Programozás nyelvek 3
2. előadás

Logikai feladatok

"Portia ládikája" első feladata:

Portiának van három ládikája (arany, ezüst, ólom). Férjhez akar menni, de intelligens férjet szeretne magának. Ezért mielőtt a kérőnek igent mondana, próbára teszi észbeli képességét.

Elrejti egy képét valamelyik ládikába, és mindegyikre egy-egy állítást ír föl, amely a kép hollétére vonatkozik.

Majd hozzátesz segítségként egy "peremfeltételt", ami alapján már egyértelműen meg lehet találni a helyes választ.

Logikai feladatok

A konkrét feladvány részletei:

- Az arany ládika felirata: "ebben a ládikában van".
- Az ezüst ládika felirata: "nem ebben a ládikában van".
- Az ólom ládika felirata: "nem az arany ládikában van".

A peremfeltétel: "az állítások közül legfeljebb egy igaz".

Logikai feladatok

Ugyanez Prolog-ban:

```
ládán(arany,arany).
ládán(ezüst,arany).
ládán(ezüst,ólom).
ládán(ólom,ezüst).
ládán(ólom,ólom).
```

vagy

```
ládán(ezüst,X) ha X='arany' vagy X='ólom'.
ládán(ólom,X) ha X='ezüst' vagy X='ólom'.
```

Logikai feladatok

A mesének szó szerint megfelelően:

```
ládán(arany,arany).
ládán(ezüst,X) ha láda(X) és nem(X='ezüst').
ládán(ólom,X) ha láda(X) és nem(X='arany').
láda(X) ha X='arany' vagy X='ezüst' vagy
           X='ólom'.
```

Más tények vagy szabályok is lehetnének, pl.:

```
arany(arany).  ezüst(arany).  ezüst(ólom).
ólom(ezüst).  ólom(ólom).

ezüst(X) ha láda(X) és nem(X='ezüst').
ólom(X) ha láda(X) és nem(X='arany').
```

Logikai feladatok

Ugyanez Prolog-ban:

```
egy_igaz(X) ha ládán(arany,X) és
    nem(ládán(ezüst,X)) és nem(ládán(ólom,X))
  vagy ládán(ezüst,X) és
    nem(ládán(ólom,X)) és nem(ládán(arany,X))
  vagy ládán(ólom,X) és
    nem(ládán(arany,X)) és nem(ládán(ezüst,X)).
mind_hamis(X) Ha láda(X) és nem(ládán(arany,X))
 és nem(ládán(ezüst,X)) és nem(ládán(ólom,X)).
legfeljebb_1_igaz(X) ha egy_igaz(X) vagy
                        mind_hamis(X).
```

Logikai feladatok

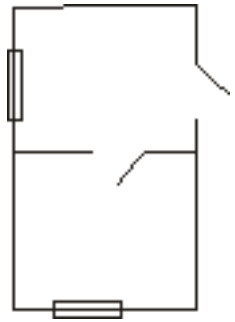
Ugyanez a másfajta tényekkel egyszerűbben :

```
egy_igaz(X) ha
        arany(X) és nem(ezüst(X)) és nem(ólom(X))
   vagy ezüst(X) és nem(ólom(X)) és nem(arany(X))
   vagy ólom(X) és nem(arany(X)) és nem(ezüst(X)).
mind_hamis(X) ha láda(X) és nem(arany(X))
                  és nem(ezüst(X)) és nem(ólom(X)).
legfeljebb_1_igaz(X) ha egy_igaz(X) vagy
                        mind_hamis(X).
```

Lakástervezés

Lakás szempontok:

- 2 szoba legyen
- az egyiken legyen bejárati ajtó
- mindkettőn legyen ablak
- legyen közöttük átjáró ajtó
- egy falon egy dolog (ajtó vagy ablak) lehet



Kiegészítő szempontok:

- semmi ne nézzen északra
- a két ablak ne szemben levő falon legyen

Lakástervezés

Tények:

```
irány(észak).          szemben(észak,dél).
irány(dél).            szemben(kelet,nyugat).
irány(kelet).          szemben(dél,észak).
irány(nyugat).         szemben(nyugat,kelet).
```

Lakás:

```
lakás(Be,Ajtó,Ablak1,Ablak2) ha
   bejáratos_szoba(Be,Ajtó,Ablak1) és
   szoba(Ajtó2,Ablak2) és szemben(Ajtó,Ajtó2)
   és nem(szemben(Ablak1,Ablak2)).
```

Lakástervezés

Lakás:

Hatékonysági szempontok – ha a az ajtó az egyik szobából nem ellenkező irányban van a másik szobából, akkor a másik szoba biztos nem jó, felesleges olyanokat vizsgálni:

```
lakás(Be,Ajtó,Ablak1,Ablak2) ha
    bejáratos_szoba(Be,Ajtó,Ablak1) és
    szemben(Ajtó,Ajtó2) és szoba(Ajtó2,Ablak2)
    és nem(szemben(Ablak1,Ablak2)).
```

Lakástervezés

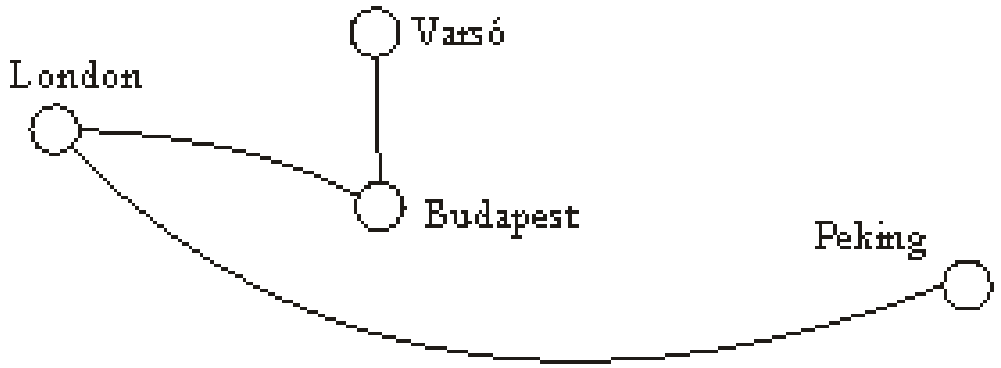
Szobák:

```
bejáratos_szoba(Be,Ajtó,Ablak) ha
   szoba(Ajtó,Ablak) és irány(Be)
   és nem(Be=észak) és nem(Be=Ajtó)
   és nem(Be=Ablak).
szoba(Ajtó,Ablak) ha
   irány(Ajtó) és irány(Ablak)
   és nem(Ablak=észak) és nem(Ajtó=Ablak).
```

Prolog feladattípusok

Feladat: ismerünk városok közötti repülőgépjáratokat, amelyek alapján el kellene döntenünk, hogy egy városból eljuthatunk-e egy másik városba, esetleg átszállásokkal.

Felhasználható tudás: a repülőgépjáratok oda-vissza közlekednek.



Prolog feladattípusok

A térkép tárolása tényekkel és szabályokkal (a szimmetria miatt):

```
út(budapest,varsó).
út(budapest,london).
út(london,peking).
van_út(X,Y) ha út(X,Y) vagy út(Y,X).
```

Prolog feladattípusok

Megoldási ötlet: vagy van közvetlen járat, vagy megpróbálunk átszállással eljutni (mint az őse szabálynál).

```
mehetünk(X,Y) ha van_út(X,Y) vagy
  van_út(X,Z) és mehetünk(Z,Y).
```

Probléma: végtelen ciklusba kerülünk.

Az oka: a repülőgép hálózatot leíró gráf irányítatlan. (Az ősöket leíró gráf irányított és körmentes volt.)

Megoldás: kellenének változók azon városok tárolására, ahol már jártunk.

Prolog feladattípusok

Megoldási elv: a Prolog program képes futás közben változtatni önmagán, új tényeket, szabályokat vehetünk fel, korábbiakat törölhetünk.

Az ehhez kapcsolódó műveletek:

```
assert(tény vagy szabály).
asserta(tény vagy szabály).
assertz(tény vagy szabály).
retract(tény vagy szabály feje).
retractall(tény vagy szabály feje).
```

Prolog feladattípusok

Megoldási ötlet: vagy van közvetlen járat, vagy megpróbálunk átszállással eljutni (mint az őse szabálynál).

```
mehetünk(X,Y) ha van_út(X,Y) vagy
  van_út(X,Z) és szabad(Z) és mehetünk(Z,Y).
szabad(Z) ha voltunk(Z) és ! és fail
             vagy asserta(voltunk(Z)).
```

Ha az adatbázisban már van `voltunk(Z)` tény, akkor oda ne menjünk még egyszer; ha nincs, akkor vegyük fel és menjünk is el oda!

Prolog feladattípusok

Újabb problémák:

- A "varsói" kitérőt megtesszük.
- Másodszori kérdésre nem kapunk választ.

Megoldás egy főprogram, amely kitakarítja a korábbi információkat és a kezdő városba visszalépést nem engedi:

```
kérdés(X,Y) ha retractall(voltunk(_)) és
  assert(voltunk(X)) és mehetünk(X,Y).
```

Prolog feladattípusok

Adott tények megszámolása (gyerekek száma):

```
szülője(anna,antal).
szülője(anna,andrea).
szülője(andrea,András).
darab(X,A) ha szülője(X,B) és szabad(B) és
    darab(X,C) és A=C+1 vagy A=0.
szabad(B) ha volt(B) és ! és fail
    vagy asserta(volt(B)).
```

Ez csak akkor működik, ha a gyerekek neve különböző!

Egyes Prolog verziókban:

```
A is C+1
```

Prolog feladattípusok

Adott tények összegzése (termékek árát tároljuk):

```
ára(kávé,1000).
ára(tea,800).
ára(kakaó,1000).
összeg(A) ha ára(X,B) és szabad(X) és
    összeg(C) és A=C+B vagy A=0.
szabad(X) ha volt(X) és ! és fail
    vagy asserta(volt(X)).
```

Ez csak akkor működik, ha a termékek neve különböző!

Egyes Prolog verziókban:

```
A is C+B
```

Prolog feladattípusok

Adott tények számának maximuma (legtöbb gyerekes szülő):

```
szülője(anna,antal).
szülője(anna,andrea).
szülője(andrea,András).
darab(X,A) ha szülője(X,B) és szabad(B) és
   darab(X,C) és A=C+1 vagy A=0.
szabad(B) ha volt(B) és ! és fail
   vagy asserta(volt(B)).
legtöbb(X) ha darab(X,A) és nem(vantöbb(A)).
vantöbb(A) ha darab(_,B) és B>A.
```

Egyes Prolog verziókban:

```
A is C+1
```

Programozás nyelvek 3
2. előadás vége