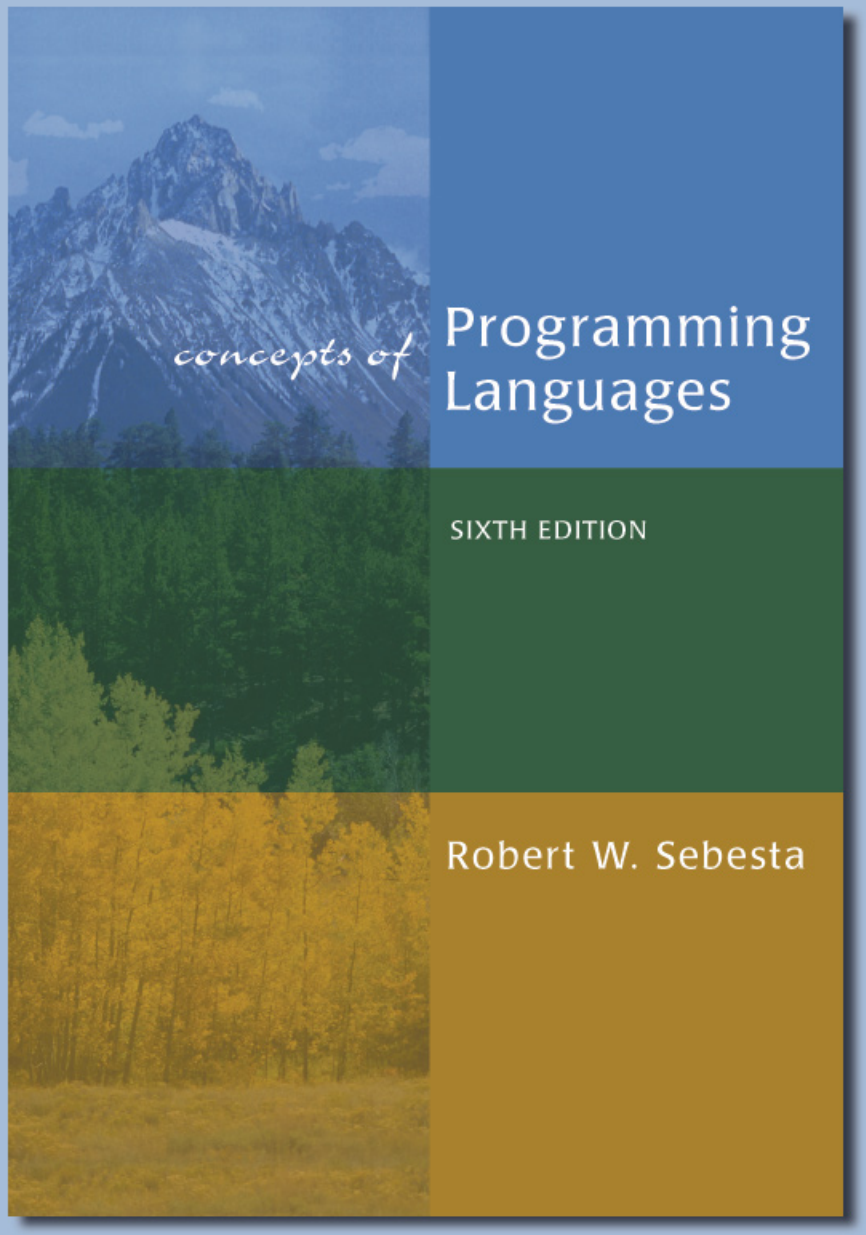
Bölüm 16

Mantık(Lojik-Logic)
Programlama Dilleri



ISBN 0-321-19362-8

PEARSON
Addison
Wesley

Bölüm 16 Topics

- Giriş
- Hüküm Hesabına(Predicate Calculus) Kısa bir Giriş
- Hüküm Hesabı(Predicate Calculus) ve Teorem İspatlama
- Mantık Programlamaya Genel Bakış
- Prolog'un Kökenleri
- Prolog'un temel elemanları
- Prolog'un eksikleri
- Mantık programlama uygulamaları

Giriş

- Mantık(Logic) programlama dili veya tanıtıcı(declarative) programlama dili
- Programları sembolik mantık biçiminde ifade etme
- Sonuçları üretmek için mantıksal çıkarsama (logical inferencing) işlemi kullanma
- Yordamsal(procedural) yerine Tanıtıcı(Declarative) :
  - Sadece sonuçların(results) özelliği belirtilir (onları üreten detaylı prosedürlerin(yordamların) (procedures) değil)

Predicate Calculus ‘a Giriş

- Önerme(Proposition): doğru olan veya olmayan bir mantıksal ifade
  - Nesneler(objects) ve bunların birbiri arasındaki ilişkilerini(relationships) içerir

Predicate Calculus ‘a Giriş

- Sembolik Mantık(Symbolic logic) biçimsel mantığın(formal logic) temel ihtiyaçları kullanılabilir :
  - Önermeleri(propositions) ifade etme
  - Önermeler arasındaki ilişkileri(relationships) ifade etme
  - Diğer önermelerden nasıl yeni önermeler çıkarılabileceğini anlatma
- Sembolik mantığın mantık programlama için kullanılan belli bir biçimine predicate calculus adı verilir

Önermeler(Propositions)

- Önermelerdeki nesneler basit terimlerle ifade edilir: sabitler(constants) veya değişkenler(variables)
- Sabit(Constant): bir nesneyi gösteren bir sembol(symbol)
- Değişken(Variable): farklı zamanlarda farklı nesneleri gösterebilen bir sembol(symbol)
  - Buyurgan dillerdeki(imperative languages) değişkenlerden(variables) farklıdır

Önermeler(Propositions)

- Atomik Önermeler(Atomic propositions) bileşik terimlerden(compound terms) oluşur
- Bileşik Terim(Compound term): matematiksel fonksiyon gibi yazılan, matematiksel ilişkinin bir elemanı
  - Matematiksel fonksiyon bir eşlemedir(mapping)
  - Bir tablo olarak yazılabilir

Önermeler(Propositions)

- Bileşik terim(Compound term) iki kısımdan oluşur
  - Functor: ilişkiyi(relationship) adlandıran fonksiyon sembolü(symbol)
  - Parametrelerin sıralı listesi (tuple)
- Örnekler:

  student(jon)

  like(seth, OSX)

  like(nick, windows)

  like(jim, linux)

Önermeler(Propositions)

- Önermeler iki biçimde belirtilir:
  - Gerçek(Fact): doğru olduğu varsayılan önerme
  - Sorgu(Query): önermenin doğruluğuna karar verilir
- Bileşik Önerme(Compound proposition):
  - İki veya daha fazla atomik önerme içerir
  - Önermeler operatörlerle bağlanır

Mantıksal Operatörler

Adı	Sembol	Örnek	Anlamı
Negation (olumsuzluk)	$\neg$	$\neg$ a	a'nın değili
Conjunction (birleşme ve ile)	$\cap$	a $\cap$ b	a ve b
disjunction (ayrılma veya ile)	$\cup$	a $\cup$ b	a veya b
equivalence (eşitlik)	$\equiv$	a $\equiv$ b	a eşittir b
Implication (içerme)	$\supset$ $\subseteq$	a $\supset$ b a $\subseteq$ b	a ,b yi içerir b ,a yı içerir

Niceleyiciler(Quantifiers)

adı	örnek	anlamı
Universal (evrensel)	∀X.P	her X için, P doğrudur
Existential (varoluşsal)	∃X.P	P nin doğru değeri için bir X değeri vardır

Cümlesel Biçim(Clausal Form)

- Aynı şeyi belirtmek için çok fazla yol
- Önermeler için standart bir form kullan
- Cümlesel Biçim(Clausal form):

$$B_1 \cup B_2 \cup \ldots \cup B_n \subset A_1 \cap A_2 \cap \ldots \cap A_m$$

şu anlama gelir : eğer bütün A lar doğru ise, o zaman en az bir B doğrudur

- Önceki(Antecedent): sağ taraf
- Sonuç(Consequent): sol taraf

Predicate Calculus ve Teorem(theorems) ispatlama

- Önermelerin (propositions) bir kullanımı bilinen aksiyomlardan(axioms) ve teoremlerden(theorems) çıkarılabilen yeni teoremler keşfetmektir
- Çözüm(Resolution): verilen önermelerden çıkarılmış önermelerin(inferred propositions) hesaplanmasına imkan veren bir çıkarım prensibi (inference principle)

Çözüm(Resolution)

- Birleştirme(Unification): eşlenme(matching) işleminin başarılı olması için önermelerdeki(propositions) değişkenler(variables) için değerler(values) bulma
- Başlatma(Instantiation): birleştirmenin(unification) başarılı olması için değişkenlere(variables) geçici değerler atama
- Bir değişkeni(variable) bir değerle başlattıktan sonra, eğer eşlenme(matching) başarısız olursa, geri-izleme(backtrack) ve farklı bir değerle yeniden başlatma yapmaya gereksinim duyabilir

Teorem İspatlama(Theorem Proving)

- Çelişki(contradiction) ile kanıt (proof) kullanır
- Hipotez(Hypotheses): bir geçerli önermeler (pertinent propositions) kümesi
- Hedef(Goal): teoremin(theorem) olumsuzluğu(negation) önerme(proposition) olarak belirtilir
- Bir tutarsızlık(inconsistency) bulunarak teorem (theorem) ispatlanır

Teorem İspatlama

- Mantık(logic) programlamanın temeli
- Önermeler(propositions) çözüm(resolution) için kullanıldığı zaman, sadece kısıtlanmış(restricted) biçim kullanılabilir
- Horn clause – sadece iki biçimi olabilir
  - Headed: sol kısımda basit atomik önerme
  - Headless: boş sol kısım (gerçek(fact)leri belirtmek için kullanılır)
- Çoğu önermeler Horn clause olarak belirtilebilir

Mantık programlamaya genel bakış

- Tanıtıcı semantik(Declarative semantics)
  - Her bir ifadenin anlamını belirlemek için basit bir yol vardır
  - Buyurgan dillerin sematiğinden daha basittir
- Programlama yordamsal değildir(nonprocedural)
  - Programlar hesaplanan bir sonuç belirtmez, fakat sonucun biçimini belirtir

Örnek: bir listeyi sıralama

- Bir sıralı listenin özelliğini tanımlamak, listeyi yeniden düzenleme işlemi değildir

$\text{sort(old\_list, new\_list)} \subset \text{permute (old\_list, new\_list)} \cap \text{sorted (new\_list)}$

$\text{sorted (list)} \subset \forall_j \text{ such that } 1 \leq j < n, \text{list}(j) \leq \text{list}(j+1)$

Prolog'un esasları

- University of Aix-Marseille
  - Doğal Dil İşleme(Natural language processing)
- University of Edinburgh
  - Otomatik Teorem İspatlama(Automated theorem proving)

Prolog'un temel elemanları

- Edinburgh Syntax
- Terim(Term): bir sabit(constant), değişken(variable), veya yapı(structure)
- Sabit(Constant): bir atom veya bir tamsayı(integer)
- Atom: Prolog'un sembolik değeri
- Atom şunlardan birinden oluşur:
  - Küçük harfle başlayan harfler(letters), rakamlar(digits), ve alt-tirelerden(underscores) oluşan bir string
  - Kesme işaretleriyle (apostrophes) yazdırılabilir ASCII karakterlerinden oluşan bir string

Prolog'un temel elemanları

- Değişken(Variable): büyük harfle başlayan, harfler(letters), rakamlar(digits), ve alt-tirelerden (underscores) oluşan herhangi bir string
- Başlatma(Instantiation): bir değişkenin bir değere bağlanması
  - Sadece bir hedefe tamamen ulaşana kadar sürer
- Yapı(Structure): atomik önerme
  functor(parametre listesi)'ı gösterir

Gerçek İfadeleri(Fact Statements)

- Hipotezler(hypotheses) için kullanılır
- Headless Horn cümleleri

```
student(jonathan).
sophomore(ben).
brother(tyler, cj).
```

Kural ifadeleri(Rule Statements)

- Hipotezler(hypotheses) için kullanılır
- Headed Horn cümlesi
- Sağ kısım: önceki(antecedent) (***if*** kısmı)
  - Basit terim veya birleşme(conjunction) olabilir
- Sol kısım: sonuç(consequent) (***then*** kısmı)
  - Basit terim olmalıdır
- Birleşme(Conjunction): mantıksal AND işlemleriyle ayrılmış çoklu terimler(multiple terms)

Kural ifadeleri(Rule Statements)

```
parent(kim,kathy):- mother(kim,kathy).
```

- Anlamı genelleştirmek için değişkenler (evrensel nesneler-universal objects) kullanabilir:

```
parent(X,Y):- mother(X,Y).
sibling(X,Y):- mother(M,X),
               mother(M,Y),
               father(F,X),
               father(F,Y).
```

Hedef İfadeleri(Goal Statements)

- Teorem ispatlama için, teorem sistemin ispat etmesini veya etmemesini istediğimiz önermenin biçimindedir – hedef ifadesi(goal statement)
- headless Horn daki aynı biçim

```
student(james)
```

- Bileşik önermeler(Conjunctive propositions) ve değişkenli önermeler de geçerli hedeflerdir

```
father(X,joe)
```

Prolog'un Çıkarsama işlemi(Inferencing Process)

- Sorgulara(Queries) hedef(goals) denir
- Eğer bir hedef(goal) bir bileşik ifade ise(compound proposition), her bir gerçek(facts) bir alt-hedeftir (subgoal)
- bir hedefin(goal) doğruluğunu(true) ispatlamak için, çıkarım kuralları(inference rules) ve/veya gerçeklerden(facts) oluşan bir zincir bulmalıdır.

Hedef(goal) Q için:

```
B :- A
C :- B
…
Q :- P
```

- Althedefi ispatlama işlemine eşleme(matching), sağlama(satisfying), veya çözüm(resolution) adı verilir

Çıkarsama işlemi(Inferencing Process)

- Aşağıdan-yukarıya çözüm, ileri zincirleme(Bottom-up resolution, forward chaining)
  - Gerçekler(facts) ve veritabanı(database) kurallarıyla (rules) başlar ve hedefe(goal) ulaştıracak sırayı bulmaya çalışır
  - Geniş bir olası doğru cevaplar kümesiyle iyi çalışır
- Yukarıdan-aşağıya çözüm, geri zincirleme(Top-down resolution, backward chaining)
  - Hedef ile başlar ve veritabanındaki gerçekler(facts) kümesine ulaştıran sırayı(sequence) bulmaya çalışır
  - Küçük bir olası doğru cevaplar kümesiyle iyi çalışır
- Prolog implementasyonları geri zincirleme(backward chaining) kullanır

Çıkarsama işlemi(Inferencing Process)

- Hedef birden fazla alt hedefe sahipse, şunlardan birini kullanır
  - Depth-first arama: diğerleriyle çaılşmadan önce ilk althedefin tamamen ispatını bulmak
  - Breadth-first arama: bütün alt hedefler üzerinde paralel çalışma
- Prolog depth-first arama kullanır
  - Daha az bilgisayar kaynağıyla yapılabilir

Çıkarsama işlemi(Inferencing Process)

- Birden çok alt-hedefi(subgoal) bulunan bir hedef(goal) ile, eğer althedeflerden birinin doğruluğunu göstermekte başarısız olunursa, alternatif bir çözüm için bir önceki althedef yeniden ele alınır: geri-izleme(backtracking)
- Bir önceki aramanın bıraktığı yerden aramaya başlanır
- Çok fazla zaman ve alan alabilir çünkü her bir alt-hedefin(subgoal) olası bütün ispatlarını bulabilir

Basit Aritmetik

- Prolog tamsayı değişkenlerini(integer variables) ve tamsayı aritmetiğini destekler
- ***is*** operatörü: sağ işlenen(operand) olarak bir aritmetik ifadeyi ve sol işlenen(operand) olarak da değişkeni(variable) alır
- A is B / 10 + C
- Atama ifadesi(assignment statement) ile aynı değildir!

Örnek

```
speed(ford,100).
speed(chevy,105).
speed(dodge,95).
speed(volvo,80).
time(ford,20).
time(chevy,21).
time(dodge,24).
time(volvo,24).
distance(X,Y) :-  speed(X,Speed),
                  time(X,Time),
                  Y is Speed * Time.
```

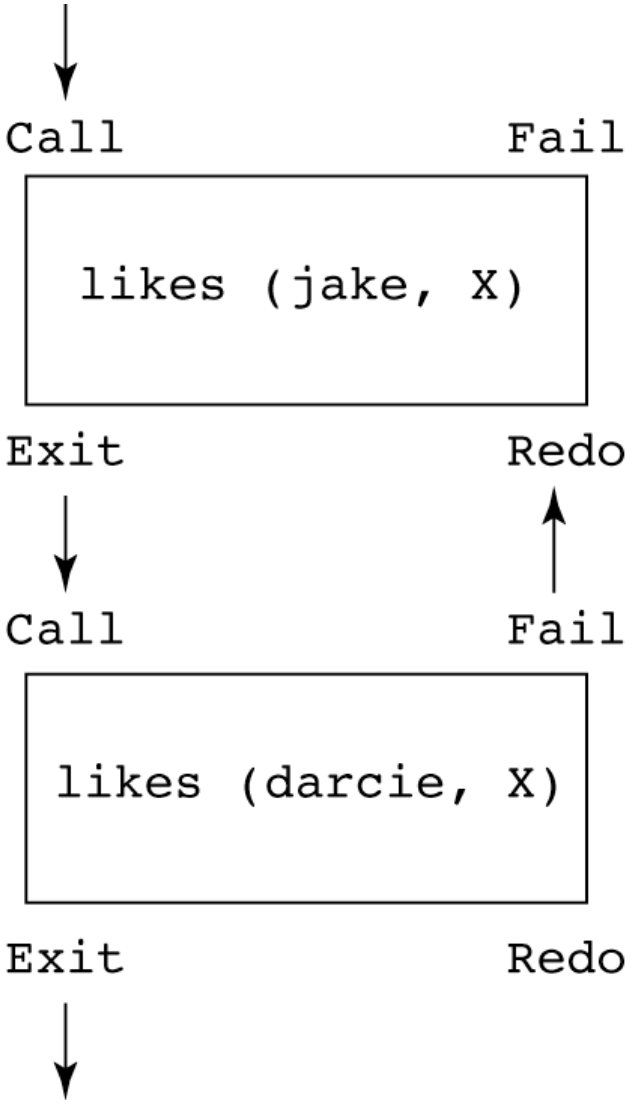
İzleme(Trace)

- Her adımdaki başlatmaları(instantiations) gösteren yerleşik yapı(built-in structure)
- Yürütmenin İzleme Modeli(Tracing model of execution)- dört olay:
  - Çağırma(Call) (hedefi(goal) gerçekleştirme çabasının başlangıcı)
  - Çıkış(Exit) (hedef gerçekleştirilmiş olunca)
  - Yinele(Redo) (geriizleme(backtrack) olur)
  - Başarısız(Fail) (hedef başarısız olduğunda)

Örnek

```
likes(jake,chocolate).
likes(jake,apricots).
likes(darcie,licorice).
likes(darcie,apricots).

trace.
likes(jake,X),
likes(darcie,X).
```



Liste yapıları(List Structures)

- Diğer temel veri yapısı(data structure) (daha önce gördüğümüz atomik önermelere(propositions) ek olarak): liste
- Liste herhangi bir sayıdaki elemanlar(elements) sırasıdır
- Elemanlar atomlar, atomik önermeler(propositions), veya diğer terimler (diğer listeler de dahil) olabilir

```
[apple, prune, grape, kumquat]
[]              (boş liste)
[X | Y]         (baş (head) X ve kuyruk(tail) Y)
```

Örnek

- append fonksiyonunun tanımı:

```
append([], List, List).
append([Head | List_1], List_2, [Head
  | List_3]) :-
  append (List_1, List_2, List_3).
```

Örnek

- reverse fonksiyonunun tanımı:

```
reverse([], []).
reverse([Head | Tail], List) :-
  reverse (Tail, Result),
  append (Result, [Head], List).
```

Prolog'un eksiklikleri(Deficiencies)

- Çözüm (Resolution) sırası kontrolü
- Kapalı-çevre varsayımı (closed-world assumption)
- Değilini alma (negation) problemi
- Yerleşik kısıtlamalar(Intrinsic limitations)

Mantık programlama uygulamaları

- İlişkisel veritabanı yönetim sistemleri(Relational database management systems)
- Exper Sistemleri(Expert systems)
- Doğal Dil işleme(Natural language processing)
- Eğitim(Education)

Sonuçlar

- Avantajlar:
  - Prolog programlar mantığa dayalıdır, bu yüzden so daha mantıksal düzenlenebilir ve yazılabilir
  - İşleme doğal olarak paraleldir, bu yüzden Prolog yorumlayıcıları(interpreters) çoklu-işlemcili makine avantajını kullanabilirler
  - Programla kısa ve özdür, bu yüzden geliştirme süresi azalmıştır– prototipleme(ilk-ürün oluşturma-prototyping) için iyidir