TEZA CHURCHA-TURINGA

Maszyna Turinga:
- ma skończenie wiele stanów
- zapisuje po jednym symbolu na liniowej taśmie

Co można zrobić za pomocą maszyny Turinga?

Wszystko!
Maszyna Turinga potrafi rozwiązać każdy efektywnie rozwiązywalny problem algorytmiczny!

Teza CT

Różne inne modele komputera uniwersalnego:
- rachunek lambda (Church)
- system produkcji dla symboli (Post)
- klasa funkcji rekurencyjnych (Kleen)
- ... i wiele innych

Wszystkie modele są **równoważne** w sensie klasy problemów algorytmicznych, które rozwiązują!

Odmiany maszyny Turinga:
- maszyna z taśmą tylko jednostronnie nieskończoną
- maszyny z wieloma taśmami i głowicami
- maszyny z taśmami(?) dwuwymiarowymi
- maszyny niedeterministyczne

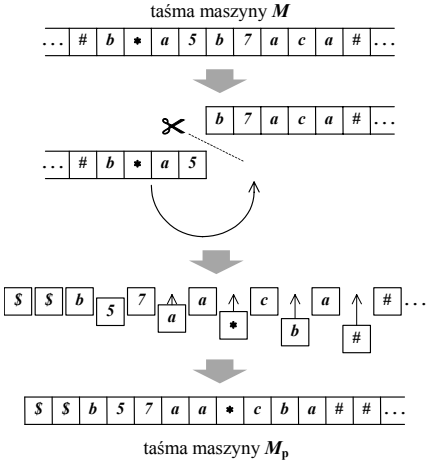
Przykład symulacji jednej maszyny na drugiej

maszyna M - taśma nieskończona z obu stron
maszyna M_p - taśma nieskończona tylko z prawej strony

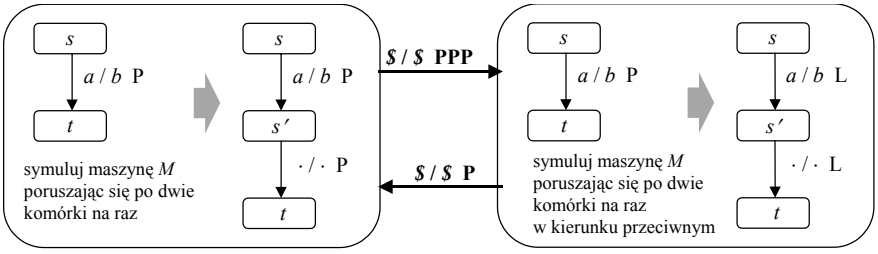
maszyna M nie jest silniejsza od maszyny M_p

☞ wystarczy pokazać, że na maszynie M_p można symulować działanie maszyny M

1. złożenie taśmy nieskończonej z obu stron w taśmę nieskończoną tylko z prawej strony



2. modyfikacja przejść międzystanowych w nowym diagramie, który składa się z dwóch diagramów maszyny M



Programy licznikowe (całkiem inny model obliczeń)

- zmienne przechowują nieujemne liczby całkowite (tzw. liczniki)
- trzy rodzaje elementarnych operacji:
  $X \leftarrow 0$, $X \leftarrow Y + 1$, $X \leftarrow Y - 1$ (przyjmujemy, że $X = 0$ dla $Y = 0$)
- dwie instrukcje sterujące: proste następstwo i skok warunkowy
  jeśli $X = 0$ ***skocz-do*** L (L jest etykietą pewnej instrukcji)
- program licznikowy zatrzymuje się wtedy, gdy:
  - występuje próba wykonania nieistniejącej instrukcji
  - występuje próba przejścia do nieistniejącej etykiety
  - wyczerpana zostaje lista instrukcji

Przykład programu licznikowego

mnożenie dwóch liczb $Z = X * Y$

```
      U ← 0
      Z ← 0
A:    jeśli X = 0 skocz-do L
      X ← X - 1
      V ← Y + 1
      V ← V - 1
B:    jeśli V = 0 skocz-do A
      V ← V - 1
      Z ← Z + 1
      jeśli U = 0 skocz-do B
```

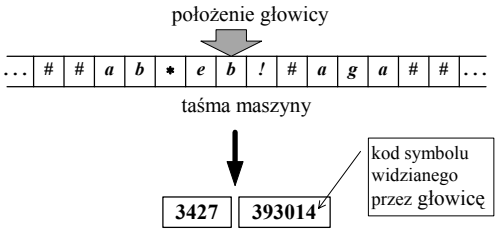
Dwa triki:
$V \leftarrow Y + 1$ oraz $V \leftarrow V - 1$ zastępują $V \leftarrow Y$
$U \leftarrow 0$ oraz ***jeśli*** $U = 0$ ***skocz-do*** B zastępują ***skocz-do*** B
Dwie pętle:
wewnętrzna (od B: do ***skocz-do*** B) - zwiększa Z o Y
zewnętrzna (od A: do ***skocz-do*** A) - wykonuje pętlę wewnętrzną X razy

Programy licznikowe są równoważne maszynom Turinga w sensie klasy problemów algorytmicznych, które rozwiązują

1. Taśmę i położenie głowicy maszyny Turinga można zakodować za pomocą dwóch liczb, a każde przejście w diagramie reprezentować odpowiednim „bloczkiem" programu licznikowego.

   np. dla alfabetu #, !, *, a, b, c, d, e, f, g kodujemy symbole

#	0	*	2	b	4	d	6	f	8
!	1	a	3	c	5	e	7	g	9

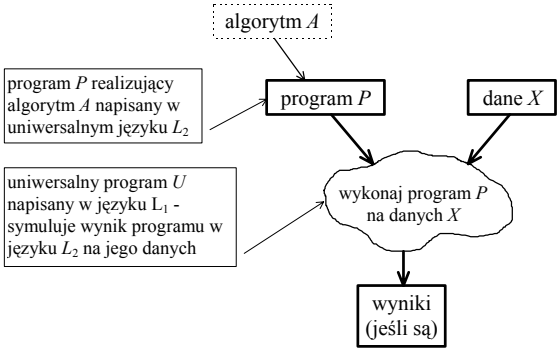


2. Wartości wszystkich liczników można rozmieścić na taśmie maszyny Turinga, rozdzielając je symbolami „*", a instrukcje programu licznikowego reprezentować w postaci specjalnych stanów w diagramie przejść

Maszyny Turinga i programy licznikowe stały się uniwersalnymi modelami dzięki wykorzystaniu nieskończonej ilości pamięci:

- w maszynach Turinga ilość informacji w pojedynczej komórce jest skończona i ograniczona, ale liczba komórek jest potencjalnie nieskończona
- w programach licznikowych liczba zmiennych-liczników jest skończona, ale w każdej można przechować dowolnie dużą liczbę, za pomocą której można zakodować potencjalnie nieskończoną ilość informacji

Konsekwencją tezy CT jest istnienie **algorytmów uniwersalnych**



- można zbudować **uniwersalną maszynę Turinga**, która może symulować działanie dowolnej maszyny Turinga na dowolnych danych (trzeba opisać na taśmie zlinearyzowany diagram przejść, reprezentując każde przejście jako parę stanów z podaną etykietą przejścia)
- można także skonstruować **uniwersalny program licznikowy**

Maszyny Turinga i programy licznikowe są wielomianowo równoważne, tzn. klasa problemów łatwo rozwiązywalnych jest taka sama dla obu modeli

Rozwijając tezę CT można dojść do wniosku, że:

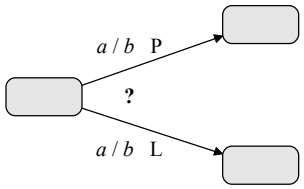
jeśli jakiś (szybki) komputer rozwiązuje pewien problem w czasie $O(f(N))$, to istnieje równoważna mu maszyna Turinga, która potrzebuje na rozwiązanie tego problemu nie więcej niż $O(p(f(N)))$ czasu, dla pewnej ustalonej funkcji wielomianowej p

Zatem:

- klasa problemów obliczalnych (rozstrzygalnych) jest silna tj. niewrażliwa na zmianę modelu obliczeń lub języka zapisu algorytmu
- klasa problemów łatwo rozwiązywalnych ***P*** jest także silna (tzw. teza obliczania sekwencyjnego, czyli wykonywanego krok po kroku)
- klasa ***NP*** jest silna
- klasa problemów o wykładniczej złożoności czasowej jest silna
- klasa problemów o liniowej złożoności czasowej nie jest silna tzn. złożoność tych problemów może zależeć od przyjętego modelu obliczeń
  np.
  problem porównania liczby wystąpień dwóch różnych symboli w danych wejściowych wymaga:
  $O(N^2)$ lub $O(N*\log N)$ na prostej maszynie Turinga,
  $O(N)$ na maszynie Turinga z dodatkowym licznikiem.

Formalnie klasy problemów *P* i *NP* definiuje się w kategoriach obliczeń na maszynie Turinga:

- problemy z klasy ***P*** są rozwiązywalne przez zwykłe maszyny Turinga w czasie wielomianowym
- problemy z klasy ***NP*** są rozwiązywalne przez niedeterministyczne maszyny Turinga w czasie wielomianowym



przejście niedeterministyczne

Gdyby niedeterministyczne maszyny Turinga spełniały kryterium sekwencyjności, to problem ***P = NP*** byłby rozwiązany pozytywnie, ale nie spełniają, gdyż wybór lepszego przejścia jest „magiczny", a bez magii oznacza konieczność równoczesnego wypróbowywania różnych możliwości!

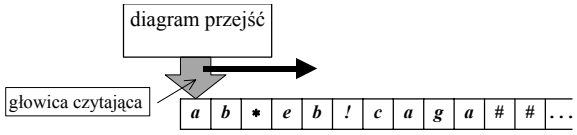
Na mocy tezy CT wystarczyło by pokazać, że pewien problem *NP*-zupełny nie może być rozwiązany za pomocą maszyny Turinga w czasie krótszym niż wykładniczy, aby wykazać, że $\boldsymbol{P \neq NP}$.

- aby wyznaczyć górne ograniczenie problemu algorytmicznego powinno się użyć najbogatszego i najsilniejszego formalizmu (z konstrukcjami sterującymi wysokiego poziomu i rozwiniętymi strukturami danych)
- aby udowodnić dolne ograniczenie należy użyć możliwie najprostszego modelu obliczeń, czyli np. maszyny Turinga lub programu licznikowego

Maszyny Turinga stosuje się powszechnie do dowodzenia dolnych ograniczeń dla złożoności problemów algorytmicznych

Pewne klasy problemów można definiować narzucając ograniczenia na działanie maszyny Turinga np.
Pspace - klasa problemów rozwiązywalnych przez maszynę Turinga, która może korzystać tylko z wielomianowo rosnącej liczby komórek na taśmie: $\boldsymbol{NP \subseteq Pspace}$

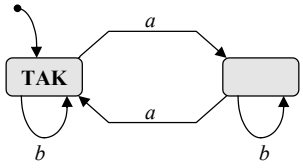
Automaty skończenie stanowe (skończone)
(takie jakby jednokierunkowe maszyny Turinga)
ograniczamy ruch głowicy maszyny T. tylko do jednego kierunku



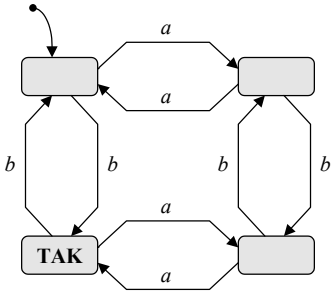
⇒ każda komórka może być odczytana tylko raz
⇒ nie można wrócić do zapisywanych komórek, czyli w problemach decyzyjnych nie jest potrzebna dla głowicy funkcja zapisu
⇒ poza daną sekwencją automat zawiera na taśmie tylko symbole puste, czyli może zatrzymywać się po osiągnięciu końca danych wejściowych (napotkaniu pustej komórki)
⇒ wystarczą stany końcowe tylko z odpowiedzią na TAK, gdyż zatrzymanie się automatu na końcu danych w każdym innym stanie można interpretować jako odpowiedź na NIE
⇒ etykieta przejścia zawiera zatem tylko symbol-wyzwalacz (nie jest potrzebny człon akcji, bo jest ona tylko jedna - przejdź do następnej komórki w kierunku ruchu głowicy)

Przykład automatu skończonego

zdefiniowanego nad alfabetem $\{a,b\}$ i badającego parzystość wystąpień symbolu *a*:



Ten automat nie liczy, ale o parzystości potrafi rozstrzygać.
A co potrafi ten automat?

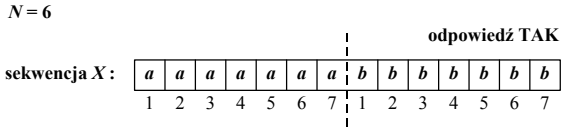


Automat skończony nie potrafi liczyć!

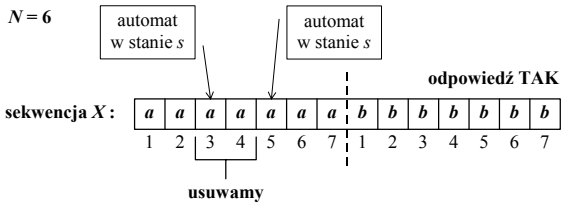
Teza:
Żaden automat skończony nie potrafi rozstrzygnąć, czy podana sekwencja wejściowa zawiera taką samą liczbę symboli *a* i *b*.

Dowód: (nie wprost)

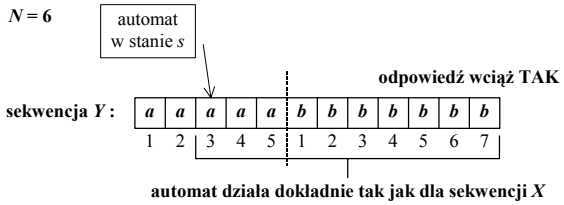
Załóżmy, że pewien automat ***F*** potrafi rozstrzygnąć podany problem decyzyjny. Niech liczba stanów automatu ***F*** wynosi ***N***. Rozważmy sekwencję wejściową ***X***, która zawiera najpierw dokładnie ***N*** + **1** symboli ***a***, a potem ***N*** + **1** symboli ***b***.



W trakcie przesuwania się głowicy wzdłuż taśmy muszą pojawić się dwie komórki, w których automat będzie w tym samym stanie ***s***, gdyż liczba komórek z tym samym symbolem jest większa od liczby stanów.



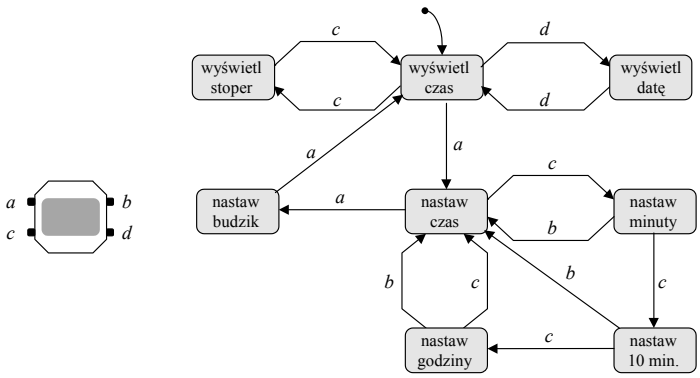
W nowej sekwencji ***Y*** usuńmy komórki, tak aby stan ***s*** był osiągany tylko raz przy trzeciej komórce.



Zatem istnieje sekwencja zawierająca różną liczbę symboli ***a*** i ***b***, dla której automat daje odpowiedź TAK. Przeczy to założeniu, że automat daje taką odpowiedź tylko dla sekwencji z taką sama liczbą symboli ***a*** i ***b*** !

□

Przykład automatu opisującego zegarek cyfrowy



Etykiety przejść w diagramie automatu skończonego nazywane są często **sygnałami** lub **zdarzeniami**

Rola abstrakcyjnych modeli obliczeń

- badania nad złożonością obliczeniową i nierozstrzygalnością
- teoria automatów
- teoria języków formalnych
- automaty ze stosem w lingwistyce strukturalnej i w konstruowaniu kompilatorów
- badanie siły niedeterminizmu np. niedeterministyczny automat ze stosem
- badania nierozstrzygalności problemów decyzyjnych dotyczących samych modeli obliczeń:
  - równoważność algorytmiczna dwóch maszyn Turinga jest nierozstrzygalna,
  - równoważność algorytmiczna dwóch automatów skończonych jest rozstrzygalna,
  - o rozstrzygalności problemu równoważności algorytmicznej dwóch automatów ze stosem nic nie wiadomo (a problem ma istotne znaczenie dla budowy języków programowania i kompilatorów)
- zagadnienie „obliczalnego” zapytania do bazy danych - jak powinny wyglądać „maszyny Turinga” dla baz danych lub baz wiedzy