Jaki język zrozumie automat?

Wojciech Dzik

Instytut Matematyki Uniwersytet Śląski Katowice
wojciech.dzik@us.edu.pl

7. Forum Matematyków Polskich,
12-17 września 2016, Olsztyn

Prosty Automat do kawy

"Przemawiamy" do automatu przy pomocy ciągów monet. Na wejściu (starcie) automatu są ciągi monet spośród:
5, 10, 20.
Automat rozpoznaje (rozumie) te ciągi, ktorych suma $= 20$, wtedy wydaje kawę.

Prosty Automat do kawy

"Przemawiamy" do automatu przy pomocy ciągów monet. Na wejściu (starcie) automatu są ciągi monet spośród:
5, 10, 20.
Automat rozpoznaje (rozumie) te ciągi, ktorych suma $= 20$, wtedy wydaje kawę.
Stany automatu wyznaczone są kwotą wrzuconych monet, mamy:

Prosty Automat do kawy

"Przemawiamy" do automatu przy pomocy ciągów monet. Na wejściu (starcie) automatu są ciągi monet spośród:
5, 10, 20.
Automat rozpoznaje (rozumie) te ciągi, ktorych suma $= 20$, wtedy wydaje kawę.
Stany automatu wyznaczone są kwotą wrzuconych monet, mamy:
stan początkowy:q_0, kwota $= 0$,

Prosty Automat do kawy

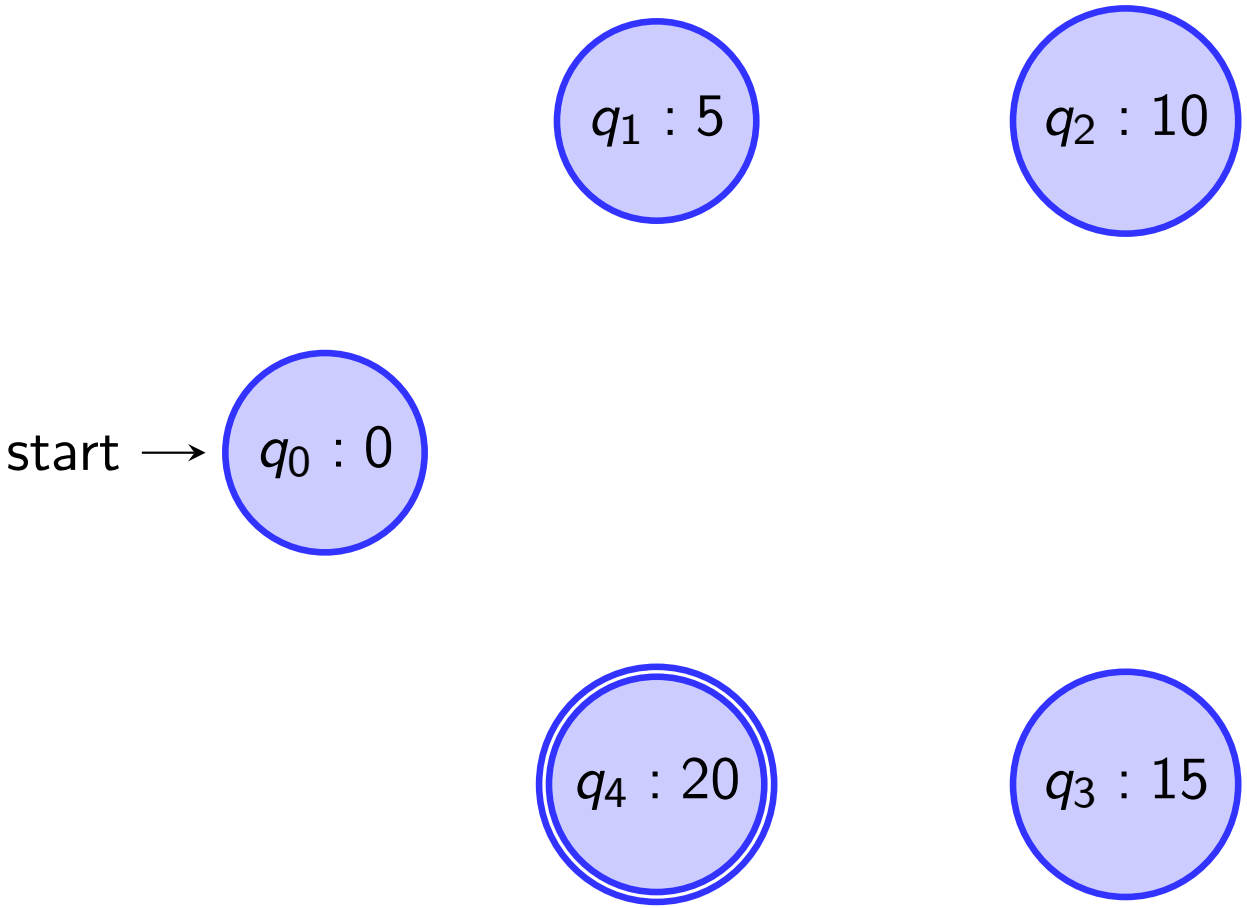
"Przemawiamy" do automatu przy pomocy ciągów monet. Na wejściu (starcie) automatu są ciągi monet spośród:
5, 10, 20.
Automat rozpoznaje (rozumie) te ciągi, ktorych suma $= 20$, wtedy wydaje kawę.
Stany automatu wyznaczone są kwotą wrzuconych monet, mamy:
stan początkowy:q_0, kwota $= 0$,
q_1 kwota $= 5$,
q_2 kwota $= 10$,
q_3 kwota $= 15$,

Prosty Automat do kawy

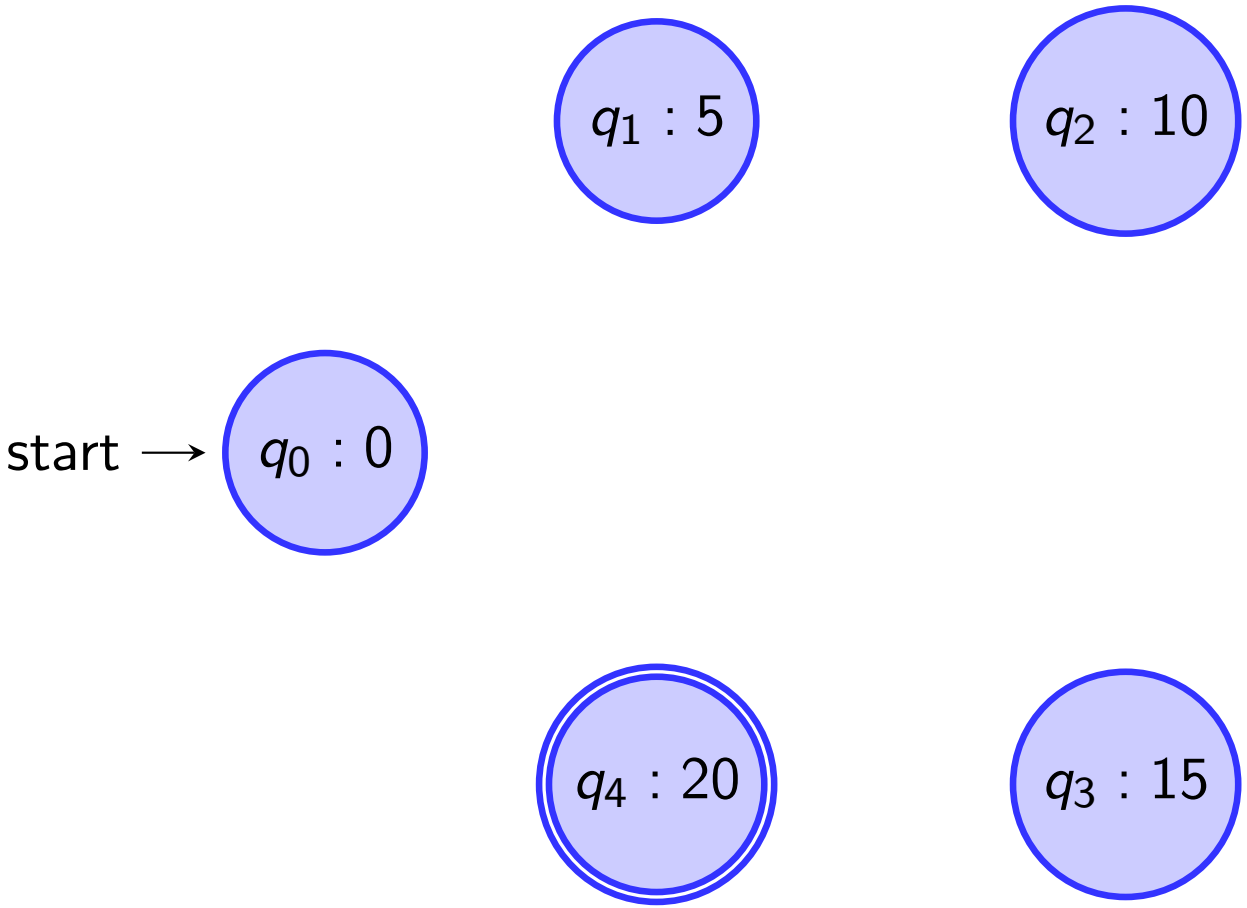
"Przemawiamy" do automatu przy pomocy ciągów monet. Na wejściu (starcie) automatu są ciągi monet spośród:
5, 10, 20.
Automat rozpoznaje (rozumie) te ciągi, ktorych suma $= 20$, wtedy wydaje kawę.
Stany automatu wyznaczone są kwotą wrzuconych monet, mamy:
stan początkowy:q_0, kwota $= 0$,
q_1 kwota $= 5$,
q_2 kwota $= 10$,
q_3 kwota $= 15$,
stan końcowy: q_4, kwota $= 20$

(nie wydaje reszty)

Automat do kawy $\mathcal{A}_K$

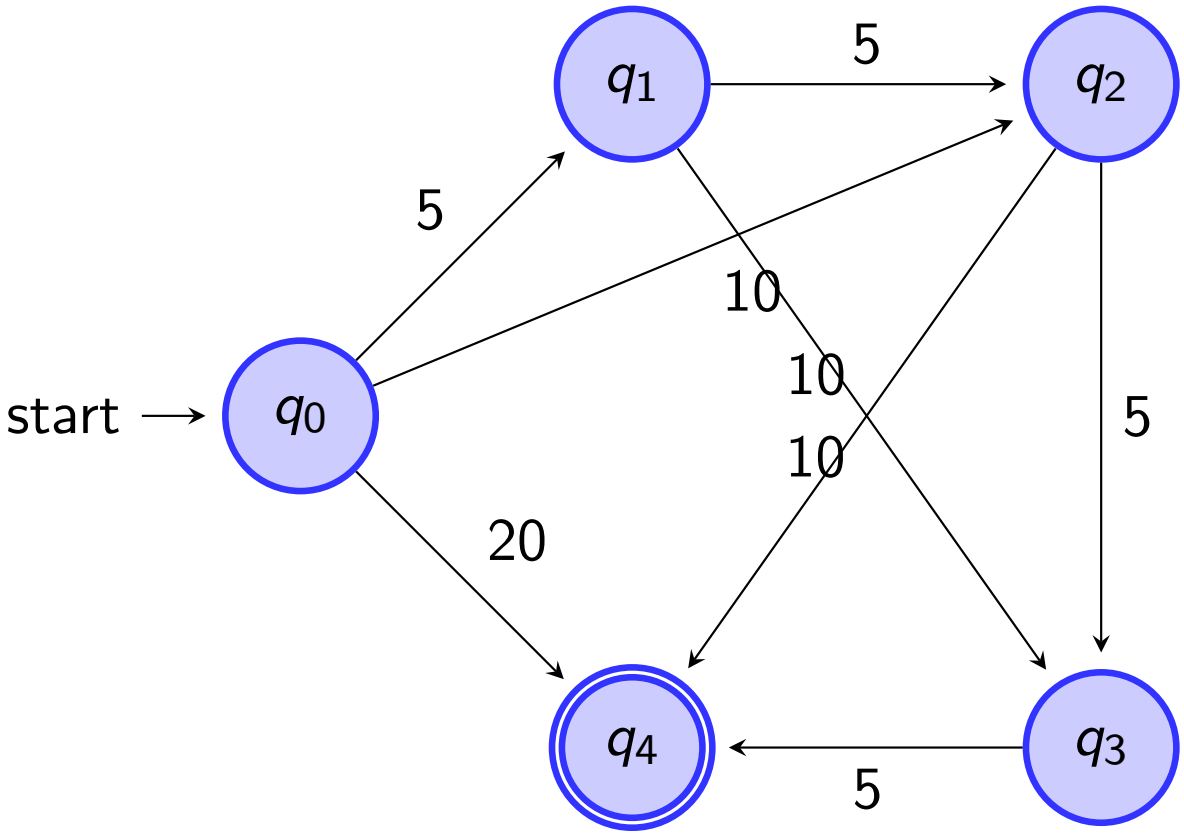


Automat do kawy $\mathcal{A}_K$

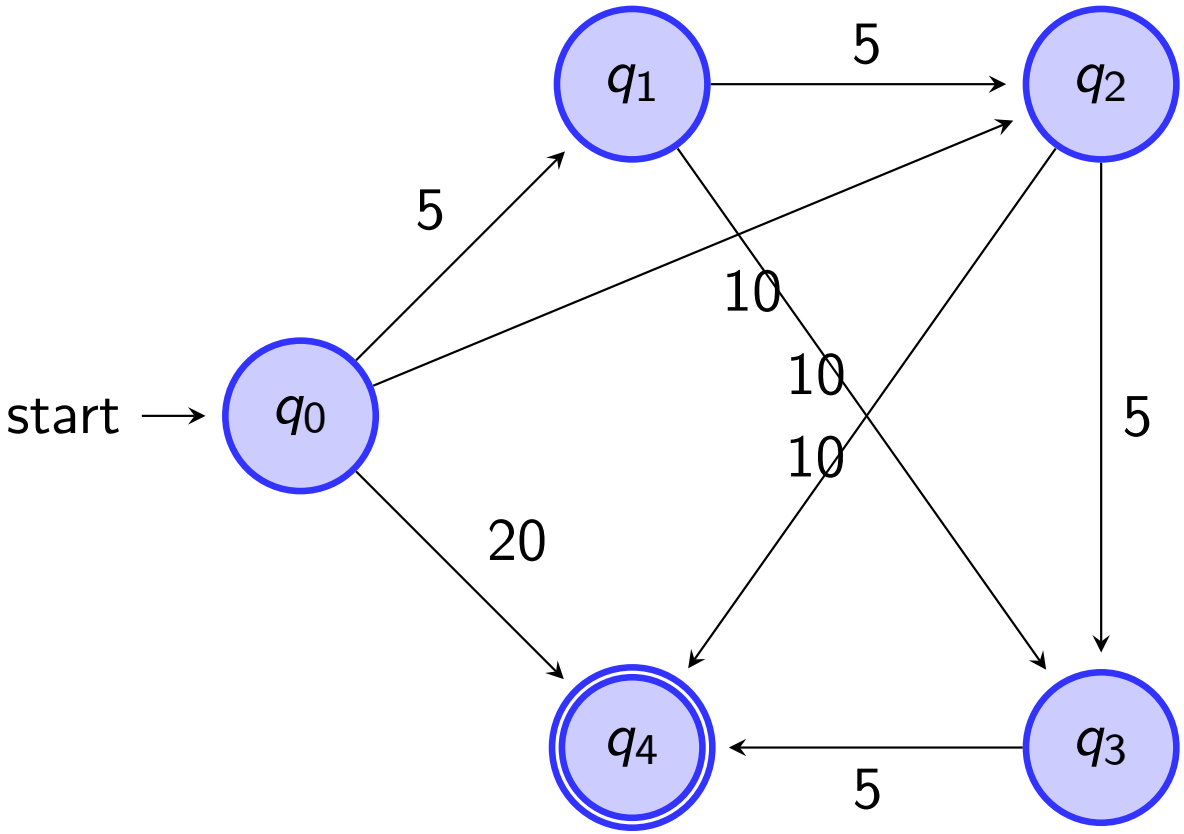


Stan początkowy:q_0, kwota $= 0$,
q_1 kwota $= 5$,
q_2 kwota $= 10$,
q_3 kwota $= 15$,
stan końcowy: q_4, kwota $= 20$

Automat do kawy $\mathcal{A}_K$

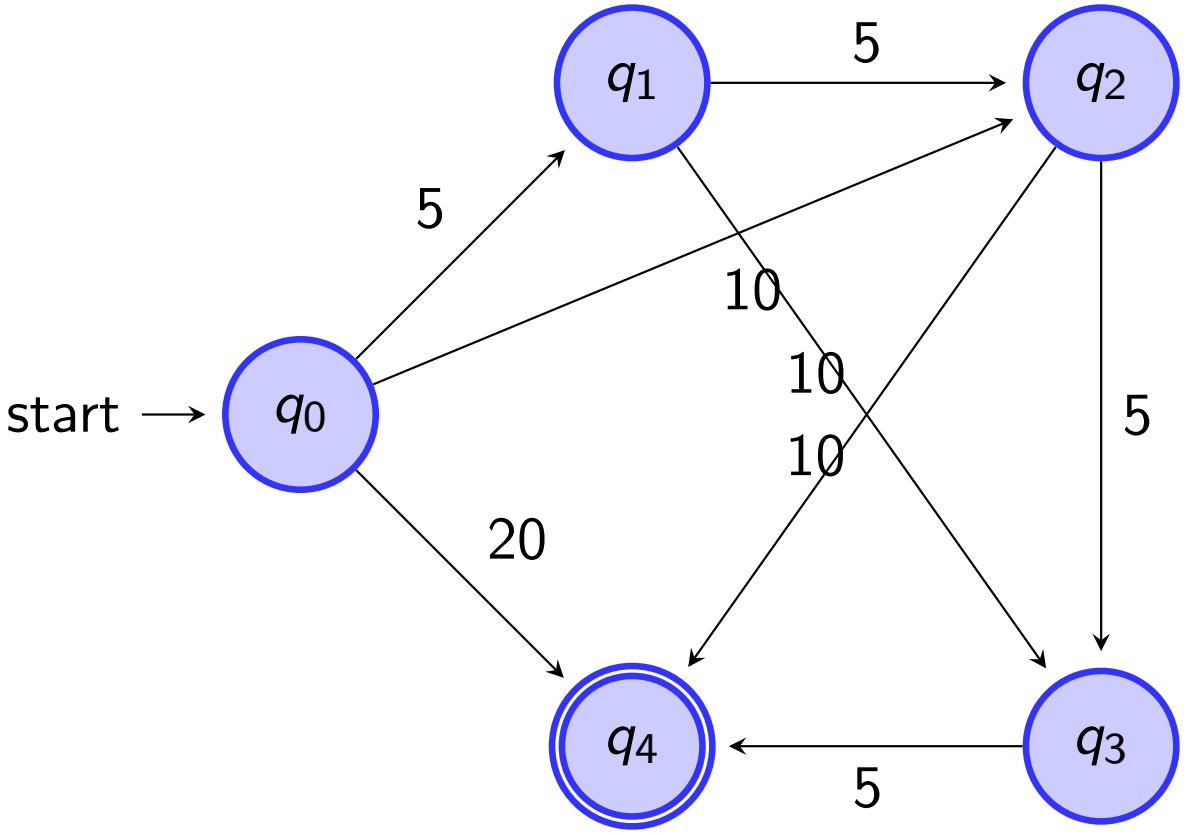


Automat do kawy $\mathcal{A}_K$

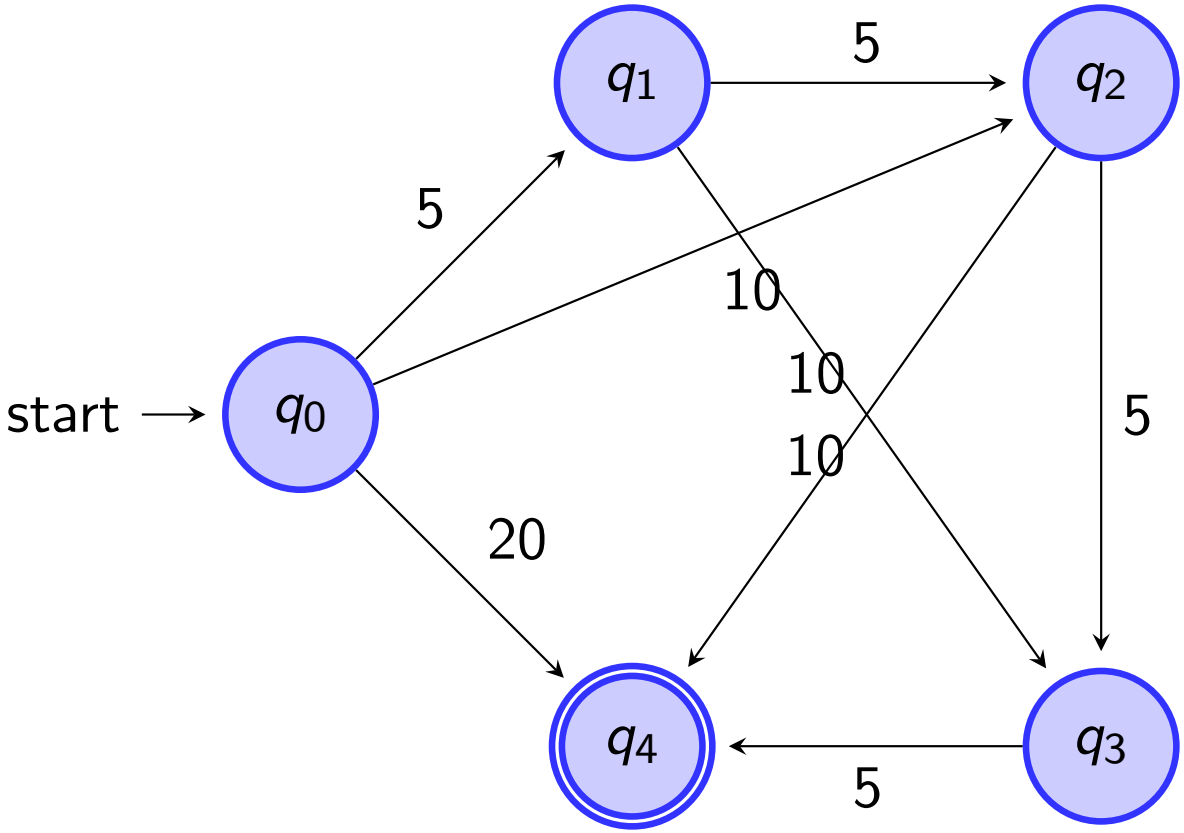


ciągi (czyli słowa), które dają sukces (kawę) - słowa rozpoznane:
(5, 5, 5, 5), (5, 5, 10), (5, 10, 5), (10, 5, 5), (10, 10), (20)

Automat do kawy $\mathcal{A}_K$

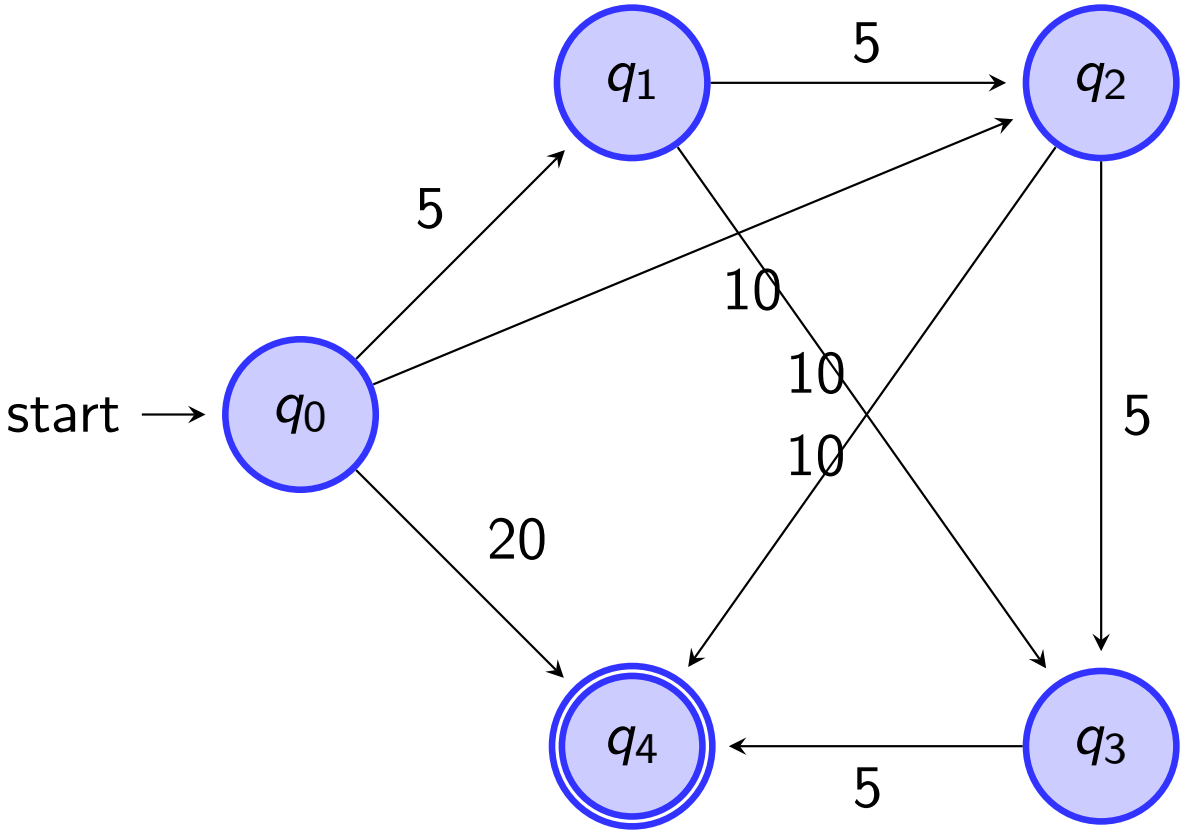


Automat do kawy $\mathcal{A}_K$



Automat rozpoznaje (”rozumie”) ciągi, czyli słowa:
(5, 5, 5, 5), (5, 5, 10), (5, 10, 5), (10, 5, 5), (10, 10), (20).
Słowa zbudowane z alfabetu $\Sigma = \{5, 10, 20\}$

Automat do kawy $\mathcal{A}_K$



Automat rozpoznaje (”rozumie”) ciągi, czyli słowa:
(5, 5, 5, 5), (5, 5, 10), (5, 10, 5), (10, 5, 5), (10, 10), (20).
Słowa zbudowane z alfabetu $\Sigma = \{5, 10, 20\}$
$L(\mathcal{A}_K) = \{(5, 5, 5, 5), (5, 5, 10), (5, 10, 5), (10, 5, 5), (10, 10), (20)\}$,
język zaakceptowany przez $\mathcal{A}_K$.

Automat skończony

Ogólnie: automat to $(Q, \Sigma, \delta, s, F)$,

Automat skończony

Ogólnie: automat to $(Q, \Sigma, \delta, s, F)$,
Q - skończony zbiór stanów,
Σ - alfabet, skończony zbiór znaków, (symboli)
$\delta : Q \times \Sigma \rightarrow Q$ - funkcja przejścia
– $\delta(q, a)$ to stan do którego przechodzimy ze stanu q na skutek wczytania znaku a,
s to stan początkowy
F - zbiór stanów końcowych (akceptujących)

Automat skończony

Ogólnie: automat to $(Q, \Sigma, \delta, s, F)$,
Q - skończony zbiór stanów,
Σ - alfabet, skończony zbiór znaków, (symboli)
$\delta : Q \times \Sigma \to Q$ - funkcja przejścia
– $\delta(q, a)$ to stan do którego przechodzimy ze stanu q na skutek wczytania znaku a,
s to stan początkowy
F - zbiór stanów końcowych (akceptujących)

Przykład.
Automat do kawy: $(\{q_0, \ldots, q_4\}, \{5, 10, 20\}, \delta, q_0, \{q_4\})$
$\delta(q_i, 5) = q_{i+1}, i = 0, \ldots, 3,$
$\delta(q_i, 10) = q_{i+2}, i = 0, 1, 2, \quad \delta(q_0, 20) = q_4.$

Automat skończony, alfabet, słowa i język

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem, ε - słowo puste

Automat skończony, alfabet, słowa i język

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem, ε - słowo puste

Przykłady.
• $\Sigma = \{a, b\}$, wtedy Σ^* :
$\varepsilon, a, b, aa, bb, ab, ba, aaa, bbb, aba, aab, baa, \ldots, bbabaaab, \ldots$

Automat skończony, alfabet, słowa i język

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem, ε - słowo puste

Przykłady.
- $\Sigma = \{a, b\}$, wtedy Σ^* :

$\varepsilon, a, b, aa, bb, ab, ba, aaa, bbb, aba, aab, baa, \ldots, bbabaaab, \ldots$
- $\Sigma = \{0, 1\}$, wtedy Σ^* :

$\varepsilon, 0, 1, 00, 11, 01, 10, 000, 111, 010, 001, 100, \ldots,$

Automat skończony, alfabet, słowa i język

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem, ε - słowo puste

Przykłady.
- $\Sigma = \{a, b\}$, wtedy Σ^* :

$\varepsilon, a, b, aa, bb, ab, ba, aaa, bbb, aba, aab, baa, \ldots, bbabaaab, \ldots$
- $\Sigma = \{0, 1\}$, wtedy Σ^* :

$\varepsilon, 0, 1, 00, 11, 01, 10, 000, 111, 010, 001, 100, \ldots,$

Działanie na słowach - konkatenacja (połączenie) słów:
$w_1 = a_1 \ldots a_n$, $w_2 = b_1 \ldots b_k$, to wtedy $w_1 w_2 = a_1 \ldots a_n b_1 \ldots b_k$,
łączne, nie przemienne,

Automat skończony, alfabet, słowa i język

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem, ε - słowo puste

Przykłady.
- $\Sigma = \{a, b\}$, wtedy Σ^* :

$\varepsilon, a, b, aa, bb, ab, ba, aaa, bbb, aba, aab, baa, \ldots, bbabaaab, \ldots$
- $\Sigma = \{0, 1\}$, wtedy Σ^* :

$\varepsilon, 0, 1, 00, 11, 01, 10, 000, 111, 010, 001, 100, \ldots,$

Działanie na słowach - konkatenacja (połączenie) słów:
$w_1 = a_1 \ldots a_n$, $w_2 = b_1 \ldots b_k$, to wtedy $w_1 w_2 = a_1 \ldots a_n b_1 \ldots b_k$,
łączne, nie przemienne,
ε jest elementem neutralnym:
$\varepsilon w = w\varepsilon = w$
monoid syntaktyczny Σ^*, z elementem neutralnym ε,

Automat skończony

$(Q, \Sigma, \delta, s, F)$ automat skończony,

Automat skończony

$(Q, \Sigma, \delta, s, F)$ automat skończony,
$\delta : Q \times \Sigma \longrightarrow Q$ - funkcja przejścia, rozszerzamy na
$\delta : Q \times \Sigma^* \longrightarrow Q$ - funkcja przejścia - następująco:

Automat skończony

$(Q, \Sigma, \delta, s, F)$ automat skończony,
$\delta : Q \times \Sigma \to Q$ - funkcja przejścia, rozszerzamy na
$\delta : Q \times \Sigma^* \to Q$ - funkcja przejścia - następująco:
$\delta(q, \varepsilon) = q$
$\delta(q, wa) = \delta(\delta(q, w), a)$

Automat skończony

$(Q, \Sigma, \delta, s, F)$ automat skończony,
$\delta : Q \times \Sigma \to Q$ - funkcja przejścia, rozszerzamy na
$\delta : Q \times \Sigma^* \to Q$ - funkcja przejścia - następująco:
$\delta(q, \varepsilon) = q$
$\delta(q, wa) = \delta(\delta(q, w), a)$

Np. W $\mathcal{A}_K$: $\delta(q_0, 555) = q_3$, $\delta(q_0, 5555) = q_4$

Język j. rozpoznany, zaakcetowany przez automat $\mathcal{A}$, tzn.
$L(\mathcal{A}) = \{w \in \Sigma^* : \delta(s, w) \in F\}$

Automat skończony

$(Q, \Sigma, \delta, s, F)$ automat skończony,
$\delta : Q \times \Sigma \to Q$ - funkcja przejścia, rozszerzamy na
$\delta : Q \times \Sigma^* \to Q$ - funkcja przejścia - następująco:
$\delta(q, \varepsilon) = q$
$\delta(q, wa) = \delta(\delta(q, w), a)$

Np. W $\mathcal{A}_K$: $\delta(q_0, 555) = q_3$, $\delta(q_0, 5555) = q_4$

Język j. rozpoznany, zaakcetowany przez automat $\mathcal{A}$, tzn.
$L(\mathcal{A}) = \{w \in \Sigma^* : \delta(s, w) \in F\}$

W $\mathcal{A}_K$ słowo 555 nie jest rozpoznane, słowo 5555 - rozpoznane

Automat skończony

$(Q, \Sigma, \delta, s, F)$ automat skończony,
$\delta : Q \times \Sigma \to Q$ - funkcja przejścia, rozszerzamy na
$\delta : Q \times \Sigma^* \to Q$ - funkcja przejścia - następująco:
$\delta(q, \varepsilon) = q$
$\delta(q, wa) = \delta(\delta(q, w), a)$

Np. W $\mathcal{A}_K$: $\delta(q_0, 555) = q_3$, $\delta(q_0, 5555) = q_4$

Język j. rozpoznany, zaakcetowany przez automat $\mathcal{A}$, tzn.
$L(\mathcal{A}) = \{w \in \Sigma^* : \delta(s, w) \in F\}$

W $\mathcal{A}_K$ słowo 555 nie jest rozpoznane, słowo 5555 - rozpoznane

Język jest regularny gdy jest rozpoznawany przez jakiś automat, czyli $L = L(\mathcal{A})$ dla pewnego automatu skończonego $\mathcal{A}$.

Języki i działania na nich

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem i ε (długości 0).

Języki i działania na nich

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem i ε (długości 0).
Język L nad alfabetem Σ to dowolny podzbiór $L \subseteq \Sigma^*$.

Języki i działania na nich

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem i ε (długości 0).
Język L nad alfabetem Σ to dowolny podzbiór $L \subseteq \Sigma^*$.

Języki i działania na nich

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem i ε (długości 0).
Język L nad alfabetem Σ to dowolny podzbiór $L \subseteq \Sigma^*$.

Działania: $(L_1, L_2 \subseteq \Sigma^*)$

- $L_1 \cup L_2 = \{w : w \in L_1 \text{ lub } w \in L_2\}$,
- złożenie $L_1 \star L_2 = \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}$, piszemy : $L_1 L_2$,

$\{\varepsilon\} = L^0$, $L = L^1$, $LL = L^2$, $LLL = L^3$ itd.

Języki i działania na nich

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem i ε (długości 0).
Język L nad alfabetem Σ to dowolny podzbiór $L \subseteq \Sigma^*$.

Działania: $(L_1, L_2 \subseteq \Sigma^*)$
- $L_1 \cup L_2 = \{w : w \in L_1 \text{ lub } w \in L_2\}$,
- złożenie $L_1 \star L_2 = \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}$, piszemy : $L_1 L_2$,

$\{\varepsilon\} = L^0$, $L = L^1$, $LL = L^2$, $LLL = L^3$ itd.
- Gwiazdka Kleenego, domknięcie Kleenego L^* języka L:

$L^* = \bigcup\{L^k : k = 0, 1, 2, \dots\}$,

Języki i działania na nich

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem i ε (długości 0).
Język L nad alfabetem Σ to dowolny podzbiór $L \subseteq \Sigma^*$.

Działania: $(L_1, L_2 \subseteq \Sigma^*)$
- $L_1 \cup L_2 = \{w : w \in L_1 \text{ lub } w \in L_2\}$,
- złożenie $L_1 \star L_2 = \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}$, piszemy : $L_1 L_2$,

$\{\varepsilon\} = L^0$, $L = L^1$, $LL = L^2$, $LLL = L^3$ itd.
- Gwiazdka Kleenego, domknięcie Kleenego L^* języka L:

$L^* = \bigcup\{L^k : k = 0, 1, 2, \dots\}$,
także $L_1 \cap L_2$ oraz dopełnienie $L' = \Sigma^* \setminus L$.

Równania na językach, np.: jeżeli $\varepsilon \notin A$ to równanie

$$X = AX \cup B$$

ma dokładnie jedno rozwiązanie $X =$

Języki i działania na nich

Σ alfabet (skończony zbiór znaków),
Σ^* zbiór wszystkich słów nad alfabetem Σ, tj. zbiór wszystkich skończonych ciągów nad alfabetem i ε (długości 0).
Język L nad alfabetem Σ to dowolny podzbiór $L \subseteq \Sigma^*$.

Działania: $(L_1, L_2 \subseteq \Sigma^*)$
- $L_1 \cup L_2 = \{w : w \in L_1 \text{ lub } w \in L_2\}$,
- złożenie $L_1 \star L_2 = \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}$, piszemy : $L_1 L_2$,

$\{\varepsilon\} = L^0$, $L = L^1$, $LL = L^2$, $LLL = L^3$ itd.
- Gwiazdka Kleenego, domknięcie Kleenego L^* języka L:

$L^* = \bigcup\{L^k : k = 0, 1, 2, \dots\}$,
także $L_1 \cap L_2$ oraz dopełnienie $L' = \Sigma^* \setminus L$.

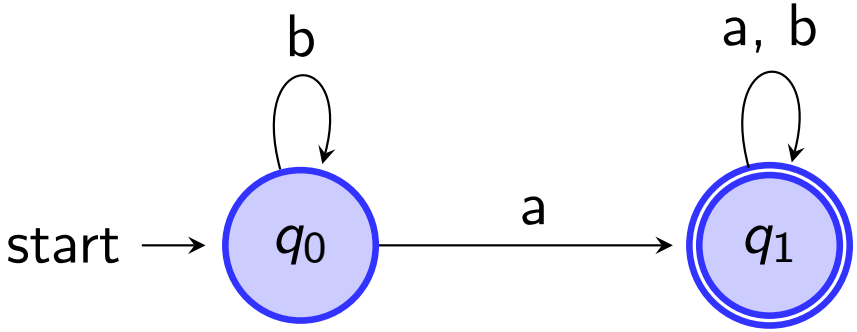
Równania na językach, np.: jeżeli $\varepsilon \notin A$ to równanie

$$X = AX \cup B$$

ma dokładnie jedno rozwiązanie $X = A^*B$ (rozw. Arden) .

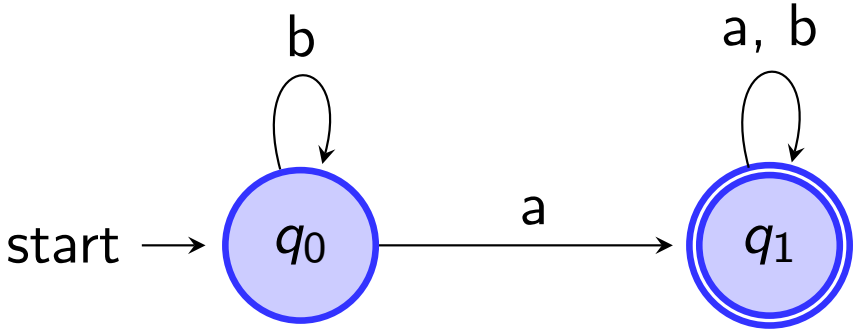
Automat skończony

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



Automat skończony

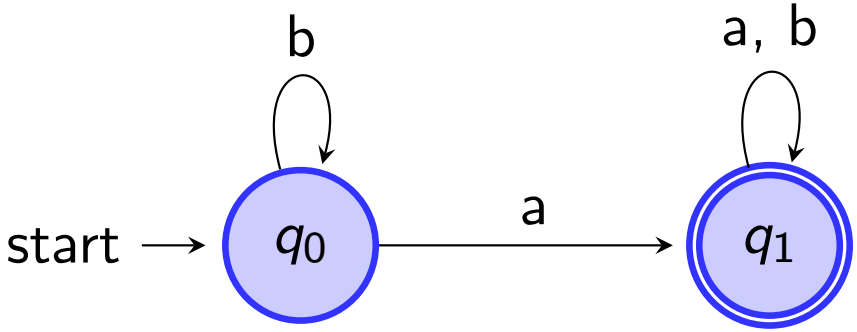
$\Sigma = \{a, b\}$



rozpoznaje słowa np. : a, ba, bba, $b \ldots ba$, ..., aa, ab, aba, abb,...
rozpoznaje wszystkie słowa zawierające choć raz literę a

Automat skończony

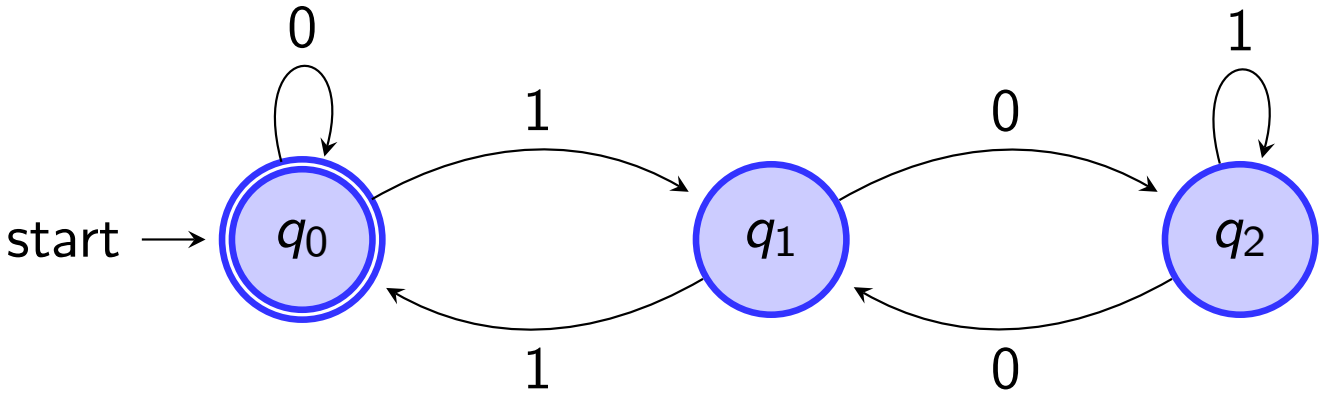
$\Sigma = \{a, b\}$



rozpoznaje słowa np. : a, ba, bba, $b \ldots ba$, ..., aa, ab, aba, abb,...
rozpoznaje wszystkie słowa zawierające choć raz literę a
$L(\mathcal{A}) = \{w \in \Sigma^* : w \text{ zawiera choć raz literę } a\}$.

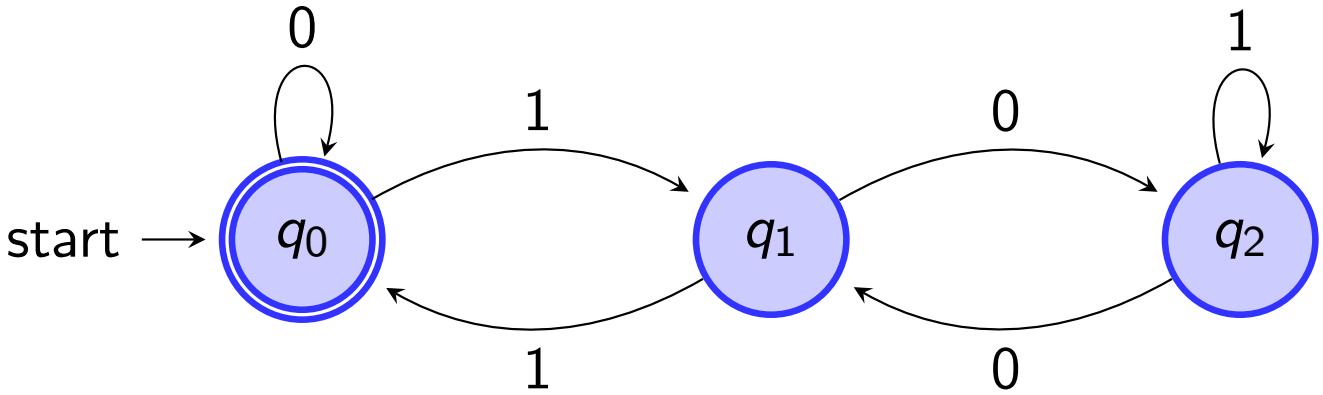
Automat "liczący"

$\Sigma = \{0, 1\}$



Automat "liczący"

$\Sigma = \{0, 1\}$

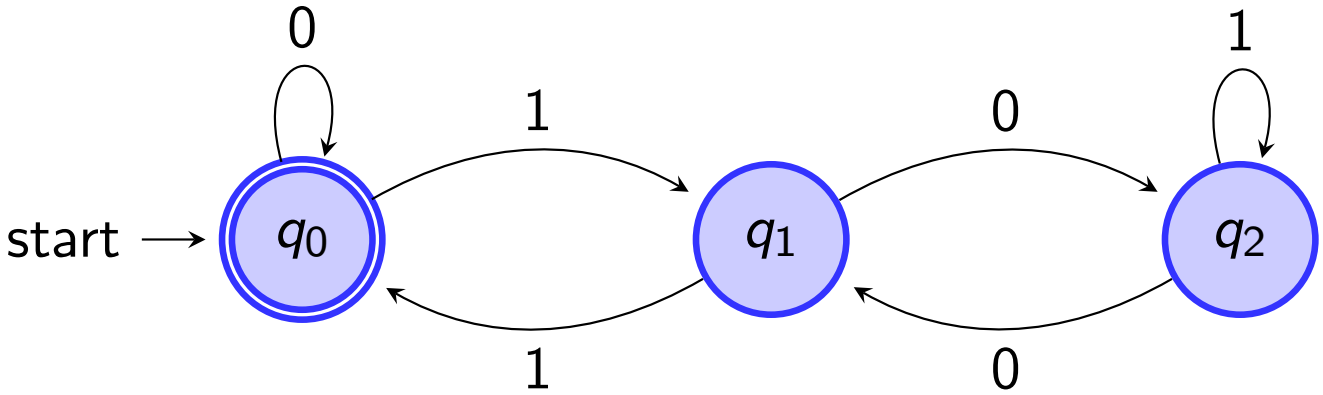


rozpoznaje np. : 0, 11, 110, 1001, 1100, 1111, 10010, .. nie 100

co to za słowa?

Automat "liczący"

$\Sigma = \{0, 1\}$



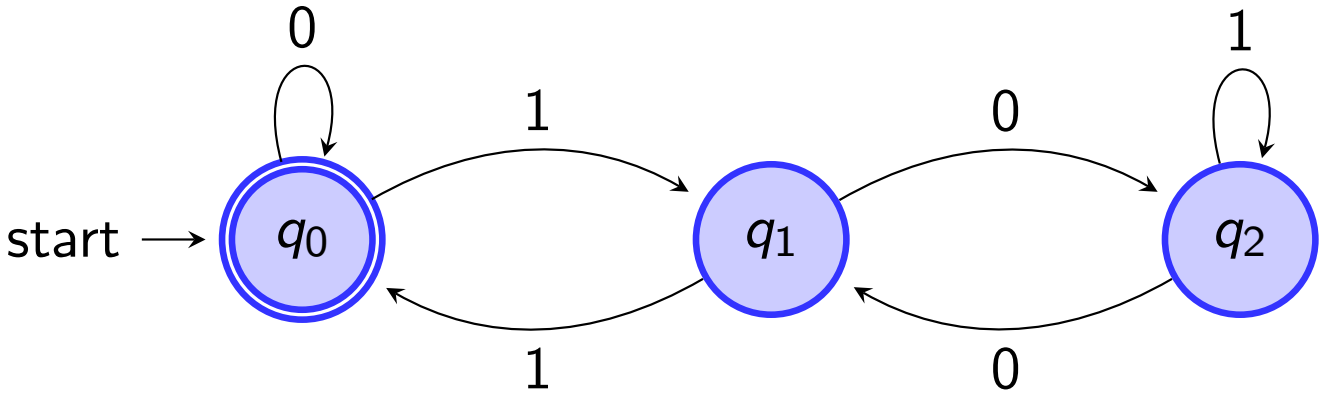
rozpoznaje np. : 0, 11, 110, 1001, 1100, 1111, 10010, .. nie 100

co to za słowa?

$0_{(2)} = 0$, $11_{(2)} = 1 * 2^1 + 1 * 2^0 = 3$,

Automat "liczący"

$\Sigma = \{0, 1\}$



rozpoznaje np. : 0, 11, 110, 1001, 1100, 1111, 10010, .. nie 100

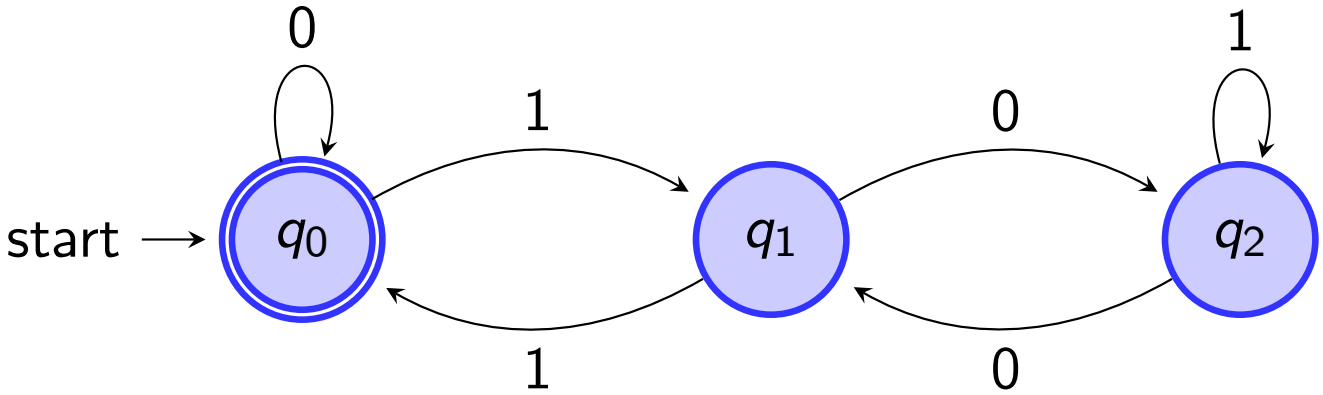
co to za słowa?

$0_{(2)} = 0$, $11_{(2)} = 1 * 2^1 + 1 * 2^0 = 3$,

$110_{(2)} = 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = 6$,

Automat "liczący"

$\Sigma = \{0, 1\}$



rozpoznaje np. : 0, 11, 110, 1001, 1100, 1111, 10010, .. nie 100

co to za słowa?

$0_{(2)} = 0$, $11_{(2)} = 1 * 2^1 + 1 * 2^0 = 3$,

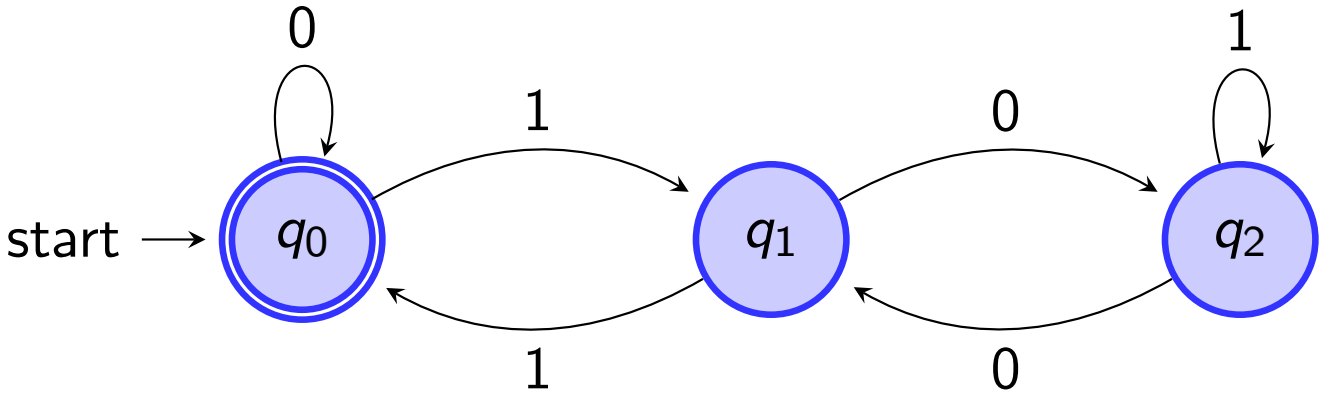
$110_{(2)} = 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = 6$,

$1001_{(2)} = 9$, $1100_{(2)} = 12$, $1111_{(2)} = 15$, $10010_{(2)} = 18$, itd.

nie rozpoznaje $100_{(2)} = 4$

Automat "liczący"

$\Sigma = \{0, 1\}$



rozpoznaje np. : 0, 11, 110, 1001, 1100, 1111, 10010, .. nie 100

co to za słowa?

$0_{(2)} = 0$, $11_{(2)} = 1 * 2^1 + 1 * 2^0 = 3$,

$110_{(2)} = 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = 6$,

$1001_{(2)} = 9$, $1100_{(2)} = 12$, $1111_{(2)} = 15$, $10010_{(2)} = 18$, itd.

nie rozpoznaje $100_{(2)} = 4$

$L(\mathcal{A}) = \{z \in \Sigma^* : z \textit{ zapis w układzie binarnym - dzieli się przez } 3\}$

Projektowanie automatów

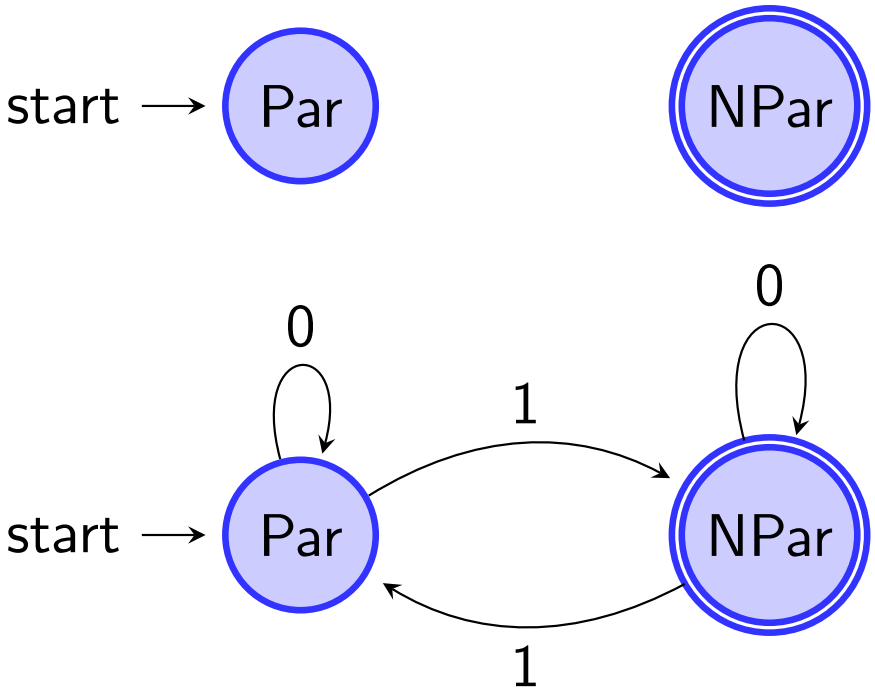
Zadanie: dla $\Sigma = \{0.1\}$, zbudować automat, który rozpoznaje wszystkie słowa nad Σ, ktore mają nieparzystą ilość jedynek. Takie skończone słowo może być bardzo długie, np. stąd do Księzyca, a automat nie ma pamięci, aby to zliczyć.

Projektowanie automatów

Zadanie: dla $\Sigma = \{0.1\}$, zbudować automat, który rozpoznaje wszystkie słowa nad Σ, ktore mają nieparzystą ilość jedynek. Takie skończone słowo może być bardzo długie, np. stąd do Księzyca, a automat nie ma pamięci, aby to zliczyć.

Projektowanie automatów

Zadanie: dla $\Sigma = \{0.1\}$, zbudować automat, który rozpoznaje wszystkie słowa nad Σ, ktore mają nieparzystą ilość jedynek. Takie skończone słowo może być bardzo długie, np. stąd do Księzyca, a automat nie ma pamięci, aby to zliczyć.



Automaty skończone niedeterministyczne

Dotychczas rozważaliśmy automaty deterministyczne -
$\delta : Q \times \Sigma \rightarrow Q$ była funkcją, $\delta(q, w)$ jednoznacznie określone.
A gdyby nie? wtedy automaty niedeterministyczne.

Automaty skończone niedeterministyczne

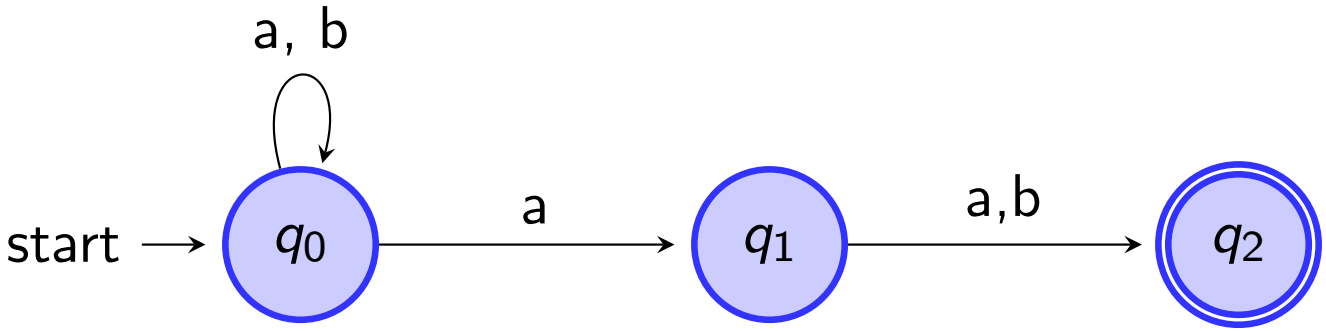
Dotychczas rozważaliśmy automaty deterministyczne - $\delta : Q \times \Sigma \rightarrow Q$ była funkcją, $\delta(q, w)$ jednoznacznie określone. A gdyby nie? wtedy automaty niedeterministyczne.

$\Sigma = \{a, b\}$. Zbudować automat rozpoznający słowa, w ktorych drugi symbol od końca $= a$.

Automaty skończone niedeterministyczne

Dotychczas rozważaliśmy automaty deterministyczne - $\delta : Q \times \Sigma \rightarrow Q$ była funkcją, $\delta(q, w)$ jednoznacznie określone. A gdyby nie? wtedy automaty niedeterministyczne.

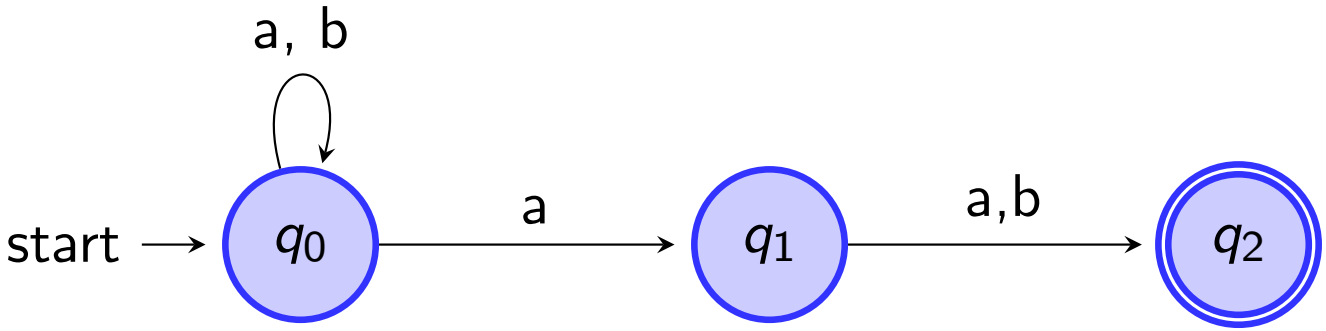
$\Sigma = \{a, b\}$. Zbudować automat rozpoznający słowa, w ktorych drugi symbol od końca $= a$.



Automaty skończone niedeterministyczne

Dotychczas rozważaliśmy automaty deterministyczne - $\delta : Q \times \Sigma \to Q$ była funkcją, $\delta(q, w)$ jednoznacznie określone. A gdyby nie? wtedy automaty niedeterministyczne.

$\Sigma = \{a, b\}$. Zbudować automat rozpoznający słowa, w ktorych drugi symbol od końca $= a$.

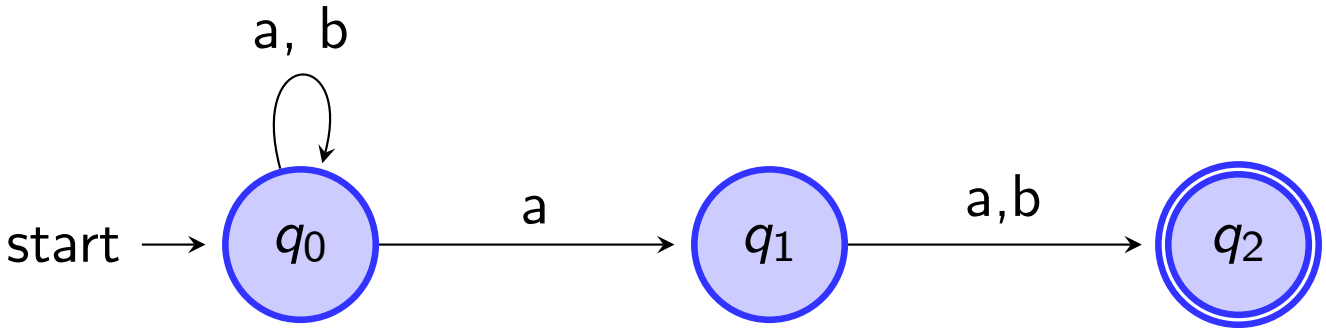


$\delta(q_0, a) = q_0$ albo $= q_1$ - automat niedeterministyczny, δ nie jest funkcją, $\delta(q_0, a) = \{q_0, q_1\}$ - zbiór,

Automaty skończone niedeterministyczne

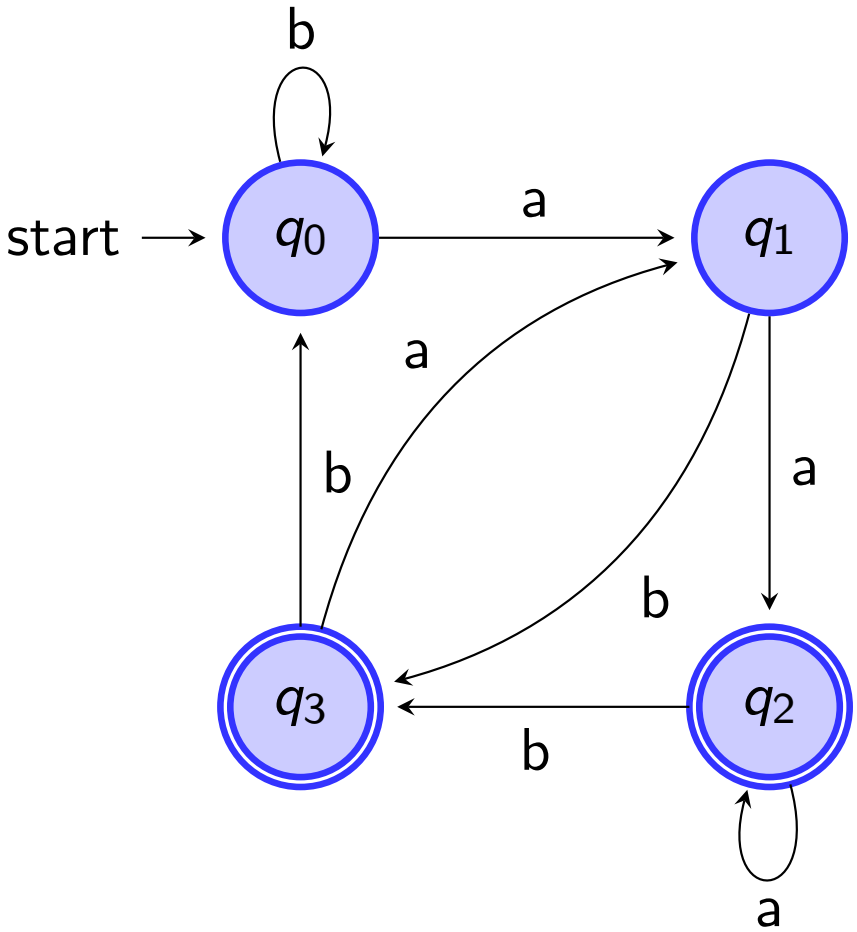
Dotychczas rozważaliśmy automaty deterministyczne -
$\delta : Q \times \Sigma \to Q$ była funkcją, $\delta(q, w)$ jednoznacznie określone.
A gdyby nie? wtedy automaty niedeterministyczne.

$\Sigma = \{a, b\}$. Zbudować automat rozpoznający słowa, w ktorych drugi symbol od końca $= a$.

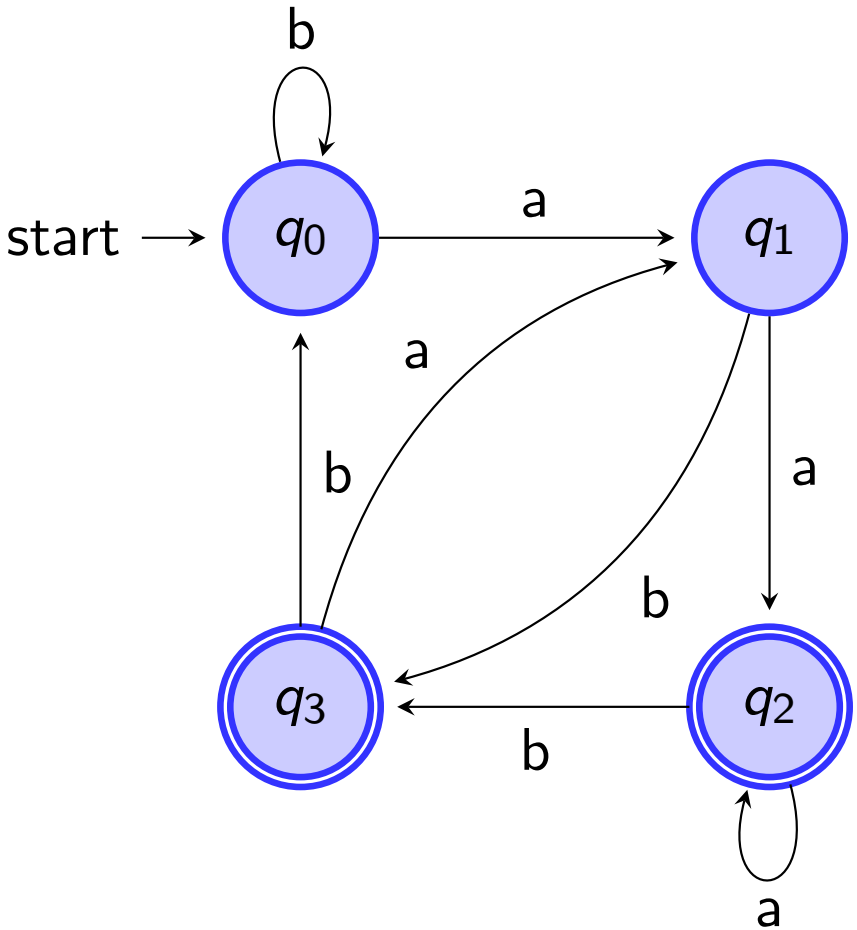


$\delta(q_0, a) = q_0$ albo $= q_1$ - automat niedeterministyczny, δ nie jest funkcją, $\delta(q_0, a) = \{q_0, q_1\}$ - zbiór,
$L(\mathcal{A}) = \{w \in \Sigma^* : \delta(s, w) \cap F \neq \emptyset\}$,
słowo jest rozpoznane jeżeli "uda się znaleć" przejście od startu s (tu q_0) do zbioru stanów końcowych F (tu q_2)

Automat deterministyczny z niedeterministycznego

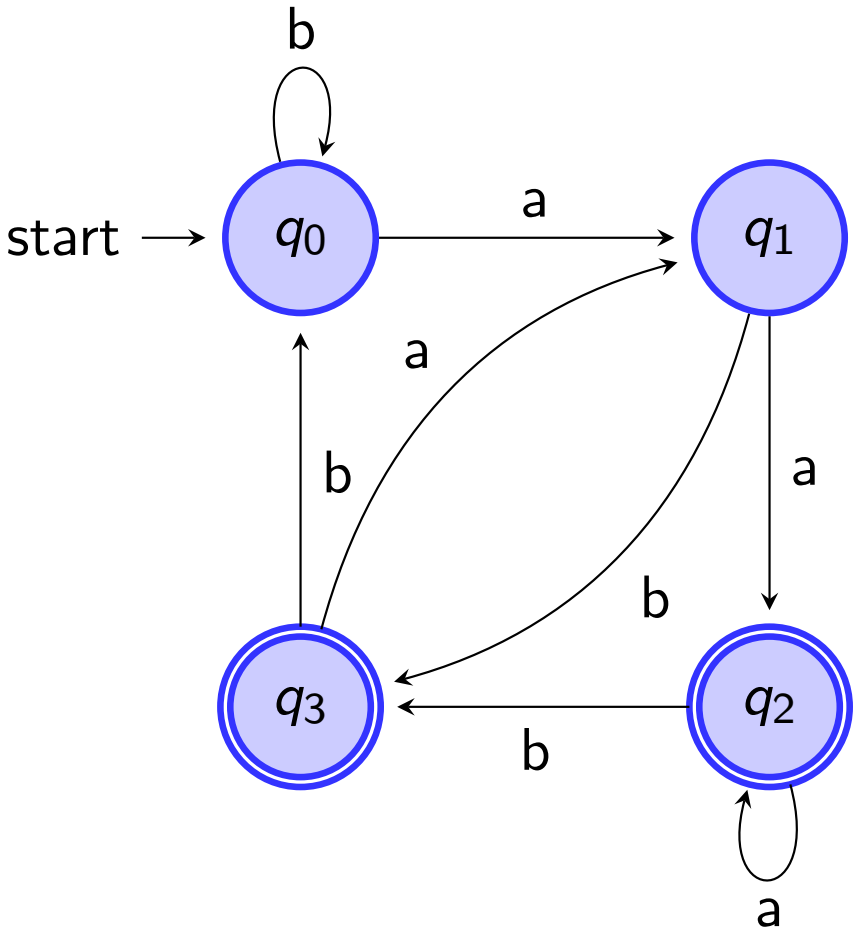


Automat deterministyczny z niedeterministycznego



$L(\mathcal{A}) = \{w \in \Sigma^* : \delta(s, w) \in F\}$, gdzie $F = \{q_2, q_3\}$,

Automat deterministyczny z niedeterministycznego



$L(\mathcal{A}) = \{w \in \Sigma^* : \delta(s, w) \in F\}$, gdzie $F = \{q_2, q_3\}$,
$L(\mathcal{A}) = \{w \in \Sigma^* : \text{w ma drugi symbol od końca } = a\}$.

Automaty skończone

Twierdzenie.
Istnieje procedura (algorytm) pozwalająca dla każdego automatu skończonego niedeterministycznego $\mathcal{A}$ zbudować automat skończony deterministyczny $\mathcal{A}'$, który rozpoznaje ten sam język, tzn. $L(\mathcal{A}) = L(\mathcal{A}')$.

Automaty skończone

Twierdzenie.
Istnieje procedura (algorytm) pozwalająca dla każdego automatu skończonego niedeterministycznego $\mathcal{A}$ zbudować automat skończony deterministyczny $\mathcal{A}'$, który rozpoznaje ten sam język, tzn. $L(\mathcal{A}) = L(\mathcal{A}')$.

Nie za darmo: jeżeli $\mathcal{A}$ ma n elementów, to deterministyczny $\mathcal{A}'$ może mieć do 2^n elementów.

Automaty skończone

Twierdzenie.
Istnieje procedura (algorytm) pozwalająca dla każdego automatu skończonego niedeterministycznego $\mathcal{A}$ zbudować automat skończony deterministyczny $\mathcal{A}'$, który rozpoznaje ten sam język, tzn. $L(\mathcal{A}) = L(\mathcal{A}')$.

Nie za darmo: jeżeli $\mathcal{A}$ ma n elementów, to deterministyczny $\mathcal{A}'$ może mieć do 2^n elementów.

Istnieją procedury pozwalające znaleźć automat skończony minimalny (ze względu na liczbę elementów).

Twierdzenie Kleenego

Fakt 1. Każdy język skończony jest regularny, tj. posiada automat skończony, który go rozpoznaje.

Lemat 1. Jeżeli $L_1 = L(\mathcal{A}_1)$ i $L_2 = L(\mathcal{A}_2)$ oraz $\mathcal{A}_1$, $\mathcal{A}_2$ są deterministyczne, to można efektywnie zbudować automat deterministyczny $\mathcal{A}_3$ taki, że
$L_1 \cup L_2 = L(\mathcal{A}_3)$.

Twierdzenie Kleenego

Lemat 2. Jeżeli $L_1 = L(\mathcal{A}_1)$ i $L_2 = L(\mathcal{A}_2)$ oraz $\mathcal{A}_1$, $\mathcal{A}_2$ są deterministyczne, to można efektywnie zbudować automat (na ogół niedeterministyczny) $\mathcal{A}_3$ taki, że $L_1 \star L_2 = L(\mathcal{A}_3)$.

Twierdzenie Kleenego

Lemat 2. Jeżeli $L_1 = L(\mathcal{A}_1)$ i $L_2 = L(\mathcal{A}_2)$ oraz $\mathcal{A}_1$, $\mathcal{A}_2$ są deterministyczne, to można efektywnie zbudować automat (na ogół niedeterministyczny) $\mathcal{A}_3$ taki, że $L_1 \star L_2 = L(\mathcal{A}_3)$.

Lemat 3. Jeżeli $L_= L(\mathcal{A})$ i $\mathcal{A}$ jest deterministyczny, to można efektywnie zbudować automat (na ogół niedeterministyczny) $\mathcal{A}_0$ taki, że $L^* = L(\mathcal{A}_0)$.

Twierdzenie Kleenego

Lemat 2. Jeżeli $L_1 = L(\mathcal{A}_1)$ i $L_2 = L(\mathcal{A}_2)$ oraz $\mathcal{A}_1$, $\mathcal{A}_2$ są deterministyczne, to można efektywnie zbudować automat (na ogół niedeterministyczny) $\mathcal{A}_3$ taki, że $L_1 \star L_2 = L(\mathcal{A}_3)$.

Lemat 3. Jeżeli $L_= L(\mathcal{A})$ i $\mathcal{A}$ jest deterministyczny, to można efektywnie zbudować automat (na ogół niedeterministyczny) $\mathcal{A}_0$ taki, że $L^* = L(\mathcal{A}_0)$.

Wniosek. Jeżeli język da się zbudować ze skończonej ilości języków skończonych, przy pomocy operacji $\cup, \star, ^*$, to jest on regularny, tzn. można efektywnie zbudować automat (na ogół niedeterministyczny), ktory go rozpoznaje.

Twierdzenie Kleenego

Lemat 2. Jeżeli $L_1 = L(\mathcal{A}_1)$ i $L_2 = L(\mathcal{A}_2)$ oraz $\mathcal{A}_1$, $\mathcal{A}_2$ są deterministyczne, to można efektywnie zbudować automat (na ogół niedeterministyczny) $\mathcal{A}_3$ taki, że $L_1 \star L_2 = L(\mathcal{A}_3)$.

Lemat 3. Jeżeli $L_= L(\mathcal{A})$ i $\mathcal{A}$ jest deterministyczny, to można efektywnie zbudować automat (na ogół niedeterministyczny) $\mathcal{A}_0$ taki, że $L^* = L(\mathcal{A}_0)$.

Wniosek. Jeżeli język da się zbudować ze skończonej ilości języków skończonych, przy pomocy operacji $\cup, \star, ^*$, to jest on regularny, tzn. można efektywnie zbudować automat (na ogół niedeterministyczny), ktory go rozpoznaje.
Twierdzenie Kleenego mówi, ze jest też na odwrót - że wszystkie języki regularne można tak otrzymać.

Twierdzenie Kleenego

Twierdzenie Kleenego.
Język jest zbudowany ze skończonej ilości języków skończonych, przy pomocy operacji $\cup . \star, ^*$,
wtedy i tylko wtedy, gdy
jest on regularny, tzn. można efektywnie zbudować automat (na ogół niedeterministyczny), ktory go rozpoznaje.

Twierdzenie Kleenego

Twierdzenie Kleenego.
Język jest zbudowany ze skończonej ilości języków skończonych, przy pomocy operacji $\cup.\star,^*$,
wtedy i tylko wtedy, gdy
jest on regularny, tzn. można efektywnie zbudować automat (na ogół niedeterministyczny), ktory go rozpoznaje.

Przykłady.
$L(\mathcal{A}) = \{w \in \Sigma^* : w \text{ zawiera choć raz literę } a\} = \{b\}^*\{a\}\{a,b\}^*$,

$L(\mathcal{A}) = \{w \in \Sigma^* : \text{w ma drugi symbol od końca } = a\} =$
$\{a,b\}^*\{a\}\{a\} \cup \{a,b\}^*\{a\}\{b\} = \{a,b\}^*\{a\}\{a,b\}$.

Języki nieregularne, bezkontekstowe

Palindrom to słowo $w = a_1 \dots a_n$ takie, że $w^R = a_n \dots a_1$, czytane od końca jest takie samo jak czytane od początku: $w = w^R$.

Języki nieregularne, bezkontekstowe

Palindrom to słowo $w = a_1 \dots a_n$ takie, że $w^R = a_n \dots a_1$, czytane od końca jest takie samo jak czytane od początku: $w = w^R$.

oko, ala, abba, możejutrotadamadatortujeżom, devillived itp.

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

Języki nieregularne, bezkontekstowe

Palindrom to słowo $w = a_1 \dots a_n$ takie, że $w^R = a_n \dots a_1$, czytane od końca jest takie samo jak czytane od początku: $w = w^R$.

oko, ala, abba, możejutrotadamadatortujeżom, devillived itp.

$\Sigma = \{a, b\}$
Palindromy (parzyste) np. $abba$, aa, bb,

Języki nieregularne, bezkontekstowe

Palindrom to słowo $w = a_1 \dots a_n$ takie, że $w^R = a_n \dots a_1$, czytane od końca jest takie samo jak czytane od początku: $w = w^R$.

oko, ala, abba, możejutrotadamadatortujeżom, devillived itp.

$\Sigma = \{a, b\}$
Palindromy (parzyste) np. $abba$, aa, bb,
Palindromy (nieparzyste) np.: a, aba, aaa, bab, bbb,

Języki nieregularne, bezkontekstowe

Palindrom to słowo $w = a_1 \dots a_n$ takie, że $w^R = a_n \dots a_1$, czytane od końca jest takie samo jak czytane od początku: $w = w^R$.

oko, ala, abba, możejutrotadamadatortujeżom, devillived itp.

$\Sigma = \{a, b\}$
Palindromy (parzyste) np. $abba$, aa, bb,
Palindromy (nieparzyste) np.: a, aba, aaa, bab, bbb,

Palindromy nie tworzą języka regularnego, ale język bezkontekstowy, tj. generowany przez gramatykę bezkontekstową (Noam Chomsky).
Podobnie język $\{a^n b^n : n = 1, 2, \dots\}$
$ab, aabb, aaabbb, \dots$ czyli $ab, a^2b^2, a^3b^3, \dots$

Języki nieregularne, bezkontekstowe

Palindrom to słowo $w = a_1 \dots a_n$ takie, że $w^R = a_n \dots a_1$, czytane od końca jest takie samo jak czytane od początku: $w = w^R$.

oko, ala, abba, możejutrotadamadatortujeżom, devillived itp.

$\Sigma = \{a, b\}$
Palindromy (parzyste) np. $abba$, aa, bb,
Palindromy (nieparzyste) np.: a, aba, aaa, bab, bbb,

Palindromy nie tworzą języka regularnego, ale język bezkontekstowy, tj. generowany przez gramatykę bezkontekstową (Noam Chomsky).
Podobnie język $\{a^n b^n : n = 1, 2, \dots\}$
$ab, aabb, aaabbb, \dots$ czyli $ab, a^2b^2, a^3b^3, \dots$

Automat skończony tu nie wystarcza, trzeba dodać tzw. stos, automaty ze stosem rozpoznają palindromy i $\{a^n b^n : n = 1, 2, \dots\}$.

Języki nieregularne, bezkontekstowe

Gramatyki bezkontekstowe (N. Chomsky)

Języki nieregularne, bezkontekstowe

Gramatyki bezkontekstowe (N. Chomsky)

Przykład: język $\{a^n b^n : n = 1, 2, \dots\}$ można generować gramatyką bezkontekstową.

Języki nieregularne, bezkontekstowe

Gramatyki bezkontekstowe (N. Chomsky)

Przykład: język $\{a^n b^n : n = 1, 2, \dots\}$ można generować gramatyką bezkontekstową.
Reguły (produkcje):
1. $S \to aSb$
2. $S \to \varepsilon$

Języki nieregularne, bezkontekstowe

Gramatyki bezkontekstowe (N. Chomsky)

Przykład: język $\{a^n b^n : n = 1, 2, \dots\}$ można generować gramatyką bezkontekstową.
Reguły (produkcje):
1. $S \to aSb$
2. $S \to \varepsilon$

Np. $S \to aSb \to ab$, 1, 2.

Języki nieregularne, bezkontekstowe

Gramatyki bezkontekstowe (N. Chomsky)

Przykład: język $\{a^n b^n : n = 1, 2, \dots\}$ można generować gramatyką bezkontekstową.
Reguły (produkcje):
1. $S \to aSb$
2. $S \to \varepsilon$

Np. $S \to aSb \to ab$, 1, 2.
$S \to aSb \to aaSbb \to aabb$ 1, 1, 2.

Języki nieregularne, bezkontekstowe

Gramatyki bezkontekstowe (N. Chomsky)

Przykład: język $\{a^n b^n : n = 1, 2, \dots\}$ można generować gramatyką bezkontekstową.
Reguły (produkcje):
1. $S \to aSb$
2. $S \to \varepsilon$

Np. $S \to aSb \to ab$, 1, 2.
$S \to aSb \to aaSbb \to aabb$ 1, 1, 2.
$S \to aSb \to aaSbb \to aaaSbbb \to aaabbb$ 1, 1, 1, 2 etc.

Jaki język zrozumie, rozpozna automat (skończony)?
język regularny, tj. zbudowany z języków skończonych, przy pomocy operacji $\cup, \star, ^*$.

Jaki język zrozumie, rozpozna automat (skończony)?
język regularny, tj. zbudowany z języków skończonych, przy pomocy operacji $\cup, \star, ^*$.

Automaty ze stosem rozpoznają języki bezkontekstowe,

Jaki język zrozumie, rozpozna automat (skończony)?
język regularny, tj. zbudowany z języków skończonych, przy pomocy operacji $\cup, \star, ^*$.

Automaty ze stosem rozpoznają języki bezkontekstowe,

Ale to jeszcze daleko do naszgo komputera, ktory jest odpowiednikiem tzw. Maszyny Turinga.

Literatura:


Devlin, Keith, *Żegnaj Kartezjuszu. Rozstanie z logiką...*, Prószyński Warszawa, 1999.


Hopcroft, J.E., Motwani, R. Ullman, J.D. *Wprowadzenie do teorii automatów, języków i obliczeń*, PWN, 2012.


Howie J. M., *Automata and Languages*, Oxford Science Publications, 1991.


Sipser, M., *Wprowadzenie do teorii obliczeń*, WNT, 2009.