Programozás nyelvek 3
1. előadás

Prolog

Alapelve

- Programozás a matematikai logika segítségével
- Egy logikai program logikai állítások halmaza
- Egy logikai program futása egy következtetési folyamat, azaz a logikai formula kiértékelése
- Azt kell megmondani, hogy mit akarunk megoldani (a hogyannal nem kell törődni)

Prolog

Története

- 1972 – Marseille (PROgramming in LOGic)
- 1975 – NIM IGÜSZI
- 1978 – PROLOG fordító
- 1981 – A japán 5. generációs projekt a logikai programozást választja
- 1982 – MPROLOG, TPROLOG
- 1985 – TC PROLOG
- 198x – Turbo PROLOG
- 1984 – mikroPROLOG

Prolog

Története

- 1986 – Prolog szabványosítás kezdete
- 1987–89 – Új logikai programozási nyelvek (CLP, Gödel stb.)
- 1989 – CS PROLOG
- 1990 – SICStus Prolog

A legfontosabb Prolog megvalósítások:

- SWI Prolog: http://www.swi-prolog.org/
- SICStus Prolog: http://www.sics.se/sicstus
- GNU Prolog: http://pauillac.inria.fr/~diaz/gnu- prolog/

Prolog

A Prolog program elemei

Tények = azonosan igaz formulák

Példa:

```
szülője(András,anna).
szülője(András,andrea).
szülője(anna,antal).
```

Egy hagyományos értelemben vett eljárás (konstans tömb) azon tények összessége, amelyek neve és paraméterszáma azonos.

A Prolog program elemei

Szabályok (következtetési szabályok)

Példa:

```
nagyszülője(X,Y) ha szülője(X,Z) és
                    szülője(Z,Y).
szülője(X,Y) ha anyja(X,Y) vagy apja(X,Y).
```

A szabályokban az **és**, illetve a **vagy** műveletek nem szimmetrikusak!

Prolog

A Prolog program elemei

Kérdések – a program utasításai (tények, szabályok) között keressünk a kérdésnek megfelelő(ke)t!

Példa:

```
szülője(András,anna)? → igaz
szülője(anna,melinda)? → hamis
szülője(András,X)? → igaz, ha X=anna (X=andrea)
szülője(X,antal)? → igaz, ha X=anna
szülője(X,Y)? → igaz ha X=András és Y=anna
                (+ további megoldások)
```

Megfelelő = a szabad paraméterek behelyettesítendők, a kötöttek egyezzenek meg – mintaillesztés!

Prolog

A kérdések megválaszolásához a tények és szabályok között keresünk a leírás sorrendjében a kérdéssel össze-illeszthetőt!

Összeilleszthető (egyesíthető), ha

- a nevük és paraméterszámuk azonos,
- az értékkel rendelkező paramétereik azonosak,
- a szabad (még ki nem töltött) változó paramétereiket pedig behelyettesítjük, hogy egyformák legyenek.

Prolog

Ha egy tény (vagy szabály) nem feleltethető meg a kérdésnek, akkor a sorrendben következővel folytatjuk.

Ha megfeleltethető, akkor megkapjuk, hogy milyen változó-paraméter beállítással felelnek meg egymásnak.

Ezután megtörténhet az összes megoldás megkeresése is.

Prolog

Példa:

```
nagyszülője(András,Antal)?
⇒ szülője(András,Z) és szülője(Z,Antal)?
1. szülője(András,Z)? → igaz, ha Z=Anna
2. szülője(Anna,Antal)? → igaz
⇒ nagyszülője(András,Antal)? → igaz
```

Kérdés: mi történne, ha a tényeket más sorrendben tárolnánk?

```
szülője(András,Andrea).
szülője(András,Anna).
szülője(Anna,Antal).
```

Prolog

Példa:

```
nagyszülője(András,Antal)?
⇒ szülője(András,Z) és szülője(Z,Antal)?
1. szülője(András,Z)? → igaz, ha Z=Andrea
2. szülője(Andrea,Antal)? → hamis
1. újra: szülője(András,Z)? → igaz, ha Z=Anna
2. szülője(Anna,Antal)? → igaz
⇒ nagyszülője(Anna,Antal)? → igaz
```

Megoldási stratégia: visszalépéses keresés

Prolog

Írásmód:

	Kulcsszavas	Írásjeles
ha	if	:-
és	and	,
vagy	or	;

Prolog

Alternatívák másképp:

```
szülője(X,Y) ha anyja(X,Y) vagy apja(X,Y).

szülője(X,Y) ha anyja(X,Y).
szülője(X,Y) ha apja(X,Y).
```

Értelmezés:

- akkor megy a **vagy** jobboldalára, ha a baloldalra nem kapott megoldást;
- akkor megy a második szabályra, ha az elsőre nem kapott megoldást;

Prolog

Ciklus helyett rekurzív formula:

```
őse(X,Y) ha szülője(X,Y) vagy
            szülője(X,Z) és őse(Z,Y).
```

Ha egy paraméter értékére nincs szükségünk:

```
szülő(X) ha szülője(X,_).
```

Ha valamiből csak egy megoldás kell:

```
egyszülő(X) ha szülő(X) és !.
```

A vágás(!) értelmezése:

- előre haladva azonosan igaz;
- visszafelé haladva hamissal fejezi be a szabályt.

Prolog

Ciklus helyett rekurzív formula:

```
őse(X,Y) ha szülője(X,Y) vagy
            szülője(X,Z) és őse(Z,Y).
```

Ha egy paraméter értékére nincs szükségünk:

```
szülő(X) ha szülője(X,_).
```

Ha valamiből csak egy megoldás kell:

```
egyszülő(X) ha szülő(X) és !.
```

A vágás(!) értelmezése:

- előre haladva azonosan igaz;
- visszafelé haladva hamissal fejezi be a szabályt.

Prolog

A ! egyéb használata: hiányzó feltételek pótlása:

```
absz(X,Y) ha X>=0 és X=Y vagy Y=-X.
```

Kérdések:

```
absz(5,5)? → igaz
absz(-5,5)? → igaz
absz(5,-5)? → igaz?????
```

Probléma: ha a második feltételrész hamis, akkor is a vagy túloldalára megy.

Helyes megoldás lehet:

```
absz(X,Y) ha X>=0 és ! és X=Y vagy Y=-X.
```

Prolog feladattípusok

Ha valamiből csak egy megoldás kell:

```
egy_szülő(X) ha szülő(X) és !.
```

Ha valamiből az összes megoldás kell:

```
összes_szülő ha szülő(A) és write(A,' ')
                és fail vagy succeed.
```

Magyarázat:

- fail – azonosan hamis logikai formula
- succeed (néha: true) – azonosan igaz logikai formula
- write – kiírás

Prolog feladattípusok

Adattípusok:

- számkonstansok – 1, -3, 2.7, 3.1e5
- szövegkonstansok –'István', 'Szent István', istvan
- műveleti jelek – +, -, *, /, <, =, >
- struktúranevek – neve(istvan)

Prolog feladattípusok

Adott tulajdonsággal nem rendelkező:

```
nem_szülő(A) ha valami(A) és nem(szülő(A)).
valami(A) ha szülője(A,_) vagy szülője(_,A).
```

Rákérdezés valami egyszerességére:

```
egyszeres_szülő(A) ha szülő(A) és
                      nem(többszörös_szülő(A)).
többszörös_szülő(A) ha szülője(A,X) és
   szülője(A,Y) és X<>Y.
```

Prolog feladattípusok

Legalább kétszeres:

```
kétszeres_szülő(A) ha szülője(A,X) és
                      szülője(A,Y) és X<>Y.
```

Pontosan kétszeres:

```
pont2_szülő(A) ha van_két_gyerek(A,X,Y) és
   nem(többszülő(A,X,Y)).
többszülő(A,X,Y) ha szülője(A,Z) és Z<>X és Z<>Y.
van_két_gyerek(A,X,Y) ha szülője(A,X) és
   szülője(A,Y) és X<>Y.
```

Prolog feladattípusok

Rákérdezés valami összességére:

```
összes_szülő(A) ha szülő(A) és
                   nem(vanmás_szülő(A)).
vanmás_szülő(A) ha szülő(B) és A<>B.
```

Rákérdezés a legnagyobbra:

```
max_korú(Y,K) ha kora(Y,K) és
                 nem(vannagyobb_korú(K)).
vannagyobb_korú(K) ha kora(_,L) és L>K.
kora(…,…).
```

Programozás nyelvek 3
1. előadás vége