CS 7220 – Computational Complexity and Algorithm Analysis

Spring 2016

Section 7: Computability – Part I
Introduction

Pascal Hitzler

Data Semantics Laboratory

Wright State University, Dayton, OH

http://www.pascal-Hitzler.de

Models of computation

- Generally, abstract from space/memory limitations
  - Assume memory is “as large as needed”
- Ignore, how long a computation takes
  - as long as it terminates in finite time.
- Often, use only numbers/integers or only (finite) strings as the things which are computed/stored in memory.
- There exist many formal models of computation.

Models of Computation

- Turing Machine (in this lecture – at the beginning)
- μ-Recursive functions (in this lecture – towards the end)
- λ-calculus (see functional programming)
- Unlimited Register Machine
- WHILE-language
- … many others …

Unlimited Register Machine (URM)

- Registers r_1, r_2, r_3, …
  holding non-negative integers

- Initialization: finite number of registers ≠ zero

- A program consists of a finite sequence of instructions.

- Available instructions:
  - Zero Z(n): set register r_n to 0
  - Successor S(n): increase r_n by 1
  - Transfer T(m,n): copy r_m to r_n
  - Jump J(m,n,p): If $r_m = r_n$, jump to instruction number p

WHILE-language

- Minimal programming language, essentially consisting of
  - Elementary arithmetic +, -, *, /
  - Boolean comparison of numbers: <, >, =, , , ≠
  - Logical AND, OR, NOT
  - Assignment of values to variables
  - WHILE loops as only control features

Are they different?

- Not really.
- All models with certain minimal capabilities have so far been shown to be equivalent.
- This is actually quite remarkable!

Uncomputable example

- **N: Natural numbers (non-negative integers): N = {0, 1, 2, 3, 4, …}**

- **P(N): set of all subsets of N**
  Examples:
  - **{0,1,2,3,4,…}**
  - **{}**
  - **{0,2,4,6,8,…}**
  - **{2,3,267,1011}**
  - **{0,1,2,3,5,8,13,21,34,…}**
  - **{2,3,5,7,11,13,17,19,23,…}**

Uncomputable example

- We say that an algorithm (in some model of computation) computes a subset S of N if
  - It outputs a stream of non-negative integers (strictly increasing).
  - It needs only finite time between two outputs.
  - If does not skip any number in S.
  - All output numbers are in S.
  - If it terminates, then it has output all integers in S.

Question: Can every set in P(N) be computed?

Uncomputable example

- Every algorithm which computes a subset of N can be expressed with a finite string.

- It is easy to define a strict order on the set of all algorithms.
  - E.g. lexicographic order.
  - E.g. convert them to bit strings and sort by binary number.

- Hence, we can assume that $\{A_0,A_1,A_2,A_3,\ldots\}$ is the set of all algorithms computing subsets of N.

Uncomputable example

Mark the output of each A_i:

	0	1	2	3	4	5	6	7	8	…
A_0		x			x	x		x		
A_1		x	x		x		x		x	
A_2	x		x	x	x			x		
A_3		x		x					x	
A_4	x	x	x		x		x	x		
A_5	x			x	x			x		
A_6		x				x			x	
…										

Uncomputable example

Now make a new subset of N by "inverting" the diagonal:

	0	1	2	3	4	5	6	7	8	…
A_0		x			x	x		x		
A_1		x	x		x		x		x	
A_2	x		x	x	x			x		
A_3		x		x					x	
A_4	x	x	x		x		x	x		
A_5	x			x	x			x		
A_6		x				x			x	
…								…		

Result: x x x

i.e. { 0, 5, 6, … }

Uncomputable example

The resulting set is not computed by any A_i!

	0	1	2	3	4	5	6	7	8	…
A_0		x			x	x		x		
A_1		x	x		x		x		x	
A_2	x		x	x	x			x		
A_3		x		x					x	
A_4	x	x	x		x		x	x		
A_5	x			x	x			x		
A_6		x				x			x	
…								…		

Result: x x x

i.e. { 0, 5, 6, … }

A_5 doesn't compute it!

Uncomputable example

The resulting set is not computed by any A_i!

	0	1	2	3	4	5	6	7	8	…
A_0		x			x	x		x		
A_1		x	x		x		x		x	
A_2	x		x	x	x			x		
A_3		x		x					x	
A_4	x	x	x		x		x	x		
A_5	x			x	x			x		
A_6		x				x			x	
…								…		

but we have all possible algorithms in the list!

Hence: we found a set which is not computable!

Looking a bit deeper

- The set of all algorithms is *countable*.
  (I.e., can be enumerated as A_0, A_1, A_2, …)
- The set P(N) is *uncountable*.
  (I.e., can*not* be enumerated as S_0, S_1, S_2, …)
  - Essentially the same proof. With a slight twist.

- This proof technique is known as "diagonalization."
  - We will need the technique for the main result in this lecture.
  - It is usually credited to Georg Cantor (1845–1918); at least he was the first to publish the diagonalization proof that P(N) is uncountable).

Exercise C1

- **Adjust the proof just given such that you prove the following:**

The set of real numbers is uncountable.

Exercise C2 (hand-in)

Show that there are languages which are not recursively enumerable.

Hint: Use diagonalization. It is possible to adjust the proof given earlier, that not all sets of non-negative integers can be computed. You do not need to spell out all details, but the argument must be convincing.