

# Ayrık Matematik (Ayrık İşlemsel Yapılar) (BİL 2109)

Fırat İsmailođlu

Prolog - I



## Prolog (Programming Logic) (Lojik Programlama)

İlk iki haftada gördüğümüz Lojik, ve daha sonraki haftalarda göreceğimiz Rekürsiyon konularının bir uygulaması olarak Prolog programlama dilini öğreneceğiz.

Prolog, herhangi bir programlama bilgisi gerektirmeyen, son derece kolay bir sintaksa sahip olan lojik programlama dilidir. 1960'lı yıllardan beri kullanılmaktadır. En çok, yapay zeka alanında kullanılır.

Geleneksel programlama dillerinde (C++, Java), ifadeler bir biri ardına çalıştırılır. Prolog, programın tamamını okur ve bunu kaydeder (yani bilgiyi ezberler). Daha sonra kullanıcı programa sorular sorduğunda (bunlara query (sorgu) denir), Prolog cevap verir.

Prolog, ücretsiz bir programdır. Bir çok IDE'si vardır. Biz en çok tercih edilen ide'lerden biri SWI-Prolog'u kullanacağız. Şuradan indirilebilir:

[www.swi-prolog.org](http://www.swi-prolog.org)



## Prolog'da Veri Yapısı

Temel olarak Prolog'da iki tür yapı vardır: gerçekler (facts) ve kurallar (rules).

### Gerçekler

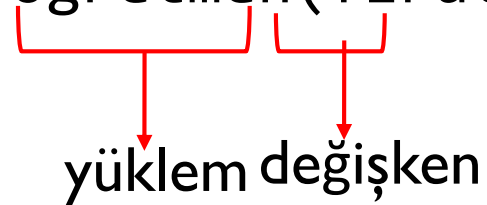
Prolog'da bir gerçek, bir yüklem (predicate) ve bu yüklem aldıkları değişkenler ile verilir.

ör.

```
ogrenci(ismail).          /* ismail öğrencidir*/
```

```
ogrenci(busra).          /* busra öğrencidir*/
```

```
ogretmen(firat).        /* firat öğretmendir*/
```

  
yüklem değişken

Yüklem birden fazla değişken alabilir:

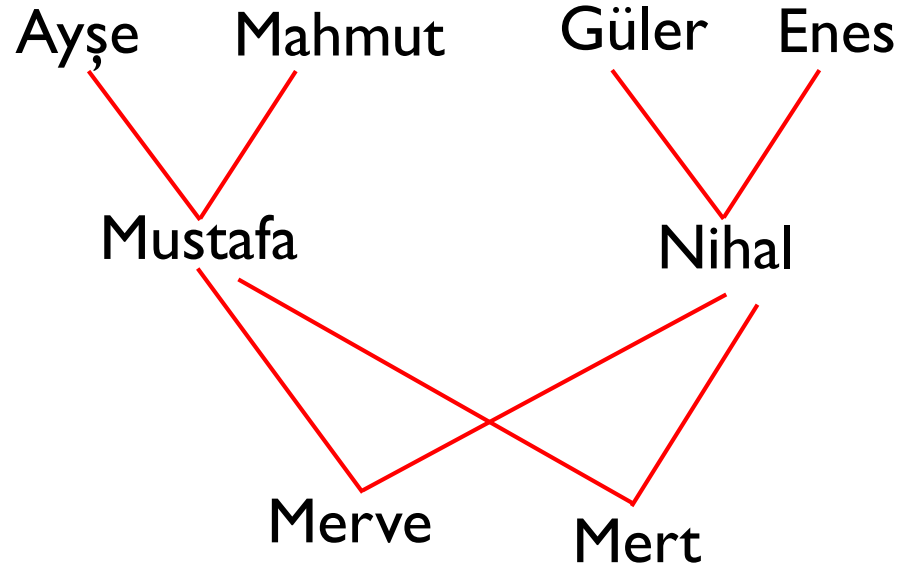
```
ogretir(firat, prolog).  /* firat, prolog öğretir*/
```

```
ogretir(firat, prolog, lojik). /* firat, prolog, lojik öğretir*/
```



Prolog, ağaç türünde veriyi kaydetmek için çok uygundur.

ör.



```
ebeveyn(mustafa, merve).  
ebeveyn(mustafa, mert).  
ebeveyn(nihal, merve).  
ebeveyn(nihal, mert).  
ebeveyn(ayse, mustafa).  
ebeveyn(mahmut, mustafa).  
ebeveyn(guler, nihal).  
ebeveyn(enes, nihal).
```

## Kurallar

Prolog'da kurallar (ise'li cümleler) :- ile verilir. Kurallar sağdan sola okunur:

buyukEbeveyn(X, Z) :- ebeveyn(X, Y), ebeveyn(Y, Z).

$\text{buyukEbeveyn}(X, Z) : -\text{ebeveyn}(X, Y), \text{ebeveyn}(Y, Z).$

(X'in Z'nin, büyük ebeveyni olabilmesi için X'in bir Y'nin ebeveyni olması, Y'nin de Z'nin ebeveyni olması gerekir).

## Programın Prolog'a Yüklenmesi

Kod blogunu bir text dosyasına yazıp, bu dosyayı sonu 'pl' ile bitecek şekilde kaydedebilirsiniz. **ör.** 'aileAgaci.pl'

Daha sonra bu dosyayı 3 farklı şekilde Prolog'a yükleyebilirsiniz.

1. [ ]. içine yazarak, prolog'da bir komut oluşturabilirsiniz:

**ör.** [aileAgaci].

Not: Bu şekilde birden fazla programı tek seferde prolog'a yükleyebilirsiniz.

**ör.** [aileAgaci, ilkPrologProgramim, ikinciPrologProgramim].

2. 'consult' komutunu kullanabilirsiniz.

**ör.** consult('aileAgaci.pl').



# Programın Prolog'a Yüklenmesi

## 3. SWI-Prolog'da, File tabı altındaki Consult... ile yükleyebilirsiniz.



Yada direkt terminal üzerinden kodunuzu yazabilirsiniz. Bunun için öncelikle `[user].` komutunu yazıp, altına istediğiniz gerçekleri ve kuralları yazın. Daha sonra `ctrl+d` ile kod yazimini bitirin.

```
?- [user].
|: ebeveyn(mustafa,merve).
|: ebeveyn(mustafa,mert).
|: buyukEbeveyn(X,Z):-ebevyn(X,Y), ebeveyn(Y,Z).
|:
% user://1 compiled 0.00 sec, 3 clauses
true.
```

## Sorgu (Query)

Gerçekleri ve kuralları yazarak kod yazımını bitirdiniz ardından yazdığınız kodu sonu pl uzantılı bir text dosyasına yazıp bu dosyayı Prolog'a yüklediniz.

Şimdi Prolog'a soru sorma zamanı!

```
ör. ? - buyukEbeveyn(ayse, merve).      /*ayse, merve'nin buyuk ebeveyni midir?*/  
      true.  
      ?- ebeveyn(guler, mustafa).      /*guler, mustafa'nın ebevyni midir?*/  
      false.  
      ?- ebeveyn(guler, nihal), ebeveyn(mahmut, mustafa).  
      true.
```

Yada örneğin Mert'in ebeveynleri kimlerdir diye sorabiliriz?

```
? - ebeveyn(X, mert).
```

```
X= mustafa;
```

```
X=nihal.
```



Bir önceki örnekte Mert'in ebeveynlerini sorgularken yazdığımızı sorgu komutundan sonra enter'a bastığımızda Prolog,  $X=mustafa$  yazıp bekler, sonraki ebeveyni görmek için ';' tuşuna basarız.

Şimdi de Nihal'in çocukları kimdir diye soralım. Bu, Nihal kimin ebeveynidir diye sormakla eşittir.

? - ebeveyn(nihal,X).

X=merve ;

X= mert.

Enes'in torunları kimlerdir?

? - buyukEbeveyn(enes,X).

X=merve ;

X= mert.





## listing komutu

listing Prolog'da yüklediğiniz bütün gerçek ve kuralları gösterir.

```
? - listing.
```

```
ör. etcil(kaplan).
```

```
etcil(insan).
```

```
otcul(insan).
```

```
otcul(inek).
```

```
hayvan(X):-etcil(X).
```

Kimler etcildir?

```
?- etcil(X).
```

```
X=kaplan;
```

```
X=insan.
```

Hem etcil hem otcul kimlerdir?

```
?-etcil(X),otcul(X).
```

```
X=insan.
```



Tüm etcillerin listesi:

```
?-listing(etci1).
```

```
etci1(kaplan).
```

```
etci1(insan).
```

```
true.
```

Insan hayvan mıdır?

```
?- hayvan(insan).
```

```
true.
```

**Veya**

Prolog'da veya noktali virguldur: ';'.

**ör.** X,Y'nin ebeveyni olabilmesi için, X,Y nin annesi olmalıdır veya X,Y'nin babasi olmalıdır.

```
ebeveyn(X,Y):- annesi(X,Y); babasi(X,Y).
```

**Not:** Yukarıdaki kural şu şekilde de verilebilirdi.

```
ebeveyn(X,Y):- annesi(X,Y).
```

```
ebeveyn(X,Y):- babasi(X,Y).
```

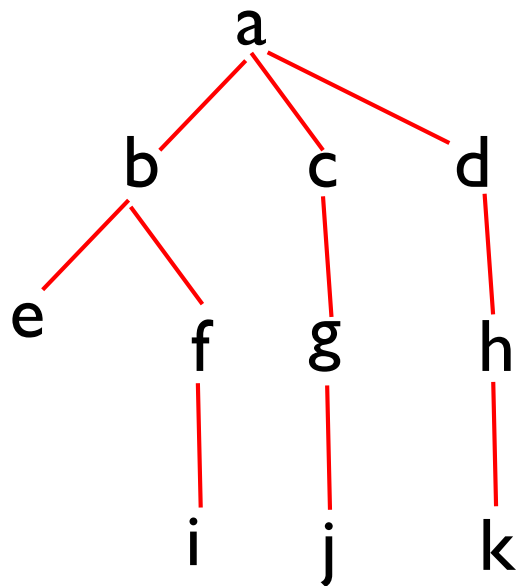


Bunun nedeni  $(p \Rightarrow q) \wedge (r \Rightarrow q) \equiv (p \vee r) \Rightarrow q$  olmasıdır.

Gerçekten:

$$\begin{aligned}(p \Rightarrow q) \wedge (r \Rightarrow q) &\equiv (\neg p \vee q) \wedge (\neg r \vee q) \\ &\equiv (\neg p \wedge \neg r) \vee q \\ &\equiv \neg(p \vee r) \vee q \\ &\equiv (p \vee r) \Rightarrow q\end{aligned}$$

ör.



ebe(a, b).

ebe(a, c).

ebe(a, d).

ebe(b, e).

ebe(b, f).

ebe(c, g).

ebe(d, h).

ebe(e, i).

ebe(g, j).

ebe(h, k).

buyukEbe(X, Y) :- ebe(Z, Y), ebe(X, Z).



?- g(U, V). /\*bütün büyük ebeveyn-torun ilişkilerini verir\*/  
?- g(M, e). /\*e'nin büyük ebeveynleri kimlerdir? \*/  
cocuk(X,Y):-ebe(Y,X). /\*X, Y'nin cocugu ise Y,X'in ebeveynidir\*/  
torun(X,Y):-buyukEbe(Y,X). /\*X, Y'nin torunu se Y,X'in büyük ebeveynidir  
kardes(X,Y):-ebe(Z,X), ebe(Z,Y), X/==Y.  
(Z,X'in ve Y'nin ebeveynidir, X, Y'den farklıdır).  
kuzen(X,Y):-ebe(Z,X), ebe(T,Y), kardes(Z,T).  
(X'in ebeveyni ile Y'nin ebeveyni kardestir).

Programa cinsiyet bilgilerini yükleyelim.

Diyelimki, a, b,c, f, j kadın olsun:

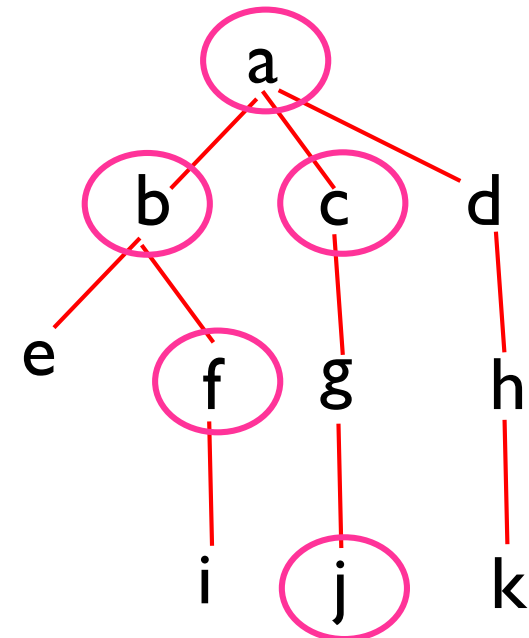
kadin(a).

kadin(b).

kadin(c).

kadin(f).

kadin(j).



Bu durumda digerleri erkek olur. Bunu acikca yazmamiza gerek yoktur. Kadın degilse erkektir:

$\text{erkek}(X) : -\text{not}(\text{kadin}(X))$ .

Bu sekilde anne ve babayi da tanımlayabiliriz.

$\text{anne}(X, Y) : -\text{kadin}(X), \text{ebe}(X, Y)$ .

$\text{baba}(X, Y) : -\text{baba}(X), \text{ebe}(X, Y)$ .

O halde teyze:

$\text{teyze}(X, Y) : - \text{kadin}(X), \text{anne}(Z, Y), \text{kardes}(X, Z)$ .

Amca:

$\text{amca}(X, Y) : - \text{erkek}(X), \text{baba}(Z, Y), \text{kardes}(X, Z)$ .

( $X, Y$ 'nin amcasi ise,  $X$  erkektir,  $Z, Y$ 'nin babasi ise  $X$  ve  $Z$  kardestir).

