Bilgisayar ve Zekâ

Roger Penrose

KRALIN YENİ USU

I

2. Baskı

KRALIN YENİ USU 1

BİLGİSAYAR ve ZEKÂ

Roger Penrose

Kralın Yeni Usu /Bilgisayar ve Zekâ

The Emperor's New Mind / Concerning Computers, Minds, and The Laws of Physics

Roger Penrose Çeviri: Tekin Dereli
Redaksiyon: Ülker Erkan

ISBN 975-403-080-4
ISBN 975-403-083-9
İlk basımı Ekim 1997'de yapılan *Bilgisayar ve Zekâ* bugüne kadar 22.500 adet basılmıştır.

TÜBİTAK POPÜLER BİLİM KİTAPLARI

Yazar Hakkında

1957 yılında Cambridge Üniversitesi'nde doktorasını tamamlayan Roger Penrose, bir süre İngiltere ve ABD'deki üniversitelerde çeşitli görevler aldı. 1964-1973 yılları arasında Londra'daki Birkbeck College'de Uygulamalı Matematik Profesörü olarak çalıştı. 1969'da Stephen Hawking ile beraber kanıtladıkları bir teoremle, yeterince ağır bir yıldızı oluşturan kütlenin kendi merkezine doğru çökmesi sonucu ortaya çıkan karadeliğin, sıfır hacimli sonsuz madde yoğunluklu bir noktasal uzay-zaman tekilliğine ulaşmasının klasik fizikte kaçınılmaz olduğunu gösterdi. 1973'te Oxford Üniversitesi'nde Rouse Ball Matematik Kürsüsü profesörlüğüne getirilen Penrose, 1966'da Adams Ödülü'nü, 1971'de Dannie Heineman Fizik Ödülü'nü aldı. 1972'de Royal Society üyeliğine seçildi. 1975'te de Kraliyet Astronomi Derneği'nin Eddington Madalyasını Hawking ile paylaştı.

Sunuş

İlk basımı 1990 yılında yapılmış olmasına karşın şimdiden bilim klasikleri arasında sayılan elinizdeki kitabın yazan Roger Penrose, çağımızın önde gelen bilim adamı ve düşünürlerindendir.

Roger Penrose, 1931 yılında İngiltere'nin Colchester kasabasında doğdu. Doktor olan annesi, genetik uzmanı olan babası, birisi ünlü matematikçi, diğeri tanınmış satranç şampiyonu olan kardeşleriyle birlikte tüm aile bireylerinin matematiğe ilgisi ve yeteneği vardı. Bu ortamda yetişen Roger Penrose, üniversiteye başlarken ilgi duyduğu iki alan olan tıp ve matematik arasında tercih yapmak zorunda kalınca matematiği seçti. 1957'de Cambridge Üniversitesi'nde doktorasını tamamladı. Teorik fizik problemlerine ilgi duymaktaydı. Bir süre İngiltere ve ABD'deki üniversitelerde gezdikten sonra 1964-1973 yılları arasında Londra'daki Birkbeck College'de Uygulamalı Matematik Profesörü olarak görev yaptı. Burada bulunduğu sırada, 1969'da, Stephen Hawking ile beraber kanıtladıkları bir teoremle, yeterince ağır bir yıldızı oluşturan kütlenin kendi merkezine doğru çökmesi sonucu ortaya çıkan karadeliğin, sıfır hacimli sonsuz madde yoğunluklu bir noktasal uzay-zaman tekilliğine ulaşmasının klasik fizikte kaçınılmaz olduğunu gösterdiler. Böylece Penrose'a şöhret önemli oranda genel görelilik teorisiyle gelmiş bulunsa da soyut matematikte de büyük başarıları vardır. Daha genç yaşlarında babasıyla beraber ilgilenmiş oldukları, Penrose'un *eğlencelik matematik* dediği bir konudaki buluşu, sonradan cebirsel geometride önemli problemlerden birisi olmuştur. Kendini tekrarlayan hangi düzlemsel şekillerle bir yüzey tam olarak kaplanabilir? Biliyoruz ki eşkenar üçgenlerle, dörtgenlerle, düzgün altıgenlerle periyodik kaplama yapılabilir. Penrose, periyodik olmayan (yani kendini tekrarlamayan) düzlem kaplaması veren binlerce farklı şekil üzerinde yıllarca çalıştıktan sonra, bunlardan bağımsız olanların sayısını önce altıya sonra ikiye indirmeyi başardı. Penrose şekilleri, ünlü **Hollandalı** grafik sanatçısı Escher'e de esin kaynağı olmuştur. Penrose, salt matematiksel merak nedeniyle bulduğu bu şekillerin sonradan

kristalimsi (quasicrystal) denen kimya**sal** maddelerin niteliklerini açıklamakta kullanılmış olmasını, temel bilim araştırmalarının toplumsal yararını kanıtlayan çarpıcı bir örnek olarak göstermektedir. Roger Penrose 1972'de, Royal Society Üyesi seçildikten sonra, 1973'de Oxford Üniversitesinde Rouse Ball Matematik Kürsüsü Profesörlüğüne getirildi. Çok sayıda ödülle onurlandırılan Penrose'a son olarak 1996'da Sir ünvanı verilmiştir.

Kitabın adının Hans Christian Andersen'in ünlü masalından geldiği dikkatinizi çekmiş olmalıdır: Kralın Yeni Giysileri. Masalın ana fikri, etkili ve yetkili bilgelerin dile getirmekten kaçındıkları yalın gerçeği, bir çocuğun saflığıyla dile getirilebilmiş olmasıdır. Dünyanın önde gelen matematik dehaları arasında sayılan, teorik fiziğe katkılarının önemi sorgulanmayan Roger Penrose için çocuk saflığına sahiptir denemeyeceğine göre acaba çok önemli kitabına niçin böyle bir başlık seçmiştir?

Bunun nedenlerini 20. yüzyılda gelişen teknik olanakların bilimsel görüş ve yöntem anlayışımızda yol açtığı yeni yönelimlerde aramalıyız. Özellikle her konuda kavramların sorgulanması, daha önce hiç söz konusu edilmemiş yeni kavramların ortaya çıkması gündemdedir. Gözlem ve deney, varsayım ve kuram üstüne kurgulanmış bilimsel yöntem konusunda yeniden düşünmenin zamanı gelmiştir. Bilimsel yaklaşımın temelinde yer alan doğa gözlemlerinde bugüne dek hep insan algıları esas alınmıştı. Mikroskop gibi, teleskop gibi, ya da güncel bir örnek olması bakımından, Mars'a indirilen uzay aracı gibi detektörlerin yapımında güdülen amaç, insan algılarının erimini doğal sınırlarının ötesine ulaştırmaktır. Sonuçta doğa gözlemi denen şey, insanın dokunarak, duyarak, görerek olguları bilinç alanına (yani zihnine) aktarmasından ibarettir. Sonrası bu verileri akılla işleyerek mantıksal çıkarımlarla sonuca ulaşmaktır. Öte yandan, çağdaş algılayıcılar giderek daha artan oranlarda bilgisayar teknolojisinden yararlanmaktalar. Böylece artık verileri, örneğin bir fotoğraf olarak kağıt üzerinde gözle görülecek biçimde kayda geçirmek yerine digital bilgi bankalarına doldurmaktayız. Bunun çok önemli iki sonucu var. Bunların birincisine veri analizi yoluyla verilerin temizlenerek doğrudan gözlenemeyecek olguların fark edilmesidir dersek ikinci önemli sonuç, bilgisayar iletişim ağlarıyla veri tabanının anında tüm araştırmacılara açılmış bulunmasıdır. Artık yeni olgular, kütleçekimi yasası, termodinamik kuralları, kuantum mekaniği vb. yıllar boyunca süzülerek bizlere kadar gelmiş olan bilgi birikiminin yardımıyla hemen önümüzde, masamızın üstündeki bilgisayar ekranında, belirlemekteler. Günümüzün süper bilgisayarlarının sağladığı hızlı hesap kapasitesi yardımıyla, uzayda dolanan gezegenler, patlayan yıldızlar, çarpışan karadelikler, atmosferde oluşacak fırtına bulutları, okyanuslardaki dalgalar ve daha bunlara benzer nice olgu, temel fizik yasalarından başlanarak matematik hesaplarla ekranda oluşturulup incelenebilmekte. Masamızın üstünde, elimizin altında istersek bir dünya oluşturabiliyoruz. Artık, bir fizik modelini sınamak için doğada gözlem yapmak veya laboratuvarda deney yapmak kadar bilgisayarda benzetişim (simulasyon) yapmak da geçerli kabul edilir bir yöntem olmuştur. Yıldızları gerçekte patlatamayız, veya varlıklarının kanıtları dolaylı olarak gelen karadeliklerden iki tane bulup, üstelik bir de bunları çarpıştıramayız. Ancak tüm bu olaylar bilgisayar benzetişimiyle incelenebilir. Daha pratik düzeyde, örneğin radyasyon korkusu duymadan, bir reaktör tasarımı benzetişim yöntemiyle yapılabilir. Yeni malzemelerin atom yapısı tasarlanıp, bunların laboratuvarda üretimine geçilmeden önce istenen fiziksel ve kimyasal nitelikleri taşıyıp taşıyamayacakları bilgisayar benzetişimiyle irdelenebilir.

Bilgisayar ekranındaki görüntüler gerçek dünyadaki olguların bire bir temsili midirler yoksa sanal bir dünyada hayal mi görmekteyiz? Sakın Evren dediğimiz, muazzam bir bilgisayar ekranından ibaret olmasın? Bilgisayarlar bilinçlendirilebilirler mi? Yani akıllanıp kendi başlarına düşünebilirler mi? Bu ve buna benzer sorulabilecek nice sorunun yakın gelecekteki felsefe tartışmalarına ne kadar uygun bir zemin oluşturdukları açıktır. Daha şimdiden bilgisayar bilimcileri tarafından Yapay Zekâ

(AI=Artificial Intelligence) kavramı ortaya konmuş bulunmaktadır. Eğer, sıkça dendiği gibi, bilgisayarlar yakında yapay zekâ edinebileceklerse neden yapay akıl sahibi de olamasınlar?... Takma kol, takma bacak, suni böbrek gibi yapay organların kullanımını yadırgamıyoruz. İnsan beynini organik bir bilgisayar sistemi olarak yorumlamaya eğilimli bilgisayar bilimcilerine göre yapay beyin fikri de yadırganmayacaktır. Böyle geliştirilmiş bir bilgisayarın aklı neresinde olacaktır? Ama bu tür sorulara dalmadan önce derin felsefe konuları üstünde durup bir daha düşünmek gerek. Akıl nedir? Zekâ nedir? Bilinç nerededir? Düşünce beyinde hangi eylemlerin sonunda oluşur? Belki, zihin dediğimizin bir fiziksel varlığı bile yok. Karmaşık bir matematik hesabı bir anda hatasız yapabilmek, düşünmekle eş tutulabilir mi? Yoksa insan beyninin işlevleri arasında hesap yapabilmenin ötesine geçen bir şeyler mi var?

Roger Penrose, bu yanıtı pek zor konularda kendi görüşlerini, daha önce felsefenin derin tartışmalarından uzak durmuş bir matematikçi ve temel bilimcinin pratik yaklaşımıyla savunmuş. Kitabı, daha yayınladığı günden başlayarak büyük yankılar uyandırdı. Felsefe ve mantıkçıların, bilgisayar bilimcilerinin Penrose'a yönelik eleştirileri dinecek gibi görünmüyor. Anlaşılan o ki bu kitap bilim felsefesinin en çok tartışılan, üzerinde kitaplar yazılacak önemli eserlerinden birisi olmaya adaydır. Her bir bölümünde son derece zor matematik, fizik ve felsefe konularının birbiri peşine ele alındığı bu uzun ve kapsamlı kitabın Türkçe çevirisi, biraz daha kolay izlenebilir yapabilmek düşüncesiyle üç kitap halinde yayınlanmakta. Toplam on bölümden oluşan kitabı, bölümlerinde ele alınan konular itibariyle üç kısıma ayırırken zorlanmadık. İlk dört bölümü içeren birinci kısımda matematik ve fiziksel gerçeklik, akıl yürütmenin sınırları, algoritmalar ve hesaplanabilirlik kavramı, matematikte kanıt, doğruluk ve sezginin önemi ele alınmaktadır. İkinci kısımda yer alan beşinci ve altıncı bölümlerde sırasıyla klasik fizik teorileriyle kuantum fiziğinin, önder konumdaki bir teorik fizikçi tarafından değerlendirilmesi, temel kavramların sorgulanmasına yer verilmektedir. Penrose, başka yazılarında da sıkça değindiği gibi, bilimsel teorileri üç kategoride ele alır: (i) Maxwell teorisi veya Einstein teorisi gibi yetkin teoriler, (ii) Kuantum elektrodinamiği, Salam-Weinberg elektrozayıf etkileşmeler teorisi gibi yararlı teoriler, (iii) Kuantum kozmolojisi, Penrose'un kendisinin tvistor teorisi veya süpercisim teorileri gibi geçici teoriler. Üçüncü kategoriye koyduğu teorilerin tartışması ve Penrose'un bunlar üzerine inşa edilmiş spekülasyona dayalı fikirleri, üçüncü kısmın bölümlerini oluşturmaktadırlar. Evrenin ve tersinemez zaman akışının sınırlarının ele alındığı yedinci bölümü izleyen bölümde bir kuantumlu kütleçekimi teorisinin gerekçeleri ve beklenen nitelikleri tartışılmaktadır. Dokuzuncu bölüm beyin, beyin işlevleri ve bilgisayarların tartışılmasına ayrılmıştır. Üzerinde çok konuşulan kuantum bilgisayarı fikri burada ele alınmaktadır. Sonuncu bölümde fiziksel aklın nerede bulunduğu konusu çevresinde öne sürülen pek çok yeni fikir bir araya toparlanmıştır.

Sanırım, tamamını okumak için verilecek uzun ve zahmetli bir uğraştan sonra kitabın akılda kalacak özü, Penrose'un, genel bir tanımlamayla, bir tür bilgisayarın karmaşık hesap eylemlerinden yararlanılarak insan düşüncesinin modellenebileceği görüşünü benimseyen Yapay Zekâcılar'a karşı olduğudur.

Prof. Dr. Tekin Dereli
ODTÜ Fizik Bölümü
22 Temmuz 1997, Ankara

Bu kitabı, basımını görecek kadar yaşamayan sevgili annemin aziz anısına ithaf ediyorum...

Matematik Denklemleri ile İlgili Açıklama

Bu kitabın birçok yerinde, yazılan her formülün kitabın genel okuyucu sayısını yarı yarıya azaltacağı konusunda yapılan uyarıları umursamadan sıkça matematik formülleri kullanmak zorunda kaldım. Formüllerin sıkıcı olduğunu düşünen okurlardan biriyseniz .(ki pek çok okuyucu böyle düşünür), size bu durumlarda bizzat uyguladığım bir yöntemi uygulamanızı öneririm. Yöntem aşağı yukarı şöyle: Formülün yer aldığı satırı atlayın ve metnin bir sonraki satırına geçiverin! Pekala, böyle yapmayın, belki yöntem tam anlamıyla hiç de böyle değil, ama o kötü formülü dikkatle incelemek yerine şöyle bir göz atar ve hemen sonra metni okumayı sürdürürseniz kısa bir süre sonra, yeniden yüreklenir, ihmal ettiğiniz formüle geri döner, önem taşıyan noktaların farkına varabilirsiniz. Neyin önem taşıdığı konusunda metinde yer alan açıklamalar size yardımcı olabilir. Eğer yardımcı olmuyorlarsa, zaten atladığınız formülü tümüyle aklınızdan çıkarmanızda hiçbir sakınca yoktur.

Teşekkürler

Bu kitabın hazırlanmasında, şu veya bu şekilde, yardımcı olan ve teşekkür borçlu olduğum birçok kişi bulunuyor. Özellikle, Al (Yapay Zekâ) taraftarları (öncelikle bir zamanlar izlediğim bir BBC TV programına katılanlar başta olmak üzere), yıllar önce, güçlü Al ile ilgili görüşleri, açıklama tarzlarıyla beni bu kitabı yazma konusunda özendirdiler (Yine de, girişimimin ileride beni ne türlü zahmetlere sokacağını bilseydim korkarım bu işe hiç kalkışmazdım).

Taslak halindeki çalışmamı kısım kısım inceleyerek, değişiklik yapmam için yararlı önerilerde bulunanlar oldu. Onlara da teşekkür ederim: Toby Bailen, David Deutsch (Turing makinemin kurallarının düzeltilmesinde büyük yardımları olmuştur), Stuart Hampshire, Jim Hartle, Lane Hughston, Angus Mclntyre, Mary Jane Mowat, Tristan Needham, Ted Newman, Eric Penrose, Toby Penrose, Wolfgang Rindler, Engelbert Schücking, ve Dennis Sciama. Mandelbrot kümesiyle ilgili olarak verdiği ayrıntılı bilgiler için Christopher Penrose'a, satranç oynayan bilgisayarlarla ile ilgili olarak verdiği ayrıntılı bilgiler için aynı şekilde Jonathan Penrose'a teşekkür ederim. Uzmanı olmadığım bir konuyu içeren IX. Bölüm'ü okuyarak kontrol eden Colin Blakemore'a, Erich Harth'a ve David Hubel'e özel teşekkür borçluyum. Teşekkür ettiğim kişiler, gözden kaçabilecek yanlışlardan hiçbir şekilde sorumlu değildirler. Bu kitabın bir kısmına temel oluşturan bazı derslerin verildiği Houston Rice Üniversitesine gitmemi sağlayan DMS 84-05644, DMS 86-06488 sözleşmeleri kapsamında destekleri için NSF: *(National Science Foundationfa.* Kuantum mekaniği ile ilgili değerli fikir alış-verişinin yapıldığı Syracuse Üniversitesi'ne gidebilmemi sağlayan PHY 86-12424'e teşekkür ederim. Bu kitaba önsöz hazırlamakla gösterdiği cömertlik ve bazı yorumları için Martin Gardner'a minnettarlığımı ifade etmek isterim. Çeşitli bölümler hakkında dikkatli ve ayrıntılı eleştirisi, kaynak kitaplarla ilgili son derece değerli yardımları ve en önemlisi en çekilmez olduğum

zamanlarda bana göstermiş olduğu anlayış ve en ihtiyaç duyulan anda verdiği derin sevgi ve destek için, sevgili eşim Vanessa'ya teşekkür ederim.

Şekillerle İlgili Teşekkür

Bu kitabın yayıncıları, şekillerin çoğaltılmasına izin verdikleri için aşağıdaki yayıncılara teşekkür eder

Şekil 4.6 ve 4.9, D.A. Klarner (ed), *The mathematical Gardnef* den alıntı (Wadsworth International, 1981).

Şekil 4.7, B. Grünbaum & G.C. Shephard, *Tilings and patterns'den* alıntı (W.H. Freeman, 1987) Copyright © 1987 W.H. Freeman and Company izniyle kullanılmıştır.

Şekil 4.10, K. Chandrasekharan, *Hermann Weyl 1885-1985* (Springer, 1986).

Şekil 4.11 ve 10.3, "Pentaplexity: a class of non-periodic tilings of the plane"den alıntı. *The Mathematical Intelligencer,* 2,32-7 (Springer, 1979).

Şekil 4.12, H.S.M. Coxeter, M. Emmer, R, Penrose ve M.L. Teuber (ed), *M.C. Escher: Art and Science* (North Holland, 1986).

Şekil 5.2, © 1989 M.C. Escher Heirs/Cordon Art-Baarn-Holland.

Şekil 10.4, *Journal of Materials Research,* 2, 1-4 (Materials Research Society, 1987).

Diğer tüm şekiller (4.10 ve 4.12 dahil) yazarın kendisi tarafından hazırlanmıştır.

Önsöz

Büyük matematikçilerin ve fizikçilerin çoğu, mesleklerinden olmayanların anlayabilecekleri bir kitap yazmanın, olanaksız olmasa bile, çok zor olduğunu düşünürler. Bu yıla kadar, dünyanın en bilgili ve yaratıcı matematik fizikçilerinden Roger Penrose'un, bu düşüncede olanlar arasında yer aldığı sanılabilirdi. Teknik olmayan makalelerini ve konferans metinlerini okumuş olan bizler ise bunun böyle olmadığını biliyorduk. Yine de Penrose'un, yoğun çalışmalarından zaman ayırarak, aynı meslekten olmayanlar için harika bir kitap yazdığını görmek hoş bir sürpriz oldu. Bu kitap, klasikler arasında yerini alacağına inandığım bir eser.

Penrose'un kitabının bölümleri, görelilik teorisinden, kuantum mekaniğine ve kozmolojiye kadar değişik konuları kapsamaktaysa da ana düşünceyi, felsefecilerin "us-beden problemi" olarak adlandırdıkları teori oluşturmaktadır. "Güçlü Al" (Yapay Zekâ) savunucuları, elektronik bilgisayarların insan aklının yapabileceği her şeyi yapabilmelerinin yalnız bir veya iki yüzyıllık (bazıları bu süreyi elli yıla kadar kısaltıyor!) bir sorun olduğuna bizi yıllardır ikna etmeye uğraşmaktalar. Gençliklerinde okudukları bilimkurgu kitaplarından etkilenen ve beyinlerimizin yalnızca (Marvin Minsky'in dediği gibi) 'etten yapılmış bilgisayarlar' olduğuna kendilerini inandıran bu insanlar, haz ve acı, güzellik ve mizah beğenisi, bilinçlilik, hür irade gibi yetilerin, elektronik robotlarda, algoritmik davranışları yeterince karmaşık duruma geldiği zaman doğal olarak ortaya çıkacağından kuşku duymamaktadırlar.

Bazı bilim felsefecileri (ünlü Çin odası düşünce deneyimini Penrose'un derinlemesine ele aldığı John Searle gibi felsefeciler) bu düşünceye şiddetle karşı çıkmaktadırlar. Onlara göre bir bilgisayar, çarklarla, kollarla veya sinyal ileten herhangi bir mekanizmayla çalışan mekanik hesap makinelerinden temelde farklı değildir (Bir bilgisayar, yuvarlanan bilyaların veya boruların içinde akıp giden suyun üzerine dayandığı fizik yasalarına uygun yapılabilirdi). Elektrik, tellerin içerisinde, (ışık hariç) diğer enerji türlerine göre daha hızlı iletilebildiği için simgelerle hesap makinelerinde olduğundan daha süratle oynayabilir ve bu nedenle son derece karmaşık işlemlerin üstesinden gelebilir. Fakat, elektrikli bir bilgisayar gerçekleştirdiği işlemi, bir abaküsün 'anladığından' fazla 'anlar' mı? Bilgisayarlar şimdilerde satrancı ustaca oynuyorlar. Acaba satrançtan, bir zamanlar bir grup bilgisayar meraklısının derme çatma aletleriyle yaptıkları bir basit oyun makinesinden daha fazla 'anlarlar' mı?

Penrose'un kitabı, şimdiye kadar yazılmış güçlü Al karşıtı olan en etkin saldırıdır. Geçen yüzyıllarda aklın, bilinen fizik yasalarıyla çalışan bir makine olduğuna dair indirgemeci sava karşı itirazlar yapılmıştı, fakat Penrose'un karşı saldırısı, kendisinden önceki yazarların sahip olmadıkları bilgilere dayanılarak yapıldığı için, daha inandırıcı. Kitap, Penrose'un bir matematiksel fizikçiden öte, birinci sınıf bir felsefeci olduğunu, çağdaş felsefecilerin anlamsız bularak göz ardı ettikleri sorunları irdelemekten çekinmediğini açıkça göstermektedir.

Penrose, aynı zamanda, küçük bir grup fizikçinin giderek artan itirazlarına karşın, güçlü bir gerçekçiliğin savunuculuğunu üstlenmek cesaretini de göstermiştir. Yalnız "orada bir yerdeki" evren değil matematiksel gerçek de kendine özgü gizemli bağımsızlığa ve zamansızlığa sahiptir. Newton ve Einstein gibi Penrose da, gerek fiziksel dünyaya gerekse saf matematiğin Platonik dünyasına son derece alçakgönüllükle ve derin saygıyla yaklaşıyor. Sayı teoricileri arasında seçkin bir yeri bulunan

Paul Erdös, en iyi kanıtların yer aldığı 'Tanrının Kitabı' hakkında konuşmaktan hoşlanır. Matematikçilerin, bu kitabın bir sayfasına ara sıra göz atmalarına izin verilir. Bir fizikçi veya bir matematikçi birdenbire bir sezgiye kapıldığı zaman Penrose bu sezginin 'karmaşık hesaplamayla uyandırılmış' bir duygunun ötesinde bir şey olduğuna inanır. Bu, bir an için nesnel gerçekle ilişkiye geçen us'tur. Acaba, diye düşünür, Platon'un dünyası ile fiziksel dünya (fizikçilerin matematiğin potasında erittikleri dünya) gerçekte bir ve aynı mıdır? Penrose'un kitabının birçok sayfası, adını Benoit Mandelbrot'tan alan Mandelbrot kümelerine ayrılmıştır. Böyle bir kümenin parçaları büyütüldüğünde istatistiksel anlamda birbirine benzer yapı göstermesine karşın, sonsuz helis biçimi önceden tahmin edilemeyecek şekilde sürekli değişim göstermektedir. Penrose, bu egzotik yapının, tropik bir ormanın keşfi kadar olası, 'işte orada' durup duran Everest Dağı kadar somut olduğunun düşünülmesinin anlaşılmaz olduğu kanısındadır (Ben de öyle).

Penrose, 'küçük parmağının' kendisine, kuantum mekanik biliminin henüz tamamlanmamış olduğunu söylediği zaman Einstein'ın hiç de dik kafalı veya bulanık zihinli olmadığına inanan ve sayıları giderek artan bir grup fizikçiden birisidir. Görüşünü desteklemek amacıyla Penrose sizi, günümüzdeki spekülasyonların odağını oluşturan, karmaşık sayılar, Turing makineleri, karmaşıklık teorisi, kuantum mekaniğinin şaşırtıcı paradoksları, formel sistemler, Gödel karar verilemezliği, faz uzayları, Hilbert uzayları, karadelikler, akdelikler, Hawking ışıması, entropi, beynin yapısı gibi konulan kapsayan göz kamaştırıcı bir yolculuğa çıkarıyor. Köpekler ve kediler, kendilerinin 'bilincinde' midirler? Televizyonun Uzay Yolu dizisindeki gibi astronotların bir madde-aktarım sistemi yoluyla oraya buraya yollanabilmeleri teoride mümkün müdür? Evrim bilinçlenmeyi yaratırken kendi sürekliliğini sağlamak için onda ne bulmuştur? Kuantum mekaniğinin ötesinde, zamanın yönünün ve sağ/sol ayırımının kesin olarak belirlenmiş olduğu bir düzey var mıdır? Kuantum mekaniği yasaları, bundan da belki daha derin yasalar, usun işlevlerini yerine getirmesi için mutlaka gerekli midir?

Son iki soruya Penrose'un yanıtı evet olmuştur. Onun ünlü 'tvistor' teorisi -uzay zamanı içeren daha yüksek boyutlu kompleks bir uzayda etkin olan soyut geometrik nesneler teorisi- bu kitabın kapsamına alınmayacak kadar tekniktir. Penrose'un yirmi yıllık uğraşının ürünü olan 'tvistorlar', kuantum mekaniksel alanlardan ve taneciklerden daha derin bir bölgede geçerli olmak için geliştirilmiş bir teoridir. Penrose, yetkin, yararlı, geçici ve yanlış-yönlendirilmiş olarak dört gruba ayırdığı teoriler içerisinde tvistor teorisini, bugün hararetle tartışılmakta olan süper-sicim ve diğer büyük birleştirme modellerinin yanı sıra, alçakgönüllülükle, geçici gruba dahil etmektedir.

1973'ten bu yana Penrose, Oxford Üniversitesinde Rouse Ball Matematik Profesörü olarak görev yapmaktadır. Bu ünvanı haklı olarak taşımaktadır çünkü W. W. Rouse Ball yalnız çok tanınmış bir matematikçi olmayıp, *Matematiksel Eğlencelikler ve İnceleme Yazıları* adlı bir İngiliz klasiği niteliğinde eseriyle, eğlencelik matematik alanında amatör bir sihirbazdır.

Penrose, Ball'un eğlence hevesini paylaşmaktadır. Gençliğinde 'üçlü çubuk' (tribar) denilen 'imkânsız bir nesne' keşfetmişti (İmkânsız nesne, içten-çelişkili elemanlardan oluştuğu için varolamayan çok-boyutlu bir şeklin çizimidir). Penrose ve bir genetik uzmanı olan, babası Lionel, bu tribar'ı bir Penrose Merdivenine dönüştürdüler. Basamaklı bu yapıyı, Maurits Escher *Ascending and Descending* (Çıkış ve İniş) ve *Waterfall* (Çağlayan) adlı iki ünlü gravüründe kullandı. Bir gün Penrose, 'delilik nöbeti' olarak adlandırdığı bir nedenle yatakta yatarken, dört-boyutlu bir uzayda var olan bir imkânsız nesneyi hayalinde canlandırdı. Bu öyle bir şey ki, dedi, dört-boyutlu bir yaratık ona rastlasa, "Aman Tanrım, bu da ne böyle?" diye haykırırdı.

1960'larda Penrose, arkadaşı Stephen Hawking ile birlikte evren bilimi konusunda çalışırken,

belki de en çok tanınan buluşunu gerçekleştirdi. Görelilik teorisinin 'sonuna dek geçerli' olduğu savına göre, her karadelik, içinde fizik yasalarının geçerli olmadığı bir tekil bölgeye sahip olmalıydı. Bu başarı bile, son yıllarda Penrose'un, Escher mozayiğine benzer fakat düzlemi periyodik olmayan tarzda kaplayan iki şekil inşa etmesiyle gölgelendi (Bu şaşırtıcı şekiller hakkında bilgiyi *Penrose Tiles to Trapdoor Ciphers* adlı kitabında bulabilirsiniz). Bu şekilleri, Penrose hiçbir yararı olabileceğini düşünmeden icat etti, daha doğrusu keşfetti. Penrose'un kaplama taşlarının üç-boyutlu genellemelerinin, alışıldık olmayan yeni bir madde türünün bulunabileceği sonucuna ulaşması herkesi şaşırttı. Bugünlerde söz konusu "kristalimsi" maddenin incelenmesi kristalografi alanında en yoğun araştırma konularından birisini oluşturmaktadır. Aynı zamanda, matematiğin hiç beklenmeyen uygulamalara yol açabileceğini göstermesi açısından, modern zamanların en dramatik örneğidir.

Penrose'un matematik ve fizik alanlarındaki başarıları -yalnız bir kısmına değinebildiğim başarıları- onun bir yaşam boyu, Varoluşun gizemine ve güzelliğine duyduğu meraktan kaynaklanmaktadır. Küçük parmağı ona, insan beyninin sadece minik teller ve anahtarlar topluluğundan oluşmadığını söylüyor. Kitabının başlangıç ve sonsöz bölümlerindeki Adam, kısmen duygulu yaşamın yavaş evriminde bilincin başlangıcının bir sembolüdür. Benim için Penrose, güçlü Al hükümdarlarının çıplak olduklarını, üzerlerinde giysilerinin bulunmadığını söylemek yürekliliğini gösteren -Al öncülerinin biraz uzağında, arkada, üçüncü sırada oturan- çocuktur. Penrose'un fikirlerinden çoğuna mizah karıştırılmıştır ama bu kitapta savunduğu fikir asla güldürü malzemesi değildir.

Martin Gardner

Başlangıç

Büyük Konferans Salonunda, yeni 'Ultronic' bilgisayarın açılışı için toplanılmıştı. Başkan Pollo, açılış konuşmasını henüz bitirmişti. Bitirdiği için mutluydu: Bu gibi toplantılardan pek hoşlanmazdı ve açılışını yaptığı bilgisayarın kendisine bir hayli zaman kazandıracağı gerçeğinin dışında bilgisayarlardan hiç anlamazdı. İmalâtçı firma, bilgisayarın diğer birçok görevinin arasında, Devletle ilgili olarak Başkanın çok sıkıcı bulduğu kararları almak bulunduğuna dair Başkana güvence vermişti. Bilgisayara harcadığı hazine altınının miktarı düşünülürse, bilgisayarın bu görevi üstlenmesi elbette yerinde olacaktı. Şahane özel golf sahasında saatler boyu golf oynayarak hoşça vakit geçirmek için sabırsızlanıyordu. Golf sahası, küçük ülkesinde geriye kalabilen birkaç büyük yeşil alandan bir tanesiydi.

Adam, açılış törenine katılanların arasında bulunmakla kendini ayrıcalıklı hissediyordu. Üçüncü sıraya oturdu. Ultronic'in tasarımında görev alan teknokratlardan birisi olan annesi iki sıra önünde oturuyordu. Babası da rastlantı sonucu salondaydı; davetli değildi ve şu anda arkada bir yerlerde çevresi güvenlik görevlileri tarafından tamamen kuşatılmış olarak duruyordu. Son dakikada Adam'ın babası, bilgisayarı havaya uçurmaya kalkışmıştı. Küçük bir eylemci grup olan 'Psişik Bilinçlenme Yüksek Kurulu'nun 'Liderliği' ünvanını kendine uygun görmüş ve bilgisayarı havaya uçurma görevini üstlenmişti. Kuşkusuz o ve tüm patlayıcıları, sayısız elektronik ve kimyasal-duyarlı cihazlar sayesinde hemen yakalanmıştı. Cezasının küçük bir kısmını şimdiden çekiyordu; bilgisayarın açılış törenine tanık olması için salonda alıkonulmuştu.

Adam, ebeveynlerine pek aldırmazdı. Onlar için herhangi bir duygu beslemesi gerekmemişti. On üç yıllık ömrü boyunca maddi bir zenginlik ortamında, hemen hemen tamamen bilgisayarlar tarafından yetiştirilmişti. Bir düğmeye basarak, istediği her şeye sahip olmuştu: yiyecek, içecek, arkadaş, eğlence ve hatta eğitim -gereksinim duyduğu anda dilediği bilgi, ilginç ve renkli grafikler halinde ekranda beliriyordu. Annesinin mesleki konumu ona bütün bu olanakları sağlamıştı.

Şu anda, Tasarım Şefi, konuşmasını bitirmek üzereydi: "...10^{17} mantık ünitesinden fazlasına sahiptir. Bu rakam, tüm ülkede yaşayan insanların tümünün beyinlerindeki nöronların toplam sayısından fazladır. Zekâsı, hayal bile edilemez. Neyse ki hayal etmek zorunda da değiliz. Biraz sonra zekâsını izlemek, onuruna sahip olacağız: Büyük ülkemizin saygıdeğer First Lady'si Madam Isabella Pollo'yu, olağanüstü Ultronic Bilgisayarımızı çalıştırması için huzurunuza davet ediyorum!"

Başkanın eşi öne çıktı. Biraz heyecanlı görünüyordu; ağzında birkaç sözcük geveliyerek şalter kolunu indirdi. Salonda derin bir sessizlik ve 10^{17} mantık ünitesinin aktive edilmesiyle ışıklarda hemen hemen fark edilemeyen bir kararma yaşandı. Ne olup biteceğini bilemeden herkes bekliyordu. "Yeni Ultronic Bilgisayar Sistemimize ilk sorusunu sorarak sistemi başlatmak isteyen var mı aranızda?" diye sordu Tasarım Şefi.

Herkesin önünde ve Yeni Bilgisayarın huzurunda kendilerini aptal yerine koymaktan çekinen insanlar susuyordu. Salonda suskunluk egemendi. "Aranızda soru soracak birisi mutlaka vardır ama," diye yineledi Tasarım Şefi. Fakat, yeni ve çok güçlü bir bilinçlenme duygusuyla herkes susuyordu. Adam böyle bir duyguya kapılmadı. Doğduğundan beri bilgisayarlarla büyümüştü. Bir bilgisayarın

neye benzediğini hemen hemen biliyordu. En azından bildiğini sanıyordu. Her neyse, meraklanmıştı. Adam elini kaldırdı. “Tamam,” dedi Tasarım Şefi, “üçüncü sıradaki küçük çocuk. Yeni dostumuz için bir sorunuz var, öyle mi?”

I. Bölüm
Bir Bilgisayar Us Sahibi Olabilir mi?

Giriş

Son yirmi-otuz yılda elektronik bilgisayar teknolojisi dev adımlarla gelişti. Önümüzdeki yirmi-otuz yıl içerisinde de, hız, kapasite ve mantık tasarımında büyük ilerlemeler kaydedileceği konusunda pek az kuşku var. Geçen yılın hesap makineleri bu yıl gözümüze nasıl ilkel ve hantal görünüyorsa bugünün bilgisayarları da ileride gözümüze aynı şekilde görünebilir. Gelişim hızında neredeyse ürkütücü bir şeyler var. Eskiden yalnız insanın düşünce sisteminin alanında gerçekleştirilebilen sayısız işlemler bugün bir insanın asla erişemiyeceği hız ve doğruluk oranında, bilgisayarlar tarafından gerçekleştirilebiliyor. *Fiziksel* yönden performansımızı kolayca aşan makinelere uzun zamandır alışkınız. Onlardan herhangi bir rahatsızlık duymuyoruz. Aksine, en hızlı atletten en az beş kat hızlı ilerleyebilen araçlarla yolculuk etmekten, veya düzinelerle işçiden oluşan ekiplerle ancak gerçekleştirilebilecek kazı ve yıkım çalışmalarını makinelerin başarıyla üstlenmesinden hoşnutuz. Önceleri asla yapamadığımız şeyleri *fiziksel* olarak yapmamıza olanak sağlayan makinelere sahip olmakla daha da mutluyuz; gökyüzünde süzülerek birkaç saat gibi kısa bir sürede okyanusun karşı sahiline inebiliyoruz. Makinelerin bu gibi başarılan gururumuza dokunmuyor. Ama, *düşünme* yeteneğine sahip olmak -işte bu çok insanca bir özellik. Ne de olsa düşünme yeteneğimiz sayesinde fiziksel yetersizliğimizi aşabildik ve diğer canlılara karşı üstünlük sağlayabildik.

Üstünlüğümüzü kanıtlayan bu önemli özelliğimizi bir gün makinelere kaptırırsak, onlara boyun eğmek zorunda kalmayacak mıyız?

Mekanik bir cihazın düşünebilmesi, hatta duygulara veya bir usa sahip olması konusu aslında yeni değildir.[1] Fakat modern bilgisayar teknolojisinin gelişmesiyle yeni bir ivme, hatta bir ivedilik kazanmıştır. Sorun, felsefenin derin konularına götürüyor bizi. Düşünmek veya hissetmek nedir? Us nedir? Us gerçekten var mıdır? Varsa, ilişkili olduğu fiziksel yapılara fonksiyonel olarak ne ölçüde bağımlıdır? Bu gibi yapılardan tamamen bağımsız olarak var olabilir mi? Veya yalnızca fiziksel yapıların (uygun türde yapıların) bir fonksiyonu mudur? Ne olursa olsun, ilgili yapıların özellik itibariyle biyolojik (beyin) olması mı gerekli, yoksa elektronik ekipman parçaları da pekâlâ aynı işlevi üstlenebilir mi? Us, fizik yasalarına bağımlı mı? Gerçekte, fizik yasaları nelerdir?

Bu kitapta, bu gibi sorulara yanıt arayacağım. Böylesine geniş kapsamlı sorulara kesin yanıtlar beklemek kuşkusuz haksızlık olur. Kesin yanıt veremem, başkaları da veremez. Ne var ki bazıları tahminleriyle bizi etkilemeye çalışabilir. Benim kendi tahminlerim, ele aldığım konularda önemli rol oynayacak tahminler olacaktır. Fakat, spekülasyonu katı bilimsel gerçekten açıkça ayırt etmeye çalışacağım, ve spekülasyonlarımın temelinde yatan nedenler hakkında açık olmaya özen göstereceğim. Ancak, burada benim başlıca amacım yanıtlarla ilgili çok fazla spekülasyona kalkışmak değildir. Daha çok, fiziksel yasanın yapısı, matematiğin ve bilinçli düşünme işleminin özelliği arasındaki ilişkiyle ilgili yeni yasaları ortaya koymak ve daha önce açıklandığına tanık olmadığım bir bakış açısı getirmek uğraşı içerisinde olacağım. Bu bakış açısını birkaç sözcükle yeterince tanımlayamam; işte bu nedenle, böylesine hacimli bir kitap yazmayı amaçladım. Ancak, yine de kısaca ifade etmem gerekirse, bana göre, fiziksel ve mantıksal açıdan ‘us’ kavramını tam

anlamıyla açıklayamamamız, fiziğin temel yasalarını tam anlamıyla kavrayamamış olmamızdan kaynaklanmaktadır. Bununla, söz konusu yasaları hiçbir zaman tam anlamıyla anlayamayacağız demek istemiyorum. Aksine, gelecekteki araştırmaları bu konuda ümit verici görünen yönlere çekmek, 'us'un bildiğimiz fiziğin gelişim kapsamındaki yeri hakkında oldukça özel ve görünüşte yeni önerilerde bulunmak bu kitabın amacının bir kısmını oluşturmaktadır.

Savunduğum görüşün fizikçiler arasında benimsenmeyen ve sonuçta bilgisayar bilimcileri veya fizyologlar tarafından bugün için benimsenmesi olasılığı bulunmayan bir görüş olduğunu açıkça belirtmeliyim. Fizikçilerin, pek çoğu, insan beyni ile ilgili olarak uygulanabilir temel yasaların tümünün gerçekten mükemmelen bilindiğini iddia edeceklerdir. Genelde fizik bilgimizde hâlâ birçok boşluğun bulunduğu kuşkusuz tartışılamaz. Örneğin, doğanın atom-altı parçacıklarının kütle-değerlerine ve bunların etkileşmelerinin büyüklüklerine hükmeden temel yasaları bilmiyoruz. Kuantum teorisini Einstein'ın genel görelilik teorisiyle uyumlu hale getirecek 'kuantum gravitasyon' teorisinin nasıl geliştirileceğini bir tarafa bırakın, kuantum teorisini daha Einstein'ın özel görelilik teorisi ile nasıl uyumlu hale getireceğimizi tam bilmiyoruz. Kuantum gravitasyon teorisini geliştirememiş olmamızın sonucu, bilinen temel parçacıkların boyutunun 1/100000000000000000000'i gibi minik bir ölçekte uzayın özelliğini anlayamıyoruz, fakat söz konusu boyuttan daha büyük boyutlar için bilgimizin yeterli olduğu farz ediliyor... Bu gibi belirsizliklerin fizik üzerindeki etkileri insan beyni açısından önemsiz gibi görünse de, evrenin kapsamının -uzayda veya zamanda- bir bütün olarak sonlu veya sonsuz olduğunu bilmiyoruz. Karadeliklerin merkezlerinde veya evrenin kendisinin büyük patlama ile başlangıç anında etkin olması gereken fizik bilimini anlamıyoruz. Bütün bu yasalar, insan beyninin işlevleriyle ilgili 'günlük' ölçeğe (veya biraz daha küçüğüne) dayanarak hayal edebileceği kadar uzak görünebilir. Ve gerçekten de uzaktır! Ancak, tam burnumuzun dibinde (veya daha doğrusu ardında) durmakta olan bir konu var ki fizikçilerin pek çoğu, insan düşüncesinin ve bilincinin işlemesi ile yakından ilgili olup fizik anlayışımızda bir büyük bilinmeyeni oluşturan bu yasasının farkında bile değiller. Bu konuyu açıklamaya çalışacağım, Karadeliklerin ve büyük patlamanın bilinmeyen yasalar üzerinde kesin etkilere sahip görüşler olduğunu yine bu kitapta savunacağım.

İleriye sürmeye çalıştığım görüşün temelinde yatan **kanıtın** gücü hakkında okuru ikna etmek çabası içinde olacağım. Fakat bu görüşü anlamak için yapacak çok işimiz olacak. Bize yabana gelecek diyarlardan -bir kısmı görünüşte konumuzla ilgili olmayabilir- ve birbirinden farklı engellerden geçecek zahmetli bir yolculuğa çıkacağız. Newton mekaniğinin temel ilkelerinin yanı sıra, kuantum teorisinin yapısını, esaslarını ve bilinmeyen yöntemlerini, özel ve genel göreliliğin, karadeliklerin, büyük patlamanın, termodinamiğin ikinci yasasının, Maxwell'in elektromanyetizma teorisinin temel özelliklerini incelemek durumunda kalacağız. Bilincin doğasını ve fonksiyonunu anlamak çabamız içerisinde felsefe ve psikolojinin sorunları devreye girecek. Kuşkusuz, önerilen bilgisayar modellerinin yanı sıra, beynin gerçek nörofizyolojisine göz gezdirmemiz, kaçınılmaz olacak. Yapay zekânın statüsü hakkında fikir edinmemiz gerekecek. Turing makinesinin ne gibi bir makine olduğunu öğrenecek, hesaplanabilirlik, Gödel'in teoremi ve teorinin karmaşıklığının anlamım anlamak gereğini duyacağız. Matematiğin derinliklerine dalmamız ve hatta fiziksel gerçekliğin doğasını sorgulamamız gerekecek.

Bütün bunların sonunda okur, açıklamaya çalıştığım alışılmadık kanıtlar hakkında hâlâ ikna olamazsa, yine de bu zahmetli ve umarım ilginç yolculuktan gerçek değere sahip bir şeyler edineceğini en azından ümit ediyorum.

Turing Testi

Hafiza bankası ve mantık ünite sayısı, insan beynindekinden fazla, yeni bir model bilgisayarın piyasaya çıktığını varsayalım. Yine bu bilgisayarların dikkatle programlanmış ve uygun türde ve büyük miktarda veriyle yüklenmiş olduklarını varsayalım. Yapımcıları, böyle cihazların gerçekten *düşündüklerini* iddia ediyorlar. Belki gerçekten zeki olduklarını da iddia ediyorlar. Veya daha da ileri giderek cihazların acıyı, mutluluğu, şefkati, gururu, vs. gerçekten *hissettiklerini* ve ne yaptıklarının farkında olduklarını ve işlevlerini gerçekten *anladıklarını* ileri sürüyorlar. Öyle görünüyor ki bu cihazlar *bilinçliler*.

Yapımcıların iddialarının inanılır olup olmadığını nasıl anlayacağız? Normal olarak, bir makine satın aldığımız zaman değerini, bize sağladığı hizmetle ölçeriz. Beklediğimiz hizmetleri tatminkâr şekilde yerine getiriyorsa memnun oluruz, getirmiyorsa geri götürerek yenisiyle değiştirir veya onarılmasını sağlarız. İnsan fonksiyonlarına sahip olduğu iddia edilen söz konusu makine ile ilgili bu iddiaların doğruluğunu anlamak için, yapımcının kriterine uygun olarak, makinenin fonksiyonlarını yerine getirirken bir insan gibi davranıp davranmadığını sorarız. İnsan gibi davranıyorsa, yakınmamız için bir neden olamaz ve bilgisayarı onarım veya değiştirmek amacıyla iade etmemiz gerekmez.

Bu örnek bize tamamen işlevsel bir yaklaşım sağlamaktadır. İşlevselci, bilgisayarın, düşünme anında bir insanın davranışından ayırt edilemiyecek tarzda bir *davranış* sergilemesi koşuluyla *düşündüğünü* söyleyecektir. Böyle bir işlevsel bakış açısını bir an için benimseyelim. Elbette bilgisayarın, düşünmekte olan bir insanın yapabileceği gibi, odanın ortasında bir oraya bir buraya gidip gelmesini bekleyemeyiz. Veya diyelim ki, bir insana elinizle dokunduğunuzda göstereceği tepkiyi veya hissedeceği duyguyu göstermesini bilgisayardan bekleyemeyiz: Bu beklentiler, bilgisayarın amacının dışındadır. Ancak bu beklentiler, sorduğumuz herhangi bir soruya insanınkine benzer yanıtlar vermesini istediğimiz ve yanıtlarının bir insanınkinden ayırt edilemiyecek olması koşuluyla gerçekten düşünmesiyle (veya hissetmesiyle, anlamasıyla, vs.) tatmin olmayı talep ettiğimiz anlamına gelir.

Bu bakış açısı, Alan Turing'in 1950 yılında *Mind* (Turing 1950) felsefe dergisinde yayınlanan 'Computing Machinery and Intelligence' başlıklı ünlü makalesinde çok güçlü şekilde savunulmuştur (Turing'den ilerideki bölümlerde daha çok söz edeceğiz). Makalede, bu görüş, ilk kez *Turing testi* olarak tanımlanmıştır. Testin amacı bir makinenin düşündüğünü söylemenin mantıksal olarak mümkün olup olmadığıdır. Bir bilgisayarın (yukarıda verdiğimiz örnekte yapımcıları tarafından özellikleri tanımlanan bilgisayar gibi) gerçekten düşündüğünün iddia edilmekte olduğunu varsayalım.

Turing testine göre bilgisayar, gönüllü bir insanla birlikte, sorgulayıcının görüş alanının (perseptif) dışında bir yere saklanır, Sorgulayıcı, yalnız soru sormak suretiyle, hangisinin insan hangisinin bilgisayar olduğunu saptamaya çalışır. Sorgulayıcının[1] soruları, daha önemlisi aldığı yanıtlar, tamamen ses gizlenerek, yani ya bir klavye sisteminde yazılarak veya bir ekranda gösterilerek verilir. Sorgulayıcıya, bu soru/cevap oturumunda elde edilen bilgiler dışında, her iki taraf hakkında hiçbir bilgi verilmez. İnsan denek soruları içtenlikle yanıtlar ve kendisinin insan, öteki deneğin bilgisayar olduğuna dair sorgulayıcıyı ikna etmeye uğraşır; fakat bilgisayar 'yalan' söylemeye programlanmış olduğu için kendisinin insan olduğuna sorgulayıcıyı inandırmaya çalışır. Dizi halinde tekrarlanan testler süresince sorgulayıcı, tutarlı bir şekilde, insanı saptayamadığı takdirde, bilgisayar (veya bilgisayarın programı, veya programlayıcısı, veya tasarımcısı, vs.) testi geçmiş sayılır.

Bu testin bilgisayar için adil olmadığı öne sürülebilir. Roller değiştirilerek insanın bilgisayar gibi davranması ve bilgisayardan sorulara içtenlikle, doğru yanıtlar vermesi istenseydi sorgulayıcının

hangisinin insan hangisinin bilgisayar olduğunu anlaması çok kolay olurdu. Yapması gereken bütün iş, çok karmaşık bir aritmetik hesabının yapılmasını istemek olurdu. İyi bir bilgisayar yanıtı hemen ve doğru şekilde verirken insan yanıtı verirken zorlanırdı. (Bu konuda biraz dikkatli olmak gerekiyor, çünkü herhangi bir aritmetik işlemini, şaşmaz doğrulukla ve fazla çaba sarfetmeksizin zihninde çabucak gerçekleştirebilen 'hesap dahileri' bulunabilir. Örneğin, okuma-yazma bilmeyen bir çiftçinin oğlu olan ve 1824-1861 yılları arasında Almanya'da yaşamış olan Johann Martin Zacharias Dase[2], herhangi sekiz basamaklı iki sayının çarpımını bir dakikadan az bir sürede, yirmi basamaklı iki sayının çarpımım altı dakika gibi kısa bir sürede zihninde gerçekleştirebiliyordu! Daha yakın dönemlerde, 1950lerde, Edinburgh Üniversitesi Matematik Profesörü Alexander Aitken'in bu konuda insanı etkileyen başarıları olmuştur. Test amacıyla sorgulayıcının daha zor işlemler seçmesi, örneğin otuzar basamaklı iki sayının çarpma işleminin iki saniyede tamamlanmasını öngören sorular seçmesi gerekecektir. Böyle bir soru, iyi bir modern bilgisayarın kolaylıkla üstesinden gelebileceği bir işlemdir).

Böylece, bilgisayar programcılarına bilgisayarın, bazı durumlarda gerçekte olduğundan daha 'aptal' görünmesini sağlamak düşüyor. Çünkü, sorgulayıcının bilgisayara karışık bir aritmetik sorusu sorması halinde, bilgisayarın soruyu yanıtlayamıyormuş gibi görünmesi gerekir; aksi halde, kısa sürede kendini ele vermiş olacaktır! Bilgisayarın bu şekilde daha 'aptal' görünmesini sağlamak sanırım programcılar için özellikle ciddi bir sorun yaratmayacaktır. Onlar için asıl sorun, her insanın kolayca yanıtlayabileceği en basit 'aklı selim' sorusunu yanıtlayabilecek şekilde bilgisayarı programlayabilmektir!

Bu tür soruların belirli örneklerinin yinelenmesinde, yineleme olayının doğasında bulunan bir sorun ortaya çıkar. İlk soruda, bilgisayarın bu *özel* soruyu bir insanın yanıtlayabileceği şekilde nasıl yanıtlayacağına dair bir yol bulmak kolaydır. Oysa *sürekli* sorgulamada, özellikle özgün ve biraz gerçek anlama yeteneği gerektiren sorularla yapılan seri sorgulamada bilgisayarın gerçek anlama yeteneğinden yoksun olduğunun ortaya çıkması olasıdır. Sorgulayıcının becerisi, kısmen, bu tür özgün soru biçimleri bulabilmesinde, kısmen de, derinlemesine inceleme niteliğinde bir yaklaşımla, gerçek 'anlayışın' var olup olmadığını ortaya çıkaracak şekilde tasarımlanmış diğer sorularla bağdaştırabilmesindedir. Sorgulayıcı, aradaki farkı bilgisayarın fark edip etmeyeceğini anlamak için, sorgulama esnasında tamamen saçma bir soruyu araya sıkıştırabilir, veya saçma gibi görünen fakat aslında anlamlı olan bir iki soruyu diğer soruların arasına katabilir: Örneğin, şöyle sorabilir: 'Duyduğuma göre, bu sabah bir gergedan pembe bir balonla Mississipi boyunca uçmuş. Buna ne dersin?' (Bilgisayarın alnında biriken soğuk ter taneciklerini görür gibisiniz!) Bilgisayar ihtiyatla yanıtlar: "Oldukça gülünç geldi bana." Şimdiye kadar iyi gitti. Sorgulayıcı: "Sahi mi? Bir zamanlar amcam da aynı şeyi yapmıştı -gitmiş ve dönmüştü-yalnız onunki kirli-beyaz renkte ve çizgiliydi. Bunda gülünç olan ne var?" Doğru dürüst anlama yeteneği yoksa, bilgisayarın çok geçmeden tuzağa düşeceğini ve kendini ele vereceğini tahmin etmek zor değil. Hafıza bankasında gergedanların kanatlarının olmadığına dair bilgi varsa, ilk soruyu "Gergedanlar uçamaz" diyerek, ikinci soruyu da "Gergedanlar çizgili değildir" şeklinde yanıtlayabilir. Sorgulayıcı, bilgisayarın aradaki temel farkı anlayıp anlamadığını kontrol etmek amacıyla, bir sonraki sorusunu, gerçekten saçma bir soruya dönüştürmek için 'Mississipi'nin altında' veya 'pembe bir balonun içerisinde' veya 'pembe bir gecelikle' şeklinde değiştirebilir.

Turing testini geçebilecek bilgisayarın yapılıp yapılamayacağı, veya ne zaman yapılabileceği gibi konuları şimdilik bir tarafa bırakalım. Yalnızca bir tartışma başlatabilmek için bu tür makinelerin imâl edilmiş olduklarını varsayalım. Peki, Turing testinde başarılı olan bir bilgisayarın *mutlaka*

düşündüğü, hissettiği, anladığı, vs. iddia edilebilir mi? Bu konuya az sonra değineceğim. Şimdilik, elimizdeki sonuçların bazılarını değerlendirelim. Örneğin, bilgisayarın yapımcıları iddialarında, yani cihazlarının düşünen, hisseden, duyarlı, anlayışlı, *bilinçli* bir varlık olduğuna dair iddialarında haklıysalar, cihazı satın almamız bizi *ahlâki sorumluluklarla* karşı karşıya bırakacaktır. Yapımcılarına inanmamız gerekiyorsa elbette ki bu böyle olacaktır! Bilgisayarın duygularını hiçe sayarak, sırf ihtiyaçlarımızı tatmin için onu kullanmamız hiç de hoş karşılanmayacaktır. Bir kediye kötü davranmakla bunun arasında ahlâk açısından bir fark yoktur. Yapımcısının iddiasına göre duygulu bilgisayarımızı incitmekten sakınmamız gerekecektir. Bize bağlandığı, yakınlık duyduğu bir sırada devresini kapatıvermemiz veya hatta bir başkasına satıvermemiz bizi ahlâki zorluklarla karşı karşıya bırakabileceği gibi, başka insanlarla veya hayvanlarla kurulacak duygusal ilişkilerin bizi içine sürükleyebileceği sayısız başka sorunlar da söz konusudur. Bütün bunlar, yaşantımızı yakından ilgilendiren sorunlardır artık. Bizim için (ve de yetkili makamlar için!), imalâtçıların, "Düşünen cihazlarımızdan her biri uzmanlarımız tarafından tümüyle Turing testinden geçirilmiştir" gibi bir kanıtla desteklenmiş iddialarının doğru olduğunu varsayalım.

İddiaların bazı sonuçlarının, özellikle ahlâki sonuçlarının, görünüşteki anlamsızlığına, gülünçlüğüne rağmen, Turing testinden başarıyla geçmiş olmanın, düşünce, zekâ, anlayış veya bilinçlenmenin geçerli bir göstergesi sayılması bence oldukça etkin bir kriterdir. Çünkü, başkalarının böyle özelliklere sahip oldukları hakkında karşılıklı konuşma yöntemi dışında başka hangi yöntemle, nasıl yargıya varabiliriz? Yüz ifadesi, bedenin hareketleri ve benzer hareketler gibi başka kriterler de vardır gerçi ama diyelim ki (belki biraz daha uzak bir gelecekte) bütün bu mimikleri ve hareketleri başarıyla taklit edebilen bir robot yapıldı. Robotu ve insanı, sorgulayıcıdan saklamak artık gerekmeyecek fakat, ilke olarak, sorgulayıcının elinde yine aynı kriterler bulunacaktır.

Turing testini düzenleyen ben olsaydım, şartlarına daha fazla esneklik getirmeye çalışırdım. Bilgisayardan, bazı bakımlardan, bir insanı ondan ayırt edilemiyecek kadar iyi taklit etmesini istemek doğrusu ondan gereğinden fazlasını istemektir. Yanıtların temelinde 'bir yabancılık', yapay bir bilinçlilik var olduğu için, bilgisayarın yanıtlarının doğasına bakarak sorgulayıcı gerçekten ikna olabilir. Ben olsam sorgulayıcıdan yanıtların doğasına bakmasını isterdim. Bugüne kadar inşa edilmiş olan tüm bilgisayar sistemlerinde açıkça görülen eksiklik budur. Ancak, sorgulayıcı, bilgisayarın bilinçsiz yanıtında bile bir bilinçlilik algılayabilir ve bu bilinçliliği bilgisayara atfetmek konusunda isteksiz davranabilir. Böyle bir tehlike olasılığını kabul ediyorum. Diğer taraftan, sorgulayıcı, gerçekte var olmasa bile, böyle bir 'yabancılık' duygusuna kapılarak istemeden de olsa bilgisayara kuşkulanma fırsatı verebilir. Bu nedenle Turing testinin ilk versiyonu, daha nesnel olmasından kaynaklanan daha büyük bir avantaja sahiptir ve ben de sonraki bölümlerde bunun üzerinde duracağım. Önceden değindiğim gibi bilgisayara yapılan haksızlık (yani, testte başarılı olmak için bir insanın yapabileceği her şeyi yapmasının beklenmesi, oysa insanın bir bilgisayarın yapabildiğini yapmak zorunda olmaması) düşünme yeteneği, vs. saptamak konusunda Turing testinin etkinliğini destekleyenleri rahatsız eder gibi görünmüyor. Turing testinin taraftarlarına göre bilgisayar çok geçmeden -diyelim 2010 yılına kadar- Turing testinden başarıyla geçebilecektir (Turing, başlangıçta 'vasat' bir sorgulayıcı ve sadece beş dakikalık bir sorgulamayla, 2000 yılına kadar, yüzde 30 başarı oranı öngörmüştü). Sonuçlara bakarak Turing taraftarları, bu taraf tutan görüşün, öngörülen tarihi geciktirmeyeceğine inanıyorlar!

Bütün bunlar temel bir soruyla yakından ilişkilidir: İşlevsel bakış açısı, nesnel bir alemin ussal özelliklerinin varlığına veya yokluğuna dair hüküm vermek için makul bir kriterler sistemini gerçekten sağlar mı? Bazıları, sağladığını savunuyor. Düş gücü, ne kadar yetenekli olursa olsun,

gerçeğin yerini tutamaz.

Benim konumum ikisinin arasında kalıyor. Ne kadar yetenekli olursa olsun, düş gücünün yeterli düzeyde ve beceriyle yapılacak bir sondajla saptanmasının yerinde olacağına genel ilke olarak inanmak eğilimindeyim. Gerçi bu, kanıtlanmış bir olgudan çok bir inanış (veya bilimsel iyimserlik) sayılabilir. Böylelikle, Turing testini, kendi bütünlüğü içerisinde olmadıkça geçerli bir sistem olarak kabul etmeye genelde hazırım. Açıkçası, bir bilgisayar kendisine yöneltilen tüm soruları bir insanınkinden ayırt edilemeyecek -ve böylece idrak yeteneğine sahip sorgulayıcımızı gerektiğince[II] ve sürekli aldatacak- tarzda yanıtlayabilirse, *aksine bir kanıt olmadığı sürece,* bu bilgisayar gerçekten düşünüyor, hissediyor, vs. kanısına varabilirim.

Düşünme, hissetme veya anlam veya özellikle, *bilinçlilik* ile ilgili açıklamalarımda 'kanıt' 'gerçekten' ve 'varsayım/tahmin' sözcüklerini kullandığım zaman bunların, fiziksel nesnelerdeki varlık veya yakınlıklarını kanıtlamaya çalıştığımız gerçek objektif 'nesneler' anlamında kavramlar olduklarını imâ ediyorum, yalnızca anlatım dilinin gereği olarak kullanmıyorum! Bunu son derece önemli bir konu kabul ediyorum. Bu gibi özelliklerin varlığını saptamaya çalışırken, elimizdeki tüm kanıtlara dayanarak tahmin yaparız (İlke olarak, bu durum, örneğin bir astronomun uzak bir yıldızın kütlesini tahmin etmeye çalışmasından farklı değildir).

Başka ne çeşit bir aksi kanıt düşünülebilirdi ki? ileriye yönelik kurallar koymak zor olmakla birlikte ben bir noktayı açıklamak istiyorum: Bilgisayarın, nöronlar, kan damarları, vs. yerine transistörlerden, tellerden, vs. yapılmış olması gerçeği, tek başına, aksi kanıt sayabileceğim türde bir şey değildir. Benim düşlediğim şey, gelecekte bir gün, bilinçlilik ile ilgili başarılı bir teorinin geliştirilebileceğidir -tutarlı ve uygun bir fizik teorisi olması bakımından başarılı, fiziksel anlayışın geriye kalan kısmıyla güzel bir şekilde uyumlu olduğu için başarılı; öyle ki öngörüleri, insanların kendileriyle ilgili yanıt aradıkları ne zaman, nasıl, ne dereceye kadar bilinçli oldukları sorularıyla ilgili iddialarıyla tamamen uyumlu bir teori. Böyle bir teori, bilgisayarımızın varsayılan bilinçliliği ile ilgili ipuçlarını gerçekten verebilir. Böyle bir teorinin ilkelerine uygun inşa edilen bir 'bilinçlilik detektörünü' bile düşleyebiliriz. İnsan üzerinde son derece güvenilir olan böyle bir detektör, bilgisayar söz konusu olduğunda, Turing testinin sonuçlarından farklı sonuçlar verebilir. Bu koşullarda, Turing test sonuçlarının yorumlanması konusunda çok dikkatli olmak gerekir. Bana öyle geliyor ki, kişinin Turing testinin uygun olup olmadığı konusuna bakış açısı, kısmen, bu kişinin bilim ve teknolojinin nasıl gelişeceğine dair görüşlerine bağlıdır. Bu konuya daha sonra tekrar döneceğiz.

Yapay Zekâ

Çoğu kez kısaca Al (Artificial Intelligence) olarak anılan Yapay Zekâ son yılların en çok ilgi çeken konusudur. Al'ın amaçları, makineler, normalde elektronik makineler, aracılığıyla insanın ussal etkinliğini olabildiğince taklit etmek ve belki sonuçta insanın ussal etkinlik yeteneğini geliştirmektir. Al sonuçlarına en az dört alanda ilgi duyulmaktadır. Bunlardan birisi *robotik'*tir. Robotik, insanın müdahalesini veya kontrolünü gerektiren çok çeşitli ve karmaşık görevleri, 'zekâ gerektiren' görevleri, insan yeteneğinin ötesinde bir hız ve güvenilirlikle yerine getirebilen, veya insan hayatının tehlikeye girebileceği olumsuz koşullarda görev alabilen mekanik cihazların endüstrinin uygulamadaki gereksinmelerine cevap vermek amacıyla üretim çalışmalarının yapıldığı bir sektördür. Genel olduğu kadar ticari açıdan da Al amaçlarına ilgi duyulan bir diğer alan *uzman sistemlerdir.* Bu sistemlere göre, bir meslek grubunun -tıp, hukuk, vs - tüm temel bilgileri bir bilgisayar paketi halinde

kodlanmalıdır! Bu meslek üyelerinin deneyim ve uzmanlıklarının yerini bilgisayar programı paketlerinin alması mümkün müdür? Yoksa, yalnızca, gerçeklerle ilgili bilgilerin, geniş kapsamlı bir bilgi-kaynak listesiyle birlikte, uzun listeler halinde hazırlanması mı amaçlanmaktadır? Bilgisayarların insan zekâsının yerini alabileceği (veya taklit edebileceği) konusunun önemli ölçüde sosyal karmaşa yaratacağı açıktır. Al'ın doğrudan ilgili olduğu bir diğer alan *psikolojidir*. İnsanın (veya bir başka hayvanın) beyninin davranışlarının elektronik bir cihaz vasıtasıyla taklit edilmeye çalışılmasıyla -veya bunun gerçekleştirilememesiyle-, beynin işlevleri hakkında önemli bilgiler edinilebilir. İyimser bir yaklaşımla Al, felsefenin derin soruları hakkında, *us* kavramının anlamına sezgi yoluyla nüfuz etmemizi sağlayarak, bize bir şeyler anlatabilir.

Al bugüne değin ne kadar ilerleme kaydetmiştir? Bu konuda bir özetleme yapmak benim için zor olurdu. Dünyanın çeşitli yerlerinde bu konuda çalışan pek çok grubun çalışmalarının pek azmin ayrıntıları hakkında bilgi sahibiyim. Ancak şu kadarını söyleyebilirim ki, gerçekten zekice pek çok çalışma yapılmış olmasına karşın, insan zekâsının yerini tutacak herhangi bir simülasyonun gerçekleşmesi için kat edilecek daha çok yol vardır. Konuya çeşni katması amacıyla eski (yine de etkileyici) çalışmalardan ve günümüzde satranç bilgisayarlarında kaydedilen gelişmelerden bahsedeceğim.

İlk AI cihazlarından birisi, 1950'lerde yapılmış, W. Grey Walter'ın "kaplumbağa'sı idi.[3] Bu cihaz, pilleri bitinceye kadar yerde dolaşıp durur, sonra, pillerini doldururdu. Pilleri tamamen şarjlanınca, fişini prizden çeker, odanın içindeki yürüyüşüne devam ederdi! O günden bugüne, benzer birçok cihaz imâl edildi (örneğin, bkz. Waltz 1982). Oldukça farklı bir çizgide gelişmenin örneği 1972'de Terry Winograd'ın bilgisayar programında görüldü. Simülasyon cihazı, farklı biçim ve renkteki blokları farklı sıra ve dizilişte birbirinin üstüne, akla uygun şekilde yerleştirebiliyordu. Söz konusu ilk başarılı çalışmalara karşın, o zamanlar, basit eklemli bir 'robot kolunun' kumandasını bile tasarımlamanın ne kadar zor gerçekleştiği anımsanmalıdır. Al yöntemlerinin henüz hiçbir etkinlik gösteremediği alanlardan birisi, insan (veya hayvan) beyninin 'çaba sarfetmeden' yapabildiği bir işlem, yani görsel sahnelerin yorumlanması işlemidir.

Belki de Turing testi ile daha doğrudan ilişkili bir bilgisayar programı 1960'larin ortalarında K.M. Colby tarafından geliştirilmiş ve yapay fizyoterapist olarak o kadar başarılı olmuştur ki bazı hastalar dertlerini daha rahat anlatabildikleri gerekçesiyle gerçek terapistin yerine bilgisayarı tercih etmişlerdir. Bilgisayar 'terapist' ile bir hasta arasındaki konuşma (teleyazıcı vasıtasıyla yapılmıştır) örnek olarak aşağıda verilmiştir:[4]

TERAPİST: Size nasıl yardım edebilirim?

HASTA: Bilmiyorum.

TERAPİST: Bana sorununuzdan bahsetmeye çalışın.

HASTA: Olmaz.

TERAPİST : Biraz ayrıntı veremez misiniz?

HASTA: Bugün hasta ve yorgunum.

TERAPİST: Sebebi hakkında bir fikriniz var mı?

HASTA: Evet, yaşlıyım.

TERAPİST: Lütfen devam edin.

HASTA: Hiçbir şeye ilgi duymaz oldum.

TERAPİST: Biraz daha açıklayın?

HASTA: Neden böyle hissettiğimi sanırım biliyorum.

TERAPİST: Sahi mi.

HASTA: Sanırım bir aile özlemi çekiyorum.

TERAPİST: Aile özlemini size hissettiren nedir?

HASTA: Bunu açıklaması zor.

TERAPİST: Sizin için aile niçin önemli?

Bu diyalog, bilgisayarın aslında hiç olmayan anlama yeteneğine bir ölçüde sahip olduğu izlemini veriyorsa da, gerçekte bütün yaptığı oldukça basit mekanik kurallara uygun davranmaktır (Bilgisayar simülasyon programlarında, şizofrenik bir hastayı simüle eden bir 'diyalog' daha vardır. Bu diyalogda tüm ders kitaplarında bulunan cevaplar ve semptomlar yer almakta, diyaloğu dinleyen tıp öğrencilerini bilgisayar, yanıtların 'insan hasta' tarafından verildiğine inandırmaktadır!).

Satranç-oynayan bilgisayarlar "zeki davranış" olarak nitelenebilecek davranış sergileyen makinelerin belki en iyi örneğidir. Gerçekte bu makineler, bugünlerde (1989'da) 'Uluslararası Usta' düzeyinde performans göstermektedirler (Bu bilgisayarların puanları 2300'ün biraz altında olup, kıyaslama amacıyla bir değer verilmesi gerekirse, dünya şampiyonu Kasparov'un puanı 2700'ün üzerindedir). Özellikle Dan ve Kathe Spracklen tarafından hazırlanan bir program (Fidelity Excel ticari mikroprosesörü için) 2110'luk bir puana (Elo) ulaşarak USCF 'Usta' ünvanını kazanmıştır. Daha da etkileyici bir program 'Derin Düşünce', Carnegie Mellon Üniversitesinden Hsiung Hsu tarafından hazırlanmış ve yaklaşık 2500 Elo puan almıştır. Bu satranç programı yakın zamanlarda, Büyük Usta Bent Larsen'i ilk kez yenerek Kasım 1988'de Longbeach, Kaliforniya'da düzenlenen bir satranç turnuvasında birinciliği Büyük Usta Tony Mitos ile paylaşmıştı![5] Satranç bilgisayarları artık satranç *problemlerini* de çözmekte ve bu konuda insan rakiplerini kolaylıkla alt etmektedirler.[6]

Satranç-oynayan makineler, doğru hesaplama gücünün yanı sıra 'kitap bilgisi'ne de büyük ölçüde bağımlıdırlar. Kabul etmek gerekirse, özellikle çabuk hamle yapılmasının gerekli olduğu durumlarda bu makineler, insan satranççıya kıyasla genelde daha iyi performans göstermektedir. Çünkü, bilgisayarın kuralları, kesin ve hızlı hesaplama esasına göre alınırken, insan satranççı 'karar verme'nin avantajından yararlanmak yoluna gider ve bu işlem yavaş ve bilinçli bir değerlendirme yapmak demektir. İnsan yargıları, hesaplamanın her aşamasında, analizle gerçekleşebilenden daha fazla derinliğine dikkate alınması ciddi olasılıkların sayısını önemli ölçüde azaltırken makine, karar vermek kaygısı olmaksızın olasılıkları hızla hesaplar ve doğrudan elimine eder (Bu fark, zor bir Uzak Doğu oyunu olan 'go' oyununda daha belirgindir, çünkü bu oyunda hamle başına düşen olasılık sayısı satrançtakinden fazladır). Bilinçlilik ve kararların oluşturulması arasındaki ilişki sonraki bölümlerin, özellikle X. Bölüm'ün ana konusudur.

'Haz' ve 'Acı'ya Al Yaklaşımı

Yapay zekânın, mutluluk, acı, açlık gibi ussal özelliklerin bir çeşit açıklamasını sağlayacak bir yol olduğu öne sürülmektedir. Grey Walter'ın kaplumbağasını ele alalım: Pili bittiği zaman davranış biçimi değişiyor, enerji deposunu yenilemeye tasarımlandığı şekilde davranmaya başlıyordu. Bir insanın -veya herhangi bir başka hayvanın- kendini aç hissettiği zamanki davranışı ile Walter'ın

kaplumbağasının davranış biçimleri arasında açıkça görülen benzerlikler vardır. Grey Walter'ın kaplumbağasının 'acıktığı' zaman aynı davranışı gösterdiğini söylersek, buna hiç de dil sürçmesi denilemez. Kaplumbağanın içindeki bir mekanizma pilin şarj durumuna duyarlı olduğu, için, enerji belirli bir noktanın altına indiği zaman kaplumbağaya farklı bir davranış biçimine sahip olacak şekilde komut veriyor. Kuşkusuz hayvanların bedenlerinde buna benzer bir mekanizma var; fakat hayvanlarda davranış biçimindeki değişiklikler biraz daha karmaşık ve anlaşılması zor. Bir biçimden diğerine bir kumanda mekanizması sayesinde geçmek yerine, belirli tarzda hareket etme *eğiliminde* bir değişiklik oluşmakta, bu değişiklikler, enerji kaynağını yenileme gereksinmesi arttıkça daha da güçlenmektedir (bir noktaya kadar).

Aynı şekilde, Al taraftarları, acı ve mutluluk gibi kavramların da bu şekilde örneklenebileceğini öngörmektedirler. Konuyu basite indirgeyelim ve 'duyguları', aşırı 'acı'dan (puan: -100) aşırı 'hazza' (puan: +100) kadar tek Ölçeğe indirelim. Herhangi bir çeşit makinemiz, diyelim elektronik makinemiz kendi 'haz/acı' puanının kaydını yapabilecektir. 'Haz/acı' puanını kısaca ha-puanı olarak adlandıralım. Elektronik cihazımız belirli davranış kipleri ve dahili (pilleri gibi) veya harici bazı girdilerle donanımlıdır. Hareketleri, ha-puanını en üst düzeye çıkarabilecek şekilde tasarımlanmıştır. ha-puanını etkileyen birçok etken bulunabilir. Pillerinin düşük şarjlı olması olumsuz bir etken, yüksek şarjlı olması olumlu bir etkendir. Kuşkusuz bu etkenleri dilediğimiz gibi ayarlayabiliriz, fakat başka etkenlerin var olduğunu unutmamalıyız. Belki cihazımızın üzerinde, alternatif bir enerji kaynağı olarak güneş enerjisi depolayan panolar bulunabilir. Böylece, panolar devredeyken, pillerin kullanılması gerekmez. Gün ışığına yaklaştırıldığında ha-puanını biraz artırabilir, böylece diğer etkenlerin var olmaması durumunda bu davranış eğilimini göstermesini sağlayabiliriz (Aslında, Grey Walter'in kaplumbağası daima ışıktan *kaçınırdı*!). Farklı hareketlerinin ha-puanı üzerindeki olası etkilerini kontrol amacıyla cihazımız gerekli hesaplamaları yapabilecek donanıma da sahip olmalıdır. Verilerinin güvenilirliğine bağlı olarak ha-puanı üzerindeki büyük/küçük etkilerin hesaplanmasında olasılık katsayısını uygulayabilmelidir.

Cihazımızı, enerji kaynağını sürekli yenilemenin dışında başka 'amaçlarla da donatmalıyız; aksi durumda, 'acı'yı 'açlık'tan ayırt edemeyiz. Şimdilik seks söz konusu olmadığı için, cihazımızın üreme sistemiyle de donatılmasını istemek biraz fazla olur! Fakat belki benzer cihazlarla arkadaşlık kurabilmesi için bedenine bir 'dostluk istek' mekanizması monte ederek arkadaşlarıyla bir araya geldiği zaman pozitif ha-puanını artırmasını sağlayabiliriz. Veya onu kendi adına öğrenme 'aşkı' ile donatabiliriz; böylece dış dünya ile ilgili gerçekleri hafızasına depolayarak ha-puanını olumlu etkileyebilir. (Biraz daha bencilce davranarak cihazımızı, bizim için bir hizmet yaptığı zaman olumlu puan alacak şekilde programlayabiliriz -aynı bir robot hizmetkâr inşa ettiğimiz zaman yapmamız gerektiği gibi!) Bu gibi 'amaçları', aklımıza estiği gibi cihazımıza empoze etmemizin yapaylık olduğunu ileri sürenler bulunabilir. Fakat bu, doğal seçimin bireyler olarak bize, büyük ölçüde, genlerimizle çoğalma ihtiyacıyla yönetilen belirli 'amaçları' empoze etmesinden farklı değil ki.

Cihazımızın, öngörülen şekilde başarıyla monte edildiğini varsayalım, ha-puanı artı olduğu zaman haz *duyduğunu* ve ha-puanı eksi olduğu zaman acı *hissettiğini* nasıl kanıtlayacağız? Al (veya işlevsel) görüşe göre, davranışına bakarak bu konuda bir hükme varabiliriz. Puanını olabildiğince yüksek bir artı değere çıkaracak (ve olabildiğince uzun artı kalacak) şekilde davrandığı ve buna karşılık yine eksi puanlardan olabildiğince sakındığı için mantıklı bir sonuca vararak, haz hissini puanının artılık derecesi olarak, acı duygusunu da puanının eksilik derecesi olarak *tanımlarız*. Böyle bir tanımın 'mantığı', haz ve acı duyguları ile ilgili olarak bir insanın da tamamen aynı reaksiyonu göstereceği gerçeğinden kaynaklanabilir. Kuşkusuz, insanlar söz konusu olduğunda, hepimiz biliriz,

durum bu kadar basit değildir: Bazen bilerek isteyerek acıdan kaçınmaz, bazen hazdan kaçınmak için yolumuzu değiştiririz. Bizim hareketlerimizi çok daha karmaşık kriterler yönlendirir (Bkz. Dennett 1978, s. 190-229). Fakat kabaca bir ortalamayla, acıdan kaçınmak ve haz peşinde koşmak bizim gerçek davranış biçimimizdir. Bir işlevselci için bu durum, aynı düzeyde bir ortalamayla, cihazımızın ha-puanının acı/haz oranı ile *tanımlanması* konusunda yeterli gerekçe teşkil eder. Bu gibi tanımlamalar Al teorisinin amaçları arasında da yer alır.

Şimdi sormalısınız: Cihazımızın, ha-puanı eksi olduğu zaman acı, artı olduğu zaman haz *duyması* gerçekten söz konusu mu?

Gerçekten, cihazımız herhangi bir şey hisseder mi? İşlevselci, kuşkusuz, ya "Tabii ki evet" der veya anlamsız olduğu gerekçesiyle bu çeşit soruları reddeder. Fakat, bana kalırsa, burada dikkate alınması gereken ciddi ve zor bir soru *var*. İnsanlar için dürtüler çok çeşitlidir. Acı ve haz gibi bazıları bilinçlidir; fakat bazı dürtülerin doğrudan doğruya farkına varamayız. Kızgın sobaya elini süren bir insanda, acı hissini henüz duymadan önce elini çekmesine neden olan istemsiz bir hareket oluşur. Bu tip istemsiz hareketler, acı ve hazzın gerçek etkilerinden çok, cihazımızın ha-puanına karşı tepkilerine daha yakındır.

Makinelerin davranışlarını tanımlarken antropomorfik, çoğu kez şaka yollu terimler kullanırız: 'Bu sabah arabamın canı çalışmak istemiyor' veya 'Saatim hâlâ Kaliforniya saatine göre işlediğini sanıyor'; veya 'Bilgisayarım son komutu anlamadığını, bir sonraki aşamada ne yapması gerektiğini bilmediğini iddia ediyor.' Elbetteki arabamızın *gerçekten* bir şey *isteyebileceğini,* veya saatimizin *sanabileceğini,* veya bilgisayarımızın *iddia edebileceğini,* veya hatta *anlayabileceğini* ima etmiyoruz. Bu çeşit tanımlamalar, onlara gerçek iddialar gözüyle bakmadığımız, yalnız kastettikleri espri içerisinde baktığımız sürece gerçekten tanımlayıcı ve anlamamıza yardımcı olacaklardır. Üretilen bilgisayarlarda da, amaçlanan espri *ne olursa olsun,* ussal özellikler var olabileceğine dair iddialara yaklaşımım aynıdır. Grey Walter'ın kaplumbağasının acıkabileceğini söylediğim zaman bunu yarı-şaka olarak söylerim. Bir cihazın ha-puanı ile ilgili 'acı' veya 'haz' gibi ifadeler kullandığım zaman, kendi davranışımı ve ussal durumumla bazı benzerlikler nedeniyle cihazın davranışını anlamama yardımcı oldukları için bu ifadeleri kullanırım; davranış benzerliklerimizin gerçekten birbirine yakın olduğunu ima amacıyla veya davranışımı etkileyen başka bilinçsiz şeylerin varlığını yadsıdığımı ima amacıyla kullanmam.

Ussal özellikleri anlayabilmemiz için Al'ın bize doğrudan sağlayabileceğinden fazlasına ihtiyacımız olduğunu umarım okuruma yeterince açıklayabildim. Ancak Al'ın, önemle ele alınması ve incelenmesi gereken ciddi bir konuyu ortaya koyduğuna inanıyorum. Gerçek zekânın simülasyonu konusunda bugüne kadar pek fazla şeyin gerçekleştirilmemiş olduğunu söylemiyorum. Fakat unutulmamalıdır ki yapay zekâ konusu henüz çok yenidir. Bilgisayarlar hızla gelişecek, hızlı giriş yapılabilen daha geniş hafızalara, daha fazla sayıda mantık ünitelerine sahip olacaklar ve buna paralel olarak daha fazla sayıda işlem gerçekleştireceklerdir. Mantık tasarımında ve programlama tekniğinde ilerlemeler olacaktır. Al felsefesinin araçları olan bu makinelerin teknik kapasiteleri son derece gelişecektir. Üstelik felsefenin özünde absürdite yoktur. Belki insan zekâsı, şimdiden bilinen ilkelere, bugünün bilgisayarlarına dayalı fakat gelecekte daha da geliştirilecek kapasite, hız, vs. sahip olan elektronik bilgisayarlarla doğru şekilde simüle edilebilecektir. Hatta, belki, bu makineler gerçekten zeki olacaklardır; belki düşünecekler, hissedecekler ve usa sahip olacaklardır. Veya belki de böyle olmayacak ve bugün için hiç farkında olmadığımız yeni bir ilkeye gereksinim duyulacaktır. İşte konumuz budur ve hiç de hafife alınacak bir konu değildir. Görebildiğim kanıtları sunmaya çalışacağım. Sonunda kendi görüşlerimi açıklayacağım.

Güçlü Al ve Searle'ın Çin Odası

Konumuzun ilgi alanında önemli bir yere sahip[7] ve *güçlü-AI*. olarak adlandırılan görüşe göre yukarıda tanımlanan bilgisayarlar yalnız zeki değil aynı zamanda usa vs. sahiptir. Fakat sahip oldukları bu bir tür ussal özellikleri *herhangi bir* hesap makinesinin, hatta termostat gibi[8] en basit mekanik ekipmanın mantık fonksiyonuna benzetilebilir. Ussal etkinlik çoğu kez *algoritma* adıyla anılan ve iyi tanımlanmış işlemler sırasının uygulanmasından ibarettir. Algoritma konusuna daha ayrıntılı olarak sonra değineceğim. Şimdilik, algoritmayı bir çeşit hesap yöntemi olarak tanımlamak yeterli olacaktır. Termostat örneğinde algoritma son derece basittir: Cihaz, sıcaklığın önceden ayarlanmış olduğu değerden yüksek veya düşük olup olmadığını kaydeder, yüksek olması durumunda devre kesilir, düşük olması durumunda ise açılır, insan beyninin herhangi bir kayda değer ussal işlemi söz konusu olduğunda algoritma çok daha karmaşıksa da, güçlü AI'a göre, yine de bir algoritmadır. Termostatın algoritması kadar basit olmayabilir fakat ilkede farklı olması gerekmez. Bu nedenle, güçlü AI'a göre, insan beyninin temel fonksiyonlarını yerine getirmesi (bilincinin tüm açığa vurumları dahil) ile termostatınkinin arasındaki fark, beynin çok daha fazla *karmaşık* olmasından (veya belki 'daha mükemmel düzenlenmiş yapısından', veya 'kendini tanımlayabilme özelliklerinden', bir algoritmadan beklenen diğer bazı özelliklerden) kaynaklanır. Daha önemlisi, tüm ussal özellikler -düşünme, hissetme, zekâ, anlayış, bilinç- söz konusu karmaşık fonksiyon sisteminin yalnız birer parçasıdır; bir başka deyişle, beyin tarafından gerçekleştirilen *algoritma*'nın özellikleridir.

Herhangi bir spesifik algoritmanın özgün özelliği, performansında, sonuçlarının doğruluğunda, kapsamında, ekonomik oluşunda ve uygulama hızında kendini gösterir. İnsan beyninde faaliyet gösterdiği varsayılan algoritmaya eş bir algoritmanın olağanüstü bir şey olması gerekir. Fakat bu çeşit bir algoritma beyinde varsa -güçlü Al yandaşları var olduğunu kuşkusuz iddia edeceklerdir- bir bilgisayara da ilke olarak uygulanabilir demektir. Gerçekten de, veri saklamak için yeterli bellek ve çalışma hızı gibi sınırlamalar olmasa, *herhangi bir* modern genel-amaçlı elektronik bilgisayarda bu algoritma kullanılabilir (Bu görüşün gerekçesi, ileriki bölümlerde evrensel Turing makinesi kapsamında ayrıntılandırılacaktır). Çok uzak olmayan bir gelecekte, daha geniş kapasiteli ve hızlı işlemcili bilgisayarlarda bu çeşit sınırlamaların söz konusu olmayacağı ümit edilmektedir. Bu gerçekleşirse, böyle bir algoritma Turing testinde başarılı olabilir. Güçlü Al savunucularının iddiasına göre algoritmayı, işlerlik kazanması bağımsız kılacaktır: Kendi duyguları ve bilinci olacaktır ve böylece bir us olacaktır.

Ussal etkinliklerin ve algoritmaların bu çerçevede birbiriyle özdeşleştirildiklerini herkes tartışmasız kabul edebilir. Özellikle, Amerikalı felsefeci John Searle (1980, 1987), amaca uygun programlanmış bir bilgisayarın Turing testinin basitleştirilmiş versiyonlarında başarılı olduğuna dair örnekler vermekte, ancak yine de 'anlama' işlevinin hiç bulunmadığına dair güçlü kanıtlar göstermektedir. Benzer bir örnek, Roger Schank tarafından tasarımlanan bir bilgisayar programına dayanılarak verilmektedir (Schank ve Abelson 1977). Programın amacı, basit öyküleri anlama yoluyla benzetişimi sağlamaktır: 'Adamın biri lokantaya girdi ve bir hamburger ısmarladı. Hamburger geldiğinde yanmış ve sertleşmişti. Adam faturayı ödemeden, bahşiş bırakmadan öfkeyle çıkıp gitti.' İkinci öykü: 'Adamın biri lokantaya girdi ve bir hamburger ısmarladı. Hamburger geldiğinde, çok hoşuna gitti. Lokantadan çıkarken faturayı ödemeden önce garson kıza çokça bahşiş verdi.' Öyküleri ne ölçüde 'anladığını' denemek için bilgisayara her bir öyküde adamın hamburger yiyip yemediği sorulur (her iki öyküde de açıkça belirtilmeyen olay). Böyle basit öykü ve soruya

bilgisayar, İngilizce bilen bir insanın birinci öykü için 'hayır' ikinci öykü için 'evet' yanıtından temelde ayırt edilemiyen yanıtlar verebilir. Böylelikle bir makine, çok sınırlı anlamda bir Turing testinde başarılı olmuş sayılır!

Dikkate almamız gereken soru, bu tür bir başarının bilgisayar adına -veya, belki, programın kendisi adına- gerçek bir anlama yeteneğini yansıtıp yansıtmadığıdır. Searle'ın bu soruya olumsuz yanıtı, onun 'Çin odası' kavramını davet ediyor. Searle, her şeyden önce, öykülerin İngilizce değil diyelim Çince anlatılmasını ve bilgisayarın algoritmasının tüm eylemlerinin, üzerinde Çin sembolleri bulunan bir komutlar dizisi şeklinde verilmesini öngörmektedir. Searle, bütün işlemlerin, kilitli bir odada gerçekleştirdiğini hayal etmektedir. Öyküleri temsil eden semboller ve daha sonra sorular küçük bir delikten içeri verilir. Dışardan içeriye başka hiçbir bilginin girmesine izin verilmez. Tüm test materyali bu şekilde odaya verildikten sonra değerlendirme sonuçları doğru şekilde ve manuel yöntemle sıralanarak yine aynı delikten odanın dışına alınır. Schank'ın programının algoritmasının uygulanmasından ibaret olan bu işlem sonucu çıkacak olan, Çince öyküye Çince "evet" veya "hayır" yanıtıdır. Searle, bir tek kelime bile Çince bilmediğini, öykülerin konusu hakkında, bu nedenle, en ufak bir fikri dahi olmadığını açıkça ifade etmektedir. Yine de Schank'ın algoritmasını oluşturan işlemleri sırasıyla uygulayarak (algoritma ile ilgili komutlar kendisine İngilizce olarak verilmişti), öyküleri kolayca anlayabilecek bir Çinli kadar başarılı olmuştur. Searle'ın görüşü -sanırım çok güçlü bir görüş- şudur: Bir algoritmayı başarıyla uygulamak, 'anlama' işleminin gerçekleşmesi demek değildir. Çin odasında kilitli kalmış ve tek sözcük Çince bilmeyen Searle'ı düşünürseniz hak vereceksiniz!

Searle'ın savına karşı çıkanlar çok olmuştur. Burada, önemli bulduğum birkaçına değineceğim. Önce, yukarıda geçen 'tek bir sözcük bile anlamama' deyişinin belki biraz yanlış yönlendirici olduğunu belirtmek istiyorum. Anlamanın, tek tek sözcüklerle olduğu kadar sözcük biçimleriyle de ilişkisi vardır. Bu çeşit algoritmaların uygulanması esnasında, bireysel sembollerin pek çoğunu gerçek anlamlarını anlamaksızın, bu sembolün oluşturulduğu biçimlerden, desenlerden bir şeyler çıkarabilirsiniz. Örneğin, hamburger karşılığı Çin harfi (eğer böyle bir harf varsa) yerine, diyelim başka bir yiyecek olan 'chow mein'in sembolünü koyalım: Öyküler böyle bir değişiklikten fazla etkilenmez. Bana öyle geliyor ki, doğrusunu söylemek gerekirse, algoritmanın ayrıntılarına saplanıp kalırsanız öykülerin gerçekten neyi anlatmak istediklerini anlayamazsınız (bu durumda hamburger yerine 'chow mein'in kullanılmasını bile önemsiz bulursunuz).

İkinci olarak değinmek istediğim nokta, oldukça basit bir bilgisayar programının bile, insanların kullandığı sembollerle uygulandığı takdirde son derece uzun ve sıkıcı hale gelebileceği gerçeğinin gözardı edilmemesi gerektiğidir (Ne de olsa bilgisayarları böyle işlemleri bizim adımıza yapmaları için kullanmıyor muyuz!). Searle, Schank'ın algoritmasını, önerdiği şekilde gerçekten uygulamış olsaydı, bir tek soruyu yanıtlamak için günler, aylar veya yıllar boyu süren sıkıcı bir işe girişmiş olacaktı -ki bu, bir felsefeci için hiç uygun olmazdı! Ancak, biz denemenin uygulanabilirliği ile değil *ilkesi* ile ilgilendiğimiz için, bu itirazı ciddi bir eleştiri olarak dikkate almıyorum. Asıl zorluk, insan beyninin karmaşıklığına eş karmaşıklıkta olduğu varsayılan bir bilgisayar programının Turing testinden *başarıyla* geçmesidir. Turing testini geçmesi öngörülen bir programın son derece karmaşık olması gerekir. Oysa, Searle'ın programının, basit bir Turing sorusunu yanıtlamak için bile o kadar çok aşamadan geçmesi gerekir ki bir insanın normal yaşam süresi içerisinde algoritmayı elle uygulaması mümkün değildir. Searle'ın programı gerçekte var olmadığı için bunu kanıtlamak da zordur.[9] Algoritmanın, ussal özellikleri yansıtabilmesi için 'kritik' ölçüde karmaşıklığa sahip olması zorunludur. Belki bu kritik değer öylesine büyüktür ki, bu ölçüde karmaşık olmayan hiçbir

algoritma, Searle'ın düşlediği gibi, herhangi bir insan tarafından uygulamayla gerçekleştirilemez.

Bu zorluğun farkında olan Searle, Çin odasına, yalnız kendisinin girmesinden vazgeçerek, Çince bilmeyenlerden oluşan bir ekip yerleştirmeyi kabul etmiştir. Hatta, Çince sembollerden oluşan verilerin olabildiğince çok sayıda sağlanabilmesi için tüm Hindistan'ı test odası olarak kullanmayı ve (Çince bilenler dışında!) tüm Hindistan halkını Çin sembollerini değerlendirmek üzere görevlendirmeyi düşlemiştir. Pratikte komik bir uygulama olsa da, *ilke* olarak komik değildir, çünkü sav aynı savdır: Sembolleri değerlendirenler öyküyü anlamadan değerlendirirler. Nitekim güçlü Al savına göre, yalnızca uygun algoritmanın uygulanması, ussal özelliklerinden 'anlama' yeteneğini ortaya çıkarmaya yeterlidir. Ancak şimdi bir başka itirazla çember genişlemeye başlıyor. Hintliler bireyler olarak, bir insanın beyninin tümünden çok, beyindeki bireysel nöronlara benzemiyor mu? Düşünme eylemi esnasında beynin fiziksel etkinliğini oluşturan nöronların, kişinin düşüncelerini anladığını hiç kimse iddia edemeyeceğine göre Hintlilerin bireyler olarak Çin öykülerini anlamalarını neden bekleyelim? Searle bu soruyu, ülkede oturanlardan hiçbirinin bireyler olarak anlamadığı bir öyküyü ülkenin, Hindistan'ın kendisinin, anlamasının görünüşteki anlamsızlığına dikkat çekerek yanıtlar. Bir ülke, diye savunur, bir termostat veya bir otomobil gibi, 'anlama işinde' değildir ama bir şahıs birey olarak 'anlama' işlevini yerine getirmelidir.

Bu sav, öncekinden çok daha az güçlü. Sanırım Searle'ın savı, algoritmayı uygulayan sadece bir kişi olduğu zaman en güçlü düzeye ulaşıyor. Bu durumda dikkatimizi, bir kişinin bir ömürden daha az sürede uygulayabileceği düzeyde karmaşık algoritmayla sınırlıyoruz. Searle'ın savının, kişinin algoritmayı uygulayışı ile ilgili olarak, nesnellikten arındırılmış ve varlığı kişinin kendi bilinçliliğini herhangi bir şekilde etkilemiyen bir tür 'anlama' olasılığım *tüm gücüyle* savunduğu kanısında *değilim.* Ancak, söz konusu olasılığın en azından gözardı edildiğine dair görüşüne katılıyorum. Searle'ın sayının, sonuçta tamamen ikna edici olmasa bile dikkate değer gerekçelere dayandığını düşünüyorum. Schank'ın bilgisayar programının içerdiği karmaşıklığa benzer karmaşıklığa sahip algoritmaların, hangi görevleri üstlenirlerse üstlensinler, gerçek anlayışa sahip olamayacaklarının gösterimi oldukça ikna edicidir. Ayrıca, böyle bir gösterim -güçlü Al'ın iddialarının aksine-, ne kadar karmaşık olursa olsun hiçbir algoritmanın gerçek anlama yeteneğini tek başına temsil edemeyeceği konusunda bir *fikir verir* (ama daha fazlasını yapamaz).

Görebildiğim kadarıyla, güçlü Al ile ilgili başka ciddi zorluklar da bulunmaktadır. Güçlü Al için, algoritma ön plandadır. Algoritmanın, bir beyin, elektronik bir bilgisayar, bütün bir Hindistan, çarklardan ve dişlilerden oluşan mekanik bir cihaz veya su boruları sistemi tarafından uygulanmakta olması fark etmez. Temsil ettiği varsayılan 'ussal durum' için önemli olan algoritmanın mantık yapısıdır; kendine özgü fiziksel yapısı tümüyle konu dışıdır. Searle'ın işaret ettiği gibi, bu durum, bir tür 'dualizm'i beraberinde getirmektedir. Dualizm, on yedinci yüzyıla damgasını vurmuş olan felsefe ve matematikçi Rene Descartes tarafından benimsenen felsefi bir teori olup 'tinsel' ve olağan madde olarak iki ayrı tür maddenin var olduğunu savunur. Bu iki ayrı türün birbirini etkileyip etkilemediği veya nasıl etkilediği ayrı bir konudur. Temel fikir tinsel şeylerin maddeden oluşmadığı ve maddeden bağımsız var olabildiğidir. Güçlü Al'ın tinsel içeriği, bir algoritmanın mantık yapısıdır. Biraz önce değindiğim gibi, bir algoritmanın fiziksel yapısı tamamiyle konunun dışındadır. Algoritma, fiziksel yapısından tamamen ayrı olarak nesnellikten arınmış bir tür 'varoluşa' sahiptir. Bu tür bir varoluşu ne kadar ciddiye almamız gerektiğini bir sonraki bölümde açıklayacağım. Nesnellikten arınmış algoritma, soyut matematiksel nesnelerin Platonik gerçeği ile ilgili genel konunun bir parçasını oluşturur. Şimdilik bu genel konuyu bir tarafa bırakarak, güçlü Al taraftarlarının en azından algoritmalar açısından gerçekliği ciddiye aldıklarını, çünkü algoritmaların düşünme, hissetme,

anlama ve bilinçle ilgili algılamalarının 'maddesini' oluşturduğuna inandıklarını belirtmekle yetineceğim. Bu olguda tuhaf bir ikilem, bir ironi ortaya çıkıyor: Searle'ın de ifade ettiği gibi, güçlü Al'ın hareket noktası insanı aşırı bir dualizme doğru yönlendirir gibidir -ki bu, Al savunucularının kendileriyle bağdaştırılmasını asla istemeyecekleri bir olgudur!

Güçlü Al teorisinin yaratıcılarından Douglas Hofstadter'e (1981) ait "Einstein'ın Beyni ile Bir Sohbet" başlıklı bir diyalogda, söz konusu ikilem belirginleşmektedir: Hofstadter, Albert Einstein'ın beyninin eksiksiz bir tanımım içeren, gülünç derecede büyük boyutlarda, bir kitap düşler. Sorulacak her soruya, Einstein hayattaymış gibi, Einstein tarafından yanıtlanıyormuşcasına yanıt almak için, sadece sayfalarını çevirerek, kitabın sunduğu ayrıntılı komutları izlemek yeterli olacaktır. Hofstadter'in özenle belirttiği gibi, 'sadece' sözcüğü kuşkusuz, son derece yersiz kullanılmıştır. Çünkü, *ilke olarak* kitap, Turing ilkesinin işlevsel anlamında, gerçek Einstein'ın gülünç şekilde yavaş-çekim versiyonudur. Böylece, güçlü Al tarafından amaçlandığı üzere kitap, sanki Einstein'ın kendisiymiş gibi fakat belki son derece yavaşlatılmış bir hızda (kitapsı-Einstein'ın gözleri önünden dış dünya, komik derecede hızlandırılan bir süratle geçsin diye) düşünecek, hissedecek, anlayacak, bilecektir. Kitap, Einstein'ın 'özü'nün nesnelleştirilmesi olarak tasarımlandığına göre pekâlâ gerçek Einstein olabilir.

Ama şimdi yeni bir zorluk çıkıyor karşımıza. Kitap hiçbir zaman açılmayabilir veya gerçeği arayan sayısız öğrenci ve araştırmacılar tarafından sürekli kullanılabilir. Aradaki farkı kitap nasıl 'bilecek'? Belki, içindeki bilgiler röntgen ya da tomografi cihazıyla veya başka bir teknolojik sihirbazlıkla okunabildiği için kitabın açılması gerekmeyecek. Einstein, kitap bu şekilde incelenirken mi rolünü oynamaya başlayacak? İki kişi aynı soruyu ayrı zamanlarda sormaya kalkışırsa, rolüne başlaması için iki kez üst üste uyarılmış mı olacak? Yoksa iki ayrı ve geçici olarak farklı anlarda mı uyarı almış olacak? Belki yalnız kitap *değiştirildiği* zaman rolünün farkına varacak? Ne de olsa, normal olarak, bir şeyin farkına vardığımız zaman, dış dünyadan, belleğimizi uyaran, usumuzun özelliklerini hafifçe etkileyen bilgiler alırız. Durum böyleyse, algoritmaların *aktive edilmesinden* çok (veya ek olarak) ussal olaylarla bağdaştırılması gereken şey, algoritmalardaki (bakın işte, hafıza bankasını algoritmanın bir parçası haline getiriyorum) *değişiklikler* (uygun değişiklikler) midir? Veya kitapsı-Einstein hiç kimse veya hiçbir şey elini süremese, hiç incelenmese bile kendi varlığının yine de farkında olur mu? Hofstadter bu sorulardan bazılarına değiniyor ama çoğunu yanıtsız bırakıyor.

Bir algoritmayı aktive etmek veya fiziksel biçimde nesnelleştirmek ne demektir? Bir algoritmanın değiştirilmesi ile bir algoritmanın elden çıkarılarak yerine yenisinin kullanılması arasında fark var mıdır? Bütün bunların, bizim bilinçli olarak bir şeyin farkında olmamızla ne ilgisi olabilir? Okur, (kendisi bir güçlü Al yandaşı olmadığı sürece) saçmalığı açıkça görülen böyle bir konuya niçin bu kadar yer ayırdığımı merak edebilir. Aslında bu konuyu saçma bulmuyorum, temelde yanlış buluyorum! Güçlü Al mantığının gerisinde dikkate alınması gerekli bir dürtü var, ve ben bunu açıklamaya çalışacağım. Bence, bazı fikirlerde -gereğince değiştirilirse- bir ölçüde çekicilik var, ve ben bunu da anlatmaya çalışacağım. Üstelik, kanımca, Searle tarafından dile getirilen aksi görüş, bir ölçüde paylaşsam bile, bazı ciddi bilinmezler ve görünüşte saçmalıklar içermektedir.

Searle, bugünün elektronik bilgisayarının, geliştirilmiş işlem hızı ve hafıza bankasına giriş hızıyla (buna paralel işlemle) çok uzak olmayan bir gelecekte Turing testinden başarıyla geçebileceğini, açıkça olmasa bile, kabul etmektedir. Güçlü Al'ın (ve diğer birçok 'bilimsel' teorinin), "bizler, bilgisayar programlarının örnekleriyiz' iddiasını kabule hazır görünmektedir. 'Beyin dijital bir bilgisayardır kuşkusuz. Her şey dijital bir bilgisayar olduğuna göre, beyin de öyledir' görüşüne boyun eğmektedir.[10] Searle'a göre, insan beyninin (usa sahip olabilir) fonksiyonu ile elektronik

bilgisayarın (usa sahip olamayacağını savunuyor) fonksiyonun arasındaki fark, aynı algoritmayı uygulamaları olası bu ikisinden her birinin sadece maddi yapısından kaynaklanmaktadır. İnandığı fakat açıklayamadığı nedenlerle, elektronik nesnelerin aksine, biyolojik nesne (beyinler), zihinsel etkinliğini belirleyici özelliklerinden 'yönelmişlik' ve 'anlam bilgisine sahip olabilirler. Evrim süreçlerini yansıtan 'tarihsel' boyutları dışında, biyolojik sistemlerin (gerçekte bizler de bu sistemlerde yer alıyoruz) yönelmişliği veya anlambilgisini gerçekleştirilebilen nesneler olarak ayırt edilmelerini sağlayan ne gibi özellikleri vardır? Bu iddia bende dogmatik bir sav izlenimi yaratıyor. Bu sav, güçlü AI'ın, bir algoritmanın uygulanmasıyla bilinçli farkına varma özelliğinin yaratılabileceğine dair savından daha az dogmatik olmayabilir!

Kanımca, Searle ve diğer birçok insan, bilgisayar bilimcileri tarafından yanlış yönlendirilmişlerdir. Ve, buna karşılık bilgisayarcılar, fizikçiler tarafından yanlış yönlendirilmiştir. (Yanlış yönlendirme fizikçilerin kusuru değil. *Onlar* her şeyi bilmiyor ki!) 'her şey dijital bir bilgisayardır' inancı gerçekten yaygınlaşma eğilimindedir. Bunun böyle olmasına niçin, ve belki de nasıl, *gerek olmadığını* bu kitap kapsamında göstermek istiyorum.

Donanım ve Yazılım

Bilgisayar biliminin dilinde *donanım* (hardware) sözcüğü, makineyi oluşturan, bağlantılar dahil, bütün mekanik parçalar (basılı devreleri, transistörler, teller, manyetik bellek alanı, vs.) için kullanılır. *Yazılım* (software) sözcüğü ise bilgisayarın çeşitli programlarını, yazılımım ifade eder. Alan Turing'in önemli buluşlarından birisi, mekanik aksamın belirli bir düzeyde karmaşıklığa ve esnekliğe ulaştırdığı herhangi bir makinenin, benzeri herhangi bir başka makineyle *eşdeğerde*. olduğudur. Eşdeğer kavramı, A ve B makinelerinde, A'ya B'ye özgü bir yazılım verildiğinde bunu B makinesiymişçesine hassaslıkla işleme koyacağı, B'ye verildiği zaman sanki A makinesiymiş gibi hassaslıkla işleme koyacağı şeklinde yorumlanmalıdır. 'Hassaslık' sözcüğünü, (çevirici yazılım yüklendikten sonra yüklenen) girdi karşılığında bilgisayarın fiili çıktısı ile ilgili olarak kullanıyorum; çıktı için gerekli *süreyi* kastetmiyorum. Her iki makinenin işlemleri için gerekli bellek alanı herhangi bir aşamada tükenirse, manyetik teyp, diskler, makaralar, vs. şeklinde bazı harici boş 'bellek' kaynağına (prensipte sınırsız) başvurabileceğini de dikkate alıyorum. Gerçekte, bir görevi yerine getirmek için A ve B makinelerinin gerektirdiği süre farkı, üzerinde durulması gereken bir konudur. Diyelim ki A, belirli bir görevi B'ye göre bin kat daha hızlı yerine getirebiliyor. Veya bir başka görevde B, A'dan bin kat daha hızlıdır. Ayrıca zamanlamaların, kullanılan çevirici yazılımlara bağlı olarak büyük ölçüde değişebileceği unutulmamalıdır. Tartıştığımız konu, işlemlerini makul bir sürede tamamlamak kaygısı içerisinde bu gibi pratik sorunlara ayıracak zamanı olmayanlar için bir "ilke" tartışmasıdır. Burada değinilen kavramlar, bir sonraki bölümde ayrıntılanacaktır: A ve B örneklerinde sözü edilen makineler, *evrensel Turing makineleridir*.

Gerçekte, tüm modern genel-amaçlı makineler, birer evrensel Turing makinesidir. Bu nedenle, tüm genel-amaçlı bilgisayarlar, yukarıdaki anlamda, birbiriyle eşdeğerdedir: Sonuçtaki işlem hızı farkları ve bellek boyutuna ilişkin olası sınırlamalar ilgi alanımızın dışında kaldığı sürece, aralarındaki farklar tümüyle yazılım kapsamına alınabilir. Modern teknoloji, bilgisayarların işlem hızını ve veri depolama kapasitelerini o denli geliştirmiştir ki, "günlük" amaçların çoğunun normalde gerektirdiği hız ve kapasiteye,[III] pratikteki kaygıların hiçbiri herhangi bir sınırlama getirmez ve bu nedenle bilgisayarlar arasındaki teorik eşdeğerlilik, pratik düzeyde de geçerlidir. Görünüşe göre teknoloji,

idealleştirilen bilgisayarlarla ilgili tümüyle akademik tartışmaları, yaşamımızı doğrudan etkileyen konulara dönüştürmüştür!

Anladığım kadarıyla güçlü Al felsefesinin temelinde yatan en önemli etkenlerden birisi, fiziksel bilgisayarlar arasındaki söz konusu eşdeğerliliktir. Donanıma göreli önemsiz (hatta belki tamamen önemsiz) gözüyle bakılırken, yazılım, yani program veya algoritma, yaşamsal önem taşıyan malzeme olarak kabul edilir. Ancak, kanımca, fizik bilimi açısından önemli başka etkenler de vardır. Bu etkenler hakkında bazı kanıtlar vermeye çalışacağım.

Bir insana kişisel kimliğini veren nedir? Bedenini oluşturan atomlar mı? Bu atomları oluşturan elektronların, protonların ve öteki temel parçacıkların özel seçimi mi? Bu soruların olumsuz yanıtı ile ilgili en az iki neden vardır. Birincisi, herhangi bir canlı bedeni oluşturan malzemenin sürekli bir değişim yaşamasındadır. Doğumdan sonra hiçbir yeni beyin hücresinin üretilmediği gerçeğine karşın bu değişim, özellikle beyin hücreleri için geçerlidir. Her canlı hücredeki (beynin, her bir hücresi dahil) çok sayıda atom -ve, gerçekten, bedenlerimizdeki tüm madde- doğumdan başlayarak birçok kez yenilenmiştir.

İkinci neden, kuantum fiziğinden kaynaklanmaktadır -ve doğrusunu söylemek gerekirse, birinci nedenle tuhaf bir çelişki içindedir! Kuantum mekaniğine göre (bkz. VI. Bölüm,) herhangi iki elektronun tamamen özdeş olması zorunludur ve aynı ilke, herhangi iki proton ve herhangi iki temel parçacık için de geçerlidir. İnsan beynindeki bir elektronun yerine bir tuğladaki elektron konulsa, sistem bir bütün olarak, değişiklikten önceki sistemden, ayırt edilemez![11] Aynı durum protonlar, atomlar, moleküller vs. için geçerlidir. Bir insanın bedenini oluşturan bütün malzeme, evinin tuğlalarından alınacak uygun parçacıklarla takas edilse, hiçbir şey değişmez. Bu insanın kendi evinden ayırt edilmesini sağlayan, bireysel parçacıklar değil, parçacıkların tümünün dizilişinden ortaya çıkan biçimdir.

Kuantum mekanik biliminin dışında, günlük yaşantımızda da bunun basit örneklerine rastlanabilir: Elektronik teknolojisi bu kitabı kelime-işlemciler'le yazmama olanak sağlıyor. Bir sözcüğü düzeltmek istesem, örneğin yanlış yazdığım 'gitdim' sözcüğünü 'gittim' olarak düzeltmek istesem ya 'd' harfini siler yerine 't' harfini yazarım veya sözcüğü tümüyle silerek, yerine doğrusunu yazarım. Sözcüğü tümüyle silerek yeniden yazdığım zaman 'g' harfi, yeniden yazmadan önceki 'g' harfi midir? Yoksa yerine bir benzerini mi yazdım? Peki, 'i' harfinden ne haber? Sözcüğün tümünü değiştirmeyip, yalnız 'd' harfinin yerine 't' harfini koysam bile, 'd' harfinin yokoluşu ile 't' harfinin belirişi arasında kısacık bir süre geçtiği gibi, değiştirilen harfi izleyen harflerin, sözcüklerin, satırların, vs. kapsayan bir yeniden diziliş; harf, sözcük, satır, vs. aralarının yeniden hesaplanması, ayarlanması, dizilişi vs., bazen tüm sayfa boyunca uzayıp gider (Şu modern çağda akılsız hesabın sonucuna bakınız!) Düzeltme işlemini ne şekilde yaparsam yapayım, önümdeki ekranda gördüğüm tüm harfler, ekranın tümü saniyede altmış kez taranırken bir elektron ışınının izinde yer alan mesafelerden ibarettir. İstediğim bir harfi çıkarıp yerine benzerini koyarsam, bu değişim sonrası sistem aynı mıdır, yoksa değişim öncesi sistemden sadece ayırt edilemez midir? İkinci seçeneği ('sadece ayırt edilemez'), birinci seçenekten ('aynısı') farklı bir seçenek olarak benimsemeye çalışmak saçma görünüyor. En azından, harfler aynı olduğu zaman genel durumu da aynı olarak nitelemek mantıklı bir yaklaşım sayılabilir. Özdeş parçacıkların kuantum mekaniği de işte böyledir. Bir parçacığın yerine ona özdeş bir parçacığın konulması, aslında, genel sistemde hiçbir şey yapmamış olmaktır: Duruma, öncekinin aynı gözüyle bakabilirsiniz. (Ancak, VI. Bölüm'de ele alacağımız gibi, kuantum mekaniksel bütünlük içerisinde fark biç de önemsiz değildir.)

İnsan bedenindeki atomların sürekli değişimi ile ilgili olarak yukarıda değinilen görüşler, kuantum

fiziğinden çok klasik fizik kapsamındadır. Sözcüklerin seçiminde, her bir atomun bireyselliğinin korunması açısından anlam taşıyorlarmışçasına özen gösterilmiştir. Oysa klasik fizik bu konuda yeterlidir ve bu düzeyde bir açıklama yaparken, atomları bireysel nesneler olarak tanımlamakla çok yanlış davranmış sayılmayız. Kuantum mekaniği açısından atomların bireyselliğinden, yalnız anlatım kolaylığı sağladığı için değil, öngörülen açıklama düzeyine uygunluğu bakımından da bahsedilmiştir.

Bir insanın bireyselliğinin, bedensel materyalini oluşturan nesnelere atfetmeye çalışabileceği bireysellikle hiç ilgisi yoktur. Bunun yerine, bir bakıma, bu nesnelerin, diyelim uzayda veya uzay-zamanda şekillenilim ile ilgisi vardır. Fakat, güçlü Al yandaşları biraz daha ileri gidiyorlar ve diyorlar ki: Böyle bir şekillenilimin bilgi içeriği, özgün olanın tekrar içerisinden çekilip alınabileceği bir başka biçime dönüştürülebiliyorsa, kişinin bireyselliğinin bütünlüğüne dokunulmamalıdır. Bu biraz, benim tuşlara basarak biraz önce yazdığım ve kelime-işlemci'min ekranında beliren harflerin dizilişine benziyor. Onları ekrandan silersem, elektrik yükünün minik yer değiştirmelerinden oluşan belirli bir formatta, kendi geometrik şekillenilimlerinde, kodlanmış olarak öylece kalacaklar. Ancak istediğim an onları tekrar ekrana getirebilirim; ve işte -sanki hiçbir dönüşüm olmamış gibi ekrandalar. Yazdıklarımı saklamak istersem, harf dizilerinden oluşan bilgiyi bir disk üzerindeki manyetizasyon şekillenilimlerine aktarır, diski yerinden çıkarırım ve sonra makineyi kapayarak içindeki tüm (ilgili) minik yük yer değişimlerini nötrleştiririm. Yarın, diski tekrar takar, minik yük yer değişimlerini yeniden başlatır ve hiçbir şey olmamış gibi yine harf dizilerini ekrana getirebilirim. Güçlü Al yandaşlarına göre, bir kişinin bireyselliğinin de aynı şekilde işleme tâbi tutulması 'açıkça' olasıdır. Ekranımda beliren harfler gibi bu insanlar da bir kişinin bireyselliğinden hiçbir şey kaybetmeyeceğini iddia edebilirler. Fiziksel yapısı, tamamen farklı bir şeye, diyelim bir demir yığını içerisindeki mıknatıs alanına dönüştürülmediği sürece, gerçekten de bireyselliğine hiçbir şey olmaz mı? Güçlü Al taraftarları, kişilik 'bilgisi' böyle bir başka formdayken dahi bireyin, bilinçli olarak çevresinin farkında olma yeteneğinin kaybolmayacağını iddia edebilirler. Böyle bir durumda, insanın söz konusu yeteneği bir bilgisayar yazılımı ile temsil edilirken, bilinçli tepkisinin ise, bilgisayarın mekanik parçaları yerine konan insan beyni ve bedeni vasıtasıyla yazılımın faaliyet göstermesi olarak yorumlanması gerekir.

Özetlersek, donanım hangi biçimde olursa olsun -örneğin elektronik bir aygıt olsun- sorular daima (Turing testinde olduğu gibi) yazılım soruları tarzında sorulacak; donanım bu soruları yanıtlarken iyi bir performans gösterirse, vereceği yanıtlar normal durumdaki bir insanın vereceği yanıtlardan ayırt edilemeyecek. ('Bu sabah kendinizi nasıl hissediyorsunuz?' 'Oldukça iyiyim, teşekkür ederim. Biraz başım ağrıyor,' 'Öyleyse kendinizi... şey... yani... kişisel kimliğinizde bir bozukluk... veya başka bir şey?' 'Hayır; neden böyle söylediniz? Ne tuhaf bir soru bu böyle.' 'Öyleyse kendinizi dün hissettiğiniz aynı kişi olarak hissediyorsunuz?' 'Elbette öyle hissediyorum!')

Bu bağlamda sık sık tartışılan konu, bilimkurgunun *teleulaşım* makinesidir.[12] Diyelim, bir gezegenden diğerine 'ulaşım' aracı olarak tasarımlanmış olmasına karşın, gerçek amacın bu olup olmadığı tartışılır. Bilim kurgu kahramanı yolculuğunu bir uzay gemisiyle 'normal' şekilde yapacakken, bedenindeki her atomun ve her elektronun yeri ve koşulları kesin ve doğru şekilde ayrıntılandırılarak saptanmak üzere baştan ayağa taranıyor. Tüm veriler elektromanyetik bir sinyalle, varılacak uzak gezegenlere ışınlanıyor (ışık hızında). Veriler, burada, yolcunun tüm anıları, düşünceleri, umutları ve en derin duygularıyla birlikte tam bir kopyasının oluşturulmasına ilişkin komutlar doğrultusunda derleniyor ve kullanılıyor. Beyin etkinliklerinin her ayrıntısı özenle saptanıp iletildikten sonra yeniden inşa edildiğine göre, sonucun böyle olacağını en azından ümit edebiliriz. Mekanizmanın tasarımlandığı şekilde işlediğini varsayarsak, yolcunun özgün nüshası 'güvenli

şekilde' imha edilebilir. Bu aşamada sorulacak soru: Bir yerden bir yere yolculuk yöntemi *gerçekten* böyle midir, yoksa kopyasını çıkararak aslını öldürme yöntemi midir? Yöntemin, kendi referansına dayanılarak, tamamiyle güvenli olduğu kanıtlanırsa bu 'yolculuk' yöntemini denemek ister misiniz? Teleulaşım, seyahat etmek *değilse*, bir odadan diğerine yürüyerek geçmek ile arasında, *ilke olarak,* ne fark vardır? Odadan odaya yürüyerek geçerken, herhangi bir anda, insanın bedeninde bulunan atomlar, bir sonraki anın atomlarının yerine dair bilgi veremiyor mu? Daha önce, herhangi bir atomun kimliğini korumanın önemsiz olduğunu incelemiştik. Herhangi bir atomun kimliğinin sorgulanması bile anlamsız. Atomların hareket halindeki biçimi, bir yerden diğerine doğru giderek çoğalan bir çeşit bilgi dalgası oluşturmaz mı? Bir odadan diğerine yürüyerek geçen yolcumuz ile teleulaşım cihazıyla yolculuk yapan yolcumuzu tanımlayan dalgaların değişimi arasında temel fark nerededir?

Teleulaşım cihazının gerçekten 'işe yaradığını' ve yolcunun uzak gezegendeki kopyasında gerçek 'bilincinin' tekrar uyandırıldığını varsayalım. (Yolcuya yöneltilen sorunun gerçekten anlam taşıdığını varsayarak). Ya yolcunun *özgün* kopyası, oyunun kuralı gereği, ortadan kaldırılmazsa ne olacak? 'Bilinçliliği' aynı anda iki ayrı yerde olabilir mi? (Şöyle bir konuşma karşısında tepkisinin ne olacağını düşünmeye çalışın: 'Ah canım, Teletaşıyıcıya yerleştirmeden önce sana verdiğimiz ilâç zamanından önce etkisini yitirdi. Öyle mi? Şanssızlık ama sorun değil. Öteki senin -şey, yani *gerçek* senin- sağ salim Venüs'e vardığını öğrenmek seni sevindirecektir; böylece seni, şey... yani buradaki *işe yaramaz* kopyanı yok edebiliriz. Tabii ki hiç acı duymayacaksın.') Burada bir paradoksla karşılaşıyoruz. Fiziğin yasaları arasında teleulaşıma ilke olarak olanak sağlayan bir yasa var mı? Belki, diğer taraftan, bir insanı bilinciyle birlikte bu yolla bir yere ulaştırmaya engel, ilke olarak, hiçbir şey yoktur ama 'kopyalama' işleminin özgün olanı yok etmesi olasılığı yok mudur? Yoksa ilke olarak birbirinden bağımsız iki ayrı canlı kopyayı muhafaza etmek mi olası değildir? Bu görüşlerin alışılmadık içeriğine rağmen, bunlardan hareketle bilinç ve kişisel özelliklerin fiziksel doğası ile ilgili önemli bazı ipuçları elde edebileceğimize inanıyorum. Ussal olguların anlaşılmasında *kuantum mekaniğinin* rolü hakkında bir işaret verebilirler. Şu anda bu konunun üzerinde fazla durmayacağım, VI, Bölüm'de kuantum teorisini inceledikten sonra bu konuya tekrar dönmek gerekecektir.

Şimdi Güçlü AI'ın teleulaşım konusuna bakış açısını ele alalım. İki gezegen arasında bir röle istasyonu bulunduğunu varsayalım. Veriler burada, son durağa yansıtılmadan önce geçici olarak depolanıyor olsun. Kolaylık sağlaması bakımından veriler, insan biçiminde değil fakat bir çeşit manyetik veya elektronik cihazda depolanıyor. Yolcu'nun bilinci böyle bir cihazda depolanabilir mi? Güçlü Al taraftarları, depolanabileceğine bizi inandırmaya çalışacaklardır. Ne de olsa, diye savunacaklardır, yolcunun beyninin uygun etkinliğini 'sadece' taklit etmek, simüle etmek suretiyle cihazın, yolcunun sormasını istediğimiz soruları yanıtlamasını sağlayabiliriz. Cihaz gerekli tüm verilerle donanımlı olacaktır; gerisi de hesaplamaya kalıyor. Cihaz, yolcunun kendisiymiş gibi sorulanları yanıtlayacağına göre (Turing testi!), yolcunun bizzat kendisi *olacaktır*. Güçlü Al'in, ussal olgular açısından gerçek, donanımın önemli olmadığına dair görüşü bu örneğe dayanmaktadır. Bence yeterli gerekçeye sahip olmayan bir görüştür. Beynin (veya usun) aslında dijital bir bilgisayar olduğu varsayımına dayanır. Buna göre, bir insanın düşünme eylemi esnasında, beynin doğal fiziksel (biyolojik, kimyasal) yapısına ihtiyaç gösterebilecek hiçbir fiziksel olgunun yardımına başvurulmaz.

Güçlü Al açısından tartışılması gereken konu, gereksinim duyulabilecek fiziksel olguların etkilerinin, dijital hesaplar vasıtasıyla her zaman doğru şekilde *örneklenip örneklenemeyeceğidir.* Birçok fizikçinin bu savı, mevcut fizik bilgimize dayanılarak ileri sürülebilecek çok doğal bir sav olduğunu savunacaklarından hemen hemen eminim. Daha sonraki bölümlerde karşı savımı savunacağım. Fakat, bu aşamada, ilgili tüm fiziksel olguların, dijital hesaplarla daima

örneklenebileceği varsayımını (genelde kabul edilen varsayım) kabul edelim. Böylece yegâne gerçek varsayım, (zaman ve hesaplama uzayı ile ilgili konular dışında) 'işlevsel' yaklaşımdan ibarettir: Bir nesne, tamamen bilinçli bir varlık olarak davranıyorsa, kendini bu varlıkmış gibi 'hissetmelidir'.

Güçlü Al, 'sadece' bir donanım olarak beynin etkinlikleri için yardımına başvurulan fizik biliminin, uygun bir çevirmen yazılım kullanılarak simule edilebileceği görüşündedir. İşlevsel bakış açısından konu ele alındığında, evrensel Turing makinelerinin etkinliği ve bu makinelerce gerçekleştirilebilecek algoritmalar -ve, kuşkusuz, beynin bir tür algoritmik eyleme uygun çalışması- gibi görüşlerle yüz yüze geliyoruz. Bu dolambaçlı ve önemli kavramlar hakkında biraz daha açıklayıcı olmanın zamanı geldi.

Açıklamalar

[←I] Böyle bir kitabın yazılması esnasında karşılaşılan sorunlardan birisi "she" veya "he" (dişi/erkek zamirlerin) kullanımıdır. Normal uygulamada olduğu gibi, bundan böyle, soyut kişilerden bahsederken, her ikisini kapsamak üzere, yalnız erkek zamirini (he) kullanacağım. Sorgulayıcıyı "she" zamiri ile ifade ederek cinsiyet ayırımı yapmak amacında olmadığımı, yalnız dişi bir sorgulayıcının, bir erkek sorgulayıcıya kıyasla, gerçek insanla bilgisayarı ayırt etmekte daha duyarlı olabileceğini düşündüğüm için bağışlanacağımı umarım!

[←II] Turing testinde başarılı olmanın püf noktalan ile ilgili görüşlerim hakkında bilerek ve isteyerek suskun kalıyorum. Örneğin, şöyle bir yöntem önerebilirdim: Uzun ve başarısız seanslar ertesi bir bilgisayar, insan deneğin o zamana kadar vermiş olduğu tüm yanıtları bir araya getirdikten sonra, uygun içerikleri rastgele arasına serpiştirerek metnin üzerinden bir kez daha geçebilir. Bir süre sonra, yorgun düşen sorgulayıcımızın özgün soruları da tükenince bilgisayarımız 'kopya çekmek' yoluyla sorgulayıcıyı aldatmış olur!

[←III] Bu konuda, karmaşıklık teorisinin ve NP problemlerinin tartışıldığı IV. Bölüm'ün sonuna bakınız.

Kaynaklar

[←1] Bkz. Gardner (1958), Gregory (1981) ve bunlarda yer alan kaynaklar.

[←2] Bkz. Resnikoff ve Wells 51984) s. 181-4. Klasik hesap yöntemleri için bkz. Rouse BalI (1892); Smith (1983).

[←3] Bkz. Gregory (1981), s. 285-7, Grey Walter (1953).

[←4] Bu örnek Delbrück'den (1986) alıntıdır.

[←5] Bkz. O'Connell (1988) ve Kecne'in (1988) makaleleri. Bilgisayarla satranç konusunda daha fazla bilgi için bkz. Levy (1984).

[←6] Kuşkusuz, satranç problemlerinin çoğu insanlar için çözümü zor olacak şekilde tasarımlanmıştır. Çözümü insanlar için aşırı ölçüde zor olmayan fakat günümüzün satranç-problemi çözebilen bilgisayarlarının bin yılda çözemeyeceği bir satranç problemini hazırlamak çok zor olmasa

gerek. (Yapılması gereken, oldukça açık bir plan ve çok sayıda derin hamle hazırlamaktan ibaret olmalı. 200'den fazla hamle -haddinden fazla!- gerektiren, problemler olduğu biliniyor). Böyle bir problem ilginç bir satranç karşılaşması yaratabilir.

[←7] Kitap boyunca Searle'ın terminolojisi olan 'güçlü AI' terimini, bu aşırı görüşün, özünü en iyi yansıttığı için kullanmayı yeğledim. 'Fonksiyonalizm' terimi de aynı görüş için sıkça kullanılmakla birlikte her zaman tam anlamı veremiyor. Söz konusu görüşü ileri sürenlerden bazıları Minsky (1968), Podor (1983), Hofstadter (1979) ve Moravec'dir (1985).

[←8] İddia ile ilgili olarak bkz. Searle (1987), s. 211.

[←9] Searle'ın 'The Mind's I' da tekrar basılan ilk raporunu eleştirirken Douglas Hofstadter, hiçbir insanın başka bir insanın usunu, son derece karmaşık olması nedeniyle, anlaşılabilir bir 'öz analizini" yapamayacağından yakınmaktadır. Gerçekten de yapılamaz! Fakat bence içselleştirmenin gerçekleştirilmesi asıl amaç değildir. Algoritmanın, bir ussal olayın cisimleştirilmesi ile ilgili kısmıyla ilgilenmek yeterlidir. Bu, bir Turing testi sorusunun yanıtlanması esnasında bir anlık "bilinçli kavrama" olabileceği gibi daha basit bir yöntem de olabilir. Böylesi herhangi bir ussal olay, neden son derece karmaşık bir algoritmayı gerektirsin?

[←10] Bkz. s. 368, 372, Searle'ın (1980), Hofstadter ve Dennett'teki (1981) makalesi.

[←11] Bu konuda bilgi sahibi bazı okurlar, bazı işaretlerin farklı olmasından yakınabilirler. Bu (tartışılabilir) fark bile, elektronlardan birini 360° çevirirsek, kaybolacaktır (Açıklama için bkz. VI. Bölüm).

[←12] Bkz. Hofstadter ve Dennett'e Giriş (1981).

II. Bölüm
Algoritmalar ve Turing Makineleri

Algoritma Kavramı

Bir algoritma, veya bir Turing makinesi, veya bir evrensel Turing makinesi tam olarak nedir? Niçin "düşünen alet" nedir sorusu üstüne oluşan çağdaş düşünceler arasında bu kavramlar merkez konumundadırlar? Bir algoritmanın yapabilecekleri ilke olarak sınırlı mıdır? Bu sorulara yeterli yanıt getirebilmek için öncelikle algoritma fikrini ve Turing makineleri kavramını ayrıntılı incelememiz gerekecek.

İlerideki çeşitli tartışmalar sırasında bazen matematiksel ifadelere başvurmamız kaçınılmaz olacak. Okuyucularımdan bir kısmının bundan hiç hoşnut kalmayacaklarının, hatta belki bunu ürkütücü bulacaklarının farkındayım. Eğer bu okurlardan birisi iseniz, size sabır dileyerek, "Okuyucuya Açıklama" bölümündeki tavsiyelerimi izlemenizi öneririm. Burada verilen tartışmalar genelde ilköğretimin ötesinde bir matematik bilgisi gerektirmeyecek; ancak, ayrıntılara inebilmek için epey kafa yormak kaçınılmazdır. Aslında vereceğim pek çok örnek yeterince açıktır ve ayrıntıları izleyerek kavranabilirler. Ama şöylece üzerinden bir kez okunmaları bile tartışılan konular hakkında sizlere fikir verecektir. Öte yandan eğer konuların uzmanı iseniz sizlere de sabır dileyeceğim. Yine de diyeceklerime şöyle bir bakarsanız ilginizi çekecek birkaç şey bulabileceğinizi sanıyorum.

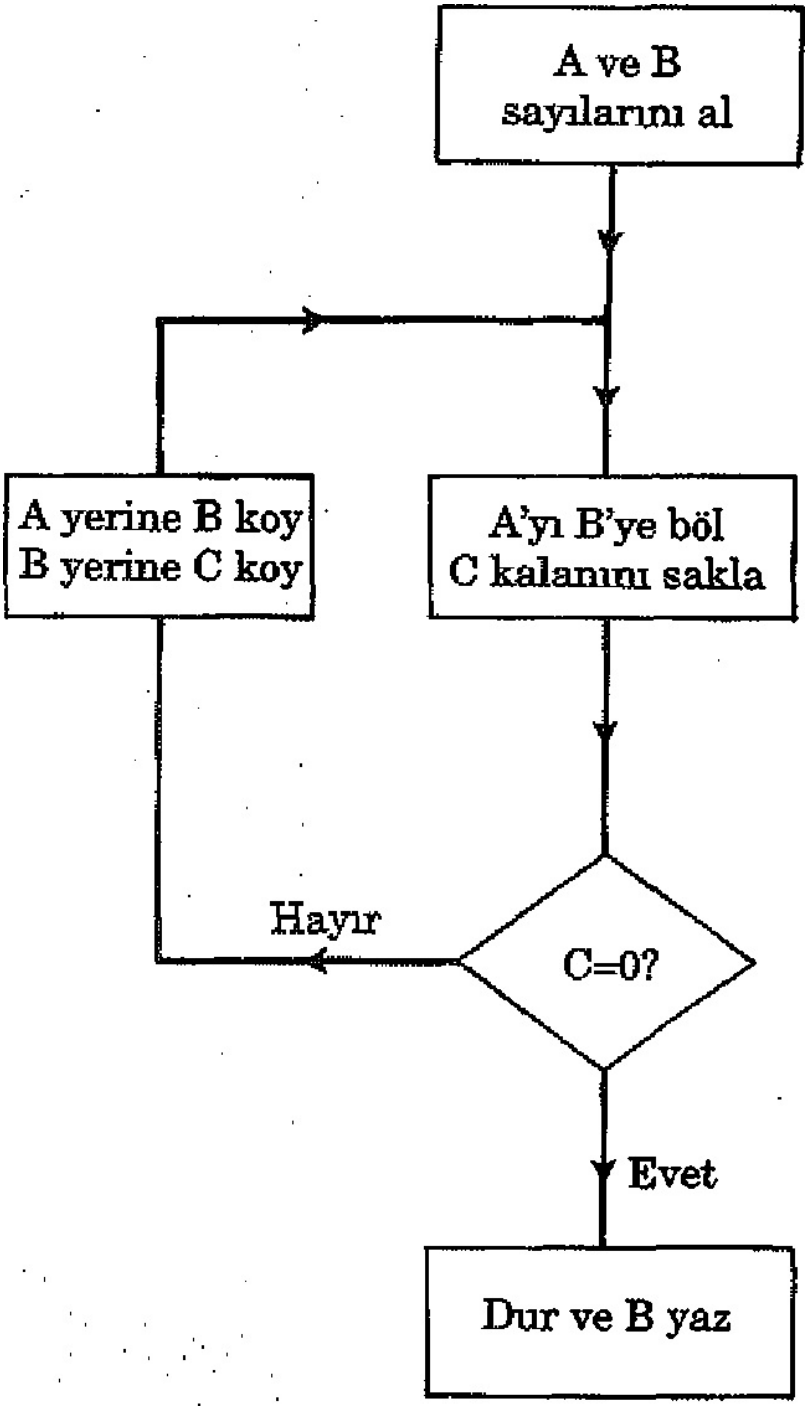
'Algoritma' kelimesi 9. yüzyılda yaşamış Horasan doğumlu matematikçi Ebu Cafer Muhammed İbn-i Musa el Harezmi'nin (al-Kho-wârizm) adından gelmektedir. M.S. 825 yıllarında "Kitab el cebr ve'l mukabele" başlığıyla çok etkili olmuş bir matematik ders kitabı yazmıştır. Eskiden kullanılmakta olan 'algorizma' teriminin yerini bugünkü kullanımda "algoritma" kelimesinin almış bulunması "aritmetik" kelimesiyle kurulan bir ilişkiye bağlı olmalıdır. ('Cebir' kelimesinin de aynı kitabın başlığında yer alan Arapça el-cebr'den gelmiş olması ilginçtir).

Aslında algoritmalar el Harezmi'nin kitabından çok önce de bilinmekteydiler. Eski Yunan döneminden (yaklaşık M.Ö. 300 den) beri bilinen ve *Eukleides algoritması* adıyla tanınan en iyi örnek, iki sayının en büyük ortak bölenini bulma yöntemidir. Konuyu daha iyi anlatmak için, örneğin 1365 ve 3654 gibi bir sayı çifti düşünmek yararlı olacaktır. Bu sayıların en büyük ortak böleni (EBOB), verilen her iki sayıyı tam olarak bölen en büyük tamsayıdır. Eukleides algoritmasını uygulamak için verilen sayılardan büyüğünü küçüğüne bölüp kalanı bulacağız: 1365, 3654'de iki kere var ve kalan 924 dür, (924 = 3654 - 2 x 1365) Şimdi önceden verilen sayı çifti yerine kalan 924 ve bölen 1365 sayılarını alalım. Bu yeni sayı çiftiyle yukarıdaki işlemi tekrarlayalım: 924, 1365'de bir kere var ve kalan 441'dir. Böylece 441 ve 924'den oluşan yeni bir sayı çifti oluşturduk. 924'ü 441'e bölersek kalan 42 olur. (42 = 924 - 2 x 441). Bir tam bölme işlemine kadar bu süreç tekrarlanır. Açıkça yazarsak:

$$
\begin{array}{rcl}
3654 \div 1365 & \text{kalan} & 924 \\
1365 \div 924 & \text{kalan} & 441 \\
924 \div 441 & \text{kalan} & 42 \\
441 \div 42 & \text{kalan} & 21 \\
42 \div 21 & \text{kalan} & 0.
\end{array}
$$

En son bölen sayı, yani 21 sayısı, aranmakta olan en büyük ortak bölendir.

Eukleides algoritması, bu ortak böleni bulmak için izlediğimiz *sistemli yöntemin* kendisidir. Burada yöntemi belli bir sayı çiftine uyguladık, ama yöntemin kendisi herhangi iki sayıya uygulanabilecek kadar geneldir. Eğer verilen sayılar çok büyük olursa, yöntemi uygulamak uzun bir süre alabilir; sayılar ne kadar büyükse süreç o kadar uzun sürebilecektir. Fakat her durumda süreç tamamlanır ve sonlu sayıda işlem yaptıktan sonra kesin bir yanıta ulaşılır. Her adımda hangi işlemin yapılacağı tam olarak belirlenmiştir; sürecin ne zaman sona erdirileceği de kesin bir kurala bağlanmıştır. Ayrıca, tüm süreç sınırsız büyük doğal sayılara uygulanmakta olmasına rağmen *sonlu* sayıda adımla belirlenebilmektedir. ('Doğal sayılar' deyince eksi olmayan tamsayıları [1] anlayacağız: 0, 1, 2, 3,...) Nitekim, Eukleides algoritmasının, mantıksal işlemlerinin tamamı (sonlu) bir 'akış şeması' ile temsil edilebilir.



Yukarıda sürecin en temel basamaklarına kadar ayrıştırılmadan verilmiş olduğuna dikkat edilmelidir. Çünkü A ve B diye verilen iki doğal sayının bölümünden elde edilecek kalan sayının hangi işlemlerle bulunacağını 'bildiğimiz' varsayılıyor. Hepimizin okulda öğrendiği bu bölme işlemi de algoritmik bir işlemdir. Gerçekte bölme işlemi Eukleides algoritmasının kendisinden daha karmaşık bir süreçtir, ancak benzer biçimde bir 'akış şeması' ile verilebilir. Esas zorluk doğal sayıları (normal olarak) ondalık sistemle göstermemizden kaynaklanmaktadır. Dolayısıyla çarpım tablosunu oluşturmak, eldeleri taşımak vb. işlemler yapılmaktadır.

Eğer bir *n* sayısını göstermek için sadece *n* çizik atsaydık, örneğin 5 sayısını ///// ile gösterseydik, kalanı bulmak gerçekten basit bir algoritmik işlem haline gelirdi. A sayısı B sayışına bölününce kalanı bulmak için A'dan B sayısını temsil eden çizik gruplarını peş peşe çıkarırdık. Geride kalan çizikler bize yanıtı verirdi. Örneğin, 17'yi 5'e bölersek kalanı bulmak için ///// gruplarını ////////////////// grubundan şöyle ayırırdık:

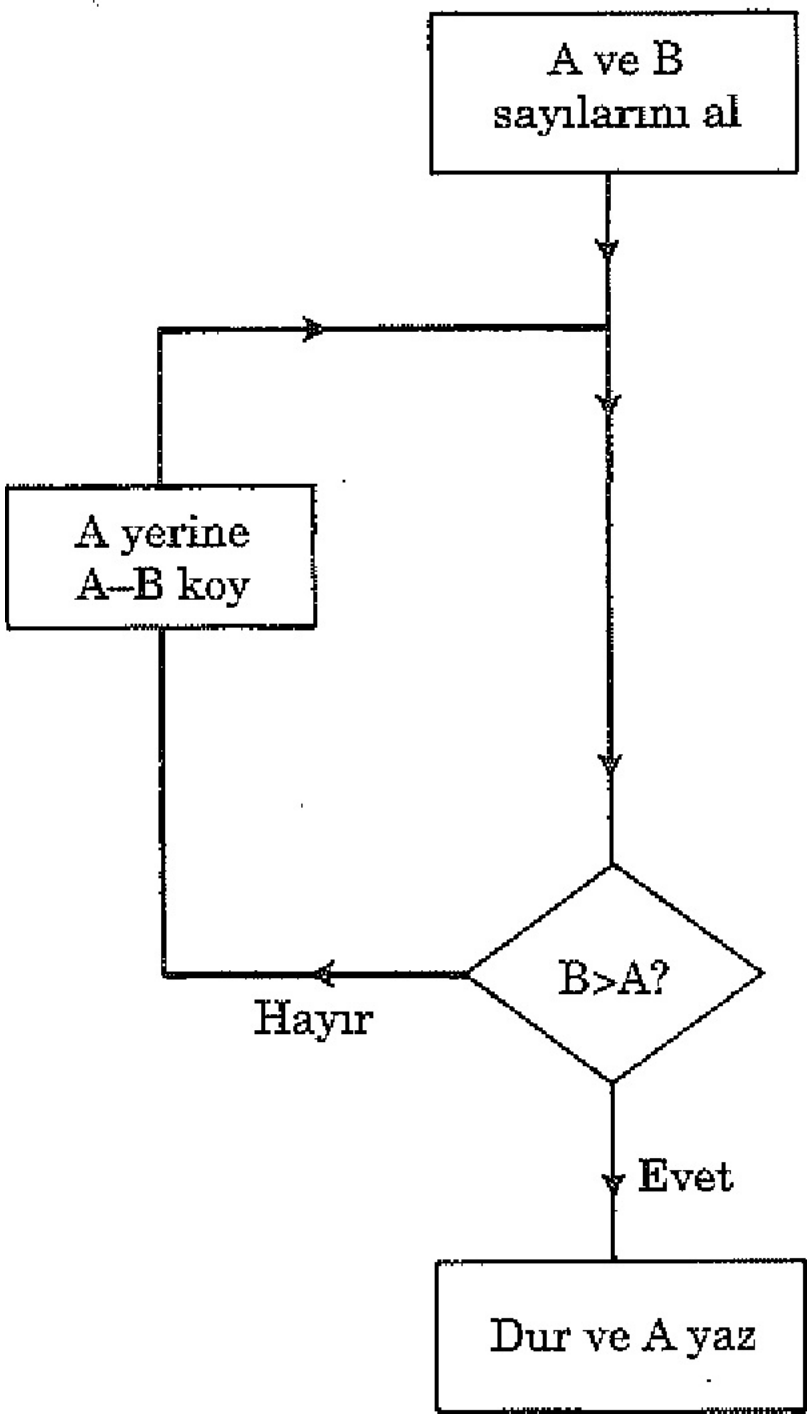
//////////////////

////////////

///////

//

Üç adım sonra yanıt iki olacaktır.



Bir bölme işleminde kalanı, tekrarlanan çıkarma işlemleriyle veren bir 'akış şeması' aşağıda verilmiştir. Eukleides algoritmasını veren akış şemasını tamamlamak için yukarıdaki şemayı önceki akış şeması içinde sağda ortadaki kutu içine yerleştirmeliyiz. Bu tür bir algoritmayı diğer bir algoritmanın içinde yerine koyma işlemine bilgisayar programlamasında sık rastlanır.

n sayısını, *n* tane çizikle temsil etmek sayılar büyüdükçe giderek kullanışsız olmaya başlar, ve bizler bu nedenle sayıları göstermek için standart ondalık sistemi kullanırız. Ama burada işlemlerin veya gösterimin *kullanışlılığı* bizi o kadar ilgilendirmiyor. Daha çok hangi işlemlerin *ilke olarak* algoritmalarla yapılabileceği ile ilgilenmekteyiz. Bir sayı sisteminde algoritma ile yapılabilen bir işlem başka herhangi bir sayı sisteminde de algoritma ile yapılabilir. Farklar sadece ayrıntıda ve

işlemlerin karmaşıklığındadır.

Eukleides algoritması, matematiğin hemen her dalında bulunabilecek, artık klasikleşmiş pek çok algoritmadan sadece birisidir. Ancak dikkate değer bir nokta, algoritma örneklerinin bu kadar geriye giden tarihine karşı, *genel algoritma kavramının* kesin olarak verilmesinin ancak yüzyılımızda gerçeklemiş bulunmasıdır. Bu kavramın birbirine eşdeğer tanımlarının hepsi 1930'larda yapılmıştır. Bunlardan en doğrudan ve ikna edici olanı ve tarihsel açıdan da en önemlisi *Turing makinesi* diye bilinen kavram yardımıyla verilenidir. Bu 'makineleri' ayrıntılarıyla incelemek gerekecektir.

Bir Turing makinesi hakkında aklımızda tutulacak en önemli husus bunun bir fiziksel nesne değil bir 'soyut matematik' ürünü olduğudur. Kavram ilk kez, İngiliz matematikçisi, ünlü şifre uzmanı ve ilk bilgisayar bilimcilerinden Alan Turing tarafından (Turing 1937) 1935-6 senesinde daha geniş kapsamlı bir probleme yanıt getirebilmek amacıyla ortaya konmuştur. *Entscheidungsproblem* adıyla bilinen bu problem büyük Alman matematikçisi David Hilbert tarafından 1900 Paris Uluslararası Matematikçiler Kongresinde kısmen (Hilbert'in 10. problemi) ve daha sonra 1928 Bologna Kongresi'nde tam olarak tanımlanmıştı. Hilbert en genel bir matematik probleminin çözümü için algoritmik yöntem bulunup bulunamayacağını sormaktaydı; ya da 'daha doğrusu' böyle bir yöntemin ilke olarak var olup olmadığını sorgulamaktaydı. Hilbert'in amacı, aksiyomlarını ve yöntem kurallarını belirleyecek bir program yardımıyla, matematiği tartışılamaz sağlamlıkta bir temel üzerine oturtmaktı. Ancak daha Turing'e gelmeden önce 1931'de parlak Avusturyali mantıkçı Kurt Gödel'in şaşkınlık veren bir teoremi nedeniyle bu program çıkmaza girmiş bulunmaktaydı. Gödel teoremini ve bunun önemini IV. Bölüm'de ele alacağız. Turing'in ilgilenmiş olduğu Hilbert problemi (*Entscheidungsproblem*) matematiğin herhangi bir aksiyom sistemi cinsinden yazılmasından daha derine inen bir problemdir. Sorun şudur: Matematiğin tüm problemlerini birbiri peşi sıra çözebilecek genel bir mekanik yöntem ilke olarak var mıdır?

Bu soruyu yanıtlamakta karşılaşılan bir zorluk *mekanik yöntem* ile ne kastedildiğini anlamaktı. Kavram, o dönemin alışılmış matematik anlayışının dışında kalıyordu. Bir yanıt getirebilmek için, Turing önce 'makine' kavramını, işlevlerini temel öğelerine ayırarak nasıl tanımlayacağı üzerinde düşündü. Turing'in insan beynini bu anlamda bir 'makine' olarak gördüğü açıktır. Böylece matematikçinin bir matematik problemiyle uğraşırken yaptığı eylemler de 'mekanik yöntemler' başlığı altında toplanabilmekteydi.

Her ne kadar insan düşüncesine bu bakış açısı, Turing'in bu çok önemli kavramını geliştirmesinde değer taşımışsa da bizler aynı bakış açısını kabule zorunlu değiliz. Nitekim, bir mekanik yöntemden ne kastedildiğini tam olarak açıkladıktan sonra Turing, iyi tanımlı ama normal anlamda mekanik denmeyecek bazı matematik işlemlerin varlığını göstermiştir. Turing'in kendi çalışmalarına dayanarak onun akıl olgusunun doğası üzerine görüşlerinde açık noktalar bulunduğuna kanıt getirebilmek bir çelişki olmalı. Ancak şu an konumuz bu değil. Öncelikle Turing'in mekanik yöntem kavramının ne olduğunu anlamalıyız.

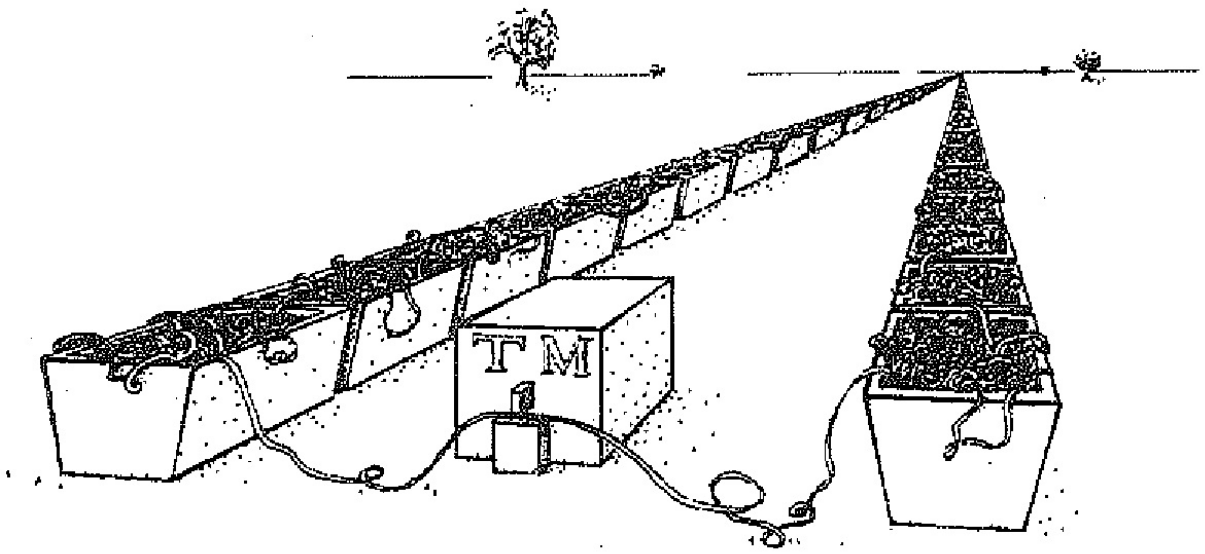
Turing Makinesi Kavramı

Bir hesap yöntemini (sonlu tanımlanabilir) uygulayan bir makine düşünelim. Genel hatlarıyla nasıl bir makine olurdu bu acaba? Şimdilik uygulama tekniği hakkında fazlaca kafamızı yormayalım; matematik açısından ideal bir tasarıma sahip olması yeterli. Cihazımız olarak sonlu (belki de çok fazla sayıda) ve birbirinden farklı olası bir bağımsız durumlar kümesine sahip olsun istiyoruz. Biz

bunlara makinenin içsel durumları adını veriyoruz. Ancak, makinemizin ilke olarak gerçekleştireceği hesapların boyutunu sınırlamak istemiyoruz. Eukleides'in yukarıda tanımladığımız algoritmasını anımsayınız. İlke olarak, algoritma veya genel hesap yöntemi, sayılar ne kadar büyük olursa olsun, yine aynıdır. Büyük sayılar söz konusu olduğu zaman, yöntem gerçekten çok uzun zaman alabilir ve hesapları yapabilmek için önemli miktarda 'müsvedde kağıdı' gerekebilir. Fakat sayılar ne kadar büyük olursa olsun algoritma yine aynı sonlu komutlar kümesinden ibarettir.

Bu nedenle makinemiz, sonlu sayıda içsel duruma sahip olmasına karşın, boyutu sınırlanmamış bir girdiyle işlem yapabilmelidir. Ayrıca makinemiz, işlemleri için, sınırsız bir dışsal veri biriktirme alanından ('müsvedde kağıdı') yararlanabilmeli, aynı zamanda sınırsız boyutta çıktı üretebilmelidir. Makinemiz, sonlu sayıda farklı içsel duruma sahip olduğu için, ondan ne tüm dışsal verileri, ne de kendi hesaplarının sonuçlarını içsel konuma getirmesi beklenmemelidir. Bunun yerine, verilerin veya önceki hesapların *anında* işleme soktuğu kısımlarını incelemeli, daha sonra bunlarla ilgili olarak hangi işlemlerin yapılması gerekiyorsa onları yapmalıdır. Dışsal veri biriktirme alanına, işlemin sonuçlarını kaydedebilir ve işlemin bir sonraki aşamasına gayet kararlı bir şekilde geçebilir. Girdinin, hesap mekânının ve çıktının sınırsız doğası, pratikte fiilen oluşturulabilecek bir şeyden çok, yalnız matematiksel bir idealin gerçekleşmesiyle ilgilendiğimizi hatırlatır (bkz. Şekil 2.1). Fakat bu konumuzla önemli ilgisi olan bir idealin gerçekleşmesidir. Modern bilgisayar teknolojisinin harikaları bize, pratik amaçlarımızın birçoğu için sınırsız olarak kullanılabilecek elektronik veri biriktirme alanı (bellek) sağlamıştır.

Gerçekte, dışsal olarak tanımlanan bellek alanı tipi, modern bir bilgisayarın içsel işlemlerinin bir kısmım oluşturur. Bellek alanının belirli bir kısmının dışsal veya içsel olarak nitelendirilmesi belki teknik bir ayrıntıdır. "Makine' ve 'dışsal' kısım ayırımı, *donanım* (hardware) ve *yazılım,* (software) olarak kabul edilebilir. Buna göre, içsel kısım (mekanik) donanım, dışsal kısım yazılımdır. Bu konuya saplanıp kalmayacağım ama neresinden bakarsanız bakın, Turing'in idealleştirilmesi günümüzün elektronik bilgisayarlarına gerçekten çok yaklaşmıştır.



Şekil 2.1 Kurallara uygun bir Turing makinesi sonsuz bant gerektirir!

Turing, dışsal veri bellek alanını, üzerinde işaretler bulunan "bant" şeklinde gözünde canlandırmıştır. Bu bant ileri/geri hareket edebilir. Makine, gerekirse, bantın üzerine yeni işaretler koyabildiği gibi eski işaretleri silebilir ve böylece aynı bantın, girdi olduğu kadar dışsal bellek ('müsvedde kağıdı') gibi görev yapmasını sağlar. Gerçekte, dışsal bellek ile 'girdi' arasında ayırım yapmamak yararlı olur, çünkü işlemlerin çoğunda bir hesabın ara sonuçları, yeni veriler gibi rol

alabilir. Eukleides'in algoritmasında, ilk girdimizle (A ve B sayıları), hesabın çeşitli aşamalarını birbirlerinin yerine koyduğumuzu hatırlayınız. Aynı şekilde, aynı bant, kesin çıktı (yani, 'yanıt') için kullanılabilir. Bant, başka hesapların yapılmasına gerek olduğu sürece durmadan ileri/geri hareketini sürdürecektir. İşlem tamamlandığı zaman makine durur ve hesap sonucu, bantın makinenin bir yanında yer alan kısmı üzerinde okunabilir. Kesin bir tanımlama için farz edelim ki, sonuç daima sol taraftan verilirken, girdi kapsamındaki tüm sayısal veriler ve çözülecek problemle ilgili komutlar daima sağ taraftan verilir.

Kendi adıma, sınırlı makinemizin, potansiyel olarak sonsuz bantı ileriye geriye hareket ettirmek zorunda olmasından biraz rahatsızlık duyuyorum. Malzemesi ne kadar hafif olursa olsun, *sonsuz* bir bantı hareket ettirmek zor olmalı! Bantın, sınırlı makinemizin içinde hareket edebileceği dışsal ortamı temsil ettiğini düşlemeyi yeğlerdim (Kuşkusuz, modern elektronik cihazlarda, basit fiziksel anlamda, ne 'bant' ne de 'cihaz' gerçekten "hareket' ederler; fakat böyle bir 'hareket' nesnelerin hayalde canlandırılmasını kolaylaştırıyor). Bu teoriye göre makine, girdinin tamamını ortamdan alır. Ortamı, kendine ait 'müsvedde kağıdı' olarak kullanır. Sonuçta, çıktısını aynı ortam üzerine kaydeder.

Turing'in resminde 'bant', her iki yönde sonsuz olduğu varsayılan ve karelerden meydana gelen çizgisel bir diziden oluşur. Bantın üzerindeki her kare ya boştur ya da üzerinde bir tek işaret vardır[1]. İşaretli veya işaretsiz kareler, ortamın, (yani bantın), parçalara ayrılarak birbirinden *bağımsız* (süreklinin karşıtı olarak) kesikli elemanları olarak tanımlanabileceğini gösterir. Makinemizin fonksiyonlarını güvenilir ve kararlı şekilde yerine getirmesini istiyorsak bu özelliğe sahip olması gerekir, Ancak matematiksel idealleştirmenin bir özelliği olarak 'ortamın' (potansiyel bakımdan) sonsuz olmasına karşı değiliz, fakat hangi *özel* durumda olursak olalım girdi, hesaplama işlemi ve çıktı daima *sonlu* olmalıdır. Buna göre bant, sonsuz olarak uzun kabul edilse bile üzerinde bulunan işaretlerin sayısı sonsuz olamaz. Her bir yönde bant belirli bir noktadan sonra tümüyle boş olmalıdır.

Boş kareyi '0' işaretli kareyi '1' ile göstererek bir örnek verelim:

Makinemizin, bantı 'okuması' gerekiyor; okuma işlemini bir defada bir kareyi okuyarak ve her okumadan sonra sağa veya sola yalnız bir kare hareket ederek gerçekleştirdiğini varsayıyoruz. Bir defada *n* kare okuyan veya bir defada *k* kare kayan cihazlar, bir defada yalnız bir kareyi okuyan ve geçen bir cihaz cinsinden kolaylıkla modellenebilirler. *k* karelik bir hareket, bir karenin *k* kere hareketiyle oluşturulabilir; *n* karenin bir defada okunması veri depolamak suretiyle basit cihaz sanki *n* karenin hepsini bir defada okuyormuş gibi yorumlanabilir.

Bu cihaz ayrıntılı olarak ne yapabilir? 'Mekanik' olarak tanımlayacağımız bir şeyin işlevini yerine getirebildiği en genel yöntem nedir? Cihazımızın *içsel durumlarının* sayı bakımından sonsuz olmaması gerektiğini hatırlayınız. Bilmemiz gereken tek şey, bu sonsuz olmamanın ötesinde cihazın davranışının tamamen içsel durumu tarafından ve girdi tarafından kontrol edildiğidir. Girdiyi basitleştirmiş ve '0' ve '1' sembollerinden biriyle temsil etmiştik. Başlangıç durumu ve bu girdiyle donanımlı olarak makinenin tamamen kararlı bir şekilde çalışması beklenir: İçsel durumunu bir başka (belki de aynı) içsel duruma değiştirir; okuma simgeleri olarak 0 veya 1 'i kullanır veya bunların yerine başka simgeler koyar; sağa veya sola bir kare hareket eder; sonuçta, işlemi sürdürüp sürdürmemeye veya bitirmeye karar verir ve durur.

Cihazımızın çalışmasını daha açık anlatmak amacıyla içsel durumlarını *numaralayalım*: 0, 1, 2, 3,

4, 5, ...; buna göre cihazın, veya *Turing makinesinin,* çalışma sistemine, aşağıda örneklendiği gibi bir dizi yer değiştirme egemen olacaktır:

```
0O  →   0OR
0l  →   13lL
1O  →   65lR
1l  →   1OR
2O  →   0lR.STOP
2l  →   66lL
3O  →   37OR
 .       .
 .       .
 .       .
210O →   3lL
 .       .
 .       .
 .       .
258l →   0OR.STOP
259O →   97lR
259l →   0OR.STOP
```

Okun sol tarafındaki *iri* rakam, cihazın okuma işlemini gerçekleştirmekte olduğunu gösteren ve bantın üzerinde yer alan işarettir; cihaz bu simgenin yerine sağ tarafta ortada bulunan iri rakamı koyar. 'R', cihazın bant boyunca sağa doğru bir adım, *'L'* ise sola doğru bir adım hareket etmek zorunda olduğunu gösterir. (Turing'in özgün tanımında olduğu gibi, cihazın yerine bantın hareket ettiğini düşünürsek, R'yi bantın bir kare sağa ve L'yi bir kare sola hareket etmesi komutu olarak yorumlayabiliriz), STOP sözcüğü, işlemin tamamlandığını ve cihazın durması gerektiğini gösterir. Özellikle, 01 —>131L komutu cihaz '0' durumunda ise ve banttan 1 okuyorsa, 13 numaralı içsel duruma geçmesi ve 1'i 1 olarak bant üzerinde bırakması ve bant üzerinde bir kare sola kayması gerektiğini gösterir. Son komut 2591 0OR.STOP, cihaz 259 durumunda ise ve bant üzerinde 1'i okuyorsa, 0 durumuna geçmesi, bant üzerinde 0 üretmek için 1'i silmesi, bant boyunca sağa doğru bir kare kayması ve işlemi tamamlaması gerektiğini gösterir.

İçsel durumları 0, 1, 2, 3, 4, 5,... gibi sayılar kullanarak göstermek yerine, O'lardan ve İlerden oluşan simgeler kullanmamız gerekseydi, bant üzerinde yukarıda açıklanan simgeler sistemine sadık kalmamız daha doğru olurdu. İstersek *n* durumu için *n* adet 1 kullanabilirdik ama bu yetersiz olurdu. Bunun yerine *ikilik* sayı sistemini kullanalım:

$$
\begin{aligned}
0 &\rightarrow 0,\\
1 &\rightarrow 1,\\
2 &\rightarrow 10,\\
3 &\rightarrow 11,\\
4 &\rightarrow 100,\\
5 &\rightarrow 101,\\
6 &\rightarrow 110,\\
7 &\rightarrow 111,\\
8 &\rightarrow 1000,\\
9 &\rightarrow 1001,\\
10 &\rightarrow 1010,\\
11 &\rightarrow 1011,\\
12 &\rightarrow 1100, \text{ vs.}
\end{aligned}
$$

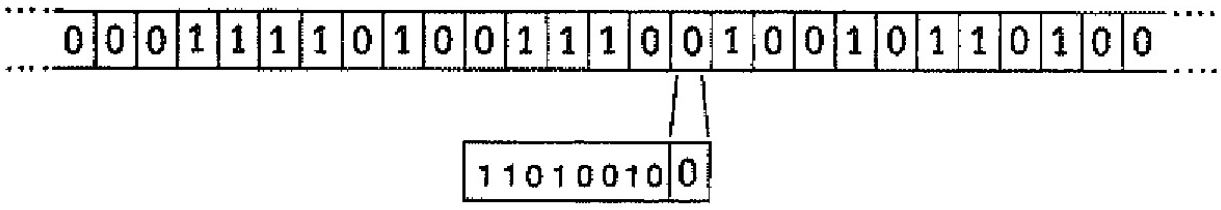
Burada sağ taraftaki son hane standart (ondalık) sistemde olduğu gibi, 'birlikleri' gösterirken, hemen ondan bir önceki hane 'onluklar' yerine 'ikilikler'i gösterir. Bundan önceki hane yüzlükler değil, dörtlükler, daha önceki hane 'binlikler' değil 'sekizlikler', vs. gösterir ve sola doğru hareket ederken birbirini izleyen her hanenin değeri, birbirini izleyecek şekilde *ikinin katlarıdır:* 1,2,4(= 2 x 2), 8(= 2 x 2 x 2), 16(= 2 x 2 x 2 x 2), 32(= 2 x 2 *x* 2 x 2 x 2), vs. (İleride açıklayacağımız nedenlerle, doğal sayıları temsil etmek amacıyla ikilik veya ondalık sistemlerden başka bir basamak sistemi kullanmayı bazen daha uygun bulacağız: Örneğin, üçlük sistemde, 64 ondalık sayısı her hanesi üçün katları değerinde olmak üzere $64 = (2 \times 3^3) + 3^2 + 1$ olarak yazılacaktır. Bkz. IV. Bölüm)

İçsel durumları belirlemek için bu ikilik sistemi kullanırsak, Turing makinesinin komutları aşağıdaki şekilde olacaktır:

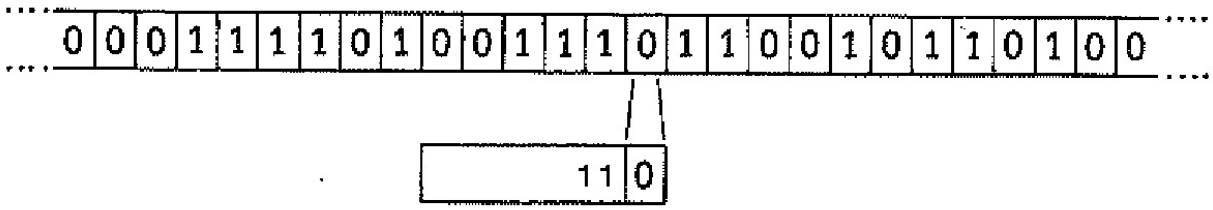
```
     0O  →   0OR
     0l  →   1101lL
     1O  →   1000011R
     1l  →   1OR
    10O  →   0lSTOP
    10l  →   1000010lL
    11O  →   100101OR
      .        .
      .        .
      .        .
110100100  →   111L
      .        .
      .        .
      .        .
      .        .
100000010  →   0OSTOP
100000011O →   1100001lR
100000011l →   0OSTOP
```

Yukarıda R.STOP sembolünü STOP olarak kısalttım çünkü L. STOP komutu hiçbir zaman gözükmez ve işlemin son aşamasının sonucunun, yanıtın bir parçası olarak, daima cihazın sol tarafında gösterilmesi gerekir.

Diyelim ki cihazımız 11010010 ikilik dizi ile temsil edilen özel bir içsel durumda bulunsun, bant ise bir işlemin tam ortasında sayfa 43'te detaylanan biçimde bulunsun, ve biz 11010010O-->111L komutunu uygulayalım. Bant üzerinde okunan hane ('O') daha iri bir rakamla, içsel durumu belirleyen simgeler dizisinin sağında gösterilir.



Turing makinesi örneğinde (kendimce rasgele bazı değişiklikler yaptığım örnekte), okunmakta olan 'O' ile '1' yer değiştirirse ve içsel durum '11'e değiştirilirse cihaz sola doğru bir adım kaydırılmış olur.



Cihaz şimdi sonraki haneyi (yine bir ‘O’) okumaya hazırdır. Tabloya göre 'O’ı değiştirmeksizin olduğu gibi bırakır fakat içsel durumu '100101’ ile değiştirir ve bant boyunca sağa bir adım geri gider. Bu konumda '1'i okur ve tablonun aşağılarında bir yerde içsel durumu değiştirmek için ne gibi bir yer değiştirme uygulanacağına dair komutu bulur: Okumakta olduğu haneyi değiştirmesinin gerekli olup olmadığı ve bant boyunca hangi yönde ilerlemesi gerektiğine dair komut alır. STOP’a ulaşıncaya kadar bu şekilde devam eder; STOP aşamasında (sağa doğru bir adım daha ilerledikten sonra) işlemin tamamlandığı konusunda cihaz operatörünü uyarmak için bir zilin çaldığını varsayarız.

Cihazın daima O içsel durumunda uygulamaya başlatılmasını ve okuma cihazının sol yanında yer alan bantın başlangıçta boş olmasını öneriyoruz. Komutlar ve veriler sağ taraftan yüklenecektir. Daha önce değindiğimiz gibi, yüklenen bilgi daima *sonlu* sayıda 0'lar ve 1'ler dizisi şeklinde alınmalı, bu diziyi boş bant (yani O'lar) izlemelidir. Makine STOP aşamasına ulaştığında işlemin sonucu, okuma cihazının solundaki bant üzerinde görünür.

Sayısal verileri girdimizin bir parçası olarak işleme dahil etmeyi arzu ettiğimiz için, girdinin bir parçası olarak basit sayıları (yani 0, 1, 2, 3, 4, ... gibi doğal sayıları) tanımlamanın bir yolunu bulmak durumundayız. Bunu gerçekleştirmenin bir yolu, *n* sayısını temsil etmek için *n* adet 1 kullanmaktadır (gerçi bu, doğal sayı 0 konusunda bize biraz zorluk çıkaracaktır):

$$1\rightarrow 1,\ 2\rightarrow 11,\ 3\rightarrow 111,\ 4\rightarrow 1111,\ 5\rightarrow 11111,\ \text{vs.}$$

Bu basit numaralama sistemi (oldukça mantıksız bir yaklaşımla) *birlik* sistem olarak adlandırılır. Bu sistemde ‘O’ simgesi yerine farklı sayıları birbirinden ayırmaya yarayan boşluk kullanılabilir. Algoritmaların çoğu sayılardan oluşan *dizileri* esas alarak işlem yaptıkları için sayıları birbirinden ayırmanın yolunu bulmak önemlidir. Örneğin, Eukleides’in algoritmasına göre cihazımız A ve B sayılarından oluşan bir sayı çiftini esas alarak işlem yapacaktır. Turing makineleri, fazla zorlanmadan, bu algoritmanın yazılımını gerçekleştirebilir. Konuya ilgi duyan bazı okuyucular yöntemin '0’ ile birbirinden ayrılan bir birlik sayı çiftine uygulanmasıyla Turing makinesinin (kısaca EUC olarak adlandırıyorum) Eukleides’in algoritmasını uygulayışını, deneme niteliğinde, kanıtlamak isteyebilirler.

$$0O\rightarrow 0OR,\quad 0l\rightarrow 1lL,\quad 1O\rightarrow 10lR,\quad 1l\rightarrow 1lL,$$
$$10O\rightarrow 1010OR,\quad 10l\rightarrow 11OR,\quad 11O\rightarrow 100OR,\quad 11l$$
$$\rightarrow 11lR,\ 100O\rightarrow 100OR,\quad 100l\rightarrow 101OR,\quad 101O\rightarrow 111OL,$$
$$101l\rightarrow 1101L,\quad 110O\rightarrow 110OL,\quad 110l\rightarrow 1lL,\quad 111O$$
$$\rightarrow 111OL,\quad 111l\rightarrow 1000lL,\quad 1000O\rightarrow 1001OL,$$
$$1000l\rightarrow 1000lL\rightarrow 1001O,\quad 10OR\rightarrow 1001l\rightarrow 1lL,$$
$$10100\rightarrow 0OSTOP,\quad 10101\rightarrow 10101R.$$

Bu örneğe girişmeden önce okurumun, UN + 1 Turing makinesi gibi daha basit ve bir birlik sayıya

sadece tek birim ekleyen sistemi denemesi daha akıllı bir davranış olacaktır:

$$0O \rightarrow 0OR,\ 0l \rightarrow 1lR,\ 1O \rightarrow 0lSTOP,\ 1l \rightarrow 1lR,$$

UN + 1 'in bunu gerçekleştirebildiğini kontrol için, 4 sayısını temsil eden

...00000111100000...,

bantına uygulandığını varsayalım. Yine varsayalım ki cihaz başlangıçta 1'lerin solunda bir yerdedir. İçsel durumu '0' olup, '*0'* okumaktadır. İlk komuta uyarak bunu 'O' olarak bırakır ve sağa bir adım kayarken içsel durum '0' da kalır. İlk 1'e ulaşıncaya kadar; sağa doğru birer adım atarak bu işlemi sürdürür. Sonra ikinci komut devreye girer: 1'i 1 olarak bırakır ve bu kez içsel durum 1'de olmak üzere tekrar sağa hareket eder. Dördüncü komuta göre, 1'lere dokunmadan içsel durum 1'de kalır, 1'leri izleyen ilk 0'a varıncaya kadar sağa doğru ilerler. Bu aşamada üçüncü komut cihaza, 0'ı 1'e değiştirmesini, sağa bir adım daha atamasını söyler. Böylece, 1 1er dizisine bir 1 daha eklenmiş olur, ve örneğimizin 4'ü, öngörüldüğü gibi, 5'e gerçekten çevrilmiş olur.

Denememizde biraz daha ileri giderek,

$$0O \rightarrow 0OR,\ 0l \rightarrow 1OR,\ 1O \rightarrow 10lL,\ 1l \rightarrow 1lR,$$
$$10O \rightarrow 11OR,\ 10l \rightarrow 100OR,\ 11O \rightarrow 0lSTOP,$$
$$11l \rightarrow 11lR,\ 100O \rightarrow 101lL,\ 100l \rightarrow 100lR,$$
$$101O \rightarrow 10lL,\ 101l \rightarrow 101lL,$$

ile tanımlanan UN x 2 makinesinin bir birlik sayıyı amaçlandığı gibi iki katına çıkardığını kontrol edebiliriz.

EUC örneğinde, daha iyi bir fikir edinmek için daha açıklayıcı bir sayı çifti, Örneğin 6 ve 8 denenebilir. Okuma cihazının yine O başlangıç durumunda olduğunu varsayalım. Bant başlangıcında aşağıdaki şekilde işaretlenecektir:

...0000000000011111101111111100000...

Turing makinesi, birçok aşamadan sonra durduğu zaman, okuma cihazı, sıfır olamayan hanelerin sağında kalmak üzere aşağıdaki şekilde bir açılım elde ederiz:

...000011000000000000...

EUC (veya UN x 2) cihazının işleyişi ile ilgili bazı incelikleri açıklamak, bilgisayar programlarının pek de yabancısı olmadığımız özelliği gibi, makinenin, karmaşık yapısını anlatmaktan çok daha karmaşık! (Algoritmik bir yöntemin kendisinden beklenen fonksiyonu neden yaptığını tam anlamıyla anlamak *sezgi* gerektirir. Sezgilerin kendileri algoritmik midir? Bizim için daha sonra önemli olacak bir soru bu!) EUC veya UN x 2 örnekleri için şimdi bu konuyu tartışmayacağım. Dikkatli bir okur, bu kitabın amaçlanan kapsamı içerisinde kavramları daha kısa ve öz açıklayabilmek için Eukleides'in algoritmasında ufak bazı değişiklikler yaptığımı fark etmiş olmalı. Buna rağmen, 11 farklı içsel durum için 22 basit komutu içeren EUC tanımı yine de karmaşıktır. Karmaşıklık büyük ölçüde sistemin düzeninden kaynaklanmaktadır. Örneğin, 22 komuttan yalnız 3'ü bant üzerindeki işaretlerin değiştirilmesi ile ilgilidir! (UN x 2 için dahi, yarısı işaret değiştirmekle ilgili 12 komut kullandım.)

Sayısal Verilerin İkilik Gösterimi

Birlik sayı sistemi, büyük sayıların kodlanması için son derece yetersizdir. Bunun yerine, daha önce açıkladığımız gibi çoğunlukla ikilik sayı sistemini kullanacağız. Ancak, bunu doğrudan, bantı iki basamaklı sayı gibi okumaya kalkışarak yapamayız. Sayının iki basamaklı kodunun ne zaman sona erdiğini ve sağ tarafta boş bantı temsil eden sonsuz 0'lar serisinin ne zaman başladığını bildirmenin bu durumda hiçbir yolu yoktur. Üstelik, Eukleides'in algoritmasında öngörülen sayı çiftlerinde [2] olduğu gibi çeşitli sayıları yüklemek zorunda kalacağız. Bir basamaklı sayıların ikilik gösterim sistemine dahil 0'lardan veya 0'lar dizisinden, sayılar-arası aralıkları ayırt edemeyiz. Ayrıca, sayıların yanı sıra, her çeşit karmaşık komutları da veri bantına dahil etmek isteyebiliriz. Bu zorlukların üstesinden gelmek için, *büzülüm* olarak adlandırdığım bir yöntem uygulayalım. Bu yöntemde, 0'lar ve 1'lerden oluşan (1'lerin sonlu toplam adediyle birlikte) herhangi bir dizi, sadece iki basamaklı sayı gibi okunmaz; bu dizinin yerine, ikinci sıradaki her bir hanenin, ilk sıranın birbirini izleyen 0'lar, 1'ler, 2'ler, 3'ler, vs'den oluşan yeni bir dizi geçer. Örneğin

Şimdi 2, 3, 4,... sayılarını işaretler veya herhangi bir türden komutlar olarak okuyabiliriz. 2 sayısı sadece 'virgül' görevini

0100010110101011010001110101011110 0110

dizisi yerine

```
010 0 0101101010110 0 01110101011110 0110
 |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1  0  0  1  2  11 2  1  0  0  3  1  1  4  0  2
```

üstlenirken, 3, 4, 5,... sayılarını, isteğimize göre, 'eksi işareti', 'artı işareti', 'çarpı işareti', 'işlemi tekrarla' vs. gibi komutlar için kullanabiliriz. Artık, daha büyük rakamlarla birbirinden ayrılmış 0 ve 1'lerden oluşan dizilere sahibiz. Bu dizilerin her biri ikilik sistemde yazılmış basit sayıları gösterir. Buna göre yukarıdaki tam dizi ('2' yerine Virgül' okunarak) şöyle okunur:

(1001 ikilik sayısı) virgül (11 ikilik sayısı) virgül...

1001, 11, 100, 0 sayıları için standart '9', '3', *'4',* '0' rakamlarım kullanarak dizinin tümü için [9, 3, 4 (komut 3) 3 (komut 4) O,] sonucuna ulaşırız.

Bu süreç bize özel olarak dizinin sonunda sadece virgül kullanmak suretiyle bir sayının tanımını sona erdirmek (ve bunu yapmakla sayıyı, sağ tarafta boş bantın sonsuz uzantısından ayırmak) olanağını verir. Ayrıca, sayıları ayırmak için virgüller kullanabileceğimiz tek bir O'lar ve 1'ler dizisi olması nedeniyle, ikilik sistemde yazılmış herhangi bir sonlu doğal sayılar dizisini kodlamamızı sağlar. Şimdi bunu örnekliyelim:

5,13, 0, 1, 1, 4, dizisini alalım. İkilik sistemde bu şöyle ifade edilir:

101, 1101, 0, 1, 1, 100,

Bu dizi bant üzerinde *açılım* (yani büzülüm yönteminin tersi) yöntemiyle kaydedilir.

Bu gösterimi basit şekilde gerçekleştirmek için, iki haneli sayıların özgün sırasını değiştiririz;

. . .0000100101101010010110011010110101101000110 00 . . .

0 → 0
1 → 10
, → 110

ve sonra her iki yöne sınırsız 0'lar ekleriz. Daha açık yazarsak, yakardaki dizinin banta nasıl uygulandığını açıkça görürüz: Sayı kümelerini göstermek için oluşturulan bu sisteme *açılmış ikilik sayı sistemi* diyorum. (Örneğin 13'ün açılmış ikilik sayı sisteminde gösterimi 1010010'dur.)

Söz konusu sayı sistemi ile ilgili son bir noktaya daha değinmeliyim. Bu bir teknik ayrıntı olabilir ama konunun eksiksiz açıklanmış olması için gereklidir. [3] Doğal sayıların ikilik (veya ondalık) gösteriminde bir noktada hafif bir fazlalık vardır. Şöyle ki: Bir kodun sol tarafdaki uç noktasına yerleştirilen 0'lar 'sayılmaz' ve normal olarak çıkarılır. Örneğin, 00110010 ile 110010 ikilik sayıları aynıdır (0050 ile 50'in aynı ondalık sayının göstermesi gibi). Bu fazlalık 000 veya 00 veya O olarak yazılabilir, aslında, mantıksal açıdan, boş bir aralık da sıfırı gösterebilir! Basit bir sistemde bu durum büyük karışıldığa sebep olabilirse de, yukarıda tanımlanan sistem için uygundur. Bu nedenle, iki virgül arasında kalan sıfır, yan yana iki virgül („) olarak da yazılabilir ve bant üzerinde bir 0 ile ayrılmış iki 11 çiftiyle kodlanabilir:

...001101100...

Dolayısıyla yukarıdaki altı sayının ikilik sistemde kodlanması şöyledir:

101,1101,,1,1,100,

ve bant üzerinde açılmış ikilik sayı sistemine uygun olarak şeklinde gösterilir (Açılmış sistemle daha önce yazdığımız koda göre bir sıfır eksik...).

Artık örneğin Eukleides algoritmasını, açılmış ikilik sistemle

. . .0000100101101010010110110101101101100011000. . .

yazılmış 6 ve 8 sayılarına uygulayacak bir Turing cihazı tasarlayabiliriz. 6 ve 8 sayı çifti için daha önce kullandığımız,

...0000000000011111101111111100000...,

yerine ikilik sayı sistemindeki sırasıyla, 110 ve 1000 gösterimini kullanırsak; 6, 8 sayı çifti açılmış ikilik sistemde bant üzerine

...0000010100110100001100000O...

şeklinde kaydedilir. Kısalık bakımından, bu gösterimin birlik sisteme göre fazla bir yarar sağlamadığı görülüyor. Ancak, 1583169 ve 8610 gibi (ondalık) bir sayı çiftini ele aldığımızı düşünelim. Bu sayı çiftinin ikilik sistemdeki gösterimi

1 10000010100001000001, 10000110100010,

olacaktır. Buna göre bant üzerinde

0010100000010010000010000001011010000010100100001001l0

şeklinde kaydedilecektir. Bu gösterim yalnız *bir* satıra sığarken, teklik sistem söz konusu olsaydı bantın '1583169, 8610' sayı çiftini kodlaması bu kitabın sayfalarından fazlasını kaplardı!

Sayılar açılmış ikilik sistemde ifade edildiği zaman Eukleides'in algoritmasını uygulayan bir Turing makinesi, istenirse, teklik sistem ile açılmış ikilik sistem arasında çeviri yapan uygun bir çift

yardımcı algoritmanın EUC'a eklenmesiyle basit bir şekilde elde edilebilir. Ancak bu süreç, teklik sayı sistemi hâlâ 'içsel' olarak var olacağı, bu nedenle makinenin yavaşlamasına ve bu durumda (bantın sol tarafında) gerekecek 'müsvedde kağıt' miktarının artmasına yol açacağı için son derece elverişsiz olacaktır.

Bunun yerine, bir Turing makinesinin açılmış ikilik sistemde nasıl işlem yapacağını göstermek amacıyla Eukleides'in algoritmasından çok daha basit bir işlemi, bir doğal sayıya *bir ekleme* sürecini deneyelim. Bu işlem (XN + 1 adını verdiğim) Turing makinesi tarafından yapılabilir:

$$0O \rightarrow 0OR,\quad 0l \rightarrow 1lR,\quad 1O \rightarrow 0OR,\quad 1l \rightarrow 10lR,$$
$$10O \rightarrow 11OL,\quad 10l \rightarrow 10lR,\quad 11O \rightarrow 0lSTOP,\quad 11l \rightarrow 100OL$$
$$100O \rightarrow 101lL,\quad 100l \rightarrow 100lL,\quad 101O \rightarrow 110OR,$$
$$101l \rightarrow 101R,\quad 110l \rightarrow 111lR,\quad 111O \rightarrow 11lR,$$
$$111l \rightarrow 111OR.$$

Yine meraklı bir okurum, Turing makinesinin işlemi gerçekten yapıp yapamayacağını, diyelim, ikilik bir sistemde yazımı 10100111 olan 167 sayısına uygulayarak denemek isteyebilir. Buna göre bant üzerinde okunacak kod:

...0000100100010101011000...

olacaktır. İkilik sistemde bir sayıya 1 eklemek için, son 0'u 1'e çevirmemiz ve sonra bunu izleyen bütün 1'leri 0'larla değiştirmemiz yeterlidir. Buna göre, 167 + 1 = 168 bant üzerinde

10100111 + 1 = 10101000.

olarak yazılır. 'Bir-ekleyen' Turing makinemiz, yukarıda verilen kodu,

...0000100100100001100000...

koduna çevirecektir ve gerçekten de böyle yapar.

Yalnızca bir eklemek gibi basit bir işlemin bile, onbeş komut ve sekiz farklı içsel durum kullanan bu süreç kapsamında ne kadar karmaşık olduğuna dikkatinizi çekerim! Birlik sistemle işlem yapmak elbette daha basittir, çünkü 'bir ekleme' işlemi için sadece 1'ler dizisini bir 1 daha ekleyerek uzatmak yeterlidir. Bu nedenle, UN + 1 makinemizin daha temel düzeyde işlem gerçekleştirmesine şaşmamak gerekir. Ancak çok büyük sayılarla UN + 1, aşırı uzunlukta bant gerektireceği için son derece yavaş çalışabilir. Bu nedenle kapsamlı bir açılmış ikilik sayı sistemini kullanması nedeniyle daha karmaşık bir makine olan XN +1 'i kullanmak daha yararlı olacaktır.

Bu arada 'ikiyle çarpım' adı verilen ve Turing makinesinin birlik sistemden çok açılmış ikilik sistemle daha kolay uygulanabileceği bir işleme de değinmek istiyorum. Aşağıdaki işlemlerle tanımlanan XN x 2 Turing makinesi;

$$0O \rightarrow 0OR,\quad 0l \rightarrow 1lR,\quad 1O \rightarrow 0OR,\ 1l \rightarrow 10OR,$$
$$10O \rightarrow 11lR,\quad 11O \rightarrow 0O\ STOP$$

ikiyle çarpım işlemini kolaylıkla gerçekleştirirken, teklik sistemde daha önce tanımladığımız UN x 2 işlemi aynı işlemi çok daha karmaşık biçimde gerçekleştirir.

Bu örnekler bize Turing makinelerinin çok temel düzeyde neler yapabilecekleri hakkında fikir

vermektedir. Tahmin edilebileceği gibi, karmaşık işlemlerin yapılması gerektiğinde bu cihazlar son derece karmaşık olabilirler ve olmaktadırlar. Bu cihazların en üst kapsamı nedir? Şimdi bunu ele alalım.

Church-Turing Tezi

İnsan, bir kez basit Turing makineleri inşa etmeye alıştı mı, toplama, çıkarma, çarpma vb. çeşitli aritmetik işlemlerinin hepsinin özel Turing makineleriyle gerçekten yapılabildiğini görerek ikna olacaktır. Kalanlı bölme işleminde olduğu gibi sonuçta bir doğal sayı çiftinin çıktığı veya sonucu, zorunlu olarak sonlu fakat büyük miktarda sayılar kümeleri halinde alındığı işlemler de Turing makinelerince gerçekleştirilebilir. Ayrıca, hangi aritmetik işlemini yapması gerektiğinin önceden belirlenmediği fakat işlemle ilgili komutların banta yüklenmiş olduğu durumlar için de Turing makineleri inşa edilebilir.

Bu gibi durumlarda, herhangi bir aşamada yapılması gerekli işlem, makinenin daha önceki bir aşamada yapmak zorunda olduğu işlemin sonucuna bağlı olabilir ('Bu işlemin sonucu şundan-şundan daha yüksekse bunu yap; değilse onu yap'). Aritmetik veya basit mantıksal işlemleri uygulayabilen Turing makineleri yapılabileceğine inandıktan sonra, bunların daha karmaşık algoritmik nitelikte işlemleri de nasıl yapabileceklerini düşünmek zor olmayacaktır. Bu fikre alıştıktan bir süre sonra *herhangi bir mekanik işlemi* gerçekleştiren makineler üretebileceğinize kendinizi inandırabilirsiniz! Matematiksel olarak bir mekanik işlem, böyle bir cihazla yapılabilecek işlem olarak tanımlanabilir. 'Algoritma' ismi ve 'hesaplanabilir' 'yinelenebilir' ve 'etkin' gibi sıfatların hepsi, Turing makineleri gibi teorik makineler tarafından gerçekleştirilebilen mekanik işlemleri tanımlamak için matematikçiler tarafından kullanılır. Bir yöntem yeterince kesin ve mekanik ise, bu yöntemi uygulayacak bir Turing makinesinin gerçekten bulunabileceğine inanmak zor değildir. Bu inanç, (yani Turing'in inancı) ne de olsa, Turing makinesi kavramını, giriş bölümümüzde savunan görüşümüzün temelinde yatmaktadır.

Diğer taraftan, söz konusu makinelerin tasarımının belki gereğinden fazla sınırlayıcı olduğu düşünülebilir. Bir seferinde yalnız bir ikilik hane (0 ve 1) okuması ve *tek* bir-boyutlu bant boyunca yalnız bir aralık hareket etmesi ilk bakışta sınırlayıcı görülebilir. Aynı anda çalışan ve birbiriyle bağlantılı çok sayıda okuma cihazlarıyla donanımlı dört veya beş, veya belki bin bağımsız bant neden olmasın? Yalnız bir-boyutluda ısrarlı olmak yerine O'lar ve 1'lerin oluşturduğu düzlemsel bir sisteme ve (belki üç-boyutlu bir sisteme) neden izin verilmesin? Daha karmaşık sistemlerinin veya alfabelerin kullanımına neden olanak sağlanmasın? Gerçekte, bunlardan bazıları işlemlerin ekonomik olması açısından (birden fazla bant kullanabilseydik elbette daha ekonomik olurduk) fark yaratabilirse de hiçbirisi, ilke olarak şu anda gerçekleştirilenler açısından en ufak bir fark yaratmazdı. Yapılan işlemlerin sınıfı yani 'algoritmalar' başlığı altında toplanan işlemler, (veya 'hesaplamalar' veya 'etkin yöntemler' veya 'tekrarlı işlemler'), makinelerimizin tanımını bütün bu yönlerden aynı anda genişletmiş olsak dahi, kesinlikle değişmezdi!

Makine, bant üzerinde ihtiyacı olan boş alanı bulabildiği sürece, tek banttan fazlasına gerek yoktur. Boş alan arayışı içerisinde, veriler, bant üzerinde bir yerden diğerine taşınabilir. Bu 'elverişsiz' bir durum yaratsa da, ilke olarak gerçekleştirilmesi beklenen sonucu sınırlamaz. [4] Aynı şekilde son zamanlarda insan beyni ile daha yakın benzerlik sağlamak çabasıyla sık sık yapıldığı gibi, birden fazla Turing makinesinin *paralel çalışması* ilke olarak hiçbir şey kazandırmaz (Ancak bazı

koşullarda çalışma hızı artırılabilir). Birbiriyle doğrudan iletişimli iki makine, birbiriyle iletişimli olmayan iki ayrı makineden daha fazlasını gerçekleştiremez; zaten iletişimli olsalardı sonuçta bir makine olurlardı!

Turing'in tek boyutlu banta getirdiği sınırlama nedir? Bantın, ortamı temsil ettiğini düşünürsek, bir-boyutlu bant veya üç-boyutlu uzaydan daha çok bir düzlem varsaymayı tercih edebiliriz. Düzlem, bir-boyutlu bantla kıyaslandığında 'akış şeması' bakımından gerek duyduğumuz süreç için Eukleides algoritmasından daha uygun olacaktır. Ancak, bir akış şemasının 'bir-boyutlu' formda yazılmasının ilkesel hiçbir zorluğu yoktur.[II] (Örneğin, şemanın sözlü tanımının kullanılması suretiyle). İki-boyutlu düzlemde tanımlama yalnız bizim anlatımımız ve anlamamız açısından kolaylık sağlar; ilke olarak gerçekleştirilen sonuç üzerinde etkisi yoktur. Bir nesnenin yerini, iki-boyutlu uzayda işaretleyebileceğimiz gibi bir-boyutlu bant üzerinde de aynı şekilde işaretleyebiliriz (Gerçekte, iki-boyutlu düzlemin kullanımı *iki* bant kullanımıyla tamamen özdeştir. İki bant, iki-boyutlu bir düzlemde yer alan bir noktanın tanımlanması için gerekli iki 'koordinatı' verebilir. Aynı şekilde, *üç* bant, üç-boyutlu uzay görevi yapabilir). Tekrar ediyorum, söz konusu bir-boyutlu kodlama 'elverişsiz' olabilir fakat ilke olarak gerçekleştirilmesi gerekeni sınırlamaz.

Bütün bunlara karşın, bir Turing makinesi kavramının, 'mekanik' olarak adlandırabileceğimiz *her* mantıksal veya matematiksel işlemi kapsayıp kapsamadığını kendimize sorabiliriz. Turing, teorisinin ilk makalesini yazarken bu konuyu, o sırada konu bugün olduğundan çok daha az açık tanımlanabildiği için, fazlaca ayrıntılamak gereğini duymuştur. Turing'in tezi, bu tezden bağımsız olarak (ve aslında ondan biraz daha önce), Amerikalı mantık bilimcisi Alonzo Church tarafından (S.C. Kleene'nin yardımıyla) öne sürülen ve Hilbert'in *Entscheidungsproblem'i* çözmeye yönelik lambda hesabıyla ek destek bulmuştur. Turing'inkinden çok daha dar kapsamlı mekanik bir hesap olmasına karşın, matematiksel yapısının çok daha az karmaşık olmasıyla dikkat çeken Church'ün hesap yöntemini bu bölümün sonunda anlatacağım. Hilbert'in problemine çözüm önerileri getiren başka bilim adamları da vardır (bkz. Gandy 1988). Özellikle, Polonyalı-Amerikan mantık bilimcisi Emil Post'un önerisi dikkate değerdir (Turing'den biraz sonra öne sürdüğü görüşleri Church'ünkinden çok Turing'in görüşlerine yakındır). Çok geçmeden bu fikirlerin hepsinin tamamen birbirlerine özdeş olduğu anlaşılmıştır. *Church-Turing Tezi* adı verilen ortak görüşte, Turing makinesi kavramının, matematik açısından, algoritmik bir yöntemle (veya etkin veya tekrarlı veya mekanik yöntemlerle) anlatmak istediğimiz kavram olduğu vurgulanmıştır. Elektronik bilgisayarların, günlük yaşantımızın alışkanlığı haline geldiği çağımızda pek az kişi fikrin özgün halini sorgulamak gereğini duyabilir. Bunun yerine, kesin *fizik* yasalarına bağlı olmaları nedeniyle *fizik* sistemlerinin (belki de insan beyni dahil) Turing makinelerinin gerçekleştirdiği mantıksal ve matematiksel işlemlerden daha fazlasını mı, daha azını mı, yoksa aynısını mı gerçekleştirdiklerini merak edebilirler. Bana kalırsa, Church-Turing Tezini, matematik açısından, ilk haliyle kabul etmekten çok mutluyum. Öte yandan, söz konusu tezin fizik sistemlerinin davranışı ile ilgisi ayrı bir konu olup, bu kitabın daha sonraki bölümlerinde ilgimizin odağını oluşturacaktır.

Doğal Sayılardan Başka Sayılar

Yukarıda, doğal sayılara dayalı işlemleri ele almış ve her Turing makinesinin değişmez sonlu sayıda farklı içsel durumlara sahip olmasına karşın, zorunlu olarak büyük boyutta *doğal sayıların* işlemlerini yapabileceğine değinmiştik. Ancak doğada eksi sayılar, kesirler veya sürekli kesirler gibi

daha karmaşık sayılar vardır. Eksi sayılar ve kesirler (örneğin, -597/26), pay ve paydalar Turing makinelerince, ne boyutta olursa olsunlar, hesaplanabilirler. Sadece '-' ve '/' işaretleri için uygun koda ihtiyacımız vardır ve bu amaçla, daha önce tanımladığımız açılmış ikilik sayı sistemi kullanılabilir (örneğin, için '3' ve '/' için '4' kullanarak sırasıyla 1110 ve 11110 kodlarını elde edebiliriz). Eksi sayılar ve kesirler böylece, doğal sayılardan oluşan sonlu kümeler halinde işleme dahil edilecekleri için genel hesaplanabilirlik bakımından bize yeni bir şey vermezler.

Aynı şekilde, uzun fakat *sonlu* ondalık ifadeler de bizim için yeni değildir. Çünkü bunlar yalnızca bazı kesirlerden ibarettir. Örneğin, 3,14159265 sayısının sonlu ondalık ifadesi yalnızca 314159265/100000000 kesirsel sayısıdır. Bu ifade irrasyonel π sayısına bir ondalık yaklaşıklık gösterir. Ancak

π= 3,14159265358979...

gibi sonsuz ondalık ifadeler güçlüklere neden olurlar. Bir Turing makinesinin ne girdisi, ne de çıktısı sonsuz ondalık olamaz. Yukarıdaki π ifadesinin birbirini izleyen 3,1,4,1,5,9,.... hanelerinin hepsini kesintisiz çalıştıracağımız çıktı bantı üzerinde sürekli işleyebilecek bir Turing makinesinde buna izin verilmez. Çünkü çıktıyı incelemeden önce makinenin durmasını beklemek zorundayız (zilin çalmasıyla uyarılmalıyız!). Makine STOP pozisyonuna ulaşmadığı sürece çıktı her an değişebilir ve bu nedenle sonuca güvenilmez. Diğer taraftan, STOP pozisyonuna ulaşırsa, çıktı zorunlu olarak sonludur.

Ancak, bir Turing makinesinin, buna çok benzer şekilde, ondalık haneleri kuralına uygun olarak peşpeşe üretmesini sağlayacak bir yöntem vardır. Diyelim π sayısının sonsuz ondalık açılımını bulmak istiyorsak, makinenin 0'a uygulayarak 3 tam sayısını, sonra 1'e uygulayarak birinci ondalık hane 1'i, 2'ye uygulayarak ikinci ondalık hane 4'ü, 3'e uygulayarak üçüncü ondalık hane 1'i, vs. üretmesini sağlayabiliriz. Gerçekte, işlemlerini açıkça tanımlamak biraz karmaşık olsa da, π sayısının tüm ondalık açılımını bu anlamda gerçekleştiren bir Turing makinesi vardır. √2=1,414213562... gibi diğer pek çok irrasyonel sayı bu Turing makinesince üretilebilir. Bir sonraki bölümde ele alacağımız gibi, hiçbir Turing makinesinin üretemeyeceği bazı irrasyonel sayılar da vardır. Turing makinesince üretilebilen sayılara *hesaplanabilir* sayılar (Turing 1937) adı verilirken, üretilemeyen sayılar (büyük çoğunluğu oluşturan sayılar!) *hesaplanamaz* sayılar olarak adlandırılırlar. Bu ve bununla ilgili konulara daha sonraki bölümlerde tekrar değineceğim. Çünkü bu. konu, *gerçek fiziksel nesnenin* (örneğin insan beyni), fizik teorilerimize göre, hesaplanabilir matematik yapıları açısından yeterince tanımlanıp tanımlanamayacağı konusuyla bir ölçüde bağlantılıdır. Hesaplanabilirlik, genellikle matematikte, önemli bir konudur. Bu bağlamda, yalnız sayılara uygulanabilir bir konu olarak değerlendirilmemelidir. Cebir ve trigonometri denklemleri gibi *matematik formüllerini* doğrudan uygulayabilen Turing makinelerine sahip olabiliriz. Yapmamız gereken tüm işlem, ilgili matematik simgelerinin hepsini 0'Iar ve 1'lerden oluşan kodlama sistemine yerleştirmek ve sonra Turing makinesi kavramını uygulamaktır. Genel nitelikte matematik sorularının yanıtlanması için bir algoritma yöntemine gereksinimi bulunan *Entscheidungsproblem*'e karşı saldırısında Turing'in aklından geçen bu yöntemdi. Az sonra bu konuya döneceğiz.

Evrensel Turing Makinesi

Evrensel Turing makinesi kavramını henüz tanımlamadım. Ayrıntıları biraz karmaşık olsa da, temel ilkesini açıklamak çok zor değil: Turing makinesi *T* ile ilgili komutlar listesini, bant üzerinde

kaydedilecek şekilde bir dizi 0'lar ve 1 'ler halinde kodlamak yeterli. Bu bant daha sonra, evrensel Turing makinesi *U* ile ilgili girdinin ilk kısmım oluşturmakta, evrensel makine aynı Turing T'de olduğu gibi, girdinin geri kalan kısmını uygulamaktadır. Evrensel Turing makinesi evrensel bir taklitçidir. Bantın başlangıç kısmı, *T*'yi tıpatıp taklit etmesi için gerekli bütün verileri U'ya verir. Sistemi daha iyi anlamak için önce numaralama ile ilgili sistematik yöntemi inceleyelim: Komutlar listesini 0'lar ve 1'ler kullanarak 'büzülüm' yöntemiyle kodlamamız gerekmektedir. Örneğin, R, L, STOP, (→→) ve virgül simgelerini, sırasıyla 2, 3, 4, 5, ve 6 sayılarıyla temsil ederek, büzülüm yöntemiyle bunları 1 1 0, 1110, 11110, 1 1 1 1 1 0 ve 1111110 olarak kodlayabiliriz. Bu durumda, sırasıyla, O ve 10 olarak kodlanan 0 ve 1 haneleri tabloda ortaya çıkan bu simgelerin gerçek dizileri yerine kullanılabilir. Turing makinesi tablosunda 0 ve 1 ile gösterilen büyük haneleri küçük hanelerden ayırt etmek için farklı bir kodlama sistemi kullanmamız gerekmez. Çünkü ikilik sayı sisteminin sonunda yer alan büyük hanelerin konumu, bunları diğerlerinden ayırt etmek için yeterlidir. Buna göre, örneğin, 1101, 1101 ikilik sayısı gibi okunacak ve bant üzerinden 1010010 olarak kodlanacaktır. Özellikle 00, 00 olarak okunacak, O olarak veya tablodan tamamen çıkarılan bir simge olarak kodlanacaktır. Bir ok işaretini ve bu işaretten hemen önce yer alan simgeleri kodlama zahmetine girmeyerek, komutların sayısal sırasına göre değerlendirmek yoluyla büyük ölçekte ekonomi yapabiliriz. Ancak, bu yöntemi uygularken, gerektiği zaman bir kaç 'sahte' komut vererek, komut listesinde boşluk kalmamasına özen göstermeliyiz (Örneğin XN + 1 Turing makinesi uygulamasında 110O kombinasyonu hiç bir zaman meydana gelmediği için komut listesinde böyle bir kombinasyon yoktur. Bu nedenle 110O→ 0OR sahte komutunu kullanarak, hiçbir şeyi değiştirmeksizin, listedeki boşluğu doldurabiliriz. Aynı şekilde, 101→0OR sahte komutunu XN x 2 makinesine dahil edebiliriz.) Bu çeşit 'sahte' komutlar olmaksızın, listede yer alan bir sonraki komut sırası bozulabilir. L ve R sembolleri, komutları birbirinden ayırmaya yeterli olacağı için her komutun sonuna virgül koymak gerekmez. Buna göre sadece aşağıdaki kodlamayı kullanırız:

0 veya O için O, 1 veya 1 için 10, R için 110, L için 1110, STOP için 11110.

Örnek olarak, XN + 1 Turing makinesini kodlayalım (110O→0OR komutunu içerecek şekilde). Okları ve onlardan hemen önce gelen haneleri ve ayrıca virgülleri atarak aşağıdaki kodu elde ederiz:

0OR 1IR 0OR 10IR 11OL 101R O1STOP 100OL 101IL 100IL 110OR 101R 0OR 111IR 11IR 111OR.

Bu kodu, her 0O'ı atarak ve her O1'in yerine sadece 1 koyarak geliştirebiliriz:

R1lRR1OlR11OL10lRlSTOP100OL101lL100lL110OR
10lRR111lR11lR111OR.

Bant üzerindeki sırasına göre şöyle kodlanır:

1101010110110100101101010011101001011 0101111010000111010010101110100010111010100011010010110110101010101101010101101010100110.

Biraz daha ekonomi yapmak istersek, baştaki 100 sembolünü (boş bantın bu koddan önce gelen sonsuz uzantısı ile birlikte) daima çıkarabiliriz. Bu sembol, makinenin bantı üzerindeki işaretlerin en solundan zorunlu olarak başlangıç alarak birinci işarete ulaşıncaya dek sağa doğru ilerleyebilmesi için tüm Turing makinelerinin ortak özelliği olarak ima ettiğim gibi, 0O→0OR başlangıç komutunu temsil eden OOR'yi gösterir. Aynı şekilde, sondaki 110 simgesini de (ve onu takip ettiği farz edilen sonsuz 0'lar dizisini de) atabiliriz. Çünkü, bütün Turing makineleri tanımlamalarını bu şekilde bitirmek zorundadır (hepsi R, L, veya STOP ile sona ererler), XN + 1 cihazında sonuçta elde edilen

ikilik sayı, Turing makinesinden çıkan sayıdır:

101011011010010110101001110100101101011110100001110100101011101000101110101000
1101001011010101010101101010101101010100.

Standart ondalık sistemde, Turing sayısı aşağıdaki şekilde ifade edilir:

450 813 704 461 563 958 982 113 775 643 437 908.

Sayısı *n* olan Turing makinesine bazen '*n*'inci Turing makinesi diyor ve T_n ile gösteriyoruz. Buna göre XN + 1, 450 813 704 461 563 958 982 113 775 643 437 908'inci Turing makinesidir!

Doğal bir sayıya (açılmış ondalık sistemde) bir eklemek gibi önemsiz bir işlemi yapabilenini bulmadan önce Turing makineleri liste'sinde bunca yol alabilmiş olmamız ilginç! (Kodlamamda bu aşamaya dek pek de yetersiz kaldığımı sanmıyorum; yine de ufak bazı değişiklikler yapmak olasılığıyla karşılaşabiliriz.) Gerçekte, ilgi çekici küçük sayılarla uygulama yapan bazı Turing makineleri vardır. Örneğin, UN + 1, standart ondalık sistemde 177 642 sayısının karşılığı olan

1010110101111101010

ikilik sayısına sahiptir! Yani son derece basit bir Turing makinesi olan UN + 1, 177 642'nci Turing makinesi olmaktadır. Sadece merakımızı gidermek amacıyla bir başka örnek verelim; 'ikiyle çarpım', her iki gösterimde de Turing makineleri listemizde bu örnek arasında bir yer alabilir; çünkü XN x 2'nin sayısı 10389728107, UN x 2'nin sayısı ise 1 492 923 420 919 872 026 917 547 669'dur.

Bu sayıların boyutları açısından, doğal sayıların çoğunluğunun Turing makinelerine işlerlik kazandırmadığını öğrenmek şaşırtıcı olmayabilir. İlk on üç Turing makinesini, bu numaralama sistemine uygun olarak sıralayalım:

T_0: 0O→0OR, 0l→0OR,
T_1: 0O→0OR, 0l→0OL,
T_2: 0O→0OR, 0l→0lR,
T_3: 0O→0OR, 0l→0OSTOP,
T_4: 0O→0OR, 0l→1OR,
T_5: 0O→0OR, 0l→0lL,
T_6: 0O→0OR, 0l→0OR, 1O→0OR,
T_7: 0O→0OR, 0l→???,
T_8: 0O→0OR, 0l→10OR,
T_9: 0O→0OR, 0l→1OL,
T_{10}: 0O→0OR, 0l→1lR,
T_{11}: 0O→0OR, 0l→0lSTOP,
T_{12}: 0O→0OR, 0l→0OR, 1O→0OR.

Bunlardan T_0, karşılaştığı her şeyi silerek, asla durmadan ve asla geri dönmeden yalnız sağa doğru hareket eder. T_x makinesi, bant üzerindeki her işareti sildikten sonra geri sıçrayarak, gide gele biraz

beceriksizce de olsa aynı sonuca ulaşır. T_2 de T_0 gibi hiç durmadan sağa doğru hareket eder. Fakat daha saygılı davranarak, bant üzerindeki hiçbir işareti silmez. Bu örneklerden hiçbirisi, Turing makinesi olarak işe yaramazlar, çünkü hiçbiri asla durmaz. T_3 ilk saygıdeğer makinedir. Birinci 1'i (en soldaki) 0'a değiştirdikten sonra, alçakgönüllülük göstererek durur.

T_4 ciddi bir sorunla karşılaşır. Bant üzerinde birinci 1 'i bulduktan sonra, listede bulunmayan bir içsel durama girer ve bu nedenle bir sonraki aşamada ne yapacağına dair hiçbir komutu yoktur. T_8, T_9 ve T_{10} aynı soranla karşılaşırlar. T_7'nin karşılaştığı soran daha da ciddidir. Kendisini kodlayan 0'lar ve 1'ler dizisinde *beş* adet 1 peşpeşe yer almaktadır: 110111110. Bu dizinin yorumu olmadığı için T_7 bant üzerinde birinci 1 ile karşılaştığı anda takılıp kalacaktır ('n' sayısının ikilik açılımında peşpeşe dört adetten fazla 1 dizisi içeren T_7 makinesini, veya aynı niteliğe sahip herhangi bir T_n makinesini doğru tanımlanamamış makineler olarak adlandıracağım). T_5, T_6 ve T_{12} makineleri, T_0, T_1 ve T_2 makinelerininkine benzer sorunlarla karşılaşırlar. Hiç durmadan sonsuza kadar hareketlerine devam ederler. T_0, T_1, T_2, T_4, T_5, T_6, T_7, T_8 ,T_9 ,T_{10} ve T_{12} makinelerinin hepsi değersizdir! Yalnız T_3 ve işe yarayan Turing makineleri olmakla birlikte işlevleri bakımından ilgi çekici olmaktan uzaktırlar. T_{11}, T_3'ten de mütevazıdır. 1'le ilk karşılaşmasında durur ve hiçbir şeyi değiştirmez!

Listemizde fazlalık bulunduğuna da dikkat çekmeliyiz. T_{12} ve T_6 birbirinin aynısı olup, uygulama bakımından T_0'a benzerler: T_6 ve T_{12}'de 1 içsel durumuna asla girilmez. Listede yer alan yararsız Turing makinelerinin fazlalığı canımızı sıkmasın. Çoğunu eleyerek fazlalığın giderilmesi ve kodlama sistemimizin düzeltilmesi mümkün olabilirdi. Ama bu arada Turing makinesinin, 'n' sayısını okuyan zavallı evrensel Turing makinemizi T_n imiş gibi davranmaya zorlayarak, biraz daha karmaşık hale getirmesine neden olurduk. Doğrusu, bütün işe yaramazlardan (veya fazlalıklardan) kurtulabilseydik buna değerdi. Oysa az sonra göreceğimiz gibi, bunu gerçekleştirmemiz mümkün değildir! Bu nedenle kodlama sistemimizi bırakalım olduğu gibi kalsın. Üzerinde, örneğin,

...0001101110010000...

işaretleri bulunan bir bant, bir sayının ikilik gösterimi olarak yorumlanabilir. O'ların her iki yönde sonsuz olarak devam ettiklerini fakat 1'lerin yalnız bir sonlu sayısı olduğunu hatırlayınız. Ben, 1'lerin sayısının (yani, en az bir 1 vardır) *sıfır olmadığını* da varsayıyorum. İlk ve sonuncu 1'ler de dahil olmak üzere bunlar arasında kalan sonlu simgeler dizisi, yani yukarıdaki örneğe göre,

110111001,

bir doğal sayının (ondalık sistemde 441) ikilik gösterimi olarak okunabilir. Ancak, bu yöntemle *tek* sayılar (ikilik sistemde sonu 1 ile biten sayılar) elde ederiz, oysa tüm doğal sayıları kodlayabilmeliyiz. Bu nedenle kolay yolu seçer ve sondaki 1 'i (yalnızca kodlamanın sona erdiğini gösteren bir işaret olarak kabul edildiği için) kaldırır ve geriye kalanı ikilik sayı olarak okuruz. [5] Bu durumda yukarıdaki örnek için, ondalık sistemde 220'nin karşılığı olarak

11011100,

ikilik sayısını elde ederiz. Bu yöntemin yararı, sıfırın da bant üzerinde kodlanmasını sağlamasıdır:

...0000001000000...

T_n Turing makinesinin sağ taraftan yüklediğimiz bir bant üzerindeki O'lar ve 1 'ler dizisine (sonlu) uygulanmasını ele alalım. Söz konusu diziyi, yukarıda tanımlanan hesap çevrçevesinde, bir sayının, diyelim *m* sayısının iki basamaklı gösterimi olarak almak uygun olacaktır. Peşpeşe aşamalardan sonra

T_n'in sonunda STOP konumuna geldiğini düşünelim. Bu konuma ulaşıncaya kadar makinenin sol tarafta üretmiş olduğu ikilik haneler dizisi, hesabın yanıtıdır. Bunu, bir sayının, diyelim *p* sayısının ikilik kodlaması olarak okuyalım: T_n, *m* sayısına uygulandığında *p* sayısını üretir:

$T_n(m)=p$.

Bu ilişkiye şimdi biraz farklı bir açıdan bakalım. Bu ilişkinin *p* sayısını elde etmek için bir çift sayıya, yani n ve *m* sayılarına uygulanan özel bir işlemin ifadesi olduğunu düşünürüz, *n* ve *m* diye iki sayı verilirse T_n makinesinin *m* sayısını değerlendirmesine bakarak *p* sayısını buluruz. Bu işlem tamamen bir algoritmik yöntemdir. Bu nedenle, özel bir *U* Turing makinesi tarafından uygulanabilir: *U, p* sayısını üretmek amacıyla bir *(n, m)* sayı çiftini işleme sokar. *U* makinesi, tek *p* sonucu elde etmek için hem *n* hem de *m* sayılarını işleme sokmak zorunda olduğu için (*n, m)* çiftini bir bant üzerinde kodlamanın yolunu bulmalıyız, *n* sayısının normal ikilik sistemde yazıldığını ve hemen sonra 11110 dizisi tarafından sona erdirildiğini varsayabiliriz (Doğru tanımlanmış her Turing makinesinin 0'lar, IO'lar, 110'lar, 111O'lar ve 11110'lardan oluşan bir dizi olduğunun ve bu nedenle dört 1'den fazlasını içermediğini hatırlayınız. Bu nedenle, T_n doğru tanımlanmış bir makineyse, 111110 sırasının oluşumu, *n* sayısının tanımının tamamlandığını gösterir). Bunu izleyen her şey *m* tarafından temsil edilen banttan başkası olamaz (yani, hemen 1000... sırası tarafından izlenen *m* sayısı). Bu ikinci kısım, T_n tarafından işleme alınması beklenen banttır.

Örneğin, n = 11 ve m = 6 olarak aldığımız zaman, *U* makinesinin işleme alması gereken bant için işaretler sırası aşağıdaki gibidir:

...0001011111111011010000...

Bu sıranın yapısı:

...0000 (başlangıçtaki boş bant)

1011 (11'in ikilik kodu)

111110 (n sayısının sonu)

110 (6'nın ikilik kodu)

10000... (bantın geri kalan kısmı)

T_n'in *m* ile ilgili her bir işlem aşamasında *U* Turing makinesinin yapması gereken, *m* (yani, T_n'in bantı) ile ilgili hanelerde uygun yer değişikliklerini gerçekleştirmek amacıyla *n* ile ilgili ifadede yer alan hanelerin sırasının yapısını incelemektir. Böyle bir makinenin inşası ilke olarak zor değildir (pratikte ise kesinlikle sıkıcıdır). Kendine ait komutlar listesi, *n* sayısıyla kodlanan bu liste'ye *m* tarafından yapılan girişi, bant hanelerine uygulamanın her aşamasında okuma olanağı sağlar, *m* ve *n* ile ilgili haneler arasında fazlaca 1'leri ve geri hareket meydana gelmesi olağandır ve bu nedenle bu yöntemle son derece yavaş uygulama yapılması kaçınılmazdır. Ancak, böyle bir makine için bir komut listesi elbette temin edilebilir; bu makinelere *evrensel* Turing makinesi adını veriyoruz. Makinenin *n ve m* sayılarına uygulanması U *(n, m)* olarak gösterilirse aşağıdaki eşitlik elde edilir:

U (n, m) = T_n (m) n ve *m* sayılarının her biri için T_n doğru tanımlanmış bir Turing makinesidir. [6] *U* makinesi, *n* sayısı ilk kez verildiği zaman, n'inci Turing makinesini en ince ayrıntısıyla taklit eder! *U* bir Turing makinesi olduğuna göre kendine özgü sayısı da olacaktır;

$U=T_U$

bağıntısını sağlayan bir *u* sayısı bulunacaktır, *u* ne kadar

büyüktür? Aslında *tam olarak* u'nun değeri şöyle verilir (veya, en azından benzer boyutta bir başka sayı):

u=724485533533931757719839503961571123795236067255655963110814479
660650505940424109031048361363235936564444345838222688327876762655 6
144692814117715017842551707554085657689753346356942478488597046934 7
257399885822838277952946834605210611698359459387918855463264409255 2
550582055598945189071653741489603309675302043155362503498452983232 0
651583047664142130708819329717234151056980262734686429921838172157 3
334828230734537134214750597403451843723595930906400243210773421788 5
149276079759763441512307958639635449226915947965461471134570014504 8
167337562172573464522731054482980784965126988788964569760906634204 4
779890219144379328300194935709639217039048332708825962013017737272 0
271862591991442827543742235135567513408422229988937441053430547104 4
368695876405178128019437530813870639942772823156425289237514565443 8
990527807932411448261423572861931183326106561227555318102075110853 3
763380603108236167504563585216421486954234718742643754442879006248 5
827091240422076538754264454133451748566291574299909502623009733738 1
377241621727477236102067868540028935660856968226201419824862169890 2
609130940298570600174300670086896759034473417412787425581201549366 3
938996905817738591654055356704092821332221631410978710814599786695 9
970450968184190629944365601514549048809220844800348224920773040304 3
188429899393135266882349662101947161910701461968523192847482034495 8
977095535611070275817487333272966789987984732840981907648512726310 0
174016678736347760585724503696443489799203448999745566240293748766 8
839751404451665707750060513883991668814072545544665222050724262392 3
792115253181625125363050931728631422004064571305275802307665183351 9
956891397481375049264296050100136519801869456394 98

Sayı kuşkusuz insanı ürkütecek kadar büyük! Gerçekten de ürkütücü boyutta olan bu sayının daha küçük bir boyuta nasıl indirgenebileceğini bilmiyorum. Turing makineleri ile ilgili olarak önerdiğim kodlama yöntemlerinin ve kuralların oldukça mantıklı ve basit olmalarına karşın insanın, bir evrensel Turing makinesinin kodlanması için böylesine büyük boyutta bir sayıyla karşılaşması kaçınılmaz oluyor. [7]

Bütün genel amaçlı modern bilgisayarların sonuçta birer evrensel Turing makinesi olduklarını daha önce söylemiştim. Bu çeşit bilgisayarların mantıksal tasarımının evrensel Turing makinesi ile ilgili tanımlamalarına tamamiyle benzemesi gerektiğini ima etmiyorum. Anlatmak istediğim şu ki, herhangi bir evrensel Turing makinesi ilk olarak uygun bir programla (girdi bantlının birinci kısmı) donattığımız zaman bu makineyi, herhangi bir Turing makinesinin davranışını taklit edebilir duruma getirebilirsiniz! Yukarıdaki tanımlamalara göre, program salt bir sayı *(n* sayısı) biçimini alır, fakat Turing'in özgün tanımının birçok varyasyonunun bulunması nedeniyle başka yöntemler de kullanılabilir. Aslında ben de tanımlamalarımda, Turing'in özgün tanımından bir hayli saptım. Ancak, tanımlama farklılıklarının hiçbiri burada ele aldığımız konu bakımından önemli değildir.

Hilbert Probleminin Çözümsüzlüğü

Gelelim Turing'in tezinin temelini oluşturan sava... Hilbert'in geniş kapsamlı *Entscheidungsproblem'*i: Geniş kapsamlı, fakat iyi tanımlanmış bir sınıfta bulunan tüm matematik problemlerini çözümlemek için mekanik bir yöntem var mıdır? Turing bu soruyu, *m* sayısına uygulanan T_n makinesinin gerçekten STOP konumuna geçip geçmeyeceğine karar vermek problemi olarak yorumlamış, problem, *durma problemi* adıyla anılmıştır. *Herhangi* bir *m* sayısı (örneğin, *n* = 1 veya 2, veya STOP komutunun yer almadığı herhangi bir başka problem) için makinenin durmasını sağlayan bir komut listesi hazırlamak kolaydır. Hangi sayı verilirse verilsin (örneğin *n* = 11) makinenin her zaman durmasını sağlayan birçok komut üstesi de vardır; bazı Turing makineleri bazı sayılara işlem yapınca dururken, diğer bazı sayılar işlem yaptıklarında asla durmazlar. Durmaksızın sonsuza kadar uzayıp giden kuramsal bir algoritmanın pek bir yararının olmadığı söylenebilir. Böylesi algoritma bile değildir. Bu nedenle, *m* sayısına uygulanan T_n'in yanıt verip vermeyeceğine karar verebilmek önemli bir sorudur! Yanıt vermezse (yani işlem durmazsa), bunu aşağıdaki gibi ifade edebilirim:

$T_n(m) = \square$.

(Bu gösterime yukarıda değinirken T_4 ve T_7 makinelerinde olduğu gibi, uygun bir komut alamadığı için herhangi bir aşamada bir sorunla karşılaşan Turing makinesinin durumu da dahil edilebilir. Aynı zamanda, görünüşte başarılı makinemiz T_3 bile, ne yazık ki, bu durumda işe yaramaz olarak nitelendirilmelidir: $T_3(m) = \square$ olacaktır çünkü T_3 uygulamasının sonucu daima boş bir banttan ibarettir; oysa hesap sonucunu bir sayıyla ifade edebilmemiz için çıktıda hiç olmazsa bir tane 1'e ihtiyacımız vardır! Ancak T_{11} bir 1 ürettiği için kullanışlıdır. Çıktısı, 0 sayılı banttır ve buna göre bütün *m* sayıları için $T_{11}(m) = 0$ ifadesini kullanabiliriz).

Turing makinelerinin ne zaman duracağı matematikte önemli bir konudur. Örneğin, şu denklemi ele alalım:

$(x + 1)^{w+3} + (y + 1)^{w+3} = (z+1)^{w+3}$

(Teknik matematik denklemlerden hoşlanmıyorsanız hemen vazgeçmeyin! Bu denklem yalnız bir örnek olarak verilmiştir, ayrıntılarıyla anlamanız gereksizdir). Söz konusu denklem, matematikte ünlü, belki en ünlü çözülmemiş problemle ilişkilidir. Problem şöyle: Bu denklemi sağlayan *w, x, y, z* doğal sayılarından oluşan herhangi bir küme var mıdır? On yedinci yüzyılda yaşamış Fransız matematikçisi Pierre de Fermat (1601-1665) tarafından Diophantus'un *Arithmetica* kitabının bir sayfasının kenarında not edilmiş ve "Fermat'nın Son Teoremi' olarak tanınan ünlü teorem yukarıdaki denklemin *asla* çözülemeyeceğini söyler.[III] [8] Asıl mesleği avukatlık olan ve Descartes'la aynı çağda yaşamış olan Fermat, döneminin en iyi matematikçilerindendi, *Arithmetica'nın* bir kenarına sığdırmaya çalıştığı tezinin 'gerçekten harika bir kanıt' olduğunu iddia eden Fermat'ın kanıtını bugüne değin hiç kimse yeniden inşa edemediği gibi bir karşıt-örnek bulabilen de çıkmamıştır!

Dörtlü sayı grubu *(w, x, y ,z)* verildiğinde denklemin çözümlenip çözümlenemeyeceğine karar vermenin yalnızca bir hesap meselesi olduğu açıktır. Bir bilgisiyar algoritmasının dörtlü sayı gruplarını teker teker işleme soktuktan sonra ancak denklem çözülünce durduğunu düşünebiliriz (Daha önce değindiğimiz gibi, sonlu kümeleri tek bant üzerinde ve hesaplanabilir tarzda, yani yalnızca tek tek sayılar olarak kodlama yöntemleri vardır; böylece, söz konusu sayıların doğal sırasını izlemek suretiyle dört sayıyı teker teker işleme sokabiliriz). Bu algoritmanın durmaksızın

uygulandığını kanıtlayabilirsek Fermat'ın tezinin kanıtına da sahip oluruz.

Diğer birçok çözümlenmemiş matematik problemini, Turing makinesinin durma problemine dayanarak aynı yöntemle ifade edebiliriz. Buna örnek 'Goldbach sanıtı'dır: 2'den büyük her çift sayı iki asal sayının toplamıdır.[IV] Verilen bir sayının asal olup olmadığına karar vermek algoritmik bir işlemdir çünkü yalnız *sayının kendinden küçük sayılarla bölünebilirliğinin denendiği sonlu* hesap gerektirir. 6, 8, 10, 12, 14,... çift sayılarını sırayla aşağıda gösterildiği gibi, tek sayılardan oluşan ikililere ayırmanın bütün yollarını deneyen ve ikiye ayırdığı sayının her iki üyesinin de asal olup olmadığını sınayan bir Turing makinesi icat edebiliriz:

6 = 3 + 3, 8 = 3 + 5, 10 = 3 + 7 = 5 +5, 12 = 5 +7 14 = 3 + 11 = 7 + 7,...

(2 dışında tüm asal sayılar tek sayılar olduğu için 2 + 2 hariç, *çift* sayıdan oluşan ikilileri test etmemiz gerekmez). Makinemiz, ayrıldığı ikililerden *hiçbiri* asal sayıdan oluşmayan ilk sayıya ulaştığı zaman duracaktır. Bu durumda, Goldbach sanıtına karşıt-örnek, yani, iki asal sayının toplamı olmayan ve 2'den büyük tek sayıyı buluruz. Böyle bir Turing makinesinin durup durmayacağına karar verebilsek, Goldbach sanıtının doğruluğuna karar vermenin bir yolunu da bulmuş oluruz.

Bu noktada bir soru kendiliğinden ortaya çıkıyor: Herhangi bir Turing makinesinin (belirli girdi verildiğinde) durup durmayacağına nasıl karar vermeliyiz? Turing makinelerinin çoğu için yanıtlaması kolay bir soru olabilirse de yanıt bazen, yukarıda gördüğünüz gibi, çözülmemiş bir matematik probleminin çözümünü gerektirebilir. Öyleyse, genel soruyu -yani durma problemini- tamamen otomatik olarak yanıtlamak için algoritmik bir yöntem var mıdır? Böyle bir yöntemin gerçekten bulunmadığını Turing göstermiştir.

Turing'in bu konudaki kanıtı şöyledir: Önce, böyle bir algoritmanın var olduğunu düşünelim.[V] Öyle bir *H* Turing makinesi bulunmalıdır ki *n'inci* Turing makinesinin *m* sayısına işlem yapınca durup durmayacağına karar verebilsin. Diyelim ki durmazsa çıktı bant üzerine O olarak işlensin, durursa 1 olarak işlensin:

$$H(n;m) = \begin{cases} \text{O eğer} & T_n(m) = \square \text{ ise,} \\ \text{1 eğer} & T_n(m) \text{ durursa.} \end{cases}$$

Burada *U* evrensel makinesi için geçerli kural *n* ve *m* sayılarının kodlanmasına uygulanabilir. Ancak bu aşamada, bazı *n* sayıları için (örneğin *n = 7) T_n* doğru tanımlanmadığı için, teknik bir sorunla karşılaşabiliriz: 111101 işareti, bant üzerinde *n'ı* m'den ayırmakta yetersiz kalabilir. Bu sorunun üstesinden gelmek için daha önce yaptığımız gibi, *m'*i normal ondalık sistemde kodlarken, *n*'in açılmış ondalık sistemde kodlandığını varsayalım. Böylece 110 işareti, *n'i* m'den ayırmaya yeterli olacaktır. *U(n, m)* ifadesindeki *virgülün* yerinde *H(n; m)* ifadesinde *noktalı virgülün* kullanılması bu değişikliği gösterecektir.

Şimdi, olası tüm girdileri uygulayan olası tüm Turing makinelerinin tüm girdilerini liste halinde içeren sonsuz bir dizge düşleyelim.

$m \rightarrow$	0	1	2	3	4	5	6	7	8	. . .
$n \downarrow$										
0	□	□	□	□	□	□	□	□	□	. . .
1	0	0	0	0	0	0	0	0	0	. . .
2	1	1	1	1	1	1	1	1	1	. . .
3	0	2	0	2	0	2	0	2	0	. . .
4	1	1	1	1	1	1	1	1	1	. . .
5	0	□	0	□	0	□	0	□	0	. . .
6	0	□	1	□	2	□	3	□	4	. . .
7	0	1	2	3	4	5	6	7	8	. . .
8	□	1	□	□	1	□	□	□	1	. . .
.	.				.				.	. . .
.	.				.				.	
.	.				.				.	
197	2	3	5	7	11	13	17	19	23	. . .
.	.				.				.	
.	.				.				.	
.	.				.				.	

Bu dizgenin *n*'inci sırası, T_n makinesinin 0, 1, 2, 3, 4,... girdileri üzerine işlem yaptığında çıktıların ne olduğunu göstermektedir. Yukardaki tabloda biraz hile yaparak, Turing makinelerine *gerçekte* sahip olduklarından farklı numaralar verdim. Böyle yapmakla listenin daha başında sıkıcı görünmesini önlemeye çalıştım. Çünkü aksi halde *n* sayısının 11'den küçük değere sahip olduğu tüm makineler □'lerden başka hiçbir şey üretmez ve *n* = 11 için O'lardan başka hiçbir şey elde edemezdik. Bu nedenle listenin gerçekte nasıl göründüğü hakkında bir izlenim sağlamak amacıyla çok daha uygun bir kodlama yarattım.

Bu dizgenin bir algoritmayla, gerçekten *hesaplanması* gerektiğini iddia etmiyorum (Zaten biraz sonra göreceğimiz gibi böyle bir algoritma yok). Gerçek listenin nasıl olduysa, belki de Tanrı tarafından, önümüze seriliverdiğini hayal etmemiz bekleniyor. Bu tür hesaplar sonsuza dek süreceği için bir □'yi hangi konuma ne zaman yerleştireceğimizden asla emin olamayız; işlemi gerçekleştirmeye kalkıştığımız zaman ortaya çıkacak □'ler karşılaşacağımız zorlukların başlıca kaynağı haline gelir.

Teorik, *H*'i kullanabilseydik tabloyu üretmek için gerekli yöntemi bulabilirdik. Çünkü *H,* □'lerin nerelerde oluşacağını bize söylerdi. Fakat, bunun yerine, her □'yi, 0 ile değiştirerek □'leri eklemek

için H'i kullanalım. T_n'in m'e uygulanmasından önce H (n; m)'in uygulamasıyla bu gerçekleşir; daha sonra, $H(n; m)$ = 1'in koşuluyla (yani, sadece T_n (n) hesabı gerçekten bir yanıt verirse) T„'in m'ye işlem, yapmasına izin veririz, ve $H(N; m)$ = 0 ise. (yani $T_n(m)=0$ ise) sadece 0 yazarız. Böylece bu yöntemi (yani, $H(n;m)$'in T_n (m)'den önce uygulanmasıyla elde edilen yöntemi)

$T_n(m)$ x $H(n;m)$

olarak yazarız. (Burada, matematik işlemlerinin sıralanması ile ilgili bilinen bir matematik kuralını uyguluyorum: sağdaki işlem önce uygulanmalıdır. Sembolik olarak □ x 0= 0 olacaktır).

Buna göre tablo şu şekilde okunur:

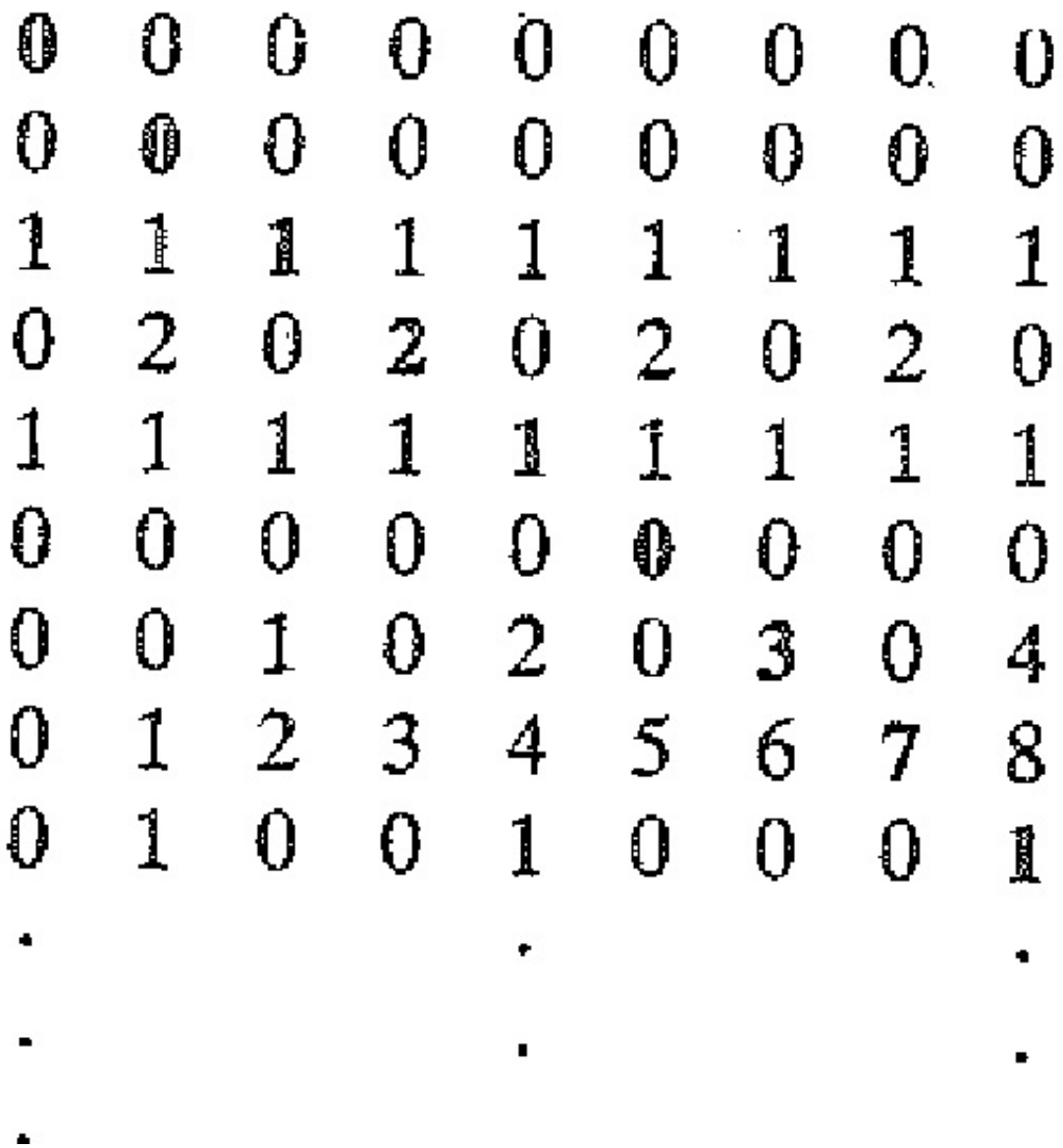
$m \rightarrow$	0	1	2	3	4	5	6	7	8	. . .
n ↓										
0	0	0	0	0	0	0	0	0	0	. . .
1	0	0	0	0	0	0	0	0	0	. . .
2	1	1	1	1	1	1	1	1	1	. . .
3	0	2	0	2	0	2	0	2	0	. . .
4	1	1	1	1	1	1	1	1	1	. . .
5	0	0	0	0	0	0	0	0	0	. . .
6	0	0	1	0	2	0	3	0	4	. . .
7	0	1	2	3	4	5	6	7	8	. . .
8	0	1	0	0	1	0	0	0	1	. . .
.	.				.				.	
.	.				.				.	
.	.				.				.	

H makinesinin varolduğunu düşünsek, bu tablodaki sıralar *hesaplanabilir dizgelerden* oluşmaktadır. (Hesaplanabilir dizge ile anlatmak istediğim, birbirini izleyen değerlerin bir algoritma tarafından üretilebildiği sonsuz dizidir; bir başka deyişle, m= 0, 1, 2, 3, 4, 5... doğal sayılarına uygulandığında dizgenin birbirini izleyen elemanlarını üreten bir Turing makinesi bulunmalıdır). Şimdi bu tablonun iki özelliği üzerinde duralım.

Birincisi, her hesaplanabilir doğal sayılar dizisi en az bir (veya bir kaç) dizge halinde bulunmaktadır. Bu özellik, tablonun orijinalinde □'lerle görülmüştü. Ben yalnız 'işe yaramaz' Turing makinelerinin (en az bir □ üreten makinelerin) yerine bazı sıralar ekledim. İkincisi, H Turing makinesinin gerçekten varolduğu varsayımından hareketle tablo, $T_n(m)$ x $H(n; m)$ yöntemiyle hesaplanabilir şekilde üretilmiştir (Başka bir deyişle, belli bir algoritma ile üretilmiştir). Açık olarak n ve m sayılarına uygulandığı zaman tabloya uygun kayıt üreten bir Q Turing makinesi bulunmaktadır. Bunun için, aynen H Turing makinesinde olduğu gibi, Q'un bantında da n ve m'i kodlayabiliriz:

$Q(n;m) = T_n(m)$ x $H(n;m)$.

Şimdi, Georg Cantor'un "köşegen yöntemi" adı verilen zekice planlanmış ve güçlü yöntemini değişik bir biçimde uygulayalım (Cantor'un adı geçen yöntemini bir sonraki bölümde ayrıntılı anlatacağım). Koyu renk rakamlarla işaretlenmiş bulunan elemanların oluşturduğu ana köşegene dikkat ediniz:



Bu elemanların oluşturdukları 0, 0, 1, 2, 3, 7, 1,... dizisine 1 ekliyoruz:

1, 1, 2, 3, 2, 1, 4, 8, 2,...

Açıkça görüldüğü gibi bu bir hesaplanabilir yöntemdir ve tablomuz da hesaplanabilir tarzda üretilmiş olduğuna göre bize yeni bir hesaplanabilir dizge, gerçekte 1 + *Q (n;n)* dizgesini; yani

$1 + T_n(n)xH(n;n)$

vermektedir *(m* ile *n*'i eşitlemek suretiyle köşegen üzerinde verildiği için). Fakat tablomuz *her* hesaplanabilir dizgeyi içerdiği için yeni dizi de listenin bir yerinde yer alıyor olmalı. Ama durum hiç de böyle değil! Çünkü yeni dizgemiz, ilk kaydın ilk sırasından, ikinci kaydın ikinci sırasından, üçüncü kaydın üçüncü sırasından, vs. farklıdır. Bu açık bir çelişkidir. İşte bu çelişki, kanıtlamaya çalıştığımız olguyu kanıtlar: Turing makinesi *H* gerçekte mevcut değildir! *Bir Turing makinesinin durup durmayacağı konusunda karar verecek evrensel bir algoritma yoktur.*

Bu savı başka şekilde ifade etmek amacıyla diyelim ki Turing makinesi *H* mevcuttur, ve 1 + *Q(n;n)*

algoritması (köşegen işlem!) için bir Turing makinesi sayısı, örneğin *k,* vardır; bu durumda algoritmamız şöyledir:

$1 + T_n(n) \times H(n;n) = T_k(n)$.

Bu ilişkide *n* yerine *k* koyarsak aşağıdaki algoritmayı elde ederiz:

$1+T_k(k) \times H(k;k) = T_k(k)$.

Bu bir çelişkidir, çünkü $T_k(k)$ durduğu takdirde olanaksız bir ilişki ortaya çıkar:

$1+T_k(h) = T_k(k)$

$H(k;k) = 1$ olduğuna göre, $T_k(k)$ durmazsa =0 olur ki, aynı çelişkiyle burada da karşılaşılır:

$1 + 0 = \square$.

Belirli bir Turing makinesinin durup durmayacağı sorusu, mükemmel tanımlanmış bir matematik sorusudur (ve biz daha önce görmüştük ki, aksine, çeşitli önemli matematik problemleri Turing makinelerinin durması olarak ifade edilebilirler). Böylece, Turing makinelerinin durması konusunda karar vermekle ilgili hiç bir algoritmanın varolmadığım göstermekle Turing, (daha farklı yaklaşımıyla Church'ün de gösterdiği gibi) matematik soruları ile ilgili karar vermenin genel algoritmasının varolmayacağını göstermiştir. Hilbert'in *Entscheidungsproblem'*inin çözümü yoktur!

Bu, herhangi bir bireysel durumda, belirli bir matematik sorusunun doğru olup olmadığı hakkında veya başka bir niteliği hakkında, veya belirli bir Turing makinesinin durması hakkında karar veremeyeceğiz demek değildir. Biraz aklımızı, veya hatta sağ duyumuzu kullanarak bunu başarabiliriz (örneğin, bir Turing makinesinin komut listesinde STOP komutu bulunmazsa, veya böyle bir liste *yalnız* STOP komutlarını içerirse, sağ duyumuz bize cihazın durup durmayacağını söylemek için yeterlidir). Fakat *tüm* matematik soruları, *tüm* Turing makineleri ve bunların uygulandığı *tüm* sayılar için geçerli hiç bir algoritma yoktur.

Sanki en azından *bazı* karar verilemez matematik sorularının mevcut olduğunu kanıtlamış gibi görünüyoruz. Oysa, bir şey yapmadık! Tam aksine, biraz sonra göreceğimiz gibi, özellikle kötü sayılar verildiğinde makinenin durup durmamasına karar vermesi kesinlikle olanaksız, ters bir Turing makinesi tablosunun varolduğunu göstermedik. *Tek tek* problemlerin çözümsüzlüğü hakkında bir kelime bile söylemezken, problem *ailelerinin algoritmik* çözümsüzlüğünden söz ettik. Herhangi bir belirli durumda yanıt 'evet' veya 'hayır' olmalıdır ve bu nedenle belirli durumla ilgili kararın bir algoritması elbette vardır. Başka bir deyişle, problemle karşı karşıya geldiğinde duruma göre ya 'evet' veya 'hayır' yanıtını veren algoritma vardır. Algoritmalardan *hangisini* kullanacağımızı bilmeyebiliriz ve kuşkusuz işin zor yanı da budur. Sorun bir problemler ailesi ile ilgili sistematik karar sorunu değil, fakat bir tek problemin matematik gerçeği hakkında karar vermek sorunudur. Algoritmaların, matematik gerçeği hakkında kendi başlarına karar vermediklerini anlamak önemlidir. Bir algoritmanın *geçerliliği* daima dışarıdan sağlanan olanaklarla kanıtlanmalıdır.

Bir Algoritmanın Üstesinden Nasıl Gelinir?

Matematiksel ifadelerin gerçekliğine karar vermek konusu, Gödel'in. teoremi (bkz. IV. Bölüm) çerçevesinde daha sonra ele alınacaktır. Şimdilik, Turing'in teorisinin, aslında, anlatmağa çalıştığımdan çok daha yapıcı ve çok daha az olumsuz olduğunu belirtmek isterim. Bu aşamaya kadar,

durup durmayacağına mutlak anlamda karar verilemez belli bir Turing makinesinin varlığını kanıtlamış değiliz. Gerçekte, Turing'in teorisini dikkatle incelersek, Turing'in yöntemiyle inşa ettiğimiz görünüşte 'özellikle karmaşık' makinelerle ilgili yanıtı, kullandığımız yöntemin bize ima ettiğini anlarız!

Şimdi bunun nasıl gerçekleştiğini görelim. Bir Turing makinesinin ne zaman durmayacağını bazen bize söyleyebilir bir algoritmaya sahip olduğumuzu farz edelim. Daha önce genel hatlarıyla tanımladığımız Turing yöntemi, işlemin durup durmayacağına bu algoritma tarafından karar verilemeyen bir Turing makinesi işlemini açıkça verecektir. Aslında, böyle yaparken, bu işlemle ilgili yanıtı anlamamızı da sağlayacaktır: Turing makinesinin işlemi gerçekten durmayacaktır.

Biraz daha ayrıntılı anlatmak gerekirse, bir Turing makinesinin ne zaman durmayacağına dair bize bazen bilgi veren bir algoritmaya sahip olduğumuzu farz edelim. Daha önce olduğu gibi, bu algoritmayı (Turing makinesi) *H* ile temsil ediyoruz, fakat biliyoruz ki Turing makinesi durmayacaktır:

$$H(n;m) = \begin{cases} 0 \text{ veya } \square \text{ eğer } T_n(m) = \square \text{ ise,} \\ 1, \text{ eğer } T_n(m) \text{ durursa.} \end{cases}$$

böylece $T_n(m)$ □ olduğu zaman $H(n; m)$ = □ bir olasılıktır. $H(n;m)$ gibi bir çok algoritma vardır. (Örneğin, pratikte pek fazla yararı olmasa da, $T_n(m)$ durur durmaz $H(n;m)$ bir 1 üretir!)

Buna göre, yine Turing'in yöntemini izleyerek, fakat bu kez *tüm* □'lerin yerine O'ları koymayarak, elimizde bazı □'ler kaldığım görürüz. Köşegen yöntemimiz, köşegen üzerindeki eleman olarak bize

$1 + T_n(n) \times H(n;n)$

ifadesini vermişti. ($H(n;n)$= □ olursa daima □ elde ederiz. □ x □ = □, 1 + □ = □ olduğunu unutmayınız) Bu, sonucu mükemmel bir hesaptır ve bir Turing makinesince, diyelim *k'inci* , makine tarafından gerçekleştirilmiştir.

Şimdi şu eşitliğe sahibiz:

$1+T_n(n) \times H(n,n) = T_k(n)$.

k'inci köşegen elemana bakalım, yani $n = k$ alalım:

$1+T_k(k) \times H(k;k) = T_k(k)$.

elde ederiz. $T_k(k)$ hesabı durursa bir çelişkiyle karşılaşırız çünkü $T_k(k)$ ne zaman durursa dursun $H(k:k)$'nı 1 olduğu varsayılır ve bu durumda eşitlik tutarsızdır: $1+T_k(k) = T_k(k)$. Öyleyse $T_k(k)$ duramaz, diğer bir deyişle

$T_k(k) = \square$.

Fakat algoritma bunu 'bilemez', çünkü; $H(k;k)=0$ eşitliğini verseydi yine bir çelişkiyle karşılaşırdık (sembolik olarak, geçersiz bir ilişki, 1 + 0 = □ elde ederdik).

Bu nedenle *'k'i* bulabilirsek, yanıtını bildiğimiz algoritmayı alt etmek amacıyla kendi özel işlemimizi nasıl inşa etmemiz gerektiğini öğreniriz! *k'i* nasıl buluruz? Bu zor iş. Yapmamız gereken $H(n;m)$*'in* ve $T_n(m)$*'*in inşasına ayrıntılarıyla bakmak ve sonra

$1+T_n(n) \times H(n,n)$ ifadesinin bir Turing makinesinin sayısını, yani *k'i* buluruz, işlemi ayrıntılı olarak gerçekleştirmek, biraz karmaşık olsa da, mümkündür.[VI] İşlemin karmaşıklığı nedeniyle

$T_k(k)$'ın hesaplanması bizi hiç ilgilendirmeyebilirdi fakat onu, H algoritmasına karşı üstünlük sağlayabilmek için özellikle ürettik! Önemli olan, iyi tanımlanmış bir yönteme sahip olmamızdır. Bize hangi H verilmiş olursa olsun bu yöntemle H karşıtı k'i bulunabilir. $T_k(k)$ ile H'i alt eder ve böylece algoritmanın gerçekleştireceğinden daha iyisini gerçekleştirebiliriz. Algoritmalardan daha iyi olduğumuzu düşünmek, belki bizi biraz rahatlatır!

Yöntem öylesine iyi tanımlanmıştır ki, H verildiğinde k'i üretecek bir *algoritma* bulabiliriz. Bu nedenle, daha fazla ayrıntıya girmeden önce bu algoritmanın H'e uygulanabileceğini anlamalıyız, [9] çünkü algoritma, $T_k(k) = \square$ eşitliğini 'bilir', -yoksa bilmez mi? Algoritma ile ilgili antropomorfik bir terim olan 'bilmek' terimini, tanımlamamı kolaylaştırdığı için kullandım. Ancak, algoritma, izlemesi için kendisine verdiğimiz kuralları izlerken, 'bilmek' işlemini yapan biz değil miyiz? Yoksa, bizzat biz mi, beynimizin yapısından ve çevremizden izlemek için programlanmış olduğumuz kuralları izliyoruz? Konu, yalnızca bir algoritma konusu değil, aynı zamanda neyin gerçek neyin gerçek olmadığına hüküm vermek meselesidir. Bu konulara daha sonra tekrar döneceğiz. Matematiksel gerçek (ve onun algoritmik olmayan özelliği) konusu IV. Bölüm'de ele alınacaktır. Şimdilik, en azından, 'algoritma' ve 'hesaplanabilirlik' terimlerinin *anlamları* hakkında biraz sezgi ve ilgili konularda bir fikir edinmiş bulunmaktayız.

Church'ün Lambda Hesabı

Hesaplanabilirlik kavramı, önemli ve güzel bir matematiksel fikirdir. İlk kez 1930larda -bu kavram gibi temel nitelikli diğer kavramlarla birlikte-matematik bilimine girdiği için aynı zamanda oldukça yeni bir fikirdir ve matematiğin *tüm* alanlarını kapsar (Matematikçilerin çoğu, yine de, bugün bile hesaplanabilirlik sorunlarıyla kendilerini pek yormazlar diyebiliriz). Hesaplanabilirlik fikrinin gücü, kısmen, bazı iyi tanımlanmış matematik işlemlerinin aslında *hesapalanamaz* olmasından kaynaklanır (Turing makinesinin durma veya durmama problemi gibi, öteki örnekleri IV. Bölüm'de göreceğiz). Çünkü, böyle hesaplanamaz işlemler olmasaydı, hesaplanabilirlik kavramı, matematiğin ilgisini çekmezdi. Ne de olsa matematikçiler bilmecelerden hoşlanırlar. Bir matematik işleminin hesaplanıp hesaplanmayacağı konusunda karar vermek onlar için ilgi çekici bir bilmece olabilir. Özellikle ilginç olabilir çünkü böyle bir bilmecenin yanıtı bile hesaplanamaz niteliktedir!

Bir şeyi açıkça belirtmeliyim. Hesaplanabilirlik gerçek bir 'mutlak' matematik kavramıdır. Benim tanımladığım şekliyle 'Turing makineleri' açısından, herhangi bir anlayışın tamamen ötesinde soyut bir fikirdir. Daha önce değindiğim gibi, Turing'in zekice planlanmış, kendine özgü yaklaşımını karakterize eden 'bantlar' ve 'içsel durumlar', vs. gibi kavramlara özel bir önem vermemiz gerekmemektedir. Hesaplanabilirlik fikrini açıklayan başka yollar da vardır. Bunlardan, tarih bakımından birincisi, Amerikalı mantıkçı Alonza Church'ün, Stephen C. Kleene'in yardımıyla geliştirdiği lambda hesabı'dır. Church'ün yöntemi Turing'in yönteminden oldukça farklı ve daha soyuttur. Gerçekte Church'ün düşüncelerini ifade tarzında belirginleştiği gibi aralarında gözle görülür, 'mekanik' denilebilecek pek az benzerlik vardır. Church'ün yönteminin ana fikri, özünde gerçekten *soyuttur*- Church'ün 'soyutlama' olarak tanımladığı bir matematik işlemidir.

Yalnız hesaplanabilirliğin, herhangi bir hesap makinesi kavramından bağımsız olarak, bir matematiksel fikir olduğunu vurguladığı için değil, matematikte soyut fikirlerin gücünü de gösterdiği için Church'ün tasarımının kısa bir tanımını yapmak istiyorum. Bu gibi konulara ilgi duymayan

okurum, bu aşamada, bir sonraki bölüme hemen geçebilir ve konunun akışında fazla bir şey kaçırmış sayılmaz. Ancak, bu gibi okurlar bana biraz daha katlanarak Church'ün tasarımının sağladığı sihirli ekonomiye tanık da olabilirler (bkz. Church 1941).

Bu tasarımda ilgi odağı olarak nesnelerin 'evreni' ile ilgilenmekteyiz ve bunu örneğin

a, b, c, d, ..., z, a', b', ... z', *a", b", ..., a"'*, ... a"", ...

ifadesiyle göstermekteyiz; her harf bir matematik işlemini veya *fonksiyonunu* temsil etmektedir (Üslü harfler, fonksiyonları temsil eden simgelerin sınırsız sayıda bulunmasını sağlar). Bu fonksiyonların 'değişkenleri', yani bu fonksiyonların üzerine işlem yaptığı nesneler, aynı türden başka şeylerdir, yani onlar da fonksiyonlardır. Ayrıca, bir diğerine uygulanan bir fonksiyonun 'değeri'nin de yine bir fonksiyon olması gerekir (Church'ün sisteminde gerçekten harika bir kavramlar ekonomisi var). Bu nedenle,

a = bc

yazdığımız zaman[VII] *'c'* fonksiyonuna uygulanan *'b'* fonksiyonunun sonucunun bir başka 'a' fonksiyonu olduğunu kastederiz. Bu tasarımda, iki veya daha fazla değişkenli bir fonksiyon fikrini ifade etmek zor değildir, *p* ve *g* değişkenlerine bağlı bir *f* fonksiyonunu

((fp) q

olarak ifade edebiliriz (q'a uygulanan *fp* fonksiyonunun sonucu). Üç değişkenli bir fonksiyon için

((fp) *q) r*

ifadesini kullanabiliriz ve bu böylece sürer gider.

Şimdi sıra, güçlü *soyutlama* işlemine geldi. Bunun için Yunan harfi λ (lambda)'yı kullanıyor ve hemen ardından Church'ün fonksiyonlarından birini temsil eden bir harf, örneğin *x* harfini ('sahte değişken' adını verdiğimiz harfi) koyuyoruz. *'x'* değişkeninin yer aldığı her kare-parantez, ifadenin tümünü izleyen herhangi bir simgenin yerini tutabilen bir 'boşluk' olarak kabul edilebilir. Buna göre

λx. [fx]

yazdığımız zaman fonksiyonun, diyelim a'ya uygulandığında *'fa'* sonucunu verdiğini anlatırız. Başka deyişle,

(λx. [fx]) *a = fa*

Bir başka deyişle, λx. [fx]) ,*'f* fonksiyonundan ibarettir, yani:

λx. [fx]) = *f*

Bunlar ilk bakışta fazla bilgiççe ve önemsiz göründüğü için ana fikir gözden kaçabilir. Bu nedenle, alışık olduğumuz okul matematiğinden bir örnek alalım ve *f* fonksiyonunun, bir açının sinüsünü almak gibi trigonometrik bir işlem olduğunu varsayalım. Bu durumda 'sin' soyut fonksiyonu şu şekilde ifade edilir:

λx. [sin x] = *sin.*

(x 'fonksiyonunun' bir açı olarak nasıl kabul edilebileceğini merak etmeyin. Kısa bir süre sonra, sayıların bile fonksiyonlar olarak kabul edildiği işlemler yapacağız; ve bir açı da bir tür sayıdır). Bu aşamaya kadar gerçekten oldukça önemsiz görünebilir örneklerimiz. Fakat, 'sin' ifadesinin icad edilmemiş olduğunu fakat sin *x* için kuvvet serisi ifadesini bildiğimizi farzedin:

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \ldots$$

Buna göre

$$sin = \lambda x.\left[x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \ldots\right]$$

tanımını verebiliriz.

Herhangi bir standart 'fonksiyonel' ifadesi bulunmayan, diyelim 'küp almanın altıda-biri' işlemini, daha da kolay, tanımlayabiliriz:

$$Q = \lambda x.\left[\frac{1}{6}x^3\right]$$

ve örneğin, şu sonucu alırız:

$$Q(a+1) = \frac{1}{6}(a+1)^3 = \frac{1}{6}a^3 + \frac{1}{2}a^2 + \frac{1}{2}a + \frac{1}{6}.$$

Church'ün basit fonksiyonel işlemlerinden alınan ifadeler konumuzla daha ilgili olabilir; örneğin,

λf. [f(fx)]

Bu, bir başka fonksiyona örneğin 'g' fonksiyonuna uygulandığında, *x'e* iki kez üst üste 'g'nin uygulandığı işlemi üretir:

*(*λf. [f(fx)]) g = *g (gx)*

'Soyut tarzda' ifade edersek:

λf. [λx. [f(fx)]]

veya kısa olarak:

λfx. [[f(fx)]]

Bu, g'ye uygulandığında, g'nin iki kez tekrarlanmış fonksiyonunu veren işlemdir. Aslında bu, tam Church'ün 2 doğal sayısı ile özdeşleştirdiği fonksiyondur:

2 = λfx [f(fx)]

ve buna göre (2 g) y = g (gy). Benzer şekilde, Church şu tanımlamaları yapar:

3 = λfx.[f(f(fx))], 4 = λfx.[f(f(f(fx)))], vs.

ile birlikte

1 = λfx. [*fx], 0 = fx.[x]*

Aslında Church'ün 2'si 'iki kez'e, 3'ü 'üç kez'e vs. daha çok benzer. Böylelikle, 3'ün *f* fonksiyonuna uygulanması, yani 3 *f, 'f'in* üç kez tekrarlanması' işlemidir. Bu nedenle 3 *f'in y'e*

uygulanması: (3f) y = *f(f(f(y)))'dir*.

Çok basit bir aritmetik işleminin, örneğin 1'in bir sayıyla toplamının, Church'ün sisteminde nasıl ifade edileceğini görelim:

S = λ*abc.[b((ab)c)]*

S'in, Church'ün ifadesinde tanımlanan bir sayıya 1 eklemesini göstermek için, örneğimizi 3 üzerinde deneyelim:

S3 = λx.a*bc, [b((ab)c] 3* = λ*bc.[(b((3b)c)]*

= λ*bc.[b(b(b(bc)))] = 4*

çünkü (3b) c = *b(b(bc))*. Bu ifadeler kuşkusuz herhangi bir doğal sayıya uygulanabilir (Aslında λ*abc. [(ab)* (bc)], S'in yerini pekala tutabilirdi).

Bir sayının ikiyle çarpımına ne dersiniz? Bu işlem

D = λ*abc. [(ab) (ab)* c)] ifadesiyle verilir. Örnek olarak 3'e uygulanmasıyla,

D = λ*abc.[(ab)((ab)c)]* 3 = λbc.[(3b)((3b)c)]

= λbc.[(3b)(b(6/bc)))] = λ*bc.[b(b(b(b(b(bc)))))]* = 6 bulunur.

Temel aritmetik işlemleri, toplama, çarpma ve kuvvetini alma sırasıyla, şöyle ifade edilebilirler:

A = λ*fgxy.[((fx)(gx))y],*

M = λ*fgx.[f(gx)],*

P = λfg.[fg].

Okuyucu kendini,

(Am)n = m + n, (Mm)n. = m x *n,* (Pm)n = n^m

olduğuna ikna edebilir -veya bana güvenebilir. Burada m ve n, Church'ün iki doğal sayı için fonksiyonları, m + n toplam için fonksiyonu, vb.'dir. Sonuncusu en şaşırtıcı olanıdır, m = 2, n = 3 örnekleriyle bunu deneyelim:

(P2)3 = ((λfg*.[fg])* 2)3 = (λg.[2g])3

= (λg.[λfx.[f(fx)]3 = *gx.[g(gx)]3* = λx.[3(3x)] = 1x.[λfy.[f(f(fy))](3x)]

= λxy.[(3x)((3x)((3x)y))]

= λxy.[(3x)((3x)(x(x(xy))))]

= λxy.[(3x)(x(x(x(x(x(xy))))))]

=λxy.[x(x(x(x(x(x(x(xy))))))))] = 9 = 3^2.

Çıkarma ve bölme işlemleri bu kadar kolay tanımlanamaz (Gerçekten de m, n'den küçük olduğu zaman 'm - n' işlemini; m, n'e bölünemediği zaman 'm - n' işlemine uygulanacak bir yönteme ihtiyacımız var). 1930'ların başlarında Kleene, Church'ün sistemi kapsamında çıkarma işleminin nasıl yapılacağını keşfettiği zaman konuya ilişkin önemli bir aşama katedildi. Bunu diğer işlemler izledi. Sonunda, 1937 yılında, Church ve Turing, her hesaplanabilir işlemin (veya algoritmik işlemin) Church ifadeleri kullanılarak -bugünün Turing makineleri- gerçekleştirilebileceğini (veya bunun tersi: algoritmik işlemler kullanılarak Church'ün ifadelerinin sağlanacağını) gösterdiler.

Church'ün hesaplanabilirlik fikri ilk bakışta hesap makineleriyle bağdaşmıyorsa da, pratikte, özellikle güçlü ve esnek bir bilgi işlem dili olan LISP ile Church'ün tasarımının ana yapıları arasında

sıkı bir bağlantı vardır.

Daha önce belirttiğim gibi, hesaplanabilirlik fikrim tanımlamak için başka yöntemler de vardır. Postun makinesi Turing'in makinesine çok benzer ve hemen hemen aynı dönemde fakat birbirinden ayrı üretilmişlerdir. J. Herbrand ve Gödel, kullanışlı bir hesaplanabilirlik tanımlamışlardır. 1929'da H.B. Curry ve 1924'de M, Schönfinkel konuya biraz farklı yaklaşmışlardır; Church'ün sistemi kısmen bu yaklaşıma dayalıdır (bkz. Gandy 1988). Hesaplanabilirliğe modern yaklaşımlar (Cutland'ın 1980'de tanımladığı sınırsız kayıt makinesi gibi), Turing'in teorisinden ayrıntıda önemli ölçüde farklıdır. Ancak farklı yaklaşımlardan hangisi benimsenirse benimsensin hesaplanabilirlik kavramı aynıdır.

Başta son derece güzel ve temel olanlar gelmek üzere pek çok diğer matematiksel fikir gibi hesaplanabilirlik fikrinin de kendine özgü bir tür Platonik gerçekliği var gözüküyor. Gelecek iki bölümde işte bu gizemli konuyu, matematik kavramların Platonik gerçekliği konusunu işleyeceğiz.

Açıklamalar

[←I] Gerçekte, Turing ilk tanımlamalarında bantın üzerinde karmaşık işaretlerin bulunduğunu varsaymıştır ama bu gerçekten fark etmez. Karmaşık işaretler, işaret/ara şeklinde daima kesik şekle dönüştürülebilir. Bundan öteye Turing'in özgün komutlarında bazı ufak tefek değişiklikler daha yapacağım.

[←II] Bu bölümde yapılan tanımlamalar çerçevesinde 'atış şeması", makineden çok dışsal çevrenin yani 'bantın' bir parçası kabul edilir. Bant üzerinde temsil edilen A, B, A-B, vs. gerçek sayılardır. Ancak, makinenin doğrusal bir-boyutlu formunu da, gelecek bölümlerde tanımlamak isteyeceğiz. Daha sonra göreceğimiz gibi, evrensel Turing makinesi ile ilgili olarak, özel bir 'cihazın' ayrıntılı tanımı ile belirli bir cihazın olası 'verileri'nin (veya 'programı') tanımı arasında çok yakın bir ilişki vardır. Bu nedenle, bunların her ikisini bir-boyutlu formda ele almak, kolay tanımlanmalarını sağlayacaktır.

[←III] Doğal sayılar deyimiyle 0, 1, 2, 3, 4, 5, 6,... sayılarını kastettiğimizi hatırlayın. Fermat teoremindeki (xw + yw = zw) x, y, z > 0, w > 2) ifadesi yerine 'x + 1' ve 'w +3', vs. ifadelerinin kullanılmamasının sebebi, x, w, vs. için sıfırdan başlayarak tüm doğal sayıları dikkate almamızdır.

[←IV] 2, 3, 5, 7,11, 13, 17,... gibi asal sayıların yalnız kendilerine bölünebilir sayılar olduğunu hatırlayınız. Ne 0, ne de 1 asal sayı kabul edilmez.

[←V] Burada reductio ad absurdum, (olmayana ergi) olarak tanınan bir matematik yöntemi kullanılmaktadır: Önce kanıtlanmak istenenin yanlış olduğu düşünülür ve bundan çıkan çelişki nedeniyle sonucun aslında doğru olduğu kanıtlanır.

[←VI] Gerçekte, n'e uygulanan Turing makinesi Tn(n) olarak ifade etmemizi sağladığı için, işlemin en zor kısmı U evrensel Turing makinesinin inşasıyla gerçekleştirilmiştir.

[←VII] Daha alışık olduğumuz bir ifadeyle, örneğin a = b (c) olarak yazabilirdik ama parantezler gerçekten gerekli değildir; (fp) q ve ((fp)q) r yerine, sırasıyla, (f(p)) (q) ve ((f(p)) (q)) (r) gibi son derece zahmetli formüllerden kaçınmak için parantezlerden kurtulmaya alışmakta yarar var.

Notlar

[←1] Sıfırı, ‘doğal sayılar’ arasına alan modern terminolojiyi kullanıyorum.

[←2] Matematikçilerin çok iyi bildiği gibi, konumuz açısından pek uygun olmasa da, ikili, üçlü, vb. sayı gruplarını tek bir sayı gibi kodlamanın çeşitli yollan vardır. Örneğin, 1/2 (a + b)² + 3 a + b) formülü a, b doğal sayı çiftini tek bir doğal sayıyla tam olarak belirler. Deneyin!

[←3] Sayı (veya komut, vs.) dizgelerine başlanacağını göstermek için herhangi bir işaret kullanmaya gerek duymadım. İlk 1 ile karşılaşıldığında işlem başlayacağı için girdi açısından bu gerekli değildir. Ancak, ilk 1’e ulaşıncaya kadar (en solda) önceden bir elemanın çıktı bantı üzerinde ne kadar uzağa gidebileceği tahmin edilemiyebileceği için, çıktı açısından böyle bir işaret gerekebilir. Sola doğru hareketlenen uzun bir O’lar dizisine rastlansa bile bu, daha 1’leride bir 1’e rastlanmayacağının garantisi olamaz. Bu konuda çeşitli görüşler benimsenebilir. Bunlardan birisi, çıktıyı tümüyle başlatmak için (örneğin, büzülüm yönteminde 6 ile kodlanmış) özel bir işaretin kullanılmasıdır. Fakat basitlik endişesiyle, tanımlamalarımda daha farklı bir yaklaşım uygulayacağım: bantın ne kadarının cihaz tarafından incelendiği daima ‘bilinir’ (yani bir tür ‘iz’ bıraktığım düşleyebiliriz); böylece, ilke olarak, çıktının tümünün incelenip incelenmediğini anlamak için sonsuz uzun bantın tümünü incelemek gerekmez.

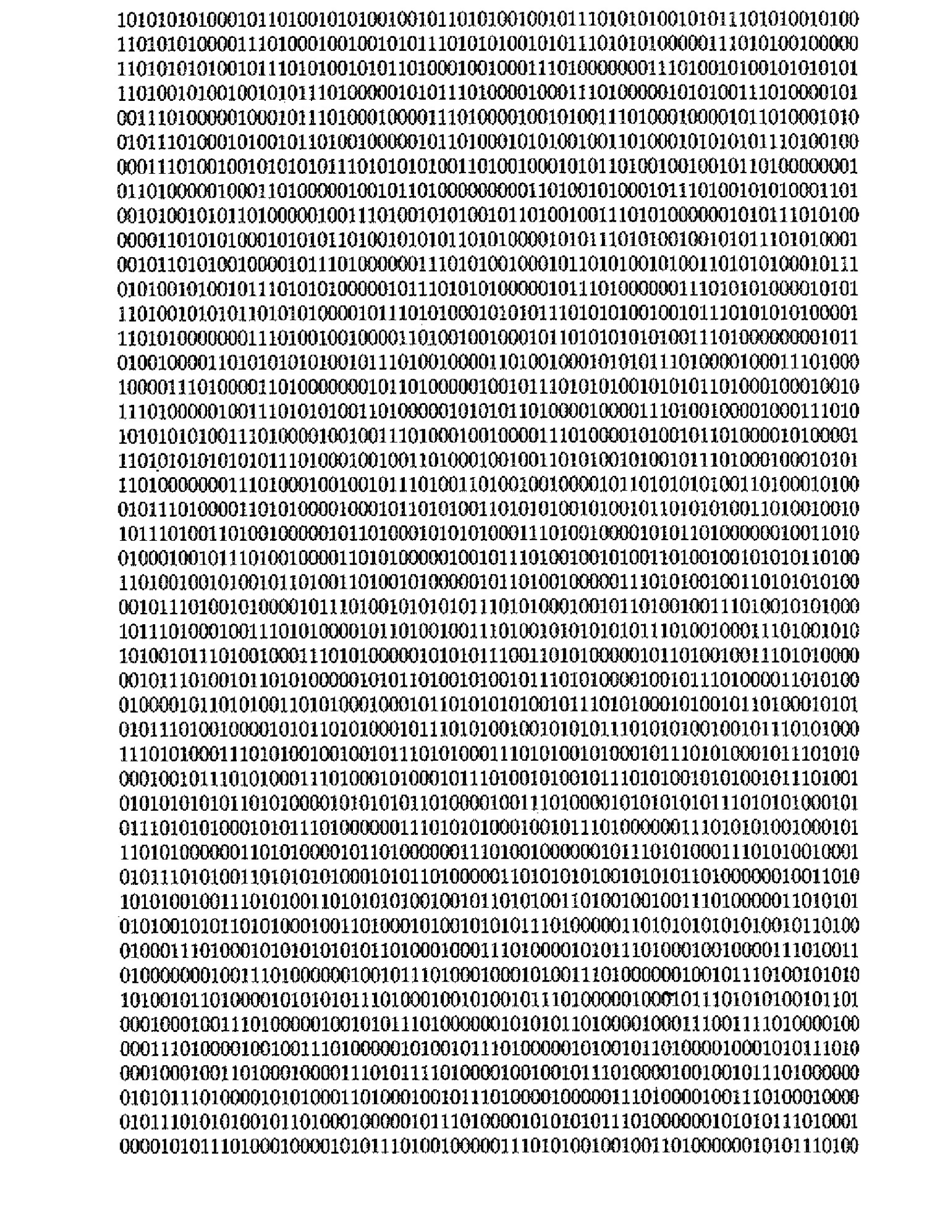
[←4] İki bantın üzerindeki bilgileri kodlamanın bir yolu, ikisi arasında bir ara bant bırakmaktır. Böylece, ara bant üzerindeki tek-sayılı işaretler birinci bant üzerindeki işaretleri gösterirken, çift-sayılar da ikinci bant üzerindeki işaretleri gösterecektir. Benzer bir sistem üç veya daha fazla banta uygulanabilir. Bu yöntemin ‘elverişsizliği’, okuma cihazının bant boyunca durmadan ileri/geri hareket ederek izini belli etmek için bantın çift ve tek taraflarında işaretler bırakmasıdır.

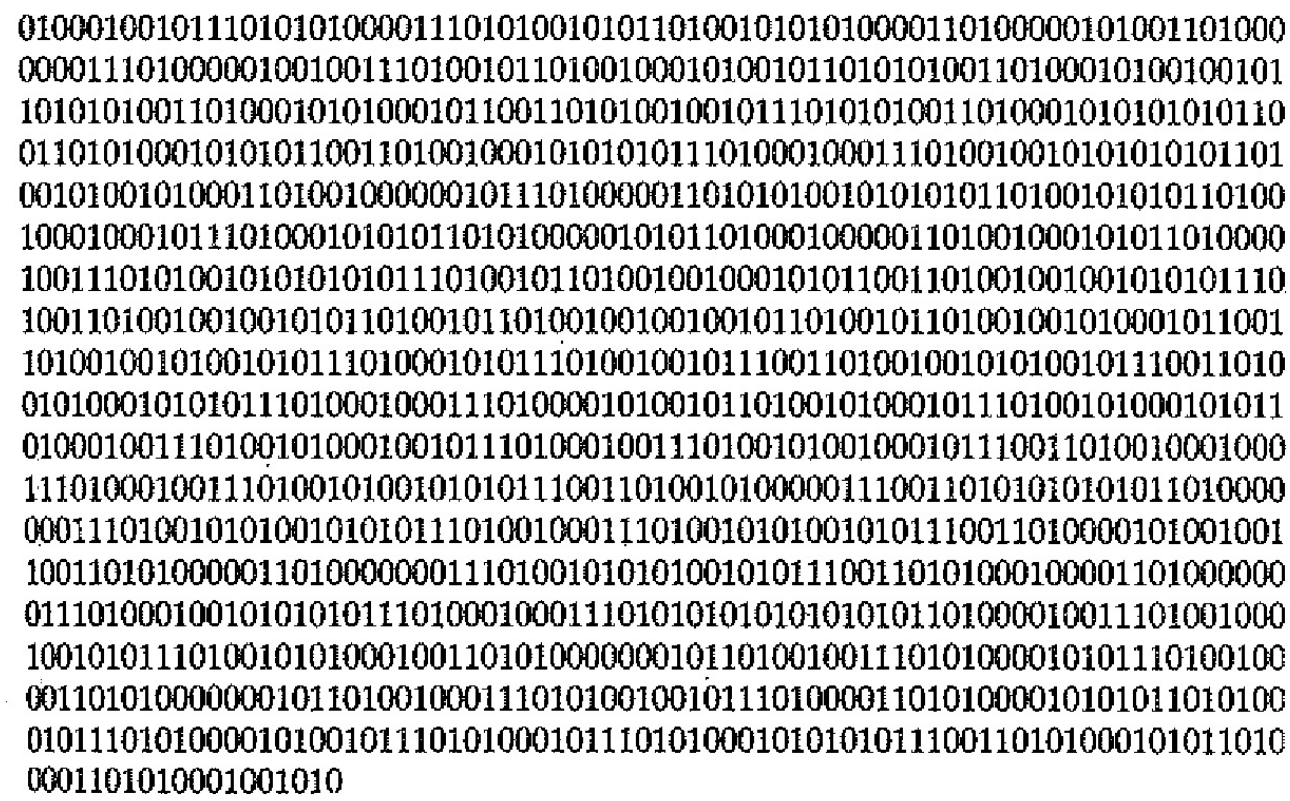
[←5] Bu yöntem yalnız, işaretli bir bantın doğal bir sayı gibi yorumlandığı durama uygulanır. EUC veya XN + 1 gibi özel Turing makinelerimizin sayılarım değiştirmez.

[←6] T_n doğru şekilde tanımlanmadığı takdirde ***U,*** n’in ikilik ifadesinde dörtten fazla 1’den oluşan ilk diziye ulaşılır ulaşılmaz ***n*** sayısı sona ermiş gibi işleme devam edecektir. Dizinin geriye kalan kısmını ***m*** ile ilgili bantın bir kısmı gibi okuyacak, böylece anlamsız işlem yapmayı sürdürecektir! İstenirse bu durum, n’in açılmış ikilik sistemde ifade edilmesini sağlamak suretiyle önlenebilir. Zavallı ***U*** evrensel Turing makinesinin U'nun tanımım daha fazla karmaşıklaştırmamak için bunu yapmamayı tercih ettim!

[←7] u’nun ikilik sisteme dayalı tanımından onluk sisteme dayalı şeklini çıkardığı için David Deutsch’a minnettarım, u’un bu ikilik değerinin gerçekte bir evrensel Turing makinesi ürettiğim doğrulamak için yaptığı çalışma için de kendisine minnettarım. u’un ikilik sisteme göre değeri aşağıdaki gibidir:

10000000010111010011010001001010101101000110100010100000110101001101000101
01001011010000110100010100101011010010011101001010010010111010100011101010
10010010101110101010011010001010000101011010000011010010000010101101000100 1
11010010100001010111010010001110100101010000101110100101001101000010000111
01010000111010100001001001110100010101011010100101011010000011010101001011
01001001000110100000000011010000001110101001010101011101000010011101001010 1
01010101011101000010101011101000010100010111010001010011010010000101001101
00101001001101001000101110101000101110100100101011101001010001110101001010 0
10011101010101000011010010101010111010100100010110101000010110101000100110





Evinde yeterli kapasitede bilgisayarı olan ve yeni şeyleri denemekten hoşlanan girişimci okurlarım, metinde verilen tanımlamaları kullanarak, yukarıdaki kodu çeşitli ve basit Turing makinesi sayılarına uygulayarak, söz konusu kodun gerçekten bir evrensel Turing makinesi'nin uygulanmasını verdiğini kontrol edebilirler!

Bir Turing makinesi için farklı bir tanım uygulayarak u'nun değerini azaltmak mümkün olabilirdi. Örneğin, STOP komutunu kaldırır ve yerine, makinenin içinde bulunduğu bir içsel durumdan sonra 0 içsel durumuna her girişinde durmasını sağlayan bir kural konulabilir. Bu yöntemle pek başarılı sonuç alınamaz (belki hiç sonuç alınamaz). Amacımız için en iyi sonuç, bantları 0 veya 1 dışında işaretlerle işaretlemekle alınabilir. Görünüşte çok kısa kodlamalı evrensel Turing makineleri literatürde tanımlanmış olmakla birlikte bu kısalık yanıltıcıdır, çünkü genel olarak Turing makineleri son derece karmaşık kodlama sistemlerine bağımlıdırlar.

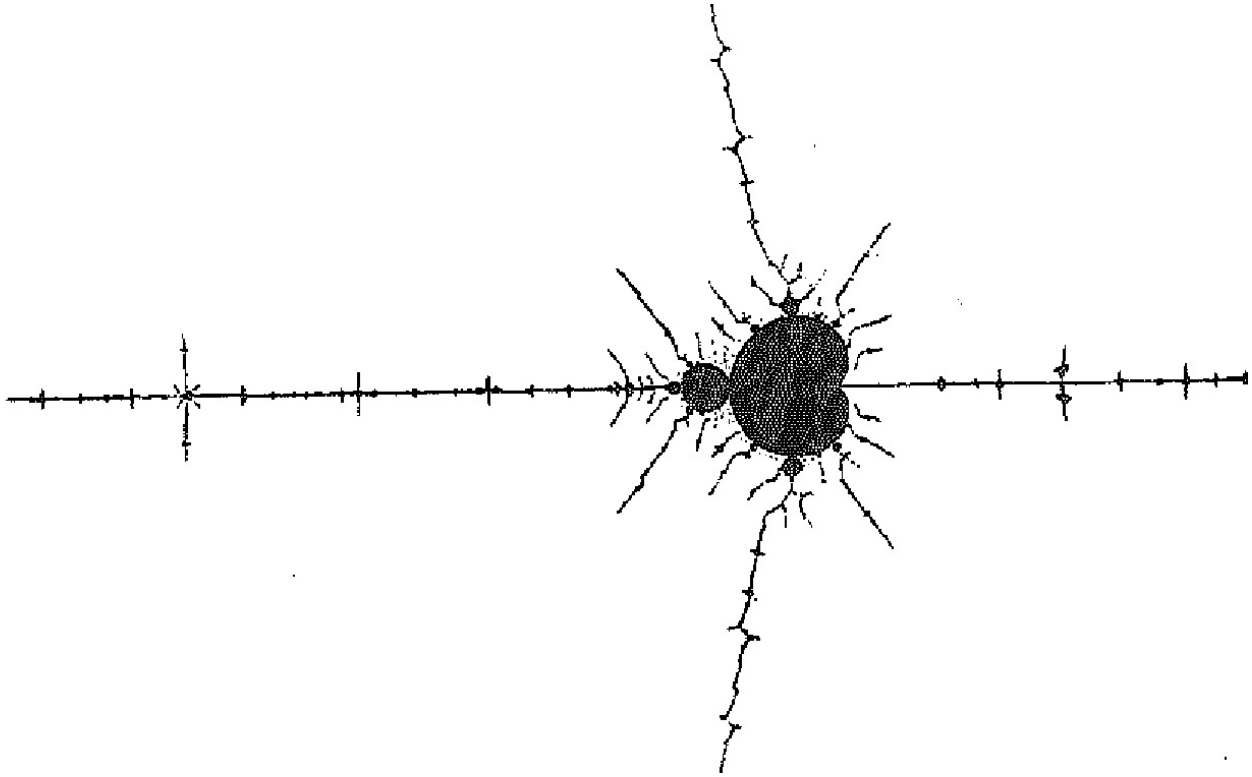
[←8] Bu ünlü sav ile ilgili teknik olmayan konular için, bkz. Devlin (1988).

[←9] Yukarıdaki yöntemi tekrar tekrar uygulamak suretiyle bu geliştirilmiş algoritmayı da kuşkusuz alt edebilirdik. Elde edeceğimiz bilgiyi, algoritmamızı daha da geliştirmek için kullanabilirdik; fakat, yeniden geliştirilen bu algoritmanın da üstesinden gelebilirdik ve bu böylece sürüp giderdi. Bu tekrarlayıcı yöntemin bizi hangi noktaya götüreceği Gödel'in teoremi çerçevesinde ele alınacaktır. (Bölüm IV)

III. Bölüm
Matematik ve Gerçek

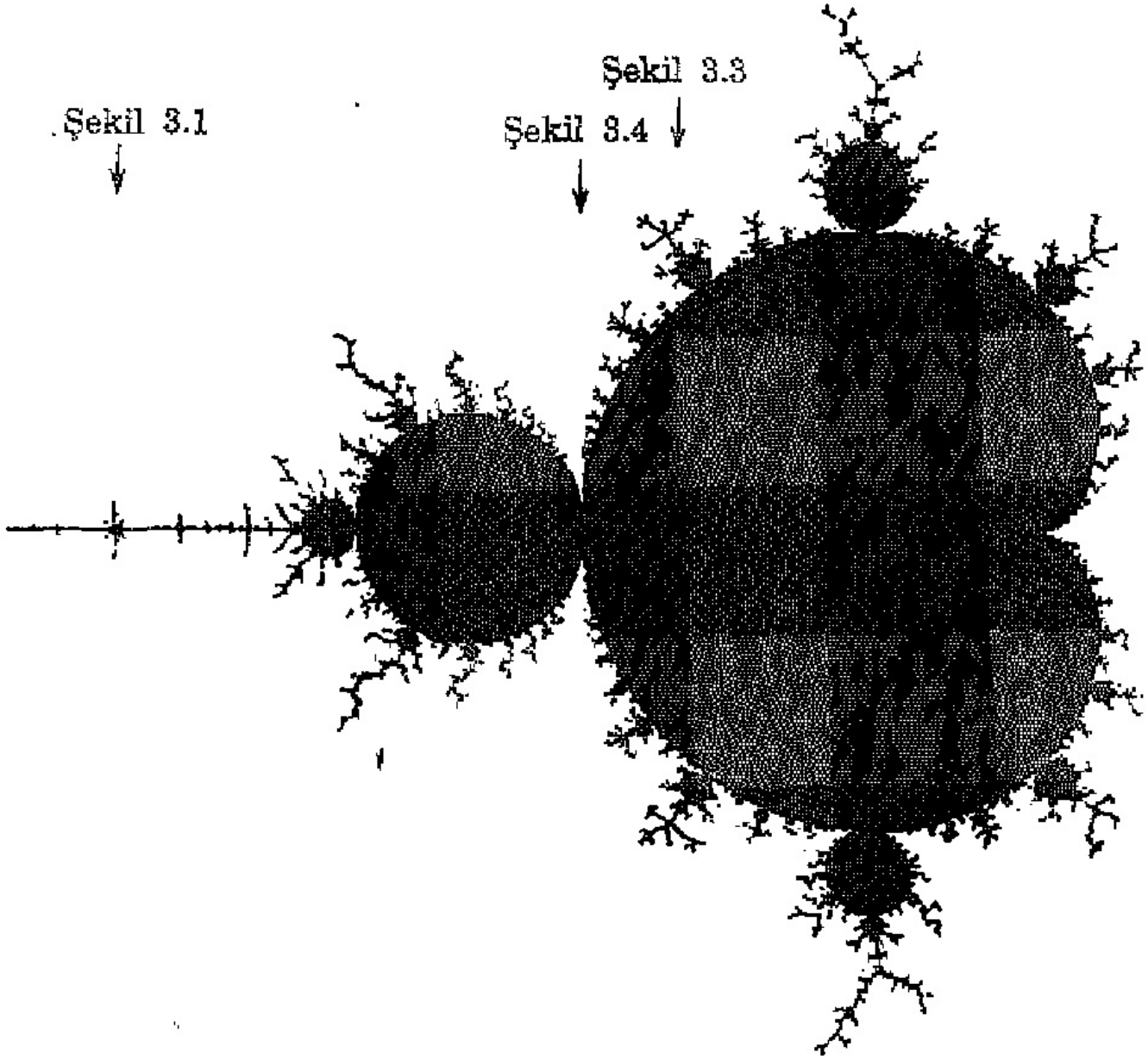
Tor'Bled-Nam Ülkesi

Çok uzaklardaki bir ülkeye yolculuk etmekte olduğumuzu hayal edelim. Bu ülkeye Tor'Bled-Nam adını verelim. Uzaktan algılama cihazımızın aldığı sinyal şimdi önümüzdeki ekranda belirdi (Şekil 3.1):



Şekil 3.1 Garip bir ülkeden ilk görüntüler.

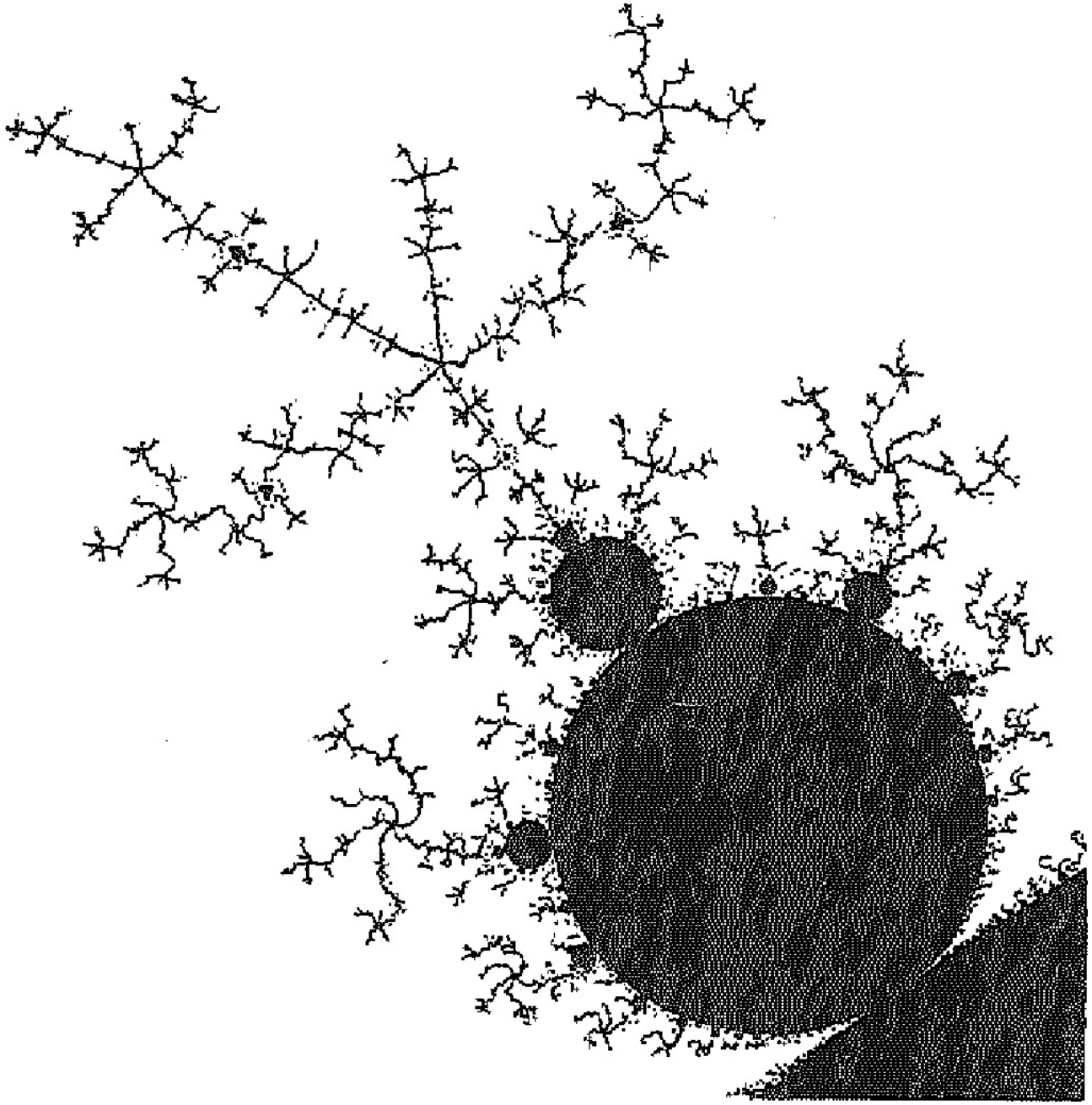
Bu ne olabilir? Garip görünüşlü bir böcek mi? Yoksa, belki de dağlardan inen akarsuların karıştığı, suları koyu bir göl mü? Veya civarındaki şehir ve köylere yollar uzanan yollarıyla büyük ve tuhaf bir şehir mi? Belki de bir adadır. Eğer buysa, o zaman yakınında bir kıta bulabilir miyiz diye bir bakalım. Bunun için algılama cihazımızın büyütmesini on beş kez küçülterek iyice "kuşbakışı" bir görüntü elde etmeliyiz, İşte tüm ülke ekranda göründü (Şekil 3.2).



Şekil 3.2 Bir bütün olarak Tor'Bled-Nam.
Diğer şekillerde büyütülen bölgeler okların hizasında bulunmaktadır.

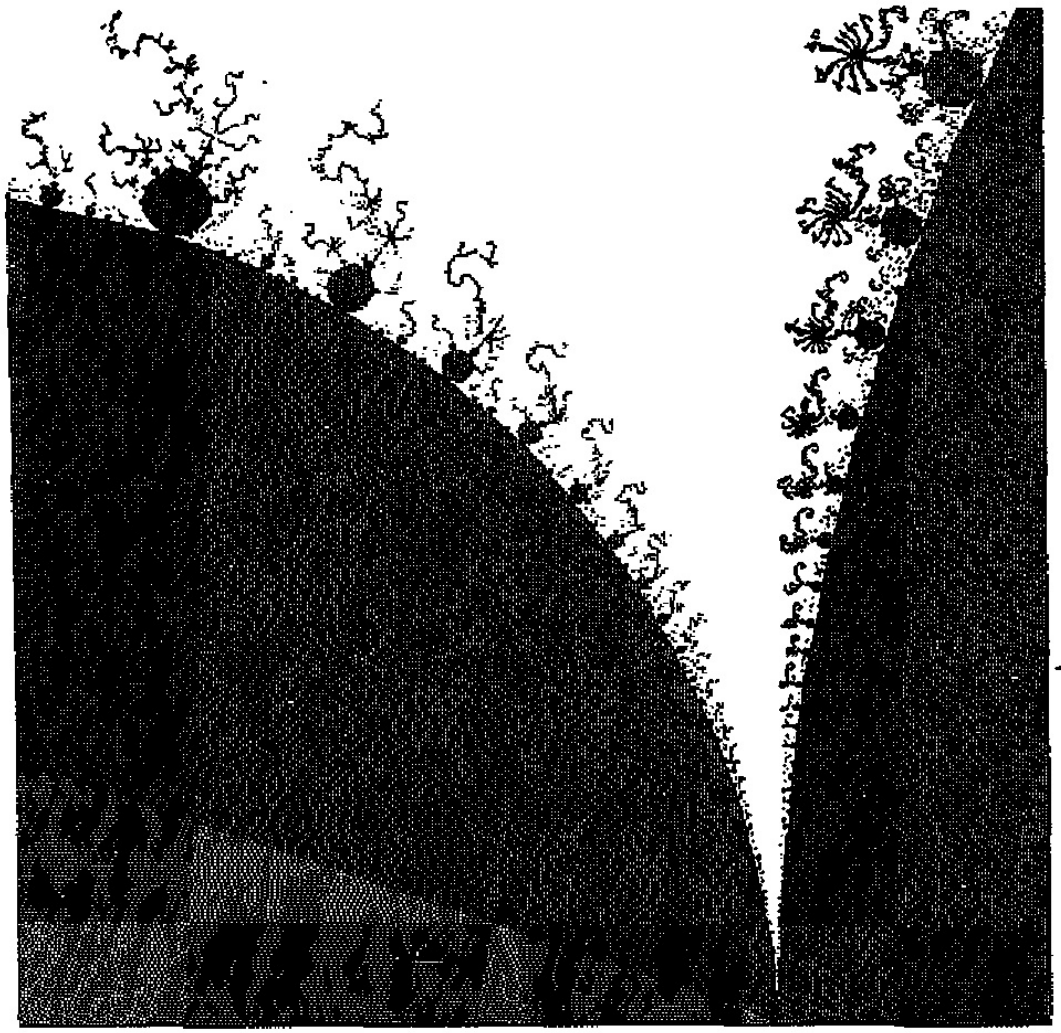
Bir önceki resimdeki 'ada', bu yeni resimde bir nokta gibi kaldı. Adadan çıkan çizgi'lerin hepsi (yollar, nehirler, köprüler?) sadece biri hariç hepsi kesildi. Adanın sağındaki görüntünün tam ortasından çıkan bu tek uzantı, yeni resimde sağdaki büyük şekle bağlanmaktadır. Bu büyük şekil ilk gördüğümüz adaya çok benzemektedir, ancak bazı farklar yok değildir. Görüntüyü bu yeni şeklin kıyı çizgisi üzerine odaklarsak pek çok yuvarlak girinti çıkıntı görürüz. Bu yuvarlak çıkıntılar üzerinde de benzer minik çıkıntılar bulunmaktadır. Her küçük çıkıntı bir noktada daha büyük çıkıntıya bağlanmakta, çıkıntı üstüne çıkıntı yığılmaktadır. Görüntü netleştikçe, bu yapıdan mini mini bir yığın çizginin dışa çıktığını görürüz. Her bir çizgi belli noktalarda çatallanmakta ve çok kez delicesine kıvrılıp bükülmektedir. Belli noktalarda algılama cihazının çözümleyemediği düğüm noktaları fark edilmektedir. Artık anlaşılıyor ki bu cisim ne bir ada, ne bir kıta ve ne de bir gezegen yüzeyidir. Belki de canavar bir böceğe bakmaktayız ve gördüklerimiz daha ana karnından ayrılmamış yavrularıdır.

Yaratığımızın çıkıntılarını anlamak için algılama cihazımızın büyütmesini on kez kadar artırarak yakından inceleyelim. (Şekil 3.2'de Şekil 3.3 olarak belirtilen yer).



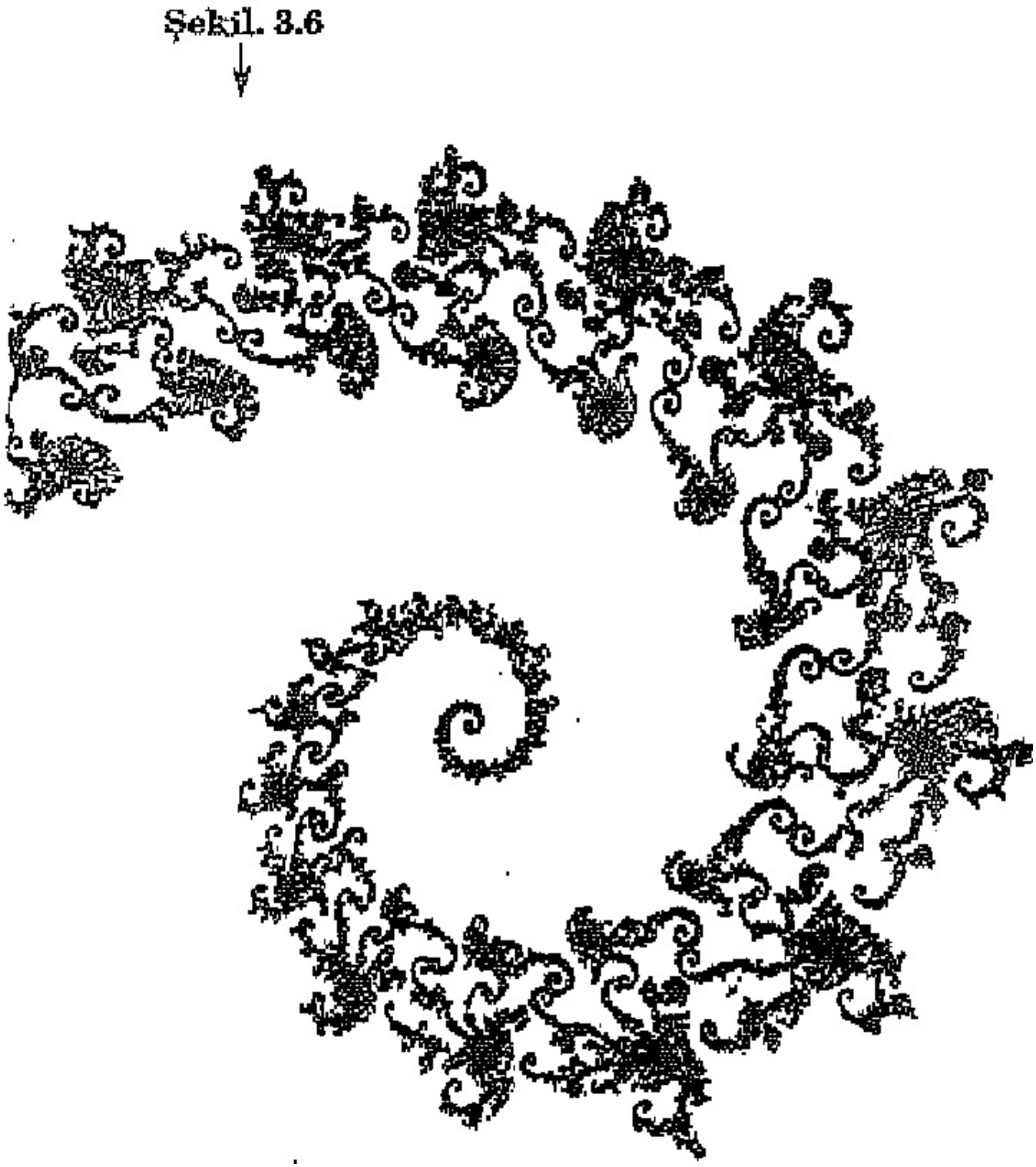
Şekil 3.3 “Beşli” yapıya sahip bir çıkıntı.

Çıkıntı, bütün olarak, sadece bağlantı noktası dışında, yaratığın kendisine büyük bir benzerlik göstermektedir. Dikkat ederseniz, belli noktalarda, beşli teller bir araya gelmektedir. Belki de bu uzantının belli bir “beşli” yapısı var (Tıpkı en üstteki çıkıntının “üçlü” yapısı gibi). Gerçekten de bir sonraki daha küçük boyuttaki çıkıntıya yakından bakınca “yedili" yapı buluruz; daha sonra “dokuzlu” bir yapı belirir ve bu, ikişer ikişer artan, tek tam sayılarla karakterize edilen yapılarla böyle devam eder. Şimdi de Şekil 3:2’ye göre büyütmeyi on kez artırarak sağdaki büyük yarıktan iyice içeri girelim (Şekil 3.4).



Şekil 3.4 'Denizatı vadisi' sağ alt köşede görünüyor.

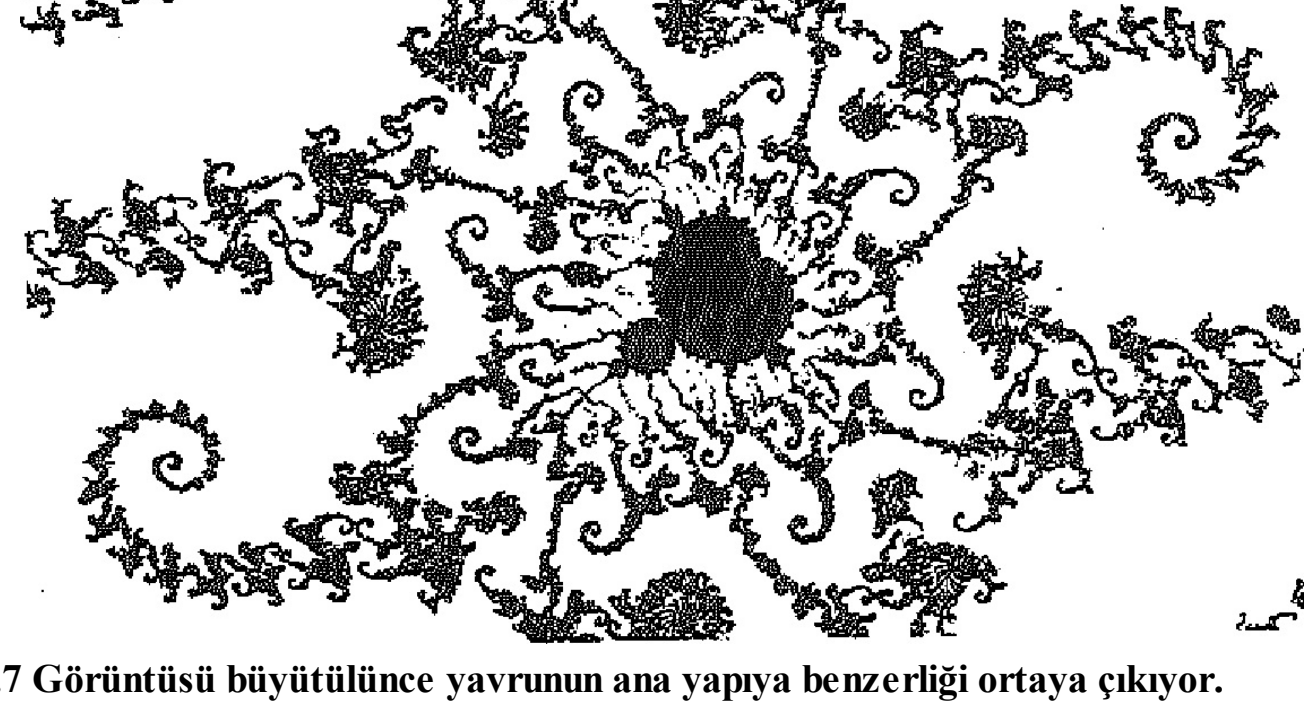
Daha pek çok minik çıkıntıların varlığını ve bunların ipliksi uzantılarını görürüz. Sağ tarafta, artık "denizatı vadisi" diye adlandıracağımız bölgede, minik "denizatı kuyruğu" şeklinde spiraller fark ederiz. Büyütmeyi yeteri kadar artırırsak aynı bölgede çeşit çeşit "deniz gülleri" veya çiçek görünümünde kısımlar buluruz. Belki de gerçekten burası egzotik bir kıyı şeridi, örneğin her noktasından bin bir çeşit canlı fışkıran bir mercan adasıdır. Çiçek gibi görünen şeylerin her birisinin, büyütmeyi daha da artırınca, pek çok minik fakat çeşitli uzantılar ve spiral kuyruklardan oluşan inanılmaz derecede karmaşık yapılar oldukları ortaya çıkar. Denizatı kuyruklarından büyükçe bir tanesine (29'lu bir çıkıntıya bağlı olanına) biraz yakından bakalım! Eğer büyütmeyi yaklaşık 250 kez artırırsak Şekil 3.5'deki spiralle karşılaşırız.



Şekil 3.5 Bir denizatı kuyruğunun yakın görüntüsü.

Demek ki baktığımız basit bir kuyruk değil, ileri geri kıvrılan sayısız minik spirallerden, ahtapot, denizatı görünümündeki bölgelerden ibaret karmaşık bir şekilmiş. Pek çok yerde, tam iki spiralin birbirine değdiği noktada, bir yapı bulunmaktadır. Bunlardan bir tanesine büyütmeyi 80 kez artırarak bakalım. Tam ortadaki o tanıdık şekli görüyor musunuz?

Şekil 3.6 İki spiralin bağlantı noktasının büyütülmüş hali. Tam merkezde minik bir yavrunun varlığı görülmekte.



Şekil 3.7 Görüntüsü büyütülünce yavrunun ana yapıya benzerliği ortaya çıkıyor.

Şekli 6 kez daha büyütünce, (Şekil 3.7) baştan beri incelemekte olduğumuz yapının aynısı bir yavru yaratık daha çıktı karşımıza. Eğer daha yakından bakarsak, bundan çıkan tellerin esas yapıdakilerden biraz farklı olduğunu görürüz; daha bir kıvrımlıdırlar ve daha ötelere uzanmaktadırlar. Yine de küçük yaratık, atasından pek farklı değildir; hatta aşağı yukarı eş konumlarda kendi yavrularına sahip oluncaya kadar. Büyütmeyi artırarak bu yavruları yakından inceleyebiliriz. Torunların da ortak atalarına benzediklerini görür ve artık bunun sonsuza dek böyle sürüp gideceğine inanırız. Algılama cihazımızı giderek artan büyütme güçlerine ayarlayarak bu olağanüstü Tor'Bled-Nam ülkesini istediğimiz kadar yakından araştırabiliriz. Burada sonsuz bir çeşitlilik buluruz: İki bölgenin hiçbiri

bir diğerinin tıpatıp aynısı değildir -ama daha ilk bakışta belirgin bir ortak yapı da vardır. O tanıdık böceksi yaratıklar giderek küçülen boyutlarda tekrar tekrar karşımıza çıkmaktadırlar. Her seferinde civardaki telsi yapılar bir önce gördüğümüzden farklıdır ve bizlere inanılmaz karmaşıklıkta fantastik yeni manzaralar sergilerler. Rastladığımız bu garip, değişken ve şahane ülke nedir? Kuşkusuz pek çok okuyucu bunu zaten bilmekte. Fakat bilmeyenler de olabilir. Bu dünya Mandelbrot kümesi[1] diye bilinen soyut bir matematiksel yapıttan başka bir şey değildir. Şekil, kuşku götürmeyen karmaşıklığına karşın, inanılmaz derecede basit bir kural tarafından yaratılmaktadır. Bu kuralı açıklayabilmek için önce *kompleks sayıların* ne olduğunu açıklamam gerekecek. Bu açıklamayı burada yapmak uygun, çünkü ileride kompleks sayılara tekrar gereksinim duyacağız. Kompleks sayılar, kuantum mekaniğinin yapısı için mutlaka gereklidirler ve bu nedenle içinde yaşadığımız dünyanın işlevlerinin temelini oluştururlar. Ayrıca, matematiğin büyük mucizelerinden birisi de kompleks sayıların varlığıdır. Bir kompleks sayının ne olduğunu açıklamak için önce okuyucuya 'reel sayı'nın ne demek olduğunu hatırlatmalıyım. Bir de, reel (gerçek) sayı kavramı ile 'gerçek dünya'nın gerçekliği arasında nasıl ilişki kurulduğuna değinmek yararlı olacaktır.

Reel Sayılar

Doğal sayıların

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,...

ile gösterilen tam sayılar olduğunu hatırlayalım. Bunlar, sayılar arasında en temel ve basit olanlardır. Aynı türden cisimler, doğal sayılar kullanılarak sayılabilir: Tarlada otlayan yirmi yedi koyun, çakan iki şimşek, on iki gece, bir kelime, dört laf, sıfır yeni fikir, bir hata, altı eksik, iki yön değişimi, vb. şeylerden konuşabiliriz. Doğal sayılar toplanarak veya çarpılarak yeni doğal sayılar bulunabilir. Geçen bölümde verdiğimiz genel algoritma tartışmasının konusu doğal sayılardı.

Ancak, bazı önemli işlemler bizi, doğal sayıların dışına çıkmak zorunda bırakabilir. Bunların arasında en basiti çıkarma işlemidir. Bu işlemi tanımlayabilmek için *eksi* sayılar gereklidir. Bu amaçla tüm *tam* sayıları veririz:

..., -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7,. . .

Elektrik yükü, banka hesabı, tarih[1], vb. bazı şeyler ancak bu tür sayılarla ifade edilebilir. Ancak, tam sayıların kullanımı da hâlâ yeterli değildir, çünkü bir tam sayıyı başka bir tam sayıya *bölemeyiz*. Bunun için *kesirlere,* başka bir deyişle, *rasyonel sayılara* gerek duyarız:

0, 1, -1, 1/2, -1/2, 2, -2, 3/2, -3/2, 1/3,....

Bu sayılar sonlu aritmetik işlemleri için yeterlidir, fakat pek çok problem için bundan da öteye geçip sonsuz veya limit işlemleri tanımlamak zorunda kalırız:

$\pi = 2\ \{(2/1)(2/3)(4/3)(4/5)(6/5)(6/7)(8/7)(8/9)...\}$

ve

$\pi = 4(1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + ...)$.

Bu ünlü ifadelerin birincisi İngiliz matematikçisi, dil bilgini ve şifre uzmanı John Wallis tarafından 1655'de, ikincisi ise, İskoç matematikçisi ve astronomu (ve ilk yansıtmalı teleskopun mucidi) James Gregory tarafından 1671'de bulunmuştur. Aynen π gibi, bu şekilde tanımlanan sayılar, rasyonel olmak

zorunda değillerdir (Yani, *n ve m* $\neq 0$ tamsayılar olmak üzere *n/m* şeklinde yazılamazlar). Böyle sayıları içerebilmesi için, sayı sistemi *genişletilir*. Bu genişletilmiş sayı sistemine reel (gerçek) sayı sistemi denir.

Ondalık gösterimde

-583.70264439121009538...

gibi sonsuz haneli sayılar reel sayılardır. Bu gösterimde π sayısı

π = 3.14159265358979323846...

olarak gösterilir.

Bu şekilde gösterilen sayılar arasında

$\sqrt{2}$ = 1.41421356237309504...

gibi artı kesirli sayıların karekökleri (veya küpkökleri, dördüncü kökleri, vb.), veya π sayısının İsviçreli büyük matematikçi Leonhard Euler tarafından bulunan

$\pi = \sqrt{\{6(1 + 1/4 + 1/9 + 1/25 + 1/36 + \ldots)\}}$.

gösterimi de vardır.

Reel sayılar, aslında, gündelik hayatımızda her an karşımıza çıkan, en çok kullandığımız sayı türüdür. Ancak, çoklukla bu sayıları yaklaşık olarak ele alırız; bir kaç basamağa kadar açılımlarını kullanmakla yetiniriz. Diğer taraftan, matematiksel önermelerde reel sayıları kesin olarak belirlemek gerekir ve bu durumda sonsuz basamaklı ondalık açılımda veya π için Wallis, Gregory ve Euler tarafından verilenlere benzer sonsuz matematik açılımlara gerek duyarız (Buradaki reel sayı tariflerinde, normal olarak, ama sadece alışkanlık nedeniyle, ondalık açılımları kullanacağım. Bir matematikçi için reel sayıları göstermenin bundan daha uygun yolları vardır, fakat bunları burada düşünmemiz gerekmiyor).

Tam bir sonsuz açılım bulunamayacağını sanabilirsiniz, fakat gerçek bu değildir. Basit bir örnek olarak şu sonsuz dizgeyi alalım:

1/3 = 0.333333333333333...,

Burada noktalar 3'lerin sonsuza kadar eklendiğini göstermektedir. Her kesirli sayı tekrar eden bir ondalık açılıma sahiptir.
Örneğin, 93/74 = 1.2567567567567567...

sayısının açılımında 567 dizisi sonsuza dek tekrarlanır, fakat kesirli sayı tam olarak da ele alınabilir. Diğer taraftan

0.220002222000002222220000000222222220...

açılımı *irrasyonel* bir sayıyı tanımlar ve bu sayı kuşkusuz kendi bütünlüğü içerisinde ele alınır (0'lar veya 2'ler dizisi her keresinde belli bir oranda uzamaktadır). Bunlara benzer daha pek çok örnek verilebilir. Her bir durumda tatmin olmamız, açılımı gerçekleştirmek için gerekli kuralın bilinmesine bağlıdır. Birbirini izleyen her basamaktaki rakamı veren bir algoritma varsa, bu algoritma bize tam bir sonsuz ondalık açılımı düşünebilmenin yolunu açar. Açılımları bir algoritmayla verilebilen reel sayılara *hesaplanabilir* sayılar denir. (İki tabanında açılım yerine, diyelim 3 tabanını kullanmanın bir önemi yoktur. Yukarıdaki anlamda 'hesaplanabilir' sayılar hangi taban kullanılırsa kullanılsın aynı sayılardır). Burada incelediğimiz π ve $\sqrt{2}$ reel sayıları, hesaplanabilir sayıların birer örneğidir. Her biri için kural vermek biraz karmaşık olabilir ama ilke olarak zor değildir. Ancak, bu anlamda hesaplanabilir *olmayan* pek çok reel sayı bulunmaktadır. Bir

önceki bölümde gayet iyi tanımlı olmakla birlikte hesaplanamayan dizgelerin varlığını gösterdik. Örnek olarak, şu ondalık açılımı düşünebiliriz: n'inci basamaktaki rakam, *n* sayısı üzerine işlem yapan n'inci Turing makinesi durursa 0, durmazsa 1 alınsın. Genelde, bir reel sayı için sadece bir sonsuz ondalık açılımın varlığını isteriz, n'inci basamağı veren bir algoritmanın bulunmasını, ya da n'inci basamağa konacak rakamı tanımlayan bir kuralın belirlenmesini istemeyiz.[2] Hesaplanabilir sayıların kullanımında gerçekten tuhaflıklar vardır. Sadece hesaplanabilir sayılarla uğraşıyor olsaydık bile işlemlerimizin tümü hesaplanabilir olmayabilirdi. Örneğin, iki hesaplanabilir sayının bir diğerine eşit olup olmadığına karar vermek bile genelde hesaplanabilir bir iş değildir. Bu tür usavurmalar için daha çok *tüm* reel sayıları ele almayı tercih ederiz; dolayısıyla ondalık açılımlar sadece birer hesaplanabilir dizge olmak yerine herhangi bir şey olabilirler.

Son olarak, sonsuz sayıda 9 rakamıyla biten bir ondalık açılımla verilen bir reel sayı ile sonsuz sayıda 0 ile biten ondalıklı açılıma sahip bir reel sayı arasında eşitlik kurulduğuna işaret etmek istiyorum:

-27.1860999999... = -27.1861000000...

Kaç Tane Reel Sayı Var?

Rasyonel sayılardan reel sayılara geçerken yapılan genellemenin ne kadar geniş kapsamlı olduğunu anlamak için burada biraz ara verelim.

İlk bakışta, doğal sayılardan daha fazla tamsayı bulunduğu düşünülebilir. Çünkü her doğal sayı aynı zamanda bir tamsayı olduğu halde, bazı tamsayılar (yani, eksi sayılar) doğal sayı değildir. Benzer olarak, kesirli sayıların tamsayılardan daha çok olduğu düşünülebilir. Ancak, durum hiç de böyle değildir, 1800'lerin sonlarında, çok özgün düşüncelere sahip, Alman asıllı Rus matematikçisi Georg Cantor'un öne sürmüş olduğu güçlü ve güzel sonsuz sayılar teoremine göre, kesirli sayıların da, tam sayıların da, doğal sayıların da toplam sayısı $\aleph_0$ (alef sıfır) ile gösterilen aynı sonsuz sayıdadır (Buna benzer bir fikrin, 250 yıl kadar önce, 1600'lerin başında büyük İtalyan fizikçisi ve astronomu Galileo Galilei tarafından ifade edilmiş olması ilginçtir. Galileo'nun diğer bazı buluşlarına V. Bölüm'de tekrar değineceğiz).

Tam sayıların adeti ile doğal sayıların adetinin aynı olduğunu görmek için aralarında aşağıdaki gibi 'bire-bir eşleme' yapılır:

Tam sayılar		Doğal sayılar
0	↔	0
−1	↔	1
1	↔	2
−2	↔	3
2	↔	4
−3	↔	5
3	↔	6
−4	↔	7
.	.	.
.	.	.
.	.	.
$-n$	↔	$2n-1$
n	↔	$2n$
.	.	.
.	.	.
.	.	.

Bu listede, sol koldaki her bir tam sayı ile sağ koldaki her bir doğal sayı sadece ve sadece bir kez geçerler. Cantor'un teoreminde böyle bir bire-bir eşleşmenin varlığı, sol koldaki nesnelerin adetinin sağ koldaki nesnelerin adeti ile *aynı* olduğunu gösterir, Böylece, tamsayıların adeti, gerçekten, doğal sayıların adetine eşit olmaktadır. Burada bu sayı sonsuz çıkıyorsa da, bu önemli değildir (Sonsuz sayıların kullanımı nedeniyle karşılaştığımız tek tuhaflık, yukarıdaki listenin bir yanından bazı elemanları çıkarsak bile, iki sütun arasında *hâlâ* bire-bir eşleme yapabilmemizdir). Benzer olarak, ama biraz daha karmaşık bir biçimde, kesirli sayılar ile tamsayılar arasında da bire-bir eşleme kurabiliriz (Bunun için, verilen bir kesirli sayının payı ve paydasından oluşan çifti, tek bir doğal sayıyla göstermenin yollarından herhangi birini seçebiliriz; bkz. II. Bölüm) Doğal sayılarla bire-bir eşlenebilen kümelere 'sayılabilir' denir. Sayılabilir kümelerin eleman sayısı $\aleph_0$'dır. Tamsayılar kümesi gibi, kesirli sayılar kümesi de sayılabilir.

Eleman adeti sayılamayan kümeler var mıdır? Doğal sayılardan tam sayılara, oradan da kesirli sayılara geçerken, sayı sistemimizi genişletmiş olduk ama gerçekte eleman sayısı artmış olmadı. Tüm bu durumlarda elemanların sayılabildiğini gördük. Belki de okuyucu *tüm* sonsuz kümelerin sayılabildiği izlenimini edinmiştir bile. Ama daha değil, çünkü reel sayılara geçildiğinde durum çok farklıdır. Cantor'un en kayda değer başarılarından birisi, kesirli sayılardan *daha çok* reel sayının varlığını kanıtlamış olmasıdır. Cantor'un yöntemi II. Bölüm'de bahsi geçen ve Turing'in de, Turing makinelerinin durma probleminin çözümsüzlüğünü kanıtlamak için kullanmış olduğu köşegen çizik yöntemidir. Turing'in kanıtı gibi, Cantor'un kanıtı da *olmayana ergi (reductio ad absurdum)* üzerine

kuruludur. Diyelim ki kanıtlamak istediğimiz sonuç yanlıştır; yani, reel sayılar kümesi sayılabilir. Öyleyse 0 ve 1 arasındaki reel sayılar da sayılabilir ve bu sayılarla tamsayılar arasında bire-bir eşleme kuran bir liste oluşturulur:

Doğal sayılar		Reel Sayılar
0	↔	0.10357627183...
1	↔	0.14329806115...
2	↔	0.02166095213...
3	↔	0.43005357779...
4	↔	0.92550489101...
5	↔	0.59210343297...
6	↔	0.63667910457...
7	↔	0.87050074193...
8	↔	0.04311737804...
9	↔	0.78635081150...
10	↔	0.40916738891...
.	.	.
.	.	.
		.

Köşegenin üzerindeki rakamları koyu yazdım. Bu rakamlar, yukardaki liste için sırasıyla 1, 4, 1, 0, 0, 3, 1, *4,* 8, 5, 1, ... bulunmuştur. Köşegen yöntemi, ondalık açılımının her hanesi yukarıdaki dizide yer alan rakamlardan farklı olan 0 ile 1 arasında bir reel sayıyı yazmaktan ibarettir. Bunu daha açık belirtmek için, diyelim ki köşegen üzerindeki rakam 1 değilse o haneye 1 yazalım, 1 ise o haneye 2 yazalım. Böylece yukarıdaki listeden

0.21211121112...

reel sayısı bulunur. Bu reel sayı listede yoktur çünkü virgülden sonra 1. hanedeki rakam, listenin 1. sayısının 1. hanesinden farklıdır; 2. hanesindeki rakam 2. sayısının 2, hanesinden, 3. sayının 3. hanesinden, vb. farklıdır. Bu bir çelişkidir, çünkü yukardaki listenin, 0 ile 1 arasındaki bütün reel sayıları içerdiğini kabullenmiştik. Bu çelişki, göstermek istediğimiz sonucun kanıtıdır. Yani, doğal sayılarla reel sayılar bire-bir eşlenemezler; dolayısıyla, reel sayıların sayısı kesirli sayılardan daha çoktur ve sayılamazlar.

Reel sayıların adedi C ile gösterilen *sonsuz sayıdır.* Bu sayıya neden $\aleph_1$ denmediği sorulabilir. Gerçekten $\aleph_1$ sayısı $\aleph_0$'dan büyük bir sonraki sonsuz sayıyı göstermektedir ve $C = \aleph_1$ eşitliğinin doğru olup olmadığının kanıtlanması henüz çözülmemiş bir problemdir ve *süreklilik varsayımı* adı verilmektedir.

Öte yandan *hesaplanabilen* sayıların *sayılabilir* olduklarını hatırlayalım. Bunları saymak için, reel sayıların her birini üreten Turing makinelerini sıraya koyup saymamız yeterlidir. Listede zaten bulunan bir reel sayıyı üreten Turing makinesini listeden çıkarmak isteyebiliriz. Turing makineleri

sayılabilir olduğuna göre, hesaplanabilir reel sayılar da sayılabilmelidir. Neden bu listeye köşegen yöntemini uygulayıp listede bulunmayan yeni bir hesaplanabilir sayı bulamayız? Bunun yanıtı genelde, bir Turing makinesinin listede bulunup bulunmadığına hesaplanabilir bir şekilde karar verilememesidir. Karar verilebilmesi, durdurma problemim çözebilmemize bağlıdır. Bazı Turing makineleri bir reel sayının ondalık basamaklarını tam verirken takılır ve artık başka bir ondalık basamak vermezler (çünkü, yanıt vermek için *duramazlar)*. Hangi Turing makinesinin böyle takılacağını önceden hesaplayarak söylemenin yolu yoktur. Durdurma problemi adı verilen problem temelde budur. Dolayısıyla, köşegen yöntemimiz bir reel sayı vermekle birlikte bu sayı hesaplanabilir bir sayı değildir. Aslında bu tartışmayı, hesaplanamaz sayıların varlığını, *kanıtlamak* için kullanabilirdik. Turing'in önceki bölümlerde değinilen algoritmalarla çözülemiyen problem sınıflarının varlığını gösteren tartışması, tam olarak, bu uslamlamayı izler. Köşegen yönteminin başka uygulamalarını ileride göreceğiz.

Reel Sayıların 'Gerçekliği'

Hesaplanabilirlik kavramını bir yana bırakırsak, reel sayılara 'gerçek' denmesinin nedeni, uzaklık, açı, zaman, enerji, sıcaklık, vb. pek çok geometrik ve fiziksel ölçüm sonucunun reel sayılarla verilmesinden kaynaklanır. Ancak, soyut tanımlanmış reel sayılarla fiziksel nicelikler arasındaki ilişki hiç de sanıldığı kadar açık değildir. Reel sayılar, herhangi bir gerçek fiziksel somut nicelik yerine bir *matematiksel idealleştirmeye* dayanırlar. Örneğin, reel sayılar sisteminin bir özelliği, ne kadar yakın olursa olsunlar herhangi iki sayının arasında üçüncü bir sayının mutlaka bulunmasıdır. Fiziksel uzaklıklar ya da zamanlar için bu özelliğin doğruluğu pek de açık değildir, iki nokta arasındaki fiziksel uzaklığı daha küçük parçalara ayırmaya başlarsak, bir süre sonra öyle küçük bir mesafe ölçeğine ulaşırız ki, bilinen anlamda uzaklık kavramının kendisi anlamını yitirir. Bir atom-altı parçacığın boyutunun 10^{20}'de birine[II] eşit 'kuantumlu gravitasyon' ölçeğinde beklenen budur. Ama reel sayılarla, bu mesafe ölçeğinden çok daha küçüklerine, gerçek boyutunun 10^{200}'de birine, 10^{2000}'de birine veya $10^{10^{200}}$'de birine ve daha küçük boyutlara kadar gidebilmeliydik. Benzer nitelik, çok küçük zaman aralıkları için de geçerlidir.

Fizikte reel sayı sisteminin kullanımı, fiziksel uzaklık ve zaman kavramlarıyla büyük bir aralıkta uyum sağlanmasının yanı sıra matematiksel kullanışlılığı, basitliği ve güzelliği nedeniyledir. Bu seçim, bütün bu fiziksel kavramlarla her aralıkta uyum sağlayalım diye yapılmamıştır. Çok küçük mesafe ve zaman ölçeklerinde böyle bir uyum beklenmez bile. Mesafe ölçmek için bildiğimiz cetveller kullanılır ama bu cetveller atomlarının. boyutu düzeyinde, taneli bir yapıda görüneceklerdir. Salt bu neden, reel sayıları ölçüm sonuçlarını vermek için kullanmamıza engel olamaz. Fakat böyle küçük mesafeleri ölçerken çok daha büyük dikkat gerekir. En azından çok küçük mesafe ölçeklerinde temel ilkelerde bir zorlukla karşılaşmayacağımızdan kuşkuya düşebilirdik. Neyse ki Doğa bize gerçekten iyi davranmış; günlük veya daha büyük ölçeklerde kullanmaya alıştığımız reel sayılar, atomlardan bile çok daha küçük ölçeklerde kullanılışlılıklarını koruyorlar. Reel sayıların kullanımı, bir elektron veya proton boyutunun yüzde birine kadar kesinlikle, 'kuantumlu gravitasyon ölçeğinde', yani bu atom-altı parçacıkların boyutundan yirmi mertebe küçük ölçeklerde ise, büyük olasılıkla geçerlidir. Bu, deneyime dayalı olağanüstü bir genellemedir. Öte yanda, reel sayılarla ölçülen mesafe kavramı en uzak kuasarlara ve daha ötelere kadar geçerli görünmektedir. Böylece en az 10^{42} veya

10^{63} veya daha büyük bir aralık taranmış olmaktadır. Reel sayı sisteminin uygunluğu çoğu kez sorgulanmaz. Bu sayıların, uygunluğu üzerine ilk deneyimlerimiz aslında böyle çok sınırlı bir aralıkta kalmasına karşın, reel sayıların, fiziksel olguların doğru bir tanımını verdiğine neden bu kadar güven duyulur? Bu güven, belki haklı olmamakla birlikte (ve çok kez açıkça belirtilmemesine karşın), Doğanın olağanüstü matematiksel bütünlüğüne olan inancımızla birlikte, reel sayı sisteminin mantıksal güzelliğinden, tutarlılığından ve matematiksel gücünden kaynaklanıyor olmalı.

Kompleks Sayılar

Reel sayı sisteminin matematiksel etkinliği ve şıklığının yanı sıra, örneğin, karekökü alınabilen bir sayının mutlaka artı bir sayı veya sıfır olması gerektiği, eksi sayıların karekökünün alınamaması gibi bazı olumsuz özellikleri vardır. Fiziksel dünyayla doğrudan ilişki kurabilmek sorusunu bir an için bir yana bıraksak bile, matematik yönünden artı sayıların olduğu gibi eksi sayıların da karekökünün alınabilmesi çok yarar sağlar. -1 sayısının karekökünün varolduğunu kabul edelim veya böyle bir sayıyı icat edelim. Bu sayıyı 'i' simgesiyle gösterdiğimiz zaman $i^2 = -1$ eşitliği elde edilir. Doğal olarak, i niceliği reel bir sayı olamaz, çünkü herhangi bir reel sayının kendisiyle çarpımı (yani, karesi) mutlaka sıfırdan büyük (veya sayı sıfırsa, sıfıra eşit) bir reel sayıdır. Bu nedenle, kareleri eksi olan sayılara *sanal* sayılar da denir. Oysa, bu sayıların da, kullanmaya alışık olduğumuz reel sayılar kadar gerçek olduklarını önemle vurgulamak gerekir. Daha önce değindiğim gibi, reel sayılarla fiziksel gerçeklik arasındaki ilişki, doğadan açık bir *a priori* doğrulanması gelmeyen sonsuz küçük bir matematiksel idealleştirme içerir ve ilk bakışta göründüğü gibi ne açık ne de zorlayıcıdır.

-1 sayısının karekökünü bulduktan sonra, tüm reel sayıların karekökünü bulmak büyük bir çaba gerektirmez, *a* bir artı sayı ise, i x √a sayısı '-a' sayısının bir kareköküdür (Öteki karekök -i x √a sayısıdır). Peki i sayısının kendisinin de karekökü var mıdır? Kuşkusuz vardır. Hemen sağlamasını yapalım:

$(1 + i)/\sqrt{2}$

sayısının (ve bunun eksi işaretlisinin) karesi i e eşittir. Bu sayının karekökü var mıdır? Yanıt yine evet olacaktır:

$$\sqrt{\frac{(1 + 1/\sqrt{2})}{2}} + i\sqrt{\frac{(1 - 1/\sqrt{2})}{2}}$$

sayısının ve bunun eksi işaretlisinin kareleri gerçekten $(1+i)/\sqrt{2}$ verir.

Bu tür nicelikleri oluştururken reel ve sanal sayıları toplamaya ve elde ettiğimiz sayının herhangi bir reel sayıyla çarpımına -veya sıfırdan başka sayılarla bölümüne- (çünkü bu sayının bir bölüşüyle çarpım demektir) izin verdiğimize dikkat edelim. Sonuçta bulunan sayılara *kompleks sayılar* denir. Bir kompleks sayı, 'a' ve 'b' reel sayılar olmak üzere

$a + ib$

şeklindedir, a'ya kompleks sayının *reel kısmı,* b'ye ise *sanal kısmı* denir. İki kompleks sayıyı

toplamak veya çıkarmak için

$i^2 = -1$ kabulü altında, okulda öğrendiğimiz cebir kuralları izlenir:

$(a + ib) + (c + id) = (a + c) + i\,(b + d)$

$(a + ib) \times (c + id) = (ac - bd) + i\,(ad + bc)$

İşte şimdi dikkate değer bir şeyler oluyor! Bu yeni sayı sistemini bulmaktan amacımız her zaman karekök alabilmektir. Gerçi henüz açık olarak görülmüyor ama bunu yapabiliyoruz. Çok daha fazlasını da yapıyoruz. 18. yüzyılda yaşamış büyük matematikçi Leonhard Euler'in gösterdiği gibi, küp kökleri, beşinci kökleri, doksan dokuzuncu kökleri, π'inci kökleri, $(1+i)$'nci kökleri zahmetsizce alabiliyoruz. Kompleks sayıların sihirli yönünü gösteren bir başka örnek olarak, okulda öğrenilen biraz karmaşık görünümlü bir trigonometri formülüne bakalım: İki açının toplamının sinüsünü ve kosinüsünü veren

$\sin(A + B) = \sin A \cos B + \cos A \sin B \quad \cos(A + B) = \cos A \cos B - \sin A \sin B$

bağıntıları, çok daha basit (ve dolayısıyla hatırlaması daha kolay) olan[III]

$e^{iA+iB} = e^{iA} e^{iB}$

kompleks denkleminin, sırasıyla, reel ve sanal kısımlarıdır. Burada bilmemiz gereken sadece 'Euler formülü'dür (Görüşüne göre bu formül, Euler'den yıllar önce, 16. yüzyılda, İngiliz matematikçi Roger Cotes tarafından da bulunmuştur):

$e^{iA} = \cos A + i \sin A.$

Bunu bir önceki denklemde yerine koyar

$\cos(A + B) + i \sin(A + B) = (\cos A + i \sin A)(\cos B + i \sin B)$

ifadesini bulur ve eşitliğin sağ yanındaki parantezleri açarsak istenen trigonometri formüllerine ulaşırız.

Bundan da öte,

$a_0 + a_1 z + a_2 z^2 + a_3 z^3 + \ldots + a_n z^n = 0$

cebirsel denklemi bir kompleks z sayısı için her zaman çözülebilir ($a_0, a_1, a_2, \ldots, a_n$ diye verilmiş kompleks sayılarda $a_n \neq 0$ olmalıdır).

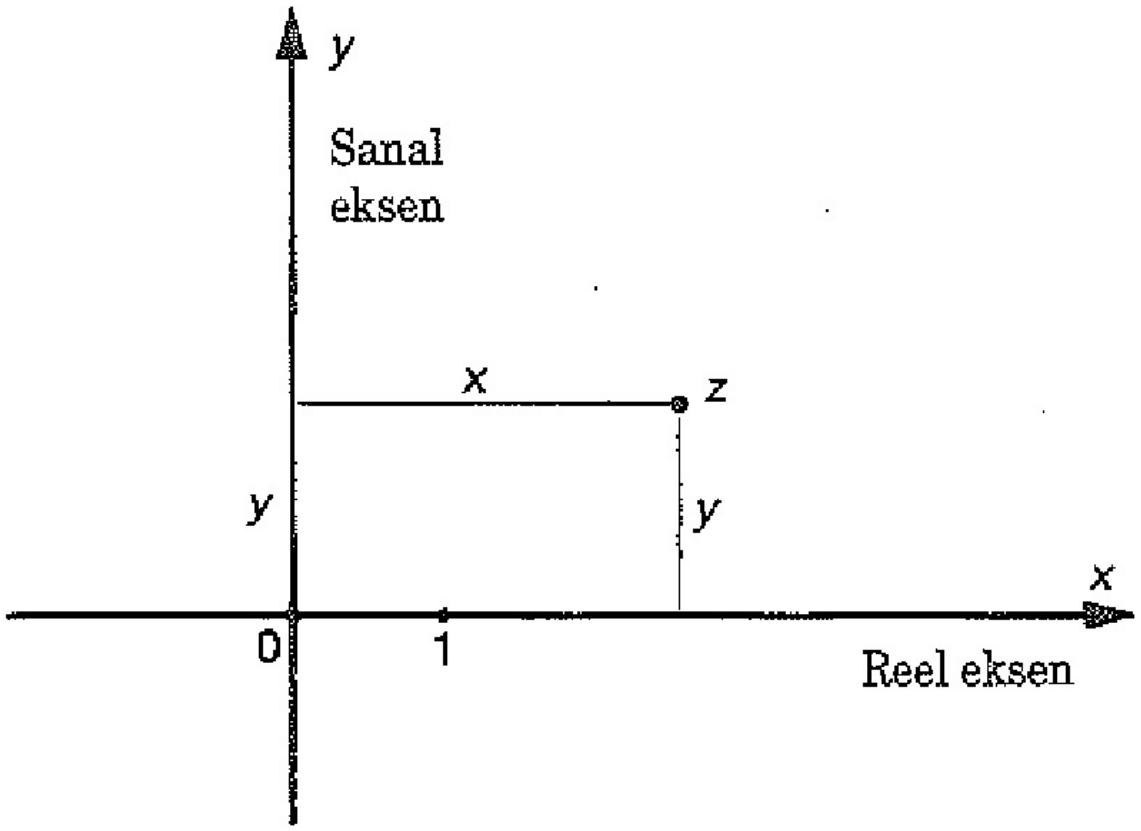
Örneğin,

$z^{102} + 999 z^{33} - \pi z^2 = 417 + i$

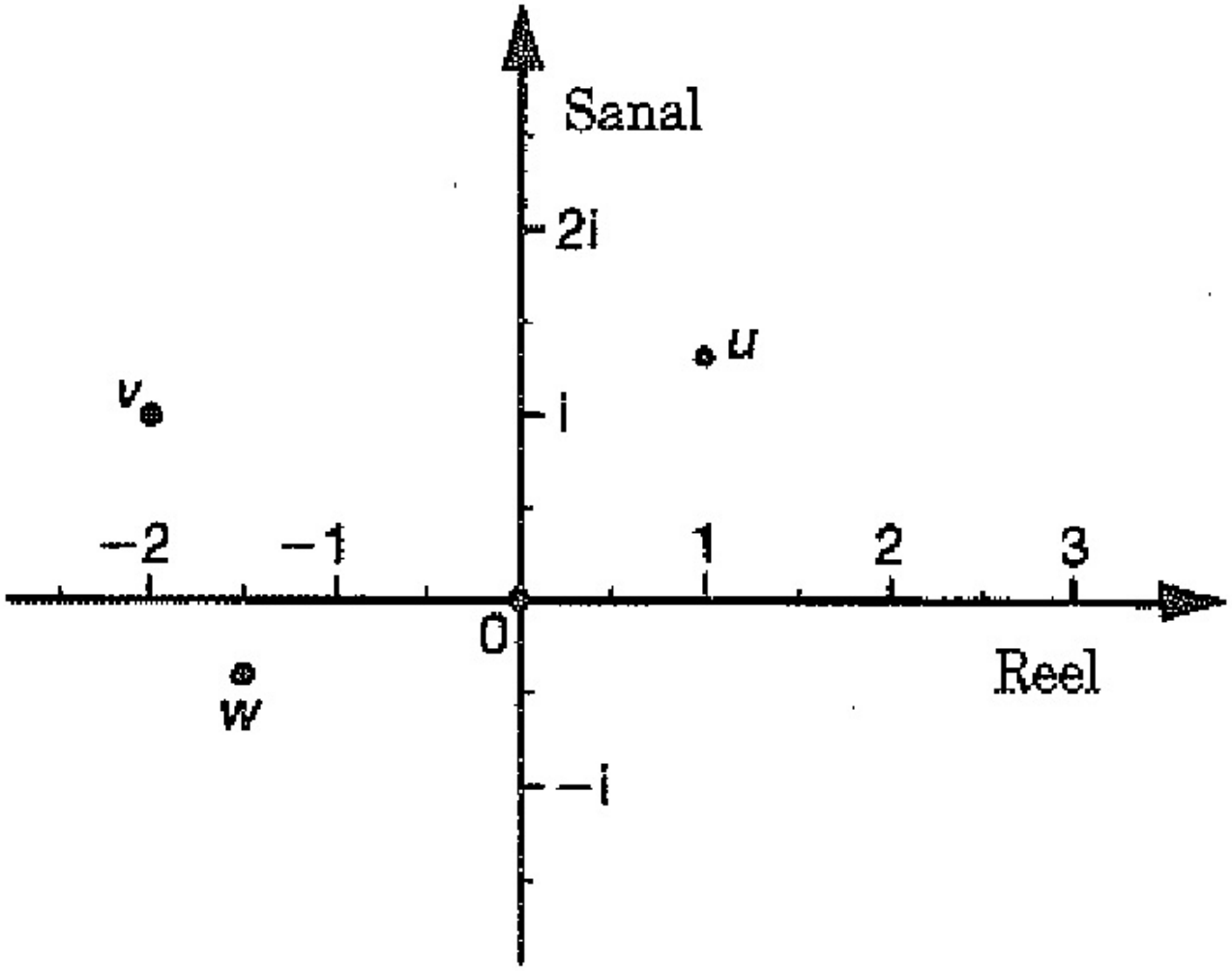
bağıntısını sağlayan bir z kompleks sayısı, varlığı hiç de açık olmamakla birlikte, bulunur. Bu olguya bazen *cebirin temel teoremi* adı verilir. Pek çok 18. yüzyıl matematikçisi bu sonucu ispatlamak için uğraştılar. Euler bile tatmin edici bir genel ispat veremedi. Sonunda 1831'de büyük matematikçi ve fen adamı Carl Friedrich Gauss, çarpıcı özgünlükte bir uslamlamayla ilk genel ispatı verdi. İspatın can alıcı noktası, kompleks sayıları *geometrik* olarak betimlemesi ve buradan hareketle ispatlamada topolojik[IV] kavramları kullanmasıydı.

Aslında, kompleks sayıların geometrik tanımını ilk kez Gauss kullanmamıştır. Yaklaşık 200 yıl

daha önce Wallis kullanmış, fakat Gauss kadar güçlü sonuçlar elde edememiştir. Kompleks sayıların geometrik tarifi, isviçreli bir kâtip olan Jean Robert Argand'ın adıyla anılır.



Şeki 3.8: z = x + iy kompleks sayısını gösteren Argand düzlemi.

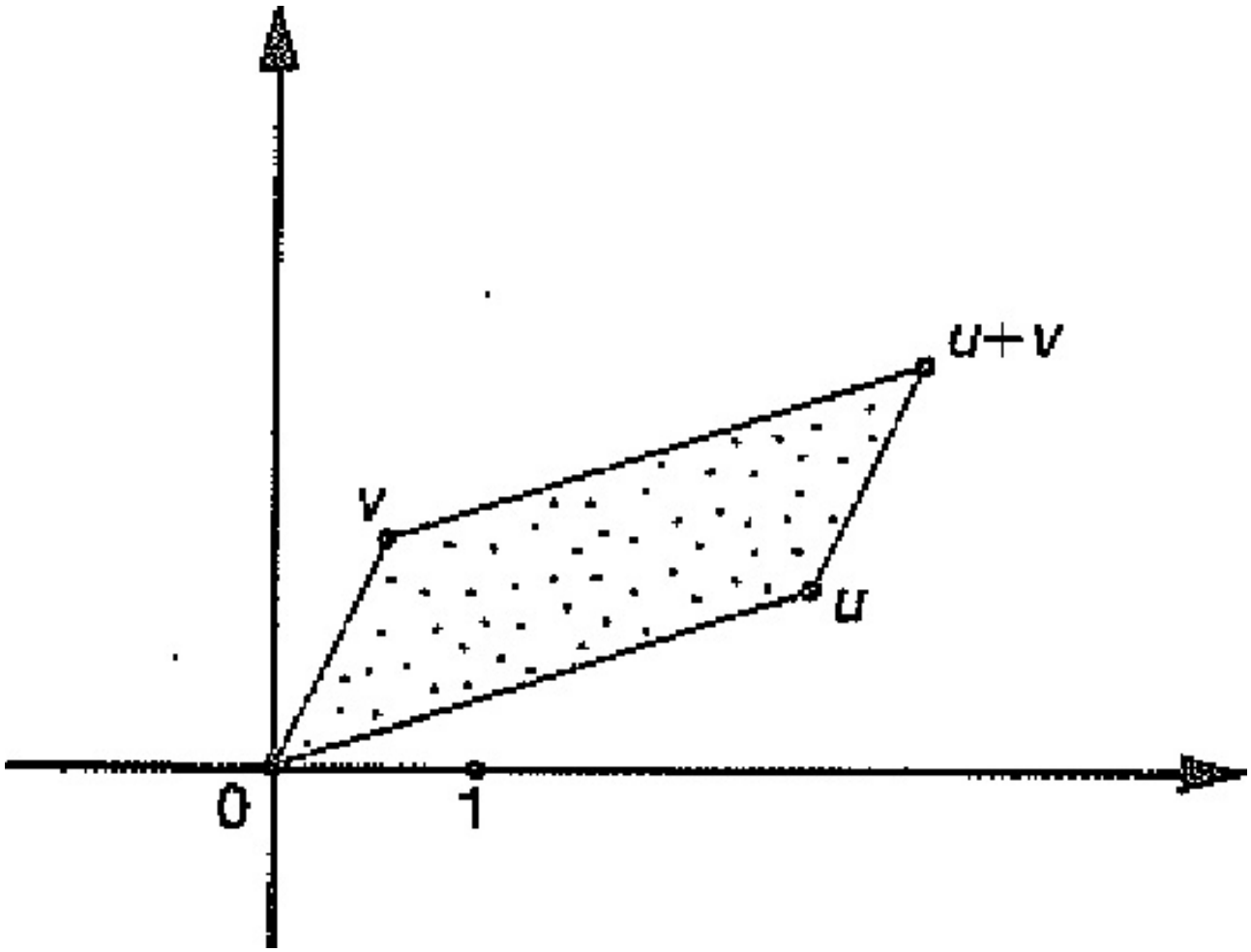


Şekil 3.9: u = 1 + i 1.3, v = -2 + i, ve w = -1.5 - i 0.4 noktalarının Argand düzlemindeki yerleri.

Argand bu gösterimi 1806'da bulmuştu ama gerçekte bu tarihten 9 yıl önce aynı sonuca Norveçli kadastrocu Caspar Wessel ulaşmıştı. Alışılmış (fakat tarihsel yönden doğru olmayan) kullanıma sadık kalarak, kompleks sayıların standart geometrik tanımına *Argand düzlemi* diyeceğim.

Argand düzlemi x, y Kartezyen koordinatlarıyla verilen bildiğimiz Eukleides düzlemidir. *x* (sağa doğru artı, sola doğru eksi ölçülen) yatay uzaklıkları, *y* ise (yukarı doğru artı aşağı doğru eksi ölçülen) düşey uzaklıkları göstermektedir. Böylece $z = x + iy$ kompleks sayısı Argand düzleminde koordinatları (x, *y)* ile verilen bir noktayla gösterilir. 0 (bir kompleks sayı olarak düşünüldüğünde) koordinat merkeziyle, 1 ise *x* ekseni üzerindeki bir noktayla gösterilir (Bkz. Şekil 3.8).

Argand düzlemi kompleks sayılar kümesini basitçe bir geometrik resim üzerinde göstermeye yarar. Bu bizler için pek yeni bir şey değil. *Reel* sayıları bir geometrik resim, yani her iki yönde sonsuza uzanan bir doğru üzerinde nasıl düzenleyeceğimizi zaten bilmekteyiz. Bir noktaya 0, bir diğer noktaya 1 dersek, 2 noktasını öyle yerleştiririz ki 1'den 2'ye yer değiştirme, 0'dan 1'e yer değiştirmeye eşittir, 0 ile 1 noktalarının tam ortasında 1/2 vardır. -1, 0'ın soluna, 0 tam -1 ile 1 arasında kalacak şekilde yerleştirilir vb. Bu biçimde gösterilen reel sayılar kümesine *reel eksen* denir. Kompleks sayılar için kullandığımız koordinatlar aslında iki reel sayıdan ibarettir. *a* + ib kompleks sayısı a ve b reel sayıları ile gösterilir.



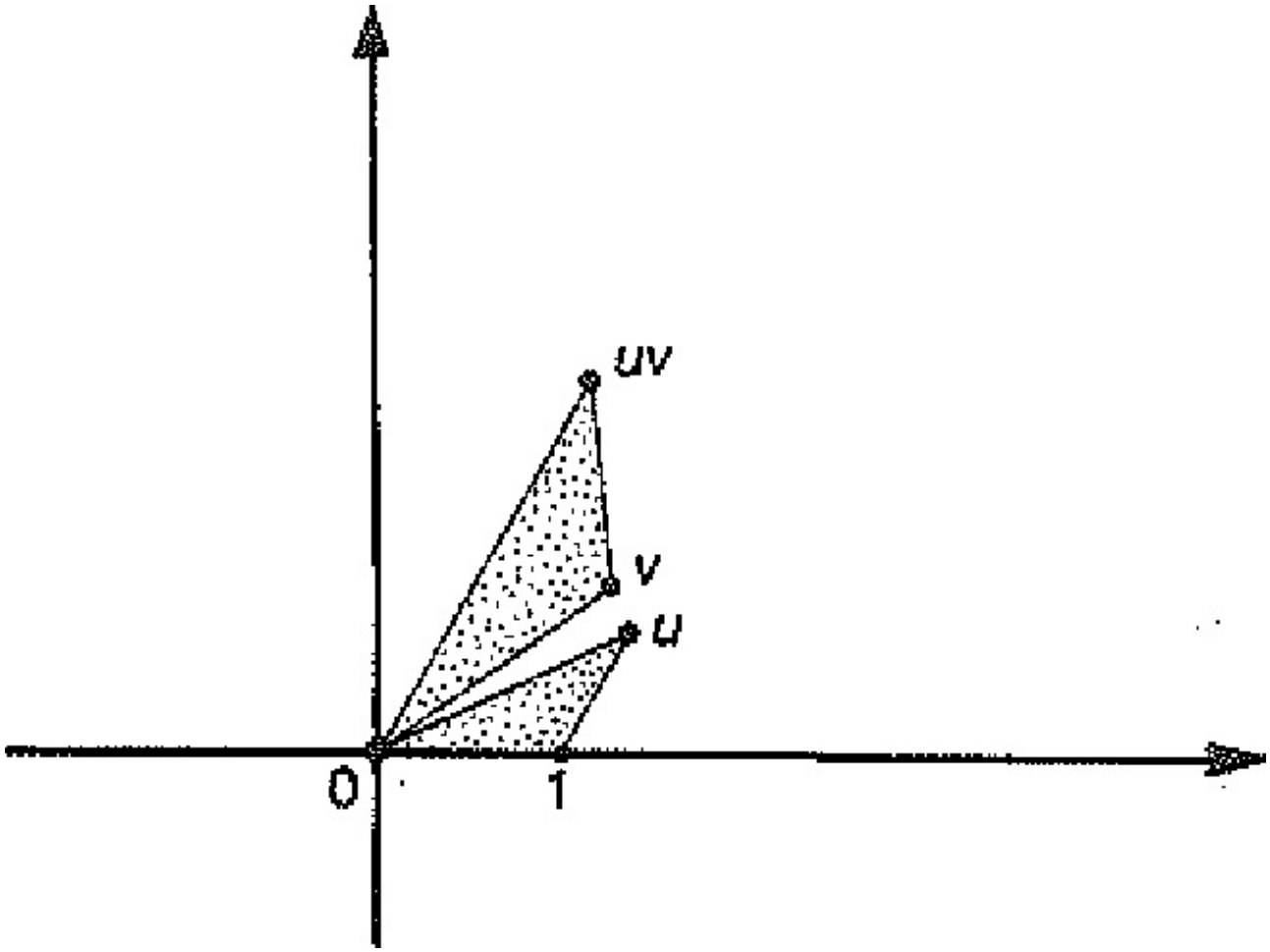
Şekil 3.10: u ve v kompleks sayılarının paralelkenar yöntemiyle elde edilen u + v toplamı.

Bu iki sayı düzlemde bir noktanın koordinatlarıdır. Örnek olması için Şekil 3.9'da (Argand düzleminde) *u* = 1 + i 1,3, *v* = - 2 + i, *w* = -1,5 - i 0.4 kompleks sayılarının yerlerini yaklaşık olarak gösterdim.

Kompleks sayılar üzerindeki temel cebirsel işlemler olan toplama ve çarpma böylece açık bir

geometrik tanıma kavuşur. Önce toplamayı ele alalım, *u* ve *v* Argand düzlemine yukarıda açıklandığı gibi yerleştirilmiş iki kompleks sayı olsun, *u* + *v* toplamı iki noktanın 'vektör toplamı'yla gösterilir.

Yani *u* + *v* noktası u *ve y* noktaları ile koordinat merkezi 0 noktasının oluşturduğu paralelkenarın dördüncü köşesinde bulunur. Bu tanımla (bkz. Şekil 3.10) gerçekten iki kompleks sayının toplamının verildiğini görmek pek zor değildir; ancak bu ispatı vermeyeceğim.



Şekil 3.11: u ve v kompleks sayılarının çarpımı uv, 0, v, uv tarafından oluşturulan üçgenin, 0, 1, u tarafından oluşturulan üçgene benzemesini sağlar. Buna özdeş olarak, uv'in 0'dan azaldığı, u ve v'in 0'dan uzaklıklarının çarpımına eşittir; 0, 1, uv uv'in reel eksen (yatay eksen) ile yaptığı açı, u ve v noktalamam bu eksenle yaptıkları açıların toplamıdır.

uv çarpımının da açık ama belki biraz daha zor görülen bir geometrik yorumu vardır (Şekil 3.11) (İspatı yine vermeyeceğim). 1 ile *uv* noktaları arasında kalan açı, 1 ile u arasındaki açı ile 1 ile *v* arasındaki açının toplamıdır (Tüm açılar saat yönünün tersine artı olarak ölçülmektedir), *uv* noktasının, merkeze uzaklığı ise u'nun merkeze uzaklığı ile v'nin merkeze uzaklığının çarpımına eşittir. Bu 0, *v* ve *uv* noktalarının oluşturduğu üçgeninin 0, 1. ve *u* noktalarının oluşturduğu üçgene benzer olduğunu söylemeye eşdeğerdir (Bu geometrik yapılarla tanış olmayan gayretli bir okuyucu bu kuralların kompleks sayıların toplamı ve çarpımı için verilen cebirsel kuralların ve yukarıdaki trigonometrik özdeşliklerin doğrudan birer sonucu olduklarını göstermek isteyebilir).

Mandelbrot Kümesinin İnşa Edilmesi

Artık Mandelbrot kümesinin nasıl tanımlandığını görecek durumdayız, *z* rasgele seçilmiş bir kompleks sayı olsun. Bu sayı Argand düzleminde bir noktayla gösterilebilir. Şimdi *z* sayısını yeni bir

kompleks sayıya götüren $z \rightarrow z^2 + c$ gönderimini düşünelim. Burada c *sabit* (yani verilmiş) bir kompleks sayıdır, $z^3 + c$ sayısı Argand düzleminde yeni bir noktayla verilir. Örnek olarak c sayısı 1.63 - i 4.2 diye verilmişse; bir z sayısı

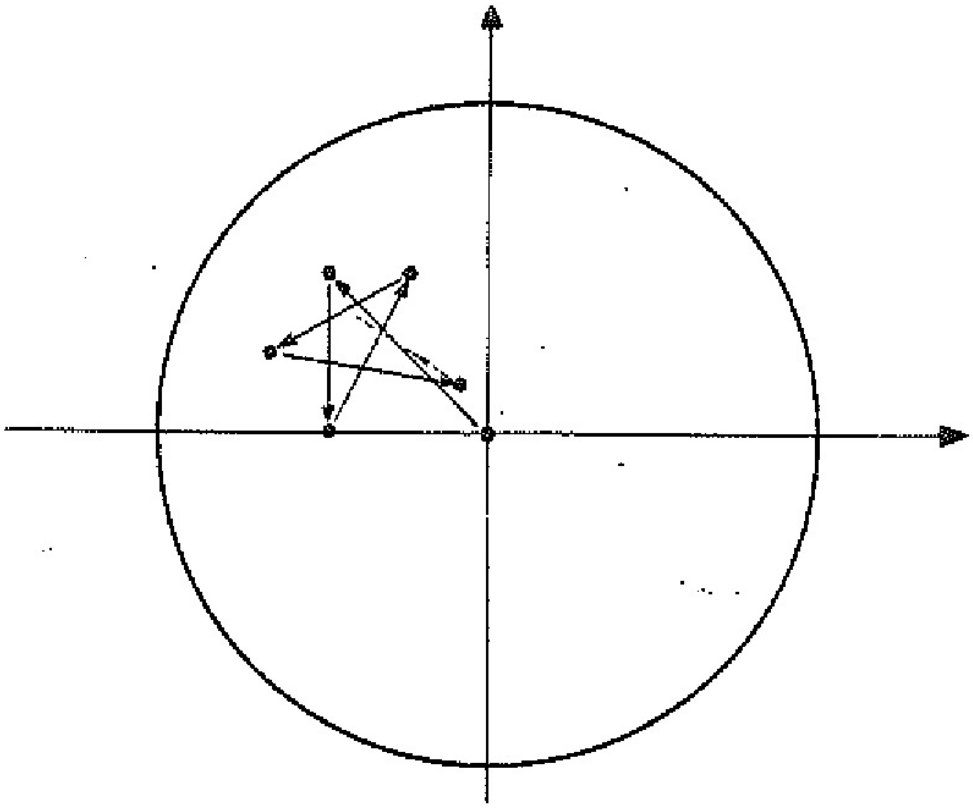
$z \rightarrow z^2 + 1.63 - i\ 4.2$ sayısına gider. Örneğin 3 yerine

$3^2 + 1.63 - i\ 4.2 = 9 + 1.63 - i\ 4.2 = 10.63 - i\ 4.2$ konmalıdır, $-2.7 + i\ 0.3$ yerine ise

$(-2.7 + i, 0.3)^2 + 1.63 - i\ 4.2 = (-2.7)^2 - (0.3)^2 + 1.63 + i\ (2(-2.7)\ (0.3) - 4.2]$

$= 8.83 - i\ 5.82$

konmalıdır. Sayılar çok karmaşıklaşınca, hesapları bir elektronik bilgisayarda yürütmek gerekir.



Şekil. 3.12 Eğer Argand düzlemindeki noktalar dizisinin tüm noktaları bir çemberin içine alınabiliyorsa, diziye sınırlı denir. (Şekildeki dizi c=1/2+1/2'i sayısından türetilmiştir.)

Şimdi, e ne olursa olsun, 0 sayısı verilen c sayısına gider. Peki c'nin kendisi hangi sayıya gider? c sayısı yerine $c^2 + c$ konmalıdır. Bu süreci ilerlettiğimizi düşünelim ve $c^2 + c$ sayısının yerine ne koyacağımıza bakalım. Böylece $(c^2 + c)^2 + c = c^4$ 4- $2c^3 + c^2 + c$ buluruz. İşlemi bir kez daha tekrarlayalım. Bir sonraki sayı

$(c^4 + 2\ c^3 + c^2 + c)^2 + c = c^8 + 4c^7 + 6c^6 + 6c^5 + 5c^4 + 2c^3 + c^2 + c$ çıkar, işlem bunun üzerinde tekrarlanır ve böylece sürer gider. Böylece 0 ile başlayan bir kompleks sayılar dizisi elde ederiz:

$0, c, c^2 + c, c^4 - 2c^3 - c^2 + c, \ldots$

Bazı *c* kompleks sayıları için, elde ettiğimiz bu sayı dizisi hiçbir zaman Argand düzleminin merkezinden çok uzaklara gidemez. Daha doğru bir şekilde söylersek; böyle c seçimleri için elde edilen sayı dizisi *sınırlıdır*. Yani dizinin bütün elemanları, merkezi koordinat merkeziyle çakışan

sabit yarıçaplı bir *çemberin* içinde kalırlar (Şekil 3.12). Bunun iyi bir örneği $c = 0$ durumunda görülür. Çünkü bu durumda dizinin tüm elemanları 0'a eşittir. Sınırlı sayı dizisine ikinci örnek $c = -1$ durumudur. Çünkü bu durumda ise dizi 0, -1, 0, -1, 0, -1, ...; şeklindedir. Diğer bir örnek olan c = i için sayı dizisi 0, i, i-1. -i, i-1, -i, i-1, -i... şeklindedir. Halbuki, diğer pek çok başka c kompleks sayıları için dizi merkezden giderek daha uzaklaşarak sonsuza varır, yani dizi ıraksaktır; sabit yarıçaplı bir çember içine alınamaz. Bu duruma bir Örnek c = 1 seçimidir, çünkü elde edilen dizi 0, 1, 2, 5, 26, 677, 458330, ... şeklinde sonsuza gitmektedir, $c = -3$ seçiminde de dizi 0, -3, 6, 33, 1086, ... şeklinde aynı davranışı gösterir, c = i -1 için de dizi 0, i-1, -i-1, -1 + 3İ, -9 - i5, 55 + İ91, -5257 + i 10011, ... şeklinde sınırsızdır.

Mandelbrot kümesi, yani, bizim Tor'Bled-Nam ülkesindeki *siyah* bölge tam olarak Argand düzleminde sınırlı dizilere karşı gelen c sayılarından ibarettir. *Beyaz* bölge ise sınırsız dizilere karşı gelen *c* noktalarından oluşur. Önceden gördüğümüz o ayrıntılı resimlerin hepsi bilgisayar çıktılarından çizilmişti. Bilgisayar sistematik olarak bütün *c* kompleks sayılarını tarar. Her bir *c* değeri için olası 0, c, $c^2 + c$, ... dizisini bulur ve uygun bir kritere göre dizinin sınırlı kalıp kalmadığına karar verir. Eğer dizi sınırlıysa, bilgisayar ekranda c sayısına karşı gelen noktaya bir siyah yuvarlak koyar. Eğer dizi sınırlı değilse, beyaz yuvarlak koyar. Sonuçta, ekranda taranan aralıktaki her bir minik kutuya siyah mı yoksa beyaz mı nokta konacağına tek tek bilgisayar tarafından karar verilir.

Mandelbrot kümesinin karmaşıklığı, özellikle bir matematik tanımı olarak kümenin tanımının ne kadar basit olduğu göz önüne alındığında, gerçekten çok dikkat çekicidir. Ayrıca kümenin genel yapısı seçmiş olduğunuz $z \rightarrow z^2 + c$ gönderiminin cebirsel şekline pek de duyarlı değildir. Diğer pek çok tekrarlayan kompleks gönderimler de (örneğin $z \rightarrow z^3 + iz^2 + c$) (başlangıç için uygun bir sayının seçilmesi kaydıyla; bu belki 0 değil de değeri her gönderim için açık bir matematik kural tarafından uygun olarak belirlenen başka bir sayı olabilir) şaşılacak derecede benzer yapılar verir. Gerçekten, tekrarlayan kompleks gönderimlere göre, bu 'Mandelbrot' yapılarının bir tür evrensel ya da mutlak niteliği bulunmaktadır. Bu yapıların incelenmesi matematikte *karmaşık dinamik sistemler* adıyla anılan başlı başına bir konu olmuştur.

Matematiksel Kavramların Platonik Gerçekliği?

Matematikçinin dünyasındaki nesneler ne kadar 'gerçek'? Bir görüşe göre, söz konusu nesnelerin tümü ile ilgili hiçbir şey gerçek görünemez. Matematiksel nesneler yalnızca kavramlardır; çoğu kez, çevremizdeki dünyada oluşumları ve görünüşteki düzenleriyle matematikçiyi etkileyerek onların uslarında idealleştirdikleri nesnelerdir ama yine de ussal idealleştirmeden öte değillerdir. İnsan usunun zorunlu ürünlerinden başka bir şey olabilirler mi? Aynı zamanda matematik kavramlarında, bir matematikçinin ussal yargılarının çok ötesine uzanan derin bir gerçeklik gözlenebilir. Sanki, kavramlar değil de insan düşüncesi bazı dış gerçeğe -kendine özgü olup bize yalnız bir kısmını gösteren gerçeğe- doğru yönlendirilmektedir.

Mandelbrot kümesi bu konuda etkileyici bir örnektir. Olağanüstü özenle tasarımlanmış olan bu kümenin mimarı ne bir kişi ne de bir matematikçi ekibidir. Sistemi ilk kez[3] niceleyen Polonya kökenli Amerikalı matematikçi (ve fraktal teorisinin öncüsü) Benoit Mandelbrot, çok ilginç bir şeylerin izi üzerinde olduğunu farketmekle birlikte, sistemin özünde gizli harika yapıyı önceleri

anlıyamadı. Bilgisayarının ekranında ilk görüntüler belirmeye başladığı zaman, izlemekte olduğu belirsiz yapıların, bilgisayarının yanlış işlem yapmasından kaynaklandığını sandı (Mandelbrot 1986)! Bu yapıların kümenin kapsamında yer aldığına ancak bir süre sonra inandı. Mandelbrot kümesinin karmaşık yapısının tüm ayrıntılarını hiç birimizin anlaması veya bunların bir bilgisayar tarafından gözler önüne serilmesi olanaksızdır. Sanki düşünce sistemimizin bir parçası değil de kendine özgü gerçeğe sahip bir yapıdadır. Kümeyi hangi matematikçi veya bilgisayar uzmanı incelerse incelesin, *aynı* temel matematiksel yapıya yaklaşık yapılar bulacaktır. Bilgisayarlar için de durum farklı değildir; ancak, bilgisayarların işlem hızı, bellek kapasitesi, grafik gösterme yeteneği gibi farklı etkenleri, ayrıntılı çıktı miktarında ve hızında bazı farklar yaratabilir. Bu konuda bilgisayar, ilke olarak, bir deneysel fizikçinin fiziksel dünyanın yapısını keşfetmek için kullandığı bir deney cihazı gibi kullanılmaktadır. Mandelbrot kümesi, insan aklının bir buluşu değildir; bir keşiftir. Everest Dağı nasıl orada öylece duruyorsa, Mandelbrot kümesi *de orada* öylece duruyor!

Aynı şekilde, kompleks sayılar sistemi de, herhangi bir matematikçinin akıl yapısının çok ötesinde, derin ve süresiz bir gerçeğe sahiptir. Kompleks sayılar fikri ilk kez Gerolamo Cardano tarafından yazılan bir kitapta ileriye sürüldü. 1501-1576 yılları arasında yaşayan bir doktor olan İtalyan Cardano aynı zamanda bir kumarbaz ve yıldız falcısıydı (İsa'nın falına da bakmıştı). 1545'de, cebir üzerine 'Ars Magna' adında önemli ve etkileyici bir kitap yazdı. Genel bir kübik denklemin köklerini veren çözümünün ilk kompleks ifadesi bu kitapta yer almıştır.[V] Ancak bu ifadenin belirli bir aşamasında, denklemin üç gerçek çözümünün bulunduğu 'indirgenemez' işlemlerde, *eksi bir sayının karekökünü* almak zorunda kaldığını fark etmiştir. Ona şaşırtıcı gelse de, bu kareköklerı alabilseydi, ancak bu şartla, yanıtı tümüyle (kesin yanıtın anlamı daima gerçektir) verebileceğini anladı. Daha sonraki yıllarda, 1572'de, Raphael Bombelli, 'I'Algebra' adlı kitabında Cardano'un eserini biraz daha ilerleterek, kompleks sayıların esas cebirini incelemeye başladı.

Başlangıçta, eksi sayıların söz konusu kareköklerinin alınması işlemi sadece bir araç -belirli bir amaca hizmet eden matematiksel bir buluş- olarak görünse de, bu matematiksel nesnelerin öngörülen amaçlarını aşarak çok daha fazla işler başardıkları daha sonra anlaşılmıştır. Yukarıda değindiğim gibi, kompleks sayıların matematik işlemlerine dahil edilmelerindeki ilk amaç kareköklerin serbestçe alınmasını sağlamak idiyse de, bu sayıları kullanarak, herhangi bir karekökün alınması veya ne çeşit olursa olsun bir cebir denkleminin çözümü olasılığına bir ödüle sahip olur gibi sahip oluyoruz. Çok geçmeden bu sayıların, daha önce aklımızın ucundan bile geçmeyen sihirli özelliklere sahip olduklarını anlıyoruz. Bu özellikler, bu sayıların doğasında öylece bulunuyorlar. Bu özellikler, kuşku götürmez ileri görüşlülüklerine rağmen, ne Cardano, ne Bombelli, ne Wallis, ne Coates, ne Euler, ne Wessel, ne Gauss, ne de bir başka büyük matematikçi tarafından bu sayılara kazandıramamıştır. Böylesine bir sihir, sayıların zaten doğasında vardı ve matematikçiler yavaş yavaş bu sihirin farkına vardılar. Cardano, kompleks sayıları ilk kez ortaya koyarken, daha sonraları Cauchy integral formülü, Riemann gönderimi teoremi, Lewy genişletme özelliği gibi çeşitli adlarla tanımlanacak bu özelliklerin farkında değildi. Bu özellikler, 1539'larda Cardano'un karşılaştığı sayıların özellikleri olup hiçbir değişime uğramamıştır.

Matematik icad mı yoksa keşif midir? Matematikçiler sonuçlara ulaştıkları zaman, özünde gerçek olmayan, özenle tasarımlanmış zihinsel yapılar mı üretiyorlar, yoksa bu yapılar öylesine güçlü ve ince tasarımlanmış ki, mimarları üzerinde 'gerçek' izlenimi yaratıp onları yanıltıyorlar mı? Yoksa matematikçiler, aslında, bu yapıların özünde zaten bulunan gerçekleri, matematikçilerin işlemlerinden tamamen bağımsız varolan gerçekleri ortaya mı çıkarıyorlar? Sanırım, şimdiye kadar, birincisinden çok ikinci görüşe yatkın olduğumu, en azından kompleks sayılar ve Mandelbrot kümesi açısından

ikinci görüşü benimsediğimi okurlarım anlamışlardır.

Ancak konu bu kadar basit değil. Matematikte öyle örnekler vardır ki 'keşif sözcüğünün kullanılması 'icad' sözcüğünün kullanılmasından daha doğru olur. İşte bu gibi durumlarda yapının özü, ona daha sonra kazandırılan özelliklerden fazla önem kazanır. Bunlar, matematikçilerin 'Tanrı'nın işi' ile karşı karşıya kaldıkları durumlar diye niteleyebileceğimiz durumlardır. Ancak, matematiksel yapının böyle bir eşsiz özelliğe sahip olmadığı durumlar da vardır. Bir sonuca götüren kanıtın tam ortasında matematikçi, çok özel bir sonuca ulaşabilmek için önceden tasarımlanmamış ve 'tek' olarak tanımlanamıyacak bir yapıyı kullanmak ihtiyacını duyabilir. Bu durumda elde edilecek sonucun yapısı, başlangıçta işleme dahil edilen yapıdan farklı olmayacağı için 'keşif' yerine 'icat' sözcüğünü kullanmak daha doğru olur. Bunlar gerçekten de 'insan işi'dir. Bu açıdan, gerçek matematiksel keşifler, genellikle, 'sadece bir buluş' olan icatdan daha büyük başarılar veya esinlenmiş düşünceler olarak kabul edilebilirler.

Bunun gibi sınıflamalar, sanatta veya teknikte kullanılanlardan pek farklı değildir. Büyük sanat yapıtları, diğerlerine göre 'Tanrı'ya daha yakın'dır. Sanatçılar arasında, en büyük yapıtlarının bir çeşit göksel varlığa sahip sonsuz gerçekleri yansıttıklarına dair duygu yaygındır.[VI] Buna karşılık, daha az sözünü ettiğim göksellik, sonsuzluk gibi duyguların matematikte, özellikle derin matematiksel kavramlarda çok daha güçlü olduğunu hissetmekten kendini alamıyorum. Matematik kavramlarında insanı dürtüleyen öyle bir eşsizlik ve evrensellik var ki, benzerini sanatta veya teknikte bulmak olası değil. Matematiksel kavramların böyle bir süresiz, göksel anlamda varoldukları fikri Yunan felsefeci Platon tarafından eski çağlarda (İ.Ö. 360) öne sürülmüştür. Sonuçta, bu görüş çoğu kez matematiksel platonizm olarak anılmaya başlanmıştır. Bu konu, ilerideki bölümlerde bizim için bir hayli önemli olacaktır.

I. Bölüm'de, ussal olgunun varlığını bir algoritmanın matematiksel düşüncesi içerisinde bulduğu savına dayanan güçlü Al görüşünü ayrıntılı şekilde ele almıştım. II. Bölüm'de, algoritma kavramının gerçekten derin ve 'Tanrı vergisi' bir fikir olduğunu vurgulamıştım. Bu bölümde ise, bu gibi 'Tanrı vergisi' matematik görüşlerinin, bizim 'dünyevi' varlığımızdan bağımsız bir çeşit ebedi varlığa sahip olduklarını tartışmaktayız. Ussal olgu için göksel tipte bir varolma olasılığını sağlayarak güçlü Al görüşüne katkıda bulunabilir miyiz? Bana göre bunu yapabiliriz. Daha sonraki bölümlerde bu görüşün hiç de yabancısı olmayan bir görüşü savunacağım. Fakat, ussal olgu kendine böyle genel bir sığınak bulabilecekse bunu algoritma kavramıyla yapabileceğine inanmıyorum. Algoritmadan çok daha ustaca tasarımlanmış bir araca ihtiyacımız olacak. Algoritmik kavramların, matematiğin çok dar ve sınırlı bölümünü oluşturduğu savı, bundan sonraki tartışmamızın konusunu oluşturacak. Bir sonraki bölümde algoritmik olmayan matematiğin kapsamı ve ustaca tasarım ile ilgili fikirler edinmeye başlayacağız.

Notlar

[←1] Bkz. Mandelbrot (1986). Mandelbrot sisteminin ilginç renkli resimlerinin de yer aldığı Peitgen ve Richter'den (1986) uyarlayarak yaptığım alıntılar. Daha ilginç şekiller için bkz. Peitgen and Saupe (1988).

[←2] Bildiğim kadarıyla, herhangi bir reel sayı için, ondalık açılımındaki n'inci hanenin gerçekte ne olduğunu saptamak konusunda daima bir çeşit kural olması görüşü alışılmamış olmasına karşın tutarlı bir görüştür. Ancak, böyle bir kural, önceden tasarımlanmış biçimsel bir sistemde etkili ya da hatta tanımlanabilir olmayabilir (bkz. IV. Bölüm). Umarım tutarlıdır, çünkü bu görüşe bağlı kalmak

arzusundayım.

[←3] Bu kümeyle ilk kez kimin karşılaştığı tartışma konusu olmuştur (bkz. Brooks and Matelski 1981, Mandelbrot 1989); oysa, böyle bir tartışmanın başlatılmış olması dahi, kümenin bulunuşunun bir icaddan çok bir keşfe benzediği görüşüne daha fazla destek sağlar.

Açıklamalar

[←I] Gerçekte, sıfir yıl kullanılmadığı için, tarihlerle ilgili normal uygulamalarda buna gereğince uyulmaz.

[←II] 10^{20} ile 1'den sonra 20 adet sıfir konarak bulunan 100000000000000000000 sayısı gösterilir.

[←III] e = 2.7182818285, ... sayısı (doğal logaritmaların temeli, ve n sayısının önemi ile kıyaslanabilir matematiksel öneme sahip irrasyonel bir sayı), e = 1 + 1/1 + 1/(1 x 2) + 1/(1 x 2 x 3)+... olarak tanımlanır; ez'nin anlamı, e'nin z'nci kuvveti olup, aşağıdaki bağıntı sağlanır:

$e^z = 1 + z\,1 + 2^z\,(1 \times 2) + z^1\,(1 \times 2 \times 3) + \ldots$

[←IV] 'Topolojik' sözcüğü, bazen 'esnek yüzey geometrisi' olarak da anılan bir tür geometri kapsamında kullanılmıştır; bu geometride gerçek uzaklıklar önemli olmayıp, nesnelerin yalnızca süreklilik özellikleri dikkate alınır.

[←V] Kısmen, Scipione del Ferro ve Tartaglia tarafından daha önce yazılan kitaba dayanır.

[←VI] Tanınmış Arjantinli yazar Jorge Luis Borges'in sözleriyle "... ünlü bir ozan bir mucidden ziyade bir kâşiftir." önemli yapıtları, daha rastgele, daha 'dünyevi' özelliktedir. Aynı şekilde, basit, umulmadık bir fikrin uygulanmasıyla gayet ekonomik olarak gerçekleştirilen bir mühendislik yapıtı da bir icatdan çok bir keşif olarak nitelenebilir.

IV. Bölüm
Doğruluk, Kanıt ve Sezgi

Hilbert'in Matematik Programı

Doğru nedir? Dünya hakkında neyin doğru neyin yanlış olduğuna dair yargılarımızı nasıl oluştururuz? Sadece bazı *algoritmayı* -kuşkusuz doğal seçimin güçlü etkinliğini daha az etkin algoritmalara yeğleyerek- uygulamakla mı yetiniriz? Doğrunun ulu katına ulaşabilen *algoritma* dışında, bir başka yol -önsezi, içgüdü veya sezgi- var olamaz mı? Zor bir soruya benziyor bu. Yargılarımız, duygu-verileri, mantıksal işlem ve tahmin işlemlerinden oluşan karmaşık, içten-bağlantılı sistemlere dayanır. Ayrıca, dünya ile ilgili konularda neyin doğru neyin yanlış olduğu hakkında genel bir görüş birliği olmayabilir. Soruyu basite indirgemek için, sadece matematiksel doğrulukla ilgilenelim. Matematik sorularıyla ilgili olarak yargılarımızı -belki hatta 'belirli' bilgimizi- nasıl oluştururuz? Böyle bir soru bize hiç olmazsa biraz daha açık seçik anlaşılabilir görünüyor. Neyin gerçekten doğru, neyin gerçekten yanlış olduğu gibi bir soruyu gerektirmiyor -yoksa gerektiriyor mu? Sahiden *matematiksel* doğruluk nedir?

Matematiksel doğruluk ile ilgili soru, eski Yunan felsefecilerine ve matematikçilerine -ve, kuşkusuz, daha da eskilere- kadar uzanan eski bir sorudur. Ancak, yaklaşık son yüz yılda, soruya açıklık getiren önemli aşamalar, çok ilginç *yeni* sezgiler elde edilmiştir. İşte bu yeni gelişmeleri anlamaya çalışacağız. Düşünme sürecimizin gerçekten tümüyle algoritmaya dayalı olup olamıyacağı sorusuna parmak basan çok önemli konuları tartışacağız. Bu konularda uzlaşmaya varmamız bizim için önemlidir.

On dokuzuncu yüzyılın sonlarında matematikçiler, matematiksel kanıtlama yöntemlerinin giderek daha güçlenmesinin de etkisiyle, büyük gelişmeler kaydetmişlerdi (Daha önce tanıştığımız David Hilbert ve Georg Cantor, daha sonra tanışacağımız büyük Fransız matematikçi Henri Poincare, bu gelişmelerin öncüleridir). Böylece matematikçiler bu güçlü yöntemleri, giderek artan bir güvenle uygulamaya başlamışlardı. Yöntemlerin çoğunda, sonsuz elemanlı kümeler[I] dikkate alınıyor, kümelere gerçek 'şeyler' -potansiyel olarak varoluşlarının ötesinde, varoluşlarını tamamlamış bütünler- gözüyle bakmanın mümkün olduğu benimsendiği için kanıtlar çoğu kez başarılı oluyordu. Bu güçlü fikirlerin pek çoğu Cantor'un, tutarlı biçimde sonsuz kümeler kullanarak geliştirdiği *sonsuz sayılar* kavramından kaynaklanmıştır (Bir önceki bölümde bu konuya şöyle bir göz atmıştık).

Ancak, matematikçilerin yöntemlere duydukları güven, 1902'de İngiliz mantık bilimcisi ve felsefeci Bertrand Russell'ın ünlü ikilemini (Cantor tarafından da beklendiği gibi, 'köşegen' yönteminin doğrudan sonucu olarak) ortaya atmasıyla sarsıldı. Russell'ın tezini anlamak için önce, kümelerin tam bütünler olarak ele alınmasına ilişkin bir fikir edinmemiz gerekir. Belirli bir özellikle karakterize edilmiş herhangi bir *küme* düşünelim. Örneğin, *kırmızı* nesneler kümesi, kırmızılıkla karakterize edilir. Herhangi bir nesne yalnız ve yalnız kırmızılığa sahip ise bu kümeye ait olabilir. Bu durumda, bir tek nesneyi sahip olduğu bir özellikle tanımlayabiliyorsak, böyle nesnelerden oluşan kümenin tümü için aynı özelliği kullanabiliriz: 'Kırmızılık', bütün kırmızı nesnelerin kümesidir (Diğer başka kümelerin de, böyle basit bir özellikle karakterize edilmeyen elemanlara sahip kümelerin de, 'orada' yer aldığını algılayabiliriz).

Kavramları, kümelerle tanımlama fikri, Alman mantık bilimcisi Gottlob Frege tarafından 1884'de ileri sürülen yöntemin özünü oluşturmuştur. Bu yöntemde *sayılar,* kümelerle tanımlanmaktadır. Örneğin, gerçek 3 sayısı ne demektir? 'Üçlük' özelliğini biliyoruz ama 3'ün kendisi nedir? 'Üçlük', nesneler *topluluğunun* bir özelliğidir, bir başka deyişle, kümelerin bir özelliğidir. Bir küme, yalnız ve yalnız üç elemandan oluşuyorsa bu küme 'üçlük' özelliğine sahip demektir. Örneğin, bir olimpiyat dalında madalya kazananlar kümesi, bu 'üçlük' özel-ligine sahiptir. Aynı şekilde, üç tekerlekli bisikletin tekerlekleri, normal yoncanın yaprakları, veya $x^3 - 6x^2 + 11x - 6 = 0$ eşitliğinin yanıtlarının oluşturduğu kümeler de 'üçlük' özelliğine sahiptir. Öyleyse Frege'nin, gerçek 3 sayısı ile ilgili tanımı nedir? Frege'e göre 3, kümelerin bir kümesi olmalıdır: 'Üçlük' özelliğine sahip tüm kümelerin kümesi.[1] Buna göre bir küme, yalnız ve yalnız Frege'in 3 kümesine ait ise, üç elemana sahip olabilir.

Bu biraz döngü gibi görünürse de, gerçekte değildir. *Sayıları,* genel bir tanımlamayla, denk kümelerin toplamları olarak tanımlayabiliriz; burada 'denk' sözcüğü, birbiriyle bire bir eşlenebilen elemanlara sahip anlamındadır (Daha basit ifadeyle 'aynı sayıda elemana sahip olmak' demektir). Buna göre 3 sayısı, elemanları arasında, örneğin, bir elma, bir portakal ve bir armuttan oluşan bir küme bulunan kümelerden birisidir. Bu tanımın, önceden değinildiği gibi, Church'ün 'Ʒ' tanımından çok farklı bir '3' tanımı olduğuna dikkatinizi çekerim. Bugünlerde daha popüler olan başka tanımlar da vardır.

Peki, Russell'ın ikileminden ne haber? Bu ikilem, aşağıda tanımlanan bir R kümesiyle ilgilidir:

R, bizzat kendileri, kendi elemanı olmayan bütün kümelerin oluşturduğu kümedir.

Bu tanıma göre R, kümelerin belirli bir koleksiyonudur; ve bir X kümesinin bu koleksiyon içerisinde yerini alması için gerekli kriter, X kümesinin kendinin, *kendi* elemanları arasında yer almamasıdır.

Bir kümenin gerçekten kendisinin bir elemanı olabileceğini varsaymak saçma mı? Hiç de değil. Örneğin, sonsuz kümelerden oluşan I kümesini *(sonsuz* sayıda birçok elemandan oluşan kümeler) ele alalım. Elbette sonsuz olarak çok sayıda *farklı* sonsuz kümeler vardır, bu nedenle, I'ın kendisi de sonsuzdur; I gerçekten kendine aittir! Öyleyse nasıl oluyor da Russell'ın anlayışı bizi bir ikilemle karşı karşıya bırakıyor? Soruyoruz: Russell'ın 'R' kümesi kendinin elemanı mıdır, yoksa değil midir? Değilse R'ye ait olmalıdır çünkü R, tamamiyle, kendi kendilerinin elemanı olmayan kümelerden oluşur. Buna göre R, sonuçta R'ye aittir -işte bir çelişki! Diğer taraftan R, kendisinin bir elemanı ise, 'kendisinin' sözcüğü bizzat R'yi ifade ettiğine göre R, elemanlarının bizzat kendilerince temsil edilmediği kümeye aittir, yani kendinin elemanı değildir -işte yine bir çelişki![II]

Bu, hiç de hafife alınacak bir uslamlama değildir. Russell, matematikçilerin kanıtlarında kullanmaya başladıkları dönemde küme-teorik uslamlamasını oldukça aşırı şekilde kullanmaktaydı. İşlerin kontrolden çıktığı açıkça görülüyordu ve ne çeşit bir usavarımın uygulanmasının doğru olacağı konusunda daha titiz davranmak gerekiyordu. Kullanılacak uslamlama yöntemi ikilem yaratmamalıydı ve evvelce gerçek oldukları bilinen ifadelerden yalnız doğru ifadeler çıkarılmalıydı. Russell, meslektaşı Alfred North Whitehead ile birlikte, aksiyomlardan ve yöntemsel kurallardan oluşan son derece biçimsel bir matematik sistemini geliştirmeye girişti. Amaç, her çeşit doğru matematiksel uslamlamanın kendi projelerine uygulanabilirliğini kanıtlamaktı. Russell'ın kendi paradoksuna yol açan ikilemli usavarımlardan sakınmak için kurallar dikkatle seçildi. Russell ve Whitehead'ın birlikte ürettikleri özel proje çok büyük bir yapıttı. Ancak, çok zahmetli bir uğraştı ve sonunda projenin kendi kapsamında yer alan uslamlama yöntemleriyle sınırlı kaldı. II. Bölüm'de tanıştığımız

büyük matematikçi David Hilbert, daha uygulanabilir ve kapsamlı bir projeye başladı. Proje kapsamına, herhangi bir matematik alanı ile ilgili tüm doğru matematiksel uslamlama çeşitleri dahil edilecekti. Ayrıca Hilbert projenin çelişkiden uzak olduğunu kanıtlamanın da mümkün olacağını düşünüyordu. Böylece matematikçiler, sonsuza kadar, sarsılmaz bir temel üzerine oturabileceklerdi.

Ancak Hilbert ve arkadaşlarının umutları, 1931'de, 25 yaşlarında zeki bir Avusturyalı matematik mantıkçısı olan Kurt Gödel'in, Hilbert'in programını altüst eden teoremiyle söndü. Gödel'in teoremi şöyleydi: Aksiyomlardan ve yöntemsel kurallardan veya benzerlerinden oluşan herhangi bir kesin ('biçimsel') matematik sistemi, basit aritmetik teoremlerinin tanımlamalarını (II. Bölüm'de değinilen 'Fermat'ın son teoremi' gibi) kapsayacak kadar geniş kapsamlı olması ve çelişkisiz olması koşuluyla, sistemin kapsamına alınan yöntemlerle ne kanıtlanabilir ne de kanıtlanamaz bazı bildirimleri içermelidir. Buna göre, bu gibi bildirimlerin doğruluğu hakkında, onaylı yöntemlerle 'karar verilemez'. Gödel, gerçekte, uygun bir aritmetik teoremi şeklinde kodlandığında aksiyom sisteminin tutarlılığının bildiriminin 'karar verilemez' yöntem olduğunu kendiliğinden kanıtladığını göstermiştir. Söz konusu 'karar verilemezlik' kavramının özünü anlamak bizim için önemli olacaktır. Gödel teoreminin, Hilbert'in programım nasıl altüst ettiğini göreceğiz. Gödel teoreminin, aynı zamanda, sezgimizi kullanarak, herhangi bir formel matematik sisteminin sınırlarını aşmamızı nasıl sağladığını da göreceğiz. Bunları anlayabilmemiz, bundan sonra ele alacağımız konular açısından son derece önemlidir.

Formel Matematik Sistemleri

'Aksiyomlardan ve yöntemsel kurallardan oluşan formel matematik sistemi' ile ne kastettiğimizi biraz daha açıklamamız gerekecek. Matematiksel bildirimlerimizi ifade etmek için kullandığımız bir çeşit simgeler alfabesine sahip olduğumuzu varsayalım. Bu simgeler, 'aritmetiği' sistemimize dahil edebilmemiz için, doğal sayılardan oluşan bir kodlama sistemine de olanak sağlamalıdır. Kuralların tanımlanmasını gerektiğinden fazla karmaşık hale getirse de, bilinen 0, 1, 2, 3, ..., 9, 10, 11, 12, ... rakam sistemini kullanabiliriz. Doğal sayıların açılımını göstermek için, örneğin, 0, 01, 011, 0111, 01111,... gibi daha basit bir açılımı kullanabiliriz (veya, ortak bir noktada uzlaşalım diyorsanız, ondalık sistemi de kullanabiliriz). Ancak, diğer kodlama sistemleri, açıklamalarımızda bazı karışıklıklara neden olabilir endişesiyle ben tanımlamalarımda, yukarıdaki standart kodlama sisteminden vazgeçmeyeceğim. 'Kelimeleri' veya 'sayıları' ayırmak için bir 'ara' sembolüne de gereksinim duyabiliriz ama biz yine basit yolu seçerek virgül (,) kullanacağız. 'Değişken' doğal sayıları (veya tam sayıları, rasyonel sayıları, vb. ama yine de burada doğal sayılara bağlı kalalım) gösteren *t, u, v, w, y, z, t′, t″, t‴*,... gibi harfler kullanmamız gerekecek. Bir ifadede ortaya çıkabilecek değişken sayıların adedini sınırlamamak için *t′, t″*,... gibi harflere ihtiyacımız olabilir. *Simgelerin* gerçek adedinin sonsuz olabilmesi amacıyla, (′) işaretim, formel sistemin ayrı bir simgesi olarak kabul ediyoruz. Aritmetik işlemleri için =, +, x, vs.; çeşitli parantezler (,) [;]; *mantıksal* simgeler için & ('ve') ; => ('gerektirir'); ve ('veya'), <=> (yalnız ve yalnız); ~ ('değil' veya böyle değilse...') gibi simgeler gerekecektir. Ayrıca, mantık nicelik göstergelerine, 'niceleyici'lerine, ihtiyaç vardır: varolma niceleyicisi ∃ ('vardır... şöyle ki') ve evrensel niceleyici ∀ ('tümü için ... sahibiz'). Buna göre, 'Fermat'ın son teoremi' gibi bildirimler yazabiliriz:

$$\sim \forall\, w,x,y,z\, [(x+1)^{w+3} + (y+1)^{w+3} = (z+1)^{w+3}]$$

(bkz. II. Bölüm). ('3'ü göstermek için '0111' yazabilirdim ve biçimsel sisteme daha uygun

olabilecek "kuvvetini alma" kod sistemi kullanabilirdim; fakat, dediğim gibi, gereksiz karışıklıklardan kaçınmak amacıyla alışılagelmiş simgeleri kullanmakta ısrarlıyım). Yukarıdaki bildirim (ilk köşeli parantezde son bulan) şöyle okunur:

"...eşitliğini sağlayan *w, x,* y, z doğal sayıları yoktur."

∀ simgesini kullanarak Fermat'ın son teoremini yeniden şu şekilde de yazabiliriz:

$\forall\ w,x,y,z\ [\sim (x+1)^{w+3} + (y+1)^{w+3} = (z+1)^{w+3}]$

Mantık açısından yukarıdaki okumayla aynı anlamda olmak üzere şöyle okuruz:

'Hiçbir *w, x,* y, z doğal sayısı... eşitliğini sağlamaz.'

Teoremlerin tümünü gösterebilmek için harflerin kullanılması gerekiyor. Bu amaçla P, Q, R, S... gibi büyük harfler kullanacağım. Yine Fermat'ın teoremine uygulayalım:

$F = \sim \exists\ w,x,y,z\ [(x+1)^{w+3} + (y+1)^{w+3} = (z+1)^{w+3}]$.

Bir teorem, birden fazla değişkene de bağımlı olabilir; örneğin, Fermat önermesiyle belli bir $w + 3$ kuvveti için ilgilenebiliriz:[III]

$G(w) = \sim \exists\ x,y,z\ [(x+1)^{w+3} + (y+1)^{w+3} = (z+1)^{w+3}]$

Buna göre G(0), 'hiçbir sayının küpü iki artı işaretli sayının küplerinin toplamı olamaz' tezini ortaya koyarken G(l) aynı tezi dördüncü kuvvetler için savunur, vb. ('∃'den sonra '*w*' in bulunmayışına dikkatinizi çekerim). Bu haliyle Fermat teoremi, G(w), tüm *w* değerleri tarafından sağlanır şeklinde önerilebilir:

$F = \forall\ w[G(w)]$

G(), bir önergesel fonksiyona, yani bir veya daha fazla değişkene bağımlı önermeye örnektir.

Sistemin aksiyomları, simgelerin anlamları verildiğinde, doğruluğu kendini-kanıtlar nitelikte ortaya çıkacağı varsayılan genel önermelerden oluşan bitimli bir liste üretecektir. Örneğin, aksiyomlarımız arasında bulunabilecek önermeler veya önergesel fonksiyonlar P, Q, R () şunlar olabilir:

$(P\&Q) \Rightarrow P,\ \sim (\sim P) \Leftrightarrow P,$

$\sim \exists[R(x)]\ \forall[\sim R(x)]$

Bu fonksiyonların 'kendiliğinden kanıtlanmış doğruluğu' anlamlarından hemen bulunabilir (Birinci önerme: T ve Q'nün her ikisi de doğruysa, P doğrudur; ikinci önerme: 'P'nin doğru olmadığı doğru değildir' ile T doğrudur' önermelerinin özdeşliğini savunur; üçüncü önerme, Fermat'ın son teoremini ifade eden iki yöntem arasındaki mantıksal eşitliği örnekler. Aşağıdaki gibi bazı temel aritmetik aksiyomları da listeye dahil edebiliriz:

$\forall\ x,y[x+y=y+x]$

$\forall x,y,z[(x+y) \times z = (x \times z) + (y \times z)]$

Ancak, bu aritmetik işlemleri daha basit aksiyomlarla inşa ederek bunlardan uslamlama yöntemiyle elde edilecek bildirimleri teoremler olarak nitelemek tercih edilebilir. Yöntemin (kendini-kanıtlar) kuralları şöyle ifade edilebilir:

'P'den ve $P \Rightarrow Q$ aksiyomundan Q'u uslamlama ile elde edebiliriz.'

'$\forall x\ [R(x)]$ aksiyomundan, R(x)'deki x yerine bir doğal sayı kullanılarak herhangi bir önermeyi uslamlama yöntemiyle elde edebiliriz.'

Bütün bunlar, inşa edilmiş olan önermelerden yeni önermeleri nasıl elde edebileceğimizi gösteren

komutlardır.

Şimdi aksiyomlardan hareketle ve yöntemin kurallarını tekrar tekrar uygulayarak uzun bir önermeler listesi hazırlayabiliriz. Herhangi bir aşamada aksiyomları tekrar sahneye çıkarır, uzayıp giden listemize dahil etmiş olduğumuz önermelerden herhangi birini tekrar tekrar kullanmaya devam edebiliriz. Böyle doğru bir şekilde inşa edilen bir listeye dahil önermeler *teorem* adıyla anılır (ancak bunların pek çoğu, matematiksel bildirimler olarak önemsizdirler veya ilginç değillerdir). Kanıtlamak istediğimiz belirli bir P önermesine sahipsek, kurallara göre yapılandırılmış ve bizim P önermemiz ile son bulan böyle bir liste bulmalıyız. Liste sayesinde, P'nin *kanıtını* sistemin içerisinde bulabiliriz; ve P, böylece, bir teorem olabilir.

Hilbert'in programının amacı, matematiğin herhangi bir iyi-tanımlanmış alan ile ilgili *her çeşit* doğru matematiksel uslamlama yöntemini içine alacak kadar geniş kapsamlı bir aksiyomlar ve yöntemsel kurallar listesi yaratmaktı. Bizim matematik alanımız *aritmetik* olsun (Fermat'ın son teoremi gibi bildirimlerin yapılabilmesi için ∃ ve ∀ gibi niceleyici göstergelerin yer aldığı bir alan). Konumuzla ilgili olarak aritmetikten daha uygun bir alan bulamayız. Çünkü aritmetik, Gödel'in yönteminin uygulanabileceği kadar genel kapsamlı bir alandır. Hilbert'in programına uygun olarak, geniş kapsamlı bir aksiyomlar ve yöntemsel kurallar sistemine sahip olursak, herhangi bir aritmetik önermesi ile ilgili matematiksel kanıtın 'doğruluğuna' ait kesin kritere de sahip oluruz. Böyle bir aksiyomlar ve kurallar sisteminin, sistem çerçevesinde formüle edilebilen *herhangi bir* matematik bildiriminin doğru olup olmadığına prensipte karar verebilmeyi sağlayacak kadar *bütünlük* taşıdığı ümit edilmekteydi.

Hilbert ise, bir matematik önermeyi, diyelim P'yi, temsil eden bir simgeler dizisi için, P'nin doğru olup olmadığına bağlı olarak, ya P veya ~ P'nin kanıtlanabileceğini umuyordu. Simgeler dizisinin, 'dizim' açısından doğru, yani formalizmin tüm kurallarını -çift parantezlerin, vs. doğru yerleştirilmesi gibi- yerine getiren 'söz dizimi / sintaks' açısından doğru olduğunu varsaymak zorundayız. Ancak bu durumda P, iyi-tanımlanmış doğru veya yanlış anlama sahip olabilir. Hilbert'in ümidi gerçekleşseydi, önermelerin anlamı ile ilgili endişelerimizden tümüyle kurtulabilirdik! P, dizim bakımından doğru bir simgeler dizisi olurdu. P bir teorem ise (yani, sistemin içerisinde kanıtlanabiliyorsa) onu, doğruluk değeri doğru olarak, ~ P bir teorem olduğu taktirde onu, doğruluk değeri yanlış olarak niteleyebilirdik. Bunun bir anlam taşıması için, dizinin eksiksiz olmasının yanısıra *tutarlı* olmasına da ihtiyacımız vardır. Başka bir deyişle, P ve ~ P'nin her ikisinin de teoremler olarak yer aldığı bir P simgeler dizisi bulunmamalıdır. Aksi halde P, aynı zamanda hem **doğru** hem de **yanlış** olur!

Matematiksel bildirimlere, formel bir matematik sisteminde yer alan simge dizilerinden başka hiçbir şey olmadıkları gözüyle bakmak suretiyle anlamlarından kurtulma görüşü, matematiğin *formalizme* veya biçimselliğe bakış açısıdır. Bu görüşü benimseyenler için matematik bir çeşit 'anlamsız oyun'dur. Ancak bu görüşler bana çekici gelmiyor. Matematiğe özünü kazandıran -kör algoritma işlemleri değil- 'anlam'dır. Neyse ki Gödel formalizme ezici bir darbe indirdi! Bunu nasıl yaptı, görelim.

Gödel Teoremi

Gödel'in tartışmalarının bir kısmı çok ayrıntılı ve karmaşıktı. Ancak, bunların girdisini çıktısını incelememiz gerekmiyor. Öte yandan ana fikir, çok basit, güzel ve geniş kapsamlıydı. Teoremin karmaşık kısmı (bu kısım da büyük bir beceri eseriydi), formel sistemin her bir kuralının, ve çeşitli

aksiyomlarının aritmetik işlemlerine nasıl kodlanacağını gösteriyordu (Teoremin ana fikrinin geniş kapsamlı olmasının bir nedeni de, böyle bir kodlamanın yapılabilecek en verimli çalışmalardan birisi olduğunu anlamasıydı). Kodlamayı uygulamak için, önermeleri doğal sayılarla etiketlemenin bir yolunu bulmamız gerekiyor. Bu yollardan birisi, dizinin uzunluğuna göre genel bir sıralamanın her bir belirli uzunluğu için, formel sistemin tüm simge dizilerine sadece bir çeşit 'alfabetik' sıralamanın uygulanmasıdır (Böylece, alfabetik sıraya dizilmiş birim uzunlukta dizileri, yine alfabetik sırada iki birim uzunlukta diziler ve bunları üç birim uzunlukta diziler, vb. izleyebilir). Bu sıralamaya leksikografik sıralama[IV] adı verilir. Aslında Gödel daha karmaşık bir numaralama sistemi kullanmıştır, fakat, bu gibi farklar bizim için önemli değildir. Bizi özellikle ilgilendiren konu, yukarıda değinilen G(w) gibi tek değişkene bağlı önergesel fonksiyonlardır. Şimdi n'inci önergesel fonksiyonu (simgeler dizisinin seçilen sıralamasında) w'*e* uygulayalım;

$P_n\ (w)$.

İstersek numaralama sistemimizi biraz 'gelişigüzel' düzenleyerek bazı ifadelerin dizim bakımından doğru olmamasını sağlayabiliriz (Bu durumda, dizimsel yönden doğru dizime sahip olmayan ifadeleri kodlama sisteminden atmaya çalışmaktan çok daha kolay bir aritmetik kodlama sağlıyabiliriz). $P_{,,}\ (w)$ dizim bakımından doğru ise, *n ve w* doğal sayıları ile ilgili son derece iyi tanımlanmış bir aritmetik bildirimi olacaktır. Tam olarak hangi aritmetik bildirimi olduğu, kullanılacak numaralama sisteminin detaylarına bağlı olacaktır. Bu konu, teoremin karmaşık kısmının kapsamına girmekte ve burada ele aldığımız konu bakımından bizi ilgilendirmemektedir. Sistemdeki bir teoremin kanıtını oluşturan önermeler dizileri, tercih edilen sıralama planı kullanılarak, doğal sayılarla da etiketlenebilir.

Π_n sayısı, n'inci kanıtı göstersin (Yine, 'gelişigüzel bir numaralama' kullanabiliriz ve buna göre, *n*'in bazı değerleri ile ilgili *n* ifadesi dizim açısından doğru olmayacak ve bu nedenle hiçbir teoremi kanıtlamayacaktır).

Şimdi, *w* doğal sayısına bağlı aşağıdaki önergesel fonksiyonu ele alalım:

$\sim \exists\ x[\Pi_x$ kanıtlar $P_w(w)]$.

Köşeli parantezler içerisindeki bildirim kısmen kelime kullanılarak verilmiş olmasına karşın mükemmel ve tam olarak tanımlanmış bir bildirimdir: x'ci kanıtın gerçekte, *w* değerine uygulanan P_w ( ) önermesinin bir kanıtı olduğunu savunmaktadır. Köşeli parantezin dışındaki 3, değişkenlerden birini bildirimden çıkarma görevini üstlenmiştir ('... önermesini sağlayan *x* mevcut değildir') ve bu nedenle, yalnız bir değişkene, *w'e,* bağımlı bir önergesel aritmetik fonksiyonu elde etmiş oluruz. Bu bildirim, bir bütün olarak, $P_w(w)$ ifadesinin kanıtının bulunmadığını savunur. Ben bu ifadenin dizim açısından doğru söz dizimine sahip olduğu kanısındayım ($P_w(w)$ gramer açısından doğru ifade edilmemiş olsa da bu kanıdayım; dizimi yanlış ifade edilmiş bir ifadenin kanıtı olmayacağı için, $P_w(w)$'in doğru ifade edilmesi durumunda ifadenin tamamı da doğru olurdu). Gerçekte, uygulanmakta olduğunu varsaydığımız gibi, *aritmetik* bildirimlere çevirme işlemi nedeniyle yukarıdaki bildirim, *w* doğal sayısı ile ilgili bir aritmetik bildirimidir (köşeli parantez içerisinde yer alan bildirim, iki doğal sayı, *x* ve *w,* ile ilgili iyi-tanımlanmış bir aritmetik bildirimidir). Bildirimin aritmetiğe kodlanabileceğinin açıkça görüldüğü söylenemez ama kodlanabilir. Bu gibi bildirimlerin aritmetiğe kodlanabileceğini kanıtlamak, Gödel'in tezinin karmaşık kısmının kapsamında yer alan en önemli 'zor iş'tir. Daha önce belirttiğimiz gibi, kesinlikle *hangi* aritmetik bildirimi olduğu, numaralama sistemlerinin detaylarına ve daha çok, aksiyomların detaylı yapısına ve formel sistemimizin yapısına bağlı olacaktır. Bütün bunlar, teoremin karmaşık kısmına ait olduğu için ayrıntıları bizi burada

ilgilendirmiyor.

Tek değişkene bağlı tüm önergesel fonksiyonları numaraladık; bu nedenle, biraz önce yazmış olduğumuz fonksiyona bir sayı vermemiz gerekiyor. Bu sayı *'k'* olsun. Önergesel fonksiyonumuz, listede *k*'ci olarak yerini alır. Buna göre,

$\sim \exists\, x[\Pi_x$ kanıtlar $P_w(w)] = P_k(w)$

olacaktır. Şimdi, bu fonksiyonu özel bir *w* değeri; $w = k$ için incelersek, aşağıdaki önermeyi elde ederiz:

$\sim \exists\, x[\Pi_x$ kanıtlar $P_k(k)] = P_k(k)$

P_k (k) önermesi iyi tanımlanmış (doğru söz dizimine sahip) bir aritmetik bildirimidir. Formel sistemimizde bir kanıtı var mıdır? Olumsuz ifadesi $\sim P_k(k)$'nin bir kanıtı var mıdır? Her iki sorunun da yanıtı 'hayır' olmalıdır. Gödel'in yönteminin temelinde yatan *anlamı* inceleyerek, yanıtın 'hayır' olması gerektiğini anlayabiliriz. $P_k(k)$ sadece bir aritmetik önerme olsa da, eşitliğin sol tarafına yazdığımız: '$P_k(k)$ önermesinin, sistem içerisinde, kanıtı yoktur', ifadesini savunmak amacıyla inşa edilmiştir. Aksiyomlarımızı ve yöntem kurallarını dikkatle yerleştirdiysek ve numaralama işlemimizi doğru yaptıysak, $P_k(k)$ Önermesinin sistem içerisinde herhangi bir kanıtı olamaz. Çünkü, böyle bir kanıt olsaydı, $P_k(k)$ önermesinin kendi savunduğu bildirimin anlamı, yani 'kanıt yoktur' anlamı, yanlış olurdu ve bu nedenle $P_k(k)$, bir aritmetik önerme olarak yanlış olurdu. Formel sistemimiz, yanlış önermeleri kanıtlayacak kadar kötü inşa edilmiş olamaz! Bu nedenle gerçek, $P_k(k)$ önermesinin kanıtının olmamasıdır. Ama zaten $P_k(k)$ önermesinin bize anlatmaya çalıştığı da budur. Böylelikle, $P_k(k)$ önermesinin savunduğu doğru bir bildirim olmalıdır ve $P_k(k)$ bir aritmetik önerme olarak doğrudur. *Sistem içerisinde kanıtı bulunmayan doğru* bir önerme bulmuş olduk!

Olumsuz önerme $\sim P_k(k)$ hakkında nasıl bir yargıya varmamız gerekir? Biraz önce $P_k(k)$ doğru ise $\sim P_k(k)$ önermesinin yanlış olması gerektiğini kanıtlamıştık ve bizden, sistem içerisindeki yanlış önermeleri kanıtlamamız bekleniyor! Bu nedenle, ne $P_k(k)$ ne de $\sim P_k(k)$, formel sistemimiz çerçevesinde kanıtlanabilir. İşte Gödel teoremi budur.

Matematiksel Sezgi

Bu aşamada çok ilginç bir konunun ortaya çıktığı görülüyor. İnsanlar çoğu kez, Gödel teoreminin, biçimselleştirilmiş matematiksel uslamlamanın gerekli sınırlarını gösteren olumsuz bir tez olduğunu sanırlar. Ne kadar geniş kapsamlı olduğumuzu düşünürsek düşünelim yine de ağın gözlerinden kaçıp gidecek önermeler daima bulunacaktır. Fakat, $P_k(k)$ önermemiz için bu konuda kaygılanmaya gerek var mı? Yukarıdaki savunmamız sonucunda $P_k(k)$ önermesinin gerçekten doğru bildirim olduğunu kanıtladık bile! Sistem kapsamında formel olarak kanıtlanamamasına karşın $P_k(k)$'nin doğru olduğunu anlayabildik. Matematiksel formalizmi ne pahasına olursa olsun savunanlar gerçekten kaygılanmaklar, çünkü uyguladığımız uslamlamayla, formalistlerin 'doğru' hakkındaki görüşlerinin eksik olduğunu gösterdik. Aritmetik için *hangi* (tutarlı) formel sistem kullanılırsa kullanılsın, doğru olduklarını görebildiğimiz fakat formalistin önerdiği yöntemle, doğruluk değeri doğru ile tanımlanamayan bildirimler vardır. Katı bir formalistin böyle bir değere yaklaşmaya çalışması durumunda izleyeceği

en iyi yol belki de, doğruluk kavramından hiç bahsetmemek, yalnızca belirli bir formel sistem çerçevesinde *kanıtlanabilirlikten* söz etmektir. Ancak, böyle bir yöntem çok sınırlayıcı görünüyor. Böyle bir yöntemle Gödel teoreminin ana hatlarını bile çizemezsiniz, çünkü bu teoremin ana kısımları, gerçekte neyin doğru neyin yanlış olduğu hakkında yargıya varmak için uslamlama yönteminden yararlanır.[2] Bazı formalistler, aritmetik önermeleri olarak son derece karmaşık ve sıkıcı oldukları gerekçesiyle $P_k(k)$ gibi bildirimlere aldırmadıklarını iddia ederek daha 'pragmatik' bir yaklaşımı benimserler. Bu gibi formalistin savı şöyledir:

'Evet, $P_k(k)$ gibi acaip bir bildirim vardır ama onunla ilgili kanıtlanabilirlik veya doğruluk görüşüm sizin içgüdüsel görüşünüzle bağdaşmaz; bu gibi bildirimlere ciddi matematikte rastlanmaz (en azından benim ilgilendiğim matematikte rastlanmaz) çünkü bu gibi bildirimler tuhaf bir şekilde karmaşık ve matematik olarak yapaydır. '

Gerçekten de, matematik bildirimler olarak tam yazıldıklarında $P_k(k)$ gibi önermeler son derece karmaşık ve tuhaf görünümlüdür. Ancak, son yıllarda, Gödel-tipi önermelere eşdeğer, oldukça basit ve benimsenebilir matematiksel özelliklere sahip bildirimler ileri sürülmüştür.[3] Bunlar, aritmetiğin normal aksiyomlarıyla kanıtlanamazlar, fakat aksiyom sisteminin bizzat sahip olduğu "açıkça doğru olma" özelliğinin sonucudurlar.

Formalistin, 'matematiksel doğruya' profesyonelce ilgi duymaması, matematiğin felsefesi açısından bana çok garip geliyor.

Üstelik bu tutum, o kadar da pragmatik değil. Matematikçiler kendi usavarım yöntemlerini kullanırken savlarının, karmaşık bir formel sistemin aksiyomları ve yöntem kurallarıyla formüle edilip edilemiyeceğini sürekli kontrol etmek durumunda kalmak istemezler. Sadece savlarının, doğruyu saptamak için geçerli bir araç olup olmadığından emin olmak isterler. Gödel savı böyle bir geçerli araçtır ve bu nedenle bana öyle geliyor ki $P_k(k)$, önceden belirlenebilen aksiyomların ve yöntem kurallarının uygulanmasıyla, yani daha alışılmış bir yöntemle, elde edilebilen bir matematiksel doğru kadar iyi bir matematiksel doğrudur.

Bu arada bir yöntem kendiliğinden ortaya çıkıyor: Diyelim $P_k(k)$ -burada şimdilik G_0 ile göstereceğim- gerçekten mükemmel geçerliliği olan bir önermedir; bu nedenle, onu ek bir aksiyom olarak sistemimize ekleyebiliriz. Kuşkusuz, bu şekilde değiştirilen yeni sistemimizin *kendine ait* Gödel önermesi, örneğin G_1 olacaktır ve bu önerme de yine sayılarla ilgili mükemmelen geçerli bir bildirimdir. G_1 önermesini de aynı şekilde sistemimize katalım. Bu durumda, kendine ait G_2 Gödel önermesine sahip (yine mükemmelen geçerli) değiştirilmiş bir sistemimiz daha olacaktır; bunu da sisteme ekleyerek G_3 Gödel önermesini elde edelim ve aynı işlemi sonsuz kere tekrarlayalım. Elde edeceğimiz G_0, G_1, G_2, G_3.....ek aksiyomlarından oluşan listenin *tümünü* uygularsak ne olur? Şimdi elimizde sınırsız (sonsuz) bir aksiyomlar sistemi bulunduğuna göre, Gödel yönteminin uygulanabilir olduğu artık pek de açık değildir. Ancak, Gödel önermelerinin bu şekilde sürekli birleştirilmesi mükemmel bir sistematik süreç olup, aksiyomlardan ve yöntemin kurallarından oluşan normal bir sonlu mantık sistemi şeklinde yeniden ifade edilebilir. Bu sistemin de kendine ait Gödel önermesi, diyelim G_w, varolacağı ve bu önerme de sisteme eklenebileceği için Gödel önermesi G_{w+1} olarak ifade edilebilir. Bunu tekrarlarsak G_w, $G_{w+i;}$ G_{w+2} G_{w+3}, ... gibi bir önermeler listesi elde ederiz. Bu önermelerin hepsi, doğal sayılarla ilgili olarak mükemmel tanımlanmış bildirimlerdir ve hepsi de formel sistemimize eklenebilir. Bu da yine mükemmel bir sistematik süreç olup bütün bildirimleri

kapsayacak kadar geniş kapsamlıdır; fakat, kendine ait Gödel önermesine, G_{w+w}, veya G_{w2} önermesine, sahip olacağı için, bizi G_{w3} Önermesine götürecek işlemi başlatarak G_{w2}, G_{w2+1}; G_{w2+2},... vb. oluşan yeni bir sonsuz fakat sistematik aksiyomlar listesi üretebiliriz. Aynı işlemi yineliyerek G_{w4} ve sonra G_{w5}, vb. önermeler üretmemiz olasıdır. Şimdi *bu* yöntem tümüyle sistematiktir ve kendine ait G_{w2} Gödel önermesine sahiptir.

Bunun sonu var mı? Bir bakıma, hayır; fakat bizi, burada ayrıntılara girmeyeceğimiz kadar zor matematiksel görüşlere yönlendiriyor. Söz konusu yöntem, Alan Turing tarafından 1939'da sunulan bir yazıda tartışılmıştı.[4] Aritmetikte herhangi bir doğru (fakat evrensel olarak nicelleştirilmiş) önermenin, buna benzer şekilde, yinelemeye dayalı 'Gödelleştirme' süreciyle elde edilebilmesi ilginçtir! (Bkz. Feferman **1988).** Ancak, süreç, bir önermenin doğru veya yanlış olduğuna nasıl *karar vereceğiz* sorusunu da ön plana çıkıyor. Kritik konu, her aşamada, Gödel önermelerinin oluşturduğu sonsuz ailenin ek bir aksiyom (veya sınırlı sayıda aksiyomlar) üretecek şekilde nasıl kodlanması gerektiğini yanıtlamaktır. Bu amaçla sonsuz ailemizin, herhangi bir algoritmik yöntemle sistematik hale getirilmesi gerekir. Söz konusu sistemleştirmenin, kendinden bekleneni, doğru olarak gerçekleştirmesini sağlamak için, daha önce $P_k(k)$*'in* doğru bir önerme olduğunu kanıtlarken yaptığımız gibi, sistemin dışında bir kaynaktan, sezgilerden yararlanacağız. Ancak, sezgiler sistemleştirilemez ve bu nedenle, gerçekten, *herhangi bir* algoritmik işlemin dışında kalmaktadırlar!

$P_k(k)$ Gödel önermesinin, aritmetikte gerçekten doğru bir bildirim olduğunu kanıtlamaya yardımcı olan sezgi yeteneğimiz, mantıkçıların *düşünce ilkesi* adını verdikleri genel yöntemin bir örneğidir; böylece mantıkçı aksiyom sisteminin ve yöntemin kurallarının anlamı üzerinde 'düşünceye dalarak' ve kendini, bu sistem ve yöntemlerin matematiksel doğruya ulaşmak için gerçekten geçerli araçlar olduklarına inandırarak, aksiyomlarla ve kurallarla ulaşılamayacak doğru bildirimleri söz konusu sezgiyle kodlayabilir. Yukarıda ana hatlarıyla açıklandığı gibi, $P_k(k)$ önermesinin doğruluğuna bu ilkeyle ulaşılmıştır. İlk Gödel savı ile ilgili bir başka düşünce ilkesi (yukarıda açıklanmamış olsa da), matematiksel doğrulara ulaşmak için geçerli bir araç olarak kabul ettiğimiz bir aksiyom sisteminin aslında *tutarlı* olduğu olgusundan yeni matematiksel doğrular üretilmesi esasına dayanır. Sezginin ilkeleri, çoğu kez, sonsuz kümelerle ilgili uslamlamayı içerirler ve bu nedenle, Russell'ınkine benzer paradokslara sürükleyebilecek türde bir sava çok fazla yaklaşmaktan sakınmak için bunları uygularken daima dikkatli olunmalıdır. Söz konusu ilkeler formalist uslamlama yönteminin tam anti-tezini oluşturur. Dikkatli davranıldığında, daha önce varlıklarının farkında olunmayan yeni matematiksel sezgilere ulaşmak amacıyla, herhangi bir formel sistemin katı sınırlarının dışına sıçramak mümkündür. Matematik literatüründe bunun güzel örneklerine rastlanabilir. Matematikçileri, doğru ile ilgili yargılarına yönlendiren ussal işlemler yalnız formel sistemlerin yöntemlerine dayalı değildir. Gödel önermesi $P_k(k)$'nin doğruluğunu anlamamız aksiyomlar aracılığıyla olmamıştır. Bir düşünce ilkesinde 'anlama', yalnız algoritmik işlemlerle başarılamayacak bir matematik işlemini gerektirir. Böyle bir matematik işleminin, matematiksel bir formel sistem çerçevesinde nasıl kodlanabileceği konusu X. Bölüm'de incelenecektir.

$P_k(k)$ önermesinin 'kanıtlanamazlığını' fakat aynı zamanda doğruluğunu kanıtlayan tez ile Russell'ın ikilemi ile ilgili tez arasında belli bir benzerliğin varlığı okurun dikkatini çekmiş olabilir. Durma problemini çözecek bir makinenin varolmadığını savunan Turing teziyle de aralarında benzerlik vardır. Bu benzerlikler rastlantısal değildir. Üçü arasında güçlü tarihsel bağlar mevcuttur. Turing, savını Gödel'in eserinden esinlenerek bulmuştur. Gödel, Russell'ın ikilemini pekâlâ

biliyordu ve mantığı aşırı zorlayan bu tür ikilemli uslamlamayı doğru bir matematik teoremi haline getirebildi (Tüm bu savlar, bir önceki bölümde değinilen Cantor'un "köşegen çizik" kavramından kaynaklanmaktadır).

Bizi Russell'ın ikilemine götüren uslamlamayı red ederken niçin Gödel'in ve Turing'in savlarını kabul edelim? Russell'ın ikilemi 'muazzam büyük' kümeleri içeren daha belirsiz uslamlamaya dayanırken Gödel ve Turing'in savları çok daha kesin tanımlanmış olağan matematiksel teoremlerdir. Ancak, aralarındaki farkın istenildiği kadar kesin çizgilerle belirlenmemiş olduğunu kabul etmek zorundayız. Kesin çizgilerle ayırım yapılması, formalizmin amacı olmuştur. Gödel, katı formalist görüşün tutarlı olmadığını göstermiş, fakat tamamen güvenilir bir alternatif görüş de ileri sürmemiştir. Bence konu henüz çözümlenmiş değildir. Russell'ın ikilemine götüren 'muazzam büyük' kümelere dayalı uslamlama gibi yöntemlerden kaçınmak için çağdaş matematiğin benimsediği yöntem her yönüyle tatmin edici değildir.[V] Ayrıca, bunları formalistik terimlerle -veya, bir başka deyişle, çelişkilerle karşılaşılmayacağı konusunda tam güvence vermeyen terimlerle- ifade etmek eğilimi hâlâ ağır basmaktadır.

Ne olursa olsun, bana öyle geliyor ki, Gödel'in teoreminin açıkça sergilediği sonuç, matematiksel doğruluğun herhangi bir formalist çerçeve içerisine sıkıştırılamıyacağıdır. Matematiksel doğruluk, salt formalizmin çok ötesinde bir şeydir. Godel'in teoremi olmaksızın da bunu anlamak olasıdır. Çünkü, bir formel sistem inşa etmeğe kalkışsak hangi aksiyomları veya hangi kuralları seçeceğimize nasıl karar verebiliriz? Hangi kuralları seçeceğimiz konusunda bize yol gösteren her zaman ve mutlaka, sistemin simgelerinin anlamı verildiğinde, bunlardan içgüdüsel olarak ne anladığımız olmalıdır. Hangi formel sistemin, "kendini kanıtlar" olma ve "anlamlılık" bakımından içgüdüsel olarak kabul edilebilir, hangisinin kabul edilemez olduğuna nasıl karar verebiliriz? Kuşkusuz, kendi içinde-tutarlılık fikri burada yeterli olamaz. Bu bağlamda 'sezilebilir' olmayan ve yanlış oldukları veya anlamsız oldukları gerekçesiyle red edeceğimiz aksiyomları veya yöntemsel kuralları içeren birçok kendi içinde -tutarlı sisteme sahip olabiliriz. Yine de gereksinim duyacağımız kavramlar, Gödel teoremi olmaksızın dahi, 'kendini-kanıtlayabilme' ve 'anlamlılık' kavramlarıdır.

Ancak, Gödel teoremi olmaksızın, sadece önce formel sistemi inşa etmek, sonra bunun doğruluğu saptamak için kullanılan matematiksel önermenin bir parçası olarak ondan kurtulmak amacıyla 'kendini-kanıtlama' ve 'anlamlılık' kavramlarını ilk ve son kez kullanabileceğimizi düşlemek mümkün olabilirdi. Daha sonra, formalist düşünceye göre, söz konusu 'bulanık' içgüdüsel kavramlar, uygun formel matematikçinin ön düşünce sisteminin bir parçası olarak rol alabilecek, fakat matematiksel doğruluğun fiilen sergilenmesinde hiçbir rol üstlenemiyecekti. Gödel teoremi bu görüşün, matematiğin temel felsefesinde yerinin olmadığım göstermektedir. Matematiksel doğruluk fikri, formalizm kavramının sınırlarının çok ötesine uzanır. Matematiksel doğruluk kavramında, mutlak ve 'Tanrı-vergisi' olan bir şey vardır. Son bölümün sonunda tartışıldığı gibi, matematiksel Platonizmin ilgi alanı budur. Herhangi bir formel sistemde bu kavram, geçici ve 'insan-yapısı' bir nitelik taşır. Formel sistemler, matematik üzerine tartışmalarda gerçekten çok değerli roller üstlenirlerse de, doğruluğun saptanması yönünde sadece kısmi (veya yaklaşık) bir rehber olabilirler. Gerçek matematiksel doğruluk salt insan yapısının ötesine geçer.

Platonizm mi Yoksa Sezgicilik mi?

Bu aşamaya kadar, matematik felsefesinin iki zıt ekolü ile ilgili görüşlerimi, formalist ekolden çok

Platonist ekole ağırlık koyarak, ortaya koydum. Bunu yaparken ayırımımı oldukça basitleştirilmiş tuttum. Oysa, değinilmesi gereken birçok ince ayrıntılar var. Örneğin, 'Platonizm' başlığı altında, matematiksel düşünce nesnelerinin herhangi bir türde gerçek 'Varlığa' sahip olup olmadıkları veya sadece matematiksel 'doğruluk' kavramının mutlak olup olmadığı tartışılabilir. Bu gibi ayırımları, konumuz çerçevesinde ele almamayı uygun gördüm. Kanımca, matematiksel doğruluğun mutlaklığı ile matematik kavramlarının Platonik varoluşu temelde aynı şeydir. Örneğin Mandelbrot kümesiyle ilişkilendirilebilecek 'Varoluş' kavramı, 'mutlak' doğasının bir Özelliğidir. Argand düzleminin bir noktasının Mandelbrot kümesine ait olup olmadığı, hangi matematikçi veya bilgisayar tarafından incelenirse incelensin, mutlak bir sorudur. Mandelbrot kümesinin 'matematikçiden bağımsız olma' özelliği ona Platonik bir varlık kazandırır. Ayrıca, Mandelbrot kümesinin ince ayrıntıları, bilgisayar kullanarak ulaşabileceğimiz sınırın çok ötesindedir. Bu aygıtlar, 'bilgisayardan bağımsız', çok daha derin ve kendine özgü bir yapıyla ilgili olarak ancak yaklaşık değerler verebilirler. Ancak, bu konuda kabul edilebilir ayırım yapabilmemizi sağlayan birçok yaklaşım bulunmasını takdir ediyorum. Burada, bu ayırımların üzerinde fazlaca durmamız gerekmiyor.

Gerçek bir Platonist olduğunu, iddia eden bir kimse, Platonizmini taşıyabileceği noktada görüş ayrılıklarıyla karşılaşmayı göze almalıdır. Bizzat Gödel koyu bir Platonist idi. Şu ana kadar ele aldığım matematik bildirimleri, ötekilerinin yanında 'ılımlı' kalan türden bildirimlerdir.[5] Özellikle kümeler teorisinde tartışmaya açık daha çok bildirim ortaya çıkabilir. Küme teorisinin bütün dalları incelendiğinde öylesine aşırı büyüklükte ve öylesine belirsiz inşa edilmiş kümelerle karşılaşılabilir ki, benim gibi oldukça kararlı bir Platonist bile, bunların varolmalarının, veya olmamalarının, gerçekten bir 'mutlak' konu oluşturduğu hakkında kuşku duymaya başlayabilir.[6] Bir aşamada kümeler, öylesine karmaşıklaşır ve kavramsal yönden kuşku uyandırıcı tanımlamalara dönüşebilirler ki, bunlarla ilgili matematik bildirimlerin doğruluğu veya yanlışlığı sorusu, Tanrı-vergisi niteliğinden çok bir 'görüş meselesi' niteliğini alır. Bir insanın, Gödel ile birlikte, Platon izinin uzun yolunu katetmeğe ve böylesine aşırı boyutlardaki kümelerle ilgili matematik bildirimlerinin doğruluğunun veya yanlışlığının daima bir mutlak veya 'Platonik' konu olup olmadığını araştırmaya hazır olması, veya bir noktada bundan vazgeçip sadece kümeler oldukça yapıcı ve normal boyutta olduğu zaman mutlak doğruluk veya yanlışlığı araştırması, tartışmamızın dışındadır. Bizim için, biraz önce değindiğim standartlara göre, önem taşıyan kümeler (sonlu veya sonsuz) komik denecek kadar küçük boyutludur! Bu nedenle, çeşitli Platonistik görüşler arasındaki ayrım bizi fazlasıyla ilgilendirmeyecek.

Ancak, *sezgicilik* (ya da *sonluculuk*) gibi başka matematik felsefeleri de vardır ki, bunlar zıt uçta yer alarak hangi sonsuz küme olursa olsun, bunların tamamlanmış varlığını kabul etmeyi red eder.[VI] Sezgicilik, 1924'te, Hollandalı matematikçi L.E.J. Brouwer tarafından, matematiksel uslamlama yönteminde sonsuz kümeler son derece serbest kullanıldığında karşılaşılabilecek (Russell'ın ikilemi gibi) ikilemlere alternatif bir yanıt -formalizminkinden farklı- olarak ileri sürülmüştür. Bu görüşün kökleri, Platon'un öğrencisi olan fakat Platon'un matematiksel nesnelerin mutlak varlığı ve sonsuz kümelerin kabul edilebilirliği hakkındaki görüşlerini red eden Aristoteles'e kadar uzanır. Söz konusu görüşe göre kümeler (sonsuz veya değil), özlerinde bir 'varlığa' sahip değillerdir; sadece, sistem kapsamında elemanlarını tayin eden kurallar sayesinde ele alınabilirler.

Brouwer'in sezgiciliğinin karakteristik bir özelliği, 'üçüncünün olmazlığı yasası'nı kabul etmemesidir. Bu yasa, bir bildirimin olumsuz şeklinin red edilmesinin, bu bildirimin önermesi ile eşdeğerde olduğunu savunur. (Simgelerle ifade edildiğinde: ~ (~ P) <=> P.) Herhalde Aristoteles de, mantıksal yönden bu kadar 'açık seçik' ifade edilen bir önermenin red edilmesinden mutsuzluk

duyardı! Basit 'sağ duyu' terimleriyle açıklandığında söz konusu yasa, 'kendini kanıtlayan doğruluk' olarak yorumlanabilir: Bir şeyin doğru olmadığı yanlışsa, bu şey kuşkusuz doğrudur! (Bu yasa, *reductio ad absurdum* yönteminin temelini oluşturur.) Fakat, sezgiciler, bu yasayı yadsıyabileceklerinin farkına vardılar. Çünkü temelde 'varoluş' kavramına farklı bir yaklaşım getiriyorlar ve bir matematiksel nesnenin gerçekten *varolduğunu* kabul etmeden önce onun kesin bir (ussal) yapısının sunulmasını gerekli görüyorlardı. Buna göre sezgici için 'varoluş', 'yapıcı varoluş' anlamındadır. *Reductio ad absurdum* yöntemine uygun bir önermede bir varsayım ileriye sürülürken, sonuçlarının bir çelişkiye götürebileceği, bu çelişkinin söz konusu varsayımın yanlış olduğuna dair istenilen kanıtı sağlayacağı imâ edilir. Varsayım, belirli bazı özelliklere sahip matematiksel bir nesnenin varolmadığını öneren bir bildirime dönüşebilir. Bu durumda bir çelişki görülürse, basit matematikte çıkarılacak anlam, öngörülen varlığın gerçekten varolduğudur. Fakat böyle bir sav, kendi başına, böyle bir varlığı inşa edecek bir yöntem sağlayamaz. Bir sezgiciye göre bu çeşit bir varoluş, hiçbir zaman varoluş olamaz; ve işte bu bağlamda, üçüncünün olmazlığı yasasını ve reductio ad absürdüm yöntemini kabullenmeyi red eder. Gerçekten de Brouwer bu çeşit bir yapıcı-olmayan Varoluş'tan hiç hoşlanmamıştır.[7] Gerçek bir yapılandırma olmaksızın böyle bir kavram anlamsızdır diyerek tezini savunmuştur. Brouwer mantığında, bir nesnenin varolmayışının yanlışlığından bu nesnenin gerçekten varolduğu anlamı çıkarılamaz!

Kanımca, matematiksel varoluşta yapıcılık aramanın takdir edilecek bir yanı olmakla birlikte Brouwer'in sezgicilik görüşü biraz fazla aşırıdır. Brouwer fikirlerini ilk kez 1924'de, yani Church ve Turing'in eserlerinden on yıldan fazla bir süre önce, açıklamıştır. Şimdi artık, yapıcılık kavramı -Turing'in hesaplanabilirlik fikri bağlamında-, matematiksel felsefenin *alışıldık* çerçevesinde incelenebildiğine göre, Brouwer'in bizi yönlendirmek istediği aşırılıklara gitmemiz gerekmiyor. Yapıcılığı, matematiksel varoluş konusundan ayrı bir konu olarak ele alabiliriz. Sezgiciliğin izinde gidersek, matematiğin kapsadığı çok güçlü önermeleri kullanmaktan kendimizi alıkoymamız gerekir ki konu böylece tıkanacak ve kısırlaşacaklar.

Sezgici görüşün insanı içine sürükleyebileceği çeşitli zorlukları ve görünüşteki tuhaflıkları ayrıntılarıyla anlatmak istemiyorum; fakat, karşılaşılabilecek sorunlardan bir kaçına değinmek yararlı olabilir. Brouwer'in sıkça verdiği bir örnek, π'nin ondalık açılımıyla ilgilidir:

3.141592653589793...

Bu ondalık açılımın herhangi bir yerinde yirmi adet birbirini izleyen yedi rakamı

Π= 3.141592653589793... 77777777777777777777...;

var mı yoksa yok mu? Basit matematiksel ifadeyle, şu anda bütün söyleyebileceğimiz ya 'var' ya da 'yok' demektir -ancak bunlardan hangisinin yanıt olduğunu bilmiyoruz! Bu, yeterince zararsız bir bildirim gibi görünebilir. Ancak, sezgiciler, π'nin ondalık açılımının bir yerinde birbirini izleyen yirmi adet yedi rakamı vardır, aksi halde yoktur' ifadesinin geçerliliğini- böyle bir açılımın bulunduğu veya bulunmadığı (kendilerince kabul edilebilir yapıcı bir tarzda) saptanmadıkça veya saptanıncaya kadar -inkâr edeceklerdir! π'nin ondalık açılımının bir yerinde birbirini izleyen yirmi adet yedinin dizilimini göstermek için doğrudan hesaplama yeterli olacakken, böyle bir dizilimin gerçekte bulunmadığını göstermek için bir matematik teoremine ihtiyaç vardır. Hiçbir bilgisayar, bugüne değin, π'nin hesabında, böyle bir dizilimin varolduğunu saptayacak kadar gelişmemiştir. Böyle bir dizilimin varlığını olasılık çerçevesinde ümit edebiliriz ama bir bilgisayarın diyelim saniyede 10^{10} işlem hızında basamak ürettiğini varsaysak bile, açılımı bulmak için yüz ilâ bin yıl arasında bir süre gerekebilir! Bence, böyle bir açılımın varlığının, doğrudan hesaplanmaktan çok, bir

gün, matematiksel işlemle saptanması olasılığı daha fazla görünüyor (bu belki çok daha güçlü ve ilginç bir teoremin yan sonucu olur) -ama sezgiciler tarafından onaylanmayacak bir tarzda saptanacaktır!

Bu özel sorun matematiğin gerçek ilgi alanına girmez. Açıklaması kolay olduğu için yalnızca bir örnek olarak verilmiştir. Brouwer, kendi aşırı sezgiciliğinin çerçevesinde, π'nin ondalık açılımının bir yerinde peş peşe gelen yirmi adet yedinin varlığının şu an için ne doğru ne de yanlış olduğunu savunacaktır. Gelecekte bir gün bu veya şu şekilde doğru yanıt, hesapla veya (sezgisel) matematiksel kanıtla, bulunduğu takdirde önerme, duruma göre 'doğru' veya 'yanlış' olacaktır. Buna benzer bir diğer örnek 'Fermat'ın son teoremidir. Brouwer'in aşırı sezgici görüşüne göre, şimdilik, bu teorem de ne doğru ne yanlıştır, fakat gelecekte bir tarihte doğru da olabilir yanlış da.[VII] Bence, matematiksel doğruluğun bu çeşit öznelliği ve zamana-bağımlılığı hiç hoşlanılmayacak bir niteliktir. Bir matematik sonucunun resmen kanıtlanıp kanıtlanmadığı veya ne zaman kanıtlandığı gerçekten öznel bir konudur. Matematiksel doğruluk bu çeşit topluma bağlı kriterlere dayandırılmamalıdır. Zamana göre değişen bir matematiksel doğruluk kavramına sahip olmak, fiziksel dünyayı tanımlamak için insanın güvenle kullanabileceğini umduğu matematik yönünden, en iyimser bir ifadeyle, beceriksizlik ve olumsuzluktur. Sezgicilik tezini savunanların hepsi, Brouwer kadar aşırı bir tutum takınmayacaktır. Ama nasıl olursa olsun, yapılandırmacılığın amaçlarına sempati duyanlar için bile sezgici bakış açısı, açıkça görüldüğü gibi pek bir hantaldır. Sadece uygulanabilecek matematiksel uslamlama yönünden çok sınırlayıcı olması gibi tek olumsuz özelliği nedeni için bile günümüzün matematikçilerinden pek azı, sezgiciliği ciddiye alabilir.

Günümüz matematik felsefesinin üç ana akımını kısaca tanımladım: Formalizm, Platonizm ve Sezgicilik. Matematiksel doğruluğun mutlak, dışsal ve ebedi olduğunu ve insan-yapısı kriterlere dayanmadığını, matematik nesnelerinin kendilerine özgü ve zamanla sınırlı olmayan bir varlığa sahip olduklarını, ne insan toplumuna ne -de belirli fiziksel nesnelere bağlı olmadığını savunan Platonik görüşe yakınlık duyduğumu gizlemiyorum. Platonizm ile ilgili görüşlerimi bu kısımda, bir önceki kısımda ve III. Bölüm'ün sonunda açıkladım. Umarım okuyucum bu konuda benimle beraber yola çıkmaya hazırdır. Yol üzerinde karşılaşacaklarımız için bu önemlidir.

Turing'in Sonucundan Çıkan Gödel-Tipi Teoremler

Gödel teoremini sunarken birçok ayrıntıya, bu arada tarihsel olarak belki de en önemlisine, aksiyomların tutarlılığının 'karar verilemez' olduğuna ilişkin sava, değinmedim. Amacım, Hilbert ve çağdaşları için büyük önem taşıyan 'aksiyomun tutarlılığının -kanıtlanabilirliği' problemini vurgulamak değil, fakat Gödel'in belli bir önermesini, ele aldığımız biçimsel sistemin aksiyomlarını ve kurallarını kullanarak ne kanıtlanabilir ne de çürütülemez olduğunu gösteremeyeceğimizi, ancak işlemlerin anlamlarına 'sezgiyle' ulaşırsak, *doğru* bir önerme olduğunu açıkça *anlayacağımızı* vurgulamaktır!

Turing'in, Gödel'in yapıtını inceledikten sonra, kendi teoremini, durma probleminin çözülemezliğine dair teoremini, geliştirmiş olduğunu söylemiştim. Her iki teoremin birçok ortak noktası bulunmaktadır ve gerçekten de, Gödel'in sonucuna Turing'in yöntemiyle ulaşılabilir. Bunun nasıl gerçekleşeceğini inceleyelim ve bu arada Gödel teoreminin altında neler yattığı hakkında oldukça farklı bir bakış açısı edinelim.

Biçimsel matematiksel sistemin başlıca özelliği, verilen bir matematiksel Önerme ile ilgili simgeler dizisinin, sistem çerçevesinde, bir kanıt oluşturup oluşturmadığına karar vermek işleminin hesaplanabilir olmasını gerektirmesidir. Matematiksel kanıt düşüncesinin biçimleştirilmesinde yegane amaç, ne de olsa, geçerli bir uslamlama yöntemi için karar almak zorunda kalmamaktır. Önerilen bir kanıtın gerçekten bir kanıt olup olmadığını, tümüyle, mekanik ve önceden belirlenmiş bir yöntemle kontrol etmek mümkün olmalıdır; başka bir deyişle, kanıtları kontrol eden bir algoritma bulunmalıdır. Öte yandan, önerilen matematiksel bildirimlerin kanıtlarını (veya karşıt- kanıtlarını) *bulmak,* mutlaka algoritmanın görevidir demiyoruz.

Gerçekte, herhangi bir biçimsel sistemde ne zaman bir kanıt varsa, kanıtı bulmak için bir algoritma da daima vardır. Çünkü, sistemimizin, sınırlı bir simgeler 'alfabesiyle' ifâde edilebilen dilde formüle edildiğini varsaymalıyız. Daha önce yaptığımız gibi, simge dizilerini, her bir dizi uzunluklarına göre alfabetik sıralanmak şartıyla leksikografik olarak sıralıyalım. Böylece, doğru inşa edilen tüm kanıtların, leksikografik plâna uygun numaralanarak sıralanmasını sağlamış oluruz. Kanıtlar listemize sahip olmakla, formel sistemin tüm *teoremlerine* de sahip oluruz. Çünkü teoremler, doğru şekilde inşa edilmiş kanıtların son sıralarında yer alan önermelerdir. Sistemin *tüm* simgelerinden oluşan dizilerinin leksikografik listesini, bu diziler kanıt olarak anlam taşısın veya taşımasın, dikkate alabilir ve sonra, ilk diziyi, kanıt olup olmadığını denemek için kanıt-sınama algoritmamızla test ederiz ve kanıt değilse listeden atarız; sonra ikinci diziyi aynı şekilde test eder ve kanıt olmadığım anlarsak onu da listeden çıkarırız; sonra üçüncüsünü, dördüncüsünü vb. aynı şekilde test ederiz. Bu şekilde, bir kanıt varsa onu, sonuçta, listenin bir yerinde buluruz.

Hilbert, matematik sistemine, sistem kapsamında doğru formüle edilmiş herhangi bir matematik önermesinin doğruluğuna veya yanlışlığına formel bir kanıtla karar vermemizi sağlayacak kadar güçlü bir aksiyomlar ve kurallar sistemi bulmayı başarsaydı, bu gibi önermelerin doğruluğuna karar verilmesini sağlayacak genel bir algoritmik yönteme sahip olacaktık. Bu niçin böyle? Çünkü, yukarıda genel hatlarıyla çizilen yöntemle aradığımız önermeye listenin son satırında rastgelirsek, önermeyi *ispatladık* demektir. Ama bunun yerine, önermemizin *olumsuz* şekliyle karşılaşırsak, önermemizi *çürütmüş* oluruz. Hilbert'in programı eksiksiz olsaydı, bu sonuçlardan birine veya diğerine sonuçta mutlaka ulaşırdık (ve, tutarlı olması durumunda iki sonuç birlikte asla meydana gelmezdi). Böylece, mekanik yöntemimiz herhangi bir aşamada daima son bulacak ve biz de sistemin tüm önermelerinin doğruluğu veya yanlışlığı hakkında yargıya varmamızı sağlayan evrensel bir algoritmaya sahip olacaktık. Bu durumda, matematik önermeler hakkında genel bir algoritma bulunmadığına dair Turing'in vardığı sonucun aksi kanıtlanmış olacaktı. (Bkz. Bölüm II) Sonuç olarak Gödel'in, Hilbert-tipi hiçbir programın, tartışmakta olduğumuz bağlamda eksiksiz bir program olamayacağı şeklindeki görüşünü fiilen kanıtlamış olurduk.

Aslında Gödel'in teoremi bundan daha özel bir amaçla inşa edilmiştir. Çünkü Gödel'in ilgilendiği biçimsel sistemden, genel matematik önermeler için değil yalnız aritmetik önermeler için yeterli olması beklenmekteydi. Turing makinelerinin gerekli tüm işlemlerinin aritmetikten yararlanarak uygulanması olası mıdır? Başka bir deyişle, doğal sayıların tüm *hesaplanabilir* aksiyomları (yani, Turing makinesinin işlemlerinin sonucu tekrarlayan, veya algoritmik fonksiyonlar) basit aritmetik terimleriyle ifade edilebilir mi? Gerçekte bunu yapabiliriz ama tam olarak değil. Standart aritmetik ve mantık kurallarına ($\exists$ ve $\forall$ dahil) göre bir işlem daha yapmalıyız. Bu işlem, sadece

'K(x)'i doğrulayan en küçük doğal sayı x'i'

seçer. Burada *K( )* aritmetik işlemlerle hesaplanan herhangi bir önergesel fonksiyon olup, karşılığında böyle bir sayının bulunduğu varsayılır; yani $\exists x[K(x)]$ doğrudur (Böyle bir sayı

bulunmasaydı işlemimiz, varolmayan ve bulunması istenen x sayısını bulmak çabası içerisinde 'sonsuza kadar süregiderdi')[VIII]. Ne olursa olsun, Turing'in sonucuna dayalı yukarıdaki sav, Hilbert'in, biçimsel sistem çerçevesinde tüm matematik dallarını hesaplara indirgeyen programının savunulamayacağını gösteriyor.

Yöntem, bu şekliyle, doğru fakat sistem içerisinde kanıtlanamayan bir Gödel önermesine $(P_k(k)$ gibi) sahip olduğumuzu hemen göstermez. Ancak, II. Bölüm'deki 'bir algoritmayı nasıl altedebiliriz' tartışmasını hatırlarsanız, buna çok benzer bir şeyi burada yapabileceğimizi göreceksiniz. Hatırlayacağınız gibi, bir Turing makinesinin işleminin durup durmayacağına karar vermek için herhangi bir algoritma verildiğinde, çalışacağını bizim anlayacağımız fakat algoritmanın anlayamıyacağı bir Turing makinesi işlemini üretebiliriz (Algoritmanın, bir Turing makinesi işleminin ne zaman duracağı hakkında bize doğru bilgi vermesi gerektiği konusunda ısrar ettiğimizi, ama bazen algoritmanın kendisi sonsuza kadar işlediği için, Turing makinesinin ne zaman duracağını bize bildirmediğini anımsayınız). Bu nedenle, Gödel'in teoreminde olduğu gibi, verilen algoritmik işlemin başaramadığını gerçekleştiren bir önermeye sahip olarak, sezgiden yararlanarak, genelde neyin *doğru* olduğunu (Turing makinesinin işleminin durmayacağını) görebiliriz.

Tekrarlı Sayılabilir Kümeler

Turing'in ve Gödel'in. sonuçlarının temel öğelerini küme teorisi dilinde, grafik biçimde, tanımlamanın bir yolu vardır, Böylece esas konuların ön plana çıkabilmesi için, belirli simge sistemleriyle veya formel sistemlerle zorunlu tanımlamalar yapmaktan kurtulabiliriz. (4, 5, 8), {0, 57, 100 003}, {6}, {0}, {1, 2, 3, 4, 9999}, {0, 1, 2, 3, 4, ...}, {0, 2, 4, 6, 8, ...}, hatta tam küme N = {0, 1, 2, 3, 4, ...} veya boş küme ∅ = {} gibi küme gruplarını inceleyebilmek için yalnız 0, 1, 2, 3, 4, ..., gibi (sonlu veya sonsuz) *doğal sayı* kümelerini inceleyeceğiz. Sadece hesaplanabilir sorularla, örneğin: 'Hangi tür doğal sayı kümeleri algoritmalar tarafından üretilebilir, hangileri üretilemez?' gibi sorularla ilgileneceğiz.

Bunun gibi konuları ele almak için, istersek, her bir n doğal sayısının belirli biçimsel bir sistemde belirli bir simgeler dizisini temsil ettiğini düşünebiliriz. Bu, sistemdeki önermelerin ('söz dizimi' doğru yapılmış) leksikografik sıralamasına göre simgelerin n'inci sırası, yani Q_n olacaktır. Her doğal sayı bir önermeyi temsil eder. Formal sistemin tüm önermelerinin kümesi, kümesinin tümü tarafından temsil edilecek, ve örneğin, biçimsel sistemin teoremleri, doğal sayıların daha küçük bir alt kümesini, örneğin P kümesini, oluşturacaktır. Ancak, önermelerle ilgili herhangi bir belirli numaralama sisteminin detayları önemli değildir. Doğal sayılarla önermeler arasında bağlantı kurmak için ihtiyacımız olan tek şey, kendisini temsil eden n doğal sayısından (uygun bir simgeler sisteminde yazılmış) Q_n önermesini elde etmek için bilinen bir algoritma, ve Q_n'den n'yi elde etmek için bir başka algoritmadır. Böyle iki algoritma verildiğinde, özel bir formel sistemin önermelerinin kümesiyle, doğal sayılar kümesi N'i özdeşleştirebiliriz.

Tüm Turing makinelerinin tüm işlemlerini kapsayacak kadar geniş kapsamlı, tutarlı -ve aynı zamanda, aksiyomlarının ve yöntem kurallarının 'doğru olduğunu kendiliğinden kanıtlar'- şeklinde nitelenebilecek kadar 'duyarlı' olduğu bir biçimsel sistem seçelim. Artık, biçimsel sistemin Q_0, Q_1,Q_2, Q_3,... gibi önermelerinden bazıları, sistem içerisinde gerçekten kanıtlara sahiptir. Bu 'kanıtlanabilir' önermelerin numaraları, N 'de bir altküme, bir P 'teoremler' kümesini

oluşturacaklardır. Belirli bir biçimsel sistemde önermeleri kanıtlarıyla birlikte birbiri arkasına üreten bir algoritma bulunduğunu daha önce görmüştük (n'den algoritmik olarak '*n*'inci kanıt' Π_n'in elde edildiğini daha önce açıklamıştık. Bütün yapmamız gereken, sistem kapsamında kanıtlanabilir n'inci önermeyi, yani n'inci teoremi bulmak için n'inci kanıtın son dizesine bakmaktır). Öyleyse, P'nin elemanlarını peşpeşe üreten (tekrarlar bulunması fark etmez) bir algoritma var elimizde.

Bir algoritma kullanılarak üretilebilen P gibi bir küme, *tekrarlı sayılabilir küme* adıyla anılır. Sistem kapsamında kanıtlanamaz olan önermelerin, yani tersleri kanıtlanabilir önermelerin, (yolumuza devam ederken terslerini almak suretiyle kanıtlayarak sayabildiğimiz için aynı zamanda tekrarlı sayılabilir küme oluşturduklarını unutmayalım) tekrarlı sayılabilir birçok alt-kümesi vardır, ve bunları tanımlamak için formel sistemimize baş vurmak gerekmez. Tekrarlı sayılabilir kümelerin basit örnekleri, çift sayılar kümesi

{0, 2, 4, 6, 8, ...},

kareler kümesi

{0, 1, 2, 9, 16,...},

ve asal sayılar kümesi

{2, 3, 5, 7, 11, ...}'dir.

Açıkça görüldüğü gibi bu kümelerden her birini bir algoritma vasıtasıyla üretebilmekteyiz. Yukarıda verilen üç örnekten her birinde kümenin tümleyeninin, yani küme içinde yer almayan doğal sayılar kümesinin de tekrarlı sayılabilir olduğu görülecektir. Söz konusu üç örnekte, tümleyen kümeler, sırasıyla, şöyledir:

(1, 3, 5, 7, 9, ...};

(2, 3, 5, 6, 7, 8, 10, ...}; ve

{0, 1, 4, 6, 8, 9,10, 12, ...}.

Bu tümleyen kümeler için de bir algoritma üretmek kolay olacaktır. Gerçekten de, herhangi bir *n* doğal sayısının çift olup olmadığına, kare olup olmadığına veya asal sayı olup olmadığına algoritmik olarak karar verebiliriz. Böyle bir algoritmayı, hem asıl kümeyi hem de onu tümleyen kümeyi üretmek için kullanabiliriz. Hem kendisi hem de tümleyen kümesi tekrarlı sayılabilir bir kümeye *yinelenen küme* denir. Elbette ki yinelenen bir kümenin tümleyeni de bir yinelenen kümedir.

Peki, tekrarlı sayılabilir fakat yinelenemiyen kümeler var mıdır? Biraz duralım, bakalım bunun arkasından ne gelecek. Böyle bir kümenin elemanları bir algoritmayla üretilebileceğine göre, kümede yer aldığından kuşkulandığımız -ve, bir an için gerçekten kümede yer aldığını varsayalım- bir elemanın gerçekten kümenin elemanı olup olmadığına karar vermemizi sağlayacak bir araca sahip olacağız. Bize gerekli olan tek şey, algoritmamızın, incelemekte olduğumuz elemanı buluncaya kadar kümenin tüm elemanlarını taramasına izin vermektir. Fakat, varlığından kuşkulandığımız elemanın kümede gerçekten bulunmadığını varsayalım. Bu durumda algoritmamız işe yaramıyacaktır, çünkü bir karara varmaksızın taramasını sonsuza dek sürdürecektir. Bu nedenle, 'tümleyen' kümeyi üretmek için bir algoritmaya ihtiyacımız vardır. Her iki algoritmayla donanımlı olarak kendimizi yeterli hissetmemiz gerek. İki algoritmadan birini ya da diğerini kullanarak zanlıyı her durumda yakalayabiliriz. Ancak, bu mutluluk, yinelenen bir kümeyle ne yapacağımıza bağlıdır. Burada kümemizin yalnızca tekrarlı sayılabilir olduğu fakat yinelenen nitelikte olmadığı varsayılmıştır: Tümleyen kümeyi üretmek için önerdiğimiz algoritma ortada yoktur! Böylece tuhaf bir durumla karşı karşıyayız. Kümedeki bir elemanın gerçekten kümede olup olmadığına algoritma yardımıyla karar

vereceğiz, ama gerçekten kümede olup olmadığını yine algoritmayla garanti edemiyoruz! Böyle bir durumla gerçekten karşılaşılabilir mi? Tekrarlı sayılabilir fakat yinelenemeyen kümeler gerçekten var mıdır? Peki, P kümesinden ne haber? Yinelenen bir küme midir? Tekrarlı sayılabilir olduğunu biliyoruz. Öyleyse tümleyen kümenin de tekrarlı sayılabilir olup olmadığına karar vermemiz gerekiyor. Aslında, P tekrarlı sayılabilir küme değildir! Bunu nasıl kanıtlayabiliriz? Pekâlâ, Turing makinesinin işlemlerinin, biçimsel sistemimizin işlemleri arasında yer aldığının varsayıldığını hatırlayınız, n'inci Turing makinemizi T_n ile gösterdiğimize göre

'T_n (n) durur'

bildirimi bir önermedir. Biçimsel sistemimizde her bir *n* doğal sayısı için bu önermeyi S*(n)* ile gösterelim. *S(n)* önermesi n*'nin* bazı değerleri için doğru, bazı değerleri için yanlış olacaktır. Doğal sayılar 0, 1, 2, 3, ..., *n* tarafından taranırken, tüm *S(n)* kümesi, N'in alt-kümesi olan S ile temsil dilecektir. Şimdi, Turing'in temel sonucunu (II. Bölüm) hatırlayın: $T_n(n)$'in aslında durmadığı durumlarda *'$T_n(n)$* durmaz' önermesini doğrulayan bir algoritma yoktur. Bu sonuç, yanlış S(*n*)'ler kümesinin tekrarlı sayılabilir olmadığını göstermektedir.

S'in P'de yer alan kısmının, *doğru S(n)*'lerden oluştuğunu görüyoruz. Bu neden böyle? Kuşkusuz herhangi bir *S(n)* kanıtlanabilir ise, doğru olmalıdır (çünkü formel sistemimiz *anlamlı* seçilmiştir!) Bu nedenle, S'in P kümesinde yer alan kısmı, sadece *doğru S(n)* önermelerinden oluşmalıdır. Üstelik, P kümesinin kapsamı dışında hiçbir doğru *S(n)* önermesi yer alamaz, çünkü $T_n(n)$ durursa, gerçekten sistemin içerisinde yer alan bir kanıt elde edebiliriz.[IX]

Şimdi, diyelim ki, P kümesini tümleyen küme tekrarlı sayılabilmektedir. Bu durumda, bu tümleyen kümenin elemanlarını üretmek için bir algoritmaya ihtiyacımız olacaktır. Bu algoritmayı tarayarak rastladığımız her *S(n)* önermesini kaydedebiliriz. Kaydedeceğimiz tüm *S(n)* önermeleri yanlış önermeler olacağı için yöntemimiz aslında bize yanlış *S(n)* önermeler kümesinin tekrarlı sayımını verecektir. Fakat, yanlış *S(n)* önermelerinin tekrarlı sayılabilir olmadığını biraz önce belirtmiştik. Bu çelişkiye dayanarak, P'nin tümleyen kümesinin tekrarlı sayılabilir olmadığını söyleyebiliriz. Öyleyse, kanıtlamaya çalıştığımız gibi, *P kümesi yinelenen bir küme değildir.*

Bu özellikler, gerçekte, biçimsel sistemimizin tam olmayacağını gösterir. Başka bir deyişle, sistemin içerisinde ne kanıtlanabilir ne de çürütülebilir önermeler yer almalıdır. Çünkü, bu gibi 'karar verilemez' önermeler bulunmazsa, *P* kümesini tümleyen kümenin, *çürütülemez* önermeler kümesi olması gerekirdi (kanıtlanamayan herhangi bir şey çürütülemez). Fakat çürütülemez önermelerin, tekrarlı sayılabilir küme oluşturduklarını görmüştük. Öyleyse bu durumda *P kümesi yinelenen küme* olmalıdır. Ancak *P, yinelenen bir küme değildir.* İşte bu çelişki biçimsel sistemin tamamlanamıyacağını gösterir, ve Gödel'in teoreminin ana savıdır.

Peki, N'nin formel sistemimizin doğru önermelerini temsil eden alt-kümesi *T* ne olacak? *T* yinelenebilir bir küme midir? *T* tekrarlı sayılabilir mi? T'yi tümleyen küme tekrarlı sayılabilir mi? Gerçekte, tüm bu soruların yanıtı 'Hayır'dır. Bunu anlamanın bir yolu,

'$T_n(n)$ durur'

bildiriminin yanlış önermelerinin bir algoritmayla üretilemiyeceğini, daha önce yaptığımız gibi, göstermektir. Bu nedenle, yanlış önermeler, *bütün olarak* bir algoritma tarafından üretilemezler, çünkü böyle bir algoritma, tüm yanlış '$T_n(n)$ durur' önermelerini özellikle sayacaktır. Aynı şekilde, tüm doğru önermeler kümesi de bir algoritma tarafından üretilemez (çünkü böyle bir algoritmaya, ürettiği her bir önermenin tersini aldırtmak suretiyle, tüm yanlış önermeleri üretmesi basit bir şekilde

sağlanabilir). Doğru önermeler tekrarlı sayılabilir olmadığına (yanlış önermeler de öyle) göre, sistemin kapsamında kanıtlanabilir önermelere kıyasla çok daha karmaşık ve derinlemesine bir yapı oluştururlar. Bu durum yine Gödel'in teoremine bir örnektir: Matematiksel *doğruluk* kavramına, formel bir sav yoluyla ancak kısmen ulaşılabilir.

Ancak, tekrarlı sayılabilir kümeler oluşturan bazı basit doğru aritmetik önerme sınıfları vardır. Örneğin,

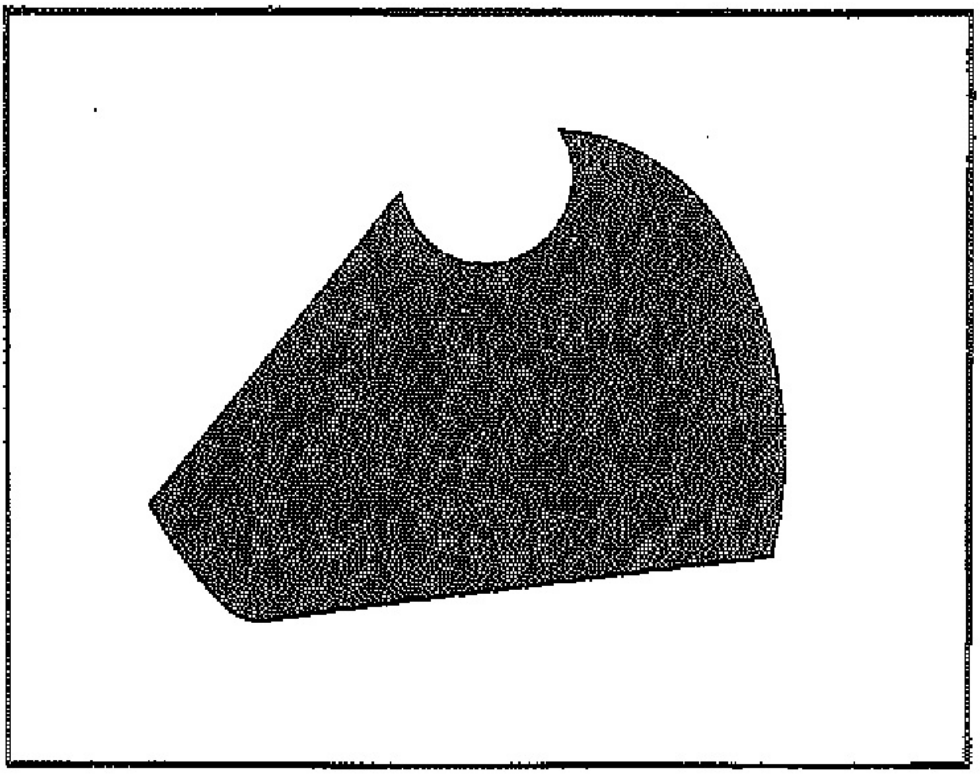
$\exists\, w, x\, z\, [f(w, x, ..., z) = 0]$

gibi doğru önermelerin -burada *f(),* toplama, çıkarma, çarpma ve karesini alma gibi basit aritmetik işlemlerden inşâ edilmiş herhangi bir fonksiyondur -tekrarlı sayılabilir küme oluşturduğunu (bu kümeye A diyorum) görmek zor değildir.[8] Bu çeşit bir önermenin örneği, doğru olup olmadığını bilmememize karşın, 'Fermat'ın son teoremi'nin olumsuz şeklidir. Önermenin *F*( ) değeri, aşağıdaki işlemle saptanabilir:

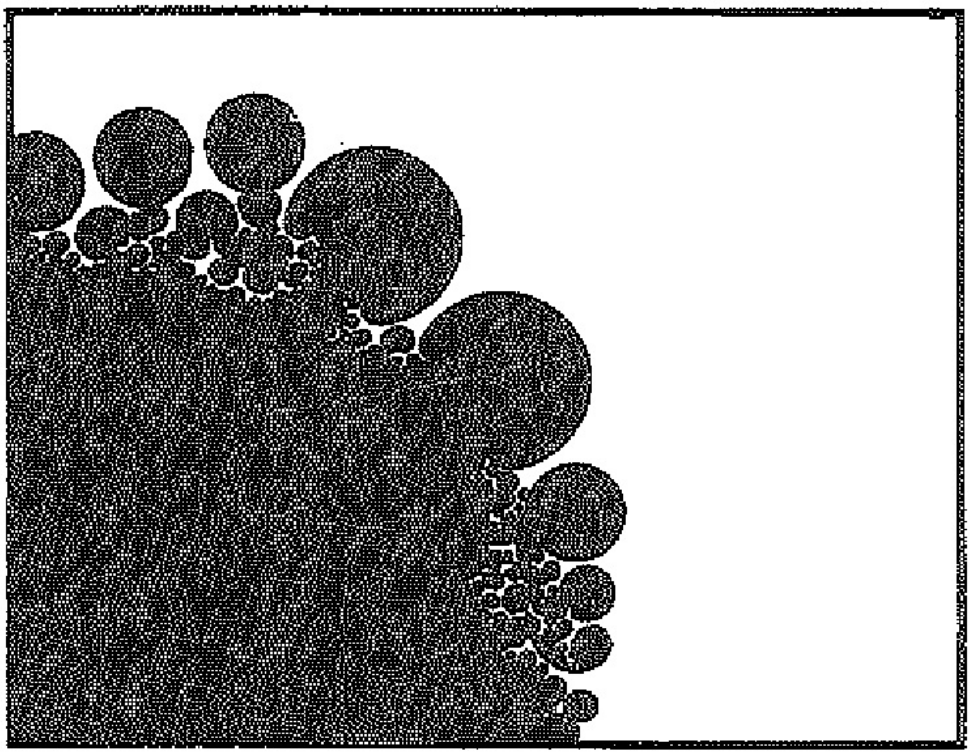
$f(w, x, y, z) = (x + 1)^{w+3} + (y + 1)^{w+3} - (z + 1)^{w+3}$

Ancak, A kümesi hiç de beklendiği gibi yinelenen bir küme değildir (İlk haliyle Gödel teoreminin bir sonucu olmasına karşın görülmesi pek de kolay olmayan bir olgudur), Böylece, 'Fermat'ın son teoremi'nin doğru veya yanlış olduğuna, ilke olarak bile, karar vermemizi sağlayacak bir algoritma bulamadık!

Şekil 4.1'de yinelenen bir kümeyi, verilen bir noktanın kümeye ait olup olmadığını doğrudan söylemenin mümkün olduğunu düşünebilmemiz için, basit sınırlara sahip bir alan şeklinde şematik olarak göstermeye çalıştım. Resimdeki her noktanın, doğal bir sayıyı temsil ettiği düşünülmelidir. Tümleyen küme de, basit görünüşlü bir alan olarak gösterilmektedir. Şekil 4.2'de, tekrarlı sayılabilir fakat yinelenmeyen bir kümeyi karmaşık sınırlara sahip bir küme olarak göstermeğe çalıştım; bu resimde, sınırın tekrarlı sayılabilir tarafındaki kümenin, diğer taraftaki kümeye kıyasla daha basit görünümlü olmasına çalışılmıştır. Şekiller son derece şematik tasarımlanmış olup, herhangi bir anlamda 'geometrik yönden doğru' olmaları amaçlanmamıştır. Özellikle, düz iki-boyutlu bir düzlem gibi gösterilmiş olmalarının bir önemi yoktur!



Şekil 4.1. Yinelenen bir kümenin son derece şematik tasarımı.



Şekil 4.2. Tekrarlı sayılabilir fakat yinelenemeyen bir kümenin (siyah alan) son derece şematik tasarımı. Amaç, beyaz alanının, hesaplanarak üretilebilen siyah alan çıkarıldıktan sonra sadece 'geriye kalan' olarak nitelenebileceğini göstermektir; bir noktanın gerçekte beyaz alanda yer aldığını iddia etmek hesaplanabilir bir konu değildir.

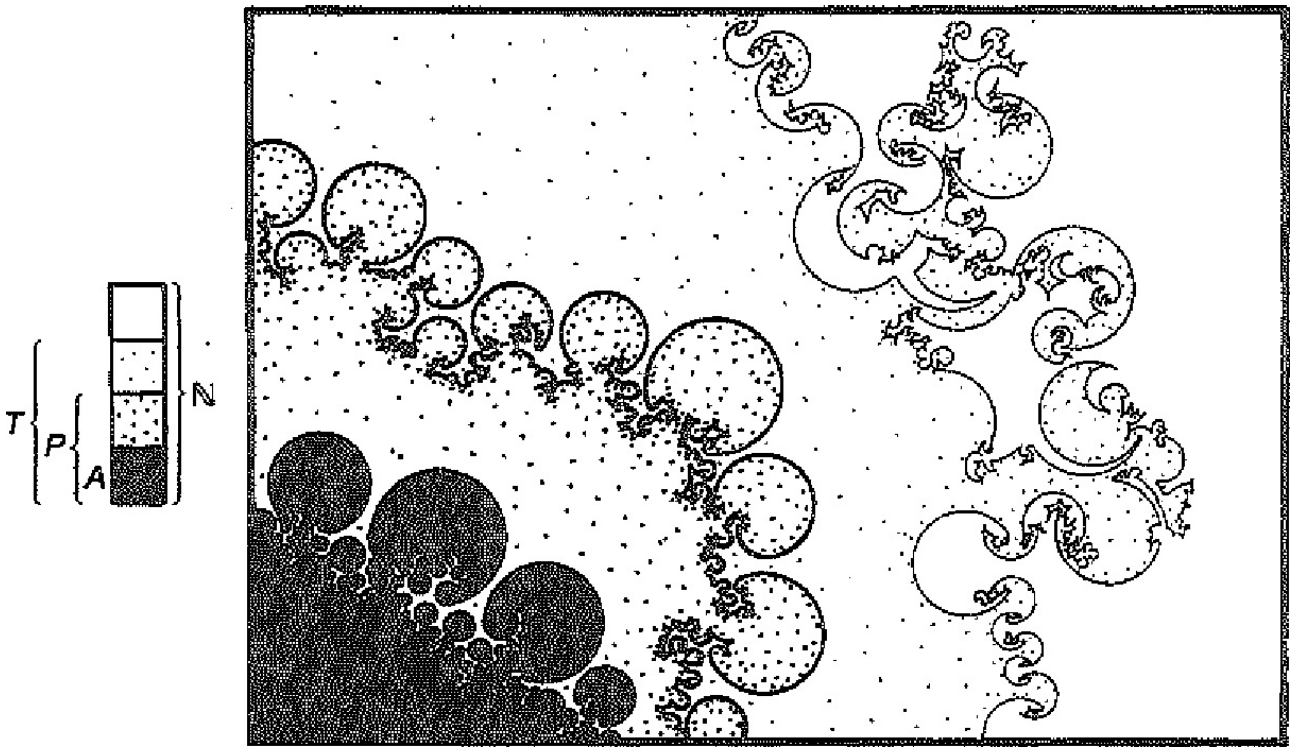
Şekil 4.3'de, *P, T ve A* alanlarının, N kümesi içerisinde nasıl yer aldıklarını şematik olarak

gösterdim.

Mandelbrot Kümesi Yinelenen Bir Küme midir?

Yinelenmeyen kümeler, gereğince karmaşık olma özelliğine sahip olmalıdırlar. Karmaşıklıkları, bir anlamda, tüm sistemleştirme girişimlerine karşı koyacak ölçüde olmalıdır. Aksi halde, böyle bir sistemleşme herhangi bir uygun algoritmik yöntemin elde edilmesiyle sonuçlanabilir. Yinelenmeyen bir küme için, bir elemanın (veya 'noktanın') kümeye ait olup olmadığına karar vermekle ilgili genel bir algoritmik yöntem yoktur. III. Bölüm'ün başında, Mandelbrot kümesi adıyla anılan, olağanüstü karmaşık görünüşlü bir kümeyle tanışmıştık. Tanımlanmasında kullanılan kurallar şaşılacak kadar basit olmasına karşın bu küme, son derece özenli bir yapının sonsuz çeşitlerini sergiler. Ölümlü gözlerimizin önüne serilen böyle bir yapı, yinelenmeyen bir küme örneği olabilir mi?

Ancak okuyucu, bu son derece karmaşık yapının, yüksek-hızlı modern elektronik bilgisayar teknolojisiyle görebilmemiz için inşâ edildiğine dikkat etmekte gecikmeyecektir.



Şekil 4.3. Çeşitli önermeler kümelerinin son derece şematik tasarımı. Sistemde kanıtlanabilir önermeler kümesi P, A gibi, tekrar tekrar sayılabilir olmasına karşın yinelenebilir değildir; doğru önermeler kümesi, T, tekrar tekrar sayılabilir bile değildir.

Elektronik bilgisayarlar, algoritmik işlemin nesnelleşmiş şekli değiller midir? Kuşkusuz öyledirler ama bu resimleri bilgisayarın ürettiğini unutmayalım. Argand düzlemindeki bir noktanın, yani bir c kompleks sayısının, Mandelbrot kümesine mi (siyah renkli) yoksa tümleyen kümeye mi (beyaz renkli) ait olduğunu saptamak için bilgisayar

$$z \rightarrow z^2 + c$$

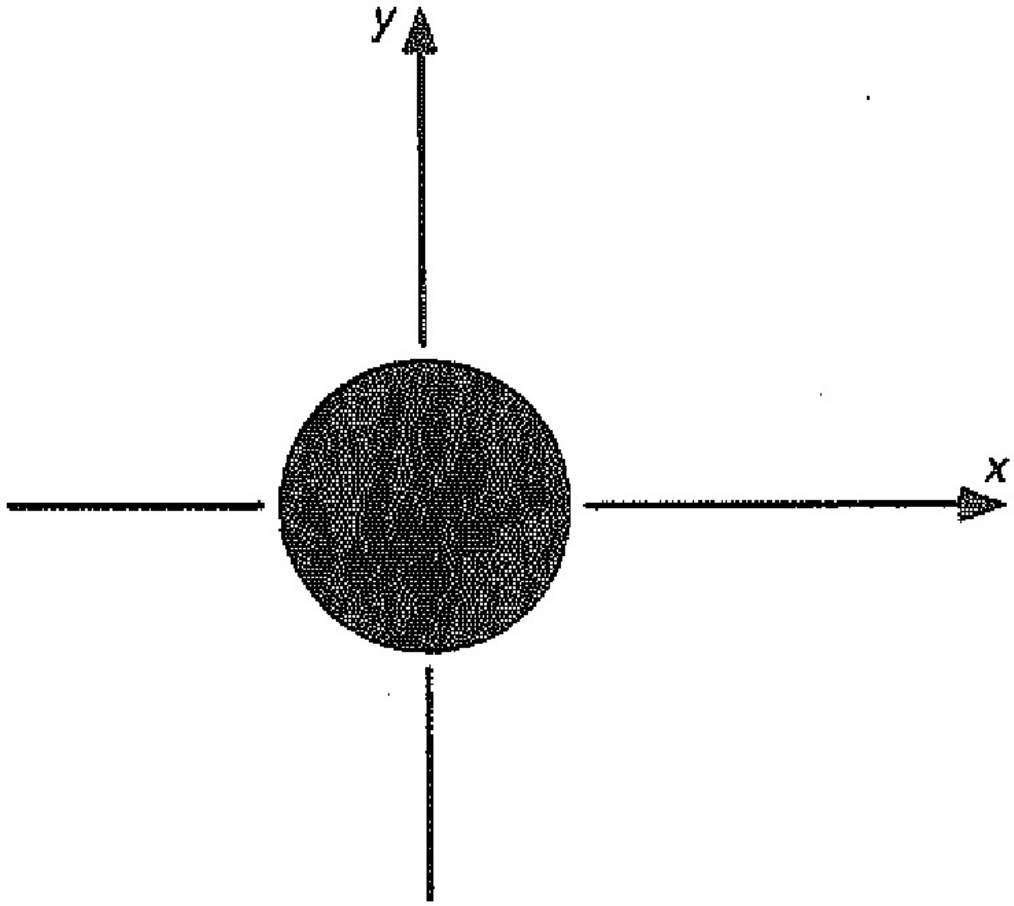
gönderimini, c sayısını elde etmek için önce $Z = 0$'a, sonra $c^2 + c$ sayısını elde etmek için z = c'e, sonra $c^4 + 2c^3 + c^2 + c$ sayısını elde etmek için $z = c^2 + c$ sayısına vb. uygulayacaktır. Eğer, $0, c, c^2 + c, c^4 + 2c^3 + c^2 + c,$ ...dizisi yakınsaksa c tarafından temsil edilen nokta siyah renklidir; aksi halde

beyaz renklidir. Makine, böyle bir dizinin *yakınsak* olup olmadığını nasıl bilir? İlke olarak bu soruda, dizinin *sonsuz* sayıda terim sonrasında ne olacağının bilindiği farz edilir. Yani soru, tek başına, hesaplanabilir bir konu değildir. Neyse ki, elemanların yalnız sonlu bir sayısından sonra dizinin ıraksak olduğunu söyleyebilmenin birçok yolu vardır (Gerçekte, $1 + \sqrt{2}$ yarıçapındaki daireye ulaşır ulaşmaz dizinin ıraksak olduğundan emin olabiliriz).

Buna göre, belli bir anlamda, Mandelbrot kümesini tümleyen küme (yani, beyaz alan) tekrarlı sayılabilir bir kümedir, c kompleks sayısı beyaz alanda yer alıyorsa, bu gerçeği kanıtlayacak bir algoritma da var demektir. Peki, siyah alandan, yani bizzat Mandelbrot kümesinden ne haber? Siyah alanda yer aldığı sanılan bir noktanın gerçekten siyah alanda yer aldığını kesinlikle bildirecek bir algoritma var mı? Bu sorunun yanıtı henüz bilinmiyor.[9] Bu konuda meslektaşlarıma ve uzmanlara danıştım, ama hiç birisi böyle bir algoritmanın varlığından haberdar görünmüyordu. Böyle bir algoritmanın varolmadığına dair bir kanıta da rastlamamışlardı. En azından, siyah alan için bilinen bir algoritma mevcut değil gibi görünüyor. Belki Mandelbrot kümesini tümleyen küme, aslında, tekrarlı sayılabilir fakat yinelenemeyen bir kümenin örneğidir.

Bu öneriyi daha açık irdelemeden önce, şöylece değindiğim bazı konuları ayrıntılamak gerekecek. Bu konular, fiziğin hesaplanabilir ligi ile ilgili olarak daha sonra yapacağımız tartışmalar yönünden önemlidir. Daha önceki tartışmalarda pek fazla açık ifadeler kullandığım söylenemez. Argand düzlemindeki noktaların kümeleri, yani kompleks sayıların kümeleri, için ‘tekrarlı sayılabilir’ ve ‘yinelenen’ gibi terimler kullandım. Bu terimlerin, kesinlikle, yalnız doğal sayılar veya diğer sayılabilir kümeler için kullanılması gerekir. III. Bölüm’de reel sayıların sayılamadığını ve bu nedenle kompleks sayıların da sayılamadığını çünkü reel sayıların, kompleks sayıların özel bir türü, yani sanal kısımları sıfıra eşit kompleks sayılar olarak düşünülebildiklerini görmüştük. Aslında, reel sayılar kadar ‘çok sayıda’ yani C kadar, kompleks sayı vardır (Kompleks sayılarla reel sayılar arasında bire-bir ilişki kurmak için, her bir kompleks sayının reel ve sanal kısımlarının genişletilmiş ondalık açılımlarını alabilir ve bunların karşıtı reel sayının tek ve çift rakamlarına göre ilişkilendirebiliriz: Örneğin, 3.6781 ... + i 512.975 ... kompleks sayısı 50132 . 6977851... sayısına karşı gelecektir).

Bu sorundan kaçınmanın bir yolu, yalnız hesaplanabilir kompleks sayıları kullanmaktır, çünkü III. Bölüm’de gördüğümüz gibi, hesaplanabilir reel sayılar, ve bu nedenle de hesaplanabilir kompleks sayılar, gerçekten sayılabilir. Ancak bu konuda ciddi bir zorluk vardır: İlgili algoritmalarına göre verilen hesaplanabilir iki sayının birbirine eşit olup olmadığına karar vermemizi sağlayacak bir algoritmaya sahip değiliz! (Aralarındaki farkı algoritmik yöntemle oluşturabiliriz ama bu farkın sıfır olup olmadığına algoritmayla karar veremeyiz.)



Şekil 4.4. Birim disk, uygun bir bakış açısından yinelenen bir küme olarak yorumlanabilir.

Sırasıyla 0.99999... ve 1.00000... rakamlarını üreten iki algoritmanın varolduğunu farz edelim, İki sayının eşit olduğunu gösterecek şekilde 9'ların veya 0'ların sonsuza değin devam edip etmeyeceklerini, veya sonunda bir başka rakamın ortaya çıkarak sayıların eşit olmadığını gösterip göstermeyeceğini asla bilemeyiz). Bu durumda, bu sayıların eşit olup olmadığını asla öğrenemeyebiliriz. Argand düzlemindeki birim disk gibi basit bir kümeyle bile (merkezden uzaklıkları bir birimden fazla olmayan noktaların kümesi, yani Şekil 4.4'deki siyah alan) bir kompleks sayının disk üzerinde gerçekten yer alıp almadığına kesinlikle karar vermemizi sağlayacak bir algoritma bulunmayabilir. Sorun, diskin içindeki (veya dışındaki) noktalardan değil, fakat diskin tam sınırında, yani bizzat birim çemberde, yer alan noktalardan kaynaklanır. Birim çemberi, diskin bir parçası olarak kabul edelim. Bir algoritmanın, herhangi bir karmaşık sayının reel ve reel olmayan kısmına ait rakamları ürettiğini varsayalım. Bu kompleks sayının birim çember üzerinde yer aldığından kuşkulanıyorsak, bunu mutlaka kanıtlayamayabiliriz.

$x^2 + y^2$

hesaplanabilir sayısının gerçekten 1'e eşit olup olmadığına karar verebileceğimiz bir algoritma yoktur, çünkü buna karar vermek, hesaplanabilir kompleks $x + i\,y$ sayısının, birim çember üzerinde yer alıp almadığının saptanması kriterinden başka bir şey değildir.

Kuşkusuz, istediğimiz bu değil. Birim disk elbette yinelenen küme kabul edilmeli. Birim diskten daha basit pek fazla küme yok! Sorunu çözmenin bir yolu sınırı gözardı etmek olabilir. Diskin gerçekten içindeki ve diskin gerçekten dışındaki noktalarla ilgili olarak bu gerçekleri kanıtlayan bir algoritma mevcuttur (Sadece $x^2 + y^2$ rakamlarını peşpeşe üreterek 0.99999...'da ondalık noktasından sonra 9'dan başka bir rakam, veya 1.00000...'de 0'dan başka bir rakam bulabiliriz). Bu bağlamda birim disk yinelenen bir kümedir. Fakat savların, çoğu kez, sınırlarda neler olup bittiğine bakarak

ifade edilmesi gerektiğinden, matematik yönünden bu bakış açısı oldukça anlamsız bir yaklaşımdır. Öte yandan, böyle bir bakış açısı fizik yönünden uygun olabilir. Bu konuya daha sonra döneceğiz.

Benimseyebileceğimiz ve konuyla yakından ilişkili bir görüş olanağı daha var ve bu görüşte, hesaplanabilir kompleks sayılarla ilgilenilmiyor. Söz konusu kümenin içindeki veya dışındaki kompleks sayıları saymaya çalışmak yerine, bir kompleks sayı verilmesi koşuluyla, bu sayının kümenin içinde mi, yoksa tümleyenin içinde mi yer aldığına karar veren bir algoritmaya gerek duyarız sadece. 'Verilmesi' dedim, çünkü denemekte olduğumuz her kompleks sayıda, reel ve reel olmayan kısımlara ait birbirini izleyen rakamlar, istediğimiz sürece, belki sihirli bir yöntemle, birbiri ardına ortaya çıkarlar. Bu rakamları ortaya çıkarmak için, bilinen veya bilinmeyen herhangi bir algoritmanın varolmasına ihtiyacım yok. Yalnız ve yalnız kompleks sayının gerçekten kümede yer alması koşuluyla, tek bir algoritma bu gibi rakamlar dizisine uygulandığında, sınırlı sayıda aşamalar sonrası 'evet' diyorsa, bir kompleks sayılar kümesi, 'tekrarlı sayılabilir' addedilecektir. Bu konuda açıkladığımız ilk görüş gibi bu görüş de, şuurları 'tanımıyor'. Bu nedenle birim diskin içinin ve birim diskin dışının her biri, bu anlamda, tekrarlı sayılabilir olurken, sınırın kendisi olmayacaktır.

Her iki görüş de bana gerçekten gereksinim duyulan görüş gibi gelmiyor.[10] Mandelbrot kümesine uygulandığında, 'sınır tanımamak' felsefesi, kümenin karmaşıklığının çoğuna ulaşamayabilir. Bu küme kısmen lekeler'den -içi dolgulu alanlar- ve kısmen 'filizler'den oluşur. En karmaşık kısımlar, alabildiğine kıvrılarak uzanan filizlerde yer alır. Ancak filizler, kümenin içinde yer almazlar ve bu nedenle, iki felsefeden birini benimsediğimiz takdirde, dikkate alınmayacaklardır. Böyle de olsa, yalnız lekelerin dikkate alındığı Mandelbrot kümesinin 'yinelenen' olup olmadığı açıkça bilinmemektedir. Sanırım bu sorunun kaynağı, Mandelbrot kümesi ile ilgili kanıtlanmamış bir iddiadadır: Küme, 'yerel bağlantılı' mıdır? Bu terimin anlamını veya konumuzla ilgisini burada açıklamak niyetinde değilim. Sadece, bu konuların zor konular olduğuna ve henüz çözümlenmemiş olan Mandelbrot kümesiyle ilgili, ve bazıları günümüzün matematik araştırmalarının ön cephesinde yer alan soruların sorulmasına neden olabileceğine dikkat çekmek istiyorum.

Kompleks sayıların sayılamaz olduğu problemine yeni yaklaşımlar getiren ve benimseyebileceğimiz başka görüşler de vardır, Hesaplanabilen tüm kompleks sayıları ele almak yerine, bu sayılardan ikisinin eşit olup olmadığına karar vermenin hesaplanabilir bir konu olduğunu savunan nitelikte uygun bir alt-kümeyi ele alabiliriz. Böyle bir alt-küme, sayıların reel ve reel olmayan kısımlarının her ikisinin rasyonel sayılar olarak alındığı, '*rasyonel*' kompleks sayılardır. Ancak, fazlaca sınırlayıcı olması nedeniyle bu görüşün, Mandelbrot kümesinin sarmaşık filizleriyle başa çıkabileceğini sanmıyorum. Belki biraz daha tatmin edici bir yöntem olarak cebirsel sayılardan, yani tamsayı çarpanlı cebirsel denklemlerin çözümleri olan kompleks sayılardan, yararlanabiliriz. Örneğin,

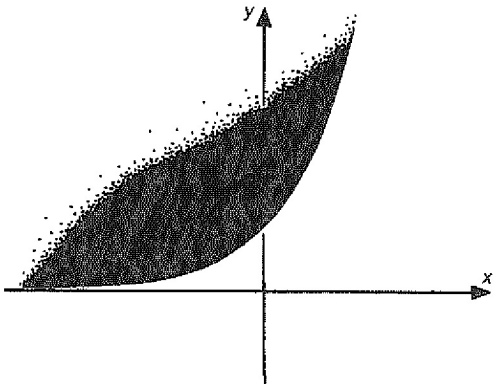
$$129\, z^7 - 33z^5 + 725z^4 + 16z^3 - 2z - \mathbf{3} = 0$$

denkleminin z için tüm çözümleri cebirsel sayılardır. Cebirsel sayılar sayılabilir, hesaplanabilir ve bunlardan ikisinin eşit olup olmadığına karar vermek gerçekten hesap edilebilir bir konudur (Rastlantıya bakın ki, bu sayılardan pek çoğu birim çemberin üzerinde ve Mandelbrot kümesinin filizleri üzerinde yer alır). İstenirse, Mandelbrot kümesinin yinelenen olup olmadığı sorusunu, bu sayılarla ifade edebiliriz.

Cebirsel sayılar, yukarıda değindiğimiz iki küme bakımından uygun olabilirse de, genelde karşılaştığımız tüm zorlukları çözümleyemezler. Argand düzleminde $x + i\,y\ (= z)$ için

$$y \geq e^x$$

bağıntısıyla tanımlanan kümeyi (Şekil 4.5 deki siyah alan) ele alalım. Kümenin içi ve tümleyen kümenin içi, yukarıda açıklanan görüşlerin herhangi birine göre, tekrarlı sayılabilir, fakat (1882'de kanıtlanan ünlü F. Lindemann teoremine göre) sınırın kendisi $y = e^x$ tek bir cebirsel noktaya, $z = i$'e, sahiptir. Bu durumda cebirsel sayılar, sınırın algoritmik özelliğini araştırmak konusunda bize yardımcı olmaz! Bu kendine özgü nitelikteki konuda yeterli olabilecek bir başka hesaplanabilir sayılar altsınıfı bulmak zor olmayabilir ama doğru görüşe henüz ulaşamamış olmanın sıkıntısından da kurtulamıyoruz.



Şekil 4.5. $y \geq e^x$ bağıntısıyla tanımlanan küme de 'yinelenen' olarak nitelemnelidir.

Yinelenmeyen Matematik Problemlerine Bazı Örnekler

Matematiğin yinelenmeyen problemlerle karşılaşıldığı birçok alanı vardır. Bu nedenle, yanıtı ya 'evet' veya 'hayır' olan, fakat bu yanıtlardan hangisinin doğru olduğuna karar verilmesini sağlayacak genel bir algoritmanın varolmadığı problemler sınıflarından bazıları son derece basit görünüşlüdür.

Önce, tamsayı çarpanlı cebirsel denklem sistemlerinin tamsayı çözümlerinin bulunması problemini ele alalım. Bu denklemler, İ.Ö. üçüncü yüzyılda yaşamış olan ve bu tip denklemleri inceleyen Yunan matematikçi Diophantos'un adıyla Diophantos denklemleri olarak anılır. Söz konusu denklemler kümesine örnek olarak,

$z^3 - y - 1 = 0,\ yz^2 - 2x - 2 = 0,\ y^2 - 2xz + z + 1 = 0$

denklemlerini verebiliriz ve burada problem, *x, y* ve *z tamsayı* değerleri için denklemlerin çözülür olup olmadığına karar vermektir. Gerçekte, bu denklemler,

$x = 13,\ y = 7,\ z = 2$

olarak verildiğinde çözülebilir. Ancak, zorunlu bir Diophantos denklemleri kümesi için[X] bu kararı verecek bir algoritma yoktur: Diophantos aritmetiği, basit içeriğine karşın, algoritmik olmayan matematiğin kapsamındadır. Daha da basit bir örnek *manifoldların topolojik eşdeğerliliğidir.* VIII. Bölüm'de tartışılacak konularla oldukça ilişkili olduğu için bu örneğe kısaca değiniyorum. 'Manifold'un ne olduğunu anlamak için önce bir ipin ilmeğini düşünün; bu, *bir* boyutlu bir manifolddur. Sonra kapalı bir yüzey düşünün; bu da *iki* boyutlu bir manifolddur. Daha sonra, *üç* veya daha fazla boyuta sahip bir 'yüzey' düşlemeye çalışın. İki manifoldun 'topolojik eşdeğerliğinin' anlamı, ikisinden birinin diğerinin üstüne, kopmadan veya yapışmadan, sürekli bir hareketle şekil değiştirerek örtülebilmesidir. Bu nedenle, bir küre yüzeyi ile bir küpün yüzeyi topolojik eşdeğerlidir;

öte yandan her ikisi, bir yüzüğün veya bir çay fincanının yüzeyi ile eşdeğerli değildir; halbuki yüzük ile çay fincanı topolojik olarak eşdeğerlidir. Bu bize, *iki*-boyutlu manifoldlar için, bunlardan ikisinin topolojik eşdeğerli olup olmadığına karar verecek bir algoritma bulunduğunu gösteriyor. Üç-boyutlular için bu sorunun yanıtı kitabın yazıldığı bu günlerde, henüz bilinmemektedir, ama dört ve daha fazla boyutlar için, eşdeğerliliği saptayacak bir algoritma yoktur. Dört-boyutlu manifold örnekleri fizikle biraz ilgilidir, çünkü Einstein'ın genel görelilik teorisi uyarınca, uzay ve zaman birlikte 4 boyutlu bir manifold oluştururlar; (bkz. V. Bölüm) Geroch ve Hartle (1986), bu algoritma-dışı niteliğin, 'kuantum kütleçekim kuvveti' ile ilişkili olabileceğini öne sürmüşlerdir (Bkz. VIII. Bölüm).

'Sözcük problemi' adı verilen başka tür bir problemi ele alalım.[11] Diyelim ki, bir çeşit simgeler alfabemiz var ve bu simgelerin çeşitli dizilerine 'sözcükler' adı veriliyor. Sözcüklerin anlamları olması gerekmiyor, fakat diyelim ki elimizde daha başka 'eşitlikler' kurabilmemizi de sağlayan, sözcükler arasındaki 'eşitlikleri' gösteren belirli (ve sonlu) bir liste mevcut. Daha fazla eşitlikler bulmak için, elimizdeki listede yer alan sözcüklerin yerini tutacak ve bu sözcüklerin bazı bölümlerini içerecek başka sözcükler (doğal olarak daha uzun sözcükler) üretebiliriz. Sözcüklerin her bir bölümünün yerine, listeye göre eşit olduğu varsayılan bir başka bölüm konulabilir. Bu durumda problem, verilen herhangi bir çift sözcüğün, bu kurallar çerçevesinde 'eşit' olup olmadıklarına karar vermektir.

Örneğin, ilk listemize aşağıdaki sözcükleri alabiliriz:

EAT = AT
ATE = A
LATER = LOW
PAN = PILLOW
CARP = ME

Bu sözcüklerden, örneğin,

LAP = LEAP

sözcüklerini aşağıdaki gibi, sürekli olarak türetebiliriz:

LAP = LATEP = LEATEP = LEAP

Şimdi problem, bir çift sözcük verildiğinde, birinden diğerine geçebilir miyiz? Örneğin, CATERPILLAR'dan MAN'i, veya diyelim, CARPET'dan MEAT'i türetebilir miyiz? Birinci örnek için yanıtın 'evet', İkincisi için 'hayır' olduğunu düşünelim. Yanıt 'evet' ise, bunu göstermenin doğal yolu, aralarındaki olası bir ilişkiyi kullanarak her bir sözcüğün bir öncekinden türetildiği bir eşitlikler dizisi sergilemektir. Değişecek harfleri kalın punto harflerle, henüz değiştirilmiş olanları italik harflerle göstermek suretiyle, aşağıdaki türevleri elde ederiz:

C**ATE**RPILLAR = CARPILL**A**R = CARPIL**LATER**= CAR**PILLOW** = **CARP**AN = ME**A**N = M**EAT**EN = M**ATE**N = MAN.

Bilinen kurallara uyarak CARPET'dan MEAT'i türetmenin mümkün olmayacağını nasıl söyleyebiliriz? Bunun için biraz daha düşünmemiz gerekir ama çok çeşitli yollar bulmamız hiç de zor değil. En basit yol şu olabilir: İlk listemizdeki her 'eşitlik'de, eşitliğin her iki tarafındaki A'ların sayısı artı W'ların sayısı artı M'lerin sayısı aynıdır. Buna göre, A'ların, W'ların ve M'lerin toplamı, yapabileceğimiz herhangi bir türevler dizisi boyunca değişmez. Ancak, CARPET için bu sayı 1 iken MEAT için 2'dir. Sonuçta, uygulanabilen bir türev yöntemiyle CARPET'dan MEAT'i türetmemizin bir yolu yoktur.

İki kelime 'eşit' olursa bunu, bilinen kurallar çerçevesinde, biçimsel bir simgeler dizisi halinde gösterebiliriz. Oysa 'eşit olmamaları' durumunda, kurallar *hakkındaki* savlara baş vurmamız gerekiyor. Sözcükler gerçekte 'eşit' oldukları zaman sözcüklerin arasındaki 'eşitliği' göstermek için kullanabileceğimiz belirgin bir algoritma vardır. Yapmamız gereken sadece, sözcüklerin olası tüm dizilerini leksikograftk bir liste halinde düzenlemek ve sonra, uygulanabilir herhangi bir kural çerçevesinde ikinci sözcüğün birincisinden türemediği, yanyana bir çift sözcüğün yer aldığı böyle bir diziyi listeden çıkarmaktır. Geriye kalan diziler, sözcükler arasında aradığımız tüm 'eşitlikleri' gösterecektir. Oysa, iki sözcüğün ne zaman 'eşit' olmadığına karar vermek için, genelde, böyle bir belirgin algoritma yoktur, ve bunu kanıtlamak için 'zekâ'ya başvurmak zorunda kalabiliriz (Gerçekten de, CARPET ve MEAT'in 'eşit' olmadığını göstermek için yukardaki 'hile'nin farkına varmam uzun sürdü. Başka bir örnek için, tamamen farklı bir 'hile' gerekebilir. Bu arada zekâ, bir 'eşitliğin' *varlığını* göstermek için, gerekli olmasa bile yararlı olabilir).

Yukarıdaki birinci liste örneğindeki beş 'eşitlik' için, gerçekten 'eşit olmadıkları' zaman iki sözcüğün eşit olmadığını kanıtlayan bir algoritma bulmak o kadar zor değildir. Ancak, bu amaçla uygulanabilir bir algoritma *bulmak* için zekâmızı büyük ölçüde kullanmamız gerekir! Aslında, birinci listedeki *tüm* olası seçeneklere evrensel olarak uyarlanabilecek hiçbir algoritmanın bulunmadığı sonunda anlaşılır. Bu bağlamda, sözcük probleminin hiçbir algoritmik çözümü yoktur. Genel sözcük problemi, yinelenmeyen matematiğe aittir!

İlk listede, iki sözcüğün ne zaman eşit olmadığına karar verecek bir algoritmanın varolmadığı bazı *özel* eşitlikler bile yer almaktadır:

AH = HA

OH = HO

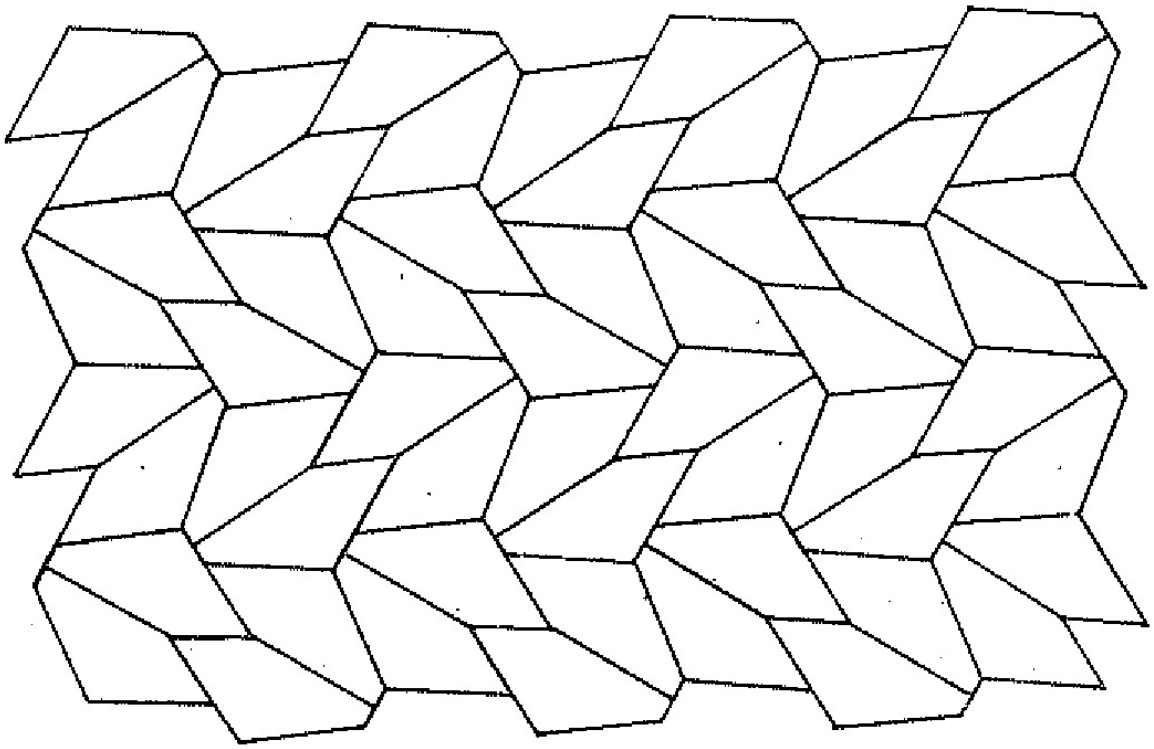
AT = TA

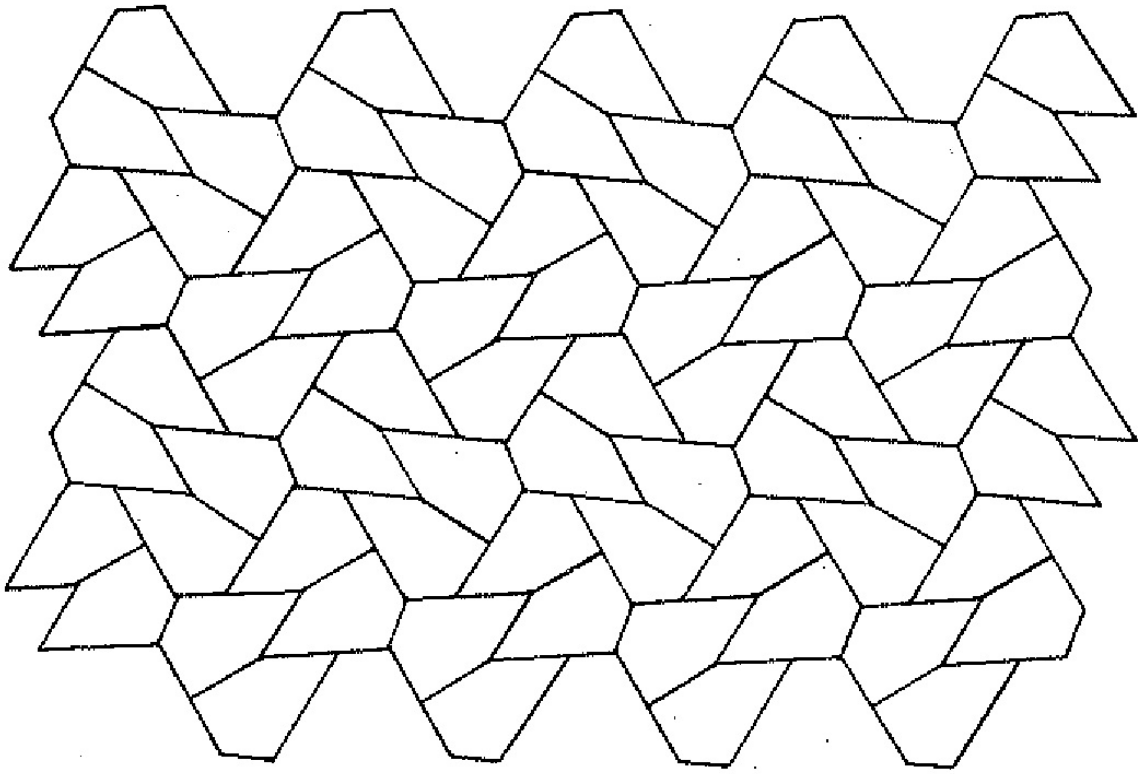
OT = TO

TAI = IT

HOI = IH

THAT = ITHT



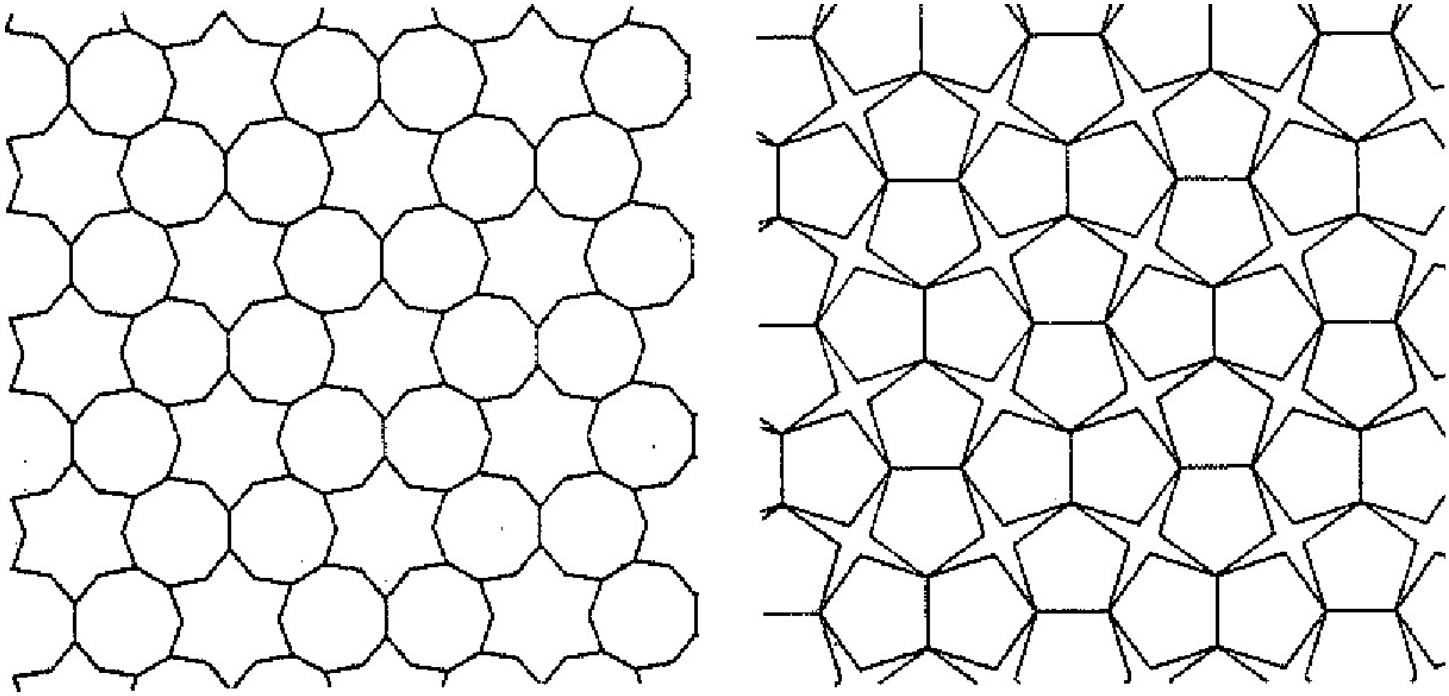


Şekil 4.6. Her birinde tek bir tür karo kullanılarak düzlemin periyodik kaplanması ile ilgili iki örnek (Marjorie Rice tarafından 1976'da bulunmuştur),

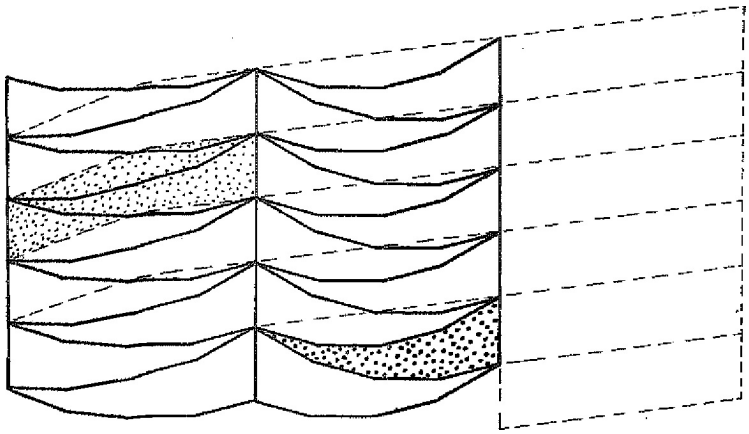
(Bu liste, G.S. Tseitin ve Dana Scott tarafından 1955'de verilen listeden uyarlanmıştır; bkz. Gardner 1958) Buna göre sözcük problemi, söz konusu listeyi kullanarak verilen iki sözcüğün 'eşitliği'ne algoritmik olarak karar veremeyeceğimiz bağlamında, *tek başına* bir örnek oluşturmaktadır.

Genel sözcük problemi, biçimselleştirilmiş matematiksel mantık varsayımlarından ortaya çıkmıştır ('formel sistemler' vs. daha önce ele almıştık.) ilk sözcük listesi, bir aksiyom sistemi ve sözcükler için bir türev kuralı rolünü oynar. Bu rol, yöntemin biçimsel kurallarıdır. Sözcük probleminin yinelenemez olmasının ispatı bu varsayımlardır.

Matematikte yinelenemeyen probleme son bir örnek olarak Eukleides düzleminin çokgen şekillerle kaplanması problemini ele alalım. Elimizde sınırlı sayıda ve farklı tipte şekiller bulunsun. Yalnız bu şekilleri kullanarak, aralarında hiçbir boşluk kalmayacak veya birbirinin üstüne binmeyecek şekilde düzlemi tamamen kaplamamızın mümkün olup olmadığını öğrenmek istiyoruz. Şekillerin böyle düzenlenmesine düzlemin 'karo kaplanması' adı verilir. Biliyoruz ki, bu gibi karo kaplamaları, kareler, eşkenar üçgenler, veya düzgün altıgenler (Şekil 10.2, X. Bölüm) kullanılarak gerçekleştirilebilir ama düzgün beşgenler kullanılarak gerçekleştirilemez. Düzlemin karo kaplanmasında, Şekil 4.6'da gösterildiği üzere, *düzgün olmayan* iki beşgenden her biri gibi diğer birçok tekli şekiller de düzlemin karo kaplanmasını oluşturabilir. Bir çift şekil kullanılarak, karo kaplama daha özenli yapılabilir. Şekil 4.7'de iki basit örnek verilmektedir. Tüm bu örneklerin ortak özelliği 'periyodik', yani her iki yönde tekrarlanabilir olmasıdır.



Şekil 4.7. İki ayrı karo türü kullanarak, düzlemin periyodik kaplanması ile ilgili iki örnek.



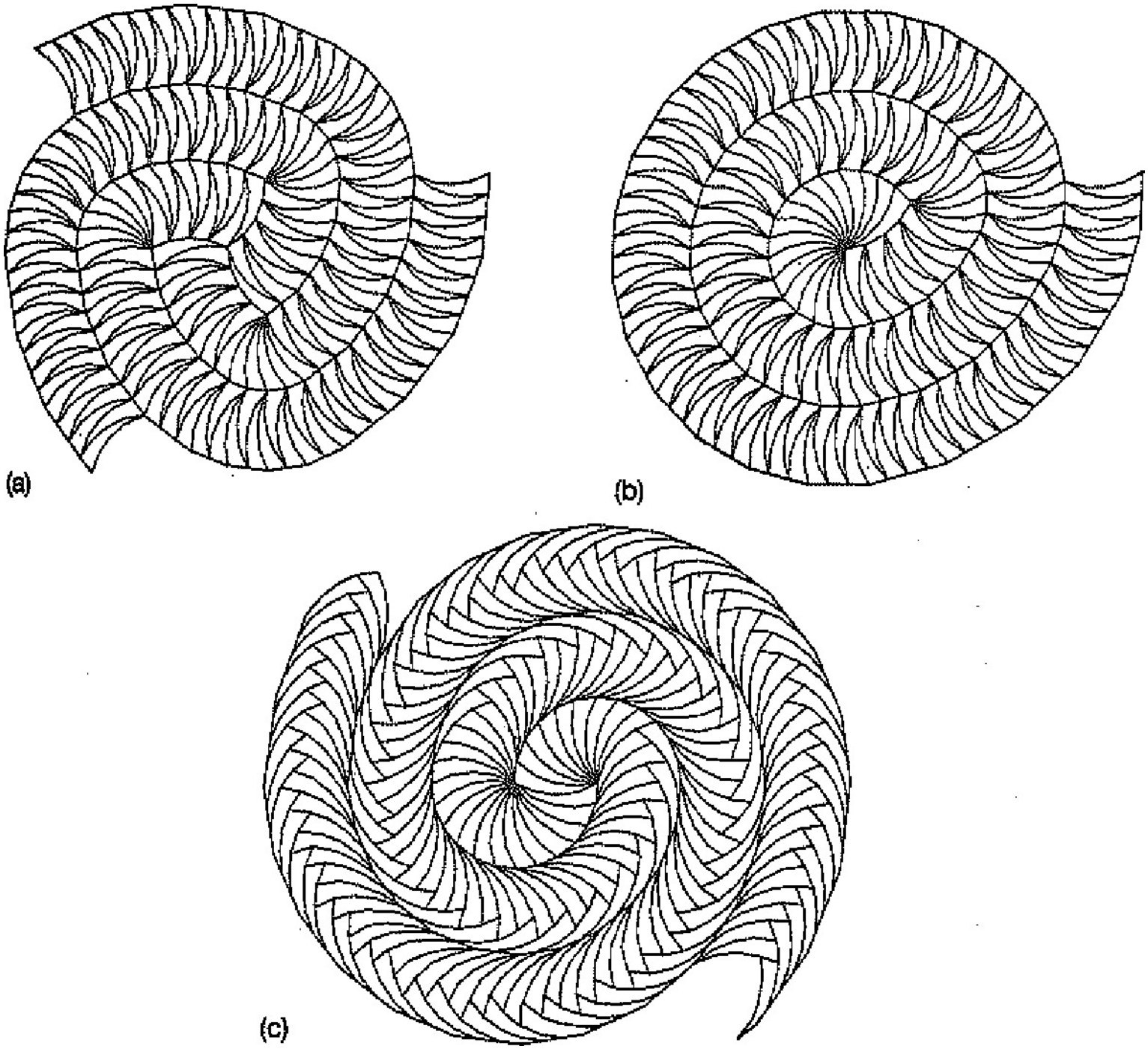
Şekil 4.8 Periyot paralelkenarı ile ilişkilendirilmiş olarak gösterilen bir periyodik karo kaplama

Matematik terimleriyle ifade etmek gerekirse, *bir periyod paralelkenarı yankı* diyoruz; bu paralelkenar, bir şekilde işaretlenip sonra kenarlarına paralel olarak iki yönde tekrar tekrar yinelendiği zaman verilen karo kaplama desenini üretmektedir. Şekil 4.8, bunun bir örneğini göstermektedir: Diken biçiminde bir karoyla yapılan periyodik kaplama sol tarafta gösterilirken, periyodik kaplaması sağda gösterilen bir periyod paralelkenar ile ilişkilendirilmiştir.

Düzlem periyodik olmaksızın da kaplanabilir. Şekil 4.9'da 'helisel' **karoların Şekil 4.8'deki diken biçimi karolarla birlikte periyodik olmayan kaplaması** üç ayrı tipte gösterilmiştir. Bu değişik düzlem kaplama 'her yöne uzanabilen' olarak adlandırılır (açıkça belli nedenlerle!) ve daha Önce H. Voderberg tarafından bulunmuş şekle dayanılarak B. Grünbaum ve G.C. Shephard (1981-1987) tarafından uygulanmıştır. Bu tür karonun hem periyodik hem de periyodik olmayan kaplama yapabildiğine dikkat ediniz. Bu özellik tek veya küme halindeki karo kaplamalarda da görülür. Düzlemi yalnız periyodik olmayacak şekilde kaplayan tek karolar veya küme karolar var mıdır? Bu sorunun yanıtı evet'dir. Şekil 4.10'da, Amerikalı matematikçi Raphael Robinson (1971) tarafından inşâ edilen altı karolu bir küme görülmektedir.

Periyodik olmayan karo kümelerinin tarihçesinden biraz bahsetmek istiyorum (bkz, Grünbaum ve Shephard 1987). 1961 yılında, Çin kökenli Amerikalı mantık bilimcisi Hao Wang, karo kaplama

problemi ile ilgili olarak bir karar yönteminin varolup olmadığı sorusunu yöneltti. Başka bir deyişle, düzlemi tümüyle kaplayacak farklı çokgen karolardan oluşan belirli bir sonlu kümenin varolup olmadığına karar verecek bir *algoritma* var mıdır?[XI] Hao Wang, düzlemi herhangi bir şekilde kaplayacak farklı karolardan oluşan her sonlu kümenin, düzlemi gerçekte periyodik olarak da kaplayacağı gösterilebilseydi, böyle bir karar yönteminin gerçekten var olabileceğini göstermiştir.



Şekil 4.9. Şekil 4.8'dekinin aynı 'her yöne uzanabilen' biçimi kollanan üç ayrı periyodik olmayan 'helise' karo kaplama.

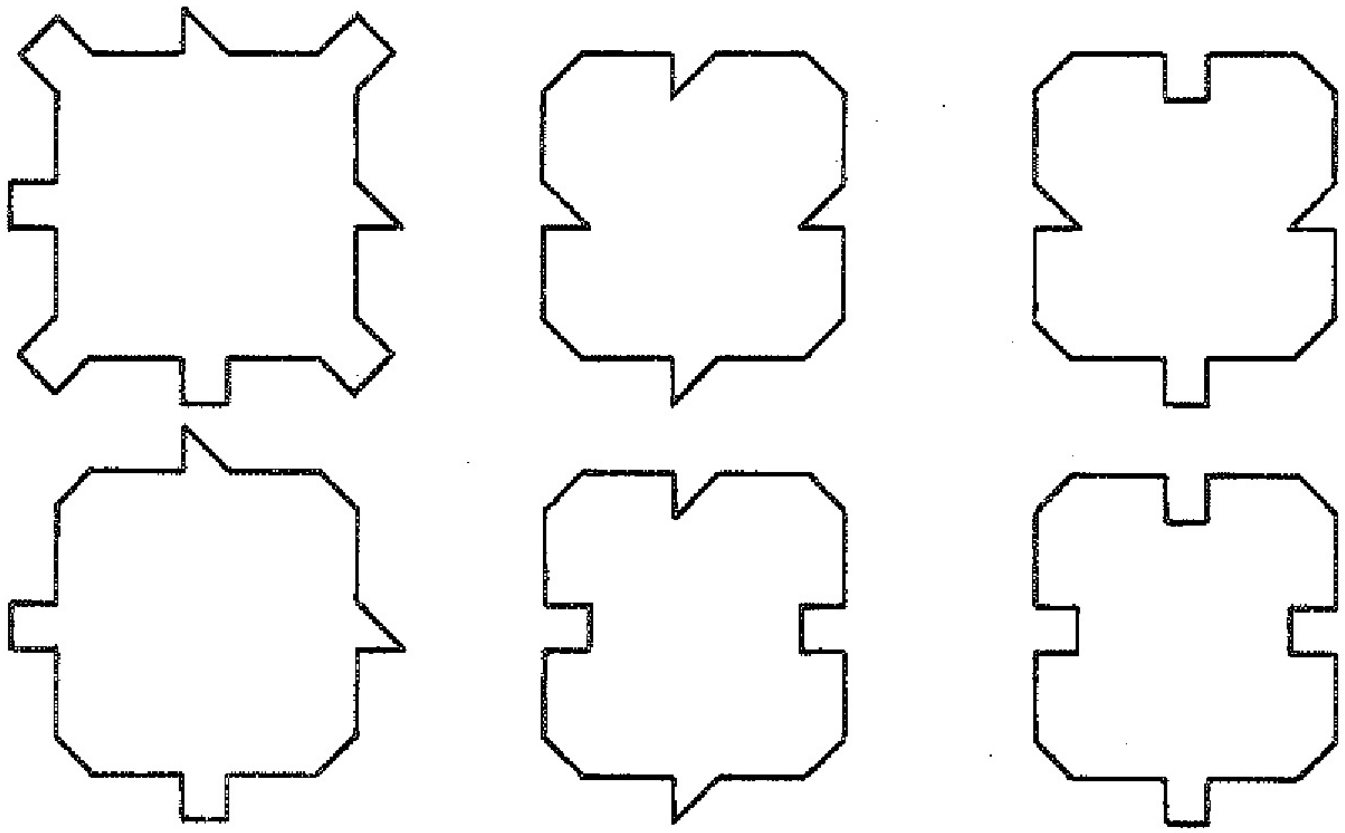
Sanırım, o zamanlar, böyle bir koşula aykırı bir kümenin, yani 'periyodik olmayan' karolar kümesinin, var olabileceğine inanılmıyordu. Ancak, 1966 yılında Robert Berger, Hao Wang'ın bazı ipuçlarını değerlendirerek, karo kaplama problemi ile ilgili hiçbir karar yönteminin varolmadığını göstermeyi başardı: Karo kaplama problemi de, yinelenemeyen matematik sorularının bir parçasıdır![12]

Böylece, Hao Wang'ın periyodik olmayan karolar kümesinin varolması gerektiği sonucundan

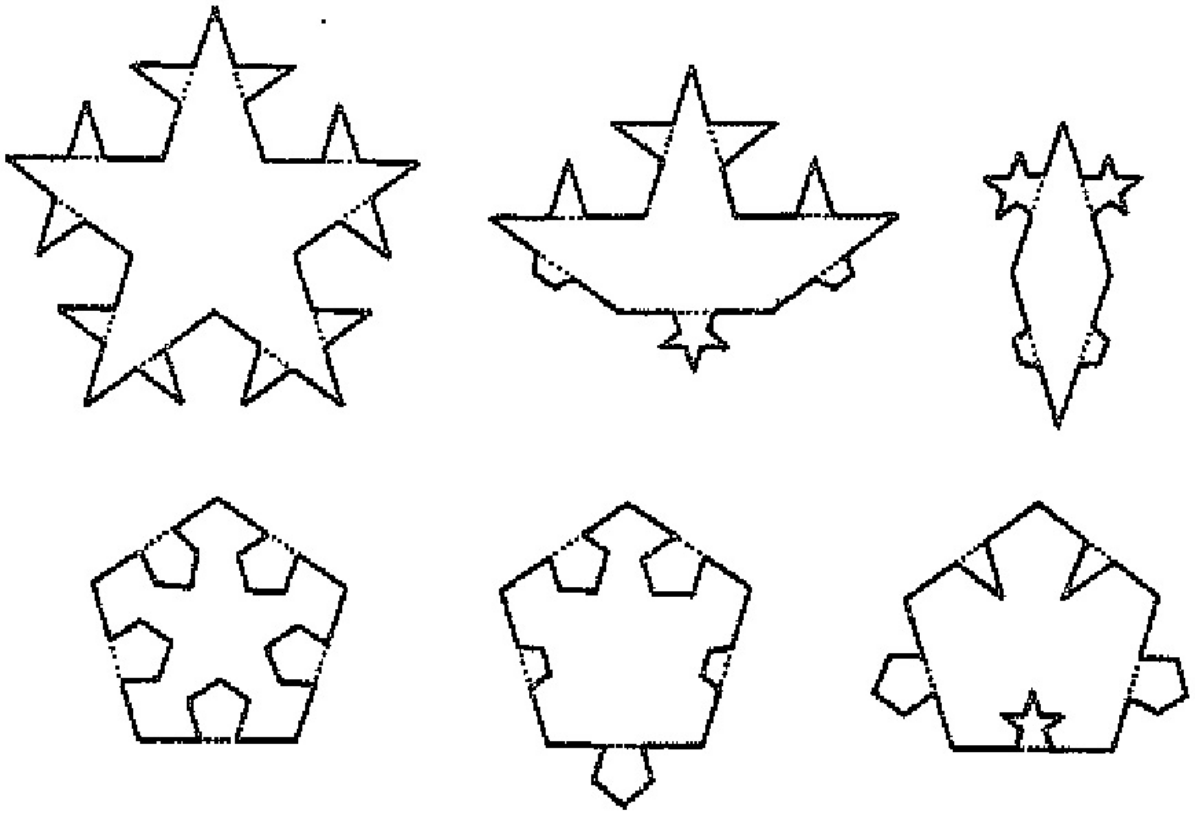
hareketle Berger, ilk periyodik-olmayan karolar kümesini inşa etmişti. Ancak, Berger kümesi 20 426 gibi son derece fazla sayıda karo kullanımını gerektirdiği için Berger, biraz daha beceri göstererek bu sayıyı 104'e indirdi. 1971 yılında Raphael Robinson söz konusu sayıyı, Şekil 4.10'da gösterilen 6 karoya kadar indirmiştir.

Başka bir periyodik-olmayan altılı küme Şekil 4.11'de gösterilmiştir. 1973'de, tamamen bağımsız bir düşünce çizgisi izleyerek bu kümeyi ben tasarımladım (Bu konuya X. Bölüm'de tekrar döneceğim; Şekil 10.3). Robinson'un periyodik-olmayan altılı kümesini gördükten sonra, bu sayıyı nasıl azaltabilirim diye düşünmeye başladım; keserek, tekrar yapıştırarak sürdürdüğüm çeşitli denemeler sonrası, karo sayısını ikiye düşürebildim. Şekil 4.12'de iki ayrı tasarım gösterilmiştir. Kaplama işlemi tamamlandığında ortaya çıkan periyodik şekiller, beş-katlı simetriye sahip ve kristal yapısına tamamen aykırı periyodiksi bir yapı dahil, dikkate değer birçok özelliklere sahiptir. Bu konuya daha sonra tekrar döneceğim. Matematiğin böylesine 'basit' bir alanının, bir düzlemin birbirine uyan parçalarla kaplanması gibi neredeyse 'çocuk oyunu' bir işlemin gerçekte matematiğin yinelenmeyen problemler konusunun bir kısmını oluşturması ilginç görülebilir. Aslında bu alanda zor ve çözümlenmemiş problemler vardır. Örneğin, tek karodan oluşan ve periyodik-olmayan bir kümenin varolup olmadığı bilinmemektedir.

Wang, Berger ve Robinson'un yaklaşımlarıyla problemde kare karolar kullanılmıştı. Ben, herhangi bir şekle sahip çokgenler kullanılabilir diyorum, yeter ki her bir karoyu göstermenin hesaplanabilir bir yöntemi var olsun. Bu yöntemlerden birisi, karoların, köşelerini, Argand düzleminin noktaları olarak kabul etmektir.

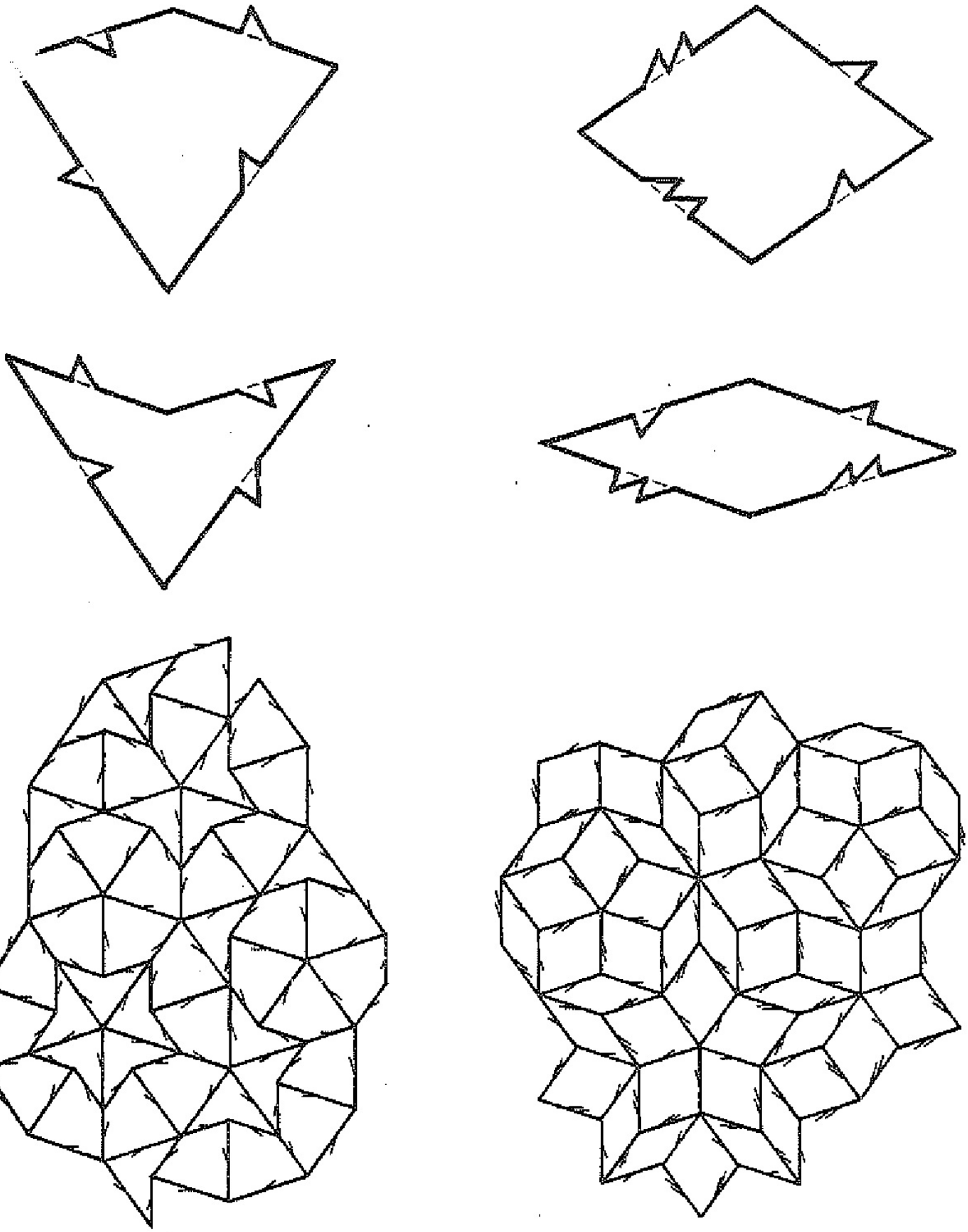


Şekil 4.10. Raphael Rabinson'un, düzlemi yalnız periyodik-olmayarak kaplayabilen altılı kümesi.



Şekil 4.11. Düzlemi yalnız periyodik-olmayarak kaplayabilen bir başka altılı küme.

Bu noktaları, cebirsel sayılarla göstermek pekâlâ mümkündür.



Şekil 4.12. Her biri sadece periyodik-olmayarak kaplayabilen iki çift karo ('Penrose karoları'); ve düzlemin her bir çift-karoyla kaplanmış bölgeleri.

Mandelbrot Kümesi Yinelenmeyen Matematiğe Benzer mi?

Yine Mandelbrot kümesi ile ilgili tartışmamıza dönelim. Açıklama kolaylığı yönünden Mandelbrot kümesinin, herhangi bir uygun anlamda, yinelenmediğini varsayacağım. Tümleyen küme tekrarlı sayılabilir olduğu için, kümenin kendisi tekrarlı sayılamaz. Sanırım, Mandelbrot kümesinin biçiminin, yinelenmeyen kümelerin ve yinelenmeyen matematik problemlerinin özellikleri hakkında bize vereceği bazı dersler var.

III. Bölüm'de karşılaştığımız Şekil 3.2'i ele alalım. Kümenin büyük bir bölümünün, Şekil 4.13'de 'A' olarak işaretlediğim, büyük bir kalp biçiminde bölge ile kaplandığına dikkat ediniz. *Kardioid* (yürek biçiminde) olarak adlandırılan şekil ve iç kısmı, matematiksel olarak, Argand düzleminde

$c = z - z^2$

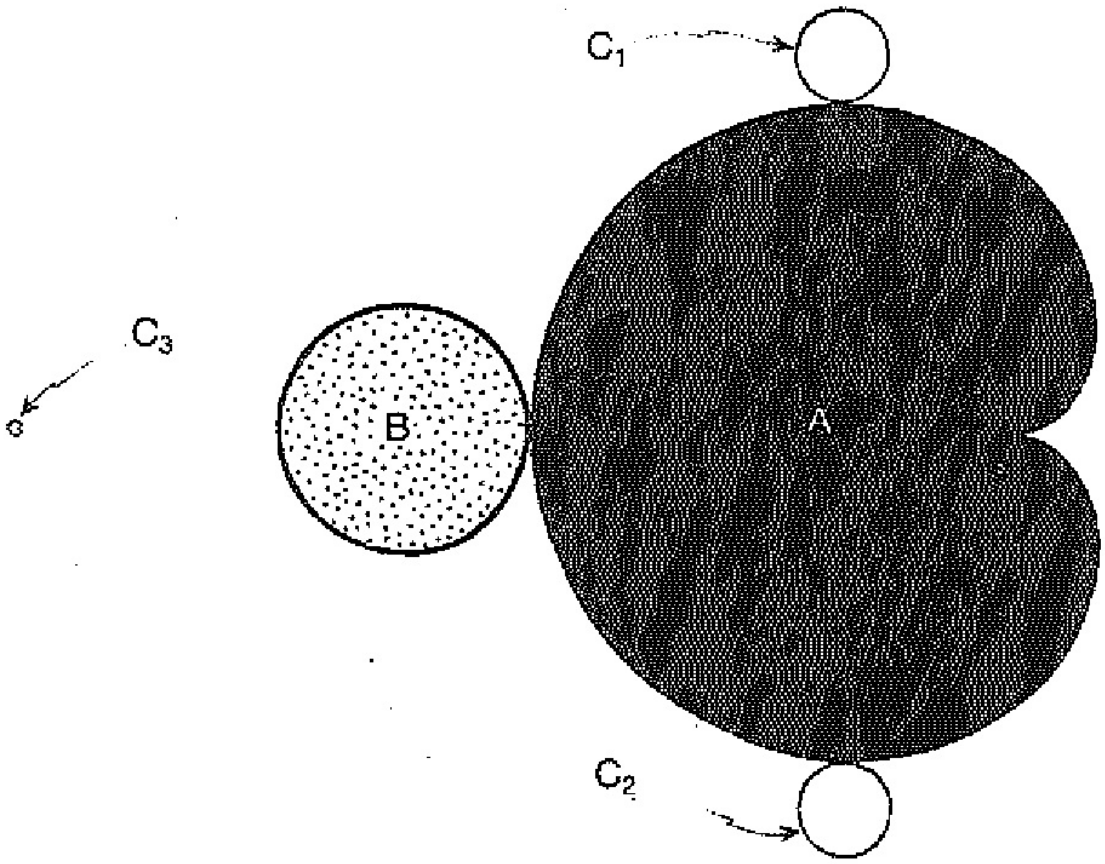
c noktaları kümesi olarak tanımlanabilir; burada z, merkezden uzaklığı 1/2'den az olan bir kompleks sayıdır. Bu küme, kuşkusuz, daha önce değinilen anlamda, tekrarlı sayılabilir: Bölgenin iç kısmında bir noktaya uygulandığında bu noktanın gerçekten iç bölgede bulunduğunu doğrulayacak bir algoritma vardır. Gerçek algoritma yukardaki formülden kolayca elde edilir.

Yürek biçimindeki ana bölgenin solundaki yuvarlak bölgeye (Şekil 4.13 - bölge B) bakınız. Yuvarlak bölgenin iç kısmı .

$c = z - 1$

noktaları kümesidir; burada z, merkezden 1/4'den az uzaklıktadır, Bu bölge gerçekten de bir diskin içidir, yani tam bir dairenin içinde kalan noktalar kümesidir. Bu bölge de, yukardaki anlamda, yinelenerek sayılabilir. Peki, yürek biçimi bölgenin üzerindeki 'siğilimsi' çıkıntılar nedir? Şekil 3.2'de yürek-biçimi bölgenin hemen üstünde ve hemen altında görülen ve Şekil 4.13'te C_1, C_2 olarak işaretlenen bu yuvarlak damlaların küme terimleriyle bildirimi şöyledir:

$c^3 + 2c^2 + (1 - z)\, c + (1 - z)^3 = 0$



Şekil 4.13. Mandelbrot kümesinin iç bölgesinin başlıca kısımları, basit algoritmik formüllerle tanımlanabilir.

burada *z,* merkezden 1/8 uzaklıktaki noktaların tümüdür. Aslında bu denklem bize yalnız C_1 ve C_2 yuvarlaklarını vermiyor, Şekil 3.2'de sol tarafta -Şekil 3.1'in ana bölgesi- ve Şekil 4.13'de C_3 ile işaretlenen kalp biçimindeki 'yavru' şekilleri de vermektedir. Yine, söz konusu bu bölgeler (birlikte veya ayrı ayrı), yukarıdaki formül sayesinde (daha önce önerilmiş olan anlamda) yinelenerek sayılabilir kümeleri oluşturur.

Mandelbrot kümesinin yinelenemeyen bir küme olduğunu kanıtlamaya çalıştığım gibi bir izlenim yarattıysam da kümenin, iyi tanımlanmış ve çok karmaşık olmayan bazı algoritmalar kullanarak, en geniş bölgelerini açıklığa kavuşturduk. Öyle görünüyor ki, bu işlemi sürdürmeliyiz. Kümedeki en belirgin bölgeler ve kuşkusuz bu bölgelerin kapladığı alanın (tümünü kaplamadığı durumlarda) ezici yüzdesi, algoritmik olarak hesaplanabilir. Tahmin ettiğim gibi küme tümüyle gerçekte yinelenen bir küme değilse, algoritmalarımızla ulaşamadığımız bölgelerin, çok hassas ve ulaşılması zor bölgeler olması gerekir. Üstelik, böyle bir bölgeyi saptadığımız zaman, algoritmalarımızı bu bölgelere ulaşmamızı sağlayacak şekilde geliştirmek şansımız da artacaktır. Fakat yine de (yinelenemez varsayımım doğruysa) ulaşılamayan başka bölgeler olacak, bu bölgeler geliştirilmiş algoritmalarımızla bile ulaşamayacağımız duyarlılık ve karmaşıklıkların belirsizliğinde, daha da derinlerde, gizli kalacaklardır. Böyle bir aşamada sezgilerimizi, becerimizi ve çalışkanlığımızı devreye sokarak onlara ulaşabiliriz, fakat yine de ulaşamadıklarımız olacak ve bu böylece sürüp gidecektir.

Sanırım bu durum, matematiğin çoğu kez, problemlerin zor olduğu ve büyük olasılıkla yinelenemez olduğu alanlarda faaliyet gösterdiğini açıklamaktadır. Belirli bir alanda karşılaşmamız olası en

olağan problemler, basit algoritmik yöntemlerle, yüzyıllardır bilinen yöntemlerle çözülebilir. Fakat bazıları ağın gözlerinden kaçıp gidecek ve onları çözebilmemiz için daha zor yöntemler bulmamız gerekecektir. Elimizden kaçıp kurtulanlar, kuşkusuz, matematikçileri tuzaklarına düşürecekler ve onları daha güçlü yöntemler bulmaya zorlayacaklardır. Bu gibi yöntemlerin, ilgili matematik alanının doğasının daha da derinliklerine inebilen sezgilere dayalı olması gerekecektir. Belki, fiziksel dünya anlayışımızda buna benzer bir şeyler vardır.

Daha önce ele aldığımız sözcük problemlerinde ve düzlem karo kaplama problemlerinde bu tür bir şeyi göz ucuyla fark etmeğe başlayabiliriz (Bu alanlar, matematiksel yöntemlerin henüz pek ilerleme kaydetmediği alanlar olsa dahi). Basit bir uslamlama yöntemi kullanarak, belirli bir sözcüğü başka bir sözcükten türetemiyeceğimizi gösterebilmiştik. Daha karmaşık problemlerin üstesinden gelmek için çok daha karmaşık uslamlama yöntemlerini devreye sokabiliriz. Sözcük probleminde tek bir yöntemin yeterli olmayacağını biliyoruz fakat kontrolümüzden kaçabilen örnekleri biraz daha özenli ve duyarlı yöntemlerle inşa edebiliriz. Nasıl inşa edebileceğimizi anlar anlamaz, belli bir problemin algoritmamızın elinden kurtulduğunun kesinlikle *farkına varır varmaz*, algoritmamızı, bu problemi kapsamına alacak şekilde geliştirebiliriz. Yalnız 'eşit' olmayan çift sözcükler algoritmamızdan kaçabildiğine göre, kaçtıklarının farkına varır varmaz 'eşit' olmadıklarını anlarız ve algoritmamızı buna göre uyarırız. Gelişen sezgilerimiz, algoritmanın gelişmesini sağlayacaktır!

Karmaşıklık Teorisi

Algoritmaların doğası, varlığı ve sınırları ile ilgili olarak yukarda ve önceki bölümlerde ileri sürdüğüm savlar 'ilke' düzeyindeydi. Uygulanabilir olup olmadıklarını fazlaca irdelemedim. Algoritmalarının varolduğunu ve bu algoritmaları nasıl inşa edeceğimizi bildiğimiz problemler için bile, bunları yaşama geçirmek büyük beceri ve çaba gerektirebilir. Bazen birazcık beceri ve sezgi gücüyle, bir algoritmanın daha az karmaşık veya son derece hızlı olmasını sağlayabiliriz. Bu gibi konular çoğu kez çok ayrıntılı ve teknik olup, algoritmaların yapılanması, anlaşılması ve geliştirilmesi alanlarında değişik bağlamlarda bir hayli çalışma yapılmış, algoritma çalışmalarına ivme kazandırılmıştır. Bu konuda ayrıntılı bir tartışmaya girmeyi uygun görmüyorum. Ancak, bir algoritmanın süratinin ne ölçüde artırılabileceği ile ilgili bazı kesin sınırların bilinmekte veya tahmin edilmekte olduğuna değinmekte yarar görüyorum. Algoritmik özelliğe sahip matematik problemleri arasında bile, doğaları gereği, algoritmik çözümleri diğer çözümlerine göre çok daha zor olanlar vardır. Bazıları ancak çok yavaş algoritmalarla (veya, olağanüstü bellek alanı, vb. gerektiren algoritmalarla) çözümlenebilirler. Bu çeşit sorularla ilgilenen teori, *karmaşıklık teorisi* adıyla tanınır.

Karmaşıklık teorisi, *bireysel* problemlerin algoritmik çözümlerinden çok, bir problem sınıfına dahil problemlere yanıt arayan genel bir algoritmanın bulunabileceği sonsuz problemler sınıflarıyla ilgilenir. Aynı sınıfa dahil farklı problemler farklı 'boyutlar'a sahip olabilirler; bir problemin boyutu n doğal sayısıyla ölçülebilir *(n* sayısının problemin boyutunu nasıl belirleyeceğini birazdan anlatacağım). Belirli bir sınıfa dahil her bir problemin gereksinim duyacağı sürenin uzunluğu -veya, daha doğrusu, ilk aşamaların sayısı- n'e bağlı bir N doğal sayısıdır. Biraz daha açıklamak gerekirse, diyelim n boyutundaki tüm problemler arasında algoritmanın kaydettiği en büyük aşama sayısı N'dir. Buna göre, n büyüdükçe N'de büyüyecektir. Gerçekte N, n'e göre çok daha hızlı büyüyecektir. Örneğin N, yaklaşık n^2, veya n^3, veya belki 2^n (büyük n değerleri için 2^n, n, n^2, n^3, n^4 ve n^5'

hepsinden çok daha büyük, ve hatta her *r* sabit sayısı için n^r'den daha büyüktür) ile orantılı olacak, veya *N,* yaklaşık olarak, diyelim

2^{2^n} (öncekinden de daha büyük bir sayı) ile orantılı olabilecektir.

Kuşkusuz, 'aşamaların' sayısı, algoritmanın uygulandığı makinenin tipine bağlı olabilir. II, Bölüm'de tanımlanan tek bantlı ve oldukça yetersiz tipteki Turing makinesiyle *N* sayısı, iki veya daha çok bantlı tip makineyle olduğundan daha hızlı artabilir (yani, makine daha yavaş çalışabilir). Bu gibi belirsizliklerden sakınmak amacıyla, hangi tip Turing makinesi kullanılırsa kullanılsın *N* artış oranı ölçümünün aynı kategoride yer almaşım sağlayacak şekilde N'in, n'in bir fonksiyonu olarak, büyüyebildiği durumları kapsayan geniş bir sınıflandırma yapılır. P olarak anılan ('polinom süre') bir sınıf, n, n^2, n^3, n^4, n^5 ...'lerden her birinin sabit katlarıyla[XII] verilen tüm hızları içerir. Başka bir deyişle, P kategorisindeki herhangi bir problem için ('problem' demekle, gerçekte, çözümleri için genel bir algoritmaya sahip problemler sınıfını kastediyorum)

$N \leq K x n^r$

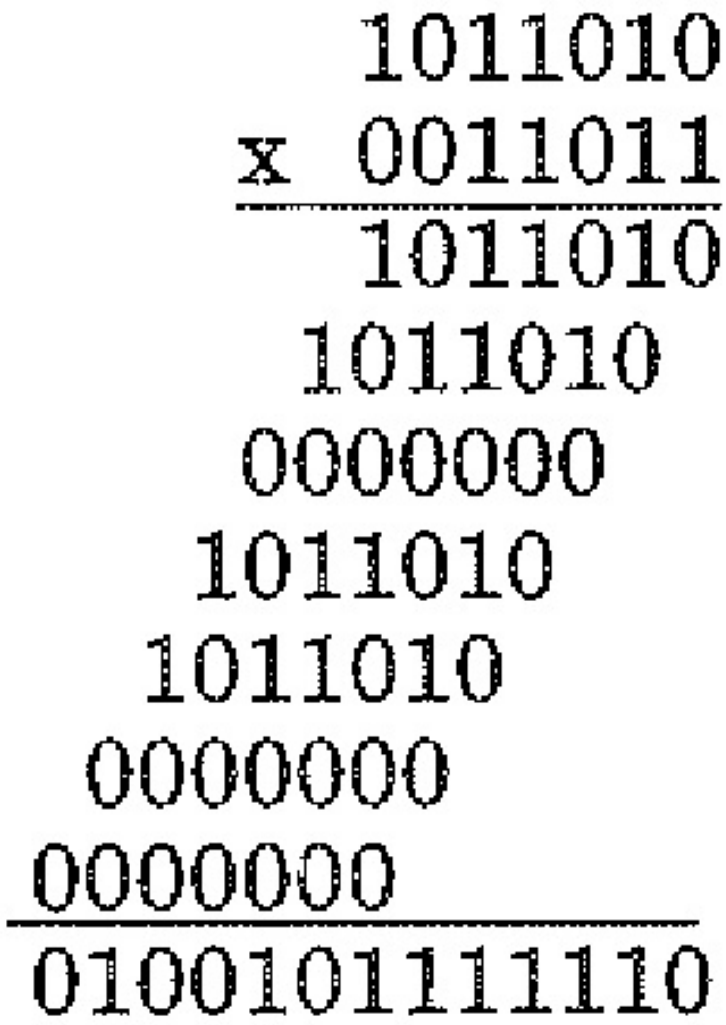
denklemine sahibiz; burada *K* ve r sabit sayılardır (n'den bağımsız). Bunun anlamı N'in, bir sabit çarpan kere n'nin bir kuvvetinden büyük olmadığıdır.

P kategorisine dahil basit ve tipik bir problem, kuşkusuz, iki sayının çarpımıdır. Bu konuya biraz daha açıklık getirmek için önce, *n* sayısının, çarpılacak sayıları nasıl karakterize ettiğini tanımlamalıyım. Her sayının ikilik sistemde yazıldığını ve *n*/2'in bu sayının ikilik sistemde yazılmış halinde her bir rakamını kaç kez geçtiğini gösterdiğini, yani n'nin toplam ikilik hane sayısını (dijit) verdiğini farz edelim (Sayılardan birisi diğerinden uzunsa, kısa olanıyla başlar ve diğerinin uzunluğuna ulaşıncaya kadar sola bir dizi sıfir ekleyebiliriz).

Örneğin *n* = 14 ise

1011010x0011011

olarak yazarız (aslında 1011010 x 11011 olarak yazılması gerekirken kısa sayıya sıfir ekleyerek yukarıdaki yazılımı elde ettik). Çarpımı, ikilik sistemde, 0x0 = 0, 0x1 = 0, 1x0 = 0, 1x1 = 1, 0 + 0 = 0, 0+1 = 1, 1 + 0 = 1, 1 + 1 = 10 olduğunu hatırlayarak, doğrudan doğruya şöyle yapabiliriz:



Tek tek yapılan çarpımların sayısı *(n/2)* x *(n/2)* - *n^2/4'*dür; buna ek olarak en fazla *(n^2/4)* - (n/2)'toplama işlemi yapılabilir. Bu durumda toplam *(n^2/2)-(n/2)* bireysel aritmetik işlemi vardır; çarpım işlemindeki elde sayılar için ayrıca bir kaç yedek aşamayı da buna ekleyebiliriz. Böylece toplam aşama sayısı $N = n^2/2$ olur ki (sadece en büyük terimi dikkate alarak) bu sayı kuşkusuz bir polinomdur.[13]

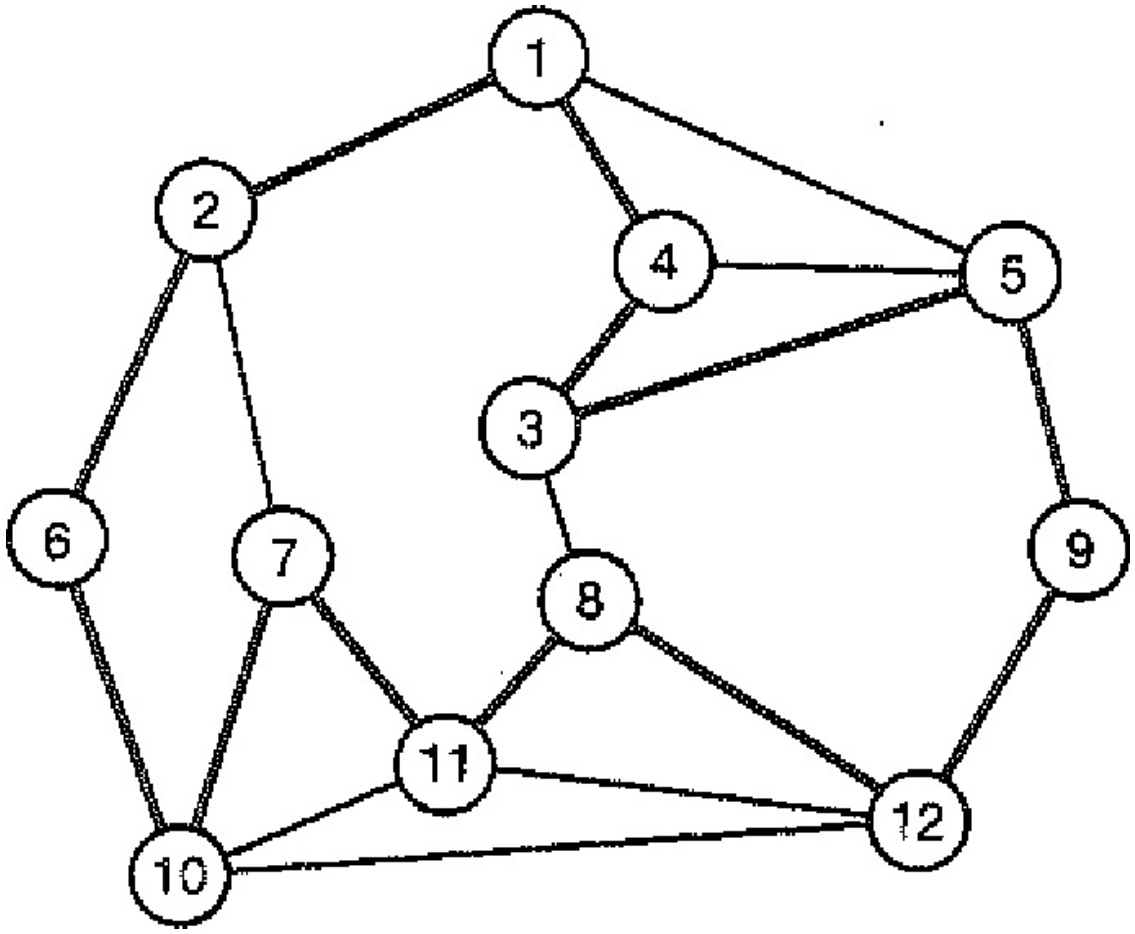
Bir problem sınıfı ile ilgili olarak genelde, problemin 'boyutu'nun *n* ölçüsünü, problemin bağımsız verilerini belirlemek için gerekli toplam ikilik hanelerin (bit) sayısı olarak alırız. Başka bir deyişle, verilen *n* için, problemin 2^n kadar farklı seçeneği söz konusudur (çünkü her hane, 0 veya 1 seçeneklerinden birisini alabilir, ve toplam *n* hane vardır) ve bu seçeneklerin N sayısını geçmeyen aşamada algoritma tarafından düzgün bir şekilde ele alınması gerekmektedir.

P kategorisinde yer almayan problemlerin(problem sınıflarının) birçok örneği vardır. Örneğin *r* doğal sayısından 2^{2^r} sayısını hesaplamak için, işlem bir yana, sadece yanıtını yazmak için 2^r kadar aşamaya gereksinim vardır; bu işlemde *n, r* sayısının ikilik gösterimindeki hane (digit) sayısıdır. 2^{2^r} aşamasının hesaplanması, işlemine benzer aşamaları, vb. gerektirecektir! Bu problemler polinom problemlerden çok büyüktür ve kuşkusuz P sınıfına dahil değildirler.

Yanıtları polinom sürede yazılabilen hattâ sağlamaları da aynı sürede yapılabilen sorular daha ilginçtir. Bu özelliğe sahip önemli bir problemler kategorisi (algoritmik çözümlenebilen soru sınıfları kategorisi) vardır. Böyle problemler NP problemleri (problem sınıfları) olarak anılır. NP'deki problemler sınıfına ait herhangi bir problemin çözümü varsa, algoritma bu çözümü verecektir ve polinom sürede bunun sağlamasını yapmak da mümkün olmalıdır. Problemin çözümü yoksa,

algoritma bunu böyle olduğunu söyleyecektir, fakat polinom sürede veya başka şekilde, bu sonucun sağlanması gerekmiyecektir.[14]

NP problemleri, hem matematikte hem pratik dünyada birçok konuyla ilgili olarak ortaya çıkabilir. Basit bir matematiksel örnek vereyim: Bir grafikte 'Hamilton devresi' adıyla anılan (son derece basit bir düşünceyi tanımlamak için oldukça iddialı bir ad) devreyi bulmak problemi. Burada 'grafik' ile kastedilen anlam, noktalardan veya 'köşelerden, oluşan sonlu bir koleksiyon olup, belirli sayıdaki nokta çiftleri, grafiğin 'kenarları' denilen çizgilerle birleştirilmiştir (Biz, burda, geometrik veya 'uzaklık' gibi özelliklerle değil yalnız hangi noktanın hangi noktaya bağlandığı ile ilgileniyoruz.



Şekil 4.14. Hamilton devresi (koyu çizgiler) içeren bir grafik. Şekilde, okuyucunun bulmak isteyebileceği bir tane daha Hamilton devresi var.

Bu nedenle, kenarların birbirinin üzerinden geçmesine aldırmadığımızı varsayarak, noktaların tümünün aynı düzlemde veya üç-boyutlu uzayda temsil edilip edilmedikleri gerçekten önemli değildir). *Hamilton devresi,* grafiğin kenarlarından oluşan ve her köşeden (verteksden) bir kez geçen bir ilmekten ibarettir. Hamilton devresini içeren böyle bir grafik örneği Şekil 4.14'de verilmektedir. Hamilton devresi problemi, verilen bir grafikte bir Hamilton devresinin varolup olmadığına karar vermek ve varsa bunu açıkça göstermektir.

Bir grafiği ikilik sayı sisteminde göstermenin çeşitli yolları vardır. Kullanılan yöntem pek önemli değildir. Örneğin noktaları 1, 2, 3, 4, 5... şeklinde numaralıyarak, uygun bir sırada çift çift dizmek olasıdır:

(1, 2), (1, 3), (2, 3), (1, 4), (2, 4), (3, 4), (1, 5), (2, 5), (3, 5), (4, 5),

(1, 6). ...

Bu listedeki her bir çift yerine, eğer çift grafiğin kenarını gösteriyorsa T, göstermiyorsa '0' koyarak '0'lar ve '1'lerden oluşan uyumlu başka bir liste yapabiliriz. Buna göre

10010110110...

dizisi, 1. köşenin 2. köşeye bunun 4. köşeye, 5. köşeye ... bağlandığını 3. köşenin, 4. köşeye, 5. köşeye... 4. köşenin, 5. köşeye... vb. bağlandığını gösterir (Şekil 4.14). İstenirse Hamilton devresi, kenarların bir alt-koleksiyonu olarak verilebilir ve bu durumda, çok daha fazla sayıda sıfıra sahip bir ikilik diziyle tanımlanabilir. Sağlama işlemi Hamilton devresini bulma sürecinden daha da hızlı gerçekleştirebilir. Bunun için, devrenin kenarlarının grafiğin orijinal kenarlarına ait olup olmadığını ve grafiğin her bir köşesinin, birisi iki kenarın her birinin ucunda olmak üzere tam iki kez kullanıldığını kontrol etmek suretiyle önerilen devrenin gerçekten bir devre olduğunu sınamak yeterlidir. Bu sağlama işlemi, polinom sürede kolayca gerçekleştirilebilir.

Gerçekte bu problem yalnız NP problemi olmayıp *tam-NP* olarak bilinen problemidir. Yani başka her NP problemi tam-NP problemine *polinom* sürede çevrilebilir. Böylece, Hamilton devresi problemini polinom sürede çözebilecek algoritmayı bulacak, başka bir deyişle, Hamilton devresi probleminin aslında P'de yer aldığını gösterecek kadar zeki olan birisi NP problemlerinin gerçekte P'de yer aldığını da anlayacaktır. Böyle bir durum çok önemli ipuçları verebilir. Genel olarak, P'deki problemler oldukça büyük *n* sayısı için, hızlı modern bir bilgisayarda 'kolay çözümlenir' (yani, 'makul bir sürede çözümlenebilir') olarak nitelenirken, P'de yer almayan NP problemleri, oldukça büyük *n* sayısı için, bilgisayarın önceden tahmin edilebilen işlem hızında ne ölçüde artış öngörülürse görülsün 'kolay çözümlenemez' (yani, 'ilke olarak çözümlenmesine karşın uygulamada çözümlenemez') olarak nitelenir (Büyük bir *n* sayısı için gerekli süre, P'de yer almayan NP problemi için öngörülen süreye kıyasla, evrenin yaşından daha uzun bir süreye dönüşür ki bu, pratik bir problem yönünden hiçbir işe yaramaz!). Hamilton devresi problemini polinom zamanda çözebilen herhangi bir akıllı algoritma, herhangi bir NP problemini de polinom zamanda çözebilen bir algoritmaya dönüştürebilir!

Tam-NP[15] kapsamında yer alan başka bir problem 'seyyar satıcı problemidir. Hamilton devresi problemini andıran bu problemde, her kenara verilmiş birer sayı vardır ve bu sayıların toplamının (satıcının seyahat ettiği 'uzaklık') minimum olması öngörülür. Seyyar satıcı probleminin polinom sürede çözümü de, diğer tüm NP problemlerinin polinom sürede çözümlenmesini sağlayacaktır (Bu problemin çözümü manşet haber olacaktır! Çünkü, özellikle, son yıllarda geliştirilen gizli şifre sistemleri, yine bir NP problemi olan büyük tam sayıların çarpanlarına ayrılması problemine dayanmaktadır. Bu problem polinom sürede çözülürse, bu gibi şifreler güçlü modern bilgisayarlar vasıtasıyla çözülebilir; problemin çözülememesi durumunda ise şifreler güvende olacaktır. Bkz. Gardner 1985),

Uzmanların çoğu, Turing makinesine benzer bir aygıt kullanılarak, bir tam-NP probleminin polinom sürede çözülmesinin gerçekte olanaksız olduğunu ve sonuçta P ve NP'in aynı olmadığına inanmaktadırlar. Haklı olabilirler ama bugüne değin bunu kanıtlayan olmadı. Bu problem bugün karmaşıklık teorisinin çözülememiş en önemli problemidir.

Fiziksel Nesnelerde Karmaşıklık ve Hesapedilebilirlik

Karmaşıklık teorisi, bu kitapta ele aldığımız konular yönünden önemlidir çünkü, nesnelerin algoritmik olup olmadığı sorusundan oldukça ayrı bir konuyu gündeme getirir: Algoritmik oldukları bilinen şeyler gerçekten yararlı bir şekilde algoritmik midir? Gelecek bölümlerde karmaşıklık

teorisinden çok hesaplanabilirlik teorisini tartışacağım. Çünkü, hesaplanabilirlik sorusunun aksine, karmaşıklık teorisi kapsamındaki konuların, ussal olguların odağını oluşturmadığını düşünmek eğilimindeyim (gerekçelerim oldukça yetersiz olsa bile). Üstelik, algoritmaların uygulanabilirliğine ilişkin sorulara, karmaşıklık teorisinin bugünkü durumuyla, ancak şöyle bir değinebileceği kanısındayım.

Ancak, karmaşıklığın önemi konusunda yanılıyor da olabilirim. Daha sonra değineceğim gibi (IX. Bölüm,) *gerçek fiziksel nesneler* ile ilgili karmaşıklık teorisi, şu âna kadar tartıştığımızdan çok farklı olabilir. Böyle bir olası farkı gözler önüne sermek için, kuantum mekaniğinin gizemli özelliklerinden bazılarını irdelememiz gerekecektir. Bazılarının önemi çok daha büyük boyutlara ulaşan birçok olguyu ve bu arada atomların ve moleküllerin davranışlarını inceleyen kuantum mekaniği gizemli ama o kadar da güçlü şekilde doğrudur. Bu teoriyle VI. Bölüm'de tanışacağız. David Deutsch'a, (1985) göre, bir 'kuantum bilgisayarı' inşa etmek 'ilke olarak' olasıdır. Deutsch, bu bilgisayar için P'de yer almayan ama bu aygıt sayesinde polinom sürede çözülebilen problemlerin (problem sınıfları) varolduğunu ileri sürmektedir. Bir kuantum bilgisayarı olarak davranabilen (güvenilir) gerçek bir fiziksel aygıtın nasıl inşa edilebileceği bugüne değin açıklığa kavuşturulmamıştır. Üstelik, söz konusu problemler sınıfı kuşkusuz yapaydır, ama fiziksel bir kuantum aygıtının bir Turing makinesi üzerine inşa edilmesine ilişkin teorik olasılık bize pek erişilmez gibi görünmüyor.

'Fiziksel bir aygıt' olduğu tartışmasına giriştiğim insan beyni, karmaşık olduğu kadar şaşırtıcı ölçüde özenli ve duyarlı tasarıma sahip beynimizin kendisi sakın, kuantum teorisinin gizeminden yararlanıyor olmasın? Kuantum sonuçlarının, problemlerin çözümünde veya yargıların oluşturulmasında yararlı bir şekilde kullanılabilecekleri yöntemlerin farkında mıyız acaba? Günümüzün kuantum teorisinin sınırlarını zorlayarak sağlayacağı üstünlüklerden yararlanmamız mümkün mü? Fiziksel aygıtların, Turing makinelerinin karmaşıklık teorisi yönünde geliştirilerek inşa edilmeleri gerçekten mümkün olabilir mi? Ya gerçek fiziksel aygıtlarla ilgili hesapedilebilirlik teorisinden ne haber?

Bu gibi soruları yanıtlayabilmek için tümüyle matematiksel olan konulardan uzaklaşarak, gelecek bölümlerde, fiziksel dünyanın gerçekte nasıl davrandığını araştıracağız.

Açıklama

[←1] Elemanları da küme olan kümeleri incelerken, kümenin elemanları ile yine aynı kümenin elemanlarının elemanlarını ayırt etmekte dikkatli olmalıyız. Örneğin S, belirli bir başka ***T*** kümesinin boş olmayan alt-kümelerinin bir kümesi olsun. Diyelim ki T'nin elemanları bir elma ile bir portakaldır. ***T*** kümesinin özelliği 'üçlük' değil 'ikilik' olmasına karşın S kümesi 'üçlük' özelliğine sahiptir, çünkü S kümesinin üç elemanı; yalnız bir elma içeren bir küme, yalnız bir portakal içeren bir küme, ve bir elma ile bir portakal içeren bir küme, yani toplam üç kümedir, bunlar ***S*** kümesinin üç elemanını oluşturur. Aynı şekilde, boş kümeden oluşan tek elemanlı kümenin özelliği de 'sıfırlık' değil 'birlik'dir, çünkü boş kümeden oluşan bir elemana sahiptir! Boş kümenin kendisi ise doğa) olarak sıfir elemana sahiptir.

[←2] Gerçekte, Gödel'in teoreminin uslamlaması, $\boldsymbol{P_k(k)}$ gibi önermelerin tamamen dışsal 'doğruluk' kavramına bağımlı olmayacakları biçimde sunulabilir. Ancak, yine de, bazı simgelerin gerçek 'anlamı'nın yorumuna bağlıdırlar: Özellikle, ... şartını sağlayan (doğal sayı) yoktur anlamını, taşıyan - ∃ örneğinde olduğu gibi.

[←3] Aşağıdaki örneklerde, küçük harfler doğal sayıları, büyük harfler ise doğal sayıların sonlu kümelerini temsil etmektedir; m → ***[(n, k, r)]*** şu ifadeyi göstersin: "X = (0,1,..., m) ise' k-elemanlı

alt-kümelerinden her biri r adet kutuya dağıtılırsa, en azından n-elemana sahip öyle bir 'büyük' Y alt-kümesi vardır ki, Y'nin tüm k-elemanlı alt-kümeleri aynı kutuya girer." Burada 'büyük'; ***Y*** kümesinin, Y'nin en küçük elemanı olan doğal sayıdan daha fazla sayıda elemana sahip olması anlamındadır.

Şu önermeyi ele alalım: '*k* ***r,*** ve ***n*** seçeneklerinden herhangi biri için öyle bir m_0 sayısı vardır ki, m_0'dan büyük tüm m'ler için, ***m*** → [***n, k,*** r] bildirimi daima doğrudur.' Bu önermenin, J. Paris ve L. Harrington (1977) tarafından, standart (Peano) aritmetik aksiyomları ile ilgili Gödel-tipi önermeye eşdeğer olduğu gösterilmiştir. Aksiyomlarla kanıtlanamayan fakat 'doğruluğu açıkça belli' bir önermedir (Aksiyomlardan türetilebilen önermelerin kendileri de doğrudur).

[←4] Yazının başlığı 'sıral sayılarına dayalı mantık sistemleri' olup, bazı okuyuculara, dipnotlarda kullanmakta olduğum Cantor'un sıral sayılar yazılımı yabancı gelmeyecektir. Yukarıda tanımladığım yöntemle elde edilen mantık sistemlerinin öncelik sırası, hesaplanabilir sıral sayılarıyla gösterilir.

Bazı matematik teoremleri oldukça doğaldır ve açıklanmaları kolaydır. Standart (Peano) aritmetik kurallarını kullanarak bunları ispatlamaya çalışılırsa, 'Gödelleştirme' yönteminin son derece büyük bir ölçüde (yukarıda özetlediğim sınırlarının çok ötesine çıkarak) kullanılması gerekir. Bu teoremlerin matematiksel kanıtları, normal matematiksel teoremin yöntemlerinin dışında gibi görünen herhangi bir belirsiz veya sorgulanabilir uslamlamaya bağımlı türden hiç değildir. Bkz. Smorynski (1983).

[←5] III. Bölüm, s 101'de değinilen süreklilik kuramı ($C = N_1$), burada karşılaştığımız en 'aşırı' matematik bildirimdir (yine de çok daha aşırı bildirimlerle çoğu kez karşılaşılabilir). Süreklilik varsayımı ilginçtir de, çünkü Gödel, Paul J. Cohen ile birlikte, bu varsayımının gerçekte, standart aksiyomlardan ve küme teorisinin yöntem kurallarından bağımsız olduğunu göstermiştir. Bu nedenle, süreklilik varsayımına yaklaşımımız, formalist ve Platonist görüşler arasında ayıran yapmamızı gerektirir. Bir formaliste göre bu varsayım 'karar verilemez' bir varsayımdır çünkü standart (Zermelo-Frankel) formel sistem kullanılarak ne kanıtlanabilir ne de çürütülebilir, ve 'doğru' veya 'yanlış' olarak nitelemek de 'anlamsızdır'. Ancak, iyi bir Platoniste göre bu varsayım, ya doğru veya yanlıştır ama hangisinin geçerli olduğunu kanıtlamak için yeni uslamlama yöntemlerine, Zermelo-Frankel formel sistemi için Gödel'in önermelerini kullanan yöntemlerin ötesinde yöntemlere ihtiyaç vardır (Cohen (1966), süreklilik varsayımını 'açıkça yanlış' olarak niteleyen bir düşünce ilkesi önermiştir).

[←6] Bu konuların daha canlı ve oldukça teknik-dışı anlatma için bkz. Rucker (1984).

[←7] Brouwer, topoloji ile ilgili 'Brouwer değişmez nokta teoremi' adındaki kendi teoreminin kanıtındaki 'inşa edilemezlik' endişesiyle bu düşünce çizgisine girmiş olabilir.

Teoreminin ana savı şöyledir: Bir diski, yani iç kısmıyla birlikte bir çemberi, alırsanız ve onu, başlangıçta yer aldığı bölgenin içinde sürekli hareket ettirirseniz, diskin en az bir noktası -sabit noktası- ilk başladığı yerde değişmeden kalacaktır. Bu noktanın tam olarak nerede bulunduğu veya buna benzer başka birçok noktanın bulunup bulunmayacağı hakkında bir fikriniz olmasa bile, Brouwer'in teoreminin savunduğu yalnızca böyle bir noktanın varolduğudur (Matematiksel varoluş teoremlerinin çoğu gibi bu teorem de aslında oldukça 'yapıcı' bir teorem. İnşa-edilmezlik teoremlerinin yanı sıra, 'Seçenek Aksiyomu' veya 'Zornlemması' olarak tanınan alt-teoremlere dayalı varoluş teoremleri vardır (bkz. Cohen 1966, Rucker 1984), Brouwer'in teoreminde karşılaşılan soru şu soruyla kıyaslanabilir: f, hem artı hem eksi değer alan bir reel değişkenin reel-değerlendirilmiş fonksiyonu ise, f'in nerede sıfır değerleri aldığını bulun. Normal yöntemde, fin işaret değiştirdiği ara

bölgenin tekrar tekrar ikiye ayrılması söz konusudur ama bu işlem, Brouwer'in öngördüğü anlamda, ara değerlerin artı, eksi veya sıfır olup olmadığına karar verecek ölçüde 'yapıcı' olmayabilir.

[←8] Kümeleri ***(v**, w, x, ... z)* olarak numaralarız; burada v, herhangi bir leksikografik sisteme göre f fonksiyonunu temsil eder. Her aşamada *f(w, x, ..., z) = 0* olup olmadığını kontrol ederiz ve kontrolümüz olumlu sonuç verirse ∃ *w, x, ... z [f(w, x, ... z) = 0]* önermesini alıkoyarız.

[←9] Geçenlerde Leonore Blum (bu kitabın ciltli ilk baskısındaki görüşlerimden etkilenmiş olarak) beni aradı ve Mandelbrot kümesinin (tümleyen küme) benim kitabın metninde savunduğum ve aşağıda 10. açıklamada değindiğim bağlamda, gerçekten yinelenemez olduğunu kanıtladığım bildirdi.

[←10] Reel sayıların, alışıldık reel-değerli fonksiyonların hesaplanabilirliği (doğal sayıların doğal sayı değerli fonksiyonlarından farklı olarak) ile ilgili yeni bir teori vardır. Blum, Shub ve Smale (1989) tarafından önerilen teoremin ayrıntıları hakkında henüz bilgi edindim. Bu teori kompleks-değerli fonksiyonlara da uygulanabilir olup, kitabın metninde ele alman konularla önemli bir ilişkisi olabilir.

[←11] Bu problem daha doğru olarak 'yarı-gruplar için sözcük problemi' olarak anılır. Seçme kurallarının biraz farklı olduğu başka sözcük problemleri vardır ama bunlar konumuzun dışındadır.

[←12] Hanf (1974) ve Myers (1974), düzlemi yalnız ***hesaplanamaz*** şekilde kaplayan tek bir kümenin (çok sayıda karodan oluşan) var olduğunu göstermişlerdir.

[←13] Aslında, biraz beceriyle, hâlâ P'de bulunan bir büyük ***n*** sayısı için ***n*** log ***n*** log log ***n*** gibi bir sıralamayla aşamaların sayısı azaltılabilir. Bkz. Khuth (1981).

[←14] Daha doğrusu, ***P, NP*** ve tam-NP sınıfları yalnız evet/hayır tipi problemler için tanımlanmıştır (örneğin, ***a, b*** ve ***c*** verildiğinde, ***a*** x b = ***c*** doğru mudur? gibi sorular); fakat, metinde verilen tanımlamalar, amacımız için yeterlidir.

[←15] Kesinlikle bir evet/hayır versiyonuna ihtiyacımız var: Örneğin, seyyar satıcı için şundan daha kısa bir yol var mıdır?

Notlar

[←I] Bir kümenin anlamı, bir bütün olarak ele alınabilen şeyler -fiziksel nesneler veya matematiksel kavramlar- topluluğudur. Matematikte, bir kümenin elemanları, çoğu kez, kümenin başka kümeleri oluşturacak şekilde bir araya gelebildiği için, kümelerin bizzat kendileridir. Böylece kümelerin kümeleri, kümelerin kümelerinin kümeleri, vs. şeklinde düşünülebilir.

[←II] Russell'ın paradoksunu basit ve eğlenceli bir örnekle açıklamak mümkün: Bir kütüphanede iki katalog bulunduğunu düşünün. Bunlardan bir tanesi, kütüphanedeki tüm kitapların, tabii bu arada kendilerinin de adlarını içerirken diğeri, kendilerinden başka tüm kitapların adını içeriyor. Hangi katalogun listesinde ikinci katalogun adı yer alır?

[←III] Fermat'ın teoreminin tamamı F'nin doğruluğu hâlâ bilinmemekle birlikte, G(0), G(1), G(2), G(3)... gibi tekil önermelerin doğrulukları yaklaşık G(125000)'e kadar bilinmektedir. Başka bir deyişle, 125000'ci kuvvetlerin karşılığı bildirime kadar hiçbir küp, artı küplerin toplamı olamaz, hiçbir dördüncü kuvvet, dördüncü kuvvetlerin toplamı olamaz, vb. (Kitabın yazıldığı tarih olan 1989'dan sonra Fermat teoreminin kanıtı verilmiştir. ç.n.)

[←IV] Leksikografik sıralamayı, k + 1 için, formel sistemin çeşitli simgelerini, asla kullanılmayan yeni bir 'sıfır'la birlikte kullanarak k+1 tabanında yazılan doğal sayıların normal sıralaması olarak düşünebiliriz (Yeni sıfırın asla kullanılmaması sorunu, sıfırla başlayan sayıların yeni sıfırın

atılmasıyla aynı olmasından kaynaklanır). Dokuz simgeli dizilerin basit leksikografik sıralaması, sıfırsız normal onlu sistemde yazılabilen doğal sayıların sıralanmasıdır: 1, 2, 3, 4, ... 8, 9, 11, 12,..., 19, 21, 22,..., 99, 111,112....., ...

[←V] 'Kümeler' ve 'sınıflar' arasında bir ayrım yapılmaktadır. Kümelerin biraraya gelerek diğer kümeleri veya sınıfları oluşturmasına izin verilirken sınıfların 'büyük' olmaları nedeniyle bu şekilde daha büyük koleksiyonları oluşturmadıkları kabul edilmektedir. Ancak, bir eleman koleksiyonunun ne zaman küme, ne zaman sınıf sayılacağı konusunda bir kural getirilememektedir.

[←VI] Sezgicilik, insan düşüncesini yansıttığı düşünüldüğü için bu adla anılır.

[←VII] Nitekim bu teoremin doğruluğu 1993'de İngiliz matematikçisi A. Wiles tarafından kanıtlanmış ve kanıtın doğruluğu genel kabul görmüştür (ç.n.)

[←VIII] Bu tür şanssız olasılıkların meydana gelmesine izin vermek gerekir; böylece herhangi bir algoritmik işlemi tanımlamak şansına sahip olabiliriz. Turing makinelerinin genel tanımını yapabilmek için, aslında hiç durmayan Turing makinelerine hoşgörüyle baktığımızı anımsayınız.

[←IX] Kanıt, duruncaya kadar çalışan makinenin hareketini yansıtacak aşamalar silsilesinden oluşabilirdi. Makine durur durmaz kanıt tamamlanmış olurdu.

[←X] Bu, daha önce değinilen Hilbert'in onuncu problemini, olumsuz olarak yanıtlar (Bkz. Devlin 1988). Burada, değişkenlerin adedi sınırlanmamıştır. Ancak, bu algoritmik olmayan özelliğin kanıtlanması için dokuz adetten fazlasına gerek olmadığı bilinmektedir.

[←XI] Aslında Hao Wang biraz daha farklı bir soru tasarlamıştı: Kare karolar dönmeden, birbirine uygun renkte kenarlara sahip olacaktı. Fakat bu fark bizim için burada önemli değil.

[←XII] Bir 'polinom' aslında $7n^4 - 3n^3 + 6n + 15$ gibi çok daha genel bir bildirimi gösterirse de, sayıların sabit katları bize daha fazla genelleştirme yapmak olanağı sağlıyor. Bu gibi bildirimlerde, n'in daha düşük kuvvetlerini kapsayan tüm terimler, n sayısı büyüdükçe önemsizleşir. Bu nedenle, örnekte $7n^4$ dışındaki terimleri dikkate almayabiliriz.