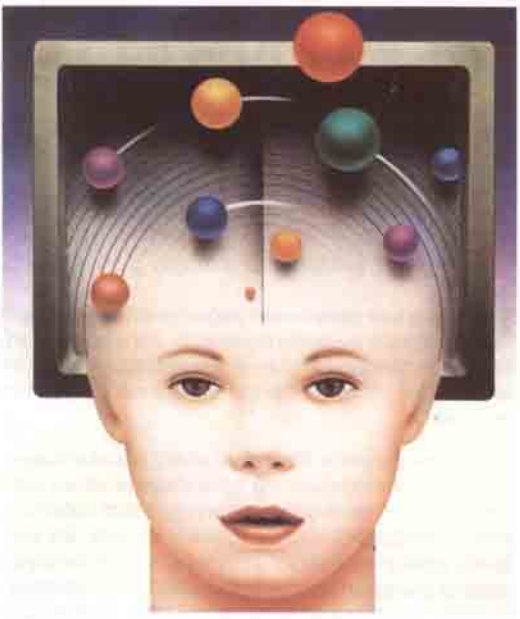
Bir Mantık Programlama Dili

PROLOG

(Bölüm I)

Selçuk TARAL*



Ülkemizde bilgisayar dilleriyle ilgilenen çoğu kimse Prolog adında bir bilgisayar dilinin olduğunu duymamış olabilir. Bunun nedeni, Prolog'un ülkemizde pek fazla ilgi uyandırmayan yapay zekâ uygulamalarında kullanılan beşinci nesil dillerden olmasıdır.

Bu makale dizisinde Prolog dilinin temel yapısını, kullandığı teknikleri ve kontrol akış mekanizmasını örneklerle anlatmaya çalışacağız. Amacımız, her dilde doğal olarak bulunan özel terim ve söz dizimi kurallarının içine fazla girmeden, bu ilginç programlama dilinin temel yapısı üzerinde genel bir fikir oluşturmaktır. Bir prolog yorumlayıcısı bulabilen okuyucuların verilen örnekleri denemelerini salık veririz.

1970'li yılların başlarında Fransa'nın Marseille-Aux Üniversitesi'nde Calmerauer ve çalışma grubu tarafından icat edilen Prolog, adını Fransızcadan alır: 'Programation et Logique'. Mantığın doğrudan doğruya bir bilgisayar dili olarak kullanılabilmesini sağlamak amacıyla yapılan çalışmalar oldukça yeni sayılabilir. 1980 yılına değin dünyada 'mantık programlaması' diye adlandırılan programlama teori ve teknikleriyle uğrayanların sayısı 100 kişiyi bile geçmezken, 1981 yılında Japonların beşinci nesil bilgisayar projesini açıklamalarıyla konuya ilgi büyük ölçüde arttı. Bu projede, gelecek kuşak bilgisayarların geliştirilmesinde mantık programlamasının oynayacağı önemli rol vurgulanıyordu. Bundan sonra Prolog dili hızlı bir şekilde gelişti ve olgunlaştı. Bugün artık her tip bilgisayarda kullanılabilen Prolog uygulamaları bulmak mümkündür. Prolog üzerine çeşitli amaç ve seviyelerde birçok kitap yayınlandı ve dilin bir standardı oluştu: Edinburgh Prolog ailesi.

Bilgisayarın belirli bir problemi çözebilmesi için kendisine problemle ve çözüm yoluyla ilgili bilgi verilmesi gerekir. Programlama dilleri aracılığıyla insan bilgisayarla iletişim kurabilir. Basic, Fortran, Cobol gibi klasik programlama dilleri bu iletişimi kolaylaştıran üst düzey diller olmalarına karşılık, temel yapıları itibariyle bilgisayarın doğrudan 'anladığı' dilin, yani makine dilinin çok geliştirilmiş şekilleridir. Bu nedenle, klasik programlama dillerinin yapısı doğal konuşma dillerinden (Türkçe, İngilizce vb.) ve mantıksal düşünme yöntemlerinden çok uzaktır. Örneğin, klasik programlama dillerinin sözdizimlerinde temel yapı elemanlarından olan GOTO, FOR-TO, REPEAT-UNTIL vb. ifadeler, bir problemi çözerken normal olarak kullandığımız mantıksal düşünce yöntemlerini hemen hiç yansıtmazlar.

* *TÜBİTAK, Bilgi İşlem Daire Başkanı.*

Klasik programlama dillerinin yapılarındaki bu yapaylığın, yazılım teknolojilerinde donanıma göre çok daha yavaş bir ilerlemenin görülmesine (yazılım krizi) neden olduğunu söyleyebiliriz. Diğer taraftan, kullanılan programlama dilinin bir problemin kavramsal çözümlenmesi aşamasında kullanıcıya yardımcı olması, yazılım geliştirme uğraşını kolaylaştırır ve yaratıcı bir faaliyet haline sokar. Böylece hızla prototip üretmek ve ortaya çıkan modülü test etmek mümkün olur. Prolog, mantıksal ve sembolik düşünmeye uygun yapısıyla, problemin tanımlanması ve çözümü için gerekli yöntemlerin geliştirilmesi aşamalarında bize yardımcı olan bir araçtır.

Burada Prolog'un mantık üzerine kurulu ve klasik programlama dillerinden tamamen farklı olan bu yapısının neden olduğu ilginç bir gözlemden söz etmeden geçemeyeceğiz. On yaşında çocuklar Prolog'u öğrenmesi kolay bir dil bulurken, deneyimli programcılar bu dili şaşırtıcı ve kafa karıştırıcı bulmuşlar! Şimdi Prolog'un bu mantıksal yapısına bir göz atalım.

Biraz mantık okumuş herkesin bileceği gibi, 'Sokrat bir insandır' ve 'Tüm insanlar ölümlüdür' cümlelerinden, 'Sokrat ölümlüdür' sonucuna varırız. Şimdi bu basit mantık probleminin bir Prolog programı olarak nasıl ifade edilebileceğini görelim. Problem önce dilin temel iki öğesi olan, 'gerçekler' ve 'kurallar' aracılığıyla tanımlanır. Gerçekler, matematiksel aksiyomlar gibi, bir veya daha fazla nesne arasında bulunan bir ilişkiyi veya bir nesneyle ilgili bir özelliği, "deklare etmek" için yazılan Prolog tümceleridir. Örneğin, 'Sokrat bir insandır' tümcesini bir Prolog gerçeği olarak şöyle yazabiliriz:

```
insan (sokrat).
```

'Kurallar' ise, içindeki koşulların doğru olmaları halinde doğru olan Prolog ifadeleridir. Örneğin, 'Tüm

insanlar ölümlüdür' veya aynı anlamdaki 'X bir insan ise, x ölümlüdür' tümcesini bir kural olarak şöyle yazabiliriz.

ölümlü (X) : — insan (X)

Bir özelliği veya bir ilişkiyi gösteren isimlere Prolog terminolojisinde yüklem (predicate) adı verilir. Yukarıda ölümlü ve insan yüklemleri parantez içindeki argümanla ilgili bir özelliği belirtirler. Parantez içinde birden fazla argüman olması halinde ise yüklem, bunlar arasındaki bir ilişkiyi tanımlar. Örneğin, ahmet ile mehmet arasında "ahmet mehmet'in babasıdır" şeklinde bir ilişki mevcut ise, bu ilişkiyi bir gerçek olarak şöyle yazabiliriz:

baba (ahmet, mehmet).

Yukarıda verilen kuralda, ':-' işareti 'eğer' anlamına gelir. X ise her değeri alabilen bir değişkeni gösterir. İşaretin sağ tarafındaki yüklemin ifade ettiği ilişki veya burada olduğu gibi özelliğin doğru olması halinde, Prolog sol taraftaki yüklemin de doğru olduğu sonucuna varır. Örneğin, kuralın sağındaki X değişkeni, insan (X) yüklemini doğru yapan bir değer alırsa, kuralın sol tarafı da aynı değer için doğru olur. Yani, X 'insan' olma özelliğini taşıyorsa 'ölümlü' olma özelliğini de taşıması gerekir. Bu örnekte insan (sokrat) bir gerçek olarak verildiği için, X = sokrat değerini alır ve sokrat'ın 'ölümlü' olduğu sonucu çıkar.

Şimdiye kadar Ahmet veya Sokrat gibi özel isimleri neden küçük harfla başlayarak yazdığımızı merak etmiş olabilirsiniz. Prolog dilinde sabit isimler ve yüklem isimleri küçük harfle, değişkenler ise büyük harfle başlar. Yüklem isimleri ve değişkenler için anlamlı sözcükler seçilmesi programı daha kolay okunur yapar. Örneğin yukarıdaki kuralı yazarken X yerine örneğin Birey sözcüğünü veya büyük harfle başlayan herhangi bir isim yazabilirdik. Burada kuralın basitliği ve yüklem isimlerinin yeteri derecede açıklayıcı olması nedeniyle, tek harfli bir değişken kullanmakla yetindik.

Klasik bir dille programlamada, yordamlar (prosedürler) ile bilgisayara tek tek hangi işleri yapacağı, hangi adımları atacağı anlatılır. Prolog'ta ise, çözülmesi istenilen problemin özelliklerini, gerçekler ve kurallar aracılığıyla deklare etmek yeterlidir. Böylece ortaya çıkan "bilgi tabanı"nı kullanarak, istenen çözümleri çıkarmak, Prolog dilinin içine yerleştirilmiş, sonuç çıkarma makinesi (inference engine) diye adlandırılan bir mekanizma sayesinde olur. Örneğin, bir önceki paragrafta, sokrat'ın bir insan olduğunu deklare eden gerçek ve insan olan herkesin ölümlü olduğunu deklare eden kuraldan oluşan çok kısa Prolog programına, sokrat'ın bir ölümlü olup olmadığını şöyle sorabiliriz:

```
baba (hamza, süleyman).
baba (süleyman, ibrahim).
baba (süleyman, ayşe).

erkek (hamza).
erkek (süleyman)
erkek (ibrahim).
erkek (hayri).

erkek çocuk (X, Y) : - baba (Y, X), erkek (X)
```

Şekil 1:

?- ölümlü (sokrat).

Prolog içine yerleştirilmiş "sonuç çıkarma makinasını" oluşturan iki temel algoritma sayesinde bu basit soruya kolayca 'evet' (yes) yanıtını verir. Prolog'un içine yerleştirilmiş bu mekanizmanın verilen gerçekler ve kurallardan giderek nasıl sonuca vardığına yukarıda biraz dokunduk. Detaya girmeden önce Prolog'a soru sormanın ne anlama geldiğini açıklayalım. Bir mantık programından bilgi almak (information retrieval), programa soru(lar) (query) yönelterek yapılır. Soru, Prolog ile yazılmış programa, nesneler arasında belli bir ilişkinin bulunup bulunmadığını veya bir nesnenin belli bir özelliği sağlayıp sağlamadığını, sorar. örneğin,

?- baba (ahmet, mehmet).

sorusu ile Prolog'a 'ahmet'in mehmet'in babası olup olmadığını' sorarız. Sorunun yanıtlanması, sorulan ilişki veya özelliğin programın mantıksal bir sonucu olup olmadığının, belirlenmesi anlamına gelir.

Sadece gerçeklerden oluşan bir programda soruların yanıtlanması basittir. Programdaki tüm gerçekler soruyla karşılaştırılır. Soruya tıpa tıp eşit olan bir gerçek varsa, Prolog'un vereceği yanıt "yes", aksi taktirde "no" olacaktır. Gerçek hayatta doğru olan, ama programın gerçekleri arasında bulunmayan bir ilişkinin doğruluğu sorulduğunda alacağımız yanıt ise "no" olacaktır. Çünkü, bu Prolog programı için sadece kendi bilgi tabanında verilen gerçek ve kurallardan çıkarılabilen ilişkiler doğrudur.

Prolog'a sorabileceğimiz sorular, sadece belli bir ilişki veya özelliğin programın gerçekleri arasında olup olmadığıyla sınırlı değildir. Soruda değişkenlerin bulunması halinde Prolog, bu değişkenlere sorudaki ilişkiyi doğru yapacak değerleri bulmaya çalışır. Örneğin, 'mehmet'in babası kimdir?' sorusunu şöyle sorabiliriz:

?- baba (X, mehmet).

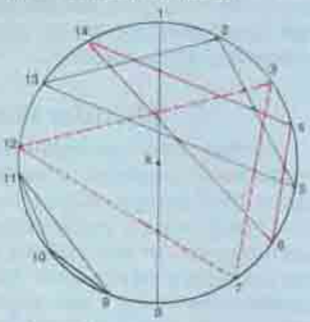
Diğer soru tiplerini bir başka örnek üzerinde anlatalım. Bu amaçla, Şekil 1'de verilen Prolog programı birden fazla gerçek ve sadece bir kuraldan oluşmaktadır.

İlk üç gerçekte verilen 'baba' yükleminin ardından ikinci sırada gelen 'erkek' yüklemi dört gerçekle verilmiştir. Programın tek kuralında ise, kuralın başı olarak adlandırılan ':-' işaretinin sol tarafında, X'in Y'nin erkek çocuğu olması ilişkisini gösteren erkek-çocuk yüklemi yer almaktadır. Kuralın gövdesi olarak adlandırılan ':-' işaretinin sağ tarafında ise baba

DÜŞÜNME KUTUSU

(Geçen sayıda yayınlanan soruların cevapları.)

KİRKMAN'IN OKUL KIZLARI PROBLEMİ: [illegible]



[illegible] Örneğin, 2-5-13 üçgeninde 2-5 kenarını gören 2345 yayı 3 birimdir: 2-3, 3-4 ve 4-5. 2-13 kenarını gören 2-1-14-13 yayı da 3 birimdir: 2-1, 1-14 ve 14-13. p = 2-5 kenarı ve q = 2-13 kenarı dersek görülüyor ki, p + q = r dir. r = 13-5 kenarı 6 yay birimidir: 13-14, 14-1, 1-2, 2-3, 3-4 ve 4-5; yani 3 + 3 = 6 gerçekleşmiştir. Toplam 4 üçgen vardır (3'ü ikizkenar ve biri üç kenarı farklı üçgen). Görülüyor ki 1'den 14'e kadar olan sayıların hepsi üçgen köşeleri olarak kullanılmıştır (1, k, 8 çapı hariç). İlk önce p + q = r veya p + q + r = 14 olacak şekilde sayılar aranır: (1,1,2), (2,4,6), (3,3,6) ve (5,5,4). İlk 3 parantezde 1 + 1 = 2, 2 + 4 = 6, 3 + 3 = 6 ve 5 + 5 + 4 = 14'dür. Bu sayılar yay birimleridir. Şimdi bunlara uyan 1 ile 14 arası sayıları bulalım: (9,10,11), (4,6,14), (2,5,13) ve (3,7,12). İşte 4 üçgeni çizmek için gerekli sayıları böyle bulduk. Örneğin (9,10,11) üçgenini alalım; 9-10 kenarı 1 yay birimi, 10-11 kenarı 1 yay birimi ve 9-11 kenarı 2 yay birimidir. Böylece 1,1,2 şartına uyulmuştur. 4-6-14 üçgenindeki kenarlar 2,4 ve 6 yay birimidir, dolayısıyla 2,4,6 şartına uyulmuştur ve şimdi üçgenlerin köşelerine ve çizdiğimiz çapa bakarak 1. günün tripletlerini yazalım: (1,k,8), (9,10,11), (4,6,14), (2,5,13), (3,7,12). Diğer günlerin tripletlerini bulmak için daire üzerindeki sayıları saat yönünde 2 yay birimi döndürelim, örneğin, o zaman 1 yerine 13, 8 yerine 6, 9 yerine 7, 10 yerine 8, 11 yerine 9 gelecektir.

Böylece

2. gün (13,k,6), (7,8,9), (2,4,12), (14,3,11), (1,5,10)
3. gün (11,k,4), (5,6,7), (14,2,10), (12,1,9), (13,3,8)
4. gün (9,k,2), (3,4,5), (12,14,8), (10,13,7), (11,1,6)
5. gün (7,k,14), (1,2,3), (10,12,6), (8,11,5), (9,13,4)
6. gün (5,k,12), (13,14,1), (8,10,4), (6,9,3), (7,11,2)
7. gün (3,k,10), (11,12,13), (6,8,2), (4,7,1), (5,9,14)

Tabii ki, k = 15'dir.

[illegible]

OLABİLİR Mİ? [illegible]

1/2 + 1/4 + 1/8 + ... + 1/2ⁿ + ... Bu serinin toplamı 1'e eşittir.

KEDİLER VE KÖPEKLER: [illegible]

Hareketler:
1) 11'deki köpek 9,7,10
2) 1'deki kedi 4,6
3) 2'deki kedi 4
4) 3'deki kedi 5,2
5) 10'daki köpek 5
6) 12'deki köpek 10
7) 6'daki kedi 9,11
8) 4'deki kedi 6,9
9) 2'deki kedi 4,6
10) 8'deki köpek 5,3
11) 10'daki köpek 8,5,2
12) 13'deki köpek 10,8,5
13) 9'daki kedi 12
14) 6'daki kedi 9
15) 2'deki köpek 4,1
16) 5'deki köpek 2
17) 9'daki kedi 7,10,13

İLK ÜÇE GİRME: Toplam olasılık 3 × 9! = 1 088 640 ile üçe girme olasılığı = 1/1 088 640. Birinci olma olasılığı 1/9! = 1/362 880 = milyonda 2,75.

EN ÇABUK MAT: 1-f4, e6 2-g4, Vh4 MAT.

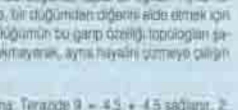
MÜZİK DURDU VE KADIN ÖLDÜ: [illegible]

ÜÇLÜ DÜĞÜM: [illegible]

İKİ KİLO BULGUR TARTMAK: [illegible]

[illegible]

CELEBES ADASI DİLİ:



ve erkek yüklemleri bulunmaktadır. Bu kuralı şöyle okuyabilirsiniz: 'X Y'nin erkek çocuğudur eğer Y X'in babasıysa ve X erkek ise.' Diğer bir deyişle bu kuralda, 'erkek-çocuk' yüklemi 'baba' ve 'erkek' yüklemleri aracılığıyla tanımlanmıştır.

Şekil 1'de verilen programa yöneltilecek 'süleyman'ın çocukları kimlerdir?' anlamına gelen,

?- baba (süleyman, X).

sorusuna, Prolog önce X = ibrahim yanıtını verir. Prolog'un bu yanıtı nasıl verdiğine geçmeden, önce Prolog'ın diğer dillerden önemli bir farkı olan, aynı soruya yeni yanıtlar arayabilme özelliğinden bahsedelim. Yukarıdaki soruya verilen ilk yanıttan sonra Prolog'a, 'veya' anlamına gelen ';' işaretiyle X için yeni değerler aramasını emredebiliriz. Alacağımız yanıt X = ayfer olacaktır.

(Devam edecek.)