Języki, Automaty i Obliczenia. Odcinek 1

Języki regularne i automaty skończone

Damian Niwiński

Październik 2003

1 Podstawowe operacje na językach

Rozważamy skończony zbiór Σ, który nazywamy *alfabetem*, a jego elementy *symbolami*. Skończony ciąg symboli nazywamy *słowem*, w szczególności ciąg pusty jest *słowem pustym* oznaczanym ϵ. Zbiór wszystkich słów nad alfabetem Σ oznaczamy Σ^*. Słowo $w \in \Sigma^*$ można przedstawić $w = (w_1, \ldots, w_n)$ gdzie $w_i \in \Sigma$, dla $i = 1, \ldots, n$. Liczba $n \geq 0$ jest *długością* słowa w oznaczaną $|w|$, zauważmy, że $|\epsilon| = 0$. Dla dwóch słów $w = (w_1, \ldots, w_n)$ i $v = (v_1, \ldots, v_m)$, *konkatenacją* w i v jest słowo

$$wv = (w_1, \ldots, w_n, v_1, \ldots, v_m)$$

Łatwo sprawdzić, że operacja konkatenacji jest łączna, tzn. $(wv)u = w(vu)$, co uzasadnia notację beznawiasową wvu. Utożsamiając słowa jednoliterowe z literami, mamy w szczególności $w = w_1 \ldots w_n$. Zauważmy, że słowo puste jest *elementem neutralnym* operacji konkatenacji, tzn. $w\epsilon = \epsilon w = w$.

Uwaga. W języku algebry mówimy, że zbiór Σ^* z operacją konkatenacji i słowem pustym tworzy strukturę zwaną *monoidem*.

Dla każdego przedstawienia $u = wv$, słowo w jest *prefiksem* (segmentem początkowym) a słowo v *sufiksem* słowa u. Relacja "być prefiksem" jest relacją częściowego porządku, którą będziemy oznaczać po prostu $\leq$. Oczywiście, słowo puste jest elementem najmniejszym w tej relacji.

Zbiory słów nazywamy *językami*. Oprócz zwykłych operacji teoriomnogościowych: sumy, przecięcia i różnicy, będziemy wykonywać nad językami pewne operacje specjalne.

Konkatenacją języków $L, M \subseteq \Sigma^*$ jest język

$$LM = \{wv \,:\, w \in L, v \in M\}$$

Nietrudno zauważyć, że konkatenacja języków jest łączna.

Pytanie. Co jest elementem neutralnym operacji konkatenacji języków?

W notacji zwykle zakłada się, że operacja konkatenacji wiąże silniej niż operacje teoriomnogościowej sumy i przecięcia, np. $AB \cup C$ znaczy tyle samo co $(AB) \cup C$. Zauważmy, że konkatenacja jest *rozdzielna* względem operacji sumy, tzn. dla dowolnych języków K, L, M,

$$K(L \cup M) = KL \cup KM$$

$$(K \cup L)M = KM \cup LM$$

Pytanie. Czy operacja konkatenacji jest rozdzielna względem operacji części wspólnej $\cap$?

Prawo rozdzielności można uogólnić na przypadek, kiedy sumowanie obejmuje dowolną rodzinę języków.

Fakt 1 *Niech $\mathcal{L} \subseteq P(\Sigma^*)$ będzie rodziną języków i niech $M \subseteq \Sigma^*$. Wówczas*

$$M \bigcup \mathcal{L} = \bigcup_{K \in \mathcal{L}} MK$$

$$(\bigcup \mathcal{L})M = \bigcup_{K \in \mathcal{L}} KM$$

$\square$

Z kolei określimy operację *iteracji* zwaną też operacją "gwiazdki"

$$L^* = \bigcup_{n<\omega} L^n$$

gdzie $L^0 = \{\epsilon\}$ i $L^{n+1} = L^n L$. (Symbol ω oznacza pierwszą nieskończoną liczbę porządkową, czyli innymi słowy zbiór liczb naturalnych.)

Ćwiczenie. (a) Czemu jest równe $\emptyset^*$?
(b) Wykazać, że dla dowolnego $L \subseteq \Sigma^*$, $(L^*)^* = L^*$.

O ile nie będzie to prowadziło do nieporozumień, będziemy opuszczać akolady $\{\}$ dla zbiorów jednoelementowych, pisząc np. $abLbaa$ zamiast $\{ab\}L\{ba\}\{a\}$.

Przykład 1 Niech $\Sigma = \{a, b\}$. Zbiór $\{aa\}^*$ obejmuje słowa złożone z parzystej ilości a, $\Sigma^* a \Sigma^*$ jest zbiorem słów zawierających przynajmniej jedno a, natomiast $(\Sigma - \{a\})^*(a(\Sigma - \{a\})^* a(\Sigma - \{a\})^*)^*$ zbiorem słów, w których liczba wystąpień a jest parzysta.

Rodzina języków nad ustalonym alfabetem wraz z operacjami teorio-mnogościowymi oraz operacjami konkatenacji i gwiazdki tworzy pewną strukturę algebraiczną. Jak to często bywa w matematyce, struktura algebraiczna zachęca do budowania i rozwiązywania równań. Udowodnimy teraz pewną własność równań, jaka będzie przydatna później.

Lemat 1 (Arden) *Niech $A, B \subseteq \Sigma^*$, $\epsilon \notin A$. Wówczas równanie*

$$X = AX \cup B$$

ma dokładnie jedno rozwiązanie w $\mathcal{P}(\Sigma^*)$ równe A^*B.

Dowód. Nietrudno sprawdzić, że $A^*B = AA^*B \cup B$, zatem $X = A^*B$ jest rozwiązaniem równania. Z kolei przypuśćmy, że jakiś język M spełnia $M = AM \cup B$, pokażemy, że $M = A^*B$.

"$\subseteq$" Przez indukcję ze względu na $|w|$, pokażemy, że $w \in M \Rightarrow w \in A^*B$. Gdy $w = \epsilon$, to ponieważ $\epsilon \notin A$ i $w \in AM \cup B$, wiec $w \in B \subseteq A^*B$. Gdy $|w| > 0$, to albo $w \in B$, albo $w = vu$, gdzie $v \in A$ i $u \in M$, przy czym $|u| < |w|$, zatem, z założenia indukcyjnego, $u \in A^*B$. W obu przypadkach $w \in A^*B$.

"$\supseteq$" Z rozdzielności konkatenacji względem dowolnej sumy, mamy w szczególności $A^*B = \bigcup_{n<\omega}(A^nB)$. Z faktu, że M spełnia równanie wynika natychmiast, że $B = A^0B \subseteq M$, oraz że $A^nB \subseteq M$ implikuje $A(A^nB) = A^{n+1}B \subseteq M$, co kończy dowód. □

Z kolei rozważymy operacje lewo- i prawostronnych *ilorazów.*

$$L^{-1}M = \{w \,:\, \exists v \in L\, vw \in M\}$$

$$LM^{-1} = \{v \,:\, \exists w \in M\, vw \in L\}$$

Uwaga. Tę notację trzeba rozumieć całościowo, samo wyrażenie L^{-1} nie oznacza żadnego języka.

Przykład 2 Zbiór słów języka L, które zaczynają się od a można przedstawić $a(a^{-1}L)$.

Wprowadzimy teraz bardzo ważną w studiach nad językami operację **podstawienia**. Niech Σ, Γ będą alfabetami i niech $f : \Sigma \to P(\Gamma^*)$ będzie funkcją przyporządkowującą literom z Σ języki nad Γ. Funkcję f rozszerzamy do funkcji $\hat{f} : \Sigma^* \to P(\Gamma^*)$ przez

$$\begin{aligned} \hat{f}(\epsilon) &= \{\epsilon\} \\ \hat{f}(w\sigma) &= \hat{f}(w)f(\sigma) \end{aligned}$$

dla $w \in \Sigma^*$, $\sigma \in \Sigma$. Wreszcie, funkcję $\hat{f}$ rozszerzamy do operacji określanej na językach (oznaczanej tak samo). Mianowicie, dla $L \subseteq \Sigma^*$, kładziemy

$$\hat{f}(L) = \bigcup_{w \in L} \hat{f}(w)$$

Funkcję $\hat{f}$ nazywamy *podstawieniem* indukowanym przez f. Jeżeli, dla każdego $\sigma \in \Sigma$, moc zbioru $f(\sigma)$, $|f(\sigma)| = 1$, podstawienie nazywamy *homomorfizmem.*

Uwaga. Gdy $\Gamma = \Sigma = \{\sigma_1, \ldots, \sigma_n\}$, podstawienie można uważac za $n + 1$ argumentową operację na $P(\Sigma^*)$, która językom $L, M_1, \ldots, M_n$, przyporządkowuje rezultat podstawienia indukowanego przez $\sigma_i \mapsto M_i$, zastosowanego do L.

Przykład 3 Podstawienie indukowane przez $a \mapsto \epsilon$ powoduje "wymazanie" litery a. Podstawienie indukowane przez $x \mapsto \{1, 2, \epsilon\}$, zastosowane do słowa $yx+zx$, daje w rezultacie zbiór

$$\{y1 + z1,\ y1 + z2,\ y1 + z,\ y2 + z1,\ y2 + z2,\ y2 + z,\ y + z1,\ y + z2,\ y + z\}$$

Ćwiczenia

1. Przedstawić język $\{ab\}^*$ jako kombinację języków $W = \{a, b\}^*$, $\{a\}$, $\{b\}$, używając operacji konkatenacji, sumy i różnicy, ale nie używając $*$.

   Uwaga : dla języka $\{aa\}^*$ jest to niemożliwe.

2. Podać przykład, że $L(K \cap M) \neq LK \cap LM$.

3. Dowieść, że dla dowolnych języków L, M, $(L^*M^*)^* = (L \cup M)^*$.

4. Jakie relacje zachodzą pomiędzy zbiorami L^*, $L^{-1}L^*$ i L^*L^{-1}?

5. Czy z faktu, że $LK = M$ wynika, że $L^{-1}M = K$?

6. Czy $(LK^{-1})M = L(K^{-1}M)$?

7. *Kody.* Zbiór $C \subseteq \Sigma^+$ nazywamy *kodem* jeśli każde słowo $w \in \Sigma^*$ dopuszcza co najwyżej jedną faktoryzację względem C (tzn., da się "odkodować"). Niech $\Sigma = \{a, b\}$. Dowieść, że zbiór $\{aa, baa, ba\}$ jest kodem, a zbiór $\{a, ab, ba\}$ nie jest.

2 Języki regularne

2.1 Wyrażenia regularne

Niech $\mathrm{Reg}(\Sigma)$ będzie najmniejszą rodziną języków nad alfabetem Σ, która zawiera zbiór $\emptyset$, dla każdej litery $\sigma \in \Sigma$ zawiera zbiór $\{\sigma\}$, a ponadto, jeśli L i M są językami w $\mathrm{Reg}(\Sigma)$, to również LM, $L \cup M$ i L^* należą do tej rodziny. Zbiory z rodziny $\mathrm{Reg}(\Sigma)$ nazywamy *językami regularnymi* nad Σ. Kiedy dopuszczamy dowolne alfabety, mówimy o *klasie języków regularnych*.

Tradycyjnie, języki regularne są reprezentowane poprzez tzw. *wyrażenia regularne*, podobnie jak np. liczby całkowite mogą być reprezentowane przez wyrażenia arytmetyczne. Wyrażeniami takimi posługiwaliśmy się już nieformalnie przed chwilą, a teraz zdefiniujemy je dokładnie. Z naszego punktu widzenia istotne jest to, że wyrażenia używane do reprezentacji języków, same są słowami nad pewnym alfabetem.

Zbiór *wyrażeń regularnych* nad alfabetem Σ, $R(\Sigma)$, określamy jako najmniejszy zbiór słów nad alfabetem $\Sigma \cup \{\emptyset, +, *, (,)\}$, taki że

- $\emptyset \in R(\Sigma)$,
- dla każdego $\sigma \in \Sigma$, $\sigma \in R(\Sigma)$,
- jeli $r, s \in R(\Sigma)$, to $(rs), (r+s), (r^*) \in R(\Sigma)$

Interpretacją wyrażenia $r \in R(\Sigma)$ jest język $L[r] \subseteq \Sigma^*$ określony indukcyjnie w łatwy do odgadnięcia sposób: $L[\emptyset] = \emptyset$, $L[\sigma] = \{\sigma\}$, $L[(r+s)] = L[r] \cup L[s]$ $L[(rs)] = L[r]\,L[s]$, $L[(r^*)] = L[r]^*$. Jak można oczekiwać, mamy następujący

Fakt 2 *Język $L \subseteq \Sigma^*$ jest regularny wtedy i tylko wtedy, gdy $L = L[r]$, dla pewnego wyrażenia regularnego $r \in R(\Sigma)$.*

Dowód. Łatwo jest widzieć, że klasa języków opisywanych przez wyrażenia regularne nad Σ (tzn. $\{L[r] : r \in R(\Sigma)\}$)

> (*) zawiera zbiory $\emptyset$ i $\{\sigma\}$, dla $\sigma \in \Sigma$, oraz jeśli L i M są w tej klasie, to należą tam również LM, $L \cup M$ i L^*.

A zatem, skoro $\mathrm{Reg}(\Sigma)$ jest najmniejszą rodziną o własności (*), to każdy język regularny jest postaci $L[r]$, dla pewnego $r \in R(\Sigma)$.

Na odwrót, jeśli klasa $\mathcal{K}$ spełnia warunek (*), to przez indukcję ze względu na długość wyrażenia regularnego r, łatwo jest wykazać, że $L[r] \in \mathcal{K}$. Stosując tę uwagę do $\mathcal{K} = \mathrm{Reg}(\Sigma)$, otrzymujemy, że każdy język $L[r]$ jest regularny. □

Uwaga 1. W zastosowaniach praktycznych zbiór wyrażeń regularnych często określa się bardziej liberalnie, dopuszczając opuszczanie nawiasów. Przyjmuje się wówczas, że "*" wiąże silniej niż "+" i konkatenacja, a konkatenacja silniej niż "+". Np. $ab^* + c$ ma taki sam sens jak $((a(b^*)) + c)$.

Uwaga 2. Nietrudno zauważyć, że ten sam język może być reprezentowany przez wiele wyrażeń regularnych. Np. $L[(a^*)] = L[(aa)^* + a(aa)^*]$. Prowadzi to do pytań o reprezentacje w jakimś sensie optymalne, pytań, które czasem okazują się nieoczekiwanie trudne. Np. algorytm znajdowania wyrażenia o minimalnym zagnieżdżeniu operacji * został podany przez Hashigushi'ego dopiero w roku 1988, a dowód poprawności liczy kilkadziesiąt stron. (Dla porównania, omawiane w następnych rozdziałach Twierdzenie Kleene'go o wyrażeniach regularnych pochodzi z r. 1956.)

Jak zobaczymy nieco później, klasa języków regularnych obejmuje wiele naturalnych przykładów języków, również o znaczeniu praktycznym. Z drugiej strony, klasa ta jest zamknięta na wiele operacji, jakie z danych języków tworzą nowy język. Wprost z definicji wynika, że suma, konkatenacja i gwiazdka języków regularnych jest językiem regularnym.

Pokażemy teraz, że również operacja podstawienia nie wyprowadza poza języki regularne.

Fakt 3 *Niech $f : \Sigma \to P(\Gamma^*)$ będzie funkcją taką, że, dla każdego $\sigma \in \Sigma$, $f(\sigma)$ jest językiem regularnym. Wówczas, jeśli $L \subseteq \Sigma^*$ jest językiem regularnym, to $\hat{f}(L)$ jest również językiem regularnym.*

Dowód. Z uwagi na indukcyjną definicję rodziny języków regularnych $\mathrm{Reg}(\Sigma)$, wystarczy wykazać, że rodzina języków regularnych $L \subseteq \Sigma^*$, dla których $\hat{f}(L)$ jest również językiem regularnym, zawiera $\emptyset$ oraz zbiory $\{\sigma\}$, dla $\sigma \in \Sigma$ i jest zamknięta na operacje binarnej sumy, konkatenacji i gwiazdki. Gdy $L = \emptyset$, lub $L = \{\sigma\}$, dla $\sigma \in \Sigma$, teza jest oczywista. Dalej, nietrudno jest sprawdzić, że

$$\begin{aligned} \hat{f}(L_1L_2) &= \hat{f}(L_1)\hat{f}(L_2) \\ \hat{f}(L_1 \cup L_2) &= \hat{f}(L_1) \cup \hat{f}(L_2) \\ \hat{f}(L_1^*) &= \hat{f}(L_1)^* \end{aligned}$$

Zatem, gdy L_1 i L_2 są jezykami regularnymi spełniającymi tezę Faktu, to spełniają ją również L_1L_2, $L_1 \cup L_2$ i L_1^*. Ta uwaga kończy dowód. □

Ćwiczenia

1. Napisać wyrażenie regularne nad alfabetem $\{0,1\}$, opisujące następujący język:
   (a) zbiór słów zawierających dwa kolejne wystąpienia tego samego symbolu,
   (b) zbiór słów nie zawierających dwóch kolejnych wystąpień tego samego symbolu,
   (c) zbiór słów o długości nieparzystej,
   (d) zbiór słów, gdzie po każdej parze kolejnych zer (niekoniecznie bezpośrednio *po*) wystąpi para kolejnych jedynek,
   (e) zbiór słów nie zawierających 101 jako podsłowa,
2. Napisać wyrażenie regularne nad alfabetem $\{0,1\}$, opisujące następujący język:
   (a) zbiór słów zawierających parzystą liczbę jedynek,
   (b) zbiór słów przedstawiających w postaci binarnej liczby naturalne podzielne przez 3.
3. Opisać w języku polskim lub dowolnym języku naturalnym zbiory oznaczone przez wyrażenia regularne:
   (a) $(1+01+001)^*(\epsilon+0+00)$
   (b) $(11+0)^*+((11)^*1+0)^*$
4. Skonstruować wyrażenie regularne nad alfabetem $\{0,1\}^3$, opisujące zbiór takich śłów $(a_1,b_1,c_1),(a_2,b_2,c_2),\ldots,(a_k,b_k,c_k)$, że $a_1a_2\ldots a_k + b_1b_2\ldots b_k = c_1c_2\ldots c_k$, gdzie ciągi $a_1a_2\ldots a_k$, $b_1b_2\ldots b_k$, $c_1c_2\ldots c_k$ traktuje się jako przedstawienia liczb naturalnych w postaci binarnej, ewentualnie poprzedzone zerami. (Jeśli trójki (a,b,c) zapisywać pionowo, to słowo takie przedstawia zwykłe dodawanie pisemne.)
5. *Lustrzanym odbiciem* słowa $w = w_1w_2\ldots w_k$ jest słowo

   $$w^R =_{df} w_kw_{k-1}\ldots w_2w_1$$

   Dla języka L,

   $$L^R =_{df} \{w^R : w \in L\}$$

   Opisać algorytm, który, dla danego wyrażenia regularnego α konstruuje wyrażenie α^R o tej samej długości, takie, że $L[\alpha^R] = (L[\alpha])^R$.
6. (*) Dowieść, że dla dowolnego podzbioru $L \subseteq \{0\}^*$, język L^* jest regularny.

2.2 Automaty skończone

Niedeterministyczny automat skończony (w dalszym ciągu krótko: automat skończony) nad alfabetem Σ jest układem

$$A = (\Sigma, Q, I, \delta, F)$$

gdzie

- Q jest skończonym zbiorem *stanów*,
- $I \subseteq Q$ jest zbiorem stanów *początkowych*,
- $\delta \subseteq Q \times \Sigma \times Q$ jest *relacją przejścia*,
- $F \subseteq Q$ jest zbiorem stanów *akceptujących*.

Elementy relacji δ będziemy nazywać *przejściami* (ang. *transition*), a fakt, że przejście (q, σ, q') należy do δ będziemy także zapisywać $q \xrightarrow{\sigma} q'$. Intuicyjnie, powyższe przejście rozumiemy następująco: jeśli automat znajduje się w stanie q, to po otrzymaniu informacji a (np. po przeczytaniu kodu litery a na taśmie) może zmienić stan na q'. Zauważmy, że stanów q' o tej własności może być wiele, może też nie być żadnego takiego stanu; na tym właśnie polega *niedeterminizm*. Jeśli dla każdego stanu q i litery a, istnieje *dokładnie jeden* stan q', taki że $q \xrightarrow{\sigma} q'$, to automat nazywamy *deterministycznym*[1]. Pojęciem tym zajmiemy się jeszcze w dalszym ciągu wykładu.

Relację przejścia δ rozszerzymy do relacji $\hat{\delta} \subseteq Q \times \Sigma^* \times Q$, określonej jako najmniejszy podzbiór $Q \times \Sigma^* \times Q$, taki że

- $(q, \epsilon, q) \in \hat{\delta}$, dla każdego $q \in Q$,
- jeśli $(q, w, r) \in \hat{\delta}$ i $(r, \sigma, p) \in \delta$, to $(q, w\sigma, p) \in \hat{\delta}$.

Stwierdzenie, że $(q, w, q') \in \hat{\delta}$, będziemy także zapisywać $q \xrightarrow{w} q'$. (Jeśli $w = \sigma$ jest słowem złożonym z jednej litery, to notacja ta pokrywa się z notacją wprowadzoną wcześniej dla przejść, bo $(q, \sigma, q') \in \hat{\delta} \Leftrightarrow (q, \sigma, q') \in \delta$.)

Relację $\hat{\delta}$ możemy scharakteryzować w sposób bardziej dynamiczny, odwołując się do możliwych ciągów kolejnych przejść, jakie automat może wykonać, przyjmując informację zawartą w dostarczonym mu słowie. Dowód poniższej charakteryzacji można łatwo przeprowadzić przez indukcję ze względu na długość słowa. Szczegóły pozostawiamy Czytelniczce lub Czytelnikowi.

Fakt 4 *Niech $q, q' \in Q$, $w \in \Sigma^*$. Wówczas $q \xrightarrow{w} q'$ wtedy i tylko wtedy, gdy $w = \epsilon$ i $q = q'$ lub $w = \sigma_1 \ldots \sigma_n$, gdzie $n > 0$ i $\sigma_i \in \Sigma$, dla $i = 1, \ldots, n$, i istnieje ciąg stanów $q_0, q_1, \ldots, q_n$, taki że*

$$q = q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \ \ldots \ q_{n-2} \xrightarrow{\sigma_{n-1}} q_{n-1} \xrightarrow{\sigma_n} q_n = q'$$

□

Ciąg przejść w postaci o jakiej mowa w powyższym fakcie, tzn.

$$(q_0, \sigma_1, q_1), (q_1, \sigma_2, q_2), \ldots, (q_{n-2}, \sigma_{n-1}, q_{n-1}), (q_{n-1}, \sigma_n, q_n)$$

lub, w notacji "strzałkowej"

$$q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \ \ldots \ q_{n-2} \xrightarrow{\sigma_{n-1}} q_{n-1} \xrightarrow{\sigma_n} q_n$$

[1] W niektórych opracowaniach można spotkać słabszą definicję, w której *dokładnie jeden* jest zastąpione przez *co najwyżej jeden*. Automat deterministyczny w tym słabszym sensie łatwo jest sprowadzić do wymaganej przez nas postaci dokładając co najwyżej jeden stan („czarną dziurę").

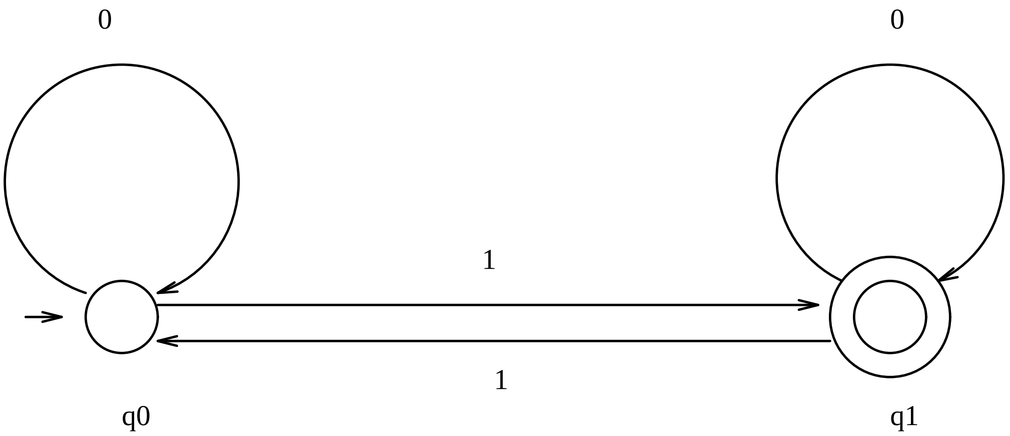
nazwiemy *przebiegiem* automatu A, na słowie $\sigma_1\sigma_2\ldots\sigma_n$, startującym ze stanu q_0 i zakończonym w stanie q_n.

Przebieg nazwiemy *obliczeniem*, gdy jego stan startowy należy do I. Obliczenie jest *akceptujące*, gdy jego stan końcowy należy do F. Słowo $w \in \Sigma^*$ jest *akceptowane* (czasem mówi się także : rozpoznawane) przez automat A, jeśli *istnieje* obliczenie akceptujące na słowie w (nie wykluczamy, że automat niedeterministyczny może mieć na słowie akceptowanym również obliczenia nie akceptujące). Zauważmy, że słowo ϵ jest akceptowane wtedy i tylko wtedy, gdy $I \cap F \neq \emptyset$. Zbiór słów akceptowanych przez A stanowi *język akceptowany* przez A, oznaczany $L(A)$. Mamy więc

$$L(A) = \{w \in \Sigma^* \,:\, q \stackrel{w}{\to} q' \text{ dla pewnych } q \in I,\, q' \in F\}$$

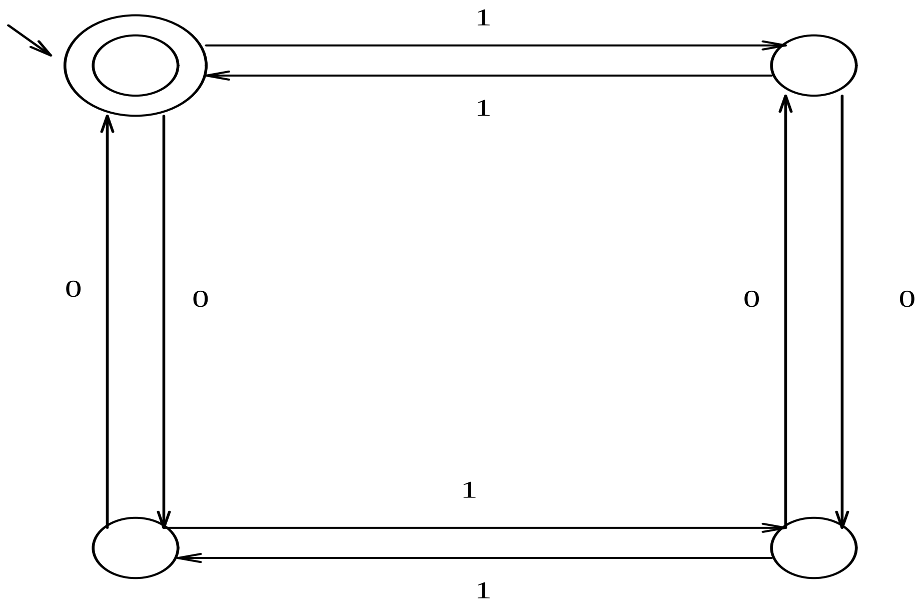
Mówimy, że język $L \subseteq \Sigma^*$ jest *rozpoznawalny* przez automat skończony (po angielsku: *finite–state recognizable*), jeśli $L = L(A)$ dla pewnego automatu A.

Przykład 4 Rozważmy automat $A = (\Sigma, Q, I, \delta, F)$, *gdzie* $\Sigma = \{0,1\}$, $Q = \{q_0, q_1\}$, $I = \{q_0\}$, $\delta = \{(q_0,0,q_0),(q_0,1,q_1),(q_1,0,q_1),(q_1,1,q_0)\}$, $F = \{q_1\}$.



Nietrudno wykazać, że $L(A)$ jest zbiorem słów, które zawierają nieparzystą liczbę jedynek.

Przykład 5 Niech $A = (\{0,1\}, \{q_0,q_1,q_2,q_3\}, \{q_0\}, \delta, \{q_0\})$ będzie automatem, którego relacja (w tym przypadku funkcja) δ przedstawiona jest na rysunku.



$L(A)$ jest zbiorem tych słów zero-jedynkowych, w których zarówno 0 jak i 1 występują parzystą ilość razy.

Ćwiczenia

1. Skonstruować automat nad alfabetem $\{0, 1, \ldots, 8, 9\}$ rozpoznający
   (a) zbiór słów reprezentujących liczby podzielne przez 7,
   (b) zbiór słów, których lustrzane odbicia są podzielne przez 7.
   (c) Skonstruować automat *deterministyczny* rozpoznający zbiór z poprzedniego punktu.
   (d) Uogólnić niniejsze zadanie.
2. Skonstruować automat niedeterministyczny rozpoznający zbiór słów nad alfabetem $\{a, b\}$ o długości co najmniej 2, takich, że przedostatnim symbolem jest a. Skonstruować automat deterministyczny rozpoznający ten sam język.
3. Narysować automat rozpoznający zbiór słów o następującej własności : pewne dwa a są rozdzielone słowem o długości $3k$, dla pewnego k.
4. Dla danego automatu A, skonstruować automat rozpoznający język $(L(A))^R$.
5. Dla danych automatów A i B, skonstruować automat rozpoznający język $L(A) \cap L(B)$.
6. Dla danych automatów A i B, skonstruować automaty rozpoznające języki $(L(A))^{-1}L(B)$ i $L(A)(L(B))^{-1}$.
7. Wykazać, że dla dowolnego automatu A nad alfabetem Σ i *dowolnego języka* $X \subseteq \Sigma^*$, istnieją automaty rozpoznające języki $X^{-1}L(A)$ i $L(A)X^{-1}$.

Uwaga. Dowód istnienia odpowiednich automatów nie może być tutaj konstruktuwny, gdyż zbiór X nie jest zadany w sposób efektywny.

8. Dla danego automatu A rozpoznającego język L, skonstruować automat rozpoznający język
$$Cycle(L) = \{vu \, : \, uv \in L\}$$

2.3 Twierdzenie Kleene'go

Naszym celem będzie teraz wykazanie, że automaty skończone rozpoznają dokładnie języki regularne. Będzie to pierwszy w tym wykładzie rezultat pokazujący równoważność dwóch modeli obliczeń: w tym wypadku wyrażeń regularnych i automatów skończonych. Podkreślmy, że, teraz i w przyszłości, interesować nas będzie nie tylko samo *istnienie* jakiegoś obiektu (np. wyrażenia regularnego opisującego język rozpoznawalny przez automat), ale także **algorytm** konstruowania tego obiektu i ewentualnie złożoność tego algorytmu.

Twierdzenie 1 *Dla każdego wyrażenia regularnego można skonstruować niedeterministyczny automat skończony, który rozpoznaje język opisany przez wyrażenie. Co więcej, automat można dobrać tak, że $|Q| \leq 2|\alpha|$, gdzie α oznacza wyrażenie, a Q zbiór stanów automatu.*

Dowód rozpoczniemy od lematu.

Lemat 2 *Dla dowolnych automatów $A = (\Sigma, Q^A, I^A, \delta^A, F^A)$ i $B = (\Sigma, Q^B, I^B, \delta^B, F^B)$ można skonstruować automaty C, D i E, takie, że $L(C) = L(A) \cup L(B)$, $L(D) = L(A)L(B)$ i $L(E) = (L(A))^*$, przy czym $|Q^C| \leq |Q^A| + |Q^B|$, $|Q^D| \leq |Q^A| + |Q^B|$, $|Q^C| \leq |Q^A| + 1$.*

Dowód Lematu. Ograniczymy się tu do podania konstrukcji, dowód ich poprawności pozostawiając Czytelnikowi.

Możemy założyć, że zbiory stanów Q^A i Q^B są rozłączne.

Automat C $Q^C = Q^A \cup Q^B$, $I^C = I^A \cup I^B$, $F^C = F^A \cup F^B$, $\delta^C = \delta^A \cup \delta^B$.

Automat D $Q^D = Q^A \cup Q^B$, $I^D = I^A$,

$$F^D = \begin{cases} F^A \cup F^B & \text{gdy } I^B \cap F^B \neq \emptyset \\ F^B & \text{w przec. przyp.} \end{cases}$$

$$\delta^C = \delta^A \cup \delta^B \cup \{(q, \sigma, p) \, : \, q \in F^A \, \wedge \, (\exists r \in I^B)(r, \sigma, p) \in \delta^B\}.$$

Automat E $Q^E = Q^A \cup \{\tilde{q}\}$, gdzie $\tilde{q}$ jest nowym stanem, $I^E = I^A \cup \{\tilde{q}\}$, $F^E = F^A \cup \{\tilde{q}\}$, $\delta^E = \delta^A \cup \{(q_f, \sigma, q) \, : \, q_f \in F \wedge (\exists p \in I^A)(p, \sigma, q) \in \delta^A\}$.

Dowód Twierdzenia 1 prowadzimy przez indukcję ze względu na złożoność wyrażenia regularnego, α. Dla $\alpha = \emptyset$ lub $\alpha = \sigma$, z łatwością można podać automat z dwoma stanami rozpoznający $L[\alpha]$. Z kolei załóżmy, że mamy już automaty rozpoznające $L[\alpha]$ i $L[\beta]$ o odpowiednio $\leq 2|\alpha|$ i $\leq 2|\beta|$ stanach. Wówczas z lematu otrzymujemy

- automat dla $L[(\alpha + \beta)]$ o liczbie stanów $\leq 2|\alpha| + 2|\beta| \leq 2|(\alpha + \beta)|$,

- automat dla $L[(\alpha\beta)]$ o liczbie stanów $\leq 2|\alpha| + 2|\beta| \leq 2|(\alpha\beta)|$,
- automat dla $L[(\alpha^*)]$ o liczbie stanów $\leq 2|\alpha| + 1 \leq 2|(\alpha^*)|$.

Na podstawie powyższych obserwacji nietrudno jest zaprojektować algorytm, który rekurencyjnie rozkłada wyrażenie regularne na wyrażenia prostsze, znajduje dla nich automaty, a następnie syntetyzuje z nich automat dla wyrażenia oryginalnego za pomocą konstrukcji C,D lub E z dowodu Lematu 2. □

Pokażemy teraz, że transformacja odwrotna, tzn. automatu w wyrażenie regularne, jest również wykonalna, choć dowód jest tu nieco trudniejszy. Właśnie teraz wykorzystamy wprowadzone już wcześniej pojęcie równania w algebrze języków.

Niech $A = (\Sigma, Q, I, \delta, F)$ będzie automatem. Dla każdego $q \in Q$, niech $L(A, q)$ będzie zbiorem słow akceptowanych przez automat, który różni się od A co najwyżej tym, że jego jedynym stanem początkowym jest q, tzn.

$$L(A, q) = \{w \,:\, (\exists p \in F)\, q \xrightarrow{w} p\}$$

Z automatem A zwiążemy teraz pewien układ równań, $E(A)$, w którym, jako niewiadomych używać będziemy zmiennych ze zbioru $\{X_q : q \in Q\}$. Jak się późniaj okaże, zbiory $L(A, q)$ będą stanowić (jedyne) rozwiązanie tego układu.

Niech $q \in Q$ i niech $\delta(q)$ będzie zbiorem wszystkich przejść o początku q, tj. przejść postaci (q, σ, q'), $q' \in Q, \sigma \in \Sigma$. Ponadto niech

$$\mathrm{Empty}(q) = \begin{cases} \epsilon & \text{gdy } q \in F \\ \emptyset & \text{w przec. przyp.} \end{cases}$$

Określamy równanie $E(q)$ następująco. Jeśli zbiór przejść $\delta(q)$ jest pusty, to

$$E(q) \,:\, X_q = \mathrm{Empty}(q)$$

jeśli zaś $\delta(q) = \{(q, \sigma_1, q_1), \ldots, (q, \sigma_m, q_m)\}$, to

$$E(q) \,:\, X_q = \sigma_{q_1} X_1 \cup \ldots \cup \sigma_m X_{q_m} \cup \mathrm{Empty}(q)$$

Niech $E(A)$ będzie zbiorem wszystkich otrzymanych w ten sposób równań,

$$E(A) \;=\; \{E(q) \,:\, q \in Q\}$$

Dowód następującego lematu pozostawiamy jako łatwe ćwiczenie.

Lemat 3 *Podstawienie $L(A, q)$ za X_q, dla $q \in Q$, jest rozwiązaniem układu równań $E(A)$ w dziedzinie $P(\Sigma^*)$.* □

Ćwiczenie. Udowodnić, że opisane powyżej podstawienie jest *jedynym* rozwiązaniem układu.

Możemy już udowodnić

Twierdzenie 2 *Istnieje algorytm, który dla dowolnego automatu skończonego A, produkuje wyrażenie regularne opisujące język $L(A)$.*

Dowód. Niech $E(A)$ będzie układem równań stowarzyszonym z A, określonym jak wyżej. Układ ten będziemy transformować do kolejnych postaci $E(A) = E_0(A), E_1(A), \ldots, E_i(A), \ldots$, zachowując następujące niezmienniki:

- dla każdego $q \in Q$, w układzie $E_i(A)$ jest dokładnie jedno równanie, którego lewą stroną jest X_q, przy czym jest ono postaci

$$X_q = \alpha_1 X_{q_1} \cup \ldots \cup \alpha_m X_{q_m} \cup \beta$$

  gdzie $\alpha_1, \ldots, \alpha_m, \beta$ są wyrażeniami regularnymi nad Σ, oraz $\epsilon \notin L[\alpha_i]$, dla $i = 1, \ldots, m$ (zakładamy, że stany $q_1, \ldots, q_m$ są różne),

- podstawienie $X_q \leftarrow L(A, q)$ jest rozwiązaniem układu $E_i(A)$, tzn. dla każdego równania, w opisanej wyżej postaci, równość

$$L(A, q) = L[\alpha_1]L(A, q_1) \cup \ldots \cup L[\alpha_m]L(A, q_m) \cup L[\beta]$$

  jest spełniona.

Z Lematu 3 warunki te są spełnione przez początkowy układ $E(A)$. Pokażemy, że po skończonej ilości kroków i przy zachowaniu niezmienników osiągniemy układ, w którym każde równanie będzie w *postaci rozwiązanej*, tzn. postaci $X_q = \beta$. Wobec drugiego niezmiennika otrzymamy $L(A, q) = L[\beta]$.

Algorytm jest następujący (będziemy używali tutaj zmiennej E, której wartościami są układy równań):

Na starcie, E staje się układem $E(A)$. Dalej, jeśli E nie jest jeszcze w postaci rozwiązanej, wybieramy dowolną zmienną występującą po prawej stronie któregoś z równań, powiedzmy X_q, i rozważamy równanie $X_q = R$. (Takie równanie jest w naszym układzie zgodnie z pierwszym niezmiennikiem.)

Z kolei rozważamy dwa przypadki.

- Jeżeli zmienna X_q występuje w R, to równanie to możemy przedstawić (używając przemienności $\cup$) $X_q = \alpha X_q \cup R_1$, gdzie R_1 nie zawiera X_q. Wówczas zastępujemy je równaniem $X_q = (\alpha^*)R_1$, a następnie na wszystkie wystąpienia zmiennej X_q w pozostałych równaniach podstawiamy $(\alpha^*)R_1$. W razie potrzeby, porządkujemy prawe strony równań, tak by każda zmienna występowała co najwyżej raz, używając prawa rozdzielności $AX \cup BX = (A + B)X$, zapewniając w ten sposób pierwszy niezmiennik. Zauważmy, że po tym kroku zmienna X_q zniknęła z prawych stron równań. Niezmiennik drugi jest spełniony przez nowe równanie $X_q = (\alpha^*)R_1$ na mocy Lematu Ardena, a dla pozostałych równań jest oczywistą konsekwencją.

- Jeśli zmienna X_q nie występuje po prawej stronie równania $X_q = R$, to wszystkie wystąpienia X_q na prawych stronach pozostałych równań zastępujemy przez R. Podobnie jak poprzednio wykonujemy ewentualne porządkowanie dla utrzymania równań w standardowej postaci. (Również i po tej operacji, ilość zmiennych występujących po prawej stronie równań zmniejszyła się o 1. Niezmienniki zostały oczywiście zachowane.)

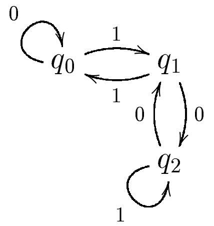
Jak wynika z komentarzy poczynionych w trakcie opisu algorytmu, układ E osiągnie w końcu postać rozwiązaną, $\{X_q = \beta_q : q \in Q\}$, i to po nie więcej niż $|Q|$ obrotach pętli (za każdym razem, z prawych stron równań znika jedna zmienna). Niezmiennik drugi daje nam więc przedstawienie $L(A, q) = L[\beta_q]$, dla każdego $q \in Q$. Oczywiście,

$$L = \bigcup_{q \in I} L[\beta_q]$$

poszukiwanym wyrażeniem regularnym dla L jest więc $\sum_{q\in I} \beta_q$. □

Uwaga. Analiza powyższego algorytmu pokazuje, że konstruowane wyrażenie osiąga rozmiar rzędu $2^{O(|Q|)}$. Najczęściej nie jest to wyrażenie najkrótsze. Jednak w pesymistycznych przypadkach, wyniku tego nie można poprawić[2].

Przykład 6 Rozważmy automat



gdzie q_0 jest stanem początkowym i akceptującym. Automat ten rozpoznaje liczby podzielne przez 3 zapisane w systemie binarnym (dopuszczamy prefiks zer). Dokładniej, automat przyjmie stan q_i jeśli przeczytana dotąd liczba daje resztę i z dzielenia przez 3.

Postępując jak w dowodzie Twierdzenia 2, otrzymujemy najpierw układ równań

$$\begin{aligned} X_{q_0} &= \varepsilon \cup 0X_{q_0} \cup 1X_{q_1} \\ X_{q_1} &= 1X_{q_0} \cup 0X_{q_2} \\ X_{q_2} &= 0X_{q_1} \cup 1X_{q_2} \end{aligned}$$

Stosując krok algorytmu oparty na Lemacie Ardena do równania dla X_{q_2}, otrzymujemy

$$X_{q_2} = 1^*0X_{q_1}$$

skąd po podstawieniu do równania dla X_{q_1} i ponownym uproszczeniu mamy

$$\begin{aligned} X_{q_1} &= 1X_{q_0} \cup 01^*0X_{q_1} \\ &= (01^*0)^*1X_{q_0} \end{aligned}$$

skąd wreszcie

$$\begin{aligned} X_{q_0} &= \varepsilon \cup 0X_{q_0} \cup 1(01^*0)^*1X_{q_0} \\ &= \varepsilon \cup (0 + 1(01^*0)^*1)X_{q_0} \\ &= (0 + 1(01^*0)^*1)^* \end{aligned}$$

co daje nam poszukiwane wyrażenie.

[2] Zob. pracę A. Ehrenheucht, P. Zeiger, *Complexity Measures for Regular Expressions*, Journal of Computer and System Science **12**, 134–146 (1976).

Twierdzenia 1 i 2 możemy podsumować w następującej równoważności.

Twierdzenie 3 (Kleene) *Język jest rozpoznawalny przez automat skończony wtedy i tylko wtedy, gdy jest regularny.*

Ćwiczenia

1. Skonstruować wyrażenia regularne reprezentujące języki z przykładów 4 i 5.

2. Niech L będzie językiem regularnym. Dowieść, że następujące języki są również regularne:

   - $\frac{1}{2}L =_{df} \{w \,:\, (\exists u)\, |u| = |w| \wedge wu \in L\}$
   - $\sqrt{L} =_{df} \{w \,:\, ww \in L\}$

3. (*) Niech L będzie językiem regularnym. Dowieść, że następujące języki są również regularne:

   (a) $\mathrm{Root}(L) = \{w \,:\, (\exists n \in N)\, w^n \in L\}$

   (b) $\mathrm{Sqrt}(L) = \{w \,:\, (\exists u)|u| = |w|^2 \wedge wu \in L\}$
   Wskazówka. $n^2 = 1 + 3 + \ldots (2n - 1)$.

   (c) $\mathrm{Log}(L) = \{w \,:\, (\exists u)|u| = 2^{|w|} \wedge wu \in L\}$
   Wskazówka. $2^n = 1 + 2 + 2^2 + 2^{n-1} + 1$.

   (d) $\mathrm{Fibb}(L) = \{w \,:\, (\exists u)|u| = F_{|w|} \wedge wu \in L\}$
   gdzie F_n jest n-tą liczbą Fibonacciego tzn.

   $F_1 = F_2 = 1$
   $F_{n+2} = F_n + F_{n+1}$

4. Niech L bedzie językiem regularnym nad alfabetem $\{0, 1\}$. Dowieść, że następujący język jest regularny:

   $\{w \,:\, w \in L \wedge$ spośród słów o dlugości $|w|$, w jest najmniejsze w porządku leksykograficznym$\}$

5. (*) Przyjmujemy, że każde niepuste słowo binarne $w \in \{0, 1\}^*$, $w = w_1 \ldots w_k$, reprezentuje pewien ułamek w przedziale $[0, 1)$,

   $$\mathrm{bin}(w) = w_1\frac{1}{2} + w_2\frac{1}{2^2} + \ldots + w_k\frac{1}{2^k}$$

   Dla liczby rzeczywistej $r \in [0, 1]$, niech

   $$L_r = \{w \,:\, \mathrm{bin}(w) \le r\}$$

   Dowieść, że język L_r jest regularny wtedy i tylko wtedy, gdy liczba r jest wymierna.

6. Udowodnić wspomniane wyżej górne oszacowanie na rozmiar wyrażenia regularnego, jakie konstruuje algorytm z dowodu Twierdzenia 2.

2.4 Kryterium nieregularności

Jak wykazać, że jakiś język nie jest regularny?

Matematyczny dowód *nieistnienia* jakiegoś obiektu (w naszym przypadku: automatu rozpoznającego język) wymaga często głębokiej analizy problemu, nawet jeśli sam fakt wydaje się intuicyjnie oczywisty. Pomocne może być wskazanie jakiejś własności przysługującej wszystkim językom regularnym — brak takiej własności wyklucza regularność języka.

Podamy teraz przykład takiej własności.

Lemat 4 (o pompowaniu) *Niech $A = (\Sigma, Q, I, \delta, F)$ będzie automatem skończonym, $|Q| = n$. Jeśli automat akceptuje słowo w o długości $|w| \geq n$, to istnieją słowa w', v, w'', takie że $w = w'vw''$, $|w'v| \leq n$, $v \neq \epsilon$, oraz, dla każdego $m = 0, 1, 2, \ldots$, $w'v^m w'' \in L(A)$.*

Dowód. Przypuśćmy, że $p \stackrel{w}{\to} q$, gdzie $p \in I$, $q \in F$ i $w = w_1 w_2 \ldots w_k$. Z charakteryzacji relacji $\hat{\delta}$ (Fakt 4), mamy przebieg

$$p = q_0 \stackrel{w_1}{\to} q_1 \stackrel{w_2}{\to} q_2 \ldots q_{k-2} \stackrel{w_{k-1}}{\to} q_{k-1} \stackrel{w_k}{\to} q_k = q$$

Jeśli teraz $k \geq |Q|$, to istnieją i, j, $1 \leq i < j \leq n$, takie że $q_i = q_j$. Połóżmy $w' = w_1 \ldots w_i$ (ϵ, jeśli $i = 0$), $v = w_{i+1} \ldots w_j$, $w'' = w_{j+1} \ldots w_k$. Oczywiście, założenia o długościach $w'v$ i v są spełnione. Przyjmując $q_i = q_j = \tilde{q}$, mamy więc $p \stackrel{w'}{\to} \tilde{q}$, $\tilde{q} \stackrel{u}{\to} \tilde{q}$ i $\tilde{q} \stackrel{w''}{\to} q$. Stąd, stosując ponownie charakteryzację z Faktu 4, nietrudno otrzymać, że $p \stackrel{w'w''}{\to} q$, a także $p \stackrel{w'vvw''}{\to} q$, $p \stackrel{w'vvvw''}{\to} q$, itd. , ogólnie $p \stackrel{w'v^m w''}{\to} q$, dla każdego $m = 0, 1, \ldots$.
□

Zauważmy, że, z Lematu o pompowaniu, jeśli A akceptuje pewne słowo w o długości nie mniejszej niż n, to A akceptuje także słowo $w'w''$, gdzie $|w'w''| < |w|$. Stosując ten argument dostatecznie wiele razy, otrzymujemy następujący fakt.

Wniosek 1 *Niech $A = (\Sigma, Q, I, \delta, F)$ będzie automatem skończonym, $|Q| = n$. Jeśli automat A akceptuje jakieś słowo, to również akceptuje pewne słowo o długości* mniejszej *niż n.* □

Przykład 7 (dowód nieregularności) Niech $\Sigma = \{a, b\}$ i $L = \{a^n b^n : n \in N\}$. Język L nie jest regularny. Istotnie, przypuśćmy przeciwnie, że L jest akceptowany przez automat o p stanach. Z Lematu o pompowaniu, słowo $a^p b^p$ można przedstawić jako uvw' gdzie $|uv| \leq p$, a zatem $u = a^i$, $v = a^j$ i $w' = a^k b^p$, gdzie $i + j + k = p$, przy czym $j > 0$. Z tezy lematu, również słowo $a^i a^j a^j a^k b^p = a^{p+j} b^p \in L$, wbrew definicji języka L.

Ćwiczenia

1. Dowieść, że zbiór palindromów nie jest regularny.

2. Dowieść, że zbiór wyrażeń regularnych nie jest regularny.

3. Dowieść, że następujące języki nie są regularne:

   - $\{a^{2^n} : n \in N\}$

- $\{a^p : p \text{ jest liczbą pierwszą}\}$
- $\{a^i b^j : \mathrm{nwd}(i,j) = 1\}$

4. Niech L będzie językiem regularnym. Dowieść, że języki:

   - $L_{+--} = \{w : (\exists u)\, |u| = 2|w| \wedge wu \in L\}$
   - $L_{++-} = \{w : (\exists u)\, 2|u| = |w| \wedge wu \in L\}$
   - $L_{-+-} = \{w : (\exists u, v)\, |u| = |v| = |w| \wedge uwv \in L\}$

   są językami regularnymi, a język

   - $L_{+-+} = \{uv : (\exists w)\, |u| = |v| = |w| \wedge uwv \in L\}$

   nie jest regularny.

5. Czy z faktu, że język $\mathit{Cycle}(L) = \{vu : uv \in L\}$ jest regularny, można wnioskować, że język L jest regularny?

6. Słowo nad alfabetem jednoliterowym można utożsamić z jego długością. Zatem język nad takim alfabetem można utożsamić ze zbiorem liczb naturalnych. Dowieść, że zbiór liczb naturalnych jest regularny wtedy i tylko wtedy, gdy można go przedstawić jako sumę skończonej liczby zbiorów postaci $\{a + b \cdot n : n \in \mathbf{N}\}$ (taki zbiór nazywa się semi–liniowym).

2.5 Problemy decyzyjne i algorytmy

Problem 1.
Dane: Automat $A = (\Sigma, Q, I, \delta, F)$ i słowo $w \in \Sigma^*$, $w = w_1 \dots w_k$.
Pytanie: Czy $w \in L(A)$?
Dla każdego $\sigma \in \Sigma$, określmy operację na zbiorach stanów $Y \subseteq Q$

$$\mathit{Step}_\sigma(Y) = \{q : (\exists p \in Y) p \xrightarrow{\sigma} q\}$$

Rozważmy następujący algorytm:

Algorytm 1
Wejście: słowo w i automat A
$X := I$;
Dla $i = 1, \dots, |w|$ **wykonuj**
$X := \mathit{Step}_{w_i}(X)$;
Wynik: wartość logiczna relacji $X \cap F \neq \emptyset$

Nietrudno sprawdzić, że algorytm istotnie rozwiązuje Problem 1. Dokładniej, po i-krotnym wykonaniu instrukcji pętli, wartością zmiennej X jest zbiór $\{q : (\exists p \in I) p \xrightarrow{w_1 \dots w_i} q\}$. Czas wykonania operacji Step_σ jest rzędu $O(|Q|^2)$, tak więc łaczny czas działania algorytmu można oszacować[3] przez $O|(w||Q|^2)$.

[3] Lepszym oszacowaniem dla operacji *Step* jest $|Q|m$, gdzie m jest maksymalną liczbą strzałek o danej etykiecie wychodzących z dowolnego stanu; wówczas łaczny czas działania algorytmu jest $O(|w|m|Q|)$.

Problem 2.
Dane: Automat $A = (\Sigma, Q, I, \delta, F)$ i słowo $w \in \Sigma^*$.
Pytanie: Czy w zawiera *prefiks* w $L(A)$?
Algorytm 2
Dane: słowo w i automat A
1. $X := I$; $i := 0$;
2. **Dopóki** $(X \cap F = \emptyset \wedge i < |w|)$, **wykonuj**
    $i := i + 1$;
    $X := Step_{w_i}(X)$;
Wynik: wartość logiczna relacji $X \cap F \neq \emptyset$

Po wykonaniu tego algorytmu, jeśli wynikiem jest prawda, a wartością zmiennej i jest $\tilde{i}$, to $w_1 \dots w_{\tilde{i}}$ jest pierwszym prefiksem w należącym do języka $L(A)$.

Algorytm 2 znajduje zastosowanie w konkretnych sytuacjach, w których interesuje nas wyszukanie danego wzorca we większym tekście, przy czym wzorzec może być zadany wyrażeniem regularnym. Sytuacja taka występuje w edytorach tekstu, pojawia się także w pierwszej fazie kompilowania, jaką jest tzw. analiza leksykalna. Np. instrukcja zastąpienia w tekście bloków symboli b ("blank") przez jeden symbol, wymaga najpierw znalezienia podsłów opisanych wyrażeniem $b(b^*)$. Przypuśćmy, że w tekście t szukamy wzorca opisanego wyrażeniem regularnym α. Problem sprowadza się do pytania, czy t zawiera prefiks należący do języka $(\Sigma)^* L[\alpha]$. Na podstawie Twierdzenia 1, możemy skonstruować automat (niedetreministyczny) A o $O(|\alpha|)$ stanach, rozpoznający język $(\Sigma)^* L[\alpha]$. Następnie, wystarczy zastosować Algorytm 2 dla tego automatu i słowa t.[4]

Bardzo naturalny jest tzw. problem niepustości, czyli jest pytanie, czy automat akceptuje w ogóle jakieś słowo.

Problem 3.
Dane: Automat $A = (\Sigma, Q, I, \delta, F)$
Pytanie: Czy $L(A) \neq \emptyset$?

Tym razem algorytm poprzedzimy dokładniejszą analizą problemu. Wygodnie będzie wprowadzić następujące pojęcie. Powiemy, że stan q jest *osiągalny* ze stanu p, jeśli istnieje $w \in \Sigma^*$, takie że $p \stackrel{w}{\to} q$. Podobnie, stan q jest *osiągalny ze zbioru stanów* E, jeśli jest osiągalny ze pewnego stanu $p \in E$. Zauważmy, że $L(A) \neq \emptyset$ wtedy i tylko wtedy, gdy część wspólna zbioru F i zbioru stanów osiągalnych z I jest niepusta. Wobec tego zadanie można sprowadzić do obliczenia tego ostatniego zbioru. W tym celu trzeba umieć, dla danych stanów p, q, rozstrzygnąć, czy $(\exists w \in \Sigma^*)\, p \stackrel{w}{\to} q$. (Jeżeli automat przedstawiamy jako graf, własność ta polega na istnieniu jakiejkolwiek ścieżki z wierzchołka p do q.) Pewna trudność wynika z faktu, że *a priori* nie ograniczamy tu długości słowa w. Jednakże, z lematu o pompowaniu (Lemat 4, por. także uwagę przed Wnioskiem 1) wiemy już, że jeśli $p \stackrel{w}{\to} q$, to istnieje v, $|v| < |Q|$, takie że $p \stackrel{v}{\to} q$.

Niech teraz operacja *Step* działająca na podzbiorach zbioru stanów Q będzie określona

[4] Nasze rozumowanie prowadzi do algorytmu działającego w czasie $O(|t||\alpha|^2)$. W rzeczywistości, algorytm można ulepszyć do liniowego względem $|\alpha|$, tj. $O(|t||\alpha|)$, por. Aho, Hopcroft,Ullman, *Projektowanie i analiza algorytmów komputerowych*, PWN 1983, str.414.

następująco:

$$Step(Y) = \bigcup_{\sigma\in\Sigma} Step_\sigma(Y)$$

Możemy już przedstawić algorytm.

Algorytm 3
Dane: Automat A
1. $X := \emptyset; X_1 := I;$
2. **Dopóki** $(X_1 \neq \emptyset)$ **wykonuj**
3. $\quad X := X \cup X_1;$
4. $\quad X_1 := Step(X_1) \setminus X;$

Wynik: wartość logiczna relacji $X \cap F \neq \emptyset$

Zauważmy, że linia 3 może być wykonana co najwyżej $|Q|$ razy (za każdym razem dokładamy coś do X), zatem algorytm na pewno się zatrzymuje po wykonaniu nie więcej niż $|Q|$ obrotów pętli 2. Dalej, nietrudno jest wykazać, przez indukcję ze względu na k, $k = 0, 1, \ldots$, że po k-krotnym wykonaniu pętli 2, wartością zmiennej X jest zbiór wszystkich stanów, które są osiągalne z I w mniej niż k krokach, tzn. zbiór $\{q : (\exists p \in I)(\exists w \in \Sigma^*)(p \stackrel{w}{\to} q) \wedge |w| < k\}$, a wartością zmiennej X_1 jest zbiór tych stanów, które są osiągalne z I po raz pierwszy w dokładnie k krokach. Zatem, z wniosku z lematu o pompowaniu (Wniosek 1) wynika, że końcową wartością zmiennej X jest zbiór wszystkich stanów osiągalnych z I, tzn. zbiór $\{q : (\exists p \in I)(\exists w \in \Sigma^*)p \stackrel{w}{\to} q\}$.

Czas wykonania operacji *Step* jest rzędu $O(|Q|^2)$, tak więc łaczny czas działania powyższego algorytmu można oszacować przez $O(|Q|^3)$. Przedstawiliśmy ten algorytm ze względu na jego prosty schemat; nie jest to jednak algorytm optymalny. Czytelnik zauważył być może, że zamiast obliczania kolejnych iteracji funkcji *Step*, możemy po prostu przeszukiwać graf automatu algorytmem przeszukiwania wgłąb (DFS), nie zwracając uwagi na etykiety krawędzi, dopóki nie na trafimy na stan akceptujący. Osiągamy w ten sposób złożoność $O(|Q| + |\delta|)$.

Problem 4.
Dane: Automat $A = (\Sigma, Q, I, \delta, F)$
Pytanie: Czy $L(A)$ jest nieskończony?
Wybieramy jedno z kilku możliwych podejść do tego problemu.

Powiemy, że stan p jest *produktywny*, jeśli jakiś stan akceptujący $q \in F$ jest osiągalny z p.

Powiemy, że stan p jest *ściśle produktywny*, jeśli jakiś stan akceptujący $q \in F$ jest osiągalny z p poprzez słowo niepuste (tj. $(\exists q \in F)(\exists w \neq \epsilon)\, p \stackrel{w}{\to} q$).

Zauważmy, że stan produktywny a nie ściśle produktywny musi być akceptujący.

Zbiór stanów produktywnych można wyznaczyć stosując wariant algorytmu 3 dla automatu *dualnego* A^{-1}, tj. automatu z odwróconą relacją przejścia

$$(q, \sigma, p) \in \delta^{-1} \Leftrightarrow (p, \sigma, q) \in \delta$$

Nietrudno zauważyć, że zbiór stanów produktywnych automatu A to zbiór stanów osiągalnych z F w automacie A^{-1}. Zbiór stanów ściśle produktywnych można wyznaczyć bardzo podobnie (rozważając, zamiast F, stany “o jeden krok przed F”); szczegóły pozostawiamy Czytelnikowi.

Z kolei powiemy, że stan p jest *ograniczony* jeśli zbiór $\{w \in \Sigma^* : (\exists q \in F) p \stackrel{w}{\to} q\}$ jest skończony. Oczywiście, $L(A)$ jest nieskończony wtedy i tylko wtedy, gdy przynajmniej jeden stan w I nie jest ograniczony. Zadanie sprowadza się więc do wyznaczenia zbioru stanów ograniczonych i porównania go z I.

Wygodnie będzie wprowadzić operację w pewnym sensie dualną do operacji *Step*,

$$BackStep(Y) = \{p \,:\, (\forall \sigma \in \Sigma)(\forall q \in Q)(p \stackrel{\sigma}{\to} q) \Rightarrow q \in Y\}$$

Poniższy algorytm, strukturalnie bardzo podobny do Algorytmu 3, oblicza zbiór stanów ograniczonych automatu A i odpowiada na pytanie, czy $L(A)$ jest nieskończony. Zakładamy przy tym, na podstawie uwag powyżej, że mamy już procedurę wyznaczającą zbiór stanów ściśle produktywnych.

Algorytm 4

Dane: Automat A
1. $X := \emptyset$; $X_1 :=$ *zbiór stanów, które* **nie** *są ściśle produktywne*;
2. **Dopóki** $(X_1 \neq \emptyset)$ **wykonuj**
3. $\quad X := X \cup X_1$;
4. $\quad X_1 := BackStep(X_1) \setminus X$;

Wynik: wartość logiczna relacji $\neg(I \subseteq X)$

Tu również, podobnie jak w algorytmie 3, linia 3 może być wykonana co najwyżej $|Q|$ razy, zatem algorytm na pewno się zatrzymuje po wykonaniu nie więcej niż $|Q|$ obrotów pętli 2. Dalej, przez indukcję ze względu na k, $k = 0, 1, \ldots$, można wykazać, że po k-krotnym wykonaniu pętli 2, wartością zmiennej X jest zbiór tych stanów p, że każde słowo akceptowane z tego stanu (tzn. takie słowo w, że $p \stackrel{w}{\to} q$ dla pewnego $q \in F$) jest długości mniejszej niż k, podczas gdy wartością zmiennej X_1 jest zbiór tych stanów p, że najdłuższe słowo akceptowane z p ma długość dokładnie k. Zatem, wartością końcową zmiennej X jest zbiór tych stanów p, że najdłuższe słowo akceptowane z tego stanu ma długość mniejszą niż $|Q|$. Z lematu o pompowaniu wiemy jednak, że jeśli z jakiegoś stanu automat akceptuje słowo o długości równej lub większej od $|Q|$, to akceptuje z tego stanu nieskończenie wiele słów. Tak więc, wartością końcową zmiennej X w naszym algorytmie jest dokładnie zbiór stanów ograniczonych automatu A.

Tu również czas działania algorytmu można oszacować przez $O(|Q|^3)$. Ulepszenie algorytmu pozostawiamy Czytelnikowi.

Ćwiczenia

1. Zaprojektować algorytm, który dla danych automatów deterministycznych A, B, rozstrzyga, czy $L(A) = L(B)$.

2. Niech A będzie ustalonym automatem deterministycznym. Zaprojektować algorytm, który dla danej liczby n oblicza ilość słów długości n akceptowanych przez A.

Literatura

- A.V.Aho, J.E.Hopcroft, J.D.Ullman, *Projektowanie i analiza algorytmów komputerowych*, w serii: *Biblioteka Informatyki*, Państwowe Wydawnictwo Naukowe, Warszawa 1983.
- J.E.Hopcroft, J.D.Ullman, *Wprowadzenie do teorii automatów, języków i obliczeń*, Wydawnictwo Naukowe PWN, Warszawa 1994.
- A.Blikle, *Automaty i gramatyki. Wstęp do lingwistyki matematycznej*, Państwowe Wydawnictwo Naukowe, Warszawa 1971.
- T.H.Cormen, Ch.E.Leiserson, R.L.Rivest, *Wprowadzenie do algorytmów*, Wydawnictwa Naukowo–Techniczne, Warszawa 1998.