Teoretyczne podstawy informatyki

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. letni 2025/26

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

Algorytmy decyzyjne, c.d.

Problem zawierania się języków $L_1 \subseteq L_2$?

- niech L_1 i L_2 będą językami regularnymi zdefiniowanymi za pomocą automatów deterministycznych z całkowitą funkcją przejścia, wówczas łatwo zbudować automat dla dopełnienia $\overline{L_2}$, problem $L_1 \subseteq L_2$ jest równoważny $L_1 \cap \overline{L_2} = \emptyset$
  - złożoność to iloczyn liczb stanów w obu automatach
  - ale jeśli definicja używa automatu niedeterministycznego lub wyrażenia regularnego, to złożoność może być większa
- jeśli L_1 jest językiem bezkontekstowym a L_2 regularnym, to powyższe rozumowanie też prowadzi do algorytmu dla rozstrzygnięcia
- tw.: jeśli L_1 jest językiem regularnym, L_2 bezkontekstowym, to *nie ma* algorytmu rozstrzygającego ten problem
  - w szczególności równość dwóch języków bezkontekstowych, w szczególności pytanie, czy $L = A^*$?

Dalsze problemy nierozstrzygalne

- dopełnienie języka bezkontekstowego L może nie być bezkontekstowe, mieliśmy przykłady
- tw.: nie ma algorytmu rozstrzygającego czy dopełnienie języka L jest bezkontekstowe
- tw.: nie ma algorytmu rozstrzygającego czy dopełnienie języka L jest puste (było przed chwilą)
- tw.: nie ma algorytmu rozstrzygającego czy dopełnienie języka L jest skończone
- wszystkie powyższe twierdzenia wymagają innych metod niż dotychczas rozważane

Języki rozpoznawane przez deterministyczne automaty ze stosem

- problem $L_1 \subseteq L_2$ jest równoważny $L_1 \cap \overline{L_2} = \emptyset$
- jeśli L_1 i L_2 są DPDA, dopełnienie też jest w tej klasie (podobnie jak dla języków regularnych)
- ale przekrój nie musi być nawet bezkontekstowy, więc ten argument nie przydaje się w dowodzie
- prawdziwe jest jednak twierdzenie (1997), daleko poza możliwościami tego wykładu: problem $L_1 \subseteq L_2$ dla języków z klasy DPDA jest rozstrzygalny (tzn. istnieje algorytm)

Maszyny Turinga

Alan Turing

- Alan Turing (1912-1954) – angielski matematyk, logik, informatyk
- kierował kryptoanalizą niemieckiej maszyny szyfrującej Enigma
- zaprojektował architekturę komputerów, t.j. maszyn wykonywujących dowolny algorytm – maszyny Turinga
- pojęcie *obliczalności*
- test Turinga – odróżnienie człowieka od maszyny
- CAPTCHA – *Completely Automated Public Turing test to tell Computers and Humans Apart*

Maszyny Turinga

- automat z nieskończoną taśmą wejściową
- głowica automatu może poruszać się w obu kierunkach, nie tylko czytać po kolei
- automat może zapisywać litery na taśmie, zarówno nadpisywać istniejące jak i zapisywać miejsca dotychczas puste

Maszyny Turinga (deterministyczne)

- $\mathcal{A} = \langle Q, A, \Gamma, B, \delta, q_0, F \rangle$ gdzie:
  Q – skończony zbiór stanów,
  $q_0 \in Q$ – stan początkowy,
  $F \subseteq Q$ – zbiór stanów końcowych (akceptujących),
  A – alfabet wejściowy
  $\Gamma \setminus A$ – alfabet roboczy
  $B \in \Gamma \setminus A$ – symbol *blank* pustego miejsca na taśmie
  $\delta : Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{L, R, N\}$ funkcja częściowa określająca przejście stanów, nadpisywanie liter oraz ruch głowicy czytającej taśmę

Ruchy maszyny Turinga

- jeśli $\delta(q, \sigma) = (p, \tau, S)$ to mówimy, że (q, σ, p, τ, S) opisuje jeden ruch automatu
  w stanie q przeczytał literę σ i przeszedł w stan p, wpisał literę τ w miejscu gdzie stała σ oraz przesunął głowicę czytającą w prawo, lewo lub wcale (S)
- konfiguracja to trójka uqw, $q \in Q$, u jest słowem przed głowicą czytającą, pierwsza litera w jest wskazywana przez głowicę
- ruch (q, σ, p, τ, N) oznacza przejście $uq\sigma v \to up\tau v$
- ruch (q, σ, p, τ, R) oznacza przejście $uq\sigma v \to u\tau pv$
- ruch (q, σ, p, τ, L) oznacza przejście $u\mu q\sigma v \to up\mu\tau v$
- obliczenie – skończony lub nieskończony ciąg konfiguracji
- konfiguracja początkowa – $q_0 w$, $w \in A^*$ (na początku wpisane jest słowo z liter alfabetu wejściowego, potem mogą pojawiać się litery z alfabetu roboczego)

Język maszyny Turinga

- konfiguracja jest *akceptująca* jeśli jej stan jest akceptujący
- obliczenie jest akceptujące jeśli jest skończone i ostatnia konfiguracja jest akceptująca
- $L(\mathcal{A}) = \{w \in A^* \mid \mathcal{A} \text{ akceptuje } w\}$
- język L jest *rekurencyjnie przeliczalny (recursively enumerable)* jeśli $L = L(\mathcal{A})$ dla pewnej maszyny Turinga $\mathcal{A}$
- $w \in L(\mathcal{A})$ oznacza, że maszyna Turinga zatrzyma swoje obliczenia i ujawni pozytywny fakt
- $w \notin L(\mathcal{A})$
  - może być zauważone po zatrzymaniu się maszyny Turinga w stanie nieakceptującym
  - ale może się okazać, że obliczenia maszyny Turinga ze słowem wejściowym w są nieskończone

Języki rekurencyjne

- język L jest rekurencyjny/ rozstrzygalny/ obliczalny, jeśli
  - $L = L(\mathcal{A})$ dla pewnej maszyny Turinga
  - obliczenia tej maszyny są skończone dla każdego słowa $w \in A^*$
- tzn. podając słowo w na taśmie wejściowej czekamy na zatrzymanie się obliczeń, by rozstrzygnąć czy $w \in L(\mathcal{A})$ czy też nie
  - do czasu zatrzymania się maszyny musimy czekać na rozstrzygnięcie
- tw.: jeśli język L i jego dopełnienie są rekurencyjnie przeliczalne, to oba są rekurencyjne
  - dw.: rekurencyjna przeliczalność dopełnienia znaczy, że jeśli słowo nie należy do języka, to doczekamy się zatrzymania maszyny w stanie nieakceptującym
    czyli na pewno doczekamy się rozstrzygnięcia, nie ma obawy nieskończonego obliczenia

Rozstrzygalność

- problem logiczny można sprowadzić do pytania, czy pewne zdanie jest prawdziwe
- równoważnie, mamy zapisy zdań prawdziwych i pytamy, czy nasze zdanie jest prawdziwe
- równoważnie, mamy język (zbiór zdań), słowo i pytamy czy $w \in L$
- rozstrzygalność: maszyna zatrzyma się na każdym słowie i da odpowiedź tak/nie
- akceptowanie: maszyn zatrzyma się na słowach, które zaakceptuje, ale ma prawo nie kończyć obliczeń dla innych

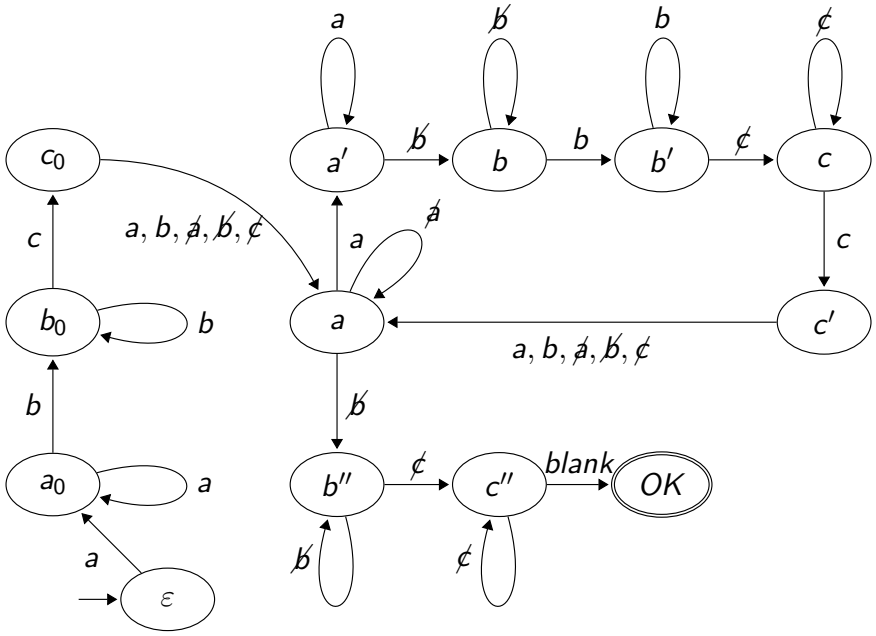
Przykład $\{a^n b^n c^n \mid n > 0\}$

- język $\{a^n b^n c^n \mid n > 0\}$ *nie* jest bezkontekstowy, tzn. nie jest rozpoznawalny przez automat ze stosem
- jest przekrojem dwóch języków bezkontekstowych, czyli jest rozpoznawalny przez automat z dwoma stosami
- jest rozpoznawalny przez maszynę Turinga

- w stanie początkowym czyta a, zamienia a na a przekreślone i przechodzi w stan szukania b
- t.j. mija a do czasu znalezienia b, które przekreśla i szuka c
- j.w., po znalezieniu zamienia na c przekreślone i wraca do początku taśmy
- błędem będzie nie znalezienie właściwej litery do przekreślenia, maszyna zatrzyma się bez akceptacji
- c.d. kolejny slajd

Przykład $\{a^n b^n c^n \mid n > 0\}$

- teraz szuka a, tj. mija a przekreślone, znajduje a, przekreśla i szuka b
- t.j. mija ew. pozostałe a, potem b przekreślone, znajduje b, przekreśla i szuka c
- po znalezieniu wraca do początku i powtarza
- być może nie znajdzie nowego a, czyli wszystkie a są już przekreślone
- po pierwszym b przekreślonym czyta jedynie litery przekreślone, najpierw b, potem c
- jeśli kolejny znak jest *blank*, to akceptuje, każdy inny znak jest nie akceptowany
- już wcześniej mógł być błąd, gdy np. liter b było więcej niż a i zostały nieprzekreślone



Maszyny Turinga wielościeżkowe

- automat z dwoma stosami może rozpoznawać więcej języków niż automat z jednym stosem
- pytanie, czy maszyna Turinga z dwoma ścieżkami ma większą moc rozpoznawania niż z jedną ścieżką
- odpowiedź: **nie**, maszyna w wieloma ścieżkami jest równoważna maszynie z jedną ścieżką
- dw: jeśli przyjąć, że wszystkie ścieżki korzystają z tej samej głowicy czytającej, to w zasadzie możemy uznać, że taśma wejściowa jest jedna – każda komórka jest zestawem komórek poszczególnych taśm
- bardziej skomplikowany jest przypadek, gdy taśm naprawdę jest więcej i każda ma swoją pozycję głowicy

Maszyny Turinga wielościeżkowe c.d.

- dane: maszyna z np. dwoma taśmami i niezależnymi głowicami
- dodajemy dwie nowe taśmy, które mają tylko jeden symbol, mianowicie pozycję głowicy 1. lub 2.
- otrzymujemy maszynę z czterema taśmami, przyjmujemy, że jedyna głowica domyślnie jest na początku
- jeden ruch maszyny z dwoma taśmami:
  - odszukanie pozycji taśmy 1. na dodatkowej taśmie, odczyt symbolu z tej taśmy, powrót
  - to samo dla taśmy 2.
  - decyzja co wpisać na obu taśmach, jak zmienić pozycje obu głowic i jak zmienić stan
  - odszukanie pozycji taśmy 1., wpis nowego symbolu, powrót
  - to samo dla taśmy 2.
  - a potem, na obu taśmach, odszukanie pozycji i ew. ich zmiana

Przykłady maszyn wielościeżkowych

- język $\{wcw \mid w \in (a+b)^*\}$
  maszyna przepisuje słowo w na drugą taśmę, gdy znajdzie separator c cofa głowicę drugiej taśmy i dalej porównuje oba napisy
- język $\{a^n \mid n$ jest potęgą $2\}$
  maszyna czyta kolejne a na pierwszej taśmie, a na drugiej ma zapis binarny n (w kolejności od najmniej znaczących bitów) i dodaje binarnie 1
  na końcu sprawdza, czy n jest potęgą 2
- maszyna Turinga z taśmą obustronnie nieskończoną
  drugą taśmą jest lewa część oryginalnej taśmy z odwróconym kierunkiem
  przejście głowicy czytającej na lewą stronę oznacza, że przechodzimy do drugiej taśmy

Maszyny Turinga a automaty ze stosem

- automat ze stosem można zaimplementować jako maszyna Turinga
  - na taśmie wejściowej niczego nie zapisujemy, ruch tylko w prawo
  - na drugiej taśmie zapisujemy zawartość stosu, przyda się symbol dna stosu
- maszynę Turinga można zaimplementować jako automat z dwoma stosami
  - na jednym stosie leży zawartość taśmy na prawo od głowicy, na drugim na lewo
  - w ogóle nie korzystamy z taśmy wejściowej dla automatu
  - ruch głowicy czytającej oznacza przenoszenie jednej komórki pomiędzy stosami

Obliczenia przez maszyny Turinga

- dotychczasowa definicja maszyny Turinga mówiła o problemie decyzyjnym, czy $w \in L(\mathcal{A})$?
- def.: funkcja $f : A^* \to \Sigma^*$ jest *obliczalna* jeśli istnieje (deterministyczna) maszyna Turinga $\mathcal{A}$ z dwiema taśmami, wejściową i wyjściową, taka, że
  - jeśli na wejściu znajduje się $w \in A^*$, to maszyna zatrzyma swoje działanie
  - i na wyjściu znajdzie się słowo $f(w)$

Obliczenie $n \cdot m$

- wejście: ciąg bitów przedstawiający n i m
- problem: jak odróżnić obie liczby
  - np. dodatkowa litera w alfabecie wejściowym (w teorii nie ma ograniczeń na liczbę liter alfabetu)
  - albo od razu dopuścić by były dwie odrębne taśmy z danymi wejściowymi
- wyjście: ciąg bitów przedstawiający iloczyn $n \cdot m$
- konstrukcja maszyny
  - przepisujemy m na taśmę pomocniczą
  - w każdym ruchu odejmujemy jedynkę od n i dodajemy m do zawartości taśmy wyjściowej
  - tak długo aż $n == 0$
- dodawanie zawartości dwóch liczb trzeba dokładniej opisać
  - lub posłużyć się inną maszyną, która to realizuje

Niedeterministyczne maszyny Turinga

- funkcja przejścia $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R, N\})$ jest niedeterministyczna, t.j. definiuje zbiór możliwych wyników
- konfiguracje układają się nie w ciąg obliczeń a w drzewo
- słowo jest zaakceptowane jeśli istnieje ścieżka do liścia z konfiguracją akceptującą
- zakładamy, że stan akceptujący nie dopuszcza dalszych obliczeń
- tw.: niedeterministyczne maszyny Turinga rozpoznają tę samą klasę języków co maszyny deterministyczne
- nie są znane żadne metody fizycznej realizacji maszyn niedeterministycznych
- w szczególności ew. powstanie komputerów kwantowych nie zrealizuje dowolnych maszyn niedeterministycznych

Przykład maszyny niedeterministycznej

- dana formuła logiki zdaniowej, tj. złożona ze zdań atomowych połączonych spójnikami logicznymi, np. negacji, implikacji, koniunkcji i dyzjunkcji
- problem: czy istnieje wartościowanie zdań atomowych takie, że formuła ma wartość *True* (spełnialność)
- albo czy ma wartość *True* dla każdego wartościowania (tautologia)
- maszyna niedeterministyczna rozstrzygająca spełnialność formuły
  - na drugą taśmę maszyna wpisuje wszystkie zdania atomowe występujące w formule
  - na trzecią taśmę maszyna wpisuje jakoś wartości *True* i *False*
  - następnie cofa się do początku pierwszej taśmy i zamienia zdania atomowe na *True* i *False* zgodnie z taśmami 2 i 3
  - jeszcze raz się cofa i oblicza wartość formuły zapisanej na pierwszej taśmie

Przykład maszyny deterministycznej

- ten sam problem może być rozwiązany przez maszynę deterministyczną poprzez wypróbowanie kolejnych możliwości
- tzn. na trzeciej taśmie maszyna wykonuje algorytm generowania wszystkich możliwych wartościowań zmiennych z drugiej taśmy
- dla każdego z wartościowań wykonuje wcześniej opisane operacje
- ta maszyna deterministyczna skończy działanie najdalej po 2^n testach, gdzie n jest liczbą zdań atomowych w formule, ograniczoną długością formuły

Determinizacja maszyny Turinga

- tw.: dla każdej niedeterministycznej maszyny Turinga M istnieje maszyna deterministyczna MD akceptująca ten sam język – $L(M) = L(MD)$
- dw.: zakładamy, że maszyna ma jedną taśmę, rozpatrujemy graf skierowany
  - wierzchołki: konfiguracje maszyny M, tj. trójki uqw, u, w słowa przed/po głowicy czytające, $q \in Q_M$ stan maszyny
  - krawędzie: wykonanie jednego kroku obliczeń
- problem: czy istnieje ścieżka od konfiguracji początkowej do jakiejś akceptującej
- NIE jest to dokładnie problem osiągalności w grafie, bo powyższy graf może być nieskończony
- ale można zapisywać i rozważać coraz większe fragmenty grafu w miarę rozwijania się obliczeń

Determinizacja c.d.

- istnieje stała d ograniczająca liczbę możliwości wyboru w maszynie niedeterministycznej, $d = \max(|\delta(q, a)|, q \in Q, a \in A) \leq |Q| \cdot |\Gamma| \cdot 3$
- na drugiej taśmie generuje się coraz dłuższe ciągi adresujące drzewo obliczeń M o stopniu rozgałęzienia $\leq d$ (przeszukiwanie wszerz)
- na trzeciej taśmie wykonuje się obliczenia odpowiadające ciągowi zapisanemu na drugiej taśmie
- jeśli maszyna M akceptuje dane wejściowe, to istnieje obliczenie, po którym się zatrzyma i ogłosi akceptację
- czyli maszyna deterministyczna dojdzie do momentu, gdy znajdzie to obliczenie i zaakceptuje dane wejściowe
- przykład ze spełnialnością formuły logiki zdaniowej:
  - maszyna deterministyczna przejrzy 2^n możliwych wartościowań n zdań atomowych zawartych w danej formule

Obliczalność – Teza Church'a-Turinga

- wszystkie pojęcia obliczalności są równoważne
- nie jest to twierdzenie, bo właśnie nie mamy definicji obliczalności, twierdzimy jedynie, że definicja poprzez maszyny Turinga jest słuszna
  - nie znamy szerszej klasy języków, które można by uznać za rozpoznawalne działaniami charakterystycznymi dla komputerów
  - definicja maszyny Turinga jest ekstremalnie minimalistyczna i trudno się nie zgodzić, że takie działania są dopuszczalne w obliczeniach
  - wszystkie definicje maszyn: deterministycznych lub nie, jedno- lub wielościeżkowych, z niektórymi taśmami dopuszczającymi tylko odczyt, tylko ruch w prawo, prowadzą do tej samej klasy języków