



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Gramatyki atrybutywne, część 2 (gramatyki L-atorytywne)

Teoria kompilacji

Dr inż. Janusz Majewski
Katedra Informatyki

Przypomnienie: rodzaje atrybutów

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

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

Atrybut $X.a$ nazywamy syntetyzowanym $\stackrel{df}{\Leftrightarrow}$
 $\exists p = (X \rightarrow \chi) \in P : X.a \in AF(p)$

Atrybut $X.a$ nazywamy dziedziczonym $\stackrel{df}{\Leftrightarrow}$
 $\exists p = (Y \rightarrow \mu X \nu) \in P : X.a \in AF(p)$



Przykład – gramatyka atrybutywna z atrybutami dziedziczonymi

$D \rightarrow TL$	$L.in \leftarrow T.type$
$T \rightarrow \underline{int}$	$T.type \leftarrow integer$
$T \rightarrow \underline{float}$	$T.type \leftarrow real$
$L \rightarrow L_1, \underline{id}$	$L_1.in \leftarrow L.in$ $L.b \leftarrow addtype(\underline{id}.lexptr, L.in)$
$L \rightarrow \underline{id}$	$L.b \leftarrow addtype(\underline{id}.lexptr, L.in)$

Funkcja $addtype()$ wstawia do tablicy symboli informację o typie zmiennej

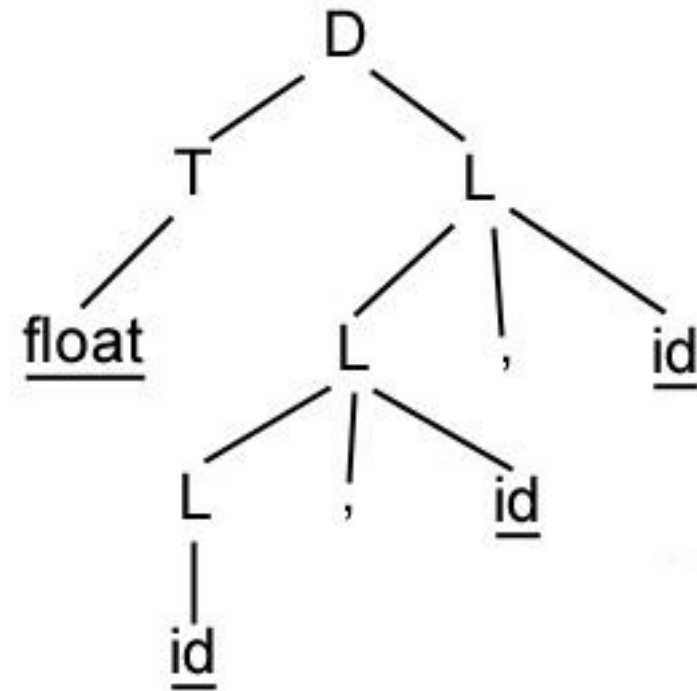
Analizowane słowo: $\underline{float} \underline{id}_1, \underline{id}_2, \underline{id}_3$

Przykład – gramatyka atrybutywna z atrybutami dziedziczonymi

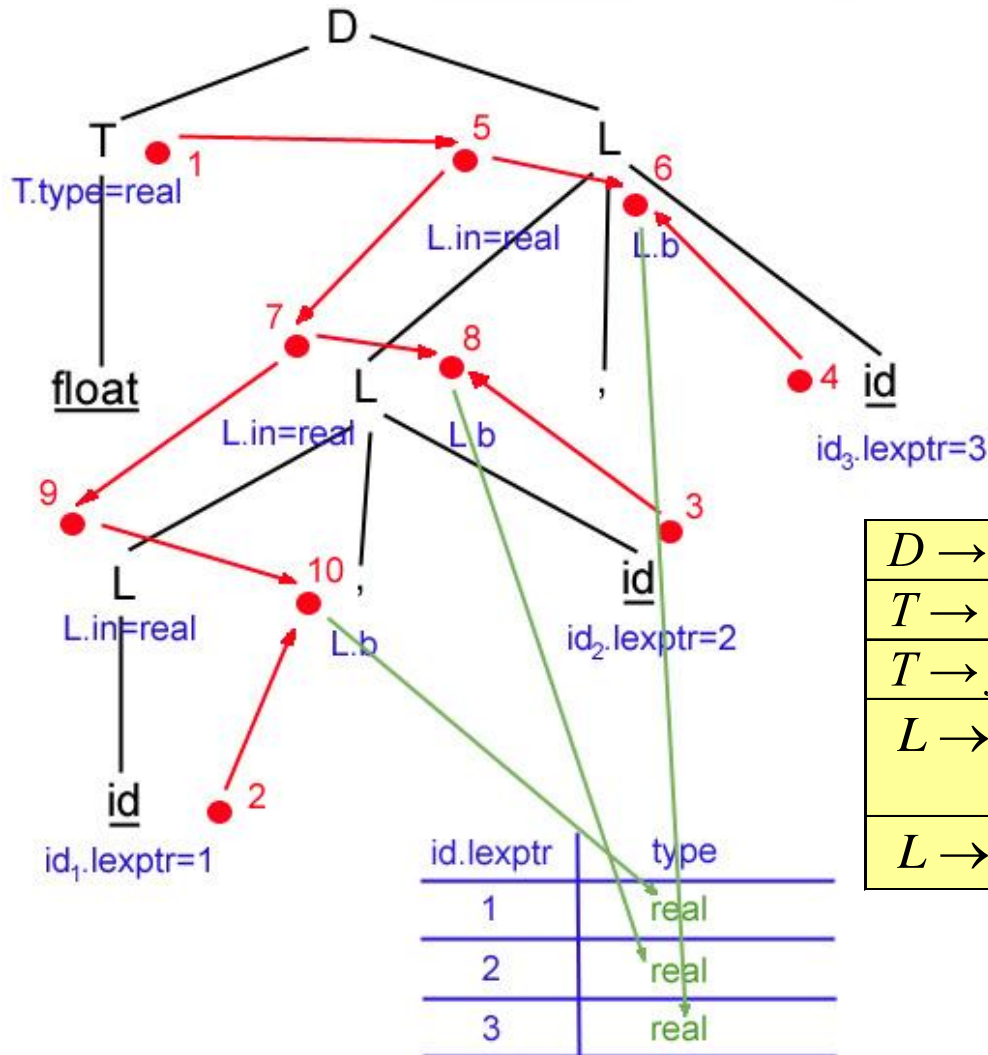
$D \rightarrow TL$	$L.in \leftarrow T.type$
$T \rightarrow \underline{int}$	$T.type \leftarrow integer$
$T \rightarrow \underline{float}$	$T.type \leftarrow real$
$L \rightarrow L_1, \underline{id}$	$L_1.in \leftarrow L.in$ $L.b \leftarrow addtype(\underline{id.lexptr}, L.in)$
$L \rightarrow \underline{id}$	$L.b \leftarrow addtype(\underline{id.lexptr}, L.in)$

Funkcja *addtype()* wstawia do tablicy symboli informację o typie zmiennej

Analizowane słowo: float id₁, id₂, id₃



Przykład – gramatyka atrybutywna z atrybutami dziedziczonymi



$D \rightarrow TL$	$L.in \leftarrow T.type$
$T \rightarrow \underline{int}$	$T.type \leftarrow integer$
$T \rightarrow \underline{float}$	$T.type \leftarrow real$
$L \rightarrow L_1, \underline{id}$	$L_1.in \leftarrow L.in$ $L.b \leftarrow addtype(id.lexptr, L.in)$
$L \rightarrow \underline{id}$	$L.b \leftarrow addtype(id.lexptr, L.in)$

Gramatyki L-atrybutywne

Definicja

Gramatyka atrybutywna jest gramatyką typu L (gramatyką L-atrybutywną), gdy każdy dziedziczony atrybut symbolu $X_j, 1 \leq j \leq n$ występującego w prawej stronie produkcji

$X_0 \rightarrow X_1 X_2 \dots X_n$ zależy tylko od:

- (1) atrybutów symboli X_1, X_2, \dots, X_{j-1} pojawiających się z lewej strony symbolu X_j w prawej części produkcji,
- (2) dziedziczonych atrybutów symbolu X_0 występującego w lewej stronie produkcji.

Każda gramatyka S-atrybutywna jest zarazem gramatyką L-atrybutywną.

Gramatyki L-atrybutywne

Dla gramatyk L-atrybutywnych możliwe jest obliczanie atrybutów podczas przeszukiwania drzewa rozbioru syntaktycznego w głąb metodą zejść rekurencyjnych (depth-first), zgodnie z ogólną procedurą DFVISIT:

```
procedure DFVISIT(n:wierzchołek);
```

```
begin
```

```
    for każdy potomek „m” wierzchołka „n” od lewej strony do prawej do
```

```
        begin
```

```
            oblicz atrybuty dziedziczone dla „m”;
```

```
            DFVISIT(m);
```

```
        end;
```

```
    oblicz atrybuty syntetyzowane dla „n”;
```

```
end;
```

Rozpoczęcie przeszukiwania=wywołanie:

```
DFVISIT(korzeń_drzewa_rozbioru_syntaktycznego);
```

Gramatyki L-atrybutywne

Gramatyki L-atrybutywne bazujące na gramatykach LL(1) umożliwiają obliczanie atrybutów równoległe 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.:

$$A \rightarrow A_1 Y \quad \{A.a \leftarrow g(A_1.a, Y.y)\}$$

$$A \rightarrow X \quad \{A.a \leftarrow f(X.x)\}$$

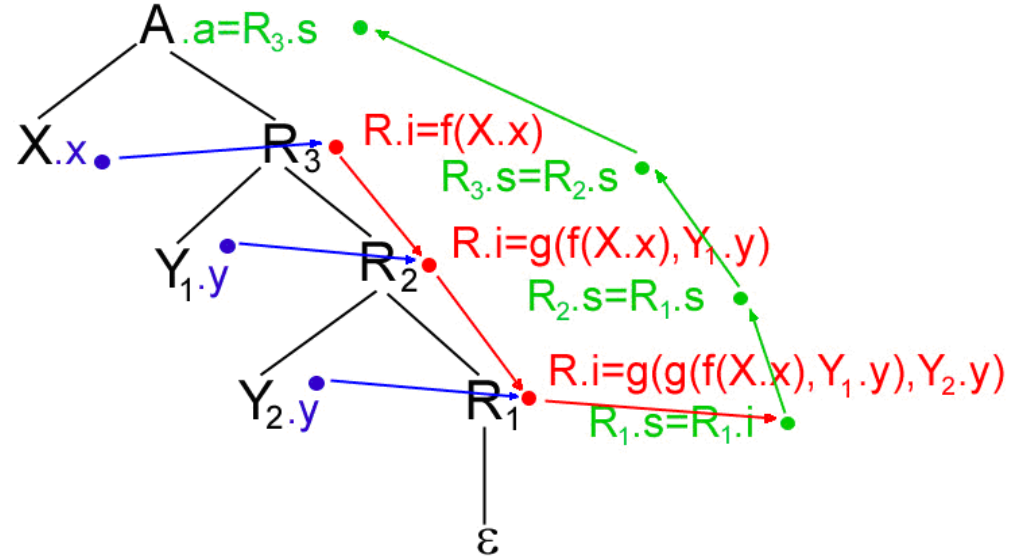
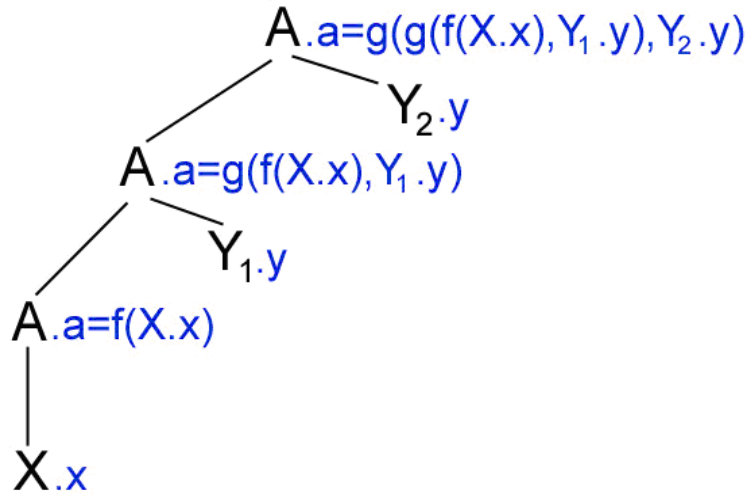
Wy: gramatyka L-atrybutywna bez lewej rekursji w postaci schematu tłumaczenia:

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

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

$$R \rightarrow \varepsilon \quad \{R.s \leftarrow R.i\}$$

Gramatyki L-atrybutywne



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

$$A \rightarrow A_1 Y \quad \{A.a \leftarrow g(A_1.a, Y.y)\}$$

$$A \rightarrow X \quad \{A.a \leftarrow f(X.x)\}$$

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

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

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

$$R \rightarrow \varepsilon \quad \{R.s \leftarrow R.i\}$$

Przykład – obliczanie wartości wyrażeń

$$E \rightarrow E_1 + T$$

$$\{E.val \leftarrow E_1.val + T.val\}$$

$$E \rightarrow E_1 - T$$

$$\{E.val \leftarrow E_1.val - T.val\}$$

$$E \rightarrow T$$

$$\{E.val \leftarrow T.val\}$$

$$T \rightarrow (E)$$

$$\{T.val \leftarrow E.val\}$$

$$T \rightarrow \underline{num}$$

$$\{T.val \leftarrow \underline{num.val}\}$$

Po usunięciu lewej rekursji:

$$E \rightarrow TR$$

$$R \rightarrow +TR$$

$$R \rightarrow -TR$$

$$R \rightarrow \varepsilon$$

$$T \rightarrow (E)$$

$$T \rightarrow \underline{num}$$

Przykład – obliczanie wartości wyrażeń

Po usunięciu lewej rekursji:

$$E \rightarrow TR$$

$$R \rightarrow +TR$$

$$R \rightarrow -TR$$

$$R \rightarrow \varepsilon$$

$$T \rightarrow (E)$$

$$T \rightarrow \underline{num}$$

i po przeróbce na gramatykę L-atrybutywną uzyskujemy schemat tłumaczenia:

$$E \rightarrow T \{R.i \leftarrow T.val\} \quad R \quad \{E.val \leftarrow R.s\}$$

$$R \rightarrow +T \{R_1.i \leftarrow R.i + T.val\} \quad R_1 \quad \{R.s \leftarrow R_1.s\}$$

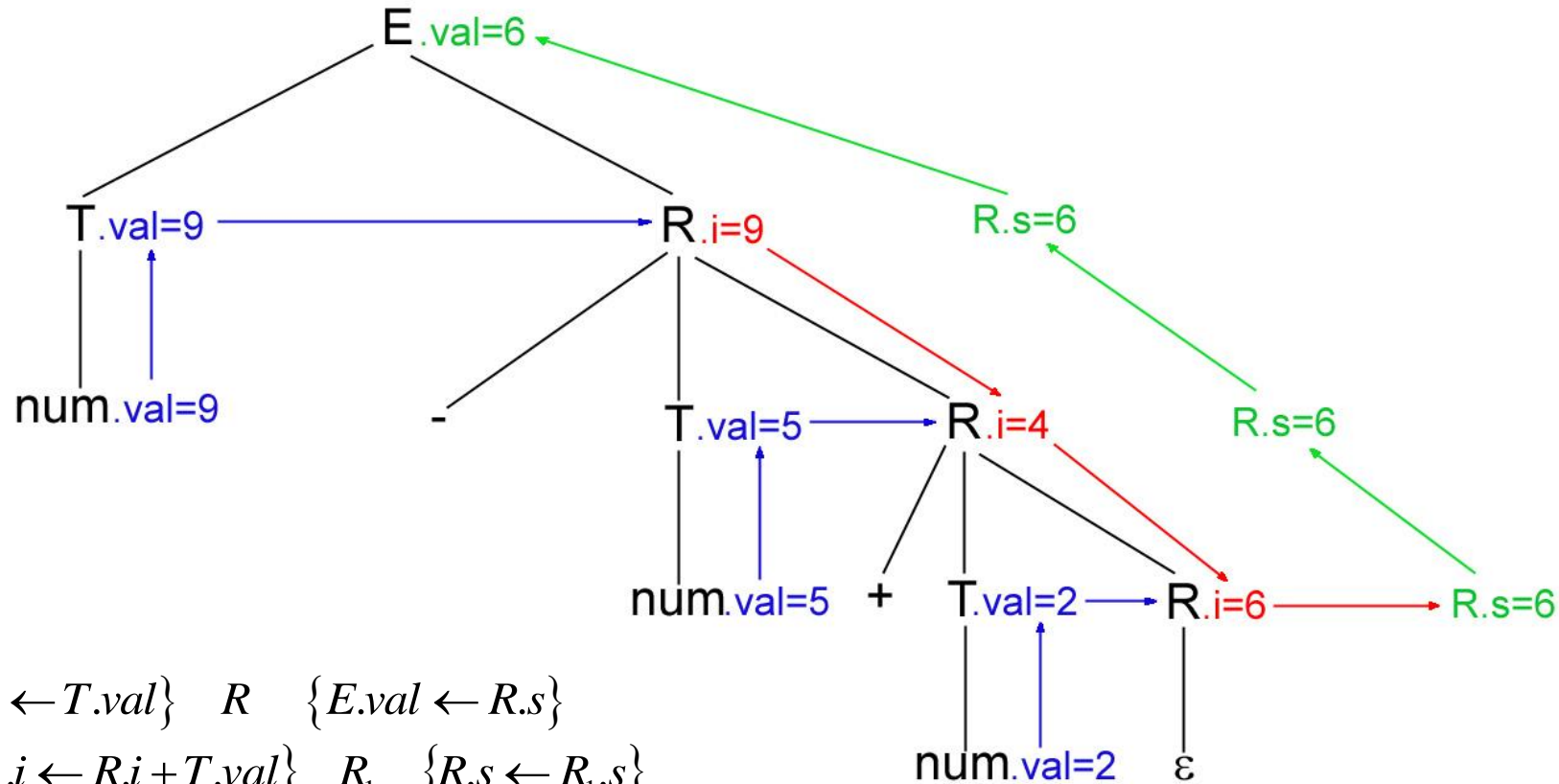
$$R \rightarrow -T \{R_1.i \leftarrow R.i - T.val\} \quad R_1 \quad \{R.s \leftarrow R_1.s\}$$

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

$$T \rightarrow (E) \{T.val \leftarrow E.val\}$$

$$T \rightarrow \underline{num} \{T.val \leftarrow \underline{num}.val\}$$

Przykład – obliczanie wartości wyrażeń



$E \rightarrow T \{R.i \leftarrow T.val\} \quad R \quad \{E.val \leftarrow R.s\}$

$R \rightarrow +T \{R_1.i \leftarrow R.i + T.val\} \quad R_1 \quad \{R.s \leftarrow R_1.s\}$

$R \rightarrow -T \{R_1.i \leftarrow R.i - T.val\} \quad R_1 \quad \{R.s \leftarrow R_1.s\}$

$R \rightarrow \epsilon \{R.s \leftarrow R.i\}$

$T \rightarrow (E) \{T.val \leftarrow E.val\}$

$T \rightarrow \underline{num} \{T.val \leftarrow \underline{num.val}\}$



Algorytm obliczania atrybutów w gramatyce L-atorybutywnej podczas parsingu rekurencyjnego

Algorytm tworzenia funkcji rekurencyjnych dla obliczania atrybutów gramatyki L-atorybutywnej bazującej na gramatyce LL(1) (obliczanie podczas parsingu)

We: schemat tłumaczenia dla LL(1) gramatyki typu L-atorybutywnego;
tablica parsera LL(1)

Wy: fragment kodu tłumacza.

1. Dla każdego nieterminala $A \in V$ konstruuje się funkcję o nazwie A . Każdemu atrybutowi dziedziczonemu symbolu nieterminalnego A odpowiadać ma jeden formalny parametr.

Funkcja ma zwracać wartość atrybutu syntetyzowanego dla A . (Przyjęte założenie, że A posiada tylko jeden atrybut syntetyzowany nie powoduje utraty ogólności, gdyż można zwrócić wskaźnik do struktury zawierającej wartości większej liczby atrybutów syntetyzowanych symbolu A). Funkcja powinna zawierać zmienną lokalną dla każdego atrybutu każdego symbolu gramatyki pojawiającego się we wszystkich produkcjach $A \rightarrow \dots$



Algorytm obliczania atrybutów w gramatyce L-atorytywnej podczas parsingu rekurencyjnego

2. W kodzie funkcji podejmuje się decyzję którą z produkcji typu $A \rightarrow \dots$ użyto w danym wierzchołku na podstawie bieżącego symbolu wejściowego z wykorzystaniem tablicy parsera LL(1). Każdej produkcji $A \rightarrow \dots$ odpowiada odrębna ścieżka w kodzie funkcji.

Algorytm obliczania atrybutów w gramatyce L-atorytywnej podczas parsingu rekurencyjnego

3. W ramach kodu dla produkcji $A \rightarrow \dots$ rozważamy terminale, nieterminala i akcje prawej strony produkcji w takiej kolejności, w jakiej występują one w schemacie tłumaczenia analizowanym od lewej do prawej:
 - (a) dla terminala X z atrybutem (syntetyzowanym) $X.x$ obliczamy ten atrybut, zachowujemy jego wartość w zmiennej lokalnej i przesuwamy wejście (bierzemy następny symbol z wejścia),
 - (b) dla nieterminala B generujemy wywołanie odpowiadającej funkcji:
 $B.s \leftarrow B(b_1, \dots, b_k)$ gdzie:
 $B.s$ to zmienna lokalna dla atrybutu syntetyzowanego symbolu B
 $B()$ - wywołanie funkcji dla symbolu B
 b_1, \dots, b_k - zmienne lokalne dla atrybutów dziedziczonych symbolu B
 - (c) kopiujemy akcje zastępując odwołania do atrybutów odpowiednimi zmiennymi lokalnymi. Ostatnią akcją będzie obliczenie atrybutu syntetyzowanego nieterminala A stojącego po lewej stronie produkcji $A \rightarrow \dots$. Wartość tę funkcja zwraca.



Algorytm obliczania atrybutów w gramatyce L-atorytywnej podczas parsingu rekurencyjnego

Tak skonstruowane funkcje nie uwzględniają kontroli sytuacji błędnych (błędna postać ciągu wejściowego).

Rozpoczęcie parsingu i tłumaczenia następuje poprzez wywołanie funkcji dla symbolu początkowego gramatyki.



Przykład – obliczanie wartości wyrażeń

	<u>num</u>	()	+	-	\$
E	$E \rightarrow TR$	$E \rightarrow TR$				
R			$R \rightarrow \varepsilon$	$R \rightarrow +TR$	$R \rightarrow -TR$	$R \rightarrow \varepsilon$
T	$T \rightarrow \underline{num}$	$T \rightarrow (E)$				

$E \rightarrow T \{R.i \leftarrow T.val\} \quad R \quad \{E.val \leftarrow R.s\}$

```

function E : val;      /*zwraca E.val*/
  var T_val_loc, R_i_loc, R_s_loc : val;
begin
  if (lookahead=num) or (lookahead='(') then
    begin /*produkcja  $E \rightarrow TR$ */
      T_val_loc := T;
      R_i_loc := T_val_loc;
      R_s_loc := R(R_i_loc);
      E := R_s_loc;
    end
  else error;
end;

```



Przykład – obliczanie wartości wyrażeń

	<u>num</u>	()	+	-	\$
E	$E \rightarrow TR$	$E \rightarrow TR$				
R			$R \rightarrow \varepsilon$	$R \rightarrow +TR$	$R \rightarrow -TR$	$R \rightarrow \varepsilon$
T	$T \rightarrow \underline{num}$	$T \rightarrow (E)$				

$R \rightarrow +T \{R_1.i \leftarrow R.i + T.val\} \quad R_1 \quad \{R.s \leftarrow R_1.s\}$

```

function R(R_i : val) : val;  /*zwraca R.s*/
  var R_i_loc, R_s_loc, T_val_loc : val;
begin
  if (lookahead = '+') then  /*R → +TR*/
    begin
      match (lookahead);
      T_val_loc:=T;
      R_i_loc:=R_i+T_val_loc;
      R_s_loc:=R(R_i_loc);
      R:=R_s_loc;
    end
  end
  .....
```

Przykład – obliczanie wartości wyrażeń

	<u>num</u>	()	+	-	\$
E	$E \rightarrow TR$	$E \rightarrow TR$				
R			$R \rightarrow \varepsilon$	$R \rightarrow +TR$	$R \rightarrow -TR$	$R \rightarrow \varepsilon$
T	$T \rightarrow \underline{\text{num}}$	$T \rightarrow (E)$				

$$R \rightarrow -T \{R_1.i \leftarrow R.i - T.val\} \quad R_1 \quad \{R.s \leftarrow R_1.s\}$$

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

```

function R(R_i : val) : val;  /*zwraca R.s*/
  var R_i_loc, R_s_loc, T_val_loc : val;
begin  if (lookahead = '+') then ..... /*R → +TR*/
      else if (lookahead = '-') then /* R → -TR */
        begin
          match (lookahead);
          T_val_loc := T;
          R_i_loc := R_i - T_val_loc;
          R_s_loc := R(R_i_loc);
          R := R_s_loc;
        end
      else if (lookahead = '$') or (lookahead = ')') then /* R → ε */
        R := R_i;
      else error;
end;
```



Przykład – obliczanie wartości wyrażeń

	<u>num</u>	()	+	-	\$
E	$E \rightarrow TR$	$E \rightarrow TR$				
R			$R \rightarrow \varepsilon$	$R \rightarrow +TR$	$R \rightarrow -TR$	$R \rightarrow \varepsilon$
T	$T \rightarrow \underline{num}$	$T \rightarrow (E)$				

$T \rightarrow (E) \quad \{T.val \leftarrow E.val\}$

$T \rightarrow \underline{num} \quad \{T.val \leftarrow \underline{num}.val\}$

```

function T : val;      /*zwraca T.val*/
  var T_val_loc, E_val_loc, num_val_loc : val;
begin
  if (lookahead = num) then  /*T → num*/
    begin
      T_val_loc := value (num);
      match (num);
      T := T_val_loc;
    end
  else if (lookahead='(') then ..... /* T → (E) */
  else error;
end;

```

Przykład – obliczanie wartości wyrażeń

	<u>num</u>	()	+	-	\$
E	$E \rightarrow TR$	$E \rightarrow TR$				
R			$R \rightarrow \varepsilon$	$R \rightarrow +TR$	$R \rightarrow -TR$	$R \rightarrow \varepsilon$
T	$T \rightarrow \underline{num}$	$T \rightarrow (E)$				

$T \rightarrow (E) \quad \{T.val \leftarrow E.val\}$

$T \rightarrow \underline{num} \quad \{T.val \leftarrow \underline{num}.val\}$

```

function T : val;      /*zwraca T.val*/
  var T_val_loc, E_val_loc, num_val_loc : val;
begin
  if (lookahead=num) then ..... /*T → num*/
  else if (lookahead='(') then /* T → (E) */
    begin
      match ('(');
      E_val_loc:=E;
      match (')');
      T:=E_val_loc;
    end
  else error;
end;
  
```



Przykład – obliczanie wartości wyrażeń

```
var lookahead : token;      /*zmienna globalna – token podglądany  
                             na wejściu*/
```

```
procedure match (t:token);  /*realizacja operacji „pop”*/  
begin  
    if lookahead = t then  
        lookahead := next_token  
    else  
        error;  
    end;
```

```
function next_token : token; /*czyta i zwraca token z wejścia; niszczy  
                             przeczytany token*/
```

```
begin ... end;
```

```
function value (t : token) : val;    /*zwraca wartość tokenu „num”*/  
begin ... end;
```



Przykład – obliczanie wartości wyrażeń

procedure error;
begin ... end;

*/*sygnalizacja i obsługa błędów*/*

function Translate : val

*/*zwraca wynik interpretacji czyli wartość analizowanego wyrażenia*/*

begin

... */*czynności wstępne*/*

lookahead := next_token; */*czytanie pierwszego tokenu*/*

Translate := E; */*wywołanie funkcji rekurencyjnej dla symbolu początkowego gramatyki*/*

end;

Algorytm obliczania atrybutów w gramatyce L-atorytywnej podczas parsingu predykcyjnego (idea algorytmu)

Algorithm

LL(1) L-Attributed Evaluation

```
FOR each predicted production  $X_0 \rightarrow X_1X_2\dots X_n$ .  
  Push  $X_0$ 's inherited attributes onto semantic stack.  
  Push  $X_1$ 's inherited attributes onto semantic stack.  
  Parse  $X_1$ , then push  $X_1$ 's synthesized attributes onto  
    semantic stack.  
  Push  $X_2$ 's inherited attributes onto semantic stack.  
  Parse  $X_2$ , then push  $X_2$ 's synthesized attributes onto  
    semantic stack ...  
  Push  $X_n$ 's inherited attributes onto semantic stack.  
  Parse  $X_n$ , then push  $X_n$ 's synthesized attributes onto  
    semantic stack.  
  Pop attributes of  $X_1X_2, \dots, X_n$ .  
  Push synthesized attributes of  $X_0$ .  
ENDFOR
```




Algorytm obliczania atrybutów w gramatyce L-atorybutywnej podczas parsingu predykcyjnego (przykład)

Gramatyka L-atorybutywna bez lewej rekursji w postaci schematu tłumaczenia:

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

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

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

Zamieniamy reguły obliczające atrybuty w **<symbole_akcji>**. Będą one umieszczane na stosie parsera jak terminale czy nieterminale i kojarzone z odpowiednimi regułami obliczającymi atrybuty. Przetwarzanie symboli akcji polega na obliczeniu atrybutu definiowanego przez regułę, zdjęciu ze stosu atrybutów niepotrzebnych argumentów reguły i położeniu na stosie atrybutów wartości obliczonego atrybutu.

$$A \rightarrow X \langle R.i \rangle R \langle A.s \rangle$$

$$R \rightarrow Y \langle R_1.i \rangle R_1 \langle R.s \rangle$$

$$R \rightarrow \varepsilon \langle R.s \rangle$$

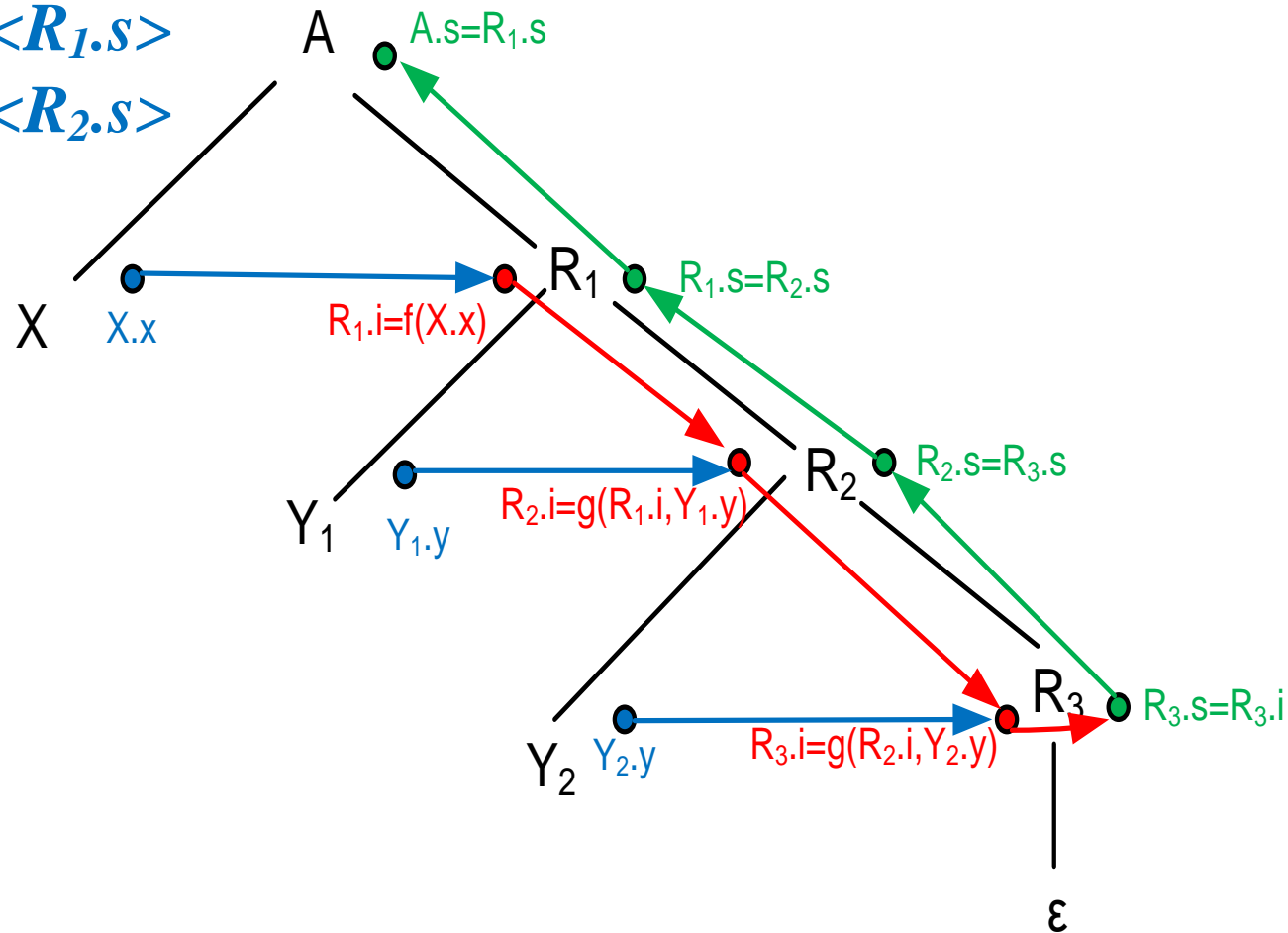
Algorytm obliczania atrybutów w gramatyce L-atrybutywnej podczas parsingu predykcyjnego - przykład

$$A \rightarrow X \langle R_1.i \rangle R_1 \langle A.s \rangle$$

$$R_1 \rightarrow Y_1 \langle R_2.i \rangle R_2 \langle R_1.s \rangle$$

$$R_2 \rightarrow Y_2 \langle R_3.i \rangle R_3 \langle R_2.s \rangle$$

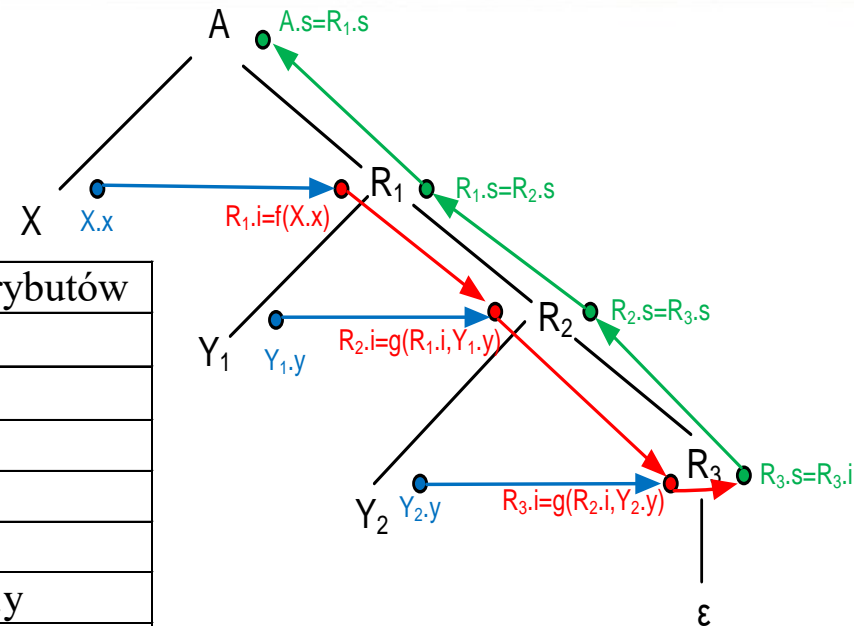
$$R_3 \rightarrow \varepsilon \langle R_3.s \rangle$$



Algorytm obliczania atrybutów w gramatyce L-atrybutywnej podczas parsingu predykcyjnego - przykład c. d.

$A \rightarrow X \langle R_1.i \rangle R_1 \langle A.s \rangle$
 $R_1 \rightarrow Y_1 \langle R_2.i \rangle R_2 \langle R_1.s \rangle$
 $R_2 \rightarrow Y_2 \langle R_3.i \rangle R_3 \langle R_2.s \rangle$
 $R_3 \rightarrow \varepsilon \langle R_3.s \rangle$

Stos parsera	Wejście	Stos atrybutów
A	XY ₁ Y ₂ \$	ε
<A.s> R ₁ <R ₁ .i> X	X Y ₁ Y ₂ \$	ε
<A.s> R ₁ <R ₁ .i>	Y ₁ Y ₂ \$	X.x
<A.s> R ₁	Y ₁ Y ₂ \$	R ₁ .i
<A.s> <R ₁ .s> R ₂ <R ₂ .i> Y ₁	Y ₁ Y ₂ \$	R ₁ .i
<A.s> <R ₁ .s> R ₂ <R ₂ .i>	Y ₂ \$	R ₁ .i Y ₁ .y
<A.s> <R ₁ .s> R ₂	Y ₂ \$	R ₂ .i
<A.s> <R ₁ .s> <R ₂ .s> R ₃ <R ₃ .i> Y ₂	Y ₂ \$	R ₂ .i
<A.s> <R ₁ .s> <R ₂ .s> R ₃ <R ₃ .i>	\$	R ₂ .i Y ₂ .y
<A.s> <R ₁ .s> <R ₂ .s> R ₃	\$	R ₃ .i
<A.s> <R ₁ .s> <R ₂ .s> <R ₃ .s>	\$	R ₃ .i
<A.s> <R ₁ .s> <R ₂ .s>	\$	R ₃ .s
<A.s> <R ₁ .s>	\$	R ₂ .s
<A.s>	\$	R ₁ .s
ε	\$	A.s





Translacja bottom-up dla gramatyk L-atorybutywnych

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

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

Przykład: wypisywanie wyrażeń w odwrotnej notacji polskiej

Gramatyka atrybutywna typu L:

$$E \rightarrow TR$$

$$R \rightarrow +T \{R_1.i \leftarrow \text{print} ('+')\} R_1$$

$$R \rightarrow -T \{R_1.i \leftarrow \text{print} ('-')\} R_1$$

$$R \rightarrow \varepsilon$$

$$T \rightarrow \underline{\text{num}} \{T.s \leftarrow \text{print}(\text{text}(\underline{\text{num}}.val))\}$$

Jest transformowana do postaci typu S:

$$E \xrightarrow{(1)} \overline{TR}$$

$$R \xrightarrow{(2)} + \overline{TMR} \mid \xrightarrow{(3)} - \overline{TNR} \mid \xrightarrow{(4)} \varepsilon$$

$$T \xrightarrow{(5)} \underline{\text{num}} \quad \{T.s \leftarrow \text{print}(\text{text}(\underline{\text{num}}.val))\}$$

$$M \xrightarrow{(6)} \underline{\varepsilon} \quad \{M.s \leftarrow \text{print} ('+')\}$$

$$N \xrightarrow{(7)} \underline{\varepsilon} \quad \{N.s \leftarrow \text{print} ('-')\}$$

Przykład: wypisywanie wyrażeń w odwrotnej notacji polskiej

state	we	wy	
	9-5+2		
<u>num</u>	-5+2		<i>shift</i>
T	-5+2	9	<i>red 5</i>
T-	5+2	9	<i>shift</i>
T- <u>num</u>	+2	9	<i>shift</i>
T-T	+2	95	<i>red 5</i>
T-TN	+2	95-	<i>red 7</i>
T-TN+	2	95-	<i>shift</i>
T-TN+ <u>num</u>	ϵ	95-	<i>shift</i>
T-TN+T	ϵ	95-2	<i>red 5</i>
T-TN+TM	ϵ	95-2+	<i>red 6</i>
T-TN+TMR	ϵ	95-2+	<i>red 4</i>
T-TNR	ϵ	95-2+	<i>red 2</i>
TR	ϵ	95-2+	<i>red 3</i>
E	ϵ	95-2+	<i>red 1</i>
			<i>acc</i>

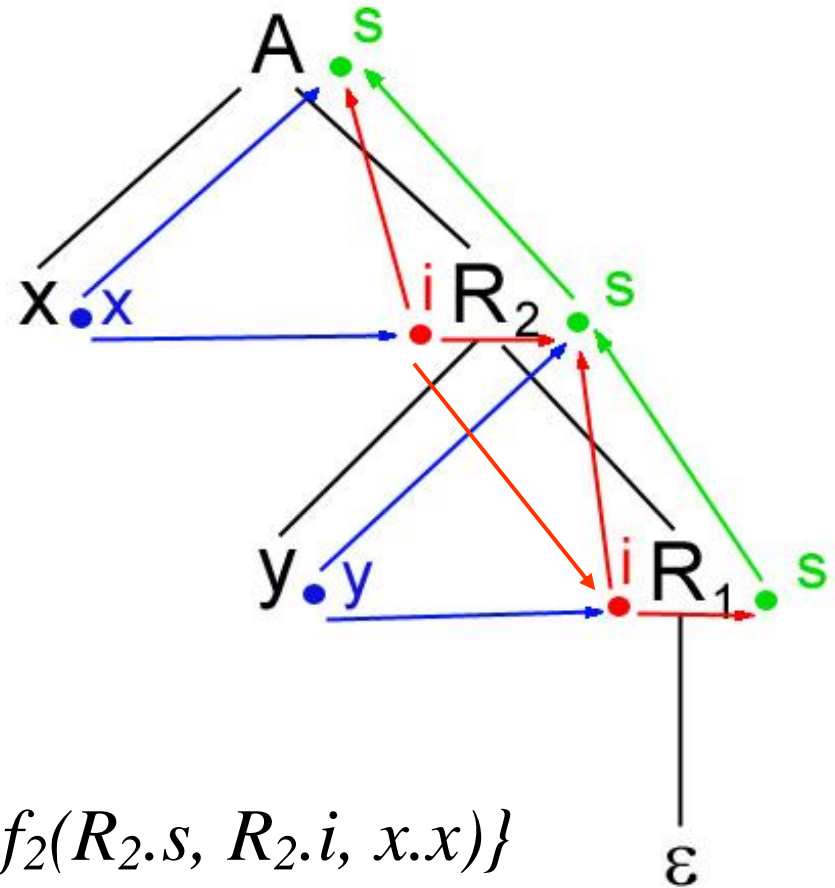
- (1) $E \rightarrow TR$
- (2) $R \rightarrow +TMR$
- (3) $R \rightarrow -TNR$
- (4) $R \rightarrow \epsilon$
- (5) $T \rightarrow \underline{num}$ $\{print(text(num.val))\}$
- (6) $M \rightarrow \epsilon$ $\{print ('+')\}$
- (7) $N \rightarrow \epsilon$ $\{print ('-')\}$

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:

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

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

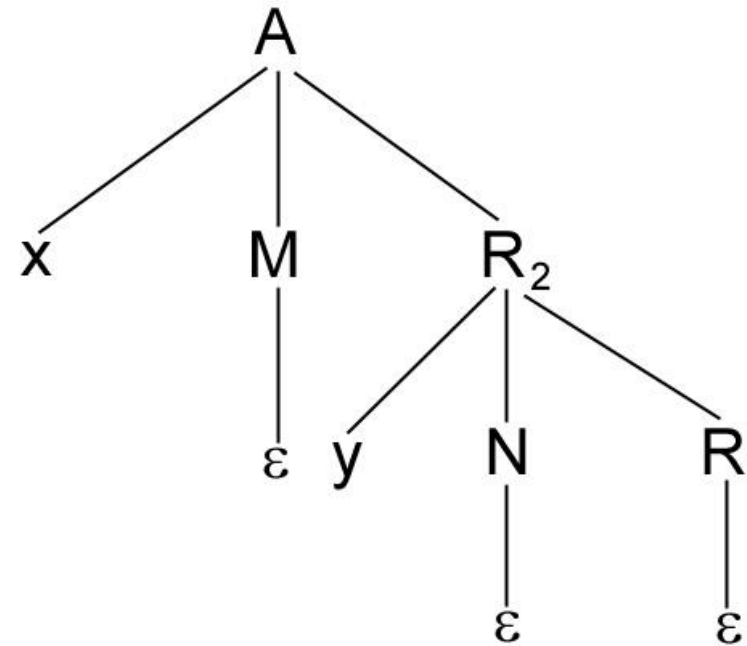
$$(3) \quad R_1 \rightarrow \varepsilon \{R_1.s \leftarrow f_5(R_1.i)\}$$

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

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

Przykład – tworzenie nowych nieterminali dla atrybutów dziedziczonych

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



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

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

Przed redukcją

R ₁	R ₁ .s	top
N	R ₁ .i	top-1
y	y.y	top-2
M	R ₂ .i	top-3
X	X.X	top-4
state	attr	

```

ntop:=top-2;
R2.s attr[ntop]:=
  R1.s   R1.i
  f (attr[top], attr[top-1],
    attr[top-3]. attr[top-2]);
  R2.i   y.y
  
```

Po redukcji

R ₂	R ₂ .s	ntop
M	R ₂ .i	
X	X.X	
state	attr	

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



Parsing bottom-up dla gramatyk L-atorybutywnych

Ogólny algorytm obliczania atrybutów w trakcie parsingu bottom-up dla gramatyk L-atorybutywnych opartych na gramatykach LL(1) (a także dla niektórych LR(1) nie będących LL(1))

We: Gramatyka L-atorybutywna oparta na gramatyce LL(1);
gramatyka syntaktyczna nie zawiera symbolu początkowego w prawych stronach produkcji.

Zał: Symbol początkowy gramatyki nie posiada atrybutu dziedzicznego. Każdy symbol gramatyki posiada jeden atrybut syntetyzowany i jeden dziedziczony.

Wy: Algorytm parsera bottom-up obliczającego atrybuty podczas analizy słowa wejściowego.



Parsing bottom-up dla gramatyk L-atorybutywnych

$A.i$ - atrybut dziedziczony A ; $A.s$ - atrybut syntetyzowany A

$X_j.i$ - atrybut dziedziczony X_j ; $X_j.s$ - atrybut syntetyzowany X_j

Każda produkcja: $A \rightarrow X_1 X_2 \dots X_j \dots X_n$ jest przekształcana do postaci:

$$A \rightarrow M_1 X_1 M_2 X_2 \dots M_j X_j \dots M_n X_n$$

$$M_1 \rightarrow \varepsilon$$

$$M_2 \rightarrow \varepsilon$$

.....

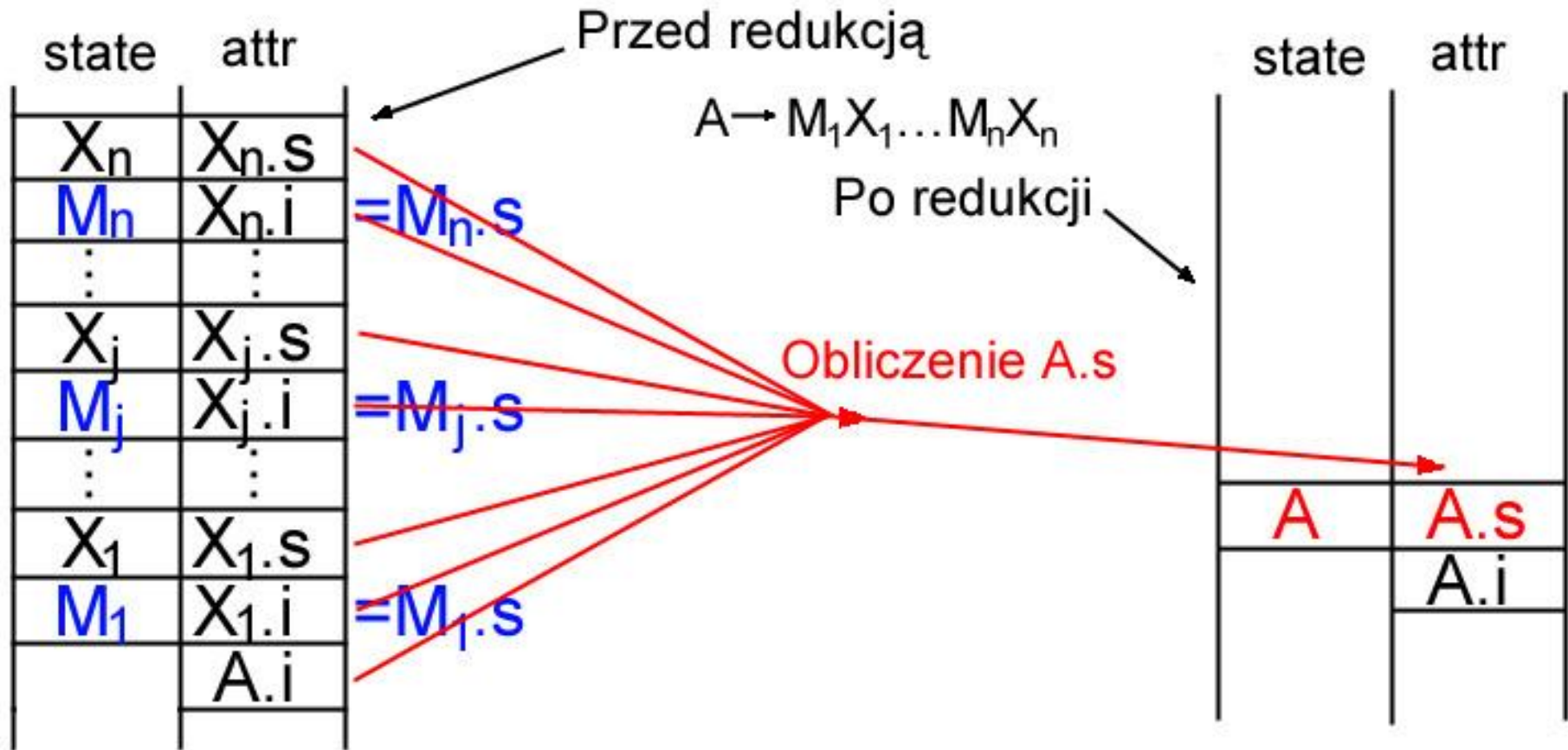
$$M_j \rightarrow \varepsilon$$

.....

$$M_n \rightarrow \varepsilon$$

Atrybut dziedziczony $X_j.i$ każdego symbolu X_j jest utożsamiany z atrybutem syntetyzowanym $M_j.s$ dodatkowego nieterminala M_j .

Parsing bottom-up dla gramatyk L-atrybutywnych



Parsing bottom-up dla gramatyk L-atrybutywnych

