Języki, Automaty i Obliczenia. Odcinek 1

# Języki regularne i automaty skończone

Damian Niwiński

Październik 2003

## 1 Podstawowe operacje na językach

Rozważamy skończony zbiór $\Sigma$, który nazywamy *alfabetem*, a jego elementy *symbolami*. Skończony ciąg symboli nazywamy *słowem*, w szczególności ciąg pusty jest *słowem pustym* oznaczanym $\epsilon$. Zbiór wszystkich słów nad alfabetem $\Sigma$ oznaczamy $\Sigma^*$. 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 $\Sigma^*$ 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*

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

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

□

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 $\omega$ 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 $\Sigma, \Gamma$ będą alfabetami i niech $f : \Sigma \to P(\Gamma^*)$ będzie funkcją przyporządkowującą literom z $\Sigma$ języki nad $\Gamma$. 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 $\Sigma$, 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 $\Sigma$. 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 $\Sigma$, $R(\Sigma)$, 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 $\Sigma$ (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

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

Zatem, gdy $L_1$ i $L_2$ są jezykami regularnymi spełniającymi tezę Faktu, to spełniają ją również $L_1 L_2$, $L_1 \cup L_2$ 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 słów $(a_1,b_1,c_1),(a_2,b_2,c_2),\ldots,(a_k,b_k,c_k)$, 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 $\alpha$ konstruuje wyrażenie $\alpha^R$ 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 $\Sigma$ 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 $\delta$ będziemy nazywać *przejściami* (ang. *transition*), a fakt, że przejście $(q, \sigma, q')$ należy do $\delta$ 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 $\delta$ 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 $\epsilon$ 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

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

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) $\delta$ 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 $\Sigma$ 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 $\alpha$ 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, $\alpha$. 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 $A$ związemy 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 $(q, \sigma, q')$, $q' \in Q, \sigma \in \Sigma$. Ponadto niech

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

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

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

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

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

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 $\Sigma$, 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.)

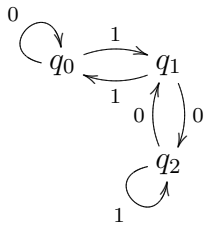
Jak wynika z komentarzy poczynionych w trakcie opisu algorytmu, układ $E$ osiągnie w końcu postać rozwiązaną, $\{X_q = \beta_q : q \in Q\}$, i to po nie więcej niż $|Q|$ obrotach pętli (za każdym razem, z prawych stron równań znika jedna zmienna). Niezmiennik drugi daje nam więc przedstawienie $L(A,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

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

Jeśli teraz $k \geq |Q|$, to istnieją $i, j$, $1 \leq i < j \leq n$, takie że $q_i = q_j$. Połóżmy $w' = w_1 \ldots w_i$ ($\epsilon$, 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 $p \xrightarrow{w'} \tilde{q}$, $\tilde{q} \xrightarrow{u} \tilde{q}$ i $\tilde{q} \xrightarrow{w''} q$. Stąd, stosując ponownie charakteryzację z Faktu 4, nietrudno otrzymać, że $p \xrightarrow{w'w''} q$, a także $p \xrightarrow{w'vvw''} q$, $p \xrightarrow{w'vvvw''} q$, itd. , ogólnie $p \xrightarrow{w'v^m w''} q$, 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:
   - $\{a^{2^n} : n \in N\}$

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

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 $Cycle(L) = \{vu : uv \in L\}$ 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 $Y \subseteq Q$

$$Step_\sigma(Y) = \{q : (\exists p \in Y) p \xrightarrow{\sigma} q\}$$

Rozważmy następujący algorytm:

**Algorytm 1**
**Wejście:** słowo $w$ i automat $A$
$X := I$;
**Dla** $i = 1, \dots, |w|$ **wykonuj**
$X := Step_{w_i}(X)$;
**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 $Step_\sigma$ 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 $\alpha$. 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 $p \xrightarrow{w} q$. 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 $(\exists w \in \Sigma^*)\, p \xrightarrow{w} q$. (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 $p \xrightarrow{w} q$, to istnieje $v$, $|v| < |Q|$, takie że $p \xrightarrow{v} q$.

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 $\{q : (\exists p \in I)(\exists w \in \Sigma^*)(p \xrightarrow{w} q) \wedge |w| < k\}$, 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 $\{q : (\exists p \in I)(\exists w \in \Sigma^*)p \xrightarrow{w} q\}$.

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. $(\exists q \in F)(\exists w \neq \epsilon)\, p \xrightarrow{w} q$ ).

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 $\{w \in \Sigma^* : (\exists q \in F) p \xrightarrow{w} q\}$ 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*,

$$\mathit{BackStep}(Y) = \{p : (\forall \sigma \in \Sigma)(\forall q \in Q)(p \xrightarrow{\sigma} q) \Rightarrow q \in Y\}$$

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. $\quad X_1 := \mathit{BackStep}(X_1) \setminus X$;

**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 $p \xrightarrow{w} q$ 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.

Języki, Automaty i Obliczenia. Odcinek 2

# Optymalizacja automatu skończonego. Własności języków regularnych.

Damian Niwiński

Październik 2003

## 1 Optymalizacja automatu skończonego

### 1.1 Determinizacja

O automatach $A$ i $B$ powiemy, że są *równoważne*, jeśli akceptują ten sam język, tzn. $L(A) = L(B)$.

Automat $A = (\Sigma, Q, I, \delta, F)$ nazwiemy *deterministycznym*, jeśli ma on dokładnie jeden stan początkowy, a relacja $\delta$ jest dobrze określoną *funkcją* z $Q \times \Sigma$ w $Q$, tzn., dla każdych $q, \sigma$, istnieje *dokładnie jedno* $q'$ takie, że $(q, \sigma, q') \in \delta$. W takim przypadku możemy pisać $\delta(q, \sigma) = q'$ zamiast $(q, \sigma, q') \in \delta$. Podobnie, jeśli zbiór stanów początkowych jest singletonem, powiedzmy, $I = \{q_I\}$, to zwykle opuszczamy nawiasy $\{\}$, przedstawiając automat $A = (\Sigma, Q, q_I, \delta, F)$.

Zauważmy, że w automacie deterministycznym, dla każdego stanu $q$ i każdego słowa $w$, istnieje dokładnie jeden przebieg automatu na słowie $w$ startujący ze stanu $q$; w szczególności istnieje dokładnie jedno obliczenie automatu na $w$ (tj. przebieg startujący ze stanu początkowego). Tak więc, również relacja $\hat{\delta}$ (okrełona w Odcinku 1, w punkcie 2.2) jest wtedy funkcją, $\hat{\delta} : Q \times \Sigma^* \to Q$, która ponadto spełnia następujące równania rekurencyjne

$$\hat{\delta}(q_I, \epsilon) = q_I \quad (1)$$

$$\hat{\delta}(q_I, w\sigma) = \delta(\hat{\delta}(q_I, w), \sigma) \quad (2)$$

Wartość $\hat{\delta}(q_I, w)$ jest właśnie tym stanem, jaki automat osiąga w obliczeniu na słowie $w$.

O deterministycznym automacie skończonym możemy myśleć jak o maszynie realizującej pewien algorytm, mianowicie algorytm rozpoznawania słów języka $L(A)$. Przykłady automatów deterministycznych widzieliśmy już w Odcinku 1 (Przykłady 4,5,6).

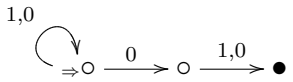
**Przykład 1** Niech $\Sigma = \{0, 1, \ldots, 9\}$ i niech $L$ będzie zbiorem słów stanowiących przedstawienia w systemie dziesiętnym liczb podzielnych przez 3. Język ten jest rozpoznawany przez następujący automat $A = (\Sigma, Q, s, \delta, F)$.

- $Q = \{s, q_0, q_1, q_2\}$

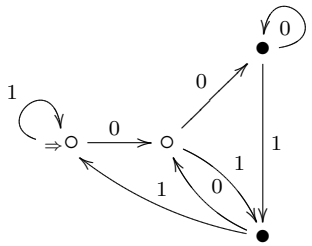
- $q_I = s$
- $\delta(s,d) = q_{d \bmod 3}$, $\delta(q_i,d) = q_{(i+d) \bmod 3}$, *dla* $d \in \Sigma, i = 0,1,2$
- $F = \{q_0\}$

**Ćwiczenie.** Uogólnić powyższy przykład, konstruując automat dla języka $L_{k,m}$, złożonego ze słów, które w systemie o podstawie $k$ reprezentują liczby naturalne podzielne przez $m$.

**Przykład 2** Zbiór słów nad alfabetem $\{0,1\}$, takich że przedostatnia cyfra jest 0 można łatwo rozpoznawać automatem niedeterministycznym:




Konstrukcja automatu deterministycznego dla tego samego zadania jest bardziej skomplikowana:




Okazuje się, że determinizacja automatu jest zawsze możliwa, choć na ogół kosztem wykładniczego wzrostu liczby stanów.

**Twierdzenie 1** *Dla każdego automatu skończonego o $n$ stanach, można skonstruować równoważny mu automat deterministyczny o nie więcej niż $2^n$ stanach.*

*Dowód.* Niech $A = (\Sigma, Q, I, \delta, F)$ będzie dowolnym niedeterministycznym automatem skończonym. Idea jest następująca: deterministyczny automat $A'$ symulujący $A$ powinien, w miarę czytania kolejnych liter słowa $w$ mieć informację o wszystkich stanach, do jakich mógłby dojść w tym miejscu automat $A$. Zauważmy, że jeśli już mamy taką informację dla jakiegoś słowa $w$, to nietrudno ją "zaktualizować" dla przedłużenia $w$ o literę $\sigma \in \Sigma$, $w\sigma$, itd.

Dokładniej, określamy deterministyczny automat $A' = (\Sigma, P(Q), I, \delta', F')$, którego stanami są zbiory stanów automatu $A$, stanem początkowym jest zbiór stanów początkowych automatu $A$, oraz

- $\delta'(X,\sigma) = \{q : (\exists p \in X)(p,\sigma,q) \in \delta\}$, dla $X \in P(Q)$, $\sigma \in \Sigma$.
- $F' = \{X : X \cap F \neq \emptyset\}$

Dla dowodu, że $L(A') = L(A)$, wystarczy pokazać, że

dla każdego $w \in \Sigma^*$, $\hat{\delta}'(I,w) = \{q \in Q : (\exists p \in I) p \xrightarrow{w} q\}$

Stosując dla funkcji $\hat{\delta}'$ równania rekurencyjne (1) i (2) ze str. 1, otrzymujemy

$$\begin{aligned}\hat{\delta}'(I,\epsilon) &= I \\ &= \{q \in Q : (\exists p \in I) p \xrightarrow{\epsilon} q\}\end{aligned}$$

oraz

$$\begin{aligned}\hat{\delta}'(I,w\sigma) &= \delta'(\hat{\delta}'(I,w),\sigma) \\ &= \{q : (\exists q' \in \hat{\delta}'(I,w))(q',\sigma,q) \in \delta\} \\ &\quad \text{(z definicji } \delta') \\ &= \{q : (\exists q' \in Q,\, \exists p \in I) p \xrightarrow{w} q' \xrightarrow{\sigma} q\} \\ &\quad \text{( z założenia indukcyjnego)} \\ &= \{q : (\exists p \in I) p \xrightarrow{w\sigma} q\}\end{aligned}$$

Ta uwaga kończy dowód. □

Udowodnione przed chwilą twierdzenie zastosujemy do wykazania ważnej własności klasy języków regularnych.

**Twierdzenie 2** *Rodzina języków regularnych nad alfabetem $\Sigma$ jest zamknięta na operacje binarnej sumy, przecięcia (tzn. części wspólnej) i różnicy.*

*Dowód.* Zamkniętość na sumę wynika wprost z definicji języków regularnych. Ze względu na zależności pomiędzy pozostałymi operacjami, wystarczy wykazać, że uzupełnienie $\Sigma^* - L$ języka regularnego $L$ jest językiem regularnym. Niech $L = L(A)$ dla deterministycznego automatu $A = (\Sigma, Q, I, \delta, F)$. Tworzymy automat $A'$, który różni się od $A$ tylko tym, że rolę stanów akceptujących przejmują teraz stany nie-akceptujące $A$, tzn. $F' = Q - F$. Nietrudno sprawdzić, że $L(A') = \Sigma^* - L$. □

**Ćwiczenie.** Podać *explicite* konstrukcję automatu rozpoznającego część wspólną $L(A) \cap L(B)$. Wskazówka: za zbiór stanów nowego automatu przyjąć $Q^A \times Q^B$.

## 1.2 Twierdzenie Myhilla-Nerode'a

Widzieliśmy już do tej pory, że język regularny może być określony na różne sposoby: poprzez wyrażenie regularne, poprzez niedeterministyczny automat skończony i wreszcie poprzez deterministyczny automat skończony. Udowodnimy teraz bardzo ważne twierdzenie, ukazujące, że *regularność* języka może być scharakteryzowana poprzez cechy strukturalne samego języka, bez odwoływania się do obiektów „z zewnątrz", które definiują język, takich jak wyrażenia regularne lub automaty. Niejako produktem ubocznym tego twierdzenia będzie konstrukcja automatu deterministycznego o minimalnej liczbie stanów.

Dla dowolnego języka $L \subseteq \Sigma^*$, określamy binarną relację $\sim_L$ na $\Sigma^*$ następująco: dla każdych $u, w \in \Sigma^*$

$$\begin{aligned}u \sim_L w &\Leftrightarrow_{def} \{u\}^{-1}L = \{w\}^{-1}L \\ &\Leftrightarrow (\forall v \in \Sigma^*)(uv \in L \Leftrightarrow wv \in L)\end{aligned}$$

**Ćwiczenie.** Sprawdzić, że $\sim_L$ jest relacją równoważności.

Klasę abstrakcji słowa $w$ w relacji $\sim_L$ będziemy oznaczać $[w]_L$. Odnotujmy dwie użyteczne własności relacji $\sim_L$.

**Lemat 1** *1.* $(\forall u, w \in \Sigma^*)(\forall \sigma \in \Sigma) u \sim_L w \Rightarrow u\sigma \sim_L w\sigma$

*2.* $L = \bigcup_{w \in L} [w]_L$. □

*Dowód.* Dla pierwszego punktu zauważmy, że $\{u\sigma\}^{-1}L = \{\sigma\}^{-1}(\{u\}^{-1}L)$, podobnie dla $w$. Zatem równość ilorazów $\{u\}^{-1}L = \{w\}^{-1}L$ implikuje $\{u\sigma\}^{-1}L = \{w\sigma\}^{-1}L$.

W punkcie 2, inkluzja "$\subseteq$" jest oczywista. Dla dowodu inkluzji przeciwnej, wystarczy wykazać, że $w \sim u$ i $w \in L$ implikuje $u \in L$, co wynika z definicji $\sim_L$ przy $v = \epsilon$. □

Przypomnijmy, że *Indeksem* relacji równoważności nazywamy liczbę jej klas abstrakcji.

**Przykład 3** Niech $\Sigma = \{0, 1\}$. Niech $\#_x(w)$ oznacza liczbę wystąpień symbolu $x$ w słowie $w$. *(a)* Niech $L$ będzie zbiorem ciągów o parzystej ilości jedynek. Wówczas $u \sim_L w$ wtedy i tylko wtedy, gdy $\#_1(u) \equiv \#_1(w) \pmod 2$. Indeks relacji $\sim_L$ jest równy 2. *(b)* Niech $M$ będzie zbiorem słów $v$, w których $\#_0(v) = \#_1(v)$. Wówczas $u \sim_M w$ wtedy i tylko wtedy, gdy $\#_1(u) - \#_0(u) = \#_1(w) - \#_0(w)$. Indeks relacji $\sim_M$ jest nieskończony.

**Twierdzenie 3 (Myhill-Nerode)** *Niech $L \subseteq \Sigma^*$. Następujące warunki są równoważne:*

1. *relacja $\sim_L$ ma skończony indeks;*
2. *język $L$ jest regularny.*

*Ponadto, jeżeli warunki te zachodzą, to indeks relacji $\sim_L$ jest równy minimalnej liczbie stanów, jaką musi mieć automat deterministyczny rozpoznający język $L$.*

*Dowód.*

*(2) $\Rightarrow$ (1)* Niech $L = L(A)$, gdzie $A = (\Sigma, Q, q_I, \delta, F)$ jest deterministycznym automatem skończonym. Określamy relację $\sim_A$ na $\Sigma^*$,

$$u \sim_A w \Leftrightarrow \hat{\delta}(q_I, u) = \hat{\delta}(q_I, w)$$

Oczywiście, ta relacja jest równoważnością o indeksie nie większym niż $|Q|$. Z drugiej strony, pokażemy, że $\sim_A \subseteq \sim_L$. Istotnie, jeśli $\hat{\delta}(q_I, u) = \hat{\delta}(q_I, w)$ i $x \in \Sigma^*$ jest dowolnym słowem, to $\hat{\delta}(q_I, ux) = \hat{\delta}(q_I, wx)$, a zatem $ux \in L \Leftrightarrow wx \in L$.

Z inkluzji $\sim_A \subseteq \sim_L$ wynika, że indeks $\sim_L$ jest niewiększy[1] niż indeks $\sim_A$, który z kolei jest niewiększy od liczby stanów automatu $A$.

*(1) $\Rightarrow$ (2)* Przypomnijmy, że $[u]_L$ oznacza klasę abstrakcji słowa $u$ w relacji $\sim_L$. Określamy deterministyczny automat $A^L = (\Sigma, Q^L, q_I^L, \delta^L, F^L)$ następująco:

- $Q^L = \{[u]_L : u \in \Sigma^*\}$

[1] Jeżeli $r, s$ są równoważnościami na tym samym zbiorze i $r \subseteq s$, to klasy abstrakcji relacji $s$ są sumami klas abstrakcji relacji $r$.

- $q_I^L = [\epsilon]_L$
- $\delta^L([u]_L, \sigma) = [u\sigma]_L$
- $F^L = \{[u]_L : u \in L\}$

Z lematu o relacji $\sim_L$ wynika, że funkcja $\delta^L$ jest dobrze określona oraz, że $[w]_L \in F \Leftrightarrow w \in L$. Zauważmy, że liczba stanów automatu $A^L$ jest równa indeksowi relacji $\sim_L$.

Dla dowodu, że $L(A^L) = L$, wystarczy więc wykazać, że dla każdego $w \in \Sigma^*$,

$$\hat{\delta^L}(q_I, w) = [w]_L$$

Argument przebiega przez indukcję ze względu na długość $w$. Stosując równania rekurencyjne dla $\hat{\delta^L}$ ze str. 1, otrzymujemy

- $\hat{\delta^L}(q_I, \epsilon) = q_I = [\epsilon]_L$,
- $\hat{\delta^L}(q_I, w\sigma) = \delta^L(\hat{\delta^L}(q_I, w), \sigma) = \delta^L([w]_L, \sigma) = [w\sigma]_L$

□

**Pytanie.** Dlaczego rozumowania z powyższego dowodu nie można rozszerzyć na przypadek dowolnego automatu *niedeterministycznego* i tym samym udowodnić, że indeks $\sim_L$ jest niewiększy od liczby stanów dowolnego automatu skończonego rozpoznającego $L$?

## 1.3 Automat minimalny

W poprzednim punkcie wykazaliśmy, że spośród wszystkich automatów deterministycznych akceptujących język regularny $L$, automat $A^L$ opisany w dowodzie Twierdzenia Myhilla-Nerode'a ma minimalną możliwą liczbę stanów.

Obecnie zastanowimy się nad możliwością *skonstruowania* automatu $A^L$. Zauważmy, że automat ten zdefiniowaliśmy odołując się do relacji $\sim_L$, o której wiemy, że istnieje, ale która *a priori* nie jest nam znana, nawet jeśli mamy dane wyrażenie regularne lub automat rozpoznający język $L$. W konkretnych przypadkach, możemy często tę relację "odgadnąć" i przy jej pomocy zbudować automat $A^L$ (tak będzie w większości zadań). Dokładniej mówiąc, wystarczy wskazać skończony zbiór słów i dowieść, że każde słowo jest równoważne jednemu z nich. Jednakże nie jest to metoda dostatecznie ogólna. Zobaczymy wszakże, że automat $A^L$ można zawsze otrzymać z dowolnego automatu deterministycznego rozpoznającego $L$ za pomocą "sklejenia" niektórych jego stanów.

Potrzebne nam będzie uściślenie pojęcia "sklejania" a także jasne kryterium, kiedy można utożsamiać dwa automaty. Poniższe pojęcia można wprowadzić w przypadku ogólnym, jednak dla naszych potrzeb wystarczy rozważać automaty deterministyczne.

**Izomorfizm.** Intuicyjnie, powinno być oczywiste, że możemy utożsamić automaty, które różnią się jedynie *nazwami* stanów. Bardziej dokładnie, niech $A = (\Sigma, Q, q_I, \delta, F)$ i $B = (\Sigma, Q', q'_I, \delta', F')$ będą automatami deterministycznymi nad tym samym alfabetem $\Sigma$, takimi, że $|Q| = |Q'|$. Funkcję *bijekcję* $h : Q \to Q'$ nazywamy *izomorfizmem* z $A$ w $B$, jeśli

- $h(q_I) = q'_I$
- $(\forall q, q' \in Q)(\forall \sigma \in \Sigma)\, h(\delta(q, \sigma)) = \delta'(h(q), \sigma)$
- $(\forall q \in Q)\, q \in F \;\Leftrightarrow\; h(q) \in F'$

Zauważmy, że jeśli $h$ jest izomorfizmem z $A$ w $B$, to funkcja odwrotna $h^{-1} : Q' \to Q$ jest izomorfizmem z $B$ w $A$.

Jeśli istnieje izomorfizm z $A$ w $B$, to o automatach $A$ i $B$ mówimy, że są *izomorficzne* lub, że są *równe z dokładnością do izomorfizmu.*

**Lemat 2** *Jeśli automat deterministyczny $A$ rozpoznający język $L$ ma tyle samo stanów, co automat $A^L$, to jest z nim izomorficzny.*

*Dowód.* Niech $L = L(A)$, gdzie $A = (\Sigma, Q, q_I, \delta, F)$ jest automatem deterministycznym i niech $A^L$ będzie automatem skonstruowanym w dowodzie Twierdzenia Myhilla-Nerode'a, przy czym zakładamy, że $|Q| = |Q^L|$.

Przypomnijmy pojęcie relacji $\sim_A$ jakie zdefiniowaliśmy w tymże dowodzie (str. 4) :

$$u \sim_A w \;\Leftrightarrow\; \hat{\delta}(q_I, u) = \hat{\delta}(q_I, w)$$

Jak zauważyliśmy wówczas, zachodzi inkluzja $\sim_A \subseteq \sim_L$, przy czym indeks relacji $\sim_A$ jest większy lub równy od indeksu relacji $\sim_L$ a mniejszy lub równy od liczby stanów automatu $A$. Jeśli jednak, jak zakładamy, te skrajne wartości są równe, to relacje $\sim_A$ i $\sim_L$ mają ten sam indeks. Skoro jednak liczba ich klas abstrakcji jest skończona, a ponadto zachodzi inkluzja $\sim_A \subseteq \sim_L$, to relacje te muszą być równe[2], a zatem

$$\hat{\delta}(q_I, u) = \hat{\delta}(q_I, w) \quad \Leftrightarrow \quad u \sim_L w \tag{3}$$

Wykażemy teraz, że funkcja $h : Q^L \to Q$, określona

$$h \;:\; [w]_L \;\mapsto\; \hat{\delta}(q_I, w)$$

jest izomorfizmem automatów w sensie określonym powyżej.

Zauważmy najpierw, że z równoważności ( 3) wynika, iż

1. funkcja $h$ jest dobrze określona, tzn. nie zależy od wyboru reprezentanta $w \in [w]_L$;
2. funkcja $h$ jest różnowartościowa.

[2]Por. Przypis 1.

Wobec założonej równoliczności zbiorów $Q$ i $Q^L$ dowodzi to, że funkcja ta jest bijekcją.

Oczywiście $h([\epsilon]_L) = \hat{\delta}(q_I, \epsilon) = q_I$.

Z kolei, mamy

$$\begin{aligned} h(\delta^L([w]_L, \sigma)) &= h([w\sigma]_L) \\ &= \hat{\delta}(q_I, w\sigma) \\ &= \delta(\hat{\delta}(q_I, w), \sigma) \\ &= \delta(h([w]_L), \sigma) \end{aligned}$$

Wreszcie, dla każdego $w \in \Sigma^*$,

$$\begin{aligned} [w]_L \in F^L &\Leftrightarrow w \in L \\ &\Leftrightarrow \hat{\delta}(q_I, w) \in F \\ &\Leftrightarrow h([w]_L) \in F \end{aligned}$$

Wykazaliśmy, że funkcja $h$ jest izomorfizmem automatów $A$ i $A^L$. □

**Konstrukcja automatu ilorazowego**. Niech $A = (\Sigma, Q, q_I, \delta, F)$ będzie automatem deterministycznym i niech $r$ będzie dowolną relacją równoważności na zbiorze jego stanów $Q$. Klasy abstrakcji relacji $r$ będziemy oznaczać $[q]_r$. Intuicyjnie, relacja $r$ "skleja" między sobą niektóre stany automatu $A$, co prowadzi do konstrukcji nowego automatu, który jednak na ogół nie musi być deterministyczny; może też rozpoznawać inny język niż automat wyjściowy. Podamy jednak warunki, które gwarantują poprawność operacji sklejania.

Powiemy mianowicie, że relacja równoważności $r \subseteq Q \times Q$ jest *kongruencją*[3], jeśli

1. $(\forall q, q')\,(q, q') \in r \wedge q \in F \Rightarrow q' \in F$
2. $(\forall q, q')(\forall \sigma)\,(q, q') \in r \Rightarrow (\delta(q, \sigma), \delta(q', \sigma)) \in r$

Jeśli relacja $r$ jest kongruencją, to określamy *automat ilorazowy*

$$A/r = (\Sigma, Q_r, [q_I]_r, \delta_r, F_r)$$

gdzie $Q_r = \{[q]_r : q \in Q\}$, $F_r = \{[q]_r : q \in F\}$, a $\delta_r : Q_r \times \Sigma \to Q_r$ jest określona przez

$$(\forall q \in Q)(\forall \sigma \in \Sigma)\,\delta_r([q]_r, \sigma) = [\delta(q, \sigma)]_r$$

Zauważmy, że z warunku 2 wynika, że definicja funkcji $\delta_r$ jest poprawna, tzn. nie zależy od wyboru reprezentanta klasy abstrakcji $[q]_r$. Tak więc, $A/r$ jest również automatem deterministycznym.

**Lemat 3** $L(A/r) = L(A)$.

[3]Termin jest wzięty z algebry. Jak wiadomo relacja na uniwersum algebry jest nazywana kongruencją, jeśli jest niezmiennicza ze względu na operacje lub, co jest równoważne, jeśli jest jądrem pewnego homomorfizmu.

*Dowód.* Wykażemy przez indukcję ze względu na długość słowa $w \in \Sigma^*$, że

$$\hat{\delta_r}([q_I]_r, w) \;=\; [\hat{\delta}(q_I, w)]_r$$

Ponieważ, z definicji zbioru $F_r$ oraz z pierwszego warunku definicyjnego kongruencji mamy, że $[\hat{\delta}(q_I, w)]_r \in F_r$ wtedy i tylko wtedy, gdy $\hat{\delta}(q_I, w) \in F$, powyższa równość wystarczy do dowodu równości $L(A/r) = L(A)$.

Z równań rekurencyjnych (1) i (2) ze str. 1 zastosowanych dla funkcji $\hat{\delta_r}$, otrzymujemy najpierw

$$\hat{\delta_r}([q_I]_r, \epsilon) \;=\; [q_I]_r$$

a następnie

$$\begin{aligned}
\hat{\delta_r}([q_I]_r, w\sigma) &= \delta_r(\hat{\delta_r}([q_I]_r, w), \sigma) \\
&= \delta_r([\hat{\delta}(q_I, w)]_r, \sigma) \\
&\quad \text{(z założenia indukcyjnego)} \\
&= [\delta(\hat{\delta}(q_I, w), \sigma)]_r \\
&\quad \text{(z definicji } \delta_r\text{)} \\
&= [\hat{\delta}(q_I, w\sigma)]_r
\end{aligned}$$

Ta uwaga kończy dowód. □

Aby z dowolnego automatu deterministycznego rozpoznającego język $L$ otrzymać metodą konstrukcji ilorazowej automat $A^L$, potrzebny jest jeszcze drobny zabieg techniczny: eliminacja stanów, które nie mogą być osiągnięte ze stanu początkowego.

Niech $A = (\Sigma, Q, q_I, \delta, F)$ będzie automatem deterministycznym. *Stanem osiągalnym automatu* nazwiemy każdy stan osiągalny ze stanu początkowego $q_I$, tzn. taki, że $q_I \xrightarrow{w} q$, dla pewnego $w$ (por. Odcinek 1, punkt 2.5). Zauważmy, że stany nieosiągalne nie występują w żadnym obliczeniu automatu $A$. Mówiąc z grubsza, możemy je bez szkody "wyrzucić" z automatu.

Bardziej dokładnie, niech $Q_o$ będzie zbiorem stanów osiągalnych automatu $A$. Określamy automat $A_{red}$ (od ang. *reduced*) następująco:

$$A_{red} = (\Sigma, Q_o, q_I, \delta_{red}, F \cap Q_o)$$

gdzie $\delta_{red} = \delta \cap Q_o \times \Sigma \times Q_o$.

Następujący fakt pozostawiamy jako łatwe ćwiczenie.

**Lemat 4** *$A_{red}$ jest automatem deterministycznym i $L(A_{red}) = L(A)$.* □

**Twierdzenie 4** *Niech $L \subseteq \Sigma^*$ będzie językiem regularnym, niech $A^L$ będzie automatem skonstruowanym w dowodzie twierdzenia Myhilla-Nerode'a i niech $A$ będzie dowolnym automatem deterministycznym rozpoznającym język $L$. Niech $A_{red}$ będzie automatem otrzymanym z $A$ przez eliminację stanów nieosiągalnych, tak jak to opisaliśmy powyżej. Wówczas istnieje kongruencja $r$ na zbiorze stanów automatu $A_{red}$, taka, że automat ilorazowy $A_{red}/r$ jest izomorficzny z $A^L$.*

*Dowód.* Dla uproszczenia notacji, załóżmy, że $A$ jest już w postaci zredukowanej, tzn. $A_{red} = A = (\Sigma, Q, q_I, \delta, F)$. Określmy relację $\sim \subseteq Q \times Q$ przez

$$q \sim q' \;\Leftrightarrow\; L(A,q) = L(A,q')$$

(Przypomnijmy, Odcinek 1, punkt 2.3, że $L(A,q) = \{w : (\exists p \in F)\, q \xrightarrow{w} p\}$.) Oczywiście $\sim$ jest relacją równoważności. Aby wykazać, że jest także kongruencją, trzeba sprawdzić iż spełnia warunki definicyjne (por. str. 7), a mianowicie

1. $q \sim q' \;\Rightarrow\; (q \in F \Leftrightarrow q' \in F)$
2. $q \sim q' \;\Rightarrow\; (\forall \sigma \in \Sigma)\delta(q,\sigma) \sim \delta(q',\sigma)$

Pierwsza własność wynika z faktu, że $q \in F \Leftrightarrow \epsilon \in L(A,q)$. Dla dowodu drugiej, przypuśćmy, że $q \sim q'$ i niech $\sigma \in \Sigma$. Niech $\delta(q,\sigma) = p$ i $\delta(q',\sigma) = p'$. Potrzebujemy udowodnić

$$L(A,p) \;=\; L(A,p')$$

Przypuśćmy, że $w \in L(A,p)$, zatem $\sigma w \in L(A,q) = L(A,q')$. Odwołując się do pojęcia obliczenia jako ciągu przejść (por. Odcinek 1, Fakt 4), otrzymujemy $q' \xrightarrow{\sigma} q'' \xrightarrow{w} q_f$, dla pewnego $q'' \in Q$ i $q_f \in F$, ale ponieważ $A$ jest automatem deterministycznym, więc $q'' = p'$, czyli $w \in L(A,p')$. W ten sposób stwierdziliśmy, że $L(A,p) \subseteq L(A,p')$, z symetrii wynika, że $L(A,p) = L(A,p')$, a zatem $p \sim p'$.

Tak więc, relacja $\sim$ jest kongruencją i z Lematu 3 wynika, że $A/\sim$ jest deterministycznym automatem rozpoznającym język $L$. Pozostaje udowodnić, że $A/\sim$ jest izomorficzny z $A^L$. Na podstawie Lematu 2, wystarczy udowodnić, że liczba stanów automatu $A/\sim$ jest równa indeksowi relacji $\sim_L$.

Zanim przystąpimy do właściwego dowodu, zrobimy pewną obserwację, jaka wynika wprost z przyjętych definicji.

(*) Jeśli $q_I \xrightarrow{w} q$, to $L(A,q) = \{w\}^{-1}L$

Określmy teraz funkcję $f : Q^L \to Q_\sim$ następująco:

$$(\forall w \in \Sigma^*)\, f([w]_L) \;=\; [\hat{\delta}(q_I, w)]_\sim$$

Odnotujmy najpierw, że z (*) wynika, że funkcja $f$ jest dobrze określona, tzn. $w \sim_L v \Rightarrow \hat{\delta}(q_I,w) \sim \hat{\delta}(q_I,v)$. (Istotnie, $w \sim_L v$ implikuje, że $\{w\}^{-1}L = \{v\}^{-1}L$, co, wobec (*) implikuje $L(A,\hat{\delta}(q_I,w)) = L(A,\hat{\delta}(q_I,v))$, skąd $\hat{\delta}(q_I,w) \sim \hat{\delta}(q_I,v)$.)

Z kolei wykażemy, że funkcja $f$ jest różnowartościowa. Istotnie, $h([w]_L) = h([v]_L)$ oznacza, że $\hat{\delta}(q_I,w) \sim \hat{\delta}(q_I,v)$, czyli $L(A,\hat{\delta}(q_I,w)) = L(A,\hat{\delta}(q_I,v))$, co wobec (*) implikuje że $\{w\}^{-1}L = \{v\}^{-1}L$, a więc $[w]_L = [v]_L$.

Wreszcie, z faktu, że wszystkie stany automatu $A$ są osiągalne, wynika, że funkcja $f$ jest *na* zbiór $Q_\sim$.

Ta uwaga kończy dowód twierdzenia. □

Nasze rozważania o automatach minimalnych podsumowuje następujący

**Wniosek 1** *Spośród wszystkich automatów deterministycznych akceptujących język regularny $L$, automat $A^L$ skonstruowany w dowodzie Twierdzenia Myhilla–Nerode'a ma* minimalną *ilość stanów, równą indeksowi relacji $\sim_L$. Ponadto, jeśli jakiś automat deterministyczny rozpoznający język $L$ ma* tyle samo *stanów co $A^L$, to jest z nim izomorficzny. Wreszcie, automat $A^L$ (z dokładnością do izomorfizmu) można otrzymać z dowolnego automatu deterministycznego rozpoznającego język $L$ poprzez eliminację stanów nieosiągalnych i konstrukcję ilorazową.* □

Jest więc w pełni uzasadnione, że automat $A^L$ będziemy nazywali *mimimalnym* automatem deterministycznym rozpoznającym język $L$, lub krótko: *automatem minimalnym* dla języka $L$.

W następującym przykładzie wykażemy, że eksponencjalne oszacowanie na wzrost liczby stanów przy determinizacji automatu niedeterministycznego (Twierdzenie 1) jest w ogólności optymalne.

**Przykład 4** Niech $\Sigma = \{0, 1\}$ *i niech* $k \geq 1$. Niech $L_k$ będzie zbiorem słów, które na $k$-tej od końca pozycji mają 1, tzn.

$$L_k = \{w \,:\, |w| \geq k \;\; i \;\; w_{|w|+1-k} = 1\}$$

Nietrudno podać automat *niedeterministyczny* o $k + 1$ stanach rozpoznający język $L_k$; pozostawiamy to jako ćwiczenie. Dla określenia liczby stanów minimalnego automatu *deterministycznego*, rozważmy relację $\sim_{L_k}$. Zauważmy najpierw, że żadne dwa różne słowa o długości $k$ nie są równoważne. Istotnie, jeśli $|w| = |u| = k$ i np. $w_{k-i} = 0$ a $u_{k-i} = 1$, to $u0^i \in L_k$, podczas gdy $w0^i \notin L_k$. Z drugiej strony, nietrudno sprawdzić, że każde słowo jest równoważne, w sensie relacji $\sim_{L_k}$, pewnemu słowu długości $k$: gdy $|w| > k$, to $w$ jest równoważne swojemu sufiksowi długości $k$, a gdy $0 \leq |w| < k$, to $w \sim_{L_k} 0^{k-|w|}w$. A zatem, indeks relacji $\sim_{L_k}$ *ergo* liczba stanów automatu minimalnego jest $2^k$.

**Pytanie.** Co powyższy przykład mówi nam na temat operacji lustrzanego odbicia?

## 1.4 Algorytm minimalizacji automatu deterministycznego

W poprzednim punkcie opisaliśmy, jak z dowolnego automatu deterministycznego rozpoznającego język $L$ można otrzymać automat minimalny $A^L$. Algorytmicznie, problem sprowadza się do dwóch zadań: eliminacji stanów nieosiągalnych i obliczenia relacji $\sim$ na zbiorze stanów. Kiedy już znamy relację $\sim$, konstrukcja automatu ilorazowego jest bezpośrednia.

Algorytm dla obliczenia stanów osiągalnych już rozważaliśmy w Odcinku 1 (Punkt 2.5, Algorytm 3). Obecnie podamy algorytm obliczania relacji $\sim$.

Niech więc $A = (\Sigma, Q, q_I, \delta, F)$ będzie automatem deterministycznym i $L = L(A)$. W związku z uwagą powyżej, możemy zakładać, że wszystkie stany automatu $A$ są osiągalne.

Naszym zadaniem jest wyznaczenie relacji $\sim \subseteq Q \times Q$, która, jak pamiętamy, była zdefiniowana następująco:

$$q \sim q' \;\Leftrightarrow\; L(A, q) = L(A, q')$$

W rzeczywistości, wyznaczymy *uzupełnienie* relacji $\sim$, tj. zbiór par $\{(q, q') : q \not\sim q'\}$. W tym celu określimy najpierw pewien graf zorientowany, $G_A$. Wierzchołkami będą wszystkie pary nieuporządkowane $\{p, q\}$, $p \neq q$. Dalej, prowadzimy krawędź z wierzchołka $\{p, q\}$ do $\{p', q'\}$, o ile $\{p, q\} \neq \{p', q'\}$ i istnieje $\sigma \in \Sigma$ t.że $p \xrightarrow{\sigma} p'$ i $q \xrightarrow{\sigma} q'$. (Zauważmy w tym miejscu, że, z powodu determinizmu automatu $A$, z każdego wierzchołka grafu $G_A$ wychodzi $|\Sigma|$ krawędzi. Zatem liczba krawędzi grafu $G_A$ jest $O(n^2)$.) W grafie $G_A$ zaznaczamy najpierw wszystkie wierzchołki postaci $\{p, q\}$, takie że $p \in F$ i $q \notin F$. Zauważmy, że, dla każdej zaznaczonej pary $\{p, q\}$, zachodzi na pewno $p \not\sim q$. Niech $Z$ będzie zbiorem wszystkich zaznaczonych wierzchołków. Twierdzimy, że

> Dla każdych $p, q \in Q$, $p \not\sim q$ wtedy i tylko wtedy, gdy istnieje ścieżka z wierzchołka $\{p, q\}$ do jakiegoś wierzchołka w zbiorze $Z$.

Istotnie, jeśli $p \not\sim q$, to mamy jakieś słowo $v = v_1 \dots v_m$, takie, że, np. $\hat{\delta}(p, v) \in F$ a $\hat{\delta}(q, v) \notin F$. Mamy więc

$$p \xrightarrow{v_1} p_1 \xrightarrow{v_2} p_2 \dots p_{m-1} \xrightarrow{v_m} p_m \in F$$

i

$$q \xrightarrow{v_1} q_1 \xrightarrow{v_2} q_2 \dots q_{m-1} \xrightarrow{v_m} q_m \notin F$$

Nietrudno widzieć, że

$$\{p, q\} \rightarrow \{p_1, q_1\} \rightarrow \dots \{p_{m-1}, q_{m-1}\} \rightarrow \{p_m, q_m\}$$

jest wówczas żądaną ścieżką w grafie, przy czym $p_m \in F$ a $q_m \notin F$, a więc wierzchołek $\{p_m, q_m\}$ należy do $Z$.

Dla udowodnienia implikacji odwrotnej, zauważmy najpierw, że jeśli $p \not\sim q$ i istnieje strzałka z $\{p', q'\}$ do $\{p, q\}$, to również $p' \not\sim q'$. Zatem teza powyższa wynika łatwo przez indukcję ze względu na długość ścieżki, jaka prowadzi z wierzchołka $\{p, q\}$ do zbioru $Z$.

Wyznaczenie relacji $\not\sim$ sprowadza się więc do obliczenia zbioru wierzchołków grafu $G_A$, dla których istnieje ścieżka prowadząca do jakiegoś wierzchołka w zbiorze $Z$. Łatwo widzieć, że interesujący nas zbiór jest zbiorem wierzchołków *osiągalnych* z $Z$ w grafie dualnym $G_A^d$ (tj., jeśli odwrócimy kierunek każdej krawędzi). Możemy więc dla jego obliczenia zastosować Algorytm 3 z Odcinka 1 (dokładniej, graf $G_A^d$, w którym krawędzie nie są etykietowane, możemy traktować jak graf automatu nad jednoliterowym alfabetem). Czas działania tego algorytmu jest proporcjonalny do liczby krawędzi w grafie, która w wypadku grafu $G_A$ jest $O(n^2)$, takie jest też więc oszacowanie czasu działania przedstawionego algorytmu. Algorytm ten znany jest jako Algorytm Moore'a.

Nietrudno pokazać, że również wyznaczenie grafu $G_A$ i zbioru $Z$ może być wykonane w czasie $O(n^2)$, co pozwala nam podsumować powyższe rozważania w następującym twierdzeniu.

**Twierdzenie 5 (Algorytm Moore'a)** *Istnieje algorytm, który dla danego automatu deterministycznego o $n$ stanach konstruuje w czasie $O(n^2)$* minimalny *automat deterministyczny rozpoznający ten sam język.*

W Ćwiczeniach przedstawimy Algorytm Hopcrofta, którego czas działania jest lepszy, mianowicie $O(n \log n)$.

Zauważmy jeszcze na koniec, że łącząc opisany wyżej algorytm z algorytmem otrzymywania automatu niedeterministycznego z wyrażenia regularnego i algorytmem determinizacji automatu niedeterministycznego, otrzymujemy metodę konstrukcji automatu minimalnego dla języka zadanego wyrażeniem regularnym. Metoda ta nie jest jednak w każdym przypadku efektywna z uwagi na eksponencjalny wzrost rozmiaru automatu w procesie determinizacji.

Zagadnienie efektywnej metody znajdywania automatu *niedeterministycznego* o najmniejszej możliwej liczbie stanów, jest problemem otwartym.

# Przykłady i ćwiczenia

## Algorytm Hopcrofta

Przedstawimy teraz inny algorytm minimalizacji automatu. Niech $A = (\Sigma, Q, q_I, \delta, F)$ będzie, jak wyżej, automatem deterministycznym. Tym razem będziemy dążyć do obliczenia relacji $\sim$ (zdefiniowanej jak wyżej) poprzez obliczenie podziału, jaki ta relacja równoważności wyznacza na zbiorze $Q$. Poszukiwany podział będzie znaleziony "metodą kolejnych przybliżeń", przy czym każdy kolejny podział będzie rozdrobnieniem poprzedniego.

Niech $X, Y \subseteq Q$, $a \in \Sigma$. Określamy $X' = \{q \in X : \delta(q,a) \in Y\}$, $X'' = \{q \in X : \delta(q,a) \notin Y\}$. Oczywiście, $X' \cup X'' = X$, $X' \cap X'' = \emptyset$. Jeżeli oba zbiory $X', X''$ są niepuste, to powiemy, że para $(Y,a)$ *łamie* zbiór $X$, a parę $\{X', X''\}$ nazwiemy *złamaniem* zbioru $X$ przez $(Y,a)$. Jeżeli $P$ jest podziałem zbioru $Q$, to *złamaniem* $P$ przez $(Y,a)$ nazywamy podział otrzymany z podziału $P$ przez zastąpienie każdego $X \in P$, który jest łamany przez $(Y,a)$, przez jego złamanie $X', X''$.

W algorytmie będziemy używać dwóch typów struktur: z jednej strony, podziału (zmienna $P$), a z drugiej, listy par $(Y,a)$ przeznaczonych do łamania (zmienna $\mathcal{L}$). Przyjmujemy oznaczenie $\bar{X} = Q - X$.

procedura **Hopcroft**
$P := \{F, \bar{F}\}$;
**jeśli** $|F| < |\bar{F}|$ **to** $\mathcal{L} := ((F,\sigma) : \sigma \in \Sigma)$
**w przeciwnym przypadku** $\mathcal{L} := ((\bar{F},\sigma) : \sigma \in \Sigma)$
**dopóki** $\mathcal{L} \neq \emptyset$ **wykonuj**

1. zdejmij parę $(Y,a)$ z listy $\mathcal{L}$;
2. oblicz zbiór $\mathcal{B}$ tych $X \in P$, że para $(Y,a)$ łamie $X$;
3. **dla** $X \in \mathcal{B}$, $b \in \Sigma$ **wykonuj**
   (a) oblicz złamanie $\{X', X''\}$ przez $(Y,a)$;
   (b) **jeśli** $(X,b) \in \mathcal{L}$ **to**
   zastąp $(X,b)$ w $\mathcal{L}$ przez $(X',b), (X'',b)$;

(c) **w przeciwnym przypadku**
**jeśli** $|X'| < |X''|$ **to** dołącz $(X', b)$ do $\mathcal{L}$
**w przeciwnym przypadku** dołącz $(X'', b)$ do $\mathcal{L}$

Należy dowieść, że algorytm działa poprawnie, tzn. zawsze zatrzymuje się i końcową wartością zmiennej $P$ jest podział wyznaczony przez relację $\sim$ z wykładu. Ponadto, należy wykazać, że czas działania algorytmu jest $O(n \log n)$.

## Algorytmy tekstowe oparte na automatach

***Rozpoznawanie wzorca w tekście.*** Niech $w \in \Sigma^*$ będzie słowem o długości $|w| = n > 0$. Minimalny automat deterministyczny rozpoznający wszystkie słowa nad $\Sigma$, których sufiksem jest $w$, ma dokładnie $n+1$ stanów. Jako stany automatu można przyjąć wszystkie prefiksy słowa $w$ (a zatem $|w|+1$ stanów), a jako przejścia $v \xrightarrow{a} u$, gdzie $u$ jest maksymalnym sufiksem słowa $va$ będącym jednocześnie prefiksem $w$.

Co więcej, można dowieść, że liczba nietrywialnych przejść ,,wstecznych", tj. przejść postaci $v \xrightarrow{a} u$, gdzie $|v| > |u| > 0$, jest nie większa niż $|w|$. Daje to możliwość zwięzłej reprezentacji automatu: wystarczy pamiętać $\leq 2 \cdot |w|$ przejść (przejścia postaci $v \xrightarrow{a} \epsilon$ możemy pominąć).
*Wskazówka.* Wykazać, że dla każdej liczby $k$ istnieje co najwyżej jedno nietrywialne przejście wsteczne $v \xrightarrow{a} u$, spełniające $|va| - |u| = k$.

***Rozpoznawanie podsłów.***

1. Dla danego słowa $w$ o długości $|w| = n$ skonstruować automat deterministyczny o $\leq 2n+1$ stanach rozpoznający dokładnie zbiór sufiksów słowa $w$.

   *Wskazówka.* Jako stany automatu można wziąć zbiory pozycji $S_x$ ($S_x \subseteq \{0, 1, \ldots, n\}$) kończących wystąpienie $x$ jako podsłowa słowa $w$. Oszacowanie na liczbę stanów wynika ze spostrzeżenia, że różne zbiory $S_x$, $S_y$ są albo w relacji inkluzji albo rozłączne, a zatem, ze względu na inkluzję, tworzą drzewo o nie więcej niż $n$ liściach.

2. Powyższy automat zawiera stan typu ,,czarna dziura", mianowicie stan $\emptyset = S_x$, gdzie $x$ nie jest podsłowem $w$. Dowieść, że liczbę przejść nie prowadzących do stanu $\emptyset$ można oszacować z góry przez $3n$.

   *Wskazówka.* Wziąć drzewo rozpinające grafu automatu i oszacować liczbę pozostałych krawędzi przez liczbę sufiksów $w$.

3. Opisany wyżej automat jest minimalny dla zbioru sufiksów. Tę samą konstrukcję można zastosować również do rozpoznawania zbioru wszystkich podsłów słowa $w$, ale otrzymany automat nie musi być minimalny. Podać przykład.

   *Wskazówka:* Patrz 4 grudnia.

# 2 Własności domknięcia klasy języków regularnych

Rodzina języków regularnych ma wiele dobrych własności, w szczególności jest zamknięta na większość możliwych do pomyślenia operacji. W poniższym twierdzeniu podsumowujemy te własności. Poszczególne punkty zostały już udowodnione w trakcie wykładu lub jako ćwiczenia.

**Twierdzenie 6** *Klasa języków regularnych jest zamknięta na*

- *operacje Boole'owskie, tzn. jeżeli* $L, M \subseteq \Sigma^*$ *są językami regularnymi, to również* $L \cup M$, $L \cap M$, $\Sigma^* - L$, $L - M$, *są językami regularnymi;*
- *konkatenację, tzn. jeżeli* $L, M \subseteq \Sigma^*$ *są językami regularnymi, to również* $LM$ *jest językiem regularnym;*
- *operację gwiazdki, tzn. jeżeli* $L$ *jest językiem regularnym, to* $L^*$ *jest również językiem regularnym;*
- *ilorazy przez dowolne zbiory, tzn. jeżeli* $L$ *jest językiem regularnym i* $X \subseteq \Sigma^*$ *jest dowolnym językiem, to* $X^{-1}L$ *i* $LX^{-1}$ *są językami regularnymi;*
- *operację lustrzanego odbicia, tzn. jeżeli* $L$ *jest językiem regularnym, to* $L^R = \{w^R : w \in L\}$ *jest językiem regularnym;*
- *podstawienie, tzn. jeśli* $f : \Sigma \to P(\Gamma^*)$ *jest funkcją taką, że, dla każdego* $\sigma \in \Sigma$, $f(\sigma)$ *jest językiem regularnym i* $L \subseteq \Sigma^*$ *jest językiem regularnym, to* $\hat{f}(L)$ *jest również językiem regularnym;*
- *obrazy homomorficzne, tzn. jeśli* $L \subseteq \Sigma^*$ *jest językiem regularnym, i* $h : \Sigma \to \Gamma^*$ *jest homomorfizmem, to* $h(L)$ *jest językiem regularnym;*
- *przeciwobrazy homomorficzne, tzn. jeśli* $M \subseteq \Gamma^*$ *jest językiem regularnym, i* $h : \Sigma \to \Gamma^*$ *jest homomorfizmem, to* $h^{-1}(M)$ *jest językiem regularnym.*

Większość powyższych własności domknięcia ma charakter efektywny, tzn. mając dane automaty lub wyrażenia regularne dla języków $L$, $M$, ..., potrafimy znaleźć automat (wyrażenie) dla języka otrzymanego w wyniku stosowanej operacji. Nie dotyczy to jedynie operacji ilorazu przez *dowolny* zbiór, który nie musi być zadany efektywnie, jeśli jednak ograniczymy się do ilorazów przez języki regularne, to operacja jest, oczywiście, efektywna.

Własności te pozwalają więc na dość elastyczne konstruowanie języków regularnych. Z drugiej strony mogą też być przydatne przy dowodzie nieregularności jakiegoś języka. Rozważmy np. język $bb^*\{a^nb^n : n \in N\}$. Język ten nie jest regularny, choć spełnia tezę lematu o pompowaniu (przynajmniej w tej formie jaką przyjęliśmy w Lemacie 4 w Odcinku 1). Jednakże stosując lewostronny iloraz przez $bb^*$ otrzymujemy język $\{a^nb^n : n \in N\}$, którego nieregularność potrafimy wykazać. Stąd, z własności domknięcia na ilorazy, możemy wnioskować o nieregularności pierwszego języka.

# Zadania o językach regularnych

1. Dowieść, że jeśli $L$ jest językiem regularnym, to zbiór prefiksów słów z $L$, zbiór sufiksów słów z $L$, a także zbiór podsłów, tzn.

$$\{w : (\exists u, v \in \Sigma^*)uwv \in L\}$$

   są językami regularnymi.

2. Skonstruować minimalny automat deterministyczny rozpoznający język $L \subseteq \{a, b\}^*$, gdzie

   - $L = \{\epsilon\}$ ,
   - $L = \{a, b, aba\}$ ,
   - $L$ jest zbiorem słów w których $a$ występuje dokładnie 1993 razy,
   - $L$ jest zbiorem słów w których liczba wystąpień $a$ daje resztę 3 z dzielenia przez 5.

3. Ile stanów ma minimalny automat deterministyczny rozpoznający, że ciąg zer i jedynek jest rozwinięciem binarnym liczby podzielnej przez 3?

4. W hotelu jest $K$ pięter. Przejazd windy bez zatrzymania z piętra $i$ na piętro $j$ $(i \neq j)$ uważamy za jednostkową akcję. Z każdą taką akcją wiążemy literę: $G$ jeśli $i < j$, $D$ jeśli $i > j$. Rozważamy możliwe trasy windy, które rozpoczynają się i kończą na parterze. Każda taka trasa wyznacza słowo nad alfabetem $\{G, D\}$. Niech $L \subseteq \{G, D\}^*$ będzie zbiorem wszystkich tak otrzymanych słów.

   (a) Dowieść, że $L$ jest regularny.

   (b) Ile stanów ma minimalny automat deterministyczny rozpoznający $L$?

5. Niech $L = \{a^{2^n} : n \geq 1\}$. Dowieść , że jedyne podzbiory $L$ będące językami regularnymi, to zbiory skończone.

6. $\Sigma$ jest dowolnym alfabetem. Ustalamy słowo $w \in \Sigma^*$ o długości $n$. Skonstruować automat deterministyczny o $n + 1$ stanach rozpoznający wszystkie słowa $u \in \Sigma^*$ zawierające $w$ jako podsłowo (tzn. $u = v_1 w v_2$; mówiąc bardziej obrazowo, konstruowany automat poszukuje wzorca $w$ w tekście $u$).

7. Niech $L$ będzie zbiorem słów $w \in \{a, b\}^*$, które zawierają przynajmniej jedno wystąpienie symbolu $b$, przy czym bezpośrednio przed *ostatnim* wystąpieniem symbolu $b$ w słowie $w$ musi wystąpić symbol $a$. Dowieść, że $L$ jest językiem regularnym. Podać liczbę stanów minimalnego automatu deterministycznego rozpoznającego język $L$. Skonstruować wyrażenia regularne opisujące każdą z klas abstrakcji relacji $\sim_L$ z twierdzenia Myhill'a–Nerode'a wyznaczonej przez $L$.

8. Pan $X$ zakupił pakiety trzech różnych akcji: $A, B$ i $C$. Każdego dnia bada wzajemne położenie wartości swoich akcji, które może być reprezentowane jako *uporządkowanie* zbioru $\{A, B, C\}$, w kierunku od najmniej do najbardziej wartościowej. (Przyjmujemy założenie, że dwie różne akcje mają zawsze *różne* wartości.) Pan $X$ postanowił, że jeśli w ciągu dwóch kolejnych dni któraś z akcji dwukrotnie spadnie na niższą pozycję, to sprzeda tę akcję. Np. jeśli kolejne notowania (w tym wypadku: uporządkowania) są $(A, B, C), (B, C, A), (C, B, A)$ to pan $X$ sprzeda pakiet akcji $C$. Rozważamy zbiór $G$ wszystkich uporządkowań liniowych zbioru $\{A, B, C\}$. Dowieść, że zbiór tych ciągów nad alfabetem $G$, które opisują takie wyniki notowań giełdowych, przy których pan $X$ nie pozbędzie się żadnego pakietu akcji, jest językiem regularnym. Znaleźć liczbę stanów automatu minimalnego akceptującego ten język.

9. (*) Pan $X$ postanowił, że każdego dnia będzie pracował lub nie, przestrzegając przy tym zasady, by na żadne siedem kolejnych dni nie przypadało więcej niż cztery dni pracy. Możliwy rozkład dni pracy pana $X$ w ciągu $n$ kolejnych dni możemy przedstawić jako $n$-bitowe słowo, gdzie 1 odpowiada dniowi pracy, a 0 dniowi odpoczynku. Dowieść, że zbiór wszystkich tak otrzymanych słów jest językiem regularnym (nad alfabetem $\{0, 1\}$). Znaleźć minimalny automat deterministyczny.

10. (*) Dowieść, że dla dowolnego języka regularnego $L$, język

$$\{w : w^{|w|} \in L\}$$

jest również regularny.

## Rozwiązania niektórych zadań

*Rozwiązanie zadania 9.*

Niech $L$ będzie opisanym językiem. Rozważmy relację $\sim_L$ z twierdzenia Myhill'a Nerode'a. Twierdzimy, że każde słowo jest równoważne pewnemu słowu długości 7. Istotnie, zauważmy najpierw, że wszystkie słowa $w \notin L$ są równoważne między sobą, a więc równoważne w szczególności słowu 1111111. Dalej, nietrudno widzieć, że każde słowo, które należy do $L$ i jest dłuższe niż 7, jest równoważne swojemu sufiksowi długości 7. Natomiast słowo $w$ o długości $|w| = i < 7$ jest równoważne słowu $0^{7-i}w$. A zatem relacja $\sim_L$ ma nie więcej niż $2^7$ klas abstrakcji, język $L$ jest więc regularny.

Powyższy argument prowadzi też w prosty sposób do wyznaczenia automatu, którego stanami są słowa zero-jedynkowe długości 7, a którego przejścia są postaci

$$a_1a_2 \dots a_7, i \;\to\; a_2a_3 \dots a_7 i$$

gdzie $i \in \{0, 1\}$, o ile ciąg $a_1a_2 \dots a_7$ zawiera nie więcej niż 4 jedynki, w przeciwnym razie odpowiednie przejście jest

$$a_1a_2 \dots a_7, i \;\to\; a_1a_2 \dots a_7$$

dla $i = 0, 1$. Stanem początkowym jest ciąg 0000000, a stanami akceptującymi są oczywiście te ciągi, które zawierają nie więcej niż 4 jedynki.

**Nie jest to jednak automat minimalny**.

Poniżej wyznaczymy liczbę klas abstrakcji relacji $\sim_L$, konstrukcja samego automatu wynika z twierdzenia Myhill'a-Nerode'a. Na mocy wcześniejszej uwagi, wystarczy odpowiedzieć na pytanie, które słowa długości 7 są równoważne.

Słowo $w \in \{0,1\}^7$ nazwiemy *pełnym*, jeśli zawiera *dokładnie* 4 jedynki.

**Lemat 1**. Dwa różne słowa pełne są nierównoważne.

Istotnie, niech $w = w_1 \dots w_7$, $v = v_1 \dots v_7$ będą pełnymi słowami i niech $i$ będzie najwyższą pozycją, gdzie słowa $w$ i $v$ się różnią, powiedzmy $w_i = 0$ i $v_i = 1$. Niech $k$ będzie liczbą jedynek leżących w słowie $w$ na lewo od pozycji $i$,

$$k = |\{j \,:\, j < i \wedge w_j = 1\}|$$

Zauważmy, że $k \geq 1$, bo w przeciwnym razie $w$ miałoby mniej jedynek niż $v$, oraz, że, oczywiście, $k < i$ (w szczegolności więc $i > 1$). Niech $u = 0^j 1^k$, gdzie $j + k = i - 1$ ($j \geq 0$). Twierdzimy, że $wu \in L$, podczas gdy $vu \notin L$. Ten ostatni fakt wynika łatwo z konstrukcji, gdyż słowo $v_i \dots v_7 u$ jest długości 7 i zawiera 5 jedynek. Dla sprawdzenia, że $wu \in L$, wystarczy sprawdzić wszystkie podsłowa długości 7 postaci $w_m \dots w_7 0^j 1^\ell$, gdzie $m < i$ i $\ell > 0$ (pozostałe podsłowa długości 7 słowa $wu$ nie mogą zawierać więcej jedynek niż $w$).

## $k = 3$, $\ell = 1$

Zauważmy jednak, że $i - m = k - \ell$ (rysunek), a zatem słowo $w_m w_{m+1} \dots w_{i-1}$ zawiera co najwyżej $k - \ell$ jedynek, podczas gdy słowo $w_i w_{i+1} \dots w_7 0^j$ zawiera $7 - k$ jedynek, całe słowo $w_m \dots w_7 0^j 1^\ell$ zawiera więc niewięcej niż $(k - \ell) + (7 - k) + \ell = 7$ jedynek. Słowa $w$ i $v$ nie są zatem równoważne, co kończy dowód Lematu 1.

**Lemat 2**. Każde słowo $w \in \{0,1\}^7$ zawierające $k < 4$ jedynki jest równoważne słowu pełnemu jakie otrzymamy z $w$ przez zastąpienie $4-k$ najwcześniej występujących zer przez jedynki.

Np., 0100100 jest równoważne 1110100. Istotnie, niech $w$ będzie słowem w którym występują $k < 4$ jedynki i niech $v$ będzie słowem otrzymanym z $w$ jak w lemacie. Oczywiście, $(\forall u) vu \in L \Rightarrow wu \in L$. Przypuśćmy, że dla pewnego $u$ implikacja odwrotna nie zachodzi, tj. $wu \in L$, ale $vu \notin L$. Wówczas możemy znaleźć $i$, takie że $v_i v_{i+1} \dots v_7 u_1 \dots u_{i-1}$ jest podsłowem $vu$ długości 7 zawierającym więcej niż 4 jedynki. Wybierzmy najmniejsze $i$ o tej własności. Pamiętajmy, że odpowiednie słowo $w_i w_{i+1} \dots w_7 u_1 \dots u_{i-1}$ zawiera nie więcej niż 4 jedynki. Zatem $v_i v_{i+1} \dots v_7$ różni się

od $w_i w_{i+1} \dots w_7$, tzn. dla pewnego $i \leq i_0$, $w_{i_0} = 0$ a $v_{i_0} = 1$. Twierdzimy dalej, że $v_{i-1} = 1$. Istotnie, gdyby $v_{i-1} = 0$, to także $w_{i-1} = 0$ i zgodnie z definicją $v$, na pozycji $i-1$ w słowie $w$ zero powinno zostać zastąpione przez 1 (wiemy, że po transformacji $w_1 \dots w_{i-2}$ w $v_1 \dots v_{i-2}$ pozostawało jeszcze jakieś zero do zastąpienia, skoro zero na pozycji $i_0 \geq i$ zostało zastąpione przez 1 !). A zatem $v_{i-1} = 1$. Wówczas jednak słowo [4] $v_{i-1} v_i \dots v_7 u_1 \dots u_{i-2}$ zawiera co najmniej tyle samo jedynek, co rozważane wcześniej słowo $v_i v_{i+1} \dots v_7 u_1 \dots u_{i-1}$, co przeczy minimalności $i$. Ta uwaga kończy dowód Lematu 2.

Pamiętając o tym, że wszystkie słowa nie należące do $L$ są równoważne, oraz że każde słowo jest równoważne pewnemu słowu o długości 7, z Lematów 1 i 2, otrzymujemy liczbę klas abstrakcji relacji $\sim_L$, która stanowi zarazem liczbę stanów minimalnego automatu deterministycznego

$$\left( \begin{array}{c} 7 \\ 4 \end{array} \right) + 1 \;\; = \;\; 36$$

**Uwaga**. Powyższa liczba może być otrzymana jeszcze na kilka innych sposobów. Na przykład, nietrudno zauważyć, że każde dwa słowa długości 7 zakończone sufiksem 000 są równoważne. Obserwacja ta może być uogólniona przez ciąg następujących stwierdzeń: każde dwa słowa długości 7 zakończone sufiksem długości 4 zawierającym dokładnie jedną jedynkę, są równoważne, podobnie każde dwa słowa zakończone sufiksem długości 5 zawierającym dokładnie dwie jedynki, itd. (ostatnie stwierdzenie dotyczy sufiksów tożsamych z całym słowem i jest trywialne). Powyższe uwagi prowadzą do wyznaczenia liczby nierównoważnych słów długości 7, które należą do $L$:

$$\left( \begin{array}{c} 3 \\ 0 \end{array} \right) + \left( \left( \begin{array}{c} 4 \\ 1 \end{array} \right) - \left( \begin{array}{c} 0 \\ 0 \end{array} \right) \right) + \left( \left( \begin{array}{c} 5 \\ 2 \end{array} \right) - \left( \begin{array}{c} 4 \\ 1 \end{array} \right) \right) + \left( \left( \begin{array}{c} 6 \\ 3 \end{array} \right) - \left( \begin{array}{c} 5 \\ 2 \end{array} \right) \right) + \left( \left( \begin{array}{c} 7 \\ 4 \end{array} \right) - \left( \begin{array}{c} 6 \\ 3 \end{array} \right) \right) = \left( \begin{array}{c} 7 \\ 4 \end{array} \right)$$

*Rozwiązanie zadania 10.* Niech $A = (\Sigma, Q, q_1, \delta, F)$ będzie deterministycznym automatem, takim że $L(A) = L$. Niech $Q = \{q_1, \dots, q_n\}$. W dalszym ciągu rozważań będziemy dla wygody notacyjnej utożsamiać funkcję $f : Q \to Q$ z ciągiem $(f(q_1), \dots, f(q_n))$. Rozpocznijmy od uwagi, że każde słowo $w \in \Sigma^*$ wyznacza funkcję $f_w : Q \to Q$ określoną

$$f_w = (\hat{\delta}(q_1, w), \dots, \hat{\delta}(q_n, w))$$

Skonstruujemy najpierw automat $A'$, który, dla każdego $w$, "oblicza" funkcję $f_w$ w następującym sensie. Zbiorem stanów automatu $A'$ jest właśnie $Q^Q$ (zbiór funkcji z $Q$ w $Q$), stanem początkowym jest funkcja identycznościowa $id = (q_1, \dots, q_n)$, a funkcja przejścia $\delta'$ jest określona przez

$$\delta'((p_1, \dots, p_n), \sigma)(q) = (\delta(p_1, \sigma), \dots, \delta(p_n, \sigma))$$

Nietrudno wykazać przez indukcję, że dla dowolnego $w$

$$\hat{\delta'}(id, w) = f_w$$

[4] Gdy $i - 2 = 0$, rozumiemy przez to, że sufiks $u_1 \dots u_{i-2}$ jest pusty.

w tym właśnie sensie automat $A'$ "oblicza" funkcję $f_w$ (ewentualne stany akceptujące nie mają tu znaczenia).

Przypuśćmy na chwilę, że dla pewnego $w$ znamy funkcję $f = f_w$. Możemy wówczas skonstruować automat, powiedzmy $A_f$, który czytając słowo $w$ potrafi "obliczyć" stan, do którego automat $A$ dochodzi po przeczytaniu słowa $w^{|w|}$. Zbiór stanów automatu $A_f$ pokrywa się ze zbiorem stanów automatu $A$, stanem początkowym jest $q_1$, a funkcja przejścia, powiedzmy $\delta''$, jest określona przez

$$\delta''(q, \sigma) = f(q)$$

Nietrudno widzieć, że istotnie

$$\hat{\delta''}(q_1, w) = \hat{\delta}(q_1, w^{|w|})$$

Oczywiście, automat skończony nie może "z góry" znać funkcji $f_w$. Jednakże, korzystając z faktu, że możliwych funkcji $f_w$ jest skończenie wiele, możemy zastosować produkt automatów $A_f$ po wszystkich możliwych funkcjach $f$, jednocześnie obliczając "właściwą" funkcję $f_w$ przy pomocy automatu $A'$. Po przeczytaniu słowa $w$ pozostaje sprawdzić, czy na współrzędnej automatu produktowego odpowiadającej obliczonej funkcji $f_w$ pojawił się stan akceptujący.

Poniżej podamy formalną konstrukcję automatu $B$, takiego że

$$L(B) = \{w \,:\, w^{|w|} \in L\}$$

Ustawmy zbiór $Q^Q$ w ciąg: $Q^Q = \{g_1, \ldots, g_{n^n}\}$ (pamiętamy $n = |Q|$). Określamy

- zbiór stanów automatu $B$: $Q^{n+n^n}$,
- stan początkowy: $(q_1, q_1, \ldots, q_1)$,
- funkcja przejścia:

$$(p_1, \ldots, p_n, s_1, \ldots, s_{n^n}), \sigma \;\rightarrow\; (\delta(p_1, \sigma), \ldots, \delta(p_n, \sigma), g_1(s_1), \ldots, g_{n^n}(s_{n^n}))$$

- stany akceptujące:

$$\{(p_1, \ldots, p_n, s_1, \ldots, s_{n^n}) \,:\, (p_1, \ldots, p_n) = g_j \,\wedge\, s_j \in F \text{ dla pewngo } 1 \leq j \leq n^n\}$$

# 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.

Języki, Automaty i Obliczenia. Odcinek 3

# Gramatyki bezkontekstowe

Damian Niwiński

Październik 2003

## 1 Gramatyki i wyprowadzenia

Gramatyka bezkontekstowa może być przedstawiona jako następujący układ

$$G = (\Sigma, V, X_I, R)$$

gdzie

- $\Sigma$ jest skończonym zbiorem symboli, który nazywamy *alfabetem gramatyki*, elementy tego zbioru nazywamy też *symbolami końcowymi* gramatyki;
- $V$ jest skończonym zbiorem symboli rozłącznym z $\Sigma$; jego elementy nazywamy *symbolami niekońcowymi* albo *zmiennymi* gramatyki;
- $X_I \in V$ jest *symbolem startowym* gramatyki;
- $R$ jest skończonym zbiorem *reguł przepisywania* (krótko: reguł), $R \subseteq V \times (\Sigma \cup V)^*$.

Zgodnie z definicją, każda reguła jest parą postaci $(X, w)$, dla pewnej zmiennej $X \in V$ i słowa $w \in (\Sigma \cup V)^*$; mówimy że jest to reguła przepisywania zmiennej $X$. Fakt, że $(X, w) \in R$, będziemy także zapisywać $X \xrightarrow{G} w$. Jeżeli $\{(X, w_1), \ldots, (X, w_k)\}$, są wszystkimi regułami przepisywania zmiennej $X$, to będziemy je zapisywać w notacji skróconej, tzw. notacji Backusa-Naura,

$$X \to w_1 | w_2 | \ldots | w_k$$

Relację $R$ rozszerzymy do relacji $\xrightarrow{G} \subseteq (\Sigma \cup V)^* \times (\Sigma \cup V)^*$ określonej następująco

$$w \xrightarrow{G} v \;\Leftrightarrow\; (\exists (X, u) \in R)\, (\exists \alpha, \beta \in (\Sigma \cup V)^*)\, w = \alpha X \beta \wedge v = \alpha u \beta$$

Mówiąc intuicyjnie, relacja $w \xrightarrow{G} v$ zachodzi, kiedy słowo $v$ może być otrzymane przez jednokrotne zastosowanie reguły $(X, u)$ do pewnego wystąpienia zmiennej $X$ w słowie $w$.

Zauważmy, że gdy $X$ jest zmienną, to wprowadzona wyżej notacja $X \xrightarrow{G} w$ na oznaczenie reguły przepisywania zmiennej $X$ jest niesprzeczna z definicją relacji $\xrightarrow{G}$, gdyż $X \xrightarrow{G} w \;\Leftrightarrow\; (X, w) \in R$.

Każdy ciąg $w_0, w_1, \ldots, w_{k-1}, w_k$, gdzie $0 \leq k$, $w_0 = w$, $w_k = v$, a ponadto $w_i \stackrel{G}{\to} w_{i+1}$ dla $i = 0, \ldots, k-1$, nazywamy *wyprowadzeniem słowa $v$ ze słowa $w$* w gramatyce $G$. Określamy relację $w \stackrel{G*}{\to} v$ przez

$$w \stackrel{G*}{\to} v \iff \text{istnieje wyprowadzenie } v \text{ z } w$$

Nietrudno jest wykazać, że relacja $\stackrel{G*}{\to}$ jest zwrotno–przechodnim domknięciem relacji $\stackrel{G}{\to}$, tj. najmniejszą relacją zwrotną i przechodnią zawierającą $\stackrel{G}{\to}$. Mówimy, że słowo $w \in \Sigma^*$ jest *wyprowadzalne* ze zmiennej $X$ w gramatyce $G$, jeśli $X \stackrel{G*}{\to} w$. Przyjmujemy oznaczenie

$$L(G, X) = \{w \in \Sigma^* \,:\, X \stackrel{G*}{\to} w\}$$

Jeśli słowo $w \in \Sigma^*$ jest wyprowadzalne z symbolu startowego gramatyki, to mówimy krótko, że słowo $w$ jest *wyprowadzalne w gramatyce $G$*, a zbiór wszystkich takich słów określamy jako *język wyprowadzalny* w gramatyce $G$ lub *generowany* przez gramatykę $G$, który oznaczamy $L(G)$. Mamy więc

$$L(G) = L(G, X_I) = \{w \in \Sigma^* \,:\, X_I \stackrel{G*}{\to} w\}$$

W dalszym ciągu, jeżeli gramatyka $G$ będzie znana z kontekstu, będziemy często opuszczali górny indeks $G$, pisząc po prostu $w \to v$ lub $w \stackrel{*}{\to} v$.

**Przykład 1** (a) Następująca gramatyka nad alfabetem $\{(, ), [, ]\}$ generuje poprawnie zbudowane ciągi nawiasów

$$X \;\to\; \epsilon \,|\, XX \,|\, (X) \,|\, [X]$$

(b) Następująca gramatyka nad alfabetem $\{(, ), +, *, 0, 1\}$ generuje poprawnie zbudowane wyrażenia arytmetyczne

$$X \;\to\; 0 \,|\, 1 \,|\, X + X \,|\, X * X \,|\, (X)$$

## 1.1 Wyprowadzenia lewostronne

Określamy relację $\stackrel{lG}{\to}$ stanowiącą podzbior relacji $\stackrel{G}{\to}$,

$$w \stackrel{lG}{\to} v \;\Leftrightarrow\; (\exists (X, u) \in R)\,(\exists \alpha \in \Sigma^*)(\exists \beta \in (\Sigma \cup V)^*) w = \alpha X \beta \;\wedge\; v = \alpha u \beta$$

Mówiąc intuicyjnie, słowo $v$ zostało otrzymane ze słowa $w$ podobnie jak w relacji $w \stackrel{G}{\to} v$, ale wystąpienie zmiennej $X$ do jakiego została zastosowana reguła przepisywania $X \stackrel{G}{\to} u$ było najbardziej na lewo wysuniętym (ang. *left-most*) wystąpieniem jakiegokolwiek symbolu niekońcowego w słowie $w$. Ciąg $w = w_0, w_1, \ldots, w_{k-1}, w_k = v$, $0 \leq k$, t. że $w_i \stackrel{lG}{\to} w_{i+1}$, dla $i = 0, \ldots, k-1$, nazywamy *wyprowadzeniem lewostronnym* (ang. *left-most derivation*) słowa $v$ ze słowa $w$. Zwrotno-przechodnie domknięcie relacji $\stackrel{lG}{\to}$ będziemy oznaczać $\stackrel{lG*}{\to}$.

## 1.2 Drzewa wyprowadzeń

Struktura drzewa jest spotykana w informatyce we wielu kontekstach, przy czym używanych jest kilka nierównoważnych modeli matematycznych tego pojęcia. Na przyklad, jeśli zdefiniujemy drzewo jako spójny graf, w którym liczba wierzchołków jest o 1 większa od liczby krawędzi, to pominiemy taki aspekt drzewa jak wyróżnienie pewnego wierzchołka jako korzenia oraz uporządkowanie następników danego wierzchołka "od lewej do prawej". Dogodnym modelem uwzględniającym oba te aspekty jest przedstawienie drzewa jako pewnego zbioru słów, lub, w przypadku drzewa etykietowanego, funkcji ze zbioru słów w zbiór etykiet. Idea jest tu bardzo prosta: w drzewie, powiedzmy, binarnym, można utożsamić wierzchołek ze ścieżką jaką można do niego dojść od korzenia, tzn. ciągiem wyboru kierunków, np. *lewo, prawo, prawo* . W ogólniejszym przypadku, za zbiór kierunków możemy przyjąć dowolny segment początkowy zbioru liczb naturalnych.

Zbiór liczb naturalnych oznaczamy $\omega$. Zgodnie z naszą konwencją, $\omega^*$ oznacza zbiór słów nad $\omega$, a więc skończonych ciągów o wyrazach naturalnych. Jeśli $v \in \omega^*$, to każdy ciąg $vi$, $i \in \omega$, nazywamy *następnikiem* ciągu $v$. Jeśli $I \subseteq \omega^*$ i $v \in I$, to przez $\mathrm{succ}(I, v)$ oznaczamy zbiór tych następników ciągu $v$, które należą do $I$. Niepusty skończony zbiór $T \subseteq \omega^*$ nazywamy *drzewem* (nieetykietowanym), jeśli spełnione są następujące warunki

- $T$ jest zamknięty na prefiksy (tzn. $(w \in T \wedge v \leq w) \Rightarrow v \in T$)
- dla każdego $v \in T$, istnieje $k \in \omega$ (zależne od $v$), takie, że $\mathrm{succ}\,(T, v) = \{v1, \ldots, vk\}$ (jeśli $k = 0$, to $\mathrm{succ}(T, v) = \emptyset$).

Elementy drzewa $T$ nazywamy wierzchołkami, słowo $\epsilon$ (które oczywiście znajduje się w każdym drzewie) nazywamy *korzeniem*, a wierzchołki $v$, dla których $\mathrm{succ}(T, v) = \emptyset$, nazywamy *liśćmi* drzewa $T$. Zauważmy, że relacja "być prefiksem"częściowo porządkuje zbiór wierzchołków drzewa[1]. Liczbę

$$\mathrm{depth}(T) = \max\{|w| \,:\, w \in \mathrm{dom}(T)\}$$

nazywamy *głębokością* drzewa $T$.

Jeśli $E$ jest dowolnym zbiorem, a $T$ jest drzewem, to każdą funkcję $t : T \to E$ nazywamy *drzewem etykietowanym* zbiorem etykiet $E$. Zbiór $T$ nazywamy wówczas *dziedziną* drzewa $t$ i oznaczamy $\mathrm{dom}(t)$ (z angielskiego *domain* = dziedzina). Pojęcia "korzeń", "liść" itd. używane w odniesieniu do drzewa etykietowanego, odnoszą się oczywiście do $\mathrm{dom}(t)$.

Jeśli $t : \mathrm{dom}(t) \to E$ jest drzewem i $w \in \mathrm{dom}(t)$, to *poddrzewem drzewa $t$ wyznaczonym przez wierzchołek $w$*, nazywamy drzewo $t.w$ określone następująco:

- $\mathrm{dom}(t.w) = \{w\}^{-1}\,\mathrm{dom}(t) = \{u : wu \in \mathrm{dom}(t)\}$,
- $t.w(u) = t(wu)$, dla $u \in \mathrm{dom}(t.w)$.

[1]Porządek ten można go rozszerzyć do porządku liniowego, np. porządku leksykograficznego na $\omega^*$ indukowanego przez standardowy porządek na $\omega$.

Zdefiniujemy teraz operację *podstawienia*, która, mówiąc intuicyjnie, pozwoli zastępować liście drzewa innymi drzewami. Niech $t : \mathrm{dom}(t) \to E$ będzie drzewem etykietowanym i niech $v_1, \ldots, v_m$ będą różnymi liśćmi drzewa $t$ (niekoniecznie wszystkimi). Niech $t_i : \mathrm{dom}(t_i) \to E$, $i = 1, \ldots, m$, będą etykietowanymi drzewami. Określimy drzewo $t[v_1 \leftarrow t_1, \ldots, v_m \leftarrow t_m]$, w skróconej notacji $t[\vec{v} \leftarrow \vec{t}]$, które stanowi rezultat jednoczesnego podstawienia w drzewie $t$ drzewa $t_i$ na wierzchołek $v_i$ odpowiednio, dla $i = 1, \ldots, m$.

- $\mathrm{dom}(t[\vec{v} \leftarrow \vec{t}]) = \mathrm{dom}(t) \cup \bigcup_{i=1,\ldots,m} v_i \mathrm{dom}(t_i)$
- $t[\vec{v} \leftarrow \vec{t}](w) = t(w)$, dla $w \in \mathrm{dom}(t) - \{v_1, \ldots, v_m\}$
- $t[\vec{v} \leftarrow \vec{t}](v_i u) = t_i(u)$, dla każdego $u \in \mathrm{dom}(t_i)$, $i = 1, \ldots, m$.

Opierając się na definicji, nie jest trudno sprawdzić, że operacja podstawienia jest łączna w następującym sensie: niech $t, r, s$ będą drzewami i niech $v \in \mathrm{dom}(t)$, $u \in \mathrm{dom}(r)$. Wówczas:

$$(t[v \leftarrow r])[vu \leftarrow s] = t[v \leftarrow r[u \leftarrow s]]$$

Niech $G = (\Sigma, V, X_I, R)$ będzie gramatyką i niech $Y \in V$. Drzewo $D : \mathrm{dom}(D) \to \Sigma \cup V \cup \{\epsilon\}$ nazwiemy *drzewem wyprowadzenia* ze zmiennej $Y$, jeśli spełnione są następujące warunki.

1. $D(\epsilon) = Y$;
2. dla każdego $w \in \mathrm{dom}(D)$, jeśli $D(w) \in \Sigma \cup \{\epsilon\}$, to $w$ jest liściem;
3. dla każdego $w \in \mathrm{dom}(D)$, jeśli $D(w) \in V$, powiedzmy $D(w) = Z$, to zachodzi jedna z następujących możliwości:
   (a) $w$ jest liściem;
   (b) istnieje reguła $Z \xrightarrow{G} \epsilon$ i $w$ ma dokładnie jeden następnik $w1$, przy czym $D(w1) = \epsilon$;

(c) istnieje reguła $Z \stackrel{G}{\to} c_1 \dots c_k$, gdzie $c_1, \dots, c_k \in \Sigma \cup V$ i $w$ ma dokładnie $k$ następników: $w1, \dots, wk$, oraz $D(w\ell) = c_\ell$, dla $\ell = 1, \dots, k$.

W przypadkach (3b) i (3c) mówimy, że w wierzchołku $w$ została zastosowana reguła $Z \stackrel{G}{\to} \epsilon$ lub $Z \stackrel{G}{\to} c_1 \dots c_k$, odpowiednio.

*Plonem* drzewa wyprowadzenia $D$ nazwiemy słowo $\alpha \in (\Sigma \cup V)^*$, jakie można otrzymać z drzewa składając etykiety wszystkich liści w porządku "od lewej do prawej". Bardziej dokładnie, niech $v_1, \dots, v_m$ będą wszystkimi liśćmi drzewa $D$ w porządku leksykograficznym[2]. Określamy słowo $y(D) \in (\Sigma \cup V)^*$ (z angielskiego *yield*), nazywane *plonem drzewa* $D$,

$$y(D) \;=\; D(v_1) \dots D(v_m)$$

Drzewo wyprowadzenia nazwiemy maksymalnym, jeśli nie zachodzi przypadek 3a, tzn. żaden liść nie jest etykietowany zmienną. Nietrudno jest wykazać, że jeśli $D$ jest maksymalnym drzewem wyprowadzenia, to $y(D) \in \Sigma^*$.

Zbiór wszystkich drzew wyprowadzenia w gramatyce $G$ ze zmiennej $X$ będziemy oznaczali przez $\mathcal{D}_G(X)$, zbiór wszystkich drzew wyprowadzenia przez $\mathcal{D}_G$.

**Lemat 1** *Niech $G = (\Sigma, V, X_I, R)$ będzie gramatyką, $X \in V$ i $w \in (\Sigma \cup V)^*$. Następujące warunki są równoważne:*

**(i)** $X \stackrel{G*}{\to} w$,

**(ii)** *istnieje drzewo wyprowadzenia $D$ ze zmiennej $X$, takie że $y(D) = w$.*

*Dowód.*

$(i) \Rightarrow (ii)$

Niech $X = w_0 \stackrel{G}{\to} w_1 \stackrel{G}{\to} \dots w_{k-1} \stackrel{G}{\to} w_k = w$ będzie wyprowadzeniem słowa $w$ ze zmiennej $X$. Dla $\ell = 0, 1, \dots, k$, określamy kolejno drzewa wyprowadzenia $D_\ell$, takie, że $y(D_\ell) = w_\ell$.

Drzewo $D_0$ określamy następująco: $\mathrm{dom}(D_0) = \{\epsilon\}$, $D_0(\epsilon) = X$.

Przypuśćmy, że jest już określone drzewo $D_\ell$, $\ell < k$ i $y(D_\ell) = w_\ell$. Z definicji wyprowadzenia mamy $w_\ell = \alpha Y \beta$ i $w_{\ell+1} = \alpha u \beta$, gdzie $Y \stackrel{G}{\to} u$ jest regułą przepisywania gramatyki $G$, a $\alpha, \beta \in (\Sigma \cup V)^*$. Załóżmy, że $w_\ell = s_1 \dots s_m$, gdzie $s_1, \dots, s_m \in \Sigma \cup V$, $\alpha = s_1 \dots s_{i-1}$, $s_i = Y$ i $\beta = s_{i+1} \dots s_m$, dla pewnego $i$, $1 \le i \le m$ (gdy $i = 1$, $\alpha = \epsilon$, gdy $i = m$, $\beta = \epsilon$). Niech $v_1, \dots, v_{m'}$ będą tymi spośród liści drzewa $D_\ell$, dla których $D_\ell(v_i) \neq \epsilon$, w porządku leksykograficznym. Jest jasne, że $m' = m$ i $D_\ell(v_j) = s_j$, dla $j = 1, \dots, m$. W szczególności, $D_\ell(v_i) = Y$.

Rozróżnimy teraz dwa przypadki.

1. $u = \epsilon$. Określamy drzewo $D_{\ell+1}$ następująco: $\mathrm{dom}(D_{\ell+1}) = \mathrm{dom}(D_\ell) \cup \{v_i 1\}$, $D_{\ell+1}(v) = D_\ell(v)$ dla $v \in \mathrm{dom}(D_\ell)$ oraz $D_{\ell+1}(v_i 1) = \epsilon$.

[2]Liść $v$ leksykograficznie poprzedza liść $w$ jeśli istnieją $u \in \omega^*$, $i, j \in \omega$ takie, że $ui \le v$, $uj \le w$ i $i < j$.

2. $u = u_1 \dots u_r$, $r > 0$. Określamy drzewo $D_{\ell+1}$ następująco: $\mathrm{dom}(D_{\ell+1}) = \mathrm{dom}(D_\ell) \cup \{v_i 1, \dots, v_i r\}$, $D_{\ell+1}(v) = D_\ell(v)$ dla $v \in \mathrm{dom}(D_\ell)$ oraz $D_{\ell+1}(v_i 1) = u_1, \dots, D_{\ell+1}(v_i r) = u_r$.

Nietrudno sprawdzić, że $D_{\ell+1}$ jest drzewem wyprowadzenia i $y(D_{\ell+1}) = w_{\ell+1}$.

$(ii) \Rightarrow (i)$

Należy wykazać, że dla każdego drzewa wyprowadzenia $D \in \mathcal{D}_X$, $X \stackrel{G*}{\to} y(D)$. Dowód poprowadzimy przez indukcję ze względu na głębokość drzewa $D$. Gdy $D$ jest drzewem o głębokości 0, to $X = y(D)$ i oczywiście $X \stackrel{G*}{\to} y(D)$. Załóżmy, że $\mathrm{depth}(D) \geq 1$, a więc w korzeniu drzewa $D$ została zastosowana pewna reguła $X \stackrel{G}{\to} u$. Wówczas zachodzi jeden z dwóch przypadków.

1. $u = \epsilon$ i korzeń drzewa $D$ (którym jest również $\epsilon$) ma dokładnie jeden następnik, mianowicie 1, przy czym $D(1) = \epsilon$.

2. $u = u_1 \dots u_r$, gdzie $r > 0$, $u_1, \dots, u_r \in \Sigma \cup V$ i korzeń $\epsilon$ ma dokładnie $r$ następników w drzewie $D$, mianowicie $1, \dots, r$, przy czym $D(i) = u_i$, dla $i = 1, \dots, r$.

W pierwszym przypadku, oczywiście $X \stackrel{G*}{\to} y(D) = \epsilon$.

Załóżmy drugi przypadek. Zauważmy najpierw, że o ile $u_i = Y$, to poddrzewo $D.i$ wyznaczone przez wierzchołek $i$ jest pewnym drzewem wyprowadzenia ze zmiennej $Y$, przy czym $\mathrm{depth}(D.i) < \mathrm{depth}(D)$, a zatem, z założenia indukcyjnego, $Y \stackrel{G*}{\to} y(D.i)$. Z kolei $u$ możemy przedstawić $u = \alpha_0 Z_1 \alpha_1 Z_2 \alpha_2 \dots Z_m \alpha_m$, gdzie, dla $i = 0, 1, \dots, m$, $\alpha_i \in \Sigma^*$ (tzn. słowa $\alpha_i$ nie zawierają zmiennych). Z definicji plonu, łatwo wynika, że

$$y(D) = \alpha_0 y(D_1) \alpha_1 y(D_2) \alpha_2 \dots y(D_\ell) \alpha_\ell$$

gdzie $D_i$ jest poddrzewem wyznaczonym przez wierzchołek (następnik korzenia) etykietowany $Z_i$.

Pamiętając o tym iż, jak już zauważyliśmy, $Z_i \stackrel{G*}{\to} y(D_i)$, dla $i = 1, \dots, \ell$, otrzymujemy

$$\begin{array}{rcl} X & \stackrel{G}{\to} & \alpha_0 Z_1 \alpha_1 Z_2 \alpha_2 \dots Z_\ell \alpha_\ell \\ & \stackrel{G*}{\to} & \alpha_0 y(D_1) \alpha_1 Z_2 \alpha_2 \dots Z_\ell \alpha_\ell \\ & \stackrel{G*}{\to} & \alpha_0 y(D_1) \alpha_1 y(D_2) \alpha_2 \dots Z_\ell \alpha_\ell \\ . & . & . \\ & \stackrel{G*}{\to} & \alpha_0 y(D_1) \alpha_1 y(D_2) \alpha_2 \dots y(D_\ell) \alpha_\ell \\ & = & y(D) \end{array}$$

□

Kiedy słowo $w$ nie zawiera symboli nieterminalnych, to udowodniony przed chwilą lemat możemy rozszerzyć o jeszcze jeden równoważny warunek: istnieje wyprowadzenie *lewostronne* słowa $w$.

**Twierdzenie 1** *Niech $G = (\Sigma, V, X_I, R)$ będzie gramatyką, i niech $w \in \Sigma^*$. Następujące warunki są równoważne:*

**(i)** *istnieje wyprowadzenie słowa $w$ ze zmiennej $X$,*

**(ii)** *istnieje maksymalne drzewo wyprowadzenia ze zmiennej $X$, którego plonem jest $w$,*

**(iii)** *istnieje wyprowadzenie lewostronne słowa $w$ ze zmiennej $X$.*

*Dowód.* Implikacja $(i) \Rightarrow (ii)$ została udowodniona w Lemacie 1 (maksymalność drzewa jest oczywista, gdy plon nie zawiera symboli niekońcowych), a implikacja $(iii) \Rightarrow (i)$ jest oczywista. Dla dowodu implikacji $(ii) \Rightarrow (iii)$, wystarczy wykazać, że dla każdego drzewa wyprowadzenia $D \in \mathcal{D}_X$, takiego że $y(D) \in \Sigma^*$, $X \overset{lG*}{\rightarrow} y(D)$. Dowód jest podobny do dowodu analogicznego faktu w dowodzie implikacji $(ii) \Rightarrow (i)$ Lematu 1. W odpowiednim miejscu, kiedy dokonujemy rozkładu $y(D) = \alpha_0 y(D_1)\alpha_1 y(D_2)\alpha_2 \dots y(D_\ell)\alpha_\ell$, wystarczy zauważyć, że drzewa $D_i$ są wtedy maksymalne i z założenia indukcyjnego mamy lewostronne wyprowadzenia $Z_i \overset{lG*}{\rightarrow} y(D_i)$. □

# 2 Postaci normalne gramatyk bezkontekstowych

W tym rozdziale przedstawimy efektywną metodę przekształcania gramatyki bezkontekstowej w równoważną gramatykę o stosunkowo prostej postaci, tzw. postaci normalnej Chomsky'ego. Jako produkt uboczny naszych rozważań uzyskamy także algorytm odpowiadający na pytanie, czy język generowany przez gramatykę jest niepusty.

## 2.1 Eliminacja symboli nieużytecznych

Mówiąc intuicyjnie, zmienną w gramatyce można uznać za nieużyteczną, kiedy nie da się jej osiągnąć w żadnym wyprowadzeniu z symbolu startowego, lub kiedy *z niej* nie można wyprowadzić żadnego słowa złożonego z symboli końcowych.

Niech $G = (\Sigma, V, X_I, R)$ będzie gramatyką. Powiemy, że zmienna $Y$ jest *osiągalna* ze zbioru zmiennych $E \subseteq V$, jeśli istnieją $X \in E$ oraz $\alpha, \beta \in (\Sigma \cup V)^*$, takie że $X \stackrel{G*}{\to} \alpha Y \beta$.

Podobnie, jak to było dla automatu, określimy funkcję, dla $E \subseteq V$

$$\mathit{Step}(E) = \{Y \,:\, (\exists X \in E)(\exists \alpha, \beta \in (\Sigma \cup V)^*) X \stackrel{G}{\to} \alpha Y \beta\}$$

Poniższy algorytm, podobny do Algorytmu 3 z Odcinka 1, oblicza zbiór zmiennych osiągalnych z $E$.

**Algorytm 5**
**Dane:** Gramatyka $G$, zbiór zmiennych $E \subseteq V$
1. $\mathcal{X} := \emptyset; \mathcal{X}_1 := I;$
2. **Dopóki** $(\mathcal{X}_1 \neq \emptyset)$ **wykonuj**
3. $\quad \mathcal{X} := \mathcal{X} \cup \mathcal{X}_1;$
4. $\quad \mathcal{X}_1 := \mathit{Step}(\mathcal{X}_1) \setminus \mathcal{X};$

**Wynik:** zbiór $\mathcal{X}$

Zauważmy, że linia 3 może być wykonana co najwyżej $|V|$ razy (za każdym razem dokładamy coś do $\mathcal{X}$), zatem algorytm na pewno się zatrzymuje po wykonaniu nie więcej niż $|V|$ obrotów pętli 2. Ponadto, łatwo jest widzieć, że, na każdym etapie wykonywania algorytmu, wszystkie zmienne w zbiorze stanowiącym wartość zmiennej $\mathcal{X}$, są osiągalne z $E$. Pozostaje wykazać, że w końcowej wartości zmiennej $\mathcal{X}$ znajdują się już wszystkie zmienne osiągalne ze zbioru $E$. Istotnie, przypuśćmy, że zmienna $Z$ jest osiągalna z $Y \in E$ przy pomocy wyprowadzenia

$$Y = w_0 \stackrel{G}{\to} w_1 \stackrel{G}{\to} \ldots \stackrel{G}{\to} w_{k-1} \stackrel{G}{\to} w_k = \alpha Z \beta$$

Wówczas można zaobserwować, że jeśli po wykonaniu pętli (2.) $i$ razy wszystkie zmienne występujące w słowie $w_\ell$ znajdują się w wartości zmiennej $\mathcal{X}$, to każda ze zmiennych występujących w słowie $w_{\ell+1}$ znajdzie się w w wartości zmiennej $\mathcal{X}$ nie później niż po wykonaniu pętli $i+1$ razy. W szczególności zmienna $Z$ znajdzie się w końcowej wartości zmiennej $\mathcal{X}$.

Z kolei, powiemy, że zmienna $X \in V$ jest *produktywna*, jeżeli $X \stackrel{G*}{\to} w$, dla pewnego $w \in \Sigma^*$. Niech, dla $u \in (\Sigma \cup V)^*$, $\mathrm{var}(u)$ oznacza zbiór wszystkich zmiennych, które występują w $u$. Określimy następującą funkcję na $P(V)$,

$$\mathit{BackStep}(E) = \{X \,:\, (\exists u \in (\Sigma \cup V)^*)(X \xrightarrow{G} u \wedge \mathrm{var}(u) \subseteq E)\}$$

Zauważmy, że jeżeli wszystkie zmienne w zbiorze $E$ są produktywne, to również wszystkie zmienne w zbiorze $\mathit{BackStep}(E)$ są produktywne. Zauważmy także, że $\mathit{BackStep}(\emptyset)$ jest zbiorem tych zmiennych, z których w jednym kroku można wygenerować słowo złożone z symboli końcowych.

Następujący algorytm oblicza zbiór wszystkich zmiennych produktywnych:

**Algorytm 6**
**Dane:** Gramatyka $G$
1. $\mathcal{X} := \emptyset$; $\mathcal{X}_1 := \mathit{BackStep}\ (\emptyset)$;
2. **Dopóki** $(\mathcal{X}_1 \neq \emptyset)$ **wykonuj**
3. $\quad\quad \mathcal{X} := \mathcal{X} \cup \mathcal{X}_1$;
4. $\quad\quad \mathcal{X}_1 := \mathit{BackStep}(\mathcal{X}) \setminus \mathcal{X}$;
**Wynik:** zbiór $\mathcal{X}$

Tu również, podobnie jak w algorytmie 5, linia 3 może być wykonana co najwyżej $|V|$ razy, zatem algorytm na pewno się zatrzymuje po wykonaniu nie więcej niż $|V|$ obrotów pętli 2. Wykażemy, że końcowa wartość zmiennej $\mathcal{X}$ jest zbiorem wszystkich zmiennych produktywnych.

Istotnie, z własności funkcji *BackStep* wynika, że wartości, jakie zmienna $\mathcal{X}$ przyjmuje w trakcie wykonywania algorytmu są zbiorami zmiennych produktywnych. Pozostaje wykazać, że końcowa wartość zmiennej $\mathcal{X}$ obejmuje już wszystkie zmienne produktywne. W tym celu przypuśćmy, że $Y$ jest zmienną produktywną i

$$Y = w_0 \xrightarrow{G} w_1 \ldots w_{k-1} \xrightarrow{G} w_k = w$$

jest wyprowadzeniem pewnego słowa $w \in \Sigma^*$. Wówczas można zaobserwować, że jeśli po wykonaniu pętli (2.) $i$ razy wszystkie zmienne występujące w słowie $w_\ell$, $\ell > 0$, znajdują się w wartości zmiennej $\mathcal{X}$, to każda ze zmiennych występujących w słowie $w_{\ell-1}$ znajdzie się w w wartości zmiennej $\mathcal{X}$ nie później niż po wykonaniu pętli $i+1$ razy. W szczególności zmienna $Y$ znajdzie się w końcowej wartości zmiennej $\mathcal{X}$.

Zauważmy, że $L(G) \neq \emptyset$ wtedy i tylko wtedy, gdy symbol startowy $X_I$ jest zmienną produktywną. Mamy więc

**Fakt 1** *Istnieje algorytm, który, dla danej gramatyki bezkontekstowej $G$ rozstrzyga, czy $L(G) \neq \emptyset$.* □

Przedstawione wyżej algorytmy zastosujemy teraz dla sprowadzenia gramatyki do takiej postaci, w której nie ma symboli nieużytecznych.

Mówimy, że gramatyki $G$ i $H$ są równoważne, kiedy $L(G) = L(H)$.

**Fakt 2** *Dla każdej gramatyki $G$ takiej, że $L(G) \neq \emptyset$ można skonstruować równoważną gramatykę $G'$, w której każda zmienna jest produktywna i osiągalna z symbolu początkowego.*

*Dowód.* Jeśli $G = (\Sigma, V, X_I, R)$ jest gramatyką i $E \subseteq V$, $X_I \in E$, to przez *obcięcie* gramatyki $G$ do zbioru $E$ rozumiemy gramatykę $G|E =_{df} (\Sigma, E, X_I, R')$, gdzie $R' = R \cap (E \times (\Sigma \cup E)^*)$. (Innymi słowy, reguła $Y \to w$ gramatyki $G$ pozostaje regułą gramatyki $G|E$, o ile $Y \in E$ i $\mathrm{var}(w) \subseteq E$.) Niech $E$ będzie zbiorem wszystkich zmiennych produktywnych gramatyki $G$. Z założenia, $X_I \in E$. Oczywiście, gramatyka $G|E$ jest równoważna gramatyce $G$. (Wszystkie zmienne występujące w wyprowadzeniu słowa $w \in \Sigma^*$ w gramatyce $G$ muszą być produktywne.) Z kolei, niech $E' \subseteq E$ będzie zbiorem tych zmiennych, które są osiągalne z $X_I$ w gramatyce $G|E$. Rozważmy gramatykę $(G|E)|E'$. Łatwo widzieć, że gramatyka ta jest równoważna gramatyce $G|E$ (a więc i $G$) i, oczywiście, wszystkie zmienne tej gramatyki są w niej osiągalne z $X_I$. Z drugiej strony, nietrudno wykazać, że każda zmienna z $E'$, która była produktywna w $G$, jest nadal produktywna w $(G|E)|E'$. Zatem gramatyka $(G|E)|E'$ spełnia żądane warunki. □

**Przykład 2** Niech

$$G = (\{a\}, \{X, Y, Z\}, X, \{(X, \epsilon), (X, YZ), (Z, \epsilon)\})$$

Wszystkie zmienne są osiągalne z $X$, ale zmienna $Y$ jest nieproduktywna. Stosując konstrukcję z powyższego dowodu, otrzymujemy

$$G|E = (\{a\}, \{X, Z\}, X, \{(X, \epsilon), (Z, \epsilon)\})$$

i

$$(G|E)|E' = (\{a\}, \{X\}, X, \{(X, \epsilon)\})$$

Zauważmy, że gdybyśmy odwrócili kolejność wykonywanych operacji i *najpierw* ograniczyli gramatykę $G$ do zmiennych osiągalnych z $X$, a *następnie* do zmiennych produktywnych, to otrzymalibyśmy kolejno gramatyki $G$ i $G|E$, a zatem taka konstrukcja nie byłaby poprawna.

## Eliminacja trywialnych reguł

Regułę postaci $X \xrightarrow{G} \epsilon$ nazywamy krótko *$\epsilon$-regułą*.

**Fakt 3 (eliminacja $\epsilon$-reguł)** *Dla każdej gramatyki $G$, można skonstruować gramatykę $G'$ bez $\epsilon$-reguł, taką, że $L(G') = L(G) - \{\epsilon\}$.*

*Dowód.* Niech $G = (\Sigma, V, X_I, R)$ będzie gramatyką. Zmienną $X \in V$ nazwiemy *zerowalną*, jeśli $X \xrightarrow{G*} \epsilon$. Zbiór zmiennych zerowalnych można wyznaczyć efektywnie przy pomocy procedury podobnej do algorytmu obliczania zbioru zmiennych produktywnych (Algorytm 6); szczegóły pozostawiamy Czytelnikowi.

Zauważmy, że nie wystarczy po prostu usunąć wszystkie $\epsilon$-reguły, bo w ten sposób moglibyśmy przy okazji uniemożliwić niektóre wyprowadzenia słów różnych od $\epsilon$ (np. gdy produkcje są $X \to YZ$, $Y \to \epsilon$, $Z \to a$, to, usuwając regułę $Y \to \epsilon$, uniemożliwimy wyprowadzenie $X \xrightarrow{*} a$).

Przedstawimy dwa sposoby, z których pierwszy jest bardziej bezpośredni i łatwy do zrozumienia, ale drugi bardziej ekonomiczny.

**Sposób 1**

Usuwamy ze zbioru $R$ wszystkie $\epsilon$-reguły, a następnie, dla każdej reguły $X \to \alpha$ dodajemy wszystkie reguły, jakie można utworzyć zastępując w $\alpha$ dowolne zmienne zerowalne przez $\epsilon$ z wyjątkiem (jeśli taka reguła mogłaby powstać) reguły $X \to \epsilon$.

Dowód, że, dla każdego $w \neq \epsilon$,

$$X \overset{G*}{\to} w \Leftrightarrow X \overset{G'*}{\to} w$$

można przeprowadzić przez indukcję ze względu na głębokość drzewa wyprowadzenia.

**Sposób 2**

Objaśnimy go na przykładzie. Tu również na początku usuwamy $\epsilon$-reguły. Dalej, przypuśćmy, że mamy regułe $X \to Z_1 \dots Z_m$ i wszystkie zmienne $Z_1, \dots, Z_m$ są zerowalne. Wprowadzamy nowe zmienne $[Z_1 Z_2 \dots Z_m]$, $[Z_2 Z_3 \dots Z_m]$, …, $[Z_{m-1} Z_m]$, $[Z_m]$ i dodajemy następujące reguły:

$$\begin{array}{rcll} X & \to & Z_i & i = 1, \dots, m \\ X & \to & Z_i[Z_{i+1} Z_{i+2} \dots Z_m] & i = 1, \dots, m \\ [Z_i Z_{i+1} \dots Z_m] & \to & Z_j & i = 1, \dots, m-1,\ j = i, i+1, \dots, m \\ [Z_i Z_{i+1} \dots Z_m] & \to & Z_j[Z_{j+1} Z_{j+2} \dots Z_m] & i = 1, \dots, m-1,\ j = i, i+1, \dots, m \end{array}$$

Zauważmy, że stosując pierwszy sposób, zastąpilibyśmy regułe $X \to Z_1 \dots Z_m$ przez $2^m - 1$ reguł, a przy drugim sposobie przez $O(m^2)$ reguł. □

Regułę postaci $X \overset{G}{\to} Y$ nazywamy *regułą jednostkową*.

**Lemat 2** *Dla każdej gramatyki $G$, takiej, że $L(G) \neq \emptyset$ i $\epsilon \notin L(G)$, można skonstruować równoważną gramatykę bez reguł jednostkowych, bez $\epsilon$-reguł i taką, że każda zmienna jest produktywna i osiągalna z symbolu początkowego.*

*Dowód.* Niech $G = (\Sigma, V, X_I, R)$ będzie gramatyką. Na mocy Faktu 3, możemy założyć, że $G$ jest już bez $\epsilon$-reguł.

Usuwamy z $R$ wszystkie reguły jednostkowe, natomiast dla każdych $X, Y$, jeśli $X \overset{G*}{\to} Y$ i $Y \to \alpha$, gdzie $\alpha$ jest różne od zmiennej, dokładamy regułę $X \to \alpha$. Zauważmy, że możemy łatwo rozstrzygnąć czy $X \overset{G*}{\to} Y$ w $|V|$ krokach, ponieważ nie ma $\epsilon$-reguł.

Niech $R'$ będzie otrzymanym zbiorem reguł.

Oczywiście, nie dodaliśmy żadnej $\epsilon$–reguły.

Dla zapewnienia ostatniego z warunków Lematu, stosujemy procedurę z dowodu Faktu 2, która eliminuje ewentualne symbole nieużyteczne. □

## 2.2 Twierdzenia o postaci normalnej gramatyk

**Twierdzenie 2 (o postaci normalnej Chomsky'ego)** *Dla każdej gramatyki $G$, takiej, że $L(G) \neq \emptyset$ i $\epsilon \notin L(G)$, można skonstruować równoważną gramatykę, w której każda reguła jest postaci $X \to \sigma$, gdzie $\sigma \in \Sigma$, lub $X \to YZ$, gdzie $Y, Z \in V$ ($X, Y, Z$ nie muszą być różne).*

*Dowód.* Dla każdego $\sigma \in \Sigma$, tworzymy nową zmienną $Z_\sigma$ i do zbioru reguł gramatyki $G$ dokładamy regułę $Z_\sigma \to \sigma$. Na mocy Lematu 2, możemy założyć, że każda reguła gramatyki $G$, która nie jest postaci $X \to \sigma$, jest postaci $X \to \alpha$, $|\alpha| \geq 2$. Modyfikujemy tę regułę zastępując w $\alpha$ każde $\sigma \in \Sigma$ przez $Z_\sigma$. W ten sposób otrzymujemy gramatykę, w której każda reguła jest postaci $X \to \sigma, \sigma \in \Sigma$, lub $X \to Z_1 \dots Z_m$, gdzie $Z_1, \dots, Z_m \in V$, $m \geq 2$. Jeżeli $m > 2$, to wprowadzamy nowe zmienne $[Z_2 Z_3 \dots Z_m]$, $\dots$, $[Z_{m-1} Z_m]$ i reguły

$$\begin{array}{rcll} X & \to & Z_1[Z_2 Z_3 \dots Z_m] & \\ [Z_i \dots Z_m] & \to & Z_i[Z_{i+1} \dots Z_m] & \text{dla } i < m-1 \\ [Z_{m-1} Z_m] & \to & Z_{m-1} Z_m & \end{array}$$

□

**Pytanie.** Jeśli gramatyka $G$ jest w postaci Chomsky'ego, to jak możemy oszacować z góry długość wyprowadzenia słowa $w \in L(G)$, w zależności od długości $w$ ?

W uzupełnieniu powyższego twierdzenia, podamy dla informacji, bez dowodu, inne twierdzenie o postaci normalnej, którego autorem jest pani S.Greibach.

**Twierdzenie 3 (o postaci normalnej Greibach)** *Dla każdej gramatyki $G$, takiej, że $L(G) \neq \emptyset$ i $\epsilon \notin L(G)$, można skonstruować równoważną gramatykę, w której każda reguła jest postaci $X \to \sigma\alpha$, gdzie $\sigma \in \Sigma$ i $\alpha \in V^*$.*

## 2.3 Algorytm rozpoznawania języków bezkontekstowych

Rozważmy następujący problem:

> **Dane** Gramatyka $G$, słowo $w \in \Sigma^*$
> **Pytanie** Czy $w \in L(G)$?

Jeśli $w = \epsilon$ to problem możemy rozwiązać obliczając zbiór zmiennych $\epsilon$-produktywnych, tzn. takich, że $X \stackrel{G*}{\to} \epsilon$ podobnie jak obliczaliśmy zbiór zmiennych produktywnych w Algorytmie 6 (str. 9), z tym, że w funkcji *Back-Step* bierzemy pod uwagę tylko te reguły w których nie występują w ogóle symbole terminalne.

Ciekawszy jest przypadek, gdy $|w| > 0$.

Moglibyśmy wówczas znaleźć gramatykę Chomsky'ego $G'$ generującą $L(G) - \{\epsilon\}$ i przeszukiwać wszystkie wyprowadzenia o ograniczonej długości (por. "Pytanie" powyżej). Metoda ta byłaby jednak zupełnie nieefektywna.

Poniżej naszkicujemy algorytm zaproponowany przez Cocke'a, Youngera i Kasamiego, i od nazwisk autorów zwany algorytmem CYK, który rozwiązuje zadanie w czasie wielomianowym.

W algorytmie tym zakładamy, że $G = (\Sigma, V, X_I, R)$ jest już w postaci Chomsky'ego (por. Ćwiczenie 12 poniżej). Niech $w = w_1 \dots w_n$, gdzie $w_1, \dots, w_n \in \Sigma$. Główną pracą algorytmu jest indukcyjne obliczanie tablicy zbiorów $V_{i,j}$, $i = 1, \dots, n$, $j = 1, \dots, n+1-i$, gdzie

$$V_{i,j} = \{X \,:\, X \stackrel{G*}{\to} w_i w_{i+1} \dots w_{i+j-1}\} \quad (*)$$

Następująca obserwacja jest kluczowa dla algorytmu:

jeśli $Y \in V_{i,j}$, $Z \in V_{i+j,k}$ i $X \xrightarrow{G} YZ$, to $X \in V_{i,k}$.

Ostateczna odpowiedź na pytanie zależy oczywiście od tego, czy $X_I \in V_{1,n}$. Przyjmijmy oznaczenie, dla zbiorów zmiennych $A, B \subseteq V$,

$$\mathit{Fusion}(A,B) = \{X \,:\, (\exists Y,Z)\, X \xrightarrow{G} YZ \wedge Y \in A \wedge Z \in B\}$$

Wówczas algorytm CYK można przedstawić następująco.

**Algorytm CYK**

**Dane:** Gramatyka $G$ w postaci Chomsky'ego, słowo $w = w_1 \dots w_n$

1. Dla $i = 1, \dots, n$, wykonuj : $V_{i,1} := \{X \,:\, X \xrightarrow{G} w_i\}$;
2. Dla $j = 2, \dots, n$, wykonuj :
3. Dla $i = 1, \dots, n+1-j$, wykonuj :
4. $V_{i,j} := \emptyset$;
5. Dla $k = 1, \dots, j-1$, wykonuj :
6. $V_{i,j} := V_{i,j} \cup \mathit{Fusion}(V_{i,k}, V_{i+k,j-k})$.

Nietrudno wykazać, że po wykonaniu dla danego $j$ instrukcji 3–6, zawartość zmiennej $V_{i,j}$ (dla każdego $i$, takiego że $j \leq n+1-i$) spełnia warunek (*). Zatem $w \in L(G)$ wtedy i tylko wtedy, gdy po zakończeniu pracy algorytmu $X_I \in V_{1,n}$.

Czas wykonania operacji *Fusion* przy odpowiedniej implementacji możemy oszacować przez rozmiar gramatyki rozumiany jako łączna długość wszystkich produkcji, $|G|$. Zatem cały algorytm działa w czasie $\mathcal{O}(n^3 \cdot |G|)$, gdzie $n = |w|$.

## Ćwiczenia

1. Podać gramatyki bezkontekstowe generujące następujące języki:
   (a) zbiór słów nad alfabetem $\{a, b\}$, które zawierają tyle samo $a$ co $b$;
   (b) zbiór słów nad alfabetem $\{a, b\}$, które zawierają dwa razy więcej $a$ niż $b$;
   (c) zbiór wyrażeń arytmetycznych nad alfabetem $\{0, 1, (, ), +, \cdot\}$, które, przy zwykłej interpretacji działań dla liczb naturalnych, mają wartość 3;
   (d) zbiór wyrażeń arytmetycznych w notacji polskiej (nad alfabetem $\{0, 1, +, \cdot\}$) o wartości 4;
   (e) zbiór poprawnie zbudowanych formuł rachunku zdań ze zmiennymi zdaniowymi $p, q$ i stałymi logicznymi $\mathbf{0}$, $\mathbf{1}$ (alfabet: $\{p, q, \mathbf{0}, \mathbf{1}, \wedge, \vee, \Rightarrow, \neg\}$);
   (f) zbiór formuł rachunku zdań bez zmiennych, o wartości logicznej "prawda";
   (g) $\{a^i b^j c^k \,:\, i \neq j \vee j \neq k\}$;
   (h) $\{a^i b^j a^k \,:\, i + k = j\}$;
   (i) $\{a, b\}^* - \{ww \,:\, w \in \{a, b\}^*\}$.
2. Dla danych gramatyk bezkontekstowych $G, H$, skonstruować gramatyki generujące języki $L(G) \cup L(H)$, $L(G)L(H)$, $(L(G))^*$, $(L(G))^R$ (=lustrzane odbicie).

3. Dowieść, że następujące warunki są równoważne dla języka $L \subseteq \Sigma^*$:

   (a) $L$ jest regularny,

   (b) $L$ jest generowany przez gramatykę bezkontekstową, w której każda reguła jest postaci $X \to \epsilon$, $X \to Y$, lub $X \to \sigma Y$, $\sigma \in \Sigma$,

   (c) $L$ jest generowany przez gramatykę bezkontekstową, w której każda reguła jest postaci $X \to \epsilon$, $X \to Y$, lub $X \to Y\sigma$, $\sigma \in \Sigma$,

   (d) $L$ jest generowany przez gramatykę bezkontekstową, w której każda reguła jest postaci $X \to \alpha$ lub $X \to \beta Y$, $\alpha, \beta \in \Sigma^*$.

4. Podać przykład gramatyki bezkontekstowej w której każda reguła jest postaci $X \to \epsilon$, $X \to Y$, $X \to \sigma Y$ lub $X \to Y\sigma$, $\sigma \in \Sigma$, ale język generowany przez gramatykę nie jest regularny.

5. (*) Czy każdy język bezkontekstowy jest generowany przez gramatykę w postaci z poprzedniego zadania?

6. Dla dowolnej gramatyki $G$ oszacować z góry liczbę reguł gramatyki Chomsky'ego generującej język $L(G) - \{\epsilon\}$, jaką otrzymamy przez konstrukcje opisane w punkcie 2.2.

7. Niech $G$ będzie gramatyką bezkontekstową, z $m$ zmiennymi i niech, dla każdej reguły $Y \xrightarrow{G} w$, $|w| \leq \ell$. Dowieść, że jeśli $X_I \xrightarrow{G*} \epsilon$, to istnieje wyprowadzenie o długości $1 + \ell + \ell^2 + \ldots + \ell^{m-1}$. Czy to oszacowanie jest optymalne?

8. Dowieść, że dla każdej gramatyki $G$ istnieje stała $C$, taka, że dla dowolnego $w \neq \epsilon$, jeśli $X_I \xrightarrow{G*} w$, to istnieje wyprowadzenie o długości $\leq C \cdot |w|$.

9. Oszacować czas działania powyższego Algorytmu 6 (str. 9) oraz wynikającego zeń algorytmu rozstrzygającego, dla danej gramatyki $G$, czy $L(G) \neq \emptyset$.

10. Oszacować długość najkrótszego słowa generowanego przez gramatykę $G$

    (a) jeśli $G$ jest w postaci Chomsky'ego,

    (b) w dowolnym przypadku.

11. Zaprojektować algorytm, który, dla danej gramatyki $G$, odpowiada na pytanie, czy język $L(G)$ jest nieskończony.

12. Dowieść, że procedury sprowadzania dowolnej gramatyki bezkontekstowej do postaci Chomsky'ego opisane w punktach 2.1 i 2.2 można przeprowadzić w czasie wielomianowym względem oryginalnej gramatyki. Oszacować rozmiar otrzymanej gramatyki.

    *Uwaga*: Jeśli oryginalna gramatyka generuje słowo puste, możemy znaleźć gramatykę Chomsky'ego dla $L(G) - \{\epsilon\}$ i dodać regułę $X_I \to \epsilon$.

Języki, Automaty i Obliczenia. Odcinek 4

# Automaty ze stosem. Własności języków bezkontekstowych

Damian Niwiński

Listopad 2003

## 1 Automaty ze stosem

### 1.1 Podstawowe definicje

Automat ze stosem może być przedstawiony jako następujący układ:

$$A = (\Sigma, \Gamma, Q, q_I, Z_I, \delta, F)$$

gdzie

- $\Sigma$ jest *alfabetem symboli wejściowych*,
- $\Gamma$ jest *alfabetem symboli stosowych*,
- $Q$ jest zbiorem stanów,
- $q_I \in Q$ jest stanem początkowym,
- $Z_I \in \Gamma$ jest *symbolem startowym*,
- $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \times Q \times \Gamma^*$ jest relacją przejścia,
- $F \subseteq Q$ jest zbiorem stanów akceptujących.

Wszystkie występujące tu zbiory są skończone.

Przejście postaci $(q, a, Z, q', \gamma)$ będziemy także zapisywać

$$q, a, Z \rightarrow_A q', \gamma$$

Należy je rozumieć następująco.

Gdy $a \in \Sigma$ :

Jeżeli automat znajduje się w stanie $q$, głowica automatu czyta na taśmie symbol $a$, a na wierzchołku stosu znajduje się symbol $Z$, to w następnej chwili czasu automat może zmienić stan na $q'$, zastąpić symbol $Z$ na szczycie stosu przez słowo $\gamma$ i przesunąć głowicę o jedno miejsce w prawo.

Gdy $a = \epsilon$ :

Jeżeli automat znajduje się w stanie $q$, a na wierzchołku stosu jest symbol $Z$, to w następnej chwili automat może zmienić stan na $q'$ i zastąpić symbol $Z$ na stosie przez słowo $\gamma$. Pozycja głowicy pozostaje przy tym nie zmieniona.

Zauważmy, że jeśli w rozważanym powyżej przejściu $\gamma = Z'Z$, to operację na stosie można określić jako

*połóż $Z'$ na stos,*

a jeśli $\gamma = \epsilon$, to jako

*zdejmij symbol z wierzchołka stosu.*[1]

*Konfigurację* automatu w danej chwili czasu możemy przedstawić jako trójkę $(q, w, \gamma)$, gdzie $q \in Q$, $w \in \Sigma^*$, $\gamma \in \Gamma^*$. Należy przez to rozumieć sytuację, kiedy automat znajduje się w stanie $q$, $w$ jest słowem, jakie pozostało jeszcze do przeczytania, a $\gamma$ jest zawartością stosu, przy czym przyjmujemy, że pierwsza litera słowa $\gamma$ odpowiada symbolowi na wierzchołku stosu itd.

Za konfigurację *początkową* uważamy każdą konfigurację postaci

$$(q_I, w, Z_I)$$

[1] Można wykazać, że automaty ze stosem używające jedynie tych dwóch operacji mają nie mniejszą siłę obliczeniową niż zdefiniowane przez nas bardziej ogólne automaty.

natomiast konfiguracją *końcową* nazwiemy dowolną konfigurację postaci $(q, \epsilon, \gamma)$ (tzn. kiedy całe słowo zostało już przeczytane). Dalej, określamy relację na konfiguracjach

$$(q, w, \gamma) \vdash_A (q', w', \gamma')$$

odzwierciedlającą fakt, że konfiguracja $(q', w', \gamma')$ może być osiągnięta z konfiguracji $(q, w, \gamma)$ w jednym kroku przez wykonanie jakiegoś przejścia automatu $A$.

Mianowicie, relacja

$$(q, aw, Z\beta) \vdash_A (q', w, \alpha\beta)$$

zachodzi, o ile

$$q, a, Z \to_A q', \alpha$$

jest przejściem automatu $A$. Zauważmy, że trójka $(q, w, \epsilon)$ (pusty stos) jest dobrze określoną konfiguracją, z której jednak nie ma już przejścia do żadnej konfiguracji.

Jak zwykle, $\vdash_A^*$ oznacza zwrotno-przechodnie domknięcie relacji $\vdash_A$.

Niech teraz $w \in \Sigma^*$. Ciąg konfiguracji $(q_0, w_0, \gamma_0)$, $(q_1, w_1, \gamma_1)$, $\ldots$, $(q_m, w_m, \gamma_m)$, gdzie $(q_0, w_0, \gamma_0)$ jest konfiguracją początkową z $w_0 = w$ oraz, dla $i < m$, $(q_i, w_i, \gamma_i) \vdash_A (q_{i+1}, w_{i+1}, \gamma_{i+1})$, nazywamy *obliczeniem* automatu $A$ na słowie $w$. Obliczenie nazwiemy *akceptującym* jeśli $(q_m, w_m, \gamma_m)$ jest konfiguracją końcową (tj. $w_m = \epsilon$), a ponadto $q_m \in F$. Język rozpoznawany przez automat ze stosem $A$ określamy jako zbiór tych słów, dla których istnieje obliczenie akceptujące; tak więc

$$L(A) = \{w \in \Sigma^* : (q_I, w, Z_I) \vdash_A^* (q_f, \epsilon, \gamma) \text{ dla pewnych } q_f \in F, \gamma \in \Gamma^*\}$$

## 1.2 Automaty akceptujące przez opróżnienie stosu

Dla automatu ze stosem $A$, określimy również język $Z(A)$ obejmujący te i tylko te słowa $w$, dla których istnieje obliczenie zakończone całkowitym opróżnieniem stosu,

$$Z(A) = \{w \in \Sigma^* : (q_I, w, Z_I) \vdash_A^* (q, \epsilon, \epsilon) \text{ dla pewnego } q \in Q\}$$

W ten sposób przyporządkowujemy automatowi język na ogół różny od $L(A)$, przy czym stany akceptujące nie odgrywają tutaj żadnej roli. Okazuje się jednak, że nowy sposób akceptacji jest w pewnym sensie równoważny poprzedniemu.

**Fakt 1** *1. Dla każdego automatu ze stosem $A$, istnieje automat ze stosem $A'$, taki że $L(A) = Z(A')$.*

*2. Dla każdego automatu ze stosem $B$, istnieje automat ze stosem $B'$, taki że $Z(B) = L(B')$.*

*Dowód. Ad (1).* Niech $A = (\Sigma, \Gamma, Q, q_I, Z_I, \delta, F)$. Określamy automat

$$A' = (\Sigma, \Gamma \cup \{Z_I'\}, Q \cup \{q_I', q_e\}, q_I', Z_I', \delta', \emptyset)$$

gdzie $\delta' \supseteq \delta$ jest rozszerzeniem $\delta$ o następujące przejścia

$$\begin{array}{rcll} q_I', \epsilon, Z_I' & \to & q_I, Z_I Z_I' & \\ q_f, \epsilon, Z & \to & q_e, Z & \text{dla } q_f \in F,\ Z \in \Gamma \cup \{Z_I'\} \\ q_e, \epsilon, Z & \to & q_e, \epsilon & \text{dla } Z \in \Gamma \cup \{Z_I'\} \end{array}$$

A zatem, automat $A'$ rozpoczyna działanie od położenia "starego" symbolu startowego $Z_I$ na "nowy" symbol $Z'_I$. Następnie automat $A'$ symuluje działanie automatu $A$, z tym, że kiedy $A$ wchodzi w stan akceptujący, $A'$ może niedeterministycznie przyjąć opcję (całkowitego) opróżniania stosu. Zanim to nastąpi, nowy symbol startowy pozostaje cały czas na dnie stosu, żeby uniknąć przypadkowej akceptcji przez $A'$ (w sensie opróżnienia stosu) w sytuacji kiedy $A$ opróżnia stos, ale nie wchodzi w stan akceptujący.

Szczegóły dowodu, że $L(A) = Z(A')$ pozostawiamy Czytelnikowi.

*Ad (2).* Niech $B = (\Sigma, \Gamma, Q, q_I, Z_I, \delta, \emptyset)$. Określamy automat

$$B' = (\Sigma, \Gamma \cup \{Z'_I\}, Q \cup \{q'_I, q_f\}, q'_I, Z'_I, \delta', \{q_f\})$$

gdzie $\delta' \supseteq \delta$ jest rozszerzeniem $\delta$ o następujące przejścia

$$\begin{array}{rcll} q'_I, \epsilon, Z'_I & \rightarrow & q_I, Z_I Z'_I & \\ q, \epsilon, Z'_I & \rightarrow & q_f, \epsilon & \text{dla } q \in Q \end{array}$$

Intuicyjnie, automat $B'$ symuluje działanie automatu $B$, z tym że na spodzie stosu pozostaje cały czas nowy symbol startowy $Z'_I$, który poza tym nie jest używany. Jeśli w pewnej chwili obliczenia symbol ten znajdzie się na wierzchołku stosu, oznacza to, że automat $B$, w odnośnej chwili symulowanego obliczenia opróżniłby swój stos. Dlatego automat $B'$ wchodzi wówczas w stan akceptujący. Pamiętajmy jednak, że do tego, by obliczenie było akceptujące nie wystarczy samo pojawienie się stanu akceptującego, potrzeba jeszcze, by konfiguracja była końcowa (tzn. by całe słowo zostało przeczytane). □

# 2 Automaty ze stosem jako modele rozpoznawania języków bezkontekstowych

W tym rozdziale wykażemy ścisłą odpowiedniość pomiędzy automatami ze stosem a gramatykami bezkontekstowymi : automaty rozpoznają dokładnie te języki, które gramatyki generują. Zanim pokażemy właściwą konstrukcję, która automatowi przyporządkowuje równoważną gramatykę i na odwrót, wykażemy pewną nieco zaskakującą a bardzo przydatną własność automatów ze stosem.

## 2.1 Redukcja stanów w automacie ze stosem

Udowodnimy, że, w automacie ze stosem, symbole stosowe mogą w pewnym sensie całkowicie przejąć rolę stanów.

**Twierdzenie 1** *Dla każdego automatu ze stosem $A$ istnieje automat $A'$* z jednym stanem, *taki że $Z(A) = Z(A')$.*

*Dowód.* Niech $A = (\Sigma, \Gamma, Q, q_I, Z_I, \delta, F)$. Określimy automat

$$A' = (\Sigma, \Gamma', \{e\}, e, Z'_I, \delta', \emptyset)$$

Nowym alfabetem symboli stosowych jest

$$\Gamma' = \{[q, Z, q'] \,:\, q, q' \in Q,\ Z \in \Gamma\} \cup \{Z'_I\}$$

Zbiór przejść $\delta'$ określony jest następująco:

- dla dowolnego $q \in Q$,
$$e, \epsilon, Z'_I \ \rightarrow_{A'}\ e, [q_I, Z_I, q]$$
jest przejściem automatu $A'$;
- o ile $q, a, Z \rightarrow_A p, Z_1 Z_2 \dots Z_m$, $m \geq 1$, jest przejściem automatu $A$, a $p_1, p_2, \dots, p_{m-1}, q' \in Q$ są dowolne, to
$$e, a, [q, Z, q'] \ \rightarrow_{A'}\ e, [p, Z_1, p_1][p_1, Z_2, p_2] \dots [p_{m-2}, Z_{m-1}, p_{m-1}][p_{m-1}, Z_m, q']$$
jest przejściem automatu $A'$;
- o ile $q, a, Z \rightarrow_A q', \epsilon$ jest przejściem automatu $A$, to
$$e, a, [q, Z, q'] \ \rightarrow_{A'}\ e, \epsilon$$
jest przejściem automatu $A'$.

Wykażemy, że dla dowolnego $w = uv$, następujące warunki są równoważne:

**Warunek 1**

$$q_I, w, Z_I \vdash^*_A q, v, Z_1 \dots Z_m$$

**Warunek 2** Dla dowolnych $p_1, \dots, p_m \in Q$,

$$e, w, Z'_I \vdash^*_{A'} e, v, [q, Z_1, p_1][p_1, Z_2, p_2] \dots [p_{m-2}, Z_{m-1}, p_{m-1}][p_{m-1}, Z_m, p_m]$$

Zauważmy od razu, że dla $v = \epsilon$ i $m = 0$, z równoważności Warunków 1 i 2 otrzymamy, że

$$q_I, w, Z_I \vdash^*_A q, \epsilon, \epsilon \iff e, w, Z'_I \vdash^*_{A'} e, \epsilon, \epsilon$$

a zatem

$$w \in Z(A) \iff w \in Z(A')$$

co zakończy dowód.

Pozostaje więc dowieść (1)$\Leftrightarrow$ (2)

(1)$\Rightarrow$ (2) Przez indukcję ze względu na długość obliczenia automatu $A$ prowadzącego od konfiguracji $(q_I, w, Z_I)$ do $(q, v, Z_1 \dots Z_m)$, wykażemy że

$$e, w, Z'_I \vdash^*_{A'} e, v, [q, Z_1, p_1][p_1, Z_2, p_2] \dots [p_{m-2}, Z_{m-1}, p_{m-1}][p_{m-1}, Z_m, p_m]$$

przy dowolnych $p_1, \dots, p_m$.

Dla obliczenia o długości 0

$$q_I, w, Z_I \vdash_A^* q_I, w, Z_I$$

mamy, dla dowolnego $p \in Q$, obliczenie automatu $A'$ o długości 1, postaci

$$e, w, Z_I' \vdash_{A'} e, w, [q_I, Z_I, p]$$

Z kolei rozważmy obliczenie automatu $A$ o długości $> 0$. Mamy zatem

$$\begin{aligned} q_I, w, Z_I &\vdash_A^* q', av, Y_1 Y_2 \dots Y_m \\ &\vdash_A q, v, Z_1 \dots Z_k Y_2 \dots Y_m \end{aligned}$$

dla pewnych $a \in \Sigma \cup \{\epsilon\}$ oraz $Y_1, \dots, Y_m$, $Z_1, \dots, Z_k$, gdzie

$$q', a, Y_1 \rightarrow_A q, Z_1 \dots Z_k$$

jest przejściem automatu $A$.

Niech $q_1, \dots, q_k, p_2, \dots, p_m \in Q$, dowolne. Z założenia indukcyjnego, dla dowolnego $p_1$, mamy

$$(*)\ e, w, Z_I' \vdash_{A'}^* e, av, [q', Y_1, p_1][p_1, Y_2, p_2] \dots [p_{m-2}, Y_{m-1}, p_{m-1}][p_{m-1}, Y_m, p_m]$$

Rozważymy teraz dwa przypadki, w zależności od tego czy $k \neq 0$.

1. Gdy $k > 0$, mamy, z definicji $A'$, przejście

$$e, a, [q', Y_1, q_k] \rightarrow_{A'} e, [q, Z_1, q_1][q_1, Z_2, q_2] \dots [q_{k-1}, Z_k, q_k]$$

Następnie, kładąc w powyższej zależności $(*)$, $p_1 = q_k$, mamy

$$e, w, Z_I' \vdash_{A'}^* e, av, [q', Y_1, q_k][q_k, Y_2, p_2] \dots [p_{m-2}, Y_{m-1}, p_{m-1}][p_{m-1}, Y_m, p_m]$$

Stąd,

$$\begin{aligned} e, w, Z_I' &\vdash_{A'}^* e, av, [q', Y_1, q_k][q_k, Y_2, p_2] \dots [p_{m-2}, Y_{m-1}, p_{m-1}][p_{m-1}, Y_m, p_m] \\ &\vdash_{A'} e, v, [q, Z_1, q_1][q_1, Z_2, q_2] \dots [q_{k-1}, Z_k, q_k][q_k, Y_2, p_2] \dots [p_{m-2}, Y_{m-1}, p_{m-1}][p_{m-1}, Y_m, p_m] \end{aligned}$$

2. Gdy $k = 0$, mamy, z definicji $A'$,

$$e, a, [q', Y_1, q] \rightarrow_{A'} e, \epsilon$$

Kładąc w zależności $(*)$, $p_1 = q$, mamy

$$e, w, Z_I' \vdash_{A'}^* e, av, [q', Y_1, q][q, Y_2, p_2] \dots [p_{m-2}, Y_{m-1}, p_{m-1}][p_{m-1}, Y_m, p_m]$$

Stąd

$$\begin{aligned} e, w, Z_I' &\vdash_{A'}^* e, av, [q', Y_1, q][q, Y_2, p_2] \dots [p_{m-2}, Y_{m-1}, p_{m-1}][p_{m-1}, Y_m, p_m] \\ &\vdash_{A'} e, v, [q, Y_2, p_2] \dots [p_{m-2}, Y_{m-1}, p_{m-1}][p_{m-1}, Y_m, p_m] \end{aligned}$$

co kończy dowód implikacji (1)⇒ (2).

(2)⇒ (1) Wykażemy implikację silniejszą, a mianowicie, że z faktu iż

$$e, w, Z'_I \vdash^*_{A'} e, v, [q, Z_1, p_1][p_1, Z_2, p_2] \dots [p_{m-2}, Z_{m-1}, p_{m-1}][p_{m-1}, Z_m, p_m]$$

dla *pewnych* $p_1, \dots, p_m$, wynika

$$q_I, w, Z_I \vdash^*_A q, v, Z_1 \dots Z_m$$

Podobnie jak poprzednio użyjemy indukcji ze względu na długość obliczenia, ale tym razem chodzić będzie o obliczenie automatu $A'$ prowadzące od konfiguracji $e, w, Z'_I$ do $e, v, [q, Z_1, p_1][p_1, Z_2, p_2] \dots [p_{m-2}, Z_{m-1}, p_{m-1}][p_{m-1}, Z_m, p_m]$. Zgodnie z definicją automatu $A'$, najkrótsze takie obliczenie ma oczywiście długość 1 i jest postaci

$$e, w, Z'_I \ \vdash_{A'} \ e, w, [q_I, Z_I, q]$$

Odpowiednie obliczenie automatu $A$ jest obliczeniem trywialnym

$$q_I, w, Z_I \vdash^*_A q_I, w, Z_I$$

Z kolei rozważmy możliwe obliczenie postaci

$$\begin{array}{lll} e, w, Z'_I & \vdash^*_{A'} & e, av, [q', Y_1, p_1][p_1, Y_2, p_2] \dots [p_{m-2}, Y_{m-1}, p_{m-1}][p_{m-1}, Y_m, p_m] \\ & \vdash_{A'} & e, v, [q, Z_1, q_1][q_1, Z_2, q_2] \dots [q_{k-1}, Z_k, p_1][p_1, Y_2, p_2] \dots [p_{m-2}, Y_{m-1}, p_{m-1}][p_{m-1}, Y_m, p_m] \end{array}$$

(Pisząc w ten sposób dopuszczamy $k = 0$, czyli symbole $Z_i$ są nieobecne i $p_1 = q$.) Ostatni krok tego obliczenia wykorzystuje przejście automatu $A'$,

$$e, a, [q', Y_1, p_1] \ \rightarrow_{A'} \ e, [q, Z_1, q_1][q_1, Z_2, q_2] \dots [q_{k-1}, Z_k, p_1]$$

gdzie

$$q', a, Y_1 \ \rightarrow_A \ q, Z_1 \dots Z_k$$

jest przejściem automatu $A$. (Poniższy argument działa zarówno dla $k > 0$ jak i dla $k = 0$, gdy ciąg $Z_1 \dots Z_k$ jest pusty.)

Z założenia indukcyjnego, mamy

$$q_I, w, Z_I \vdash^*_A q', av, Y_1 \dots Y_m$$

Stąd

$$\begin{array}{lll} q_I, w, Z_I & \vdash^*_A & q', av, Y_1 \dots Y_m \\ & \vdash_A & q, v, Z_1 \dots Z_k Y_2 \dots Y_m \end{array}$$

Ta uwaga kończy dowód twierdzenia. □

## 2.2 Równoważność gramatyk bezkontekstowych i automatów ze stosem

**Twierdzenie 2** *Dla każdego automatu ze stosem $A$, można skonstruować gramatykę bezkontekstową $G$, taką że $L(A) = L(G)$.*

*Dowód.* Niech $L = L(A)$. Na podstawie Faktu 1 i Twierdzenia 2.1, możemy znaleźć automat z jednym stanem, $A'$, taki że $L = Z(A')$, powiedzmy

$$A' = (\Sigma, \Gamma, \{e\}, e, Z_I, \delta, \emptyset)$$

Określamy gramatykę $G$ w ten sposób, że

$$G = (\Sigma, \Gamma, Z_I, R)$$

gdzie zbiór reguł $R$ jest zdefiniowany następująco:

$$Z \xrightarrow{G} aY_1 \dots Y_m \Leftrightarrow e, a, Z \rightarrow_{A'} e, Y_1 \dots Y_m$$

gdzie $Z, Y_1, \dots, Y_m \in \Gamma$, $a \in \Sigma \cup \{\epsilon\}$.

Pozostaje dowieść

$$Z(A') = L(G)$$

"$\subseteq$" Udowodnimy następujący fakt.

(*) Przypuśćmy, że $w = uv$ i $e, w, Z_I \vdash^*_{A'} e, v, \gamma$. Wówczas $Z_I \xrightarrow{G*} u\gamma$.

Dowód przeprowadzimy przez indukcję ze względu długość obliczenia częściowego prowadzącego od konfiguracji $(e, w, Z_I)$ do $(e, v, \gamma)$. Jeśli to jest obliczenie długości 0, to teza jest oczywista. Jeśli obliczenie jest dłuższe niż 0, to mamy

$$\begin{aligned} e, w, Z_I &\vdash^*_{A'} e, av, Y_1 \dots Y_m \\ &\vdash_A e, v, Z_1 \dots Z_k Y_2 \dots Y_m \end{aligned}$$

gdzie $a \in \Sigma \cup \{\epsilon\}$, $u = u'a$, dla pewnego $u'$, $w = u'av$ i $e, a, Y_1 \rightarrow_{A'} e, Z_1 \dots Z_k$ jest przejściem automatu $A'$. Wówczas $Y_1 \xrightarrow{G} aZ_1 \dots Z_k$ jest regułą gramatyki $G$. Korzystając z założenia indukcyjnego mamy więc

$$Z_I \xrightarrow{G*} u'Y_1 \dots Y_m \xrightarrow{G} u'aZ_1 \dots Z_k Y_2 \dots Y_m$$

co kończy dowód (*). Biorąc pod uwagę przypadek gdy $v = \gamma = \epsilon$, otrzymujemy, że $e, w, Z_I \vdash^*_{A'} e, \epsilon, \epsilon$ implikuje $Z_I \xrightarrow{G*} w$.

"$\supseteq$" Udowodnimy następujący fakt.

(**) Przypuśćmy, że $w = uv \in \Sigma^*$ i $Z_I \xrightarrow{l-G*} u\gamma$, $\gamma \in \Gamma^*$. Wówczas $e, w, Z_I \vdash^*_{A'} e, v, \gamma$.

Dowód przeprowadzimy przez indukcję ze względu na długość lewostronnego wyprowadzenia słowa $u\gamma$ ze zmiennej $Z_I$. Kiedy wyprowadzenie ma długość 0, to $u = \epsilon$, $\gamma = Z_I$ i teza jest oczywista. Kiedy wyprowadzenie jest dłuższa niż 0, to mamy

$$\begin{aligned} Z_I &\overset{l-G*}{\rightarrow} u'Y_1 \dots Y_m \\ &\overset{l-G}{\rightarrow} u'aZ_1 \dots Z_k Y_2 \dots Y_m \end{aligned}$$

gdzie, jak poprzednio, $a \in \Sigma \cup \{\epsilon\}$, $u = u'a$, $w = u'av$ i $Y_1 \overset{G}{\rightarrow} aZ_1 \dots Z_k$ jest regułą gramatyki $G$, a zatem, zgodnie z definicją $G$, $e, a, Y_1 \rightarrow_{A'} e, Z_1 \dots Z_k$ jest przejściem automatu $A'$. Korzystając z założenia indukcyjnego mamy

$$\begin{aligned} e, w, Z_I &\vdash^*_{A'} e, av, Y_1 \dots Y_m \\ &\vdash_{A'} e, v, Z_1 \dots Z_k Y_2 \dots Y_m \end{aligned}$$

Przypuśćmy teraz, że $Z_I \overset{G*}{\rightarrow} w$. Na podstawie Twierdzenia 1 z Odcinka 3, istnieje wyprowadzenie *lewostronne* $w$ z $Z_I$. Z (**) wynika, że wówczas $e, w, Z_I \vdash^*_{A'} e, \epsilon, \epsilon$, czyli $w \in Z(A')$.

Ta uwaga kończy dowód twierdzenia. □

**Twierdzenie 3** *Dla każdej gramatyki bezkontekstowej $G$, można skonstruować automat ze stosem $A$, taki że $L(G) = L(A)$.*

*Dowód.* Niech $G$ będzie gramatyką bezkontekstową. Jeśli $L(G) = \emptyset$, to zadanie jest trywialne, możemy więc założyć, że $L(G) \neq \emptyset$. Na podstawie Faktu 3 i Twierdzenia 2 z Odcinka 3, możemy skonstruować gramatykę w postaci normalnej Chomsky'ego, generującą język $L(G) - \{\epsilon\}$. Zbudujemy automat $A'$, taki że $Z(A') = L(G')$. Na podstawie Faktu 1, można dalej znaleźć automat, powiedzmy $A''$, taki że $L(A'') = L(G')$. Modyfikacja tego automatu w taki sposób by akceptował również słowo puste, o ile należy ono do $L(G)$, jest łatwą konstrukcją, którą pozostawiamy Czytelnikowi.

Niech

$$G' = (\Sigma, V, X_I, R)$$

Określamy automat

$$A' = (\Sigma, V, \{e\}, e, X_I, \delta, \emptyset)$$

gdzie relacja przejść $\delta$ jest określona następująco:

- dla każdej reguły postaci $X \overset{G}{\rightarrow} YZ$,

$$e, \epsilon, X \rightarrow_{A'} e, YZ$$

- dla każdej reguły postaci $X \overset{G}{\rightarrow} \sigma$, $\sigma \in \Sigma$,

$$e, \sigma, X \rightarrow_{A'} e, \epsilon$$

Pozostaje dowieść

$$Z(A') = L(G')$$

Dowód tej równości opuszczamy, ponieważ jest analogiczny do dowodu podobnej równości w dowodzie Twierdzenia 2. Można powiedzieć więcej: jeżeli do określonego przed chwilą automatu zastosujemy konstrukcję z tamtego dowodu, to otrzymana gramatyka okaże się dokładnie (!) gramatyką $G'$. A zatem powyższa równość została już *de facto* udowodniona w dowodzie Twierdzenia 2. □

**Wniosek 1** *Następujące warunki są równoważne, dla języka $L \subseteq \Sigma^*$.*

1. *$L$ jest generowany przez gramatykę bezkontekstową,*
2. *$L$ jest akceptowany przez niedeterministyczny automat ze stosem.*

Języki generowane przez gramatyki bezkontekstowe nazywamy *językami bezkontekstowymi* (ang. *context-free languages*). Gramatyki bezkontekstowe są metodą *syntezy* języków bezkontekstowych. Poznaliśmy już także narzędzie *analizy* tych języków: niedeterministyczny automat ze stosem. Udowodniliśmy, że obie metody: syntezy i analizy, dokładnie sobie odpowiadają, tzn. automaty ze stosem rozpoznają dokładnie te języki, które gramatyki syntezują.

Klasa języków bezkontekstowych stanowi rozszerzenie klasy języków regularnych i obejmuje wiele naturalnych przykładów języków nie będących regularnymi. Wiele elementów języków programowania, jak np. wyrażenia arytmetyczne, struktura bloków itp., da się opisać językami bezkontekstowymi. Stąd, ogromne znaczenie tych języków w konstrukcji kompilatorów.

# Ćwiczenia

1. Skonstruować automaty ze stosem rozpoznające poznane wcześniej języki bezkontekstowe: zbiór palindromów, zbiór poprawnie uformowanych ciągów nawiasów, zbiór słów, które mają dwa razy więcej $b$ niż $a$, zbiór ciągów, które nie są postaci $ww$.

2. Dla liczby naturalnej $n$, niech $\mathrm{bin}(n) \in \{0,1\}^*$ będzie binarnym przedstawieniem liczby $n$. Skonstruować automat ze stosem rozpoznający język

$$\{\mathrm{bin}(n)\$\mathrm{bin}(n+1)^R \,:\, n \in N\}$$

3. Dowieść, że dla każdego automatu ze stosem $A$, można skonstruować automat ze stosem *o dwóch stanach* $A'$, taki że $L(A) = L(A')$.

4. Dowieść, że automatowi $A'$ z poprzedniego zadania można postawić dalszy wymóg, że każde przejście jest postaci

$$q, a, Z \rightarrow_{A'} q', \alpha$$

gdzie $|\alpha| \leq 2$ ($q, q'$ dowolne).

5. Dowieść, że dla każdego automatu ze stosem $A$, można skonstruować równoważny mu automat ze stosem $A''$, w którym każde przejście jest postaci

$$q, a, Z \ \rightarrow_{A''} \ q', YZ$$

lub

$$q, a, Z \ \rightarrow_{A''} \ q', \epsilon$$

(*) Czy dla tego automatu można nadal ograniczyć liczbę stanów?

6. Mając dany automat ze stosem akceptujący język $L$, skonstruować automaty ze stosem akceptujące następujące języki:

- $\text{Prefix}(L) = \{w \ : \ (\exists v) wv \in L\}$
- $\text{Suffix}(L) = \{w \ : \ (\exists u) uw \in L\}$
- $\text{Subword}(L) = \{w \ : \ (\exists u, v) uwv \in L\}$
- (*) $L^R = \{w^R \ : \ w \in L\}$
- (**) $\text{Cycle}(L) = \{vw \ : \ wv \in L\}$

7. Mając dany automat ze stosem akceptujący język $L$ i automat skończony akceptujący język $R$, skonstruować automaty ze stosem akceptujące następujące języki:

- $LR^{-1}$
- $R^{-1}L$
- $L \cap R$

8. Dla danych językow regularnych $L$ i $M$, skonstruować automat ze stosem rozpoznający język $\bigcup_{i \in N}(L^i \cap M^i)$. Podać przykład, że zbiór ten nie musi być regularny.

9. (*) Dowieść, że dla każdego automatu ze stosem $A$ istnieje stała $C$ (zależna od automatu), taka, że dla każdego słowa $w \in Z(A)$, istnieje obliczenie akceptujące (przez pusty stos) o długości $\leq C|w|$. Wskazówka: oszacować wysokość stosu w obliczeniu akceptującym.

10. (*) Niech $A$ będzie automatem ze stosem. Dowieść, że zbiór słów, które są możliwymi zawartościami stosu automatu $A$, jest językiem regularnym. Formalnie, mamy na myśli zbiór

$$\{\alpha \in \Gamma^* \ : \ (\exists w, v \in \Sigma^*)(\exists q \in Q) \, q_I, w, Z_I \ \vdash_A \ q, v, \alpha\}$$

Wywnioskować stąd, że zbiór słów, które są możliwymi zawartościami stosu automatu $A$ w jakimś obliczeniu *akceptującym*, jest językiem bezkontekstowym.

11. Automat ze stosem $A = (\Sigma, \Gamma, Q, q_I, Z_I, \delta, F)$ nazywamy *deterministycznym*, jeśli w każdej sytuacji możliwy jest co najwyżej jeden ruch. Dokładniej:

- jeśli, dla pewnej pary $q, Z$, zachodzi $q, \epsilon, Z \rightarrow_A p, \alpha$ przy pewnych $p, \alpha$, to dla żadnego $\sigma \in \Sigma$, nie zachodzi $q, \sigma, Z \rightarrow_A p', \alpha'$, dla żadnych $p', \alpha'$;

- dla każdych $q, \sigma, Z$, istnieje co najwyżej jedna para $p, \alpha$, taka że $q, \sigma, Z \rightarrow_A p, \alpha$.

Niech $L$ będzie zbiorem palindromów nad alfabetem $\{a, b\}$. Dowieść, że jeśli $L = Z(A)$, to automat $A$ jest niedeterministyczny. (Uwaga: analogiczny fakt można dowieść, gdy $L = L(A)$, ale jest to znacznie trudniejsze.)

12. (**) Niech
$$M = \{a^n b^n \,:\, n \in N\} \cup \{a^n b^{2n} \,:\, n \in N\}$$
Nietrudno jest podać niedeterministyczny automat ze stosem rozpoznający ten język. Wykazać, że *nie* istnieje *deterministyczny*[2] automat ze stosem $A$, taki, że $M = L(A)$.

# 3 Lematy o pompowaniu

W tym rozdziale poznamy dwa *lematy o pompowaniu* i ich zastosowanie do dowodzenia, że jakiś język nie jest bezkontekstowy.

**Lemat 1 (o pompowaniu)** *Niech $L \subseteq \Sigma^*$ będzie językiem bezkontekstowym. Wówczas istnieje stała $M$ zależna tylko od $L$, taka, że dla każdego słowa $\alpha$, jeśli $\alpha \in L$ i $|\alpha| \geq M$, to $\alpha$ można przedstawić $\alpha = \alpha_1\gamma_1\beta\gamma_2\alpha_2$ tak, że spełnione są następujące warunki:*

- $\gamma_1\gamma_2 \neq \epsilon$,
- $|\gamma_1\beta\gamma_2| \leq M$,
- *dla każdego $i \geq 0$, $\alpha_1\gamma_1^i\beta\gamma_2^i\alpha_2 \in L$.*

*Dowód.* Niech $G$ będzie gramatyką bezkontekstową w postaci Chomsky'ego generującą język $L - \{\epsilon\}$, $G = (\Sigma, V, X_I, R)$. Niech $|V| = k$, kładziemy
$$M = 2^k$$
Niech $D$ będzie drzewem wyprowadzenia słowa $\alpha$ o długości $|\alpha| \geq M$. Wówczas głębokość $D$ jest $\geq k + 1$. Istotnie, nietrudno wykazać przez indukcję, że maksymalne słowo, jakie można wygenerować przy pomocy drzewa o głębokości $i$, ma długość $2^{i-1}$. Niech $p_0, p_1, p_2, \ldots, p_\ell$ będzie pewną ścieżką o maksymalnej długości w drzewie $D$. Mamy oczywiście $\ell \geq k + 1$, przy czym $D(p_\ell) \in \Sigma$, natomiast, dla $m < \ell$, $D(m) \in V$. Muszą zatem istnieć dwa wierzchołki na tej ścieżce, w których pojawia się ta sama zmienna, powiedzmy $D(p_i) = D(p_j) = X$, $i < j$. Wybierzmy "pierwszą od końca" parę o tej własności, tzn. załóżmy, że $(\forall m, n > i) m \neq n \Rightarrow D(p_m) \neq D(p_n)$. Oczywiście, podciąg $p_i, p_{i+1}, \ldots, p_{\ell-1}$ ma wówczas nie więcej niż $k+1$ elementów. Rozważmy *poddrzewo* drzewa $D$ wyznaczone przez wierzchołek $p_i$, które nazwiemy $D'$, dokładniej (patrz odc. 3, str. 3)
$$\begin{aligned} \mathrm{dom}(D') &= \{u \,:\, p_i u \in \mathrm{dom}(D)\} \\ D'(u) &= D(p_i u)\,, \text{ dla } u \in \mathrm{dom}(D') \end{aligned}$$

[2]Więcej wiadomości o automatach deterministycznych będzie można znaleźć w przygotowywanym następnym odcinku.

Nietrudno sprawdzić, że $D'$ jest również drzewem wyprowadzenia, lecz tym razem ze zmiennej $X = D(p_i)$. Zauważmy, że głębokość tego drzewa jest $\leq k+1$, co wynika z maksymalności ścieżki $p_1, p_2, \ldots, p_\ell$ i uczynionej wyżej uwagi o mocy podciągu $p_i, p_{i+1}, \ldots, p_{\ell-1}$. Niech $\gamma$ będzie plonem drzewa $D'$, $\gamma = y(D')$. Na mocy zauważonej wcześniej zależności pomiędzy głębokością drzewa wyprowadzenia a długością generowanego słowa,

$$|\gamma| \leq 2^k = M$$

Z kolei niech $D''$ będzie poddrzewem $D$ wyznaczonym przez wierzchołek $p_j$, analogicznie do $D'$. Zauważmy, że jest to również drzewo wyprowadzenia ze zmiennej $X$. Niech $\beta = y(D'')$.

Mamy więc

$$\gamma = \gamma_1 \beta \gamma_2$$

dla pewnych $\gamma_1, \gamma_2$.

Twierdzimy, że

$$\gamma_1 \gamma_2 \neq \epsilon$$

Istotnie, zgodnie z definicją gramatyki Chomsky'ego, wierzchołek $p_i$ ma dwa następniki w drzewie $D$, $p_i1$ i $p_i2$. Jednym z nich jest $p_{i+1}$, a drugi nie leży na wybranej przez nas ścieżce. Przypuśćmy, że $p_{i+1} = p_i2$. Niech $D_1$ i $D_2$ będą poddrzewami drzewa $D$ wyznaczonymi przez wierzchołki $p_i1$ i $p_i2$ odpowiednio. Nietrudno zauważyć, że $y(D_1)$

jest prefiksem słowa $\gamma_1$. Ponieważ, z definicji postaci gramatyki Chomsky'ego, $y(D_1) \neq \epsilon$, otrzymujemy $\gamma_1 \neq \epsilon$. W przypadku, gdy wierzchołkiem leżącym na ścieżce jest $p_i 1$, argument przebiega podobnie.

Dla zakończenia dowodu pozostaje jeszcze wykazać, że spełniony jest ostatni warunek lematu o pompowaniu. Niech $v \in \{1,2\}^*$ będzie tym słowem, że $p_j = p_i v$. (Intuicyjnie, $v$ wyznacza ścieżkę z wierzchołka $p_i$ do wierzchołka $p_j$.) Określimy ciąg drzew wyprowadzenia $E_0, E_1, \ldots$ i pomocniczo ciąg słów $u_2, u_3, \ldots \in \{1,2\}^*$, następująco:

- $E_0$ jest rezultatem podstawienia w drzewie $D$ drzewa $D''$ na wierzchołek $p_i$, w notacji symbolicznej (por. Odc. 3, str. 4), $E_0 = D[p_i \leftarrow D'']$.
- $E_1 = D$.
- $E_2$ jest rezultatem podstawienia w

  drzewie $D$ drzewa $D'$ na wierzchołek $p_j$, tj. $E_2 = D[p_j \leftarrow D']$, i $u_2 = p_j v$.
- $E_3$ jest rezultatem podstawienia w drzewie $E_2$ drzewa $D'$ na wierzchołek $u_2$, tj. $E_3 = E_2[u_2 \leftarrow D']$, i $u_3 = u_2 v$.
- Ogólniej, dla $n \geq 2$, $E_{n+1}$ jest rezultatem podstawienia w drzewie $E_n$ drzewa $D'$ na wierzchołek $u_n$, tj. $E_{n+1} = E_n[u_n \leftarrow D']$, i $u_{n+1} = u_n v$.

Niech $\alpha_1, \alpha_2$ będą tymi słowami, że $\gamma = \alpha_1 \beta \alpha_2$. Opierając się wprost na powyższych definicjach, nietrudno wykazać, że

$$\begin{array}{rcc} y(E_0) & = & \alpha_1\beta\alpha_2 \\ y(E_1) & = & \alpha_1\gamma_1\beta\gamma_2\alpha_2 \\ y(E_2) & = & \alpha_1\gamma_1\gamma_1\beta\gamma_2\gamma_2\alpha_2 \\ y(E_3) & = & \alpha_1\gamma_1\gamma_1\gamma_1\beta\gamma_2\gamma_2\gamma_2\alpha_2 \\ \ldots & \ldots & \ldots \\ y(E_n) & = & \alpha_1\gamma_1^n\beta\gamma_2^n\alpha_2 \end{array}$$

□

**Przykład 1** Zbiór $\{a^n b^n a^n : n \in N\}$ nie jest bezkontekstowy. Istotnie, niech $M$ będzie stałą z lematu i niech $n > M$. Przypuśćmy, że $a^n b^n a^n = \alpha_1\gamma_1\beta\gamma_2\alpha_2$ jest dekompozycją z lematu. Nietrudno zauważyć, że $\gamma_1, \gamma_2 \in \{a\}^* \cup \{b\}^*$. Analizując poszczególne przypadki, łatwo dochodzimy do sprzeczności. Na przykład, gdy $\gamma_1 \in \{a\}^*$ i $\gamma_2 \in \{b\}^*$, to również słowo $a^{n+|\gamma_1|}b^{n+|\gamma_2|}a^n$ powinno należeć do języka, wbrew definicji.

Jeśli $w \in \Sigma^*$ i $|w| = m$, to elementy zbioru $\{1, \ldots, m\}$ będziemy nazywali *pozycjami* w słowie $w$. W kolejnym lemacie będziemy rozważali słowa z wyróżnionymi pozycjami, tj., formalnie, pary $(w, P)$, gdzie $P \subseteq \{1, \ldots, |w|\}$.

**Lemat 2 (lemat Ogdena)** *Niech $L \subseteq \Sigma^*$ będzie językiem bezkontekstowym. Wówczas istnieje stała $M$ zależna tylko od $L$, taka, że dla każdego słowa $\alpha \in L$ z dowolnie wyróżnionymi co najmniej $M$ pozycjami, $\alpha$ można przedstawić $\alpha = \alpha_1\gamma_1\beta\gamma_2\alpha_2$ tak, że spełnione są następujące warunki:*

- *$\gamma_1$ i $\gamma_2$ zawierają wspólnie co najmniej jedną wyróżnioną pozycję,*
- *słowo $\gamma_1\beta\gamma_2$ zawiera nie więcej niż $M$ wyróżnionych pozycji,*
- *dla każdego $i \geq 0$, $\alpha_1\gamma_1^i\beta\gamma_2^i\alpha_2 \in L$.*

*Dowód* (szkic). Argument jest podobny jak w dowodzie lematu o pompowaniu. Niech $G = (\Sigma, V, X_I, R)$ będzie gramatyką bezkontekstową w postaci Chomsky'ego generującą język $L - \{\epsilon\}$ i niech $|V| = k$. Kładziemy tym razem

$$M = 2^{k+1}$$

Przypuśćmy, że $D$ jest drzewem wyprowadzenia słowa $\alpha$, w którym wyróżnione jest $\geq M$ pozycji (oczywiście wówczas także $|\alpha| \geq M$). Strategia wyboru ścieżki $p_0, p_1, \ldots, p_\ell$ jest następująca. Oczywiście, $p_0 = \epsilon$. Z kolei przypuśćmy, że już wybraliśmy do naszej ścieżki wierzchołki $p_0, p_1, \ldots, p_m$, $0 \leq m < \ell$. Wierzchołek $p_m$ ma w drzewie $D$ dwa następniki: $p_m1$ i $p_m2$, które wyznaczają dwa poddrzewa drzewa $D$, powiedzmy, $D_1$ i $D_2$ odpowiednio. Wybieramy ten następnik, w którego poddrzewie znajduje się *więcej* wyróżnionych pozycji, a w przypadku remisu wybieramy, powiedzmy, $p_m1$. Nietrudno zauważyć, że tak określona ścieżka $p_0, p_1, \ldots, p_\ell$ prowadzi do liścia odpowiadającego jakiejś wyróżnionej pozycji w generowanym słowie $\alpha$.

Każdy wierzchołek $p_m$ na ścieżce o tej własności, że *oba* poddrzewa wyznaczone przez następniki $p_m$ zawierają jakieś wyróżnione pozycje, nazwiemy *punktem rozgałęzienia*. Zauważmy, że jeżeli $p_m$ jest punktem rozgałęzienia, to poddrzewo wyznaczone przez $p_{m+1}$ zawiera istotnie *mniej* ale *co najmniej połowę* pozycji, jakie znajdowały się w drzewie wyznaczonym przez $p_m$. Pamiętając o tym, że w drzewie wyznaczonym przez $p_0$ (tj. drzewie $D$) jest $\geq 2^{k+1}$ wyróżnionych pozycji, nietrudno wykazać, że na ścieżce $p_0, p_1, \ldots, p_\ell$ musi leżeć co najmniej $k+1$ punktów rozgałęzienia. Wśród tych punktów możemy więc znaleźć dwa takie, powiedzmy $p_i, p_j$, że $i < j$ i $D(p_i) = D(p_j)$. Dalej dowód przebiega bardzo podobnie jak w przypadku lematu o pompowaniu. Mianowicie, wybieramy "pierwszą od końca" parę $p_i, p_j$ o powyższej własności i słowa $\alpha_1, \gamma_1, \beta, \gamma_2, \alpha_2$ określamy analogicznie, jak w tamtym dowodzie.

Aby wykazać, że słowo $\alpha_1\beta\alpha_2$ zawiera nie więcej niż $M$ wyróżnionych pozycji, wystarczy zauważyć, że $p_i$ jest co najwyżej "$k+1$-szym od końca" punktem rozgałęzienia na ścieżce, oraz, że, ogólnie, poddrzewo wyznaczone przez "$m$-ty od końca" punkt rozgałęzienia na naszej ścieżce, zawiera co najwyżej $2^m$ pozycji. Fakt, że jedno ze słów $\gamma_1$, $\gamma_2$ zawiera co najmniej jedną wyróżnioną pozycję, wynika z tego, że $p_i$ jest punktem rozgałęzienia. Ostatni warunek lematu jest otrzymany analogicznie, jak w przypadku dowodu lematu o pompowaniu. □

**Przykład 2** Język

$$L = \{a^i b^j c^k \,:\, i \neq j,\, j \neq k,\, i \neq k\}$$

nie jest bezkontekstowy. Istotnie, niech $n$ będzie stałą z lematu Ogdena. Rozważmy słowo $\alpha = a^n b^{n+n!} c^{n+2\cdot n!}$, w którym są wyróżnione wszystkie pozycje $a$ i tylko te. Przypuśćmy, że $\alpha = \alpha_1\gamma_1\beta\gamma_2\alpha_2$ jest dekompozycją z lematu. Wówczas przynajmniej jedno ze słów $\gamma_1, \gamma_2$ musi zawierać przynajmniej jedno $a$. Nietrudno widzieć, że musi zachodzić $\gamma_1 \in \{a\}\{a\}^*$ oraz $\gamma_2 \in \{a\}^* \cup \{b\}^* \cup \{c\}^*$. W każdym przypadku możemy łatwo dojść do sprzeczności. Przypuśćmy na przyład, że $\gamma_2 \in \{b\}^*$. Niech $p = |\gamma_1|$, $q = |\gamma_2|$. Mamy $1 \leq p \neq n$, niech więc $p_1 = n!/p$. Wówczas, z trzeciego punktu lematu Ogdena, również słowo

$$a^{n+p\cdot(2p_1)} b^{n+p\cdot(2\ p_1)} c^{n+2\cdot n!}$$

powinno należeć do $L$, ale $n + p \cdot (2p_1) = n + 2 \cdot n!$, co przeczy definicji $L$. □

# 4 Własności domknięcia

W tym punkcie rozważymy *własności domknięcia*, tzn. na jakie operacje jest, a na jakie nie jest zamknięta klasa języków bezkontekstowych. Dowody poniższych faktów pozostawiamy jako ćwiczenia.

**Fakt 2** *Rodzina języków bezkontekstowych jest zamknięta na operacje skończonej sumy, konkatenacji i iteracji ("gwiazdki").*

**Uwaga**. Rodzina języków bezkontekstowych nie jest natomiast zamknięta na operacje przecięcia i uzupełnienia. Istotnie, niech

$$M = \{a^n b^n a^n \,:\, n \in N\}$$

będzie językiem, o którym wykazaliśmy, że nie jest bezkontekstowy (Przykład 1). Nietrudno pokazać, że

$$\begin{aligned} M &= \{a^n b^n \,:\, n \in N\}a^* \cap a^*\{b^n a^n \,:\, n \in N\} \\ M &= \{a,b\}^* - \{a^i b^j a^k \,:\, i \neq j \text{ lub } j \neq k\} \end{aligned}$$

gdzie języki występujące po prawych stronach równości *są* bezkontekstowe.

**Fakt 3** *Rodzina języków bezkontekstowych jest zamknięta na przecięcia z językami regularnymi, tzn. jeśli język $L$ jest bezkontekstowy a język $R$ jest* regularny*, to język $L \cap R$ jest bezkontekstowy.*

**Fakt 4** *Rodzina języków bezkontekstowych jest zamknięta na ilorazy przez języki regularne, tzn. jeśli język $L$ jest bezkontekstowy a język $R$ jest* regularny*, to języki $LR^{-1}$ i $R^{-1}L$ są bezkontekstowe.*

**Fakt 5** *Rodzina języków bezkontekstowych jest zamknięta na* podstawienia*, tzn. jeśli język $L$ jest bezkontekstowy i $f : \Sigma \to P(\Gamma^*)$ jest funkcją taką, że, dla każdego $\sigma \in \Sigma$, $f(\sigma)$ jest językiem bezkontekstowym, to $\hat{f}(L)$ jest językiem bezkontekstowym.*

**Fakt 6** *Rodzina języków bezkontekstowych jest zamknięta na przeciwobrazy homomorficzne, tzn. jeśli $f : \Sigma \to \Gamma^*$ jest homomorfizmem i $M \subseteq \Gamma^*$ jest bezkontekstowy, to język*

$$\hat{f}^{-1}(M) = \{w \in \Sigma^* \,:\, \hat{f}(w) \subseteq M\}$$

*jest bezkontekstowy.*

# Ćwiczenia

1. Dowieść, że każdy język bezkontekstowy nad jednoliterowym alfabetem jest regularny.

   *Wskazówka.* Skorzystać z lematu o pompowaniu i relacji przystawania *modulo*.

2. Czy przy pomocy automatu skończonego można rozpoznać, ze w słowie nad alfabetem $\{a, b, c\}$ jedna z liter występuje co najmniej tyle samo razy co dwie pozostałe? Czy można rozpoznać tę własność przy pomocy automatu ze stosem?

3. Dowieść, że język $\{ww : w \in \Sigma^*\}$ nie jest bezkontekstowy, o ile $|\Sigma| \geq 2$.

4. Niech $L$ będzie językiem regularnym. Które z następujących języków są regularne lub bezkontekstowe?

   - $\{ww : w \in L\}$
   - $\{ww^R : w \in L\}$
   - $\{wv^R : w, v \in L\}$

5. Dowieść, że język $\{a^i b^j c^k : i \neq j,\ j \neq k\}$ nie jest bezkontekstowy.

6. Dowieść, że język $\{w\$v : v \text{ jest podsłowem } w\}$ nie jest bezkontekstowy, natomiast język $\{w\$v^R : v \text{ jest podsłowem } w\}$, jest bezkontekstowy.

7. Czy następujacy zbiór jest językiem bezkontekstowym:

   > Zbior słów $p\$w$, gdzie $p$ jest wyrażeniem regularnym nad alfabetem $\{a, b\}$, a $w$ jest słowem należącym do języka wyznaczonego przez $p$.

   Analogiczne pytanie dla $p\$w^R$.

8. Niech $A$ – alfabet, $A' = \{a' : a \in A\}$. Określamy homomorfizmy $h, h_1, h_2$ z $A \cup A'$ w $A$ następująco: $h(a) = h(a') = a$, $h_1(a) = a$, $h_1(a') = \epsilon$, $h_2(a) = \epsilon$, $h_2(a') = a$.

   Definiujemy *przeplot* języków $L_1$ i $L_2$:

   $$\text{Shuffle}(L_1, L_2) = \{x : (\exists y \in h^{-1}(x)) h_1(y) \in L_1 \wedge h_2(y) \in L_2\}$$

   Dowieść, że

   (a) przeplot dwóch jezyków regularnych jest regularny,

   (b) przeplot języka bezkontekstowego z regularnym jest bezkontekstowy,

   (c) przeplot dwóch jezykow bezkontekstowych nie musi być bezkontekstowy (wskazówka: $\{a^n b^m c^n d^m : n, m \in N\}$ nie jest bezkontekstowy)

9. Język bezkontekstowy nazywamy *liniowym*, jeśli można go wygenerować gramatyką, w której po prawej stronie dowolnej reguły występuje co najwyżej jedna zmienna. Dowieść, że następujące języki bezkontekstowe nie są liniowe:

   (a) $\{a^i b^i c^j d^j : i, j \in \mathbf{N}\}$,

   (b) $L = \{x \in (a + b)^*, \#_a(x) = \#_b(x)\}$

   *Wskazówka.* Sformułować lemat o pompowaniu dla języków liniowych oparty na dekompozycji $\alpha = \alpha_1\gamma_1\beta\gamma_2\alpha_2$, w której ograniczone jest $\alpha_1\gamma_1\gamma_2\alpha_2$.