AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Gramatyki atrybutywne

Definicje kierowane składnią

Dr inż. Janusz Majewski
Języki formalne i automaty

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 wyborem instrukcji maszynowych.

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)

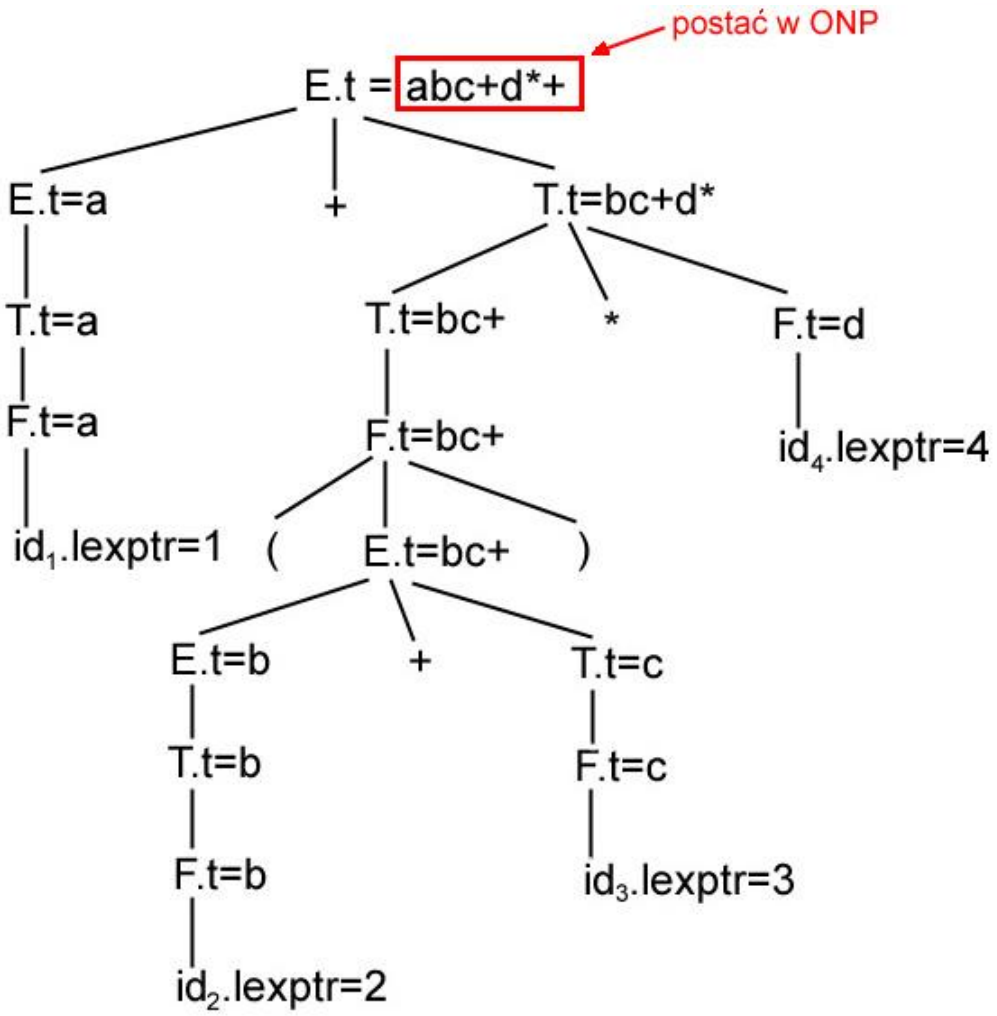
$E \rightarrow E_1 + T$	$E.t \leftarrow E_1.t \parallel T.t \parallel$ '+'
$E \rightarrow T$	$E.t \leftarrow T.t$
$T \rightarrow T_1 * F$	$T.t \leftarrow T_1.t \parallel F.t \parallel$ '*'
$T \rightarrow F$	$T.t \leftarrow F.t$
$F \rightarrow (E)$	$F.t \leftarrow E.t$
$F \rightarrow \underline{id}$	$F.t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

gdzie: $\parallel$ - operator konkatenacji tekstów

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

Po analizie leksykalnej: $\underline{id}_1+(\underline{id}_2+\underline{id}_3)*\underline{id}_4$

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



Przykład – translacja instrukcji przypisania do kodu trójadresowego

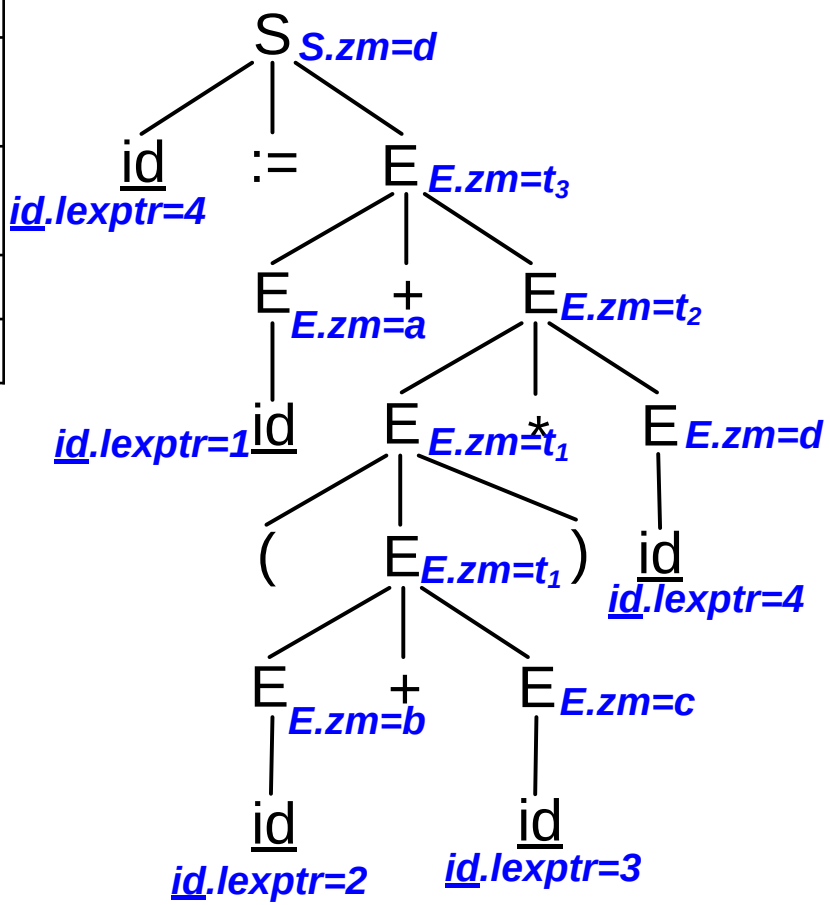
$S \rightarrow \underline{id} := E$	***S.zm ← name(id.lexptr)*** ***gen(S.zm ‖ ":=" ‖ E.zm)***
$E \rightarrow E_1 + E_2$	***E.zm ← new_temp( )*** ***gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "+" ‖ E_2.zm***
$E \rightarrow E_1 * E_2$	***E.zm ← new_temp( )*** ***gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "*" ‖ E_2.zm***
$E \rightarrow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

gdzie: ‖ - operator konkatenacji tekstów

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

Po analizie leksykalnej: **$\underline{id}_4$:=$\underline{id}_1$+($\underline{id}_2$+$\underline{id}_3$)*$\underline{id}_4$**

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



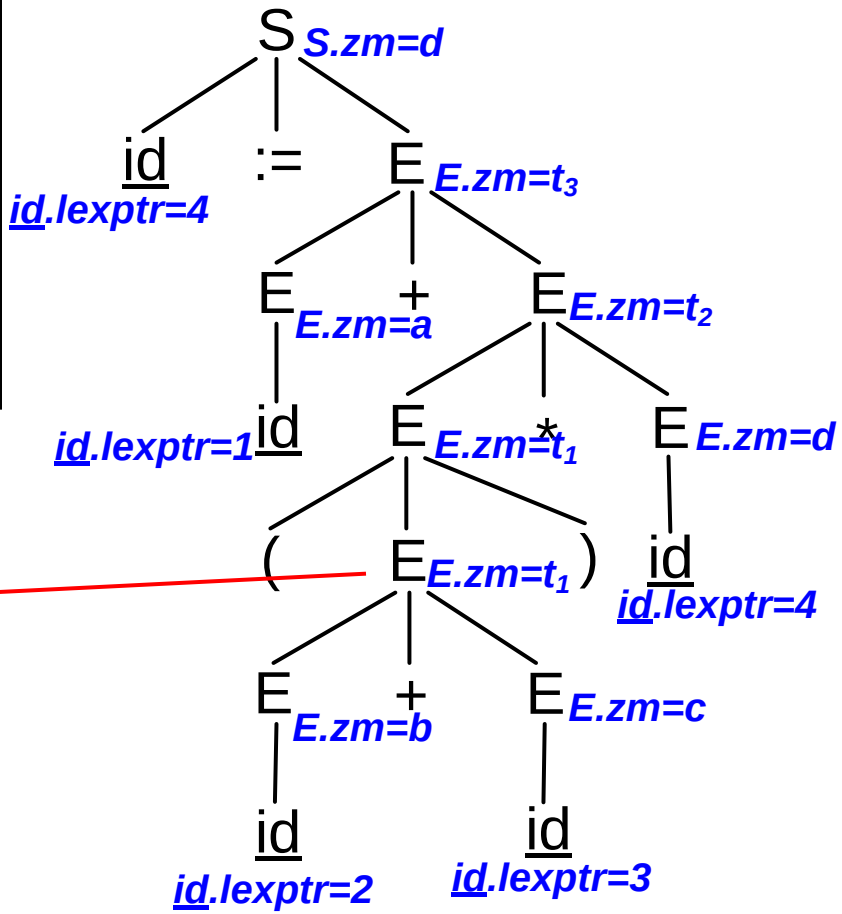
Przykład – translacja instrukcji przypisania do kodu trójadresowego

$S \rightarrow \underline{id} := E$	**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 – translacja instrukcji przypisania do kodu trójadresowego

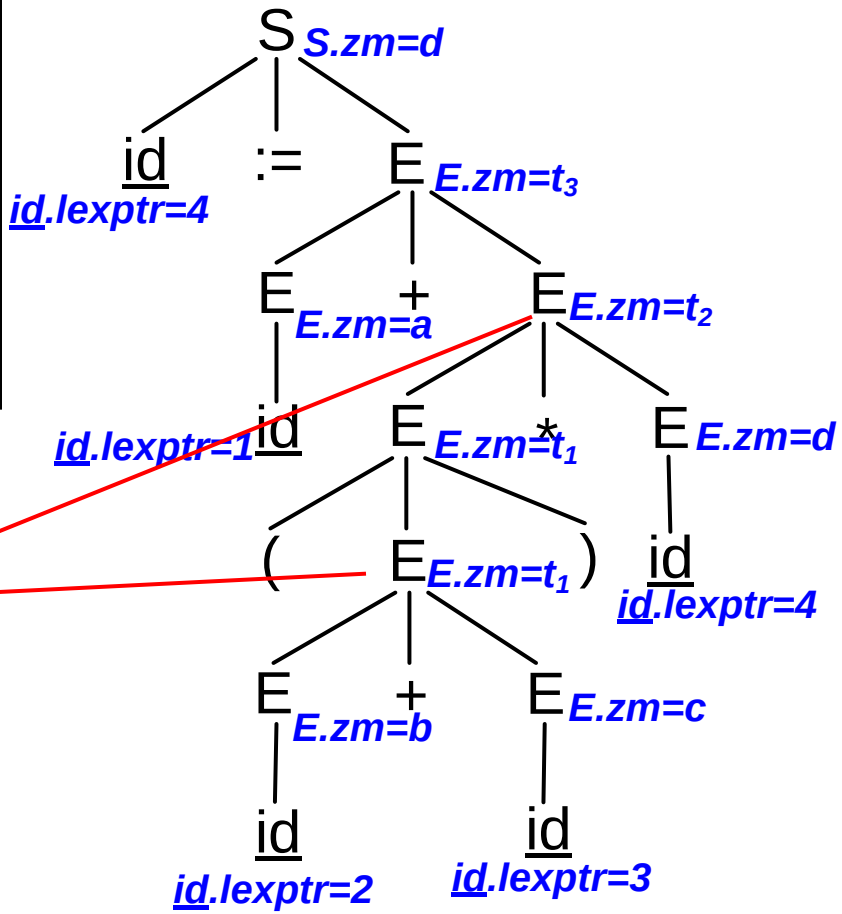
$S \rightarrow \underline{id} := E$	**S.zm ← name(id.lexptr)** **gen(S.zm ‖ ":=" ‖ E.zm)**
$E \rightarrow E_1 + E_2$	**E.zm ← new_temp( )** **gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "+" ‖ E_2.zm**
$E \rightarrow E_1 * E_2$	**E.zm ← new_temp( )** **gen(E.zm ‖ ":=" ‖ E_1.zm ‖ "*" ‖ E_2.zm**
$E \rightarrow (E_1)$	**E.zm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

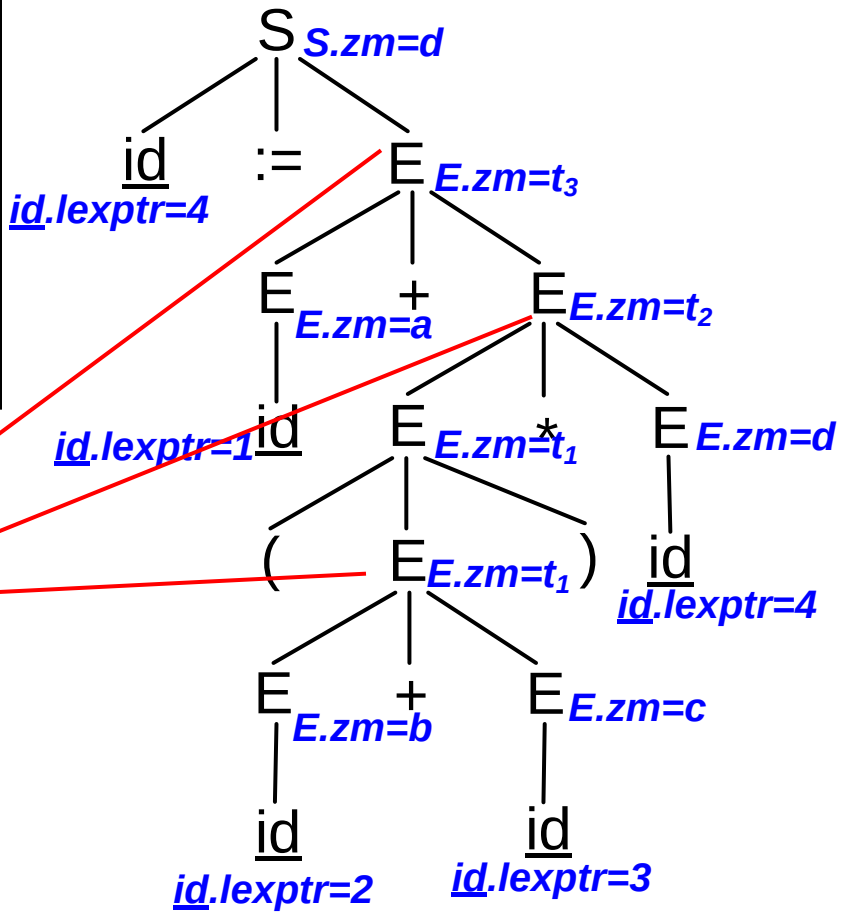
słowo źródłowe: **d:=a+(b+c)*d**

Tłumaczenie:

$t_1 := b + c$

$t_2 := t_1 * d$

$t_3 := a + t_2$



Przykład – 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**

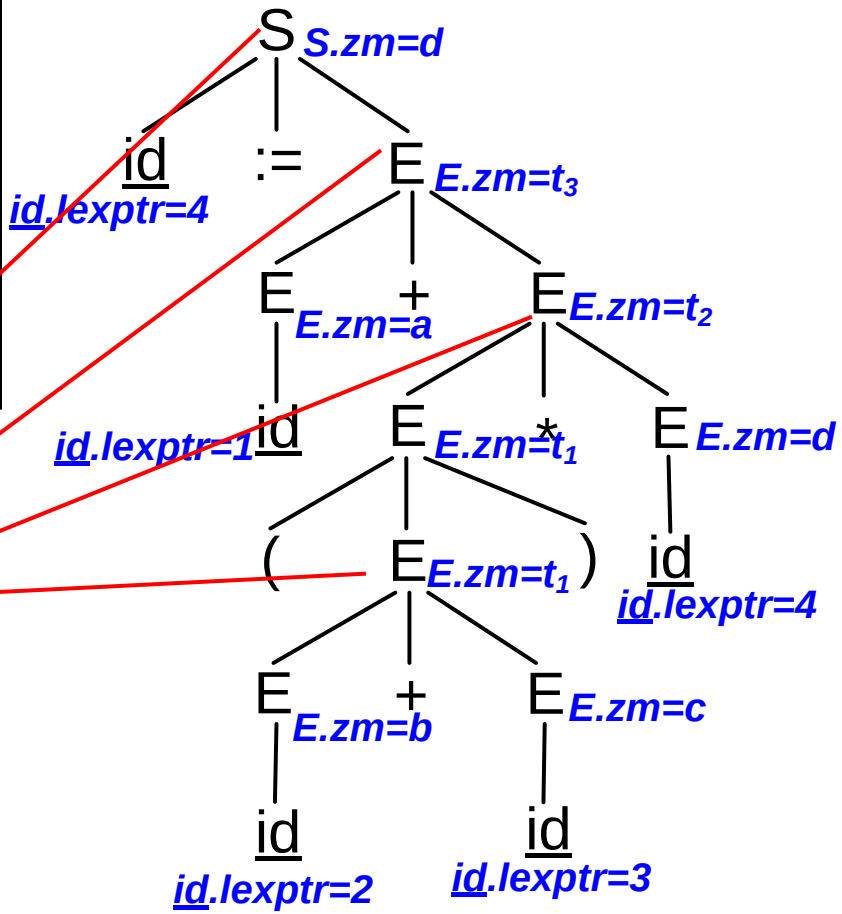
Tłumaczenie:

$t_1 := b + c$

$t_2 := t_1 * d$

$t_3 := a + t_2$

$d := t_3$



Przykład – kalkulator

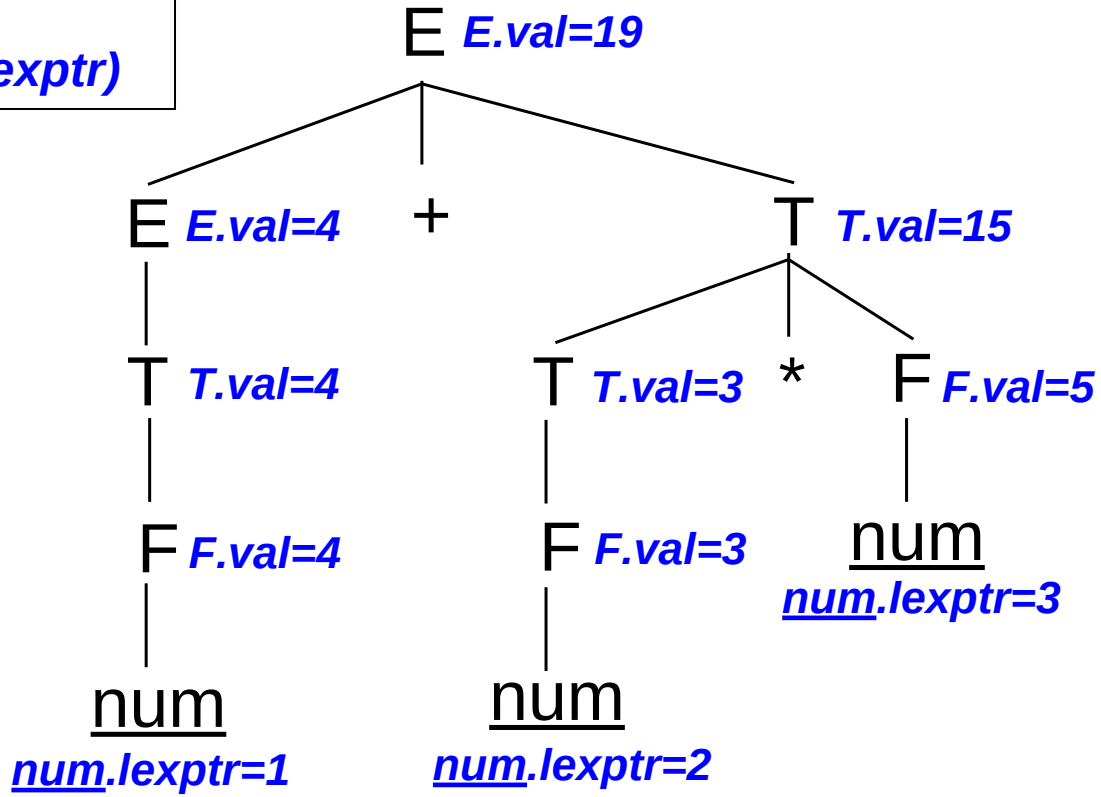
$E \rightarrow E_1 + T$	$E.val \leftarrow E_1.val + T.val$
$E \rightarrow T$	$E.val \leftarrow T.val$
$T \rightarrow T_1 * F$	$T.val \leftarrow T_1.val * F.val$
$T \rightarrow F$	$T.val \leftarrow F.val$
$F \rightarrow (E)$	$F.val \leftarrow E.val$
$F \rightarrow \underline{num}$	$F.val \leftarrow value(\underline{num}.lexptr)$

Słowo źródłowe: **4+3*5**

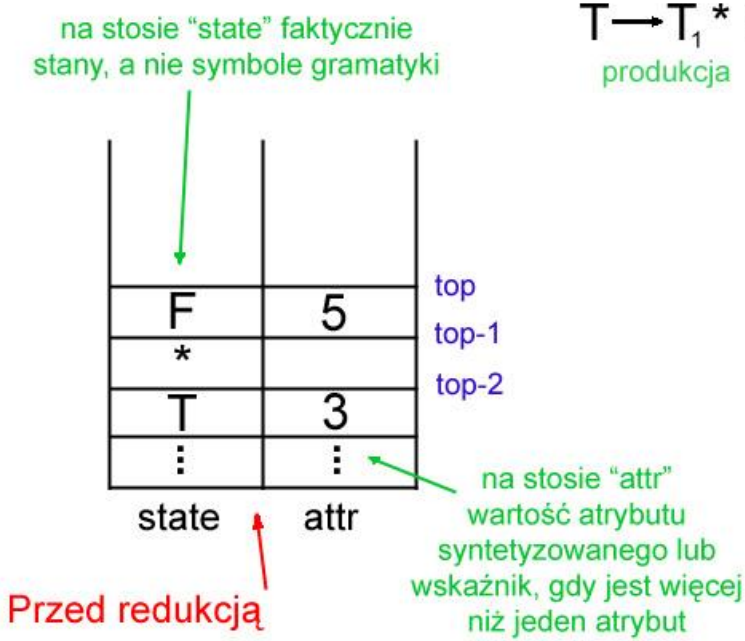
Po skanerze: $\underline{num}_1 + \underline{num}_2 * \underline{num}_3$

Tablica symboli:

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



Obliczanie atrybutów równocześnie z parsingiem „bottom-up”



$T \rightarrow T_1 * F$

produkcja

$T.val \leftarrow T_1.val * F.val$

reguła semantyczna

```
ntop:=top-2
attr[ntop]:=attr[top-2]*attr[top]
```

fragment kodu translatora

Ogólnie:

```
ntop:=top-r+1
```

poprzedni wierzchołek stosu

długość prawej strony produkcji

Gramatyki L-atrybutywne

Gramatyki L-atrybutywne bazujące na gramatykach LL(1) umożliwiają obliczanie atrybutów równolegle z parsingiem top-down.

Przekształcanie gramatyk S-atrybutywnych w gramatyki L-atrybutywne bez lewej rekursji i przedstawianie ich w postaci schematów tłumaczenia.

We: gramatyka S-atrybutywna z lewą rekursją, np.:

$A \to A_1 Y \qquad \{A.a \leftarrow g(A_1.a, Y.y)\}$

$A \to X \qquad \{A.a \leftarrow f(X.x)\}$

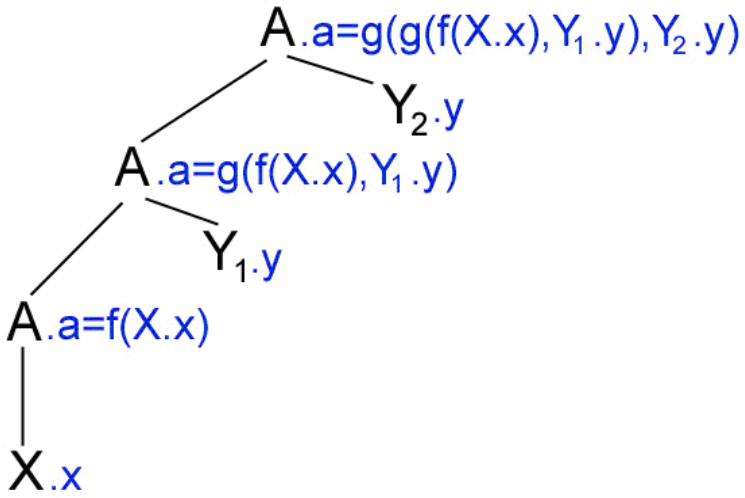
Wy: gramatyka L-atrybutywna bez lewej rekursji w postaci schematu tłumaczenia:

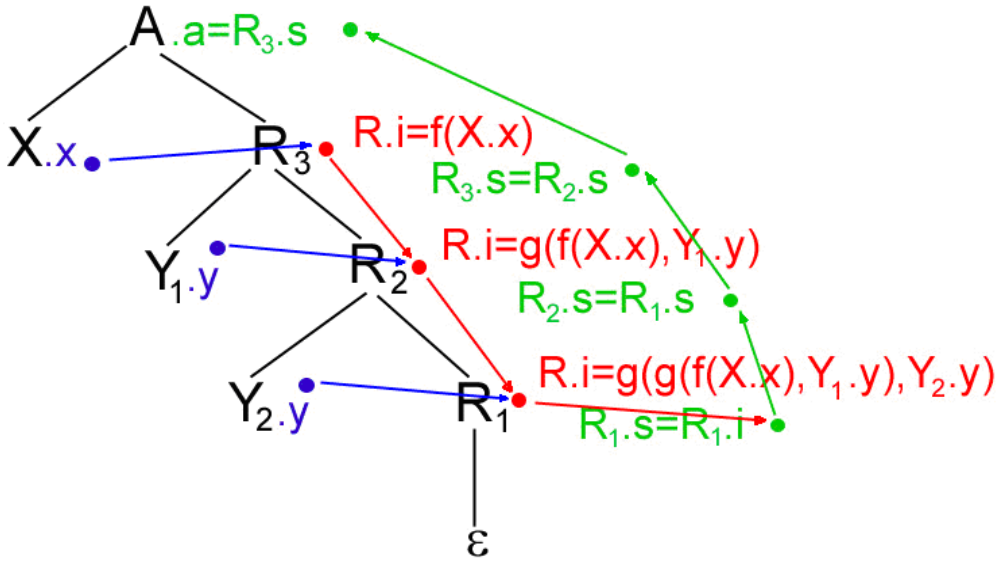
$A \to X \ \{R.i \leftarrow f(X.x)\}\ R\ \{A.a \leftarrow R.s\}$

$R \to Y \ \{R_1.i \leftarrow g(R.i, Y.y)\}\ R_1\ \{R.s \leftarrow R_1.s\}$

$R \to \varepsilon \ \{R.s \leftarrow R.i\}$

Gramatyki L-atrybutywne





We: gramatyka S-atrybutywna z lewą rekursją, np.:

$A \to A_1 Y \qquad \{A.a \leftarrow g(A_1.a, Y.y)\}$

$A \to X \qquad \{A.a \leftarrow f(X.x)\}$

Wy: gramatyka L-atrybutywna bez lewej rekursji w postaci schematu tłumaczenia:

$A \to X \ \{R.i \leftarrow f(X.x)\}\ R\ \{A.a \leftarrow R.s\}$

$R \to Y \ \{R_1.i \leftarrow g(R.i, Y.y)\}\ R_1\ \{R.s \leftarrow R_1.s\}$

$R \to \varepsilon \ \{R.s \leftarrow R.i\}$

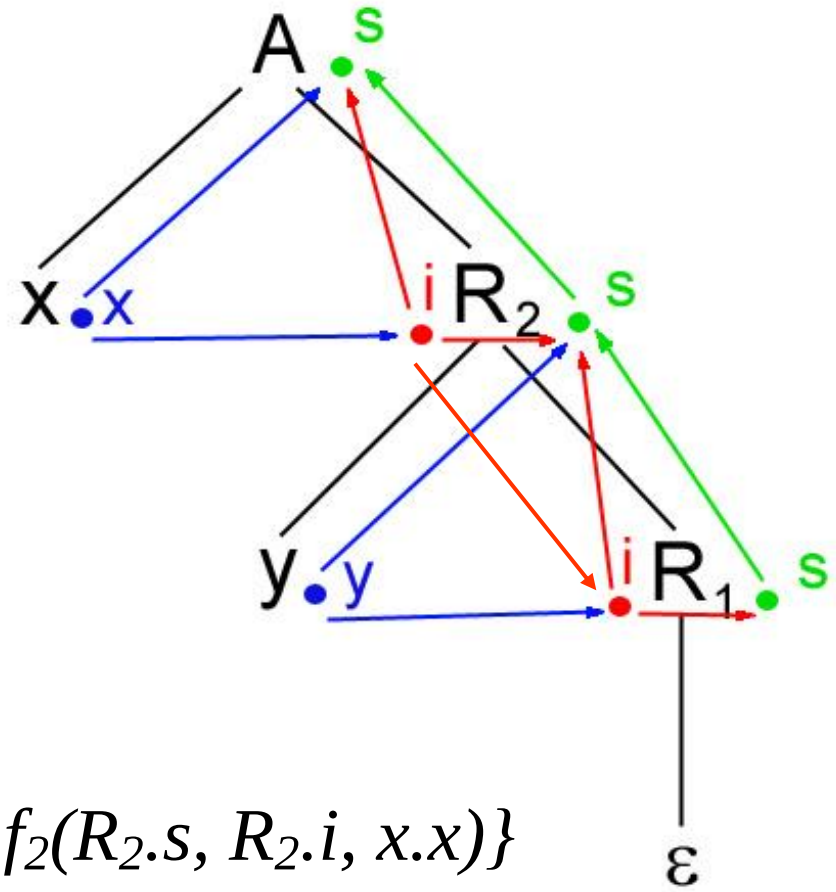
Translacja bottom-up dla gramatyk L-atrybutywnych

Metoda poniższa może być stosowana do tych wszystkich gramatyk, co poprzednio (tzn. dla L-atrybutywnych definicji opartych na gramatykach LL(1)), a także dla wielu (choć nie wszystkich) gramatyk L-atrybutywnych opartych na gramatykach LR(1).

Na przykładzie zostanie przedstawione likwidowanie „akcji wewnętrznych” w schematach tłumaczenia poprzez wprowadzenie dodatkowych nieterminali.

Przykład – tworzenie nowych nieterminali dla atrybutów dziedziczonych

$$A \rightarrow xR$$
$$R \rightarrow yR$$
$$R \rightarrow \varepsilon$$



$A \rightarrow x\ \{R_2.i \leftarrow f_1(x.x)\}\ R_2\ \{A.s \leftarrow f_2(R_2.s, R_2.i, x.x)\}$

$R_2 \rightarrow y\ \{R_1.i \leftarrow f_3(y.y, R_2.i)\}\ R_1\ \{R_2.s \leftarrow f_4(R_1.s, R_1.i, R_2.i, y.y)\}$

$R_1 \rightarrow \varepsilon\ \{R_1.s \leftarrow f_5(R_1.i)\}$

Przykład – tworzenie nowych nieterminali dla atrybutów dziedziczonych

Gramatyka:

$A \rightarrow x$ **$\{R_2.i \leftarrow f_1(x.x)\}$** R_2 $\{A.s \leftarrow f_2(R_2.s, R_2.i, x.x)\}$

$R_2 \rightarrow y$ **$\{R_1.i \leftarrow f_3(y.y, R_2.i)\}$** R_1 $\{R_2.s \leftarrow f_4(R_1.s, R_1.i, R_2.i, y.y)\}$

$R_1 \rightarrow \varepsilon$ $\{R_1.s \leftarrow f_5(R_1.i)\}$

jest transformowana do postaci:

(1) $A \rightarrow x\, M\, R_2 \{A.s \leftarrow f_2(R_2.s, R_2.i, x.x)\}$

(2) $R_2 \rightarrow y\, N\, R_1 \{R_2.s \leftarrow f_4(R_1.s, R_1.i, R_2.i, y.y)\}$

(3) $R_1 \rightarrow \varepsilon \{R_1.s \leftarrow f_5(R_1.i)\}$

(4) $M \rightarrow \varepsilon \{R_2.i \leftarrow f_1(x.x)\}$

(5) $N \rightarrow \varepsilon \{R_1.i \leftarrow f_3(y.y, R_2.i)\}$

Przykład – tworzenie nowych nieterminali dla atrybutów dziedziczonych

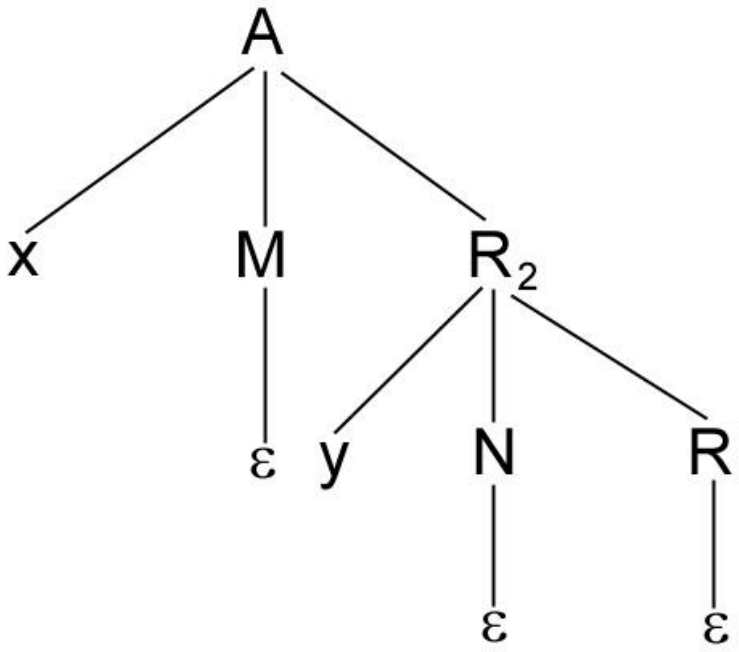
(1) $A \to x\,M\,R_2 \left\{ A.s \leftarrow f_2\left(R_2.s, R_2.i, x.x\right) \right\}$

(2) $R_2 \to y\,N\,R_1 \left\{ R_2.s \leftarrow f_4\left(R_1.s, R_1.i, R_2.i, y.y\right) \right\}$

(3) $R_1 \to \varepsilon \left\{ R_1.s \leftarrow f_5\left(R_1.i\right) \right\}$

(4) $M \to \varepsilon \left\{ R_2.i \leftarrow f_1\left(x.x\right) \right\}$

(5) $N \to \varepsilon \left\{ R_1.i \leftarrow f_3\left(y.y, R_2.i\right) \right\}$

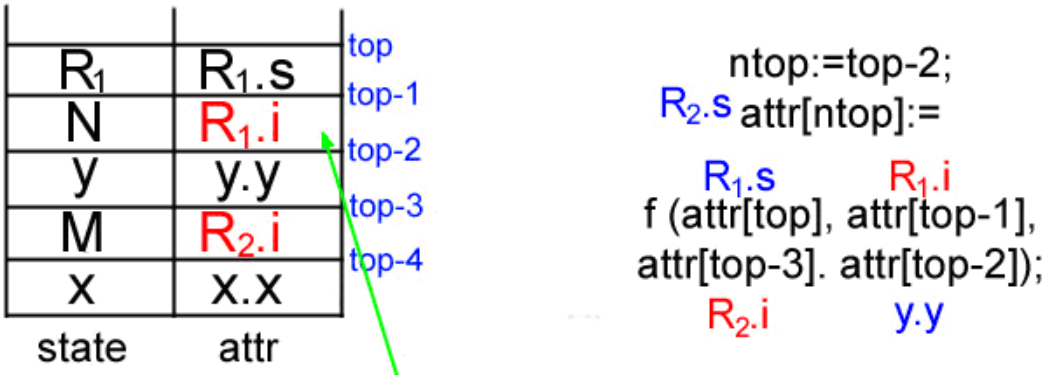


state	attr	we	
ε	ε	xy	shift
x	$x.x$	y	red 4
$x\,M$	$x.x\ \ R_2.i$	y	shift
$x\,M\,y$	$x.x\ \ R_2.i\ \ y.y$	ε	red 5
$x\,M\,y\,N$	$x.x\ \ R_2.i\ \ y.y\ \ R_1.i$	ε	red 3
$x\,M\,y\,N\,R_1$	$x.x\ \ R_2.i\ \ y.y\ \ R_1.i\ \ \ R_1.s$	ε	red 2
$x\,M\,R_2$	$x.x\ \ R_2.i\ \ \ R_2.s$	ε	red 1
A	$A.s$	ε	acc

Przykład – tworzenie nowych nieterminali dla atrybutów dziedziczonych

$$(2) \quad R_2 \rightarrow y\, N\, R_1 \left\{ R_2.s \leftarrow f_4\left(R_1.s, R_1.i, R_2.i, y.y\right) \right\}$$

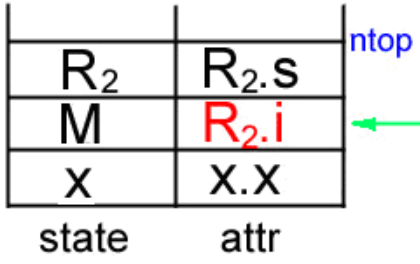
Przed redukcją



ntop:=top-2;
$R_2.s$ attr[ntop]:=

$R_1.s$ $R_1.i$
f (attr[top], attr[top-1],
attr[top-3]. attr[top-2]);
$R_2.i$ y.y

Po redukcji



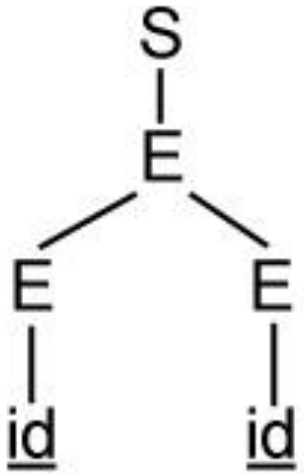
atrybut dziedziczony zawsze bezpośrednio pod atrybutem syntetyzowanym symbolu R, o ile symbol R znajduje się na stosie

Przykład – „dekorowanie” drzewa rozbioru (1)

$S \to E$	$E.i \leftarrow g(E.s)$ $S.r \leftarrow E.t$
$E \to E_1 E_2$	$E.s \leftarrow f_s(E_1.s, E_2.s)$ $E_1.i \leftarrow f_{i1}(E.i)$ $E_2.i \leftarrow f_{i2}(E.i)$ $E.t \leftarrow f_t(E_1.t, E_2.t)$
$E \to \underline{id}$	$E.s \leftarrow \underline{id}.s$ $E.t \leftarrow h(E.i)$

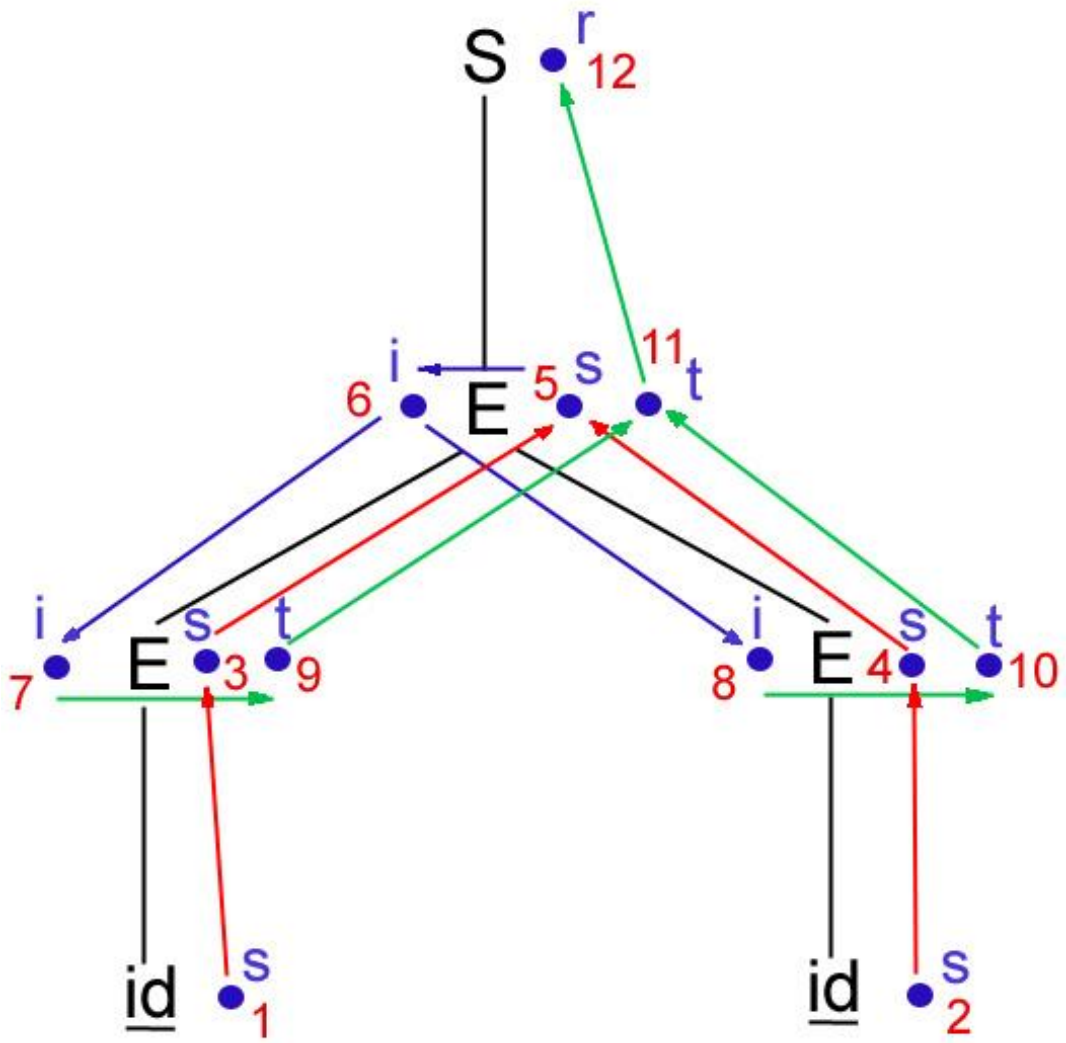
Analizowane słowo:

id id

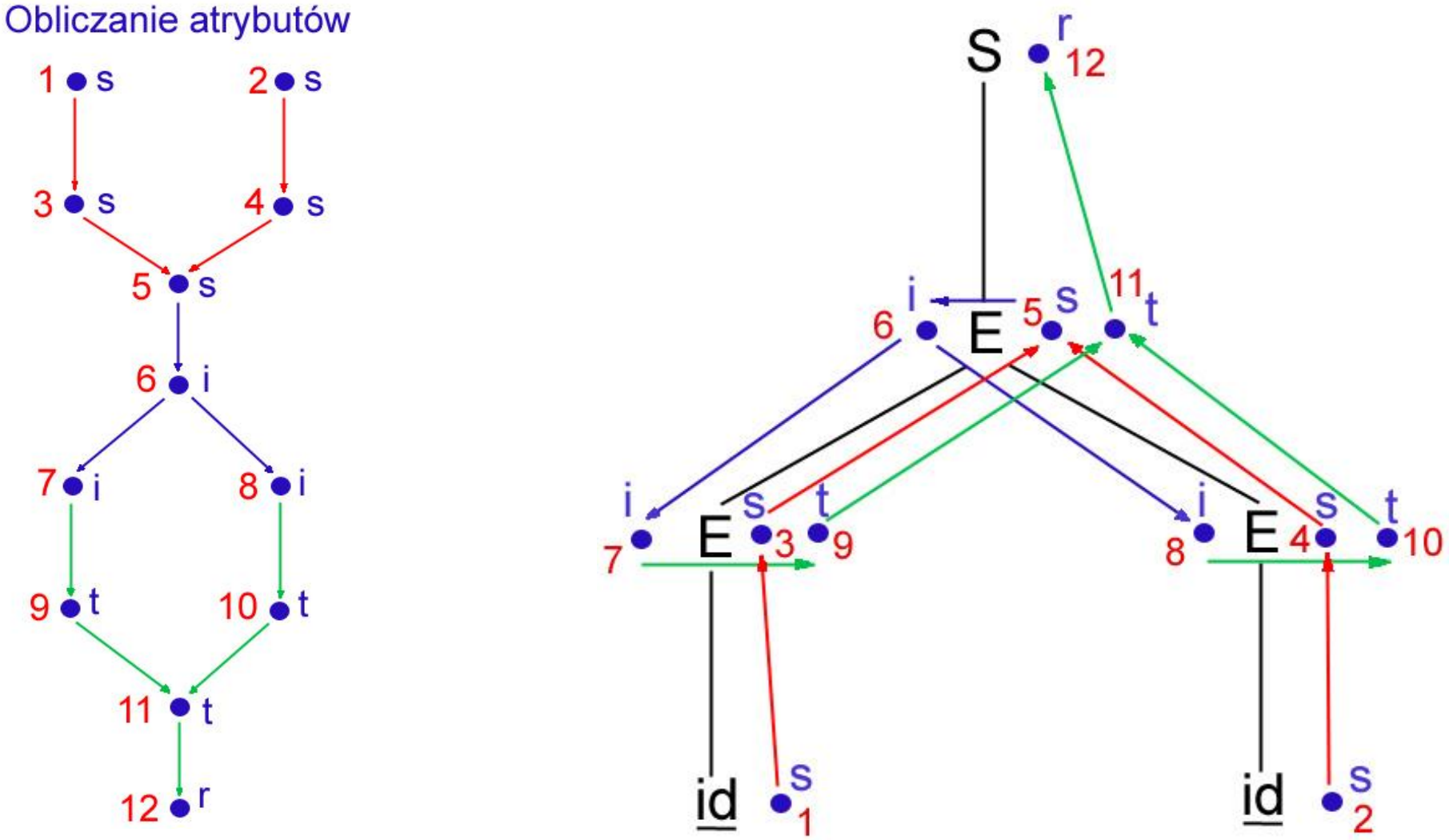


Przykład – „dekorowanie” drzewa rozbioru (2)

$S \to E$	$E.i \leftarrow g(E.s)$ $S.r \leftarrow E.t$
$E \to E_1 E_2$	$E.s \leftarrow f_s(E_1.s, E_2.s)$ $E_1.i \leftarrow f_{i1}(E.i)$ $E_2.i \leftarrow f_{i2}(E.i)$ $E.t \leftarrow f_t(E_1.t, E_2.t)$
$E \to \underline{id}$	$E.s \leftarrow \underline{id}.s$ $E.t \leftarrow h(E.i)$



Przykład – „dekorowanie” drzewa rozbioru (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:*

$E \rightarrow \underline{num}$	$\{E.type \leftarrow integer\}$
$E \rightarrow \underline{num}\ .\ \underline{num}$	$\{E.type \leftarrow real\}$
$E \rightarrow \underline{id}$	$\{E.type \leftarrow lookup(\underline{id}.lexptr)\}$
$E \rightarrow E_1\ \underline{op}\ E_2$	$\{E.type \leftarrow$ <u>if</u> $E_1.type = integer$ <u>and</u> $E_2.type = integer$ then $integer$ <u>else</u> <u>if</u> $E_1.type = integer$ <u>and</u> $E_2.type = real$ <u>then</u> $real$ <u>else</u> <u>if</u> $E_1.type = real$ <u>and</u> $E_2.type = integer$ <u>then</u> $real$ <u>else</u> <u>if</u> $E_1.type = real$ <u>and</u> $E_2.type = real$ <u>then</u> $real$ <u>else</u> $type\_error\}$

Przykłady reguł semantycznych

- *Kontrola typu instrukcji:*

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