Semestrální práce z předmětu KIV/TI

Lukáš Černý
A13B0286P
luccerny@students.zcu.cz

Jan Dvořák
A13B0293P
dvorakj@students.zcu.cz

12. prosince, 2014

Obsah

Kapitola 1

Úvod

1.1 Zadání

Vytvořte program, který umožní vizualizovat odvozování řetězců postupným zadáváním přepisovacích pravidel. Gramatiku si program načte ze souboru. Poté vypíše jednotlivá přepisovací pravidla a startovací symbol.

Zadání jsme si rožšířili o grafické uživatelské rozhraní. Tím jsme docílili toho, že pro běžného uživatele bude ovládání jednodušší.

Kapitola 2

Analýza problému

2.1 Návrh řešení

Abychom věděli co přepisovat, je třeba nejdříve nalézt nejlevější neterminální symbol. Pokud takový symbol v řetězci neexistuje, jedná se o konečný řetězec. Nejjednodušším řešením je procházení všech znaků a určování, zda nepatří do množiny neterminálních.

Algorithm 1

```
1: for znak in retezec do
2:     if znak = N then
3:         return znak
4: return −1
```

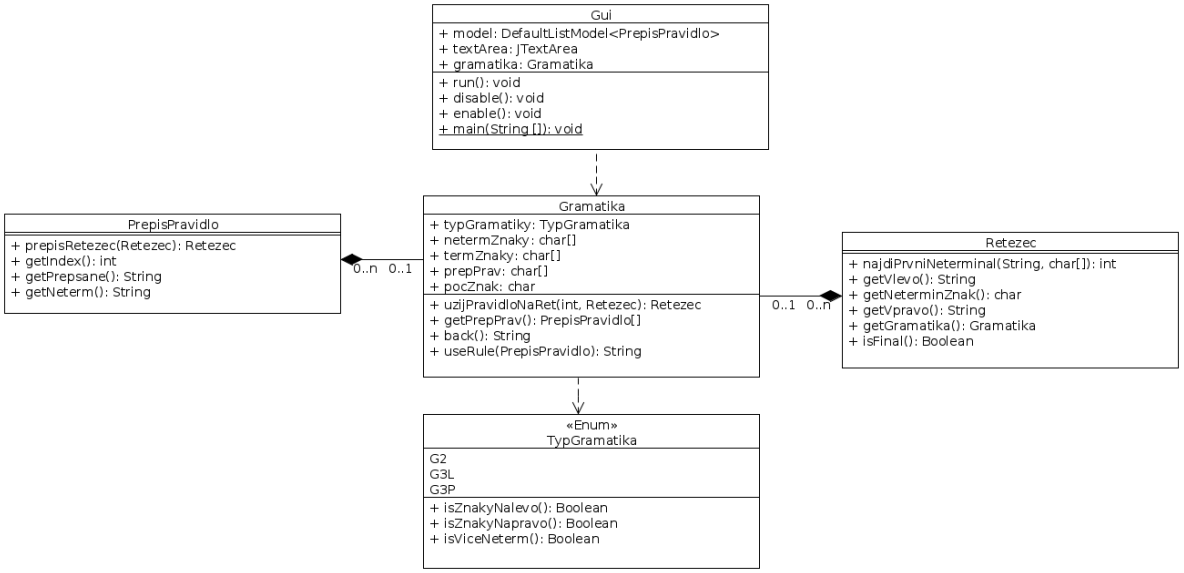
Dalším problémem byl, jak zařídit funci *undo*. Tento problém se dá řešit jednoduše. Postačí nám k tomu zásobník. Pokaždé když nějaký řetězec odvodíme, vložíme ho do něj. Když pak zavoláme metodu *undo*, jednoduše vytáhneme naposledy vložený prvek a současný zahodíme.

Je také důležité určit, zda zadaná gramatika vůbec splňuje podmínky pro to, aby byla daného typu. Pro tento účel se dají určit jednoduchá pravidla definovaná trojicí hodnot typu boolean:

- prvky nalevo od nejlevějšího neterminálního znaku
- znaky napravo od nejlevějšího neterminálního znaku
- více neterminálních znaků

Poté můžeme říct, že pokud chceme, aby gramatika byla typu G3P, musí každé z jejich pravidel splňovat pravidlo (true, false, false).

2.2 UML



Obrázek 2.1: UML diagram navrženého řešení problému

2.3 Formát vstupních souborů

```
G2 // typ souboru
3 // Počet prvků množiny neterminálních symbolů, neterminály jsou pak
implicitně A,B,C
2 // Počet prvků množiny terminálních symbolů, výstupy jsou pak implicitně
a, b
// prázdný řetězec budeme kódovat jako $
A // Počáteční symbol
// Přepisovací pravidla
A -> aBb | bCa
B -> aBb | $
C -> bCa | $
```

Kapitola 3

Uživatelská dokumentace

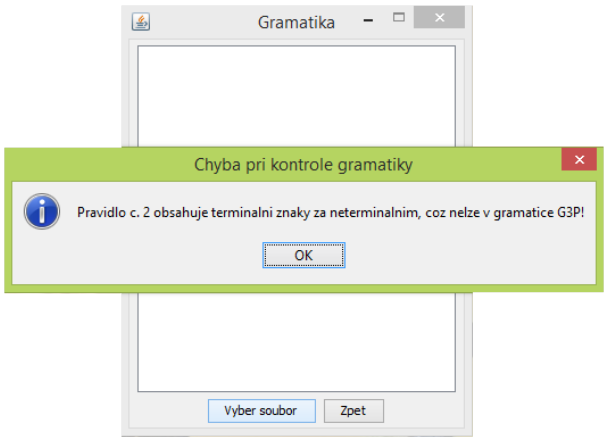
3.1 Spuštění

Pro spuštění aplikace je zapotřebí mít na svém počítači správně nainstalovanou Javu. Následné spuštění provedete poklepáním na program s názvem *Gramatika.jar*.

3.1.1 Načtění souboru

Při spuštění programu se zobrazí okno(viz. 3.1). Ovšem v tento moment není možné program jakkoliv ovládat. Všechny komponenty, kromě *Vyber soubor*, nereagují na pokyn uživatele. Nejdříve musíte načíst data z datového souboru uloženém na disku s koncovkou *gr*. Můžete použit také testovací soubory (*gramatika1.gr, gramatika2.gr, gramatika3.gr*), které jsou uloženy ve stejné složce, jako souštěný program. Načtete-li jiný soubor, program vyhodí hlášku (viz. 3.2) a umožní načíst nový soubor.

Obrázek 3.1: Spuštění programu



Obrázek 3.2: Chyba při kontrole gramatiky

3.2 Ovládání

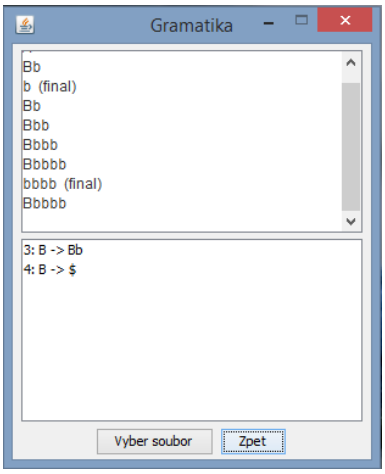
Po načtení datového souboru se Vám nastaví dostupná přepisovací pravidla pro počáteční znak.

Pro výběr dvakrát poklepejte na přepisovacího pravidlo a postupně se Vám bude vypisovat skládaný řetězec (viz. 3.3). Tento postup opakujte do doby, kdy budou k dispozici přepisovací pravidla.

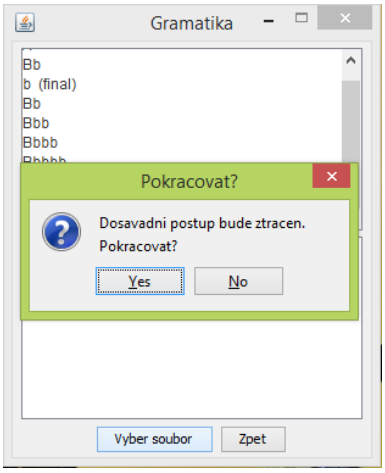
Při kroku zpět využijte tlačítko *Zpet*. Dostanete-li se touto operací až na začátek, program vypíše hlášku, že již nejde jít dále zpět a tlačítko se stane nestisknutelným.

Rozhodnete-li se načíst jiný datový soubor, budete upozorněni hláškou (viz.

3.4) a po načtení pokračujte obvyklým způsobem ovládání.



Obrázek 3.3: Výběr přepisovacích pravidel



Obrázek 3.4: Upozornění při načtení nového souboru

3.3 Zhodnocení práce

Podařilo se nám vytvořit zcela funkční program, který vizualizuje gramatiku. Splňujeme všechny požadované body a zadání jsme si ještě rozšířili o grafické uživatelské rozhraní.