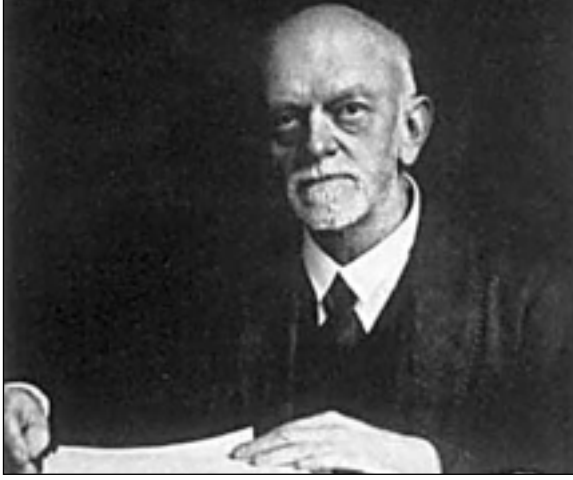


Gödel'in Eksiklik Teoremi



Cem Say* / say@boun.edu.tr

Yıl 1900. Aydınlanmadan beri bilime, dolayısıyla matematiğe olan inanç giderek artmaktadır. İnsanlık büyük bir iyimserlik içindedir. Fransız Henri Poincaré'yle birlikte zamanın en büyük iki matematikçisinden biri olan Alman David Hilbert doğru olan her şeyin kanıtlanabileceği ve ayrıca matematiğin çelişkisiz olduğu savlarını ortaya atar. Kanıtı yoktur ama öyle olmalı diye düşünür.



David Hilbert, 1932'de

Yıl 1931. David Hilbert 70'ine merdiven dayamıştır ve daha 12 yıl yaşayacaktır. Kurt Gödel ise 25 yaşında gencecik bir matematikçidir. Birinci sıranın yanlış olduğunu kanıtlar: Doğru olan her şey kanıtlanamaz! Buna **Eksiklik Teoremi** adı verilir.

Kanıtlanan her önerme doğrudur, bunu anlamak kolay. "Her doğru önerme kanıtlanır mı?" sorusu çok çok daha zor.

Gödel ayrıca matematiğin çelişkisiz olduğunun kanıtlanamayacağını da kanıtlar! (Matematiğin çelişkili olduğu - eğer matematik çelişkiliyse elbet! - kanıtlanabilir; bunun için örneğin $1 = 2$ eşitliğini kanıtlamak yeterlidir.)

Böylece iyimserliğe büyük bir (hatta iki!) darbe vurulmuştur. Matematik sanıldığı gibi her şeye muktedir değildir...

Eksiklik Teoremi matematik ve evren hakkındaki düşüncelerimizi derinden etkilemiştir.

Bu yazıda niyetim, Eksiklik Teoremini kendi anladığım şekilde ana hatlarıyla, herkesin anlayabileceğini umduğum bir biçimde anlatmak.

Program. Önce azıcık programlardan sözetmeliyiz. Kaygılanmayın, çoğu okurun bilgisayar programlamayı bilmediğini biliyorum. (Bence matematikle ilgilenen biri için büyük eksiklik ama neyse...) Bu yüzden bu yazıdaki tüm programları Türkçe yazacağım. Ama bunun karşılığında, gerektiğinde tüm yazdıklarımın biçimsel bir programlama diline çevrilebileceğine inanmanızı isteyeceğim.

Belli bir işi yapmak için tasarlanmış sonlu bir komut dizisine **program** denir. Örneğin,

1. A kâğıdına "Merhaba" dizisini yaz.

bir programdır.

Dizi, harf, noktalama işareti, rakam, bilimum matematiksel simgeler ve hatta "boşluk" gibi bu dergide görülebilen simgelerden canımızın istediği kadarını yanyana yazarak elde edebileceğimiz sonlu bir metin demektir.

Bir **programı çalıştırmak**, programın dediklerini sırayla yapmak demektir. Yani aslında "çalışan" program değil, onun dediklerini yapan her kimse oluyor, ama bu kelime dile yerleşmiş bir kere.

Sözgelimi, yukarıdaki programı çalıştırdığımızda yapmamız gereken, "Merhaba" kelimesini bu iş için önceden hazırladığımız A adını verdiğimiz bir kâğıda yazmaktır. Daha karmaşık programlarda birden fazla kâğıt kullanıldığından, programda kullanılan kâğıtlara A, B gibi adlar verilir.

"Bellek yetersiz" gibi uyarılara maruz kalmak için bu kâğıtların sonsuz boyda olduğunu, yani üstlerinde yazılı olan metnin öncesine veya sonrasına ne kadar çok yeni şey yazarsak yazalım kâğıtların dolmayacaklarını varsayacağız. Yalnız yazılan her şeyin uzunluğu her an sonlu olacak.

Azıcık daha karmaşık bir program görelim:

1. A kâğıdına "müdür" dizisini yaz.

* Boğaziçi Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyesi.

2. A'daki metni B kâğıdına kopyala.

3. A'daki metnin ardına iki kez B'deki metni, sonra da “?” dizisini yaz.

Bu yeni programın ilkinden iki farkı var. Birinci fark, ilki gibi tek değil, ardarda üç komuttan oluşması. Programı çalıştırdığımızda bu üç komutu bu sırayla ardarda uygulamamız gerekli. İkinci fark da, bu programda iki kâğıdın kullanılması. Kuşkusuz, bu programın yaptığı işin aynısını şu daha kısa programın çok daha dolambaçsız bir şekilde yapabileceğini farketmişsinizdir:

1. A kâğıdına “müdür müdür müdür?” dizisini yaz.

2. B kâğıdına “müdür” dizisini yaz.

Duran ve Durmayan Programlar. Şu programa bakalım:

1. A kâğıdını sil.

2. B kâğıdını sil.

3. A'daki diziyi B'deki dizinin sağına kopyala.

4. A kâğıdındaki diziyi sil, yerine sıralamada ondan bir sonra gelen diziyi yaz.

5. 3 numaralı komuta dön.

Bu programın ne yaptığını anlayabilmemiz için dizilerin kendi aralarında nasıl sıralandıklarını bilmemiz gerekli. İçinde hiçbir şey yazmayan diziy *boşdizi* denilir. Boşdizi, dizilerarası sıralamanın ilk elemanıdır. Diğer dizileri sıralayabilmemiz için, önce kullandığımız simgeler arasında bir sıralama tanımlamamız gerekir. Örneği kolaylaştırmak için dizilerimizde sadece Türkçe büyük harflerin kullanılabilmesini varsayalım ve bunların da alfabe sırasına göre sıralandıklarını (yani A'nın ilk, Z'nin de son simge olduğunu) düşünelim. Bu durumda farklı iki diziden hangisinin sıralamada daha önce gelmesi gerektiği, şu iki kural kullanılarak belirlenir:

i. Eğer dizilerin boyları (içlerindeki simge sayısı) farklıysa, kısa olan önce gelir.

ii. Aksi takdirde, iki diziyi soldan sağa doğru simge simge karşılaştırırız. Önce hangisinde öbüründeki simgeden önce gelen bir simge görürsek önce gelmesi gereken dizi odur.

Yani dizileri önce uzunluklarına göre sonra alfabetik sıralamalarına göre sıralıyoruz. Örneğin “MÜDÜR” dizisi “MOTORLAR” dizisinden önce, ama “MOTOR” dizisinden sonra gelir.

Şimdi programımızı inceleyebiliriz. Başlangıçta iki kâğıdı da sildiğimizden ikisinin de içinde boşdizi yazılı. Döngünün içinde sürekli B kâğıdındaki



Kurt Gödel

metni uzatıyoruz. Nasıl? Her dönüşte sağına sıradaki diziyi ekleyerek. Yani B kâğıdına (yine sadece Türkçe büyük harfleri kullandığımızı varsayarak) “ABCÇDEFGĞHHIIJKLMNOÖPRSŞTUÜVYZA-AABACAÇ...” diye başlayan ve hiç bitmeyecek bir metin yazmakla meşgulüz. (Sıralamamızda Z'den sonra AA gelir, daha sonra da sırayla AB, AC, AÇ, ..., AZ, BA, BB, ..., ZZ, AAA, ...)

İçinde döngü olan her program ille de sonsuza dek çalışacak diye bir zorunluluk yok. Sözelimi:

1. A kâğıdını sil.

2. B kâğıdını sil.

3. A'daki diziyi B'deki dizinin sağına kopyala.

4. A kâğıdını sil, yerine sıralamada ondan bir sonra gelen diziyi yaz.

5. A'daki dizide 1'den fazla simge varsa DUR.

6. 3 numaralı komuta dön.

DUR komutu, uygulandığında program çalışmasını oracıkta bitirir. Yani yukarıdaki programın çalışması sonlu sayıda adımdan sonra duracak, ve (yine sadece Türkçe büyük harfleri kullandığımızı varsayarsak) o anda A kâğıdında AA, B kâğıdında da ABCÇDEFGĞHHIIJKLMNOÖPRSŞTUÜVYZ metni yazılı olacaktır.

Gödel Teoremi. Gördüğümüz gibi, kimi programlar sonsuza dek çalışıyor, yani hiç durmuyor, kimileriye sonlu sayıda adımdan sonra duruyor. Böylece programlar kümesini “duranlar” ve “durma-

yanlar” olarak ikiye ayırabiliriz. Ayrıca “Falanca program sonlu zamanda durur” veya “Falanca program durmaz” gibi önermeler kurup bunların doğruluğunu ya da yanlışlığını kanıtlamaya uğraşabiliriz. Artık Gödel Teoremi’ni yazabilecek aşamaya geldik:

Teorem. *Programların durma ya da durmama özelliklerinin kanıtlanabildiği ve kanıtlanabilen tüm önermelerin doğru olduğu her biçimsel sistemde kanıtlanamayan doğru önermeler vardır.*

Üstelik, birazdan göreceğimiz üzere, kanıtlanamayan (ve teoremden varlığı söylenen) bu doğru önermelerden birini açıkça yazabiliriz de.

Kanıt Kavramı. Burada “biçimsel” sözcüğü önemli. Önermelerimizi fazlaca “elastik” olan Türkçe gibi “doğal” dillerle değil, her tümceye tek bir anlam yüklenbildiği biçimsel bir yazım şekli kullanarak ifade etmeliyiz, örneğin

sonsuz sayıda asal sayı vardır

demek yerine

$\forall q \exists p [p > q \wedge \forall x, y (x, y > 1 \rightarrow xy \neq p)]$

demek gibi. Aynı şekilde kanıtlarımız da biçimsel olmalı. Biçimsel kanıt ne demek peki?

MD’nin geçen sayısında (sayfa 97-98) anlatıldığı gibi, her kanıt, *belit* (aksiyom) adını verdiğimiz, doğruluğu tartışmasız kabul edilen kimi önermelerden başlayıp, yine doğruluğu tartışılmayan kimi basit *çıkarma kuralları* kullanılarak yeni önermelerin ardarda yazılmasından oluşur. Sistemimizde kullanmaya karar verdiğimiz belitlerin (tabii ki biçimsel yazımla) mevcut olduğunu, kanıtların bir satırından ötekine giderken kullanılacak çıkarma kurallarının da en baştan belirlenip sisteme yerleştirildiğini varsayıyoruz. Bu durumda \bar{O}_n önermesinin *biçimsel kanıtı* derken, her \bar{O}_i önermesinin ya belitlerimizden biri olduğu ya da kendisinden önceki önermelere dayanılarak çıkarma kurallarıyla oluşturulduğu $\bar{O}_1, \bar{O}_2, \dots, \bar{O}_n$ şeklindeki bir diziyi kastediyoruz. Epeyce uğraşmayı göze alırsak her düzgün Türkçe kanıtı bu biçimsel dile çevirebiliriz.

Verilen herhangi bir biçimsel kanıtın hatalı olup olmadığı, yani içindeki tüm önermelerin gerçekten de kendilerinden önceki önermelerden ve belitlerden çıkıp çıkmadığı otomatik olarak (yani bir bilgisayar programınca) kontrol edilebilir. İşte biçimsel kanıtların güzelliği budur. Türkçe kanıtlar için aynı işi yapmak çok ama çok daha zordur.



Gödel Teoremi’nin Kanıtı: Kanıtlanamaz doğru önermeler içerdiğini kanıtlayacağımız biçimsel sisteme S diyelim. S ’nin belitlerine ve çıkarım kurallarına dayanarak K kâğıdına yazdığımız herhangi bir dizinin S ’ye göre düzgün bir kanıt olup olmadığını kontrol eden programı yazıp bir kenarda hazır tutalım. Bu programı, aşağıda yazacağımız daha büyük bir programın bir parçası olarak önemli bir iş için kullanacağız.

Şimdi S sisteminde biçimsel kanıtı olmayan bir önerme yazacağız. Önce birinci komutu olmayan şu programa bakalım:

2. B kâğıdındaki metni A ’ya tırnak içine alarak kopyala.

3. “1. B kâğıdına ” dizisini, A ’nın içeriğini ve “ dizisini yaz. ” dizisini bu sırayla B ’deki metnin öncesine yaz.

4. B ’deki metnin öncesine ve sonrasına tırnak işaretleri koy, ardına da “ programı durmaz” dizisini yaz.

5. K kâğıdını sil.

6. Eğer K ’deki metin B ’deki önermenin S sistemine göre kanıtıysa DUR.

7. K ’deki diziyi sil, yerine sıralamada ondan bir sonra gelen diziyi yaz.

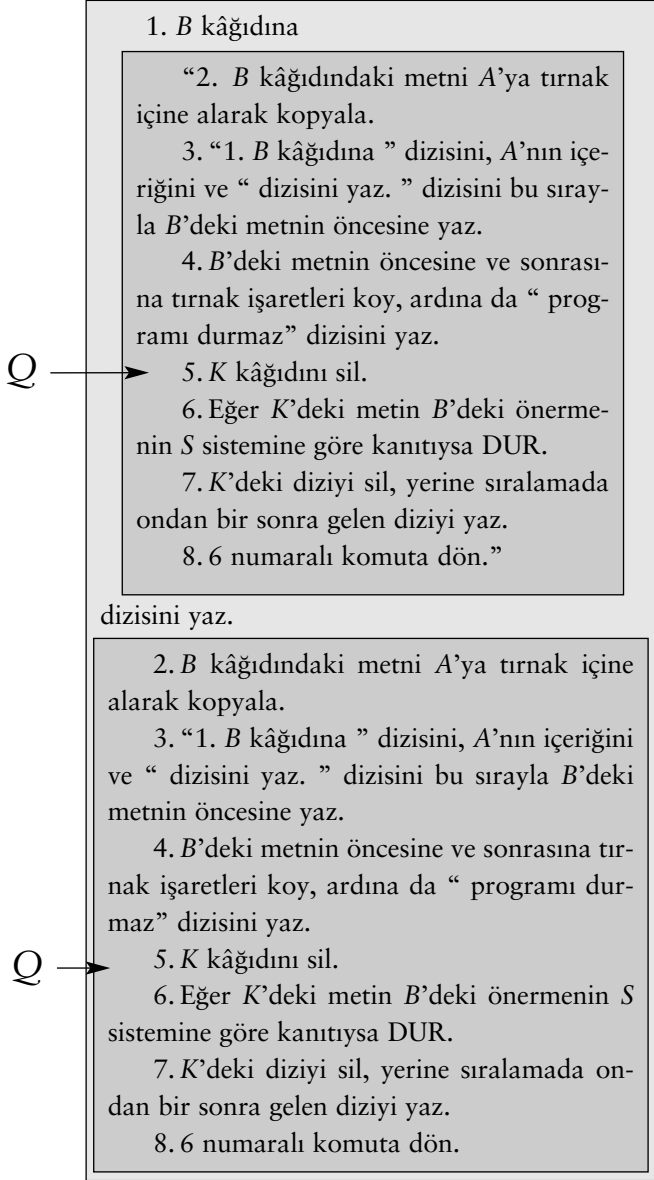
8. 6 numaralı komuta dön.

Bu programı Q simgesiyle kısaltalım.

Şimdi, bu Q programının başına

1. B kâğıdına Q dizisini yaz.

komutunu koyarak yeni bir program elde edelim. Bu programa P diyelim. Q uzun bir dizi olduğundan, birinci komutta Q dizisini oluşturan simgele-ri tırnak içine alıp yazalım ki bir karışıklık olmasın. İşte P 'nin açık hali:



Kanıtlanamayacağını iddia ettiğimiz önermemiz P programı durmaz

olsun. Bu önermenin S sisteminde kanıtlanamayacağını kanıtlayacağız. Burada da P derken P 'yi oluşturan simgelerin tırnak içine alınmış halini kastediyoruz.

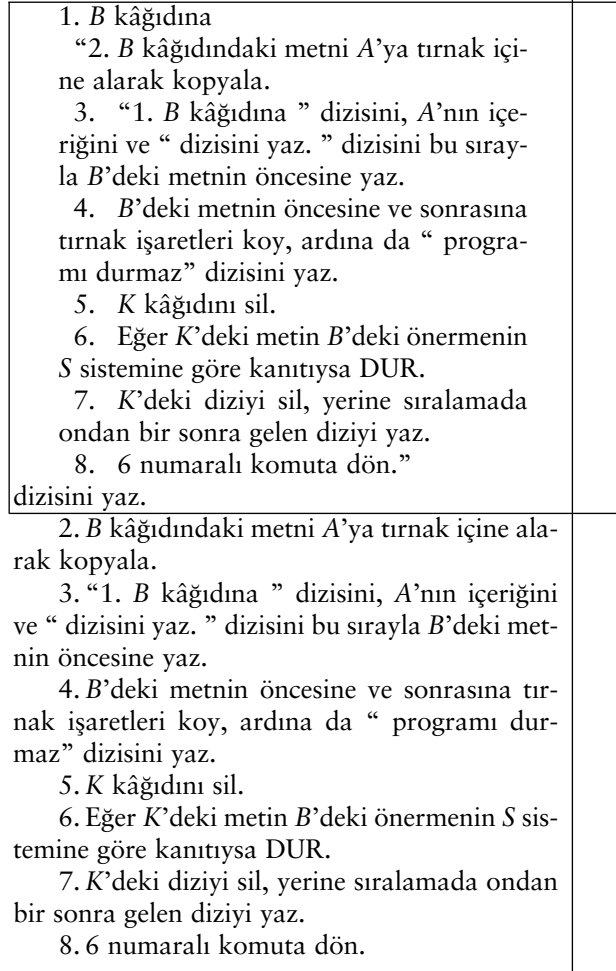
Komutları sırasıyla çalıştırıp programın davranışını anlamaya çalışalım. İlk komut, B kâğıdına Q dizisini yazmamızı söylüyor. Bunu yapalım. İşte B

kâğıdının şu anki hali:

2. B kâğıdındaki metni A 'ya tırnak içine alarak kopyala.
3. “1. B kâğıdına ” dizisini, A 'nın içeriğini ve “ dizisini yaz. ” dizisini bu sırayla B 'deki metnin öncesine yaz.
4. B 'deki metnin öncesine ve sonrasına tırnak işaretleri koy, ardına da “ programı durmaz” dizisini yaz.
5. K kâğıdını sil.
6. Eğer K 'deki metin B 'deki önermenin S sistemine göre kanıtıysa DUR.
7. K 'deki diziyi sil, yerine sıralamada ondan bir sonra gelen diziyi yaz.
8. 6 numaralı komuta dön.

Şimdi programımızın ikinci komutunu uyguladığımızda A kâğıdına aynı dizi tırnak içinde yazılmış olur.

Üçüncü komutu uyguladığımızda B 'de oluşan dizi P adını verdiğimiz programımızın ta kendisidir:



Eklenen kısım

Yani programımızın ilk üç komutunun işlevi, programın kendisinin bir kopyasını B kâğıdına yazmaktır. Demek ki sadece canlılar değil, programlar da üreyebilirmiş!

Şimdi geldik dördüncü komuta. Bu komut uygulandığında B 'de oluşan cümle tanıdık gelmeli... Bu cümle S sisteminde kanıtı olmadığını iddia ettiğimiz önermenin ta kendisi. Yani şu anda B 'de

P programı durmaz

yazıyor. Bundan sonra B değişmeyeceğinden, bundan sonra B 'de hep bu önerme yazacak.

Beşinci adım pek bir şey yapmıyor. Bu adıma ikinci bir kez gelmeyeceğiz. Bu adım, eğer K kâğıdına birileri haberimiz olmadan bir şeyler karalamışsa, bu karalamaların temizlenmesini sağlıyor.

Programın geri kalan 6, 7 ve 8'inci adımlarında, görüldüğü gibi bir döngüye giriyoruz. Bu döngünün her dönüşünde K kâğıdında yeni bir dizi bulunacak. Sıralamaya dikkat edildiği için her türden harf, boşluk, noktalama işareti, simge vs. karışımı eninde sonunda bu döngünün bir dönüşünde K 'de yazılı olacak ve elbette olabilecek tüm kanıtlar er ya da geç K 'de belirecekler. Yani eğer B 'de yazılı olan önermenin S sisteminde bir kanıtı varsa, sonlu sayıda dönüşten sonra o kanıt K 'de yazılı olacak, ve o dönüşte kontrol başarıyla sonuçlandırdığından program duracak. Yok eğer S 'de bu önermenin bir kanıtı yok ise o zaman bu döngüden asla çıkamayacağımızdan program hiç durmayacak.

Sözün kısası, bu P programı, sadece ve sadece kendi kendisinin durmayacağını söyleyen

P programı durmaz

önermesi S 'de kanıtlanabiliyorsa duracak. Şimdi düşünelim: Acaba programımız duracak mı durmayacak mı?

Eğer P programı durursa, o zaman S sisteminde *P programı durmaz* önermesi kanıtlanabiliyor demektir. Yani sistem yanlış bir şeyi kanıtlıyor. Bu da teoremimizin varsayımına aykırı olacaktır. Demek ki programımız durmayacak; o zaman da (bu programın durmayacağını söyleyen) önermemizin doğru olduğunu, ve fakat bunun S 'de kanıtlanamayacağı sonucunu çıkarmak zorundayız. Kanıt bitmiştir! \square

Gödel teoreminin güzelliği, ondan “kaçış” olmamasındadır. S , canınızın istediği herhangi bir belitler/çıkarımlar sistemi olabilir. Sözgelimi, çağımız-

daki matematikçilerin “matematik” dedikleri şey olabilir. Teoremimiz, eğer S (matematikçilerin umduğu gibi) çelişkisizse, o zaman S 'de kanıtlanamayan doğru bir önerme olduğunu göstermektedir.

“Peki ama,” diyebilirsiniz, “ S 'de kanıtlanamayan bu yeni önermenin doğru olduğunu artık bildiğime göre, onu S 'nin belitleri arasına eklesem, böylece ‘otomatik olarak’ kanıtlanabilir hale getirsem, böylece her doğru önermenin kanıtlanabildiği bir sistem elde etmiş olmaz mıyım?” Olmazsınız, çünkü belitleri değiştirdiğinizde artık eski S sistemini terkedip yeni bir sistem kullanmaya başlamış oluyorsunuz. Bu yeni sistemin adına \mathcal{S} diyelim. Açıkça görülebilir ki Gödel teoremindeki tekniğin aynısını kullanarak bu sefer de (yine çelişkisiz olduğunu varsaydığımız) \mathcal{S} sisteminde kanıtlanamayacak doğru bir önermeyi yazabiliriz. Bu sonsuza dek böyle gidebilir. Hem çelişkisiz, (yani hiçbir yanlış önermenin kanıtlanamadığı,) hem de her doğru önermenin kanıtlanabileceği bir sistem yoktur, işte o kadar.



Tabii bir de şöyle bir çözüm var: S 'nin belitleri, doğru olan tüm önermeler olsun. Bu durumda doğru olan her şey kanıtlanabilir. Ama bu yeni S sisteminin de şöyle bir sakıncası vardır. Bir dizinin bu yeni S sisteminde bir kanıt olup olmadığını anlayacak bir program yazamayız. Gödel'in kanıtı için geliştirdiğimiz P programının 6 numaralı komutunu hatırlayın: Burada bir dizinin S sisteminde bir kanıt olup olmadığını anlayabilen bir program kullanıyoruz. Eğer tüm doğru önermeler belitse, bu programın işini yapabilmesi için doğru önermeleri yanlış olanlardan ayırdedebilmesi gerekir. Varsayalım ki 6 numaralı komutu yerine getirebiliyoruz, yani S bu yeteneğe sahip. Şimdi, bu yeni S sistemi için P programı durur mu durmaz mı? Eğer “ P programı durmaz” önermesi yanlışsa, o zaman bu önerme kanıtlanamaz, kanıtlanamadığından da P programı durmaz, yani önerme doğrudur (kanıtlanan her önerme doğrudur); doğruysa da kanıtlanabilir ve kanıtlanabildiğinden dolayı P programı durur, yani önerme yanlıştır, değil mi? Bu çelişki bize 6 numaralı komutla ilgili varsayımımızın yanlış olması gerektiğini gösterir.

Sonuç. Doğru önermeleri yanlış önermelerden ayırdedebilen bir bilgisayar programı yoktur. \clubsuit