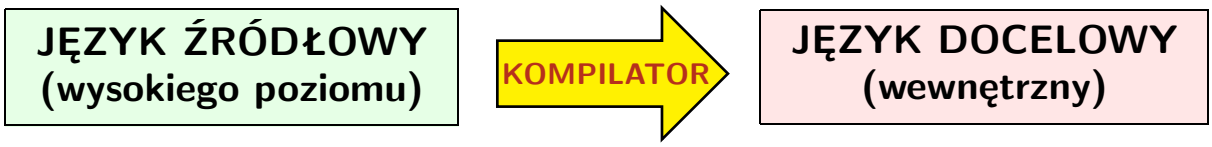
Stefan Sokołowski

JĘZYKI FORMALNE I METODY KOMPILACJI

wykład 1

Inst. Informatyki Stosowanej, ANS
Elbląg, 2024/2025

Języki formalne



Kompilator musi „znać" oba języki...

Jak program komputerowy nauczyć języka?
Najpierw trzeba ten język ściśle **zdefiniować**.

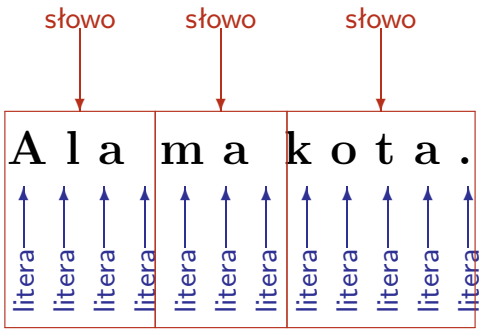
Języki formalne mają u podstaw sposoby ścisłego definiowania języków.

Języki formalne

litera — słowo — język

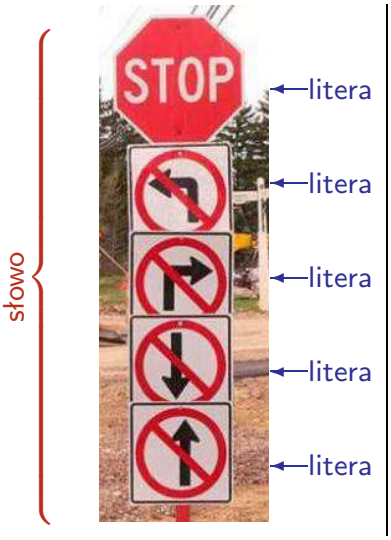
- litera: niekoniecznie ze standardowego alfabetu
- słowo: skończony ciąg liter
- język: zbiór słów

PRZYKŁAD:



Języki formalne

PRZYKŁADY:



litery:

```
class Test {
  public static void main
    ( String [ ] args ) {
    System . out . println
      ( "To tylko test." ) ;
  }
}
```

Proste języki formalne (przykłady)

$\{a, aback, abandon, abase, \ldots, zoom\}$
— język (skończony) składający się ze słów języka angielskiego

$\{0, 1, 10, 11, 100, 101, \ldots\}$
— język (nieskończony) składający się z liczb binarnych

puste **(to nie jest litera!)**

$\left\{a^n b^n \,\middle|\, n \geqslant 0\right\} = \{\lambda, ab, aabb, aaabbb, \ldots\}$
— język (nieskończony) składający się z takich słów, w których

- występują tylko litery a i b,
- liczba liter a jest równa liczbie liter b,
- wszystkie litery a poprzedzają wszystkie litery b.

Języki formalne

DEFINICJE:
Σ — skończony niepusty *alfabet*, czyli zbiór tzw. *liter*
słowo — dowolny ciąg skończonej długości o elementach z Σ
Σ^* — zbiór **wszystkich** słów
język — dowolny zbiór słów $L \subseteq \Sigma^*$

PRZYKŁAD: nie mylić! —
λ — słowo *puste* (ciąg liter o długości 0)
Uwaga: to nie jest litera!
$\emptyset \stackrel{\text{def}}{=} \{\,\}$ — język pusty (nie zawierający żadnego słowa)
$\Lambda \stackrel{\text{def}}{=} \{\lambda\}$ — język zawierający wyłącznie słowo puste

PRZYKŁAD:
$1(*2-1)(1-(2-*1)()$ — słowo nad alfabetem $\Sigma \stackrel{\text{def}}{=} \{(,), -, *, 1, 2\}$
$\{\lambda, a, b, ab, ba\}$ — język nad alfabetem $\Sigma \stackrel{\text{def}}{=} \{a, b\}$ zawierający pięć słów
$\left\{a^n \,\middle|\, n \geqslant 0\right\} = \{\lambda, a, aa, aaa, \ldots\}$ — jęz. nieskończ. nad alf. $\Sigma \stackrel{\text{def}}{=} \{a\}$

Jak porządnie opisać język nieskończony?

- $\{\lambda, ab, aabb, aaabbb, \ldots\}$ — co oznaczają kropeczki?
- $\left\{a^n b^n \,\middle|\, n \geqslant 0\right\}$ — lepiej, ale są niepożądane elementy obce (liczby, zmienne, relacje, . . .)

Język definiujemy na dwa sposoby. Przez

- **generator** — cokolwiek wyprodukuje, uważamy za należące do języka
- **akceptor** — cokolwiek zatwierdzi, uważamy za należące do języka

Opis języka programowania przeznaczony dla programisty powinien być **generatorem** — ma wyjaśniać, jak pisać poprawne programy.

Kompilator języka programowania powinien zawierać **akceptor** — sprawdzający, czy program na jego wejściu należy do języka.

Generatory: gramatyki bezkontekstowe

DEFINICJA: Gramatyka bezkontekstowa G — czwórka $\langle \Sigma, N, P, S \rangle$

- alfabet *terminalny* Σ;
- alfabet *nieterminalny* lub *pomocniczy* N (rozłączny z Σ);
- skończony zbiór P *produkcji* postaci $A \to w$, gdzie
  $A \in N$ (czyli jest pojedynczym nieterminalem),
  $w \in (N \cup \Sigma)^*$ (czyli jest słowem z terminali i nieterminali);
- wybrany nieterminal $S \in N$ nazywany *nieterminalem początkowym* albo *aksjomatem* gramatyki.

PRZYKŁAD:

Terminale:	a, b	Produkcje:	$X \to \lambda$
Nieterminale:	X		$X \to aXa$
Aksjomat:	X		$X \to bXb$

Jak gramatyka definiuje język?

DEFINICJA: Wywodliwość —

$uXv \Rightarrow_G uwv$ czyli
uwv jest *bezpośrednio wywodliwe* z uXv
$\iff$ def
w P jest produkcja $X \to w$

$w_1 \Rightarrow^*_G w_n$ czyli
w_n jest *wywodliwe* z w_1
$\iff$ def
$w_1 \Rightarrow_G w_2 \Rightarrow_G w_3 \ldots \Rightarrow_G w_{n-1} \Rightarrow_G w_n$

DEFINICJA: Język $L(G)$ *generowany przez gramatykę* G — składa się z tych słów wywodliwych **z aksjomatu**, które już nie zawierają nieterminali: $L(G) \stackrel{\text{def}}{=} \left\{ w \in \Sigma^* \,\middle|\, S \Rightarrow^*_G w \right\}$.

PRZYKŁAD:

Gramatyka:
$X \to \lambda$
$X \to aXa$
$X \to bXb$

$X \Rightarrow^* bbaabb \in L(G)$

LEMAT:
$L(G) = \left\{ ww^R \,\middle|\, w \in \{a,b\}^* \right\}$

$X \Rightarrow bXb \Rightarrow bbXbb \Rightarrow bbaXabb \Rightarrow bbaabb$

Jak gramatyka definiuje język?

NIE MYLIĆ:

- $\to$ — formalny symbol w produkcji w gramatyce
- $\Rightarrow$ — relacja bezpośredniej wywodliwości:
  jeśli $X \to w$ to $X \Rightarrow w$
  ale np. $0X1 \Rightarrow 00X11$
  a **nieprawda, że** $0X1 \to 00X11$
- $\Rightarrow^*$ — relacja wywodliwości:
  jeśli $w_1 \Rightarrow w_2$ to $w_1 \Rightarrow^* w_2$
  ale np. $X \Rightarrow^* 0011$
  a **nieprawda, że** $X \Rightarrow 0011$

$X \to \lambda$
$X \to 0X1$

Jak gramatyka definiuje język?

PRZYKŁAD:

Gramatyka G:

$$\begin{array}{lcl} E & \to & T \\ E & \to & E + T \\ T & \to & F \\ T & \to & T * F \\ F & \to & a \\ F & \to & (\, E \,) \end{array}$$

$$\begin{array}{l} E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \\ \Rightarrow a + T * F \Rightarrow a + F * F \Rightarrow a + a * F \\ \Rightarrow a + a * a \in L(G) \end{array}$$

Jak gramatyka definiuje język?

PRZYKŁAD:

Gramatyka G:

$$\begin{array}{lcl} E & \to & T \\ E & \to & E + T \\ T & \to & F \\ T & \to & T * F \\ F & \to & a \\ F & \to & (\, E \,) \end{array}$$

$$\begin{array}{l} E \Rightarrow T \Rightarrow F \Rightarrow (\, E \,) \Rightarrow (\, E + T \,) \Rightarrow (\, T + T \,) \\ \Rightarrow (\, F + T \,) \Rightarrow (\, a + T \,) \Rightarrow (\, a + F \,) \\ \Rightarrow (\, a + a \,) \in L(G) \end{array}$$

$L(G) =$ wyrażenia *arytmetyczne* zbudowane z a, $+$, $*$, $($ i $)$