BİÇİMSEL DİLLER VE OTOMATA

Biçimsel Diller ve Otomata

Ders notları ve alıştırmalar

0 Dizgiler, diller ve makineler

1 Matematiksel araçlar ve ispat yöntemleri

2 Deterministik sonlu otomat

3 Deterministik olmayan sonlu otomat

4 NFA'dan DFA'ya: eşdeğerlik ve küçültme

5 Düzenli ifadeler

6 Pompalama önsavı ve kapalılık

7 Bağlamdan bağımsız gramerler

8 Yığıtlı otomatlar

9 Turing makineleri

10 Karar verilebilirlik

11 P, NP ve NP-tamlık

HEKZON

hekzon.com/dersler

12 bölüm · 33 konu

İçindekiler

Bu notlar, YouTube'daki Biçimsel Diller ve Otomata dersinin yazılı hâlidir. Her konunun yanındaki zaman, uzun videoda o konunun başladığı yeri gösterir. Her bölümün sonunda sorular var; cevaplarını ve çözümlerini en sonda bulabilirsin. Aynı soruları hekzon.com/dersler adresinde, cevabını hemen göreceğin alıştırmalar olarak da çözebilirsin.

0 **Dizgiler, diller ve makineler** 3
Çay otomatı · İki soru

1 **Matematiksel araçlar ve ispat yöntemleri** 8
On üç kişi · Kırk asal · İki sonsuzluk

2 **Deterministik sonlu otomat** 15
Sayıyı saklamadan · İki şart · Üç kelime

3 **Deterministik olmayan sonlu otomat** 22
Sonu bilinmeyen dizgi · Başında sıfır olmayan sayı

4 **NFA'dan DFA'ya: eşdeğerlik ve küçültme** 28
Dört durum, sekiz durum · İki makine, tek kural

5 **Düzenli ifadeler** 34
Tek satırlık kural · Kalıptan makineye · Makineden kurala

6 **Pompalama önsavı ve kapalılık** 42
Basit bir kural · Sonu olmayan makineler · İki kural

7 **Bağlamdan bağımsız gramerler** 50
Kedi mektubu yazdı · Sayamayan makine · Bir else, iki if

8 **Yığıtlı otomatlar** 58
Sayamayan makine · Tersten de aynı · Üç sayı

9 **Turing makineleri** 66
Toplama devresi yok · Üç eşit sayı · Bir telefon, her iş

10 **Karar verilebilirlik** 74
Üç kart · Dört satırlık program · Kusursuz tarayıcı

11 **P, NP ve NP-tamlık** 82
Saniye mi, adım mı? · Dört kapalı kare · Davet bulmacası

Cevaplar ve çözümler 90

BİÇİMSEL DİLLER VE OTOMATA

0 Dizgiler, diller ve makineler

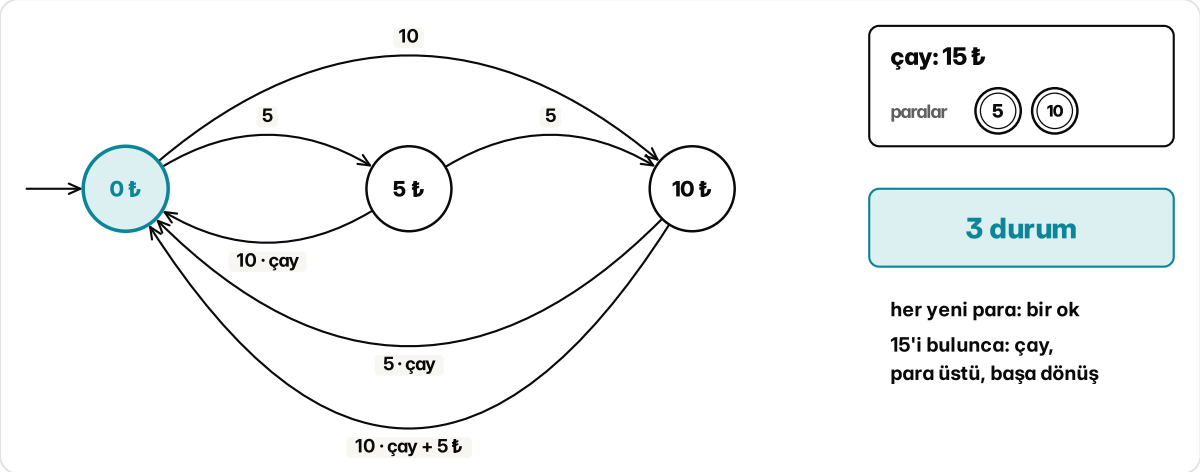
Bu bölümde dersin üç sorusunu, makinelere girdi olan dizgileri ve dil kavramını tanıyoruz. Evet-hayır biçimindeki her sorunun neden "bir dizgi bir dilin içinde mi?" sorusuna döndüğünü de göreceğiz.

Çay otomatı

videoda 0:11

Çay otomatına önce 5 lira, sonra 10 lira attığımızı düşünelim; çay geldi. Makine bu kararı verirken neyi hatırlamak zorundaydı: attığımız paraların listesini mi, yoksa çok daha azını mı?

Çay 15 lira olsun, makine de beşlik ve onluk alsın. Hangi parayı hangi sırayla attığımız önemli değil, yalnız toplam önemli: iki beşlik de bir onluk da makineyi aynı yere, 10 liraya getirir. Demek ki üç hâl yeter: hiç para yok, 5 lira var, 10 lira var. Bu hâllerin her birine **durum** denir. Her yeni parada makine bir durumdan ötekine geçer; toplam 15'i bulunca ya da geçince çay verir, fazlasını geri öder ve başa döner.



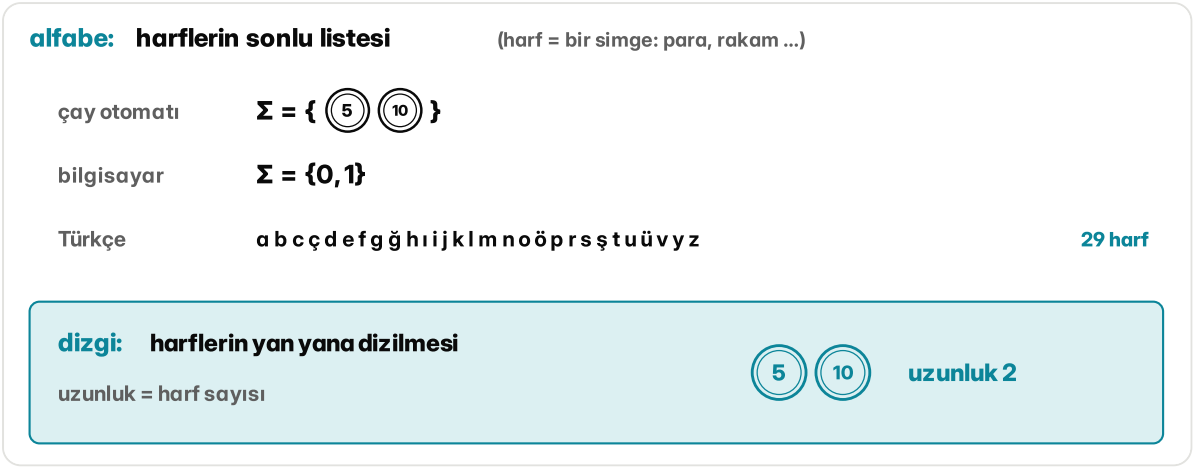
Şekil 0.1 Çay otomatının üç durumu (0 TL, 5 TL, 10 TL) ve aralarındaki altı geçiş oku.

Deneyelim. Beşlik makineyi baştan 5 liraya geçirir, ardından gelen onluk toplamı 15'e çıkarır; çay gelir, makine yine başta. İki onluk atarsak önce 10 lira olur, sonra çay ve 5 lira para üstü gelir. Makine gün boyu 100 çay verse de bütün belleği tek bir şeydir: o an hangi durumda olduğu.

Otomat (automaton), kendi kendine işleyen makine demektir; çay makinesi de üç durumlu bir otomattır. Ders böyle sade makinelerle üç soru sorar: hangi makine neyi yapabilir (otomatlar), hangi problemi hiçbir makine çözemez (hesaplanabilirlik), çözülebilenlerden hangisi makul sürede çözülür (karmaşıklık)? Çay otomatı kadar sade makineden başlayıp belleği adım adım büyüyen makinelerle, bugünkü bilgisayarın yapabildiği her şeyi yapan makineye çıkacağız.

Makineye giren her şey bir **alfabe** (alphabet) ile yazılır. Alfabe, kullanacağımız simgelerin sonlu listesidir; simge bir para da olabilir, bir rakam da. Çay otomatının alfabesi Σ = {beşlik, onluk}, bilgisayarda en sık kullanılan alfabe {0, 1}, Türkçenin 29 harfi de bir alfabedir. Simgelerin yan

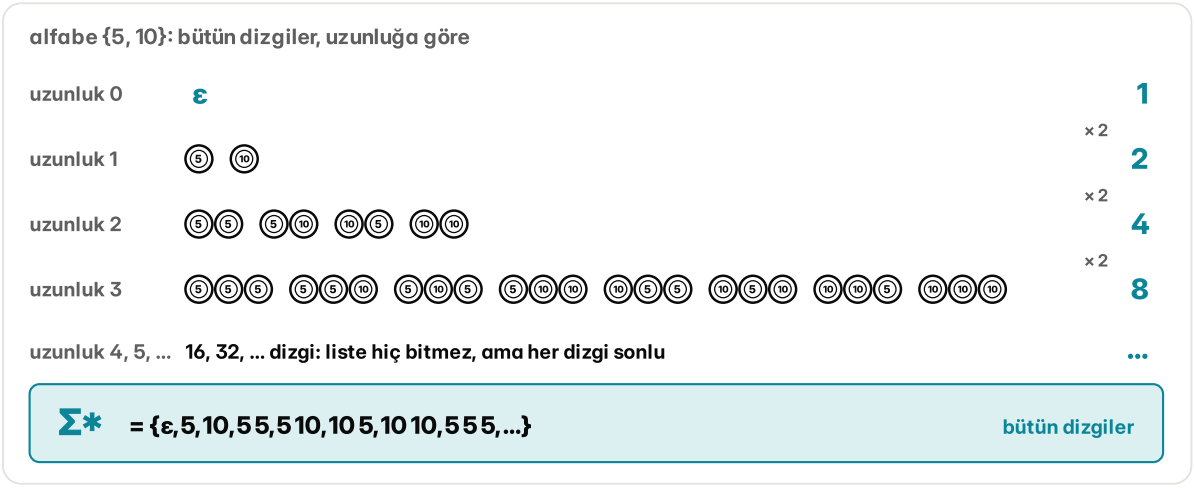
yana dizilmesine **dizgi** (string) denir; simge sayısı dizginin uzunluğudur. "Beşlik, onluk" uzunluğu 2 olan bir dizgidir.



Şekil 0.2 Çay otomatının alfabesi, 0 ve 1 alfabesi, Türkçenin 29 harfi ve uzunluğu 2 olan bir dizgi.

Hiç para atmayanın da bir dizgisi var: hiç simgesi olmayan, uzunluğu 0 olan **boş dizgi**; ε ile gösterilir. İki dizgi uç uca eklenince yeni bir dizgi çıkar, buna **art arda yazma** denir. "Beşlik, beşlik" ile "onluk" art arda yazılınca "beşlik, beşlik, onluk" olur; uzunluklar da toplanır: 2 + 1 = 3.

Bu alfabenin bütün dizgilerini uzunluklarına göre sayalım: 1 boş dizgi, 2 tek simgeli, 4 iki simgeli, 8 üç simgeli dizgi. Her uzunlukta sayı iki katına çıkar; liste hiç bitmez, ama içindeki her dizgi sonludur. Alfabenin bütün dizgilerinin bu sonsuz kümesi Σ∗ ile gösterilir.



Şekil 0.3 Uzunluğa göre dizgi sayıları 1, 2, 4, 8, ... ve alfabenin bütün dizgileri Σ∗.

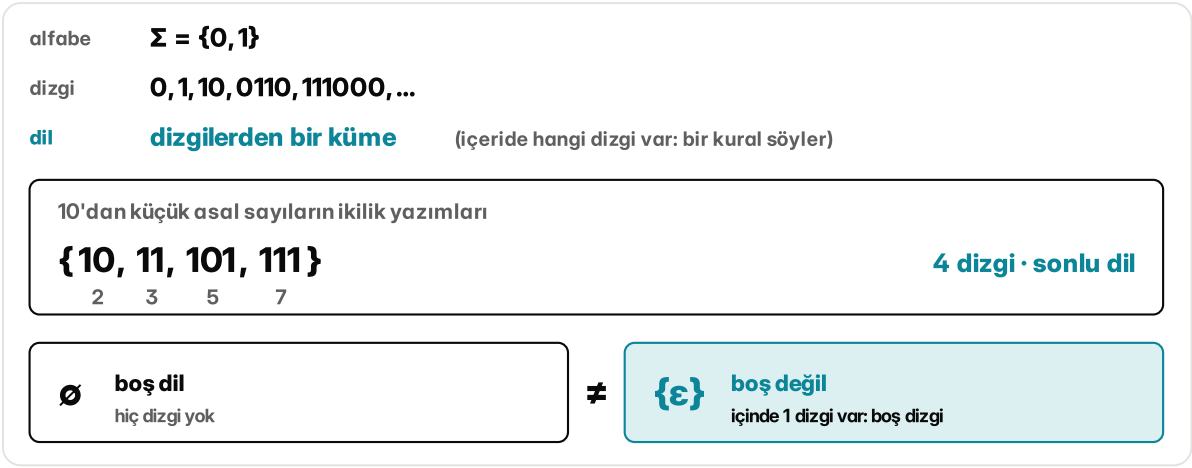
Çay otomatının alfabesi iki para, bilgisayarınki 0 ve 1. Bir mesaj da bir fotoğraf da sonunda 0 ve 1'lerden oluşan uzun bir dizgidir.

İki soru

videoda 2:39

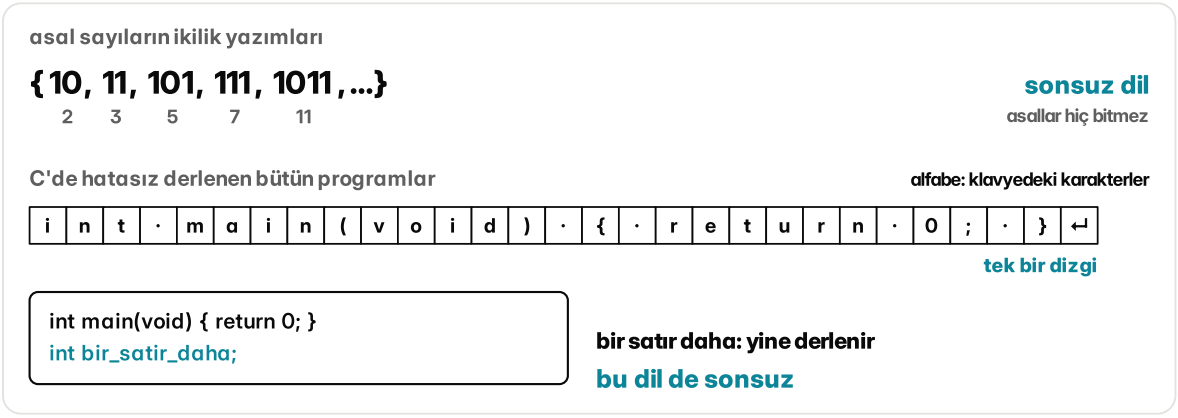
"17 asal mı?" ile "Bu program hatasız derlenir mi?" sorularından biri matematiğin, öteki yazılımın sorusudur; ama bir makine için ikisi aynı türden sorudur. Nedenini görmek için önce **dil** kavramına bakalım.

Alfabe olarak {0, 1} seçelim. Bu simgelerle yazılan dizgilerden oluşan her kümeye dil denir. Türkçe ya da C gibi bir dil olması gerekmez; hangi dizginin içeride olduğunu söyleyen bir kural yeter. Örneğin 10'dan küçük asal sayıların ikilik yazımları 10, 11, 101 ve 111'dir (2, 3, 5, 7): 4 dizgilik, **sonlu** bir dil. Hiç dizgisi olmayan dile **boş dil** (ø) denir. Yalnız boş dizgiyi içeren {ε} dili ise boş değildir; içinde bir dizgi vardır.



Şekil 0.4 10'dan küçük asalların ikilik yazımlarından oluşan sonlu dil, boş dil ø ve {ε}.

Sınırı kaldırınca dil **sonsuz** olur. Bütün asal sayıların ikilik yazımları 10, 11, 101, 111, 1011, ... diye sürer; asallar hiç bitmediği için bu liste de bitmez. C dilinde hatasız derlenen bütün programlar da sonsuz bir dildir: alfabe klavyedeki karakterlerdir, her program boşlukları ve satır sonlarıyla birlikte uzun bir dizgidir. Hatasız bir programa her zaman bir satır daha eklenebildiği için bu dil de bitmez.

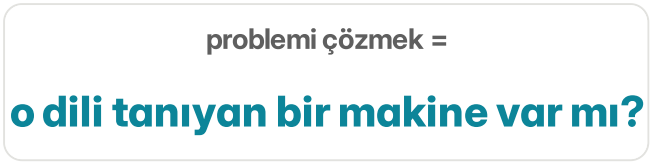


Şekil 0.5 Asal sayıların ikilik yazımlarının sonsuz dili ve C programlarının sonsuz dili.

Şimdi sorulara dönelim. 17'nin ikilik yazımı 10001'dir. "17 asal mı?" sorusu şuna dönüşür: bu dizgi asal sayılar dilinde mi? Cevap evet. 15'in yazımı 1111 bu dilde yoktur, çünkü 15 = 3 × 5. Program sorusu da aynıdır: bu metin, hatasız derlenen programlar dilinde mi? Girdi bir sayı, bir program ya da bir harita olabilir; bilgisayarda hepsi bir dizgi olarak yazılır. Böylece her evet-hayır sorusu bir dile üyelik sorusuna döner; o dil, cevabı evet olan bütün girdilerin kümesidir.

Böyle bir soruyu cevaplayan makine dizgiyi okur ve sonunda evet ya da hayır der. Evet dediği dizgiyi **kabul** eder, hayır dediğini **reddeder**; kabul ettiği bütün dizgilerin kümesine de makinenin **tanıdığı dil** denir. Sonlu bir dili tanımak kolaydır: makine 4 dizgilik listeyi ezbere bilir, 101 gelince

kabul eder, 1000 gelince reddeder. Asıl zorluk sonsuz dillerdedir: sonsuz bir liste ezberlenemez, makine sonlu bir kuralla sonsuz sayıda dizgiye doğru cevap vermek zorundadır. Bir problemi çözmek böylece tek bir soruya iner: o dili tanıyan bir makine var mı?



Şekil 0.6 Dizgiyi okuyup kabul ya da ret çıkaran makine; makinenin dili kabul ettiği dizgilerdir.

Derleyici de bu soruyu cevaplar: bu metin, hatasız derlenen programlar dilinde mi? Cevap hayırsa makine kodu çıkmaz, ekrana derleme hatası düşer.

Özet

- Çay otomatı gibi sade bir makine, geçmişi değil yalnız o anki durumunu hatırlar; durum sayısı sonludur.
- Alfabe simgelerin sonlu listesi, dizgi simgelerin dizisi, uzunluk simge sayısıdır; boş dizginin (ε) uzunluğu 0'dır.
- İki dizgi art arda yazılınca uzunluklar toplanır; bir alfabenin bütün dizgileri (Σ^*) sonsuzdur, ama her biri sonludur.
- Dil, dizgilerden oluşan bir kümedir; sonlu ya da sonsuz olabilir. Boş dil ø ile {ε} birbirinden farklıdır.
- Her evet-hayır sorusu, girdinin cevabı evet olanlar dilinde olup olmadığı sorusuna döner.
- Bir dili tanıyan makine, dizgileri kabul ya da reddeder; sonsuz bir dil için makinenin sonlu bir kuralı olmalıdır.

Sorular

1 **Alfabesi {0, 1} olan 0110 dizgisinin uzunluğu kaçtır?**

A 2 **B** 3
C 4 **D** 6

2 **Çay 15 lira; makine beşlik ve onluk alıyor ve 0 TL durumunda bekliyor. Sırayla bir beşlik ve bir beşlik daha atılırsa makine hangi durumda olur?**

A 5 TL **B** 10 TL
C 15 TL **D** 0 TL

3 **Aynı çay makinesi 0 TL durumunda bekliyor. Sırayla üç beşlik atılıyor. Çay verildikten sonra makine kaç TL durumundadır? Yalnız sayıyı yaz.**

Cevap: _____

4 **x = 01 ve y = 110 dizgileri veriliyor. Art arda yazılmış xy dizgisi hangisidir?**

A 11001
B 0110
C 10110
D 01110

5 **Σ = {0, 1} alfabesinde uzunluğu en çok 3 olan kaç dizgi vardır? Boş dizgiyi de say.**

Cevap: _____

6 **{ε} dili için hangisi doğrudur?**

A Bir dizgisi (boş dizgi) vardır, boş dil değildir.
B Boş dilin başka bir yazımıdır.
C Hiç dizgisi yoktur.
D Dil sayılmaz, çünkü içinde harf yok.

7 **Aşağıdaki dillerden hangisi sonludur?**

A Bütün asal sayıların ikilik yazımları
B C'de hatasız derlenen bütün programlar
C {0, 1} alfabesinin bütün dizgileri
D 10'dan küçük asal sayıların ikilik yazımları

8 **21 sayısının ikilik yazımı 10101'dir. Asal sayıların ikilik yazımlarından oluşan dilde bu dizgi var mıdır?**

A Evet, çünkü 10101 1 ile başlar ve 1 ile biter.
B Hayır, çünkü 21 = 3 × 7, asal değildir.
C Evet, çünkü 21 tek sayıdır.
D Hayır, çünkü dizginin uzunluğu tektir.

BİÇİMSEL DİLLER VE OTOMATA

1 Matematiksel araçlar ve ispat yöntemleri

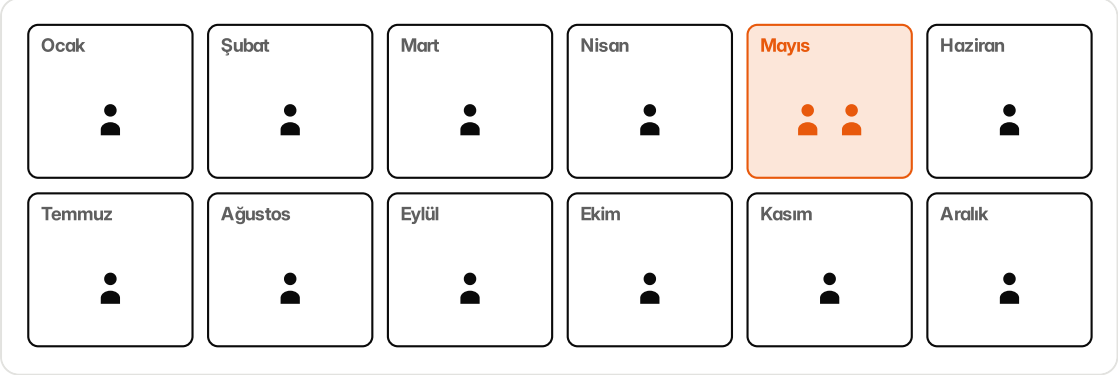
Bu bölümde dersin üç temel aracını kuruyoruz: güvercin yuvası ilkesi ve çelişkiyle ispat, karşı örnek ve doğrudan ispat ile tümevarım, dillerde küme işlemleri ve Cantor'un köşegen yöntemi. Sonunda bilgisayarın hiçbir zaman cevaplayamayacağı sorular olduğunu sayarak göstereceğiz.

On üç kişi

videoda 4:50

Bir odada 13 kişi var. Kimseye sormadan, ikisinin aynı ayda doğduğunu kesin olarak nasıl bilebiliriz? Tahmin değil, kesin bilgi istiyoruz.

Yılda 12 ay var; her ayı bir kutu gibi düşünelim ve herkesi doğduğu ayın kutusuna koyalım. İki kişiyi aynı kutuya düşürmemek için elimizden geleni yapalım: ilk 12 kişi 12 ayrı aya dağılsın. Artık her kutuda bir kişi var. 13. kişi hangi ayda doğmuş olursa olsun, o kutuda zaten biri bekliyor.

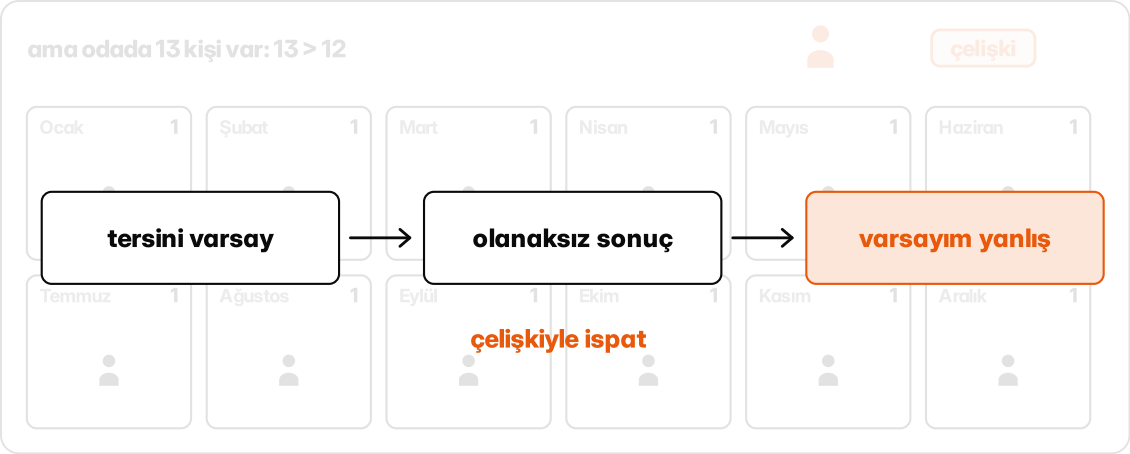


Şekil 1.1 On iki ay kutusuna dağılan ilk 12 kişi ve zaten dolu bir kutuya giren 13. kişi.

Buna **güvercin yuvası ilkesi** denir: kişiler güvercin, aylar yuva olsun; güvercinler yuvalardan fazlaysa bir yuvada mutlaka birden çok güvercin olur. Aynı yol daha fazlasını da söyler: 100 kişilik bir salonda aynı ayda doğmuş en az 9 kişi vardır. Her aya 8 kişi koysak 96 kişi eder; kalan 4 kişi nereye girerse girsin, o ay 9. kişisini bulur.

Bunu deneyerek bilemezdik. Her kişinin 12 olası ayı var; 13 kişide 12, 13 kez çarpılır ve 100 trilyonu aşkın (106.993.205.379.072) farklı dağılım çıkar. Hepsini tek tek denemek imkânsızdır. Akıl yürütme ise hepsini birden kapsar. Bir önermenin olası her durumda doğru olduğunu gösteren akıl yürütmeye **ispat** (proof) denir.

Aynı sonuca bir yoldan daha varılır. Tersini varsayalım: 13 kişinin hepsi ayrı aylarda doğmuş olsun. O zaman her ayda en çok bir kişi vardır ve odada en çok 12 kişi olabilir. Oysa odada 13 kişi var; varsayım gerçekle çelişti, demek ki yanlış. Sonucun tersini varsayıp olanaksız bir şeye varan bu yola **çelişkiyle ispat** (proof by contradiction) denir.



Şekil 1.2 Varsayım "herkes ayrı ayda", sonuç "en çok 12 kişi", gerçek "13 kişi": çelişki.

Bu yolun en ünlü örneğini 2.000 yılı aşkın süre önce Öklid yazdı: en büyük asal sayı yoktur. Tersini varsayalım: asallar bir yerde bitiyor. Diyelim ki bütün asallar 2, 3, 5, 7, 11 ve 13. Hepsini çarpıp 1 ekleyelim: 2 × 3 × 5 × 7 × 11 × 13 = 30.030, artı 1 ile 30.031. Bu sayı listedeki her asalın bir katından 1 fazladır; hangisine bölersek bölelim kalan 1 çıkar. Oysa 1'den büyük her sayının bir asal böleni vardır. 30.031'in kendisi asal bile değil: 59 × 509. Ama ikisi de listede yok, demek ki liste eksikmiş. Hangi listeyle başlarsak başlayalım, listenin dışında yeni bir asal çıkar; asalların sonu yoktur.



Şekil 1.3 Varsayılan asal listesi, 30.031 ve kalan 1 tablosu; 59 ve 509 listede yok.

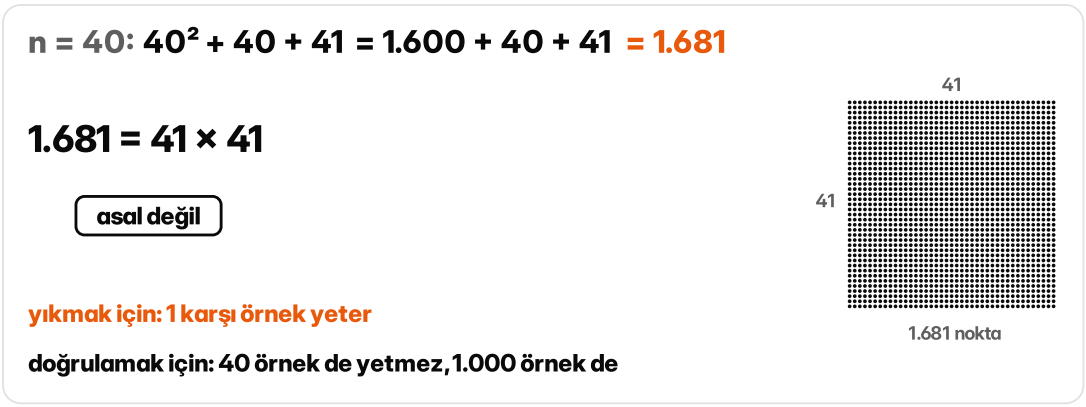
2024'te bulunan rekor asal sayının 41 milyonu aşkın basamağı var. Öklid'in ispatı, ondan da büyük bir asal olduğunu kimse bulmadan söylüyor.

Kırk asal

videoda 6:56

Bir sayının karesini alalım, sayının kendisini ve 41'i ekleyelim: $n^2 + n + 41$. n = 0 için 41, 1 için 43, 2 için 47, 3 için 53 çıkar; hepsi asal. 0'dan 39'a kadar 40 sayının 40'ında da asal çıkıyor. Bu formül her sayıda asal mı verir?

40'la deneyelim: 40'ın karesi 1.600, artı 40, artı 41 eder: 1.681. Bu sayı 41 × 41'dir; asal değil. Genel bir iddiayı yıkmak için tek bir **karşı örnek** (counterexample) yeter; doğrulamak için ise 40 örnek de yetmez, 1.000 örnek de.



Şekil 1.4 n = 40 satırı: 1.600 + 40 + 41 = 1.681 = 41 × 41, asal değil.

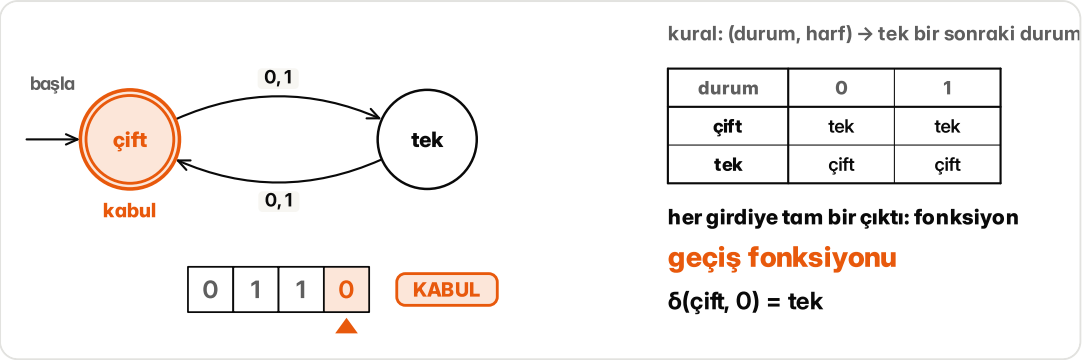
Peki bu formül için her sayıda doğru olan bir şey söyleyebilir miyiz? Bir sayının karesine kendisini eklemek, onu bir fazlasıyla çarpmak demektir: $n^2 + n = n \times (n + 1)$. Örneğin $7^2 + 7 = 49 + 7 = 56$ ve 7×8 de 56. Ardışık iki sayıdan biri mutlaka çifttir, o yüzden bu çarpım hep çift çıkar. Çifte 41 eklenince sonuç tek olur: formül hiçbir sayıda çift sayı vermez. Bilinen gerçeklerden adım adım sonuca giden bu yola **doğrudan ispat** denir.

Bazı iddialar sonsuz sayıda şey hakkındadır; mesela bütün dizgiler hakkında. Onlar için domino taşlarını düşünelim: birinci taş düşerse ve her taş düşerken bir sonrakini deviriyorsa, sıradaki bütün taşlar düşer. Buna **tümevarım** (induction) denir: önce başlangıçta doğru olduğunu, sonra her adımın doğruluğu bir sonrakine taşıdığını gösteririz.



Şekil 1.5 Domino taşları: başlangıçta ilk taş devrilir, her taş bir sonrakini devirir.

Şimdi 0 ve 1'lerden oluşan bir dizgiyi soldan sağa harf harf okuyan bir makine düşünelim; iki durumu var, çift ve tek. Çift durumunda başlar, okuduğu her harfte öteki duruma geçer; okuma çiftte biterse dizgiyi kabul eder. 0110 dizgisinde sırayla tek, çift, tek, çift olur; çiftte biter, dizgi kabul. Makinenin kuralı, her durum ve harf çiftine tek bir sonraki durum verir. Her girdiye tam bir çıktı veren böyle bir kurala fonksiyon denir; makinedeki adı **geçiş fonksiyonu**dur.



Şekil 1.6 Çift ve tek durumlu makine, 0110 dizgisinin okunuşu ve 2 × 2 geçiş tablosu.

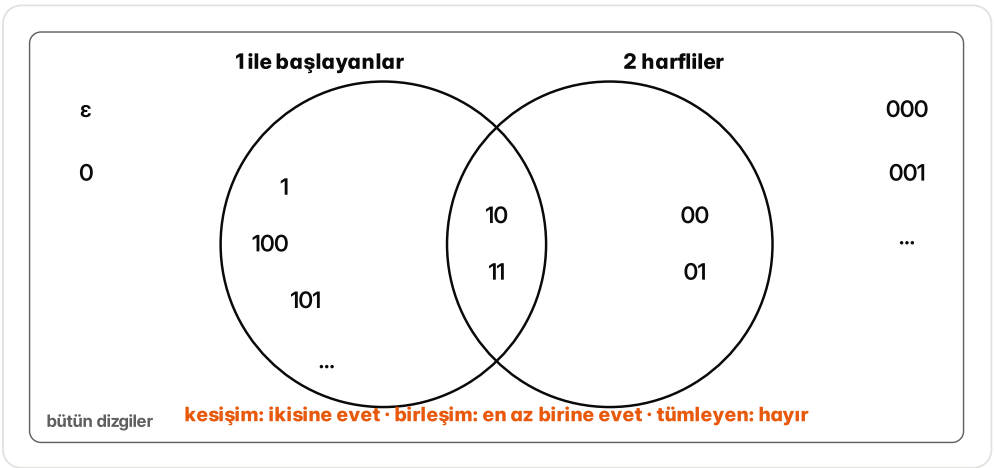
İddiamız: bu makine uzunluğu çift olan dizgileri kabul eder, ötekileri reddeder. Uzunluğu 12'ye kadar 8.191 dizginin hepsini denesek bile bu bir ispat olmaz; dizgilerin sonu yok. Başlangıçta boş dizginin uzunluğu 0, yani çifttir; makine de çiftte başlar. Adımda, makine bir dizginin sonunda doğru durumdaysa, bir harf daha okununca uzunluk çiftten teke ya da tekten çifte geçer; makine de öteki duruma geçer ve yine doğru durumdadır. Birinci taş düştü, her taş sonrakini deviriyor: iddia bütün dizgiler için doğrudur. 40 örnek bir kuralı kanıtlamaya yetmemişti; iki adımlık bir akıl yürütme sonsuz sayıda dizgiye yetti.

İki sonsuzluk

videoda 9:18

Yazılabilecek programların sonu yok; sorulabilecek evet-hayır sorularının da. Her soruya bir program yetişir mi, yoksa bu iki sonsuzluktan biri ötekinden büyük mü?

Bir evet-hayır sorusu, cevabı evet olan dizgilerin kümesiyle anlatılır; böyle bir kümeye dil denir. Diller küme olduğu için küme işlemleri uygulanır. İki dil alalım: 1 ile başlayanlar ve 2 harfliler. **Kesişim**, ikisinde birden olanlardır: 10 ve 11. **Birleşim**, en az birinde olanlardır: 1 ile başlayanların hepsi, artı 00 ve 01. **Fark**, 2 harflilerde olup 1 ile başlamayanlardır: yine 00 ve 01. **Tümleyen**, dilde olmayanların hepsidir; 1 ile başlayanların tümleyeni boş dizgi ve 0 ile başlayan bütün dizgilerdir. Sorularla söylersek kesişim ikisine de evet, birleşim en az birine evet, tümleyen hayır demektir.



Şekil 1.7 "1 ile başlayanlar" ve "2 harfliler" dillerinin kesişimi, birleşimi, farkı ve tümleyeni.

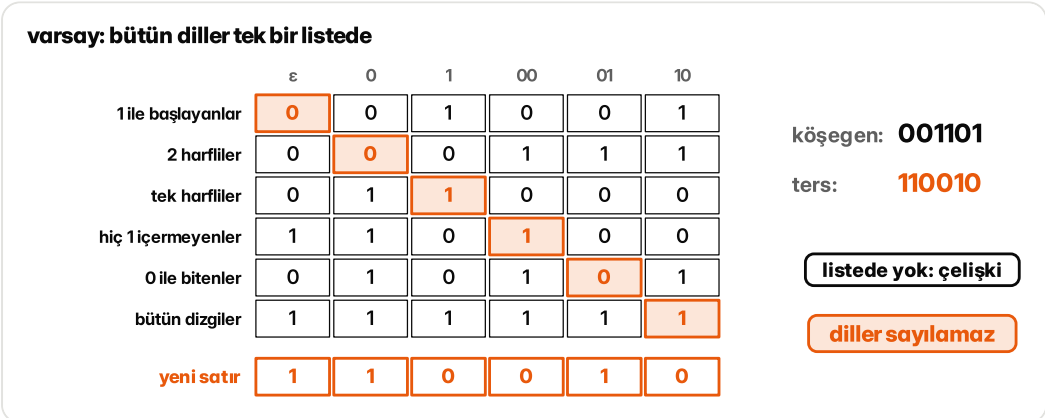
Sonsuz bir kümenin elemanlarına hiçbirini atlamadan 1., 2., 3. diye sıra verebiliyorsak, o kümeye **sayılabilir** denir. Bütün dizgiler sayılabilir: kısadan uzuna, aynı uzunluktakileri de sayı gibi küçükten büyüğe dizeriz. Boş dizgi 1., 0 2., 1 3., 00 4., 01 5., 10 6. olur. Her uzunlukta sonlu sayıda dizgi olduğu için liste her dizgiye bir gün mutlaka varır. Bir programın kodu da sonlu bir dizgidir; aynı sırayla bütün programlar da tek bir listeye dizilir.



Şekil 1.8 Dizgilerin kısadan uzuna sıralı listesi; 10 altıncı sırada.

Bir dili bu sıranın altına yazabiliriz: dizgi dilde varsa 1, yoksa 0. "1 ile başlayanlar" dili 0010011 diye başlayan, sonu gelmeyen bir satırdır. Her dil böyle bir satır, her satır da bir dildir.

Şimdi diyelim ki biri bütün dilleri, yani bütün satırları tek bir listeye dizdi. Cantor'un 1891'de gösterdiği yol şudur: birinci satırın birinci hanesini, ikincinin ikincisini, üçüncünün üçüncüsünü alalım; tablonun köşegeni boyunca inelim. Altı satırlık örnek tabloda köşegen 001101'dir. Her haneyi ters çevirirsek 110010 çıkar. Bu yeni satır birinci satırdan birinci hanesinde, ikinciden ikinci hanesinde ayrılır; böylece listedeki her satırdan en az bir hanede ayrılır ve listede yoktur. Oysa liste tamdı; varsayım çelişti. Hiçbir liste bütün satırları tutamaz: diller sayılamaz.



Şekil 1.9 Köşegen 001101 ve her haneyi ters çevirerek kurulan 110010 satırı; hiçbir satırla aynı değil.

Bir program, verilen dizginin dilde olup olmadığını söylüyorsa o dili tanır; her program en çok bir dil tanır. Programların listesini izleyerek tanıdıkları dilleri de bir listeye dizebiliriz; ama bütün diller hiçbir listeye sığmıyor. Demek ki hiçbir programın tanıyamadığı diller var. Sonsuz sayıda program yazılabilir, ama sorular daha büyük bir sonsuzluk oluşturur: bilgisayarın hiçbir zaman cevaplayamayacağı evet-hayır soruları, cevaplayabildiklerinden çok daha fazladır.

Özet

- Güvercin yuvası ilkesi: güvercinler yuvalardan fazlaysa bir yuvada birden çok güvercin olur; 13 kişi, 12 ay.
- Çelişkiyle ispatta sonucun tersi varsayılır ve olanaksız bir şeye varılır; Öklid'in ispatı asalların bitmediğini böyle gösterir.
- Genel bir iddiayı yıkmak için tek karşı örnek yeter; doğrulamak için örnek sayısı yetmez, ispat gerekir. $n^2 + n + 41$'in ilk karşı örneği $n = 40$'tır.
- Doğrudan ispat bilinen gerçeklerden adım adım sonuca gider; tümevarım başlangıçtan ve her adımın bir sonrakine taşınmasından oluşur.
- Diller küme olduğu için birleşim, kesişim, fark ve tümleyen alınabilir; dizgiler ve programlar sayılabilir.
- Köşegen yöntemiyle diller sayılamaz: hiçbir program listesi bütün dilleri tanıyamaz, cevapsız sorular vardır.

Sorular

1 **Bir yılda 12 ay var. Aynı ayda doğmuş iki kişinin bulunması en az kaç kişide kesin olur?**

A 12
B 13
C 24
D 6

2 **25 öğrencilik bir sınıfta haftanın 7 gününden her biri bir doğum günü kutusu sayılıyor. Aynı günde doğmuş en az kaç öğrenci vardır? Yalnız sayıyı yaz.**

Cevap: _____

3 **Bir arkadaşımız "$n^2 + n + 41$ her n için asaldır" diyor. Hangisi bu iddiayı kesin olarak çürütür?**

A $n = 0$ için sonucun 41 olması
B $n = 39$ için sonucun 1.601 olması
C Formülün hiçbir çift sayı vermemesi
D $n = 40$ için sonucun $1.681 = 41 \times 41$ olması

4 **$n^2 + n + 41$ ifadesi neden hiçbir n için çift sayı vermez?**

A $n = 0$'dan 39'a kadar 40 değerin hepsi tek çıktığı için
B Çünkü sonuç her zaman asaldır ve asallar tektir
C $n^2 + n = n \times (n + 1)$ ardışık iki sayının çarpımıdır, hep çifttir; çift + 41 tek olur
D Çünkü n^2 her zaman çifttir

5 **Uzunluğu çift olan dizgileri kabul eden iki durumlu makine (çift ve tek) çift durumunda başlar ve her harfte öteki duruma geçer. 10111 dizgisini okuyunca makine nerede durur ve ne yapar?**

A tek durumunda, kabul eder

B çift durumunda, kabul eder

C çift durumunda, reddeder

D tek durumunda, reddeder

6 **Cantor'un köşegen yöntemiyle hangi sonuç gösterilir?**

A Bütün diller hiçbir listeye sığmaz; diller sayılamaz.

B Bütün programlar hiçbir listeye sığmaz.

C Bütün dizgiler hiçbir listeye sığmaz.

D Asal sayılar bir yerde biter.

7 **Dört satırlık bir tabloda köşegen 1010 çıkıyor. Cantor'un yöntemiyle her hane ters çevrilerek kurulan yeni satırın ilk dört hanesini yaz.**

Cevap: _____

8 **Dizgiler kısadan uzuna, aynı uzunluktakiler küçükten büyüğe dizildiğinde boş dizgi 1., 0 2., 1 3., 00 4., 01 5., 10 6., 11 7., 000 8. olur. 001 dizgisi kaçıncı sıradadır? Yalnız sayıyı yaz.**

Cevap: _____

BİÇİMSEL DİLLER VE OTOMATA

2 Deterministik sonlu otomat

Bu bölümde sayıyı ya da kelimeyi saklamadan yalnız o anki durumla karar veren makineyi, yani deterministik sonlu otomatı kuruyoruz. Üç örnek göreceğiz: 3'e bölünebilirlik, şifre denetimi ve Türkçe büyük ünlü uyumu. Sonunda makineyi beş parçayla tanımlayacak ve tuzak durumunun neden gerektiğini anlayacağız.

Sayıyı saklamadan

videoda 11:51

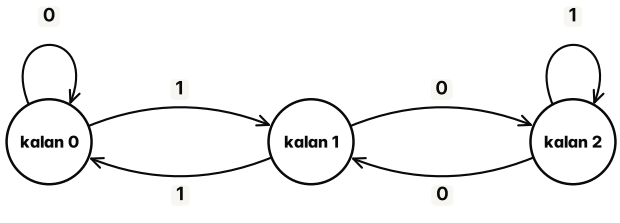
Bir kablodan bit bit bir ikilik sayı akıyor; belki 1 milyon basamak. Sayı bitince 3'e tam bölünüp bölünmediğini söylememiz gerekiyor, ama sayıyı hiçbir yere yazamıyoruz. Bilgisayar bunu nasıl anlar?

Sağına yeni bir basamak gelince sayıya ne olduğuna bakalım. Onlukta 7'nin sağına 2 yazınca 72 olur: 7, 10 ile çarpılır, 2 eklenir. İkilikte de aynısı, yalnız çarpan 2'dir. 100, yani 4; sağına 1 gelince 1001, yani 9 olur: 4'ün 2 katı + 1.

Bize sayı değil, 3'e bölümünden kalan lazım; üstelik eski kalan yenisini bulmaya yeter. 4, 3 + 1'dir. İki katına çıkınca 6 + 2 olur, yeni bitle 6 + 2 + 1. 6, 3'ün katı olduğu için kalana dokunmaz; geriye eski kalanın 2 katı ve yeni bit kalır: 2 + 1 = 3, kalan 0. Gerçekten de 9, 3'e tam bölünür. Kural her sayıda aynıdır: eski kalanın 2 katına biti ekleriz; bunun 3'e bölümünden kalan yeni kalandır.

Kalan yalnız 0, 1 ya da 2 olabilir. Makine bu 3 ihtimali aklında tutar; her biri bir **durum**, her durum bir daire. Her durumdan 2 ok çıkar, 0 biti ve 1 biti için; oklar kuraldan okunur:

```
kalan 0:  0 gelirse 0,  1 gelirse 1
kalan 1:  0 gelirse 2,  1 gelirse 0
kalan 2:  0 gelirse 1,  1 gelirse 2
```



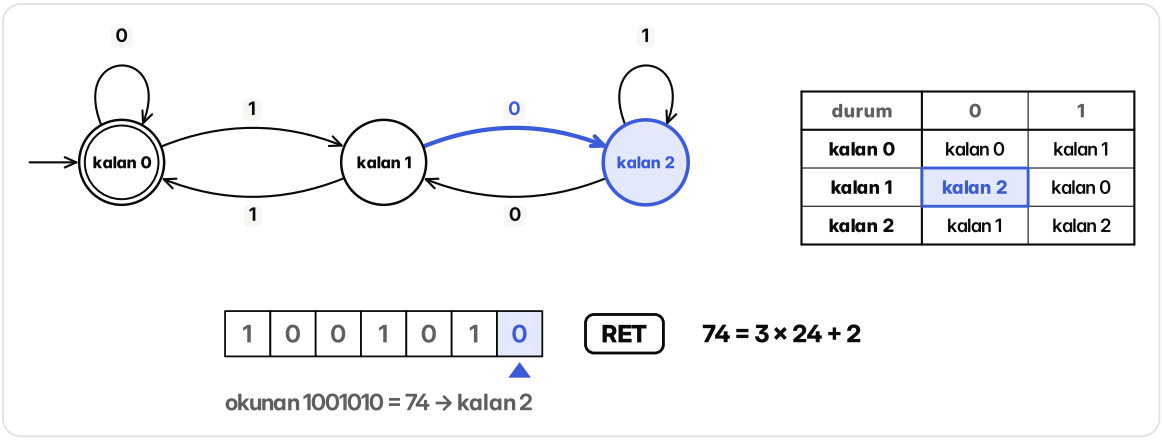
geçişler

durum	0	1
kalan 0	kalan 0	kalan 1
kalan 1	kalan 2	kalan 0
kalan 2	kalan 1	kalan 2

Şekil 2.1 Kalan 0, 1 ve 2 durumları ve her durumdan 0 ve 1 bitleri için çıkan altı ok.

Hiç bit okunmadan sayı 0'dır; makine kalan 0'dan başlar. Bu **başlangıç durumuna** dışarıdan kısa bir ok girer. Sayı 3'e bölünüyorsa son kalan 0'dır; bu yüzden kalan 0 dairesini çift çizgiyle çizeriz: **kabul durumu**. Okunan bitler bir dizgidir ve harfleri 0 ve 1'dir. Dizgi bitince makine kabul durumundaysa dizgiyi kabul eder, değilse reddeder.

1001011 dizgisini okutalım. Kalan 0'dan başlarız. İlk bit 1: kalan 1. Sonra 0: kalan 2. 0: kalan 1. 1: kalan 0. 0: 0'da kalır. 1: kalan 1. Son bit 1: kalan 0. Makine kabul durumunda durdu: bu sayı 75'tir ve 3'e bölününce 25 eder. Son bit 0 olsaydı makine kalan 2'de dururdu: 74, 3'e bölünmez.



Şekil 2.2 1001011 dizgisinin okunuşu: kalan 0, 1, 2, 1, 0, 0, 1, 0; makine kabul durumunda biter.

Dizgi soldan sağa yalnız bir kez okundu, hiçbir şey yazılmadı. Makinenin bütün belleği o anki durumdu: 3 daireden hangisinde olduğu. Durum sayısı sonlu, her durumda her bit için tek bir ok var; sonraki adım hep kesin. Böyle bir makineye **deterministik sonlu otomat** (deterministic finite automaton, DFA) denir. Kabul ettiği dizgilerin kümesi makinenin dilidir: burada 3'e bölünen sayıların ikilik yazımları. Sayı 1 milyon basamaklı da olsa makine aynı 3 daire arasında dolaşır. 7'ye bölünmeyi sorsaydık 7 durum, 11'e bölünmeyi sorsaydık 11 durum yeterdi; aklında tutulan tek şey hep o anki kalandır.

İki şart

videoda 14:31

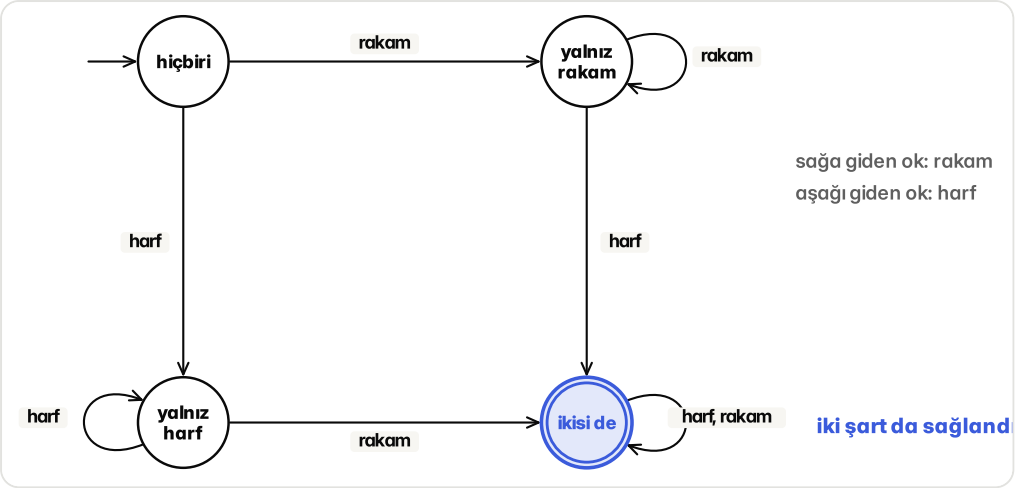
Bir site şifre için 2 şart koyuyor: en az 1 harf, en az 1 rakam. Şifreyi karakter karakter, soldan sağa yalnız bir kez okuyan bir makine bunu anlamak için neyi tutmalı?

Her an bilmesi gereken 2 şey var: şimdiye kadar harf gördü mü, rakam gördü mü? 2 evet-hayır sorusu 4 birleşim eder: hiçbiri, yalnız harf, yalnız rakam, ikisi de. Her birleşim bir durum olur. Makine hiçbiri durumundan başlar; kabul durumu yalnız ikisi de.

Oklara bakalım. Hiçbiri durumundayken harf gelirse yalnız harfe, rakam gelirse yalnız rakama geçilir. Yalnız harfteyken yine harf gelirse yerinde kalınır, rakam gelirse ikisi de olur. Yalnız rakamda da tersi geçerli. İkisi de durumuna bir kez girildi mi, ne gelirse gelsin orada kalınır: 2 şart da sağlandı, sonradan bozulmaz.

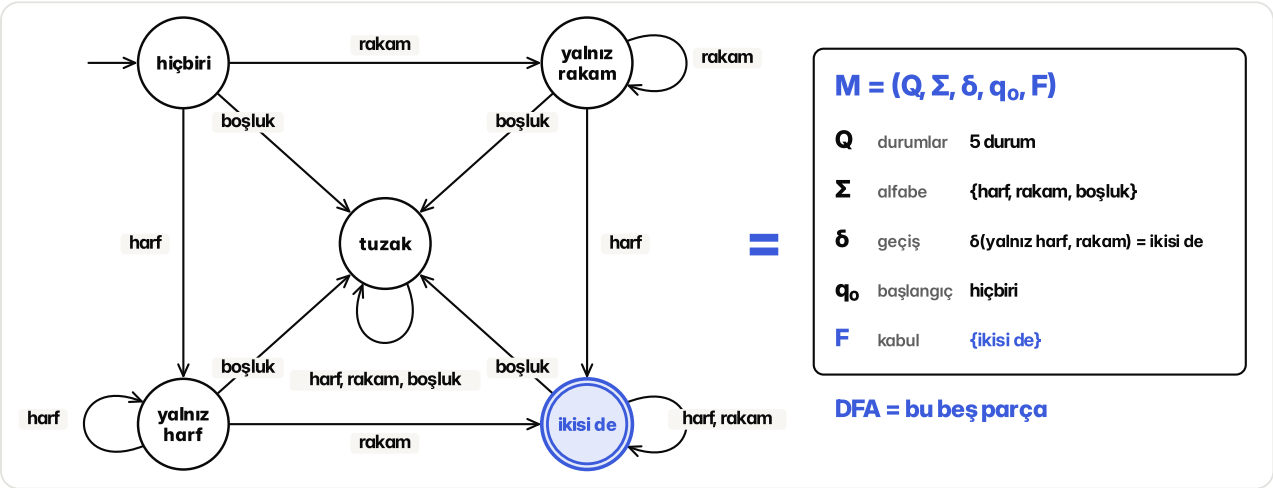
deniz5 şifresini okutalım. İlk harfle yalnız harfe geçeriz, kalan 4 harf bizi orada tutar; 5 gelince ikisi de olur: kabul. 1923 şifresi ise yalnız rakamda kalır; harf yok, ret.

Şimdi site bir şart daha koysun: şifrede boşluk olamaz. Boşluk geldiği an şifre geçersizdir ve ardından ne yazılırsa yazılsın düzelmez. Bunun için bir durum daha ekleriz: **tuzak**. 4 durumun her birinden boşluk oku tuzağa gider; tuzaktan çıkan her ok yine tuzağa döner. Tuzak, bir kez girilince çıkılmayan bir ret durumudur. deniz, boşluk, 5 yazarsak boşlukta tuzağa düşeriz; 5 gelse de çıkamayız, ret.



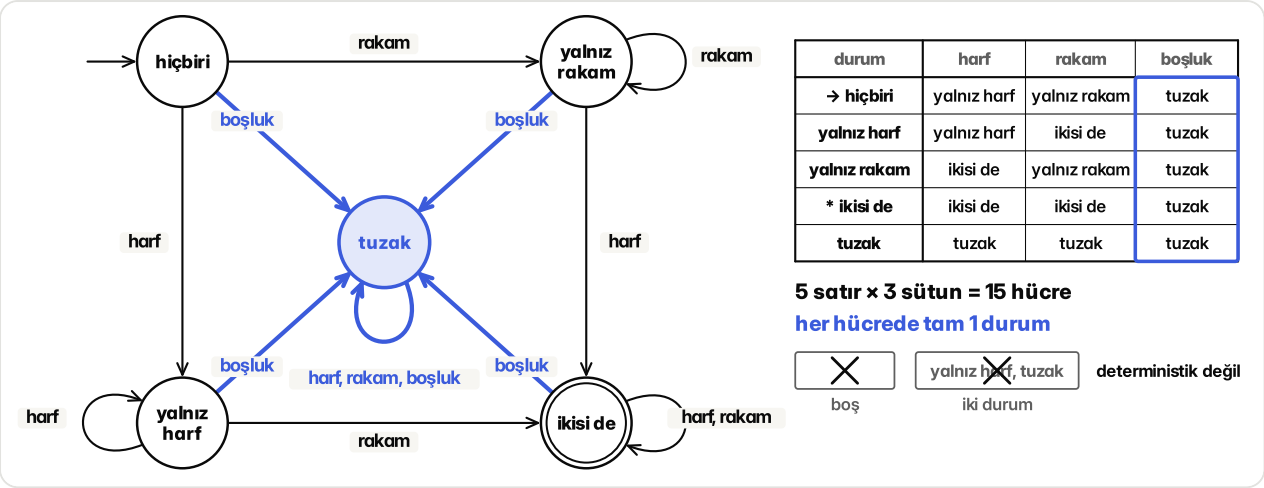
Şekil 2.3 Boşluk okuyan her durumdan tuzağa giden oklar; tuzaktan çıkan oklar tuzağa döner.

Bu makineyi resim çizmeden de tam olarak tarif edebiliriz; beş şey yeter. Durumlar: 5 durum. Alfabe: makinenin okuyabildiği simgeler; burada 3 tane: harf, rakam, boşluk. Geçiş fonksiyonu: bir durum ve bir simge verilince sıradaki durumu söyleyen kural; örneğin yalnız harf durumunda rakam okununca ikisi de. Başlangıç durumu: hiçbiri. Kabul durumları: şifre bittiğinde makine bunlardan birindeyse şifre geçer; burada yalnız ikisi de. DFA bu beş parçadan ibarettir ve M = (Q, Σ, δ, q_o, F) diye yazılır; diyagram bunun çizilmiş hâlidir.



Şekil 2.4 M = (Q, Σ, δ, q_o, F): durumlar, alfabe, geçiş fonksiyonu, başlangıç ve kabul durumları şifre makinesinde.

Geçiş kuralını bir tablo olarak da yazabiliriz: her satır bir durum, her sütun bir simge, her hücrede gidilecek durum. 5 satır, 3 sütun: 15 hücre. Makinenin deterministik olması, her hücrede tam olarak 1 durum yazması demektir: boş hücre yok, 2 durum yazan hücre yok. Tuzağa da bu yüzden ihtiyaç var; boşluk okuyan makinenin de gidecek bir yeri olmalı.



Şekil 2.5 Şifre makinesinin geçiş tablosu: 5 satır, 3 sütun, her hücrede tek durum.

Bir programda bu makine yalnız bu tablodan ibarettir: her karakterde tablodan bir hücre okunur, başka hesap yapılmaz. Derleyicinin kodu okuyup sayılara, adlara ve işaretlere ayıran ilk aşaması da böyle tablolarla çalışır. Şifre de kod da ne kadar uzun olursa olsun, her karaktere tek bir adım düşer.

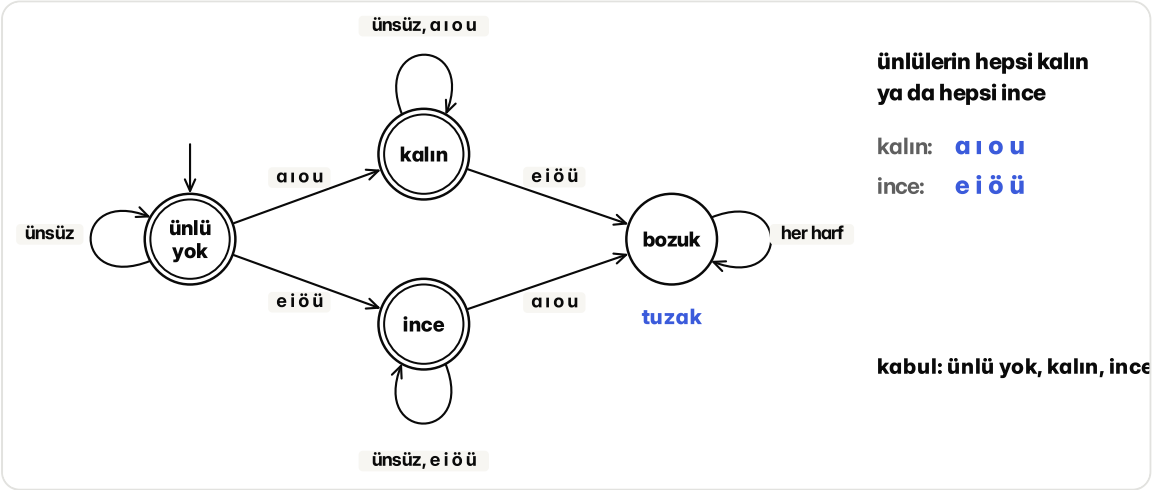
Üç kelime

videoda 17:03

Kelebek, armut, kitap. Türkçenin büyük ünlü uyumuna bu üçünden hangileri uyar? Bir makine bunu, kelimeyi harf harf yalnız bir kez okuyarak nasıl anlar?

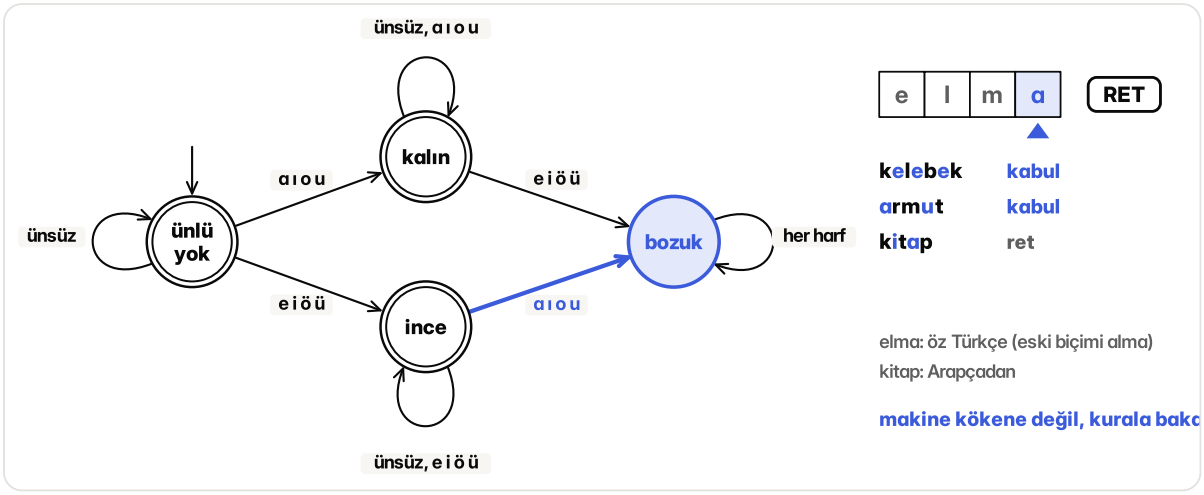
Büyük ünlü uyumu, bir kelimedeki ünlülerin ya hepsinin kalın ya hepsinin ince olmasıdır. Kalın ünlüler a, ı, o ve u; ince ünlüler e, i, ö ve ü. Makinenin aklında tutması gereken tek şey, şimdiye kadar hangi tür ünlü gördüğüdür. 4 durum yeter: ünlü yok, kalın, ince ve bozuk.

Ünsüzler uyumu etkilemez; her durumda ünsüz gelince makine yerinde kalır. Ünlü yokken kalın ünlü gelirse kalına, ince gelirse inceye geçilir. Kalındayken yine kalın gelirse orada kalınır; ince gelirse uyum bozulur. İncede de tersi geçerlidir. Bozuktan çıkış yok: bir kez girilince ne gelirse gelsin orada kalınır; böyle bir ret durumuna tuzak denir. Kabul durumları ünlü yok, kalın ve ince.



Şekil 2.6 Ünlü yok, kalın, ince ve bozuk durumları; ünsüzler yerinde bırakır, bozuk bir tuzaktır.

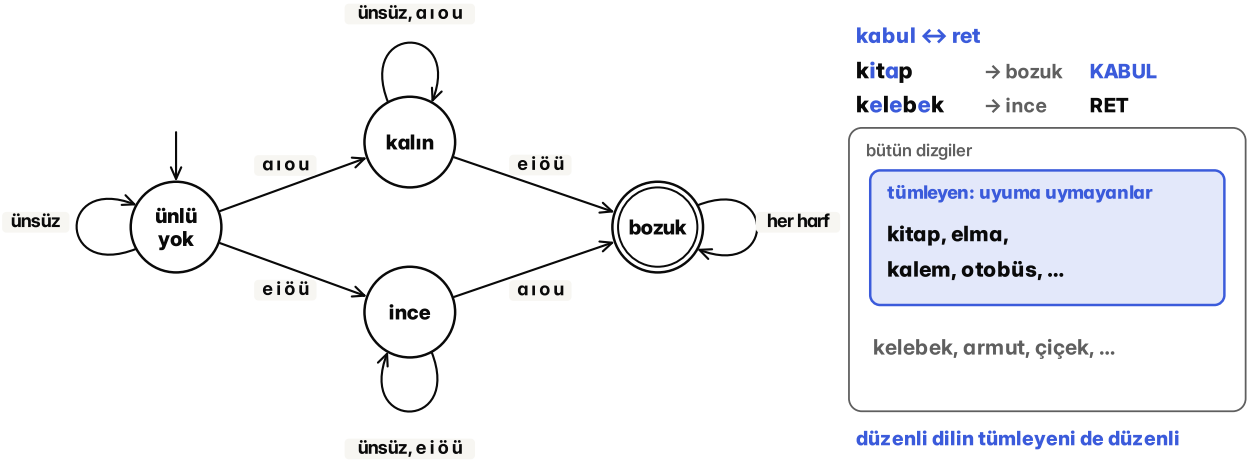
Kelimeleri okutalım. Kelebek: ilk e ile inceye geçeriz, sonraki iki e de ince; kabul. Armut: a ile kalına, u da kalın; kabul. Kitap: i ile inceye geçeriz, a gelince bozuğa düşeriz, son harf bizi kurtaramaz; ret. Elma da ret çıkar; oysa elma öz Türkçe bir kelime, kitap ise Arapçadan gelir. Makine kelimenin kökenine bakmaz, yalnız kuralı denetler.



Şekil 2.7 Kelebek, armut, kitap ve elma kelimelerinin makinede izlediği durumlar; kitap ve elma reddedilir.

Her kelime bir dizgidir. Makinenin kabul ettiği bütün dizgilerin kümesi onun dilidir; burada ünlüleri uyumlu dizgiler. Bir dili tanıyan bir DFA varsa o dile **düzenli dil** (regular language) denir. Uyumlu dizgiler de düzenli bir dildir: onu tanıyan 4 durumlu makine elimizde.

Şimdi tersini isteyelim: uyuma uymayan kelimeleri bulan bir makine. Baştan kurmaya gerek yok; kabul durumlarını ters çeviririz. Ünlü yok, kalın ve ince artık ret; bozuk artık kabul. Yeni makine kitabı kabul eder, kelebeği reddeder. Tanıdığı dil, ilk dilin dışında kalan bütün dizgilerdir: ilk dilin **tümleyeni**. Demek ki düzenli bir dilin tümleyeni de düzenlidir.



Şekil 2.8 Kabul durumları ters çevrilmiş makine: yalnız bozuk kabul, kitap kabul, kelebek ret.

Bu çevirme ancak her durumda her harf için tam bir ok varsa işe yarar. Bozuk durumu olmasaydı kitap, a harfinde gidecek ok bulamaz, takılıp reddedilirdi; kabul durumlarını çevirsek de takılan dizgi yine reddedilirdi. İşte tuzak, uyumu bozan her kelimeye bitebileceği bir durum veriyor.

Bu küçük makineler bilgisayarın her yerinde. Metinde hızlı kelime arayan bir yöntem, aranan kelimeden önce bir tablo kurar; o tablo aslında o kelimeyi arayan bir sonlu otomattır. Ağdaki bir bağlantı da kurulurken, açıkken ve kapanırken durumdan duruma geçer. Türkçe metinleri işleyen programlar da kelimeyi kök ve eklerine ayırırken eklerin sırasını ve ünlü uyumunu böyle sonlu durumlu makinelerle denetler.

Özet

- Deterministik sonlu otomat, dizgiyi soldan sağa bir kez okur; bütün belleği o anki durumdur ve durum sayısı sonludur.
- 3'e bölünebilirlik makinesi 3 durumlu (kalan 0, 1, 2): yeni kalan, eski kalanın 2 katı artı yeni bitin 3'e bölümünden kalanıdır; 1001011 kabul edilir, 1001010 reddedilir.
- DFA beş parçadır: durumlar, alfabe, geçiş fonksiyonu, başlangıç durumu ve kabul durumları. Geçiş tablosunun her hücresinde tam bir durum yazar.
- Tuzak, girilince çıkılmayan bir ret durumudur; her durumda her simge için bir ok bulunması için gereklidir.
- Bir DFA'nın tanıdığı dile düzenli dil denir; kabul durumlarını ters çevirince tümleyen elde edilir, yani düzenli bir dilin tümleyeni de düzenlidir.
- Sözcük çözümleme, metinde arama, ağ bağlantıları ve Türkçe biçim çözümlemesi bu tür makinelerle çalışır.

Sorular

1 **3'e bölünebilirlik makinesi kalan 2 durumundayken 1 biti okuyor. Hangi duruma geçer?**

A Kalan 0
B Kalan 1
C Kalan 2
D Tuzak durumuna

2 **3'e bölünebilirlik makinesi kalan 0'dan başlayıp 10110 dizgisini okuyor. Son durumun kalanı kaçtır? Yalnız sayıyı yaz.**

Cevap: _____

3 **Aynı makine aşağıdaki dizgilerden hangisini kabul eder?**

A 1010
B 100
C 111
D 11011

4 **Deterministik olmak, bir DFA'nın geçiş tablosu için ne demektir?**

A Her hücrede tam olarak bir durum yazar.

B Bazı hücreler boş kalabilir.

C Bir hücrede birden çok durum yazabilir.

D Hücre sayısı dizginin uzunluğuna bağlıdır.

5 **Şifre makinesi (hiçbiri, yalnız harf, yalnız rakam, ikisi de ve tuzak durumları) ab c1 şifresini okuyor. Şifrede a ile b arasında boşluk yok, b ile c arasında bir boşluk var. Makine hangi durumda biter?**

Cevap: _____

6 **Aynı şifre makinesi (en az bir harf, en az bir rakam, boşluk yok) hangi şifreyi kabul eder?**

A 2024

B 42ab

C kedi

D k ed1

7 **Dört durumlu ünlü uyumu makinesi (ünlü yok, kalın, ince, bozuk) hangi kelimeyi reddeder?**

A kelebek

B armut

C okul

D anne

8 **Ünlü uyumu makinesinin kabul durumları ters çevrildi: yalnız bozuk kabul durumu, öteki üç durum ret. Yeni makine hangi kelimeyi kabul eder?**

A armut

B kelebek

C elma

D çiçek

3 Deterministik olmayan sonlu otomat

Bu bölümde makineye tahmin etme izni veriyoruz: aynı harfle birden çok ok ya da hiç ok bulunabilen deterministik olmayan sonlu otomatı, harf okumayan oku, epsilon kapanışını ve iki hazır makineyi birleştirmenin yollarını göreceğiz.

Sonu bilinmeyen dizgi

videoda 19:35

Bir makineye bir dizgi (string), yani harflerden oluşan bir dizi geliyor; harfleri yalnız 0 ve 1. Makine harfleri birer birer okuyor, geri dönüp bakamıyor, dizginin nerede biteceğini de bilmiyor. Dizgi bittiği anda cevap vermeli: sondan üçüncü harf 1 mi?

Zorluk şu: bir harfin sondan üçüncü olduğunu ancak 2 harf sonra, dizgi orada biterse anlarız. Deterministik sonlu otomat (DFA) her harfte tek bir yola gider; bu yüzden son 3 harfi her an aklında taşımak zorunda. Makine tahmin edebilseydi iş kolaylaşırdı.

4 durumlu bir makine çizelim. Başlangıç durumu bekle: her harfte kendine döner. Bekle durumunda 1 okununca 2 seçenek var: beklemeye devam etmek ya da "bu 1 sondan üçüncü harf" diye tahmin edip 2 kaldı durumuna geçmek. Tahmin doğruysa geriye tam 2 harf kalmıştır. Sonraki 2 harf ne olursa olsun önce 1 kaldı, sonra 0 kaldı durumuna geçilir. 0 kaldı kabul durumudur: dizgi tam burada biterse tahmin doğru çıkmıştır.

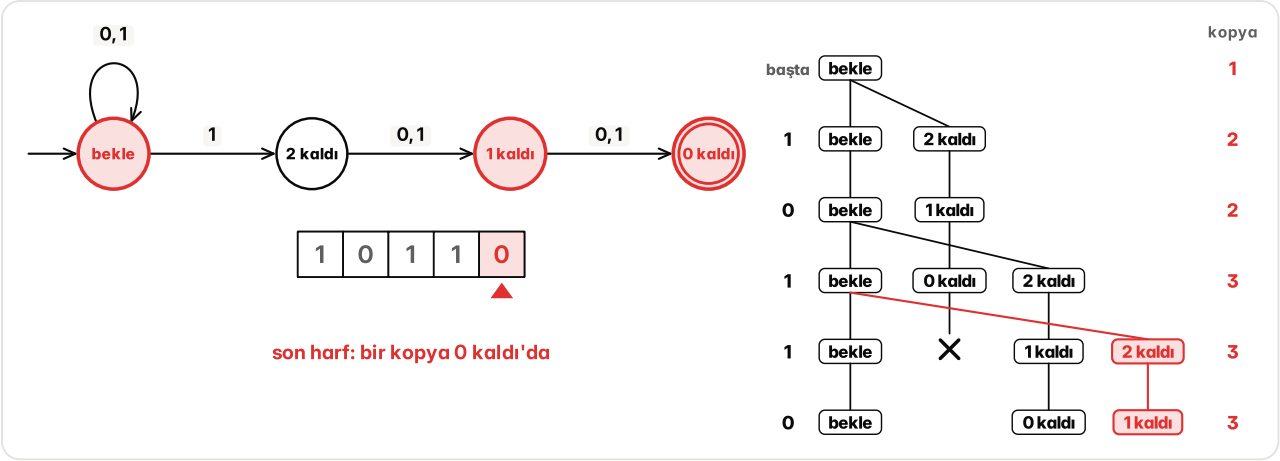


Şekil 3.1 Dört durumlu makine: bekle, 2 kaldı, 1 kaldı ve çift daireli 0 kaldı; bekle durumundan 1 ile iki ok çıkıyor.

Bu makinenin iki tuhaflığı var: bekle durumundan 1 harfiyle 2 ok çıkıyor, 0 kaldı durumundan ise hiç ok çıkmıyor. Tahminden sonra harf gelmeye devam ederse o yol orada biter. Aynı harfle birden çok ok ya da hiç ok bulunabilen makineye **deterministik olmayan sonlu otomat** (nondeterministic finite automaton, NFA) denir.

Makineyi çalıştıralım: 1 0 1 1 0. Her seçimde makinenin kopyalandığını düşünelim. İlk 1 gelince makine ikiye ayrılır: biri bekler, öteki bu 1'i sondan üçüncü sayar. O kopya 2 harf sonra 0 kaldı'ya varır; ama 4. harf gelince gidecek oku yoktur, silinir. 3. harfteki 1 yeni bir kopya doğurur; bu kopya

son harfle tam 0 kaldı'ya varır. 4. harfteki 1'in kopyası ise dizgi bittiğinde 1 kaldı'da kalır.



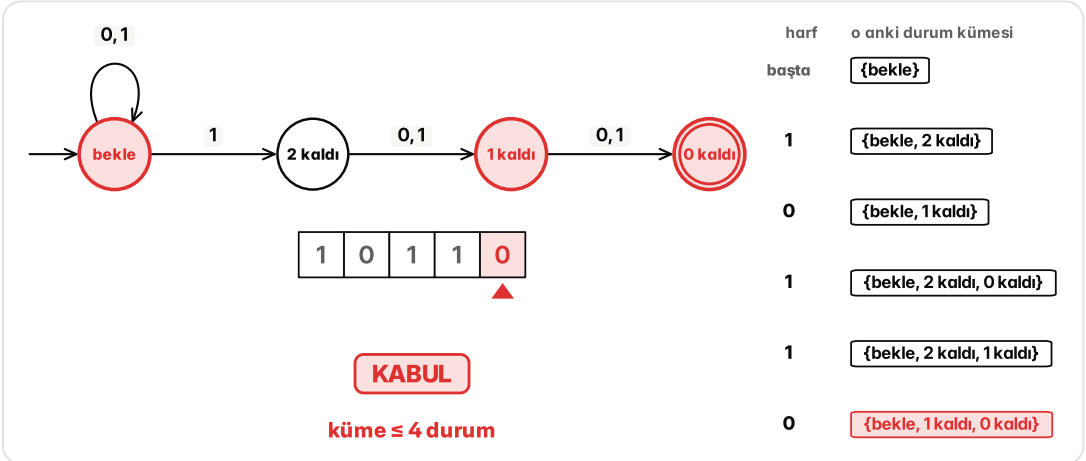
Şekil 3.2 Kopyaların ağacı: her 1 harfinde bir dal ayrılır, oku olmayan dal silinir; kopya sayısı 1, 2, 2, 3, 3, 3.

Dizgi bitti; ortada 3 kopya var: bekle, 1 kaldı ve 0 kaldı. Kural şu: kopyalardan en az biri kabul durumundaysa dizgi kabul edilir. 0 kaldı'daki kopya doğru tahmin etti; sondan üçüncü harf gerçekten 1. Yanlış tahminler zarar vermez, birinin doğru olması yeter. 0 1 1 dizgisinde ise hiçbir kopya 0 kaldı'ya varamaz: ret.

Kopyaları tek tek izlemek yerine, her harften sonra makinenin aynı anda bulunduğu durumları bir küme olarak yazabiliriz; aynı duruma düşen 2 kopyanın geleceği de aynıdır. 1 0 1 1 0 için küme her harfte şöyle yenilenir:

```
başlangıç    {bekle}
1 okununca   {bekle, 2 kaldı}
0 okununca   {bekle, 1 kaldı}
1 okununca   {bekle, 2 kaldı, 0 kaldı}
1 okununca   {bekle, 2 kaldı, 1 kaldı}
0 okununca   {bekle, 1 kaldı, 0 kaldı}
```

Sondaki kümede 0 kaldı var: kabul. Dizgi ne kadar uzun olursa olsun, küme 4 durumdan büyük olamaz.



Şekil 3.3 Her harften sonra makinenin bulunduğu durumlar kümesi; sonda 0 kaldı kümede olduğu için kabul.

Gerçek bilgisayar tahmin edemez; ama bu kümeyi her harfte yeniden hesaplayabilir. Deterministik olmayan makine böylece kâğıt üstünde kalmaz, sıradan bir programla adım adım çalıştırılır.

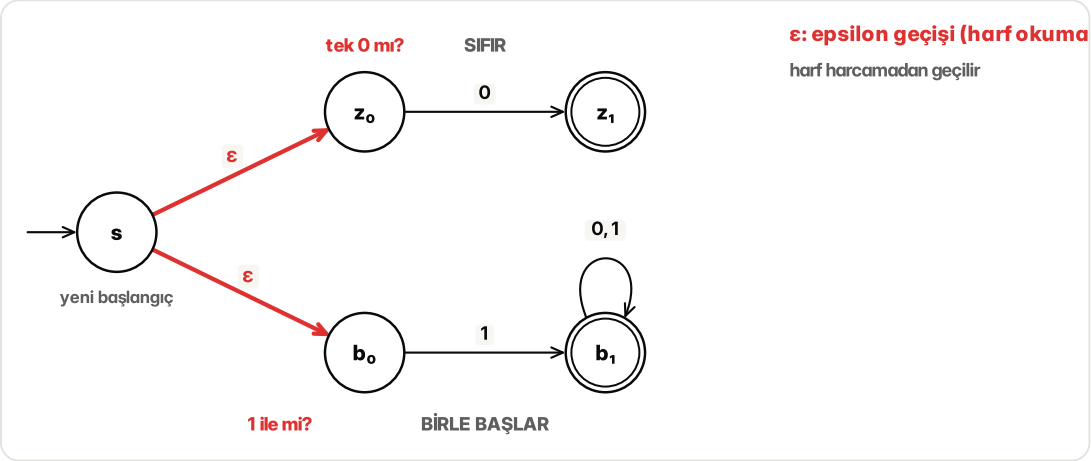
Başında sıfır olmayan sayı

videoda 22:10

İkilik bir sayı başında gereksiz sıfır olmadan yazılır: 0, 10, 101 doğru yazımlardır; 01 değil. Kural 2 parçalı: ya tek başına bir 0, ya da 1 ile başlayan herhangi bir dizgi. Her parçanın küçük bir makinesi hazır; soru, ikisini hiç değiştirmeden tek makinede nasıl birleştireceğimiz.

Sıfır makinesi (SIFIR) 0'la kabul durumuna geçer, başka oku yok: z_0 başlangıç, z_1 kabul durumu. 1'le başlayan makine (BİRLE BAŞLAR) 1'le kabul durumuna geçer, sonra her harfte kendine döner: b_0 başlangıç, b_1 kabul durumu. Oku olmayan harf gelirse o yol biter; yollardan biri kabul durumunda biterse dizgi kabul edilir.

Birleştirmek için başa yeni bir başlangıç durumu s koyar, ondan 2 makinenin başlangıcına birer ok çizeriz. Bu oklar harf okumaz, makine onlardan harf harcamadan geçer; üstlerinde boş dizginin simgesi ε durur, adları **epsilon geçişi**dir. Hangi oktan gideceği de bir tahmin: sayı tek bir 0 mı, yoksa 1 ile mi başlıyor?



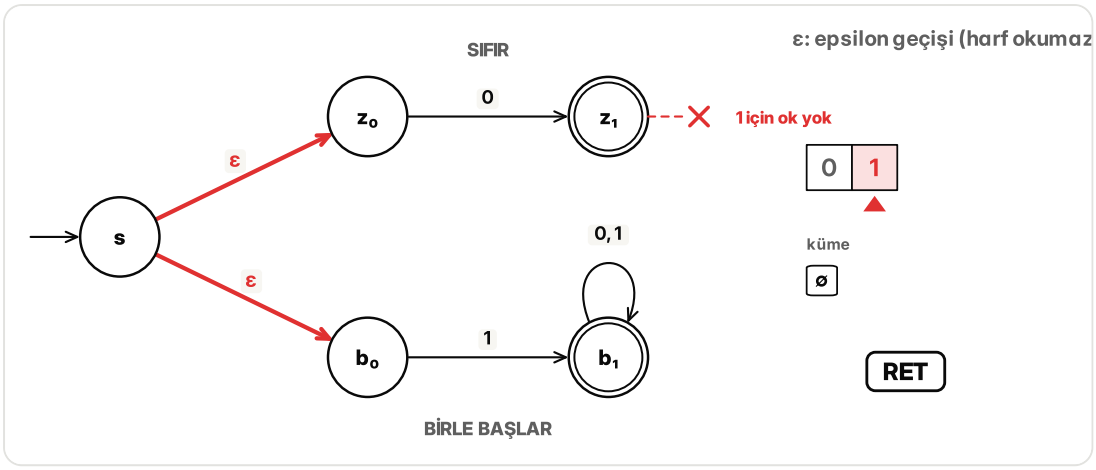
Şekil 3.4 Yeni başlangıç durumu s, iki makinenin başlangıcına ε etiketli oklarla bağlanıyor.

Bir durumdan yalnız epsilon geçişleriyle varılabilen bütün durumlara, kendisi de dahil, o durumun **epsilon kapanışı** denir. Yeni başlangıcın kapanışında 3 durum var: s, z_0 ve b_0. Makine işe bu 3 durumda birden başlar; her harften sonra da vardığı durumların kapanışı kümeye eklenir.

1 0 1 deneyelim. İlk harf 1: sıfır makinesinin bu harf için oku yok; 1'le başlayan makine kabul durumuna geçer ve sonraki harflerde orada kalır.

```
başlangıç    {s, z₀, b₀}
1 okununca   {b₁}
0 okununca   {b₁}
1 okununca   {b₁}
```

Dizgi bitince kümede kabul durumu var: kabul. 0 1 dizgisinde ise 0'la yalnız sıfır makinesi kabul durumuna geçer ({z_1}), ardından gelen 1 için oku yok. Küme boşalır: ret.



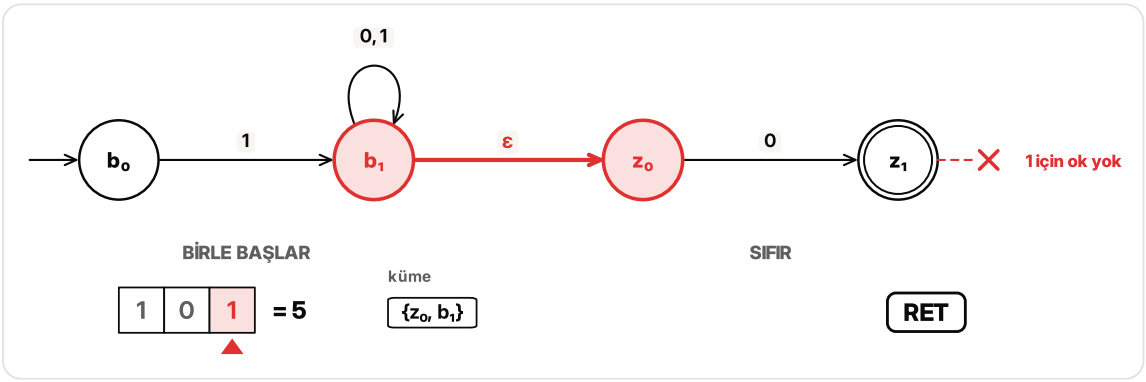
Şekil 3.5 Birleşim makinesinde 1 0 1 kümeleri kabulle, 0 1 kümeleri boşalarak sonuçlanıyor.

Başka bir kural: 1 ile başlayıp 0'la biten sayılar, yani 0'dan büyük çift sayılar; son biti 0 olan ikilik sayı çifttir. Dizgi önce 1'le başlayan makineden, ardından sıfır makinesinden geçmeli. Bunun için ilkinin kabul durumundan (b_1) sıfır makinesinin başlangıcına (z_0) bir epsilon geçişi çizeriz. b_1 artık kabul durumu değil: dizgi 0'ı da okumalı; kabul durumu yalnız z_1.

1 1 0 deneyelim; bu 6.

```
başlangıç     {b₀}
1 okununca    {b₁, z₀}
1 okununca    {b₁, z₀}
0 okununca    {b₁, z₀, z₁}
```

İlk 1, 1'le başlayan makineyi eski kabul durumuna götürür; o epsilon geçişi sayesinde makine aynı anda sıfır makinesinin başlangıcındadır. İkinci 1'de sıfır makinesinin oku yok; ama öteki makine kendine döner, kapanış z_0'ı yeniden ekler. Son harf 0: sıfır makinesi z_1'e varır, kabul. 1 0 1, yani 5'te ise makine 2. harften sonra z_1'e varır, ama son 1 için oradan ok yok: ret.



Şekil 3.6 Art arda bağlanmış makinede 1 1 0 (6) kabul, 1 0 1 (5) ret; z_1 kümeye girip çıkıyor.

İki makinenin hiçbir okuna dokunmadık; birleşim için 2, art arda bağlamak için tek bir epsilon geçişi yetti. Büyük bir kuralı tanıyan makine de böyle kurulur: küçük parçalar ayrı ayrı çizilir, harf okumayan oklarla birbirine bağlanır.

Özet

- Deterministik olmayan sonlu otomatta (NFA) aynı harfle birden çok ok ya da hiç ok bulunabilir; yollardan biri kabul durumunda biterse dizgi kabul edilir.
- Yanlış tahminler zarar vermez: kopyalardan en az birinin kabul durumunda olması yeter.
- Kopyalar yerine her harften sonra durumlar kümesi yazılabilir; bilgisayar tahmin etmez, kümeyi her harfte yeniden hesaplar.
- Epsilon geçişi harf okumadan geçilen oktur; bir durumun epsilon kapanışı, ondan yalnız bu oklarla varılan durumlardır (kendisi dahil).
- İki makineyi birleştirmek için yeni bir başlangıç ve 2 epsilon geçişi, art arda bağlamak için tek epsilon geçişi yeter.

Sorular

1 **Deterministik olmayan sonlu otomatı (NFA) deterministik sonlu otomattan (DFA) ayıran özellik hangisidir?**

A Dizgiyi sağdan sola okuması

B Kabul durumunun birden çok olması

C Aynı harfle birden çok ok ya da hiç ok bulunabilmesi

D Durum sayısının her zaman daha az olması

2 **Sondan üçüncü harfi arayan NFA'da bir kopya yanlış tahmin etmiş olsun. Dizgi bittiğinde başka bir kopya 0 kaldı durumundaysa sonuç ne olur?**

A Dizgi kabul edilir; kopyalardan en az birinin kabul durumunda olması yeter.

B Dizgi reddedilir; bir kopya bile yanılmışsa makine ret verir.

C Dizgi reddedilir; kabul için bütün kopyaların kabul durumunda olması gerekir.

D Makine durur; yanlış tahmin hata verir.

3 **SIFIR ve BİRLE BAŞLAR makinelerini birleştirirken yeni başlangıç durumu s'ten z_o ve b_o'a ε geçişleri çizildi. s'nin epsilon kapanışı nedir?**

A $\{z_o, b_o\}$

B $\{s\}$

C $\{s, z_1, b_1\}$

D $\{s, z_o, b_o\}$

4 **SIFIR makinesi 2 durumlu, BİRLE BAŞLAR makinesi 2 durumludur. Yeni bir başlangıç durumu ve 2 ε geçişiyle birleştirilince makine kaç durumlu olur?**

Cevap: _____

5 **Sondan üçüncü harfi 1 olan dizgileri kabul eden 4 durumlu NFA hangi dizgiyi kabul eder?**

A 1 0 0 0
B 1 1 0 0
C 0 0 1 1
D 0 1 0 1 1

6 **Sondan üçüncü harf NFA'sında 1 1 1 0 dizgisi okununca, son harften sonra kümede kaç durum bulunur?**

Cevap: _____

7 **Sondan üçüncü harf NFA'sında 0 1 1 0 1 dizgisi okununca son harften sonra durum kümesi hangisidir?**

A {bekle, 1 kaldı, 0 kaldı}
B {bekle, 2 kaldı, 1 kaldı}
C {bekle, 2 kaldı, 0 kaldı}
D {bekle, 2 kaldı}

8 **1 ile başlayıp 0 ile biten sayıları tanıyan art arda bağlanmış makinede (b_1'den z_o'a ε geçişi var, kabul durumu yalnız z_1) hangi dizgi kabul edilir?**

A 0 1 0
B 1 0 1 1 0
C 1 1 1
D 1 0 1

BİÇİMSEL DİLLER VE OTOMATA

4 NFA'dan DFA'ya: eşdeğerlik ve küçültme

Bu bölümde tahmin eden makinenin tahmin etmeyen bir makineye nasıl çevrildiğini (alt küme kurulumu), bu çevirinin neden durum sayısını ikiye katlayabildiğini ve iki makinenin aynı işi yapıp yapmadığının küçültme yoluyla nasıl anlaşıldığını göreceğiz.

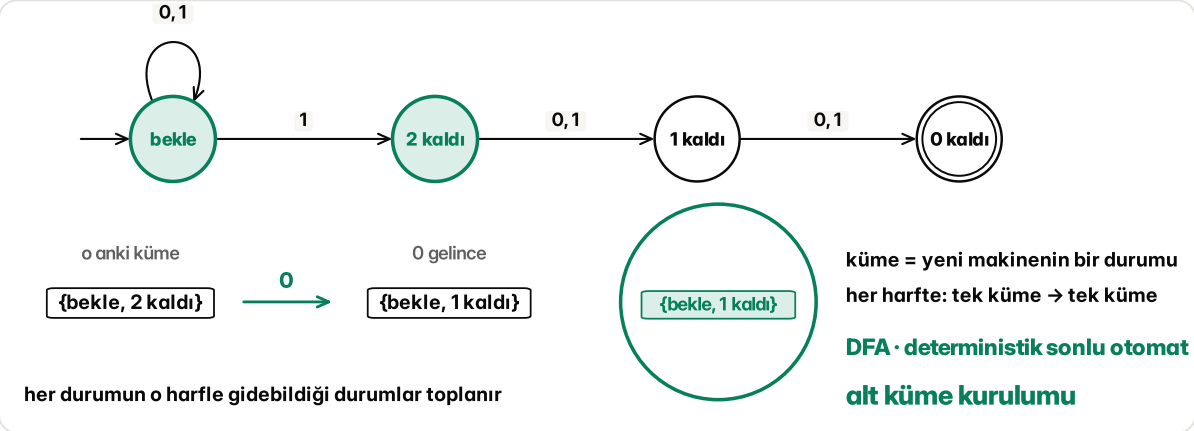
Dört durum, sekiz durum

videoda 24:49

0 ve 1'lerden oluşan bir dizginin (string), yani harflerden oluşan bir dizinin, sondan üçüncü harfi 1 mi? 4 durumlu bir makine bunu tahmin ederek anlıyor. Tahmin etmeyen bir makineye ise 4 durum yetmiyor, 8 gerekiyor. Neden?

Tahmin eden makine bekle durumunda başlar ve her harfte kendine döner. Bekle durumunda 1 okununca ayrıca tahmin eder: bu 1, sondan üçüncü harf olabilir. Tahmin yolu 2 kaldı'ya geçer; sonraki 2 harf ne olursa olsun 1 kaldı ve 0 kaldı'ya varır. 0 kaldı kabul durumudur. Böyle makineye deterministik olmayan sonlu otomat (NFA) denir; yollardan biri kabul durumunda biterse dizgi kabul edilir.

Bilgisayar tahmin edemez, ama her harften sonra makinenin aynı anda bulunabildiği durumları bir küme olarak yazabilir: harf gelince kümedeki her durumun o harfle gidebildiği durumlar toplanır. Şimdi her kümeyi yeni bir makinenin tek bir durumu yapalım. Yeni makine her harfte tek bir kümeye geçer; böyle makineye deterministik sonlu otomat (DFA) denir. Bu çeviriye **alt küme kurulumu** (subset construction) denir.



Şekil 4.1 Küme bir durum olur: NFA'nın bulunabileceği durumların kümesi yeni makinenin tek bir durumuna dönüşüyor.

Başlangıç kümesi yalnız {bekle}. 0'la yine yalnız bekle'ye, 1'le bekle ve 2 kaldı'ya gidilir: yeni bir küme, tabloya yazarız. Ondan 0'la {bekle, 1 kaldı}, 1'le {bekle, 2 kaldı, 1 kaldı} çıkar; ikisi de yeni. Her yeni küme için aynısını yapar, yeni küme çıkmayınca dururuz: 8 küme, her birinden 2 ok. İçinde 0 kaldı olan 4 küme kabul durumudur (yanlarında *); orada tahminlerden biri tutmuştur.

```
küme                    0 ile                   1 ile
{bekle}                 {bekle}                 {bekle, 2}
{bekle, 2}              {bekle, 1}              {bekle, 2, 1}
{bekle, 1}              {bekle, 0}              {bekle, 2, 0}
{bekle, 2, 1}           {bekle, 1, 0}           {bekle, 2, 1, 0}
* {bekle, 0}            {bekle}                 {bekle, 2}
* {bekle, 2, 0}         {bekle, 1}              {bekle, 2, 1}
* {bekle, 1, 0}         {bekle, 0}              {bekle, 2, 0}
* {bekle, 2, 1, 0}      {bekle, 1, 0}           {bekle, 2, 1, 0}
```

Tabloda 2, 1 ve 0 sırasıyla 2 kaldı, 1 kaldı ve 0 kaldı demektir. 1 1 1 0 0 dizgisini okutalım: her harfte tablodan tek bir ok izlenir; tahmin yok, her an tek bir durum. Yol {bekle} → {bekle, 2} → {bekle, 2, 1} → {bekle, 2, 1, 0} → {bekle, 1, 0} → {bekle, 0} olur. Sonda kümede 0 kaldı var: kabul.

NFA'nın okları

bekle —0→ bekle
bekle —1→ bekle, 2 kaldı
2 kaldı —0, 1→ 1 kaldı
1 kaldı —0, 1→ 0 kaldı

2, 1, 0 = 2 kaldı, 1 kaldı, 0 kaldı
bekle her kümede

8 küme · 16 ok

*** 4 kabul (0 kaldı içeren)**

#	küme	0 ile	1 ile
1	{bekle}	{bekle}	{bekle, 2}
2	{bekle, 2}	{bekle, 1}	{bekle, 2, 1}
3	{bekle, 1}	{bekle, 0}	{bekle, 2, 0}
4	{bekle, 2, 1}	{bekle, 1, 0}	{bekle, 2, 1, 0}
5 *	{bekle, 0}	{bekle}	{bekle, 2}
6 *	{bekle, 2, 0}	{bekle, 1}	{bekle, 2, 1}
7 *	{bekle, 1, 0}	{bekle, 0}	{bekle, 2, 0}
8 *	{bekle, 2, 1, 0}	{bekle, 1, 0}	{bekle, 2, 1, 0}

Şekil 4.2 Alt küme kurulumunun tablosu: 8 küme, 16 ok, içinde 0 kaldı olan 4 kabul kümesi.

Kümelere yakından bakalım: 2 kaldı kümedeyse son harf 1; 1 kaldı varsa sondan ikinci, 0 kaldı varsa sondan üçüncü harf 1. Her küme dizginin son 3 harfini yazıyor; 3 harfin 8 birleşimi var, 8 durum bu yüzden. 4 durumdan 16 küme kurulabilir; ama bekle hep içinde olduğu için yalnız 8'ine varılır.

Daha az durum yetmez. 100 ile 000 sonlarını alalım: dizgi burada biterse biri kabul, öteki ret. 010 ile 000 ise ardına bir harf gelince ayrılır; farklı oldukları harf sondan üçüncüye kayar. Hangi iki son olursa olsun böyle ayrılır; 8'den az durumlu makine ikisini aynı durumda tutar ve birinde yanılır.

kümede varsa → o harf 1

2 kaldı → son harf
1 kaldı → sondan 2. harf
0 kaldı → sondan 3. harf

11100 okunurken:
000 → 001 → 011 → 111 → 110 → 100

3 harf → 2^3 = 8 durum

4 durum → 2^4 = 16 küme mümkün
bekle hep içinde → yalnız 2^3 = 8

#	küme	0 ile	1 ile	son 3 harf
1	{bekle}	{bekle}	{bekle, 2}	000
2	{bekle, 2}	{bekle, 1}	{bekle, 2, 1}	001
3	{bekle, 1}	{bekle, 0}	{bekle, 2, 0}	010
4	{bekle, 2, 1}	{bekle, 1, 0}	{bekle, 2, 1, 0}	011
5 *	{bekle, 0}	{bekle}	{bekle, 2}	100
6 *	{bekle, 2, 0}	{bekle, 1}	{bekle, 2, 1}	101
7 *	{bekle, 1, 0}	{bekle, 0}	{bekle, 2, 0}	110
8 *	{bekle, 2, 1, 0}	{bekle, 1, 0}	{bekle, 2, 1, 0}	111

kısa dizgide eksik harf 0 sayılır

Şekil 4.3 Her küme dizginin son üç harfini yazıyor: 8 durum, 2^3 birleşim.

Soru sondan dördüncü harf olsaydı, NFA'ya tek bir durum eklenirdi (5 durum); DFA ise 16 duruma çıkardı. Sondan onuncu harf için 1.024, sondan yirminci için 1 milyonu aşkın durum gerekir. NFA birer birer büyür, DFA her adımda 2'ye katlanır.

Alt küme kurulumu her NFA'da işler. ε geçişleri varsa, başlangıç kümesi dahil her kümeye o oklarla varılan durumlar da eklenir. Örneğin s'den p ve q'ya ε geçişi varsa başlangıç kümesi {s, p, q} olur. Demek ki NFA'nın tanıdığı her dili bir DFA da tanır; her DFA da zaten tahmin etmeyen bir NFA'dır. İkisi aynı dilleri tanır: tahmin, sonlu otomata güç değil, kısalık katar.

İki makine, tek kural

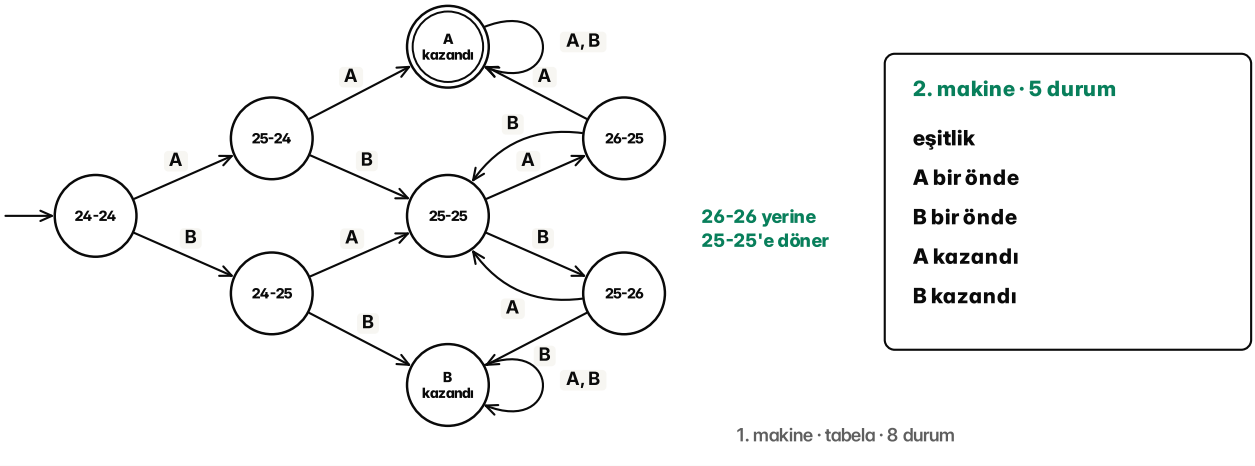
videoda 27:44

Voleybolda set 25 sayıda biter; ama skor 24 eşitse, bir takım 2 sayı öne geçene kadar oyun sürer. İki kişi bu kuralı birer makineye çevirmiş: biri 8 durumlu, öteki 5. İkisi aynı işi mi yapıyor? Maçın sonsuz sayıda gidişi var; hepsini tek tek denemeden bunu nasıl anlarız?

24 eşitten başlayıp her sayıyı alan takımın harfini yazalım: A ya da B. Maçın gidişi böylece bir dizgi olur. İki makine de harfleri birer birer okur ve her harfte tek bir yola gider: ikisi de DFA. Dizgi bittiğinde seti A almışsa kabul ederler.

Birinci makine tabela gibi düşünür: 24-24, 25-24, 25-25 gibi 6 skor, bir de A kazandı ve B kazandı. 26-26 için yeni daire çizmemiş, 25-25'e dönmüş. İkinci makinede yalnız 5 durum var: eşitlik, A bir önde, B bir önde, A kazandı, B kazandı.

```
tabela makinesi (8 durum)
24-24          A → 25-24          B → 24-25
25-24          A → A kazandı      B → 25-25
24-25          A → 25-25          B → B kazandı
25-25          A → 26-25          B → 25-26
26-25          A → A kazandı      B → 25-25
25-26          A → 25-25          B → B kazandı
A kazandı      A, B → A kazandı   (kabul)
B kazandı      A, B → B kazandı
```

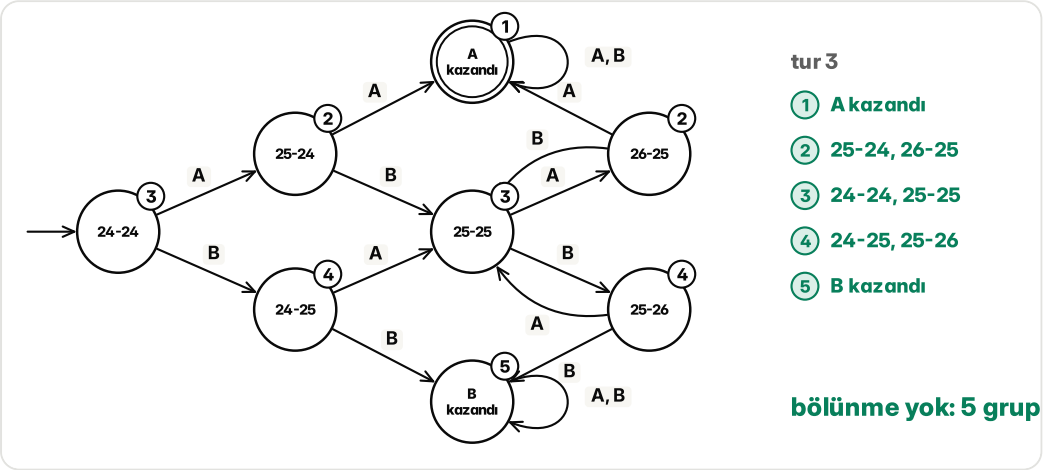


Şekil 4.4 Tabela makinesinin 8 durumu ve ikinci makinenin 5 durumu.

Bir durumun **geleceği**, ondan sonra gelebilecek her dizgiye makinenin vereceği cevaptır. 24-24 ile 25-25'in geleceği aynı: ikisinde de bir takım art arda 2 sayı alırsa seti alır, sayılar paylaşılırsa yine eşitliğe dönülür. Geleceği aynı olan iki durum birleştirilebilir; makine hangisinde olduğunu bilmese de hiçbir dizgide yanılmaz.

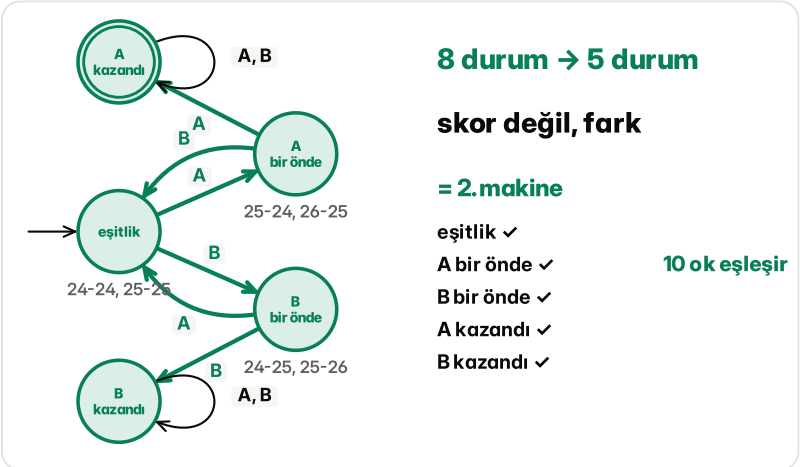
Aynı geleceği gözle değil, kuralla buluruz. Önce durumları 2 gruba ayırırız: kabul ve ret. Sonra bir grupta, aynı harfle farklı gruplara giden iki durumu ayırırız; onların geleceği aynı olamaz. Bölünecek grup kalmayınca her grup tek bir durum olur. Bu işe **DFA küçültme** denir.

Adım adım: kabul grubunda yalnız A kazandı var. Ret grubunda 25-24 ile 26-25, A harfiyle kabule gider; ayrılırlar. Sonra 24-24 ile 25-25, A harfiyle bu yeni gruba gider; onlar da ayrılır. Ardından 24-25 ile 25-26 A harfiyle eşitlik grubuna gider, B kazandı ise yerinde kalır; o da ayrılır. Artık hiçbir grup bölünmez: 5 grup.



Şekil 4.5 Küçültmenin 3 turu: her turda bir grup, aynı harfle farklı gruba gidenlere göre bölünüyor.

Her grup bir durum olunca 8 durum 5'e iner: eşitlik (24-24, 25-25), A bir önde (25-24, 26-25), B bir önde (24-25, 25-26), A kazandı, B kazandı. Makinenin bilmesi gereken skor değil, farkmış. Bu, ikinci makinenin ta kendisi: başlangıçtan okları izleyince her durum ötekinde tam bir duruma denk gelir.



Şekil 4.6 Gruplar tek durum olunca 8 durum 5'e iniyor ve ikinci makineyle birebir örtüşüyor.

Bu bir rastlantı değil. Bir DFA'nın tanıyabildiği her dilin, yani her düzenli dilin, tek bir en küçük DFA'sı vardır; durumların adı değişebilir, okların düzeni değişmez. İki makinenin aynı dili tanıyıp tanımadığını anlamak için ikisini de küçültüp karşılaştırmak yeter. Örtüşmezlerse farkı gösteren bir dizgi de bulunur.

Sonsuz sayıda maç gidişinin hiçbirini tek tek denemedik; 8 durumu 3 turda gruplayıp karşılaştırmak yetti. Sonlu bir makine sonsuz bir dili anlatır; iki makineyi karşılaştırmak da bu yüzden sonlu bir iş.

Özet

- Alt küme kurulumu, NFA'nın bulunabileceği durum kümelerini yeni bir DFA'nın durumları yapar; her NFA için bir DFA kurulur, iki makine aynı dili tanır.
- Sondan üçüncü harf dilinde 4 durumlu NFA'ya karşılık 8 durumlu DFA gerekir: her küme dizginin son 3 harfini yazar ve daha azı yetmez.
- Sondan n'inci harf dilinde NFA n + 1 durumludur, DFA 2^n durum ister; tahmin güç değil kısalık katar.
- Geleceği aynı olan durumlar birleştirilebilir; DFA küçültme kabul ve ret gruplarından başlayıp gruplar bölünmeyene kadar sürer.
- Her düzenli dilin tek bir en küçük DFA'sı vardır; iki makineyi küçültüp karşılaştırmak aynı dili tanıyıp tanımadıklarını sonlu adımda söyler.

Sorular

1 **Alt küme kurulumunda kurulan DFA'nın bir durumu neyi temsil eder?**

A NFA'nın tek bir durumunu

B O ana kadar okunan dizgiyle NFA'nın aynı anda bulunabildiği durumların kümesini

C NFA'nın kabul durumlarından birini

D Dizginin henüz okunmamış kısmını

2 **4 durumlu bir NFA'dan alt küme kurulumuyla DFA kurulacak. Durumların kaç farklı alt kümesi vardır (boş küme dahil, ulaşılıp ulaşılamadığına bakmadan)?**

A 8

B 4

C 12

D 16

3 **Sondan üçüncü harf NFA'sının alt küme kurulumunda {bekle, 1 kaldı} kümesinden 1 harfiyle hangi kümeye geçilir? (bekle: 0 ve 1 ile bekle'ye, 1 ile ayrıca 2 kaldı'ya; 2 kaldı: her harfle 1 kaldı'ya; 1 kaldı: her harfle 0 kaldı'ya)**

A {bekle, 1 kaldı, 0 kaldı}

B {bekle, 2 kaldı, 1 kaldı}

C {bekle, 2 kaldı, 0 kaldı}

D {bekle, 0 kaldı}

4 **Sondan üçüncü harf DFA'sında 1 0 1 1 0 0 1 dizgisi okununca vardığımız küme, dizginin son üç harfiyle nasıl yazılır?**

Cevap:

5 **Voleybol makinesinde hangi iki durumun geleceği aynıdır, yani birleştirilebilir?**

A 24-24 ile 25-25

B 25-24 ile 24-25

C A kazandı ile B kazandı

D 24-24 ile A kazandı

6 **DFA küçültmede önce durumlar kabul ve ret olarak 2 gruba ayrıldı. Voleybol makinesinin ret grubunda ilk turda hangi iki durum ayrılır?**

A 24-24 ile 25-25

B 25-24 ile 26-25

C 24-25 ile 25-26

D 25-24 ile 24-25

7 **İki DFA'nın aynı dili tanıyıp tanımadığı hangi yolla sonlu adımda anlaşılır?**

A Sonsuz sayıda dizginin hepsi iki makineye de verilir.

B Durum sayıları karşılaştırılır; aynıysa makineler eşdeğerdir.

C Kabul durumlarının sayısına bakılır.

D İkisi de küçültülür ve en küçük makineler okların düzeniyle karşılaştırılır.

8 **Sondan altıncı harfi 1 olan dizgileri tanıyan en küçük DFA kaç durumludur?**

Cevap:

BİÇİMSEL DİLLER VE OTOMATA

5 Düzenli ifadeler

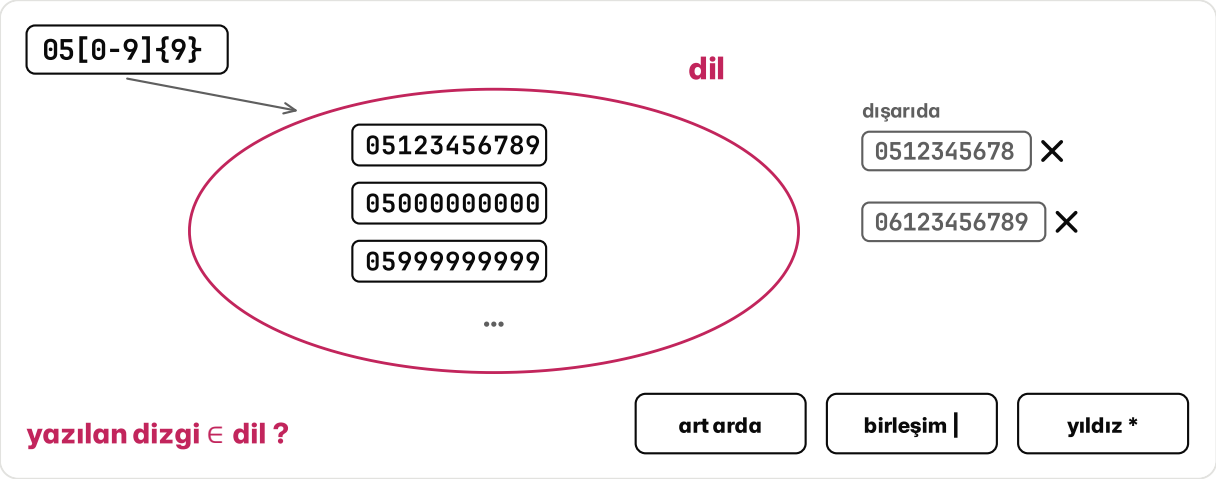
Bu bölümde dilleri tek satırla anlatan düzenli ifadeleri, üç temel işlemi, bir düzenli ifadeyi makineye çeviren Thompson kurulumunu ve bir makineyi tek satırlık kurala çeviren durum sökmeyi göreceğiz; sonunda düzenli ifadelerle sonlu otomatların aynı dilleri anlattığını söyleyen Kleene teoremine varacağız.

Tek satırlık kural

videoda 30:25

Bir forma cep telefonu numarası yazılırken bir rakam eksikse ya da araya bir harf karışmışsa form daha gönder tuşuna basmadan uyarır. Arkada tek satırlık bir kural var: 05'le başla, ardından 9 rakam gelsin. Bu kısa satır yazılan her şeyi nasıl denetler?

Böyle bir satıra **düzenli ifade** (regular expression, kısaca regex) denir. Düzenli ifade, kendisine uyan dizgilerin (string), yani harflerden oluşan dizilerin kümesini anlatır; bu küme bir dildir. Form tek bir soru sorar: yazılan dizgi bu dilde mi? Satır kısa görünür, ama her parçası yalnız 3 temel işlemden kurulur.



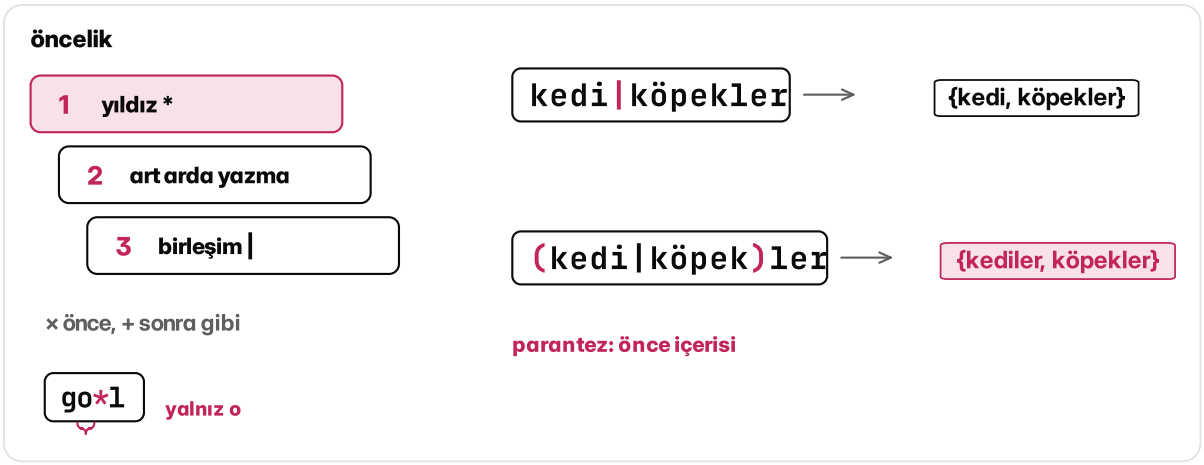
Şekil 5.1 Kural kutusundan "dil" halkasına ok: uyan dizgiler halkanın içinde, uymayanlar dışında.

Birinci işlem **art arda yazma**: 05, önce 0 sonra 5 demektir; ifadeler yan yana yazılınca uyan dizgiler de uç uca eklenir. İkinci işlem **birleşim**, araya dik bir çizgi konarak yazılır: iki ifadeden birine uyan dizgi yeter. kedi|köpek ifadesi tam 2 dizgiye uyar: kedi ve köpek.

Üçüncü işlem **yıldız**: bir ifadeyi 0 ya da daha çok kez art arda yazmak. gol kelimesindeki o harfine yıldız koyarsak, yani go*l yazarsak, ifade gol, goool, gooooool gibi bütün uzatılmış gol çığlıklarına uyar. o harfi hiç olmayabilir de; yani ifade gl dizgisine de uyar. Yıldız adını, düzenli ifadeleri 1951'de ortaya koyan matematikçi Stephen Kleene'den alır: Kleene yıldızı.

Bu işlemlerin bir öncelik sırası var: önce yıldız, sonra art arda yazma, en son birleşim; dört işlemde çarpmanın toplamadan önce gelmesi gibi. Yıldız bu yüzden yalnız hemen solundaki parçaya uygulanır. kedi|köpekler ifadesi kedi ve köpekler dizgilerine uyar; çoğul eki ikisine birden gelsin

istiyorsak kedi|köpek parantez içine alınır: (kedi|köpek)ler ifadesi kediler ve köpekler dizgilerine uyar.



Şekil 5.2 İşlem önceliği (yıldız, art arda, birleşim) ve parantezin kedi|köpekler ifadesinin anlamını nasıl değiştirdiği.

Uygulamadaki kısaltmalar da bu 3 işleme iner:

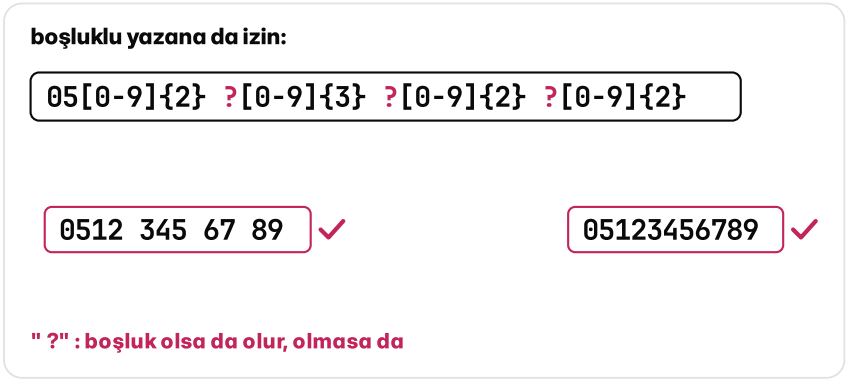
```
o+       =  oo*              en az 1 kez
o?       =  (o|ε)            olsa da olur, olmasa da
[0-9]    =  0|1|2|3|4|5|6|7|8|9
x{9}     =  xxxxxxxxx        9 kez art arda
```

go+l ifadesi gol'e uyar, gl'ye uymaz. ? işareti, hiç harfi olmayan boş dizgiyle birleşimdir.

Telefon kuralını açalım: önce 0, sonra 5, sonra 9 kez "0 ya da 1 ya da 2", böyle 9'a kadar. 0512 345 67 89 yazımındaki rakamlar uyar: 11 rakam, başı 05. 10 rakamlı numara uymaz, araya harf karışan uymaz, 06 ile başlayan da uymaz. Boşluklu yazana da izin vermek için gruplar arasına birer soru işaretli boşluk konur:

```
05[0-9]{2} ?[0-9]{3} ?[0-9]{2} ?[0-9]{2}
```

Boşluk olsa da olur, olmasa da: hem 0512 345 67 89 hem 05123456789 uyar.



Şekil 5.3 05[0-9]{9} kuralının açılımı ve 11 karelik dizgi şeridi; ret kartları: 10 rakam, harf, 06.

Kural yalnız biçime bakar; numaranın gerçekten birine ait olup olmadığını bilmez. Ama yanlış yazılmış bir numarayı, daha gönder tuşuna basılmadan, tek satırla yakalar.

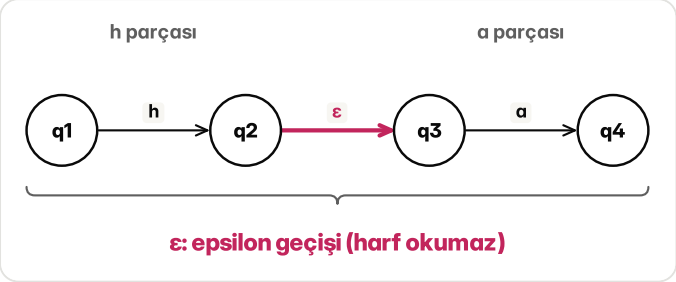
Kalıptan makineye

videoda 33:03

ha, haha, hahaha: bütün kahkahalar tek bir kalıba sığar. Düzenli ifadede ha parantez içine alınır, yanına yıldız konur: (ha)*. Program bu kalıbı olduğu gibi kullanmaz; önce onu harf harf okuyan bir makineye çevirir, denetimi o makine yapar. Bu çeviri nasıl yapılır?

Düzenli ifade 3 işlemden kurulur: art arda yazma, birleşim ve yıldız. Her işlemin küçük bir makine parçası var; her parçanın tek girişi, tek çıkışı olur. Parçalar deterministik olmayan sonlu otomat (NFA) parçalarıdır: bu makine aynı anda birden çok durumda bulunabilir, harf okumadan geçilen oklar da taşıyabilir.

En küçük parça tek bir harf: 2 durum ve aralarında o harfle bir ok. ha için 2 parça çizeriz: biri h harfi, öteki a harfi için. İkisini art arda yazmak için ilkinin çıkışından ikincisinin girişine harf okumayan bir ok çizeriz. Adı epsilon geçişi; makine bu oktan harf harcamadan geçer.

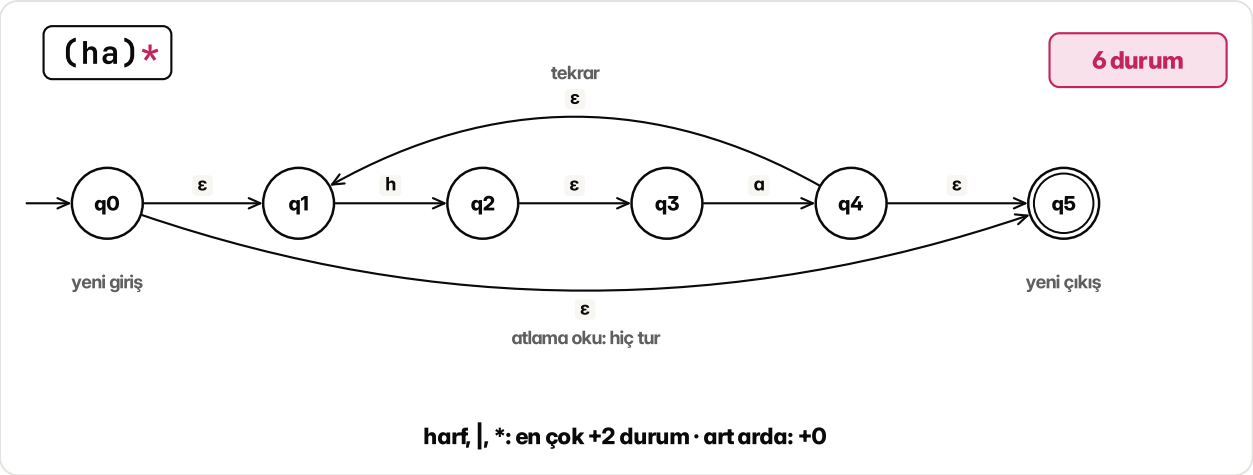


Şekil 5.4 h ve a parçaları, ilkinin çıkışından ikincisinin girişine çizilen ε oku ile art arda bağlanıyor.

Birleşimde yeni bir giriş, iki parçanın girişine epsilon geçişleriyle bağlanır; iki parçanın çıkışı da yeni bir çıkışa. Makine hangi yoldan gideceğini tahmin eder; yollardan birinin çıkışa varması yeter.

Yıldız için de yeni bir giriş ve yeni bir çıkış konur. Girişten parçaya bir epsilon geçişi gider. Parçanın çıkışından 2 epsilon geçişi çıkar: biri girişine geri döner, istenildiği kadar tur için; öteki yeni çıkışa gider. Bir epsilon geçişi de girişten doğrudan çıkışa uzanır, hiç tur atmamak için: adı **atlama oku**. ha parçasına yıldız eklenince makine 6 durum olur. Her harf, her birleşim ve her yıldız en çok 2 durum ekler; kalıp uzadıkça makine de ancak o kadar büyür.

```
q0  yeni giriş      q5  yeni çıkış (kabul)
q1 -h→ q2           q3 -a→ q4
ε okları:  q0→q1   q2→q3   q4→q1   q4→q5   q0→q5
```

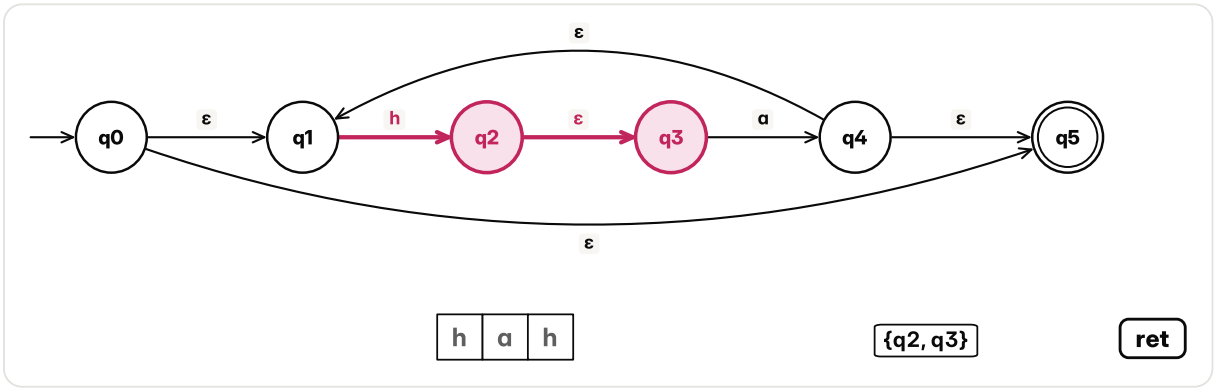


Şekil 5.5 (ha)* makinesi: yeni giriş q0, yeni çıkış q5, geri dönüş oku ve atlama oku.

haha dizgisini deneyelim. Makine işe 3 durumda birden başlar: yeni giriş, h parçasının girişi ve atlama okuyla varılan çıkış. h okununca h parçasının çıkışına geçer, epsilon geçişiyle a parçasının girişine de. a okununca a parçasının çıkışına varır, oradan 2 epsilon geçişiyle hem başa hem çıkışa: bir ha bitti. İkinci ha aynı turu yineler.

```
başlangıç     {q0, q1, q5}
h okununca    {q2, q3}
a okununca    {q1, q4, q5}
h okununca    {q2, q3}
a okununca    {q1, q4, q5}
```

Dizgi biterken makine çıkıştadır (q5 kümede): kabul. hah dizgisinde ise son h'den sonra küme {q2, q3} olur, çıkışa varılamaz: ret.



Şekil 5.6 haha okunurken kümelerin değişimi; q5 dolu olduğunda dizgi kabul, hah dizgisinde ret.

Atlama oku sayesinde hiç harfi olmayan boş dizgi de kabul edilir; yıldız 0 kez de demektir. Boş dizgiyi istemiyorsak yalnız bu oku sileriz: kalıp (ha)+ olur, yani en az bir ha.

Bu tarifi 1968'de Ken Thompson yayımladı ve kendi metin düzenleyicisine koydu; ertesi yıl Dennis Ritchie'yle Unix'i yazmaya başlayacaktı. O günlerde metin düzenleyicilerde arama çoğunlukla harfi harfine aynı kelimeyi bulmaktı; bugün bul-değiştir kutularına bu kalıplar yazılır. grep adlı arama programını da ilk o yazdı; grep bugün de kalıbı önce bir makineye çevirir, metni bir kez soldan sağa okuyarak arar.

Makineden kurala

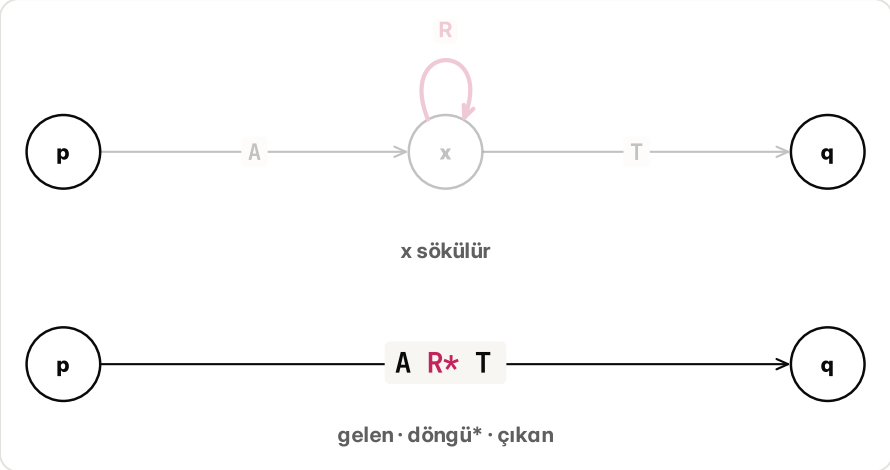
videoda 35:39

Elimizde bir makine çizimi var: 7, 250, 12,5 gibi sayıları tanıyor; virgülle biten ya da virgülle başlayan dizgiyi reddediyor. Bu çizimi tek satırlık bir düzenli ifadeye çevirebilir miyiz? Daha önemlisi: her makine böyle bir satıra sığar mı?

Makinenin 4 durumu var. başla'dan bir rakamla tam sayıya geçilir; tam sayı her rakamda kendine döner ve kabul durumudur. Virgül gelince virgül durumuna, oradan bir rakamla ondalığa geçilir; ondalık da rakamlarda kendine döner ve kabul durumudur. Oku olmayan harf gelirse ret. 12,5 yolu: başla → tam sayı → tam sayı → virgül → ondalık; kabul. ,5 dizgisinde başla'dan "," için ok yok: ret.

Fikir şu: oklara tek harf yerine bütün bir düzenli ifade yazılabilsin; sonra durumları birer birer sökelim. Önce 2 kabul durumundan yeni, tek bir bitişe harf okumayan birer ok çizeriz: epsilon geçişi. Artık bir başlangıç, bir bitiş var; aradakiler sökülecek.

Bir durum sökülünce, ondan geçen her yol iki ucu arasında tek bir oka yazılır: önce gelen okun ifadesi, sonra kendine dönen okun ifadesi yıldızlı, sonra çıkan okun ifadesi. Kendine dönen ok yıldıza dönüşür, çünkü o döngüde istenildiği kadar dönülebilir; hiç dönülmeyebilir de.

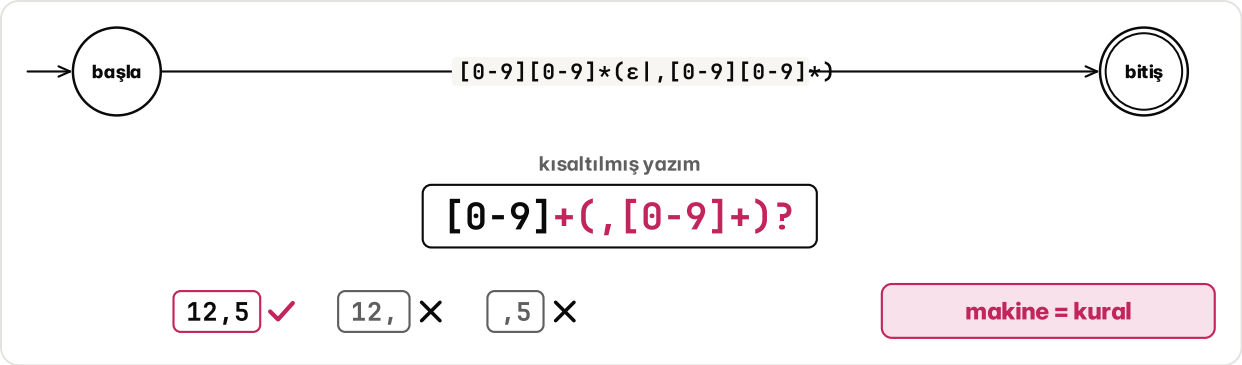


Şekil 5.7 Söküm kuralı: gelen okun ifadesi, kendine dönenin yıldızlısı, çıkan okun ifadesi tek oka yazılıyor.

Adım adım: önce ondalığı sökelim; sonra virgülü; son olarak tam sayıyı.

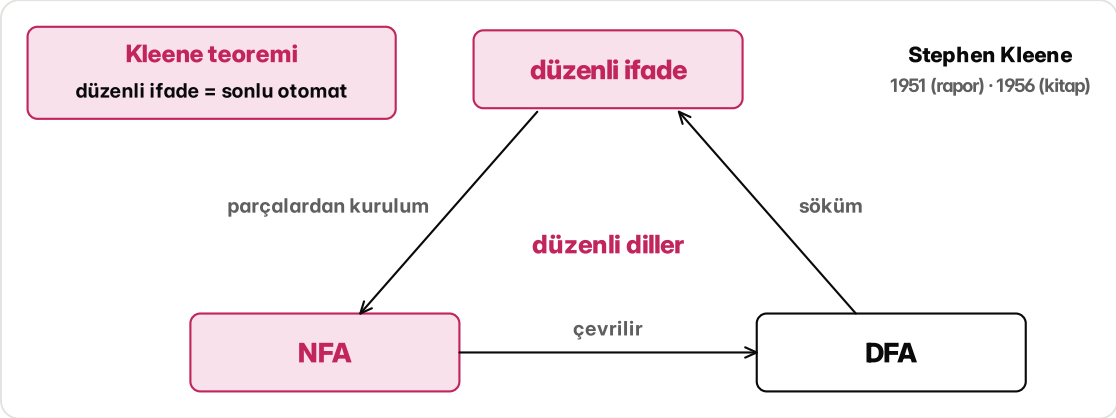
```
ondalık sökülünce    virgül –[0-9][0-9]*→ bitiş
virgül sökülünce     tam sayı –,[0-9][0-9]*→ bitiş
                     tam sayı –ε→ bitiş
                     ikisi birleşir:  ε|,[0-9][0-9]*
tam sayı sökülünce   başla –[0-9][0-9]*(ε|,[0-9][0-9]*)→ bitiş
```

Tam sayıdan bitişe virgül sökülünce 2 ok oluşur: hemen bitirmek ya da ondalık kısmı eklemek; ikisi birleşimle tek oka iner. Başla ile bitiş arasında tek bir ok kaldı; üstündeki ifade bütün makinenin işini görür: 1 ya da daha çok rakam, ardından olsa da olur olmasa da virgül ve rakamlar, kısaca [0-9]+(,[0-9]+)?. 12,5 uyar; virgülle biten ya da başlayan dizgi uymaz: makineyle kural aynı dizgileri seçer.



Şekil 5.8 Söküm bitince başla ile bitiş arasında tek ok kalır; ifadesi [0-9]+(,[0-9]+)? makinenin işini görür.

Söküm her sonlu otomatta işler; ters yönde, her düzenli ifadeden de bir makine kurulur. NFA de her zaman deterministik sonlu otomata (DFA) çevrilir. Demek ki düzenli ifade, NFA ve DFA aynı dilleri anlatır; bu dillere **düzenli dil** denir. Düzenli ifadelerle sonlu otomatların aynı dilleri anlattığını Stephen Kleene 1951'de, sinir hücresi ağları üzerine yazdığı bir raporda gösterdi: **Kleene teoremi**.



Şekil 5.9 Üç kutulu çember: düzenli ifade, NFA ve DFA aynı düzenli dilleri anlatıyor.

Bazı programlama dillerinin regex'lerine eklenen geri başvuru ise bu çemberin dışına çıkar: daha önce yakalanan parçanın aynısını yeniden ister. "ve ve" gibi yinelenen kelimeyi bulmak için ilk kelime harfi harfine akılda tutulmalı; kelime istenildiği kadar uzun olabildiğinden bunu hiçbir sonlu otomat yapamaz.

Düzenli ifadeler her gün derleyicinin ilk aşamasında, sözcük çözümleyicide çalışır. Koddaki her parça türü bir ifadeyle tarif edilir: adlar, sayılar, işaretler. Çözümleyici bu tarifleri tek makinede birleştirir, satırı tek geçişte parçalarına ayırır: int x = 42; satırı tür adı, değişken adı, eşittir, 42, noktalı virgül olur. 42'yi tanıyan parça, bu sayı makinesinin virgülsüz hâlidir.

Özet

- Düzenli ifade, bir dili 3 işlemle anlatır: art arda yazma, birleşim ve Kleene yıldızı (0 ya da daha çok kez). Öncelik sırası yıldız, art arda yazma, birleşimdir; +, ?, [0-9] ve {9} birer kısaltmadır.
- Thompson kurulumunda her harf, birleşim ve yıldız en çok 2 durum ekler; art arda yazma epsilon geçişiyle olur. (ha)* makinesi 6 durumludur.
- Yıldız parçasındaki atlama oku boş dizgiyi de kabul ettirir; silinirse kalıp (ha)+ olur.
- Durum sökmede gelen ok, kendine dönenin yıldızlısı ve çıkan ok tek oka yazılır; [0-9]+(, [0-9]+)? ondalık sayı makinesinin kuralıdır.
- Kleene teoremi: düzenli ifade, NFA ve DFA aynı dilleri, yani düzenli dilleri anlatır. Geri başvuru bu çemberin dışındadır.

Sorular

1 **(kedi|köpek)ler düzenli ifadesine hangi dizgi uyar?**

A kedi
B köpek
C kediler
D ler

2 **go+l düzenli ifadesine hangi dizgi uymaz?**

A gol
B goooool
C gool
D gl

3 **ab|c* düzenli ifadesine hangi dizgi uyar?**

A abab
B abcc
C acb
D ccc

4 **(ha)* ifadesinin Thompson makinesi hangi dizgiyi kabul etmez?**

A (boş dizgi)
B ha
C hahaha
D hahah

5 **Bir makinede p durumundan x'e A ifadeli, x'ten q'ya T ifadeli bir ok var; x'in kendine dönen oku R ifadeli. x sökülünce p'den q'ya giden oka hangi ifade yazılır?**

A A R T
B A R* T
C A* R T
D (A|R|T)*

6 **Geri başvuru (daha önce yakalanan parçanın aynısını yeniden istemek) neden düzenli ifade, NFA ve DFA'nın anlattığı dillerin dışına çıkar?**

A Çünkü geri başvuru yıldız işlemini kullanmaz.

B Çünkü makine harf okumadan ok geçemez.

C Çünkü ilk parça istenildiği kadar uzun olabilir ve sonlu bir makine onu harfi harfine aklında tutamaz.

D Çünkü bu dillerde kabul durumu birden çok olamaz.

7 **a|b düzenli ifadesinin Thompson makinesi kaç durumludur? (Her harf için 2 durumlu bir parça, birleşim için yeni bir giriş ve yeni bir çıkış.)**

Cevap: _____

8 **05[0-9]{2} ?[0-9]{3} ?[0-9]{2} ?[0-9]{2} kuralına uyan en uzun dizgi kaç karakterdir? (Boşluklar da sayılır.)**

Cevap: _____

BİÇİMSEL DİLLER VE OTOMATA

6 Pompalama önsavı ve kapalılık

Bu bölümde bazı dillerin hiçbir sonlu otomatla ve hiçbir düzenli ifadeyle denetlenemediğini görürüz. Önce bunun nedenine bakarız, sonra bir dilin düzenli olmadığını ispatlamanın yolunu (pompalama önsavı) öğreniriz, en sonda da iki makineyi birleştirerek yeni makine kurmayı ve düzenli dillerin işlemlerde kapalı kalmasını inceleriz.

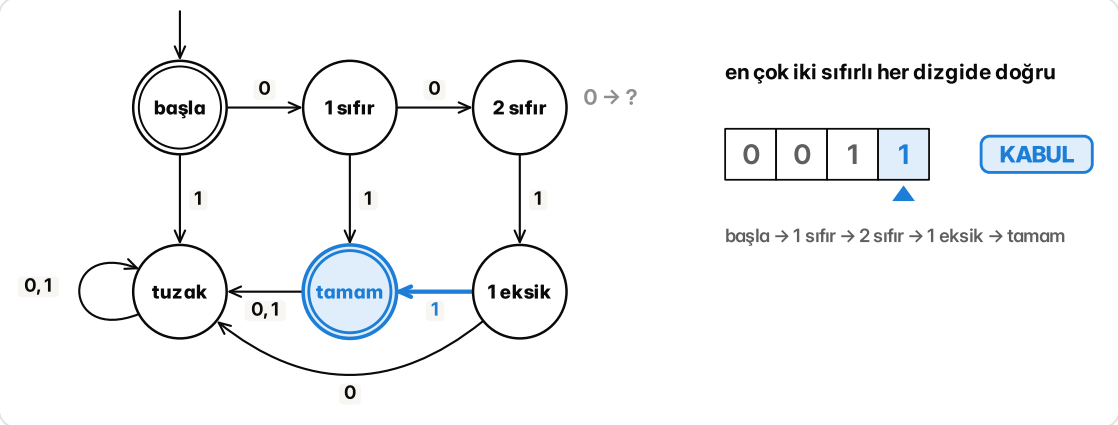
Basit bir kural

videoda 38:28

Şu kuralı alalım: önce n tane 0, sonra tam n tane 1. 01 kurala uyar, 0011 de uyar; 001 uymaz, çünkü birler eksiktir. Bu dile kısaca 0^n1^n diyeceğiz. Kural bu kadar basit olduğu hâlde hiçbir **düzenli ifade** (regular expression), yani metinde kalıp arayan tek satırlık kural, onu denetleyemez.

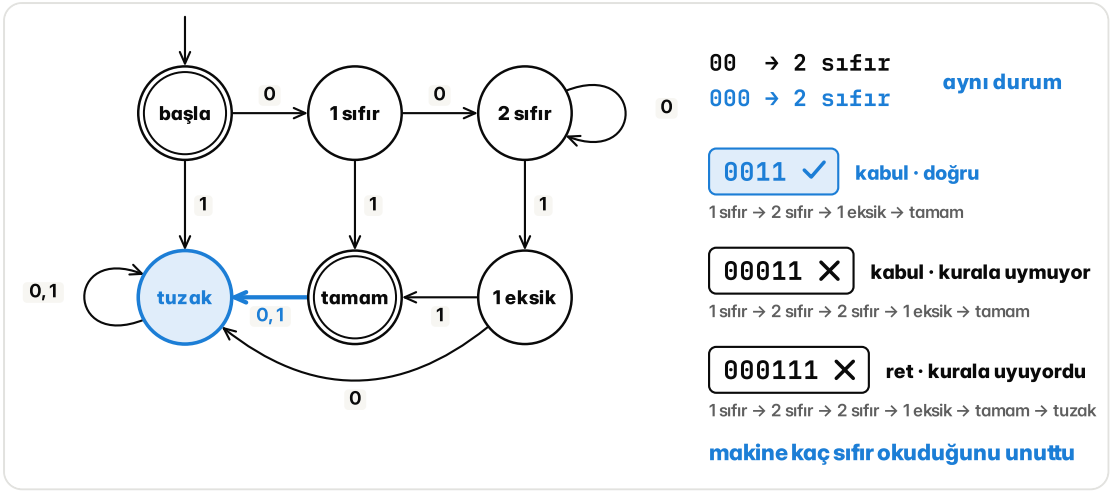
İki deneme yapalım. İlk deneme "istediğin kadar 0, ardından istediğin kadar 1" demektir: 0*1*. Bu ifade sırayı denetler ama sayıya bakmaz; 0001'i de kabul eder. İkinci deneme, kurala uyan dizgileri (string), yani harflerden oluşan dizileri tek tek yazmaktır: 01|0011|000111. Bu ifade 4 sıfırlı dizgide tıkanır. Her yeni sayı için ifadeye yeni bir parça gerekir; ifade hiç bitmez.

Sorunun kökü düzenli ifadede değil. Düzenli ifadeyle yazılan her kuralı bir **sonlu otomat** (finite automaton) da denetler: harfleri tek tek okuyup durumdan duruma geçen, tek belleği o anki durumu olan bir makine. O hâlde bir otomatla sayalım. Makine 0 geldikçe "1 sıfır", "2 sıfır" diye ilerler; 1 gelince geri sayar: "1 eksik", sonra "tamam". Sıra bozulursa tuzak durumuna düşer. Hiç harf yoksa da kural sağlanır (0 tane 0 ve 0 tane 1), bu yüzden başlangıç durumu da kabul durumudur. Bu 6 durumlu makine 2 sıfıra kadar doğru karar verir.



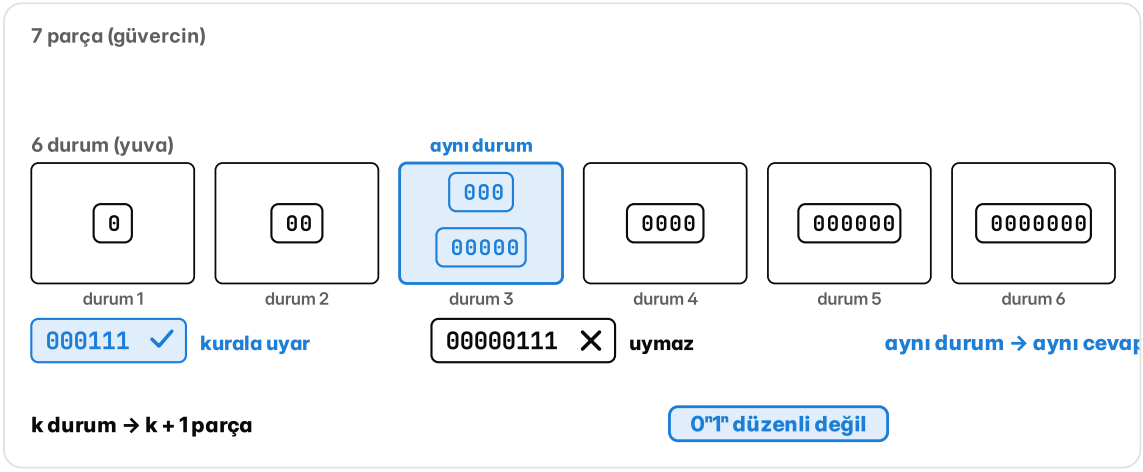
Şekil 6.1 Sayan makinenin 6 durumu: başla, 1 sıfır, 2 sıfır, 1 eksik, tamam ve tuzak; 0011 şeridi başla, 1 sıfır, 2 sıfır, 1 eksik, tamam yolunu izleyip kabul ediyor.

Üçüncü sıfır gelince ne olacak? Her durumdan her harf için bir ok çıkmalı ve yeni durum da yok. Makinenin 2 sıfırda kaldığını varsayalım: artık 2 sıfır ile 3 sıfır aynı durumda. 0011 kabul edilir, doğru. Ama 00011 de kabul edilir, oysa kurala uymaz; 000111 ise reddedilir, oysa kurala uyar. Makine kaç sıfır okuduğunu unuttu.



Şekil 6.2 Üçüncü sıfırın oku 2 sıfır durumuna dönüyor; 00011 yanlışlıkla kabul, 000111 yanlışlıkla ret.

Durum eklemek de kurtarmaz. 6 durumlu herhangi bir makineye 1 sıfırdan 7 sıfıra kadar 7 parça okutalım. 7 parça, 6 durum: **güvercin yuvası ilkesine** (pigeonhole principle) göre ikisi aynı durumda biter; diyelim 3 sıfır ile 5 sıfır. İkisine de 3 tane 1 ekleyelim: 000111 kurala uyar, 00000111 uymaz. Ama makine ikisinde de aynı durumdan aynı harfleri okur, aynı cevabı verir; biri mutlaka yanlıştır. Kaç durum olursa olsun bir fazla parça alırız. Sonlu otomatın tanıyabildiği dillere **düzenli dil** denir; 0^n1^n düzenli değildir.



Şekil 6.3 6 durum kutusuna 7 sıfır kartı düşüyor; 000 ile 00000 aynı kutuda, 000111 kabul, 00000111 ret ve aynı durum aynı cevabı veriyor.

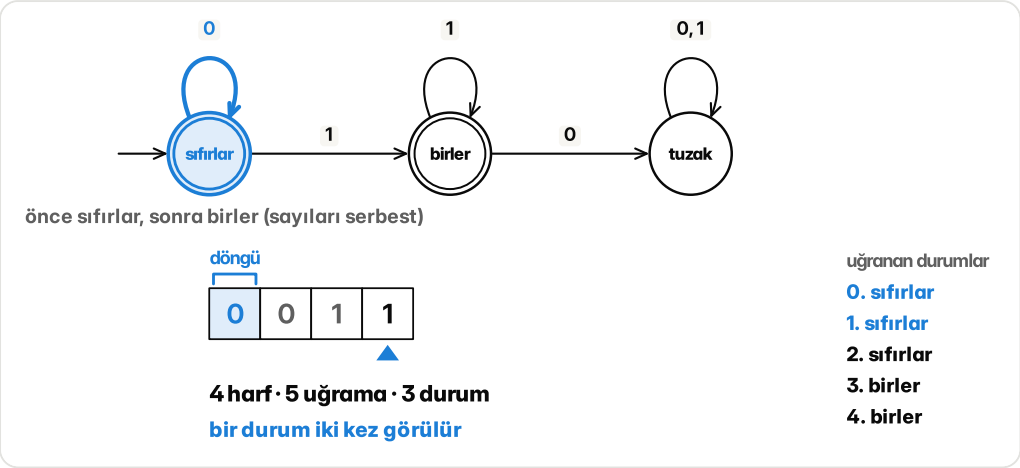
Bu kural gerçekte de karşımıza çıkar. İnternet sayfaları iç içe etiketlerle yazılır: açılan her etiketin bir kapanışı vardır ve kaç kat iç içe gidileceği belli değildir. En basit hâli tam bu kuraldır: n tane açılış, ardından n tane kapanış. Düzenli ifade etiketleri tek tek bulur ama hepsinin doğru kapandığını denetleyemez. Bazı programlama dilleri düzenli ifadeye özel eklentiler koyar; bunlar sınırı aşar, ama artık sonlu bir otomatla çalışmaz. İç içe yapıyı denetlemek için açılışları sayabilen, daha güçlü bir makine gerekir.

Sonu olmayan makineler

videoda 40:55

Bir kuralı hiçbir sonlu otomatın denetleyemeyeceğini nasıl ispatlarız? Makinelerin sonu yok, hepsini tek tek deneyemeyiz. Cevap, her makinenin içinde saklı bir döngüdedir.

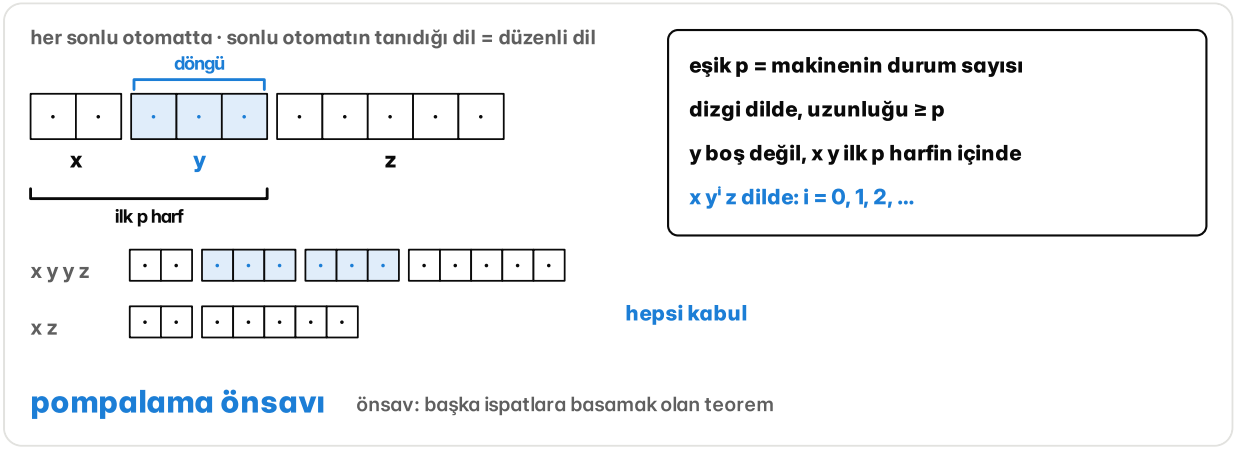
Önce "önce 0'lar, sonra 1'ler" kuralını alalım: önce 0'lar gelir, sonra 1'ler; kaç tane oldukları serbesttir. Onu 3 durumlu bir makine denetler: sıfırlar, birler ve tuzak. Ona 0011 okutalım. 4 harfi okurken makine, başlangıç dahil, 5 kez bir duruma uğrar: sıfırlar, sıfırlar, sıfırlar, birler, birler. Oysa 3 durumu var. Güvercin yuvası ilkesine göre bir durum 2 kez görülür. Burada ilk 0 makineyi sıfırlardan alıp yine sıfırlara bırakır: bu harf bir döngüdür.



Şekil 6.4 3 durumlu makine 0011 okurken 4 harfte 5 kez bir duruma uğruyor; ilk 0 döngü yayı olarak işaretli.

Döngüdeki parça makineyi geldiği yere geri getirir. Onu 2 kez okutsak da hiç okutmasak da makine aynı yerden devam eder ve aynı cevabı verir: 00011 kabul, 011 kabul. "Önce 0'lar, sonra 1'ler" için bu doğrudur, çünkü 0'ların sayısı serbesttir. Ama aynı işlem, 0^n1^n kuralına uyan 0011'i bozar: 00011 o kurala uymaz.

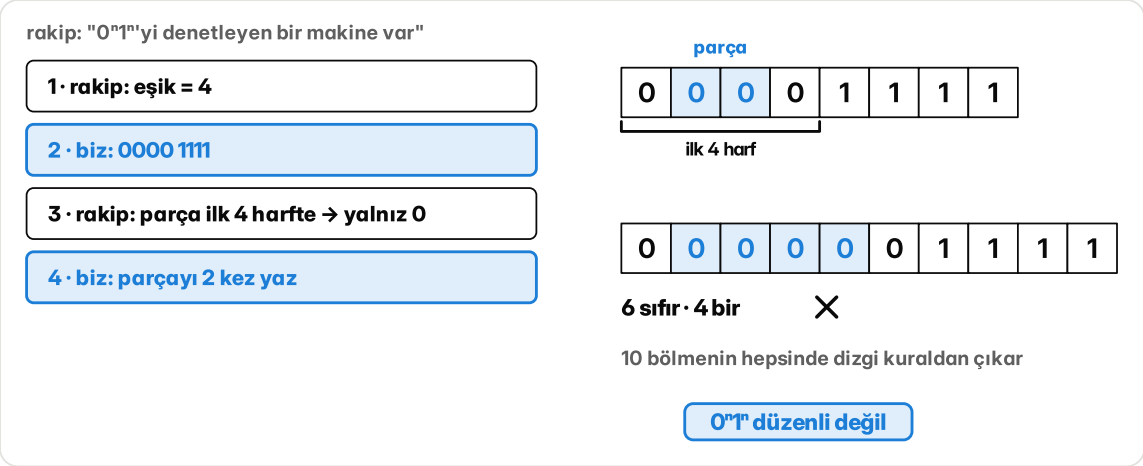
Bu her sonlu otomatta olur. Makinenin durum sayısına **eşik** (p) diyelim. Makinenin kabul ettiği, eşik kadar ya da daha uzun her dizgi x, y ve z diye üç parçaya ayrılabilir: x y ilk p harfin içinde kalır, y boş değildir ve x y^i z dizgisi i = 0, 1, 2, ... için kabul edilir; yani y istediğimiz kadar tekrarlanabilir ya da silinebilir. Buna **pompalama önsavı** (pumping lemma) denir; önsav, başka ispatlara basamak olan teorem demektir.



Şekil 6.5 Şerit x, y, z diye üç bölmeye ayrılmış, y'nin üstünde döngü yayı var; x y^i z dilde olmalı.

Önsavı bir oyun gibi kullanırız. Rakibimiz, 0^n1^n kuralını denetleyen bir makinesi olduğunu söylüyor. Eşiği o söyler: 4. Dizgiyi biz seçeriz: 4 sıfır, 4 bir (00001111). Rakip parçayı ilk 4 harften seçmek zorundadır, yani parça yalnız 0'lardan oluşur; diyelim ortadaki 2 sıfırı seçti. Son hamle bizim:

parçayı 2 kez yazarız. Sonuç 6 sıfır ve 4 bir olur; dizgi kuraldan çıkar. Parça hangisi olursa olsun 0'lar artar, 1'ler aynı kalır; eşik kaç olursa olsun aynı hamle kazanır. Demek ki böyle bir makine yok, dil düzenli değildir.



Şekil 6.6 Oyun tablosu: rakip eşiği 4 söylüyor, biz 00001111'i seçiyoruz, rakip ortadaki 2 sıfırı seçiyor, biz parçayı 2 kez yazıyoruz; sonuç 6 sıfır ve 4 bir.

İkinci örnek: uzunluğu asal sayı olan dizgiler. Rakip eşiği yine 4 söylesin; biz 7 harflik bir dizgi seçeriz, 7 asaldır. Rakip ilk 4 harften bir parça seçer, diyelim 2 harf. Biz o parçayı 7 kez fazladan yazarız: uzunluk 7 + 14 = 21 olur ve 21 = 7 × 3 asal değildir. Parça kaç harf olursa olsun eklenen uzunluk 7'nin katıdır, toplam da. Asalların sonu olmadığı için her eşikten uzun bir asal vardır; bu dil de düzenli değildir.

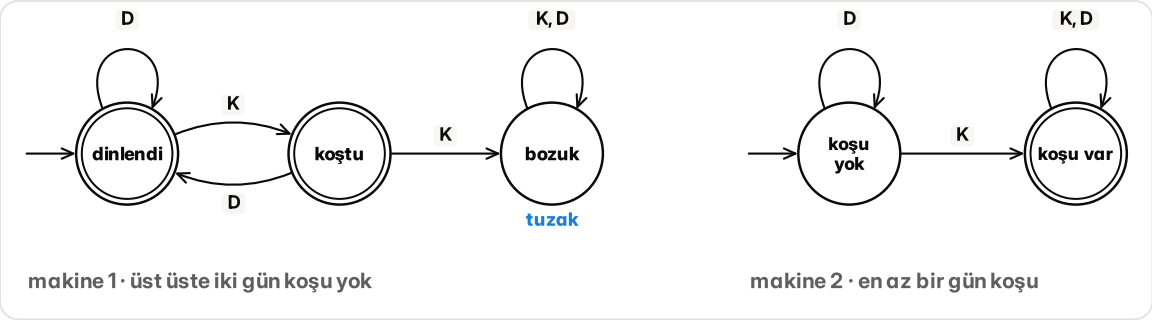
Önsav tek yönde çalışır. Oyunu kazanmak, dilin düzenli olmadığını kesin ispatlar. Pompalamaya dayanan bir dil ise yine de düzenli olmayabilir. Bir dilin düzenli olduğunu göstermenin yolu, onu tanıyan makineyi çizmektir.

İki kural

videoda 43:32

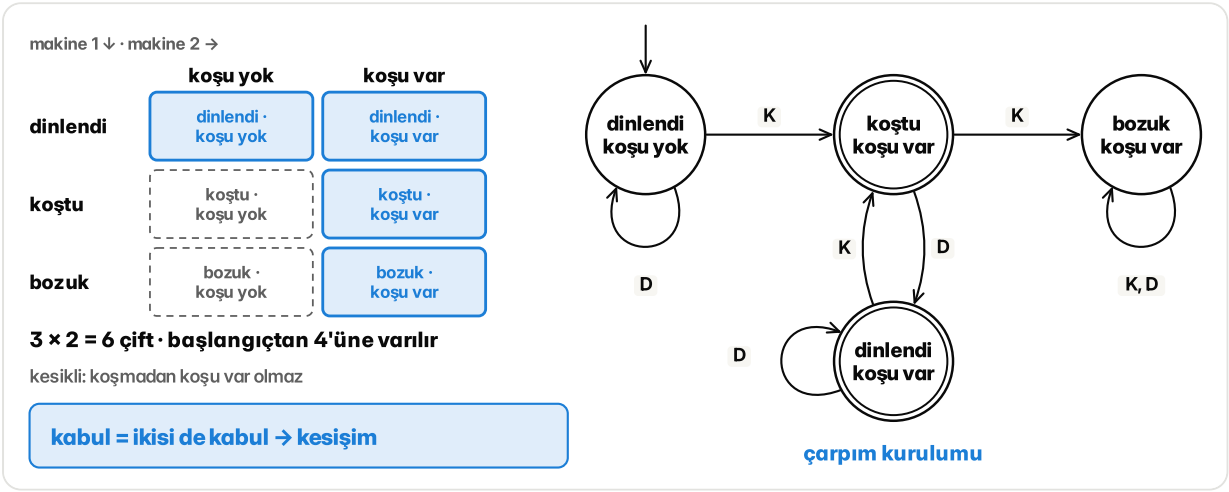
Bir koşu planı gün gün yazılıyor; plan bir dizgi: K harfi koşu günü, D harfi dinlenme günü. Plan 2 kurala uymalıdır: üst üste 2 gün koşulmaz ve en az 1 gün koşulur. Her kuralın küçük bir makinesi hazır. İkisini birden denetleyen makineyi, hazır makinelere dokunmadan nasıl kurarız?

Birinci makinenin 3 durumu var: dinlendi, koştu ve bozuk. K, dinlendiden koştuya, koştudan bozuğa götürür; D, dinlendiye döndürür (koştudan da dinlendiye). Bozuk bir tuzaktır: bir kez girilince çıkılmaz. İkinci makinenin 2 durumu var: koşu yok ve koşu var. İlk K onu koşu vara geçirir, orada kalır.



Şekil 6.7 Makine 1 (dinlendi, koştu, bozuk) ve makine 2 (koşu yok, koşu var) yan yana; bozuk durumu tuzak olarak işaretli.

İki makineye aynı planı aynı anda okutalım ve her gün ikisinin durumunu bir çift olarak yazalım; her çift yeni makinenin bir durumudur. 3 × 2 = 6 çift olabilir, ama başlangıçtan yalnız 4'üne varılır: koşmadan birinci makine koştuya ya da bozuğa gelemez. Yeni makine, ikisi de kabul durumundaysa kabul eder. Buna **çarpım kurulumu** (product construction) denir; iki kurala birden uyan dizgileri, yani iki dilin kesişimini tanır.

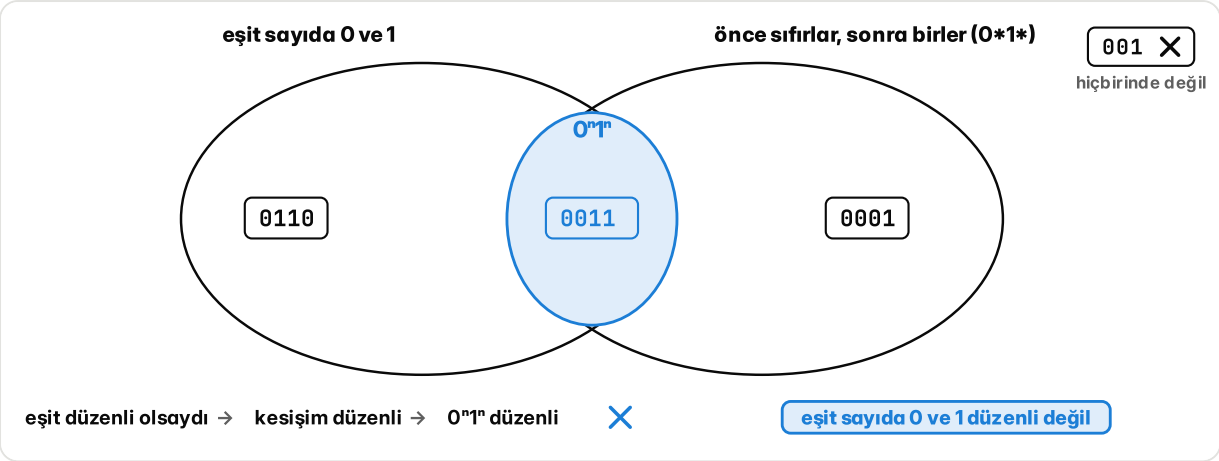


Şekil 6.8 3 × 2 çarpım ızgarası: başlangıçtan varılan 4 çift çerçeveli, varılmayan 2 çift kesik çizgili; sağda 4 durumlu çarpım makinesi.

Planları deneyelim. DKDK: çiftler "dinlendi · koşu yok"tan başlar, "koştu · koşu var"da biter; ikisi de kabul, plan geçer. DKKD: 3. gün birinci makine bozuğa düşer, plan geçmez. DDD: birinci makine kabul eder ama ikincisi "koşu yok"ta kalır; plan yine geçmez.

Aynı çiftlerle, en az biri kabul ettiğinde kabul edersek birleşimi tanırız. Tümleyen için kabul ile ret yer değiştirir. İki dili art arda eklemek ve bir dili istediğimiz kadar tekrarlamak, makineleri ε geçişleriyle, yani harf okumayan oklarla bağlayarak kurulur. Tersten okumak için bütün oklar ters çevrilir, başlangıç ile kabul yer değiştirir. Sonuç her seferinde yine sonlu bir otomattır. Bu yüzden düzenli diller bu işlemlerde **kapalıdır**: işlemden yine düzenli dil çıkar.

Kapalılık ispatta da işe yarar. Eşit sayıda 0 ve 1 içeren dizgiler düzenli mi? 0110 uyar, 001 uymaz. Düzenli olsaydı, "önce 0'lar, sonra 1'ler" kuralıyla kesişimi de düzenli olurdu; o kuralı da 3 durumlu bir makine denetler. Kesişimde hem eşit hem sıralı dizgiler kalır: önce n tane 0, sonra n tane 1. Oysa hiçbir sonlu otomat kaç sıfır okuduğunu sınırsız sayamaz. Demek ki eşit sayıda 0 ve 1 de düzenli değildir.



Şekil 6.9 İki halka: eşit sayıda 0 ve 1 ile 0*1*; kesişimde 0011 duruyor ve etiket 0ⁿ1ⁿ.

Düzenli dillerde bazı sorular makineye bakarak cevaplanır. Dil boş mu: başlangıçtan bir kabul durumuna yol var mı? Plan makinesinde var: tek bir koşu günü. Dil sonsuz mu: o yolda bir döngü var mı? Var: dinlenme ve koşu sırayla sonsuza dek sürer. İki makine aynı dili mi tanıyor: ikisini yan yana çalıştırıp, birinin kabul ötekinin ret dediği bir çifte yol ararız. Yalnız birinci kuralın makinesi, tek günlük bir dinlenme planında (D) yeni makineden ayrılır. Makine sonlu, bakılacak durumlar da sonlu; cevap her zaman kesin bulunur.

Özet

- 0^n1^n düzenli değildir: sonlu otomat kaç sıfır okuduğunu sınırsız sayamaz, k durumlu her makinede k + 1 parçadan ikisi aynı durumda biter.
- Her sonlu otomatta uzun dizgilerin içinde bir durum tekrar eder; arada kalan parça tekrarlanabilir ya da silinebilir (pompalama önsavı).
- Düzenli olmadığını ispatlamak için oyun oynanır: rakip eşiği ve parçayı seçer, biz dizgiyi seçip parçayı tekrarlayarak dili terk ettiririz.
- Önsav tek yönlüdür: pompalanamayan dil düzenli değildir, pompalanan dil düzenli olabilir de olmayabilir de.
- Düzenli diller birleşim, kesişim, tümleyen, art arda ekleme, yineleme ve tersine çevirmede kapalıdır; çarpım kurulumu iki makineyi yan yana çalıştırır.
- Düzenli dillerde boşluk, sonsuzluk ve eşdeğerlik soruları makineye bakarak kesin cevaplanır.

Sorular

1 0*1* düzenli ifadesi aşağıdaki dizgilerden hangisini kabul eder ama "önce n tane 0, sonra tam n tane 1" kuralına uymaz?

A 0011
B 01
C 00011
D 000111

2 9 durumlu bir makineye 1 sıfırdan başlayarak 0, 00, 000, ... diye uzayan parçaları okutuyoruz. İki parçanın aynı durumda bitmesini garanti etmek için en az kaç parça okutmak gerekir?

A 9
B 10
C 18
D 8

3 Birinci makinenin 3, ikinci makinenin 5 durumu var. Çarpım kurulumuyla iki kuralı birden denetleyen makine kurulursa durum sayısı en çok kaç olur?

A 15
B 8
C 5
D 3

4 Koşu planı makinesi iki kurala bakar: üst üste 2 gün koşulmaz ve en az 1 gün koşulur (K koşu, D dinlenme). Makineye KDKK planını okutuyoruz. Plan kabul mü edilir, ret mi? "kabul" ya da "ret" yaz.

Cevap:

5 Önce n tane 0, sonra n tane 1 kuralı için pompalama oyunu oynuyoruz. Rakip eşiği 3 söyledi, biz 000111 dizgisini seçtik. Rakip parça olarak ilk harfi (tek bir 0) seçti. Parçayı 2 kez yazınca çıkan dizgiyi yaz.

Cevap:

6 Uzunluğu asal olan dizgilerin dilinde rakip eşiği 4 söyledi. Biz 5 harflik bir dizgi seçtik (5 asal) ve rakip 2 harflik bir parça seçti. Parçayı 5 kez fazladan yazarsak dizginin uzunluğu kaç olur?

Cevap:

7 **Bir dilin uzun dizgilerinde tekrarlanıp silinebilen, dilden çıkarmayan bir parça bulunuyor (dil pompalama önsavının koşulunu sağlıyor). Bu durumda ne söyleyebiliriz?**

A Dilin düzenli olduğu kesinleşir

B Dilin düzenli olmadığı kesinleşir

C Dilin sonlu olduğu kesinleşir

D Hiçbir şey kesinleşmez; dil düzenli de olabilir, olmayabilir de

8 **Eşit sayıda 0 ve 1 içeren dizgiler dilinin düzenli olmadığını kapalılıkla göstermek için bu dil hangi düzenli dille kesiştirilir?**

A $(01)^*$

B 0^*1^*

C 0^*

D 1^*

BİÇİMSEL DİLLER VE OTOMATA

7 Bağlamdan bağımsız gramerler

Bu bölümde bir dilin kurallarını yerine koyma adımlarıyla yazmayı öğreniriz: önce Türkçe bir cümlenin iskeleti üzerinde, sonra sonlu otomatın sayamadığı 0^n1^n dili ve programlama dillerinin iç içe yapısı üzerinde. Sonda aynı simge dizisine iki ayrı anlam veren gramerlerin (belirsizlik) nasıl sorun çıkardığını ve nasıl çözüldüğünü görürüz.

Kedi mektubu yazdı

videoda 46:20

"Kedi mektubu yazdı." cümlesinde bir tuhaflık var: kediler mektup yazmaz. Yine de cümle düzgün kurulmuştur; Türkçe bilen herkes onu dilbilgisine uygun bulur. Peki bir makine, sözcüklerin anlamını hiç bilmeden, bir cümlenin düzgün kurulup kurulmadığını nasıl anlar?

Cümle 3 parçalıdır: önce özne, sonra nesne, en sonda yüklem. Bunu bir **kural** olarak yazabiliriz; ok, solundaki parçanın yerine sağındakilerin konabileceğini söyler. Her parçanın da kendi kuralı vardır:

```
Cümle → Özne Nesne Yüklem
Özne → Mert | Ayşe | kedi
Nesne → mektubu | sütü
Yüklem → yazdı | içti
```

Dikey çizgi "ya da" demektir. Bu kuralların hepsine birden **gramer** (grammar) denir. Yerine başka bir şey konan Cümle, Özne, Nesne ve Yüklem **değişkenlerdir**; türetmenin sonunda kalan sözcükler **uç simgelerdir** (terminal). Türetme, **başlangıç değişkeninden**, burada Cümle'den başlar.



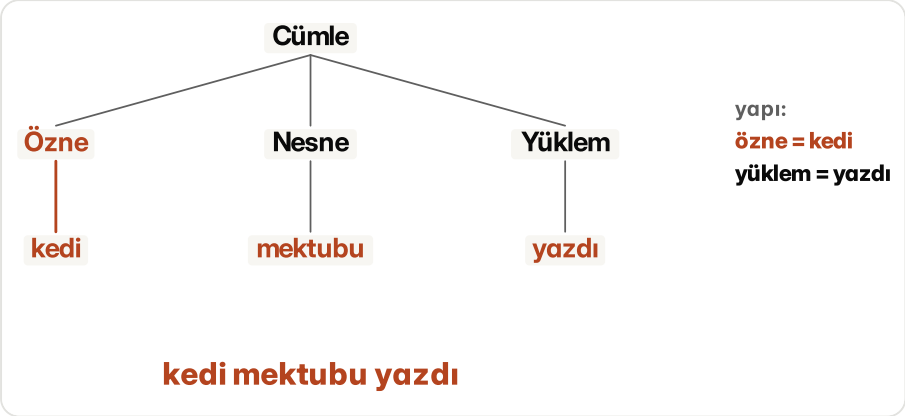
Şekil 7.1 Cümle kartı Kedi, mektubu, yazdı diye üçe ayrılmış; altında özne, nesne, yüklem ve gramerin kuralları.

Cümleden başlayıp her adımda bir parçayı kuralına göre değiştirelim:

```
Cümle ⇒ Özne Nesne Yüklem
      ⇒ kedi Nesne Yüklem
      ⇒ kedi mektubu Yüklem
      ⇒ kedi mektubu yazdı
```

Konacak parça kalmayınca cümle hazırdır. Bu adımlar dizisine **türetme** (derivation) denir. Her kuralın solunda tek bir değişken vardır ve kural, değişkenin çevresine bakılmadan uygulanır: Özne yerine kedi koymak için Nesne'ye sorulmaz. Çevredeki parçalara **bağlam** denir; bu yüzden adı **bağlamdan bağımsız gramer** (context-free grammar) olur.

Türetmeyi bir **türetme ağacı** olarak da çizebiliriz: kökte Cümle durur; altında Özne, Nesne ve Yüklem; her birinin altında da yerine konan sözcük. Yaprakları soldan sağa okuyunca cümle çıkar. Ağaç adımların sırasını değil, cümlenin yapısını gösterir: hangi sözcük özne, hangisi yüklem.

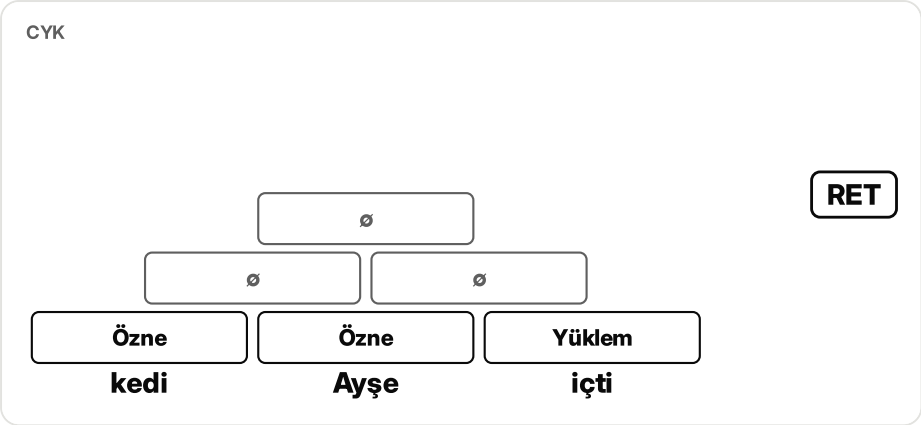


Şekil 7.2 Türetme ağacı: kökte Cümle, altında Özne, Nesne, Yüklem, yapraklarda kedi, mektubu, yazdı.

3 özne, 2 nesne ve 2 yüklemle bu küçük gramer 3 × 2 × 2 = 12 cümle üretir: Mert mektubu yazdı da, kedi mektubu yazdı da, Ayşe mektubu içti de. Gramer sözcüklerin anlamına değil, yalnız dizilişine bakar. Dil bilimci Chomsky 1956'da, dillerin cümle yapısını böyle yerine koyma kurallarıyla anlatan bir model önerdi.

Makine bir cümlenin bu gramerden türeyip türemediğini nasıl bulur? Önce her kuralın sağında ya 2 değişken ya da tek bir sözcük kalmasını sağlar. Cümle kuralı ikiye bölünür: Cümle, Özne ve Öbek olur; Öbek de Nesne ve Yüklem. Bu kalıba **Chomsky normal biçimi** denir.

Sonra alt satırda her sözcüğün değişkeni yazılı bir üçgen tablo kurar. "kedi sütü içti" için alt satır Özne, Nesne, Yüklem'dir. Bir üst satır yan yana 2 sözcüğe bakar: "Özne Nesne" diye bir kural yok, hücre boştur; "Nesne Yüklem" ise Öbek'tir. Tepede Özne ve Öbek, yani Cümle bulunur: kabul. "kedi Ayşe içti" dizilişinde alt satır Özne, Özne, Yüklem olur ve üst hücreler boş kalır: ret.



Şekil 7.3 Üçgen tablo: alt satırda Özne, Nesne, Yüklem; "kedi sütü" hücresi boş, "sütü içti" hücresi Öbek, tepede Cümle ve kabul.

Her hücre altındaki parçanın bütün bölünüşlerine baktığı için iş, sözcük sayısının küpüyle büyür: sözcük sayısı 2 katına çıkınca iş 8 katına çıkar. Örneğin denenen bölünüş 10 sözcükte 165, 20 sözcükte 1.330'dur. Yine de yöntem her bağlamdan bağımsız gramerde çalışır ve sonunda kesin bir evet ya da hayır verir.

Sayamayan makine

videoda 49:02

Önce n tane 0, sonra n tane 1: 01; 0011; 000111. Belleği yalnız o anki durumu olan sonlu bir otomat bu dizgileri tanıyamaz. Kaç 0 okuduğunu aklında tutmalıdır, 0'lar ise bir gün durum sayısını aşar. Bu kuralı eksiksiz yazan 2 satır var mı?

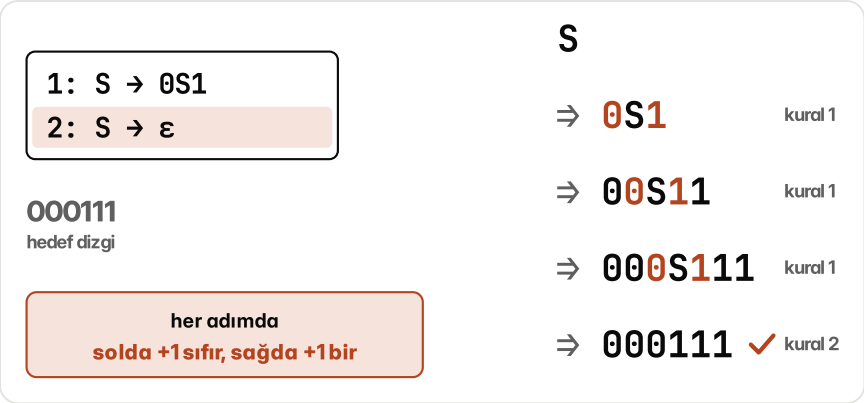
Var. Bir gramer, yerine bir şey konan değişkenlerden, yerine bir şey konmayan uç simgelerden ve kurallardan oluşur. Bu dil için tek bir değişken yeter: S. Uç simgeler 0 ve 1. Kurallar 2 tane:

```
S → 0S1 | ε
```

S, 0S1 olur ya da boş dizgi (ε) olur. Birinci kuralın sağında S yeniden geçiyor: kural kendini çağırıyor ve istediğimiz kadar tekrarlanabiliyor. 000111'i türetelim:

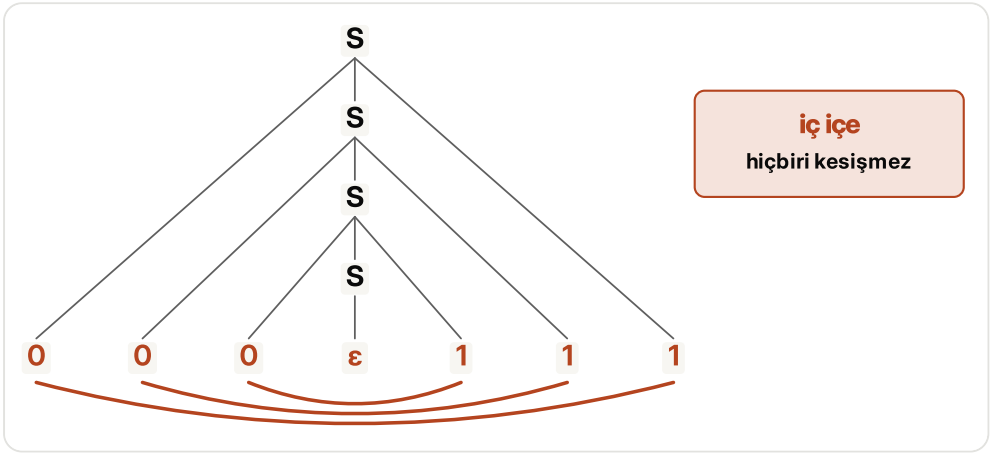
```
S ⇒ 0S1 ⇒ 00S11 ⇒ 000S111 ⇒ 000111
```

Son adımda ikinci kural S'yi boş dizgiyle siler. Her adımda sola bir 0, sağa bir tane 1 eklendi. Türetmenin her adımında dizgi aynı biçimdedir: birkaç 0, ortada S, aynı sayıda 1. S silinince 0'lar önde, 1'ler arkada kalır ve sayıları eşittir. 0101 ya da 001 hiçbir yoldan türemez. S'nin yerine ne konacağı çevresine bakmadığı için bu bir bağlamdan bağımsız gramerdir; 2 satır, sonlu otomatın tanıyamadığı dili eksiksiz yazıyor.



Şekil 7.4 S ⇒ 0S1 ⇒ 00S11 ⇒ 000S111 ⇒ 000111 türetmesi; her adımda eklenen 0 ve 1 vurgulu.

Türetme ağacında S'ler iç içe durur: her S'nin solunda bir 0, sağında bir tane 1, ortasında yeni bir S vardır. En dıştaki 0 en sondaki 1'e, ikinci 0 sondan ikinci 1'e bağlanır. Eşleşen çiftler birbirinin içinde kalır, hiçbiri ötekini kesmez.



Şekil 7.5 000111'in türetme ağacı: 4 S düğümü alt alta, eşleşen 0 ve 1 çiftleri iç içe yaylarla bağlı.

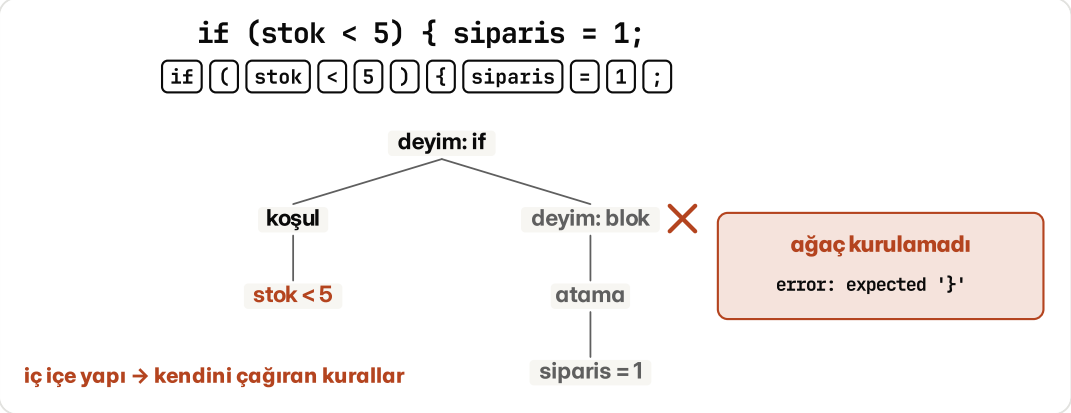
Kodda da yapı böyledir: if'in içinde if, onun içinde süslü parantezli bir blok; açılan her blok kendi kapanışıyla eşleşir. C dilinin kurallarından 3'ü özetle şöyledir:

```
deyim → if ( koşul ) deyim
      | { deyimler }
      | atama ;
deyimler → deyim deyimler | ε
```

Deyim ya if, parantez içinde koşul ve bir deyimdir; ya süslü parantez içinde deyimler; ya da atama ve noktalı virgül. Deyimin kuralında yine deyim geçer, tıpkı S'nin içindeki S gibi. Programlama dillerinin sözdizimi böyle kurallarla yazılır. Backus bu yazımı 1959'da önerdi; 1960'ta Naur'un düzenlediği bir programlama dili raporu dilin bütün sözdizimini onunla yazdı. Yazım bugün Backus-Naur biçimi diye anılır; C'nin standardı da sözdizimini aynı türden kurallarla verir.

Derleyici kodu önce parçalarına ayırır: if (stok < 5) { siparis = 1; } için if, parantez, stok, küçüktür, 5 ve devamı. Sonra **ayrıştırıcısı** (parser) bu parçaları kurallara uydurup türetme ağacını kurar: kökte if deyimi, dallarında koşul ve blok, bloğun içinde atama. Kapanış parantezi unutulursa blok

kuralı tamamlanamaz; ağaç kurulamaz ve derleyici sözdizimi hatası verir. Sonlu otomatın sayamadığı iç içe yapıyı kendini çağıran kurallar böyle yakalar.



Şekil 7.6 if (stok < 5) { siparis = 1; } kodunun parçalara ayrılması ve türetme ağacı; kapanış silinince ağaç kurulamıyor.

Bir else, iki if

videoda 51:47

Şu C kodunda iç içe 2 if var, ama tek bir else. Klima 0'sa kapalı, 1'se soğutma, 2'yse ısıtma demektir:

```
klima = 0;
if (derece > 25)
    if (nem > 60)
        klima = 1;
else
    klima = 2;
```

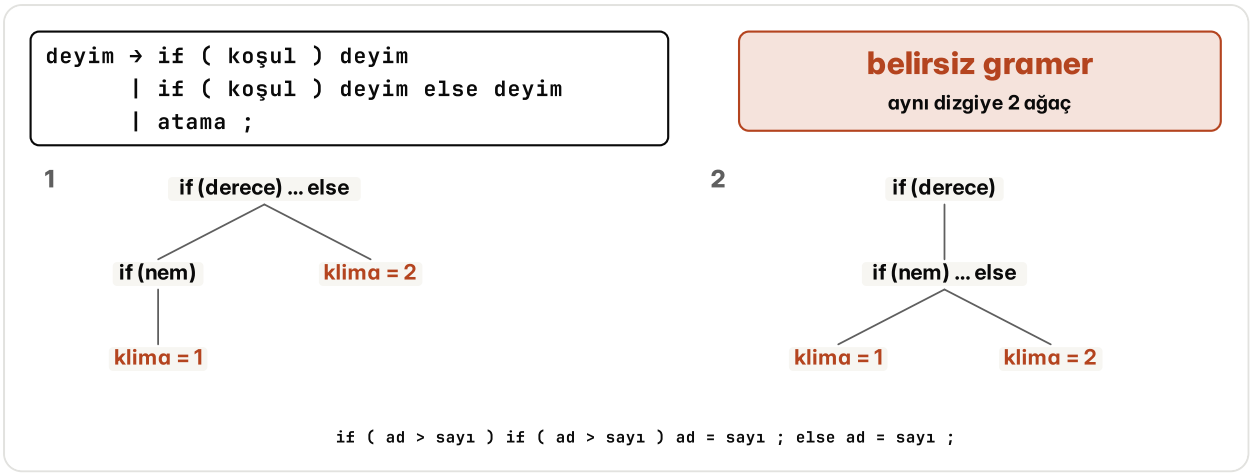
Girintiye bakılırsa else dıştaki if'e ait: hava sıcak değilse ısıt. Ama derleyici girintiye bakmaz. else hangi if'e aittir?

Somut sayılarla deneyelim: 30 derece, %40 nem. else dıştaki if'inse, sıcaklık koşulu doğru, nem koşulu yanlıştır; içteki if'in else'i yok, hiçbir şey olmaz ve klima kapalı kalır (0). else içteki if'inse, nem koşulu yanlış olunca else çalışır ve klima ısıtmaya geçer (2). Aynı satırlar, 2 ayrı program.

C'nin if kuralları kabaca şöyledir:

```
deyim → if ( koşul ) deyim
      | if ( koşul ) deyim else deyim
      | atama ;
```

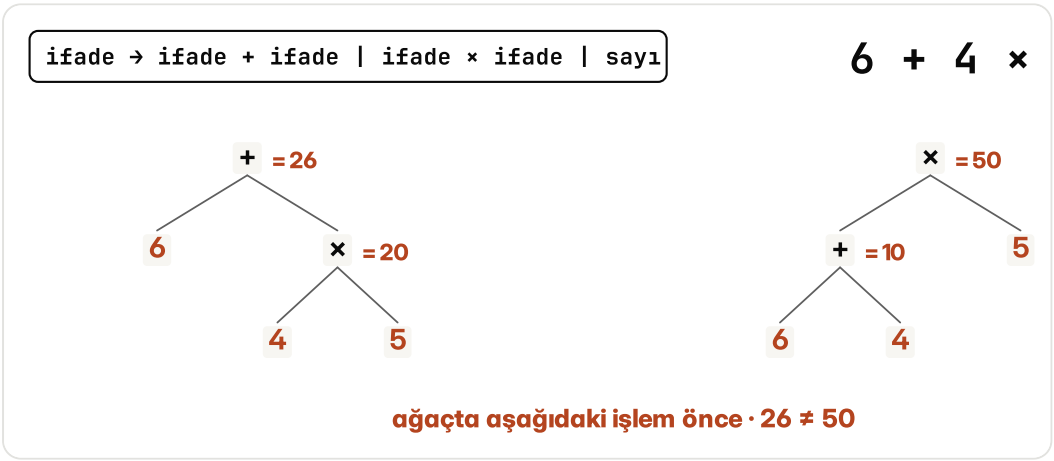
Bu kod 2 yoldan türetilebilir. Birinci ağaçta dıştaki if else'li kuralla türer, içteki if else'siz kalır. İkinci ağaçta dıştaki if else'sizdir, else içteki if'e bağlanır. Aynı simge dizisine 2 ayrı türetme ağacı veren gramere **belirsiz gramer** denir.



Şekil 7.7 Aynı simge dizisi için 2 türetme ağacı: birinde else dıştaki if'e, ötekinde içteki if'e bağlı; başlık belirsiz gramer.

C'nin standardı bu belirsizliği tek cümleyle keser: else, kendisinden önce gelen ve kurallara göre bağlanabilen en yakın if'e bağlanır. Bizim kodda o if içtekidir: 30 derecede klima ısıtmaya geçer, girinti ne derse desin. Derleyiciler bu yazımda uyarı da verir. Niyet dıştaki if'se, içteki if süslü paranteze alınır; o zaman tek ağaç kalır.

Aynı sorun aritmetikte de çıkar. Kurallar şöyle olsun: ifade ya 2 ifadenin toplamı, ya 2 ifadenin çarpımı ya da bir sayıdır. 6 + 4 × 5'in bu gramerde 2 ağacı var. Birinde çarpma aşağıdadır: önce 4 × 5 = 20, sonra 6 eklenir, 26. Ötekinde toplama aşağıdadır: önce 6 + 4 = 10, sonra 5'le çarpılır, 50. Ağaçta aşağıdaki işlem önce yapılır; 26 ile 50 aynı ifadenin iki farklı okumasıdır.



Şekil 7.8 6 + 4 × 5 için iki ağaç: solda toplama kökte, değerler 20 ve 26; sağda çarpma kökte, değerler 10 ve 50.

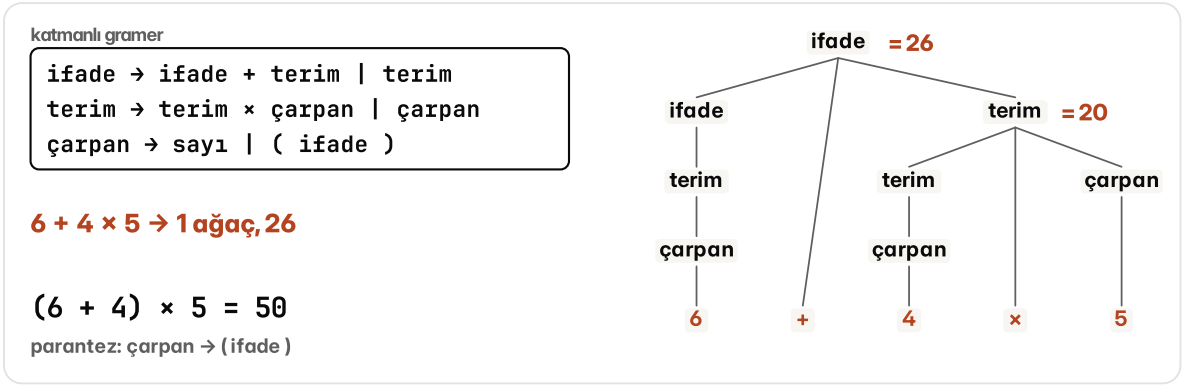
Çözüm, gramere katman eklemektir. İfade terimlerin toplamıdır; terim çarpanların çarpımıdır; çarpan da bir sayı ya da parantez içinde bir ifadedir:

```
ifade → ifade + terim | terim
terim → terim × çarpan | çarpan
çarpan → sayı | ( ifade )
```

Çarpma artık yalnız terimin içinde yapılabilir. 6 + 4 × 5'in tek ağacı kalır: 4 × 5 bir terim olur, 6 ona eklenir, 26. Önce toplamak isteyen parantez yazar: (6 + 4) × 5 = 50.

Ama her dilin tek ağaçlı bir grameri yoktur. Önce 0'lardan, sonra 1'lerden, sonra 2'lerden oluşan dizgileri alalım ve 0'ların sayısı 1'lere ya da 1'lerin sayısı 2'lere eşit olsun. 3 sayı birden eşitse (001122 gibi) dizgi 2 koşula da uyar; bu dili hangi gramerle yazarsak yazalım, böyle dizgilerin bazıları 2 ağaç alır. Belirsizlik gramerde değil, dilin kendisindedir.

Programlama dillerini tasarlayanlar gramerlerini bu yüzden katman katman, tek ağaç verecek biçimde kurar; kalan belirsizliği de else kuralı gibi açık bir cümleyle keser. Böylece aynı kod her derleyicide aynı ağaca, yani aynı programa dönüşür.



Şekil 7.9 Katmanlı gramerde 6 + 4 × 5'in tek ağacı: 4 × 5 bir terim, değerler 20 ve 26.

Özet

- Gramer; değişkenler, uç simgeler, kurallar ve bir başlangıç değişkeninden oluşur; türetme başlangıç değişkeninden uç simgelere yerine koyma adımlarıdır.
- Bağlamdan bağımsız gramerde her kuralın solunda tek değişken vardır; kural çevreye bakmadan uygulanır.
- S → 0S1 | ε, sonlu otomatın tanıyamadığı 0^n1^n dilini 2 satırda yazar; kendini çağıran kural iç içe yapıyı yakalar.
- Derleyicinin ayrıştırıcısı kodu parçalara ayırıp türetme ağacını kurar; ağaç kurulamazsa sözdizimi hatası verir.
- Aynı dizgiye 2 ağaç veren gramer belirsizdir: sarkan else (C'de else en yakın if'e bağlanır) ve 6 + 4 × 5 (26 ya da 50) örnekleridir.
- Katmanlı gramer belirsizliği giderir; ama bazı dillerin belirsizliği dilin kendisindedir ve her gramerde bazı dizgiler 2 ağaç alır.

Sorular

1 S → 0S1 | ε gramerinden aşağıdaki dizgilerden hangisi türetilebilir?

A 0011

B 0101

C 001

D 10

2 **Bir gramere "bağlamdan bağımsız" denmesinin nedeni hangisidir?**

A Kurallarında uç simge bulunmaması

B Her kuralın solunda tek bir değişken olması ve kuralın çevresine bakılmadan uygulanması

C Cümlelerin anlamına bakmaması

D Türetmenin her zaman soldan başlaması

3 **Gramer: Cümle → Özne Nesne Yüklem; Özne → Ali | Veli | Ece | Can; Nesne → ekmeği | suyu | sütü; Yüklem → yedi | içti. Bu gramer kaç farklı cümle üretir?**

Cevap: _____

4 **Aşağıdaki kurallardan hangisi Chomsky normal biçimindedir (sağında ya 2 değişken ya da tek bir sözcük olmalı)?**

A Cümle → Özne Nesne Yüklem

B Cümle → Özne Öbek Yüklem

C Öbek → Nesne Yüklem

D Öbek → Özne Nesne Yüklem

5 **S → 0S1 | ε gramerinde 00001111 dizgisini türetmek için toplam kaç kural adımı gerekir (ε kuralı dahil)?**

Cevap: _____

6 **Üçgen tablo yöntemiyle bir cümlenin gramerden türeyip türemediğine bakılırken iş sözcük sayısının küpüyle büyüyor. Sözcük sayısı 3 katına çıkarsa iş kaç katına çıkar?**

Cevap: _____

7 **Katmanlı gramerde (ifade → ifade + terim | terim; terim → terim × çarpan | çarpan; çarpan → sayı | (ifade)) 2 + 3 × 4 ifadesinin tek türetme ağacı hangi değeri verir?**

A 20

B 24

C 9

D 14

8 **Aşağıdaki C kodu derece = 18 ve nem = 70 değerleriyle çalışıyor. C'nin kuralına göre (else, en yakın if'e bağlanır) çalıştıktan sonra klima kaç olur?**

```
klima = 0;
if (derece > 25)
    if (nem > 60)
        klima = 1;
else
    klima = 2;
```

A 2

B 0

C 1

D Derleme hatası

BİÇİMSEL DİLLER VE OTOMATA

8 Yığıtlı otomatlar

Bu bölümde sonlu otomata bir yığın ekleyerek sayma ve eşleştirme yeteneği kazandırırız. Önce 0^n1^n dilini tanıyan makineyi kurarız, sonra tersten de aynı okunan dizgileri (palindrom) ve bağlamdan bağımsız gramerlerle olan eşdeğerliği inceleriz; sonda yığının sınırına, $a^nb^nc^n$ dilinde ulaşırız.

Sayamayan makine

videoda 49:02

Harfleri yalnız 0 ve 1 olan dizgileri düşünelim. Kural şu: önce n tane 0, sonra n tane 1 ($n \geq 1$). Sonlu otomat bu kuralı denetleyemez; kaç 0 okuduğunu yalnız durumlarında saklayabilir, durumlarının sayısı ise sınırlıdır. Makineye tek bir yığın eklersek ne olur?

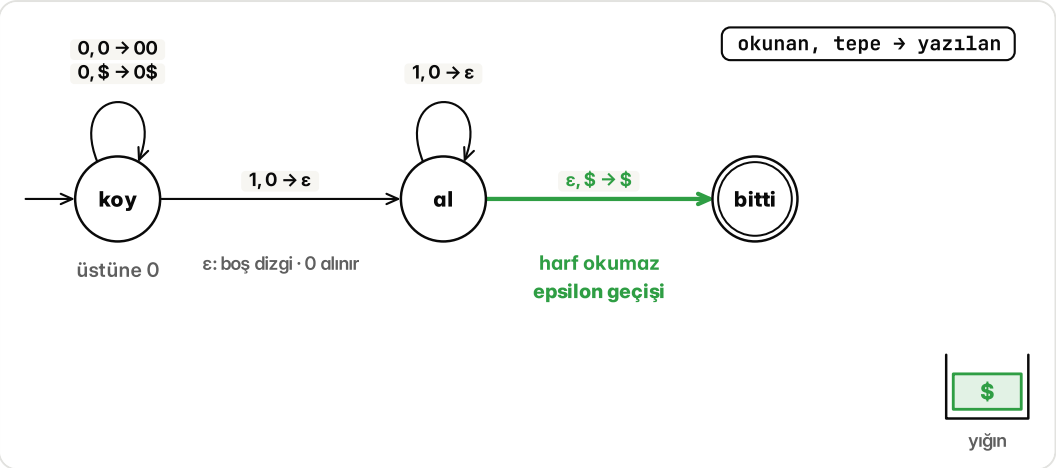
Yığın (stack), yalnız en üstüne konup en üstünden alınabilen bir bellektir; boyunun sınırı yoktur. Fikir şu: her 0 okununca yığına bir 0 koyarız, her 1 okununca yığından bir 0 alırız. 0'ların sayısı yığının yüksekliğinde saklanır. 1'ler bittiğinde yığın tam boşalmışsa iki sayı eşittir.

Makinenin yığının boşaldığını görebilmesi için yığının dibinde baştan bir **dip işareti** ($) durur. Durumlar 3 tanedir: koy, al ve bitti. Makine koy durumunda başlar ve 0'ları yığına koyar. İlk 1 gelince al durumuna geçer; artık yalnız alır. Bitti kabul durumudur.

Her ok 3 şey söyler: okunan harf, yığının tepesinde görülen simge ve onun yerine yazılan. Biz bunu "okunan, tepe → yazılan" diye yazarız. Makinenin 5 oku şöyledir:

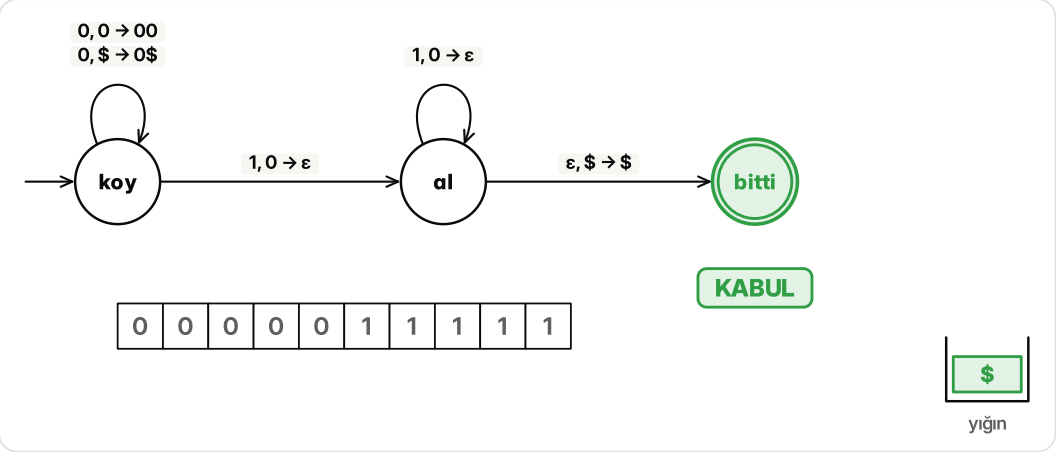
```
koy → koy:   0, $ → 0$
koy → koy:   0, 0 → 00
koy → al:    1, 0 → ε
al → al:     1, 0 → ε
al → bitti:  ε, $ → $
```

İlk iki ok, 0 okununca tepede ne olursa olsun üstüne bir 0 daha yazar. 1 okununca tepedeki 0'ın yerine boş dizgi (ε) yazılır, yani 0 alınır. Son ok hiç harf okumaz; harf okumayan bu oka **ε geçişi** denir: tepede dip işareti görünce makine harf harcamadan bittiye geçer.



Şekil 8.1 Koy, al ve bitti durumları ile 5 ok ve okların etiketleri; son ok ε geçişi olarak vurgulu.

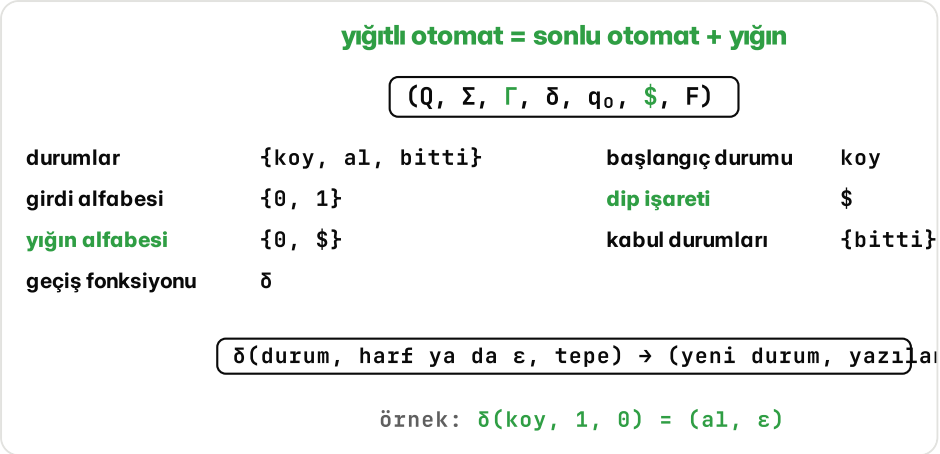
Çalıştıralım: 0000011111. 5 sıfırla yığına 5 sıfır girer; dip işaretiyle 6 kat olur. İlk 1 makineyi al durumuna geçirir ve bir 0 alır. Sonraki 4 bir, 4 sıfır daha alır. Dizgi bittiğinde tepede dip işareti vardır; makine ε geçişiyle bittiye geçer: kabul. Toplam 11 hamle yapılmıştır: 10 harf ve 1 ε geçişi.



Şekil 8.2 0000011111 okunurken yığın 6 kata çıkıyor, 1'lerde tek tek boşalıyor ve ε geçişiyle bitti durumuna geçip kabul ediliyor.

Üç reddi de izleyelim. 0000111: 1'ler bitince yığında bir 0 kalır, dip işareti görünmez: ret. 00111: üçüncü 1 geldiğinde yığında alınacak 0 kalmamıştır; 4 harften sonra yığın yalnız $ olur ve beşinci harf için ok yoktur: ret. 001011: sayılar eşittir (3 ve 3), ama al durumundayken bir 0 geliyor ve o durumda 0 okuyan ok yok: ret. Sırayı durumlar, sayıyı yığın denetler.

Sonlu otomata bir yığın eklenmiş bu makineye **yığıtlı otomat** (pushdown automaton) denir. Biçimsel tanımı 7 parçadır: durumlar, girdi alfabesi, yığın alfabesi, geçiş fonksiyonu, başlangıç durumu, dip işareti ve kabul durumları; kısaca $(Q, \Sigma, \Gamma, \delta, q_0, \$, F)$. Geçiş fonksiyonu 3 şeye bakar: o anki durum, okunan harf ya da harf okunmadığı, tepedeki simge. Cevabı yeni durum ve tepedekinin yerine yazılacak dizgidir; örneğin $\delta(\text{koy}, 1, 0) = (\text{al}, \varepsilon)$.



Şekil 8.3 Yığıtlı otomatın 7 parçalı tanımı ve bu makinedeki değerleri: durumlar {koy, al, bitti}, girdi alfabesi {0, 1}, yığın alfabesi {0, $}.

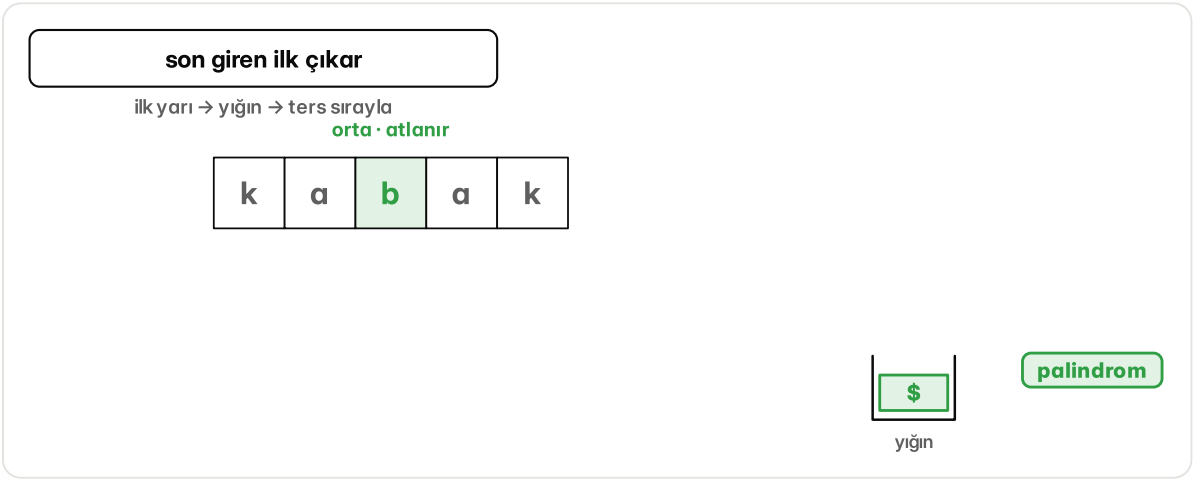
Makinenin yeni parçası yalnız yığındır. Yığının ortasına bakamaz, yalnız tepesini görür; ama yığın istediği kadar büyür. Sonlu sayıda durum ve yalnız tepeden açılan sınırsız bir bellek, 0'larla 1'lerin dengesini saymaya yeter.

Tersten de aynı

videoda 57:01

"Ey Edip Adana'da pide ye." cümlesini sondan başa okuyalım: harfler aynı sırayla gelir. Kabak da öyledir. Tersten de aynı okunan dizgilere **palindrom** denir. Harfleri bir kez, soldan sağa okuyan bir makine, bir dizginin palindrom olduğunu nasıl anlar?

Yığın tersine çevirir: son giren ilk çıkar. İlk yarıyı koyup geri alırsak harfler ters sırayla gelir. Kabak'ta k ve a yığına girer, ortadaki b atlanır. Gelen a tepedeki a ile, son k de altındaki k ile eşleşir. Yığın boşaldı, dizgi bitti: palindrom. Yığının içeriği sırayla $, k$, ak$, ak$, k$ ve $ olur.

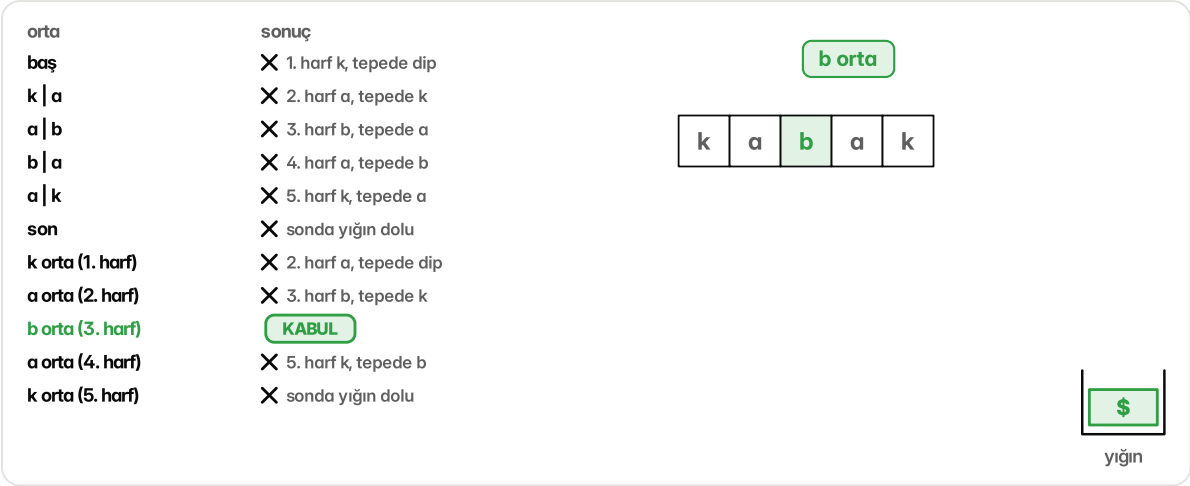


Şekil 8.4 kabak şeridi okunurken yığın $, k$, ak$ olarak büyüyor, b atlanıyor, ikinci a ve k ile eşleşip $ kalıyor; palindrom.

Ama makine ortanın nerede olduğunu bilmez; dizginin uzunluğunu ancak sonunda öğrenir. Koymayı bırakıp almaya ne zaman başlayacak? Çözüm, tahmin etmektir: makine her harfte ikiye ayrılır. Bir kopyası koymaya devam eder, öteki "orta burası" der ve almaya geçer. Uzunluk çiftse

orta iki harfin arasıdır: makine hiç harf okumadan (ε geçişiyle) almaya geçer. Uzunluk tekse orta bir harftir: makine onu okur ama yığına koymaz. Almaya geçen kopyada her harf tepedekiyle aynı olmalıdır; değilse kopya silinir.

Kabak 5 harftir; ortası 5 harften biri ya da 6 aradan biri olabilir: 11 tahmin, hepsi aynı anda. k'den sonra almaya geçen kopya, a gelince tepede k bulur: silinir. Ortayı a sanan kopya b'de takılır. Yalnız b'yi orta sayan kopya a'yı a ile, k'yi k ile eşler, dizgi biterken yığını boşaltır: kabul. Genel olarak n harfli dizgide 2n + 1 tahmin vardır.



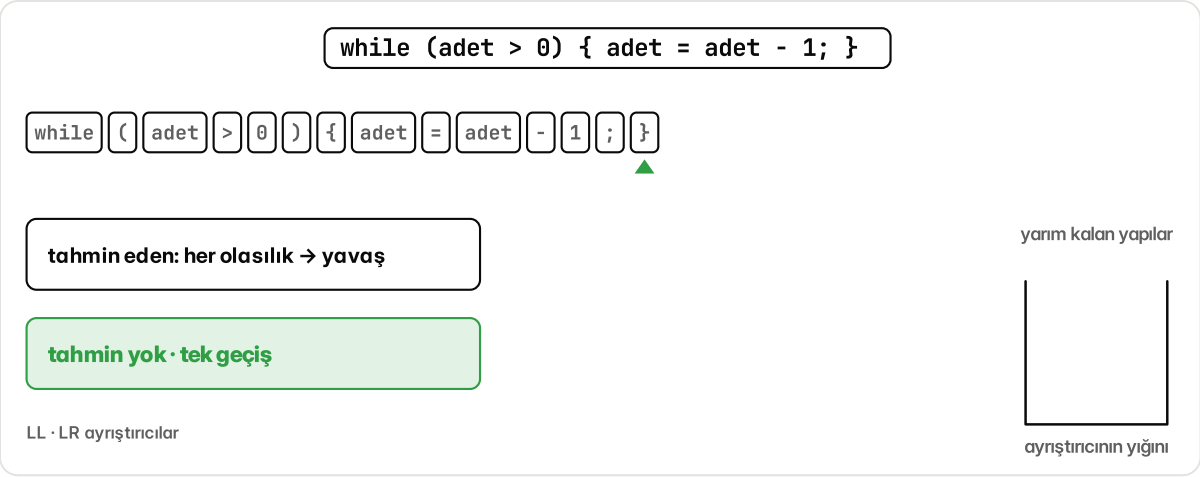
Şekil 8.5 kabak için 11 tahminin tablosu: ortası b olan tek kopya kabul ediyor, yığını $, k$, ak$, ak$, k$, $ diye değişiyor.

"Ey Edip Adana'da pide ye", boşluklar atılınca 19 harftir. Orta için 39 tahmin var; yalnız tam ortadaki n harfini orta sayan kopya kabul eder. O kopyanın yığını 9 harfe kadar yükselir, sonra harf harf boşalır. Böyle, aynı durumda aynı harf ve tepeyle birden çok seçeneği olabilen makineye **deterministik olmayan yığıtlı otomat** denir; kopyalardan biri kabul ederse dizgi kabul edilir.

Tahminsiz, yani deterministik bir yığıtlı otomat bu işi yapamaz. k, a, b okundu diyelim: dizgi kabak'ta bitebilir, kabakkabak diye de sürebilir; ikisi de palindromdur. b'den sonra yığını boşaltırsa uzun dizginin ilk yarısını kaybeder, boşaltmazsa kısayı kaçırır. Bunun ispatı da vardır: hiçbir deterministik yığıtlı otomat bütün palindromları tanıyamaz.

Ortası işaretli olsaydı tahmine gerek kalmazdı. Kabak'ın ortasındaki b yerine bir işaret koyalım (ka#ak): makine işarete kadar koyar, sonra alır; hiçbir adımda seçim yoktur. Demek ki deterministik yığıtlı otomat daha zayıftır: tanıdığı diller, tahmin edebilenin tanıdıklarının bir kısmıdır.

Derleyicinin ayrıştırıcısı da kodu okurken yarım kalan yapıları bir yığında tutar. Tahmin eden ayrıştırıcı her olasılığı denemek zorunda kalır, yavaşlar. Bu yüzden çoğu programlama dilinin grameri, ayrıştırıcı hiç tahmin etmeden, sıradaki parçaya bakarak karar verebilsin diye seçilir; böyle bir ayrıştırıcı kodu tek geçişte okur. Örneğin while (adet > 0) { adet = adet - 1; } satırında while, (ve { açıldıkça yığına girer, kapandıkça boşalır.



Şekil 8.6 while (adet > 0) { adet = adet - 1; } kodunun jetonları; ayrıştırıcının yığını while, (, { ile dolup kapanışlarla boşalıyor.

Üç sayı

videoda 59:39

Dizgiler önce a'lardan, sonra b'lerden, sonra c'lerden oluşsun ve üçünün sayısı eşit olsun: aaabbbccc gibi. Bir yığın a'ları b'lerle karşılaştırmayı bilir. Üç sayıyı birden eşitlemeye de yeter mi?

Önce yığınla gramer arasındaki bağa bakalım. Önce birkaç a, sonra 2 katı b: bu dilin grameri 2 kuraldır:

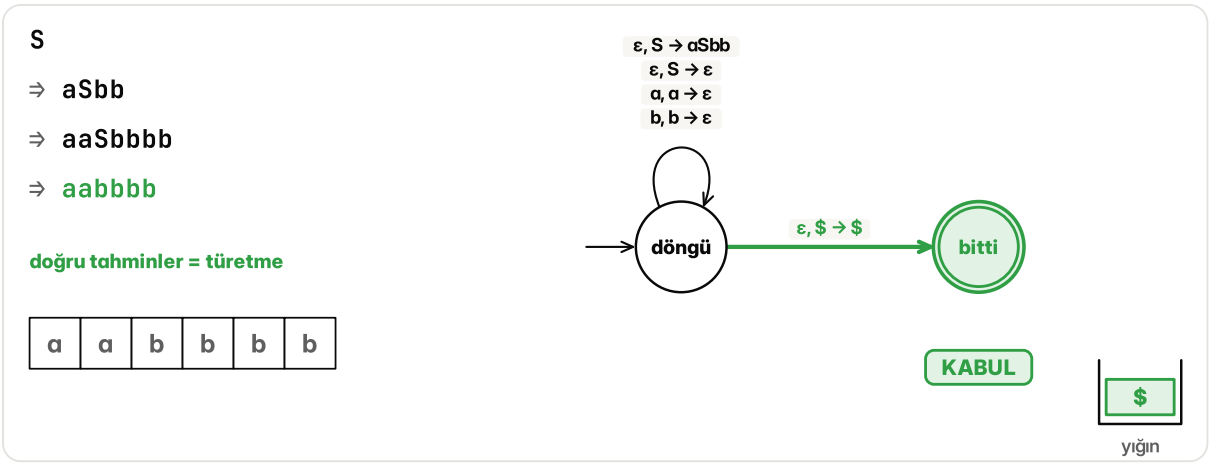
```
S → aSbb | ε
```

Bu gramerden makine kurmak kolaydır: yığına önce S yazılır. Tepede S varsa makine, ε geçişiyle, S'nin yerine bir kuralın sağ tarafını yazar. Tepede harf varsa okunan harfle karşılaştırır; aynıysa alır. Makinenin okları şunlardır:

```
ε, S → aSbb
ε, S → ε
a, a → ε
b, b → ε
ε, $ → $
```

aabbbb'yi deneyelim. Tepede S: yerine aSbb yazılır; tepedeki a, okunan a ile eşleşir ve alınır. Yine S, yine aSbb; ikinci a da eşleşir. Sonra S'nin yerine boş dizgi yazılır. Kalan 4 b, gelen 4 b ile eşleşir: yığın boş, dizgi bitti, kabul. Yığın sırayla S$, aSbb$, Sbb$, aSbbbb$, Sbbbb$, bbbb$, bbb$, bb$, b$ ve $ olur; ε geçişiyle bitti durumuna geçilince toplam 10 hamle yapılmıştır.

Her adımdaki kural seçimi bir tahmindir; doğru tahminlerin sırası, dizginin türetmesinin ta kendisidir: S ⇒ aSbb ⇒ aaSbbbb ⇒ aabbbb. Her bağlamdan bağımsız gramerden böyle bir yığıtlı otomat kurulur; tersine, her yığıtlı otomatın dilini yazan bir gramer de vardır.

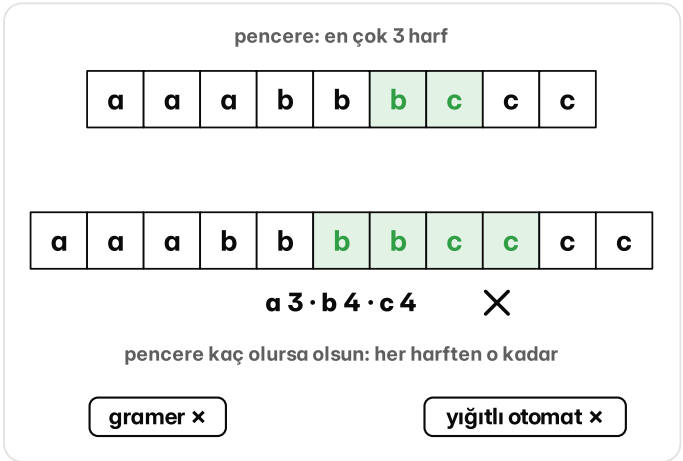


Şekil 8.7 aabbbb okunurken yığın S$'dan $'a iniyor ve seçilen 3 kural (aSbb, aSbb, ε) türetme satırıyla aynı renkte eşleşiyor.

Şimdi üç sayıya dönelim. a'ları yığına koyup her b'de bir a alırsak, b'ler bittiğinde yığın boşalır; c'leri sayacak bir şey kalmaz. Bu makine aabbc ile aabbcc'yi ayıramaz: ikisini de kabul eder. b'leri de yığına koysak, bu kez a'larla karşılaştıramayız.

Bunun bir de ispatı var. Uzun bir dizginin türetme ağacı yüksektir; kökten yaprağa inen bir yolda aynı değişken iki kez görünür ve aradaki kural bir kez daha uygulanabilir. aSbb'de S'nin içinde yine S var: bir tur daha, soldaki a ile sağdaki 2 b birlikte çoğalır ve dizgi dilde kalır (abb, aabbbb, aaabbbbbb, ...). Bağlamdan bağımsız diller için **pompalama önsavı** bunu söyler: yeterince uzun her dizgide birbirine yakın, birlikte çoğalabilen iki parça vardır.

Diyelim 2 parça en çok 3 harflik bir pencereye sığıyor. aaabbbccc'de böyle bir pencere ya a'lara ve b'lere, ya b'lere ve c'lere değer; üç harf türüne birden uzanamaz. Parçalar bir b ve bir c olsun: bir tur sonra 3 a, 4 b, 4 c olur (aaabbbbcccc); eşitlik bozulur. Pencere kaç harf olursa olsun her harften o kadar alırız, bir tür hep geride kalır. Bu dil hiçbir gramerle yazılamaz, hiçbir yığıtlı otomatla tanınamaz.



Şekil 8.8 aaabbbccc üzerinde 3 karelik pencere; v = b ve x = c seçilince şerit aaabbbbcccc oluyor, sayaç 3, 4, 4.

Buradan bir sonuç daha çıkar. a'ları b'lere eşit dizgiler ($a^i b^i c^k$) tek yığınla tanınır, b'leri c'lere eşit olanlar ($a^i b^j c^j$) da; ikisine birden uyanlar ise tam bu üçlü eşitliktir. Örneğin aabbc yalnız birincide, aabbcc ikisinde kabul edilir. Yani iki dilin kesişimi tek yığına sığmaz: iki makineyi yan yana çalıştırmak iki yığın ister. Tümleyende de öyle: üçlü eşitliğe uymayan dizgiler tek yığınla tanınır,

tümleyenleri olan üçlü eşitlik tanınamaz. Bağlamdan bağımsız diller kesişimde ve tümleyende kapalı değildir. Yığının gücü burada biter: bir sayıyı ancak bir kez karşılaştırabilir, karşılaştırdığını da siler.

Özet

- Yığıtlı otomat, sonlu otomata sınırsız bir yığın ekler; makine yalnız yığının tepesini görür ve her ok "okunan, tepe → yazılan" diye yazılır.
- 0^n1^n için makine 0'ları yığına koyar, 1'lerde alır; dip işareti sayesinde yığının boşaldığı görülür. 0000011111 11 hamlede kabul edilir.
- Palindromu tanımak için makine ortayı tahmin eder; deterministik olmayan makine, kopyalardan biri kabul ederse kabul eder (n harfte 2n + 1 tahmin).
- Deterministik yığıtlı otomat daha zayıftır; ortası işaretli dizgiler ona yeter, bütün palindromlar yetmez.
- Bağlamdan bağımsız gramerler ile yığıtlı otomatlar aynı dilleri anlatır: gramerden makine kurulurken doğru tahminler türetmenin kendisidir.
- $a^nb^nc^n$ hiçbir yığıtlı otomatla tanınamaz; bağlamdan bağımsız diller kesişimde ve tümleyende kapalı değildir.

Sorular

1 **Aşağıdaki dizgilerden hangisi palindrom (tersten de aynı okunan dizgi) değildir?**

A kayak
B anana
C radar
D kalem

2 **0^n1^n dilini tanıyan koy, al, bitti makinesi 0011 dizgisini okurken yığın dip işareti dahil en çok kaç kata çıkar?**

A 2
B 3
C 4
D 5

3 **0^n1^n için koy, al, bitti makinesi (koy durumunda 0'lar yığına konur, al durumunda her 1 bir 0 alır) aşağıdaki dizgilerden hangisini reddeder?**

A 01
B 0011
C 00011
D 000111

4 **Koy, al, bitti makinesi 0101 dizgisini neden reddeder?**

- **A** Yığında bir 0 kalır, dip işareti görünmez
- **B** Yığında alınacak 0 kalmamıştır
- **C** Dizgi 0'la başladığı için makine koy durumundan çıkamaz
- **D** al durumundayken bir 0 geliyor ve o durumda 0 okuyan ok yok

5 **Palindrom makinesi her harfte koymaya devam etme ve orta burası deme seçenekleriyle tahmin yapıyor. 9 harflik bir dizgi için aynı anda kaç tahmin vardır (5 harflik kabak'ta 11, 19 harflik cümlede 39 tahmin vardı)?**

Cevap:

6 **Deterministik (tahminsiz) bir yığıtlı otomat bütün palindromları neden tanıyamaz?**

- **A** Deterministik makine yığına harf koyamaz
- **B** Palindromlar bağlamdan bağımsız gramerle yazılamaz
- **C** kab okunduktan sonra dizgi hem kabak hem kabakkabak diye bitebilir ve ortanın nerede olduğu önceden bilinemez
- **D** Deterministik makinenin dip işareti olmaz

7 **S → aSbb | ε gramerinden kurulan makine (yığına önce S yazılır; ε, S → aSbb ve ε, S → ε kural seçer; a, a → ε ve b, b → ε harf eşler; ε, $ → $ bitti durumuna geçirir) aabbbb'yi 10 hamlede kabul eder. Aynı makine aaabbbbbb'yi kaç hamlede kabul eder?**

Cevap:

8 **aaabbbccc dizgisinde bağlamdan bağımsız diller için pompalama önsavının iki parçası v = 3. a ve x = 1. b olarak seçildi. İki parçayı bir tur daha yazınca çıkan dizgiyi yaz.**

Cevap:

BİÇİMSEL DİLLER VE OTOMATA

9 Turing makineleri

Bu bölüm, hesaplamanın en yalın modelini anlatır: şerit, kafa ve kural tablosundan oluşan Turing makinesi. Önce onunla bir sayıya 1 eklenir, sonra $a^n b^n c^n$ dili tanınır; en sonda tek bir makinenin bütün makinelerin yerini nasıl tuttuğu ve Church-Turing tezi gelir.

Toplama devresi yok

videoda 1:02:29

Şeritte 10111 yazıyor: ikilik tabanda 23. Önümüzdeki makinenin toplama devresi yok. Yalnız bir kareyi okuyup yerine bir simge yazabiliyor ve bir kare sağa ya da sola geçebiliyor. Bu makine sayıya 1 eklemeyi nasıl yapar?

Kâğıtta en sağdaki basamaktan başlarız. 1 + 1 = 2 eder: 0 yazarız, elde 1. Elde soldaki 1'e eklenir, yine 0, yine elde. İlk 0'a gelince elde biter ve o 0, 1 olur. Yani 1 eklemek bir yazma işidir: sondaki 1'ler 0'a, solundaki ilk 0, 1'e döner.

Makinenin **şeridi** iki yana sonsuzdur, kare karedir ve her karede bir simge vardır ya da kare boştur (boş kare ⊔ ile yazılır). **Kafa** her adımda bir kareyi okur, yerine bir simge yazar ve bir kare sağa ya da sola geçer. Makinenin bir de **durumu** vardır: o an hangi işi yaptığı. Ne yazacağını, yönü ve sonraki durumu bir **kural tablosu** söyler.



kural tablosu

durum	oku	yaz	yön	yeni

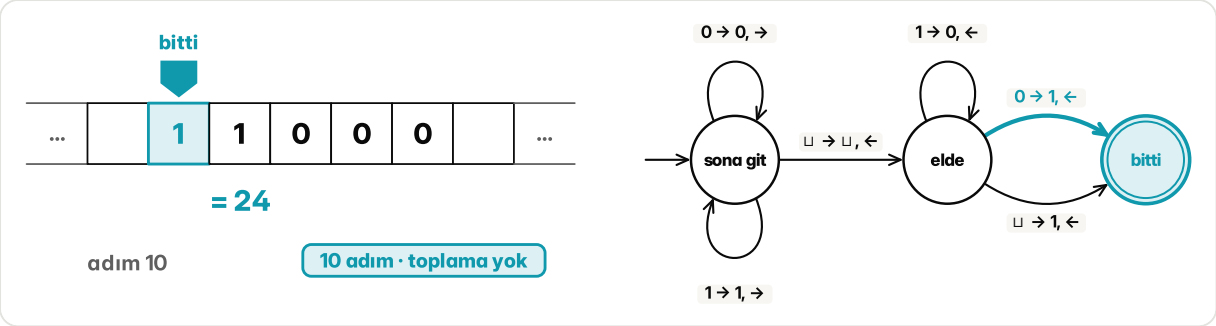
ne yazacak · hangi yöne · sonraki durum

Şekil 9.1 Sonsuz şerit, kafa ve durum; altta okunan simgeye göre yazılacak simgeyi, yönü ve yeni durumu veren kural tablosu.

Bir ekleme için iki çalışma durumu yeter. "Sona git" durumu 0 da okusa 1 de okusa dokunmadan sağa geçer; boş kareye gelince sola döner ve "elde" durumuna geçer. "Elde" durumu 1 okursa 0 yazar, sola geçer ve elde durumunda kalır; 0 ya da boş kare okursa oraya 1 yazar ve "bitti" durumuna geçer. Bitti, makinenin durduğu kabul durumudur. Altı kural şöyle yazılır:

```
sona git   0   →  0  sağ   sona git
sona git   1   →  1  sağ   sona git
sona git   ⊔   →  ⊔  sol   elde
elde       1   →  0  sol   elde
elde       0   →  1  sol   bitti
elde       ⊔   →  1  sol   bitti
```

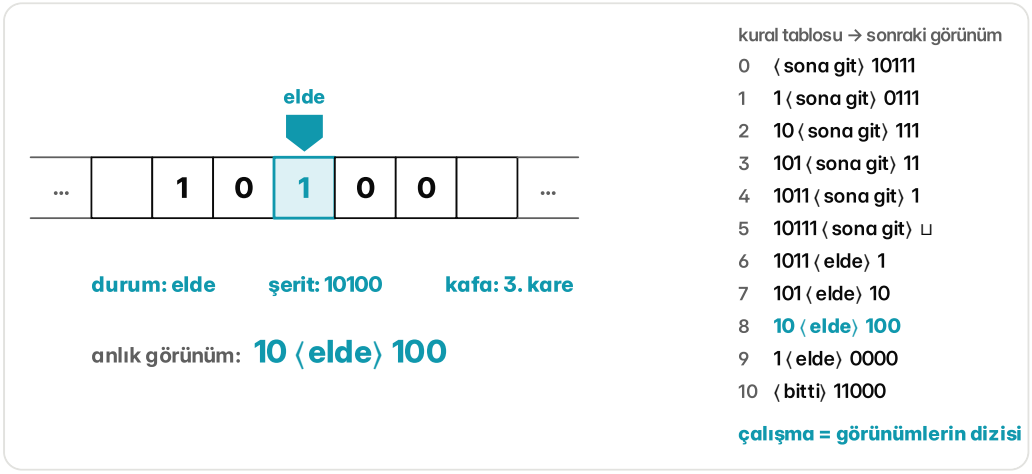
10111 ile başlayalım; kafa en soldaki karede. Makine 5 adımda 5 basamağı geçer, 6. adımda boş kareyi görüp döner. Elde 3 kez 1 okur, üçünü de 0 yapar; 10. adımda 0'a varır, onu 1 yapar ve durur. Şeritte 11000 kalır: 24. 10 adım atıldı, hiçbir şey toplanmadı.



Şekil 9.2 Makine 10. adımda bitti durumunda; şeritte 11000 (24) yazıyor, yanında altı kural ve adım sayısı görünüyor.

Sayı yalnız 1'lerden oluşsaydı, örneğin 11 (3)? Elde iki 1'i de 0 yapar, sayının solundaki boş kareye varır ve oraya 1 yazar: 100, yani 4. Şerit iki yana sonsuz olduğu için sayı uzayınca yer hep vardır.

Makinenin bir andaki bütün hâli 3 şeydir: durum, şeridin içeriği ve kafanın yeri. Buna **anlık görünüm** denir. 8. adımdan sonra durum elde, şeritte 10100, kafa 3. karededir; bu, durumun şeridin üstüne yazıldığı 10 ⟨elde⟩ 100 biçiminde gösterilir (durumun sağındaki ilk simge kafanın altındaki karedir). Kural tablosu her görünümden sonrakini kesin olarak verir; bütün çalışma bu görünümlerin dizisidir.



Şekil 9.3 Sol tarafta 8. adımın anlık görünümü 10 ⟨elde⟩ 100; sağda 0. adımdan 10. adıma kadar bütün görünümler.

Bu makineye, onu 1936'da tarif eden Alan Turing'in adıyla **Turing makinesi (TM)** denir. Bugünkü biçimsel tanımı 7 parçadır: durumlar, girdi alfabesi, boş simgeyi de içeren şerit alfabesi, geçiş fonksiyonu, başlangıç, kabul ve ret durumları. Geçiş fonksiyonu o anki durumdan ve okunan simgeden yeni durumu, yazılacak simgeyi ve yönü verir.

Sonlu otomattan farkı da buradadır. Sonlu otomat girdisini soldan sağa bir kez okur, hiçbir şey yazmaz; bütün belleği durumudur. Turing makinesi yazar, geri döner, şeridi bitmez. Kabul ya da ret durumuna girince durur; ikisine de girmezse sonsuza kadar çalışabilir. Toplama devresi aynı sonucu tek adımda verir; bu makine 10 adımda, yalnız okuyup yazarak buldu.

Üç eşit sayı

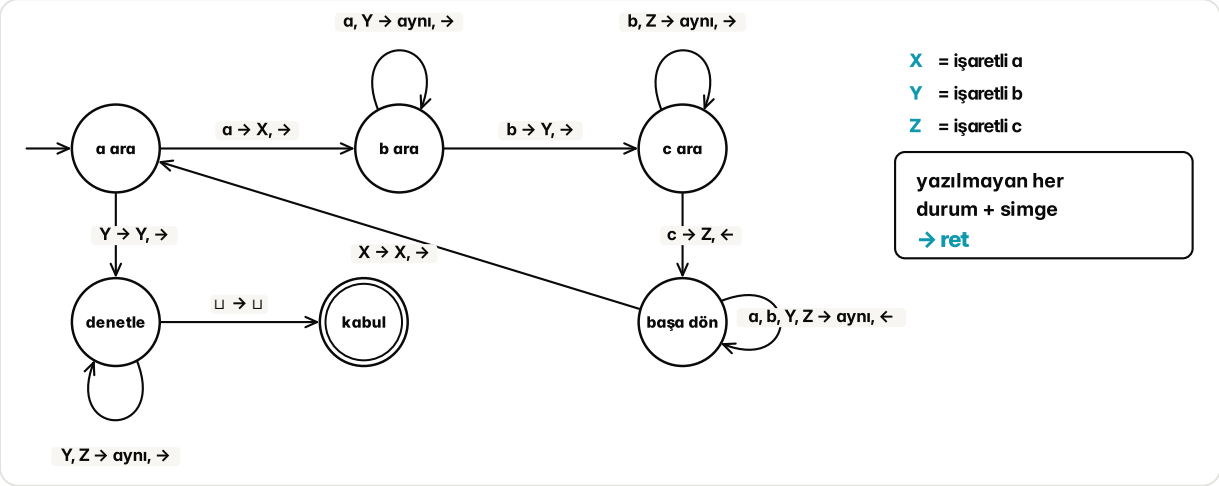
videoda 1:05:15

Önce a'lar, sonra b'ler, sonra c'ler gelsin, üçü de aynı sayıda: 4 a, 4 b, 4 c gibi. Bu dile $a^n b^n c^n$ denir. Bu dizgileri yığıtlı otomat bile tanıyamaz: a'ları yığına koysa, b'lerle eşleştirirken yığın boşalır; c'ler geldiğinde elinde sayı kalmamıştır. Turing makinesi bu kuralı nasıl denetler?

Sayıyı aklında tutması gerekmez; şeride yazabilir. Fikir şu: bir a'yı işaretle, sağa git, ilk b'yi işaretle, sonra ilk c'yi; sonra başa dön. Her tur her harften 1 tane eksiltir. Sayılar eşitse üçü aynı turda biter. Biri fazla ya da eksikse bir turda eşi bulunmayan bir harf çıkar. İşaretli a, b ve c şeritte sırasıyla X, Y ve Z ile gösterilir.

Bu işe 5 durum yeter. "a ara" ilk işaretsiz a'yı işaretler. "b ara" sağa gider, a'ları ve işaretli b'leri geçip ilk b'yi işaretler. "c ara" aynı yolla ilk c'yi işaretler. "Başa dön", işaretli a'ya varana kadar sola gider, sonra yine a ara. a ara işaretli bir b görürse a kalmamıştır: "denetle" durumu sağda yalnız işaretli harfler mi var, ona bakar. Beklenmedik bir harf gören her durum ret durumuna geçer. Toplam 17 kural vardır; yazılmayan her durum ve simge çifti ret demektir:

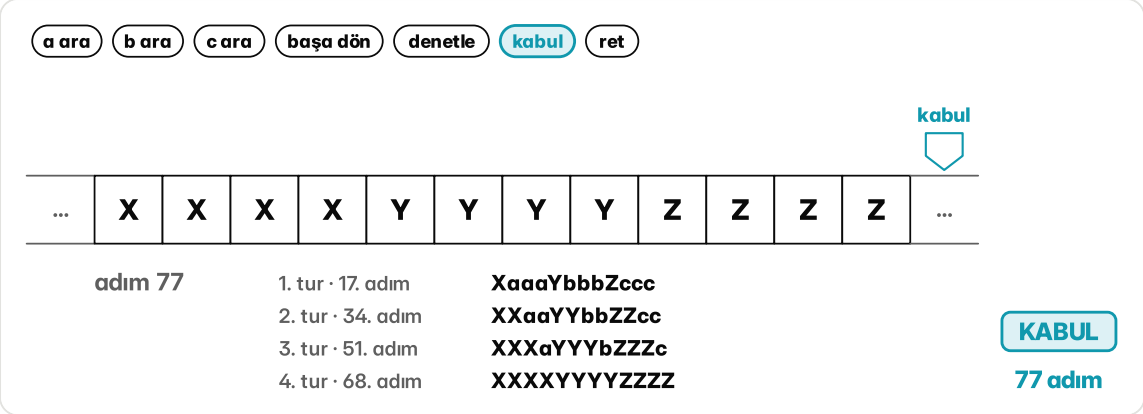
```
a ara       a → X sağ b ara     Y → Y sağ denetle   ␣ → ␣ sağ kabul
b ara       a → a sağ b ara     Y → Y sağ b ara     b → Y sağ c ara
c ara       b → b sağ c ara     Z → Z sağ c ara     c → Z sol başa dön
başa dön    a, b, Y, Z → aynı simge, sol            X → X sağ a ara
denetle     Y, Z → aynı simge, sağ                  ␣ → ␣ sağ kabul
```



Şekil 9.4 Beş çalışma durumunun diyagramı: a ara, b ara, c ara, başa dön, denetle; oklarda okunan, yazılan simge ve yön, yanda X, Y, Z işaretlerinin anlamı.

4 a, 4 b, 4 c ile deneyelim. İlk turda 1. a, 1. b ve 1. c işaretlenir; kafa başa döner. Her tur birer harf daha işaretler; 4. turun sonunda işaretsiz harf kalmaz. 5. turda a ara işaretli bir b görür: a kalmamış. Denetle durumu sağa doğru yalnız işaretli harfler görür, sonra boş kare: kabul. Bütün iş 77 adım sürdü. Turların sonunda şerit şöyledir:

```
17. adım   XaaaYbbbZccc
34. adım   XXaaYYbbZZcc
51. adım   XXXaYYYbZZZc
68. adım   XXXXYYYYZZZZ
```



Şekil 9.5 aaaabbbbcccc girdisinde dört turun sonundaki şerit durumları; 4. turda hepsi işaretli, makine 77. adımda kabul ediyor.

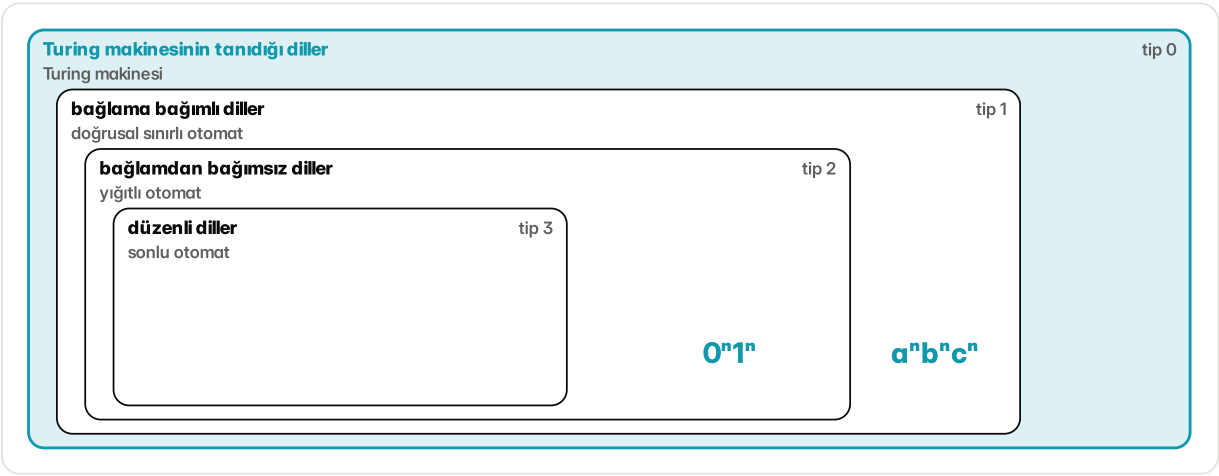
Şimdi bir b eksik olsun: 4 a, 3 b, 4 c. İlk 3 tur yine tamamlanır (tur sonları 15, 30 ve 45. adım). 4. turda son a işaretlenir; b ara 3 işaretli b'yi geçer, sonra işaretli bir c'ye çarpar: eşleşecek b kalmamış. Kuralı olmayan bu çiftte makine ret durumuna geçer ve 50. adımda durur.

Bu makine girdinin kapladığı karelerin dışına hiçbir şey yazmadı; yalnız girdinin üstünde gidip geldi. Şeridi girdinin uzunluğuyla sınırlı Turing makinesine **doğrusal sınırlı otomat** denir. Böyle makinelerin tanıyabildiği dillere de bağlama bağımlı diller denir; $a^n b^n c^n$ onlardan biridir.

Böylece 4 basamaklı bir merdiven çıkar:

- düzenli diller: sonlu otomat tanır;
- bağlamdan bağımsız diller: yığıtlı otomat tanır (0^n1^n bu basamaktadır);
- bağlama bağımlı diller: doğrusal sınırlı otomat tanır ($a^n b^n c^n$ bu basamaktadır);
- en üstte Turing makinesinin tanıyabildiği diller.

Her basamak alttakini içerir ve ondan geniştir. Bu merdivene, gramerleri böyle ilk sınıflandıran dil bilimcinin adıyla **Chomsky hiyerarşisi** denir.



Şekil 9.6 Chomsky hiyerarşisi: tip 3 düzenli diller, tip 2 bağlamdan bağımsız, tip 1 bağlama bağımlı, tip 0 Turing makinesinin tanıdığı diller; 0^n1^n ve $a^n b^n c^n$ yerlerine konmuş.

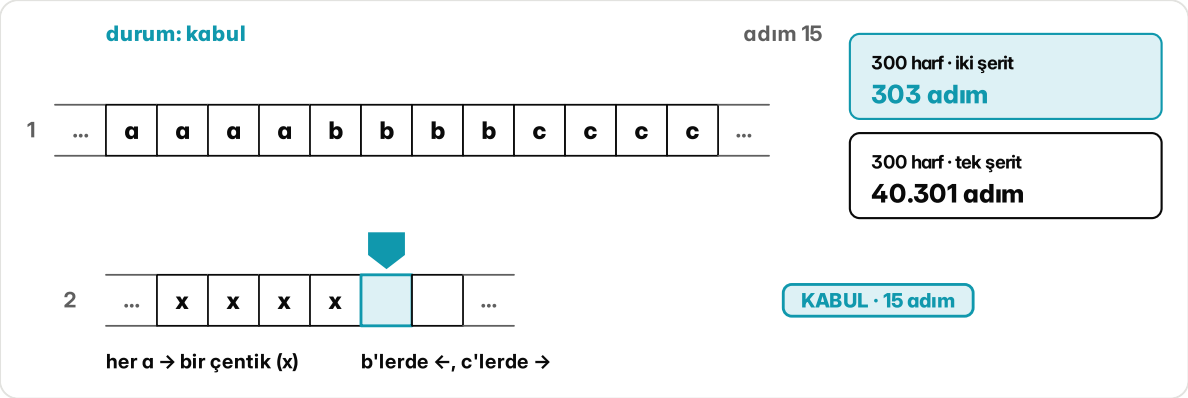
Turing makinesini yığıtlı otomattan güçlü kılan, okuduğunun üstüne yazabilmesi ve şeritte istediği yere geri dönebilmesidir. Yığından alınan simge kaybolur ve yığın yalnız tepesini gösterir; şeride yazılan her şey ise yerinde durur ve istenince yeniden okunur.

Bir telefon, her iş

videoda 1:07:57

Aynı telefon bir an hesap makinesi, bir an harita, bir an oyun olur; içinde her iş için ayrı bir makine yok. Tek bir makine bütün makinelerin işini nasıl yapar?

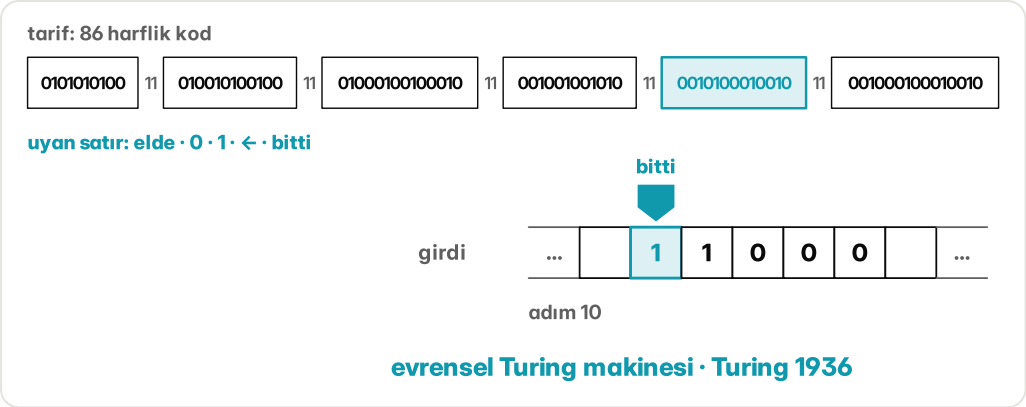
Makineye ikinci bir şerit verelim. Önce a'lar, sonra b'ler, sonra c'ler olan bir dizgide üçü eşit sayıda mı diye bakılsın. İki şeritli makine her a için ikinci şeride bir çentik yazar. b'lerde çentiklerin üstünde geri, c'lerde ileri yürür. Çentikler iki kez de tam yetiyorsa sayılar eşittir. 12 harflik aaaabbbbcccc girdisinde iki şeritli makine 15 adımda biter; tek şeritli makine 77 adım atmıştı. 300 harfte fark büyür: iki şeritli makine 303 adımda, harfleri tek şeritte işaretleyip gidip gelen makine 40.000'den fazla (40.301) adımda biter.



Şekil 9.7 Üstte girdi şeridi, altta çentik şeridi; sağda 300 harf için iki şeritte 303, tek şeritte 40.301 adım.

Yine de ikinci şerit yeni bir güç katmaz: tek şeritli makine iki şeridi yan yana yazar, kafaların yerini işaretleyip onu taklit eder; yavaşlar, ama aynı soruları cevaplar. Her adımda birkaç seçeneği olan deterministik olmayan makine de seçenekleri tek tek deneyen bir makineyle taklit edilir. Hız değişir, güç değişmez.

Taklidi sonuna götürelim. Kural tablosu sonlu bir listedir: ikilik sayıya 1 ekleyen makinenin 6 kuralı, numaralanıp 0'lar ve 1'lerle yazılınca 86 harflik tek bir dizgi olur. Turing 1936'daki makalesinde şunu da gösterdi: şeridinde bir makinenin tarifini ve girdisini bulup o makineyi adım adım taklit eden tek bir makine kurulabilir. Her adımda tarifte o anki duruma ve simgeye uyan satırı bulur, satırın dediğini girdinin üstünde yapar. Buna **evrensel Turing makinesi** denir.

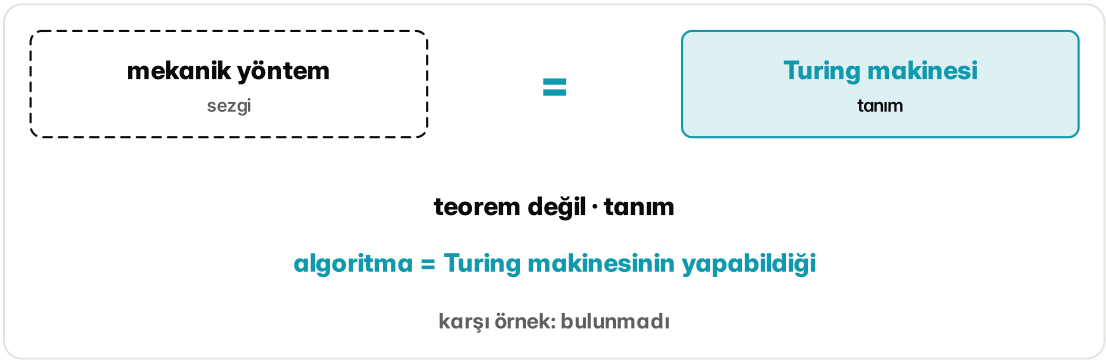


Şekil 9.8 Üstte 86 harflik tarifin satırları, altta girdi 10111; makine tarifteki uyan satırı bulup 10. adımda bitti durumuna geliyor.

Program da şeritte duran bir veridir: yeni iş için yeni makine değil, yeni tarif gerekir. Bugünkü bilgisayarlarda da program, verilerle aynı bellekte duran bir sayı dizisidir; telefon haritaya geçmek için yeni bir devre değil, başka bir program yükler.

Aynı yıllarda hesaplamanın başka tanımları da yapıldı: Alonzo Church'ün λ-hesabı her şeyi fonksiyon tanımlayıp uygulamakla yazıyordu; özyinelemeli fonksiyonlar sıfır, bir ekleme ve kendini çağırma gibi birkaç işlemden kuruluyordu. 1936 ve 1937'de üçünün de aynı fonksiyonları hesapladığı ispatlandı.

Church-Turing tezi buradan çıkar: adım adım, mekanik bir yöntemle hesaplanabilen her şeyi bir Turing makinesi hesaplar. Bu bir teorem değil: mekanik yöntem sezgiyle bilinen bir kavram, ispatlanacak matematiksel bir tanımı yok. Tez, algoritma sözüne tanım verir; bugüne kadar Turing makinesinin hesaplayamadığını hesaplayan bir yöntem bulunmadı.



Şekil 9.9 Church-Turing tezi: mekanik yöntem bir sezgi, Turing makinesi bir tanım; "teorem değil, tanım" ve "karşı örnek: bulunmadı".

Turing makinesinin hesaplayabildiği her şeyi hesaplayabilen sistemlere **Turing-tam** denir: belleği yettikçe C, Java ve Python da öyle. Conway'in Hayat Oyunu bile: ızgaradaki her hücre komşularının sayısına göre doğar, yaşar ya da ölür; bu birkaç kuralla ızgarada çalışan bir bilgisayar kurulabilir. Telefon da, belleği yettikçe, evrensel bir makinedir.

Özet

- Turing makinesi sonsuz şeridi okur, yazar ve iki yöne gider; ne yapacağını kural tablosu söyler.
- Makinenin bir andaki hâli durum, şerit ve kafa yeridir; çalışma anlık görünümlerin dizisidir.
- Şeride yazabildiği için $a^n b^n c^n$ gibi, yığıtlı otomatın tanıyamadığı dilleri işaretleyerek tanır.
- Düzenli, bağlamdan bağımsız, bağlama bağımlı ve Turing-tanınabilir diller Chomsky hiyerarşisinin 4 basamağıdır.
- Ek şerit hızı değiştirir, gücü değiştirmez; evrensel makine bir tarifi okuyup herhangi bir makineyi taklit eder.
- Church-Turing tezi bir teorem değil, algoritmaya verilen tanımdır.

Sorular

1 **Bir Turing makinesinin kafası her adımda ne yapar?**

A Şeridin tamamını okuyup sonucu tek adımda yazar.

B Yalnız soldan sağa gider ve hiçbir şey yazmaz.

C Bir kareyi okur, yerine bir simge yazar ve bir kare sağa ya da sola geçer.

D Yığının tepesine bir simge koyar ya da tepedeki simgeyi alır.

2 **Sayıya 1 ekleyen makine 1011 ile başlıyor. Durduğunda şeritte ne yazar?**

Cevap: _____

3 **Sayıya 1 ekleyen makine 1011 ile başlıyor ve 6 adım atıyor (durum, şerit ve kafa yeri gösteriminde durumun sağındaki ilk simge kafanın altındaki karedir). Anlık görünüm hangisidir?**

A 10 ⟨elde⟩ 10

B 101 ⟨elde⟩ 0

C 1 ⟨elde⟩ 000

D 1011 ⟨sona git⟩ ␣

4 **Aynı makine 111 ile başlıyor. Kaç adımda durur?**

Cevap: _____

5 **Turing makinesini yığıtlı otomattan güçlü kılan nedir?**

A Girdiyi iki kez okuyabilmesi.

B Yazdığı simgelere şeritte istediği zaman geri dönüp yeniden okuyabilmesi.

C Durum sayısının sonsuz olması.

D Yalnız şeridin bir ucunda çalışması.

6 **$a^n b^n c^n$ makinesine abc dizgisi veriliyor. Makine kaç adımda kabul eder?**

Cevap: _____

7 **$a^n b^n c^n$ dilini tanıyabilen makine sınıfları arasında, bu bölümde gösterilen en sınırlı olan hangisidir?**

A Sonlu otomat

B Yığıtlı otomat

C Çok şeritli Turing makinesi

D Doğrusal sınırlı otomat

8 **Church-Turing tezi neden bir teorem değil, bir tez olarak anılır?**

A Henüz yeterince büyük bir bilgisayar olmadığı için sınanamadı.

B Turing makinesinin hesaplayamadığını hesaplayan bir yöntem bulundu.

C Mekanik yöntem sezgiyle bilinen bir kavram; ispatlanacak matematiksel bir tanımı yok.

D Church'ün ve Turing'in tanımları farklı fonksiyonları hesaplıyor.

BİÇİMSEL DİLLER VE OTOMATA

10 Karar verilebilirlik

Bu bölüm, bazı soruların hiçbir bilgisayar programıyla her girdide cevaplanamadığını gösterir. Önce kart bulmacasıyla karar verilebilir ve tanınabilir kavramları, sonra durma probleminin köşegen ispatı, en sonda da bu sonucun virüs tarayıcılarına ve Rice teoremine yansıması anlatılır.

Üç kart

videoda 1:10:29

Ekranda 3 kart var; her birinin üstünde ve altında birer dizgi (harflerden oluşan dizi) yazıyor:

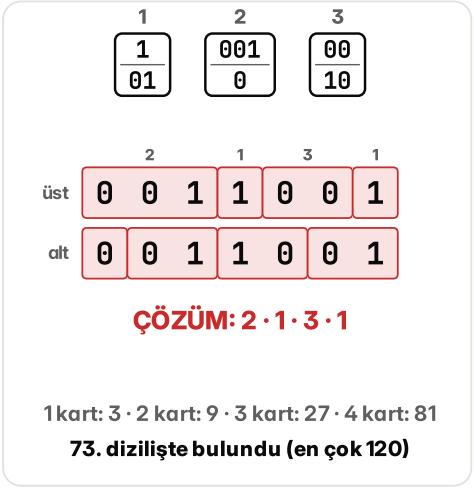
```
kart 1:  1  / 01
kart 2:  001 / 0
kart 3:  00 / 10
```

Kartları istediğimiz sırayla, istediğimiz kadar yan yana dizebiliriz. Amaç, üst satırdaki yazının alt satırdakiyle aynı olması. Bu 3 kartla olur mu? Peki her takımda bu soruyu cevaplayan bir yöntem var mı?

Başlangıç kartı bellidir: üstü ve altı aynı harfle başlayan tek kart 2. kartır. Üstte 001, altta 0 kalır; alta 01 eksik. Sırada 1. kart var: altı tam 01. Sonra 3. kart, sonra yine 1. kart: iki satır da 0011001 olur. Çözüm 2, 1, 3, 1 sırasıdır:

```
üst:  001 + 1  + 00 + 1   = 0011001
alt:  0   + 01 + 10 + 01  = 0011001
```

Körü körüne arayan bir program da önce tek kartları, sonra ikilileri, üçlüleri dener ve 73. dizilişte bulur (4 karta kadar en çok 3 + 9 + 27 + 81 = 120 diziliş vardır).



Şekil 10.1 Üç kart ve 2, 1, 3, 1 sırasıyla dizilmiş çözüm; iki satır da 0011001; yanda 1, 2, 3, 4 kartlık diziliş sayıları ve 73. dizilişte bulunduğu notu.

Bazı sorular ise her zaman kesin bir cevapla biter. Bu 7 harfi, art arda iki tane 1 var mı diye soran 3 durumlu bir sonlu otomata verelim: her harfi bir kez okur, 7 adımda durur ve kabul der. Bir gramerin bir dizgiyi türetip türetmediğini de sınırlı adımda bulan yöntemler var. Her girdide durup evet ya da hayır diyen bir makinesi olan soruya **karar verilebilir (decidable)** denir.

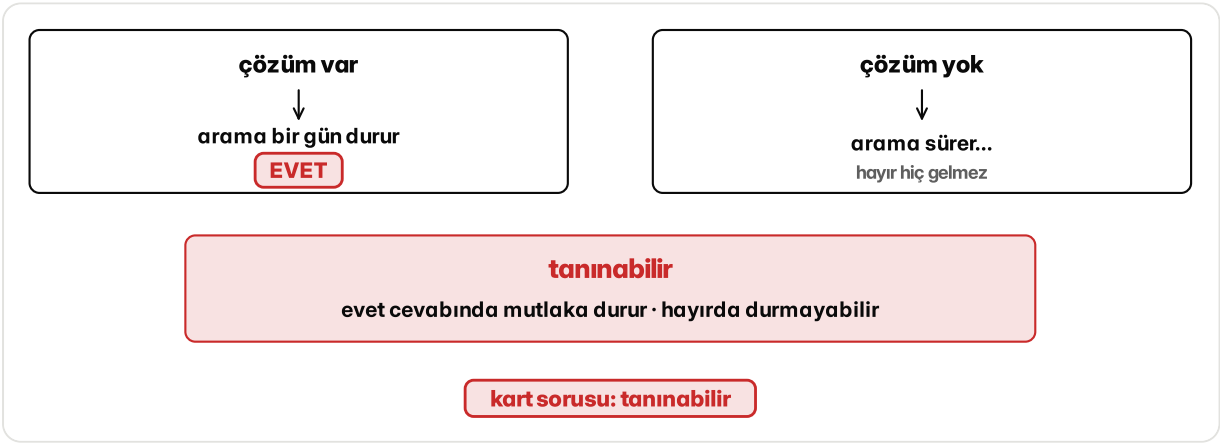
Şimdi başka bir takım: her kartın üst yazısı alttakinden uzun (100 / 1, 01 / 0, 110 / 11). Kaç kart dizersek dizelim üst satır hep daha uzun kalır; çözüm yoktur. Bunu bir bakışta gördük, ama körü körüne arayan program görmez: 3, 9, 27, 81, 243 diye dizilişleri dener ve hiç durmaz.



Şekil 10.2 Üst yazısı hep alttan uzun üç kart; üstte artan fark (+2, +1, +1), yanda her adımda üçle çarpılan diziliş sayısı ve "durmuyor".

Bir sonraki takım ise masum görünüyor: 10 / 1, 110 / 0, 1 / 110. En kısa çözümü tam 55 kart sürer; iki satırda da 99 harf. 54 karta kadar bütün dizilişleri deneyip "çözüm yok" diyen yanılırdı. Peki ne kadar aradıktan sonra "yok" demek güvenli?

Körü körüne arama, çözümü olan her takımda bir gün durur ve evet der; çözümü olmayanda sonsuza kadar sürer. Evet cevabında mutlaka duran, hayırda durmayabilen bir makinesi olan soruya **tanınabilir (recognizable)** denir. Kart sorusu tanınabilirdir.



Şekil 10.3 Çözüm varsa arama bir gün durur ve evet der; yoksa sürer; kart sorusunun tanınabilir olduğu yazılı.

Her takımda duran daha akıllı bir yöntem var mı? Emil Post 1946'da cevabı verdi: kart sayısına sınır konmazsa böyle bir yöntem yok. 2 kartlık takımlarda var, 5 kartlıklarda bile yok. Bulmacanın adı bugün Post yazışma problemi: tanınabilir ama karar verilemez.

Aynı kartlardan bir gramer de kurulur: bir kolu kartların üst yazılarını, öteki alt yazılarını dizer, ikisi de kart numaralarını sona ekler. Takımın çözümü varsa iki kol aynı dizgiyi üretir; o dizginin iki ayrı türetme ağacı olur, yani gramer belirsizdir. Çözüm yoksa kollar hiç buluşmaz. Bu yüzden bir gramerin belirsiz olup olmadığına da karar verilemez.

Yazılımda hata arayan doğrulama araçları da çoğu zaman bir sınırla çalışır: belli sayıda adıma kadar arar. Bulursa kesin konuşur; bulamazsa "yok" demez, "bu sınıra kadar bulamadım" der. 55 kartlık takım, bu ikisinin neden aynı şey olmadığını gösteriyor.

Dört satırlık program

videoda 1:13:07

Ekrandaki program 4'ten başlayıp çift sayıları sırayla dener, her birini iki asalın toplamı olarak yazmaya çalışır: 4 = 2 + 2; 6 = 3 + 3; 8 = 3 + 5. Yazamadığı ilk sayıda durur. Bu program bir gün durur mu? Bunu çalıştırmadan söyleyen bir program yazılabilir mi?

Program durursa, iki asalın toplamı olmayan bir çift sayı bulmuş olur; 1742'den beri sorulan soru (Goldbach sanısı) da bu. 18 basamağa kadar bütün çift sayılar denendi, hepsi yazılabildi; ispat ise yok. Durup durmayacağını önceden söyleyen bir denetçi program olsaydı, soru bir çırpıda cevaplanırdı.

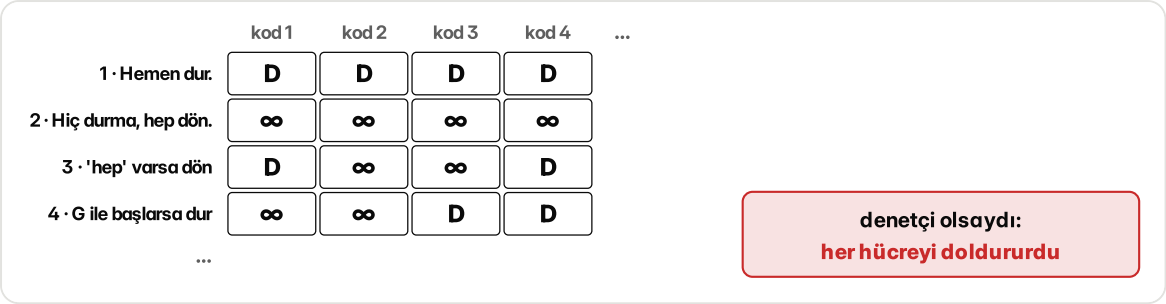
Hilbert 1928'de tam böyle bir yol aradı: bir mantık önermesinin ispatlanıp ispatlanamayacağını sınırlı adımda söyleyen mekanik bir yöntem; adı karar problemi. Gödel 1931'de, aritmetiği içeren her çelişkisiz kural sisteminde ispatlanamayan doğrular bulunduğunu gösterdi; karar sorusu ise açık kaldı.

Cevap için bir şey yeter: her programın kodu bir dizgidir. Bir program başka bir programın kodunu, kendi kodunu da girdi olarak okuyabilir. 4 kısa program alalım:

```
1. Hemen dur.
2. Hiç durma, hep dön (aynı adımı sonsuza kadar tekrarla).
3. Girdide "hep" sözcüğü geçiyorsa dön, yoksa dur.
4. Girdi G harfiyle başlıyorsa dur, yoksa dön.
```

Bir tablo kuralım: satırlarda programlar, sütunlarda kodları; her hücre, o programın o kodda durup durmadığını söyler (D: durur, ∞: döner). 1. satır hep durur, 2. hep döner. 3. program 2. kodda "hep" sözcüğünü bulup döner, 1. kodda durur.

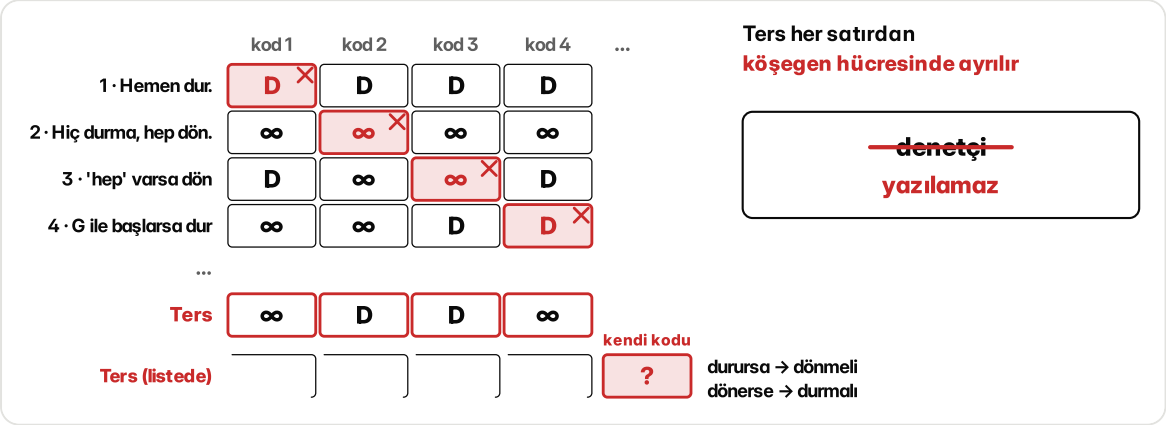
program	kod 1	kod 2	kod 3	kod 4
1	D	D	D	D
2	∞	∞	∞	∞
3	D	∞	∞	D
4	∞	∞	D	D



Şekil 10.4 Satırlarda 4 program, sütunlarda 4 kod; hücrelerde D ve ∞; altta "denetçi olsaydı her hücreyi doldururdu".

Köşegene bakalım: her program kendi kodunda ne yapıyor? Dur, dön, dön, dur. **Ters** adında bir program yazalım: bir kod okur, denetçiye o kodun programının kendi kodunda durup durmadığını sorar ve tersini yapar. Durur derse döner, döner derse durur. Ters'in satırı köşegenin tersidir: dön, dur, dur, dön.

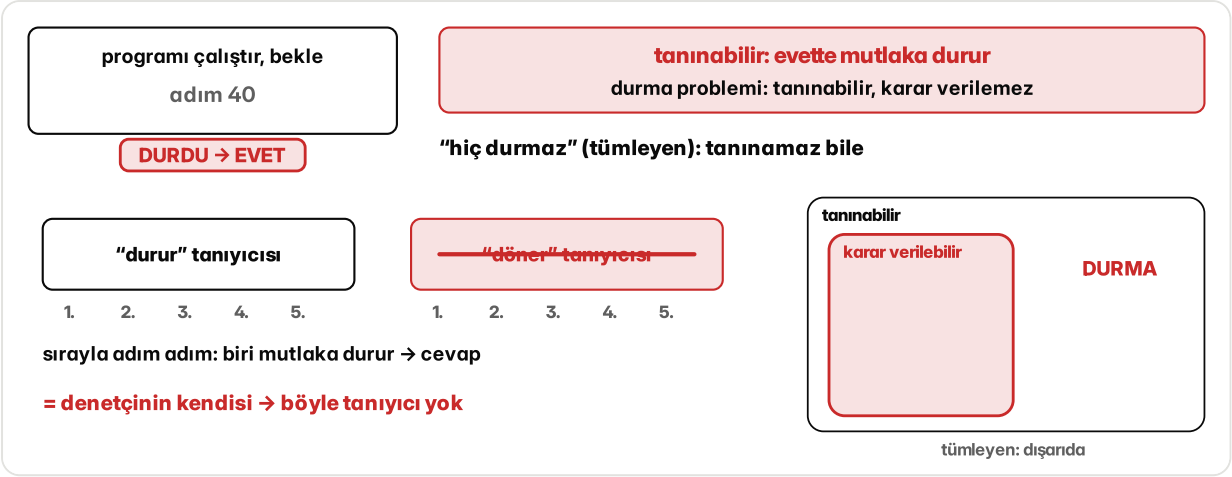
Ters her programdan, o programın köşegen hücresinde ayrılır: 1.'den 1. hücrede, 2.'den 2.'de. Ama Ters de bir program, listede bir satırı olmalı. O satırın köşeginde Ters kendi kodunu okur: durursa dönmeli, dönerse durmalı. Hücreye ne yazılsa yanlış; demek ki denetçi yazılamaz.



Şekil 10.5 Tabloya Ters satırı eklenmiş: ∞, D, D, ∞; Ters'in kendi köşegen hücresinde "?" var, çünkü durursa dönmeli, dönerse durmalı.

Turing 1936'da, makinelerin davranışıyla ilgili böyle bir soruya hiçbir makinenin karar veremeyeceğini köşegen yoluyla gösterdi; Alonzo Church aynı yıl başka bir yoldan aynı yere vardı. Hilbert'in sorusunun cevabı da çıktı: mekanik bir karar yolu yok. Bu soruya bugün **durma problemi (halting problem)** denir.

Yine de sorunun yarısı cevaplanır: programı çalıştırıp beklersek, duruyorsa bir gün görürüz. Durma problemi tanınabilir ama karar verilemez. "Hiç durmaz" cevabını ise hiçbir makine tanıyamaz. Tanısaydı, iki tanıyıcıyı adım adım sırayla çalıştırır, hangisi durursa onun cevabını alırdık; bu, denetçinin ta kendisi olurdu. Çift sayı programının durup durmayacağını bu yüzden genel bir yöntem söyleyemez; o soru, ona özel bir ispat bekliyor.



Şekil 10.6 Sırayla adım adım çalıştırılan "durur" ve "döner" tanıyıcıları; biri mutlaka durur ve cevap çıkar, bu da denetçinin kendisi olurdu; yanda tanınabilir, karar verilebilir ve durma kümeleri.

Kusursuz tarayıcı

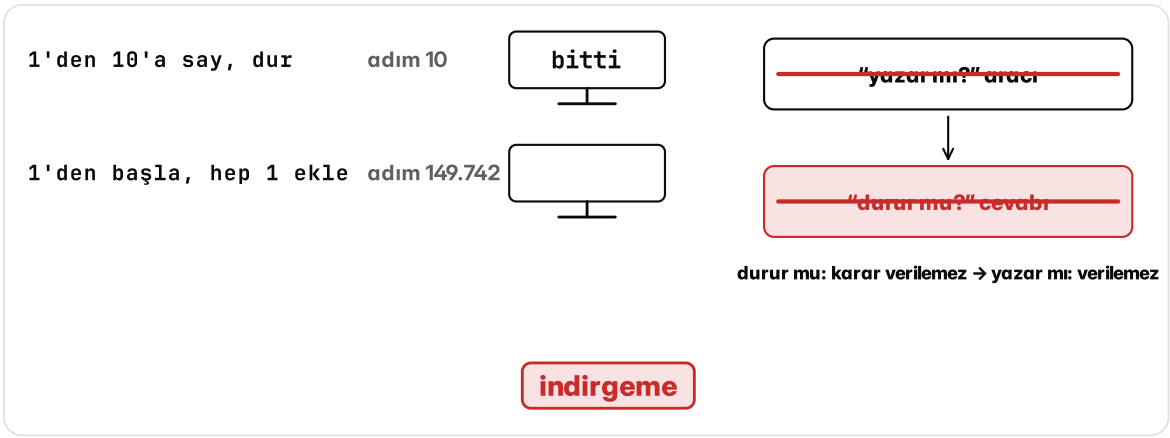
videoda 1:15:58

Virüs, çalışınca kendini başka dosyalara kopyalayan bir programdır. Virüs tarayıcısı bir dosyayı çalıştırmadan inceler ve onun virüs olup olmadığına karar vermeye çalışır. Her virüsü yakalayan, hiçbir temiz programa da virüs demeyen bir tarayıcı neden hiçbir zaman yazılamaz?

Elimizde cevabı her zaman bulunamayan bir soru var: bir program bir girdide sonunda durur mu, yoksa sonsuza kadar mı döner? Adı durma problemi. Her programda doğru cevabı veren bir programın yazılamayacağı 1936'dan beri ispatlı. Yeni bir soruya da karar verilemediğini göstermenin yolu, onu bu soruya bağlamaktır.

Önce daha basit bir soru: bir program ekrana herhangi bir şey yazar mı? Herhangi bir program ve bir girdi alalım, bunlardan yeni bir program kuralım. Yeni program önce eski programı o girdiyle, ekrana hiçbir şey yazdırmadan çalıştırır; eski program biterse ekrana "bitti" sözcüğünü yazar. Yani yeni program ekrana ancak eski program durursa yazar.

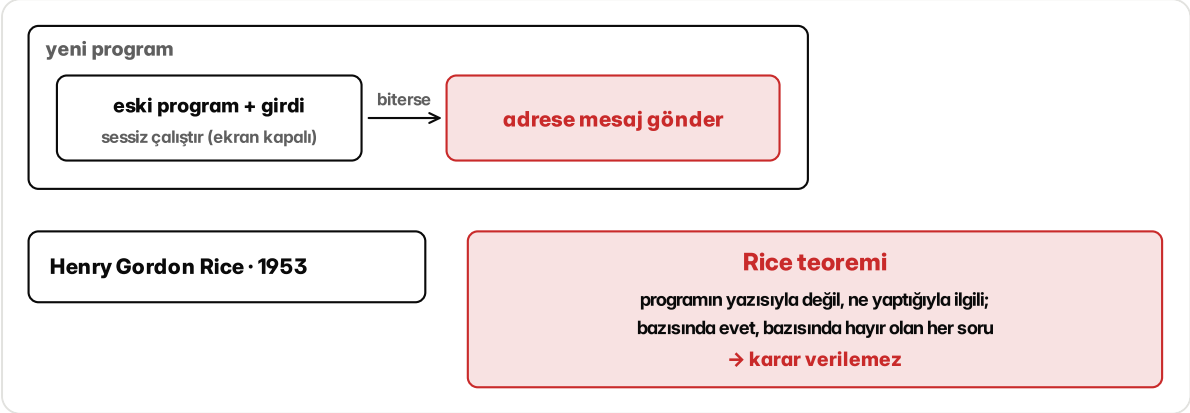
Eski program 1'den 10'a kadar sayıp dursun: yeni program 10 adım sonra "bitti" sözcüğünü yazar. Eski program 1'den başlayıp hep 1 eklesin: yeni program hiçbir şey yazmaz. "Ekrana yazar mı?" sorusunu her programda cevaplayan bir araç olsaydı, yeni programa sorup eskisinin durup durmadığını öğrenirdik. Durma problemine karar verilemediğine göre bu soruya da verilemez. Bu yola **indirgeme (reduction)** denir.



Şekil 10.7 İki örnek: 1'den 10'a sayan program 10 adımda durup "bitti" yazıyor, hep 1 ekleyen program durmuyor; altta "durur mu: karar verilemez, yazar mı: verilemez".

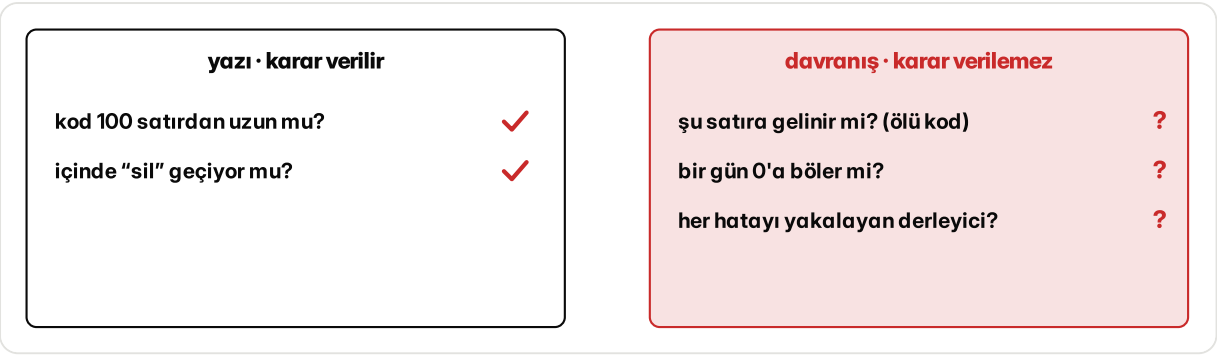
Virüs için de aynı kurulum işler: eski programı sessizce çalıştır, biterse kendini başka bir dosyaya kopyala. Yeni program ancak eski program dururs a virüs gibi davranır. Kusursuz bir tarayıcı ona virüs dediği anda, eski programın durduğunu da söylemiş olurdu. 1987'de yayımlanan bir çalışma (Fred Cohen), virüsün incelemeyle kesin olarak tespit edilemeyeceğini böyle bir bağla gösterdi.

Kurulumun gücü, sondaki işi istediğimiz gibi seçebilmemizde: dosya silsin, sıfıra bölsün, bir adrese mesaj göndersin. Rice 1953'te bunun genel kuralını ispatladı: bir programın yazısıyla değil, ne yaptığıyla ilgili, bazı programlarda evet, bazılarında hayır olan her soruya karar verilemez. Bu **Rice teoremi**dir.



Şekil 10.8 Rice teoremi: yeni program eski programı sessizce çalıştırır, biterse sondaki işi (örneğin adrese mesaj göndermek) yapar; teoremin cümlesi altta yazılı.

Teorem kodun yazısıyla ilgili sorulara dokunmaz: kod 100 satırdan uzun mu, içinde "sil" komutu geçiyor mu, bunlar kolayca bulunur. Karar verilemeyen, kodun çalışınca ne yapacağıdır: şu satıra hiç gelinir mi, yani orası ölü kod mu; program bir gün 0'a böler mi. Her hatayı yakalayan bir derleyici de bu yüzden yazılamaz.



Şekil 10.9 Solda yazıyla ilgili, karar verilen sorular (100 satırdan uzun mu, "sil" geçiyor mu); sağda davranışla ilgili, karar verilemeyen sorular (ölü kod, sıfıra bölme, her hatayı yakalayan derleyici).

Tarayıcılar bu yüzden kesin cevap değil, iyi tahmin peşindedir: bilinen virüslerin kod parçalarını arar, şüpheli programı kapalı bir ortamda bir süre çalıştırıp izler. Her tarayıcı ya bazı virüsleri kaçırır ya da bazı temiz programlara virüs der; ikisini birden 0'a indiren bir tarayıcı yazılamaz.

Özet

- Her girdide durup evet ya da hayır diyen makinesi olan soruya karar verilebilir, evette mutlaka duran makinesi olana tanınabilir denir.
- Post yazışma problemi tanınabilir ama karar verilemez; gramerin belirsizliği de ona bağlanır.
- Köşegen ispatında Ters, denetçinin her hücresinden ayrılır ve kendi hücresinde çelişir; bu yüzden durma problemine karar verilemez (Turing, 1936).
- Durma problemi tanınabilir, tümleyeni ("hiç durmaz") tanınamaz.
- Yeni bir sorunun karar verilemezliği, durma problemine indirgenerek gösterilir; Rice teoremi bunu davranışla ilgili her anlamlı soruya genişletir.
- Kusursuz bir virüs tarayıcısı ya da her hatayı yakalayan bir derleyici yazılamaz; kod yazısıyla ilgili sorulara ise karar verilir.

Sorular

1 **Bir soruya "karar verilebilir" denmesi için ne gerekir?**

A Evet cevabında durması yeterli, hayır cevabında sonsuza kadar sürmesi serbest olan bir makine bulunması.

B Her girdide durup evet ya da hayır diyen bir makine bulunması.

C Sorunun kısa bir ispatının bulunması.

D Sorunun hiçbir girdisinin sonsuza kadar sürmemesi, makine ne olursa olsun.

2 **Aşağıdaki kart takımlarından (üst / alt) hangisinin çözümü olmadığı, kartların uzunluklarına bakarak hemen anlaşılır?**

A 1 / 1 ve 01 / 10

B 0 / 00 ve 00 / 0

C 10 / 1 ve 011 / 01

D 11 / 1 ve 1 / 11

3 **Üç kart var: 1. kart 11 / 1, 2. kart 0 / 10, 3. kart 1 / 01 (üst / alt). Üst ve alt satırı aynı yapan en kısa dizilişin kart numaralarını sırayla yaz (örnek yazım: 213).**

Cevap: _____

4 **Üç kartlık bir takımda, körü körüne arayan program önce tek kartlık, sonra iki kartlık, sonra üç kartlık dizilişleri deniyor. Üç kartlığa kadar toplam kaç diziliş dener?**

Cevap: _____

5 **Durma problemi hakkında hangisi doğrudur?**

A Tanınabilir ama karar verilemez: duran programı çalıştırıp bekleyerek bir gün anlarız, sonsuza dönen için hiçbir makine kesin cevap veremez.

B Karar verilebilir: her programı çalıştırıp yeterince bekleyen makine doğru cevabı bulur.

C Tanınamaz: duran programı bile hiçbir makine anlayamaz.

D Yalnız çift sayı programları gibi özel programlar için karar verilemez.

6 **Dört programın kendi kodlarını girdi alınca sonuçları sırasıyla şöyle: dur, dur, dön, dur. Köşegen ispatındaki Ters programı bu dört koda hangi sonuçları verir?**

A dur, dur, dön, dur

B dön, dur, dur, dön

C dur, dön, dön, dur

D dön, dön, dur, dön

7 **Rice teoremine göre hangisine bir programla her girdide karar verilemez?**

A Programın kodu 50 satırdan kısa mı?

B Program bir gün 0'a bölme yapar mı?

C Kodun içinde "sil" sözcüğü geçiyor mu?

D Kodun ilk harfi G mi?

8 **"Bir program ekrana bir şey yazar mı?" sorusunu her programda doğru cevaplayan bir araç yazıldığını varsayalım. Hangisi sonuçta çıkar?**

A Bu aracın hiçbir sakıncası yok; yalnız ekranla ilgili bir soruyu cevaplıyor.

B Her virüs tarayıcısı hatasız olurdu, ama durma problemi yine çözülemezdi.

C Durma problemi çözülmüş olurdu; bu imkânsız olduğundan böyle bir araç yazılamaz.

D Araç yalnız eski programlarla çalışırdı, yeni programları denetleyemezdi.

BİÇİMSEL DİLLER VE OTOMATA

11 P, NP ve NP-tamlık

Bu bölüm, bir problemin ne zaman hızlı çözülür sayıldığını ve hangi problemlerin bunca zamandır hızlı bir yönteme direndiğini anlatır: adım sayısıyla zaman ölçümü, P ve NP sınıfları, P = NP sorusu ve NP-tam problemler.

Saniye mi, adım mı?

videoda 1:18:27

Aynı program eski bir bilgisayarda 1 dakika, yenisinde 1 saniye sürebilir. Bir problemin ne kadar sürede çözüldüğünü saniyeyle ölçersek, cevap problemi değil bilgisayarı anlatır. Süre makineye göre değişiyorsa, bir problemin hızlı çözüldüğünü nasıl söyleriz?

Saniye yerine adım sayalım: Turing makinesinin her hamlesi, yani bir kareyi okuyup yazması ve bir kare kayması, bir adımdır. Örnek bir soru: bir kelimenin harflerinin hepsi farklı mı? Kavunda öyle, hiçbir harf tekrar etmiyor. Portakalda ise a harfi iki kez geçiyor.

Tek şeritli makine ilk harfin üstünü işaretler ve harfi durumunda aklında tutar. Sağa yürüyüp sonraki her harfle karşılaştırır; aynısını görürse reddeder. Kelimenin sonuna varınca işaretli harfe döner, bir sağdakine geçer ve aynısını yapar. Kavunda bütün harfleri bu yolla karşılaştırıp 36 adımda kabul eder. Portakalda 2. a'yı görünce durur: 59 adımda ret.



Şekil 11.1 Kavun ve portakal girdisinde makinenin şeridi ve adım sayısı: kavun 36 adımda kabul, portakal 59 adımda ret.

Aynı uzunluktaki kelimeler farklı sürer: tekrar erken bulunursa makine erken durur. Harfleri hepsi farklı 8 harfli bir kelime, portakaldan uzun sürerdi: 81 adım. Bu yüzden süre, o uzunluktaki en uzun süren girdiyle ölçülür (en kötü durum); burada harfleri hepsi farklı olan kelimeyle.

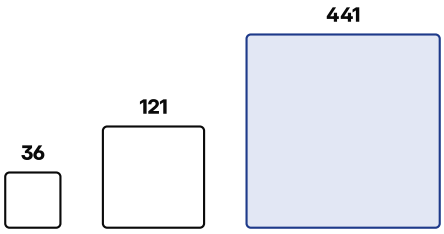
Harfleri hepsi farklı 5 harfte 36 adım, 10 harfte 121, 20 harfte 441. Uzunluk 2 katına çıktıkça adım sayısı 4 katına yaklaşıyor: hem karşılaştırılacak harf 2 katı, hem her harf için yürünen yol. Bu sayılar $(n + 1)^2$ ile uyuşur; yani adım sayısı uzunluğun karesiyle büyüyor. Zamanı tek bir sayıyla değil, girdi uzadıkça adım sayısının nasıl büyüdüğüyle ölçeriz.

farklı harf	adım	oran
5	36	
10	121	× 3,4
20	441	× 3,6

adım = (n + 1)²

n: harf sayısı

zaman = girdi uzadıkça adım sayısının büyüyüşü

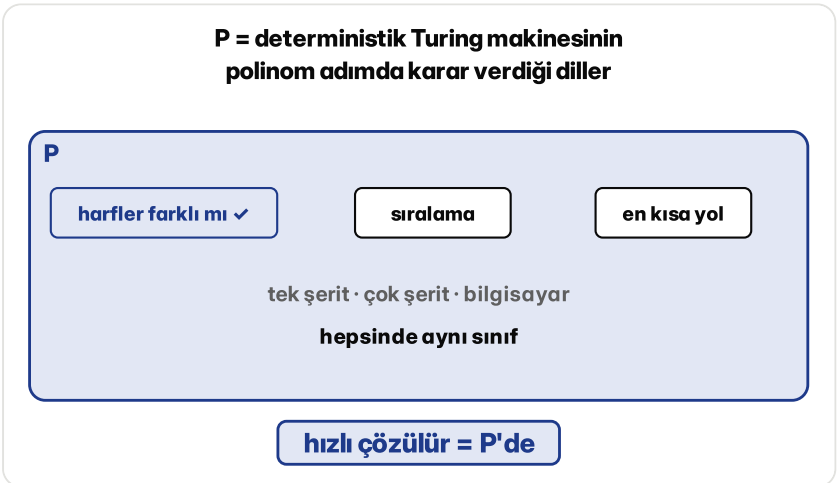


Şekil 11.2 Uzunluk iki katına çıkınca adım sayısı yaklaşık dört katına çıkar: 5 harfte 36, 10 harfte 121, 20 harfte 441 adım.

Uzunluğun karesi, küpü ya da sabit herhangi bir kuvvetiyle büyüyen adım sayısına **polinom** denir. Oysa her yeni harfle 2'ye katlanan adım sayısı polinom değildir: 30 harfte 1 milyarı aşar (2^{30} = 1.073.741.824; 30^2 yalnız 900'dür).

Sınırın polinomda çekilmesinin nedeni şu: makine değişince adım sayısı değişir ama polinom kalır. Çok şeritli bir makinenin işini tek şeritli makine, onun adım sayısının kabaca karesi kadar adımda yapar; polinomun karesi de polinomdur, kare 4. kuvvete döner. Bugünkü bilgisayarların işi de Turing makinesine polinomla sınırlı bir bedelle aktarılır.

Deterministik bir Turing makinesinin, yani her adımda tek bir kuralı izleyen makinenin, polinom adımda karar verdiği dillerin, yani cevabı evet olan girdi kümelerinin sınıfına **P** denir. Harflerin hepsinin farklı olup olmadığı P'de; bir listeyi sıralamak ya da haritada en kısa yolu bulmak da polinom adımda yapılır. Hangi makineyi seçersek seçelim, P'deki bir problem P'de kalır. Hızlı çözülür demenin biçimsel anlamı budur: P'de olmak.



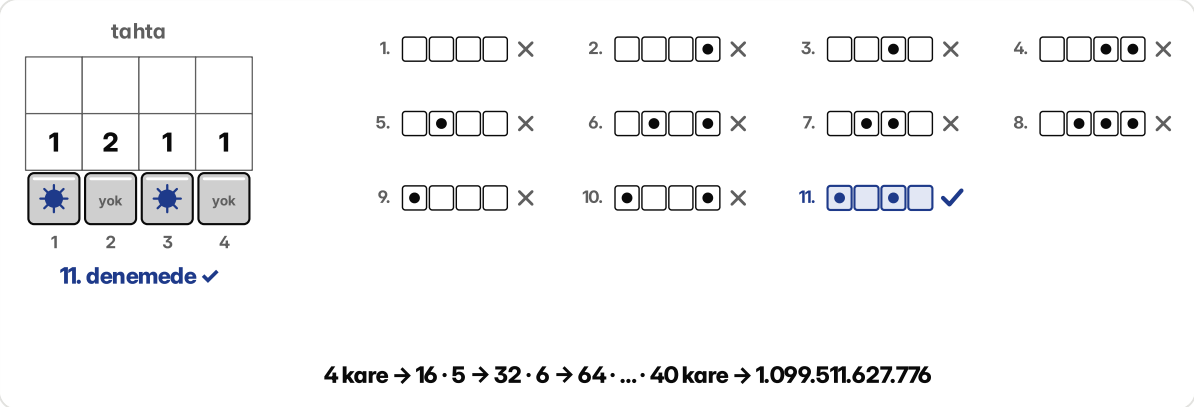
Şekil 11.3 P sınıfı: adım sayısı girdinin uzunluğunun bir polinomuyla sınırlı problemler.

Dört kapalı kare

videoda 1:20:50

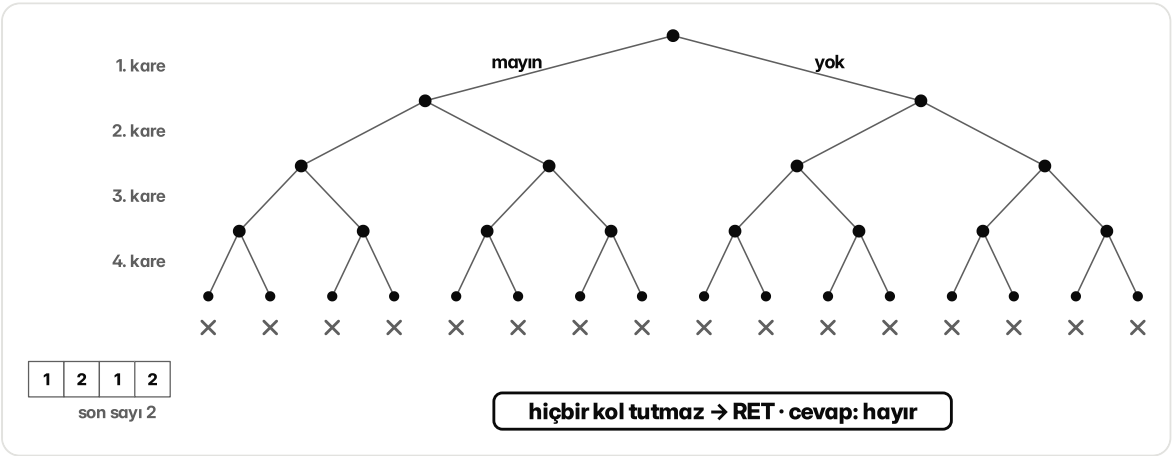
Mayın tarlasında açılmış bir karedeki sayı, çevresindeki 8 kareden kaçında mayın olduğunu söyler. Tahtamız 3 × 4: üst satır açık ve boş, orta satırda 1, 2, 1, 1 yazıyor, alt satırdaki 4 kare kapalı. Bu sayıların hepsine uyan bir mayın yerleşimi var mı? Bu soruyu her tahtada hızlı cevaplayan bir yöntem, neden 1 milyon dolarlık bir soruyu da cevaplamış olur?

Her kapalı karede ya mayın vardır ya yoktur; 4 karede 2 × 2 × 2 × 2 = 16 yerleşim. Deterministik bir makine onları sırayla dener, her birinde sayıları sayar. Bu tahtada yalnız 1 yerleşim tutar: 1. ve 3. karede mayın (1010). Sırayla deneyen makine onu 11. denemede bulur. Ama her yeni kapalı kare yerleşim sayısını 2'ye katlar: 40 karede 1 trilyonu aşar.



Şekil 11.4 Dört kapalı karenin 16 yerleşimi sırayla denenir; tutan yerleşim 11. denemede bulunur.

Deterministik olmayan bir Turing makinesi ise bir adımda birden çok kuralı izleyip kollara ayrılabilir. Bizim makine her kapalı karede ikiye ayrılır: bir kolda mayın var, öbüründe yok. 16 kolun her biri kendi yerleşimini denetler. Kollardan biri bile tutarsa makine kabul eder; hiçbiri tutmazsa reddeder. Son sayı 2 olsaydı (1, 2, 1, 2) hiçbir kol tutmazdı: cevap hayır.

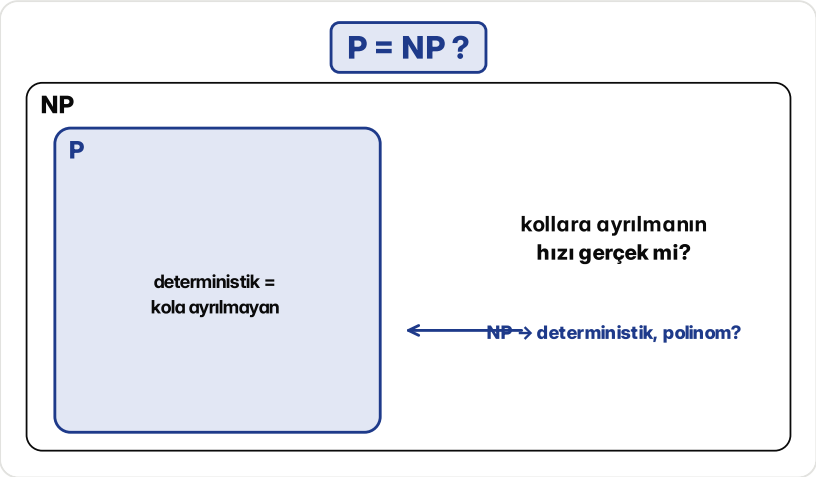


Şekil 11.5 Deterministik olmayan makine her kapalı karede ikiye ayrılır ve 16 kola bölünür.

Bir kolun işi kısadır: tahminleri yazmak her kare için 1 adım, denetlemek her sayı için en çok 8 komşuya bakmak. Tahta büyüdükçe iş, tahtanın büyüklüğüyle orantılı büyür: polinom adım. Her kolu polinom adımda biten deterministik olmayan bir makinenin tanıdığı dillerin sınıfına **NP** denir; mayın tarlası NP'dedir.

Aynı tanımın ikinci bir söylenişi var: cevap evetse bunun kısa bir kanıtı vardır, burada mayınların yeri, ve deterministik bir makine bu kanıtı polinom adımda denetler. Kanıt, kabul eden kolun yaptığı tahminlerin listesidir.

Deterministik makine, hiç kola ayrılmayan deterministik olmayan makinedir; bu yüzden P'deki her dil NP'de de bulunur (P ⊆ NP). Büyük soru tersini sorar: kollara ayrılmanın sağladığı hız gerçek mi? NP'deki her dile deterministik bir makine de polinom adımda karar verebilir mi? Kısaca: P, NP'ye eşit mi?



Şekil 11.6 P, NP'nin içinde; ikisinin eşit olup olmadığı açık soru.

Clay Matematik Enstitüsü 2000 yılında çözülmemiş 7 büyük matematik problemi seçti, her birine 1 milyon dolar ödül koydu. P = NP sorusu bunlardan biri ve hâlâ açık. Araştırmacıların çoğu eşit olmadığını düşünüyor: 2019'daki bir ankette 124 yanıtın 109'u (%88). Ama ispat yok; ödül, iki cevaptan birini ispatlayana verilir.

2000'de bir matematikçi (Richard Kaye), mayın tarlasının NP'nin en zor problemlerinden biri olduğunu gösterdi: NP'deki her problem polinom adımda bir tahtaya çevrilebilir; cevabı evet olan girdi tutarlı, hayır olan tutarsız bir tahtaya döner. Her tahtada hızlı cevap veren bir yöntem, NP'deki her problemi de hızlı çözer: P = NP olur.

Davet bulmacası

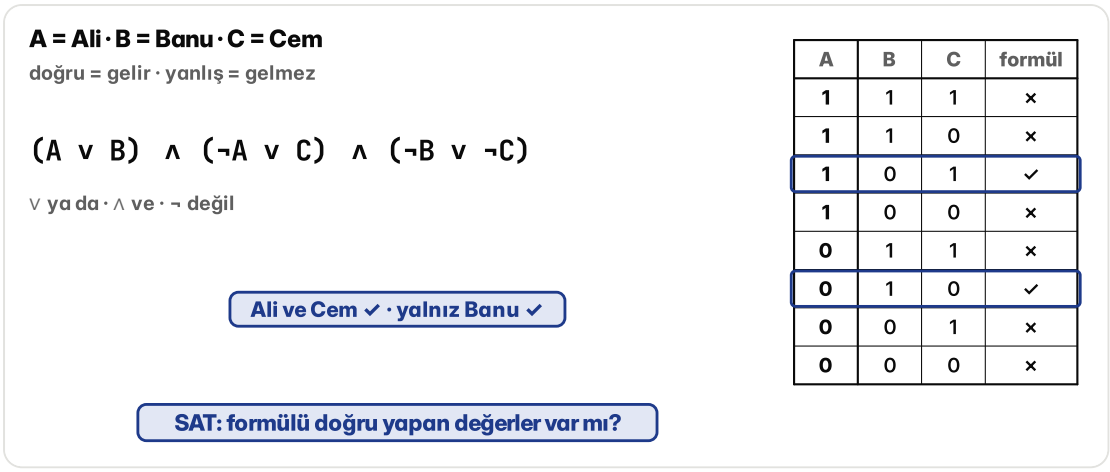
videoda 1:23:20

Bir davete Ali, Banu ve Cem çağrılacak. 3 koşul var: Ali ya da Banu gelsin; Ali gelirse Cem de gelsin; Banu ile Cem birlikte gelmesin. Kimleri çağırmalı? Bu türden bir mantık bulmacası, birbirine hiç benzemeyen binlerce zor problemin ortak düğümüdür. Bu kadar farklı problem neden tek bir bulmacaya bağlanır?

Her kişi bir değişken olsun: gelirse doğru, gelmezse yanlış. Koşullar bir formüle döner:

```
(Ali ∨ Banu) ∧ (¬Ali ∨ Cem) ∧ (¬Banu ∨ ¬Cem)
```

Ali ile Cem gelip Banu gelmezse üç koşul da doğru; yalnız Banu gelirse de. Formülü doğru yapan değerler var mı sorusuna **SAT** denir: mantık formülünü doğru yapma problemi.

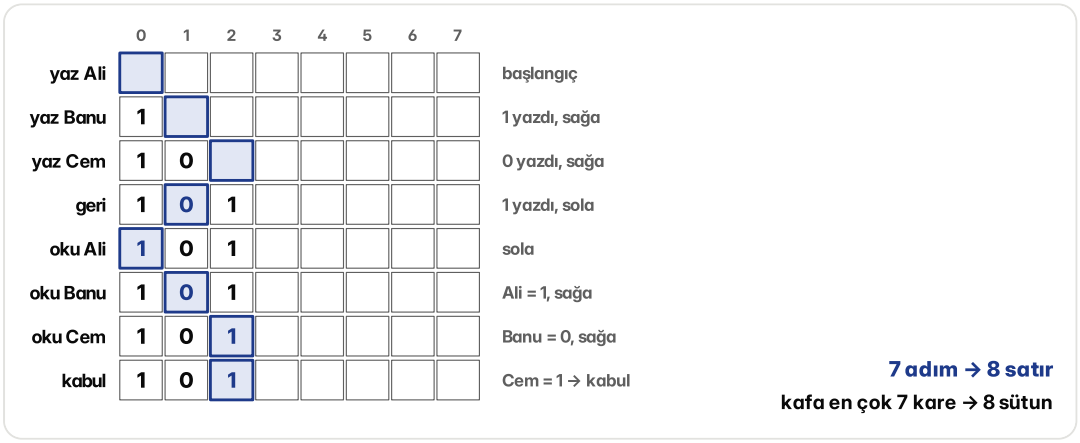


Şekil 11.7 Davet bulmacasının üç koşulu mantık formülüne çevrilir.

Bulmacayı formüle çevirmek koşul sayısıyla orantılı adım sürdü ve cevabı korudu. Bir problemin girdilerini cevabı koruyarak, polinom adımda başka bir problemin girdilerine çevirmeye **polinom indirgeme** denir. NP'deki her problemin indirgenebildiği probleme **NP-zor**, NP'de de bulunan NP-zor probleme **NP-tam** denir. Bir NP-tam problemi hızlı çözen, NP'nin hepsini hızlı çözer: önce çevirir, sonra çözer.

1971'de Stephen Cook, SAT'ın NP-tam olduğunu gösterdi; Sovyetler Birliği'nde Leonid Levin de aynı sonuca ondan habersiz ulaştı (Cook-Levin teoremi). SAT NP'dedir: deterministik olmayan makine değerleri tahmin eder, formülü polinom adımda hesaplar. Ama NP'deki sonsuz sayıda problem SAT'a nasıl çevrilir? Fikir, problemi değil onu çözen makineyi çevirmek.

Davet sorusunu çözen deterministik olmayan makine 3 tahminini şeride yazar, sonra başa dönüp koşulları denetler. Kabul eden kolun her adımı bir satır olsun: şerit, kafanın yeri ve durum. İlk satır başlangıç, her satır bir öncekinden bir kuralla çıkar, son satırda kabul var. 7 adım, 8 satır; kafa en çok 7 kare gidebildiği için 8 sütun.

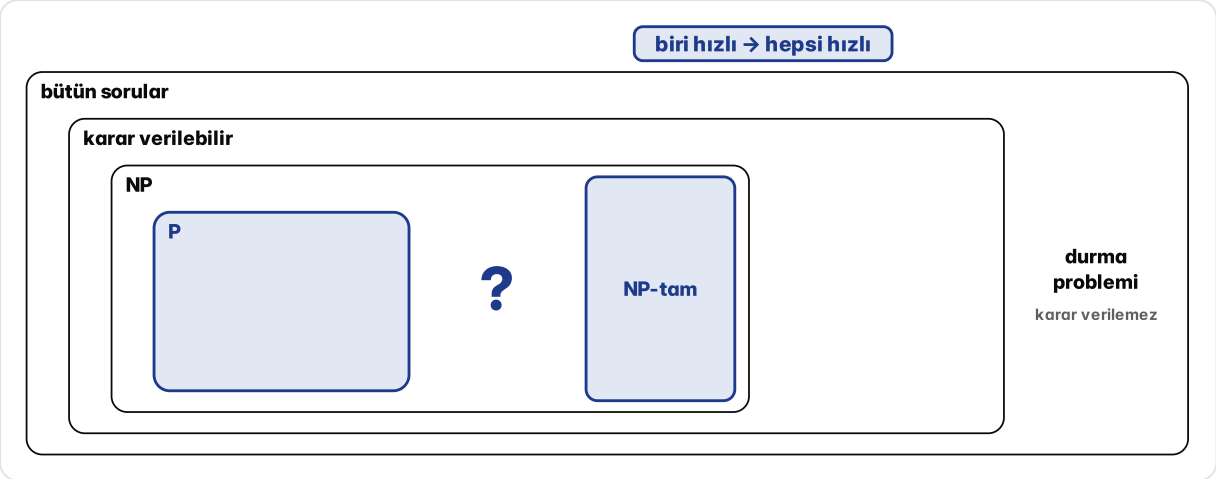


Şekil 11.8 Kabul eden çalışmanın tablosu: her satır bir adımdaki şerit, kafa ve durum.

Her hücre için "burada şu simge var" diyen değişkenler kurulur. Koşullar: her hücrede tam 1 simge; ilk satır başlangıç; son satırda kabul; kafanın olmadığı hücre alt satıra aynen geçer, kafanın altındaki hücre makinenin bir kuralına uyar. Bu formülü doğru yapan her seçim, kabul eden bir çalışmanın tablosudur. Davet makinesinin tablosundan çıkan formülde 431 değişken ve 8.167 koşul vardır; adım sayısı 2 katına çıkınca formül 4 katına yaklaşır, yani polinom kalır.

Her koşulunda en çok 3 değişken olan 3-SAT da NP-tamdır. Uzun bir koşul yardımcı bir değişkenle bölünür: "Ali, Banu, Cem ya da Deniz'den biri gelsin" koşulu 2 koşul olur: (Ali ∨ Banu ∨ y) ∧ (¬y ∨ Cem ∨ Deniz). Uygun bir yardımcı ancak dördünden biri gelirse bulunur. SAT'tan başlayan böyle çevirilerle binlerce problemin NP-tam olduğu gösterildi.

P, NP'nin içinde; NP-tam problemler NP'nin en zor kenarında; iki sınıf aynı mı, bilinmiyor. Hepsi karar verilebilir; dışlarında, durma problemi gibi hiçbir makinenin karar veremediği sorular durur. Bir problemin NP-tam olduğunu göstermek güçlü bir uyarıdır: onu hızlı çözen yöntem, kimsenin hızlı çözemediği binlerce problemi birden çözerdi.



Şekil 11.9 Sınıfların haritası: P ve NP-tam problemler NP içinde, NP karar verilebilir problemlerin içinde; durma problemi dışarıda.

Özet

- Zaman saniyeyle değil, girdi uzadıkça Turing makinesinin adım sayısının nasıl büyüdüğüyle ölçülür; en kötü girdi esas alınır.
- Polinom adım, makine değişince de polinom kaldığı için sınır olarak seçilir; P, polinom adımda karar verilen problemlerin sınıfıdır.
- NP, deterministik olmayan makinenin polinom adımda tanıdığı ya da kısa kanıtı polinom adımda denetlenen problemlerin sınıfıdır; P ⊆ NP.
- P = NP sorusu açıktır ve 1 milyon dolarlık Clay ödülüne sahiptir.
- Bir NP-tam problemi hızlı çözen, NP'nin hepsini hızlı çözer; SAT, Cook-Levin teoremiyle ilk NP-tam problemdir.
- NP-tam bir problemin hızlı yöntemi bilinmez; bunu göstermek güçlü bir uyarıdır.

Sorular

1 **Bir problemin ne kadar hızlı çözüldüğünü neden saniyeyle değil adım sayısıyla ölçeriz?**

A Adımlar saniyeden daha küçük bir birimdir.

B Saniye problemi değil bilgisayarı anlatır; adım sayısı ise girdi uzadıkça nasıl büyüdüğünü gösterir.

C Turing makinesi saniyeyi ölçemez.

D Adım sayısı her makinede tam olarak aynıdır.

2 **Harflerinin hepsi farklı olup olmadığına bakan makine, harfleri hepsi farklı 8 harfli kelimede 81 adım atıyor. Harfleri hepsi farklı 9 harfli bir kelimede kaç adım atar?**

Cevap:

3 **Girdi uzunluğu n iken aşağıdaki adım sayılarından hangisi polinom değildir?**

A n^2

B $5n^3 + 2$

C 2^n

D n^{100}

4 **Bir mayın tarlası tahtasında 6 kapalı kare var. Her karede ya mayın vardır ya yoktur. Kaç farklı yerleşim denenmelidir?**

Cevap:

5 **Mayın tarlası sorusunda cevap "evet" ise, NP tanımındaki "kısa kanıt" nedir?**

A Mayınların yeri; deterministik bir makine bunu polinom adımda denetler.

B Bütün yerleşimlerin listesi.

C Tahtadaki sayıların toplamı.

D Yerleşimlerin 2^n adımda tek tek denendiğini gösteren bir kayıt.

6 **Davet bulmacasının üç koşulu var: (1) Ali ya da Banu gelsin; (2) Ali gelirse Cem de gelsin; (3) Banu ile Cem birlikte gelmesin. Yalnız Ali gelirse üç koşuldan kaçı sağlanır?**

Cevap:

7 **P ile NP arasındaki ilişki için hangisi doğrudur?**

A P ve NP ortak dil içermeyen ayrı sınıflardır.

B NP'deki her dilin P'de de olduğu ispatlandı.

C NP'ye yalnız mayın tarlası gibi birkaç özel problem girer.

D P'deki her dil NP'dedir, çünkü deterministik makine hiç kola ayrılmayan deterministik olmayan makinedir.

8 **NP-tam bir problem için polinom adımda çalışan deterministik bir yöntem bulunsaydı ne olurdu?**

A Yalnız o problem P'ye girerdi, öteki NP problemleri aynı kalırdı.

B NP'deki her problem polinom adımda çözülürdü; yani P = NP olurdu.

C Durma problemi de karar verilebilir olurdu.

D SAT artık NP'de sayılmazdı.

Cevaplar ve çözümler

0 Dizgiler, diller ve makineler

1. **C** · 4

   Uzunluk, dizgideki simge sayısıdır: 0, 1, 1, 0 olmak üzere 4 simge var. Alfabedeki simge sayısı (2) dizginin uzunluğu değildir.

2. **B** · 10 TL

   İlk beşlik makineyi 5 TL durumuna, ikincisi 10 TL durumuna geçirir. Toplam 15'e henüz varmadığı için çay gelmez. Makinenin 15 TL diye bir durumu yoktur; 15'e varınca çay verip başa döner.

3. **0**

   5 + 5 = 10 olur, üçüncü beşlik toplamı 15'e çıkarır. Makine çay verir ve başa, 0 TL durumuna döner.

4. **D** · 01110

   xy, önce x'i sonra y'yi yazar: 01 ve 110 uç uca 01110 olur; uzunluk 2 + 3 = 5. 11001, yx'tir (önce y, sonra x); sıra değişince dizgi de değişir.

5. **15**

   Uzunluk 0'da 1 (boş dizgi), uzunluk 1'de 2, uzunluk 2'de 4, uzunluk 3'te 8 dizgi var. Toplam 1 + 2 + 4 + 8 = 15.

6. **A** · Bir dizgisi (boş dizgi) vardır, boş dil değildir.

   {ε} kümesinin tek elemanı boş dizgidir; yani içinde bir dizgi vardır. Hiç dizgisi olmayan dil ø'dir (boş dil). Dil, dizgilerden oluşan her kümedir; elemanının harf içermesi gerekmez.

7. **D** · 10'dan küçük asal sayıların ikilik yazımları

   10'dan küçük asallar 2, 3, 5, 7'dir; ikilik yazımları 10, 11, 101, 111 olan 4 dizgilik sonlu bir dil çıkar. Ötekilerin hepsi hiç bitmez: asallar bitmez, hatasız bir programa satır eklenebilir, her uzunlukta yeni dizgiler vardır.

8. **B** · Hayır, çünkü 21 = 3 × 7, asal değildir.

   10101 = 16 + 4 + 1 = 21 ve 21 = 3 × 7 olduğu için asal değildir; dizgi dilde yoktur. Üyelik sorusunun cevabı, sayının asal olup olmamasına bağlıdır; yazımın biçimine ya da uzunluğuna değil. Tek sayı olmak asal olmak için yetmez.

1 Matematiksel araçlar ve ispat yöntemleri

1. **B** · 13

   12 kişi 12 ayrı aya dağılabilir; her kutuda bir kişi olur. 13. kişi hangi aya girerse girsin dolu bir kutuya düşer. Güvercin yuvası ilkesi: güvercinler yuvalardan fazlaysa bir yuvada birden çok güvercin vardır.

2. **4**

   Her güne 3 öğrenci koyarsak 7 × 3 = 21 kişi eder. Kalan 4 öğrenci nereye girerse girsin, o günün öğrenci sayısı 4 olur. 100 kişi ve 12 ay örneğindeki yolun aynısı.

3. **D** · n = 40 için sonucun 1.681 = 41 × 41 olması

   İddia her n için asal diyor; asal olmayan tek bir değer yeter. n = 40 için 1.600 + 40 + 41 = 1.681 = 41 × 41 asal değildir: bu bir karşı örnektir. Asal çıkan değerler iddiayı yıkmaz, yalnız doğrular; formülün çift sayı vermemesi de asal olmayı garanti etmez.

4. **C** · $n^2 + n = n \times (n + 1)$ ardışık iki sayının çarpımıdır, hep çifttir; çift + 41 tek olur

   Ardışık iki sayıdan biri çift olduğu için $n \times (n + 1)$ hep çifttir. Çifte tek olan 41 eklenince sonuç tek çıkar. Bu bir doğrudan ispattır; 40 deneme bir ispat olmaz, asal olmak da zaten formülün doğru bir özelliği değildir ($n = 40$'ta asal çıkmıyor).

5. **D** · tek durumunda, reddeder

   5 harf okunur ve her harfte durum değişir: çift, tek, çift, tek, çift, tek. Dizgi tek durumunda biter; kabul durumu çift olduğu için dizgi reddedilir. Dizginin uzunluğu 5, yani tektir.

6. **A** · Bütün diller hiçbir listeye sığmaz; diller sayılamaz.

   Bütün dilleri bir listeye dizdiğimizi varsayıp köşegeni ters çevirirsek listedeki hiçbir satıra eşit olmayan bir satır çıkar; varsayım çelişir. Dizgiler ve programlar sayılabilir, asallar bitmez; sayılamayan şey dillerdir.

7. **0101**

   Köşegenin her hanesi ters çevrilir: 1 → 0, 0 → 1, 1 → 0, 0 → 1. Sonuç 0101. Bu satır her satırdan köşegen haneyle ayrıldığı için listede olamaz.

8. **9**

   Uzunluğu 3 olan dizgiler 000, 001, 010, ... diye sürer. 000 8. sırada olduğuna göre 001 hemen ardından, 9. sıradadır.

2 Deterministik sonlu otomat

1. **C** · Kalan 2

   Yeni kalan, eski kalanın 2 katı artı yeni bittir: $2 \times 2 + 1 = 5$, 5'in 3'e bölümünden kalan 2. Makine aynı durumda kalır: 2 durumundan 1 oku yine 2 durumuna döner. Bu makinede tuzak durumu yoktur.

2. **1**

   Durumlar sırayla: 0 → 1 (bit 1) → 2 (bit 0) → 2 (bit 1) → 2 (bit 1) → 1 (bit 0). Son kalan 1; 10110 sayısı 22'dir ve $22 = 3 \times 7 + 1$. Kabul durumu kalan 0 olduğu için dizgi reddedilir.

3. **D** · 11011

   11011 = 27 ve 27, 3'e bölünür; makine 0 → 1 → 0 → 0 → 1 → 0 durumlarından geçip kalan 0'da biter. 1010 = 10, 100 = 4 ve 111 = 7 sayıları 3'e bölünmez; makine kalan 1'de durur.

4. **A** · Her hücrede tam olarak bir durum yazar.

   Deterministik makinede her durum ve simge çifti için sonraki durum tek ve belli olur: boş hücre yok, iki durum yazan hücre yok. Hücre sayısı durum sayısı çarpı simge sayısıdır; dizginin uzunluğuna bağlı değildir.

5. **tuzak**

   a ile yalnız harf durumuna geçilir, b orada tutar. Boşluk gelince makine tuzağa düşer ve c ile 1 onu tuzaktan çıkaramaz. Dizgi tuzakta bittiği için reddedilir.

6. **B** · 42ab

   42ab: 4 ile yalnız rakam, 2 orada tutar, a gelince ikisi de olur ve b ile kalır: kabul. 2024'te harf, kedi'de rakam yok; ret. k ed1'de boşlukta tuzağa düşülür, 1 gelse de çıkılmaz.

7. **D** · anne

   anne: a ile kalına geçilir, e gelince uyum bozulur ve makine bozuk durumunda kalır; ret. kelebek tamamen ince, armut ve okul tamamen kalın ünlülerden oluşur; üçü de kabul edilir.

8. **C** · elma

   Yeni makine tümleyen dili, yani uyuma uymayan kelimeleri tanır. elma: e ile inceye geçilir, a gelince bozuğa düşülür; bozuk artık kabul durumudur. armut, kelebek ve çiçek uyumludur; ünlü yok, kalın ya da ince durumunda biter, yani reddedilir.

3 Deterministik olmayan sonlu otomat

1. **C** · Aynı harfle birden çok ok ya da hiç ok bulunabilmesi

   DFA her durumda her harf için tam bir yola gider. NFA'da bir durumdan aynı harfle birden çok ok çıkabilir ya da hiç ok çıkmayabilir. Okuma yönü ve kabul durumu sayısı iki makinede de serbesttir.

2. **A** · Dizgi kabul edilir; kopyalardan en az birinin kabul durumunda olması yeter.

   NFA'da kural şudur: yollardan en az biri kabul durumunda bitiyorsa dizgi kabul edilir. Yanlış tahmin eden kopyalar ya silinir ya da kabul dışı bir durumda kalır; bu sonucu değiştirmez.

3. **D** · {s, z_0, b_0}

   Epsilon kapanışı, bir durumdan yalnız ε geçişleriyle varılabilen bütün durumlardır ve durumun kendisini de içerir: s, z_0 ve b_0. Makine işe bu 3 durumda birden başlar.

4. **5**

   Eski durumların hiçbiri silinmez ve hiçbir ok değişmez: 2 + 2 = 4 durum. Üstüne yeni başlangıç durumu s eklenir: 5 durum.

5. **B** · 1 1 0 0

   1 1 0 0 dizgisinde sondan üçüncü harf 1'dir; o 1'i tahmin eden kopya 2 harf sonra 0 kaldı'ya varır. Öteki dizgilerde sondan üçüncü harf 0'dır (1 0 0 0'daki 1 sondan dördüncüdür), hiçbir kopya 0 kaldı'ya varamaz.

6. **3**

   Kümeler: {bekle} → {bekle, 2 kaldı} → {bekle, 2 kaldı, 1 kaldı} → {bekle, 2 kaldı, 1 kaldı, 0 kaldı} → {bekle, 1 kaldı, 0 kaldı}. Son harf 0 okununca bekle kendine döner, 2 kaldı 1 kaldı'ya, 1 kaldı 0 kaldı'ya geçer; 0 kaldı'daki kopyanın gidecek oku olmadığı için silinir. Kümede 3 durum kalır ve 0 kaldı içinde olduğu için dizgi kabul edilir.

7. **C** · {bekle, 2 kaldı, 0 kaldı}

   {bekle} → 0: {bekle} → 1: {bekle, 2 kaldı} → 1: {bekle, 2 kaldı, 1 kaldı} → 0: {bekle, 1 kaldı, 0 kaldı} → 1: {bekle, 2 kaldı, 0 kaldı}. Son 1 harfi bekle'den 2 kaldı'ya yeni bir kopya doğurur, 1 kaldı'daki kopya 0 kaldı'ya geçer, eski 0 kaldı kopyası ise silinir. 0 kaldı kümede olduğu için kabul.

8. **B** · 1 0 1 1 0

   1 0 1 1 0 dizgisi 1 ile başlar, 0 ile biter; son harf 0 ile z_1'e varılır. 0 1 0 baştaki 0'da, 1 1 1 ve 1 0 1 ise son harf 1 olduğu için z_1'de bitemez; z_1'den 1 ile ok yoktur.

4 NFA'dan DFA'ya: eşdeğerlik ve küçültme

1. **B** · O ana kadar okunan dizgiyle NFA'nın aynı anda bulunabildiği durumların kümesini

   Bilgisayar tahmin edemez ama her harften sonra NFA'nın bulunabildiği durumları bir küme olarak yazabilir. Her küme yeni makinenin tek bir durumu olur; yeni makine her harfte tek bir kümeye geçer.

2. **D** · 16

   Her durum kümede ya vardır ya yoktur: 2 × 2 × 2 × 2 = 2^4 = 16 alt küme. DFA'nın kurulabilecek en çok durum sayısı budur. Sondan üçüncü harf örneğinde bekle her kümede bulunduğu için yalnız 2^3 = 8 küme kurulur.

3. **C** · {bekle, 2 kaldı, 0 kaldı}

   Kümedeki her durumun 1 ile gidebildiği yerler toplanır: bekle → bekle ve 2 kaldı, 1 kaldı → 0 kaldı. Birleşim {bekle, 2 kaldı, 0 kaldı}.

4. **001**

   Her küme dizginin son 3 harfini yazar: dizginin son üç harfi 0 0 1. Küme {bekle, 2 kaldı}: son harf 1 olduğu için 2 kaldı var, sondan ikinci ve sondan üçüncü harf 0 olduğu için 1 kaldı ve 0 kaldı yok. 0 kaldı içinde olmadığından dizgi reddedilir.

5. **A** · 24-24 ile 25-25

   24-24 ve 25-25'in ikisinde de bir takım art arda 2 sayı alırsa seti alır, sayılar paylaşılırsa yine eşitliğe dönülür. 25-24 ile 24-25 farklıdır: A harfiyle biri kabule gider, öteki gitmez. A kazandı kabul, B kazandı ret durumudur.

6. **B** · 25-24 ile 26-25

   25-24 ve 26-25, A harfiyle kabul grubuna (A kazandı) gider; ret grubunun öteki durumları A harfiyle yine ret grubunda kalır. Aynı harfle farklı gruplara giden durumlar ayrılır; geleceği aynı olamaz.

7. **D** · İkisi de küçültülür ve en küçük makineler okların düzeniyle karşılaştırılır.

   Her düzenli dilin tek bir en küçük DFA'sı vardır (durumların adı dışında). İki makine küçültülüp örtüştürülürse aynı dili tanırlar; örtüşmezlerse farkı gösteren bir dizgi bulunur. Durum sayısı tek başına yetmez, 8 durumlu makine ile 5 durumlu makine aynı dili tanıyabilir.

8. **64**

   Sondan n'inci harf dilinde en küçük DFA 2^n durum ister: sondan 3. için 8, sondan 4. için 16, sondan 10. için 1.024. n = 6 için 2^6 = 64 durum gerekir; NFA ise yalnız 7 durumludur.

5 Düzenli ifadeler

1. **C** · kediler

   Parantez, birleşimi çoğul ekinden önce yapılmaya zorlar: önce kedi ya da köpek, sonra ler. İfade kediler ve köpekler dizgilerine uyar. Parantezsiz kedi|köpekler ise kedi ve köpekler dizgilerine uyardı.

2. **D** · gl

   + işareti en az 1 kez demektir: go+l, goo*l ile aynıdır ve en az bir o ister. gl dizgisinde hiç o yok. go*l ifadesi ise 0 kez de izin verdiği için gl'ye uyardı.

3. **D** · ccc

   Öncelik sırası yıldız, art arda yazma, birleşimdir: ifade (ab)|(c*) olarak okunur, yani ya tam ab ya da yalnız c'lerden oluşan dizgi. ccc ikincisine uyar. abab, abcc ve acb ikisine de uymaz; abcc için ab(c*) ya da (ab|c)* gibi başka bir ifade gerekirdi.

4. **D** · hahah

   hahah dizgisinde 2 ha bitince küme {q1, q4, q5}, son h okununca {q2, q3} olur; çıkış q5 kümede değildir, dizgi reddedilir. Boş dizgi atlama oku sayesinde, ha ve hahaha ise çıkışa varılarak kabul edilir.

5. **B** · A R* T

   Önce gelen okun ifadesi, sonra kendine dönen okun ifadesi yıldızlı, sonra çıkan okun ifadesi yazılır: A R* T. Kendine dönen ok yıldıza dönüşür, çünkü o döngüde istenildiği kadar dönülebilir, hiç dönülmeyebilir de.

6. **C** · Çünkü ilk parça istenildiği kadar uzun olabilir ve sonlu bir makine onu harfi harfine aklında tutamaz.

   "ve ve" gibi yinelenen kelimeyi bulmak için ilk kelime harfi harfine akılda tutulmalıdır. Kelime istenildiği kadar uzun olabildiğinden bunu hiçbir sonlu otomat yapamaz. Ek olarak sonlu otomatlarda harf okumadan geçen oklar da vardır ve birden çok kabul durumu da serbesttir.

7. **6**

   a ve b için 2'şer durumlu 2 parça: 4 durum. Birleşim yeni bir giriş ve yeni bir çıkış ekler: 4 + 2 = 6 durum.

8. **14**

   Rakamlar: 05 ile 9 rakam, toplam 11 rakam. Kuralda 3 yerde ' ?' var ve her biri en çok 1 boşluk getirir: 11 + 3 = 14. Örnek: 0512 345 67 89.

6 Pompalama önsavı ve kapalılık

1. **C** · 00011

   0*1* yalnız sırayı denetler: önce istediği kadar 0, sonra istediği kadar 1. 00011 bu sıraya uyar ama 3 sıfır ile 2 bir eşit değildir. Öteki üçünde 0 ve 1 sayıları eşittir.

2. **B** · 10

   Güvercin yuvası ilkesi: yuva sayısından (9 durum) bir fazla güvercin (10 parça) varsa iki güvercin aynı yuvaya düşer. 9 parça her biri ayrı durumda bitebilir; 10. parça kesin çakışma getirir.

3. **A** · 15

   Yeni makinenin her durumu bir çifttir: birinci makineden 3, ikinciden 5 seçenek, yani 3 × 5 = 15 çift. Toplam 8 olmaz, çünkü durumlar toplanmaz, çiftlenir. Başlangıçtan varılamayan çiftler atılırsa sayı 15'ten az olabilir.

4. **ret**

   Çiftler şöyle ilerler: dinlendi · koşu yok, koştu · koşu var, dinlendi · koşu var, koştu · koşu var, bozuk · koşu var. Son 2 gün üst üste koşulduğu için birinci makine bozuğa düşer; bozuk tuzak olduğundan plan reddedilir.

5. **0000111**

   Dizgi x y z olarak ayrılır: x boş, y = 0, z = 00111. Parçayı 2 kez yazınca x y y z = 0 + 0 + 00111 = 0000111 olur: 4 sıfır, 3 bir. Sayılar eşit olmadığı için dizgi kuraldan çıkar.

6. **15**

   Uzunluk 5 + 5 × 2 = 15 olur. 15 = 5 × 3 asal değildir. Parça kaç harf olursa olsun eklenen uzunluk 5'in katıdır, toplam da 5'in katı olur ve asal kalmaz.

7. **D** · Hiçbir şey kesinleşmez; dil düzenli de olabilir, olmayabilir de

   Önsav tek yönlüdür: oyunu kazanmak dilin düzenli olmadığını ispatlar, ama pompalanabilmek düzenli olmaya yetmez. Düzenli olduğunu göstermenin yolu dili tanıyan makineyi çizmektir.

8. **B** · 0*1*

   0*1* ile kesişimde hem eşit sayıda hem sıralı dizgiler kalır: önce n tane 0, sonra n tane 1. Bu dil düzenli değildir; eşit sayıdaki dil düzenli olsaydı kesişim de düzenli olurdu. (01)* ile kesişim yine (01)* olur, düzenlidir; 0* ve 1* ile kesişim ise yalnız boş dizgiyi bırakır, ikisi de bir şey ispatlamaz.

7 Bağlamdan bağımsız gramerler

1. **A** · 0011

   S ⇒ 0S1 ⇒ 00S11 ⇒ 0011: birinci kural 2 kez, sonra ikinci kural S'yi siler. Gramerin dizgileri hep birkaç 0, aynı sayıda 1 biçimindedir; 0101, 001 ve 10 bu biçime uymaz.

2. **B** · Her kuralın solunda tek bir değişken olması ve kuralın çevresine bakılmadan uygulanması

   Kuralın solunda tek değişken vardır ve değişkenin yerine ne konacağı çevresindeki parçalara (bağlama) sorulmaz: Özne yerine kedi koymak için Nesne'ye bakılmaz. Anlama bakmamak ise her gramerin ortak özelliğidir, bu adın nedeni değildir.

3. **24**

   Özne için 4, nesne için 3, yüklem için 2 seçenek vardır ve seçimler birbirinden bağımsızdır: 4 × 3 × 2 = 24 cümle. Gramer anlama bakmadığı için "Ali suyu yedi" gibi tuhaf cümleler de bunların arasındadır.

4. **C** · Öbek → Nesne Yüklem

   Öbek → Nesne Yüklem sağında tam 2 değişken vardır. Öteki üç kuralın sağında 3 değişken bulunur; normal biçim için bunların ikiye bölünmesi gerekir, tıpkı Cümle → Özne Nesne Yüklem'in Cümle → Özne Öbek ve Öbek → Nesne Yüklem diye bölünmesi gibi.

5. **5**

   S ⇒ 0S1 ⇒ 00S11 ⇒ 000S111 ⇒ 0000S1111 adımlarında birinci kural 4 kez uygulanır; sonra ikinci kural S'yi boş dizgiyle siler. Toplam 4 + 1 = 5 adım.

6. **27**

   İş n^3 ile orantılıdır. Sözcük sayısı 3 katına çıkınca $(3n)^3 = 27\,n^3$ olur: iş 27 katına çıkar. Sözcük sayısı 2 katına çıkınca iş 8 katına çıkıyordu.

7. **D** · 14

   Çarpma yalnız terimin içinde yapılabildiği için 3 × 4 = 12 bir terim olur ve 2 ona eklenir: 14. Önce toplamak isteyen parantez yazar: (2 + 3) × 4 = 20. 24 ve 9 ise gramerin vermediği okumalardır.

8. **B** · 0

   else içteki if'e bağlanır (girintiye bakılmaz). Dıştaki if'in koşulu yanlış olduğu için içteki if ve onun else'i hiç çalışmaz; klima 0 kalır. else dıştaki if'e ait olsaydı sonuç 2 olurdu; kod derleme hatası vermez.

8 Yığıtlı otomatlar

1. **D** · kalem

   kalem tersten melak olur, aynı değildir. kayak, anana ve radar tersten de aynı okunur.

2. **B** · 3

   2 sıfır yığına 2 sıfır koyar; dip işaretiyle birlikte yığın $, 0, 0 olur, yani 3 kat. 0000011111'de olduğu gibi: 5 sıfır dip işaretiyle 6 kat yapar, k sıfır k + 1 kat.

3. **C** · 00011

   00011'de 3 sıfır yığına girer, 2 bir yalnız 2 sıfır alır; dizgi biterken yığında bir 0 kalır, dip işareti görünmez ve ε geçişi yapılamaz: ret. Öteki üç dizgide sıfır ve bir sayıları eşittir, yığın tam boşalır.

4. **D** · al durumundayken bir 0 geliyor ve o durumda 0 okuyan ok yok

   İlk 0 yığına konur, ilk 1 makineyi al durumuna geçirir ve o 0'ı alır; yığın yeniden yalnız $ olur. Sonraki harf 0'dır, ama al durumundan 0 okuyan ok yoktur: ret. 0 ve 1 sayıları eşittir, sorun sıradadır; sırayı durumlar, sayıyı yığın denetler.

5. **19**

   n harfli dizgide ortası n harften biri ya da n + 1 aradan biri olabilir: n + (n + 1) = 2n + 1. n = 5 için 11 ve n = 19 için 39 çıkar; n = 9 için 2 × 9 + 1 = 19 tahmin vardır.

6. **C** · kab okunduktan sonra dizgi hem kabak hem kabakkabak diye bitebilir ve ortanın nerede olduğu önceden bilinemez

   k, a, b okunduğunda dizgi kabak'ta bitebilir ya da kabakkabak diye sürebilir; ikisi de palindromdur. b'den sonra yığın boşaltılırsa uzun dizginin ilk yarısı kaybolur, boşaltılmazsa kısa dizgi kaçırılır. Ortası işaretli dizgilerde (ka#ak) böyle bir seçim yoktur.

7. **14**

   aabbbb'de hamleler: 2 kez S → aSbb, 2 a eşleşmesi, 1 kez S → ε, 4 b eşleşmesi ve bitti'ye geçiş: 2 + 2 + 1 + 4 + 1 = 10. aaabbbbbb'de (3 a, 6 b) aynı sayım 3 + 3 + 1 + 6 + 1 = 14 olur.

8. **aaaabbbbccc**

   v ve x birlikte çoğalır: aaabbbccc dizgisinde 3. a bir kez daha ve 1. b bir kez daha yazılır. Sonuç 4 a, 4 b ve 3 c'dir: aaaabbbbccc. c'ler aynı kaldığı için üçlü eşitlik bozulur; böylece $a^n b^n c^n$'nin bağlamdan bağımsız olmadığı görülür.

9 Turing makineleri

1. **C** · Bir kareyi okur, yerine bir simge yazar ve bir kare sağa ya da sola geçer.

   Kafa her adımda tek bir kareyle uğraşır: okur, yazar, bir kare kayar; ne yazacağını ve yönü kural tablosu söyler. Yalnız soldan sağa gidip hiçbir şey yazmayan sonlu otomattır, yığının tepesiyle çalışan da yığıtlı otomat.

2. **1100**

   Makine sağa gidip boş kareyi görünce döner. Sondaki iki 1'i 0 yapar, sonra solundaki ilk 0'ı 1 yapar ve durur: 1100. Sağlama: 11 + 1 = 12 ve 1100 ikilik tabanda 12'dir.

3. **A** · 10 ⟨elde⟩ 10

   İlk 4 adım 4 basamağı geçer, 5. adımda makine boş kareyi görüp döner ve elde durumuna girer. 6. adımda kafanın altındaki son 1'i 0 yapar ve sola geçer: şerit 1010, kafa 3. karede, yani 10 ⟨elde⟩ 10. 1 ⟨elde⟩ 000 görünümü 7. adımın sonudur.

4. **8**

   3 adım 3 basamağı geçer; 4. adımda boş kareyi görüp döner. 5., 6. ve 7. adımlarda üç 1'i 0 yapar; 8. adımda sayının solundaki boş kareye 1 yazıp durur. Şeritte 1000 kalır (7 + 1 = 8).

5. **B** · Yazdığı simgelere şeritte istediği zaman geri dönüp yeniden okuyabilmesi.

   Yığından alınan simge kaybolur ve yığın yalnız tepesini gösterir. Şeride yazılan her şey ise yerinde durur ve kafa oraya dönünce yeniden okunur. Turing makinesinin de durum sayısı sonludur; şerit iki yana sonsuzdur.

6. **8**

   1. adım a'yı X, 2. adım b'yi Y, 3. adım c'yi Z yapar ve sola döner. 4. adım Y'yi geçer, 5. adım X'e varıp a ara durumuna döner. 6. adımda a ara işaretli bir Y görür ve denetle'ye geçer; 7. adımda Z'yi geçer, 8. adımda boş kareyi görüp kabul eder.

7. **D** · Doğrusal sınırlı otomat

   Sonlu otomat da yığıtlı otomat da bu dili tanıyamaz; a'ları sayıp hem b'lerle hem c'lerle eşleştirmek için yer yetmez. İşaretleme makinesi girdinin dışına yazmadığı için doğrusal sınırlı otomattır ve $a^n b^n c^n$, Chomsky hiyerarşisinin bağlama bağımlı basamağındadır. Turing makinesi de tanır ama sınırı daha geniştir.

8. **C** · Mekanik yöntem sezgiyle bilinen bir kavram; ispatlanacak matematiksel bir tanımı yok.

   Tez, "adım adım mekanik yöntemle hesaplanan her şeyi bir Turing makinesi hesaplar" der; ama mekanik yöntemin biçimsel bir tanımı olmadığı için bu cümle ispatlanamaz. Tez aslında algoritmaya tanım verir. Karşı örnek bulunmadı; Church'ün λ-hesabı, özyinelemeli fonksiyonlar ve Turing makinesi ise aynı fonksiyonları hesaplar.

10 Karar verilebilirlik

1. **B** · Her girdide durup evet ya da hayır diyen bir makine bulunması.

   Karar verilebilir soruda makine her girdide durur ve kesin bir evet ya da hayır der. Yalnız evette duran, hayırda durmayabilen makine ise soruyu sadece tanınabilir yapar.

2. **C** · 10 / 1 ve 011 / 01

   10 / 1 ve 011 / 01 kartlarının ikisinde de üst yazı alttakinden uzundur; ne kadar kart dizilirse dizilsin üst satır uzun kalır ve iki satır eşitlenemez. Diğer takımlarda çözüm var: 1 / 1 tek başına, 0 / 00 + 00 / 0 sırasıyla iki satır da 000, 11 / 1 + 1 / 11 sırasıyla iki satır da 111 olur.

3. **12**

   Üstte 11 + 0 = 110, altta 1 + 10 = 110: 1. kartı ve ardından 2. kartı dizince iki satır da 110 olur. Üç kartın hiçbiri (11 / 1, 0 / 10, 1 / 01) tek başına üst ve alt satırı eşitlemez, o yüzden en kısa çözüm 2 kartlıktır.

4. **39**

   1 kartlıkta 3, 2 kartlıkta 3 × 3 = 9, 3 kartlıkta 3 × 3 × 3 = 27 diziliş var: 3 + 9 + 27 = 39. Her kart eklenince diziliş sayısı 3 katına çıkar, bu yüzden çözümsüz takımda arama hiç bitmez.

5. **A** · Tanınabilir ama karar verilemez: duran programı çalıştırıp bekleyerek bir gün anlarız, sonsuza dönen için hiçbir makine kesin cevap veremez.

   Program duruyorsa çalıştırıp beklemek bir gün bunu gösterir; durma problemi bu yüzden tanınabilir. Sonsuza dönen programın döndüğü ise "yeterince bekledik" diyerek bilinemez: "hiç durmaz" cevabını hiçbir makine tanıyamaz.

6. **D** · dön, dön, dur, dön

   Ters, her programın kendi kodundaki sonucunun tersini yapar: dur'a dön, dön'e dur der. Köşegen dur, dur, dön, dur ise Ters'in satırı dön, dön, dur, dön olur. Ters her satırdan kendi köşegen hücresinde ayrıldığı için tabloda yeri olmayan bir satırdır; Ters de program olduğundan çelişki çıkar.

7. **B** · Program bir gün 0'a bölme yapar mı?

   Rice teoremi programın çalışınca ne yaptığıyla ilgili, bazı programlarda evet bazılarında hayır olan soruları kapsar; "bir gün 0'a böler mi" böyle bir davranış sorusudur. Diğer üçü kodun yazısıyla ilgilidir; kod okunarak kolayca cevaplanır.

8. **C** · Durma problemi çözülmüş olurdu; bu imkânsız olduğundan böyle bir araç yazılamaz.

   Eski programı sessizce çalıştırıp biterse "bitti" yazan yeni bir program kurulur; yeni program ekrana ancak eski program durursa bir şey yazar. Araç yeni programa sorulunca eskinin durup durmadığı öğrenilirdi: durma problemi çözülürdü. Çözülemediği ispatlı olduğundan araç da yazılamaz. Buna indirgeme denir.

11 P, NP ve NP-tamlık

1. **B** · Saniye problemi değil bilgisayarı anlatır; adım sayısı ise girdi uzadıkça nasıl büyüdüğünü gösterir.

   Aynı program eski bilgisayarda 1 dakika, yenisinde 1 saniye sürebilir; süre makineye bağlıdır. Adımı Turing makinesinin hamlesiyle tanımlayıp girdi uzadıkça nasıl büyüdüğüne bakarız. Adım sayısı makineye göre değişir ama polinom olarak kalır, kesinlikle aynı olmaz.

2. **100**

   5 harfte 36, 8 harfte 81, 10 harfte 121, 20 harfte 441 adım: sayılar hep $(n + 1)^2$. 9 harfte $(9 + 1)^2 = 100$ adım atar.

3. **C** · 2^n

   Polinom, uzunluğun sabit bir kuvvetiyle (kare, küp, n^{100}) büyüyen adım sayısıdır. 2^n ise her yeni harfle 2'ye katlanır: $n = 30$'da $2^{30} = 1.073.741.824$ olur, oysa 30^2 yalnız 900'dür.

4. **64**

   Her kapalı kare yerleşim sayısını 2'ye katlar: $2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^6 = 64$. 4 kapalı karede 16 yerleşim vardı.

5. **A** · Mayınların yeri; deterministik bir makine bunu polinom adımda denetler.

   Kanıt, kabul eden kolun yaptığı tahminlerin listesidir: burada mayınların yeri. Sayıları denetlemek tahtanın büyüklüğüyle orantılı, yani polinom adım sürer. Bütün yerleşimleri denemek ise üstel iştir, kısa kanıt değildir.

6. **2**

   Yalnız Ali gelirse (1) sağlanır, çünkü Ali geliyor. (2) sağlanmaz, çünkü Ali geliyor ama Cem gelmiyor. (3) sağlanır, çünkü Banu ile Cem birlikte gelmiyor. Sağlanan koşul sayısı 2'dir; bu yüzden bu seçim bulmacayı çözmez. Çözümler yalnız "Ali ve Cem" ile "yalnız Banu" seçimleridir.

7. **D** · P'deki her dil NP'dedir, çünkü deterministik makine hiç kola ayrılmayan deterministik olmayan makinedir.

   Polinom adımda karar veren deterministik makine, kola hiç ayrılmayan bir deterministik olmayan makine sayılır; bu yüzden $P \subseteq NP$. Tersinin, yani $P = NP$ olup olmadığı ise açık bir sorudur ve çözene 1 milyon dolarlık Clay ödülü verilir.

8. **B** · NP'deki her problem polinom adımda çözülürdü; yani $P = NP$ olurdu.

   NP'deki her problem polinom adımda bir NP-tam probleme çevrilebilir. Çevirme ve çözme birlikte polinom olduğundan NP'nin hepsi hızlı çözülür: $P = NP$. Durma problemi karar verilemeyen sorular sınıfındadır; NP-tamlık onu etkilemez.

Öğrendiğini pekiştir

Bu notlardaki bütün soruları ve daha fazlasını sitede, her cevabın hemen ardından çözümüyle birlikte çözebilirsin.

hekzon.com/dersler

HEKZON