

# Teoretyczne modele obliczeń

## automat skończony

- inaczej „finite state machine”

- definicja formalna:

jest to piątka  $(A, Q, q_0, F, d)$ , gdzie

„A” jest skończonym zbiorem symboli wejściowych

„Q” jest skończonym zbiorem stanów

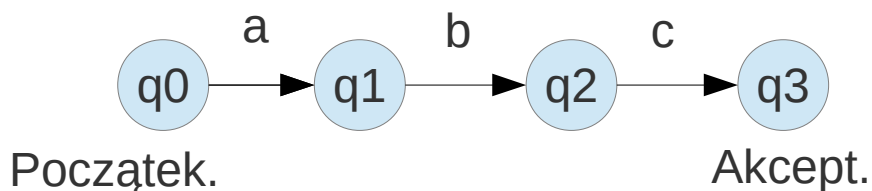
„ $q_0$ ” jest wyróżnionym stanem początkowym należącym do Q

„F” jest zbiorem stanów akceptujących (końcowych), będącym podzbiorem Q

„d” jest funkcją przejścia (tranzycją), częściowo określoną (!),

przypisującą parze  $(q, a)$  nowy stan „p”,

w którym znajdzie się automat po przeczytaniu symbolu „a” w stanie „q”



$A = \{a, b, c\}$

$Q = \{q_0, q_1, q_2, q_3\}$

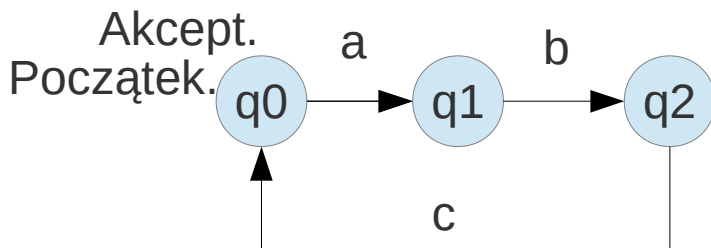
$F = \{q_3\}$

$d(q_0, a) = q_1$

$d(q_1, b) = q_2$

...

Akceptuje tylko napis: abc



Jakie słowa akceptuje ten automat ?  
abc, abcabc, ...

# Teoretyczne modele obliczeń

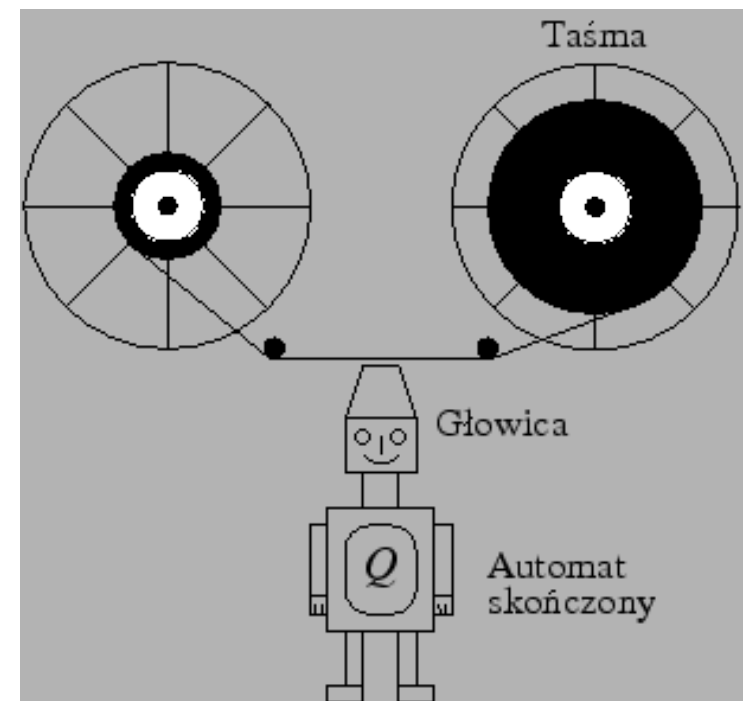
## automat skończony

- „język” to zbiór „słów” (ciągów symboli);
- automat skończony **akceptuje język L**
  - jeśli kończy w stanie akceptującym dla każdego słowa z L
  - natomiast dla słowa spoza L kończy w stanie NIE-akceptującym
- automaty skończone **deterministyczne** i **niedeterministyczne**
  - w niedetermin z jednego stanu (wierz) może wychodzić 0,1, $\geq 1$  strzałka
  - są możliwe różne sekwencje stanów przy przechodzeniu przez słowo na we
- języki akceptowane przez automat skończony deterministyczny można opisać przy pomocy tzw „wyrażeń regularnych”  $abc$ ,  $(abc)^*$ , itp.
- są różne ulepszenia automatu skończonego...

- czy współczesne komputery to automaty skończone ?

*Odp: nie !!!*

trzeba jeszcze dodać odpowiednik „pamięci”  
w maszynie Turinga jest to „taśma z głowicą”



# Teoretyczne modele obliczeń

## maszyna Turinga

- formalna def **maszyny Turinga** (MT) *trochę uproszczona...*

To szóstka  $(Q, \delta, A, q_0, \_, F)$ , gdzie:

„Q” – skończony zbiór stanów

$q_0 \in Q$  – stan początkowy,

„F” – zbiór stanów końcowych / akceptujących

„A” – skończony zbiór symboli na taśmie

„\_” – symbol pusty

„ $\delta$ ” – funkcja tranzycji opisana następująco:

$$\delta : Q \times A \rightarrow Q \times A \times \{L, R\}$$

jest to funkcja pobierająca aktualny stan maszyny oraz symbol wejściowy, a dającą w wyniku symbol, jaki ma zostać zapisany na taśmie (pod głowicą), kolejny stan maszyny oraz przesunięcie głowicy (L lub R).

uwaga !!:  $\delta$  jest częściowo określona na  $Q \times A$

- elementy składowe się MT:

1. nieskończona taśma, są na niej symbole z A oraz „\_”

2. zmodyfikowany automat skończony

$d(q,a)$  podawało nowy stan w automacie skończonym...

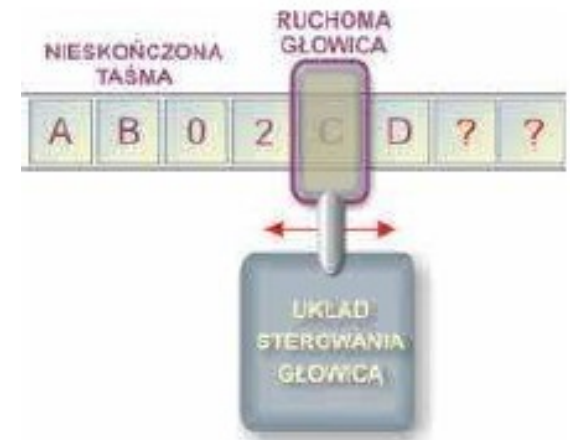
$\delta(q,a)$  podaje nowy stan, co zapisać na taśmie, ruch głowicy L/R

- jak działą MT?

na taśmie umieszcza się dane wejściowe...

MT zaczyna działać i **zatrzymuje się** gdy brak wartości  $\delta(q,a)$

odczytujemy z taśmy wynik lub zadowala nas „czy stan końcowy jest w F”



# Teoretyczne modele obliczeń

## maszyna Turinga

- MT może obliczać funkcję o wartościach TAK/NIE jak i o wartościach będących ciągiem symboli z  $A$ ; argument tej funkcji umieszcza się na taśmie jako ciąg symboli z  $A$

- tekstowy zapis działania MT („egzekucji”):

1 q1 + 1 |- 1 1 q2 1 |- 1 1 1 q2

gdzie „1”, „+” to symbole na taśmie, q1 i q2 to stany, głowica jest na symbolu za „q?”

|-\* oznacza, że wykonano wiele kroków...

- graficzny opis MT:

- def funkcji **obliczalnej** w sensie Turinga:

funkcja „f” z dziedziną „D”

jest obliczalna w sensie Turinga

jeśli istnieje MT taka, że dla każdego  $w \in D$

$q_0 \xrightarrow{|-*} q_n f(w)$

tzn, że po zakończeniu działania

MT jest w stanie  $q_n$ , a na stosie

na prawo od głowicy jest ciąg  $f(w)$

// argument i wynik  $f()$  to ciągi symboli z  $A$

- wszystko co obliczają dzisiejsze komputery

jest „obliczalne w sensie Turinga” = teza Churcha Turinga

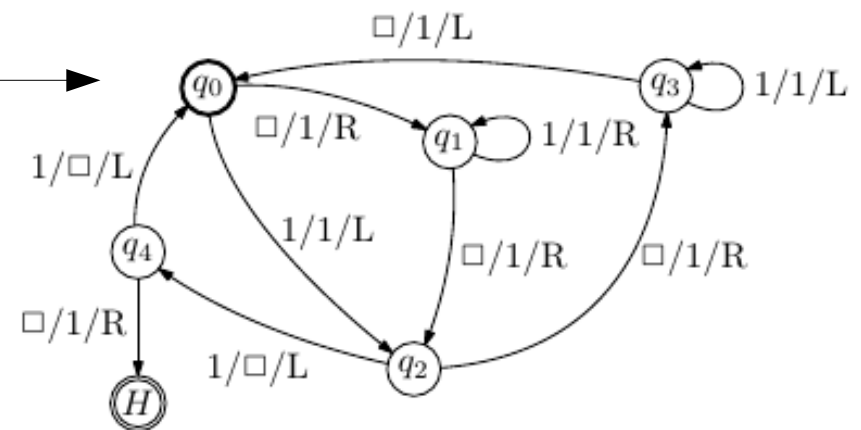


Figure 5: A candidate for a 5-state Busy Beaver

oznaczenia na łukach:

stary symbol/ nowy / L lub R

# Teoretyczne modele obliczeń

## maszyna Turinga

### - uniwersalna maszyna Turinga

1. można opisać MT w postaci ciągu symboli i umieścić ten opis na taśmie
  2. MT może symulować działanie MT na wejściu
- przypomina to współczesne komputery które przechowują program w pamięci!!!

### - istnieją różne uogólnienia MT, z wieloma taśmami, niedeterministyczne, ...

### - złożoność obliczeniowa MT:

jest bardzo prosta do zdefiniowania!! (to główna zaleta MT)

**złożoność czasowa** to liczba tranzycji

w najgorszym wypadku (dla *we* o danym rozmiarze)

**złożoność pamięciowa** to liczba użytych komórek taśmy

w najgorszym wypadku (dla *we* o danym rozmiarze)

Co to jest „rozmiar *we*” w MT ? Odp liczba symboli na taśmie

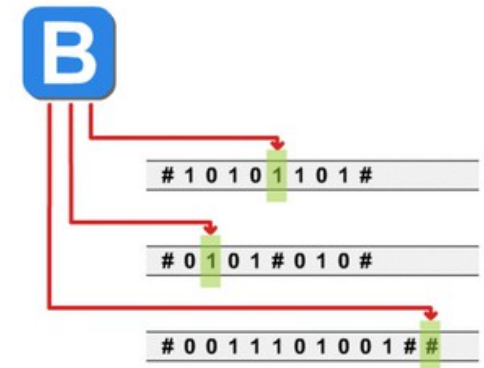
### - **problem stopu** (halting problem): czy MT się zatrzyma dla konkretnych danych wejściowych

**Tw:** problem czy dana MT (lub program) zatrzyma się

dostając na wejściu „1” jest *nierozstrzygalny* (= funkcja tak/nie jest nieobliczalna)

tzn. nie istnieje MT, która otrzymując opis „dowolnej MT” i dowolnego „wejścia 1” dawałaby odpowiedź czy maszyna się zatrzyma...

*Dowód jest zadziwiająco prosty...*



# Teoretyczne modele obliczeń

## maszyna Turinga

- eksperymenty z MT:

symulator o nazwie „Alana”...

przykładowy program dla symulatora:

# początkowy stan taśmy:

1 2 2 1 3 4 4 1

# "obliczenia":

(q0 1 q0 2 R)

(q0 2 q0 3 R)

(q0 3 q0 4 R)

(q0 4 q0 5 R)

# cofanie głowicy:

(q0 \_ q1 L)

(q1 2 L)

(q1 3 L)

(q1 4 L)

(q1 5 L)

# stan początkowy

[q0]

Def funkcji  $\delta$

Uwaga: ten symulator stosuje pewne skróty przy def  $\delta$ :

$(q1 A q1 A L) = (q1 A q1 L) = (q1 A L)$

$(q1 A q2 A L) = (q1 A q2 L)$

- funkcja „nieobliczalna”: **pracowity bóbr** (busy beaver)  
MT ma  $n$  -stanów, pustą taśmę, skończony zbiór symboli;  
 $\delta$  powinno być takie aby MT zapisało na taśmie  
możliwie dużą liczbę „1” oraz się zatrzymało.  
 $B(n)$  to maksymalna liczba 1 jaka da się uzyskać...  
Funkcja ta jest nieobliczalna w sensie Turinga !!!

# Teoretyczne modele obliczeń

## maszyna RAM

- maszyna RAM to model powstały już w czasach komputerów (przypomina je !!)

### MASZYNA RAM

Random Access Machine (RAM) składa się z:

- Taśmy wejściowej, z której maszyna wczytuje dane wejściowe;
- Taśmy wyjściowej, na której maszyna zapisuje wyniki działania;
- nieskończenie wiele rejestrów  $R_0, R_1, \dots$ , z których każdy może przechowywać liczbę całkowitą;
- listy instrukcji (programu)  $\Pi = (\pi_1, \dots, \pi_m)$ .

Przez  $r_i$  oznaczamy wartość przechowywaną w rejestrze  $R_i$ .

READ, WRITE

LOAD, STORE

- program składa się z rozkazów (przypominających assembler współczesnych procesorów)
- istnieje rejestr „licznik rozkazów” który wskazuje bieżący rozkaz
- rejestry mogą pamiętać dowolnie dużą liczbę
- liczba rejestrów jest nieskończona (ale program używa skończonej ich liczby)
- rozmiar we maszynie RAM: liczba bitów na taśmie wejściowej  
(w zasadzie na we są liczby całkowite, ale zapisane na ograniczonej liczbie bitów)

# Teoretyczne modele obliczeń maszyna RAM

## INSTRUKCJE W MODELU RAM

W skład ciągu instrukcji  $\Pi$  mogą wchodzić następujące operacje:

LOAD <b>op</b>	READ <b>op</b>	STORE <b>op</b>	WRITE <b>op</b>
ADD <b>op</b>	SUB <b>op</b>	MULT <b>op</b>	DIV <b>op</b>
JUMP <b>et</b>	JGTZ <b>et</b>	JZERO <b>et</b>	HALT

**op** może przyjmować postać:

- liczby całkowitej – ADD 3;
- wartości rejestru – MULT  $r_3$ ;
- wskaźnika do wartości rejestru  $r_{r_i}$  – STORE  $r_{r_7}$ .

**et** etykieta (numer) instrukcji, np. JZERO 7, JGTZ **początek**.

GTZ=greater  
than zero

## AKUMULATOR I LICZNIK

Rejestr  $r_0$  jest nazywany **akumulatorem** i ma specjalne znaczenie. Służy jako przestrzeń robocza do przechowywania danych dla bieżącej operacji. Kolejność wykonywania instrukcji jest kontrolowana przez **licznik programu**  $\kappa$ , który przechowuje numer aktualnie przetwarzanej instrukcji.

Dokładniejszy opis rozkazów maszyny RAM jest [tutaj](#)



# Teoretyczne modele obliczeń

## maszyna RAM

- rola akumulatora  $r_0$ :  $\text{ADD } r_2 \Leftrightarrow r_0 := r_0 + r_2$
- przykład programu dla maszyny RAM:

Etykieta	Instrukcja	Działanie
	READ $r_1$	Wczytanie $n$ do $r_1$
	LOAD $r_1$	Wczytanie $n$ do akumulatora
	JGTZ pos	if ( $r_1 = 0$ )
	WRITE 0	then return(0)
	JUMP endif	Skok na koniec programu
pos:	LOAD $r_1$	else - $r_1 = n$ do akumulatora
	STORE $r_2$	$r_2 := r_1$
	SUB 1	$r_0 := r_0 - 1$
	STORE $r_3$	$r_3 := r_1 - 1$
while:	LOAD $r_3$	$r_3$ do akumulatora
	JGTZ continue	while $r_3 > 0$
	JUMP endwhile	gdy $r_3 = 0$ - zakończ
continue:	LOAD $r_2$	$r_2$ do akumulatora
	MULT $r_1$	$r_0 := r_2 * r_1$
	STORE $r_2$	$r_2 := r_2 * r_1$
	LOAD $r_3$	$r_0 := r_2$
	SUB 1	$r_0 := r_0 - 1$
	STORE $r_3$	$r_3 := r_3 - 1$
endwhile:	WRITE $r_2$	return( $r_2$ )
endif:	HALT	Koniec programu

$r_1 > 0$   
( $r_0 > 0$ )

# Teoretyczne modele obliczeń

## maszyna RAM

- **czas działania** maszyny RAM  
jest to bardziej skomplikowane niż w maszynie Turinga...  
rozkazy mają różny czas działania w zależności od ich typu:
  - \* arytmetyczne: czas to liczba bitów operandów
  - \* adresowanie pośrednie: liczba bitów adresu
- **złożoność czasowa** masz RAM to czas działania jako funkcja rozmiaru we  
rozmiar we to liczba bitów na taśmie wejściowej
- związek maszyn RAM i Turinga:  
obliczalność funkcji - taka sama !!  
złożoność czasowa: wielomianowa równoważność  $f(n) \leq O(f(n)^{\text{const}})$   
złożoność pamięciowa: liniowa równoważność

### SYMULACJA TM W MODELU RAM

**Twierdzenie:** Jeżeli zadanie rozpoznania języka  $L$  przez deterministyczną maszynę Turinga  $M$  ma złożoność czasową  $f(n)$  ( $n$  – rozmiar danych wejściowych) to istnieje program RAM, który oblicza funkcję charakterystyczną  $\chi_L$  języka  $L$  w czasie  $O(f(n))$ .

### SYMULACJA PROGRAMU RAM W MODELU TM

**Twierdzenie:** Jeżeli program  $\Pi$  maszyny RAM wylicza funkcję całkowitoliczbową  $\phi$  w czasie  $f(n)$  to istnieje deterministyczna maszyna Turinga  $M$  o siedmiu taśmach taka, że  $M$  oblicza  $\phi$  w czasie  $O(f(n)^3)$ .

# Teoretyczne modele obliczeń

## maszyna RAM

- związek maszyny RAM ze współczesnymi komputerami:  
„rejstry maszyny RAM” = „rejstry procesora i pamięć operacyjna”
- prosty komputer przypominający maszynę RAM, programowany w asm:  
porównaj rejestry Z80 (A, B, BC, HL, itp) + 48Kb ram z rejestrami maszyny RAM...
- maszyna RAM a współczesne języki programowania:

Python, JavaScript, Tcl

C, C++, Pascal

Asembler (różne arch procesorów)

procesor, pamięć, i/o

Jakby (uniwersalna) maszyna RAM...  
równoważna z maszyną Turinga  
*Wniosek:* w pythonie nie można obliczyć  
więcej niż na maszynie Turinga...