Uniwersytet Śląski
Wydział Matematyki, Fizyki i Chemii

Co komputer wie, a czego nigdy nie będzie wiedział, czyli krótko acz treściwie o maszynach Turinga

Autor: Mateusz Szymański

21 grudnia 2012

Spis treści

1 Model teoretyczny komputerów

1.1 Definicje

Zanim zacznę, żeby była jasność, o czym mówimy, trzeba będzie zdefiniować kilka pojęć.

Definicja

Maszyną Turinga nazywamy wyidealizowany model maszyny liczącej, który jest zdolny wykonywać, bez ograniczeń co do czasu wykonywania/ilości danych, pewien z góry zadany **jeden algorytm**, tj. uporządkowany zbiór operacji, którego wynikiem jest rozwiązanie (np. wartość logiczna prawda.

Nazwa pochodzi od Alana Turinga, brytyjskiego matematyka i logika, uznawanego za ojca dzisiejszej informatyki, który wprowadził to pojęcie.

Wniosek

Z definicji można wyciągnąć następujące wnioski:

1. Fizyczne komputery nie są równoważne maszynom Turinga, gdyż operują jedynie skończonymi informacjami.
2. Można rozważyć tzw. Uniwersalną Maszynę Turinga, która potrafi symulować inne maszyny Turinga (nie musimy mieć osobnych maszyn, by rozwiązywać różnych problemów).

Definicja

Model teoretyczny nazywamy **zupełnym**, jeżeli o każdym zdaniu bez zmiennych wolnych w tej teorii, możemy orzec, czy jest prawdziwe czy nie. Model teoretyczny nazywamy **niesprzecznym**, jeżeli nie można wykazać jednocześnie prawdziwości pewnego zdania, jak i jego negacji. Problem jest **rozstrzygalny**, jeżeli istnieje metoda wykazania prawdziwości pewnego zdania w skończonej ilości kroków.

Logiki proste (tzw. pierwszego rzędu) są systemami zupełnymi i niesprzecznymi. Ale, jeżeli tylko wprowadzi się pojęcie liczby naturalnej, nie możemy mieć systemu, który jest zarówno niesprzeczny, jak i zupełny (Kurt Gödel). Innymi słowy, w niesprzecznym systemie istnieje taki problem, o którym nie możemy powiedzieć, czy jest prawdziwy czy nie, a w zupełnym – istnieją formuły zarówno prawdziwe, jak i fałszywe. Jesteśmy w stanie zrezygnować z zupełności na rzecz niesprzeczności. Nie interesuje nas możliwość odpowiedzenia na wszystkie pytania bez pewności prawdziwości danej odpowiedzi; taki model jest bezużyteczny.
W naszym informatycznym rozumieniu napotkamy się na problem, dla którego albo nie znajdziemy rozwiązania (np. rozwiązania liczbowego czy wartości logicznej) w skończonym czasie, albo nie istnieje sam algorytm, który mógłby prowadzić do **poprawnego** rozwiązania.

1.2 Problem stopu

Dla przykładu wprowadźmy sobie **całkowitą** zmienną $V = 0$ i wykonajmy poniższy kod:

```
DOPÓKI (1==1)
  V += 1;
```

Powyższy algorytm oczywiście nigdy się nie przerwie, jest **niezatrzymujący się** niezależnie od V. Natomiast, modyfikując ten algorytm:

```
DOPÓKI (1==1)
  V += 1;
  JEŻELI (V==20)
    PRZERWIJ;
```

możemy łatwo stwierdzić, że program się zatrzyma, o ile $V < 20$. Powiedzmy, że chcemy znaleźć taki algorytm, które będzie w stanie przewidzieć, czy dowolny wprowadzony algorytm się zatrzyma. Załóżmy, że taki istnieje (oznaczmy $S(P, V)$), który dla dowolnego programu P i danych V:

1. zatrzymuje się i zwraca 1, jeżeli P zatrzymuje się na danych wejściowych V,
2. zatrzymuje się i zwraca 0 w przeciwnym razie.

Stwórzmy program T na bazie S, który zatrzymuje się wtedy i tylko wtedy, gdy wprowadzony program P zapętla się na swoim własnym kodzie jako dane wejściowe (tj. $S(P,P) = 1$). Zastanówmy się, czy algorytm zapętli się, jeżeli za P wstawimy kod T. Jeżeli się nie zapętli, to $S(T,T) = 1$, a więc T musi wpaść w pętlę – sprzeczność. Z drugiej strony, jeżeli $S(T,T) = 0$, czyli program się zapętla, to zgodnie z programem, powinien się zatrzymać, co jest kolejną sprzecznością. Zatem istnienie takiego algorytmu prowadzi do sprzeczności.

Wniosek

Problem stopu jest nierozwiązywalny

Wniosek

Mimo wszystko, możemy skonstruować algorytm rozwiązujący problem stopu dla pewnej, ustalonej struktury algorytmów.

To, że nie istnieje maszyna, która mogłaby porównywać wszystkie możliwe programy nie przeczy istnieniu maszyny, która może sprawdzać złożoność pewnych ustalonych konstrukcji, jak np. pętle w przykładzie – wystarczy, że zostanie sprawdzony poniższy fragment:

```
JEŻELI (V==X)
```

a jego porównanie z danymi wejściowymi (V) da nam jasną odpowiedź, czy algorytm się zatrzyma.

Wniosek

Nie można sprowadzić matematyki do szeregu algorytmów.

W szczególności z niemożności porównywania złożoności obliczeniowej wynika niemożność porównywania maszyn Turinga innymi maszynami Turinga w ogólności. Ale jest to w dużym stopniu możliwe dla człowieka. Dzięki temu matematycy nie muszą się obawiać o swoją pracę ;).

1.3 Liczby nieobliczalne

Istnieją liczby nieobliczalne – takie liczby rzeczywiste, które nie są rezultatem jakiegokolwiek obliczeniowego algorytmu maszyny Turinga. Nie powinno nas to szczególnie dziwić. Istnieje nieskończenie wiele liczb, których nie jesteśmy w stanie zapisać, używając skończonej ilości znaków i działań algebraicznych. Dowód istnienia takich liczb opiera się na metodzie przekątniowej Cantora, która posłużyła wykazaniu, że liczb rzeczywistych jest „więcej” niż naturalnych. Co ciekawe, liczb obliczalnych jest tyle samo w sensie mocy zbioru, co liczb naturalnych czy wymiernych. e, π czy ϕ (złoty podział) są np. liczbami niewymiernymi obliczalnymi.

2 Komputery fizyczne

Komputer, który stoi u nas w pokojach jest zubożoną wersją maszyny Turinga, operuje tylko skończonymi danymi i ograniczoną możliwością obliczeniową. Stąd wypływa wniosek:

Wniosek

Komputer nie potrafi operować liczbami innymi niż wymierne.

Nowoczesne programy matematyczne „oszukują nas”, używając do zapisu symboli, np. $\sqrt{2}$ jako rozwiązania pewnego równania kwadratowego. Jednak, gdybyśmy „spytali” naszą maszynę, co rozumie przez ten symbol, musiałaby wyliczyć rozwinięcie dziesiętne, tj. istnieje jakiś algorytm, który pozwala znaleźć to rozwinięcie z dowolną dokładnością. Nie odpowie jednak na pytanie, jaka cyfra stoi na bilionowym miejscu po przecinku, obecne możliwości komputerów na to nie pozwalają.

Wniosek

Komputer nie zna geometrii.

Komputery znane nam dzisiaj nie znają geometrii. Figury wyświetlane na monitorze są niedoskonałe, np. okrąg wyświetlany jest jako łamana „schodkowa". Duża rozdzielczość potrafi sprawić wrażenie gładkości, ale jest to tylko złudzenie. Ale, nawet jeżeli monitor komputera miałby nieskończoną rozdzielczość, dalej nie byłby w stanie rysować chociażby najprostszych figur czy krzywych – aby narysować krzywą ciągłą, potrzebna jest wyrysowana **nieprzeliczalna liczba** punktów. Natomiast maszyna Turinga może wykonywać tylko **przeliczalną** liczbę operacji. Dodatkowo, wśród liczb rzeczywistych znajduje się nieskończenie wiele liczb nieobliczalnych. To wiąże się z kolejnym problemem, który opisany został niżej.

Wniosek

Komputer nie potrafi operować danymi ciągłymi.

Jak już stwierdziliśmy, maszyna licząca posługuje się przeliczalną liczbą danych. Dane ciągłe są nieprzeliczalne i jedyną opcją dla komputera jest próbkowanie danych i zapisywanie ich w sposób dyskretny, ale dostatecznie gęsty. Z powodu tej wady komputerów niektórzy audiofile unikają urządzeń cyfrowych i kurczowo trzymają – według nich komputer cyfrowy nie jest w stanie zasymulować sygnału analogowego nawet przy dużej gęstości danych.

Wniosek

Niektóre problemy dla komputera są bardzo złożone, podczas gdy dla człowieka uznawane są za łatwe.

Rozpoznawanie kształtów w przestrzeni jest czymś, z czym człowiek żyje i ma się dobrze. Jednak dla komputera jest niebotycznie złożonym procesem obliczeniowym. Komputer nie potrafi myśleć skojarzeniami, selektywnie wybierać tylko ważne bez analizy wszystkich danych (skąd ma wiedzieć, które są ważne, jeżeli ich nie sprawdzi uprzednio?). Obecnie komputer jest też kompletnie niezdolny do „zabawy w psychologa" i oceny zachowań ludzkich nieobarczonej poważnymi błędami pomiarowymi. Zatem istnieje pewna klasa problemów, w których nawet niezbyt rozgarnięty człowiek może czuć się lepszy od dzisiejszych supermaszyn. Kluczowym elementem, który do tego się przyczynia jest możliwość nauki i powiększania swoich możliwości – algorytmy komputera są sztywne i z góry zaplanowane, a więc ich proces ewentualnej nauki może odbywać się tylko w zaplanowanych ramach.

Wniosek

Komputer nie jest w stanie oceniać cech subiektywnych (czy kwiatek jest „ładny", czy film jest „dobry"), jeżeli algorytm operuje tylko obiektywnymi kryteriami.

To wydaje się dosyć oczywiste, że z komputera ciężko uczynić duszę towarzystwa, z którą można by podyskutować nad czymś innym niż fakty. Co prawda, możemy zaprogramować go tak, by za fakty uznawał także konkretne opinie lub szereg opinii, np. „kwiatki są ładne", ale nie będzie to opinia nabyta na podstawie doświadczeń. Poza zdolnością nauki, do tego niezbędna byłaby umiejętność klasyfikowania rzeczy jako „złe" lub „dobre", a więc namiastka moralności.

Wniosek

Część problemów człowiek rozwiązuje jak maszyna Turinga.

Zadania „szkolne" często wykonujemy według ustalonego algorytmu. Z dokładnością do ludzkich pomyłek i w dużym uproszczeniu możemy stwierdzić, że też zachowujemy się jak maszyna Turinga.

2.1 Formalizacja

Głównym „problemem" komputera jest, że nie potrafi operować danymi, które nie są ściśle zdefiniowane. Wszystkie elementy algorytmu muszą być dokładnie sformalizowane, w innym wypadku poprawny dla dobrze określonych danych algorytm może być wadliwy, tj. zwrócić nieprawidłową odpowiedź bądź zapętlić

się. Wyobraźmy sobie, że każdy element rzeczywistości człowiek musiałby ująć matematycznie, by móc chociażby go zanalizować, a co dopiero przekształcić. Załóżmy jednak, że zechcemy podjąć się formalizacji tylko jednej ze sfer życia człowieka – obrazu. Mocno upraszczając, uznajmy, że człowiek odbiera obraz, rozróżniając jedynie kolory. Jednym z prostszych sposób jest zapisanie koloru jako wektora.

$$c = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

gdzie R, G, B to składowe koloru – czerwony, zielony oraz niebieski, a $[0,0,0]$ to całkowita ciemność, $[1,1,1]$ – oślepiająca biel. Każdy kolor możemy dzięki temu jednoznacznie opisać odpowiednim wektorem. Będziemy musieli przyjąć teraz kolejne uproszczenie, mianowicie, że człowiek odbiera i zapamiętuje otoczenie tak samo, jak monitor wyświelta obraz – w pewnym prostokącie o pewnej rozdzielczości pojedynczych punktów z przypisanym kolorem (pikseli), niech to będzie $n \times m$. A zatem, nasz obraz jest ciągiem punktów (wektorów):

$$(x, y, c)$$

gdzie (x, y) są współrzędnymi na tym prostokącie, a c jest przypisanym kolorem. Informatycznie będzie to pewna tablica postaci:

$$c[x, y, i]$$

gdzie x, y są współrzędnymi, natomiast i od 1 do 3 jest składową koloru. Skonstruujmy prosty algorytm, który porówna dwa obrazy. Danymi wejściowymi będzie oczywiście obraz wzorcowy i porównywany, a rezultatem liczba określająca stopień zbieżności obrazów. Będziemy porównywać piksel po pikselu, jak bardzo różnią się kolory między obrazami na miejscu (x, y). Następnie zsumujemy tę różnicę w następujący sposób:

$$\sum_{x=1}^{n}\sum_{y=1}^{m} a_{xy}, \quad a_{xy} = |c1[x,y,1] - c2[x,y,1]| + |c1[x,y,2] - c2[x,y,2]| + |c1[x,y,3] - c2[x,y,3]|$$

```
A = 0;
DLA x OD 1 DO n
  DLA y OD 1 DO m
    A += |c1[x,y,1]-c2[x,y,1]|+|c1[x,y,2]-c2[x,y,2]|+|c1[x,y,3]-c2[x,y,3]|
```

Wynikiem będzie jakaś liczba określająca stopień podobieństwa, im mniejsza – tym lepiej. Ten algorytm jest bardzo wadliwy, wystarczy przesunąć obraz, przeskalować lub skorygować barwy, by pozornie te same obrazy okazały się dla programu różne. Co prawda, da się lekko zmodyfikować ten algorytm i uodpornić go na takie zabiegi, tym niemniej stopień trafności tego porównania będzie niski. To pokazuje, jak ciężko jest modelować algorytm symulujący tak błahą dla człowieka czynność jaką jest porównywanie obrazu.

3 Zakończenie

Jak widać, maszyny Turinga obarczone są sporymi ograniczeniami. Aby móc je ominąć, rozważa się istnienie tzw. **hiperkomputerów**, tj. maszyn o większych możliwościach. Jednakże do tej pory nie wiadomo, czy prawa fizyki pozwalają na skonstruowanie „silniejszych" komputerów niż maszyny Turinga. Największe nadzieje wiąże się z komputerami kwantowymi i komputerami opartymi na sieciach neuronowych.

Pytanie

Czy umysł ludzki funkcjonuje jak pewna specyficzna maszyna Turinga, tyle że wyposażone w inne struktury algorytmiczne?