{(S(S(KS)(S(KK)I))(KI))FE} \twoheadrightarrow_w$
$(\underline{S(KS)(S(KK)I)F})(KIF)E \twoheadrightarrow_w$
$(\underline{KSF})(S(KK)IF)(KIF)E \twoheadrightarrow_w$
$\underline{S(S(KK)IF)(KIF)E} \twoheadrightarrow_w$
$\underline{S(KK)IFE}(KIFE) \twoheadrightarrow_w$
$\underline{KKF}(IF)E(KIFE) \twoheadrightarrow_w$
$\underline{K(IF)E}(KIFE) \twoheadrightarrow_w$
$\underline{IF}(KIFE) \twoheadrightarrow_w$
$F(\underline{KIF}E) \twoheadrightarrow_w$
$F(\underline{IE}) \twoheadrightarrow_w$
FE □

A λ-kalkulusban láttuk, hogy egy λ-absztrakció törzsében levő x szabad

változó csak akkor helyettesíthető F-fel, ha F-ben nincs szabad x változó, hiszen ez az x változó a helyettesítés után kötötté válna. Hasonló állítás mondható ki a kombinátor logikában is.

2.4.7. Tétel. (A zárójeles absztrakció és a helyettesítés)

Ha $x \not\equiv y$ *és* $x \notin FV(F)$, *akkor*

$(\lambda^* x \,.\, E)[y := F] \equiv \lambda^* x \,.\, E[y := F]$.

A tétel az E szerkezete szerinti indukcióval bizonyítható. A tétel feltételei nem feleslegesek, ezekre különösen a több változó szerint zárójeles absztrakciókban kell ügyelni, mint azt a következő példa is mutatja.

2.4.8. Példa. *(Hibás helyettesítés)*

$\lambda^* yx \,.\, y \equiv \mathsf{S}(\mathsf{K}\mathsf{K})\mathsf{I}$,

és így

$(\lambda^* yx \,.\, y)x \equiv (\mathsf{S}(\mathsf{K}\mathsf{K})\mathsf{I})x \;\twoheadrightarrow_w$
$\mathsf{K}\mathsf{K}x(\mathsf{I}x) \;\twoheadrightarrow_w^+$
$\mathsf{K}x$.

Ugyanakkor 2.4.5 tétel alapján

$(\lambda^* yx \,.\, y)x \equiv (\lambda^* y \,.\, (\lambda^* x \,.\, y))x \;\twoheadrightarrow_w^+ \; (\lambda^* x \,.\, y)[y := x]$,

és nem figyelve a fenti tétel feltételeit:

$(\lambda^* x \,.\, y)[y := x] \equiv$
$\lambda^* x \,.\, y[y := x] \equiv$
$\lambda^* x \,.\, x \equiv \mathsf{I}$,

azaz eredményül az identitás kombinátort kapnánk. □

Ezt a problémát a λ-kalkulusban az α-konverzióval tudtuk megoldani. A λ-kalkulus α-konverziójához hasonló állítás mondható ki a zárójeles absztrakcióra is.

2.4.9. Tétel. (Az α-konverzió tétele)

Minden E CL-kifejezésre teljesül, hogy

$\lambda^* x\,.\,E \equiv \lambda^* y\,.\,E[x := y]$, *ha* $y \notin FV(E)$.

Mint már korábban említettük, többfajta absztrakciós algoritmus létezik. Most megadjuk a λ_1^* és a λ_2^* algoritmusokat.

2.4.10. Definíció. A λ_1^* absztrakciós algoritmus:

A $\lambda_1^* x\,.\,E$ kifejezést az E szerkezete szerint definiáljuk:

$$\begin{array}{lll}
\lambda_1^* x\,.\,x & \equiv \mathsf{I}, & \\
\lambda_1^* x\,.\,y & \equiv \mathsf{K}y, & \textit{ha } x \not\equiv y, \\
\lambda_1^* x\,.\,E & \equiv \mathsf{K}E, & \textit{ha } E \in \{\mathsf{K}, \mathsf{S}\}, \\
\lambda_1^* x\,.\,EF & \equiv \mathsf{S}(\lambda_1^* x\,.\,E)(\lambda_1^* x\,.\,F), & \textit{egyébként.}
\end{array}$$

A definícióból látható, hogy a $\lambda_1^* x$ absztrakcióval $\mathsf{K}E$ típusú eredményt csak akkor kapunk, ha $E \in \{\mathsf{K}, \mathsf{S}, y\}$, és minden FG applikáció absztrakciója, függetlenül attól, hogy az x szabad változóként szerepel-e benne, egy S-sel kezdődő hármas applikáció lesz. Ezért a λ_1^* absztrakcióval hosszabb kifekezéseket kapunk, mint a λ^* -gal.

2.4.11. Példa. *(A λ_1^* absztrakció)*
Ezzel az absztrakcióval a 2.4.2. és 2.4.3. példák eredményei a következők:

(a) $\lambda_1^* x\,.\,yx(yy) \equiv$
$\mathsf{S}(\lambda_1^* x\,.\,yx)(\lambda_1^* x\,.\,yy) \equiv$
$\mathsf{S}(\mathsf{S}(\lambda_1^* x\,.\,y)(\lambda_1^* x\,.\,x))(\lambda_1^* x\,.\,yy) \equiv$
$\mathsf{S}(\mathsf{S}(\mathsf{K}y)(\lambda_1^* x\,.\,x))(\lambda_1^* x\,.\,yy) \equiv$
$\mathsf{S}(\mathsf{S}(\mathsf{K}y)\mathsf{I})(\lambda_1^* x\,.\,yy) \equiv$
$\mathsf{S}(\mathsf{S}(\mathsf{K}y)\mathsf{I})(\mathsf{S}(\lambda_1^* x\,.\,y)(\lambda_1^* x\,.\,y)) \equiv$
$\mathsf{S}(\mathsf{S}(\mathsf{K}y)\mathsf{I})(\mathsf{S}(\mathsf{K}y)(\lambda_1^* x\,.\,y)) \equiv$
$\mathsf{S}(\mathsf{S}(\mathsf{K}y)\mathsf{I})(\mathsf{S}(\mathsf{K}y)(\mathsf{K}y))$,

(b) $\lambda_1^* yx\,.\,xy \equiv$

$\lambda_1^* y\,.\,(\lambda_1^* x\,.\,xy) \equiv$
$\lambda_1^* y\,.\,(S\,(\lambda_1^* x\,.\,x)(\lambda_1^* x\,.\,y)) \equiv$
$\lambda_1^* y\,.\,(S\,I(K y)) \equiv$
$S\,(\lambda_1^* y\,.\,S\,I)(\lambda_1^* y\,.\,K y) \equiv$
$S\,(S\,(\lambda_1^* y\,.\,S)(\lambda_1^* y\,.\,I))(\lambda_1^* y\,.\,K y) \equiv$
$S\,(S\,(K\,S)(K\,I))(\lambda_1^* y\,.\,K y) \equiv$
$S\,(S\,(K\,S)(K\,I))(S\,(\lambda_1^* y\,.\,K)(\lambda_1^* y\,.\,y)) \equiv$
$S\,(S\,(K\,S)(K\,I))(S\,(K\,K\,K)\,I),$

(c) $\lambda_1^* xy\,.\,xy \equiv$
$\lambda_1^* x\,.\,(\lambda_1^* y\,.\,xy) \equiv$
$\lambda_1^* x\,.\,(S\,(\lambda_1^* y\,.\,x)(\lambda_1^* y\,.\,y)) \equiv$
$\lambda_1^* x\,.\,(S\,(K x)\,I) \equiv$
$S\,(\lambda_1^* x\,.\,(S\,(K x)))(\lambda_1^* x\,.\,I) \equiv$
$S\,(S\,(\lambda_1^* x\,.\,S)(\lambda_1^* x\,.\,K x))(\lambda_1^* x\,.\,I) \equiv$
$S\,(S\,(K\,S)(\lambda_1^* x\,.\,K x))(\lambda_1^* x\,.\,I) \equiv$
$S\,(S\,(K\,S)(S\,(\lambda_1^* x\,.\,K)(\lambda_1^* x\,.\,x)))(\lambda_1^* x\,.\,I) \equiv$
$S\,(S\,(K\,S)(S\,(K\,K)\,I))(\lambda_1^* x\,.\,I) \equiv$
$S\,(S\,(K\,S)(S\,(K\,K)\,I))(K\,I).$ □

2.4.12. Definíció. A λ_2^* absztrakciós algoritmus:

A $\lambda_2^* x\,.\,E$ kifejezést az E szerkezete szerint definiáljuk:

$\lambda_2^* x\,.\,x \equiv I$
$\lambda_2^* x\,.\,E \equiv K E,$ *ha* $x \notin FV(E),$
$\lambda_2^* x\,.\,Ex \equiv E,$ *ha* $x \notin FV(E),$
$\lambda_2^* x\,.\,EF \equiv S\,(\lambda_2^* x\,.\,E)(\lambda_2^* x\,.\,F),$ *egyébként.*

Felhívjuk a figyelmet arra, hogy a definíció harmadik sorában a λ-kalkulus η-konverziójához hasonló azonosságot látunk.

2.4.13. Példa. *(A λ_2^* absztrakció)*

A λ_2^* absztrakcióval az előző példák eredményei a következők:

(a) $\lambda_2^* x\,.\,yx(yy) \equiv$
$\mathsf{S}(\lambda_2^* x\,.\,yx)(\lambda_2^* x\,.\,yy) \equiv$
$\mathsf{S}y(\lambda_2^* x\,.\,yy) \equiv$
$\mathsf{S}y(\mathsf{K}(yy)),$

(b) $\lambda_2^* yx\,.\,xy \equiv$
$\lambda_2^* y\,.\,(\lambda_2^* x\,.\,xy) \equiv$
$\lambda_2^* y\,.\,(\mathsf{S}(lcskx\,.\,x)(\lambda_2^* x\,.\,y)) \equiv$
$\lambda_2^* y\,.\,(\mathsf{SI}(\mathsf{K}y)) \equiv$
$\mathsf{S}(\lambda_2^* y\,.\,\mathsf{SI})(\lambda_2^* y\,.\,\mathsf{K}y) \equiv$
$\mathsf{S}(\mathsf{K}(\mathsf{SI}))\mathsf{K},$

(c) $\lambda_2^* xy\,.\,xy \equiv$
$\lambda_2^* x\,.\,(\lambda_2^* y\,.\,xy) \equiv$
$\lambda_2^* x\,.\,x \equiv$
$\mathsf{I}.$ □

Majd látni fogjuk, hogy a Curry-Howard izomorfizmus alapján a kombinátor logika a Hilbert-stílusú intuicionista ítéletkalkusnak felel meg. Mivel a zárójeles absztrakció alkalmazása „eltünteti" a CL-kifejezésekből a változókat, a zárójeles absztrakció éppen azt a Schönfinkel és Curry által kitűzött célt valósította meg, hogy a logikai formulákban ne legyenek változók, és így a matematika, a logika megalapozását egy változó nélküli formális rendszerrel kísérelje meg.

2.5. Zárójeles absztrakció új kombinátorokkal

A λ^* zárójeles absztrakció definíciójából látható, hogy egy CL-kifejezés átalakításakor a λ^* egyes lépései megduplázzák a CL-kifejezésben levő applikációk darabszámát. Általában azt mondhatjuk, hogy ha a CL-kifejezés hossza m, és a CL-kifejezésben n változó van, akkor az n változóra alka-

lmazott λ^* zárójeles absztrakció alkalmazása után a CL-kifejezés hossza $O(mn^3)$.

A CL-kifejezések hosszának csökkentésére két módszer adható meg:

1. optimalizálás,
2. új kombinátorok bevezetése.

Az *egyszerű* CL+*ext* kombinátor logikára az *optimalizálás* a következő *újraírási szabályokkal* adható meg, az újraírási szabály azt jelenti, hogy $\mapsto$ baloldalán álló kifejezés a jobboldalon levő kifejezéssel helyettesíthető.

- $S(KE)I \quad \mapsto E$,
- $S(KE)(KI) \mapsto (EF)$,

hiszen minden G-re

$S(KE)IG \rightarrow_w KEG(IG) \rightarrow_w E(IG) \rightarrow_w EG$,

és ebből az *ext* kiterjesztési axióma alapján az első optimalizációs szabályt kapjuk meg. Hasonlóan,

$S(KE)(KF)G \rightarrow_w KEG(KFG) \rightarrow_w E(KFG) \rightarrow_w EF$,

és

$K(EF)G \rightarrow_w EF$.

Így az $=_w$ egyenlőség definíciója alapján

$S(KE)IG =_w K(EF)G$,

ebből pedig ismét az *ext* kiterjesztési axióma felhasználásával a második optimalizációs szabály adódik.

2.5.1. A $CL^{(C)}$ kombinátor logika és a λ_3^* absztrakció

Most az *egyszerű* CL kombinátor logika konstansainak halmazát bővítsük a B és C új kombinátorokkal.

2.5.1. Definíció. A $CL^{(C)}$ kombinátor logika konstansai:

$< konstans > ::= K \mid S \mid B \mid C$

A $CL^{(C)}$ kombinátor logika *bázisa* tehát a $\{K, S, B, C\}$ halmaz. Az új kombinátorokra vonatkozó *konverziós szabályok* a következők:

2.5.2. Definíció. A $CL^{(C)}$ kombinátor logika új gyenge redukciói:

$B\,E\,F\,G \rightarrow_w E(FG),$
$C\,E\,F\,G \rightarrow_w EGF.$

Látható (2.7. ábra), hogy a B kombinátor függvény-kompozíciót hajt végre, hiszen kiírva néhány zárójelet

$$((B\,E)F)G \rightarrow_w E(FG),$$

ezért a kombinátort *kompozítornak* nevezzük. A B kombinátor fordított gyenge redukciója pedig éppen a *currizást* valósítja meg. A C az E két argumentumát cseréli fel, vagy ha az E és F mindegyikét függvénynek tekintjük, akkor a G argumentumot az E-hez, az „eggyel balra levő" függvényhez viszi át. A C kombinátor a *permutátor* nevet kapta.

Könnyen belátható, hogy az új kombinátorokkal a $CL^{(C)}+ext$ kombinátor logikában a következő optimalizációk hajthatók végre:

- $S(KE)F \mapsto BEF,$
- $SE(KF) \mapsto CEF,$

hiszen például az első újraírási szabályra

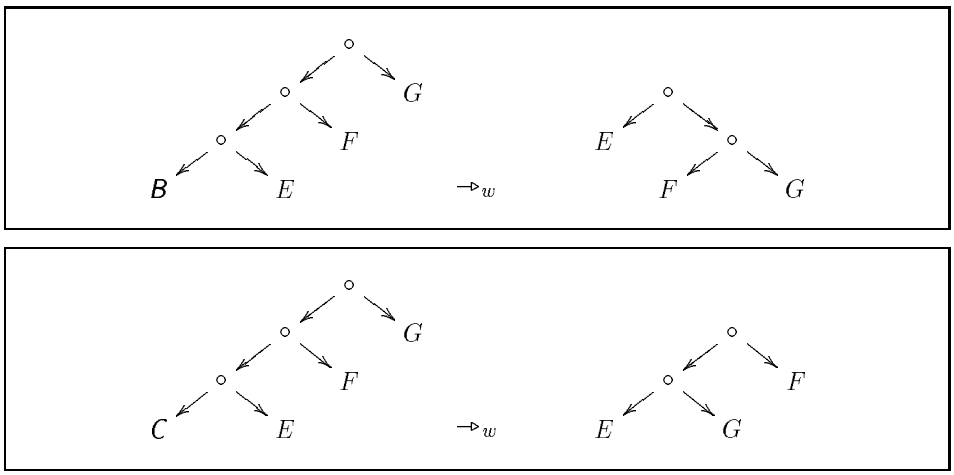
$$S(KE)FG \rightarrow_w KEG(FG) \rightarrow_w E(FG),$$

és

$$B\,E\,F\,G \rightarrow_w E(FG),$$

így

$$S(KE)FG = B\,E\,F\,G,$$



2.7. ábra. A B és C kombinátorok gráf-újraírási szabálya

és ebből az *ext* kiterjesztési axióma alapján éppen az első optimalizációt kapjuk meg. Megjegyezzük, hogy az első újraírási szabályból az *ext* axióma alapján

- $S\,(K\,E) \mapsto B\,E$.

Az újraírási szabályok felhasználásával azonnal látható, hogy a $CL^{(C)}$+*ext* kombinátor logikában

$$B \equiv S(KS)K,$$
$$C \equiv S(BBS)(KK)$$
$$\equiv S(S(K(S(KS)K))S)(KK).$$

A $\{K, S, B, C\}$ bázisú, bővített kombinátor logika $CL^{(C)}$ jelölése arra utal, hogy Curry javasolta azt, hogy a λ-kifejezések CL-kifejezésekre való alakításakor minden SEF kifejezés generálásakor vizsgáljuk meg a fenti újraírási szabályok alkalmazhatóságát. Ezt a módszert *Curry-algoritmusnak* nevezzük. Az algoritmus alkalmazásakor a CL-kifejezés hossza csak *négyzetesen* növekszik.

2.5.3. Példa. *(A Curry-algoritmus)*
A Curry-algoritmussal a 2.4.2. és 2.4.3. példák eredményei a következők:

(a) $\lambda_2^* x\,.\,yx(yy) \equiv$
$S\,(\lambda_2^* x\,.\,yx)(\lambda_2^* x\,.\,yy) \equiv$
$S\,y(\lambda_2^* x\,.\,yy) \equiv$
$S\,y(K(yy)) \mapsto C\,y(yy),$

(b) $\lambda_2^* yx\,.\,xy \equiv$
$\lambda_2^* y\,.\,(\lambda_2^* x\,.\,xy) \equiv$
$\lambda_2^* y\,.\,(S(\lambda_2^* x\,.\,x)(\lambda_2^* x\,.\,y)) \equiv$
$\lambda_2^* y\,.\,(S\,I(K\,y)) \mapsto$
$\lambda_2^* y\,.\,C\,I\,y \equiv$
$C\,I,$

(c) $\lambda_2^* xy\,.\,xy \equiv$
$\lambda_2^* x\,.\,\lambda_2^* y\,.\,xy \equiv$
$\lambda_2^* x\,.\,x \equiv$
$I.$ □

2.5.4. Példa. *(Az $f(xx)$ absztrakciója a Curry-algoritmussal)*
$\lambda_2^* f x\,.\,f(xx) \equiv \lambda_2^* f\,.\,\lambda_2^* x\,.\,f(xx) \equiv$
$\lambda_2^* f\,.\,S\,(\lambda_2^* x\,.\,f)(\lambda_2^* x\,.\,xx) \equiv$
$\lambda_2^* f\,.\,S\,(K\,f)(S\,(\lambda_2^* x\,.\,x)(\lambda_2^* x\,.\,x)) \equiv$
$\lambda_2^* f\,.\,S\,(K\,f)(S\,I\,I) \mapsto$
$\lambda_2^* f\,.\,B\,f(S\,I\,I) \equiv$
$S\,(\lambda_2^* f\,.\,B\,f)(\lambda_2^* f\,.\,S\,I\,I) \equiv$
$S\,B\,(K(S\,I\,I)) \mapsto$
$C\,B\,(S\,I\,I).$ □

Az optimalizációs lépések beépíthetők az absztrakciós algoritmusba is, a B és C kombinátorok felhasználásával a λ_2^* zárójeles absztrakció

a következőképpen módosítható. Az absztrakciót *Turner-absztrakciónak* nevezzük és λ_3^*-mal jelöljük. Ha egy n változót tartalmazó, m hosszú CL-kifejezésre a λ_3^* absztrakciós algoritmust használjuk, akkor $O(mn^2)$ hosszúságú kifejezést kapunk.

2.5.5. Definíció. A λ_3^* Turner-absztrakciós algoritmus:

A $\lambda_3^* x \,.\, E$ kifejezést az E szerkezete szerint definiáljuk:

$$
\begin{array}{lll}
\lambda_3^* x \,.\, x & \equiv \mathsf{I} & \\
\lambda_3^* x \,.\, E & \equiv \mathsf{K}E, & \textit{ha } x \notin FV(E), \\
\lambda_3^* x \,.\, Ex & \equiv E, & \textit{ha } x \notin FV(E), \\
\lambda_3^* x \,.\, EF & \equiv \mathsf{B}\,E(\lambda_3^* x \,.\, F), & \textit{ha } x \notin FV(E), \\
\lambda_3^* x \,.\, EF & \equiv \mathsf{C}\,(\lambda_3^* x \,.\, E)F, & \textit{ha } x \notin FV(F), \\
\lambda_3^* x \,.\, EF & \equiv \mathsf{S}\,(\lambda_3^* x \,.\, E)(\lambda_3^* x \,.\, F), & \textit{egyébként.}
\end{array}
$$

2.5.6. Példa. *(A λ_2^* absztrakció és az optimalizálás)*
Megmutatjuk, hogy a λ_3^* Turner-absztrakció 4. szabálya a λ_2^* absztrakciós algoritmusból és az újraírási szabályokból levezethető. Tegyük fel, hogy $x \notin FV(E)$. Ekkor

$$
\begin{array}{l}
\lambda_2^* x \,.\, EF \equiv \\
\mathsf{S}\,(\lambda_2^* x \,.\, E)(\lambda_2^* x \,.\, F) \equiv \\
\mathsf{S}\,(\mathsf{K}E)(\lambda_2^* x \,.\, F).
\end{array}
$$

Az $\mathsf{S}(\mathsf{K}E)F \mapsto \mathsf{B}\,EF$ újraírási szabályt alkalmazva

$$\mathsf{S}\,(\mathsf{K}E)(\lambda_2^* x \,.\, F) \mapsto \mathsf{B}\,E(\lambda_2^* x \,.\, F),$$

azaz éppen a λ_3^* negyedik szabályát kaptuk meg. □

2.5.7. Példa. *(A λ_3^* Turner-absztrakció)*
Ezzel az absztrakcióval a 2.4.2. és 2.4.3. példák eredményei a következők:

$$
\begin{array}{l}
\textit{(a)} \quad \lambda_3^* x \,.\, yx(yy) \equiv \\
\mathsf{C}\,(\lambda_3^* x \,.\, yx)(yy) \equiv
\end{array}
$$

$C\,y(yy)$,

(b) $\lambda^*_3 yx\,.\,xy \equiv$
$\lambda^*_3 y\,.\,\lambda^*_3 x\,.\,xy \equiv$
$\lambda^*_3 y\,.\,C(\lambda^*_3 x\,.\,x)y \equiv$
$\lambda^*_3 y\,.\,CIy \equiv$
CI,

(c) $\lambda^*_1 xy\,.\,xy \equiv$
$\lambda^*_1 x\,.\,\lambda^*_1 y\,.\,xy \equiv$
$\lambda^*_1 x\,.\,x \equiv$
I. □

2.5.8. Példa. *(Az $x(yy)$ Turner-absztrakciója)*
$\lambda^*_3 xy\,.\,x(yy) \equiv \lambda^*_3 x\,.\,\lambda^*_3 y\,.\,x(yy) \equiv$
$\lambda^*_3 x\,.\,Bx(\lambda^*_3 y\,.\,yy) \equiv$
$\lambda^*_3 x\,.\,Bx(S(\lambda^*_3 y\,.\,y)(\lambda^*_3 y\,.\,y)) \equiv$
$\lambda^*_3 x\,.\,Bx(SII) \equiv$
$C(\lambda^*_3 x\,.\,Bx)(SII) \equiv$
$CB(SII)$. □

2.5.9. Példa. *(Az Y kombinátor)*
Az előző példa eredményét felhasználva,

$\lambda^*_3 x\,.\,(\lambda^*_3 y\,.\,x(yy))(\lambda^*_3 y\,.\,x(yy)) \equiv$
$\lambda^*_3 x\,.\,(Bx(SII))(Bx(SII)) \equiv$
$S(\lambda^*_3 x\,.\,(Bx(SII)))(\lambda^*_3 x\,.\,(Bx(SII))) \equiv$
$S(C(\lambda^*_3 x\,.\,Bx)(SII))(C(\lambda^*_3 x\,.\,Bx)(SII)) \equiv$
$S(CB(SII))(CB(SII))$.

A 2.8.1. pontban majd látni fogjuk, hogy ez a kifejezés éppen az Y fixpont kombinátor CL-kifejezése. □

2.5.2. A $CL^{(T)}$ kombinátor logika és a λ_4^* absztrakció

A zárójeles absztrakciók algoritmusaiból látható, hogy az eredményül kapott CL-kifejezések hosszát lényegesen az applikációkra alkalmazott absztrakciók növelik, különösen igaz ez a többváltozós absztrakciókra.

Turner keresett egy olyan kombinátort, amivel a többváltozós absztrakcióval kapott CL-kifejezések hossza legfeljebb *négyzetesen* növekszik. Bevezette az új S' kombinátort, és megadta az S' gyenge redukcióját:

$$\mathsf{S}'C\ EFG\ \twoheadrightarrow_w\ C\ (EG)(FG), \qquad C \in \mathcal{C}^0.$$

Az S' gyenge redukcióját a következők indokolják.

$(\lambda_3^* x\,.\,CEF)G \equiv$
$\mathsf{S}(\lambda_3^* x\,.\,CE)(\lambda_3^* x\,.\,F)G \equiv$
$\mathsf{S}(\mathsf{S}(\lambda_3^* x\,.\,C)(\lambda_3^* x\,.\,E))(\lambda_3^* x\,.\,F)G \equiv$
$\mathsf{S}(\mathsf{S}(\mathsf{K}C)(\lambda_3^* x\,.\,E))(\lambda_3^* x\,.\,F)G\ \twoheadrightarrow_w$
$\mathsf{S}(\mathsf{K}C)(\lambda_3^* x\,.\,E)G((\lambda_3^* x\,.\,F)G)\ \twoheadrightarrow_w$
$\mathsf{K}CG((\lambda_3^* x\,.\,E)G)((\lambda_3^* x\,.\,F)G)\ \twoheadrightarrow_w$
$C((\lambda_3^* x\,.\,E)G)((\lambda_3^* x\,.\,F)G).$

Ugyanakkor az S' gyenge redukcióját alkalmazva

$\mathsf{S}'C(\lambda_3^* x\,.\,E)(\lambda_3^* x\,.\,F)G\ \twoheadrightarrow_w$
$C((\lambda_3^* x\,.\,E)G)((\lambda_3^* x\,.\,F)G),$

azaz

$$(\lambda_3^* x\,.\,CEF)G =_w \mathsf{S}'C(\lambda_3^* x\,.\,E)(\lambda_3^* x\,.\,F)G,$$

és az *ext* kiterjesztési axióma alapján

$$(\lambda_3^* x\,.\,CEF) =_w \mathsf{S}'C(\lambda_3^* x\,.\,E)(\lambda_3^* x\,.\,F),$$

látható tehát, hogy a CEF zárójeles absztrakciója átvihető az E és F absztrakciójába.

2.5.10. Példa. *(A $\lambda^*_3 xyzv\,.\,EF$ kifejezés)*

$\lambda^*_3 xyzv\,.\,EF \equiv$
$\lambda^*_3 x\,.\,\lambda^*_3 y\,.\,\lambda^*_3 z\,.\,\lambda^*_3 v\,.\,EF \equiv$
$\lambda^*_3 x\,.\,\lambda^*_3 y\,.\,\lambda^*_3 z\,.\,S(\lambda^*_3 v\,.\,E)(\lambda^*_3 v\,.\,F) =_w$
$\lambda^*_3 x\,.\,\lambda^*_3 y\,.\,S'S(\lambda^*_3 z\,.\,\lambda^*_3 v\,.\,E)(\lambda^*_3 z\,.\,\lambda^*_3 v\,.\,F) =_w$
$\lambda^*_3 x\,.\,S'(S'S)(\lambda^*_3 y\,.\,\lambda^*_3 z\,.\,\lambda^*_3 v\,.\,E)(\lambda^*_3 y\,.\,\lambda^*_3 z\,.\,\lambda^*_3 v\,.\,F) =_w$
$S'(S'(S'S))(\lambda^*_3 x\,.\,\lambda^*_3 y\,.\,\lambda^*_3 z\,.\,\lambda^*_3 v\,.\,E)(\lambda^*_3 x\,.\,\lambda^*_3 y\,.\,\lambda^*_3 z\,.\,\lambda^*_3 v\,.\,F) \equiv$
$S'(S'(S'S))(\lambda^*_3 xyzv\,.\,E)(\lambda^*_3 xyzv\,.\,F).$ □

Az B és C kombinátorokhoz hasonlóan vezessük még be a B' és C' kombinátorokat, az új kombinátor logikát $CL^{(T)}$-vel jelöljük, a T kitevő Turner nevére utal.

2.5.11. Definíció. A $CL^{(T)}$ kombinátor logika konstansai:

$< konstans > ::= K \mid S \mid B \mid C \mid S' \mid B' \mid C'$

Az új kombinátorokra vonatkozó *konverziós szabályok* legyenek a következők:

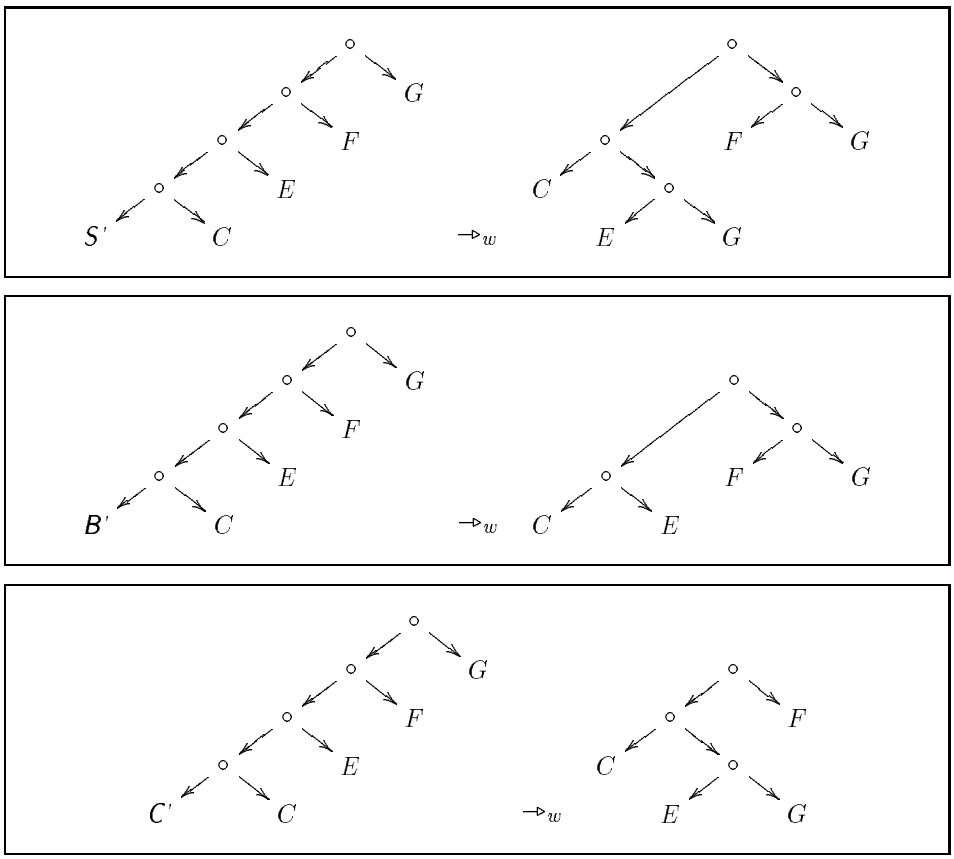
2.5.12. Definíció. A $CL^{(T)}$ kombinátor logika új gyenge redukciói:

$$\begin{aligned} &S'C\ EFG \to_w C\ (EG)(FG), && C \in \mathcal{C}^0,\\ &B'C\ EFG \to_w C\ E(FG), && C \in \mathcal{C}^0,\\ &C'C\ EFG \to_w C\ (EG)F, && C \in \mathcal{C}^0. \end{aligned}$$

A S' kombinátor az $S'C\ EF$ függvény G argumentumát „átadja” mind az E, mind az F függvénynek, a B' kombinátor a B kombinátorhoz hasonlóan függvény-kompozíciót végez, a C' pedig a C-hez hasonlóan, felcseréli az argumentumokat, vagy az E és F mindegyikét függvénynek tekintve, a G argumentumot az E-hez, az „eggyel balra levő” függvényhez viszi át (2.8. ábra).

Határozzunk meg néhány optimalizáló újraírási szabályt. Az

$S'C\ (KE)FG \to_w C\ (KEG)(FG) \to_w C\ E(FG),$
$B'C\ EFG \to_w C\ E(FG)$



2.8. ábra. Az S', B' és C' kombinátorok gráf-újraírási szabálya

és

$$S'C\ E(KF)G \rightarrow_w C\ (EG)(KFG) \rightarrow_w C\ (EG)F,$$
$$C'C\ EFG \rightarrow_w C\ (EG)F$$

egyenletekből a következő két optimalizáló újraírási szabályt kapjuk:

- $S'C\ (KE)F \mapsto B'C\ EF$,
- $S'C\ E(KI)F \mapsto C'C\ EF$.

Könnyen bizonyíthatók a következő szabályok is:

- $S\,(BC\ E)F \mapsto S'C\ EF$,
- $S\,(KC\ E)F \mapsto B'C\ EF$,
- $S\,(BC\ E)(KF) \mapsto C'C\ EF$,
- $B\,(C\ E)F \mapsto B'C\ EF$,
- $B\,EI \mapsto E$,
- $C\,(BC\ E)F \mapsto C'C\ EF$.

Az újraírási szabályok felhasználásával azonnal látható, hogy a $CL^{(T)}+ext$ kombinátor logikában

$S' \equiv B\,(B\,S)\,B$
$B' \equiv B\,B$
$C' \equiv B\,(B\,C)\,B$

Ha a λ^*_3 absztakcióba beépítjük az optimalizációkat, akkor a λ^*_4 absztrakciós algoritmust kapjuk meg.

2.5.13. Definíció. A λ^*_4 absztrakciós algoritmus:

A $\lambda^*_4 x\,.\,E$ kifejezést az E szerkezete szerint definiáljuk:

$$
\begin{array}{lll}
\lambda^*_4 x\,.\,x & \equiv I & \\
\lambda^*_4 x\,.\,E & \equiv KE, & \textit{ha } x \notin FV(E), \\
\lambda^*_4 x\,.\,Ex & \equiv E, & \textit{ha } x \notin FV(E), \\
\lambda^*_4 x\,.\,EFG & \equiv B'EF(\lambda^*_4 x\,.\,G), & \textit{ha } x \notin \{FV(E) \cup FV(F)\}, \\
\lambda^*_4 x\,.\,EFG & \equiv C'E(\lambda^*_4 x\,.\,F)G, & \textit{ha } x \notin \{FV(E) \cup FV(G)\}, \\
\lambda^*_4 x\,.\,EFG & \equiv S'E(\lambda^*_4 x\,.\,F)(\lambda^*_4 x\,.\,G), & \textit{ha } x \notin FV(E), \\
\lambda^*_4 x\,.\,EF & \equiv B\,E(\lambda^*_4 x\,.\,F), & \textit{ha } x \notin FV(E), \\
\lambda^*_4 x\,.\,EF & \equiv C(\lambda^*_4 x\,.\,E)F, & \textit{ha } x \notin FV(F), \\
\lambda^*_4 x\,.\,EF & \equiv S\,(\lambda^*_4 x\,.\,E)(\lambda^*_4 x\,.\,F), & \textit{egyébként.}
\end{array}
$$

Ha egy CL-kifejezés hossza m, és n változót tartalmaz, akkor a változókkal a λ_4^* absztrakciós algoritmust elvégezve $\Theta(mn)$ hosszú kifejezést kapunk.

Az S' kombinátor szerepét már láttuk, most megmutatjuk a B' és a C' hasznosságát.

2.5.14. Példa. *(A B' és a C' kombinátor)*
Ha az EF kifejezésben $x_i \notin FV(E)$ $(i = 1, \ldots, k)$, akkor

$$\lambda_4^* x_1 x_2 \ldots x_k \,.\, EF =_w$$
$$B'(B'(\ldots(B'B)\ldots))E(\lambda_4^* x_1 x_2 \ldots x_k \,.\, F),$$

ez utóbbi kifejezésben pontosan k darab B' kombinátor van. Látható tehát, hogy csak a második tagra kell az absztrakciókat elvégezni.

Hasonlóan, ha az EF kifejezésben $x_i \notin FV(F)$ $(i = 1, \ldots, k)$, akkor

$$\lambda_4^* x_1 x_2 \ldots x_k \,.\, EF =_w$$
$$C'(C'(\ldots(C'C)\ldots))(\lambda_4^* x_1 x_2 \ldots x_k \,.\, E)F. \qquad \square$$

2.5.3. További kombinátorok

A fentieken kívül, elsősorban a CL-kifejezések hosszának csökkentésére, még sok kombinátort definiáltak, amelyek adott, speciális felépítésű CL-kifejezések absztrakcióiban nagyon jól használhatók.

Ilyen például a J és a J' kombinátor, amit M. S. Joy vezetett be 1985-ben:

$$JEFG \quad \rightarrow\!\!\!\triangleright_w \quad EF,$$
$$J'EFGH \quad \rightarrow\!\!\!\triangleright_w \quad EFG.$$

A λ_4^* absztrakciós algoritmus ezzel a két új kombinátorral a következőképpen módosítható:

2.5.15. Definíció. A λ_J^* absztrakciós algoritmus:

A $\lambda^*_J x\,.\,E$ kifejezést az E szerkezete szerint definiáljuk:

$$
\begin{array}{lll}
\lambda^*_J x\,.\,x & \equiv I & \\
\lambda^*_J x\,.\,E & \equiv K\,E, & ha\ x \notin FV(E),\\
\lambda^*_J x\,.\,Ex & \equiv E, & ha\ x \notin FV(E),\\
\lambda^*_J x\,.\,EFG & \equiv J'\,EF(\lambda^*_J x\,.\,G), & ha\ x \notin \{FV(E) \cup FV(F) \cup FV(G)\},\\
\lambda^*_J x\,.\,EFG & \equiv B'\,EF(\lambda^*_J x\,.\,G), & ha\ x \notin \{FV(E) \cup FV(F)\},\\
\lambda^*_J x\,.\,EFG & \equiv C'\,E(\lambda^*_J x\,.\,F)G, & ha\ x \notin \{FV(E) \cup FV(G)\},\\
\lambda^*_J x\,.\,EFG & \equiv S'\,E(\lambda^*_J x\,.\,F)(\lambda^*_J x\,.\,G), & ha\ x \notin FV(E),\\
\lambda^*_J x\,.\,EF & \equiv J\,E(\lambda^*_J x\,.\,F), & ha\ x \notin FV(E) \cup FV(F),\\
\lambda^*_J x\,.\,EF & \equiv B\,E(\lambda^*_J x\,.\,F), & ha\ x \notin FV(E),\\
\lambda^*_J x\,.\,EF & \equiv C\,(\lambda^*_J x\,.\,E)F, & ha\ x \notin FV(F),\\
\lambda^*_J x\,.\,EF & \equiv S\,(\lambda^*_J x\,.\,E)(\lambda^*_4 x\,.\,F), & egyébként.
\end{array}
$$

Joy eredeti absztrakciójában nem szerepelt a K kombinátor, mivel

- $JIEF \mapsto E$,

és ezért Joy a fenti absztrakciós algoritmus második sorát a

$$\lambda^*_J x\,.\,E \quad \equiv \quad JIE, \quad ha\ x \notin FV(E)$$

alakban adta meg.

Ha egy CL-kifejezés hossza m, és n változót tartalmaz, akkor a változókkal a λ^*_J absztrakciós algoritmust elvégezve $O(m(2n+1))$ hosszú kifejezést kapunk.

J. B. Rosser is definiált 1935-ben egy J nevű kombinátort:

$$JEFGH \twoheadrightarrow_w EF(EHG)$$

Ennek a J kombinátornak egy fontos tulajdonsága az, hogy az $\{I, J\}$ halmaz a kombinátor logika egy bázisát (2.6. pont) adja. Megjegyezzük, hogy ebben a bázisban például az S kombinátor kifejezése a zárójeleket nem számítva 266 szimbólumot, azaz összesen 266 darab I és J kombinátort tartalmaz.

Megemlítjük még a W kombinátort, amit *duplikátor* kombinátornak

nevezünk:

$W\,E\,F \;\twoheadrightarrow_w\; E\,F\,F$

A W kombinátorra

$W \equiv S\,S\,(K\,I)$
$\equiv S\,S\,(S\,K)$
$\equiv C\,S\,I,$

és például

- $S\,E\,I \mapsto W\,E.$

Szintén J. B. Rosser vezette be a C_* kombinátort, ami a

$C_*\,E\,F \;\twoheadrightarrow_w\; F\,E$

transzformációt végzi. Ezzel a kombinátorral

$C_* \equiv J\,I\,I,$
$C_* \equiv C\,I,$
$C \equiv J\,C_*\,(J\,C_*)\,(J\,C_*),$
$B \equiv C\,(J\,I\,C)\,(J\,I),$
$W \equiv C\,(C\,(B\,C\,(C\,(B\,J\,C_*)\,C_*))\,C_*).$

1986-ban M. Sheevel definiálta a B^* kombinátort, melyre

$B^*\,C\,E\,F\,G \;\twoheadrightarrow_w\; C(E(F\,G)).$

A λ_J^* absztrakciós algoritmusba ekkor a

$\lambda_J^* x\,.\,E(F\,G) \equiv B^*\,E\,F(\lambda_J^* x\,.\,G), \quad ha \quad x \notin \{FV(E) \cup FV(F)\}$

építhető be.

Erre a kombinátorra két optimalizáló újraírási szabály használható:

- $S(KC)(BEF) \mapsto B^*CEF$
- $BC(BEF) \quad \mapsto B^*CEF$

2.6. Bázis

A 2.1.1. definíció szerint az egyszerű kombinátor logika kifejezései csak a K és S konstansokat tartalmazhatják, és ezeket a kombinátorokat az egyszerű kombinátor logika bázisának neveztük.

2.6.1. Definíció. Az egyszerű kombinátor logika kifejezéseinek bázisa:

Legyen $\mathcal{C}_b \subset \mathcal{C}$. A $\mathcal{C}_b$ halmaz a $\mathcal{C}' \subset \mathcal{C}$ halmaz bázisa, ha minden $E \in \mathcal{C}'$-re van olyan $F \in (\mathcal{C}_b)^+$, melyre $E =_w F$.
A $\mathcal{C}^0$ halmaz bázisát a kombinátor logika bázisának nevezzük.

A fentiek szerint tehát az egyszerű kombinátor logika bázisa a két elemű $\{K, S\}$ halmaz, hiszen minden $E \in \mathcal{C}^0$-ra $E \in \{K, S\}^+$. Hasonlóan, a $CL^{(C)}$ kombinátor logika bázisa a $\{K, S, B, C\}$ halmaz, a $CL^{(T)}$ kombinátor logika bázisa pedig $\{K, S, B, C, S', B', C'\}$.

Láttuk, hogy ezeknek a bázisoknak az elemei kifejezhetők a K és S kombinátorokkal, így ezekhez a kombinátor logikákhoz is választható a $\{K, S\}$ halmaz bázisként.

M. Schönfinkel 1924-ben a kombinátor logika bázisának a $\{B, C, I, K, S\}$ halmazt választotta, bár a bázis kombinátorait nem így, hanem rendre Z, T, I, C, S-nek nevezte. Azt is megmutatta, hogy a B, C és I kombinátorok kifejezhetők a K és S kombinátorokkal.

Haskell B. Curry 1927-ben, Schönfinkel dolgozatainak ismerete nélkül, a kombinátor logika bázisának a $\{B, C, W, I\}$ halmazt határozta meg. Amikor Schönfinkel munkáit megismerte, a K kombinátorral azonnal kibővítette a rendszerét, de az S kombinátornak nem sok jelentőséget tulajdonított. Az S kombinátor igazi szerepét a zárójeles absztrakciók kidolgozása hozta meg. Az S kombinátor ebben a rendszerben a következő:

$S \equiv B(BW)(BBC)$.

A három bázis közötti kapcsolatot a 2.9. ábra mutatja.

	$\{K,S\}$	*Schönfinkel* $\{B,C,I,K,S\}$	*Curry* $\{B,C,W,I,K\}$
K	K	K	K
S	S	S	$B(BW)(BBC)$
I	SKK	I	I
B	$S(KS)K$	B	B
C	$S(S(KS)(S(KK)S))(KK)$	C	C
W	$SS(SK)$	CSI	W

2.9. ábra. Bázisok összehasonlító táblázata

Most megmutatjuk, hogy van olyan CL-kifejezés, amellyel mind a K, mind az S leírható, azaz a kombinátor logikának van egyelemű bázisa is, sőt, mint látni fogjuk, több ilyen egyelemű bázis is létezik. Mivel az egyelemű bázisokkal is leírhatóak a különböző kombinátor logikák kombinátorai, megállapíthatjuk, hogy a különböző kombinátor logikák között csak *szintaktikai* különbség van. Így tehát a továbbiakban elegendő csak egy kiválasztott bázisú kombinátor logikával foglalkoznunk, ez a bázis legyen az $\{S, K\}$ halmaz. A kifejezések egyszerűbb leírására ezt a halmazt még kibővíthetjük az I kombinátorral.

Legyen

$$X_1 \equiv S(S(SI(KK))(KS))(KK),$$

és megmutatjuk, hogy $K = X_1X_1X_1$ és $S = X_1(X_1X_1)$, azaz az $\{X_1\}$ egy egyelemű bázis.

Az egyszerűbb számolás érdekében optimalizáljuk a X_1 kifejezést:

$$X_1 \equiv S(S(\underline{SI(KK)})(KS))(KK) \mapsto$$
$$S(\underline{S(CIK)(KS)})(KK) \mapsto$$
$$\underline{S(C(CIK)S)(KK)} \mapsto$$

$C(C(CIK)S)K$.

Ezzel a kifejezéssel

$X_1X_1 \equiv \underline{C(C(CIK)S)KX_1} \rightarrow_w$
$\underline{C(CIK)SX_1\,K} \rightarrow_w$
$\underline{CIKX_1\,SK} \rightarrow_w$
$\underline{IX_1\,KSK} \rightarrow_w$
$X_1\,KSK \equiv$
$\underline{C(C(CIK)S)KKSK} \rightarrow_w$
$\underline{C(CIK)SKKSK} \rightarrow_w$
$\underline{CIKKSKSK} \rightarrow_w$
$\underline{IK}KSKSK \rightarrow_w$
$\underline{KKSKSK} \rightarrow_w$
$\underline{KKSK} \rightarrow_w$
KK,

így

$X_1X_1X_1 \equiv$
$\underline{KKX_1} \rightarrow_w$
K,

és

$X_1(X_1\,X_1) \equiv$
$\underline{C(C(CIK)S)K(KK)} \rightarrow_w$
$\underline{C(CIK)S(KK)K} \rightarrow_w$
$\underline{CIK(KK)SK} \rightarrow_w$
$\underline{I(KK)KSK} \rightarrow_w$
$\underline{KKKSK} \rightarrow_w$
$\underline{KSK} \rightarrow_w$

S.

Könnyen belátható, hogy a következő X_2, X_3, X_4 és X_5 CL-kifejezések is egyelemű bázisokat alkotnak:

$X_2 \equiv S(S\,I(K^3\,K))(K\,K))(K^2\,S)$,

amelyre $X_2 X_2 \twoheadrightarrow_w^+ S$ és $X_2 S \twoheadrightarrow_w^+ K$,

$X_3 \equiv S(S\,I(S\,(S\,I(K\,S))(K^4\,I)))(K\,K)$,

amelyre $X_3 X_3 X_3 \twoheadrightarrow_w^+ K$ és $X_3 K \twoheadrightarrow_w^+ S$,

$X_4 \equiv S(S\,I(S\,(S\,I(K\,S))(K^2\,K)))(K\,K)$,

amelyre $X_4 X_4 X_4 \twoheadrightarrow_w^+ K$ és $X_4 K \twoheadrightarrow_w^+ S$,

$X_5 \equiv C(C\,I\,S)K$,

amelyre $X_5^3 X_5 \twoheadrightarrow_w^+ K$ és $X_5^4 X_5 \twoheadrightarrow_w^+ S$.

2.7. Konstansok és konstans függvények

A λ-kalkulushoz hasonlóan, a kombinátor logikában is van arra lehetőség, hogy konstansokat, konstans függvényeket definiáljunk, az egyszerű típusnélküli kombinátor logika egyes CL-kifejezéseit konstansoknak és a konstansokon értelmezett függvényeknek tekinthetjük. Ezekre a CL-kifejezésekre a szokásos aritmetikai, logikai tulajdonságok is teljesülnek.

2.7.1. A logikai konstansok és műveletek

A logikai konstansoknak a következő CL-kifejezéseket feleltethetjük meg:

$\mathit{true} \equiv K$,

$\mathit{false} \equiv K\,I$.

Az *if* nevű *if-then-else* CL-kifejezést az

if $\equiv EFG$

alakban definiálhatjuk, ahol E a *true* vagy *false* értékű feltétel, F és G pedig a *then-ág* és az *else-ág* CL-kifejezései.

2.7.1. Példa. *(Az* if true EF *applikáció az* E *kifejezésre, az* if false EF *pedig az* F*-re redukálható)*

if true $EF \equiv$
true $EF \equiv$
$\underline{KEF} \twoheadrightarrow_w E$,

if false $EF \equiv$
false $EF \equiv$
$\underline{KIEF} \twoheadrightarrow_w$
$\underline{IF} \twoheadrightarrow_w F.$ □

Használva az imperatív nyelvekből jól ismert *optimalizált kiértékelés* módszerét, meghatározzuk a logikai függvények CL-kifejezéseit.

and $EF \approx$ *if* EF *false* $\equiv$
EF *false*,

or $EF \approx$ *if* E *true* $F \equiv$
E *true* F,

és hasonlóan

not $E \approx$ *if* E *false true* $\equiv$
E *false true*.

Ezekből pedig például a λ_4^* zárójeles absztrakcióval

and $\equiv \lambda_4^* yx.xy$ *false* $\equiv \ldots \equiv$
$C'C(CI)(KI)$,

or $\equiv \lambda_4^* xy.x$ *true* $y \equiv \ldots \equiv$

CIK,

$not \equiv \lambda_4^* x.x\ false\ true \equiv \ldots \equiv$
$C(CI(KI))K$.

2.7.2. Példa. *(and false true)*

$and\ false\ true \equiv$
$\underline{C'C(CI)(KI)(KI)}K \rightarrow_w$
$\underline{C(CI(KI))(KI)}K \rightarrow_w$
$\underline{CI(KI)K}(KI) \rightarrow_w$
$\underline{IK}(KI)(KI) \rightarrow_w$
$\underline{K(KI)(KI)} \rightarrow_w$
KI. □

2.7.3. Példa. *(not false)*

$not\ false \equiv \underline{C(CI(KI))K(KI)} \rightarrow_w$
$\underline{CI(KI)(KI)}K \rightarrow_w$
$I(KI)(KI)K \rightarrow_w$
$\underline{I(KI)}(KI)K \rightarrow_w$
$\underline{KI(KI)}K \rightarrow_w$
$\underline{IK} \rightarrow_w$
K □

2.7.2. Listák és listákon értelmezett műveletek

A *láncolt lista* adatszerkezetkonstruktorának, a fejelem és maradék rész szelektoroknak a következő CL-kifejezéseket feleltethetjük meg:

$$\begin{aligned} cons &\equiv BC(CI), \\ head &\equiv CIK, \\ tail &\equiv CI(KI). \end{aligned}$$

2.7.4. Példa. *(Az $\{E_3, E_2, E_1\}$ lista fejeleme)*

A lista konstruktor CL-kifejezését felhasználva az $\{E_3, E_2, E_1\}$ lista a következő:

$\mathsf{cons}\, E_3(\mathsf{cons}\ E_2E_1) \equiv$
$\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_3(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1),$

és ennek fejeleme

$\underline{\mathsf{C}\,\mathsf{I}\mathsf{K}(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_3(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1))} \rightarrow_w$
$\underline{\mathsf{I}(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_3(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1))\mathsf{K}} \rightarrow_w$
$\underline{\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_3}(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1)\mathsf{K} \rightarrow_w$
$\underline{\mathsf{C}(\mathsf{C}\,\mathsf{I}E_3)(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1)\mathsf{K}} \rightarrow_w$
$\underline{\mathsf{C}\,\mathsf{I}E_3\mathsf{K}}(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1) \rightarrow_w$
$\underline{\mathsf{I}\mathsf{K}}E_3(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1) \rightarrow_w$
$\underline{\mathsf{K}E_3(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1)} \rightarrow_w$
$E_3.$ □

2.7.5. Példa. *(Az $\{E_3, E_2, E_1\}$ lista maradék része)*

Az előző példában is szereplő lista maradékrésze a következő:

$\underline{\mathsf{C}\,\mathsf{I}(\mathsf{K}\,\mathsf{I})(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_3(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1))} \rightarrow_w$
$\underline{\mathsf{I}(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_3(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1))}(\mathsf{K}\,\mathsf{I}) \rightarrow_w$
$\underline{\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_3}(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1)(\mathsf{K}\,\mathsf{I}) \rightarrow_w$
$\underline{\mathsf{C}(\mathsf{C}\,\mathsf{I}E_3)(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1)(\mathsf{K}\,\mathsf{I})} \rightarrow_w$
$\underline{\mathsf{C}\,\mathsf{I}E_3(\mathsf{K}\,\mathsf{I})}(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1) \rightarrow_w$
$\underline{\mathsf{I}(\mathsf{K}\,\mathsf{I})}E_3(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1) \rightarrow_w$
$\underline{\mathsf{K}\,\mathsf{I}E_3}(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1) \rightarrow_w$
$\underline{\mathsf{I}(\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1)} \rightarrow_w$
$\mathsf{B}\,\mathsf{C}(\mathsf{C}\,\mathsf{I})E_2E_1 \equiv$
$\mathsf{cons}\, E_2E_1 \equiv \{E_2, E_1\}.$ □

2.7.3. A számkonstansok és aritmetikai műveletek

A kombinátor logikában is könnyen definiálható egy normált, adekvát számjegyendszer. Legyen

$$\begin{aligned} \mathsf{c_0} &\equiv \mathsf{KI}, \\ \mathsf{succ} &\equiv \mathsf{SB}, \\ \mathsf{zero} &\equiv \mathsf{C(CI(true\ false))true}, \end{aligned}$$

de a *pred* egy kissé bonyolult:

$$\mathsf{pred} \equiv \mathsf{C(C(CI(S(BC(CI))(SB)(CI)(KI)))(C(CI(KI))(KI)))K}.$$

Az n szám CL-kifejezését jelöljük $\mathsf{c_n}$-nel, vagy röviden $\overline{n}$-nel.

2.7.6. Példa. *(Ebben a számjegyrendszerben a számok tehát a következők)*

$$\begin{aligned} \overline{0} &\equiv && \mathsf{KI}, \\ \overline{1} &\equiv \mathsf{succ}\ \ulcorner 0 \urcorner \equiv && \mathsf{SB(KI)}, \\ \overline{2} &\equiv \mathsf{succ}\ \ulcorner 1 \urcorner \equiv && (\mathsf{SB})^2(\mathsf{KI}), \\ \overline{3} &\equiv \mathsf{succ}\ \ulcorner 2 \urcorner \equiv && (\mathsf{SB})^3(\mathsf{KI}), \\ \ldots & && \ldots \end{aligned}$$

□

A természetes számok fenti kombinátorait *iterátoroknak* is nevezhetjük, mivel a $\mathsf{c_n}EF$ CL-kifejezés normál formája egy olyan $E(E(\ldots(E)\ldots))F \equiv E^nF$ CL-kifejezés, amelyben az E pontosan n-szer szerepel, azaz az E-t pontosan n-szer alkalmazzuk az F kifejezésre.

2.7.7. Példa. *(A $\mathsf{c_2}$ EF kifejezés)*

$$\begin{aligned} &\mathsf{c_2}\ EF \equiv \\ &(\mathsf{SB})^2(\mathsf{KI}) \equiv \\ &\underline{\mathsf{SB(SB(KI))}EF} \ \rightarrow_w \\ &\underline{\mathsf{BE(SB(KI)E)}F} \ \rightarrow_w \end{aligned}$$

$E(\underline{SB(KI)}EF) \rightarrow_w$
$E(B\,E(\underline{KIE})F) \rightarrow_w$
$E(\underline{B\,E\,I\,F}) \rightarrow_w$
$E(E(\underline{IF})) \rightarrow_w$
$E(EF)$. □

2.7.8. Példa. *(zero $c_0 \rightarrow_w^+$ true és zero $c_2 \rightarrow_w^+$ false)*

$zero\ c_0 \equiv$
$\underline{C\,(C\,I\,(true\ false))\,true\ c_0} \rightarrow_w$
$\underline{C\,I\,(true\ false)\,c_0}\,true \rightarrow_w$
$\underline{I\,c_0}\,(true\ false)\,true \rightarrow_w$
$c_0\,(true\ false)\,true \equiv$
$\underline{K\,I\,(true\ false)}\,true \rightarrow_w$
$\underline{I\,true} \rightarrow_w$
$true$,

és

$zero\ c_2 \equiv$
$\underline{C\,(C\,I\,(true\ false))\,true\ c_2} \rightarrow_w$
$\underline{C\,I\,(true\ false)\,c_2}\,true \rightarrow_w$
$\underline{I\,c_2}\,(true\ false)\,true \rightarrow_w$
$c_2\,(true\ false)\,true \equiv$
$\underline{S\,B\,(S\,B\,(K\,I))\,(true\ false)}\,true \rightarrow_w$
$\underline{B\,(true\ false)\,(S\,B\,(K\,I)\,(true\ false))\,true} \rightarrow_w$
$true\ false\,(S\,B\,(K\,I)\,(true\ false)\,true) \equiv$
$\underline{K\,false\,(S\,B\,(K\,I)\,(true\ false)\,true)} \rightarrow_w$
$false$. □

Hasonlóan a λ-kalkulushoz, a kombinátor logikában is megadhatók az összeadás, szorzás és hatványozás műveletek CL-kifejezései. Ha c_i és c_j a

műveletek két operandusa, $\mathsf{exp}\,\mathsf{c}_i\mathsf{c}_j$ a $c_i^{c_j}$ értékét jelöli, akkor

$\mathsf{add}\,\mathsf{c}_i\mathsf{c}_j \equiv \mathsf{S'B}\,\mathsf{c}_i\mathsf{c}_j$,
$\mathsf{mul}\,\mathsf{c}_i\mathsf{c}_j \equiv \mathsf{B}\,\mathsf{c}_i\mathsf{c}_j$,
$\mathsf{exp}\;\;\mathsf{c}_i\mathsf{c}_j \equiv \mathsf{c}_j\mathsf{c}_i$.

Könnyen belátható, hogy a természetes számoknak megfeleltetett CL-kifejezésekre ezek a műveletek éppen a megfelelő aritmetikai művelet eredményének megfelelő szám CL-kifejezését adják.

2.7.9. Példa. *(A c_i és a c_j összeadásának eredménye c_{i+j})*

$\mathsf{add}\;\;\mathsf{c}_i\mathsf{c}_j \equiv \mathsf{S'B}\,((\mathsf{SB})^i(\mathsf{KI}))((\mathsf{SB})^j(\mathsf{KI}))$,

és ha ennek a kifejezésnek az E és az F a két operandusa, akkor eredményül egy olyan kifejezést kapunk, amelyben az F-re az E-t pontosan $i+j$-szer applikáljuk.

$\underline{\mathsf{S'B}((\mathsf{SB})^i(\mathsf{KI}))((\mathsf{SB})^j(\mathsf{KI}))EF} \;\rightarrow_w$
$\underline{\mathsf{B}((\mathsf{SB})^i(\mathsf{KI})E)((\mathsf{SB})^j(\mathsf{KI})E)F} \;\rightarrow_w$
$((\mathsf{SB})^i(\mathsf{KI})E)\underline{((\mathsf{SB})^j(\mathsf{KI})EF)} \;\rightarrow_w^+$
$\underline{((\mathsf{SB})^i(\mathsf{KI})E)(E^jF)} \;\rightarrow_w^+$
$E^i(E^jF) \equiv$
$E^{i+j}F.$ □

2.7.4. A konstansos kombinátor logika

Ha az egyszerű kombinátor logikát kibővítjük a most definiált konstansokkal és a konstansokon értelmezett műveletekkel, akkor *konstansos kombinátor logikáról* beszélünk. Az 2.1.1. definícióban meghatározott egyszerű CL-kifejezések tehát a következőképpen bővíthetők:

2.7.10. Definíció. A konstansos kombinátor logika kifejezései:

Tegyük fel, hogy adott egy nem feltétlenül véges, egymástól páronként különböző változókat (szimbólumokat), a K *és* S *konstansokat, a logikai és számkonstansokat, az ezeken értelmezett műveleteket, valamint a nyitó- és csukózárójelet tartalmazó halmaz. Ezen a halmazon, mint ábécén értelmezett kombinátor logikai kifejezések a következő szavak:*

$$
\begin{array}{lcl}
< \textit{kifejezés} > & ::= & < \textit{változó} > \\
 & | & < \textit{konstans} > \\
 & | & < \textit{applikáció} > \\
< \textit{konstans} > & ::= & \mathsf{K} \mid \mathsf{S} \\
 & | & < \textit{számkonstans} > \\
 & | & < \textit{logikai konstans} > \\
 & | & < \textit{művelet} > \\
< \textit{applikáció} > & ::= & (\, < \textit{kifejezés} > < \textit{kifejezés} > \,)
\end{array}
$$

A CL-kifejezések könnyebben és kényelmesebben írhatók és olvashatók, ha a kombinátor logika beépített, előre definiált konstansokat, és az ezeken a konstansokon értelmezett előre definiált függvényeket tartalmaz. Ezért a továbbiakban a számkonstansokat leíró CL-kifejezéseket a reprezentált számkonstanssal, a függvényeket a szokásos műveleti jelekkel jelölhetjük, tehát például $\mathsf{c_0}$ helyett 0-t, az add helyett a + jelet is írhatjuk.

2.7.11. Példa. *(Matematikai kifejezések a konstansos kombinátor logikában)*

Felhasználva a zárójeles absztrakció applikációjának azt a tulajdonságát, hogy (2.4.5. tétel)

$$(\lambda^* x \,.\, E)F \;\twoheadrightarrow_w^+\; E[x := F],$$

a konstansos kombinátor logika lehetőséget ad a matemetikai kifejezések CL-kifejezésekkel történő egyszerű leírására.

mat. kif.	*funk. kif.*	*CL-kifejezés*
1	1	$K1$
x	x	I
$x+1$	$+\ x\ 1$	$C+\ 1$
$1+x$	$+\ 1\ x$	$+\ 1$
$x+y$	$+\ x\ y$	$+$
$2x$	$*\ 2\ x$	$B(*\ 2\ I) \quad \mapsto \quad *\ 2$
x^2	$*\ x\ x$	$S\ *\ I \quad \mapsto \quad W*$
x^2+2x+1	$+\ (\ +\ (*\ x\ x)(*\ 2\ x))\ 1$	$C'+(S'+\ (W*)(\ *\ 2))\ 1$

□

2.8. Rekurzió, rekurzív függvények

Az **??**. pontban láttuk, hogy a λ-kalkulusban a rekurzív függvények az Y fixpont kombinátor felhasználásával rekurzió nélkül leírhatók. Hasonló mondható el a kombinátor logikára is, ezzel foglakozunk a következőkben.

2.8.1. A fixpont kombinátor

Könnyen belátható, hogy egy E CL-kifejezés fixpontját az

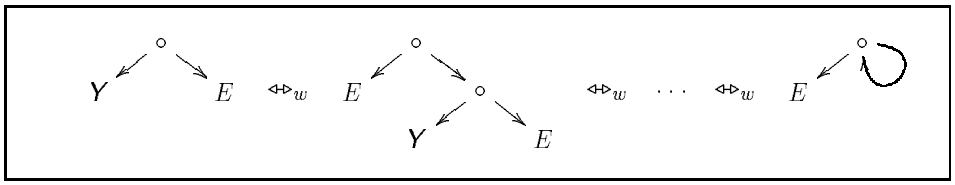
$Y \equiv S(CB(SII))(CB(SII))$

kombinátor előállítja:

$YE \equiv \underline{S(CB(SII))(CB(SII))E} \rightarrow_w$
$\underline{CB(SII)E}(CB(SII)E) \rightarrow_w$
$\underline{BE(SII)(CB(SII)E)} \rightarrow_w$
$E(\underline{SII(CB(SII)E)}) \rightarrow_w$
$E(\underline{I(CB(SII)E)}(I(CB(SII)E))) \rightarrow_w$
$E(CB(SII)E(\underline{I(CB(SII)E)})) \rightarrow_w$
$E(\underline{CB(SII)}\underline{E}(\underline{CB(SII)}\underline{E})) \leftarrow_w$

$E(S(CB(SII))(CB(SII))E) \equiv$
$E(YE)$.

Ezt a kombinátort H. B. Curry vezette le a Russel paradoxonból (lásd 2.8.2. pont), a redukció gráf-újraírási szabálya a 2.10.. ábrán látható.



2.10. ábra. Az Y kombinátor gráf-újraírási szabálya

Az Y kombinátor CL-kifejezése a K és S kombinátorokkal a következő:

$$\begin{aligned} Y \quad \equiv \quad & S(S(S(KS)(S(KK)I))(K(SII))) \\ & (S(S(KS)(S(KK)I))(K(SII))). \end{aligned}$$

A λ-kalkulusbeli Θ Turing fixpont kombinátor megfelelője a következő CL-kifejezés:

$\Theta \equiv (B(SI)(SII))(B(SI)(SII))$,

erre a fixpont kombinátorra az is teljesül, hogy

$\Theta E \twoheadrightarrow_w^+ E(\Theta E)$.

A λ-kalkulushoz hasonlóan, az Y kombinátort is felhasználhatjuk a rekurzív függvények rekurzió nélküli leírására.

2.8.1. Példa. *(A fac függvény leírása)*
Az **??**. pontban láttuk, hogy a

$fac(n) = if\ (n = 0)\ then\ 1\ else\ (n * fac(n-1))$

függvény definícióját a λ-kalkulus jelöléseivel átírva

$\mathsf{fac} \equiv \lambda n\,.\,\mathsf{if}\ (=\ n\ 0)\ 1\ (*\ n\ (\mathsf{fac}\ (-\ n\ 1)))$,

és ebből

$\mathsf{fac} \equiv \mathsf{Y}\ (\lambda f n\,.\,\mathsf{if}\ (=\ n\ 0)\ 1\ (*\ n\ f(-\ n\ 1)))$.

A zárójelben levő kifejezésre a kombinátor logika λ_4^* zárójeles absztrakcióját végrehajtva a

$\mathsf{fac} \equiv \mathsf{Y}(\mathsf{B}\,(\mathsf{S}\,(\mathsf{C'if}(\mathsf{C}\ -\ 1)\ 1))(\mathsf{B}\,(\mathsf{S}*)(\mathsf{C}\,\mathsf{B}\,(\mathsf{C}\ -\ 1))))$

kifejezést kapjuk. □

2.8.2. A Russel-paradoxon

A **??**. fejezetben láttuk, hogy a *Russel-paradoxon* leírható a λ-kalkulusban, most megmutatjuk, hogy a paradoxon leírható a kombinátor logikában is. Megadjuk paradoxon CL-kifejezését, és az Y fixpont kombinátort most is a paradoxonból vezetjük le.

Jelölje az $x \in E$ relációt az Ex kifejezés, melyre

$$Ex = \begin{cases} \mathsf{true}, & \text{ha } x \in B, \\ \mathsf{false}, & \text{ha } x \notin B. \end{cases}$$

Ha $A = \{x \mid x \notin x\}$, azaz A a „rendes” halmazok halmaza, akkor az $x \in A$ és az $x \notin x$ relációnak az

$Ax =_w \mathsf{not}(xx)$

CL-kifejezés felel meg, amelyből

$Ax =_w \underline{\mathsf{not}}(\underline{xx}) \leftarrow_w$
$\underline{\mathsf{B\,not}}\ \underline{x}\ \underline{x} \leftarrow_w$
$\mathsf{W}(\mathsf{B\,not})x$.

Tehát a kiterjesztési axióma alapján

$A =_w W(B\,\mathsf{not})$,

és ebből

$A \leftarrow_w B\,W\,B\,\mathsf{not}.$

Ha azt vizsgáljuk, hogy az A halmaz „rendes" halmaz-e, akkor most is az

$AA \twoheadrightarrow^+_w \mathsf{not}\,(AA)$

kifejezést kapjuk, amiből látható, hogy az AA a **not** fixpontja. Ugyanakkor

$AA \equiv \underline{B\,W\,B\,\mathsf{not}(B\,W\,B\,\mathsf{not})} \leftarrow_w$
$\underline{S(B\,W\,B)(B\,W\,B)\,\mathsf{not}} \leftarrow_w$
$W\,S\,(B\,W\,B)\,\mathsf{not}$,

és ha a $W\,S\,(B\,W\,B)$ kifejezést Y-nal jelöljük, akkor azt kapjuk, hogy

$Y\,\mathsf{not} \twoheadrightarrow^+_w AA$,

azaz az AA-t az $Y\,\mathsf{not}$ határozza meg. Mivel az AA a **not** fixpontja, az

$Y \equiv W\,S\,(B\,W\,B)$

kifejezés egy *fixpont kombinátor.*

Az Y fixpont-kombinátor tehát a Russel-paradoxonból levezethető, és ezért nevezte Curry az Y-t paradoxon-kombinátornak.

Könnyen belátható, hogy ez a fixpont kombinátor a korábban megadott

$Y \equiv S\,(C\,B\,(S\,I\,I))(C\,B\,(S\,I\,I))$

kombinátorral ekvivalens.

2.8.3. *CL*-definiálható függvények

Ebben a pontban függvényen az $f : \mathbb{N}^n \to \mathbb{N}$ leképezést értjük, az n és m betűkkel a természetes számokat jelöljük.

Az **??** pontban foglakoztunk a λ-definiálhatósággal, most azt vizsgáljuk meg, hogy a kombinátor logikában hogyan definiálhatók a függvények. Természetesen nem meglepő, azt az eredményt kapjuk, mint a λ-kalkulusban, a parciális rekurzív függvények osztálya megegyezik a *CL*-definiálható függvények osztályával.

2.8.2. Definíció. *CL*-definiálható függvény:

Azt mondjuk, hogy az f n-változós függvény CL-definiálható, vagy kombinátorosan definiálható, ha ezt a függvényt az $F \in \mathcal{C}^0$ kombinátor reprezentálja, azaz minden $m_1, m_2, \ldots m_n$ -re

$F\ \overline{m_1}\ \overline{m_2} \ldots \overline{m_n} = \overline{f(m_1, m_2, \ldots, m_n)}$.

Mivel a számjegyrendszer normált számjegyrendszer, ezért feltehetjük, hogy a fenti definícióban szereplő számjegyek normál formában vannak, azaz minden szám *CL*-kifejezésének, így a jobboldalon álló kifejezésnek is van normál formája.

Most megmutatjuk, hogy a *primitív rekurzív függvények* (**??**. definíció) *CL*-definiálhatók.

1. A $Z = 0$ nullváltozós nulla értékű konstans függvény *CL*-kifejezése:

   $\overline{0} \equiv \mathsf{K}\mathsf{I}$.

2. A $succ(x) = x + 1$ *egységgel növelő függvény* *CL*-kifejezése:

   $\mathsf{succ} \equiv \mathsf{S}\,\mathsf{B}$.

3. Az $U_i^n(m_1, m_2, \ldots, m_n) = m_i \ \ (1 \leq i \leq n)$ *szelektor függvény* *CL*-kifejezése:

   $\lambda^* m_1 m_2 \ldots m_n \,.\, m_i$.

4. Ha a $g, h_1, \ldots, h_p$ primitív rekurzív függvények CL-kifejezései rendre $G, H_1, \ldots, H_p$, akkor a *helyettesítéssel* definiálható

   $f(m_1, m_2, \ldots, m_n) =$
   $g(h_1(m_1, m_2, \ldots, m_n), h_2(m_1, m_2, \ldots, m_n), \ldots, h_p(m_1, m_2, \ldots, m_n))$

   primitív rekurzív függvény CL-kifejezése:

   $F \equiv \lambda^* m_1 m_2 \ldots m_n \,.\, G(H_1 m_1 m_2 \ldots m_n) \ldots (H_p m_1 m_2 \ldots m_n).$

5. A primitív rekurzió műveletét az

   $$\begin{aligned} f(0, m_1, \ldots, m_n) &= g(m_1, \ldots, m_n) \\ f(succ(m), m_1, \ldots, m_n) &= h(m, f(m, m_1, \ldots, m_n), m_1, \ldots, m_n) \end{aligned}$$

   függvényekkel adtuk meg. Jelölje a g és h primitív rekurzív függvények CL-kifejezéseit G és H.

   Az f primitív rekurzív függvény CL-kifejezése megadható az Y fixpont kombinátorral is, de erre a CL-kifejezésre P. I. Bernays adott egy rekurzió mentes leírást:

   $F \equiv \lambda^* m m_1 \ldots m_n \,.\, \mathsf{R}(G m_1 \ldots m_n)(H' m_1 \ldots m_n) m,$

   ahol R egy olyan kombinátor, amelyre

   $$\begin{aligned} \mathsf{R}\, E_1 E_2\, \overline{0} &\to_w E_1 \\ \mathsf{R}\, E_1 E_2 (\mathsf{succ}\ n) &\to_w E_2 n (\mathsf{R}\, E_1 E_2 n), \end{aligned}$$

   és

   $H' \equiv \lambda^* m_1 \ldots m_n u v \,.\, H u v m_1 \ldots m_n.$

   A 2.4.5. tétel és az R gyenge redukcióinak az alkalmazásaival

   $F\ \overline{0} \overline{m_1} \ldots \overline{m_n} \to_w^+$
   $\mathsf{R}(G\overline{m_1} \ldots \overline{m_n})(H'\overline{m_1} \ldots \overline{m_n})\overline{0} \to_w$

$G\overline{m_1}\ldots\overline{m_n}$,

és

$F(\mathsf{succ}\,\overline{m})\overline{m_1}\ldots\overline{m_n} \twoheadrightarrow_w^+$
$R(G\overline{m_1}\ldots\overline{m_n})(H'\overline{m_1}\ldots\overline{m_n})(\mathsf{succ}\,\overline{m}) \twoheadrightarrow_w$
$H'\overline{m_1}\ldots\overline{m_n}\ \overline{m}(R(G\overline{m_1}\ldots\overline{m_n})(H'\overline{m_1}\ldots\overline{m_n})\overline{m}) \twoheadleftarrow_w^+$
$H'\overline{m_1}\ldots\overline{m_n}\ \overline{m}(F\overline{m}\ \overline{m_1}\ldots\overline{m_n}) \twoheadrightarrow_w^+$
$H\overline{m}(F\overline{m}\ \overline{m_2}\ldots\overline{m_n})\overline{m_1}\ldots\overline{m_n}$,

azaz az 5. pontban leírt primitív rekurzió teljesül. Most már csak az R kombinátort kell megadnunk. Legyen

$R \equiv \lambda^* xyu\,.\,u(Qy)(P\overline{0}x)\overline{1}$,

ahol

$Q \equiv \lambda^* yv\,.\,P(\mathsf{succ}\,(v\ \overline{0}))(y(v\ \overline{0})(v\ \overline{1}))$,
$P \equiv \lambda^* xyz\,.\,z(Ky)x$.

A P kombinátorról belátható, hogy

$$PEF\,\overline{0} \twoheadrightarrow_w^+ E,$$
$$PEF(\mathsf{succ}\ m) \twoheadrightarrow_w^+ F.$$

Ekkor

$R E_1 E_2\ \overline{0} \twoheadrightarrow_w^+$
$\overline{0}(QE_2)(P\ \overline{0}E_1)\overline{1} \equiv$
$KI(QE_2)(P\ \overline{0}E_1)\overline{1} \twoheadrightarrow_w^+$
$P\ \overline{0}E_1\overline{1} \twoheadrightarrow_w^+$
E_1.

Hasonlóan az is könnyen belátható, hogy

$R E_1 E_2(\mathsf{succ}\ n) \twoheadrightarrow_w^+$

$E_2 n(\mathsf{R} E_1 E_2 n)$.

Tehát a primitív rekurzív függvények a fenti 1–5. pontokban megadott módszerekkel *CL*-definiálhatók, így a következő tételt kaptuk:

2.8.3. Tétel. (Kleene tétel)

Minden primitív rekurzív függvény CL-definiálható.

A *teljes rekurzív függvényeket* a primitív rekurzív függvények definíciójában szereplő alapfüggvényekkel, a helyettesítés, a primitív rekurzió és a *minimalizálás* műveleteivel kapjuk meg (**??**. definíció). Megmutatjuk, hogy a minimalizálás is leírható a kombinátor logikában.

6. A **??**. pontban láttuk, hogy a minimalizálás

   $f(m_1, \ldots, m_n) = h(\mu m(g(m_1, \ldots, m_n, m) = 0))$

   függvényének megadásában szereplő $\mu m(g(m_1, \ldots, m_n, m) = 0)$ előállítható egy $min(m_1, \ldots, m_n, m)$ függvény felhasználásával, az

   $$m = 0\ ;$$
   $$min(m_1, \ldots, m_n, m) = \mathit{if}\quad g(m_1, \ldots, m_n, m) = 0$$
   $$\mathit{then}\ m$$
   $$\mathit{else}\ min(m_1, \ldots, m_n, succ(m))\ ;$$

   programmal. Ekkor tehát

   $min(m_1, \ldots, m_n, 0) = \mu m(g(m_1, \ldots, m_n, m) = 0)$,

   és így

   $f(m_1, \ldots, m_n) = h(min(m_1, \ldots, m_n, 0))$.

   Ha az f, h és a min függvény *CL*-kifejezése F, H és min, akkor

   $F \equiv \lambda^* m_1 \ldots m_n \,.\, H(\mathsf{min}\ m_1 \ldots m_n \overline{0})$.

   Most már csak a min függvénynek kell a *CL*-kifejezését megadnunk.

Ezt megadhatjuk az Y fixpont kombinátor felhasználásával, de P. I. Bernays erre is kidolgozott egy rekurzió mentes leírást.

Jelölje G a g függvény CL-kifejezését. Ekkor

$\mathsf{min} \equiv \lambda^* m_1 \ldots m_n m \,.\, Q(G m_1 \ldots m_n) m,$

ahol

$Q \equiv \lambda^* xy \,.\, Tx(xy)(Tx)y,$
$T \equiv \lambda^* x \,.\, P\ \overline{0}(\lambda^* uv \,.\, u(x(\mathsf{succ}\ v))u(\mathsf{succ}\ v)),$
$P \equiv \lambda^* xyz \,.\, z(\mathsf{K}y)x,$

ahol P megegyezik a primitív rekurzió 5. pontjában szereplő P kifejezéssel.

Először a Q CL-kifejezésnek azt a tulajdonságát bizonyítjuk be, hogy

$$\begin{array}{llll} QE_1E_2 & \twoheadrightarrow_w^+ & E_2, & \text{ha } E_1E_2 =_w \overline{0}, \\ QE_1E_2 & \twoheadrightarrow_w^+ & QE_1(\mathsf{succ}\ E_2), & \text{ha } E_1E_2 =_w \overline{m},\ \overline{m} \neq_w \overline{0}. \end{array}$$

Megjegyezzük, hogy mindkét sor baloldalán a $(QE_1)E_2$ kifejezés van, a feltételben viszont az (E_1E_2) kifejezés szerepel.

$QE_1E_2 \equiv$
$(\lambda^* xy \,.\, Tx(xy)(Tx)y)E_1E_2 \ \twoheadrightarrow_w^+$
$TE_1(E_1E_2)(TE_1)E_2 \equiv$
$(\lambda^* x \,.\, P\ \overline{0}(\lambda^* uv \,.\, u(x(\mathsf{succ}\ v))u(\mathsf{succ}\ v)))E_1(E_1E_2)(TE_1)E_2 \ \twoheadrightarrow_w$
$P\ \overline{0}(\lambda^* uv \,.\, u(E_1(\mathsf{succ}\ v))u(\mathsf{succ}\ v))(E_1E_2)(TE_1)E_2.$

Ha $E_1E_2 = \overline{0}$, akkor a P kombinátor korábban már megadott $PEF\ \overline{0} \ \twoheadrightarrow_w^+ \ E$ redukciójának végrehajtása után

$\overline{0}(TE_1)E_2 \equiv$
$\mathsf{K}I(TE_1)E_2 \ \twoheadrightarrow_w^+$

E_2,

és ha $E_1E_2 = \overline{m}$, akkor a $PEF(\mathsf{succ}\ m) \quad \twoheadrightarrow_w^+ \quad F$ alapján

$(\lambda^* uv\,.\,u(E_1(\mathsf{succ}\ v))u(\mathsf{succ}\ v))(TE_1)E_2 \ \twoheadrightarrow_w^+$
$TE_1(E_1\underline{(\mathsf{succ}\ E_2)})(TE_1)\underline{(\mathsf{succ}\ E_2)} \ \twoheadleftarrow_w$
$(\lambda^* y\,.\,T\underline{E_1}(\underline{E_1}y)(T\underline{E_1})y)(\mathsf{succ}\ E_2) \ \twoheadleftarrow_w$
$(\lambda^* xy\,.\,Tx(xy)(Tx)y)E_1(\mathsf{succ}\ E_2) \equiv$
$QE_1(\mathsf{succ}\ E_2)$.

Ezzel a Q megadott tulajdonságait bebizonyítottuk.

A ***min*** függvény fenti definíciója alapján

$\mathsf{min}\ \overline{m_1} \ldots \overline{m_n}\ \overline{m} \ \twoheadrightarrow_w^+$
$Q(G\overline{m_1} \ldots \overline{m_n})\overline{m}$,

és felhasználva a Q tulajdonságait, ha $G\overline{m_1} \ldots \overline{m_n}\ \overline{m} =_w \overline{0}$, akkor

$Q(G\overline{m_1} \ldots \overline{m_n})m \ \twoheadrightarrow_w^+ \ \overline{m}$.

Ellenkező esetben, azaz ha $G\overline{m_1} \ldots \overline{m_n}\ \overline{m} \neq_w \overline{0}$, akkor

$Q(G\overline{m_1} \ldots \overline{m_n})\overline{m} \ \twoheadrightarrow_w^+$
$Q(G\overline{m_1} \ldots \overline{m_n})(\mathsf{succ}\ \overline{m}) \ \twoheadleftarrow_w^+$
$\mathsf{min}\ \overline{m_1} \ldots \overline{m_n}(\mathsf{succ}\ \overline{m})$.

Megmutattuk tehát, hogy a ***min*** kifejezés valóban a *min* függvény *CL*-kifejezése.

A fenti 1–6. pontokkal a teljes rekurzív függvények *CL*-definiálhatók, így kimondhatjuk a következő tételt:

2.8.4. Tétel. (Kleene tétel)

Minden teljes rekurzív függvény CL-definiálható.

A λ-kalkulushoz hasonlóan, a *parciális rekurzív függvények* a normál formával a kombinátor logikában is definiálhatók úgy, hogy ahol az

$f(m_1, m_2, \ldots, m_n)$ nincs értelmezve, ott az f-nek megfeleltetett F CL-kifejezésre az $F\overline{m_1}\,\overline{m_2}\ldots\overline{m_n}$ kifejezésnek ne legyen normál formája.

Ezzel a definícióval azonban hasonló problémákat kapunk, mint a λ-kalkulusban, a parciális rekurzív függvényeket a kombinátor logikában is a *megoldhatóság* alkalmazásával tudjuk csak pontosan definiálni. Ezzel a témakörrel nem foglalkozunk, de megemlítjük a parciális rekurzív függvények és a CL-definiálhatóság kapcsolatát:

2.8.5. Tétel. (Kleene tétel)

Minden parciális rekurzív függvény CL-definiálható.

2.9. A λ-kalkulus és a kombinátor logika kapcsolata

Úgy tűnik, különösen a λ^* absztrakció bevezetése után, hogy a kombinátor logika és a λ-kalkulus között sok hasonlóság van. A következőkben a kombinátor logika és a λ-kalkulus közötti kapcsolatnak, a kifejezések konvertálhatóságának tulajdonságait vizsgáljuk meg.

2.9.1. Definíció. A $\mathcal{C} \Rightarrow \Lambda$ traszformáció:

Ha $E \in \mathcal{C}$, akkor alakítsa át ezt a CL-kifejezést egy λ-kifejezésre a

$(\cdot)_\Lambda : \mathcal{C} \Rightarrow \Lambda$

leképezés, ahol

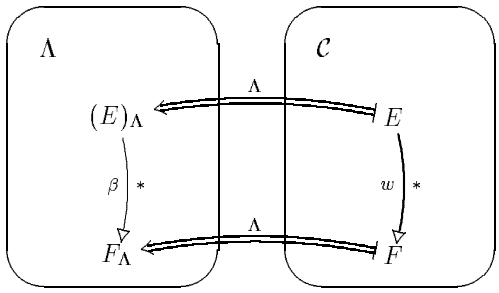
$$\begin{array}{lcl} (x)_\Lambda & \mapsto & x, \\ (\mathsf{K})_\Lambda & \mapsto & \mathsf{K}, \\ (\mathsf{S})_\Lambda & \mapsto & \mathsf{S}, \\ (EF)_\Lambda & \mapsto & (E)_\Lambda (F)_\Lambda. \end{array}$$

A kombinátor logika gyenge redukciója „megfelel" a λ-kalkulus β-redukciójának, ezt mondja ki a következő tétel (2.11. ábra).

2.9.2. Tétel. (A $(\cdot)_\Lambda$ leképezés gyenge redukció tartó)

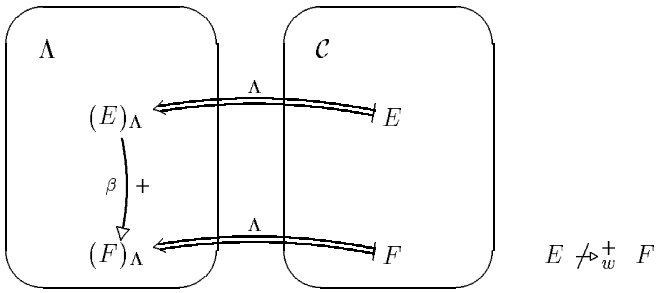
Ha az E és F CL-kifejezésekre $E \twoheadrightarrow_w^* F$, akkor $(E)_\Lambda \twoheadrightarrow_\beta^* (F)_\Lambda$.

A tétel az $E \twoheadrightarrow_w^* F$ szerkezetére vonatkozó indukcióval látható be.



2.11. ábra. A $(\cdot)_\Lambda : \mathcal{C} \rightleftarrows \Lambda$ leképezés gyenge redukció tartó

A $(\cdot)_\Lambda$ leképezés azonban a β-redukciót nem tartja meg, azaz ha az $(E)_\Lambda$ és $(F)_\Lambda$ λ-kifejezésekre $(E)_\Lambda \twoheadrightarrow_\beta^+ (F)_\Lambda$, ebből még nem feltétlenül következik, hogy $E \twoheadrightarrow_w^+ F$ (2.12. ábra).



2.12. ábra. A $(\cdot)_\Lambda : \mathcal{C} \rightleftarrows \Lambda$ leképezés és a β-redukció

2.9.3. Példa. *(Az $S(KI)K$ kifejezés)*

Ha $E \equiv S(KI)K$, akkor

$(S(KI)K)_\Lambda \mapsto^+ S(KI)K \equiv$

$(\lambda xyz\,.\,xz(yz))(\mathsf{K}\mathsf{I})\mathsf{K} \twoheadrightarrow^{+}_{\beta}$
$\lambda z\,.\,\mathsf{K}\,z \equiv$
$\lambda z\,.\,(\lambda xy.x)z \twoheadrightarrow_{\beta}$
$\lambda z\,.\,\lambda y\,.\,z \equiv \mathsf{K}\,.$

Mivel $(\mathsf{K})_{\Lambda} \mapsto \mathsf{K}$, az $\mathsf{S}(\mathsf{K}\mathsf{I})\mathsf{K} \twoheadrightarrow^{+}_{w} \mathsf{K}$ redukciót kellene igazolni, de nyilvánvaló, hogy ez nem lehetséges. □

Most vizsgáljuk meg a fordított irányú leképezést. Jelölje az E λ-kifejezés átalakítását $(E)_{\mathcal{C}}$, tehát definiáljuk a

$(\cdot)_{\mathcal{C}} : \Lambda \Rightarrow \mathcal{C}$

leképezést.

2.9.4. Definíció. A $\Lambda \Rightarrow \mathcal{C}$ transzformáció:

Ha $E \in \Lambda$, akkor alakítsa át ezt a λ-kifejezést egy CL-kifejezésre a

$(\cdot)_{\mathcal{C}} : \Lambda \Rightarrow \mathcal{C}$

leképezés, ahol

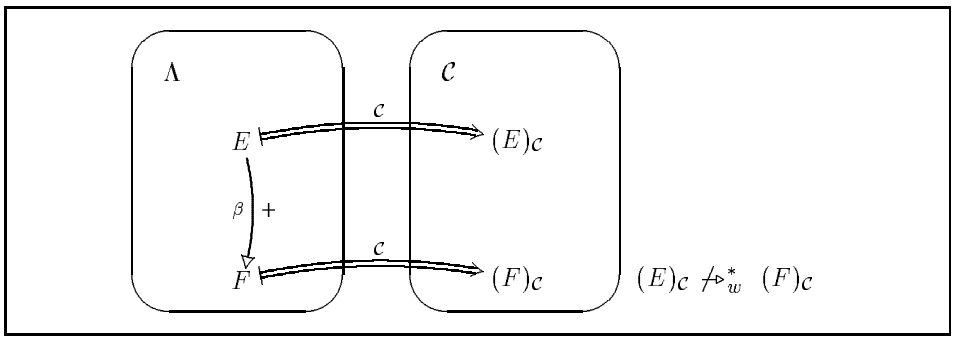
$$\begin{array}{lcl} (x)_{\mathcal{C}} & \mapsto & x, \\ (EF)_{\mathcal{C}} & \mapsto & (E)_{\mathcal{C}}(F)_{\mathcal{C}}, \\ (\lambda x\,.\,E)_{\mathcal{C}} & \mapsto & \lambda^{*}x\,.\,(E)_{\mathcal{C}}. \end{array}$$

Azt várnánk, hogy ha $E \twoheadrightarrow^{+}_{\beta} F$ a λ-kalkulusban, akkor az $(E)_{\mathcal{C}} \twoheadrightarrow^{+}_{w} (F)_{\mathcal{C}}$ levezetésnek is teljesülnie kellene a kombinátor logikában, ez a tulajdonság azonban, mint a következő példa mutatja, *nem* teljesül (2.13. ábra).

2.9.5. Példa. *(A $\lambda x\,.\,(\mathsf{I}y)$ kifejezés)*
A λ-kalkulusban

$\lambda x\,.\,(\mathsf{I}y) \equiv \lambda x\,.\,(\underline{(\lambda x\,.\,x)y}) \twoheadrightarrow_{\beta} \lambda x\,.\,y,$

és a két λ-kifejezésre



2.13. ábra. $(\cdot)_{\mathcal{C}} : \Lambda \to \mathcal{C}$ és a redukciók

$(\lambda x\,.\,((\lambda x\,.\,x)y))_{\mathcal{C}} \mapsto$
$\lambda^* x\,.\,((\lambda x\,.\,x)y)_{\mathcal{C}} \mapsto$
$\lambda^* x\,.\,((\lambda x\,.\,x)_{\mathcal{C}}(y)_{\mathcal{C}}) \mapsto$
$\lambda^* x\,.\,((\lambda^* x\,.\,(x)_{\mathcal{C}})(y)_{\mathcal{C}}) \mapsto$
$\lambda^* x\,.\,((\lambda^* x\,.\,x)(y)_{\mathcal{C}}) \mapsto$
$\lambda^* x\,.\,((\lambda^* x\,.\,x)y) \equiv$
$\lambda^* x\,.\,(\mathsf{I}y) \equiv$
$\mathsf{S}(\lambda^* x\,.\,\mathsf{I})(\lambda^* x\,.\,y) \equiv$
$\mathsf{S}(\mathsf{KI})(\lambda^* x\,.\,y) \equiv$
$\mathsf{S}(\mathsf{KI})(\mathsf{K}y)$,

valamint

$(\lambda x\,.\,y)_{\mathcal{C}} \mapsto$
$\lambda^* x\,.\,(y)_{\mathcal{C}} \mapsto$
$\lambda^* x\,.\,y \equiv$
$\mathsf{K}y$,

és nyilvánvaló, hogy a kombinátor logikában

$\mathsf{S}(\mathsf{KI})(\mathsf{K}y) \not\to_w^+ \mathsf{K}y$. □

A 2.9.3. és 2.9.5. példák adják meg annak a magyarázatát, hogy a kombinátor logikában a redukálások és a normál forma a „gyenge" jelzőt kapja. A *gyenge* jelző azt jelenti, hogy ha egy CL-kifejezés nem redukálható, azaz (gyenge) normál formában van, akkor előfordulhat, hogy a neki megfeleltetett λ-kifejezésnek van redexe, tehát a neki megfeleltetett λ-kifejezésben további redukciókat lehet végehajtani.

2.9.6. Következmény.

A normál forma nem invariáns tulajdonság a λ-kalkulus és a kombinátor logika között.

A kombinátor logikában is definiálhatjuk a *megoldhatóságot*, azt mondjuk, hogy az E CL-kifejezés megoldható, ha léteznek olyan $F_1, F_2, \ldots, F_n$ CL-kifejezések, amelyekre $EF_1F_2 \ldots F_n =_w I$. A *megoldhatóság* (és a vele ekvivalens *fej normál forma*) viszont már invariáns tulajdonság a λ-kalkulus és a kombinátor logika között, azaz ha $E \in \Lambda$ és $G \in \mathcal{C}$, akkor

- E megoldható akkor és csak akkor, ha $(E)_{\mathcal{C}}$ megoldható,
- G megoldható akkor és csak akkor, ha $(G)_{\Lambda}$ megoldható.

A normál forma nem-invariáns tulajdonságát az okozza, hogy a kombinátor logikában a λ-kalkulus ξ-szabálya nem érvényes, azaz ha $E \twoheadrightarrow_w^+ F$, akkor $\lambda^* x \,.\, E \not\twoheadrightarrow_w^+ \lambda^* x \,.\, F$.

Azonban, ha a $CL{+}ext$ kalkulusban vizsgáljuk a CL-kifejezések egyenlőségét, azt tapasztaljuk, hogy ebben a kalkulusban a $(\cdot)_{\mathcal{C}}$ leképezés a β-redukciókat megtartja.

2.9.7. Példa. *(A 2.9.5. példa a CL+ext kalkulusban)*

Ha a 2.9.5. példa két eredményére az E kifejezést applikáljuk, akkor

$\underline{S(KI)(Ky)E} \twoheadrightarrow_w$
$\underline{KIE}(KyE) \twoheadrightarrow_w$
$\underline{I(KyE)} \twoheadrightarrow_w$
$\underline{KyE} \twoheadrightarrow_w y,$

és

$\underline{\mathsf{K}yE} \twoheadrightarrow_w y,$

azaz az *ext* axióma alapján a két kifejezés egyenlő. □

Az *ext* axiómában megfogalmazott *kiterjesztés* egy kicsit gyengíthető, a β-redukció megtartása még a gyengébb feltétel mellett is teljesül.

2.9.8. Definíció. Funkcionális *CL*-kifejezés:

Az E CL-kifejezést funkcionálisnak nevezzük, ha a következő CL-kifejezések egyikével ekvivalens:

K, $\mathsf{K}F$, S, $\mathsf{S}F$, $\mathsf{S}FG$.

Megjegyezzük, hogy egy E CL-kifejezés $\lambda^*.E$ CL-kifejezése mindig funkcionális, ezért a $\lambda x.F$ λ-absztrakciónak megfeleltetett $(\lambda x.F)_{\mathcal{C}} \mapsto \lambda^* x.(F)_{\mathcal{C}}$ CL-kifejezés mindig funkcionális, a λ-kifejezés „függvény jellege" az átalakítás után kapott CL-kifejezésben is megmarad. Ez a tulajdonság csak a λ^* és λ_1^* absztrakciókra áll fenn.

A λ+*wext*-kalkulusban a kiterjesztési axiómát (?? és ?? definíció) csak olyan Ex, Fx λ-kifejezésekre mondtuk ki, ahol E és F λ-absztrakció. Mivel a λ-absztrakcióknak a funkcionális CL-kifejezések felelnek meg, ha a kiterjeszthetőséget csak a funkcionális CL-kifejezésekre engedjük meg, akkor a kiterjeszthetőséget most is *gyengének* nevezzük.

2.9.9. Definíció. Gyenge kiterjeszthetőség:

Ha E és F funkcionális CL-kifejezések, $Ex =_w Fx$, és $x \notin FV(E)$, $x \notin FV(F)$, akkor $E =_w F$.

Ha a kiterjesztés helyett a gyenge kiterjesztést adjuk meg axiómaként, akkor a CL+*wext* kalkulust kapjuk meg.

2.9.10. Definíció. Az egyszerű típusnélküli gyenge kiterjesztett kombinátor logika:

A gyenge kiterjesztett kombinátor logikát úgy kapjuk meg, hogy a kombinátor logika axiómáit kiegészítjük a

II.vi. E *és* F *funkcionális,*

$$Ex =_w Fx \;\Rightarrow\; E =_w F \qquad \textit{gyenge kiterjeszthetőség} \qquad (wext)$$

axiómával.

A $CL{+}wext$ kombinátor logika a β-redukciókat megtartja, ezt mondja ki a következő tétel.

2.9.11. Tétel. (A $(\cdot)_{\mathcal{C}}$ leképezés β-redukció tartó $CL{+}wext$ -ben)

Ha az E *és* F λ*-kifejezésekre* $E \twoheadrightarrow_\beta^* F$*, akkor a* $CL{+}wext$ *kombinátor logikában* $(E)_{\mathcal{C}} \twoheadrightarrow_w^* (F)_{\mathcal{C}}$*.*

Megjegyezzük, hogy a $CL{+}ext$ kombinátor logika esetén a β-redukciók és gyenge redukciók tartása csak a $\lambda\eta$-kalkulusra, vagy az ezzel ekvivalens $\lambda{+}ext$-kalkulusra teljesül, CL kombinátor logika esetén a $\lambda{+}wext$-kalkulusra pedig csak akkor, ha a λ_2^*, λ_3^* vagy a λ_4^* zárójeles absztrakciót használjuk.

(2.14. ábra.)

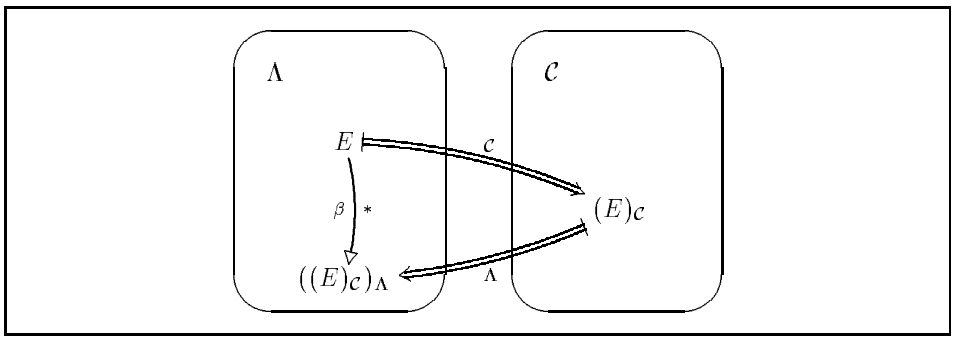
$$\begin{array}{ccccc} CL & \subseteq & CL+wext & \subseteq & CL+ext \\ \Updownarrow{\scriptstyle \lambda_2^*} & & \Updownarrow{\scriptstyle \lambda^*} & & \Updownarrow \\ \lambda+wext & \subseteq & \lambda & \subseteq & \lambda+ext \equiv \lambda\eta \end{array}$$

2.14. ábra. A λ-kalkulus és a kombinátor logika

Ezek után vizsgáljuk meg, hogy a $(\cdot)_{\mathcal{C}}$ és a $(\cdot)_\Lambda$ leképezések kompozíciója milyen eredményt ad.

2.9.12. Tétel. (A $((\cdot)_{\mathcal{C}})_\Lambda$ leképezés)

Minden E λ*-kifejezésre* $E \twoheadrightarrow_\beta^* ((E)_{\mathcal{C}})_\Lambda$*.*



2.15. ábra. Az E és az $((E)_{\mathcal{C}})_\Lambda$ kapcsolata

A tétel (2.15. ábra) következménye az, hogy ha $E \in \Lambda^0$, azaz az E nem tartalmaz változókat, akkor $(E)_{\mathcal{C}}$-ben csak K, S kombinátorok vannak, és így $((E)_{\mathcal{C}})_\Lambda$ is csak a K, S kombinátorokból áll. Mivel a fenti tétel alapján

$E =_\beta ((E)_{\mathcal{C}})_\Lambda$,

igaz a következő állítás:

2.9.13. Következmény.

Ha $E \in \Lambda^0$, akkor az E leírható egy olyan λ-kifejezéssel, ami csak a K és S kombinátorokat tartalmazza.

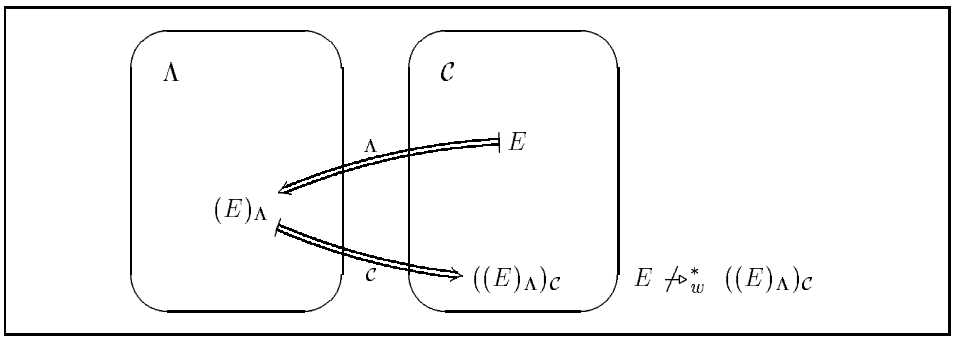
Ez a következmény az alapja annak, hogy a K és S kombinátorokat a λ-kalkulus *bázisának* nevezzük.

A fenti tétel a leképezések fordított sorrendű elvégzésére nem teljesül, azaz ha $E \in \mathcal{C}$, akkor $E \not\twoheadrightarrow_w^* ((E)_\Lambda)_{\mathcal{C}}$ (2.16. ábra).

Ezt mutatja a következő példa.

2.9.14. Példa. *(A $((K)_\Lambda)_{\mathcal{C}}$ kifejezés)*

$((K)_\Lambda)_{\mathcal{C}} \mapsto$
$(\mathsf{K})_{\mathcal{C}} \equiv$
$(\lambda xy\,.\,x)_{\mathcal{C}} \mapsto$
$\lambda^* x\,.\,(\lambda y\,.\,x)_{\mathcal{C}} \mapsto$



2.16. ábra. Az E és az $((E)_\Lambda)\mathcal{C}$ kapcsolata

$\lambda^* x\,.\,(\lambda^* y\,.\,(x)_\mathcal{C}) \mapsto$
$\lambda^* x\,.\,(\lambda^* y\,.\,x) \equiv$
$\lambda^* x\,.\,\mathsf{K}x \equiv$
$\mathsf{S}(\lambda^* x\,.\,\mathsf{K})(\lambda^* x\,.\,x) \equiv$
$\mathsf{S}(\mathsf{KK})(\lambda^* x\,.\,x) \equiv$
$\mathsf{S}(\mathsf{KK})\mathsf{I}$,

és nyilvánvaló, hogy

$\mathsf{K} \not\rightarrowtriangle^*_w \ \mathsf{S}(\mathsf{KK})\mathsf{I}$. □

Ha a fenti 2.9.14. példa K és $\mathsf{S}(\mathsf{KK})\mathsf{I}$ kifejezéseire xy-t applikáljuk, akkor

$\underline{\mathsf{K}xy} \ \rightarrowtriangle_w \ x$,

és

$\underline{\mathsf{S}(\mathsf{KK})\mathsf{I}xy} \ \rightarrowtriangle_w$
$\underline{\mathsf{KK}x}(\mathsf{I}x)y \ \rightarrowtriangle_w$
$\underline{\mathsf{K}(\mathsf{I}x)y} \ \rightarrowtriangle_w$
$\underline{\mathsf{I}x} \ \rightarrowtriangle_w \ x$,

azaz a fenti két kifejezés a $CL{+}ext$ kalkulusban megegyezik. Az *ext* axióma helyettesíthető a gyengébb *wext* axiómával, és így kimondható a következő tétel.

2.9.15. Tétel. (A $((\cdot)_\Lambda)_\mathcal{C}$ leképezés)

A $CL{+}wext$ kombinátor logikában minden E CL-kifejezésre $E \twoheadrightarrow_w^* ((E)_\Lambda)_\mathcal{C}$.

A. FÜGGELÉK

A kombinátor logika kombinátorai

- A könyvben szereplő kombinátorok és gyenge redukcióik

$\mathsf{K}EF$	$\rightarrow_w$	E	
$\mathsf{S}EFG$	$\rightarrow_w$	$EG(FG)$	
$\mathsf{I}E$	$\rightarrow_w$	E	
$\mathsf{B}EFG$	$\rightarrow_w$	$E(FG)$	
$\mathsf{C}EFG$	$\rightarrow_w$	EGF	
$\mathsf{S'}CEFG$	$\rightarrow_w$	$C(EG)(FG)$	$C \in \mathcal{C}^0$
$\mathsf{B'}CEFG$	$\rightarrow_w$	$CE(FG)$	$C \in \mathcal{C}^0$
B^*CEFG	$\rightarrow_w$	$C(E(FG))$	$C \in \mathcal{C}^0$
$\mathsf{C'}CEFG$	$\rightarrow_w$	$C(EG)F$	$C \in \mathcal{C}^0$
$\mathsf{W}EF$	$\rightarrow_w$	EFF	
$\mathsf{J}EFG$	$\rightarrow_w$	EF	(Joy)
$\mathsf{J}EFGH$	$\rightarrow_w$	$EF(EHG)$	$(Rosser)$
$\mathsf{J'}EFGH$	$\rightarrow_w$	EFG	
C_*EF	$\rightarrow_w$	FE	
$\mathsf{Y}E$	$\leftrightarrow_w$	$E(\mathsf{Y}E)$	
ΘE	$\rightarrow_w$	$E(\Theta E)$	

- Néhány kombinátor kifejezése

$$
\begin{array}{lcll}
I & \equiv & SKE & \\
I & \equiv & SB(KI) & \\
I & \equiv & CKC & \\
I & \equiv & WK & \\
K & \equiv & BS(BK) & \\
S & \equiv & B(BW)(BBC) & \\
S & \equiv & B(B(BW)C)(BB) & \\
B & \equiv & S(KS)K & \\
B & \equiv & BSK & \\
B & \equiv & C(JIC)(JI) & (Rosser) \\
C & \equiv & S(S(K(S(KS)K))S)(KK) & \\
C & \equiv & S(BBS)(KK) & \\
C & \equiv & JC_*(JC_*)(JC_*) & (Rosser) \\
B' & \equiv & BB & \\
C' & \equiv & B(BC)B & \\
S' & \equiv & B(BS)B & \\
S' & \equiv & B(B(B(B(BW)C)(BB)))B & \\
W & \equiv & CSI & \\
W & \equiv & SS(SK) & \\
W & \equiv & SS(KI) & \\
W & \equiv & S(CI) & \\
W & \equiv & C(S(CC)(CC)) & \\
W & \equiv & C(C(BC(C(BJC_*)C_*))C_*) & (Rosser) \\
C_* & \equiv & CI & \\
C_* & \equiv & JII & (Rosser) \\
Y & \equiv & WS(BWB) & \\
Y & \equiv & SSI(SB(K(SII))) & \\
Y & \equiv & S(CB(SII))(CB(SII)) &
\end{array}
$$

- Néhány optimalizáló újraírási szabály

$S(KE)I$	$\mapsto$	E	
$S(KE)(KF)$	$\mapsto$	$K(EF)$	
$S(KE)F$	$\mapsto$	BEF	
$SE(KF)$	$\mapsto$	CEF	
$S(KE)$	$\mapsto$	BE	
$S(BC\ E)F$	$\mapsto$	$S'C\ EF$	
$S(KC\ E)F$	$\mapsto$	$B'C\ EF$	
$S(KC)(BEF)$	$\mapsto$	B^*CEF	
$S(BC\ E)(KF)$	$\mapsto$	$C'C\ EF$	
SEI	$\mapsto$	WE	
$B(C\ E)F$	$\mapsto$	$B'C\ EF$	
$BC(BEF)$	$\mapsto$	B^*CEF	
BEI	$\mapsto$	E	
BIE	$\mapsto$	E	
$B(BEF)$	$\mapsto$	$B'EF$	
$C(BC\ E)F$	$\mapsto$	$C'C\ EF$	
$C(BEF)$	$\mapsto$	$C'EF$	
$S'C\ (KE)F$	$\mapsto$	$B'C\ EF$	
$S'C\ E(KF)$	$\mapsto$	$C'C\ EF$	
$C'EIF$	$\mapsto$	CEF	
JIE	$\mapsto$	E	(Joy)
$Y(KE)$	$\mapsto$	E	

B. FÜGGELÉK

Definíciók, tételek jegyzéke

Irodalom

[Bar84] Hendrik Pieter Barendregt, *The Lambda Calculus, Its Syntax and Semantics*, North-Holland, 1984.

[Hin86] J. Roger Hindley, Jonathan P. Seldin, *Introduction to Combinators and λ-Calculus*, Cambridge University Press, 1986.

[Pey87] Simon L. Peyton Jones, *The Implementation of Functional Programming Languages*, Prentice Hall, 1987.

[Dil88] Antoni Diller, *Compiling Functional Languages*, John Wiley & Sons Ltd., 1988.

[Bir88] Richard Bird, Philip Wadler, *Introduction to Functional Programming*, Prentice Hall, 1988.

[Hud89] Paul Hudak, *Conception, Evolution, and Application of Functional Programming Languages*, ACM Computing Surveys, Vol.21, No.3, (September 1989)

[Csö92] Csörnyei Zoltán, *Bevezetés a fordítóprogramok elméletébe, 1. rész [Analízis]*, Tankönyvkiadó, Budapest, 1992.

[Csö93] Csörnyei Zoltán, *Bevezetés a fordítóprogramok elméletébe, 2. rész [Szintézis]*, ELTE Kiadó, Budapest, 1993.

[Fis93] Alice E. Fischer, Frances S. Grodzinsky, *The Anatomy of Programming Languages*, Prentice-Hall International, Inc., 1993.

[Pla93] Rinus Plasmeijer, Marko van Eekelen, *Functional Programming and Parallel Graph Rewriting*, Addison-Wesley Publishing Company, 1993.

[Wil95] Reinhard Wilhelm, Dieter Maurer, *Compiler Design*, Addison-Wesley Publishing Company, 1995.

[Gor96] Mike Gordon, *Introduction to Functional Programming*, Lecture Notes, University of Cambridge, 1996.

[Pau96] Lawrence C. Paulson, *Foundation of Functional Programming*, Lecture Notes, University of Cambridge, 1996.

[Har97] John Harrison, *Introduction to Functional Programming*, Lecture Notes, University of Cambridge, 1997.

[Ong98] C.-H. Luke Ong, *Lambda Calculus*, Lecture Notes, Oxford University, 1998.

[Ker00] Andrew D. Ker, *Lambda Calculus*, Lecture Notes, Oxford University, 2000.

Névmutató

Tárgymutató