Obliczalność i złożoność

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. zimowy 2025/26

inf.ug.edu.pl/~amb/oblicz

Wstęp

- Literatura
  - J. Jędrzejowicz, A. Szepietowski, *Języki, automaty, złożoność obliczeniowa*, Wyd. UG, 2008
  - J. E. Hopcroft, J. D. Ulmann, *Wprowadzenie do teorii automatów, je'zyków i obliczeń*, PWN, 1994 (nowsze wydanie ma współautora R. Motwani)
- zaliczenia
  - ćwiczenia, kolokwium zaliczeniowe
  - egzamin końcowy

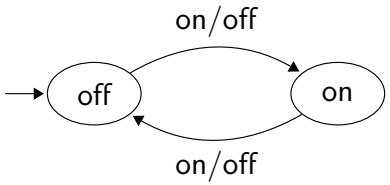
Obliczalność i złożoność

Przegląd tematyki

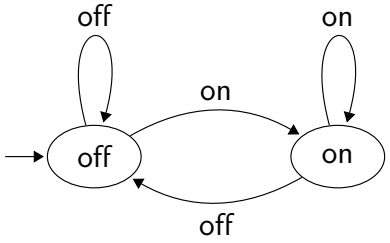
- automaty – idealizacja obliczeń, zmiana stanu pod wpływem czynników zewnętrznych
- maszyny Turinga – abstrakcyjne pojęcie komputera, Alan Turing opracował pojęcie, potem zaimplementowano je używając elektroniki
- są prostsze urządzenia niż maszyny Turinga: automaty skończone, automaty ze stosem
- złożoność obliczeniowa – niektóre obliczenia są nierealistycznie długie, klasa problemów NP-trudnych
- specyfikacja – wyrażenia regularne, gramatyki

Automaty skończone

Pilot do telewizora

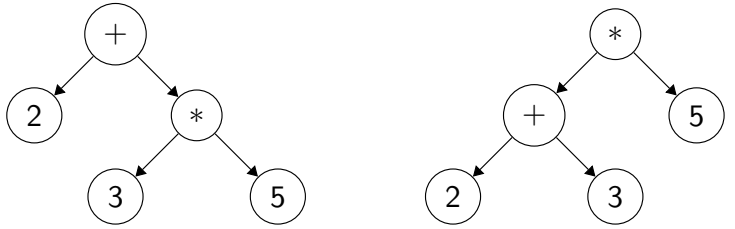


Kluczyk do samochodu



Gramatyki

- Wyrażenie ::= Liczba | Wyrażenie Operator Wyrażenie
  Operator ::= $+ | - | * | \div$
  Liczba (wiadomo jakie są)
- $2 + 3 * 5$ jest wyrażeniem arytmetycznym
- można to udowodnić na dwa sposoby
- albo pierwszym wyrażeniem jest liczba 2 a drugim iloczyn, czyli jakby $2 + (3 * 5)$
- albo pierwszym wyrażeniem jest suma a drugim liczba, czyli $(2 + 3) * 5$



Języki regularne, automaty skończone

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
- jeśli ograniczmy długość ciągów, to A^{max} jest skończony, $|A^{max}| = |A|^{max}$

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 $u^R = u_k \dots u_1$

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 $u \in L$

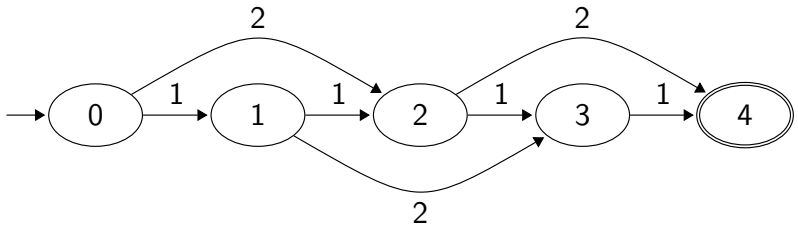
- czy dana liczba jest pierwsza?
- czy dany program w Python v3.11 jest syntaktycznie poprawny? (nie pytamy czy działa w ogóle, czy działa sensownie, czy jest w miarę optymalny, czy jest dobrze napisany i łatwy w analizie przez innych)
- czasami pytanie czy $L \neq \emptyset$ jest nieoczywiste

Operacje na językach

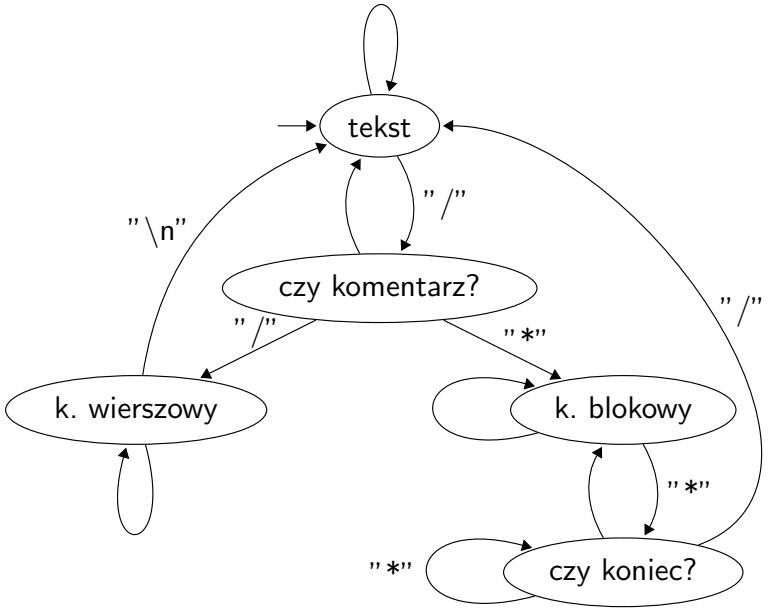
- alfabet jest ustalony, czasami może być wyliczony
- jeśli $A \subseteq B$, to $A^* \subseteq B^*$
  czyli jeśli L jest językiem nad A to również nad B
  jeśli $L \subseteq B^*$ ale wszystkie użyte litery są z A, to $L \subseteq A^*$
- przekrój $L_1 \cap L_2$, suma $L_1 \cup L_2$ (ozn. $L_1 + L_2$),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



Automat skończony, deterministyczny

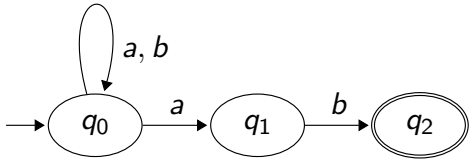
- $\mathcal{A} = \langle Q, A, \delta, q_0, F \rangle$ gdzie:
  Q – skończony zbiór stanów,
  $q_0 \in Q$ – stan początkowy,
  $F \subseteq Q$ – zbiór stanów końcowych (akceptujących),
  A – alfabet, którym etykietowane są przejścia,
  δ funkcja częściowa określająca przejście stanów pod wpływem symbolu: $(q, a) \mapsto \delta(q, a)$.
- 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: δ może być f. 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 *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

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
- 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



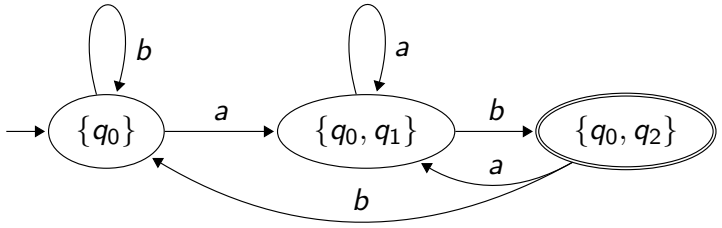
- słowo *abbab* jest akceptowane, ale pewne obliczenia prowadzą na manowce

Automaty niedeterministyczne a determistyczne

- 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 $\{q_0\}$
  $\delta_2(q_2, a)$ 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, jeśli zawiera stan akceptujący w Q_1
- w przykładzie $\delta_2(\{q_0\}, a) = \{q_0, q_1\}$,
  $\delta_2(\{q_0, q_1\}, b) = \{q_0, q_2\}$,
  $\delta_2(\{q_0, q_2\}, b) = \{q_0\}$
  $\{q_0\}abbab \mapsto \{q_0, q_1\}bbab \mapsto \{q_0, q_2\}bab \mapsto \{q_0\}ab \mapsto \{q_0, q_1\}b \mapsto$
  $\{q_0, q_2\}$ stan akceptujący (bo $q_2 \in F_1$)

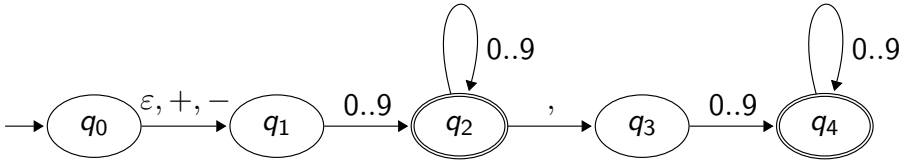
Przykład c.d.

- tw.: powyższy automat akceptuje dokładnie 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
- równoważny automat deterministyczny (po odrzuceniu nieosiągalnych stanów)



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

Wyrażenia regularne

- służą do definiowania języków (regularnych)
- dany alfabet A, wyrażenia regularne to m.in. $\emptyset$, ε, a dla każdego $a \in A$ definiują one języki $\emptyset$, $\{\varepsilon\}$, $\{a\}$
- jeśli E i F są wyrażeniami regularnymi, to są nimi również $E + F$, EF, E^*, (E)
  definiują one języki $L(E) \cup L(F)$, $L(E) \cdot L(F)$, $L(E)^*$ oraz $L(E)$
- np. $(\varepsilon + a)(ba)^*(\varepsilon + b)$
  definiuje język, w którym a i b wystepują naprzemian, nie narzucamy czy musi zaczynać się lub kończyć konkretną literą
  słowo puste również wchodzi do tego języka

Twierdzenie Kleene'ego

- wyrażenia regularne są równoważne automatom skończonym (tzn. generują te same języki)
- dw. cz.1: dla wyrażenia budujemy automat (z cichymi przejściami) generujący ten sam język
  - dla $\emptyset$ jedyny stan początkowy nie jest akceptujący
  - dla ε jedyny stan początkowy jest akceptujący
  - dla $a \in A$ automat ma jedną strzałkę etykietowaną a
  - suma: ze stanu początkowego mamy dwa ciche przejścia do stanów początkowych obu automatów
  - konkatenacja: dodajemy ciche przejścia ze stanów akceptujących pierwszego automatu do początkowego drugiego
  - gwiazdka Kleene'ego: ciche przejścia ze stanów akceptujących z powrotem do początkowego
  - nawiasy niczego nie zmieniają

Twierdzenie Kleene'ego, c.d.

- dw. cz.2: dla automatu deterministycznego budujemy wyrażenie regularne
  niech $Q = \{q_0, \ldots, q_n\}$
  - niech $R^k_{i,j}$ zawiera słowa, które pozwalają przejść od q_i do q_j poprzez stany pośrednie $q_0, \ldots, q_{k-1}$
  - te języki są generowane przez wyrażenia regularne
  - $r^0_{i,j} = \Sigma_{\delta(q_i,a)=q_j} a$, pusta suma $\emptyset$, jeśli $i = j$ dodatkowo $+\varepsilon$,
  - czyli wyłącznie bezpośrednie przejścia z q_i do q_j
  - $r^{k+1}_{i,j} = r^k_{i,j} + r^k_{i,k} \, (r^k_{k,k})^* \, r^k_{k,j}$
  - czyli użycie q_k oznacza przejście do q_k, być może pętlenie się w tym stanie, i przejście do q_j

  osteczne wyrażenie regularne jest sumą $r^k_{i,j}$ dla $k = n + 1$ (wszystkie stany wolno odwiedzać), $i = 0$ (zaczynamy od stanu początkowego) dla wszystkich j takich, że $q_j \in F$ (i kończymy w stanie akceptującym)

Wyrażenia regularne w językach programowania

- oprócz gwiazdki $*$ – dowolnego powtórzenia, są jeszcze wersje
  $+$ – powtórzenie co najmniej jeden raz
  ? – element opcjonalny, wystąpienie 0 lub 1 raz
  różne wersje określające dokładniej liczbę wystąpień, minimalną, maksymalną, zakres
- oznaczenia na całe zbiory symboli
  [...] – wylicza możliwy zbiór symboli
  są oznaczenia na uzupełnienie zbioru, tylko litery, tylko cyfry, znaki białe, zakresy znaków
  poszukiwanie wzorca na początku lub na końcu wiersza
- przy operacji zamiany wzorca na inny można odwoływać się do znalezionych fragmentów wzorca
- zawsze trzeba przestudiować dokumentację, różne języki mają pewne możliwości a pewnych nie