

# Spis treści

<b>1</b>	<b>Literatura</b>	<b>2</b>
<b>2</b>	<b>Język</b>	<b>2</b>
2.1	Funkcje języka . . . . .	2
2.2	Nauka o języku . . . . .	3
2.3	Definicja . . . . .	3
<b>3</b>	<b>Alfabet</b>	<b>3</b>
<b>4</b>	<b>Słowo</b>	<b>3</b>
4.1	Konkatenacja . . . . .	3
4.2	Podsłowo . . . . .	3
4.3	Długość słowa . . . . .	3
4.4	Potęga słowa . . . . .	3
4.5	Odbicie . . . . .	4
<b>5</b>	<b>Język</b>	<b>4</b>
5.1	Konkatenacja języków . . . . .	4
5.2	Potęga języka . . . . .	4
5.3	Odbicie języka . . . . .	4
5.4	Dzielenie słów . . . . .	4
<b>6</b>	<b>Domknięcie Kleenego</b>	<b>5</b>
<b>7</b>	<b>Automaty</b>	<b>5</b>
7.1	Deterministyczne automaty skończone . . . . .	5
7.1.1	Funkcja przejść . . . . .	6
7.1.2	Rozszerzona funkcja przejść . . . . .	6
7.1.3	Przykład . . . . .	6
7.2	Niedeterministyczne automaty skończone . . . . .	7
7.2.1	Rozszerzona funkcja przejść . . . . .	7
7.2.2	Twierdzenie Scotta . . . . .	8
7.2.3	Przekształcenie $\text{nidet} \rightarrow \text{det}$ . . . . .	8
7.3	Automaty z przejściem . . . . .	9
7.3.1	Domknięcie stanu . . . . .	10
7.3.2	Domknięcie zbioru stanów . . . . .	10
7.3.3	Rozszerzona funkcja przejść . . . . .	10
7.3.4	Przekształcenie $\epsilon \rightarrow \text{ndet}$ . . . . .	10
<b>8</b>	<b>Wyrażenia regularne</b>	<b>10</b>
8.1	Operacje . . . . .	10
8.2	Przykłady . . . . .	11
8.3	Tw. Kleenego . . . . .	11
8.3.1	$w = u + v$ . . . . .	11
8.3.2	$w = uv$ . . . . .	11
8.3.3	$w = u^*$ . . . . .	11
<b>9</b>	<b>Klasy języków</b>	<b>12</b>
9.1	Języki regularne . . . . .	12
9.1.1	Lemat o pompowaniu dla języków regularnych . . . . .	12
9.1.2	Przechodniość regularności . . . . .	13
9.2	Bezkontekstowe . . . . .	13
9.2.1	Część wspólna języków bezkontekstowych . . . . .	13

<b>10 Gramatyki</b>	<b>13</b>
10.1 Wyprowadzanie słowa w jednym kroku . . . . .	14
10.2 Wyprowadzanie słowa w wielu krokach . . . . .	14
10.3 Język generowany przez gramatykę . . . . .	14
10.3.1 Przykład prosty . . . . .	14
10.3.2 Przykład złożony . . . . .	14
10.3.3 Przykład prosty . . . . .	14
10.3.4 Przykład złożony . . . . .	14
10.4 Rodzaje gramatyk . . . . .	14
10.5 Gramatyki typu 3 . . . . .	15
10.5.1 Gramatyki Normalne typu 3 . . . . .	15
10.5.2 Gramatyki liniowe . . . . .	15
10.6 Gramatyki bezkontekstowe . . . . .	15
10.6.1 Problem należenia słowa pustego . . . . .	15
10.6.2 Gramatyki normalne bezkontekstowe . . . . .	15
10.6.3 Problem słowa . . . . .	15
10.6.4 Lemat o pompowaniu dla gramatyk bezkontekstowych . . . . .	16
10.7 Gramatyki kontekstowe . . . . .	16
10.8 Gramatyki ogólne . . . . .	16
10.9 Przekształcenie automatu na gramatykę . . . . .	16
10.10 Przekształcenie gramatyki na automat . . . . .	16
<b>11 Algorytmy</b>	<b>17</b>
11.1 Problem . . . . .	17
11.1.1 Problem rozstrzygalny . . . . .	17
11.1.2 Problem pustości . . . . .	17
11.1.3 Problem skończoności . . . . .	17
11.1.4 Problem nieskończoności . . . . .	18
11.1.5 Problem równości . . . . .	18

## 1 Literatura

- J.E. Hopcroft "Wprowadzenie do teorii automatów i obliczeń"
- M. Sipser "Wprowadzenie do teorii obliczeń"
- G.E. Revesz "Introduction to formal languages"
- H.R. Lewin, Papadimitriou "Elements of the Theory of Computation"

## 2 Język

$$\infty \text{ zdań} + n \text{ reguł} = \text{język}$$

### 2.1 Funkcje języka

1. Poznawcza
2. Społeczna
3. Ekspresywna

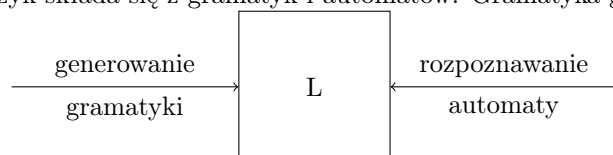
## 2.2 Nauka o języku

1. syntaktyka - budowa
2. semantyka - co znaczy?
3. pragmatyka - jak się używa?

Przykład:  $2 + 3 \cdot 4$ : różna semantyka  $\rightarrow$  wieloznaczność syntaktyczna

## 2.3 Definicja

Język składa się z gramatyk i automatów. Gramatyka generuje język, automat rozpoznaje język.



## 3 Alfabet

Alfabet to zbiór atomowych dozwolonych symboli

Przykład:  $\{a, b, c, d\}$

## 4 Słowo

Słowo to skończony ciąg symboli nad alfabetem.

- $\varepsilon$  - słowo puste
- $\{K, L, O, P, S\} \neq "KLOPS"$ , ponieważ słowa mają dodane znaczenie, w postaci tego do którego języka należą.

### 4.1 Konkatenacja

- Dla  $P = a_1 \dots a_n$  i  $Q = b_1 \dots b_n$ , to  $PQ = a_1 \dots a_n b_1 \dots b_n$
- $P\epsilon = P$
- $\epsilon\epsilon = \epsilon$

### 4.2 Podśłowo

- $P = Q_1|Q|Q_2$
- $Q \subset P$

### 4.3 Długość słowa

- $|\epsilon| = 0$
- $|Pa| = |P| + 1$
- $|PQ| = |P| + |Q|$

### 4.4 Potęga słowa

- $P^0 = \epsilon$
- $P^{n+1} = P^n P$

## 4.5 Odbicie

- $\epsilon^{-1} = \epsilon$
- $(Pa)^{-1} = aP^{-1}$

## 5 Język

Zbiór dozwolonych słów nad alfabetem.

- $V^*$  - zbiór wszystkich języków
- $V^+ = V^* \epsilon$
- $L \in V^*$
- $\{a, ab\} \neq \epsilon, a, ab$  ponieważ inaczej operacje na językach by nie działały

### 5.1 Konkatenacja języków

$$L_1 = \{a, aa\}, L_2 = \{b, aba\}, L_1L_2 = \{ab, aaba, aab, aaaba\}$$

$L_2 \backslash L_1$	a	aa
b	ab	aab
aba	aaba	aaaba

Tabela 1: Tabela konkatenacji języków  $L_1$  i  $L_2$

$|L_1L_2| \leq |L_1| \cdot |L_2|$  bo eps wszystko psuje

$$L_1 = \{a^n : n \geq 0\}, L_2 = \{b^n : n \geq 0\}, L_1L_2 = \{a^n b^m : n, m \geq 0\}$$

### 5.2 Potęga języka

$$L = \{a, ab\}, L^0 = \{\epsilon\}, L^1 = \{a, ab\}, L^2 = L \cdot L$$

Potęgowanie na językach jest dziwne

$$L = \{a^n : n \geq 0\}, L^2 = \{a^n a^m : a, m \geq 0\} = \{a^n : a \geq 0\} = L$$

Potęgowanie języku **nie** zwiększyło mocy

$$L = \{a^n : n > 0\}, L^2 = \{a^n a^m : a, m > 0\} = L \setminus \{a\} = \{a^n : n > 1\}$$

Potęgowanie języku **zmniejszyło** moc

### 5.3 Odbicie języka

$$L^{-1} = \{P^{-1} : P \in L\}$$

### 5.4 Dzielenie słów

$P \in L^n \rightarrow$  można podzielić  $P$  na  $n$  (niekoniecznie różnych) słów

$$L = \{a, ab\}, "aabababab" \in L^n, n = ?$$

Jest to problem wykładniczy, który wymaga stworzenia drzewa różnych możliwości.

## 6 Domknięcie Kleenego

$$L^* = \bigcup_{n \geq 0} L^n$$

$$L^+ = \bigcup_{n \geq 1} L^n$$

$$L_1 = \{a\}, L_1^* = \{a^n : n \geq 0\}, L_1^+ = \{a^n : n > 0\}$$

$$L_2 = \{\epsilon, a\}, L_2^* = \{a^n : n \geq 0\} = L_2^+$$

$L = \{aa, ab, ba, bb\}, L^* = \{P \in \{a, b\}^* : 2||P||\} =$  wszystkie słowa nad alfabetem  $a, b$  o parzystej długości

- $L^+ \subset L^*$
- $\epsilon \in L \rightarrow L^+ = L^*$
- $(L^*)^* = L^*$
- $L_1 \subset L_2 \rightarrow L_1^* \subset L_2^*$

$$L = \{a^n : n > 1\}, L^1 \neq L^2, L^* = L$$

## 7 Automaty

- nieskończona taśma
- rejestry
- w każdym rejestrze symbol z alfabetu  $T$
- głowica, która porusza się od lewej do prawej po rejestrach taśmy, aż do momentu, kiedy napotka pusty rejestr. Głowica zawsze jest w jednym ze stanów z zbioru stanów

### 7.1 Deterministyczne automaty skończone

Automat skończenie stanowy jest uporządkowaną piątką

$$\mathfrak{A} = \langle K, T, \delta, q_0, H \rangle$$

- $K$  zbiór stanów
- $T$  alfabet - symbole z tego alfabetu znajdują się w rejestrach
- $\delta : K \times T \rightarrow K$  funkcja przejścia automatu
- $q_0$  stan początkowy automatu
- $H$  zbiór stanów akceptowalnych/końcowych

### 7.1.1 Funkcja przejść

Zbiory  $K$  i  $T$  są skończone, co oznacza, że funkcję  $\delta$  można przedstawić w formie tabelki. Przykład:

$$K = \{q_0, q_1, q_2\}, T = \{a, b\}, H = \{q_2\}$$

$$\delta : K \times T \rightarrow K$$

$K \backslash T$	a	b
$\rightarrow q_0$	$q_2$	$q_1$
$q_1$	$q_0$	$q_1$
$\underline{q_2}$	$q_1$	$q_2$

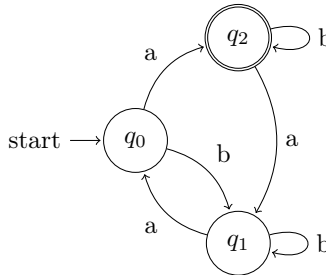


Diagram 1: Tabela konkatencji języków  $L_1$  i  $L_2$  oraz graf przejść automatu

### 7.1.2 Rozszerzona funkcja przejść

$$\hat{\delta} : K \times T^* \rightarrow K$$

- $\hat{\delta}(q, \epsilon) = q$
- $\hat{\delta}(q, Pa) = \delta(\hat{\delta}(q, P), a)$

$$L(\mathfrak{A}) = \{P \in T^* : \hat{\delta}(q_0, P) \in H\}$$

### 7.1.3 Przykład

Narysuj diagram przejścia deterministycznego automatu skończenie stanowego  $\mathfrak{A}$  w którym  $T = \{0, 1\}$ ,  $P \in L(\mathfrak{A})$  wtedy i tylko wtedy gdy w  $P$  występuje na pierwszym od końca miejscu.

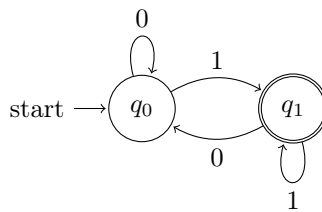


Diagram 2: Diagram przejścia automatu do wykrywania 1 na pierwszym miejscu od końca

Narysuj diagram przejścia deterministycznego automatu skończenie stanowego  $\mathfrak{A}$  w którym  $T = \{0, 1\}$ ,  $P \in L(\mathfrak{A})$  wtedy i tylko wtedy gdy w  $P$  występuje na drugim od końca miejscu 1.

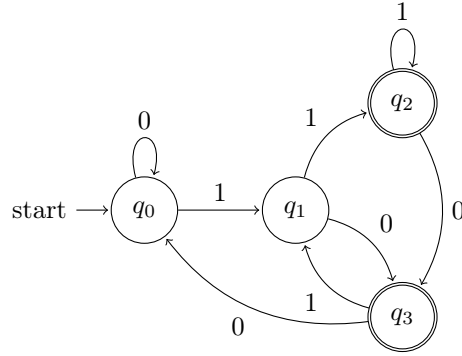


Diagram 3: Diagram przejścia automatu do wykrywania 1 na drugim miejscu od końca

Widać na diagramie 3 wprost zależność że w zależności od miejsca od końca na którym ma być jeden rośnie ilość stanów. Ilość stanów maszyny  $|K|$  do wykrywania 1 na  $n$ -tym miejscu od końca można wyrazić w następujący sposób:  $|K| = 2^n$

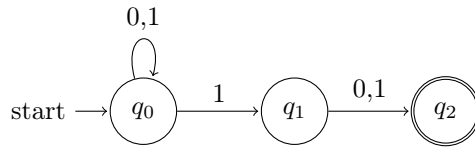
## 7.2 Niedeterministyczne automaty skończone

- zamiast jednego stanu początkowego jest zbiór stanów początkowych
- niedeterministyczna funkcja przejścia, która zwraca zbiór wyjściowych stanów

$$\mathfrak{A} = \langle K, T, \delta, Q_0, H \rangle$$

gdzie oznaczenia są identyczne jak dla deterministycznego automatu z dwoma różnicami:

- $\delta : K \times T \rightarrow a \in K$  funkcja przejścia automatu
- $Q_0$  zbiór stanów początkowych automatu

Diagram 4: Niedeterministyczna wersja automatu  $\mathfrak{A}$  z rysunku 3

Jak widać zamiast 4 stanów potrzeba tylko 3, to dlatego, że dla wersji niedeterministycznej  $|K| = n + 1$

### 7.2.1 Rozszerzona funkcja przejść

$$\hat{\delta}: P(K) \times T^* \rightarrow P(K)$$

- $\hat{\delta}(A, \epsilon) = A$
- $\hat{\delta}(A, Pa) = \bigcup_{q \in \hat{\delta}(A, P)} \delta(q, a)$

$$\hat{\delta}(\{p\}, a) = \delta(p, a)$$

$$L(\mathfrak{A}) = \{P \in T^* : \hat{\delta}(Q_0, P) \cap H \neq \emptyset\}$$

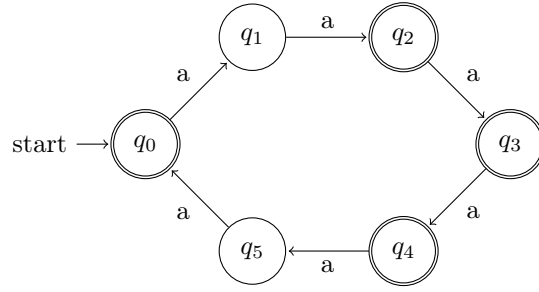


Diagram 5: Diagram przejścia automatu do wykrywania słów o długości podzielnej przez 2 lub 3

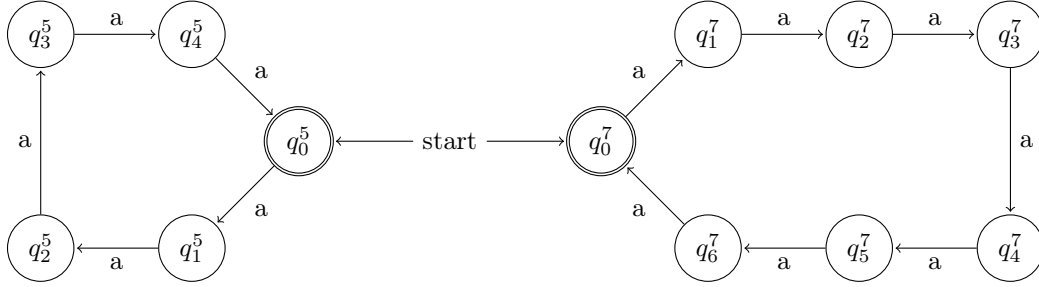


Diagram 6: Diagram przejścia automatu ndet do wykrywania słów o długości podzielnej przez 5 lub 7

### 7.2.2 Twierdzenie Scotta

- każdy **niedeterministyczny** automat skończony można zastąpić równoważnym deterministycznym automatem skończonym

$$\mathfrak{L}_{ndet} \subset \mathfrak{L}_{det}$$

- każdy deterministyczny automat skończony można zastąpić równoważnym **niedeterministycznym** automatem skończonym

$$\mathfrak{L}_{det} \subset \mathfrak{L}_{ndet}$$

- liczba stanów automatu deterministycznego jest wykładnicza w stosunku do liczby stanów automatu niedeterministycznego

$$\mathfrak{L}_{det} = \mathfrak{L}_{ndet}$$

Zatem co nam daje niedeterministyczność? Przede wszystkim prostotę, ale kosztem wykładniczej złożoności.

Narysuj diagram przejścia deterministycznego automatu skończenie stanowego  $\mathfrak{A}$  w którym  $T = \{a\}$ ,  $P \in L(\mathfrak{A})$  wtedy i tylko wtedy gdy  $(2||P|) \vee (3||P|)$ .

Jak widzimy na diagramie 5 przyjmuje postać cyklu o okresie 6, ponieważ  $NWW(3, 2) = 6$ . Problem z diagramami deterministycznym się pojawia dla wyższych liczb, np.: 7 i 5, wtedy  $NWW(7, 5) = 35$ . Zatem narysujmy diagram niedeterministyczny 6.

Jako, że automaty niedeterministyczne pozwalają na kilka stanów początkowych, to tworzymy diagram niespójny, który w zależności od tego czy  $|P|$  jest podzielne przez 5 czy 7 przechodzi do odpowiedniego pod-automatu. Najłatwiej to można sobie wyobrazić jako dwa równoległe automaty z alternatywą na koniec.

### 7.2.3 Przekształcenie ndet $\rightarrow$ det

$$\mathfrak{A} = \langle K, T, \delta, Q_0, H \rangle$$

$$\mathfrak{A}' = \langle K', T', \delta', q'_0, H' \rangle$$

$$L(\mathfrak{A}) = L(\mathfrak{A}')$$



- $T' = T$  bo nie ma sensu zmieniać taśmy
- $K' = P(K)$  **Wykładniczy wzrost liczby stanów**  
w przekształceniu będziemy używać systemu etykiet (konstrukcja potęgowa) aby zamieniać zbiory stanów na pojedyncze stany

$$\{q_0, q_1\} = q^{01}$$

$$P(K) = \{q^\emptyset, q^1, q^0, q^{10}, \dots\}$$

- $q'_0 = Q_0$  - stan odpowiadający zbiorowi stanów początkowych
- $H' = \{A \in K' : A \cap H \neq \emptyset\}$
- $\delta'(A, a) = \bigcup_{q \in A} \delta(q, a)$

Dla automatu 4 zbudujemy równoważny automat deterministyczny. Korzystając z powyższych zasad otrzymujemy:

- $K' = P(K) = \{\emptyset, \{q_0\}, \dots, \{q_1, q_2\}, \{q_0, q_1, q_2\}\} = \{q^\emptyset, q^0, \dots, q^{12}, q^{012}\}$
- $H' = \{q^2, q^{12}, q^{02}, q^{012}\}$
- $q'_0 = \{q_0\} = q^0$

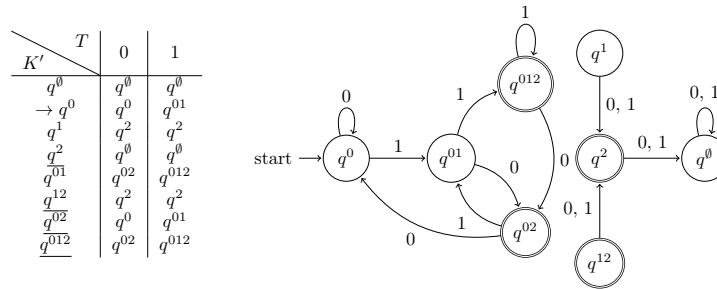


Diagram 7: Zdeterminizowany automat z rysunku 4 i jego tabela przejść

Jak widać na diagramie 7 liczba stanów wzrosła z 3 do 8, co jest zgodne z przewidywaniami. Jednocześnie widać, że diagram 3 zawiera się w diagramie 7, co niekoniecznie oznacza, że są sobie równoważne, lecz jako, że stany dodatkowe są nieosiągalne to te dwa automaty są równoważne. **Nie zawsze równoważność automatów będzie tak oczywista.**

### 7.3 Automaty z przejściem

Co jeśli moglibyśmy zmienić stan ale nie ruszyć głowicy? Wtedy mamy do czynienia z automatem z przejściem.

$$\epsilon \in T, \epsilon = \text{nie ruszaj głowicy automatu}$$

$$\delta : K \times (T \cup \{\epsilon\}) \rightarrow P(K)$$

**Każdy automat z przejściem jest niedeterministyczny**

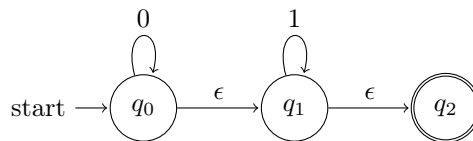


Diagram 8: Automat z przejściem

Automat przedstawiony na rysunku 8 akceptuje języki o następującej postaci  $L(\mathcal{A}) = \{0^n 1^m 2^k : n, m, k \geq 0\}$ . **Nie istnieją różne epsilony:**  $00\epsilon 1\epsilon 22 = 00122$

### 7.3.1 Domknięcie stanu

$E(q)$  = zbiór stanów osiągalnych z  $q$  przez dowolną liczbę epsilonów

1.  $q \in E(q)$
2.  $r \in E(q) \wedge p \in \delta(r, \epsilon) \rightarrow p \in E(q)$

### 7.3.2 Domknięcie zbioru stanów

$$E(A) = \bigcup_{q \in A} E(q)$$

### 7.3.3 Rozszerzona funkcja przejść

$$\hat{\delta}: P(K) \times T^* \rightarrow P(K)$$

- $\hat{\delta}(A, \epsilon) = E(A)$
- $\hat{\delta}(A, Pa) = \bigcup_{q \in \hat{\delta}(A, P)} E(\delta(q, a))$

### 7.3.4 Przekształcenie $\epsilon \rightarrow \text{ndet}$

$$\mathfrak{A}' = \langle K', T', \delta', Q'_0, H' \rangle$$

$$T' = T, K' = K, H' = H, Q'_0 = E(Q_0), \delta'(A, a) = E(\delta(q, a))$$

$$\mathfrak{L}_{\text{ndet}} = \mathfrak{L}_\epsilon = \mathfrak{L}_{\text{det}}$$

$\delta_\epsilon$	a	b	$\epsilon$	$\Rightarrow$	$\delta_{\text{ndet}}$	a	b
$\rightarrow q_0$	$\emptyset$	$\{q_1\}$	$\{q_2\}$		$\rightarrow q_0$	$E(\emptyset) = \emptyset$	$\{q_1\}$
$q_1$	$\{q_1, q_2\}$	$\{q_2\}$	$\emptyset$		$q_1$	$\{q_1, q_2\}$	$\{q_2\}$
$\underline{q_2}$	$\{q_0\}$	$\emptyset$	$\emptyset$		$\rightarrow \underline{q_2}$	$E(\{q_0\}) = \{q_0, q_2\}$	$\emptyset$

Diagram 9: Przekształcenie automatu z przejściami na niedeterministyczny

## 8 Wyrażenia regularne

$\text{Reg}(V)$  = zbiór wyrażeń regularnych nad alfabetem  $V$

- $\epsilon \in \text{Reg}(V)$
- $a \in \text{Reg}(V)$
- $a \in V \rightarrow a \in \text{Reg}(V)$
- $u, v \in \text{Reg}(V) \rightarrow (u + v), (u \cdot v), (u^*) \in \text{Reg}(V)$

### 8.1 Operacje

W kolejności od najwyższego priorytetu do najniższego

1.  $P \in \text{Reg}(V) \rightarrow L(P) \neq \emptyset$
2.  $L(u^*) = (L(u))^*$
3.  $L(uv) = L(u) \cdot L(v)$
4.  $L(u + v) = L(u) \cup L(v)$
5.  $L(u) = \{u\}$

## 8.2 Przykłady

$$L(ba^*) = \{ba^n : n \geq 0\}$$

$$L(ba^*) = L(b)L(a^*) = \{b\} \cdot (L(a))^* = \{b\} \cdot \{a\}^* = \{ba^n : n \geq 0\}$$

$L((a+b)^*ab(a+b)^*)$  - wszystkie słowa nad alfabetem  $\{a, b\}$  zaczynające się od  $a$  i kończące się  $b$

## 8.3 Tw. Kleenego

$$\forall_{v \in \text{Reg}(V)} L(v) \subset \mathfrak{L}_{det}$$

Każdy język generowany przez wyrażenie regularne jest językiem akceptowanym przez automat skończony. Dowód opiera się na konstrukcji automatu  $\epsilon$  odpowiadającego operatorowi wyrażenia regularnego.

### 8.3.1 $w = u + v$

$$L(u + v) = L(u) \cup L(v)$$

$$L(u) = L(\mathfrak{A}_u), L(v) = L(\mathfrak{A}_v)$$

$$L(\mathfrak{A}) = L(\mathfrak{A}_u) \cup L(\mathfrak{A}_v)$$

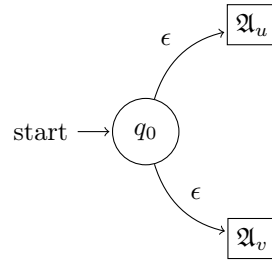


Diagram 10: Konstrukcja automatu dla wyrażenia regularnego  $u + v$

### 8.3.2 $w = uv$

$$L(uv) = L(u) \cdot L(v)$$

$$L(u) = L(\mathfrak{A}_u), L(v) = L(\mathfrak{A}_v)$$

$$L(\mathfrak{A}) = L(\mathfrak{A}_u) \cdot L(\mathfrak{A}_v)$$

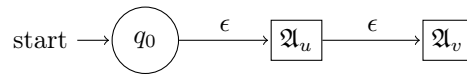


Diagram 11: Konstrukcja automatu dla wyrażenia regularnego  $uv$

### 8.3.3 $w = u^*$

$$L(u^*) = (L(u))^*$$

$$L(u) = L(\mathfrak{A}_u)$$

$$L(\mathfrak{A}) = (L(\mathfrak{A}_u))^*$$

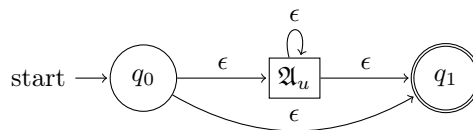


Diagram 12: Konstrukcja automatu dla wyrażenia regularnego  $u^*$

## 9 Klasy języków

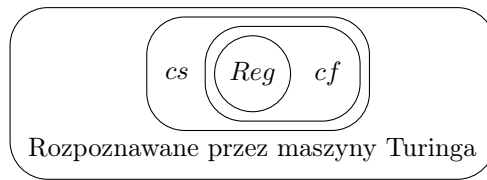
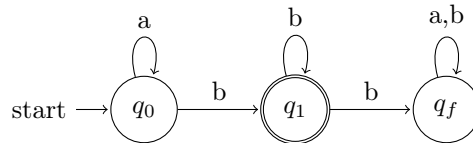
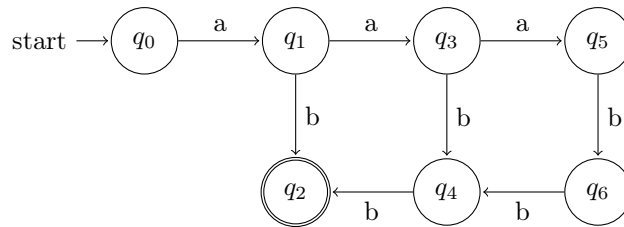


Diagram 13: Nadzbiory języków regularnych

### 9.1 Języki regularne

Diagram 14: Konstrukcja automatu dla automatu  $a^*b^*$ 

Język regularny to język akceptowany przez wyrażenie regularne, czyli język akceptowany przez automat skończony.

Diagram 15: Konstrukcja automatu akceptującego język  $L = \{a^n b^n : 1 \leq n \leq 3\}$  bez śmietnika

**Czy można zapisać automat deterministyczny (lub nie) skończenie stanowy, akceptujący język  $L = \{a^n b^n : n \geq 1\}$  ?**

Nie da się, ponieważ wymagałoby to nieskończonej ilości stanów. Zatem język  $L$  nie jest językiem regularnym.

#### 9.1.1 Lemat o pompowaniu dla języków regularnych

Służy do dowodzenia, że język **nie** jest regularny.

Jeżeli  $L = L(\mathfrak{A})$

to: istnieje  $k \geq 0$ , taki, że każde słowo  $P \in L$  o długości  $|P| \geq k$  można zapisać jako  $P = XYZ$ , spełniające warunki:

- $Y \neq \epsilon$
- $|XY| \leq k$
- $\forall_{i \geq 0} XY^i Z \in L$

Parafrazuąc: jeśli język jest regularny, to nie ma w nim słowa, którego nie mógłbyś podzielić na trzy części, takie, że środkowa część jest powtarzalna.

Lemat o pompowaniu jest warunkiem koniecznym ale nie wystarczającym. To oznacza, że wszystkie języki regularne spełniają warunek lematu o pompowaniu, ale nie wszystkie języki spełniający warunek lematu o pompowaniu są regularne. Warunkiem dostatecznym jest zdefiniowanie automatu skończenie stanowego akceptującego język.

#### Przykład zgodny

Dlaczego język  $L = \{a^n b^m : n, m \geq 1\}$  jest regularny?

- $k > 2, P \in L$
- $|P| \geq k \rightarrow n + m \geq k$
- $P = a^{n-x}a^xb^m$  czyli  $P = XYZ$  gdzie  $X = a^{n-x}, Y = a^x, Z = b^m$
- $n - x + x < k \leftrightarrow |XY| \leq k$
- dla  $i \geq 0$   $XY^iZ = a^{n-x}a^{xi}b^m = a^{n+x(i-1)}b^m$ , co jak widzimy jest w języku  $L$

**Przykład niezgodny**

Dlaczego język  $L = \{a^n b^n : n \geq 1\}$  nie jest regularny?

- $k > 2, P \in L$
- $|P| \geq k \rightarrow 2n \geq k$
- $|XY| \leq k \rightarrow XY = a^n \vee XY = b^n$
- dla  $i = 2$   $XY^2Z = a^n b^n b^n \notin L$

Zatem muszą istnieć języki nieregularne

**9.1.2 Przechodność regularności**

Z tw. Kleenego:

$$L_1, L_2 \in \mathfrak{L}_{reg} \rightarrow L_1 \cup L_2, L_1 \cdot L_2, L_1^*, L_2^* \in \mathfrak{L}_{reg}$$

$$L_1 \cup L_2 = L(v + w) = L(v) \cup L(w) = L_1 \cup L_2$$

$$L_1 \cap L_2 = \overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$$

$$L_3 \subsetneq L_2 \subsetneq L_1 \subsetneq L_0$$

**9.2 Bezkontekstowe****9.2.1 Część wspólna języków bezkontekstowych**

$$\{a^n b^n c^k : n \geq 1\} \cap \{a^n b^k c^k : n \geq 1\} = \{a^n b^n c^n : n \geq 1\}$$

**10 Gramatyki**

Gramatyki służą do generowania języków. Działanie gramatyk wyraża się przy pomocy reguły przepisującej.

$(P, Q)$  – słowa

Jeżeli  $P \rightarrow Q$  oraz  $U = P_1 P P_2$  to  $U \Rightarrow P_1 Q P_2$

Jeżeli reguła przepisująca jest wykonywana wielokrotnie to używamy oznaczenia  $\xRightarrow{*}$ .

$$G = \langle V_N, V_T, S, F \rangle$$

gdzie:

- $V_N$  - zbiór nieterminali (symboli zastępowanych)
- $V_T$  - zbiór terminali (symboli niezastępowanych)
- $S$  - symbol początkowy
- $F$  - zbiór reguł przepisujących

$$V_N \cap V_T = \emptyset$$

## 10.1 Wyprowadzanie słowa w jednym kroku

Dla gramatyki  $G$  mówimy, że  $W$  jest wyprowadzane w jednym kroku z  $U$  (oznaczane  $U \Rightarrow W$ ) jeżeli istnieje reguła  $P \rightarrow Q \in F$  taka, że  $U = P_1 P Q_2$  oraz  $W = P_1 Q Q_2$ .

## 10.2 Wyprowadzanie słowa w wielu krokach

Dla gramatyki  $G$  mówimy, że  $W$  jest wyprowadzane w wielu krokach z  $U$  (oznaczane  $U \Rightarrow^* W$ ) wtedy gdy  $U = W$ , lub gdy istnieją słowa  $R_1 \dots R_n : n > 1$  gdzie  $R_1 = U$  a  $R_n = W$ .

## 10.3 Język generowany przez gramatykę

$$L(G) = \{W \in V_T^* : S \Rightarrow^* W\}$$

### 10.3.1 Przykład prosty

$$F = \{S \rightarrow aaSA, S \rightarrow \epsilon, A \rightarrow bA, A \rightarrow b\}$$

$$S \Rightarrow^* a^{2n} S A^n \Rightarrow a^{2n} A^n \Rightarrow^* a^{2n} b^m$$

$$L(G) = \{a^{2n} b^m : n \geq 0, m \geq 0\} \cup \{\epsilon\}$$

### 10.3.2 Przykład złożony

$$F = \{S \rightarrow abScd, S \rightarrow A, A \rightarrow dcAba, A \rightarrow \epsilon\}$$

$$S \Rightarrow^* (ab)^n S (cd)^n \Rightarrow (ab)^n A (cd)^n \Rightarrow^* (ab)^n (dc)^m A (ba)^m (cd)^n \Rightarrow (ab)^n (dcba)^m (cd)^n$$

$$L(G) = \{(ab)^n (dcba)^m (cd)^n : n \geq 1, m \geq 0\} \cup \{\epsilon\}$$

### 10.3.3 Przykład prosty

$$L(G) = \{a^n : n \geq 0\} = L(a^*)$$

$$G = \langle \{S\}, \{a\}, S, \{S \rightarrow aS, S \rightarrow \epsilon\} \rangle$$

### 10.3.4 Przykład złożony

$$L(G) = \{a^n b^n : n \geq 1\} \notin \mathfrak{L}_{reg}$$

$$G = \langle \{S\}, \{a, b\}, S, \{S \rightarrow aSb, S \rightarrow ab\} \rangle$$

## 10.4 Rodzaje gramatyk

- Typu 0 (ogólne)
- Typu 1 (kontekstowe)
- Typu 2 (bezkontekstowe)
- Typu 3 (regularne)

$$G_2 \subset G_1 \subset G_0, G_3 \subset G_0$$

$$L(G_n) = \mathfrak{L}_n = L(\mathfrak{A}_n)$$

## 10.5 Gramatyki typu 3

$$F = \{A \rightarrow aB, A \rightarrow a, A \rightarrow \epsilon : A, B \in V_N, a \in V_T\}$$

### 10.5.1 Gramatyki Normalne typu 3

$$F = \{A \rightarrow aB, A \rightarrow \epsilon : A \in V_N, a \in V_T\}$$

Liczba kroków generacji =  $|P|$ . Każda gramatyka regularna może być zapisana w postaci gramatyki normalnej typu 3.

### 10.5.2 Gramatyki liniowe

Gramatyki regularne inaczej nazywa się gramatykami liniowymi prawostronnymi. Gramatyki liniowe prawostronne to gramatyki, w których  $F = \{A \rightarrow Pb\}$ . Gramatyki liniowe lewostronne to gramatyki, w których  $F = \{A \rightarrow bP\}$ . **Gramatyki liniowe lewostronne są równoważne gramatykom liniowym prawostronnym.** Gramatyki liniowe to gramatyki kontekstowe, w których  $F = \{A \rightarrow P_1B_2\}$

## 10.6 Gramatyki bezkontekstowe

$$F = \{A \rightarrow P : A \in V_N, P \in \{V_N \cup V_T\}^*\}$$

### 10.6.1 Problem należenia słowa pustego

Dla każdej gramatyki bezkontekstowej  $G$  można efektywnie skonstruować równoważną jej gramatykę bezkontekstową  $G'$ , w której nie ma reguł przepisujących symbol początkowy w słowo puste, za wyjątkiem sytuacji jeśli  $\epsilon \in L(G)$ . Wówczas jedyną regułą w  $F'$  zawierającą  $\epsilon$  będzie  $S' \rightarrow \epsilon$  i  $S'$  nie pojawia się po prawej stronie żadnej reguły.

Problem należenia słowa pustego jest rozstrzygalny dla gramatyk bezkontekstowych. Algorytm wygląda następująco:

1. Przekształć gramatykę  $G$  na równoważną jej gramatykę  $G'$ , w której nie ma reguł przepisujących symbol początkowy w słowo puste.
2. Sprawdź, czy  $S' \rightarrow \epsilon \in F'$ .

### 10.6.2 Gramatyki normalne bezkontekstowe

Gramatyka bezkontekstowa jest normalna, jeżeli każda reguła przepisująca ma postać  $A \rightarrow a$  lub  $A \rightarrow BC$ , **efektywnie tworząc drzewo binarne.**

Dla każdej  $\epsilon$ -wolnej gramatyki bezkontekstowej można efektywnie skonstruować równoważną jej gramatykę bezkontekstową normalną.

**Zamiana CF  $\rightarrow$  CF norm**

- Pozbądź się terminalów z reguł innych niż te o kształcie  $A \rightarrow a$  kosztem nowych nieterminali.

$$\{A \rightarrow abXc\} \Rightarrow \{A \rightarrow X_aX_bXX_c, X_a \rightarrow a, X_b \rightarrow b, X_c \rightarrow c\}$$

- Zamień reguły o kształcie  $A \rightarrow a_1a_2 \dots a_nB$  na reguły o kształcie  $A \rightarrow a_1X_1, X_1 \rightarrow a_2X_2, \dots, X_{n-1} \rightarrow a_nB$ .

$$\{A \rightarrow X_1X_2X_3X_4\} \Rightarrow \{A \rightarrow X_1Z_1, Z_1 \rightarrow X_2Z_2, Z_2 \rightarrow X_3X_4\}$$

### 10.6.3 Problem słowa

Problem słowa jest rozstrzygalny dla gramatyk bezkontekstowych. Algorytm siłowy ma złożoność  $O(\overline{\overline{F}}^{|P|})$  gdzie  $|P|$  to długość słowa. Z kolei algorytm "CYK" ma złożoność  $O(|P|^3)$ .

### 10.6.4 Lemat o pompowaniu dla gramatyk bezkontekstowych

Istnieje liczba naturalna  $p$  zależna od  $L$ , taka, że dla każdego słowa  $P \in L$  o długości  $|P| \geq p$  istnieje podział  $P = UXWYZ$  spełniający warunki:

- $XY \neq \epsilon$
- $|XWY| \leq p$
- $\forall_{i \geq 0} UX^iWY^iZ \in L$

Dla normalnych gramatyk  $n = \overline{\overline{V_n}}$ ,  $p = 2^n$

### 10.7 Gramatyki kontekstowe

$$F = \{Q_1AQ_2 \rightarrow Q_1PQ_2 : Q_1, Q_2, A, P \in \{V_N \cup V_T\}^* \setminus S\} \cup \{S \rightarrow \epsilon\}$$

Reguły w  $F$  nie mogą pozwolić na zmniejszenie się ciągu.

### 10.8 Gramatyki ogólne

Bez ograniczeń na reguły

### 10.9 Przekształcenie automatu na gramatykę

Dla  $\mathfrak{A} = \langle K, T, \delta, q_0, H \rangle$ ,  $\mathfrak{A} \in \mathfrak{A}_{det}$

- $V_T = T$
- $V_N = K \cup \{S\}$
- $\delta(q_0, a) = p$  to  $p \rightarrow a \in F$
- $\delta(q, a) = p$  to  $p \rightarrow qa \in F$
- $q \in H$  to  $S \rightarrow p \in F$
- $q_0 \in H$  to  $S \rightarrow \epsilon \in F$

### 10.10 Przekształcenie gramatyki na automat

Dla  $G = \langle V_N, V_T, S, F \rangle$

- $K = V_N$
- $T = V_T$
- $Q_0 = \{S\}$
- $H = \{A \in K : A \rightarrow \epsilon \in F\}$
- $A \rightarrow aB \in F$  to  $B \in \delta(A, a)$



## 11 Algorytmy

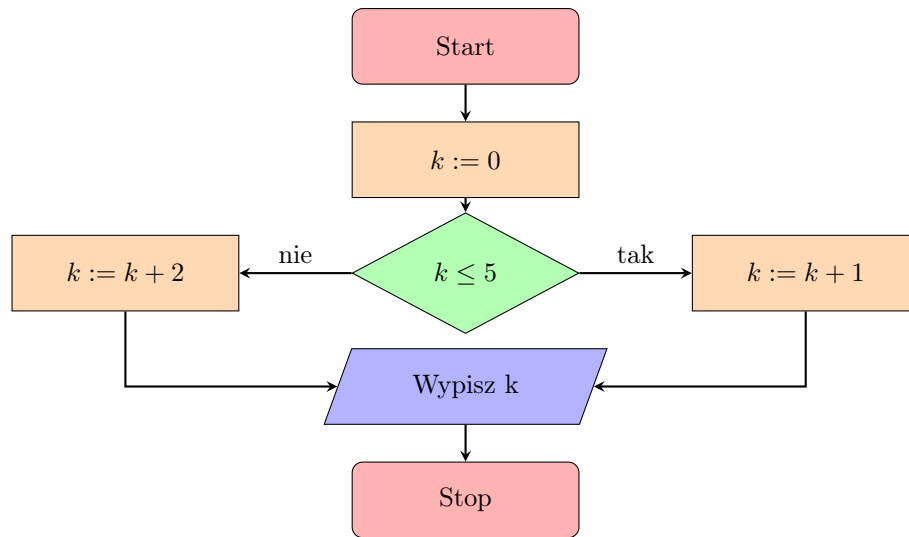


Diagram 16: Schemat blokowy przykładowego algorytmu

Algorytm to zbiór procedur, które dla danego wejścia dają określony wynik. My zakładamy, że algorytm zwraca TAK lub NIE.

### 11.1 Problem

Problem to zbiór instancji, oraz zbiór instancji pozytywnych problemu.

$$\mathcal{P} = \langle \mathbb{I}, \mathbb{P} \rangle$$

$$\mathbb{P} \subset \mathbb{I}$$

Warto zauważyć zatem, że  $\mathbb{I}$  działa jak alfabet, a  $\mathbb{P}$  jak język nad tym alfabetem. Algorytm dla danego problemu określa czy dana instancja należy do zbioru pozytywnego (języka), w podobny sposób jak automat określa co jest w języku. Zatem algorytm działa trochę jak funkcja  $\mathcal{A} : \mathbb{I} \rightarrow \{\top, \perp\}$ .

#### 11.1.1 Problem rozstrzygalny

Problem rozstrzygalny to taki problem, dla którego istnieje algorytm, który dla każdej instancji zwraca TAK lub NIE.

#### 11.1.2 Problem pustości

$$\mathbb{I} = \text{wszystkie automaty} = \mathbb{A}$$

$$\mathbb{P} = \{\mathfrak{A} \in \mathbb{A} : L(\mathfrak{A}) = \emptyset\}$$

Jest to problem **rozstrzygalny** dla języków regularnych.

#### 11.1.3 Problem skończoności

$$\mathbb{I} = \mathbb{A}$$

$$\mathbb{P} = \{\mathfrak{A} \in \mathbb{A} : |L(\mathfrak{A})| < \infty\}$$

Jest to problem **rozstrzygalny** dla języków regularnych.

**11.1.4 Problem nieskończoności**

$$\mathbb{I} = \mathbb{A}$$

$$\mathbb{P} = \{\mathfrak{A} \in \mathbb{A} : |L(\mathfrak{A})| = \infty\}$$

Jest to problem **rozstrzygalny** dla języków regularnych, ponieważ jest to odwrotne pytanie do problemu skończoności.

**11.1.5 Problem równości**

$$\mathbb{I} = \mathbb{A} \times \mathbb{A}$$

$$\mathbb{P} = \{(\mathfrak{A}_1, \mathfrak{A}_2) \in \mathbb{A} \times \mathbb{A} : L(\mathfrak{A}_1) = L(\mathfrak{A}_2)\}$$

Jest to problem **rozstrzygalny** dla języków regularnych.

- Zbuduj  $\mathfrak{B}$ , takie, że  $L(\mathfrak{B}) = (L(\mathfrak{A}_1) \cap \overline{L(\mathfrak{A}_2)}) \cup (L(\mathfrak{A}_2) \cap \overline{L(\mathfrak{A}_1)})$  (XOR mnogościowe)
- Zastosuj problem pustości na  $\mathfrak{B}$