Teoretyczne podstawy informatyki

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. letni 2025/26

`inf.ug.edu.pl/~amb/TPI`

Wstęp

- Literatura
  - J. Jędrzejowicz, A. Szepietowski, *Języki, automaty, złożoność obliczeniowa*, Wyd. UG, 2008
  - J. E. Hopcroft, J. D. Ulmann, *Wprowadzenie do teorii automatów, języków i obliczeń*, PWN, 1994 (nowsze wydanie ma współautora R. Motwani)
- zaliczenia
  - brak ćwiczeń, wykład obowiązkowy
  - egzamin końcowy

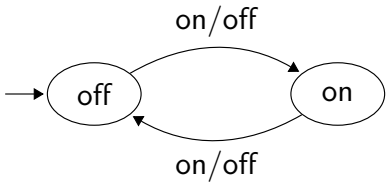
Przegląd tematyki

- automaty – idealizacja obliczeń, zmiana stanu pod wpływem czynników zewnętrznych
- języki, automaty skończone, deterministyczne i niedeterministyczne, generujące języki
- wyrażenia i języki regularne
- lemat o pompowaniu dla języków regularnych, twierdzenie Myhill-Nerode'a i minimalizacja automatu
- gramatyki i języki bezkontekstowe
- niedeterministyczne automaty ze stosem, równoważność gramatyk bezkontekstowych i niedeterministycznych automatów ze stosem
- postać normalna Chomsky'ego, lemat o pompowaniu dla języków bezkontekstowych
- języki deterministyczne i jednoznaczne
- maszyny Turinga

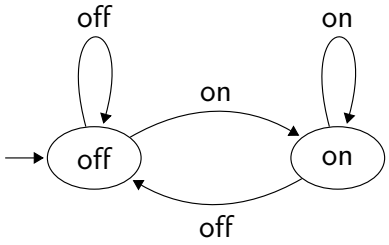
Automaty skończone Języki regularne

Automaty skończone

Pilot do telewizora



Kluczyk do samochodu



Alfabety, słowa

- *alfabet* Σ lub A – zbiór skończony, niepusty, w nim symbole/litery
- *słowo* – skończony ciąg elementów z A
  mogą się powtarzać (w zbiorze nie)
  znamy kolejność tych elementów (w zbiorze nie)
- A^* – zbiór wszystkich słów
  zawsze jest słowo puste $\varepsilon \in A^*$
  istnieją ciągi dowolnej długości: np. $\spadesuit \in A$, $\spadesuit\spadesuit\spadesuit\spadesuit\ldots\spadesuit \in A^*$ czyli A^* jest zawsze nieskończony
- $\{0,1\}^*$ – ciągi bitów, więcej w informatyce nie trzeba
- inne możliwości – $A = \{a, b\}$, $A = \{a, b, c, \#\}$, A jest alfabetem łacińskim, albo polskim, albo znaków alfanumerycznych, albo ASCII, albo ASCII latin-2, albo UTF-8 w różnych odmianach, albo ...
- jeśli ograniczymy długość słów, to $\{w \in A^* \mid |w| \leq max\}$ jest skończony, moc $\leq \Sigma_{i=0}^{max}|A|^i < |A|^{m+1}(|A|-1)^{-1}$, jeśli $|A| \geq 2$

Języki

- język nad alfabetem A – dowolny zbiór słów nad A, czyli $L \subseteq A^*$
- np. $L = \emptyset$, albo $L = \{\varepsilon\}$, albo $L = A^*$
  albo skończony zbiór podany enumeratywnie np. $A = \{0, 1\}$, $L = \{10011, 0011101, 0, 10101, \varepsilon, 110\}$
- A = litery polskie, L = słowa w pewnym słowniku języka polskiego
- A = znaki z klawiatury, L = poprawne programy w pewnym języku programowania (C, C++, C#, itd.)
- $A = \{0, 1\}$, L liczby podzielne przez 2 (słowa traktujemy jako zapis binarny), albo przez 5, albo liczby liczby pierwsze
- przyjmujemy, że $A \subseteq A^*$, ciąg jednolementowy utożsamiamy z tym elementem
  $L = A$ też jest językiem nad A
- pytanie: w jaki sposób definiujemy języki

Problem generyczny

- dany język L nad alfabetem A, dane słowo $u \in A^*$

  czy $u \in L$

- czy dana liczba jest pierwsza (język liczb pierwszych)
- czy dana formuła boolowska jest tautologią (język tautologii boolowskich)
- czy dany program w Python v3.11 jest syntaktycznie poprawny (Python v3.11)
  (nie pytamy czy działa w ogóle, czy działa sensownie, czy jest w miarę optymalny, czy jest dobrze napisany i łatwy w analizie przez innych)
- czasami nawet pytanie czy $L \neq \emptyset$ jest nieoczywiste

Operacje na słowach

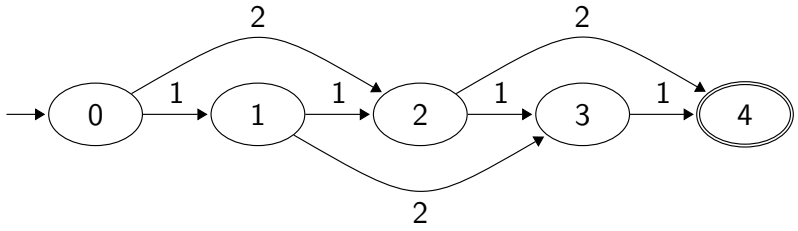
- *konkatenacja* – $u, v \in A^*$, $u = u_1 \dots u_k$, $v = v_1 \dots v_\ell$, wówczas $u \cdot v$ (ozn. również uv) jest równe $u = u_1 \dots u_k v_1 \dots v_\ell$
- w szczególności $u \cdot \varepsilon = \varepsilon \cdot u = u$
  łączna $u \cdot (v \cdot w) = (u \cdot v) \cdot w$
  nieprzemienna, np. $abba \cdot baba \neq baba \cdot abba$, chociaż $11 \cdot 1 = 1 \cdot 11$
- *podsłowo* = podciąg kolejnych elementów, czyli $u \sqsubset v$ wtw $\exists\, x, y.v = xuy$
- *przedrostek* (prefiks) – $\exists\, y.v = uy$
  przyrostek (postfiks) – $\exists\, x.v = xu$
- długość – $|u_1 \dots u_k| = k$, $|\varepsilon| = 0$
  $|u|_a$ – liczba wystąpień litery $a \in A$ w słowie $u \in A^*$
- dla $u = u_1 \dots u_k$ odbiciem jest $u^R = u_k \dots u_1$

Operacje na językach

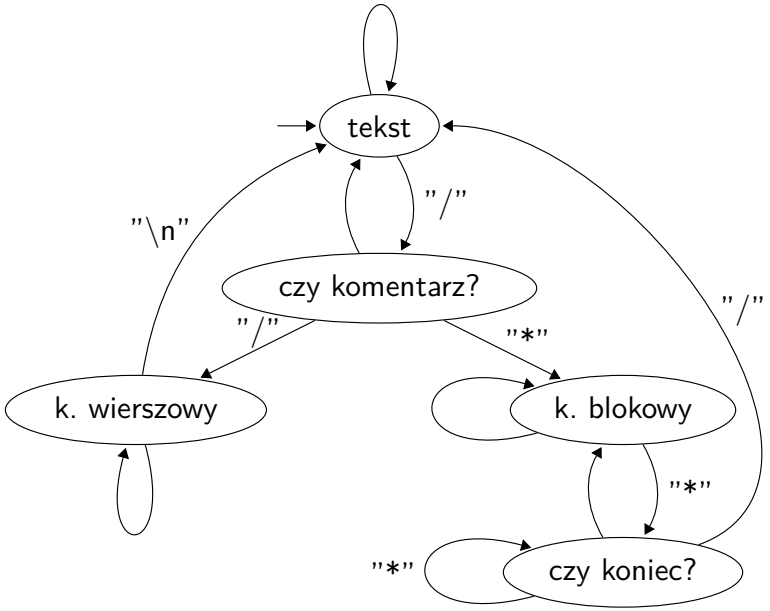
- alfabet jest ustalony, czasami może być odtworzony
- jeśli $A \subseteq B$, to $A^* \subseteq B^*$
  czyli jeśli L jest językiem nad A to również nad B
  jeśli $L \subseteq B^*$ ale wszystkie użyte litery są z A, to $L \subseteq A^*$
- przekrój $L_1 \cap L_2$, suma $L_1 \cup L_2$ (ozn. $L_1 + L_2$), różnica $L_1 \setminus L_2$
- konkatenacja $L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1, v \in L_2\}$
- $L^{\ell+1} = L \cdot L^\ell$, $L^1 = L$, $L^0 = \{\varepsilon\}$
- $L^* = \bigcup_{i=0}^{\infty} L^i$, $L^+ = \bigcup_{i=1}^{\infty} L^i$
- np. dla $L = A \subseteq A^*$, $L^* = A^*$
  a jeśli $L = A^2$, to $L^* = \{u \in A^* \mid |u| \text{ parzysta}\}$

Automaty skończone

- automat przyjmuje monety 1 zł i 2 zł i wydaje żeton gdy otrzyma 4 zł
- musi być zapamiętana suma już wrzuconych monet, nie ma szans by w jednym kroku zakończyć działanie
- mamy stan początkowy oraz akceptujący, gdy wydawany jest żeton
- wrzucenie monety w danym stanie ma jednoznaczny skutek, wiadomo dokąd prowadzi przejście



Przykład: program usuwający komentarze w języku C



Automat skończony, deterministyczny

- $\mathcal{A} = \langle Q, A, \delta, q_0, F \rangle$ gdzie:
  Q – skończony zbiór stanów,
  $q_0 \in Q$ – stan początkowy,
  $F \subseteq Q$ – zbiór stanów końcowych (akceptujących),
  A – alfabet, którym etykietowane są przejścia,
  δ funkcja częściowa określająca przejście stanów pod wpływem odczytania symbolu: $(q, a) \mapsto \delta(q, a)$.
- w sumie graf skierowany z etykietowanymi strzałkami
- automat $\mathcal{A}$ akceptuje słowo $u = u_1 \dots u_\ell$ jeśli istnieje ścieżka w automacie od stanu początkowego q_0 do (dowolnego) stanu końcowego etykietowana słowem u
- uwaga: można spowodować, że δ jest funkcją całkowitą, poprzez dodanie dodatkowy stanu "blokady", do którego trafiają wszystkie brakujące przejścia

Obliczenia w automacie

- dany automat $\mathcal{A}$, stan $q \in Q$ i słowo $u = u_1 \dots u_\ell$
  konfiguracja qu oznacza, że automat znajduje się w stanie q a na wejściu jest słowo u
  jeśli $\delta(q, u_1)$ jest określone, to przechodzimy do konfiguracji $\delta(q, u_1)u_2 \dots u_\ell$, czyli kolejny stan i przyrostek słowa
- obliczenie – ciąg kolejnych konfiguracji
- automat przeczytał całe słowo jeśli konfiguracja ma postać $q\varepsilon$
  automat *akceptuje* słowo u jeśli $q \in F$
- automat *nie akceptuje* słowa u
  albo jeśli w obliczeniu dochodzimy do $q\varepsilon$ ale $q \notin F$
  albo jeśli dochodzimy do konfiguracji qu i $\delta(q, u_1)$ nie jest określone
- druga możliwość nie będzie zachodzić, jeśli założymy, że funkcja przejścia jest całkowita

Przykład

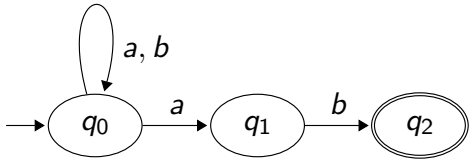
- $\mathcal{A} = \langle Q, A, \delta, q_0, F \rangle$ gdzie:
  $Q\{q_0, q_1, q_2, q_3\}$ – skończony zbiór stanów,
  $q_0 \in Q$ – stan początkowy,
  $F = \{q_0\} \subseteq Q$ – zbiór stanów końcowych (akceptujących),
  $A = \{0, 1\}$ – alfabet, którym etykietowane są przejścia,
  δ funkcja przejścia:

δ	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

- zadanie: obliczyć język generowany przez ten alfabet
- odp.: parzysta liczba zer i parzysta liczba jedynek (dowód wymaga analogicznego stwierdzenia dla wszystkich stanów)

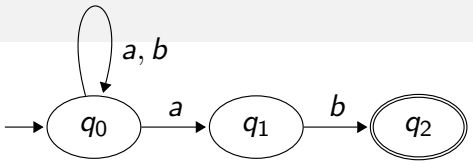
Automat niedeterministyczny

- wszystko jak dotychczas ale nie ma funkcji przejścia, jest relacja
- tzn. możliwy jest przypadek, że przejściem jest zarówno $(q, a) \mapsto p_1$ jak i $(q, a) \mapsto p_2$, $q, p_1, p_2 \in Q$, $p_1 \neq p_2$
- obliczenie nie jest deterministyczne, konfiguracja qu może prowadzić do wielu kolejnych konfiguracji, mamy całe drzewo możliwych obliczeń



- zarówno $q_0 a \mapsto q_0\varepsilon$ jak i $q_0 a \mapsto q_1\varepsilon$
- automat akceptuje słowo u, jeśli *istnieje* obliczenie prowadzące od $q_0 u$ do $q\varepsilon$, $q \in F$, mimo, że inne obliczenia nie dają pozytywnego wyniku

Przykład c.d.



- słowo *abbab* jest akceptowane:
  $q_0abbab \mapsto q_0bbab \mapsto q_0bab \mapsto q_0ab \mapsto q_1b \mapsto q_2$
- ale pewne obliczenia prowadzą na manowce
  $q_0abbab \mapsto q_1bbab \mapsto q_2bab$ i nie ma dalszego ruchu choć słowo jeszcze się nie skończyło, albo
  $q_0abbab \mapsto q_0bbab \mapsto q_0bab \mapsto q_0ab \mapsto q_0b \mapsto q_0$
- tw.: powyższy automat akceptuje słowa kończące się ciągiem *ab*
- dw: dowodzimy przez indukcję, że
  dla dowolnego słowa u, $q_0 \in \delta_2(q_0, u)$
  $q_1 \in \delta_2(q_0, u)$ wtw gdy u kończy się na a
  $q_2 \in \delta_2(q_0, u)$ wtw gdy u kończy się na ab

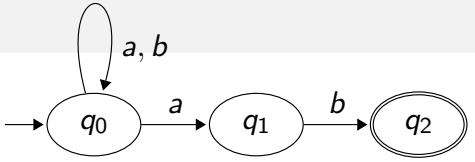
Automaty niedeterministyczne a determistyczne

- dany automat niedeterministyczny z funkcją przejścia
  $\delta : Q \times A \to \mathcal{P}(Q)$
- przedłużamy tę funkcję na wszystkie słowa, $\delta' : Q \times A^* \to \mathcal{P}(Q)$
  - $\delta'(q, \varepsilon) = \{q\}$
  - $\delta'(q, ua) = \bigcup_{p \in \delta'(q,u)} \delta(p, a)$
- czyli $L = \{w \in A^* \mid \delta'(q_0, w) \cap F \neq \emptyset\}$
- konstrukcja ta pozwala zdefiniować nowy automat, którego stanami będą zbiory stanów oryginalnego automatu

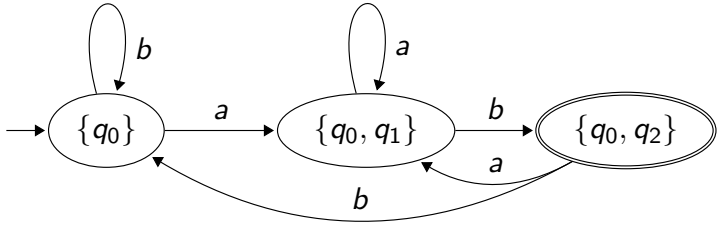
Automaty niedeterministyczne a determistyczne, c.d.

- automaty $\mathcal{A}_1$ i $\mathcal{A}_2$ nazywamy równoważnymi, jeśli języki akceptowane są identyczne
- tw.: dla każdego automatu niedeterministycznego $\mathcal{A}_1$ istnieje równoważny ale deterministyczny $\mathcal{A}_2$
- dw.: $Q_2 = \mathcal{P}(Q_1)$, nowy stan początkowy to $\{q_0\}$,
  $\delta_2(q_2, a) = \bigcup_{q \in q_2} \delta_1(q, a)$ to zbiór stanów, do których można dojść w $\mathcal{A}_1$ zaczynając od dowolnego stanu $q \in q_2$,
  $q_2 \in Q_2$ jest akceptujący wtw. jeśli zawiera stan akceptujący w Q_1,
  $F_2 = \{q_2 \in \mathcal{P}(Q_1) \mid q_2 \cap F_1 \neq \emptyset\}$
- pokazujemy przez indukcję, że dla wszystkich słów $u \in A^*$,
  $\delta'_2(\{q_0\}, u) = \delta'_1(q_0, u)$,
  czyli języki akceptowane przez $\mathcal{A}_1$ i $\mathcal{A}_2$ są identyczne
- nie wszystkie zbiory stanów w Q_1 wystąpią w obliczeniach stanów, do których można dojść z q_0,
  elementy nieosiągalne w $\mathcal{P}(Q_1)$ możemy usunąć ze zbioru Q_2

Przykład c.d.



- w przykładzie $\delta_2(\{q_0\}, a) = \{q_0, q_1\}$,
  $\delta_2(\{q_0, q_1\}, b) = \{q_0, q_2\}$, $\delta_2(\{q_0, q_2\}, b) = \{q_0\}$
- obliczenie: $\{q_0\}abbab \mapsto \{q_0, q_1\}bbab \mapsto \{q_0, q_2\}bab \mapsto \{q_0\}ab \mapsto \{q_0, q_1\}b \mapsto \{q_0, q_2\}$ stan akceptujący (bo $q_2 \in F_1$)



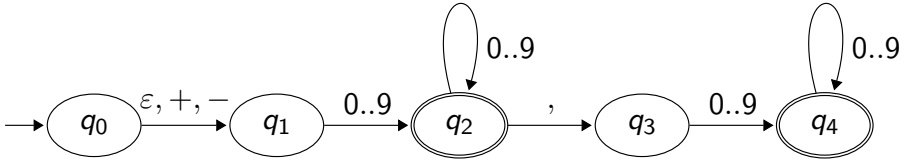
- inne stany automatu $\mathcal{P}(\{q_0, q_1, q_2\})$ nie są osiągalne ze stanu początkowego, można je pominąć

Przykład jeszcze jeden

- $A = \{0, 1\}$, $L = \{w \in A^* \mid w_{|w|-100} = 1\}$
- czyli ciąg jest zaakceptowany, jeśli ma długość co najmniej 100 i na setnym miejscu od końca stoi jedynka
- automat niedeterministyczny ma 101 stanów, większość przejść jest dostępna dla zera i jedynki, ale z q_0 do q_1 prowadzi tylko 1
- implementacja jest oczywista: pamiętamy ostatnich 100 znaków, na końcu sprawdzamy czy 100 od końca to 1
- automat deterministyczny musi pamiętać wszystkie możliwe stany rejestru, czyli 2^{100} możliwości

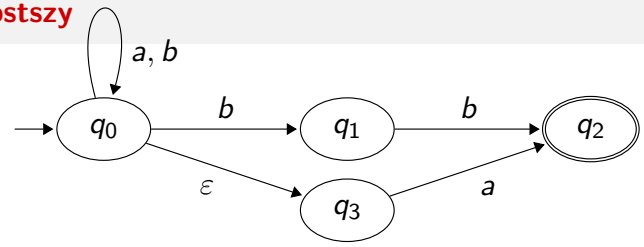
Automaty z cichymi przejściami

- oprócz dotychczasowych możliwości dopuszczalne są przejścia bez odczytu symbolu z alfabetu
- przykład: liczba rzeczywista – na początku może ale nie musi być znak, potem niepusty ciąg cyfr, może przecinek, jeśli jest to dalej niepusty ciąg cyfr



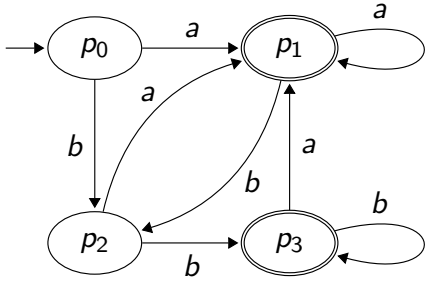
- tw.: dla każdego automatu z cichymi przejściami istnieje równoważny automat deterministyczny
- dowód analogiczny jak dla niedeterminizmu

Przykład prostszy



- produkuje słowa kończące się albo podwójnym b albo pojedynczym a
- definicja automatu deterministycznego

$\delta(p, x)$	a	b
$p_0 = \{q_0\}$	p_1	p_2
$p_1 = \{q_0, q_2, q_3\}$	p_1	p_2
$p_2 = \{q_0, q_1, q_3\}$	p_1	p_3
$p_3 = \{q_0, q_1, q_2, q_3\}$	p_1	p_3



- końcowe są stany p_1 i p_3

Operacje na językach regularnych

- problem: dane języki regularne L_1, L_2, co można powiedzieć o sumie $L_1 \cup L_2$, przekroju $L_1 \cap L_2$, różnicy $L_1 \setminus L_2$, konkatenacji $L_1 \cdot L_2$, iteracji L_1^*?
- dopełnienie $L_1^c = A^* \setminus L_1$ wymaga ustalenia przedtem alfabetu, w poprzednich przykładach alfabetem były wszystkie symbole wystepujące w językach
- tw.: klasa języków regularnych jest zamknięta na wszystkie wymienione operacje

Iloczyn kartezjański automatów

- dane dwa automaty (deterministyczne) $\mathcal{A}_1$ i $\mathcal{A}_2$ nad tym samym alfabetem A, definiujące języki L_1 i L_2, funkcje przejścia całkowite
- w iloczynie kartezjańskim $Q_1 \times Q_2$ definiujemy przejścia jako skoordynowane przejścia, tj. $\delta(\langle q_1, q_2\rangle, a) = \langle \delta_1(q_1, a), \delta_2(q_2, a)\rangle$
- czyli każdy automat działa niezależnie i rejestrujemy ich stany
- stan początkowy jest parą stanów początkowych
- $L_1 \cap L_2$ będzie definiowany przez automat, którego stan akceptujący jest parą dwóch stanów akceptujących
- $L_1 \cup L_2$ będzie definiowany przez automat, który zaakceptuje gdy jedna ze współrzędnych będzie stanem akceptującym
- można też określić automaty akceptujące obie różnice języków

Operacja dopełnienia

- dany automat deterministyczny $\mathcal{A} = \langle Q, A, \delta, q_0, F\rangle$
- zakładamy dodatkowo, że funkcja przejścia jest całkowita
- wówczas dla każdego słowa u istnieje dokładnie jedno obliczenie $q_0 u \to \ldots \to q\varepsilon$
- definiujemy automat $\mathcal{A}' = \langle Q, A, \delta, q_0, Q \setminus F\rangle$
- czyli automat działający dokładnie tak samo tylko odwracający znaczenie akceptowania, akceptuje dokładnie te słowa, które nie są akceptowane przez oryginalny automat
- automat ten definiuje dopełnienie języka $A^* \setminus L(\mathcal{A})$
- dowód, że $L_1 \setminus L_2$ jest językiem regularnym można przeprowadzić również zauważając, że $L_1 \setminus L_2 = L_1 \cap (A^* \setminus L_2)$

Konkatenacja/ iteracja

- tw.: dane języki regularne z odpowiadającymi automatami $\mathcal{A}_1$ oraz $\mathcal{A}_2$, ich konkatenacja też jest językiem regularnym
- dowód: budujemy automat jako rozłączną sumę stanów obu automatów, wszystkie przejścia pozostają w mocy
  - stanem początkowym jest stan początkowy $\mathcal{A}_1$
  - akceptują stany akceptujące $\mathcal{A}_2$
  - dodajemy ciche przejścia ze stanów akceptujących $\mathcal{A}_1$ do stanu początkowego $\mathcal{A}_2$
- tw.: iteracja języka $L(\mathcal{A})^*$ jest językiem regularnym
- dowód: budujemy nowy automat dodając ciche przejścia ze stanów akceptujących $\mathcal{A}$ do stanu początkowego, dodatkowo dołączamy stan początkowy jako akceptujący
- dla $L(\mathcal{A})^+$ nie będzie potrzeby sztucznego dołączania stanu początkowego do akceptujących