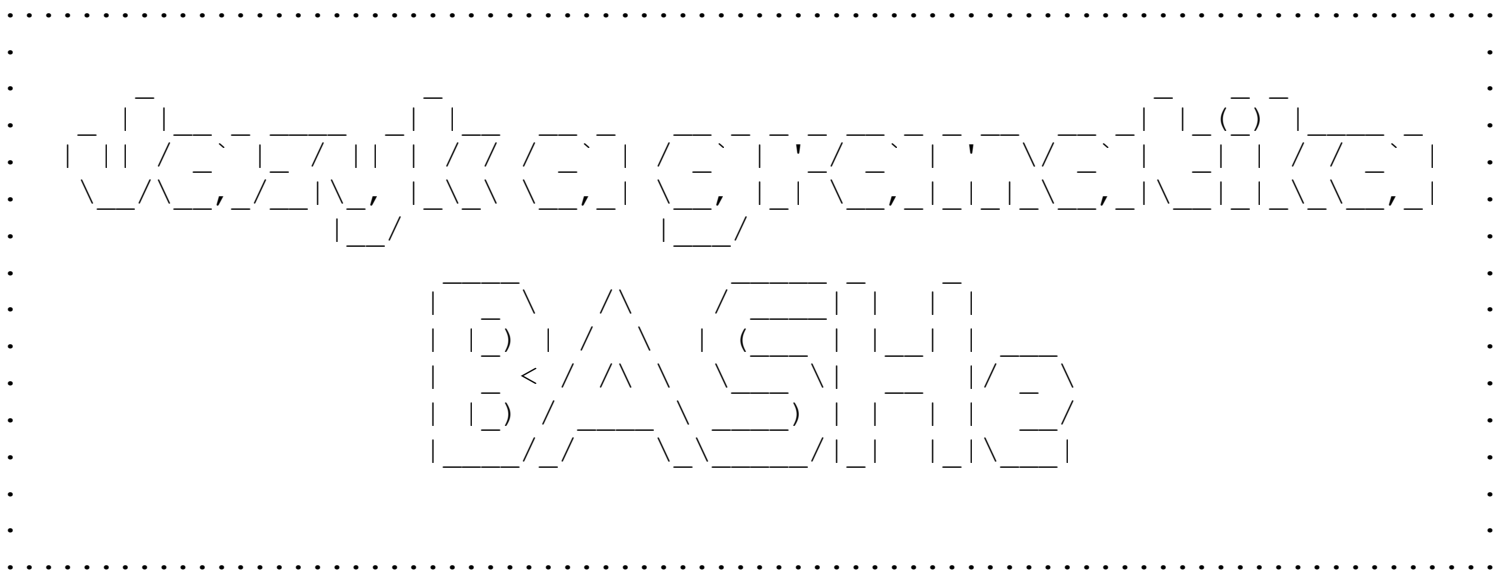
Jan Baier a Ondra Guth (FIT ČVUT)
14/10/2014

Obsah

Jazyk formálně

* Abeceda – *konečná* množina symbolů
* Řetězec – *konečná* posloupnost symbolů abecedy
* Jazyk – množina řetězců

S jazykem umíme dělat operace:

* Sjednocení, průnik, rozdíl
* Komplement (doplněk)
* Součin (zřetězení)
* N-tá mocnina

Gramatika

"Matematická" definice:

Uspořádaná čtveřice G = (N,T,P,S)

* N – *konečná* množina neterminálních symbolů
* T – *konečná* množina terminálních symbolů
* P – *konečná* množina "pravidel",
  podmnožina množiny $(N \cup T)^*.N.(N \cup T)^* \times (N \cup T)^*$
* S – počáteční symbol, $S \in N$

Zápis gramatiky – Klasicky matematicky

$G = (\{A, S\}, \{0, 1\}, P, S)$

$P = \{$
$\quad S \rightarrow 0A,$
$\quad A \rightarrow 1A \mid 0$
$\}$

$S \rightarrow 0A \rightarrow 01A \rightarrow 011A \rightarrow 0110$

Zápis gramatiky – Backus-Naurova forma (BNF)

G = ({<celé číslo>,<číslice>}, {0,1,2,3,4,5,6,7,8,9}, P, <celé číslo>)

P:

<celé číslo> ::= <číslice><celé číslo> | <číslice>

<číslice> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Klasifikace gramatik – Chomského hierarchie

1. Neomezená (typ 0)
   * Odpovídá klasické definici gramatiky

2. Kontextová (typ 1)
   * Pravidla ve tvaru $\gamma A \delta \rightarrow \gamma \alpha \delta$
   * $\gamma, \delta \in (N \cup T)^*$, $\alpha \in (N \cup T)^+$, $A \in N$
   * $S \rightarrow \varepsilon$ pouze ve speciálním případě

3. Bezkontextová (typ 2)
   * Pravidla ve tvaru $A \rightarrow \alpha$
   * $\alpha \in (N \cup T)^*$, $A \in N$

4. Regulární (typ 3)
   * Pravidla ve tvaru $A \rightarrow aB$ nebo $A \rightarrow a$
   * $A, B \in N$, $a \in T$
   * $S \rightarrow \varepsilon$ pouze ve speciálním případě

Klasifikace gramatik – BG

$G = (\{S\}, \{(,)\}, P, S)$

$P = \{$
$\quad S \rightarrow SS \mid (S) \mid \varepsilon$
$\}$

Klasifikace gramatik – KG

$G = (\{S, B, W, X\}, \{a, b, c\}, P, S)$

$P = \{$
$S \rightarrow abc \mid aSBc,$
$cB \rightarrow WB,$
$WB \rightarrow WX,$
$WX \rightarrow BX,$
$BX \rightarrow Bc,$
$bB \rightarrow bb$
$\}$

Klasifikace gramatik – RG

$G = (\{S, L\}, \{0, 1\}, P, S)$

$P = \{$
$\quad S \rightarrow 0L \mid 1S \mid 1,$
$\quad L \rightarrow 0S \mid 1L \mid 0$
$\}$

Klasifikace gramatik – NG

$G = (\{S, A, B\}. \{0, 1, 2\}, P, S)$

$P = \{$
$S \rightarrow 0AS \mid 1BS \mid 2,$
$A0 \rightarrow 0A,$
$A1 \rightarrow 1A,$
$A2 \rightarrow 20,$
$B0 \rightarrow 0B,$
$B1 \rightarrow 1B,$
$B2 \rightarrow 21$
$\}$

Klasifikace jazyků – Chomského hierarchie

1. Rekurzivně spočetné jazyky
    * Rozpoznatelné Turingovým strojem

2. Kontextové jazyky
    * Rozpoznatelné lineárně omezeným Turingovým strojem

3. Bezkontextové jazyky
    * Rozpoznatelné zásobníkovým automatem

4. Regulární jazyky
    * Rozpoznatelné konečným automatem

Schéma překladače

Zdrojový kód

→ *Lexikální analyzátor* →

Posloupnost symbolů

→ *Syntaktický analyzátor* →

Derivační strom

→ *Zpracování sémantiky* →

Vnitřní forma / Volání interpretačních podprogramů

→ *Generátor kódu* / *Interpret* →

Cílový program

Lexikální analýza

* Jazyk lexikálních tokenů je regulární
* Lze popsat regulární gramatikou
* Realizace konečným automatem
* Generátory lexikálních analyzátorů – lex, flex

Příklady:

* Jazyk identifikátorů
* Jazyk celých čísel

Hlavní úkol:

* Rozpoznat lexikální elementy (tokeny)
    * Identifikátor
    * Klíčové slovo
    * Literál
    * Speciální symbol
* Vynechat oddělovače lexikálních elementů
    * Mezery, nové řádky
    * Komentáře
* Rozpoznat a reagovat na speciální direktivy
    * #include v C/C++

Lexikální analyzátor a BASH

```
LineTerminator = \r|\n|\r\n
InputCharacter = [^\r\n]

Identifier = [a-zA-Z][a-zA-Z0-9_]*

Comment = "#"  {InputCharacter}* {LineTerminator}?
Shebang = "#!" {InputCharacter}* {LineTerminator}?

StringCharacter = [^\r\n\"\\]

/* string literal */
\"{StringCharacter}+\"      { return token(TokenType.STRING); }

/* Bash keywords */
"if"                        { return token(TokenType.KEYWORD,  IF); }
"fi"                        { return token(TokenType.KEYWORD, -IF); }

"@"                         { return token(TokenType.OPERATOR); }

/* comments */
{Shebang}                   { return token(TokenType.COMMENT2); }
{Comment}                   { return token(TokenType.COMMENT); }
. | {LineTerminator}        { /* skip */ }
```

Syntaktická analýza

Metoda shora dolů:

1. Expanze

   $\delta(q,\varepsilon,A) = \{(q,\alpha): A \rightarrow \alpha \in P\}; \forall A, A \in N$

2. Srovnání

   $\delta(q,a,a) = \{(q,\varepsilon)\}; \forall a, a \in T$

Metoda zdola nahoru:

1. Přesun

   $\delta(q,a,\varepsilon) = \{(q,a)\}; \forall a, a \in T$

2. Redukce

   $\delta(q,\varepsilon,\alpha) = \{(q,A): A \rightarrow \alpha \in P\}; \forall A, A \in N$

3. Přijetí

   $\delta(q,\varepsilon,\#S) = \{(r,\varepsilon)\}$

Syntaktická analýza – Metoda shora dolů

1. S → aSc
2. S → bSc
3. S → ε

(q, abcc, S)
  ⇒ (q, abcc, aSc) expanze 1.
  ⇒ (q, bcc, Sc) srovnání *a*
  ⇒ (q, bcc, bScc) expanze 2.
  ⇒ (q, cc, Scc) srovnání *b*
  ⇒ (q, cc, cc) expanze 3.
  ⇒ (q, c, c) srovnání *c*
  ⇒ (q, ε, ε) srovnání *c*

Pozn.: vrchol zásobníku je vlevo

Syntaktická analýza – Příklad

Mějme gramatiku G = ({S}, {a, b, c}, P, S) s pravidly P:

1. S → aSc
2. S → bSc
3. S → ε

Tato gramatika generuje třeba větu *abcc*

S → aSc → abScc → abcc

Syntaktická analýza – Metoda zdola nahoru

1. S → aSc
2. S → bSc
3. S → ε

(q, abcc, #)
⇒ (q, bcc, #a) přesun *a*
⇒ (q, cc, #ab) přesun *b*
⇒ (q, cc, #abS) redukce 3.
⇒ (q, c, #abSc) přesun *c*
⇒ (q, c, #aS) redukce 2.
⇒ (q, ε, #aSc) přesun *c*
⇒ (q, ε, #S) redukce 1.
⇒ (r, ε, ε) přijetí

Pozn.: vrchol zásobníku je vpravo

LL(k) syntaktická analýza

* Deterministická verze analýzy shora dolů
* Pravidlo se určí na základě *k* dopředu prohlížených symbolů
* Rozkladová tabulka, funkce FIRST a FOLLOW
* Implementace rekurzivním sestupem
* Generátory LL(1) analyzátorů – ANTLR, LLnextgen

LR(k) syntaktická analýza

* Deterministická verze analýzy zdola nahoru
* Pravidla pro redukci se určují na základě obsahu zásobníku
* Funkce BEFORE a EFF
* Generátory LALR(k) analyzátorů – yacc, bison

Tabulka symbolů

* Ukládání identifikátorů během lexikální analýzy
* Kontextové vazby
* Informace pro generování kódu
* Typová kontrola

Abstraktní syntaktický strom (AST)

* Reprezentace syntaktické struktury
* Vnitřní uzly jsou operátory
* Listy jsou operandy
* Překlad a optimalizace

BASH - EBNF

```
<letter> ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|
             A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

<digit> ::= 0|1|2|3|4|5|6|7|8|9

<number> ::= <digit>
           | <number> <digit>

<word> ::= <letter>
         | <word> <letter>
         | <word> '_'

<assignment_word> ::= <word> '=' <word>

<simple_command_element> ::= <word>
                           |  <assignment_word>
                           |  <redirection>
```

Děkuji za pozornost!

Nyní je prostor pro *vaše* dotazy…