Universality & Undecidability

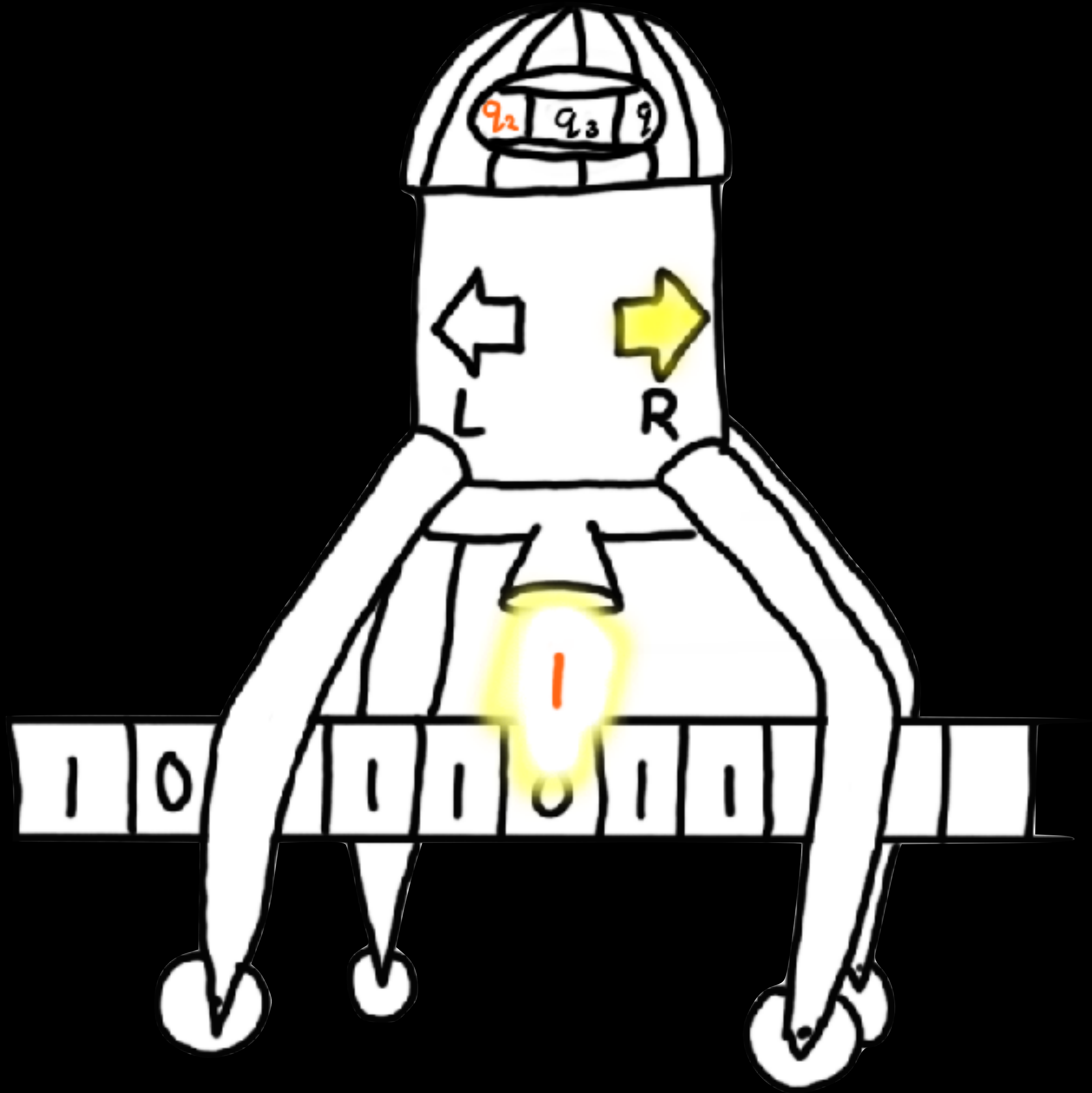
in computation *Now*
elsewhere *Later*

Gemma De las Cuevas
Institute for Theoretical Physics, University of Innsbruck
June 21, 2022
Solstice of Foundations, Zürich
@Gemma_DLC

Today, at approximately 11:15 Zürich time, is the summer solstice

Universality & Undecidability

in computation



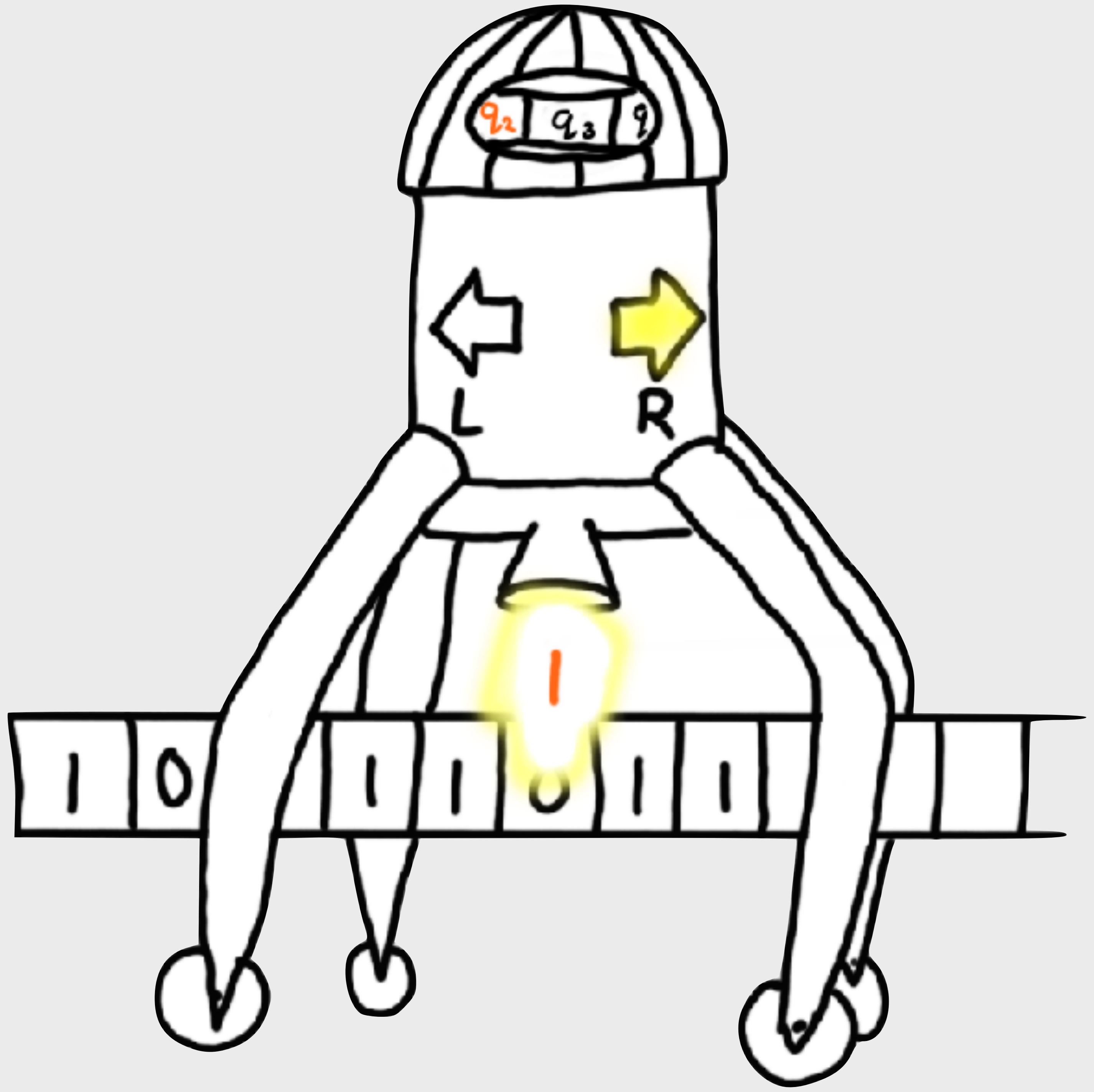
Turing machines

- A Turing machine is a model of computation.

Head

Action of the head on the tape

Tape



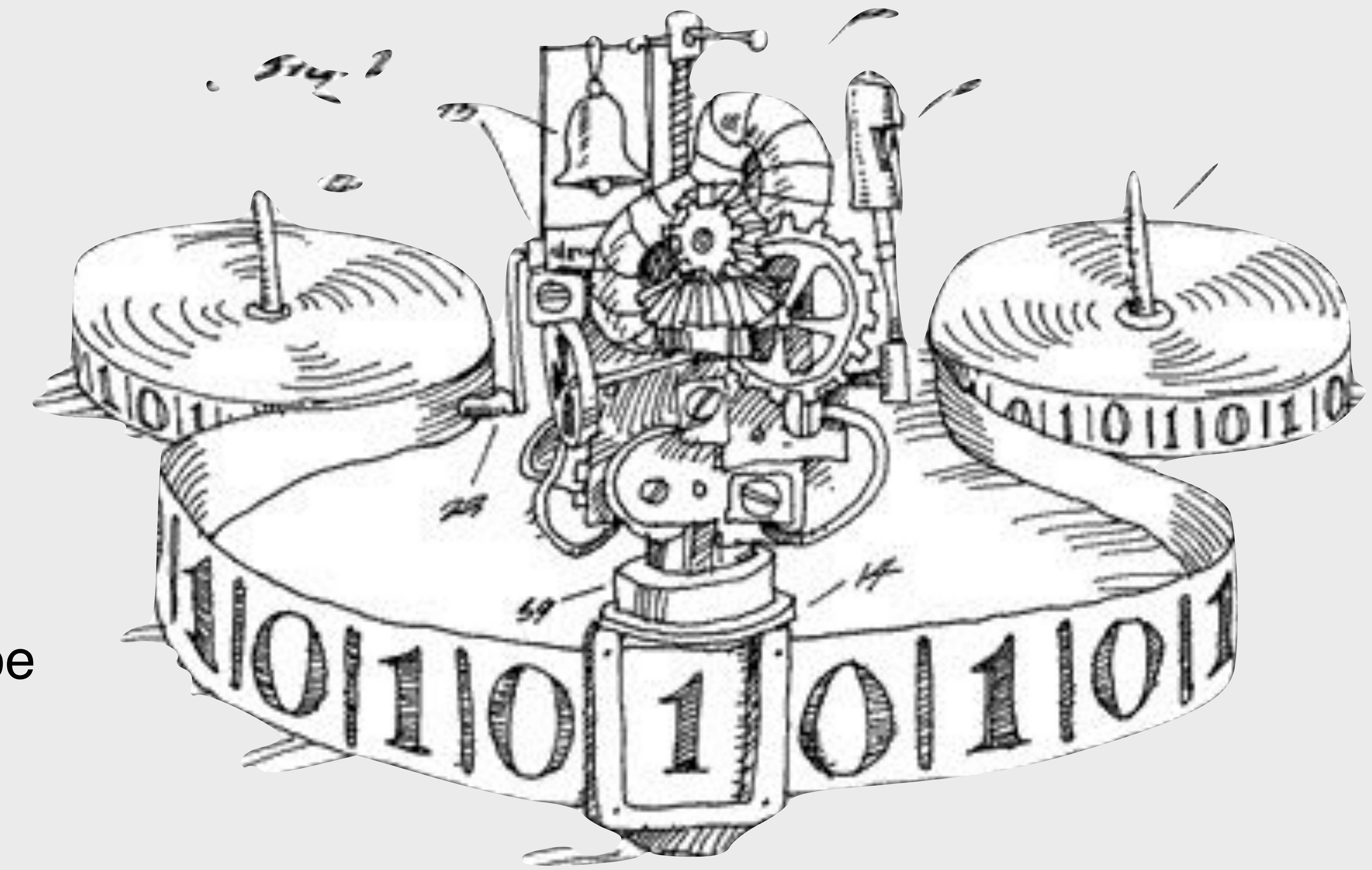
Ultimately, the head accepts or rejects (or gets stuck in a loop, i.e. does not halt) the input written on the tape.

Turing machines

- A Turing machine is a model of computation.

Head

Tape



Action of the head on the tape

Ultimately, the head accepts or rejects (or gets stuck in a loop, i.e. does not halt) the input written on the tape.

Turing machines

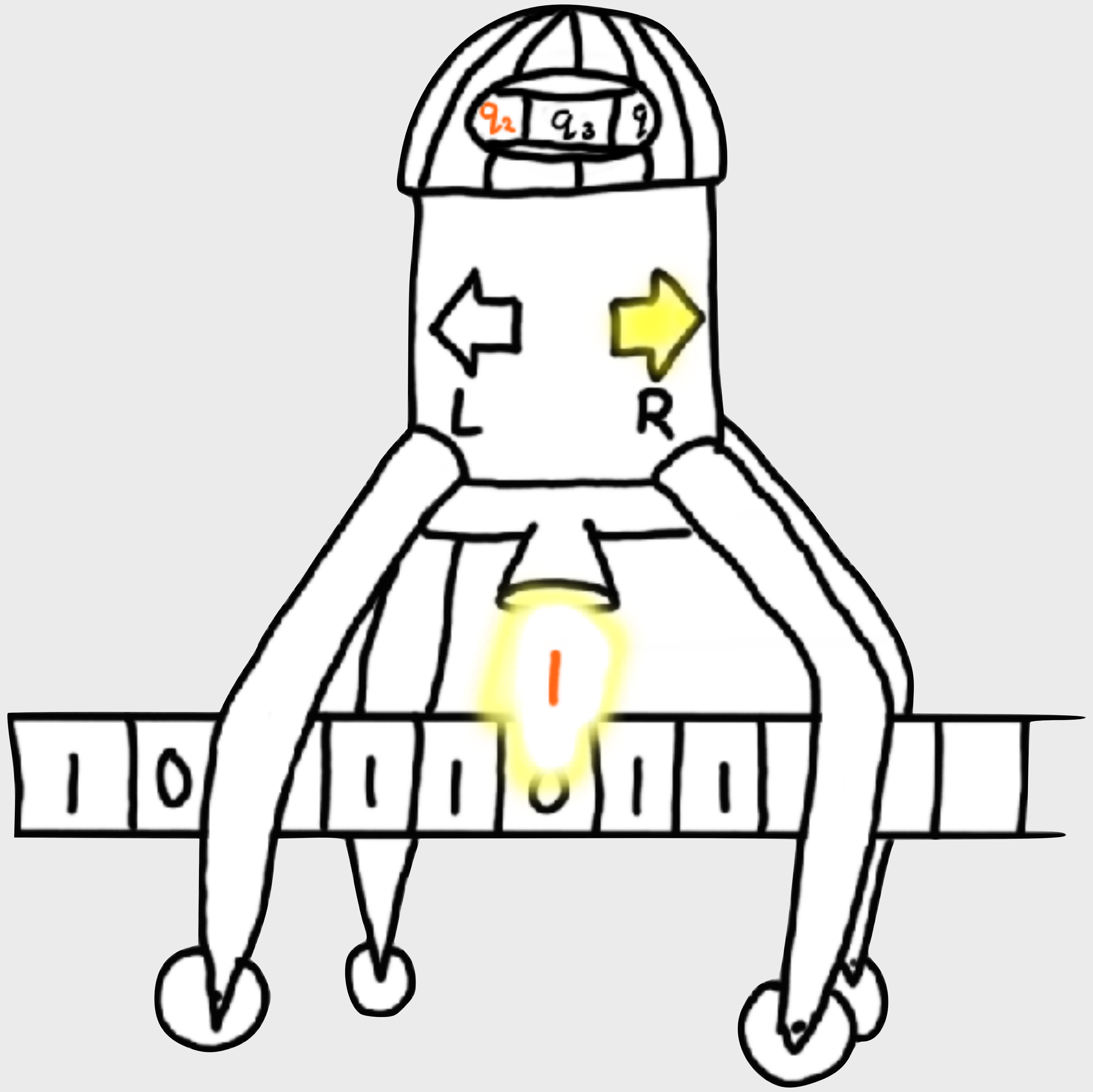
- Head
  - In a state from a finite set of states, $q \in \Gamma$
  - A start state, an accept state, and a rejecting state
- Tape
  - With symbols from a finite alphabet, $r \in \Sigma$, and a blank symbol $\sqcup$
  - Of unbounded length
- At each step, the head reads a cell of the tape, overwrites it, changes its state, and moves left or right:

  $(q, r) \to (p, s, M)$ where $M \in \{\text{Left}, \text{Right}\}$

  $(q_i, r_i) \to (p_i, s_i, M_i)$ for i in a finite set.

  - The program is this finite set of transition rules.

A finite alphabet is a finite number of symbols, such as $\{0,1\}$ or $\{a, b, c\}$ or $\{ \diamond \}$

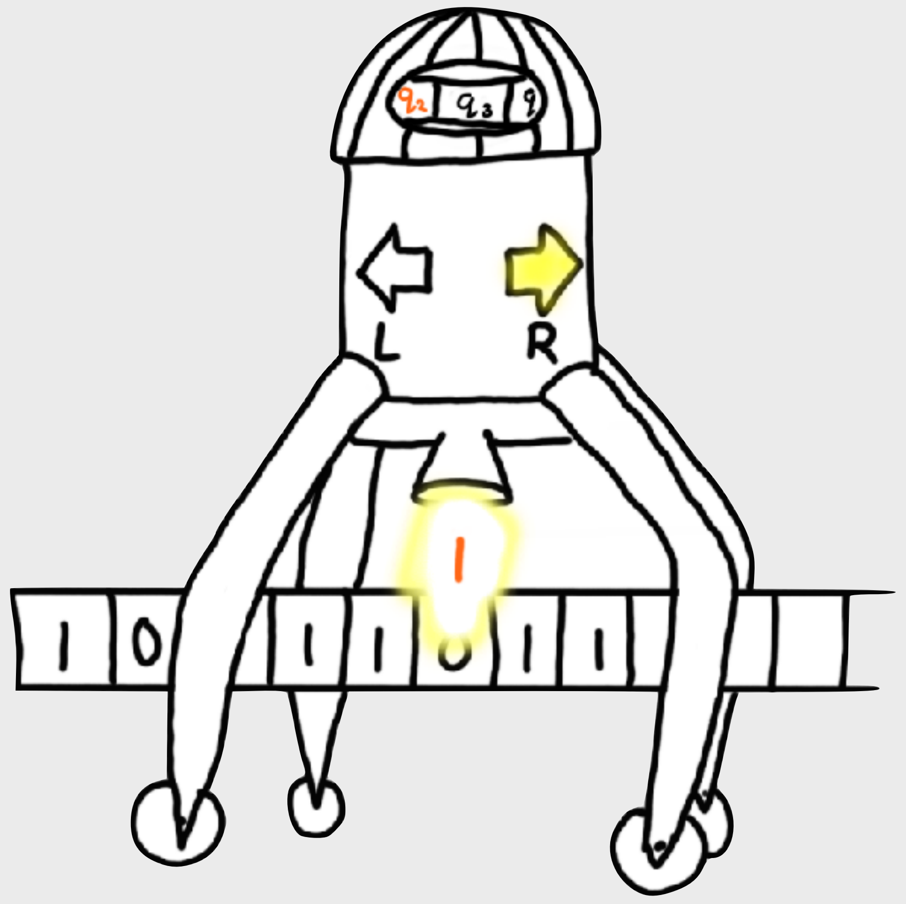


Turing machines

- Intuitively, a Turing machine is what a human computer with finite memory and a notebook with a symbol per page can do.
- Very robust definition — the following don't change their power
  - Adding more tapes, making the tape infinite instead of semiinfinite, a unary tape alphabet, having access to another Turing machine, having nondeterministic rules…
- In practice very cumbersome to write algorithms for Turing machines
- Turing machines are one of several models of computation
  - λ calculus, partial recursive functions, tag systems…

Church-Turing thesis

A function on the natural numbers can be calculated by an effective method if and only if it is computable by a Turing machine.



Universal Turing machines

- It seems that for every function one wants to compute, one needs to build a new machine. This is not the case!

- There is a single program that can run any computation - the **universal Turing machine**.

- The universal Turing machine has fixed transition rules, and it is fed

  [description of the Turing machine to be simulated] # [input]

$$x \longrightarrow \boxed{T} \longrightarrow T(x)$$

$$d(T,x) \longrightarrow \boxed{U} \longrightarrow T(x)$$

- This is a reprogrammable machine:
  - Hardware: the program of the universal Turing machine
  - Software: the description of the algorithm which is part of the tape.

Universal Turing machines

- Let's encode a program as data
  - Consider Turing machine T, and fix the head alphabet to be unary
  - Identify Left with 1 and Right with 2.
  - Encode transition rule $(q, r) \to (p, s, M)$ as the string $10^p 10^r 10^q 10^s 10^M$
  - Encode multiple transition rules as as $\underbrace{10^{p_1}10^{r_1}10^{q_1}10^{s_1}10^{M_1}\,1\,10^{p_2}10^{r_2}10^{q_2}10^{s_2}10^{M_2}\,1\;\ldots\;1\,10^{p_n}10^{r_n}10^{q_n}10^{s_n}10^{M_n}}_{\text{this string encodes a program}} =: d(T)$

Unary encoding of the naturals:
$n \mapsto \underbrace{\diamond \diamond \ldots \diamond}_{n} = \diamond^n$

Universal Turing machines

- On input $d(T)\#x$, the universal Turing machine U acts as follows:
  - It has three tapes
    - The top tape contains $d(T)$
    - The middle tape initially contains x and later holds the simulated contents of T's tape
    - The bottom tape contains the current state of T and the current position of T's head
  - U simulates T on input x one step at a time, shuttling back and forth between $d(T)$ and the simulated contents of T's tape.
  - In each step, U updates T's state and simulated tape contents as dictated by T's transition function.
  - If ever T halts and accepts or halts and rejects, U does the same.

Universal Turing machines

- If you build a universal Turing machine, you can run any algorithm.

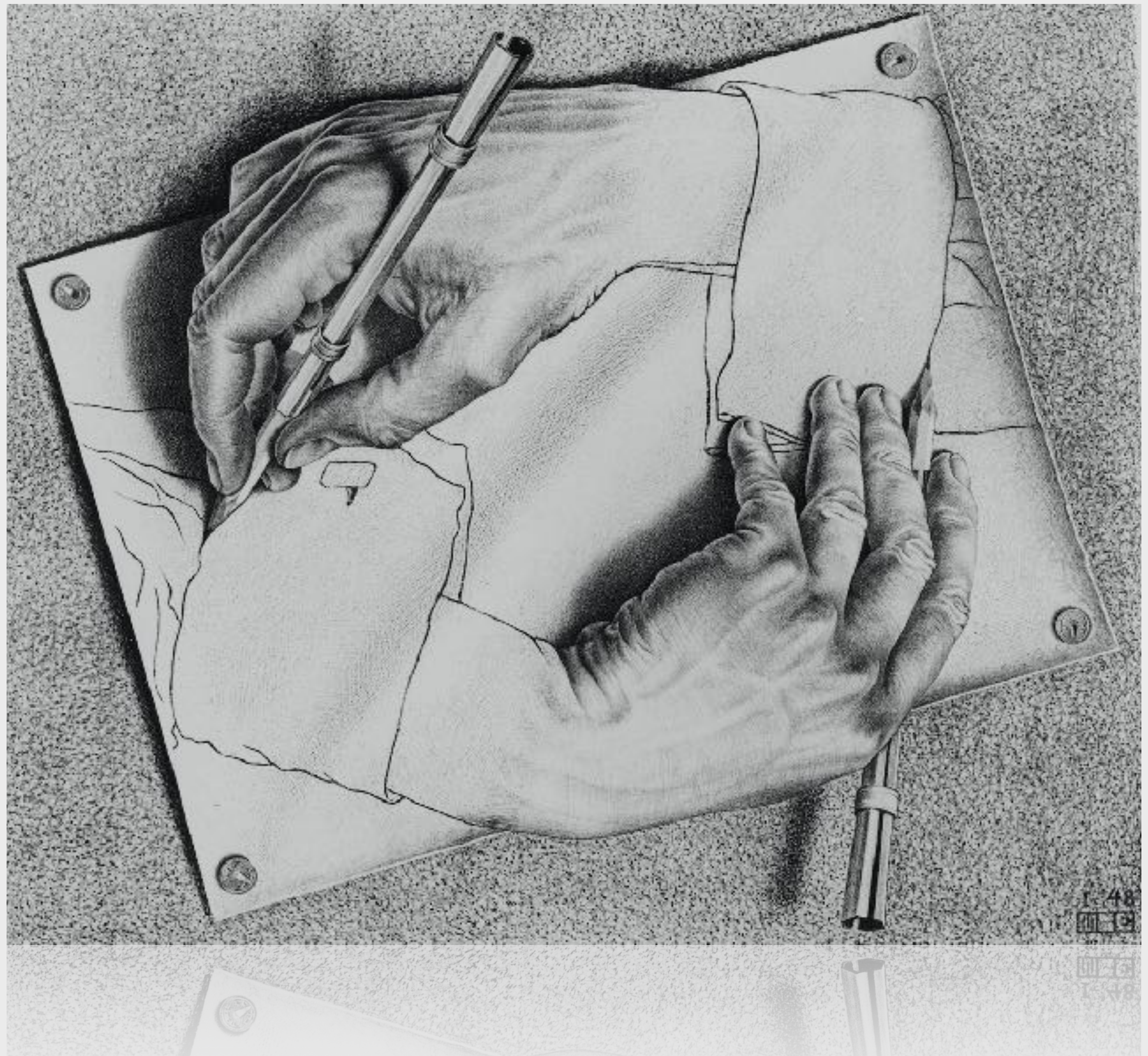
Universal Turing machines

- Conceptually, programs seemed to be at a higher hierarchical level than data
- This distinction is only superficial: programs can be encoded as data, which tell the machine which program to run.
- Strange loop, or tangled hierarchy

The program

$\downarrow$

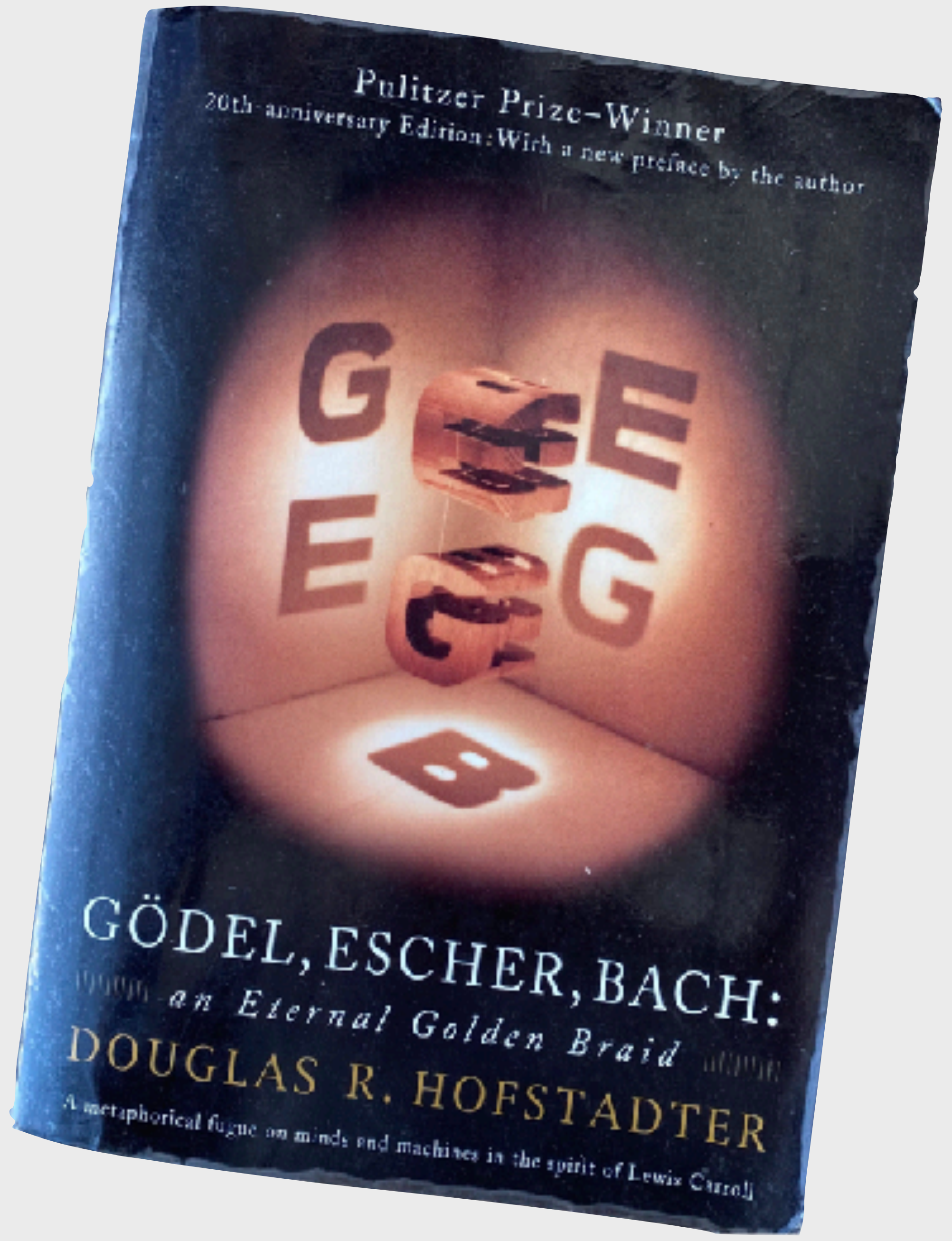
The data



The program

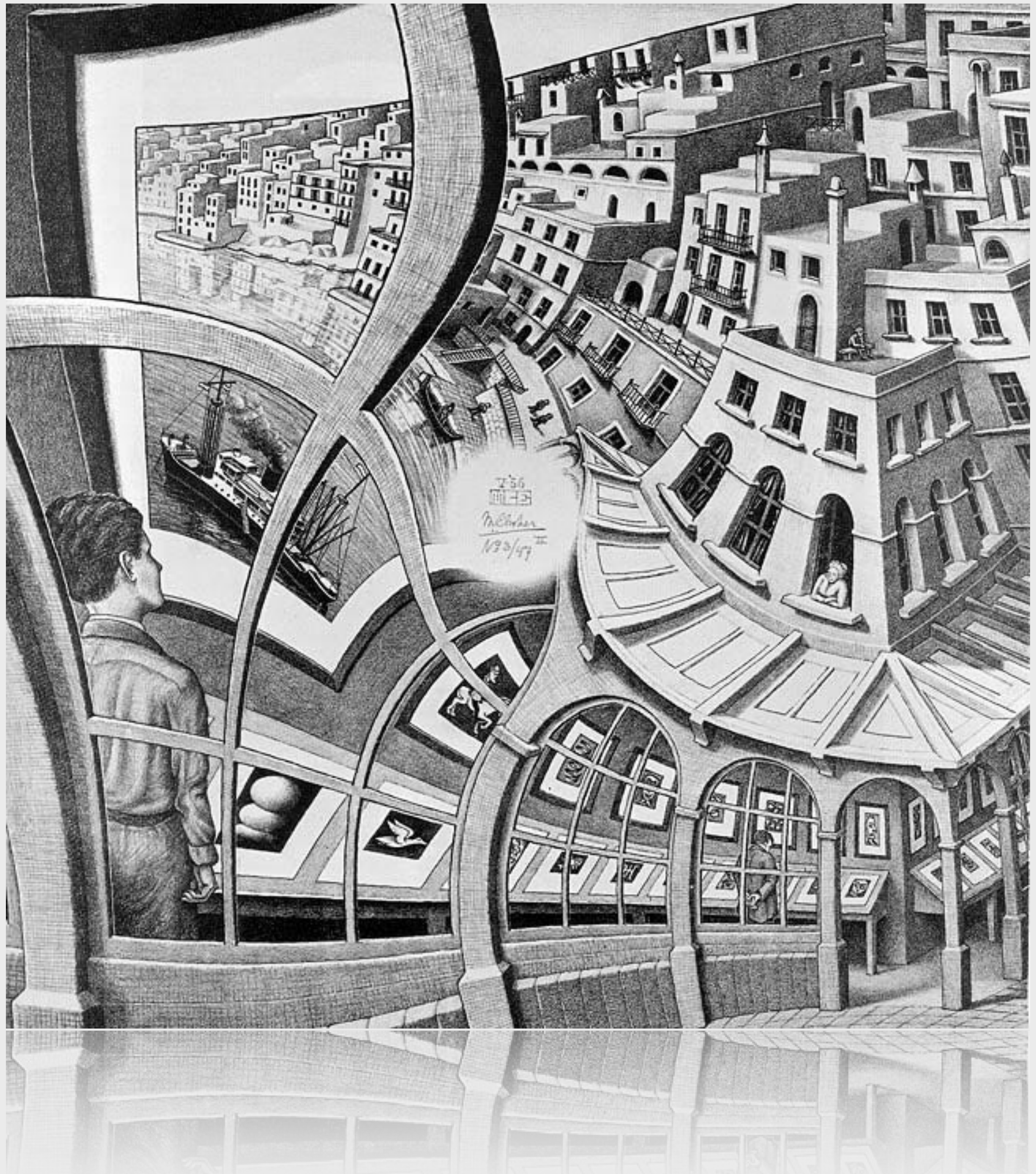
$\uparrow$

The data



Universal Turing machines

- Strange loop, or tangled hierarchy



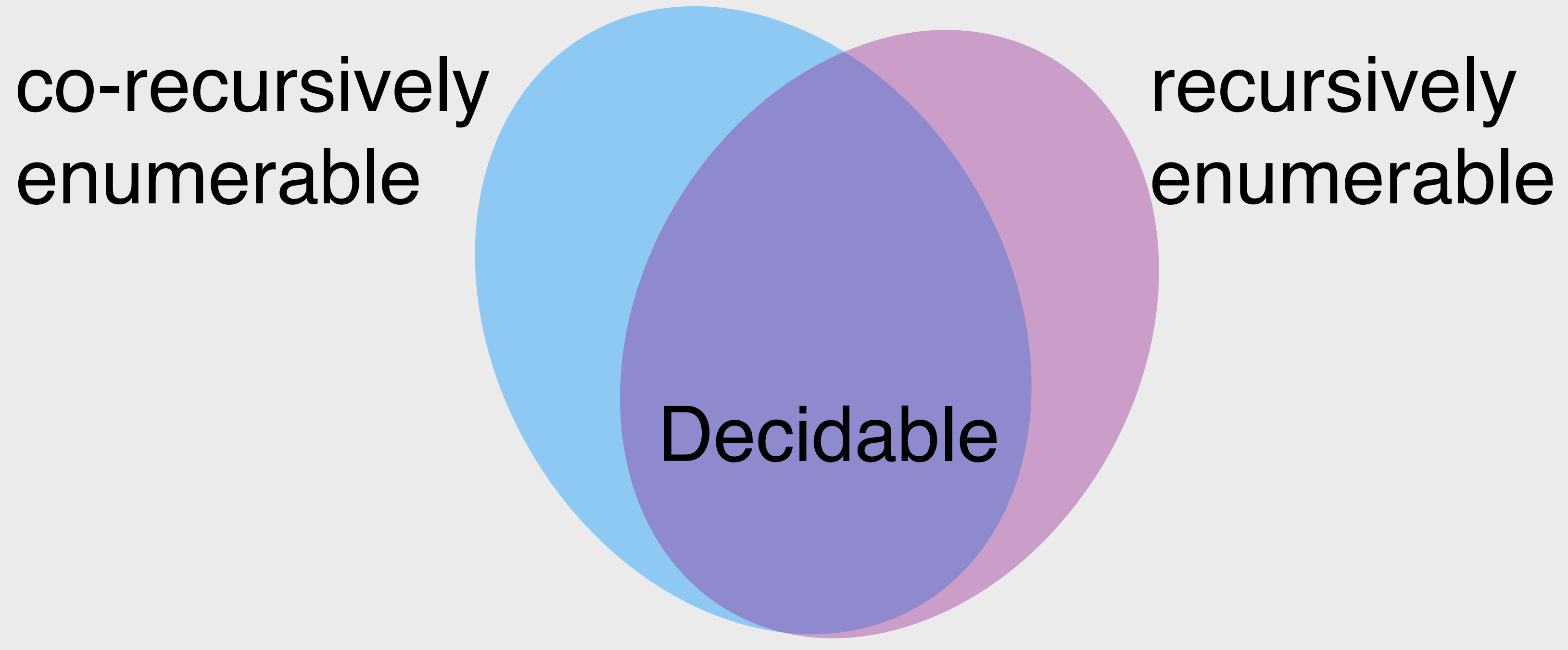
Machines and their languages

- A **formal language** is $L \subseteq \Sigma^*$ where Σ is a finite alphabet.

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n \text{ where } \Sigma^n = \Sigma \times \ldots \times \Sigma \quad n \text{ times}$$

- The set of strings accepted by Turing machine M is $L(M)$.
  - A Turing machine M **accepts** L if, for every $x \in L$, M accepts x.
  - If, additionally, for every $x \notin L$, M rejects x, then M **decides** L.

- A language L is **recursively enumerable** if it is accepted by a Turing machine.
- A language L is **decidable** if it is decided by a Turing machine.



- That decidable $\neq$ recursively enumerable was shown by Turing.
- The polynomially bounded version of this problem is the famous $P \neq NP$ conjecture.

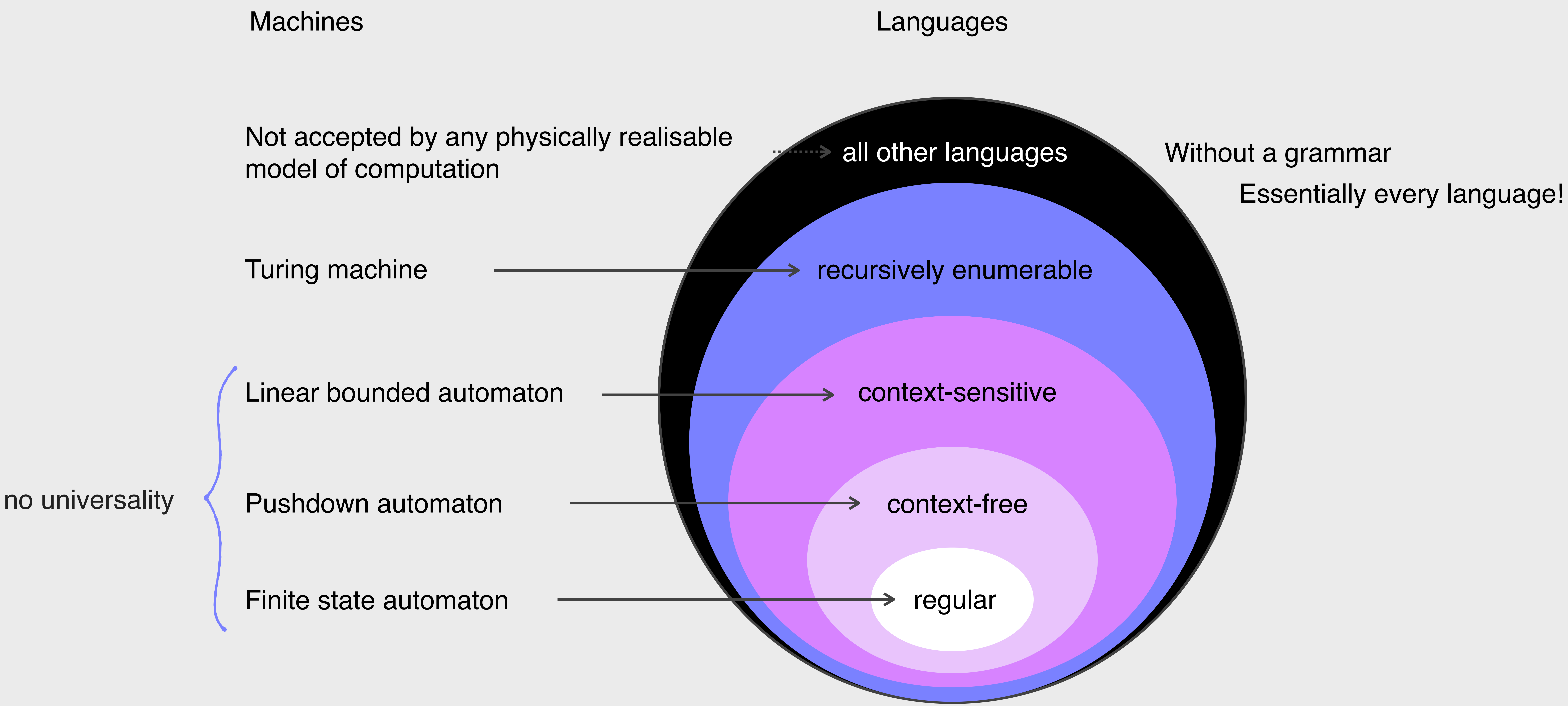
Machines and their languages

- A language accepted by a Turing machine has a **grammar**.
- A grammar is a finite set of production rules.

Machine M	Language $L(M)$
Program (finite set of transition rules)	**Grammar** (finite set of production rules)

They are a mould that can be applied an unbounded number of times

Weaker machines and their languages



Uncomputable functions

a.k.a. langauges without grammar

- A Turing machine computes a function $f : \mathbb{N} \to \{0,1\}$
- This function expresses an **attribute** of the natural numbers, i.e. a property that a natural number can have:

$$f(n) = \begin{cases} 1 & \text{if } n \text{ has that property} \\ 0 & \text{if it doesn't} \end{cases}$$

E.g. being an odd number, or being a prime.

- Every such f can be identified with its extension, $\{n \in \mathbb{N} \mid f(n) = 1\} \quad \in \wp(\mathbb{N})$
- Thus, there are as many attributes as elements of the power set, $|\wp(\mathbb{N})| = 2^{|\mathbb{N}|}$.
- A Turing machine can be encoded as a finite string, hence the number of Turing machines is $|\mathbb{N}|$.
- By Cantor's Theorem $|\mathbb{N}| < |\wp(\mathbb{N})|$
- So **essentially every such function is uncomputable**, i.e. essentially every attribute has no grammar.
- But the distribution of interesting functions / attributes is not uniform…

The halting problem

- An interesting, yet **uncomputable** problem.
- The halting problem: Given the code of a machine $d(T)$ and an input x, will T halt on x?
- Uncomputable, i.e. the language $L = \{d(T)\#x \mid T \text{ halts on } x\}$ is not decidable
    - Assume it were computable, i.e. there's a machine M that accepts every yes instance and rejects every no instance.
    - Construct a new machine P which halts if M does not halt, and does not halt if M halts.
    - Feed $d(P)\#d(P)$ to P
    - P halts if and only if P does not halt on P.
    - So P cannot exist, so M cannot exist.
- It is proven by self-reference and negation.
- It is one of the many incarnations of the liar paradox.