Neprocedurální programování

Prolog 1

Úvod do logického programování

Robert Kowalski
*1941

Alain Colmerauer
1941 - 2017

Co je to „neprocedurální programování“ ?

Procedurální (imperativní) programování

- Python, C, C#, Java, ...
- přiřazovací příkaz
- popisujeme, jak úlohu vyřešit

Neprocedurální (deklarativní) programování

- programování bez přiřazovacího příkazu

Neprocedurální programování

Logické programování

- popisujeme problém, který chceme řešit
- prostředky matematické logiky
- Prolog - Programmation en Logique

Funkcionální programování

- program = definice funkcí
- výpočet = aplikace funkce na argumenty
  - » skládání funkcí
  - » “matematické” funkce bez vedlejších efektů
- LISP - List Processing
- Haskell - Haskell Curry

Programmation en Logique

1971 Robert Kowalski (Edinburgh)
Alain Colmerauer (Marseille)

1972 první interpret Prologu

- A. Colmerauer, Philippe Roussel
- 1. program v Prologu = francouzský QA systém

1977 David Warren (Edinburgh)

- kompilátor Prologu
- edinburgský dialekt

1983 Warren Abstract Machine (WAM)

- architektura paměti, instrukční sada
- standardní cíl kompilátorů Prologu

Prolog : Historie

1995 ISO Prolog standard

(1995) ISO/IEC 13211-1 *General Core* [html]

(2000) ISO/IEC 13211-2 *Modules* [html]

Prolog : Aplikace

Výuka a výzkum

Zpracování přirozeného jazyka

AI, automatické dokazování vět

Webové aplikace, sémantický web [github]

Deduktivní databáze – Datalog [NDBI050]

Deklarativní OOP – Logtalk [web]

Programování s omezujícími podmínkami [NAIL120] [NOPT042]

- SICStus Prolog [web]
- Picat [web]

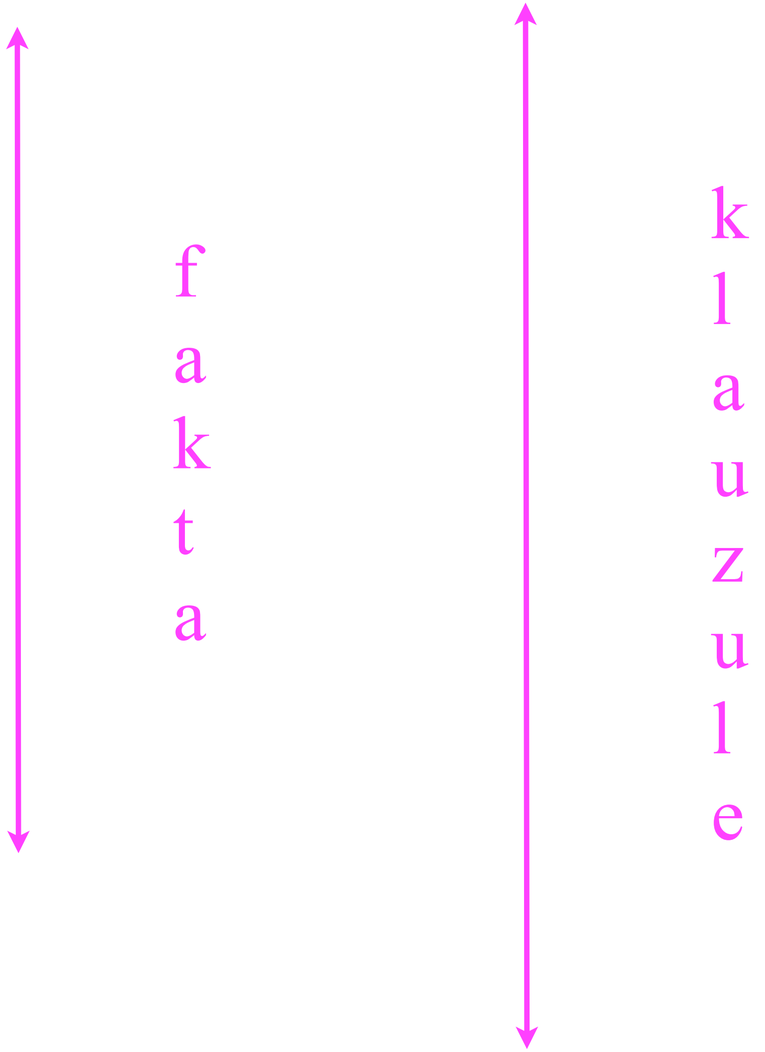
AnsProlog [wiki], procedurální generování obsahu [pdf]

Jednoduchý program v Prologu

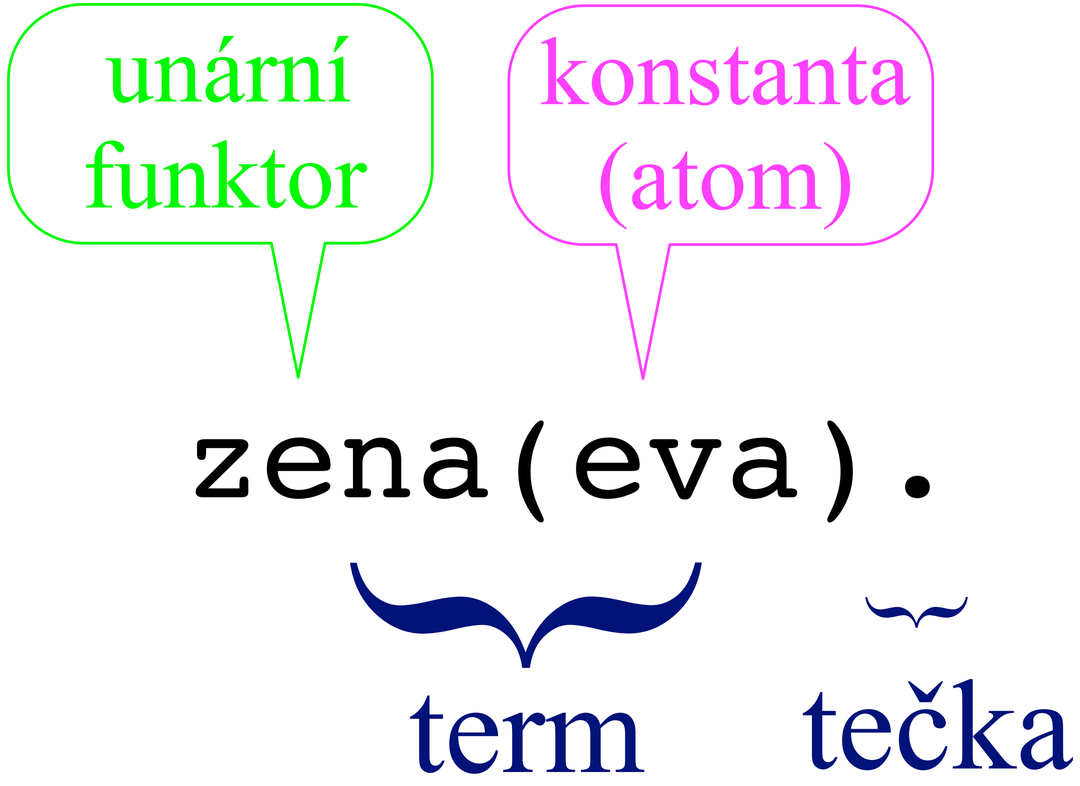
```
zena(eva).                % Eva je žena.
muz(adam).                % Adam je muž.
muz(kain).                % Kain je muž.
muz(abel).                % Abel je muž.
rodic(eva,kain).          % Eva a Adam jsou
rodic(adam,kain).         % rodiči Kaina.
rodic(eva,abel).          % Oba jsou i
rodic(adam,abel).         % rodiči Abela.
```

Jednoduchý program v Prologu

```
zena(eva).
muz(adam).
muz(kain).
muz(abel).
rodic(eva,kain).
rodic(adam,kain).
rodic(eva,abel).
rodic(adam,abel).
```



Fakta v Prologu



Jednoduchý program v Prologu

```
zena(eva).
muz(adam).
muz(kain).
muz(abel).
rodic(eva,kain).
rodic(adam,kain).
rodic(eva,abel).
rodic(adam,abel).
```

Jednoduchý program v Prologu

```
zena(eva).
muz(adam).
muz(kain).
muz(abel).
rodic(eva,kain).
rodic(adam,kain).
rodic(eva,abel).
rodic(adam,abel).
```

atomy

Jednoduchý program v Prologu

```
zena(eva).
muz(adam).
muz(kain).
muz(abel).
rodic(eva,kain).
rodic(adam,kain).
rodic(eva,abel).
rodic(adam,abel).
```

atomy

funktory

Procedury v Prologu

```
zena(eva).
```

} procedura definuje predikát zena/1

```
muz(adam).
muz(kain).
muz(abel).

rodic(eva,kain).
rodic(adam,kain).
rodic(eva,abel).
rodic(adam,abel).
```

Procedury v Prologu

```
zena(eva).
```

} procedura definuje predikát zena/1

```
muz(adam).
muz(kain).
muz(abel).
```

} procedura definuje predikát muz/1

```
rodic(eva,kain).
rodic(adam,kain).
rodic(eva,abel).
rodic(adam,abel).
```

Procedury v Prologu

```
zena(eva).
```

} procedura definuje predikát zena/1

```
muz(adam).
muz(kain).
muz(abel).
```

} procedura definuje predikát muz/1

```
rodic(eva,kain).
rodic(adam,kain).
rodic(eva,abel).
rodic(adam,abel).
```

} rodic/2

SWI Prolog

http://swi-prolog.org/

- 1987 - nyní
- Jan Wielemaker, Vrije Universiteit Amsterdam
- Sociaal-Wetenschappelijke Informatica
- open source (BSD)
- Windows, Unix, macOS

XPCE

- nástroj pro tvorbu GUI
- nezávislý na platformě (a na jazyce)

SWI Prolog

Editor zdrojového kódu

- PceEmacs: vestavěný editor SWI - Prologu
  - » klon editoru Emacs
  - » implemetován v Prologu + XPCE
  - » automatické odsazování, zvýrazňování syntaxe, ...
  - » `?- emacs.`
  - » `?- edit(file('test.pl')).  % nový`
  - » `?- edit('test.pl').  % existující`
  - » `?- edit(test).  % .pl lze vynechat`
  - » menu File / Edit
- Váš oblíbený editor
  - » lze nastavit v **`Settings / User init file`**
- Visual Studio Code
  - » rozšíření New-VSC-Prolog

Práce v SWI-Prologu

Spuštění SWI-Prologu

- `?-`

Editor → soubor rodina.pl

Překlad

- `?- consult(rodina).`
- `?- [rodina].`
- `?- ['C:/Prolog/rodina.pl'].`
- menu File / Consult
- `?- make.`

Výpočet → zadání dotazu

- `?- muz(adam).`

Dotazy a odpovědi

Uživatel položí dotaz

- zadá cíl
- Prolog se pokouší cíl splnit
- unifikace & backtracking

```
?- muz(adam).
?- muz(eva).
?- muz(X). % X je proměnná
```

- hledáme všechny muže
- více řešení

Výpis všech řešení

Povely při výpisu násobných řešení

- pro další řešení zadej `;`
- pro návrat zadej `.` (`enter`)
- pro plný výpis Prologem zkráceného řešení zadej `w`

```
?- rodic(X,kain). % Kainovy rodiče
?- rodic(adam,Y). % Adamovy děti
```

- vstup a výstup není určen předem

```
?- rodic(X,kain),muž(X).   % Kdo je Kainův
                             otec?
```

- složený dotaz s konjukcí (`,`)

Jednoduchý program v Prologu II

```
zena(eva).
muz(adam). muz(kain). muz(abel).
rodic(eva,kain). rodic(adam,kain).
rodic(eva,abel). rodic(adam,abel).
```

pravidlo

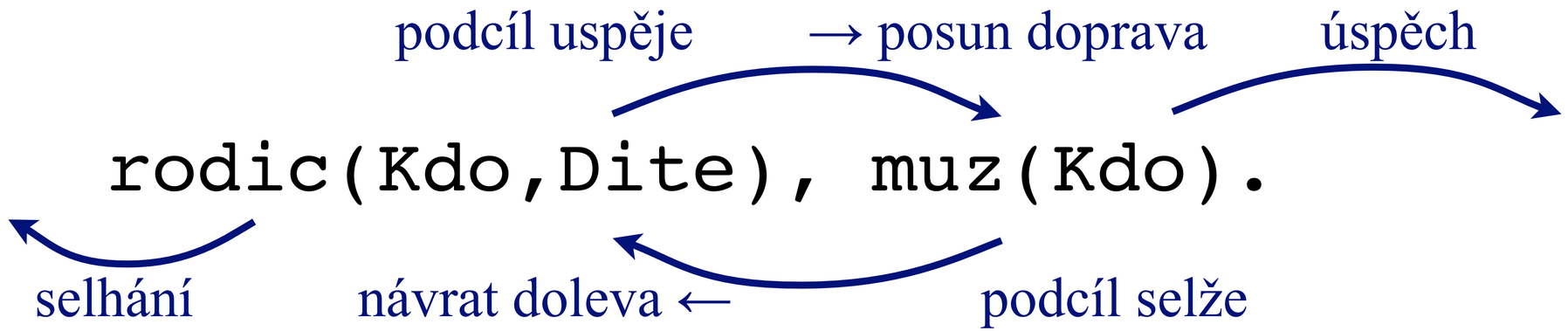
Vyhodnocení pravidla

```
?- otec(X,kain).
```
% Kdo je Kainův otec?

Jak splnit cíl, který je hlavou pravidla?

- je třeba splnit tělo
- tělo je konjunkcí podcílů ⇒ třeba splnit každý podcíl

  » vyhodnocení podcílů zleva doprava



Procedura s více pravidly

```
% clovek(C):- C je člověk.
clovek(C):- zena(C).
clovek(C):- muz(C).
?- clovek(X).
```

Poučení

- klauzule jsou při splňování cíle procházeny v pořadí, v jakém jsou zapsány

Disjunkce

```
% clovek(C):- C je člověk.
clovek(C):- zena(C).
clovek(C):- muz(C).
```

Alternativní zápis

```
clovek(C):- zena(C); muz(C).
```

Disjunkce

- středník `;` reprezentuje logickou spojku nebo
- konjunkce (`,`) “váže více” nežli disjunkce (`;`)

Další predikáty

```
% bratr(Bratr,Osoba):- Bratr je bratrem Osoby.
bratr(Kdo,Či):- rodic(R,Kdo),
                rodic(R,Či),
                muz(Kdo).
```

☀ Je tato definice korektní?

✘ Není!

`Kdo` a `Či` musí být různé osoby!

- `Kdo \= Či`

Procedura bratr/2

```
% bratr(Bratr,Osoba):- Bratr je bratrem Osoby.
bratr(Kdo,Či):- rodic(R,Kdo),
                rodic(R,Či),
                muz(Kdo),
                Kdo \= Čí.
```

✎ Problém

- přepište pravidlo jako formuli predikátového počtu
- a zkuste doplnit kvantifikátory (∀, ∃)

✎ Problém

Sestavte následující predikáty

```
% tchyne(Kdo,Čí):- Kdo je tchyní osoby Čí.
% sestrenice(Kdo,Čí):- Kdo je sestřenicí
                        osoby Čí.
% svagr(Kdo,Čí):- Kdo je švagrem osoby Čí.
```

Anonymní proměnná

```
% rodic(X):- X má dítě.
rodic(X):- rodic(X,Y).
```

Ekvivalentní zápis

```
rodic(X):- rodic(X,_).
```

- znak _ označuje anonymní proměnnou
- „ na jménu této proměnné nezáleží ”
- dva různé výskyty _ označují různé proměnné!

Problém genealogický

Vzal jsem si za ženu vdovu, která již měla dospělou dceru.

Můj otec, který nás často navštěvoval, se do mé (nevlastní) dcery zamiloval a oženil se s ní.

Tak se můj otec stal mým zetěm a má (nevlastní) dcera mojí (nevlastní) matkou.

Problém genealogický

O několik měsíců později má žena porodila syna, který se tak stal švagrem mého otce a současně mým strýcem.

Žena mého otce – tedy má (nevlastní) dcera – později také porodila syna, který se tak stal mým bratrem a současně i vnukem ...

✎ Problém genealogický

① V jazyce Prolog popište fakta z příběhu.

② Přidejte pravidla pro definici příbuzenských vztahů.

③ Formulujte dotazy, které ověří platnost tvrzení uvedených v příběhu, jako např. „můj syn se tak stal mým strýcem" apod.

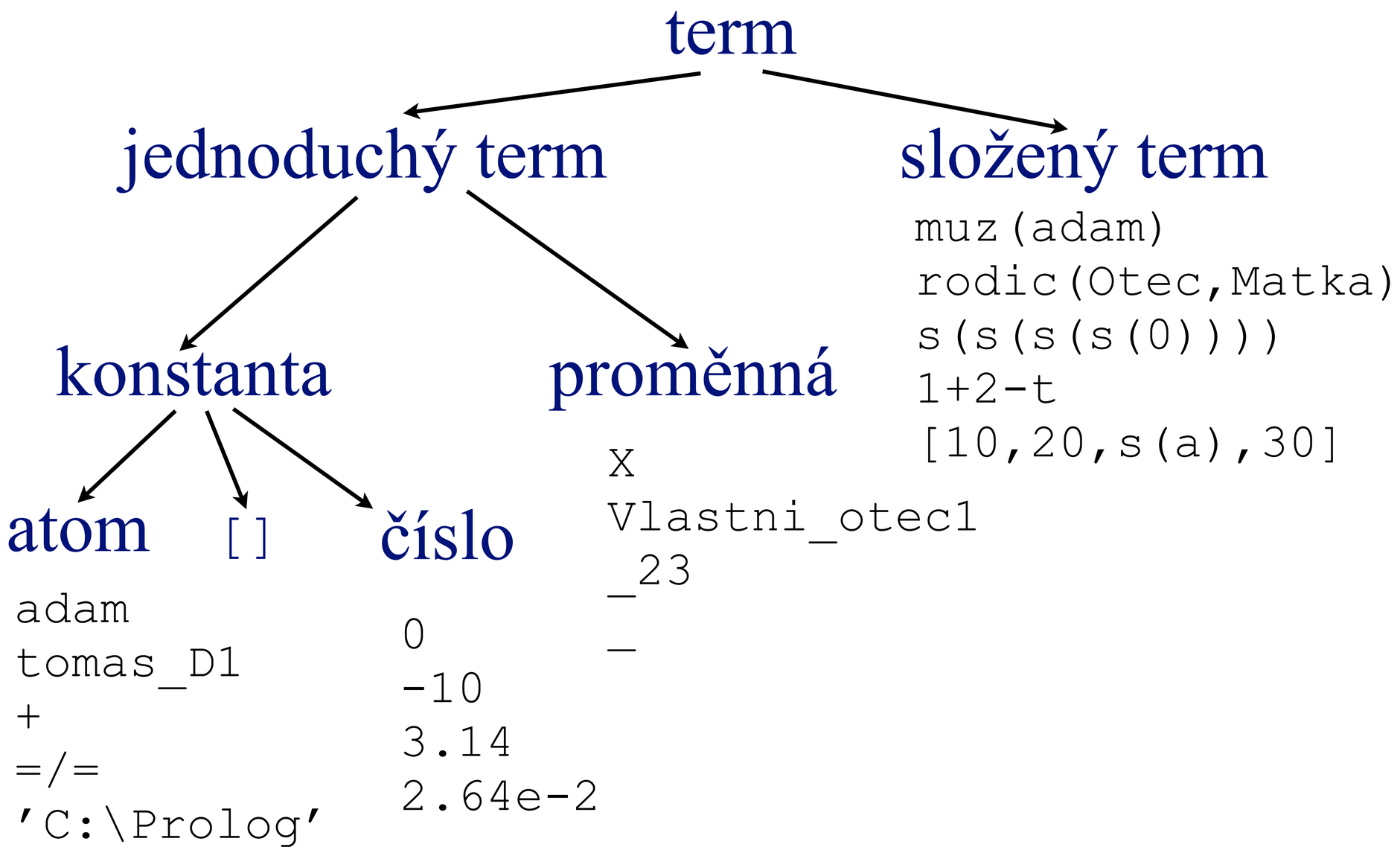
④ Formulujte dotazy, které dokáží tvrzení

„Já jsem svým dědečkem"

a

„Má žena je mojí babičkou"

K syntaxi Prologu : Termy



Jednoduché termy

Atom

- začíná malým písmenem a obsahuje pouze písmena, číslice a _
  - např. `prednaska_Prolog1`
- obsahuje pouze tyto speciální znaky
  - `+ - * / \ ~ ^ < > = : . ? @ # $ &`
  - např. `?-  <==>  :-  +`
  - kromě `/*` (začátek komentáře)
- středník ; vykřičník !
- je tvořen znaky uzavřenými mezi apostrofy
  - např. `'C:\Prolog'  'Adam'`

Jednoduché termy

Proměnná

- začíná velkým písmenem nebo _
- obsahuje pouze písmena, číslice a _

Složené termy (struktury)

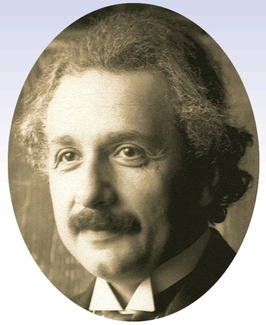
Rekurzivní definice

- funktor$(arg_1, arg_2, ..., arg_n)$
- n - ární funktor & n argumentů
- funktor je (syntakticky) atom
- argumenty jsou opět termy
- funktor je určen jménem i aritou: rodic/1 i rodic/2

☀ Příklady

- `rodic(X,kain)`
- `prednaska(datum(19,2,2024),kod('NPRG005'))`
- `s(s(s(s(0))))`

Příklad: Einsteinova hádanka



Tři přátelé

Ada, Bill a Steve

programují ve třech různých jazycích

C#, Swift a Prolog

a používají tři různé operační systémy

Linux, MacOS a Windows.

Žádní dva z nich přitom

- ✘ neprogramují ve stejném jazyce
- ✘ ani nepoužívají stejný operační systém.

Víme, že

Bill bydlí vedle uživatele operačního systému MacOS.

Uživatel Windows programuje v jazyce C#.

Bill nesnáší Linux.

Ada chodí k programátorovi v jazyce Swift obdivovat MacOS.

✎ Problém: Kdo programuje v Prologu ?

✎ Einsteinova hádanka (jednodušší)

```
% prolog(Kdo) :- Kdo programuje v Prologu.
?- prolog(Kdo).
   Kdo = tenKdoProgramujeVPrologu
```

Nápověda

- pokuste se "přeložit" zadání z přirozeného jazyka do Prologu
- podobně jako u příbuzenských problémů můžete začít fakty, ovšem místo `muz/1` či `zena/1` zde může stačit jen `osoba/1`
- pro definici predikátu `prolog/1` bude třeba pravidlo, do jehož těla musíme zakomponovat "indicie" ze zadání
- skutečnost, že hodnoty dvou proměnných `X` a `Y` budou stejné, lze vyjádřit pomocí operátoru `=` , tj. termem `X = Y`
- skutečnost, že hodnoty dvou proměnných nebudou stejné či proměnná nebude nabývat nějaké hodnoty, lze vyjádřit pomocí operátoru `\=`

✎ Einsteinova hádanka (složitější)

Sestavte definici predikátu `reseni/1`, který vrátí úplné řešení hádanky ve tvaru složeného prologovského termu.

```
?- reseni(R).

   R = pratele( osoba(prvni, prvniJazyk, prvniOS),
                osoba(druhy, druhyJazyk, druhyOS),
                osoba(treti, tretiJazyk, tretiOS))
```

Einsteinova hádanka: datová struktura

```
pratele( osoba(prvni,prvniJazyk,prvniOS),
         osoba(druhy,druhyJazyk,druhyOS),
         osoba(treti,tretiJazyk,tretiOS))
```