Logikai programozás

és

szakértői rendszerek

Dr. Kusper Gábor

(előadás és gyakorlat)

Sümegi Adrienn

IV. évf.

I. félév

1. előadás IX. 13.

Prolog

I. imperatív

C, pascal, delphi, C#, java

objektum orientált

II. deklaratív

a) funkcionális ; lisp, sml, pascal, clean

b) logikai : prolog

I és II Különbsége

hogyan? → mit? (mit kell megoldani)
(hogyan oldjuk meg a problémát)

I.: • hozzá kell rendelni a változó címéhez

• megcsinálják az adatszerkezetet, a hangsúly: hogyan működjön

II.: • a programozó definiálja / deklarálja a problémát → a rendszer mondja meg, hogyan tovább

- Problémafüggő, hogy melyikben előnyebb programozni.

logikai probléma : prolog-ban előnyebb

II. van matematikai alapja

a)

b) matematikai logika → elsőrendű matematikai logika nyelve

II/a : alapegysége : a fgv

II/b : -"- : a predikátum
olyan fgv, amely I v. H értéket ad vissza, megfelel egy kérdésnek.

II: nincs mellékhatás
↳ az alprogram megváltoztatja a környezetét

alprogram környezet: globális változó, fájlok, képernyő ... stb

I program ~~nem~~ ✓ tud a képernyőre írni ⇒ mert nem csinálhat környezeti változást.

II/b: VISUAL PROLOG

- általában a főprogram tud a képernyőre írni

II/a: van értékadás

II/b: nincs értékadás

(• szekvencia, szelekció, iteráció, értékadás → minden általános célú program tudja)

- értékadás helyett mintaillesztés van
- onnan dönthető el, hogy az egyenlőség jel értékadás, akkor a bal oldal a változó, jobb oldal az érték, amit felvesz a változó, mintaillesztésnél nincs kitüntetett oldal
  ↳ x=5 v. 5=x ⇒ kérdés: az egyik illeszthető-e a másikra, v. illeszthetők egymásra?

Ajánlott irodalom:

Futó Iván: Mesterséges intelligencia

Nyékiné: Programozási nyelvek (fájl-ba)

Chin-Liang Chang: Symbolic Logic and Mechanical Theorem Proving

domains

predicates

clauses → program tudása az adott programról

goal

tények szabályok

predicates: deklarálju a predikátumokat (név és paraméterek típusa)

domains: típusdeklaráció

goal: kérdés

- A program a goalban adott kérdésre próbál válaszolni a tények és a szabályok alapján. A hogyan-t a program elrejti előlünk.

program:

```
domains
    ember = szimbol;
predicates
    fia(ember, ember);
clauses
    fia(eva, peter).
```

⇒ eva fia peter ⇒ ez egy TÉNY

prolog-ban amit nagy betűvel kezdünk, az változó, a kisbetűs a konstans.

```
fia(eva, X) :- X = peter
```

ez egy SZABÁLY

szabály feje — ha igaz „if" — szabály törzse

Ha x illeszkedik arra, hogy peter, ⇒ x fia evának.

AKKOR ← HA

ugyal

fia (éva, máté)

Érdés: Máté fia-e Évának. Nem, vagy előfordulhat, de akkor nem vezethető le az adott tudásbázisból.

fia (éva, péter) → levezethető, a válasz igen.

fia (x, péter) → Van-e olyan x, akinek a fia péter.

Yes, x = éva

fia (x, x) → ezt is meg lehet érdemi.

(Van-e olyan x, amelynek a fia x)

∃

- Minden olyan változó, amely a célban van, az egzisztenciális (∃) kvantor.

(Minden olyan x-re, ahol igaz a törzs, teljesül a fej.)

- Minden változó, amely a szabály fején van, univerzális kvantorral kötött. (Nem kell írni, és automatikus)

Kvantor pl.: ∀, ∃, Σ, ∫ → d kell

$\forall_x P(x,y) \Leftrightarrow Q(y)$ a lokális változó a kötött változó
a szabad változó a paraméter

Iteráció: if, case

clauses

fia (éva, péter)
fia (éva, máté)

pascalban:

function fia (x, y: ember): boolean

vagy is

fia := false;

```
if x = 'éva' then
    if y = 'péter' then
        fia := huc
    else if y = 'máté' then
        fia := huc;
end;
```

imperatív esetben a fgv-nek egy változata van, egy vagy változata, ebben az egyes esetet szelekciós utasításokkal válogatjuk szét.

deklaratív nyelvekben a fgv-nek több ilyen változata van, amelyek közötti mintaillesztéssel választjuk ki a megfelelő esetet.

esetválasztás mintaillesztéssel → így rövidebben

↳ magasabb absztrakciós szinten van

prologban van szelekció, ezt esetválasztás mintaillesztéssel nevezzük. (nincs → ez felel meg annak)

- minden, ami iterációval megoldható, megoldható rekurzióval is, és fordítva, így a két programozási eszköz egyenértékű.
- iteráció gyorsabb, nem kell verem

  rekurzió lassabb, tárigénye van veremre

  ↓

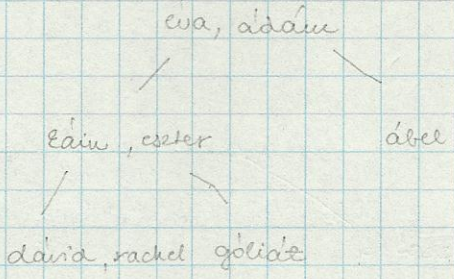
  az alprogram meghívja önmagát

  nem lehet végtelen → biztosítani kell a kilépési feltételt (bázisfeltétel), amely, ha teljesül, megáll a rekurzió

- függvényhívás esetén elmentjük a lokális változót és a regisztereket a veremben (annak előtte)
- rekurzió átalakítható ⇒ magasabb absztrakciós szintet képvisel, mint az iteráció

- deklarációs progr. nyelvben nincs for ciklus .. helyettük rekurzió van / rekurzív predikátum

Van olyan szabály, amelyiknek a fejében lévő predikátum megjelenik a szabály törzsében . → azt mondjuk, hogy a predikátum rekurzív



domains

emberek = symbol;

predicates

gyermeke (emberek, emberek);

ősei (emberek, emberek);

clauses

gyermeke (éáin, éva).

gyermeke (éáin, ádám).

gy (ábel, éva). gy (ábel, ádám).

gy (dávid, éáin). gy (dávid, eszter). gy (góliát, éáin).

gy (góliát, eszter).

① ősei (X, Y) :- gy (Y, X)

ősei (X, Y) :- gy (Y, Z), gy (Z, X)

↓ rekurzió áll, mert nem hívjuk, hanyadik szintű a fa.

X éva
|
Z éáin
|
Y dávid

helyette: őse(X,Y) :- gy(Y,Z), őse(X,Z)

megcserélhető (most farekurzív)

2. előadás

IX. 20.

A rekurzív predikátumoknak 2 változata van:

- egy rekurzív
- egy nem rekurzív

- A prolog a predikátumokat a felírás sorrendjében értékeli ki, ezért az alapesetnek meg kell előznie az általánosat.

DEF.: Azt mondjuk, hogy egy predikátum FARokREKURZÍV, ha a rekurzív változatára igaz az, hogy az önmagára hivatkozás a törzs utolsó eleme.

- Ami farekurzív predikátum, azzal abból a fordító ciklust készít. A farekurzió nem használ vermet.

őse(x,y) :- gy(x,z), gy(z,x).

x őse y-nak ⇔, ha ∃ olyan z, hogy y gyermeke z-nek és z gyermeke x-nek.

A változók mindig nagybetűvel kezdődnek. Ha a szabály törzsében van olyan változó, amely nem jelenik meg a szabály fejében, ⇒ azt a változót EGZISZTENCIÁLIS kvantor köti. (ilyen most a z.)

azokat a változókat, melyek benne vannak a szabály fejében UNIVERZÁLIS kvantor köti.

Nézzük meg azt a predikátumot, ami három szintű!

őse (x,y) :- gy (y, y1), gy (y1, y2), gy (y2, x).

Ezt nem lehet a végtelenségig folytatni → rekurzió kell!

őse (x,y) :- gy (y,z), őse (x,z) ;

Ez farokrekurzió, és ezt majd az ①-nél lévő állítja meg.

A prolog program logikai szemantikája

- A szintaktikai szabályok gyűjteménye, amely megmondja, hogy a szimbólumokat hogyan kell egymás mellé írni.
- A szemantikai hiba program futástörténi hibája.
- A prolog prog-ból készítek egy logikai kifejezést, ez hányadrendű?
  Ez elsőrendű logikai kifejezés
  0-adrendű nyelvnél van a konjunktív normál forma.
  (1. rendű nyelvnél van a prenex alak)

KNF: $f = f_1 + f_2 + \dots + f_n$ konj. norm. forma, ahol $n \geq 1$ és f_i literálok diszjunkciója.

DEF: Literálnak nevezzük valamely logikai változót és annak negáltját.

Literálok halmazát megkapjuk, ha az összes változó halmazát vesszük a változók negáltjainak halmazával unió jával.
A diszjunktív normál forma vagy-okkal van.