AXIOM İLE CEBİR PROBLEMLERİNİN ÇÖZÜMÜ

Esma TAYRAN

Yüksek Lisans Tezi

Matematik Anabilim Dalı

Haziran - 2016

AXIOM İLE CEBİR PROBLEMLERİNİN ÇÖZÜMÜ

Esma TAYRAN

Dumlupınar Üniversitesi
Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliği Uyarınca
Fen Bilimleri Enstitüsü Matematik Anabilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır.

Danışman: Prof. Dr. Erdal ULUALAN

Haziran - 2016

KABUL VE ONAY SAYFASI

Esma TAYRAN'ın YÜKSEK LİSANS tezi olarak hazırladığı *"Axiom ile Cebir Problemlerinin Çözümü"* başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

09/06/2016

Üye : Prof. Dr. Erdal ULUALAN (Danışman)

Üye : Doç. Dr. Ahmet BOZ

Üye : Yrd. Doç. Dr. Alper ODABAŞ

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

Prof. Dr. Hasan GÖÇMEZ
Fen Bilimleri Enstitüsü Müdürü

ETİK İLKE VE KURALLARA UYGUNLUK BEYANI

Bu tezin hazırlanmasında Akademik kurallara riayet ettiğimizi, özgün bir çalışma olduğunu ve yapılan tez çalışmasının bilimsel etik ilke ve kurallarına uygun olduğunu, çalışma kapsamında teze ait olmayan veriler için kaynak gösterildiğini ve kaynaklar dizininde belirtildiğini, Yüksek Öğretim Kurulu tarafından kullanılmak üzere önerilen ve Dumlupınar Üniversitesi tarafından kullanılan İntihal Programı ile tarandığını ve benzerlik oranının %27 çıktığını beyan ederiz. Aykırı bir durum ortaya çıktığı takdirde tüm hukuki sonuçlara razı olduğumuzu taahhüt ederiz.

Danışman Adı Soyadı	Öğrenci Adı Soyadı
İmza	İmza

AXIOM İLE CEBİR PROBLEMLERİNİN ÇÖZÜMÜ

Esma TAYRAN

Matematik Anabilim Dalı, Yüksek Lisans Tezi, 2016

Tez Danışmanı: Prof.Dr. Erdal ULUALAN

ÖZET

Bu tez beş bölümden oluşmaktadır. Birinci bölümde Axiom sisteminin tanıtılması ve bilgisayara kurulumunun nasıl yapılacağı anlatılmıştır. İkinci bölümde Axiom kullanarak yapılabilecek bazı işlemlere yer verilmiştir. Üçüncü bölümde Soyut Cebirin bazı konularına ait sorular Axiom ile çözülmüştür. Dördüncü bölümde [Brown ve Tonks(1992)] de verilen Simplisel Operatörler ile ilgili paket ve örneklere yer verilmiştir. Beşinci bölümde Axiom'da kullanılan bazı sistem komutlarına yer verilmiştir.

Anahtar Kelimeler: Axiom, Devirli Gruplar, FriCAS, OpenAxiom, Simetrik Gruplar, Simplisel Operatörler, Sonlu Cisimler

SOLVING ALGEBRA PROBLEMS WITH AXIOM

Esma TAYRAN

Department of Mathematics, MS. Thesis, 2016

Thesis Supervisor : Prof.Dr. Erdal ULUALAN

SUMMARY

This thesis consists of five chapters. In the first chapter, we give a short introduction of Axiom system and how it is installed on the computer. In the second chapter, it is mentioned that some problems general mathematics can be solved by using Axiom. In the third chapter, some of problems of Abstract Algebra are solved by using Axiom. In the fourth chapter, a package about Simplicial Operators and some examples in the article written by [Brown ve Tonks(1992)] are mentioned. In the fifth chapter, some of system commands used in Axiom are given.

Keywords: Axiom, Cyclic Groups, Finite Fields, FriCAS, OpenAxiom, Simplicial Operators, Symetric Groups

TEŞEKKÜR

Bu tezin hazırlanma sürecinde desteklerini ve teşviklerini esirgemeyen sayın hocam Prof. Dr. Erdal ULUALAN'a, çalışmam boyunca hep yanımda olan anneme, babama, kardeşlerime ve arkadaşlarıma teşekkür ederim.

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

1. TEMEL BİLGİLER

1.1. Axiom Nedir?

Axiom 1971 yılında IBM çatısı altında "ScratchPad" adıyla geliştirilen açık kaynak kodlu bir bilgisayar cebiri sistemidir. 90'lı yıllarda NAG (Numerical Algoritms Group)'a satıldıktan sonra 2001 yılında ücretli ve ücretsiz olmak üzere iki farklı sürüm ile kullanıcılara sunuldu. Halen çeşitli kullanıcılar tarafından, ücretsiz olan "OpenAxiom" ve "FriCAS" dağıtımları geliştirilmekte ve kullanılmaktadır.

1.2. Axiom'un Bilgisayara Kurulumu

Ubuntu İşletim Sisteminde Kurulumu

Çoğu Linux tabanlı sistem Axiom'un kurulmasına olanak tanır. Bu sistemlerde kurulum yapmak için aşağıdaki komutun yazılması yeterlidir;
sudo apt-get install openaxiom

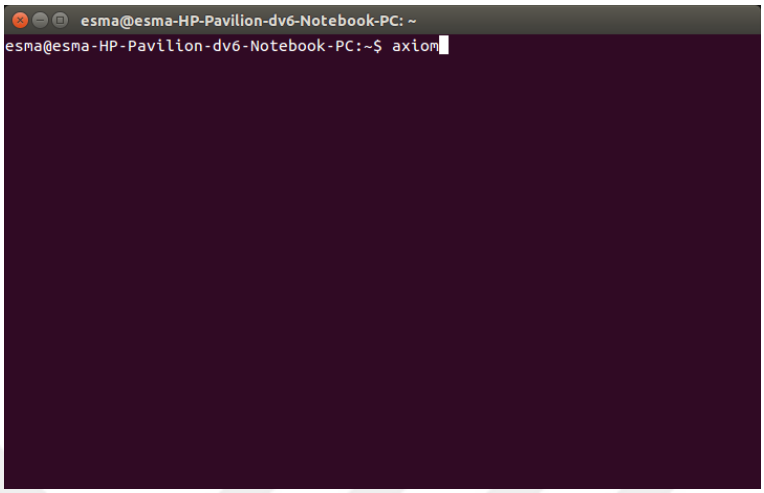
Axiomun kurulumu "Ubuntu Yazılım Merkezi" aracılığıyla da gerçekleştirilebilir. Bunun için "Ubuntu Yazılım Merkezinde" arama çubuğuna "Axiom" yazarak programın bulunması ve sonrasında kurulumu sağlanır. Uçbirimde (terminal) "axiom" yazılarak Axiom'un çalıştırılabildiği HyperDoc ortamı ve interaktif olan terminal ortamı açılmış olur.

Windows ve Mac OS İşletim Sistemlerinde Kurulumu

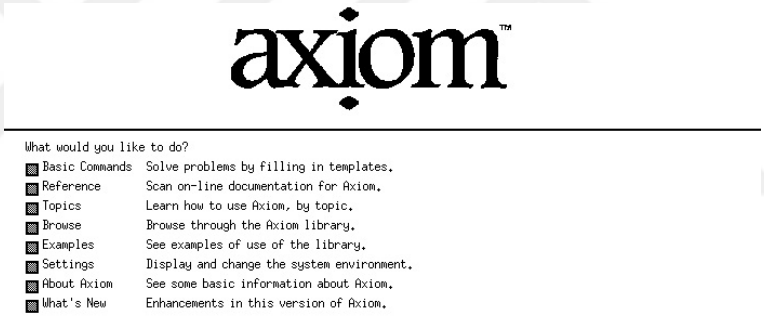
Windows ve diğer sistemlerdeki kurulum için
"http://axiom-developer.org/axiom-website/download.html"
adresindeki yönergeler uygulanarak kurulum sağlanır.

HyperDoc Ortamı

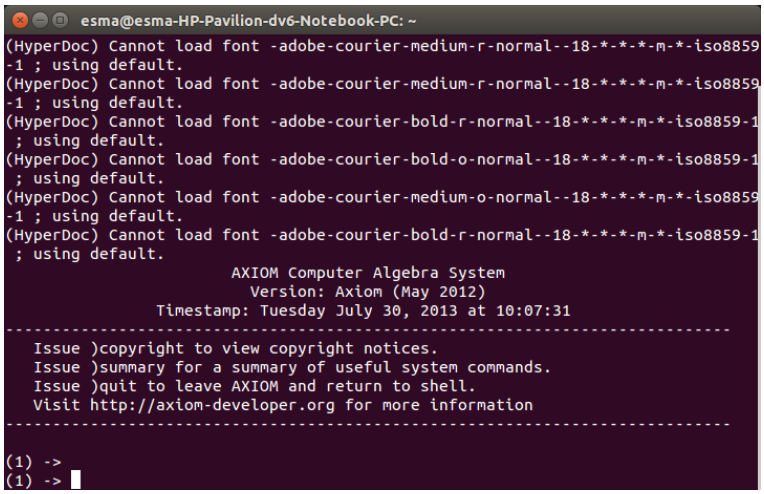
HyperDoc, Axiom'un kullanıldığı ortamlardan biridir. İnteraktif çalışma ortamı ve çevrimiçi kaynaklara erişim imkanı sunar. Axiom da önceden tanımlanmış bazı işlemlerin kullanıldığı bir ortamdır.



Şekil 1.1.: Axiom'un açılması.



Şekil 1.2.: HyperDoc ortamı.



Şekil 1.3.: Terminal ortamı.

2. AXIOM'UN GENEL KULLANIMI

Axiom, polinomlar, matrisler ve kuvvet serileri gibi cebirsel ifadelerin oluşturduğu veri tipleri de dahil olmak üzere birçok farklı veri tipini destekler. Kullanıcı kendi tanımladığı veri tiplerini ve işlemlerini tanımlayabilir. Axiom, neredeyse tüm işlem çeşitlerini destekleyen geniş bir sayısal hafızaya sahiptir.

Axiom temel anlamda hesap makinesi olarak kullanılabilir. Örneğin sinüs 1.2 değerinin hesaplanması aşağıdaki gibidir:

```
sin(1.2)
```

0.9320390859 6722634967

Type: Float

Sonuç ön tanımlı olarak, virgülden sonra 20 basamaklı olur. Bunu değiştirmek için "digits()" fonksiyonu kullanılır.

Axiom, sonucun veri tipini kullanıcıya gösteririr. Böylece bazı deneysel hesaplamaların yapıldığı durumlarda tahmin edilen veri tipinin doğru olup olmadığı belirlenir. Yukarıdaki örnekte sonucun veri tipi "float"tır.

Aşağıdaki işlem yapılırsa;

```
2/6
```

$$1/3$$

Type: FractionInteger

sonucu elde edilir.

Sonucun belirli bir veri tipinde olması isteniyorsa "::" operatörü kullanılır. Buna göre yukarıdaki işlemin sonucunun ondalıklı sayı olması isteniyorsa:

```
(2/6)::Float
```

0.3333333333 3333333333

Type: Float

Ondalıklı olarak verilen bir ifadenin kesirli haline ulaşmak isteniyorsa, Axiom sonuca en yakın kesirli ifadeyi hesaplar.

```
%::Fraction Integer
```

$$\frac{1}{3}$$

Type: FractionInteger

% sembolü, "bir önceki işlemin sonucu" anlamına gelir. Bu komut ile hesaplanan kesirli ifadeler, yuvarlama hatalarından dolayı, asıl kesir olmayabilir.

Ondalıklı sayıları tamsayılara çevirmek için "round()" ve "truncate()" fonksiyonları kullanılabilir. "round()" fonksiyonuyla ondalıklı sayı, en yakınındaki tam sayıya yuvarlanır. "truncate()" fonksiyonuyla ise ondalıklı sayının virgülden sonraki basamakları silinerek tam sayı hali elde edilir. Her iki fonksiyon da ondalıklı sayı olan sonuçlarda kullanılır. Bir ondalıklı sayının, sadece ondalıklı kısmına ulaşmak için "fractionPart()" fonksiyonu kullanılır.

```
round(2.18935)
```

2.0

```
Type: Float
```

```
round(-2.18935)
```

-2.0

```
Type: Float
```

```
truncate(7.459)
```

7.0

```
Type: Float
```

```
truncate(-7.459)
```

-7.0

```
Type: Float
```

```
fractionPart(-4.05327)
```

-0.05327

```
Type: Float
```

Bir sayının tamdeğerini bulmak için "abs()" fonksiyonu kullanılabilir. Bu fonksiyon, ondalıklı veya tamsayılı değerler alır. "sign()" fonksiyonu girilen sayıya, "$-1, 0, 1$" değerlerinden birini karşılık getirir.

```
abs(6)
```

6

```
Type: PositiveInteger
```

```
abs(-2)
```

$$2$$

```
                                                    Type: PositiveInteger
```

```
abs(-215.9851)
```

$$215.9851$$

```
                                                        Type: Float
```

```
sign(-12435.78129)
```

$$-1$$

```
                                                      Type: Integer
```

```
sign(0)
```

$$0$$

```
                                                 Type: NonNegativeInteger
```

```
sign(24561.43670)
```

$$1$$

```
                                                    Type: PositiveInteger
```

Axiom da çeşitli verilerin doğruluklarının kontrol edilmesinde "?" operatörü kullanılır. Aşağıdaki örneklerde, çeşitli verilerin pozitif veya negatif olup olmadığı, tek veya çift sayı olup olmadığı, asal sayı olup olmadığı gibi özellikler incelenmiştir.

```
positive?(-125)
```

false

```
                                                      Type: Boolean
```

```
negative?(-55)
```

true

```
                                                      Type: Boolean
```

```
zero?(21)
```

false

```
                                                      Type: Boolean
```

```
one?(1)
```

true

Type: Boolean

```
odd?(27)
```

true

Type: Boolean

```
odd?(7.485)
```

false

Type: Boolean

```
even?(-28)
```

true

Type: Boolean

```
prime?(41)
```

true

Type: Boolean

```
prime?(-41)
```

false

Type: Boolean

Sık kullanılan bazı işlemler şunlardır:

sin(x)	x açısının sinüsü
cos(x)	x açısının kosinüsü
tan(x)	x açısının tanjantı
asin(x)	x açısının arcsinüsü
acos(x)	x açısının arccosu
atan(x)	x açısının arctanjantı
gcd(x,y)	x ve y nin en büyük ortak böleni
lcm(x,y)	x ve y nin en küçük ortak katı
max(x,y)	x ve y den büyük olanı
min(x,y)	x ve y den küçük olanı
factorial(x)	x in faktöriyeli
factor(x)	x in asal çarpanları
divide(x,y)	x/y nin bölüm ve kalanı

+	Toplama	−	Çıkarma
-	Sayısal olumsuzluk	∼	Mantıksal olumsuzluk
/\	"ve" bağlacı	\/	"veya" bağlacı
and	Mantıksal "ve"	or	Mantıksal "veya"
not	Mantıksal olumsuzluk	**	Üs alma
*	Çarpma	/	Bölme
quo	Bölüm	rem	Kalan
<	Küçüktür	>	Büyüktür
<=	Küçük eşittir	>=	Büyük eşittir

Bazı Axiom makroları:

%i	-1 in karekökü
%e	Doğal logaritma tabanı
%pi	Pi sayısı
%infinity	Sonsuz sembolü
%plusInfinity	Pozitif sonsuzluk
%minusInfinity	Negatif sonsuzluk

Aşağıda sembolik terim içeren bazı basit cebirsel ifadelerin Axiom ile gösterimleri incelenmiştir.

```
xKare := x**2
```

$$x^2$$

```
                                                    Type: Polynomial Integer
```

Yukarıdaki örnekteki ":=" fonksiyonu, cebirsel ifadeye bir değer ataması yapılırken kullanılır. Sonucun veri tipinin "Polynomial Integer" olması, sonucun tamsayı katsayılı bir polinom olduğunu belirtir.

```
xFonk := 1.25*x**2
```

$$1.25x^2$$

```
                                                      Type: Polynomial Float
```

```
xFonk := x**3.5
```

$$x^3\sqrt{x}$$

```
                                                    Type: Expression Float
```

```
xyFonk := x**3 + y**3-2*x*y
```

$$y^3 - 2xy + x^3$$

```
                                                    Type: Polynomial Integer
```

Yukarıdaki polinomların çözümleri "eval" fonksiyonu yardımıyla yapılır. Aşağıdaki örneklerde xFonk ve xyFonk olarak tamımlanan polinomların, önceden belirlenen x ve y değerleri için çözümleri yapılmıştır:

```
eval(xFonk,x=2)
```

$$11.3137084989 \quad 8476039$$

```
                                                    Type: Expression Float
```

```
eval(xyFonk, [x=2, y=1.3])
```

$$4.957$$

```
                                                      Type: Polynomial Float
```

Axiomda, "-1" in karekökü için "%i" makrosu kullanılarak karmaşık sayılardaki işlemler kolayca yapılabilir.

```
(2/5 + %i)**4
```

$$\frac{41}{625} - \frac{168}{125}\%i$$

```
                                                    Type: Complex Fraction Integer
```

Bir karmaşık sayının gerçek ve sanal kısımlarının gösterimi için "real" ve "imag" fonksiyonları kullanılır. Bir karmaşık sayının eşleniğinin bulunması için ise "conjugate" fonksiyonu kullanılır:

```
real(2+5*%i)
```

$$2$$

```
                                                    Type: Positive Integer
```

```
imag(2+5*%i)
```

$$5$$

```
                                                    Type: Positive Integer
```

```
conjugate(2+5*%i)
```

$$2-5\%i$$

```
                                                    Type: Complex Integer
```

"factor" fonksiyonu karmaşık sayılar için de kullanılabilir. Ancak sonuç bir tamsayının sonucu kadar belirgin değildir.

```
factor(121+12*%i)
```

$$-\%i(1+2\%i)(46+29\%i)$$

```
                                          Type: Factored(Complex(Integer))
```

Axiom tüm sayısal sonuçlar için öntanımlı olarak ondalıklı sayıdır ve reel sayılarda 20 basamak gösterilir. Sayının tamsayı olan kısmı 20 basamaktan fazla ise virgülden sonra sadece tamsayı olan kısım gösterilir. Gösterilen basamak sayısını değiştirmek için "digits" fonksiyonu kullanılır. Bu fonksiyonun ilk kullanımında, öntanımlı olan basamak sayısının kaç olduğu gösterilir. Sonrasında hesaplanması istenen ifade yazılır. Örneğin, π nin 40 basamağını göstermek aşağıdaki gibidir:

```
digits(40)
```

$$20$$

```
                                                    Type: PositiveInteger
```

```
%pi::Float
```

$$3.1415926535\quad 8979323846\quad 2643383279\quad 502884197$$

```
                                                                    Type: Float
```

Axiom uzun sonuçları gösterirken, okumanın kolay olması için, her 10 basamakta bir boşluk bırakır. Bu durum "outputSpacing" fonksiyonuyla değiştirilebilir. Eğer boşluk bırakılması istenmiyorsa parametre 0 alınır. Reel sayıların görünüşlerini değiştirmek için kullanılabilecek diğer iki fonksiyon "outputFloating" ve "outputFixed" dir. Bu fonksiyonlardan birincisi ondalıklı sayıların üslü ifade olarak gösterimini sağlar. İkinci fonksiyon ise sabit nokta gösterimini sağlar. Örneğin:

```
outputFloating() ; %
```

$$0.31415926535897932384626433832795028841 97E1$$

```
                                                                    Type: Float
```

```
outputFloating(3); 0.00345
```

$$0.345E-2$$

```
                                                                    Type: Float
```

```
outputFixed();%
```

$$0.00345$$

```
                                                                    Type: Float
```

```
outputFixed(3);%
```

$$0.003$$

```
                                                                    Type: Float
```

```
outputGeneral();%
```

$$0.00345$$

```
                                                                    Type: Float
```

";" ile bir satıra birden fazla ifadenin yazılması sağlanır. Onluk taban dışındaki gösterimlerde "radix" fonksiyonu kullanılır. İlk parametre değiştirilmesi istenen sayıyı, ikinci parametre ise değiştirilmek istenen tabanı gösterir.

```
radix(5**12,20)
```

$$3G5HBB5$$

```
                                                    Type: RadixExpansion 20
```

```
radix(4/15,3)
```

$$0.0\overline{2101}$$

```
                                                    Type: RadixExpansion 3
```

Rasyonel sayılar, "decimal" fonksiyonuyla tekrar eden ondalıklı ifade olarak veya "continuedFraction" komutuyla devam eden kesirler olarak gösterilebilirler. İrrasyonel sayılar için bu fonksiyonlar kullanılamaz.

```
decimal(19/7)
```

$$2.\overline{714285}$$

```
                                                    Type: DecimalExpansion
```

```
continuedFraction(1146/213)
```

$$5 + \frac{1|}{|2} + \frac{1|}{|1} + \frac{1|}{|1} + \frac{1|}{|1} + \frac{1|}{|2} + \frac{1|}{|3}$$

```
                                          Type: ContinuedFraction Integer
```

Kısmi kesirlerin sıkıştırılmış ve genişletilmiş hallerine sırasıyla "partialFraction" ve "padicFraction" fonksiyonlarıyla ulaşılır. Birinci fonksiyon, kesrin payı ve paydası olmak üzere iki parametre alır. İkinci fonksiyon ise parametre olarak bir kesir alır ve "compactFraction" fonksiyonunun aksine, kesri genişletir:

```
partialFraction(6543,20)
```

$$327 + \frac{3}{2^2} - \frac{3}{5}$$

```
                                       Type: PartialFraction Integer
```

```
padicFraction(%)
```

$$327 + \frac{1}{2} + \frac{1}{2^2} - \frac{3}{5}$$

```
                                       Type: PartialFraction Integer
```

```
compactFraction(%)
```

$$327 + \frac{3}{2^2} - \frac{3}{5}$$

```
                                       Type: PartialFraction Integer
```

```
padicFraction(6543/20)
```

$$\frac{6543}{20}$$

```
                                                    Type: PartialFraction Fraction Integer
```

"firstNumer" fonksiyonuyla ilk kesrin payı ve "firstDenom" fonksiyonuyla ilk kesrin paydası bulunur. Parçalı kesrin tam kısmı "wholePart" fonksiyonuyla ve kaç tane kesirli ifadenin olduğu "numberOfFractionalTerms" fonksiyonuyla gösterilir.

```
z := partialFraction(6543,20)
```

$$327 + \frac{3}{2^2} - \frac{3}{5}$$

```
                                                    Type: PartialFraction Integer
```

```
wholePart(z)
```

$$327$$

```
                                                    Type: PositiveInteger
```

```
numberOfFractionalTerms(z)
```

$$2$$

```
                                                    Type: PositiveInteger
```

```
p := nthFractionalTerm(z,1)
```

$$\frac{3}{2^2}$$

```
                                                    Type: PartialFraction Integer
```

```
firstNumer(p)
```

$$3$$

```
                                                    Type: Integer
```

```
firstDenom(p)
```

$$2^2$$

```
                                                    Type: Factored Integer
```

"PrimeField" veri tipi ile asal sayılar için modüler aritmetik işlemi yapılabilir. Örneğin, $mod 5$ e göre işlem yapmak için aşağıdakini yazmak gereklidir:

```
x : PrimeField 5 := 4
```

$$4$$

```
                                                    Type: PrimeField 5
```

Yukarıdaki ifadeler şu anlama gelir:

```
x, mod 5 e göre 4 olsun.
```

```
x**4 + 2
```

$$3$$

```
                                                    Type: PrimeField 5
```

```
1/x
```

$$4$$

```
                                                    Type: PrimeField 5
```

$mod5$ e göre bir polinom oluşturmak şu şekilde yapılabilir:

```
(2*a**3 + 26*a - 34)::Polynomial PrimeField 5
```

$$2a^3 + a + 1$$

```
                                          Type: Polynomial PrimeField 5
```

Asal olmayan değerler için şu şekilde işlem yapılabilir:

```
y : IntegerMod 9 := 15
```

$$6$$

```
                                                    Type: IntegerMod 9
```

```
y*3-32
```

$$4$$

```
                                                    Type: IntegerMod 9
```

Axiom ile denklem sistemlerinin çözümü yapılabilir. Eğer denklem ve bilinmeyen sayısı eşitse, ve sembolik katsayılar yoksa, reel kökleri bulmak için "solve" fonksiyonu, karmaşık kökleri bulmak için ise "complexSolve" fonksiyonu kullanılır. Böylece istenen sonucun nasıl olması gerektiği belirtilmiş olur. Aşağıda h parametresine bağlı iki denklemden oluşan bir sistem verilmiştir.

```
S(h) == [x**2-2*y**2 - h,x*y-y-5*x + 5]
```

```
                                                    Type: Void
```

```
solve(S(19))
```

$$\left[[y^2+9=0, \quad x=1], [y=5, \quad x^2-69=0]\right]$$

```
Type: List (List (Equation (Polynomial (Fraction (Integer)))))
```

$1/10^{20}$ doğruluğunda $S(19)$ un reel köklerinin bulunması:

```
solve(S(19),1/10**20)
```

$$\left[\left[y=5, x=-\frac{2451682632253093442511}{295147905179352825856}\right],\right.$$
$$\left.\left[y=5, x=\frac{2451682632253093442511}{295147905179352825856}\right]\right]$$

```
Type: List (List (Equation (Polynomial (Fraction (Integer)))))
```

$S(19)$ un karmaşık köklerinin bulunması:

```
complexSolve(S(19),10.e-20)
```

$$[[y=5.0, x=8.3066238629180748526],$$
$$[y=5.0, x=-8.3066238629180748526],$$
$$[y=-3.0i, x=1.0], [y=3.0i, x=1.0]]$$

```
Type: List (List (Equation(Polynomial(Complex (Float)))))
```

Denklem sisteminin sembolik kökleri var ve kökü ifadeli çözümü bulunmak isteniyorsa, "radicalSolve" fonksiyonu kullanılır.

```
radicalSolve(S(a),[x,y])
```

$$\left[\left[x=-\sqrt{a+50}, y=5\right], \left[x=\sqrt{a+50}, y=5\right],\right.$$

$$\left.\left[x=1, y=\sqrt{\frac{-a+1}{2}}\right], \left[x=1, y=-\sqrt{\frac{-a+1}{2}}\right]\right]$$

```
Type: List (List (Equation (Expression (Integer))))
```

3. AXIOM İLE SOYUT CEBİR ÖRNEKLERİ

3.1. Polinomların Çarpanlarına Ayrılması

Axiom, geniş bir katsayı çeşitliliğine sahip olduğundan, bir çok polinomun çarpanlarına ayrılmasını sağlar.

Tamsayı ve Rasyonel Sayı Katsayılı Polinomlar

Aşağıda, u polinomunun "factor u" fonksiyonu ile çarpanlarına ayrılışı gösterilmiştir.

```
u := (4*x**3-2*y**2-1)*(12*x**5+x**3*y-12)
```

$$-2x^3y^3+(-24x^5+24)y^2+(4x^6-x^3)y+48x^8-12x^5-48x^3+12$$

```
                                                    Type: Polynomial Integer
```

```
factor u
```

$$-(x^3y+12x^5-12)(2y^2-4x^3+1)$$

```
                                          Type: Factored Polynomial Integer
```

Aşağıdaki örnekte rasyonel katsayılı v polinomunun çarpanlarına ayrılışı gösterilmiştir.

```
v :=(3*x**2-(3/4)*x**2-1)*(6*x**4+(2/3)*x**2-15)
```

$$\frac{27}{2}x^6-\frac{9}{2}x^4-\frac{413}{12}x^2+15$$

```
                                          Type: Polynomial Fraction Integer
```

```
factor v
```

$$\frac{27}{2}\left(x-\frac{2}{3}\right)\left(x+\frac{2}{3}\right)\left(x^4+\frac{1}{9}x^2-\frac{5}{2}\right)$$

```
                                    Type: Factored Poynomial Fraction Integer
```

Sonlu Cisim Katsayılı Polinomlar

Sonlu cisim katsayılı polinomlar çarpanlarına ayrılabilir. Aşağıdaki örnekte, "PF(17)" ($\mathbb{Z}_{17}$) cismi üzerinde tanımlı bir u polinomu verilmiştir.

```
u : POLY(PF(17)) := 3*x**4+2*x**2+13*x+15
```

$$3x^4+2x^2+13x+15$$

```
                                          Type: Polynomial PrimeField 17
```

u polinomu, p asal sayı olmak üzere, *modp* ye göre çarpanlarına ayrılır.

```
factor u
```

$$3(x+7)^3(x+13)$$

```
                                                    Type: Factored Polynomial PrimeField 17
```

Aşağıdaki örnekte u polinomunun katsayıları, 17^3 elemanlı bir sonlu cismin elemanları olacak şekilde değiştirilmiştir.

```
factor(u :: POLY FFX(PF 17,3))
```

$$3(x+7)^3(x+13)$$

```
                    Type: Factored Polynomial FiniteFieldExtension(PrimeField 19,3)
```

Basit Genişletilmiş Cisim Katsayılı Polinomlar

Rasyonel sayıların basit cebirsel genişlemeleri katsayılı polinomlar, çarpanlarına ayrılabilir.

a, a^2+a+1 polinomunun sembolik kökü olsun.

```
a := rootOf(a**2+a+1)
```

$$a$$

```
                                                    Type: AlgebraicNumber
```

```
p:=(x**3+a**2*x*y)*(a*x**2+a*x+a*y**2)**2
```

$$axy^5+\big(-a-1\big)x^3y^4+\big(2ax^3+2ax^2\big)y^3+$$

$$\Big(\big(-2a-2\big)x^5+\big(-2a-2\big)x^4\Big)y^2+\big(ax^5+2ax^4+ax^3\big)y+$$

$$\big(-a-1\big)x^7+\big(-2a-2\big)x^6+\big(-a-1\big)x^5$$

```
                                                    Type: Polynomial AlgebraicNumber
```

Aşağıdaki örnekte, p polinomu a değerine göre yeniden çarpanlarına ayrılmıştır.

```
factor(p,[a])
```

$$ax\big(y+ax^2\big)\big(y^2+x^2+x\big)^2$$

```
                                    Type: Factored Polynomial AlgebraicNumber
```

Aşağıdaki örnekte, x^2+3 polinomunun a değerine göre çarpanlarına ayrılışı gösterilmiştir.

```
factor(x**2+3)
```

$$x^2+3$$

```
                                    Type: Factored Polynomial Integer
```

```
factor(x**2+3,[a])
```

$$(x - 2a - 1)(x + 2a + 1)$$

```
Type: Factored Polynomial AlgebraicNumber
```

Benzer şekilde $x^6 + 108$ polinomunun a elemanına göre çarpanlarına ayrılışı şu şekildedir:

```
factor(x**6+108,[a])
```

$$(x^3 - 12a - 6)(x^3 + 12a + 6)$$

```
Type: Factored Polynomial AlgebraicNumber
```

Aşağıdaki örnekte, $b^3 - 2$ polinomunun sembolik kökü b tanımlanmış ve $x^6 + 108$ polinomunun b köküne göre çözümü incelenmiştir.

```
b := rootOf(b**3-2)
```

$$b$$

```
Type: AlgebraicNumber
```

```
factor(x**6+108,[b])
```

$$(x^2 - 3bx + 3b^2)(x^2 + 3b^2)(x^2 + 3bx + 3b^2)$$

```
Type: Factored Polynomial AlgebraicNumber
```

$x^6 + 108$ polinomunun a ve b kökleri cinsinden çarpanlarına ayrılışı şu şekildedir:

```
factor(x**6+108,[a,b])
```

$$(x + (-2a - 1)b)(x + (-a - 2)b)(x + (-a + 1)b)$$

$$(x + (a - 1)b)(x + (a + 2)b)(x + (2a + 1)b)$$

```
Type: Factored Polynomial AlgebraicNumber
```

Rasyonel Katsayılı Fonksiyonların Çarpanlarına Ayrılması

Rasyonel sayı katsayılı polinomları çarpanlarına ayırmak için, pay ve paydayı ayrı ayrı çarpanlarına ayıran "factorFraction" fonksiyonu kullanılır.

```
factorFraction((x**2-4)/(y**2-4))
```

$$\frac{(x - 2)(x + 2)}{(y - 2)(y + 2)}$$

```
Type: Fraction Factored Polynomial Integer
```

3.2. Polinomların Sembolik Kökleri Üzerindeki İşlemler

Bu bölümde, bir polinomun bir veya birden fazla sembolik kökü üzerindeki bazı işlemler incelenmiştir.

Polinomun bir kökünün bulunması

Bir $p(x)$ polinomunun, x sembolik kökünü bulmak için "rootOf" komutu kullanılır. Aşağıdaki örnekte a^4+1 polinomunun a kökü oluşturulmuştur.

```
a := rootOf(a**4+1,a)
```

$$a$$

```
                                                    Type: Expression Integer
```

a yı tanımlayan cebirsel ifadeyi görmek için, "definingPolynomial" komutu kullanılır.

```
definingPolynomial a
```

$$a^4+1$$

```
                                                    Type: Expression Integer
```

```
b := rootOf(b**2-a-1,b)
```

$$b$$

```
                                                    Type: Expression Integer
```

```
a+b
```

$$b+a$$

```
                                                    Type: Expression Integer
```

```
% ** 5
```

$$\left(10a^3+11a^2+2a-4\right)b+15a^3+10a^2+4a-10$$

```
                                                    Type: Expression Integer
```

```
rootOf(c**2+c+1,d)
```

$$c$$

```
                                                    Type: Expression Integer
```

```
zeroOf(d**2+d+1,d)
```

$$\frac{\sqrt{-3}-1}{2}$$

```
                                                    Type: Expression Integer
```

```
rootOf(e**5-2,e)
```

$$e$$

```
                                                    Type: Expression Integer
```

```
zeroOf(f**5-2,f)
```

$$\sqrt[5]{2}$$

```
                                                    Type: Expression Integer
```

Polinomun tüm köklerinin bulunması

Bir polinomun tüm sembolik köklerini bulmak için "rootsOf" fonksiyonu kullanılır. Bu fonksiyon, $p(x)$ polinomunun tüm köklerini liste halinde gösterir. Eğer $p(x)$, n-inci dereceden bir köke sahipse, ilgili kök listede n defa gösterilir.

Aşağıda $x^4 + 1$ in tüm kökleri hesaplanmıştır.

```
l := rootsOf(x**4+1,x)
```

$$[\%x0, \%x0\%x1, -\%x0, -\%x0\%x1]$$

```
                                          Type: List Expression Integer
```

$\%x0$, $\%x1$ ve $\%x2$ değişkenleri, $x^4 + 1$ denkleminin ilk üç köküne bağlıdır.

```
%x0**5
```

$$-\%x0$$

```
                                                    Type: Expression Integer
```

$\%x0$, $\%x1$ ve $\%x2$ kökleri $x^4 + 1 = 0$ polinomunu sağlıyor olsalar da herbirisi farklı birer cebirsel sayıdır. Her bir kökün tanımladığı cebirsel ifadeyi görmek için, "definingPolynomial" komutu kullanılır.

```
definingPolynomial %x0
```

$$\%0^4 + 1$$

```
                                                    Type: Expression Integer
```

```
definingPolynomial %x1
```

$$\%1^2 + 1$$

```
                                                    Type: Expression Integer
```

```
definingPolynomial %x2
```

$$-\%2 + \%\%var$$

```
                                                    Type: Expression Integer
```

Aşağıda "zerosOf" fonksiyonuyla $y^4 + 1$ polinomunun kökleri bulunmuştur.

```
zerosOf(y**4+1,y)
```

$$\left[\frac{\sqrt{-1}+1}{\sqrt{2}}, \frac{\sqrt{-1}-1}{\sqrt{2}}, \frac{-\sqrt{-1}-1}{\sqrt{2}}, \frac{-\sqrt{-1}+1}{\sqrt{2}}\right]$$

```
                                                    Type: List Expression Integer
```

3.3. Sonlu Cisimler

Sonlu cisimler (Galois Cisimleri), rasyonel, reel veya karmaşık sayılar cisimlerinde olduğu gibi benzer kurallar içerisinde toplama, çarpma ve bölme işlemleri (ve değişme, birleşme ve dağılma özellikleri) yapılabilen sonlu bir cebirsel yapıdır. Diğer üç cisimden farklı olarak, herhangi bir sonlu cisim için karakteristiği olan bir pozitif p asal sayısı vardır, öyle ki sonlu cisimden alınan herhangi bir x elemanı için $px = 0$ dır. Gerçekte, bir sonlu cismin eleman sayısı, karakteristiğinin bir kuvvetidir ve her p asal sayısı ve n pozitif tamsayısı için, izomorfizme bağlı, sadece bir tane p^n elemanlı sonlu cisim vardır.

$n = 1$ iken, cisim p elemanlıdır ve asal cisimdir. Sonlu cisim genişlemesi yapmak için birden fazla yöntem vardır ve Axiom kullanıcının istediği yöntemi seçmesini sağlar. Dahası bir cisim üzerinde farklı genişlemeler ve farklı genişleme temsillerinin oluşturulması için gerekli işlemler yapılabilir. Bu işlemleri yapmak için uygun olan paketin seçilmesi gereklidir.

Modüler Aritmetik ve Asal Cisimler

n pozitif bir tamsayı olsun. Bilindiği gibi, tamsayılar üzerinde yapılan toplama, çıkarma veya çarpma işlemlerinin sonuçları birer tamsayıdır. Benzer şekilde herhangi bir tamsayının, n tamsayısı ile yaptığı kalanlı bölme işleminin sonucu da yine tamsayıdır. Böylece, "modulo n" veya "mod n" işleminden bahsedilebilir. Axiom'da herhangi bir pozitif n tamsayısı için, $modn$ işlemini yapmak için "IntegerMod" fonksiyonu kullanılır.

```
(x,y) : IntegerMod 12
```

```
                                                    Type: Void
```

```
(x,y) := (16,7)
```

```
                                                                Type: IntegerMod 12
```

```
[x - y , x * y]
```

$$[9, 4]$$

```
                                                                Type: IntegerMod 12
```

Eğer n tamsayısı asal değilse, sadece sınırlı olarak ters eleman kavramı ve bölme işlemi vardır.

7 ve 12 aralarında asal olduğundan, 7 nin "mod 12" ye göre çarpımsal tersi mevcuttur. "recip" fonksiyonuyla çarpımsal ters bulunur.

```
recip y
```

$$7$$

```
                                                    Type: Union(IntegerMod 12,...)
```

Eğer n sayısı, p ile aralarında asal ise, bu durumda tersi vardır ve bölme işlemi tanımlıdır. n asal iken "IntegerMod" yerine "PrimeField" (kısaca PF) fonksiyonu kullanılır.

```
z : PrimeField 11 := 8
```

$$8$$

```
                                                                Type: PrimeField 11
```

```
inv z
```

$$7$$

```
                                                                Type: PrimeField 11
```

z nin tersini bulmak için $1/z$ ve $z**(-1)$ ifadeleri kullanılabilir.

```
9/z
```

$$8$$

```
                                                                Type: PrimeField 11
```

Seçilen herhangi bir bölge üzerindeki rastgele bir elemanını bulmak için "random" fonksiyonu kullanılır. Aşağıdaki örnekte 127 elemanlı Galois Cisminin (GF) rastgele elemanları bulunmuş ve bu elemanlarla bazı işlemler yapılmıştır.

```
GF127 := PF 127
```

$$PrimeField127$$

```
                                                                Type: Domain
```

```
a := random()$GF127
```

$$78$$

```
                                                                Type: PrimeField 127
```

```
b : GF127 := 5
```

$$5$$

```
                                                                Type: PrimeField 127
```

```
c := a/b
```

$$41$$

```
                                                                Type: PrimeField 127
```

```
c * b -a
```

$$18$$

```
                                                                Type: PrimeField 127
```

```
pe := primitiveElement()$GF127
```

$$3$$

```
                                                                Type: PrimeField 127
```

bu cismin "ilkel elemanı" 3 tür ve kuvvetleri, sıfırdan farklı olan tüm elemanları oluşturur.

```
[pe**i for i in 0..99]
```

$$[1, 3, 9, 27, 81, 116, 94, 28, 84, 125, 121, 109, 73, 92, 22, 66, 71, 86, 4,$$
$$12, 36, 108, 70, 83, 122, 112, 82, 119, 103, 55, 38, 114, 88, 10, 30, 90,$$
$$16, 48, 17, 51, 26, 78, 107, 67, 74, 95, 31, 93, 25, 75, 98, 40, 120, 106$$
$$64, 65, 68, 77, 104, 58, 47, 14, 42, 126, 124, 118, 100, 46, 11, 33, 99,$$
$$43, 2, 6, 18, 54, 35, 105, 61, 56, 41, 123, 115, 91, 19, 57, 44, 5, 15$$
$$8, 24, 72, 89, 13, 39, 117, 97, 37, 111]$$

```
                                                                Type: PrimeField 127
```

Aşağıdaki örnekte, "discreteLog" fonksiyonu yardımıyla sıfırdan farklı b elemanının, ilkel elemanın hangi kuvveti olduğu bulunmuştur.

```
ex := discreteLog(y)
```

87

Type: PositiveInteger

```
pe ** ex
```

5

Type: PrimeField 127

Sıfırdan farklı bir x elemanının derecesi, $x^t = 1$ şartını sağlayan en küçük t pozitif tamsayısı idi. Axiom ile bu tamsayıyı bulmak için "order" fonksiyonu kullanılır.

```
order b
```

42

Type: PositiveInteger

Bir ilkel elemanın derecesi $p - 1$ ile tanımlıdır.

```
order pe
```

126

Type: PositiveInteger

Devirli Grup Temsilleri

Her sonlu cisimde, cismin sıfırdan farklı elemanlarının kuvvetleri şeklinde yazılan, ilkel eleman gibi elemanlar vardır. "FiniteFieldCyclicGroup" (kısaca FFCG) da sıfırdan farklı elemanlar, cismin sabit bir ilkel elemanının (cismin devirli çarpımsal grubunun üreteçi) kuvvetleri olarak temsil edilir. Çarpma (ve dolayısıyla kuvvet) işlemini temsilde kullanmak kolaydır. Toplama işlemi yapmak için ilkel eleman, ilkel polinomun (tüm kökleri ilkel olan indirgenemez polinom) kökü gibi düşünülür. "FiniteFieldCyclicGroup" bölgesini kullanmak için bir asal sayı ve genişletme derecesine ihtiyaç vardır.

Aşağıdaki örnekte 81 elemanlı Galois Cismi üzerinde, 3. dereceden ilkel polinom ve genişletme derecesi 4 seçilmiştir.

```
GF81 := FFCG(3,4);
```

```
                                                                Type: Domain
```

```
a := primitiveElement()$GF81
```

$$\%F^{1}$$

```
                                                Type: FiniteFieldCyclicGroup(3,4)
```

GF81 de hesaplama yapılabilir.

```
b := a**12 - a**5 + a
```

$$\%F^{72}$$

```
                                                Type: FiniteFieldCyclicGroup(3,4)
```

```
b
```

$$\%F^{72}$$

```
                                                Type: FiniteFieldCyclicGroup(3,4)
```

```
discreteLog b
```

$$72$$

```
                                                              Type: PositiveInteger
```

```
GF9 := FF(3,2);
```

```
                                                                Type: Domain
```

```
GF729 := FFCGX(GF9,3);
```

```
                                                                Type: Domain
```

```
r := (random()$GF729) ** 20
```

$$\%H^{420}$$

```
                    Type: FiniteFieldCyclicGroupExtension(FiniteField(3,2),3)
```

```
trace(r)
```

$$0$$

```
                                                              Type: FiniteField(3,2)
```

```
GF3 := PrimeField 3;
```

```
                                                                Type: Domain
```

Polinom, soru işaretiyle gösterilen "isimsiz" bir değişkene sahiptir.

```
f := createPrimitivePoly(4)$FFPOLY(GF3)
```

$$?^4+?+2$$

```
                                  Type: SparseUnivariatePolynomial PrimeField 3
```

```
GF81 := FFCGP(GF3,f);
```

```
                                                                Type: Domain
```

Aşağıda bu cismin rastgele bir elemanı oluşturulmuştur:

```
random($GF81)
```

$$\%K^{13}$$

```
     Type: FiniteFieldCyclicGroupExtensionByPolynomial(PrimeField 3,?**4+?+2)
```

Sonlu Cisimlerde Dönüşüm İşlemleri

K bir sonlu cisim olsun.

```
K := PrimeField 3
```

PrimeField 3

```
                                                                Type : Domain
```

Eğer m, n nin bir böleni ise, K üzerindeki m dereceli cisim genişlemesi K_m, K üzerindeki n dereceli cisim genişlemesi K_n nin alt cismidir.

"FiniteFieldHomomorphisms" bölgesi, bir sabit sonlu taban cisminin farklı genişlemeleri arasında ve bu sonlu cisimlerin farklı temsilleri arasında dönüşüm işlemleri yapılmasını sağlar. m ve n gibi iki eleman alınsın,

```
(m,n) := (4,8)
```

8

```
                                                          Type : PositiveInteger
```

cisim genişlemesi oluşturulsun,

```
Km :=FiniteFieldExtension(K,m)
```

FiniteFieldExtension(PrimeField 3,4)

```
                                                    Type : Domain
```

ve daha küçük bir cisimden iki tane rastgele eleman alınsın.

```
Kn := FiniteFieldExtension(K,n)
```

FiniteFieldExtension(PrimeField 3,8)

```
                                                    Type : Domain
```

```
a1 := random()$Km
```

$$2\%A^3 + \%A^2$$

```
                                    Type : FiniteFieldExtension(PrimeField 3,4)
```

```
b1 := random()$Km
```

$$\%A^3 + \%A^2 + 2\%A + 1$$

```
                                    Type : FiniteFieldExtension(PrimeField 3,4)
```

m, n yi böldüğünden, K_m K_n nin alt cismidir.

```
a2 := a1 :: Kn
```

$$\%B^4$$

```
                                    Type : FiniteFieldExtension(PrimeField 3,8)
```

Böylece, K_m nin elemanları K_n nin elemanları cinsinden yazılabilir.

```
b2 := b1 :: Kn
```

$$2\%B^6 + 2\%B^4 + \%B^2 + 1$$

```
                                    Type : FiniteFieldExtension(PrimeField 3,8)
```

Aşağıda bu cisimlerin elemanları üzerinde bazı işlemler yapılmıştır.

```
a1+b1 - ((a2+b2) :: Km)
```

$$0$$

```
                                    Type : FiniteFieldExtension(PrimeField 3,4)
```

```
a1*b1 - ((a2*b2) :: Km)
```

$$0$$

```
                                                    Type : FiniteFieldExtension(PrimeField 3,4)
```

K_m ve K_n nin diğer şekillerde temsil edilmeleri durumunda farklı dönüştürmeler mevcuttur. Örneğin, K_m devirli çarpımsal grup ve K_n normal taban olsun.

```
Km := FFCGX(K,m)
```

FiniteFieldCyclicGroupExtension(PrimeField 3,4)

```
                                                                                    Type : Domain
```

```
Kn := FFNBX(K,n)
```

FiniteFieldNormalBasisExtension(PrimeField 3,8)

```
                                                                                    Type : Domain
```

```
(a1,b1) := (random()$Km,random()$Km)
```

$$\%C^{13}$$

```
                                                Type : FiniteFieldCyclicGroupExtension(PrimeField 3,4)
```

```
a2 := a1 :: Kn
```

$$2\%D^{q^6} + 2\%D^{q^5} + 2\%D^{q^4} + 2\%D^{q^2} + 2\%D^q + 2\%D$$

```
                                                Type : FiniteFieldNormalBasisExtension(PrimeField 3,8)
```

```
b2 := b1 :: Kn
```

$$2\%D^{q^7} + \%D^{q^6} + \%D^{q^5} + \%D^{q^4} + 2\%D^{q^3} + \%D^{q^2} + \%D^q + \%D$$

```
                                                Type : FiniteFieldNormalBasisExtension(PrimeField 3,8)
```

İşlemleri kontrol edelim.

```
a1+b1 - ((a2+b2) :: Km)
```

$$0$$

```
                                                Type : FiniteFieldCyclicGroupExtension(PrimeField 3,4)
```

```
a1*b1 - ((a2*b2) :: Km)
```

$$0$$

```
                                    Type : FiniteFieldCyclicGroupExtension(PrimeField 3,4)
```

Sonlu Cisimlerde Bazı İşlemler

"FiniteFieldPolynomialPackage" (kısaca FFPOLY), sonlu cisimler üzerinde genelleme, sayma ve test etme gibi işlemlerin yapılmasını sağlar. Birçok genelleştirilmiş polinomda bir tane belirsiz değişken vardır. Bu belirsizlik soru işaretiyle (?) belirtilir.

Aşağıdaki örnekte beş elemanlı bir cisim kullanılmıştır.

```
GF5 := PF 5
```

```
                                                                    Type : Domain
```

`createIrreduciblePoly` komutu kullanılarak herhangi pozitif dereceli indirgenemez polinomlar genelleştirilebilir.

```
f := createIrreduciblePoly(8)$FFPOLY(GF5)
```

$$?^8+?^4+2$$

```
                                    Type : SparseUnivariatePolynomial PrimeField 5
```

Aşağıdaki örneklerde bu polinomun bazı özellikleri incelenmiştir. "primitive?" fonksiyonu ile polinomun ilkel olup olmadığı test edilir.

```
primitve?(f)$FFPOLY(GF5)
```

```
                                                false
```

```
                                                                    Type : Boolean
```

"normal?" ile polinomun normal olup olmadığı test edilir.

```
normal?(f)$FFPOLY(GF5)
```

```
                                                false
```

```
                                                                    Type : Boolean
```

GF5 üzerinde, 8. dereceden bir ilkel polinom elde etmek için aşağıdaki yazılır:

```
p := createPrimitivePoly(8)$FFPOLY(GF5)
```

$$?^8+?^3+?^2+?+2$$

```
                                    Type : SparseUnivariatePolynomial PrimeField 5

primitive?(p)$FFPOLY(GF5)

                                            true

                                                                    Type : Boolean
```

Bu polinom normal değildir:

```
normal?(p)$FFPOLY(GF5)

                                            false

                                                                    Type : Boolean
```

Ancak normal olan bir polinom isteniyorsa aşağıdakini yazmak gereklidir:

```
n := createNormalPoly(8)$FFPOLY(GF5)
```

$$?^8 + 4?^7 + ?^3 + 1$$

```
                                    Type : SparseUnivariatePolynomial PrimeFİeld 5
```

Bu polinom ilkel değildir:

```
primitive?(n)$FFPOLY(GF5)

                                            false

                                                                    Type : Boolean
```

Hem ilkel hem de normal bir polinoma ihtiyaç varsa "createPrimitiveNormalPoly" veya "createNormalPrimitivePoly" fonksiyonları kullanılır.

```
createPrimitiveNormalPoly(8)$FFPOLY(GF5)
```

$$?^8 + 4?^7 + 2?^5 + 2$$

```
                                    Type : SparseUnivariatePolynomial PrimeField 5

GF5 := PF 5;

                                                                    Type : Domain

h := monomial(1,8)$SUP(GF5)
```

$$?^8$$

```
                                    Type : SparseUnivariatePolynomial PrimeField 5
```

Genelleştirmek için şu ifade kullanılır:

```
nh := nextIrreduciblePoly(h)$FFPOLY(GF5)
```

$$?^8 + 2$$

```
                                Type : Union(SparseUnivariatePolynomial PrimeField 5,...)
```

Bu polinom, "createIrreduciblePoly" ile üretilen polinomla aynı değildir.

```
createIrreduciblePoly(3)$FFPOLY(GF5)
```

$$?^3 + ? + 1$$

```
                                    Type : SparseUnivariatePolynomial PrimeField 5
```

Her seferinde aynı ifadeyi yazarak adım adım 8 inci dereceden 5 elemanlı indirgenemez polinomlara ulaşılablilir.

```
nh := nextIrreduciblePoly(nh)$FFPOLY(GF5)
```

$$?^8 + 3$$

```
                                Type : Union(SparseUnivariatePolynomial PrimeField 5,...)
```

Bu şekildeki tüm indirgenemez polinomların sayısı öğrenilebilir:

```
numberOfIrreduciblePoly(5)$FFPOLY(GF5)
```

$$624$$

```
                                                        Type : PositiveInteger
```

```
m := monomial(1,1)$SUP(GF5)
```

$$?$$

```
                                    Type : SparseUnivariatePolynomial PrimeField 5
```

```
f := m**3 + 4*m**2 + m + 2
```

$$?^3 + 4?^2 + ? + 2$$

```
                                    Type : SparseUnivariatePolynomial PrimeField 5
```

```
f1 := nextPrimitivePoly(f)$FFPOLY(GF5)
```

$$?^3 + 4?^2 + 4? + 2$$

```
                                Type : Union(SparseUnivariatePolynomial PrimeField 5,...)
```

```
nextPrimitivePoly(f1)$FFPOLY(GF5)
```

$$?^3 + 2?^2 + 3$$

```
                                Type : Union(SparseUnivariatePolynomial PrimeField 5,...)
```

Axiom ile verilen bir dereceden herhangi bir polinom oluşturulabilir;

```
random(5)$FFPOLY(GF16)
```

$$?^5 + (\%F\%G + 1)?^4 + \%F\%G?^3 + (\%G + \%F + 1)?^2 + ((\%F + 1)\%G + \%F)? + 1$$

```
                    Type : SparseUnivariatePolynomialFiniteFieldExtension
                                (FiniteFieldExtension(PrimeField 2,2),2)
```

veya verilen iki sınır arasındaki bir dereceden polinom oluşturulabilir.

```
random(3,9)$FFPOLY(GF16)
```

$$?^3 + (\%F\%G + 1)?^2 + (\%G + \%F + 1)? + 1$$

```
                    Type : SparseUnivariatePolynomialFiniteFieldExtension
                                (FiniteFieldExtension(PrimeField 2,2),2)
```

3.4. Simetrik Gruplar

Bu bölümde simetrik, alterne ve dihedral grupların elemanlarının Axiom ile nasıl gösterildiği incelenmiştir. Permütasyon gruplarının devir indeksleri (cycle index), permütasyonun devirlerinin boyutunu belirler. Örneğin, $(3^2 \quad 2 \quad 1^2)$ şeklindeki bir ifade, $(s_3^2 \quad s_2 \quad s_1^2)$ devirini temsil eder ve şu anlama gelir: 3 uzunluğunda iki adet devir, 2 uzunluğunda bir adet devir ve 1 uzunluğunda iki adet devir. Bir permütasyon grubunun devir indeksi, permütasyon sayıları ile bölünmüş permütasyonların devir indisleridir.

"complete" fonksiyonu ile indisi n olan simetrik grup bulunur. Buna göre S_3 simetrik grubu aşağıdaki gibidir:

```
complete 3
```

$$\frac{1}{3}(3) + \frac{1}{2}(2 \quad 1) + \frac{1}{6}(1^3)$$

```
                                Type : SymmetricPolynomial(Fraction(Integer))
```

S_3 simetrik grubunun elemanları

$$S_3 = \{I, (1 \quad 2 \quad 3), (1 \quad 3 \quad 2), (2 \quad 3), (1 \quad 2), (1 \quad 3)\}$$

şeklinde ayrık devirlerin çarpımı halinde gösterilebilir. Burada;

$$\frac{1}{3}(3) + \frac{1}{2}(2 \quad 1) + \frac{1}{6}(1^3)$$

ifadesine karşılık gelen elemanlar aşağıdaki tabloda gösterilmiştir.

(3)	(1 2 3), (1 3 2)
(2 1)	(2 3), (1 2), (1 3)
(1 3)	I=(1)(2)(3)

S_3 grubunun eleman sayısı altı olduğundan, $\frac{1}{3}$ katsayısı, 3 uzunluğunda devire sahip iki eleman olduğunu; $\frac{1}{2}$ katsayısı, (2 1) şeklinde üç tane eleman olduğunu ve $\frac{1}{6}$ katsayısı ise (1^3) şeklinde bir tane eleman olduğunu belirtir.

Benzer şekilde S_4 simetrik grubu aşağıdaki gibidir:

```
complete 4
```

$$\frac{1}{4}(4) + \frac{1}{3}(3 \quad 1) + \frac{1}{8}(2^2) + \frac{1}{4}(2 \quad 1^2) + \frac{1}{24}(1^4)$$

```
Type : SymmetricPolynomial(Fraction(Integer))
```

Yukarıdaki ifadeye karşılık gelen elemanlar aşağıdaki tabloda gösterilmiştir.

(4)	(1 2 3 4), (1 2 4 3), (1 3 2 4),
	(1 3 4 2), (2 3 1 4), (2 3 4 1)
(3 1)	(1 2 3), (1 3 2), (1 2 4), (1 4 2),
	(1 3 4), (1 4 3), (2 3 4), (2 4 3)
(2^2)	(1 2)(3 4), (1 3)(2 4), (1 4)(2 3)
$(2 \quad 1^2)$	(1 2), (1 3), (1 4), (2 3), (2 4), (3 4)
(1^4)	I=(1)(2)(3)(4)

Aşağıdaki örnek ile, 40320 elemanlı S_8 simetrik grubunun devirlerinin nasıl ve kaç tane olduğuna dair yorum yapılabilmesi kolaylaşmış olur.

```
complete 8
```

$$\frac{1}{8}(8) + \frac{1}{7}(7 \quad 1) + \frac{1}{12}(6 \quad 2) + \frac{1}{12}(6 \quad 1^2) + \frac{1}{15}(5 \quad 3) + \frac{1}{10}(5 \quad 2 \quad 1) + \frac{1}{30}(5 \quad 1^3) +$$

$$\frac{1}{32}(4^2) + \frac{1}{12}(4 \quad 3 \quad 1) + \frac{1}{32}(4 \quad 2^2) + \frac{1}{16}(4 \quad 2 \quad 1^2) + \frac{1}{96}(4 \quad 1^4) + \frac{1}{36}(3^2 \quad 2) +$$

$$\frac{1}{36}(3^2 \quad 1^2) + \frac{1}{24}(3 \quad 2^2 \quad 1) + \frac{1}{36}(3 \quad 2 \quad 1^3) + \frac{1}{360}(3 \quad 1^5) + \frac{1}{384}(2^4) +$$
$$\frac{1}{96}(2^3 \quad 1^2) + \frac{1}{192}(2^2 \quad 1^4) + \frac{1}{1440}(2 \quad 1^6) + \frac{1}{40320}(1^8)$$

`Type : SymmetricPolynomial(Fraction(Integer))`

A_n alterne grubu ile ilgili bazı örnekler aşağıdaki gibidir.

3 elemanlı $A_3 = \{I, (1 \quad 2 \quad 3), (1 \quad 3 \quad 2)\}$ alterne grubu Axiom'da şu şekilde gösterilir:

`alternating 3`

$$\frac{2}{3}(3) + \frac{1}{3}(1^3)$$

`Type : SymmetricPolynomial(Fraction(Integer))`

Benzer şekilde A_4 alterne grubu,

`alternating 4`

$$\frac{2}{3}(3 \quad 1) + \frac{1}{4}(2^2) + \frac{1}{12}(1^4)$$

`Type : SymmetricPolynomial(Fraction(Integer))`

şeklinde olup, sekiz tane 3 uzunluğunda bir tane ve 1 uzunluğunda bir tane ayrık devirin çarpımından; üç tane 2 uzunluğunda iki adet ayrık devirin çarpımından ve bir tane 1 uzunluğundaki dört tane ayrık devirin çarpımından oluştuğu söylenebilir.

A_7 alterne grubunun elemanları hakkında yorum yapabilmek için,

`alternating 7`

$$\frac{2}{7}(7)+\frac{1}{5}(5 \quad 1^2)+\frac{1}{4}(4 \quad 2 \quad 1)+\frac{1}{9}(3^2 \quad 1)+\frac{1}{12}(3 \quad 2^2)+ \quad 136(3 \quad 1^4)+\frac{1}{24}(2^2 \quad 1^3)+\frac{1}{2520}(1^7)$$

`Type : SymmetricPolynomial(Fraction(Integer))`

sonucu incelenebilir.

$2n$ mertebeli D_n dihedral grupları ile ilgili örnekler aşağıdaki gibidir.

$$\mathbf{a} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$$

ve

$$\mathbf{b} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$$

elemanları için $D_3 = \{I, (1 \quad 2 \quad 3), (1 \quad 3 \quad 2), (1 \quad 2), (1 \quad 3), (3 \quad 3)\}$ şekinde olup

`dihedral 3`

$$\frac{1}{3}(3) + \frac{1}{2}(2 \quad 1) + \frac{1}{6}(1^3)$$

```
Type : SymmetricPolynomial(Fraction(Integer))
```

bulunur.

Aşağıdaki tabloda D_3 ün elemanlarının dağılımı görülmektedir.

(3)	(1 2 3),(1 3 2)
(2 1)	(1 2),(1 3),(2 3)
(1^3)	I=(1)(2)(3)

Aşağıdaki örnekte karenin simetrileri grubu olan D_4 incelenmiştir.

```
dihedral 4
```

$$\frac{1}{4}(4)+\frac{3}{8}(2^2)+\frac{1}{4},(2 \quad 1^2)+\frac{1}{8}(1^4)$$

```
Type : SymmetricPolynomial(Fraction(Integer))
```

burada

$$\mathbf{a}=\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$$

ve

$$\mathbf{b}=\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{pmatrix}$$

elemanları için

$$D_4=\{I,(1 \quad 2 \quad 3 \quad 4),(1 \quad 4 \quad 3 \quad 2),(1 \quad 3)(2 \quad 4),(1 \quad 4)(2 \quad 3),(1 \quad 2)(3 \quad 4),(1 \quad 3),(2 \quad 4)\}$$

olup elemanlar aşağıdaki tabloda görüldüğü gibidir.

(4)	(1 2 3 4),(1 4 3 2)
(2^2)	(1 3)(2 4),(1 2)(3 4),(1 4)(2 3)
$(2 \quad 1^2)$	(1 3),(2 4)
(1^4)	I=(1)(2)(3)(4)

4. SİMPLİSEL GRUPLAR

Simplisel Grupların Axiom'daki uygulamaları, 1992 yılında Ronald Brown ve Andrew Tonks'un beraber yayınladıkları "Calculations with Simplicial and Cubical Groups in Axiom" adlı çalışmada yapılmıştır. Bu çalışmada Simplisel Gruplar ile ilgili işlemleri kullanmak için bir paket tanımlanmıştır. Aşağıda, ilgili çalışmada yer alan Simplisel Operatörler paketi verilmiştir.

```
++ " Simplicial Operators"
++
++ Author: Larry A.Lambe
++ Date  : 01/26/91
++
++ Modified : Andy Tonks
++ Date     : August 1992
)abbrev domain SOP SimplicialOperators
SimplicialOperators : Exports == Implementation where
I ==> Integer
L ==> List
NNI ==> NonNegativeInteger
O ==> OutputForm
left ==> 1
right ==> 2
Exports ==> Join(OrderedSet, Monoid) with
s : (I,$)  -> $
++ s(i,x) returns the ith degeneracy operator composed with x
s : I  -> $
++ s(i) gives the ith degeneracy operator
d : (I,$)  -> $
++ d(i,x) returns the composite of the ith face operator with x
d : I  -> $
++ d(i) gives the ith face operator
deriv : $  -> $
++ deriv(x) returns the derivative of the simplicial map x
Implementation ==> add
```

```
-- representation
Rep := L L I
-- declarations
x,y :$
-- define
x = y == x =$Rep y
x < y == x <$Rep y
1     == [[],[]]
-- s assumes that its argument is in normal form
s(i,xx) ==
-- non destructive version
x := [copy xx.left,copy xx.right]
null x.left => [[i]$(L I),x.right]
dum:L I
k := x.left.first
while i <= k repeat
dum := cons(k+1,dum)
x.left := x.left.rest
if x.left = [] then leave
k := x.left.first
dum:= reverse cons(i,dum)
x.left := append(dum,x.left)
x
--d assumes that its argument is in normal form
d(i,xx)==
-- non destructive version
x := [copy xx.left, copy xx.right]
absorbed:I :=0
middle:I :=0
dum:(L I) := []
while not null x.left repeat
k := x.left.first
x.left := x.left.rest
```

```
if i<k then dum := cons (k-1,dum)
else if (i>k+1) then
i:=i-1
dum := cons(k,dum)
else
absorbed := 1
leave
dum := reverse dum
x.left := append(dum,x.left)
absorbed = 1 => x
dum := []
while not null x.right repeat
k := x.right.first
x.right := x.right.rest
if i >=k then (dum := cons(k,dum);
i:= i+1)
else (dum := reverse dum;
middle := 1;
leave)
middle = 1 =>
x.right := cons(k,x.right)
x.right := append(dum,cons(i,x.right))
x
dum := reverse cons(i,dum)
x.right := append(dum,x.right)
x
s(i::I) == [[i],[]]
d(i::I) == [[],[i]]
x * y ==
x = 1 => y
y = 1 => x
dum:Rep := [[ss for ss in y.left],[dd for dd in y.right]]
for j in reverse x.right repeat
```

```
dum := d(j,dum)
for i in reverse x.left repeat
dum := s(i,dum)
dum
deriv x ==
[[i+1 for i in x.left],[j+1 for j in x.right]]
coerce(x):O==
x = 1 => id :: O
hconcat(hconcat([sub(s::O,j::O) for j in x.left]),_
hconcat([sub(d::O,j::O) for j in x.right]))
```

Aşağıda, makalede kullanılan bazı örneklere yer verilmiştir.

```
x: SimplicialOperators
                                                    Type : Void
x:=s(3)
```

$$s_3$$

```
                                          Type : SimplicialOperators
```

Yukarıdaki örneklerde SimplicialOperators bölgesinden bir x elemanı alınmış ve s(3) değerine atanmıştır. "s(3)" değeri 3. dejenere operatördür.

```
x**5
```

$$s_7 s_6 s_5 s_4 s_3$$

```
                                          Type : SimplicialOperators
s(1)*x*d(2)*x
```

$$s_4 s_3 s_1 d_2$$

```
                                          Type : SimplicialOperators
```

"d(2)" değeri 2. yüz operatörüdür.

```
deriv(%)
```

$$s_5 s_4 s_2 d_3$$

```
                                          Type : SimplicialOperators
x<s(1)
```

false

```
                                                                Type : Boolean
```

```
[d(i)*x for i in 1..5]
```

$$[s_2d_1, s_2d_2, id, id, s_3d_4]$$

```
                                                  Type : List SimplicialOperators
```

Axiom'da bir paket tanımlamak için, kodların SPAD diliyle yazılmış olması gerekmektedir. ".spad" uzantılı paket dosyası ")compile dosyaAdı.spad" komutuyla açılarak, ilgili işlemler yapılabilir. Simplisel Operatörler Paketi SPAD diliyle yazılmıştır.

5. AXIOM SİSTEM KOMUTLARI

)close

Bu komut ile terminal ortamından çıkmadan HyperDoc ortamının kapatılması sağlanır. Böylece terminal ortamında yazılan veriler kaybedilmemiş olur. Bu komut ile çıkış yapmak istendiğinde Axiom şu uyarıyı verir:

```
This is the last Axiom session. Do you want to kill Axiom?
```

Eğer kapatmak isteniyorsa klavyeden "y" harfine, kapatmak istenmiyorsa "n" harfine basmak yeterlidir.

)clear all

Bu komut ile çalışma alanındaki fonksiyonlar, tanımlamalar, değişken atamaları ve değişkenlerin hepsi silinir. Çalışma alanını boşaltmak ve adım sayıcıyı 1 den başlatmak için

```
)clear all
```

komutu kullanılır. Çalışma alanındaki herşeyin silinmesi ancak adım sayıcının kaldığı yerden devam etmesi isteniyorsa

```
)clear properties all
```

kullanılır. "x" elemanına dair tüm verilerin silinmesi için

```
)clear properties x
```

kullanılır. "x, y ve f" eleamanlarına dair tüm verilerin silinmesi için

```
)clear properties x y f
```

kullanılır. Özellikler (properties) için "p" yazmak yeterlidir.

```
)clear p all
```

```
)clear p x y f
```

)display

Bu komut çalışma ortamındaki içeriğin görüntülenmesi için kullanılır.

```
)display all
```

komutu ile Axiom sisteminde önceden tanımlanmış olan makroların görüntülenmesi sağlanır.

```
)display operations
```

veya kısaca

```
)display op
```

komutuyla Axiomda kullanılabilecek tüm fonksiyonların görüntülenmesi sağlanır.

`)display op` *opName*

komutu ile belirli bir fonksiyonun özellikleri görüntülenir.

)help

Bu komut sistem komutlarıyla ilgili yardım sunar.

`)help` *commandName*

komutu ile herhangi bir sistem komutu hakkında yardım alınır.

)quit

Bu komut Axiom'u sonlandırmak ve işletim sistemine dönmek için kullanılır.

```
)set quit unprotected
```

komutu ile terminalde yazılmış olan verilerin kaydedilmeden Axiom'un kapatılması sağlanır.

```
)set quit protected
```

komutu ile terminalde yazılmış olan verilerin kaydedilip ardından Axiom'un kapatılması sağlanır.

)set

`)set` komutu sistem değişkenlerini görmek veya oluşturmak için kullanılır. Bu komutun kullanılabileceği tüm alanları görmek için `)set` yazmak yeterlidir.

)show

Bu komut Axiom bölge, paket ve kategoriler hakkındaki bilgileri gösterir. Herhangi bir seçenek verilmemişse, `)operations` seçeneği verilmiş sayılır. Örneğin;

```
)show POLY
```

```
)show POLY )operations
```

```
)show Polynomial
```

```
)show Polynomial )operations
```

yukarıdaki komutların her biri Polinomlar hakkındaki temel bilgileri gösterir ve sonrasında işlemlerin listesini verir. Polinomlar değişken olarak bir Halkaya (örneğin, Tamsayılar (Integer)) ihtiyaç duyduklarından yukarıdaki tüm komutlar belirlenmemiş bir R halkası üzerinde tanımlanır.

)undo

Bu komut terminal ortamında, kullanıcının önceden yaptığı işlemlerde değişiklik yapmak için kullanılır. `)undo` önceden yazılmış olan herhangi bir adım sayısını değişken olarak alabilir.

```
)undo n
```

```
)undo n )after
```

Bu komutlar n - inci adımdaki işleme geri dönülmesini sağlar. n pozitif bir tamsayı ise, adım sayısını gösterir. n negatif bir tamsayı ise, n önceki komut anlamına gelir.

KAYNAKLAR DİZİNİ

Brown, R., Tonks, A.,(1992), Calculations with Simplicial and Cubical Groups in AXIOM, Journal of Symbolic Computation (1994) 17, s.159-170.

Jenks, R.D., ve Sutor, R.S.,(1991), AXIOM: the Scientific Computation System, http://www.axiom-developer.org/axiom-website/bookvol0.pdf

http://www.axiom-developer.org/axiom-website/bookvol1.pdf

http://www.axiom-developer.org/axiom-website/bookvol2.pdf.

http://www.axiom-developer.org/axiom-website/bookvol10.2.pdf.

http://www.axiom-developer.org/axiom-website/bookvol10.4.pdf.

Li, X., Maza, M. M.(2006), Efficient Implementation of Polynomial Arithmetic in a Multiple-Level Programming Environment, Mathematical Software - ICMS 2006, Springer Berlin Heidelberg, s.12-23.

Smith, J., Dos Reis, G., Järvi, J. (2007), Algorithmic Differentiation in Axiom, ISSAC'07, s.347-354.