RESOLUTION İLE RASYONEL FONKSİYONLARIN SEMBOLİK İNTEGRASYONU

SYMBOLIC INTEGRATION OF RATIONAL FUNCTIONS BY RESOLUTION

İ. ONUR KIYMAZ

Hacettepe Üniversitesi
Fen Bilimleri Enstitüsü Yönetmeliğinin
Matematik Anabilim Dalı İçin Öngördüğü
BİLİM UZMANLIĞI TEZİ
olarak hazırlanmıştır.

1997

Fen Bilimleri Enstitüsü Müdürlüğüne

Bu çalışma jürimiz tarafından MATEMATİK ANABİLİM DALI'nda BİLİM UZMANLIĞI TEZİ olarak kabul edilmiştir.

Başkan : ...Doç..Dr..L.M.BROWN.....

Üye : ...Prof.Dr.Seref MİRASYEDİOĞLU.....

Üye : ...Prof..Dr..Aydın..TİRYAKİ.....

ONAY

Bu tez ... / ... / 1997 tarihinde Enstitü Yönetim Kurulunca belirtilen yukarıdaki jüri üyeleri tarafından kabul edilmiştir.

12/09/1997

Prof. Dr. Gültekin GÜNAY
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

ÖZET

Q ve P polinom fonksiyonları olmak üzere, Q / P biçimindeki rasyonel fonksiyonların integrallerinin sembolik çözümlerine, birinci dereceden mantık yapısı içinde ulaşmak için uygun yöntem araştırılmıştır. Ele alınan yöntem, birinci dereceden mantık yapısı içerisinde tanımlanarak clause kümeleri oluşturulmuş, bu kümelerden resolution ilkesi ile sonuç üretilmiştir. Bulunan sonuçların doğruluğu geliştirilen önteorem ve teoremlerle verilmiştir.

ABSTRACT

The aim of this work is to investigate a deductive approach for symbolic integration of rational functions with the form Q / P, where P and Q are polynomial functions in the context of first order logic. The correctness of the deductive approach which obtains symbolic solutions by applying the resolution principle to a clause generated by using Herbrand Theorem and the rule of integration of rational functions has been established in a Lemma and theorem.

TEŞEKKÜR

Tez çalışmam süresince büyük yardımlarını gördüğüm, bilgi ve deneyiminden yararlandığım, tez yöneticim ve danışmanım sayın Prof. Dr. Şeref Mirasyedioğlu'na, çalışmamıza gösterdiği yakın ilgi ve önerilerinden ötürü sayın Doç. Dr. L. M. Brown'a, cebirsel kısımdaki ön çalışmalarımıza sağladıkları katkılardan ötürü sayın Doç. Dr. Ziya Argün'e ve sayın Yrd. Doç. Dr. Ahmet Arıkan'a ayrıca her konudaki yardımları için sayın Araş. Gör. Tolga Güyer'e içtenlikle teşekkür ederim.

İÇİNDEKİLER DİZİNİ

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Mantıksal Programlama, Yapay Zeka ve Otomatik Teorem İspatlama çalışmalarının doğal bir sonucu olarak 1970'li yılların başında yaygın uygulama alanları bulur. Bu konudaki gelişmelere, Herbrand'ın 1930 yılında tanıttığı temel teoremler, Davis, Putnam, Gilmore ve Prawitz'in 1960'lı yıllarda geliştirdikleri Resolution prensipleri ve 1965 yılında Robinson'un tanıttığı Unification Algoritmasının önemli katkıları olmuştur (Davis ve Putnam, 1960).

Mantığın bir programlama dili gibi kullanılması fikri ise 1972 yılında Kowalski tarafından ortaya atılmıştır. PROLOG (PROgraming in LOGic), bu fikir üzerine geliştirilen ve sembolik işlem yapabilen ilk bilgisayar dillerinden biridir.

Mantıksal Programlama çalışmaları, bilgisayarların hızla gelişmesi ile uygulama alanı bulmuştur. Günümüzde bilgisayarlar gelişmiş hesaplayıcılar olmaktan çıkıp, çok boyutlu matris işlemlerini, sembolik hesaplamaları vb. yapabilen makineler haline gelmiştir.

Tez çalışmamızda, Q ve P polinom fonksiyonları olmak üzere, Q/P şeklindeki rasyonel fonksiyonların sembolik yöntemlerle integralinin hesaplanmasına çalışılmıştır. Sonuca ulaşmak için :

i) Cebirsel olarak bir rasyonel fonksiyonun integralinin alınması problemi ve problemin, mantık ortamına taşımaya en elverişli çözüm yolu araştırıldı.

ii) Oluşturulan çözüm yolu birinci dereceden mantık yapısın içinde tanımlandı.

iii) Çözüm yoluna karşılık gelen temel clause'lar oluşturuldu.

iv) Oluşturulan temel clause kümesinden resolution ile çözüm üretildi.

v) Üretilen çözümlerin doğruluğu, kurulan önteorem ve teoremlerle ispatlandı.

Ele alınan konular tezimizde beş bölüm altında incelenmiştir.

Birinci bölüm giriştir.

İkinci bölümde, rasyonel fonksiyonların integralinin çözümünde kullanılacak cebirsel yönteme temel teşkil eden alt yöntemler anlatılmıştır.

Üçüncü bölümde, rasyonel fonksiyonların sembolik integralini bulmada kullanılacak yöntem olan Horowitz-Ostrogradski yöntemi tanıtılmıştır.

Dördüncü bölüm, birinci dereceden mantık yapısının tanıtımına ayrılmıştır. Bu bölümde Herbrand Teoreminden başlanarak Skolem Standart Formu anlatılmış ve Resolution ilkesi verilmiştir.

Beşinci ve son bölümde ise çözüm yolu birinci dereceden mantık yapısına taşınmış, geliştirilen sembolik yöntem anlatılmıştır.

Eklere ayrılan birinci bölümde, ikinci bölümde anlatılan alt yöntemlerin algoritmaları verilmiştir. İkinci ek olarak geliştirilen sembolik yöntemin, REDUCE programlama dilinde hazırlanan program kodu açıklamalı olarak anlatılmıştır.

2. CEBİRSEL YAKLAŞIM

Bu bölümde, rasyonel fonksiyonların integrallenmesi konusunda kullanacağımız bazı temel kavramlar cebirsel anlamda incelenecektir. Oluşturulan yöntemlerin algoritmik formları EK-1 de verilmiştir.

2.1. Tek Değişkenli İki Polinomun En Büyük Ortak Böleninin Bulunması

A ve B, $Z[x]$ halkasına ait, en büyük ortak bölenini (ebob) hesaplamak istediğimiz, tek değişkenli iki polinom olsun. Bu iki polinomun derecesi sıfırdan farklı ortak çarpanı olduğunu varsayalım, bu durumda $A=PQ$ eşitliğini sağlayacak bir Q polinomu vardır.

2.1.1. Tanım : p bir asal sayı, $A = \sum_{i=0}^{n} a_i x^i$ biçiminde bir polinom ve $\overline{a}_i = a_i \bmod p$ (i = 1,2,...,n) olmak üzere, $A_p = \sum_{i=0}^{n} \overline{a}_i x^i$ polinomu şeklinde tanımlanır.

Bu durumda $A_p = P_p Q_p$ eşitliği geçerlidir. Diğer bir deyişle P, A ve B 'yi bölerse P_p de A_p ve B_p 'yi böler. Böylece P_p polinomların ortak çarpanı olur, ancak bu durum polinomların ebob'ni olduğunu garantilemez

2.1.1. Örnek : $A = x-3$, $B = x+2$ için $P = 1$ olur, fakat $A_5 = B_5 = x+2$ olduğundan $ebob(A_5, B_5) = x+2$ olur. Oysa $P_5 = 1$ dir.

Bu sorunu ortadan kaldırmak için öncelikle polinomların ebob'ninin katsayılarını sınırlamaya çalışmalıyız.

2.1.1. Teorem (Landau-Mignotte Eşitsizliği) : a_i ve b_i tamsayı olmak üzere, $Q = \sum_{i=0}^{q} b_i x^i$, $P = \sum_{i=0}^{p} a_i x^i$ polinomunun bir böleni olsun. Bu durumda Landau (1905) ve Mignotte (1974; 1982) tarafından verilen aşağıdaki eşitsizlik geçerlidir (Davenport, Siret, Tournier, 1993).

$$\sum_{i=0}^{q} |b_i| \le 2^q \left|\frac{a_p}{b_q}\right| \sqrt{\sum_{i=0}^{p} a_i^2}$$

2.1.1. Sonuç : $A = \sum_{i=0}^{\alpha} a_i x^i$ ve $B = \sum_{i=0}^{\beta} b_i x^i$ polinomlarının ebob' ninin tüm katsayıları

$$2^{\min(\alpha,\beta)} ebob(a_\alpha, b_\beta) \min\left(\frac{1}{|a_\alpha|}\sqrt{\sum_{i=0}^{\alpha} a_i^2}, \frac{1}{|b_\beta|}\sqrt{\sum_{i=0}^{\beta} b_i^2} \right)$$

ile sınırlıdır (Davenport, Siret, Tournier, 1993).

2.1.2. Sonuç : $A = \sum_{i=0}^{\alpha} a_i x^i$ ve $B = \sum_{i=0}^{\beta} b_i x^i$ polinomlarının ebob' ninin tüm katsayıları

$$2^{\min(\alpha,\beta)} ebob(a_0, b_0) \min\left(\frac{1}{|a_0|}\sqrt{\sum_{i=0}^{\alpha} a_i^2}, \frac{1}{|b_0|}\sqrt{\sum_{i=0}^{\beta} b_i^2} \right)$$

ile sınırlıdır (Davenport, Siret, Tournier, 1993).

İki polinomun ebob'ninin katsayıları, polinomların kendi katsayılarından daha büyük olabilir.

2.1.2. Örnek :

$A = x^5+3x^4+2x^3-2x^2-3x-1 = (x+1)^4(x-1)$

$B = x^6+3x^5+3x^4+2x^3+3x^2+3x+1 = (x+1)^4(x^2-x+1)$

$ebob(A,B) = x^4+4x^3+6x^2+4x+1 = (x+1)^4$

2.1.2. Tanım : P, $P(x)=a_nx^n+a_{n-1}x^{n-1}+ \ldots +a_1x+a_0$ biçiminde tanımlı bir fonksiyon olsun. Bu durumda a_n katsayısına P polinomunun baş katsayısı adı verilir.

2.1.1. Önteorem : Eğer p, $ebob(A,B)$'nin baş katsayısını bölemez ise $ebob(A_p, B_p)$'nin derecesi, $ebob(A , B)$'nin derecesinden eşit ya da büyüktür (Davenport, Siret, Tournier, 1993).

Yukarıda verilen önteoremi kullanabilmek için ebob'ni (en azından ebob'nin baş katsayısını) bilmemiz gerekmektedir. Bunun yerine kullanımı daha kolay olan aşağıdaki sonucu verebiliriz.

2.1.3. Sonuç : Eğer p, A ve B'nin baş katsayılarını bölemezse (birini bölebilir ancak her ikisini de bölemez) $ebob(A_p, B_p)$'nin derecesi, $ebob(A,B)$'nin derecesinden büyük ya da eşittir (Davenport, Siret, Tournier, 1993).

Yukarıda verilen lemma ve sonuçlara dayanarak, hesapladığımız ebob'nin doğruluğunu kontrol edebiliriz. Eğer hesapladığımız sonucun derecesi $ebob(A_p, B_p)$'nin derecesine eşitse (burada p, Sonuç 2.1.3'ü sağlıyorsa) ve sonuç A ve B'yi bölüyorsa ebob'dir.

2.1.2. Önteorem : $C = ebob(A,B)$ olsun. Eğer p, Sonuç 2.1.3'ü sağlıyor ve [1]$Res(A/C,B/C)$'yi bölemiyorsa $C_p = ebob(A_p, B_p)$ dir (Davenport, Siret, Tournier, 1993).

2.1.3. Tanım : Eğer $ebob(A_p, B_p) = ebob(A, B)_p$ ise bu problemin indirgemesine *iyi indirgeme*, aksi halde *zayıf indirgeme* denir.

Özel olarak, eğer A ve B aralarında asalsa, A_p ve B_p aralarında asal olacak şekilde her zaman bir p bulunabilir.

2.2. Square-Free Ayrıştırması

2.2.1. Tanım : R, sıfır karakteristikli bir integral halkası olmak üzere, $P \in R(x)$ polinomunu alalım. P polinomu $P = \prod_{i=1}^{n} (x - a_i)^{n_i}$ şeklinde yazılabilir. $n_i > 1$ olacak şekilde $(x - a_i)^{n_i}$ çarpanlarına P polinomunun *katlı çarpanları* denir.

$P \in R(x)$ olsun. Bu durumda P'nin katlı çarpanları olabilir. (Yani Q bir polinom olmak üzere Q^2 , P yi bölebilir.)

---

[1] $Res(f,g)$ ile f ve g fonksiyonlarının resultantı kastedilmiştir. Resultant kavramı 2.3. Bölümde ayrıntılı olarak açıklanacaktır

P 'nin

$$P = \prod_{i=1}^{n} (x - a_i)^{n_i}$$

biçiminde lineer çarpanların bir çarpımı olarak yazılabildiğini kabul edelim. Bu durumda P'nin türevi

$$P' = \sum_{i=1}^{n} \left(n_i (x - a_i)^{n_i - 1} \prod_{\substack{j=1 \\ i \neq j}}^{n} (x - a_i)^{n_j} \right)$$

olur.

2.2.1. Önteorem : $\forall\ i$ için $ebob(P,P') = \prod_{i=1}^{n} (x - a_i)^{n_i - 1}$ olarak yazılabilir.

İspat : Açıktır ki $\forall\ i$ için $(x - a_i)^{n_i - 1}$ hem P hem de P' 'nü böler. Ayrıca P'yi bölen her polinom $m \leq n_i$ olmak üzere $(x\text{-}a_i)^m$ çarpanlarından oluşmalıdır. Burada $m = n_i$ olamaz çünkü $(x - a_i)^{n_i}$, P' 'nün biri hariç bütün terimlerini bölebilir, yani P' 'nü bölemez. O halde

$$ebob(P,P') = \prod_{i=1}^{n} (x - a_i)^{n_i - 1}$$

şeklinde olmalıdır. □

2.2.1. Sonuç : $\forall\ i$ için $\dfrac{P}{ebob(P,P')} = \prod_{i=1}^{n} (x - a_i)$ olur (Davenport, Siret, Tournier, 1993).

Açıkça $P / ebob(P, P')$ ye Q dersek

$$ebob(Q, ebob(P,P')) = \prod_{\substack{i=1 \\ n_i > 1}}^{n} (x - a_i)$$

elde ederiz ve buradan

$$\frac{Q}{ebob(Q,\gcd(P,P'))} = \prod_{\substack{i=1 \\ n_i=1}}^{n}(x-a_i)$$

eşitliğini buluruz.

Sonuç 2.2.1 eşitliğinin sol tarafı P'nin tüm tek katlı çarpanlarını içerir. Eğer P'ye uyguladığımız tüm işlemleri $ebob(P,P')$ ne uygularsak, $ebob(P,P')$'nün tüm tek katlı çarpanlarını elde ederiz. Bunlar aynı zamanda P'nin çift katlı çarpanları olur. Aynı şekilde devam ederek P'nin tüm üç katlı, dört katlı vb.. çarpanları da elde edilebilir.

Genel olarak $R(x)$'de P_i ler P'nin i katlı çarpanları olmak üzere, basit işlemlerle, P'yi ΠP_i^i formuna ayrıştırabiliriz. P'nin bu ayrışımında her P_i katlı çarpan değildir ve P_i ler aralarında asaldır.

2.2.2. Tanım : $\forall$ i için, P_i ler aralarında asal ve katlı çarpan olmamak üzere $P=\Pi P_i^i$ ayrıştırmasına *square-free* ayrıştırması denir.

2.3. Resultant

Aralarında asal iki polinomun, ortak çarpanlarının olup olmadığına karar verebilmek için bilgisayar cebrinde *Resultant* kavramı kullanılır.

$f(x)$ ve $g(x)$, katsayıları R halkasında olan iki polinom olsun. $f = \sum_{i=0}^{n} a_i x^i$ ve $g = \sum_{i=0}^{m} b_i x^i$ şeklinde yazabiliriz.

2.3.1. Tanım : f *ve* g'nin *Sylvester Matrisi* aşağıdaki şekilde tanımlıdır. Bu matrisde m satır a_i'lerden, n satır b_i'lerden oluşur.

$$\begin{bmatrix} a_n & a_{n-1} & \dots & a_1 & a_0 & 0 & 0 & \dots & 0 \\ 0 & a_n & a_{n-1} & \dots & a_1 & a_0 & 0 & \dots & 0 \\ & & & & \dots & & & & \\ 0 & \dots & 0 & a_n & a_{n-1} & \dots & a_1 & a_0 & 0 \\ 0 & \dots & 0 & 0 & a_n & a_{n-1} & \dots & a_1 & a_0 \\ b_m & b_{m-1} & \dots & b_1 & b_0 & 0 & 0 & \dots & 0 \\ 0 & b_m & b_{m-1} & \dots & b_1 & b_0 & 0 & \dots & 0 \\ & & & & \dots & & & & \\ 0 & \dots & 0 & b_m & b_{m-1} & \dots & b_1 & b_0 & 0 \\ 0 & \dots & 0 & 0 & b_m & b_{m-1} & \dots & b_1 & b_0 \end{bmatrix}_{(m+n)x(m+n)}$$

2.3.2. Tanım : f ve g'nin *resultantı*, *Res(f,g)* şeklinde yazılır, fonksiyonlara karşılık gelen Sylvester Matrisinin (bir değişken içeren) determinantıdır.

2.3.1. Örnek : $f = (x+1)(x-2) = x^2 - x - 2$ ve $g = (x-1)^2(x+2) = x^3 - 3x + 2$ olsun. O halde bu iki fonksiyonlarının resultantı, Sylvester Matrisinin determinantıdır.

$$\begin{vmatrix} 1 & 0 & -3 & 2 & 0 \\ 0 & 1 & 0 & -3 & 2 \\ 1 & -1 & -2 & 0 & 0 \\ 0 & 1 & -1 & -2 & 0 \\ 0 & 0 & 1 & -1 & -2 \end{vmatrix} = 16 = Res(g,f)$$

2.3.1. Önerme : *Res(f,g)* ile *Res(g,f)* işaret farkıyla birbirlerine eşittir (Davenport, Siret, Tournier, 1993).

2.3.2. Önerme : f ve g'nin ortak çarpanlarının olabilmesi için gerek ve yeter koşul $Res(f,g) = 0$ olmasıdır (Davenport, Siret, Tournier, 1993).

2.3.2. Örnek : $f = (x+1)(x-2) = x^2 - x - 2$ ve $g = (x+1)^2(x+2) = x^3 + 4x^2 + 5x + 2$ olsun. f ve g ortak çarpan içerdiğinden Önerme 2.3.2. gereği $Res(g,f)=0$ olmalıdır. Yani fonksiyonlara karşılık gelen Sylvester Matrisinin determinantı 0 olmalıdır.

$$Res(g,f)=\begin{vmatrix}1 & 4 & 5 & 2 & 0\\ 0 & 1 & 4 & 5 & 2\\ 1 & -1 & -2 & 0 & 0\\ 0 & 1 & -1 & -2 & 0\\ 0 & 0 & 1 & -1 & -2\end{vmatrix}=0$$

2.3.3. Önerme : Eğer α_i'ler f'in kökleri ise

$$\mathrm{Re}s(f,g)=a_n^m\prod_{i=1}^{n}g(\alpha_i)$$

dir (Davenport, Siret, Tournier, 1993).

2.3.4. Önerme : Eğer β_i'ler g 'nin kökleri ise

$$\mathrm{Re}s(f,g)=(-1)^{mn}b_m^n\prod_{i=1}^{m}f(\beta_i)$$

dir. (Davenport, Siret, Tournier, 1993)

2.3.5. Önerme : $\mathrm{Re}s(f,g)=a_n^m b_m^n\prod_{i=1}^{n}\prod_{j=1}^{m}(\alpha_i-\beta_j)$ (Davenport, Siret, Tournier, 1993).

2.3.3. Tanım : f 'nin diskriminantı, $Disc(f)=a_n^{2n-2}\prod_{i=1}^{n}\prod_{\substack{j=1\\ i\neq j}}^{n}(\alpha_i-\alpha_j)$.

2.3.6. Önerme : $Disc(f)\in R$ olmak üzere, $Res(f, f')=a_nDisc(f)$ dir (Davenport, Siret, Tournier, 1993).

2.4. Çinlilerin Kalan Teoremi

Bu kesimde Çinlilerin Kalan Teoremini algoritmik olarak inceleyerek ilerideki konularda kullanacağımız algoritmalar için bir zemin oluşturmaya çalışacağız. Bu teoremin kullanacağımız iki ayrı durumunu (tamsayı ve polinom durumu) ayrı ayrı ele alacağız.

2.4.1. Teorem (Çinlilerin Kalan Teoremi - tamsayı durumu -) : M ve N aralarında asal iki tamsayı olsun. Bu durumda her (a,b) tamsayı çifti için $x \equiv c \pmod{MN}$ olacak şekilde bir c tamsayısının bulunması için gerek ve yeter koşul $x \equiv a \pmod{M}$ ve $x \equiv b \pmod{N}$ olmasıdır (Davenport, Siret, Tournier, 1993).

2.4.1. Sonuç : N_1 , ... , N_n ikişer ikişer aralarında asal ve a_1 , ... , a_n herhangi tamsayılar ise $x \equiv c \pmod{\prod_{i=1}^{n} N_i}$ olacak şekilde bir c tamsayısının bulunması için gerek ve yeter koşul her i için $x \equiv a_i \pmod{N_i}$ olmasıdır (Davenport, Siret, Tournier, 1993).

2.4.2. Teorem (Çinlilerin Kalan Teoremi - genelleştirilmiş durum -) : M ve N iki tamsayı olsun. Bu durumda her (a,b) tamsayı çifti için $a \neq b \pmod{ebob(M,N)}$ olmak üzere, $x \equiv a \pmod{M}$ ve $x \equiv b \pmod{N}$ özelliğini sağlayan bir c tamsayısının bulunabilmesi için gerek ve yeter koşul $x \equiv c \pmod{okek(M,N)}$ olmasıdır (Davenport, Siret, Tournier, 1993).

2.4.3. Teorem (Çinlilerin Kalan Teoremi - polinom durumu -) : M ve N aralarında asal iki monik polinom olsun. Bu durumda her (a,b) polinom çifti için $x \equiv c \pmod{MN}$ olacak şekilde bir c polinomunun bulunması için gerek ve yeter koşul $x \equiv a \pmod{M}$ ve $x \equiv b \pmod{N}$ olmasıdır (Davenport, Siret, Tournier, 1993).

2.4.2. Sonuç : N_1 , ... , N_n ikişer ikişer aralarında asal polinomlar ve a_1 , ... , a_n herhangi polinomlar ise $x \equiv c \pmod{\prod_{i=1}^{n} N_i}$ olacak şekilde bir c polinomunun bulunması için gerek ve yeter koşul her i için $x \equiv a_i \pmod{N_i}$ olmasıdır (Davenport, Siret, Tournier, 1993).

2.4.4. Teorem (Çinlilerin Kalan Teoremi - genelleştirilmiş durum -) : M ve N iki polinom olsun. Bu durumda her (a,b) polinom çifti için $a \neq b \pmod{ebob(M,N)}$ olmak üzere, $x \equiv a \pmod{M}$ ve $x \equiv b \pmod{N}$ özelliğini sağlayan bir c polinomunun bulunabilmesi için gerek ve yeter koşul $x \equiv c \pmod{okek(M,N)}$ olmasıdır (Davenport, Siret, Tournier, 1993).

2.5. Genişletilmiş Euclidean Algoritması

Euclid Algoritmasına göre a ve b şeklindeki iki tamsayının ebob'ni aşağıdaki biçimde hesaplanır.

1. $abs(a) < abs(b)$ ise

    2. $t := a$

    3. $a := b$

    4. $b := t$

5. $r <> 0$ olduğu sürece

    6. $t := kalan(a,b)$

    7. $a := b$

    8. $b := t$

9. $ebob := a$

Burada *kalan* fonksiyonu ile a / b bölümünden kalan sonuç hesaplanmaktadır. *abs* fonksiyonu ise bilinen "mutlak değer" fonksiyonudur. Bu algoritma sadece basit bir ebob hesabı yapmakla kalmaz. a ve b değerleri, her adımda, başlangıç değerlerinin bir lineer kombinasyonu olurlar. Bu özellikten de yararlanarak Genişletilmiş Euclidean Algoritmasını aşağıdaki şekilde yeniden oluşturabiliriz.

Bu algoritmada [..] ile sayı çiftleri gösterilmekte, *der* fonksiyonu ile polinomun derecesi kastedilmektedir. A ve B ise a ve b değerlerine karşılık gelen başlangıç değerleridir. Algoritma sonucu ilk dönen değer (burada a) a ve b'nin ebob'ni olup ikinci bulunan değer (burada A) $[u,v]$ sayı çiftini tutar. Eğer a ve b'nin ebob'nine p dersek, dönen değerler $p=ua+vb$ eşitliğini sağlarlar.

1. $der(a) < der(b)$ ise

    2. $t := a$

3. $a := b$

4. $b := t$

5. A : = [0,1]

6. B : = [1,0]

7. aksi halde

8. A : = [1,0]

9. B : = [0,1]

10. $r <> 0$ olduğu sürece

11. $t :=$ *kalan*(a,b)

12. T : = A-[a / b]B

13. $a := b$

14. $b := t$

15. A : = B

16. B : = T

17. Sonuç olarak a ve A değerlerini al, akışı durdur.

3. RASYONEL FONKSİYONLARIN İNTEGRALLENMESİ

Her f rasyonel fonksiyonu, P, Q ve R, $R=\prod_{i=0}^{n}(x-a_i)^{n_i}$ olacak biçimde birer polinom fonksiyonları, Q ve R aralarında asal ve Q'nun derecesi R'nin derecesinden küçük olmak üzere, $P+Q/R$ formunda yazılabilir. Biliyoruz ki

$$\int(f+g)=\int f+\int g \qquad (1)$$

eşitliği geçerlidir. Ancak cebirsel integrasyon kuramında bu her zaman doğru değildir. $\int(f+g)$ nin integrali varken $\int f$ ve $\int g$ nin integralleri olmayabilir.

3.1. Örnek : $\int x^x+\log(x)x^x=x^x$ olmasına karşılık $\int x^x$ ve $\int \log(x)x^x$ tek başlarına (sonlu formda) integralleri yoktur.

Aslında (1) eşitliği üç integralden ikisinin integrali varsa kullanılmalıdır. Oysa bir P polinomu her zaman sonlu bir integrale sahiptir, bu sebepten $f=P$ ve $g=Q/R$ için (1) eşitliği her zaman sağlanır.

Bu durumda f'nin integrallenmesi problemini çok daha kolay olan, P polinomunun ve Q/R rasyonel fonksiyonunun integrallenmesi problemine indirgemiş olduk.

3.1. Horowitz - Ostrogradski Yöntemi

$\int Q/R$ integralinin hesaplanabilmesi için Horowitz (1969,1971), Rus matematikçi Ostrogradski tarafından da bilinen aşağıdaki yöntemi vermiştir.

Bu yöntemin amacı $\int Q/R$ integralini $Q_1/R_1+\int Q_2/R_2$ formuna indirgemektir. Burada kalan integral çözüldüğünde sadece logaritmik toplamlar içereçektir. Ayrıca biliyoruz ki R_1 bir üs indirgenmiş olarak R ile aynı çarpanları içerirken, R_2 hiç bir katlı çarpan içermez. Bu sebepten R_1R_2 , R'nin tüm çarpanlarını verir. Square-free ayrıştırması gereği $R_1=ebob(R,R')$, ve $R_2=R/ebob(R,R')$ olarak tanımlanabilir. O halde

$$\int Q/R = Q_1/R_1 + \int Q_2/R_2$$

için türev alarak

$$Q/R = (Q_1/R_1)' + Q_2/R_2$$

$$= Q_1'/R_1 - Q_1.R_1'/R_1^2 + Q_2/R_2$$

$$= (Q_1'.R_2 - Q_1.S + Q_2.R_1)/R$$

elde edilir ki burada $S = R_1'.R_2/R_1$ dir (buradaki bölme işlemi kalansızdır). Böylece problemimiz

$$Q = Q_1'.R_2 - Q_1.S + Q_2.R_1 \qquad (2)$$

polinomunun çözümü problemine indirgenmiş oldu. (2) eşitliğinde Q, S, R_1, ve R_2 biliniyor, Q_1 ve Q_2 bilinmiyor ancak tespit edilebilir ; şöyle ki m ve n sırasıyla R_1 ve R_2'nin dereceleri olmak üzere Q_1 ve Q_2'nin dereceleri m ve n' den küçüktür. Böylece $Q_1 = \sum_{i=0}^{m-1} a_i x^i$ ve $Q_2 = \sum_{i=0}^{n-1} b_i x^i$ şeklinde yazılabilir. (2) denklemi $m+n$ bilinmeyenli $m+n$ lineer denklem içeren bir sistem olarak yeniden yazılabilir. Son olarak bu sistem çözülerek $\int Q/R$ integrasyon problemi $\int Q_2/R_2$ integrasyonu problemine indirgenebilir.

3.2. Logaritmik Kısmın Çözümü

Yukarıdaki yöntem ile $\int Q/R$ integralini $Q_1/R_1 + \int Q_2/R_2$ formuna getirmiş ve kalan integral çözüldüğünde sadece logaritmik toplamlar içereceğinden bahsetmiştik. Şimdi kalan integralin çözümüne nasıl ulaşacağımıza bakalım.

Kabul edelim ki

$$\int Q/R = \sum_{i=1}^{n} c_i \log v_i \qquad (3)$$

integralin bir çözümü olsun. Burada c_i'ler sabit, v_i'ler ise rasyonel fonksiyonlardır. Ancak $\log(a / b) = \log a - \log b$ olduğundan genelliği bozmadan v_i'lerin polinom fonksiyonları olduklarını söyleyebiliriz. Bunun dışında v_i'lerin aralarında asal olduğunu ve tüm c_i sabitlerinin birbirlerinden farklı olduklarını varsayabiliriz.

(3) denkleminin türevini alarak

$$Q/R = \sum_{i=1}^{n} \frac{c_i v_i'}{v_i} \qquad (3^*)$$

elde edilir. Burada v_i'leri square-free kabulümüz, bu toplamın sadeleştirilemeyeceğini gerektirirken, v_i'lerin aralarında asal olması kabulümüzde bu toplamda hiç bir sadeleştirmenin yapılamayacağını gerektirir. Bu gerektirmeler sonucunda v_i'ler R'nin tüm çarpanları olmalıdır. Yani $R = \prod_{i=1}^{n} v_i$ yazılabilir. Buradan v_i çarpanlarını türevlersek $R' = \sum v_i' u_i$ olur ki burada $u_i = \prod_{j \neq i} v_j$ dir. (3^*) denkleminde payda düzenlenecek olursa $Q = \sum c_i v_i' u_i$ elde edilir. Q ve R' için elde edilen bu iki eşitlik aşağıdaki dedüksiyonları sağlar :

$v_k = ebob(0, v_k)$

$= ebob(Q - \sum c_i v_i' u_i , v_k)$

$= ebob(Q - c_k v_k' u_k , v_k)$, diğer tüm u_i'ler v_k'lara bölünebildiğinden

$= ebob(Q - c_k \sum v_i' u_i , v_k)$ olur. Benzer olarak

$= ebob(Q - c_k R' , v_k)$ elde edilir.

Bu hesaplamaların yardımıyla aşağıdaki hesaplamaları yapabiliriz :

$ebob(Q - c_k R' , R) = ebob(Q - c_k R' , \prod_{i=1}^{n} v_i)$

$= \prod_{i=1}^{n} ebob(Q - c_k R' , v_i)$, v_i'ler aralarında asal

olduğundan

$= ebob(Q - c_k R', v_k)$, diğer tüm terimler yok olduğundan

$= v_k$

olur.

Bu durumda eğer c_k'lar biliniyorsa v_k'lar hesaplanabilir. Burada c_k'lar $ebob(Q-yR',R)\neq 1$ değerlerine karşılık gelen y'lerdirler. Bu y değerlerini resultant yardımıyla hesaplayabiliriz. $Res_x(Q-yR', R)$ polinomunun kökleri c_k sabitlerini verir. c_k sabitleri ile tespit edilen v_k'lar (3) denkleminde yerine yazılarak

$$\int Q/R = Q_1/R_1 + \int Q_2/R_2 = Q_1/R_1 + \sum_{i=1}^{n} c_i \log v_i \qquad (4)$$

rasyonel fonksiyonun integrale ulaşılmış olur.

3.2.1. Örnek : $f = (x+3)/(x^2+2x+1)$ olmak üzere $\int f$ integralini yukarıda verilen Horowitz - Ostrogradski yöntemini kullanarak hesaplamak için öncelikle $R' = 2x+2$ türevi hesaplanır. Buradan $R_1 = ebob(R,R') = x+1$, $R'_1 = 1$ ve $R_2 = R / R_1 = x+1$ sonuçları elde edilir. R_1'in derecesi $m=1$, R_2'nin derecesi ise $n=1$ dir. Q_1 ve Q_2 yi tespit edebiliriz.

$Q_1 = \sum_{i=0}^{0} a_i x^i = a_0$ ve $Q_2 = \sum_{i=0}^{0} b_i x^i = b_0$ Ayrıca $Q'_1 = 0$ ve $S = 1$ olduğu da açıktır. Bulduğumuz bu değerleri yerine yazarak :

$$Q = Q'_1 R_2 - Q_1 S + Q_2 R_1$$

$$= 0.R_2 - a_0.1 + b_0(x+1)$$

$$= b_0 x + (b_0 - a_0) = x + 3$$

elde edilir. Buradan $b_0 = 1$, $a_0 = -2$ dolayısıyla $Q_1 = -2$ ve $Q_2 = 1$ sonuçları elde edilir. O halde f rasyonel fonksiyonunun integrali

$$\int f = Q_1/R_1 + \int Q_2/R_2 = -2/(x+1) + \int \frac{1}{x+1} dx$$

olacaktır. Ancak kalan integralin hesabı için "logaritmik kısmın çözümü" adlı kesiminde anlatılan yöntem uygulanmalıdır. Yöntem gereği $R_2' = 1$ hesap edilmeli, daha sonra $Res_x(1\text{-}y,x+1)=-y+1$ polinomu bulunmalıdır. Bu polinomun kökü bize c_1 sabitinin değerini verecektir. Burada $y=1$ olduğundan $c_1=1$ olacaktır. Buradan v_1 polinomu $v_1=ebob(0,x+1)$ eşitliğinden hesap edilebilir. $v_1=x+1$ şeklinde bulunur. O halde kalan integralin sonucu $\int Q_2 / R_2 = \log(x+1)$ olur. Dolayısıyla f rasyonel fonksiyonunun sembolik integrali $\int f = Q_1 / R_1 + \int Q_2 / R_2 = -2 / (x+1) + \log(x+1)$ olarak bulunur.

3.2.2. Örnek : $f = (x^2-3x+2) / (x^3+4x^2+5x+2)$ olmak üzere f 'nin rasyonel integralini, yukarıda anlatılan yöntemi adım adım izleyerek aşağıdaki gibi hesaplayabiliriz.

$Q = x^2-3x+2$

$R = x^3+4x^2+5x+2$

$R' = 3x^2+8x+5$

$R_1 = ebob(R,R') = x+1$, $m = derece(R_1)=1$, $R_1' = 1$

$R_2 = R / R_1 = x^2+3x+2$, $n = derece(R_2)= 2$

$Q_1 = \sum_{i=0}^{0} a_i x^i = a_0$, $Q_1' = 0$

$Q_2 = \sum_{i=0}^{1} b_i x^i = b_0 + b_1 x$

$S = R_1' R_2 / R_1 = x+2$

$Q = x^2-3x+2 = Q_1' R_2 - Q_1 S + Q_2 R_1 = - a_0 (x+2) + (b_0+b_1 x)(x+1)$

$Q = x^2 b_1 + x(b_1+b_0-a_0) + (b_0 - 2a_0)$

$b_1 = 1$, $b_1+b_0-a_0 = -3$, $b_0 - 2a_0 = 2$ denklem sistemi elde edilir. Sistem çözülerek

$b_0 = -10$, $b_1 = 1$, $a_0 = -6$ bulunur. Buradan

$\int Q/R = \frac{-6}{x+1} + \int \frac{x-10}{x^2+3x+2}$ olur. Kalan integralin çözümü için

$Q_2 = x - 10$

$R_2 = x^2 + 3x + 2$

$R_2' = 2x + 3$

$Res(Q_2 - yR_2', R_2) = Res([x - 10] - y[2x + 3], x^2 + 3x + 2) = -y^2 + y + 132$

Bu denklemin kökleri $c_1 = -11$, $c_2 = 12$ olarak bulunur.

$v_1 = ebob(Q_2 - c_1R_2', R_2) = x + 1$

$v_2 = ebob(Q_2 - c_2R_2', R_2) = x + 2$ olur.

O halde

$$\int f = \int \frac{x^2 - 3x + 2}{x^3 + 4x^2 + 5x + 2} = \frac{-6}{x+1} + 12\log(x+2) - 11\log(x+1)$$

sonucu elde edilir.

4. BİRİNCİ DERECEDEN MANTIK

Bu bölümde kullanılan tüm tanım, teorem ve önermeler "Birinci Mertebeden Doğrusal Diferensiyel Denklemlere Dedaktif Yaklaşım" (Tolga Güyer, 1996) adlı Bilim Uzmanlığı Tezinden alınmıştır.

4.1. Temel Kavramlar

Birinci dereceden mantığın bir konu üzerine uygulanmasında kullanılan dilin sembolleri, konunun niteliğine göre değişkenlik göstermektedir. Ancak kullanılan temel kavramlar, *birinci dereceden dil* olarak adlandırılan bu dillerin tümünde ortaktır. Bu kavramları aşağıdaki gibi sıralayabiliriz:

(i) Değişken sembolleri.

(ii) Sabit sembolleri.

(iii) Fonksiyon sembolleri.

(iv) Predicate sembolleri.

(v) Bağlaç sembolleri : "$\neg$", "$\wedge$", "$\vee$", "$\rightarrow$", "$\leftrightarrow$".

(vi) Niceleyici sembolleri : "$\forall$", "$\exists$".

(vii) Noktalama sembolleri : "(", ")", ",".

Değişkenler ve sabitler, birinci dereceden mantığın en küçük birimleridir. Fonksiyon ve predicate sembolleri, değişken, sabit ya da fonksiyon sembollerini bileşen olarak bulundurabilirler.

Bağlaç sembollerinin anlamları, önermeler mantığındaki ile aynıdır. "$\neg$" : 'değil' sembolü, "$\wedge$" : 've' sembolü, "$\vee$" : 'veya' sembolü, "$\rightarrow$" : 'gerektirme' sembolü ve "$\leftrightarrow$" : 'çift yönlü gerektirme' sembolüdür. Niceleyici sembollerinden "$\forall$" evrensel niceleyiciyi, "$\exists$" ise varlık niceleyicisini sembolize etmektedir.

Bu bölümde verilen tanımlarda, predicate sembolü olarak "p", "q", "r"; fonksiyon sembolü olarak "f", "g", "h", "k" ,"j"; değişken sembolü olarak "*x*", "*y*", "z", "*u*", "*v*" ve sabit sembolü olarak "*a*", "*b*", "*c*" harfleri kullanılacaktır.

4.1.1. Tanım : *Terim,* özyinelemli olarak aşağıdaki gibi tanımlanır:

(i) Bir sabit, bir terimdir.

(ii) Bir değişken, bir terimdir.

(iii) f bir n-bileşenli fonksiyon sembolü ve $t_1, \ldots, t_n$ terimler olmak üzere $f(t_1, \ldots, t_n)$ bir terimdir.

Eğer bir terim hiçbir değişken içermiyorsa *kapalı terim* adını alır.

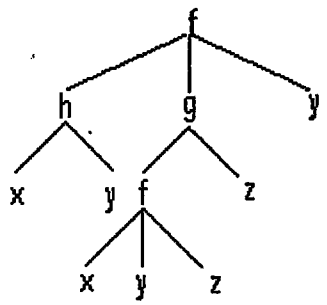
4.1.1. Örnek : f ,g ve h fonksiyon sembolleri; *a, b* sabitler ve *x, y* değişkenler olmak üzere aşağıdakiler birer terimdir:

$$g(f(x,h(x,y)), f(g(x,y),y))$$

$$h(f(a),g(a,h(a,b))).$$

Terimlerin gösteriminde karşılaşılan en önemli güçlük, büyük terimlerde iç içe yazılmalardan kaynaklanan parantez karmaşasıdır. Bu gibi durumlarda, terimlerin ağaç yapısı biçimindeki gösterimleri ile, onların oldukça açık ve anlaşılabilir olmalarını sağlayabiliriz. Örnek olarak, f(h(x,y),g(f(x,y,z),z),y) teriminin ağaç yapısı biçimindeki gösterimi Şekil 4.1.'de verilmiştir.

Ağaç yapısıyla gösterilen bir t terimini göz önüne alırsak, yapının en üstünde yer alan terim, t'nin *kökü* adını alır. Kökün alt kısımlarında yer alan diğer terimler ise *düğüm* adını alır. Terimin özyinelemli tanımını düşünecek olursak, bu yapıda bulunan düğümler de, kendi altlarında yer alan düğümlerin oluşturdukları ağaç yapılarının kökleri olacaktır. Diğer bir deyişle, bir terimin ağaç yapısındaki gösteriminde, uç noktalarda yer alan düğümlerin dışındaki tüm düğümler, aynı zamanda birer köktürler.



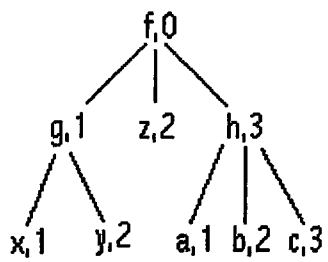
Şekil 4.1. Terimlerin ağaç yapısı biçimindeki gösterimleri.

4.1.2. Tanım : Ağaç yapısında verilen bir terimin, kökünün ve her bir düğümünün aşağıdaki kurallar yardımı ile numaralandırılmasıyla oluşan gösterimine o terimin *belirli ağaç gösterimi* adı verilir.

1. Terimin kökü 0 tamsayısı ile numaralandırılır. Kökün altında yer alan her bir düğüm soldan sağa doğru artan sıra ile 1'den başlayan tamsayılarla numaralandırılır.

2. Bir n tamsayısı ile numaralandırılmış olan bir düğümün altında yer alan diğer düğümler, soldan sağa doğru artan sıra ile 1'den başlayan tamsayılarla numaralandırılır.

4.1.2. Örnek : f(g(*x*,*y*),h(*a*,*b*,c),*z*) teriminin belirli ağaç gösterimi Şekil 4.2.'de verilmiştir.



Şekil 4.2. Terimlerin belirli ağaç gösterimleri.

4.1.3. Tanım : Bir (*iyi tanımlı*) *formül* özyinelemli olarak aşağıdaki gibi tanımlanır :

(i) p bir n-bileşenli predicate sembolü ve $t_1,...,t_n$ terimler olsun. Bu durumda $p(t_1,...t_n)$ bir formüldür. (Bu formüle *atomik formül* adı verilir.)

(ii) F ve G iki formül olsun. Bu durumda $(\neg F)$, $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$ ve $(F \leftrightarrow G)$ birer formüldür.

(iii) F bir formül ve x bir değişken olsun. Bu durumda $\forall x(F)$ ve $\exists x(F)$ birer formüldür.

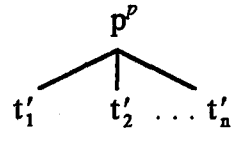
4.1.3. Örnek : p ve q predicate sembolleri, f bir fonksiyon sembolü ve x, y değişkenler olmak üzere aşağıdakiler birer formüldür :

$$\forall x \, \forall y \, (p(x,f(y)) \rightarrow q(x))$$

$$\neg \, (\exists x \, (p(x,f(y)) \vee \neg \, q(f(y))))$$

Terimlerin olduğu gibi atomik formüllerin de belirli ağaç gösterimleri tanımlanabilir.

4.1.4. Tanım : p bir n-bileşenli predicate sembolü ve $t_1, t_2, \ldots, t_n$ terimler olmak üzere $p(t_1, t_2, \ldots, t_n)$ atomik formülünü göz önüne alalım. $t'_1, t'_2, \ldots, t'_n$ sırasıyla $t_1, t_2, \ldots, t_n$ terimlerinin belirli ağaç gösterimleri olsun. Bu durumda, Şekil 4.3.'de verilen yapıya, $p(t_1, t_2, \ldots, t_n)$ atomik formülünün *belirli ağaç gösterimi* denir. Bu gösterimde p^p, belirli ağaç yapısının *kök predicate*'i adını alır. Burada p'nin üzerinde yer alan "*p*" sembolü, p'nin bir predicate olduğunu vurgulamaktadır.



Şekil 4.3. Atomik formüllerin belirli ağaç gösterimleri.

4.1.5. Tanım : F bir formül olmak üzere, $\forall x$'in -$\exists x$'in- $\forall x(F)$ -$\exists x(F)$- içindeki *alanı* F'dir. Bir değişkenin bir formül içersinde *sınırlı* olarak bulunması demek, ya o değişkenin bir niceleyicinin ardından gelmesi, ya da bir niceleyicinin alanı içersinde bulunması demektir.

Eğer bir değişken bir formül içersinde sınırlı değilse *serbest* adını alır.

4.1.6. Tanım : Hiçbir serbest değişken içermeyen formüle *kapalı formül* adı verilir.

4.1.7. Tanım : Bir atom ya da o atomun değiline bir *literal* adı verilir. *Pozitif literal* atomdur. *Negatif literal* ise atomun değilidir.

4.1.8. Tanım : Her bir L_i bir literal ve $x_1,...,x_s$; $L_1 \vee L_2 \vee ... \vee L_m$ içindeki değişkenler olmak üzere,

$$\forall x_1 ... \forall x_s (L_1 \vee ... \vee L_m)$$

biçimindeki formüle bir *clause* adı verilir.

4.1.4. Örnek : Aşağıdaki formüller birer clause'dur:

$$\forall x \forall y \forall z \, (p(x,y) \vee q(y,z) \vee \neg r(x,z))$$

$$\forall x \forall y \forall z (p(x,y) \vee \neg q(f(x,y),z) \vee r(z))$$

4.2. Birinci Dereceden Mantıkta Bir Formülün Yorumlanması

4.2.1. Tanım : Boş olmayan bir D kümesine bir *yorumlama bölgesi* adı verilir. {T,F} kümesi ise sırasıyla "doğru" ve "yanlış" doğruluk değerlerinin kümesini göstermektedir.

4.2.2. Tanım : Birinci dereceden mantıkta bir F formülünün *yorumlanması*, bir D yorumlama bölgesi ile F içindeki her sabit, fonksiyon sembolü ve predicate sembolüne aşağıdaki biçimde bir 'değer' atanması biçiminde gerçekleştirilir :

(i) Her sabite D'nin bir elemanı atanır.

(ii) $D^n = \{(x_1,...,x_n) \mid x_1 \in D,...,x_n \in D\}$ olmak üzere, her n-bileşenli fonksiyon sembolüne D^n'den D'ye bir dönüşüm atanır.

(iii) Her n-bileşenli predicate sembolüne D^n'den { T , F } kümesine bir dönüşüm atanır.

Bu tanımdan sonra, niceleyicilerin kazandıkları anlamlar daha açık olarak şu şekilde ifade edilebilir : Bir F formülünün, bir D yorumlama bölgesi üzerinde yorumlanmasında $\forall x$, "D içindeki her x elemanı için" ve $\exists x$, "D içinde bir x elemanı vardır." anlamını taşır.

Genel olarak bir D yorumlama bölgesi üzerinde bir formülün herhangi bir yorumlamasında formüle **T** ya da **F** değerlerinden birisi aşağıdaki kurallar yardımı ile atanır :

1. Eğer G ve H formüllerinin doğruluk değerleri **T** ya da **F** olarak hesaplanmışsa, $\neg$G, G$\wedge$H, G$\vee$H, G$\rightarrow$H ve G$\leftrightarrow$H formüllerinin doğruluk değerleri Çizelge 4.1. kullanılarak hesaplanabilir.
2. Eğer G formülünün doğruluk değeri D içindeki her d elemanı için **T** oluyorsa, $\forall x$G formülünün doğruluk değeri **T**'dir; aksi durumda $\forall x$G formülü **F** değerini alır.
3. Eğer G formülünün doğruluk değeri D içindeki en az bir d elemanı için **T** oluyorsa, $\exists x$G formülünün doğruluk değeri **T**'dir; aksi durumda $\exists x$G formülü **F** değerini alır.

4.2.3. Tanım : Kapalı bir G formülü, bir I yorumlamasında **T** değerini alıyorsa I'ya G'nin *modeli* adı verilir.

G	H	$\neg$G	G$\wedge$H	G$\vee$H	G$\rightarrow$H	G$\leftrightarrow$H
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Çizelge 4.1. Formüllerin doğruluk değerlerinin hesaplanması.

4.2.4. Tanım : G bir formül ve I bir yorumlama olsun. Bu durumda;

- Eğer $\exists$(G) I'da doğru ise G formülü I'da *sağlanabilir* denir.
- Eğer $\forall$(G) I'da doğru ise G formülü I'da *geçerlidir* denir.
- Eğer $\exists$(G) I'da yanlış ise G formülü I'da *sağlanamaz* denir.
- Eğer $\forall$(G) I'da yanlış ise G formülü I'da *geçersizdir* denir.

4.2.5. Tanım : S kapalı formüllerin bir kümesi ve I bir yorumlama olsun. Eğer I, S'deki her formül için bir model oluyorsa I'ya S'nin bir *modeli* denir.

4.2.6. Tanım : S kapalı formüllerin bir kümesi olsun. Bu durumda;

- Eğer S için model olan en az bir I yorumlaması varsa S *sağlanabilir* denir.
- Eğer S için her I yorumlaması bir model oluyorsa S *geçerlidir* denir.
- Eğer S için model olan hiçbir I yorumlaması yoksa S *sağlanamaz* denir.
- Eğer S için model olmayan en az bir I yorumlaması varsa S *geçersizdir* denir.

4.2.7. Tanım : S kapalı formüllerin bir kümesi ve F kapalı bir formül olsun. Eğer her I yorumlaması için, I'nın S için bir model olması, I'nın F için de bir model olmasını gerektiriyorsa, F'ye S'nin *mantıksal sonucu* denir.

4.2.1. Önerme : S kapalı formüllerin bir kümesi ve F kapalı bir formül olsun. Bu durumda F'nin S'nin mantıksal sonucu olması için gerekli ve yeterli koşul, $S \cup \{\neg F\}$ kümesinin sağlanamaz olmasıdır.

4.2.1. Örnek : $S=\{p(y), \forall x\ (\neg p(x) \vee q(x))\}$ ve $F=q(y)$ olsun. S'nin mantıksal sonucunun F olduğunu göstereceğiz. I, S için model olan herhangi bir yorumlama olsun. Bu durumda $p(y)$, I'da doğrudur. Bununla beraber $\forall x\ (\neg p(x) \vee q(x))$'de I'da doğru olduğundan, $\neg p(y) \vee q(y)$'da doğru olacaktır. Buradan $q(y)$'nın da I'da doğru olduğu sonucu çıkar. O halde, Tanım 4.2.7. gereğince, F, S'nin mantıksal sonucu olur.

4.3. Prenex Normal Formu

4.3.1. Tanım : Birinci dereceden mantıkta bir F formülünün *Prenex normal formu*'nda olması için gerekli ve yeterli koşul F'nin

$$Q_1x_1 \ldots Q_nx_n \ (M)$$

biçiminde olmasıdır. Burada her bir (Q_ix_i), i=1,...,n, $(\forall x_i)$ ya da $(\exists x_i)$ niceleyicilerinden birisi ve M hiçbir niceleyici içermeyen bir formüldür. Burada, $Q_1x_1 \ldots Q_nx_n$ bölümüne formülün *niceleyiciler bölümü,* M'ye ise formülün *matris bölümü* adı verilir.

4.3.1. Örnek : Aşağıdaki formüller Prenex normal formundadır :

$$\forall x \forall y \forall z(p(x,y) \vee q(y,z) \vee \neg r(x,z))$$

$$\forall x \forall y \ (p(x,y) \vee \neg q(f(x,y),y) \vee r(x))$$

Birinci dereceden mantıkta herhangi bir formül Prenex Normal Formuna dönüştürülebilir. Bunun için, F ve G, x değişkenini bulunduran formüller, H, x değişkenini bulundurmayan bir formül, z, F ve G formülleri içinde bulunmayan bir değişken, Q, $\forall$ ya da $\exists$ niceleyicilerinden birisi ve Q_1 ve Q_2, ikisi birbirinden farklı olmak koşuluyla, $\forall$ ya da $\exists$ niceleyicilerinden birisi olmak üzere aşağıda verilen kurallar kullanılır:

1. $F \leftrightarrow G = (F \rightarrow G) \wedge (G \rightarrow F)$
2. $F \rightarrow G = \neg F \vee G$
3. $\neg (\neg F) = F$
4. $\neg(F \vee G) = \neg F \wedge \neg G$
5. $\neg(F \wedge G) = \neg F \vee \neg G$
6. $\neg(\forall x (F)) = \exists x (\neg F)$

7. $\neg(\exists x\,(F)) = \forall x\,(\neg F)$

8. $\forall x\,(F) \wedge \forall x(G) = \forall x\,(F \wedge G)$

9. $\exists x\,(F) \vee \exists x(G) = \exists x\,(F \vee G)$

10. $Qx\,(F) \vee H = Qx\,(F \vee H)$

11. $Qx\,(F) \wedge H = Qx\,(F \wedge H)$

12. $Q_1x\,(F) \vee Q_2x\,(G) = Q_1x\,Q_2z\,(F \vee G)$

13. $Q_1x\,(F) \wedge Q_2x\,(G) = Q_1x\,Q_2z\,(F \wedge G)$

Bu kurallarda kullanılan eşitlik simgesi, eşitliğin sol ve sağ tarafındaki formüllerin aynı doğruluk değerini taşıdıklarını göstermektedir. Diğer bir deyişle, F ve G iki formül olmak üzere, F=G olması için gerekli ve yeterli koşul, F ve G'nin her yorumlama altında aynı doğruluk değerine sahip olmalarıdır.

4.3.2. Örnek : $\forall x\forall y(\exists z\ p(x,y,z)\vee(\forall xq(x,y)\rightarrow\exists x\neg r(x,y)))$ formülünü aşağıdaki biçimde Prenex normal formuna dönüştürebiliriz:

$\forall x\forall y(\exists z\ p(x,y,z)\vee(\forall xq(x,y)\rightarrow\exists x\neg r(x,y)))$

$= \forall x\forall y(\exists z\neg p(x,y,z)\vee(\neg(\forall xq(x,y))\vee\exists x\neg r(x,y)))$ *Kural 2'nin uygulanması.*

$= \forall x\forall y(\exists z\neg p(x,y,z)\vee(\exists x(\neg q(x,y))\vee\exists x\neg r(x,y)))$ *Kural 6'nın uygulanması.*

$= \forall x\forall y(\exists z\neg p(x,y,z)\vee(\exists x(\neg q(x,y)\vee\neg r(x,y))))$ *Kural 9'un uygulanması.*

$= \forall x\forall y\exists z\exists u(\neg p(x,y,z)\vee(\neg q(u,y)\vee\neg r(u,y)))$ *Kural 12'nin uygulanması.*

Bulunan son formül, verilen formülün Prenex normal formunda ifade edilmiş biçimidir.

4.4. Skolem Standart Formu

4.4.1. Tanım : F, Prenex normal formunda bir formül olsun. F formülünün *arakesit normal formunda* olması için gerekli ve yeterli koşul F'nin $F_1 \wedge F_2 \wedge \ldots \wedge F_n$ biçiminde olmasıdır.

4.4.1. Örnek : $\forall x \forall y \exists z(p(x,y) \wedge q(x,y,z))$ Prenex normal formu, aynı zamanda arakesit normal formundadır.

4.4.2. Tanım : F, arakesit normal formunda bir formül olsun. F'nin *Skolem standart formunda* olması için gerekli ve yeterli koşul, Q_1, Q_2,...,Q_n'lerin her biri birer evrensel niceleyici olmak üzere F'nin,

$$Q_1x\ Q_2x \ldots Q_nx\ (F)$$

biçiminde olmasıdır.

Bundan sonra Skolem standart formu yerine kısaca 'standart form' deyimi kullanılacaktır.

Birinci dereceden mantıkta herhangi bir formül standart forma dönüştürülebilir. Bunun için, formülün varlık niceleyicilerinden bağımsız olarak ifade edilmesi gerekmektedir. Bunu, verilen formülün niceleyiciler bölümünde her bir varlık niceleyicisi için, o varlık niceleyicisinin alanında bulunan değişkeni, bu niceleyiciden önce gelen evrensel niceleyicilerin alanlarında bulunan değişkenlerin bir fonksiyonu ile değiştirerek yapabiliriz. Burada seçilen fonksiyon sembolü, formülün matris bölümünde bulunmayan bir fonksiyon sembolü olmalıdır. Bu biçimdeki fonksiyonlara *Skolem Fonksiyonları* adı verilir.

4.4.2. Örnek : Örnek 4.3.2. de bulunan $\forall x \forall y \exists z \exists u(\neg p(x,y,z) \vee (\neg q(u,y) \vee \neg r(u,y)))$ prenex normal formu aşağıdaki şekilde skolem standart forma dönüştürülebilir.

$\forall x \forall y \exists z \exists u(\neg p(x,y,z) \vee (\neg q(u,y) \vee \neg r(u,y)))$

$\forall x \forall y \exists u(\neg p(x,y,f(x,y)) \vee (\neg q(u,y) \vee \neg r(u,y)))$

$\forall x \forall y(\neg p(x,y,f(x,y)) \vee (\neg q(g(x,y),y) \vee \neg r(g(x,y),y)))$

Bulunan son formül, verilen formülün Skolem standart formunda ifade edilmiş biçimidir. Burada f ve g fonksiyonları birer Skolem fonksiyonudur.

4.4.3. Tanım : Standart formdaki $F = \forall x_1 \forall x_4 .. \forall x_n \ (F_1 \wedge F_2 \wedge ... \wedge F_k)$ formülünü göz önüne alalım. $S^F = \{ F_1, F_2, ..., F_k \}$ kümesine F formülünün *clause kümesi* adı verilir.

4.4.1. Teorem : S^F, standart formdaki F formülünün clause kümesi olsun. Bu durumda F formülünün sağlanamaz olması için gerekli ve yeterli koşul S^F'nin sağlanamaz olmasıdır.

4.5. Herbrand Teoremi

Mekanik teorem ispatlama konusunda oldukça önemli bir atılım, 1930 yılında Herbrand tarafından yapılmıştır. Tanım 2.2.6. gereğince, geçerli bir formülün tüm yorumlamalar altında doğru değerini aldığını biliyoruz. Herbrand, verilen formülün yanlış değerini alacağı bir yorumlama arayan algoritmayı geliştirmiştir. Eğer formül geçerli ise, bu biçimde bir yorumlama elde edilememekte ve algoritma sonlu bir adımda durmaktadır. Bu yöntem, birçok modern otomatik ispatlama prosedürüne temel oluşturmuştur.

4.5.1. Tanım : Verilen bir S clause kümesinin H(S) ile gösterilen *Herbrand evrenseli,* indirgemeli olarak aşağıdaki biçimde tanımlanır:

(i) S'deki herhangi bir clause'da yer alan herhangi bir sabit sembolü, H(S)'nin bir elemanıdır. Eğer S'deki hiçbir clause sabit sembolü bulundurmuyorsa H(S), *"a"* sabitini içerir.

(ii) Eğer f, S'de n-bileşenli bir fonksiyon sembolü ve $t_1, t_2, ..., t_n$, H(S)'nin elemanları iseler, $f(t_1, t_2, ..., t_n)$'de H(S)'nin bir elemanıdır.

4.5.2. Tanım : S bir clause kümesi olmak üzere, H(S)'nin bir elemanına S'nin *Herbrand terimi* ya da *temel terimi* denir. S'nin bir *temel clause'u* ise, S'deki bir clause'un değişkenin S'nin temel terimleri ile düzgün olarak değiştirilmesi ile elde

edilir. Burada düzgün kelimesi ile anlatılmak istenen, değiştirme işlemi sırasında, farklı konumlarda bulunan aynı değişkenlerin, aynı temel terimlerle değiştirilmeleridir. Bu biçimde, bir S clause kümesinden türetilen tüm temel clause'ların kümesine S'nin *temel clause kümesi* denir ve S_H ile gösterilir. Bir F formülünün temel clause kümesi ise F'nin clause kümesinin temel clause kümesidir ve $(S^F)_H$ ile gösterilir.

4.5.1. Teorem : (Herbrand Teoremi) F Skolem standart formunda bir formül olsun. F'nin sağlanamaz olması için gerekli ve yeterli koşul F'nin sağlanamaz olan sonlu bir temel clause kümesinin bulunmasıdır. (Yani $(S^F)_H$'nin T gibi sonlu bir altkümeye sahip olmasıdır.)

4.5.1. Örnek : $F=\forall z((p(a)\vee p(b))\wedge(p(a)\vee q(b))\wedge\neg p(z))$ formülü veriliyor. Buradan S^F = { $(p(a)\vee p(b))$, $(p(a)\vee q(b))$,$\neg p(z)$ } olarak bulunur. $T\subset (S^F)_H$ olmak üzere T = { $(p(a)\vee p(b))$, $\neg p(a)$, $\neg p(b)$ } kümesinin sağlanamaz olduğu açıktır. O halde teorem gereğince S^F sağlanamazdır.

4.6. Dönüşümler

4.6.1. Tanım : i=1,...,n için v_i'ler birbirlerinden farklı değişkenler ve her bir t_i bir terim olmak üzere $\{t_1 / v_1 ,..., t_n / v_n\}$ biçimindeki sonlu kümeye bir *dönüşüm* adı verilir. Dönüşümler σ_i , i=1,..,n , sembolleri ile gösterilecektir.

Eğer $t_1,...,t_n$ temel terimler ise dönüşüm *temel dönüşüm* adını alır. Hiçbir eleman içermeyen dönüşüme ise *boş dönüşüm* adı verilir ve $\varnothing$ simgesi ile gösterilir.

4.6.1. Örnek : Aşağıdakiler birer dönüşümdür:

$$\{u/x, f(y)/z\}$$

$$\{g(f(a,b))/x, a/y, h(a,y)/z\}$$

4.6.2. Tanım : σ = { $t_1 / v_1 ,..., t_n / v_n$} bir dönüşüm ve F bir formül olsun. Bu durumda $F\sigma$ formülü, F içindeki her bir v_i , i=1,..,n , değişkeninin t_i terimi ile değiştirilmesi ile elde edilen formüldür ve F'nin *örneklemesi* adını alır.

4.6.2. Örnek : $F = p(x,y,z) \vee (q(u,f(x,y),z) \vee \neg r(u,v))$ formülü ile $\sigma_1 = \{a/x, k(y)/z\}$ ve $\sigma_2 = \{h(f(a,b))/u, g(x,z)/y\}$ dönüşümleri için $F\sigma_1$ ve $F\sigma_2$ aşağıdaki gibi olur:

$$F\sigma_1 = p(a,y,k(y)) \vee (q(u,f(a,y),k(y)) \vee \neg r(u,v)$$

$$F\sigma_2 = p(x,g(x,z),z) \vee (q(h(f(a,b))),f(x,g(x,z)),z) \vee \neg r(h(f(a,b)),v))$$

4.6.3. Tanım : $\sigma_1 = \{ t_1 / x_1, \ldots, t_n / x_n \}$ ve $\sigma_2 = \{e_1 / y_1, \ldots, e_m / y_m\}$ iki dönüşüm olsun. $\sigma_1 \circ \sigma_2$ ile gösterilen σ_1 ve σ_2 dönüşümlerinin bileşkesi,

$$\{t_1\sigma_2 / x_1 , \ldots , t_n\sigma_2 / x_n , e_1 / y_1 , \ldots , e_m / y_m\}$$

kümesinden, $t_i\sigma_2 = x_i$ olan $t_i\sigma_2 / x_i$ elemanları ile $y_j \in \{x_1,\ldots,x_n\}$ olan e_j / y_j elemanlarının silinmesi ile elde edilir.

4.6.1. Önerme : σ bir dönüşüm olsun. Bu durumda $\sigma \circ \varnothing = \varnothing \circ \sigma = \sigma$ olur.

4.6.3. Örnek : $\sigma_1 = \{x/y, u/v\}, \sigma_2 = \{u/x, a/v\}$ ve $\sigma_3 = \{z/u, g(x)/v\}$ olmak üzere, $(\sigma_1 \circ \sigma_2) \circ \sigma_3$ dönüşümünü aşağıdaki biçimde bulabiliriz :

$$(\sigma_1 \circ \sigma_2) \circ \sigma_3 = \{u/y, u/v, u/x, \underline{a/v}\} \circ \sigma_3$$

$$= \{u/y, u/v, u/x\} \circ \{z/u, g(x)/v\}$$

$$= \{z/y, z/v, z/x, z/u, \underline{g(x)/v}\}$$

$$= \{z/y, z/v, z/x, z/u\}$$

Burada altı çizilerek belirtilen elemanlar, tanım gereğince dönüşüm kümesinden silinen elemanlardır.

4.7. Unification Algoritması ve Unification Teoremi

4.7.1. Tanım : t_1 ve t_2 iki terim olsun. d_1 ve d_2, sırasıyla t_1 ve t_2'nin belirli ağaç gösterimlerinde birbirlerinden farklı iki düğüm olsun. Eğer d_1 ve t_1'in kökünü birleştiren yol ile d_2 ve t_2'nin kökünü birleştiren yollar aynı yollar ise, t_1 teriminin d_1 düğümünün kök durumunda olduğu alt terimi ile t_2 teriminin d_2 düğümünün kök

durumunda olduğu alt teriminin oluşturdukları kümeye, t_1 ve t_2'nin *uyumsuzluk kümesi* adı verilir.

4.7.1. Örnek : $f(x,g(a,x),h(x,y))$ ve $f(u,g(a,x),h(x,y))$ terimlerinin uyumsuzluk kümesi $\{x,u\}$dir.

4.7.2. Tanım : σ bir dönüşüm ve t_1, t_2 iki terim olsun. σ'nın, t_1 ve t_2'nin bir *unifier*'i olması için gerekli ve yeterli koşul $t_1\sigma=t_2\sigma$ olmasıdır.

4.7.3. Tanım : σ, t_1 ve t_2 terimlerinin bir unifier'i olsun. σ'nın t_1 ve t_2'nin *en genel unifier*'i olması için gerekli ve yeterli koşul t_1 ve t_2'nin her σ_1 unifier'i için $\sigma_1=\sigma\circ\sigma_2$ olacak biçimde bir σ_2 dönüşümünün bulunmasıdır.

Pratik olarak tanım, verilen terimlerin en genel unifier'inin bulunmasında yetersiz kalacaktır. Bunun için, *unification algoritması* adı verilen aşağıdaki algoritma kullanılabilir :

Unification Algoritması

t_1 ve t_2 terimler olmak üzere, $W=\{t_1, t_2\}$ kümesini göz önüne alalım.

1.Adım : $k=0$, $W_k=W$ ve $\sigma_k=\varnothing$ olarak alınır.

2.Adım : Eğer W_k bir tek terim içeriyorsa algoritma sonlanır; σ_k, W için en genel unifier'dir. Aksi durumda W_k'nın uyumsuzluk kümesi D_k bulunur.

3.Adım : D_k kümesi; v_k, e_k teriminde bulunmayan bir değişken sembolü olmak üzere, v_k ve e_k elemanlarını bulunduruyorsa 4'ncü adıma geçilir. Aksi durumda, yani D_k içersindeki e_k terimi, yine D_k içersinde bulunan bir v_k değişkenini bulunduruyorsa, algoritma sonlanır. Bu durumda W kümesinin en genel unifier'i yoktur.

4.Adım : $\sigma_{k+1}=\sigma_k\circ\{e_k / v_k\}$ ve $W\sigma_{k+1}=W_{k+1}=W_k\{e_k / v_k\}$ olarak alınır.

5. Adım : $k=k+1$ alınır ve 2'nci adıma dönülür.

4.7.2. Örnek : $W=\{p(x,g(a,b),h(x,y)),p(u,g(a,b),h(x,y))\}$ kümesinin en genel unifier'ini bulalım:

1) $k\leftarrow 0$, $W_0=W=\{p(x,g(a,b),h(x,y)),p(u,g(a,b),h(x,y))\}$, $\sigma_0=\varnothing$.

2) W_0 birden fazla terim içerdiğinden işlem sonlanmaz; W_0'ın D_0 uyumsuzluk kümesi bulunur: $D_0=\{x,u\}$.

3) Bir $v_0=u\in D_0$ değişkeni var ve bu değişken $t_0=x\in D_0$ teriminde bulunmamaktadır. Dolayısıyla bir sonraki adıma geçilebilir.

4) $\sigma_1=\sigma_0\circ\{t_0/v_0\}=\varnothing\circ\{x/u\}=\{x/u\}$,

$$W_1=W_0\sigma_1=\{p(x,g(a,b),h(x,y)),p(u,g(a,b),h(x,y)\}\circ\{x/u\}$$

$$=\{p(x,g(a,b),h(x,y)),p(x,g(a,b),h(x,y))\}$$

$$=\{p(x,g(a,b),h(x,y))\}$$

5) $k\leftarrow 1$ ve 2'nci adıma dönülür.

6) W_1 tek terim içerdiğinden işlem sonlanır;

7) W_1 tek terim içerdiğinden σ_1, W'nin en genel unifier'idir.

4.7.1. Teorem : (Unification Teoremi) t_1 ve t_2 iki terim olsun. Eğer t_1 ve t_2 bir unifier'e sahip değilse algoritma 3'ncü adımda başarısızlıkla sonuçlanır. Aksi durumda algoritma her zaman 2'nci adımda sonlanacaktır. Bu durumda elde edilen en son σ_k, t_1 ve t_2'nin en genel unifier'i olur.

Yukarıda verilen unification algoritması, sonuç olarak iki terimin en genel unifier'ini vermektedir. Bu sebeple, *ikili unification* olarak adlandırılabilir. Ancak bu işlem, *katlı unification* adı verilen yöntemle sonlu sayıda terim içeren bir kümenin en genel unifier'ini bulmaya yönelik olarak genelleştirilebilir. Bu, ikili unification'ın ardışık olarak uygulanması ile yapılır.

Terimlerden oluşan $T=\{t_0, t_1, t_2, \ldots, t_n\}$ kümesini ve bir σ dönüşümünü göz önüne alalım. Eğer $\sigma t_0=\sigma t_1=\sigma t_2=\ldots=\sigma t_n$ oluyorsa, σ, T için bir *unifier* olur. T kümesinin böyle bir unifier'e sahip olduğunu varsayalım. Bu durumda, her biri ikili unification algoritması kullanılarak hesaplanabilecek elemanlardan oluşmak üzere, bir dönüşümler dizisini aşağıdaki biçimde oluşturabiliriz :

σ_1, t_0 ve t_1 terimlerinin en genel unifier'idir.

σ_2, $t_0\sigma_1$ ve $t_2\sigma_1$ terimlerinin en genel unifier'idir.

σ_3, $t_0(\sigma_1 \circ \sigma_2)$ ve $t_3(\sigma_1 \circ \sigma_2)$ terimlerinin en genel unifier'idir.

:

:

:

σ_n, $t_0(\sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_{n-1})$ ve $t_n(\sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_{n-1})$ terimlerinin en genel unifier'idir.

Bu durumda $(\sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_{n-1} \circ \sigma_n)$, T kümesi için *en genel unifier* olacaktır.

4.8. Resolution İlkesi

Verilen bir S clause kümesinin sağlanamazlığının gösterilmesinde Herbrand Teoreminin uygulanması, S'deki eleman sayısı arttıkça güçleşmektedir. Çünkü böyle bir S kümesi için S_H'nin sağlanamaz olan sonlu bir altkümesinin sezgisel olarak bulunması oldukça güçtür. Böyle bir altkümenin elde edilmesi için kullanılacak algoritmalardan da, işlem sayısının yine S'nin eleman sayısı ile doğru orantılı olarak üstel bir biçimde artması sebebiyle, etkin olarak sonuç alınamamaktadır.

1965 yılında Robinson tarafından tanıtılan Resolution ilkesi, bir S clause kümesinin sağlanamaz olduğunun gösterilmesi için doğrudan S'nin kendisine uygulanan bir yöntemdir. Temelde resolution ilkesi, Davis ve Putnam (1960) tarafından verilen *bir literal kuralı*'nın bir uzantısı niteliğindedir.

4.8.1. Tanım : Bir C clause'unu göz önüne alalım. Eğer C içinde, aynı işareti taşıyan iki ya da daha fazla literal σ gibi bir en genel unifier'e sahipse, $C\sigma$'ya C'nin bir *çarpanı* adı verilir.

4.8.2. Tanım : C_1 ve C_2 ortak değişken bulundurmayan iki clause olsunlar. L_1 ve L_2 sırasıyla C_1 ve C_2 içersinde iki literal olsun. Eğer L_1 ve $\neg L_2$, σ gibi bir en genel unifier'e sahip ise, bu durumda,

$$(C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma)$$

clause'una C_1 ve C_2'nin *ikili resolventi* denir.

4.8.3. Tanım : C_1 ve C_2 clause'larının *resolventi*, aşağıdaki ikili resolventlerden birisidir:

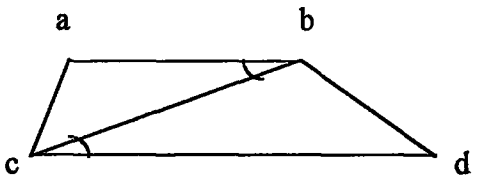
1. C_1 ile C_2'nin bir ikili resolventi,
2. C_1 ile C_2'nin bir çarpanının ikili resolventi,
3. C_1'in bir çarpanı ile C_2'nin ikili resolventi,
4. C_1'in bir çarpanı ile C_2'nin bir çarpanının ikili resolventi.

Resolution ilkesi, verilen bir clause kümesinden bir sonuç elde etmek amacıyla ardışık olarak resolventlerin üretildiği bir yöntemdir. Elde edilecek bu sonuç, amaca göre değişkenlik gösterir. Eğer resolution ilkesi, bir teoremin ispatlanmasında kullanılıyorsa, bulunacak sonuç, □ simgesi ile gösterilen boş clause olacaktır. Bunun, verilen clause kümesinin sağlanamaz olması ile ilgisi, daha sonra verilecek resolution'un tamlığı teoremi ile anlaşılacaktır. Diğer yandan, eğer resolution, sonuç olarak bir cevabın beklendiği bir probleme uygulanıyorsa, bu cevap bir *sonlandırma clause*'u ile elde edilecektir.

4.8.4. Tanım : Resolution ilkesi kullanılarak, bir S clause kümesinden, sonuç olarak bir C clause'unun elde edilmesi işlemine bir *dedüksiyon* adı verilir.

4.8.1. Teorem : (Resolution İlkesinin Tamlığı) Bir S clause kümesinin sağlanamaz olması için gerekli ve yeterli koşul, boş clause'a bir dedüksiyonun bulunmasıdır.

4.8.2. Örnek : Resolution ilkesinden yararlanarak şekildeki gibi bir abcd yamuğunda açı(abc) = açı(bcd) eşitliğini gösterelim.



Şekil 4.4. abcd Yamuğu

Bu teoremi resolution kullanarak ispatlamak için öncelikle tanımlamalarımızı yapalım:

Y(x, y, u, v) : üst-sol köşe x, üst-sağ köşe y, alt-sağ köşe u ve alt-sol köşe v olmak üzere xyuv yamuğunu göstersin.

P(x, y, u, v) : xy doğru parçasının uv doğru parçasına paralel olduğunu göstersin.

E(x, y, z, u, v, w) : xyz açısının uvw açısına eşit olduğunu göstersin.

Yukarıdaki tanımlamalardan yararlanarak aşağıdaki kuralları yazabiliriz :

$A_1 : (\forall x)(\forall y)(\forall u)(\forall v)\{Y(x,y,u,v) \rightarrow P(x,y,u,v)\}$: Yamuk tanımı.

$A_2 : (\forall x)(\forall y)(\forall u)(\forall v)\{P(x,y,u,v) \rightarrow E(x,y,v,u,v,y)\}$: Paralel iki doğruyu kesen bir doğruda iç-ters açılar eşittir kuralı.

$A_3 : Y(a,b,c,d)$: Şekil 4.4. ile verilen bilgi.

İspatı yapabilmek için

$$A_1 \wedge A_2 \wedge A_3 \neg E(a,b,d,c,d,b)$$

formülünün sağlanamaz olduğunu göstermeliyiz. Önce formülü standart forma çevirelim:

$S = \{ \neg Y(x,y,u,v) \vee P(x,y,u,v) , \neg P(x,y,u,v) \vee E(x,y,v,u,v,y) , Y(a,b,c,d) , \neg E(a,b,d,c,d,b) \}$

Şimdi S clause kümesinin sağlanamaz olduğunu resolution ile gösterelim.

(1) $\neg Y(x,y,u,v) \vee P(x,y,u,v)$: S kümesinde bir clause

(2) $\neg P(x,y,u,v) \vee E(x,y,v,u,v,y)$: S kümesinde bir clause

(3) $Y(a,b,c,d)$: S kümesinde bir clause

(4) $\neg E(a,b,d,c,d,b)$: S kümesinde bir clause

(5) $\neg P(a,b,c,d)$: (2) ve (4) clause'larının bir resolventi

(6) $\neg Y(a,b,c,d)$: (1) ve (5) clause'larının bir resolventi

(7) $\square$: (3) ve (6) clause'larının bir resolventi

S clause kümesinin resolution'ından elde edilen son clause $\square$ olduğundan S kümesi sağlanamazdır. O halde Şekil 4.4. deki gibi bir yamukta açı(abd) = açı(bdc) olduğu resolution kullanılarak ispatlanmıştır.

5. RESOLUTION İLE SEMBOLİK ÇÖZÜM

5.1. Tanımlamalar

Bu kesimde, rasyonel fonksiyonların integralinin çözümünde kullanılacak clause'lar birinci dereceden mantık yapısı içerisinde tanımlanacaktır. Tanımlanan clause'ların REDEUCE kodları Ek - 2 de verilmiştir.

5.1.1. Tanım : A, B, C polinom sembolleri, n,m sabit sembolleri, x bir değişken ve L, L1, L2, L3 liste sembolleri olmak üzere çalışmamızda kullanılacak predicateler aşağıdaki şekilde tanımlanır :

1. =(A, B) : $A(x) = B(x)$

2. coeff(A, L) : $L = \{ a_i , i = 0,1,..,n : A(x) = a_n x^n + ... + a_1 x + a_0 \}$

3. d(A, B) : $B(x) = dA(x) / dx$

4. deg(A, n) : $n = derece(A(x))$

5. findroot(A, L, x) : $L = \{ a_i , i = 0,1,..,n : A(x) = (x-a_n)...(x-a_1)(x-a_0) \}$

6. gcd(A, B, C) : $A(x) = ebob(B(x), C(x))$

7. resultant(A, B, C) : $A(x) = Res(B(x), C(x))$

8. control(A, cnt) : $cnt = \begin{cases} 1 & ; a_i \in IR \\ 0 & ; a_i \notin IR \end{cases} : A(x) = (x-a_n)...(x-a_1)(x-a_0)$

9. gcd1(L, B, L1) : $L1 = \{ ebob(A_i(x), B(x)) : A_i(x) \in L \}$

10. sys(A, B, L) : $L = \{a_i-b_i=0, i=0,1,...,max(n,m) : A(x) = a_n x^n + ... + a_1 x + a_0$

$; B(x) = b_m x^m + ... + b_1 x + b_0 \}$

11. solsys(L,L1,L2,L3,n,m): $L1 = \{ a_i , i=0,1,...,max(n,m) : a_i-b_i=0$ ve $a_i, b_i \in L \}$

$L2 = \{ b_i , i=0,1,...,max(n,m) : a_i-b_i=0$ ve $a_i, b_i \in L \}$

$$L3 = \{ a_i , b_i : i = 0,1,...,max(n,m) \}$$

12. sub(L, A, B) : $B(x) = A(n_i)$, $n_i \in L$, $i=0,1,...,k$

13. sub1(L, A, L1) : $L1 = \{ A(n_i) , n_i \in L , i=0,1,...,k \}$

14. sub2(L,L1,L2,x) : $L2 = \{ A_i(n_i) , n_i \in L, A_i(x) \in L1 , i=0,1,...,k \}$

15. sum(n, L, A) : $A(x) = \sum_{i=1}^{n} B_i$; $B_i \in L$

16. sum1(n, L, A) : $A(x) = \sum_{i=0}^{n} a_i x^i$; $L = \{a_0, a_1, a_2, \dots , a_n \}$

17. intras(A, B) : A(x) = Q/R için B(x) = Q1/R1+∫ Q2 / R2

18. intlog(A, B, C) : A(x) ve B(x) polinom olmak üzere $C(x) = \sum_{i=1}^{n} c_i \log(v_i)$

19. integral(A) : $\int A(x)dx$: A(x) rasyonel bir fonksiyon

20. solution(A) : üretilen çözüm

5.1.2. Tanım : x, y matematiksel ifadeler, n sabit sembolü olmak üzere toplam, fark, çarpım, bölüm, kuvvet ve logaritma fonksiyonları aşağıdaki şekilde tanımlanır :

1. +(x, y) : x + y

2. -(x, y) : x - y

3. *(x, y) : x * y

4. /(x, y) : x / y

5. ^(x, n) : x^n

6. log(x) : log(x)

5.1.3. Tanım : Horowitz-Ostrogradski Yöntemine bağlı olarak geliştirilen clause'ların ifadeleri aşağıdaki biçimdedir. Burada, kolaylık açısından, N_i ile clause'lardaki değişkenler gösterilmek üzere $\forall_i N_i$, o clause'a ait evrensel niceleyicilerdir.

(F_1) $\forall_i N_i$ integral(/(Q,R))∧control(R,1)→intras(/(Q,R),+(/(Q1,R1),S))

(F_2) $\forall_i N_i$ intras(/(Q,R),S)∧d(R,DR)∧gcd(R,DR,R1)∧=(R2,/(R,R1))

∧d(R1,DR1)∧deg(R1,M)∧deg(R2,N)∧sum1(M-1,LA,AX)

∧sum1(N-1,LB,BX)∧d(AX,DAX)

∧=(NQ,+(-(*(DAX,R2),*(AX,S)),*(BX,R1)))∧coeff(Q,CQ)

∧coeff(NQ,CNQ)∧sys(CQ,CNQ,L)∧solsys(L,C,S1,S2,N,M)

∧sub(S1,AX,Q1)∧sub(S2,BX,Q2)→intlog(Q1,R1,/(Q2,R2))

(F_3) $\forall_i N_i$ intlog(Q1,R1,/(Q2,R2))∧d(R2,DR2)

∧resultant(Z,-(Q2,*(Y,DR2)),R2)∧findroot(Z,C,Y)

∧sub1(C,-(Q2,*(Y,DR2)),VL)∧gcd1(VL,R2,V)

∧sub1(C,*(Y,log(W)),E)∧sub2(V,E,F,W)∧sum(N,F,SOL)

→solution(+(/(Q1,R1),SOL))

(F_4) . . . (F_{19}) clause'ları Tanım 5.1.1 de verilen 1-15 predicate'lerinden oluşmuştur.

(F_{20}) $\forall_i N_i$ integral(A)

5.1.4. Tanım : Horowitz-Ostrogradski Yöntemine bağlı olarak geliştirilen clause'ların standart formdaki ifadeleri aşağıdaki biçimdedir.

(A_1) $\forall_i N_i$ ¬integral(/(Q,R))∨¬control(R,1)∨ intras(/(Q,R),+(/(Q1,R1),S))

(A_2) $\forall_i N_i$ ¬intras(/(Q,R),S)∨¬d(R,DR)∨¬gcd(R,DR,R1)∨¬¬=(R2,/(R,R1))

$\vee\neg$d(R1,DR1)$\vee\neg$deg(R1,M)$\vee\neg$deg(R2,N)$\vee\neg$sum1(M-1,LA,AX)

$\vee\neg$sum1(N-1,LB,BX)$\vee\neg$d(AX,DAX)

$\vee\neg$=(NQ,+(-(*(DAX,R2),*(AX,S)),*(BX,R1)))$\vee\neg$coeff(Q,CQ)

$\vee\neg$coeff(NQ,CNQ)$\vee\neg$sys(CQ,CNQ,L)$\vee\neg$solsys(L,C,S1,S2,N,M)

$\vee\neg$sub(S1,AX,Q1)$\vee\neg$sub(S2,BX,Q2)$\vee$intlog(Q1,R1,/(Q2,R2))

(A_3) $\forall_i N_i$ $\neg$intlog(Q1,R1,/(Q2,R2))$\vee\neg$d(R2,DR2)

$\vee\neg$resultant(Z,-(Q2,*(Y,DR2)),R2)$\vee\neg$findroot(Z,C,Y)

$\vee\neg$sub1(C,-(Q2,*(Y,DR2)),VL)$\vee\neg$gcd1(VL,R2,V)

$\vee\neg$sub1(C,*(Y,log(W)),E)$\vee\neg$sub2(V,E,F,W)$\vee\neg$sum(N,F,SOL)

$\vee$solution(+(/(Q1,R1),SOL))

(A_4) . . . (A_{19}) clause'ları Tanım 5.1.1 de verilen 1-15 predicate'lerinden oluşmuştur.

(A_{20}) $\forall_i N_i$ integral(A)

5.1.5. Tanım : $A = \forall_i N_i(A_1 \wedge A_2 \wedge \ldots \wedge A_{19} \wedge A_{20})$ formülü için S^A clause kümesi $S^A = \{ A_1, A_2, \ldots, A_{20} \}$ şeklinde tanımlıdır. Burada S^A kümesi Horowitz-Ostrogradski yöntemine uygun olarak geliştirilen clause'lardan oluşmaktadır.

5.1.1. Teorem : Tanım 5.1.5 de verilen S^A kümesinin mantıksal sonucu solution(A) dır.

İspat : Önerme 4.2.1 gereği solution(A) formülünün S^A kümesinin bir mantıksal sonucu olması için gerekli ve yeterli koşulun $S^A \cup (\neg$solution(A)) kümesinin sağlanamaz olması olduğunu biliyoruz. Teorem 4.8.1. (Resolution İlkesinin Tamlığı) den bir S clause kümesinin sağlanamaz olması için gerekli ve yeterli koşulun boş clause'a ($\square$) bir dedüksiyon bulunması olduğunu biliyoruz. Buradan $S^A \cup (\neg$solution(A)) clause kümesinden resolution sonucu $\square$ clause'a ulaşırsak

solution(A) formülünün S^A kümesinin mantıksal sonucu olduğunu gösterebiliriz. O halde

(1) ¬integral(/(Q,R))∨¬control(R,1)∨ intras(/(Q,R),+(/(Q1,R1),S))

: S^A kümesinden bir clause

(2) ¬intras(/(Q,R),S)∨¬d(R,DR)∨¬gcd(R,DR,R1)∨¬=(R2,/(R,R1))

∨¬d(R1,DR1)∨¬deg(R1,M)∨¬deg(R2,N)∨¬sum1(M-1,LA,AX)

∨¬sum1(N-1,LB,BX)∨¬d(AX,DAX)

∨¬=(NQ,+(-(*(DAX,R2),*(AX,S)),*(BX,R1)))∨¬coeff(Q,CQ)

∨¬coeff(NQ,CNQ)∨¬sys(CQ,CNQ,L)∨¬solsys(L,C,S1,S2,N,M)

∨¬sub(S1,AX,Q1)∨¬sub(S2,BX,Q2)∨intlog(Q1,R1,/(Q2,R2))

: S^A kümesinden bir clause

(3) ¬intlog(Q1,R1,/(Q2,R2))∨¬d(R2,DR2)

∨¬resultant(Z,-(Q2,*(Y,DR2)),R2)∨¬findroot(Z,C,Y)

∨¬sub1(C,-(Q2,*(Y,DR2)),VL)∨¬gcd1(VL,R2,V)

∨¬sub1(C,*(Y,log(W)),E)∨¬sub2(V,E,F,W)∨¬sum(N,F,SOL)

∨solution(+(/(Q1,R1),SOL))

: S^A kümesinden bir clause

(4) . . . (19) : S^A kümesindeki clause'lar

(20) integral(A) : S^A kümesinden bir clause

(21) ¬solution(A) : verilen formül

(22) $\neg$control(R,1)$\vee$ intras(/(Q,R),+(/(Q1,R1),S)) : (1)-(20)

(23) intras(/(Q,R),+(/(Q1,R1),S)) : (11)-(22)

:

:

(39) intlog(Q1,R1,/(Q2,R2)) : (15)-(38)

:

:

(48) solution(+(/(Q1,R1),SOL)) : (18)-(47)

(49) □ : (21)-(48)

resolution işlemleri sonucu solution(A), S^A kümesinin mantıksal sonucudur. □

ÖNERİLER

Rasyonel fonksiyonların integrallerinin sembolik olarak bulunması için geliştirilen bu yöntemin, trigonometrik ve logaritmik fonksiyonların integrallerinin sembolik çözümleri, kompleks integrallerin sembolik çözümleri ve çözüme ulaşmak için sembolik integrasyon gerektiren tüm çalışmalarda uygulama alanı bulacağı ümit edilmektedir.

KAYNAKLAR

Chang, C., L., Lee, R., C., T., 1973, *Symbolic Logic and Mechanical Theorem Proving* : Academic Press, London.

Davenport, J.,H., Siret, Y., Tournier, E., 1993, *Computer Algebra, Systems and Algorithms for Algebraic Computation* : Academic Press, London.

Davis, M., Putnam, H., 1960, *A Computing Procedure for Quantification Theory* : J. Assoc. Comput. Mach., 7, 201-215.

Fitting, M., 1990, *First Order Logic and Automated Theorem Proving* : Springer-Verlag, 97-152.

Harrington, S., J., 1979, *A New Symbolic Integration System in REDUCE* : Computer J., 22, 2.

Horowitz, E., 1971, *Algorithms for Partial Fraction Decomposition and Rational Function Integration* : Proc. 2nd ACM Symp. Symbolic and Algebraic Manipulation, Los Angeles.

Lloyd, J., W., 1987, *Foundations of Logic Programming* : Springer-Verlag, 4 - 26, 35 - 136.

Mignotte, M., 1982, *Some Useful Bounds. Symbolic and Algebraic Computation* : Springer-Verlag, Wien, New York.

Robinson, J., A., 1965, *A Machine - Oriented Logic Based on the Resolution Princible* : Journal of the Association for Computing Machinery, 12, 1, 23-41.

Hearn, A., C., 1991, *REDUCE User's Manual. Version 3.4.* : RAND, Santa Monica.

Algoritmalar **EK-1**

Bölüm 2'de verilen bazı yöntemlerin algoritmik formları aşağıdadır.

Tek Değişkenli İki Fonksiyon için EBOB Algoritması :

1. M := *Landau_Mignotte_Sınırı* (A,B)

2. X := *ebob*(*baş_katsayı*(A),*baş_katsayı*(B))

3. p := *asal_bul*(X)

4. C := *modüler_ebob*(A,B,p)

5. *derece*(C) = 0 ise EBOB := 1, akışı durdur.

6. K := p

7. Sonuç := C

8. K <= 2M olduğu sürece

    9. p := *asal_bul*(X)

    10. C := *modüler_ebob*(A,B,p)

    11. *derece*(C) < *derece*(Sonuç) ise 5. adıma dön

    12. *derece*(C) = *derece*(Sonuç) ise

        13. Sonuç := *ÇKT*(Sonuç,K,C,p)

        14. K := K*p

15. *böler*(Sonuç,A) ve *böler*(Sonuç,B) ise EBOB := Sonuç , akışı durdur.

16. 3. adıma dön

Algoritmada yer alan fonksiyonlar :

Landau_Mignotte_Sınırı(X , Y) : X ve Y fonksiyonlarına Landau-Mignotte eşitsizliğini uygulayarak X fonksiyonunun katsayılarını sınırlar.

ebob(X , Y) : X ve Y tamsayılarının Euclid Algoritmasıyla ebob'nini bulur.

baş_katsayı(X) : X fonksiyonun baş katsayısını bulur.

asal_bul(X) : Her seferinde X'i bölemeyen bir asal sayı bulur.

modüler_ebob(X , Y , n) : Euclid Algoritmasını (mod n)'e göre uygular.

derece(X) : X polinomunun derecesini bulur.

ÇKT(X , Y , U , n) : Her iki polinomun katsayılarına Çinlilerin Kalan Teoremini uygular.

böler(X , Y) : X tamsayısının Y polinomunu bölüp bölemediğine bakar.

RESULTANT Algoritması

1. n := *derece*(f)

2. m := *derece*(g)

3. n > m ise

    4. r := $(-1)^{nm}$**resultant*(g,f)

5. Aksi halde

    6. a_n := *baş_katsayı*(f)

    7. n := 0 ise

        8. r := a_n^m

    9. Aksi halde

        10. h := *kalan*(g,f)

        11. h := 0 ise

            12. r := 0

        13. Aksi halde

            14. p := *derece*(h)

            15. r := a_n^{m-p}**resultant*(f,h)

16. Sonuç := r , akışı durdur.

Algoritmada yer alan fonksiyonlar :

kalan(X , Y) : X ile Y polinomundan kalanı verir.

resultant(X , Y) : Rekörsif olarak kendini çağırır.

Kuralların REDUCE kodları **EK - 2**

Bölüm 5' de verilen kuralların REDUCE dilindeki tanımlamaları aşağıdaki gibidir.

```
1. =( A, B )          : A := B;

2. coeff( A, L )      : L := coeff( A, x );

3. d( A, B )          : B := df( A, x );

4. deg( A, n )        : n := deg( A, x );

5. findroot( A, L, x ) : L := solve( A , x );

6. gcd( A, B, C )     : A := gcd( B, C );

7. resultant( A, B, C ): A := resultant( B, C, x );

8. control( A, cnt )  : dr := df(A,i);
                        if dr = 0 then cnt := 1 else cnt := 0;

9. gcd1( L, B, L1 )   : L1 := {};
                        n := lenght( L );
                        for i := 1 : n do
                        <<
                              A := first( L );
                              m := gcd( A, B );
                              L1 := append( L1, m );
                              L := rest( L );
                        >>;
```

```
10. sys( A, B, L )      : la := coeff( A, x ); lb := coeff( B, x );
                          lla := length( la ); llb := length( lb );
                          if  lla < llb then
                          <<
                              w := llb - lla;
                              for i := 1 : w do
                                  la := append( la, {0} );
                              degree := degree + w;
                          >>;
                          for k := degree step -1 until 0 do
                          <<
                              L := append( L, { 0 = first( lb ) - first( la ) } );
                              la := rest( la ); lb := rest( lb );
                          >>;
11. solsys(L,C,A,B,n,m): D := solve( L, C ) ;
                          for i := 1 : n do
                          <<
                              L1 := append( L1, first( D ) ) ;
                              D := rest( D ) ;
                          >>;
```

```
                for i := 1 : m do
                <<
                    L2 := append( L2, first( D ) ) ;
                    D := rest( D ) ;
                >>;
12. sub( L, A, B )      : n := lenght( L );
                for  i := 1 : n do
                <<
                    c := first( L );  A := sub( c, A );
                    L := rest( L );
                >>;
                B := A;
13. sub1( L, A, L1 )    : L1 := {};
                n := lenght( L );
                for i := 1 : n do
                <<
                    c := first( L );  A := sub( c, A );
                    L1 := append( L1, A );
                    L := rest( L );
                >>;
```

14. sub2(L,L1,L2,W): L2 := {} ;

n := lenght(L) ;

for i := 1 : n do

<<

FL := first(L) ; EL := first(L1) ;

L2 := append(L2, sub(W = FL, EL)) ;

L := rest(L) ; L1 := rest(L1) ;

>>;

15. sum (n, L, A) : A := 0;

for i = 1 : n do

<<

A := A + first(L);

L := rest(L);

>>;

16. sum1(n, L, A) : for i = 1 : n do

<<

A := A + first(L)*x^i ;

L := rest(L) ;

>> ;

REDUCE Programı Kaynak Kodu **EK-3**

```
PROCEDURE LP(P1,P2); { Logaritmik kısmın çözümü }
BEGIN
ARRAY K(DZ);
ARRAY V(DZ);
QQ:=P1;
RR:=P2;
RESULT:=0;
DRR:=DF(RR,X);
Z:=RESULTANT(QQ-Y*DRR,RR,X);
{ Resultant işlemiyle Z polinomu oluşturuluyor }
DZ:=DEG(Z,Y);
KOK:=SOLVE(Z,Y);
{ kökler hesaplanıyor }
FOR I:=1:DZ DO
<<
K(I):=FIRST(KOK);
KOK:=REST(KOK);
>>;
FOR J:=1:DZ DO
<<
TRANS:=SUB(K(J),QQ-Y*DRR);
{ bulunan kökler fonksiyonda yerine yazılıyor }
V(J):=GCD(TRANS,RR); { Vi ler oluşturuluyor }
RESULT:=RESULT+SUB(K(J),Y*LOG(V(J)));
{ Sonuç hesaplanıyor }
>>;
RETURN RESULT;
END;
PROCEDURE RINT(F);
```

```
{ Girilen F rasyonel fonksiyonunun sembolik integralini Horowitz-Ostrogradski
yöntemi ile hesaplar }
BEGIN
A:={A0,A1,A2,A3,A4};
{ Q1 = ∑ai x^i polinomundaki ai'ler. i=0,...,5 }
B:={B0,B1,B2,B3,B4};
{ Q2 = ∑bi x^i polinomundaki bi'ler. i=0,...,5 }
ARRAY DQ1(10);
ARRAY Q1(10);
ARRAY Q2(10);
ARRAY C(10);
ARRAY SL(10);
DQ1(10)=0;
Q1(10)=0;
Q2(10)=0;
Q:=NUM(F);
{ Girilen F rasyonel fonksiyonunun payını Q değişkenine aktarır. }
R:=DEN(F);
{ Girilen F rasyonel fonksiyonunun paydasını R değişkenine aktarır. }
YA:=A;
{ A değişken kümesini yedekler. }
YB:=B;
{ B değişken kümesini yedekler. }
DEGREE:=DEG(Q,X);
DR:=DF(R,X);
R1:=GCD(R,DR);
IF R1 NEQ 1 THEN
<<
DR1:=DF(R1,X);
R2:=R/R1;
S:=(DR1*R2)/R1;
M:=DEG(R1,X);
```

```
N:=DEG(R2,X);
FOR I:=M-1 STEP -1 UNTIL 0 DO
<<
Q1(I):=FIRST(A)*X^I;
{ Q1( i ) = a_i x^i  aktarımı }
Q1(10):=Q1(10)+Q1(I);
{ Q_1 = Σ a_i x^i yığmalı toplaması }
DQ1(I):=DF(Q1(I),X);
{ DQ( i ) = (a_i x^i)'  aktarımı }
DQ1(10):=DQ1(10)+DQ1(I);
{ Q'_1 = Σ (a_i x^i)' yığmalı toplaması }
A:=REST(A);
{ A kümesinin ilk elemanını siler }
>>;
FOR J:=N-1 STEP -1 UNTIL 0 DO
{ A kümesi için yapılan aktarımların aynısı B kümesi için tekrarlanıyor }
<<
Q2(J):=FIRST(B)*X^J;
Q2(10):=Q2(10)+Q2(J);
B:=REST(B);
>>;
SON:=DQ1(10)*R2-Q1(10)*S+Q2(10)*R1;
{ Q = (Q'_1 * R_2) - (Q_1 * S) + (Q_2 * R_1) aktarımı }
SS:={};
LQ:=COEFF(Q,X);
{ Q polinomunun katsayılarını LQ listesine aktarır }
LSON:=COEFF(SON,X);
{ SON polinomunun katsayılarını LSON listesine aktarır }
LLQ:=LENGTH(LQ);
LLSON:=LENGTH(LSON);
IF LLQ<LLSON THEN
<<
```

```
W:=LLSON-LLQ;
{ Kaç işlem yapılması gerektiğini hesaplar ve W değişkenine aktarır }
FOR I:=1:W DO LQ:=APPEND(LQ,{0});
{ W kere LQ listesine 0 eklenir. Bu işlemin amacı LQ ve LSON listesinin eleman
sayılarını aynı yapmaktır. }
DEGREE:=DEGREE+W;
>>;
FOR K:=DEGREE STEP -1 UNTIL 0 DO
<<
SS:=APPEND(SS,{0=FIRST(LSON)-FIRST(LQ)});
{ SS listesine Q= (Q'1 * R2) - (Q1 * S) + (Q2 * R1) eşitliğinden oluşan katsayılar
aktarılır }
LQ:=REST(LQ);
LSON:=REST(LSON);
>>;
AA:={};
FOR L:=0:M-1 DO
<<
AA:=CONS(FIRST(YA),AA);
YA:=REST(YA);
>>;
BB:={};
FOR H:=0:N-1 DO
<<
BB:=CONS(FIRST(YB),BB);
YB:=REST(YB);
>>;
AB:=APPEND(AA,BB);
{ Değişkenler listesi birleştiriliyor }
SSS:=SOLVE(SS,AB);
{ SSS listesindeki eşitlikler AB listesindeki değişkenlere göre çözülüyor }
SSS:=FIRST(SSS);
```

```
YSSS:=SSS;
FOR U:=DEGREE STEP -1 UNTIL 0 DO
<<
Q1(10):=SUB(FIRST(SSS),Q1(10));
SSS:=REST(SSS);  { Q1 oluşturuluyor }
>>;
FOR V:=DEGREE STEP -1 UNTIL 0 DO
<<
Q2(10):=SUB(FIRST(YSSS),Q2(10));
YSSS:=REST(YSSS);  { Q2 oluşturuluyor }
>>;
POLY:=Q2(10)/R2;
{ Q2 / R2 }
POLY1:=NUM(POLY);
POLY2:=DEN(POLY);
LP(POLY1,POLY2);
{ Q2 / R2 rasyonel fonksiyonu logaritmik kısmın çözümü için LP alt programına
gönderiliyor }
WRITE "[",Q1(10)/R1,"]+[",RESULT,"]";
>>
ELSE
<<
LP(Q,R);
WRITE RESULT;
>>
END;
```

ÖZGEÇMİŞ

Adı Soyadı : .....

Doğum Yeri : .....

Doğum Yılı : .....

Medeni Hali : .....

Eğitim ve Akademik Durumu :

Lise 19... / 19 ... .

Lisans 19... / 19... .....

Yabancı Dil : .....

İş Tecrübesi :

19... / 19 ... .

19... / 19 ... .