Grup Cebirlerde Hesaplamalar

Mustafa ÜNLÜ

YÜKSEK LİSANS TEZİ
Matematik Anabilim Dalı
Temmuz 2010

Computations in Group Algebras

Mustafa ÜNLÜ

MASTER DISSERTATION
Department of Mathematics
July 2010

Grup Cebirlerde Hesaplamalar

Mustafa ÜNLÜ

Eskişehir Osmangazi Üniversitesi

Fen Bilimleri Enstitüsü

Lisansüstü Yönetmeliği Uyarınca

Matematik Anabilim Dalı

Cebir ve Sayılar Teorisi Bilim Dalında

YÜKSEK LİSANS TEZİ

Olarak Hazırlanmıştır

Danışman: Yrd. Doc. Dr. Alper ODABAS

Temmuz 2010

ONAY

Matematik Anabilim Dalı yüksek lisans öğrencisi Mustafa ÜNLÜ' nün YÜKSEK LİSANS TEZİ olarak hazırladığı "**Grup Cebirlerde Hesaplamalar**" başlıklı bu çalışma, jürimizce lisans üstü yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

Danışman : Yrd. Doç. Dr. Alper ODABAŞ

İkinci Danışman : –

Yüksek Lisans Tez Savunma Jürisi:

Üye : Prof. Dr. Zekeriya ARVASİ

Üye : Prof. Dr. Mahmut KOÇAK

Üye : Yrd. Doç. Dr. İlker AKÇA

Üye : Yrd. Doç. Dr. Enver Önder USLU

Üye : Yrd. Doç. Dr. Alper ODABAŞ

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... tarih ve ..... sayılı kararıyla onaylanmıştır.

Prof. Dr. Nimetullah BURNAK

Enstitü Müdürü

ÖZET

Bu tez çalışması üç bölümden oluşmaktadır. Birinci bölümde cebir ve grup cebir yapısının temel özellikleri verilmiştir. Cebirin bir bilgisayar uygulaması olan GAP, yeni matematiksel yapıların bilgisayar ortamına aktarılmasında kullanılan güçlü bir programlama dilidir. GAP oldukça gelişmiş, anlaşılması kolay ve serbest kullanıma açık bir dile sahiptir. Özellikle grup teorisinde güçlüdür. İkinci bölümünde GAP programlama dili ve GAP fonksiyon yazımı üzerinde durulmuştur. Tezin son bölümünde grup cebirlerler, cat1-cebirler, cat-1 Lie cebirler ve çaprazlanmış modüllerde hesaplamalar verilmiştir.

Anahtar Kelimeler: GAP, Grup Cebirler, Lie Cebirler, Cat1-Lie Cebirler.

SUMMARY

This thesis consist of three chapters. In the first chapter some basic properties of algebras and group algebras have been given. The powerful computer algebra system GAP provides a high level programming language with several advantages for coding of new mathematical structures. It has a highly developed, easy to understand programming language, incorporated. It is especially powerful for group theory. In the second chapter GAP programming language and GAP function writing has been asserted. In the last chapter, computations of group algebras, cat1-algebras, cat1-Lie algebras and crossed modules have been given.

Keywords: GAP, Group algebras, Lie algebras, Cat1- Lie algebras.

TEŞEKKÜR

Beni bu çalışmaya sevkeden ve yöneten, çalışma boyunca değerli yardımlarını esirgemeyen,

Sayın hocalarım,

Prof. Dr. Zekeriya ARVASİ *ve* **Yrd. Doç Dr. Alper ODABAŞ'**a

desteklerini hep yanımda hissettiğim aileme sonsuz saygı ve teşekkürlerimi sunarım.

İÇİNDEKİLER

BÖLÜM 1

GRUP CEBİRLER

1.1 Giriş

Cayley (1854) tarafından tanımlanan grup cebir yapısı herhangi bir gruptan cebir elde etmenin yanı sıra özellikle temsil teorisi için kullanışlı bir araçtır. Bu bölümde grup cebir yapısı ve bu yapıyı elde edebilmek için kullanılan diğer cebirsel yapılar adım adım anlatılmıştır. Bu çalışmamızda sıkça söz edeceğimiz, modül teoride de iyi bilinen aşağıdaki kavramları kısaca hatırlatalım.

R-modül: R bir halka olsun. Her $m, m_1, m_2 \in M$ ve $r, r_1, r_2 \in R$ için

$$\begin{aligned} R \times M &\longrightarrow M \\ (r,m) &\longmapsto rm \end{aligned}$$

çarpımı ile

$$\begin{aligned} r(m_1+m_2) &= rm_1+rm_2 \\ (r_1+r_2)m &= r_1m+r_2m \\ (r_1r_2)m &= r_1(r_2m) \end{aligned}$$

şartlarını sağlayan, bir M toplamsal abelyan gruba *sol R-modül* denir. Çarpım

$$\begin{aligned} M \times R &\longrightarrow M \\ (m,r) &\longmapsto mr \end{aligned}$$

şeklinde sağdan tanımlı ise *M bir sağ R-modül* yapısı oluşturur.

Örnek 1.1 R bir halka olmak üzere, herhangi bir A abelyan grubu $r \in R$, $a \in A$ için

$$\begin{aligned} R \times A &\longrightarrow A \\ (r,a) &\longmapsto ra = 0 \end{aligned}$$

tanımlaması ile bir R- modül yapısı oluşturur.

Birimli R-modül: R birimli bir halka, M bir R-modül olmak üzere, her $m \in M$ için

$$1_R m = m$$

ise M ye *birimli R-modül* denir.

Örnek 1.2 Her G toplamsal abelyan grup bir birimli $\mathbb{Z}$-modüldür.

Serbest R-Modül: M birimli sol (sağ) R-modül bir R-bazına sahipse M ye *serbest sol (sağ) R-modül* denir.

Örnek 1.3 R, bir cisim ise R-modül bir vektör uzayı oluşturur.

Örnek 1.4 Bir komütatif R halkasının N ideali, bir R-modüldür. Çünkü; N ideal olduğundan bir abelyan grup yapısı oluşturur ve her $r \in R$ için $rN \subset N$ dir. Dolayısıyla,

$$\begin{array}{ccc} R \times N & \longrightarrow & N \\ (r,n) & \longmapsto & rn \end{array}$$

çarpımı kapalıdır ve modül şartları sağlanır.

Modül homomorfizmi: M ve N, iki R-modül olsun.

$$f : M \longrightarrow N$$

fonksiyonu, her $x, y \in M$ ve $r \in R$ için,

$$\begin{array}{rcl} f(x+y) & = & f(x)+f(y) \\ f(rx) & = & rf(x) \end{array}$$

şartlarını sağlıyor ise $f : M \to N$ ye bir *R-modül homomorfizmi* denir.

Alt modül: M bir R-modül olsun. M', M nin alt grubu olmak üzere

$$m' \in M' \text{ ve her } r \in R \text{ için } rm' \in M'$$

ise M' ye M nin bir *alt modülü* denir.

Örnek 1.5 $f : M \to N$ bir R-modül homomorfizmi olsun. Bu durumda

$$\text{Çek} f = \{m \in M : f(m) = 0\}$$

M nin alt modülüdür. Çünkü; Çek f, M nin alt grubu ve $m \in M$, $r \in R$ için

$$f(rm) = rf(m) = r0 = 0$$

olduğundan $rm \in$Çek f dir. Ayrıca,

$$f(M) = \{n \in N : n = f(m), m \in M\}$$

de N nin alt modülüdür. Çünkü; $f(M)$, N nin alt grubu ve $n \in f(M)$, $r \in R$ için

$$nr = f(m)r = f(mr)$$

ve M bir R-modül olduğundan $mr \in M$ dir. Dolayısıyla,

$$nr \in f(M)$$

elde edilir.

Bilineerlik: R komütatif halka, A, B ve G, R-modül olmak üzere

$$f : A \times B \longrightarrow G$$

fonksiyonu,

$$\begin{aligned} f(a+a',b) &= f(a,b)+f(a',b) \\ f(a,b+b') &= f(a,b)+f(a,b') \\ rf(a,b) &= f(ra,b) = f(a,rb) \end{aligned}$$

şartlarını sağlıyor ise f ye *R-bilineer* denir.

Bir grubun herhangi bir küme üzerine etkisi:

G bir grup ve S herhangi bir küme olmak üzere, her $x \in S$, $g_1, g_2 \in G$ için

$$\begin{aligned} G \times S &\longrightarrow S \\ (g,x) &\longmapsto gx \end{aligned}$$

fonksiyonu

$$ex = x \quad \text{ve} \quad (g_1g_2)x = g_1(g_2x)$$

eşitliklerini sağlıyor ise bu fonksiyona G grubunun S kümesi üzerine *sol etkisi* denir. *Sağ etki* ise xg olarak tanımlanır.

Örnek 1.6 S_n, simetri grubunun $I_n = \{1,2,\ldots,n\}$ kümesi üzerine etkisi

$$\begin{aligned} S_n \times I_n &\longrightarrow I_n \\ (\sigma,x) &\longmapsto \sigma(x) \end{aligned}$$

ile tanımlanır.

Örnek 1.7 H, bir G grubunun alt grubu olmak üzere, H ın G üzerine etkisi, eşlenik fonksiyonu adı verilen

$$\begin{aligned} H \times G &\longrightarrow G \\ (h,x) &\longmapsto hxh^{-1} \end{aligned}$$

fonksiyonu ile tanımlanır.

1.2 Cebirler

A ve B birimli değişmeli halka ve $f : A \longrightarrow B$ halka homomorfizmi olsun. $a \in A, b \in B$ için

$$\begin{array}{ccc} A \times B & \longrightarrow & B \\ (a,b) & \longmapsto & a \cdot b = f(a)b \end{array}$$

işlemi tanımlayalım. Bu çarpımla birlikte B de bir A-modül yapısı oluşturur. Böylece B halka yapısı ile birlikte aynı zamanda modül yapısına da sahiptir.

Tanım 1.1 B halka yapısı aynı zamanda A-modül yapısına sahip ise B ye A-cebir denir.

Tanım 1.2 R değişmeli halka, A, B ve G birer R-modül olmak üzere $f : A \times B \to G$ fonksiyonu

$$\begin{array}{ll} i) & f(a+a',b) = f(a,b) + f(a',b) \\ ii) & f(a,b+b') = f(a,b) + f(a,b') \\ ii) & rf(a,b) = f(ra,b) = f(a,rb) \end{array}$$

şartlarını sağlıyor ise f ye R-bilineer denir.

Uyarı 1.3 1) A değişmeli fakat B değişmeli olmasın. Her $a \in A$ ve $b_1, b_2 \in B$ için

$$a(b_1 b_2) = (ab_1)b_2 = b_1(ab_2)$$

ise B ye A-cebir denir.

2) A değişmeli halka ve B bir A-modül olsun. Eğer

$$\begin{array}{rccc} f: & B \times B & \longrightarrow & B \\ & (b_1, b_2) & \longmapsto & \langle b_1, b_2 \rangle \end{array}$$

bilineer fonksiyonu ise B bir A-cebirdir. f *bilineer* ise $b_1 \longmapsto \langle b_1, b_2 \rangle$ ve $b_2 \longmapsto \langle b_1, b_2 \rangle$ homomorfizmleri

$$\begin{array}{rl} f(b_1 + b_1') & = \langle b_1 + b_1', b_2 \rangle = \langle b_1, b_2 \rangle + \langle b_1', b_2 \rangle \\ & = f(b_1) + f(b_1') \\ f(b_1 b_1') & = \langle b_1 b_1', b_2 \rangle = \langle b_1, b_2 \rangle \langle b_1', b_2 \rangle \end{array}$$

$$\begin{array}{rl} f(b_2 + b_2') & = \langle b_1, b_2 + b_2' \rangle = \langle b_1, b_2 \rangle + \langle b_1, b_2' \rangle \\ & = f(b_2) + f(b_2') \end{array}$$

şartlarını sağlar.

Örnek 1.8 Her R halkası toplamsal Abelyan grup ve her Abelyan grup bir $\mathbb{Z}-$modül olmak üzere, $\forall n \in \mathbb{Z}$, $g_1, g_2 \in R$ için

$$\begin{array}{rcll} n.(g_1,g_2) & = & (g_1g_2)+...+(g_1g_2) & \\ & = & (g_1+g_1+...+g_1)\,g_2 & (\because g_1,g_2 \in R, R \text{ halka}) \\ & = & (n.g_1)\,g_2 & \\ n.(g_1g_2) & = & (g_1g_2)+...+(g_1g_2) & \\ & = & g_1\,(g_2+g_2+...+g_2) & \\ & & g_1\,(n.g_2) & \end{array}$$

olup $n.(g_1g_2) = (n.g_1)\,g_2 = g_1\,(n.g_2)$ eşitliği sağlanır. R bir $\mathbb{Z}-$cebirdir.

Örnek 1.9 Her değişmeli R halkası bir $R-$cebirdir. Her $r, r_1, r_2 \in R$ için

$$\begin{array}{rcll} r.(r_1r_2) & = & r\,(r_1r_2) & (\because \text{etki tanımı}) \\ & = & (rr_1)\,r_2 & (\because R \text{ halka, H2}) \\ & = & (r.r_1)\,r_2 & \end{array}$$

$r.(r_1r_2) = r\,(r_1r_2) = (rr_1)\,r_2 = (r_1r)\,r_2 = r_1\,(rr_2) = r_1\,(r.r_2)$ olup cebir şartı sağlanır.

Örnek 1.10 R, S halka ve $\phi : R \to S$ dönüşümü ise

$$I_m\phi \subseteq Mer(S) = \{s \in S \mid ss' = s's\,,\ \forall s' \in S\}$$

olacak şekilde bir halka homomorfizması olsun. M bir S-modül ise $r \in R$, $m \in M$ için $r \cdot m = (\phi(r)) \odot m$ şeklinde tanımlanan etkiyle birlikte M bir R-modül ve S bir R-cebirdir.

$$\begin{array}{rcl} \mathbb{R} \times M & \to & M \\ (r,m) & \longmapsto & r \cdot m = (\phi(r)) \odot m \end{array}$$

M1-) $(M,+)^{\sim}$ Abel grup $(\because M, S\text{-modül})$

M2-) $$\begin{array}{rcll} r \cdot (m_1+m_2) & = & (\phi(r)) \odot (m_1+m_2) & \\ & = & (\phi(r) \odot m_1) + (\phi(r) \odot m_2) & (\because M, S\text{-modül}, M2) \\ & = & r \cdot m_1 + r \cdot m_2 & (\because \text{ etki tanımı}) \end{array}$$

M3-) $$\begin{array}{rcll} (r_1+r_2) \cdot m & = & \phi(r_1+r_2) \odot m & \\ & = & [\phi(r_1)+\phi(r_2)] \odot m & (\because \phi \text{ halka homomorfizmi}) \\ & = & \phi(r_1) \odot m + \phi(r_2) \odot m & (\because M, S\text{-modül}) \\ & = & r_1 \cdot m + r_2 \cdot m & \end{array}$$

M4-) $$\begin{array}{rcl} (r_1r_2) \cdot m & = & \phi(r_1r_2) \odot m \\ & = & (\phi(r_1)\phi(r_2)) \odot m \\ & = & \phi(r_1) \odot (\phi(r_2) \odot m) \\ & = & \phi(r_1) \odot (r_2 \cdot m) \\ & = & r_1 \cdot (r_2 \cdot m) \end{array}$$

şartları sağlandığından M bir $\mathbb{R}$-modülüdür. Burada S bir S-modülü olduğundan M yerine S alabiliriz.Bu durumda S bir R-modüldür.

S 'nin bir R-cebir olduğunu gösterelim.S halka ve R-modül olduğundan S'nin R-cebir olması için $\forall r \in R$, $s_1, s_2 \in S$ için

$$r \cdot (s_1 s_2) = (r \cdot s_1) s_2 = s_1 (r \cdot s_2)$$

eşitliği sağlanmalıdır.

$$\begin{aligned}
r.(s_1 s_2) &= \phi(r) \odot (s_1 s_2) \\
&= \phi(r)(s_1 s_2) \\
&= (\phi(r) s_1) s_2 \\
&= (r \cdot s_1) s_2 \\
r.(s_1 s_2) &= \phi(r) \odot (s_1 s_2) \\
&= \phi(r)(s_1 s_2) \\
&= (\phi(r) s_1) s_2 \\
&= (s_1 \phi(r) s_2 \\
&= s_1 (\phi(r) s_2) \\
&= s_1 (r \cdot s_2)
\end{aligned}$$

olup S bir R-cebirdir.

Örnek 1.11 A birimli değişmeli halka olsun.

$$\begin{array}{rccc} f: & A & \hookrightarrow & A[X] \\ & a & \longmapsto & a = a + 0x + \ldots \end{array}$$

fonksiyonu bir homomorfizmdir. Böylece $A[X]$ bir A-cebirdir.

Örnek 1.12 V F-vektör uzayı olsun.

$$Hom_F(V,V) = \{ f \mid f : V \longrightarrow V \text{ lineer dönüşüm} \}$$

bir F-cebirdir.

$$\begin{array}{ccc} F \times Hom(V,V) & \longrightarrow & Hom(V,V) \\ (\alpha, f) & \longmapsto & \alpha \cdot f : \begin{array}{ccc} V & \longrightarrow & V \\ v & \longmapsto & (\alpha \cdot f)(v) = f(\alpha v) \end{array} \end{array}$$

şeklinde tanımlanan işlemle $Hom(V,V)$ bir F-cebirdir.

Örnek 1.13 A birimli değişmeli halka ve $Mat_{n\times n}(A)$ $n\times n$ tipindeki matrisler halkası verilsin. Bu durumda $Mat_{n\times n}(A)$ bir A-cebirdir.

$$\begin{array}{ccc} A\times Mat_{n\times n}(A) & \longrightarrow & Mat_{n\times n}(A) \\ (a,(a_{ij})) & \longmapsto & a\cdot(a_{ij})=(aa_{ij}) \end{array}$$

olmak üzere genel olarak B bir A-cebir ise $Mat_{n\times n}(B)$ bir A-cebirdir.

1.3 Grup Cebirleri

$R[G]$ ile gösterilen grup halka yapısı bir halkadır. Grup halka kavramı ilk olarak Cayley (1854) tarafından tanımlanmıştır. Bu yapıda R bir halka G ise bir gruptur. Bir grup halkanın elemanları G grubunun elemanlarının sonlu lineer kombinasyonları ile R nin elemanlarının katsayı olarak kullanılmasıyla oluşur.

Her $R[G]$ grup halkası için $R\leq R[G]$ olduğundan $R[G]$ bir R-modüldür. Eğer R bir cisim ise (değişmeli halka), grup halka yapısı *grup cebir* olarak adlandırılır. $R=\mathbb{Z}$ alınırsa $\mathbb{Z}[G]$, $\mathbb{Z}$-cebirine tam grup halka (integral group ring) adı verilir. Grup cebirler hakkında ayrıntılı bilgi için (Connel, 1963) ve (Passman, 1977) çalışmalarına bakılabilir.

Tanım 1.4 K bir cisim, $(G,*)$ bir grup olsun. Her $i\in I$ için $a_i\in K$ ve $g_i\in G$ olmak üzere, her elemanı

$$a_1g_1+a_2g_2+\cdots+a_ng_n$$

formunda olan, G nin elemanlarının sonlu lineer kombinasyonları ile K nın elemanlarını katsayı kabul edilmesinden oluşan $K[G]$ kümesini göz önüne alalım.

$K[G]$ nin herhangi elemanı genellikle

$$\sum_{g\in G}a_gg$$

biçiminde gösterilir. Aşağıdaki işlemlerle birlikte $K[G]$, K üzerinde bir cebirdir. Bu cebire *grup cebir* denir.

Toplama:

$$\sum_{g\in G}a_gg+\sum_{g\in G}b_gg=\sum_{g\in G}(a_g+b_g)g$$

Skalerle çarpma :

$$a\sum_{g\in G}a_gg=\sum_{g\in G}(aa_g)g$$

Çarpma:

$$\left(\sum_{g\in G} a_g g\right)\left(\sum_{h\in G} b_h h\right) = \sum_{g,h\in G} (a_g b_h) g * h$$

$|G| = n$ ve $|K| = m$ olmak üzere $|K[G]| = m^n$ dir.

Örnek 1.14 $G = C_3 = \langle g \rangle$ 3. mertebeden devirli grup olsun. $z_1, z_2, z_3 \in \mathbb{C}$ olmak üzere $\mathbb{C}[G]$ grup cebirinin herhangi bir elemanı

$$r = z_1 + z_2 g + z_3 g^2$$

şeklinde yazılır. $s \in \mathbb{C}[G]$ başka bir eleman olmak üzere

$$s = w_1 + w_2 g + w_3 g^2$$

elemanların toplamı

$$r + s = z_1 + w_1 + (z_2 + w_2)g + (z_3 + w_3)g^2$$

ve çarpımı

$$rs = z_1 w_1 + z_2 w_3 + z_3 w_2 + (z_1 w_2 + z_2 w_1 + z_3 w_3)g + (z_1 w_3 + z_3 w_1 + z_2 w_2)g^2$$

biçimindedir.

Örnek 1.15 $G = C_3 = \langle g \rangle$ 3. mertebeden devirli grup ve $K = \mathbb{Z}_2$ olmak üzere $K[G]$ grup cebirinin elemanları

$$\mathbb{Z}_2[C_3] = \left\{0, 1, g, g^2, 1+g, 1+g^2, g+g^2, 1+g+g^2\right\}$$

şeklindedir.

Önerme 1.5 a) K cisim ve G Abelyen ise $K[G]$ grup cebiri de değişmelidir.

b) H, G nin bir alt grubu ise $K[H]$ da $K[G]$ nin bir alt grubudur. Benzer şekilde S, K nin bir alt halkası ise $S[G]$ de $K[G]$ nin bir alt halkasıdır.

$f : G \to H$ herhangi bir grup homomorfizmi olmak üzere $K[f] : K[G] \to K[H]$ grup cebir homomorfizmi

$$\sum_{g\in G} a_g g \longmapsto \sum_{g\in G} a_g f(g)$$

şeklinde tanımlanabilir. $f' : H \to L$ başka bir grup homomorfizmi ise $K[ff'] = K[f]K[f']$ dir. Grup cebir tanımı ve grup cebir homomorfizm yardımıyla aşağıdaki önerme verilebilir.

Önerme 1.6 Herhangi bir grup alındığında, herzaman bir K-cebir

$$K[.]: \ \mathbf{Gr} \ \rightarrow \ \mathbf{K\text{-}Alg}$$

funktoru ile elde edilir.

$K[.]$ grup cebir funktorunun aksine herhangi bir cebirden bir grup elde edilebilir. Cebirdeki çarpma unutularak toplamsal abelyen grup elde edilir, bu da unutulabilir (forgetful)

$$\mathbf{K\text{-}Alg} \rightarrow \mathbf{Ab}$$

funktorunu verir. Ayrıca bilindiği üzere cebirdeki çarpmaya göre tersi bulunabilen elemanların oluşturduğu küme bir altgruptur. Bu gruba cebirin terslenebilen elemanları grubu denir, bu da

$$U(.): \ \mathbf{K\text{-}Alg} \ \rightarrow \mathbf{Gr}$$

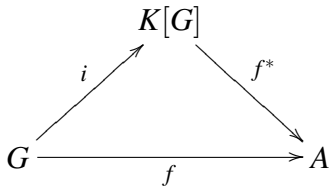
funktorunu verir. Genel olarak komutatif olmayan cebirlerin terslenebilen elemanları grubu abelyen olmak zorunda değildir. Buradan, ispatı (Barker, 2003) tarafından yapılan aşağıdaki önermeyi verebiliriz.

Önerme 1.7 $K[.]$: **Gr** $\rightarrow$ **K-Alg** grup cebir funktoru $U(.)$: **K-Alg** $\rightarrow$ **Gr** funktorunun sol ekidir. Böylece G bir grup ve A bir K-cebir olmak üzere

$$\mathbf{Gr}(G, U(A)) \quad \cong \quad \mathbf{K\text{-}Alg}(K[G], A)$$

izomorfizmi vardır.

Önerme 1.8 (Evrensellik Özelliği) G bir grup ve K bir cisim olsun. A, $K \subset A$ biçiminde herhangi bir halka ve $f : G \rightarrow A$ grup homomorfizmi olsun. Bu durumda $i : G \rightarrow K[G]$ içine dönüşüm olmak üzere



diyagramı değişmeli olacak şekilde (yani $f^* \circ i = f$) birtek

$$\begin{array}{rccc} f^*: & K[G] & \longrightarrow & A \\ & \sum\limits_{g \in G} a_g g & \longmapsto & \sum\limits_{g \in G} a_g f(g) \end{array}$$

homomorfizmi vardır.

$A = K[H]$ alınırsa $f^* : K[G] \rightarrow K[H]$ birtek grup cebir homomorfizmi vardır. Ayrıca;

ii f birebir ise f^* birebirdir.

iii f örten ise f^* da örtendir.

Tanım 1.9 $\varepsilon : K[G] \to K$

$$\varepsilon\left(\sum_{g\in G} a_g g\right) = \sum_{g\in G} a_g$$

homomorfizmine agümentasyon homomorfizmi denir. Bu homomorfizmin çekirdeğine ise agümentasyon ideali denir ve $\Delta(G)$ ile gösterilir.

Grup cebirlerin temel özelliklerinden birisi de sağ-sol simetri özelliğidir. Herhangi bir gruptaki her elemanın tersi varolduğundan herhangi $R[G]$ grup halka üzerinde aşağıdaki gibi homomorfizm tanımlanabilir.

Önerme 1.10 R değişmeli halka ve G bir grup olsun.

$$\left(\sum_{g\in G} a_g g\right) \longmapsto \sum_{g\in G} a_g g^{-1}$$

şeklinde tanımlanan $* : R[G] \to R[G]$ fonksiyonu aşağıdaki özellikleri sağlayan bir homomorfizmdir.

i $*(\alpha+\beta) = *(\alpha)+*(\beta)$,

ii $*(\alpha\beta) = *(\beta)*(\alpha)$,

iii $*(*(\alpha)) = \alpha$.

Her $g,h \in G$ için $(gh)^{-1} = h^{-1}g^{-1}$ olduğundan bu özelliklerin sağlandığı kolaylıkla gösterilebilir ve $*$ bir antiotomorfizmdir.

BÖLÜM 2

GRUP ALGORİTMA VE PROGRAMLAMA

2.1 Giriş

Grup Algorima ve Programlama (GAP) özellikle cebir ve sayılar teorisi için Pascal ve C programlama dilleri kullanılarak yazılmış bir sembolik hesaplama programıdır. Diğer programlama dilleri gibi değişkenlere, sabitlere, denetim deyimlerine, aritmetik operatörlere ve döngü deyimlerine sahip olan GAP programlama dili içerisinde, kullanılacak fonksiyonlar için ayrılmış özel kelimeler vardır. Programlama dili hakkında ayrıntılı bilgilere (GAP, 2008) den de bakılabilir. Bu tezde kullanılan derlemelerin bir bölümü (Odabaş, 2002) den alınmıştır.

GAP programı büyük harf küçük harf duyarlılığına sahiptir. Hemen hemen tüm anahtar kelimelerin küçük harfler kullanılarak yazıldığına dikkat edilmelidir. Örneğin `return` anahtar kelimesindeki harflerden herhangi birisi büyük harfle yazılırsa bu kelime anahtar kelime olmaktan çıkar. Anahtar kelimeler arasında boşluk kullanılmamalıdır.

Değişken isimleri tanımlanırken herhangi bir karakter uzunluğu sınırlaması olmamasına rağmen GAP programı için yalnızca ilk 1023 karakter anlamlıdır. GAP programı için ilk 1023 karaktere kadar aynı olan iki değişken isimleri 1023 ten sonraki karakterler farklı olsa dahi aynı değişkeni gösterecektir.

GAP programında herhangi bir komutta sağ taraftaki kısım deyim olarak adlandırılır. Deyimler değer yapılarını ifade etmekte kullanılırlar. Genel olarak deyimler, bir değer olmayıp bir veya birkaç tane değerin aritmetik operatörlerle veya fonksiyonlarla bir işlem gerçekleştirmesini sağlayan yapılardır. Örneğin 3+5 bir deyim, bu deyimin sonucu olan 8 bir değerdir. GAP programında # işareti bir açıklama yapmak için kullanılır.

GAP

```
gap> 3+5; # bilinen toplama islemi
8
gap> 3*5; # bilinen carpma islemi
15
```

Diğer programlama dillerinde olduğu gibi GAP programında da değişkenler, bir değere karşılık gelmesi için kullanılırlar. Her bir değişken ismine bir tanımlayıcı (identifier) denir.

GAP programında değişken tanımlama işlemi PASCAL programlama dilindeki gibidir. Bir tanımlayıcı (identifier) genel olarak `tanımlayıcı_ismi:=değer;` biçiminde oluşturulur. Bir ifade tanımlayıcı ismi olarak tanımlanmamışsa bu değeri deyim olarak kullanmak hata mesajına sebep olacaktır. Daha önce oluşturulmuş bir tanımlayıcı için yeni bir değer girildiğinde, tanımlayıcı artık yeni değeri kullanmaya başlayacaktır.

GAP
```
gap> x:=5;
5
gap> x;
5
gap> y;
Variable: 'y' must have a value
gap> x:=9;
9
gap> x;
9
```

GAP programında tanımlayıcı ismi olarak anahtar kelimeler kullanılamaz. Anahtar kelimelerin dışında GAP programı içerisinde kullanılmak istenilen tanımlayıcı isimleri, GAP programı kütüphanesinde bir fonksiyon olarak tanımlanmışsa bu isimler de kullanılamaz, GAP programı bu değerin salt okunur olduğunu belirtip hata verecektir.

2.2 GAP Programında Fonksiyon Derleme

GAP programında özel fonksiyonlar da oluşturulabilir. Fonksiyonlar oluşturulurken üç farklı yöntem kullanılabilir. Birinci yöntemde fonksiyon oluşturabilmek için -> simgesi kullanılır. Örnek olarak, yazılan sayıların karesini hesaplayan bir fonksiyon ;

GAP
```
gap> kuvvet4:=x -> x^2;
function( x ) ... end
gap>
```

biçiminde yazılabilir. Yukarıdaki komutlar yardımıyla yazılan fonksiyon aşağıdaki gibi kullanılır.

GAP
```
gap> kuvvet4(2);
4
```

```
gap> kuvvet4(4);
16
```

GAP programında fonksiyon oluşturmak için kullanılan ikinci yöntem ise `function` komutudur .

GAP

```
gap> fonk:=function(x)
> local y;
> y:=x^2+2*x+1;
> return y;
> end;
function( x ) ... end
```

Yukarıdaki kodlar $y = f(x) = x^2 + 2x + 1$ şeklindeki bir fonksiyonu hesaplamak için yazılmış kodlardır. Buradaki `function(x)` komutu yazılacak fonksiyonun temel parametresinin x olacağını göstermektedir. Fonksiyon tanımlanırken kullanılacak diğer değişkenler yani yerel değişkenler de `local` komutundan sonra belirtilmelidir. `return` komutu ise fonksiyon içerisinde tanımlanan işlemler gerçekleştirildikten sonra geriye bir değer gönderilmesini sağlar.

GAP

```
gap> fonk(10);
121
gap> fonk(1);
4
gap> fonk(2);
9
```

GAP programı kullanırken bir kez yazılmış olan bir fonksiyon, daha sonra tekrar kullanılabilmektedir. Bu işlemi gerçekleştirmek için yazılan fonksiyonu bir şekilde kaydetmek gerekir. `LogTo` komutunda GAP programı kullanırken yapılan işlemlerin bir kaydının tutulduğundan daha önce bahsedilmişti, ancak bu komut yazılan fonksiyonu tekrar çağırmak için yeterli değildir. `InputLogTo` komutu oluşturulan fonksiyonların tekrar yüklenmesine olanak sağlar.

GAP

```
gap> InputLogTo("fonk");
gap> fonk:=function(x)
> local y;
```

```
> y:=x^2+2*x+1;
> return y;
> end;
function( x ) ... end
gap> InputLogTo();
```

Yukarıdaki `InputLogTo("fonk");` komutu kendisinden sonraki komutların "fonk" adlı bir dosya içerisinde saklanılması sağlar. Bu dosya GAP programının kurulduğu dizinin içerisine oluşturulacaktır. GAP programından çıkıp oluşturulan "fonk" adlı dosya bir metin editörü ile açılırsa bu dosyanın içeriği aşağıdaki gibi olacaktır.

GAP

```
fonk:=function(x)
local y;
y:=x^2+2*x+1;
return y;
end;
InputLogTo();
```

Bu dosyada `InputLogTo();` satırı silinir ve dosya tekrar kaydedilirse daha sonra bu yazılan fonksiyon ne zaman kullanılmak istenirse `Read("DosyaAdı");` komutu kullanılarak dosya GAP programına okutulabilir .

GAP

```
gap> Read("fonk");
gap> fonk(1);
4
gap> fonk(3);
16
gap> fonk(1234567);
1524158146623
```

GAP programında fonksiyon oluşturmanın üçüncü ve en kullanışlı yöntemi ise direkt olarak metin editörleri kullanarak `gi` dosyaları oluşturmaktır. `gi` dosyaları, içerisinde birden fazla fonksiyon barındırabilen ve doğrudan GAP programı tarafından okunabilen dosyalardır. `gi` dosyaları oluşturulduktan sonra GAP programı `Read("fonk.gi")` şeklindeki bir komutla karşılaştığında `fonk.gi` dosyasının içerisinde yer alan her fonksiyonun kullanılabilir duruma geldiğini anlar.

Aşağıda kodları verilen fonksiyon bir metin editörü içine yazılarak `kuvvet.gi` adıyla kaydedilebilir. Fonksiyon verilen bir x sayısı için x^2+2x+1 değerini hesaplayacaktır.

GAP
```
kuv:=function(x)
local y;
y:=x^2+2*x+1;
return y;
end;
```

Şimdi, var olan `kuvvet.gi` dosyasını `Read()` komutu kullanarak GAP programına okutup daha sonra bu dosya içerisindeki fonksiyon kullanılabilir.

GAP
```
gap> Read("kuvvet.gi");
gap> kuv(2);
9
gap> kuv(10);
121
```

`Read()` komutu bir dosyanın GAP programında okutulup çalışmaya hazır hale getirilmesi için kullanılır. `()` işaretleri arasına dosyanın adı ve varsa uzantısı yazılmalıdır. `ReadAsFunction()` komutu da yine GAP programına bir dosya okutmak için kullanılabilir. `ReadAsFunction()` komutunun, kullanılacak olan dosyadaki fonksiyon için bir fonksiyon adı verilmesine, `function` ve `end` komutlarının bulunmasına ihtiyacı yoktur. Bunun anlamı metin içersindeki ifadeler `local` satırı ile başlar.

Aşağıda verilen program kodları bir metin editörü içinde oluşturulabilir ve bu dosya `carp.gi` adıyla kaydedilebilir.

GAP
```
local y;
y:=12;
return y*100;
```

Oluşturulmuş olan `carp.gi` dosyası `ReadAsFunction` komutu kullanılarak GAP programına okutulabilir ve aynı anda fonksiyon olarak kullanılabilir.

GAP
```
gap> ReadAsFunction("carp.gi")();
1200
```

Aşağıda kodları verilen fonksiyon bir metin editörü içinde oluşturulabilir ve bu dosya `tekcift.gi` adıyla kaydedilebilir. Yazılan fonksiyon bir sayının tek mi yoksa çift mi olduğunu inceleyecektir.

GAP

```
Tekcift:=function(x)
local k;
k:=(-1)^x;
if x=0 then
Print(x," sayisi tek veya cift ozellige sahip degildir. \n");
elif k=1 then
Print(x," sayisi cift sayidir. \n");
else
Print(x," sayisi tek sayidir. \n");
fi;
end;
```

Oluşturulmuş olan `tekcift.gi` dosyası `Read()` komutu kullanılarak GAP programına okutulabilir ve daha sonra bu dosya içerisindeki `Tekcift` isimli fonksiyon kullanılabilir.

GAP

```
gap> Read("tekcift.gi");
gap> Tekcift(46);
46 sayisi cift sayidir.
gap> Tekcift(23);
23 sayisi tek sayidir.
gap> Tekcift(0);
0 sayisi tek veya cift ozellige sahip degildir.
```

GAP programı kullanarak yarıçapı verilen bir dairenin alanını hesaplayan bir fonksiyon oluşturulabilir. Metin editörü kullanarak `daire.gi` adı altında bir dosya oluşturulabilir. Yazdılan fonksiyon πr^2 değerini hesaplayıp ekrana yazdıracaktır.

GAP

```
daire:=function(r)
local pi,alan;
pi:=22/7;
alan:=pi*r^2;
return alan;
end;
```

Yukarıdaki kodlar yazılarak oluşturulan `daire.gi` dosya GAP programına okutulabilir ve içerisinde yer alan `daire` isimli fonksiyonu kullanılarak herhangi bir r değeri için daire alanı hesaplanabilir.

GAP

```
gap> Read("daire.gi");
gap> daire(2);
88/7
gap> daire(112);
39424
```

GAP programı kullanarak yarıçapı verilen bir silindirin alanını hesaplayan fonksiyon yazılabilir. Metin editörü kullanarak `silindir.gi` adı altında bir dosya oluşturulabilir. Yazılan fonksiyon $2\pi r(r+h)$ değerini hesaplayıp ekrana yazdıracaktır.

GAP

```
AlanSilindir := function(arg)
local narg, alan, dikkat, hata, pi, r, h;
narg:= Length(arg);
dikkat:= " Dikkat: yar\i cap ve yukseklik olmak uzere iki deger girilmelidir. \n";
hata:= " Hata: Yar\i cap ve yukseklik degerleri pozitif olmal\i d\i r. \n";
if ((narg < 2) or (narg >= 3)) then
Print(dikkat);
return false;
fi;
r:=arg[1];
h:=arg[2];
pi:=22/7;
alan:=(2*pi*r)*(r+h);
if (arg[1]<=0) or (arg[2]<=0) then
Print(hata);
return false;
fi;
return alan;
end;
```

Fonksiyon oluşturma esnasında kullanılan `narg` değişkeni, fonksiyon içerisinde listesi verilen elemanların uzunluğunu bulur. Oluşturulan fonksiyon için elemanların uzunluğu iki olmalıdır çünkü bir silindirin alanını bulmak için yarıçap ve yükseklik değerlerine ihtiyaç vardır. GAP programı kullanıcısı `AlanSilindir` fonksiyonunu kullanırken ikiden fazla veya eksik değer girerse yazılan fonksiyon uygun bir hata mesajı verecektir. Bir silindirin alanı hesaplanırken yarıçap ve yükseklik değerleri sıfır veya sıfırdan küçük bir sayı olamayacağından fonksiyon oluşturulurken girilen değerlerin pozitif sayı olup olmadığını denetleyen ve uygun mesajla ekrana yazdıran bir komut da hata değişkeni ile eklenebilir.

Yukarıdaki kodlar yazılarak oluşturulan `silindir.gi` adlı dosya GAP programına okutulabilir ve içerisinde yer alan `AlanSilindir` isimli fonksiyon kullanılarak herhangi bir r ve h

değerleri için silindirin alanı hesaplanabilir.

GAP

```
gap> Read("silindir.gi");
gap> AlanSilindir(13,40);
30316/7
gap> AlanSilindir(7,21);
1232
gap> AlanSilindir(3,6,6);
 Dikkat: yar\i cap ve yukseklik olmak uzere iki deger girilmelidir.
false
gap> AlanSilindir(3,-2);
 Hata: Yar\i cap ve yukseklik degerleri pozitif olmal\i d\i r.
false
```

Bir sayının pozitif tam bölenlerinin toplamı, sayının iki katı oluyorsa bu sayıya mükemmel sayı denilmektedir. Örneğin 6 sayısı mükemmel bir sayıdır çünkü $1+2+3+6=12=2\times 6$ dır. GAP programı kullanarak verilen bir sayının mükemmel bir sayı olup olmadığını hesaplayan bir fonksiyon oluşturulabilir. Metin editörü kullanılarak `mukemmel.gi` adı altında bir dosya oluşturulabilir.

GAP

```
Mukemmel:=function(n)
local mu,i,top,kat;
top:=0;
kat:=2*n;
for i in [1..n] do
mu:=(n mod i);
if (mu=0) then
top:=top+i;
fi;
od;
if (kat=top) then
Print(" ",n ," sayisi mukemmel sayidir.\n");
else
Print(" ",n ," sayisi mukemmel sayi degildir.\n");
fi;
end;
```

Yukarıdaki kodlar yazılarak oluşturulan `mukemmel.gi` dosyası GAP programına okutulabilir ve içerisinde yer alan `Mukemmel` isimli fonksiyon kullanılarak herhangi bir n sayısının mükemmel sayı olup olmadığı incelenebilir.

GAP

```
gap> Read("mukemmel.gi");
gap> Mukemmel(6);
```

```
 6 sayisi mukemmel sayidir.
gap> Mukemmel(12);
 12 sayisi mukemmel sayi degildir.
gap> Mukemmel(10001224);
 10001224 sayisi mukemmel sayi degildir.
```

Birden büyük olmak üzere kendisinden ve birden başka tam böleni olmayan sayılara asal sayılar denir. GAP programı kullanılarak verilen bir n sayısından küçük olan kaç tane asal sayı olduğunu hesaplayan bir fonksiyon oluşturulabilir. (Örnek olarak 10 sayısından küçük 4 tane asal sayı olup bunlar 2,3,5 ve 7 dir) Metin editörü kullanarak `kacasal.gi` adı altında bir dosya oluşturulabilir.

GAP

```
Kacasal:=function(n)
local i,j,kactane,asalmi;
kactane:=0;
i:=n-1;
while i>1 do
asalmi:=1;
j:=2;
while j<i do
if (i mod j=0) then
asalmi:=0;
fi;
j:=j+1;
od;
if (asalmi=1) then
kactane:=kactane+1;
fi;
i:=i-1;
od;
return kactane;
end;
```

Yukarıdaki kodlar yazılarak oluşturulan `kacasal.gi` dosyası GAP programına okutulabilir ve içerisinde yer alan `Kacasal` isimli fonksiyon kullanılarak herhangi bir n sayısından küçük kaç tane asal sayı olduğu hesaplanabilir.

GAP

```
gap> Read("kacasal.gi");
gap> Kacasal(2);
0
gap> Kacasal(3);
1
gap> Kacasal(10);
```

```
4
gap> Kacasal(100);
```

BÖLÜM 3

GRUP CEBİRLERDE HESAPLAMALAR

3.1 Giriş

Grup cebir cebirsel yapısı bilgisayar ortamına Victor Bovdi, Alexander Konovalov, Richard Rossmanith, Csaba Schneider tarafından 2009 da yazılmış olan GAP, LAGUNA ortak paketi ile aktarılmıştır. Herhangi bir R halkası ve G grubu yardımıyla bir sol R-modül oluşturulabilir. Buradaki R halkası yerine F cismi alınırsa oluşan yapı bir cisim olacaktır. Örneğin 32. mertebeden Dihedral grup, 2. mertebeden Galois cismi ve $\mathbb{Z}_4$ halkası kullanılarak $K[G]$ ve $R[G]$ grup halkaları oluşturulabilir.

GAP

```
gap> G:=DihedralGroup(32);
<pc group of size 32 with 5 generators>
gap> K:=GaloisField(2);
GF(2)
gap> KG:=GroupRing(K,G);
<algebra-with-one over GF(2), with 5 generators>
gap> R:=Integers mod 4;
(Integers mod 4)
gap> RG:=GroupRing(R,G);
<free left module over (Integers mod 4), and ring-with-one, with 5 generators>
```

Herhangi bir grup halkası verildiğinde, bu halkayı oluşturan grup ve halkayı bulmak için `LeftActionDomain` ve `UnderlyingGroup` denetim deyimleri kullanılabilir.

GAP

```
gap> UnderlyingGroup(KG);
<pc group of size 32 with 5 generators>
gap> LeftActingDomain(KG);
GF(2)
gap> UnderlyingRing(RG);
(Integers mod 4)
gap> UnderlyingField(KG);
GF(2)
```

`GroupRing` denetim deyimi ile oluştuşturulmuş cebirsel yapının bir cebir olup olmadığını denetlemek için `IsGroupAlgebra` denetim deyimi kullanılır. Yapı bir cebir ise GAP programı `true` yanıtı verir.

GAP

```
gap> IsGroupAlgebra(KG);
true
gap> IsAlgebra(KG);
true
gap> IsGroupAlgebra(RG);
false
gap> IsLeftModule(RG);
true
```

Tanım 3.1 Her $a \in R$ için $a \in aRa^2R$ özelliğini sağlayan R halkasına $s-$zayıf düzenli ($s-$weakly regular) halka denir. s-zayıf düzenli grup cebirler ise Connel (1969) ve Gupta (1984) tarafından incelenmiştir.

Örnek 3.1 Karakterisitiği iki olan $Z_2 = (0,1)$ cismi ve $G = \langle g : g^3 = 1 \rangle$ devirli grubunu gözönüne alalım. $Z_2G = \{0, 1, g, g^2, 1+g, 1+g^2, g+g^2, 1+g+g^2\}$ şeklindeki Z_2G grup cebiri $s-$zayıf düzenlidir.

Her grup cebiri $s-$zayıf düzenli olmak zorunda değildir.

Örnek 3.2 $G = \langle g : g^2 = 1 \rangle$ ve $Z_2 = (0,1)$ olsun. $1+g \in Z_2G$ olmasına rağmen $1+g \notin 1+gZ_2G$ $(1+g)^2Z_2G = \{0\}$ olduğundan Z_2G $s-$zayıf düzenli değildir.

Teorem 3.2 $G = \langle g : g^{2n} = 1 \rangle$ ve $Z_2 = (0,1)$ olsun bu durumda Z_2G grup cebiri $s-$zayıf düzenli değildir.

İspat. Z_2G nin $a = 1+g+\cdots+g^{2n-1}$ elemanını alırsak $a \notin aZ_2Ga^2G$ olduğundan Z_2G $s-$zayıf düzenli değildir. □

Teorem 3.3 G çift sayı mertebeneden elemana sahip bir grup ve $Z_2 = (0,1)$ olsun. Bu durumda Z_2G grup cebiri $s-$zayıf düzenli değildir.

İspat. n çift tam sayı olmak üzere $g^n = 1$ özelliğinde $g \in G$ var olsun. $a = 1+g+\cdots+g^{n-1} \in Z_2G$ ve $a \notin aZ_2Ga^2Z_2G$ elde edilir. □

Teorem 3.4 S_n $n.$ dereceden simetrik grup ve $Z_2 = (0,1)$ olsun. Z_2S_n grup cebiri $s-$zayıf düzenli değildir.

İspat. $\alpha = 1 + \begin{pmatrix} 1 & 2 & 3 & \cdots & i & \cdots & j & \cdots & n \\ 1 & 2 & 3 & \cdots & j & \cdots & i & \cdots & n \end{pmatrix} \in Z_2S_n$ alınırsa açıkça $\alpha \notin \alpha Z_2S_n\alpha^2Z_2S_n$ olduğundan Z_2S_n grup cebiri $s-$zayıf düzenli değildir. □

Teorem 3.5 G, p mertebeden elemana sahip bir grup ve $Z_p = (0,1,\ldots,p-1)$ biçiminde karakteristiği p olan bir halka olsun. Bu durumda Z_pG grup halkası $s-$zayıf düzenli değildir.

Teorem 3.6 R birimli bir halka olsun. Her $a \in R$ için $a^3 = a$ oluyor ise R halkası $s-$zayıf düzenlidir.

İspat. Açıkça her $a \in R$ için $a \in aRa^2R$ dir. ($a = a.1.a^2.1$) □

Teorem 3.7 R birimsiz bir halka olsun. Her $a \in R$ için $a^5 = a$ oluyorsa R halkası $s-$zayıf düzenlidir.

İspat. Her $a \in R$ için $a = a.a.a^2.a = a^5 \in aRa^2R$ olur. □

Teorem 3.8 R sonlu mertebenden birimsiz ve nilpotent elemana sahip değilse R halkası $s-$zayıf düzenlidir.

Teorem 3.9 R birimli ve sıfır bölensiz bir halka olsun. R halkası $s-$zayıf düzenlidir ancak ve ancak her $a \in R$ için $a^2 = 1$ veya $ba^2.c = 1$ olacak şekilde $b,c \in R$ elemanları vardır.

İspat. R sıfır bölen içermeyen birimli bir halka ve $s-$zayıf düzenli olsun. Bu durumda her $a \in R$ için $a^2 = 1$ veya $ba^2c = 1$ olduğunu göstermeliyiz. R $s-$zayıf düzenli olduğundan her $a \in R$ için $a \in aba^2c$ olacak şekilde $b,c \in R$ vardır. Eğer $b = c = 1$ ise $a = a^3$ elde edilir. Buradan $a(1-a^2) = 0$ olur R sıfır bölen içermediğinden $a^2 = 1$ elde edilir. Eğer $b \neq 1$, $c \neq 1$ ise $a = aba^2c$ olduğundan $a(1-ba^2c) = 0$ ve R sıfır bölen içermediğinden $1 = ba^2c$ elde edilir. Tersine her $a \in R$ için $1 = ba^2c$ veya $a^2 = 1$ ise R sıfır bölen içermediğinden $s-$zayıf düzenli olduğu açıktır. □

Grup cebirlerin s-zayıf düzenli olup olmadığını inceleyebilmek için herhangi bir GAP fonksiyonu bulunmamaktadır. Bu kapsamda bu tez çalışması içerisinde s-zayıf düzenli halkalar için bir GAP fonksiyonu Odabaş (2009) tarafından oluşturulmuş ve elde edilen sonuçlar yayınlanmıştır. Bu kodlama kullanıcı tarafından verilen herhangi bir grup cebirin s-zayıf düzenli olup olmadığını grup cebir için ispatlamış önermeleri bilgisayar ortamında kontrol ederek sonuç olarak vermektedir.

GAP

```
#############################################################
##
##  HasZeroDiv( <D> ) . . . . halkanin sifir bolenlik testi
##  Nil2( <R> ) . . . . nilpotenent eleman testi
```

```
##
HasZeroDiv:=function (R)
    local   elms, zero, i, k;
           elms := Enumerator( R );
        zero := Zero( R );
        for i  in [2..Length(elms)]  do
            for k  in [i..Length(elms)]  do
                if elms[i] * elms[k] = zero then
                    return true;
                fi;
            od;
        od;
        return false;
        end ;

Nil2:= function(R)
    local a,x,list,elm,elR;
    list:=[];
    a:=Size(R);
    for elm in R do
      if IsZero(elm^a) then
      Add(list,elm);;
      fi;
      if not Length(list)=1 then
      return false;
      fi;
    od;
    return true;
    end;

#####################################################################
##
#  SWeak( <R>,<G> ) . . . .  s-zayif duzenli halka testi
##
SWeak:= function (R,G)
    local RG,n,a,b,c,d,m,list,list2,s,x,y,i,j,k,one,elms;
RG:=GroupRing(R,G);;
One(RG);;
###
#1-- R=Z_2 ve G 3. mertebeden devirli ise RG s-weakly
###
Print("1. Durum Kontrol Ediliyor..... ");
  if (R=GF(2)) and (IsCyclic(G) and Size(G)=3) then
    Print("=> Eslesme 1. Durum \n");
    return true;
  fi;
Print("Eslesme yok ##  \n");
###
#2-- R=Z_2 ve G 2. mertebeden devirli ise RG s-weakly degil.
###
Print("2. Durum Kontrol Ediliyor..... ");
  if (R=GF(2)) and (IsCyclic(G) and Size(G)=2) then
```

```
    Print("=> Eslesme 2. Durum \n");
    return false;
  fi;
Print("Eslesme yok ##  \n");
###
#3-- R=Z_2 ve G n. mertebeden devirli (n cift) ise RG s-weakly degil.
###
Print("3. Durum Kontrol Ediliyor..... ");
  if (R=GF(2)) and (IsCyclic(G) and (Size(G) mod 2)=0) then
    Print("=> Eslesme 3. Durum \n");
    return false;
  fi;
Print("Eslesme yok ##  \n");
###
#4-- R=Z_2 ve G n. mertebeden herhangi bir elemena sahipse (
#    n cift) ise RG s-weakly degil.
###
Print("4. Durum Kontrol Ediliyor..... ");
  if (R=GF(2)) and (ForAny(G,i-> Order(i) mod 2=0)) then
    Print("=> Eslesme 4. Durum \n");
    return false;
  fi;
Print("Eslesme yok ##  \n");
###
#5-- R=Z_2 ve G=S_n simetrik grup ise ise RG s-weakly degil.
###
Print("5. Durum Kontrol Ediliyor..... ");
  if (R=GF(2)) and (IsSymmetricGroup(G)) then
    Print("=> Eslesme 5. Durum \n");
    return false;
  fi;
Print("Eslesme yok ##  \n");
###
#6-- R=Z_p (p asal) ve G p mertebeden elemana sahipse RG s-weakly degil.
###
Print("6. Durum Kontrol Ediliyor..... ");
  if (IsPrime(Size(R))) and
    (ForAny(G,i-> Order(i) mod Size(R)=0)) then
    Print("=> Eslesme 6. Durum \n");
    return false;
  fi;
Print("Eslesme yok ##  \n");
###
#7-- RG birimli halka ve her a elemani i\U{e7}in a^3=a ise RG s-weakly.
###
Print("7. Durum Kontrol Ediliyor..... ");
  if (HasIdentity(RG)) and (ForAll(RG,a-> a^3=a)) then
    Print("=> Eslesme 7. Durum \n");
    return true;
  fi;
Print("Eslesme yok ##  \n");
###
```

```
#8-- RG birimsiz halka ve her a elemani icin a^5=a ise RG s-weakly.
###
Print("8. Durum Kontrol Ediliyor..... ");
  if (HasIdentity(RG)=false) and (ForAll(RG,a-> a^5=a)) then
    Print("=> Eslesme 8. Durum \n");
    return true;
  fi;
Print("Eslesme yok ##  \n");
###
#9-- RG sonlu birimsiz halka ve nilpotenet elemani yok ise RG s-weakly.
###
Print("9. Durum Kontrol Ediliyor..... ");
  if (HasIdentity(RG)=false) and (Nil2(RG)) then
    Print("=> Eslesme 9. Durum \n");
    return true;
  fi;
Print("Eslesme yok ##  \n");
Print("## Eslesme yok ## -- \n");
end;

################
SWeak2:= function (arg)
    local RG,n,a,b,c,d,m,list,list2,s,x,y,i,j,k,one,elms,narg,R,G;
narg:= Length(arg);
if (narg = 1) then
RG:=arg[1];;
fi;
if (narg = 2) then
R:=arg[1];;
G:=arg[2];;
RG:=GroupRing(R,G);;
fi;
One(RG);;
###
#10-- RG birimli fakat sifir bolensiz halka RG s-weakly ancak ve ancak
##    her a icin a^2=1 yada b*a^2*c=1.
###
Print("10. Durum Kontrol Ediliyor..... ");
  if (HasIdentity(RG)=true) and (HasZeroDiv(RG)=false) then
        Print("=> Eslesme 10. Durum \n");
    one := One( RG );
   elms := Enumerator( RG );
        for i  in [2..Length(elms)]  do
        a:=elms[i];
        if ((a^2=one)=false) and
        ((ForAny(RG,b->(ForAny(RG,c->b*(a^2)*c=one))))=false)  then
        return false;
        fi;
        od;
  #(ForAll(RG,a-> (ForAny(RG,b->(ForAny(RG,c->b*(a^2)*c=one))))))
    return true;
  fi;
```

```
Print("Eslesme yok ##  \n");
###
#11-- RG birimsiz sifir bolensiz halka RG s-weakly ancak ve ancak
##    her a icin a=a*b*a^2*c.
###
Print("11. Durum Kontrol Ediliyor..... ");
  if (HasIdentity(RG)=false) and (HasZeroDiv(RG)=false) then
        Print("=> Eslesme 11. Durum \n");
    if ((ForAll(RG,a->
    (ForAny(RG,b->(ForAny(RG,c->a*b*(a^2)*c=a))))))=false)  then
    return false;
    fi;
    return true;
  fi;
Print("Eslesme yok ##  \n");
###
#12-- TANIM.
###
Print("Tanim Kontrol Ediliyor..... ");
    Print("=> Eslesme Tanim \n");
    if ((ForAll(RG,a->
(ForAny(RG,b->(ForAny(RG,c->a*b*(a^2)*c=a))))))=false)  then
    return false;
    fi;
    return true;
Print("Eslesme yok ##  \n");
    Print("Eslesme yok -- \n");
end;
```

31 dereceden küçük 92 grup ve $\mathbb{Z}_2$ halkasının oluşturduğu grup cebirlerin $s-$zayıf düzenli olup olmadığını yukarıdaki kodlar yardımıyla hesaplanarak GAP grup numaralarıyla birlikte aşağıdaki tabloda verilmiştir.

Halka Id	GAP#	Grup Id	Grup Cebir Mertebesi	$s-$Zayıf Düzenli
$\mathbb{Z}_2$	1/1	I	2	+
$\mathbb{Z}_2$	2/1	c2	4	-
$\mathbb{Z}_2$	3/1	c3	8	+
$\mathbb{Z}_2$	4/1	c4	16	-
$\mathbb{Z}_2$	4/2	k4	16	-
$\mathbb{Z}_2$	5/1	c5	32	+
$\mathbb{Z}_2$	6/1	s3	64	-
$\mathbb{Z}_2$	6/2	c6	64	-
$\mathbb{Z}_2$	7/1	c7	128	+
$\mathbb{Z}_2$	8/1	c8	256	-
$\mathbb{Z}_2$	8/2	c4c2	256	-
$\mathbb{Z}_2$	8/3	d8	256	-
$\mathbb{Z}_2$	8/4	q8	256	-
$\mathbb{Z}_2$	8/5	$c2^3$	256	-
$\mathbb{Z}_2$	9/1	c9	512	+
$\mathbb{Z}_2$	9/2	$c3^2$	512	+
$\mathbb{Z}_2$	10/1	d10	1024	-
$\mathbb{Z}_2$	10/2	c10	1024	-
$\mathbb{Z}_2$	11/1	c11	2048	+
$\mathbb{Z}_2$	12/1	q12	4096	-
$\mathbb{Z}_2$	12/2	c12	4096	-
$\mathbb{Z}_2$	12/3	a4	4096	-
$\mathbb{Z}_2$	12/4	d12	4096	-
$\mathbb{Z}_2$	12/5	c6c2	4096	-
$\mathbb{Z}_2$	13/1	c13	8192	+
$\mathbb{Z}_2$	14/1	d14	16384	-
$\mathbb{Z}_2$	14/2	c14	16384	-
$\mathbb{Z}_2$	15/1	c15	32768	+
$\mathbb{Z}_2$	16/1	c16	65536	-
$\mathbb{Z}_2$	16/2	$c4^2$	65536	-
$\mathbb{Z}_2$	16/3	c4c2$\rtimes$c2	65536	-
$\mathbb{Z}_2$	16/4	c4$\rtimes$c4	65536	-
$\mathbb{Z}_2$	16/5	c8c2	65536	-
$\mathbb{Z}_2$	16/6	c2$\rtimes$c8	65536	-
$\mathbb{Z}_2$	16/7	d16	65536	-
$\mathbb{Z}_2$	16/8	qd16	65536	-
$\mathbb{Z}_2$	16/9	q16	65536	-
$\mathbb{Z}_2$	16/10	c4k4	65536	-
$\mathbb{Z}_2$	16/11	d8c2	65536	-
$\mathbb{Z}_2$	16/12	q8c2	65536	-
$\mathbb{Z}_2$	16/13	d8y4	65536	-
$\mathbb{Z}_2$	16/14	$c2^4$	65536	-
$\mathbb{Z}_2$	17/1	c17	131072	+
$\mathbb{Z}_2$	18/1	d18	262144	-
$\mathbb{Z}_2$	18/2	c18	262144	-
$\mathbb{Z}_2$	18/3	c3s3	262144	-
$\mathbb{Z}_2$	18/4	$c3^2\rtimes$c2	262144	-
$\mathbb{Z}_2$	18/5	c6c3	262144	-
$\mathbb{Z}_2$	19/1	c19	524288	+

Halka Id	GAP#	Grup Id	Grup Cebir Mertebesi	$s-$Zayıf Düzenli
$\mathbb{Z}_2$	20/1	q20	1048576	-
$\mathbb{Z}_2$	20/2	c20	1048576	-
$\mathbb{Z}_2$	20/3	c4$\ltimes$c5	1048576	-
$\mathbb{Z}_2$	20/4	d20	1048576	-
$\mathbb{Z}_2$	20/5	c10c2	1048576	-
$\mathbb{Z}_2$	21/1	c7$\ltimes$c3	2097152	+
$\mathbb{Z}_2$	21/2	c21	2097152	+
$\mathbb{Z}_2$	22/1	d22	4194304	-
$\mathbb{Z}_2$	22/2	c22	4194304	-
$\mathbb{Z}_2$	23/1	c23	8388608	+
$\mathbb{Z}_2$	24/1	c3$\ltimes$c8	16777216	-
$\mathbb{Z}_2$	24/2	c24	16777216	-
$\mathbb{Z}_2$	24/3	sl(2,3)	16777216	-
$\mathbb{Z}_2$	24/4	q24	16777216	+
$\mathbb{Z}_2$	24/5	s3c4	16777216	+
$\mathbb{Z}_2$	24/6	d24	16777216	-
$\mathbb{Z}_2$	24/7	q12c2	16777216	-
$\mathbb{Z}_2$	24/8	d8$\ltimes$c3	16777216	-
$\mathbb{Z}_2$	24/9	c12c2	16777216	-
$\mathbb{Z}_2$	24/10	d8c3	16777216	-
$\mathbb{Z}_2$	24/11	q8c3	16777216	-
$\mathbb{Z}_2$	24/12	s4	16777216	-
$\mathbb{Z}_2$	24/13	a4c2	16777216	-
$\mathbb{Z}_2$	24/14	d12c2	16777216	-
$\mathbb{Z}_2$	24/15	c6k4	16777216	-
$\mathbb{Z}_2$	25/1	c25	33554432	+
$\mathbb{Z}_2$	25/2	$c5^2$	33554432	+
$\mathbb{Z}_2$	26/1	d26	67108864	-
$\mathbb{Z}_2$	26/2	c26	67108864	-
$\mathbb{Z}_2$	27/1	c27	134217728	+
$\mathbb{Z}_2$	27/2	c9c3	134217728	+
$\mathbb{Z}_2$	27/3	$c3^2\ltimes$c3	134217728	+
$\mathbb{Z}_2$	27/4	c9$\ltimes$c3	134217728	+
$\mathbb{Z}_2$	27/5	$c3^3$	134217728	+
$\mathbb{Z}_2$	28/1	q28	268435456	-
$\mathbb{Z}_2$	28/2	c28	268435456	-
$\mathbb{Z}_2$	28/3	d28	268435456	-
$\mathbb{Z}_2$	28/4	c14c2	268435456	-
$\mathbb{Z}_2$	29/1	c29	536870912	+
$\mathbb{Z}_2$	30/1	d6c5	1073741824	-
$\mathbb{Z}_2$	30/2	d10c3	1073741824	-
$\mathbb{Z}_2$	30/3	d30	1073741824	-
$\mathbb{Z}_2$	30/4	c30	1073741824	-

K cisminin karakteristiği, G nin bazı elemanlarının mertebesini bölüyor ise $K[G]$ grup cebiri *modüler* olarak adlandırılır. GAP programında `IsFModularGroupAlgebra` denetim deyimi grup cebirlerin modüler olup olmadığını kontrol eder.

GAP

```
gap> IsFModularGroupAlgebra(GroupRing(GF(3),SymmetricGroup(6)));
true
```

```
gap> IsFModularGroupAlgebra(GroupRing(GF(3),CyclicGroup(7)));
false
gap> Characteristic(GF(3));
3
gap> List(CyclicGroup(7),Order);
[ 1, 7, 7, 7, 7, 7, 7 ]
```

Bir grubun birimden farklı her elemanının mertebesi bir p asal sayısının kuvveti ise bu gruba p-grup denir. Bu özellikten dolayı p-gruplar nilpotent özellik gösterirler. K karakteristiği p olan cisim ve G de aynı p asal sayısı için p-grup ise $K[G]$ grup cebirine p-modüler denir. GAP programında `IsPModularGroupAlgebra` denetim deyimi bir $K[G]$ grup cebirinin p-modüler olup olmadığını denetler.

GAP

```
gap> List(G,Order);
[ 1, 2, 4, 4, 8, 8, 8, 8, 16, 16, 16, 16, 16, 16, 16, 16, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 2, 2, 2, 2 ]
gap> IsPGroup(G);
true
gap> IsNilpotent(G);
true
gap> PrimePGroup(G);
2
gap> Characteristic(K);
2
gap> IsPModularGroupAlgebra(KG);
true
gap> IsPModularGroupAlgebra( GroupRing( GF( 2 ), SymmetricGroup( 6 ) ) );
false
```

`Support` denetim deyimi, $\alpha_i \in R$ ve $g_i \in G$ olmak üzere $R[G]$ grup halkasının herhangi $x = \alpha_1 \cdot g_1 + \alpha_2 \cdot g_2 + \cdots + \alpha_k \cdot g_k$ elemanındaki $g_i \in G$ lerin listesini verir.

GAP

```
gap> KG:=GroupRing(GF(3),CyclicGroup(8));
<algebra-with-one over GF(3), with 3 generators>
gap> eL:=Elements(KG);;
gap> Size(KG);
6561
gap> x:=eL[6001];
(Z(3)^0)*f2+(Z(3)^0)*f1*f2+(Z(3)^0)*f1*f3+(Z(3)^0)*f2*f3+(Z(3))*f1*f2*f3
gap> Support(x);
[ f2, f1*f2, f1*f3, f2*f3, f1*f2*f3 ]
```

`Length` denetim deyimi, bir x elamanını oluşturan lineer toplamdaki elemanların sayısını verir. Açıkca bu sayı G nin eleman sayısını geçemez.

GAP
```
gap> Length(x);
5
```

`Augmentation` denetim deyimi $x = \alpha_1 \cdot g_1 + \alpha_2 \cdot g_2 + \cdots + \alpha_k \cdot g_k$ şeklindeki grup halka elemanının $\alpha_1 + \alpha_2 \cdots + \alpha_k$ katsayılarının toplamını verir.

GAP
```
gap> Augmentation(x);
0*Z(3)
```

$R[G]$ grup halkasının bir x elemanının tersinin olup olmadığını hesaplamak için `IsUnit` denetim deyimi, varolan tersi hesaplamak için `InverseOp` denetim deyimi kullanılır. $x\hat{}-1$ ifadesi de tersi hesaplamak için kullanılabilir.

GAP
```
gap> IsUnit(x);
false
gap> x^-1;
fail
gap> y:=eL[1001];
(Z(3)^0)*<identity> of ...+(Z(3))*f1+(Z(3))*f2+(Z(3)^0)*f3+(Z(3)^0)*f1*f2+(
Z(3))*f2*f3+(Z(3)^0)*f1*f2*f3
gap> IsUnit(y);
true
gap> y^-1;
(Z(3)^0)*<identity> of ...+(Z(3)^0)*f1+(Z(3))*f2+(Z(3)^0)*f3+(Z(3)^0)*f1*f3+(
Z(3))*f2*f3+(Z(3))*f1*f2*f3
```

Bu operasyon $R[G] \to R$ biçiminde, tanım 1.9 da verilen agümantasyon homomorfizmini oluşturmak için kullanılır. $R[G]$ nin herhangi bir elemanının bu homomorfizm altındaki görüntsünü bulmak için `Augmentation` denetim deyimi de kullanılabilir.

GAP
```
gap> F := GF( 2 ); G := SymmetricGroup( 3 ); FG := GroupRing( F, G );
GF(2)
Sym( [ 1 .. 3 ] )
gap> f:=AugmentationHomomorphism(KG);
[ (Z(3)^0)*f1, (Z(3)^0)*f2, (Z(3)^0)*f3 ] -> [ Z(3)^0, Z(3)^0, Z(3)^0 ]
```

```
gap> IsSurjective(f);
true
gap> Augmentation(x)=Image(f,x);
true
gap> Augmentation(x+y);
Z(3)
```

Tanım 1.9 da verilen agümentasyon ideali, agümentasyon homomorfizminin çekirdeğinden oluşur. Başka bir deyişle `Augmentation` denetim deyimi R nin sıfırını olan $R[G]$ nin elemanları kümesini verir.

GAP

```
gap> A:=AugmentationIdeal(KG);
<two-sided ideal in <algebra-with-one of dimension 8 over GF(3)>,
  (3 generators)>
gap> IsIdeal(KG,A);
true
gap> eA:=Elements(A);;
gap> Image(f,eA[8]);
0*Z(3)
gap> A=Kernel(f);
true
```

`Units` denetim deyimi bir grup cebirinin tersinir olan elemanlarının oluşturduğu $T(K[G])$ grubunu oluşturur. GAP programı bu grubu oluşturmak için

$$T(K[G]) = K^* \times V(K[G])$$

direk çarpımını kullanır.

GAP

```
gap> T := Units( KG );
#I  LAGUNA package: Computing the unit group ...
<group of size 32768 with 15 generators>
gap> GeneratorsOfGroup( U )[5];
(Z(2)^0)*f2+(Z(2)^0)*f3+(Z(2)^0)*f2*f3
gap> IsSubgroup(T,V);
true
gap> FH := GroupRing( GF(3), SmallGroup(27,3) );
<algebra-with-one over GF(3), with 3 generators>
gap> T := Units( FH );
#I  LAGUNA package: Computing the unit group ...
<group of size 5083731656658 with 27 generators>
gap> x := GeneratorsOfGroup( T )[1];
Tuple( [ Z(3), (Z(3)^0)*<identity> of ... ] )
gap> x in FH;
```

```
false
gap> x[1] * x[2] in FH;
true
```

3.2 Çaprazlanmış Modül

Çaprazlanmış modül kavramı, (Whitehead, 1949) tarafından tanımlanmıştır. Whitehead, özellikle relatif homotopi gruplarının cebirsel yapıları üzerine yaptığı çalışmasında çaprazlanmış modüllere yer vermiştir. O zamandan itibaren çaprazlanmış modül kavramı diğer alanlarda da önemli bir yer tutmuştur. Bu konuda yapılan önemli çalışmalardan bazıları (Brown, 1982,1984) , (Brown ve Higgins, 1981) ve (Brown ve Huebschmann, 1981) dir.

GAP programı kullanılarak gruplar üzerinde çaprazlanmış modüllerin bazı özellikleri (Brown ve Wensley, 1995,1996,2003) ile (Alp ve Wensley, 2000) çalışmalarında incelenmiştir. Asosyatif ve değişmeli cebirler üzerinde çaprazlanmış modül kavramı, farklı bir adla (Lichtenbaum, Schlessinger, 1967) ve (Gerstenhaber, 1966) çalışmalarında karşımıza çıkar. (Porter, 1986) çalışmasında değişmeli cebirler üzerinde çaprazlanmış modül kavramını tanımlamıştır. Bununla birlikte, (Arvasi ve Porter, 1996) çalışmalarında değişmeli cebirler için çaprazlanmış modüllerle ilgili birçok önemli sonuçlar elde etmişlerdir. Değişmel cebirler üzerinde çaprazlanmış modüllerin bilgisayar ortamına aktarılması grup cebirler ve GAP yardımıyla Odabaş (2009) tarafından yapılmıştır. Bu bölüde GAP ile oluşturulan temel çaprazlanmış modüllerin hesaplamalarından bahsedeceğiz.

k değişmeli bir halka, R, k-cebir ve S, bir R-cebir olsun. $\partial : S \to R$ homomorfizmi, her $r \in R$ ve $c, c' \in S$ için

$$\begin{array}{ll} \text{Cm1:} & \partial(r.c) = r(\partial c) \\ \text{Cm2:} & (\partial c) = cc' \end{array}$$

özelliklerini sağlayan ∂ ya çaprazlanmış R-modül denir ve (S, R, ∂) ile gösterilir. Yalnızca Cm1 şartını sağlayan cebirsel yapılara ön çaprazlanmış modül denir. Burada $r \in R$ nin $c \in S$ ye etkisi $r.c$ ile gösterilmiştir. $\partial : S \to R$ fonksiyonuna (S, R, ∂) fonksiyonuna çaprazlanmış modülünün boundary dönüşümü adı verilir.

GAP programında bir çaprazlanmış modül oluşturmak için `XModAlg()` fonksiyonu kullanılır. Çaprazlanmış modülü oluşturan temel yapı `XModAlgObj()` objesidir. Diğer tüm fonksiyonlar bu objeye bağlanmıştır. Oluşturulan objenin (ön)çaprazlanmış modül olup olmadığını

denetlemek için

```
IsPreXModAlg()
IsXModAlg()
```

derlemeleri kullanılır.

Elde edilen çaprazlanmış modüllerde hesaplamalar yapabilmek için yazılmış fonksiyonlar vardır bu fonksiyonlardan bazıları aşağıda verilmiştir.

```
Size()
=
ViewObj()
PrintObj()
Display()
Name()
XModAlgAction()
Boundary()
```

Örnek 3.3 R herhangi bir cebir ve I, R nin ideali verildiğinde her zaman $\partial : I \to R$ içine fonksiyonu bir çaprazlanmış modüldür. Örneğin R, GF_3 halkası ve C_2 devirli grubu kullanılarak oluşturulan grup cebiri ve I agümentasyon ideal alınırsa çaprazlanmış modül aşağıdaki şekilde oluşturulabilir.

GAP

```
gap> R:=GroupRing(GF(3),CyclicGroup(2));
<algebra-with-one over GF(3), with 1 generators>
gap> I:=AugmentationIdeal(R);
<two-sided ideal in <algebra-with-one over GF(3), with 1 generators>,
(1 generators)>
gap> IsAlgebra(R);
true
gap> IsIdeal(R,I);
true
gap> CM1:=XModAlgByIdeal(R,I);
[Algebra( GF(3), [ (Z(3))*<identity> of ...+(Z(3)^0)*f1
] )->AlgebraWithOne( GF(3), [ (Z(3)^0)*f1 ] )]
gap> IsPreXModAlg(CM1);
true
gap> IsXModAlg(CM1);
true
```

```
gap> Size(CM1);
[ 3, 9 ]
gap> Name(CM);
"[..->..]"
gap> Display(CM1);
Crossed module [..->..] :-
: Source algebra has generators:
[ (Z(3))*<identity> of ...+(Z(3)^0)*f1 ]
: Range algebra has generators:
[ (Z(3)^0)*<identity> of ..., (Z(3)^0)*f1 ]
: Boundary homomorphism maps source generators to:
[ (Z(3))*<identity> of ...+(Z(3)^0)*f1 ]
```

Yukarıda verilen örneğin terside doğrudur. Diğer bir değişle $\partial : S \to R$ herhangi bir çaprazlanmış modül verildiğinde $\partial(S) = J$, R nin bir idealidir.

GAP

```
gap> f:=Boundary(CM1);
MappingByFunction(
<two-sided ideal in <algebra-with-one of dimension 2 over GF(3)>,
(dimension 1 )>, <algebra-with-one of dimension 2 over GF(3)>,
function( i ) ... end )
gap> J:=Image(f);
<algebra over GF(3), with 1 generators>
gap> IsIdeal(R,J);
true
```

Örnek 3.4 M , R-modülü verildiğinde $0 : M \to R$ sıfır fonksiyonu bir çaprazlanmış modüldür. (Porter, 1986) Örneğin M, GF_2 halkası ve $G = \langle (1,2,3,4,5) \rangle$ grubu kullanılarak oluşturulan modül olmak çaprazlanmış modül aşağıdaki şekilde oluşturulabilir.

GAP

```
gap> A:=GF(2);
GF(2)
gap> B:=Group((1,3,4,5));
Group([ (1,3,4,5) ])
gap> R:=GroupRing(A,B);
<algebra-with-one over GF(2), with 1 generators>
gap> CM2:=XModAlgByModule(R,A);
[AlgebraWithOne( GF(2), ... )->GF(2)]
gap> IsPreXModAlg(CM2);
true
gap> IsXModAlg(CM2);
true
gap> Size(CM2);
[ 16, 2 ]
gap> Source(CM2);
<algebra-with-one of dimension 4 over GF(2)>
```

```
gap> SetName(Source(CM2),"RG");
gap> Range(CM2);
GF(2)
gap> SetName(Range(CM2),"R");
gap> Name(CM2);
"[RG->R]"
gap> Display(CM2);
Crossed module [RG->R] :-
: Source algebra RG has generators:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,3,4,5) ]
: Range algebra R has generators:
 [ Z(2)^0, Z(2)^0 ]
: Boundary homomorphism maps source generators to:
 [ 0*Z(2), 0*Z(2) ]
gap> p:=XModAlgAction(CM2);
MappingByFunction(
[ [ 0*Z(2), <zero> of ... ], [ 0*Z(2), (Z(2)^0)*(1,5,4,3) ],
 [ 0*Z(2), (Z(2)^0)*(1,4)(3,5) ],
 [ 0*Z(2), (Z(2)^0)*(1,4)(3,5)+(Z(2)^0)*(1,5,4,3) ],
 [ 0*Z(2), (Z(2)^0)*(1,3,4,5) ],
 [ 0*Z(2), (Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,5,4,3) ],
 [ 0*Z(2), (Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,4)(3,5) ],
 [ 0*Z(2), (Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,4)(3,5)+(Z(2)^0)*(1,5,4,3) ],
 [ 0*Z(2), (Z(2)^0)*() ], [ 0*Z(2), (Z(2)^0)*()+(Z(2)^0)*(1,5,4,3) ],
 [ 0*Z(2), (Z(2)^0)*()+(Z(2)^0)*(1,4)(3,5) ],
 [ 0*Z(2), (Z(2)^0)*()+(Z(2)^0)*(1,4)(3,5)+(Z(2)^0)*(1,5,4,3) ],
 [ 0*Z(2), (Z(2)^0)*()+(Z(2)^0)*(1,3,4,5) ],
 [ 0*Z(2), (Z(2)^0)*()+(Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,5,4,3) ],
 [ 0*Z(2), (Z(2)^0)*()+(Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,4)(3,5) ],
 [ 0*Z(2), (Z(2)^0)*()+(Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,4)(3,5)+(Z(2)^0)*
 (1,5,4,3) ], [ Z(2)^0, <zero> of ... ],
 [ Z(2)^0, (Z(2)^0)*(1,5,4,3) ], [ Z(2)^0, (Z(2)^0)*(1,4)(3,5) ],
 [ Z(2)^0, (Z(2)^0)*(1,4)(3,5)+(Z(2)^0)*(1,5,4,3) ],
 [ Z(2)^0, (Z(2)^0)*(1,3,4,5) ],
 [ Z(2)^0, (Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,5,4,3) ],
 [ Z(2)^0, (Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,4)(3,5) ],
 [ Z(2)^0, (Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,4)(3,5)+(Z(2)^0)*(1,5,4,3) ],
 [ Z(2)^0, (Z(2)^0)*() ], [ Z(2)^0, (Z(2)^0)*()+(Z(2)^0)*(1,5,4,3) ],
 [ Z(2)^0, (Z(2)^0)*()+(Z(2)^0)*(1,4)(3,5) ],
 [ Z(2)^0, (Z(2)^0)*()+(Z(2)^0)*(1,4)(3,5)+(Z(2)^0)*(1,5,4,3) ],
 [ Z(2)^0, (Z(2)^0)*()+(Z(2)^0)*(1,3,4,5) ],
 [ Z(2)^0, (Z(2)^0)*()+(Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,5,4,3) ],
 [ Z(2)^0, (Z(2)^0)*()+(Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,4)(3,5) ],
 [ Z(2)^0, (Z(2)^0)*()+(Z(2)^0)*(1,3,4,5)+(Z(2)^0)*(1,4)(3,5)+(Z(2)^0)*
 (1,5,4,3) ] ], RG, function( x ) ... end )
```

Yukarıda ifadenin terside doğrudur. Yani $\partial : S \to R$ herhangi bir çaprazlanmış modül ve $I = \partial(S)$ verildiğinde *Çek* ∂, R/I-modüldür.

GAP

```
gap> f2:=Boundary(CM2);
```

```
MappingByFunction( RG, R, function( r ) ... end )
gap> L:=Kernel(f2);
<vector space over R, with 16 generators>
gap> IsLeftModule(L);
true
```

Örnek 3.5 $M(R)$, çarpım cebiri ve

$$\begin{array}{rccc} \delta_r: & R & \to & R \\ & x & \mapsto & \delta_r(x) = rx \end{array}$$

olmak üzere,

$$\begin{array}{rccc} \mu: & R & \to & M(R) \\ & r & \mapsto & \delta_r \end{array}$$

homomorfizması

$$\begin{array}{ccc} M(R) \times R & \to & R \\ (\delta, r) & \mapsto & \delta(r) \end{array}$$

şeklinde tanımlı etki ile birlikte bir çaprazlanmış modüldür. (Arvasi ve Ege, 2003) Örneğin R, GF_3 halkası ve $G = D_2$ dihedral grubu kullanılarak oluşturulan grup cebir olmak üzere çaprazlanmış modül aşağıdaki şekilde oluşturulabilir.

GAP

```
gap> R:=GroupRing(GF(5),DihedralGroup(2));
<algebra-with-one over GF(5), with 1 generators>
gap> Size(R);
25
gap> MR:=MultipleAlgebra(R);;
gap> CM3:=XModAlgByMultipleAlgebra(R);
[AlgebraWithOne( GF(5), ... )-> Mul. Alg.(AlgebraWithOne( GF(5), ... ))]
gap> IsPreXModAlg(CM1);
true
gap> IsXModAlg(CM1);
true
gap> Size(CM3);
[ 25, 25 ]
```

(C,R,∂) bir çaprazlanmış R-modül olsun. C', C nin bir alt cebri ve

$$\partial' = \partial|_{C'} : C' \to R$$

∂ nın C' ye kısıtlanmışı olmak üzere, (C',R',∂') çaprazlanmış modül ise (C',R',∂') ye (C,R,∂) nin alt çaprazlanmış modülü denir.

Aşağıdaki derlemeler alt çaprazlanmış modüllerde hesaplamalar yapmak için kullanılır.

```
SubXModAlg ()
SubPreXModAlg()
IsSubXModAlg()
IsSubPreXModAlg()
AlgebraActionType()
```

Örnek 3.6 $M(R)$, çarpım cebiri çaprazlanmış modülü örneğinde R, GF_2 halkası ve $G = D_2$ dihedral grubu kullanılarak oluşturulan grup cebir olmak üzere çaprazlanmış modül aşağıdaki şekilde oluşturulabilir. Bu çaprazlanmış modülde R cebirinin bir elemanı yardımıyla oluşturulan idealden $M(R)$ ye tanımlanan kısıtlama ile alt çaprazlanmış modül tanımlanır.

GAP

```
gap> A:=GroupRing(GF(2),DihedralGroup(2));
<algebra-with-one over R, with 1 generators>
gap> XM1:=XModAlg(A);
[AlgebraWithOne( R, ... )-> Mul. Alg.(AlgebraWithOne( R, ... ))]
gap> AlgebraActionType(XModAlgAction(XM1));
"Type2"
gap> B:=Source(XM1);
<algebra-with-one of dimension 2 over R>
gap> eB:=Elements(B);
[ <zero> of ..., (Z(2)^0)*<identity> of ...,
(Z(2)^0)*<identity> of ...+(Z(2)^0)*f1, (Z(2)^0)*f1 ]
gap> I:=Ideal(B,[eB[3]]);
<two-sided ideal in <algebra-with-one of dimension 2 over R>,
(1 generators)>
gap> IsIdeal(B,I);
true
gap> SM1:=SubXModAlg(XM1,I);
[Algebra( R, [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f1 ] )
-> Mul. Alg.(Algebra( R, [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f1 ] ))]
gap> IsSubPreXModAlg(XM1,SM1);
true
gap> IsSubXModAlg(XM1,SM1);
true
```

Örnek 3.7 Bir ideal yardımıyla tanımlanan çaprazlanmış modül örneğinde R, GF_2 halkası ve $G = D_2$ dihedral grubu kullanılarak oluşturulan grup cebir olmak üzere R nin herhangi bir I ideali yardımıyla çaprazlanmış modül oluşturduk. I nın J gibi bir ideali yardımıyla oluşan yeni çaprazlanmış modül bir alt çaprazlanmış modüldür.

GAP

```
gap> XM2:=XModAlg(B,I);
[Algebra( GF(2),[ (Z(2)^0)*()+(Z(2)^0)*(1,2,3,4)+(Z(2)^0)*(1,3)(2,4)
+(Z(2)^0)*(1,4,3,2)] )->AlgebraWithOne( GF(2),
[ (Z(2)^0)*(1,2,3,4) ] )]
```

```
gap> Display(XM2);
Crossed module [..->..] :-
: Source algebra has generators:
[ (Z(2)^0)*()+(Z(2)^0)*(1,2,3,4)+(Z(2)^0)*(1,3)(2,4)
+(Z(2)^0)*(1,4,3,2) ]
: Range algebra has generators:
[ (Z(2)^0)*(), (Z(2)^0)*(1,2,3,4) ]
: Boundary homomorphism maps source generators to:
[ (Z(2)^0)*()+(Z(2)^0)*(1,2,3,4)+(Z(2)^0)*(1,3)(2,4)
+(Z(2)^0)*(1,4,3,2) ]
gap> AlgebraActionType(XModActionAlg(XM2));
"Type1"
gap> eI:=Elements(I);
[ <zero> of ..., (Z(2)^0)*()+(Z(2)^0)*(1,2,3,4)+(Z(2)^0)*(1,3)(2,4)
+(Z(2)^0)*(1,4,3,2) ]
gap> J:=Ideal(I,[eI[1]]);
<two-sided ideal in I, (1 generators)>
gap> I=J;
false
gap> SM2:=SubPreXModOfAlg(PM,J);
[Algebra( GF(2), [], <zero> of ... )->B]
gap> IsSubXMod(XM2,SM2);
true
gap> Display(SM2);
Crossed module [..->B] :-
: Source group has generators:
[ ]
: Range group B has generators:
[ (Z(2)^0)*(), (Z(2)^0)*(1,2,3,4) ]
: Boundary homomorphism maps source generators to:
[ ]
```

3.3 Cat^1-Cebirler

Cat^1-gruplar kavramı ilk olarak homotopi n-tipler için cebirsel bir model olarak (Loday, 1982) tarafından tanımlanmıştır. Daha sonra (Ellis, 1988) de cebir versiyonunu tanımlamıştır.

A ve R, bir k-cebir olsun.

$$A \underset{e}{\overset{s}{\underset{t}{\rightleftarrows}}} R$$

s ve t örten homomorfizm ve e içine homomorfizm olmak üzere

$$\begin{array}{ll} \text{Cat1:} & se = id_R = te \\ \text{Cat2:} & \text{Çek}t\text{Çek}s = \{0_A\} \end{array}$$

özelliklerini sağlayan $C = (e;t,s : A \to R)$ cebirsel sistemine Cat^1-cebir denir. Yalnızca

Cat1 şartını sağlayan cebirsel sisteme ön Cat^1-cebir denir. Buradaki s,t ve e sırasıyla başlangıç, bitiş ve gömme homomorfizmi olarak adlandırılırlar.

Cat^1-cebirler üzerinde hesaplama yapabilmek için (Odabaş, 2009) tarafından yazılan `Cat1Alg()` fonksiyonu kullanılır. Bu fonksiyon yardımıyla eldeki bilgileri veri haline getirecek olan `PreCat1AlgObj()` objesi oluşturulmuş ve diğer tüm fonksiyonlar bu objeye bağlanmıştır. Oluşturulan objenin (ön)Cat^1-cebir olup olmadığını denetlemek için aşağıdaki fonksiyonlar kullanılır.

```
IsPreCat1Alg()
IsCat1Alg()
```

Cat^1-cebirler üzerinde hesaplamalar yapmak için kulanılan GAP fonksiyonlarından bazılları aşağıdadır;

```
Size()
=
ViewObj()
PrintObj()
Display()
IsIdentityCat1Alg()
Name()
Boundary()
Kernel()
Source
Range
Head()
Tail()
RangeEmbedding()
```

Örnek 3.8 R herhangi bir cebir ve $t,h,e : R \to R$ birim dönüşümler olsun. $C = (e;t,h : R \to R)$ cebirsel yapısı bir Cat^1-cebir dir. Örneğin GF_4 sonlu cismi (Galois cismi) ve k_4 klein 4 grubu kullanılarak oluşturulan grup cebir ve birim dönüşümler ile birlikte aşağıdaki Cat^1-cebir elde edilir.

GAP

```
gap> H:=GF(4);
GF(2^2)
gap> k4:=Group((1,2),(3,4));
Group([ (1,2), (3,4) ])
gap> R:=GroupRing(H,k4);
<algebra-with-one over GF(2^2), with 2 generators>
gap> Size(R);
256
gap> gR:=GeneratorsOfAlgebra(R);
[ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(3,4) ]
gap> f:=AlgebraHomomorphismByImages(R,R,gR,gR);
[ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(3,4) ] ->
[ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(3,4) ]
gap> IsAlgebraHomomorphism(f);
true
gap> C:=Cat1Alg(f,f,f);
[AlgebraWithOne( GF(2^2), [ (Z(2)^0)*(1,2), (Z(2)^0)*(3,4)
 ] ) -> AlgebraWithOne( GF(2^2), [ (Z(2)^0)*(1,2),
(Z(2)^0)*(3,4) ] )]
gap> IsCat1Alg(C);
true
gap> Size(C);
[ 256, 256 ]
gap> Display(C);
Cat1-algebra [..=>..] :-
: source algebra has generators:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(3,4) ]
: range algebra has generators:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(3,4) ]
: tail homomorphism maps source generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(3,4) ]
: head homomorphism maps source generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(3,4) ]
: range embedding maps range generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(3,4) ]
: the kernel is trivial.
```

Örnek 3.9 : Aşağıdaki örnekte oluşan cebirsel yapı Cat^1-cebir aksiyonlarını sağlamadığı için Cat^1-cebir yapısı oluşmaz.

GAP

```
gap> G:=Group((1,2,3,4));
Group([ (1,2,3,4) ])
gap> K:=Subgroup(G,[(1,3)(2,4)]);
Group([ (1,3)(2,4) ])
gap> IsSubgroup(G,K);
true
gap> A:=GroupRing(GF(2),G);
<algebra-with-one over GF(2), with 1 generators>
gap> em:=Embedding(G,A);
```

```
<mapping: Group( [ (1,2,3,4) ] ) -> AlgebraWithOne( GF(2), ... ) >
gap> H:=ImagesSet(em,K);
<group with 1 generators>
gap> B:=Algebra(GF(2),GeneratorsOfGroup(H));
<algebra over GF(2), with 1 generators>
gap> gA:=GeneratorsOfAlgebra(A);
[ (Z(2)^0)*(), (Z(2)^0)*(1,2,3,4) ]
gap> gB:=GeneratorsOfAlgebra(B);
[ (Z(2)^0)*(1,3)(2,4) ]
gap> f:=AlgebraHomomorphismByImages(A,B,gA,[One(B),gB[1]]);
[ (Z(2)^0)*(), (Z(2)^0)*(1,2,3,4) ] -> [(Z(2)^0)*(),
(Z(2)^0)*(1,3)(2,4) ]
gap> e:=InclusionMappingAlgebra(A,B);
[ (Z(2)^0)*(1,3)(2,4) ] -> [ (Z(2)^0)*(1,3)(2,4) ]
gap> CC:=PreCat1(f,f,e);
te <> range identity
Error, not a pre-cat1-group called from
PreCat1Obj( tres, hres, eres ) called from
PreCat1ByTailHeadEmbedding( arg[1], arg[2], arg[3] ) called from
<function>( <arguments> ) called from read-eval-loop
```

Cat^1-cebir oluşturmak için kullanılan üç morfizmin gerekli şartları her zaman sağlanmaz, ancak sonlu mertebeden grupların bir listesi (Odabaş, 2009) tarafından oluşturulmuştur. GAP programı kullanıcıları bu listeyi kullanarak kolaylıkla (ön)Cat^1-cebirler elde edebilirler. Bunun için `Cat1AlgSelect()` fonksiyonu 4 temel parametreyle birlikte `Cat1Alg()` fonksiyonuna bağlı olrak çalışmaktadır.

`Cat1AlgSelect( <gf>, <size>, <gpnum>, <num> );`

`<gf>` : Galois cisminin mertebesi,

`<size>` : Kullanılacak grubun mertebesini,

`<gpnum>` : Aynı mertebeden farklı grupların listesindeki sırasını,

`<num>` : Verilen ilk üç değere göre oluşturulan Cat^1-cebirin sıra sayısıdır.

Bu fonksiyonunun kullanıma örnek aşağıdaki gibidir.

GAP

```
gap> C:=Cat1AlgSelect(4,6,2,2);
[GF(2^2)_c6 -> GF(2^2)_triv]
gap> Size(C);
[ 4096, 4 ]
```

```
gap> Display(C);
Pre-cat1-algebra [GF(2^2)_c6=>GF(2^2)_triv] :-
: source algebra has generators:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3)(4,5) ]
: range algebra has generators:
 [ (Z(2)^0)*(), (Z(2)^0)*() ]
: tail homomorphism maps source generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*() ]
: head homomorphism maps source generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*() ]
: range embedding maps range generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*() ]
: kernel has generators:
 [ (Z(2)^0)*()+(Z(2)^0)*(1,2,3)(4,5), (Z(2)^0)*()+(Z(2)^0)*(1,3,2),
 (Z(2)^0)*()+(Z(2)^0)*(4,5), (Z(2)^0)*()+(Z(2)^0)*(1,2,3),
 (Z(2)^0)*()+(Z(2)^0)*(1,3,2)(4,5) ]
: boundary homomorphism maps generators of kernel to:
 [ <zero> of ..., <zero> of ..., <zero> of ...,
<zero> of ..., <zero> of ... ]
: kernel embedding maps generators of kernel to:
 [ (Z(2)^0)*()+(Z(2)^0)*(1,2,3)(4,5), (Z(2)^0)*()+(Z(2)^0)*(1,3,2),
 (Z(2)^0)*()+(Z(2)^0)*(4,5), (Z(2)^0)*()+(Z(2)^0)*(1,2,3),
 (Z(2)^0)*()+(Z(2)^0)*(1,3,2)(4,5) ]
```

`Cat1AlgSelect()` fonksiyonu ek özellikleriyle birlikte geniç bir kullanım alanına sahiptir ve 4 parametrenin tamamı girilmeden de çalışmaktadır. Tek parametreli kullanımda diğer parametreler sıfır kabul edilip, verilen mertebeden Galois cismi için Cat^1-cebir oluşturmakta kullanılabilecek en büyük grup mertebeleri hakkında bilgi verilmektedir.

GAP

```
gap> Cat1AlgSelect(11);
There are groups having orders maximum 9 for GF(11) in the list.
fail
```

İki parametreli kullanımda diğer iki parametre sıfır kabul edilip, verilen mertebeden Galois cismi ve verilen mertebeden grup için Cat^1-cebir oluşturmakta kullanılacak kaç farklı grup olduğu hakkında bilgi verilmektedir.

GAP

```
gap> Cat1AlgSelect(11,8);
 0 is a invalid gpnum number.
 where 0 < gpnum <= 5 for gpsize 8.
Usage: Cat1Alg( GF(num), gpsize, gpnum, num );
[ "GF(11)_c8", "GF(11)_c4c2", "GF(11)_d8", "GF(11)_q8",
"GF(11)_c2^3" ]
```

Üç parametreli kullanımda son parametre sıfır kabul edilip, verilen mertebeden Galois cismi ve verilen gruptan elde edilin grup cismi için kaç farklı Cat^1-cebir yapısı oluşturulabileceği ve Cat^1-cebir yapılarının dönüşümleri hakkında bilgi verilmektedir. Kullanıcı son parametreye girebileceği değerler için bu şekilde seçim yapabilmektedir.

GAP

```
gap> Cat1AlgSelect(11,8,5);
There are 13 cat1-structures for the algebra GF(11)_c2^3.
  Range Alg,          Tail Genimg,          Head Genimg
|-----------------------------------------------------|
| GF(11)_c2^3,          identity map         identity map            |
| GF(11)_triv,         [ 1, 1, 1, 1 ],         [ 1, 1, 1, 1 ]         |
| GF(11)_c2,         [ 1, 1, 1, 2 ],         [ 1, 1, 1, 2 ]         |
| GF(11)_c2,         [ 1, 1, 2, 2 ],         [ 1, 1, 2, 2 ]         |
| GF(11)_c2,         [ 1, 2, 2, 2 ],         [ 1, 2, 2, 2 ]         |
| GF(11)_c2,         [ 1, 2, 1, 2 ],         [ 1, 2, 1, 2 ]         |
| GF(11)_c2,         [ 1, 2, 1, 2 ],         [ 1, 1, 1, 2 ]         |
| GF(11)_c2,         [ 1, 2, 2, 2 ],         [ 1, 1, 2, 2 ]         |
| GF(11)_c2,         [ 1, 2, 2, 2 ],         [ 1, 1, 1, 2 ]         |
| GF(11)_c2,         [ 1, 2, 2, 2 ],         [ 1, 2, 1, 2 ]         |
| GF(11)_k4,         [ 1, 1, 2, 3 ],         [ 1, 1, 2, 3 ]         |
| GF(11)_k4,         [ 1, 3, 2, 3 ],         [ 1, 3, 2, 3 ]         |
| GF(11)_k4,         [ 1, 3, 2, 3 ],         [ 1, 1, 2, 3 ]         |
|-----------------------------------------------------|
Usage:  Cat1Alg( GF(num), gpsize, gpnum, num );
Algebra has generators [ (Z(11)^0)*(), (Z(11)^0)*(1,2),
(Z(11)^0)*(3,4), (Z(11)^0)*(5,6) ]
13
```

Girilen dört parametreden yanlış aralıklarda girilen parametre için yukarıdaki gibi yardım ekranı tekrar görüntülenir. Aşağıdaki fonksiyonlar Cat^1-cebirler üzerindeki hesaplamalar için kullanılır.

```
Cat1Alg()
Is2dAlgObject()
PreCat1AlgByEndomorphisms()
PreCat1AlgByTailHeadEmbedding()
Cat1AlgSelect()
Cat1AlgOfXModAlg()
```

Tanım 3.10 $C = (e;t,s : A \to R)$ bir Cat^1-cebir olsun. A',A nın bir alt cebir ve R', R nin bir alt cebiri olmak üzere

$$t' : t|_{A'} : A' \to R', \;\; s' : t|_{A'} : A' \to R' \;\; \text{ve } e' : e|_{R'} : R' \to A'$$

kısıtlanmış cebir morfizmleri Cat1 ve Cat2 şartını sağlıyorsa $C' = (e';t',s' : A' \to R')$ cebirsel yapısına $C = (e;t,s : A \to R)$ nin alt Cat^1-cebiri denir. Yalnızca Cat1 şartı sağlanıyorsa ön alt Cat^1-cebir denir.

Alt Cat^1-cebirlerin hesaplamalarında aşağıdaki fonksiyonlar sıklıkla kullanılır.

```
SubCat1Alg()
SubPreCat1Alg()
IsSubCat1Alg()
IsPreSubCat1Alg()
```

Örnek 3.10 $GF_2[C_6]$, 2. mertebeden Galois cismi ve 6. mertebeden devirli grup yardımıyla üretilen grup cebir ve $GF_2[C_3]$, 2. mertebeden Galois cismi ve 3. mertebeden devirli grup yardımıyla üretilen grup cebirler olmak üzere; `Cat1AlgSelect()` fonksiyonu yardımıyla $C = (e;t,s : GF_2[C_6] \to GF_2[C_3])$ Cat^1-cebirini oluşturalım. $GF_2[C_6]$ ve $GF_2[C_3]$ cebirlerinin alt cebirleri yardımıyla aşağıdaki gibi bir alt Cat^1-cebir oluşur.

GAP

```
gap> C:=Cat1AlgSelect(2,6,2,4);
[GF(2)_c6 -> GF(2)_c3]
gap> Display(C);
Cat1-algebra [GF(2)_c6=>GF(2)_c3] :-
: source algebra has generators:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3)(4,5) ]
: range algebra has generators:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3) ]
: tail homomorphism maps source generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3) ]
: head homomorphism maps source generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3) ]
: range embedding maps range generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3) ]
: kernel has generators:
 [ (Z(2)^0)*()+(Z(2)^0)*(4,5), (Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,2,3)(4,5),
 (Z(2)^0)*(1,3,2)+(Z(2)^0)*(1,3,2)(4,5) ]
: boundary homomorphism maps generators of kernel to:
 [ <zero> of ..., <zero> of ..., <zero> of ... ]
: kernel embedding maps generators of kernel to:
 [ (Z(2)^0)*()+(Z(2)^0)*(4,5), (Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,2,3)(4,5),
 (Z(2)^0)*(1,3,2)+(Z(2)^0)*(1,3,2)(4,5) ]
gap> A:=Source(C);
GF(2)_c6
gap> B:=Range(C);
GF(2)_c3
gap> eA:=Elements(A);;
```

```
gap> eB:=Elements(B);;
gap> AA:=Subalgebra(A,[eA[1],eA[2],eA[3]]);
<algebra over GF(2), with 3 generators>
gap> A=AA;
false
gap> BB:=Subalgebra(B,[eB[1],eB[2]]);
<algebra over GF(2), with 2 generators>
gap> BB=B;
false
gap> CC:=SubCat1Alg(C,AA,BB);
[Algebra( GF(2), [ <zero> of ..., (Z(2)^0)*(), (Z(2)^0)*()
+(Z(2)^0)*(4,5) ] ) -> Algebra( GF(2),
[ <zero> of ..., (Z(2)^0)*() ] )]
gap> IsSubCat1Alg(C,CC);
true
gap> Display(CC);
Cat1-algebra [..=>..] :-
: source algebra has generators:
 [ <zero> of ..., (Z(2)^0)*(), (Z(2)^0)*()+(Z(2)^0)*(4,5) ]
: range algebra has generators:
 [ <zero> of ..., (Z(2)^0)*() ]
: tail homomorphism maps source generators to:
 [ <zero> of ..., (Z(2)^0)*(), <zero> of ... ]
: head homomorphism maps source generators to:
 [ <zero> of ..., (Z(2)^0)*(), <zero> of ... ]
: range embedding maps range generators to:
 [ <zero> of ..., (Z(2)^0)*() ]
: kernel has generators:
 [ <zero> of ..., (Z(2)^0)*()+(Z(2)^0)*(4,5) ]
: boundary homomorphism maps generators of kernel to:
 [ <zero> of ..., <zero> of ... ]
: kernel embedding maps generators of kernel to:
 [ <zero> of ..., (Z(2)^0)*()+(Z(2)^0)*(4,5) ]
```

Teorem 3.11 Cat^1-cebirler **Cat1Alg** kategorisi ile çaprazlanmış modüller **XMod** kategorisi doğal denk kategorilerdir. (Porter, 1987)

Yukarıdaki teorem ile birlikte bilgisayar ortamına aktarılan herhangi bir çaprazlanmış modülden bir Cat^1-cebir ve yine bilgisayar ortamına aktarılmış herhangi bir Cat^1-cebirden bir çaprazlanmış modül elde ederek oluşan kategoriksel denkliğin bilgisayar programı yazılmıştır.

Cat^1-cebirlerle cebirler üzerinde çaprazlanmış modüllerin denkliğini göstermek için aşağıdaki fonksiyonlar kullanılmaktadır.

```
PreCat1AlgByPreXModAlg()
PreXModAlgByPreCat1Alg()
```

`XModAlgByCat1Alg()`

`Cat1AlgByXModAlg()`

`KernelDenkt()`

`KernelDenkh()`

`SDproduct()`

Örnek 3.11 R herhangi bir cebir ve I ,R nin ideali verildiğinde her zaman $\partial : I \to R$ içine fonksiyonunun bir çaprazlanmış modül olduğunu biliyoruz. R, GF_3 halkası ve C_2 devirli grubu kullanılarak oluşturulan grup cebiri ve agümentasyon ideal alınırsa çaprazlanmış modül elde edilir. Bu çaprazlanmış modülden Cat^1-cebir aşağıdaki şekilde oluşturulabilir. Yeni oluşan Cat^1-cebirden tekrar ilk çaprazlanmış modül elde edilir.

GAP

```
gap> R:=GroupRing(GF(3),CyclicGroup(2));
<algebra-with-one over GF(3), with 1 generators>
gap> I:=AugmentationIdeal(R);
<two-sided ideal in <algebra-with-one over GF(3), with 1 generators>,
 (1 generators)>
gap> CM:=XModAlgByIdeal(R,I);
[Algebra( GF(3), [ (Z(3))*<identity> of ...+(Z(3)^0)*f1
 ] )->AlgebraWithOne( GF(3), [ (Z(3)^0)*f1 ] )]
gap> IsXModAlg(CM);
true
gap> C:=Cat1AlgByXModAlg(CM);
[AlgebraWithOne( GF(3), [ (Z(3)^0)*f1 ] ) IX Algebra( GF(3),
[ (Z(3))*<identity> of ...+(Z(3)^0)*f1 ] ) -> AlgebraWithOne( GF(3),
[ (Z(3)^0)*f1 ] )]
gap> IsCat1Alg(C);
true
gap> SM:=XModAlgByCat1Alg(C);
[..->..]
gap> SM=CM;
true
```

Oluşturulan tüm fonksiyonlar `Cat1Alg` ve `XModAlg` fonksiyonlarına bağlandığından bu iki fonksiyon kullanılarak ta işlemler yapılabilir. `Cat1AlgSelect` yerinede `Cat1Alg` fonksiyonu kullanılabilir.

GAP

```
gap> C:=Cat1Alg(4,6,1,1);
[GF(2^2)_s3 -> GF(2^2)_s3]
gap> IsCat1Alg(C);
true
gap> Display(C);
Cat1-algebra [GF(2^2)_s3=>GF(2^2)_s3] :-
```

```
: source algebra has generators:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: range algebra has generators:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: tail homomorphism maps source generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: head homomorphism maps source generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: range embedding maps range generators to:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: the kernel is trivial.
gap> CM:=XModAlg(C);
[Algebra( GF(2^2), [], <zero> of ... )->GF(2^2)_s3]
gap> IsXModAlg(CM);
true
gap> Display(CM);
Crossed module [..->GF(2^2)_s3] :-
: Source group has generators:
 [ ]
: Range group has generators:
 [ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: Boundary homomorphism maps source generators to:
 [ ]
gap> CC:=Cat1Alg(CM);
[GF(2^2)_s3=>GF(2^2)_s3]
gap> CC=C;
true
```

3.4 Lie Cebiri

F bir cisim ve L, F vektör uzayı olsun.

$$\begin{array}{ccc} L \times L & \longrightarrow & L \\ (x,y) & \longmapsto & [x,y] \end{array}$$

bilineer fonksiyonu

L1) Her $x \in L$ için $[x,x] = 0$

L2) Her $x,y,z \in L$ için $[x,[y,z]] + [y,[z,x]] + [z,[x,y]] = 0$

şartlarını sağlıyorsa L ye bir Lie cebir denir.

Lie cebiri hakkında ayrıntılı bilgi için (Erdmann ve Wildon, 2006) çalışmasına bakılabilir.

Genellikle $[x,y]$ Lie braketine x ve y nin komutatörü, ayrıca L2'ye de Jacobi özdeşliği denir.

$[\ ,\ ]$ Lie braket ikili işlemi,

$$\begin{aligned} 0 = [x+y, x+y] &= [x,x]+[x,y]+[y,x]+[y,y] \\ &= [x,y]+[y,x] \end{aligned}$$

ile bilineerdir. Ayrıca L1 durumu,

L1′) Her $x, y \in L$ için $[x,y] = -[y,x]$ şeklinde de ifade edilebilir. F cisminin karakteristiği 2 değil ise L1′ de $x = y$ alınarak L1 yerine L1′ kullanılır.

Örnek 3.12 M, A üzerinde bir cebir olsun. $[,] : M \times M \to M$ fonksiyonu

$$[x,y] = xy - yx$$

şeklinde tanımlansın. $[,]$ fonksiyonun iki lineer olduğunu gösterelim.

i)

$$[x,x] = xx - xx = 0$$

ii)

$$\begin{aligned} [x,[y,z]]+[y,[z,x]]+[z,[x,y]] &= [x,yz-zy]+[y,zx-xz]+[z,xy-yx] \\ &= x(yz-zy)-(yz-zy)x+(xy-yx)z \\ &\quad -(zx-xz)y+z(xy-yx)-(xy-yx)z \\ &= 0 \end{aligned}$$

olur. Böylece M, $[,]$ ile birlikte Lie cebiridir.

Bir Lie cebiri oluşturmak için LAGUNA ortak paketiyle oluşturulan `LieAlgebraByDomain` veya `LieAlgebra` denetim deyimi kullanılabilir.

GAP

```
gap> G:=SymmetricGroup(6);
Sym( [ 1 .. 6 ] )
gap> KG:=GroupRing(GF(8),G);
<algebra-with-one over GF(2^3), with 2 generators>
gap> L:=LieAlgebraByDomain(KG);
#I  LAGUNA package: Constructing Lie algebra ...
<Lie algebra over GF(2^3)>
```

Tanım 3.12 M bir Lie cebiri olsun. Her $x, y \in M$ için $[x,y] = 0$ oluyorsa M ye abelyen Lie cebiri denir. (Amoya, 1974)

A grup cebiri üzerinde tanımlı Lie cebirinin değişmeli olup olmadığını test etmek için `IsLieAbelyan` denetim deyimi kullanılır.

GAP
```
gap> G := SymmetricGroup( 3 ); FG := GroupRing( GF( 2 ), G);
Sym( [ 1 .. 3 ] )
<algebra-with-one over GF(2), with 2 generators>
gap> L := LieAlgebra( FG );
#I  LAGUNA package: Constructing Lie algebra ...
<Lie algebra over GF(2)>
gap> IsAbelian( G );
false
gap> IsAbelian( L );
true
gap> IsLieAbelian( L );
false
```

`IsLieAlgebraOfGroupRing` denetim deyimi L Lie cebirinin oluşturulduğu asosyetif cebirin bir grup halkası olup olmadığını denetler.

GAP
```
gap> IsLieAlgebraOfGroupRing( L );
true
```

`LieCentre` denetim deyimi bir $K[G]$ grup cebiri üzerinde tanımlanmış Lie cebirinin merkezini verir. Bir Lie cebirinin merkezi G grubunun merkezi ve K cismi tarafından üretilen grup, cebirin Lie cebiridir. L Lie cebirinin merkezi olan C bir ideal oluşturur.

$R[G]$ grup halkası tarafından oluşturulmuş L Lie cebirinden G grubunu elde etmek için `UnderlyingGroup` denetim deyimi kullanılır.

GAP
```
gap> F := GF( 2 ); G := SymmetricGroup( 3 ); FG := GroupRing( F, G );
GF(2)
Sym( [ 1 .. 3 ] )
<algebra-with-one over GF(2), with 2 generators>
gap> L := LieAlgebra( FG );
#I  LAGUNA package: Constructing Lie algebra ...
<Lie algebra over GF(2)>
gap> UnderlyingGroup( L );
Sym( [ 1 .. 3 ] )
gap> LeftActingDomain( L );
GF(2)
```

Örnek 3.13 V,F üzerine sonlu-boyutlu vektör uzayı olsun. V den V ye bütün lineer fonksiyonların kümesi $GL(V)$ olarak yazılır. Bu da yine F üzerine bir vektör uzayıdır. $[\ ,\ ]$ Lie braket

işlemi, her $x, y \in GL(V)$ için, $\circ$ işlemi fonksiyonların bileşke işlemi olmak üzere

$$[x, y] := x \circ y - y \circ x$$

şeklindedir. Böylelikle $GL(V)$ bir Lie cebirdir. Ayrıca $GL(V)$ ye *genel lineer cebir* denir.

Tanım 3.13 F üzerinde L_1 ve L_2 iki Lie cebiri olsun. $\varphi : L_1 \longrightarrow L_2$ lineer fonksiyonu her $x, y \in L_1$ için

$$\varphi([x, y]) = [\varphi(x), \varphi(y)]$$

ise φ ye bir homomorfizm denir. Burada eşitliğin sol tarafındaki L_1 in, sağ tarafındaki ise L_2 nin braket işlemidir. φ bire bir ve örten ise φ bir izomorfizmdir.

Örnek 3.14 L bir Lie cebiri olsun. $x, y \in L$ için

$$\begin{array}{rccl} \mathrm{Ad}: & L & \longrightarrow & \mathrm{GL}(L) \\ & x & \longmapsto & ((\mathrm{Ad})(x))(y) = [x, y] \end{array}$$

fonksiyonu bir Lie cebir homomorfizmidir. Bu homomorfizme Adjoint homomorfizmi denir. Ad nin çekirdeği L nin merkezidir.

`NaturalBijectionToLieAlgebra` denetim deyimi $f : A \to L$ şeklinde birebir örten fonksiyonu oluşturur. Bu fonksiyon, cebir izomorfizmi olmamasına rağmen bir vektör uzayı izomorfizmidir.

GAP

```
gap> F := GF( 2 ); G := SymmetricGroup( 3 ); FG := GroupRing( F, G );
GF(2)
Sym( [ 1 .. 3 ] )
<algebra-with-one over GF(2), with 2 generators>
gap> t := NaturalBijectionToLieAlgebra( FG );
MappingByFunction( <algebra-with-one over GF(2), with
2 generators>, <Lie algebra over GF(
2)>, <Operation "LieObject">, function( y ) ... end )
```

`NaturalBijectionToAssociativeAlgebra` denetim deyimi yukarıda tanımlanan f fonksiyonunun $f^{-1} : L \to A$ biçiminde ters fonksiyonu verir.

GAP

```
gap> G := SymmetricGroup(3); FG := GroupRing( GF( 2 ), G );
Sym( [ 1 .. 3 ] )
<algebra-with-one over GF(2), with 2 generators>
gap> L := LieAlgebra( FG );
#I  LAGUNA package: Constructing Lie algebra ...
```

```
<Lie algebra over GF(2)>
gap> s := NaturalBijectionToAssociativeAlgebra( L );
MappingByFunction( <Lie algebra over GF(2)>, <algebra-with-one over GF(
2), with 2 generators>, function( y ) ... end, <Operation "LieObject"> )
gap> InverseGeneralMapping( s ) = NaturalBijectionToLieAlgebra( FG );
true
```

3.5 Lie Cebirlerin Çaprazlanmış Modülleri

Kassel ve Loday (1982) çalışmasında Lie cebirler üzerinde çaprazlanmış modülleri tanımlamıştır. Bu bölümde Lie cebirleri üzerinde çaprazlanmış modüllerin genel özellikleri ve hesaplamaları verilmiştir. GAP programlama dili kullanılarak Lie cebirlerinin çaprazlanmış modülleri bilgisayar ortamına Aslan (2010) tarafından aktarılmıştır.

M ve S iki Lie $\mathbf{K}$ -cebirler olmak üzere M üzerinde S nin Lie etkisi, aşağıdaki aksiyomları sağlayan

$$\begin{array}{ccc} S \times M & \longrightarrow & M \\ (s,m) & \longmapsto & s \cdot m \end{array}$$

dönüşümdür. Her $k \in \mathbf{K}$, $m, m' \in M$ ve $s, s' \in S$ için

Tanım 3.14 i. $k(s \cdot m) = (ks) \cdot m = s \cdot (km)$

ii. $s \cdot (m + m') = s \cdot m + s \cdot m'$

iii. $(s + s') \cdot m = s \cdot m + s' \cdot m$

iv. $[s, s'] \cdot m = s(s' \cdot m) - s'(s \cdot m)$

v. $s \cdot [m, m'] = [s \cdot m, m'] + [m, s \cdot m']$

M ve S iki Lie $\mathbf{K}$ -cebir olsun.

$$\mu \ : \ M \longrightarrow S$$

bir Lie $\mathbf{K}$ -cebir morfizmi ve

$$\begin{array}{ccc} S \times M & \longrightarrow & M \\ (s,m) & \longmapsto & s \cdot m \end{array}$$

S nin M üzerine Lie etkisi ile birlikte her $m, m' \in M$ ve $s, s' \in S$ için

Tanım 3.15 ÇM1) $\mu(s\cdot m)=[s,\mu(m)]$

ÇM2) $\mu(m)\cdot m'=[m,m']$

şartları sağlanıyor ise (M,S,μ) üçlüsüne Lie çaprazlanmış modül denir. Sadece (ÇM1) aksiyomunu sağlayan (M,S,μ) üçlüsüne Lie ön çaprazlanmış modül denir. (ÇM2) özelliğine de Peiffer özdeşiği denir.

Örnek: R bir Lie cebir ve I,R nin bir ideali olsun.

$$\begin{array}{rccc}\partial: & I & \longrightarrow & R\\ & i & \longmapsto & i\end{array}$$

içine dönüşümünü ele alalım. R nin I üzerine etkisi

$$\begin{array}{ccc}R\times I & \longrightarrow & I\\ (r,i) & \longmapsto & ri=[r,i]\end{array}$$

şeklinde Lie çarpım işlemi olarak verilsin. Bu durumda çaprazlanmış modül aksiyomlarının sağlandığını gösterelim.

ÇM1) $\partial(r\cdot i)=\partial[r,i]=[r,i]=[r,\partial i]$

ÇM2) $\partial(r)\cdot r'=rr'=[r,r']$

olduğundan (I,R,∂) bir çaprazlanmış modül yapısı oluşturur.

Örnek 3.15 M herhangi bir S -bimodül olsun.

$$\begin{array}{ccc}M\times M & \longrightarrow & M\\ (m_1,m_2) & \longmapsto & [m_1,m_2]=0\end{array}$$

çarpımı tanımlanırsa, M bir Lie S -cebir yapısı oluşturur. Bu durumda

$$\begin{array}{rccc}0: & M & \longrightarrow & R\\ & x & \longmapsto & 0(x)=0\end{array}$$

şeklinde verilen sıfır morfizminin

$$\begin{array}{ccc}S\times M & \longrightarrow & M\\ (s,m) & \longmapsto & s\cdot m=sm\end{array}$$

etkisi ile bir çaprazlanmış modül yapısı oluşturduğunu gösterelim.

ÇM1) $0(r\cdot m)=0[r,m]=0=[r,0m]$

ÇM2) $\partial(m)\cdot m' = 0\cdot m' = 0 = [m,m']$

olduğundan $(M,S,0)$ bir çaprazlanmış modül yapısı oluşturur.

Lie cebirleri üzerindeki çaprazlanmış modülleri oluşturmak için `XModLieAlg` fonksiyonu kullanılır. perasyonuna bağlanmıştır. Bir Lie cebiri çaprazlanmış modül oluşturmak için eldeki bilgileri bir veri haline getirecek olan `XModAlgLieObj` objesi oluşturulmuş ve diğer tüm fonksiyonlar bu objeye bağlanmıştır.

Kaynak Kodu

```
##############################################################################
###
##M  XModLieAlgObj( <bdy>, <act> ) . .  make pre-crossed module of Lie Algebras
###
InstallMethod( XModLieAlgObj, "for homomorphism and action", true,
    [ IsAlgebraHomomorphism, IsLieAlgebraAction ], 0,
function( bdy, act )

    local  A,B,filter,fam,PM,AB,BB;
    fam := FamilyObj( [ bdy, act ] );
    filter := IsPreXModLieAlgObj;
    B:= Source(bdy);
    A:= Range(bdy);
    AB:=Source(act);
    BB:=Range(act);
    if not ( B = BB ) then
        return false;
    fi;
    PM := Objectify( NewType( fam, filter ), rec() );
    SetSource( PM, B );
    SetRange( PM, A );
    SetBoundary( PM, bdy );
    SetXModLieAlgAction( PM, act );
    SetIs2dLieAlgObject( PM, true );
    return PM;
end );
```

Oluşturulan objenin Lie (ön)çaprazlanmış modül olup olmadığını denetlemek için aşağıdaki fonksiyonlar yazılmıştır.

`IsPreXModLieAlg`

`IsXModLieAlg`

Aşağıdaki fonksiyonlar Lie cebirleri üzerindeki çaprazlanmış modüllerin hesaplamalarında sıklıkla kullanılırlar.

Source

Range

Kernel

Boundary

GAP

```
gap> KG:=GroupRing(GF(2),DihedralGroup(8));
<algebra-with-one over GF(2), with 3 generators>
gap> L:=LieAlgebra(KG);
#I  LAGUNA package: Constructing Lie algebra ...
<Lie algebra of dimension 8 over GF(2)>
gap> D:=LieDerivedSubalgebra(L);
<Lie algebra of dimension 3 over GF(2)>
gap> IsIdeal(L,D);
true
gap> LX:=XModLieAlg(L,D);
[Algebra( GF(2), [ LieObject( (Z(2)^0)*f1+(Z(2)^0)*f1*f3 ),
  LieObject( (Z(2)^0)*f2+(Z(2)^0)*f2*f3 ), LieObject( (Z(2)^0)*f1*f2+(Z(2)^
    0)*f1*f2*f3 ) ] )->Algebra( GF(2),
[ LieObject( (Z(2)^0)*<identity> of ... ), LieObject( (Z(2)^0)*f1 ),
  LieObject( (Z(2)^0)*f2 ), LieObject( (Z(2)^0)*f3 ),
  LieObject( (Z(2)^0)*f1*f2 ), LieObject( (Z(2)^0)*f1*f3 ),
  LieObject( (Z(2)^0)*f2*f3 ), LieObject( (Z(2)^0)*f1*f2*f3 ) ] )]
gap> IsXModLieAlg(LX);
true
gap> Display(LX);
Crossed module [..->..] :-
: Source lie algebra has generators:
  [ LieObject( (Z(2)^0)*f1+(Z(2)^0)*f1*f3 ),
  LieObject( (Z(2)^0)*f2+(Z(2)^0)*f2*f3 ), LieObject( (Z(2)^0)*f1*f2+(Z(2)^
    0)*f1*f2*f3 ) ]
: Range lie algebra has generators:
  [ LieObject( (Z(2)^0)*<identity> of ... ), LieObject( (Z(2)^0)*f1 ),
  LieObject( (Z(2)^0)*f2 ), LieObject( (Z(2)^0)*f3 ),
  LieObject( (Z(2)^0)*f1*f2 ), LieObject( (Z(2)^0)*f1*f3 ),
  LieObject( (Z(2)^0)*f2*f3 ), LieObject( (Z(2)^0)*f1*f2*f3 ) ]
: Boundary homomorphism maps source generators to:
  [ LieObject( (Z(2)^0)*f1+(Z(2)^0)*f1*f3 ),
  LieObject( (Z(2)^0)*f2+(Z(2)^0)*f2*f3 ), LieObject( (Z(2)^0)*f1*f2+(Z(2)^
    0)*f1*f2*f3 ) ]
gap> Size(LX);
[ 8, 256 ]
gap> Boundary(LX);
MappingByFunction( <Lie algebra of dimension 3 over GF(
2)>, <Lie algebra of dimension 8 over GF(2)>, function( i ) ... end )
```

Örnek 3.16 M bir Lie S -cebir ve

$$\pi_2 \ : \ M \ \longrightarrow \ S$$

ikinci izdüşüm fonksiyonu bir Lie S -cebir morfizmidir. S nin $S \ltimes M$ üzerine Lie etkisi $s' \in S$ ve $(s,m) \in S \ltimes M$ için

$$s'(m,s) = (s' \cdot m, [s', s])$$

şeklinde tanımlansın. Bu durumda

$$\begin{array}{rccc} \pi_2 : & M \times G & \longrightarrow & G \\ & (m,g) & \longmapsto & g \end{array}$$

ve

$$\begin{array}{ccc} G \times (M \times G) & \longrightarrow & M \times G \\ (g', (m,g)) & \longmapsto & g' \cdot (m,g) = (g' \cdot m, [g', g]) \end{array}$$

olmak üzere

ÇM1)

$$\begin{aligned} \pi_2(g' \cdot (m,g)) &= \pi_2(g' \cdot m, [g', g]) \\ &= [g', g] \\ &= [g', \pi_2((m,g))] \end{aligned}$$

olup $(S \ltimes M, S, \pi_2)$ bir ön çaprazlanmış modüldür.

Diğer taraftan

ÇM2)

$$\begin{aligned} \pi_2((m \cdot g')) \cdot (m,g) &= g' \cdot (m,g) \\ &= (g' \cdot m, [g', g]) \end{aligned} \quad \text{.....I}$$

ve

$$[(m', g'), (m,g)] = ([m', m], [g', g]) \text{.....II}$$

olduğundan I=II olması için

$$(g' \cdot m, [g', g]) = ([m', m], [g', g]) \Longrightarrow g' \cdot m = [m', m]$$

olması gerekirdi. Fakat bu mümkün olmadığından genellikle $(S \ltimes M, S, \pi_2)$ bir çaprazlanmış modül değildir.

3.6 Cat^1-Lie Cebirleri

Bu bölümde Lie cebirleri üzerinde tanımlanan cat^1-cebirlerin tez kapsamında yazılan hesaplamaları tanıtılmıştır. Bir $LC = (e;t,h : G \to P)$

cat^1 -Lie cebiri $t,h : G \to P$ ve $e : P \to G$ biçiminde üç Lie cebiri homomorfizminden oluşur. Bu homomofizmler aşağıdaki şartları sağlar.

Cat1LieAlg1: $teh = h$ ve $het = t$
Cat1LieAlg2: $[\text{Çek}t\ \text{Çek}h] = 0$

Yalnızca Cat1LieAlg1 şartını sağlayan cebirsel sisteme ön cat^1 -Lie cebiri denir. Cat^1 -cebir oluşturmak için kullanılan yöntemlerin geliştirilmesiyle yazılan aşağıdaki derlemeler cat^1 -Lie cebiri oluşturmak için kullanılırlar.

```
Cat1LieAlg(t,h,e)
PreCat1LieAlgByTailHeadEmbedding(t,h,e)
PreCat1LieAlgByEndomorphisms(t,h)
PreCat1LieAlgObj(LC)
PreCat1LieAlg(LC)
IsIdentityCat1LieAlg(LC)
IsCat1LieAlg(LC)
IsPreCat1LieAlg(LC)
```

Cat1LieAlg fonksiyonu kullanıcının girdiği verilere göre hangi operasyonun kullanılacağını seçer. PreCat1LieAlgObj operasyonu cat^1 -Lie cebir yapısının aşağıda verilen niteliklerini veri olarak saklamak için kullanılır.

```
Source(LC)
Range(LC)
Tail(LC)
Head(LC)
Kernel(LC)
Boundary(LC)
```

Cat1-Lie cebir oluşturmak için kullanılan üç morfizm gerekli şartları her zaman sağlamadığından cat^1 -cebir yapısına benzer olarak Cat1LieAlgSelect() operasyonu

tez kapsamında oluşturulmuş ve bu operasyon Cat1LieAlg() fonksiyonuna bağlanmıştır. Cat1LieAlgSelect() operasyonunu kullanmak için dört temel parametre vardır :

```
Cat1LieAlgSelect( <gf>, <size>, <gpnum>, <num> );
```

<gf> : Lie cebirini oluşturacak Galois cisminin mertebesi,

<size> : Kullanılacak grubun mertebesini,

<gpnum> : Aynı mertebeden farklı grupların liste içerisindeki sırasını,

<num> : Verilen ilk üç değere göre oluşturulan cat^1 -Lie cebirin sıra sayısıdır.

Bu fonksiyonunun kullanıma örnek aşağıdaki gibidir.

GAP

```
gap> LC:=Cat1LieAlg(8,3,2,1);
 2 is a invalid gpnum number.
 where 0 < gpnum <= 1 for gpsize 3.
Usage:  Cat1LieAlg( GF(num), gpsize, gpnum, num );
[ "Lie(GF(2^3)_c3)" ]
gap> LC:=Cat1LieAlg(8,3,1,1);
#I  LAGUNA package: Constructing Lie algebra ...
[Lie(GF(2^3)_c3) -> Lie(GF(2^3)_c3)]
gap> Display(LC);

Cat1-Lie algebra [Lie(GF(2^3)_c3)=>Lie(GF(2^3)_c3)] :-
: source algebra has generators:
  [ LieObject( (Z(2)^0)*() ), LieObject( (Z(2)^0)*(1,2,3) ),
  LieObject( (Z(2)^0)*(1,3,2) ) ]
:  range algebra has generators:
  [ LieObject( (Z(2)^0)*() ), LieObject( (Z(2)^0)*(1,2,3) ),
  LieObject( (Z(2)^0)*(1,3,2) ) ]
: tail homomorphism maps source generators to:
  [ LieObject( (Z(2)^0)*() ), LieObject( (Z(2)^0)*(1,2,3) ),
  LieObject( (Z(2)^0)*(1,3,2) ) ]
: head homomorphism maps source generators to:
  [ LieObject( (Z(2)^0)*() ), LieObject( (Z(2)^0)*(1,2,3) ),
  LieObject( (Z(2)^0)*(1,3,2) ) ]
: range embedding maps range generators to:
  [ LieObject( (Z(2)^0)*() ), LieObject( (Z(2)^0)*(1,2,3) ),
  LieObject( (Z(2)^0)*(1,3,2) ) ]
: the kernel is trivial.

gap> Size(LC);
[ 512, 512 ]
```

Alt cat1-Lie cebirlerini oluşturmak için aşağıdaki derlemeler tez kapsamında yazılmıştır.

```
SubCat1LieAlg(arg)
SubPreCat1LieAlg(arg)
IsSubCat1LieAlg(LC,KC)
SubCat1LieAlg(LC,KC)
```

Cat1-lie cebirler Cat1LieAlg kategorisi ile Lie cebirleri üzerinde çaprazlanmış modüller XModLieAlg kategorisi doğal denk kategorilerdir. Aşağıdaki derlemler bu denkliği kullanarak bir cat1-Lie cebirinden, çaprazlanmış modül elde etmek ve tersi için kullanılırlar.

```
PreCat1LieAlgByPreXModLieAlg (LX)
PreXModLieAlgByPreCat1LieAlg (LC)
XModLieAlgByCat1LieAlg (LC)
Cat1LieAlgByXModLieAlg (LX)
KernelDenkt (t)
KernelDenkh (h)
SDproduct (LC)
```

Program kullanıcısın, operasyonları daha rahat kullanabilmesi amacıyla bu derlemeler Cat1LieAlg() ve XModLieAlg() fonksiyonlarına bağlanarak yalnızca bu iki fonksiyon kullanılarak işlemler yapılabilir hale getirilmiştir.

GAP

```
gap> LC:=Cat1LieAlg(5,5,1,1);
#I  LAGUNA package: Constructing Lie algebra ...
[Lie(GF(5)_c5) -> Lie(GF(5)_c5)]
gap> Display(LC);

Cat1-Lie algebra [Lie(GF(5)_c5)=>Lie(GF(5)_c5)] :-
: source algebra has generators:
  [ LieObject( (Z(5)^0)*() ), LieObject( (Z(5)^0)*(1,2,3,4,5) ),
  LieObject( (Z(5)^0)*(1,3,5,2,4) ), LieObject( (Z(5)^0)*(1,4,2,5,3) ),
  LieObject( (Z(5)^0)*(1,5,4,3,2) ) ]
:  range algebra has generators:
  [ LieObject( (Z(5)^0)*() ), LieObject( (Z(5)^0)*(1,2,3,4,5) ),
  LieObject( (Z(5)^0)*(1,3,5,2,4) ), LieObject( (Z(5)^0)*(1,4,2,5,3) ),
  LieObject( (Z(5)^0)*(1,5,4,3,2) ) ]
: tail homomorphism maps source generators to:
  [ LieObject( (Z(5)^0)*() ), LieObject( (Z(5)^0)*(1,2,3,4,5) ),
```

```
  LieObject( (Z(5)^0)*(1,3,5,2,4) ), LieObject( (Z(5)^0)*(1,4,2,5,3) ),
  LieObject( (Z(5)^0)*(1,5,4,3,2) ) ]
: head homomorphism maps source generators to:
  [ LieObject( (Z(5)^0)*() ), LieObject( (Z(5)^0)*(1,2,3,4,5) ),
  LieObject( (Z(5)^0)*(1,3,5,2,4) ), LieObject( (Z(5)^0)*(1,4,2,5,3) ),
  LieObject( (Z(5)^0)*(1,5,4,3,2) ) ]
: range embedding maps range generators to:
  [ LieObject( (Z(5)^0)*() ), LieObject( (Z(5)^0)*(1,2,3,4,5) ),
  LieObject( (Z(5)^0)*(1,3,5,2,4) ), LieObject( (Z(5)^0)*(1,4,2,5,3) ),
  LieObject( (Z(5)^0)*(1,5,4,3,2) ) ]
: the kernel is trivial.

gap> LX:=XModLieAlg(LC);
[Algebra( GF(5), [], LieObject( <zero> of ... ) )->Lie(GF(5)_c5)]
gap> Display(LX);

Crossed module of Lie algebras [..->Lie(GF(5)_c5)] :-
: Source algebra has generators:
  [  ]
: Range algebra has generators:
  [ LieObject( (Z(5)^0)*() ), LieObject( (Z(5)^0)*(1,2,3,4,5) ),
  LieObject( (Z(5)^0)*(1,3,5,2,4) ), LieObject( (Z(5)^0)*(1,4,2,5,3) ),
  LieObject( (Z(5)^0)*(1,5,4,3,2) ) ]
: Boundary homomorphism maps source generators to:
  [  ]

gap> LCC:=Cat1LieAlg(LX);
[Lie(GF(5)_c5)=>Lie(GF(5)_c5)]
gap> LCC=LC;
true
gap>
```

BÖLÜM 4

SONUÇLAR VE TARTIŞMA

Cayley teoremi gereğince her grup bir permütasyonik grubun alt grubuna izomorf olduğundan permütasyonik gruplar yardımıyla tüm gruplar elde edilebilir. Bilgisayar ortamında cebirler üzerinde hesaplamalar yapabilmek için permütasyon grupların bu özelliğinden faydalandık. Grup cebir kavramı bu amaçlar herhangi bir cisim ve permütasyonik grup kullanarak cebir elde etmek için kullanılır. Yapılan bu çalışma ile birlikte özellikle grup teorisi alanında güçlü bir program olan GAP ile Lie cebirleri üzerindeki cat^1-cebirler ilk kez bilgisayar ortamına aktarılmış ve bu yapı üzerindeki hesaplamalar yapılabilmiştir. Yine cebirler üzerinde çaprazlanmış modüller ve cat^1-cebirlerlerle ilgili hesaplamalar tez kapsamında verilmiştir. Fizikte non-komütatif geometri temelinde süpersimetrik kuantum mekaniği, kuantum grup ve cebirleri, kuantum grup simetriler sistemlerde, quantum bilgi teorisinde ve kuantum şifrelemede kullanım alanlarına sahip bu çalışma fiziksel uygulama ve örneklerle daha zengin bir hale getirilebilir.

Yapılan tüm hesaplamalara rağmen, GAP programı cebirler üzerinde aktör çaprazlanmış modül, çaprazlanmış modüllerin direkt çarpımı ve çaprazlanmış kare gibi yapıların incelenebilmesine olanak sağlayamamıştır. Bundan dolayı cebirler alanındaki daha ileri bazı çalışmalar için GAP programı en elverişli program olmayabilir. Bu alandaki diğer güçlü programlar olan CoCoA (Computations in Commutative Algebra) ve MAGMA (Computational Algebra System) kullanılarak bu cebirsel yapılar için bir inceleme yapılabilir. Yapılan bu tez çalışması ışığında, cat^2-cebir ve cat^2-Lie cebirlerinin GAP programı ile bilgisayar ortamına aktarılması mümkün gözükmektedir.

KAYNAKLAR DİZİNİ

Alp, M. and Wensley, 2000, Crossed Modules and Cat^1-groups in GAP, version 2.1 Manual for the XMOD share package.

Amoya, R. , Stewart, I., 1974, Infinite-dimensional lie algebras, Nordhoff.

Arvasi, Z., Porter T., 1996, Simplicial and Crossed Resolutions of Commutative Algebras, Journal of Algebras,181, 426-448.

Arvasi, Z., Ege, U., 2003, Annihilators, Multipliers and Crossed Modules, Applied Categorical Structures, Vol.11, 487-506.

Aslan, A. F., 2010, GAP ile Grup Cebirlerin Lie Cebirleri, Yüksek Lisans Tezi, Osmangazi Universitesi.

Aytekin, A., 2005, Lie Cebirlerin Çaprazlanmış Modülleri, Yüksek Lisans Tezi, Dumlupınar Üniversitesi.

Barker, M., 2003, Representations of Crossed Modules and Cat^1-Groups, Ph.D. Thesis, University of Wales, Bangor.

Bovdi, V., Konovalov, A., Rossmanith, R. , SchneiderBarker, C., 2009, LAGUNA : Lie AlGebras and UNits of group Algebras, GAP Share Package.

Brown, R., Wensley, Ch. D., 1995, Christopher. On finite induced crossed modules, and the homotopy 2-type of mapping cones, Theory Appl. Categ. 1 , No. 3, 54-70.

Brown, R., Wensley, Ch. D., 1996, Computing Crossed Modules Induced by an Inclusion of a Normal Subgroup, with Applications to Homotopy 2-types, Theory Appl. Categ., 2, 1, 3-16.

Brown, R., Wensley, Ch. D., 2003, Computation and homotopical applications of induced crossed modules, . J. Symbolic Comput. 35 (2003), no. 1, 59-72.

Casas, J.M., 1990, Invariantes de Módulos Cruzados en Álgebras de Lie, Ph.D.Thesis, University of Santiago.

Cayley, A., 1854, On the theory of groups as depending on the symbolic equation $\theta^n = 1$, Phil. Mag. 7, 40-47.

Ellis, G.J., 1988, Higher Dimensional Crossed Modules of Algebras, J.Pure Appl. Algebra 52 , 277-282.

Ege, U., 1998, Çaprazlanmış Modüller, Yüksek Lisans Tezi, Osmangazi Universitesi.

Erdmann, K. , Wildon, M. J. , Introduction to Lie Algebras, 2006, Oxford University, UK.

GAP:, GAP - Groups, Algortihms, and Programming, Version 4, 2008, Lehrstuhl D für Mathematik, U. St. Andrews, Scotland.

Kassel, C., Loday, J.L. 1982, Extensions centrales d'algèbres de Lie, Ann. Inst. Fourier (Grenoble) 33, 119-142.

Bovdi, V., Konovalov, A., Rossmanith, R., Schneider, C., 2003, LAGUNA - Lie AlGebras and UNits of group Algebras, GAP share package.

Milies, C. P., Sehgal, S. K, 2002, An Introduction to Group Rings Kluwer Academic Publishers.

Norrie, K.J., 1987, Crossed Modules and Analogues of Group Teorems, Ph.D.Thesis, King's College.

Odabaş, A. , GAP (Grup, Algoritma, Programlama) ile Cebirler ve Sayılar Teorisi, 2004, Yüksek Lisans Tezi, Dumlupınar Üniversitesi.

Odabaş, A. , GAP (Grup, Algoritma, Programlama) ile Cebirler Üzerinde Çaprazlanmış Modüller, 2009, Doktora Tezi, Eskişehir Osmangazi Üniversitesi.

Passman, D. S. , 1977, The Algebraic Structure Of Group Rings, Wiley, New York.

Porter, T., 1978, Some Categorical Results in the Category of Crossed Modules in Commutative Algebra, J. Algebra, 109, 415-429.

Porter, T., 1986, Homology of Commutative Algebras and an Invariant of Simis and Vasconceles, J. Algebra, 99, 458-465.

Shammu, N.M., 1992, Algebraic and an Categorical Structure of Category of Crossed Modules of Algebras, Ph.D. Thesis,U.C.N.W.

Whitehead, J. H. C., 1949, Combinatorial Homotopy I, Bull. Amer. Math. Soc., 55, 453-496.