
AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

# Wyrażenia regularne, automaty skończone

## Języki formalne i automaty

**Dr inż. Janusz Majewski**
**Katedra Informatyki**

# Definiowanie języków formalnych

- Wyliczenie wszystkich poprawnych napisów w danym języku (bezużyteczne, gdy język zawiera nieskończenie wiele napisów).
- Podanie zasad **budowy** wszystkich poprawnych napisów w danym języku (**gramatyka języka**).
- Podanie algorytmu orzekającego, czy dany napis jest poprawnym napisem w danym języku (**automat abstrakcyjny**).
- Podanie „ogólnego wzoru” określającego postać wszystkich poprawnych napisów w danym języku (**wyrażenie regularne** wykorzystywane do definiowania szczególnie prostych języków zwanych językami regularnymi).

# Zbiory (języki) regularne

Niech $\Sigma$ będzie alfabetem.

*Zbiór (język) regularny* nad alfabetem $\Sigma$ definiujemy następująco:

*1)* $\varnothing$ – zbiór pusty jest zbiorem regularnym,

*2)* $\{\varepsilon\}$ – zbiór zawierający łańcuch pusty jest zbiorem regularnym,

*3)* $\{a\}$ – $(\forall a \in \Sigma)$ zbiór zawierający łańcuch złożony z pojedynczego symbolu alfabetu jest zbiorem regularnym,

*4)* jeśli $P$ i $Q$ są zbiorami regularnymi nad $\Sigma$ to zbiorami regularnymi są także:

*a)* $P \cup Q$ – suma teoriomnogościowa zbiorów $P$ i $Q$,

*b)* $PQ$ – złożenie (konkatenacja) zbiorów $P$ i $Q$,

*c)* $P^*$ - domknięcie Kleene'a zbioru $P$.

*5)* nic innego poza tym, co wynika z punktów (1) – (4), nie jest zbiorem regularnym.

# Wyrażenia regularne (1)

Wyrażenia regularne służą do uproszczonego oznaczania zbiorów regularnych. Niech $\Sigma$ będzie alfabetem. Wyrażenia regularne nad alfabetem $\Sigma$ definiujemy następująco:

***1)*** $\varnothing$ – jest wyrażeniem regularnym oznaczającym zbiór pusty $\varnothing$ będący zbiorem regularnym,

***2)*** $\varepsilon$ – jest wyrażeniem regularnym oznaczającym zbiór zawierający łańcuch pusty $\{\varepsilon\}$ będący zbiorem regularnym,

***3)*** ***a*** – $(\forall a \in \Sigma)$ jest wyrażeniem regularnym oznaczającym zbiór zawierający łańcuch złożony z pojedynczego symbolu alfabetu będący zbiorem regularnym,

4) jeśli ***p*** i ***q*** są wyrażeniami regularnymi oznaczającymi odpowiednio zbiory regularne $P$ i $Q$ nad $\Sigma$ to wyrażeniami regularnymi są także:

   ***a)*** ***p|q*** – wyrażenie regularne oznaczające $P \cup Q$ – sumę teoriomnogościową zbiorów $P$ i $Q$ będącą zbiorem regularnym,

   ***b)*** ***pq*** – wyrażenie regularne oznaczające $PQ$ – złożenie (konkatenację) zbiorów $P$ i $Q$ będące zbiorem regularnym,

   ***c)*** ***p**** - wyrażenie regularne oznaczające $P^*$ - domknięcie Kleene'ego zbioru P będące zbiorem regularnym.

5) nic innego poza tym, co wynika z punktów (1) – (4), nie jest wyrażeniem regularnym.

# Przykłady

Przykład:

Niech Σ = {a, b}. Zbiorem regularnym nad $\Sigma$ jest np. zbiór:

$\{\varepsilon, a, ab, abb, abbb, abbbb, ...\} = \{\varepsilon\} \cup \{a\}\{b\}^*$

Ten zbiór regularny zapisujemy w formie wyrażenia regularnego jako:

**ε|ab***.

Przykład:

Wyrażenie regularne:

**(0|1)*011**

odpowiada zbiorowi regularnemu:

$(\{0\} \cup \{1\})^* \{0\}\{1\}\{1\} = \{0, 1\}^* \{011\}$

będącemu dowolnym ciągiem zer i jedynek zakończonym sekwencją: 011.

# Wyrażenia regularne (2)

- Dwa wyrażenia **p i q** regularne są równe (równoważne), gdy odpowiadające im zbiory regularne *P* i *Q* są równe (identyczne).
- W zapisie wyrażeń regularnych można stosować nawiasy.
- Zapisując wyrażenia regularne stosujemy następujące priorytety operatorów:
  - **( )** - najwyższy,
  - *****
  - **·** - (konkatenacja)
  - **|** - najniższy.

# Tożsamości

Niech **p**, **q** i **r** będą dowolnymi wyrażeniami regularnymi. Prawdziwe są następujące zależności i tożsamości:

**p|q = q|p**
**p|(q|r) = (p|q)|r**
**p(qr) = (pq)r**
**pq|pr = p(q|r)**
**pq|rq = (p|r)q**
**εp = pε = p**
**∅p = p∅ = ∅**
**ε* = ε**
**∅* = ε**

**p* = p|p* = (p|ε)***
**(p*)* = p** = p***
**p|p = p**
**p|∅ = p**
**ε|p* = p***
**ε|pp* = p***
**ε|p*p = p***
**pqq*|pq* = pq***
**(p|q)* = (p*|q*)* = (p*q*)***

# Przykłady (1)

Przykład:

**abb*|ab* = ab***

bo:

**abb*|ab*** = {ab}{b}* ∪ {a}{b}* = {ab, abb, ...} ∪ {a, ab, abb, ...} =
= {a, ab, abb, ...} = {a}{b}*

Przykład:

**(ab|a)*a = a(ba|a)***

bo:

**(ab|a)*** - to łańcuchy zbudowane z liter a i b, rozpoczynające się literą a, w których żadne dwie litery b, o ile w ogóle występują, nie występują obok siebie,

**(ba|a)*** - to łańcuchy zbudowane z liter a i b, kończące się literą a, w których żadne dwie litery b, o ile w ogóle występują, nie występują obok siebie,

**(ab|a)*a** oraz **a(ba|a)*** - to łańcuchy zbudowane z liter a i b, rozpoczynające się i kończące się literą a, w których żadne dwie litery b, o ile w ogóle wystepują, nie występują obok siebie.

# Przykłady (2)

Przykład:

**(a|b)* ≠ a*|b***

bo:

**(a|b)*** = {a, b}* = {ε, a, b, aa, ab, ba, bb, aaa, aab, ...}

**a*|b*** = {a}* ∪ {b}* = {ε, a, aa, aaa, ...} ∪ {ε, b, bb, bbb, ...} =
=={ε, a, b, aa, bb, aaa, bbb, ...}

Przykład:

**b(ab|b)* ≠ aa*b(aa*b)***

bo:

**b(ab|b)*** - to łańcuchy zbudowane z liter a i b, rozpoczynające się i kończące się literą b, w których żadne dwie litery a, nie występują obok siebie,

**aa*b(aa*b)*** - to łańcuchy zbudowane z liter a i b, rozpoczynające się literą a, kończące się literą b, w których żadne dwie litery b nie występują obok siebie.

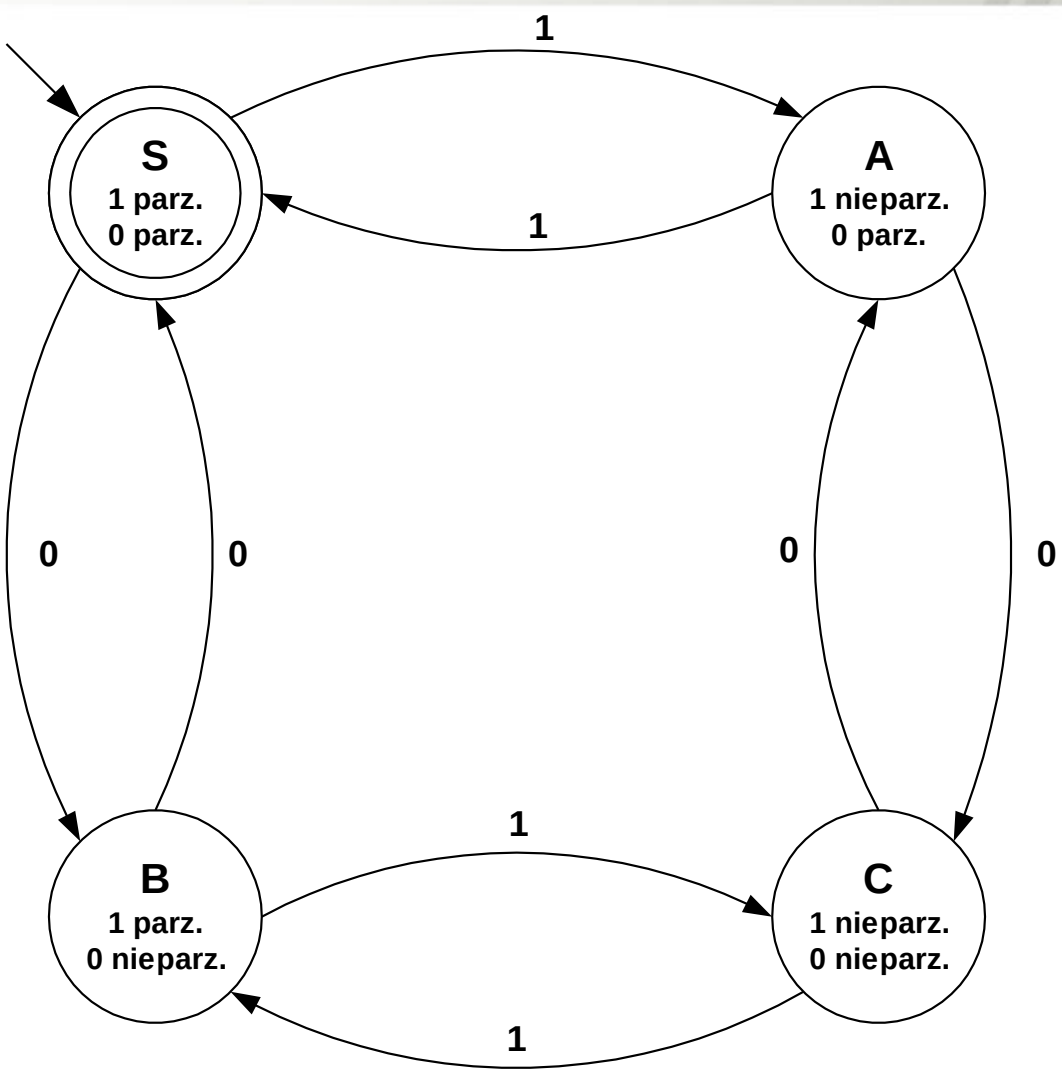
# Automat skończony – przykład

Rozważamy język nad alfabetem binarnym $\Sigma = \{0, 1\}$ składający się z łańcuchów zero-jedynkowych o tej własności, że liczba zer w każdym łańcuchu jest parzysta i liczba jedynek w każdym łańcuchu też jest parzysta. Wszystkie łańcuchy binarne możemy podzielić na cztery grupy:

- S – łańcuchy z parzystą liczbą jedynek i parzystą liczbą zer,
- A – łańcuchy z parzystą liczbą jedynek i nieparzystą liczbą zer,
- B – łańcuchy z nieparzystą liczbą jedynek i parzystą liczbą zer,
- C – łańcuchy z nieparzystą liczbą jedynek i nieparzystą liczbą zer.

Analizujemy łańcuch zero-jedynkowy symbol po symbolu od lewej strony. Przed rozpoczęciem analizy jesteśmy w grupie S (łańcuch pusty zawiera zero jedynek i tyleż zer, więc liczba jedynek i liczba zer w tym łańcuchu są parzyste). Jeśli pierwszym symbolem jest jedynka – przechodzimy do grupy A (wtedy liczba jedynek jest nieparzysta, a liczba zer jest dalej parzysta), zaś jeśli pierwszym symbolem jest zero – przechodzimy do grupy B (wtedy liczba zer jest nieparzysta, a liczba jedynek jest dalej parzysta). Dalej analizujemy podobnie kolejne symbole łańcucha. Np. jeśli jesteśmy w grupie A i przeczytamy zero – przechodzimy do grupy C, w której zarówno liczba jedynek, jak i zer są nieparzyste.

# Graf automatu



# Automat skończony – przykład c.d.

Wreszcie, gdy przeczytaliśmy cały łańcuch, sprawdzamy, czy zatrzymaliśmy się w grupie S. Jeśli tak – badany łańcuch spełnia nałożony nań warunek parzystej liczby jedynek i parzystej liczby zer.

Opisana procedura i zamieszczony wcześniej rysunek grafu ilustrują właściwie proces odpowiadania na pytanie: czy dany łańcuch jest słowem należącym do danego języka.

Procedurę odpowiadania na pytanie o przynależność badanego łańcucha do danego języka nazywamy automatem (w tym przypadku jest to tzw. automat skończony).

# Automat skończony – przykład c.d.

Jest to pewien (w naszym przypadku deterministyczny) algorytm postępowania, polegający na czytaniu badanego łańcucha symbol po symbolu i przechodzenia od jednego stanu do drugiego. Stany reprezentowane są przez kółka (węzły grafu), zaś przejścia pomiędzy stanami, to skierowane krawędzie, opisane (etykietowane) odpowiednimi symbolami alfabetu, z którego pochodzą symbole łańcucha. Jeden ze stanów jest wyróżniony jako stan początkowy (na rysunku – jest to stan oznaczony krótką strzałką dochodząc do niego z zewnątrz). Od tego stanu zawsze rozpoczynamy „wędrówkę" po grafie. Niektóre stany są traktowane jako stany końcowe – akceptujące (są one zaznaczone kółkami rysowanymi podwójną linią). Jeśli w trakcie naszej „wędrówki" po grafie zatrzymamy się w takim stanie, przeczytawszy wejście do końca, to akceptujemy badany łańcuch, jeśli zatrzymamy się w stanie nie będącym stanem końcowym – nie akceptujemy analizowanego łańcucha. Zatrzymanie się w naszym przypadku może być tylko spowodowane przeczytaniem badanego słowa do końca i stwierdzeniem, że już nic nie pozostało do przeczytania.

Opisany powyżej algorytm nosi nazwę automatu skończonego (a dokładnie deterministycznego i zupełnego automatu skończonego).

# Definicja automatu skończonego

Automatem skończonym nazywamy piątkę:

$A = < \Sigma, Q, F, q_0, \delta >$,

gdzie:

$\Sigma$ – zbiór symboli terminalnych (alfabet wejściowy)

$Q$ – skończony zbiór stanów

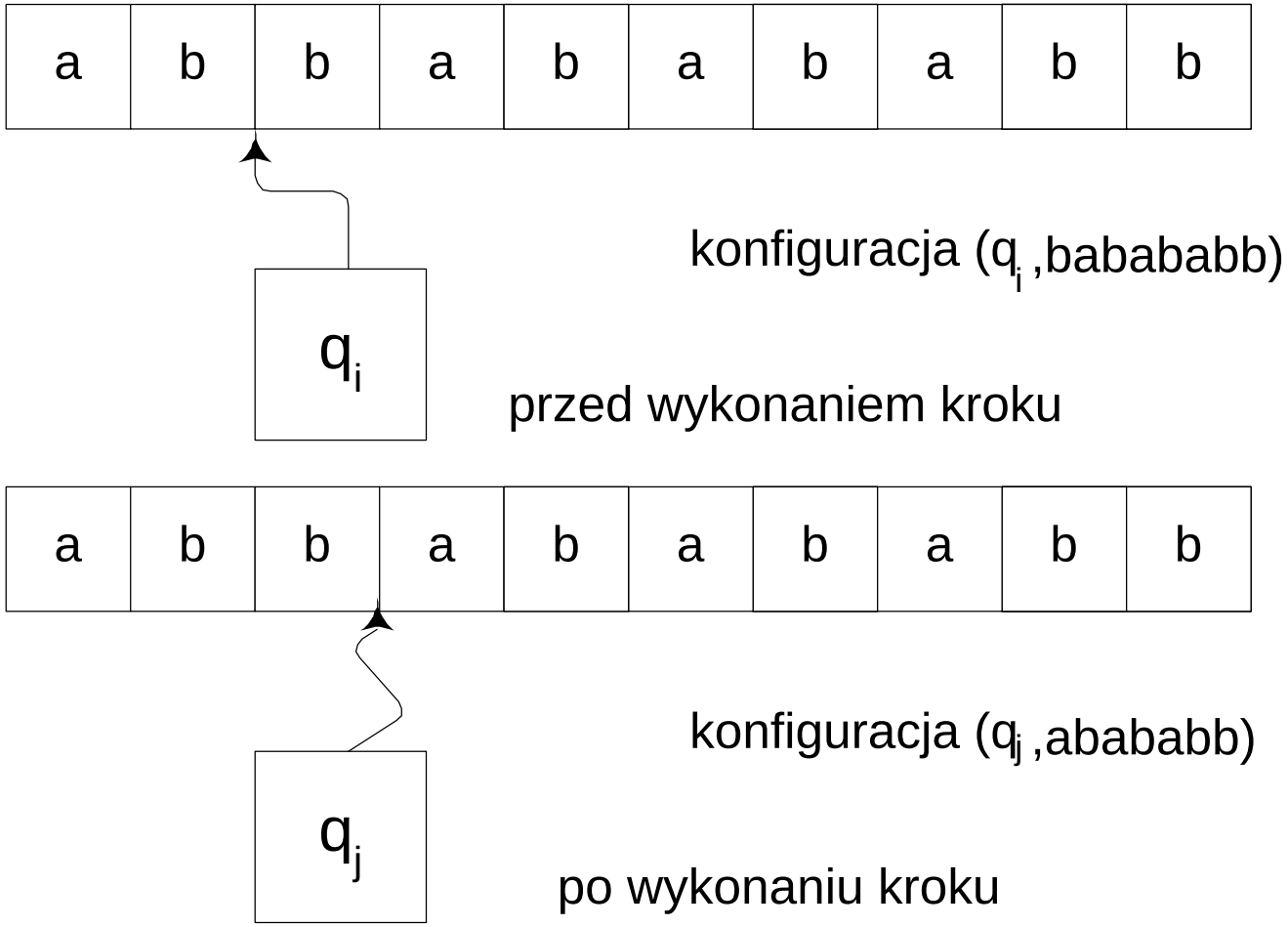
$F \subseteq Q$ – podzbiór stanów końcowych

$q_0 \in Q$ – stan początkowy

$\delta$ – funkcja przejścia

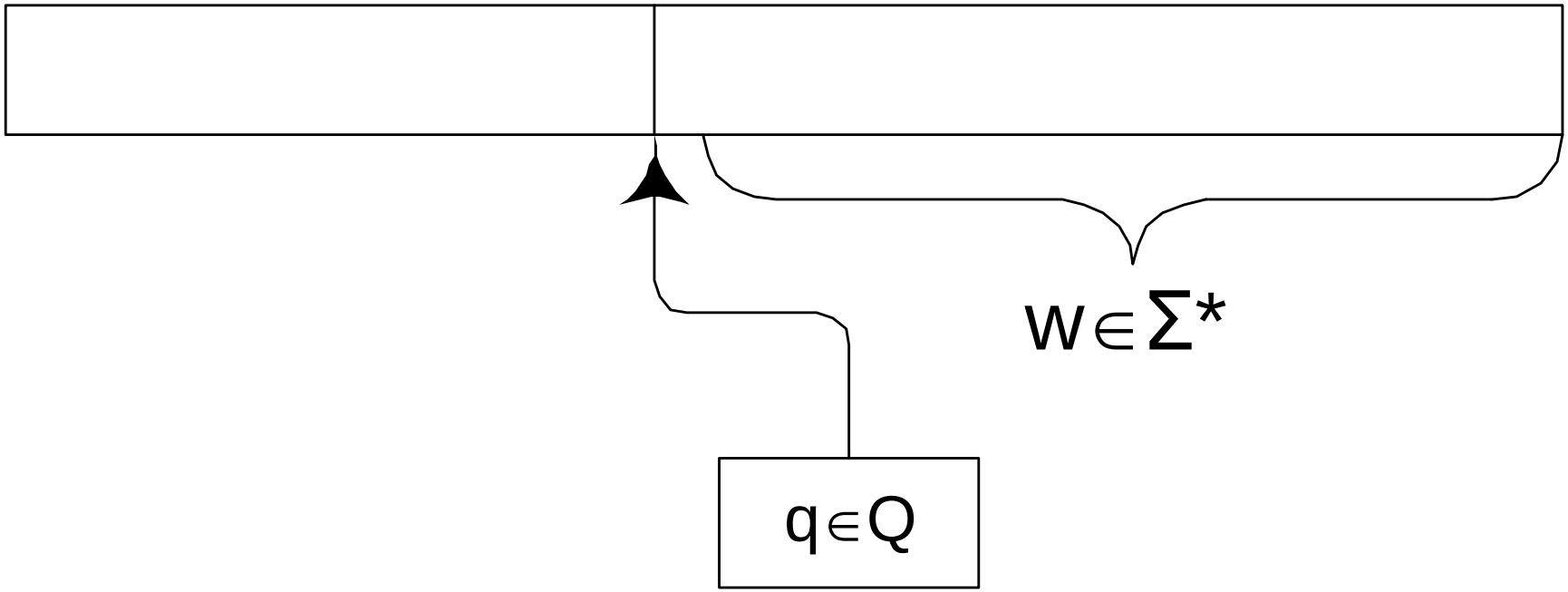
$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \alpha 2^Q$

# Konfiguracja automatu (1)



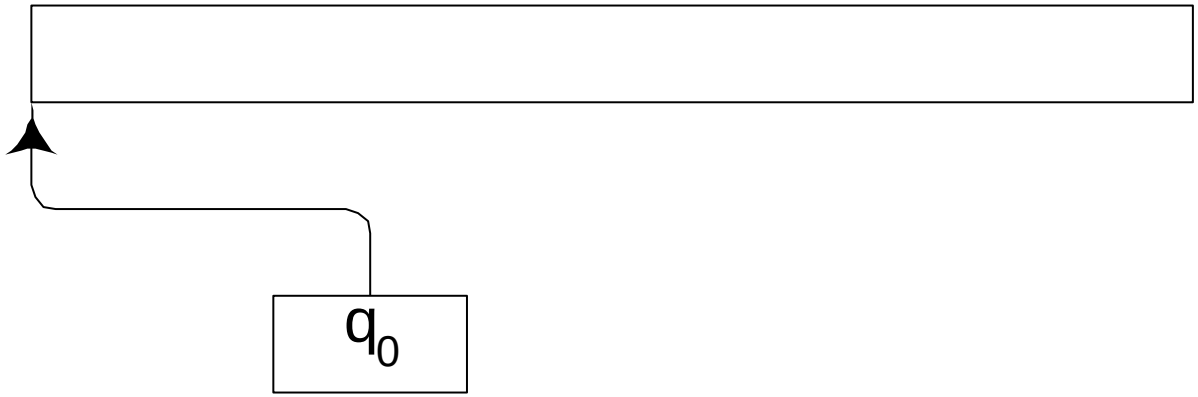
# Konfiguracja automatu (2)

Konfiguracja automatu to dwójka: (q, w), gdzie *q* jest aktualnym stanem, zaś *w* jest nieprzeczytaną przez automat częścią słowa zapisanego na taśmie wejściowej

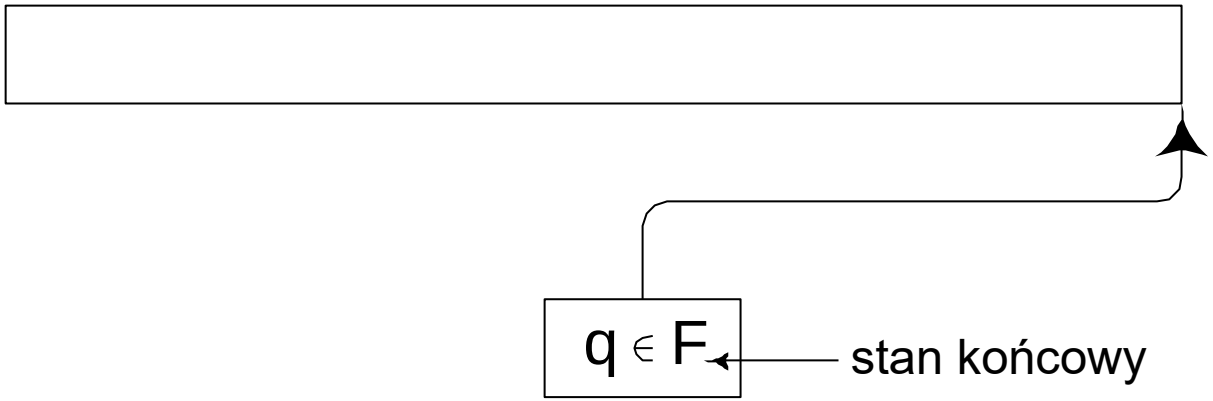


# Konfiguracja automatu (3)

Konfiguracja początkowa:




Konfiguracja końcowa akceptująca:



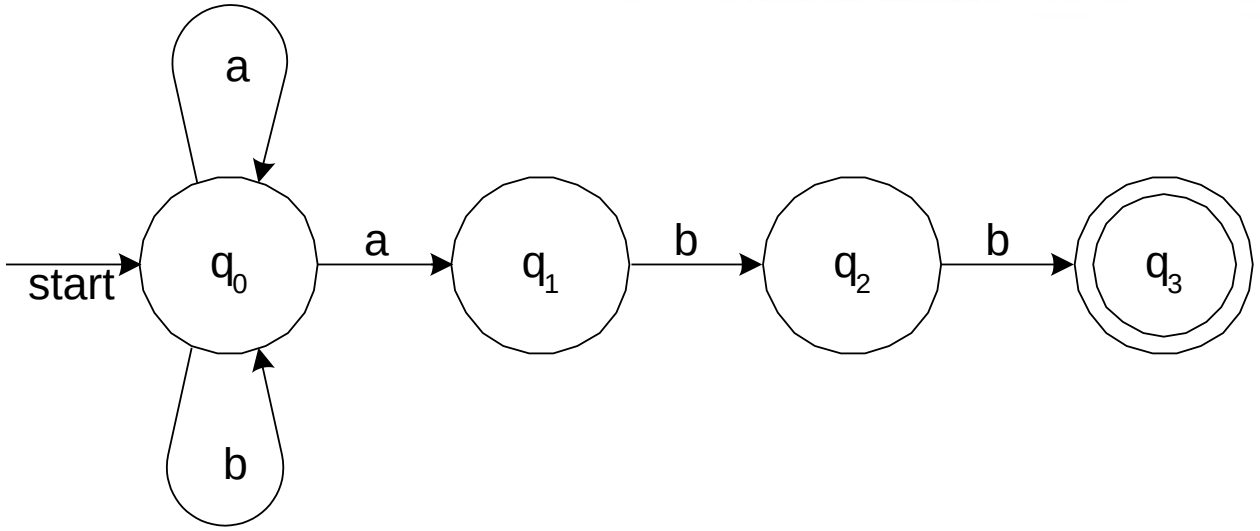
# Akceptacja języka przez automat

$x \in \Sigma^*$ jest słowem akceptowanym przez automat A (skończony), jeśli automat A startując z konfiguracji początkowej ze słowem x zapisanym na taśmie, **może** zatrzymać się po przeczytaniu do końca całego słowa x w konfiguracji akceptującej (w stanie akceptującym).

Język L jest akceptowany przez automat A (co oznaczamy L(A) ), jeśli język ten zbiorem wszystkich słów akceptowanych przez automat i tylko tych słów.

# Przykład (1)

Przykład: Automat niedeterministyczny akceptujący język regularny opisany wyrażeniem regularnym **(a|b)*abb**




$\Sigma=\{a,b\}$

$Q=\{q_0, q_1, q_2, q_3\}$

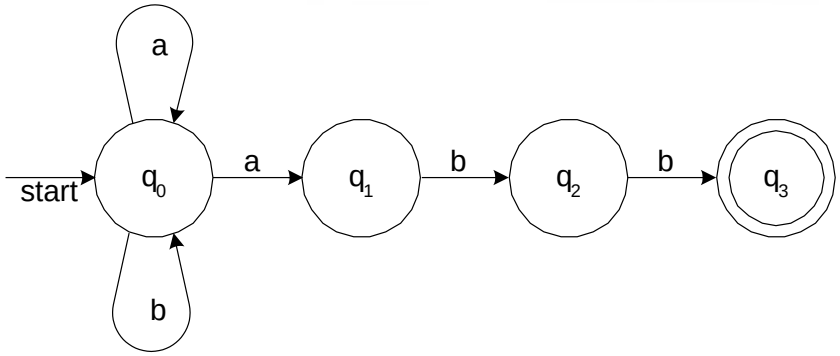
$F=\{q_3\}$

$q_0$ – stan początkowy

$\delta$ - funkcja przejścia:

| stan | a | b |
|---|---|---|
| $q_0$ | $\{q_0, q_1\}$ | $\{q_0\}$ |
| $q_1$ | $\varnothing$ | $\{q_2\}$ |
| $q_2$ | $\varnothing$ | $\{q_3\}$ |
| $q_3$ | $\varnothing$ | $\varnothing$ |

# Przykład (2)




Analizowane słowo: *aabb* $\in$ L( **(a|b)*abb** )

Możliwe wyprowadzenia:

- ( $q_0$ , aabb ) $\vdash$ ( $q_0$, abb ) $\vdash$ ( $q_1$, bb ) $\vdash$ ( $q_2$, b ) $\vdash$ ( $q_3$, $\varepsilon$) – konfiguracja końcowa akceptująca)
- ( $q_0$ , aabb ) $\vdash$ ( $q_0$, abb ) $\vdash$ ( $q_0$, bb ) $\vdash$ ( $q_0$, b ) $\vdash$ ( $q_0$, $\varepsilon$) – konfiguracja blokująca, bo $q_0 \notin F$
- ( $q_0$ , aabb ) $\vdash$ ( $q_1$, abb ) – konfiguracja blokująca, słowo nie zostało przeczytane do końca

Automat nie jest deterministyczny. Jednak **istnieje** wyprowadzenie (ciąg kroków), które doprowadza do konfiguracji akceptującej, więc zgodnie z definicją automat akceptuje to słowo.

# Własności automatów skończonych

Automat skończony A jest zupełny, gdy jego funkcja przejścia jest taka, że dla każdego stanu istnieje co najmniej jedno przejście przez każdy symbol alfabetu wejściowego. Automat zupełny przeczyta bez zablokowania się każde wejście do końca.

Automat skończony A jest deterministyczny, gdy jego funkcja przejścia jest taka, że dla każdego stanu istnieje co najwyżej jedno przejście przez każdy symbol alfabetu wejściowego. Automat deterministyczny nie może mieć także $\varepsilon$-przejść. Automat deterministyczny jest „wewnętrznie jednoznaczny”.

Automat skończony A jest deterministyczny i zupełny, gdy nie posiada $\varepsilon$-przejść oraz jego funkcja przejścia jest taka, że dla każdego stanu istnieje dokładnie jedno przejście przez każdy symbol alfabetu wejściowego.

# Przykład (3)

Przykład: Deterministyczny i zupełny automat akceptujący język opisany wyrażeniem regularnym **(a|b)*abb**

$\Sigma = \{a, b\}$

$Q = \{0, 1, 2, 3\}$

$F = \{3\}$

$q_0 = 0$

$\delta$ - funkcja przejścia:

| Stan | a | b |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 2 |
| 2 | 1 | 3 |
| 3 | 1 | 0 |

Przypomnienie poprzedniego przykładu : automat niedeterministyczny i niezupełny