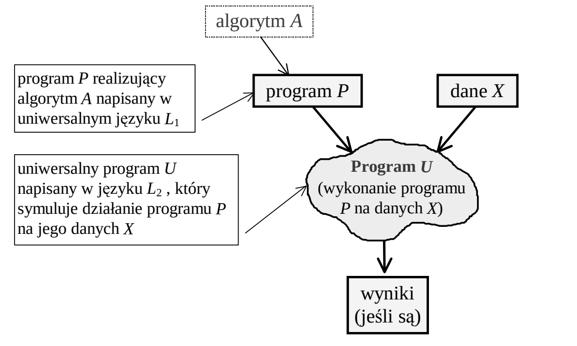
Konsekwencje tezy CT

MT jest uniwersalną MO

Można zbudować **uniwersalną maszynę Turinga**, która może symulować działanie każdej innej maszyny Turinga z dowolnymi dopuszczalnymi danymi zapisanymi na jej taśmie (trzeba w tym celu opisać na taśmie zlinearyzowany diagram przejść, reprezentując każde przejście jako parę stanów z podaną etykietą przejścia)

Zatem konsekwencją tezy CT jest istnienie **programów uniwersalnych**, które mogą symulować działanie każdego innego programu zapisanego w dowolnym języku dla dowolnych dopuszczalnych dla tego programu danych, tzn. kończyć działanie w taki sam sposób jak symulowany program i podawać taki sam wynik, jak gdyby rzeczywiście ten program został uruchomiony dla tych danych.



Rozwijając tezę CT można dojść do następującego wniosku.

Jeśli na jakimś (dowolnie szybkim) komputerze (maszynie obliczeniowej) można rozwiązać pewien problem algorytmiczny w czasie **O(*f* (*N*))**, to istnieje równoważna temu komputerowi maszyna Turinga, która potrzebuje na rozwiązanie tego problemu nie więcej niż **O(*p*(*f* (*N*)))** czasu, dla pewnej ustalonej funkcji wielomianowej ***p***.

Tzn. złożoność np. O(N^2) może wzrosnąć do O(N^7) lub nawet do O(N^{64}), ale nie do O(2^N). W rzeczywistości „przejście" na MT dla znanych problemów wiąże się z funkcjami wielomianowymi niezbyt wysokiego rzędu np. $p(x) = x^4$ lub x^5. Zatem „dobry" wielomianowy algorytm nie stanie się nieakceptowalnie gorszy po redukcji do modelu uniwersalnego typu MT.

Jeśli analizujemy algorytmy dla różnych klas problemów w oparciu o różne modele obliczeń i różne języki ich zapisu, to z tezy CT wynika, że:

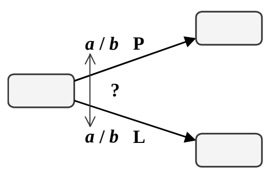
- klasa problemów **obliczalnych** jest **silna** tj. niewrażliwa na zmianę modelu lub języka,
- klasa problemów łatwo rozwiązywalnych **P** jest **silna** (jest to tzw. *teza obliczania sekwencyjnego*, czyli wykonywanego krok po kroku),
- klasa **NP** jest **silna**,
- klasa problemów o **wykładniczej** złożoności czasowej jest **silna**,
- klasa problemów o liniowej złożoności czasowej <u>nie jest</u> silna tzn. ocena złożoności tych problemów może zależeć od przyjętego modelu obliczeń.

A jak teza CT ma się do rozstrzygnięcia, czy P = NP?

Jeśli chcemy formalnie zdefiniować klasy problemów P i NP, to najlepiej jest to zrobić w kategoriach obliczeń na maszynie Turinga:

- rozmiarem danych wejściowych jest liczba komórek na taśmie MT potrzebna do zapisania zakodowanych danych wejściowych,
- czas działania algorytmu mierzony jest liczbą przejść pomiędzy stanami jaka jest potrzebna do wyznaczenia zamierzonego wyniku,
- problemy z klasy **P** są rozwiązywalne w czasie wielomianowym przez zwykłe maszyny Turinga,
- problemy z klasy **NP** są rozwiązywalne w czasie wielomianowym przez <u>niedeterministyczne</u> maszyny Turinga.

Przykład przejścia niedeterministycznego



Gdyby niedeterministyczne maszyny Turinga spełniały kryterium sekwencyjności, to problem **P = NP** byłby rozstrzygnięty pozytywnie, ale nie spełniają, gdyż wybór właściwego przejścia jest „magiczny", a bez wyroczni oznacza konieczność <u>równoczesnego</u> wypróbowywania różnych możliwości, aby w tym samym czasie sprawdzić jakie będą konsekwencje każdej z nich!

Na mocy tezy CT wystarczyło by pokazać, że pewien problem NP-zupełny nie może być rozwiązany za pomocą MT w czasie krótszym niż wykładniczy, aby wykazać, że **P ≠ NP**.

- aby wyznaczyć jak najniższe <u>górne</u> oszacowanie złożoności problemu algorytmicznego należy użyć jak najbogatszego i najsilniejszego formalizmu zapisu algorytmu (z instrukcjami sterującymi wysokiego poziomu i rozwiniętymi strukturami danych)
- aby udowodnić jak najwyższe <u>dolne</u> oszacowanie złożoności problemu algorytmicznego należy użyć możliwie najprostszego modelu obliczeń, czyli np. MT.

MT stosuje się powszechnie do dowodzenia dolnych oszacowań dla złożoności problemów algorytmicznych, w tym także do dowodzenia ich nierozstrzygalności.

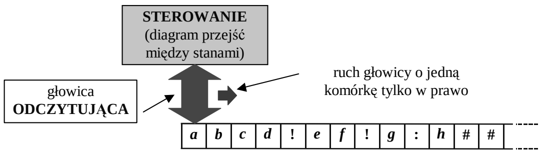
Pewne ciekawe klasy problemów algorytmicznych można definiować narzucając dodatkowe ograniczenia na działanie MT.

Np. klasa **Pspace** – klasa problemów rozwiązywalnych przez MT, która może korzystać tylko z wielomianowo przyrastającej liczby komórek na taśmie.

Okazuje się, że **NP ⊆ Pspace**

Automaty skończenie stanowe (skończone)

(rodzaj jednokierunkowej MT, nie będący MO)



PAMIĘĆ (jednostronnie nieskończona taśma)

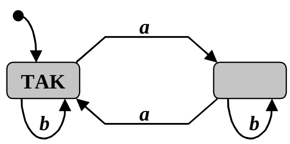
- każda komórka taśmy może być odczytana <u>tylko raz</u> ,
- nie można wrócić do raz odczytanych komórek, czyli nie jest potrzebna dla głowicy funkcja zapisu

- poza zapisanymi sekwencyjnie danymi wejściowymi automat zawiera na taśmie tylko symbole puste, z zatem może po prostu zatrzymywać się po przeczytaniu do końca danych wejściowych, czyli po napotkaniu symbolu #,
- W problemach decyzyjnych wystarczy umieścić w diagramie tylko stany końcowe z odpowiedzią TAK, gdyż zatrzymanie się automatu na końcu danych w każdym innym stanie można interpretować jako odpowiedź NIE,
- etykieta przejścia zawiera tylko jeden symbol – wyzwalacz przejścia (nie jest potrzebny człon akcji, bo jest ona tylko jedna – przejdź w prawo do następnej komórki)

MT może potencjalnie wykonać nieskończenie wiele przejść pod stanu do stanu. W automacie skończonym liczba przejść jest równa liczbie komórek z danymi, które trzeba kolejno odczytać i jest zatem zawsze skończona.

Przykład automatu skończonego

Automat pracuje na dwuelementowym alfabecie {*a*, *b*} i bada parzystość wystąpień symbolu *a* w ciągu danych wejściowych.

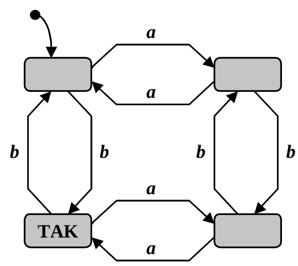


1. Zestaw danych: | *a* | *b* | *b* | *a* | *a* |

2. Zestaw danych: | *b* | *a* | *b* | *b* | *a* |

Ten automat nie potrafi zliczyć wystąpień *a*, ale o parzystości potrafi rozstrzygnąć

A co potrafi rozstrzygnąć ten automat?



Automaty skończone nie potrafią liczyć!

Teza:

Żaden automat skończony z alfabetem $\{a, b\}$ nie potrafi rozstrzygnąć, czy wejściowy ciąg symboli zawiera taką samą liczbę symboli a i b.

Dowód: (przez zaprzeczenie)

Załóżmy, że pewien automat ***F*** potrafi rozstrzygnąć podany problem decyzyjny.

Niech liczba stanów tego automatu ***F*** wynosi ***N***.
Rozważmy wejściową sekwencję symboli ***X***, która zawiera najpierw dokładnie ***N* + 1** symboli ***a***, a potem ***N* + 1** symboli ***b***.

Np. dla $N = 6$:

a	*a*	*a*	*a*	*a*	*a*	*a*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	#
1	2	3	4	5	6	7	1	2	3	4	5	6	7	

Odpowiedź powinna być TAK!

W trakcie przesuwania się głowicy wzdłuż taśmy muszą pojawić się dwie komórki z symbolem a, w których automat będzie na diagramie w tym samym stanie ***s***, gdyż liczba komórek z tym samym symbolem jest większa od liczby stanów (zasada „szufladkowa")

automat w stanie ***s*** — automat w stanie ***s***

X															
	a	*a*	*a*	*a*	*a*	*a*	*a*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	#
	1	2	3	4	5	6	7	1	2	3	4	5	6	7	

Zbudujmy nową sekwencję wejściową ***Y*** przez usunięcie z sekwencji ***X*** zaznaczonych dwóch komórek, tak aby stan ***s*** był osiągany tylko raz przy trzeciej komórce.

automat w stanie ***s***

Y													
	a	*a*	*a*	*a*	*a*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	#
	1	2	3	4	5	1	2	3	4	5	6	7	

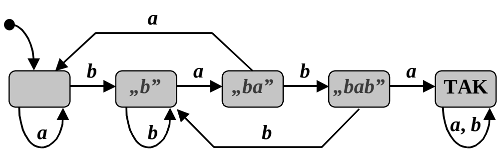
Dla podanej sekwencji ***Y*** automat będzie działał tak samo jak dla sekwencji ***X***, bo tak samo przy trzeciej komórce znajdzie się w stanie ***s*** i w dalszym swoim działaniu w zaznaczonym zakresie odczyta taką samą sekwencje symboli, jaka w sekwencji ***X*** zaczynała się od piątej komórki.

Zatem istnieje wejściowa 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*** !

c.b.d.o.

Przykład automatu skończonego

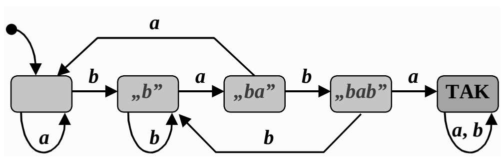
Automat pracuje na dwuelementowym alfabecie $\{a, b\}$ i rozstrzyga czy w ciągu danych wejściowych pojawiła się choć raz sekwencja „baba".



a	*b*	*b*	*a*	*a*	*b*	*a*	*b*	*b*	*a*	*b*	*a*	*b*	*a*	*a*	#

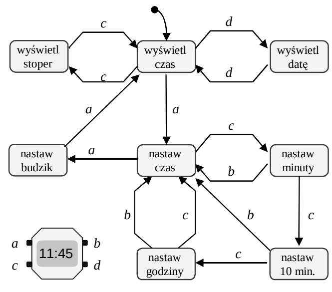
a	*b*	*b*	*a*	*a*	*b*	*a*	*b*	*b*	*a*	*b*	*a*	*b*	*a*	*a*	#

STOP



Automaty skończone są wygodnym modelem opisującym działanie prostych urządzeń sterowanych zewnętrznymi sygnałami: windy, pralki, telefony komórkowe itp.

Przykład automatu opisującego działanie zegarka cyfrowego



Etykiety automatu skończonego nazywane są wtedy **sygnałami** lub **zdarzeniami**

Do czego są wykorzystywane abstrakcyjne MO lub automaty?

- Do badania złożoności obliczeniowej i rozstrzygalności problemów
- W teorii automatów do ich analizy i syntezy
- W teorii języków formalnych
- W lingwistyce strukturalnej (automaty ze stosem) i w konstruowaniu kompilatorów
- Do badania siły niedeterminizmu np. niedeterministyczny automat ze stosem
- Do badania rozstrzygalności problemów decyzyjnych dotyczących samych modeli obliczeń

- Do badania rozstrzygalności 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)
- Do analizy zagadnienia „obliczalnego” zapytania do bazy danych (za pomocą specjalnych „maszyn Turinga” dla baz danych lub baz wiedzy)