Praktikum z logického programování (kombi) - Cvičení č. 3

KOMBINOVANÉ: SEZNAMY

Co je to seznam (obecně)?

- Abstraktní datový typ (ADT)
  - Kolekce prvků
  - Kolekce operací nad těmito prvky
    - Vložení prvku
    - Odstranění prvku
    - Vyhledání prvku
    - Obecný průchod
    - …
  - Možné implementace pomocí
    - Pole
    - Spojový seznam
    - …

Seznamy v Prologu

- Rekurzivní dynamická struktura skládající se z prvků jakéhokoliv typu
  - Délka není pevně stanovena
  - Součástí může být i seznam ← podseznam → složitější struktury jako například stromy
- Induktivní definice:
  - **[ ]** je prázdný seznam
  - Pokud **H** je jakýkoli prvek a **T** je seznam, pak **[H | T]** je také seznam.
    - **H** – **hlava, čelo** nebo **head**
    - **T** – **tělo** nebo **tail**
- Operátor **|** umožňuje přístup k jednotlivým částem seznamu

Seznamy v Prologu

- Názorně pomocí binárního operátoru tečka
  - Reprezentace neprázdných seznamů → tečkový pár
  - **[ ]** je prázdný seznam
  - Neprázdný seznam je poté **. (a , [ ])**
  - **. (b, . (a , [ ])** je také seznam
- Možná vyjádření seznamu o čtyřech prvcích
  - [a, b, c, d]
  - [a | [b, c, d]]
  - [a, b | [c, d]]
  - .(a, [b, c, d])
  - .(a, .(b, .(c, .(d, []))))

Vlož prvek na začátek seznamu

- vloz(?Prvek, ?Seznam, ?VyslednySeznam)
  - **vloz(P, Seznam, [P|Seznam]).**
  - **P** je prvek, který chceme přidat na začátek výsledného seznamu
  - **Seznam** je seznam, ke kterému chceme na začátek přidat prvek **P**
- V rámci klauzule:
  - **Prvek = a, Seznam = [x, y, z],**
    NovySeznam = [Prvek | Seznam].

Je prvek členem seznamu?

- Vestavěný predikát member/2
- Prvek(?HledanyPrvek, ?SeznamVstup)
- **prvek(X, [X|_]) :- !.**
  - Porovnáme hledaný prvek **X** s prvkem na čele seznamu
  - Tělo seznamu nás nezajímá _
  - Pokud **true**, dále neprohledáváme **!**
- **prvek(X, [_|T]) :- prvek(X, T).**
  - První predikát nám dal odpověď, jestli je prvek na čele seznamu
  - Zde nás nezajímá čelo seznamu _, ale zajímá nás tělo pro které provádíme rekurzivní hledání

Spojení dvou seznamů

- Vestavěný predikát append/3
- spoj(?Seznam1, ?Seznam2, ?VystupniSeznam)
- **spoj(L1, L2, L) :- L1 = [], L = L2.**
  - Ekvivalentně: **spoj([], L, L).**
  - Pokud je první seznam prázdný, na výstup dej druhý seznam
- **spoj(L1, L2, L) :- L1 = [H|T1], L = [H|T], spoj(T1, L2, T).**
  - Ekvivalentně: **spoj([H|T1], L2, [H|T]) :- spoj(T1, L2, T).**
  - Vlož prvek z čela prvního seznamu do čela výstupního seznamu
  - Pro tělo prvního seznamu a druhý seznam proveď rekurzi a získej tělo výstupního seznamu

Odstranění prvku seznamu

- Vestavěný predikát delete/3 – deprecated!
  - Je hodně způsobů jakým můžeme odstranit prvek/prvky
- odstran(?Co, ?ZeSeznamu, ?Vysledek).
  - Odstraní všechny výskyty
- **odstran(_, [], []) :- !.**
  - Pokud máme prázdný seznam, vrátíme ho
- **odstran(X, [X|T], L) :- odstran(X, T, L), !.**
  - Pokud jsme na čele seznamu narazili na hledaný prvek, zahodíme jej (hlava seznamu se již nepoužije) a pokračuj rekurzivně s tělem, protože může obsahovat další výskyty
- **odstran(X, [H|T1], [H|T2]):- odstran(X, T1, T2).**
  - Pokud v čele seznamu není hledaný prvek, přidej prvek z čela seznamu vstupního na čelo seznamu výstupního a rekurzivně pokračuj s tělem

Délka seznamu

- Kolik prvků má seznam na jedné úrovni – bez podseznamů
- Vestavěný predikát length/2
- Delka(+Seznam, -Delka)
- **delka([], 0).**
  - Prázdný seznam má délku 0
- **delka([_|T], N) :- delka(T, N1), N is N1 + 1.**
  - Nezajímá nás hlavička, přes tělo procházíme rekurzivně až do prázdného seznamu a při navracení inkrementujeme čítač

Je seznam lineární?

- Obsahuje seznam pouze atomické prvky, tedy ty, které nejsou strukturou?
- **atomic(Term)** – vrací **true**, pokud je term vázaný (tj. není volnou proměnnou) a není složený
  - atomické datové typy: **atom** (atom/1), **řetězec** (string/1), **celé číslo** (integer/1), **číslo s plovoucí desetinnou čárkou** (float/1), racionální číslo (rational/1) a blob (blob/2).
  - Symbol **[ ]** (prázdný seznam) je atomický, ale není atom
- **linear([]).**
- **linear([H|T]):- atomic(H), linear(T).**

Podseznam

- Jak zjistíme, že jeden seznam je podseznamem jiného?
  - Seznam má následující strukturu: **prefix, část, suffix**
- podseznam(?Cast, ?Celek).
- **podseznam(Cast,Celek) :-**
  append(_,Pom,Celek),
  append(Cast,_,Pom),
  Cast \= [],
  !.
- Vyzkoušíme si, jak to funguje, budeme krokovat!

Úkoly

1. Vlož prvek na začátek seznamu, pokud seznam daný prvek ještě neobsahuje
   - Prototyp: prepend_unique(+Element, +List, -NewList)
2. Vlož prvek na konec seznamu
   - Prototyp: appendx(+Element, +List, -NewList)
3. Počet atomických prvků v seznamu
   - Prototyp: count_atomic(+List, -Count)
   - Využijte vhodný pomocný predikát zjišťující atomičnost
4. Poslední prvek seznamu
   - Prototyp: last_member(+List, -Element)
5. Předposlední prvek seznamu
   - Prototyp: penultimate_element(+List, -Element)