Syntaktická analýza metodou rekurzivního sestupu

Základní vlastnosti:

- **Zvládá práci s bezkontextovými gramatikami BKG (nutné pro popis syntaxe programovacích jazyků, nejsou regulární). Obecný tvar pravidel:**

$$A \rightarrow \alpha \qquad \text{kde } \alpha \in (N \cup T)^*$$

- **Používá deterministická analýza shora dolů (sestup)**
- **Je to metoda vyjádřená vzájemně se volajícími podprogramy (rekurzivní procedury)**

Pozn.:Obecně je analýza BK jazyka úloha řešitelná metodou s návratem s kubickou složitostí (když se nepodaří generovat/akceptovat daný řetězec, vrátíme se a zkusíme jinou možnost). Pokud ji zvládneme rekurzivním sestupem, pak je složitost lineární.

Princip:

- **Každému neterminálnímu symbolu A odpovídá procedura A**
- **Tělo procedury je dáno pravými stranami pravidel pro A**

  A → X11 X12 … X1n | X21 X22 ... X2m | ... | Xp1 Xp2 ... Xpq

  pravé strany musí být rozlišitelné na základě symbolů vstupního řetězce aktuálních v okamžiku uplatnění příslušné pravé strany
- **Je-li rozpoznána pravá strana Xi1 Xi2 ... Xik,**

  pak prováděj pro j = 1 .. K

  1. **Je-li symbol Xij neterminální, vyvolá se v A proceduru Xij**
  2. **Je-li " Xij terminální, ověří A přítomnost Xij ve vstupním řetězci a zajistí přečtení dalšího symbolu ze vstupu**

- **Rozpoznané pravidlo analyzátor oznámí (např. výpisem čísla pravidla)**
- **Chybnou strukturu vstupního řetězce oznámí chybovým hlášením**

Pro rozpoznání správné pravé strany musí platit:

- ✓ **-řetězce derivovatelné z pravých stran začínají různými terminálními symboly**
- ✓ **-při prázdné pravé straně se musí navíc lišit i od těch terminálních symbolů, které se mohou vyskytnout v derivacích za neterminálem z levé strany pravidla.**

Např.
příkaz může začínat *if, while, do, identifikátorem, call, …* tím lze rozlišovat,
ale jak poznat neúplný podm.příkaz od úplného (s *else*)?
Prodiskutujte to.

Př. **Gramatika příkazu (použijeme zde metasymboly opakování { + T } a { * F }**
je to tzv. iterační zápis gramatiky, gramatika s regulární pravou stranou)

(1,2)	**S → V = E | if E then S Z**
(3,4)	**Z → else S | e**
(5)	**E → T { + T }**
(6)	**T → F { * F }**
(7,8)	**F → (E) | V**
(9)	**V → a I**
(10,11)	**I → (E) | e**

1.Zkuste ji napsat i normálně (bez metasymbolů) a diskutujte možný problém s pravidly

E → E { + E } | E { * E } | (E) | …

2. Zjistěte, jak vypadají množiny *first* terminálních symbolů, kterými začínají řetězce odvoditelné z jednotlivých pravých stran pravidel (dovolí to provést výběr té správné pr.strany na kterou se má expandovat neterminál korespondující dané proceduře)

Symbol	first	follow
S		
Z		
E		
T		
F		
V		
I		

3.Zjistěte, jak vypadají množiny *follow* terminálních symbolů, které následují ve vstupu po provedení příslušné procedury a zkuste si jak vypadá struktura věty

if a then if a then a = a else a = a

Řešení ad 1.

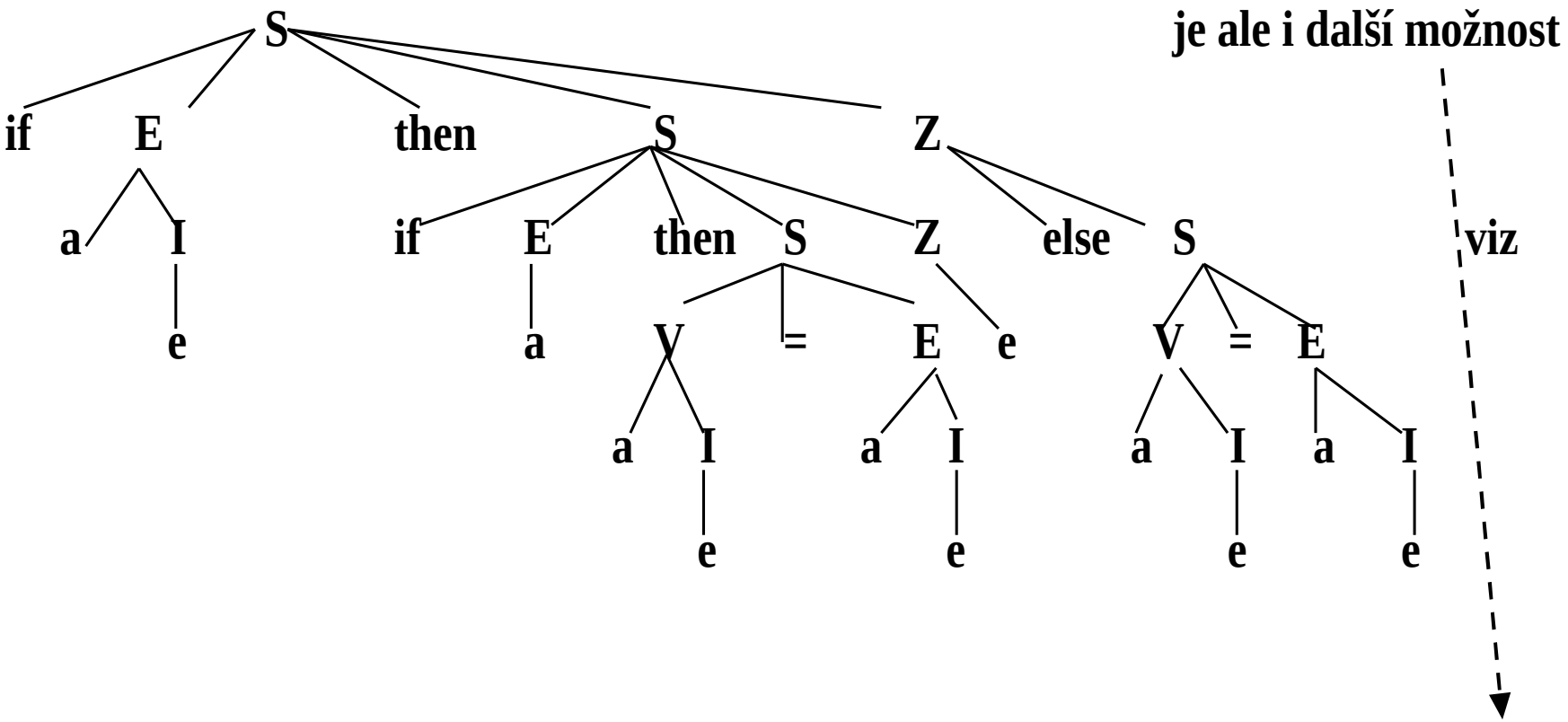
E → T | E + T T → F | T * F

a co tohle? E → T | T + E T → F | F * T

Řešení ad 2.

Symbol	first	follow
S	a, if	e, else
Z	e, else	e, else
E	a, (	), then, e, else
T	a, (	), then, e, else, +
F	a, (	), then, e, else, +, *
V	a	), then, e, else, +, *, =
I	e, (	), then, e, else, +, *, =

Řešení ad 3.



Tohle je vnoření neúplného podm. příkazu do úplného, gramatika umožňuje i opak

Dokument
ikace Microsoft W

- **Lexikální analýzu bude provádět procedura CTI**
- **Hlášení chyb bude provádět procedura CHYBA**
- **Posloupnost přepisovacích pravidel vypisuje procedura TISK**

```
program SYNTAKTICKA_ANALYZA
definice vyjmenovaneho typu SYMBOL = (IDENT, PRIRAZ, PLUS, KRAT,  LEVA,
PRAVA, IFS, THENS, ELSES);
 promenna  N typu SYMBOL;
 procedura CTI(vystupni parametr S typu SYMBOL)   ...
 procedura CHYBA(vstupni parametr CISLO typu integer)  ...
 procedura TISK(vstupni parametr CISLO typu integer)  ...

 procedura S
 { if N =  IFS then
                { TISK(2); CTI(N); E;
                     if N ≠ THENS then CHYBA(2);
                     else { CTI(N); S; Z;
                            }
                }
     else
       { TISK(1); V; if N ≠ PRIRAZ then CHYBA(1);
                          else { CTI(N); E;
                                 }
       }   /* vstupni řetězec patří do jazyka */
 }
```

```
procedura Z
{ if N = ELSES then { TISK(3); CTI(N); S;
                              }
  else TISK(4);      /*bude se chovat tak, jak jsme nakreslili strom, nebo jinak? */
}
 procedura E
{ TISK(5); T;
     while N = PLUS do {  CTI(N); T;
                                 }
}
 procedura T
{ TISK(6); F;
     while N = KRAT do {  CTI(N); F;
                                  }
}
 procedura F
{ if N = LEVA then { TISK(7); CTI(N); E;
                                    if N ≠ PRAVA then CHYBA(7)
                                                 else CTI(N)
                           }
  else { TISK(8); V; }
}
```

```
 procedura V
{ if N ≠ IDENT then CHYBA(9)  else { TISK(9);CTI(N);I;
                                                    }
}

procedura I
{ if N = LEVA then { TISK(10); CTI(N); E;
                                          if N ≠ PRAVA then CHYBA(10)
                                                                else CTI(N)
                    }
                          else TISK(11);
}

procedura  MAIN
{  CTI(N); S;
}
```

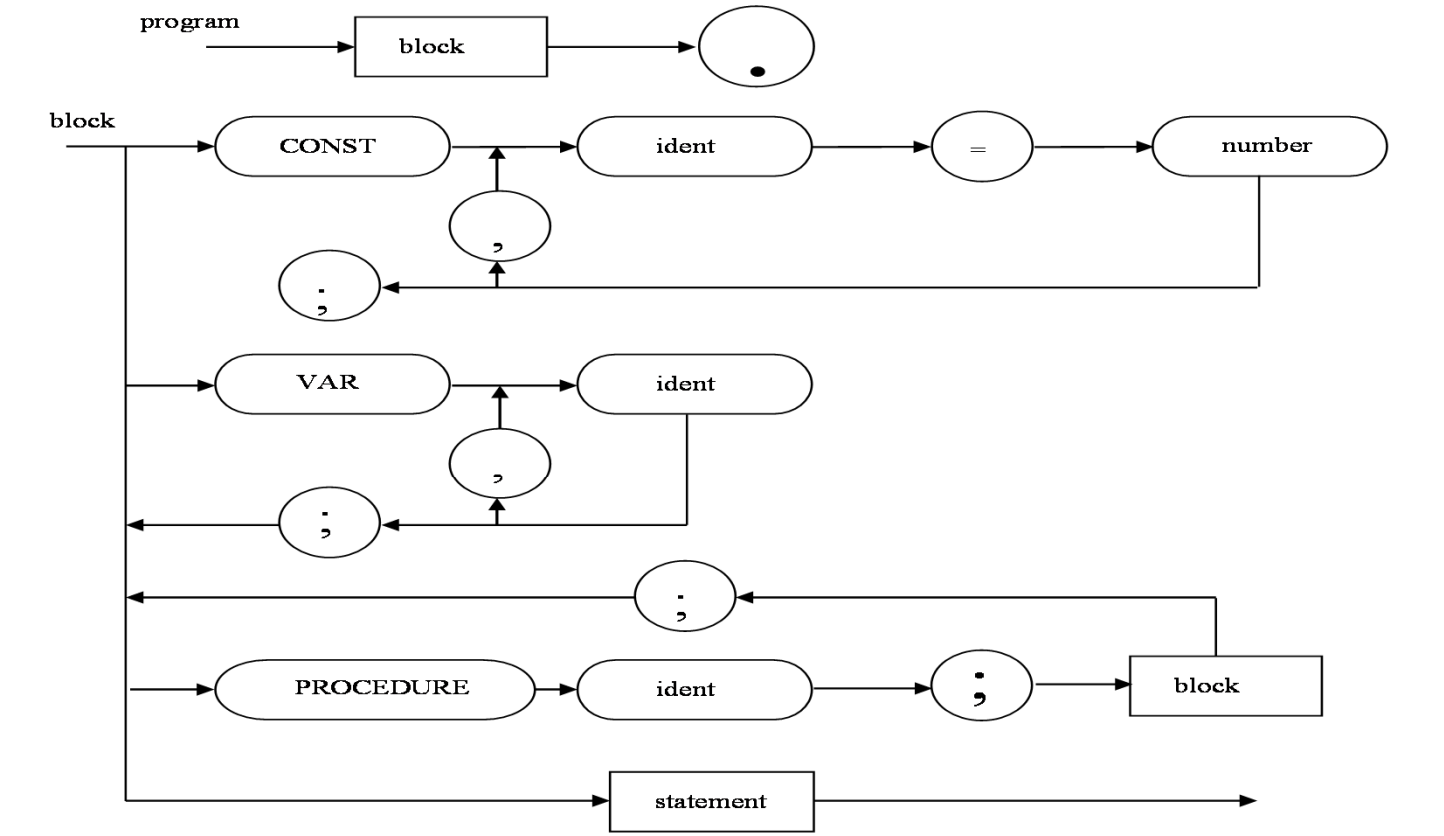
Př.

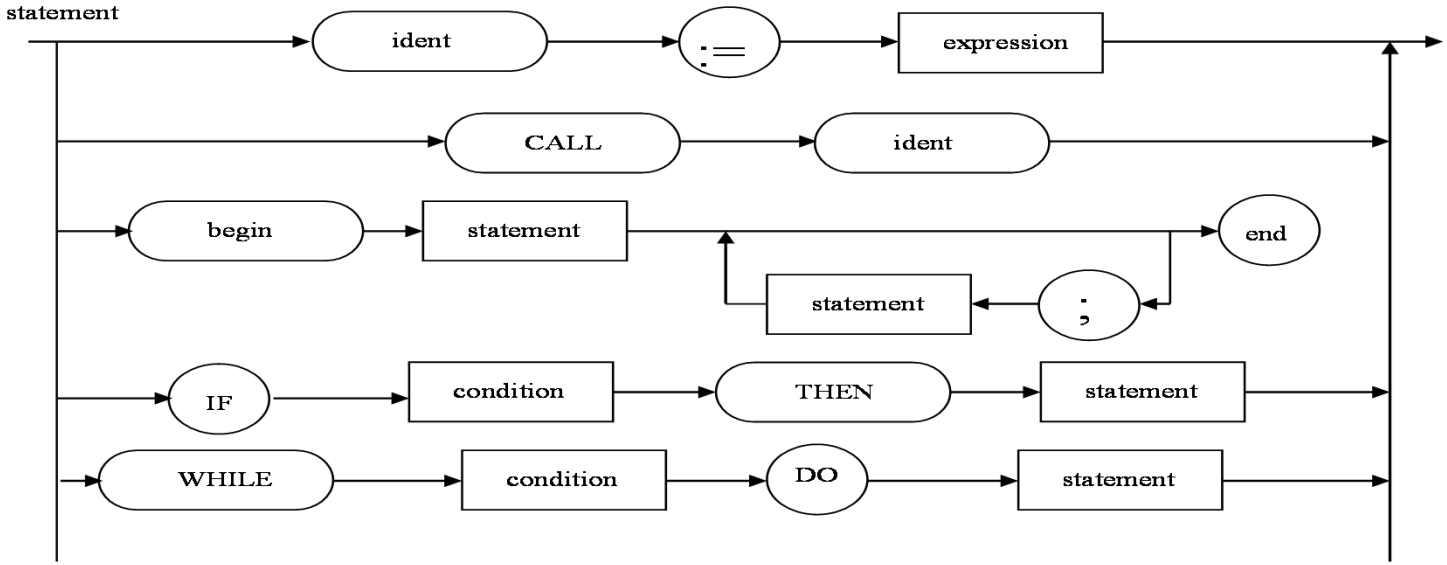
Ověřme na větě: if a then a = a + a

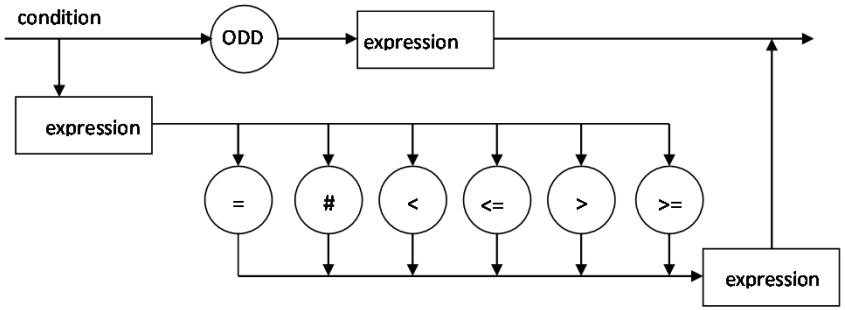
Zkuste i tuto: if a then if a then a = a else a = a

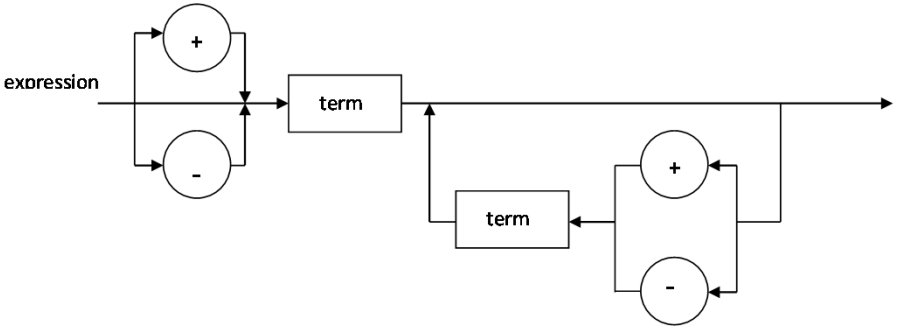
Syntax jazyka PL0
(syntaktické diagramy)

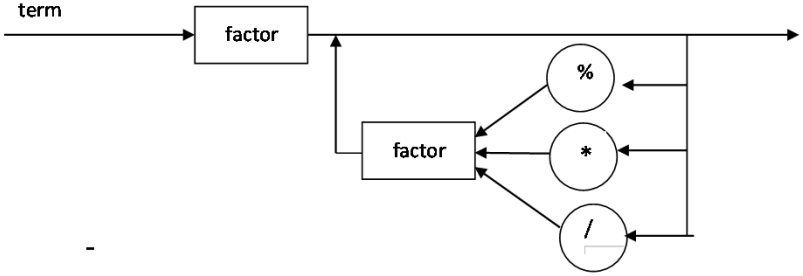
Dokument
ikace Microsoft W

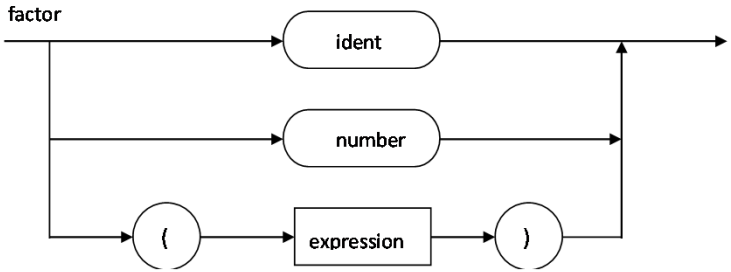

program
block
.
block
CONST
ident
=
number
,
;
VAR
ident
,
;
;
PROCEDURE
ident
;
block
statement


statement
ident
:=
expression
CALL
ident
begin
statement
end
statement
;
IF
condition
THEN
statement
WHILE
condition
DO
statement


condition
ODD
expression
expression
=
#
<
<=
>
>=
expression


expression
+
-
term
term
+
-


term
factor
%
factor
*
/


factor
ident
number
(
expression
)

Pokračování synt. Diagr.

část 2 syntax PL0

X	FIRST (X)	FOLLOW (X)
BLOCK	const , var , procedure ident , call , begin if , while , ε	. , ;
STATEMENT	ident , call , begin if , while , ε	. , ; , end
CONDITION	odd , + , − , ident number , (	then , do
EXPRESSION	(, ident , number , + , −	) . , ; . = #, <, <=, >, >= then , do , end
TERM	(, ident , number	) . , ; . = #, <, <=, >, >= then , do , end + , −
FACTOR	(, ident , number	) . , ; . = #, <, <=, >, >= then , do , end + , − , * , /, %

First a Follow pro PL0.

PL0 First Follow

Zpracování chyb v PL0

Panický způsob zotavování

používá triviální strategii:
vynechá text až do místa, kde se snadno vzpamatuje.
Snadno se vzpamatuje v místě s významným symbolem.

Předpoklady:

- **Každý typ příkazu začíná jiným symbolem.**
- **„ „ deklarace „ „ „ .**
- **Každá vyvolaná procedura se provede až do konce (žádný chybový výstup).**

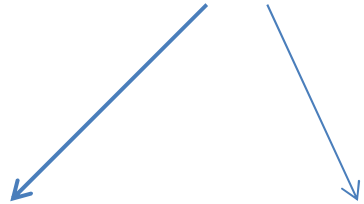
Zásady:

1. **Každá procedura má parametr – množinu následujících symbolů.**
2. **Při chybě je přeskočen vstupní text až k legálně následujícímu symbolu za prováděnou procedurou.**
3. **Na konci procedury je proveden Test, který ověří, že příští symbol patří do množiny následovníků.**
4. **Pro zmenšení vynechávaných úseků se do následovníků přidávají symboly ze začátku důležitých konstrukcí (tzv. STOP SYMBOLY).**

Zdůvodněte si to.

Činnost zajišťuje procedura Test

Test (má parametry s1,s2 typu_množina_symbolů a n pro označení chyby (celočíselné)



Následující symboly

Stop symboly

Procedura Test je využitelná i k ověření akceptovatelnosti symbolů na začátku procedur SA. Symboly s1 , s2 mají pak jiný význam

Počáteční

Následující

Princip:
Test zjišťuje, zda čtený symbol je v s1.
Pokud je, tak to je OK,
pokud není, tak přeskakuje až narazí na symbol ze sjednocení s1, s2.

1	pouzito "=" misto ":="
2	za "=" musi nasledovat cislo
3	za identifikatorem ma nasledovat "="
4	za "const", "var", "procedure" musi nasledovat identifikator
5	chybi strednik nebo carka
6	nespravny symbol po deklaraci procedury
7	je ocekavan prikaz
8	neocekavany symbol za prikazovou casti bloku
9	ocekavam tecku
10	nespravny symbol v prikazu
11	nedeklarovany identifikator
12	prirazeni konstante a procedure neni dovoleno
13	operator prirazeni je ":="
14	za "call" musi nasledovat identifikator
15	volani konstanty nebo promenne neni dovoleno
16	ocekavano "then"
17	ocekavano "}" nebo ";"
18	ocekavano "do"
19	nespravne pouzity symbol za prikazem
20	ocekavam relaci
21	jmeno procedury nelze pouzit ve vyrazu
22	chybi uzaviraci zavorka
23	faktor nesmi koncit timto symbolem
24	vyraz nesmi zacinat timto symbolem
30	prilis velke cislo

Dokument
ikace Microsoft W