Organizační úvod

Poznámka (Organizační úvod)
Bude Moodle. DL1. Kódem je kód cvičení.

Prerekvizity nejsou formální, ale je vhodné mít za sebou ADS1 + logiku.

Zkouška má povinnou písemnou část s materiály z přednášky a překladači Prologu a Haskellu. Zadání i odevzdání prostřednictvím Moodle UK. Zápočet ke zkoušce je doporučený.

Zápočet: Zápočtový program v Prologu / Haskellu + cvičící si mohou volit další podmínky.

1 Úvod

Definice 1.1 (Neprocedurální programování)

Programování bez přiřazovacího příkazu. Lze tak programovat dvěma způsoby: Logické programování = popisujeme problém, který chceme řešit prostředky matematické logiky (Prolog). Funkcionální programování = program je definice funkcí, výpočet je pak aplikace funkce na argumenty (Haskell, Lisp).

Definice 1.2 (Prolog)

U zrodu stáli Robert Kowalski (Edinburgh) a Alain Colmerauer (Marseille).

Aplikace: výuka a výzkum, zpracování přirozeného jazyka, AI, automatické dokazování vět, expertní systémy, dotazovací systémy, systémy řízení, …

Existuje ohromné množství implementací. My budeme používat SWI Prolog.

Definice 1.3 (Syntaxe Prologu)

Před tečkou je term = funktor použitý na konstantu (např. zena(eva). nebo rodic(eva, kain).). Tím (několika řádky začínající vždy stejným funktorem) jsme vytvořili proceduru, která definuje predikát (funktor/počet parametrů) tak, že dává true právě tehdy, když je použitý na tuto konstantu.

Funktor použitý na proměnnou (standardně s velkým prvním písmenem) je definice proměnné.

Čárka je konjunkce. Disjunkce se píše jako středník (využívá se také, když chceme vrátit další možné splnění formule). Konjunkce má vyšší prioritu.

Definice pomocí predikátoru se dělá pomocí `:-`.

Nerovná se značí `\==`. (Ve skutečnosti zjišťuje, zda se dají termy unifikovat, a pokud

ano, tak vrátí false.)

Kanonický tvar si můžeme vypsat pomocí predikátu `display/1`.

`\=` je operátor, který neuspěje (vrátí `false`), pokud se jeho strany dají unifikovat.

`_` je anonymní proměnná („může na jejím místě být cokoliv“).

`+-*` jsou binární operátory, kterým můžeme přiřadit význam.

Definice 1.4 (Unifikace)

Dva termy lze unifikovat, pokud jsou identické, nebo se stanou identickými po substituci vhodné hodnoty do proměnným v obou termech.

Prolog se vždy snaží najít nejobecnější substituci (tedy dvě proměnné položí rovné a nebude za ně nic dosazovat, pokud to stačí). Dělá to tak, že unifikuje právě tehdy, když

- oba termy jsou stejné konstanty,
- jeden term je proměnná a druhý konstanta (pak se dosadí),
- oba termy jsou proměnné (pak se položí rovny),
- oba termy mají stejnou hlavu (první funktor) a její argumenty se dají unifikovat (rekurze).

Definice 1.5 (Algoritmus splňování cíle)

Unifikační algoritmus + backtracking.

Pozor
Záleží na pořadí.

Platnost proměnné je omezena na pravidlo.

Poznámka
V komentářích se `+` označuje vstupní argument (musí to být základní term bez volných proměnných) a `-` argument výstupní (proměnná).

Definice 1.6 (Seznam)

`[]` je prázdný seznam, neprázdný sezname napíšeme jako `[a, b, …]`.

Se seznamy se pracuje hlavně za pomoci `|`, kterým se oddělí konečný počet prvních termů: `[HLAVA1, HLAVA2, … | ZBYTEK]`.

Asociované seznamy se vyrábí tak, že si definujeme nějaký binární operátor (ať už `+-*` nebo libovolný funktor s dvěma argumenty) a v seznamu pak bude tento operátor

aplikovaný na dvojici klíč–proměnná.

První prvek se tedy bere pomocí `|`, poslední se musí získat rekurzí (tedy je to pomalé).

Existuje `member/2`, `select/3` (aplikuje se na prvek, seznam a seznam, prostřední seznam se rozpadne na jeden z jeho prvků a zbytek), `delete/3` (jako select, ale vypustí všechny „instance“ prvku, pozor, má jiné pořadí argumentů), `last/2`, `append/3` (argumenty jsou 3 seznamy, první dva se řetězí na třetí), `\permutation/2`.