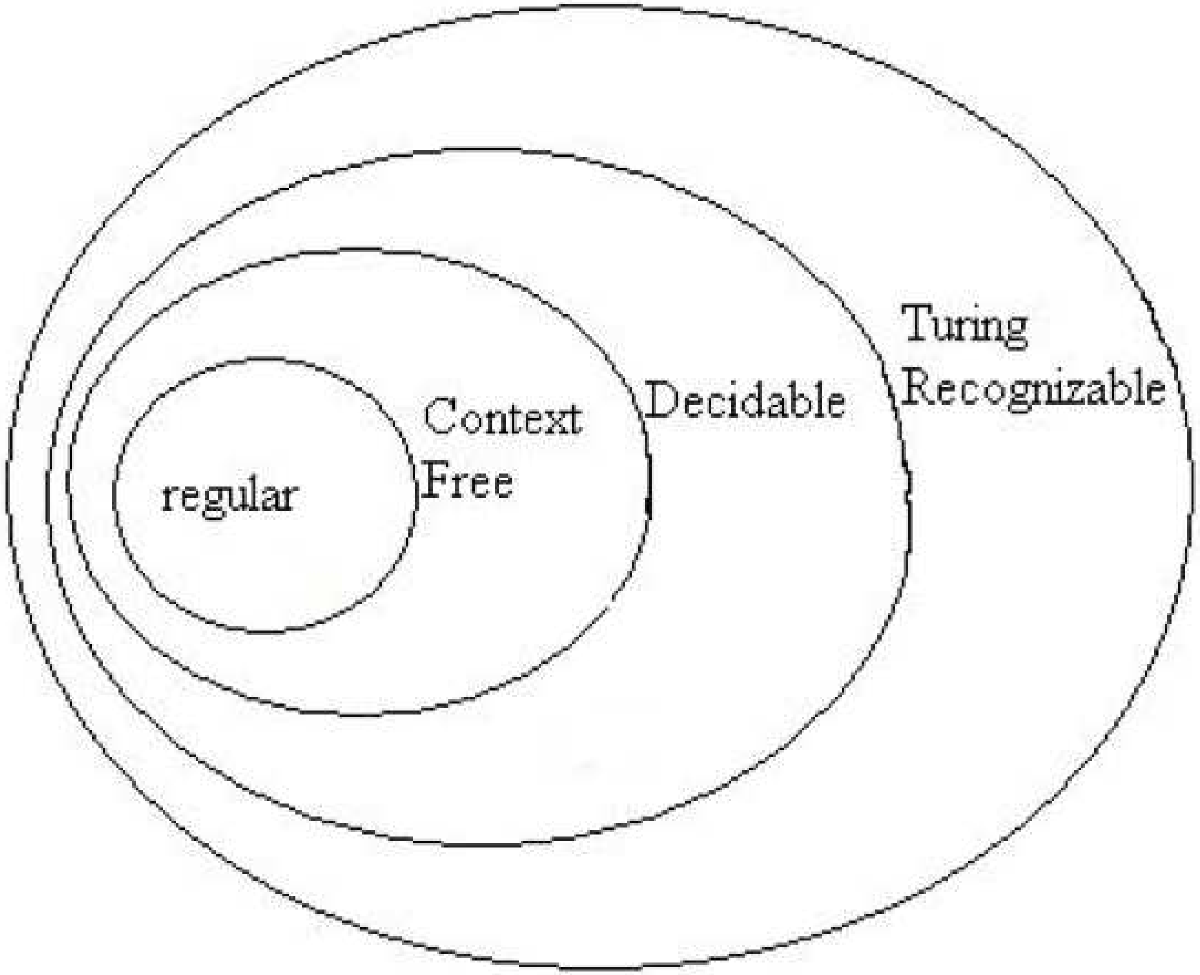
The hierarchy of languages



The Halting Problem (HP)

Does a given TM M accept a given input string w?

$$A_{TM} = \{\langle M, w\rangle : M \text{ is a Turing machine that accepts } w\}$$

- Theorem: A_{TM} is Turing-recognizable

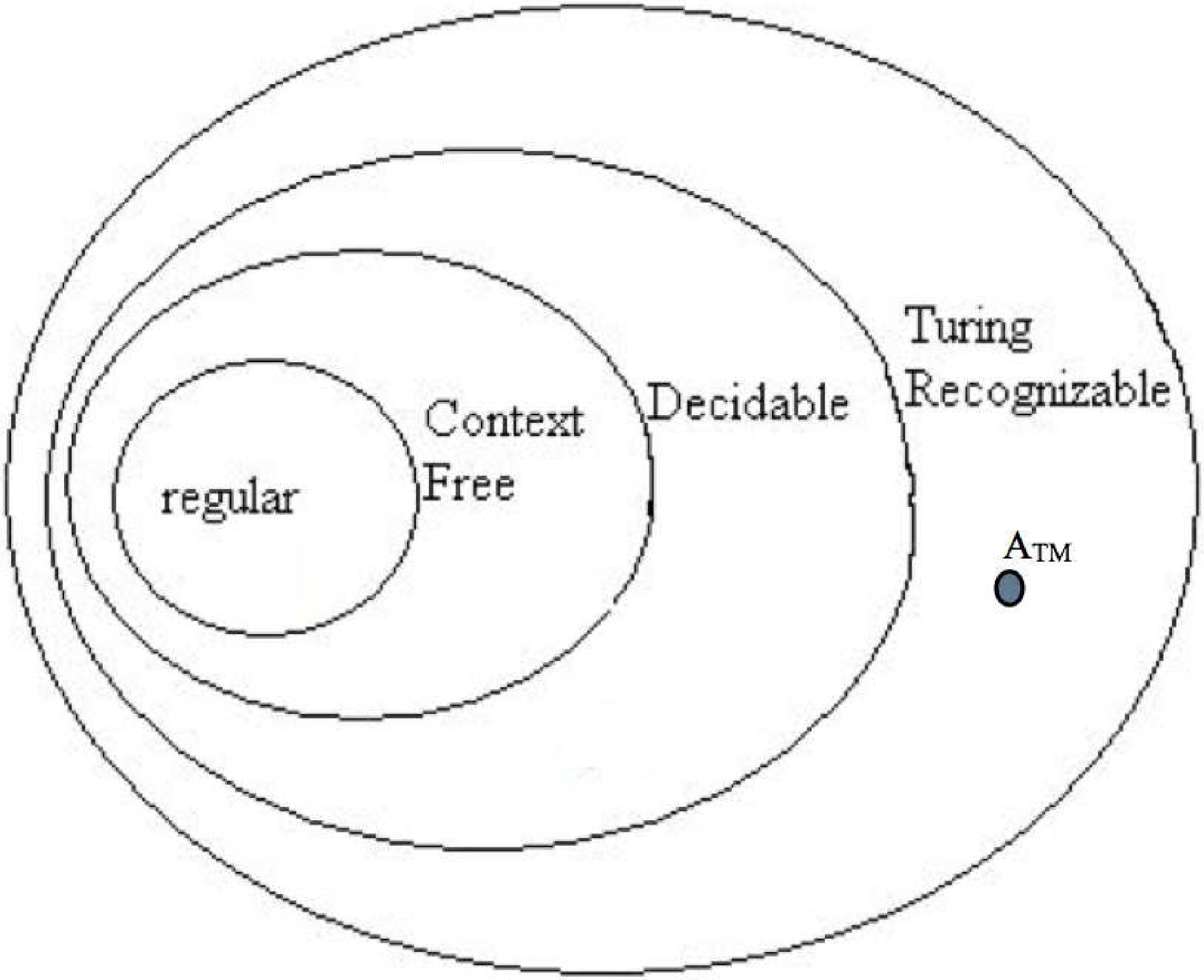
Proof: We present a TM U that recognizes A_{TM}.

$U =$ "On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on input w.
2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *reject*."

U will loop on $\langle M, w\rangle$ if M loops on w. If the algorithm had some way to determine that M was not halting on w, it would *reject*. We'll see that this can't be done.

The HP is Undecidable



A short mathematical detour

The diagonalization method.

- Proposed in 1873 by Georg Cantor as an ingenious method to compare the sizes of infinite sets.
- Cantor's definition of set size: Let A and B be two sets.
  - A and B have the same size if there is a one-to-one, onto function $f : A \to B$. Such a function is called a correspondence.
    - A function f is called one-to-one if $f(a) \neq f(b)$ whenever $a \neq b$.
    - A function $f : A \to B$ is called onto if $\forall b \in B, \exists a \in A : f(a) = b$.
  - A set is countable if it is finite or it has the same size as $\mathcal{N}$.

Example

Proving (by diagonization) that the set $\mathcal{L}$ of all infinite binary languages in **uncountable.**

- Assume it is countable, i.e., it has the same size as $\mathcal{N}$ and there is a correspondence $f : \mathcal{N} \to \mathcal{L}$.

x	$f(x)$
0	$\underline{1}10010\cdots$
1	$1\underline{1}0010\cdots$
2	$11\underline{0}000\cdots$
$\vdots$	$\vdots$

- We can build an infinite binary string c that makes f impossible to exist.
- Each i-th bit of c is different from the i-th bit of $f(i)$, e.g., $c = 001\cdots$.
- $c \neq f(n)$ for any n.
- Therefore, $\mathcal{L}$ is uncountable.

Proving the HP is undecidable

A proof by diagonalization.

$A_{TM} = \{\langle M, w\rangle : M \text{ is a Turing machine that accepts } w\}$

- We assume A_{TM} is decidable and there is a TM H that decides A_{TM}:

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

- Next we construct a rather interesting TM D that makes the existence of H impossble.
  - it uses H as a subroutine.
  - it calls H to determine what M does when the input to M is its own description $\langle M\rangle$.

Proof (cont.)

$D =$ "On input $\langle M \rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M \rangle \rangle$
2. Output the opposite of what H outputs, i.e., if H accepts, reject; if H rejects, accept."

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

What happens then when we run D on $\langle D \rangle$?

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

Therefore, neither TM D nor TM H can exist. $\square$

The diagonalization in the proof of HP

The diagonalization becomes apparent when we analyze the behaviour of H and D.

- Let's consider the table below representing TMs and their encodings.
- The entries tell whether a machine in a given row accepts the input in a given column.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\cdots$
M_1	$accept$		$accept$	
M_2	$accept$	$accept$	$accept$	
M_3		$accept$		$\cdots$
$\vdots$			$\vdots$	

- Let's run H on these inputs and see what we get.

Result of running $H(M_i, \langle M_j \rangle)$

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\cdots$
M_1	$accept$	$reject$	$accept$	
M_2	$accept$	$accept$	$accept$	
M_3	$reject$	$accept$	$reject$	$\cdots$
$\vdots$		$\vdots$		

- Since D is also a TM, it must occur in the list $M_1, M_2, \cdots$ of all TMs.
- Lets see D in the table.

Resulting of running $D(\langle M_i \rangle)$

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	$\underline{accept}$	$reject$	$accept$		$accept$	
M_2	$accept$	$\underline{accept}$	$accept$		$reject$	
M_3	$reject$	$accept$	$\underline{reject}$	$\cdots$	$accept$	$\cdots$
$\vdots$			$\vdots$	$\ddots$		
D	$reject$	$reject$	$accept$		$\underline{?}$	

- D computes the opposite of the diagonal entries.
- The contradiction occurs at the '?', where the entry must be the opposite of itself.

Turing-unrecognizable languages

- We have just demonstrated a language, A_{TM}, that is undecidable.
- Before we go on a hunt for a Turing-unrecognizable language, we should first find out whether such language actually exists.
- Are there more problems than computer programs?
  - If there are more languages than Turing machines, and given that a TM recognizes a single language, then certainly there are some languages that are Turing-unrecognizable.
- Once we have a satisfactory answer for this question, we shall search for such Turing-unrecognizable language.

Are there more problems than programs?

- Let's first show that the set of TMs is countable:
  - The set of strings Σ^* is countable (there are finitely many strings of each length).
  - Therefore, the set of TMs is also countable, because each TM M has an encoding $\langle M \rangle$ selected from Σ^*.
- Now lets show that the set of all languages is uncountable.

Are there more problems than programs?

- The set of all infinite binary strings $\mathcal{B}$ is uncountable
- Let $\mathcal{L}$ be the set of all languages over $\Sigma = \{0, 1\}$. We'll prove that $\mathcal{L}$ is uncountable by showing a correspondence $f : \mathcal{L} \to \mathcal{B}$.
- Let $A \in \mathcal{L}$. Sort Σ^* and A. We define a *"Characteristic sequence"* for A, as follows: the ith bit of an infinite binary string $\mathcal{X}_A$ is a 1 if $s_i \in A$ and a 0 if $s_i \notin A$

$$
\begin{array}{lllllllllll}
\Sigma^* & = & \{ & \epsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & \cdots\} \\
A & = & \{ & & 0, & & 00, & 01, & & & 000, & 001, & \cdots\} \\
\mathcal{X}_A & = & & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \cdots
\end{array}
$$

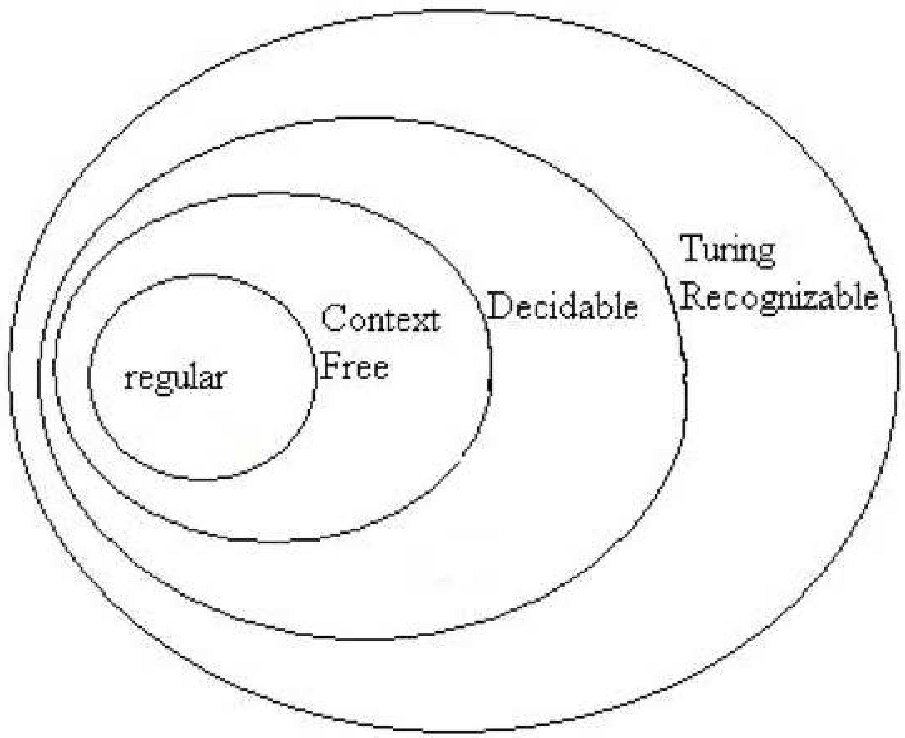
$f : \mathcal{L} \to \mathcal{B}$, where $f(A)$ equals a *characteristic sequence* of A, is one-to-one, onto. Therefore $\mathcal{L}$ is also uncountable.

Quod erat demonstrandum □

Therefore, the theorem

> *"Some languages are not Turing-recognizable."*

is finally proven, as we have shown that there are more languages than TMs.



A Turing-unrecognizable language

- Now that we know Turing-unrecognizable languages do exist, it is time to look at a specimen:
  - The *diagonalization language*, L_d: let w and M denote ordered sets of all possible input strings and TMs, respectively. Then L_d is the set of strings w_i such that $w_i \notin L(M_i)$.
- before proving that L_d is Turing-unrecognizable, let's define a binary version of this language.

The Diagonalization Language

- Suppose there is a binary code for all turing machines M
- Suppose we sort and enumerate all binary strings w and all Turing machines M
- Now consider the table below representing the acceptance of strings by Turing machines
- A row in the table represents the *characteristic vector* of $L(Mi)$, i.e., 1s represent a input string wi that Mi accepts.

	w1	w2	w3	w4	...
M1	0	1	0	0	...
M2	1	1	0	1	...
M3	1	0	1	0	...
M4	0	0	0	1	...
.	.	.	.	.	...

The Diagonalization Language

- The language L_d, the *diagonalization language*, is the set of strings w_i such that w_i is not in $L(M_i)$
- To construct characteristic vector for L_d we complement the diagonal of the table, thus obtaining 1000...
- Notice that this vector disagrees in some column with every row of the table.
- Therefore, the diagonal cannot be the characteristic vector for any TM.

	w1	w2	w3	w4	...
M1	0	1	0	0	...
M2	1	1	0	1	...
M3	1	0	1	0	...
M4	0	0	0	1	...
.	.	.	.	.	...
Md	1	0	0	0	..?.

Exercises

- Show that the language below is decidable.

$$\{\langle A\rangle : \ A \text{ is a DFA that recognizes } \Sigma^*\}$$

- Show that the problem of testing whether a CFG generates some string in 1^* is decidable. I.e., show that the language below is decidable

$$\{\langle G\rangle : \ G \text{ is a CFG over } \{0,1\}^* \text{ and } 1^* \cap L(G) \neq \{\}\}$$