KARAMANOĞLU MEHMETBEY ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GAP (GRUP, ALGORİTMA VE PROGRAMLAMA)VE KNUTH-BENDIX ALGORİTMASI İLE İLGİLİ UYGULAMALAR

YÜKSEK LİSANS

Yasemin SAĞIR

Matematik Anabilim Dalı
Cebir ve Sayılar Teorisi Programı

HAZİRAN 2022

KARAMANOĞLU MEHMETBEY ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GAP (GRUP, ALGORİTMA VE PROGRAMLAMA) VE KNUTH-BENDIX ALGORİTMASI İLE İLGİLİ UYGULAMALAR

YÜKSEK LİSANS TEZİ

Yasemin SAĞIR
(190801055)

Matematik Anabilim Dalı
Cebir ve Sayılar Teorisi Programı

Tez Danışmanı: Prof. Dr. Eylem GÜZEL KARPUZ

HAZİRAN 2022

TEZ ONAYI

Yasemin SAĞIR tarafından hazırlanan GAP (Grup, Algoritma ve Programlama) ve Knuth-Bendix Algoritması ile İlgili Uygulamalar adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Karamanoğlu Mehmetbey Üniversitesi Fen Bilimleri Enstitüsü Matematik Ana Bilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman:
Prof. Dr. Eylem GÜZEL KARPUZ

Jüri Üyeleri İmza:

Prof. Dr. Ahmet Sinan ÇEVİK

Prof. Dr. Ahmet İPEK

Prof. Dr. Eylem GÜZEL KARPUZ

Tez Savunma Tarihi: 10/ 06/ 2022

Yukarıdaki sonucu onaylarım

Doç. Dr. Ahmet KAYABAŞI
Enstitü Müdürü

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

(İmza)
Yasemin SAĞIR

ÖNSÖZ

Çalışma sürecinde her türlü yol gösterici olan, desteğini ve engin bilgilerini benden esirgemeyen danışmanım Prof. Dr. Eylem GÜZEL KARPUZ'a engin teşekkürlerimi sunarım.

Tüm hayatım boyunca benim yanımda olan, aldığım kararları her zaman destekleyen, cesaretlendiren, moral veren annem Sevim SÖNMEZ'e, babam Mustafa SÖNMEZ'e ve sevgiyle benden desteğini esirgemeyen eşim Sinan SAĞIR'a ve çocuklarım Atakan SAĞIR ve Yunus Emre SAĞIR'a teşekkür ederim.

Haziran 2022

Yasemin SAĞIR
(Matematik Öğretmeni)

İÇİNDEKİLER

Sayfa

KISALTMALAR

GAP **:** Grup, Algoritma ve Programlama

SEMBOLLER

$\iota(w)$	: w kelimesinin başlangıç harfi
$\tau(w)$	: w kelimesinin bitiş harfi
1_w	: Boş kelime
$l(w)$	: w kelimesinin uzunluğu
$\approx$	: Serbest olarak iki kelimenin denkliği
$[w]$	: w kelimesinin denklik sınıfı
$F(X)$	: X kümesi üzerindeki serbest grup
$\langle X ; R \rangle$	: Grup sunuşu
$w_1 \approx_{\wp} w_2$	: w_1 ve w_2 kelimeleri $\wp$ sunuşuna bağlı olarak denktir
$[w]_{\wp}$	: $\wp$ sunuşuna bağlı olarak w kelimesinin denklik sınıfları
$[1]_{\wp}$	: $\wp$ sunuşuna bağlı grubun birimi
$G(\wp)$	: $\wp$ sunuşunun temsil ettiği grup
$\wp_M$	: M monoidin sunuşu
A^+	: A kümesindeki elemanlarla oluşturulan en az bir uzunluklu kelimelerin kümesi
A^*	: $A^+ \cup \{1\}$
$\rightarrow_R^*$	: R tarafından üretilen bir indirgeme bağıntısı
$\leftrightarrow_R^*$	: R tarafından üretilen Thue kongrüans
$H(\lambda_q)$	: Hecke grubu
$\overline{H}(\lambda_q)$	: Genişletilmiş Hecke grubu

ÖZET

Yüksek Lisans

GAP (GRUP, ALGORİTMA VE PROGRAMLAMA) VE KNUTH-BENDIX ALGORİTMASI İLE İLGİLİ UYGULAMALAR

Yasemin SAĞIR

Karamanoğlu Mehmetbey Üniversitesi
Fen Bilimleri Enstitüsü
Matematik Ana Bilim Dalı

Danışman: Prof. Dr. Eylem GÜZEL KARPUZ

Haziran, 2022, 66 sayfa

Bu tez yedi bölümden oluşmaktadır. İlk bölümde; tezin genel amacından bahsedilmiştir. İkinci bölümde; sunuş kavramı, genelleştirilmiş Hecke grubu ve karar verme problemleri hakkında bilgiler verilmiştir.

Üçüncü bölümde; kelime probleminin çözümünde önemli bir metot olan yeniden yazma sisteminden bahsedilmiştir.

Dördüncü bölümde; Hesaplamalı Grup Teorisi üzerine çalışmalar yapan ve özellikle Hesaplamalı Ayrık Cebirde kullanılan bir sistem olan Grup, Algoritma ve Programlama (GAP) ile ilgili genel bilgiler verilmiştir. Daha sonra ise, GAP dili hakkında kısa bilgiler sunulmuştur.

Beşinci bölümde; bir GAP paket programı olan *idrel* incelenmiştir ve bu paket program fonksiyonlarının uygulamaları verilmiştir. Daha sonra bu paketin uygulamaları olarak genişletilmiş Hecke grubunun monoid sunuşları incelenmiştir. Son olarak ise, genişletilmiş ve genelleştirilmiş Hecke grubunun monoid sunuşu için kelime probleminin çözülebilirliği verilmiştir.

Altıncı bölümde; *idrel* paketi kapsamında çalışan kayıtlı yeniden yazma sistemi, kayıtlı Knuth-Bendix hesaplaması ve bir kelimenin kayıtlı indirgenmesi ile ilgili fonksiyonlar incelenmiştir. Bu fonksiyonların $\mathbb{Z}_3 \times \mathbb{Z}_4$ direkt çarpım grubu üzerine uygulamaları yapılmıştır.

Son bölümde, diğer bölümlerdeki sonuçların bir değerlendirilmesi yapılmıştır.

Anahtar Kelimeler: Sunuş, yeniden yazma sistemi, kelime problemi, genişletilmiş Hecke grubu, grup, algoritma ve programlama

ABSTRACT

MsThesis

GAP (GROUP, ALGORITHM AND PROGRAMMING) AND APPLICATIONS WITH KNUTH-BENDIX ALGORITHM

Yasemin SAĞIR

Karamanoğlu Mehmetbey University
Graduate School of Natural andApplied Sciences
Department of Mathematics

Supervisor: Prof. Dr. Eylem GÜZEL KARPUZ

June, 2022, 66 pages

This thesis consists of seven chapters. In the first part; the general purpose of the thesis is mentioned. In the second part; it has been given some informations about presentation, generalized Hecke group and decision problems.

In the third part; the rewriting system, which is an important method in solving the word problem, is mentioned.

In the fourth part; general information about Group, Algorithm and Programming (GAP) which is a system that works on Computational Group Theory and is especially used in Computational Discrete Algebra, is given. Then, brief information about the GAP language is presented.

In the fifth part; *idrel*, a GAP package program, has been examined and applications of functions of this package have been given. Then, as applications of this package, the monoid presentations of the extended Hecke group has been examined. Finally, the solvability of the word problem for the monoid representation of the extended and generalized Hecke group was given.

In the sixth part; functions related to the logged rewriting system, the logged Knuth-Bendix calculation and the logged reduction of a word, working within the scope of the *idrel* package, are examined. These functions are applied on $\mathbb{Z}_3 \times \mathbb{Z}_4$ direct product of groups.

In the last part; an evaluation of the results obtained in the previous sections is made.

Key Words: Presentation, rewriting system, word problem, generalized Hecke group, group, algorithm and programming

1. GİRİŞ

Birleştirilmiş grup teorisinin temelini serbest gruplar, üreteç ve bağıntılar ile elde edilen sunuş kavramı oluşturmaktadır. Sunuş kavramı sadece cebirde değil matematiğin birçok alanında özellikle de geometrik topolojide çalışma alanı oluşturmaktadır. Bir grubun sunuşu, grubun yapısı ve birçok cebirsel özellikleri hakkında bilgi vermektedir. Bu sunuş kavramını kullanarak grubun kelime probleminin çözülebilirliği hakkında bilgi edinilebilir. Kelime problemi, 1911 yılında M. Dehn tarafından literatüre kazandırılan üç temel karar verme problemlerinden birisidir. Bu problem; üzerinde çalışılan grup, monoid ya da yarıgruptan alınan iki kelimenin birbirine eşit olup olmadığını araştıran bir metodun ya da algoritmanın varlığı problemidir. Kelime probleminin matematikteki hangi cebirsel yapılar için çözülebilir, hangileri için çözülemez olduğu yönündeki çalışmalar bu yapıların sınıflandırılması açısından oldukça önemlidir.

Grup, Algoritma ve Programlama (GAP); özellikle Hesaplamalı Grup Teorisi üzerine çalışmalar yapan, hesaplamalı ayrık cebirde kullanılan bir sistemdir. GAP, bir programlama dilidir. GAP, bu dilde yazılmış cebirsel algoritmaları uygulayan binlerce fonksiyondan oluşan bir kitaplığın yanı sıra cebirsel nesnelerin büyük veri kitaplıklarını sağlar. GAP, grupları ve temsillerini, halkaları, vektör uzaylarını, cebirleri, kombinatoryal yapıları ve daha fazlasını incelemek için araştırma ve öğretimde kullanılır. Özel kullanımınız için çalışabilir, kolayca değiştirebilir veya genişletebilirsiniz. GAP, C ile yazılmış bir çekirdeğe sahiptir. Bu programla yazılan paket programlar, ilgili alanda çalışan matematikçilerin elde ettikleri sonuçlar için bir uygulama alanı sağlamaktadır. Bu program, *www.gap-system.org* internet adresinden bilgisayarlara kurulabilmektedir.

Bu tez çalışmasında, cebirde önemli grup yapılarından olan sonlu iki devirli grubun direkt çarpım grubunun ve genişletilmiş Hecke grubunun monoid sunuşları incelenmiş, bu yapıların tam yeniden yazma sistemleri üreteçler arasında uygun sıralamalar seçilerek elde edilmiştir. Elde edilen bu sonuçların bir GAP paketi olan *idrel* kullanılarak uygulamaları yapılmıştır. Ayrıca genişletilmiş ve genelleştirilmiş

Hecke grubunun monoid sunuşu için kelime probleminin çözülebilirliği verilmiştir. Son olarak ise, bir sunuşun yeniden yazma sisteminin elde edilmesi aşamasında bulunan kuralları kaydeden kayıtlı yeniden yazma sistemi incelenmiştir ve sonlu iki devirli grubun direkt çarpım grubunun sunuşu için uygulamalar yapılmıştır.

2. TEMEL KAVRAMLAR VE BİLGİLER

2.1 Giriş

Bu bölümde kombinatoryal grup teorisinin temelini oluşturan serbest grup ve grup sunuşları verilmiştir. Ayrıca genişletilmiş Hecke grubu, genişletilmiş ve genelleştirilmiş Hecke gruplarının sunuşları ve karar verme problemleri üzerine kısa bilgi sunulmuştur. Bu bölümde verilen temel tanım ve kavramlar ile ilgili detaylı bilgilere (Cohen, 1989; Çetinalp, 2016; Karpuz, 2006; Lyndon ve Schupp, 1977; Magnus ve diğerleri, 1976; Rotman, 1984) kaynaklarından ulaşılabilir.

2.2 Sunuşlar

Üreteçler ve bağıntılar tarafından bir grubun en eski sunuşlarından biri, 1856'da İrlandalı matematikçi William Rowan Hamilton tarafından, ikosahedral grubun bir sunumu olan icosian hesabında verilmiştir. Bununla ilgili ilk sistematik çalışma, 1880'lerin başında Felix Klein'ın öğrencisi Walthervon Dyck tarafından kombinatoryal grup teorisinin temellerini atarak yapılmıştır.

Bazı özel problemlerin çözümlerinde de kullanılan sunuş (takdim) kavramı, birleştirilmiş grup ve yarı grup teorisinde önemli bir yere sahiptir. Cebirsel problemlerin çözüme ulaşmasına yardımcı olurlar. Sunuş kavramı, sadece cebirsel yapı hakkında bilgi vermek veya cebirsel yapının grubun genel bir karakterizasyonunu belirlemek için kullanılmazlar. Aynı zamanda birçok cebirsel problemin çözümünde bir araç olarak da kullanılabilmektedirler.

X boştan farklı bir küme olmak üzere; bu küme ile $x \leftrightarrow x^{-1}$ $(x \in X)$ eşleşmesinden faydalanarak X^{-1} kümesini tanımlayalım ve $X^{\pm} = X \cup X^{-1}$ olsun. Elde edilen $X^{\pm}$ kümesinin her bir elemanına *harf* denir. Harflerle elde edilen, $x_1^{\varepsilon_1} x_2^{\varepsilon_2} \cdots x_n^{\varepsilon_n}$ ($x_i \in X$, $\varepsilon_i = \pm 1$ ve $1 \le i \le n$ $(n \in \mathbb{N})$) şeklindeki ifadeye *kelime* denir. X kümesi üzerinde tanımlı bir kelime, $x_i^{\varepsilon_i} x_i^{-\varepsilon_i} (x_i \in X, \varepsilon_i = \pm 1)$ harf çiftini içermiyorsa bu kelimeye *indirgenmiş kelime* denir. Bu duruma ilave olarak, $x_1^{\varepsilon_1} x_2^{\varepsilon_2} \cdots x_n^{\varepsilon_n}$ gibi bir kelime için,

$x_1^{\varepsilon_1} \neq x_n^{-\varepsilon_n}$ eşitsizliği gerçekleşiyor ise bu kelime devirsel indirgenmiş kelime olarak adlandırılır.

X üreteç kümesi ve X kümesi üzerindeki devirsel indirgenmiş kelimelerden oluşan R bağıntı kümesi olsun. $\wp = \langle X ; R \rangle$ ikilisine bir *grup sunuşu* denir. Üreteç ve bağıntı kümelerinin her ikisinin de sonlu olması durumunda $\wp$ sunuşu sonludur denir.

Örnek 2.2.1:

- X üreteç kümesine sahip serbest grubun sunuşu $\langle X \ ; \varnothing \rangle$ biçimindedir.

- 3 mertebeli bir devirli grubun sunuşu $A = \langle a ; a^3 = 1 \rangle$ ve 4 mertebeli devirli bir grubun sunuşu $B = \langle b ; b^4 = 1 \rangle$ olmak üzere; bu grupların serbest çarpımının sunuşu $\langle a, b \ ; \ a^3 = 1, b^4 = 1 \rangle$ biçimindedir.
- $2n$ mertebeli bir dihedral grubun sunuşu $D_n = \langle a, b \ ; \ a^n = 1, b^2 = 1, (ab)^2 = 1 \rangle$ biçimindedir.
- Rankı 2 olan serbest abelyan grubun sunuşu $\mathbb{Z} \times \mathbb{Z} = \langle a, b \ ; \ ab = ba \rangle$ biçimindedir.
- Klein 4-grubunun sunuşu $V_4 \cong D_2 = \langle a, b \ ; \ a^2 = 1, b^2 = 1, (ab)^2 = 1 \rangle$ biçimindedir.
- Baumslag-Solitar grubunun sunuşu $BS(m,n) = \langle a, b \ ; \ a^n = ba^m b^{-1} \rangle$ biçimindedir.

2.3 Genişletilmiş Hecke Grubu, Genişletilmiş ve Genelleştirilmiş Hecke Grubu ve Sunuşları

Hecke grubu; Erich Hecke tarafından 1936 yılında literatüre kazandırılmıştır. Hecke, 1936 yılındaki çalışmasında; $\lambda \in \mathbb{Z}^+$ olmak üzere, $x(z) = -\frac{1}{z}$ ve $u(z) = z + \lambda_q$ kesirli doğrusal dönüşümleri ile üretilen ve $H(\lambda)$ ile ifade edilen grup yapıları tanımlamıştır. Burada $x.u$ alınırsa $y = x.u = -\frac{1}{z + \lambda_q}$ elde edilmektedir.

$H(\lambda_q)$ Hecke grubu, 2 ve q mertebeli devirli grupların serbest çarpımına izomorftur ve bu grubun sunuşu $H(\lambda_q)=\langle x,\ y\ ; x^2, y^q\rangle = C_2 * C_q$ ile gösterilir.

$H(\lambda_q)$ Hecke grup yapısında, $q=3$ değerine karşılık gelen $H(\lambda_3)$ Hecke grubu modüler grup olarak adlandırılır ve bu grup $PSL(2,\mathbb{Z})$ grubudur. $H(\lambda_3)$ modüler grubun üreteçlerine $r(z)=1/\bar{z}$ yansıma dönüşümü eklenerek $\overline{H}(\lambda_3)=\overline{M}$ genişletilmiş modüler grubu elde edilir. $\overline{H}(\lambda_q)$ genişletilmiş Hecke grubu $D_{2*\mathbb{Z}_2}D_q$ birleştirilmiş serbest çarpımına izmorftur (burada D_2 grubu 4 mertebeli, D_q grubu ise $2q$ mertebeli dihedral gruptur) ve bu grubun sunuşu

$$\overline{H}(\lambda_q)=\langle x,\ y,\ z,\ r\ ;\ x^2, y^q, z^2, r^2, (xr)^2, (yr)^2, (zr)^2\rangle$$

biçiminde gösterilir. Bu sunuşta $q=2$ alınırsa genişletilmiş Hecke grubunun sunuşu

$$\overline{H}(\lambda_2)=\langle x,\ y,\ z,\ r\ ;\ x^2, y^2, z^2, r^2, (xr)^2, (yr)^2, (zr)^2\rangle$$

ile ifade edilir.

Yukarıda verilen grup yapılarının genelleştirilmesi ve ilgili grupların komutatör alt grupları ile ilgili çalışmalar (Doğrayıcı ve Şahin, 2020; Huang, 1999) kaynaklarında bulunabilmektedir. Buna göre; $n\geq 2$ olmak üzere; $p_1, p_2, \ldots, p_n \in \mathbb{Z}$ için, genelleştirilmiş Hecke grubunun sunuşu $H(p_1,\ldots,p_n)=\langle x_i\ ;\ x_i^{p_i}=1\rangle \cong C_{p_1} * \ldots * C_{p_n}$ biçimindedir. Ayrıca genişletilmiş ve genelleştirlmiş Hecke grubunun sunuşu ise

$$\overline{H}(p_1,\ldots,p_n)=\langle x_i,\ r\ ;\ x_i^{p_i}=r^2=1,\ rx_i=x_i^{-1}r\rangle$$

veya

$$\overline{H}(p_1,\ldots,p_n)=\langle x_i,\ r\ ;\ x_i^{p_i}=r^2=(x_ir)^2=1\rangle \cong D_{p_1} *_{\mathbb{Z}_2} \ldots *_{\mathbb{Z}_2} D_{p_n}$$

şeklindedir.

Bu alt bölümdeki açıklamalarla ilgili daha detaylı bilgiler (Çevik ve diğerleri, 2008; Hecke, 1936; Huang, 1999; Karpuz ve Çevik, 2012; Koruoğlu ve diğerleri, 2007; Şahin ve diğerleri, 2004; Newman, 1962) gibi kaynaklardan elde edilebilir.

2.4 Karar Verme Problemleri

Gruplar için tanımlanan grup sunuşları gruplara özel bazı problemlerin çözümünde kullanılmaktadır. Cevabı "evet" veya "hayır" olan problemlere karar verme problemleri denir. Verilen bir problemi çözmek için bir algoritma (veya metot) bulunabiliyorsa bu karar verme problemine *çözülebilirdir*, böyle bir algoritma bulunamıyor ise bu karar verme problemine *çözülemez* denir. Karar verme problemleri sadece matematikte değil, birçok mühendislik dalında da çalışılmaktadır.

Matematik alanındaki karar verme problemleri ilk olarak Max Dehn tarafından 1911 yılında literatüre kazandırılmıştır. Bu problemler; kelime problemi, eşlenik problemi ve izomorfizma problemi olmak üzere üç temel başlık altında toplanmıştır. Bu bölümde bu problemler hakkında kısa bir bilgi verilmiştir. Daha detaylı bilgiler (Çetinalp, 2016; Karpuz, 2006; Magnus ve diğerleri,1976) kaynaklarından elde edilebilir.

Kelime Problemi: Sonlu sunuşlu bir G grubu için, bir $w \in G$ nin bu grubun birimine eşit olup olmadığına karar veren bir algoritmanın varlığının araştırılması problemidir. Bu şekilde bir algoritma veya metod bulunabiliyorsa bu sonlu sunumlu G grubu için *kelime problemi çözülebilirdir* denir.

Aşağıda çözülebilir kelime problemine sahip bazı grup yapıları verilmiştir.

- Sonlu üreteçli abelyan gruplar,
- Sonlu üreteçli nilpotent gruplar,
- Sonlu sunumlu residual gruplar,
- Sonlu üreteçli metabelyan gruplar,
- Polycyclic gruplar,
- Sonlu sunumlu Hopfian gruplar.

Eşlenik Problemi: G sonlu sunuşa sahip bir grup olsun. G grubunun üreteçleri ile elde edilen herhangi bir u ve v kelimelerinin G nin eşlenik elemanları olup olmadığına karar veren bir algoritmanın varlığının araştırılması problemidir.

İzomorfizma Problemi: Sonlu sunuşa sahip herhangi iki grubun birbirine izomorf olup olmadığına karar veren bir algoritmanın var olup olmaması problemidir.

3. YENİDEN YAZMA SİSTEMİ

3.1 Giriş

Yeniden yazma; matematik, bilgisayar bilimi ve mantıkta bir formülün alt terimlerini diğer terimlerle değiştirmeye yönelik çok çeşitli yöntemleri kapsar. Bu tür yöntemler, yeniden yazma sistemleri (yeniden yazma motorları veya indirgeme sistemleri olarak da bilinir) ile elde edilebilir. Yeniden yazma sistemleri en temel biçimleriyle, bir dizi nesneden ve bu nesnelerin nasıl dönüştürüleceğine ilişkin ilişkilerden oluşur. Literatürde yeniden yazma sisteminin aşağıda verilen alt dallarıyla ilgili çeşitli çalışmalar bulunmaktadır. Bu bölümde bu alt dallardan pozitif kelime yeniden yazma sistemi ile ilgili kısa bilgi verilmiştir.

- Terim yeniden yazma sistemi
- Graf yeniden yazma sistemi
- Paralel yeniden yazma sistemi
- Sonsuz yeniden yazma sistemi
- Yüksek mertebe yeniden yazma sistemi
- Pozitif kelime yeniden yazma sistemi

3.2 Pozitif Kelime Yeniden Yazma Sistemi

Bu alt bölüm ile ilgili detaylı bilgilere ve bazı grup ve monoid yapıları üzerindeki çalışmalara (Book, 1987; Book ve Otto, 1993; Çetinalp, 2016; Çetinalp, 2020; Karpuz, 2006; Sims, 1994) kaynaklarından ulaşılabilir.

Sonlu bir X alfabesi için, X^* bu alfabedeki harflerden oluşan bütün kelimelerin kümesi ve λ boş kelime olsun. X kümesi üzerinde tanımlanan yeniden yazma kuralı $(l,r) \in X^* \times X^*$ şeklindeki sıralı çiftlerden oluşmaktadır. Bu kural $l \to r$ ifadesi ile gösterilir. Buradaki l kelimesi kuralın sol yanı, r kelimesi ise kuralın sağ yanı olarak adlandırılır. X kümesi üzerindeki yeniden yazma sistemi, X kümesi üzerindeki yeniden yazma kurallarının bir kümesidir. Bu sistem R ile gösterilir. X^*

kümesindeki kelimeler arasındaki bu bağıntı $(\to r)$ için aşağıdaki kural tanımlanmaktadır.

X üzerindeki u ve v pozitif kelimeler olmak üzere, $u \to_R v$ olması için gerek ve yeter koşul $x, y \in X^*$ ve $(l, r) \in R$ için, $u = xly$ ve $v = xry$ olmasıdır.

Aşağıda verilen tanımlara, önerme ve teoremlere (Book ve Otto, 1993) kaynağından ulaşılabilir.

Tanım 3.2.1: $u \to_R v$ olacak şekilde $u \in X^*$ kelimesi için bir $v \in X^*$ kelimesi varsa bu u kelimesine *indirgenir kelime* denir. Aksi durumda ise (yani $u \to_R v$ olacak biçimde bir v kelimesi yoksa) bu u kelimesine *indirgenemez kelime* denir.

Önerme 3.2.2: Tanımlı $\to_R$ bağıntısının yansımalı, geçişmeli kapanışı olan $\to^*_R$ kuralı, aslında R tarafından üretilen bir indirgeme bağıntısıdır.

Tanım 3.2.3: $u, v \in X^*$ için eğer $u \to^*_R v$ bağıntısı varsa ve v kelimesi indirgenemez ise, bu v kelimesine u kelimesinin *normal formu* denir.

Tanım 3.2.4 : Bu $\to_R$ bağıntısının yansımalı, simetrik ve geçişmeli kapanışı $\leftrightarrow^*_R$ ile gösterilirse, bu kural R tarafından üretilen bir *Thue kongrüansı*dır. Bir $w \in X^*$ kelimesinin kongrüans sınıfı $\{u \in X^* \mid u \leftrightarrow^*_R w\}$ ile ifade edilir. Bu kongruans sınıfı $[w]_R$ ile gösterilir. Buna ek olarak $X^* / \leftrightarrow^*_R$ ifadesi bütün kongrüans sınıflarının kümesini göstermektedir. Alınan u ve v kelimelerinin kongrüans sınıflarının çarpımı $[u]_R [v]_R = [uv]_R$ ile ifade edilir. Bu işlemin birleşme özelliği vardır ve $[\lambda]_R$ da birim elemandır. Böylece $X^* / \leftrightarrow^*_R$ yapısı bir monoid oluşturur ve $[X; R]$ çifti de bir monoid sunuşudur. Bu sunuştaki üreteç kümesi sonlu ise bu monoide sonlu üreteçli monoid, hem üreteç hem de bağıntı kümelerinin sonlu olması durumunda ise bu monoide sonlu sunumlu monoid denir.

Bir R yeniden yazma sistemi için kelime problemi, verilen u ve v kelimeleri için $u \leftrightarrow^*_R v$ nin sağlanıp-sağlanmamasıdır. Bunun için öncelikle bazı tanımların verilmesi gerekmektedir.

R yeniden yazma sistemi olmak üzere,

Tanım 3.2.5:

i) R sistemi için kelimeler arasında $w_1 \rightarrow_R w_2 \rightarrow_R w_3 \rightarrow_R \ldots$ biçiminde sonsuz bir zincir oluşmuyorsa, bu yeniden yazma sistemine *Noetherian* ya da *sona ermiş* denir.

ii) R sisteminden alınacak bütün $u, v, w \in X^*$ kelimeleri için,

$$u \rightarrow^*_R v \quad \text{ve} \quad u \rightarrow^*_R w \text{ iken} \quad v \rightarrow^*_R z \quad \text{ve} \quad w \rightarrow^*_R z$$

olacak biçimde bir $z \in X^*$ kelimesi varsa, bu yeniden yazma sistemine *elmas kuralı* denir.

Tanım 3.2.6: Hem Noetherian hem de elmas kuralı özelliklerini sağlayan yeniden yazma sistemine *tam yeniden yazma sistemi* denir.

Bir tam yeniden yazma sisteminin en önemli sonucu aşağıda verilmiştir.

Teorem 3.2.7: R yeniden yazma sistemi tam olmak üzere, bu sistem içindeki her bir kelime tek bir normal forma sahip olacağından, bu R yeniden yazma sistemi için kelime problemi çözülebilirdir.

Bir $\langle X ; R \rangle$ sunuşunun yeniden yazma sisteminin tam olduğunu göstermek için aşağıda verilen aşamalar uygulanmaktadır.

I. Aşama: İlk olarak verilen yeniden yazma sisteminin sona ermiş olduğu gösterilmelidir. Bunun için, X^* kümesindeki kelimeler arasında bir indirgeme sıralamasının seçilmesi gerekmektedir. Bu sıralamalar; uzunluk sıralaması, ağırlık sıralaması, soldan sözlük sıralaması, uzunluk ve soldan sözlük sıralaması, uzunluk ve sağdan sözlük sıralamasıdır. Bunlardan en elverişli ve en sık kullanılanı uzunluk ve soldan sözlük sıralamasıdır.

II. Aşama: Yeniden yazma sisteminin sona erdiğinin gösterilmesinden sonraki adım elmas kuralının sağlandığını göstermektir. Bunun için ise; yeniden yazma sistemindeki kuralların sol yanlarının uygun çakışmaları kontrol edilir. Bu çakışmalardan elde

edilen kritik çiftlerin çözülebilir olması gerekmektedir. Daha açık bir şekilde kritik çift ile anlatılmak istenen; $u, v, w, p, q \in X^*$ kelimeleri için, $uv = p$ ve $vw = q$ (v boş kelimeden farklı) bağıntılarından elde edilen $\{pw, uq\}$ biçimindeki kelime çiftleridir. Bu $\{pw, uq\}$ kritik çiftin çözülebilir olması demek ise, $pw \to^*_R z$ ve $uq \to^*_R z$ olacak şekilde bir $z \in X^*$ kelimesinin elde edilebilir olması anlamına gelmektedir.

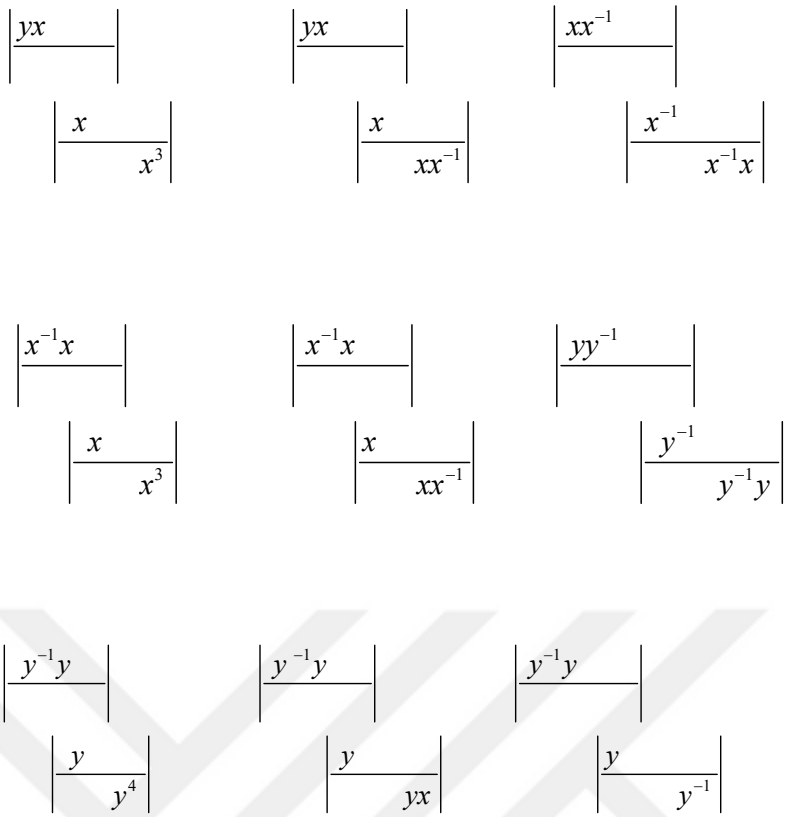
Bir kritik çiftin çözülebilir olmaması durumunda bu kritik çifteki kelimelere uygulanan indirgeme adımlarından sonra elde edilen en son kelimelerin yeniden yazma sistemine yeni bir kural olarak eklenmesi gerekmektedir. Bu durum eklenen yeni kurallarla birlikte elde edilen tüm kuralların çakışmalarının kontrol edilmesi işlemi ile devam eder. Bu süreç *Knuth-Bendix Algoritması* olarak adlandırılır. Eğer bu algoritma başarı ile sonuçlanırsa (yani yeni elde edilen yeniden yazma sistemindeki kuralların elmas kuralını sağlaması), verilen sunuşun temsil ettiği grup için kelime problemi çözülebilirdir. Sonuç olarak, Knuth-Bendix Algoritması verilen bir sunuşun bağıntı kümesindeki kuralları elmas kuralının sağlandığı yeniden yazma sistemine dönüştüren bir algoritmadır.
Bu konu hakkındaki detaylı bilgilere (Book, 1987; Book ve Otto, 1993; Sims, 1994) kaynaklarından ulaşılabilir.

Örnek 3.2.8: $y^{-1} \rangle x^{-1} \rangle y \rangle x$ harfsel sıralamayı dikkate alarak

$xx^{-1} \to 1, \quad x^{-1}x \to 1, \quad yy^{-1} \to 1, \quad y^{-1}y \to 1,$
$x^3 \to 1, \quad y^4 \to 1, \quad yx \to xy$

biçiminde yedi kuraldan oluşan yeniden yazma sistemini ele alalım ve bu yeniden yazma sisteminde elmas kuralının sağlandığını gösterelim. Bu yedi yeniden yazma kuralının sol yanlarına bakarak çakışan bütün kelimeleri inceleyelim. Bu çakışan kelimeler şematik olarak aşağıda verilmiştir.

$$\left|\frac{x^3}{\left|\frac{x}{xx^{-1}}\right|}\right| \qquad \left|\frac{y^4}{\left|\frac{y}{yx}\right|}\right| \qquad \left|\frac{y^4}{\left|\frac{y}{yy^{-1}}\right|}\right|$$



Bu çakışan kelimelerden elde edilen kritik çiftler

$$\{x^{-1},x^2\},\ \{x,y^3xy\},\ \{y^{-1},y^3\},\ \{xyx^2,y\},\ \{xyx^{-1},y\},\ \{x,x\},$$
$$\{x^2,x^{-1}\},\ \{x^{-1},x^{-1}\},\ \{y,y\},\ \{y^3,y^{-1}\},\ \{x,y^{-1}xy\},\ \{y^{-1},y^{-1}\}$$

biçimindedir. Elde edilen bu kritik çiftlerin çözülebilir olduğunu yani her birinin tek bir kelimeye indirgenip indirgenmediğini inceleyelim.

(1) $x^3x^{-1}\to\begin{Bmatrix}x^2\\x^{-1}\end{Bmatrix}$ (2) $y^4x\to\begin{Bmatrix}y^3xy\to y^2xy^2\to yxy^3\to xy^4\to x\\x\end{Bmatrix}$

(3) $y^4y^{-1}\to\begin{Bmatrix}y^3\\y^{-1}\end{Bmatrix}$ (4) $yx^3\to\begin{Bmatrix}y\\xyx^2\to x^2yx\to x^3y\to y\end{Bmatrix}$

(5) $yxx^{-1}\to\begin{Bmatrix}y\\xyx^{-1}\end{Bmatrix}$ (6) $xx^{-1}x\to\begin{Bmatrix}x\\x\end{Bmatrix}$

(7) $x^{-1}x^3\to\begin{Bmatrix}x^{-1}\\x^2\end{Bmatrix}$ (8) $x^{-1}xx^{-1}\to\begin{Bmatrix}x^{-1}\\x^{-1}\end{Bmatrix}$

(9) $yy^{-1}y\to\begin{Bmatrix}y\\y\end{Bmatrix}$ (10) $y^{-1}y^4\to\begin{Bmatrix}y^{-1}\\y^3\end{Bmatrix}$

(11) $y^{-1}yx\to\begin{Bmatrix}y^{-1}xy\\x\end{Bmatrix}$ (12) $y^{-1}yy^{-1}\to\begin{Bmatrix}y^{-1}\\y^{-1}\end{Bmatrix}$

$(2), (4), (6), (8), (9)$ ve (12) ile gösterilen kritik çiftler aynı kelimeye indirgenirken, $(1), (3), (5), (7), (10)$ ve (11) ile gösterilen kritik çiftler aynı kelimeye indirgenememiştir. Bu durum yeniden yazma sisteminin elmas kuralının sağlanmadığı anlamına gelmez. Bu durumda yukarıda bahsedilen Knuth-Bendix algortiması uygulanır. Bu algoritmanın bir uygulaması 5.3 alt bölümde verilmiştir.

4. GRUP, ALGORİTMA VE PROGRAMLAMA (GAP)

4.1 Giriş

Grup, Algoritma ve Programlama (GAP); özellikle Hesaplamalı Grup Teorisi üzerine çalışmalar yapan, hesaplamalı ayrık cebirde kullanılan bir sistemdir. GAP, bir programlama dilidir. GAP, GAP dilinde yazılmış cebirsel algoritmaları uygulayan binlerce fonksiyondan oluşan bir kitaplığın yanı sıra cebirsel nesnelerin büyük veri kitaplıklarını da sağlar. GAP, grupları ve temsillerini, halkaları, vektör uzaylarını, cebirleri, kombinatoryal yapıları ve daha fazlasını incelemek için araştırmada ve öğretimde kullanılır. Özel kullanımınız için çalışabilir, kolayca değiştirebilir veya genişletebilirsiniz. GAP, C ile yazılmış bir çekirdeğe sahiptir.

Bu program ilk olarak, 1986-1997 yılları arasında Almanya'nın Aachen şehrindeki RWTH Lehrstuhl D für Mathematik (LDFM) bölümünde geliştirilmiştir. 1997'den sonra GAP'ın gelişimi İngiltere'deki St Andrews Üniversitesi tarafından koordine edilmiştir. Şu anda; Aachen, Braunschweig, Fort Collins, Kaiserslautern ve St Andrews'daki GAP Merkezleri, GAP' ın daha da geliştirilmesi ve sürdürülmesini ortaklaşa koordine etmektedir.

GAP açık kaynak kodlu bir programdır, kolay ve anlaşılır bir kullanıma sahiptir. Birçok işletim sisteminde (Unix, Windows ve Macintosh) çalışmaktadır. GAP programı çekirdeğinde var olan programlarla birlikte kütüphanesinde bulunan ortak paket programlarla da çalışabilmektedir.

Bu bölümde; GAP programlama dili hakkında kısa bilgi verilmiştir. Bu konu ile ilgili ayrıntılı bilgilere (*https://www.gap-system.org/index.html*; Aslan, 2010; Odabaş, 2004; Şimşek, 2018) çalışmalarından ulaşılabilir.

GAP programı kapsamında yazılmış birçok paket program vardır. Bu paket programlara ayrıntılı bir şekilde

https://www.gap-system.org/Packages/packages.html

kaynağından ulaşılabilmektedir. Bu paket programların çalışıldığı konular özet olarak aşağıdaki gibi sınıflandırılabilir.

• Temel Yetenekler

• Grup ve Grup Elemanları

• Permütasyon ve Matris Grupları

• Sonlu Sunumlu Gruplar

• Polycyclic Gruplar

•Vektör Uzayları, Modüller ve Cebirler

•Graflar

•Yarıgruplar, Monoidler ve Grupların Diğer Türevleri

• Kelimeler, Yeniden Yazma Sistemi ve Otomata Teorisi

4.2 Bilgisayar Donanımı

Son versiyonu Mart 2021 de çıkan GAP (4.11.1 versiyonu) programı, internet üzerinden ücretsiz olarak kullanılabilmektedir. GAP programının çalışabilmesi için, bilgisayarın en az 20 MB RAM e ve Pentium 133 MHz veya daha hızlı işlemciye sahip olması gerekmektedir. Programın tamamının bilgisayara yüklenebilmesi için yaklaşık olarak 320 MB yere ihtiyaç duyulmaktadır. GAP programı tüm paketleri ile birlikte kısa süre içerisinde yüklenip kullanıma hazır hale gelmektedir.

4.3 GAP Dili

GAP programında karşılaşılan bazı durumlar ve sık kullanılan komutların bazıları hakkında aşağıda kısaca bilgi verilmiştir.

Programda yazılan komutların sonuna ";" konulması durumunda program komutun bittiğini anlar ve sonuç verir. Aksi durumda sonuç vermemektedir. GAP programından çıkmak istenildiğinde *quit;* komutu verilir.

LogTo("DosyaAdı); komutu yapılan uygulamanın kaydedilmesi için kullanılır. Bu komut ile yapılan uygulamayı veya çalışmayı *.log* uzantılı metin dosyasına kaydeder.

Programda yapılan bir uygulama bir hata döngüsü içerisine girerse bu döngüden çıkması yani normal *gap>* halini alması için *quit;* komutunun verilmesi veya *Ctrl+D* tuşlarına basılması gerekmektedir.

GAP programı kullanıcılarının bilmesi gereken hususlardan biri de büyük küçük harf kullanımı arasındaki farktır. Örneğin; *LoadPackage* ve *Loadpackage* aynı işlemi gerçekleştirmez.

Programda uygulama yapılırken her komut sonunda bir noktalı virgül konularak iki farklı komutun aynı satırda bulunması sağlanır. Bir komutun sonuna iki noktalı virgül konulması durumunda komut ekrana yazılmamaktadır.

Anahtar Kelimeler

GAP programlama dilindeki;

and, do, elif, else, end, fi, for, function, if, in, local, mod, not, od, or, repeat, return, then, until, while, quit, QUIT, break, rec, continue

biçimindeki anahtar kelimeler değişken ismi olarak yazılamazlar.

GAP programı büyük-küçük harf duyarlılığına sahip olduğu için hemen hemen tüm özel kelimelerin küçük harfle yazılmış olmasına dikkat edilmelidir. Örneğin; return özel kelimesindeki harflerden herhangi birisi büyük harfle yazılırsa bu kelimeler özel kelime olma özelliğini kaybeder. Return, rEturn, reTurn, retUrn, retuRn, returN kelimeleri özel kelimeler olmayıp değişken ismi olarak kullanılabilirler. Özel kelimeler arasında boşluk kullanılmamalıdır. el if şeklindeki bir yazılım elif özel kelimesinin yaptığı işlevi yerine getiremeyecektir.

Karşılaştırma Operatörleri

GAP programlama dilinde aşağıdaki temel karşılaştırma operatörleri kullanılmaktadır.

GAP

```
< , > , <=, >=, <>, =
  =  operatörü bir aileden iki değerin eşitliğini gerçekleştirir.
  <  operatörü bir aileden iki değerin küçüklük karşılaştırmasını yapar.
  >  operatörü bir aileden iki değerin büyüklük karşılaştırmasını yapar.
  <> operatörü bir aileden iki değerin eşit olup olmadığı karşılaştırır.
  <= operatörü bir aileden iki değerin küçük eşitlik karşılaştırmasını
yapar.
  >= operatörü bir aileden iki değerin büyük eşitlik karşılaştırmasını
yapar.
```

Yukarıda verilen karşılaştırma operatörlerinden bazılarının uygulaması aşağıda verilmiştir.

```
GAP
gap> x:=18;
18
gap> y:=12;
12
gap> x<y;
false
gap> x>y;
true
gap> x<>y;
true
```

Aritmetik Operatörler

Temel aritmetik operatörler aşağıda verilmiştir.

```
GAP
        /, +, - , *, ^, mod
x/y  komutu; birinci sayısının ikinci sayıya bölünmesidir.
x+y  komutu; birinci sayı ile ikinci sayının toplamını verir.
x-y  komutu; birinci sayı ile ikinci sayının farkını verir.
x*y  komutu; birinci sayı ile ikinci sayının çarpımını verir.
x^y  komutu; x sayısının y. dereceden kuvvetinin alınmasıdır.
x mod y komutu; x sayısının mod y cinsinden değerinin   bulunmasıdır.
-  operatörü; bir sayının işaretini de değiştirebilir.
```

Matematikte olduğu gibi GAP dilinde de işlem önceliği vardır. Parantez içinde yazılan işlemler öncelikli olarak yapıldıktan sonra üst alma işlemi, daha sonra ise çarpma veya bölme ve toplama veya çıkarma işlemleri yapılır.

Aşağıdaki örnekte aritmetik operatörlerin bir uygulaması verilmiştir.

```
GAP
gap> x:=2*(15^13);
3892390136718750
gap> y:=(2*15)^13;
159432300000000000000
gap> x=y;
false
gap> 2*3^4;
162
```

Mantıksal Operatörler

Mantıksal operatörler GAP programında kıyaslama yapılacağı bir durumda kullanılır. Bu operatörler *and* (ve), *not* (değil) ve *or* (veya) olmak üzere üç tanedir. *and* (ve) operatörü, karşılaştırma yapılacak olan iki değerden herhangi biri doğru ise doğru

sonucunu verir. *or* (veya) operatörü ise, karşılaştırma yapılacak olan iki değer de doğru ise doğru sonucunu verir.

Program Denetim Deyimleri

if Deyimi

Bir şarta bağlı işlemleri gerçekleştirmek için kullanılan bir deyimdir. Komutun bir uygulaması aşağıda verilmiştir.

```
GAP
gap> q:=-35;
-35
gap> if 0<q then
> sgnk:=1;
> elif q<0 then
> sgnk:=-2;
> else
> sgnk:=0;
> fi;
gap> sgnk;
-2
```

while Deyimi

Bu deyim belli bir şart ile birlikte döngü oluşturmak için kullanılır. Bu deyimin bir uygulaması aşağıda verilmiştir.

```
GAP
gap> para:=20;
20
gap> yil:=0;
0
gap> while para<150 do
> para:=para+(para*(16/100));
> yil:=yil+1;
> od;
gap> yil;
14
```

for Deyimi

for deyimi de belli bir koşulla birlikte döngü oluşturmak için kullanılan bir deyimdir. *for* deyiminin en önemli özelliği bir döngü sayacına sahip olmasıdır. Bu denetim deyiminin bir uygulaması aşağıda verilmiştir.

```
                GAP
gap> faktoriyel:=1;
1
gap> for i in [1..6] do
> faktoriyel:=faktoriyel*i;
> od;
gap> faktoriyel;
720
```

print Deyimi

Bir veya daha fazla tanımlayıcı isminin tuttuğu değeri ekrana yazdırmak için *print* komutu kullanılır. *print* komutu genel olarak döngüler içerisinde kullanılır. Birden fazla tanımlayıcı ismi tek bir komut içerisinde virgülle birbirinden ayırarak kullanılabiliriz.

5. *idrel* PAKETİ VE UYGULAMALARI

5.1 Giriş

idrel paketi ile ilgili ilk çalışmalar, A. Heyworth tarafından Bangor Üniversitesindeki doktora tez çalışması ile başlamıştır. A. Heyworth ve C. D. Wensley tarafından yazılan bu paket, sunuş kavramı ve yeniden yazma sistemleri ile ilgili uygulamalar yapılmasına olanak sağlamaktadır. *idrel* paketi 2015 yılında GAP kütüphanesine eklenmiştir. Bu paket ile ilgili bazı bilgiler bu bölümde verilmiştir. Ancak detaylı bilgilere *www.gap-system.org/Packages/idrel* adresinden ulaşılabilir. Bu paket; bir grup (monoid) sunuşunun bağıntıları arasındaki ilişkileri kullanarak,

- Yeniden yazma, kayıtlı yeniden yazma,
- Monoid polinomları,
- Modül polinomları,
- Y-dizileri

ile ilgili uygulamalar yapmaktadır.

Bu bölümde, *idrel* paketindeki bazı fonksiyonlar kısaca verildikten sonra, 2. Bölümde verilen genişletilmiş Hecke gruplarının monoid sunuşları üzerine bazı uygulamalar yapılmıştır. Bu bölümde yapılan çalışmalara benzer uygulamalara (Şimşek, 2018) kaynağından ulaşılabilmektedir.

GAP programı açıldıktan sonra "*LoadPackag*e" komutu verilerek *idrel* paketi yüklenir. *idrel* paketi GAP programına çağırıldığında "true" yazması paketimizin problemsiz çalıştığını ifade eder.

```
                                        GAP
gap> LoadPackage("idrel");
──────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────
Loading  AutoDoc 2018.02.14 (Generate documentation from GAP source
code)
by Sebastian Gutsche (http://wwwb.math.rwth-aachen.de/~gutsche/) and
   Max Horn (http://www.quendi.de/math).
Homepage: https://gap-packages.github.io/AutoDoc
──────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────
Loading IdRel 2.41 (Identities among Relations)
by Anne Heyworth and Chris Wensley (http://pages.bangor.ac.uk/~mas023/)
------------------------------------------------------------------------------
true
```

idrel paketi için tipik bir girdi, bir sonlu sunumlu grup sunuşudur. Bu, üreteçler kümesi üzerindeki bir F serbest grubunu ve bir R bağıntı kümesini gerektirir (serbest

gruptaki kelimeler). Bu pakette, bağıntılar arasındaki birimler $\left[\left[r_1\ ,u_1\right],\left[r_2\ ,u_2\right],...,\left[r_k,u_k\right]\right]$ listeler halindeki Y-dizileri ile temsil edilirler. Burada $r_1,r_2,...,r_k$ grup bağıntıları ve onların tersleri, $u_1,u_2,...,u_k$ lar ise F serbest grubundaki kelimelerdir. F serbest grubundaki bir Y-dizisi; çarpımı $(u_1^{-1}r_1u_1)(u_2^{-1}r_2u_2)...(u_k^{-1}r_ku_k)$ olarak değerlendirilir. Eğer bu Y-dizisi serbest F grubunda birimi belirtmiş ise o zaman birim Y-dizisi olarak değerlendirilir.
Bir birim Y-dizisi, grup sunuşunun bağıntıları arasında birimi temsil eder. Paketin ana işlevi bağıntılar arasındaki birimin modülünü üreten bir küme üretmektir.

5.2 *idrel* Paketinin Fonksiyonları

idrel paketinin temel fonksiyonları ve bu fonksiyonlarla ilgili örnekler aşağıda verilmiştir. Bu fonksiyonlarla ilgili detaylı bilgilere (Şimşek, 2018) tezinden bakılabilir.
$\mathbb{Z}_3=\left\langle x\ ;\ x^3=1\right\rangle$ ve $\mathbb{Z}_4=\left\langle y\ ;\ y^4=1\right\rangle$ gruplarının $\mathbb{Z}_3\times\mathbb{Z}_4$ direkt çarpım grubunun sunuşu $\left\langle x,\ y\ ;\ x^3=1, y^4=1, xy=yx\right\rangle$ biçimindedir. Bu sunuşun monoid sunuşu ise,
$\mathbb{Z}_3\times\mathbb{Z}_4=\left\langle x,\ x^{-1},\ y,\ y^{-1}\ ;\ x^3=y^4=1,\ xy=yx,\ xx^{-1}=1,\ x^{-1}x=1,\ yy^{-1}=1,\ y^{-1}y=1\right\rangle$
şeklindedir. Bu alt bölümde $\mathbb{Z}_3\times\mathbb{Z}_4$ direkt çarpım grubunun sunuşu kullanılarak ilgili fonksiyonların uygulamaları yapılmıştır. Uygulamalarda bu grup "$q6$" ile gösterilmiştir.

- ***MonoidPresentationFpGroup*** **Fonksiyonu**

```
                                        GAP
gap> fgmon:=FreeGroupOfPresentation(mon);
<free group on the generators [ q6_M1, q6_M2, q6_M3, q6_M4 ]>
```

- ***FreeGroupOfPresentation*** **Fonksiyonu**

```
                                        GAP
gap> mon:=MonoidPresentationFpGroup(q6);
monoid presentation with group relators [ q6_M1^3,
q6_M2^4,q6_M1*q6_M2*q6_M3*q6_M4 ]
```

- ***GroupRelatorsOfPresentation*** **Fonksiyonu**

```
                                                    GAP
gap> genfgmon:=GeneratorsOfGroup(fgmon);
[ q6_M1, q6_M2, q6_M3, q6_M4 ]
```

- ***InverseRelatorsOfPresentation*** **Fonksiyonu**

```
                                                    GAP
gap> invrels:=InverseRelatorsOfPresentation(mon);
[ q6_M1*q6_M3, q6_M2*q6_M4, q6_M3*q6_M1, q6_M4*q6_M2 ]
```

- ***HomomorphismOfPresentation*** **Fonksiyonu**

Aşağıdaki uygulamada da görüldüğü gibi x^+, y^+, x^- ve y^- monoid üreteçleri sırasıyla $q6\_M1$, $q6\_M2$, $q6\_M3$ ve $q6\_M4$ ile tanımlanmıştır.

```
                                                    GAP
gap> hompres:=HomomorphismOfPresentation(mon);
[ q6_M1, q6_M2, q6_M3, q6_M4 ] -> [ f1, f2, f1^-1, f2^-1 ]
```

- ***RewritingSystemFpGroup*** **Fonksiyonu**

$\mathbb{Z}_3 \times \mathbb{Z}_4 = \langle x,\ x^{-1},\ y,\ y^{-1}\ ;\ x^3 = y^4 = 1,\ xy = yx,\ xx^{-1} = 1,\ x^{-1}x = 1,\ yy^{-1} = 1,\ y^{-1}y = 1 \rangle$

sunuşunun on beş tane yeniden yazma kuralı vardır. Bu kurallar,

$(1)\ xx^{-1} \to id$ $(2)\ yy^{-1} \to id$ $(3)\ x^{-1}x \to id$ $(4)\ y^{-1}y \to id$ $(5)\ xy^2x^{-1} \to y^2$

$(6)\ yx^{-1}y^{-1} \to y$ $(7)\ y^3 \to y^{-1}$ $(8)\ xyx^{-1} \to y$ $(9)\ x^{-2} \to x$ $(10)\ x^2 \to x^{-1}$

$(11)\ y^{-2} \to y^2$ $(12)\ y^{-1}x^{-1} \to x^{-1}y^{-1}$ $(13)\ y^{-1}x \to xy^{-1}$ $(14)\ x^{-1}y \to yx^{-1}$ $(15)\ yx \to xy$

şeklindedir. Bu kurallar *RewritingSystemFpGroup* komutu ile aşağıda elde edilmiştir.

```
                                                    GAP
gap> rws:=RewritingSystemFpGroup(q6);
 [ [ q6_M1*q6_M3, <identity ...> ], [ q6_M2*q6_M4, <identity ...> ],
 [ q6_M3*q6_M1, <identity ...> ],[ q6_M4*q6_M2, <identity ...> ],
 [ q6_M1*q6_M2^2*q6_M3, q6_M2^2 ], [ q6_M2*q6_M3*q6_M4, q6_M3 ],
 [ q6_M2^3, q6_M4 ],[ q6_M1*q6_M2*q6_M3, q6_M2 ], [ q6_M3^2, q6_M1 ],
 [ q6_M1^2, q6_M3 ], [ q6_M4^2, q6_M2^2 ],[ q6_M4*q6_M3, q6_M3*q6_M4 ],
 [ q6_M4*q6_M1, q6_M1*q6_M4 ], [ q6_M3*q6_M2, q6_M2*q6_M3 ],
 [ q6_M2*q6_M1, q6_M1*q6_M2 ] ]
```

- ***OnePassKB*** **Fonksiyonu**

$\mathbb{Z}_3 \times \mathbb{Z}_4 = \langle x,\ x^{-1},\ y,\ y^{-1}\ ;\ x^3 = y^4 = 1,\ xy = yx,\ xx^{-1} = 1,\ x^{-1}x = 1,\ yy^{-1} = 1,\ y^{-1}y = 1 \rangle$

monoid sunuşunun yedi tane bağıntıdan oluştuğu görülmektedir. Bu bağıntıların sol yanlarının çakışan kelimeleri incelendiğinde yedi tane daha yeni bağıntı elde edildiği görülmüştür. Bu yeni bağıntılar,

$(1)\ yx^{-1}y^{-1} \to x^2 \quad (2)\ x^2 \to x^{-1} \quad (3)\ y^3 \to y^{-1} \quad (4)\ xyx^{-1} \to y$
$(5)\ yx^{-1}y^{-1} \to x^{-1} \quad (6)\ y^3 \to y^{-1} \quad (7)\ x^2 \to x^{-1}$

biçimindedir. Yukarıda elde edilen bağıntıların *OnePassKB* fonksiyonu ile çıktısı aşağıda verilmiştir.

GAP

```
gap> r1:=OnePassKB(r0);
 [ [ q6_M1^3, <identity ...> ], [ q6_M2^4, <identity ...> ],
 [ q6_M1*q6_M2*q6_M3*q6_M4, <identity ...> ],
 [ q6_M1*q6_M3, <identity ...> ], [ q6_M2*q6_M4, <identity ...> ],
 [ q6_M3*q6_M1, <identity ...> ],[ q6_M4*q6_M2, <identity ...> ],
 [ q6_M2*q6_M3*q6_M4, q6_M1^2 ], [ q6_M1^2, q6_M3 ],[ q6_M2^3, q6_M4],
 [ q6_M1*q6_M2*q6_M3, q6_M2 ],[ q6_M1^2, q6_M3 ],
 [ q6_M2*q6_M3*q6_M4, q6_M3 ], [ q6_M2^3, q6_M4 ]]
gap> Length(r1);
14
```

- ***RewriteReduce*** **Fonksiyonu**

OnePassKB fonksiyonu ile elde edilen bağıntılardan olan $x^3 \to 1$ bağıntısı $x^2 \to x^{-1}$ bağıntısından elde edileceği için silinmektedir. Buna benzer

$(1)\ y^4 \to 1 \quad (2)\ y^{-1}xy \to x \quad (3)\ x^{-1}yx^{-1} \to xy \quad (4)\ y^{-1}x^2y \to x^{-1} \quad (5)\ y^{-1}xy^{-1} \to xy^2$

bağıntıları da silindiği için aşağıdaki sekiz bağıntı elde edilmiştir.

GAP

```
gap> r1:=RewriteReduce(r1);
  [ [ q6_M1^2, q6_M3 ], [ q6_M1*q6_M3, <identity ...> ],
  [ q6_M2*q6_M4, <identity ...> ], [ q6_M3*q6_M1, <identity ...> ],
  [ q6_M4*q6_M2, <identity ...> ], [ q6_M1*q6_M2*q6_M3, q6_M2 ],
  [ q6_M2^3, q6_M4 ], [ q6_M2*q6_M3*q6_M4, q6_M3 ] ]
```

- ***Knuth-Bendix*** **Fonksiyonu**

Knuth-Bendix fonksiyonu, çakışan kelimelerin en son indirgenmesini gerçekleştirerek artık indirgeme yapılamayacak kelimelerden oluşan yeni bağıntılar elde eder. Örneğin; $xy^{-1}x^{-1} \to y^{-1}$ bağıntısının indirgeme işlemleri yapıldığında en son indirgemesi olan $y^{-1}x^{-1} \to x^{-1}y^{-1}$ bağıntısı bulunur.

```
GAP
gap> r2:=KnuthBendix(r1);
  [ [ q6_M1^2, q6_M3 ], [ q6_M1*q6_M3, <identity ...> ],
  [ q6_M2*q6_M1, q6_M1*q6_M2 ], [ q6_M2*q6_M4, <identity ...> ],
  [ q6_M3*q6_M1, <identity ...> ], [ q6_M3*q6_M2, q6_M2*q6_M3 ],
  [ q6_M3^2, q6_M1 ], [ q6_M4*q6_M1, q6_M1*q6_M4 ],
  [ q6_M4*q6_M2, <identity ...> ], [ q6_M4*q6_M3, q6_M3*q6_M4 ],
  [ q6_M4^2, q6_M2^2 ], [ q6_M1*q6_M2*q6_M3, q6_M2 ],
  [ q6_M2^3, q6_M4 ], [ q6_M2*q6_M3*q6_M4, q6_M3 ],
  [ q6_M1*q6_M2^2*q6_M3, q6_M2^2 ] ]
gap> Length(r2);
15
```

- ***ElementsOfMonoidPresentation*** **Fonksiyonu**

Bir grubun elemanlarının normal formunu, tam yeniden yazma sistemindeki kurallar kullanılarak elde edilen indirgenmiş kelimeler oluşturmaktadır. $\mathbb{Z}_3 \times \mathbb{Z}_4 = \langle x,\ x^{-1},\ y,\ y^{-1}\ ;\ x^3 = y^4 = 1,\ xy = yx,\ xx^{-1} = 1,\ x^{-1}x = 1,\ yy^{-1} = 1,\ y^{-1}y = 1 \rangle$ sunuşu için bu liste aşağıda verilmiştir.

$$\{1,\ x,\ y,\ x^{-1},\ y^{-1},\ xy,\ xy^{-1},\ y^2,\ yx^{-1},\ x^{-1}y^{-1},\ xy^2,\ y^2x^{-1}\}$$

Yukarıda verilen normal form kümesi, *ElementsOfMonoidPresentation(q6)*; fonsiyonu ile aşağıda verilmiştir.

```
GAP
gap> elq6:=Elements(q6);
[ <identity ...>, f1, f1^2, f2, f2^3, f1*f2, f1*f2^3, f1^2*f2,
f1^2*f2^3, f2^2, f1*f2^2, f1^2*f2^2 ]
gap> elmonq6:=ElementsOfMonoidPresentation(q6);
[ <identity ...>, q6_M1, q6_M2, q6_M3, q6_M4, q6_M1*q6_M2, q6_M1*q6_M4,
q6_M2^2, q6_M2*q6_M3, q6_M3*q6_M4, q6_M1*q6_M2^2, q6_M2^2*q6_M3 ]
```

5.3 *idrel* Paketinde Knuth-Bendix Uygulamaları

Genişletilmiş Hecke Grubunun Uygulaması-1

Bu alt bölümde, 2.3 alt bölümde verilen

$\overline{H}(\lambda_q) = \langle\ x,\ y,\ z,\ r\ ;\ x^2,\ y^q,\ z^2,\ r^2,\ (xr)^2 = 1, (yr)^2 = 1, (zr)^2 = 1 \rangle$

genişletilmiş Hecke grubunda $q = 2$ alınarak bulunan $\overline{H}(\lambda_4) = \langle\ x,\ y,\ z,\ r\ ;\ x^2, y^4, z^2, r^2, (xr)^2 = 1, (yr)^2 = 1, (zr)^2 = 1 \rangle$ grubu için, *idrel* paketinin bir uygulaması verilmiştir. $\overline{H}(\lambda_2)$ genişletilmiş Hecke grubunun monoid sunuşu

$$\overline{H}(\lambda_2)=\left\langle \begin{array}{c} x,\ y,\ z,\ r,\ x^{-1},\ y^{-1},\ z^{-1},\ r^{-1}\ ;\ x^2=y^2=z^2=r^2=1, xx^{-1}=1, x^{-1}x=1, yy^{-1}=1, y^{-1}y=1, \\ zz^{-1}=1, z^{-1}z=1, rr^{-1}=1, r^{-1}r=1 \end{array} \right\rangle$$

biçimindedir. Bu sunuşun tam yeniden yazma sistemini hesaplamak için ilk olarak kelimeler arasında uzunluk sıralaması ve uzunlukları eşit olan kelimeler için üreteçler arasında $r^{-1}>r>z^{-1}>z>y^{-1}>y>x^{-1}>x$ biçiminde harfsel sıralama seçilir. Bu sıralama kullanılarak $\overline{H}(\lambda_2)$ genişletilmiş Hecke grubunun yeniden yazma sistemi aşağıdaki kurallardan oluşmaktadır.

(1) $x^2=1$ (2) $y^2=1$ (3) $z^2=1$ (4) $r^2=1$ (5) $(xr)^2=1$
(6) $(yr)^2=1$ (7) $(zr)^2=1$ (8) $xx^{-1}=1$ (9) $x^{-1}x=1$ (10) $yy^{-1}=1$
(11) $y^{-1}y=1$ (12) $zz^{-1}=1$ (13) $z^{-1}z=1$ (14) $rr^{-1}=1$ (15) $r^{-1}r=1$

Yukarıdaki $(1)-(15)$ arasındaki bağıntılardan elde edilen çakışan kelimeler ve ilgili kritik çiftler aşağıda verilmiştir.

$(1)\cap(5): x^2rxr,\ (rxr,x)$
$(2)\cap(6): y^2ryr,\ (ryr,y)$
$(3)\cap(7): z^2rzr,\ (rzr,z)$
$(4)\cap(14): r^2r^{-1},\ (r^{-1},r)$
$(5)\cap(14): xrxrr^{-1},\ (r^{-1},xrx)$
$(6)\cap(14): yryrr^{-1},\ (r^{-1},yry)$

$(1)\cap(8): x^2x^{-1},\ (x^{-1},x)$
$(2)\cap(10): y^2y^{-1},\ (y^{-1},y)$
$(3)\cap(12): z^2z^{-1},\ (z^{-1},z)$
$(5)\cap(4): xrxr^2,\ (r,xrx)$
$(6)\cap(4): yryr^2,\ (r,yry)$

$(7)\cap(4): zrzr^2,\ (r,zrz)$
$(8)\cap(9): xx^{-1}x,\ (x,x)$
$(9)\cap(5): x^{-1}xrxr,\ (rxr,x^{-1})$
$(10)\cap(11): yy^{-1}y,\ (y,y)$
$(11)\cap(6): y^{-1}yryr,\ (ryr,y^{-1})$

$(7)\cap(14): zrzrr^{-1},\ (r^{-1},zrz)$
$(9)\cap(1): x^{-1}x^2,\ (x,x^{-1})$
$(9)\cap(8): x^{-1}xx^{-1},\ (x^{-1},x^{-1})$
$(11)\cap(2): y^{-1}y^2,\ (y,y^{-1})$
$(11)\cap(10): y^{-1}yy^{-1},\ (y^{-1},y^{-1})$

$(12)\cap(13): zz^{-1}z,\ (z,z)$
$(13)\cap(7): z^{-1}zrzr,\ (rzr,z^{-1})$
$(14)\cap(15): rr^{-1}r,\ (r,r)$
$(15)\cap(14): r^{-1}rr^{-1},\ (r^{-1},r^{-1})$

$(13)\cap(3): z^{-1}z^2,\ (z,z^{-1})$
$(13)\cap(12): z^{-1}zz^{-1},\ (z^{-1},z^{-1})$
$(15)\cap(4): r^{-1}r^2,\ (r,r^{-1})$

Yukarıdaki verilen bütün kritik çiftlerden aşağıdaki yeni kurallar elde edilmiştir.

(16) $rxr \to x$ (17) $x^{-1} \to x$ (18) $ryr \to y$ (19) $y^{-1} \to y$
(20) $rzr \to z$ (21) $z^{-1} \to z$ (22) $r^{-1} \to r$ (23) $xrx \to r$
(24) $xrx \to r^{-1}$ (25) $yry \to r$ (26) $yry \to r^{-1}$ (27) $zrz \to r$
(28) $zrz \to r^{-1}$ (29) $rxr \to x^{-1}$ (30) $ryr \to y^{-1}$ (31) $rzr \to z^{-1}$

Elde edilen bu yeni kurallardan (16) kuralı (5)'i, (17) kuralı (8) ve (9)'u, (18) kuralı (6)'yı, (19) kuralı (10) ve (11)'i, (20) kuralı (7)'yi, (21) kuralı (12) ve (13)'ü, (22) kuralı (14) ve (15)'i, (24) kuralı (23)'ü, (26) kuralı (25)'i, (28) kuralı (27)'yi, (29) kuralı (16)'yı, (30) kuralı (18)i, (31) kuralı (20) kuralını indirger. Yani kritik çiftlerin aynı kelime olacak şekilde sonuçlanmasını sağlar. İndirgenen kurallar silinir. İndirgenmeyen bağıntıların kendi arasında çakışmalarından elde edilen kelimeler ve kritik çiftler aşağıda verilmiştir.

$(1) \cap (24)$: x^2rx, (rx, xr^{-1})
$(2) \cap (26)$: y^2ry, (ry, yr^{-1})
$(3) \cap (28)$: z^2rz, (rz, zr^{-1})
$(4) \cap (29)$: r^2xr, (xr, rx^{-1})
$(4) \cap (30)$: r^2yr, (yr, ry^{-1})
$(4) \cap (31)$: r^2zr, (zr, rz^{-1})
$(24) \cap (1)$: xrx^2, $(r^{-1}x, xr)$
$(26) \cap (2)$: yry^2, $(r^{-1}y, yr)$
$(28) \cap (3)$: zrz^2, $(r^{-1}z, zr)$
$(29) \cap (4)$: rxr^2, $(x^{-1}r, rx)$
$(29) \cap (30)$: $rxryr$, $(x^{-1}yr, rxy^{-1})$
$(29) \cap (31)$: $rxrzr$, $(x^{-1}zr, rxz^{-1})$
$(30) \cap (4)$: ryr^2, $(y^{-1}r, ry)$
$(30) \cap (29)$: $ryrxr$, $(y^{-1}xr, ryx^{-1})$
$(30) \cap (31)$: $ryrzr$, $(y^{-1}zr, ryz^{-1})$
$(31) \cap (4)$: rzr^2, $(z^{-1}r, rz)$
$(31) \cap (29)$: $rzrxr$, $(z^{-1}xr, rzx^{-1})$
$(31) \cap (30)$: $rzryr$, $(z^{-1}yr, rzy^{-1})$

Elde edilen yeni kurallar aşağıda verilmiştir.

(32) $rx \to xr$ (33) $ry \to yr$ (34) $rz \to zr$ (35) $rxy \to xyr$ (36) $rxz \to xzr$
(37) $ryx \to yxr$ (38) $ryz \to yzr$ (39) $rzx \to zxr$ (40) $rzy \to zyr$

Yukarıdaki yeni kurallardan (32) kuralı $(24),(29),(35)$ ve (36) kurallarını , (33) kuralı $(26),(30),(37)$ ve (38) kurallarını, (34) kuralı ise $(28),(31),(39)$ ve (40) kurallarını indirger. İndirgenen kurallar silinir.

İndirgenmeyen kuralların çakışmasından elde edilen çakışan kelimeler mevcut kurallarla çözülebilir. Bu çözümlemeler aşağıda gösterilmiştir.

$(4)\cap(32): r^2x \to \begin{Bmatrix} rxr\to xr^2\to x \\ x \end{Bmatrix}$ $\quad$ $(32)\cap(1): rx^2 \to \begin{Bmatrix} r \\ xrx\to x^2r\to r \end{Bmatrix}$

$(4)\cap(33): r^2y \to \begin{Bmatrix} ryr\to yr^2\to y \\ y \end{Bmatrix}$ $\quad$ $(33)\cap(2): ry^2 \to \begin{Bmatrix} r \\ yry\to y^2r\to r \end{Bmatrix}$

$(4)\cap(34): r^2z \to \begin{Bmatrix} rzr\to zr^2\to z \\ z \end{Bmatrix}$ $\quad$ $(34)\cap(3): rz^2 \to \begin{Bmatrix} r \\ zrz\to z^2r\to r \end{Bmatrix}$

Sonuç olarak; $\overline{H}(\lambda_2)$ genişletilmiş Hecke grubunun monoid sunuşu için *tam yeniden yazma sistemi* aşağıdaki kurallardan oluşmaktadır.

(1) $x^2 \to 1$
(2) $y^2 \to 1$
(3) $z^2 \to 1$
(4) $r^2 \to 1$
(17) $x^{-1} \to x$
(19) $y^{-1} \to y$
(21) $z^{-1} \to z$
(22) $r^{-1} \to r$
(32) $rx \to xr$
(33) $ry \to yr$
(34) $rz \to zr$

$\overline{H}(\lambda_2)$ Genişletilmiş Hecke Grubunun Normal Form Yapısı:

$\overline{H}(\lambda_2)$ genişletilmiş Hecke grubunun yukarıdaki *tam yeniden yazma sistemi*ndeki 11 bağıntı kullanılarak elemanlarının normal formu Wr^{ε} biçiminde elde edilmektedir. Burada $\varepsilon = 0,1$ ve W kelimesi $x, y \; ve \; z$ üreteç elemanlarından oluşan indirgenmiş bir kelimedir.

Sonuç olarak; $\overline{H}(\lambda_2)$ genişletilmiş Hecke grubu (monoid sunuşu) için *kelime problemi* çözülebilirdir.

$\overline{H}(\lambda_2)$ genişletilmiş Hecke grubunun yeniden yazma sisteminin uygulaması *idrel* paket programı kullanılarak aşağıda verilmiştir.

GAP

```
gap> F:=FreeGroup(4);
<free group on the generators [ f1, f2, f3, f4 ]>
gap> x:=F.1;;y:=F.2;;z:=F.3;;r:=F.4;;
gap> rels:=[x^2,y^2,z^2,r^2,(x*r)^2,(y*r)^2,(z*r)^2];;
gap> q2:=F/rels;;
gap> SetName(q2,"q2");
gap> frq2:=FreeRelatorGroup(q2);
    q2_R
gap> GeneratorsOfGroup(frq2);
    [ q2_R1, q2_R2, q2_R3, q2_R4, q2_R5, q2_R6, q2_R7 ]
gap> frhomq2:=FreeRelatorHomomorphism(q2);
    [ q2_R1, q2_R2, q2_R3, q2_R4, q2_R5, q2_R6, q2_R7 ] -> [ f1^2,
f2^2, f3^2, f4^2, (f1*f4)^2, (f2*f4)^2, (f3*f4)^2 ]
gap> mon:=MonoidPresentationFpGroup(q2);;
gap> fgmon:=FreeGroupOfPresentation(mon);
  <free group on the generators [ q2_M1, q2_M2, q2_M3, q2_M4, q2_M5,
q2_M6, q2_M7, q2_M8 ]>
gap> genfgmon:=GeneratorsOfGroup(fgmon);
    [ q2_M1, q2_M2, q2_M3, q2_M4, q2_M5, q2_M6, q2_M7, q2_M8 ]
gap> gprels:=GroupRelatorsOfPresentation(mon);
    [ q2_M1^2, q2_M2^2, q2_M3^2, q2_M4^2, (q2_M1*q2_M4)^2,
(q2_M2*q2_M4)^2, (q2_M3*q2_M4)^2 ]
gap> invrels:=InverseRelatorsOfPresentation(mon);
    [ q2_M1*q2_M5, q2_M2*q2_M6, q2_M3*q2_M7, q2_M4*q2_M8, q2_M5*q2_M1,
q2_M6*q2_M2, q2_M7*q2_M3, q2_M8*q2_M4 ]
gap> hompres:=HomomorphismOfPresentation(mon);
    [ q2_M1, q2_M2, q2_M3, q2_M4, q2_M5, q2_M6, q2_M7, q2_M8 ] -> [ f1,
f2, f3, f4, f1^-1, f2^-1, f3^-1, f4^-1 ]
gap> rws:=RewritingSystemFpGroup(q2);
    [ [ q2_M4^2, <identity ...> ], [ q2_M3^2, <identity ...> ],
    [ q2_M2^2, <identity ...> ], [ q2_M1^2, <identity ...> ],
    [ q2_M4*q2_M3, q2_M3*q2_M4 ], [ q2_M4*q2_M2, q2_M2*q2_M4 ],
    [ q2_M4*q2_M1, q2_M1*q2_M4 ], [ q2_M8, q2_M4 ],
    [ q2_M7, q2_M3 ], [ q2_M6, q2_M2 ], [ q2_M5, q2_M1 ] ]
gap> monrels:=Concatenation(gprels,invrels);
    [ q2_M1^2, q2_M2^2, q2_M3^2, q2_M4^2, (q2_M1*q2_M4)^2,
(q2_M2*q2_M4)^2, (q2_M3*q2_M4)^2, q2_M1*q2_M5, q2_M2*q2_M6,
  q2_M3*q2_M7, q2_M4*q2_M8, q2_M5*q2_M1, q2_M6*q2_M2, q2_M7*q2_M3,
q2_M8*q2_M4 ]
gap> id:=One(monrels[1]);
     <identity ...>
gap> a0:=List(monrels,a->[a,id]);
    [ [ q2_M1^2, <identity ...> ], [ q2_M2^2, <identity ...> ],
    [ q2_M3^2, <identity ...> ], [ q2_M4^2, <identity ...> ],
    [ (q2_M1*q2_M4)^2, <identity ...> ],
    [ (q2_M2*q2_M4)^2, <identity   ...> ],
    [ (q2_M3*q2_M4)^2, <identity ...> ],[ q2_M1*q2_M5, <identity ...>],
[ q2_M2*q2_M6, <identity ...> ],[ q2_M3*q2_M7, <identity ...> ],
    [ q2_M4*q2_M8, <identity ...> ],
[ q2_M5*q2_M1, <identity ...> ],[ q2_M6*q2_M2, <identity ...> ],
    [ q2_M7*q2_M3, <identity ...> ], [ q2_M8*q2_M4, <identity ...> ] ]
gap> a1:=OnePassKB(a0);
    [ [ q2_M1^2, <identity ...> ], [ q2_M2^2, <identity ...> ],
    [ q2_M3^2, <identity ...> ], [ q2_M4^2, <identity ...> ],
    [ (q2_M1*q2_M4)^2, <identity ...> ], [ (q2_M2*q2_M4)^2, <identity
...> ], [ (q2_M3*q2_M4)^2, <identity ...> ],
    [ q2_M1*q2_M5, <identity ...> ], [ q2_M2*q2_M6, <identity ...> ],
    [ q2_M3*q2_M7, <identity ...> ], [ q2_M4*q2_M8, <identity ...> ],
    [ q2_M5*q2_M1, <identity ...> ], [ q2_M6*q2_M2, <identity ...> ],
    [ q2_M7*q2_M3, <identity ...> ], [ q2_M8*q2_M4, <identity ...> ],
    [ q2_M4*q2_M1*q2_M4, q2_M1 ], [ q2_M5, q2_M1 ],
    [ q2_M4*q2_M2*q2_M4, q2_M2 ], [ q2_M6, q2_M2 ],
    [ q2_M4*q2_M3*q2_M4, q2_M3 ], [ q2_M7, q2_M3 ], [ q2_M8, q2_M4 ],
    [ q2_M1*q2_M4*q2_M1, q2_M4 ], [ q2_M1*q2_M4*q2_M1, q2_M8 ],
    [ q2_M2*q2_M4*q2_M2, q2_M4 ], [ q2_M2*q2_M4*q2_M2, q2_M8 ],
    [ q2_M3*q2_M4*q2_M3, q2_M4 ], [ q2_M3*q2_M4*q2_M3, q2_M8 ],
    [ q2_M5, q2_M1 ],[ q2_M4*q2_M1*q2_M4, q2_M5 ], [ q2_M6, q2_M2 ],
    [ q2_M4*q2_M2*q2_M4, q2_M6 ], [ q2_M7, q2_M3 ],
    [ q2_M4*q2_M3*q2_M4, q2_M7 ], [ q2_M8, q2_M4 ] ]
gap> Length(a1);
35
```

```
GAP

gap> a1:=RewriteReduce(a1);
    [ [ q2_M5, q2_M1 ], [ q2_M6, q2_M2 ], [ q2_M7, q2_M3 ],
    [ q2_M8,  q2_M4 ], [ q2_M1^2, <identity ...> ],
    [ q2_M2^2, <identity ...> ], [ q2_M3^2, <identity ...> ],
    [ q2_M4^2, <identity ...> ], [ q2_M1*q2_M4*q2_M1, q2_M4 ],
    [ q2_M2*q2_M4*q2_M2, q2_M4 ], [ q2_M3*q2_M4*q2_M3, q2_M4 ],
    [ q2_M4*q2_M1*q2_M4, q2_M1 ],  [ q2_M4*q2_M2*q2_M4, q2_M2 ],
    [ q2_M4*q2_M3*q2_M4, q2_M3 ] ]
gap> a2:=KnuthBendix(a1);
    [ [ q2_M5, q2_M1 ], [ q2_M6, q2_M2 ], [ q2_M7, q2_M3 ],
    [ q2_M8, q2_M4 ], [ q2_M1^2, <identity ...> ],
    [ q2_M2^2, <identity ...> ], [ q2_M3^2, <identity ...> ],
    [ q2_M4*q2_M1, q2_M1*q2_M4 ],[ q2_M4*q2_M2, q2_M2*q2_M4 ],
    [ q2_M4*q2_M3, q2_M3*q2_M4 ],[ q2_M4^2, <identity ...> ] ]
gap> Length(a2);
11
gap> a2:=RewriteReduce(a2);
    [ [ q2_M5, q2_M1 ], [ q2_M6, q2_M2 ], [ q2_M7, q2_M3 ],
    [ q2_M8, q2_M4 ], [ q2_M1^2, <identity ...> ],
    [ q2_M2^2, <identity ...> ], [ q2_M3^2, <identity ...> ],
    [ q2_M4*q2_M1, q2_M1*q2_M4 ],[ q2_M4*q2_M2, q2_M2*q2_M4 ],
    [ q2_M4*q2_M3, q2_M3*q2_M4 ], [ q2_M4^2, <identity ...> ] ]
gap> a3:=KnuthBendix(a2);
    [ [ q2_M5, q2_M1 ], [ q2_M6, q2_M2 ], [ q2_M7, q2_M3 ],
    [ q2_M8, q2_M4 ], [ q2_M1^2, <identity ...> ],
    [ q2_M2^2, <identity ...> ], [ q2_M3^2, <identity ...> ],
    [ q2_M4*q2_M1, q2_M1*q2_M4 ],  [ q2_M4*q2_M2, q2_M2*q2_M4 ],
    [ q2_M4*q2_M3, q2_M3*q2_M4 ], [ q2_M4^2, <identity ...> ] ]
gap> a3:=RewriteReduce(a3);
    [ [ q2_M5, q2_M1 ], [ q2_M6, q2_M2 ], [ q2_M7, q2_M3 ],
    [ q2_M8, q2_M4 ], [ q2_M1^2, <identity ...> ],
    [ q2_M2^2, <identity ...> ], [ q2_M3^2, <identity ...> ],
    [ q2_M4*q2_M1, q2_M1*q2_M4 ],[ q2_M4*q2_M2, q2_M2*q2_M4 ],
    [ q2_M4*q2_M3, q2_M3*q2_M4 ], [ q2_M4^2, <identity ...> ] ]
gap> Length(a3);
11
```

Genişletilmiş Hecke Grubunun Uygulaması-2

$\overline{H}(\lambda_4)$ genişletilmiş Hecke grubunun monoid sunuşu

$$\overline{H}(\lambda_4)=\left\langle x,\, y,\, r,\, x^{-1},\, y^{-1},\, r^{-1}\ ;\ x^2=y^4=r^2=1,(xr)^2=(yr)^2=1,xx^{-1}=x^{-1}x=1,\right.$$
$$\left.yy^{-1}=y^{-1}y=1,rr^{-1}=r^{-1}r=1\right\rangle$$

biçimindedir. Bu sunuşun tam yeniden yazma sistemini hesaplamak için ilk olarak kelimeler arasında uzunluk sıralaması, uzunlukları eşit olan kelimeler için ise üreteçler arasında $r^{-1}>r>y^{-1}>y>x^{-1}>x$ biçiminde harfsel sıralama seçilmektedir. Bu sıralama dikkate alınarak $\overline{H}(\lambda_4)$ grubunun yeniden yazma sistemi aşağıdaki bağıntılardan oluşmaktadır.

(1) $x^2=1$ (2) $y^4=1$ (3) $r^2=1$

(4) $(xr)^2=1$ (5) $(yr)^2=1$ (6) $xx^{-1}=1$

(7) $x^{-1}x=1$ (8) $yy^{-1}=1$ (9) $y^{-1}y=1$

(10) $rr^{-1}=1$ (11) $r^{-1}r=1$

Yukarıda $(1)-(11)$ ile verilen bağıntılardan elde edilen çakışan kelimeler ve ilgili kritik çiftler aşağıda verilmiştir.

$(1)\cap(4): x^2rxr, (rxr, x)$
$(1)\cap(6): x^2x^{-1}, (x^{-1}, x)$
$(2)\cap(5): y^4ryr, (ryr, y^3)$
$(2)\cap(8): y^4y^{-1}, (y^{-1}, y^3)$
$(3)\cap(10): r^2r^{-1}, (r^{-1}, r)$
$(4)\cap(3): xrxr^2, (r, xrx)$
$(4)\cap(10): xrxrr^{-1}, (r^{-1}, xrx)$
$(5)\cap(3): yryr^2, (r, yry)$
$(5)\cap(10): yryrr^{-1}, (r^{-1}, yry)$
$(6)\cap(7): xx^{-1}x, (x, x)$
$(7)\cap(1): x^{-1}x^2, (x, x^{-1})$
$(7)\cap(4): x^{-1}xrxr, (rxr, x^{-1})$
$(7)\cap(6): x^{-1}xx^{-1}, (x^{-1}, x^{-1})$
$(8)\cap(9): yy^{-1}y, (y, y)$
$(9)\cap(2): y^{-1}y^4, (y^3, y^{-1})$
$(9)\cap(5): y^{-1}yryr, (ryr, y^{-1})$
$(9)\cap(8): y^{-1}yy^{-1}, (y^{-1}, y^{-1})$
$(10)\cap(11): rr^{-1}r, (r, r)$
$(11)\cap(3): r^{-1}r^2, (r, r^{-1})$
$(11)\cap(10): r^{-1}rr^{-1}, (r^{-1}, r^{-1})$

Yukarıdaki verilen kritik çiftlerden aşağıdaki yeni bağıntılar elde edilmiştir.

(12) $rxr \to x$ (13) $x^{-1} \to x$ (14) $ryr \to y^3$ (15) $y^3 \to y^{-1}$
(16) $r^{-1} \to r$ (17) $xrx \to r$ (18) $xrx \to r^{-1}$ (19) $yry \to r$
(20) $yry \to r^{-1}$ (21) $rxr \to x^{-1}$ (22) $ryr \to y^{-1}$

(12) kuralı (4) kuralını, (13) kuralı (6) ve (7) kurallarını, (14) kuralı (5) ve (22) kurallarını, (15) kuralı (2) kuralını, (16) kuralı (10) ve (11) kurallarını , (18) kuralı (17) kuralını, (20) kuralı (19) kuralını ve (21) kuralı da (12) kuralını indirgediği için bu kurallar silinir.

Yeni elde edilen kurallarla çakışmalarından elde edilen kelimeler ve kritik çiftler aşağıda verilmiştir.

$(1)\cap(18): x^2rx, (rx, xr^{-1})$ $(15)\cap(8): y^3y^{-1}, (y^{-2}, y^2)$

$(3)\cap(14): r^2yr, (yr, ry^3)$ $(15)\cap(20): y^3ry, (y^{-1}ry, y^2r^{-1})$

$(3)\cap(21): r^2xr, (xr, rx^{-1})$ $(18)\cap(1): xrx^2, (r^{-1}x, xr)$

$(9)\cap(15): y^{-1}y^3, (y^2, y^{-2})$ $(20)\cap(8): yryy^{-1}, (r^{-1}y^{-1}, yr)$

$(9)\cap(20): y^{-1}yry, (ry, y^{-1}r^{-1})$ $(20)\cap(15): yry^3, (r^{-1}y^2, yry^{-1})$

$(14)\cap(3): ryr^2, (y^3r, ry)$ $(21)\cap(3): rxr^2, (x^{-1}r, rx)$

$(14)\cap(21): ryrxr, (y^3xr, ryx^{-1})$ $(21)\cap(14): rxryr, (x^{-1}yr, rxy^3)$

Bu kritik çiftlerden elde edilen yeni kurallar;

(23) $rx \to xr$ (24) $ry^{-1} \to yr$ (25) $y^{-2} \to y^2$

(26) $ry \to y^{-1}r$ $(ry \to yr)$ (27) $ryx \to y^{-1}xr$ (28) $y^{-1}ry \to y^2r$

(29) $ry^2 \to yry^{-1}$ (30) $rxy^{-1} \to xyr$

biçimindedir. Bu kurallardan (23) kuralı $(18),(21),(30)$ kurallarını ve (26) kuralı $(14),(20),(27),(28),(29)$ kurallarını indirger. İndirgenen kurallar silinir.

$(23),(24),(25),(26)$ kuralları ile önceki indirgenemeyen kuralların çakışmasından elde edilen çakışan kelimelerin kritik çiftleri aynı sonuçları verecektir. Bunlardan bazıları aşağıdaki gibidir.

$$(24)\cap(25): ry^{-2} \to \begin{Bmatrix} yry^{-1} \to y^2r \\ ry^2 \to yry^{-1} \to y^2r \end{Bmatrix}$$

$$(26)\cap(15): ry^3 \to \begin{Bmatrix} y^{-1}ry^2 \to y^{-1}yry^{-1} \to ry^{-1} \to yr \\ ry^{-1} \to yr \end{Bmatrix}$$

Yukarıda $(1)-(30)$ ile verilen yeniden yazma sistemindeki bağıntılardan bazıları birbirlerinden elde edilebildiği için bu bağıntıların silinmesi gerekmektedir. Uygun işlemler yapıldıktan sonra $\overline{H}(\lambda_4)$ grubunun monoid sunuşu için *tam yeniden yazma sistemi* aşağıdaki kurallardan oluşmaktadır.

(1) $x^2 = 1$ (3) $r^2 = 1$ (8) $yy^{-1} = 1$ (9) $y^{-1}y = 1$

(13) $x^{-1} = x$ (15) $y^3 = y^{-1}$ (16) $r^{-1} = r$ (23) $rx = xr$

(24) $ry^{-1} = yr$ (25) $y^{-2} = y^2$ (26) $ry = yr$

$\overline{H}(\lambda_4)$ Genişletilmiş Hecke Grubunun Elemanlarının Normal Form Yapısı: $\overline{H}(\lambda_4)$ genişletilmiş Hecke grubunun tam yeniden yazma sistemindeki bağıntılar kullanılarak elde edilen elemanlarının normal formu Wr^{ε} biçimindedir. Burada $\varepsilon = 0,1$ ve W kelimesi x, y, y^{-1} üreteç elemanlarından oluşan indirgenmiş bir kelimedir.

Sonuç olarak; $\overline{H}(\lambda_4)$ genişletilmiş Hecke grubu (monoid sunuşu) için *kelime problemi çözülebilirdir.*

idrel paketi kullanılarak $\overline{H}(\lambda_4)$ grubunun Knuth-Bendix uygulaması aşağıda verilmiştir.

```
                                    GAP
gap> F:=FreeGroup(3);;
gap> x:=F.1;;y:=F.2;;r:=F.3;;
gap> rels:=[y^4,x^2,r^2,(x*r)^2,(y*r)^2];;
gap> q0:=F/rels;;
gap> SetArrangementOfMonoidGenerators(q0,[1,-1,2,-2,3,-3,4,-4]);
gap> SetName(q0,"q0");
gap> frq0:=FreeRelatorGroup(q0);;
gap> genq0:=GeneratorsOfGroup(q0);;
gap> genfrq0:=GeneratorsOfGroup(frq0);;
gap> frhomq0:=FreeRelatorHomomorphism(q0);
    [ q0_R1, q0_R2, q0_R3, q0_R4, q0_R5 ] -> [ f2^4, f1^2, f3^2,
(f1*f3)^2, (f2*f3)^2 ]
gap> monq0:=MonoidPresentationFpGroup(q0);
monoid presentation with group relators [ q0_M3^4, q0_M1^2, q0_M5^2,
(q0_M1*q0_M5)^2, (q0_M3*q0_M5)^2 ]
gap> fgmon:=FreeGroupOfPresentation(monq0);
   <free group on the generators [ q0_M1, q0_M2, q0_M3, q0_M4, q0_M5,
q0_M6 ]>
gap> genfgmon:=GeneratorsOfGroup(fgmon);;
gap> gprels:=GroupRelatorsOfPresentation(monq0);;
```

```
                                    GAP
gap> Print("gprels=\n");PrintOneItemPerLine(gprels);
gprels=
   q0_M3^4
   q0_M1^2
   q0_M5^2
   (q0_M1*q0_M5)^2
   (q0_M3*q0_M5)^2
gap> invrels:=InverseRelatorsOfPresentation(monq0);;
gap> Print("invrels=\n");PrintOneItemPerLine(invrels);
invrels=
   q0_M1*q0_M2
   q0_M3*q0_M4
   q0_M5*q0_M6
   q0_M2*q0_M1
   q0_M4*q0_M3
   q0_M6*q0_M5
gap> hompres:=HomomorphismOfPresentation(monq0);
[ q0_M1, q0_M2, q0_M3, q0_M4, q0_M5, q0_M6 ] -> [ f1, f1^-1, f2, f2^-1,
f3, f3^-1 ]
gap> rws:=RewritingSystemFpGroup(q0);;
gap> Print("rws=\n");PrintOneItemPerLine(rws);
rws=
 [ q0_M3*q0_M4, <identity ...> ]
 [ q0_M4*q0_M3, <identity ...> ]
 [ q0_M3^3, q0_M4 ]
 [ q0_M5^2, <identity ...> ]
 [ q0_M1^2, <identity ...> ]
 [ q0_M5*q0_M4, q0_M3*q0_M5 ]
 [ q0_M5*q0_M3, q0_M4*q0_M5 ]
 [ q0_M5*q0_M1, q0_M1*q0_M5 ]
 [ q0_M4^2, q0_M3^2 ]
 [ q0_M6, q0_M5 ]
 [ q0_M2, q0_M1 ]
gap> l0:=InitialLoggedRules(q0);
 [ [ q0_M1*q0_M2, [  ], <identity ...> ], [ q0_M3*q0_M4, [  ],  <identity
...> ], [ q0_M5*q0_M6, [  ], <identity ...> ],
 [ q0_M2*q0_M1, [  ], <identity ...> ], [ q0_M4*q0_M3, [  ], <identity
...> ], [ q0_M6*q0_M5, [  ], <identity ...> ],
 [ q0_M3^4, [ [ 7, <identity ...> ] ], <identity ...> ],
 [ q0_M1^2, [ [ 8, <identity ...> ] ], <identity ...> ],
 [ q0_M5^2, [ [ 9, <identity ...> ] ], <identity ...> ],
 [ (q0_M1*q0_M5)^2, [ [ 10, <identity ...> ] ], <identity ...> ],
 [ (q0_M3*q0_M5)^2, [ [ 11, <identity ...> ] ], <identity ...> ] ]
gap> logrws:=LoggedRewritingSystemFpGroup(q0);;
```

GAP

```
gap> Print("logrws=\n");PrintOneItemPerLine(logrws);
logrws=
 [ q0_M5^2, [ [ 9, <identity ...> ] ], <identity ...> ]
 [ q0_M4*q0_M3, [  ], <identity ...> ]
 [ q0_M3*q0_M4, [  ], <identity ...> ]
 [ q0_M1^2, [ [ 8, <identity ...> ] ], <identity ...> ]
 [ q0_M3^3, [ [ 7, <identity ...> ] ], q0_M4 ]
 [ q0_M5*q0_M4, [ [ -11, q0_M3*q0_M6 ], [ 9, <identity ...> ] ], q0_M3*q0_M5
]
 [ q0_M5*q0_M3, [ [ -9, q0_M4*q0_M6 ], [ 11, q0_M3 ] ], q0_M4*q0_M5 ],[
q0_M5*q0_M1, [ [ -9, q0_M2*q0_M6 ], [ 10, q0_M1 ],         [ -8, <identity
...> ] ], q0_M1*q0_M5 ]
[ q0_M4^2, [ [ -7, <identity ...> ] ], q0_M3^2 ]
[ q0_M6, [ [ -9, <identity ...> ] ], q0_M5 ]
[ q0_M2, [ [ -8, <identity ...> ] ], q0_M1 ]
gap> monrels:=Concatenation(gprels,invrels);;
gap> id:=One(monrels[1]);;
gap> r0:=List(monrels,r->[r,id]);;
gap> r1:=OnePassKB(r0);
  [ [ q0_M3^4, <identity ...> ], [ q0_M1^2, <identity ...> ],
  [ q0_M5^2, <identity ...> ], [ (q0_M1*q0_M5)^2, <identity ...> ],
  [ (q0_M3*q0_M5)^2, <identity ...> ], [ q0_M1*q0_M2, <identity ...> ],
  [ q0_M3*q0_M4, <identity ...> ], [ q0_M5*q0_M6, <identity ...> ],
  [ q0_M2*q0_M1, <identity ...> ],[ q0_M4*q0_M3, <identity ...> ],
  [ q0_M6*q0_M5, <identity ...> ], [ q0_M5*q0_M3*q0_M5, q0_M3^3 ],
  [ q0_M3^3, q0_M4 ], [ q0_M5*q0_M1*q0_M5, q0_M1 ], [ q0_M2, q0_M1 ],
  [ q0_M6, q0_M5 ], [ q0_M1*q0_M5*q0_M1, q0_M5 ],
  [ q0_M1*q0_M5*q0_M1, q0_M6 ], [ q0_M3*q0_M5*q0_M3, q0_M5 ],
  [ q0_M3*q0_M5*q0_M3, q0_M6 ], [ q0_M2, q0_M1 ],
  [ q0_M5*q0_M1*q0_M5, q0_M2 ], [ q0_M3^3, q0_M4 ],
  [ q0_M5*q0_M3*q0_M5, q0_M4 ], [ q0_M6, q0_M5 ] ]
gap> Length(r1);
25
gap> r2:=KnuthBendix(r1);
  [ [ q0_M2, q0_M1 ], [ q0_M6, q0_M5 ], [ q0_M1^2, <identity ...> ],
  [ q0_M3*q0_M4, <identity ...> ],
  [ q0_M4*q0_M3, <identity ...> ], [ q0_M4^2, q0_M3^2 ],
  [ q0_M5*q0_M1, q0_M1*q0_M5 ], [ q0_M5*q0_M3, q0_M4*q0_M5 ],
  [ q0_M5*q0_M4, q0_M3*q0_M5 ], [ q0_M5^2, <identity ...> ],
  [ q0_M3^3, q0_M4 ] ]
gap> Length(r2);
11
gap> r3:=KnuthBendix(r2);
  [ [ q0_M2, q0_M1 ], [ q0_M6, q0_M5 ], [ q0_M1^2, <identity ...> ],
  [ q0_M3*q0_M4, <identity ...> ], [ q0_M4*q0_M3, <identity ...> ],
  [ q0_M4^2, q0_M3^2 ], [ q0_M5*q0_M1, q0_M1*q0_M5 ],
  [ q0_M5*q0_M3, q0_M4*q0_M5 ], [ q0_M5*q0_M4, q0_M3*q0_M5 ],
  [ q0_M5^2, <identity ...> ], [ q0_M3^3, q0_M4 ] ]
gap> Length(r3);
11
```

Yukarıdaki uygulamada, $\overline{H}(\lambda_4)$ genişletilmiş Hecke grubunun

$$\overline{H}(\lambda_4)=\left\langle x,\ y,\ r,\ x^{-1},\ y^{-1},\ r^{-1}\ ;\ x^2=y^4=r^2=1,(xr)^2=(yr)^2=1,xx^{-1}=x^{-1}x=1,\right.$$
$$\left.yy^{-1}=y^{-1}y=1,rr^{-1}=r^{-1}r=1\right\rangle$$

monoid sunuşu için yeniden yazma sistemi ve Knuth-Bendix algoritması uygulanarak tam yeniden yazma sistemi elde edilmiştir. *Length(r2)*; komutu verilerek de bu kuralların 11 tane olduğu görülmüştür. Program çıktısındaki $q0\_M1,\ q0\_M2,\ q0\_M3,\ q0\_M4,\ q0\_M5$ ve $q0\_M6$ ifadeleri $x,\ x^{-1},\ y,\ y^{-1},\ r$ ve r^{-1} üreteçlerini göstermektedir.

5.4 Genişletilmiş ve Genelleştirilmiş Hecke Grubunun Kelime Problemi

2.3 alt bölümde verilen genişletilmiş ve genelleştirlmiş Hecke grubunun $\overline{H}(p_1,...,p_n)=\left\langle x_i\ ,\ r\ ;\ x_i^{p_i}=r^2=(x_ir)^2=1\right\rangle\cong D_{p_1}*_{\mathbb{Z}_2}...*_{\mathbb{Z}_2}D_{p_n}\ (1\le i\le n)$ sunuşunda $p_1=p_2=...=p_n=2$ alınması durumunda elde edilen $\overline{H}(2,2,...,2)$ grubunun sunuşu

$$\overline{H}\left(2,2,...,2\right)=\left\langle\ x_i\ ,\ r\ \ ;\ \ x_i^{\,2}=1,\ r^2=1,\ (x_ir)^2=1\right\rangle$$

biçimindedir. Bu grubun monoid sunuşu,

$$\left\langle\ x_i,\ x_i^{-1},\ r,\ r^{-1}\ \ ;\ \ x_i^{\,2}=1,\ r^2=1,\ \left(x_ir\right)^2=1,\ x_ix_i^{-1}=1,\ x_i^{-1}x_i=1,\ rr^{-1}=1,\ r^{-1}r=1\right\rangle$$

şeklindedir. Bu genişletilmiş ve genelleştirlmiş Hecke grubunun monoid sunuşunun tam yeniden yazma sistemini hesaplamak için ilk olarak kelimeler arasında uzunluk sıralaması, uzunlukları eşit olan kelimeler için ise üreteçler arasında $r^{-1}>r>x_i^{-1}>x_i$ biçiminde harfsel sıralama seçilir. Bu sıralama dikkate alınarak

$(1)\ x_i^{\,2}=1$ $(2)\ r^2=1$ $(3)\ \left(x_ir\right)^2=1$ $(4)\ x_ix_i^{-1}=1$
$(5)\ x_i^{-1}x_i=1$ $(6)\ rr^{-1}=1$ $(7)\ r^{-1}r=1$

kuralları genelleştirilmiş Hecke grubunun yeniden yazma sistemini oluşturmaktadır. Kuralların çakışmasından elde edilen çakışık kelimeler ve kritik çiftler aşağıda verilmiştir.

$(1)\cap(3)$: $x_i^2 r x_i r, (r x_i r, x_i)$
$(1)\cap(4)$: $x_i^2 x_i^{-1}, (x_i^{-1}, x_i)$
$(2)\cap(6)$: $r^2 r^{-1}, (r^{-1}, r)$
$(3)\cap(2)$: $x_i r x_i r^2, (r, x_i r x_i)$
$(3)\cap(6)$: $x_i r x_i r r^{-1}, (r^{-1}, x_i r x_i)$
$(4)\cap(5)$: $x_i x_i^{-1} x_i, (x_i, x_i)$
$(5)\cap(1)$: $x_i^{-1} x_i^2, (x_i, x_i^{-1})$
$(5)\cap(3)$: $x_i^{-1} x_i r x_i r, (r x_i r, x_i^{-1})$
$(5)\cap(4)$: $x_i^{-1} x_i x_i^{-1}, (x_i^{-1}, x_i^{-1})$
$(6)\cap(7)$: $r r^{-1} r, (r, r)$
$(7)\cap(2)$: $r^{-1} r^2, (r, r^{-1})$
$(7)\cap(6)$: $r^{-1} r r^{-1}, (r^{-1}, r^{-1})$

Bu kritik çiftlerden elde edilen yeni bağıntılar aşağıda verilmiştir.

$(8)\ r x_i r \to x_i$ $(9)\ x_i^{-1} \to x_i$ $(10)\ r^{-1} \to r$
$(11)\ x_i r x_i \to r$ $(12)\ x_i r x_i \to r^{-1}$ $(13)\ r x_i r \to x_i^{-1}$

Bu kurallardan (8) kuralı (3) kuralını, (9) kuralı (4) ve (5) kurallarını, (10) kuralı $(6),(7)$ kurallarını, (12) kuralı (11) kuralını, (13) kuralı da (8) kuralını indirger. İndirgenen kuralların silinmesinden sonra geriye kalan kuralların çakışmasından bulunan kelimeler ve ilgili kritik çiftler aşağıda verilmiştir.

$(1)\cap(12)$: $x_i^2 r x_i, (r x_i, x_i r^{-1})$
$(2)\cap(13)$: $r^2 x_i r, (x_i r, r x_i^{-1})$
$(12)\cap(1)$: $x_i r x_i^2, (r^{-1} x_i, x_i r)$
$(13)\cap(2)$: $r x_i r^2, (x_i^{-1} r, r x_i)$

Bu çakışmalardan yeni bir bağıntı oluşur. Bu bağıntı; $(14)\ r x_i \to x_i r$ dır. (14) kuralı (12) ve (13) kurallarını indirger. Dolayısıyla (12) ve (13) kuralları silinir. $(1),(2),(9),(10)$ ve (14) kurallarının çakışmasından aşağıdaki sonuçlar elde edilmiştir.

$(2)\cap(14)$: $r^2x_i \to \left\{ \begin{matrix} rx_ir \to x_ir^2 \to x_i \\ x_i \end{matrix} \right\}$

$(14)\cap(1)$: $rx_i^{\,2} \to \left\{ \begin{matrix} r \\ x_irx_i \to x_i^{\,2}r \to r \end{matrix} \right\}$

Görüldüğü gibi kritik çiftler aynı kelimeye indirgenmiştir. Sonuç olarak; $\overline{H}(2,2,...,2)$ genişletilmiş ve genelleştirilmiş Hecke grubunun monoid sunuşu için *tam yeniden yazma sistemi* aşağıdaki kurallardan oluşmaktadır.

$(1)\ x_i^{\,2} \to 1 \quad (2)\ r^2 \to 1 \quad (9)\ x_i^{-1} \to x_i \quad (10)\ r^{-1} \to r \quad (14)\ rx_i \to x_ir$

$\overline{H}(2,2,...,2)$ Genişletilmiş ve Genelleştirilmiş Hecke Grubunun Normal Form Yapısı:

$\overline{H}(2,2,...,2)$ genişletilmiş ve genelleştirilmiş Hecke grubunun elemanlarının normal formu Wr^{ε} biçiminde elde edilmektedir. Burada $\varepsilon = 0,1$ ve W kelimesi x_i $(1 \le i \le n)$ üreteç elemanlarından oluşan indirgenmiş bir kelimedir.

Sonuç olarak; $\overline{H}(2,2,...,2)$ genişletilmiş ve genelleştirilmiş Hecke grubu (monoid sunuşu) için *kelime problemi* çözülebilirdir.

Not: Bu alt bölümün önemi burada el ile i indislerine bağlı olarak yapılan yeniden yazma sisteminin hesaplanmasında GAP programının dolayısıyla *idrel* paketinin uygulanamamasıdır. Çünkü GAP kapsamında yapılan uygulamalar çalışılan cebirsel yapının sonlu sayıda üretece bağlı olmasıyla sınırlıdır.

6. KAYITLI YENİDEN YAZMA SİSTEMİ

Bu bölümde, kayıtlı yeniden yazma sistemi, onun fonksiyonları ve bu fonksiyonların uygulamaları verilmiştir. Bu bölümde verilen fonksiyonlar *idrel* paketi kapsamında çalışmaktadır.

Kayıtlı yeniden yazma sistemi grup sunuşu ile ilişkili bir yapıdır. Her kayıtlı yeniden yazma kuralı, standart yeniden yazma kurallarına ek olarak bir kayıt ya da grubun orijinal bağıntıları cinsinden kuralı açıklayan bir kayıt bileşeni içerir. Bu şekildeki bir kural

$$[\, u \,,\, [\, L_1 \,, L_2 \,, \ldots, L_k \,] \,,\, v \,]$$

şeklinde bir sıralı üçlü olarak yazılır. Burada $[\, u \,, v \,]$ yeniden yazma kuralıdır ve n_i grup bağıntısı ve w_i bir kelime olmak üzere $L_i = [\, n_i \,, w_i \,]$ biçimindedir. Bu üç bileşen $u = n_1^{w_1} \ldots n_k^{w_k} v$ birimini sağlar.

Not: Bu bölümde elde edilen tüm uygulamalar 5.2 alt bölümde verilen $\mathbb{Z}_3 \times \mathbb{Z}_4$ direkt çarpım grubu üzerine yapılmıştır.

6.1 Kayıtlı Knuth - Bendix Hesaplaması

6.1.1 LoggedOnePassKB fonksiyonu

LoggedOnePassKB fonksiyonu, *OnePassKB* fonksiyonu ile elde edilen yeniden yazma sistemini tamamlamak için eklenen tüm kuralları bulur. Bu fonksiyonu *loggedrules* fonksiyonuna eklemenin bir sonucu tanımlanan monoidi değiştirmeden sisteme yeni kayıtlı kurallar eklemektir.

Aşağıdaki örnekte, ilk olarak 5.2 alt bölümde incelenen $\mathbb{Z}_3 \times \mathbb{Z}_4$ direkt çarpım grubu için GAP uygulamasında $"q6"$ ile tanımlanan sunuş, kayıtlı kuralların başlangıç kümesine dönüştürülmektedir. Ardından Knuth-Bendix'in bir geçişi uygulanmaktadır. Fonksiyon listelerin iki elemanlı bir listesini vermektedir. İlk eleman, kaydedilen kuralların bir kümesidir ve ikinci eleman, r-dizilerinin boş bir listesidir.

```
                                    GAP
gap> l0:=ListWithIdenticalEntries(7,0);;
gap> for j in [1..7] do
> r:=r0[j];
> if (j<3) then
> l0[j]:=[r[1],[ [j,id] ],r[2]];
> else
> l0[j]:=[r[1],[ ],r[2]];
> fi;
> od;
gap> l0;
  [ [ q6_M1^3, [ [ 1, <identity ...> ] ], <identity ...> ],
  [ q6_M2^4, [ [ 2, <identity ...> ] ], <identity ...> ],
  [ q6_M1*q6_M2*q6_M3*q6_M4, [  ], <identity ...> ],
  [ q6_M1*q6_M3, [  ], <identity ...> ],
  [ q6_M2*q6_M4, [  ], <identity ...> ],
  [ q6_M3*q6_M1, [  ], <identity ...> ],
  [ q6_M4*q6_M2, [  ], <identity ...> ] ]
gap> l1:=LoggedOnePassKB(q6,l0);;
gap> Length(l1[1]);
14
gap> l1[1][11];
[ q6_M1*q6_M2*q6_M3, [  ], q6_M2 ]
```

Yukarıdaki uygulamada son işlem olarak verilen "l1[1][11] adımı bize "$q6$" sunuşu için yapılan *OnePassKB(r0)* işlemindeki 11. bağıntıyı vermektedir.

6.1.2 LoggedKnuth-Bendix fonksiyonu

LoggedRewriteReduce fonksiyonu, kayıtlı yeniden yazma sisteminden gerekli olmayan kuralları kaldırır. Bu fonksiyon *RewriteReduce* fonksiyonu ile aynı prensipte çalışır.

LoggedKnuthBendix fonksiyonu, yeni kurallar eklenmeye ve gereksiz kurallar ilave edilmeyene kadar *LoggedOnePassKB* ve *LoggedRewriteReduce* fonksiyonlarını tekrar tekrar uygular. Çıktı indirgenmiş tam kayıtlı yeniden yazma sistemidir.

```
                                    GAP
gap> l8:=LoggedRewriteReduce(q6,l1[1]);;
gap> Perform(l8,Display);
[ q6_M1^2, [ [ 1, <identity ...> ] ], q6_M3 ]
[ q6_M1*q6_M3, [  ], <identity ...> ]
[ q6_M2*q6_M4, [  ], <identity ...> ]
[ q6_M3*q6_M1, [  ], <identity ...> ]
[ q6_M4*q6_M2, [  ], <identity ...> ]
[ q6_M1*q6_M2*q6_M3, [  ], q6_M2 ]
[ q6_M2^3, [ [ 2, <identity ...> ] ], q6_M4 ]
[ q6_M2*q6_M3*q6_M4, [ [ -1, <identity ...> ], [ 1, <identity ...> ] ],
q6_M3 ]
gap> Length(l8);
8
gap> l2:=LoggedKnuthBendix(q6,l8);;
gap> l2[1];
  [ [ q6_M1^2, [ [ 1, <identity ...> ] ], q6_M3 ],
  [ q6_M1*q6_M3, [  ], <identity ...> ],
  [ q6_M2*q6_M1, [  ], q6_M1*q6_M2 ],
  [ q6_M2*q6_M4, [  ], <identity ...> ],
  [ q6_M3*q6_M1, [  ], <identity ...> ],
  [ q6_M3*q6_M2, [  ], q6_M2*q6_M3 ],
  [ q6_M3^2, [ [ -1, <identity ...> ] ], q6_M1 ],
  [ q6_M4*q6_M1, [ [ -2, <identity ...> ], [ 2, q6_M1^-1 ] ], q6_M1*q6_M4
],
  [ q6_M4*q6_M2, [  ], <identity ...> ],
  [ q6_M4*q6_M3, [  ], q6_M3*q6_M4 ],
  [ q6_M4^2, [ [ -2, <identity ...> ] ], q6_M2^2 ],
  [ q6_M1*q6_M2*q6_M3, [  ], q6_M2 ],
  [ q6_M2^3, [ [ 2, <identity ...> ] ], q6_M4 ],
  [ q6_M2*q6_M3*q6_M4, [ [ -1, <identity ...> ],
  [ 1, <identity ...> ] ], q6_M3 ],
  [ q6_M1*q6_M2^2*q6_M3, [  ], q6_M2^2 ] ]
gap> Length(l2[1]);
15
```

6.2 Bir Kelimenin Kayıtlı İndirgenmesi

6.2.1 LoggedReduceWordKB fonksiyonu

Bir kelime ve yeniden yazma sistemi verildiğinde, *LoggedOnePassReduceWord* fonksiyonu bir kelimenin bir kez indirgemesini yapar ve bunu kullanılan kuralın kayıt bölümünü ve yer değiştirilen bölümün orijinal kelimesindeki kısmı kullanarak kaydeder.

LoggedReduceWordKB fonksiyonu, kelime artık indirgenemeyecek duruma gelene kadar *OnePassLoggedReduceWork* fonksiyonunu tekrar tekrar uygular. İndirgemenin

her adımı orijinal kelimenin orijinal bağıntılar ve indirgenemez kelime cinsinden nasıl ifade edilebileceğini göstererek kaydedilir.

Loggedrules tamamlandığında, indirgenmiş kelime o grup elemanı için tek normal formdur. İndirgemenin kaydı, kuralların uygulanma sırasına bağlıdır.

ShorterLoggedrule fonksiyonu, *ShorterRule* fonksiyonu ile aynı kriterleri kullanarak kaydedilen bir kuralın diğerinden daha iyi olup olmadığına karar verir.

```
                                    GAP
gap> w0;
q6_M2^5*q6_M1^5
gap> lw1:=LoggedOnePassReduceWord(w0,l0);
   [ [ [ 1, q6_M2^-5 ], [ 2, <identity ...> ] ], q6_M2*q6_M1^2 ]
gap> lw2:=LoggedReduceWordKB(w0,l0);
   [ [ [ 1, q6_M2^-5 ], [ 2, <identity ...> ] ], q6_M2*q6_M1^2 ]
gap> lw2:=LoggedReduceWordKB(w0,l2[1]);
   [ [ [ 1, q6_M2^-5 ], [ 2, <identity ...> ],
   [ 1, q6_M2^-2*q6_M4^-1 ] ], q6_M2*q6_M3 ]
```

6.2.2 LoggedRewritingSystemFpGroup fonksiyonu

Bir grup sunuşu verildiğinde, *LoggedRewritingSystemFpGroup* fonksiyonu, bağıntılara bağlı olarak kaydedilen yeniden yazma sistemini belirler. Bir grup sunuşu ile ilişkili ilk kaydedilen yeniden yazma sistemi, iki tür kuraldan oluşur. Bunlar, monoid sunuşundaki iki tür kuralın kayıtlı versiyonudur. Grubun her reel bağıntısı için;

$$[\, rel\, ,\, [\, [\, j\, ,\, id\,]\,]\, ,\, id\,]$$

biçiminde bir kayıtlı kuralı vardır.

Her ters bağıntı için, $[\, gen*inv\, ,\, [\ ]\, ,\, id\,]$ biçiminde bir kayıtlı kuralı vardır. Fonksiyon kayıtlı yeniden yazma sisteminin tamamlanmasını sağlar. Son sistemdeki kurallar, kısmen *ShorterLoggedRule f*onksiyonu tarafından sıralanır.

GAP

```
gap> lrws:=LoggedRewritingSystemFpGroup(q6);;
gap> Perform(lrws,Display);
[ q6_M4*q6_M2, [  ], <identity ...> ]
[ q6_M3*q6_M1, [  ], <identity ...> ]
[ q6_M2*q6_M4, [  ], <identity ...> ]
[ q6_M1*q6_M3, [  ], <identity ...> ]
[ q6_M1*q6_M2^2*q6_M3, [ [ 7, <identity ...> ], [ 7, q6_M4 ] ], q6_M2^2 ]
[ q6_M2*q6_M3*q6_M4, [ [ 7, q6_M1 ] ], q6_M3 ]
[ q6_M2^3, [ [ 6, <identity ...> ] ], q6_M4 ]
[ q6_M1*q6_M2*q6_M3, [ [ 7, <identity ...> ] ], q6_M2 ]
[ q6_M3^2, [ [ -5, <identity ...> ] ], q6_M1 ]
[ q6_M1^2, [ [ 5, <identity ...> ] ], q6_M3 ]
[ q6_M4^2, [ [ -6, <identity ...> ] ], q6_M2^2 ]
[ q6_M4*q6_M3, [ [ -7, q6_M1*q6_M2 ] ], q6_M3*q6_M4 ]
[ q6_M4*q6_M1, [ [ -6, <identity ...> ], [ -7, q6_M4^2 ], [ -7, q6_M2^-1
], [ -7, <identity ...> ], [ 6, q6_M1^-1 ] ], q6_M1*q6_M4 ]
[ q6_M3*q6_M2, [ [ -7, q6_M1 ] ], q6_M2*q6_M3 ]
[ q6_M2*q6_M1, [ [ -7, <identity ...> ] ], q6_M1*q6_M2 ]
gap> Length(lrws);
15
```

Yukarıdaki uygulamada son işlem olarak verilen *"Length(lrws)"* adımı bize *"LoggedRewritingSystemFpGroup(q6)"*işleminin sonucu olarak 15 tane bağıntı elde edildiğini göstermektedir.

7. SONUÇ VE ÖNERİLER

Bu tezde, *idrel* paketi incelenmiştir ve bu paket kapsamında birleştirilmiş grup teoride önemli bazı grup yapılarının sunuşları kullanılarak tam yeniden yazma sistemleri ve dolayısıyla elemanlarının normal form yapıları elde edilmiştir. 2015 yılında GAP kütüphanesine kabul edilen ve 2019 yılında son versiyonu oluşturulan bu paket, A. Heyworth ve C. D. Wensley tarafından hazırlanmıştır. Bu paket kapsamında yapılan uygulamalar tezin 5. ve 6. bölümlerinde bulunmaktadır.

5.2 alt bölümünde, *idrel* paketinin bazı fonksiyonlarının uygulamaları $\mathbb{Z}_3 \times \mathbb{Z}_4$ direkt çarpım grubunun sunuşu kullanılarak verilmiştir.

5.3 alt bölümünde, genişletilmiş Hecke grubunun $\overline{H}(\lambda_q)$ sunuşunda $q = 2$ ve $q = 4$ özel durumları alınarak elde edilen sunuşları incelenerek bu grubun monoid sunuşları için tam yeniden yazma sistemleri elde edilmiştir. Dolayısıyla bu yapıların elemanlarının normal formları oluşturulmuştur. Ayrıca incelenen genişletilmiş Hecke gruplarının ($\overline{H}(\lambda_2)$ ve $\overline{H}(\lambda_4)$) sunuşları kullanılarak *idrel* paketinde uygulamaları yapılmıştır.

5.4 alt bölümünde, genişletilmiş ve genelleştirilmiş Hecke grubunun $\overline{H}(p_1,...,p_n)$ sunuşunda $p_1 = p_2 = ... = p_n = 2$ alınması ile elde edilen $\overline{H}(2,2,...,2)$ monoid sunuşu incelenmiştir. Bu sunuşun tam yeniden yazma sistemi dolayısıyla da bu sistemdeki kurallara bağlı olarak elemanlarının normal form yapısı elde edilmiştir. Bu genişletilmiş ve genelleştirilmiş Hecke grubunun sunuşundaki üreteç sayısı i indisine bağlı olduğu için bu şekildeki genel bir sunuşun *idrel* paketinde uygulamaları yapılamamaktadır.

6. bölümde kayıtlı yeniden yazma sistemi ve onun fonksiyonları incelenmiştir. Bir grubun monoid sunuşunun tam yeniden yazma sistemini *idrel* paketinde oluştururken kaydettiği indirgeme aşamaları 5.2 alt bölümünde incelenen $\mathbb{Z}_3 \times \mathbb{Z}_4$ direkt çarpım grubu üzerine uygulanarak verilmiştir.

KAYNAKLAR

Aslan, A. F. (2010). GAP ile Grup Cebirlerinin Lie Cebirleri. *Yüksek Lisans Tezi, Eskişehir Osman Gazi Üniversitesi,* Eskişehir.

Book, R. V. (1987). Thue Systems as Rewriting Systems. *J. Symbolic Computation*, 3(1-2), 39-68.

Book, R. V. ve Otto, F. (1993). String Rewriting Systems. *Springer-Verlag, New York*.

Cohen, D. E. (1989). *Combinatorial Group Theory: Topological Approach*. Cambridge University Press.

Çetinalp, E. K. (2016). Bazı Grup ve Monoid Yapıları İçin Karar Verme Problemleri ve Büyüme Serileri. *Yüksek Lisans Tezi, Karamanoğlu Mehmetbey Üniversitesi Fen Bilimleri Enstitüsü,* Karaman.

Çetinalp, E. K. (2020). Grupların Çapraz Çarpımı ve Cebirsel Özellikleri. *Doktora Tezi, Karamanoğlu Mehmetbey Üniversitesi Fen Bilimleri Enstitüsü,* Karaman.

Çevik, A.S., Özgür, N.Y., Şahin, R. (2008). The extended Hecke groups as semi-direct products and related results, Int. J. Appl. Math. Stat. 13 (S08), 63-72.

Doğrayıcı, G. ve Şahin, R. (2020). Commutator subgroups of generalized Hecke and extended generalized Hecke groups, II. *Turkish J. Mathematics,* 44, 2123-2131.

Hecke, E. (1936). Uber die Bestimmung Dirichletscher Reihen durch ihre Funktionalgleichung, *Math. Ann*. 112 (1), 664–699.

Huang ,S. (1999). Generalized Hecke groups and Heckepolygons, *Ann. Acad. Sci. Fenn. Math*. 24(1), 187–214.

Karpuz, E. G. (2006). Grup ve Monoid Cebirsel Yapısında Karar Verme Problemleri. *Yüksek Lisans Tezi, Balıkesir Üniversitesi Fen Bilimleri Enstitüsü*, Balıkesir.

Karpuz, E. G. ve Çevik, A. S. (2012). Gröbner-Shirshov bases for extended modular, extended Hecke and Picard groups, *Mathematical Notes*, 92 (5), 636-642.

Koruoğlu, R., Şahin, R. ve İkikardeş, S. (2007). The normal subgroup structure of the extended Hecke groups, *Bull. Braz. Math. Soc*.(N. S.) 38(1), 51-65.

Lyndon, C. R. ve Schupp, E. P. (1977). Combinatorial Group Theory. *Springer-Verlag, Berlin Heildberg, New York*.

Magnus, N., Karrass, A. ve Solitar. D. (1976). Combinatorial Group Theory. *Dover Publications, Inc. New York.*

Newman, M. (1962). The structure of some subgroups of the modular group,

Illionis J. Math., 8, 480-487.

Odabaş, A. (2004). GAP (Grup, Algoritma, Programlama) ile Cebirler ve Sayılar Teorisi. Y*üksek Lisans Tezi, Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü*, Kütahya.

Rotman, J. J. (1984). An Introduction to the Theory of Groups. *Wm. C. Brown Publishers, Dubuque, United States of America.*

Sims, C. C. (1994). Computation with Finitely Presented Group. *Cambridge UniversityPress.*

Şahin, R., İkikardeş, S. ve Koruoğlu Ö. (2004). On the power subgroups of the extended modular group $\bar{\Gamma}$, *Turkish J. Math.* 28(2), 143-151.

Şimşek, M. (2018). GAP (Grup, Algoritma ve Programlama) ve Yeniden Yazma Sistemi. *Yüksek Lisans Tezi, Karamanoğlu Mehmetbey Üniversitesi Fen Bilimleri Enstitüsü,* Karaman.

www.gap-system.org/index.html

https://www.gap-system.org/Packages/idrel.html