NMIN102

Programování 2

Práce s výrazy

RNDr. Michal Žemlička, Ph.D.

Průchod stromem

Celým binárním vyhledávacím stromem (a tedy i AVL-stromem či červeno-černým stromem) můžeme systematicky procházet více způsoby:

preorder – nejdříve provdeme operaci v uzlu, který jsme navštívili, pak zpracujeme levý podstrom, pak pravý;

inorder – nejdříve zpracujeme levý podstrom, pak uzel, v němž jsme, nakonec pravý podstrom;

postorder – nejdříve zpracujeme uzly v levém a právém podstromě, nakonec samotný uzel, v němž jsme.

Notace aritmetických výrazů

Aritmetický výraz můžeme zapisovat více způsoby. Ukažme si ty nejčastější:

infixová notace – oprátor zapisujeme mezi operandy; musíme řešit, co vyhodnotit dříve (dohodou, závorkami; příklad: 2 + 3)

prefixová notace – píšeme nejdříve operátor, pak jeho operandy (např.: + 2 3)

postfixová notace – nejdříve uvádíme operandy, pak teprve operátor (např.: 2 3 +)

Průchody stromem a notace aritmetických výrazů

Zaznamenáme-li aritmetický výraz do binárního stromu tak, že v listech jsou hodnoty a v nelistových uzlech operátory, můžeme výše uvedenými třemi průchody zapsat daný výraz postupně ve všech třech notacích.

Vstup a výstup různých notací

- Aritmetický výraz umíme (pomocí různých průchodů stromové reprezentace) vypsat ve všech třech notacích.
- Jak je ale máme načíst?

Načtení výrazu v prefixové notaci

- Podíváme se, co je na vstupu:
  - je-li to číslo, načteme listový uzel a vrátíme se (přejdeme na další volný odkaz)
  - je-li to operátor, načteme jej, vytvoříme pro něj uzel a přejdeme na jeho levý odkaz an podstrom

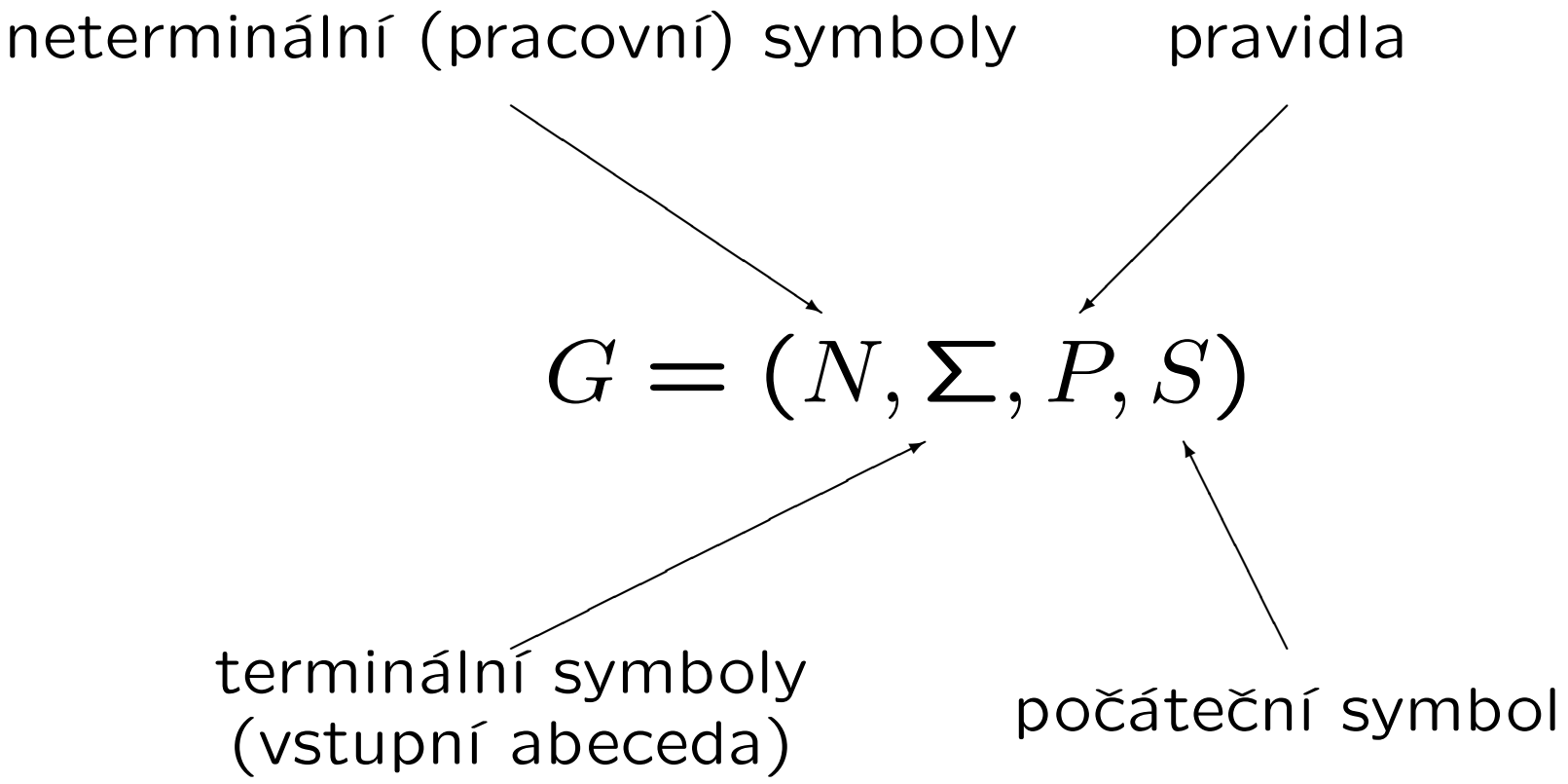
Načtení výrazu v postfixové notaci

- Podíváme se, co je na vstupu:
  - je-li to číslo, načteme listový uzel a uložíme jej na zásobník
  - je-li to operátor, načteme jej, vytvoříme pro něj uzel a příslušný počet podstromů vezmeme z vrcholu zásobníku, napojíme je jako potomky uzlu a uložíme výsledný strom na zásobník

Načtení výrazu v infixové notaci

- můžeme postupnými průchody hledat operátory nejvyšší úrovně, dělit dle něj výraz na části a rekursivně je analyzovat
- ... nebo můžeme zkusit najít jiný postup, jak poznat, co kam patří

Gramatika


neterminální (pracovní) symboly
pravidla
$G = (N, \Sigma, P, S)$
terminální symboly
(vstupní abeceda)
počáteční symbol

Výraz – vstupní abeceda (Σ)

Abychom mohli rozpoznávat jednoduché aritmetické výrazy, hodí se nám rozpoznávat tyto jednodušší části:

- čísla – posloupnosti za sebou stojících číslic
- operátory (+,-,*,/)
- konec vstupu

Výraz – pracovní abeceda (N) a startovní symbol

Aritmetický výraz můžeme dekomponovat na tyto části:

E výraz (expression) – součty a rozdíly členů

T člen (term) – součiny a podíly faktorů

F faktor – hodnota nebo výraz v závorce

Startovním symbolem (tím, odkud začneme analyzovat) bude E.

Pravidla gramatiky jednoduchého aritmetického výrazu (P)

$$
\begin{aligned}
E &\rightarrow T \\
E &\rightarrow \text{-}T \\
E &\rightarrow E + T \\
E &\rightarrow E \text{ - } T \\
T &\rightarrow F \\
T &\rightarrow T * F \\
T &\rightarrow T \,/\, F \\
F &\rightarrow \mathit{num} \\
F &\rightarrow (\ E\ )
\end{aligned}
$$

Rozdělení pravidel gramatiky jednoduchého aritmetického výrazu

E → T E → -T	E → E + T E → E - T
T → F	T → T * F T → T / F
F → *num* F → (E)	

ještě trošku upravíme ....

E	T -T	E + T E - T
T	F	T * F T / F
F	*num* (E)	

... doplníme konce ....

Abychom vždy poznali, jaké pravidlo jsme dořešili, napíšeme si jeho značku na jeho konec

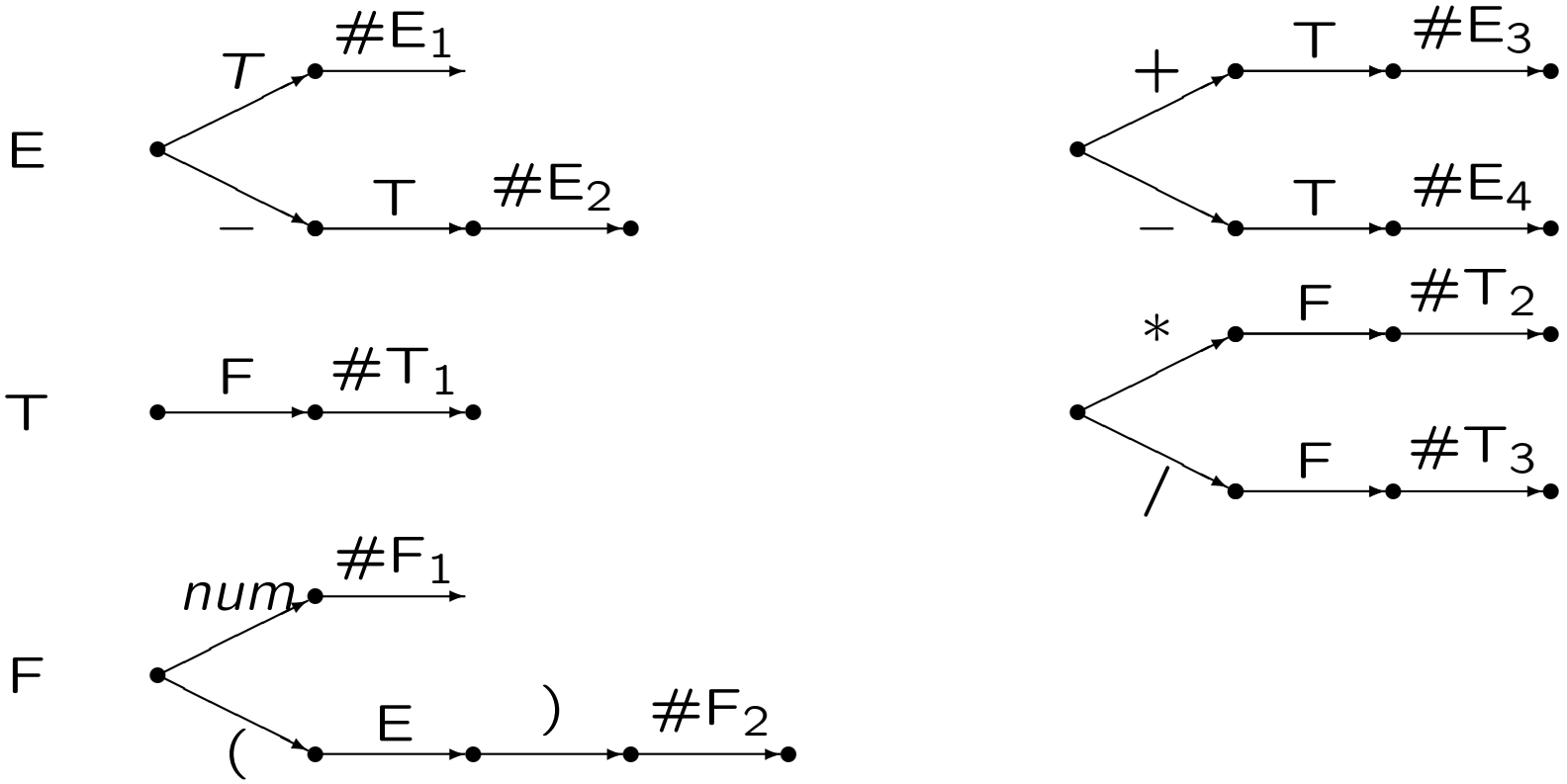
E	T $\#E_1$ -T $\#E_2$	E + T $\#E_3$ E - T $\#E_4$
T	F $\#T_1$	T * F $\#T_2$ T / F $\#T_3$
F	*num* $\#F_1$ (E) $\#F_2$	

... ještě zjednodušíme ....

Ve skupinách napravo vždy všechna pravidla začínají stehným symbolem. Odstraníme je ...

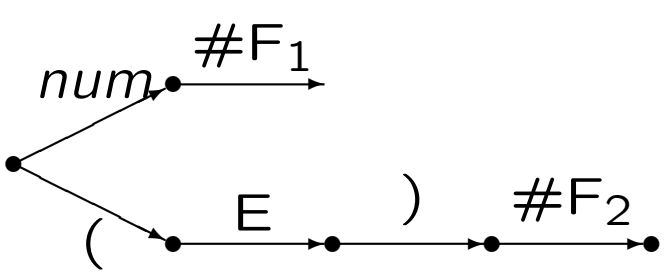
E	T #E_1 -T #E_2	+ T #E_3 - T #E_4
T	F #T_1	* F #T_2 / F #T_3
F	*num* #F_1 (E) #F_2	

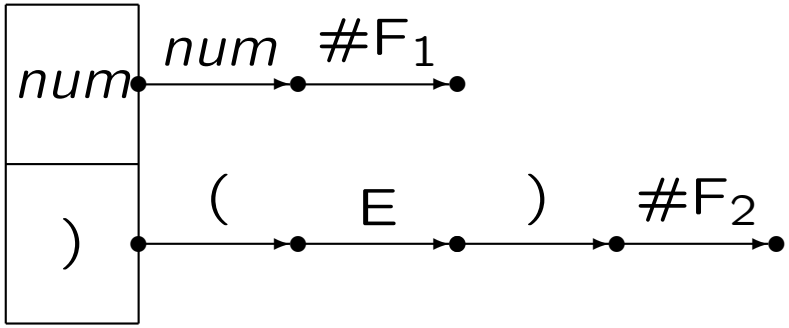
Uděláme stromečky ....



Volba větve výpočtu ....

Abychom věděli, kterou větví se dát, uzel zvětšíme. Před každou větev si napíšeme symboly, které jsou pro danou větev přípustné. Předpokládáme, že různé větve připouštějí výhradně různé symboly.





Jak to ale zakódovat?

- Jednomu symbolu bude odpovídat jeden podprogram
- Větvení ošetříme pomocí CASE nebo IF (podle počtu větví a složitosti podmínky).
- Pozor, větvění může být více, každé z nich je pak třeba ošetřit ...
- Jsou-li stromečky dva, druhý vložíme do WHILE cyklu, kde v podmínce je, zda je k ve výhledu některý ze symbolů z větvícího uzlu, případně jímž může pravidlo začínat
- Pro analýzu voláme ten podprogram, který odpovídá startovnímu symbolu
- *Analýza rekurzivním sestupem*