Hesaplanabilirlik Kuramı: Anahtar Kavramlar

Uğraşmak istediğimiz genel problem şu: verili bir küme ya da bağıntı için, üyelik denetimi sağlayan bir algoritma hangi durumda vardır? Bu problemi başlangıçta çok daha sınırlı görünen başka bir probleme indirgeyebiliriz; şöyle ki, verili bir *doğal sayılar* kümesi için, üyelik denetimi sağlayan bir algoritma hangi durumlarda vardır? Verilen problemi sayılara ilişkin bir problem olarak *kodlayarak* bu indirgemeyi gerçekleştirebiliriz. Birkaç örnek bunun nasıl yapıldığını bize gösterecektir.

Örnek: Halihazırda biliyoruz ki bir Önermeler Dizgesi (ÖD) cümlesinin geçerli olup olmadığını denetlemek için bir algoritma vardır. Bu problemin sayılara ilişkin bir problem olarak nasıl kodlanabileceğini görelim. Öncelikle, aşağıda gösterildiği gibi, ÖD dilimizin her bir yalın sembolüyle bir rakam kodunu eşleştiririz:

1, **"("** in kodudur

2, **")"** in kodudur

3, **"∨"** nın kodudur

4, **"∧"** nin kodudur

5, **"→"** nin kodudur

6, **"↔"** ın kodudur

7, **"¬"** in kodudur

8, **"A"** nın kodudur

9, **"B"** nin kodudur

10, **"C"** nin kodudur

11, **"D"** nin kodudur

Bu böyle devam eder. ÖD dilinin bir cümlesini sonlu bir semboller dizisi olarak düşünebiliriz, böylelikle bir cümleyi sonlu bir sayılar dizisi olarak kodlayabiliriz. Hatta, az sonra göreceğimiz gibi, sonlu bir doğal sayılar dizisini tek bir doğal sayı olarak kodlamak mümkündür. Bu iki kodlamayı birleştirerek her bir cümleyle bir kod sayısını eşleştiririz. Kodlamadan bahsederken aklımda olan şu: girdi olarak bir ÖD cümlesi verildiğinde çıktı olarak bu cümlenin kod sayısını veren bir algoritma vardır. Ayrıca girdi olarak bir sayı verildiğinde ilk önce bu sayının bir ÖD cümlesinin kodu olup olmadığını saptayan, ve öyleyse, size ilgili cümleyi veren bir algoritma da vardır. Böylesi bir kodlamayı elde eder etmez görürüz ki verilen bir ÖD cümlesinin geçerli olup olmadığını saptama problemi, verilen bir doğal sayının geçerli bir ÖD cümlesinin kodu olup olmadığını denetleme problemine indirgenmiş olur.

Bir doğal sayılar ikilisini kodlama. x ve y doğal sayılar için ikili şöyle olsun: $(x,y) = ½(x^2 + 2xy + y^2 + x + 3y)$. *İkili*, sıralı doğal sayı ikililerinin kümesinden doğal sayılar kümesine giden bir tameşleme fonksiyonudur[1]. Verili bir x ve y için, size ikili(x,y)'yi veren bir algoritma ve verili bir z için ikili(x,y) = z üzerinden size benzersiz x ve y sayılarını veren bir başka algoritma vardır. x = 1inci(z) ve y = 2nci(z) yazalım.

Sonlu bir doğal sayılar kümesini kodlama. F sonlu bir doğal sayılar kümesi olduğunda, $Kod(F) = \sum \{2^n: n \in F\}$ olsun. Kod, sonlu doğal sayı kümelerinden doğal sayılar kümesine giden birebir ve örten bir fonksiyondur. Verili bir F için Kod(F)'yi hesaplayacak mekanik bir

[1] A kümesinden B kümesine giden bir *fonksiyon*un, şunu karşılayan bir f ⊆ A □ B kümesi olduğunu hatırlayın: A'nın her bir a öğesi için, <a,b> ∈ f 'i sağlayacak B'nin bir ve yalnız bir b öğesi vardır. <a,b> ∈ f ise f(a) = b yazarız. A, fonksiyonun *önalanı*, ve B'nin a ∈ A'lar için f(a) = b eşitliğini sağlayan bütün b öğelerinin kümesi de fonksiyonun ardalanıdır. f'nin ardalanı B'nin tamamı olduğunda f'nin *örten* olduğu söylenir. A'daki her bir a ve a* için, f(a) = f(a*) olduğunda a = a* oluyorsa, f'nin *birebir* olduğu söylenir. f hem örten hem de birebir ise f'nin *tameşleme* olduğu söylenir. f, A'dan B'ye giden bir tameşlemeyse f'nin *tersi* olan f^{-1}, yani {<b,a>: <a,b> ∈ f}, B'den A'ya giden bir tameşlemedir..

yöntem vardır. Ayrıca verili bir n için, Kod(F) = n üzerinden benzersiz F kümesini hesaplayan karşı yönde bir algoritma da vardır: n ikili gösterimde yazılır ve F, 1'lerin ortaya çıktığı yerlerin kümesi olarak alınır.

Sonlu bir diziyi kodlama. $<s_0, s_1, s_2,..., s_n>$ sonlu dizi olduğunda, dizinin kodu Kod({ikili(0,s_0), ikili(1,s_1), ikili(2,s_2),..., ikili(n,s_n)}) olsun.

Örnek: Biri İngilizce'nin bütün ifadelerini alfabetik sıraya göre listeleyen ve diğeri Rusça'nın bütün ifadelerini alfabetik sıraya göre listeleyen sonsuz iki liste oluşturabiliriz. İngilizce bir ifade verildiğinde ifadenin listedeki konumunu bulabilmemizi sağlayan mekanik bir yöntem ve ayrıca verili bir n sayısı için listedeki n-inci ifadeyi bulmamızı sağlayan bir başka yöntem vardır. Bu durumda, İngilizce bir ifade ve Rusça bir ifade verildiğinde, ikincisinin birincisinin kabul edilebilir bir çevirisi olup olmadığını belirleme problemi, verili bir m ve n için, ikili(m,n)'nin ikili(i,j)'nin bir öğesi olup olmadığını belirleme şeklindeki aritmetik probleme denk olur: i-inci İngilizce anlatım j-inci Rusça anlatımın kabul edilebilir bir çevirisidir.

Örnek: Yıldız Savaşları nükleer savunma sisteminin esası, saldıran füzelerin radar izlerini girdiler olarak ve saldıran füzeleri ateş edip düşürecek ABD füzelerine verilen talimatları çıktılar olarak kabul eden devasa bir süper-bilgisayar olarak tahayyül edilir. Şimdi gelen bir füzenin izleyebileceği her eğri için müstakil bir rakam kodu atamak mümkün değildir, çünkü bu gibi mümkün eğriler doğal sayılardan fazladır. Bununla birlikte, gelen füzeleri algılaması için kullandığımız cihaz sınırlı duyarlığa sahiptir, o kadar ki birbirine çok yakın eğriler arasında ayrım yapamayacaktır. Cihazların birbirinden ayırdedemediği eğrileri gruplarsak, yalnızca sonlu sayıda ayırdedilebilirlik sınıfı olduğunu görürüz; işte bunlara rakam kodları atanabilir. Bilgisayarın savunma füzelerine gönderdiği talimatlara da aynı şekilde rakam kodları verilebilir. Böylece bilgisayarın çözmeye çalıştığı problem, bir rakam problemi olarak kodlanabilir.

Şimdi bazı anahtar tanımlar sunacağız. Bu tanımlarda, matematik kesinlik kazandırılmamış, konuşma diline ait kavramlar geçmekte, ama bunlar yine de bazı teoremleri ispatlamamıza izin verecek kadar kesin olacaklar:

Tanımlar. Doğal sayılardan doğal sayılara giden bir *kısmi fonksiyon*, $\mathbb{N} \times \mathbb{N}$'nin aşağıdaki koşulu sağlayan f gibi bir altkümesidir:

$$(\forall x)(\forall y)(\forall z)((\langle x,z\rangle \in f \wedge \langle y,z\rangle \in f) \rightarrow x = y)$$

Başka bir deyişle f, □'nin bir altkümesinden □'nin bir altkümesine giden bir fonksiyondur. Bir f kısmi fonksiyonu, tanım kümesi f'nin tamamıysa ve ancak böyleyse, *tam*dır. Dolayısıyla bizim kullanımımızda tam fonksiyonlar bir tür kısmi fonksiyonlardır.

Bir f kısmi fonksiyonu, şöyle bir algoritma varsa, ve ancak o varsa, *hesaplanabilir*dir: girdi olarak n verildiğinde n, f'nin tanım kümesindeyse algoritma çıktı olarak f(n)'yi verir; n, f'nin tanım kümesinde değilse algoritma hiçbir çıktı vermez.

S için bir *karar yöntemi*, S'nin hususi fonksiyonunu hesaplayan bir algoritmadır; yani girdi S'nin üyesi ise çıktı 1, ama girdi S'nin üyesi değilse çıktı 0'dır. S için bir karar yöntemi varsa, ve ancak bu varsa, S *karar verilebilir*dir.

S için bir *ispat yöntemi*, S'yi tanım kümesinde içerecek ve bir girdiye 1 değerini, girdi S'nin bir öğesiyse ve ancak böyleyse atayacak bir kısmi fonksiyonu hesaplayan bir algoritmadır.

S için bir *sayma yöntemi,* S'nin öğelerini listeleyen bir algoritmadır. Bir n sayısını listedeki n-inci öğeye götüren fonksiyon, tanım kümesi doğal sayıların

bir başlangıç dilimi olan ve ardalanı S olan hesaplanabilir bir kısmi fonksiyondur.

S için bir sayma yöntemi varsa, ve ancak bu varsa, S *etkin olarak sayılabilir*dir.

Teorem. S etkin olarak sayılabilir ise, ve ancak böyleyse, S için bir ispat yöntemi vardır.

İspat: (⇒) S için bir ispat yöntemi, Dom(f) ⊇ S ve f(n) = 1 olmak üzere, bir f kısmi fonksiyonunu, $n \in \mathrm{Dom}(f)$ için, $n \in S$ ise, ve ancak böyleyse, hesaplar. f için bir sayma yöntemi bulmak istiyoruz. İşte işlemeyen bir öneri:

İlk olarak f(0)'ı hesapla. f(0) = 1 ise 0'ı listeye koy.

İkinci olarak f(1)'i hesapla. f(1) = 1 ise 1'i listeye koy.

Üçüncü olarak f(2)'yi hesapla. f(2) = 1 ise 2'yi listeye koy.

Dördüncü olarak f(3)'ü hesapla. f(3) = 1 ise 3'ü listeye koy.

Ve benzerleri.

Sorun şudur ki algoritma, f'nin önalanında olmayan bir şeye rastladığında çıkmaza girer. Algoritmayı, önalanda olmayan bir sayı ile karşılaştığında onu daha büyük sayıları hesaba katmaya devam etmekten alıkoymayacak şekilde düzeltmek istiyoruz. Şunu deneyebiliriz:

İlk olarak f(0)'ı hesaplamayı dene. 0 ∉ Dom(f) ise adım 2'ye geç. 0 ∈ Dom(f) ve f(0) = 1 ise 0'ı listeye koy.

İkinci olarak f(1)'i hesaplamayı dene. 1 ∉ Dom(f) ise adım 3'e geç. 1 ∈ Dom(f) ve f(1) = 1 ise 1'i listeye koy.

Üçüncü olarak f(2)'yi hesaplamayı dene. 2 ∉ Dom(f) ise adım 4'e geç. 2 ∈ Dom(f) ve f(2) = 1 ise 2'yi listeye koy.

Dördüncü olarak f(3)'ü hesaplamayı dene. $3 \notin \text{Dom}(f)$ ise adım 5'e geç. $3 \in \text{Dom}(f)$ ve f(3) = 1 ise 3'ü listeye koy.

Ve benzerleri.

Sorun şu ki genellikle bir sayının f'nin önalanında olup olmadığını bize söyleyecek bir denetlemeye sahip olmayacağız. n, f'nin tanım kümesindeyse f için olan algoritmamız f(n)'yi hesaplayacaktır; ama n tanım kümesinde değilse algoritma, bir çıktı vermeksizin beklenir şekilde sonsuza kadar çalışmaya devam edecektir. Yöntemimizi, f(n)'yi hesaplamaya çalışmaktan asla vazgeçmediğimiz ama bunu yapmanın bizi n'den daha büyük sayıları hesaba katmaktan alıkoymayacağı bir şekilde düzenlemek istiyoruz. Bunu farklı hesaplamaları birbirine bağlayan *eksiksizce uyma* denilen bir teknikle yerine getiririz:

Adım 1. f(0)'ı hesaplama denemesinde bir adım at. Başarırsan f(0) = 1 olup olmadığına bak. Öyleyse 0'ı listeye koy.

Adım 2. f(1)'i hesaplama denemesinde bir adım at. Başarırsan f(1) = 1 olup olmadığına bak. Öyleyse 1'i listeye koy.

Adım 3. f(0)'ı hesaplama denemesinde bir başka adım at (f(0)'ı adım 1'de hesaplamayı zaten başarmadıysan). f(0)' hesaplamada başarılı olursan onun 1'e eşit olup olmadığına bak. Öyleyse 0'ı listeye koy.

Adım 4. f(2)'yi hesaplama denemesinde bir adım uygula. Başarırsan ve f(2) = 1 ise 2'yi listeye koy.

Adım 5. f(1)'i hesaplama denemesinde bir başka adım at (f(1)'i adım 2'de hesaplamayı zaten başarmadıysan). f(1)'i hesaplamada başarılı olursan ve o 1'e eşitse 1'i listeye koy.

Adım 6. f(0)'ı hesaplama denemesinde bir başka adım at (f(0)'ı adım 1 veya adım 3'te zaten hesaplamadıysan). f(0)'ı hesaplamada başarılı olursan ve o 1'e eşitse 0'ı listeye koy.

Adım 7. f(3)'ü hesaplama denemesinde bir adım uygula. Başarırsan ve f(3) = 1 ise 3'ü listeye koy.

Adım 8. f(2)'yi hesaplama denemesinde bir başka adım uygula (f(2)'yi adım 4'te hesaplamayı zaten başarmadıysan). f(2)'yi hesaplamada başarılı olursan ve o 1'e eşitse 2'yi listeye koy.

Adım 9. f(1)'i hesaplama denemesinde bir başka adım uygula (f(1)'i adım 2 veya adım 5'te hesaplamayı zaten başarmadıysan). f(1)'i hesaplamada başarılı olursan ve o 1'e eşitse 1'i listeye koy.

Adım 10. f(0)'ı hesaplama denemesinde bir başka adım uygula (f(0)'ı hesaplamada başarılı olmamışsan). f(0)'ı hesaplamada başarılı olursan ve o 1'e eşitse 0'ı listeye koy.

Adım 11. f(4)'ü hesaplama denemesinde bir adım uygula. Başarılı olursan ve f(4) = 1 ise 4'ü listeye koy.

Adım 12. f(3)'ü hesaplama denemesinde bir başka adım uygula (onu hesaplamada zaten başarılı olmadıysan). f(3)'ü hesaplamada başarılı olursan ve o 1'e eşitse 3'ü listeye koy.

Adım 13. f(2)'yi hesaplama denemesinde bir başka adım uygula (onu başarılı bir şekilde zaten hesaplamamışsan). f(2)'yi hesaplamada başarılı olursan ve o 1'e eşitse 2'yi listeye koy.

Adım 14. f(1)'i hesaplama denemesinde bir başka adım uygula (onu henüz hesaplamamışsan). f(1)'i hesaplamada başarılı olursan ve o 1'e eşitse 1'i listeye koy.

Adım 15. f(0)'ı hesaplama denemesinde bir başka adım uygula (onu henüz hesaplamamışsan). f(0)'ı hesaplamada başarılı olursan ve o 1'e eşitse 0'ı listeye koy.

Ve benzerleri.

(⇐)S için bir sayma yöntemine sahipsek, S için ispat yöntemimiz şu olacak:

n verili olsun. S'yi listelemeye başla. n listede belirirse, her belirdiğinde 1 çıktısını ver.⊠

Teorem. Bir küme, kendisi ve tümleyeni etkin olarak sayılabilir ise, ve ancak böyleyse, karar verilebilirdir.

İspat: (⇒)S karar verilebilir ise S'nin hususi fonksiyon olan χ_s'yi hesaplamak için bir algoritma vardır. Bu algoritma S için bir ispat yöntemi de olacaktır. n'yi, 1 - χ_s(n)'ye uygulayan algoritma, S'nin tümleyeni için bir ispat yöntemi olacaktır.

(⇐) n verili olsun. Hem S'yi hem de tümleyenini aynı anda listelemeye başla. S'nin listesinde n varsa 1 çıktısını ver. Tümleyenin listesinde n varsa 0 çıktısını ver.⊠

Karar verilebilir ve etkin olarak sayılabilir kümelerden bahsettik, ama karar verilebilir ve etkin olarak sayılabilir bağıntılardan da bahsedebiliriz; ve aynı teoremler aynı ispatlarla geçerli olacaktır. Benzer şekilde, tek yerli fonksiyonlar için "hesaplanabilir kısmi fonksiyon"u tanımladık, ama birden fazla işleniği olan fonksiyonlardan da bahsedebiliriz. Elimizde şu var:

Teorem. Tek işlenikli kısmi bir fonksiyon, bir ikili bağıntı olarak alındığında etkin olarak sayılabilir ise, ve ancak böyleyse, hesaplanabilirdir.

İspat: (⇒) Varsayalım ki f hesaplanabilir bir kısmi fonksiyondur. İşte bir ikili bağıntı olarak düşünüldüğünde f için bir ispat yöntemi: m ve n verildiğinde f(m)'yi hesaplamayı dene. Bir çıktı elde edersen onun n'ye eşit olup olmadığını denetle. Eşitse çıktıya 1 ver.

(⇐) f için bir sayma yöntemi verildiğinde f'yi hesaplamak için bir algoritma: m verildiğinde f'yi saymaya başla. Listede bir ikili belirdiğinde ilk bileşeninin m'ye eşit olup olmadığını kontrol et. Eşitse ikinci bileşeni çıktı olarak ver. ⊠

> **Teorem.** Tek işlenikli bir tam fonksiyon, bir ikili bağıntı olarak alındığında karar verilebilir ise ve ancak böyleyse, hesaplanabilirdir.

İspat: (⇒) f hesaplanabilir bir tam fonksiyon olarak verildiğinde işte bir karar yöntemi: m ve n verildiğinde f(m)'yi hesaplamaya başla. f(m) n'ye eşitse çıktıya 1 ver. f(m) n'den farklı ise çıktıya 0 ver.

(⇐) Her karar verilebilir tam fonksiyon etkin olarak sayılabilir bir kısmı fonksiyon olacaktır ve de bu nedenle hesaplanabilir olacaktır.⊠

> **Teorem.** Etkin olarak sayılabilir iki kümenin birleşimi de etkin olarak sayılabilirdir.

İspat: A ve B için sayma yöntemleri verildiğinde işte A ∪ B için bir ispat yöntemi: n verildiğinde A'yı ve B'yi aynı anda listelemeye başla. n iki listeden herhangi birinde belirirse 1 çıktısını ver.⊠

> **Teorem.** Etkin olarak sayılabilir iki kümenin kesişimi de etkin olarak sayılabilirdir.

İspat: A ve B için sayma yöntemleri verildiğinde işte kesişim için bir ispat yöntemi: n verildiğinde A'yı saymaya başla. n listede belirirse A için endişelenmeyi bırak ve B'yi listelemeye başla. n belirirse 1 çıktısını ver.⊠

Teorem. Bir küme, hesaplanabilir bir kısmi fonksiyonun tanım kümesi ise ve ancak böyleyse, etkin olarak sayılabilirdir.

İspat: (⇒)A etkin olarak sayılabilir ise A için bir ispat yöntemi vardır. Tanım gereği bu şu demektir: hesaplanabilir öyle bir f kısmi fonksiyonu vardır ki herhangi bir n için, n, f'nin önalanında ve f(n) = 1 ise, ve ancak böyleyse, n, A'ya aittir. Aşağıdaki algoritmayla bir g hesaplanabilir fonksiyonu tanımlanır:

f(n)'yi hesaplamaya başla. Bir değer elde edersen onun 1'e eşit olup olmadığını denetle. Öyleyse 1 çıktısını ver.

Bu durumda A, g'nin önalanıdır.

(⇐) f hesaplanabilir bir tam fonksiyon ise f'nin tanım kümesi için bir ispat yöntemi aşağıdaki gibidir:

f(n)'yi hesaplamaya başla. Bir çıktı elde edersen 1 çıktısını ver.

Teorem. Bir küme, hesaplanabilir bir kısmi fonksiyonun değer kümesi ise, ve ancak böyleyse, etkin olarak sayılabilirdir.

İspat: (⇒) A etkin olarak sayılabilir ise, o, n'yi listedeki n-inci sayıya götüren kısmi fonksiyonun ardalanıdır.

(⇐) f hesaplanabilir ise bir ikili bağıntı olarak kabul edildiğinde etkin olarak sayılabilirdir. f'nin ardalanını listeleyecek bir algoritma aşağıdaki gibidir:

f'yi listele. Her ne zaman bir sıralı ikili belirirse, onun ikinci üyesini çıktı olarak ver.⊠

Teorem. Bir küme, ya bir boş küme ya da hesaplanabilir bir tam fonksiyonun ardalanı ise, ve ancak böyleyse, etkin olarak sayılabilirdir.

İspat: (⇒) Varsayalım ki A etkin olarak sayılabilirdir ve boş değildir. A sonsuzsa n'yi listedeki n-inci öğeye götüren fonksiyon, ardalanı A olan, hesaplanabilir bir tam fonksiyondur. A sonluysa, A = $\{a_0, a_1, a_2,...,a_k\}$ biçimindedir. Bu durumda A, aşağıdaki gibi tanımlanan f fonksiyonunun ardalanıdır:

i = 0 ise f(i) = a_0.

i = 1 ise f(i) = a_1.

i = 2 ise f(i) = a_2.

.....

i = k ise f(i) = a_k.

i > k ise f(i) = a_k.

(⇐) A boş kümeyse, asla çıktı vermeyen tembel algoritmayla sayılır. A hesaplanabilir bir tam fonksiyonun ardalanıysa hesaplanabilir bir kısmi fonksiyonun da ardalanıdır.⊠

Teorem. Bir küme, ya sonlu ise ya da birebir, hesaplanabilir bir tam fonksiyonun değer kümesiyse, ve ancak böyleyse, etkin olarak sayılabilirdir.

İspat: (⇒) A sonsuzsa ve etkin olarak sayılabilir ise yinelemeler olmaksızın listelenebilir. Listeleme yöntemi, "bir sayı listeye sadece henüz listelenmemişse eklenebilsin" şeklinde değiştiriverilir. Bu durumda n'yi listedeki n-inci öğeye götüren fonksiyon, ardalanı f olan birebir, hesaplanabilir bir tam fonksiyondur.

(⇐) A sonluysa, etkin olarak sayılabilirdir. A birebir, hesaplanabilir bir tam fonksiyonun ardalanıysa, hesaplanabilir bir kısmi fonksiyonun da ardalanıdır.⊠

Teorem. A ve B etkin olarak sayılabilir kümelerse aşağıdaki özellikleri sağlayan, etkin olarak sayılabilir C ve D kümeleri vardır:

$C \subseteq A$

$D \subseteq B$

$C \cap D = \square$

$C \cup D = A \cup B$

İspat: İşte C ve D için sayma yöntemleri: A ve B'yi aynı anda listelemeye başla. Bir n sayısı, B'nin listesinde henüz belirmemiş olduğu bir aşamada A'nın listesinde belirirse n'yi C'nin listesine koy. n, A'nın listesinde henüz belirmemiş olduğu bir aşamada B'nin listesinde belirirse n'yi D'nin listesine koy. n, aynı aşamada her iki listede birden belirirse n'yi A'nın listesine koy.⊠

Teorem. R, $(\forall x)(\exists y)R(x,y)$ biçiminde etkin olarak sayılabilir bir bağıntı ise her x için $R(xf(x))$'i sağlayan f gibi bir hesaplanabilir tam fonksiyonu vardır.

İspat: İşte f'yi hesaplamak için bir algoritma: n verildiğinde R'ye ait sıralı ikilileri, ilk bileşeni n'ye eşit olan bir tanesine gelinceye kadar saymaya başla. Böyle bir ikili ile ilk karşılaştığın anda ikinci bileşenini çıktı olarak ver.⊠

F, (diyelim ki tek işlenikli) bir hesaplanabilir kısmi fonksiyonsa, f'yi hesaplayan bir bilgisayar programı vardır. Yani bir n sayısı girdi olarak verildiğinde bir süre için hesaplayacak, sonra f(n) çıktısını verip n, f'nin önalanında olduğu takdirde duracak olan bir program vardır. n, f'nin önalanında yer almıyorsa program herhangi bir çıktı vermeksizin sonsuza kadar çalışmaya devam edecektir. (Buna biraz ileride daha detaylı şekilde bakacağız.) Her bir f hesaplanabilir kısmi fonksiyonu için m-inci makinenin f'yi hesapladığı bir m sayısı olsun diye mümkün bütün programı alfabetik sıraya (ya da buna benzer bir şeye) göre düzenleyerek yazabiliriz. m ve n verildiğinde m-inci programı yazabilir, sonrasındaysa programın varsa n çıktısı üzerine hangi çıktıyı verdiğini hesaplayabiliriz. Durma problemi şudur: m ve n verildiğinde m-inci programın n girdisi verildiğinde durup durmadığına karar

vermek. Durma problemi için sadece hesaplamayı devam ettirmeye bağlı bulunan bir ispat yöntemi vardır. Ancak herhangi bir karar yöntemi yoktur.

Teorem. Durma problemi için herhangi bir karar yöntemi yoktur.

İspat: Böyle bir karar yöntemi olsaydı aşağıdaki yöntem hesaplanabilir bir tam fonksiyonu − buna f diyelim − hesaplayacaktı:

> m verili olsun. m-inci makine m girdisi üzerine k çıktısını sağlıyorsa, k+1 çıktısını ver. m-inci makine m girdisi üzerine durmazsa 0 çıktısını ver.

f hesaplanabilir olduğu için f'yi hesaplayan bir makine vardır; j-inci makine diyelim. f tam olduğu için j-inci makine her girdi üzerine bir çıktı sağlar. Daha belirli olarak, j-inci makine, j girdisi üzerine bir çıktı sağlar ve şunu elde ederiz:

> 1 + j-inci makinenin j girdisi üzerine çıktısı
>
> = f(j) [f'nin tanımlanma yolu bakımından]
>
> = j-inci makinenin j girdisi üzerine çıktısı [j'nin seçilme yolu bakımından]

Çelişki.⊠

> **Teorem.** Ayrık, etkin olarak sayılabilir öyle A ve B kümeleri vardır ki A'yı kapsayan ve B'den ayrık olan herhangi bir karar verilebilir küme yoktur.

İspat: A = {m: m-inci makine m girdisine 0 çıktısını verir} olsun. B = {m: m-inci makine

m girdisine 1 çıktısını verir} olsun. Bu durumda A ve B, ayrık ve etkin olarak

sayılabilir olurlar. A'yı kapsayan ve B'den ayrık olan karar verilebilir bir C kümesi varmış gibi davranalım. C karar verilebilir olduğundan, hususi fonksiyonu da hesaplanabilirdir. k-ıncı makinenin C'nin tümleyeninin hususi fonksiyonunu hesapladığını söyleyelim.

K, C'ye aitse $\chi_c(k) = 1$'dır; ve de bu nedenle k-ıncı makine, k girdisine 1 çıktısını sağlar, ki bu da k'nnın B'ye ait olduğu anlamına gelir. Ama bu imkansızdır, çünkü B, C'den ayrıktır.

Dolayısıyla k, C'ye ait değildir ve $\chi_c(k) = 0$'dır; yani k-ıncı makine, k girdisine 0 çıktısını verir. Ama bu, k'nnın C'nin bir altkümesi olan A'ya ait olduğu anlamına gelir. Çelişki.⊠

> **Teorem.** Bir hesaplanabilir tam fonksiyona genişletilemeyen bir hesaplanabilir kısmi fonksiyon vardır.

İspat: En son teoremden A ve B'yi kullanarak g gibi şöyle bir hesaplanabilir kısmi fonksiyon tanımla:

$g(n) = 1$, $n \in A$ ise

$= 0$, $n \in B$ ise

Reductio ad absurdum için, g'yi genişleten h gibi bir hesaplanabilir tam fonksiyon bulunduğunu kabul edelim. Bu durumda bir n girdisini h(n)'nin en yüksek değerine götüren fonksiyon ve 1, A'yı kapsayan ve B'den ayrık olan bir karar verilebilir kümenin karakteristik fonksiyonu olacaktı.

Şunun hatırlanması önemlidir: etkin olarak sayılabilirlik ve karar verilebilirlik *kümeler*in özellikleridir. Bir kümenin karar verilebilir olup olmadığı, kümenin nasıl adlandırıldığına bağlı değildir; aynı şekilde bu, bilgi durumumuza da bağlı değildir. Bir küme, sıklıkla, birçok farklı şekilde adlandırılabilir. S, {n: n, P özelliğine sahiptir} olabilir ve o, {n: n, Q özelliğine sahiptir} de olabilir; ve (boşluğun bir Arap rakamıyla doldurulduğu[2]) " , P

[2] S'nin karar verilebilir olması için "Anza gölündeki balıkların sayısı P özelliğine sahip midir?" gibi soruları cevaplayabilmek zorunda değiliz.

özelliğine sahip midir?" biçimindeki tüm soruları cevaplamak için bir algoritmaya sahip olduğumuz ama " , Q özelliğine sahip midir?" biçimindeki tüm soruları cevaplamak için bir algoritmaya sahip olmadığımız ortaya çıkabilir. Böyle bir durumda S, karar verilebilir sayılacaktır. Bir S kümesi, şöyle P özelliği[3] bulunuyor ise, ve ancak böyleyse, karar verilebilirdir: S, P özelliğine sahip sayıların kümesidir ve " , P özelliğine sahip midir?" biçimindeki soruları cevaplamak için bir algoritma vardır. S'nin, Q özelliğine sahip sayıların kümesi olması ve " , Q özelliğine sahip midir?" biçimindeki soruları cevaplamak için bilinen herhangi bir algoritma bulunmaması koşullarını sağlayan bir başka Q özelliğinin var olduğu gerçeği, S'nin karar verilebilirliğini bozmaz.

Örneğin D = {n sayıları: π'nin ondalık açılımında n veya daha fazla sayıda ardışık 7'ler dizisi vardır} olsun. D, açıkça etkin olarak sayılabilirdir. Sayma yöntemi, basitçe, π'nin ondalık açılımını yazmaya başlamak ve n tane 7'ler dizisiyle karşılaştığınızda n'yi listeye eklemektir. D bir de karar verilebilir midir? Hiç kimse, " , D'ye ait midir?" biçimindeki soruları nasıl cevaplayacağını bilmez. Hiç kimse, 1000'nin D'ye ait olup olmadığını ya da 1,000,000'un D'ye ait olup olmadığını bilmez. Bilindiği kadarıyla her sayı D'ye ait olabilir. Yine de D karar verilebilirdir. Her sayının D'ye ait olduğu durumda D'nin karar yöntemi aşağıdaki gibidir:

Girdi her ne olursa olsun 1 çıktısını ver.

Her sayı D'ye ait değilse, D = {n: n ≤ k}'yi sağlayan bir k sayısı vardır. Bu durumda D için bir karar yöntemi şudur:

---

[3] Burada özellik kavramını kullanışım bir tür gevezelikten ibarettir: Gezginin, atlık özelliğine sahip olduğunu söylemek, Gezginin bir at olduğunu söylemenin yalnızca bir başka yoludur. Aynı düşünceyi, metafizik içinde kördüğüm olmamak kaydıyla, yüklemler hakkında konuşarak da ifade edebiliriz. S, şu koşulu sağlayan bir ϕ yüklemi bulunuyor ise, ve ancak böyleyse, karar verilebilirdir: S = {n: ϕ(n)}, ve ϕ(x) açık cümlesinden, "x"in her bir bağsız geçişlerini bir rakamla yer değiştirerek elde edilen cümlelerin doğruluk değerlerini belirlemek için bir algoritma vardır. Kendisi için böylesi bir algoritmanın olmadığı S = {n: ψ(n)}'yi sağlayan bir başka ψ yükleminin olup olmadığı önemli değildir.

Girdi $\leq$ k ise 1 çıktısını ver. Girdi $>$ k ise 0 çıktısını ver.

Öyle ya da böyle, D için bir karar yöntemi vardır.

Aynı şey fonksiyonlar için de geçerlidir: bir kısmi fonksiyon bir sıralı ikililer kümesidir ve yine hesaplanabilir olup olmadığı nasıl adlandırıldığına bağlı değildir. Örneğin Süreklilik Varsayımı, kümeler kuramındaki en ünlü ispatlanmamış varsayımdır.[4] Bilinen, sadece hiç kimsenin şimdiye kadar Süreklilik Varsayımı'nı ispatlamayı ya da çürütmeyi beceremediği değil, aynı zamanda varsayımın halihazırda kabul edilen kümeler kuramı aksiyomlarına dayanarak ispatlanamaz ya da çürütülemez de olduğudur. Şimdi aşağıdaki gibi tanımlanan c fonksiyonunu göz önüne alın:

f(n) = n+1, Süreklilik Varsayımı doğruysa

= n, Süreklilik Varsayımı doğru değilse

Halihazırda kabul edilen kümeler kuramı aksiyomlarına dayanarak c fonksiyonunun herhangi bir değerini belirlemek mümkün değildir. Yine de c hesaplanabilirdir. c ya ardıl fonksiyonudur ki hesaplanabilirdir, ya da özdeşlik götürümüdür ki yine hesaplanabilirdir.

Etkin olarak sayılabilir kümelerin kümesinin bazı genel yapı özelliklerini saptamayı başardık, ama henüz etkin olarak sayılabilir kümelerin tam olarak hangileri olduklarını söylemeye girişmedik. Nihayetinde yapacağımız, etkin olarak sayılabilir kümeleri, aritmetik dilinin bilhassa yalın olan tamdeyimleriyle adlandırılan kümeler olarak teşhis etmek olacak. Öyleyse şimdi yapmamız gereken aritmetik dilini tanıtmaktır.

[4] Burada yapmaya çalıştığımız şeyi gerçekleştirmek için bu varsayımın ne dediğinin bir önemi yoktur.