Fall 2013
Lecture 27: Turing machines and decidability

highlights

• Cardinality
  • A set S is countable iff we can write it as $S = \{s_1, s_2, s_3, \ldots\}$ indexed by $\mathbb{N}$
  • Set of rationals is countable
    – “dovetailing”
  • $\Sigma^*$ is countable
    – $\{0,1\}^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, \ldots\}$
  • Set of all (Java) programs is countable

what about the real numbers?

Q: Is every set is countable?

A: Theorem [Cantor] The set of real numbers (even just between 0 and 1) is NOT countable

Proof is by contradiction using a new method called diagonalization

proof by contradiction

• Suppose that $\mathbb{R}^{(0,1)}$ is countable
• Then there is some listing of all elements $\mathbb{R}^{(0,1)} = \{r_1, r_2, r_3, r_4, \ldots\}$
• We will prove that in such a listing there must be at least one missing element which contradicts statement “$\mathbb{R}^{(0,1)}$ is countable”
• The missing element will be found by looking at the decimal expansions of $r_1, r_2, r_3, r_4, \ldots$
real numbers between 0 and 1: $\mathbb{R}^{[0,1)}$

- Every number between 0 and 1 has an infinite decimal expansion:
  
  $1/2 = 0.\overline{5}$
  $1/3 = 0.\overline{3}$
  $1/7 = 0.\overline{142857}$
  $\pi \approx 0.\overline{141592}$
  $1/5 = 0.\overline{2}$

Representations of real numbers as decimals

Representation is unique except for the cases that decimal ends in all 0’s or all 9’s.

$x = 0.\overline{19999}$
$10x = 1.\overline{9999}$
$9x = 1.8$ so $x = 0.\overline{2}$

Won’t allow the representations ending in all 9’s

All other representations give different elements of $\mathbb{R}^{[0,1)}$

supposed listing of $\mathbb{R}^{[0,1)}$

<table>
<thead>
<tr>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
<th>$r_4$</th>
<th>$r_5$</th>
<th>$r_6$</th>
<th>$r_7$</th>
<th>$r_8$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.3</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.7</td>
<td>0.6</td>
<td>...</td>
</tr>
</tbody>
</table>

supposed listing of $\mathbb{R}^{[0,1)}$

<table>
<thead>
<tr>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
<th>$r_4$</th>
<th>$r_5$</th>
<th>$r_6$</th>
<th>$r_7$</th>
<th>$r_8$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.3</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.7</td>
<td>0.6</td>
<td>...</td>
</tr>
</tbody>
</table>

...
**flipped diagonal**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>r</strong></td>
<td>0.</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Flipping Rule:</strong></td>
<td>If digit is 5, make it 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>r</strong></td>
<td>0.</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td><strong>r</strong></td>
<td>0.</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td><strong>r</strong></td>
<td>0.</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td><strong>r</strong></td>
<td>0.</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td><strong>r</strong></td>
<td>0.</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td><strong>r</strong></td>
<td>0.</td>
<td>7</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td><strong>r</strong></td>
<td>0.</td>
<td>6</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>4</td>
</tr>
</tbody>
</table>
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ...

**flipped diagonal number D**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D</strong></td>
<td>0.</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D</strong> is in $\mathbb{R}^{[0,1)}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>But for all $n$, we have $D \neq r_n$ since they differ on $n^{th}$ digit (which is not 0 or 9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Rightarrow$ list was incomplete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Rightarrow$ $\mathbb{R}^{[0,1)}$ is not countable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**why doesn’t this show that the rationals aren’t countable?**

- The set of real numbers is not countable

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$D_1$</strong></td>
<td>0.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td><strong>$D_2$</strong></td>
<td>0.</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td><strong>$D_3$</strong></td>
<td>0.</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td><strong>$D_4$</strong></td>
<td>0.</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>2</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td><strong>$D_5$</strong></td>
<td>0.</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td><strong>$D_6$</strong></td>
<td>0.</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td><strong>$D_7$</strong></td>
<td>0.</td>
<td>7</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td><strong>$D_8$</strong></td>
<td>0.</td>
<td>6</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>4</td>
</tr>
</tbody>
</table>
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ...

**flipped diagonal number D**

- The set of all functions $f : \mathbb{N} \rightarrow \{0,1,...,9\}$ is not countable
• There exist functions that cannot be computed by any program
  – The set of all functions $f : \mathbb{N} \rightarrow \{0,1,...,9\}$ is not countable
  – The set of all (Java/C/C++) programs is countable
  – So there are simply more functions than programs

• Are any of these functions, ones that we would actually want to compute?
  – The argument does not even give any example of something that can’t be done, it just says that such an example exists

• We haven’t used much of anything about what computers (programs or people) can do
  – Once we figure that out, we’ll be able to show that some of these functions are really important

before Java...more from our brief history of reasoning

• 1930’s
  – How can we formalize what algorithms are possible?
    Turing machines (Turing, Post)
    basis of modern computers
    Lambda Calculus (Church)
    basis for functional programming
    $\mu$-recursive functions (Kleene)
    alternative functional programming basis

Church-Turing Thesis
Any reasonable model of computation that includes all possible algorithms is equivalent in power to a Turing machine

• Evidence
  – Intuitive justification
  – Huge numbers of equivalent models to TM’s based on radically different ideas
components of Turing’s intuitive model of computation

• **Finite Control**
  – Brain/CPU that has only a finite # of possible “states of mind”

• **Recording medium**
  – An unlimited supply of blank “scratch paper” on which to write & read symbols, each chosen from a finite set of possibilities
  – Input also supplied on the scratch paper

• **Focus of attention**
  – Finite control can only focus on a small portion of the recording medium at once
  – Focus of attention can only shift a small amount at a time

what is a Turing machine?

• **Recording medium**
  – An infinite read/write “tape” marked off into cells
  – Each cell can store one symbol or be “blank”
  – Tape is initially all blank except a few cells of the tape containing the input string
  – Read/write head can scan one cell of the tape - starts on input

• In each step, a Turing machine
  – Reads the currently scanned symbol
  – Based on state of mind and scanned symbol
    - Overwrites symbol in scanned cell
    - Moves read/write head left or right one cell
    - Changes to a new state

• Each Turing Machine is specified by its finite set of rules

sample Turing machine

<table>
<thead>
<tr>
<th></th>
<th>_</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁</td>
<td>(1,s₂)</td>
<td>(1,s₂)</td>
<td>(0,s₂)</td>
</tr>
<tr>
<td>s₂</td>
<td>(H,s₃)</td>
<td>(R,s₁)</td>
<td>(R,s₁)</td>
</tr>
<tr>
<td>s₃</td>
<td>(H,s₃)</td>
<td>(R,s₁)</td>
<td>(R,s₁)</td>
</tr>
</tbody>
</table>

_ _ 1 1 0 1 1 _ _
**what is a Turing machine?**

**turing machine ≡ ideal Java/C program**

- Ideal C/C++/Java programs
  - Just like the C/C++/Java you’re used to programming with, except you never run out of memory
    - Constructor methods always succeed
    - `malloc` never fails
- Equivalent to Turing machines except a lot easier to program!
  - Turing machine definition is useful for breaking computation down into simplest steps
  - We only care about high level so we use programs

**turing's big idea: machines as data**

- Original Turing machine definition
  - A different “machine” $M$ for each task
  - Each machine $M$ is defined by a finite set of possible operations on finite set of symbols
    - $M$ has a finite description as a sequence of symbols, its “code”
- You already are used to this idea:
  - We’ll write $<P>$ for the code of program $P$
  - i.e. $<P>$ is the program text as a sequence of ASCII symbols and $P$ is what actually executes

**turing’s idea: a universal turing machine**

- A Turing machine interpreter $U$
  - On input $<P>$ and its input $x$, $U$ outputs the same thing as $P$ does on input $x$
  - At each step it decodes which operation $P$ would have performed and simulates it.
- One Turing machine is enough
  - Basis for modern stored-program computer
  - Von Neumann studied Turing’s UTM design
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 $x$
  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”

proof by contradiction

• Suppose that $H$ is a Turing machine that solves the Halting problem
  
  Function $D(x)$:
  
  ```
  if $H(x,x)=1$ then
    while (true); /* loop forever */
  else
    no-op; /* do nothing and halt */
  endif
  ```

• What does $D$ do on input $<D>$?
  – Does it halt?

proof by contradiction

• Does $D$ halt on input $<D>$?

$D$ halts on input $<D>$

$\iff H$ outputs 1 on input $(<D>,<D>)$

[since $H$ solves the halting problem and so $H(<D>,x)$ outputs 1 iff $D$ halts on input $x$]

$\iff D$ runs forever on input $<D>$

[since $D$ goes into an infinite loop on $x$ iff $H(x,x)=1$]

This is contradiction. Our only assumption was that $H$ exists.

that’s it!

• We proved that there is no computer program that can solve the Halting Problem.

• This tells us that there is no compiler that can check our programs and guarantee to find any infinite loops they might have.