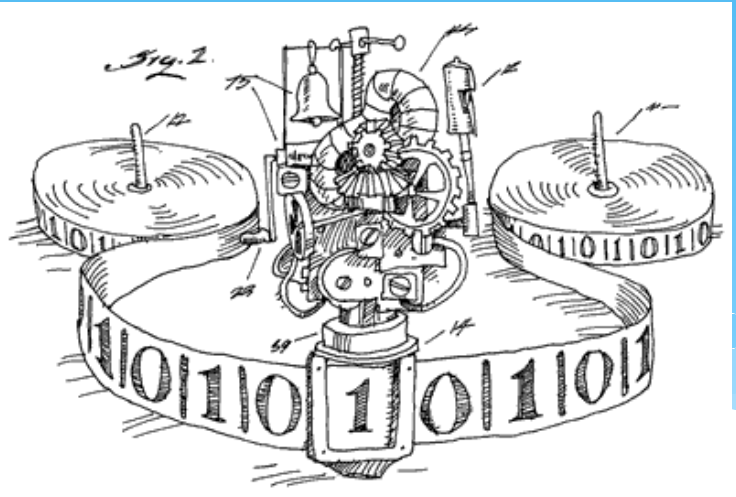
EECS 376: Foundations of Computer Science

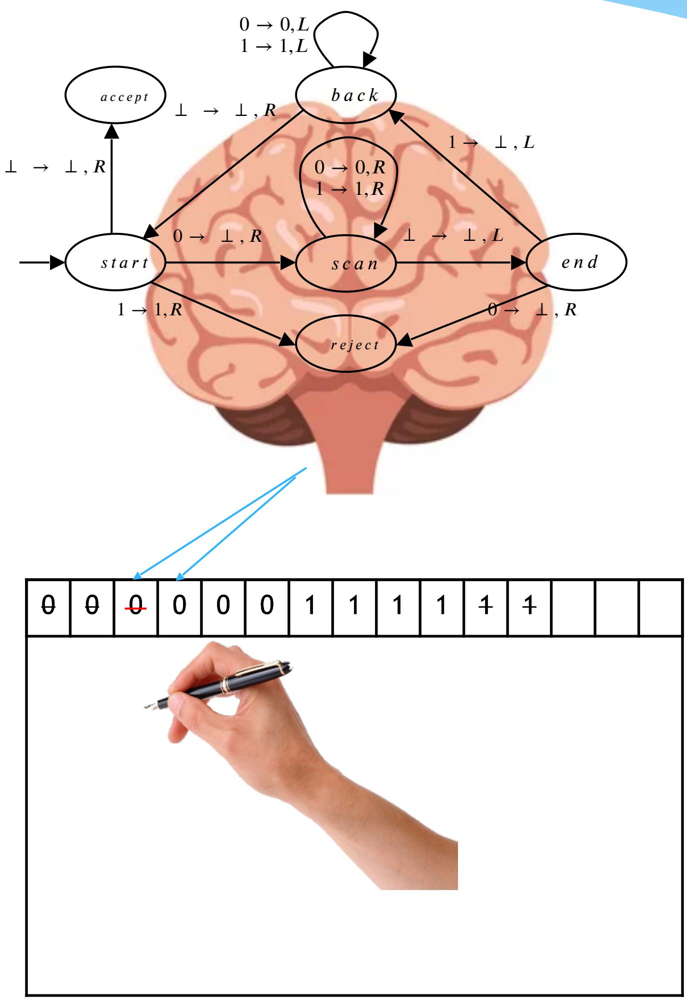
Chris Peikert
1 February 2023



Today’s Agenda

* Recap: Turing Machines, Church-Turing thesis
* **Deciders** (vs “loopers”) and **decidability**
* **Diagonalization** and an **undecidable language**

Turing Machines: Essential Features



1. *Finite* **alphabets**: input & tape

2. *Finite* "brain"/logic/"**code**": state machine

3. *Infinite* **storage:** "tape"

4. Sequence of *local* computations, specified by the code:
   - Read a single symbol
   - Write a single symbol
   - Move a single cell
   - Update active state

5. Runs until it enters a *terminal* state—if ever!

Decision Programs

* **Q:** Suppose we run a TM M on string x. What are the possible outcomes?
  * $M(x)$ either: (i) accepts, (ii) rejects, *or* (iii) ***"loops" (forever)***

* **Definition:** A TM M ***decides*** language L if M:
  1. *accepts* every string $x \in L$ ("$M(x)$ accepts"), and
  2. *rejects* every string $x \notin L$ ("$M(x)$ rejects").

We say that M is a ***decider*** (for L), and L is ***decidable***.
* **Note:** By definition, M *does not loop* on any input!

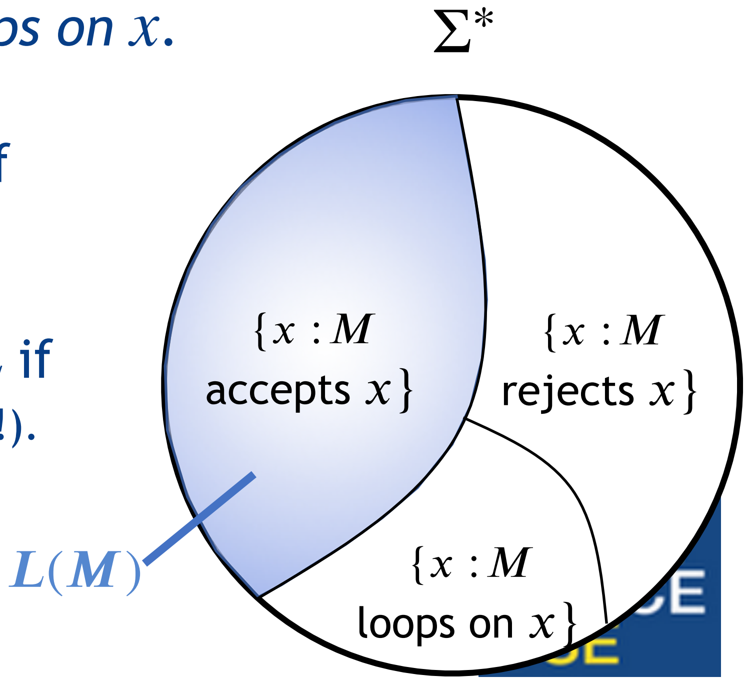
EECS ECE CSE

More Generally: Language of a TM

* **Definition:** The **language** of a TM M is $L(M) := \{x : M \text{ accepts } x\}$.
* **Question:** What if $x \notin L(M)$? ($M(x)$ does not accept.)
* **Answer:** Then M either *rejects* x, or *loops on* x.

* **Conclusion:** TM M decides language L iff $L(M) = L$ and M halts on *every* input.

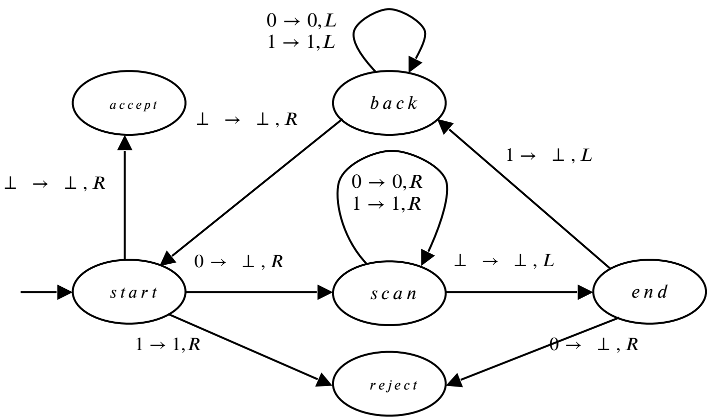
* **Definition**: TM M **recognizes** language L if $L(M) = L$ (regardless of whether M ever loops!).
* More on this later…



Code vs TMs

* **Claim:** *Any* TM can be simulated by a "Boolean" function on strings, written in FPL code (e.g., C++).

* **Q:** Can any "Boolean" function on strings written in FPL code be simulated by a TM?
* **A:** Yes: implement an FPL 'interpreter' with a TM.



Key Idea: TM $\equiv$ "bool M(string x)"

simulateM(string x):
// *simulates* TM **M** on string x
// - hard-coded transition function
// - maintain state & tape cells
return accept/reject according to M

Summary So Far

- **General:** Any *finite* object (integer, graph, PDF, **C++ code**) can be encoded as a finite string. A TM takes a finite string as input.
- **Church-Turing thesis:**

  "Anything that is computable by some mechanical device (a 'computer') is computable by some **Turing machine**."

- **In short:** Turing Machines ≡ computer programs.

- **Implication:** If a problem *is not* decidable by a TM, it *cannot* be solved by *any* computer! (including future/alien technology)

Can *every* decision problem be decided by some TM?

Proving Decidability

* **Q:** To prove that a language L is decidable, must we design a TM?
* **A:** No! *Simulation* lets us write an algorithm in our FPL.
* **Example:** $L = \{(a, b) : a, b \in \mathbb{N}, \ \gcd(a, b) = 1\}$
* $M(a, b)$:
  * Compute $Euclid(a, b)$
  * If $Euclid(a, b) = 1$, accept, else reject
* **Analysis (Correctness):**
  * $(a, b) \in L \implies \gcd(a, b) = 1 \implies \text{Euclid}(a, b) = 1 \implies M(a, b)$ accepts
  * $(a, b) \notin L \implies \gcd(a, b) \neq 1 \implies \text{Euclid}(a, b) \neq 1 \implies M(a, b)$ rejects

Another Example

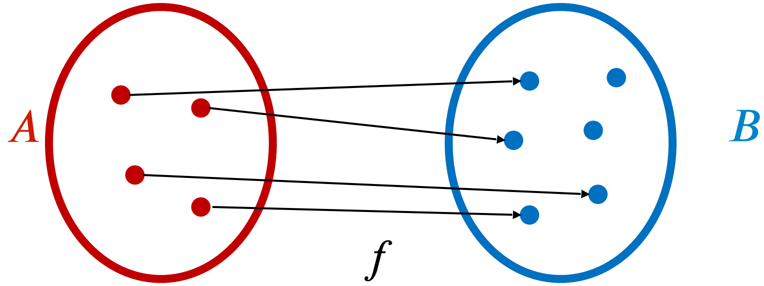
* **Claim:** If L is decidable, then $L' = L \cup \{\varepsilon\}$ is decidable.
* **Proof:** By hypothesis, there is some TM D that decides L. We use it to define another TM D' that decides L':
* $D'(x)$:
  1. If $x = \varepsilon$, accept.
  2. Run $D(x)$ and output the same answer.
* **Analysis (correctness, incl. halting):**
  * $D(x)$ halts (because D is a decider) $\Longrightarrow D'(x)$ halts
  * $x \in L' \iff x = \varepsilon$ or $x \in L \iff x = \varepsilon$ or $D(x)$ accepts $\iff D'(x)$ accepts
* **Conclusion:** D' is a decider for L'.

Undecidable Languages?

* **Definition:** A language L is ***decidable*** if there exists a TM (program) M that decides L.
* **Question:** Do there exist *undecidable* languages?
  I.e., are there problems that no computer can solve?
* **Idea:** Let $\mathscr{L}$ be the set of all languages, $\mathscr{M}$ be the set of all TMs.
* If we could show that $|\mathscr{M}| < |\mathscr{L}|$, we would be done!
* **Why:** Each TM M decides at most 1 language.
* **Problem:** Both $\mathscr{M}$ and $\mathscr{L}$ are infinite! Can we do anything about it?

Functions and Set Cardinality

* A function $f: A \rightarrow B$ maps each element $a \in A$ to an element $f(a) \in B$.
* A function is ***injective (1-to-1)*** if each element $a \in A$ is mapped to a <u>*different*</u> element $f(a) \in B$.
  * Formally: $\forall a_1, a_2 \in A\,.\ \ a_1 \neq a_2 \Longrightarrow f(a_1) \neq f(a_2)$.



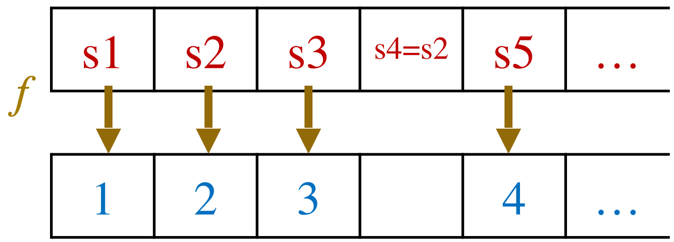
* If an injective $f: A \rightarrow B$ exists, "$|A| \leq |B|$"

Countable Sets

- **Informal Def:** A set S is ***countable*** if it is "no larger than" the naturals $\mathbb{N} = \{0, 1, 2, \ldots\}$, i.e., "$|S| \leq |\mathbb{N}|$".
- **Formal Def:** S is countable if there exists an injective function $f : S \to \mathbb{N}$. [f says: "s is the ith element of S"]
- **Claim:** Any finite set is countable.
- **Proof:** Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set with n elements. Then $f(s_i) = i$ is an injection from S to $\mathbb{N}$.
- **Q:** Which *infinite* sets are countable ("countably infinite")?

Proving Countability

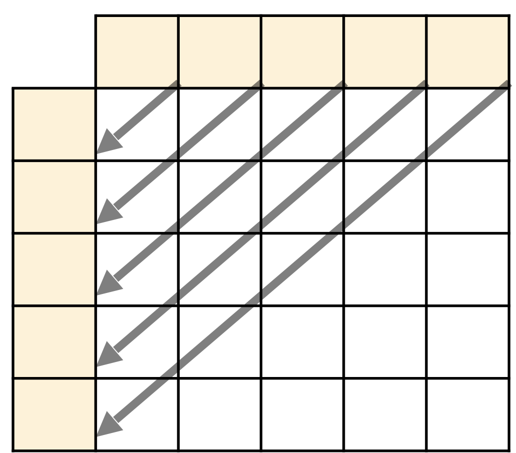
- S is countable if there is an injective $f : S \to \mathbb{N}$.
- We can prove that a set S is countable by explicitly defining such a function.
- Or, we can show how to *list* elements of S so that each element $s \in S$ appears *somewhere* in the list.
  - This implicitly defines an injective $f : S \to \mathbb{N}$:



Countably Infinite Sets

* $\mathbb{N}$ **By Definition:** $0 \quad 1 \quad 2 \quad \ldots$
* $\mathbb{Z}$ **Try:** $0 \quad 1 \quad 2 \quad 3 \quad \ldots \quad -1 \quad -2$
* **Does not work!** we will never get beyond "…"
* $\mathbb{Z}$ **Try:** $0 \quad 1 \quad -1 \quad 2 \quad -2 \quad \ldots$
* $\mathbb{Q}^{+}$:? List x/y where $x + y = 1$, then $x + y = 2$, etc..

	1	2	3	4	…
1	1/1	1/2	1/3	1/4	…
2	2/1	~~2/2~~	2/3	~~2/4~~	…
3	3/1	3/2	~~3/3~~	3/4	…
4	4/1	~~4/2~~	4/3	~~4/4~~	…
…	…	…	…	…	~~…~~



Finite Binary Strings

* Let $S = \{0,1\}^*$ be the set of all (finite) binary strings.
* **Claim:** S is countable.
* **Proof:** List the elements of S in ***lexicographic order***: by length, and then character by character
  * $\varepsilon,0,1,00,01,10,11,000,001,\ldots$
* Every element $s \in S$ appears on the list: s appears in the block of all strings of length $|s| \in \mathbb{N}$.

Uncountable Sets

* **Q:** How do we show that a set S is *uncountable*?
* **A:** Prove that no injective $f: S \to \mathbb{N}$ exists.
* **How?** Proof by contradiction! Template:
    1. Assume there exists a list of elements of S such that every $s \in S$ appears *somewhere* in the list.
    2. Use it to ‘construct’ some $s^* \in S$ that is not in the list.
    3. Contradiction! So, no such list can exist.
* ***Diagonalization*** is the usual technique for this.

Diagonalization

- Let S be the set of *infinite-length* binary sequences.
- Suppose there is some list X in which every element of S appears. Represent it in a table.
- Take the 'diagonal' bits and flip them: $s^* = \overline{s_1[1]}\ \overline{s_2[2]}\ \overline{s_3[3]} \cdots$

X

s1	0	1	1	0	1	0	…
s2	1	0	0	0	0	0	…
s3	0	1	1	1	0	1	…
s4	0	0	0	0	0	0	…
s5	1	1	1	1	1	1	…
…	…	…	…	…	…	…	…

s*	1	1	0	1	0	…

EECS ECE CSE

Diagonalization

- **Claim:** $s^* = \overline{s_1[1]}\,\overline{s_2[2]}\,\overline{s_3[3]}\cdots$ is not in the list X. Formally: $s^* \neq s_i$ for *every* i.
- This is because $s^*[i] \neq s_i[i]$, i.e., s^* and s_i differ in their ith positions, by construction.
- This contradicts the assumption that list X exists!

X

s1	0	1	1	0	1	0	…
s2	1	0	0	0	0	0	…
s3	0	1	1	1	0	1	…
s4	0	0	0	0	0	0	…
s5	1	1	1	1	1	1	…
…	…	…	…	…	…	…	…

s*	1	1	0	1	0	…

Back to Our Question

* **Conclusion:** The set of all *infinite* binary sequences is uncountable.
* **Diagonalization Summary:** For any *candidate* list of such sequences, there is a sequence *not* in that list.
* **Recall:** Let $\mathscr{L}$ be the set of all languages, $\mathscr{M}$ be the set of all TMs. Can we show that $|\mathscr{M}| < |\mathscr{L}|$?
* **Q1:** Is $\mathscr{M}$ countable?
* **A:** Yes. (We'll see.)
* **Q2:** Is $\mathscr{L}$ countable?
* **A:** No! (We'll see.)

An Undecidable Language (1/3)

- **Claim:** The set of *deciders* is *countable*.
- **Idea:** Use lex. ordering on *source code* $\in \{0,1\}^*$. (Every TM has an encoding as a string!)

$$\langle M_1 \rangle = \text{“bool } \mathbf{A}\text{(string x): } \mathbf{return} \text{ F”}$$
$$\langle M_2 \rangle = \text{“bool } \mathbf{A}\text{(string x): } \mathbf{return} \text{ T”}$$
$$\langle M_3 \rangle = \text{“bool } \mathbf{A}\text{(string x): for i=1…x: …”}$$
$$\langle M_4 \rangle = \text{“bool } \mathbf{A}\text{(string x): let x=x-1 …”}$$
$$\vdots$$

An Undecidable Language (2/3)

- **Claim:** Any language L is *equivalent* to an *infinite* binary sequence.
- **Idea:** List $\Sigma^* = \{s_1, s_2, s_3, s_4, \ldots\}$.
  - Then $L \equiv x_1x_2x_3x_4\cdots$, where $x_i \in \{0,1\}$ indicates if $s_i \in L$.
- **Example:** suppose $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$.

Language decided by "bool **M**(string x): **return** ($|x|$ is even)" $\equiv 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0 \cdots$

An Undecidable Language (3/3)

- Take list of languages (sequences) $L(M_i)$ for deciders M_i.
- There is a language L^* that is not in the list: 'flip the diagonal'.
- **Claim:** No TM decides L^*. Every decider M_i appears in the list, but 'behaves incorrectly' on string s_i !

X

	s1	s2	s3	s4	s5	s6	…
M1	1	0	0	1	1	0	…
M2	0	1	1	0	0	0	…
M3	1	1	1	1	1	1	…
M4	0	0	0	0	0	0	…
M5	1	0	1	0	0	0	…
…	…	…	…	…	…	…	…

L*	0	0	0	1	1	…

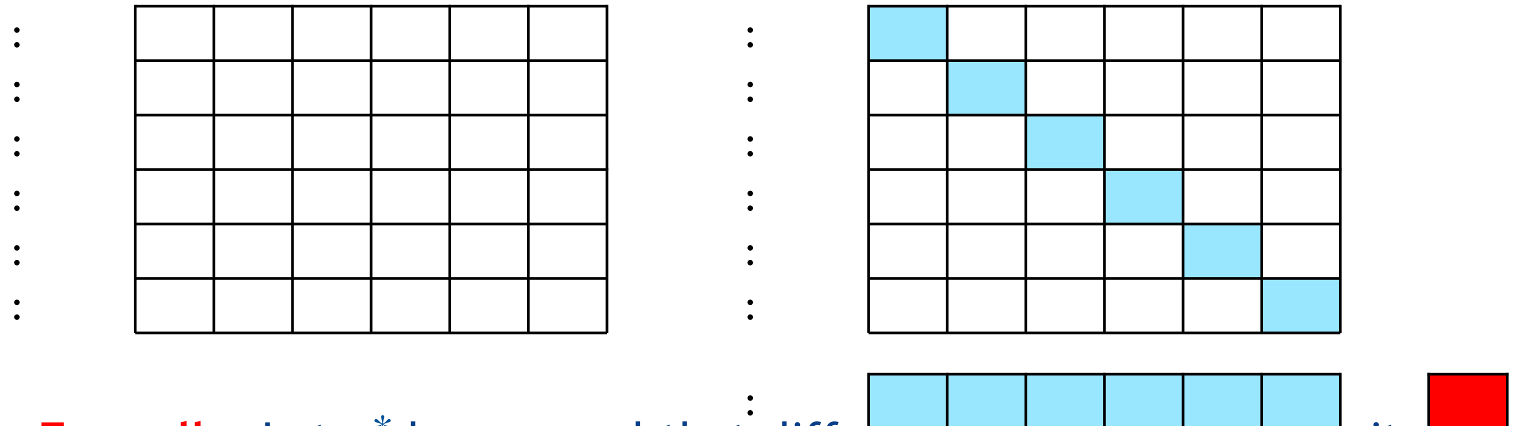
Conclusion

* **Theorem:** There exists an undecidable language L^*.
* **Interpretation:** There is a problem that no computer program can solve correctly (on all inputs).
* **Question:** What problem does L^* represent? Do we care about it? Would it be useful to solve?
* **Answer:** We do not know, since the proof is 'non-constructive': only shows *existence* of L^*!
* **Next time:** Some "useful" undecidable languages.

Extra

The Reals are Uncountable

* **Proof:** A real in $[0,1)$ is represented by an infinite sequence of digits. Assume there were a complete list of reals $[0,1)$.
* **Task:** Find a real in $[0,1)$ that is not on the list and we are done!



* **Formally:** Let r^* be any real that differs from r_k in the k^{th} digit.
* **Question:** Is r^* on the list?
* **Answer:** No. r^* is on the list $\Longrightarrow \exists k$ s.t. $r^* = r_k$ but $r^* \neq r_k$!