Metody Kompilacji
Wykład 3

Translacja sterowana składnią

- Translacja sterowana składnią odbywa się poprzez dołączenie zasad(reguł) lub fragmentów kodu do produkcji w gramatyce.

Translacja sterowana składnią

Na przykład, dla produkcji

$expr \rightarrow expr_1 + term$

możemy przetwarzać *expr* wykorzystując strukturę produkcji zgodnie z poniższym pseudokodem:

przetwarzaj $expr_1$;

przetwarzaj *term;*

przetwarzaj +;

Translacja sterowana składnią

- Atrybut jest to pewna wartość powiązana z konstrukcją języka programowania.

- Przykładami atrybutów są typy danych wyrażeń, liczba instrukcji wygenerowanego kodu lub lokalizacja pierwszej instrukcji generowanego kodu.

Translacja sterowana składnią

- Schemat translacji sterowany składnią jest to gramatyka, gdzie do każdej produkcji jest przypisany fragment kodu -- akcja semantyczna.
- Fragmenty kodu są wykonywane podczas gdy produkcja jest wykorzystywana przez parser.

Translacja sterowana składnią

Notacja postfiksowa

- Notacja postfiksowa może być zdefiniowana w następujący sposób:

1. Jeżeli *E* oznacza zmienną lub stałą, to notacja postfiksowa dla *E* jest samo *E*.

Translacja sterowana składnią

Notacja postfiksowa

- 2. Jeżeli E jest wyrażeniem postaci E_1 **op** E_2, gdzie **op** jest operatorem binarnym,

  to notacją postfiksową dla E jest $E_1{}' E_2{}'$ **op**, gdzie $E_1{}' E_2{}'$ są notacjami postfiksowymi dla E_1 i E_2, odpowiednio.

Translacja sterowana składnią

Notacja postfiksowa

- 3. Jeżeli E jest wyrażeniem o postaci (E_1), to notacja postfiksowa dla E jest taka sama jak notacja postfiksowa dla E_1.

Translacja sterowana składnią

Notacja postfiksowa

- Przykład: Notacją postfiksową dla wyrażenia (9-5)+2 jest 95-2+.
- Oznacza to, że przekład dla 9, 5 i 2 jest reprezentowany przez te same stałe zgodnie z regułą (1).
- Tłumaczeniem 9-5 zgodnie z reguła (2) jest 95- . Tłumaczeniem dla (9-5) zgodnie z regułą (3) jest 9-5.

Translacja sterowana składnią

Notacja postfiksowa

- Nawiasy nie są potrzebne w notacji postfiksowej, ponieważ pozycja i liczba argumentów operatorów w sposób jednoznaczny określają kolejność wykonywania operatorów.

Translacja sterowana składnią

Atrybuty syntezowane

- Atrybuty kojarzymy z nieterminalami i terminalami.
- Żeby obliczyć atrybuty nieterminali, dodajemy reguły do produkcji.
- Reguły te opisują w jaki sposób obliczane są atrybuty w węzłach drzewa parsowania.

Translacja sterowana składnią

Definicja sterowana składnią

- Definicja sterowana składnią określa się przez:

  1. Zbiór atrybutów dla każdego symbolu gramatycznego.
  2. Zbiór reguł semantycznych do obliczania wartości atrybutów związanych z symbolami występującymi w produkcji.

Translacja sterowana składnią

Atrybuty syntezowane

Atrybuty mogą być obliczone w następujący sposób. Dla danego ciągu wejściowego *x* tworzymy drzewo parsowania.

Następnie stosujemy reguły semantyczne w każdym węźle drzewa parsowania w następujący sposób.

Translacja sterowana składnią

Atrybuty syntezowane

- Załóżmy, że węzeł *N* w drzewie parsowania jest oznaczony symbolem *X*.
- Wtedy zapisujemy wartość atrybutu *a* w tym węźle jako *X.a* .
- Drzewo parsowania, pokazujące wartości atrybutów w każdym węźle, nazywamy drzewem parsowania z przypisami.

Translacja sterowana składnią

Atrybuty syntezowane

- Wartość atrybutu syntezowanego dla węzła *N* w drzewie parsowania oblicza się w oparciu o atrybuty jego dzieci i atrybutu własnego.

Przykład obliczenia atrybutów

Definicja sterowana składnią do tłumaczenia postaci infiksowej na postfiksową.

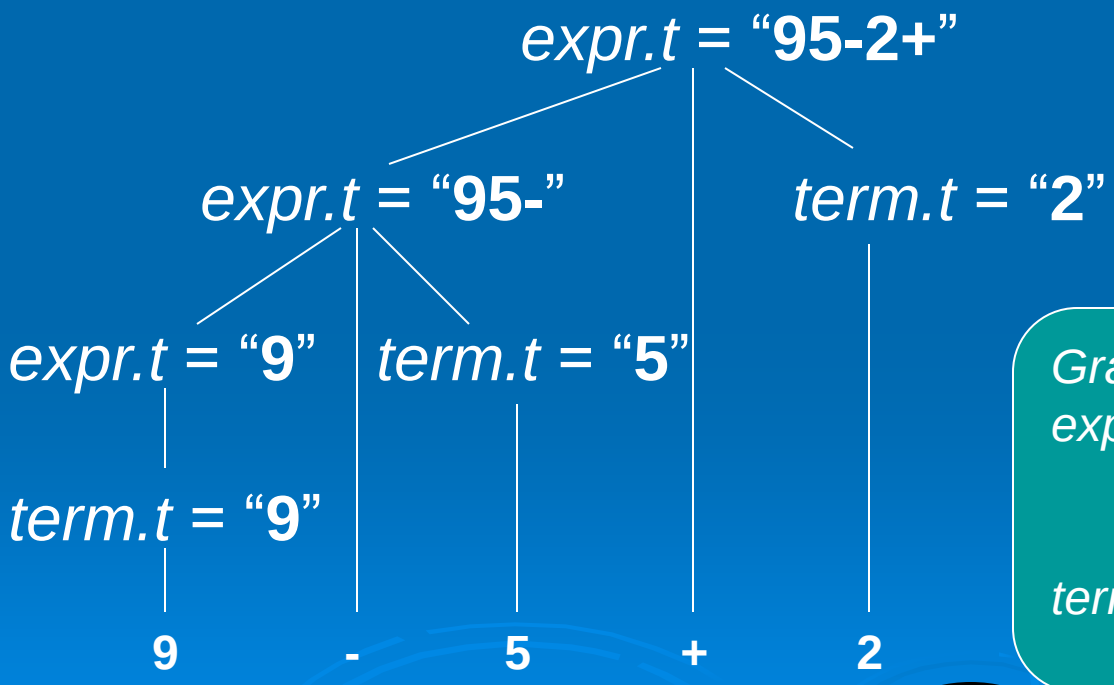
Produkcja	Reguły semantyczne
$expr \rightarrow expr_1$ **+** $term$	$expr.t := expr_1.t \,\|\|\, term.t \,\|\|$ "+"
$expr \rightarrow expr_1$ **-** $term$	$expr.t := expr_1.t \,\|\|\, term.t \,\|\|$ "-"
$expr \rightarrow term$	$expr.t := term.t$
$term \rightarrow$ **0**	$term.t :=$ "0"
$term \rightarrow$ **1**	$term.t :=$ "1"
…	…
$term \rightarrow$ **9**	$term.t :=$ "9"

Znak „|| „ oznacza operator konkatenacji

Translacja sterowana składnią

Atrybuty syntezowane

Wartości atrybutów w węzłach drzewa parsowania dla ciągu: 9-5+2

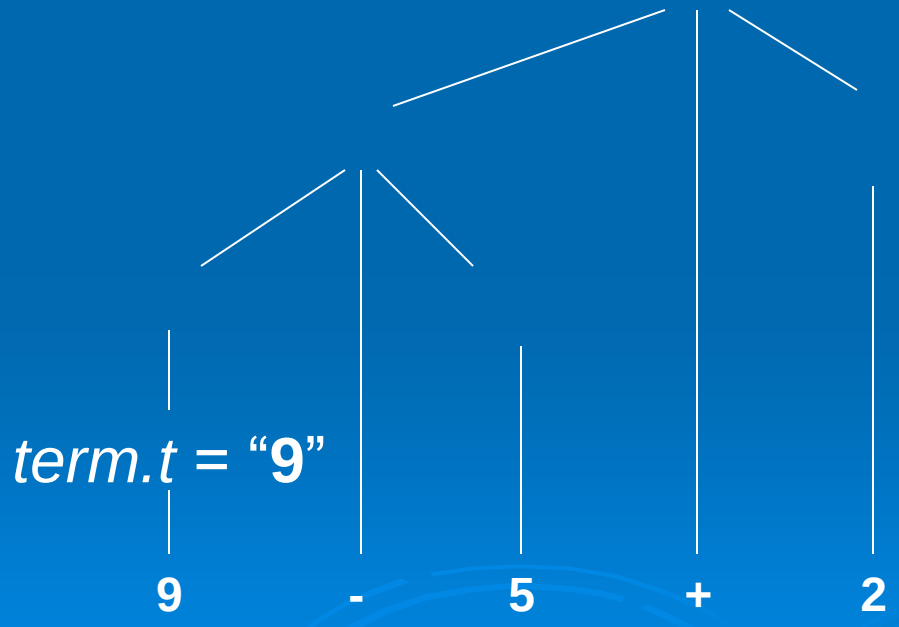


Gramatyka:
expr -> expr + term
| expr - term
| term
term->0,1,2,…,9

Translacja sterowana składnią

Atrybuty syntezowane

Wartości atrybutów w węzłach drzewa parsowania dla ciągu: 9-5+2, krok1

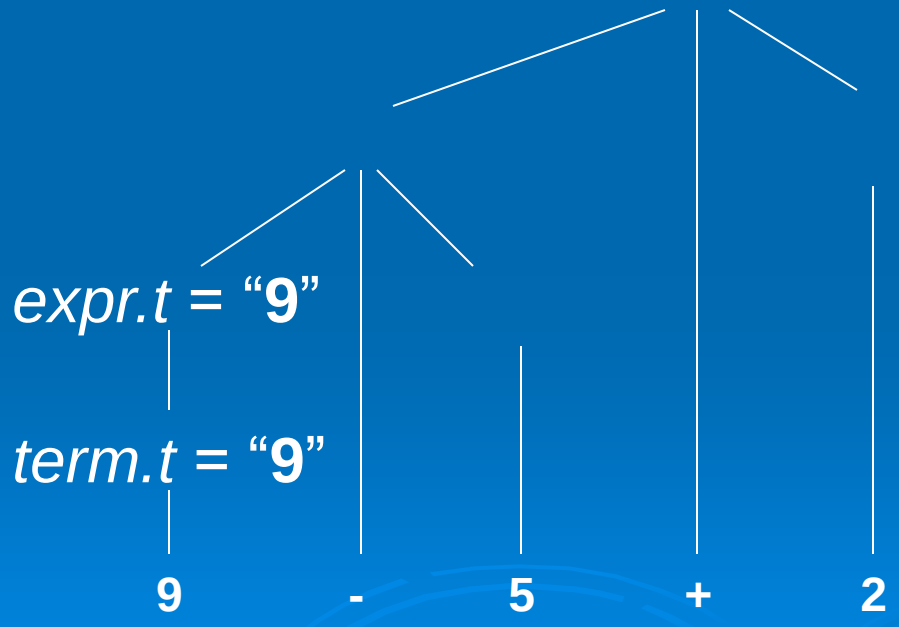


Gramatyka:
expr -> expr + term
I expr - term
I term
term->0,1,2,…,9

Translacja sterowana składnią

Atrybuty syntezowane

Wartości atrybutów w węzłach drzewa parsowania dla ciągu: 9-5+2, krok2

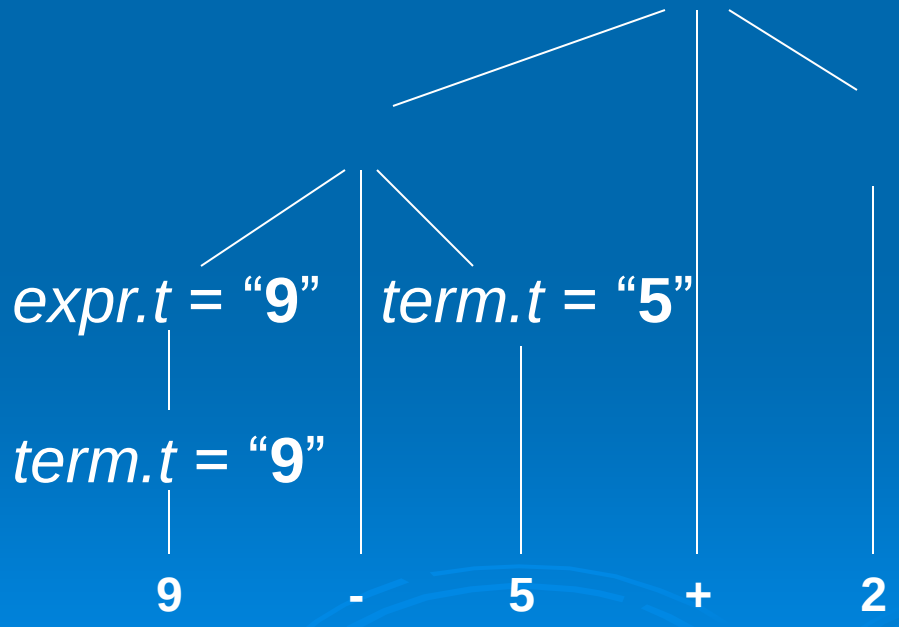


Gramatyka:
expr -> expr + term
I expr - term
I term
term->0,1,2,…,9

Translacja sterowana składnią

Atrybuty syntezowane

Wartości atrybutów w węzłach drzewa parsowania dla ciągu: 9-5+2, krok 3

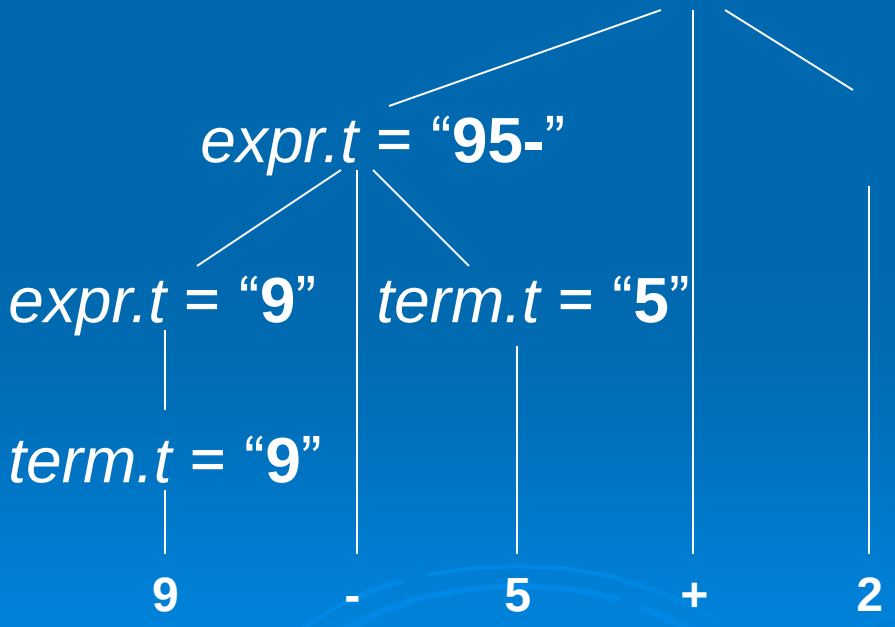


Gramatyka:
expr -> expr + term
I expr - term
I term
term->0,1,2,…,9

Translacja sterowana składnią

Atrybuty syntezowane

Wartości atrybutów w węzłach drzewa parsowania dla ciągu: 9-5+2, krok 4

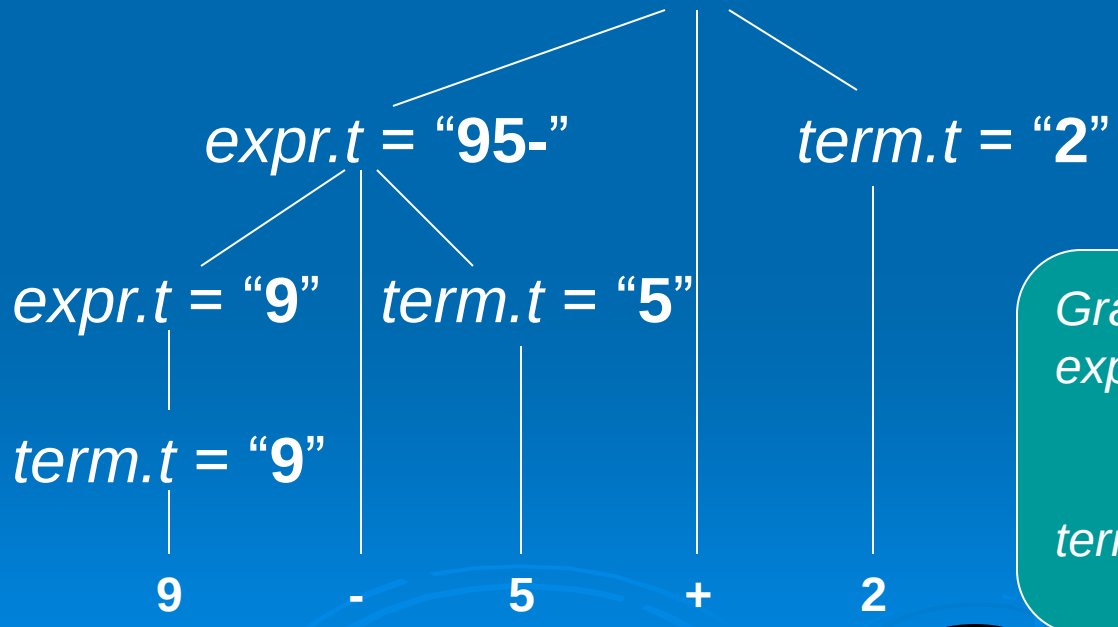


Gramatyka:
expr -> expr + term
I expr - term
I term
term->0,1,2,…,9

Translacja sterowana składnią

Atrybuty syntezowane

Wartości atrybutów w węzłach drzewa parsowania dla ciągu: 9-5+2, krok 5

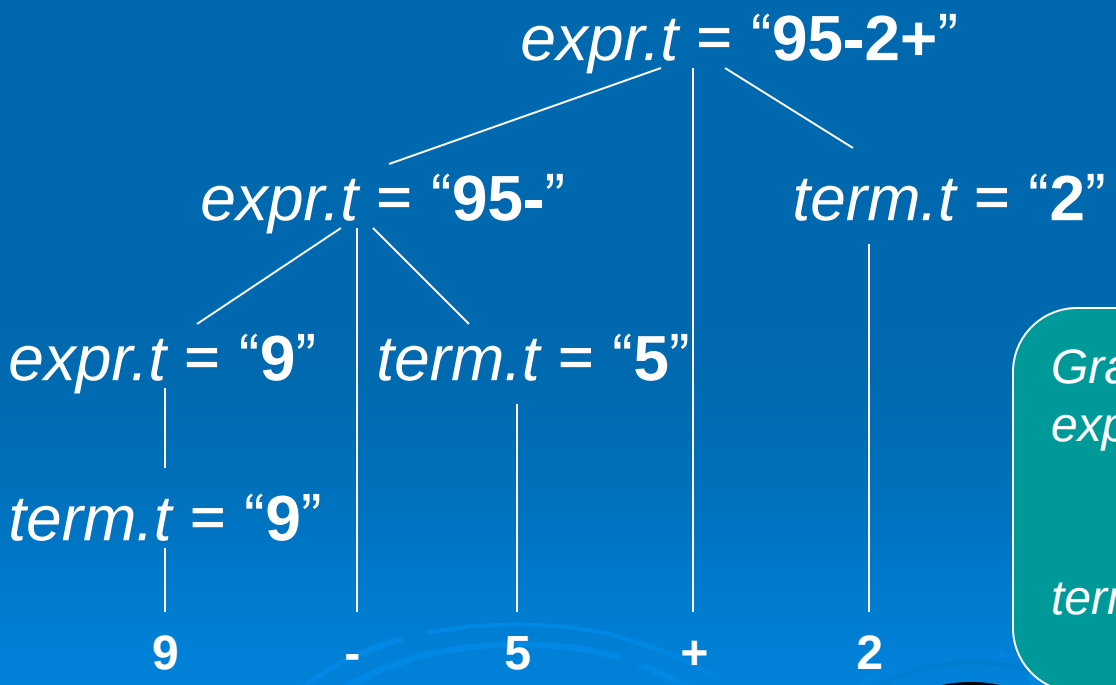


Gramatyka:
expr -> expr + term
I expr - term
I term
term->0,1,2,…,9

Translacja sterowana składnią

Atrybuty syntezowane

Wartości atrybutów w węzłach drzewa parsowania dla ciągu: 9-5+2, krok 6



Gramatyka:
expr -> expr + term
| expr - term
| term
term->0,1,2,…,9

Translacja sterowana składnią

Przechodzenie drzewa w głąb

- Przechodzenie przez drzewo jest używane do obliczania atrybutów oraz wykonania fragmentów kodu(akcji) w schemacie translacji.
- Przechodzenie drzewa zaczyna się od jego korzenia i dalej odwiedza sie każdy węzeł drzewa w pewnej kolejności.

Translacja sterowana składnią

Przechodzenie drzewa w głąb

- Przechodzenie drzewa w głąb (*depth-first traversal*) zaczyna się od korzenia i polega na odwiedzaniu dzieci każdego węzła w dowolnej kolejności, niekoniecznie od lewej do prawej.

Translacja sterowana składnią

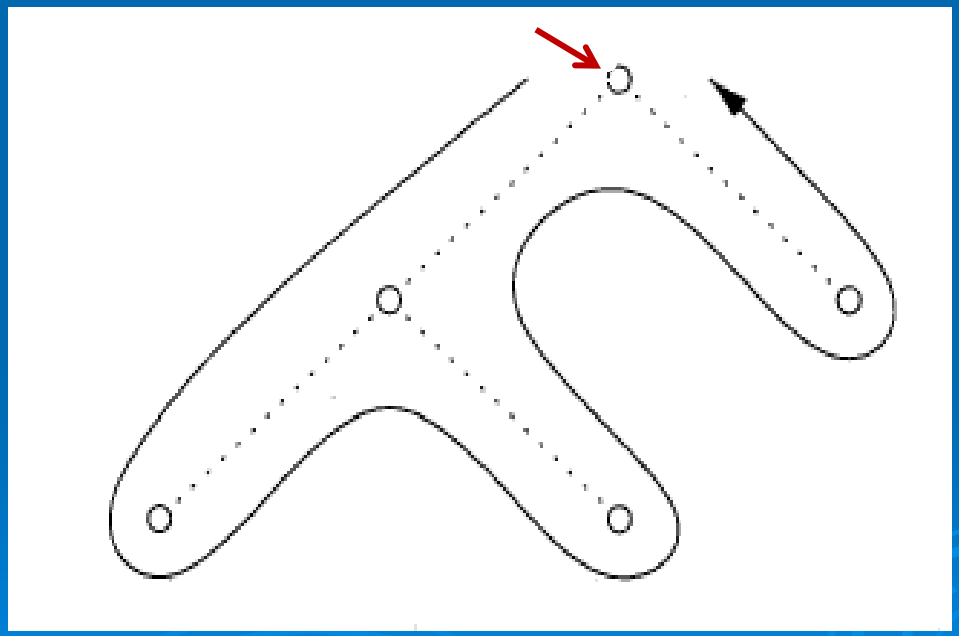
Przechodzenie drzewa w głąb

- Ono się nazywa "w głąb", ponieważ odwiedza nieodwiedzone dziecko węzła, gdy tylko może, więc odwiedza węzły jak najdalej od korzenia i tak szybko, jak jest to możliwe.

Translacja sterowana składnią

Przechodzenie drzewa w głąb

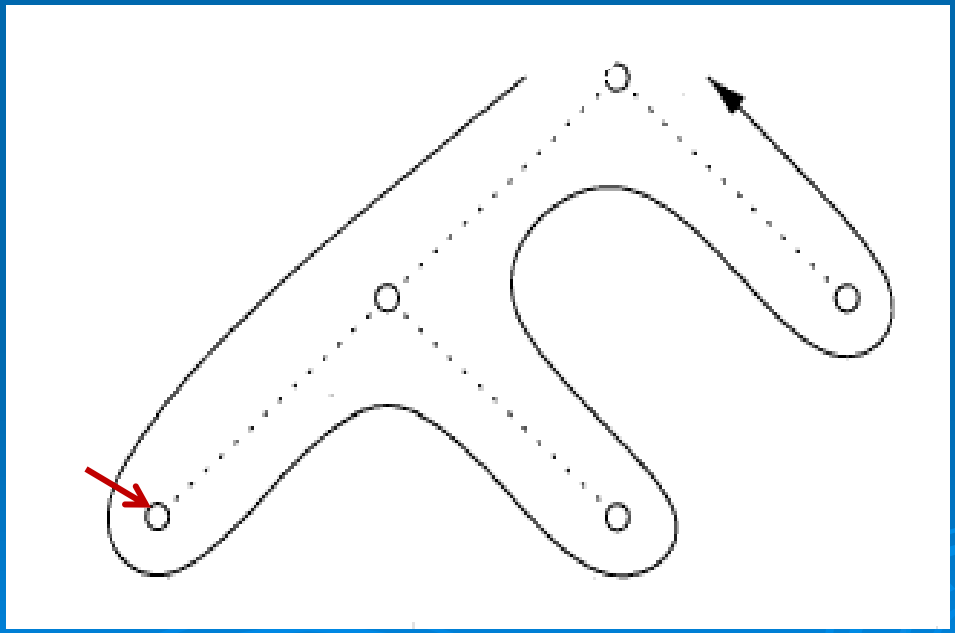
Przykład przechodzenia drzewa w głąb:



Translacja sterowana składnią

Przechodzenie drzewa w głąb

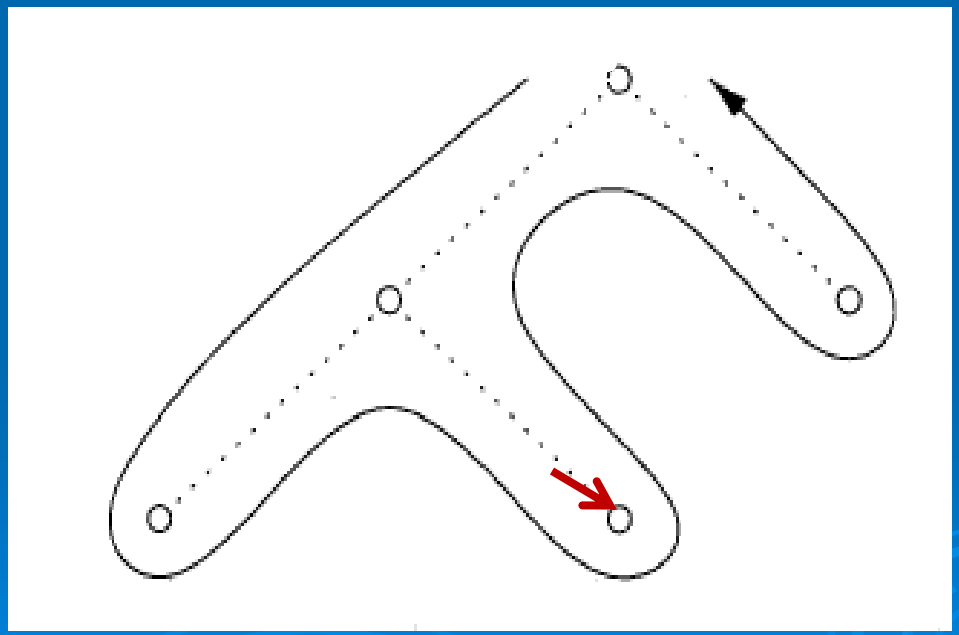
Przykład przechodzenia drzewa w głąb:



Translacja sterowana składnią

Przechodzenie drzewa w głąb

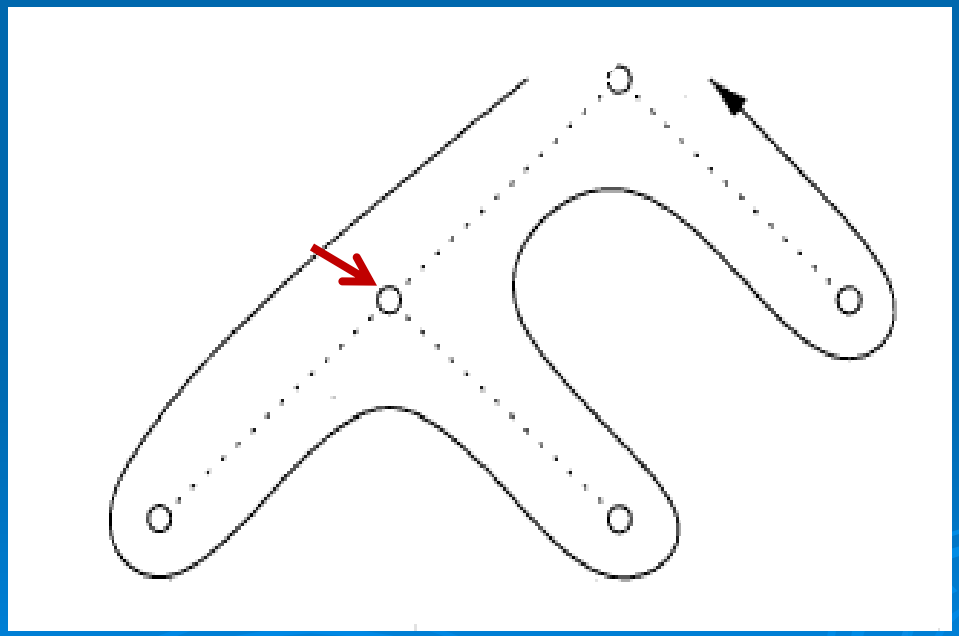
Przykład przechodzenia drzewa w głąb:



Translacja sterowana składnią

Przechodzenie drzewa w głąb

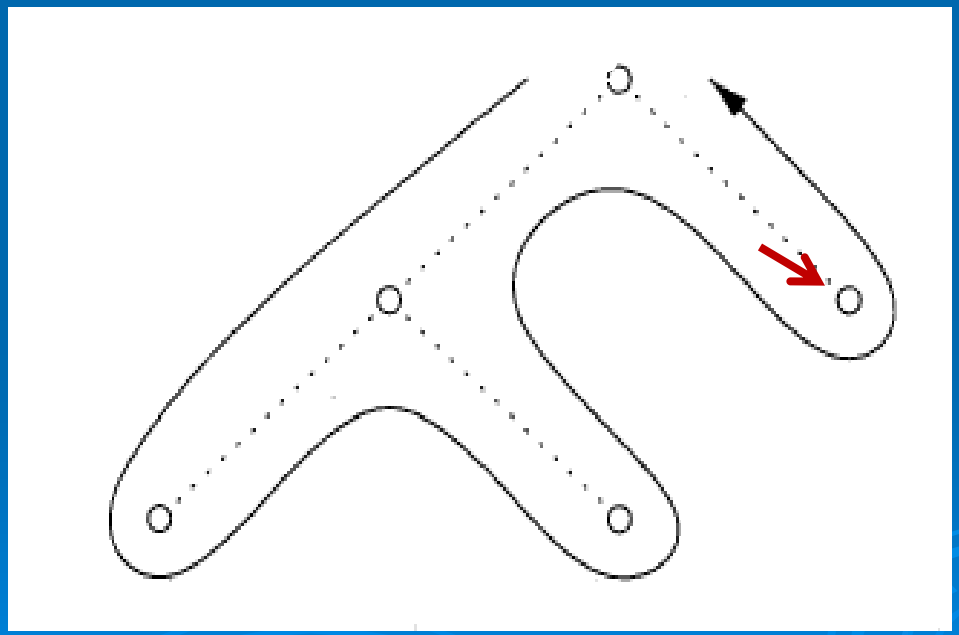
Przykład przechodzenia drzewa w głąb:



Translacja sterowana składnią

Przechodzenie drzewa w głąb

Przykład przechodzenia drzewa w głąb:



Translacja sterowana składnią

Przechodzenie drzewa w głąb

- Definicja sterowana składnią nie narzuca żadnej konkretnej kolejności do obliczenia atrybutów w drzewie parsowania;

  każda kolejność, która oblicza atrybut a

  po wszystkich innych atrybutach, od których a zależy, jest akceptowalna.

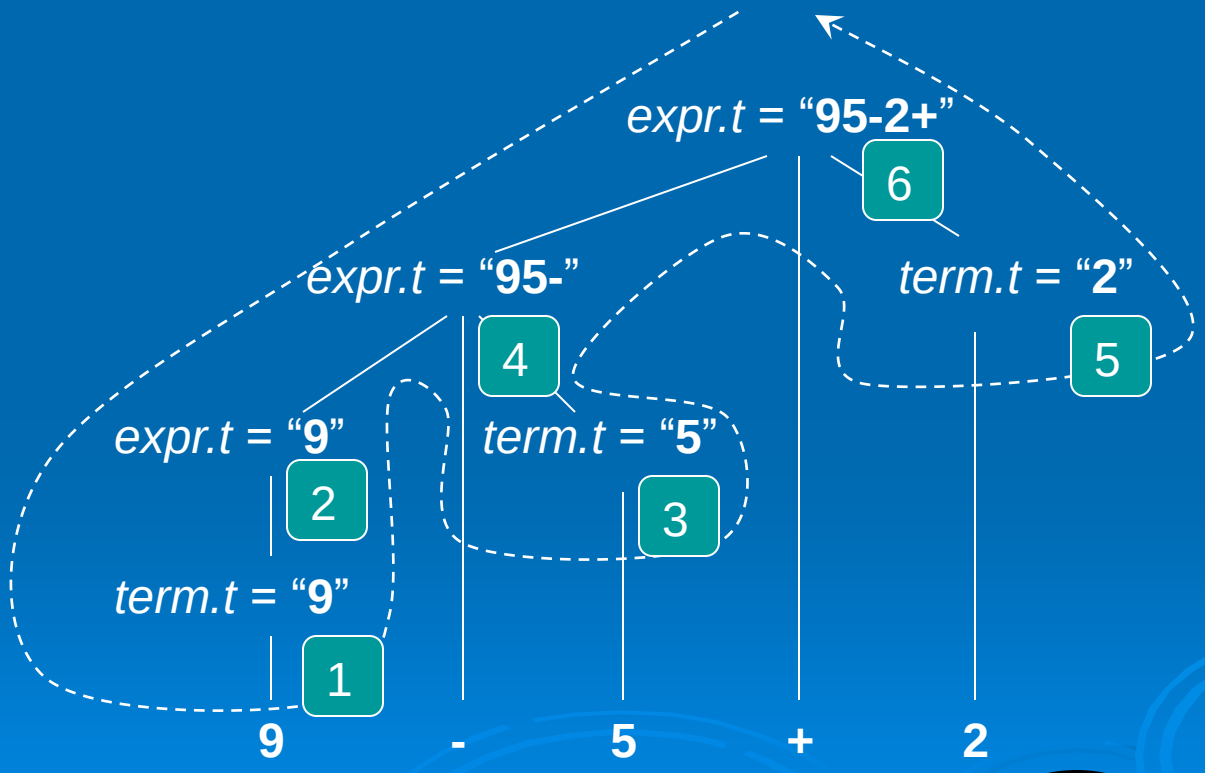
Translacja sterowana składnią

Przechodzenie drzewa w głąb

```
procedure visit(node N)
 {
   for (each child C of N, from left to right)
        visit (C) ;
    zastosuj reguły semantyczne w węzle N;
 }
```

Jest rekurencyjna i zapewnia, że w pierwszej kolejności wartości semantyczne będą obliczone dla liści drzewa

Przechodzenie drzewa w głąb przykład



Translacja sterowana składnią

Schemat translacji

- Schemat translacji sterowany składnią jest to notacja dla określenia translacji, która powstaje po dołączeniu fragmentów kodu do produkcji w gramatyce.

Translacja sterowana składnią

Schemat translacji

- Fragmenty kodu, dodane do produkcji, nazywamy akcjami semantycznymi.

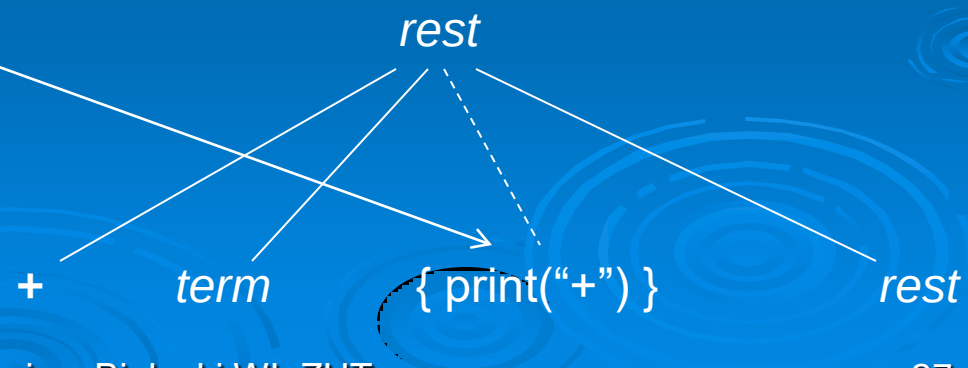
Schemat translacji

- Pozycja, gdzie akcja ma zostać wykonana jest pokazana przez umieszczenie jej w klamrach w prawej stronie produkcji:

  rest → **+** *term* { print(“+”) } *rest*

Akcja semantyczna

Dodatkowy węzeł reprezentuje akcję semantyczną, jest połączony linią przerywaną z węzłem, odpowiadającym głowie (lewej stronie)



Translacja sterowana składnią

Schemat translacji

Akcje do translacji na notację postfiksową:

expr -> expr1 + term	{print('+')}
expr -> expr1- term	{print('-')}
expr -> term	
term -> 0	{print ('0')}
term -> 1	{print (' 1')}
...	
term -> 9	{print ('9')}

Akcje semantyczne

Translacja sterowana składnią

Schemat translacji

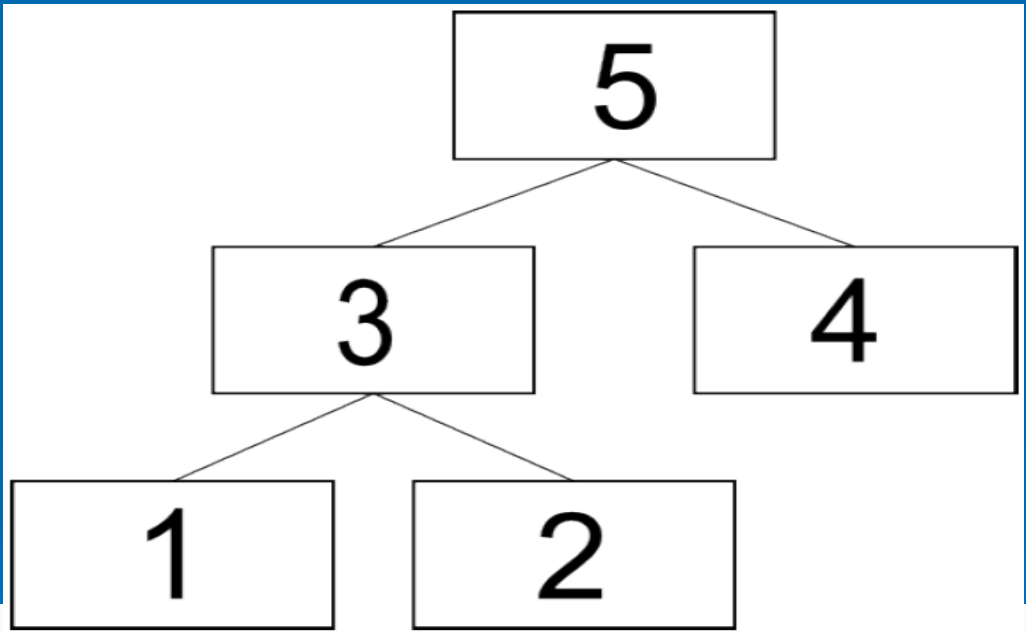
- Implementacja schematu translacji musi zapewnić, że akcje semantyczne

  zostaną wykonane w kolejności, w jakiej pojawiają się one w trakcie przechodzenia post-order.

Przechodzenie drzewa

Postorder –Postfiksowe – Wsteczne

Drzewa: poszukiwanie postorder



Numery przy wierzchołkach oznaczają kolejność ich odwiedzania

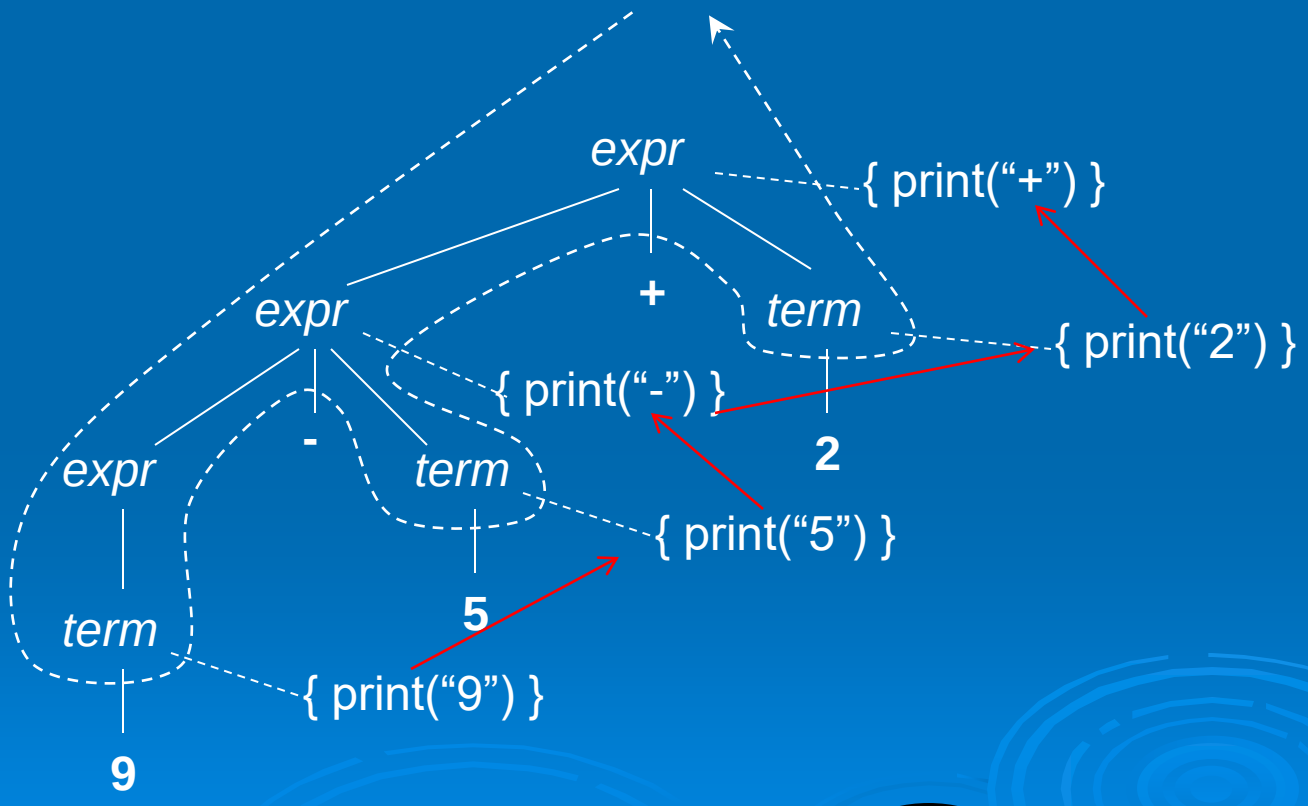
- Zanim odwiedzimy dany wierzchołek, odwiedzimy wszystkich jego potomków.
- Poruszamy się od najniższej generacji w górę.

Translacja sterowana składnią

Schemat translacji

- Implementacja nie musi w rzeczywistości skonstruować drzewa parsowania o ile zapewnia ona, że akcje semantyczne są wykonywane jak gdyby skonstruowaliśmy drzewo syntaktyczne i następnie akcje są wykonywane zgodnie z przechodzeniem post-order.

Schemat translacji



Translacja **9-5+2** do postaci **95-2+**

Prosty kalkulator

Definicja sterowana składnią:

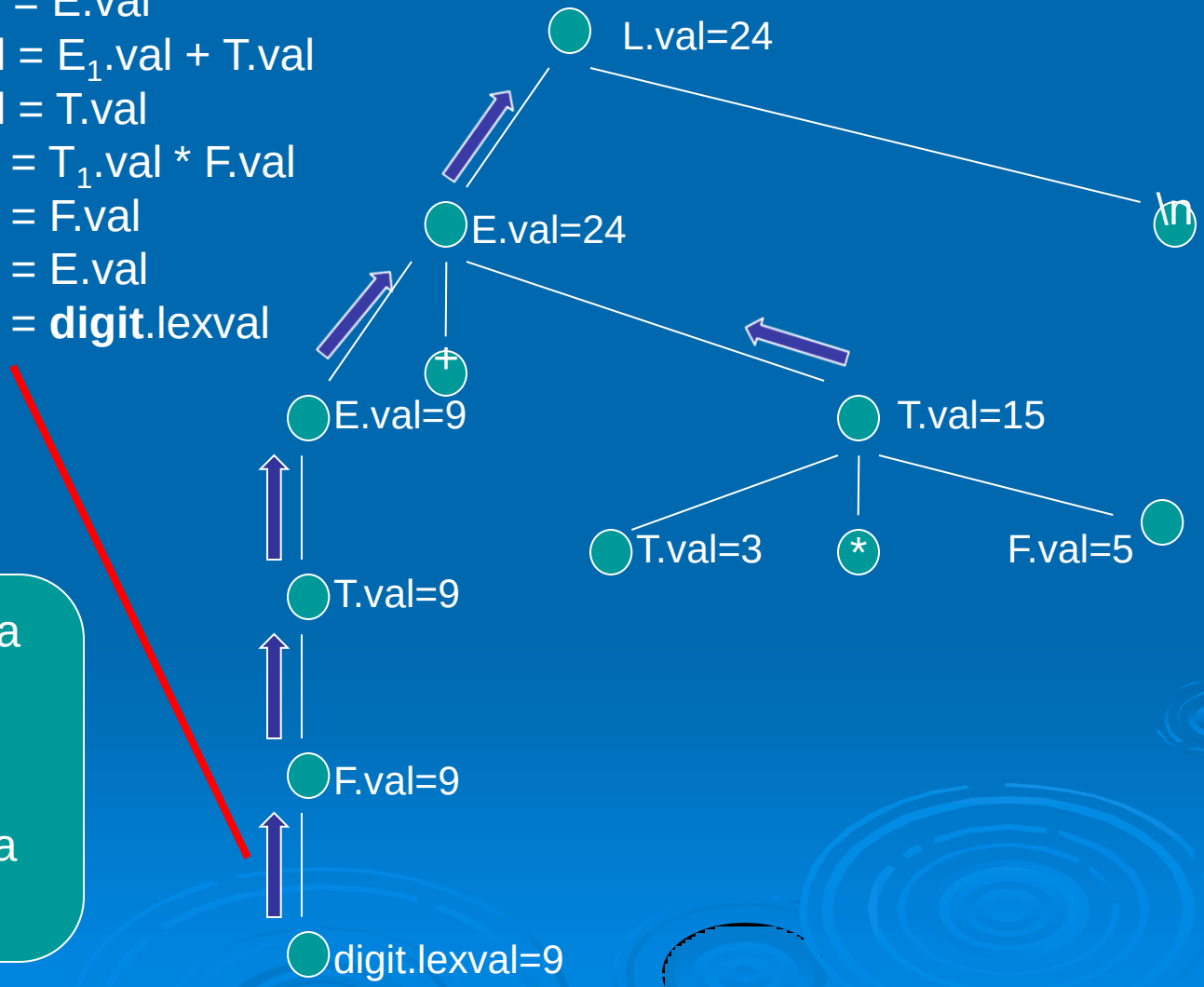
L ::= E \n	L.val = E.val
E ::= E_1 + T	E.val = E_1.val + T.val
E ::= T	E.val = T.val
T ::= T_1 * F	T.val = T_1.val * F.val
T ::= F	T.val = F.val
F ::= (E)	F.val = E.val
F ::= **digit**	F.val = **digit**.lexval

Produkcje

Akcje semantyczne

Obliczenie: “9+3*5\n”

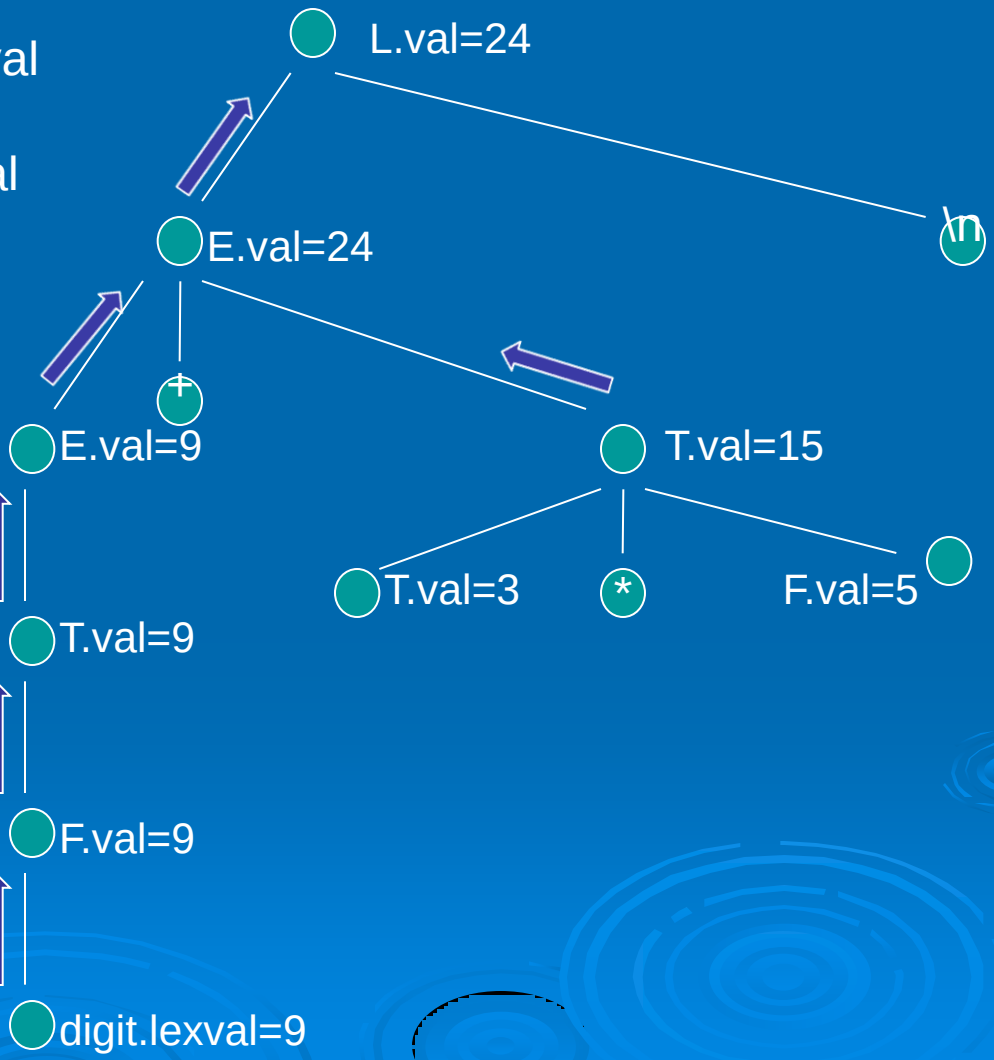
L	::= E \n	L.val = E.val
E	::= E_1 + T	E.val = E_1.val + T.val
E	::= T	E.val = T.val
T	::= T_1 * F	T.val = T_1.val * F.val
T	::= F	T.val = F.val
F	::= (E)	F.val = E.val
F	::= **digit**	F.val = **digit**.lexval



Uwaga: Produkcja
F ::= (E)
nie została wykorzystana do tworzenia drzewa parsowania

Obliczenie: “9+3*5\n”

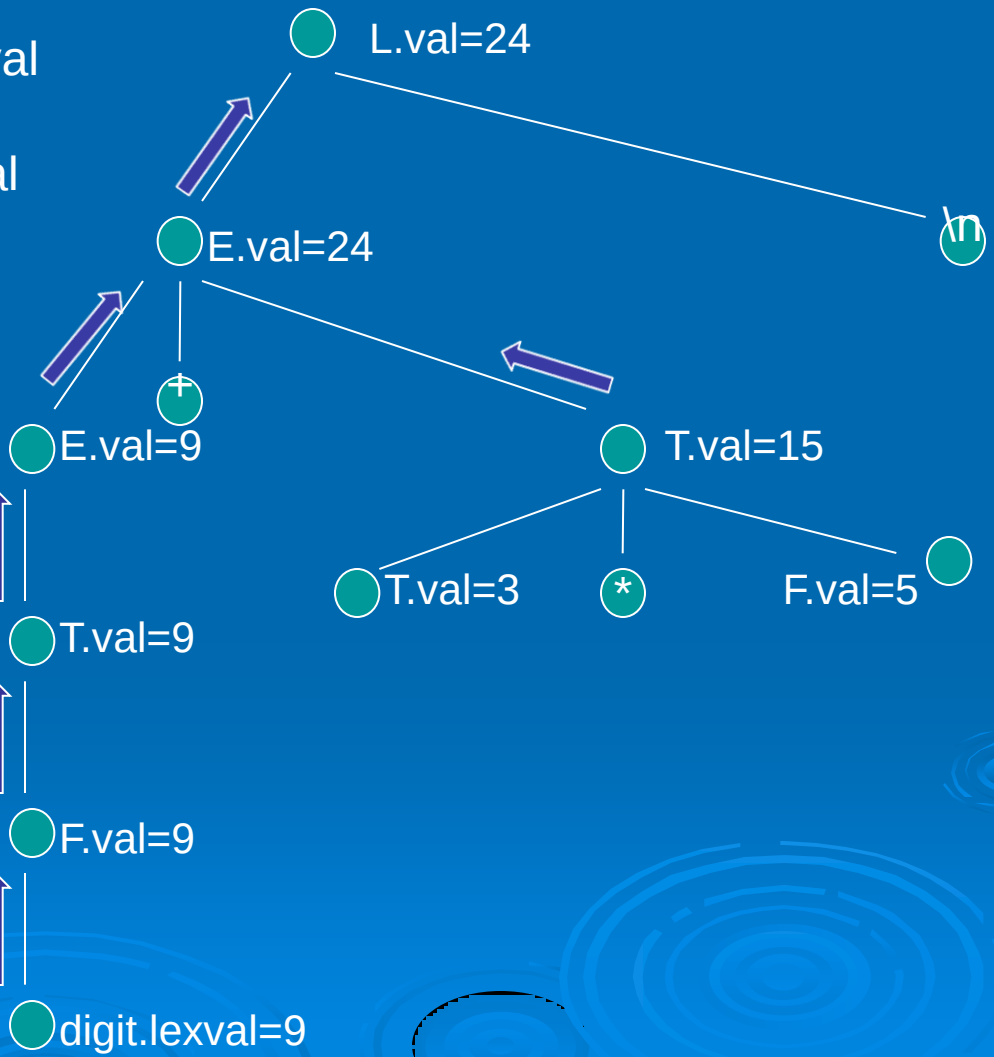
L	::= E \n	L.val = E.val
E	::= E_1 + T	E.val = E_1.val + T.val
E	::= T	E.val = T.val
T	::= T_1 * F	T.val = T_1.val * F.val
T	::= F	T.val = F.val
F	::= (E)	F.val = E.val
F	::= **digit**	F.val = **digit**.lexval



Uwaga: Produkcja
F ::= (E)
nie została wykorzystana do tworzenia drzewa parsowania

Obliczenie: “9+3*5\n”

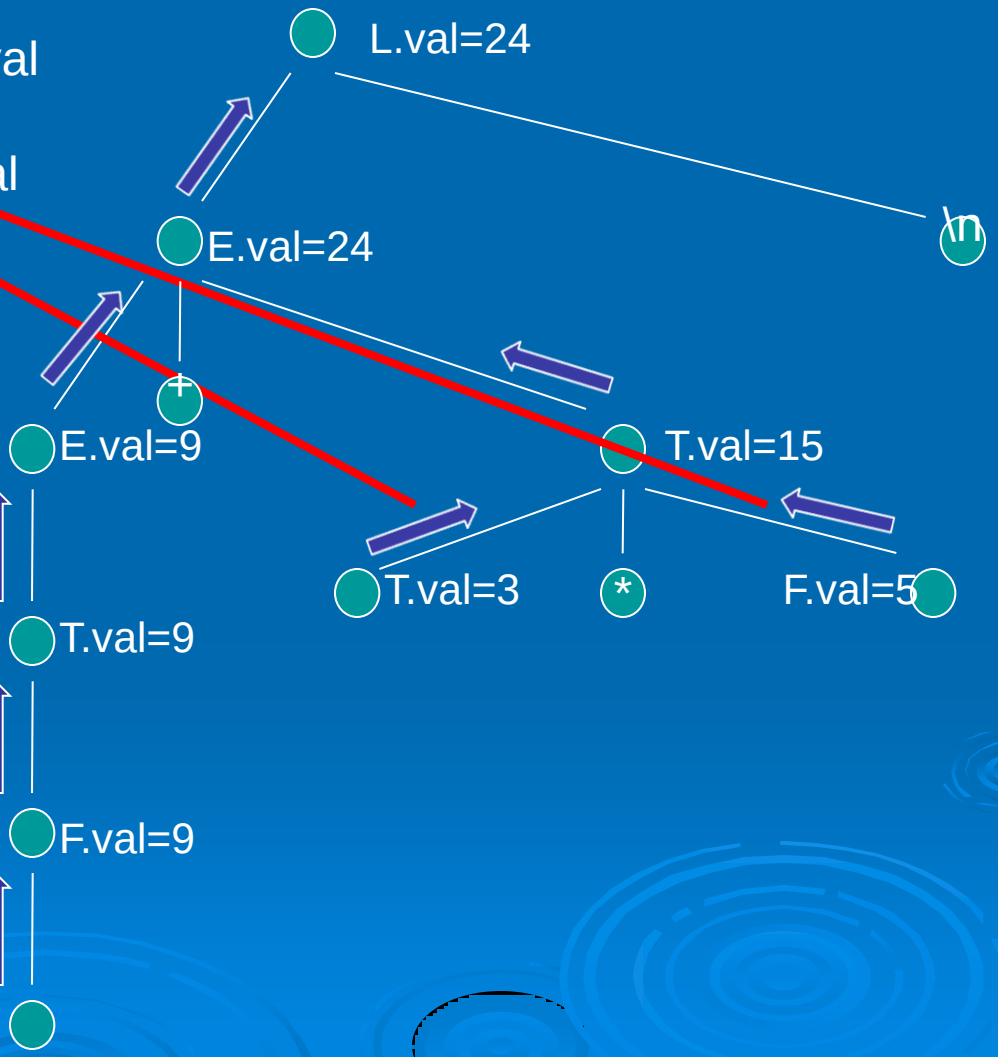
L	::= E \n	L.val = E.val
E	::= E_1 + T	E.val = E_1.val + T.val
E	::= T	E.val = T.val
T	::= T_1 * F	T.val = T_1.val * F.val
T	::= F	T.val = F.val
F	::= (E)	F.val = E.val
F	::= **digit**	F.val = **digit**.lexval



Uwaga: Produkcja
F ::= (E)
nie została wykorzystana do tworzenia drzewa parsowania

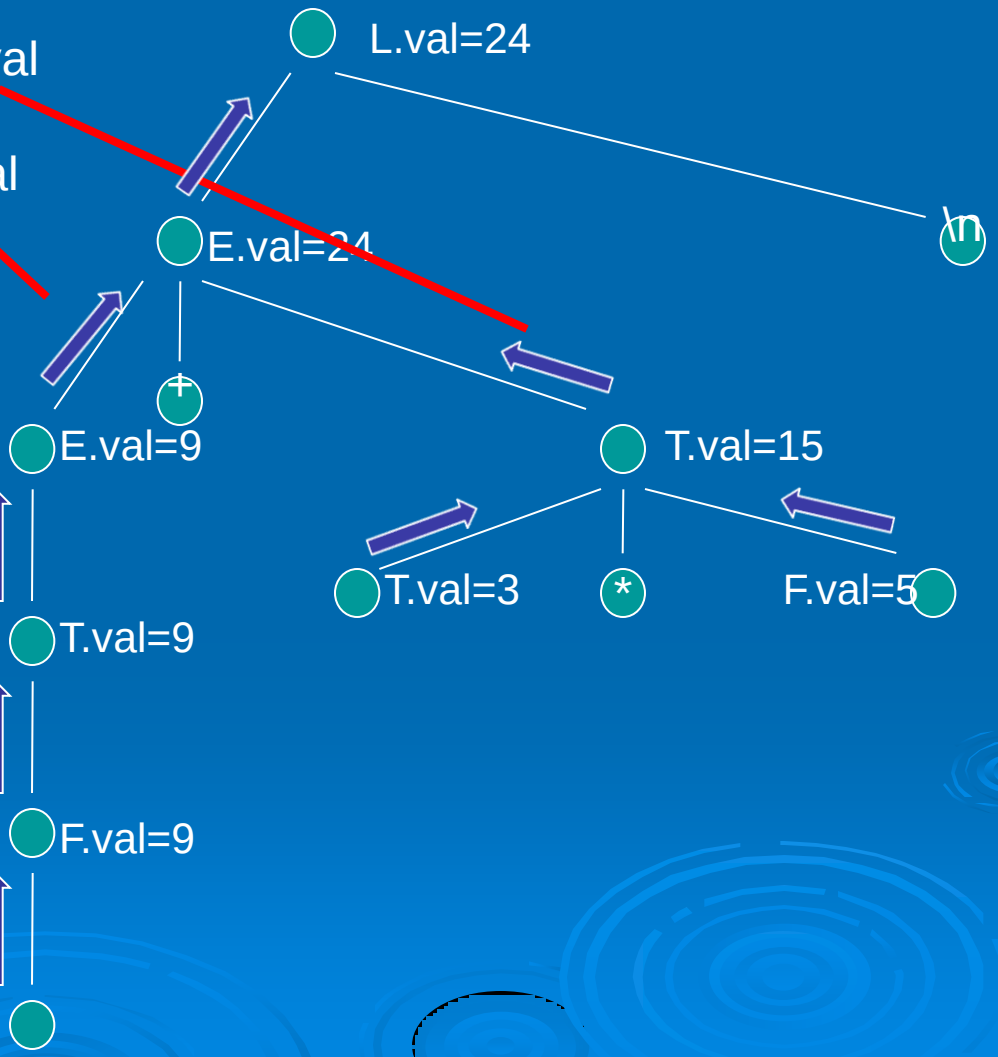
Obliczenie: “9+3*5\n””

L ::= E \n	L.val = E.val
E ::= E_1 + T	E.val = E_1.val + T.val
E ::= T	E.val = T.val
T ::= T_1 * F	T.val = T_1.val * F.val
T ::= F	T.val = F.val
F ::= (E)	F.val = E.val
F ::= **digit**	F.val = **digit**.lexval



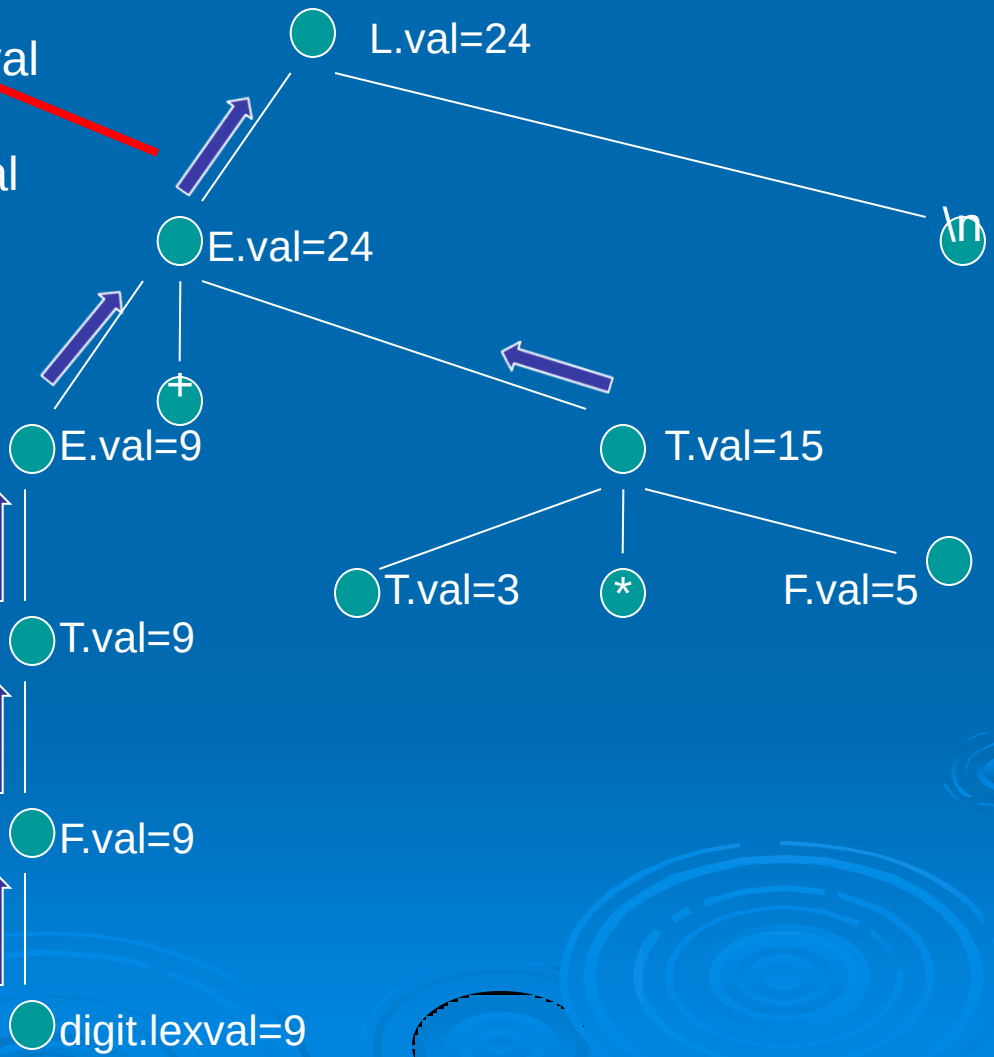
Obliczenie: “9+3*5\n””

L	::= E \n	L.val = E.val
E	::= E_1 + T	E.val = E_1.val + T.val
E	::= T	E.val = T.val
T	::= T_1 * F	T.val = T_1.val * F.val
T	::= F	T.val = F.val
F	::= (E)	F.val = E.val
F	::= **digit**	F.val = **digit**.lexval



Obliczenie: “9+3*5\n”

L	::= E \n	L.val = E.val
E	::= E_1 + T	E.val = E_1.val + T.val
E	::= T	E.val = T.val
T	::= T_1 * F	T.val = T_1.val * F.val
T	::= F	T.val = F.val
F	::= (E)	F.val = E.val
F	::= **digit**	F.val = **digit**.lexval



Dziękuję za uwagę