

# Bakalářský projekt

## téma:

Tvorba  $\text{\LaTeX}$ ovských maker pro sazbu bezkontextových gramatik  
(tematický okruh přednášek *Formální jazyky a automaty*)



vypracoval: David Svoboda  
vedoucí projektu: RNDr. Ivana Černá

Fakulta Informatiky  
MU Brno  
24. června 1999

# Obsah

<b>1 Slovo úvodem</b>	<b>3</b>
<b>2 Řešení</b>	<b>4</b>
2.1 Příklad použití . . . . .	4
2.2 Syntaxe . . . . .	5
2.3 Implementace . . . . .	5
2.3.1 for-cyklus . . . . .	5
2.3.2 expanze makra . . . . .	6
2.3.3 test přítomnosti znaku v řetězci . . . . .	6
2.3.4 ověřování . . . . .	6
2.3.5 rozřídování . . . . .	7
2.3.6 načítání ze vstupu . . . . .	8
2.3.7 výstup . . . . .	9
<b>Literatura</b>	<b>10</b>

## 1 Slovo úvodem

Úkolem bylo vytvořit několik (pro přehlednost co nejméně)  $\text{\LaTeX}$ ovských maker, která by zjednodušovala práci při sazbě bezkontextových gramatik (dále jen CFG). Vstupem vytvořeného makra měla být sada přepisovacích pravidel – výstupem celá gramatika.

Proč celá gramatika? Copak pravidla jednoznačně určují celou CFG? Podle úmluvy se za *neterminály* považují vždy písmena velké abecedy, zbytek jsou *terminály*. Jako kořen gramatiky se obvykle volí první neterminál a pro označení gramatiky a množiny pravidel písmena  $G$  a  $P$ . Tímto je již zadávaná CFG určena jednoznačně. V případě, že by měl uživatel zájem změnit označení gramatiky či množiny pravidel, jsou k dispozici další makra. Implícitně se ale předpokládá výše uvedené značení.

## 2 Řešení

K tomu, aby vše běhalo tak jak má, bylo třeba vytvořit celou řadu ne zrovna triviální makro. V několika případech jsem úspěšně čerpal z Olšákova *TEXbooku naruby*, jindy muselo vzniknout nové makro. Proto, aby se budoucí uživatel nemusel učit nazpaměť hromadu pouček, jsem se snažil vytvořit co nejintuitivnější ovládání.

Vzhledem k tomu, že vstupem jsou přepisovací pravidla (která je na papíře každý zvyklý psát ve tvaru *levá strana*  $\rightarrow$  *pravá strana*), zvolil jsem jako šipku dvojici symbolů  $\rightarrow$  (spojovník a většítka). Pro symbol alternativní volby přepisu neterminálu jsem vybral znak | (kolmítko). Množina přepisovacích pravidel se zapisuje do *závorek* tvořených řídicími sekvencemi `\pravidla` a `\konecpravidel`.

### 2.1 Příklad použití

Zvolme libovolnou gramatiku:

```
\pravidla
F->ab|ha
G->er A->sEd|5r
A->tgt|gt{\epsilon}
S->fd|Gr|tG{\beta}y E->f{\alpha}|rett F->ei T->kj
E->aFg
F->fr F->wE|fi T->hU|j{\varepsilon}
\konecpravidel
```

Takto vypadá výsledek:

$$\begin{aligned} \overline{G} &= (\{F, G, A, E, S, T, U\}, \{a, b, h, e, r, s, d, \mathfrak{s}, t, g, f, y, i, k, j, w\}, \overline{P}, F), \text{ kde} \\ \overline{P} &= \{ F \rightarrow ab \quad | \quad ha, \\ &\quad G \rightarrow er, \\ &\quad A \rightarrow sEd \quad | \quad \mathfrak{s}r, \\ &\quad A \rightarrow tg \quad | \quad g\epsilon, \\ &\quad S \rightarrow fd \quad | \quad Gr \quad | \quad tG\beta y, \\ &\quad E \rightarrow f\alpha \quad | \quad rett, \\ &\quad F \rightarrow ei, \\ &\quad T \rightarrow kj, \\ &\quad E \rightarrow aFg, \\ &\quad F \rightarrow fr, \\ &\quad F \rightarrow wE \quad | \quad fi, \\ &\quad T \rightarrow hU \quad | \quad j\varepsilon \} \end{aligned}$$

Fakt, že gramatika i pravidla nesou jiné označení než implicitní  $G$  a  $P$ , je způsoben použitím makro:

```
\pojmenujgramatiku
\pojmenujpravidla
```

## 2.2 Syntaxe

Z výše uvedeného je již snadno vidět, jakým způsobem je možné pravidla zadávat. Přesto zde uvedu podmínky, které přesně vymezují korektnost použití:

```
\pravidla
< left > → < right > [< left > → < right > [...]]
...
\konecpravidel
s tím, že

< left >, < right > ∈ (< terminals > ∪ < nonterminals >)+
< nonterminals > = {A, B, C, ..., Z}
< terminals > = (< nonterminals > ∪ < space >)
```

## 2.3 Implementace

Zpočátku jsem nadefinoval řadu jednoduchých maker a registrů potřebných pro pozdější použití. Makra `\pojmenujgramatiku` a `\pojmenujpravidla` jak je již zřejmé z výše uvedeného textu, definují implicitní označení gramatiky a množiny pravidel. Snadno je lze předefinovat podle vlastní potřeby.

```
\newif\ifnext
\def\pojmenujgramatiku{G} %implicitni nastaveni jmena gramatiky
\def\pojmenujpravidla{P} %implicitni nastaveni oznaceni mnoziny pravidel
\def\mnozinapraavidel{}
\def\neterminaly{}
\def\terminaly{}
\def\koren{}
\newcount\tempnum
\def\oddelovac{&& \ | \ &&}
```

### 2.3.1 for–cyklus

Velice důležitou roli má primitivní for-cyklus. Jeho činnost jsem velice ocenil při testování přítomnosti znaku v seznamu.

```
%*****
% obecný for-cyklus
%*****
\def\for #1#2\endfor{\def\forbody #1{#2}\let\next=\forcycle%
\expandafter\next#1;}
%
\def\forcycle#1{\if \noexpand #1:\let\next=\relax%
\else \forbody #1%
\fi \next}
```

### 2.3.2 expanze makra

Makro `\přidej` způsobí expanzi stávajícího makra zadaného jako první parametr o nový element (druhý parametr).

```
%*****  
% expanze jiz definovaneho makra o novy retezec  
%*****  
\long\def\přidej#1#2{%  
  \xdef#1{\expandafter #1#2}  
}
```

### 2.3.3 test přítomnosti znaku v řetězci

Pro zařídování znaků mezi seznam terminálů či neterminálů je třeba vědět, zda uvažovaný znak již není mezi uvedenými (aby se neopakoval).

```
%*****  
% je zvoleny znak v retezci?  
%*****  
{\newcount\obsazen%  
  \obsazen=0%  
  \gdef\ifin#1#2{%  
    \def\test##1##2{%  
      \if##1##2 \advance \obsazen by 1%  
    \fi}%  
  \edef\retezec{#2}%  
  \edef\znak{#1}%  
  \for\retezec{\test\znak##1}\endfor%  
  \ifnum \obsazen>0 \nexttrue%  
  \else \nextfalse%  
  \fi%  
  \ifnext}}
```

### 2.3.4 ověřování

Makra `\overneterminal` a `\overterminal` ověřují, zda znak, který je jim předložen na vstupu jako parametr, spadá do jejich kompetence (je to buď terminál, nebo neterminál). V v kladném případě ho přidají do seznamu (Pokud tam již není – zabezpečuje test `\ifin`).

```
%*****  
% Je zadany znak jiz mezi terminaly/neterminaly?  
%*****  
\def\overneterminal#1{%  
  \ifx\empty\neterminaly \gdef\koren{#1}  
  \fi  
  \ifin#1\neterminaly \relax%  
  \else  
    \if#1\empty\relax  
    \else  
      \if:\neterminaly:\přidej\neterminaly{#1}
```

```

    \else\pridej\terminaly{,#1}
    \fi
  \fi
\fi}
%
\def\overterminal#1{%
  \ifin#1\terminaly \relax%
  \else
  \if#1\empty\relax
  \else
  \if:\terminaly:\pridej\terminaly{#1}
  \else\pridej\terminaly{,#1}
  \fi
  \fi
  %\add\terminaly#1
\fi}
%
```

### 2.3.5 rozřídování

Makro `\roztridznaky` postupně prochází všechny znaky (jeden za druhým) z právě probíraného prepisovacího pravidla a snaží se je rozřídít – viz výše uvedená makra `\overneterminal`, `\overterminal`, `\ifin`.

```

%*****
% zatřídování jednotlivých znaků mezi terminaly/neterminaly
%*****
\def\roztridznaky #1 {%
  \for{#1}{
    \ifcat\noexpand##1\epsilon %je to znak recke abecedy
    \if##1\epsilon\relax
    \else
    \if##1\varepsilon\relax %nechce ani \epsilon
    \else\pridej\terminaly{,#1} %ani \varepsilon
    \fi
    \fi
  \else
  \ifnum'##1>64 \ifnum'##1<91 \overneterminal##1
  \else \overterminal##1
  \fi
  \else \overterminal##1
  \fi
  \fi
}
\endfor}}
%
```

### 2.3.6 načítání ze vstupu

Makro `\pravidla` má na starosti načítání jednotlivých přepisovacích pravidel ze vstupního textu. Ty potom předhodí makru `\dekompozice` (viz dále), které zajistí jejich prozkoumání a správné rozčlenění do množin terminálových a ne-terminálových symbolů.

Podrobné vysvětlování jednotlivých obrátů a dobrých nápadů v rámci tohoto makra přenechávám na autora *TEXbooku naruby* Petra Olšáka.

```
%*****
% nacistani udaju ze vstupu:
% postupne se ctou jednotlivia pravidla a zpracovavaji
%*****
{\catcode'\^M=13 \catcode'\|=13%
 \gdef\pravidla{\bgroup \catcode'\^M=13 \catcode'\|=13%
 \let|= \oddelovac%
 \let^M= \jedenradek}%
 \gdef\jedenradek #1^M{\def\temp{#1}%
 \ifx \temp\posledniradek \def\next ##1 ^M {\uzavripravidla}%
 \else \relax%\message{Dalsi radek:}%
 \tempnum=0 \let\next= \polozka %
 \fi \next #1 ^M }%
 \gdef\poslednipolozka{^M} \gdef\posledniradek{\konecpravidel}%
 }
 \def\polozka #1 {\advance\tempnum by 1 \def\temp{#1}%
 \ifx \temp\poslednipolozka \let\next= \jedenradek%
 \else \dekompozice #1 %
 \fi \next}
%
%*****
% makro rozebirajici strukturu vlozenych pravidel
%*****
{
 \catcode'\|=13%
 \gdef\dekompozice #1->#2 {%
 \catcode'\|=13%
 \let|= \oddelovac
 \pridej\mnozinapraavidel{ $ #1 $ & $ \rightarrow $ & $ #2 , $ \cr }
 \let|= \empty%
 \roztridznaky #1
 \roztridznaky #2 }
}
```

### 2.3.7 výstup

Úplně na závěr pro provedení makra `\pravidla` se spustí makro `\uzavripravidla`, které způsobí vlastní vysázení gramatiky.

```
%*****  
% vystupni rutina  
%*****  
\def\uzavripravidla{%  
  \parindent=0pt  
  \par  
  $\pojmenujgramatiku=(%  
  $\lbrace\ne terminaly\rbrace,%  
  $\lbrace \terminaly \rbrace,%  
  $\pojmenujpravidla,\koren), kde$ \par  
  \pridej\mnozinapravidel{last}  
  \def\uprav ##1,$\cr last{\gdef\mnozinapravidel{##1\ \rbrace$\cr}}  
  \expandafter\uprav\mnozinapravidel  
  \edef\mnozinapravidel{\halign{#### && #### \cr \mnozinapravidel}}  
  %\message{[\mnozinapravidel]}  
  %\halign{## && ## \cr neco& \vbox{\mnozinapravidel} \cr}  
  \hbox{\vbox{\hbox{$\pojmenujpravidla=\lbrace \ $}}\vtop{\mnozinapravidel}}  
  \egroup}
```

## Literatura

- [1] P. Olšák, *TeX book naruby*. Konvoj, 1997
- [2] J. Rybička, *L<sup>A</sup>TeX pro začátečníky – 2. vydání*. Konvoj, 1999
- [3] zápisky z předášek M. Křetínského *Formální jazyky a automaty*, 1997