Praktikum z logického programování (kombi) - Cvičení č. 1

KOMBINOVANÉ: ÚVOD A ZÁKLADY

Organizace

- 4x cvičení v pátky od cca 8h v učebně B3a
  - 15. 3., 26. 4., 10. 5., 24. 5.
- Ukončení: zápočet
- Podmínky zápočtu:
  - Povinná účast na cvičení min. 75 % (3 z 4)
  - Aktivní práce na cvičeních
  - Písemná forma ověření studijních výsledků

Co je to Prolog?

1. **Logický programovací jazyk**:
    1. Prolog je logický programovací jazyk, který se zaměřuje na popis cílů výpočtu.
    2. Programátor specifikuje, co chce dosáhnout, a systém se snaží najít odpovídající řešení.
2. **Deklarativní charakter**:
    1. Prolog patří mezi deklarativní jazyky.
    2. Programátor popisuje, co chce, aniž by specifikoval konkrétní postup výpočtu.
3. **Jednoduchá syntaxe**:
    1. Syntaxe Prologu je snadno čitelná a použitelná.
    2. Umožňuje specifikovat fakta, pravidla a dotazy.
4. **Využití v umělé inteligenci a lingvistice**:
    1. Prolog se často používá v oblasti umělé inteligence, prohledávání databází a zpracování přirozeného jazyka.
5. **Původ názvu**:
    1. Název „Prolog“ pochází z francouzského „PROgrammation en LOGique“ („logické programování“).

Zápis klauzulí v Prologu

- **Program v Prologu je konečná neprázdná množina Hornových klauzulí**
  - Klauzule s nejvýše jedním pozitivním literálem
- Dva druhy klauzulí:
  - **fakty** – tvrzení bez předpokladů, je to obdoba toho, co jsme měli v předchozí kapitole jako speciální axiomy
  - **pravidla** – obecná tvrzení ve tvaru „Závěr platí, pokud platí všechny jeho předpoklady zároveň.“
- *Používání programu* spočívá v zadávání dotazů (cílových klauzulí) – **Hornových klauzulí** bez pozitivních literálů. Prolog dotazy vyhodnocuje podle programu a podle vnitřních pravidel (obdoba logických axiomů).

Zápis klauzulí v Prologu

- Příkazy se ukončují tečkou
- Pravidla jsou tvaru ***hlava pravidla :- tělo pravidla.***
  - Pro klausulární logiku antecedent → konsekvent
    - Antecedent === tělo pravidla v prologu
    - Konsekvent === hlava pravidla v prologu
- Výzva pro dotaz má tvar ***?- dotaz.***

	Klauzulární logika	**Predikátová klauzule**	**Zápis v Prologu**
Pravidlo	$B, C, D \rightarrow A$	$A \vee \neg B \vee \neg C \vee \neg D$	A :- $B, C, D.$
Fakt	$\rightarrow A$	A	$A.$
Dotaz	$B, C, D \rightarrow$	$\neg B \vee \neg C \vee \neg D$	?- $B, C, D.$

Zápis klauzulí v Prologu

- Disjunkce (**OR**) v těle pravidla (**;** středník mezi atomy)
  - koupit(X) :- snedeno(X); vypito(X).
  - Dle vztahu v klauzulární logice
    (F1 OR F2 -> K) <=> ((F1 -> K) AND (F2 -> K)) je ekvivalentní pro dvě pravidla:
    - **koupit(X) :- snedeno(X).**
    - **koupit(X) :- vypito(X).**
- Konjunkce (**AND**) v hlavě pravidla (**,** čárka mezi atomy)
  - Nepoužívá se v Prologu! ← Hornovy klauzule!
  - Převod dle vztahu v klauzulární logice
    (A -> F1 AND F2) <=> ((A -> F1) AND (A -> F2)) na dvě pravidla!

Zápis elementů

- Proměnné zapisujeme velkým počátečním písmem
  Promenna, X, Auto
- Konstanty zapisujeme malým počátečním písmenem
  konstanta, zluta, jan, 8, "Petr", 'Alice‘
- predikat/2: predikat(parametr1, parametr2)
  otec(jan, klara).
- Pravidla: závěr :- předpoklad1, předpoklad2, …
  lovi(Kdo, Koho) :- kocka(Kdo), mys(Koho).
  V predikátové logice: ***kocka(Kdo) & mys(Koho) -> lovi(Kdo, Koho)***

Prolog a klausulární logika

- Klauzule: hlava :- tělo. vs. antecedent → konsekvent
  - Atomy a spojky (čárka AND a středník OR)
  - Hornovy klauzule - maximálně jeden atom v hlavě (konsekventu)
  - Fakta: pouze_hlava.
  - Dotaz: ?- pouze_telo.
  - Pravidlo: hlava :- telo.
- Atom (atomická formule)
  - Logická konstanta (true, false)
  - Predikát s danou aritou (počet argumentů - termů)
    - Uživatelsky definované nebo vestavěné
    - predikat/2: predikat(term1, term2).
    - Vrací true nebo false

Prolog a klausulární logika

- Term
  - Proměnná: X ← vždy počáteční velké písmeno
  - Anonymní proměnná: _, _1, … ← začíná podtržítkem, náhrada za existenční termy
  - Konstanta: eva, "Jan", 'ALÍK', 8 ← malé počáteční písmeno nebo apostrof/uvozovky nebo číslo (integer i float)
  - Funktor s danou aritou (počet argumentů - termů)
    - Uživatelsky definované nebo vestavěné
    - funktor/2: funktor(term1, term2).
    - Vrací hodnotu!
    - Infixová A + B nebo prefixová +(A,B) notace ← podobně u predikátů (atomů)

Prolog a klausulární logika

- Bázový atom – atom, jehož parametry (termy) jsou bázové termy
- Bázový term – term, který neobsahuje proměnnou (ani anonymní!)

Příklad – převod z klauzulární logiky

1. „Jahoda je červená.“

Klauzulární logika: ?

Prolog: ?

2. „Psi mají čtyři nohy.“

Klauzulární logika: ?

Prolog: ?

3. „Děti mají rády sladká jídla.“

Klauzulární logika: ?

Prolog: ?

Příklad – převod z klauzulární logiky

1. „Jahoda je červená.“

Klauzulární logika: → barva(jahoda, cervena)

Prolog: ?

2. „Psi mají čtyři nohy.“

Klauzulární logika: pes(X) → pocet_nohou(X, 4)

Prolog: ?

3. „Děti mají rády sladká jídla.“

Klauzulární logika: dite(X), jidlo(Y), chut(Y, sladky) → ma_rad(X, Y)

Prolog: ?

Příklad – převod z klauzulární logiky

1. „Jahoda je červená.“

Klauzulární logika: → barva(jahoda, cervena)

Prolog: barva(jahoda,cervena).

2. „Psi mají čtyři nohy.“

Klauzulární logika: pes(X) → pocet_nohou(X, 4)

Prolog: pocet_nohou(X,4) :- pes(X).

3. „Děti mají rády sladká jídla.“

Klauzulární logika: dite(X), jidlo(Y), chut(Y, sladky) → ma_rad(X, Y)

Prolog: ma_rad(X,Y) :- dite(X), jidlo(Y), chut(Y,sladky).

Znalostní báze

- **Množina tvrzení (pravidel)** o uzavřeném modelovaném světě **mezi nimiž je vztah konjunkce (AND)**
- Pravidla jsou tvaru **hlava pravidla :- tělo pravidla.**
- Skládá se z:
  - **fakta** – množina elementů, která má prázdné tělo a neprázdnou hlavu (ekvivalent v klausulární logice je **true -> Konsekvent**)
    - pes(azor).
  - **pravidla** – množina elementů, která nemá prázdné ani tělo ani hlavu (v klasulární logice **Anteceden -> Konsekvent)**
    - ma_prázdniny(X) :- skolak(X), obdobi(leto).

Pravidla pro sestavování znalostní báze

- předem si promyslíme názvy predikátů a konstant tak, aby byly čitelné pro uživatele, případně můžeme přidávat komentáře,
- pokud se proměnná vyskytuje v klauzuli pouze jednou, použijeme anonymní proměnnou,
- funktory používáme jen tehdy, když je to opravdu nutné a bereme na vědomí, že funktor lze používat spíše jen jako argument predikátu,
- v bázi uvedeme nejdřív fakty a pak pravidla, zvláště v případě, že některá pravidla mají v hlavě shodný predikát s příslušným faktem,
- jestliže se v pravidle vyskytuje rekurze, toto pravidlo uvedeme jako poslední ze všech klauzulí, které mají v hlavě tentýž predikát,
- pokud je v klauzuli atom negovaný predikátem not, pak tento atom uvádíme v klauzuli jako poslední,
- všímáme si také nepřímé rekurze

?- Dotazy

- Zajímá nás pravdivost predikátu – všechny atomy jsou bázové
  ?- prcha(micka, zoubek).
  true
- Zajímají nás všechny kombinace pro které je dotaz splnitelný – proměnná
  ?- prcha(X, zoubek).
  X = micka
- Existuje hodnota, kterou lze dosadit – anonymní proměnná
  ?- prcha(_, zoubek).
  true

Zvýraznění syntaxe

- jasně červené jsou predikáty, které jsou použity pouze v klauzulích typu fakt nebo v hlavách klauzulí odpovídajících pravidlům (v predikátové nebo kauzulární logice by byly jen „za implikací“),
- **tučné černé** jsou predikáty nacházející se v hlavách klauzulí, které se vyskytují i v tělech jiných klauzulí,
- netučné černé jsou predikáty v tělech klauzulí,
- modré jsou operátory a některé vestavěné predikáty (například write/1),
- červenohnědé jsou proměnné,
- **tučné tmavé červené** jsou chyby,
- zelené jsou komentáře (komentářovým symbolem je procento).

Zvýraznění syntaxe

- Proč Prolog zabarvuje predikáty v hlavách klauzulí (a ve faktech) některé jasně červenou a jiné **černou** barvou?
- Protože ty **černé** se v jiných klauzulích nacházejí na opačné straně implikace, a tedy je možné použít je jako „spojovací materiál“ při unifikaci a následné rezoluci, když z takových dvou klauzulí chceme něco odvodit.
- Atomy s predikáty zabarvenými jasně červenou barvou nejsou chybné, jen se Prologu moc nelíbí, že nebude možné využít je pro rezoluci s jinou klauzulí.

Příklad: ma_rad

- Zadání:
  - Petr má rád květiny, Ivanu a televizi.
  - Jan má rád jitrnice a televizi.
  - Věra má ráda všechno, co má rád Jan
- Dotazy:
  - Má rád Petr televizi?
  - Má rád Jan květiny?
  - Co má rád Jan?
  - Kdo má rád jitrnice?
  - Co má rád Petr a zároveň Jan?
  - Kdo má co (koho) rád?

Příklad: zvířata

- Zadání
  - Hektor a Lea jsou lvi, Zulejda je zajíc.
  - Lvi jsou silní a velcí, mají žlutou barvu.
  - Zajíci jsou rychlí a mají hnědou barvu.
  - Kdo je silný a velký, je králem zvířat.
- Dotazy
  - Je Zulejda lev?
  - Je Lea králem zvířat?
  - Kdo je lev?
  - Kdo je rychlý?
  - Kdo je hnědý?
  - Kdo má jakou barvu?

Nápověda

- **apropos/1** – pro hledání vestavěných predikátů, kde přesně nevíme jak se jmenuje, ale známe nějaké klíčové slovo
  - apropos('into the database‘).
- **help/1** – potřebujeme nápovědu k vestavěnému predikátu
  - help(asserta).
  - **+argument** musí být instanciovaný, tedy mít už předem přiřazenu nějakou hodnotu
  - **-argument** zde má být proměnná
  - **?argument** instanciovaný parametr nebo proměnná
  - **@argument** parametr nebude vázán unifikací
  - **:argument** jedná se o název predikátu

Příklad: sub_string

- help('sub_string‘).
- sub_string(+String, ?Before, ?Length, ?After, ?SubString)
- sub_string("zadanretezec123", 5, 7, Zbytek, Substr).
  - Sub - vrací podřetězec délky 7 začínající na páté pozici
  - Zbytek – vrací počet zbývajících znaků
- sub_string("zadanretezec123", 5, 7, _, Substr).
  - Když nás nezajímá výsledek nějakého parametru, použijeme anonymní proměnnou
- sub_string("vstup123 x2123yt", Pozice, _, _, "123").
  - Zajímají nás všechny pozice hledaného podřetězce

Přiřazování, porovnávání, unifikace

- Jiný přístup nežli v procedurálních jazycích (např. C)
- Predikáty → true/false
- Operator is
  - Číslo is Výraz
  - Pokud je Číslo (zpravidla volná proměnná) úspěšně unifikováno s výsledkem Výrazu vrací true
    - Číslo je v relaci s hodnotou výrazu
  - Všechny proměnné, které se (případně) vyskytují ve Výrazu, musí být v okamžiku splňování cíle inicializované (vázané) na číselnou hodnotu jinak dojde k chybě!
  - ?- X is 1+4.
    - X = 5
  - ?- 5 is 1+4.
    - true

Přiřazování, porovnávání, unifikace

- Operátor =
  - Uspěje, pokud se oba argumenty povede unifikovat
    - ?- X = 1+4.
      - X = 1+4
  - Pokud je jedna proměnná specifikovaná a druhá nespecifikovaná, stanou se obě proměnné specifikované stejnou hodnotou
    - ?- X is 5, X = Y.
      - X = Y, Y = 5
  - Pokud jsou obě proměnné specifikované, porovnají se
    - ?- X is 5, Y is 6, X = Y.
      - false

Přiřazování, porovnávání, unifikace

- Operátor =
  - Jsou-li obě proměnné nespecifikovány, stanou se sdílenými. Stane-li se později jedna z nich specifikovanou, bude stejnou hodnotou specifikována i druhá proměnná
    - X = Y, Y = Z, Z is 1+4, write(X).
      - 5
  - Dvě struktury jsou si rovny, pokud se jedná o dva stejné funktory, souhlasí počet parametrů a jednotlivé termy jsou si rovny
    - ?- 1+2 = 1+2.
      - true
    - ?- 1+2 = 2+1.
      - false

Přiřazování, porovnávání, unifikace

- Operátor =:= (rovno)
  - Provede výpočet, neprovede unifikaci → nelze ho použít na volnou proměnnou
    - ?- 1+2 =:= 2+1.
      - true
    - ?- X =:= 1.
      - Arguments are not sufficiently instantiated
- Operátor == (totožnost)
  - Neprovádí výpočet, pouze porovná
  - Slouží pro porovnání řetězců
    - ?- 5 + 6 == 5 + 6.
      - true
    - ?- 6 + 5 == 5 + 6.
      - false

Přiřazování, porovnávání, unifikace

- Další operátory porovnání
  - Aritmetické
    - <, >, =<, <= ← pozor kde se píše rovnítko!
- Tabulka s porovnáním operátorů

Predikát	Význam	Provádí unifikaci?	Provádí výpočet?
`=`	shoduje se	ano, obě strany	ne
`\=`	neshoduje se, totéž jako not(...=...)	ne	ne
`is`	přiřazení	ano, levá strana	ano, pravá strana
`=:=`	vyhodnotí, ověří shodu	ne	ano, obě strany
`=\=`	vyhodnotí, ověří neshodu	ne	ano, obě strany
`==`	pouze porovná, ověří shodu	ne	ne
`\==`	pouze porovná, ověří neshodu	ne	ne

Aritmetické operátory - funktory

- Vrací hodnotu
- Pro přiřazení do proměnné se použije is
  - ?- X is 5 ** 2
- Infixová i prefixová varianta
- +, -, *, /, ** (mocnina)
- // (celočíselné dělení), rem (zbytek po celočíselném dělení //)
- div (celočíselné dělení se zaokrouhlením dolů) , mod (zbytek po celočíselném dělení div)
- max (maximum), min (minimum)
  - Pouze prefixová varianta
  - Arita 2 – max(term_cislo_1, term_cislo_2).

Bitové operátory - funktory

- >> (bitový posun vpravo), << (bitový posun vlevo)
- \ (bitová negace) – unární funktor
  - ?- X is \ 1.
- \/ (bitové OR), /\ (bitové AND)
- xor (bitový XOR)

Testování typů údajů

Při práci s aritmetickými výrazy (ale i jindy) potřebujeme znalost o skutečném typu hodnoty uložené do proměnné či jiného termu. K tomuto účelu slouží vestavěné predikáty

Predikát	**Význam – testuje, zda je** `argument` **...**
`atom(argument)`	atom, také řetězcová konstanta
`atomic(argument)`	atom, konstantní číslo, řetězec
`integer(argument)`	celé číslo, také v proměnné
`float(argument)`	racionální číslo, také v proměnné
`number(argument)`	číslo, také v proměnné
`string(argument)`	řetězec, také v proměnné
`var(argument)`	volná proměnná
`nonvar(argument)`	vázaná proměnná

Testování typů údajů

- Volná vs. vázaná proměnná
  - Vázaná proměnná je asociovaná s konkrétní hodnotou
  - Volná proměnná nemá stanoveno co obsahuje
    - ?- X=2,var(X).
      - false
    - ?- X=2,nonvar(X).
      - X = 2

Jednoduchý vstup a výstup

- write/1 – vypíše svůj parametr, nevyhodnocuje ho
- display/1 – provede něco podobného, výraz chápe jako výraz, ale nevyhodnotí ho
- nl/0 – přechod na nový řádek (new line)
- read/1 – načte vstup od uživatele

Negace

- Prologu lze využívat pouze Hornovy klauzule
  - Maximálně jeden pozitivní literál → maximálně jeden atom v konsekventu → maximálně jeden atom v hlavě klausule (pravidla)
    - Nemůžeme libovolně přesouvat atomy z antecedentu (tělo) do konsekventu (hlava) a naopak
- Speciální predikát not
  - not(p(a,Y)).
    - Nejdříve se vyhodnotí predikát p → true/false → hodnota se převrátí
    - Když p je true → fakt je odvoditelný z báze
      - not(p) → negace není odvoditelná z báze