Gramatiky pro generování regulárních výrazů

K (nejednoznačné) gramatice

$$R \longrightarrow a \mid b \mid R + R \mid RR \mid R^* \mid (R)$$

lze sestrojit ekvivalentní gramatiku, která je jednoznačná:

Gramatiky pro generování regulárních výrazů

K (nejednoznačné) gramatice

$$R \longrightarrow a \mid b \mid R + R \mid RR \mid R^* \mid (R)$$

lze sestrojit ekvivalentní gramatiku, která je jednoznačná:

$$\begin{aligned} R &\longrightarrow T + R \mid T \\ T &\longrightarrow FT \mid F \\ F &\longrightarrow F^* \mid (R) \mid C \\ C &\longrightarrow \ a \mid b \end{aligned}$$

Gramatiky pro generování regulárních výrazů

K (nejednoznačné) gramatice

$$R \longrightarrow a \mid b \mid R + R \mid RR \mid R^* \mid (R)$$

lze sestrojit ekvivalentní gramatiku, která je jednoznačná:

$$\begin{aligned} R &\longrightarrow T + R \mid T \\ T &\longrightarrow FT \mid F \\ F &\longrightarrow F^* \mid (R) \mid C \\ C &\longrightarrow \ a \mid b \end{aligned}$$

jiný ekvivalentní tvar, bez tzv. levé rekurze $F \longrightarrow F^*$:

Gramatiky pro generování regulárních výrazů

K (nejednoznačné) gramatice

$$R \longrightarrow a \mid b \mid R + R \mid RR \mid R^* \mid (R)$$

lze sestrojit ekvivalentní gramatiku, která je jednoznačná:

$$\begin{aligned} R &\longrightarrow T + R \mid T \\ T &\longrightarrow FT \mid F \\ F &\longrightarrow F^* \mid (R) \mid C \\ C &\longrightarrow \ a \mid b \end{aligned}$$

jiný ekvivalentní tvar, bez tzv. levé rekurze $F \longrightarrow F^*$:

pravidla $F \longrightarrow F^* \mid (R) \mid C$ nahradíme pravidly

$$\begin{aligned} F &\longrightarrow BH \\ B &\longrightarrow (R) \mid C \\ H &\longrightarrow {}^*H \mid \varepsilon \end{aligned}$$

Gramatika vhodná pro syntaktickou analýzu shora dolů

1. $R \longrightarrow T + R$
2. $R \longrightarrow T$
3. $T \longrightarrow FT$
4. $T \longrightarrow F$
5. $F \longrightarrow BH$
6. $B \longrightarrow (R)$
7. $B \longrightarrow C$
8. $H \longrightarrow {}^*H$
9. $H \longrightarrow \varepsilon$
10. $C \longrightarrow a$
11. $C \longrightarrow b$

```
def syntAnalRV(rv):

    idx = {'val' : 0}

# 1: R --> T+R, 2: R --> T
    def zpR():
        zpT()
        if  idx['val'] >= len(rv):
            print 2 #  2: R --> T
        elif rv[idx['val']] == '+':
            idx['val'] += 1
            zpR()
            print 1 #  1: R --> T+R
```

```
# 3: T --> FT, 4: T --> F
def zpT():
  zpF()
  if (idx['val'] < len(rv) and
                rv[idx['val']] in ('a', 'b', '('):
      zpT()
      print 3 #  3: T --> FT
  else:
      print 4 #  4: T --> F
```

```
# 5: F --> BH
    def zpF():
        zpB()
        zpH()
        print 5 # 5: F --> BH
```

```
# 6: B --> (R), 7: B --> C
    def zpB():
        if (rv[idx['val']] in ('a', 'b')):
            zpC()
            print 7 # 7: B --> C
        elif (rv[idx['val']] == '('):
            idx['val'] += 1
            zpR()
            if (rv[idx['val']] != ')'):
                raise Exception("ERROR 1")
            else:
                print 6 # 6: B --> (R)
                idx['val'] += 1
        else:
            raise Exception("ERROR 2")
```

```
# 8: H --> *H , 9: H --> epsilon
    def zpH():
        if (idx['val'] < len(rv) and rv[idx['val']] == '*'):
            print 8 # 8: H --> *H
            idx['val'] += 1
            zpH()
        else:
            print 9 #  9: H --> epsilon

# 10: C --> a, 11: C --> b
    def zpC():
        if (rv[idx['val']] == 'a'):
            print 10 #  10: C --> a
        elif(rv[idx['val']] == 'b'):
            print 11 #  11: C --> b
        else:
            raise Exception("ERROR 3")
        idx['val'] += 1
```

```
    zpR();
    print "OK"
# end of syntaAnalRV(rv)


syntAnalRV("(aa+b)*")
```

automat NFA(node v)
{

if ($v.symb =$ 'a') return

	a	b	ε
$\rightarrow q_1$	q_2	-	-
(q_2)	-	-	-

;

if ($v.symb =$ 'b') return

	a	b	ε
$\rightarrow q_1$	-	q_2	-
(q_2)	-	-	-

;

automat NFA(node v)
$\{$

if ($v.symb =$ 'a') return

	a	b	ε
$\rightarrow q_1$	q_2	-	-
(q_2)	-	-	-

;

if ($v.symb =$ 'b') return

	a	b	ε
$\rightarrow q_1$	-	q_2	-
(q_2)	-	-	-

;

if ($v.symb =$ 'R' and $v.rule =$ "$R \longrightarrow R + T$")
return $\mathrm{UNION}(\,\mathrm{NFA}(v.succ_1), \mathrm{NFA}(v.succ_3)\,)$;

automat NFA(node v)
$\{$

if $(v.symb = \text{`}a\text{'})$ return

	a	b	ε
$\rightarrow q_1$	q_2	-	-
(q_2)	-	-	-

;

if $(v.symb = \text{`}b\text{'})$ return

	a	b	ε
$\rightarrow q_1$	-	q_2	-
(q_2)	-	-	-

;

if $(v.symb = \text{`}R\text{'}$ and $v.rule = \text{“}R \longrightarrow R + T\text{”})$
return $\mathrm{UNION}(\mathrm{NFA}(v.succ_1), \mathrm{NFA}(v.succ_3))$;
if $(v.symb = \text{`}T\text{'}$ and $v.rule = \text{“}T \longrightarrow FT\text{”})$ return
$\mathrm{CONC}(\mathrm{NFA}(v.succ_1), \mathrm{NFA}(v.succ_2))$;

automat NFA(node v)
{

if ($v.symb =$ 'a') return

	a	b	ε
$\rightarrow q_1$	q_2	-	-
(q_2)	-	-	-

;

if ($v.symb =$ 'b') return

	a	b	ε
$\rightarrow q_1$	-	q_2	-
(q_2)	-	-	-

;

if ($v.symb =$ 'R' and $v.rule =$ "$R \longrightarrow R + T$")
return $\mathrm{UNION}(\mathrm{NFA}(v.succ_1), \mathrm{NFA}(v.succ_3))$;
if ($v.symb =$ 'T' and $v.rule =$ "$T \longrightarrow FT$") return
$\mathrm{CONC}(\mathrm{NFA}(v.succ_1), \mathrm{NFA}(v.succ_2))$;
if ($v.symb =$ 'F' [and $v.rule =$ "$F \longrightarrow BH$"]):
if $v.succ_2.succ_1.symb =$ '$*$' return $\mathrm{ITER}(\mathrm{NFA}(v.succ_1))$
else return $\mathrm{NFA}(v.succ_1)$;

automat NFA(node v)
{

if ($v.symb =$ 'a') return

	a	b	ε
$\rightarrow q_1$	q_2	-	-
(q_2)	-	-	-

;

if ($v.symb =$ 'b') return

	a	b	ε
$\rightarrow q_1$	-	q_2	-
(q_2)	-	-	-

;

if ($v.symb =$ 'R' and $v.rule =$ "$R \longrightarrow R + T$")
return $\mathrm{UNION}(\mathrm{NFA}(v.succ_1), \mathrm{NFA}(v.succ_3))$;
if ($v.symb =$ 'T' and $v.rule =$ "$T \longrightarrow FT$") return
$\mathrm{CONC}(\mathrm{NFA}(v.succ_1), \mathrm{NFA}(v.succ_2))$;
if ($v.symb =$ 'F' [and $v.rule =$ "$F \longrightarrow BH$"]):
if $v.succ_2.succ_1.symb =$ '$*$' return $\mathrm{ITER}(\mathrm{NFA}(v.succ_1))$
else return $\mathrm{NFA}(v.succ_1)$;
if ($v.symb =$ 'B' and $v.rule =$ "$B \longrightarrow (R)$") return $\mathrm{NFA}(v.succ_2)$);
if ($v.symb =$ 'H') return () # nevraci zadny automat;

automat NFA(node v)
{

if ($v.symb =$ 'a') return

	a	b	ε
$\rightarrow q_1$	q_2	-	-
(q_2)	-	-	-

;

if ($v.symb =$ 'b') return

	a	b	ε
$\rightarrow q_1$	-	q_2	-
(q_2)	-	-	-

;

if ($v.symb =$ 'R' and $v.rule =$ "$R \longrightarrow R + T$")
return $\mathrm{UNION}(\mathrm{NFA}(v.succ_1), \mathrm{NFA}(v.succ_3))$;
if ($v.symb =$ 'T' and $v.rule =$ "$T \longrightarrow FT$") return
$\mathrm{CONC}(\mathrm{NFA}(v.succ_1), \mathrm{NFA}(v.succ_2))$;
if ($v.symb =$ 'F' [and $v.rule =$ "$F \longrightarrow BH$"]):
if $v.succ_2.succ_1.symb =$ '$*$' return $\mathrm{ITER}(\mathrm{NFA}(v.succ_1))$
else return $\mathrm{NFA}(v.succ_1)$;
if ($v.symb =$ 'B' and $v.rule =$ "$B \longrightarrow (R)$") return $\mathrm{NFA}(v.succ_2)$);
if ($v.symb =$ 'H') return () # nevraci zadny automat;
otherwise return $\mathrm{NFA}(v.succ_1)$
}