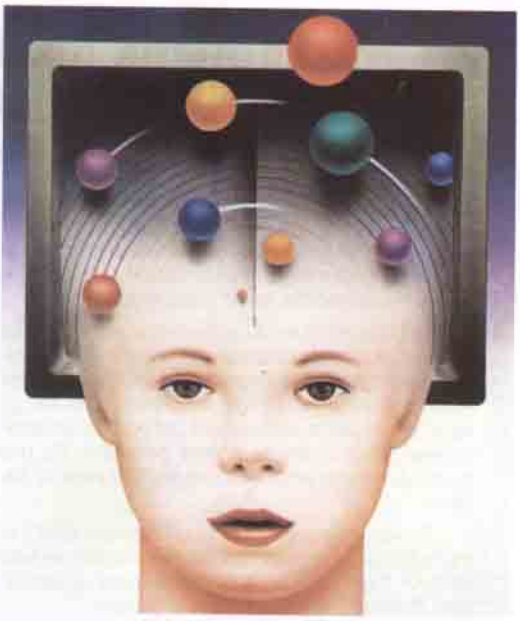
Bir Mantık Programlama Dili

PROLOG

(Bölüm II)

Selçuk TARAL*



Geçen bölümde, bir Prolog programının gerçekler ve kurallardan oluşan yapısını örneklerle anlatmıştık. Gerçek ve kurallar için bazen *Prolog tümceleri* (clauses) deyimi de kullanılır. Bu bölümde Prolog'un *sonuç çıkarma* mekanizmasının, programın tümcelerinden giderek nasıl mantıksal sonuçlar çıkardığını anlatmaya çalışacağız. Birinci bölümdeki örnek programı anımsayacaksınız. Bu programa **'?-baba (süleyman, X).'** sorusunu yönelttiğimizde Prolog önce **X = ibrahim** yanıtını vermişti. Prolog bu çözüme **birleşme algoritması** diye çevirebileceğimiz, *"unification algorithm"* adında bir yöntemle varır. Şimdi bu algoritmayı aynı örnek üzerinde açıklamaya çalışalım. Birleşme algoritması mantık programlamasının temelinde yatan bir yöntemdir.

Prolog, çözüm bulmak için programın tümcelerini *yukarıdan aşağıya doğru tek tek tarar* ve soruya *uyum sağlayan* (matching) ilk gerçeği veya kuralı bulur. *Uyum* iki ilişkinin *aynı yüklem adında* olmaları ve *aynı sayıda argümanı içermeleri* halinde sağlanır. Birleşme ise (unification) ancak uyum sağlandıktan sonra söz konusu olabilir. Programa yönelttiğimiz '?-baba (süleyman, X).' sorusu, programın ilk gerçeği ile uyum sağlar, ancak birleşme gerçekleşmez; çünkü birinci argümanları farklı sabit isimlerdir: süleyman ve hamza. *İki sabit terim (atom) ancak eşit olduklarında birleşebilirler.*

Prolog şimdi 'baba' yükleminin ikinci gerçeğini dener ve birleşme sağlanır. Birinci argümanlar eşit, ikinci argümanlar ise soruda *değişken*, gerçekte ise bir sabittir. *Değişken sabitin değerine bağlanır* (binding), böylece birleşme sağlanmış olur. Sonuç olarak, Prolog ilk çözümü verir: **X = ibrahim.** Prolog'tan aynı soruya yeni bir yanıt araması istendiğinde, *X değişkeni yeniden serbest kalır* ve tarama işlemi *kaldığı yerden* devam eder. Bu kez programın üçüncü gerçeği **X = ayşe** çözümünü verir. Tekrar tarama istendiğinde, bu noktadan sonra 'baba' yüklemi için yazılmış başka bir tümce bulunmadığı için, alacağımız yanıt "başka çözüm yok" anlamındaki "no" olur. Prolog'un bu şekilde yeni çözümler aramasında kullandığı yönteme, *geriye doğru iz sürme* anlamında **"backtracking" algoritması** denir. Bu algoritmayı ilerde anlatacağız.

Aynı programa bu kez şu soruyu yöneltelim: **'? - erkek_çocuk (Z, süleyman).'** Bu ilişkiyi sağlayacak Z değerini bulmak amacıyla Prolog, 'erkek_çocuk' yüklemini tanımlayan programın tek kuralını (erkek_çocuk (X,Y): - baba (Y,X), erkek (X).) harekete geçirir. Sorunun ve kuralın başındaki ilişkinin birinci argümanları Z ve X değişkenleridir. Bu iki değişkenin *birleşmesi* şu anlama gelir: Değişkenlerden biri herhangi bir değere bağlanırsa, diğeri de *aynı değere* bağlanır. Kuralın ikinci argümanı olan Y değişkeni ise, yukarıda olduğu gibi sabitle birleşip, 'süleyman' değerini alır. *Kural içinde değişkenin bir değere bağlanması, kuralın tümü için geçerlidir.* Bu nedenle, kuralın sağ tarafındaki Y değişkeni de 'süleyman' sabit değerine bağlanmış olur.

Sonuçta **'? - erkek_çocuk (Z, süleyman).'** sorusu, yukarıda anlattığımız şekilde tek erkek_çocuk kuralının başıyla (erkek_çocuk (X, süleyman)) birleşmesi sonucu, kuralın gövdesiyle yer değiştirerek **'? - baba (süleyman, X), erkek (X).'** sorusuna dönüşür. Diğer bir deyişle, Prolog 'erkek_çocuk' ilişkisi için baba ve erkek ilişkileri içeren yeni bir soru türetmiş olur. *Soruların kurallar aracılığı ile yeni sorulara indirgenmesi işlemi, mantık programlamasıyla hesaplamanın temel adımıdır.*

Prolog'a bir soru yöneltmekle aslında ona bilgi tabanındaki tümcelerden giderek sağlamasını istediğimiz bir **hedef** ("goal") gösteriyoruz. Sorunun değişkenler içermesi halinde (bulunabilirse), hedefi sağlayacak değer(ler) rapor edilecektir. Prolog bilgi tabanından, hedeflenen sonuca götürecek gerçek ve kuralları bulduğunda, başarı sağlanır (goal satisfied), aksi takdirde başarı sağlanmaz (goal failed). Verilen hedef bir kuralın başı ise, Prolog kuralın gövdesindeki ilişkileri ara veya *alt hedefler* olarak devreye sokar ve bunları sağlamaya çalışır. Örnekte erkek_çocuk (Z, süleyman) sorusuna yanıt bulmak hedefi, kuralın "ateşlenmesiyle" baba (süleyman, X) ve erkek (X) alt hedeflerini sağlama görevine dönüşmüş olur.

Prolog'un bağlaçlı soruları nasıl yanıtladığına geçmeden önce, Şekil 2'de verilen gerçek ve kuralları da ekleyerek bilgi tabanımızı biraz zenginleştirelim.

* *TÜBİTAK, Bilgi İşlem Daire Başkanı.*

```
anne (fatma, ibrahim).
anne (fatma, ayşe).          kadın (fatma).
anne (ayşe, hayri).          kadın (ayşe).

ebeveyn (X,Y) : - baba (X,Y).
ebeveyn (X,Y) : - anne (X,Y).

dede (X,Y) : -
  baba (X,Z), ebeveyn (Z,Y).

erkek_kardeş (X,Y) : -
  ebeveyn (Z,X), ebeveyn (Z,Y), erkek (X).

amca (X,Y) : -
  erkek_kardeş (X,Z), baba (Z,Y).
```

Şekil 2

Şekil 2'de anne ve kadın yüklemlerine ait gerçekleri, programın fazla yer tutmaması için aynı satırlarda yazdık. Aynı yükleme ait gerçekleri alt alta ve aralarında boşluk kalacak şekilde gruplar halinde yazmanız programı daha kolay okunur yapacaktır. Gerçekler bittikten sonra kurallar da aynı şekilde yazılır. İlerde göreceğimiz gibi, bir yüklemin tanımlanması birden fazla kuralın yazılmasını gerektiriyorsa, kuralların yazılış sırası önemlidir. Sıralamada yapılacak hatalar bazı beklenmedik sonuçlar doğurabilir.

Şimdi yukarıda verilen bağlaçlı soruya (?- baba (süleyman, X), erkek (X).) geri dönelim. Prolog bu soruyu yanıtlamak için alt hedefleri *soldan sağa doğru sırayla* sağlamaya çalışır. Önce birinci alt hedefi sağlayan X değerini bulur: **X = ibrahim.** X, değişkeni ortak olduğu için ikinci alt hedef **'? - erkek (ibrahim).'** sorusuna dönüşür. Bilgi tabanında böyle bir gerçek bulunduğundan yanıt olumlu olacaktır. Her iki alt hedef sırasıyla sağlandığından, X'in aldığı bu değer bize ilk çözüm olarak verilir; **Z = ibrahim** (yanıt niçin Z'nin bir değeri olarak verilir?).

Yeni çözüm istediğimiz zaman *"backtracking"* algoritması devreye girer. Böylece Prolog, hedefte son olarak bir değişkene değer bağladığı noktaya geri döner. Örnekte **baba (süleyman, X)** alt hedefinin programdaki **"baba (süleyman, ibrahim)."** gerçeğiyle birleşmesi sonucu X değişkeni 'ibrahim' değerini almıştı. Şimdi Prolog aynı hedefe alternatif bir çözüm aramaya kaldığı yerden devam eder ve **"baba (süleyman, ayşe)."** gerçeğiyle birleşme sonucu X, bu kez 'ayşe' sabit değerine bağlanır. Ancak bilgi tabanında ayşe'nin erkek olduğuna dair bir gerçek bulunmadığından (bulunsaydı, Prolog bunu "ayşe hiç erkek olur mu?" diye sormadan gerçek olarak kabul edecekti!), Prolog ilk alt hedefe geri döner ve X'i serbest bırakıp yeni bir çözüm peşinde koşar. Ancak programda "baba" yüklemi için yazılmış başka bir tümce bulamadığından, yeni bir çözüm üretemez ve bu durumu ekrana "no" yazarak bildirir.

"Backtracking" algoritmasını bir labirentte çıkış noktalarını aramaya benzetebilirsiniz. Birden fazla yol ayrımının bulunduğu köşelere işaret bırakarak, bir önce seçtiğiniz yolun çıkmaza girmesi veya yeni bir çözüm aramak isteğiyle bu noktalara geri döndüğünüzde, son seçimi yaptığınız noktadan başlayarak, daha önce denemediğiniz yeni yolları deneyebilirsiniz. Bu benzetmede birden fazla yolla karşılaşıldığı zaman birini seçmek, Prolog'ta bir değişkenin birleşme algoritması sonucu bir değere bağlamasına karşılık gelir.

Şekil 2'de verilen ebeveyn yüklemi iki kuralla tanımlanmıştır. Kuralların ikisi de basit birer gerçeğe dayandıklarından, bunları 'veya' anlamına gelen ';' işaretini kullanarak tek bir kural gibi yazabilirsiniz:

ebeveyn (X,Y) : - anne (X,Y); baba (X,Y).

Ancak birden fazla kuralla tanımlanabilen yüklemler için, sadece yukarıdaki gibi basit olanları veya ile birleştiririz. Genelde her kural alt alta ve ayrı olarak yazılır.

Burada 'erkek_kardeş' ilişkisi için ortak tek bir ebeveyn olmasını yeterli bulduk (Aynı anne ve babadan olmalarını şart koşsaydık, kuralı nasıl yazardınız?). Bu kuralda Z değişkeninin ortak ebeveyni gösteren kullanımına dikkatinizi çekeriz. Yine dikkat ederseniz, 'amca' yüklemini tanımlayan kural 'erkek_kardeş' yüklemini tanımlayan kurala dayanır; önce 'erkek_kardeş' sonra 'amca' ilişkisini tanımladık. Son kuralı şöyle okuyabiliriz: *"X ile gösterilen kişi Y ile gösterilen kişinin amcasıdır* ***eğer*** *X Z ile gösterilen bir kişinin erkek kardeşi ve Z de Y'nin babası ise."* Kurallar, bu şekilde *mantıksal aksiyomlar* gibi okunabilmelerinin (declarative reading) yanı sıra, atılacak adımları gösteren birer yordam gibi de okunabilirler (procedural reading). Örneğin, 'amca' yüklemi için verilen tek kuralın yordamsal okunuşu şöyledir: *"X'in Y'nin amcası olduğunu ispat etmek için önce X'in Z ile gösterilen bir başka kişinin erkek kardeşi olduğunu ve sonra bu Z'nin de Y'nin babası olduğunu ispat et."* Yukarıda '? - erkek_çocuk (Z, süleyman),sorusuna yanıt ararken 'erkek_çocuk' kuralı bir yordam olarak yorumlanmıştı.

Diğer bazı akrabalık ilişkileri (hala, dayı vb.) için tümcelerin yazılması işini ilgilenen okuyuculara bırakıyoruz. Yine ilgilenen okuyuculara, örnek programdan **?- dede (hamza, Kim)** veya **?- erkek_kardeş (Kim, ayşe)** gibi soruların yanıtlarını, birleşme ve "backtracking" algoritmalarını kullanarak çıkarmalarını öneririz.

Yukarıda verdiğimiz kuralların tümünde yeni ilişkiler, önceden verilmiş ilişkiler kullanılarak tanımlanmışlardı. Prolog dilinin en önemli özelliklerinden biri **özyinelemeli** (recursive) kuralların çok sık kullanımıdır. Özyinelemeli fonksiyonlar veya yordamlar, *tanımlarında kendi kendilerini çağırırlar.* Örneğin, pozitif bir tam sayının (N) faktoriyeli o sayıdan bir eksik olan sayının (N-1) faktoriyeli ile kendinin çarpımı olarak tanımlanır: N! = Nx(N-1)!. Özyinelemeli ilişkiye iyi bir örnek, yukarıda verilen programa ekleyebileceğimiz, "ata" ilişkisidir. Bu ilişkiyi iki kuralla tanımlayabiliriz:

SAYILARIN DİLİ

Dr. Üstün AYDINGÖZ

- Karalarda fotosentez ile ele geçirilen güneş enerjisinin insanlar ve evcil hayvanlar tarafından tüketilen kısmının yüzdesi: 3.
- ABD'de tedavüldeki kâğıt paraların eser miktarda da olsa kokain içerenlerinin yüzdesi: 97.
- Bir Japon çalışanın ücretli izin hakkının kullandığı kısmı, ortalama olarak: 1:2.
- Japonya'da her yıl mezun olan mühendislerin hukukçulara oranı: 10:1.
- ABD'de aynı oran: 1:10.
- 20 yıl önce Jüpiter gezegeninin yakınından geçmek üzere uzaya gönderilen ve daha sonra Güneş Sistemi'nin dışına çıkan Pioneer 10 uzay aracının şu sıralarda Dünya'ya uzaklığı, kilometre olarak: 8 milyar.
- Pioneer 10'un şu sıralardaki hızı, saniyede kilometre olarak: 13.
- Kuru meyveler katılmış ortalama bir kekin yoğunluğunun maun ağacının yoğunluğuna oranı: 1:1.
- Bir yetişkinin vücudundaki benlerin sayısı, ortalama olarak: 25.
- Günümüzde ABD nüfusunda zencilerin yeri, yüzde olarak: 12.
- Aynı yüzde, 1890 yılında: 12.
- Amerikalıların evlat edindiği yabancı bebekler arasında Afrika'dan gelenlerin yüzdesi: 0.1.
- Geçtiğimiz yıl bütün dünyadaki Rolls-Royce satışlarındaki değişiklik, yüzde olarak: —48.
- New York şehrinde bir hırsızın kilitli bir arabanın içine ulaşması için gerekli tahmini süre, saniye olarak: 27.
- Bir otomobilin elektrikli âletler kullanılmadan satılabilir parçalara ayrıldığı rekor süre, dakika olarak: 8,7.
- Amerikan Ulusal Havacılık ve Uzay Dairesi NASA'nın tahminlerine göre uzay mekiğinin uzay "çöplüğü"ne ait bir parçaya çarpması olasılığı: 1:30.
- Amerikan uzay mekiğine uzay uçuşu sırasında çarpan cisimlerin hasar verdiği mekik camlarının sayısı: 17.
- Bu camların her birinin ortalama maliyeti, ABD doları olarak: 50 000.
- Yetişkin bir insan vücudundaki kan damarlarının uzunluğu, kilometre olarak: 100 000.
- Amerikan uzay mekiğinin fırlatılışı sırasında kullanılan güç, beygir gücü olarak: 44 milyon.
- Körfez Savaşı sırasında müttefik güçler tarafından Kuveyt'e atılan bütün patlayıcılardan patlamamış olanların bölümü: 1:3.

Buradaki verilerin bir bölümü **Harper's Index**'ten alınmıştır; diğerleri çeşitli kaynaklardan derlenmiştir. Hangi verinin hangi kaynaktan alındığını öğrenmek isterseniz mektupla başvurunuz.

```
ata (X,Y) :-
    ebeveyn (X,Y).
ata (X,Y) :-
    ebeveyn (X,Z), ata (Z,Y).
```

Birinci kural ebeveynleri 'ata' olarak tanımlar. Özyinelemeli olan ikinci kural ise şöyle okunabilir: *X ile gösterilen kişi Y ile gösterilen kişinin atasıdır, eğer X Z ile gösterilen bir kişinin ebeveyni ve Z de Y'nin atası ise.* Prolog hesap yapmak için de kullanılabilir. Örneğin, özyinelemeli faktoriyel fonksiyon aşağıda verildiği şekilde tanımlanabilir.

```
faktoriyel (0,1).
faktoriyel (N, Sonuç) :-
    N1 is N-1,
    faktoriyel (N1, AraSonuç),
    Sonuç is AraSonuç x N.
```

Birinci gerçek 0 sayısının faktoriyelinin 1 olduğunu deklare eder. Programın özyinelemeli olan esas kuralı ise, birinci argümandaki N sayısının faktoriyelini hesaplayıp neticeyi Sonuç adındaki değişkene yazar. Örneğin, **?- faktoriyel (5, Sonuç)** sorusunun yanıtı Sonuç = 120 olur. Özyinelemeli kurallarla ilgili bilgi ve yeni örnekleri bir sonraki yazımıza bırakıyoruz.

(Devamı edecek.)

DÜZELTME

Birinci bölümün son tümcesi X = ayfer yanıtı yanlış yazılmış; doğru yanıt **X = ayşe** olacaktı. Ayrıca, "**ölümlü (X) :- insan (X).**" kuralının yazılışında sondaki nokta işareti unutulmuş; *Prolog'ta tüm tümcelerin sonuna nokta konulması gerekir.*