Wstęp do informatyki

Wykład 12

Gramatyki, odwrotna notacja polska

Temat wykładu

- Gramatyki bezkontekstowe jako narzędzie do opisu <u>składni</u> języków programowania
- Odwrotna notacja polska (ONP):
  - Konwersja „standardowych” wyrażeń do postaci ONP.
  - Wyznaczanie wartości wyrażenia w postaci ONP.

Gramatyki bezkontekstowe: motywacja

Definicja języka programowania

Składnia – reguły definiujące formalnie poprawne programy i konstrukcje językowe (poprawne składniowo/syntaktycznie)

Semantyka – znaczenie konstrukcji językowych:

- semantyka operacyjna
- semantyka denotacyjna

Składnia języka

Przykład. Składnia wyrażeń arytmetycznych:

- argumenty: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9;
- operatory binarne (dwuargumentowe): $+$, $*$, $-$
- nawiasy: (,)

Poprawne:

$5 + 2 * 4 + 0 * 7$

$((5 + 2) * 4) + 0 * 7$

Niepoprawne:

$5 + 2 * 4 + 0\ *$

$((5 + 2 * 4) + (0 * 7$

UWAGA: przyjmujemy dla uproszczenia, że argumenty to tylko **cyfry** (np. 723 nie może być argumentem)

Składnia wyrażeń arytmetycznych

Def. Poprawne wyrażenia

- Cyfra jest poprawnym wyrażeniem
- Jeśli W1 i W2 to poprawne wyrażenia, to poprawnymi wyrażeniami są również:
  - W1 + W2
  - W1 $*$ W2
  - (W1)

Składnia wyrażeń arytmetycznych

Definicja za pomocą „reguł przepisywania”:

S – symbol startowy

$S \to 0, \ldots, S \to 9$

$S \to S + S$

$S \to S * S$

$S \to (S)$

Składnia wyrażeń arytmetycznych

Stosowanie reguł przepisywania

(wyprowadzenie):

$S \Rightarrow S * S \Rightarrow S + S * S \Rightarrow$

$(S) + S * S \Rightarrow$

$(S * S) + S * S \Rightarrow$

$(5 * S) + S * S \Rightarrow$

$(5 * 4) + S * S \Rightarrow$

$(5 * 4) + 1 * S \Rightarrow$

$(5 * 4) + 1 * 2$

REGUŁY:

$S \rightarrow 0, \ldots, S \rightarrow 9$

$S \rightarrow S + S$

$S \rightarrow S * S$

$S \rightarrow (S)$

Składnia wyrażeń arytmetycznych

Stosowanie **reguł przepisywania**

(wyprowadzenie):

- startujemy od symbolu startowego S
- kontynuujemy przepisywanie aż do uzyskania napisu składającego się wyłącznie z cyfr, operatorów i nawiasów

Składnia wyrażeń arytmetycznych

Napisy, których nie można uzyskać stosując reguły przepisywania:

REGUŁY:

$S \to 0, \dots, S \to 9$

$S \to S + S$

$S \to S * S$

$S \to (S)$

- $(5 * 4))$

  Narusza zasadę: parzysta liczba nawiasów (wstawiane parami)

- $)(5 + 4)($

  Narusza zasadę: każdy nawias zamykający jest wstawiany wraz z <u>poprzedzającym</u> go nawiasem otwierającym „do pary”

Napisy, języki i gramatyki formalnie

Napisy i języki

Pojęcia:

- A – skończony zbiór (alfabet)
- Słowo/napis – skończony ciąg elementów z A (może być pusty)
- A* - zbiór wszystkich słów
- L - język nad alfabetem A to podzbiór A*

Przykład:

- A = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, ∗, (,) }
- A* - zbiór napisów ze znakami z A, np. 0∗1+,   ()
- L – napisy odpowiadające poprawnym wyrażeniom

Napisy i języki

Pojęcia:

- A – skończony zbiór (alfabet)
- słowo – skończony ciąg elementów z A (może być pusty)
- A* - zbiór wszystkich słów
- L - język nad alfabetem A to podzbiór A*

Przykład:

- A – zbiór liter alfabetu polskiego i znaków przestankowych
- A* - zbiór napisów ze znakami z A, np. aą;.że
- L – poprawne <u>składniowo</u> teksty w języku polskim

Napisy i języki

Oznaczenia:

- ε - słowo puste
- a^* - zbiór napisów złożonych z dowolnej liczby powtórzeń litery a czyli $\{\varepsilon, a, aa, aaa, aaaa, \ldots\}$
- a^+ - zbiór napisów złożonych z dodatniej liczby powtórzeń litery a czyli $\{a, aa, aaa, aaaa, \ldots\}$
- a^k – napis złożony z k powtórzeń litery a

Gramatyka bezkontekstowa

Gramatyka bezkontekstowa $G(N,T,P,S)$ jest zdefiniowana poprzez:

- N – zbiór **nieterminali** (symbole pomocnicze)
- T – zbiór **terminali** (symbole podstawowe)
- P – zbiór produkcji – podzbiór $N \times (N \cup T)^*$
- S – symbol startowy, element N

Gramatyka bezkontekstowa

Konwencje:

- Duże litery alfabetu – **nieterminale** (elementy zbioru N)
- Małe litery i inne znaki – **terminale** (zbiór T)

Gramatyka bezkontekstowa

Gramatyka bezkontekstowa $G(N,T,P,S)$ jest zdefiniowana poprzez:

- N – zbiór nieterminali (symbole pomocnicze)
- T – zbiór terminali (symbole podstawowe)
- P – zbiór produkcji – podzbiór $N \times (N \cup T)^*$
- S – symbol startowy, element N

Przykład:

- $N =\{ S \}$
- $T =\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, *, (,) \}$
- $P = \{ S \to 0, \ldots, S \to 9, S \to S + S, S \to S * S, S \to (S) \}$
- symbol startowy: S

Gramatyka bezkontekstowa

Wyprowadzenie (formalnie) w gramatyce

$G(N,T,P,S)$ to ciąg napisów

$$x_1 \Rightarrow x_2 \Rightarrow \dots \Rightarrow x_n$$

taki, że

- $x_1 = S$
- $x_n \in T^*$
- x_{i+1} powstaje przez zastosowanie produkcji w x_i (czyli zastąpienie lewej strony jednej produkcji jej prawą)

Gramatyka - wyprowadzenie

Gramatyka:

- N ={ S }
- T ={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, $*$, (,) }
- P = { $S \to 0, \ldots, S \to 9, S \to S + S,$ $S \to S * S, S \to (S)$ }
- symbol startowy: S

Wyprowadzenie:

$S \Rightarrow S * S \Rightarrow$

$\underline{S + S} * S \Rightarrow$

$\underline{(S)} + S * S \Rightarrow$

$(\underline{S * S}) + S * S \Rightarrow$

$(\underline{5} * S) + S * S \Rightarrow$

$(5 * \underline{4}) + S * S \Rightarrow$

$(5 * 4) + \underline{1} * S \Rightarrow$

$(5 * 4) + 1 * \underline{2}$

Język a gramatyka bezkontekstowa

Gramatyka bezkontekstowa $G(N,T,P,S)$ definiuje język $L(G)$ złożony ze wszystkich słów nad alfabetem T (czyli $w \in T^*$), które można uzyskać przy pomocy **wyprowadzeń**.

Przykład:

- $N = \{ S \}$
- $T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, *, (,) \}$
- $P = \{ S \to 0, \ldots, S \to 9, S \to S + S,$
  $S \to S * S, S \to (S) \}$
- symbol startowy: S
- $L(G)$ – zbiór poprawnych wyrażeń

Gramatyka bezkontekstowa - przykłady

Przykład. G(N,T,P,S)

- N = { S }
- T = { a, b }

P = {

- $S \to aSb$
- $S \to \varepsilon$

}

Przykłady napisów

aaaaabb $\notin$ L(G)

ab $\in$ L(G)

ba $\notin$ L(G)

aabb $\in$ L(G)

L(G) – zbiór słów postaci $a^k b^k$, gdzie $k \geq 0$,

formalnie

$L(G)=\{w : w=a^i b^i,\ i \in \{0,1,2,\ldots\}\}$

Gramatyka bezkontekstowa - przykłady

Przykład. Liczba ze znakiem

G(N,T,P,S)

- N = { S, Z, C, X}
- T = { -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

P = {

- $S \rightarrow Z\ C\ X$
- $Z \rightarrow \varepsilon$
- $Z \rightarrow -$
- $C \rightarrow 0$, $C \rightarrow 1$, …, $C \rightarrow 9$
- $X \rightarrow \varepsilon$
- $X \rightarrow C\ X$

}

L(G) – liczby całkowite

Gramatyka bezkontekstowa - przykłady

Przykład. G(N,T,P,S)

- N = { S, A, B, C }
- T = { a, b }

P = {

- $S \rightarrow A\ b\ B$
- $A \rightarrow aA$
- $A \rightarrow \varepsilon$ (słowo puste)
- $B \rightarrow bB$
- $B \rightarrow \varepsilon$ (słowo puste)

}

L(G) – zbiór słów zaczynających się ciągiem liter a (być może pustym), za nim ciąg liter b (niepusty)

$L(G) = a^*b^+$

Przykłady słów

aaaaabb $\in$ L(G)

ab $\in$ L(G)

b $\in$ L(G)

ba $\notin$ L(G)

aa $\notin$ L(G)

Dlaczego nazwa „bezkontekstowa”?

Bezkontekstowość:

- Użycie produkcji w wyprowadzeniu niezależne od **kontekstu** (symboli w otoczeniu zastępowanego nieterminala)

Inne typy gramatyk

- kontekstowe
- regularne
- ściśle kontekstowe, …

Po co gramatyki?

1. **Precyzyjny** a zarazem intuicyjny opis, np. dla składni języków programowania (choć nie wszystkie elementy języka da się opisać!!!);
2. Istnieją efektywne narzędzia (algorytmy i programy) weryfikujące przynależność do języka definiowanego przez gramatykę – p. przedmioty JFiZO, AiSD, metody programowania, metody translacji
3. Twórca języka programowania i translatora:
   – definiuje jego składnię z użyciem gramatyk;
   – korzysta z narzędzi programistycznych (punkt 2) do sprawdzania czy napisany program jest poprawny składniowo.

Gramatyki:
drzewo wyprowadzenia

Drzewo wyprowadzenia – przykład

Produkcje:

$S \to 0, \ldots, S \to 9$

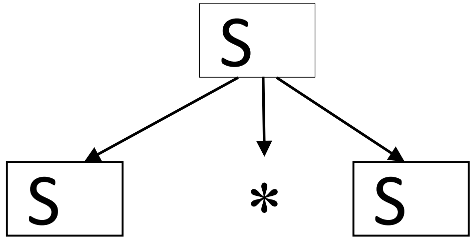
$S \to S + S$

$S \to S * S$

$S \to (S)$

Wyprowadzenie:

$S \Rightarrow S * S$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \to 0, \ldots, S \to 9$

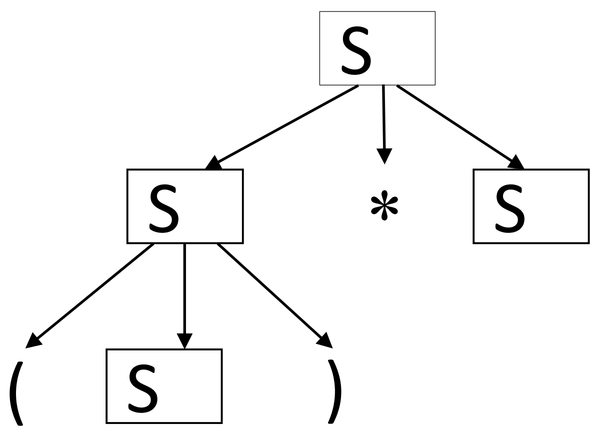
$S \to S + S$

$S \to S * S$

$S \to (S)$

Wyprowadzenie:

$S \Rightarrow S * S \Rightarrow (S) * S$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \to 0, \ldots, S \to 9$

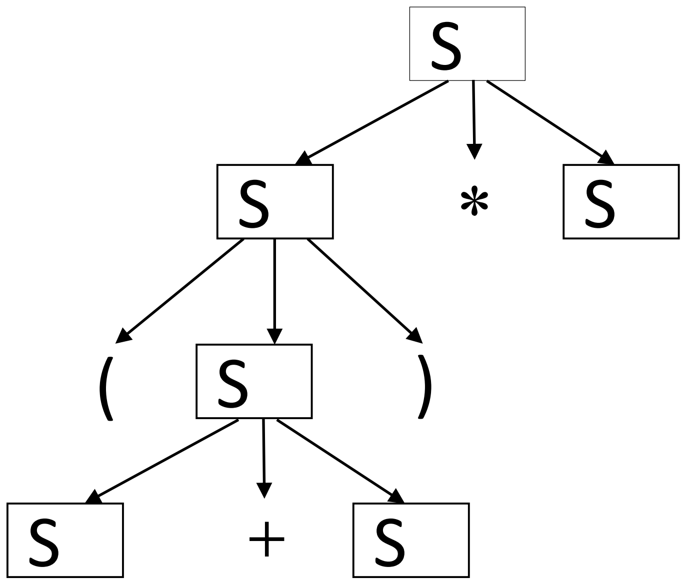
$S \to S + S$

$S \to S * S$

$S \to (S)$

Wyprowadzenie:

$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$
$(S + S) * S$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \to 0, \ldots, S \to 9$

$S \to S + S$

$S \to S * S$

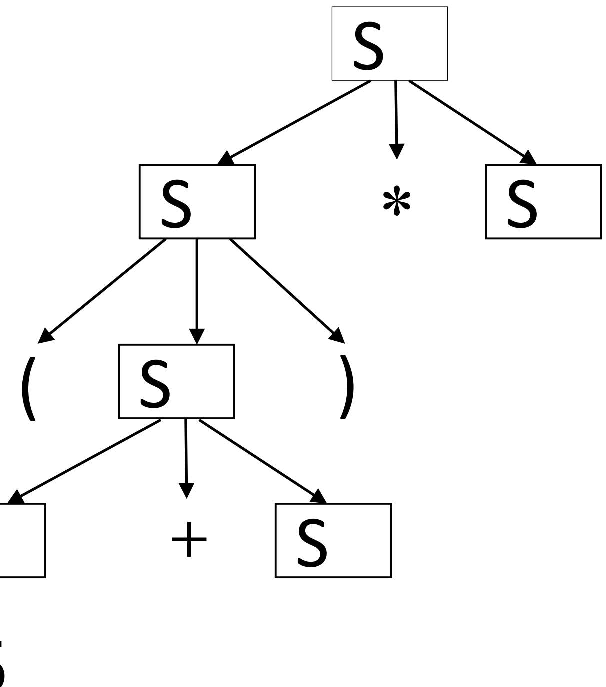
$S \to (S)$

Wyprowadzenie:

$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$

$(S + S) * S \Rightarrow$

$(5 + S) * S$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \to 0, \ldots, S \to 9$

$S \to S + S$

$S \to S * S$

$S \to (S)$

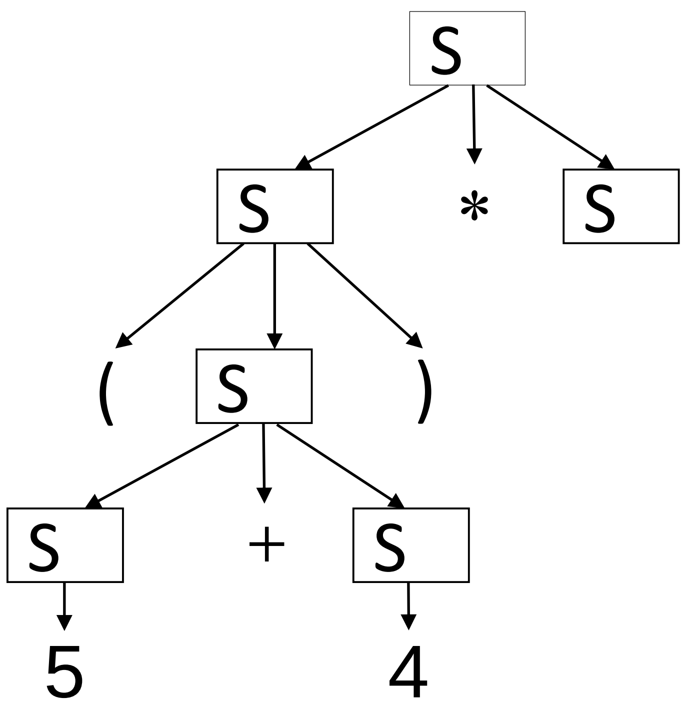
Wyprowadzenie:

$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$

$(S + S) * S \Rightarrow$

$(5 + S) * S \Rightarrow$

$(5 + 4) * S$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \to 0, \ldots, S \to 9$

$S \to S + S$

$S \to S * S$

$S \to (S)$

Wyprowadzenie:

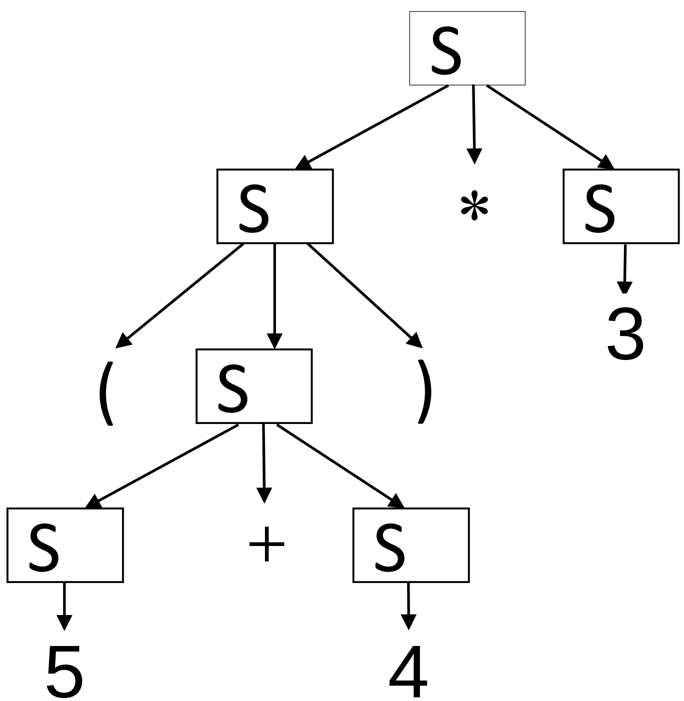
$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$

$(S + S) * S \Rightarrow$

$(5 + S) * S \Rightarrow$

$(5 + 4) * S \Rightarrow$

$(5 + 4) * 3$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \to 0, \ldots, S \to 9$

$S \to S + S$

$S \to S * S$

$S \to (S)$

Wyprowadzenie:

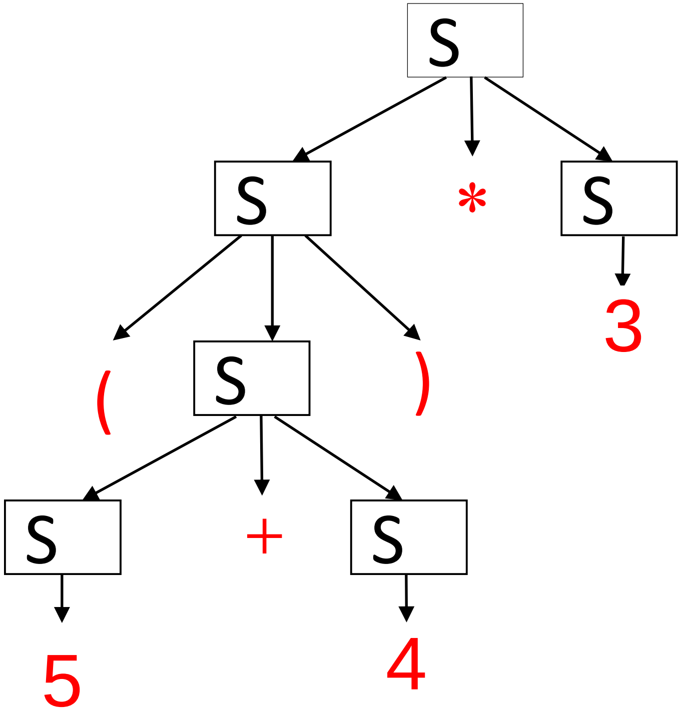
$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$

$(S + S) * S \Rightarrow$

$(5 + S) * S \Rightarrow$

$(5 + 4) * S \Rightarrow$

$(5 + 4) * 3$



Drzewo wyprowadzenia – formalnie

Drzewo (ukorzenione, z porządkiem „od lewej do prawej”):

Struktura złożona z powiązanych węzłów:

- Każdy węzeł może mieć dzieci uporządkowane od lewej do prawej.
- Każdy węzeł poza korzeniem ma dokładnie jednego rodzica
- Korzeń nie ma rodzica.
- **Dokładnie** jeden węzeł jest korzeniem.

Drzewo wyprowadzenia – formalnie

Drzewo wyprowadzenia w gramatyce

G(N,T,P,S):

- Każdy węzeł ma etykietę ze zbioru $N \cup T$
- Etykieta korzenia to S (symbol startowy)
- Etykiety liści ze zbioru T
- Etykiety węzłów wewnętrznych (nie-liści) z N
- Jeśli v ma etykietę X, to jego dzieci mają etykiety $Y_1 \ldots Y_n$ takie, że $X \rightarrow Y_1 \ldots Y_n$ jest produkcją z P.

Drzewo wyprowadzenia – przykład

Produkcje:

$S \to 0, \ldots, S \to 9$

$S \to S + S$

$S \to S * S$

$S \to (S)$

Wyprowadzenie:

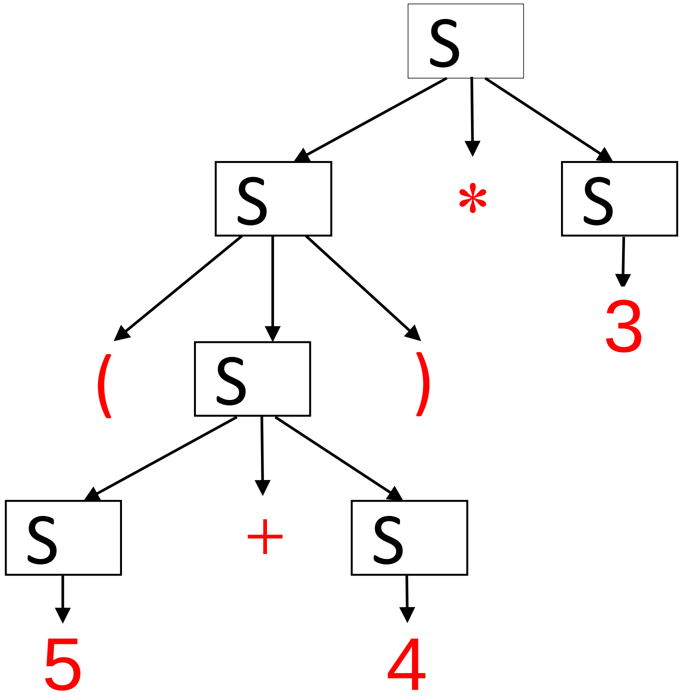
$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$

$(S + S) * S \Rightarrow$

$(5 + S) * S \Rightarrow$

$(5 + 4) * S \Rightarrow$

$(5 + 4) * 3$



Wyprowadzony napis:

- Etykiety liści drzewa w porządku od lewej do prawej
- Przykład: $(5+4) * 3$

Notacja BNF (Backus-Naur)

Notacja BNF:

- Extended Backus-Naur Form: bardziej intuicyjna od formalnego opisu
- Używana głównie do definiowania składni języków programowania

Pojęcie	**Gram. bezkontekst.**	**BNF**
Nieterminale	A, B, C, D, …	⟨napis⟩
Terminale	Małe litery, inne znaki	Dowolne znaki poza ⟨ , ⟩
Produkcje	$\rightarrow$	::=
	$A \rightarrow w_1$, $A \rightarrow w_2$	$\langle$napis$\rangle$ ::= $w_1 \mid w_2$

Gramatyka bezkontekstowa i BNF - przykład

Przykład. Liczba ze znakiem

G(N,T,P,S)

- N = { S, C, X}
- T = { -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

P = {

- $S \rightarrow$ - X
- $S \rightarrow$ X
- $C \rightarrow 0$, $C \rightarrow 1$, …, $C \rightarrow 9$
- $X \rightarrow C$
- $X \rightarrow C\ X$

}

Przykład. Liczba ze znakiem w BNF

G(N,T,P, ⟨liczba_ze_zn⟩)

- N = { ⟨liczba_ze_zn⟩, ⟨cyfra⟩, ⟨liczba⟩}
- T = { -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

P = {

- ⟨liczba_ze_zn⟩ ::= - ⟨liczba⟩ | ⟨liczba⟩
- ⟨cyfra⟩ ::= 0 | 1 | 2 | 3 |4 | 5 | 6 | 7 | 8 | 9
- ⟨liczba⟩ ::= ⟨cyfra⟩ | ⟨cyfra⟩ ⟨liczba⟩

}

Gramatyka bezkontekstowa i BNF - przykład

Przykład. Wyrażenie

G(N,T,P,S)

- N = { S, C}
- T = { -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

P = {

$C \to 0, \ldots, C \to 9$

$S \to C$

$S \to S + S$

$S \to S * S$

$S \to (S)$

}

Przykład. Wyrażenie w BNF

- N = { ⟨wyr⟩, ⟨arg⟩}
- T = { -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

P = {

- ⟨arg⟩ ::= 0 | 1 | 2 | 3 |4 | 5 | 6 | 7 | 8 | 9
- ⟨wyr⟩ ::= ⟨arg⟩ | (⟨wyr⟩) |
  ⟨wyr⟩ + ⟨wyr⟩ | ⟨wyr⟩ ∗ ⟨wyr⟩

}

Notacja EBNF (Extended Backus-Naur)

Notacja EBNF:

- Extended Backus-Naur Form: rozszerzenie BNF
- Używana głównie do definiowania składni języków programowania

Pojęcie	**BNF**	**EBNF**
Nieterminale	⟨napis⟩	napis
Terminale	Dowolne znaki poza ⟨ , ⟩	Dowolne znaki w cudzysłowie
Produkcje	::=	=
	$w_1 \mid w_2$	$w_1 \mid w_2$

Notacja EBNF (Extended Backus-Naur)

Notacja EBNF: dodatkowe konwencje

Opis	**EBNF**
0 lub 1 wystąpienie x	[x]
Dowolna nieujemna liczba wystąpień x	{ x }

BNF i EBNF - przykład

Przykład. Liczba ze znakiem w BNF

G(N,T,P, ⟨liczba_ze_zn⟩)

- N = { ⟨liczba_ze_zn⟩, ⟨cyfra⟩, ⟨liczba⟩}
- T = { -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

P = {

- ⟨liczba_ze_zn⟩ ::= - ⟨liczba⟩ | ⟨liczba⟩
- ⟨cyfra⟩ ::= 0 | 1 | 2 | 3 |4 | 5 | 6 | 7 | 8 | 9
- ⟨liczba⟩ ::= ⟨cyfra⟩ | ⟨cyfra⟩ ⟨liczba⟩

}

Przykład. Liczba ze znakiem w EBNF

G(N,T,P, ⟨liczba_ze_zn⟩)

- N = { liczba_ze_zn, cyfra, liczba}
- T = { -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

P = {

- liczba_ze_zn = [”-”] liczba
- cyfra = ”0” | ”1” | ”2” | ”3” | ”4” | ”5” | ”6” | ”7” | ”8” | ”9”
- liczba = **cyfra** { cyfra }

BNF i EBNF - przykład

Przykład. Wyrażenie w BNF

G(N,T,P, ⟨wyr⟩)

- N = { ⟨wyr⟩, ⟨arg⟩}
- T = { -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

P = {

- ⟨arg⟩ ::= 0 | 1 | 2 | 3 |4 | 5 | 6 | 7 | 8 | 9
- ⟨wyr⟩ ::= ⟨arg⟩ | (⟨wyr⟩) | ⟨wyr⟩ + ⟨wyr⟩ | ⟨wyr⟩ ∗ ⟨wyr⟩

}

Przykład. Wyrażenie w EBNF

G(N,T,P, ⟨wyr⟩)

- N = { wyr, arg}
- T = { -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

P = {

- arg = ”0” | ”1” | ”2” | ”3” | ”4” | ”5” | ”6” | ”7” | ”8” | ”9”
- wyr = arg | ”(” wyr ”)” | wyr ”+” wyr | wyr ”∗” wyr

}

Gramatyki – podsumowanie

Pojęcia:

- Alfabet, słowo, język
- Gramatyka bezkontekstowa, wyprowadzenie, drzewo wyprowadzenia
- Język definiowany przez gramatykę
- Notacja BNF
- Notacja EBNF

Korzyści z gramatyk:

- Precyzyjna definicja składni
- Narzędzia do sprawdzania przynależności do języka zdefiniowanego gramatyką

Odwrotna Notacja Polska (ONP) - motywacje

Wartościowanie wyrażeń

Dotychczas:

Składnia – czy wyrażenie jest poprawnie napisane

Teraz:

Semantyka (częściowa) – jak obliczyć wartość wyrażenia, czyli rozwiązać problem:

Wejście: poprawne wyrażenie w
Wyjście: wartość wyrażenia w

Wartościowanie wyrażeń

Wyrażenie:

$w = w_1 \dots w_n$

gdzie $w_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, (,), +, *\}$
oraz w należy do języka $L(G)$ dla $G(N,T,P,S)$
gdzie

$P=\{S\rightarrow 0 \dots S\rightarrow 9;$
$S \rightarrow (S); S\rightarrow S+S; S \rightarrow S*S\}$

Przypomnienie:
Dla uproszczenia zakładamy, że argumenty to liczby **jednocyfrowe**.

Wartościowanie wyrażeń

Uproszczenie:

- wszystkie operatory mają taki sam priorytet (kolejność można wymusić nawiasami)
- operacje wykonujemy „od lewej do prawej”

Wartość fi(w) wyrażenia $w=w_1\ldots w_n$:

- Jeśli wyrażenie $w \in \{0,1,\ldots,9\}$, wówczas jego wartość jest równa liczbie w
- Jeśli w=u op v dla wyrażeń u i v i operatora op, to fi(w) = fi(u) op fi(v)

  Uwaga: podział dla najkrótszego v, wynika z kolejności wykonywanie operacji „od lewej do prawej”.

Wartościowanie wyrażeń

Wartość fi(w) wyrażenia $w=w_1\ldots w_n$:

- Jeśli wyrażenie $w \in \{0,1,\ldots,9\}$, wówczas jego wartość jest równa liczbie w
- Jeśli w=u op v dla wyrażeń u i v i operatora op, to fi(w) = fi(u) op fi(v)

  Uwaga: podział dla najkrótszego v, nie ma priorytetów.

Przykład

fi(7)=7

fi(3+7)=fi(3)+fi(7)=10

$fi(3 + (7 * 2)) = fi(3) + fi(7 * 2) = 3 + 14 = 17$

$fi(3 + 7 * 2) = fi(3 + 7) * fi(2) = 10 * 2 = 20$

gdyż nie uwzględniamy priorytetów, i „od lewej do prawej”!!!

Wartościowanie wyrażeń

Problem z algorytmem wartościującym wg rekurencyjnej definicji:

jak rozbić wyrażenie w na 3 człony: u, op, v?

Przykład

Jak rozbić na u op v poniższe wyrażenie:

(5+7)-(2+3*4)*(2+3-5*(5-9))

Wartościowanie wyrażeń

Przykład

Jak rozbić na *u op v* poniższe wyrażenie:

(5+7) - (2+3*4) * (2+3-5*(5-9))

Gdy nie ma priorytetów – „jak najkrótsza prawa strona” („wykonuj od lewej”):

((5+7)-(2+3*4)) * ((2+3-5*(5-9)))

Z priorytetami:

((5+7))-((2+3*4) *(2+3-5*(5-9)))

Odwrotna notacja polska (ONP)

ONP – motywacja:

- zapis wyrażenia ułatwiający wartościowanie
- kolejność działań można ustalić bez nawiasów

Def. Postać ONP wyrażenia $w=w_1\ldots w_n$:

- Jeśli wyrażenie $w \in \{0,1,\ldots,9\}$, wówczas ONP(w) jest równa w
- Jeśli w=u op v dla wyrażeń u, v i operatora op, to
  ONP(w) = ONP(u) ONP(v) op

ONP – przykłady

ONP($3 + 4$) $= 3\ 4\ +$

ONP($(3+4)*5$) $= 3\ 4\ +\ 5\ *$

ONP($3 * (4+5)$) $= 3\ 4\ 5\ +\ *$

ONP($5 - (3*2)$) $= 5\ 3\ 2\ *\ -$

ONP($(5 - (3*2)) + 7$) $= 5\ 3\ 2\ *\ -\ 7\ +$

Przyjmując założenie o wykonywaniu operacji „od lewej do prawej:

ONP($5 - (3*2) * 7$) $=$

ONP($(5 - (3*2)) * 7$) $=$

$5\ 3\ 2\ *\ -\ 7\ *$

ONP – kluczowa własność

- W wyrażeniach ONP nie potrzeba nawiasów
- Wyrażenia ONP jednoznacznie określają kolejność wykonywania operacji:

  standardowo: $(a + b) * c$

  ONP: $a\ b + c\ *$

  standardowo: $a + (b * c)$

  ONP: $a\ b\ c * +$

ONP – problemy algorytmiczne

ONP – problemy algorytmiczne

Problem obliczania wartości wyrażenia:

Wejście: postać ONP wyrażenia w

Wyjście: wartość wyrażenia w

Problem zamiany na ONP:

Wejście: wyrażenie w postaci „standardowej”

Wyjście: postać ONP wyrażenia w

ONP – algorytm obliczania wartości wyrażenia

Problem obliczania wartości wyrażenia

Wartość fi(w) wyrażenia $w=w_1 \ldots w_n$ w ONP:

- Jeśli wyrażenie $w \in \{0,1,\ldots,9\}$, wówczas jego wartość jest równa liczbie w
- Jeśli w=u v op dla wyrażeń u i v i operatora op, to fi(w) = fi(u) op fi(v)

Problem obliczania wartości wyrażenia:

Wejście: postać ONP wyrażenia w

Wyjście: wartość wyrażenia w

Problem obliczania wartości wyrażenia

Problem obliczania wartości wyrażenia:

Wejście: postać ONP wyrażenia w

Wyjście: wartość wyrażenia w

Przykład

Wejście: 5 3 + 7 *

Wyjście: 56

Narzędzie - stos

Używamy stosu jako abstrakcyjnej struktury danych:

- StosPusty() – tworzy pusty stos i zwraca go jako wynik
- Wstaw(A,x) – wstawia element x na szczyt stosu A
- Zdejmij(A) – zdejmuje i zwraca jako wynik element ze szczytu stosu A (o ile stos nie jest pusty)
- CzyPusty(A) – prawda gdy stos A jest pusty, fałsz w przeciwnym przypadku

ONP – wartość wyrażenia

Wejście: $w=w_1\ w_2\ \ldots\ w_n$ – wyrażenie w postaci ONP

Wyjście: wartość wyrażenia w

1. A ← StosPusty()
2. Dla i=1,2,…,n:
   a) $x \leftarrow w_i$
   b) Jeśli x to argument (liczba): Wstaw(A,x)
   c) Jeśli x to operator:
      i. $y_2 \leftarrow$ Zdejmij(A)
      ii. $y_1 \leftarrow$ Zdejmij(A)
      iii. Wykonaj działanie $y_1\ x\ y_2$, wynik na stos:
         - $z \leftarrow y_1\ x\ y_2$
         - Wstaw(A,z)
3. Wynik ← Zdejmij(A)

ONP wartościowanie – przykład

$w = 3 * ((4 + 5) - (2 + 7))$

ONP(w) = 3 4 5 + 2 7 + – *

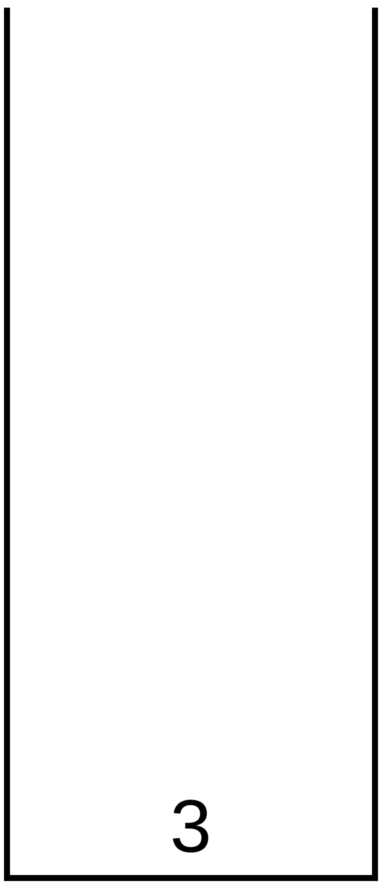
stos

ONP wartościowanie – przykład

w = 3∗ ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – ∗

stos



ONP wartościowanie – przykład

w = 3∗ ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – ∗

stos

4

3

ONP wartościowanie – przykład

$w = 3 * ((4 + 5) - (2 + 7))$

$ONP(w) = 3\ 4\ 5\ +\ 2\ 7\ +\ -\ *$

stos

5

4

3

ONP wartościowanie – przykład

w = 3∗ ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – ∗

stos

9

3

ONP wartościowanie – przykład

w = 3∗ ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – ∗

stos

2

9

3

ONP wartościowanie – przykład

$w = 3 * ((4 + 5) - (2 + 7))$

$\mathrm{ONP}(w) = 3\ 4\ 5\ +\ 2\ 7\ +\ -\ *$

stos

7

2

9

3

ONP wartościowanie – przykład

$w = 3 * ((4 + 5) – (2 + 7))$

ONP(w) = 3 4 5 + 2 7 + – *

stos

9

9

3

ONP wartościowanie – przykład

w = 3 ∗ ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – ∗

stos

0

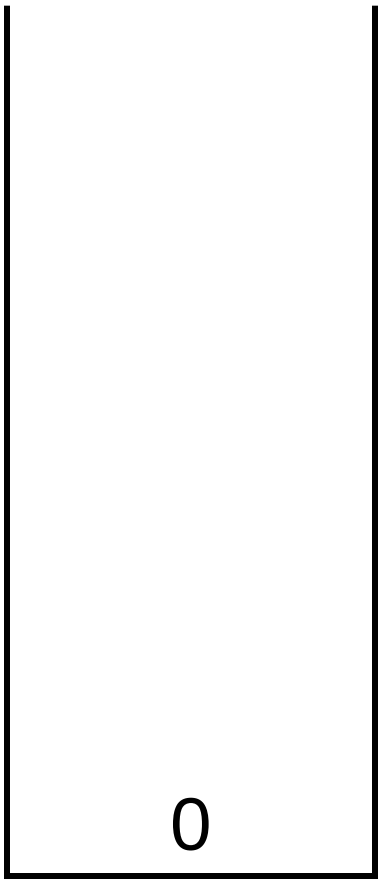
3

ONP wartościowanie – przykład

$w = 3 * ((4 + 5) - (2 + 7))$

$ONP(w) = 3\ 4\ 5 + 2\ 7 + - *$

stos



ONP – wartość wyrażenia

Tw. (Poprawność)
Po krokach 1. i 2. algorytmu na stosie znajduje się wartość wyrażenia w.

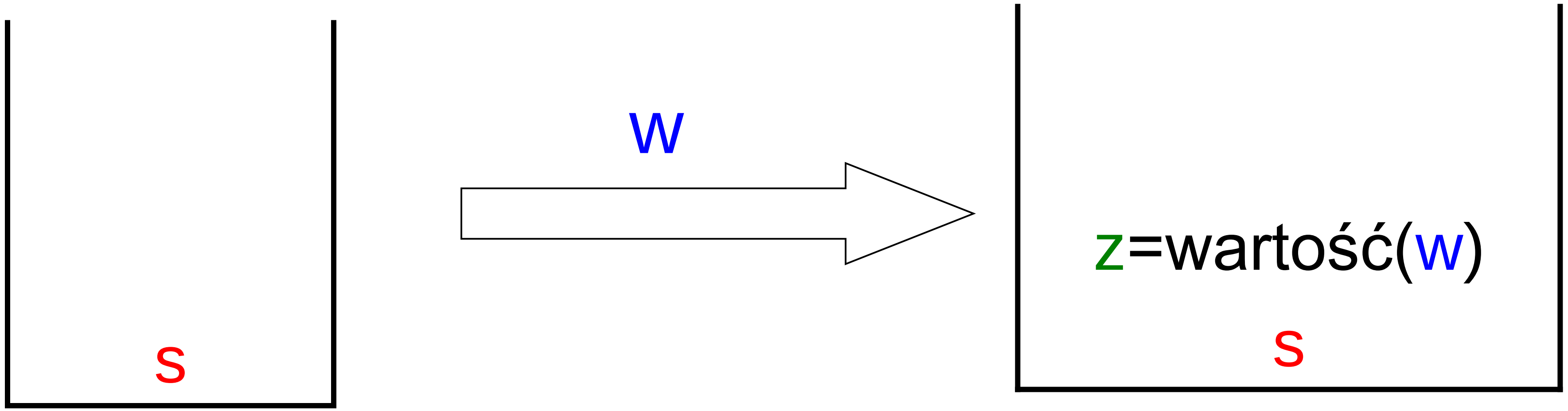
Dowód (szkic)
Indukcja ze względu na długość w:

- w ma długość 1: w to liczba wstawiana na stos w 2.b)
- **Zał.:** tw. spełnione dla w o długości <n oraz n>1
  Teza: tw. spełnione dla w o długości n
  Dowód kroku indukcyjnego: w ma postać u v x,
  gdzie u, v to wyrażenia ONP krótsze niż n, x to operator oraz:
  - a) po przeczytaniu u na stosie wartość u (z zał. ind.)
  - b) po przecz. v na stosie wartość u i wartość v
  - c) po przecz. x na stosie wartość w (2.c)

ONP – wartość wyrażenia

Uwaga. Dla pełnej poprawności dowodu (punkt b)) powinniśmy dowodzić silniejszej tezy:

> Jeśli czytanie wyrażenia w zaczynamy z zawartością stosu s, to po przeczytaniu wyrażenia w zawartość stosu będzie równa s z, gdzie z to wartość wyrażenia w (z jest na szczycie).



Algorytm zamiany wyrażenia na postać ONP – ver. 1 (bez priorytetów)

Zamiana na ONP – ver. 1 (bez priorytetów)

We: $w=w_1\ w_2\ \ldots\ w_n$ – poprawne wyrażenie w postaci „tradycyjnej” (infiksowej), elementy w_i mogą być postaci:

- argument: 0,1,2,…,9
- operator: +, *
- nawias (lub)

Wy: ONP(w)

Przypomnienie: przyjmujemy, że wszystkie operatory mają takie same priorytety i wykonujemy „od lewej do prawej”!

ONP – bez priorytetów

Bez priorytetów (jak najkrótsza prawa strona):

ONP($3*4+5$) = ONP($(3*4)+5$) = $3\ 4\ *\ 5\ +$

ONP($3+4*5$) = ONP($(3+4)*5$) = $3\ 4\ +\ 5\ *$

Gdy uwzględnimy priorytety:

ONP($3*4+5$) = ONP($(3*4)+5$) = $3\ 4\ *\ 5\ +$

ONP($3+4*5$) = ONP($3+(4*5)$) = $3\ 4\ 5\ *\ +$

Zamiana na ONP – ver. 1 (bez priorytetów)

Algorytm 1

1. A ← StosPusty()
2. Dla i=1,2,…,n:
   - $x \leftarrow w_i$
   - Jeśli x to argument (liczba): wypisz x
   - Jeśli x to nawias ”(”: Wstaw(A, ”(”)
   - Jeśli x to operator:
     - Dopóki Szczyt(A) to operator: Wypisz Zdejmij(A)
     - Wstaw(A,x)
   - Jeśli x to nawias ”)”:
     - Dopóki Szczyt(A)≠”(”:
       - Wypisz Zdejmij(A)
     - Zdejmij(A)
3. Dopóki A nie jest pusty: Wypisz Zdejmij(A)

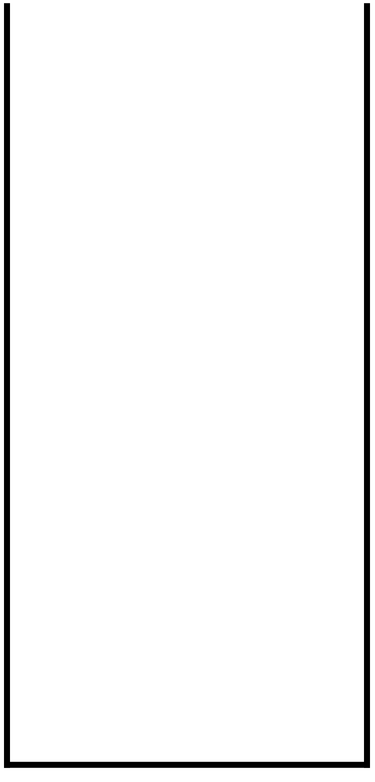
Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście:

Stos:



Zamiana na ONP – przykład

$w = 3 * ((4 + 5) - (2 + 7))$

$ONP(w) = 3\ 4\ 5 + 2\ 7 + - *$

Wyjście: 3

Stos:

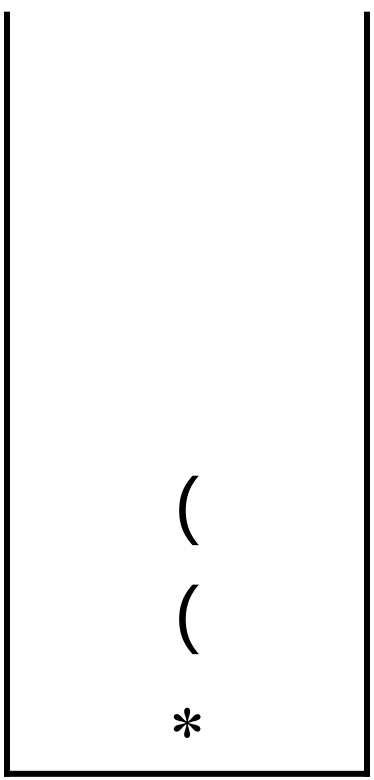
Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7)))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3

Stos:



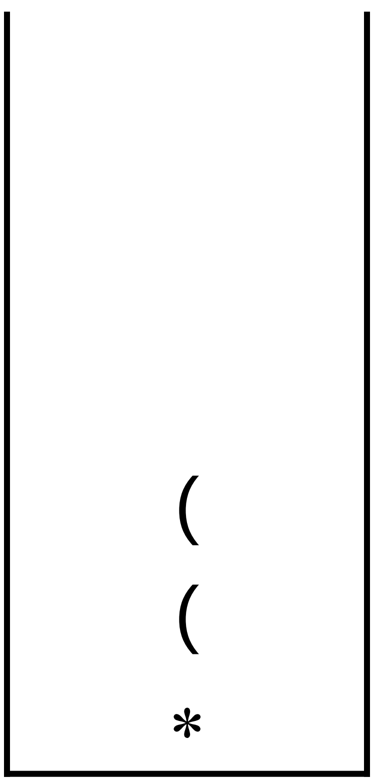
Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4

Stos:



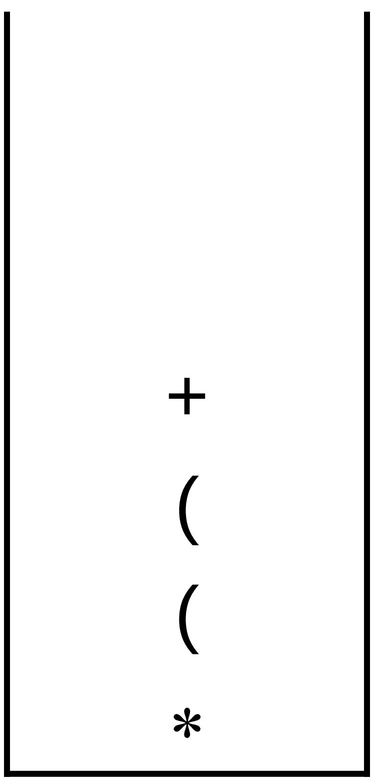
Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4

Stos:



Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4 5

Stos:

+
(
(
*

Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4 5 +

Stos:

(

*

Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4 5 +

Stos:

-

(

*

Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4 5 +

Stos:

(

-

(

*

Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7)))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4 5 + 2

Stos:

(
-
(
*

Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4 5 + 2

Stos:

+
(
-
(
*

Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4 5 + 2 7

Stos:

+
(
-
(
*

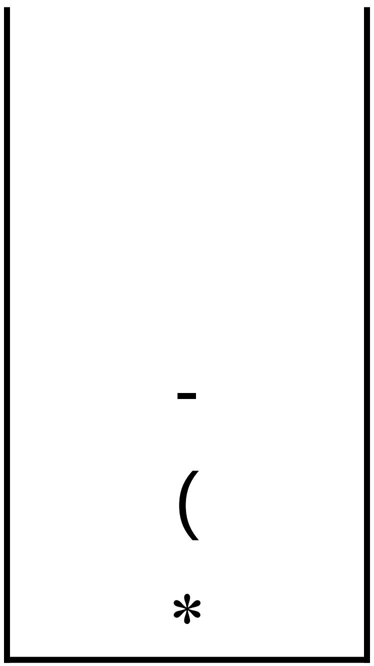
Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4 5 + 2 7 +

Stos:



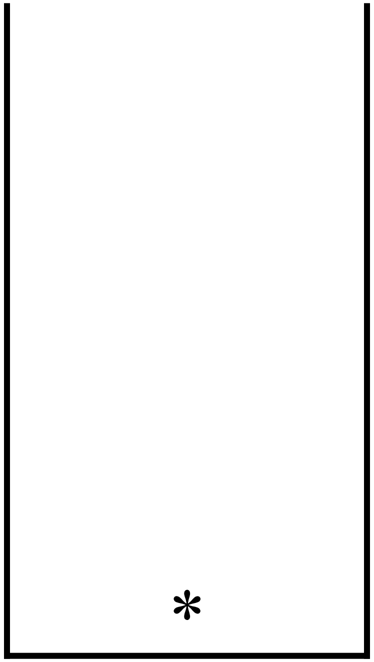
Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7)))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4 5 + 2 7 + –

Stos:



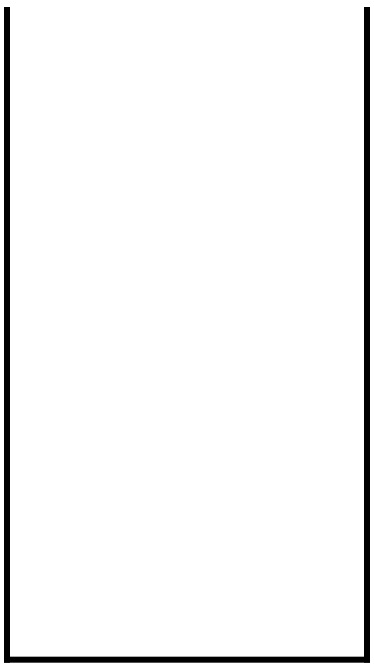
Zamiana na ONP – przykład

w = 3 * ((4 + 5) – (2 + 7)))

ONP(w) = 3 4 5 + 2 7 + – *

Wyjście: 3 4 5 + 2 7 + – *

Stos:



O poprawności algorytmu zamiany wyrażenia na postać ONP

Zamiana na ONP – poprawność

Intuicje:

- operatory w ONP są umieszczane **za** ich argumentami: dlatego operatory odkładamy na stos (do wypisania na wyjściu później), a argumenty od razu wypisujemy na wyjście;
- dlaczego używamy stosu (last in, first out): odpowiada realizacji „**rekurencyjnych**” zagłębień wynikających z definicji poprawnych wyrażeń

Zamiana na ONP – poprawność

Obserwacja o nawiasach
W każdym poprawnym wyrażeniu, przy odczytaniu),
Algorytm 1 zdejmuje ze stosu odpowiadający mu (,
nie zdejmuje niczego poniżej zdejmowanego (.

Dowód:

- indukcja ze względu na długość wyrażenia pomiędzy (i).
- z wykorzystaniem faktu, że tylko odczytanie) powoduje zdjęcie ze stosu (

Zamiana na ONP – poprawność

Lemat o nawiasach.

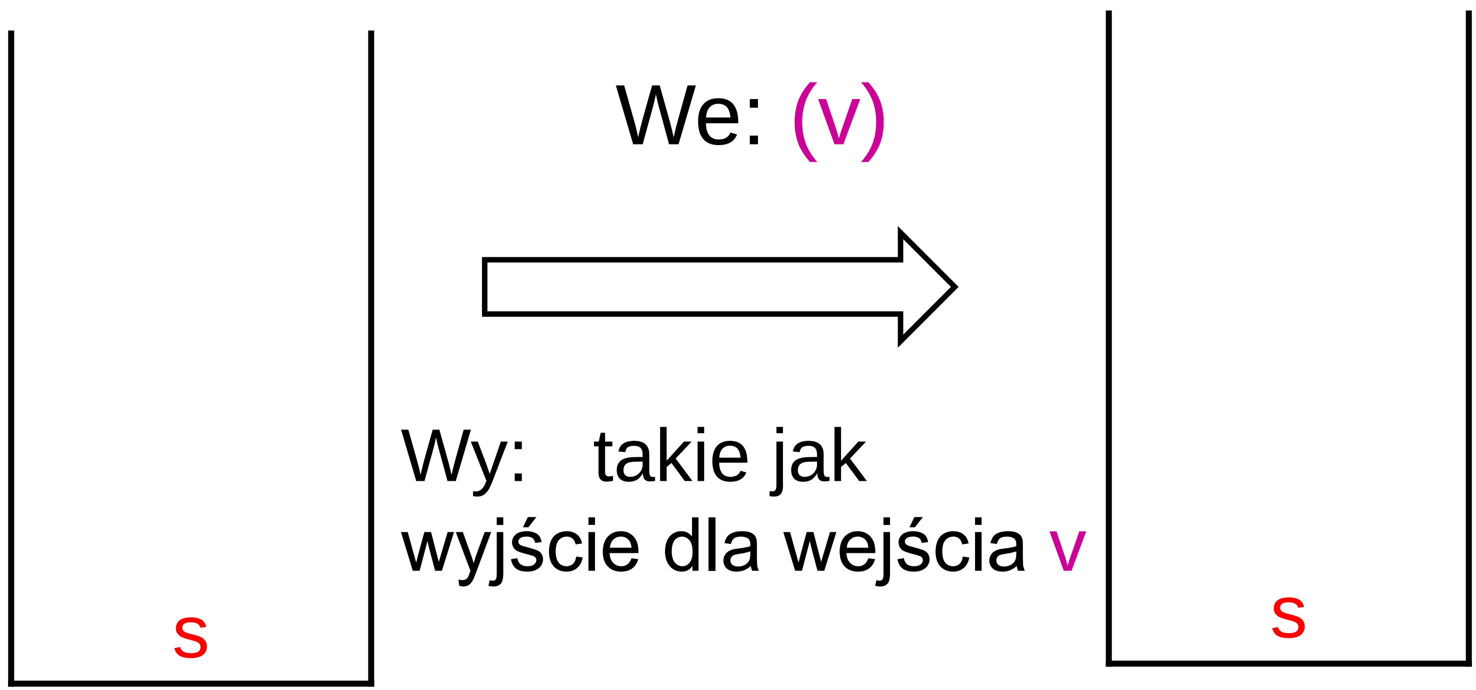
Załóżmy, że w = x (v) w, gdzie v to poprawne wyrażenie oraz zawartość stosu po przeczytaniu x jest równa s.

Wówczas, podczas czytania (v):

a) żaden element s nie będzie zdjęty

b) na wyjściu zostanie wypisane słowo, które algorytm wypisuje dla wyrażenia v,

c) po przeczytaniu (v) zawartość stosu będzie równa s



Zamiana na ONP – ver. 1 (bez priorytetów)

Algorytm 1

1. A←StosPusty()
2. Dla i=1,2,…,n:
   - $x \leftarrow w_i$
   - Jeśli x to argument (liczba): wypisz x
   - Jeśli x to nawias "(": Wstaw(A,x)
   - Jeśli x to operator:
     - Dopóki Szczyt(A) to operator: Wypisz Zdejmij(A)
     - Wstaw(A,x)
   - Jeśli x to nawias ")":
     - Dopóki Szczyt(A)≠"(":
       - Wypisz Zdejmij(A)
     - Zdejmij(A)
3. Dopóki A nie jest pusty: Wypisz Zdejmij(A)

Zamiana na ONP – poprawność

Lemat o nawiasach.

Załóżmy, że w = x (v) w, gdzie v to poprawne wyrażenie oraz zawartość stosu po przeczytaniu x jest równa s.

Wówczas, podczas czytania (v):

a) żaden element s nie będzie zdjęty
b) na wyjściu zostanie wypisane słowo, które algorytm wypisuje dla wyrażenia v,
c) po przeczytaniu (v) zawartość stosu będzie równa s

Dowód (szkic).

Punkty a) i c): z Obserwacji o nawiasach

Punkt b):

z a) i c) wynika, że na dla wejścia w = x (v) w, algorytm zachowuje się na fragmencie v tak samo jakby miał na wejściu v:

- zaczyna ze stosem s (
- nie „widzi" s na stosie (patrz a)), a („zobaczy" dopiero czytając)
- odczytanie) ze słowa x (v) w odpowiada krokowi 3 algorytmu uruchomionego na wejściu v

Zamiana na ONP – poprawność

Lemat o końcu obliczeń.

Załóżmy, że Algorytm 1 uruchamiamy na poprawnym wyrażeniu w. Niech s to zawartość stosu po zakończeniu kroku 2, przed krokiem 3. Wówczas s zawiera tylko operatory (nie ma nawiasów otwierających)

Zamiana na ONP – poprawność

Lemat o końcu obliczeń.

Załóżmy, że Algorytm 1 uruchamiamy na poprawnym wyrażeniu w. Niech s to zawartość stosu po zakończeniu kroku 2, przed krokiem 3. Wówczas s zawiera tylko operatory (nie ma nawiasów otwierających)

Dowód

- wprost z Obserwacji o nawiasach.

Zamiana na ONP – poprawność

Obserwacja.

Każde poprawne wyrażenie w ma postać:

w to liczba („argument”) LUB

w = u p v LUB

w = (u)

gdzie

u – poprawne wyrażenie

p – operator

v – to argument lub v=(x) dla poprawnego wyrażenia x.

Zamiana na ONP – poprawność

Tw. Algorytm 1 podaje na wyjściu poprawną postać ONP (bez uwzględnienia priorytetów).

Zamiana na ONP – poprawność

Tw. Algorytm 1 podaje na wyjściu poprawną postać ONP (bez uwzględnienia priorytetów).

Szkic dowodu indukcyjnego względem długości w:

1. Długość w równa 1: proste
2. Krok indukcyjny:

   Zakładamy prawdziwość dla wyrażeń o długości < n

   I. Weźmy w = u p v o długości n **(inne przypadki podobne)**, gdzie p to operator, oraz

   a) u - wyrażenie

   b) v postaci (x) dla wyrażenia x:

   Wówczas

   - ONP(w) = ONP(u) ONP(v) p
   - Po przeczytaniu u p na wyjściu ONP(u), a stos zawiera tylko p – z zał. ind. i lematu o końcu obliczeń (czytając p zdejmujemy ze stosu operatory tak jak w kroku 3)
   - Po przeczytaniu v = (x) – na wyjściu ONP(u) ONP(v) p (z lematu o nawiasach)

Zamiana na ONP – poprawność

Tw. Algorytm 1 podaje na wyjściu poprawną postać ONP (bez uwzględnienia priorytetów).

Szkic dowodu indukcyjnego względem długości w, **ciąg dalszy** kroku indukcyjnego:
II. Weźmy $w=(u)$: poprawność wynika z poprawności działania dla u (długość u mniejsza niż n, więc możemy skorzystać z założenia indukcyjnego).

Zamiana na ONP – poprawność

Wejście: u p v,

Czytamy	**Stos po przeczytaniu**	**Wyjście**
u p	p	ONP(u)
v		ONP(u) ONP(v) p

Zamiana na ONP z priorytetami operatorów

Wartościowanie i ONP dla wyrażeń z priorytetami

Uproszczenia:

- ~~wszystkie operatory mają taki sam priorytet (kolejność można wymusić nawiasami)~~
- Kolejność od lewej do prawej, ale z uwzględnieniem priorytetów:

standardowo	ONP	
5 + 7 + 9	5 7 + 9 +	~~5 7 9 + +~~
5 + 7 * 9	5 7 9 * +	

Wartościowanie i ONP dla wyrażeń z priorytetami

Wartość fi(w) wyrażenia $w=w_1 \dots w_n$:

- Jeśli wyrażenie $w \in \{0,1,\dots,9\}$, wówczas ONP(w)=w
- Jeśli w=u p v dla wyrażeń u i v i operatora p, oraz
  - u nie można przedstawić jako x q y dla wyrażeń x, y i operatora q o mniejszym priorytecie od priorytetu p
  - v nie można przedstawić jako x q y dla wyrażeń x, y i operatora q o mniejszym lub równym priorytecie priorytecie od priorytetu p

  to fi(w) = fi(u) p fi(v) oraz ONP(w)=ONP(u)ONP(v)p

Wartościowanie i ONP - priorytety

ONP($3 + 4 \ast 5$) = $3\ 4\ 5 \ast +$

- poprawny podział: 3 $+$ $4 \ast 5$
- niepoprawny podział: $3+4$ $\ast$ 5

Uwaga: na lewo od $\ast$ mamy $3+4$ i $+$ ma mniejszy priorytet od $\ast$

ONP($3 \ast 4 + 5$) = $3\ 4 \ast 5 +$

- poprawny podział: $3 \ast 4$ $+$ 5
- niepoprawny podział: 3 $\ast$ $4+5$

Uwaga: na lewo od $+$ nie ma operatora o mniejszym priorytecie

ONP($3 + 4 + 5$) = $3\ 4 \ast 5 +$

- poprawny podział: $3+4$ $+$ 5
- niepoprawny podział: 3 $+$ $4+5$

Zamiana na ONP – ver. 2 (z priorytetami)

We: $w=w_1\ w_2\ \ldots\ w_n$ – poprawne wyrażenie w postaci „tradycyjnej” (infiksowej)

Wy: ONP(w)

Zamiana na ONP – ver. 2 (z priorytetami)

Algorytm 2

1. A←StosPusty()
2. Dla i=1,2,…,n:
   - $x \leftarrow w_i$
   - Jeśli x to argument: wypisz x
   - Jeśli x to nawias „(„: Wstaw(A, „(„)
   - Jeśli x to operator:
     - Dopóki Szczyt(A) to operator o większym priorytecie niż x bądź równym priorytetowi x:
       - Wypisz Zdejmij(A)
     - Wstaw(A,x)
   - Jeśli x to nawias „)”:
     - Dopóki Szczyt(A)≠”(„:
       - Wypisz Zdejmij(A)
     - Zdejmij(A)
3. Dopóki A nie jest pusty: Wypisz Zdejmij(A)

Dla porównania….

Zamiana na ONP – ver. 1 (bez priorytetów)

Algorytm 1

1. A←StosPusty()
2. Dla i=1,2,…,n:
   - $x \leftarrow w_i$
   - Jeśli x to argument: wypisz x
   - Jeśli x to nawias ”(”: Wstaw(A, „(„)
   - Jeśli x to operator:
     - Dopóki Szczyt(A) to operator: Wypisz Zdejmij(A)
     - Wstaw(A,x)
   - Jeśli x to nawias ”)”:
     - Dopóki Szczyt(A)≠”(”:
       - Wypisz Zdejmij(A)
     - Zdejmij(A)
3. Dopóki A nie jest pusty: Wypisz Zdejmij(A)

Zamiana na ONP – ver. 2 (z priorytetami)

Intuicja

Gdy czytamy operator p to zamykamy „niewidoczne” nawiasy wynikające z wyższych priorytetów wcześniejszych operatorów na tym samym poziomie nawiasowania (w tym samym podwyrażeniu).

Przykład

$5 * 4 + 3$

$5 + 4 * 3$

$7 + (5 + 4 * 3)$

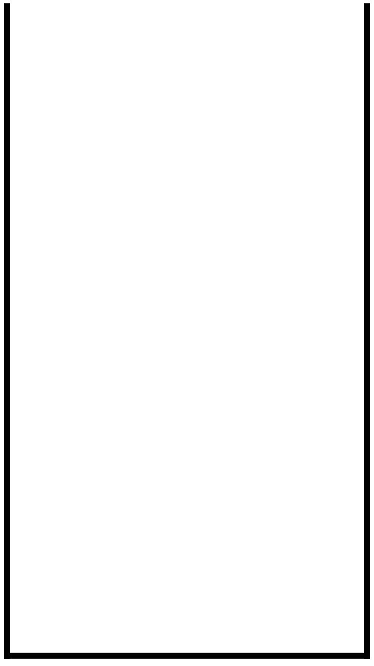
Zamiana na ONP – ver. 2 (z priorytetami)

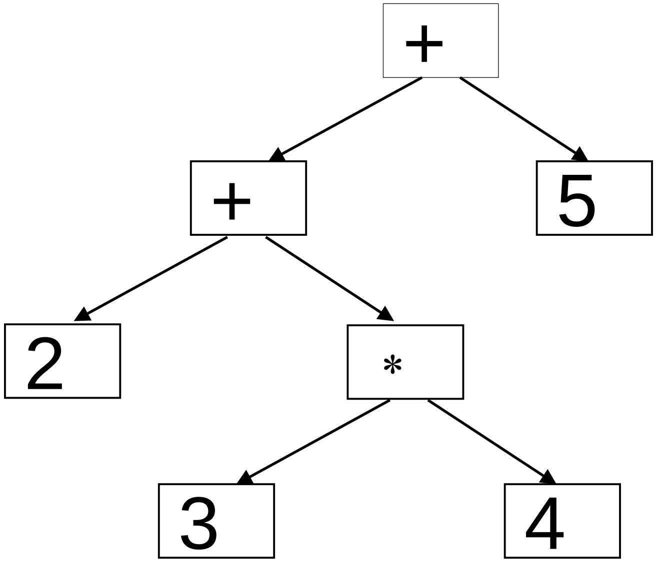
w = 2 + 3 * 4 + 5

ONP(w) = 2 3 4 * + 5 +

Wyjście:

Stos:





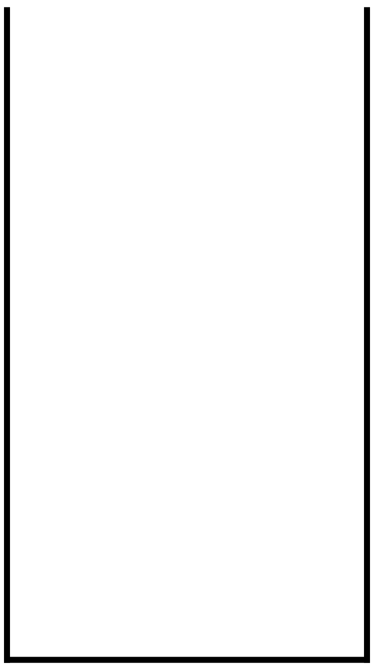
Zamiana na ONP – ver. 2 (z priorytetami)

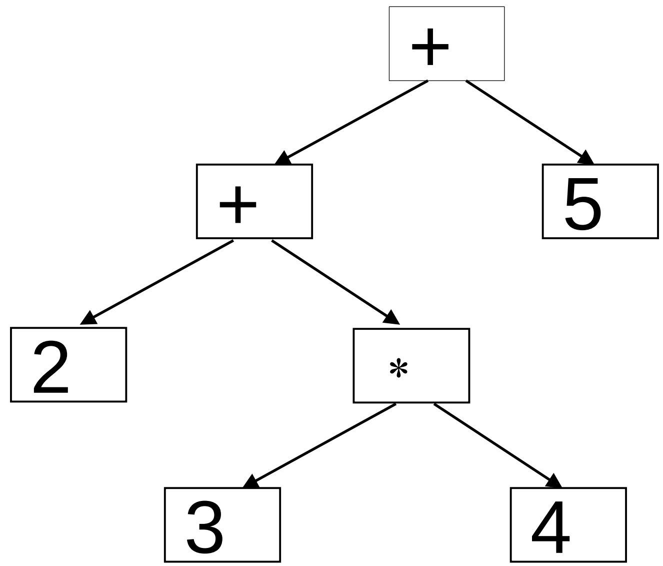
w = 2 + 3 * 4 + 5

ONP(w) = 2 3 4 * + 5 +

Wyjście: 2

Stos:





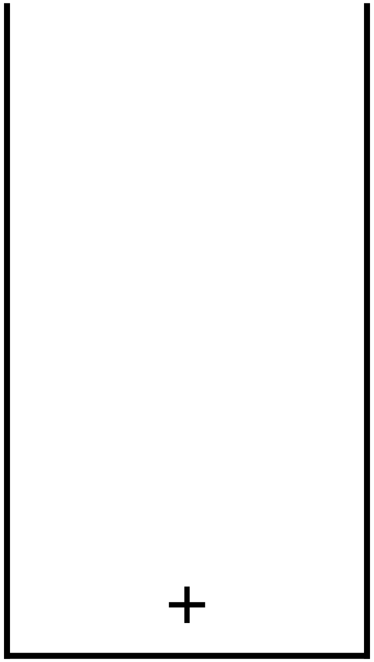
Zamiana na ONP – ver. 2 (z priorytetami)

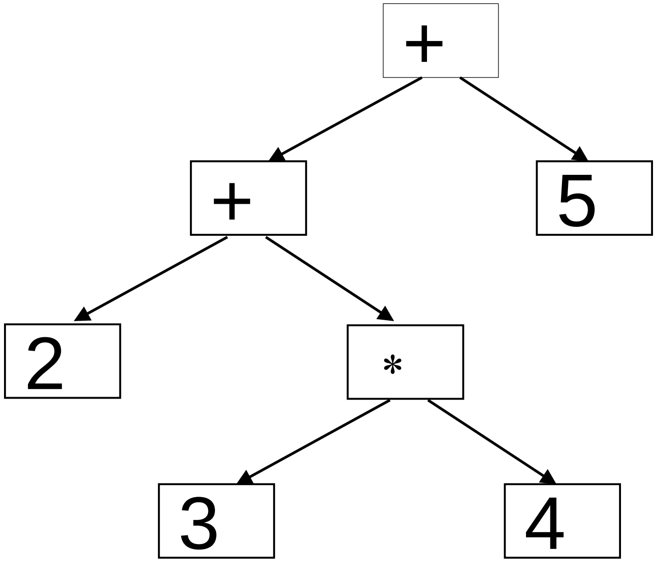
w = 2 + 3 * 4 + 5

ONP(w) = 2 3 4 * * 5 +

Wyjście: 2

Stos:





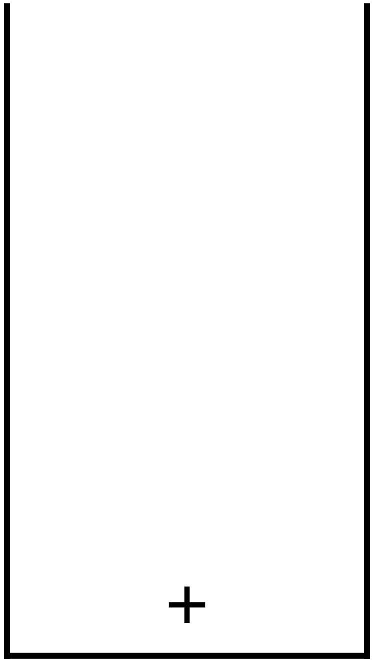
Zamiana na ONP – ver. 2 (z priorytetami)

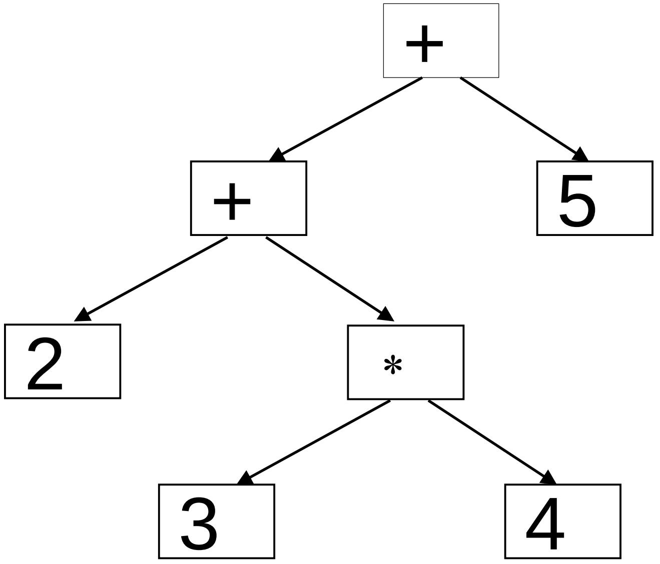
w = 2 + 3 * 4 + 5

ONP(w) = 2 3 4 * * 5 +

Wyjście: 2 3

Stos:





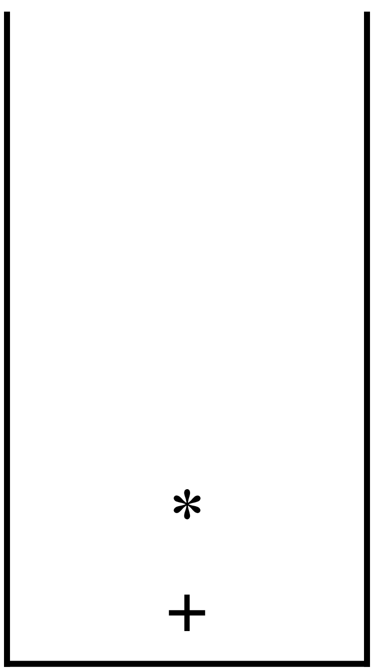
Zamiana na ONP – ver. 2 (z priorytetami)

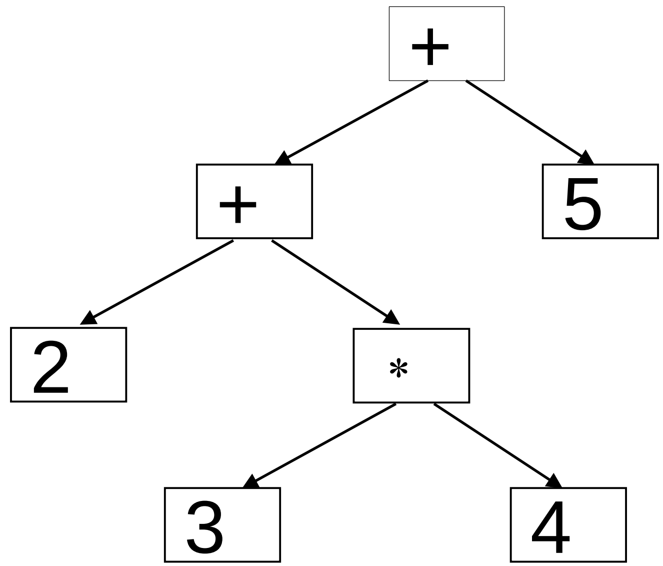
w = 2 + 3 * 4 + 5

ONP(w) = 2 3 4 * * 5 +

Wyjście: 2 3

Stos:





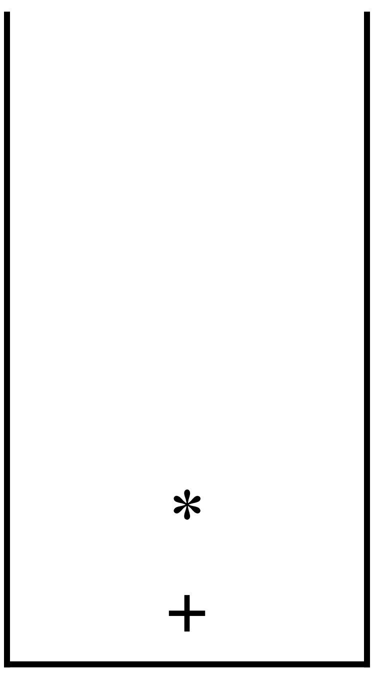
Zamiana na ONP – ver. 2 (z priorytetami)

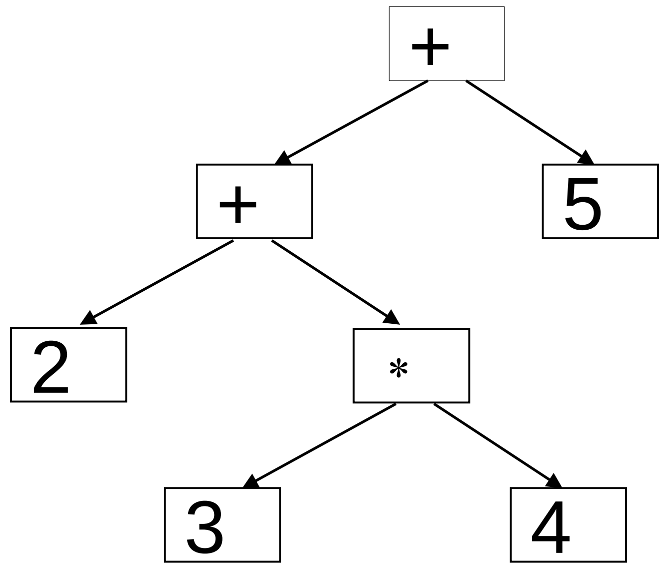
w = 2 + 3 * 4 + 5

ONP(w) = 2 3 4 * + 5 +

Wyjście: 2 3 4

Stos:





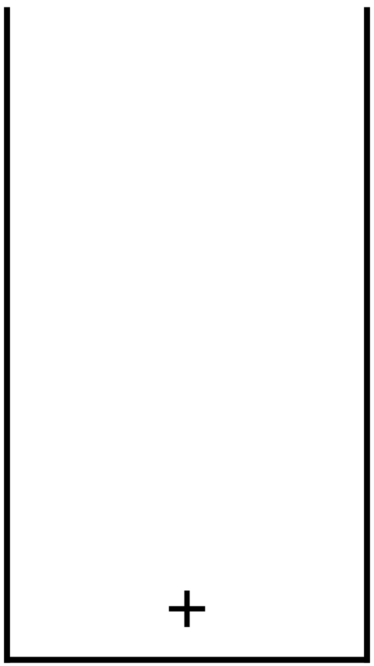
Zamiana na ONP – ver. 2 (z priorytetami)

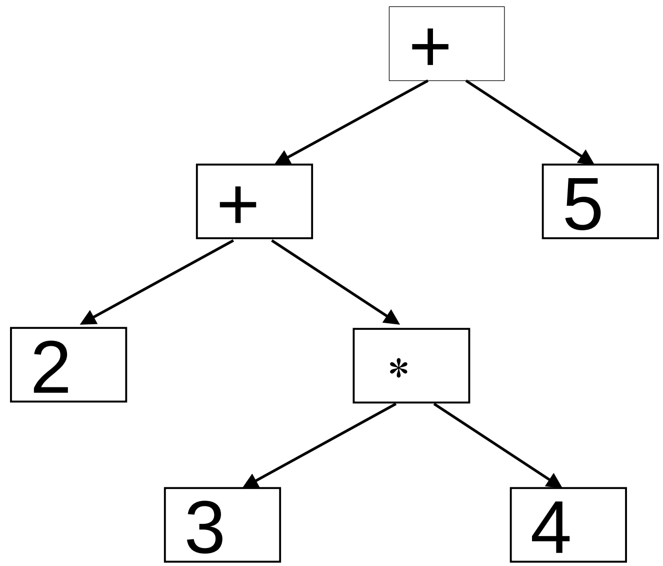
w = 2 + 3 * 4 + 5

ONP(w) = 2 3 4 * + 5 +

Wyjście: 2 3 4 * +

Stos:





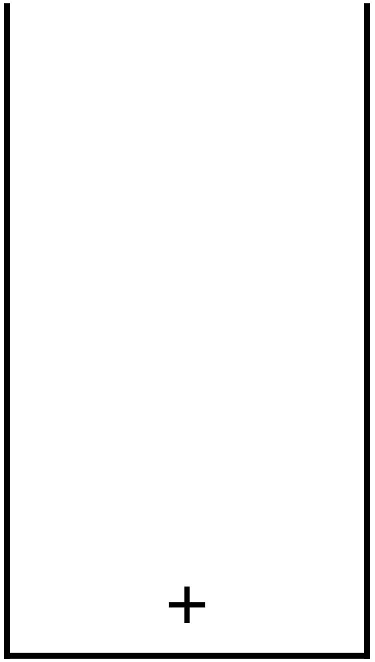
Zamiana na ONP – ver. 2 (z priorytetami)

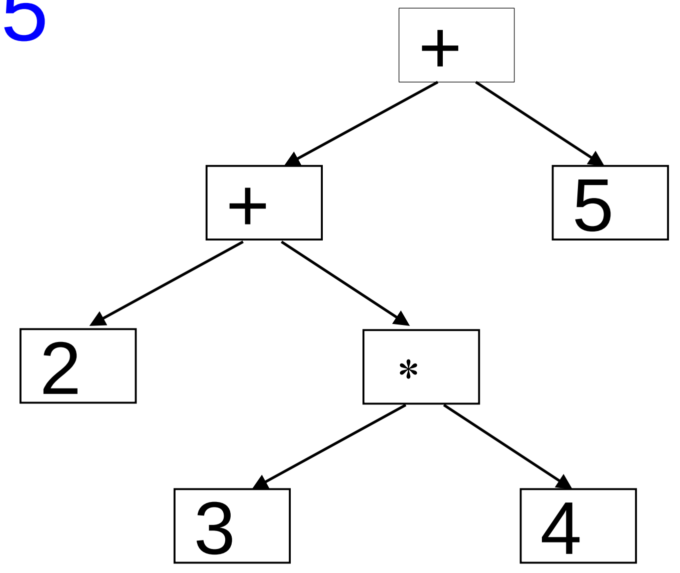
w = 2 + 3 * 4 + 5

ONP(w) = 2 3 4 * + 5 +

Wyjście: 2 3 4 * + 5

Stos:





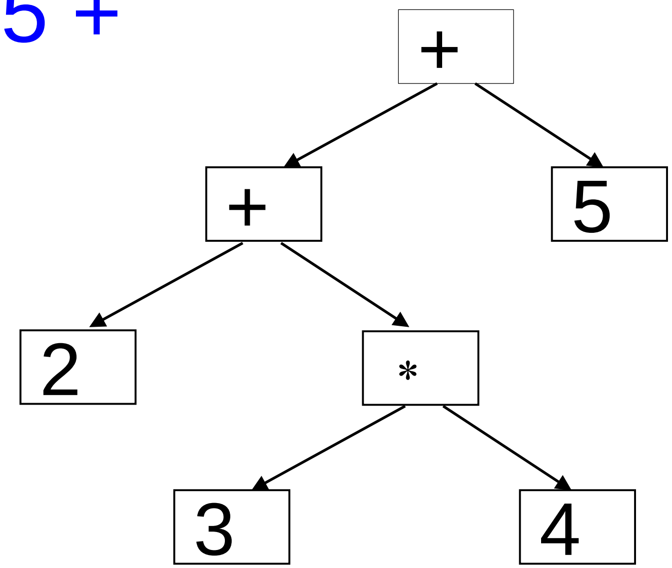
Zamiana na ONP – ver. 2 (z priorytetami)

w = 2 + 3 ∗ 4 + 5

ONP(w) = 2 3 4 ∗ + 5 +

Wyjście: 2 3 4 ∗ + 5 +

Stos:



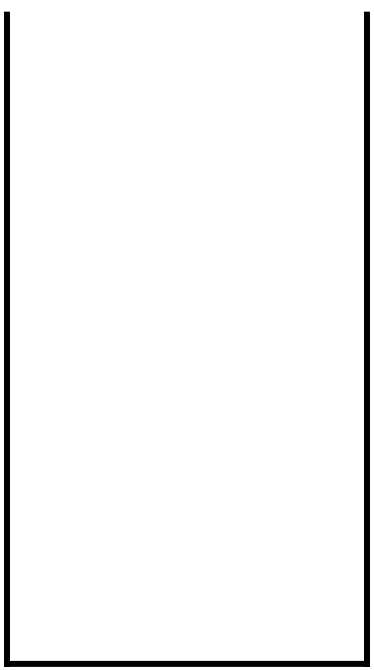
Zamiana na ONP – ver. 2 (z priorytetami)

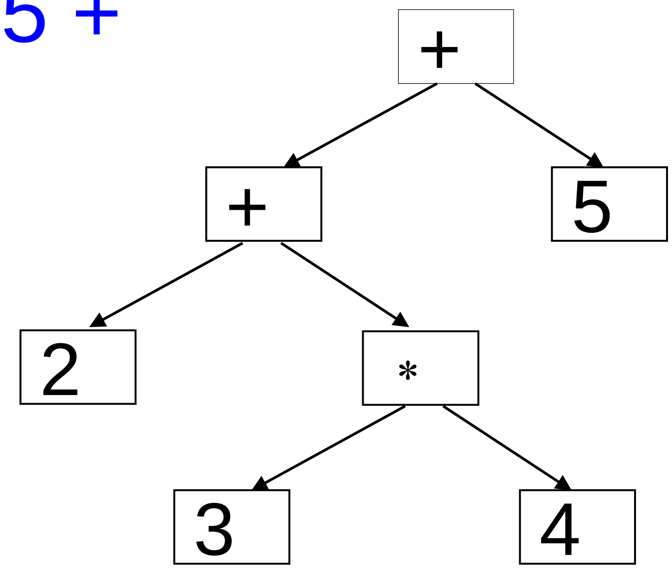
w = 2 + 3 * 4 + 5 = (2 + (3 * 4)) + 5

ONP(w) = 2 3 4 * + 5 +

Wyjście: 2 3 4 * + 5 +

Stos:





Podsumowanie ONP

- Prosty algorytm wartościowania wyrażenia;
- Efektywna konwersja wyrażeń w formie standardowej do postaci ONP
  - Konwersja bez priorytetów operatorów – ze szkicem dowodu poprawności na wykładzie
  - Konwersja z priorytetatmi operatorów – bez dowodu poprawności na wykładzie