BÖLÜM 2

KANITLAR

Matematiksel Sistemler, Direkt Kanıt ve Dolaylı Kanıt

- **Aksiyomlar** daima doğru kabul edilen önermedirler.
- **Tanımlamalar** var olanlardan yeni kavramlar türetmek için kullanılırlar.
- **Teorem** doğruluğu gösterilmiş olan bir önermedir.
- **Lemma** çok ilginç olmayan ve ama bir başka teorem için kullanılan bir teoremdir.
- **Sonuç** bir başka teoremden kolayca elde edilebilen bir teoremdir.
- **Kanıt** bir teoremin doğruluğunun gösterilmesidir
- **Mantık** bir kanıtın analizi için kullanılan bir araçtır.

- Teoremler genellikle

  “Her $x_1,x_2,\ldots,x_n$ için
  $p(x_1,x_2,\ldots,x_n)$
  ise, bu durumda
  $q(x_1,x_2,\ldots,x_n)$
  ‘dir” biçimindedirler.

Kanıt Çeşitleri

- P önermesi doğru kabul edilip Q önermesinin doğruluğu gösterilirse buna **dolaysız kanıt** denir.
- Yani bu

  $P \rightarrow Q$

  olduğunun gösterilmesidir.

- P önermesini doğru ve Q önermesini yanlış kabul edip bir çelişki elde etmeye **çelişkiye vararak kanıt yöntemi** denir.
- Bir çelişki $r \wedge r'$ biçiminde bir önermedir. Çelişkiye vararak kanıt yöntemine **dolaylı kanıt** da denir.

Karşıt Örnek

- $\forall x P(x)$
  ifadesinin doğru olmadığını göstermek için, P(x) önermesini yanlış yapan tanım kümesinden bir x öğesi bulmalıyız.
- Böyle bir x öğesine **karşıt örnek** denir.

Hipotezlere Göre Kanıt

- Eğer orjinal hipotez farklı hipotezlere bölümlene-biliyorsa buna **hipotezlere göre kanıt** denir. Eğer p→q gösterilecekse ve p deyimi $p_1 \vee p_2 \vee \cdots \vee p_n$ deyimine denkse. Bu durumda

$$(p_1 \vee p_2 \vee \cdots \vee p_n) \rightarrow q$$

yerine

$$(p_1 \rightarrow q) \wedge (p_2 \rightarrow q) \wedge \cdots \wedge (p_n \rightarrow q)$$

olduğunu gösterilir.

Denkliğe Göre Kanıtı

- Bazı teoremler

  “p gerekli ve yeterli koşul q”

  şeklindedir.

  $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

  olduğundan

  “eğer p ise q”

  ve

  “eğer q ise p”

  ifadelerini göstermek yeterlidir.

Olmayana Ergi Yoluyla Kanıt

- Kabul edelim ki, $p \rightarrow q$ ifadesinin kanıtlamak yerine ona dengi olan karşıt tersini alarak

  $q' \rightarrow p'$

  ifadesinin kanıtlayabiliriz. Buna **olmayana ergi yoluyla kanıt** (contrapositive) denir.

Varlık Kanıtı

$\exists x P(x)$

‘ in kanıtlanmasına **varlık kanıtı** denir.

Kararlılık Kanıtları

- **Kararlılık** 1965 yılında J.A. Robinson tarafından önerilen bir kanıt tekniğidir. Bu tek bir kurala dayanmaktadır.
  "eğer her iki $p \vee q$ ve $p' \vee r$ önermeleri doğruysa, bu durumda $q \vee r$ önermesi de doğrudur"

- Bir kararlılık ile kanıtta hipotez ve sonuç bir **cümlecik** (clauses) olarak yazılır.
- Bir cümlecik bir değişken ya da değişkenin olumsuzunun "ya da" 'lar ile bağlanmasından oluşur.

- **Özel Hal:**
  Kararlılık kuralıyla

  “Eğer p$\vee$q ve p’ doğruysa, q ‘da doğrudur”

  “Eğer p ve p’$\vee$r doğruysa, r ‘de doğrudur”

Örnek 2.2.9:

"Eğer ben çalışırsam ya da zekiysem, dersimden geçerim. Eğer ben dersimden geçersem, bir sonraki dersimi alabilirim. O halde ben bir sonraki dersimi alamazsam, ben zeki değilim"

Burada önermelerimizi şöyle alabiliriz:

s= "ben çalışırım"

g= "ben zekiyim"

p= "dersimden geçerim"

n= "bir sonraki dersi alırım"

- **Örnek 2.2.10:**

  “Eğer ben çalışırsam ya da zekiysem, dersimden geçerim. Bir sonraki dersimi almaya gerek yoktur. Eğer dersimden geçersem, bir sonraki dersimi almalıyım. O halde çalışmama gerek yoktur.”

  s= “ben çalışırım”

  g= “ben zekiyim”

  p= “dersimden geçerim”

  n= “bir sonraki dersi alırım”

Matematiksel Tümevarım

- Kabul edelim ki, her bir n pozitif tamsayısı için bir S(n) önermesi var olsun.
  Ayrıca, yine kabul edelim ki,
  - S(1) doğru; (2.4)
  - Her i<n+1 için S(i) doğru olduğunda,

    S(n+1) doğru olsun. (2.5)

  Bu durumda her pozitif n tamsayısı için S(n) doğrudur.
- Burada (2.4) 'e temel adım ve (2.5) 'e de tümevarım adımı denir.

Örnek

```
FUNCTION SQ(A)
1.    C ← 0
2.    D ← 0
3.    WHILE (D ≠ A)
        a.    C ← C+A
        b.    D ← D+1
4.    RETURN(C)
END OF FUNCTION SQ
```

```
SUBROUTINE EXP(N,M;R)
1.    R ← 1
2.    WHILE (M>0)
       a.    R ← RxN
       b.    M ← M-I
3.    RETURN
END OF SUBROUTINE EXP
```

```
FUNCTION GCD(X,Y)
1. WHILE (X≠Y)
       a. IF (X>Y) THEN
              i. X ← X – Y
       b. ELSE
              i. Y ← Y – X
2. RETURN (X)
END OF FUNCTION GCD
```