# Obliczenia inspirowane Naturą

## Wykład 01 – Modele obliczeń

Jarosław Miszczak

IITiS PAN Gliwice

05/10/2016

# Obliczanie i maszyna Turinga

## Co to znaczy obliczać?

- Zadaniem informatyki jest określienie możliwości obliczeniowych maszyn (komputerów).
- Aby formalnie badać własności komputerów potrzebne są model które:
  - pozwalają uchwycić możliwości obliczeniowe realnych maszyn;
  - dają możliwość dowodzenia twierdzeń na temat ich własności.

# Obliczanie i maszyna Turinga
## Deterministyczna maszyna Turinga

Maszyna Turinga dysponuje bardzo skromnymi środkami obliczeniowymi...

- Wszystkie operacje maszyny wykonywane są na taśmie, na której znajdują się symbole z pewnego skończonego zbioru $\Sigma$.
- Maszyna w jednym kroku może pobrać za pomocą czytnika jeden symbol $x \in \Sigma$ z taśmy, wykonać (lub nie) ruch nad taśmą i zmienić stan wewnętrzny.

...lecz pomimo to potrafi wykonać wiele skomplikowanych zadań.

# Obliczanie i maszyna Turinga

## Deterministyczna maszyna Turinga

### Definicja

*Deterministyczną maszyną Turinga nazywamy czwórkę uporządkowaną* $M = (K, \Sigma, \delta, q_0)$ *złożoną z:*

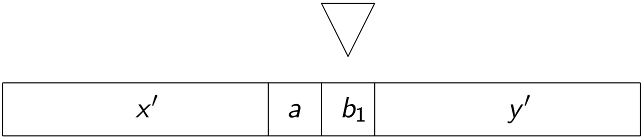
- *skończonego zbioru* $K$ *stanów zawierającego stan początkowy* $q_0 \in K$
- *skończonego zbioru* $\Sigma$ *symboli (alfabetu) zawierającego symbol pusty* $\sqcup$ *i symbol końcowy* $\triangleleft$*.*
- *Funkcji*
  $\delta\colon K \times \Sigma \to (K \cup \{\text{„stop”}, \text{„tak”}, \text{„nie”}\}) \times \Sigma \times \{\leftarrow, -, \rightarrow\}$

*gdzie* $\leftarrow, \rightarrow, -$ *oznaczają odpowiednio ruch w prawo, w lewo i pozostanie na miejscu.*

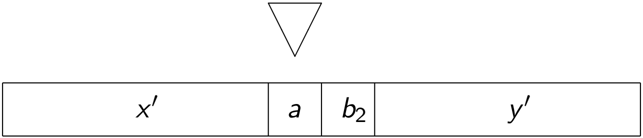
# Obliczanie i maszyna Turinga

## Przykład

Przykład: **funkcja** przejścia $\delta(q_i, b_1) = (q_2, b_2, -1)$




(a) Konfiguracja $(q_i, x'a, b_1 y')$



# Obliczanie i maszyna Turinga

## Rozpoznawanie i rozstrzyganie

Maszyna Turinga pozwala operować na słowach – możemy obliczać wartość funkcji lub określać czy dane słowo należy do jakiegoś języka.

- Językiem nazywamy dowoly podzbiór ciągów symboli maszyny Turinga, $L \subset (\Sigma - \{\sqcup\})^*$.
- Elementy języka to słowa.
- Dowolny problem można zakodować w postaci języka – wybór sposobu kodowania nie ma większego znaczenia (o ile nie kodujemy unarnie).

# Obliczanie i maszyna Turinga
## Rozpoznawanie i rozstrzyganie

Maszyny Turinga możemy traktować jako algorytmy.

- Ustalmy sobie język $L$.
- Niech $M$ będzie maszyną Turinga, taką, że dla $X \in L$, $M(x) = \mathrm{tak}$, a jeżeli $x \notin L$, $M(x) = \mathrm{nie}$.
- Wówczas mówimy, że $M$ **rozstrzyga** $L$.
- Jeżeli $M$ zawsze daje 'tak' na $x \in L$, ale może nie zatrzymać się na $x \notin L$, to mówimy, że $M$ **rozpoznaje** $L$.

# Obliczanie i maszyna Turinga
## Rozpoznawanie i rozstrzyganie

Warto zwrócić uwagę na:

# Obliczanie i maszyna Turinga
## Rozpoznawanie i rozstrzyganie

Warto zwrócić uwagę na:

- asymetrię w określeniu rozstrzygania i rozpoznawania;

# Obliczanie i maszyna Turinga

## Rozpoznawanie i rozstrzyganie

Warto zwrócić uwagę na:

- asymetrię w określeniu rozstrzygania i rozpoznawania;
- wątpliwą użyteczność rozpoznawania – nigdy nie wiadomo czy czekamy wystarczająco długo na zatrzymanie się maszyny.

# Obliczanie i maszyna Turinga

## Uniwersalna maszyna Turinga

- Najprostszym rozszerzeniem tego modelu maszyny Turinga jest dodanie możliwości operowania na kilku ciągach danych – otrzymujemy wówczas maszynę Turinga z wieloma taśmami, którą trochę łatwiej się *programuje*.
- Ale taka modyfikacja nie pozwala na zwiększenie mocy obliczeniowej modelu – każda maszyna z wieloma ciągami może być symulowana w czasie $O(f(x)^2)$ na maszynie z jednym ciągiem.

# Obliczanie i maszyna Turinga

## Uniwersalna maszyna Turinga

- Uniwersalna maszyna Turinga (UMT) może zasymulować dowolną maszynę Turinga.
- Przyjmuje ona na wejściu numer maszyny Turinga oraz jej ciąg wejściowy.
- w przypadku UMT zakładamy, że stanami i symbolami są liczby całkowite.
- Działanie UMT opisuje równość $U(M;x) = M(x)$.

# Niedeterministyczna maszyna Turinga

## Definicja

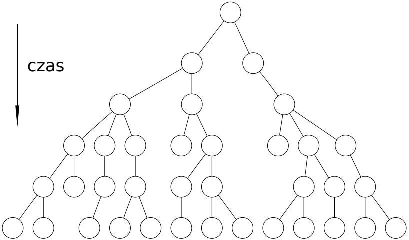
*Niedeterministyczna* maszyna Turinga nie jest *rozsądną* modyfikacją maszyny Turinga, ale jest ważna ze względy na określenie problemów trudnych do rozwiązania.

### Definicja

*Niedeterministyczną maszyną Turinga nazywamy czwórkę uporządkowaną $M = (K, \Sigma, \delta, q_0)$ złożoną z:*

- *skończonego zbioru $K$ stanów zawierającego stan początkowy $q_0 \in K$*
- *skończonego zbioru $\Sigma$ symboli (alfabetu) zawierającego symbol pusty $\sqcup$ i symbol końcowy $\triangleleft$.*
- *Relacji $\delta \subset K \times \Sigma \times (K \cup \{„stop”, „tak”, „nie”\}) \times \Sigma \times \{\leftarrow, -, \rightarrow\}$*

# Niedeterministyczna maszyna Turinga
## Zasada działania




C.H. Papadimitriou, *Złożoność obliczeniowa*, WNT, 2007.

# Niedeterministyczna maszyna Turinga
## Zasada działania

Maszyny niedeterministyczne zyskują swoją siłe dzięki niewielkim wymaganiom co do rozwiązywania problemu.

Mówimy, że maszyna $N$ rozstrzyga język $L$, jeżeli dla każdego $x \in L$ maszyna zatrzymuje się w stanie 'tak'.

- Słowo jest akceptowane jeżeli istnieje pewna ścieżka obliczeń która prowadzi do akceptacji.
- Słowo jest akceptowane nawet jeżeli większość wyborów obliczeń prowadzi do odrzucenia.
- Słowo jest odrzucane gdzy żadne obliczenie nie prowadzi do akceptacji.

# Niedeterministyczna maszyna Turinga
## Zasada działania

Jak to interpretować?

1. Maszyna wybiera jedną z dróg a nastepnie zachowuje się tak jak maszyna deterministyczna.
2. Wiele maszyn wykonuje program równocześnie.

Ale każda z tych interpretacji wskazuje że niedeteministyczna maszyna Turinga nie jest rozsądnym modelem obliczeń.

# Klasy złożoności

W jaki sposób możemy zmierzyć koszt wykonania algorytmu?
Możemy podać

- liczbę kroków, które musi wykonać maszyna Turinga dla problemu;
- ile taśmy zużywa maszyna Turinga.

# Klasy złożoności

- Dla zadanej funkcji $f(n) : \mathbf{N} \mapsto \mathbf{N}$ określamy klasę złożoności **TIME**$(f(n))$ jako zbiór języków dla których istnieje maszyna Turinga z wieloma ciągami rozstrzygająca język w czasie $f(n)$.

# Klasy złożoności

- **P** (ang. *polynomial time*) – wszystkie problemy dla których istnieje deterministyczna maszyna Turinga kończąca swoje działanie w czasie wielomianowym względem wielkości problemu.
- **NP** (ang. *nondeterministic polynomial time*) – wszystkie problemy dla których istnieje niedeterministyczna maszyna Turinga kończąca swoje działanie w stanie „tak” w czasie wielomianowym względem wielkości problemu.

# Klasy złożoności

- Maszyny nideterministyczne mogą z łatwością przeszukiwać wykładniczą przestrzeń rozwiązań.
- Niedeterminizm pozwala rozróżnić między problemami łatwymi i trudnymi.

**Hipoteza**

**P$\neq$NP**

# Automaty skończone
## Znaczenie automatów

- Nie zawsze potrzebujemy pełnej mocy maszyny Turinga do rozwiązania problemu.
- Bardzo użytecznym modelem są *automaty skończone*

# Automaty skończone

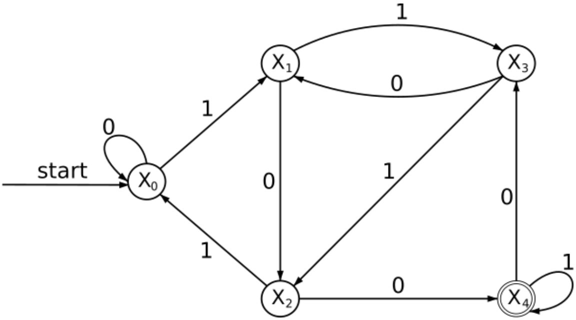
## Definicja

Deterministyczne automat skończony to piątka $(\Sigma, K, F, \delta, q_0)$, gdzie

- $\Sigma$ to zbiór symboli,
- $K$ to zbiór stanów,
- $A$ to zbiór stanów akceptujących,
- $q_0 \in K$ to stan początkowy,
- $\delta$ to funkcja przejścia.

# Automaty skończone

## Przykład



- Analogicznie jak w przypadku maszyny Turinga możliwe jest wprowadzenie niedeterministycznej wersji automatu skończonego.
- Jednak w tym wypadku okazuje się, że niedeterminizm nie zwiększ mocy obliczeniowej modelu.

# Inne warianty maszyn Turinga

Maszyna Turinga jest w dużej mierze określona przez określenie relacji przejścia. Ważne warianty to:

- **obliczenia odwracalne**, które pozwalają na:
  - niski pobór energii;
  - możliwość cofnięcia się do stanu przed wystąpieniem błędu (*np.* przy programowaniu robotów);
- **obliczenia kwantowe**, które pozwalają na
  - wykorzystanie superpozycji (czyli równoczesnego wykonywania operacji na wielu danych);
  - fizyczne operowanie na nanostrukturach (pojedyńczych atomach lub dobrze zdefiniowanych grupach atomów);
  - wykorzystanie pojedyńczych fotonów w kryptografii.

# Inne warianty maszyn Turinga
## Obliczenia odwracalne

Obliczenia odwracalne pozwalają na:

- niski pobór energii;
- możliwość cofnięcia się do stanu przed wystąpieniem błędu (*np.* przy programowaniu robotów, przy odtwarzaniu błędów w tranzakajach w bazach danych);

# Inne warianty maszyn Turinga

## Obliczenia odwracalne

Przy rozważaniu odwracalności możemy wyróżnić

- odwracalność logiczną – z wyniku możemy odtworzyć argumenty;
- odwracalność fizyczną – możliwe jest przywrócenie stanu układu na którym wykonujemy obliczenia.

# Inne warianty maszyn Turinga
## Obliczenia kwantowe

Obliczenia kwantowe pozwalają na

- wykorzystanie superpozycji (czyli równoczesnego wykonywania operacji na wielu danych);
- fizyczne operowanie na nanostrukturach (pojedyńczych atomach lub dobrze zdefiniowanych grupach atomów);
- wykorzystanie pojedyńczych fotonów w kryptografii.

# Nierozstrzygalność
## Klasyfikacja języków

Ograniczając możliwości maszyny Turinga możemy wprowadzić hierarchię języków.

- języki **rekurencyjnie przeliczalne** – języki dla których istnieje akceptująca je maszyna Turinga (nie musi się zatrzymywać!);
- języki **rekurencyjne** – języki dla których istnieje rozpoznająca je maszyna Turinga;

**Stwierdzenie**

Jeżeli język jest rekurencyjny, to jest rekurencyjnie przeliczalny.

# Nierozstrzygalność
## Klasyfikacja języków

Możemy też rozpatrywać języki rozpoznawane w innych modelach obliczeń.

- języki **bezkontekstowe** – języki dla których istnieje rozpoznający je (niedeterministyczny) automat ze stosem ($\equiv$ gramatyki bezkontekstowe $\equiv$ Backus-Naur Form);
- języki **regularne** – języki dla których istnieje rozpoznający jest automat skończony ($\equiv$ wyrażenia regularne – sed, gawk, pcre);

J.E.F. Friedl, *Wyrażenia regularne*, Helion, Gliwice, 2001.

# Nierozstrzygalność

## Problem stopu

- Okazuje się że istnieją problemy dla których nie można podać algorytmu.
- Najważniejszym przykładem takiego problemu jest **problem stopu**.

# Nierozstrzygalność
## Problem stopu

Do opisania problemu stopu potrzeba jest nam uniwersalna maszyna Turinga.

**Problem stopu**

$H = \{M; x : M(x) \neq \nearrow\}$

Język $H$ zawiera wszystkie słowa kodujące maszyny Turinga wraz z ze słowami wejściowymi, takimi, że maszyna się na nich zatrzymuje.

# Nierozstrzygalność
## Problem stopu

**Stwierdzenie**

Język $H$ jest rekurencyjnie przeliczalny.

# Nierozstrzygalność
## Problem stopu

**Stwierdzenie**

Język $H$ jest nie jest rekurencyjny.