Metody Kompilacji
Wykład 2
Gramatyki

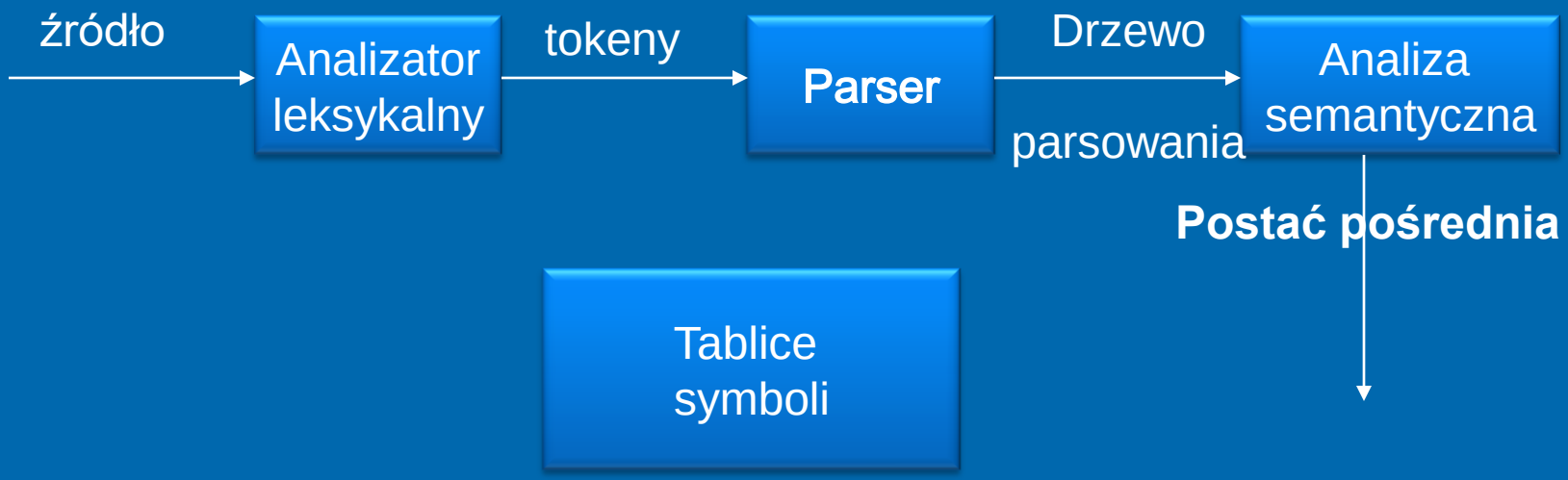
Składnia języka

- Składnia języka programowania opisuje właściwą strukturę programu, natomiast semantyka języka określa co program robi – czyli jaki jest jego sens.

Struktura kompilatora

- Każdy kompilator ma przód i tył, struktura przodu kompilatora jest pokazana na następnym slajdzie.

Struktura przodu kompilatora



(Są wykorzystywane przez wszystkie fazy kompilacji)

Języki programowania

1 + 2 * 3 = 7

1 + * 2 3 = ???

Czy te ciągi są poprawnie zbudowanym wyrażeniem arytmetycznym?

Języki programowania

1 + 2 * 3 = 7

1 + * 2 3 = ???

Czy te ciągi są poprawnie zbudowanym wyrażeniem arytmetycznym?

Żeby odpowiedzieć na pytanie potrzebujemy gramatyki, czyli zbioru produkcji

Pojęcie produkcji

- Produkcja jest to para uporządkowana, na przykład

$$S \rightarrow 1$$

Gramatyka

$S \rightarrow A\ B$

$A \rightarrow 1$

$A \rightarrow A\ 1$

$B \rightarrow 0$

$B \rightarrow B\ 0$

Gramatyka jest to zbór produkcji

Gramatyka

$S \rightarrow A\ B$

$A \rightarrow 1$

$A \rightarrow A\ 1$

$B \rightarrow 0$

$B \rightarrow B\ 0$

- S jest symbolem początkowym

Gramatyka

$$S \rightarrow A\ B$$
$$A \rightarrow 1$$
$$A \rightarrow A\ 1$$
$$B \rightarrow 0$$
$$B \rightarrow B\ 0$$

Symbole nieterminalne
N = {S, A, B}.
Występują po prawej stronie produkcji,
mogą wystąpić również po lewej stronie produkcji(ale nie muszą)

Gramatyka

$S \rightarrow A\ B$

$A \rightarrow 1$

$A \rightarrow A\ 1$

$B \rightarrow 0$

$B \rightarrow B\ 0$

Symbole terminalne T = {0, 1} –występują tylko po prawej stronie produkcji

Wyprowadzenia

1) $S \rightarrow A\ B$
2) $A \rightarrow 1$
3) $A \rightarrow A\ 1$
4) $B \rightarrow 0$
5) $B \rightarrow B\ 0$

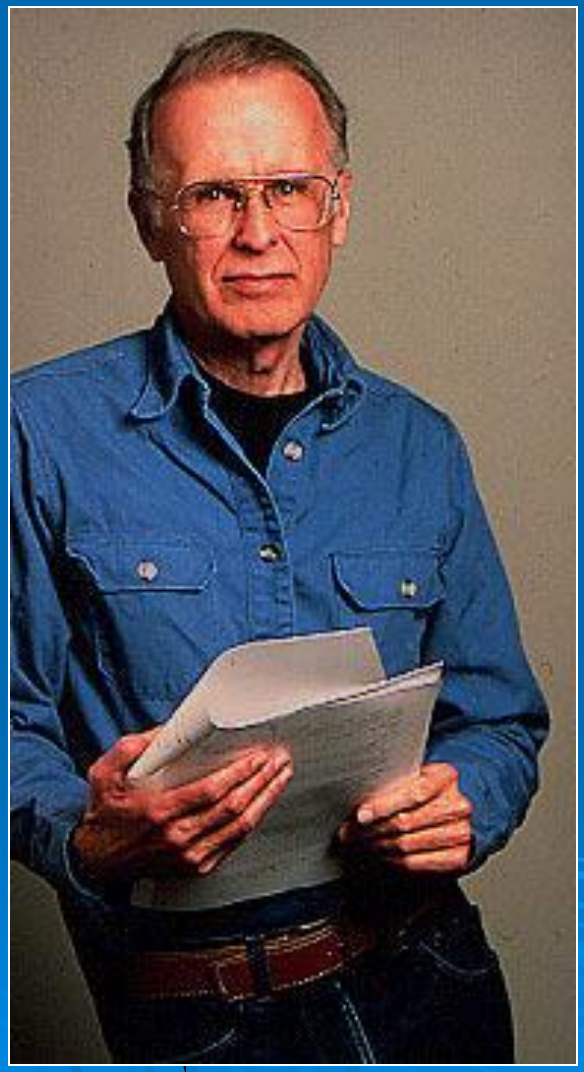
Wyprowadzenia:

$S \overset{1}{\Rightarrow} A\ B \overset{2}{\Rightarrow} 1\ B \overset{3}{\Rightarrow} 1\ 0$

$S \overset{+}{\Rightarrow} 1\ 0$ Z S można wyprowadzić 10 stosując jedną lub więcej produkcji

Specyfikacja BNF

Backus



Naur

Specyfikacja BNF

- Backus-Naur Form (BNF) jest formą używaną do wyrażenia gramatyk bezkontekstowych.
- Specyfikacja BNF jest to zbiór produkcji o postaci:

symbol -> expression

gdzie *symbol* jest to nieterminal, natomiast *expression* jest to sekwencja jednej lub większej liczby symboli terminalnych i/lub nieterminalnych.

Specyfikacja BNF

- Większą liczbę sekwencji oddzielamy kreską pionową '|', wskazując wybór.
- Symbole, które nigdy nie pojawiają się po lewej stronie produkcji są to terminale (są pogrubione na piśmie).
- Symbole pojawiające się po lewej stronie produkcji są to nieterminale i są pisane kursywą.

Specyfikacja BNF

Przykład produkcji:

stmt -> **if** (*expr*) *stmt* **else** *stmt*

Gramatyka bezkontekstowa

Gramatyka bezkontekstowa (*A context-free grammar*)

zawiera:

1. Zbiór symboli terminalnych(terminali). Terminale są to elementarne symbole języka zdefiniowanego przez gramatykę.

Gramatyka bezkontekstowa

- 2. Zbiór symboli nieterminalnych (zmienne syntaktyczne, nieterminale).
- Każdy nieterminal reprezentuje sekwencję terminali w sposób, który poznamy na kolejnych wykładach.

Gramatyka bezkontekstowa

3. Zbiór produkcji, gdzie każda produkcja składa się z nieterminala, zwanego głową lub lewą stroną produkcji, strzałki oraz sekwencji terminali i/lub nieterminali, nazywanej ciałem lub prawą stroną produkcji.

Gramatyka bezkontekstowa

4. Jeden wyznaczony symbol nieterminalny zwany symbolem startowym.

Gramatyka bezkontekstowa

Gramatyka jest to skończony zbiór produkcji, w którym lewa strona pierwszej produkcji wskazuje symbol startowy.

Gramatyka bezkontekstowa

Przykład gramatyki:

list -> list + digit (1)

list -> list – digit (2)

list -> digit (3)

digit -> 0 |1| 2| 3| 4| 5| 6| 7| 8| 9 (4)

Gramatyka bezkontekstowa

Ciała trzech produkcji, których lewa strona jest tym samym symbolem *list*, można pogrupować:

list -> list + digit | list - digit | digit

Gramatyka bezkontekstowa

- Według przedstawionej definicji symbole terminalne są jak niżej:

+, -, 0,1, 2, 3, 4, 5, 6, 7, 8, 9

Gramatyka bezkontekstowa

- Nieterminale to są :

  list i *digit*

- Symbolem startowym jest *list* .

Przykład gramatyki

$G = < \{list, digit\}, \{\mathbf{+}, \mathbf{-}, \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}, \mathbf{8}, \mathbf{9}\}, P, list >$

Z produkcjami $P =$

$list \rightarrow list \; \mathbf{+} \; digit$

$list \rightarrow list \; \mathbf{-} \; digit$

$list \rightarrow digit$

$digit \rightarrow \mathbf{0} \mid \mathbf{1} \mid \mathbf{2} \mid \mathbf{3} \mid \mathbf{4} \mid \mathbf{5} \mid \mathbf{6} \mid \mathbf{7} \mid \mathbf{8} \mid \mathbf{9}$

Gramatyka bezkontekstowa

Ciąg symboli (napis) jest to sekwencja składająca się na zero lub więcej symboli.

Ciąg, który nie zawiera żadnego symbolu, jest nazywany napisem pustym, zapisujemy go jako ε.

Gramatyka bezkontekstowa

Wyprowadzenia

Korzystając z gramatyki, wyprowadzamy napisy zaczynając zawsze od symbolu startowego i wielokrotnie zastępujemy pojedynczy nieterminal przez prawą stronę produkcji, której lewa strona jest tym nieterminalem.

Gramatyka bezkontekstowa

Wyprowadzenia

Dla gramatyki:

list -> list + digit | list - digit | digit

digit -> 0 |1| 2| 3| 4| 5| 6| 7| 8| 9

Możemy wyprowadzić:

list ⟶ *digit* ⟶ 0

list ⟶ *list + digit* ⟶ *digit + digit* ⟶

1 + *digit* ⟶ *1+2*

Gramatyka bezkontekstowa

- Parsowanie ma na wejściu ciąg terminali i zastanawia się, jak wyprowadzić ten ciąg z symbolu startowego gramatyki;

  jeśli nie można wyprowadzić takiego ciągu, to jest raportowany błąd składni.

Gramatyka bezkontekstowa

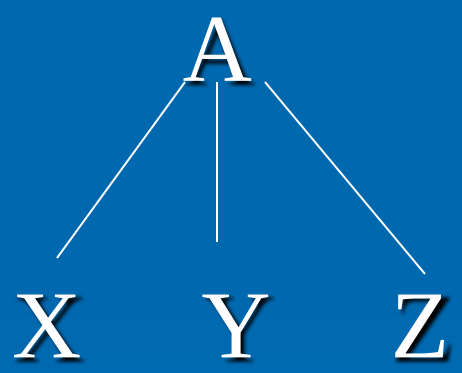
- Drzewo parsowania pokazuje obrazowo w jaki sposób z symbolu startowego gramatyki można wyprowadzić zdanie wejściowe.
- Dla produkcji:

  A -> XYZ,

  drzewo parsowania ma postać:

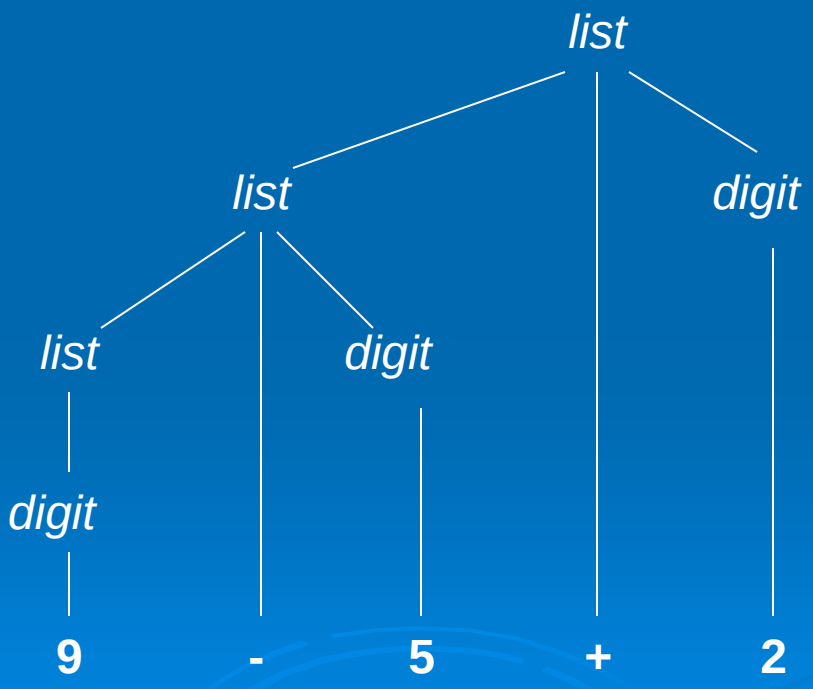
Gramatyka bezkontekstowa

Drzewo parsowania



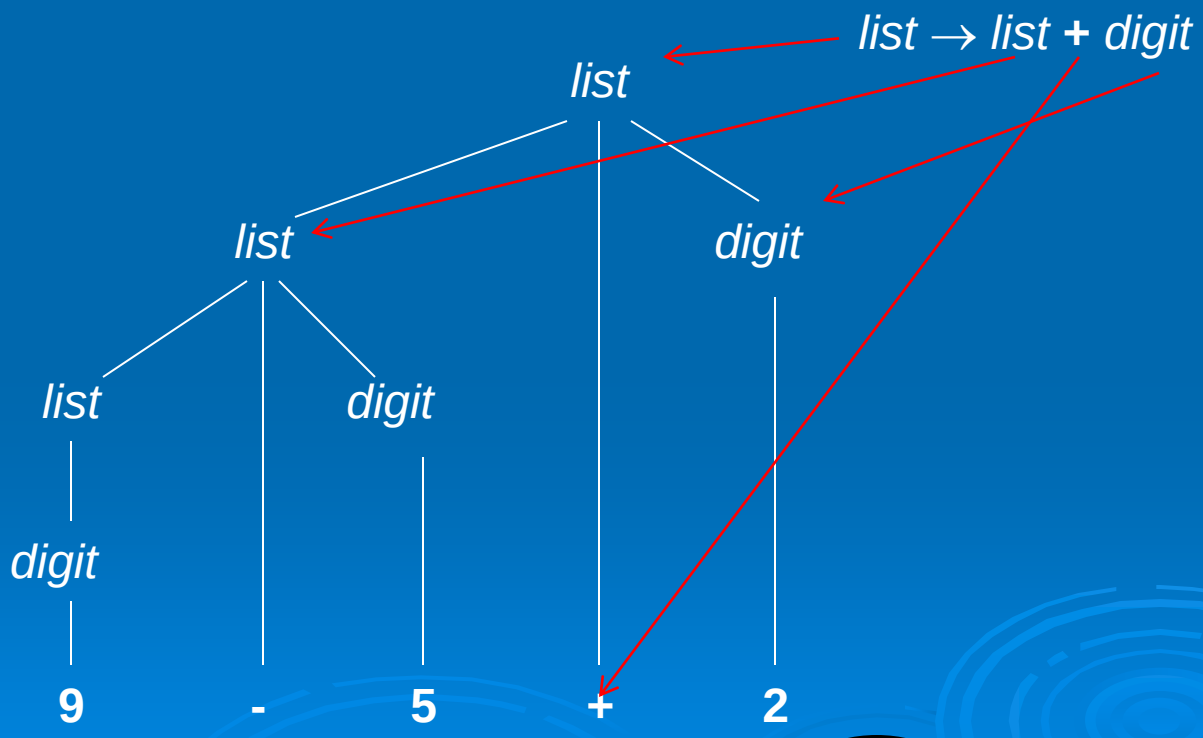
Drzewo parsowania

Dla napisu 9-5+2 i gramatyki *G* drzewo parsowania ma postać



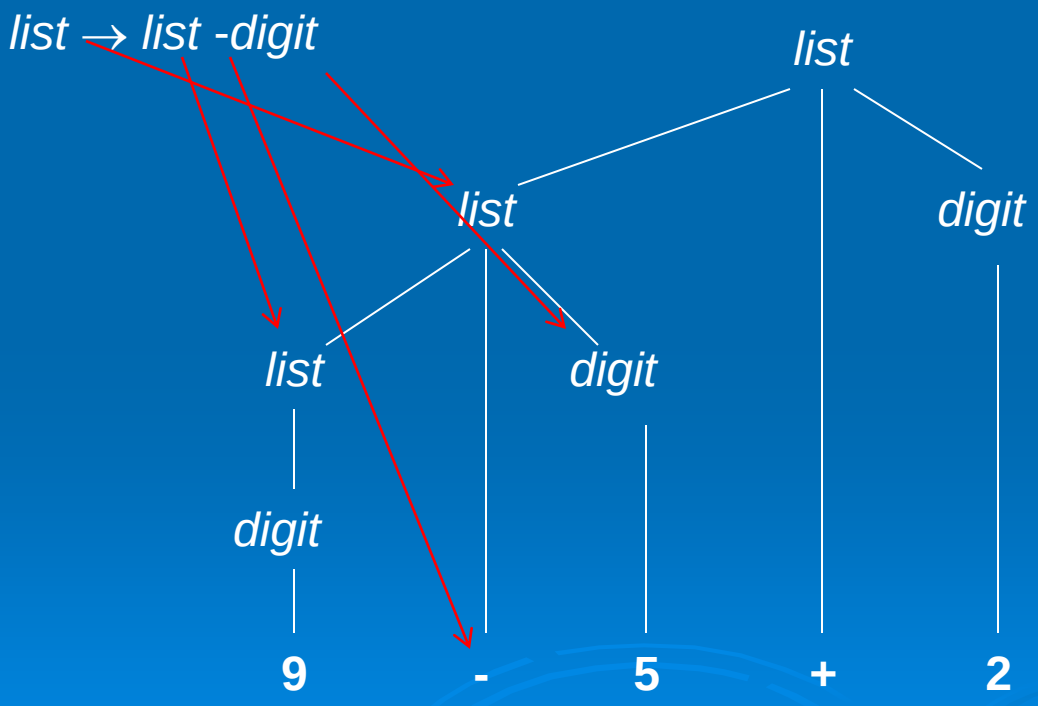
Drzewo parsowania

Dla napisu 9-5+2 i gramatyki *G* drzewo parsowania ma postać: krok 1



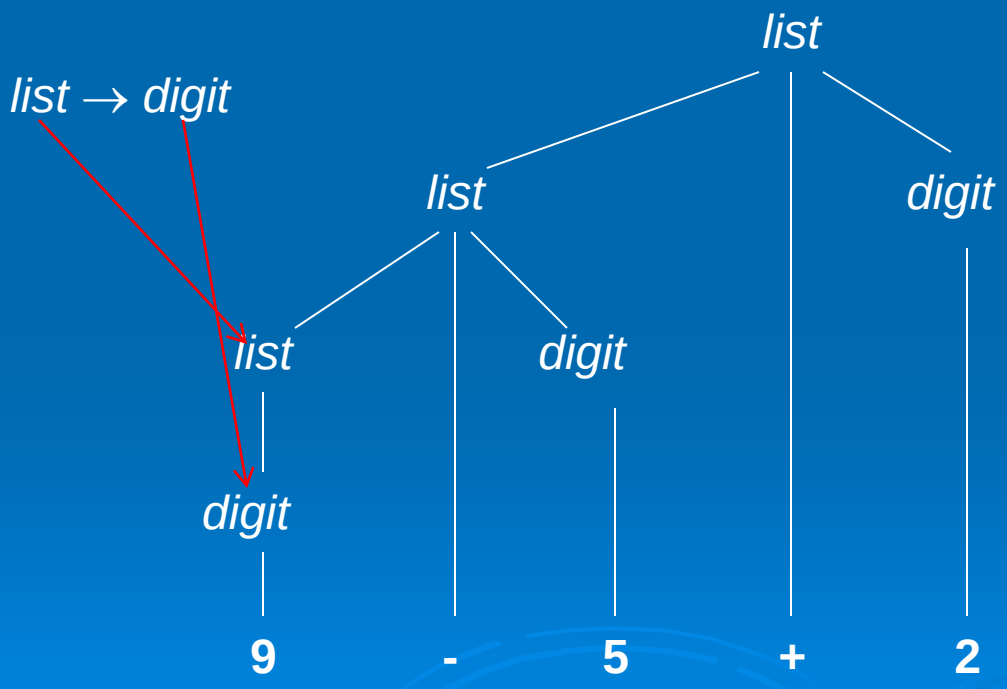
Drzewo parsowania

Dla napisu 9-5+2 i gramatyki *G* drzewo parsowania ma postać: krok 2



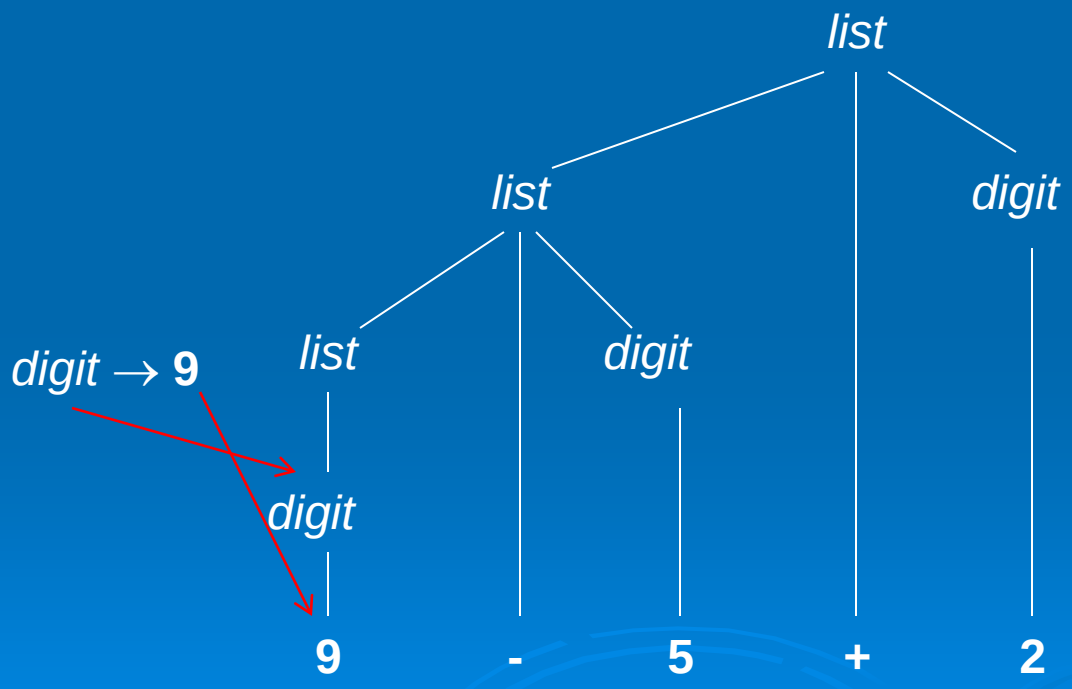
Drzewo parsowania

Dla napisu 9-5+2 i gramatyki *G* drzewo parsowania ma postać: krok 3



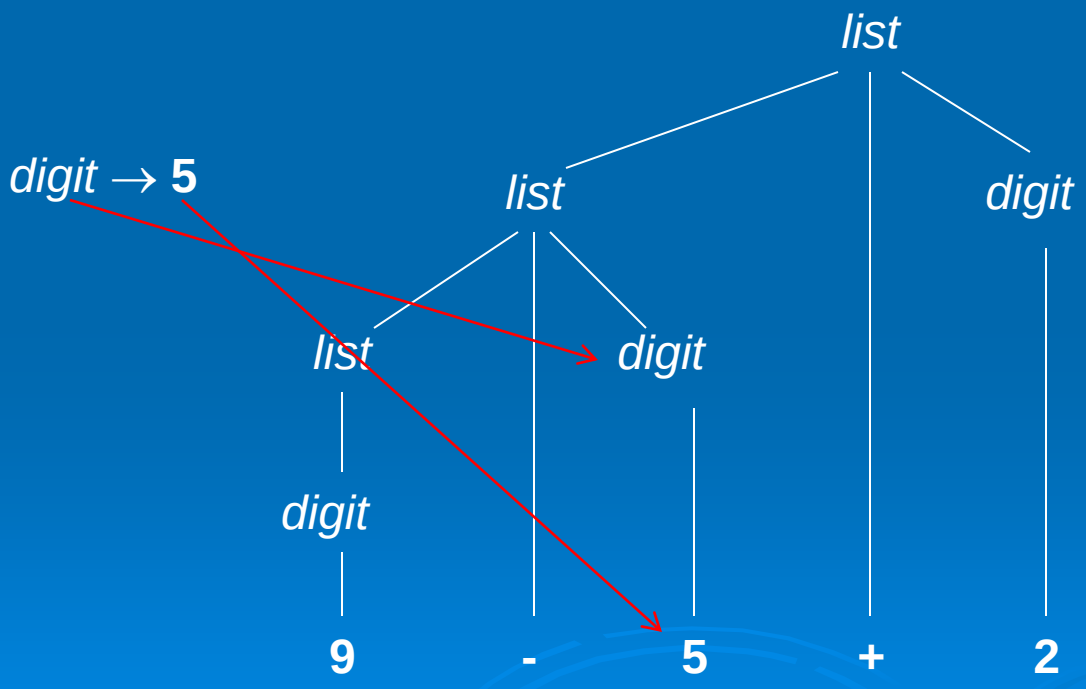
Drzewo parsowania

Dla napisu 9-5+2 i gramatyki *G* drzewo parsowania ma postać: krok 4



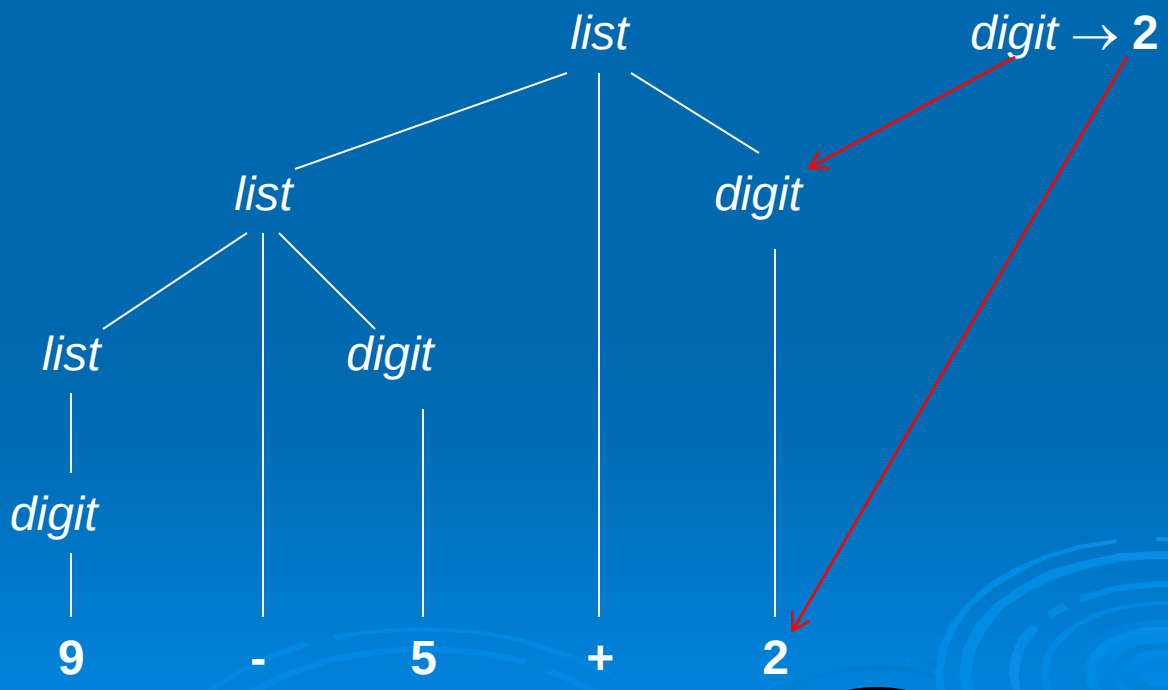
Drzewo parsowania

Dla napisu 9-5+2 i gramatyki *G* drzewo parsowania ma postać: krok 5



Drzewo parsowania

Dla napisu 9-5+2 i gramatyki *G* drzewo parsowania ma postać: krok 6



Drzewo parsowania

Formalnie, dla gramatyki bezkontekstowej, drzewo parsowania posiada następujące właściwości:

- 1. Korzeń jest oznaczony symbolem startowym.
- 2. Każdy liść jest oznaczony przez terminal lub ε.
- 3. Każdy węzeł wewnętrzny jest oznaczony przez nieterminal.

Drzewo parsowania

- 4. Jeśli *A* jest to nieterminal, który oznacza pewien węzeł wewnętrzny, oraz *X1*, *X2*, . . . , *Xn* są to etykiety dzieci tego węzła od lewej do prawej strony, to istnieje produkcja

  A -> X1X2 . . Xn,

  gdzie każde *X1*, *X2*, . . . , *Xn* oznacza symbol terminalny lub nieterminalny.
- W szczególnym przypadku, jeśli *A* ->ε jest produkcją, to węzeł oznaczony przez *A* ma jedno dziecko oznaczone przez ε.

Drzewo parsowania

- **Terminologia związana z drzewem**

  Drzewo składa się z jednego lub większej liczby węzłów.

  Węzły mogą mieć etykiety, które zazwyczaj są symbolami gramatycznymi.

  Kiedy rysujemy drzewo, często reprezentujemy węzły tylko przez te etykiety.

Drzewo parsowania

- **Terminologia związana z drzewem**
- Dokładnie jeden węzeł jest korzeniem.
- Wszystkie węzły oprócz korzenia mają unikalnego rodzica; korzeń nie ma rodzica.
- Gdy rysujemy drzewo, to umiejscawiamy rodzica zawsze powyżej dziecka, rodzic i dziecko są połączone krawędzią.

Drzewo parsowania

- **Terminologia związana z drzewem**

- Jeśli węzeł *N* jest rodzicem węzła *M*, to *M* jest dzieckiem *N*.
- Dzieci jednego węzła nazywane są rodzeństwem.

Drzewo parsowania

- **Terminologia związana z drzewem**

Węzeł bez dzieci jest to liść.

Inne węzły - te z jednym lub większą liczbą dzieci - są to węzły wewnętrzne.

Drzewo parsowania

Terminologia związana z drzewem

- Potomkiem węzła *N* jest albo sam węzeł *N*, lub dziecko *N*, lub dziecko dziecka i t.d.
- Mówimy, że węzeł *N* jest przodkiem węzła *M* jeśli *M* jest potomkiem *N*.

Gramatyka bezkontekstowa

Niejednoznaczność

- Dla danej gramatyki, dla ciągu terminali może istnieć więcej niż jedno drzewo parsowania.

- Taka gramatyka jest nazywana niejednoznaczną.

Gramatyka bezkontekstowa

Niejednoznaczność

Ponieważ napis, dla którego istnieje więcej niż jedno drzewo parsowania zwykle ma więcej niż jedno znaczenie, musimy zaprojektować gramatykę jednoznaczną, lub używać gramatyk niejednoznacznych z dodatkowymi zasadami rozwiązywania niejednoznaczności.

Gramatyka bezkontekstowa

Niejednoznaczność

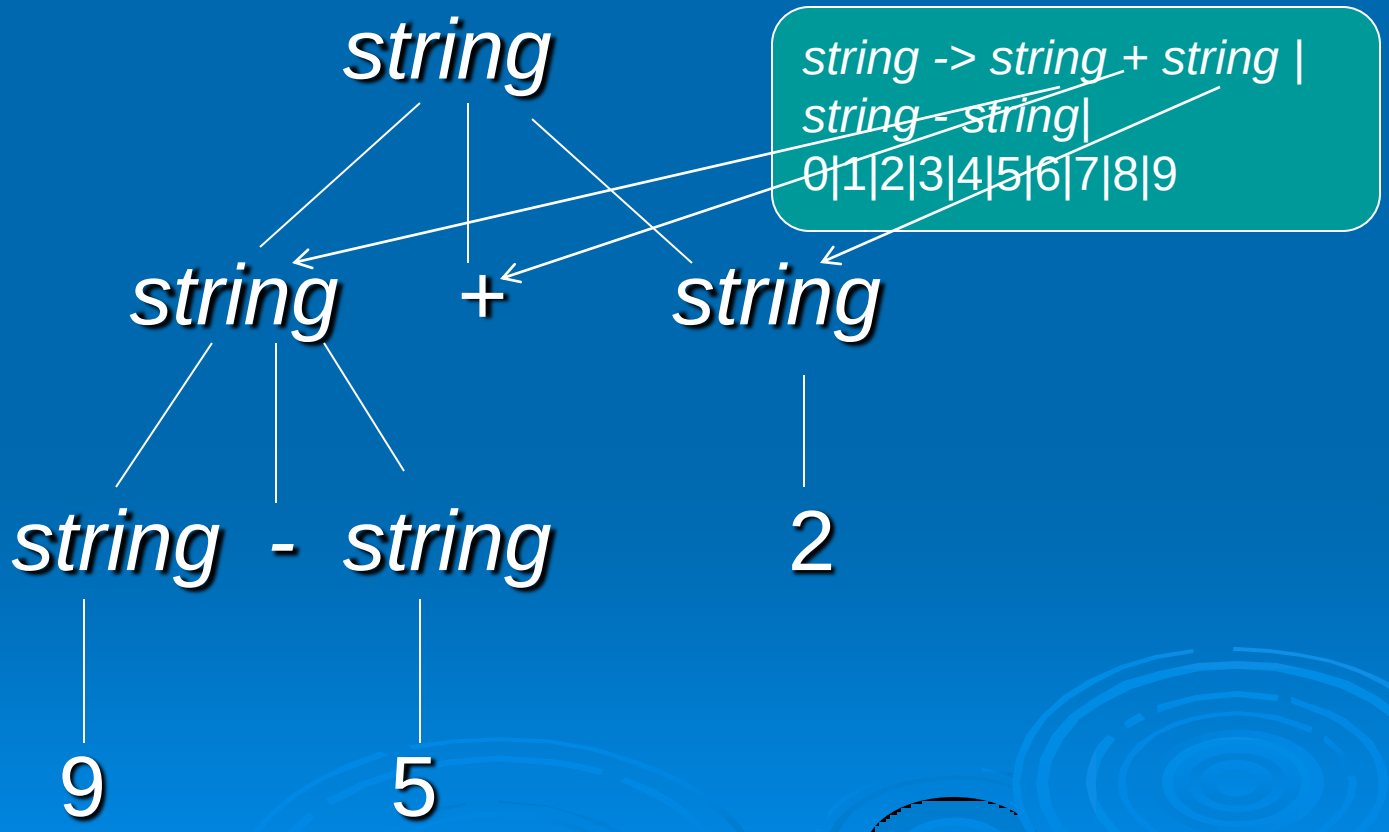
Korzystając z gramatyki:

- *string -> string + string | string - string*| 0|1|2|3|4|5|6|7|8|9,

dla napisu 9-5+2 można utworzyć dwa drzewa parsowania:

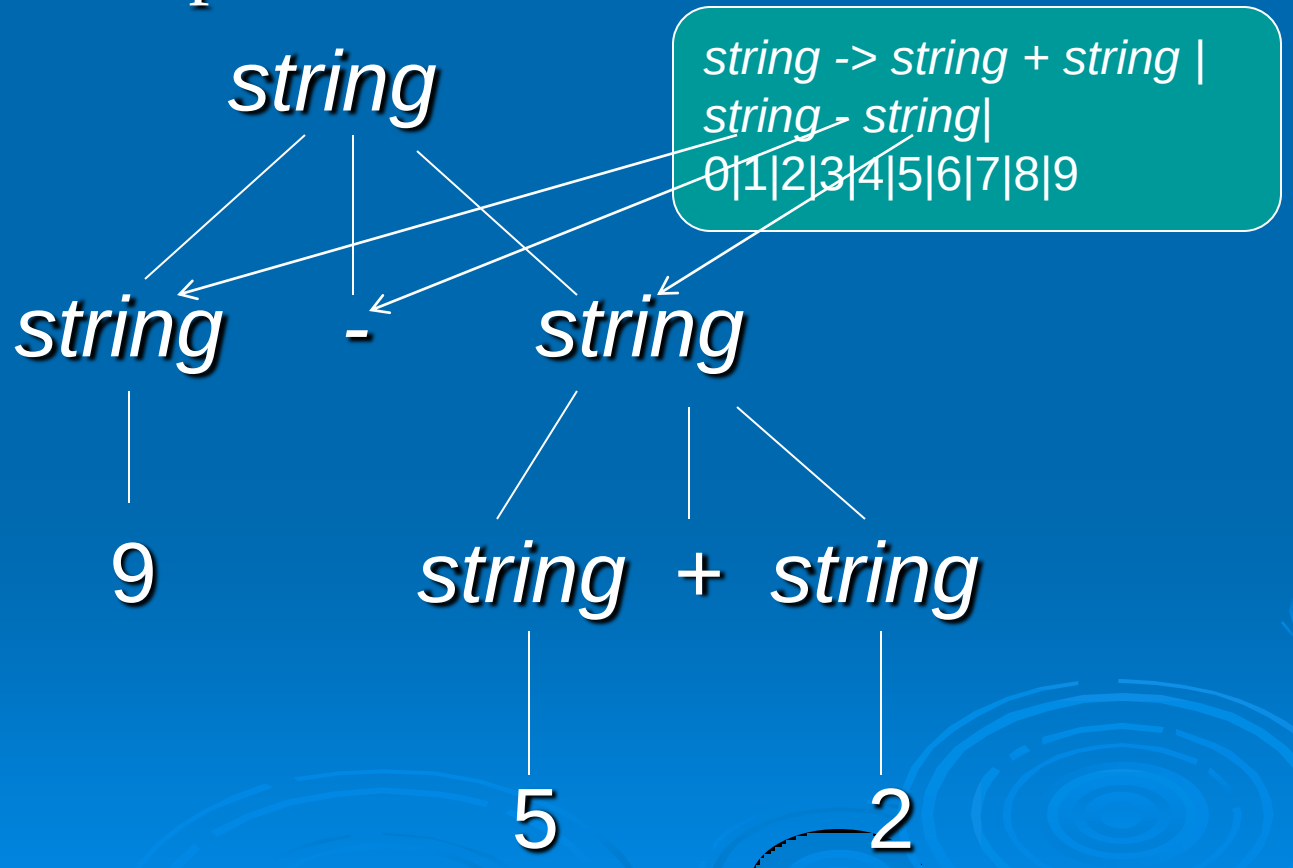
Gramatyka bezkontekstowa

Pierwsze drzewo parsowania:



Gramatyka bezkontekstowa

Drugie drzewo parsowania:



Łączność operatorów

Zgodnie z konwencją, **9+5+2** jest równoważne z **(9+5)+2** i **9-5-2** jest równoważne z **(9-5)-2.**

Gdy argument **5** ma operatory po jego lewej i prawej stronie, reguły są potrzebne do podjęcia decyzji, który z operatorów odnosi się do tego argumentu.

Łączność operatorów

- Mówimy, że operator + jest łączny lewostronnie, ponieważ argument, który ma znak plus po obu jego stronach należy do operatora po jego lewej stronie.
- W większości języków programowania cztery operatory arytmetyczne: dodawanie, odejmowanie, mnożenie i dzielenie są łączne lewostronnie.

Łączność operatorów

- Niektóre operatory, takie jak potęgowanie, są łączne prawostronnie.
- Operator przypisania = też jest łączny prawostronnie, to jest wyrażenie **a=b=c** traktuje się w taki sam sposób, jak wyrażenie **a= (b=c) .**

Łączność operatorów

- Ciągi takie jak **a=b=c** są generowane przez następującą gramatykę:

right -> letter = right | letter

letter -> a | b | …| z

Łączność operatorów

- Kontrast między drzewem parsowania dla operatora - łącznego lewostronnie i drzewem parsowania dla operatora = łącznego prawostronnie jest pokazany na następnym slajdzie.

Łączność operatorów

$list \rightarrow list + digit \mid$

$\quad \mid list - digit \mid digit$

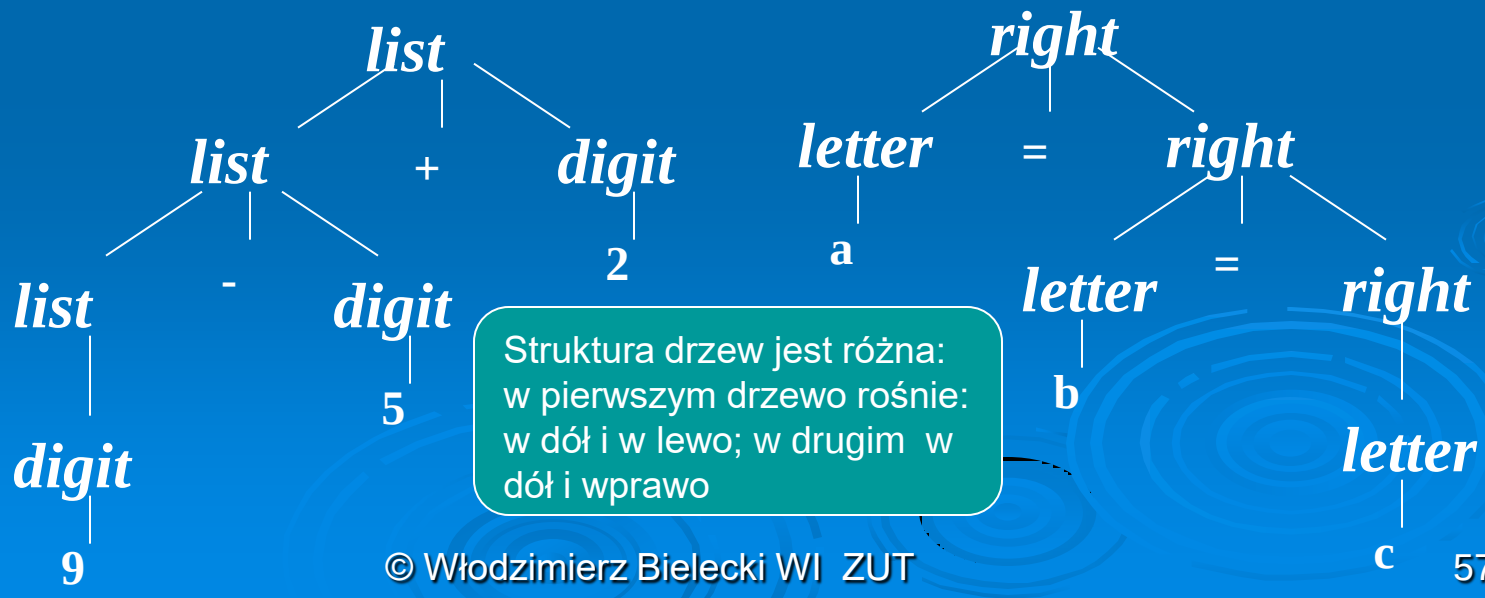
$digit \rightarrow 0 \mid 1 \mid 2 \mid \ldots \mid 9$

9-5+2 jest równoważne z (9-5)+2
Łączność lewostronna

$right \rightarrow letter = right \mid letter$

$letter \rightarrow a \mid b \mid c \mid \ldots \mid z$

a=b=c jest równoważne z a=(b=c)
Łączność prawostronna



Pierwszeństwa operatorów

Rozważmy wyrażenie **9+5*2.**

Istnieją dwie możliwe interpretacje tego wyrażenia : **(9+5) *2** lub **9+ (5*2).**

Zasadę łączności stosuje się do wystąpień tego samego operatora, więc nie rozwiązuje ona dwuznaczności.

Pierwszeństwa operatorów

- Gramatykę dla wyrażeń arytmetycznych można skonstruować w oparciu o tabelę reprezentującą pierwszeństwa operatorów.
- Zaczynamy od czterech operatorów arytmetycznych i tabeli pierwszeństwa, pokazującej operatory w kolejności rosnącego priorytetu.

Pierwszeństwa operatorów

Operatory na tej samej linii mają taką samą łączność i pierwszeństwo:

łączne lewostronnie: + -

łączne lewostronnie : * /

Gramatyka jednoznaczna

Zasada tworzenia gramatyki jednoznacznej: należy dodatkowo wprowadzić N+1 nieterminali, gdzie N jest to liczba poziomów pierwszeństwa.

Dla naszego przykładu N=2.

Gramatyka jednoznaczna

Pierwszeństwa operatorów

- Tworzymy dwa nieterminale *expr* i *term* dla dwóch poziomów pierwszeństwa oraz dodatkowy nieterminal *factor* do generowania podstawowych jednostek w wyrażeniach.
- Podstawowe jednostki w wyrażeniach są to cyfry i wyrażenia w nawiasach:

  factor -> **digit** | (*expr*)

Gramatyka jednoznaczna

Pierwszeństwa operatorów

Rozważmy teraz operatory binarne * i /, które mają najwyższy priorytet.

Odpowiednie produkcje mają postać:

*term -> term * factor*

| *term / factor*

| *factor*

Gramatyka jednoznacznaowa

- Produkcje, które odpowiadają za wyrażenia z operatorami + i – mają postać:

expr -> expr + term

| *expr - term*

| *term*

Gramatyka jednoznaczna

Łącząc powyższe gramatyki, uzyskujemy następującą gramatykę jednoznaczną:

expr -> expr + term | expr - term | term

*term -> term * factor | term / factor |factor*

*factor -> **digit** | (expr)*

Gramatyka jednoznaczna

Podsumowanie:

Krok 1:

$factor \rightarrow \mathbf{digit} \mid (\ expr\ )$

Krok 2:

$term \rightarrow term * factor$
$\quad \mid term\ /\ factor$
$\quad \mid factor$

Krok 3:

$expr \rightarrow expr\ +\ term$
$\quad \mid expr - term$
$\quad \mid term$

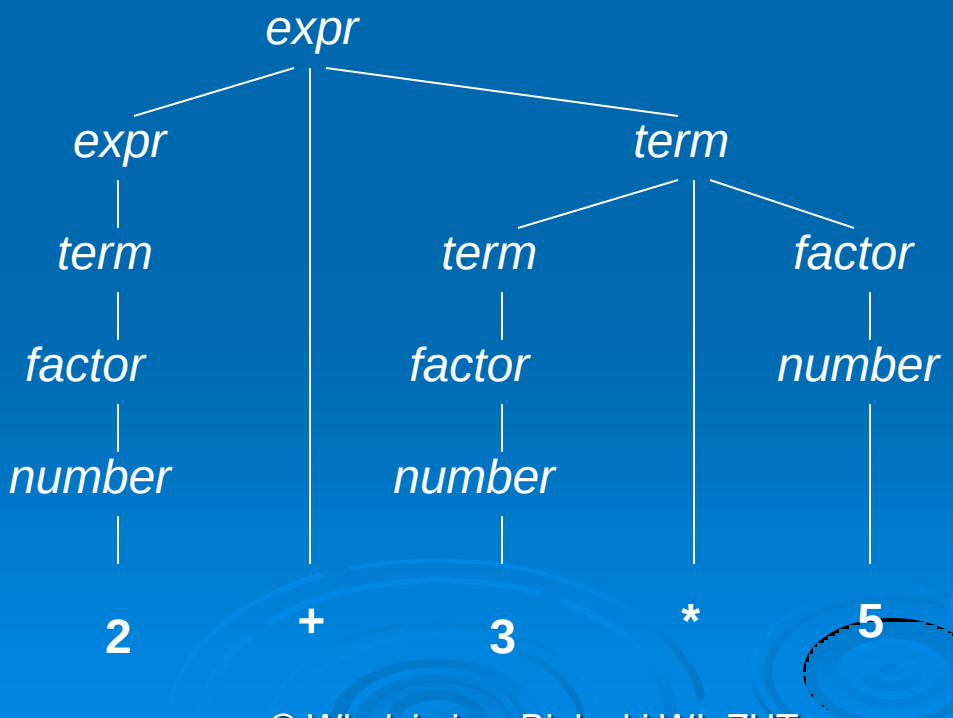
Krok 4:

$expr \rightarrow expr + term \mid expr - term \mid term$
$term \rightarrow term * factor \mid term\ /\ factor \mid factor$
$factor \rightarrow \mathbf{digit} \mid (\ expr\ )$

Gramatyka jednoznaczna

expr → *expr* **+** *term* | *term*
term → *term* ***** *factor* | *factor*
factor → *number* | **(** *expr* **)**

Zdanie **2+3*5** jest parsowane w sposób jednoznaczny:



Dziękuję za uwagę