CS 410/610, MTH 410/610
Theoretical Foundations of Computing

Fall Quarter 2010

Slides 1

Pascal Hitzler
Kno.e.sis Center
Wright State University, Dayton, OH
http://www.knoesis.org/pascal/

Today's Session

1. Discussion: What is "computable?"
2. Uncomputable – an example
3. Lecture overview
4. This lecture in the context of others
5. Organizational matters

Which things can be computed?

Which things cannot be computed?

What exactly is “computation”?

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: <, >, =, $\leq$, $\geq$, $\neq$
  - 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!**

Today's Session

1. Discussion: What is "computable?"
2. Uncomputable – an example
3. Lecture overview
4. This lecture in the context of others
5. Organizational matters

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 (0) x (5) x (6)

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 1 (hand-in)

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

The set of all subsets of N is uncountable.

Today's Session

1. Discussion: What is "computable?"
2. Uncomputable – an example
3. Lecture overview
4. This lecture in the context of others
5. Organizational matters

Lecture overview

- **Model of computation: Turing Machines**

- **The Church-Turing Thesis**
  (“Turing Machines are universal computers.”)

- **A famous uncomputable problem: The Halting problem**
  (“There is no algorithm which can check for all other algorithms whether they will terminate”)

- **Another model of computation: μ-Recursive Functions**

Today’s Session

1. Discussion: What is “computable?”
2. Uncomputable – an example
3. Lecture overview
4. This lecture in the context of others
5. Organizational matters

This lecture in the context of others

- Three parts to “Foundations of Computing”
  - Formal Languages and Automata (Sudkamp Part II)
  - This lecture: Computability Theory (Sudkamp Part III)
  - Computational Complexity (Sudkamp Part IV)

- Key outcomes from this lecture
  - Turing Machines, Church-Turing Thesis, Halting Problem
  - Practice dealing with formal (mathematical) notions and techniques
  - Learning to be formally precise
  - Understand the fundamental limitations of computing

The problem with abstraction

Is the following true?

No C are B.

All B are A.

Therefore, some A are not C.

The problem with abstraction

Is the following true?

No flying things are penguins.

All penguins are birds.

Therefore, some birds are not fliers.

[Example taken from Newsweek August 16, 2010, page 24: “The Limits of Reason” by Sharon Begley.]

Today's Session

1. Discussion: What is "computable?"
2. Uncomputable – an example
3. Lecture overview
4. This lecture in the context of others
5. Organizational matters

Organizational Matters

- **Office Hours: Thursdays 2-3, Joshi 389.**
  Email contact preferred.
  No office hour on 9th of September!

- **Textbook (required):**
  Thomas A. Sudkamp, Languages and Machines, Third Edition, Addison Wesley, 2006.

- **Grading:**
  Midterm exam: 30%
  Final exam: 50%
  Exercises: 20%
  (I may adjust in your favor)

Organizational Matters

- **We will frequently make exercise sessions.**
  You will get exercises, to be done at home. Some will be marked as graded by me (hand-in exercises). They are all discussed afterwards in class.
  Each exercise counts 5 points.
  An average of 4 points counts as 100% for the exercises.
  Exercises are due *one week after I pose them – at the beginning of the class*.

- **Webpage/slides.**
  I prefer to use a public website:
  http://knoesis.wright.edu/faculty/pascal/teaching/s10/complexity.html

Organizational Matters

- I will be absent
  - Sept. 9th (substitute: Prof. TK Prasad)
  - Sept. 21st and 23rd
  - Nov. 9th (week before exams)
  - How do we make up for this?
    - Extra sessions on Fridays, 6pm-7:40pm
      - September 17th
      - October 1st
      - October 8th
    - Room to be determined.

Course overview

Tentative

We cover most of chapters 8, 9,   11, 12, 13

- **Schedule**
  - **Week 1: Introduction; Turing Machines**
  - **Week 2: Turing Machines continued**