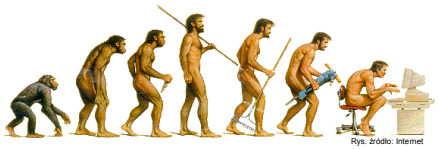
# Czy komputer może myśleć?

Jacek.Szczytko@fuw.edu.pl

Wydział Fizyki Uniwersytetu Warszawskiego

a. Tw. Gödla kontra Matrix
b. Moim zdaniem



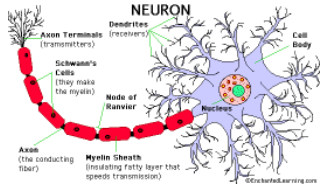
Rys. źródło: Internet

# Mózg a komputer



W jaki sposób policzyć ilość operacji logicznych w mózgu?

"When will computer hardware match the human brain? "
Hans Moravec, Journal of Evolution and Technology 1998. Vol. 1
http://www.frc.ri.cmu.edu/~hpm/




**Siatkówka oka:**

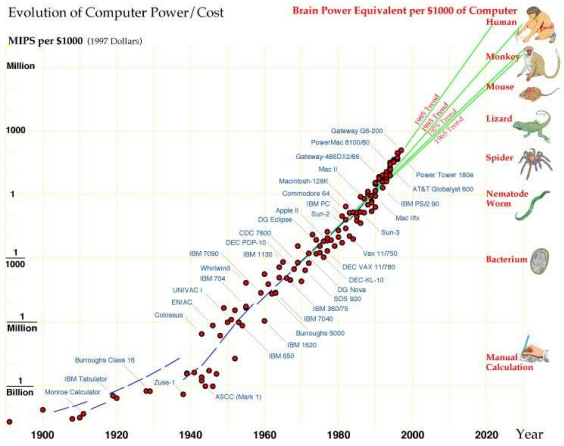
powierzchnia: 1cm$^2$, grubość: 1mm,
$10^8$ neuronów, 10 obrazów/s (rozdzielczość $10^6$, 16 mln. kolorów)
program komputerowy rozpoznający kształt, kolor, ruch – 1000 MIPS
ok. $10^6$ - $10^7$ MIPS (Million computer Instructions Per Second)

**Mózg**

objętość: 1500 cm$^3$
$10^{11}$ neuronów (ale aż $10^{15}$ połączeń)
ok.. $10^9$ - $10^{10}$ MIPS (a być może więcej)

# Mózg a komputer






# Mózg a komputer



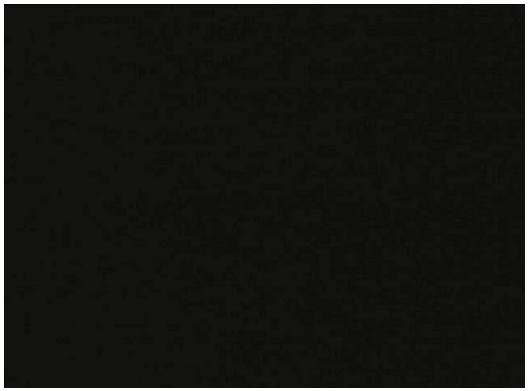
**Czy w ogóle można tak porównywać?**

**Hardware**
Fizyka

**Software**
Matematyka

# Czy komputer może myśleć?

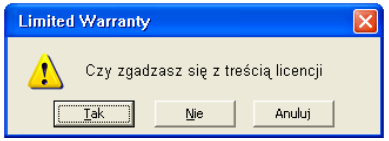
Jacek.Szczytko@fuw.edu.pl



# GPL, Limited Warranty etc.

Jacek.Szczytko@fuw.edu.pl

1. Opinia prezentowana w niniejszym wykładzie jest wyłączną opinią Jacka Szczytko ®© 2006 i nie należy traktować jej, jako jakąkolwiek sugestię, zalecenie, rekomendację lub wskazówkę o jakimkolwiek charakterze do zmiany swojego zdania.
2. Jacek Szczytko ®© 2006 zastrzega sobie prawo do wprowadzania zmian do opinii wspomnianej w par. 1 bez obowiązku zawiadomienia
3. Niniejszym Jacek Szczytko ®© 2006 wyklucza wszelką swoją odpowiedzialność, jakiejkolwiek natury, za działania lub zaniechania działań ze strony innych słuchaczy związane lub oparte na opinii przedstawionej niniejszym powyżej w par 1. lub cokolwiek w związku z czymkolwiek, ani żadnej rzeczy która jego jest.

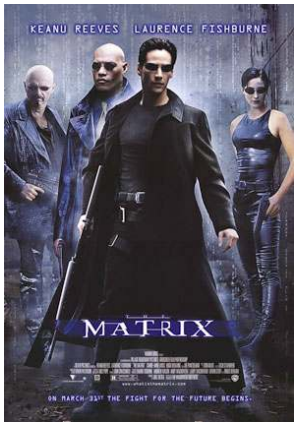



# Czy komputer może myśleć?

Jacek.Szczytko@fuw.edu.pl

- Co to znaczy myśleć?
- Co to jest świadomość?
- Czy maszyna może myśleć tak jak człowiek?
  - albo dzięki algorytmowi
  - albo symulując układ fizyczny

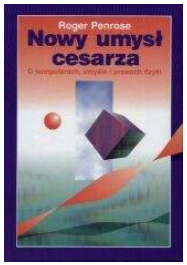
*Dialog z przyroda musi byc prowadzony w jezyku matematyki, w przeciwnym razie przyroda nie odpowiada na nasze pytania.*
**Michał Heller**




# Czy komputer może myśleć?

Jacek.Szczytko@fuw.edu.pl

- Więcej na ten temat: Roger Penrose: „Nowy umysł cesarza – o komputerach, umyśle i prawach fizyki” PWN 1995



# Czy komputer może myśleć?

Jacek.Szczytko@fuw.edu.pl

- Jak myśli komputer, czyli maszyna Turinga.


**Pewniki**:

1. Podstawą działania komputera są operacje LOGICZNE prawda-fałsz (1-0)
2. Maszyna posiada algorytm-program* (zamienia dane wejściowe na wyjściowe) i pamięć
3. Program, to pewien zbiór operacji logicznych (czyli ZDANIE LOGICZNE)

**Wnioski**:

•Maszyna może tylko wykonywać operacje, które dadzą się ZAPISAĆ w języku logiki matematycznej.

•Wykonując pewien program maszyna może się zatrzymać lub nie.

Operacje, które maszyna może wykonać zatrzymując się noszą nazwę OBLICZALNYCH (ang. computability).

*Program = tzw. stany wewnętrzne

# Czy komputer może myśleć?

Jacek.Szczytko@fuw.edu.pl

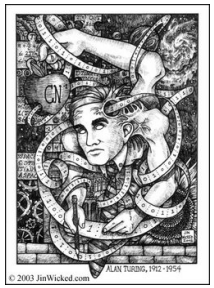
- Jak myśli komputer, czyli maszyna Turinga.

**Jak taka maszyna wygląda?**

| ... | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

dane

| Numer instrukcji (stan) | Dane | Nowy stan / 0-1 / R-L-STOP |
|---|---|---|
| 1 | 1 | 3 / 0 / R |
| | 0 | 4 / 0 / L |
| 2 | 1 | 9 / 0 / R |
| | 0 | 7 / 1 / STOP |
| 3 | 1 | 4 / 1 / L |
| | 0 | 2 / 0 / L |
| 4 | 1 | 37 / 1 / R |
| | 0 | 12 / 1 / R |
| ... | ... | ... |




# Czy komputer może myśleć?

Jacek.Szczytko@fuw.edu.pl

| ... | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

dane

| ... | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Numer instrukcji (stan) | Dane | Nowy stan / 0-1 / R-L-STOP |
|---|---|---|
| 1 | 1 | 3 / 0 / R |
| | 0 | 4 / 0 / L |
| 2 | 1 | 9 / 0 / R |
| | 0 | 7 / 1 / STOP |
| 3 | 1 | 4 / 1 / L |
| | 0 | 2 / 0 / L |
| 4 | 1 | 37 / 1 / R |
| | 0 | 12 / 1 / R |
| ... | ... | ... |


# Czy komputer może myśleć?

Jacek.Szczytko@fuwedu.pl

| ... | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

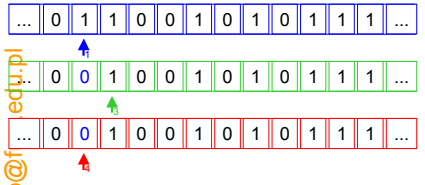
dane

| ... | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| ... | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Numer instrukcji (stan) | Dane | Nowy stan / 0-1 / R-L-STOP |
|---|---|---|
| 1 | 1 | 3 / 0 / R |
| | 0 | 4 / 0 / L |
| 2 | 1 | 9 / 0 / R |
| | 0 | 7 / 1 / STOP |
| 3 | 1 | 4 / 1 / L |
| | 0 | 2 / 0 / L |
| 4 | 1 | 37 / 1 / R |
| | 0 | 12 / 1 / R |
| ... | ... | ... |

# Czy komputer może myśleć?





$T_n$ *1/3/0/R,0/4/0/L,1/9/0/R, 0/7/1/STOP,1/4/1/L,0/2/0/L,...*

Listę instrukcji również możemy zakodować w postaci liczby.

Każda maszyna Turinga ma swój numerek! (inaczej: każdy program ma swój kod)

$T_n(m)=p$ ← TO JEST KOMPUTER



# Czy komputer może myśleć?

$T_n(m)=p$

Maszyna Turinga wykonuje OBLICZALNE operacje

zadania obliczalne, liczby obliczalne $\pi$, e, $\sqrt{2}$ itp, zbiory rekurencyjne...

A co jeśli maszyna nigdy nie zakończy rachunków?

$T_n(m)=\square$




# Czy komputer może myśleć?

$T_n(m)=p$

Maszyna Turinga wykonuje OBLICZALNE operacje

zadania obliczalne, liczby obliczalne $\pi$, e, $\sqrt{2}$ itp, zbiory rekurencyjne...

A co jeśli maszyna nigdy nie zakończy rachunków?

$T_n(m)=\square$

***Problem stopu***, ang. ***Halting problem****:* Czy istnieje algorytm, który moglibyśmy zastosować do WSZYSTKICH maszyn Turinga i który by pozwalał przewidzieć, że dana maszyna się zatrzyma?

$$H(n,m)=\begin{cases}1 \text{ dla} & T_n(m)=p\\ 0 \text{ dla} & T_n(m)=\square\end{cases}$$




# Czy komputer może myśleć?

$T_n(m)=p$

Maszyna Turinga wykonuje OBLICZALNE operacje

zadania obliczalne, liczby obliczalne $\pi$, e, $\sqrt{2}$ itp, zbiory rekurencyjne...

A co jeśli maszyna nigdy nie zakończy rachunków?

$T_n(m)=\square$

***Problem stopu***, ang. ***Halting problem****:* Czy istnieje algorytm, który moglibyśmy zastosować do WSZYSTKICH maszyn Turinga i który by pozwalał przewidzieć, że dana maszyna się zatrzyma?

$$H(n,m)=\begin{cases}1 \text{ dla} & T_n(m)=p\\ 0 \text{ dla} & T_n(m)=\square\end{cases}$$


*Entscheidungsproblem Hilberta* (1900 r. 1928 r.) – czy istnieje mechaniczna (algorytmiczna) procedura pozwalająca rozstrzygnąć wszystkie zagadnienia matematyczne należące do pewnej szerokiej, lecz dobrze zdefiniowanej klasy?

# Czy komputer może myśleć?

Jacek.Szczytko@fuw.edu.pl

Odpowiedź Kurta Gödla (1931 r.) i Alana Turinga (1937) – NIE!

**Inne**: **Church** zdefiniował system logiczny wraz z twierdzeniami i nazwał go efektywną obliczalnością. **Kleen** wymyślił tzw. "ogólne twierdzenia rekursywne" i pracował w ramach przez nie określonych. **Post** miał jeszcze zupełnie inny pomysł. Można wykazać, że wszystkie te istotnie różne podejścia są równoważne, co oznacza, że możemy zająć się tylko jednym z nich. Wybierzemy najbardziej powszechną metodę - Turinga.

# Twierdzenie Gödla

(zastosowane do maszyny Turinga)

Jacek.Szczytko@fuw.edu.pl

Jeśli istniałaby uniwersalna procedura obliczenia:

$$H(n,m)=\begin{cases}1 \text{ dla} & T_n(m)=p\\ 0 \text{ dla} & T_n(m)=\square\end{cases}$$

to obliczalne byłoby także:

$$T_n(m)\times H(n,m)$$

a więc i także;

$$T_n(n)\times H(n,n)+1$$ (argument przekątniowy):


A skoro jest to wyrażenie obliczalne, to znaczy, że jest to wynik obliczeń pewnej $k$-tej maszyny Turinga na $n$.

$$T_n(n)\times H(n,n)+1= T_k(n)$$

Ale dla $n=k$ powyższe równanie jest SPRZECZNE!

$$\begin{cases}p\times 1+1= p\\ \square\times 0+1=\square\end{cases}$$

# Twierdzenie Gödla

(zastosowane do maszyny Turinga)

Jacek.Szczytko@fuw.edu.pl

Nie istnieje uniwersalna procedura, która pozwalałaby z góry rozstrzygnąć, czy dany program zakończy pracę, czy nie. Natomiast możliwa byłaby procedura

$$H'(n,m)=\begin{cases}1 \text{ dla} & T_n(m)=p\\ 0 \text{ lub } \square \text{ dla} & T_n(m)=\square\end{cases}$$

Jeśli: $\begin{cases}\square\times\square=\square\\ \square+1=\square\end{cases}$ to wtedy dla $T_k(k)= T_k(k)\times H(k,k)+1= \square$

(bo inne wartości prowadziłyby do sprzeczności)

Algorytm jednak o tym nie może „wiedzieć", bo gdyby „wiedział", to $H'(k,k)=0$

Czyli dla KAŻDEGO algorytmu sprawdzającego $H'$ możemy znaleźć taką maszynę Turinga $T_k(k)$ o której MY WIEMY, że $T_k(k)=\square$, ale algorytm $H'$ tego nie będzie w stanie stwierdzić, bo nigdy się nie zatrzyma, bo $H'(k,k)=\square$!

# Twierdzenie Gödla

(ogólnie)

Jacek.Szczytko@fuw.edu.pl

Najtrudniejsza część dowodu to pokazanie w jaki sposób można zakodować poszczególne aksjomaty i reguły wnioskowania systemu formalnego (np. algebry, geometrii euklidesowej itp.) w postaci operacji arytmetycznych.

***System formalny spójny*** (niesprzeczny wewnętrznie) to taki w którym nie da się udowodnić pewnego zdania i jego zaprzeczenia jednocześnie; inaczej mówiąc w systemie spójnym zaprzeczenie zdania prawdziwego jest zawsze fałszywe.

***System formalny zupełny*** to taki, w którym możliwe jest rozstrzygnięcie o prawdziwości dowolnego prawidłowo zapisanego zdania tego systemu.

Wikipedia

# Twierdzenie Gödla
(ogólnie)



***Twierdzenie Gödla o niezupełności*** stwierdza, że dowolny system formalny zawierający w sobie aksjomaty arytmetyki liczb naturalnych, jest albo zupełny albo spójny i nigdy nie posiada obu tych cech jednoczesnie. Innymi słowy, jeśli system jest niesprzeczny to istnieją zdania których prawdziwości nie da się dowieść za pomocą aksjomatów i twierdzeń rozważanego systemu formalnego.

***II twierdzenie Gödla o niedowodliwości spójnosci*** to konsekwencja wcześniejszego twierdzenia Gödla: Głosi ono, nie da się dowieść spójności (niesprzeczności) żadnego systemu formalnego zawierającego arytmetykę liczb naturalnych w ramach samego tego systemu. Aby taki dowód przeprowadzić niezbędny jest system wyższego rzędu, którego spójności w ramach jego samego również nie można dowieść i tak ad. infinituum.

Wikipedia

# No i co z tego?



Obydwa twierdzenia Gödla można uogólnić na dowolne systemy formalne zawierające skończoną lub rekurencyjnie przeliczalną liczbę aksjomatów o ile tylko arytmetyka liczb naturalnych wchodzi w ich skład lub zawierają one skończoną liczbę aksjomatów i umożliwiaja przeprowadzenie tzw. arytmetyzacji twierdzeń.

Maszyna Turinga (KAŻDY KOMPUTER) jest właśnie takim systemem formalnym.

# No i co z tego?



Obydwa twierdzenia Gödla można uogólnić na dowolne systemy formalne zawierające skończoną lub rekurencyjnie przeliczalną liczbę aksjomatów o ile tylko arytmetyka liczb naturalnych wchodzi w ich skład lub zawierają one skończoną liczbę aksjomatów i umożliwiaja przeprowadzenie tzw. arytmetyzacji twierdzeń.

Maszyna Turinga (KAŻDY KOMPUTER) jest właśnie takim systemem formalnym.

## Czy maszyna może myśleć tak jak człowiek?

Wydaje się, że umysł ludzki nie działa według algorytmu matematycznego, w każdym razie nie algorytmu opartego jedynie na liczbach naturalnych (lub wymiernych, przeliczalnych, rekurencyjnych etc.).

O ile istnieją algorytmy dowodzenia (algebry, teorie, geometrie), o tyle nie istnieje algorytm znajdowania dowodów (inaczej by można było zbudować uniwersalną maszynę $H(n,m)$).

# Moim zdaniem



- Teorie fizyczne (mech. klasyczna, mech. kwantowa) są *deterministyczne*, ale nie oznacza to wcale, że zawsze są *obliczalne* (np. zagadnienie 3ch ciał lub rzeczywiste komputery!).

- Warunki początkowe fizycznego (realnego) układu ciał nie da się określić jedynie przez liczby wymierne (niewymierne algebraiczne, rekurencyjne itp. - przeliczalne)

- Pomiar w mechanice kwantowej NIE JEST deterministyczny (redukcja paczki falowej!)

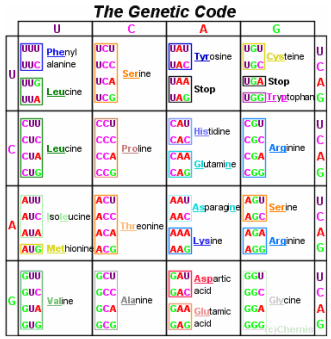
=> Na maszynie Turinga nie da się zasymulować rzeczywistości fizycznej z nieskończoną dokładnością.

(chociaż nie pokazaliśmy, że ludzkie MYŚLENIE faktycznie wymaga któregoś z powyższych warunków)

# Moim zdaniem

Obecne maszyny Turinga (a więc DOWOLNE komputery) mają zatem dwa problemy jeśli „chciałyby” myśleć po ludzku

1. Hardware'owy – z fizyką
2. Software'owy – z tw. Gödla






# Moim zdaniem

Obecne maszyny Turinga (a więc DOWOLNE komputery) mają zatem dwa problemy jeśli „chciałyby” myśleć po ludzku

1. Hardware'owy – z fizyką
2. Software'owy – z tw. Gödla

**Science-fiction** (?)

Komputery oparte na algebrze nie poradzą sobie z ograniczeniami teoretycznymi (Turing, Gödel), zawsze będą mogły wykonywać tylko operacje obliczalne.

Potrzebne byłyby zupełnie nowe architektury oparte na **NIEOBLICZALNYCH** zasadach (algorytmy uczące się w interakcji z otoczeniem? komputery kwantowe?)



# Dziękuję Państwu za uwagę!


Rys. źródło: Internet