190600213201 5
Programlama Dilleri Temelleri

BAİBÜ Bilgisayar Müh.

Ders 9

Dr. Öğr. Üyesi İsmail Hakkı Parlak

Kaynaklar: Watt, David A., Programming Language Design Concepts, Wiley

Bazı şekiller kaynak kitaptan kopyalanmıştır.

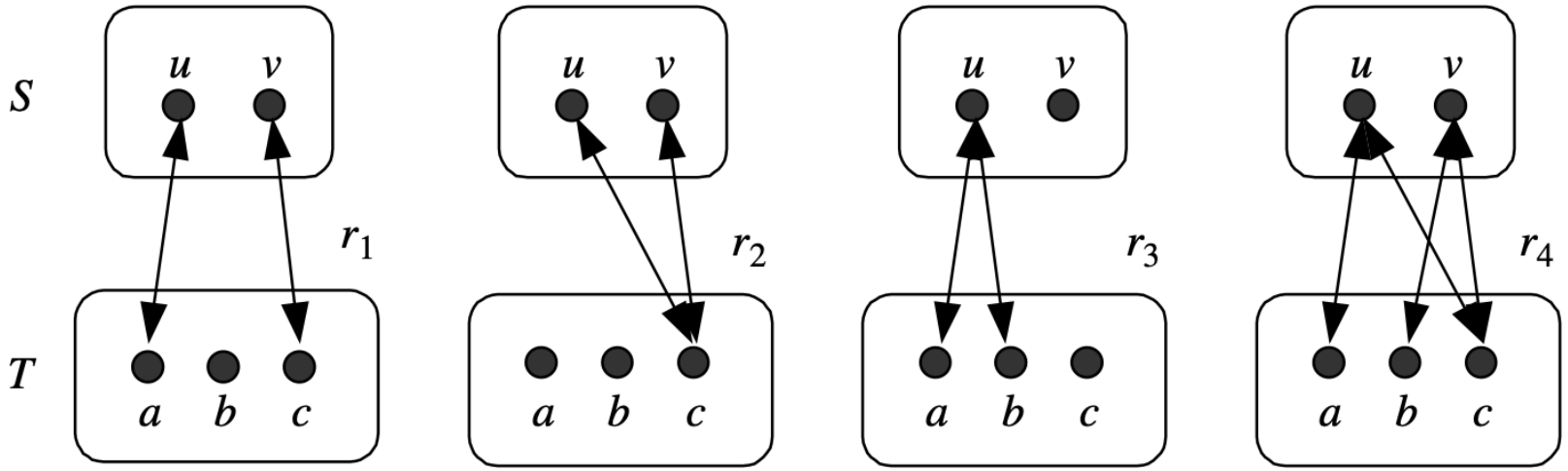
Mantıksal Programlama

- Zorunlu (imperative) ve fonksiyonel programlama kadar farklı paradigmaların bile ortak bir noktası vardır: *bir program girdileri okur ve çıktıları yazar*.
- Çıktılar işlevsel olarak girdilere bağlı olduğundan, zorunlu veya fonksiyonel bir program, girdilerden çıktılara bir eşleme (mapping) uygulamak olarak görülebilir.
- Bir mantıksal program ise bir ilişkiyi uygular (implement eder). İlişkiler (relations), eşlemelerden daha genel olduğundan, mantık programlama, zorunlu veya fonksiyonel programlamadan potansiyel olarak daha yüksek düzeydedir (high-level).

Mantıksal Programlama

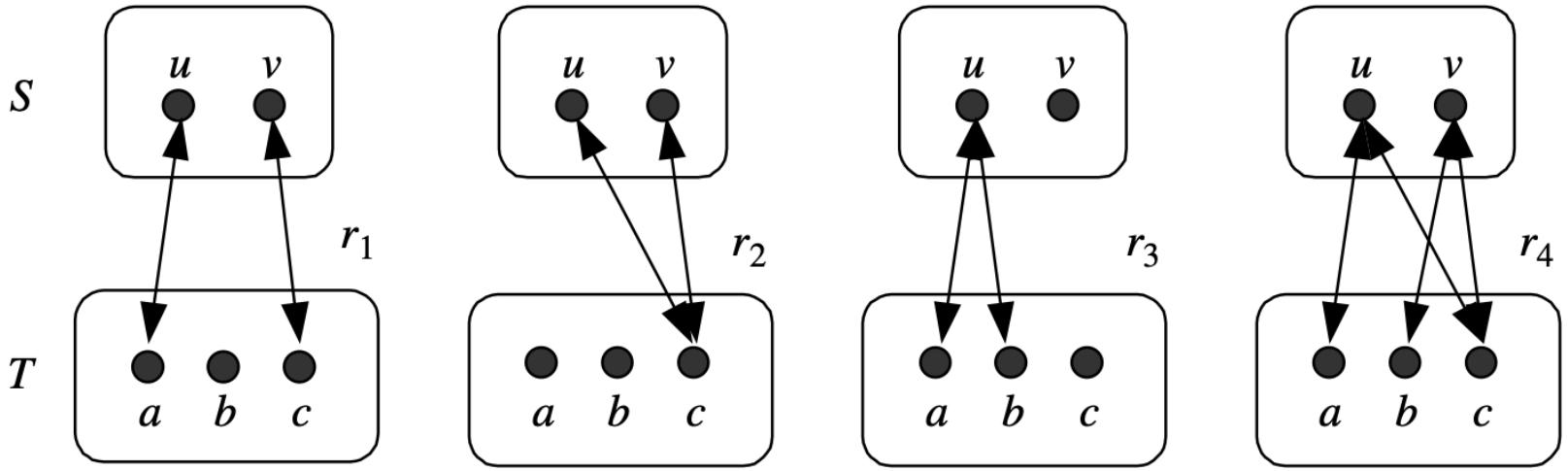
- S ve T iki değer kümesi olsun. Eğer S içindeki bütün x'ler ve T içindeki bütün y'ler için değeri doğru (True) veya yanlış (False) olan bir ***r***(x, y) mevcutsa, ***r***, S ve T arasında bir ilişkidir (relation) denir.
- Örneğin ">" işareti sayılar arasında bir ilişkidir. Çünkü sayı kümesindeki bütün x, y ikilileri için x > y ifadesi ya True ya da False olmak zorundadır.

Mantıksal Programlama



- Yukardaki şekildeki çift başlı oklar, ilişkinin doğru (True) olduğu her bir değer çiftini birbirine bağlar. Örneğin, r_1(u, a) ve r_1(v, c) doğrudur.
- S ve T arasındaki bir ilişkide, S'deki belirli bir değer T'deki birçok değerle ilişkili olabilir. Genel olarak, eşlemeler (mappings) çoktan bire (many-to-one), ilişkiler ise çoktan çoğadır (many-to-many).
- Imperative ve fonksiyonel programlama, esas olarak mapping'lerin uygulanmasıyla ilgilidir. Bir mapping ***m*** implement edildikten sonra, şu sorguyu yapabiliriz: a verildiğinde, m(a)'nın değerini belirleyin.

Mantıksal Programlama



- Mantıksal programlama, ilişkileri uygulamakla ilgilidir. Bir ***r*** ilişkisi implement edildikten sonra şu tarz sorgular yapılabilir:
  - a ve u verildiğinde, r(a, u)'nun doğru olup olmadığını belirleyin.
  - a verildiğinde, r(a, y) doğru olacak şekilde tüm y'leri bulun.
  - u verildiğinde, r(x, u) doğru olacak şekilde tüm x'leri bulun.
  - r(x, y) doğru olacak şekilde tüm x ve y'leri bulun.

İlişkiler (Relations)

- Point = point(Float × Float) *xy* düzlemindeki tüm noktaların kümesini temsil etsin.
- Ancak ve ancak p noktası orijinde ise doğru olan bir *orijin*(p) ilişkisi düşünün. Bu Point kümesinde tekli bir ilişkidir. Şu şekilde tanımlanabilir:
  - orijin(nokta(x, y)) ancak ve ancak x == 0 ve y == 0
- Şimdi *içeride*(p, r) ilişkisini ele alalım, bu ancak ve ancak p noktası orijinde merkezli r yarıçaplı çemberin içindeyse doğrudur. Bu, Point ve Float arasındaki ikili bir ilişkidir. Şu şekilde tanımlanabilir:
  - içeride(point(x, y), r) ***iff*** $x^2 + y^2 < r^2$

Prolog

- Prolog (**Pro**grammation en **Log**ique) bir mantık programlama dilidir.
- Prologdaki bütün veri yapılarına terim (Term) denir.
- Bir terim şunlardan oluşabilir:
  - **Atom** veya **sayı** olan bir **sabit** (constant).
  - Bir **değişken** (variable).
  - **Yapı**lar (structures).

ref: https://www.dai.ed.ac.uk/groups/ssp/bookpages/quickprolog/node5.html
ref: https://ocw.metu.edu.tr/course/view.php?id=103#section-1

Atomlar

- Tek tırnak içine alınmış herhangi bir şey (ör. 'Naber').
- Başında bir küçük harf bulunan herhangi bir karakter, sayı veya alt çizgi karakteri dizisi (örn. bu_bir_Atomdur).
- Herhangi bir sembol dizisi: +, -, *, /, \, ^, >, <, =, ', :, ., ?, @, #, $, &. (ör. ***+****+@).
- Özel atomlardan herhangi biri: **[]**, **{}**, **;**, **!**

ref: https://www.dai.ed.ac.uk/groups/ssp/bookpages/quickprolog/node5.html
ref: https://ocw.metu.edu.tr/course/view.php?id=103#section-1

Sayılar

- Bir sayı şunlardan biri olabilir:
  - Bir tamsayı (integer) (ör. 99).
  - Bir ondalıklı sayı (float) (ör. 99.91).
  - Bilimsel notasyonlu sayı (ör. 5e7)

Değişkenler

- Başında bir büyük harf olan herhangi bir karakter, sayı veya alt çizgi karakteri dizisi (örn. X, Hey_naber).
- Önünde bir alt çizgi olan herhangi bir karakter dizisi, sayı veya alt çizgi karakteri (ör. _hey_Ho).
- Kendi başına bir alt çizgi karakteri (ör. _). Kendi başına bulunan alt çizgiye anonim değişken denilir. Önemsiz (don't care) değerleri ifade etmek için kullanılır.

Yapılar

- Prolog'daki yapılar, birkaç bileşeni olan, ancak tek bir nesne olarak ele alınan basit nesnelerdir. C dilindeki struct gibi.
- Bir atom ile başlarlar.
- Devamında parantez içinde virgülle ayrılmış bir veya daha fazla argümanları olur.
- Argümanlar herhangi bir Prolog terimi olabilir.
- Örn:
  - tarih(12, aralik, 2022)
  - kisi(ali, dogum(2004, adana), kilo(80), a_RH_pos)

Dil Elemanları

- Bir Prolog programı, önerme (clause) veya yüklemlerden (predicate) oluşur.
- Birim önermeleri, bir nokta ile biten yapı veya atom terimleridir. Birim önermeleri gerçekler (facts) olarak kabul edilir.
  - buyuk(karpuz, elma).
- Birim olmayan önermeler (kurallar, rules) bir baş önerme (head clause) ve bir gövdeden (body) oluşur.
  - anneanne(X, Y) :- anne(X, Z), anne(Z, Y).
- Baş önermenin ispatı için gövdenin ispatı gerekir.
- Gövde virgülle (and) veya noktalı virgülle (or) ayrılmış yapılardan oluşur.
- Gövdedeki yapılar kanıtlanması gereken hedef (goal) önermelerdir.

ref: https://ocw.metu.edu.tr/course/view.php?id=103