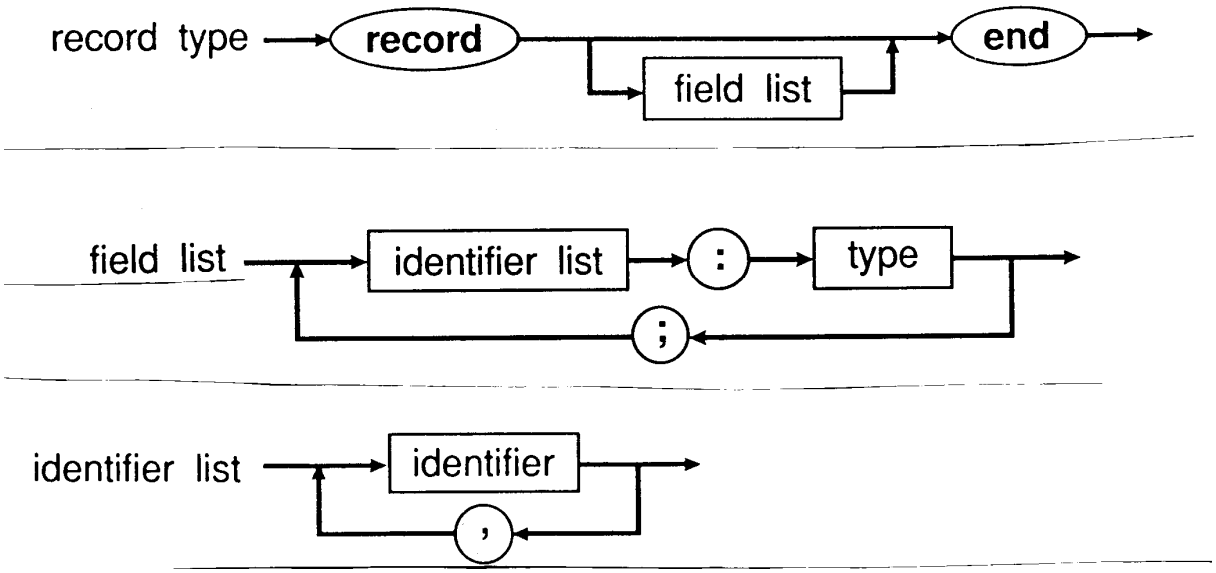
„PASCAL“ – syntax

Syntaktický graf



BNF Backusova-Naurova forma

record type **::= record end | record** ***field list*** **end**

field list **::=** ***identifier list*** **:** ***type*** **{ ;** ***identifier list*** **:** ***type*** **}**

identifier list **::=** ***identifier*** **{ ,** ***identifier*** **}**

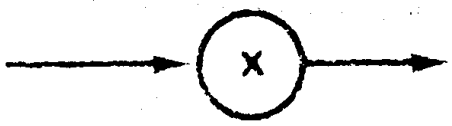
gramatika
znak, symbol, metasymbol, věta

terminální symboly - dále malá písmena
neterminální symboly *kurziva* – dále velká písmena
přepisovací pravidla

Konstrukce syntaktického grafu

1. Neterminální symbol *A* se zobrazí podle pravidel 2 – 6.

2. Výskyt terminálního symbolu x v ξ_i se zobrazí



Rozpoznání symbolu a načtení dalšího

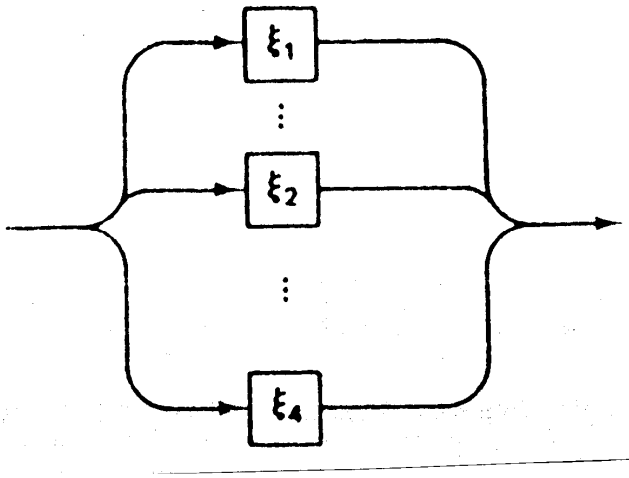
3. Neterminální symbol *B* se zobrazí

Volání programu na analýzu B.

4. Přepisovací pravidlo tvaru

$$A ::= \xi_1 \mid \xi_2 \mid \ldots \mid \xi_n$$

se zobrazí



5. Řetězec ξ tvaru

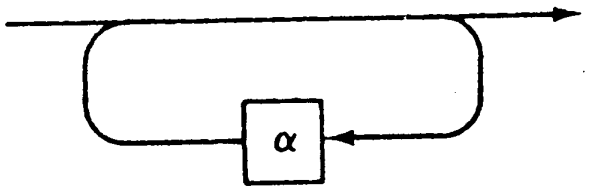
$$\xi = \alpha_1 \alpha_2 \ldots \alpha_m$$

se zobrazí

6. Řetězec ξ tvaru

$$\xi = \{\alpha\}$$

se zobrazí



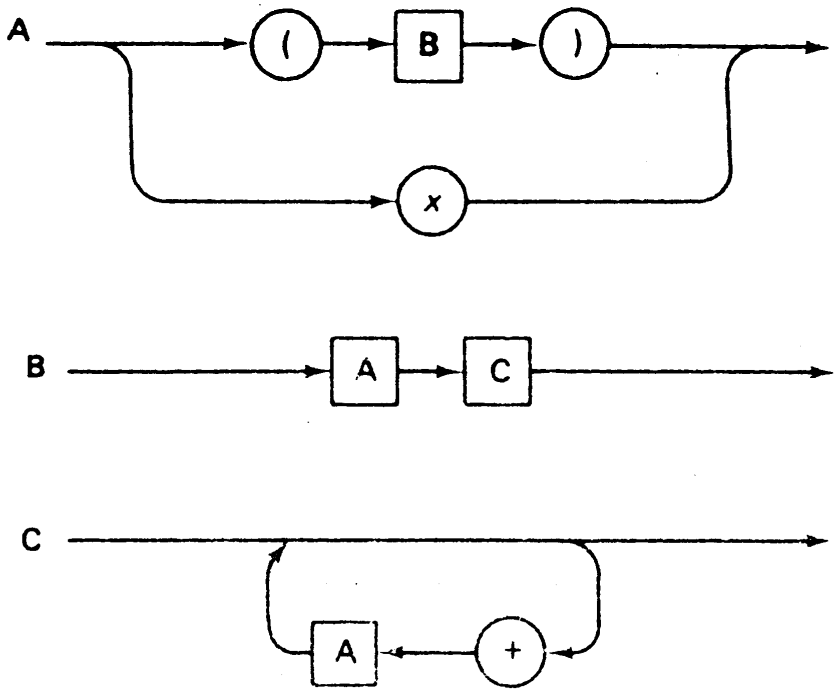
Příklad

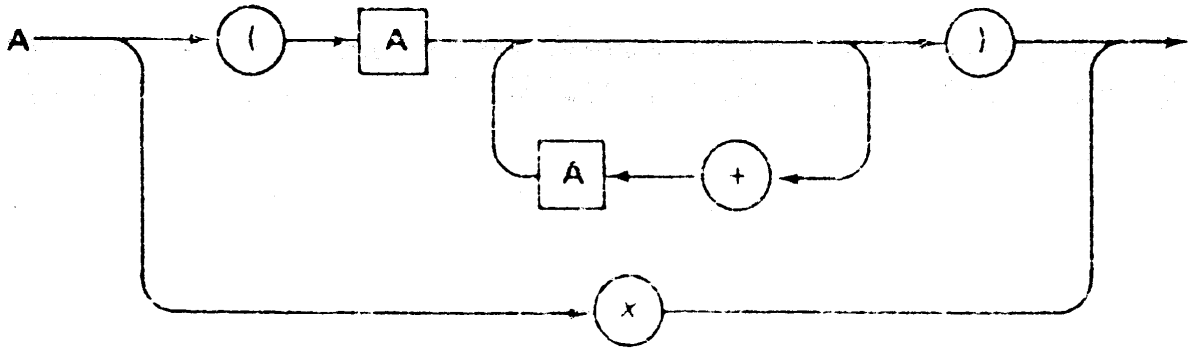
***A* ::= x|(*B*)**
B* ::= *AC
***C* ::= {+*A*}**

z *A* generuje

x
(x)
(x+x)
((x))

Grafy





Redukovaný syntaktický graf

Konstrukce syntaktického analyzátoru z grafu

procedura getsym rozpozná další symbol
zjednodušení symboly=znaky (viď příklad)
getsym=read(ch)

T(S)* příkaz překladem grafu *S

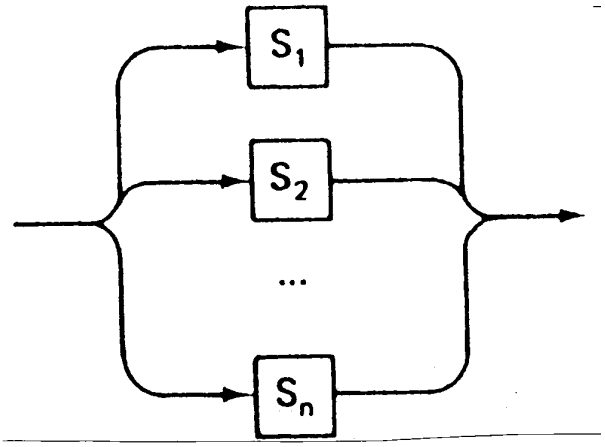
0. Každý graf se přeloží do procedury

1. Posloupnost

se přeloží

begin $T(S_1)$; $T(S_2)$; ...; $T(S_n)$ **end**

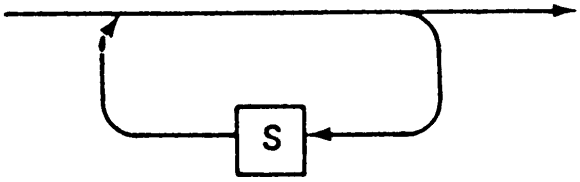
2. Větvení



se přeloží **case** *ch* **of**
L_1:$T(S_1)$;
...
L_n:$T(S_n)$
else error
end

kde L_i je množina začátečních symbolů S_i

3. Smyčka



se přeloží

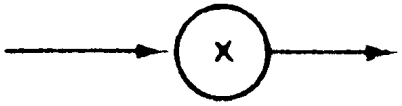
while ch **in** L **do** $T(S)$

kde L **je množina začátečních symbolů** S

4. Vrchol grafu s jiným grafem – neterminálním symbolem

se přeloží na volání procedury A

6. Vrchol s terminálním symbolem



se přeloží na příkaz

if $ch = x$ **then** read (ch) **else** error

```
program analyzator(input, output);
  var ch:char;
  procedure A;
  begin if ch = ´x´ then read(ch)else
    if ch = ´(´ then
    begin read(ch); A;
      while ch = ´+´ do
        begin read(ch); A end;
      if ch = ´)´ then read(ch)
                  else error
    end
  end;
begin read(ch); A
end.
```

Niklaus Wirth: Algoritmy a štruktúry údajov.s.378-397

Příkazový řádek – návrh gramatiky

```
ŘÁDEK ::= PŘÍKAZ VSTPŘES {´|´ PŘÍKAZ}
                  VYSPŘES ´eol´

PŘÍKAZ ::= JMÉNO {PARAMETR}

JMÉNO ::= řetězec

PARAMETR ::= ´-´ písmeno

VSTPŘES ::= null |´<´JMÉNO

VYSPŘES ::= null |´>´JMÉNO
```

přidáme argumenty

```
PŘÍKAZ ::= JMÉNO {PARAMETR} {ARGUMENT}

ARGUMENT ::= řetězec
```

přidáme asynchronní vykonávání

```
AŘÁDEK ::= ... ´&´
```