CSE 417: Algorithms and Computational Complexity

Winter 2001

Lecture 18

Instructor: Paul Beame

1

Turing Machines

- Church-Turing Thesis
  - Any reasonable model of computation that includes all possible algorithms is equivalent in power to a Turing machine
- Evidence
  - Huge numbers of equivalent models to TM's based on radically different ideas

2

Universal Turing Machine

- A Turing machine interpreter U
  - On input the code of a program (or Turing machine) P and an input x, U outputs the same thing as P does on input x
  - Basis for modern stored-program computer
- Notation:
  - We'll write <P> for the code of program P and <P,x> for the pair of the program code and input

3

Halting Problem

- Given: the code of a program P and an input x for P, i.e. given <P,x>
- Output: 1 if P halts on input x and 0 if P does not halt on input x
- Theorem (Turing): There is no program that solves the halting problem

  "The halting problem is undecidable"

4

Proof ideas: Countability (Cantor 1875)

- Defn: A set S is countable iff there is a function mapping the natural numbers N onto S.
  - i.e. we can write $S=\{s_1, s_2, s_3, \ldots\}$, i.e. $f(i)=s_i$
- All finite sets are countable.
- The natural numbers are countable
- The integers are countable
  - Z={0,1,-1,2,-2,3,-3,4,-4,5,-5,...}
  - 1 2 3 4 5 6 7 8 9 10 11 ...

5

Countability

- The set of all finite strings with any alphabet is countable,
  - e.g. binary strings

    1 2 3 4 5 6 7 8 9 10 11 12

  - $S=\{\underbrace{\varepsilon}_{0}, \underbrace{0, 1}_{1}, \underbrace{00, 01, 10, 11}_{2}, \underbrace{000, 001, 010, 011, 100}_{3}, \ldots\}$
  - i.e. list strings in order of increasing length
- Any program code is a string and given any string w, we can interpret w as a program (syntactically incorrect programs are just no-ops) so the set of all programs is countable, too.

6

Uncountability

- The set of all functions f from the natural numbers N to {0,1} is not countable.
- Suppose it were and we had a list of all such functions $\{f_1, f_2, f_3, \ldots\}$
- We build an infinite table of these functions

7

function \ input	1	2	3	4	5	6	7	8	9	10	11	12	....
f_1	0	1	1	0	1	1	1	0	0	0	1	1	....
f_2	1	1	0	1	0	1	1	0	1	1	1	0	....
f_3	1	0	1	0	0	0	0	0	0	0	1	1	....
f_4	0	1	1	0	1	0	1	1	0	1	0	1	....
f_5	0	1	0	0	1	1	1	0	0	0	1	1	....
f_6	1	1	0	1	1	1	1	0	1	1	1	0	....
f_7	1	0	1	0	0	1	0	0	0	0	1	1	....
f_8	0	1	1	0	0	0	1	1	0	1	0	1	....
.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.

8

function \ input	1	2	3	4	5	6	7	8	9	10	11	12	....
f_1	0	1	1	0	1	1	1	0	0	0	1	1	....
f_2	1	1	0	1	0	1	1	0	1	1	1	0	....
f_3	1	0	1	0	0	0	0	0	0	0	1	1	....
f_4	0	1	1	0	1	0	1	1	0	1	0	1	....
f_5	0	1	0	0	1	1	1	0	0	0	1	1	....
f_6	1	1	0	1	1	1	1	0	1	1	1	0	....
f_7	1	0	1	0	0	1	0	0	0	0	1	1	....
f_8	0	1	1	0	0	0	1	1	0	1	0	1	....
.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.

9

function \ input	1	2	3	4	5	6	7	8	9	10	11	12	....
f_1	1	1	1	0	1	1	1	0	0	0	1	1	....
f_2	1	0	0	1	0	1	1	0	1	1	1	0	....
f_3	1	0	0	0	0	0	0	0	0	0	1	1	....
f_4	0	1	1	1	1	0	1	1	0	1	0	1	....
f_5	0	1	0	0	0	1	1	0	0	0	1	1	....
f_6	1	1	0	1	1	0	1	0	1	1	1	0	....
f_7	1	0	1	0	0	1	1	0	0	0	1	1	....
f_8	0	1	1	0	0	0	1	0	0	1	0	1	....
.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.

10

Diagonalization

- Define function D from N to {0,1} such that
  - $D(i)=1-f_i(i)$
  - i.e. we flipped the diagonal elements
- D must be different from every function in the list since D differs from f_i on input i
- Contradicts our assumption that the list had all such functions!
- Corollary: There is some function f from N to {0,1} not computed by any program
  - more functions than programs!

11

Undecidability of the Halting Problem

- Suppose that there is a program H that computes the answer to the Halting Problem
- We'll build a similar table with all the possible programs down one side and all the possible inputs along the other and do a diagonal flip to produce a contradiction

12

input

program code	ε	0	1	00	01	10	11	000	001	010	011	....
ε	0	1	1	0	1	1	1	0	0	0	1	....
0	1	1	0	1	0	1	1	0	1	1	1	....
1	1	0	1	0	0	0	0	0	0	0	1	....
00	0	1	1	0	1	0	1	1	0	1	0	....
01	0	1	1	1	1	1	1	0	0	0	1	....
10	1	1	0	0	0	1	1	0	1	1	1	....
11	1	0	1	1	0	0	0	0	0	0	1	....
000	0	1	1	1	1	0	1	1	0	1	0	....
001	.	.	.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	.	.	.	
.												

Entries are 1 if program P given by the code halts on input x and 0 if it runs forever

13

input

program code	ε	0	1	00	01	10	11	000	001	010	011	....
ε	1	1	1	0	1	1	1	0	0	0	1	....
0	1	0	0	1	0	1	1	0	1	1	1	....
1	1	0	0	0	0	0	0	0	0	0	1	....
00	0	1	1	1	1	0	1	1	0	1	0	....
01	0	1	1	1	0	1	1	0	0	0	1	....
10	1	1	0	0	0	0	1	0	1	1	1	....
11	1	0	1	1	0	0	1	0	0	0	1	....
000	0	1	1	1	1	0	1	0	0	1	0	....
001	.	.	.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	.	.	.	
.												

Want to create a new program whose halting properties are given by the flipped diagonal

14

Diagonal construction

- Suppose H exists
- Now define a new program D such that
  - D on input x:
    - runs H checking if the program P whose code is x halts when given x as input; i.e. does P halt on input <P>
    - if H outputs 1 then D goes into an infinite loop
    - if H outputs 0 then D halts.

15

Finishing the argument

- D must be different from any program in the list.

- Suppose it has code <D>
  - then D halts on input <D> iff
  - H outputs 0 given program D and input <D> iff
  - P runs forever on input <P>

16