NÜMERİK FONKSİYONLARIN DEĞİŞİK KATEGORİLERDE GÖSTERİMİ

YÜKSEK LİSANS TEZİ

Savaş TANRIÖVEN

DANIŞMAN
Yard.Doç. Dr. Enver Önder USLU

MATEMATİK ANABİLİM DALI
HAZİRAN 2010

AFYON KOCATEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

NÜMERİK FONKSİYONLARIN DEĞİŞİK KATEGORİLERDE GÖSTERİMİ

Savaş TANRIÖVEN

DANIŞMAN
Yard.Doç. Dr. Enver Önder USLU

MATEMATİK ANABİLİM DALI

HAZİRAN 2010

ONAY SAYFASI

Yard.Doç. Dr. Enver Önder USLU danışmanlığında,

Savaş TANRIÖVEN tarafından hazırlanan

Nümerik Fonksiyonların Değişik Kategorilerde Gösterimi

başlıklı bu çalışma lisansüstü eğitim ve öğretim yönetmeliğinin ilgili maddeleri

uyarınca

……./……/……..

tarihinde aşağıdaki jüri tarafından

Matematik Anabilim Dalında

Yüksek lisans tezi olarak oy birliği/oy çokluğu ile kabul edilmiştir.

Başkan:

Üye :

Üye :

Afyon Kocatepe Üniversitesi
Fen Bilimleri Enstitüsü Yönetin Kurulu'nun
...../...../..... tarih ve
……. sayılı kararıyla onaylanmıştır.

Doç. Dr. Rıdvan ÜNAL
Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

NÜMERİK FONKSİYONLARIN DEĞİŞİK KATEGORİLERDE GÖSTERİMİ

Savaş TANRIÖVEN

Afyon Kocatepe Üniversitesi

Fen Bilimleri Enstitüsü

Matematik Anabilim Dalı

Danışman: Yard.Doç. Dr. Enver Önder USLU

Bu çalışma üç bölümden oluşmaktadır. İlk bölüm giriş kısmına ayrılmıştır. İkinci bölümde, çalışmamız için gerekli olan temel kavramlar, Kategoriler ve Funktorlar, Üçlüler (Monad veya Triple), Bir Deductive Sistem Anlamında Önermesel Kalkülüs, Kartezyen Kapalı Kategoriler, Typed λ-Kalkülüs ve C- monoidler hatırlatılmıştır. Üçüncü bölümde, Recursive fonksiyonlar bir Nümerik Fonksiyon Olarak incelenmiş ve Dedekind`in bir önermesinin genellemesi yapılarak, Kartezyen Kapalı Kategorilerde Recursive fonksiyonlar gösterilmiştir.

2010, 61 sayfa

Anahtar Kelimeler: Kategoriler, Kartezyen Kapalı Kategoriler, Recursive(Özyineli) Fonksiyonlar

ABSTRACT

Msc Thesis

REPRESENTING NUMERICAL FUNCTIONS IN VARIOUS CATEGORIES
Savaş TANRIÖVEN

Afyon Kocatepe University

Institute for the Natural and Applied Sciences

Supervisor: Asst. prof. Dr. Enver Önder USLU

This thesis consists of three chapters. In the first chapter is devoted to the introduction section. In the second chapter, some required preparatory notions, Categories and Funktors, Monads or Triples, Propositional Calculus as a deductive system, Cartesian Closed Categories, Typed λ-Calculus and C- monoids are recalled. In the third chapter, Recursive Functions are investigated as Numerical Functions. A Dedekind`s proposition was generalized and Recursive Functions were represented in Cartesian Closed Categories.

2010, Page 61

Key Words: Categories, Cartezian Closed Categories, Recursive Functions

TEŞEKKÜR

Bu çalışmamda danışmanlığımı yapan sayın kıymetli hocam Yard.Doç. Dr. Enver Önder USLU'ya göstermiş olduğu sabır, ilgi, destek ve yardımlarından dolayı teşekkürü bir borç bilirim. Ayrıca çalışmam boyunca samimi desteklerini esirgemeyen ve bana fevkalade sabır gösteren sevgili eşim ve çocuklarıma, maddi ve manevi desteklerini hep arkamda hissettiğim annem, babam ve kardeşlerime sonsuz teşekkür ederim.

Savaş TANRIÖVEN

AFYONKARAHİSAR, Haziran 2010

İÇİNDEKİLER

SİMGELER VE KISALTMALAR DİZİNİ

$\mathcal{C}$	Bir $\mathcal{C}$Kategorisi
A,B,C,..	Bir $\mathcal{C}$Kategorisinin nesneleri
Mor(A,B)	Bir $\mathcal{C}$Kategorisi üzerinde A dan B ye bir morfizm
f, g,...	Dönüşümler ailesi
Ob($\mathcal{C}$)	$\mathcal{C}$nin objeler sınıfı
$I_A{}'$	Birim dönüşüm
S	Kümeler kategorisi
S^*	Noktalanmış kümeler kategorisi
Grp	Grup kategorisi
$Ring_1$	Halka homomorfizmalar kategorisi
${}_R$Mod	R moduller kategorisi
Top	Topolojik uzaylar
Htp	Homotopik denklik sınıfları
(T,ε,μ)	Kategori üzerinde bir monad
$\langle f, g \rangle$	Operatörlerin çarpımı
$\Leftarrow$	İse bağlacı
$\square$	Ve bağlacı
$\rightarrow$	A dan Bye bir ok
$A \times B$	A ve B nin kartezyan çarpımı
CCC	Kartezyen kapalı kategoriler

ŞEKİLLER DİZİNİ

1. GİRİŞ

Kategori teori son 1940 lar da ilk defa S. Maclane tarafından kurulmuş ve ve bu güne kadar önemli bir gelişim göstermiş ve bugün Matematiğin en temel branşlarından biri haline gelmiştir. Kategori teori en genel anlamda matematiğin farklı konularına ait ortak dil olarak ifade edilebilir. Bu bağlamda her bir konu, tanın, formül ve ispat kategori dili ile yeniden yazılmakta ve bu ifadelerin farklı konulardaki karşılıkları araştırılmaktadır. Mevcut bu tezde değişik kategoriler içinde nümerik fonksiyonların gösterimlerini ayrıntılı biçimde incelenecek ve özel örnekler verilecektir. Tezde matematiğin farklı iki konusu olan nümerik fonksiyonlar ve kategori teorinin ortak kullanımları incelenecek ve uygulamaya yönelik örnekler de araştırılacak bu bağlamda bilgisayar örnekleri verilecektir. Tez bir arştırma tezi olup mevcut bilgilerin daha detaylı ve anlaşılır biçimde sunumu olarak hazırlanmıştır.

2. TEMEL KAVRAMLAR

Bu bölümde çalışmamız için gerekli olan bazı temel kavramlar ve sonraki bölümlerde ihtiyaç duyulacak olan bazı Kategoriler, Funktorlar ve Üçlüler hatırlatılacaktır. Bu bölümde kullandığımız temel referanslar (Blyth T.S,1986), (Borceux F.,1994), (Lambek J. 1986) dır.

2.1 Kategoriler ve Funktorlar

Küme teorinin genellemesi olarak düşünülebilir. Tek elemanlı bir kümeden diğer tüm kümelere bir dönüşüm tanımlanabilir. Kategori teoride bu elemanın ismi Terminal Objedir.

Soru 2.1.1. : Boş kümeden başka bir kümeye dönüşüm tanımlanabilir mi?

$\mathcal{C}$; A,B,C,… gibi objelerden oluşan bir sınıf ve

1. Her A,B objesine karşılık Mor(A,B) kümesi
2. A,B,C objelerine karşılık

$$Mor(A,B) \times Mor(B,C) \to Mor(A,C)$$

$$(f,g) \longmapsto gf$$

şeklinde bir dönüşümler ailesi (Kompozisyon diye adlandırılır) yapılarından oluşan bir sistem olsun.

Mor(A,B) nin elemanlarına morfizm denir ve objeler sınıfı Ob($\mathcal{C}$) şeklinde gösterilir. Burada Mor(A,B) kümeleri ayrık kümelerdir.

Eğer $\mathcal{C}$ ailesi aşağıdaki aksiyomları sağlarsa $\mathcal{C}$ bir kategori denir.

1) Assosyatiflik; her $A,B,C,D \epsilon Ob(\mathcal{C})$ ve her $f \epsilon Mor(A,B), g \epsilon Mor(B,C), h \epsilon Mor(C,D)$ için
$$h(gf) = (hg)f$$
2) Birimlik; her $A \epsilon Ob(\mathcal{C})$ için birim morfizm diye adlandırılan ve $I_A \epsilon Mor(A,A)$ ile gösterilen ve her $B, C \epsilon Ob(\mathcal{C})$ ve $f \epsilon Mor(A,B), g \epsilon Mor(C,A)$ için
$$fI_A = f, \quad I_A g = g$$

şartını sağlayan bir morfizm vardır.

Örnek 2.1.2. : Tüm kümeler ve aralarında tanımlanan fonksiyonlar bir kategori oluşturur ve bu kategori **S** veya **Set** ile gösterilir.

Burada neden tanımlamada sınıf kullanıldığına dikkat edilmelidir. Tüm kümelerin kümesi? Tüm kümelerin sınıfı?

Mor($\mathcal{C}$); tüm morfizm kümelerinin birleşimini göstersin.

$Ob(\mathcal{C}) = \phi$ ise $\mathcal{C}$ ye boş kategori denir.

$A \epsilon Ob(\mathcal{C})$ için I_A bir tektir. Eğer $I_A{}'$, A objesi için birim dönüşüm ise

$$I_A = I_A{}' I_A = I_A{}'$$

dir.

Örnekler 2.1.3. :

1. **S** – Category of Sets;
2. Sıralı kümeler kategorisi; Objeleri üzerinde sıralama bağıntısı (yansıma, anti-simetri ve geçişme özellikleri olan bağıntı) tanımlı kümeler ve morfizmleri ise sıralama koruyan fonksiyonlar olan kategori.

   $$f \epsilon Mor(A, B)$$

   olması için $a, a' \epsilon A \; ve \; a \leq a' \; olduğunda \; f(a) \leq f(a')$ olmalı.

3. **S*** noktalanmış kümeler kategorisi; A boştan farklı bir küme ve $a \epsilon A$ olmak üzere (A,a) ikilisine noktalanmış küme denir. f, A dan B ye bir fonksiyon ve (A,a), (B,b) noktalanmış kümeler olmak üzere f fonksiyonu f(a)=b şartını sağlıyorsa f ye (A,a) dan (B,b) ye bir noktalanmış fonksiyon denir. Objeleri noktalanmış kümeler ve morfizmleri de noktalanmış dönüşümler olan kategoriye noktalamış kümeler kategorisi denir.
4. **Grp** kategorisi; objeleri gruplar ve morfizmleri grup homomorfizmleri olan grup kategorisi.

5. **Ab** objeleri Abelian gruplar ve morfizmleri yine grup homomorfizmleri olan Abelian grup kategorisi
6. **$\mathbf{Ring_1}$** veya **$\mathbf{Ri_1}$** objeleri asosyatif birimli halkaları ve morfizmleri halka homomorfizmaları olan kategori.

   R bir Abelian grup olsun.

$$(a+b)c = ac + bc, \qquad a(b+c) = ab + ac$$

$$(ab)c = a(bc) \quad ve \quad 1a = a = a1 \quad her\ a\epsilon A\ için$$

   Şartları sağlanıyorsa R ye birimli asosyatif halka denir.

7. **$_{\mathbf{R}}$Mod;** objeleri R-modüller ve morfizmleri R-modül homomorfizmaları olan kategori.(Burada R birimli asosyatif halkadır.)

$$r(a+a') = ra + ra', \qquad (r+r')a = ra + r'a, \ \ (rr')a = r(r')a, \ \ 1a = a$$

$$F(a+a') = F(a) + F(a'), \ \ f(ra) = rF(a)$$

   Hatırlanacağı üzere R bir cisim olursa R-modüllere vektör uzayı denir.

8. **Top;** Topolojik uzaylar ve sürekli dönüşümler.
9. **Htp;** Topolojik uzaylar ve homotopi denklik sınıfları.
10. **Top***; Noktalanmış topolojik uzaylar ve noktalanmış sürekli dönüşümler.
11. **Htp*;** Noktalanmış topolojik uzaylar ve noktalanmış homotopi sınıfları.
12. Her sıralı küme bir kategoridir? A bir sıralı küme Ob($\mathcal{A}$)=A olsun.

$$a,b\epsilon A\ için \quad Mor(a,b) = f(x) = \begin{cases} \{(a,b)\}, & a \leq b \\ \emptyset, & aksi\ halde \end{cases}$$

   olsun. Bu durumda $\mathcal{A}$ bir kategori olur.

13. Bir kategori olarak grup; A bir grup olsun. A grubu bir $\mathcal{A}$ kategorisi tanımlar öyle ki Ob($\mathcal{A}$)={$*$} tek elemanlı küme olmak üzere $Mor(*,*) = A$ dır ve kompozisyon işlemi grubun işlemidir.
14. Doğal sayılar; Doğal sayılar kümesi üzerinde "$a \leq b \ \Leftrightarrow \ a|b$" şeklinde sıralama bağıntısını ele alalım. Bu bağıntıyı kullanarak örnek 12 de olduğu gibi doğal sayılar kümesi üzerinde bir kategori tanımlanabilir.

15. Denklik bağıntısı; M bir küme ve R, bu küme üzerinde bir denklik bağıntısı olsun. Ob($\mathcal{M}$)=M olsun. $(m,m')\epsilon R\ ise\ Mor(m,m') = \{(m,m')\}\ ve\ (m,m') \notin R\ ise\ Mor(m,m') = \emptyset$ olsun. Bu tanımlamalarla $\mathcal{M}$ bir kategoridir.

$\mathcal{L}$ bir kategori olsun. $A \neq B$ özelliğindeki her $A,B\epsilon Ob(\mathcal{L})$ için $Mor(A,B) = \emptyset\ ve\ her\ A\epsilon Ob(\mathcal{L})$ için $Mor(A,A) = \{I_A\}$ ise $\mathcal{L}$ ye discrete(ayrık) kategori denir.

Örnek 12 gereğince her sınıf bir discrete kategori tanımlar. Tersine her discrete kategori de bir sınıf olarak ifade edilebilir.

Örnek 12, 13, 14, ve 15 te görüleceği üzere, bu kategorilerin ortak özellikleri objelerinin küme oluşturmasıdır. Eğer bir kategorinin objeleri bir küme oluşturuyorsa bu kategoriye small kategori veya diyagram scheme denir.

Tanım 2.1.4. : (Monic) $\mathcal{C}$ bir kategori ve f bir morfizm olsun. f nin sol sadeleştirme özelliği varsa f ye monic denir. Küme kategorisinde f nin monic olması için gerek ve yeter şart injektif olmasıdır. Aynı zamanda bu ifade R-mod kategorisi içinde geçerlidir. Fakat bu denkliğin olmadığı kategorilerde vardır.

Örnek 2.1.5. : G bir Abelyan grup olsun. Her $g\epsilon G\ ve\ her\ n\epsilon N$ için $g = ng'$ olacak şekilde $g'\epsilon G$ varsa g ye division yani bölünebilir denir.

DivAb bölünebilir Abelyen gruplar kategorisi olsun. $Q\ ve\ {}^{Q}/_{Z}$ bu kategorinin objelerindendir.

$$\frac{p}{q}\epsilon Q\ \ ve\ n\epsilon N\ için\ \ \frac{p}{q} = n\left(\frac{p}{nq}\right)$$

$$\frac{p}{z} + Z = n\left(\frac{p}{nq} + Z\right)$$

$$h = Q \longrightarrow Q/Z$$

fonksiyonunu $q\epsilon Q\ \ için\ \ h(q) = q + Z$ şeklinde tanımlayalım. Bu morfizm monic olup injektif değildir.

$$f,g: A \longrightarrow Q$$

Dönüşümleri **DivAb** da iki morfizm olsun ve $f \neq g$ olsun. Şimdi h nin soldan sadeleştirilebilir yani monic olduğunu gösterelim. $f \neq g$ için $hof \neq hog$ olduğunu göstermemiz yeterli. $f \neq g$ olduğundan $s \neq 1 \; ve \; s \neq -1 \; ve \; f(a) - g(a) = \frac{r}{s} \neq 0$ olacak biçimde $a \epsilon A$ vardır. Gerçekten $f \neq g \Rightarrow \exists a \epsilon A \; için \; f(a) \neq g(a)$ dolayısıyla $f(a) - g(a) \neq 0$ olur. Bu durumda

$$f(a) - g(a) = \frac{r}{s} \neq 0 \qquad \frac{r}{s} \epsilon Q$$

vardır. $s \epsilon N \; olsun. s \neq 1$ olsun. $b = sa$ olacak şekilde $a \epsilon A$ vardır.

$$f(sa) - g(sa) = r$$

$$f(a + \cdots + a) - g(a + \cdots + a) = r$$

$$sf(a) - sg(a) = r \quad \Rightarrow \quad f(a) - g(a) = \frac{r}{s}$$

Gerçektende $f \neq g$ ise en az bir $b \epsilon A$ için $f(ab) \neq g(b)$ dir. Dolayısıyla $f(b) - g(b) \neq 0$ ve $f(b) - g(b) = r \neq 0$ olacak biçimde $r \epsilon Q$ vardır. $s \neq 1$ ve $s \epsilon N$ olsun. A bölünebilir olduğundan $\exists a \epsilon A$ için $sa = b$ dir. Buradan

$$f(b) - g(b) = r$$

ise

$$f(sa) - g(sa) = r \quad \Rightarrow \quad f(a + \cdots + a) - g(a + \cdots + a) = r$$

$$[f(a) + \cdots + f(a)] - [g(a) + \cdots + g(a)] = r \quad \Rightarrow \quad sf(a) - sg(a) = r$$

$$f(a) - g(a) = \frac{r}{s} \neq 0 \quad ve \;\; s \neq 1 \quad ve \;\; s \neq -1$$

Yine A bölünebilir olduğundan $a = rc$ olacak biçimde $c \epsilon A$ vardır.

$$r\big(f(c) - g(c)\big) = rf(c) - rg(c) = f(rc) - g(rc) = f(a) - g(a) = \frac{r}{s}$$

olduğundan $f(c) - g(c) = \frac{1}{s}$ dir. yani $h(f(c)) \neq h8g(c))$ dir. (Eğer $\frac{1}{s} \epsilon Z\ olsaydı\ f(c) - g(c) \epsilon Z\ olduğundan\ f(c) + Z = g(c) + Z\ \ yani\ h\big(f(c)\big) = h\big(g(c)\big)\ olurdu)$ Sonuç olarak h moniktir.

$h: Q \longrightarrow Q/Z$ dönüşümü injektif değildir. $1,2 \epsilon Q\ \ ve\ 1 \neq 2\ olmasına\ karşın\ f(1) = f(2)\ dir.$ Dolayısıyla f injektif değildir.

Tanım 2.1.6. : $\mathcal{C}$ bir kategori olmak üzere $\mathcal{C}$ deki $h: A \longrightarrow B$ morfizmi sağdan sadeleştirme özelliğine sahipse yani;

$$g, f: A \longrightarrow B$$

Ve

$$foh = goh$$

olduğunda $f = g$ oluyorsa h ye epik denir. Küme kategorisinde bir dönüşümün epik olması için gerek ve yeter şart surjektif olmasıdır. (Küme kategorisinde) Fakat aynen monikte olduğu gibi bu ifade her kategoride geçerli değildir.

Örnek 2.1.7. : Halka kategorisinde $i: Z \longrightarrow Q$ içine dönüşümünü düşünelim. Bu dönüşüm surjektif değildir fakat epiktir. $(z \epsilon Z\ \ için\ \ i(z) = z)$

Dönüşümün surjektif olmadığı açıktır. Şimdi dönüşümün epik olduğunu yani sağ sadeleştirme özelliğine sahip olduğunu gösterelim.

$i: Z \longrightarrow Q$ ve $g, h: Q \longrightarrow A$ dönüşümleri halka kategorisinde iki morfizm ve $goi = hoi$ olsun. Her $n \epsilon Z\ için\ (goi)(n) = (hoi)(n)\ \ ve\ buradan\ g(n) = h(n)\ olur.\ \ \frac{m}{n} \epsilon Q\ için$

$$g\left(\frac{m}{n}\right) = g(m.n^{-1}.1) = g(m)g(n^{-1})g(1)$$

$$= h(m)g(n^{-1})g(1) = h(m)g(n^{-1})h(n.n^{-1})$$

$$= h(m)g(n^{-1})h(n)h(n^{-1})$$

$$= h(m)g(n^{-1})g(n)h(n^{-1})$$

$$= h(m)g(n^{-1}.n)h(n^{-1})$$

$$= h(m)g(1)h(n^{-1})$$

$$= h(m)h(1)h(n^{-1})$$

$$= h(m.1.n^{-1}) = h(\frac{m}{n})$$

yani $\frac{m}{n}\epsilon Q$ için $g\left(\frac{m}{n}\right) = h(\frac{m}{n})$ elde edilir ki bu da i dönüşümünün epik olduğunu gösterir.

2.2 ÜÇLÜLER (Monad veya Triple)

Bu bölümde tezin kalan kısmı için önemli bir yer teşkil edecek olan üçlü (Monad veya Triple) kavramı üzerinde durulacaktır. Monad kavramı Monaid kavramının genellemesi olarak tanımlanabilir.

Bilindiği üzere monaid, üzerinde asosyatiflik özelliği sağlayan bir çarpımın tanımlı olduğu ve bu işleme göre birim elemanın olduğu bir kümedir. Şimdi herhangi bir küme yardımıyla bir monaid tanımlayalım. (Buradaki monoid yapısı oluştururken [Borceux2] bölüm 4 ten faydalanılmıştır.)

M herhangi bir küme olsun. M nin elemanlarından oluşan tüm sonlu dizilerin kümesi T(M) olsun. $f: M \to N$ bir fonksiyon olmak üzere $T(f): T(M) \to T(N)$ dönüşümü

$$\big(T(f)\big)\big(f(x_1), \dots, f(x_n)\big)$$

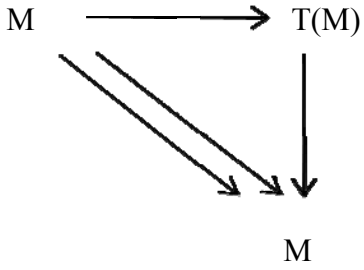
Şeklinde tanımlansın. Buradaki asıl amacımız $\xi: T(M) \to M$ şeklinde bir fonksiyon yardımıyla üzerinde bir monoid tanımlamaktır. $(x_1, \dots, x_n)\epsilon T(M)$ için $\xi(x_1, \dots, x_n)$ elemanına $(x_1, \dots, x_n)$ dizisinin bileşke elemanı denir.

Öncelikle "normalizasyon şartı" olarak bilinen $\xi(x) = x$ şartı sağlanmalıdır. Bu ifadeyi

$$\xi_M: M \to T(M)$$

$$x \longmapsto (x)$$

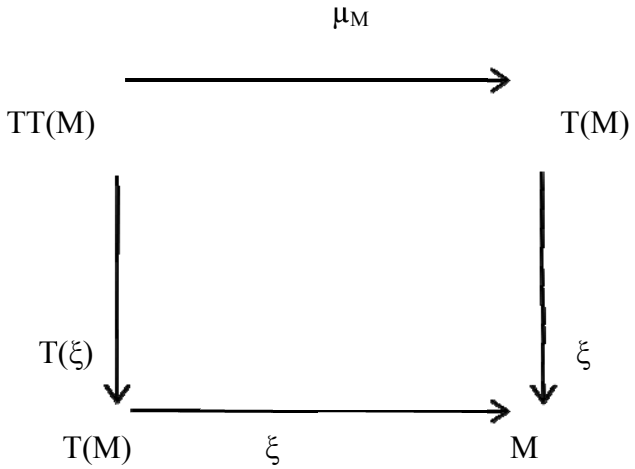
Fonksiyonunu kullanarak,



Diyagramının değişmeli olması şeklinde düşünebiliriz. Şimdi daha genel bir asosyatiflik şartının sağlandığını, yani,

eşitliğinin varlığı gösterilmelidir. M deki elemanların sonlu dizilerinde sonlu diziler seçmek, TT(M) den bir eleman seçmekle aynı işlemdir. Dolayısıyla bu işlemi

Şeklinde dizilerin dizisinden birleşik bir dizi oluşturan bir fonksiyon olarak düşünebiliriz. Genel asosyatiflik aksiyomu



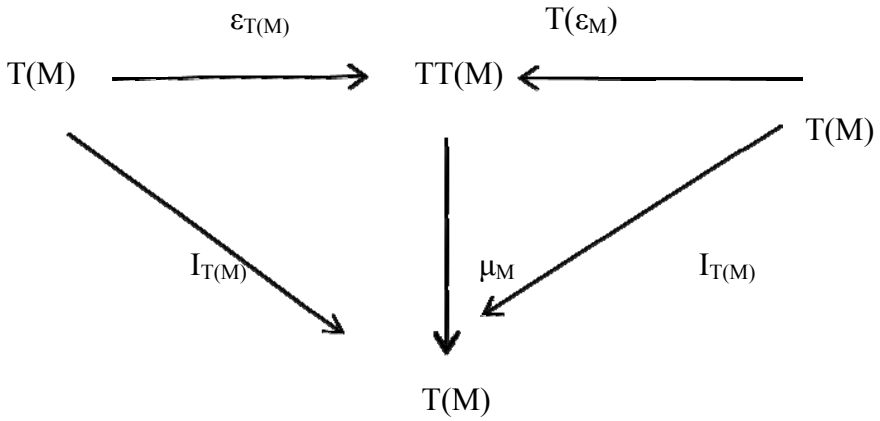
dır. Bu yapılan tanımlamaların klasik monoid tanımı ile denk olduğu kolayca gösterilebilir. Burada çarpımı

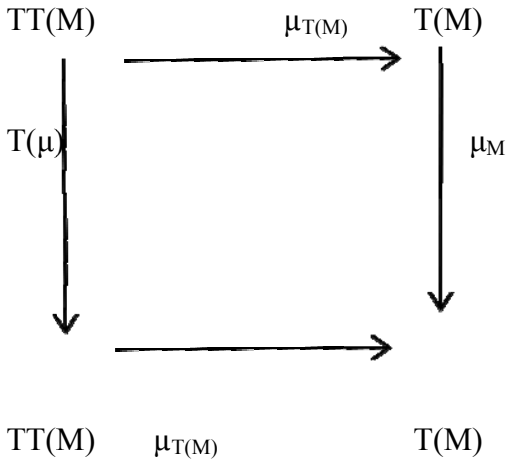
şeklinde tanımlayalım.

olduğundan tanımlanan işlem asosyatiftir. Boş dizinin bileşkesi olsun.

Ve benzer şekilde x1=x bulunur.

Burada tanımlanan dönüşümleri aranan iki aksiyoma karşılık gelen ve bu dönüşümler

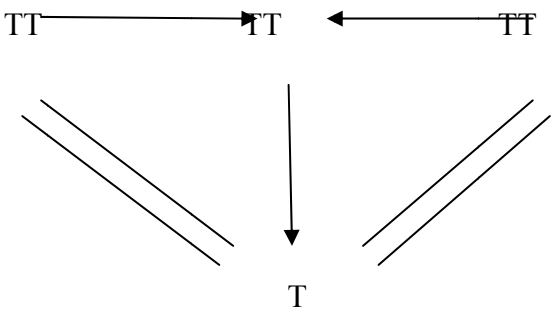


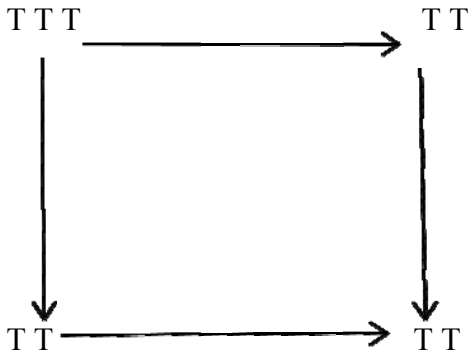


diyagramlarını değişmeli yapar. Şimdi verdiğimiz bu monoid tanımının bir genellemesi olarak düşünebileceğimiz "üçlü" (monoid, triple) tanımını verelim.

Tanım 2.2.1. : $\mathcal{C}$ bir kategori, $\mathcal{C}$ $\mathcal{C}$ bir functor doğal transformasyonlar olmak üzere

şartları sağlanıyorsa, bir başka ifade ile

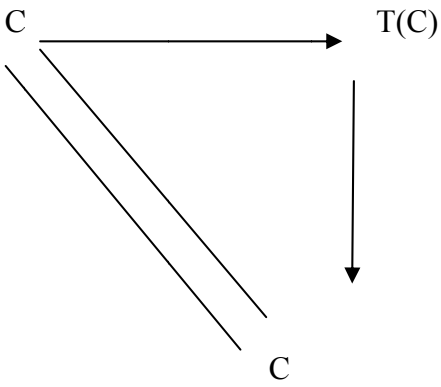


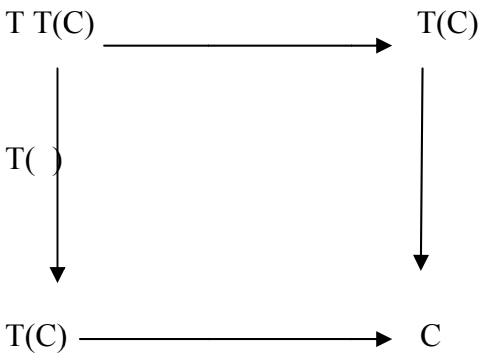


Diyagramları değişmeli ise (T, ,) üçlüsüne $\mathcal{C}$ üzerinde bir monad denir. Birinci şart "birimlik" ikinci şart ise asosyatiflik şartı olarak adlandırılır.

Tanım 2.2.2. : T( ), $\mathcal{C}$ kategorisi üzerinde bir monad olsun. $\mathcal{C}$),
olmak üzere

şartları sağlanıyorsa (C,) ikilisine T monaidi üzerinde bir cebir denir.

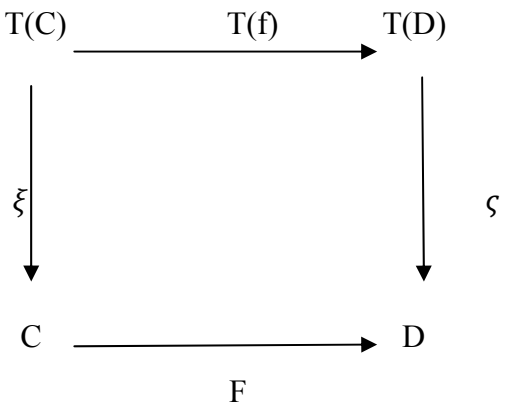




Tanım 2.2.3. : (D,ς) ve (C,ξ), T üzerinde iki cebir olsun. $f \epsilon Hom(C, D)$ morfizması

$$f o \xi = \varsigma o T(f)$$

şartını sağlıyorsa, yani,



diyagramı değişmeli ise, $f: C \to D$ ye T- cebir morfizmi denir.

Önerme 2.2.4. : T=(T,ε, μ) bir $\mathcal{C}$ kategorisi üzerinde bir monad olsun. T- cebirler ve morfizmları bir kategori oluşturur.

Oluşturulan bu kategori genelde $\mathcal{C}^{\mathrm{T}}$ ile gösterilir. Bu kategori aynı zamanda **T**(T, ε, μ) monadının "Eilenberg- Moore" kategorisi denir.

Örnek 2.2.5. : M=(M,1,.) bir monaid olmak üzere her A kümesi için $T(A) = M \times A$ ve

$$\varepsilon_A: A \to M \times A$$

$$a \longmapsto (1, a)$$

Ve

$$\mu_A: M \times (M \times A) \longrightarrow M \times A$$

$$\big(m, (m', a)\big) \longmapsto (m.m', a)$$

olsun. Böylelikle (T, ε, μ) üçlüsü **Küme** kategorisi üzerinde bir monad olup birimlik ve asosyatiflik şartlarını sağlar.

$$\mu o(\varepsilon * T)(A) = \mu o(\varepsilon\big(T(A)\big) = \mu o(\varepsilon(m \times A) = \mu o\big(\varepsilon(\{m \times a | m\epsilon M, a\epsilon A\})\big)$$
$$= \mu o\big(\{\big(1, (m, a)\big) : m\epsilon M, a\epsilon A\}\big) = \{(1.m, a) | m\epsilon M, a\epsilon A\}$$
$$= \{(m, a) | m\epsilon M, a\epsilon A\} = T(A)$$

$$\mu o(T * \varepsilon)(A) = \mu o(T\big(\varepsilon(A)\big) = \mu o\big(T(\{(1, a) | a\epsilon A\})\big) = \mu o(m \times \{(1, a) | a\epsilon A\})$$
$$= \mu o\big(\{\big(m, (1, a)\big) : m\epsilon M, a\epsilon A\}\big) = \{(m.1, a) | m\epsilon M, a\epsilon A\}$$
$$= \{(m, a) | m\epsilon M, a\epsilon A\} = T(A)$$

olduğundan

$$\mu o(\varepsilon * T) = 1_T = \mu o(T * \varepsilon)$$

bulunur.

$$\mu o(\mu * T) = \mu\big(\mu(m \times (m \times T(A))\big) = \mu\big(\mu(\{(m'', (m', (m, a))) | m, m', m''\epsilon M, a\epsilon A\})\big)$$
$$= \mu(\{(m''.m', (m, a)) | m, m', m''\epsilon M, a\epsilon A\})$$
$$= \big\{\big((m''.m').m, a\big) | m, m', m''\epsilon M, a\epsilon A\big\}$$

$$\mu o(T * \mu) = \mu\left(T\big(\mu(T^2(A))\big)\right) = \mu\left(T\mu\big(m \times (m \times A)\big)\right)$$
$$= \mu\left(T\mu\big(\{\big(m', (m, a)\big) | m, m'\epsilon M, a\epsilon A\}\big)\right)$$
$$= \mu T(\{(m'.m, a) | m, m'\epsilon M, a\epsilon A\})$$
$$= \mu(M \times \{(m'.m, a) | m, m'\epsilon M, a\epsilon A\})$$
$$= \mu\big(\{\big(m'', (m'.m, a)\big) | m, m', m''\epsilon A, a\epsilon A\}\big)$$
$$= \{(m''.(m'.m), a) | m, m', m''\epsilon A, a\epsilon A\}$$

M monoid olduğundan $m, m', m''\epsilon A$ için

$$(m''.m').m = m''.(m'.m)$$

olur. Dolayısıyla

$$\mu o(\mu * T) = \mu o(T * \mu)$$

bulunur. Sonuç olarak birimlik ve asosyatiflik şartları sağlandığından (T, ε, μ) yapısı **Küme** kategorisi üzerinde bir monaiddir.

Şimdi tanımlanan bu monad üzerinde cebir tanımlayalım. $\xi(m, a) \equiv ma$ yazalım. Ve bu çarpımın

$$1a = a, (m.m')a = m(m'a), \qquad m, m' \epsilon M, a \epsilon A$$

Şartlarını sağladığını kabul edelim. Böylece (A, ξ) bir cebir olur. Bir başka ifade ile tanımlanan T(T,ε, μ) monad üzerinde, bir M- kümedir.

3. KARTEZYEN KAPALI KATEGORİLER VE λ -KALKÜLÜS

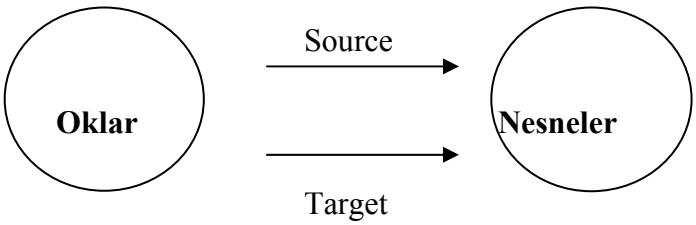
Bu bölümde deductive anlamında önermesel kalkülüs, CCC kategoriler, Doğal sayı nesneleri, typed $\boldsymbol{\lambda}$-kalkülüs, Doğal sayı nesneleri, Recursive fonksiyonlar ve bunların bilgisayar uygulamaları incelenmiştir.Bu bölümde faydalandığımız temel kaynaklar, (R. Voreadou,1977), (H. Volger, 1975), (A. Wolf, 1974),(Peter R. 1951),(Odifreddi P.G. 1989),(Rogers H, 1987),(Soare R.1987),(Kleene S. 1952),(Evren Ş, 2006) dır.

3.1.BİR DEDUCTİVE SİSTEM ANLAMINDA ÖNERMESEL KALKÜLÜS

Giriş 3.1.1.

Bu bölümde temel bazı matematiksel yapıların kategori dilindeki karşılıkları verilecektir. Bu bağlamda CCC kategorileri ve bu kategorilerdeki sınıflandırılmış $\boldsymbol{\lambda}$-kalkülüs sistemi detaylı biçimde incelenecektir.

Tanım 3.1.2. : Graf iki sınıf ve bunlar arasında tanımlanan iki dönüşümden oluşur. Bu dönüşümler Source ve Target dönüşümleri olup sınıflar ise **Ok** lar ve **Nesneler** sınıfı olarak adlandırılır.



$f \epsilon \boldsymbol{Oklar}$ ve A,B ϵ **Nesneler** ve $source(f) = A,\ target(f) = B\ durumu\ f: A \longrightarrow B$ şeklinde gösterilir.

Tanım 3.1.3. : $\mathcal{C}$ bir graf olsun.

$R1a$, $I_A: A \longrightarrow A$ şeklinde bir özel ok ve

$R1b$, $f: A \longrightarrow B,\ g: B \longrightarrow C$ için $gf: A \longrightarrow C$ şeklinde bir işlemle birlikte $\mathcal{C}$ grafına bir **deductive** sistem denir.

Deductive sistemin nesneleri formüller, oklar, ispatlar(veya deduction) ve oklar üzerinde tanımlanan işlem ise inferencenin bir kuralı olarak düşünülebilir.

Conjunction(Bağlaçlar) kalkülüs, doğruluk ve conjunction ile ilgilenen bir deductive sistemdir. Böylece A ve B gibi verilen formüllerin $A \wedge B$ conjunctionını oluşturmak için $\wedge(= ve)$ ikili işlemi ve $T(= doğru)$ formülünün verildiğini kabul ediyoruz. Ayrıca aşağıdaki kural ve oklar sistem içinde mevcuttur

$R.2$ $\quad O_A: A \longrightarrow$ T

$R3a.$ $\quad \pi_{A,B}: A \wedge B \longrightarrow A$

$R3b.$ $\quad \pi'_{A,B} A \wedge B \longrightarrow B$

$R3c.$ $\quad f: C \longrightarrow A, g: C \longrightarrow B \ \ varsa$

$\langle f, g \rangle: C \longrightarrow A \wedge B \ mevcuttur.$

Örnek 3.1.4. : Conjunction için değişme özelliğinin varlığını ispatlayalım.

R3b ve R3a dan $\pi_{A,B}: A \wedge B \longrightarrow A \ \ ve \ \ \pi'_{A,B} A \wedge B \longrightarrow B$ vardır. R3c gereğince

$$\langle \pi'_{A,B}, \pi_{A,B} \rangle: A \wedge B \longrightarrow B \wedge A$$

bulunur. Yani $\wedge$ bağlacı için $A \wedge B$ varsa $B \wedge A$ nın da doğru olduğu bu şekilde ispatlanabilir.

Örnek 3.1.5 : Şimdi değişme özelliğinin varlığını ispatlayalım. Yani

$$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$$

olduğunu gösterelim. Burada $(A \wedge B) \wedge C$ ve $A \wedge (B \wedge C)$ birer formül (nesne) dir. Bu iki formül arasında bir ispat (ok) tanımlamalıyız. Burada $(A \wedge B) \wedge C$ nin doğru olduğunu kabul ediyoruz.

$\pi_{A \wedge B, C}: (A \wedge B) \wedge C \longrightarrow A \wedge B$

$\pi'_{A\wedge B,C}: (A\wedge B)\wedge C \longrightarrow C$

$\pi_{A,B}: A\wedge B \longrightarrow A$

$\pi'_{A,B}: A\wedge B \longrightarrow B$

okları ve bileşimleri yardımıyla

$\pi_{A,B}\pi'_{A\wedge B,C}: (A\wedge B)\wedge C \longrightarrow A$

$\pi'_{A,B}\pi_{A\wedge B,C}: (A\wedge B)\wedge C \longrightarrow B$

buradan hareketle

$\langle \pi_{A,B}\pi_{A\wedge B,C}, \langle \pi'_{A,B}\pi_{A\wedge B,C}, \pi'_{A\wedge B,C}\rangle\rangle : (A\wedge B)\wedge C \longrightarrow A\wedge(B\wedge C)$

bulunur. Yani $\alpha_{A,B,C}: (A\wedge B)\wedge C \longrightarrow A\wedge(B\wedge C)$ ispatı (oku) bulunur.

Örnek 3.1.6 : $f: A \longrightarrow B \ ve \ g: C \longrightarrow D$ mevcut ise yani A formülünün doğruluğundan hareketle f ispatı ile B formülü bulunuyorsa, yine C nin doğruluğundan hareketle g ispatı ile D formülü bulunuyorsa $A\wedge B$ den $B\wedge D$ nin doğruluğu bulunabilir ve ispat $f\wedge g$ şeklinde gösterilir.

$\pi_{A,C}: A\wedge C \longrightarrow A \ ve \ f: A \longrightarrow B \ den \ f\pi_{A,C}: A\wedge C \longrightarrow B$

$\pi'_{A,C}: A\wedge C \longrightarrow C \ ve \ g: C \longrightarrow D \ den \ g\pi'_{A,C}: A\wedge C \longrightarrow D$

bulunur. R3c gereğince $f\wedge g: A\wedge C \longrightarrow B\wedge D$ bulunur. Burada $f\wedge g = \langle f\pi_{A,C}, g\pi'_{A,C}\rangle$ dir.

Bir conjunction kalkülüs üzerinde $\Leftarrow (ise)$ ikili işlemi tanımlayarak bir pozitif intuitionistic önermesel kalkülüs elde edilir. Böylece A ve B birer formül ise, $T, A\wedge B \ ve \ A \Leftarrow B$ de birer formüldür. Burada $\Leftarrow$ bağlacıyla ilgili olarak

$R4a.$ $\quad \varepsilon_{A,B}: (A \Leftarrow B)\wedge B \longrightarrow A$

$R4b.$ $\quad h: C\wedge B \longrightarrow A \ varsa \ h*: C \longrightarrow A \Leftarrow B$ mevcuttur.($h* = \wedge^{C}_{A,B}(h)$ şeklinde ifade edilir.)

Burada R4a yardımıyla R4b den

$R'4b.$ $\quad \eta_{C,B}: C \longrightarrow (C\wedge B) \Leftarrow B$

$R'4c.$ $\quad g: D \longrightarrow A \quad mevcut\ ise\ \ g \Leftarrow I_A: (D \Leftarrow B) \longrightarrow (A \Leftarrow B)$

sonuçları elde edilir. Burada

$\eta_{C,B} = I^*_{C \wedge B}, \qquad g \Leftarrow I_A \equiv \left(g\varepsilon_{B,D}\right)^*$

dir. Eğer bu sistem üzerinde 1(yanlış) formülü ve bir $\vee$ $(veya)$ işlemi tanımlanırsa bu sisteme bir intuitionistic kalkülüs denir. Bu arada 1 formülü ve $\vee$ bağlacı

$R5.$ $\quad \square_A: A \longrightarrow 1$

$R6a.$ $\quad \kappa_{A,B}: A \longrightarrow A \vee B$

$R6b.$ $\quad \kappa'_{A,B}: B \longrightarrow A \vee B$

$R6c.$ $\quad \zeta^C_{A,B}: (A \Rightarrow C) \wedge (B \Longrightarrow C) \longrightarrow (A \vee B) \Rightarrow C$

buradan hareketle

$f: A \longrightarrow C \quad ve \quad g: B \longrightarrow C$

varsa

$[f, g]: A \vee B \longrightarrow C$

vardır.

3.2.KARTEZYEN KAPALI KATEGORİLER(CCC)

Tanım 3.2.1. : Kategori, ispatları arasında aşağıdaki ilişkilerin mevcut olduğu bir deductive (R1a ve R2b şartlarını sağlayan graf) bir sistemdir.

$her\ f: A \longrightarrow B \quad g: B \longrightarrow C \quad h: C \longrightarrow D \quad için$

$fI_A = f \qquad I_B f = f \qquad (hg)f = h(gf)$

Bu noktada öncelikli olarak Kartezyen kapalı kategorilerin klasik tanımını ve daha sonrasında conjunction kalkülüs cinsinden tanımını verip birbiriyle denkliğini vereceğiz.

Öncelikle klasik tanımı vereceğiz.

Tanım 3.2.2. : $\mathcal{C}$ bir kategori olsun. $\mathcal{C}$ de tüm sonlu çarpımlar mevcut (böylece terminal object mevcut) ve her $B \epsilon ob(\mathcal{C})$ için $(-) \times B: \mathcal{C} \rightarrow \mathcal{C}$ functorunun $(-)^B: \mathcal{C} \rightarrow \mathcal{C}$ sağ ek functoru varsa, yani

$Hom(A \times B, C)$ ve $Hom(A, C^B)$ kümeleri izomorf ise $\mathcal{C}$ kategorisine Kartezyen kapalı kategori denir.

Şimdi deductive sistem veya conjunction kalkülüs cinsinden Kartezyen kategorileri ve Kartezyen kapalı kategorileri tanımlayalım.

Tanım 3.2.3. : C bir kategori ve bir conjunction kalkülüs olsun. Eğer her

$f: C \longrightarrow A,\ g: C \longrightarrow B,\ h: C \longrightarrow A \wedge B$ için

$E3.\ her\ f: A \longrightarrow T\ \ için\ f = 0_A$

$E3a.\ \pi_{A,B}\langle f, g\rangle = f$

$E3b.\ \pi'_{A,B}\langle f, g\rangle = g$

$E3c.\ \langle \pi_{A,B}h, \pi'_{A,B}h\rangle = h$

Burada dikkat edileceği üzere T, terminal nesne $A \wedge B$ ise A ve B nesnelerinin(formüllerinin) çarpımına karşılık gelmektedir.

Bu tanımlama eşitlikler cinsinden tanımlama olarak bilinir. Klasik kategori dili ile anlatılacak olursa; çarpım objeleri ve terminal objesi olan kategorilere Kartezyen kategori denir.

Tanım 3.2.4. : $\mathcal{C}$, R4 yapısına sahip üzerinde $\Leftarrow$ işlemi ve özellikleri tanımlı olan bir Kartezyen kategori olsun. Eğer her $h: C \wedge B \longrightarrow A$ ve $k: C \longrightarrow A \Leftarrow B$ için

$E4a.\quad \varepsilon_{A,B}\langle h^{*}\pi_{C,B}, \pi'_{C,B}\rangle = h$

$E4b.\quad \left(\varepsilon_{A,B}\langle k\pi_{C,B}, \pi'_{C,B}\rangle\right)^{*} = k$

şartları sağlanıyorsa $\mathcal{C}$ ye bir Kartezyen kategori denir ve literatürde CCC ile (Cartesian closed category) gösterilir.

Klasik kategori dili ile anlatılacak olursa; $A \Leftarrow B$ den kasıt A^B yeni *Hom(B,A)* olup ek şartlarından kasıt ise anlaşılacağı üzere $Hom(C \times B, A)\ ve\ Hom(C, Hom(B, A)$ kümeleri arasında birebir eşlemenin olması bir başka ifade ile bu kümelerin izomorf olmasıdır.

Şimdi CCC örnekleri verelim.

1. Kümeler kategorisi olan **Set** bir CCC dir. Bilindiği üzere A,B herhangi iki küme olmak üzere $A \times B$ kartezyen çarpımı **Set** kategorisinde A ve B nesnelerinin çarpımına karşılık gelmektedir. A,B,C birer küme olmak üzere $Hom(C \times B, A)\ ve\ Hom(C, Hom(B, A))$ arasındaki bire bir eşlemeyi kuralım.
   $f: C \times B \longrightarrow A$ olsun. Şimdi f ye karşılık $f^{*}: C \longrightarrow Hom(B, A)$ tanımlamalıyız. Her bir $c \epsilon C$ nin f^{*} altındaki görüntüsü $f_C^{*}: B \longrightarrow A$ olsun. f_C^{*} fonksiyonunu $f_C^{*}(b) = f(c, b)$ olsun. Bu durumu fonksiyon şeklinde ifade edelim.

$$\varphi: Hom(C \times B, A) \longrightarrow Hom(C, Hom(B, A))$$

$$(f: C \times B \longrightarrow A) \longmapsto (f^{*}: C \longrightarrow Hom(B, A))$$

$$c \longmapsto (f^{*}: B \longrightarrow A)$$

$$b \longmapsto f(c, b)$$

Bu noktada

$$\psi: Hom(C, Hom(B, A)) \longrightarrow Hom(C \times B, A)$$

$$g \longmapsto (g': C \times B \longrightarrow A)$$

$$(c, b) \longmapsto \big(g(c)\big)(b)$$

Olsun. $\varphi\psi(g) = g$ ve $\psi\varphi(f) = f$ olduğunda φ dönüşümü bire bir eşlemedir. Yani $Hom(C \times B, A)$ ve $Hom(C, Hom(B, A))$ izomorftur. Böylece **Set** kategorisi bir CCC dir.

Örnek 3.2.5. : Benzer şekilde **Top** topolojik uzaylar kategorisininde CCC olması beklenebilir. Fakat B^A üzerindeki topoloji kompakt-open topoloji olup istenen şartları **Top** kategorisinin yerel kompakt uzaylardan oluşan veya kompakt üretilmiş hausdorff uzaylardan oluşan alt kategori ile CCC dir.

Örnek 3.2.6. : Grp, gruplar kategorisi CCC değildir. Bu kategoride başlangıç nesnesi (initial object) tek elemanlı gruptur. Başlangıç grubu olmayan bir G grubu için $(-) \times G$ functoru başlangıç objesini korumaz. Yani başlangıç nesnesinin bu functor altındaki görüntüsü başlangıç nesnesi olmayabilir. Yani bu functorun sağ ek functoru yoktur. Dolayısıyla **Grp** kategorisi CCC değildir.

Örnek 3.2.7. : Benzer şekilde **Abgrp** kategorisin de bir CCC değildi.

Örnek 3.2.8. : Kısmı sıralı kümeler kategorisi **Pos** bir CCC dir ve Y^X, monoton $X \longrightarrow Y$ dönüşümlerinden oluşmaktadır. ($f \leq g \Leftrightarrow f(x) \leq g(x)$ $her\ x \epsilon X$)

Örnek 3.2.9. : Her Boolean cebir bir CCC dir.

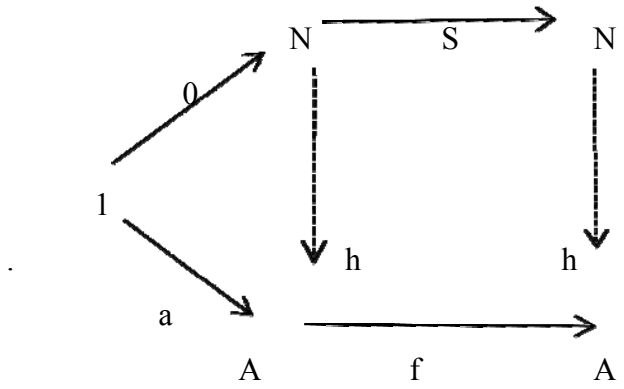
Örnek 3.2.10. : Small kategoriler kategorisi **Cat** bir CCC dir.

3.3. CCC LERDE DOĞAL SAYILAR NESNELERİ

$\mathcal{C}$ bir CCC olsun. $\mathcal{C}$ deki tüm $1 \overset{a}{\to} A \overset{f}{\to} A$ biçiminde tanımlı diyagramlar kategorisinin başlangıç nesnesi olan

$$1 \overset{O}{\to} N \overset{S}{\to} N$$

Objesine bir doğal sayılar nesnesi denir. Bu durum



Diyagramı ile resmedilebilir. Burada diyagramı değişmeli yapacak biçimde bir tek h olmalı. h nin sadece varlığını biliyorsak (tek olması söz konusu değil) ye zayıf doğal sayılar nesnesi denir.

3.4.GRAFLARLA ÜRETİLEN SERBEST CCC LER

Bir X grafı verilsin. Bu graf yardımıyla pozitif intutionistic kalkülüs D(x) ve CCC F(x) üretilebilir.

D(x) in formül ve ispatları şu şekilde tanımlanır; X in tüm köşeleri (vertexleri) formüller, bir formül, A ve B formül ise yani yine bir formül ve yani da bir formül, X in tüm okları, ler ise A ve B formülleri için ispatlar ve tüm ispatlar ve kuralları altında kapalı.

D(x) yapısına CCC için gerekli tüm eşitlikler eklenerek F(x) Kartezyen kapalı kategorisi elde edilir.

3.5.TYPED -KALKÜLÜS

Sınıflandırılmamış λ-kalkülüs fonksiyonel programlama için bir temel oluşturması sebebiyle (teorik) bilgisayar bilimlerinde bir yeniden doğuş(Rönesans) olarak kabul edilmektedir. Bir CCC nin nesneleri (objects) ve morfizmleri sırasıyla λ-kalkülüsün type ve terimleri olarak algılanmaktadır.

Bir formal teori olan typed λ-kalkülüsü şu şekilde tanımlayabiliriz; bu teori (veya sistem) "types" diye adlandırılan sınıflardan ve her bir type için "terimler" sınıfından ve terimler arasındaki "eşitlikler" sınıfından oluşmaktadır ve bu sistem aşağıdaki şartları sağlamalıdır.
($a\epsilon A$ $ifadesi$ a $nın$ bir $terim$ $olduğunu$ ve A $type$ $olduğunu$ $ifade$ $etnektedir.$)

(a) Typelar; type ların sınıfı iki temel type mutlaka içerir ve aşağıdaki biçimde tanımlanan işlemler (operation) altında kapalıdır.

(a1) 1 ve N type dir.(Bunlar temel type ler)

(a2) A ve B type ise A×B ve B^A type dir.

(b) Terimler; Terimler sınıfı değişkenler ve belirli temel sabitler kullanılarak belirli terim oluşturma işlemleri yardımıyla aşağıdaki biçimde serbest (freely) üretilmiştir.

(b1) Her bir A tipine ait sayılabilir sayıda $x_i^A \epsilon A$ $(i = 0,1,2,3, \dots)$ değişkenleri mevcuttur. (Değişkenleri $1 \rightarrow A$ $morfizmi$ $olarak$ düşünü$lebilir.$)

(b2) $* \epsilon 1$

(b3) $a\epsilon A, b\epsilon B$ ve $c\epsilon A \times B$ ise $\langle a, b\rangle \epsilon A \times B,$ $\pi_{A,B}(c)\epsilon A$ ve $\pi'_{A,B}(c)\epsilon B$

(b4) Eğer $f\epsilon b^A$ ve $a\epsilon A$ ise $\varepsilon_{B,A}(f, a)\epsilon B$

(b5) Eğer $x\epsilon A$ ve $\varphi(x)\epsilon B$ ise $\lambda_{x\epsilon A}\varphi(x)\epsilon B^A$

(b6) $0\epsilon N,$ $n\epsilon N$ ise $s(n)\epsilon N$

(b7) $a\epsilon A,$ $h\epsilon A^A$ ve $n\epsilon N$ ise $I_A(a, h, n)\epsilon A$

Nasıl ki kategori teori kümenin genellemesi ise λ-kalkülüs te kümeler üzerinde tanımlı olan fonksiyon cebirinin genellemesi olarak düşünülebilir. Mesela $\lambda_{x\epsilon A}\varphi(x)$, $x \longmapsto \varphi(x)$ ifadesini , $\varepsilon_{B,A}(f, a)\epsilon B$ terimi f(a) değerini temsil etmektedir.

Çeşitli yazarlar ihtiyaçlara göre λ-kalkülüs şartlarını değişik biçimlerde genişletip daraltmaktadırlar. Burada biz daha çok [Lambek] te ve [BRICS] te yer alan tanımları kullanacağız. Ve bundan genel olarak verilen ifade ve önermelerin daha iyi anlaşılması için örneklemeler yazacağız.

Şimdi λ-kalkülüs teki son kısım olan " Eşitlikler" sınıfının özelliklerine bakalım.

$a = b \ ise \ \varepsilon_{B,A}(f, a) = \ \varepsilon_{B,A}(f, b)$ ve $\varphi(x) = \varphi'^{(x)} \ ise \ \lambda_{x\epsilon A}\varphi(x) = \lambda_{x\epsilon A}\varphi'(x)$

eşitlikleri sağlanmalıdır.

Bu tanımlamalar özellikle fonksiyonel programlamada (teorik bilgisayar bilimlerinde) kullanılmaktadır. Örnek olarak en temel anlamda bu programlamayı bilgisayara fonksiyonun ne demek olduğunu temel işleçler ve sınıflar cinsinden tanımlamak olarak düşünülebilir. Bu bağlamda daha geniş uygulamalar ve kullanım alanları [category for software], [ ], [ ], [ ] kaynaklarında bulunabilir.

3.6. C-MONOİDLER

Tanım 3.6.1. : Kartezyen monoid kısaca C- monoid üzerinde ekstra $(\pi, \pi', \varepsilon, *, \langle \ \rangle)$ yapıları tanımlı olan bir monoiddir. Hatırlanacağı üzere bu yapılar kategorinin Kartezyen olma özelliğinde tanımlanmıştır. Burada $\pi, \pi' \ ve \ \varepsilon$, M nin elemanları , $(-)^*$ bir tekli işlem ve $\langle -, - \rangle$ bir ikli işlem olmak üzere a,b,c,d ve e için

(1) $\pi\langle a, b\rangle = a$
(2) $\pi'\langle a, b\rangle = b$
(3) $\langle \pi c, \pi' c\rangle = c$
(4) $\varepsilon\langle d^*\pi, \pi'\rangle = d$
(5) $(\varepsilon\langle e\pi, \pi'\rangle)^* = e$

Dikkat edileceği üzere bu eşitlikler son nesne hariç Kartezyen kapalı kategori şartlarıdır. Yani C- monoid ler son nesnesi olmayan CCC olarak düşünülebilir. Bilindiği üzere bir monoid tek elemanlı bir küçük(small) kategoridir. Eğer terminal obje olsaydı bir obje ve bir morfizmden oluşan herhangi özelliği olmayan kategori elde edilir. Bu yüzden C-monoid de son nesne şartı aranmamaktadır. Doğal olarak C-monoid ler üzerinde de λ-kalkülüse benzer bir yapı tanımlanabilir. C-monoid bir nesneli olduğundan "type"

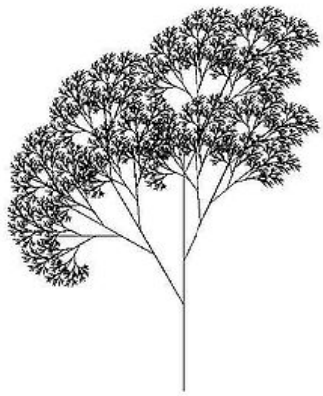
sınıfları yoktur. Dolayısıyla bu yapı daha geneldir. Ayrıntılı bilgi [category of software], [ ], [ ] kaynaklarında mevcuttur.

4. CCC LERDE RECURSİVE FONKSİYONLARI

4.1. RECURSİVE FONKSİYONLARI

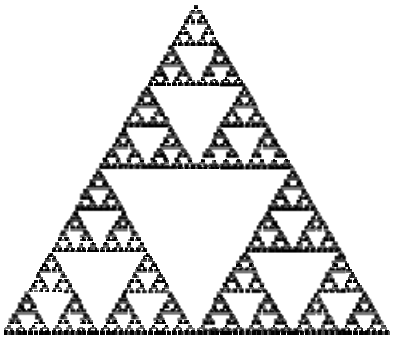
4.1.1. Recursive (Özyineleme):

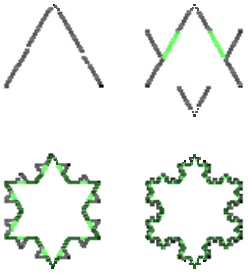
Basitce, kendi-kendini tekrarlamak olarak tarif edebileceğimiz recursive(özyineleme), aşağıdaki figürlerdede görüldüğü gibi, tabiatta doğal olarak farklı şekillerde karşımıza çıkar.Matematik biliminde Fraktallar(Şekil-4.4) olarak adlandırılan bu durum, aslında nesnenin yada şeklin bir alt birimine inildiğinde yine kendisiyle karşılaşılması durumudur.Örneğin; en küçük dalı bile, bitkinin tamamına benzeyen "bin perçem" otu (Şekil - 4.1), karşılıklı konan iki aynanın, birbirlerinin karşılıklı görüntülerini sonsuz kere göstermeleri(Şekil- 4.2) ve minik üçgenlerden oluşan üçgenlerin oluşturduğu büyük üçgen(Şekil- 4.3) .



Şekil - 4.1- *Bin perçem otu*

Şekil- 4.2-*Karşılıklı aynaların sonsuz görüntüleri*





Şekil-4.3–*Küçük üçgenlerin oluşturduğu büyük üçgen* Şekil- 4.4- *Fraktal oluşumu*

Biz bu çalışmada özyinelemenin(recursive) fonksiyon anlamı üzerinde duracağız ve recursive fonksiyonlar ile bunların bazı bilgisayar uygulamalarını inceleyeceğiz.

4.1.2. Genel Özyineli Fonksiyonların (General Recursive functions)kısa tarihi gelişimi

Özyineli(**recursion**) ve Hesaplanabilirlik(**computability**) konuları, **Gödel**, **Church** ve **Kleene** tarafından 1930 a ortaya çıkarılmıştır.Diğer matematikçiler, daha önceleri **Hilbert** tarafından ortaya konulan sorulara kısmen alaka gösteriyorlardı.Ondokuzuncu yüzyılın sonunda Hilbert, 1899 da, bu konuyla alakalı bir geometri alıştırması ortaya koydu ve bunu 1900 de tamamlayıp sundu.Bu alıştırma, Reel sayılar sistemi için, geometrideki indirgemeyi içeren bir soruydu ve **Dedekind** sonuçları tarafından aritmetiği yapıldı(en azından bir ikinci sıralı sistem olarak).1904 te Hilbert, 1928 deki kendisinin sonlu programına göre nasıl olduğu bilinen aritmetiğin içeriği olan, ispatı önerdi.Hilbert, bir düzen içinde kurulmuş olan sonlu matematiksel ispatları kullanarak, "bunun karşıtı olanlar elde edilemez" önerisini bildirdi.

"Entscheidungsproblem", Kleene durumuyla yakından alakalıdır.Bu kararlı problem ilk düzenli mantık ilmi için ilk olarak 1920 de Hilbert tarafından üniversitelerde verilen derslerde ortaya çıktı ve 1928 de Hilbert ve **Ackermann** tarafından tanımlandı.Bu bize bir kararlı yöntem verdi.(Entscheidungverfohren) "işlemlerin bir sınırlı sayıyısıyla verilen mantıksal ifadenin geçerliliğine bir karar vermeye izin verir". (Hilbert ve Ackermann, 1928). Hilbert bunu matematiksel mantığın esas problemi olarak gösterdi.

Gödel 1931 de (Rosser tarafından geliştirilerek modern terimler içinde yer alan) "Tamamlanmamış olan temel sayılar teorisinin herhangi T kararlı uzantısını kabaca iddia eden birinci tamamlanmamış teoremini" ispatladı.Aritmetize edilen Gödel ispatı onun ikinci tanımlanmamış teoremi olarak oluşturuldu ve bu, "onun kendi sabitiyle, böyle bir T ispatlanamaz" ifadesini iddia ediyordu ve bu Hilbertin programı için bir başarısızlıktı(aksamaydı).

Onun 1931 deki Gödel ispatı bir ilkel özyineli(**pirimitive recursive**) fonksiyonu kavramı olarak kullanıldı(ki o, onun tarafından "**recursive**" olarak adlandırıldı(eine recursive ffunction)). Çünkü bu fonksiyonlar, aritmetik için, Gödel in formal sistemi olan P içinde kolayca sembolize edilebilirdi. Ve bu fonksiyonlar ona bütün syntactic

nesneler için imkanlar sağladı. "Gödel sayıları"; sonuç olarak diyebiliriz ki o kendi referansını oluşturamadı ve bu konuyla alakalı olarak eksikti(yetersizdi).Bununla birlikte Gödel, ilkel özyineli(pirimitive recursive) fonksiyonlarının, bütün etkili hesaplanabilir fonksiyonlarını içermediğini ve 1934 o fonksiyonların geniş bir sınıfının Herbrond un önceki ortaya konulan fikrine bağlı olduğunu önerdiğinin farkına vardı. Gödel bu fonksiyonları "Genel özyineli**(general recursive functions**) fonksiyonlar " olarak isimlendirdi.Herbrand 7 nisan 1931 de Gödel e bir mektup yazdı. Mektupta, "eğer φ, bir bilinmeyen fonksiyonu gösterirse ve Ψ_1,Ψ_2 ,..... Ψ_k lar bilinmeyen fonksiyonlarsa, ve eğer Ψler ve φ en güzel büçimde biri, bir diğerinin yerine geçirilirse hesaplamaların sonuçlarının kesin çiftleri eşitlenir, daha sonra eğer fonksiyon denklemlerinin sonuçkümesi bire sahipse ve sadece φ için bir çözüm varsa φ fonksiyonu bir recursive fonksiyon dur" yazıyordu. Gödel bu tanımın üzerine onu daha etkili yapmak için iki koşul ekledi; birincisi, φ nin en dıştaki sembol olma şartıyla, fonksiyon denklemlerinin sol tarafı standart yapı içinde olmalı . Ve ikincisi;

n_1, n_2 ,.....n_j kadar olan doğal sayıların her bir kümesi için bir m sırası çıkar.Örnek olarak ; $\varphi(n_1,....n_j) = m$ bir hesaplanmış denklemdir. Kleene 1936, 1943 ve 1952 de Gödelin kuralının seşeneklerini tanıttı ve bu Herbrand-Gödel tanımının bir eşitlik formülünü ortaya koyar.

4.1.3. Özyineli fonksiyonlar(Recursive Functions)

Özyineli fonksiyon terimi(Recursive functions); genelde özyineli ile tanımlanan fonksiyonlar için kullanılır.Hesaplanabilir(computability) fonksiyon anlamındada kullanılır.

Klein, 1952 de , f herhangi bir fonksiyon olmak üzere, pozitif tamsayılar(nonnegative) ın bir ilkel özyineli fonksiyonunu tanımladı.Klein bu tanımlamayı yaparken, Denklemlerin eşitiğini(noncontradictory), bileşke işlemini, (1)f,g,h gibi fonksiyon sembollerini, (2)pozitif tamsayılar için çeşitli değişkenler(ör:x,y,z vb.), (3)sıfır sabitini ve (4) $S(x)=x+1$ ardıl(successor) fonksiyonundan faydalandı.

Örneğin:

x ve y nin çarpımını hesaplayan, x ve y ye bağlı, $f(x,y)$ fonksiyonu;

$$f(x,0) = 0 \qquad (1)$$

$$f(x,S(y)) = g(f(x,y),x) \qquad (2)$$

$$g(x,0) = x \qquad (3)$$

$$g(x,S(y)) = S(g(x,y)) \qquad (4)$$

Şunu hemen belirtelim ki; denklem verilen her pozitif değeri için, *f* in değerlerini yegane(uniquely) olarak saptayamayabilir.Bu durumda tanım kısmidir(partial). Eğer girilen her değer için, *f* in değeerlerini hesaplarsa, işte o zaman "tanım tamdır"(total) deriz. Genelde Özyineli fonksiyon(recursive function) terimi tek başına kullanıldığında, tam özyineli fonksiyon(total recursive function) tanımı kastedilir, anlaşılır.Bazı yazarlar, genel özyineli fonksiyon"(general recursive function) ifadesini, kısmi özyineli fonksiyonlar(partial recursive function) yerine kullanırken, diğer bir kısım yazarda "tam özyineli fonksiyon"(total recursive function) yerine kullanmaktadırlar.

Özyineli fonksiyonlar üzerine çalışan bilim insanlarından bir diğeride Rosa Peter dir.Rosa peter Gödelin Çalışmalarını inceledikten sonra matematikte "***özyineli fonksiyonlar teorisi***"(Recursive function theory) olarak bilinen alanı kurdu ve 1950 lerin ortasından itibaren bu teoriyi bilgisayar teorisine uygulamaya girişti.1976 yılında yayınlanmış son eseri "Recursive Functions in Computer Theory" isimli kitabı bu alandaki en önemli eserlerden birisi olarak kabul edilmektedir.

4.1.4. İlkel özyineli fonksiyonlar(Pirimitive Recursive Functions)

İlkel özyineli fonksiyonlar, ilkel özyineleme ve bileşke işlemi ana işlem olarak alınarak tanımlandı. İlkel özyineli fonksiyonlar, Özyineli fonksiyonların bir kesin(strict) altkümesidir.(Özyineli fonksiyonlar, aynı zamanda hesaplanabilir fonksiyon olarakda bilinir.)Bu terim Rosa Mayer tarafından türetildi.

Hesaplanabilirlik teorisinde, İlkel özyineli fonksiyonlar, fonksiyonların bir sınıfıdır.Bu fonksiyonlar aynı zamanda ispat teoridede çok önemlidir.

Sayılar teorisinde çalışılan normal fonksiyonların bir çoğu ilkel özyinelidir.Örneğin: Toplama, Bölme, Faktöriyel, Üstel ve n.nci asal fonksiyonları, ilkel özyinelidir.Bu sebeple bunlar gerçek değer fonksiyonlarına yaklaşımlardır.(Brainerd ve Landweber, 1974). Aslında, bazıları bilinmesine rağmen, ilkel özyinelemeli olmayan fonksiyonların geliştirilmesi gerçekten zordur.İlkel özyineli fonksiyonların kümesi, Complexity teoride PR olarak bilinir.

Her ilkel özyineli fonksiyon, bir genel özyineli fonksiyondur.

Tanım 4.1.4.1. :

Tüm fonksiyonların sınıfı, temel operatorlerin sonlu sayıdaki uygulamları kullanılarak tanımlanabilir, bir temel fonksiyonla, "İlkel tekrarlı fonksiyonların sınıfı" ile başlıyoruz.Bu Fonksiyonlar sınıfı üç önemli özelliğe sahiptir:

1- Tüm ilkel tekrarlı fonksiyonlar tamdır.
2- Tüm ilkel tekrarlı fonksiyonlar etkin bir biçimde hesaplanabilir.
3- Most common tam fonksiyonlar ilkel tekrarlıdır.

İlk iki özellik, oldukça açıktır, aşikardır: her bir fonksiyon tamdır ve çok basit hesaplanabilirdir ve argümentleride tam ise, operator sonuçlarının her ikiside tam fonksiyonların içindedir.Ayrıca, operatörlerin etkileri dahi çok basitçe hesaplanabilirdir.

Üçüncü özellik, ne kesinkes belirtilmiş ne de açıkca tamdır.

İlkel özyineli fonksiyonlar, Number teoretic fonksiyonlar arasında, doğal sayılardan doğal sayılara ($NxNx \dots xN \to N$)bir fonksiyondur.Bu fonksiyonlar bazı n doğal sayıları için, n argumant alırlar ve bunlara n-ary denir.

İlkel özyineli fonksiyonlar, aşağıdaki aksiomlarla birlikte verilir;

1. Sıfır Sabit fonksiyon: $zero(x)\ def\ =\ 0$, 0-ary sıfır sabit fonksiyonu, ilkel özyinelidir.
2. Ardıl fonksiyon: $succ(x)\ def\ =\ x\ +\ 1$, 1-ary, ardıl fonksiyonu, kendi argümantının ardılına döner ve ilkel özyinelidir.
3. İzdüşüm fonksiyonu: Her $n \geq 1$ ve $i, (1 \leq k \leq n)$ için,

$proj_k^n\ (x_1, x_2, ..., x_n)def = x_k$,

n-ary izdüşüm fonksiyonu P_k^n, k. cı argümantına döner ve ilkel özyinelidir.

Aşağıdaki aksiyomlarla verilen işlemleri uygulayarak daha komplex ilkel özyineli fonksiyonlar elde edilebilir;

1. **Bileşke:** Bir f , n-ary ilkel özyineli fonksiyonu ve $g_1,...,gn$, n, m-ary ilkel özyineli fonksiyonları verilsin. f ve $g_1,...,gn$ fonksiyonlarının bileşkesi, m-ary

$$h(x_1,x_2,....,x_m) = f(g_1(x_1,x_2,....,x_m), ..., g_k(x_1,x_2,....,x_m))$$

bir ilkel özyinelidir.

2. **İlkel özyineleme:** Bir f , k-ary ilkel özyineli fonksiyonu ve bir g, (k+2)-ary ilkel özyineli fonksiyonları verilsin.

$$h(0,x_1,...,x_k) = f(x_1,....,x_k) \quad \text{ve}$$
$$h(S(y),x_1,....,x_k) = g(y,h(y,x_{1,...,}x_k),x_{1,....,}x_k).$$

olarak alınırsa bir h, (k+1)-ary fonksiyonu, f ve g nin ilkel özyinelemesi gibi tanımlanabilir.

4.1.4.2.İzdüşüm fonksiyonların rolü :

İzdüşüm(projection) fonksiyonları, yukardaki fonksiyonların tekrar bölümlerindeki aşikar katılıktan kurtulmak için kullanılabilir.

Değişik izdüşüm fonksiyonlarıyla birlikte bileşke işlemi kullanılarak, bir fonksiyondan, başka bir fonksiyonun argümentlerinin altkümesine geçilebilir. Örneğin; eğer g *ve* h, *2*-ary ilkel özyineli fonksiyon ise,

$$f(a,b,c) = g(h(c,a),h(a,b))$$

Aynı zamanda ilkel özyinelidir.Bir şekli tanım kullanılarak izdüşüm fonksiyonları

$$f(a,b,c) = g(h(P_3^3(a,b,c),P_1^3(a,b,c),h(P_1^3(a,b,c),P_2^3(a,b,c))))$$

yazılabilir.

4.1.4.3.İfadeyi Nümeric Fonksiyona Dönüştürmek

Bazı ortamlarda, ilkel özyineli fonksiyonları hesaba katmak doğaldır. O, ya gerçekçi değerler { t= true, f=false } ile birlikte karışık rakamlar alır, yada gerçek değer sonuçları, bir çıktı gibidir.(Kleene [1952 pp.226-227]). Bu, herhangi bir tarzın numaralarıyla birlikte gerçekçi değeri tanımlayarak yerini alabilir. Örneğin; 1 ile t ve 0 ile f, gerçekçi değerinin belirlenmesi bilinendir. Önceleri bu özdeşleşme yapılmışdı.Bu, 0 ile 1 arasında gidip gelen *A* kümesinin karakteristik fonksiyonudur, ki o bize, A da bir sayı olup olmadığını söyler.

4.1.4.4.Bilgisayar Dili Tanımı:

İlkel özyineli programlama dilinin bir örneği, temel aritmetic operatorleri içerir; (e.g. + and -, or ADD and SUBTRACT), koşullu ve mukayeseli(IF-THEN, EQUALS, LESS-THAN), sınırlı döngüler.Ki orada, o bilinen yada tüm döngülere kadar hesaplanabilir üst sınırdır, ve while döngüsü gibi yada İF-THEN e ilaveten GOTO gibi daha genel kontrol yapıları yoktur. Eklenen Sınırsız döngüler (WHILE, GOTO), parçalı özyineli dil veya tam-turing yapar, hemen hemen tüm gerçek bilgisayar dilleri gibidir.
Keyfi bilgisayar programları, yada Turing makinaları, genel analiz de duraksayan(halting) problem olup olmadığının farkına varamazlar.Aslında İlkel özyineli fonksiyonlar çok iyi analiz edebilirler ve Kontrolle bir tek şey söyleyebilirler: programın gösterim süresi nedir? Bu bir çelişki değildir; ilkel özyineli programlar, tüm mümkün programların bir isteğe bağlı olmayan(non-arbitrary) altkümesidir, özel olarak inşa edilmiş, analiz edilebilirdir.

4.1.4.5.Örnekler:

Bir çok sayı-teorisi fonksiyonları,bir tek değişkenli üzerinde özyineleme uygulayarak tanımlanabilir ve ilkel özyinelidir.Temel örnekler, toplama ve sınırlı çıkarma fonksiyonları içerir.

4.1.4.6.ÖNCEL(Predecessor):

PRED olarak tanımlanan "Öncel fonksiyon", hemen hemen "ardıl(successor) fonksiyonun" tersidir(inverse), burada ki tek istisna sıfırdır ki öncelin tanımı zero olarak yapıldı.

$$PRED(x) = \begin{cases} x - 1, & x > 0 \\ 0, & x = 0 \end{cases}$$

"Öncel fonksiyon", ilkel özyinelinin terimlerinde kolayca tanımlandı.Bunu görmek için iki durumu inceleyerek başlayalım;

$$PRED(0) = 0$$

$$PRED(m+1) = m$$

Bununla fonksiyonu tamamen tanımlamış olduk. Daha sonra bu durumların her birini, uygun arity nin bir fonksiyonu gibi ifade ederiz. Zira *PRED* bir "birli(unary) fonksiyon"dur.Bizim sırasıyla *f* ve *g*, boşlu(null-ary) ve ikili(binary) fonksiyonlara ihtiyacımız var.Özellikle, *f* ve *g* nin, ilkel tekrarlının(pirimitive recursive) tanımına uygun olarak, aşağıdaki gibi olmasını isteriz;

$$PRED(0) = f()$$

$$PRED(m+1) = g(m, PRED(m)),$$

Bu durumda, her iki fonksiyon basit izdüşümlerine uygundur,

$$f = PROJ_0^0 \quad \text{ve} \quad g = PROJ_0^0.$$

Böylece;

$$\text{PRED} = \text{PREC}(\text{PROJ}_0^0, \text{PROJ}_0^0)$$

Şunu belirtelim ki; burada, bu final ifadede doğal sayı argümentleri yoktur.(() PRED fonksiyon,işlemler, parantezler ve temel fonksiyonların terimlerinde tamamen ifade edildi.Sembollerle gösterildiği sürece doğal sayılar, verilen fonksiyonların nasıl kurulduğu hakkında çalışırken gayet kullanışlıdır.En son final tanımında asla gözükmezler.Bu her zamanki kurallarla tutarlıdır, *f*, tek başına bir fonksiyon sembolüdür ve x de değerlendirildiğinde tek başına bir fonksiyon belirtir.

4.1.4.7.Toplama:

Benzer şekilde, bir ikili(binary) fonksiyon kullanarak toplama işlemi tanımlanabilir. Matematiksel olarak ,

$ADD(x,y) = x+y$

ADD toplam fonksiyonunu tanımlamayı ümit ederiz.

Tekrar bir kere daha, Toplam fonksiyonunu tam olarak tanımlayabilmek için öncelikle iki durumu tekrar gözden geçirelim;

$$ADD(0,y) = y$$

$$ADD(m+1,y) = ADD(m,y)+1.$$

Şunu belirtelim ki; ikinci durum, birinci argumentin azaltılmasıyla birlikte kendisine ilişkin bir özyineleme kullanır,ki o ilkel özyinelemeden bağımsız alınan bir şeydir.Zira ardıl fonksiyon bir tek artabilir, aşağıdaki gibi, sırasıyla birli(unary) ve üçlü(ternary), f ve g ilkel özyineli fonksiyonlarını kolayca bulabiliriz;

$ADD(0,y) = f(y)$ ($\text{add}(0,y) = P_1^1(y)$)

$ADD(m+1,y) = g(m, ADD(m,y),y)$ ($\text{add}(S(n),x) = S(P_1^3(\text{add}(n,x),n,x)$)

Burada P_1^3 İzdüşüm fonksiyondur ve 3 argumant alır ve birinciye döner. P_1^1 açıkça birim fonksiyondur; kalıntısı yukardaki ilkel özyineleme operatörünün tanımıyla gereklidir ve f in rolünü oynar. S ve P_1^3 Ilkel özyinelilerin bileşkesi, g nin rolünü oynar.

Özellikle, $f = PROJ_1^1$ ve $g = COMP(SUCC , PROJ_2^3)$. Yani, g fonksiyonu, ikinci argumantına ardıl fonksiyonu kolayca uygular, diğer argumantlarıda ihmal eder.Böylelikle

$$ADD = PREC(PROJ_1^1 , COMP (SUCC , PROJ_2^3)).$$

Sonucunu çıkarırız.

4.1.4.8.Çıkarma:

Bir sınırlı çıkarma fonksiyonu bu bağlamda iyice düşünüldü.Çünkü ilkel özyineli fonksiyonlar tam sayılardan ziyade doğal sayıları kullanır ve doğal sayılar çıkarma işlemine göre kapalı değildir Bu sınırlı çıkarma işlemi fonksiyonu sub(a,b), $b - a$ ya geri döner, eğer bu negatif değil ve sıfıra dönerse.

Öncel fonksiyon, ardıl fonksiyonun zıttı gibi rol yapar ve aşağıdaki kurallarla özyineli olarak tanımlanır; (pred = öncel)

pred(0)=0,

pred(n+1) = n.

Bu kurallar, ilkel özyineleme ile daha düzenli tanım haline getirilebilir,

pred(0) = 0,

pred(S(n)) =P_2^2(pred(n),n).

Sınırlı çıkarma fonksiyonu, Toplama yoluna benzeşen tarzda öncel fonksiyondan tanımlanabilir.Toplama yolu ardıldan tanımlandı.

sub(0,x) = P_1^1(x),

sub(S(n),x) = pred(P_1^3(sub(n,x),n,x)).

Burada sub(a,b), b-a `a benzerdir, sadelik yararına, argümentlerin dizisi, ilkel özyinelinin tanıma en uygun şartlarından bir anahtara sahiptir.Bu, uygun izdüşüm ile birlikte bileşkeyi kullanarak kolayca düzeltebilir.

Diğer ilkel özyineli fonksiyonlar, üst alma ve primality("heryere tıklayabilirsin, ancak, sadece buraya tıklama") testi içerir. Verilen e,f,g,h ilkel özyineli fonksiyonlar $e \leq f$ olduğunda g nin değerine geri döner, aksi takdirde h ın değeri özyinelidir.

4.1.4.9.Çarpma:

Çarpmanın tanımı, tıpkı toplamaya benzer.

$$MULT(0,y) = 0$$

$$MULT(m+1,y) = MULT(m,y) + y$$

Olacak şekilde bir

$$MULT(x,y) = x*y$$

Çarpma(MULT) fonksiyonunu tanımlarız,

Biz şimdi aşağıdaki gibi, bir birli(unary) *f* fonksiyonu ve üçlü(ternary) *g* fonksiyonu tanımlayalım;

$$MULT(0,y) = f(y)$$

$$MULT(m+1,y) = g(m , MULT(m,y), y)$$

Sırasıyla $f = ZERO$ ve $g = COMP (ADD,PROJ_2^3,PROJ_3^3)$ alırsak, sonuç olarak

$$MULT = PREC(ZERO , COMP(ADD , PROJ_2^3 , PROJ_3^3))$$

elde ederiz.

4.1.4.10. Faktöriyel:

$$FACT(0) = 1$$

$$FACT(m+1) = g(m,FACT(m))$$

Olmak üzere ;

$$FACT(x) = x!$$

Faktöriyel fonksiyonunu tanımlayalım.Şimdi sırasıyla, birli(unary) *f* ve ikili(binary) *g* fonksiyonlarını aşağıdaki gibi tanımlayalım;

$$FACT(0) = f()$$

$$FACT(m+1) = g(m,FACT(m)).$$

Bayağı *f* fonksiyonu, $COMP(SUCC,PROJ_0^0)$ üzerinden, izdüşüm ve ardıl fonksiyonları tarafından kurulur. Malesef, *g* fonksiyonu *m* yi atlar ve $m+1$ `i ister.Bu da ilaveten, bir ardıl fonksiyonla bir bileşim gerektirir. Böylece; *g* fonksiyonu, $COMP(SUCC,PROJ_1^2)$ ve $PROJ_2^2$ nin çarpımından hesaplanmalıdır. Böylelikle,

$$FACT=PREC(COMP(SUCC,PROJ_0^0),COMP(MULT,COMP(SUCC,PROJ_1^2),PROJ_2^2))$$

Sonucunu elde ederiz.

4.1.4.11.Diğer örnekler:

İlkel özyineli fonksiyonların tanımlanmasında ki gibi ipuçlarıyla, bu fonksiyonlar hakkında bir çok örnek verebiliriz. Ancak burada tamamen düzenli yapılardan ziyade birkaç konvansiyon(convention) kullanacağız. Örneğin; Bir fonksiyon başka bir fonksiyon içine basitce konularak *COMP* operatorü elendi(eliminated). Bu tanımların, şekli eşitliklere nasıl dönüştürüldüğünün açık ve net olması gerekir.

$$ABSDIFF(x,y) = ADD(PSUB(x,y),PSUB(y,x))$$

$$ZERO?(x) = PREC(SUCC(PROJ_0^0),ZERO(PROJ_1^2))$$

$$EQUAL?(x,y) = ZERO?(ABSDIFF(x,y))$$

$$NOT(x) = PREC(PROJ_0^0 , SUCC(ZERO(PROJ_1^2)))$$

$$BRANCH(p,x,y) = PREC(PROJ_2^2,PROJ_3^4)$$

Biz burada "?"ile birlikte işleminin kurallarını kullandık.ABSDIFF fonksiyonu, argumentlerinin mutlak farkına döner, $|x - y|$. BRANCH fonksiyonu, eğer birinci doğru(non-zero) ise ve üçüncü yoksa ikinci argümente döner. Şunu belirtelim ki; bir bayağı yolda, *ZERO?, NOT* ve *BRANCH* fonksiyonlarının hepsi ilkel özyinelemeyi(pirimitive recursive) kullanır. Yani ilkel öz yineleme basitce sıfır olan yada olmayan ilk argümentleri birbirinden ayırır ve özyinelemeyi bütünüyle yok sayar.

4.1.4.12.Tamsayılar ve Rasyonel sayılar üzerinde işlemler:

Gödel numberings kullanılarak, ilkel özyineli fonksiyonlar, tamsayılar ve rasyonel sayılar gibi diğer objeler de işlemeye genişletilebilir.Eğer tamsayılar, standard yolda Gödel numaralarıyla kodlanmışsa, aritmetik işlemler toplamayı çıkarmayı ve çarpmayı içerir ve hepside ilkel özyinelidir.Benzer şekilde, eğer rasyoneller, Gödel sayılarıyla temsil edilmişse, alan işlemlerinin tamamı ilkel özyinelidir.

4.1.4.13.Özyineli fonksiyonlara bağlantı(İlişki)

Parçalı özyineli fonksiyonların daha geniş bir sınıfı, sınırsız arama işlemleriyle ortaya konularak tanımlandı.Bu operatörün kullanımı, parçalı fonksiyonlarda sonuçlanabilir, şöyleki; bir bağlantıyla, en çok, her argument için bir değer, fakat her argument için herhangi değerler alır. Bir eşdeğer tanım şöyle yapılabilir, bir parçalı özyineli fonksiyon tektir ve turing makinalarıyla hesaplanabilir.

Bir "tam özyineli fonksiyon", bir "parçalı özyineli fonksiyondur" ve her girdi için tanımlıdır.

Her ilkel özyineli fonksiyon, tam özyinelidir fakat tersi doğru değildir.

$A(m,n)$ Ackerman fonksiyonu, tam özyineli fonksiyonların iyi bilinen örneğidir ancak ilkel özyineli değildir.Ackerman fonksiyonlarını kullanarak, tam özyineli fonksiyonların bir alt kümesi gibi, ilkel özyineli fonksiyonların bir karakterizasyonu vardır.Bu karakterizasyon durumu şu dur:

Bir fonksiyon ilkel özyinelidir, ancak ve ancak, n , ilkel özyineli fonksiyonların argümentlerinin toplamı olmak üzere, daha az adımla veya A(m,n) Turing makinaları ile hesaplanabilir m doğal sayısı vardır.

İlkel özyineli fonksiyonların önemli bir özelliği, tüm "tam özyineli" fonksiyonların kümesinin "enumerabily özyineli" alt kümesi olmasıdır. Bunun anlamı şudur: orada $f(e,n)$ tek hesaplanabilir fonksiyonu vardır, şöyle ki;

- Her ilkel özyineli g fonksiyon için, her n için, $g(n) = f(e,n)$ olacakşekilde bir e vardır. Ve
- Her e için $h(n)$ fonksiyonu, $\lambda n\, f(e,n)$ ilkel özyinelisi gibi tanımlandı.

Aynı zamanda, ilkel özyineli fonksiyonlar, Tam hesaplanabilir fonksiyonların en geniş özyinelemeli enumerable kümesi değildir.

4.1.4.14.Sınırlamalar:

İlkel özyineli fonksiyonlar, "hangi fonksiyonlar hesaplanabilirler" ile alakalı sezgilerimizle yakından alakalıdır.Kesinlikle, Başlangıç fonksiyonu, sezgiyle ve yeni ilkel özyineli fonksiyonlar oluşturabilecek iki işlemle hesaplanabilir.Bununla birlikte, ilkel özyineli fonksiyonların kümesi, her hesaplanabilir fonksiyonu içermez.Bu cantorun köşegen argümantlarının değişik biçimiyle birlikte görülebilir. Bu argument,

ilkel özyineli olmayan bir hesaplanabilir fonksiyon gösterir.İspat aşağıdaki şekildedir: İlkel özyineli fonksiyonlar, hesaplanabilir numaralı olabilir.Bu numaralama, her ne kadar asli fonksiyonlarda tek olmasada, fonksiyonların tanımı üzerinde yeganedir. Turing makinaları yad "μ-özyineli fonksiyonlar" gibi hesaplanabilirliğin formal modeli altında tanımlanabilen numaralama, bu anlamda hesaplanabilirdir.Ancak bir church-Turing tezine başvurmak muhtemelen yeterli olacaktır.Satırlar, bu numaralama altında, tek argümantın ilkel özyineli fonksiyonu olduğunda, bir matrix hesaba katmalıyız.Burada kolonlar doğalsayıdır.Herbir (i, j) , j sayısı üzerinden hesaplanmaya başlanan$i$.ci birli ilkel özyineli fonksiyona uyuşur. Bunu $f_i(j)$ gibi yazabiliriz. Şimdi $g(x) = S(f_x(x))$ i hesaba katalım. g bu matrisin köşegeninde durur ve basitce bulunan değere kolayca eklenir.Bu fonksiyon hesaplanabilirdir fakat ilkel özyineli fonksiyon açıklıkla çıkmaz, mümkün olan ilkel özyineli fonksiyonların en son ki değerinden farklılık gösterir.Bu hesaplanabilir fonksiyon olmalıdır ama ilkel özyineli değildir.

Bu argument, hesaplanabilir fonksiyonların herhangi birine uygulanabilir, bu yolla enumerated yapılabilir, "Machines that always halt" yazısında açıklandığı gibi. Belirtelimki aynı zamanda parçalı özyineli fonksiyonlar açıkça sayılabilir.Örneğin, enumerating Turing makinası şifresiyle.

İlkel özyineli olmayan tam özyinelilerin diğer örnekleri bilinen örneklerdir. Şöyle ki:

- Ackerman(m,m) fonksiyonu, birli tam özyineli fonksiyondur ama ilkel özyineli değildir.
- Paris-Harrington teoremi, ilkel özyineli olmayan, bir tam özyineli fonksiyon gerektirir.Çünkü bu fonksiyon Ramsey teori ile harekete geçirilmiştir, bazen ackermana göre daha "doğal" olarak kabul eder.
- "Sudan fonksiyonu".

4.1.4.15.**μ-özyineli fonksiyonlar ve λ-özyineli fonksiyonlar**

Matematiksel mantık ve bilgisayar bilimlerinde, "μ-özyineli fonksiyonlar", doğal sayılardan doğal sayılara, bir sezgisel algıyla, hesaplanabilir, "parçalı(partial) fonksiyonların" bir sınıfıdır. Aslında, μ-özyineli fonksiyonlar ,computibility(hesaplanabilirlik) teoride , Turing makinaları ile hesaplanabilir fonksiyonlar olarak gösterilmektedir. "μ-özyineli fonksiyonlar", "ilkel özyineli fonksiyonlar"(primitive recursive functions) ile yakından ilgilidir ve tümevarımsal

tanımları, μ-özyineli fonksiyonlar ile kurulmuştur. Bununla beraber her μ-özyineli fonksiyon, bir ilkel özyineli değildir, en meşhur örneği "Ackermann fonksiyonu"dur.

Parçalı fonksiyonların diğer bir sınıfıda, λ-özyineli fonksiyonlardır ve "Markov algoritması" ile hesaplanabilirdir.

Tüm özyineli fonksiyonların kümesi, hesaplama karmaşıklığı teorisinde, R gibi bilinir.

Tanım 4.1.4.15.1 :

"μ-özyineli fonksiyon"(veya parçalı μ-özyineli fonksiyonlar), doğal sayıların sonlu tuplesini alan ve tek bir sayıya dönen parçalı fonksiyonlardır. Bu fonksiyonlar başlangıç fonksiyonlarını içeren Parçalı fonksiyonların en küçük sınıfıdır ve birleşme işlemine göre kapalıdır.

Başlangıç fonksiyonlarını içeren, bileşkeye göre kapalı olan, ilkel özyineli fonksiyonların en küçük sınıfına, "ilkel özyineli fonksiyonların sınıfı" denir. Tüm ilkel özyineli fonksiyonlar tam(total) olduğunda, bu durum "parçalı özyineli fonksiyon"lar için doğru değildir. Örneğin: "Ardıl fonksiyonun" minimizasyonu, tanımlanmamış. Tam özyineli fonksiyonların kümesi, parçalı özyineli fonksiyonların bir alt kümesidir ve ilkel özyineli fonksiyonların bir üst kümesidir, öyleki Ackerman fonksiyonlarına benzeyen bu fonksiyonlar, tam özyineli olarak ispatlanır, ilkel olarak değil.

Aşağıdaki İlk üç fonksiyona "temel" veya "başlangıç" fonksiyonları denir(Klene,1952)

Sabit fonksiyon:
Her bir n ve k değeri için;

$$f(x_1,x_2,\ldots,x_k) = n$$

Alternatif tanımlar, sabit fonksiyon yerine, ardıl fonksiyonların bileşkesini ve sıfır fonksiyonunu kullanırlar ve daima sıfıra geri döner.

Ardıl fonksiyon S:

$$S(x)=_{def} f(x)=x+1$$

İzdüşüm fonksiyonu P_i^k (aynı zamanda I_i^k birim fonksiyonuda denir):

Tüm doğal sayıları için

$i,k,\ 1 \leq i \leq k$ olacak şekilde $P_i^k =_{def} f(x_1,x_2,\ldots,x_k) = x_i$ dır.

Bileşke fonksiyonu(aynı zamanda yerine koyma operatorude denir):

Bir m-ary, $h(x_1,\ldots,x_m)$ fonksiyonu ve

k-ary, $g_1(x_1,\ldots,x_k),\ldots,g_m(x_1,\ldots,x_k)$ fonksiyonları verilsin. Öyleyse

$$h \circ (g_1,\ldots,g_m) =_{def} f(x_1,\ldots,x_k) = h(g_1(x_1,\ldots,x_k),\ldots,g_m(x_1,\ldots,x_k))$$

dır.

İlkel özyineli fonksiyon ρ:

k-ary $g(x_1,\ldots,x_k)$ fonksiyonu ve

k+2-ary $h(y,z,x_1,\ldots,x_k)$ fonksiyonu verilsin, öyleyse

$f(0,x_1,\ldots,x_k) = g(x_1,\ldots,x_k)$ olduğunda $\rho(g,h) =_{def} f(y,x_1,\ldots,x_k)$ dır.

Küçültme operatörü μ:

(k+1)-ary parçalı fonksiyonu $f(y,x_1,\ldots,x_k)$ olsun

$\mu(f)(x_1,\ldots,x_k) = z \Leftrightarrow_{\text{def}} \exists\, y_0,\ldots,y_z$ dır. Şöyleki;

$y_i = f(i,x_1,\ldots,x_k)$ $(i=0,\ldots\ldots,z)$

$y_i \neq 0$ $(i=0,\ldots,z-1)$

$y_z = 0$

Sezgiyle, Şayet fonksiyon tüm daha küçük argümentler için tanımlanırsa, küçültme fonksiyonunun nedenleri sıfıra geri döner(return) ve en küçük argumantı bulur.

4.2. BİLGİSAYAR İÇİN ÖZYİNELEME (RECURSION)

4.2.1. Giriş

Kendini doğrudan veya dolaylı olarak çağıran fonksiyonlara özyineli (recursive) fonksiyonlar adı verilir. Özyineleme (recursion), iterasyonun (döngüler, tekrar) yerine geçebilecek çok güçlü bir programlama tekniğidir. Orijinal problemin küçük parçalarını çözmek için, bir alt programın kendi kendini çağırmasını sağlayarak, tekrarlı işlemlerin çözümüne farklı bir bakış açısı getirir. Yeni başlayan programcılara, bir fonksiyon içinde atama değiminin sağında fonksiyon isminin kullanılmaması gerektiği söylenmekle birlikte, özyineli programlamada fonksiyon ismi doğrudan veya dolaylı olarak fonksiyon içinde kullanılır.

4.2.2. Faktöryel Fonksiyonu

Faktöryel fonksiyonunun matematik ve istatistik alanında önemi büyüktür. Verilen pozitif bir n tamsayısı için n faktöryel, n! Şeklinde gösterilir ve n ile 1 arasındaki tüm tamsayıların çarpımına eşittir.

Örnekler :

0! = 1

1! = 1

5! = 5 * 4 * 3 * 2 * 1 = 120

Faktöryel fonksiyonunun tanımı (definition of factorial function) :

n! = 1 if n==0

n! = n * (n-1) * (n-2) * ... * 1 if n>0

n tamsayısını alıp n faktöryelin değerini döndüren bir algoritmayı şu şekilde oluşturabiliriz :

```
prod = 1;
for(x=n; x>0; x--)
  prod *= x;
return prod;
```

Böyle bir algoritma tekrarlı (iterative) bir algoritmadır. Çünkü, belirli bir şart gerçekleşinceye kadar süren belirgin bir döngü veya tekrar vardır.

Faktöryel fonksiyonunun diğer bir tanımı :

n! = 1 if n==0

n! = n * (n-1)! if n>0

Bu, faktöryel fonksiyonunun kendi terimleri cinsinden tanımlanmasıdır. Böyle bir tanımlama, bir nesneyi kendi cinsinden daha basit olarak ifade etmesinden dolayı **özyineli(recursive)** tanımlama olarak adlandırılır.

Örnek :

$b_n = nb_{n-1}$

$b_0 = 1$ olmak üzere;

$b_4=4*b_3$

$= 4 * 3 * b_2$

$= 4 * 3 * 2 * b_1$

$= 4 * 3 * 2 * 1 * b_0$

$= 4 * 3 * 2 * 1 * 1$

$= 4 * 3 * 2 * 1$

$= 4 * 3 * 2$

$= 4 * 6$

$= 24$

Veya;

3! = 3 * 2! = 3 * 2 * 1! = 3 * 2 * 1 * 0!

= 3 * 2 * 1 * 1

= 3 * 2 * 1

= 3 * 2

= 6

/* n! değerini hesaplayan C fonksiyonu */

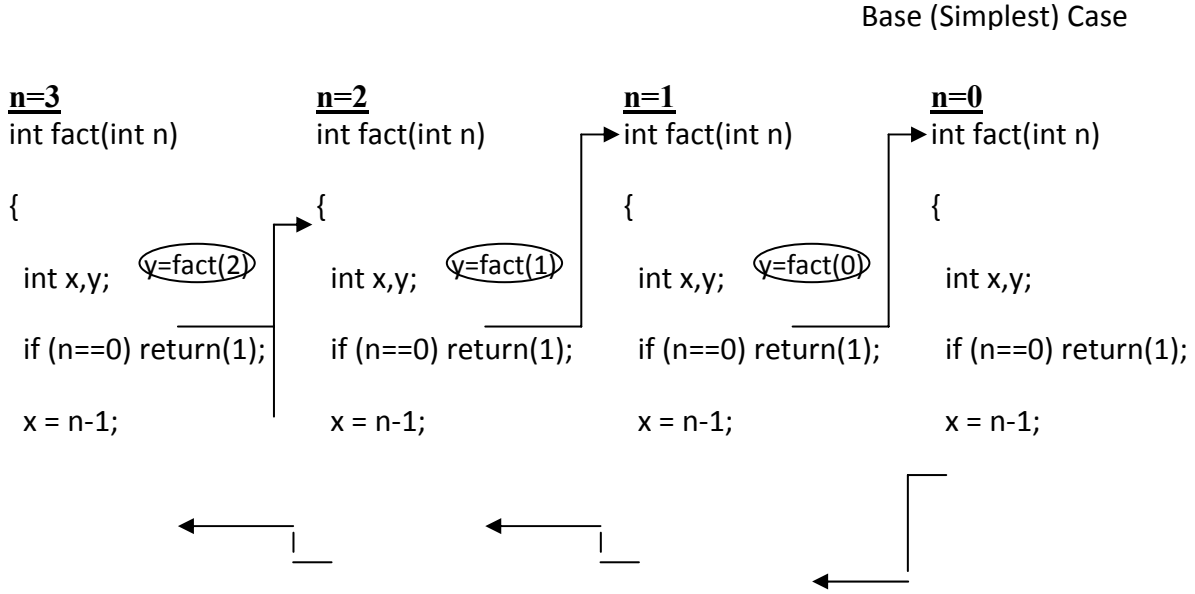
Recursive

```
int fact(int n)
{
  int x,y;
  if (n==0) return(1);
  x = n-1;
  y = fact(x); /*Kendini çağırıyor*/
  return (n*y);
}
```

Iterative

```
int fact(int n)
{
  int x, prod;
  prod = 1;
  for(x=n; x>0; x--)
    prod *= x;
  return (prod);
}
```

Bu fonksiyonlar diğer bir fonksiyondan "printf("%d", fact(3));" şeklinde çağrılabilir.



n! fonksiyonunun iyileştirilmesi:

Özyineli fonksiyonun her çağrılışında bellekte x ve y degiskenleri için yer ayrılması istenmiyorsa fonksiyon kısaltılabilir :

```
int fact(int n)
{
return ( (n==0) ? 1 : n * fact(n-1) );
}
```

4.2.3. Fibonacci Dizisi (Fibonacci Sequence)

Bilgisayar bilimlerinde çok sık kullanılan sayı serileridir. Bu sayıların önemi **özyineli (recursive)** fonksiyonlar ile kolayca yazılabilmesidir.
Fibonacci serisinin ilk iki sayısı 1′dir. Diğer sayılar ise kendinden önceki iki sayının toplamıdır.

0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,……

1. eleman : 0
2. eleman : 1
3. eleman : 0+1 = 1
4. eleman : 1+1 = 2
5. eleman : 1+2 = 3
6. eleman : 2+3 = 5

.....

Fibonacci dizisinin tanımı :

$$fib(n) = \begin{cases} 0 & if\ n = 0 \\ 1 & if\ n = 1 \\ fib(n-1) + fib(n-2) & if\ n \geq 2) \end{cases}$$

Veya;

fib(n) = n if n==0 or n==1

fib(n) = fib(n-2) + fib(n-1) if(n>=2)

Bir örnek hesaplama;

$b_n = b_{n-1} + b_{n-2}$

$b_1 = 1, b_0 = 0$

alınırsa, n=4 için örnek hesaplama aşağıdaki gibi olur;

$b_4 = b_3 + b_2$

$= b_2 + b_1 + b_1 + b_0$

$= b_1 + b_0 + 1 + 1 + 0$

$= 1 + 0 + 1 + 1 + 0$

$= 3$

Örnek hesaplama :

fib0=1

fib1=1

fib2=fib0+fib1=1+1=2

fib3=fib1+fib2=1+2=3

fib4=fib2+fib3=2+3=5

…

bu işlemin kodu aşağıdaki şekilde yazılabilir:

```
int fibonacci(int n)

{
  if (0 == n) {
      return 0;
   } else if (1 == n) {
      return 1;
   } else {
```

```
        return fibonacci(n - 2) + fibonacci(n - 1);
    }
}
```

Yukarıda verilen bu recursive (kendi kendini çağıran) koda bakıldığında ve kodun tahlili yapıldığında aşağıdaki fonksiyon iç içe çağırma ağacı (recursion tree) fark edilir:

```
fibonacci(4)
  +-----------------------------
  |                       |
fibonacci(2)              fibonacci(3)
  +-----------------          +---------------
  |             |          |            |
fibonacci(0)   fibonacci(1)  fibonacci(1)  fibonacci(2)
                                          +-----------
                                          |         |
                                      fibonacci(0) fibonacci(1)
```

yani yukarıdaki örnekte, fibonacci(4) fonksiyonu için çağırma işlemleri sırasıyla gösterilmiştir.

4.2.4. **Fibonacci Serisini basan kod**

Yukarıdaki kodlarda, verilen sıradaki fibonacci serisinin elemanını basan basamakları yazmıştık. Verilen sayıya kadar olan bütün fibonacci sayılarını basan c dilinde ki kod şekil 4.5 deki gibidir.

```c
#include <stdio.h>
#include <conio.h>
//www.bilgisayarkavramlari.com
int f(int n){
    int a = 1;
    int b = 1;
    for(int i = 0;i<=n;i++){
        printf("%d ",a);
        int c = a+b;
        a=b;
        b=c;
    }
}
int main(){
    f(8);
    getch();
}
```

Şekil – 4.5- *fibonacci serisini basan iterative kod*

Kodun çalışan hali şekil 4.6 daki gibidir.

Şekil – 4.6- *iterative fibonacci kodunun çalışan hali*

Yukarıdaki kod dikkat edileceği üzere iteratif (döngü ile) yazılmış bir koddur. Şayet aynı işi **özyineli (recursive)** olarak yapmak istersek şekil 4.7 deki problem ile karşılaşırız:

```c
#include <stdio.h>
#include <conio.h>
//www.bilgisayarkavramlari.com
int f(int n){
   if(n==0||n==1)
                return 1;
   int deger = f(n-1)+f(n-2);
   printf("%d ",deger);
   return deger;
}
int main(){
    f(8);
    getch();
}
```

Şekil – 4.7 -*recuesive fibonaci kod*

Yukarıdaki kodun çalışan hali şekil 4.8 deki gibidir :

Şekil 4.8 *recursive fibonacci kodun çalışan hali*

Görüldüğü üzere 8. elemanı bastırmak için hesapladığımız bütün ara değerler ekrana bastırılıyor. Yani yukarıdaki recursion tree (özyineleme ağacının) tamamı basılıyor. Bu durumda özyineli kodlamanın bu iş için uygun olmadığını söyleyebiliriz. Ancak yine de özyineli olarak fibonacci serisini basmakta ısrar edilirse aşağıdaki şekilde performansı hiçe sayan bir kod şekil 4.9 daki gibi yazılabilir:

```
1  #include <stdio.h>
2  #include <conio.h>
3  int f(int n){
4     if(n==0||n==1)
5                  return 1;
6     int deger = f(n-1)+f(n-2);
7     return deger;
8  }
9  int fibo(int n){
10       if(n>0)
11          fibo(n-1);
12       printf("%d ",f(n));
13 }
14 int main(){
15     fibo(8);
16     getch();
17 }
```

Şekil 4.9- *yeniden düzenlenmiş fibonacci kodu*

Yukarıdaki kodun çalışan hali şekil 4.10 daki gibidir.

Şekil 4.10 *düzenlenmiş fib. Kodun çalışan hali*

Yukarıdaki bu yeni özyineli fonksiyonda her değer için tekrar ve tekrar fibonacci ağacı oluşturularak seri hesaplanmaktadır.

4.2.5. İkili Arama (Binary Search)

Özyineleme sadece matematiksel fonksiyonların tanımlamasında kullanılan başarılı bir araç değildir. Arama gibi hesaplama etkinliklerinde de oldukça kullanışlıdır. Belirli bir

sayıdaki eleman içerisinde belli bir elemanı bulmaya çalışma işlemine arama adı verilir. Elemanların sıralanması arama işlemini kolaylaştırır. İkili arama da sıralanmış elemanlar üzerinde gerçekleştirilir. Bir dizinin iki parçaya bölünmesi ve uygun parçada aynı işlemin sürdürülmesi ile yapılan arama işlemidir. Telefon rehberinin ortasını açıp, soyadına göre önce ise ilk yarıda, sonra ise ikinci yarıda aynı işlemi tekrarlama yolu ile arama, telefon rehberindeki baştan itibaren sona doğru sıra ile herkese bakmaktan çok daha etkindir.

```
int binsrch(int a[], int x, int low, int high)
{
 int mid;
 if(low>high)
  return(-1);
 mid=(low+high)/2;
 return(x==a[mid] ? mid : x<a[mid] ? binsrch(a,x,low,mid-1) : binsrch(a,x,mid+1,high) );
}
```

```
int i; int a[8] = { 1,3,4,5,17,18,31,33 }; // a dizisi : 1  3  4  5  17  18  31  33
i = binsrch(a,17,0,7); // 0 ve 7. elemanlar arasında 17 sayısının aratılmasını sağlar.
```

---

binsrch(a,17,0,7);

mid = (0+7)/2 = 3. eleman

a[mid] = 5 => 17>5

binsrch(a,17,4,7);

mid = (4+7)/2 = 5.eleman

a[mid] = 18 => 17<18

binsrch(a,17,4,4);

mid = (4+4)/2 = 4. eleman

a[mid] = 17 => 17==17

return(17); // Bulundu

a ve x global değişken olarak tanımlanırsa fonksiyon ve çağrımı şu şekilde olacaktır :

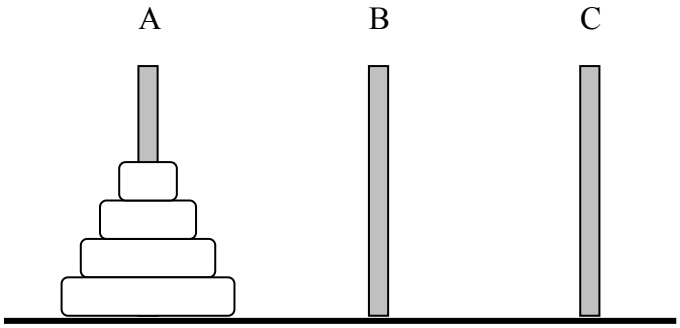
```
int binsrch(int low, int high)
{
  int mid;
  if(low>high)
    return(-1);
  mid=(low+high)/2;
  return(x==a[mid] ? mid : x<a[mid] ? binsrch(low,mid-1) : binsrch(mid+1,high)
);
}
... i = binsrch(0,n-1);
```

4.2.6. Hanoi Kuleleri Problemi (Towers of Hanoi Problem)

Üç kule (A,B,C) olan bir sistemde yarıçapı birbirinden farklı 4 tane diskin A kulesine yerleştirildiğini düşünün (Şekil 4.11).

Kurallar :

- Bir diskin altında yarıçapı daha küçük bir disk bulunamaz. Bir diskin üzerine yarıçapı daha büyük bir disk yerleştirilemez.
- Bir anda bir kulenin sadece en üstündeki disk diğer bir kuleye yerleştirilebilir. Bir anda bir disk hareket ettirilebilir.



Şekil 4.11- *Hanoi Kulesi*

Hanoi Kulesinin İlk Yerleştirimi

Problemde istenen : B direğinden de yararlanarak tüm disklerin C'ye yerleştirilmesi.

Fonksiyonu şu şekilde tanımlarsak;

$$hanoi(n) = \begin{cases} 1 & if\ n = 1 \\ 2.hanoi(n-1) + 1 & if\ n > 1 \end{cases}$$

Yerleştirme işlemini en kısa kaç adımda yapabileceğimizi bulabiliriz.

$$h_n = 2h_{n-1} + 1$$
$$h_1 = 1$$

alırsak, n=4 halkanın kaç adımda A dan Cye aktarılacağını şöyle buluruz;

hanoi(4) = 2*hanoi(3) + 1
= 2*(2*hanoi(2) + 1) + 1
= 2*(2*(2*hanoi(1) + 1) + 1) + 1
= 2*(2*(2*1 + 1) + 1) + 1

= 2*(2*(3) + 1) + 1
= 2*(7) + 1
= 15

Görüldüğü gibi burada da yine fonksiyon, işlemi yaparken sonuca ulaşmak için sürekli kendi kendini çağırmakta veya özyinelemektedir.

4.6.7 Hanoi Towers Probleminin Çözümü ve C Programı Haline Getirilmesi:

Önce n disk için düşünelim.

- 1 disk olursa, doğrudan A'dan C'ye konulabilir (Doğrudan görülüyor).
- (n-1) disk cinsinden çözüm üretebilirsek özyinelemeden yararlanarak n disk için de çözümü bulabiliriz.
- 4 diskli bir sistemde, kuralara göre en üstteki 3 diski B'ye yerleştirebilirsek çözüm kolaylaşır.

Genelleştirirsek :

1. n==1 => A'dan C'ye diski koy ve bitir.
2. En üstteki n-1 diski A'dan C'den yararlanarak B'ye aktar.
3. Kalan diski A'dan C'ye koy.
4. Kalan n-1 diski A'dan yararlanarak B'den C'ye koy.

Problem deyimi : "Hanoi Probleminin Çözümü"

Problem şu an tam olarak tanımlı değil, problem deyimi yeterli değil.

Bir diskin bir kuleden diğerine taşınmasını bilgisayarda nasıl temsil edeceğiz?

Programın Girdileri nelerdir? Çıktıları nelerdir? belli değil.

Girdi/Çıktı tasarımı herhangi bir problemin çözümünün programın algoritmasının oluşturulmasında önemli yer tutar. Oluşturulacak algoritmanın yapısı da büyük ölçüde girdi ve çıktılara bağlı olacaktır. Uygun girdi ve çıktı tasarımı, algoritmaların geliştirilmesini kolaylaştırabilir ve etkinlik sağlar.

Girdi :

disk sayısı : n

kuleler : A, B, C (uygun)

Çıktı :

disk nnn'i, yyy kulesinden alıp zzz kulesine yerleştir.

nnn : disk numarası. En küçük disk 1 numara (en küçük sayı olması doğrudan)

yyy ve zzz'de kule adı.

Ana programdan towers fonksiyonunun çağrılması :

```
void main()
{
  int n;
  scanf("%d",&n);
  towers(parameters);
}
```

Şimdi parametreleri belirleme aşamasına gelindi :

```
#include <stdio.h>
void towers(int, char, char, char);
void main()
{
  int n;
  scanf("%d",&n);
```

```
  towers(n,’A’,’B’,’C’);
}

void towers(int n, char frompeg, char topeg, char auxpeg)
{
  if(n==1) {
    printf("\n%d%s%c%s%c%s",n,". diski ",frompeg," kulesinden alıp ",
                                    topeg, " kulesine yerleştir");
    return;
  };
  towers(n-1, frompeg,auxpeg,topeg); // n-1 diskin yardımcı kuleye konulması
  printf("\n%d%s%c%s%c%s",n,". diski ",frompeg," kulesinden alıp ",topeg, " kulesine yerleştir");
  towers(n-1, auxpeg,topeg,frompeg); // n-1 diskin hedef kuleye konulması
}
```

Programın n=3 disk için çalıştırılması sonucu oluşan ekran çıktısı :

1. diski A kulesinden alıp C kulesine yerleştir

2. diski A kulesinden alıp B kulesine yerleştir

1. diski C kulesinden alıp B kulesine yerleştir

3. diski A kulesinden alıp C kulesine yerleştir

1. diski B kulesinden alıp A kulesine yerleştir

2. diski B kulesinden alıp C kulesine yerleştir

1. diski A kulesinden alıp C kulesine yerleştir

4.2.8. Tek Bağlı Listede Elemanların Ters Sırada Listelenmesinde Özyineleme

```
Procedure RevPrint (list:ListType)
 begin
  if list<>NIL
   then
    begin
     RevPrint(List^.Next);
     write(list^.info)
    end;
 end;
```

Ekran Çıktısı **:** E D C B A

Elemanları bir yığıta koyup ters sırada listelemek yerine doğrudan özyineleme kullanmak daha basit ve doğal.

4.2.9. Özyineleme Zinciri

Özyineli bir fonksiyonun kendisini doğrudan çağırması gerekmez. Dolaylı olarak da çağırabilir. Örnek bir yapı şu şekildedir :

```
a(parametreler)
{
.....
  b(değerler);
.....
}
```

```
b(parametreler)
{
.....
  a(değerler);
.....
}
```

4.2.10 Özyineleme (Recursion) ve İterasyon (Iteration)

Herhangi bir fonksiyonun iteratif (iterative) yani tekrarlı versiyonu, özyineli (recursive) versiyonundan zaman (time) ve yer (space) bakımından genelde daha etkindir. Bunun nedeni, özyinelemede fonksiyonun her çağrılışında fonksiyona giriş ve çıkışta oluşan yüklerdir.

Bununla birlikte genelde yapısı uygun olan problemlerin çözümünde özyinelemenin kullanılması daha doğal ve mantıklıdır. Tanımlamalardan çözüme doğrudan ulaşılabilir. Yığıt kullanımı gerektiren durumlarda özyineli olmayan çözümlerin geliştirilmesi zordur ve hataları gidermek için daha fazla çaba harcamak gerekir. Örnek olarak, tek bağlı listedeki elemanların ters sırada yazdırılması verilebilir. Özyineli çözümde yığıt otomatik olarak oluşmaktadır ve ayrıca yığıt kullanmaya gerek kalmaz.

İterasyonda "Control Structure" olarak döngüler yolu ile tekrar kullanılırken, özyinelemede seçim yapısı kullanılır. İterasyonda tekrar doğrudan sağlanırken, özyinelemede sürekli fonksiyon çağrıları ile sağlanır. İterasyon, döngü durum şartı geçersizliğinde sonlanırken, özyineleme en basit duruma (simplest case = base case) ulaşıldığında sonlanır. İterasyonda kullanılan sayaç değeri değiştirilerek problem çözülürken, özyinelemede orijinal problemin daha basit sürümleri oluşturularak çözüme ulaşılır.

4.3. CCC LERDE RECURSİVE FONKSİYONLARI

GİRİŞ 4.3.1. :

Bu bölümde dedekind in bir önermesinin genellemesi yapılacaktır. Dedekind, **Set** kümeler kategorisinde bir w kümesinin şu şekilde karakterize edilebileceğini söyler; X herhangi bir küme, $x \epsilon X \ ve \ f: X \longrightarrow X$ bir fonksiyon olsun. Bu durumda $F(0) = x \ ve \ F(x+1) = f(F(x))$ olacak biçimde bir tek $F: w \longrightarrow X$ dönüşümü vardır.

4.3.2. Recursive Fonksiyonlar

Tanım 4.3.2.1. : $N = \{0,1,2,\dots\}$ olmak üzere C , $N^k \longrightarrow N$ fonksiyonlarının kümesi olsun. Bir başka ifade ile sayısal (numerical) fonksiyonların kümesi olsun.

(1) $g_1(a), \dots, g_m(a), \quad h(b_1, \dots, b_m) \epsilon$ C için

$$h\big(g_1(a), \dots, g_m(a)\big) \epsilon \text{ C}$$

oluyorsa C ye substitution altında kapalı,

(2) $g(a)\ ve\ h(n,m,a)\ için\ f(n,a) \epsilon$ C oluyorsa

(Burada $f(o,a) = g(a)\ ve\ f(s(n),a) = h(n, f(n,a), a)$)

C ye primitive özyineleme altında kapalı,

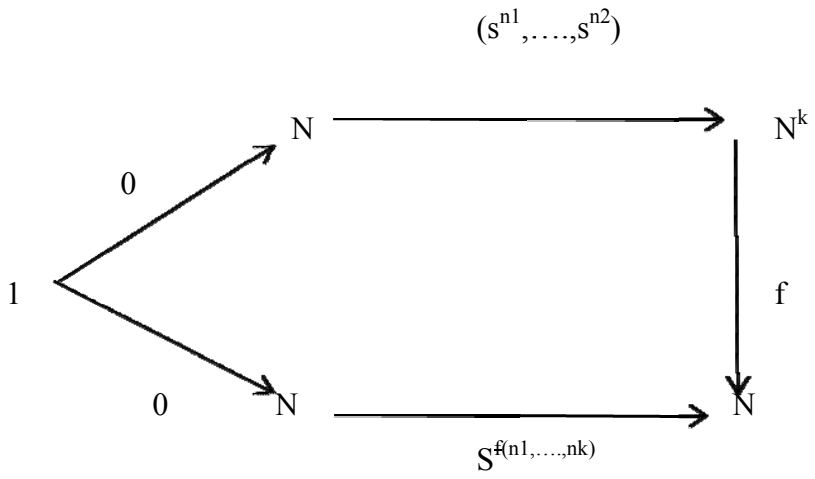
(3) Her $a \epsilon N^k\ için\ g(a,n) \epsilon$ C olduğunda $g(a,n) = 0$ olacak biçimde $n \epsilon N$ varsa minimizasyon altında kapalı denir.

$S_n \equiv n+1, O_n \equiv 0\ ve\ \pi_i(n_1, \dots, n_k) \equiv n_k \quad (i = 1,2, \dots, k)$

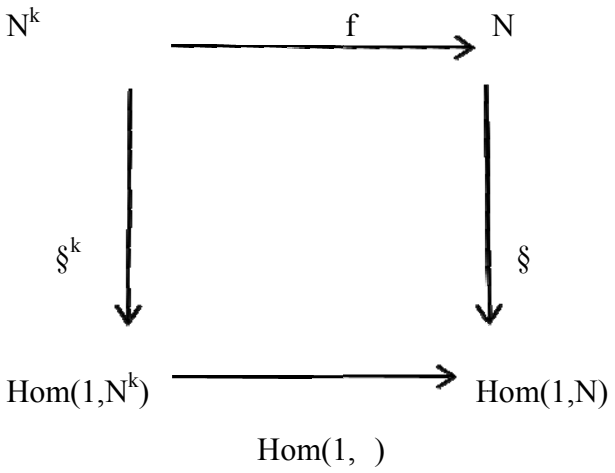
Fonksiyonları en temel nümerik fonksiyonlardan bazılarıdır. Bu fonksiyonların her biri primitive özyinelidir. Yine primitive özyineli(recursive) fonksiyonlar bileşke işlemi altında kapalıdır.

Şimdi nümerik fonksiyonların CCC lerde gösterimini inceliyelim. Burada $N^k \longrightarrow N$ fonksiyonlarının zayıf doğal sayılar nesnesine sahip CCC lerde oklar cinsinden nasıl ifade edilebileceğini araştıracağız.

Tanım 4.3.2.2. : C, ,zayıf doğal sayılar nesnesi N yi içeren bir Kartezyen kapalı kategori ve $f: N^k \longrightarrow N$ bir nümerik fonksiyon olsun. Doğal sayılardan oluşan her $(n_1, \dots, n_k)$ sıralı k-lisi için



Diyagramını değişmeli yapacak biçimde dönüşümü varsa, veya



Diyagramı değişmeli ise dönüşümüne ile gösteriliyor denir. Benzer şekilde type -kalkülüs içinde recursive fonksiyonlarının gösterimi tanımı verilebilir.

KAYNAKLAR

- **Blyth T.S. 1986.,** Categories, Longman,
- **Borceux F. 1994.,** Handbook of Categorical algebra2, Cambridge university Press,
- **Brainerd, W.S. 1974.,** Landweber, L.H. , *Theory of Computation*, Wiley, ISBN 0-471-09585-0,
- **George Boolos, 2002. John Burgess, Richard Jeffrey,** *Computability and Logic: Fourth Edition*, Cambridge University Press, Cambridge, UK. Cf pp. 70-71,
- **John McCarthy, 1960.** *Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I*, Communications of the ACM, 3, 184-195, April
- **Kleene, S. C. 1952.** Introduction to Metamathematics. Princeton, NJ: Van Nostrand,
- **Kleene Stephan, 1952.** *Introduction to Metamathematics*, North-Holland Publishing Company, New York, 11th reprint 1971: (2nd edition notes added on 6th reprint). In Chapter XI. General Recursive Functions §57,
- **Lambek J. 1986.,** Scoot P.J., İntroduction to higher order categorical logic, Cambridge,
- **Marvin L. Minsky, 1967**. *Computation: Finite and Infinite Machines*, Prentice-Hall, Inc. Englewood Cliffs, N.J,
- **Odifreddi, P. 2001.** In Combinatorics, Computability and Logic (Ed. C. S. Calude, M. J. Dinneen, and S. Sburlan). London: Springer-Verlag,
- **Odifreddi, P.G. 1999,** *Classical Recursion Theory*, North Holland; second edition,
- **Péter, R**. **Rekursive 1951**. Funktionen in der Komputer-Theorie. Budapest: Akad. Kiado,
- **Rogers, H. 1987.** Theory of Recursive Functions and Effective Computability. Cambridge, MA: MIT Press,

- **Schnorr, C. P. 1974**. Rekursive Funktionen und ihre Komplexität. Stuttgart, Germany: Teubner,
- **Soare Robert I. 1987,** *Recursively Enumerable Sets and Degrees*, Springer-Verlag,
- **Soare Robert I. Soare,1995** *Computability and Recursion*
- **Şeker, Şadi Evren, 2006**, www.bilgisayarterimleri.com ,
- **Wolfram, S. 2002.** A New Kind of Science. Champaign, IL: Wolfram Media, p. 907,

ÖZGEÇMİŞ

Adı Soyadı : Savaş TANRIÖVEN

Doğum Yeri : Kayseri

Doğum Tarihi : 09. 07. 1976

Medeni Hali : Evli

Yabancı Dili : İngilizce

Eğitim Durumu (Kurum ve Yıl)

Lise : Kayseri Merkez Teknik Lisesi, Elektrik Bölümü 1994

Lisans : Selçuk Üniversitesi, Fen-Edebiyat Fakültesi, Matematik Bölümü, 2000

Çalıştığı Kurum/Kurumlar ve Yıl

…