AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Gramatyki atrybutywne, część 1 (gramatyki S-atrybutywne)

Teoria kompilacji

Dr inż. Janusz Majewski
Katedra Informatyki

Gramatyki atrybutywne

- Do przeprowadzenia poprawnego tłumaczenia, oprócz informacji na temat składni języka podlegającego tłumaczeniu, translator musi posiadać możliwość korzystania z wielu innych informacji dotyczących tłumaczonych konstrukcji. Informacje te mają charakter semantyczny. Dotyczą np. typu zmiennych, typu wyrażeń, wielkości pamięci dla zmiennych, adresów odpowiednich zapisów w tablicach, wartości wyrażeń w procesie interpretacji, itd.
- Informacje semantyczne mają charakter atrybutów stowarzyszonych z konstrukcjami języka podlegającego tłumaczeniu. Jednym ze sposobów formalnego określenia semantyki języka i zdefiniowania akcji podejmowanych w procesie translacji jest aparat gramatyk atrybutywnych.

Atrybutowane drzewa rozbioru

Analiza semantyczna i generowanie kodu oparte są na drzewie rozbioru syntaktycznego. Każdy wierzchołek drzewa (symbol gramatyki) jest „udekorowany” atrybutami opisującymi własności wierzchołka. Dla podkreślenia tego faktu takie drzewo nazywa się często „atrybutowanym drzewem rozbioru syntaktycznego”. Informację zgromadzoną w atrybutach wierzchołka wyprowadza się z jego otoczenia. Obliczenie atrybutów i sprawdzenie ich zgodności jest zadaniem analizy semantycznej. Optymalizację i generację kodu również można opisać w podobny sposób, używając atrybutów do sterowania przekształcaniem drzewa i w końcu do sterowania wyborem instrukcji maszynowych.

Wykorzystanie gramatyk atrybutywnych

Teoretyczny mechanizm gramatyk atrybutywnych może być wykorzystany:

- w analizie semantycznej do skontrolowania tych wszystkich aspektów kodu źródłowego, które nie zostały zbadane przez analizator leksykalny i syntaktyczny,
- przy wykonywaniu tłumaczenia do kodu pośredniego,
- przy obliczaniu pewnych wartości związanych z drzewem rozbioru (np. konstrukcja kalkulatora software’owego),
- do wykonywania pewnych akcji związanych z węzłami drzewa rozbioru (np. zapis lub odczyt z tablicy symboli, utworzenie lub wykorzystanie innych struktur danych, zapis lud odczyt z pliku, wydrukowanie czegoś, itp.); akcje te nie są „wyliczeniem” wartości atrybutów w sensie dosłownym.

Odwrotna notacja polska (notacja postfiksowa)

Przypomnienie:
notacja nawiasowa = notacja infiksowa
odwrotna notacja polska = notacja postfiksowa

Niech będą dane dwa zbiory:
OPERATOR – zbiór operatorów binarnych
OPERAND – zbiór pojedynczych operandów

1. Jeśli wyrażenie ***E*** w notacji infiksowej jest pojedynczym operandem ($E \in$ OPERAND), to postać postfiksowa tego wyrażenia to ***E***.
2. Jeśli $\boldsymbol{E_1 \Theta E_2}$ jest wyrażeniem w postaci infiksowej, E_1 i E_2 są wyrażeniami w notacji infiksowej, $\Theta \in$ OPERATOR, to:
   $\boldsymbol{E_1' E_2' \Theta}$ jest zapisem postfiksowym wyrażenia $E_1 \Theta E_2$
   przy czym: E_1' i E_2' – są odpowiednikami wyrażeń E_1 i E_2 w notacji postfiksowej
3. Jeśli ***(E)*** jest wyrażeniem infiksowym, to:
   E' jest zapisem wyrażenia *(E)* w notacji postfiksowej
   przy czym *E'* jest odpowiednikiem wyrażenia *E* w notacji infiksowej.

Przykłady

Notacja infiksowa: **a+b*c**
Notacja postfiksowa: **abc*+**

Notacja infiksowa: **a*b+c**
Notacja postfiksowa: **ab*c+**

Notacja infiksowa: **(a+b)*c**
Notacja postfiksowa: **ab+c***

Notacja infiksowa: **a*(b+c)**
Notacja postfiksowa: **abc+***

Przykład – translacja wyrażeń do odwrotnej notacji polskiej (ONP)

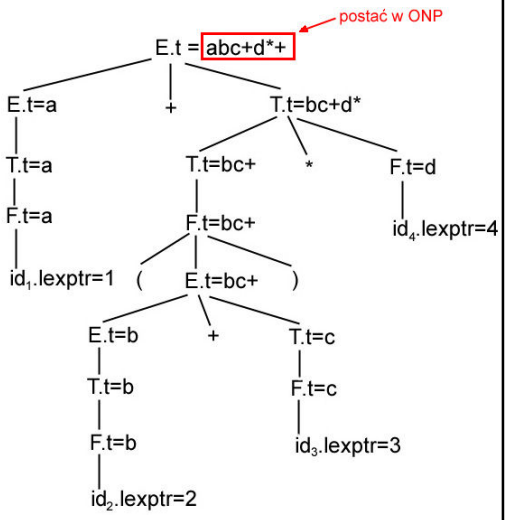
$E \rightarrow E_1 + T$	$E.t \leftarrow E_1.t \parallel T.t \parallel$ '+'
$E \rightarrow T$	$E.t \leftarrow T.t$
$T \rightarrow T_1 * F$	$T.t \leftarrow T_1.t \parallel F.t \parallel$ '*'
$T \rightarrow F$	$T.t \leftarrow F.t$
$F \rightarrow (E)$	$F.t \leftarrow E.t$
$F \rightarrow id$	$F.t \leftarrow name(id.lexptr)$

gdzie: $\parallel$ - operator konkatenacji tekstów

Rozważane słowo źródłowe: **a+(b+c)*d**

Po analizie leksykalnej: $id_1+(id_2+id_3)*id_4$

id.lexptr	name(id.lexptr)
1	a
2	b
3	c
4	d



Przykład akcji – translacja instrukcji przypisania do kodu trójadresowego

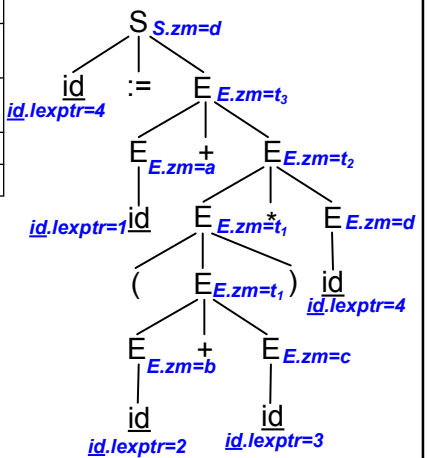
$S \rightarrow id := E$	$S.zm \leftarrow name(id.lexptr)$ $gen(S.zm \parallel ":=" \parallel E.zm)$
$E \rightarrow E_1 + E_2$	$E.zm \leftarrow new\_temp( )$ $gen(E.zm \parallel ":=" \parallel E_1.zm \parallel "+" \parallel E_2.zm$
$E \rightarrow E_1 * E_2$	$E.zm \leftarrow new\_temp( )$ $gen(E.zm \parallel ":=" \parallel E_1.zm \parallel "*" \parallel E_2.zm$
$E \rightarrow (E_1)$	$E.zm \leftarrow E_1.zm$
$E \rightarrow id$	$E.zm \leftarrow name(id.lexptr)$

gdzie: $\parallel$ - operator konkatenacji tekstów

Rozważane słowo źródłowe: **d:=a+(b+c)*d**

Po analizie leksykalnej: $id_4:=id_1+(id_2+id_3)*id_4$

id.lexptr	name(id.lexptr)
1	a
2	b
3	c
4	d



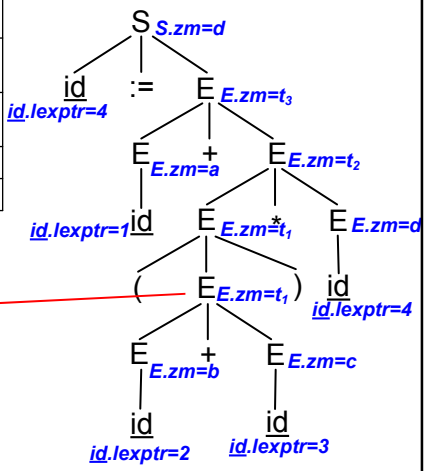
Przykład akcji – translacja instrukcji przypisania do kodu trójadresowego

$S \rightarrow \underline{id} := E$	***S.zm ← name(id.lexptr)*** ***gen(S.zm ‖ ":=" ‖ E.zm)***
$E \rightarrow E_1 + E_2$	**E.zm ← new_temp( )** ***gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "+" ‖ E_2.zm***
$E \rightarrow E_1 * E_2$	***E.zm ← new_temp( )*** ***gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "*" ‖ E_2.zm***
$E \rightarrow (E_1)$	***E.zm ← E_1.zm***
$E \rightarrow \underline{id}$	***E.zm ← name(id.lexptr)***

słowo źródłowe: **d:=a+(b+c)*d**

Tłumaczenie:

t_1 := b + c



Przykład akcji – translacja instrukcji przypisania do kodu trójadresowego

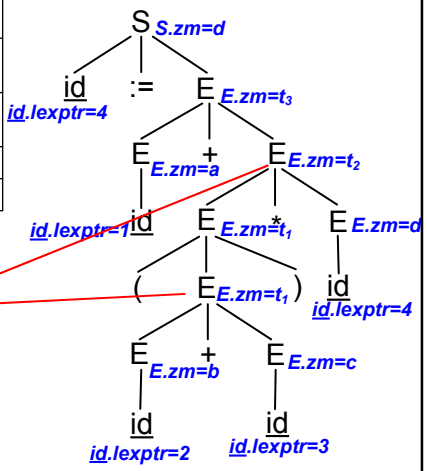
$S \rightarrow \underline{id} := E$	***S.zm ← name(id.lexptr)*** ***gen(S.zm ‖ ":=" ‖ E.zm)***
$E \rightarrow E_1 + E_2$	**E.zm ← new_temp( )** ***gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "+" ‖ E_2.zm***
$E \rightarrow E_1 * E_2$	***E.zm ← new_temp( )*** ***gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "*" ‖ E_2.zm***
$E \rightarrow (E_1)$	***E.zm ← E_1.zm***
$E \rightarrow \underline{id}$	***E.zm ← name(id.lexptr)***

słowo źródłowe: **d:=a+(b+c)*d**

Tłumaczenie:

t_1 := b + c

t_2 := t_1 * d



Przykład akcji – translacja instrukcji przypisania do kodu trójadresowego

$S \rightarrow \underline{id} := E$	***S.zm ← name(id.lexptr)*** ***gen(S.zm ‖ ":=" ‖ E.zm)***
$E \rightarrow E_1 + E_2$	**E.zm ← new_temp( )** ***gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "+" ‖ E_2.zm***
$E \rightarrow E_1 * E_2$	***E.zm ← new_temp( )*** ***gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "*" ‖ E_2.zm***
$E \rightarrow (E_1)$	***E.zm ← E_1.zm***
$E \rightarrow \underline{id}$	***E.zm ← name(id.lexptr)***

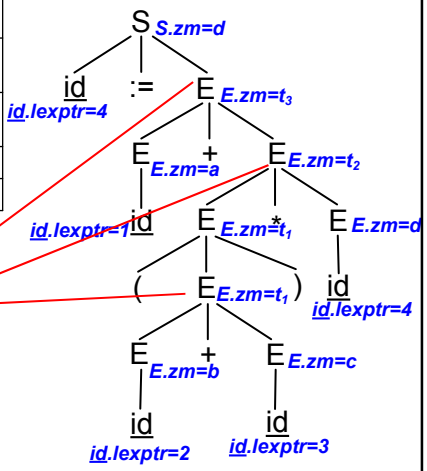
słowo źródłowe: **d:=a+(b+c)*d**

Tłumaczenie:

$t_1 := b + c$

$t_2 := t_1 * d$

$t_3 := a + t_2$



Przykład akcji – translacja instrukcji przypisania do kodu trójadresowego

$S \rightarrow \underline{id} := E$	***S.zm ← name(id.lexptr)*** ***gen(S.zm ‖ ":=" ‖ E.zm)***
$E \rightarrow E_1 + E_2$	**E.zm ← new_temp( )** ***gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "+" ‖ E_2.zm***
$E \rightarrow E_1 * E_2$	***E.zm ← new_temp( )*** ***gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "*" ‖ E_2.zm***
$E \rightarrow (E_1)$	***E.zm ← E_1.zm***
$E \rightarrow \underline{id}$	***E.zm ← name(id.lexptr)***

słowo źródłowe: **d:=a+(b+c)*d**

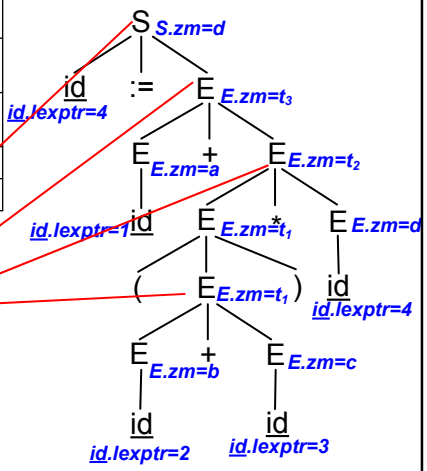
Tłumaczenie:

$t_1 := b + c$

$t_2 := t_1 * d$

$t_3 := a + t_2$

$d := t_3$



Przykłady reguł semantycznych

- *Deklaracje typu:*

$P \rightarrow D\ ;\ S$	
$D \rightarrow D\ ;\ D$	
$D \rightarrow \underline{id}\ :\ T$	$\{addtype(\underline{id}.entry,\ T.type)\}$
$T \rightarrow \underline{char}$	$\{T.type \leftarrow char\}$
$T \rightarrow \underline{integer}$	$\{T.type \leftarrow integer\}$
$T \rightarrow \underline{boolean}$	$\{T.type \leftarrow boolean\}$
$T \rightarrow \uparrow T_1$	$\{T.type \leftarrow pointer(T_1.type)\}$
$T \rightarrow \underline{array}\ [\underline{num}]\ of\ T_1$	$\{T.type \leftarrow array(1..\underline{num}.val,\ T_1.type)\}$

Przykłady reguł semantycznych

- *Ustalanie typu wyrażeń:*

$E \rightarrow \underline{num}$	$\{E.type \leftarrow integer\}$
$E \rightarrow \underline{num}\ .\ \underline{num}$	$\{E.type \leftarrow real\}$
$E \rightarrow \underline{id}$	$\{E.type \leftarrow lookup(\underline{id}.lexptr)\}$
$E \rightarrow E_1\ \underline{op}\ E_2$	$\{E.type \leftarrow$ if $E_1.type = integer$ and $E_2.type = integer$ then *integer* else if $E_1.type = integer$ and $E_2.type = real$ then *real* else if $E_1.type = real$ and $E_2.type = integer$ then *real* else if $E_1.type = real$ and $E_2.type = real$ then *real* else *type_error*$\}$

Przykłady reguł semantycznych

- *Kontrola typu instrukcji:*

$S \rightarrow \underline{id} := E$	*{S.type ←* if *lookup(id.entry) = E.type* then *void* else *type_error}*
$S \rightarrow \underline{if}\ E\ \underline{then}\ S_1$	*{S.type ←* if *E.type = boolean* then *S_1.type* else *type_error}*
$S \rightarrow \underline{while}\ E\ \underline{do}\ S_1$	*{S.type ←* if *E.type = boolean* then *S_1.type* else *type_error}*
$S \rightarrow S_1 ; S_2$	*{S.type ←* if *S_1.type = void* and *S_2.type = void* then *void* else *type_error}*

Gramatyka atrybutywna (1)

- Gramatyka atrybutywna jest oparta na gramatyce bezkontekstowej $G = \langle V, \Sigma, P, S \rangle$.
- Łączy ona z każdym symbolem gramatyki $X \in (V \cup \Sigma)$ zbiór atrybutów $A(X)$.
- Każdy atrybut $X.a \in A(X)$ reprezentuje specyficzną własność symbolu X i może przyjąć którąkolwiek wartość z pewnego zbioru wartości.
- $X.a$ oznacza atrybut „a” będący elementem zbioru $A(X)$. Każdy wierzchołek w drzewie rozbioru syntaktycznego słowa z języka $L(G)$ jest związany ze zbiorem wartości atrybutów dla tego symbolu X, który stanowi etykietę tego wierzchołka. Wartości atrybutów ustala się za pomocą reguł semantycznych.

Gramatyka atrybutywna (2)

Zbiór $R(p)$:

$$R(p) \stackrel{df}{=} \left\{ X_i.a \leftarrow f\left(X_j.b, \ldots, X_k.c\right) \right\}$$

jest zbiorem reguł semantycznych dla produkcji $p \in P$ takiej, że:

$p = \left(X_0 \rightarrow X_1 \ldots X_n\right)$

$0 \le i \le n$

$0 \le j \le n$

$0 \le k \le n$

Reguła semantyczna $X_i.a \leftarrow f\left(X_j.b, \ldots, X_k.c\right)$ definiuje atrybut $X_i.a$ za pomocą atrybutów $X_j.b, \ldots, X_k.c$ symboli z tej samej produkcji.

Gramatyka atrybutywna (3)

Gramatyka atrybutywna AG jest trójką:

$AG = \langle G, A, R \rangle$

gdzie:

$G = \langle V, \Sigma, P, S \rangle \in \mathcal{G}_{BK}$ – gramatyka bezkontekstowa

$A = \bigcup_{X \in (V \cup \Sigma)} A(X)$ – skończony zbiór atrybutów

$R = \bigcup_{p \in P} R(p)$ – skończony zbiór reguł semantycznych

Jeśli $A(X) \cap A(Y) \neq \varnothing$ to $X = Y$

Dla każdego wystąpienia X w drzewie rozbioru syntaktycznego słowa z języka $L(G)$, do obliczenia każdego atrybutu $X.a \in A(X)$ można zastosować co najwyżej jedną regułę semantyczną.

Gramatyka atrybutywna (4)

Dla każdej produkcji $p = (X_0 \to X_1 \ldots X_n) \in P$ zbiorem definiujących wystąpień atrybutów jest:

$$AF(p) = \{X_i.a : X_i.a \leftarrow f(\ldots) \in R(p)\}$$

Atrybut $X.a$ nazywamy syntetyzowanym $\overset{df}{\Leftrightarrow}$

$$\exists p = (X \to \chi) \in P : X.a \in AF(p)$$

Atrybut $X.a$ nazywamy dziedziczonym $\overset{df}{\Leftrightarrow}$

$$\exists p = (Y \to \mu X \nu) \in P : X.a \in AF(p)$$

Gramatyka atrybutywna (5)

$\forall X \in (V \cup \Sigma)$ zachodzi $AS(X) \cap AI(X) = \varnothing$

gdzie:

$$AS(X) \overset{df}{=} \{X.a : \exists p = (X \to \chi) \in P \wedge X.a \in AF(p)\}$$

$AS(X)$ – zbiór atrybutów syntetyzowanych dla $X \in (V \cup \Sigma)$

$$AI(X) \overset{df}{=} \{X.a : \exists q = (Y \to \mu X \nu) \in P \wedge X.a \in AF(q)\}$$

$AI(X)$ – zbiór atrybutów dziedziczonych dla $X \in (V \cup \Sigma)$

Ponadto istnieje co najwyżej jedna reguła $X.a \leftarrow f(\ldots)$ w zbiorze $R(p)$ dla każdego $p \in P$ i $X.a \in A(X)$.

Gramatyka atrybutywna (6)

Definicja:
Gramatyka atrybutywna jest zupełna, jeśli dla wszystkich symboli $X \in (V \cup \Sigma)$ są spełnione następujące warunki:

$\forall p = (X \to \chi) \in P : AS(X) \subseteq AF(p)$

$\forall q = (Y \to \mu X \nu) \in P : AI(X) \subseteq AF(q)$

$AS(X) \cup AI(X) = A(X)$

Ponadto: $AI(S) = \varnothing$

Definicja
Gramatyka atrybutywna jest dobrze zdefiniowana jeśli dla każdego drzewa rozbioru syntaktycznego słowa z $L(G)$ wszystkie atrybuty są efektywnie obliczalne.

Gramatyki S-atrybutywne

Gramatyką atrybutywną klasy S (gramatyką S-atrybutywną) nazywamy gramatykę atrybutywną zawierającą tylko i wyłącznie atrybuty syntetyzowane.

Mogą one być obliczone z wykorzystaniem drzewa rozbioru syntaktycznego metodą bottom-up ($\to$ por. poprzedni przykład z translacją wyrażeń do ONP; $\to$ por. następny przykład kalkulatora).

Przykład – kalkulator

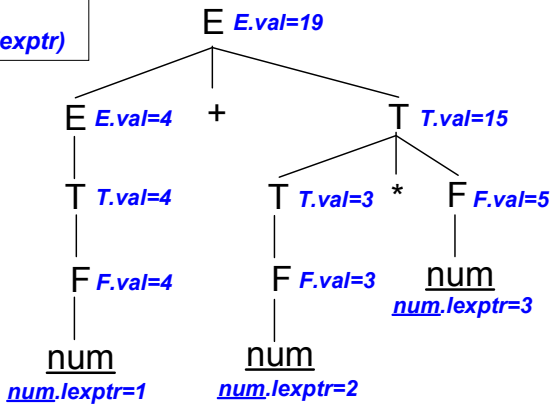
$E \rightarrow E_1 + T$	$E.val \leftarrow E_1.val + T.val$
$E \rightarrow T$	$E.val \leftarrow T.val$
$T \rightarrow T_1 * F$	$T.val \leftarrow T_1.val * F.val$
$T \rightarrow F$	$T.val \leftarrow F.val$
$F \rightarrow (E)$	$F.val \leftarrow E.val$
$F \rightarrow num$	$F.val \leftarrow value(num.lexptr)$

Słowo źródłowe: **4+3*5**

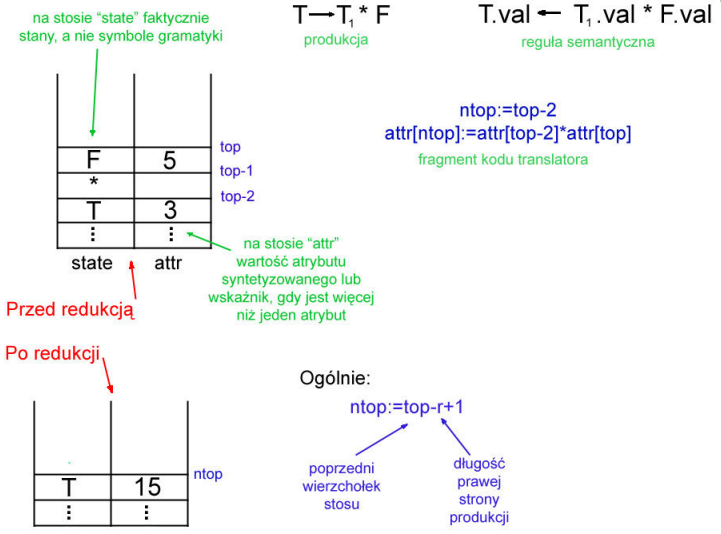
Po skanerze: $num_1 + num_2 * num_3$

Tablica symboli:

num.lexptr	value(num.lexptr)
1	4
2	3
3	5



Translacja „bottom-up” dla gramatyk S-atrybutywnych



Przykład – kalkulator c.d.

$L \rightarrow E\$$	$L.b \leftarrow print(E.val)$	$print(attr[top-1])$
$E \rightarrow E_1 + T$	$E.val \leftarrow E_1.val + T.val$	$attr[ntop] := attr[top-2] + attr[top]$
$E \rightarrow T$	$E.val \leftarrow T.val$	
$T \rightarrow T_1 * F$	$T.val \leftarrow T_1.val * F.val$	$attr[ntop] := attr[top-2] * attr[top]$
$T \rightarrow F$	$T.val \leftarrow F.val$	
$F \rightarrow (E)$	$F.val \leftarrow E.val$	$attr[ntop] := attr[top-1]$
$F \rightarrow \underline{num}$	$F.val \leftarrow value(\underline{num}.lexptr)$	$attr[ntop] := value(attr[top])$

Założenie: po *shift* $\underline{num}$ na stosie *attr* umieszczana jest wartość $\underline{num}.lexptr$

Słowo wejściowe: $\underline{num}_1 * \underline{num}_2 + \underline{num}_3\$$ (3*5+4)

$\underline{num}.lexptr$	$value(\underline{num}.lexptr)$
1	3
2	5
3	4

Przykład – kalkulator c.d.

Słowo wejściowe: $\underline{num}_1 * \underline{num}_2 + \underline{num}_3\$$ (3*5+4)

wejście	state	attr	Produkcja
num*num+num$			
*num+num$	num	1	
*num+num$	F	3	$F \rightarrow \underline{num}$
*num+num$	T	3	$T \rightarrow F$
num+num$	T*	3□	
+num$	T*num	3□2	
+num$	T*F	3□5	$F \rightarrow \underline{num}$
+num$	T	15	$T \rightarrow T * F$
+num$	E	15	$E \rightarrow T$
num$	E+	15□	
$	E+num	15□3	
$	E+F	15□4	$F \rightarrow \underline{num}$
$	E+T	15□4	$T \rightarrow F$
$	E	19	$E \rightarrow E + T$
	E$	19□	
	L	(*)	

(*) – wydrukowanie wartości =19