Gramatyki to instrukcje umożliwiające zbudowanie zdań z symboli terminalnych ułożonych według ścisłych reguł w nich zawartych. Każdy język programowania wymaga pewnej składni formalnie zapisywanej w postaci gramatyki. Napisanie programu w nieprawidłowej składni uniemożliwia jego kompilację. Błędy składniowe mogą mieć dwa główne źródła:

- zastosowanie niezdefiniowanego w gramatyce terminala
- użycie terminali w kolejności niezgodnej z regułami zawartymi w gramatyce

Kolejno przykładami powyższych błędów w języku C są:

- ```
  foreach (a in tab)
  {
       c = a * 12;
  }
  ```
- ```
  if (c == true) )
       c = a * 12;
  else
       c = 0;
  ```

Analizę programu pod względem składni realizuje analizator składniowy, który w momencie napotkania na błąd generuje stosowny komunikat.

Zadaniem laboratorium będzie napisanie analizatora leksykalnego do następującej gramatyki:

```
S ::= W ; Z
Z ::= W ; Z | ε
W ::= P | P O W
P ::= L | ( W )
L ::= C | C L
C ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
O ::= * | : | + | - | ^
```

Gramatyka ta umożliwia budowanie ciągów liczb całkowitych i operacji arytmetycznych (zdań arytmetycznych), np.:

```
(12*3)+5-(234+3)^3; 8:13;
```

Powyższa gramatyka nie spełnia założeń gramatyk klasy LL(1). Dlatego konieczne jest poprawienie produkcji W oraz L tak aby można było zbudować

prosty analizator dla tej gramatyki. Po poprawieniu produkcje te będą wyglądały następująco:

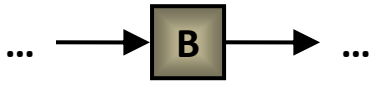
```
W  ::= P W'
W' ::= O W | ε
L  ::= C L'
L' ::= L | ε
```

Aby ułatwić zaimplementowanie analizatora składniowego należy przedstawić podaną gramatykę w postaci diagramów składniowych (*diagramów kolejowych*), które stanowią zapis schematu blokowego programu analizatora.

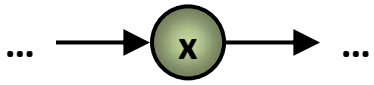
Diagramami składni rządzą następujące reguły:

- każdą produkcję zastępujemy diagramem:

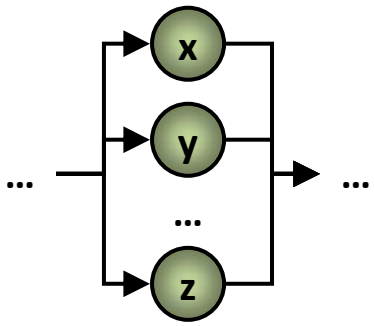

- symbole pomocnicze przedstawiamy w kwadratowych blokach




- symbole końcowe przedstawiamy w okrągłych blokach

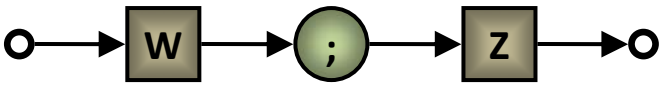



- alternatywy reprezentowane są przez rozgałęzienia

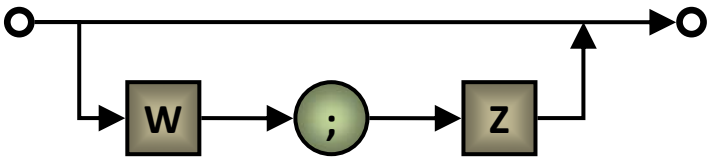



Zgodnie z tymi regułami naszą gramatykę można przedstawić w postaci następujących diagramów składni:

S:

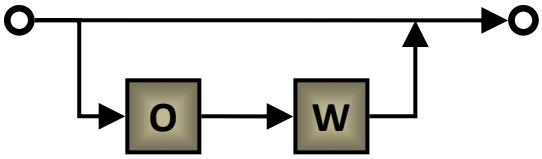



Z:

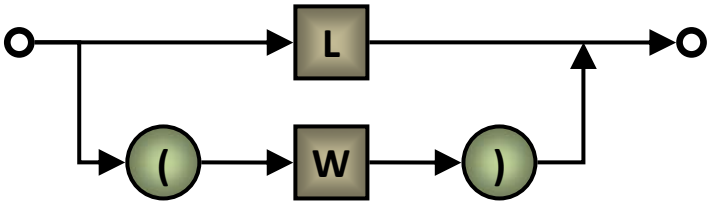



W:

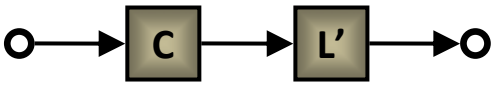

W':



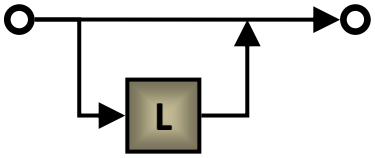

P:



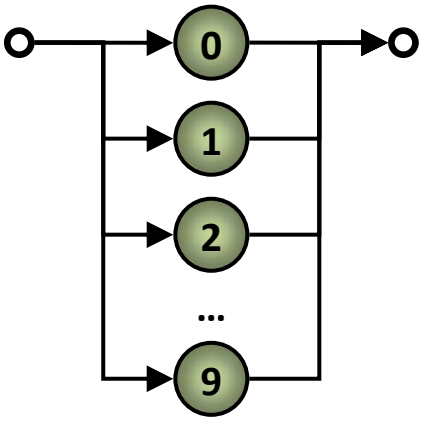

L:



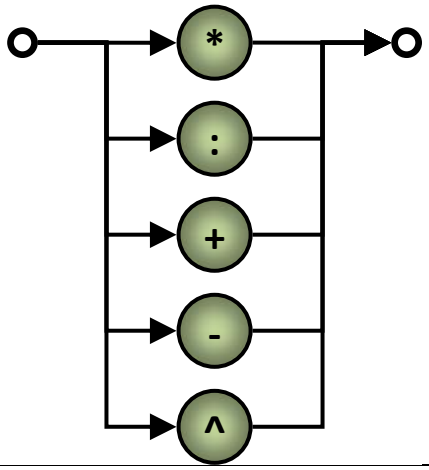

L':



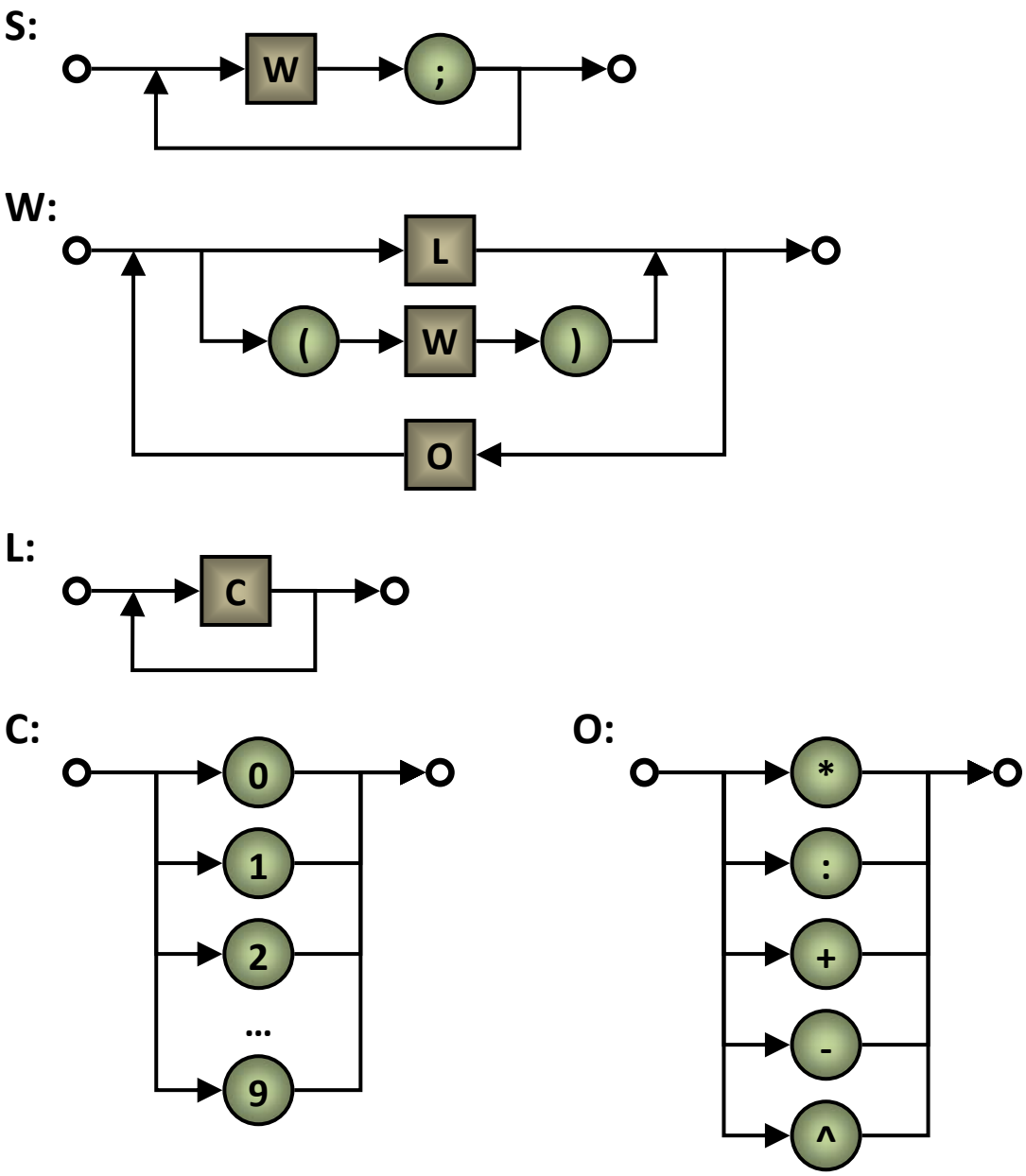

C:




O:



Kolejno podstawiając diagramy do siebie można je uprościć oraz zredukować ich ilość.




Następnie na podstawie takiego diagramu składni można już napisać program analizatora.

# Zadanie

Aby umożliwić pełne wykorzystanie nieprzeciętnych umiejętności każdego z Państwa zadanie tego laboratorium zostało podzielone na różne stopnie trudności. Stopnie te należy przechodzić kolejno od najprostszego do najtrudniejszego.

**Stopień 1.** Napisać w jednym z wymienionych języków: Java, C/C++, C#, analizator składniowy przyjmujący jako parametr ciąg znaków będących sprawdzanym zdaniem, jako wynik ma się pojawić komunikat o prawidłowości lub nieprawidłowości badanego zdania. Program ten może być w formie konsolowej.

**Stopień 2.** Jak wyżej z tą różnicą, że program działa w środowisku graficznym, czyli musi być miejsce do wpisania badanego zdania oraz klawisz lub inny wyzwalacz do rozpoczęcia analizy, wymagane jest również okienko lub wyróżniony fragment interfejsu reprezentujący komunikat o poprawności badanego zdania.

**Stopień 3.** Przedstawiona gramatyka działa jedynie na liczbach całkowitych. Należy zmodyfikować ją w taki sposób, aby mogła pracować na liczbach zmiennoprzecinkowych. Po dokonaniu zmian w gramatyce należy wykreślić dla niej diagram składni oraz zaimplementować.

Do zaliczenia tego ćwiczenia wymagany jest jedynie pierwszy stopień, pozostałe są dla osób, które nie chcą równać w dół i z spokojnym sumieniem nosić dumne miano inżyniera i mistrza nauki (*magistra*).