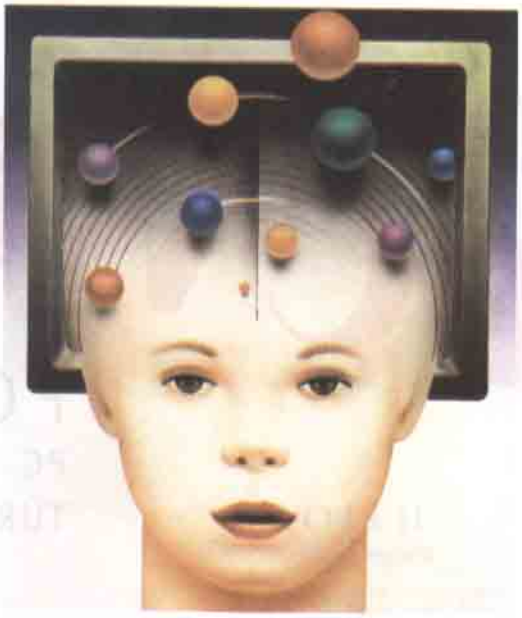
Bir Mantık Programlama Dili

PROLOG

(Bölüm IV)

Selçuk TARAL*



Geçen bölümü listelerle ilgili iki temel ilişkinin, **üye** ve **ekle**'nin programlarını vererek kapatmıştık. Bu bölümde liste (list) adı verilen bu zengin veri yapısı üzerine ilginç bazı yeni ilişkileri tanımlayan programlar vereceğiz. Ama önce üye ve ekle ilişkilerine geri dönüp bazı ek gözlemlerde bulunmak istiyoruz.

Geçen bölümün sonlarında okuyucuya şu soruyu yöneltmiştik: Prolog, '**?-üye(a,[b,c,d,e].**' sorusunu nasıl 'no' ile yanıtlar? Şimdi bu sorunun yanıtını verelim: Verilen listenin başındaki eleman 'a' olmadığı için gerçek sağlanamayacak, bu kez programın kuralı çalışacak ve aynı harf listenin kuyruğunda aranacaktır. Ancak, programın gerçeği liste boşalana kadar sağlanmayacaktır. Daha önce belirtmiştik; Boş liste [XIY] şeklinde ikiye bölünerek yazılamaz. Bu nedenle '?-üye(a,[ ]).' sorusu ne programın gerçeğiyle ne de kuralının başıyla birleşebilir. Sonuçta hedef başarısızlıkla sonuçlanır. Prolog bu durumu bize, 'a sabiti [b,c,d,e] listesinin bir elemanı değildir' anlamında, 'no' yanıtıyla bildirecektir.

Geçen bölümde üye ilişkisini tanımlayan programın, bir elemanın bir listenin içinde olup olmadığını belirlemek için nasıl kullanıldığını göstermiştik. Örneğin, '**?- üye(3,[1,2,3].**' sorusunun, '3 sayısı [1,2,3] listesinin bir elemanı mıdır?' anlamına geldiğini biliyorsunuz. Aynı ilişki bir listenin elemanlarını bulmak için de kullanılabilir. Aynı programa şimdi, '[a,b,c] listesinin elemanları nelerdir' anlamında, '**?-üye(X,[a,b,c]).**' sorusunu yönelterek, bu listenin elemanlarını sırayla yazdırabilirsiniz. Tahmin edebileceğiniz gibi, ilk yanıt sorunun gerçek ile birleşmesi sonucu, X = a olacaktır. Prolog'a, yeniden aramasını söylediğinizde, sırasıyla X = b ve X = c yanıtlarını alacaksınız (Üye ilişkisinin bir üçüncü kullanım şeklini düşünebiliyor musunuz?).

Geçen bölümde **ekle** ilişkisinin iki listeyi arka arkaya ekleyerek yeni bir liste elde ettiğini belirtmiştik. Şimdi bu programın yordamsal okunuşuna bir göz atalım. Programın gerçeği, boş listeye herhangi bir başka listenin eklenmesiyle yine aynı listenin elde edileceğini deklare eder. Programın kuralının yordamsal okunuşu ise şöyledir: *Başı X, kuyruğu ise Xs ile gösterilen bir listeye, Ys ile gösterilen bir listeyi eklemek için, önce X'i oluşturulan yeni listenin başı yapın, daha sonra özyinelemeli olarak Xs ve Ys ile aynı işleme, birinci liste boşalana değin, devam edin. Birinci liste boşaldığı zaman, programın gerçeğini kullanarak, Ys'yi yeni oluşturulan listenin sonuna kuyruk olarak ekleyin.* Böylece ikinci listeyi birinci listenin arkasına eklemiş olursunuz.

Program, '**?- ekle([1,2],[3,4],L).**' sorusuna, **L = [1,2,3,4]** yanıtını aşağıdaki adımlardan sonra verir:

1. **ekle([1|[2]],[3,4],[1|Zs]) ← ekle([2],[3,4],Zs).**
   X = 1, Xs = [2], Ys = [3,4], L = [1|Zs].
2. **ekle([2|[ ]],[3,4],[2|Zs1]) ←ekle ([ ],[3,4],Zs1).**
   X = 2, Xs = [ ], Ys = [3,4], Zs = [2|Zs1].
3. **ekle([ ],[3,4],[3,4]).**
   Doğru. Zs1 = [3,4].

İlk iki adımda, birinci liste boş olmadığı için, gerçekle birleşme olmaz ve programın kuralı çalışır; Birinci listenin ilk elemanı olan 1 sayısı, kural başının soruyla bütünleşmesi sonucu, yeni oluşturulan ve L değişkeni ile gösterilen listenin ilk elemanı olarak taşınır. Taşıma işleminin kuralın başında yapılmasını önce yadırgayabilirsiniz, ancak alıştıktan sonra bu tekniğin Prolog programlarının estetiğini artırdığını göreceksiniz.

Birinci adım sonunda oluşturulan liste, başı 1 sayısı, kuyruğu ise henüz belli olmayan ve Zs değişkeniyle gösterilen bir listedir; **L = [1|Zs].** Kuralın özyinelemeli uygulanmasıyla bu kez birinci listenin ikinci elemanı olan 2 sayısı yeni listenin kuyruğunun birinci elemanı olarak taşınır; **Zs = [2|Zs1].** Birinci listenin boşalmasıyla artık programın gerçeğiyle birleşme sağlanır ve Liste'nin henüz belirlenmemiş olan kuyruğu, Zs1, [3,4] listesine bağlanır. Sonuçta, Zs **[2|[3,4]]** listesine, esas aradığımız L ise **[1|[2|[3,4]]]** listesine bağlanmış olur. Sonuç bize **L = [1,2,3,4]** olarak verilir.

Aynı programa, '**?- ekle(Y,Z,[a,b,c,d]).**' sorusunu yöneltmek, tahmin edebileceğiniz gibi, 'hangi listelerin eklenmesinden [a,b,c,d] listesi elde edilir' an-

* *TÜBİTAK, Bilgi İşlem Daire Başkanı.*

lamına gelir ve Prolog tüm olası çözümleri aşağıdaki sırada verir:

Y = [ ]	**Z = [a,b,c,d]**
Y = [a]	**Z = [b,c,d]**
Y = [a,b]	**Z = [c,d]**
Y = [a,b,c]	**Z = [d]**
Y = [a,b,c,d]	**Z = [ ]**

Bu bölüme listeler üzerine yeni ilişkiler tanımlayan örnek programlarla devam edelim; Şekil 4'de listeler üzerine iki yeni ilişkinin programlarını verdik. Birincisi, bir liste, bir eleman ve bu elemanın listeden çıkartılması sonucu ortaya çıkan liste arasındaki ilişkiyi tanımlar. Bu eleman birden fazla kere listede yer alıyorsa, hepsi listeden çıkarılır. Örneğin, programa, **'?- çıkart([1,3,1,4],1,Kalan).'** sorusunu yöneltirsek, yanıt **Kalan = [3,4]** olur.

```
çıkart([X|Xs],X,Ys) :-
            çıkart(Xs,X,Ys).
çıkart([X|Xs],Z,[X|Ys]) :-
            X ≠ Z, çıkart(Xs,Z,Ys).
çıkart([ ],X,[ ]).


seç(X,[X|Xs],Xs).
seç(X,[Y|Ys],[Y|Zs]) :-
            seç(X,Ys,Zs).
```

Şekil 4

İkinci program ise biraz farklı olarak verilen elemanı listeden ilk gördüğü yerden sadece bir kere çıkarır ve geri kalan listeyi verir. Örneğin, **'?- seç(a,[b,a,c,a],X).'** sorusuna program, **X = [b,c,a]** yanıtını verir. Aynı programın çok daha sık rastlanan kullanımı ise, bir listeden bir elemanı seçmek ve geri kalanların listesini vermek şeklinde olur. Bu amaçla programa **'?- seç(X,[1,2,3],Xs).'** hedefi verildiğinde, ilk yanıt **X = 1, Xs = [2,3]** olur. Programdan aynı hedefe yeni çözümler bulması istenilirse, önce X = 2, Xs = [1,3], daha sonra X = 3, Xs = [1,2] yanıtlarını alırız. Programların çalışmalarının daha detaylı olarak incelenmesini okuyucuya bırakıyoruz.

```
a.    tersi([ ],[ ]).
      tersi([X|Xs],Zs) :-
                  tersi(Xs,Ys),
                  ekle(Ys,[X],Zs).

b.    tersi(Xs,Ys) :- tersi(Xs,[ ],Ys).

      tersi([X|Xs],Akü,Ys) :-
                  tersi(Xs,[X|Akü],Ys).
      tersi([ ],Ys,Ys).
```

Şekil 5

Prolog'ta aynı ilişkiyi tanımlamak için farklı programlama teknikleri kullanabilirsiniz. Örneğin, Şekil 5'te verilen iki program aslında iki liste arasındaki aynı ilişkiyi, *birinin diğerinin tersi olmasını,* tanımlarlar.

Programların her ikisi de, biribirlerinden çok farklı görünmekle birlikte, örneğin **'?- tersi([a,b,c],L).'** sorusuna **L = [c,b,a]** yanıtını verirler. Birinci programın okunuşu şöyledir; *Boş listenin tersi yine boş listedir. Başı X elemanı, kuyruğu Xs olan bir listenin tersi Xs'in tersi olan Ys listesine X elemanını eklemekle elde edilir.*

İkinci program ise yeni bir teknik kullanarak bir listeyi tersine çevirir; Birinci kural argüman sayısını bir artırarak, birinci listenin elemanlarını toplayacağı yeni bir listeyi ilişkiye ekler.Bu ek listeye, içine birinci listenin elemanlarını dolduracağımız için, **Akü** adını verdik. Dikkat ederseniz, Akü listesi önce boş liste olarak programın kuralını çalıştırır ve kuralın ilk özyinelemeli işleyişiyle, birinci listenin ilk elemanı Akü'ye taşınır. Kuralın ikinci çalışmasında ise, birinci listenin kuyruğunun ilk elemanı, yani listenin ikinci elemanı, Akü'nün başına taşınır. Bu işlem birinci liste boşalana kadar devam eder. Birinci liste boşaldığında, Akü listesine birinci listenin elemanları *ters sırada* dizilmiş olur (Neden ters sırada?). Şimdi artık, programın gerçeği Akü'yü Ys ile birbirine bağlar ve sonuç olarak Akü listesiyle aynı içerikte olan Ys listesini verir.

Gelecek bölümde daha uzun programlar yazmaya ve bazı tanınmış bulmacaları Prolog kullanarak çözmeye başlayacağız. Bu bölümü, bir alıştırma ile kapatalım. Aşağıda verilen program bir listeyle ilgili hangi ilişkiyi tanımlar?

```
son(X,[X]).
son(X,[X|Xs]) :- son(X,Xs).
```

DÜZELTMELER

Geçen bölümde birinci sayfanın son paragrafında tek elemanı 1 olan listenin ilk yazılışı, .(1,()) değil, .(1,[ ]) şeklinde olacaktı. Ayrıca, programların bulunduğu mavi çerçevenin altında şekil 3 yazısı unutulmuş. Düzeltir özür dileriz.

(Devam edecek.)

SİZ OLSAYDINIZ

(Satranç Dünyası'nın çözümleri.)

Çözüm I: 1..Kc1 2.Şf2 (2.Ff1 Kc7! 3.Vg6 Ve7 4.Vh6 Vh7) 2..Kc7! 3.Vg6 Kf8 4.Vh6 Şg8 kazanır (Hickl-Seirawan, Zagrep 1987).

Çözüm II : 1.Ag6! Af4 (1..Vb6 2.Af8 Af4 3.Afe6! Ag2 4.Ag5 Ae1 5.Vh5 beyaz üstün) 2.Af5! Ae2 3.Ke2 Vd1 4.Kd1 hg6 5.Ae7 Şh7 6.Kd3 Şh6 7.f4 Şh5 8.c3! Ka7 9.Fd1 Ff3 10.Kf3 Şg4 11.Ke5 Kd7 12.Kg5 Şh4 13.Kh3 mat (Waldmann-Dallas, Berlin 1987).

Çözüm III : 1..Fg5! 2.Fg5 Afe4 3.Fe7 Af2 4.Ff8? (4.Fc5 Şh1 5.Fd3! Ag3 6.hg4 Kh6 7.Ff8 Şf8 daha iyiydi.) 4..Şf8 5.Fg2 hg3 6.Kdg1 Ah1 7.Fh1 Kg6 8.Kg6 fg6 kazanır (Kubikova-Ivanka, Balatonföldvar 1987).