Alternativní metody programování

Pavel Satrapa
pavel.satrapa@tul.cz

Literatura

- Bratko I.: **Prolog Programming for Artificial Intelligence**, Addison Wesley, 2000
- Barták R.: **On-line Guide to Prolog Programming** http://kti.ms.mff.cuni.cz/~bartak/prolog/
- Felleisen, Findler, Flatt, Krishnamurthi: **How to Design Programs**, www.htdp.org
- Abelson, Sussman: **Structure and Interpretation of Computer Programs**, http://mitpress.mit.edu/sicp/full-text/book/book.html

Paradigma programování

- základní přístup (idea, filosofie) při vytváření programu
- předurčuje pohled programátora na řešený problém, dostupné konstrukce konkrétních jazyků apod.
- historický vývoj
  - větší abstrakce
  - řešení velkých projektů
- neexistuje oficiální klasifikace

Pravěk

- **strojový kód**
  - přirozený jazyk procesoru
  - absolutní adresy
  - uživatelsky velmi nevstřícný
- **jazyk symbolických adres (assembly language)**
  - strojový kód s lidskou tváří
  - instrukce mnemotechnicky
  - proměnné

Imperativní (procedurální)

- nejstarší – 50. léta
- přímočarý popis algoritmu
- postup řešení krok za krokem
- ukládání mezivýsledků do proměnných
- základ většiny běžných programovacích jazyků – Pascal, C a mnohé další (v současných verzích zpravidla obohaceny o další prvky)

Modulární

- 80. léta
- cíl: usnadnit týmovou práci rozdělením úlohy na části – moduly
- **modul** je syntakticky (a logicky) ucelená část programu řešící určitou část zadání
  - **rozhraní** definuje vztah s okolím
  - **implementace** popisuje vlastní realizaci
- Modula 2, Ada

Příklad modulu (Pascal)

```
unit Uloziste;

interface
uses DatoveTypy;
procedure UlozPolozku ( P : Polozka );
procedure NactiPolozku ( ID : Integer; var P : Polozka );

implementation
var Data : file of Polozka;
procedure UlozPolozku begin ... end;
procedure NactiPolozku begin ... end;

begin
Assign ( Data, JmenoSouboru ); Reset ( Data );
end.
```

rozhraní

implementace

tělo

název modulu

využívá jiný modul

nabízí ostatním

implementace slíbeného zvenčí nedostupná

inicializace

Objektově orientované (1)

- původem z 60. let, prosadilo se v 80.–90. letech
- spojení dat a operací (podprogramů)
- **zapouzdření (enkapsulace)**
  - spojení dat a metod
  - ortodoxně: k datům není přímý přístup
- **dědičnost**
  - hierarchie tříd
  - potomci dědí po rodičích

Objektově orientované (2)

- **polymorfismus**
  - chování metody závisí na tom, pro který konkrétní objekt se volá
  - určuje se dynamicky za běhu
- **třída** – obecná definice položek a metod (typ)
- **objekt** – konkrétní exemplář dané třídy naplněný daty (proměnná)
- Simula 67, SmallTalk, Java,...

Řízené událostmi (1)

- asynchronní
- (víceméně) náhodně přicházejí události (stisk klávesy, příchod zprávy)
- jádro (smyčka událostí) vybere a zavolá obslužný podprogram, podobá se obsluze přerušení
- části programu mohou vyvolat událost – předávání informací ostatním
- typické pro uživatelská rozhraní

Smyčka událostí

čekání na událost

dokud nebyla obsloužena
    výběr obsluhujícího (podle typu události)
    volání obslužné funkce (ohlásí, zda událost
        obsloužila nebo ne)

- nekonečná smyčka
- pokud nikdo neobslouží – chyba

Řízené událostmi (2)

- nevyžaduje speciální programovací jazyk, často jen sada knihoven
  - „předvařená“ kostra aplikace
  - uživatel připojuje svůj kód na připravená místa – naplní hlavní menu, navěsí na jeho položky obslužné podprogramy,...
- definována pravidla eskalace událostí – nevyřízenou událost dostane další podprogram
- často využívá objektově orientovaný přístup

Paralelní

- současný běh programu
- vlákna | procesy | procesy na různých strojích
- potřeba vzájemné synchronizace a interakce
  - zasílání zpráv
  - sdílená paměť
- implementace: knihovna nebo specializovaný programovací jazyk

Logické

- aplikace matematické logiky v programování
- interpretem je rezoluční stroj, který z faktů a pravidel odvozuje/ověřuje další fakta
- především deklarativní přístup (popis logických vztahů), procedurální složka potlačena
- klasickým představitelem je Prolog

Funkcionální

- výpočet je vyhodnocením matematické funkce (definuje vztah mezi vstupními a výstupními hodnotami)
- vše je výraz, vyhodnocením vznikne hodnota
- v klasické podobě se vyhýbá stavovým datům (proměnným) – průběžné hodnoty se předávají v parametrech a výsledcích funkcí
- časté použití rekurze a funkcí vyššího řádu
- Lisp, Scheme, Haskell,...

Speciální

- některé jazyky se zcela vymykají
- např. jazyky zásobníkové – Forth, **PostScript**
- operace berou parametry z vrcholu zásobníku a výsledky ukládají tamtéž
- (3 + 5) * (12 – 6) se v PostScriptu řekne
  3 5 add 12 6 sub mul
- použitím základních vykreslovacích operací umožňuje popsat stránku

Velmi speciální

- **ArnoldC**
- (imperativní) programovací jazyk založený na hláškách Arnolda Schwarzeneggera
- zajímavá práce s výrazy: a = (4 + b) * 2

```
GET TO THE CHOPPER a
HERE IS MY INVITATION 4
GET UP b
YOU'RE FIRED 2
ENOUGH TALK
```

- https://github.com/lhartikk/ArnoldC/

Prolog – historie

- PROgramming in LOGic
- vznikl počátkem 70. let jako jazyk pro umělou inteligenci a výpočetní lingvistiku
- cíl: pokud možno abstraktní vyjádření faktů a logických vztahů mezi nimi, potlačení imperativní složky
- praxe: jasně daná pravidla vyhodnocování – např. záleží na pořadí pravidel

Implementace Prologu

- celá řada
- zpravidla interprety – interaktivní způsob práce
- většinou velmi rustikální (zapomeňte na GUI)
- nejčastěji používaný je **SWI-Prolog**
  www.swi-prolog.org

Práce s Prologem

- **definice faktů** o objektech a vztazích mezi nimi
- **definice pravidel** pro odvozování dalších faktů
- **kladení otázek**
  - ověř, zda platí ...
  - najdi hodnoty, pro které platí ...
- obvykle interaktivní využití – interpret jazyka

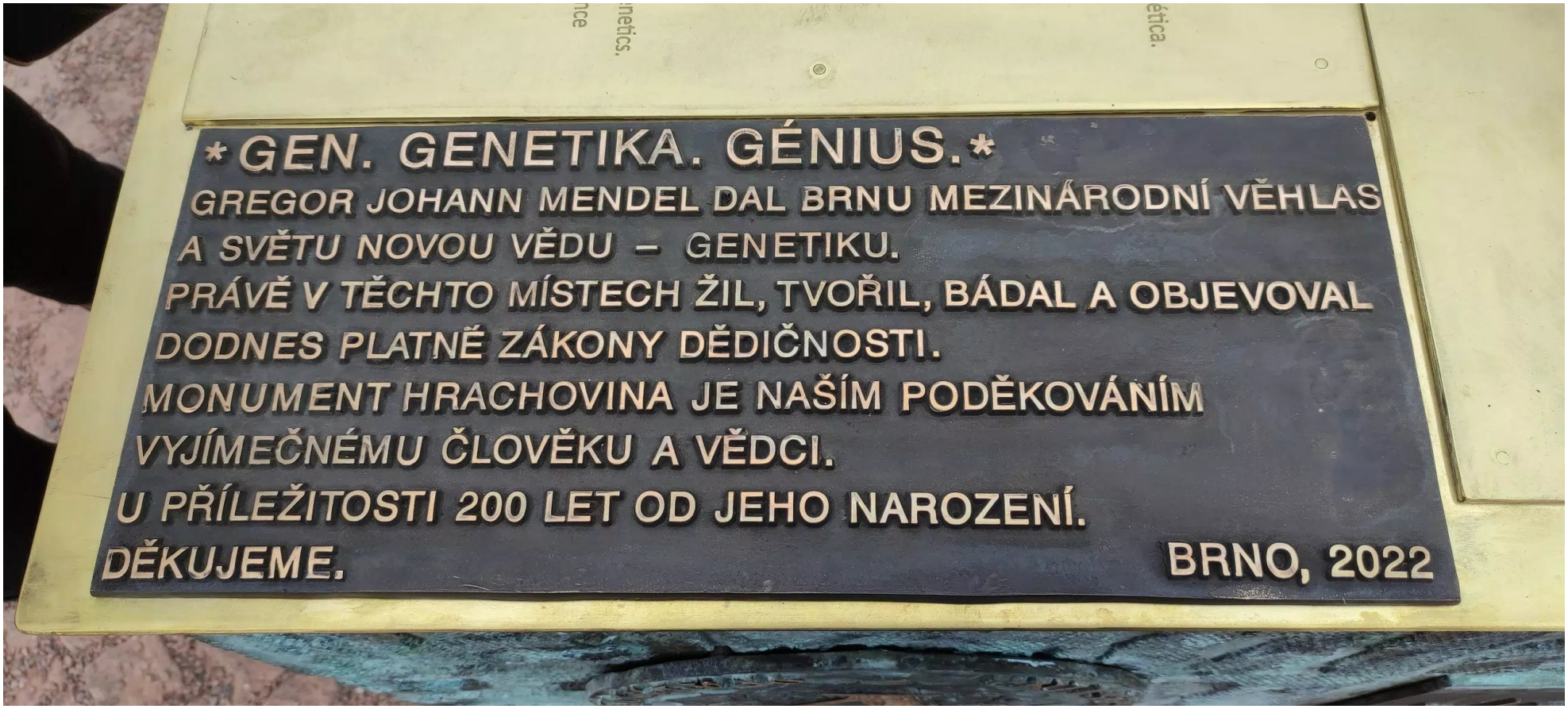
Konstanty

- **čísla**
  - původní Prolog podporoval jen celá čísla
  - většina implementací umožňuje i reálná
  - 19, –5
- **atomy**
  - řetězec znaků začínající malým písmenem
  - posloupnost znaků uzavřená v apostrofech
  - jana, 'Emanuel', '20. 2. 2007'

Fakt

- poznatky (fakta) o vlastnostech objektů či vztazích mezi objekty
- základní syntaxe:
  název_predikátu(argumenty).
- nezapomenout na **tečku na konci**, ukončuje klauzuli
- počet argumentů = arita predikátu
  např. rodic/2
- rodic(jana,petr).

Prolog v umění



fotografie: @sonja_ruda

Tuto. Tabulku. Nejspíš. Vytvořil.
Prologovský programátor.

Pravidlo

- složitější součást predikátu – umožňuje **odvozovat nová fakta** z již existujících (implikace)
- tvar: *hlavička :- tělo.*
- **hlavička** definuje odvozovaný fakt, **tělo** podmínky, za nichž je pravdivý, obsahuje jeden či více **cílů**
- pokud se interpretu podaří odvodit, že tělo je pravdivé, ověřil pravdivost hlavičky
- dite(X,Y) :- rodic(Y,X).

Predikát

- tvořen sadou klauzulí se **stejným jménem a stejnou aritou** – rodic(jana,petr) a rodic(jana) jsou dva různé predikáty: rodic/2 a rodic/1
- může obsahovat fakta i pravidla
- fungují jako alternativy, **platnost predikátu lze dokázat libovolnou z nich**
- cech(praotec).
  cech(premysl).
  cech(X) :- skace(X).

Dotaz

- Prolog rozeznává dva režimy:
  - **konzultační** (zadávají se fakta a pravidla) a
  - **dotazovací** (kladou se otázky)
- **ověření faktu** – je jana rodičem petra?
  ?- rodic(jana,petr).
- **nalezení vyhovující hodnoty** – kdo je rodičem petra?
  ?- rodic(X,petr).
  zobrazí první nalezenou hodnotu,
  stisknutím **;** lze požadovat další

Proměnná

- začíná velkým písmenem
- **v pravidlech a faktech** – popisuje účastníky vztahu
- **v dotazech** – reprezentuje hledané objekty
- **volná proměnná** – o hodnotě zatím nic nevíme, interpret se ji snaží najít
- **vázaná proměnná** – z dřívějších kroků při řešení vyplývá její hodnota, je svázána s určitou hodnotou

Rozsah platnosti proměnné

- rozsahem platnosti proměnné je jen **jedna klauzule**
- stejnojmenná proměnná v sousední klauzuli nemá s touto nic společného (i když jsou součástí stejného predikátu)
- **hodnotu získává srovnáváním** při aplikaci klauzule
- jakmile byla přiřazena, hodnota se dále nemění (jen odvoláním pravidla a jeho novou aplikací v nových podmínkách)

Jednoduchý příklad

```
rodic(jana,petr).
rodic(emil,petr).
rodic(ivana,jana).
rodic(alfons,jana).
rodic(otylie,emil).
prarodic(X,Y) :- rodic(X,Z), rodic(Z,Y).
```

definice faktů

pravidlo

```
?- prarodic(ivana,petr).        Yes
?- prarodic(otylie,emil).       No
?- prarodic(X,petr).            X=ivana; X=alfons; X=otylie; No
```

dotazování

vytvořeno s podporou
projektu ESF