PROLOG
PROgramování v LOGice

Olga Štěpánková

Programovací jazyky

„Jak?“ x **„Co?“**

se má udělat.

- **Procedurální** (imperativní)
  - Pascal, C++
  - Java, …
  - „JAK“
- **Deklarativní**
  - Prolog
  - CLP, …
  - „CO“

Proměnná

Adresovatelné místo v paměti

Význam jako v matematice

Krátká historie

- Začátek 70tých let – Edinburgh (Kowalski, M. van Emden), Marseilles (Colmerauer)
- Polovina 70tých let – Edinburgh (Warren – efektivní implementace)
- 90tá léta – Constraint Logic Programming

- viz Programovací prostředky pro UI (Štěpánková, Csontó) 8. kapitola v Mařík a spol.: *Umělá inteligence* (2), str. 257-306, Academia 1997

Terminologické opakování

- **konstanta** – 7, václav, vozidlo
- **proměnná** – X
- **predikát** – vyjadřuje nějaký vztah (relaci) – větší(X,Y)
- **arita** – počet argumentů predikátu nebo funkce– větší/2
- **atomická formule** – nejjednodušší zápis užívající jeden predikát, kterému lze přiřadit pravdivostní hodnotu
- **literál** – atomická formule nebo její negace
- **klauzule** – konečná disjunkce literálů, speciální případ **formule**
- **teorie** – množina formulí (předpokládáme, že platí všechny současně a společně charakterizují vlastnosti světa, o němž uvažujeme)
- **unifikace** – „účelová" náhrada volných proměnných (např. konstantou) tak, aby se 2 výrazy (formule) ztotožnily
- **rezoluce** – metoda hledání sporu v konečné množině klauzulí

¬ matka(M,v)

¬ zena(X) V ¬ rodic(X,Y) V matka(X,Y).

¬ zena(M) V ¬ rodic(M,v)

M/lud

zena(lud)

zena(dra)

M/lud

¬ rodic(lud,v)

¬ rodic(dra,v)

rodic(dra,v)

rodic(ludmila, vratislav).
rodic(drahomira, vaclav).
rodic(drahomira, boleslav).
rodic(vratislav, vaclav).
rodic(vratislav, boleslav).
zena(ludmila). **zena(drahomira).**
muz(vaclav). **muz(boleslav).**
(zena(X) & rodic(X,Y)) → matka(X,Y).
¬ zena(X) V ¬ rodic(X,Y) V matka(X,Y).
Můžeme z těchto informací zjistit, kdo je Václavova matka?
Dokažte, ze ∃ **M matka(M, vaclav).**
Použijte resoluci, tedy ověřte, že doplněním **¬ ∃ M matka(M, vaclav)** k výchozím klauzulím vznikne sporná množina.

Princip prog. jazyka Prolog

Program

VSTUP → **Systém automatic-kého dokazování** → **VÝSTUP**

Program jako logická teorie

- **PROGRAM je vlastně teorie v logice 1.řádu**
- Schopnost programu získat požadovanou informaci závisí na použitém algoritmu pro automatické dokazování
- **PROLOG** používá **SLD rezoluci** (**S**election rule driven **L**inear resolution for **D**efinite clauses), cílová orientace na dokazovaný vztah, t.j. **relace**
  - *viz* Řešení a dokazování vět (Štěpánková, Štěpánek), 3. kapitola v **Mařík a spol.: Umělá inteligence (1),** str. 67-98, Academia 1993
- Definite clause = Hornova klauzule, t.j. klauzule, ve které se vyskytuje **nanejvýš 1 pozitivní literál**

Program v Prologu tvoří 2 typy formulí

- **Fakt** – elementární **bezpodmínečně pravdivé tvrzení** vyjádřené jako 1 atom
  - rodič (ludmila, václav)**.**
- **Pravidlo** – **podmíněné tvrzení** „jestliže jsou splněny současně všechny podmínky tvořící tělo pravidla, pak platí i hlava p. (hlava je vždy popsána jediným atomem)"
  - v Prologu **A:-B1,...,Bk.**
    A je **hlava**, seznam **B1,...,Bk** tvoří **tělo**
  - matka(X,Y) :-rodič(X,Y), žena(X)**.**
  - fibonacci(D,Y) :- C1 is D-1, C2 is D-2, fibonacci(C1,X1), fibonacci(C2,X2), Y is X1+X2 **.**
    rekursivní pravidlo

Každá formule musí být ukončena tečkou !

Spouštění program v Prologu

- Program se spouští pomocí **dotazu**, který se formuluje pomocí atomických formulí v jazyce tohoto programu
  - ?- podcíl1, podcíl2,…, podcíl-n.
  - ?- rodič(X,václav), rodič(X,boleslav).

- Význam dotazu:

  „***Existuje*** *taková* ***substituce za použité proměnné****, že konjunkce* (podcíl1 & podcíl2 & ... & podcíl-n) *je důsledkem faktů a pravidel, která tvoří program?“.*

  Například „*Existuje v databázi českých králů osoba* X, *která je současně rodičem václava i boleslava?*“

Příklad

- Zdrojový kód "**rodič**"

```
% komentář příbuzenské vztahy
rodic(drahomira, vaclav).        rodic(drahomira, boleslav).
zena(ludmila).                   zena(drahomira).
muz(vaclav).                     muz(boleslav).
matka(X,Y) :- zena(X), rodic(X,Y).
```

- Tento program lze využít pro zodpovězení řady dotazů:
  - „*Má Václav nějaké děti*?“ ?- rodic(vaclav,X).
  - „*Platí, že Ludmila je matkou Boleslava*?“ ?- matka(ludmila,boleslav).
  - „*Jsou Václav a Boleslav sourozenci? Mají společného rodiče*?“ ?- rodic(Y,václav), rodic(Y,boleslav).
  - …

Prolog předpokládá **uzavřenost světa**, tj. **zná** jen ty objekty, jejichž existence je nutným důsledkem programu! Např. na dotaz „*Kdo je matkou Ludmily*?“ nebo přesněji „*Existuje někdo, kdo je matkou L.?*“ tj. ?-matka(X,ludmila). Prolog odpovídá **No**.

Aritmetika v Prologu

Máme k dispozici aritmetické operace, pomocí nichž vznikají termy. S termy Prolog zachází jako s výrazy, které upravuje jen pomocí substituce za proměnné. Výpočet hodnoty termu se neprovádí automaticky, ale až na **pokyn**.

- běžné aritmetické operace
  - Sčítání +, odčítání -, násobení *, dělení /
- aritmetické predikáty představují **pokyn** k vyhodnocení
  - **=:=** označuje „rovnost“: platí, mají-li oba argumenty po vyhodnocení stejnou vypočtenou hodnotu
  - **is** přiřazení, X is 2+3, pravá strana nesmí obsahovat volné proměnné
  - < , <= , >= , >

POZOR na predikát =
= neoznačuje aritmetický predikát, ale **pokyn** k unifikaci obou termů

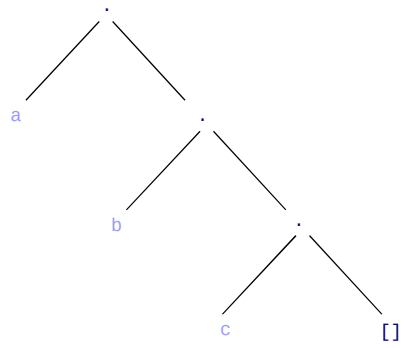
Výhody užití jazyka logiky 1.řádu (1)

- **Strukturování informace** a přímý přístup k odpovídajícím datům
- Příklad **rodina**

```
family(
person(JménoM, PrijmeníM, narozen(DM,MM,RM), pracuje(KdeM,PlatM)),
person(JménoZ, PrijmeníZ, narozen(DZ,MZ, RZ), pracuje(KdeZ,PlatZ)),
Seznam_dětí).
```

- „Najděte rodinu, ve které je manžel o 10 let mladší než manželka.“
  - ?- family(person(_,PrijmeníM,narozen(_,_,RM),pracuje(_,_)), person(_,PrijmeníZ, narozen(_,_,RZ), pracuje(_,_)), Seznam_deti), RM > RZ + 10.

Zápis seznamu



Několik ekvivalentních zápisů seznamu :

- .(a,.(b,.(c,[])))
- [a|[b|[c|[]]]]
- [a|[b|[c]]]
- [a|[b,c]]
- [a,b,c]
- [a,b|[c]]
- ...

Seznam libovolné délky:	list([]). list([First|Rest]):-list(Rest).
Seznam sudé délky:	evenlist([]). evenlist([First,Second|Rest]):-evenlist(Rest).

Výhody užití jazyka logiky 1.řádu (2)

- **Definice vztahu (relace) má nejen deklarativní (popisný) význam**, ale i výkonnou funkci díky procedurální interpretaci LP
- Příklad: **kryptogram**

```
  JA
  SE
 MAM
```

```
reseni(J,A,S,E,M) :- cislice(J), ..., cislice(M),
      10*J + A +10*S + E =:= 100*M + 10*A + M.
cislice(0).  cislice(1).  cislice(2).  cislice(3).  cislice(4).
cislice(5).  cislice(6).  cislice(7).  cislice(8).  cislice(9).
```

  - **?- reseni(J,A,S,E,M).** J=0, A=1, S=0, E=9, M=0 ….
  - **?- reseni(J,A,S,E,M), M>0.** J=1, A=0, S=9, E=1, M=1 ….
- Příklad: **definice nejmenšího prvku v seznamu**

Výhody užití jazyka logiky 1.řádu (3)

- Mizí pevná hranice mezi vstupními a výstupními proměnnými - tentýž program může být využit pro realizace nějaké funkce i odpovídající inverzní funkce: **Invertibilita logických programů**.
- Příklad 1: **rodič**

  ?- matka(ludmila,X).

  ?- matka(X,vaclav).

- Příklad 2: program **spojeni**(S1,S2, Vysledek) na spojení dvou seznamů do jediného tak, že výsledek tvoří seznam vzniklý zařazením prvků za sebe, např. platí

  ?- spojeni([1,2,3],[4,5],X). Yes, X=[1,2,3,4,5]

  ?- spojeni([1,2,3],Z,[1,2,3,4,5]) Yes, Z=[4,5].

  ?- spojeni(X, Z, [1,2,3,4,5]). Yes, X=[], Y=[1,2,3,4,5]

  Yes, X=[1], Y=[2,3,4,5]

  Yes, X=[1,2], Y=[3,4,5], ...

Příklad 3: **žárovky**

```
% Zapis souboru "rozhodovacich pravidel" pro ovladaci panel se 3 zarovkami
% predstavujimi vstup a jednim ridicim elementem (0-1) ve formatu
% r(Control_dec,Bulb1,Bulb2,Bulb3). Pro pripraveny soubor pravidel zjistete
/* 1. Pro ktere kombinace zarovek je rozhodnuti "1"?
2. Jake je rozhodnuti pro stav zarovek (0,0,0)? Proc?
3. Pripraveny soubor pravidel neni deterministicky – napiste dotaz, který najde
   kombinaci vstupnich hodnot, pro kterou pravidla nabizeji 2 ruzne vystupy!
4. Pozmente pouzitou databazi a overte, ze i v tomto pripade postup pouzity v
   bode 3 funguje!
5. Zjistete, zda taz hodnota B1 a B3 vzdy vede k rozhodnuti „1“! Overte opet na
   pozmenenem souboru.
6. Napiste pravidlo, ktere overi, zda je pro rozhodnuti dulezita hodnota
   zarovky 2? (ruznost 2 vyrazu V1, V2 zkontrolujte napr. not V1=V2) */
r(1,1,1,1).  r(0,1,1,0).  r(0,0,1,1).  r(0,0,1,0).
r(1,1,0,1).  r(1,0,0,1).  r(1,1,1,0).  r(1,0,1,0).
s(0).        s(1).
```

Souhrn

Prolog má velmi jednoduchou syntaktickou stavbu:

Objekty:
- konstanty, proměnné a složené termy označují objekty
- Proměnné vždy začínají velkým písmenem
- funkční symboly slouží pro pojmenování (začínají proto také malým písmenem), ale nikdy nejsou přímo vyhodnocovány

Vztahy:
- predikáty vyjadřují vztahy mezi objekty
- Klauzule popisují pravdivá tvrzení
  Proměnné v různých klauzulích programu jsou na sobě zcela vzájemně nezávislé

Při **vyhodnocování dotazu** se postupuje tak, že se hledá klauzule, jejíž „hlavu“ lze s dotazem unifikovat:
- Takových klauzulí může být víc – odpovídání na dotaz lze tedy chápat jako proces hledání
- Na dotaz můžeme dostat 0, 1 či více odpovědí.
- Při formulaci dotazu obvykle není předem dán vzor „vstupně výstupních parametru“ – invertibilita logických programů.

Literatura

- Mařík a kol. *Umělá inteligence* (2), Academia (1997)
- Polák J. : *Prolog*, Grada (1992)
- P. Jirků, P. Štěpánek, O. Štěpánková: *Programování v jazyku Prolog*, SNTL 1991
- Peter Flach: *Simply Logical - Intelligent Reasoning by Example*, John Wiley & Sons 1994, 2007
- ...
- viz web předmětu