Prefixové, infixové a postfixové výrazy

V matematické logice, teoretické informatice, ale i v aplikacích computer-science se lze setkat s výrazy v prefixové, infixové i postfixové notaci. Předpokládám, že prefixová a infixová notace je čtenářům již známa ze základních kursů paradigmat programování (I. semestr studia MI) a matematika (I. pololetí, I. ročníku ZŠ). Zabývat se proto budeme pouze prací s postfixovými výrazy volitelně obsahujícími symboly zastupující funkce proměnlivé arity a jejich vyhodnocováním.

Výrazy zapsané v postfixovém tvaru neobsahují závorky, pořadí interpretace operací je jednoznačně dáno strukturou výrazu a aritou jednotlivých operací. V postfixovém tvaru lze navíc výrazy velice jednoduše vyhodnocovat. Mezi známé badatele používající postfixový zápis v matematice patřil vynikající polský matematik a filosof Jan ŀ ukasiewicz, který v 30. letech minulého století zapisoval logické formule v postfixové notaci.

Definujme nyní strukturu a interpretaci postfixového výrazu.

Definice (postfixový výraz).

1. Libovolné *číslo*, je postfixový výraz, libovolný *symbol*, je postfixový výraz.
2. Je-li f symbol navázaný na funkci *pevné* arity n (v jazyku Scheme se jedná například o funkci vracející celočíselný zbytek po dělení) a $e_1, \ldots, e_n$ jsou postfixové výrazy, pak je sekvence $e_1, \ldots, e_n, f$ postfixový výraz. Je-li f symbol navázaný na funkci *proměnlivé* arity n (v jazyku Scheme se jedná například o vnitřní funkci sčítání) a $e_1, \ldots, e_k$ jsou postfixové výrazy, pak je sekvence $e_1, \ldots, e_k, k, f$ postfixový výraz.
3. Nic jiného není postfixový výraz.

Rozdíl v manipulaci se symboly zastupujícími procedury pevné a proměnlivé arity je jasný. Pokud má operace proměnlivou aritu, arita musí být ve výrazu uvedena kvůli jeho jednoznačnosti. Například dva různé prefixové výrazy jazyka Scheme `(/ 3 4 (+ 2 4 6))` a `(/ 3 4 2 (+ 4 6))` mají pochopitelně zcela jiný význam. V postfixu mají tyto dva výrazy tvar `(3 4 2 4 6` *3* `+` *3* `/)` a `(3 4 2 4 6` *2* `+` *4* `/)`. To jest pokud by ve výrazech nebyla uvedena arita (zde je zvýrazněna kursívou), oba seznamy by byly identické, to je ale nemyslitelné vzhledem k jejich rozdílnému významu – vyhodnocením prvního získáme `1/16`, vyhodnocením druhého `3/80`. Uvádění arity je jistá daň, kterou musíme zaplatit za absenci závorek.

Jedním z alternativních pohledů na postfixový výraz je jeho lineární charakter. Vezmeme-li prefixový/infixový výraz representovaný stromem a linearisujeme-li jej post-order, získáme právě ekvivalentní postfixový výraz.

Interpretace postfixových výrazů je jednoduchá a používá se při ní jeden *zásobník.*

Interpretace výrazu.

- Pokud je na vstupu *číslo,* přesuneme jej na vrchol zásobníku. Pokud je na vstupu *symbol,* na který *není navázána funkce,* přesuneme navázanou hodnotu na vrchol zásobníku.
- Pokud je na vstupu symbol navázaný na n-ární funkci f, odebereme horních n prvků $x_1, \ldots, x_n$ ze zásobníku, vyhodnotíme výsledek $y = f(x_1, \ldots, x_n)$ a výslednou hodnotu y uložíme na vrchol zásobníku. Symbol odebereme ze vstupu.
- Pokud je na vstupu symbol navázaný na funkci *f proměnlivé arity,* odebereme vrchol zásobníku, který považujeme za aktuální počet argumentů k. Potom odebereme dalších k prvků $x_1, \ldots, x_k$ ze zásobníku, vyhodnotíme výsledek $y = f(x_1, \ldots, x_k)$ a výslednou hodnotu y uložíme na vrchol zásobníku. Symbol odebereme ze vstupu.
- Po zpracování celého vstupu obsahuje zásobník jediný prvek – *výsledek vyhodnocení.* Pokud během vyhodnocování zásobník „podteče," nebo po zpracování vstupu je zásobník prázdný, nebo obsahuje víc jak jeden prvek, zadaný výraz nebyl korektní postfixový výraz a vyhodnocení *končí chybou.*

Při práci s postfixovými výrazy v jazyku Scheme je výhodné representovat je jako lineární seznamy skládající se pouze s čísel a symbolů. Tuto representaci zvolíme i my pro naše výukové účely. Pro zjištění arity operací navázaných na symboly lze s výhodnou použít procedury `eval`, `procedure-arity` a predikáty `number?` a `procedure?`, jejich popis naleznete v dokumentaci k interpretu.