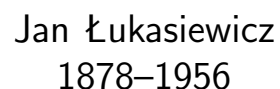


JĘZYKI FORMALNE I METODY KOMPILACJI

Inst. Informatyki Stosowanej, ANS
Elbląg, 2025/2026

Notacja beznawiasowa Łukasiewicza

$$= * 4 \boxed{+ 2 3} 20$$
$$4 \boxed{23} + * 20 =$$


Jak to czytać?

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

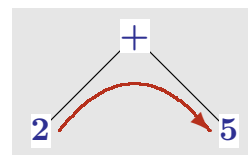
unary — jednoargumentowy
binary — dwuargumentowy
ternary — trójargumentowy
...

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

Wyrażenia — drzewa (nie całkiem drz. wyvodu).

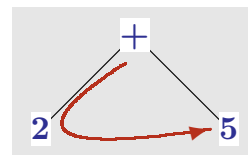
Różne **notacje** — sposoby chodzenia po drzewach.

Notacja beznawiasowa a chodzenie po drzewach



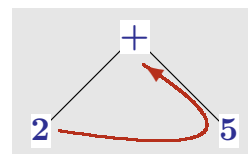
zwykła notacja (*infix*):

$2 + 5$



Notacja Łukasiewicza (*prefix*):

$+ 2 5$

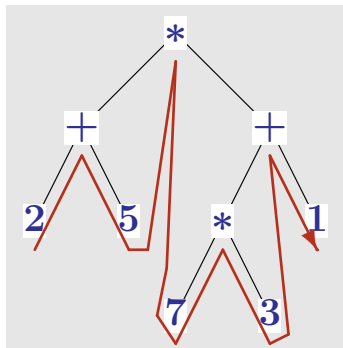


Odwrotna notacja
polska (*postfix*):

$2 5 +$

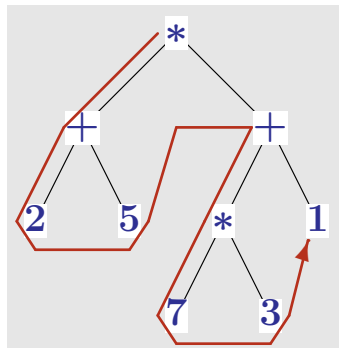
Notacja beznawiasowa a chodzenie po drzewach

PRZYKŁAD: (bardziej złożony)



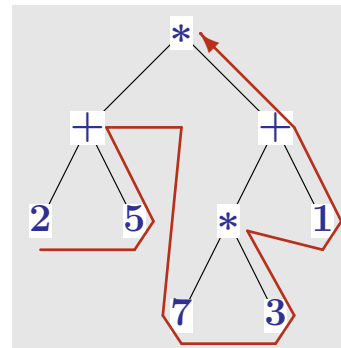
infix — notacja
zwykła, **nawiasowa**:

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



prefix — notacja
Łukasiewicza:

$* + 2 5 + * 7 3 1$

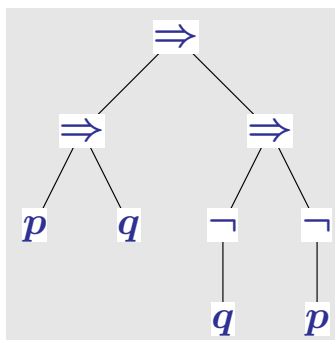


postfix — notacja
odwrotna polska:

$2 5 + 7 3 * 1 + *$

Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (wyrażenia



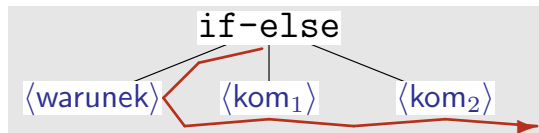
Zwykła notacja (*infix*): $(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$

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

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

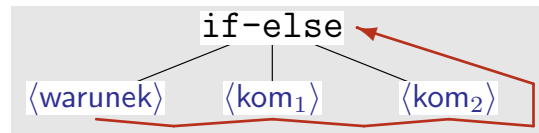
Notacja nawiasowa i beznawiasowa

PRZYKŁAD: (operatory inne niż binarne)



prefix:

if-else <warunek> <kom₁> <kom₂>



postfix:

<warunek> <kom₁> <kom₂> if-else

infix:

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

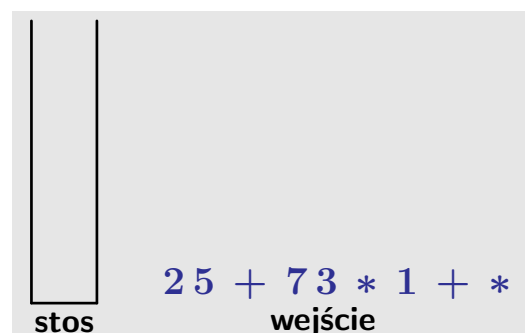
Obliczanie wyrażeń w odwrotnej notacji polskiej

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:

1. Suma:

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

2. Złączenie:

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

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)

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

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

PRZYKŁAD: (złączenie)

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

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

Działania na językach

Własności działań:

- 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:

- **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

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:

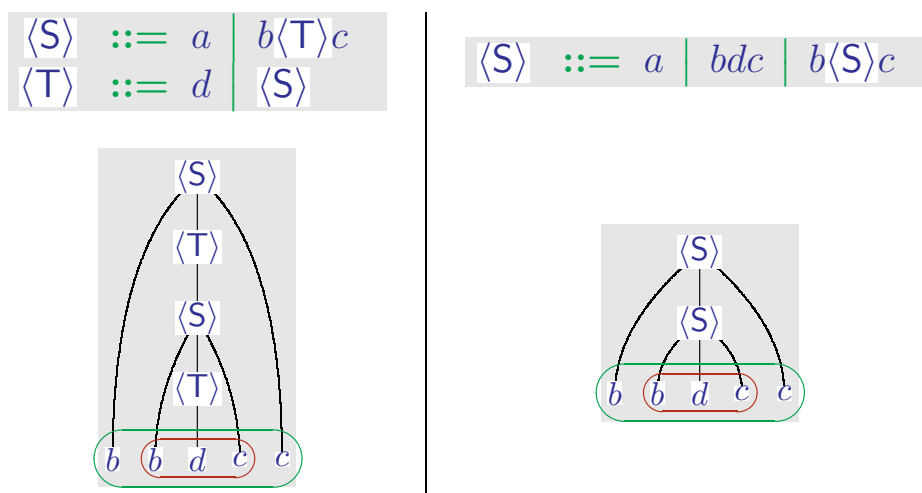
$$\begin{array}{|l|l|l|} \hline \langle S \rangle & ::= & a \mid b\langle T \rangle c \\ \hline \langle T \rangle & ::= & d \mid \langle S \rangle \\ \hline \end{array} \longrightarrow \begin{array}{|l|l|l|} \hline S & = & a \mid bTc \\ \hline T & = & d \mid S \\ \hline \end{array} \xrightarrow{\text{podstawienie}}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

$$\xrightarrow{\text{rozdzielność}} \begin{array}{|l|l|l|} \hline S & = & a \mid bdc \mid bSc \\ \hline \end{array} \longrightarrow \begin{array}{|l|l|l|} \hline \langle S \rangle & ::= & a \mid bdc \mid b\langle S \rangle c \\ \hline \end{array}$$

Przekształcenia gramatyk

Wpływ przekształceń na kształt drzewa wyvodu

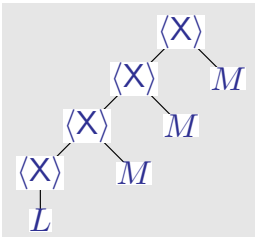
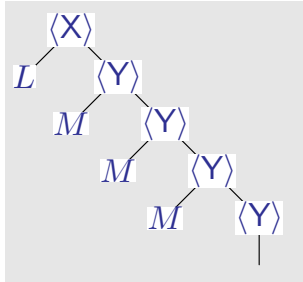
Przykład: słowo *bbdccc*



Wniosek:

Drzewa są różne, ale ustalają takie same pierwszeństwa.

Dbajmy o drzewa!

$\langle X \rangle ::= L \mid \langle X \rangle M$	$\begin{cases} \langle X \rangle ::= L \langle Y \rangle \\ \langle Y \rangle ::= \lambda \mid M \langle Y \rangle \end{cases}$
$\langle X \rangle \Rightarrow \langle X \rangle M \Rightarrow \langle X \rangle MM \Rightarrow \langle X \rangle MMM \Rightarrow LMMM$	$\langle X \rangle \Rightarrow L \langle Y \rangle \Rightarrow LM \langle Y \rangle \Rightarrow LMM \langle Y \rangle \Rightarrow LMMM \langle Y \rangle \Rightarrow LMMM$
	

- Gramatyki **niebieska** i **czerwona** generują **te same języki**, ale drzewa wyvodu 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 $\langle X \rangle$.

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 łąčeniem do lewej**.