Church-Turing Savı

Artık biliyoruz ki her bir ∑ kümesi veya bağıntısı etkin olarak sayılabilirdir. Yineleme teorisinin esas savı aynı zamanda bunun tersinin de geçerli olduğudur ki böylece şunu elde ederiz:

> **Church-Turing Savı.** Bir küme veya bağıntı ∑ ise, ve ancak böyleyse, etkin olarak sayılabilirdir.

Bir kümenin, kendisi ve tümleyeni etkin olarak sayılabilir olduğunda ve ancak böyle olduğunda karar verilebilir olduğunu, ve ayrıca bir kümenin, kendisi ve tümleyeni ∑ olduğunda ve ancak böyle olduğunda Δ olduğunu bildiğimiz için, Church-Turing savının şunu gerektirdiğini görürüz: bir küme veya bağıntı, Δ ise ve ancak böyleyse karar verilebilirdir. Ayrıca, bir kısmi fonksiyon ∑ ise ve ancak böyleyse hesaplanabilirdir; ve bir tam fonksiyon Δ ise ve ancak böyleyse, hesaplanabilirdir.

"Hesaplanabilir", burada *ilkece hesaplanabilir* anlamına gelmektedir: öyle bir mekanik yöntem vardır ki bıkmadan ve hatasızca uygulanırsa fonksiyonu hesaplayacaktır. Kavram yaşlanmayı, ölümü veya sınırlı disk alanını hesaba katmaz. Bir hesaplama, evrendeki parçacık sayısından daha çok sayıda bellek birimi kullansa bile buna izin veririz. Teorik amaç, hesaplanması mümkün olanın uç noktada bir dış limitini vermek, sonrasındaysa bu ideale pratikte yakınsama işini, daha uygulamaya dönük olarak düşünen mühendislere bırakmaktır.

Kimse Church-Turing'i kesin olarak ispatlamayı umamaz. Karar verilebilirlikle ve etkin olarak sayılabilirlikle ilgili iddiaları kesin olarak ispatlayabilmemiz için öncelikle bu kavramların matematiksel olarak kesin karakterizasyonlarına sahip olmamız gerekiyor

ve bize bu karakterizasyonları sunması için de söz konusu sava bel bağlıyoruz. Gerçekten de bu sav lehinde epey bir miktarda kanıt vardır. Şimdi bunların bazılarını inceleyelim.

Elimizdeki delillerden en güçlüsü şundan ibarettir: bilinen her sayma yöntemi bir ∑ kümesi oluşturur ve bilinen her karar yöntemi bir Δ kümesi belirler. Bundan daha fazlası vardır. Bilinen her sayma yöntemi *açıkça* ∑ olan bir küme meydana getirir. İşin birkaç inceliğini öğrenir öğrenmez küme meydana getirmenin bir algoritmasını bir kez bilince kümeyi tanımlayan bir formülü yazmak kolay olacaktır. Çıkmaza girdiğimiz tek durum algoritmayı gerçekten bilmediğimiz ya da onu açık bir şekilde bilmediğimiz durumdur. Örneğin sözdizimi düzgün İngilizce cümlelerin kod sayılarını listeleyen bir ∑ formülünü nasıl yazacağımızı bilmeyiz; ama bunun nedeni, kümeyi oluşturan bir algoritmanın olduğunu tahmin etsek bile onun ne olduğunu bilmeyişimizdir.

Church-Turing savı 1930'larda ileri sürüldükten sonra makul sayıda girişim, özellikle Δ olmayan bir karar yöntemi ortaya koyarak savın yanlış olduğunu gösterme programına adandı. Bu girişimlerin hiçbiri başarılı olmadı. Bir karşı örnek elde etmedeki kesin başarısızlık böyle bir karşı örneğin olmadığını oldukça muhtemel kılmaktadır; ve aynı başarısızlık yüzünden, savın bir karşı örneği olacaksa bunun bugün sahip olduğumuz herhangi bir algoritmadan oldukça farklı bir algoritma olması gerektiği aklıselimce kesindir.

Bilinen algoritmaları almak ve onları yeni algoritmalar yaratmak amacıyla kullanmak için belli standart yöntemler vardır; örneğin değişken değiştirimi ve yinelemeli tanımlama. ∑ kısmi fonksiyonlar sınıfının bu türden bilinen tüm teknikler altında kapalı olduğu gerçeği, Church-Turing savı lehindeki bir başka delildir. Savın herhangi bir karşı örneği, bütünüyle özgün birtakım hesaplama yöntemleri içermek zorunda olacaktır.

Bilinen algoritmaları karmaşık şekillerde birleştirmek suretiyle bir karşı örnek elde etme ümidi yoktur.

1930'lar ve 1940'lar boyunca birçok isim, farklı yaklaşımları ve farklı yolları tutarak hesaplanabilirliği anlama problemi üzerinde çalıştı. Hepsi de aynı cevaba ulaştı. Bu yakınlık, ulaştıkları cevabın doğru cevap olduğunu varsaymamız için bize bir gerekçe sunar. Genel anlamda, bir problem üzerinde birbirinden bağımsız olarak çalışan farklı zeki insanların hepsi aynı cevaba ulaşıyorlarsa, bu, ulaşılan cevabın doğru cevap olduğunu düşünmeye bir gerekçe oluşturur. Ve farklı araştırmacılar tarafından uygulanan yöntemler de birbirinden çok farklılaşıyorsa durum bilhassa böyledir, çünkü bu hepsinin aynı hatayı yapmış olma olasılığını azaltır.

Birbirinden epey farklı birkaç yaklaşımla başlayıp aynı kavramla karşılaşıyorsanız, bu, bulmuş olduğunuz kavramın temel ve doğal bir kavram olduğunu varsaymaya gerekçedir; çünkü böylelikle, tek bir bakış açısından kurulan kavramların çoğuna musallat olan keyfilikten kaçınmak muhtemeldir.

∑ kümeleri vasıtasıyla etkin olarak sayılabilir kümelerin belirlenmesine ulaşan bu yöntemlerin bazılarını tanımlayayım. Bu yöntemlerin hepsinin aynı sonucu verdiğini tek tek ispatlamak yoğun emek gerektirdiği ve bir hayli meşakkatli olduğu için burada buna girişmeyeceğim.

Turing aygıtları. Alan Turing, Cambridge'de verdiği lisans bitirme tezinde, yalın esaslarına indirerek söylersek, hesaplayan bir insan failinin, hesaba dayalı bir problemi, bir kurallar sistemi uygulayarak çözerken ne yaptığının bir modelini vermeye çalışmıştır. Fail, sembolleri soldan başlayıp sağ kenar boşluğuna gelene kadar devam edip tekrar soldan, bu sefer bir satır aşağıdan başlayarak zikzaklı bir tarzda yazar. İlk

basitleştirmemiz, satırları keserek ayırmak ve tutkalla birbirlerine yapıştırmaktır ki böylece hepsi tek, çok uzun bir satır boyunca uzanırlar. Bu sayede failimiz, sembolleri çok uzun bir şerit boyunca yan yana yazar.[1] Aslına bakılırsa, şeridin sonsuz olduğunu farz edeceğiz, çünkü bir problemi çözmekte başarısız olmuş olmamızın tek nedeni müsvedde kağıdını bitirmiş olmamızsa, o problemi, ilkece çözülemez olarak addetmek istemiyoruz.

Fail, bazen bir sonraki kısımda ne yazacağına karar verirken önceki yaptıklarına dönüp bakacaktır. Failin önceki çalışması üzerinde bir seferde tek bir sembole baktığını varsaymakta gerçek bir genelleme kaybı yoktur ki böylece failin şeridin üzerinde bir seferde tek bir sembolün üzerinden geçirdiği, bazen sola doğru bir hane ve bazen de sağa doğru bir hane hareket ettirdiği ve bazen de eski bir sembolü silmek veya yeni bir tane yazmak için duraklattığı hareketli bir şerit okuyucuya sahip olduğunu düşünebiliriz. Çalışmasında bir seferde tek bir sembole sahip olmasıyla failin yaptıklarını tam anlamıyla mümkün en basit adımlara ayırmış oluyoruz.

Sonlu, numaralanmış bir ayrıntılı talimatlar listesi, faile hesaplamaya nasıl devam edeceğini anlatır. "İnceliyor olduğun hane boşsa içine 'Q' harfini yaz ve talimat 112'ye git" ve "İnceliyor olduğun hane içine 'W' yazılmış halde ise onu sil ve talimat 13'e ilerle" ve "İnceliyor olduğun hane içine 'B' yazılmış halde ise sağa doğru bir hane ilerle ve talimat 99'a git" gibi şeyler. Girdi sayısı n ise hesaplama, okuyucunun bir n tane "1" dizisinin sol ucunda olduğu, aksi takdirde boş olan bir şeridin üzerinde başlar. Hesaplama

---

1 Bir problemi resimleyerek betimlemek, bütün sembolik gösterimlerimizi tek bir boyutta yapmaya zorlansaydık bizden yakayı kurtaracak olan bir çözüm bulmamıza yardımcı olabilirdi. Öyleyse, yeni problem çözümlerini keşfetmeyi sağlayan yaratıcı süreçleri tanımlamaya çalışıyor olsaydık sembollerimizin bir satırda bulunmasını talep etmek, korkunç çarpıklıklar ortaya çıkaracaktı. Ama bizim amacımız bu değil. Biz, hesaba dayalı algoritmaların bütünüyle mekanik olan uygulamasına bakmak istiyoruz; bu, yaratıcılığa artık gerek olmayan bir problem çözme aşamasıdır.

nihayet bittiğinde okuyucu tam olarak m tane "1"ler dizisinin sol ucundaysa, m çıktıdır. Hesaplama hiçbir zaman bitmezse n, hesaplanan kısmi fonksiyonun tanım kümesinde değildir. Aygıt belli bir girdi üzerinde durursa bu girdinin bir aygıt tarafından *kabul edildiği* söylenir.

Bu tür bir talimatlar sistemini infaz eden mekanik cihaza *Turing aygıtı* denir. Turing böylesi aygıtları insani hesaplamanın bir modeli olarak önerdi. Kuşkusuz, aygıtın yaptığı şey hesaplayıcı insan failinin yaptığı şey değildir; insan failinin yaptığı şeyin bir hayli basitleştirilmiş ve stilize edilmiş bir taklididir. Ama farklılıklar önemsiz ayrıntılardadır, asli hesaplama kapasitelerinde değil. Bir insani bilgisayarın[2] yapabildiği her şey, Turing'in ileri sürdüğü üzere, bir Turing aygıtıyla taklit edilebilir.

Buradaki ayrıntılar epey keyfidir. Sembollerin sayısı, sonlu olduğu sürece az veya çok olabilir. Girdiler ve çıktılar, "1"ler dizileri yerine Arap rakamları olabilir. Silme, karalama işleri için yedek şeritlere izin verebiliriz. *Belirlenimci olmayan* hesaplamalara bile izin verebiliriz. Bunlar, birbiriyle çelişen talimat kümeleriyle programlanmış aygıtlardır, ve böylece aygıt, belli safhalarda hangi talimatı takip edeceğini rasgele seçer.

Ayrıntıları nasıl düzenlersek düzenleyelim sonuç aynıdır. Bir fonksiyon, $\sum$ ise ve ancak böyleyse, bir Turing aygıtı tarafından hesaplanır. Bir küme, $\sum$ ise ve ancak böyleyse, bir Turing aygıtı tarafından kabul edilir.

2

Turing bir "bilgisayar"dan bahsettiğinde bir hesaplayıcı insan failini kastetmişti çünkü 30'lu yılların başlarında yazdığı sırada elektronik bilgisayarlar henüz icat edilmiş değildi. 40'lı yıllarda (yani, Alman donanma şifresini kırmakla meşgul olduğu savaş döneminin ardından) ilk elektronik bilgisayarlardan birini geliştirmek için çalışmaya başladı.

Tescil makineleri. Turing aygıtlarıyla amaçlanan, hesaplama yapan insan faillerinin yaptığı şeyin bir modelini sağlamaktı. Tescil makineleri elektronik bilgisayarların yaptığı şeyin bir modeli olarak tasarlanır. Bir makine sınırsız sayıda[3] bellek yeri veya *tescil* içerir ve bu makine için bir program, bu bellek yerlerinde içerilen sayıları işlemeyi sağlayan basit talimatlardan meydana gelir. Böylelikle belirli bir tescildeki sayıya 1'i ekleyebiliriz; belirli bir tescili 0'a eşit olacak şekilde ayarlayabiliriz; iki farklı tescilin içeriklerini karşılaştırabilir sonrasındaysa iki içeriğin eşit olup olmadığına dayanarak bundan sonra ne yapılacağına karar verebiliriz vs.

Bir tescil makinesinin S cümleler kümesini *kabul ettiği*, şu doğruysa ve ancak böyleyse, söylenir: makine 0 tescilinde n ile ve öteki tüm tescillerde de 0 ile başlatılırsa, n'nin S'ye ait olduğu durumda makine er geç duracak, n'nin S'ye ait olmadığı durumda hesaplama durmadan devam edecektir.

S, ∑ olduğunda ve ancak böyle olduğunda S'yi kabul eden bir tescil makinesi vardır. Bir kez daha, bu sonuç aksaklığa dayanıklıdır, öyle ki, örneğin, belirlenimci olmayan tescil makinelerine izin verirsek yeni hiçbir şey elde etmeyiz.

μ-yinelemeli fonksiyonları. Bir sonraki nitelememiz, hesaplanabilir fonkiyonların aritmetikte hangi yollarla tanımlandığına, daha hususi olarak da yeni hesaplanabilir fonksiyonların eskiller üzerinden hangi yollarla tanımlandığına bakacak. Bu tür iki yöntemi, yani değişken değiştirimi ve yinelemeli tanımlamayı, daha önce ele aldık.

Örneğin, elimizde 0 ve s varsa +, •, ve E'yi yinelemeli bir şekilde tanımlayabiliriz:

3

Unutmamak gerekir ki nitelemeye çalıştığımız *ilkece* hesaplanabilirliktir.

$x + 0 = x$

$x + (y+1) = s(x + y)$

$x \bullet 0 = 0$

$x \bullet (y+1) = (x \bullet y) + x$

$xE0 = 1$

$xE(y+1) = (xEy) \bullet x$

İşte verili hesaplanabilir fonksiyonlardan yenilerini oluşturmak için başka bir yöntem:

> **Tanım.** n-öğeli bir f kısmi fonksiyonu verildiğinde, $\mu x_0[f(x_0,x_1,...,x_n) = 0]$, yandaki gibi tanımlanan n-öğeli kısmi fonksiyondur:$<x_1,...,x_n>$ girdi olarak verildiğinde, $f(y,x_1,...,x_n)$ tanımlanıyor ve 0'a eşit oluyor, her bir $z < y$ için, $f(z,x_1,...,x_n)$ tanımlanıyor ve 0'dan farklı oluyorsa, ve ancak böyleyse, $\mu x_0[f(x_0,x_1,...,x_n) = 0]$ tanımlanacak ve y'ye eşit olacaktır.

Dikkat ederseniz f tam olsa bile $\mu x_0[f(x_0,x_1,...,x_n) = 0]$ tam olmak zorunda değildir çünkü $f(x_0,x_1,...,x_n)$'yi 0'a eşit yapan herhangi bir x_0 değeri olmayabilir.

Açıktır ki f hesaplanabilir ise $\mu x_0[f(x_0,x_1,...,x_n) = 0]$ da hesaplanabilirdir. Sadece, 0 çıktısını elde edinceye kadar ardışık sayıları yerine koyun.

> **Tanım.** *μ-yinelemeli fonksiyonları*, ardıl fonksiyonu, 0 sabit fonksiyonunu (her girdi için 0'ı veren birli fonksiyon), ve izdüşüm fonksiyonlarını ($j \leq n$ olmak kaydıyla her j ve n için, $<x_1,x_2,...,x_n>$'yi x_j'ye götüren bir izdüşüm fonksiyonu vardır; muhasebe işleri için onlara ihtiyaç duyarız) içeren en

küçük tam kısmi fonksiyonlar sınıfını oluştururlar; ve değişken değiştirimi, yinelemeli tanımlama ve μ operatörü altında kapalıdır.

Bir kısmi fonksiyon, ∑ ise, ve ancak böyleyse, μ-yinelemelidir .

Markov algoritmaları. Bir sonraki ∑ kümeleri nitelememiz aritmetik hesaplamalardan ziyade anlamdizim hesaplamalarına dayanır. Sonlu bir alfabeyle başlarız. Bir *kelime*, sonlu bir harfler dizisidir (boş dizi dahil). Bir *üretim sistemi*, kelimelerin sıralı ikililerinin sonlu bir kümesidir. Bir σ kelimesi, <β,γ>'nin üretim sistemi içinde olduğu, ρ ile başlayıp σ ile biten

$$\alpha\hat{}\beta\hat{}\delta \rightarrow \alpha\hat{}\gamma\hat{}\delta$$

biçiminde bir dönüştürümler (transformations) sistemi varsa, ve ancak böyleyse, bir ρ kelimesinden *türetilebilir*dir; burada "^" bitiştirme operatörünü simgeler. Herhangi bir σ kelimesi için, S'den bir boş kelime türetimi olduğunda, ve ancak böyle olduğunda, $\sigma \in S$ oluyorsa, ve ancak böyleyse, S kelimeler kümesi, verili üretim sistemi tarafından *kabul edilir*. Bir küme, onu kabul eden bir üretim süreci varsa, ve ancak böyleyse, ∑'dır.

Bir kuram içinde temsil edilebilirlik. Son nitelememizin üzerine önemle eğildiği fikir şudur: S için bir "ispat yöntemi" varsa, n, S'ye ait olduğunda, n'nin S'ye ait olduğunun ispatlanmasını mümkün kılan bir S tanımı yazılabilmelidir. Doğal sayı sistemini tanımlayan Γ gibi öyle bir kuram ve S'yi tanımlayan σ gibi öyle bir yüklem olmalıdır ki σ([n]), n, S'ye aitse, ve ancak böyleyse, Γ'nin bir sonucudur. Bu duruma bir isim veriyoruz:

Tanım. Herhangi bir n için, σ([n]), Γ'nin bir sonucu olduğunda ve ancak böyle olduğunda, n, S'nin bir elemanı oluyorsa, ve ancak böyleyse, σ formülü, Γ teorisi içinde S'yi *zayıf bir şekilde temsil eder*.

n, S'ye aitse σ([n])'nin bir ispatını sağlayarak n'nin S'ye ait olduğunu ispatlayabilirsiniz. n, S'ye ait değilse n'nin S'ye ait olmadığını ispatlamanın herhangi bir yöntemine zorunlu olarak sahip olmazsınız. Her ne zaman n, S'ye ait değilken n'nin S'ye ait olmadığını ¬σ([n])'yi ispatlamak suretiyle ispatlayabildiğiniz durumlarda σ'nin S'yi *güçlü bir şekilde temsil ettiği* söylenir:

Tanım. Herhangi bir n için, σ([n]), Γ'nin bir sonucu olduğunda ve ancak böyle olduğunda n, S'nin bir elemanı oluyorsa, ve ¬σ([n]), Γ'nin bir sonucu olduğunda ve ancak böyle olduğunda n, S'nin dışında oluyorsa, ve ancak böyleyse, σ formülü, Γ teorisi içinde S'yi *güçlü bir şekilde temsil eder*.

Sizden bu konuda sözüme güvenmenizi istemeyeceğim; bunun yerine aşağıdaki sonucu bilffil ispatlayacağız:

Teorem. S bir doğal sayılar kümesi, Γ da bir aksiyomlar kümesi olmak kaydıyla, Γ içinde S'yi zayıf bir şekilde temsil eden σ gibi bir aritmetik dili formülü olduğunda, ve ancak böyle olduğunda, S, ∑'dir . Γ içinde S'yi güçlü bir şekilde temsil eden bir σ formülü varsa, ve ancak böyleyse, S, Δ'dir.

Ne zaman olağan matematiksel uslamlamayı biçimselleştirmeye kalkışsak, sıklıkla kendimizi sınırlı bir aksiyomlar sistemi içinde değil de sınırlı bir aksiyomlar ve aksiyom *şemaları* sistemi içinde çalışırken buluruz. Örneğin, matematiksel tümevarım ilkesi, aritmetik dili içinde tek bir aksiyom tarafından değil de *tümevarım aksiyom şeması* tarafından temsil edilir:

$$((R(0) \wedge (\forall x)(R(x) \rightarrow R(sx))) \rightarrow (\forall x)R(x))$$

Bu şemadan, "R" yerine bir formül koyup sonrasındaysa ortaya çıkan bağsız değişkenleri bağlamak için başına tümel niceleyiciler ekleyerek elde ettiğimiz sonsuz sayıda cümlenin her biri, bir *tümevarım aksiyomudur*. Tek tek aksiyomların yanı sıra aksiyom şemalarına da izin verdiğimiz zaman önceki teoremimiz desteklenmiş olur:

Teorem. Bir S doğal sayılar kümesi için aşağıdakiler denktir:

S, $\sum$'dir.

S, birtakım sonlu aksiyomlar sistemleri içinde zayıf bir şekilde temsil edilir.

S, birtakım sonlu aksiyomlar ve aksiyom şemaları sistemleri içinde zayıf bir şekilde temsil edilir.

S, birtakım $\sum$ aksiyomlar ve aksiyom şemaları sistemleri içinde zayıf bir şekilde temsil edilir.

Benzer şekilde, şunlar denktir:

S, Δ'dir.

S, birtakım sonlu aksiyomlar sistemleri içinde güçlü bir şekilde temsil edilir.

S, birtakım sonlu aksiyomlar ve aksiyom şemaları sistemleri içinde güçlü bir şekilde temsil edilir.

S, birtakım ∑ aksiyomlar ve aksiyom şemaları sistemleri içinde güçlü bir şekilde temsil edilir.

Matematik, kendi tarihi boyunca bir ispatlar bilimi olmuştur. Matematiksel akıl yürütmede kullanılan aksiyomatik metodun üstünlüğünden dolayı bu teorem Church-Turing Savı için kuvvetli bir kanıt sağlar. S kümesini sayan bir algoritma varsa bu algoritmayı eksiksiz bir biçimde tanımlamak ve sonrasındaysa hesaplamaların hatasız olduğunu matematiksel olarak doğrulamak mümkün olmalıdır. Bu doğrulama, geleneksel matematik havası taşıyacak olursa, sonlu bir aksiyomlar ve aksiyom şemaları sistemi içinde biçimselleştiriledebilir. S, teoremin söylediğine göre, ∑ aksiyom sistemi içinde zayıf bir şekilde temsil edilecektir.