Computational Models - Lecture 7
Fall 04/05

- $R \subseteq RE \cap coRE$.

- The Church-Turing Thesis (again)

- David Hilbert’s Tenth Problem

- Universal Turning machine

- The halting problem

- Sipser’s book, 3.3, & 4.1, 4.2
A Short Midterm Review First
Decidability vs. Enumerability

- **Decidability** is a **stronger notion** than enumerability.

- If a language \( L \) is **decidable** then clearly it is **enumerable** (the other direction does **not** hold, as we’ll show in a couple of lectures).

- It is also clear that if \( L \) is **decidable** then so is \( \overline{L} \), and thus \( \overline{L} \) is also **enumerable**.

- Let \( \mathcal{RE} \) denote the class of enumerable languages, and let \( \text{co}\mathcal{RE} \) denote the class of languages whose **complement** is enumerable.

- Let \( \mathcal{R} \) denote the class of decidable languages. Then what we just saw is \( \mathcal{R} \subseteq \mathcal{RE} \cap \text{co}\mathcal{RE} \).
Decidability vs. Enumerability (2)

**Theorem:** \( R = \text{RE} \cap \text{coRE} \).

**Proof:** We should prove the \( \supseteq \) direction. Namely if \( L \in \text{RE} \cap \text{coRE} \), then \( L \in R \).

In other words, if both \( L \) and its complement are enumerable, then \( L \) is decidable.

Let \( M_1 \) be a TM that accepts \( L \).

Let \( M_2 \) be a TM that accepts \( \overline{L} \).

We describe a TM, \( M \), that decides \( L \).

On input \( x \), \( M \) runs \( M_1 \) and \( M_2 \) in parallel.

If \( M_1 \) accepts, \( M \) accepts.

If \( M_2 \) accepts, \( M \) rejects.

Should now show that indeed \( M \) decides \( L \).
Reformulation

**Theorem:** A language is decidable if and only if it is both enumerable and co-enumerable.

**Proof:** We must prove two directions:

- If $L$ is decidable, then both $L$ and $\overline{L}$ are enumerable.
- If $L$ and $\overline{L}$ are both enumerable, then $L$ is decidable,
One Direction

Claim: If $L$ is decidable, then both $L$ and $\overline{L}$ are enumerable.

Proof: Pop quiz!
Other Direction

**Claim:** If $L$ and $\overline{L}$ are both enumerable, then $L$ is decidable,

- Let $M_1$ be the acceptor for $L$, and
- $M_2$ the acceptor for $\overline{L}$.

$M =$ On input $w$

1. Run both $M_1$ and $M_2$ in parallel.
2. If $M_1$ accepts, accept; if $M_2$ accepts, reject.
Parallel?

**Question:** What does it mean to run $M_1$ and $M_2$ in parallel?

- $M$ has two tapes
- $M$ alternates taking steps between $M_1$ and $M_2$. 
Claim

We claim that $M$ decides $L$.
- Every string is in $L$ or $\overline{L}$.
- Either $M_1$ or $M_2$ accepts input $w$.
- Because $M$ halts whenever $M_1$ or $M_2$ accepts $M$ always halts, and is a decider.
- Moreover, $M$ accepts strings in $L$ and rejects strings in $\overline{L}$.

Therefore, $M$ decides $L$, so $L$ is decidable.
Revised View of the World of Languages

- Regular
- Context Free
- Decidable
- Enumerable

Slides modified by Benny Chor, based on original slides by Maurice Herlihy, Brown Univ.
Church-Turing Thesis

Formal notions appeared in 1936:

- $\lambda$-calculus of Alonzo Church
- Turing machines of Alan Turing
- Counter machines
- Unrestricted grammars
- Two stack automata
- Random access machines (RAMs)

These definitions look very different, but are provably equivalent.
Church-Turing Thesis

These definitions look very different, but are provably equivalent.

The Church-Turing Thesis:

“The intuitive notion of reasonable models of computation equals Turing machine algorithms”.

Wild Models

What about “wild” models of computation?
Consider MUntel’s $\aleph$-AXP2© processor (to be released XMAS 2004).

- Like a Turing machine, except
- Takes first step in 1 second.
- Takes second step in $1/2$ second.
- Takes $i$-th step in $2^{-i}$ seconds . . .

After 2 seconds, the $\aleph$-AXP© decides any enumerable language!

**Question:** Does the $\aleph$-AXP© invalidate the Church-Turing Thesis?
Hilbert’s 10th Problem

In 1900, David Hilbert delivered a now-famous address at the International Congress of Mathematicians in Paris, France.

- Presented **23 central mathematical problems**
- challenge for the next (20th) century
- the **10th problem** directly concerned algorithms

**November 2003**: significant progress on the 6th problem.

But for us, start with some background on the **10th ...**
Polynomials

- A term is a product of variables and a constant coefficient, e.g. $6x^3yz^2$.

- A polynomial is a sum of terms, e.g. $6x^3yz^2 + 3xy^2 - x^3 - 10$.

- A root of a polynomial is an assignment of values to variables so that the polynomial equals zero.

- For example, $x = 5$, $y = 3$, and $z = 0$ is a root of the polynomial above.

- Here, we are interested in integral roots, namely an assignment of integers to all variables.

- Some polynomials have integral roots, some don’t (e.g. $x^2 - 2$).
Hilbert’s Tenth Problem

The Problem: Devise an algorithm that tests whether a polynomial has an integral root.

Actually, what he said (translated from German) was “to devise a process according to which it can be determined by a finite number of operations”.

Note that

- Hilbert explicitly asks that algorithm be “devised”
- apparently Hilbert assumes that such an algorithm must exist, and someone “only” need find it.
Hilbert’s Tenth Problem

- We now know no algorithm exists for this task.
- Mathematicians of 1900 could not have proved this, because they didn’t have a formal notion of an algorithm.
- Intuitive notions work fine for constructing algorithms (we know one when we see it).
- Formal notions are required to show that no algorithm exists.
Hilbert’s Tenth Problem

In 1970, 23 years old Yuri Matijasevič, building on work of Martin Davis, Hilary Putnam, and Julia Robinson, proved that no algorithm exists for testing whether a polynomial has integral roots

(a survey of the proof)
Reformulating Hilbert’s Tenth Problem

Consider the language:

\[ D = \{ p \mid p \text{ is a polynomial with an integral root} \} \]

Hilbert’s tenth problem asks whether this language is decidable.

We now know it is not decidable, but it is enumerable!
Univariate Polynomials

Consider the simpler language:

\[ D_1 = \{p \mid p \text{ is a polynomial over } x \text{ with an integral root} \} \]

Here is a Turing machine that accepts \( D_1 \).

On input \( p \),

- evaluate \( p \) with \( x \) set successively to \( 0, 1, -1, 2, -2, \ldots \)
- if \( p \) evaluates to zero, accept.
Univariate Polynomials (2)

\[ D_1 = \{ p \mid p \text{ is a polynomial over } x \text{ with an integral root} \} \]

Note that

- If \( p \) has an integral root, the machine accepts.
- If not, \( M_1 \) loops.
- \( M_1 \) is an acceptor, but not a decider.
Univariate Polynomials (3)

\[ f := x \rightarrow x^3 - 300x^2 + 10000x + 1000000 \]
\[ g := x \rightarrow 200x^2 - 2000x - 1000000 \]
\[ \text{plot([f(x), g(x)], x=-100..300, color=[red, blue], thickness=3);} \]
Univariate Polynomials (4)

In fact, $D_1$ is decidable.

Can show that all real roots of $p[x]$ lie inside interval

$$\left( -\frac{|kc_{max}|}{c_1}, \frac{|kc_{max}|}{c_1} \right),$$

where $k$ is number of terms, $c_{max}$ is max coefficient, and $c_1$ is high-order coefficient.

By Matijasevič theorem, such effective bounds on range of real roots cannot be computed for multivariable polynomials.
Encoding, and Universal TM
Encoding

- Input to a Turing machine is a string of symbols.
- But we want algorithms that work on graphs, matrices, polynomials, Turing machines, etc.
- Need to choose an encoding for objects.
- Can often be done in many reasonable ways.
- Sometimes distinguish between $X$, the object, and $\langle X \rangle$, its encoding.
Encoding

Consider strings representing undirected graphs.

A graph is connected if every node can be reached from any other node by traveling along edges.

Define the language:

\[ A = \{ \langle G \rangle \mid G \text{ is a connected undirected graph} \} \]
High-Level Description

High-level description of a machine that decides

\[ A = \{ \langle G \rangle \mid G \text{ is a connected undirected graph} \} \]

On input \( \langle G \rangle \), encoding of graph \( G \)

- select first node of \( G \) and mark it.
- repeat until no new nodes marked:
  - For each node in \( G \), mark if attached by an edge to a node already marked.
  - scan nodes of \( G \) to determine whether they are all marked. If so, accept, otherwise reject.
Some Details

**Question:** How is $G$ encoded?

![Graph](image)

**Answer:** List of nodes, followed by list of edges.
More Details

On input $M$ checks that input is valid graph encoding

- two lists
- first is list of numbers
- second is list of pairs
- first list contains no duplicates (element distinctness subroutine)
- every node in second list appears in first

Now ready to start “step one”.
Detailed Algorithm

On input $\langle G \rangle$, encoding of graph $G$

1. mark first node with a dot on leftmost digit.
2. loop:
   - Scans list and “underlines” undotted node $n_1$.
   - $M$ rescans and “underlines” dotted node $n_2$.
   - $M$ scans edges.
   - $M$ tests each edge if it is $(n_1, n_2)$.
   - If so, dot $n_1$, remove underlines, goto Step 2.
   - If not, check next edge. When no more edges, move underline to next dotted $n_2$.
   - when no more dotted vertexes, move underlines: new $n_1$ is next undotted node and new $n_1$ is first dotted node. Repeat Step 2. When no more undotted nodes, go to Step 3.
3. $M$ scans the list of nodes. If all dotted, accept, otherwise reject.
Univeral Turing Machines

We now define the universal Turing machine, $U$. On input $\langle M, w \rangle$, where $M$ is a TM and $w$ a string

1. Checks that $\langle M, w \rangle$ is a proper encoding of a TM, followed by a string from $\Sigma^*$. 
2. Simulates $M$ on input $w$. 
3. If $M$ on input $w$ enters its accept state, $U$ accept, and if $M$ on input $w$ ever enters its reject state, $U$ reject.

Notice that as a consequence, if $M$ on input $w$ enters an infinite loop, so does $U$ on input $\langle M, w \rangle$. 
Universal Turing Machines (2)

- The universal machine $U$ obviously has a fixed number of states (100 should do).
- Despite this, it can simulate machines $M$ with many more states.
- Most of you have seen a universal machine, and have even used one!
- For example, *Dr. Scheme* (interpreter) is a universal *Scheme* machine.
- It accepts a two part input: “Above the line” – the program (parallel to $\langle M \rangle$), and “below the line” the input to run it on (parallel to $w$).
- Universal machines inspired the development of stored-program computers in the 40s and 50s.
Halting Problem

One of the most philosophically important theorems of the theory of computation.

Computers (and computation) are not omnipotent – they are limited in a very fundamental way.

Many common problems are unsolvable, e.g.

- Does a program sort an array of integers?
- Problem is well defined: Both program and specification are precise mathematical objects.
- Hey, proving program \( \cong \) specification should be just like proving that triangle 1 \( \cong \) triangle 2 . . .
- Well, this is not the case!
CFG, NFA, DFA Reminders

Let $E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \}$

- We saw that $E_{CFG}$ is a decidable language.

- $A_{CFG} = \{ \langle M, w \rangle \mid \text{string } w \text{ is accepted by PDA } M \}$. Saw that the language $A_{CFG}$ is decidable.

- $A_{NFA} = \{ \langle M, w \rangle \mid \text{string } w \text{ is accepted by NFA } M \}$.

- $A_{DFA} = \{ \langle M, w \rangle \mid \text{string } w \text{ is accepted by DFA } M \}$

- Saw both $A_{NFA}$ and $A_{DFA}$ are also decidable.

- What would happen with Turing Machines?
Accepting Problem

Does a Turing machine accept a string?

\[ A_{TM} = \{ \langle M, w \rangle | M \text{ is a TM that accepts } w \} \]

**Theorem:** \( A_{TM} \) is undecidable.

Recall that the corresponding languages for DFAs, NFAs, and CFGs, namely \( A_{DFA} \), \( A_{NFA} \), and \( A_{CFG} \), are decidable.
The Acceptance Problem

\[ A_{TM} = \{ \langle M, w \rangle | M \text{ is a TM that accepts } w \} \]

Before approaching the proof of undecidability, we first prove

**Theorem:** \( A_{TM} \) is enumerable.

**Proof:** The universal machine accepts \( A_{TM} \). ♠
The Acceptance Problem

\[ A_{\text{TM}} = \{ \langle M, w \rangle | M \text{ is a TM that accepts } w \} \]

We prove \( A_{\text{TM}} \) is undecidable by diagonalization.

But first, a short “diagonalization reminder”.
Comparing Sizes of Sets

Suppose $A$ and $B$ are two sets, and we wish to compare their sizes.

If both $A$ and $B$ are finite, we can count how many elements each of them has, and compare the numbers.

This method does not generalize to infinite sets.
Comparing Sizes of Sets (2)

Alternatively, we can pair the elements of $A$ and $B$. If they pair perfectly, they have equal sizes.
Correspondence

**Question:** What does it mean to say that two infinite sets are the *same size*?

Answered by Georg Cantor in 1873: Pair them off.

A map \( f : A \rightarrow B \) is a *correspondence* if \( f \) satisfies
- \( f \) one-to-one: if \( a_1 \neq a_2 \) then \( f(a_1) \neq f(a_2) \).
- \( f \) onto: for every \( b \in B \), there is an \( a \in A \) such that \( f(a) = b \).

**Question:** What does it mean to say that sets \( A \) and \( B \) are the *same size*?

**Answer:** \( A \) and \( B \) are the *same size* if there is a correspondence from \( A \) to \( B \).
Correspondence (2)

**Question:** In a crowded room, how can we tell if there are more people than chairs, or more chairs than people?

**Answer:** Establish a correspondence: ask everyone to sit down.

(c.f., Mathematician’s trick for counting a herd of cows ...)

Slides modified by Benny Chor, based on original slides by Maurice Herlihy, Brown Univ. – p.41
Correspondence

**Claim:** The set $\mathcal{N}$ of natural numbers has the same size as the set $\mathcal{E}$ of even numbers

**Proof:** Let $f(i) = 2i$.

**Remark:** The set $\mathcal{E}$ is a proper subset of the set $\mathcal{N}$, yet they are the same size!
Countable Sets

**Definition:** A set $A$ is *countable* if

- either $A$ is finite, or
- $A$ has the same size as $\mathcal{N}$, the natural numbers.

We have just seen that $\mathcal{E}$ is countable.

A countable set is sometimes said to have size $\aleph_0$.

**Claim:** The set $\mathcal{Z}$ of integers is countable.

**Proof:** Define $f : \mathcal{N} \rightarrow \mathcal{Z}$ by

$$f(i) = \begin{cases} 
  i/2 & \text{if } i \text{ is even} \\
  -(\lfloor i/2 \rfloor + 1) & \text{if } i \text{ is odd}
\end{cases}$$
Pop Quiz

In Heaven, there is a hotel with a countable number of rooms.

One day, the society of Prophets, Oracles, and AI Researchers holds a 3-day convention that books every room in the hotel.

Then one more guest arrives, claiming he invented Lisp, and angrily demanding a room.

You are the manager. What do you do?

Answer: Ask the guest in room $i$ to move to room $i + 1$, and put the newcomer in room 1.
Pop Quiz #2

Then a countable number of guests arrive, all angrily demanding rooms. *(What a noise!)*

Now what do you do?

**Answer:** Ask the guest in room $i$ to move to room $2i$, and put the newcomers in the *odd-numbered rooms*. 
Rational Numbers

Let

\[ \mathbb{Q} = \left\{ \frac{m}{n} \mid m, n \in \mathbb{N} \right\} \]

**Theorem:** \( \mathbb{Q} \) is countable.

This claim may seem counterintuitive.

Idea

- list \( \mathbb{Q} \) as 2-dim array
- begin counting with the first row . . .

Why doesn’t this work?
Enumerate numbers along northeast and diagonals, skipping duplicates.

Does this mean that every infinite set is countable?
The Real Numbers

Every *real number* has a decimal representation. For example, \( \pi = 3.1415926 \ldots \), \( \sqrt{2} = 1.4142136 \ldots \), and \( 0 = 0.0000000 \ldots \).

Let \( \mathcal{R} \) be the set of real numbers.

**Theorem:** \( \mathcal{R} \) is uncountable.

\( \mathcal{R} \) is sometimes said to have size \( \aleph_1 \).

- This is Cantor’s historic proof, which
- introduced the *diagonalization* method.
The Real Numbers

Assume there is a correspondence between $\mathbb{N}$ and $\mathbb{R}$. Write it down:

<table>
<thead>
<tr>
<th>$n$</th>
<th>$f(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.14159...</td>
</tr>
<tr>
<td>2</td>
<td>55.55555...</td>
</tr>
<tr>
<td>3</td>
<td>40.18642...</td>
</tr>
<tr>
<td>4</td>
<td>15.20601...</td>
</tr>
</tbody>
</table>

We now show that there is a number $x$ not in this list.
Diagonalization

Pick $0 \leq x \leq 1$, so its significant digits follow decimal point. Will ensure $x \neq f(n)$ for all $n$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$f(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$3.14159...$</td>
</tr>
<tr>
<td>2</td>
<td>$55.55555...$</td>
</tr>
<tr>
<td>3</td>
<td>$40.18643...$</td>
</tr>
<tr>
<td>4</td>
<td>$15.20607...$</td>
</tr>
</tbody>
</table>
First fractional digit of $f(1)$ is 1, so pick first fractional digit of $x$ to be something else (say, 2).

Second fractional digit of $f(2)$ is 5, so pick second fractional digit of $x$ to be something else (say, 6).

and so on . . .

$x = 0.2691 . . .
Diagonalization

A similar proof shows there are languages that are not enumerable.

- the set of Turing machines is countable, but
- the set of languages is uncountable!

Ergo,

- there exist languages that are not enumerable *(why?)*
- indeed, “most” languages are not enumerable *(explain)*
∃ Countably Many Turing Machines

Claim: The set of strings, $\Sigma^*$, is countable.

Proof: List strings of length 0, then length 1, then 2, and so on. This exhausts all of $\Sigma^*$. The union of countably many finite sets is countable.
∃ Countably Many Turing Machines (2)

Claim: The set of all Turing machines is countable.

Proof: Each TM $M$ has an encoding as a string $\langle M \rangle$. Therefore there is a one-to-one mapping from the set of all TMs into (but not onto) $\Sigma^*$. Since $\Sigma^*$ is countable, so is the set of all TMs.
The Set of All Languages is Uncountable

Let $\mathcal{B}$ be the set of of infinite binary sequences.

**Claim:** $\mathcal{B}$ is uncountable.

**Proof:** Diagonalization argument, essentially identical to the proof that $\mathcal{R}$ is uncountable.

(Additional helpful clue: think of binary sequence as binary expansion!)
The Set of Languages is Uncountable (2)

Let $\mathcal{L}$ be the set of all languages over alphabet $\Sigma$. Recall $\mathcal{B}$ is the set of infinite binary sequences. We give a correspondence

$$\chi : \mathcal{L} \rightarrow \mathcal{B}$$

called the language’s *characteristic sequence*.

- Let $\Sigma^* = \{s_1, s_2, s_3, \ldots\}$ (in lexicographic order).
- Each language $L \in \mathcal{L}$ is associated with a unique sequence $\chi(L) \in \mathcal{B}$:
- the $i$-th bit of $\chi(L)$ is 1 if and only if $s_i \in L$. 
The Set of Languages is Uncountable (3)

Each language $L \in \mathcal{L}$ has a unique sequence $\chi(L) \in \mathcal{B}$:
the $i$-th bit of $\chi(L)$ is 1 if and only if $s_i \in L$.

Example:

$\Sigma^* \{\varepsilon, 0, 1, 00, 01, 10, 11, 000 \ldots\}$
$A \{ 0, 00, 01, \ldots\}$
$\chi(A) \{0, 1, 0 1, 1, 0 0 1 \ldots\}$

The map $\chi : \mathcal{L} \rightarrow \mathcal{B}$

- is one-to-one and onto (why?),
- and is hence a correspondence.
- It follows that $\mathcal{L}$ is uncountable.
TMVs vs. Languages

We saw that the set of all Turing machines is countable.

We saw that the set $\mathcal{L}$ of all languages over alphabet $\Sigma$ is uncountable.

Therefore there are languages that are not accepted by any TM.

This is an existential proof – it does not explicitly show any such language.
Halting, Again

At long last, we are able to prove the undecidability of

$$A_{TM} = \{\langle M, w \rangle | M \text{ is a TM that accepts } w \}.$$ 

Proof: By contradiction. Suppose a TM, $H$, is a decider for $A_{TM}$.

On input $\langle M, w \rangle$, where $M$ is a TM and $w$ is a string, $H$ halts and accepts if and only if $M$ accepts $w$. Furthermore, $H$ halts and rejects if $M$ fails to accept $w$. 
Halting (2)

On input \(\langle M, w \rangle\), where \(M\) is a TM and \(w\) is a string, \(H\) halts and accepts if and only if \(M\) accepts \(w\). Furthermore, \(H\) halts and rejects if \(M\) fails to accept \(w\).

\[
H(\langle M, w \rangle) = \begin{cases} 
\text{accept} & \text{if } M \text{ accepts } w \\
\text{reject} & \text{if } M \text{ does not accept } w
\end{cases}
\]
Halting (3)

Now we construct a new TM, $D$, with $H$ as a subroutine.

$D$ does the following

- Calls $H$ to determine what TM, $M$, does when the input to $M$ is its own description, $\langle M \rangle$.
- When $D$ determines this, it does the opposite.
- So $D$ rejects if $M$ accepts $\langle M \rangle$, and accepts if $M$ does not accept $\langle M \rangle$. 
Halting (4)

More precisely, $D$ does the following:

- Run $H$ on input $\langle M, \langle M \rangle \rangle$.
- Output the opposite of what $H$ outputs:
  - If $H$ accepts, reject, and
  - If $H$ rejects, accept.
Self Reference (4)

Don’t be confused by the notion of running a machine on its own description!

Actually, you should get used to it.

- Notion of self-reference comes up again and again in diverse areas.
- Read “Gödel, Escher, Bach, an Eternal Golden Braid”, by Douglas Hofstadter.
- This notion of self-reference is the basic idea behind Gödel’s revolutionary result.

Compilers do this all the time . . . .
The Punch Line

So far we have,

$$D(\langle M \rangle) = \begin{cases} reject & \text{if } M \text{ accepts } \langle M \rangle \\ accept & \text{if } M \text{ does not accept } \langle M \rangle \end{cases}$$

What happens if we run $D$ on its own description?

$$D(\langle D \rangle) = \begin{cases} reject & \text{if } D \text{ accepts } \langle D \rangle \\ accept & \text{if } D \text{ does not accept } \langle D \rangle \end{cases}$$

Oh, oh...

Or, more accurately, a contradiction (to what?)
Once Again

- Assume that TM $H$ decides $A_{TM}$.
- Then use $H$ to build a TM, $D$, that when given $\langle M \rangle$, accepts exactly when $M$ does not accept.
- Run $D$ on its own description.
- $D$ does:
  - $H$ accepts $\langle M, w \rangle$ when $M$ accepts $w$.
  - $D$ rejects $\langle M \rangle$ exactly when $M$ accepts $\langle M \rangle$.
  - $D$ rejects $\langle D \rangle$ exactly when $D$ accepts $\langle D \rangle$.
- Last step leads to contradiction.
- Therefore neither TM $D$ nor $H$ can exist.
- So $A_{TM}$ is undecidable!
Diagonalization

This proof is diagonalization in transparent disguise. To unveil this, let’s start by making a table.

<table>
<thead>
<tr>
<th></th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>accept</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>accept</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Entry $(i, j)$ is accept if $M_i$ accepts $\langle M_j \rangle$, and blank if $M_i$ rejects or loops on $\langle M_j \rangle$. 
Diagonalization (2)

<table>
<thead>
<tr>
<th></th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>accept</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td></td>
<td></td>
<td>reject</td>
<td>reject</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
</tr>
</tbody>
</table>

Run $H$ on the corresponding inputs. In the new table, each entry $(i, j)$ states whether $H$ accepts $\langle M_i, \langle M_j \rangle \rangle$.

<table>
<thead>
<tr>
<th></th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
</tr>
</tbody>
</table>

Slides modified by Benny Chor, based on original slides by Maurice Herlihy, Brown Univ.
Diagonalization (3)

Now we add $D$ to the table.

- By assumption, $H$ is a TM, and therefore so is $D$.
- It occurs on the list $M_1, M_2, \ldots$ of all TMs.
- $D$ computes the opposite of the diagonal entries.
- At diagonal entry, $D$ computes its own opposite!

<table>
<thead>
<tr>
<th></th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\ldots$</th>
<th>$\langle D \rangle$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$D$</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td></td>
<td>$\text{??}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
</tbody>
</table>
A Non-enumerable Language

- We already saw a non-decidable language: $A_{TM}$.
- Can we do better (i.e., worse)?
- Mais, oui!
- We now display a language that isn’t even enumerable . . .
A Non-enumerable Language

Earlier we saw

**Theorem:** If $L$ and $\overline{L}$ are both enumerable, then $L$ is decidable.

**Corollary:** If $L$ is not decidable, then either $L$ or $\overline{L}$ is not enumerable.

**Definition:** A language is co-enumerable if it is the complement of an enumerable language.

Reformulating theorem

**Theorem:** A language is decidable if and only if it is both enumerable and co-enumerable.
$\overline{A_{TM}}$ is not Enumerable

**Theorem:** If $L$ and $\overline{L}$ are both enumerable, then $L$ is decidable.

- We proved that $A_{TM}$ is undecidable.
- On the other hand, we saw that the universal TM, $U$, accepts $A_{TM}$.
- Therefore $A_{TM}$ is enumerable.
- If $A_{TM}$ were also enumerable, then by theorem $A_{TM}$ was decidable.
- Therefore $A_{TM}$ is not enumerable.
Question: Are there any languages in the area marked ??? ?

Answer: Yes, heaps (why?)