Prolog – logické programování

Tomáš Matoušek

Úvod

Cílem přednášky bude představit netradiční programovací jazyk *Prolog* mající úzkou vazbu na matematickou logiku. *Logické programování* v Prologu se dosti liší od tradičního programování v jazycích, jako jsou Pascal, C++, C♯, Java apod., které se v programátorské praxi běžně používají a které se nazývají *procedurální*. Tyto jazyky se vyznačují tím, že autor programu musí přesně napsat, jak se daný problém má řešit. Prolog je naopak jeden z tzv. *deklarativních* jazyků, což znamená že programátor specifikuje, co se má řešit, a už nechá na Prologu, jak tento problém konkrétně vyřeší.

Vzhledem k těmto vlastnostem Prologu není pro pochopení obsahu přednášky podstatné, zda již programovat umíš či ne. Naopak ti, kteří jsou již zvyklí na programování v některém z procedurálních jazyků, mohou mít s přístupem Prologu zpočátku o malinko větší potíže, než ti, kteří zatím nikdy neprogramovali.

Na přednášce nebudeme jazyk Prolog nijak formalizovat ani rozebírat žádné složité algoritmy. Spíše si na příkladech budeme ukazovat, jak Prolog vypadá a vysvětlíme si jeho filosofii.

Pro zajímavost uveďme, že logické programování se prakticky používá především v oblasti umělé inteligence, kde se vedle Prologu uplatňují i další podobné jazyky, jako je např. LISP.

Základy

Program v Prologu pracuje s *termy*, které lze rozdělit na *jednoduché termy* a *struktury*. Jednoduché termy jsou proměnné (začínají velkým písmenem nebo podtržítkem) a konstanty, což jsou celá čísla nebo *atomy*.

Atomy jsou téměř libovolné „nápisy“ začínající malým písmenem, celé sestavené ze speciálních symbolů, nebo libovolné znaky uzavřené do apostrofů. Atomy nemají žádnou vnitřní strukturu ani význam. Prolog umí o atomech říci pouze to, zda jsou si dva z nich rovny.

Příklad. `tomáš`, `dino`, `anša`, `:-`, `=<`, `+`, `'Milý příteli!'` jsou atomy.

Struktury jsou termy, které obsahují jiné termy. Struktura je tvořena *funktorem* a argumenty. Funktor je atom s danou četností (počtem parametrů). Argumentem může být libovolný term.

Příklad. `datum(30,duben,2005)`, `datum(30,duben)`, `přímka(bod(1,5),bod(8,10))`, `+(a,b)` jsou struktury.

Unární a binární struktura může být definována jako operátor a pak její použití lze zapisovat i v infixové nebo postfixové notaci. Lze tedy např. místo `+(a,b)` psát `a + b`, význam je tentýž. Poznamenejme, že `1 + 1` není totéž co `2`. První zápis je binární struktura, jejímž prvním i druhým parametrem je konstanta 1. Druhý zápis je konstanta 2.

Program v Prologu je posloupnost *pravidel* ve tvaru `hlava :- tělo.` a *faktů* ve tvaru `hlava.`. Fakta jsou jen zkrácené zápisy pro pravidla `hlava :- true.`. Hlava pravidla je *predikát*, tj. term, o kterém rozhodujeme, zda platí či neplatí. Tělo pravidla je tvořeno predikáty, které jsou spojeny operátory čárka (konjunkce) a/nebo středník (disjunkce). Binární operace `:-` odpovídá implikaci $\Leftarrow$. Pravidlo tedy vyjadřuje skutečnost, že pokud platí tělo pravidla, pak platí i jeho hlava.

Příklad. Definujeme binární predikát *matka*, který říká, zda první term je matkou druhého. Poté můžeme definovat predikát *sourozenec* tak, že X je sourozenec Y, pokud existuje matka M taková, že X, Y jsou její různé děti.

```
matka(jana,pavel).
matka(jana,jan).
sourozenec(X,Y):-matka(M,X),matka(M,Y),X\=Y.
```

Prolog nyní umí *dokázat*, že Pavel a Jan jsou sourozenci a umí dokonce najít všechny termy, o kterých lze dokázat, že jsou sourozenci. Jak Prolog při takovém dokazování postupuje, si povíme na přednášce.

Příklad. (Seznamy)

V Prologu se n-ticím říká seznamy a zapisují se do hranatých závorek, např. `[1,2,3]`. Prázdný seznam se značí `[]`. Zápis `[H|T]` nám „označí“ první prvek seznamu H a zbytek seznamu T.

Jednoduchá operace se seznamy je smazání právě jednoho výskytu daného termu ze seznamu. Definujeme tedy ternární predikát `delete_one(X,S,T)`, který říká „T je seznam obsahující prvky seznamu S bez právě jednoho výskytu X“. Predikát vyjadřující tuto vlastnost lze definovat takto:

```
delete_one(X,[X|U],U).
delete_one(X,[Y|U],[Y|V]):-delete_one(X,U,V).
```

První pravidlo (fakt) říká, že máme-li seznam začínající prvkem X, pak zbytek tohoto seznamu je seznamem bez jednoho výskytu X. Druhé pravidlo říká, že je-li V seznam, který vznikl z U odebráním jednoho prvku X, pak také seznamy U, V budou po připojení Y na začátek splňovat požadovanou vlastnost. Z opačného pohledu pravidlo převádí úlohu smazání prvku ze seznamu o délce n na tutéž úlohu pro seznam o délce

$n-1$. Takovému postupu se říká *rekurze* a je základním prostředkem používaným v logickém programování.

Příklad. (Permutace) Posledním příkladem, který zde uvedeme, je predikát, jenž říká, zda jsou dané dva seznamy svými permutacemi.

```
perm([],[]).
perm(Q,[X|P]):-delete_one(X,Q,Z),perm(Z,P).
```

První pravidlo říká, že prázdný seznam je permutací prázdného seznamu. Druhé pak, že pokud ze seznamu Q odmažeme právě jeden výskyt prvku X a výsledek je permutací P, pak také Q je permutací $[X|P]$.

Pokud máme takto definovaný predikát, můžeme se Prologu zeptat, jaké jsou všechny permutace daného seznamu. K takovému zeptání se slouží operátor `?-`.

```
?- perm([1,2,3],X).
```

Prolog najde všechna X, pro která predikát `perm([1,2,3],X)` platí:

```
X = [1, 2, 3];
X = [1, 3, 2];
X = [2, 1, 3];
X = [2, 3, 1];
X = [3, 1, 2];
X = [3, 2, 1];
```