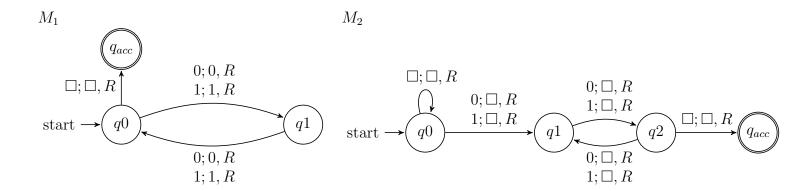
## Day20

| Acceptance problem         |           |   |
|----------------------------|-----------|---|
| for Turing machines        | $A_{TM}$  | $\{\langle M, w \rangle \mid M \text{ is a Turing machine that accepts input string } w\}$                |
| Language emptiness testing |           |   |
| for Turing machines        | $E_{TM}$  | $\{\langle M \rangle \mid M \text{ is a Turing machine and } L(M) = \emptyset\}$                          |
| Language equality testing  |           |   |
| for Turing machines        | $EQ_{TM}$ | $\{\langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are Turing machines and } L(M_1) = L(M_2)\}$ |



Example strings in  $A_{TM}$ 

Example strings in  $E_{TM}$ 

Example strings in  $EQ_{TM}$ 



To prove that a computational problem is **decidable**, we find/ build a Turing machine that recognizes the language encoding the computational problem, and that is a decider.

How do we prove a specific problem is **not decidable**?

How would we even find such a computational problem?

Counting arguments for the existence of an undecidable language:

- The set of all Turing machines is countably infinite.
- Each recognizable language has at least one Turing machine that recognizes it (by definition), so there can be no more Turing-recognizable languages than there are Turing machines.
- Since there are infinitely many Turing-recognizable languages (think of the singleton sets), there are countably infinitely many Turing-recognizable languages.
- Such the set of Turing-decidable languages is an infinite subset of the set of Turing-recognizable languages, the set of Turing-decidable languages is also countably infinite.

Since there are uncountably many languages (because  $\mathcal{P}(\Sigma^*)$  is uncountable), there are uncountably many unrecognizable languages and there are uncountably many undecidable languages.

Thus, there's at least one undecidable language!

## What's a specific example of a language that is unrecognizable or undecidable?

To prove that a language is undecidable, we need to prove that there is no Turing machine that decides it.

**Key idea**: proof by contradiction relying on self-referential disagreement.

**Theorem**:  $A_{TM}$  is not Turing-decidable.

**Proof**: Suppose towards a contradiction that there is a Turing machine that decides  $A_{TM}$ . We call this presumed machine  $M_{ATM}$ .

By assumption, for every Turing machine M and every string w

- If  $w \in L(M)$ , then the computation of  $M_{ATM}$  on  $\langle M, w \rangle$
- If  $w \notin L(M)$ , then the computation of  $M_{ATM}$  on  $\langle M, w \rangle$  \_\_\_\_\_\_

Define a **new** Turing machine using the high-level description:

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

- 1. Run  $M_{ATM}$  on  $\langle M, \langle M \rangle \rangle$ .
- 2. If  $M_{ATM}$  accepts, reject; if  $M_{ATM}$  rejects, accept."

| Is $D$ a Turing machine?  |  |  |  |
|---|--|--|--|
| Is $D$ a decider?   |  |  |  |
| What is the result of the computation of $D$ on $\langle D \rangle$ ? |  |  |  |
|   |  |  |  |
|   |  |  |  |
|   |  |  |  |
|   |  |  |  |
|   |  |  |  |
|   |  |  |  |
|   |  |  |  |
|   |  |  |  |
|   |  |  |  |
|   |  |  |  |

## **Summarizing:**

- $A_{TM}$  is recognizable.
- $A_{TM}$  is not decidable.

Recall definition: A language L over an alphabet  $\Sigma$  is called **co-recognizable** if its complement, defined as  $\Sigma^* \setminus L = \{x \in \Sigma^* \mid x \notin L\}$ , is Turing-recognizable.

and Recall Theorem (Sipser Theorem 4.22): A language is Turing-decidable if and only if both it and its complement are Turing-recognizable.

- $A_{TM}$  is recognizable.
- $A_{TM}$  is not decidable.
- $\overline{A_{TM}}$  is not recognizable.
- $\overline{A_{TM}}$  is not decidable.