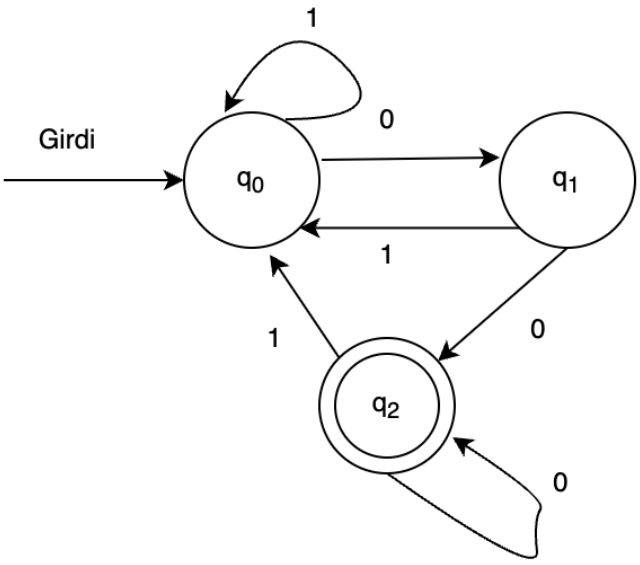
Bilgisayarlar Nasıl Çalışabilir?

MURAT ERŞEN ÜNAL

Hacettepe Üniversitesi Matematik Bölümü, Yüksek Lisans Öğrencisi
mersenunal@gmail.com

Evde, işte, okulda, otobüste... kısaca her yerde karşımıza çıkıyor bilgisayarlar. Bazen bir masanın üzerinde bazen içini görmediğimiz bir kutuda. Hepsine bilgisayar diyoruz ama aslında hepsi başka bir iş yapıyor. Hatta bir bilgisayarla farklı zamanlarda farklı işler yapıyoruz. Akşamları okuldan gelince oyun oynarken gece yatmadan ödevi yapmadığımızı fark edip sabaha kadar ödev yapıyoruz. Sonra da sabah derste uyuyoruz... Farklı işler yapması bakımından beynimize benziyor bilgisayarlar. Örneğin, beynimiz uyurken rüyalarla meşgulken, sabah sabah okula giderken metroda,otobüste veya sokakta gördüğümüz kimi insanları inceler hatta bazılarıyla ilgili hayaller bile kurar. O sırada reklam tabelalarında gördüğü bir yazıyı da okumayı ihmal etmez. Çok ilginçtir insan okuyabildiği bir kelimeyi görünce okur. Neyse konuyu dağıtmayalım, Aynı bilgisayarın farklı işlemleri yapabilmesi programlanabilir olmasının sonucudur. Günümüzde kullandığımız ortalama bir diz üstü bilgisayar saniyede yaklaşık olarak 10^9 adet işlem yapabilmektedir. Bunu en ama en kaba haliyle şöyle düşünebilirsiniz, bir bilgisayar bir saniyede 1 milyar tane toplama işlemi yapabilmektedir. Ancak bu bahsettiğimiz bilgisayarın nasıl çalıştığını tam olarak anlayabilmek için; otomat teoriden yarı iletken fiziğine, bilgisayar programlamadan dijital devre tasarımına kadar birçok alanda derinlemesine bilgi sahibi olmak gerekir. Bu yüzden başlığımız “Bilgisayarlar Nasıl Çalışır?" değil. Biz bu yazıda belli bir kurala göre davranan bir mekanizma ile basit bir problemi nasıl çözebiliriz onu göstereceğiz. Ve böylece bu yazıyı okuyan biri üzerinde, bilgisayarların nasıl çalışabileceğine dair bir önsezi yaratabiliriz. Birazdan anlatacaklarımız şimdiden gözünüzü korkutmasın herkesin anlayabileceği şeylerden söz edeceğiz.

Otomatalardan söz edeceğiz. Otomataların önemli olmasının bir sebebi, Otomataları basit dijital devre elemanlarıyla gerçekleyebiliyor olmamız. Yani verilen bir otomata gibi davranan bir dijital devre oluşturabilen mekanizmalarımız var. Bu mekanizmalara da belki ileride değiniriz. Öncelikle otomataları inceleyelim.



Şekil 1. 4 ile bölünebilme kuralını tanıyan otomata

Şekil 1'de bir otomatanın basit bir görseli görülmektedir. Bu otomata q_0 konumunda başlamaktadır ve girdi olarak 0 ve 1 harflerinden oluşan bir dizi kabul etmektedir. Bir otomata bulunduğu konum ve girdisinin incelediği elemanına göre bir sonraki konumunu güncellemektedir. Bu güncelleme yöntemi şekil üzerinde oklarla gösterilmiştir. Örneğin q_2 konumunda olan bir otomata; 1 harfi geldiğinde q_0 konumuna gidecek, 0 harfi geldiğinde ise yine q_2 konumunda kalacaktır. Girdinin tüm elemanları incelendiğinde otomatanın son olarak bulunduğu konum ise o girdiye verdiği cevap olarak adlandırılır. Örneğin şekildeki otomataya 01001 dizisini girdi olarak verdiğimizde otomata aşağıda adım adım anlatıldığı şekilde hareket edip cevap verecektir. Burada dikkat edilmelidir ki okumayı soldan sağa doğru yapıyoruz. Bu durum farklı kaynaklarda farklılıklar göstermektedir. Fakat genel durumu değiştiren herhangi bir yargı yoktur, sadece sıralamanın tersten yapılması olasıdır ve şaşırılacak bir durum değildir. Yukarıda belirtildiği üzere makul dillerin sıralama farklılığı ile oluşturulduğu düşünülebilir. Buradaki dil kavramı insanlar tarafından konuşulan dil olarak algılanmamalıdır. Nasıl insanlar bildikleri bir dili anlıyor ise otomatalarda çeşitli dilleri anlayacak şekilde inşa edilir.

Şimdi az önce bahsettiğimiz girdiyi alan Şekil 1'de tasfir edilmiş otomatanın çalışma adımlarını verelim.

1. Otomata q_0 konumunda başlayacaktır.
2. Girdinin ilk elemanı 0 harfi olduğu için otomata q_0 konumundan q_1 konumuna okla belirtildiği gibi gidecektir.
3. Otomata artık q_1 konumundadır.
4. Girdinin sonraki elemanı 1 harfi olduğundan otomata q_1 konumundan q_0 konumuna gidecektir.
5. Otomata artık q_0 konumundadır.
6. Girdinin sonraki elemanı 0 harfi olduğundan otomata q_0 konumundan q_1 konumuna gidecektir.
7. Otomata artık q_1 konumundadır.
8. Girdinin sonraki elemanı 0 harfi olduğundan otomata q_1 konumundan q_2 konumuna gidecektir.
9. Otomata artık q_2 konumundadır.
10. Girdinin sonraki elemanı 1 harfi olduğundan otomata q_2 konumundan q_0 konumuna gidecektir.
11. Otomata artık q_0 konumundadır.
12. Girdinin bütün elemanları incelendiğinden ve otomata q_0 konumunda olduğundan otomatanın girdiye verdiği cevap q_0 olacaktır.

Otomataları ilkel bilgisayarlar olarak düşünebiliriz. Günümüzde bilgisayarları, bilgisayarın tasarım aşamasında tanımlanmış prosedürleri kullanarak farklı türden problemleri çözmek için kullanabilmekteyiz. Örneğin optimizasyon problemlerini çözmek ve incelemek bilgisayar programlarının en önemli görevlerindendir ve optimizasyon problemlerinin önemini kavramak sezgisel olarak kolaydır. Biz bu yazıda karar problemleriyle ilgileneceğiz. Okur herhangi bir karar problemini şu şekilde düşünebilir: verilen bir karar probleminin cevabı ya evettir ya da hayırdır. Tabii bu noktada belirtmeliyiz ki bu problemler objektif problemler yani bir cevabı olan problemler olmalıdır. Bir cevabının olmasından kastımız, problemin cevabının sorulan kişiye, yere, zamana veya sorudan başka bir şeye bağlı olmamasıdır. Örneğin, "Bal güzel midir?" sorusu bu bağlamda gerçek bir karar problemi değildir. Çünkü sorunun cevabı kişiye ya da bazı durumlara göre farklılık gösterebileceğinden kesin bir cevabı yoktur. Ama "2 sayısı 3 sayısından büyük müdür?" sorusu gerçek bir karar problemidir. Çünkü soruda verilen her şeyin net bir tanımı olduğu gibi sorunun net ve kesin bir cevabı da vardır. Karar problemleri birçok fenomeni ifade etmek için kullanılabilir ve karar problemlerinin çözümü başka birçok problemin de çözümünün bulunmasında da yardımcı olur. Örneğin verilen bir optimizasyon probleminin optimum değerine "Bu optimum değer herhangi bir x değerinden büyük mü?" gibi karar problemleriyle de yaklaşabiliriz.

Yukarıda bir problemin, karar problemi olarak kabul edilebilmesi için gerçek bir cevabının olması gerektiğini belirtmiştik. Şimdi biraz daha temel bir durumdan bahsedelim: Bir problemin var olması için önce o problemi yazılı olarak ifade edebilmemiz gerekir. Bir problemi yazılı olarak ifade etmek ise basitçe uygun bir alfabe, genelde binary alfabe $\{0,1\}$, kullanarak ifade etmektir. Detaylar için okuyucular [1] kaynağını inceleyebilirler.

Şimdi otomatalara geri dönelim. Otomataya verdiğimiz girdiyi belirli bir karar probleminin yazılı hali, otomatanın verdiği cevabı ise otomatanın problem için bulduğu cevap olarak düşünebiliriz. Şekil 1'de görülebileceği gibi q_0 ve q_1 konumları tek çemberden oluşmaktayken q_2 konumu çift çemberden oluşmaktadır. Bunu şöyle düşünebiliriz; eğer otomata tek çemberden oluşan bir konumu cevap olarak veriyorsa soruya "Hayır" cevabını veriyor, çift çemberden oluşan bir konumu cevap olarak veriyorsa soruya "Evet" cevabını veriyordur. Bir otomata sonlu sayıda tek veya çift çemberli konumlardan oluşabilir. Buradaki önemli hususlardan biri, bir otomatanın her konumu için ve kabul ettiği alfabenin her harfi için gideceği yeni konumun tanımlanmış olmasıdır. Ayrıca kolayca anlaşılabileceği gibi otomata farklı uzunlukta girdileri de kabul edebilmektedir. Kısacası vereceği cevap, girdinin uzunluğuna değil girdinin temsil ettiği karar problemine bağlı olmasıdır.

Aslında Şekil 1'de görselleştirdiğimiz otomata verilen bir doğal sayının 4 sayısı ile bölünüp bölünemeyeceğinin cevabını vermektedir. Girdi olarak da ilgili doğal sayının 2'lik tabandaki yazılımını kabul etmektedir. Bu örneği vermemizin en önemli sebebi okura otomataların basit ve anlaşılır bir şekilde karar problemlerini çözmek için nasıl kullanılabileceğini aktarmaktır.

Artık otomataların resmi ve matematiksel tanımını verebiliriz.

Tanım 1.
Bir $D = (Q, \Sigma, \delta, q_0, F)$ 5'lisi,

- Q: Konumlar kümesi,
- q_0: Başlangıç konumu,
- Σ: Alfabe,
- $F \subseteq Q$: Evet cevaplı konumlar,
- δ: $Q \times \Sigma \to Q$, geçiş fonksiyonu,

olmak üzere kararlı bir otomata olarak adlandırılır.

Bu tanımı daha iyi anlayabilmek için Şekil 1'i inceleyelim. Şekildeki otomatanın: Konumlar kümesi $Q = \{q_0, q_1, q_2\}$ kümesi, alfabesi $\Sigma = \{0, 1\}$ kümesi, başlangıç konumu q_0, evet cevaplı konumları $F = \{q_2\}$ kümesidir. Geçiş fonksiyonu δ ise aşağıdaki tabloda belirtildiği biçimde tanımlıdır.

Anlık Konum \Giren harf	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

Görüldüğü üzere evet ya da hayır cevabı veren konumların varlığı ve herhangi bir konumdan girdinin ilgili konumundaki alfabe elemanına göre hangi konuma gideceği kesin olarak belirlidir. Bu durum en basit otomatadan en karmaşık otomataya değişmeyen bir gerçektir.

Şimdi Şekil 1'deki otomatayı daha detaylı inceleyelim. Örneğin bu otomataya 1100 girdisini verirsek, otomata, ilk konum başlangıç konumu q_0 olmak üzere sırasıyla: q_0, q_0, q_0, q_1, q_2 konumlarında bulunur ve sonuç olarak q_2 konumu bu girdiye verdiği cevap konumu olur. Veya otomataya 00 girdisini verirsek, otomata ilk konum başlangıç konumu olmak üzere sırasıyla : q_0, q_1, q_2 konumlarında olur. Bu gözlemler ile hangi konumda olursa olsun iki ardışık 0 okunduğu anda yeni konum q_2 olacaktır. Kısacası okurların da kolayca fark edebileceği üzere bu otomata, son iki harfi 0 olan bütün girdilere "Evet" cevabını verir. Diğer taraftan hangi konumda olursa olsun herngi bir konumda bulunan bu ototmata 01, 10 veya 11 girdi parçaları için q_2 dışındaki bir konumda bulunacaktır. Bir başka deyişle son iki harfi 0 olmayan tüm girdilere de "Evet" cevabını vermiyordur. 4'e bölünen bütün sayıların ikilik tabanda son iki hanesi 0 olduğunu biliyoruz. Dolayısıyla bu otomata 4'e bölünen sayıların 2'lik tabandaki yazılışlarına "Evet" cevabını veriyorken diğer durumlarda "Hayır" cevabını vermektedir. Yani bu basit otomota örneği bir sayının 4'e bölünüp bölünemeyeceğini belirleyebilmektedir.

Otomatalar günlük hayatta karşımıza çıkan bazı basit problemleri çözmek için kullanılmaktadır. Örneğin, kütüphanede su almak için kullandığımız otomat makineleri aslında birer otomatadır. Bilgisayarlar çalışma prensibi bakımından otomatalara göre biraz, ama çok fazla değil, daha karışık makinelerdir. Ama bu çalışma prensibini gerçek hayatta işe yarayan bir sisteme dönüştürmek kocaman bir endüstridir ve birçok farklı disiplini içinde barındırır. Biz bu yazımızda basit bir problemi matematiksel mekanizmalar kullanarak nasıl çözebileceğimizi otomatalar üzerinden anlattık. Bilgisayarlar aynı yolun ileri bir safhası olarak düşünülebilir.

■ Kaynaklar

[1] Michael R. Garey and David S. Johnson. Computers And Intractability A Guide to Theory of NP-Completeness. W. H. Freeman and Company, 1979