

Stefan Sokołowski

JĘZYKI FORMALNE I METODY KOMPILACJI

wykład 5

Inst. Informatyki Stosowanej, ANS

Elbląg, 2025/2026

Notacja beznawiasowa Łukasiewicza

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):



Jan Łukasiewicz
1878–1956

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):

Wyrażenia (arytmetyczne, logiczne, itp.) można zapisywać **bez nawiasów** w postaci

działanie $\arg_1 \arg_2 \dots \arg_n$



Jan Łukasiewicz
1878–1956

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):

Wyrażenia (arytmetyczne, logiczne, itp.) można zapisywać **bez nawiasów** w postaci

działanie $\arg_1 \arg_2 \dots \arg_n$

Np.:

$= * 4 + 2 3 20$



Jan Łukasiewicz
1878–1956

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):

Wyrażenia (arytmetyczne, logiczne, itp.) można zapisywać **bez nawiasów** w postaci

działanie $\arg_1 \arg_2 \dots \arg_n$

Np.:

$$= * 4 \boxed{+ \begin{matrix} 2 & 3 \\ 2+3 \end{matrix}} 20$$



Jan Łukasiewicz
1878–1956

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):

Wyrażenia (arytmetyczne, logiczne, itp.) można zapisywać **bez nawiasów** w postaci

działanie $\arg_1 \arg_2 \dots \arg_n$

Np.:

$$= \boxed{* 4 \boxed{+ 2 3} 20}$$

$4 * (2 + 3)$



Jan Łukasiewicz
1878–1956

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):

Wyrażenia (arytmetyczne, logiczne, itp.) można zapisywać **bez nawiasów** w postaci

działanie $\arg_1 \arg_2 \dots \arg_n$

Np.:

$$= * 4 \boxed{+ 2 3} 20$$

$$\quad \quad \quad \underbrace{\quad}_{2+3}$$

$$\quad \quad \quad \underbrace{\quad}_{4*(2+3)}$$

$$\underbrace{\quad}_{4*(2+3)=20}$$



Jan Łukasiewicz
1878–1956

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):

Wyrażenia (arytmetyczne, logiczne, itp.) można zapisywać **bez nawiasów** w postaci

działanie $\arg_1 \arg_2 \dots \arg_n$

Np.:

$$= * 4 \boxed{+ 2 3} 20$$

$2+3$

$4 * (2+3)$

$4 * (2 + 3) = 20$

Odwrotna notacja Łukasiewicza

(*Reverse Polish*) stosowana w informatyce



Jan Łukasiewicz
1878–1956

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):

Wyrażenia (arytmetyczne, logiczne, itp.) można zapisywać **bez nawiasów** w postaci

działanie $\arg_1 \arg_2 \dots \arg_n$

Np.:

$$= * 4 \boxed{+ 2 3} 20$$

$2+3$

$4 * (2+3)$

$4 * (2 + 3) = 20$

Odwrotna notacja Łukasiewicza

(*Reverse Polish*) stosowana w informatyce:

$\arg_1 \arg_2 \dots \arg_n$ działanie



Jan Łukasiewicz
1878–1956

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):

Wyrażenia (arytmetyczne, logiczne, itp.) można zapisywać **bez nawiasów** w postaci

działanie $\arg_1 \arg_2 \dots \arg_n$

Np.:

$$= * 4 \boxed{+ 2 3} 20$$

$$\quad \quad \quad \underbrace{\quad}_{2+3}$$

$$\quad \quad \quad \underbrace{\quad}_{4*(2+3)}$$

$$\underbrace{\quad}_{4*(2+3)=20}$$



Jan Łukasiewicz
1878–1956

Odwrotna notacja Łukasiewicza

(*Reverse Polish*) stosowana w informatyce:

$\arg_1 \arg_2 \dots \arg_n$ działanie

Np.:

$$4 \ 2 \ 3 \ + \ * \ 20 \ =$$

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):

Wyrażenia (arytmetyczne, logiczne, itp.) można zapisywać **bez nawiasów** w postaci

działanie $\arg_1 \arg_2 \dots \arg_n$

Np.:

$$= * 4 \boxed{+ 2 3} 20$$

$$\boxed{4 * (2 + 3) = 20}$$

Odwrotna notacja Łukasiewicza

(*Reverse Polish*) stosowana w informatyce:

$\arg_1 \arg_2 \dots \arg_n$ działanie

Np.:

$$4 \boxed{2 3 +} * 20 =$$



Jan Łukasiewicz
1878–1956

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):

Wyrażenia (arytmetyczne, logiczne, itp.) można zapisywać **bez nawiasów** w postaci

działanie $\arg_1 \arg_2 \dots \arg_n$

Np.:

$$= * 4 \boxed{+ 2 3} 20$$

$2+3$

$4 * (2+3)$

$4 * (2 + 3) = 20$

Odwrotna notacja Łukasiewicza

(*Reverse Polish*) stosowana w informatyce:

$\arg_1 \arg_2 \dots \arg_n$ działanie

Np.:

$$\boxed{4 \ 2 \ 3 \ +} * 20 =$$



Jan Łukasiewicz
1878–1956

Notacja beznawiasowa Łukasiewicza

Notacja Łukasiewicza

(w świecie nazywana *Polish*):

Wyrażenia (arytmetyczne, logiczne, itp.) można zapisywać **bez nawiasów** w postaci

działanie $\arg_1 \arg_2 \dots \arg_n$

Np.:

$$= * 4 \boxed{+ 2 3} 20$$

$$\quad \quad \quad \underbrace{\quad}_{2+3}$$

$$\quad \quad \quad \underbrace{\quad}_{4*(2+3)}$$

$$\underbrace{\quad}_{4*(2+3)=20}$$

Odwrotna notacja Łukasiewicza

(*Reverse Polish*) stosowana w informatyce:

$\arg_1 \arg_2 \dots \arg_n$ działanie

Np.:

$$\boxed{4 \ 2 \ 3 \ +} \ * \ 20 \ =$$



Jan Łukasiewicz
1878–1956

Jak to czytać?

Jak to czytać?

(na **cz**erwono pod operatorem — liczba argumentów czyli „**arność**”)

Łukasiewicza	odwrotna	czyli	klasycznie
$+_2 2\ 5$			

Jak to czytać?

(na **cz**erwono pod operatorem — liczba argumentów czyli „**arność**”) . . .

unary — jednoargumentowy
binary — dwuargumentowy
ternary — trójąrgumentowy

Łukasiewicza	odwrotna	czyli	klasycznie
$+_2 2 5$			

Jak to czytać?

(na **cz**erwono pod operatorem — liczba argumentów czyli „**arność**”) . . .

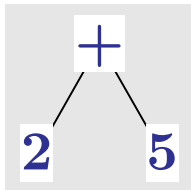
unary — jednoargumentowy
binary — dwuargumentowy
ternary — trójąrgumentowy

Łukasiewicza	odwrotna	czyli	klasycznie
$+_2 2 5$	$2 5 +_2$		

Jak to czytać?

(na **cz**erwono pod operatorem — liczba argumentów czyli „**arność**”) . . .

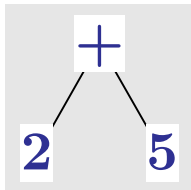
unary — jednoargumentowy
binary — dwuargumentowy
ternary — trójąrgumentowy

Łukasiewicza	odwrotna	czyli	klasycznie
$+_2 2 5$	$2 5 +_2$		

Jak to czytać?

(na **cz**erwono pod operatorem — liczba argumentów czyli „**arność**”) . . .

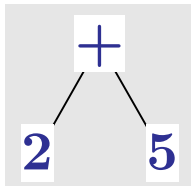
unary — jednoargumentowy
binary — dwuargumentowy
ternary — trójąrgumentowy

Łukasiewicza	odwrotna	czyli	klasycznie
$+_2 2 5$	$2 5 +_2$		$2 + 5$

Jak to czytać?

unary — jednoargumentowy
 binary — dwuargumentowy
 ternary — trójąrgumentowy
 ...

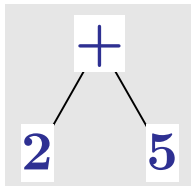
(na czerwono pod operatorem — liczba argumentów czyli „arność”)

Łukasiewicza	odwrotna	czyli	klasycznie
$+ \underset{2}{2} 5$	$2 5 \underset{2}{+}$		$2 + 5$
$* \underset{2}{+} \underset{2}{2} 5 - \underset{2}{3} 1$			

Jak to czytać?

(na **cz**erwono pod operatorem — liczba argumentów czyli „**arność**”) . . .

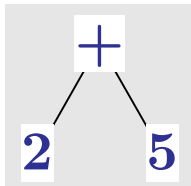
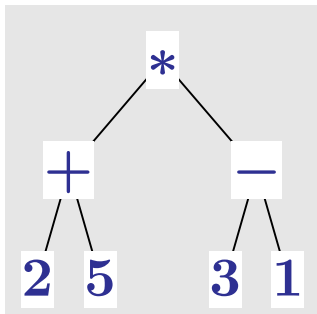
unary — jednoargumentowy
binary — dwuargumentowy
ternary — trójąrgumentowy

Łukasiewicza	odwrotna	czyli	klasycznie
$+ 2 5$ ₂	$2 5 +$ ₂		$2 + 5$
$* + 2 5 - 3 1$ _{2 2 2}	$2 5 + 3 1 - *$ _{2 2 2}		

Jak to czytać?

(na **czzerwono** pod operatorem — liczba argumentów czyli „**arność**”) . . .

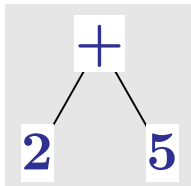
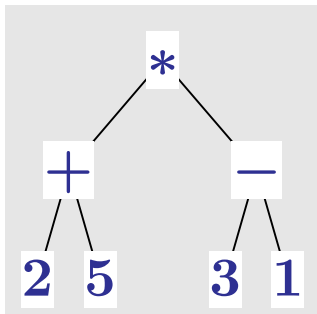
unary — jednoargumentowy
binary — dwuargumentowy
ternary — trójąrgumentowy

Łukasiewicza	odwrotna	czyli	klasycznie
$+ \underset{2}{2} 5$	$2 5 \underset{2}{+}$		$2 + 5$
$* \underset{2}{+} \underset{2}{2} 5 \underset{2}{-} 3 1$	$2 5 \underset{2}{+} 3 1 \underset{2}{-} \underset{2}{*}$		

Jak to czytać?

unary — jednoargumentowy
 binary — dwuargumentowy
 ternary — trójąrgumentowy
 ...

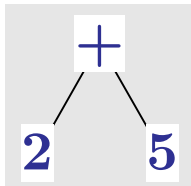
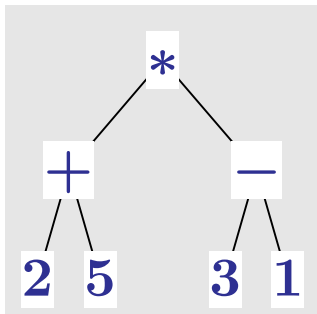
(na czerwono pod operatorem — liczba argumentów czyli „arność”)

Łukasiewicza	odwrotna	czyli	klasycznie
$+ \underset{2}{2} 5$	$2 5 \underset{2}{+}$		$2 + 5$
$* \underset{2}{+} \underset{2}{2} 5 \underset{2}{-} 3 1$	$2 5 \underset{2}{+} 3 1 \underset{2}{-} \underset{2}{*}$		$(2 + 5) * (3 - 1)$

Jak to czytać?

unary — jednoargumentowy
 binary — dwuargumentowy
 ternary — trójąrgumentowy
 ...

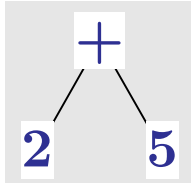
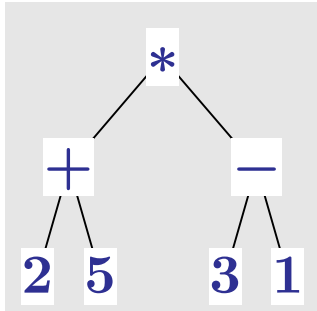
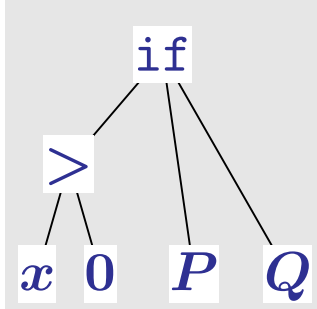
(na czerwono pod operatorem — liczba argumentów czyli „arność”)

Łukasiewicza	odwrotna	czyli	klasycznie
$+ \underset{2}{2} 5$	$2 5 \underset{2}{+}$		$2 + 5$
$* \underset{2}{+} \underset{2}{2} 5 - \underset{2}{3} 1$	$2 5 \underset{2}{+} 3 1 - \underset{2}{*} \underset{2}{2}$		$(2 + 5) * (3 - 1)$
			$\text{if}(x > 0) P \text{ else } Q$

Jak to czytać?

unary — jednoargumentowy
 binary — dwuargumentowy
 ternary — trójąrgumentowy
 ...

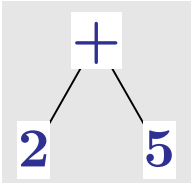
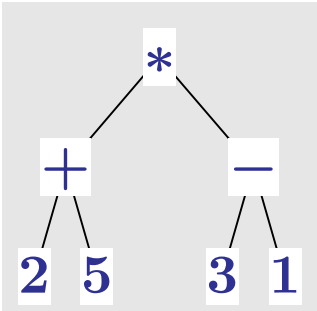
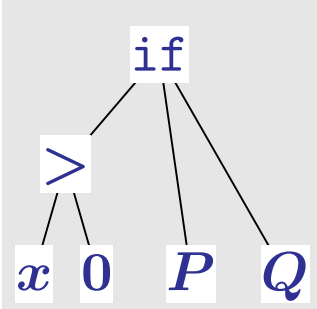
(na czerwono pod operatorem — liczba argumentów czyli „arność”)

Łukasiewicza	odwrotna	czyli	klasycznie
$+ \underset{2}{2} 5$	$2 5 + \underset{2}$		$2 + 5$
$* + \underset{2}{2} 5 - \underset{2}{3} 1$	$2 5 + \underset{2}{3} 1 - \underset{2}{*}$		$(2 + 5) * (3 - 1)$
			$\text{if}(x > 0) P \text{ else } Q$

Jak to czytać?

unary — jednoargumentowy
 binary — dwuargumentowy
 ternary — trójąrgumentowy
 ...

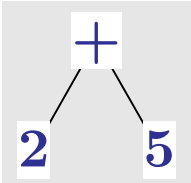
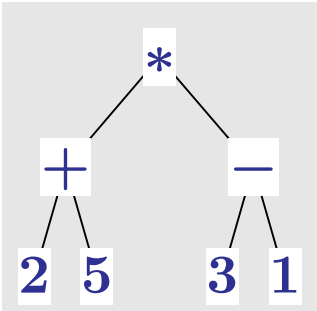
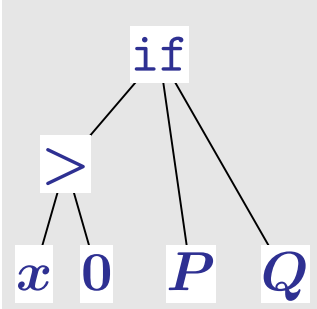
(na czerwono pod operatorem — liczba argumentów czyli „arność”)

Łukasiewicza	odwrotna	czyli	klasycznie
$+ \underset{2}{2} 5$	$2 5 \underset{2}{+}$		$2 + 5$
$* \underset{2}{+} \underset{2}{2} 5 \underset{2}{-} 3 1$	$2 5 \underset{2}{+} 3 1 \underset{2}{-} *$		$(2 + 5) * (3 - 1)$
$\text{if} \underset{3}{>} \underset{2}{x} 0 P Q$	$x 0 \underset{2}{>} P Q \underset{3}{\text{if}}$		$\text{if}(x > 0) P \text{ else } Q$

Jak to czytać?

unary — jednoargumentowy
 binary — dwuargumentowy
 ternary — trójąrgumentowy
 ...

(na czerwono pod operatorem — liczba argumentów czyli „arność”)

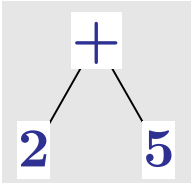
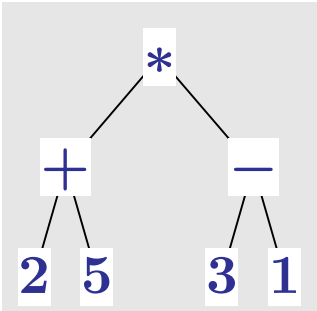
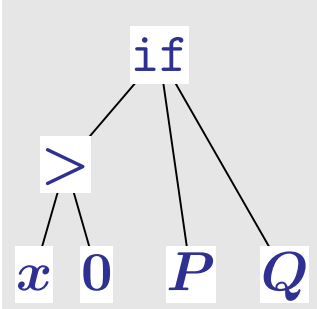
Łukasiewicza	odwrotna	czyli	klasycznie
$+ \underset{2}{2} 5$	$2 5 \underset{2}{+}$		$2 + 5$
$* \underset{2}{+} \underset{2}{2} 5 \underset{2}{-} 3 1$	$2 5 \underset{2}{+} 3 1 \underset{2}{-} *$		$(2 + 5) * (3 - 1)$
$\text{if} \underset{3}{>} \underset{2}{x} 0 P Q$	$x 0 \underset{2}{>} P Q \underset{3}{\text{if}}$		$\text{if}(x > 0) P \text{ else } Q$

Wyrażenia — drzewa (nie całkiem drz. wywodu).

Jak to czytać?

unary — jednoargumentowy
 binary — dwuargumentowy
 ternary — trójąrgumentowy
 ...

(na czerwono pod operatorem — liczba argumentów czyli „arność”)

Łukasiewicza	odwrotna	czyli	klasycznie
$+ \underset{2}{2} 5$	$2 5 \underset{2}{+}$		$2 + 5$
$* \underset{2}{+} \underset{2}{2} 5 \underset{2}{-} 3 1$	$2 5 \underset{2}{+} 3 1 \underset{2}{-} *$		$(2 + 5) * (3 - 1)$
$\text{if} \underset{3}{>} \underset{2}{x} 0 P Q$	$x 0 \underset{2}{>} P Q \underset{3}{\text{if}}$		$\text{if}(x > 0) P \text{ else } Q$

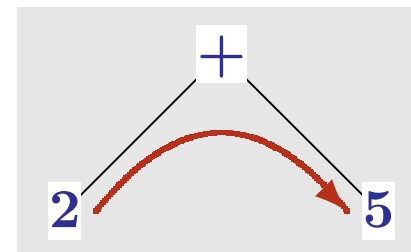
Wyrażenia — drzewa (nie całkiem drz. wywodu).

Różne **notacje** — sposoby chodzenia po drzewach.

Notacja beznawiasowa a chodzenie po drzewach



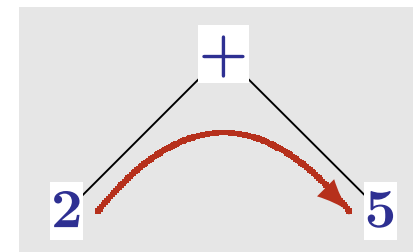
Notacja beznawiasowa a chodzenie po drzewach



zwykła notacja (*infix*):

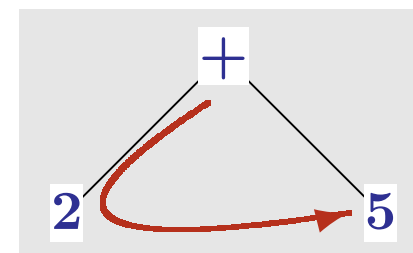
$$2 + 5$$

Notacja beznawiasowa a chodzenie po drzewach



zwykła notacja (*infix*):

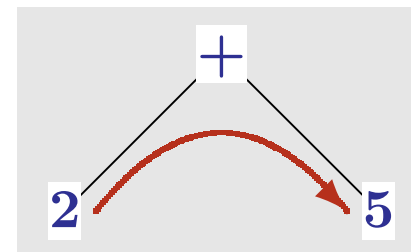
$$2 + 5$$



Notacja Łukasiewicza (*prefix*):

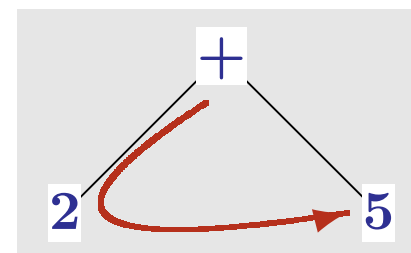
$$+ 2 5$$

Notacja beznawiasowa a chodzenie po drzewach



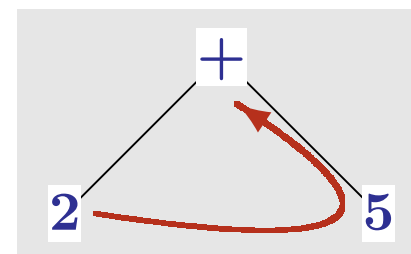
zwykła notacja (*infix*):

$$2 + 5$$



Notacja Łukasiewicza (*prefix*):

$$+ 2 5$$

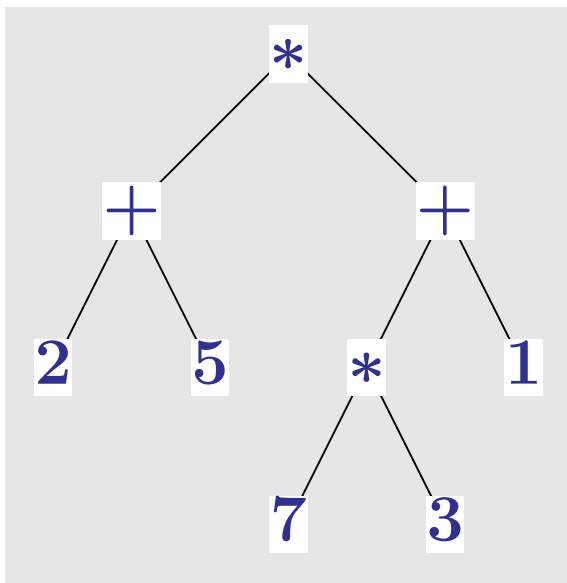


Odwrotna notacja
polska (*postfix*):

$$2 5 +$$

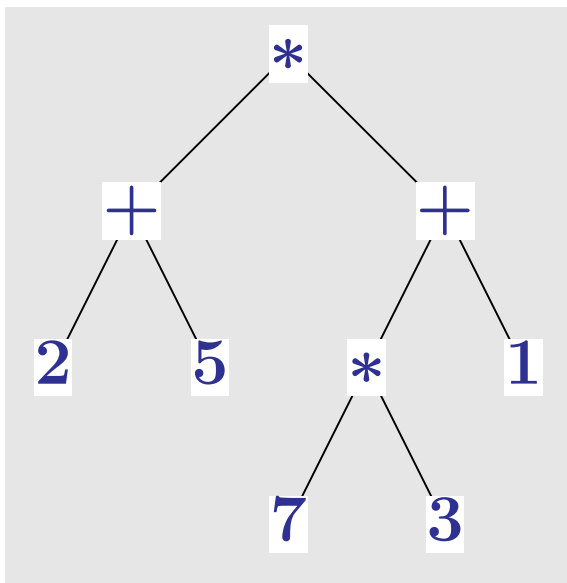
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



Notacja beznawiasowa a chodzenie po drzewach

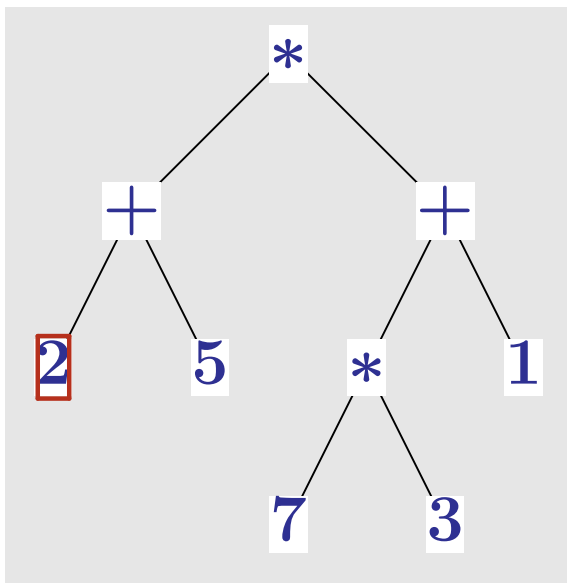
PRZYKŁAD: (bardziej złożony)



infix — notacja
zwykła, **nawiasowa**

Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)

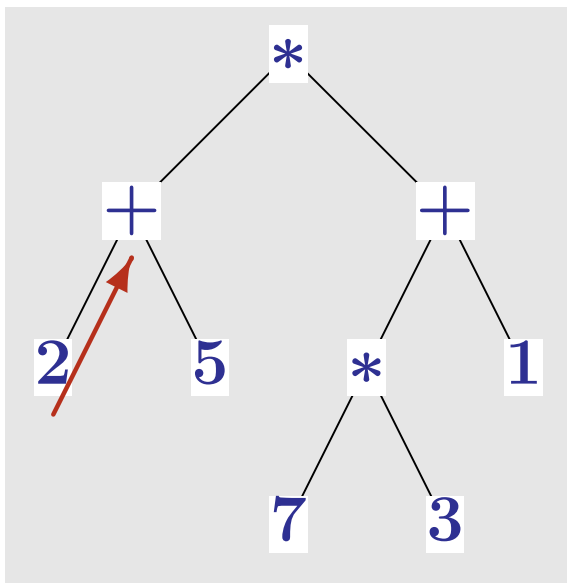


infix — notacja
zwykła, **nawiasowa**:

2

Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)

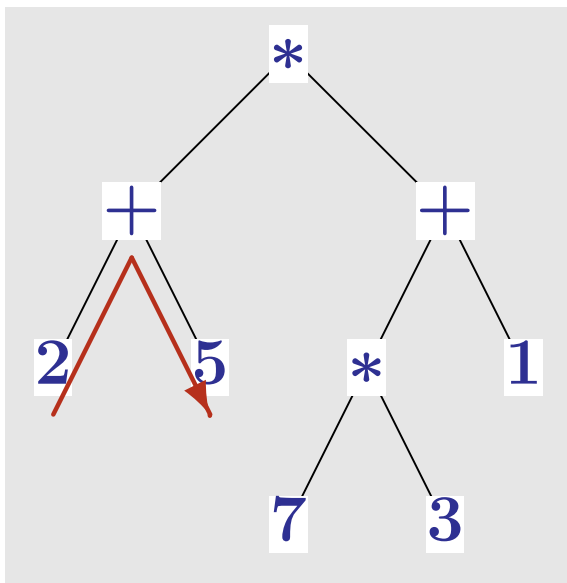


infix — notacja
zwykła, **nawiasowa**:

2 +

Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)

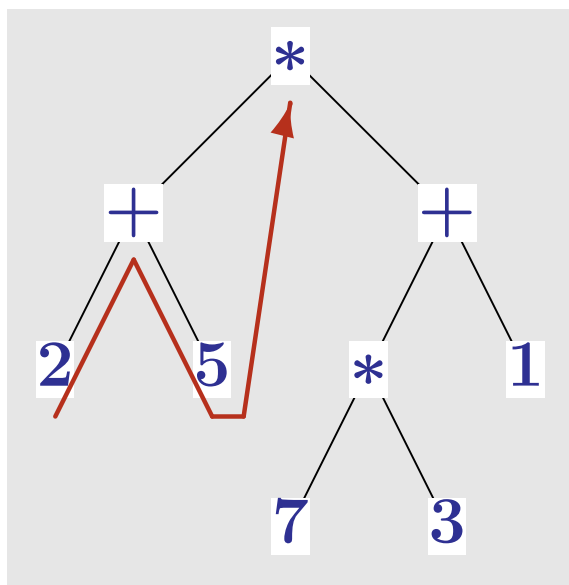


infix — notacja
zwykła, **nawiasowa**:

$$2 + 5$$

Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)

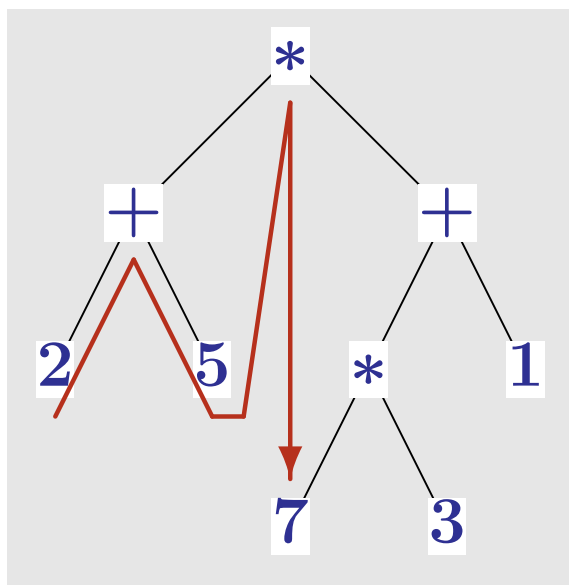


infix — notacja
zwykła, **nawiasowa**:

$(2 + 5) *$

Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)

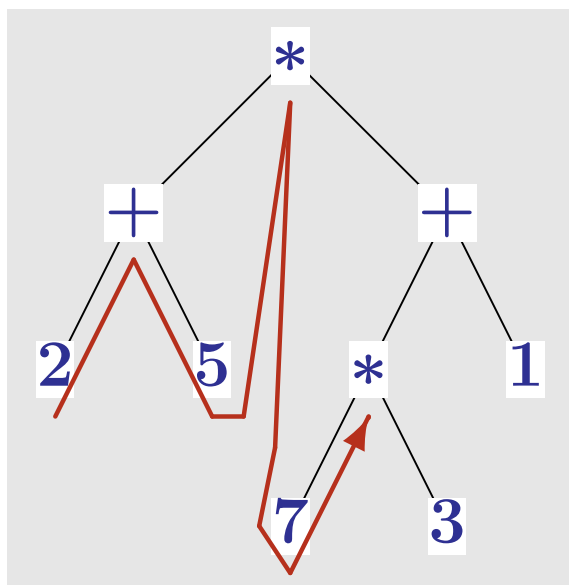


infix — notacja
zwykła, **nawiasowa**:

$(2 + 5) * (7 \quad)$

Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)

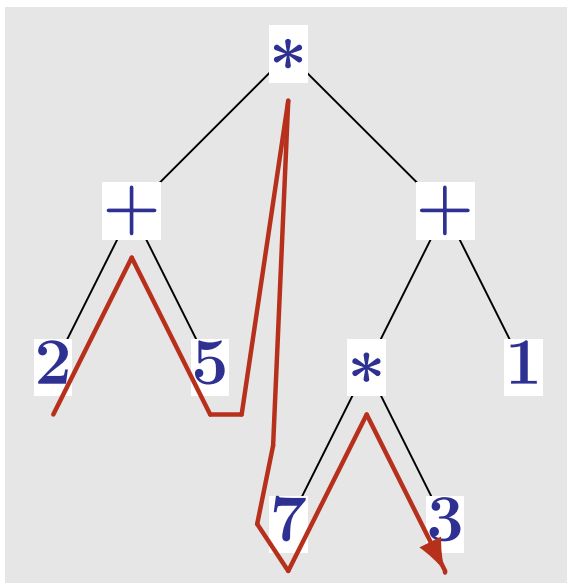


infix — notacja
zwykła, **nawiasowa**:

$(2 + 5) * (7 * 3)$

Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)

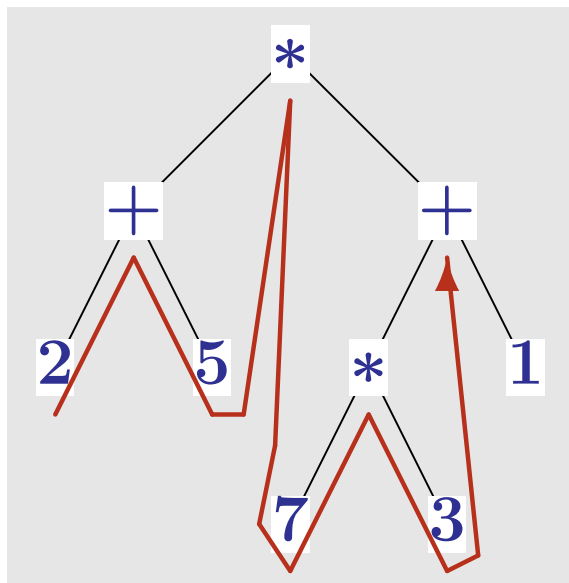


infix — notacja
zwykła, **nawiasowa**:

$(2 + 5) * (7 * 3 \quad)$

Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)

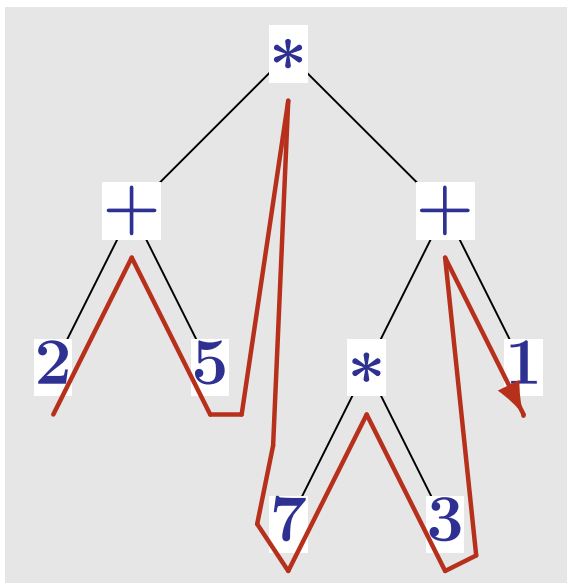


infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 +)$$

Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)

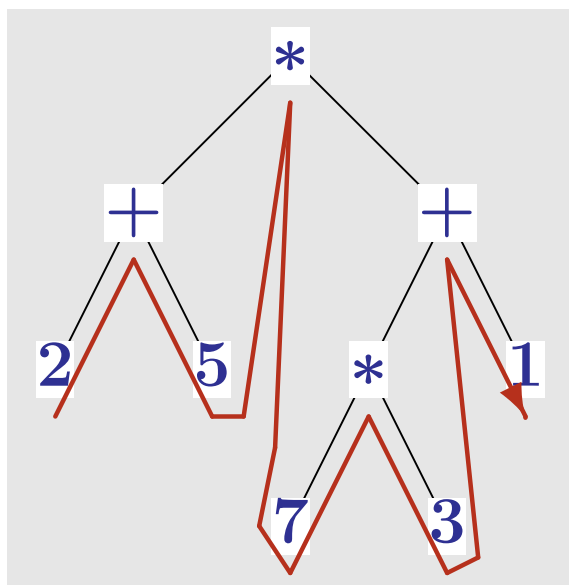


infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$

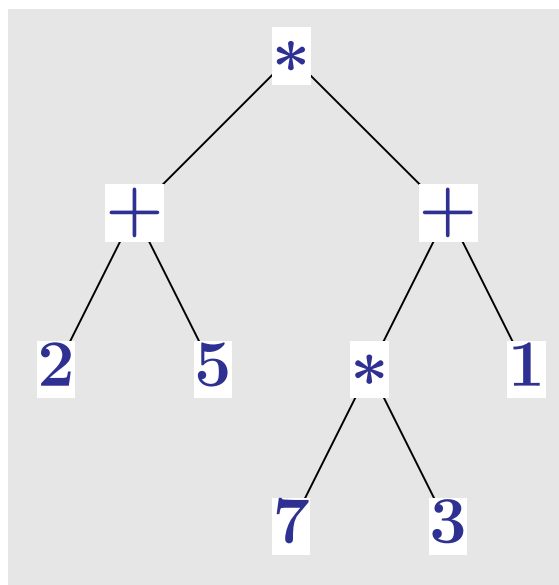
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



infix — notacja
zwykła, **nawiasowa**:

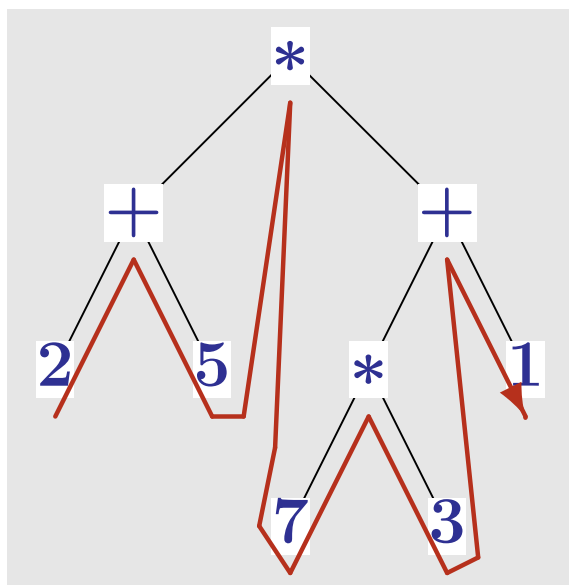
$$(2 + 5) * (7 * 3 + 1)$$



prefix — notacja
Łukasiewicza

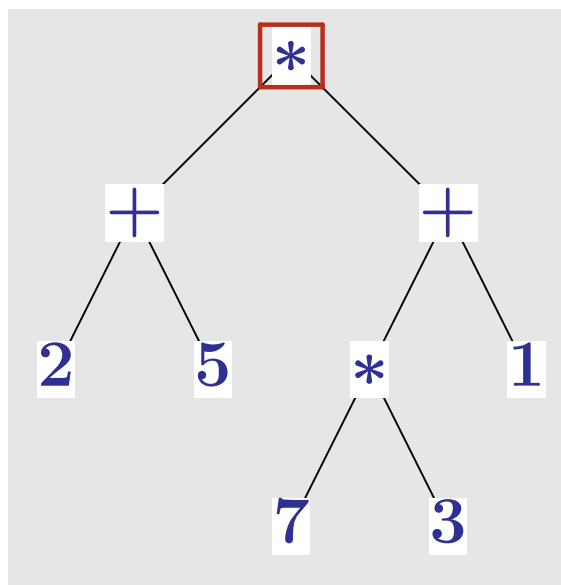
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$

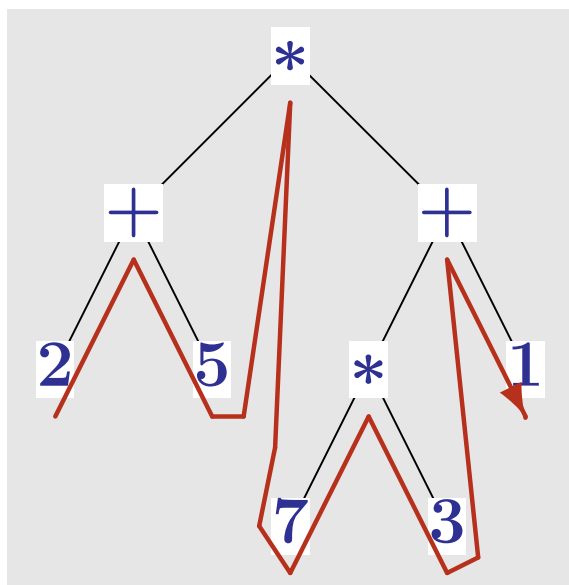


prefix — notacja
Łukasiewicza:

*

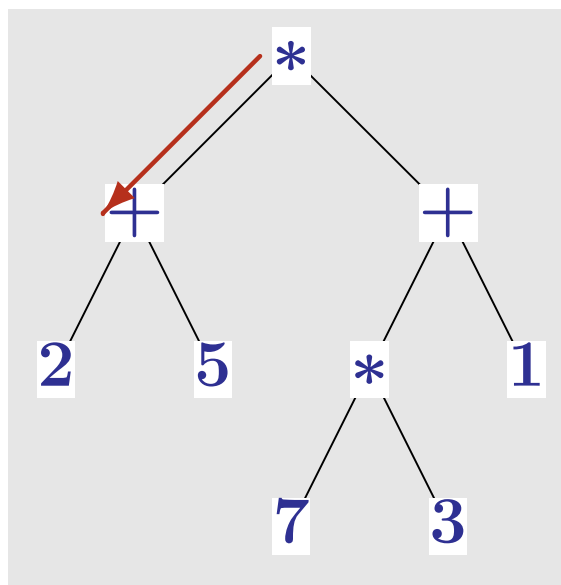
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$

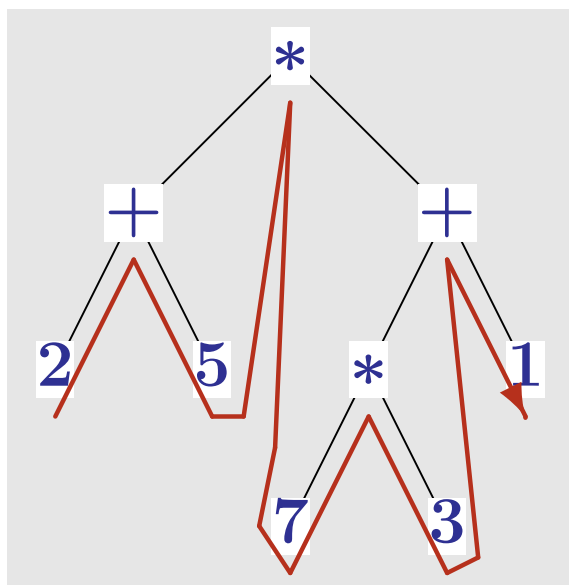


prefix — notacja
Łukasiewicza:

$$* +$$

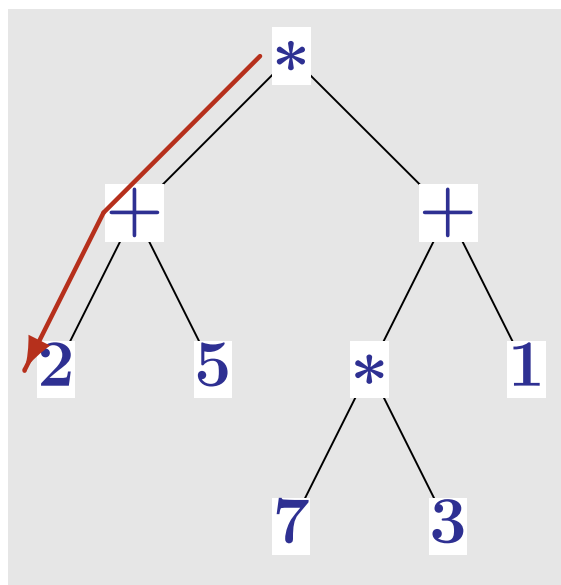
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$

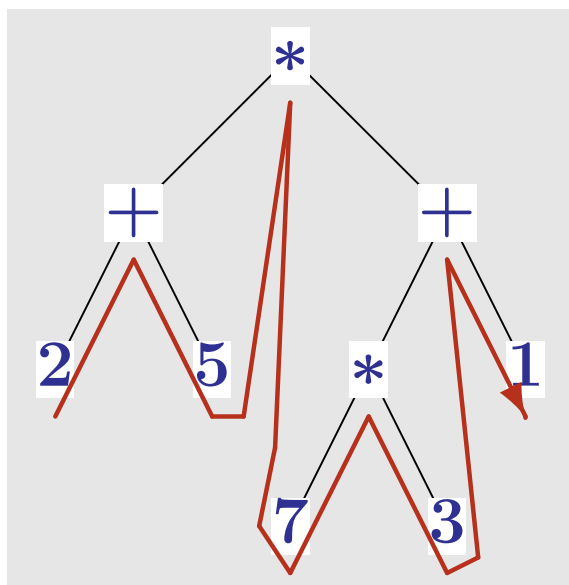


prefix — notacja
Łukasiewicza:

$$* + 2$$

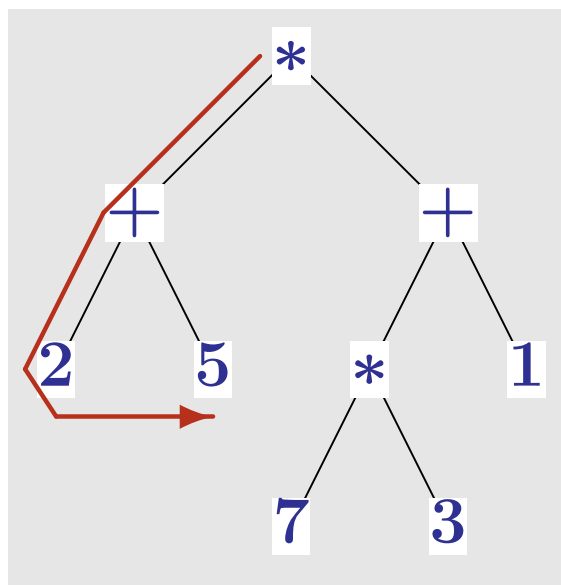
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$

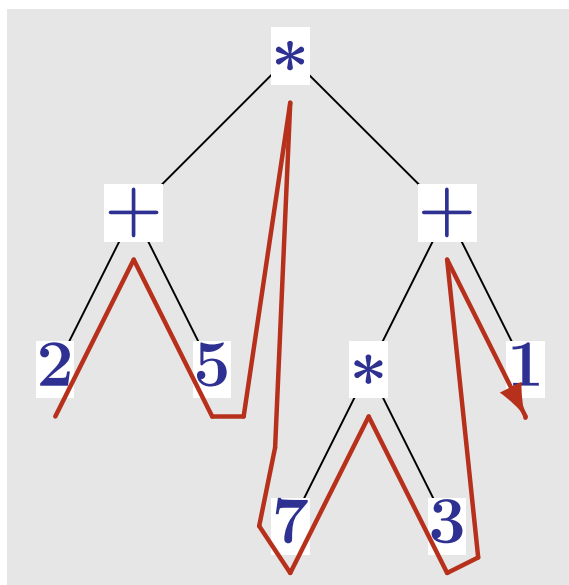


prefix — notacja
Łukasiewicza:

$$* + 2 5$$

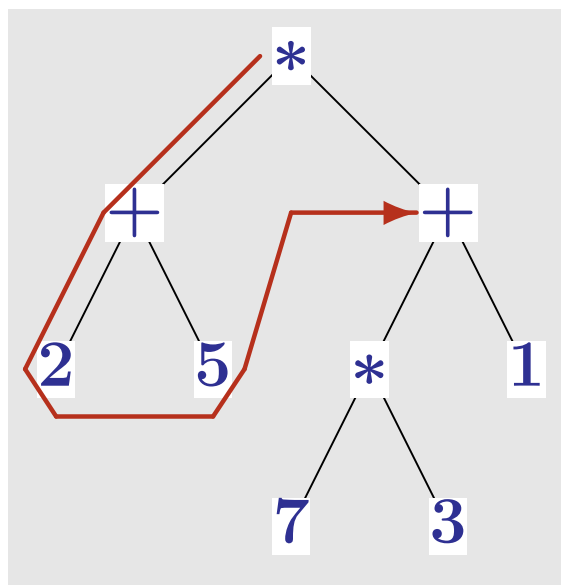
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$

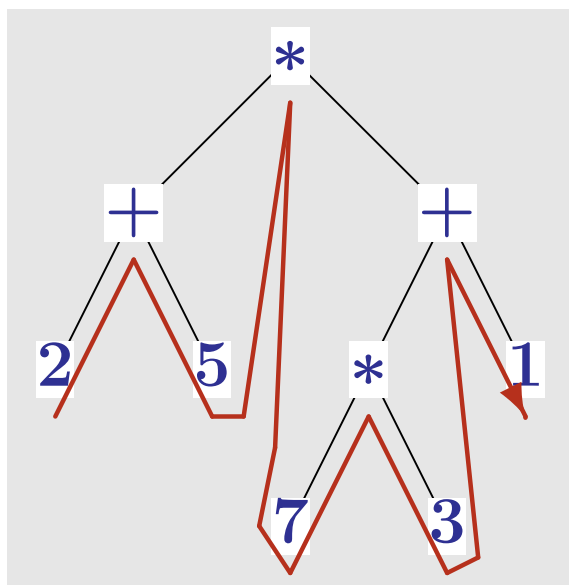


prefix — notacja
Łukasiewicza:

$$* + 2 5 +$$

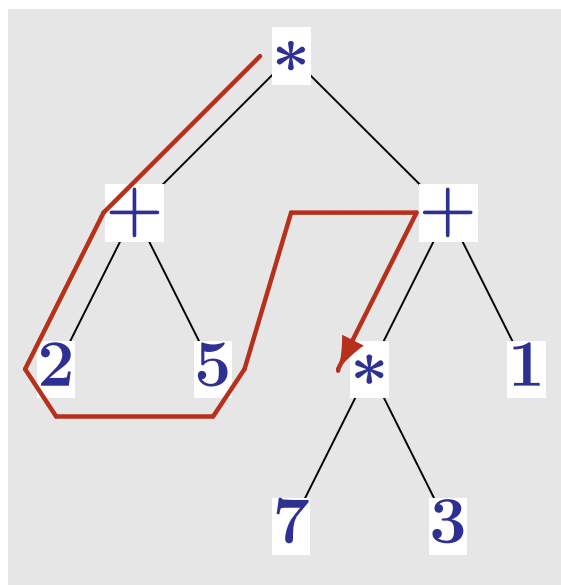
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$

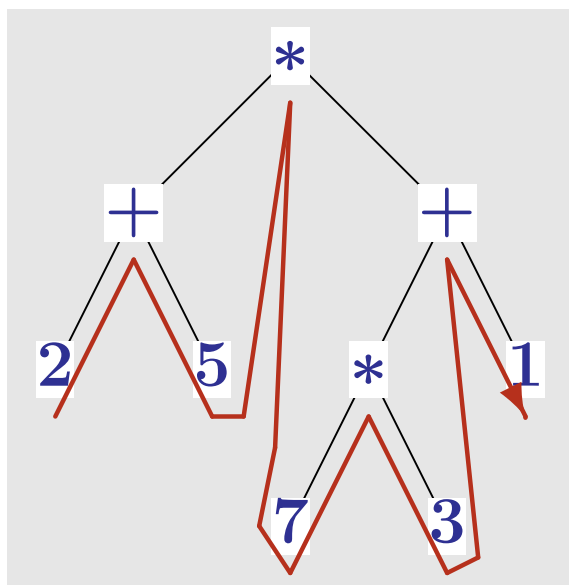


prefix — notacja
Łukasiewicza:

$$* + 2 5 + * 7 3 1$$

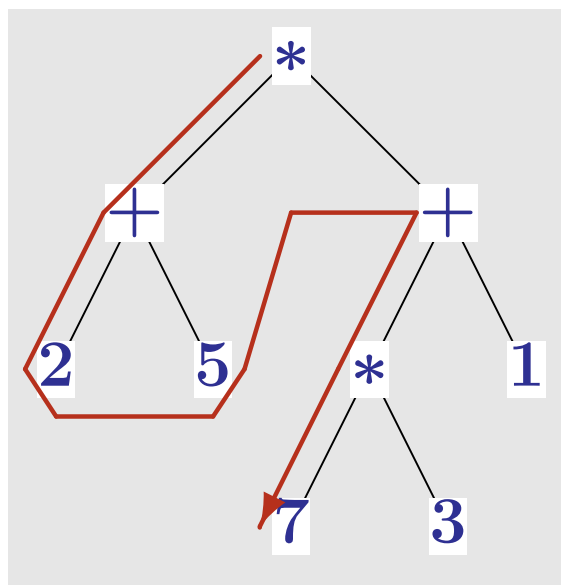
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$

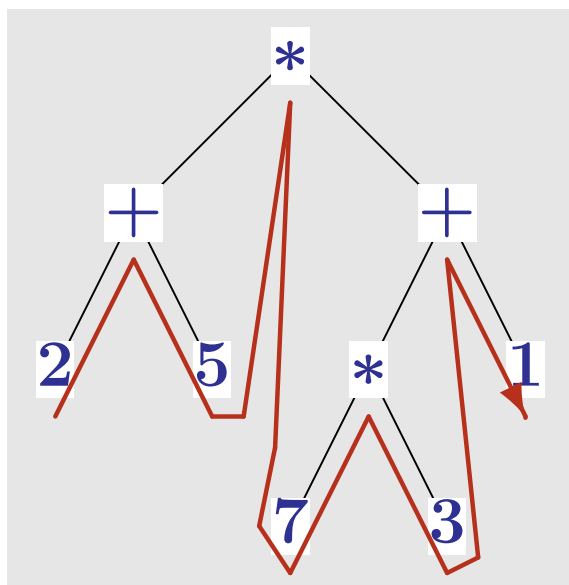


prefix — notacja
Łukasiewicza:

$$* + 2 5 + * 7$$

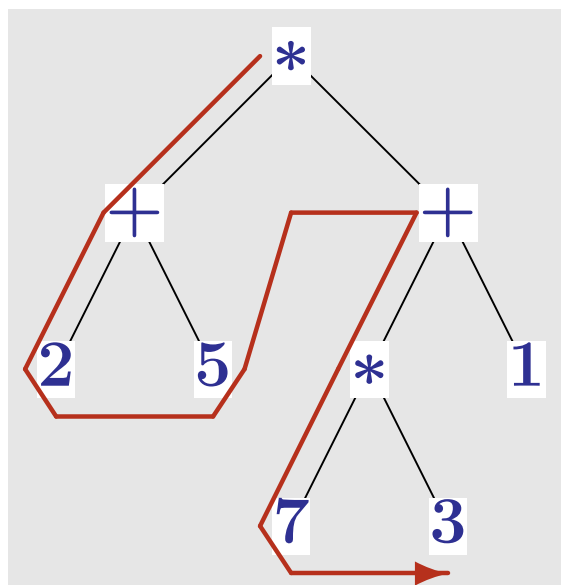
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$

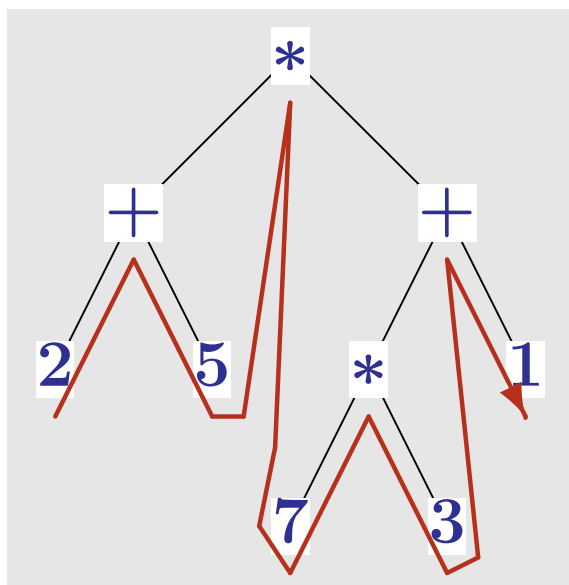


prefix — notacja
Łukasiewicza:

$$* + 2 5 + * 7 3$$

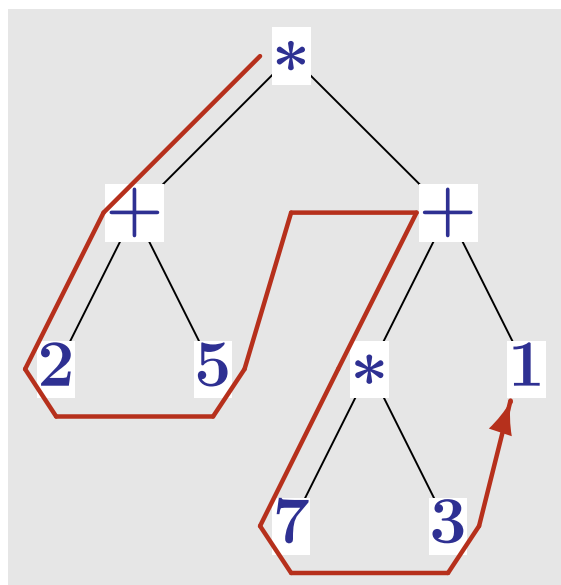
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$

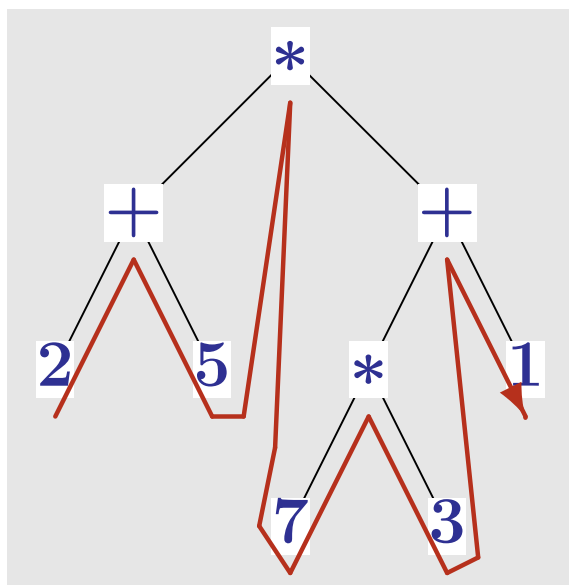


prefix — notacja
Łukasiewicza:

$$* + 2 5 + * 7 3 1$$

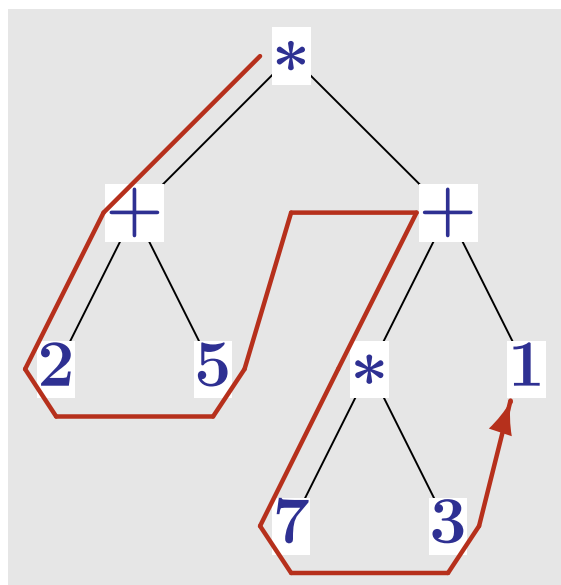
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



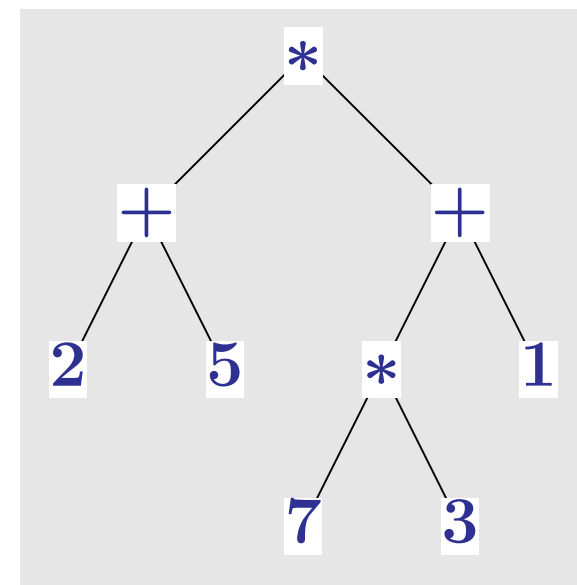
infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$



prefix — notacja
Łukasiewicza:

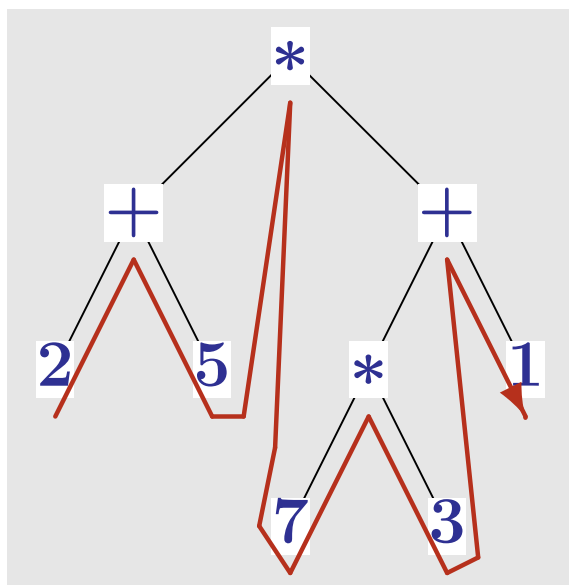
$$* + 2 5 + * 7 3 1$$



postfix — notacja
odwrotna polska

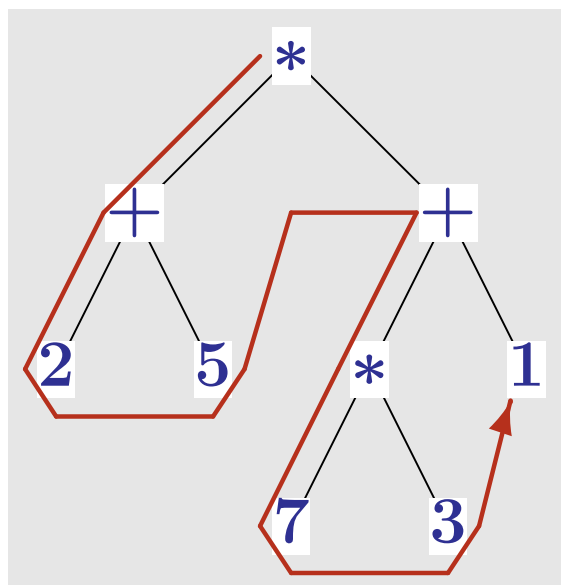
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



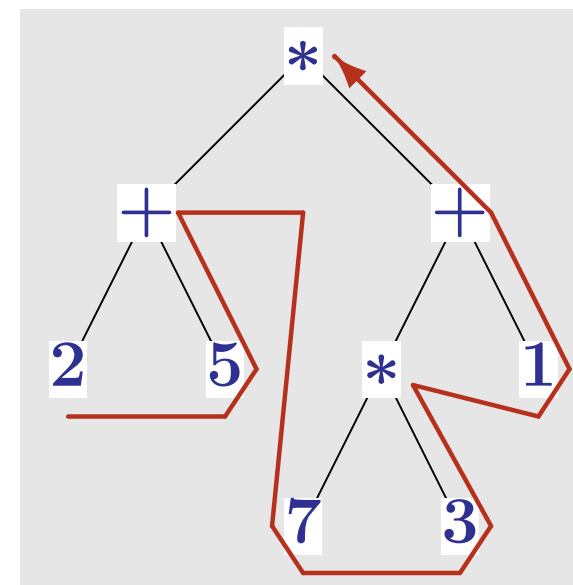
infix — notacja
zwykła, **nawiasowa**:

$$(2 + 5) * (7 * 3 + 1)$$



prefix — notacja
Łukasiewicza:

$$* + 2 5 + * 7 3 1$$

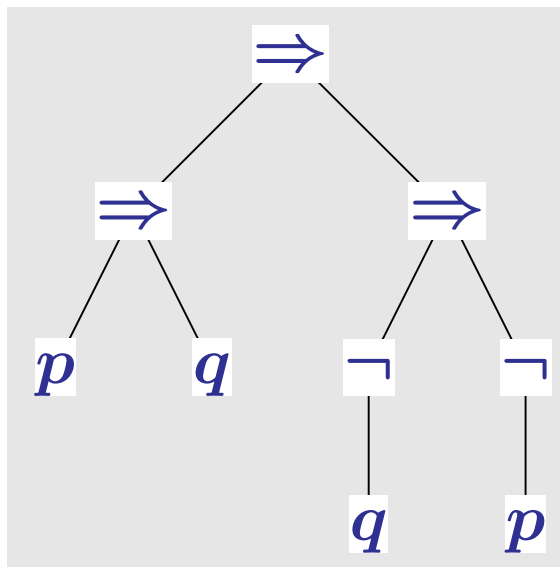


postfix — notacja
odwrotna polska:

$$2 5 + 7 3 * 1 + *$$

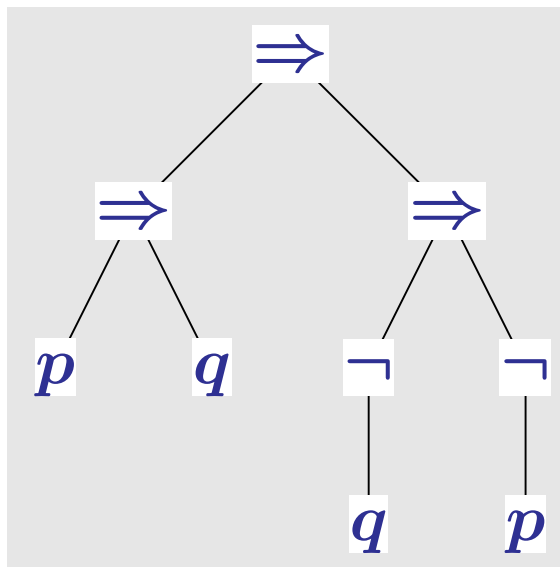
Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (wyrażenia



Notacja nawiasowa i beznawiasowa

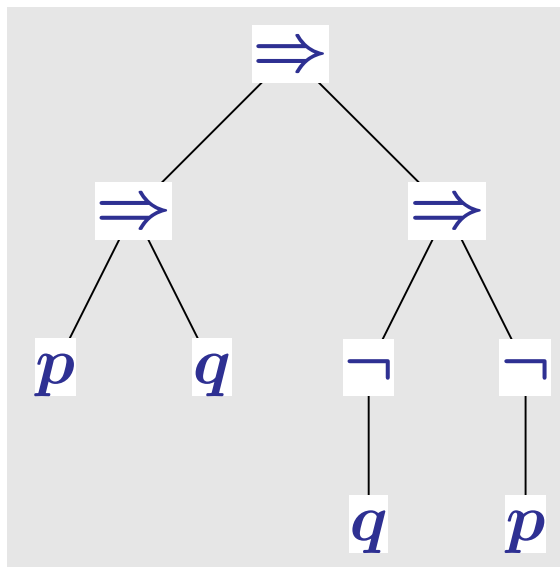
PRZYKŁAD: (wyrażenia



Zwykła notacja (*infix*): $(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$

Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (wyrażenia

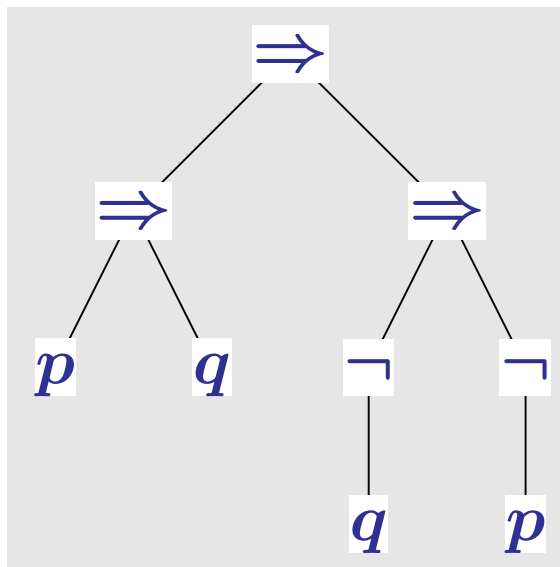


Zwykła notacja (*infix*): $(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$

Notacja Łukasiewicza (*prefix*): $\Rightarrow \Rightarrow p \ q \Rightarrow \neg q \neg p$

Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (wyrażenia



Zwykła notacja (*infix*): $(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$

Notacja Łukasiewicza (*prefix*): $\Rightarrow \Rightarrow p \ q \Rightarrow \neg q \neg p$

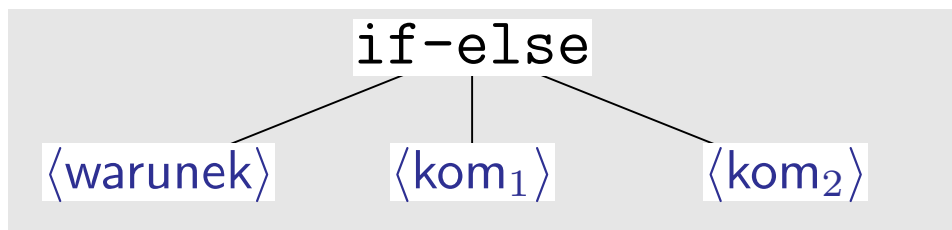
Odwrotna notacja polska (*postfix*): $p \ q \Rightarrow \ q \neg \ p \neg \Rightarrow \Rightarrow$

Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (operatory inne niż binarne)

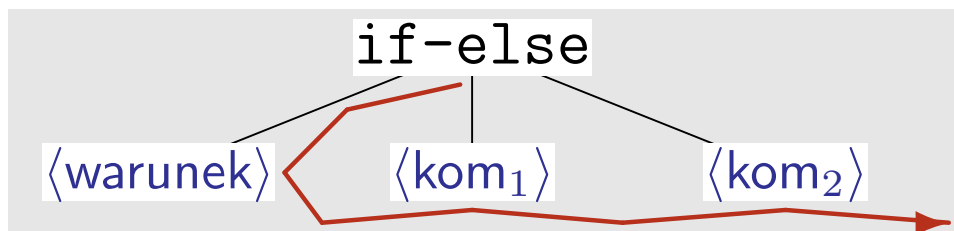
Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (operatory inne niż binarne)



Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (operatory inne niż binarne)

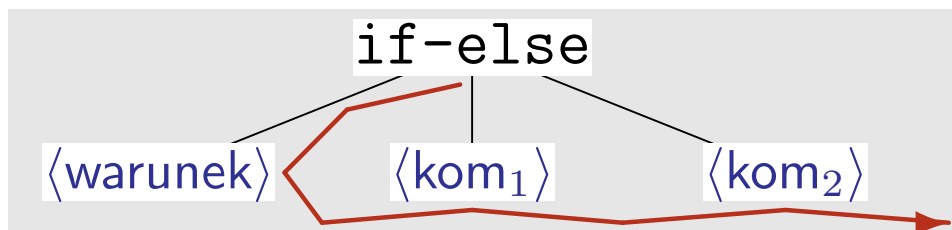


prefix:

if-else <warunek> <kom₁> <kom₂>

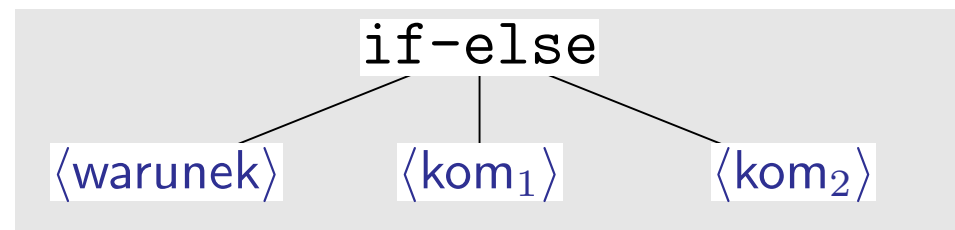
Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (operatory inne niż binarne)



prefix:

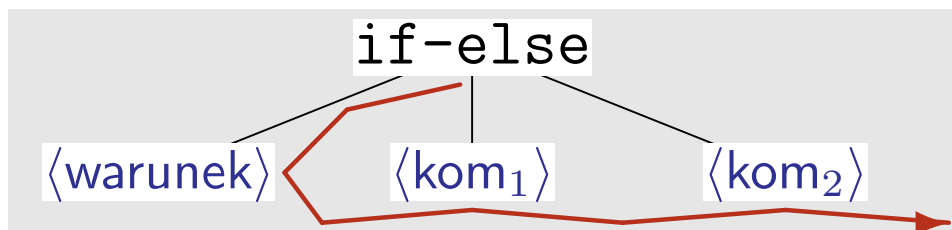
if-else <warunek> <kom₁> <kom₂>



postfix

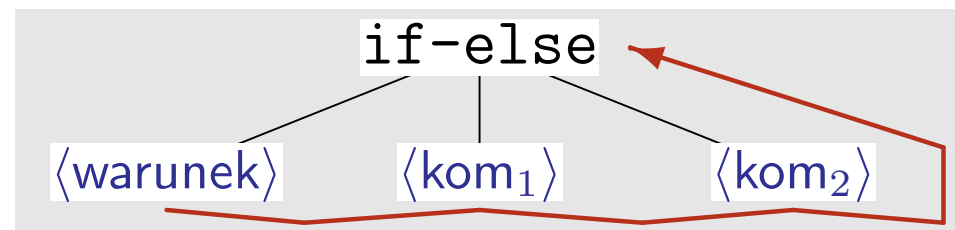
Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (operatory inne niż binarne)



prefix:

if-else <warunek> <kom₁> <kom₂>

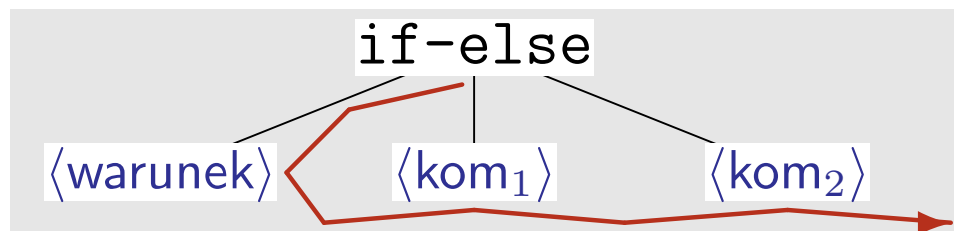


postfix:

<warunek> <kom₁> <kom₂> if-else

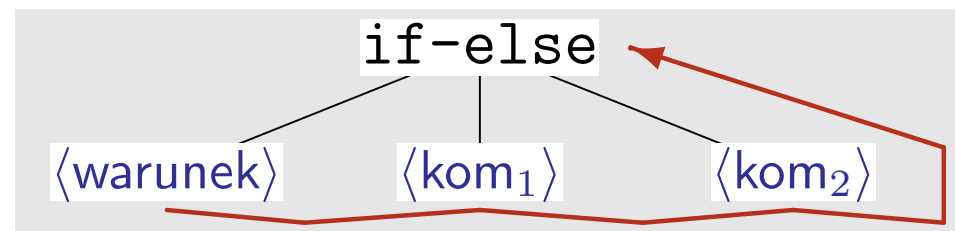
Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (operatory inne niż binarne)



prefix:

if-else <warunek> <kom₁> <kom₂>



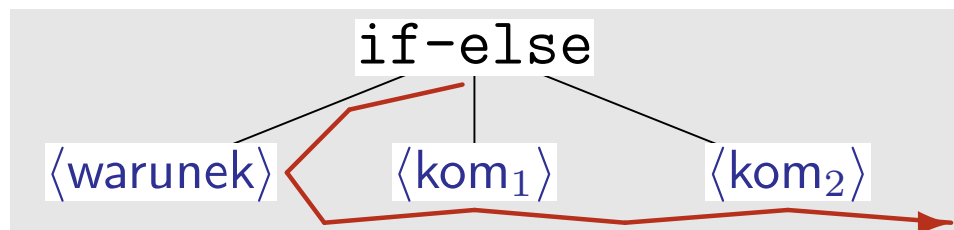
postfix:

<warunek> <kom₁> <kom₂> if-else

infix

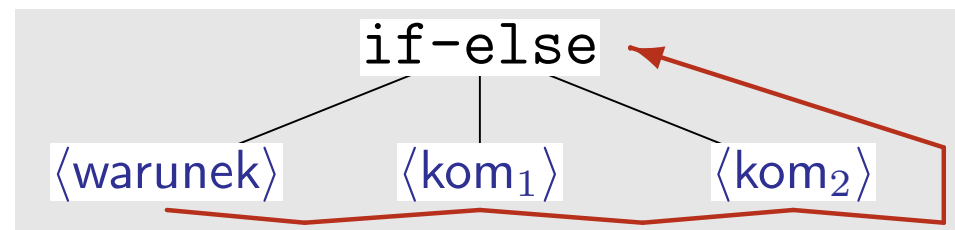
Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (operatory inne niż binarne)



prefix:

if-else <warunek> <kom₁> <kom₂>



postfix:

<warunek> <kom₁> <kom₂> if-else

infix:

nie da się zapisać nie rozbijając operatora (*distfix*)

Obliczanie wyrażeń w odwrotnej notacji polskiej

Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $\Delta(op)$.

Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```
do {  
    jeśli na wejściu jest liczba  $\ell$ ,  
        to przełóż ją z wejścia na stos;  
    jeśli na wejściu jest operator  $op$ ,  
        to {  
            zdejmij ze stosu  $A(op)$  liczb,  
            zastosuj do nich operację  $op$ ,  
            jej wynik włóż na stos;  
        }  
} while (jest jeszcze coś na wejściu);
```

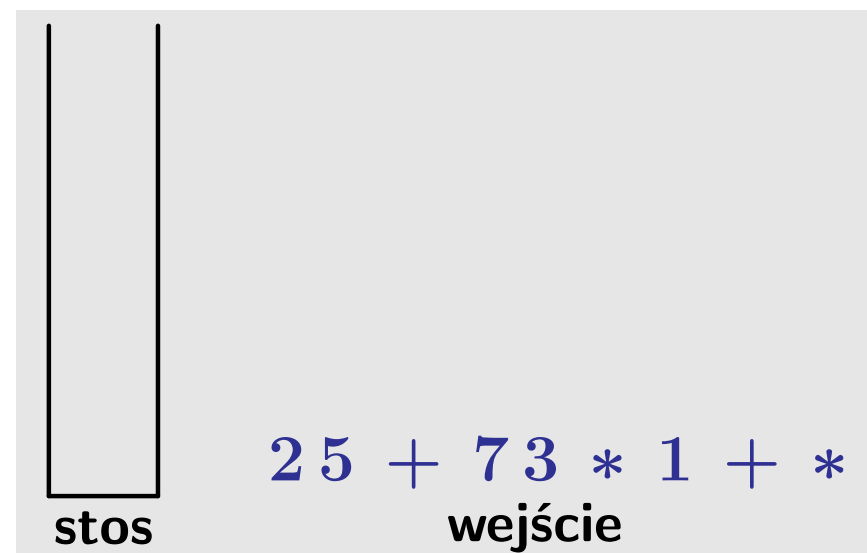
Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```
do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);
```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

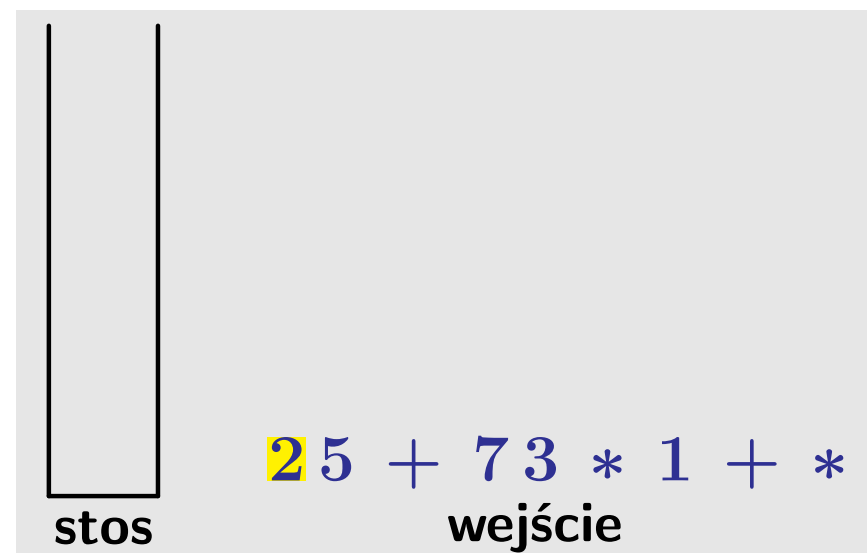
Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```

do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);

```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

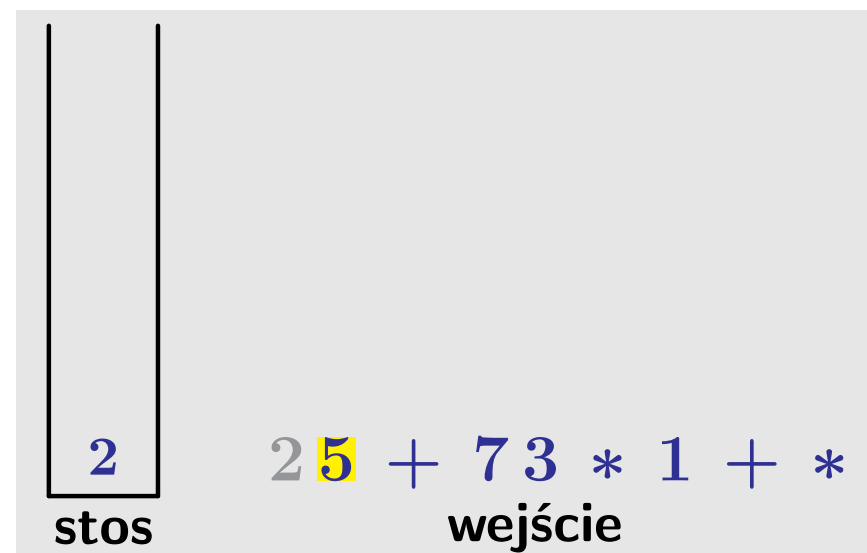
Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```

do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);

```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

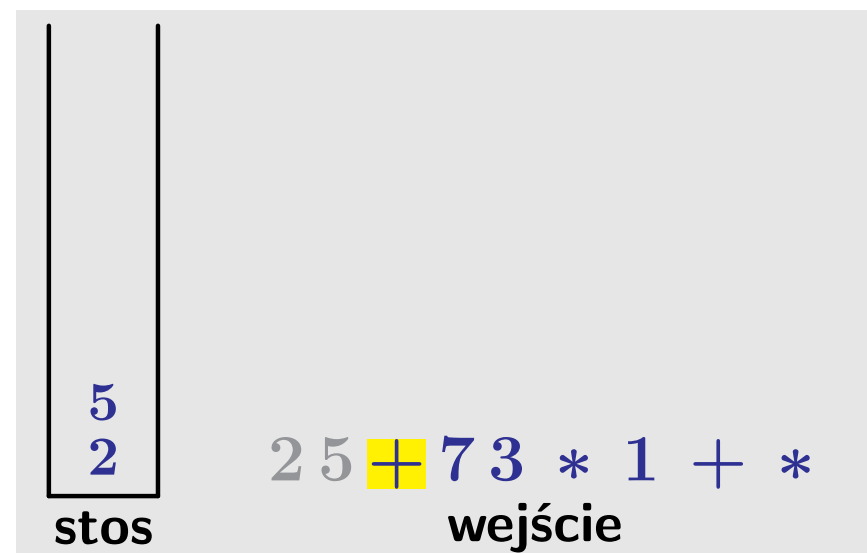
Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```

do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);

```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

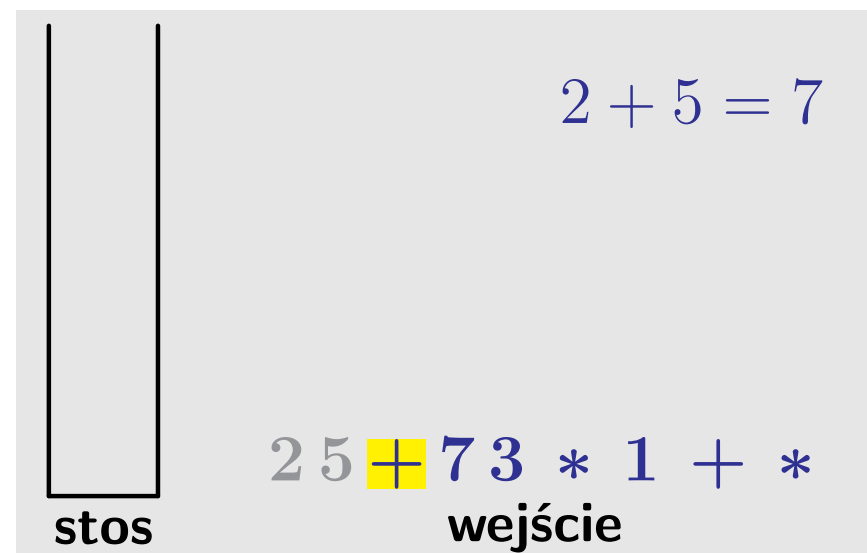
Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```

do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);

```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

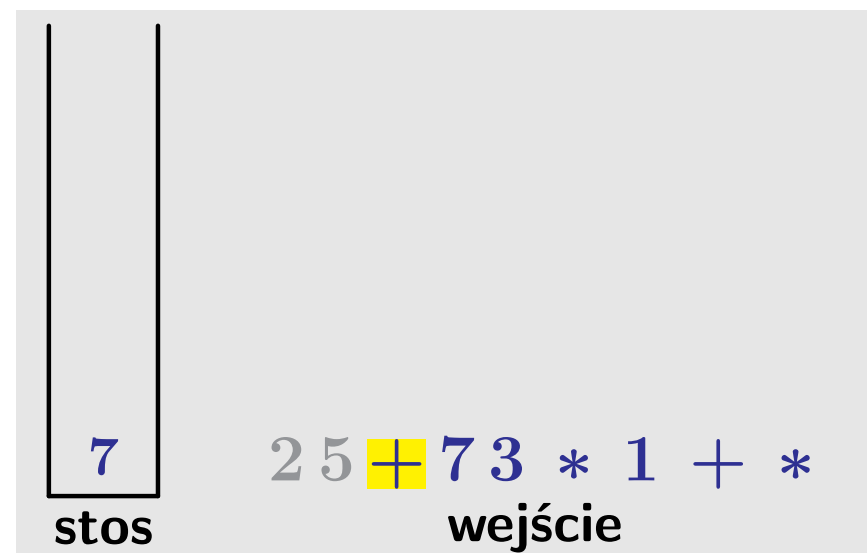
Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```

do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);

```



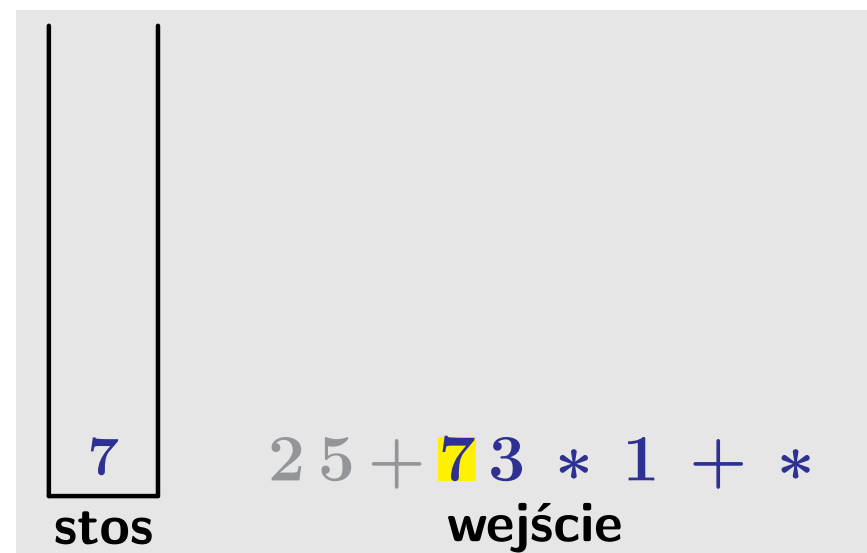
Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```
do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);
```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```

do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);

```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

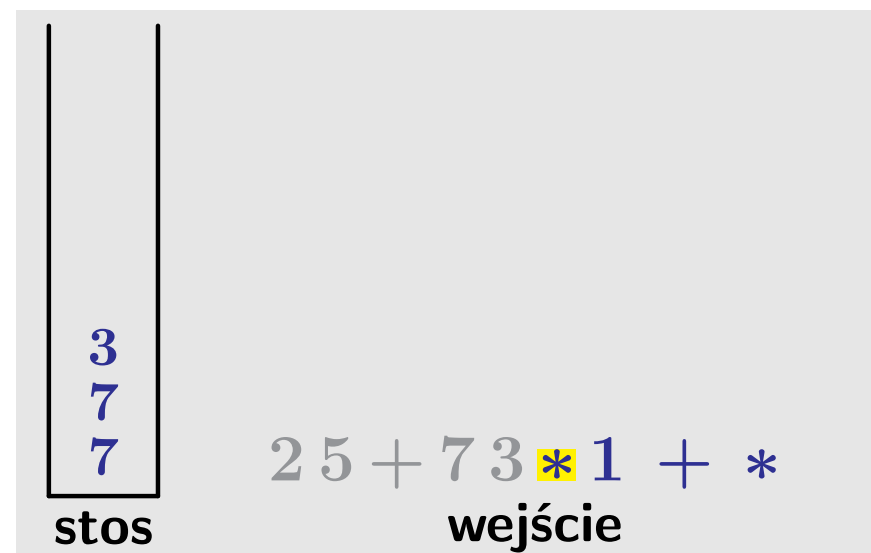
Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```

do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);

```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

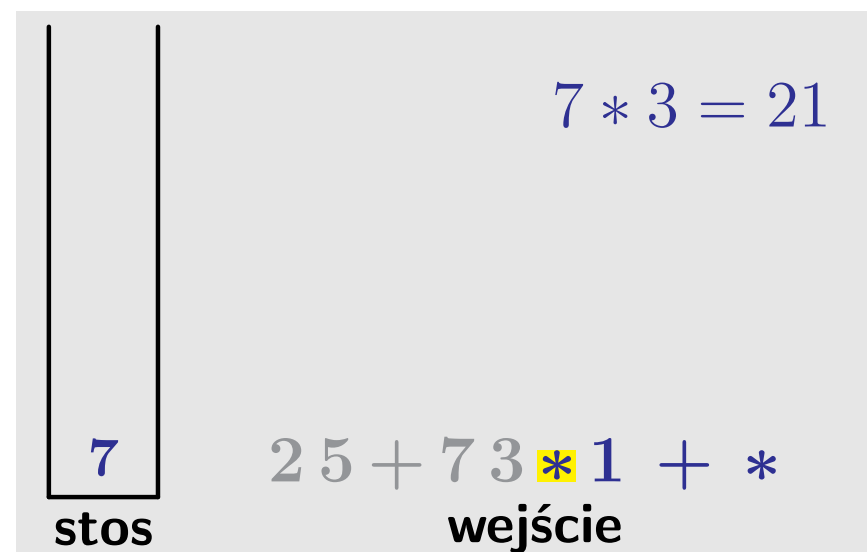
Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```

do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);

```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

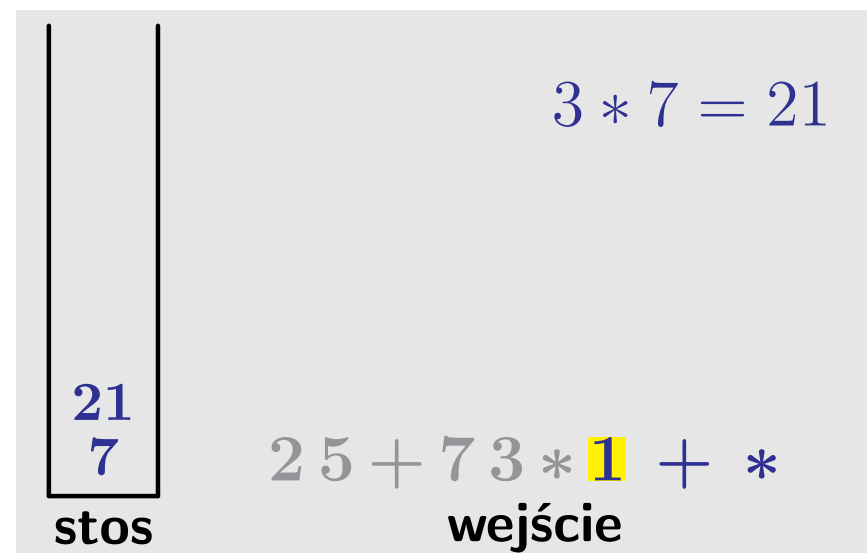
Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```

do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);

```



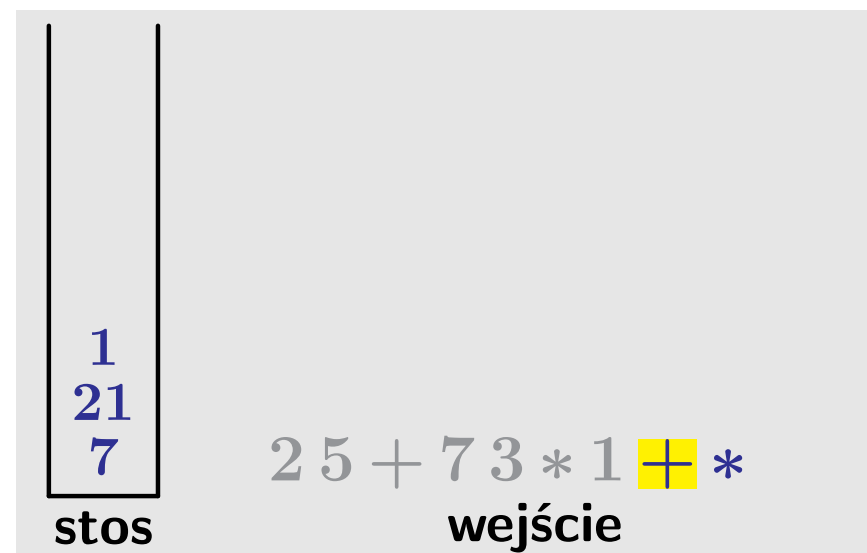
Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```
do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);
```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

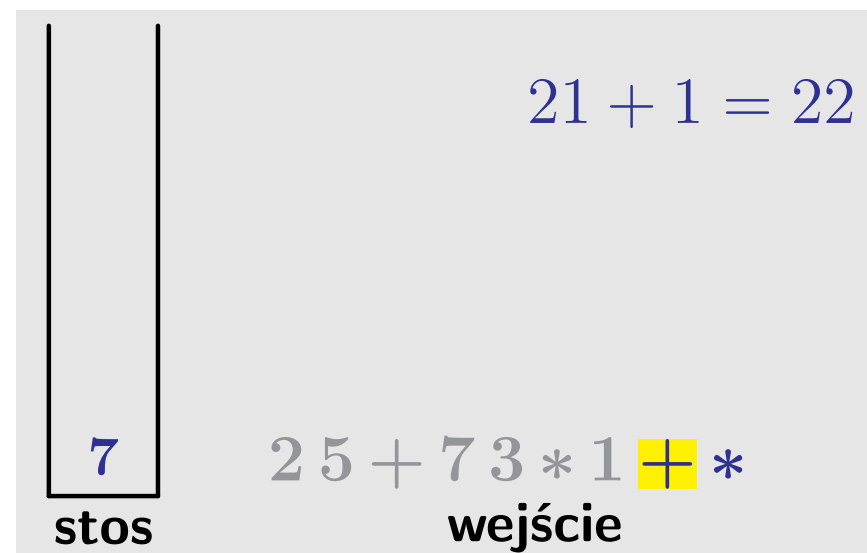
Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```

do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);

```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```

do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);

```



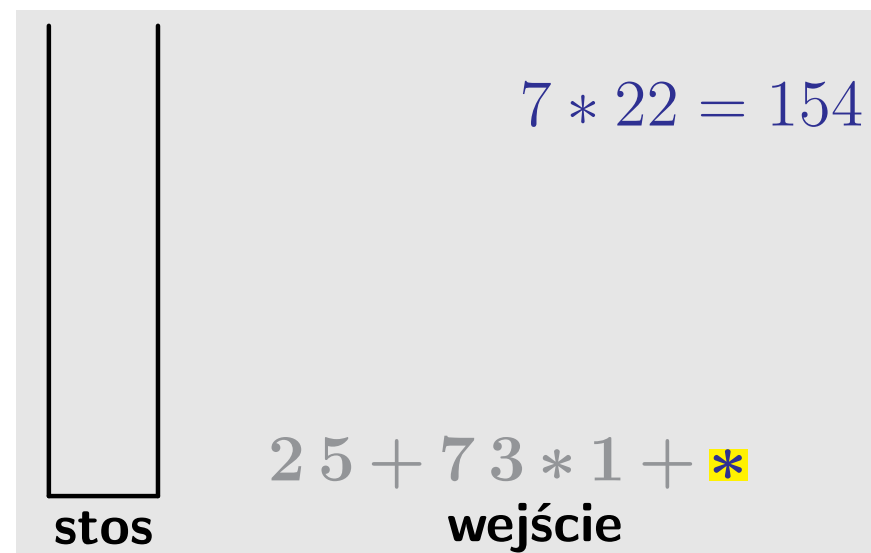
Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```
do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);
```



Obliczanie wyrażeń w odwrotnej notacji polskiej

Założenie: każdy operator op „wie” ile potrzebuje argumentów: $A(op)$.

Np. $A(+) = 2$, $A(\neg) = 1$, $A(\text{if}(\dots) \dots \text{else} \dots) = 3$.

Algorytm:

```
do {
  jeśli na wejściu jest liczba  $\ell$ ,
    to przełóż ją z wejścia na stos;
  jeśli na wejściu jest operator  $op$ ,
    to {
      zdejmij ze stosu  $A(op)$  liczb,
      zastosuj do nich operację  $op$ ,
      jej wynik włóż na stos;
    }
} while (jest jeszcze coś na wejściu);
```



Działania na językach

— takie jak podstawowe konstrukcje wyrażeń regularnych

Działania na językach

— takie jak podstawowe konstrukcje wyrażeń regularnych:

1. Suma:

$$L \mid M \stackrel{\text{def}}{=} \left\{ w \mid w \in L \text{ lub } w \in M \right\}$$

Działania na językach

— takie jak podstawowe konstrukcje wyrażeń regularnych:

1. Suma:

$$L \mid M \stackrel{\text{def}}{=} \left\{ w \mid w \in L \text{ lub } w \in M \right\}$$

2. Złączenie:

$$L M \stackrel{\text{def}}{=} \left\{ uv \mid u \in L \text{ i } v \in M \right\}$$

Działania na językach

— takie jak podstawowe konstrukcje wyrażeń regularnych:

1. Suma:

$$L \mid M \stackrel{\text{def}}{=} \left\{ w \mid w \in L \text{ lub } w \in M \right\}$$

2. Złączenie:

$$L M \stackrel{\text{def}}{=} \left\{ uv \mid u \in L \text{ i } v \in M \right\}$$

3. Iteracja (domknięcie Kleenego):

$$L^* \stackrel{\text{def}}{=} \left\{ u_1 u_2 \dots u_n \mid u_1, u_2, \dots, u_n \in L \text{ i } n \geq 0 \right\}$$

Działania na językach

— takie jak podstawowe konstrukcje wyrażeń regularnych:

1. Suma:

$$L \mid M \stackrel{\text{def}}{=} \{w \mid w \in L \text{ lub } w \in M\}$$

2. Złączenie:

$$L M \stackrel{\text{def}}{=} \{uv \mid u \in L \text{ i } v \in M\}$$

3. Iteracja (domknięcie Kleenego):

$$\begin{aligned} L^* &\stackrel{\text{def}}{=} \{u_1 u_2 \dots u_n \mid u_1, u_2, \dots, u_n \in L \text{ i } n \geq 0\} \\ &= \{\lambda\} \mid L \mid L^2 \mid L^3 \mid \dots \end{aligned}$$

Działania na językach

PRZYKŁAD: (suma)

$$L \mid M \stackrel{\text{def}}{=} \{w \mid w \in L \text{ lub } w \in M\}$$

Działania na językach

PRZYKŁAD: (suma)

$$L \mid M \stackrel{\text{def}}{=} \{w \mid w \in L \text{ lub } w \in M\}$$

- $\{\text{czerwony, zielony}\} \mid \{\text{dom, kapelusz, rower}\}$

Działania na językach

PRZYKŁAD: (suma)

$$L \mid M \stackrel{\text{def}}{=} \{w \mid w \in L \text{ lub } w \in M\}$$

- $\{\text{czerwony, zielony}\} \mid \{\text{dom, kapelusz, rower}\}$
= $\{\text{czerwony, zielony, dom, kapelusz, rower}\}$

Działania na językach

PRZYKŁAD: (suma)

$$L \mid M \stackrel{\text{def}}{=} \{w \mid w \in L \text{ lub } w \in M\}$$

- $\{\text{czerwony, zielony}\} \mid \{\text{dom, kapelusz, rower}\}$
= $\{\text{czerwony, zielony, dom, kapelusz, rower}\}$
- $\{\lambda, a, a^2, a^3, \dots\} \mid \{\lambda, b, b^2, b^3, \dots\} = \{\lambda, a, b, a^2, b^2, a^3, b^3, \dots\}$

Działania na językach

PRZYKŁAD: (suma)

$$L \mid M \stackrel{\text{def}}{=} \{w \mid w \in L \text{ lub } w \in M\}$$

- $\{\text{czerwony, zielony}\} \mid \{\text{dom, kapelusz, rower}\}$
 $= \{\text{czerwony, zielony, dom, kapelusz, rower}\}$
- $\{\lambda, a, a^2, a^3, \dots\} \mid \{\lambda, b, b^2, b^3, \dots\} = \{\lambda, a, b, a^2, b^2, a^3, b^3, \dots\}$

PRZYKŁAD: (złączenie)

$$L M \stackrel{\text{def}}{=} \{uv \mid u \in L \text{ i } v \in M\}$$

Działania na językach

PRZYKŁAD: (suma)

$$L \mid M \stackrel{\text{def}}{=} \{w \mid w \in L \text{ lub } w \in M\}$$

- $\{\text{czerwony, zielony}\} \mid \{\text{dom, kapelusz, rower}\}$
 $= \{\text{czerwony, zielony, dom, kapelusz, rower}\}$
- $\{\lambda, a, a^2, a^3, \dots\} \mid \{\lambda, b, b^2, b^3, \dots\} = \{\lambda, a, b, a^2, b^2, a^3, b^3, \dots\}$

PRZYKŁAD: (złączenie)

$$L M \stackrel{\text{def}}{=} \{uv \mid u \in L \text{ i } v \in M\}$$

- $\{\text{czerwony, zielony}\} \{\text{dom, kapelusz, rower}\}$

Działania na językach

PRZYKŁAD: (suma)

$$L \mid M \stackrel{\text{def}}{=} \{w \mid w \in L \text{ lub } w \in M\}$$

- $\{\text{czerwony, zielony}\} \mid \{\text{dom, kapelusz, rower}\}$
 $= \{\text{czerwony, zielony, dom, kapelusz, rower}\}$
- $\{\lambda, a, a^2, a^3, \dots\} \mid \{\lambda, b, b^2, b^3, \dots\} = \{\lambda, a, b, a^2, b^2, a^3, b^3, \dots\}$

PRZYKŁAD: (złączenie)

$$L M \stackrel{\text{def}}{=} \{uv \mid u \in L \text{ i } v \in M\}$$

- $\{\text{czerwony, zielony}\} \{\text{dom, kapelusz, rower}\}$
 $= \left\{ \begin{array}{l} \text{czerwondom, czerwonykapelusz, czerwonyrower,} \\ \text{zielondom, zielonykapelusz, zielonyrower} \end{array} \right\}$

Działania na językach

PRZYKŁAD: (suma)

$$L \mid M \stackrel{\text{def}}{=} \{w \mid w \in L \text{ lub } w \in M\}$$

- $\{\text{czerwony, zielony}\} \mid \{\text{dom, kapelusz, rower}\}$
 $= \{\text{czerwony, zielony, dom, kapelusz, rower}\}$
- $\{\lambda, a, a^2, a^3, \dots\} \mid \{\lambda, b, b^2, b^3, \dots\} = \{\lambda, a, b, a^2, b^2, a^3, b^3, \dots\}$

PRZYKŁAD: (złączenie)

$$L M \stackrel{\text{def}}{=} \{uv \mid u \in L \text{ i } v \in M\}$$

- $\{\text{czerwony, zielony}\} \{\text{dom, kapelusz, rower}\}$
 $= \left\{ \begin{array}{l} \text{czerwondom, czerwonykapelusz, czerwonyrower,} \\ \text{zielondom, zielonykapelusz, zielonyrower} \end{array} \right\}$
- $\{\lambda, a, a^2, a^3, \dots\} \{\lambda, b, b^2, b^3, \dots\}$
 $= \{\lambda, a, b, a^2, ab, b^2, a^3, a^2b, ab^2, b^3, \dots\}$

Działania na językach

Własności działań

Działania na językach

Własności działań:

- Łączność:

$$K \mid (L \mid M) = (K \mid L) \mid M \qquad K(LM) = (KL)M$$

Działania na językach

Własności działań:

- Łączność:

$$K \mid (L \mid M) = (K \mid L) \mid M \qquad K(LM) = (KL)M$$

- Przemienność:

$$L \mid M = M \mid L$$

Uwaga: złączenie nie jest przemienne!

Działania na językach

Własności działań:

- Łączność:

$$K \mid (L \mid M) = (K \mid L) \mid M \qquad K(LM) = (KL)M$$

- Przemienność:

$$L \mid M = M \mid L$$

Uwaga: złączenie nie jest przemienne!

- Elementy neutralne:

$$M \mid \emptyset = M \qquad \Lambda M = M \Lambda = M$$

$\emptyset \stackrel{\text{def}}{=} \{\}$ — język pusty

$\Lambda \stackrel{\text{def}}{=} \{\lambda\}$ — język zawierający tylko słowo puste

Działania na językach

Własności działań:

- Łączność:

$$K \mid (L \mid M) = (K \mid L) \mid M \qquad K(LM) = (KL)M$$

- Przemienność:

$$L \mid M = M \mid L$$

Uwaga: złączenie nie jest przemienne!

- Elementy neutralne:

$$M \mid \emptyset = M \qquad \Lambda M = M \Lambda = M$$

$\emptyset \stackrel{\text{def}}{=} \{\}$ — język pusty

$\Lambda \stackrel{\text{def}}{=} \{\lambda\}$ — język zawierający tylko słowo puste

- Rozdzielność (wchodzenie pod nawias, wyłączanie przed/za nawias):

$$K(L \mid M) = KL \mid KM \qquad (L \mid M)K = LK \mid MK$$

Działania na językach

Podobieństwa do arytmetyki

Działania na językach

Podobieństwa do arytmetyki:

- **suma** języków jest jak **dodawanie** liczb

Działania na językach

Podobieństwa do arytmetyki:

- **suma** języków jest jak **dodawanie** liczb,
język $\emptyset$ jest jak liczba 0

Działania na językach

Podobieństwa do arytmetyki:

- **suma** języków jest jak **dodawanie** liczb, język $\emptyset$ jest jak liczba 0 ;
- **złączenie** języków jest jak **mnożenie** liczb

Działania na językach

Podobieństwa do arytmetyki:

- **suma** języków jest jak **dodawanie** liczb, język $\emptyset$ jest jak liczba 0 ;
- **złączenie** języków jest jak **mnożenie** liczb, język Λ jest jak liczba 1 .

Działania na językach

Podobieństwa do arytmetyki:

- **suma** języków jest jak **dodawanie** liczb, język $\emptyset$ jest jak liczba 0 ;
- **złączenie** języków jest jak **mnożenie** liczb, język Λ jest jak liczba 1 .

Różnice:

- złączenie języków *nie jest* przemienne

Działania na językach

Podobieństwa do arytmetyki:

- **suma** języków jest jak **dodawanie** liczb, język $\emptyset$ jest jak liczba 0 ;
- **złączenie** języków jest jak **mnożenie** liczb, język Λ jest jak liczba 1 .

Różnice:

- złączenie języków *nie jest* przemienne;
- *nie ma* działań odwrotnych do sumy ani złączenia (czyli odpowiedników odejmowania i dzielenia).

Przekształcenia gramatyk

Gramatyka jako układ równań na językach

Przekształcenia gramatyk

Gramatyka jako układ równań na językach

Interpretujemy:

- *nieterminale* jako języki złożone z tych słów, które dają się z nich wyprowadzić

Przekształcenia gramatyk

Gramatyka jako układ równań na językach

Interpretujemy:

- *nieterminale* jako języki złożone z tych słów, które dają się z nich wyprowadzić;
- *terminale* jako języki 1-literowe

Przekształcenia gramatyk

Gramatyka jako układ równań na językach

Interpretujemy:

- *nieterminale* jako języki złożone z tych słów, które dają się z nich wyprowadzić;
- *terminale* jako języki 1-literowe;
- *znak* $::=$ jako równość języków.

Przekształcenia gramatyk

Gramatyka jako układ równań na językach

Interpretujemy:

- *nieterminale* jako języki złożone z tych słów, które dają się z nich wyprowadzić;
- *terminale* jako języki 1-literowe;
- *znak* $::=$ jako równość języków.

Wtedy gramatyka staje się *układem równań* na językach, przy czym nieterminale są jego *niewiadomymi*.

Przekształcenia gramatyk

Gramatyka jako układ równań na językach

Interpretujemy:

- *nieterminale* jako języki złożone z tych słów, które dają się z nich wyprowadzić;
- *terminale* jako języki 1-literowe;
- *znak* ::= jako równość języków.

Wtedy gramatyka staje się *układem równań* na językach, przy czym nieterminale są jego *niewiadomymi*.

Przekształcanie gramatyk do postaci równoważnych odpowiada wykonywaniu działań na językach, określonych tymi układami.

Przekształcenia gramatyk

Gramatyka jako układ równań na językach

Interpretujemy:

- *nieterminale* jako języki złożone z tych słów, które dają się z nich wyprowadzić;
- *terminale* jako języki 1-literowe;
- *znak ::=* jako równość języków.

Wtedy gramatyka staje się *układem równań* na językach, przy czym nieterminale są jego *niewiadomymi*.

Przekształcanie gramatyk do postaci równoważnych odpowiada wykonywaniu działań na językach, określonych tymi układami.

Przykład:

$\langle S \rangle$	$::=$	a	$ $	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$ $	$\langle S \rangle$

Przekształcenia gramatyk

Gramatyka jako układ równań na językach

Interpretujemy:

- *nieterminale* jako języki złożone z tych słów, które dają się z nich wyprowadzić;
- *terminale* jako języki 1-literowe;
- *znak ::=* jako równość języków.

Wtedy gramatyka staje się *układem równań* na językach, przy czym nieterminale są jego *niewiadomymi*.

Przekształcanie gramatyk do postaci równoważnych odpowiada wykonywaniu działań na językach, określonych tymi układami.

Przykład:

$$\begin{array}{|lcl} \langle S \rangle & ::= & a \mid b\langle T \rangle c \\ \langle T \rangle & ::= & d \mid \langle S \rangle \end{array} \longrightarrow \begin{array}{|lcl} S & = & a \mid bTc \\ T & = & d \mid S \end{array}$$

Przekształcenia gramatyk

Gramatyka jako układ równań na językach

Interpretujemy:

- *nieterminale* jako języki złożone z tych słów, które dają się z nich wyprowadzić;
- *terminale* jako języki 1-literowe;
- *znak ::=* jako równość języków.

Wtedy gramatyka staje się *układem równań* na językach, przy czym nieterminale są jego *niewiadomymi*.

Przekształcanie gramatyk do postaci równoważnych odpowiada wykonywaniu działań na językach, określonych tymi układami.

Przykład:

$$\begin{array}{|l|l|l|} \hline \langle S \rangle & ::= & a \mid b\langle T \rangle c \\ \hline \langle T \rangle & ::= & d \mid \langle S \rangle \\ \hline \end{array} \longrightarrow \begin{array}{|l|l|l|} \hline S & = & a \mid bTc \\ \hline T & = & d \mid S \\ \hline \end{array} \xrightarrow{\text{podstawienie}} \begin{array}{|l|l|l|} \hline S & = & a \mid b \{ d \mid S \} c \\ \hline \end{array}$$

Przekształcenia gramatyk

Gramatyka jako układ równań na językach

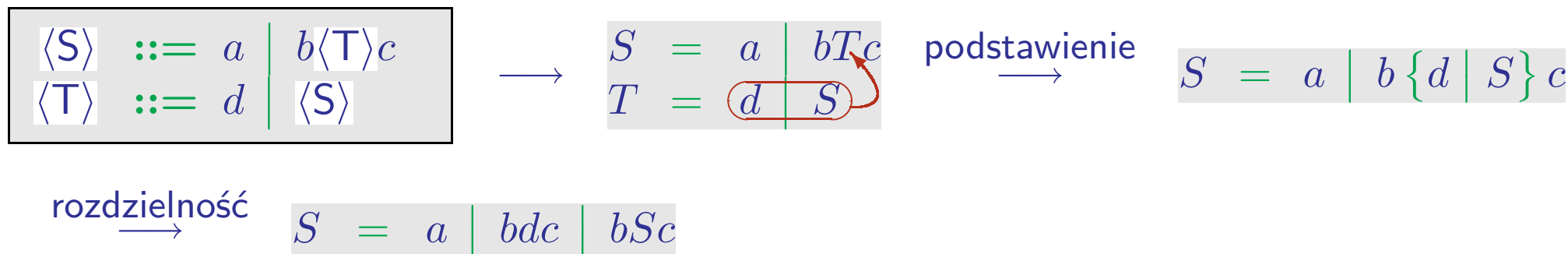
Interpretujemy:

- *nieterminale* jako języki złożone z tych słów, które dają się z nich wyprowadzić;
- *terminale* jako języki 1-literowe;
- *znak ::=* jako równość języków.

Wtedy gramatyka staje się *układem równań* na językach, przy czym nieterminale są jego *niewiadomymi*.

Przekształcanie gramatyk do postaci równoważnych odpowiada wykonywaniu działań na językach, określonych tymi układami.

Przykład:



Przekształcenia gramatyk

Gramatyka jako układ równań na językach

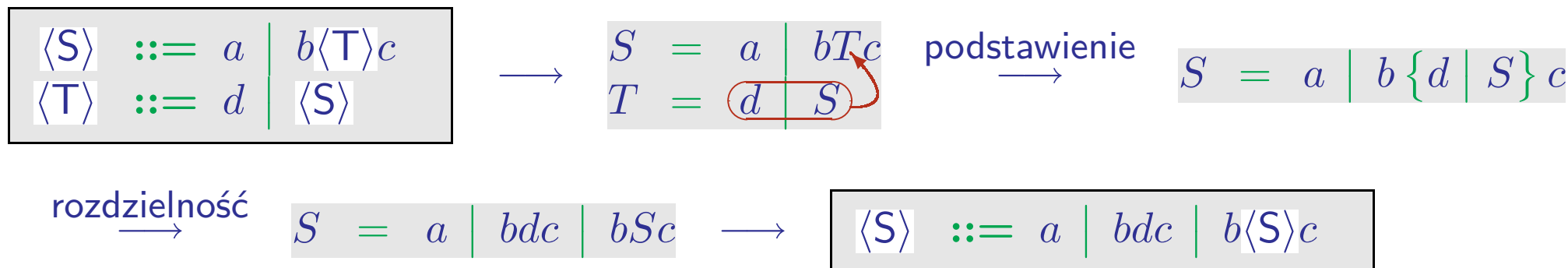
Interpretujemy:

- *nieterminale* jako języki złożone z tych słów, które dają się z nich wyprowadzić;
- *terminale* jako języki 1-literowe;
- *znak ::=* jako równość języków.

Wtedy gramatyka staje się *układem równań* na językach, przy czym nieterminale są jego *niewiadomymi*.

Przekształcanie gramatyk do postaci równoważnych odpowiada wykonywaniu działań na językach, określonych tymi układami.

Przykład:



Przekształcenia gramatyk

Wpływ przekształceń na kształt drzewa wywodu

Przekształcenia gramatyk

Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

Przekształcenia gramatyk

Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

$\langle S \rangle$	$::=$	a	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$\langle S \rangle$

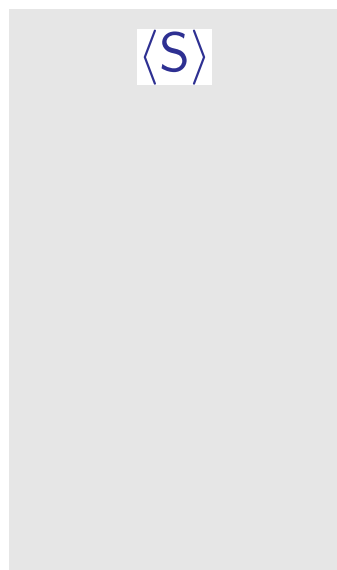
$\langle S \rangle$	$::=$	a	bdc	$b\langle S \rangle c$
---------------------	-------	-----	-------	------------------------

Przekształcenia gramatyk

Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

$\langle S \rangle$	$::=$	a	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$\langle S \rangle$



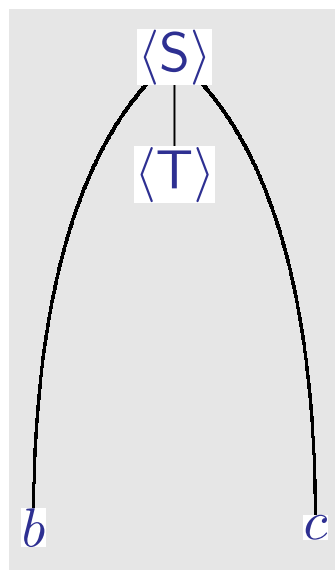
$\langle S \rangle$	$::=$	a	bdc	$b\langle S \rangle c$
---------------------	-------	-----	-------	------------------------

Przekształcenia gramatyk

Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

$\langle S \rangle$	$::=$	a	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$\langle S \rangle$



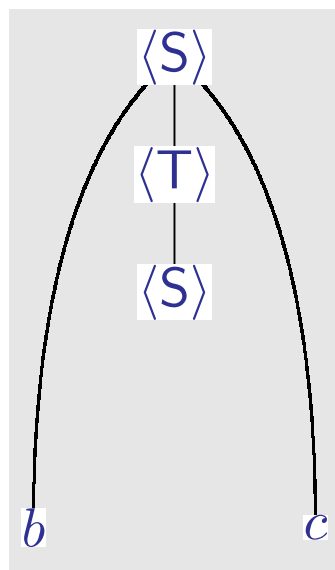
$\langle S \rangle$	$::=$	a	bdc	$b\langle S \rangle c$
---------------------	-------	-----	-------	------------------------

Przekształcenia gramatyk

Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

$\langle S \rangle$	$::=$	a	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$\langle S \rangle$



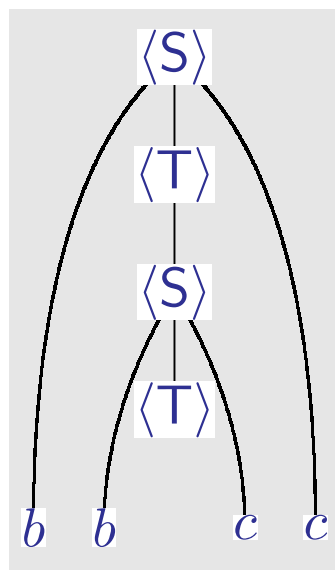
$\langle S \rangle$	$::=$	a	bdc	$b\langle S \rangle c$
---------------------	-------	-----	-------	------------------------

Przekształcenia gramatyk

Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

$\langle S \rangle$	$::=$	a	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$\langle S \rangle$



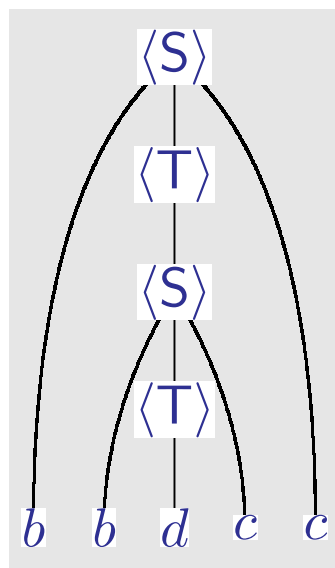
$\langle S \rangle$	$::=$	a	bdc	$b\langle S \rangle c$
---------------------	-------	-----	-------	------------------------

Przekształcenia gramatyk

Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

$\langle S \rangle$	$::=$	a	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$\langle S \rangle$



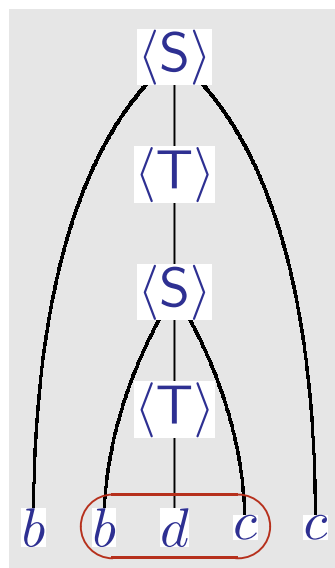
$\langle S \rangle$	$::=$	a	bdc	$b\langle S \rangle c$
---------------------	-------	-----	-------	------------------------

Przekształcenia gramatyk

Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

$\langle S \rangle$	$::=$	a	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$\langle S \rangle$



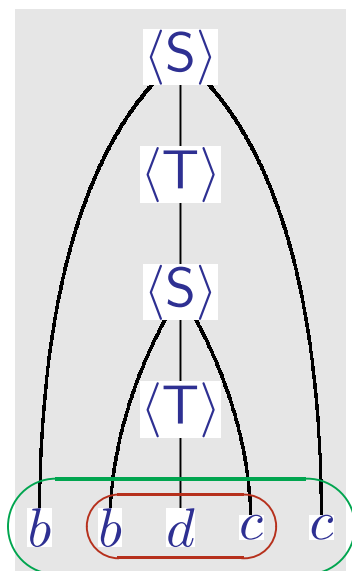
$\langle S \rangle$	$::=$	a	bdc	$b\langle S \rangle c$
---------------------	-------	-----	-------	------------------------

Przekształcenia gramatyk

Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

$\langle S \rangle$	$::=$	a	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$\langle S \rangle$



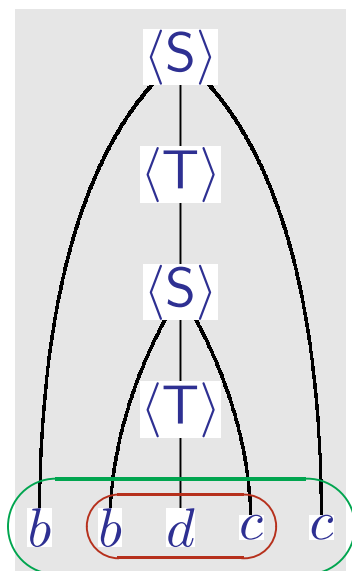
$\langle S \rangle$	$::=$	a	bdc	$b\langle S \rangle c$
---------------------	-------	-----	-------	------------------------

Przekształcenia gramatyk

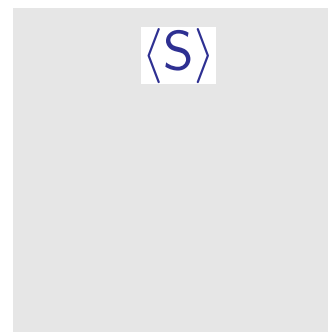
Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

$\langle S \rangle$	$::=$	a	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$\langle S \rangle$



$\langle S \rangle$	$::=$	a	bdc	$b\langle S \rangle c$
---------------------	-------	-----	-------	------------------------

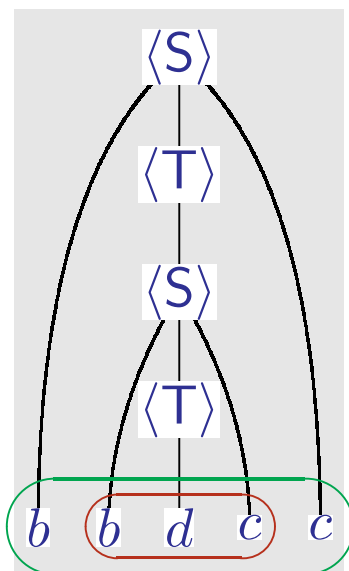


Przekształcenia gramatyk

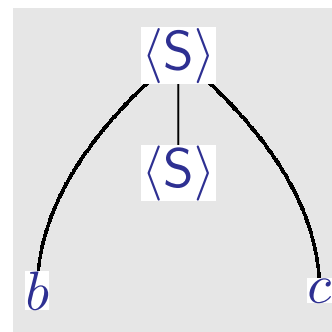
Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

$\langle S \rangle$	$::=$	a	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$\langle S \rangle$



$\langle S \rangle$	$::=$	a	bdc	$b\langle S \rangle c$
---------------------	-------	-----	-------	------------------------

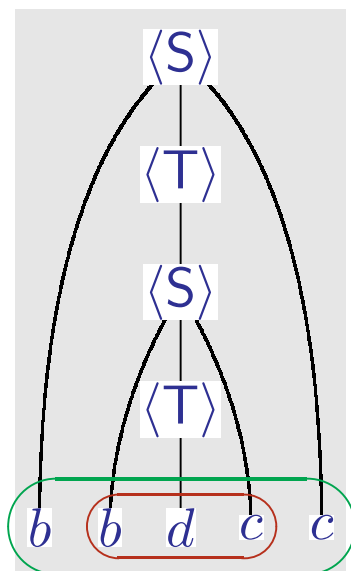


Przekształcenia gramatyk

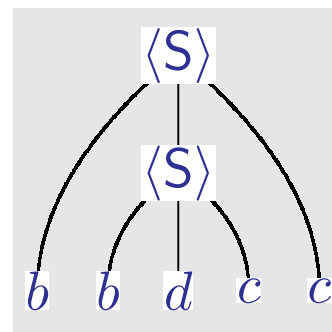
Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

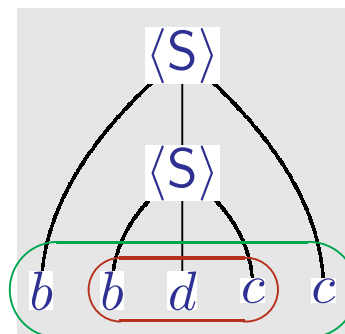
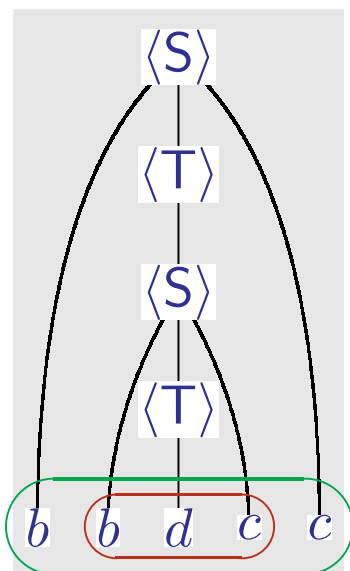
$\langle S \rangle$	$::=$	a	$ $	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$ $	$\langle S \rangle$



$\langle S \rangle$	$::=$	a	$ $	bdc	$ $	$b\langle S \rangle c$
---------------------	-------	-----	-----	-------	-----	------------------------



Przykład: słowo *bbdcc*

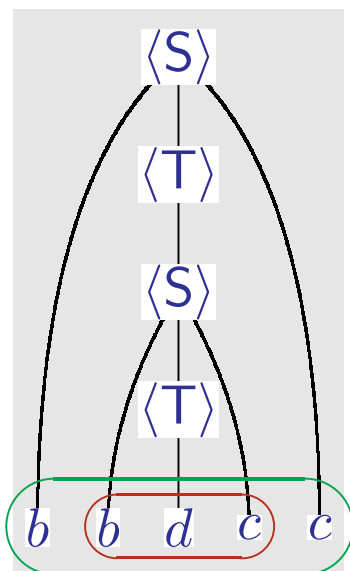
$$\langle S \rangle ::= a \mid bdc \mid b\langle S \rangle c$$


Przekształcenia gramatyk

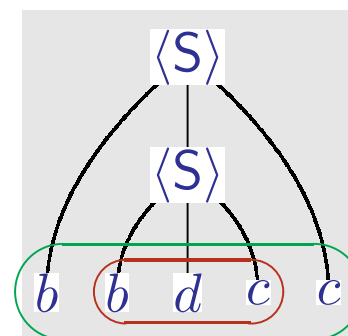
Wpływ przekształceń na kształt drzewa wywodu

Przykład: słowo *bbdcc*

$\langle S \rangle$	$::=$	a	$b\langle T \rangle c$
$\langle T \rangle$	$::=$	d	$\langle S \rangle$



$\langle S \rangle$	$::=$	a	bdc	$b\langle S \rangle c$
---------------------	-------	-----	-------	------------------------



Wniosek:

Drzewa są różne, ale ustalają takie same pierwszeństwa.

Dbajmy o drzewa!

Dbajmy o drzewa!

$$\langle X \rangle ::= L \mid \langle X \rangle M$$

$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

Dbajmy o drzewa!

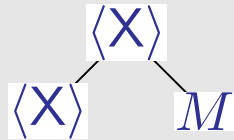
$$\langle X \rangle ::= L \mid \langle X \rangle M$$

$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$
 $\langle X \rangle$
 $\langle X \rangle$

Dbajmy o drzewa!

$$\langle X \rangle ::= L \mid \langle X \rangle M$$

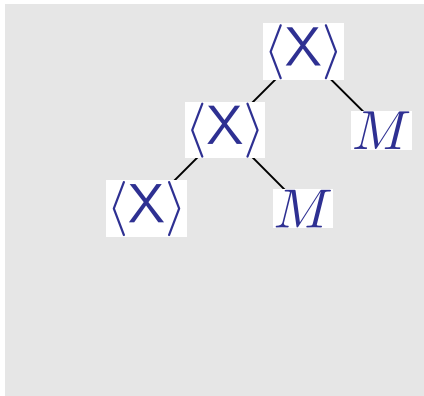
$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

$$\langle X \rangle \Rightarrow \langle X \rangle M$$


Dbajmy o drzewa!

$$\langle X \rangle ::= L \mid \langle X \rangle M$$

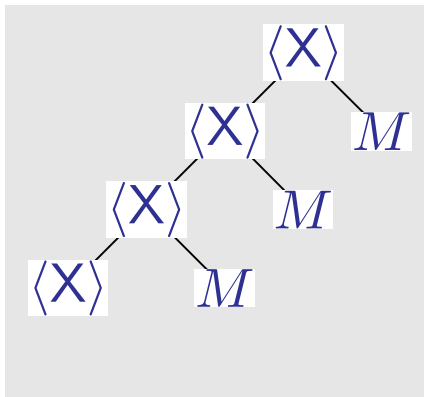
$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

$$\langle X \rangle \Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle M M$$


Dbajmy o drzewa!

$$\langle X \rangle ::= L \mid \langle X \rangle M$$

$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

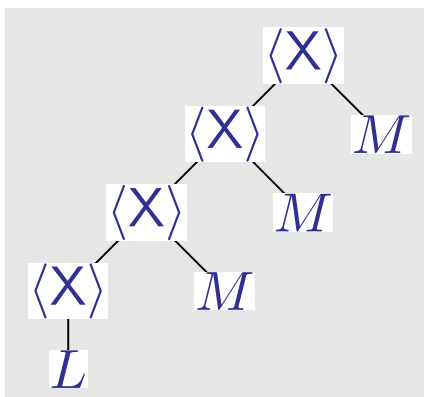
$$\langle X \rangle \Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle M M \Rightarrow \langle X \rangle M M M$$


Dbajmy o drzewa!

$$\langle X \rangle ::= L \mid \langle X \rangle M$$

$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

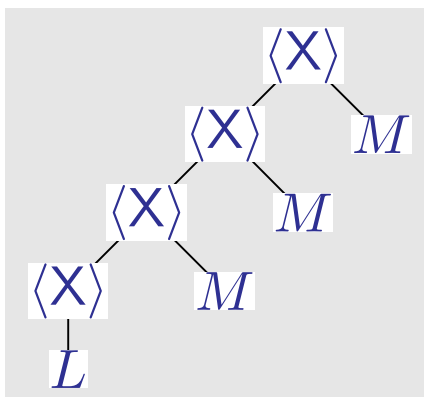
$$\begin{aligned} \langle X \rangle &\Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle MM \Rightarrow \langle X \rangle MMM \\ &\Rightarrow LMMM \end{aligned}$$



Dbajmy o drzewa!

$$\langle X \rangle ::= L \mid \langle X \rangle M$$

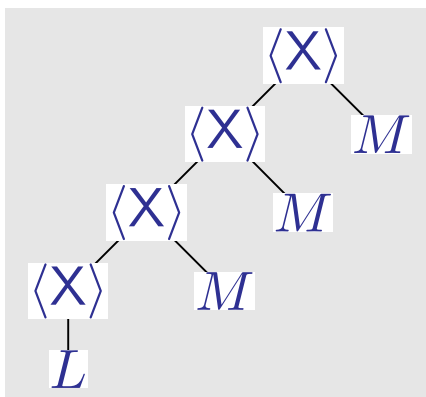
$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

$$\begin{aligned} \langle X \rangle &\Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle MM \Rightarrow \langle X \rangle MMM \\ &\Rightarrow LMMM \end{aligned}$$
 $\langle X \rangle$

 $\langle X \rangle$

Dbajmy o drzewa!

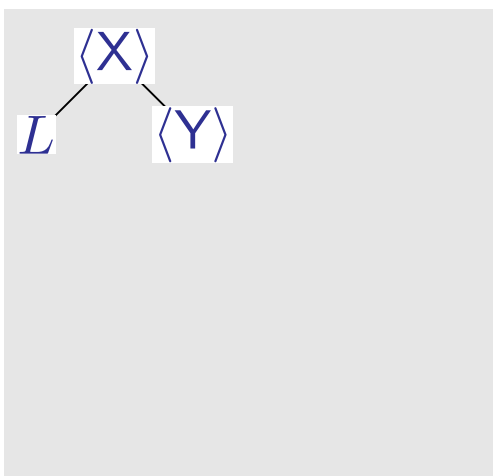
$$\langle X \rangle ::= L \mid \langle X \rangle M$$

$$\langle X \rangle \Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle MM \Rightarrow \langle X \rangle MMM \\ \Rightarrow LMMM$$



$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

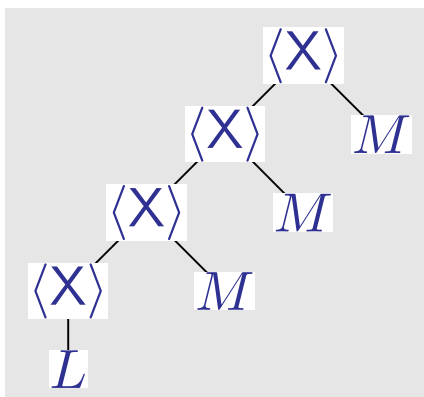
$$\langle X \rangle \Rightarrow L \langle Y \rangle$$



Dbajmy o drzewa!

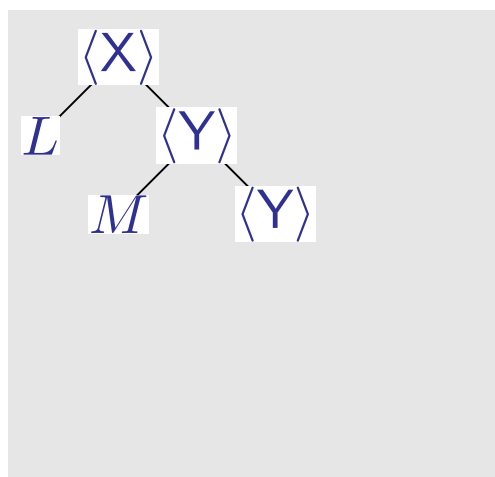
$$\langle X \rangle ::= L \mid \langle X \rangle M$$

$$\langle X \rangle \Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle MM \Rightarrow \langle X \rangle MMM \\ \Rightarrow LMMM$$



$$\left\{ \begin{array}{l} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{array} \right.$$

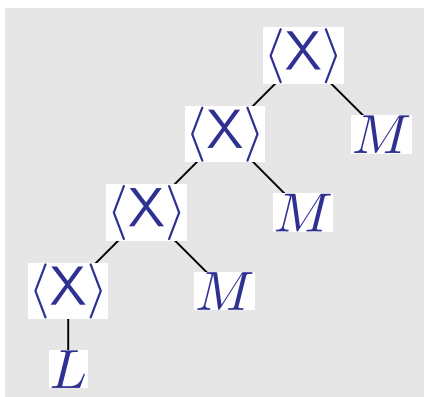
$$\langle X \rangle \Rightarrow L \langle Y \rangle \Rightarrow LM \langle Y \rangle$$



Dbajmy o drzewa!

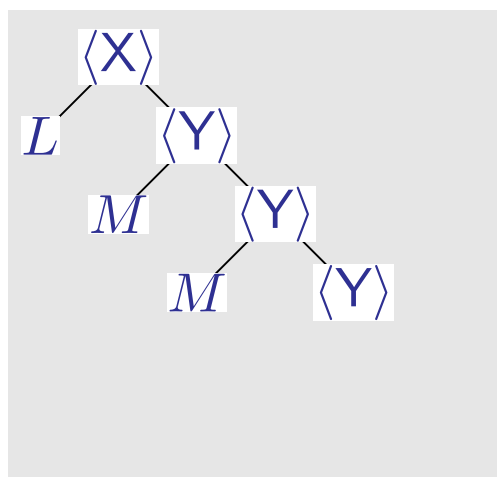
$$\langle X \rangle ::= L \mid \langle X \rangle M$$

$$\langle X \rangle \Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle MM \Rightarrow \langle X \rangle MMM \Rightarrow LMMM$$



$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

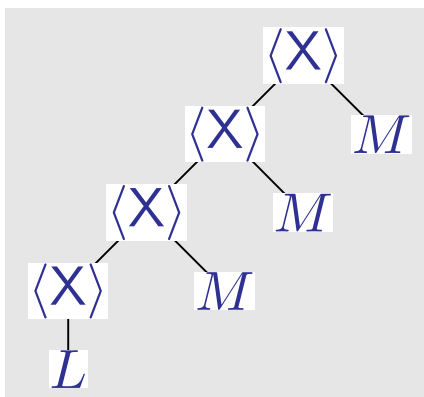
$$\langle X \rangle \Rightarrow L \langle Y \rangle \Rightarrow LM \langle Y \rangle \Rightarrow LMM \langle Y \rangle$$



Dbajmy o drzewa!

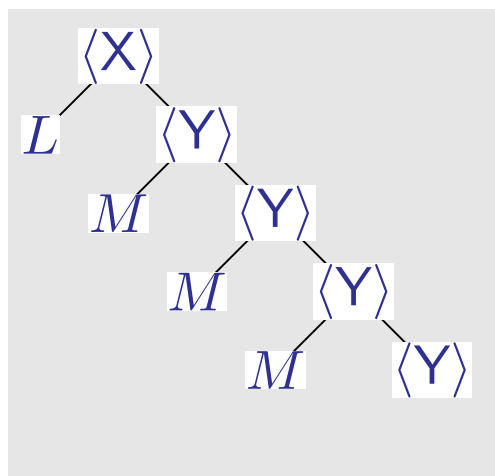
$$\langle X \rangle ::= L \mid \langle X \rangle M$$

$$\begin{aligned} \langle X \rangle &\Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle MM \Rightarrow \langle X \rangle MMM \\ &\Rightarrow LMMM \end{aligned}$$



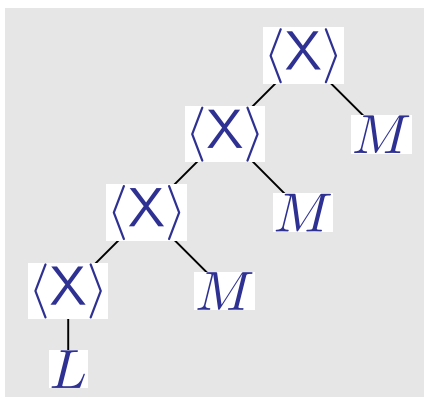
$$\begin{cases} \langle X \rangle ::= L \mid \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

$$\begin{aligned} \langle X \rangle &\Rightarrow L \langle Y \rangle \Rightarrow LM \langle Y \rangle \Rightarrow LMM \langle Y \rangle \\ &\Rightarrow LMMM \langle Y \rangle \end{aligned}$$

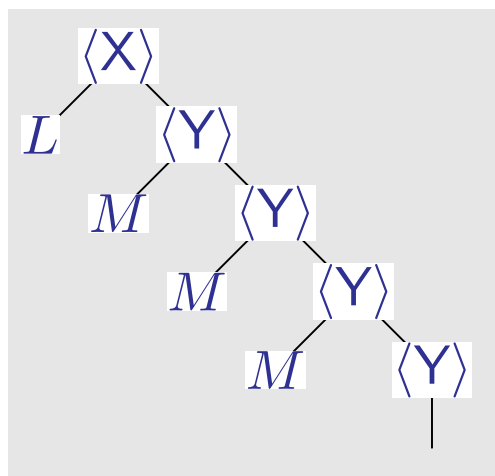


Dbajmy o drzewa!

$$\langle X \rangle ::= L \mid \langle X \rangle M$$

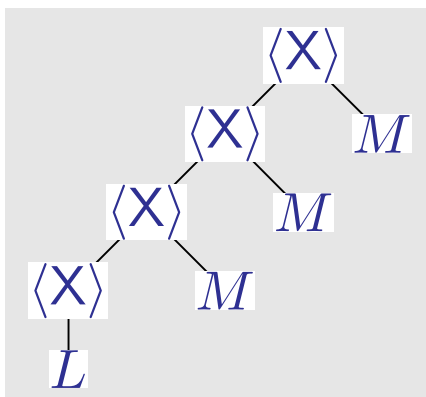
$$\langle X \rangle \Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle MM \Rightarrow \langle X \rangle MMM \\ \Rightarrow LMMM$$


$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

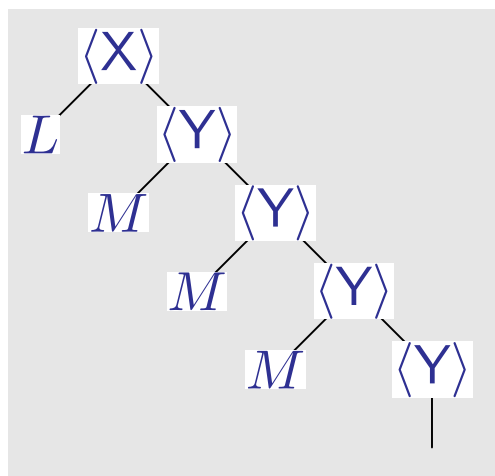
$$\langle X \rangle \Rightarrow L \langle Y \rangle \Rightarrow LM \langle Y \rangle \Rightarrow LMM \langle Y \rangle \\ \Rightarrow LMMM \langle Y \rangle \Rightarrow LMMM$$


Dbajmy o drzewa!

$$\langle X \rangle ::= L \mid \langle X \rangle M$$

$$\langle X \rangle \Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle MM \Rightarrow \langle X \rangle MMM \Rightarrow LMMM$$


$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

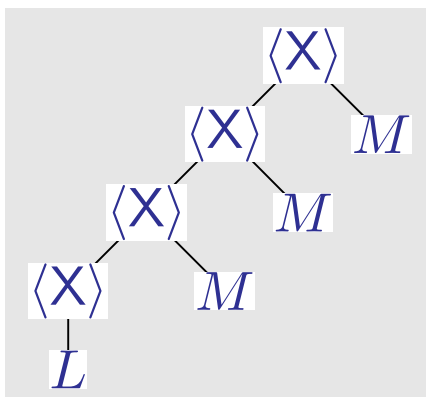
$$\langle X \rangle \Rightarrow L \langle Y \rangle \Rightarrow LM \langle Y \rangle \Rightarrow LMM \langle Y \rangle \Rightarrow LMMM \langle Y \rangle \Rightarrow LMMM$$


- Gramatyki **niebieska** i **czerwona** generują **te same języki**, ale drzewa wywodu ich słów dają różne pierwszeństwa.

Dbajmy o drzewa!

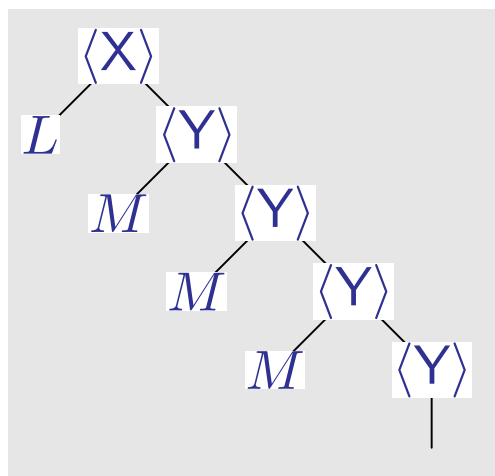
$$\langle X \rangle ::= L \mid \langle X \rangle M$$

$$\langle X \rangle \Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle MM \Rightarrow \langle X \rangle MMM \Rightarrow LMMM$$



$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

$$\langle X \rangle \Rightarrow L \langle Y \rangle \Rightarrow LM \langle Y \rangle \Rightarrow LMM \langle Y \rangle \Rightarrow LMMM \langle Y \rangle \Rightarrow LMMM$$

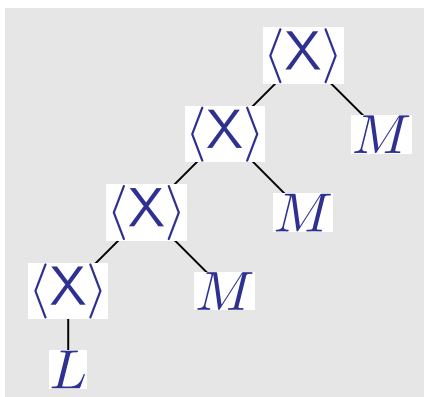


- Gramatyki **niebieska** i **czerwona** generują **te same języki**, ale drzewa wywodu ich słów dają różne pierwszeństwa.
- Z punktu widzenia struktury logicznej słów, **nie można zastąpić** jednej przez drugą.

Dbajmy o drzewa!

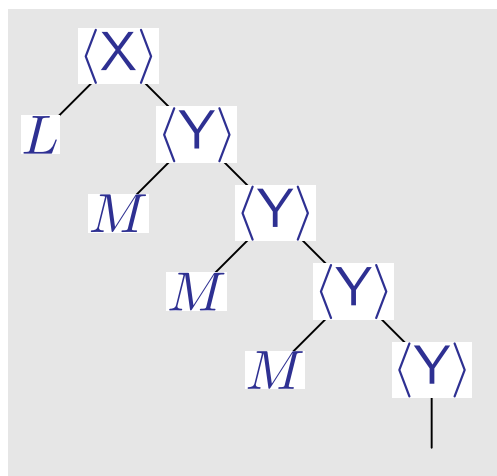
$$\langle X \rangle ::= L \mid \langle X \rangle M$$

$$\langle X \rangle \Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle MM \Rightarrow \langle X \rangle MMM \Rightarrow LMMM$$



$$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$$

$$\langle X \rangle \Rightarrow L \langle Y \rangle \Rightarrow LM \langle Y \rangle \Rightarrow LMM \langle Y \rangle \Rightarrow LMMM \langle Y \rangle \Rightarrow LMMM$$



- Gramatyki **niebieska** i **czerwona** generują **te same języki**, ale drzewa wywodu ich słów dają różne pierwszeństwa.
- Z punktu widzenia struktury logicznej słów, **nie można zastąpić** jednej przez drugą.
- Ostrzeżenie:** *ChatGPT* o tym nie wie.

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Niektóre rodzaje parserów nie radzą sobie z rekursją lewostronną (będzie o tym dalej).

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Niektóre rodzaje parserów nie radzą sobie z rekursją lewostronną (będzie o tym dalej). Rozwiązaniem jest przekształcenie gramatyki:

$$\langle X \rangle ::= L \mid \langle X \rangle M \longrightarrow \langle X \rangle ::= LM^*$$

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Niektóre rodzaje parserów nie radzą sobie z rekursją lewostronną (będzie o tym dalej). Rozwiązaniem jest przekształcenie gramatyki:

$$\langle X \rangle ::= L \mid \langle X \rangle M \longrightarrow \langle X \rangle ::= LM^*$$

Czy język LM^* spełnia niebieskie równanie?

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Niektóre rodzaje parserów nie radzą sobie z rekursją lewostronną (będzie o tym dalej). Rozwiązaniem jest przekształcenie gramatyki:

$$\langle X \rangle ::= L \mid \langle X \rangle M \longrightarrow \langle X \rangle ::= LM^*$$

Czy język LM^* spełnia niebieskie równanie?

$$X = LM^*$$

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Niektóre rodzaje parserów nie radzą sobie z rekursją lewostronną (będzie o tym dalej). Rozwiązaniem jest przekształcenie gramatyki:

$$\langle X \rangle ::= L \mid \langle X \rangle M \longrightarrow \langle X \rangle ::= LM^*$$

Czy język LM^* spełnia niebieskie równanie?

$$X = LM^* = L\{\Lambda \mid M \mid M^2 \mid M^3 \mid \dots\}$$

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Niektóre rodzaje parserów nie radzą sobie z rekursją lewostronną (będzie o tym dalej). Rozwiązaniem jest przekształcenie gramatyki:

$$\langle X \rangle ::= L \mid \langle X \rangle M \longrightarrow \langle X \rangle ::= LM^*$$

Czy język LM^* spełnia niebieskie równanie?

$$\begin{aligned} X &= LM^* = L \{ \Lambda \mid M \mid M^2 \mid M^3 \mid \dots \} \\ &= L \mid L \{ M \mid M^2 \mid M^3 \mid \dots \} \end{aligned}$$

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Niektóre rodzaje parserów nie radzą sobie z rekursją lewostronną (będzie o tym dalej). Rozwiązaniem jest przekształcenie gramatyki:

$$\langle X \rangle ::= L \mid \langle X \rangle M \longrightarrow \langle X \rangle ::= LM^*$$

Czy język LM^* spełnia niebieskie równanie?

$$\begin{aligned} X &= LM^* = L \{ \Lambda \mid M \mid M^2 \mid M^3 \mid \dots \} \\ &= L \mid L \{ M \mid M^2 \mid M^3 \mid \dots \} = L \mid L \{ \Lambda \mid M \mid M^2 \mid \dots \} M \end{aligned}$$

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Niektóre rodzaje parserów nie radzą sobie z rekursją lewostronną (będzie o tym dalej). Rozwiązaniem jest przekształcenie gramatyki:

$$\langle X \rangle ::= L \mid \langle X \rangle M \longrightarrow \langle X \rangle ::= LM^*$$

Czy język LM^* spełnia niebieskie równanie?

$$\begin{aligned} X &= LM^* = L \{ \Lambda \mid M \mid M^2 \mid M^3 \mid \dots \} \\ &= L \mid L \{ M \mid M^2 \mid M^3 \mid \dots \} = L \mid L \{ \Lambda \mid M \mid M^2 \mid \dots \} M \\ &= L \mid LM^* M \end{aligned}$$

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Niektóre rodzaje parserów nie radzą sobie z rekursją lewostronną (będzie o tym dalej). Rozwiązaniem jest przekształcenie gramatyki:

$$\langle X \rangle ::= L \mid \langle X \rangle M \longrightarrow \langle X \rangle ::= LM^*$$

Czy język LM^* spełnia niebieskie równanie?

$$\begin{aligned} X &= LM^* = L \{ \Lambda \mid M \mid M^2 \mid M^3 \mid \dots \} \\ &= L \mid L \{ M \mid M^2 \mid M^3 \mid \dots \} = L \mid L \{ \Lambda \mid M \mid M^2 \mid \dots \} M \\ &= L \mid LM^* M = L \mid XM \end{aligned}$$

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Niektóre rodzaje parserów nie radzą sobie z rekursją lewostronną (będzie o tym dalej). Rozwiązaniem jest przekształcenie gramatyki:

$$\langle X \rangle ::= L \mid \langle X \rangle M \longrightarrow \langle X \rangle ::= LM^*$$

Czy język LM^* spełnia niebieskie równanie?

$$\begin{aligned} X &= LM^* = L \{ \Lambda \mid M \mid M^2 \mid M^3 \mid \dots \} \\ &= L \mid L \{ M \mid M^2 \mid M^3 \mid \dots \} = L \mid L \{ \Lambda \mid M \mid M^2 \mid \dots \} M \\ &= L \mid LM^*M = L \mid XM \end{aligned}$$

Konstruując parser należy więc

- zastąpić w gramatyce lewostronną rekursję „gwiazdkowaniem”

Rekursja lewostronna

— kiedy jakaś produkcja z nieterminalu $\langle X \rangle$ zaczyna się od $\langle X \rangle$.

Niektóre rodzaje parserów nie radzą sobie z rekursją lewostronną (będzie o tym dalej). Rozwiązaniem jest przekształcenie gramatyki:

$$\langle X \rangle ::= L \mid \langle X \rangle M \longrightarrow \langle X \rangle ::= LM^*$$

Czy język LM^* spełnia niebieskie równanie?

$$\begin{aligned} X &= LM^* = L \{ \Lambda \mid M \mid M^2 \mid M^3 \mid \dots \} \\ &= L \mid L \{ M \mid M^2 \mid M^3 \mid \dots \} = L \mid L \{ \Lambda \mid M \mid M^2 \mid \dots \} M \\ &= L \mid LM^*M = L \mid XM \end{aligned}$$

Konstruując parser należy więc

- zastąpić w gramatyce lewostronną rekursję „gwiazdkowaniem”,
- do fragmentów „gwiazdkowanych” budować drzewo **z łączeniem do lewej**.