Prednášky z Úvodu do deklaratívneho programovania

Ján Kľuka

Letný semester 2010/2011

Obsah

I. Organizácia kurzu Deklaratívne programovanie v CL Explicitné definície **3**

1. Organizácia kurzu **3**
1.1. Rozvrh a kontakty . . . 3
1.2. Pravidlá hodnotenia . . . 3

2. Deklaratívne programovanie **3**
2.1. Príklad . . . 4

3. Jazyk CL **5**

4. Explicitné definície **5**
4.1. S jednou podmienkou . . . 5
4.2. S viacerými podmienkami . . . 6

II. Rekurzia **7**

5. Rekurzívne definície **7**
5.1. Primitívna rekurzia . . . 7
5.2. Course-of-values rekurzia . . . 8

III. Priraďujúce diskriminácie, simulácia cyklov chvostovou rekurziou **9**

6. Priraďujúce diskriminácie **9**
6.1. Priradenie . . . 9
6.2. Monadická diskriminácia . . . 9

7. Simulácia cyklov chvostovou rekurziou **10**
7.1. Simulácia `while` cyklu . . . 10
7.2. Efektívny výpočet Fibonacciho postupnosti . . . 11
7.3. Simulácia `for` cyklu . . . 13

VI. Párovacia funkcia CL, *n*-tice a zoznamy **14**

8. Párovacia funkcia CL **14**
8.1. Programovanie s párovacou funkciou . . . 14
8.2. Tupling . . . 15

9. Zoznamy **15**
9.1. Programovanie so zoznamami . . . 16

VII. Tupling a simulácia cyklov na zoznamoch. Predikáty na zoznamoch 18

10. Tupling zoznamových operácií 18
10.1. Split = Take,Drop . . . 18
10.2. Zip, Unzip . . . 18

11. Simulácia cyklov na zoznamoch 19
11.1. Súčet prvkov zoznamu . . . 19
11.2. Obrátenie zoznamu . . . 19

12. Predikáty na zoznamoch 20
12.1. Predikáty . . . 20
12.2. Prefix . . . 21
12.3. Suffix . . . 22
12.4. Substring . . . 22

VIII. Triedenie zoznamov Zoznamová reprezentácia množín 23

13. Triedenie zoznamov 23
13.1. Špecifikácia triedenia . . . 23
13.2. Triedenie vsúvaním . . . 24
13.3. Triedenie zlučovaním . . . 25

14. Zoznamová reprezentácia konečných množín 26

IX. Binárne stromy 27

15. Binárne stromy 27
15.1. Kódovanie stromov párovacími konštruktormi . . . 27
15.2. Operácie na binárnych stromoch . . . 28
15.3. Binárne vyhľadávacie stromy . . . 29

XI. Číslovanie binárnych stromov Kombinatorika na zoznamoch 32

15. Binárne stromy 32
15.3. Binárne vyhľadávacie stromy (dokončenie) . . . 32
15.4. Číslovanie vrcholov v binárnych stromoch . . . 33

16. Kombinatorika na zoznamoch 34
16.1. Podpostupnosti . . . 34
16.2. Ďalšie úlohy . . . 35

XII. Postfixový stroj a numerické výrazy 36

17. Postfixový stroj a numerické výrazy 36
17.1. Postfixový stroj . . . 36
17.2. Numerické výrazy s premennými . . . 37
17.3. Kompilácia . . . 39

XIII. Generovanie XML/XHTML 41

18. XML/XHTML 41
18.1. Štruktúra . . . 41
18.2. Kódovanie . . . 42
18.3. Neformálne typovanie . . . 43

19. Organizácia: Opravný test a skúška 44

I. prednáška
Organizácia kurzu
Deklaratívne programovanie v CL
Explicitné definície

15. februára 2011

1. Organizácia kurzu

1.1. Rozvrh a kontakty

I.1 Organizácia kurzu

Prednášky utorok 11:30, poslucháreň B

Ján Kľuka

Cvičenia

streda	8:10	H6	1iai{1,2,6}
	16:30	H6	1iai{3,4,5}
	16:30	F1-248	2iai* (max. 24)
	18:10	H6	2iai*, 3iai*

Dodržte termín cvičení podľa ročníka a krúžku! (Inak ste hostia, nemôžeme sa vám venovať.)

Prváci: Možná výmena človek za človeka (e-mail↓)

Cvičenia o 16:30 v F1-248:

- iba 2iai*, max. 24 študentov
- nutné prihlásiť sa (e-mail↓)
- priorita podľa dátumu a času: 1. zápisu, 2. prijatia prihlasovacieho e-mailu
- deadline: 23. 2. 2011 15:00

Konzultácie streda 14:00, I-33b

Web http://dai.fmph.uniba.sk/courses/udp/

E-mail *udp(zavináč)lists.dai.fmph.uniba.sk*

1.2. Pravidlá hodnotenia

I.2 Organizácia kurzu

Semester
- *2* testy po *30* bodov, 6./7. a 11. týždeň
- Opravný test, 1. týždeň skúškového obdobia
  - doplnenie do 30 bodov
  - ak získate za semester aspoň 10 bodov
- Podmienka pripustenia ku skúške zisk $\geq$ 30 bodov

Skúška
- Test za *40* bodov

Známky

A	$\geq$ 90 bodov
B	$\geq$ 80 bodov
C	$\geq$ 70 bodov
D	$\geq$ 60 bodov
E	$\geq$ 50 bodov
FX	$<$ 50 bodov

2. Deklaratívne programovanie

I.3 Deklaratívne programovanie

- Spôsob (paradigma) programovania
- Dôraz na to, *čo* sa počíta, nie (len) na to, *ako* sa počíta
- Založené na matematickej logike

- Program je zapísaný v nejakom jazyku logiky
- Význam programu: matematický objekt Program *definuje* funkciu/reláciu
- *Výhoda:* uľahčuje tvorbu *správnych* programov
- Deklaratívne programovacie jazyky
  - funkcionálne (Lisp, ML, Haskell, . . .)
  - logické (Prolog, Trilogy, . . .)

I.4 Tvorba správnych programov

Tvorba správneho programu má 3 časti:

Špecifikácia požadované vlastnosti programu, aké výstupy má počítať pre aké vstupy

Implementácia samotný program, podrobný návod na počítanie výstupov

Verifikácia overenie, že program má požadované vlastnosti (spĺňa špecifikáciu)

Deklaratívne programovanie:

- Všetky časti tvorby správneho programu sú realizované v jednom jazyku logiky

2.1. Príklad

I.5 Príklad

Neformálne zadanie

- Potrebujeme program, ktorý vypočíta *najväčší spoločný deliteľ* dvoch prirodzených čísel

Špecifikácia

- Presný popis zadania v logickom jazyku
- Program v deklaratívnom jazyku bude definovať funkciu gcd s vlastnosťou:

$$x \neq 0 \vee y \neq 0 \rightarrow \gcd(x,y) \mid x \wedge \gcd(x,y) \mid y \wedge \\ \forall d(d \mid x \wedge d \mid y \rightarrow d \leq \gcd(x,y))$$

I.6 Príklad

Implementácia

- Program realizuje euklidovský algoritmus
- Využíva vlastnosti deliteľnosti:

$$x \mid 0$$
$$x \neq 0 \wedge z \mid y \rightarrow z \mid y \leftrightarrow z \mid y \bmod x$$

- Program v jazyku podmienených rovností:

$$\gcd(x,y) = y \qquad \leftarrow x = 0$$
$$\gcd(x,y) = \gcd(y \bmod x, x) \leftarrow x \neq 0$$

- **Výpočet:** Zjednodušovanie výrazov podľa podmienených rovností do základného tvaru (číslo)
- Napríklad $\gcd(21,12) = \cdots$
- Výpočet vždy skončí, lebo

$$x \neq 0 \rightarrow y \bmod x < x$$

I.7 Príklad

Špecifikácia – požadovaná vlastnosť

$$x \neq 0 \vee y \neq 0 \rightarrow \gcd(x,y) \mid x \wedge \gcd(x,y) \mid y \wedge \\ \forall d(d \mid x \wedge d \mid y \rightarrow d \leq \gcd(x,y))$$

Implementácia – program v jazyku podmienených rovností

$$\gcd(x,y) = y \qquad \leftarrow x = 0$$
$$\gcd(x,y) = \gcd(y \bmod x, x) \leftarrow x \neq 0$$

Má program požadovanú vlastnosť?

Testovanie *Výpočtom* overíme pre *niektoré* vstupy

Verifikácia *Dôkazom* overíme pre *všetky* vstupy

- Dôkaz úplnou matematickou indukciou

3. Jazyk CL

I.8 Jazyk CL

- Jednoduchý deklaratívny programovací jazyk
- Základ: logika prvého rádu s aritmetikou
- Syntax programov: podmienené rovnosti
- Význam programov: funkcie na prirodzených číslach
- Prostredie pre programovanie, testovanie, špecifikáciu a verifikáciu programov
- Autori: Pavol Voda, Ján Komara, Ján Kľuka

I.9 Niektoré funkcie zabudované v CL

- Sčítanie ($+$) a násobenie ($\cdot$) prirodzených čísel
- Modifikované odčítanie

$$x \dot{-} y = \begin{cases} x - y & \text{ak } x \geq y, \\ 0 & \text{inak} \end{cases}$$

- Delenie a zvyšok po delení na prirodzených číslach

$$x \div 0 = 0 \wedge x \bmod 0 = 0$$
$$y \neq 0 \rightarrow x \bmod y < y \wedge x = (x \div y) \cdot y + (x \bmod y)$$

4. Explicitné definície

I.10 Explicitné definície

Explicitná definícia funkcie

- Program, ktorý vypočíta výsledok ako výraz zložený iba z premenných, číselných konštánt a predtým definovaných funkcií
- Jednoduché explicitné definície majú iba jednu rovnosť, napríklad:

$$Z(x) = 0$$
$$Twice(x) = 2 \cdot x$$
$$Square(x) = x \cdot x$$
$$Cube(x) = x \cdot Square(x)$$

4.1. Explicitné definície s jednou podmienkou

I.11 Explicitné definície s podmienkami

- Zložitejšie definície majú viacero podmienených rovností

$$\mathrm{sgn}(x) = 0 \leftarrow x = 0$$
$$\mathrm{sgn}(x) = 1 \leftarrow x \neq 0$$

$$\min(x, y) = x \leftarrow x \leq y$$
$$\min(x, y) = y \leftarrow x > y$$

- Podmienená rovnosť sa nazýva *klauzula* (clause)
- Argumenty v rôznych klauzulách funkcie musia byť rovnaké premenné
- Podmienky v klauzulách musia byť *výlučné*
- Podmienky musia pokryť *všetky možné prípady*
- Potom sa vždy dá použiť práve jedna klauzula

I.12 Explicitné definície s podmienkami

- CL nedovoľuje v klauzulách hocijaké podmienky
- Niektoré dovolené podmienky:

$$f(\cdots) = \cdots \leftarrow t = s \qquad f(\cdots) = \cdots \leftarrow t \leq s$$
$$f(\cdots) = \cdots \leftarrow t \neq s \qquad f(\cdots) = \cdots \leftarrow t > s$$

  t a s sú výrazy Dovolenú sadu podmienok voláme *diskriminácia*

- Je dovolené napríklad:

$$f(x) = x \quad \leftarrow x{+}7 = x{\cdot}7 \qquad g(x, y) = x{\cdot}y \quad \leftarrow y > x$$
$$f(x) = x{+}1 \leftarrow x{+}7 \neq x{\cdot}7 \qquad g(x, y) = x{+}y \leftarrow y \leq x$$

- Nie je dovolené napríklad:

$$f(x, y) = x{+}y \leftarrow x = y \qquad g(x, y, z) = y{+}z \leftarrow y \leq x$$
$$f(x, y) = x{\cdot}y \quad \leftarrow y \neq x \qquad g(x, y, z) = x{+}z \leftarrow y = x$$

4.2. Explicitné definície s viacerými podmienkami

I.13 Explicitné definície s podmienkami

- Podmienky v klauzulách možno kombinovať logickou spojkou $\wedge$
- Dovolené sú iba niektoré kombinácie podmienok:

$$\min_3(x, y, z) = x \leftarrow x \leq y \wedge x \leq z$$
$$\min_3(x, y, z) = z \leftarrow x \leq y \wedge x > z$$
$$\min_3(x, y, z) = y \leftarrow x > y \wedge y \leq z$$
$$\min_3(x, y, z) = z \leftarrow x > y \wedge y > z$$

- Nie je dovolené napríklad:

$$\min_3(x, y, z) = x \leftarrow x \leq y \wedge x \leq z$$
$$\min_3(x, y, z) = y \leftarrow y \leq z \wedge y \leq x$$
$$\min_3(x, y, z) = z \leftarrow z \leq x \wedge z \leq y$$

I.14 Explicitné definície s podmienkami

$$\min_3(x, y, z) = x \leftarrow x \leq y \wedge x \leq z \quad (1)$$
$$\min_3(x, y, z) = z \leftarrow x \leq y \wedge x > z \quad (2)$$
$$\min_3(x, y, z) = y \leftarrow x > y \wedge y \leq z \quad (3)$$
$$\min_3(x, y, z) = z \leftarrow x > y \wedge y > z \quad (4)$$

Každé dve klauzuly:

- majú rôzne prvé podmienky, ktoré patria do jednej diskriminácie [(1) a (3), (1) a (4), (2) a (3), (2) a (4)], alebo
- majú rovnakú prvú podmienku a ich druhé podmienky patria do jednej diskriminácie [(1) a (2), (3) a (4)]

II. prednáška
Rekurzia

22. februára 2011

II.1 Opakovanie 1. prednášky

- Explicitné definície – skladanie už definovaných funkcií

$$\text{Square}(x) = x{\cdot}x$$
$$\text{Cube}(x) = x{\cdot}\text{Square}(x)$$

- Explicitné klauzálne definície – podmienené výpočty

$$\min(x,y) = x \leftarrow x \leq y$$
$$\min(x,y) = y \leftarrow x > y$$

Diskriminácie – dovolené kombinácie podmienok

$$\cdots \leftarrow s \leq t \qquad \cdots \leftarrow s = t$$
$$\cdots \leftarrow s > t \qquad \cdots \leftarrow s \neq t$$

vzájomne výlučné, pokrývajúce všetky prípady

$$\neg(s \leq t \wedge s > t) \qquad \neg(s = t \wedge s \neq t)$$
$$(s \leq t \vee s > t) \qquad (s = t \vee s \neq t)$$

▶ Chýba nám *opakovanie*

5. Rekurzívne definície

II.2 Princíp rekurzie

- Opakovanie v deklaratívnom programovaní – rekurzia
- *Rekurzia* vyjadruje hodnotu funkcie pre nejaký argument pomocou hodnoty *tej istej funkcie* pre *jednoduchšie argumenty*

$$0! = 1$$
$$(n+1)! = (n+1){\cdot}n!$$

- Rôzne druhy rekurzie – rôzne zjednodušenia rekurzívnych argumentov

5.1. Primitívna rekurzia

II.3 Primitívna rekurzia

- *Primitívna rekurzia*: rekurzívny argument je o 1 menší
- Súčet čísel od 0 po n:

$$\sum_{i=0}^{n} i = \begin{cases} 0 & \text{ak } n = 0, \\ n + \sum_{i=0}^{n-1} i & \text{inak} \end{cases}$$

- CL definícia funkcie $\text{Sum}(n) = \sum_{i=0}^{n} i$ pomocou diskriminácie na rovnosť:

$$\sum_{i=0}^{n} i = 0 \qquad \leftarrow n = 0$$
$$\sum_{i=0}^{n} i = n + \sum_{i=0}^{n-1} i \leftarrow n \neq 0$$

- Výpočet:

$$\begin{aligned} \sum_{i=0}^{5} i &= 5 + \sum_{i=0}^{4} i = 5 + (4 + \sum_{i=0}^{3} i) = \cdots \\ &= 5 + (4 + (3 + (2 + (1 + \sum_{i=0}^{0} i)))) \\ &= 5 + (4 + (3 + (2 + (1 + 0)))) \\ &= 5 + (4 + (3 + (2 + 1))) = 5 + (4 + (3 + 3)) \\ &= 5 + (4 + 6) = 5 + 10 = 15 \end{aligned}$$

- Čo je potrebné na počítačovú realizáciu výpočtu?
  Zásobník

II.4 Primitívna rekurzia

- Predpokladajme, že máme zabudované iba sčítanie a odčítanie
- Definícia násobenia $\mathrm{Mul}(x, y) = x{\cdot}y$ primitívnou rekurziou:

$$\begin{aligned} x{\cdot}y &= 0 && \leftarrow x = 0 \\ x{\cdot}y &= \cdots (x \dot{-} 1){\cdot}y \cdots && \leftarrow x \neq 0 \end{aligned}$$

5.2. Course-of-values rekurzia

II.5 Course-of-values rekurzia

- Na pripomenutie:
  - Rekurzia vyjadruje hodnotu funkcie pre nejaký argument pomocou hodnoty tej istej funkcie pre *jednoduchšie argumenty*
  - Primitívna rekurzia: rekurzívny argument je o 1 menší
- *Course-of-values* rekurzia: rekurzívny argument je *ľubovoľné menšie* číslo
- Príklad: Vieme iba sčitovať, odčitovať, násobiť, chceme delenie $\mathrm{Div}(x, y) = x \div y$:

$$y \neq 0 \rightarrow \exists r(x = (x \div y){\cdot}y + r \wedge r < y)$$

- Definícia pomocou course-of-values rekurzie:

$$\begin{aligned} x \div y &= 0 && \leftarrow y = 0 \\ x \div y &= 0 && \leftarrow y \neq 0 \wedge x < y \\ x \div y &= ((x \dot{-} y) \div y) + 1 && \leftarrow y \neq 0 \wedge x \geq y \end{aligned}$$

$$y \neq 0 \rightarrow x \dot{-} y < x$$

II.6 Course-of-values rekurzia

- Najväčší spoločný deliteľ:

$$\begin{aligned} x \neq 0 \vee y \neq 0 \rightarrow \gcd(x, y) \mid x \wedge \gcd(x, y) \mid y \wedge \\ \forall d(d \mid x \wedge d \mid y \rightarrow d \leq \gcd(x, y)) \end{aligned}$$

- Využijeme

$$\begin{gathered} x \mid 0 \\ x \neq 0 \wedge z \mid y \rightarrow z \mid y \leftrightarrow z \mid y \bmod x \\ x \neq 0 \rightarrow y \bmod x < x \end{gathered}$$

- Definícia course-of-values rekurziou:

$$\begin{aligned} \gcd(x, y) &= y && \leftarrow x = 0 \\ \gcd(x, y) &= \gcd(y \bmod x, x) && \leftarrow x \neq 0 \end{aligned}$$

II.7 Course-of-values rekurzia

- Fibonacciho postupnosť $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$

$$\begin{aligned} \mathrm{fib}_0 &= 0 \\ \mathrm{fib}_1 &= 1 \\ \mathrm{fib}_{n+2} &= \mathrm{fib}_{n+1} + \mathrm{fib}_n \end{aligned}$$

- Funkcia $\mathrm{Fib}(n) = \mathrm{fib}_n$ počíta n-tý prvok Fibonacciho postupnosti
- Definícia course-of-values rekurziou
  - s *dvoma* rekurzívnymi volaniami, a
  - so zloženými podmienkami:

$$\mathrm{fib}_n = \cdots \leftarrow \cdots$$

III. prednáška
Priraďujúce diskriminácie, simulácia cyklov chvostovou rekurziou

1. marca 2011

6. Priraďujúce diskriminácie

III.1 Známe diskriminácie

Zatiaľ poznáme iba dva druhy diskriminácií (dovolených kombinácií podmienok) v CL:

$$\cdots \leftarrow s \leq t \qquad \cdots \leftarrow s = t$$
$$\cdots \leftarrow s > t \qquad \cdots \leftarrow s \neq t$$

Vzájomne výlučné, pokrývajúce všetky prípady

$$\neg(s \leq t \wedge s > t) \qquad \neg(s = t \wedge s \neq t)$$
$$(s \leq t \vee s > t) \qquad (s = t \vee s \neq t)$$

6.1. Priradenie

III.2 Priradenie

CL umožňuje priradiť hodnotu *novej* premennej

$$\cdots \leftarrow \cdots \wedge t = x$$

Nová premenná na *pravej strane* rovnosti (ako v query)

Diskriminácia s jedinou možnosťou (vždy pravdivou)
Vždy platí:

$$\exists x(t = x)$$

Príklady

$$n! = 1 \quad \leftarrow n = 0$$
$$n! = n{\cdot}m! \leftarrow n \neq 0 \wedge n \dot{-} 1 = m$$

$$\gcd(x, y) = y \quad \leftarrow x = 0$$
$$\gcd(x, y) = \gcd(x_1, y_1) \leftarrow x \neq 0 \wedge y \bmod x = x_1 \wedge x = y_1$$

6.2. Monadická diskriminácia

III.3 Monadická diskriminácia

Monadická diskriminácia –kombinácia testu a priradenia

$$\cdots \leftarrow t = 0 \qquad \cdots \leftarrow t = 0$$
$$\cdots \leftarrow t = x + 1 \qquad \cdots \leftarrow t \neq 0 \wedge t \dot{-} 1 = x$$

x je *nová* premenná, t je ľubovoľný výraz

Podmienky sú vzájomne výlučné a pokrývajú všetky prípady:

$$\neg(s = 0 \wedge \exists x(s = x + 1)) \qquad (s = 0 \vee \exists x(s = x + 1))$$

Príklad

$$n! = 1 \quad \leftarrow n = 0$$
$$n! = n{\cdot}m! \leftarrow n = m + 1$$

Monadickú diskrimináciu môžeme *dosadiť* do argumentov:

$$0! = 1$$
$$(m + 1)! = (m + 1){\cdot}m!$$

III.4 Zovšeobecnená monadická diskriminácia

Zovšeobecnená monadická diskriminácia

$$\cdots \leftarrow t = 0$$
$$\cdots \leftarrow t = 1$$
$$\vdots$$
$$\cdots \leftarrow t = \underline{k - 1}$$
$$\cdots \leftarrow t = x + \underline{k}$$

x je *nová* premenná, k je *konštanta*, t je ľubovoľný výraz

Príklad

$$\begin{aligned}
\text{fib}_n &= 0 && \leftarrow n = 0\\
\text{fib}_n &= 1 && \leftarrow n = 1\\
\text{fib}_n &= \text{fib}_{m+1} + \text{fib}_m && \leftarrow n = m + 2
\end{aligned}$$

Po dosadení do argumentov:

$$\begin{aligned}
\text{fib}_0 &= 0\\
\text{fib}_1 &= 1\\
\text{fib}_{n+2} &= \text{fib}_{n+1} + \text{fib}_n
\end{aligned}$$

7. Simulácia cyklov chvostovou rekurziou

III.5 `While` **cyklus v Pascale**

- Ako by ste programovali výpočet $n!$ v Pascale?
- Napríklad `while` cyklom

```
function Fact(n: Integer): Integer;
var f: Integer;
begin
    f := 1;
    while n <> 0 do begin
        f := f * n;
        n := n - 1
    end;
    Result := f
end;
```

- Ako by ste simulovali tento výpočet v CL?

7.1. Simulácia `while` cyklu

III.6 **Simulácia** `while` **cyklu v CL**

1. Pascalovský program rozdelíme na 2 časti:

a) inicializácia

```
f := 1;
```

b) cyklus a vrátenie výsledku

```
while n <> 0 do begin
    f := f * n;
    n := n - 1
end;
Result := f
```

III.7 **Simulácia** `while` **cyklu v CL**

2. Simulácia cyklu pomocnou funkciou *While_fact*:

- Dva argumenty – premenné n a f
- Keď $n \neq 0$,
  - vykoná sa telo cyklu, teda premenné f a n dostanú nové hodnoty;
  - cyklus sa zopakuje s novými hodnotami.
- Keď $n = 0$,
  - funkcia skončí a vráti hodnotu z premennej f.

$$\begin{aligned}
\mathit{While_fact}(n, f) &= \overbrace{\mathit{While_fact}(n_1, f_1)}^{\text{opakovanie}} \leftarrow \overbrace{n \neq 0}^{\text{podmienka}} \wedge \overbrace{f \cdot n = f_1 \wedge n \dot{-} 1 = n_1}^{\text{telo cyklu}}\\
\mathit{While_fact}(n, f) &= \underbrace{f}_{\text{výsledok}} \leftarrow \underbrace{n = 0}_{\text{negácia podmienky}}
\end{aligned}$$

Invariant:

$$\mathit{While\_fact}(n, f) = f \cdot n!$$

III.8 Simulácia `while` cyklu v CL

3. Inicializácia a spustenie cyklu

- Hlavná funkcia *Fact* spustí simuláciu cyklu s počiatočnými hodnotami premenných n a f.

$$Fact(n) = While\_fact(n, 1)$$

III.9 Simulácia `while` cyklu v CL

```
function Fact(n: Integer): Integer;
var f: Integer;
begin
    f := 1;
    while n <> 0 do begin
        f := f * n;
        n := n - 1
    end;
    Result := f
end;
```

$$While\_fact(n, f) = While\_fact(n_1, f_1) \leftarrow n \neq 0 \wedge f \cdot n = f_1 \wedge n \dot{-} 1 = n_1$$
$$While\_fact(n, f) = f \leftarrow n = 0$$

$$Fact(n) = While\_fact(n, 1)$$

Stručnejšia verzia s monadickou diskrimináciou:

$$While\_fact(n+1, f) = While\_fact(n, f \cdot (n+1))$$
$$While\_fact(0, f) = f$$

III.10 Simulácia `while` cyklu v CL

Priebeh výpočtu obyčajnej definície $n!$:

$$\begin{aligned} 5! &= 5 \cdot 4! \\ &= 5 \cdot (4 \cdot 3!) \\ &= \cdots \\ &= 5 \cdot (4 \cdot (3 \cdot (2 \cdot (1 \cdot 0!)))) \\ &= 5 \cdot (4 \cdot (3 \cdot (2 \cdot (1 \cdot 1)))) \\ &= 5 \cdot (4 \cdot (3 \cdot (2 \cdot 1))) \\ &= \cdots \\ &= 5 \cdot 24 \\ &= 120 \end{aligned}$$

Priebeh výpočtu *Fact*:

$$\begin{aligned} Fact(5) &= While\_fact(5, 1) \\ &= While\_fact(4, 5) \\ &= While\_fact(3, 20) \\ &= \cdots \\ &= While\_fact(0, 120) \\ &= 120 \end{aligned}$$

- *While_fact*: *chvostová* rekurzia – žiadne výpočty po návrate z rekurzívneho volania
- Nepotrebuje zásobník

7.2. Efektívny výpočet Fibonacciho postupnosti

III.11 Fibonacciho postupnosť *ne*efektívne

Na pripomenutie: n-tý prvok Fibonacciho postupnosti

$$\begin{aligned} \mathrm{fib}_0 &= 0 \\ \mathrm{fib}_1 &= 1 \\ \mathrm{fib}_{n+2} &= \mathrm{fib}_{n+1} + \mathrm{fib}_n \end{aligned}$$

Výpočet obyčajnou rekurziou je veľmi neefektívny:

$$\begin{array}{ccccccc} & & & \mathrm{fib}_5 & & & \\ & & \mathrm{fib}_4 & + & & \mathrm{fib}_3 & \\ & \mathrm{fib}_3 & + & \mathrm{fib}_2 & & \mathrm{fib}_2 & + \mathrm{fib}_1 \\ \mathrm{fib}_2 & + \mathrm{fib}_1 & \mathrm{fib}_1 + \mathrm{fib}_0 & & \mathrm{fib}_1 + \mathrm{fib}_0 & & \\ \mathrm{fib}_1 + \mathrm{fib}_0 & & & & & & \end{array}$$

III.12 Fibonacciho postupnosť a králiky

Efektívny výpočet:

- pamätáme si dva prvky postupnosti
- počítame nasledujúci prvok

Analógia – množenie králikov:

- Dostali sme 1 pár králičích mláďat.
- Za 1 rok:
  - každý pár dospelých králikov vychová 1 pár mláďat;
  - každé mláďa narodené predchádzajúci rok dospeje.
- Koľko párov dospelých králikov budeme mať po n rokoch?

rok	mláďatá	dospelí
0	1	$0 = \text{fib}_0$
1	$0 = \text{fib}_0$	$1 = \text{fib}_1$
2	$1 = \text{fib}_1$	$1 = \text{fib}_2$
3	$1 = \text{fib}_2$	$2 = \text{fib}_3$
4	$2 = \text{fib}_3$	$3 = \text{fib}_4$
5	$3 = \text{fib}_4$	$5 = \text{fib}_5$
6	$5 = \text{fib}_5$	$8 = \text{fib}_6$

rok	mláďatá	dospelí
$i{+}1$	$m{=}\text{fib}_i$	$d{=}\text{fib}_{i+1}$
$i{+}2$	$d{=}\text{fib}_{i+1}$	$m{+}d{=}\text{fib}_{i+2}$

III.13 Fibonacciho postupnosť efektívne v Pascale

Efektívny výpočet v Pascale:

```
function Fib(n: Integer): Integer;
var m, d, m1, d1: Integer;
begin
    if n = 0 then
        Result := 0
    else begin
        n := n - 1;
        m := 0; { fib(0) }
        d := 1; { fib(1) }
        while n <> 0 do begin
            { m = fib(i) & d = fib(i+1) }
            m1 := d;
            d1 := m + d;
            m := m1; d := d1;
            { m = fib(i+1) & d = fib(i+2) }
            n := n - 1
        end;
        Result := d
    end
end;
```

III.14 Fibonacciho postupnosť efektívne v CL

Efektívny výpočet v CL:

- Simulácia `while` cyklu:

$$\begin{aligned} &\mathit{While\_fib}(n, m, d) = \mathit{While\_fib}(n \dot{-} 1, m_1, d_1) \\ &\qquad\qquad \leftarrow n \neq 0 \wedge d = m_1 \wedge d + m = d_1 \\ &\mathit{While\_fib}(n, m, d) = d \quad \leftarrow n = 0 \end{aligned}$$

  Stručnejšie

$$\begin{aligned} &\mathit{While\_fib}(n+1, m, d) = \mathit{While\_fib}(n, d, d+m) \\ &\mathit{While\_fib}(0, m, d) = d \end{aligned}$$

  Invariant:

$$\mathit{While\_fib}(n, \text{fib}_i, \text{fib}_{i+1}) = \text{fib}_{n+i+1}$$

- Úvodné testy, inicializácia a spustenie cyklu:

$$\begin{aligned} &\mathit{Fib}(n) = 0 \qquad\qquad\qquad\quad \leftarrow n = 0 \\ &\mathit{Fib}(n) = \mathit{While\_fib}(n \dot{-} 1, 0, 1) \leftarrow n \neq 0 \end{aligned}$$

  Stručnejšie

$$\begin{aligned} &\mathit{Fib}(0) = 0 \\ &\mathit{Fib}(n+1) = \mathit{While\_fib}(n, 0, 1) \end{aligned}$$

7.3. Simulácia `for` cyklu

III.15 Simulácia `for` cyklu

Výpočet faktoriálu `for` cyklom:

```
function Fact(n: Integer): Integer;
var f: Integer;
begin
    f := 1;
    for i := 1 to n do
        f := f * i;
    Result := f
end;
```

Simulácia v CL

- Simulácia `for` cyklu:

  $For\_fact(i, n, f) = For\_fact(i + 1, n, f \cdot i) \leftarrow i \leq n$
  $For\_fact(i, n, f) = f \qquad\qquad\qquad\qquad \leftarrow i > n$

  Spätná rekurzia – rekurzívna premenná i rastie, kým nedosiahne n

- Inicializácia premenných a štart cyklu:

  $Fact(n) = For\_fact(1, n, 1)$

VI. prednáška
Párovacia funkcia CL, n-tice a zoznamy

22. marca 2011

VI.1 Rekapitulácia

Na predošlých dvoch prednáškach a cvičeniach sme videli:

- Dyadická reprezentácia čísel kóduje konečné postupnosti (reťazce) symbolov 1, 2
- Cantorova párovacia funkcia kóduje usporiadané dvojice čísel

8. Párovacia funkcia CL

VI.2 Párovacia funkcia CL

Párovacia funkcia zabudovaná v CL:

- Založená na očíslovaní binárnych stromov (Catalanove čísla)
- Zapisuje sa binárnym operátorom , (čiarka)
  - najnižšia priorita: $1+2, 3+4 \equiv ((1+2),(3+4))$
  - implicitne sa zátvorkuje doprava: $1,2,3,4 \equiv 1,(2,(3,4))$
- Základné párovacie vlastnosti:

$$(x,y) = (u,v) \to x = u \wedge y = v$$
$$x < (x,y) \wedge y < (x,y)$$
$$x = 0 \vee \exists y \exists z (x = (y,z))$$

- Výhodnejšie vlastnosti ako Cantorova párovacia funkcia (dĺžka párovacieho zápisu čísel rastie podobne ako dĺžka binárneho zápisu)

VI.3 Usporiadané n-tice

- Párovacia funkcia kóduje usporiadané dvojice čísel
- Usporiadané trojice kódujeme vnorením dvojíc doprava

$$x_1, (x_2, x_3) \equiv x_1, x_2, x_3$$

  Všimnite si implicitné zátvorkovanie doprava

- Usporiadané n-tice:

$$x_1, (x_2, (x_3, \ldots (x_{n-1}, x_n) \ldots)) \equiv x_1, x_2, x_3, \ldots, x_{n-1}, x_n$$

8.1. Programovanie s párovacou funkciou

VI.4 Programovanie s párovacou funkciou

- Pár vytvoríme výrazom „s, t“
- Párová diskriminácia:

$$\cdots \leftarrow t = 0$$
$$\cdots \leftarrow t = u, v$$

  Druhý prípad priradí zložky páru do nových premenných u a v

VI.5 Programovanie s párovacou funkciou

- Príklad: funkcia počítajúca n-tý a $(n+1)$-ý prvok Fibonacciho postupnosti: $\text{Twofib}(n) = (\text{fib}_n, \text{fib}_{n+1})$

$$\text{Twofib}(0) = 0, 1$$
$$\text{Twofib}(n+1) = 0 \qquad \leftarrow \text{Twofib}(n) = 0$$
$$\text{Twofib}(n+1) = b, a+b \leftarrow \text{Twofib}(n) = a, b$$

- Prípad v druhej klauzule nemôže nastať, môžeme ju vynechať

  CL si výsledok 0 doplní *default*om

$$\text{Twofib}(0) = 0, 1$$
$$\text{Twofib}(n+1) = b, a+b \leftarrow \text{Twofib}(n) = a, b$$

- Defaulty slúžia pre prípady, keď výpočet nemá zmysel

  Nevynechávajte klauzuly, keď funkcia vracia 0 ako riadny, očakávaný výsledok!

VI.6 Programovanie s párovacou funkciou

- Výpočet:

$$\begin{array}{lcl} \text{Twofib}(5) = & & 5, 8 \\ & \downarrow & \uparrow \\ \text{Twofib}(4) = & & 3, 5 \\ & \downarrow & \uparrow \\ \text{Twofib}(3) = & & 2, 3 \\ & \downarrow & \uparrow \\ \text{Twofib}(2) = & & 1, 2 \\ & \downarrow & \uparrow \\ \text{Twofib}(1) = & & 1, 1 \\ & \downarrow & \nearrow \\ \text{Twofib}(0) = & 0, 1 & \end{array}$$

- CL nepočíta číselnú hodnotu párovacej funkcie, pokiaľ to nie je nevyhnutné

  V pamäti vytvorí pascalovský `record`

8.2. Tupling

VI.7 Tupling

- Pomocou párovania môžeme súčasne počítať funkcie,
  - ktoré majú podobnú štruktúru (diskriminácie, rekurzívne argumenty)
  - a ich výsledky potrebujeme súčasne

  Technika sa nazýva *tupling*

- Napríklad celočíselné delenie a zvyšok:

$$\begin{array}{ll} x \div y = 0 & \leftarrow y = 0 \\ x \div y = 0 & \leftarrow y \neq 0 \wedge x < y \\ x \div y = ((x \dot{-} y) \div y) + 1 & \leftarrow y \neq 0 \wedge x \geq y \end{array}$$

$$\begin{array}{ll} x \bmod y = 0 & \leftarrow y = 0 \\ x \bmod y = x & \leftarrow y \neq 0 \wedge x < y \\ x \bmod y = (x \dot{-} y) \bmod y & \leftarrow y \neq 0 \wedge x \geq y \end{array}$$

  spojíme do jednej funkcie

$$\text{Divmod}(x, y) = x \div y, x \bmod y$$

VI.8 Tupling

- Spojená funkcia:

$$\begin{array}{ll} \text{Divmod}(x, y) = 0, 0 & \leftarrow y = 0 \\ \text{Divmod}(x, y) = 0, x & \leftarrow y \neq 0 \wedge x < y \\ \text{Divmod}(x, y) = q + 1, r & \leftarrow y \neq 0 \wedge x \geq y \wedge \\ & \quad \text{Divmod}(x - y, y) = q, r \end{array}$$

9. Zoznamy

VI.9 Kódovanie konečných postupností

- Často potrebujeme spracúvať postupnosti údajov

– Napríklad spočítať priemerný počet bodov z testu

- Ako zakódujeme postupnosti čísel *ľubovoľnej*, vopred neznámej dĺžky?
  – Napríklad *8, 2, 0, 4, 6* alebo *1, 3, 5, 7*

VI.10 Kódovanie konečných postupností

- Rovnako ako usporiadané n-tice?

$$x = 8, 2, 0, 4, 6$$
$$y = 1, 3, 5, 7$$

- Dekódovanie:

$$\begin{array}{lll} x = a_1, x_1 & \rightsquigarrow & a_1 = 8 \quad x_1 = 2, 0, 4, 6 \\ x_1 = a_2, x_2 & \rightsquigarrow & a_2 = 2 \quad x_2 = 0, 4, 6 \\ x_2 = a_3, x_3 & \rightsquigarrow & a_3 = 0 \quad x_3 = 4, 6 \\ x_3 = a_4, x_4 & \rightsquigarrow & a_4 = 4 \quad x_4 = 6 \\ x_4 = a_5, x_5 & \rightsquigarrow & a_5 = 1 \quad x_5 = 1 \end{array}$$

- *Nefunguje*, lebo každé kladné číslo kóduje dvojicu!

$$x = 8, 2, 0, 4, 6 = 8, 2, 0, 4, 1, 1 = 8, 2, 0, 4, 1, 0, 0$$

VI.11 Kódovanie konečných postupností

- Nie každé číslo kóduje dvojicu: 0
- Nie každé číslo kódujúce dvojicu kóduje trojicu: $x_1, 0$

⋮

- Nie každé číslo kódujúce n-ticu kóduje $(n+1)$-ticu: $x_1, x_2, \ldots, x_{n-1}, 0$

VI.12 Kódovanie konečných postupností

- n-prvkovú postupnosť čísel $x_1, x_2, \ldots, x_n$ zakódujeme párovaním ako usporiadanú $(n+1)$-ticu, ktorej posledná zložka je 0:

$$x_1, x_2, \ldots, x_n, 0$$

  Zaručí jednoznačné dekódovanie

$$xs = 8, 2, 0, 4, 6, 0$$
$$ys = 1, 3, 5, 7, 0$$

- Takémuto kódu hovoríme ***zoznam***
- 0 je *prázdny zoznam*, kóduje prázdnu postupnosť
- Konvencia:

  Premenenné obsahujúce zoznamy majú príponu -*s*

$$xs \qquad ys \qquad zs \qquad \ldots$$

  (anglické množné číslo, čítame „ixy“, „ypsilony“, „zety“, . . .)

9.1. Programovanie so zoznamami

VI.13 Programovanie so zoznamami

Naprogramujme funkciu Conc(xs, ys), ktorá zreťazí zoznamy xs a ys

▶ Napríklad $\text{Conc}\big((1, 2, 3, 0), (4, 5, 6, 0)\big) = 1, 2, 3, 4, 5, 6, 0$

Ako hľadať riešenie?

1. Uvedomíme si, že vstup je zoznam:
   - je buď *prázdny*: $xs = 0$
   - alebo má *prvý prvok a zvyšok*: $xs = x, zs$

Párová diskriminácia

2. Určíme výsledok pre prázdny zoznam
   - $\mathrm{Conc}(\underbrace{0}_{xs}, \underbrace{(4,5,6,0)}_{ys}) = 4,5,6,0$
3. *Výsledok pre zvyšok* zoznamu
   - $\mathrm{Conc}(\underbrace{(2,3,0)}_{zs}, \underbrace{(4,5,6,0)}_{ys}) = 2,3,4,5,6,0$

   upravíme na výsledok pre *celý zoznam*
   - $\mathrm{Conc}(\underbrace{(\underbrace{1}_{x}, \underbrace{2,3,0}_{zs})}_{}, \underbrace{(4,5,6,0)}_{ys}) = \underbrace{1}_{x}, \underbrace{2,3,4,5,6,0}_{\mathrm{Conc}(zs,ys)}$

VI.14 Programovanie so zoznamami

- Výsledná funkcia

$$\begin{aligned} \mathrm{Conc}(xs, ys) &= ys && \leftarrow xs = 0 \\ \mathrm{Conc}(xs, ys) &= x, \mathrm{Conc}(zs, ys) && \leftarrow xs = x, zs \end{aligned}$$

- Párovú diskrimináciu môžeme dosadiť do argumentov a premenovať premennú *zs* na *xs*:

$$\begin{aligned} &\mathrm{Conc}(0, ys) = ys \\ &\mathrm{Conc}((x, xs), ys) = x, \mathrm{Conc}(xs, ys) \end{aligned}$$

- Výpočet:

$$\begin{aligned} &\mathrm{Conc}\big((1,2,3,0),(4,5,6,0)\big) \\ &\quad = 1, \mathrm{Conc}\big((2,3,0),(4,5,6,0)\big) \\ &\quad = 1,2, \mathrm{Conc}\big((3,0),(4,5,6,0)\big) \\ &\quad = 1,2,3, \mathrm{Conc}\big(0,(4,5,6,0)\big) \\ &\quad = 1,2,3,4,5,6,0 \end{aligned}$$

VI.15 Zabudované zreťazenie

- Zreťazenie je zabudované do CL
- Operátor $xs +\!+\, ys \equiv xs \oplus ys$

$$xs \oplus ys = \mathrm{Conc}(xs, ys)$$

- Vyššia priorita ako párovanie:

$$\begin{aligned} &xs \oplus a, ys \equiv (xs \oplus a), ys && (\dagger\dagger\dagger) \\ &xs \oplus (a, ys) && (\mathrm{OK}) \end{aligned}$$

Chyba v (†††): zreťazujeme zoznam *xs* s prvkom *a*

Zreťazovať však možno *iba zoznamy*!

VII. prednáška
Tupling a simulácia cyklov na zoznamoch. Predikáty na zoznamoch

29. marca 2011

10. Tupling zoznamových operácií

10.1. Split = Take,Drop

VII.1 Tupling zoznamových operácií – Split

- Operácie

$$\begin{aligned}\mathsf{Take}(\quad 0,\quad xs) &= 0\\ \mathsf{Take}(i+1,\quad 0) &= 0\\ \mathsf{Take}(i+1,\ x, xs) &= x, \mathsf{Take}(i, xs)\end{aligned}$$

$$\begin{aligned}\mathsf{Drop}(\quad 0,\quad xs) &= xs\\ \mathsf{Drop}(i+1,\quad 0) &= 0\\ \mathsf{Drop}(i+1,\ x, xs) &= \mathsf{Drop}(i, xs)\end{aligned}$$

  majú rovnakú štruktúru diskriminácií a rekurzie

- Ak potrebujeme výsledky oboch $\rightarrow$ dva prechody zoznamom
- Chceme spojenú jednoprechodovú funkciu – *tupling*$^{\uparrow 8.2}$

$$\mathsf{Split}(i, xs) = \mathsf{Take}(i, xs), \mathsf{Drop}(i, xs)$$

- Príklad s náčrtom rekurzie:

$$\begin{aligned}\mathsf{Split}(2,\quad 3,5,7,11,13,0) &= \quad (3,5,0),(7,11,13,0)\\ \mathsf{Split}(3,\ 2,3,5,7,11,13,0) &= (2,3,5,0),(7,11,13,0)\end{aligned}$$

VII.2 Tupling zoznamových operácií – Split

$$\begin{aligned}\mathsf{Split}(\quad 0,\quad xs) &= \quad 0, xs\\ \mathsf{Split}(i+1,\quad 0) &= \quad 0,\ 0\\ \mathsf{Split}(i+1,\ x, xs) &= (x, ts), ds \leftarrow \mathsf{Split}(i, xs) = ts, ds\end{aligned}$$

10.2. Zip, Unzip

VII.3 Tupling zoznamových operácií – Unzip

- Zip – spojenie dvoch zoznamov rovnakej dĺžky do *zoznamu dvojíc*

$$\mathsf{Zip}((0,1,2,0),(7,6,5,0)) = (0,7),(1,6),(2,5),0$$

- Unzip – rozdelenie zoznamu dvojíc na dva zoznamy

$$\mathsf{L}(xs) = \mathsf{L}(ys) \rightarrow \mathsf{Unzip}\,\mathsf{Zip}(xs, ys) = xs, ys$$

- V podstate tuplingová úloha: spájame funkcie

$$\begin{aligned}\mathsf{Firsts}((0,7),(1,6),(2,5),0) &= 0,1,2,0\\ \mathsf{Seconds}((0,7),(1,6),(2,5),0) &= 7,6,5,0\end{aligned}$$

$$\mathsf{Unzip}(xys) = \mathsf{Firsts}(xys), \mathsf{Seconds}(xys)$$

- Príklad s náčrtom rekurzie:

$$\begin{aligned}\mathsf{Unzip}(\quad (1,6),(2,5),0) &= \quad (1,2,0),\quad (6,5,0)\\ \mathsf{Unzip}((0,7),(1,6),(2,5),0) &= (0,1,2,0),(7,6,5,0)\end{aligned}$$

VII.4 Tupling zoznamových operácií – Unzip

$$\begin{aligned}\mathsf{Unzip}(0) \quad &= \quad 0,\quad 0\\ \mathsf{Unzip}((x,y),xys) &= (x,xs),(y,ys) \leftarrow \mathsf{Unzip}(xys) = xs, ys\end{aligned}$$

11. Simulácia cyklov na zoznamoch

11.1. Súčet prvkov zoznamu

VII.5 Cykly na zoznamoch v Pascale

Súčet prvkov zoznamu v Pascale:

```
function Suml(xs: PVrchol): Integer;
var s, y: Integer; ys: PVrchol;
begin
    s := 0;
    while xs <> nil do begin
        y  := xs^.Info;
        ys := xs^.Next;
        s  := s + y;
        xs := ys
    end;
    Result := s
end;
```

VII.6 Simulácia cyklov na zoznamoch v CL

- Simulácia `while` cyklu pre súčet prvkov:

$$\begin{aligned} \mathrm{Suml\_while}(xs, s) &= s && \leftarrow xs = 0 \\ \mathrm{Suml\_while}(xs, s) &= \mathrm{Suml\_while}(ys, s + y) && \leftarrow xs = y, ys \end{aligned}$$

- Inicializácia:

$$\mathrm{Suml}(xs) = \mathrm{Suml\_while}(xs, 0)$$

- S dosadením do argumentov:

$$\begin{aligned} \mathrm{Suml\_while}(\quad 0, s) &= s \\ \mathrm{Suml\_while}((x, xs), s) &= \mathrm{Suml\_while}(xs, s + x) \end{aligned}$$

$$\mathrm{Suml}(xs) = \mathrm{Suml\_while}(xs, 0)$$

11.2. Obrátenie zoznamu

VII.7 Neefektivita obrátenia zoznamu zreťazovaním

- Na minulej prednáške/cvičení:

$$\begin{aligned} \mathrm{Rev}(0) &= 0 \\ \mathrm{Rev}(x, xs) &= \mathrm{Rev}(xs) \oplus (x, 0) \end{aligned}$$

- Na $xs \oplus ys$ treba $\mathrm{L}(xs)$ krokov

- Výpočet Rev:

$$\begin{array}{rcl} \mathrm{Rev}(1,2,3,4,0) & = \mathrm{Rev}(2,3,4,0) \oplus (1,0) = (4,3,2,0) \oplus (1,0) & 3 \\ & \swarrow \qquad\qquad\qquad\qquad\quad \uparrow & \\ \mathrm{Rev}(2,3,4,0) & = \mathrm{Rev}(3,4,0) \oplus (2,0) = (4,3,0) \oplus (2,0) & 2 \\ & \swarrow \qquad\qquad\qquad\qquad\quad \uparrow & \\ \mathrm{Rev}(3,4,0) & = \mathrm{Rev}(4,0) \oplus (3,0) = (4,0) \oplus (3,0) & 1 \\ & \swarrow \qquad\qquad\qquad\qquad\quad \uparrow & \\ \mathrm{Rev}(4,0) & = \mathrm{Rev}(0) \oplus (4,0) = (0) \oplus (4,0) & 0 \\ & \searrow \qquad\qquad\qquad\qquad\quad \uparrow & \\ & \mathrm{Rev}(0) = 0 & \\ \hline & & 3{+}2{+}1{+}0 = 6 \end{array}$$

- Vo všeobecnosti $(n^2 + n)/2$ krokov, kde $n = \mathrm{L}(xs) \dot{-} 1$

VII.8 Efektívne obrátenie zoznamu

- Zoznam xs možno obrátiť na $\mathrm{L}(xs)$ krokov:

xs	rs
1,2, 3, 4, 0	0
2,3, 4, 0	1,0
3, 4, 0	2,1, 0
4, 0	3, 2, 1, 0
0	4, 3, 2, 1, 0

- Vlastne `while` cyklus:

```
rs := 0;
while xs <> 0 do begin
    (y,ys) := xs;
    rs := y,rs;
    xs := ys
end;
Result := rs
```

VII.9 Efektívne obrátenie zoznamu v CL

- Efektívne obrátenie zoznamu simuláciou `while` cyklu:

  $\text{Rev\_while}(\quad 0, rs) = rs$
  $\text{Rev\_while}((x, xs), rs) = \text{Rev\_while}(xs, (x, rs))$

  $\text{Rev}(xs) = \text{Rev\_while}(xs, 0)$

12. Predikáty na zoznamoch

12.1. Predikáty

VII.10 Predikáty

- *Predikát* je vlastnosť objektu alebo n-tice objektov
- Predikáty na zoznamoch – vlastnosti zoznamov
  - xs je prázdny zoznam

    $$\text{Empty}(xs) \leftrightarrow xs = 0$$

  - zoznam xs je palindróm

    $$\text{Palindrome}(xs) \leftrightarrow xs = \text{Rev}(xs)$$

  - a je prvkom zoznamu xs

    $$a \; \varepsilon \; xs \equiv \text{In}(a, xs) \leftrightarrow \exists ys \exists zs (xs = ys \oplus (a, zs))$$

  - ys je prefixom (začiatočným úsekom) zoznamu xs

    $$\text{Prefix}(ys, xs) \leftrightarrow \exists zs (xs = ys \oplus zs)$$

  ...

VII.11 Charakteristické funkcie

- Namiesto predikátu P môžeme naprogramovať jeho *charakteristickú funkciu* P_*:

  $$P_*(x) = \begin{cases} 1 & \text{ak } P(x) \text{ platí,} \\ 0 & \text{ak } P(x) \text{ neplatí} \end{cases}$$

- Napríklad

  $\text{Empty}_*(xs) = 1 \leftarrow xs = 0$
  $\text{Empty}_*(xs) = 0 \leftarrow xs \neq 0$

  $\text{Palindrome}_*(xs) = 1 \leftarrow xs = Rev(xs)$
  $\text{Palindrome}_*(xs) = 0 \leftarrow xs \neq Rev(xs)$

  $\text{In}_*(a, 0) = 0$
  $\text{In}_*(a, (x, xs)) = 1 \leftarrow a = x$
  $\text{In}_*(a, (x, xs)) = 1 \leftarrow a \neq x \wedge \text{In}_*(a, xs) = 1$
  $\text{In}_*(a, (x, xs)) = 0 \leftarrow a \neq x \wedge \text{In}_*(a, xs) = 0$

VII.12 Definovanie predikátov

- V CL môžeme predikáty definovať priamo
  - podobne ako funkcie
  - *vynechávame* klauzuly pre prípady, že predikát *neplatí*
- Napríklad

$$\begin{aligned}
&\mathsf{Empty}(xs) \leftarrow xs = 0\\
&\cancel{\neg\mathsf{Empty}(xs) \leftarrow xs \neq 0}\\
&\\
&\mathsf{Palindrome}(xs) \leftarrow xs = Rev(xs)\\
&\cancel{\neg\mathsf{Palindrome}(xs) \leftarrow xs \neq Rev(xs)}\\
&\\
&\cancel{\neg\mathsf{In}(a, 0)}\\
&\mathsf{In}(a, (x, xs)) \leftarrow a = x\\
&\mathsf{In}(a, (x, xs)) \leftarrow a \neq x \wedge \mathsf{In}(a, xs)\\
&\cancel{\neg\mathsf{In}(a, (x, xs)) \leftarrow a \neq x \wedge \neg\mathsf{In}(a, xs)}
\end{aligned}$$

- Predikát $\mathsf{In}(a, xs)$ je zabudovaný

  Infixová forma: $a \in xs$, zapisujeme `a in xs`

  Negácia: $a \notin xs$, zapisujeme `a !in xs`

VII.13 Diskriminácia na platnosť predikátu

- Diskriminácia na platnosť predikátu

$$\begin{aligned}
\cdots &\leftarrow P(x)\\
\cdots &\leftarrow \neg P(x)
\end{aligned}$$

- $\neg P(x)$ zapisujeme `~P(x)`

12.2. Prefix

VII.14 Prefix

- Prefix – začiatočný úsek zoznamu

$$\mathsf{Prefix}(ys, xs) \leftrightarrow \exists zs(xs = ys \oplus zs)$$

- Napríklad

$$\begin{aligned}
&\mathsf{Prefix}(&0,& (2,3,5,7,11,0))\\
&\mathsf{Prefix}(&(2,3,0),& (2,3,5,7,11,0))\\
\neg&\mathsf{Prefix}(&(2,3,0),& 0)\\
\neg&\mathsf{Prefix}((2,3,5,0),& &(2,3,0))\\
\neg&\mathsf{Prefix}((1,2,3,0),& (2,3,5,7,11,0))&
\end{aligned}$$

- Všimnite si:

$$\begin{aligned}
ys &= 2,3,5, \mid 0\\
xs &= \underbrace{2,3,5,}_{\text{prvky } ys} \mid \underbrace{7,11,0}_{\text{čokoľvek}}
\end{aligned}$$

- Ako zadefinujeme rekurzívne?

VII.15 Prefix

- Riešenie

$$\begin{aligned}
&\mathsf{Prefix}(ys, xs) \leftarrow ys = 0\\
&\cancel{\neg\mathsf{Prefix}(ys, xs) \leftarrow ys = y, zs \wedge xs = 0}\\
&\mathsf{Prefix}(ys, xs) \leftarrow ys = y, zs \wedge xs = x, us \wedge x = y \wedge\\
&\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{Prefix}(zs, us)\\
&\cancel{\neg\mathsf{Prefix}(ys, xs) \leftarrow ys = y, zs \wedge xs = x, us \wedge x = y \wedge}\\
&\qquad\qquad\qquad\qquad\qquad\qquad \cancel{\neg\mathsf{Prefix}(zs, us)}\\
&\cancel{\neg\mathsf{Prefix}(ys, xs) \leftarrow ys = y, zs \wedge xs = x, us \wedge x \neq y}
\end{aligned}$$

- S dosadením do argumentov:

$$\begin{aligned}
&\mathsf{Prefix}(0, xs)\\
&\mathsf{Prefix}((y, ys), (x, xs)) \leftarrow x = y \wedge \mathsf{Prefix}(ys, xs)
\end{aligned}$$

12.3. Suffix

VII.16 Suffix

- Suffix – koncový úsek zoznamu

$$\text{Suffix}(zs, xs) \leftrightarrow \exists ys(xs = ys \oplus zs)$$

- Napríklad

$$\begin{aligned} &\text{Suffix}((7,11,0), &(7,11,0)) &\quad (1)\\ &\text{Suffix}((7,11,0), &(2,3,5,7,11,0)) &\quad (2)\\ \neg&\text{Suffix}((7,11,0), &0)\\ \neg&\text{Suffix}((7,11,0), &(5,7,11,13,0)) \end{aligned}$$

- Všimnite si:

$$\begin{array}{lcl} zs & = & \phantom{\underbrace{2,3,5,}_{\text{čokoľvek}}} \mid 7,11,0 \\ xs & = & \underbrace{2,3,5,}_{\text{čokoľvek}} \mid \underbrace{7,11,0}_{zs} \end{array}$$

- Ako zadefinujeme rekurzívne?
- Návod: testujeme rovnosť/nerovnosť zs a xs (1), rekurzívne prechádzame iba xs (2)

12.4. Substring

VII.17 Substring

- Substring – súvislý úsek zoznamu

$$\text{Substring}(us, xs) \leftrightarrow \exists ys \exists zs(xs = ys \oplus us \oplus zs)$$

- Napríklad

$$\begin{aligned} &\text{Substring}(\ (5,7,0), &(5,7,11,13,0))\\ &\text{Substring}(\ (5,7,0), &(2,3,5,7,11,13,0))\\ \neg&\text{Substring}((5,11,0), &(2,3,5,7,11,13,0)) \end{aligned}$$

- Všimnite si:

$$\begin{array}{lcl} us & = & \phantom{\underbrace{2,3}_{\text{čokoľvek}}} \mid 5,7, \mid 0 \\ xs & = & \underbrace{2,3}_{\text{čokoľvek}} \mid \underbrace{5,7,}_{\text{prvky } us} \mid \underbrace{11,13,0}_{\text{čokoľvek}} \end{array}$$

- Ako zadefinujeme rekurzívne?
- Ktorý z predchádzajúcich predikátov nám pomôže?

VIII. prednáška
Triedenie zoznamov
Zoznamová reprezentácia množín

5. apríla 2011

13. Triedenie zoznamov

13.1. Špecifikácia triedenia

VIII.1 Špecifikácia triedenia

Funkcia Sort(xs) = ys *utriedi* zoznam xs, ak súčasne platí:

1. zoznam ys obsahuje všetky prvky zoznamu xs vrátane prípadných opakovaní
   - ys je *permutáciou* xs
   - formálne: predikát $\mathrm{Perm}(xs, ys) \equiv (xs \sim ys)$
2. zoznam ys je usporiadaný od najmenšieho prvku po najväčší
   - formálne: predikát $\mathrm{Ord}(ys)$

Formalizácia špecifikácie:

$$\mathrm{Sort}(xs) \sim xs$$
$$\mathrm{Ord}(\mathrm{Sort}(xs))$$

VIII.2 Permutácia zoznamu

Ako zistíme, či je zoznam xs permutáciou zoznamu ys?

- Každý prvok sa v xs nachádza rovnaký počtekrát ako v ys
- Potrebujeme počet výskytov prvku a v zozname xs
  - Funkcia $\mathrm{Count}(a, xs) \equiv \#_a(xs)$
- Potom máme

$$xs \sim ys \leftrightarrow \forall a(\#_a(xs) = \#_a(ys))$$

VIII.3 Permutácia zoznamu – rekurzívny predikát

Ako vypočítame $xs \sim ys$ rekurziou?

1. Rekurzívny výpočet $\#_a(xs)$

   $\#_a(0) = 0$
   $\#_a(x, xs) = \#_a(xs) + 1 \leftarrow a = x$
   $\#_a(x, xs) = \#_a(xs) \quad \leftarrow a \neq x$

2. Pomocný predikát Eq_counts:

   $$\mathrm{Eq\_counts}(zs, xs, ys) \leftrightarrow \forall a(a \,\varepsilon\, zs \rightarrow \#_a(xs) = \#_a(ys))$$

   Naprogramujeme zoznamovou rekurziou na zs

3. Na otestovanie $xs \sim ys$ stačí porovnať počty výskytov prvkov, ktoré sa nachádzajú v xs alebo v ys

   $xs \sim ys \leftarrow \mathrm{Eq\_counts}(xs \oplus ys, xs, ys)$

VIII.4 Usporiadaný zoznam

Ako zistíme, či je zoznam xs usporiadaný?

- Každý prvok a v zozname xs je menší alebo rovnaký ako každý nasledujúci prvok b v zozname xs
- Vyjadrenie indexovaním:

$$\mathrm{Ord}(xs) \leftrightarrow \forall i \forall j(i < j \wedge j < \mathrm{L}(xs) \rightarrow xs[i] \leq xs[j])$$

- Vyjadrenie zreťazením:

$$\begin{aligned}&\mathrm{Ord}(xs) \leftrightarrow\\ &\forall a \forall b \forall ws \forall ys \forall zs (xs = ws \oplus (a, ys) \oplus (b, zs) \rightarrow a \leq b)\end{aligned}$$

- Stačí porovnávať susediace prvky:

$$\mathrm{Ord}(xs) \leftrightarrow \forall a \forall b \forall ys \forall zs (xs = ys \oplus (a, b, zs) \rightarrow a \leq b)$$

VIII.5 Usporiadaný zoznam – rekurzívny predikát

Ako vypočítame $\mathrm{Ord}(xs)$ rekurziou bez indexovania?

- Prázdny zoznam a jednoprvkové zoznamy sú usporiadané
- V dvoj- a viacprvkovom zozname porovnávame susedov
  - ▶ Pozor! Musíme porovnať *všetky dvojice* susedov

$$\begin{aligned}&\mathrm{Ord}(0)\\ &\mathrm{Ord}(a, 0)\\ &\mathrm{Ord}(a, b, xs) \leftarrow a \leq b \wedge \mathrm{Ord}(b, xs)\end{aligned}$$

VIII.6 Špecifikácia triedenia – rekapitulácia

Funkcia $\mathrm{Sort}(xs)$ utriedi zoznam xs, ak spĺňa

$$\begin{aligned}\mathrm{Sort}(xs) &\sim xs\\ \mathrm{Ord}(\mathrm{Sort}(xs)),&\end{aligned}$$

pričom

$$\begin{aligned}&xs \sim ys \leftrightarrow \forall a (\#_a(xs) = \#_a(ys))\\ &\mathrm{Ord}(us) \leftrightarrow\\ &\quad \forall a \forall b \forall xs \forall ys \forall zs (us = xs \oplus (a, ys) \oplus (b, zs) \rightarrow a \leq b)\end{aligned}$$

VIII.7 Algoritmy triedenia zoznamov

- Všetky algoritmy triedenia zoznamov spĺňajú rovnakú špecifikáciu
- Líšia sa efektivitou – dĺžkou výpočtu triedenia
- Algoritmy na triedenie polí sa dajú prispôsobiť na zoznamy
- Opakované indexovanie je neefektívne, používame postupné prechody zoznamami

13.2. Triedenie vsúvaním

VIII.8 Triedenie vsúvaním – insertion sort

- Jednoduchý, ale neefektívny algoritmus
- Postupne vsúvame prvky vstupného zoznamu do (rekurzívne) *utriedeného* zoznamu

$$\begin{aligned}&\mathrm{Isort}(0) = 0\\ &\mathrm{Isort}(x, xs) = \mathrm{Ins}(x, \mathrm{Isort}(xs))\end{aligned}$$

- Vsunutie prvku do utriedeného zoznamu

$$\begin{aligned}\mathrm{Ins}(a, xs) &\sim a, xs\\ \mathrm{Ord}(xs) &\rightarrow \mathrm{Ord}\,\mathrm{Ins}(a, xs)\end{aligned}$$

Zoznamovou rekurziou

$$\begin{aligned}&\mathrm{Ins}(a, 0) = a, 0\\ &\mathrm{Ins}(a, (x, xs)) = a, x, xs \qquad \leftarrow a \leq x\\ &\mathrm{Ins}(a, (x, xs)) = x, \mathrm{Ins}(a, xs) \leftarrow a > x\end{aligned}$$

VIII.9 Triedenie vsúvaním – výpočet

$$\begin{aligned}
&\mathsf{Isort}(3,7,2,0,5,0)\\
&\quad = \mathsf{Ins}(3,\mathsf{Isort}(7,2,0,5,0))\\
&\quad = \mathsf{Ins}(3,\mathsf{Ins}(7,\mathsf{Isort}(2,0,5,0)))\\
&\quad \ldots\\
&\quad = \mathsf{Ins}(3,\mathsf{Ins}(7,\mathsf{Ins}(2,\mathsf{Ins}(0,\mathsf{Ins}(5,\mathsf{Isort}(0))))))\\
&\quad = \mathsf{Ins}(3,\mathsf{Ins}(7,\mathsf{Ins}(2,\mathsf{Ins}(0,\mathsf{Ins}(5,0)))))\\
&\quad = \mathsf{Ins}(3,\mathsf{Ins}(7,\mathsf{Ins}(2,\mathsf{Ins}(0,(5,0)))))\\
&\quad = \mathsf{Ins}(3,\mathsf{Ins}(7,\mathsf{Ins}(2,(0,5,0))))\\
&\quad = \mathsf{Ins}(3,\mathsf{Ins}(7,(0,2,5,0)))\\
&\quad = \mathsf{Ins}(3,(0,2,5,7,0))\\
&\quad = 0,2,3,5,7,0
\end{aligned}$$

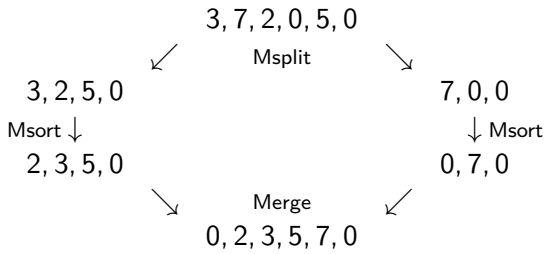
Dĺžka výpočtu $\mathsf{Ins}(a,xs)$: priemerne $\mathsf{L}(xs)/2$ krokov

Dĺžka výpočtu $\mathsf{Isort}(xs)$: priemerne $\sum_{i=0}^{\mathsf{L}(xs)} i/2 \approx (\mathsf{L}(xs))^2$

13.3. Triedenie zlučovaním

VIII.10 Triedenie zlučovaním – merge sort

- Efektívny algoritmus, metóda rozdeľuj a panuj
- Zoznam rozdelíme na dve približne rovnako dlhé časti
  - ▶ Funkcia Msplit
- Časti rekurzívne utriedime
- Utriedené časti efektívne zlúčime
  - ▶ Funkcia Merge



$$\begin{aligned}
&\mathsf{Msort}(0) = 0\\
&\mathsf{Msort}(a,0) = a,0\\
&\mathsf{Msort}(a,b,xs) = \mathsf{Merge}(\mathsf{Msort}(a,ys),\mathsf{Msort}(b,zs))\\
&\qquad\qquad \leftarrow \mathsf{Msplit}(xs) = ys,zs
\end{aligned}$$

VIII.11 Triedenie zlučovaním – špecifikácia

Rozdelenie zoznamu na približne rovnako dlhé časti

$$\begin{aligned}
&\exists ys \exists zs(\mathsf{Msplit}(xs) = ys,zs)\\
&\mathsf{Msplit}(xs) = ys,zs \rightarrow xs \sim ys \oplus zs\\
&\qquad\qquad \wedge (\mathsf{L}(ys) = \mathsf{L}(zs) \vee \mathsf{L}(ys) = \mathsf{L}(zs)+1)\\
&\qquad\qquad \wedge \forall i(xs[2{\cdot}i] = ys[i] \wedge xs[2{\cdot}i+1] = zs[i])
\end{aligned}$$

Zlúčenie dvoch utriedených zoznamov

$$\begin{aligned}
&\mathsf{Merge}(xs,ys) \sim xs \oplus ys\\
&\mathsf{Ord}(xs) \wedge \mathsf{Ord}(ys) \rightarrow \mathsf{Ord}\,\mathsf{Merge}(xs,ys)
\end{aligned}$$

VIII.12 Triedenie zlučovaním – implementácia

$$\begin{aligned}
&\mathsf{Msplit}(0) = 0, 0\\
&\mathsf{Msplit}(a,0) = (a,0), 0\\
&\mathsf{Msplit}(a,b,xs) = (a,ys),(b,zs) \leftarrow \mathsf{Msplit}(xs) = ys,zs
\end{aligned}$$

$$\begin{aligned}
&\mathsf{Merge}(0, ys) = ys\\
&\mathsf{Merge}((x,xs), 0) = x,xs\\
&\mathsf{Merge}((x,xs),(y,ys)) = x,\mathsf{Merge}(xs,(y,ys)) \leftarrow x \le y\\
&\mathsf{Merge}((x,xs),(y,ys)) = y,\mathsf{Merge}((x,xs),ys) \leftarrow x > y
\end{aligned}$$

14. Zoznamová reprezentácia konečných množín

VIII.13 Konečné množiny

- Konečná množina je kontainer – objekt, ktorý obsahuje iné objekty (prvky)
  - Záleží *iba* na príslušnosti prvku do množiny ($\in$)
  - Nezáleží na počte výskytov prvku
  - Nezáleží na poradí prvkov
- Implementácie rôznymi dátovými štruktúrami
  - Bitové polia, polia, zoznamy, rôzne druhy stromov, ...
  - Rôzne implementácie – rôzna zložitosť operácií

VIII.14 Množiny ako usporiadané zoznamy

- Jednoduchá implementácia množín: usporiadané zoznamy bez opakovania
- Vyjadrenie pomocou porovnania susediacich prvkov:

$$\mathrm{Set}(xs) \leftrightarrow \forall a \forall b \forall ys \forall zs (xs = ys \oplus (a, b, zs) \rightarrow a < b)$$

  Rekurzívna definícia – podobne ako Ord
- Na definovanie operácií budeme používať *trichotomickú* diskrimináciu

$$\cdots \leftarrow s < t$$
$$\cdots \leftarrow s = t$$
$$\cdots \leftarrow s > t$$

VIII.15 Príslušnosť a extenzionalita

- Príslušnosť do množiny

$$\mathrm{Set}(xs) \rightarrow a \in xs \leftrightarrow a \; \varepsilon \; xs$$

- Pre neprázdne množiny:

$$\mathrm{Set}(x, xs) \wedge a < x \rightarrow a \notin (x, xs)$$

- Využijeme pri rekurzívnej definícii predikátu $\in$

~~$a \notin 0$~~
~~$a \notin x, xs \leftarrow a < x$~~
$a \in x, xs \leftarrow a = x$
$a \in x, xs \leftarrow a > x \wedge a \in xs$
~~$a \notin x, xs \leftarrow a > x \wedge a \notin xs$~~

- Platí extenzionalita

$$\mathrm{Set}(xs) \wedge \mathrm{Set}(ys) \rightarrow xs = ys \leftrightarrow \forall a (a \in xs \leftrightarrow a \in ys)$$

VIII.16 Operácie na množinách

- Vlastnosti zoznamovej reprezentácie využijeme pri binárnych operáciách na množinách
- Napríklad zjednotenie $\mathrm{Union}(xs, ys) \equiv xs \cup ys$

$$\mathrm{Set}(xs) \wedge \mathrm{Set}(ys) \rightarrow \mathrm{Set}(xs \cup ys)$$
$$\mathrm{Set}(xs) \wedge \mathrm{Set}(ys) \rightarrow a \in xs \cup ys \leftrightarrow a \in xs \vee a \in ys$$

- Princíp implementácie je rovnaký ako pri Merge

$$\begin{aligned}
0 \cup ys &= ys \\
(x, xs) \cup 0 &= x, xs \\
(x, xs) \cup (y, ys) &= x, (xs \cup (y, ys)) \leftarrow x < y \\
(x, xs) \cup (y, ys) &= x, (xs \cup ys) \leftarrow x = y \\
(x, xs) \cup (y, ys) &= y, ((x, xs) \cup ys) \leftarrow x > y
\end{aligned}$$

- Ďalšie operácie ($\subseteq$, $\cap$, $\setminus$, $\triangle$) sa implementujú podobne

IX. prednáška
Binárne stromy

12. apríla 2011

15. Binárne stromy

15.1. Kódovanie stromov párovacími konštruktormi

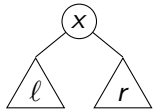
IX.1 Binárne stromy

Binárny strom – dátová štruktúra definovaná rekurzívne:

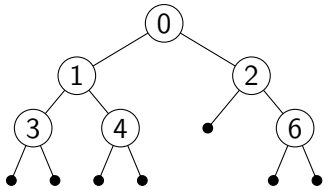
- Prázdny strom

  •

- Vrchol s hodnotou x a dvoma potomkami ℓ a r, ktoré sú binárnymi stromami



Napríklad



IX.2 Párovacie konštruktory

- Binárne stromy by sme v CL mohli zakódovať párovaním:
  - Prázdny strom: 0
  - Uzol s hodnotou x a potomkami ℓ a r: trojica (x, ℓ, r)
- Použijeme elegantnejší spôsob – *párovacie konštruktory*

$$C(x_1, x_2, \ldots, x_n) = \underline{k}, x_1, x_2, \ldots, x_n$$

  $\underline{k}$ je číselná konštanta – *tag*

  - Z hodnoty konštruktora sa dajú jednoznačne dekódovať jeho argumenty

$$C(x_1, \ldots, x_n) = C(y_1, \ldots, y_n) \to x_1 = y_1 \wedge \cdots \wedge x_n = y_n$$

  - Žiadna hodnota nemôže byť súčasne výsledkom dvoch konštruktorov s rôznymi tagmi

$$m \neq n \to (m, x) \neq (n, y)$$

  $\Rightarrow$ Konštruktory s rôznymi tagmi sú *diskriminovateľné*

IX.3 Kódovanie binárnych stromov

Binárny strom *zakódujeme* konštruktormi takto:

- Prázdny strom:

$$\mathsf{E} \equiv \bullet = 0, 0$$

- Uzol s hodnotou x a potomkami ℓ a r:

$$\mathsf{Nd}(x, \ell, r) \equiv \frac{x}{\ell \mid r} = 1, x, \ell, r$$

Binárny strom *dekódujeme* diskrimináciou

$$\cdots \leftarrow t = \mathsf{E} \qquad \cdots \leftarrow t = \bullet$$

$$\cdots \leftarrow t = \mathsf{Nd}(x, \ell, r) \qquad \cdots \leftarrow t = \frac{x}{\ell \mid r}$$

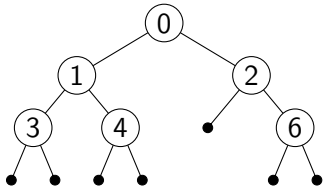
Po dosadení do argumentov:

$$f(\mathsf{E}) = \cdots \qquad f(\bullet) = \cdots$$

$$f\,\mathsf{Nd}(x, \ell, r) = \cdots \qquad f\left(\frac{x}{\ell \mid r}\right) = \cdots$$

IX.4 Kódovanie binárnych stromov – príklad

Napríklad strom



zakódujeme

$$\begin{array}{l}\mathsf{Nd}(0,\\ \quad \mathsf{Nd}(1,\\ \quad\quad \mathsf{Nd}(3,\mathsf{E},\mathsf{E}),\\ \quad\quad \mathsf{Nd}(4,\mathsf{E},\mathsf{E})),\\ \quad \mathsf{Nd}(2,\\ \quad\quad \mathsf{E},\\ \quad\quad \mathsf{Nd}(6,\mathsf{E},\mathsf{E})))\end{array} \equiv \frac{0}{\frac{1}{\frac{3}{\bullet\mid\bullet}\mid\frac{4}{\bullet\mid\bullet}}\Big|\frac{2}{\bullet\mid\frac{6}{\bullet\mid\bullet}}}$$

IX.5 Formát binárnych stromov

- Nie každé číslo kóduje binárny strom
- Číslo, ktoré kóduje binárny strom, sa dá zapísať pomocou konštruktorov E, Nd
- Predikát $\mathsf{Bt}(t)$ platí iba pre kódy binárnych stromov:

  $\mathsf{Bt}(\mathsf{E})$
  $\mathsf{Bt}\,\mathsf{Nd}(x,\ell,r) \leftarrow \mathsf{N}(x) \wedge \mathsf{Bt}(\ell) \wedge \mathsf{Bt}(r)$
  $\equiv$
  $\mathsf{Bt}(\bullet)$
  $\mathsf{Bt}\left(\frac{x}{\ell\mid r}\right) \leftarrow \mathsf{N}(x) \wedge \mathsf{Bt}(\ell) \wedge \mathsf{Bt}(r)$

- V CL takéto predikáty slúžia aj na formátovanie výsledkov query

15.2. Operácie na binárnych stromoch

IX.6 Rekurzia na binárnych stromoch

Operácie na binárnych stromoch definujeme *rekurziou na binárnych stromoch*

$$\begin{aligned} f(\mathsf{E}) &= \cdots \\ f\,\mathsf{Nd}(x,\ell,r) &= \cdots f(\ell) \cdots f(r) \cdots \end{aligned}$$

Špeciálny prípad course-of-values rekurzie, kódy potomkov sú menšie čísla ako kód ich rodičovského uzla

$$\ell < \mathsf{Nd}(x,\ell,r) \wedge r < \mathsf{Nd}(x,\ell,r)$$

IX.7 Operácie na binárnych stromoch

Veľkosť (počet uzlov) binárneho stromu $\mathsf{Sz}(t) \equiv |t|_b$:

$$\begin{array}{ll} \mathsf{Sz}(\mathsf{E}) = 0 & |\bullet|_b = 0 \\ \mathsf{Sz}\,\mathsf{Nd}(x,\ell,r) = 1 + \mathsf{Sz}(\ell) + \mathsf{Sz}(r) & \left|\frac{x}{\ell\mid r}\right|_b = 1 + |\ell|_b + |r|_b \end{array}$$

Predikát „x je prvkom t“ $\mathsf{Inbt}(a,t) \equiv a\ \varepsilon_b\ t$:

$$\begin{aligned} a\ \varepsilon_b\ \tfrac{x}{\ell\mid r} &\leftarrow a = x \\ a\ \varepsilon_b\ \tfrac{x}{\ell\mid r} &\leftarrow a \neq x \wedge a\ \varepsilon_b\ \ell \\ a\ \varepsilon_b\ \tfrac{x}{\ell\mid r} &\leftarrow a \neq x \wedge \neg(a\ \varepsilon_b\ \ell) \wedge a\ \varepsilon_b\ r \end{aligned}$$

IX.8 Prechody binárnymi stromami

Uzly binárneho stromu môžeme spracúvať v rôznom poradí

- Preorder: rodič, ľavý potomok, pravý potomok
- Inorder: ľavý potomok, rodič, pravý potomok
- Postorder: ľavý potomok, pravý potomok, rodič

Prechod stromu a uloženie hodnôt z vrcholov do zoznamu v poradí inorder:

$$\mathsf{Inorder}(\bullet) = 0$$
$$\mathsf{Inorder}\left(\frac{x}{\ell \mid r}\right) = \mathsf{Inorder}(\ell) \oplus (x, \mathsf{Inorder}(r))$$

IX.9 Prechod binárnym stromom – výpočet

$$\begin{aligned}
&\mathsf{Inorder}\left(\frac{0}{\frac{1}{\frac{3}{\bullet \mid \bullet} \mid \frac{4}{\bullet \mid \bullet}} \;\Big|\; \frac{2}{\bullet \mid \frac{6}{\bullet \mid \bullet}}}\right) \\
&= \mathsf{Inorder}\left(\frac{1}{\frac{3}{\bullet \mid \bullet} \mid \frac{4}{\bullet \mid \bullet}}\right) \oplus \left(0, \mathsf{Inorder}\left(\frac{2}{\bullet \mid \frac{6}{\bullet \mid \bullet}}\right)\right) \\
&= \left(\mathsf{Inorder}\left(\frac{3}{\bullet \mid \bullet}\right) \oplus \left(1, \mathsf{Inorder}\left(\frac{4}{\bullet \mid \bullet}\right)\right)\right) \\
&\quad \oplus \left(0, \mathsf{Inorder}(\bullet) \oplus \left(2, \mathsf{Inorder}\left(\frac{6}{\bullet \mid \bullet}\right)\right)\right) \\
&= \cdots \\
&= ((0 \oplus (3,0)) \oplus (1, 0 \oplus (4,0))) \oplus (0, 0 \oplus (2, 0 \oplus (6,0))) \\
&= ((3,0) \oplus (1,4,0)) \oplus (0,2,6,0) \\
&= (3,1,4,0) \oplus (0,2,6,0) \\
&= 3,1,4,0,2,6,0
\end{aligned}$$

IX.10 Efektívne prechody binárnymi stromami

- Výpočet Inorder je neefektívny – zreťazenie prechádza znova zoznam $\mathsf{Inorder}(\ell)$
- Zefektívnenie – odstránenie zreťazenia
- Trik: Pomocná premenná (akumulátor) *as* – zoznam, ktorý pripojíme *za* výpis (pod)stromu inorderom
- Požadovaná vlastnosť efektívnej (akumulátorovej) funkcie:

$$\mathsf{Inordera}(t, as) = \mathsf{Inorder}(t) \oplus as \qquad (*)$$

- Funkciu Inordera naprogramujeme bez zreťazenia
  Program odvodíme z požadovanej vlastnosti $(*)$

IX.11 Efektívne prechody binárnymi stromami

$$\begin{aligned}
&\mathsf{Inordera}(\bullet, as) \overset{(*)}{=} \mathsf{Inorder}(\bullet) \oplus as = 0 \oplus as = as \\
&\mathsf{Inordera}\left(\frac{x}{\ell \mid r}, as\right) \\
&\quad \overset{(*)}{=} \mathsf{Inorder}\left(\frac{x}{\ell \mid r}\right) \oplus as \\
&\quad = \big(\mathsf{Inorder}(\ell) \oplus (x, \mathsf{Inorder}(r))\big) \oplus as \\
&\quad = \mathsf{Inorder}(\ell) \oplus \big((x, \mathsf{Inorder}(r)) \oplus as\big) \\
&\quad = \mathsf{Inorder}(\ell) \oplus (x, \underline{\mathsf{Inorder}(r) \oplus as}) \\
&\quad \overset{(*)}{=} \underline{\mathsf{Inorder}(\ell) \oplus (x, \underline{\mathsf{Inordera}(r, as)})} \\
&\quad \overset{(*)}{=} \underline{\mathsf{Inordera}\big(\ell, (x, \mathsf{Inordera}(r, as))\big)}
\end{aligned}$$

15.3. Binárne vyhľadávacie stromy

IX.12 Binárne vyhľadávacie stromy

Binárny vyhľadávací strom

- Binárny strom
- V každom uzle s hodnotou x platí súčasne:
  - x je väčšie ako všetky hodnoty v ľavom potomkovi

– x je menšie ako všetky hodnoty v pravom potomkovi
– obaja potomkovia sú binárne vyhľadávacie stromy

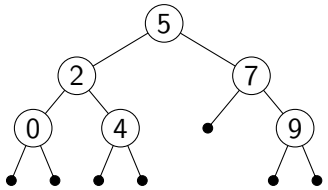
Vyjadrenie rekurzívnym predikátom:

$\mathrm{Bst}(\bullet)$
$\mathrm{Bst}\left(\frac{x}{\ell \mid r}\right) \leftarrow x \succ \ell \wedge x \prec r \wedge \mathrm{Bst}(\ell) \wedge \mathrm{Bst}(r)$

Cvičenie: rekurzívne definície predikátov

$$x \succ t \leftrightarrow \forall y (y \in_b t \rightarrow x > y)$$
$$x \prec t \leftrightarrow \forall y (y \in_b t \rightarrow x < y)$$

IX.13 Binárny vyhľadávací strom – príklad



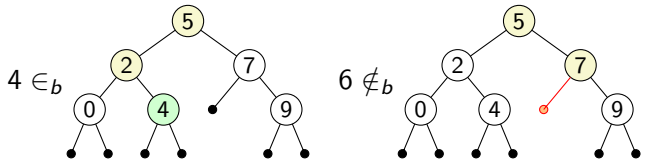
IX.14 Využitie vyhľadávacej vlastnosti

- Binárne vyhľadávacie stromy reprezentujú množiny
  - ▶ Podobne ako ostro usporiadané zoznamy minule
- Využitie vyhľadávacej vlastnosti pri predikáte „byť prvkom“:

$a \in_b \frac{x}{\ell \mid r} \leftarrow a = x$
$a \in_b \frac{x}{\ell \mid r} \leftarrow a < x \wedge a \in_b \ell$
$a \in_b \frac{x}{\ell \mid r} \leftarrow a > x \wedge a \in_b r$

Ak $a < x$, prehľadávame iba ℓ, a nemôže byť v r

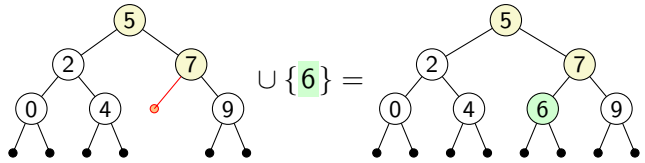
Prehľadávame iba jednu cestu stromom



IX.15 Vkladanie do vyhľadávacieho stromu

Princíp vkladania hodnoty do vyhľadávacieho stromu:

- Pokúsime sa nájsť vkladanú hodnotu
  a) Hodnota sa v strome nenachádza (na jej mieste je prázdny strom) ⇒ vyrobíme nový uzol

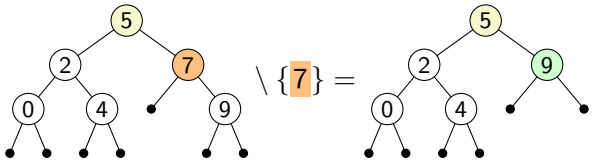


  b) Hodnota sa v strome nachádza ⇒ strom sa nemení
- Cestou naspäť strom zrekonštruujeme

IX.16 Zmazanie z vyhľadávacieho stromu I

Zmazanie hodnoty je komplikovanejšie ako vloženie

- Pokúsime sa nájsť zmazávanú hodnotu:
  a) Hodnotu nenájdeme ⇒ strom sa nezmení
  b) Hodnotu nájdeme a niektorý potomok je prázdny ⇒ nahradíme uzol druhým potomkom

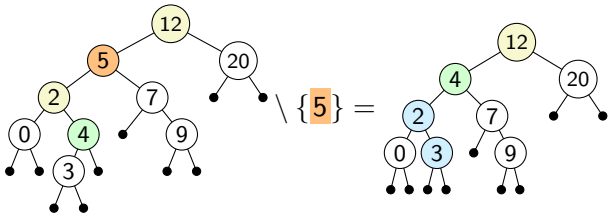


- Pokúsime sa nájsť zmazávanú hodnotu:

  ...

  c) Hodnotu nájdeme a obaja potomkovia sú neprázdni:

  1. Nájdeme náhradu za zmazávanú hodnotu:
     napríklad maximum ľavého podstromu
  2. Odstránime náhradu z pôvodného miesta
  3. Použijeme náhradu namiesto zmazávanej hodnoty



Kroky 1 a 2 – pomocná tuplingová funkcia

- Cestou naspäť strom zrekonštruujeme

XI. prednáška
Číslovanie binárnych stromov
Kombinatorika na zoznamoch

27. apríla 2011

15. Binárne stromy

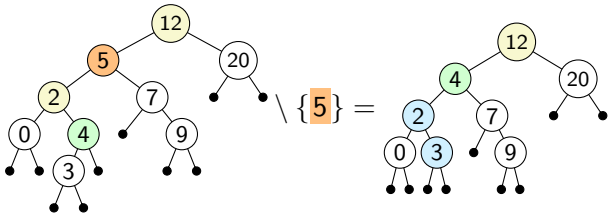
15.3. Binárne vyhľadávacie stromy (dokončenie)

XI.1 Zmazanie z vyhľadávacieho stromu

- Najkomplikovanejší prípad:

  Nájdeme zmazávanú hodnotu a obaja potomkovia sú neprázdni:

  1. Nájdeme náhradu za zmazávanú hodnotu:
     napríklad maximum ľavého podstromu
  2. Odstránime náhradu z pôvodného miesta
  3. Použijeme náhradu namiesto zmazávanej hodnoty



- Podrobnejšie o krokoch 1 a 2

XI.2 Maximum vyhľadávacieho stromu

- Nájdenie a zmazanie maxima vyhľadávacieho stromu:

$$\mathrm{Bst}(t) \wedge t \neq \bullet \to \mathrm{Maxt}(t)\ \varepsilon_b\ t$$
$$\mathrm{Bst}(t) \wedge t \neq \bullet \wedge x\ \varepsilon_b\ t \to x \leq \mathrm{Maxt}(t)$$

$$\mathrm{Bst}(t) \wedge t \neq \bullet \to \mathrm{Bst}\,\mathrm{Delmaxt}(t)$$
$$\mathrm{Bst}(t) \wedge t \neq \bullet \to x\ \varepsilon_b\ \mathrm{Delmaxt}(t) \leftrightarrow x\ \varepsilon_b\ t \wedge x \neq \mathrm{Maxt}(t)$$

- Programy hľadajú maximum v najpravejšom uzle:

$$\mathrm{Maxt}\left(\tfrac{x}{\ell \mid r}\right) = x \qquad \leftarrow r = \bullet$$
$$\mathrm{Maxt}\left(\tfrac{x}{\ell \mid r}\right) = \mathrm{Maxt}(r) \qquad \leftarrow r \neq \bullet$$

$$\mathrm{Delmaxt}\left(\tfrac{x}{\ell \mid r}\right) = \ell \qquad \leftarrow r = \bullet$$
$$\mathrm{Delmaxt}\left(\tfrac{x}{\ell \mid r}\right) = \tfrac{x}{\ell \mid \mathrm{Delmaxt}(r)} \leftarrow r \neq \bullet$$

- Rovnaké diskriminácie a argumenty rekurzie $\Rightarrow$ kandidáti na *tupling*

XI.3 Maximum vyhľadávacieho stromu – tupling

- Spojená funkcia – extrakcia maxima:

$$\mathrm{Bst}(t) \wedge t \neq \bullet \to \mathrm{Extract\_max}(t) = \mathrm{Maxt}(t), \mathrm{Delmaxt}(t)$$

$$\mathrm{Bst}(t) \wedge t \neq \bullet \to$$
$$\exists m \exists t_1 \big(\mathrm{Extract\_max}(t) = m, t_1 \wedge m\ \varepsilon_b\ t \wedge \mathrm{Bst}(t_1)$$
$$\wedge \forall x (x\ \varepsilon_b\ t \to x \leq m)$$
$$\wedge \forall x (x\ \varepsilon_b\ t_1 \leftrightarrow x\ \varepsilon_b\ t \wedge x \neq m)\big)$$

- Program:

$$\mathrm{Extract\_max}\left(\tfrac{x}{\ell \mid r}\right) = x, \ell \qquad \leftarrow r = \bullet$$
$$\mathrm{Extract\_max}\left(\tfrac{x}{\ell \mid r}\right) = m, \tfrac{x}{\ell \mid r_1} \leftarrow r \neq \bullet$$
$$\wedge\ \mathrm{Extract\_max}(r) = m, r_1$$

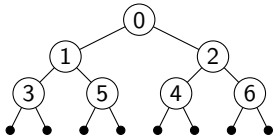
15.4. Číslovanie vrcholov v binárnych stromoch

XI.4 Úplné binárne stromy

- Hĺbka stromu – dĺžka najdlhšej cesty koreň–list:

  $\mathsf{d}(\bullet) = 0$
  $\mathsf{d}\left(\frac{x}{\ell \mid r}\right) = 1 + \max(\mathsf{d}(\ell), \mathsf{d}(r))$

- Úplný binárny strom:
  - na každej úrovni okrem poslednej: všetky uzly neprázdne
  - na poslednej úrovni sú iba prázdne stromy



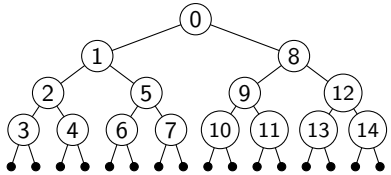
- Vzťah veľkosti a hĺbky úplného stromu:

$$|t|_b + 1 = 2^{\mathsf{d}(t)}$$

XI.5 Číslovanie vrcholov v úplných stromoch I

- Číslovanie uzlov úplného stromu číslami 0, 1, 2, ... v poradí preorder:

Full_pre(4) =



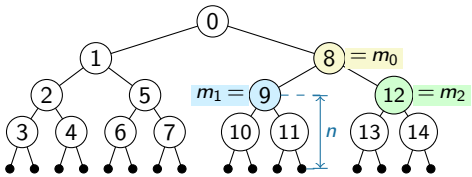
- Na nájdenie rekurzívneho riešenia musíme problém *zovšeobecniť*
- Zovšeobecnenie umožní číslovať podstromy

XI.6 Číslovanie vrcholov v úplných stromoch II

- Zovšeobecnenie – $\mathsf{Full\_pre}_1(n, m)$:

  Číslovanie uzlov úplného stromu hĺbky n číslami m, $m+1$, $m+2$, ... v poradí preorder

- Schéma riešenia:



  Číslovanie v poradí *preorder*:

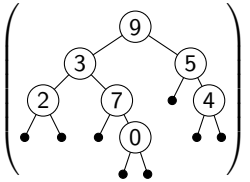
  1. očíslujeme koreň podstromu $m_0 = m$
  2. očíslujeme ľavý podstrom od $m_1 = m_0 + 1$
  3. očíslujeme pravý podstrom od $m_2 = m_1 + s$

- Aké je s pre úplný strom hĺbky n?
- Ako sa číslovanie zmení pre inorder, postorder?

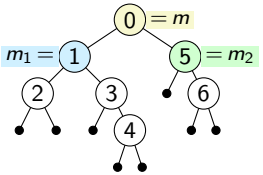
XI.7 Číslovanie vrcholov v ľubovoľných stromoch

- Číslovanie uzlov *ľubovoľného* stromu číslami 0, 1, 2, ... v poradí preorder:

Num_pre



=



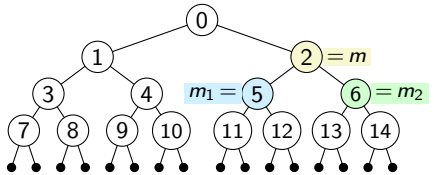
- Rovnaký postup ako pri úplných stromoch:

– Zovšeobecníme na číslovanie od m – Num_pre$_1(t, m)$
– Aký je rozdiel medzi číslom ľavého a pravého dieťaťa?
– Rozdiel vieme počítať aj bez pomocnej funkcie – tupling

XI.8 Číslovanie úplných stromov do šírky

- Číslovanie uzlov úplného stromu číslami 0, 1, 2, ... v poradí prechodu do šírky:

Full_br(4) =



- Problém zovšeobecníme pre podstrom s koreňom s číslom m
- Aký je vzťah medzi číslom uzla m a číslami jeho detí m_1 a m_2?

16. Kombinatorika na zoznamoch

16.1. Podpostupnosti

XI.9 Podpostupnosti

- Zoznam ys je *vybranou podpostupnosťou* zoznamu $xs = x_0, x_1, x_2, \ldots, x_n, 0$, ak

$$ys = x_{i_0}, x_{i_1}, \ldots, x_{i_k}, 0$$

pre nejakú rastúcu postupnosť $0 \leq i_0 < i_1 < i_2 < \cdots < i_k \leq n$

- Neformálne:
  – ys vznikne vynechaním niektorých prvkov zo zoznamu xs
  – vzájomné poradie zostávajúcich prvkov sa nezmení
- Napríklad:

$$xs = 3, 5, 2, 11, 17, 7, 13, 0$$
$$ys = \quad 5, 2, \qquad 7, \quad 0$$

- Predikát Subseq(ys, xs)

XI.10 Generovanie všetkých podpostupností

- Generovanie všetkých podpostupností zoznamu – funkcia Subseqs(xs)
- Rekurzívna úloha na zoznamoch:
  1. Základný prípad pre prázdny zoznam 0
  2. Rekurzívny prípad pre neprázdny zoznam x, xs

XI.11 Všetky podpostupnosti – základný prípad

1. Jedinou podpostupnosťou prázdneho zoznamu je prázdny zoznam

$$\text{Subseq}(ys, 0) \leftrightarrow ys = 0$$

Klauzula definície funkcie:

Subseqs$(0) = 0, 0$

▶ Jednoprvkový zoznam, ktorého jediným prvkom je prázdny zoznam

2. Podpostupnosti zoznamu x, xs:
   a) všetky x, zs, kde zs je vybranou podpostupnosťou xs
      ▶ x zahrnieme do podpostupnosti
   b) všetky vybrané podpostupnosti xs
      ▶ x vynecháme

   $\mathsf{Subseq}(ys, (x, xs)) \leftrightarrow$
   $\exists zs\big(ys = x, zs \wedge \mathsf{Subseq}(zs, xs)\big) \vee \mathsf{Subseq}(ys, xs)$

Príklad:

$$\begin{aligned} \mathsf{Subseqs}(\text{'ak'}) &= \quad \text{'ak'},\ \text{'a'},\ \text{'k'},\ \text{''}, 0 \\ \mathsf{Subseqs}(\text{'tak'}) &= (\text{'tak'}, \text{'ta'}, \text{'tk'}, \text{'t'}, 0) && \text{a)} \\ &\quad \oplus (\ \text{'ak'},\ \text{'a'},\ \text{'k'},\ \text{''}, 0) && \text{b)} \end{aligned}$$

Klauzula definície funkcie:

$$\begin{aligned} \mathsf{Subseqs}(x, xs) = {} & \mathsf{Map\_pair}\big(x, \mathsf{Subseqs}(xs)\big) \\ & \oplus \mathsf{Subseqs}(xs) \end{aligned}$$

16.2. Ďalšie úlohy

XI.13 Ďalšie kombinatorické úlohy

Premyslite si:

- k-prvkové podpostupnosti,
- variácie (ako permutácie, ale niektoré prvky možno vynechať)
- k-prvkové variácie,
- súvislé podreťazce[↑12.4]

XII. prednáška
Postfixový stroj a numerické výrazy

3. mája 2011

17. Postfixový stroj a numerické výrazy

17.1. Postfixový stroj

XII.1 Postfixový stroj

- Jednoduchý výpočtový model, virtuálny stroj
- Pamäť: zásobník čísel

5
3
7
2

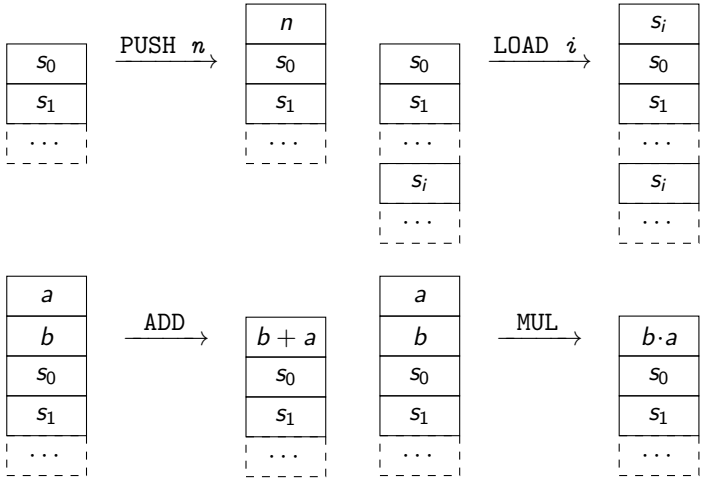
- Program: postupnosť inštrukcií

```
LOAD 2
PUSH 4
MUL
LOAD 2
LOAD 2
MUL
ADD
```

Inštrukcie modifikujú zásobník

Pridávajú/odoberajú čísla na vrch/z vrchu zásobníka

XII.2 Inštrukcie postfixového stroja

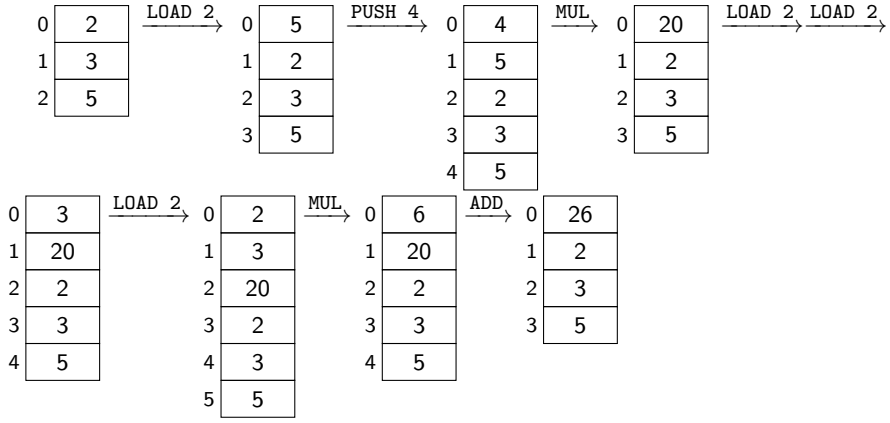


XII.3 Príklad výpočtu programu

Počítajme program

```
LOAD 2, PUSH 4, MUL, LOAD 2, LOAD 2, MUL, ADD
```

so zásobníkom, v ktorom sú čísla 2, 3, 5:

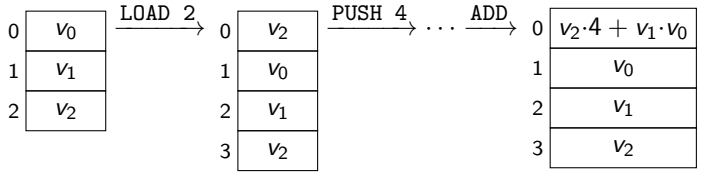


XII.4 Príklad výpočtu programu – všeobecnejšie

Počítajme program

```
LOAD 2, PUSH 4, MUL, LOAD 2, LOAD 2, MUL, ADD
```

so zásobníkom, v ktorom sú čísla v_0, v_1, v_2:



XII.5 Kódovanie postfixových strojov

Postfixové stroje ľahko *zakódujeme* v CL:

- Zásobník: zoznam čísel; vrch zásobníka: prvý prvok
- Inštrukcie: párové konštruktory (podobne ako E a Nd)

  $\texttt{PUSH}(n) \equiv \mathsf{Ci}(n) = 0, n$
  $\texttt{LOAD}(i) \equiv \mathsf{Vi}(i) = 1, i$
  $\texttt{ADD} \equiv \mathsf{Ai} = 2, 0$
  $\texttt{MUL} \equiv \mathsf{Mi} = 3, 0$

  Predikát $\mathsf{Instr}(x)$ platí, ak x kóduje inštrukciu:

  $\mathsf{Instr}\,\texttt{PUSH}(n) \leftarrow \mathsf{N}(n)$
  $\mathsf{Instr}\,\texttt{LOAD}(i) \leftarrow \mathsf{N}(i)$
  $\mathsf{Instr}(\texttt{ADD})$
  $\mathsf{Instr}(\texttt{MUL})$

- Program: zoznam inštrukcií

  $\mathsf{Program}(0)$
  $\mathsf{Program}(i, is) \leftarrow \mathsf{Instr}(i) \wedge \mathsf{Program}(is)$

XII.6 Simulácia postfixových strojov

Zakódovaný postfixový stroj môžeme *simulovať*:

- Funkcia $\mathsf{Run}(is, ss)$
- Simuluje beh postfixového stroja s programom is a zásobníkom ss
- Vráti vrch zásobníka na konci programu
- Klauzálna definícia:

$$\begin{aligned}
\mathsf{Run}(0 &, (s, ss)) &&= s \\
\mathsf{Run}((\texttt{PUSH}(n), is), ss &) &&= \mathsf{Run}(is, (n, ss)) \\
\mathsf{Run}((\texttt{LOAD}(i), is), ss &) &&= \mathsf{Run}(is, (ss[i], ss)) \\
\mathsf{Run}((\texttt{ADD}, is) &, (a, b, ss)) &&= \mathsf{Run}(is, (b + a, ss)) \\
\mathsf{Run}((\texttt{MUL}, is) &, (a, b, ss)) &&= \mathsf{Run}(is, (b \cdot a, ss))
\end{aligned}$$

17.2. Numerické výrazy s premennými

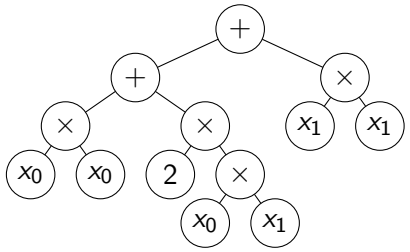
XII.7 Numerické výrazy s premennými

Numerický výraz s premennými je

- číselná konštanta $k \in \mathbb{N}$
  - ▶ 3, 0, 777, ...
- premenná x_i s indexom $i \in \mathbb{N}$
  - ▶ x_2, x_{17}, x_0, ...
- súčtový výraz $(t_1 + t_2)$, kde t_1 a t_2 sú numerické výrazy s premennými
  - ▶ $(x_1 + 5)$, $(x_5 + x_0)$, $(x_3 + (x_0 + x_2))$, ...
- súčinový výraz $(t_1 \times t_2)$, kde t_1 a t_2 sú numerické výrazy s premennými
  - ▶ $(2 \times x_0)$, $(x_4 \times x_1)$, $(((x_1 \times x_1) + (2 \times x_1)) + 1)$, ...

XII.8 Numerické výrazy ako stromy

- Numerické výrazy majú stromovú štruktúru
- Napríklad $((x_0 \times x_0) + (2 \times (x_0 \times x_1))) + (x_1 \times x_1)$



XII.9 Kódovanie numerických výrazov

Numerické výrazy môžeme v CL zakódovať podobne ako binárne stromy:

- Párové konštruktory (podobne ako E a Nd)

$$\begin{array}{lll} n^\bullet & \equiv \mathsf{Ct}(n) & = 0, n \\ x_i^\bullet & \equiv \mathsf{Vt}(i) & = 1, i \\ t_1 +^\bullet t_2 & \equiv \mathsf{At}(t_1, t_2) & = 2, t_1, t_2 \\ t_1 \times^\bullet t_2 & \equiv \mathsf{Mt}(t_1, t_2) & = 3, t_1, t_2 \end{array}$$

- Predikát $\mathsf{Term}(t)$ platí, ak t kóduje numerický výraz:

$$\begin{array}{ll} \mathsf{Term}(n^\bullet) & \leftarrow \mathsf{N}(n) \\ \mathsf{Term}(x_i^\bullet) & \leftarrow \mathsf{N}(i) \\ \mathsf{Term}(t_1 +^\bullet t_2) & \leftarrow \mathsf{Term}(t_1) \wedge \mathsf{Term}(t_2) \\ \mathsf{Term}(t_1 \times^\bullet t_2) & \leftarrow \mathsf{Term}(t_1) \wedge \mathsf{Term}(t_2) \end{array}$$

XII.10 Kódovanie numerických výrazov – príklad

- Výraz $((x_0 \times x_0) + (2 \times (x_0 \times x_1))) + (x_1 \times x_1)$ zakódujeme ako

$$\begin{array}{l} ((x_0^\bullet \times^\bullet x_0^\bullet) +^\bullet (2^\bullet \times^\bullet (x_0^\bullet \times^\bullet x_1^\bullet))) +^\bullet (x_1^\bullet \times^\bullet x_1^\bullet) \\ \equiv \mathsf{At}(\mathsf{At}(\mathsf{Mt}(\mathsf{Vt}(0), \mathsf{Vt}(0)), \\ \qquad \mathsf{Mt}(\mathsf{Ct}(2), \\ \qquad\quad \mathsf{Mt}(\mathsf{Vt}(0), \mathsf{Vt}(1)))), \\ \quad \mathsf{Mt}(\mathsf{Vt}(1), \mathsf{Vt}(1))) \end{array}$$

XII.11 Vyhodnocovanie numerických výrazov

Ohodnotenie premenných:

- zoznam $vs = v_0, v_1, v_2, \ldots$
- priraďuje premennej x_i hodnotu v_i

Hodnota výrazu pre dané ohodnotenie premenných $[\![t]\!]^{vs}$:

- Napríklad:

$$\begin{array}{l} t = ((x_0^\bullet \times^\bullet x_0^\bullet) +^\bullet (2^\bullet \times^\bullet (x_0^\bullet \times^\bullet x_1^\bullet))) +^\bullet (x_1^\bullet \times^\bullet x_1^\bullet) \\ [\![t]\!]^{3,5,7,0} = ((3{\cdot}3) + (2{\cdot}(3{\cdot}5))) + (5{\cdot}5) \end{array}$$

- Všeobecne:

$$\begin{array}{ll} [\![n^\bullet]\!]^{vs} & = n \\ [\![x_i^\bullet]\!]^{vs} & = vs[i] \\ [\![t_1 +^\bullet t_2]\!]^{vs} & = [\![t_1]\!]^{vs} + [\![t_2]\!]^{vs} \\ [\![t_1 \times^\bullet t_2]\!]^{vs} & = [\![t_1]\!]^{vs} \cdot [\![t_2]\!]^{vs} \end{array}$$

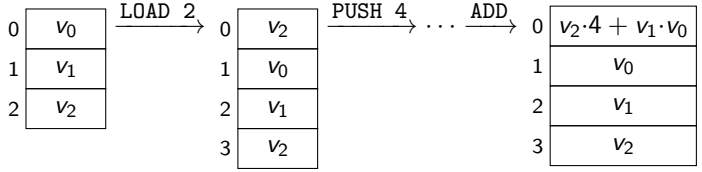
17.3. Kompilácia

XII.12 Programy a výrazy – príklad

Program

$$\texttt{LOAD 2, PUSH 4, MUL, LOAD 2, LOAD 2, MUL, ADD}$$

so zásobníkom, v ktorom sú čísla v_0, v_1, v_2:



vypočíta na vrch zásobníka hodnotu numerického výrazu

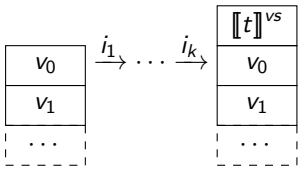
$$(x_2 \times 4) + (x_1 \times x_0)$$

pri ohodnotení premenných $vs = v_0, v_1, v_2, 0$

$$\begin{aligned}&\mathrm{Run}((\texttt{LOAD}(2), \texttt{PUSH}(4), \texttt{MUL}, \texttt{LOAD}(2), \texttt{LOAD}(2), \texttt{MUL}, \texttt{ADD}, 0), vs)\\ &\quad = [\![(x_2^\bullet \times^\bullet 4^\bullet) +^\bullet (x_1^\bullet \times^\bullet x_0^\bullet)]\!]^{vs}\end{aligned}$$

XII.13 Programy a výrazy – kompilácia

- Numerické výrazy – jazyk vyššej úrovne
- Programy postfixového stroja – jazyk nižšej úrovne
- Preklad výrazov do programov – *kompilácia*
- Kompilátor – funkcia $\mathrm{Comp}(t)$
- Vytvorí program $i_1, i_2, \ldots, i_k, 0$, ktorý na vrch zásobníka s hodnotami premenných vypočíta hodnotu výrazu t



$$\mathrm{Term}(t) \to \mathrm{Program}\ \mathrm{Comp}(t)$$
$$\mathrm{Term}(t) \to \mathrm{Run}(\mathrm{Comp}(t), vs) = [\![t]\!]^{vs}$$

XII.14 Kompilácia – príklad

Videli sme, že

$$\begin{aligned}&\mathrm{Run}((\texttt{LOAD}(2), \texttt{PUSH}(4), \texttt{MUL}, \texttt{LOAD}(2), \texttt{LOAD}(2), \texttt{MUL}, \texttt{ADD}, 0), vs)\\ &\quad = [\![(x_2^\bullet \times^\bullet 4^\bullet) +^\bullet (x_1^\bullet \times^\bullet x_0^\bullet)]\!]^{vs}\end{aligned}$$

Chceme, aby kompilátor skompiloval výraz

$$(x_2^\bullet \times^\bullet 4^\bullet) +^\bullet (x_1^\bullet \times^\bullet x_0^\bullet)$$

do programu

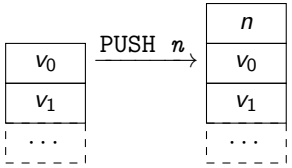
$$\texttt{LOAD}(2), \texttt{PUSH}(4), \texttt{MUL}, \texttt{LOAD}(2), \texttt{LOAD}(2), \texttt{MUL}, \texttt{ADD}, 0$$

teda

$$\begin{aligned}&\mathrm{Comp}((x_2^\bullet \times^\bullet 4^\bullet) +^\bullet (x_1^\bullet \times^\bullet x_0^\bullet))\\ &\quad = \texttt{LOAD}(2), \texttt{PUSH}(4), \texttt{MUL}, \texttt{LOAD}(2), \texttt{LOAD}(2), \texttt{MUL}, \texttt{ADD}, 0\end{aligned}$$

XII.15 Kompilácia – konštanty

- Hodnotou výrazu $n^\bullet$ je číslo n
- Aký program vypočíta na vrch zásobníka hodnotu n?

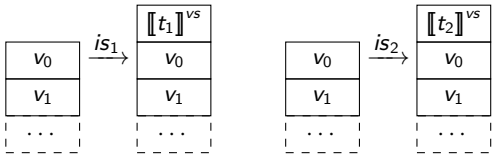


- Klauzula kompilátora:

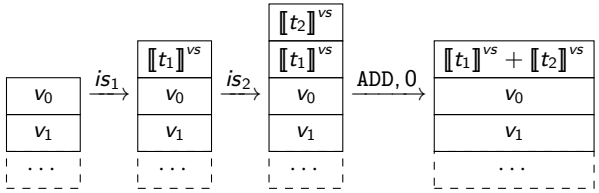
  $\mathrm{Comp}(n^\bullet) = \texttt{PUSH}(n), 0$

XII.16 Kompilácia – sčítanie

- Hodnotou výrazu $t_1 +^\bullet t_2$ je číslo $[\![t_1]\!]^{vs} + [\![t_2]\!]^{vs}$
- Akým programom ju vypočítame?
  1. Skompilujeme výrazy t_1 a t_2 Dostaneme programy $\text{Comp}(t_1) = is_1$ a $\text{Comp}(t_2) = is_2$

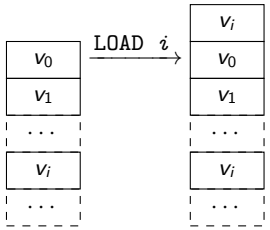


  2. Zreťazíme ich a pridáme inštrukciu pre sčítanie



XII.17 Kompilácia – premenná

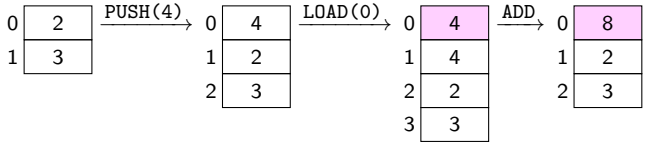
- Hodnotou výrazu $x_i^\bullet$ je číslo $vs[i]$
- Aký program dá na vrch zásobníka hodnotu $vs[i]$?



- Skompilujme $4^\bullet +^\bullet x_0^\bullet$:

$$\texttt{PUSH}(4), \texttt{LOAD}(0), \texttt{ADD}, 0$$

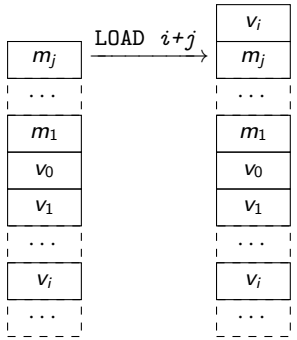
- Otestujme s ohodnotením $2, 3, 0$:



ZLE! `LOAD(0) malo načítať 2`

XII.18 Kompilácia – medzivýsledky

- Kompilátor si musí pamätať počet j *medzivýsledkov* na zásobníku
- $\text{Comp}(t) = \text{Comp}_1(0, t)$
- Aký program dá na vrch zásobníka hodnotu $vs[i]$?



- Počet medzivýsledkov vzrastie po vypočítaní t_1 vo výrazoch $t_1 +^\bullet t_2$ a $t_1 \times^\bullet t_2$

$$\text{Comp}_1(j, t_1 +^\bullet t_2) = \text{Comp}_1(j, t_1) \oplus \text{Comp}_1(\,j+1\,, t_2) \oplus (\texttt{ADD}, 0)$$

XIII. prednáška

Generovanie XML/XHTML

10. mája 2011

18. XML/XHTML

18.1. Štruktúra

XIII.1 Čo je XML/XHTML

XML/XHTML je formát na zápis štruktúrovaného textu

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Príklad</title>
  </head>
  <body>
    <p>Príklad
    <abbr title="Extensible Hypertext
          Markup Language">XHTML</abbr><br/>
    dokumentu.</p>
  </body>
</html>
```

XIII.2 Štruktúra XML/XHTML dokumentu

Štruktúra v XML dokumente – *elementy*

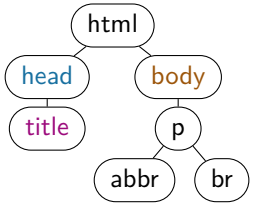
- úsek textu medzi otváracím tagom ( ) a príslušným zatváracím tagom ( )

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Príklad</title>
  </head>
  <body>
    <p>Príklad
    <abbr title="Extensible Hypertext
          Markup Language">XHTML</abbr><br/>
    dokumentu.</p>
  </body>
</html>
```

XIII.3 Strom elementov

Elementy vytvárajú stromovú štruktúru:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
      <title>Príklad</title>
  </head>
  <body>
    ...
  </body>
</html>
```



Čo v strome chýba?

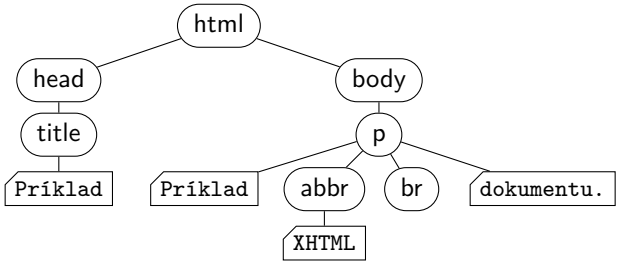
XIII.4 Neštruktúrovaný text

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Príklad</title>
  </head>
  <body>
    <p>Príklad
    <abbr title="Extensible Hypertext
          Markup Language">XHTML</abbr><br/>
    dokumentu.</p>
  </body>
</html>
```

V strome elementov chýba neštruktúrovaný text

XIII.5 Strom elementových a textových uzlov

Pridáme uzly pre neštruktúrovaný text:



Všeobecný strom s dvoma druhmi uzlov:

- elementové uzly – ľubovoľne veľa detí (aj 0, br)
- textové uzly – listy, žiadne deti

18.2. Kódovanie

XIII.6 Kódovanie stromu dokumentu v CL

- Strom XML dokumentu má 2 druhy uzlov
- Každý druh zakódujeme osobitným konštruktorom (podobne ako E a Nd)
- Textové uzly: Xs(t)

  t – text v uzle – reťazec
- Elementové uzly: Xe(n, as, cs)

  n – meno elementu (html, body, p, ...) – reťazec

  as – atribúty elementu

  cs – deti elementu

XIII.7 Kódovanie atribútov elementu

Xe(n, as, cs)

```
<a href="http://example.com/" class="external">link</a>
```

- Atribút je dvojica *meno*=*hodnota*
- Kódujeme dvojicou reťazcov:

  $\text{Attr}(n, v) \leftarrow \text{Str}(n) \wedge \text{Str}(v)$

- Všetky atribúty elementu zakódujeme zoznamom:

  $\text{Lattr}(0)$
  $\text{Lattr}(a, as) \leftarrow \text{Attr}(a) \wedge \text{Lattr}(as)$

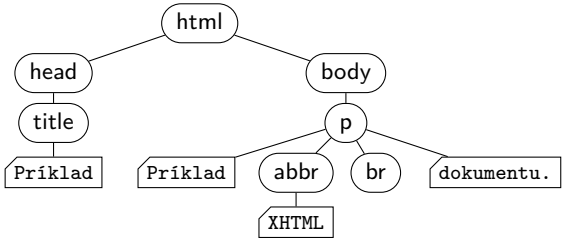
- Napríklad:

  $$(\text{'href'}, \text{'http://example.com/'}), (\text{'class'}, \text{'external'}), 0$$

XIII.8 Kódovanie detí elementu

Xe(n, as, cs)

- Elementové uzly majú rôzny počet detí:



- Deti zakódujeme do *zoznamu* uzlov XML stromu

XIII.9 Kódovanie XML dokumentu – príklad

```
Xe('html', (('xmlns', 'http://www.w3.org/1999/xhtml'), 0),
    Xe('head', 0,
        Xe('title', 0, Xs('Príklad'), 0),
        0),
    Xe('body', 0,
        Xe('p', 0,
            Xs('Príklad '),
            Xe('abbr',(('title','Extensible . . . Language'),0),
                Xs('XHTML'),
                0),
            Xe('br',0,0),
            Xs(' dokumentu.'),
            0),
        0),
 0)
```

XIII.10 Kódovanie XML – predikát

- Štruktúru binárneho stromu sme popísali predikátom:

  $\mathsf{Bt}(\bullet)$
  $\mathsf{Bt}\left(\frac{x}{\ell \mid r}\right) \leftarrow \mathsf{N}(x) \wedge \mathsf{Bt}(\ell) \wedge \mathsf{Bt}(r)$

- Ako popíšeme štruktúru XML stromu?
  - Predikát $\mathsf{X}(x)$ – x kóduje uzol XML stromu
  - Pomocný predikát $\mathsf{Lx}(xs)$ – xs je zoznam XML uzlov

XIII.11 Kódovanie XML – predikát

- Prirodzená definícia *vzájomnou rekurziou*:

  $\mathsf{X}\,\mathsf{Xe}(n, as, cs) \leftarrow \mathsf{Str}(n) \wedge \mathsf{Lattr}(as) \wedge \mathsf{Lx}(cs)$
  $\mathsf{X}\,\mathsf{Xs}(t) \leftarrow \mathsf{Str}(t)$

  $\mathsf{Lx}(0)$
  $\mathsf{Lx}(x, xs) \leftarrow \mathsf{X}(x) \wedge \mathsf{Lx}(xs)$

- CL neumožňuje vzájomnú rekurziu:

  $\mathsf{Lx}(0)$
  $\mathsf{Lx}(\mathsf{Xe}(n, as, cs), xs) \leftarrow \mathsf{Str}(n) \wedge \mathsf{Lattr}(as) \wedge \mathsf{Lx}(cs) \wedge \mathsf{Lx}(xs)$
  $\mathsf{Lx}(\mathsf{Xs}(t), xs) \leftarrow \mathsf{Str}(t) \wedge \mathsf{Lx}(xs)$

  $\mathsf{X}\,\mathsf{Xe}(n, as, cs) \leftarrow \mathsf{Str}(n) \wedge \mathsf{Lattr}(as) \wedge \mathsf{Lx}(cs)$
  $\mathsf{X}\,\mathsf{Xs}(t) \leftarrow \mathsf{Str}(t)$

XIII.12 Zobrazenie XML a kostra XHTML

- Ak v CL zakódujeme strom XML dokumentu, zobrazíme ho ako skutočné XML formátom Xml:

  Query: $\mathsf{Xe}(\text{'pokus'}, 0, \mathsf{Xs}(\text{'text'}), 0), 0 = xs : \mathsf{Xml}$

  Results: $xs =$ <pokus>text</pokus>

- Formátom Xml môžeme zobrazovať *iba* hodnoty xs:
  - jednoprvkové zoznamy uzlov: $\mathsf{Lx}(xs) \wedge \mathsf{L}(xs) = 1$
  - jediným prvkom je Xe
  - ▶ XML dokument musí mať jeden koreňový element

- Skratková funkcia $\mathsf{Xhtml}(t, cs)$ – kostra XHTML dokumentu

  t – titulok – reťazec

  cs – obsah tela (body) – Lx

  Formát Xml zobrazí výsledok ako skutočné XHTML

18.3. Neformálne typovanie

XIII.13 Neformálne typovanie

- CL je beztypový jazyk – všetky dáta sú čísla
- Môžeme typovať neformálne predikátmi

- $f :: T_1 \to T_2$ znamená:
  - T_1, T_2 sú predikáty
  - argument funkcie f je typu T_1
  - výsledok je typu T_2
  - skratka za:
$$T_1(x) \to T_2\, f(x)$$

- Napríklad funkcie na binárnych stromoch:

$$\begin{aligned} \text{Preorder} &:: \text{Bt} \to \text{Ln} \\ \text{Insert} &:: (\text{Bt}, \text{N}) \to \text{Bt} \\ \text{Extract\_max} &:: \text{Bt} \to (\text{N}, \text{Bt}) \end{aligned}$$

- Ako otypujeme Xe, Xs, Xhtml?

19. Organizácia: Opravný test a skúška

XIII.14 Opravný test

Termín: streda 18. mája o 13:30 v H3+H6

Podmienka: skóre zo semestrálnych testov ≥ 10 a < 30

Účel: doplniť body do 30

Materiál: úlohy podľa ex02–ex10

Systém: 12 úloh po 3 body
zarátajú sa iba body po odčítaní 10
súčet s bodmi zo semestra nepresiahne 30

XIII.15 Skúška

Termíny: štvrtky o 13:30
riadne: 19. mája, 2. a 9. júna v H3+H6

opravné: 16. a 23. júna v H6
najviac 40 študentov na termín

Podmienka: skóre z testov ≥ 30

Materiál: úlohy najmä podľa ex11, ex12
projektový štýl

Konzultácie: utorky 17. mája–14. júna, streda 22. júna
14:00–16:00 H3