Lecture 5: Decidability and Undecidability

Mika Göös

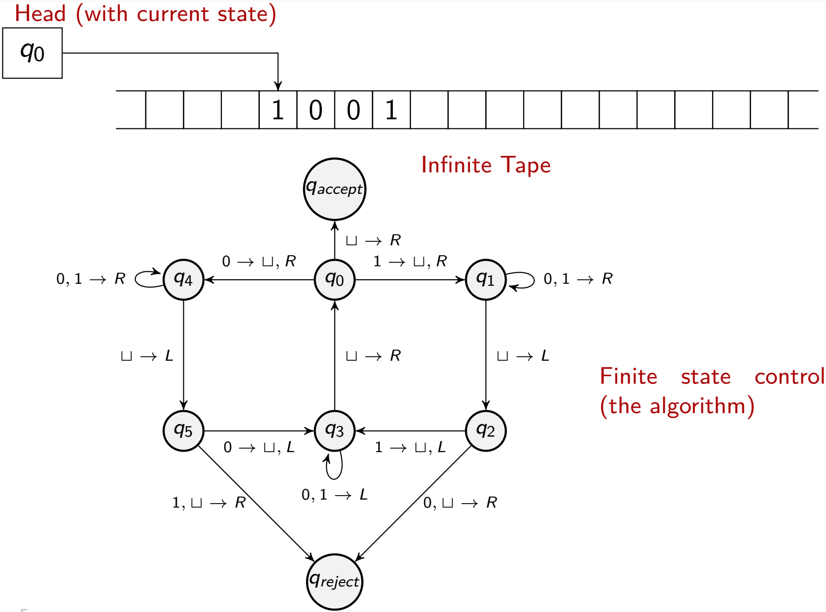
EPFL School of Computer and Communication Sciences

Lecture 5

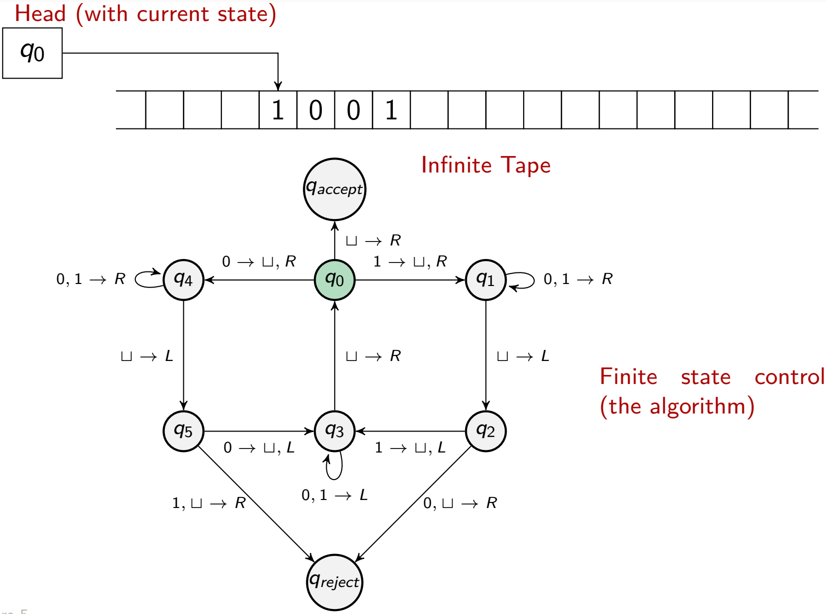
And thus the Turing Machine was born.

Recall

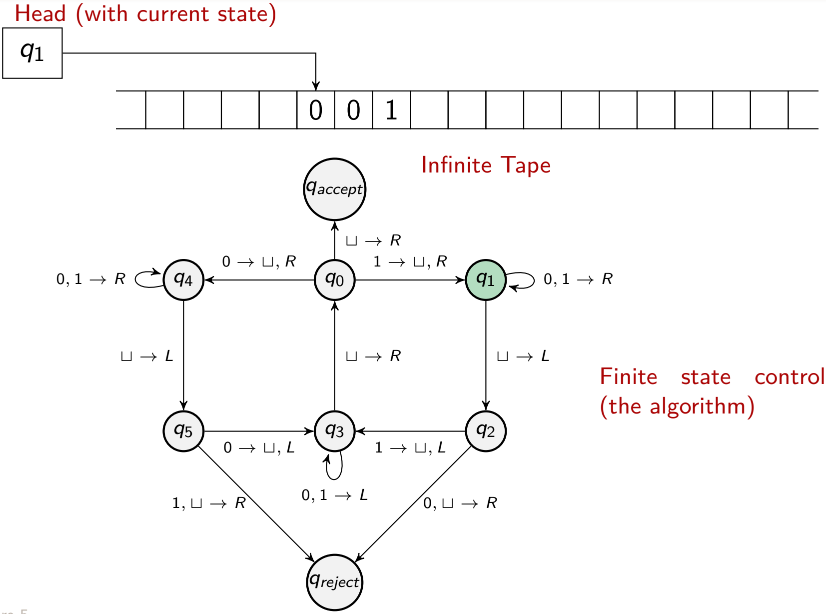
The Turing Machine



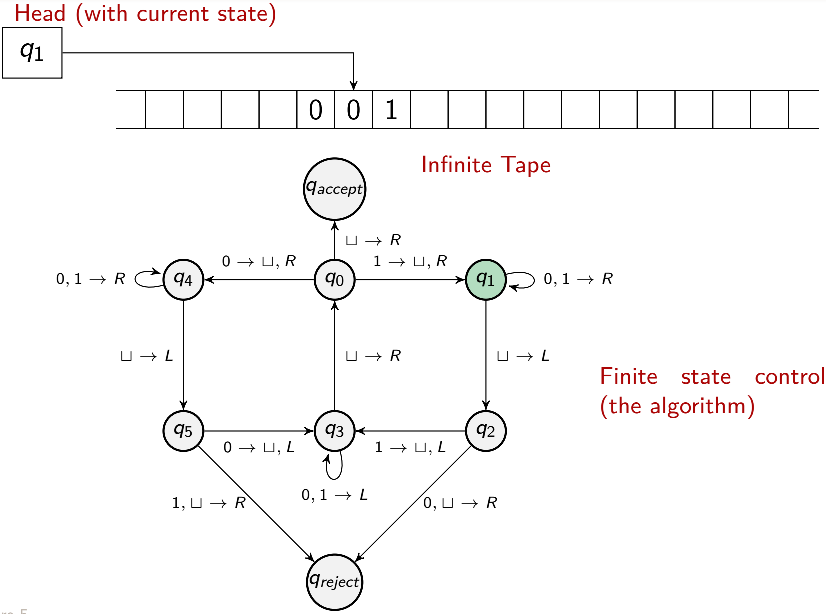
The Turing Machine



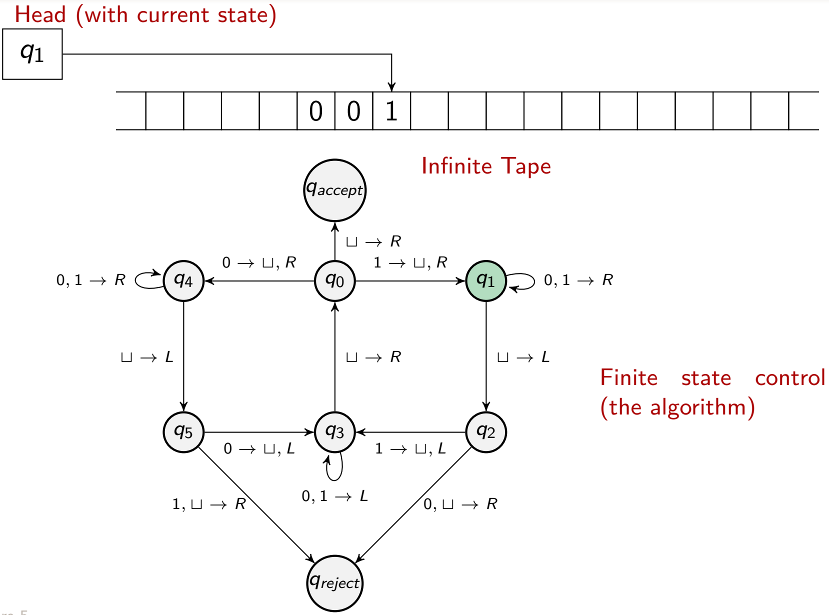
The Turing Machine



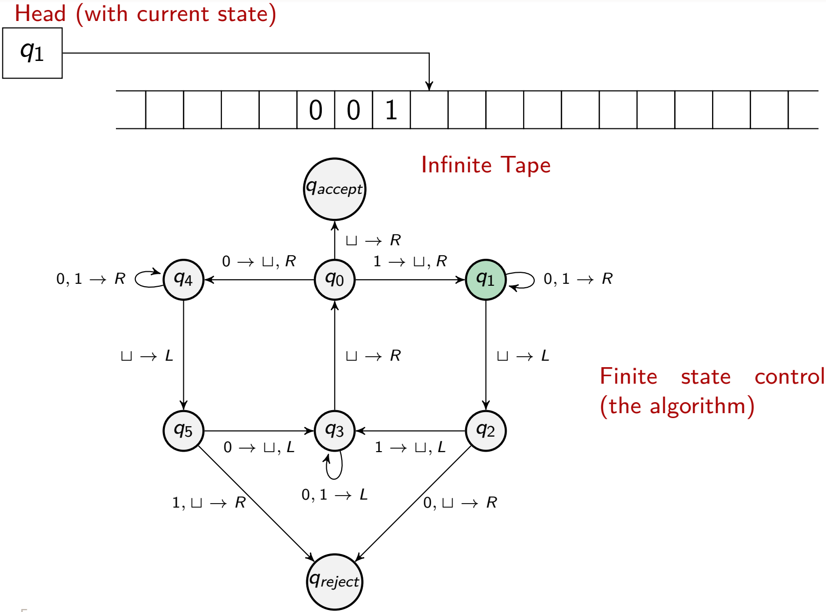
The Turing Machine



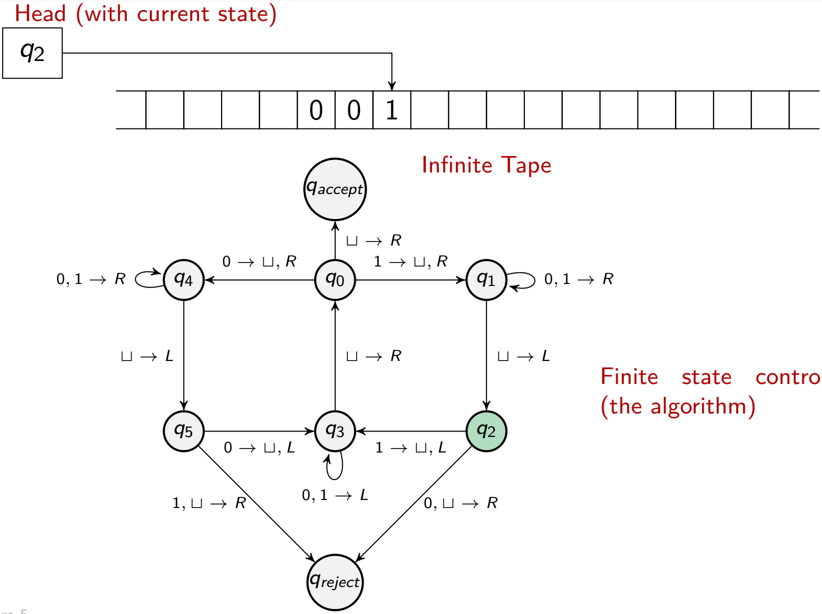
The Turing Machine



The Turing Machine

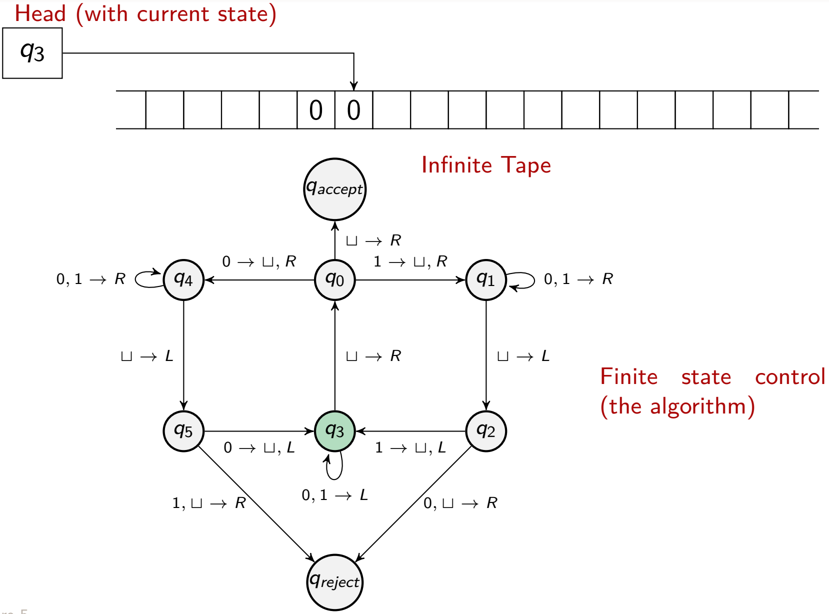


The Turing Machine

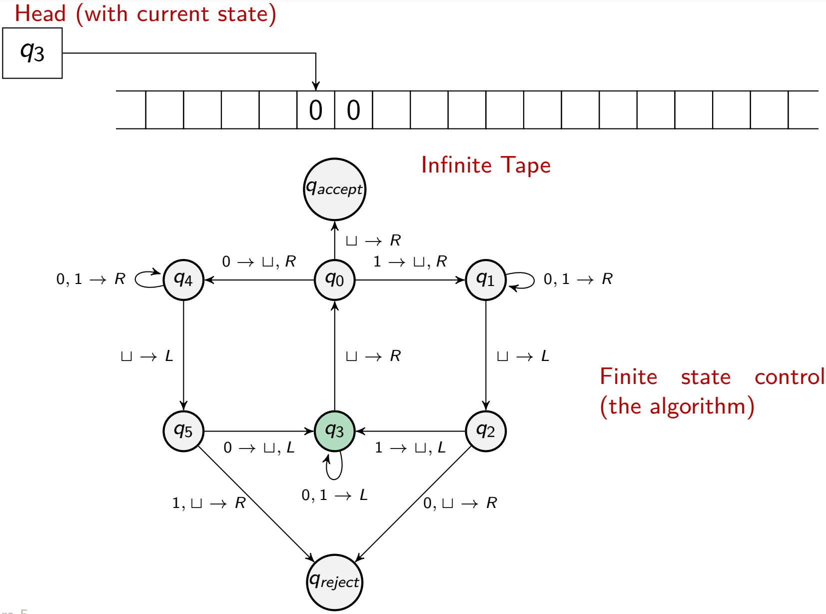


Finite state control (the algorithm)

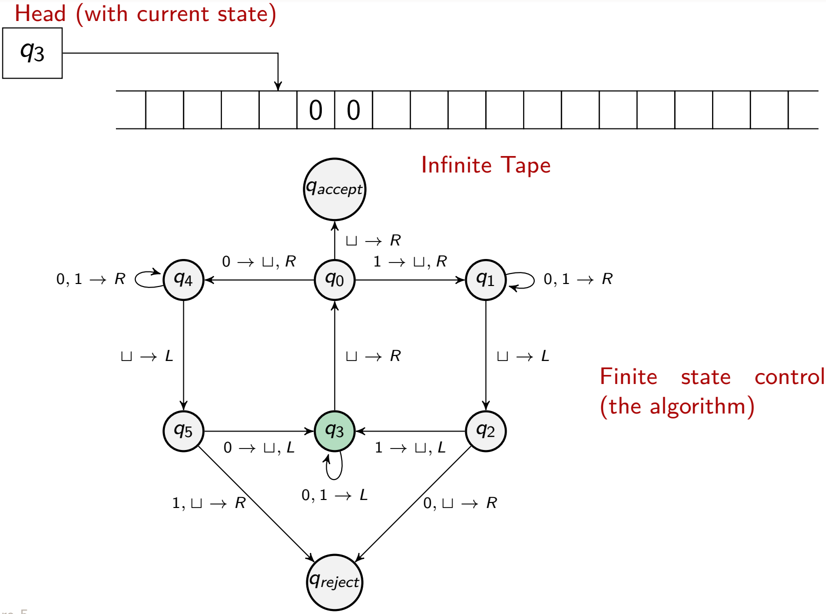
The Turing Machine



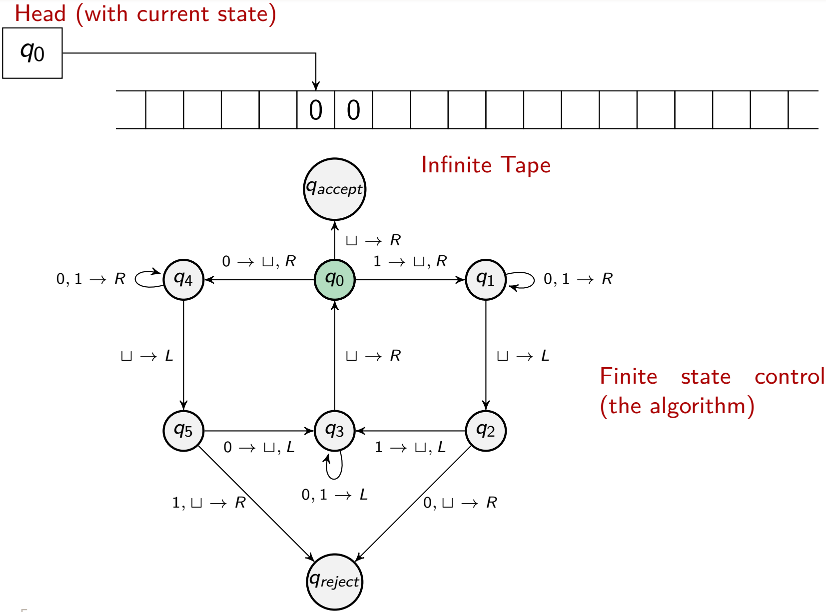
The Turing Machine



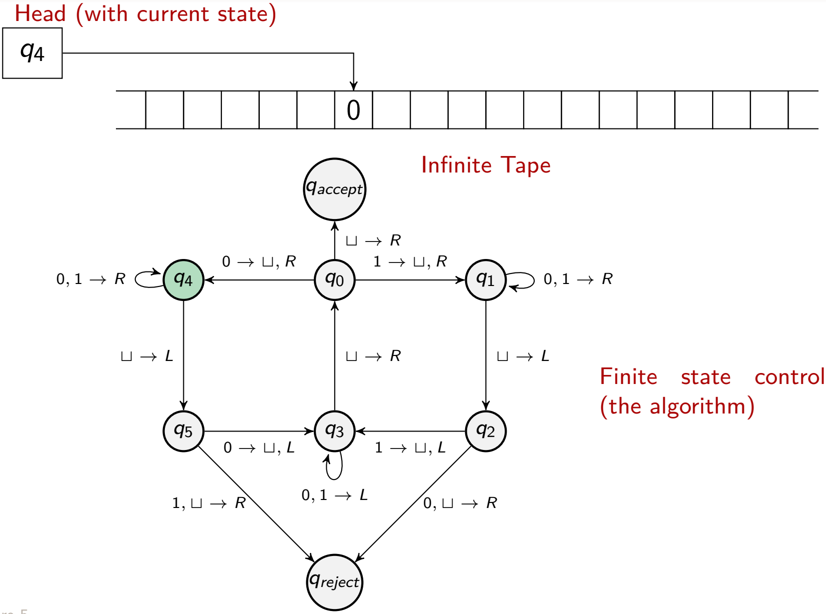
The Turing Machine



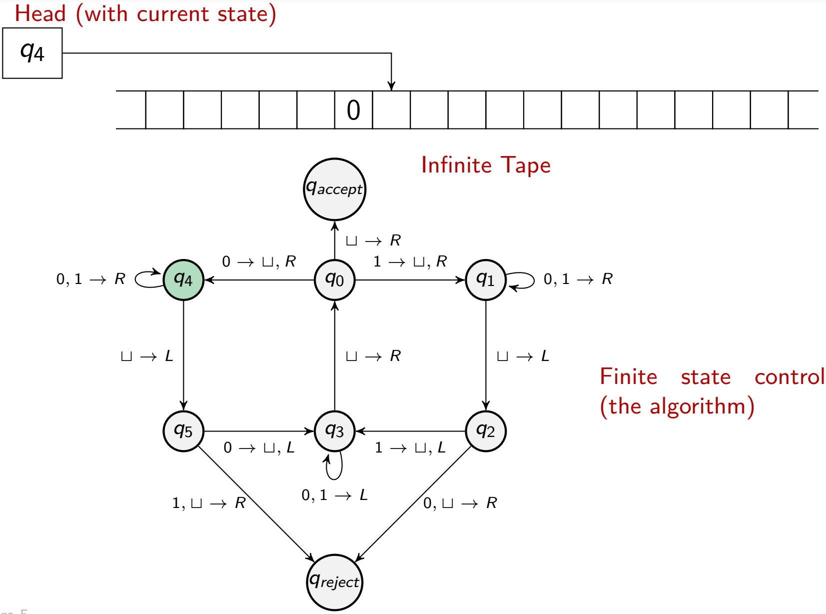
The Turing Machine



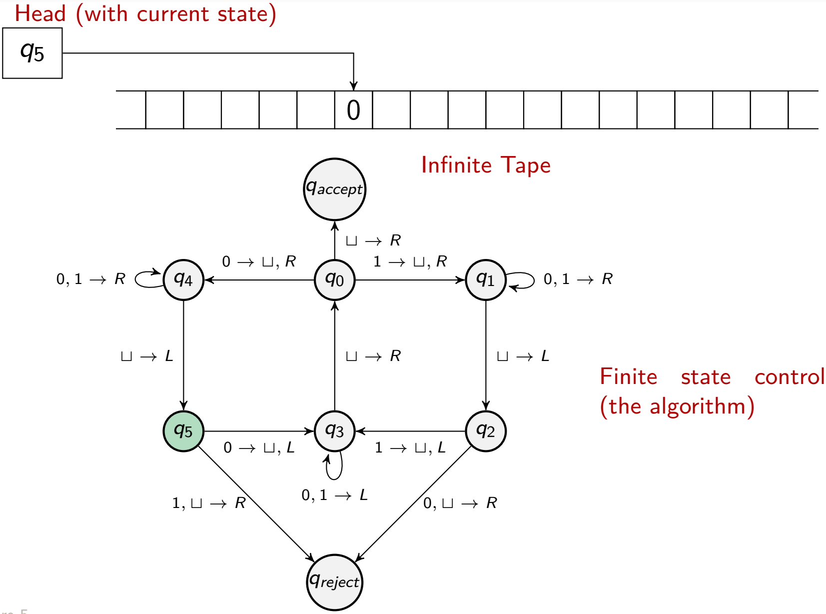
The Turing Machine



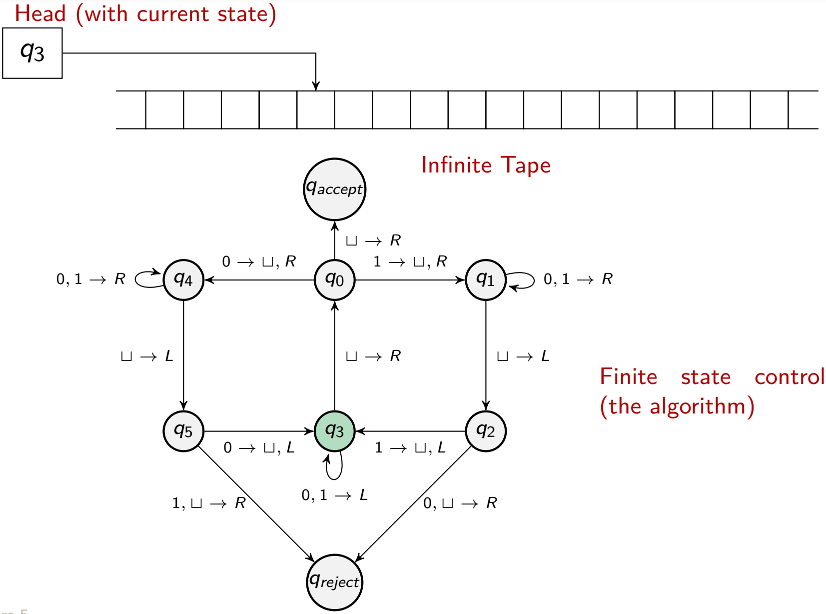
The Turing Machine



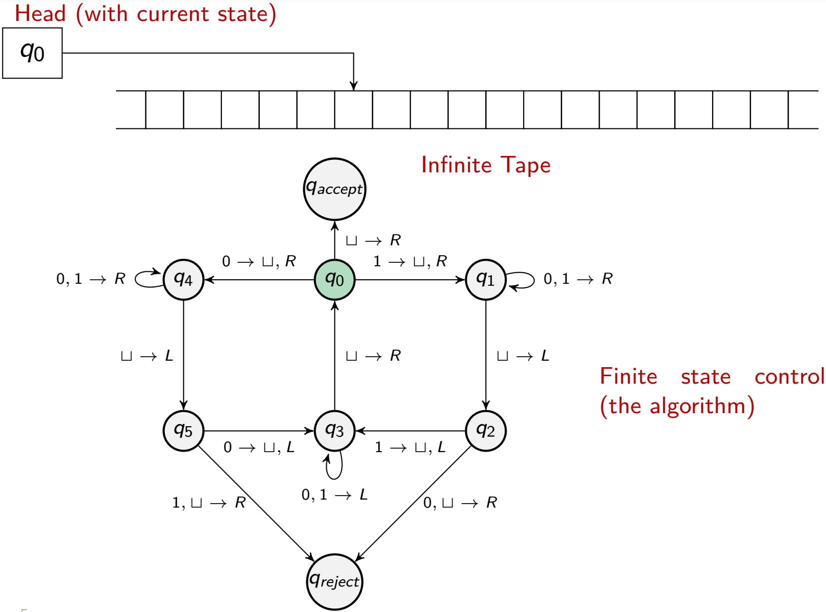
The Turing Machine



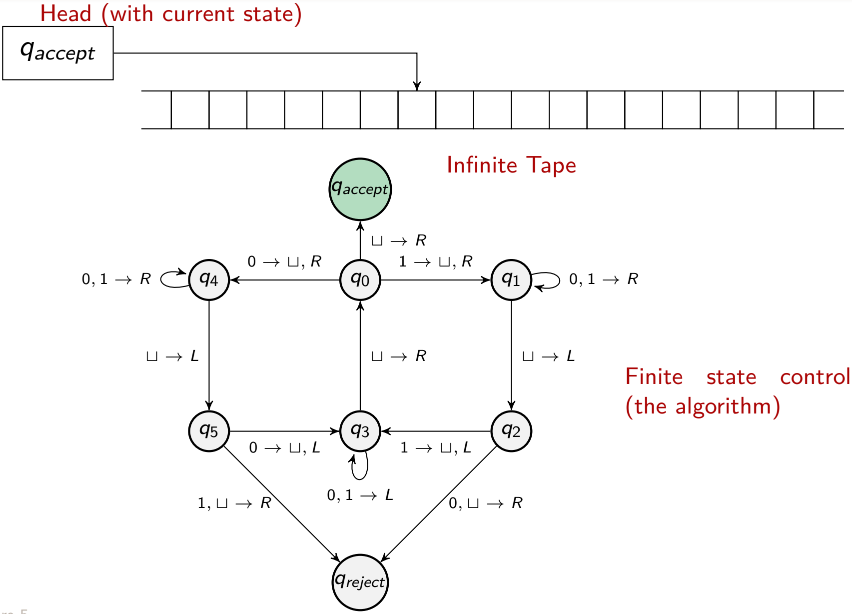
The Turing Machine



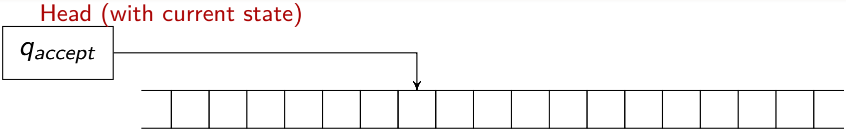
The Turing Machine

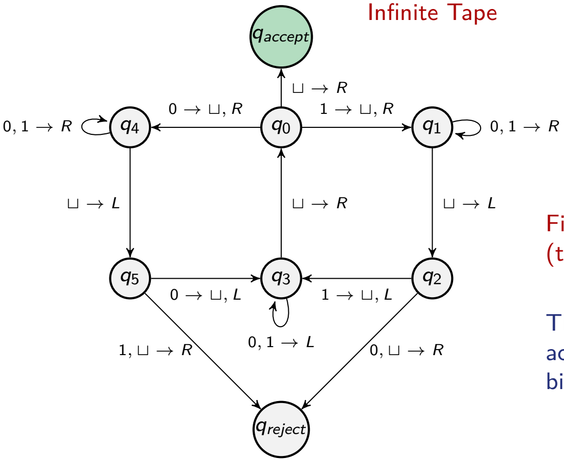


The Turing Machine



The Turing Machine

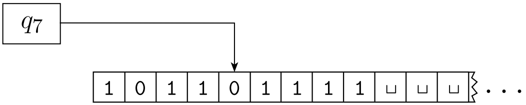




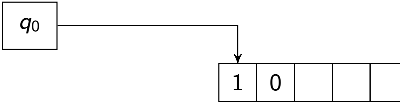
Finite state control (the algorithm)

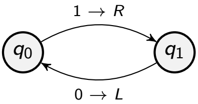
This Turing machine accepts even length binary palindromes

Turing Machine



- Infinite tape
- Tape alphabet contains input alphabet plus ⊔ (blank symbol) plus maybe more symbols
- Head has states (corresponding to the finite control automata)
- Exactly **one** Accept state and exactly **one** Reject state *(where computation immediately ends)*
- Remaining states *"computation in progress"*
- May never reach an accept state. May never halt!





Formal Definition of a TM

A **Turing Machine** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

Formal Definition of a TM

A **Turing Machine** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,

Formal Definition of a TM

A **Turing Machine** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the blank symbol $\sqcup$,

Formal Definition of a TM

A **Turing Machine** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the blank symbol $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,

Formal Definition of a TM

A **Turing Machine** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the blank symbol $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,

Formal Definition of a TM

A **Turing Machine** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the blank symbol $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state,

Formal Definition of a TM

A **Turing Machine** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the blank symbol $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{accept} \in Q$ is the accept state, and

Formal Definition of a TM

A **Turing Machine** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the blank symbol $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{accept} \in Q$ is the accept state, and
7. $q_{reject} \in Q$ is the reject state, where $q_{accept} \neq q_{reject}$.

Turing-Recognizable/Decidable Languages

A TM machine M recognizes a language $L \subseteq \Sigma^*$ iff for all inputs $w \in \Sigma^*$:

1. If $w \in L$ then M accepts w and
2. If $w \notin L$ then M either rejects w or never halts

Such languages are called (Turing)-Recognizable

Turing-Recognizable/Decidable Languages

A TM machine M recognizes a language $L \subseteq \Sigma^*$ iff for all inputs $w \in \Sigma^*$:

1. If $w \in L$ then M accepts w and
2. If $w \notin L$ then M either rejects w or never halts

Such languages are called (Turing)-Recognizable

A TM machine M decides a language $L \subseteq \Sigma^*$ iff for all inputs $w \in \Sigma^*$:

1. M halts on w, and
2. M accepts w iff $w \in L$

Such languages are called (Turing)-Decidable

Church-Turing Thesis

Intuitive notion of algorithms	equals	*Turing machine algorithms*

Church-Turing Thesis

Intuitive notion of algorithms	equals	*Turing machine algorithms*

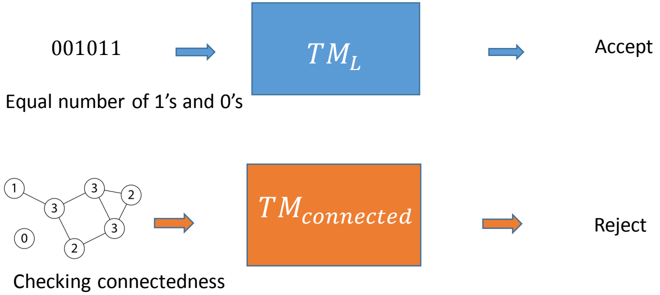
- All algorithms we know of can be executed on TMs
- Anything you write in C, Java, Scala, Python and so on
- The definition is also robust to variations: if we allow for many tapes instead of one, then nothing changes

To show language decidable or recognizable it sufficient to describe an algorithm in a high level language (Since by above any such algorithm can be run on a TM)

Decidable Languages

What can TMs compute?

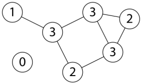
What can TMs compute?



What can TMs compute?

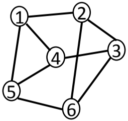
001011 → TM_L → Accept

Equal number of 1's and 0's

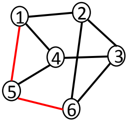


→ $TM_{connected}$ → Reject

Checking connectedness



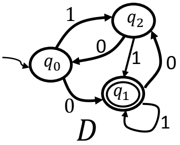
1,6 → $TM_{Shortestpath}$ →



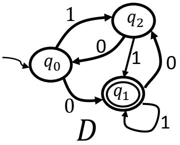
Shortest path problem

And more…

How to encode inputs

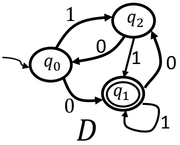


How to encode inputs



$\boldsymbol{\Sigma}$	$\boldsymbol{Q}$	$\boldsymbol{F}$	$\boldsymbol{q_0}$	*
q_0	1	q_2	*	
q_0	0	q_1	*	
q_1	0	q_2	*	
⋮	...			

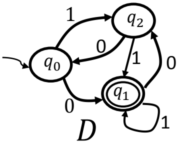
How to encode inputs



$\boldsymbol{\Sigma}$	$\boldsymbol{Q}$	$\boldsymbol{F}$	$\boldsymbol{q_0}$	*
q_0	1	q_2	*	
q_0	0	q_1	*	
q_1	0	q_2	*	
⋮	...			

Q	Σ	F	q_0	*	q_0	1	q_2	*	q_0	0	...

How to encode inputs



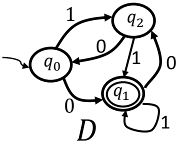
Σ	Q	F	q_0	*
q_0	1	q_2	*	
q_0	0	q_1	*	
q_1	0	q_2	*	
⋮	...			

Q	Σ	F	q_0	*	q_0	1	q_2	*	q_0	0	...

$\langle D \rangle \in \{0,1\}^\star$:

0	1	...	1	1	...	0	0	1	0	1	1	0	0	0	1	0	...	0	0	1	...

How to encode inputs



$\boldsymbol{\Sigma}$	$\boldsymbol{Q}$	$\boldsymbol{F}$	$\boldsymbol{q_0}$	$*$
q_0	1	q_2	$*$	
q_0	0	q_1	$*$	
q_1	0	q_2	$*$	
$\vdots$	…			

Q	Σ	F	q_0	$*$	q_0	1	q_2	$*$	q_0	0	…

$\langle \boldsymbol{D} \rangle \in \{\mathbf{0}, \mathbf{1}\}^{\star}$:

0	1	…	1	1	…	0	0	1	0	1	1	0	0	0	1	0	…	0	0	1	…

Anything can be encoded using binary strings!

Checking Emptiness of a DFA

$$E_{DFA} = \{\langle D \rangle : L(D) = \emptyset\}$$

Checking Emptiness of a DFA

$$E_{DFA} = \{\langle D \rangle : L(D) = \emptyset\}$$

First approach:

1. For each string $s \in \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$
2. Simulate D on s
   - If D accepts s, THEN *reject*
   - ELSE, pick the next string and go to 2.

Checking Emptiness of a DFA

$$E_{DFA} = \{\langle D \rangle : L(D) = \emptyset\}$$

First approach:

1. For each string $s \in \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$
2. Simulate D on s
   - If D accepts s, THEN *reject*
   - ELSE, pick the next string and go to 2.

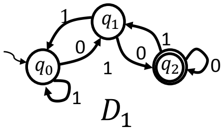
If $L(D) = \emptyset$, then this TM will never halt!

Checking Emptiness of a DFA (2nd attempt)

$$E_{DFA} = \{\langle D \rangle : L(D) = \emptyset\}$$

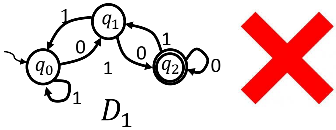
Checking Emptiness of a DFA (2nd attempt)

$$E_{DFA} = \{\langle D \rangle : L(D) = \emptyset\}$$



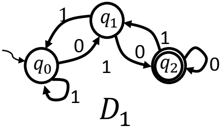
Checking Emptiness of a DFA (2nd attempt)

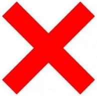
$$E_{DFA} = \{\langle D \rangle : L(D) = \emptyset\}$$

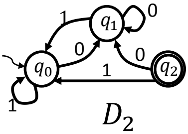


Checking Emptiness of a DFA (2nd attempt)

$$E_{DFA} = \{\langle D \rangle : L(D) = \emptyset\}$$

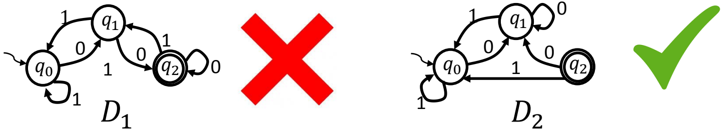






Checking Emptiness of a DFA (2nd attempt)

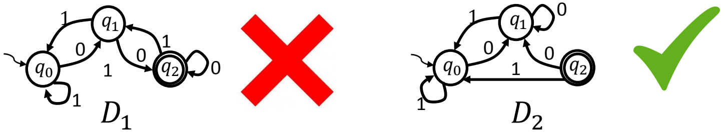
$$E_{DFA} = \{\langle D \rangle : L(D) = \emptyset\}$$



Language accepted by a DFA is non-empty iff there is an accepting state that can be reached from the starting state by a sequence of transitions

Checking Emptiness of a DFA (2nd attempt)

$$E_{DFA} = \{\langle D \rangle : L(D) = \emptyset\}$$



Language accepted by a DFA is non-empty iff there is an accepting state that can be reached from the starting state by a sequence of transitions

Given $\langle D \rangle$ with $D = (Q, \Sigma, \delta, q_0, F)$

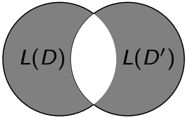
1. Initialize, $R = \{q_0\}$
2. For each $q \in R$ and $q' \in Q \setminus R$, check if there exists a transition of the form $\delta(q, a) = q'$ for some $a \in \Sigma$
3. If at least one such q' is found, add q' to R and go back to Step 2
4. Accept iff $R \cap F = \emptyset$

Two DFAs recognize the same language?

$$EQ_{DFA} = \{\langle D, D' \rangle : L(D) = L(D')\}$$

Two DFAs recognize the same language?

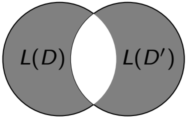
$$EQ_{DFA} = \{\langle D, D' \rangle : L(D) = L(D')\}$$



Fact: $L(D) = L(D')$ if and only if $L(D) \oplus L(D') = \emptyset$

Two DFAs recognize the same language?

$$EQ_{DFA} = \{\langle D, D' \rangle : L(D) = L(D')\}$$

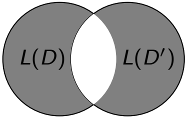


Fact: $L(D) = L(D')$ if and only if $L(D) \oplus L(D') = \emptyset$

- Here $L(D) \oplus L(D') = \left(L(D) \cap \overline{L(D')}\right) \cup \left(\overline{L(D)} \cap L(D')\right)$

Two DFAs recognize the same language?

$$EQ_{DFA} = \{\langle D, D' \rangle : L(D) = L(D')\}$$

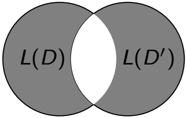


Fact: $L(D) = L(D')$ if and only if $L(D) \oplus L(D') = \emptyset$

- Here $L(D) \oplus L(D') = \left(L(D) \cap \overline{L(D')}\right) \cup \left(\overline{L(D)} \cap L(D')\right)$
- Thus $L(D) \oplus L(D')$ is regular

Two DFAs recognize the same language?

$$EQ_{DFA} = \{\langle D, D' \rangle : L(D) = L(D')\}$$



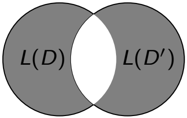
Fact: $L(D) = L(D')$ if and only if $L(D) \oplus L(D') = \emptyset$

- Here $L(D) \oplus L(D') = \left(L(D) \cap \overline{L(D')}\right) \cup \left(\overline{L(D)} \cap L(D')\right)$
- Thus $L(D) \oplus L(D')$ is regular
- Let $D^{\oplus}$ be DFA accepting $L(D) \oplus L(D')$

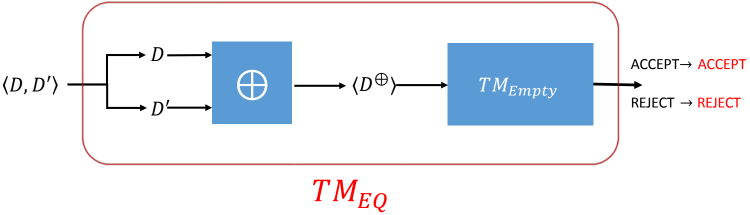
Two DFAs recognize the same language?

$$EQ_{DFA} = \{\langle D, D' \rangle : L(D) = L(D')\}$$

Fact: $L(D) = L(D')$ if and only if $L(D) \oplus L(D') = \emptyset$



- Here $L(D) \oplus L(D') = \left(L(D) \cap \overline{L(D')}\right) \cup \left(\overline{L(D)} \cap L(D')\right)$
- Thus $L(D) \oplus L(D')$ is regular
- Let $D^{\oplus}$ be DFA accepting $L(D) \oplus L(D')$



Mika, we get it: Turing machines can compute a lot

but can they compute anything?

Mika, we get it: Turing machines can compute a lot

but can they compute anything? NO

Theorem (Turing 1936): Halting problem

$$HALT = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ halts on input } w\}$$

is undecidable

Intuitively the difficulty of is to conclude in a finite number of steps whether M loops forever or is just slow to halt . . .

Halting problem is recognizable

$HALT = \{\langle M, w \rangle : M \text{ is a TM and } M \text{ halts on input } w\}$

Halting problem is recognizable

$$HALT = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ halts on input } w\}$$

$HALT$ is recognizable! The following TM U recognizes $HALT$

$U =$ 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 halts (enters accept/reject state), *accept*.

Halting problem is recognizable

$$HALT = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ halts on input } w\}$$

HALT is recognizable! The following TM U recognizes *HALT*

$U =$ 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 halts (enters accept/reject state), *accept*.

- Note that U loops on input $\langle M, w\rangle$ if M loops on w, which is why this machine does not decide *HALT*

Halting problem is recognizable

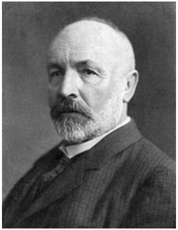
$$HALT = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ halts on input } w\}$$

$HALT$ is recognizable! The following TM U recognizes $HALT$

$U =$ 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 halts (enters accept/reject state), *accept*.

- Note that U loops on input $\langle M, w\rangle$ if M loops on w, which is why this machine does not decide $HALT$
- Since we will show that $HALT$ is undecidable this shows that recognizers *are* more powerful than deciders



George Cantor (1845–1918)

Diagonalization

Comparing the size of infinite sets

In 1873, Cantor thought about the size of infinite sets

Comparing the size of infinite sets

In 1873, Cantor thought about the size of infinite sets

- Is the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots, \}$ of the same size as the set of even numbers $\{2, 4, 6, \ldots, \ldots\}$?

Comparing the size of infinite sets

In 1873, Cantor thought about the size of infinite sets

- Is the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots, \}$ of the same size as the set of even numbers $\{2, 4, 6, \ldots, \ldots\}$?
- Is the size of $\mathbb{N}$ equal to the size of the set of rational numbers $\mathbb{Q} = \{\frac{m}{n} : m, n \in \mathbb{N}\}$?

Comparing the size of infinite sets

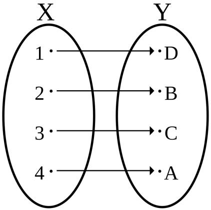
In 1873, Cantor thought about the size of infinite sets

- Is the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots, \}$ of the same size as the set of even numbers $\{2, 4, 6, \ldots, \ldots\}$?
- Is the size of $\mathbb{N}$ equal to the size of the set of rational numbers $\mathbb{Q} = \{\frac{m}{n} : m, n \in \mathbb{N}\}$?
- Is the size of $\mathbb{N}$ equal to the size of the set of real numbers $\mathbb{R}$?

Comparing the size of infinite sets

Recall that a function $f : X \to Y$ is bijective if it is

- **one-to-one** it never maps two different elements to the same place – that is, $f(a) \neq f(a')$ whenever $a \neq a'$
- **onto** it hits every element of Y – that is, for every $b \in Y$ there is an $a \in X$ such that $f(a) = b$



Definition. A set A is countable if either it is finite or it has the same size as $\mathbb{N}$ (i.e., there is a bijection between A and $\mathbb{N}$)

Comparing the size of infinite sets

In 1873, Cantor thought about the size of infinite sets

- Is the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots, \}$ of the same size as the set of even numbers $\{2, 4, 6, \ldots, \ldots\}$?

Comparing the size of infinite sets

In 1873, Cantor thought about the size of infinite sets

- Is the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots,\}$ of the same size as the set of even numbers $\{2, 4, 6, \ldots, \ldots\}$? YES

$\mathbb{N}$ and the set of even numbers are of the same size by the bijection $f(n) = 2n$:

n	$f(n)$
1	2
2	4
3	6
$\vdots$	$\vdots$

Comparing the size of infinite sets

In 1873, Cantor thought about the size of infinite sets

- Is the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots, \}$ of the same size as the set of even numbers $\{2, 4, 6, \ldots, \ldots\}$? YES
- Is the size of $\mathbb{N}$ equal to the size of the set of rational numbers $\mathbb{Q} = \{\frac{m}{n} : m, n \in \mathbb{N}\}$?

Comparing the size of infinite sets

In 1873, Cantor thought about the size of infinite sets

- Is the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots, \}$ of the same size as the set of even numbers $\{2, 4, 6, \ldots, \ldots\}$? YES
- Is the size of $\mathbb{N}$ equal to the size of the set of rational numbers $\mathbb{Q} = \{\frac{m}{n} : m, n \in \mathbb{N}\}$? YES

$\mathbb{N}$ and $\mathbb{Q}$ are of the same size by a bit more complex bijection (see book).

Comparing the size of infinite sets

In 1873, Cantor thought about the size of infinite sets

- Is the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots, \}$ of the same size as the set of even numbers $\{2, 4, 6, \ldots, \ldots\}$? YES
- Is the size of $\mathbb{N}$ equal to the size of the set of rational numbers $\mathbb{Q} = \{\frac{m}{n} : m, n \in \mathbb{N}\}$? YES

$\mathbb{N}$ and $\mathbb{Q}$ are of the same size by a bit more complex bijection (see book).

May seem bizzarre at first but incredibly influential!

> *"No one shall expel us from the paradise which Cantor has created for us."*
>
> *– David Hilbert*

Comparing the size of infinite sets

In 1873, Cantor thought about the size of infinite sets

- Is the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots, \}$ of the same size as the set of even numbers $\{2, 4, 6, \ldots, \ldots\}$? YES
- Is the size of $\mathbb{N}$ equal to the size of the set of rational numbers $\mathbb{Q} = \{\frac{m}{n} : m, n \in \mathbb{N}\}$? YES
- Is the size of $\mathbb{N}$ equal to the size of the set of real numbers $\mathbb{R}$?

Comparing the size of infinite sets

In 1873, Cantor thought about the size of infinite sets

- Is the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots, \}$ of the same size as the set of even numbers $\{2, 4, 6, \ldots, \ldots\}$? YES
- Is the size of $\mathbb{N}$ equal to the size of the set of rational numbers $\mathbb{Q} = \{\frac{m}{n} : m, n \in \mathbb{N}\}$? YES
- Is the size of $\mathbb{N}$ equal to the size of the set of real numbers $\mathbb{R}$? NO

Diagonalization

The reals are uncountable

- Suppose toward contradiction that there is a bijection $f : \mathbb{N} \to \mathbb{R}$
- We reach a contradiction by defining an $x \in \mathbb{R}$ such that no number $a \in \mathbb{N}$ maps to x, i.e., $f(a) = x$
- For every $n \in \mathbb{N}$, the n-th fractional digit of x is selected to be different from the n-th digit of $f(n)$
- Example:

n	$f(n)$
1	$3.\underline{1}4159\ldots$
2	$55.5\underline{5}555\ldots$
3	$0.12\underline{3}45\ldots$
4	$0.500\underline{0}0\ldots$
$\vdots$	$\vdots$

$$x = 0.4641\ldots$$

- By definition there is no $n \in \mathbb{N}$ such that $f(n) = x$ which contradicts that f is onto.

We now use Diagonalization to

- First prove that there is a language that is not decidable
- Then prove that the specific language

$$HALT = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ halts on } w\}$$

  is not decidable

There are undecidable languages

- **Turing machines are countable:** enumerate all encodings

There are undecidable languages

- **Turing machines are countable:** enumerate all encodings

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	$\langle M_6 \rangle$	$\dots$
M_1	A	∞	R	A	A	R	$\dots$
M_2	R	R	A	A	∞	A	$\dots$
M_3	R	∞	A	∞	R	R	$\dots$
M_4	A	∞	R	R	R	∞	$\dots$
M_5	∞	∞	A	A	A	A	$\dots$
M_6	R	A	R	∞	A	∞	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

There are undecidable languages

- **Turing machines are countable:** enumerate all encodings

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	$\langle M_6 \rangle$	...
M_1	A	∞	R	A	A	R	...
M_2	R	R	A	A	∞	A	...
M_3	R	∞	A	∞	R	R	...
M_4	A	∞	R	R	R	∞	...
M_5	∞	∞	A	A	A	A	...
M_6	R	A	R	∞	A	∞	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

There are undecidable languages

- **Turing machines are countable:** enumerate all encodings

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	$\langle M_6 \rangle$	$\dots$
M_1	A	∞	R	A	A	R	$\dots$
M_2	R	R	A	A	∞	A	$\dots$
M_3	R	∞	A	∞	R	R	$\dots$
M_4	A	∞	R	R	R	∞	$\dots$
M_5	∞	∞	A	A	A	A	$\dots$
M_6	R	A	R	∞	A	∞	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Let $DIAG = \{\langle M_i \rangle : M_i \text{ doesn't accept } \langle M_i \rangle\} = \{\langle M_2 \rangle, \langle M_4 \rangle, \langle M_6 \rangle, \dots\}$

There are undecidable languages

- **Turing machines are countable:** enumerate all encodings

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	$\langle M_6 \rangle$	$\dots$
M_1	A	∞	R	A	A	R	$\dots$
M_2	R	R	A	A	∞	A	$\dots$
M_3	R	∞	A	∞	R	R	$\dots$
M_4	A	∞	R	R	R	∞	$\dots$
M_5	∞	∞	A	A	A	A	$\dots$
M_6	R	A	R	∞	A	∞	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Let $DIAG = \{\langle M_i \rangle : M_i \text{ doesn't accept } \langle M_i \rangle\} = \{\langle M_2 \rangle, \langle M_4 \rangle, \langle M_6 \rangle, \dots\}$

Theorem *DIAG is undecidable*

Proof by contradiction

- Suppose M decided $DIAG$
- Then $M = M_i$ for some $i \in \mathbb{N}$
- We have $L(M_i) = DIAG$

Does $\langle M_i \rangle \in L(M_i)$? Two cases

- If $\langle M_i \rangle \in L(M_i)$ then by def. of $DIAG$, $\langle M_i \rangle \notin DIAG$ – contradiction
- If $\langle M_i \rangle \notin L(M_i)$ then by def. of $DIAG$, $\langle M_i \rangle \in DIAG$ – contradiction

Thm: $HALT = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ halts on } w\}$ is undecidable

Proof by contradiction

Assume on the contrary that H is a decider for $HALT$

We construct a decider D for $DIAG = \{\langle M\rangle : M \text{ doesn't accept } \langle M\rangle\}$

Thm: $HALT = \{\langle M, w\rangle : M$ is a TM and M halts on $w\}$ is undecidable

Proof by contradiction

Assume on the contrary that H is a decider for *HALT*

We construct a decider D for $DIAG = \{\langle M\rangle : M$ doesn't accept $\langle M\rangle\}$

- D: On input $\langle M\rangle$, run H on input $\langle M, \langle M\rangle\rangle$

Thm: $HALT = \{\langle M, w \rangle : M$ is a TM and M halts on $w\}$ is undecidable

Proof by contradiction

Assume on the contrary that H is a decider for *HALT*

We construct a decider D for $DIAG = \{\langle M \rangle : M$ doesn't accept $\langle M \rangle\}$

- D: On input $\langle M \rangle$, run H on input $\langle M, \langle M \rangle \rangle$
    1. If H rejects (i.e., M loops on $\langle M \rangle$), *accept*

Thm: $HALT = \{\langle M, w\rangle : M$ is a TM and M halts on $w\}$ is undecidable

Proof by contradiction

Assume on the contrary that H is a decider for *HALT*

We construct a decider D for $DIAG = \{\langle M\rangle : M$ doesn't accept $\langle M\rangle\}$

- D: On input $\langle M\rangle$, run H on input $\langle M, \langle M\rangle\rangle$
  1. If H rejects (i.e., M loops on $\langle M\rangle$), *accept*
  2. If H accepts (i.e., M halts on $\langle M\rangle$), run M on input $\langle M\rangle$. When M accepts/rejects, output the opposite.

Need to prove that D decides *DIAG*:

Thm: $HALT = \{\langle M, w\rangle : M$ is a TM and M halts on $w\}$ is undecidable

Proof by contradiction

Assume on the contrary that H is a decider for $HALT$

We construct a decider D for $DIAG = \{\langle M\rangle : M$ doesn't accept $\langle M\rangle\}$

- D: On input $\langle M\rangle$, run H on input $\langle M, \langle M\rangle\rangle$
  1. If H rejects (i.e., M loops on $\langle M\rangle$), *accept*
  2. If H accepts (i.e., M halts on $\langle M\rangle$), run M on input $\langle M\rangle$. When M accepts/rejects, output the opposite.

Need to prove that D decides $DIAG$:

1. D halts on all inputs

Thm: $HALT = \{\langle M, w\rangle : M$ is a TM and M halts on $w\}$ is undecidable

Proof by contradiction

Assume on the contrary that H is a decider for *HALT*

We construct a decider D for $DIAG = \{\langle M\rangle : M$ doesn't accept $\langle M\rangle\}$

- D: On input $\langle M\rangle$, run H on input $\langle M, \langle M\rangle\rangle$
  1. If H rejects (i.e., M loops on $\langle M\rangle$), *accept*
  2. If H accepts (i.e., M halts on $\langle M\rangle$), run M on input $\langle M\rangle$. When M accepts/rejects, output the opposite.

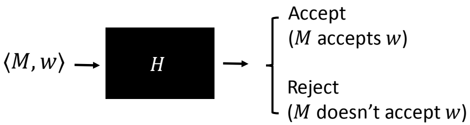
Need to prove that D decides *DIAG*:

1. D halts on all inputs
2. D accepts $\langle M\rangle \iff M$ does not accept $\langle M\rangle \iff \langle M\rangle \in DIAG$

Thm: $A_{TM} = \{\langle M, w \rangle : M \text{ is a TM and } M \text{ accepts } w\}$ is undecidable

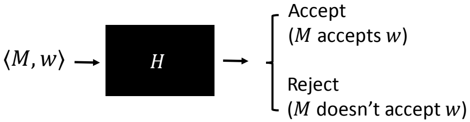
Thm: $A_{TM} = \{\langle M, w \rangle : M \text{ is a TM and } M \text{ accepts } w\}$ is undecidable

Proof by contradiction Assume on the contrary that H is a decider for A_{TM}

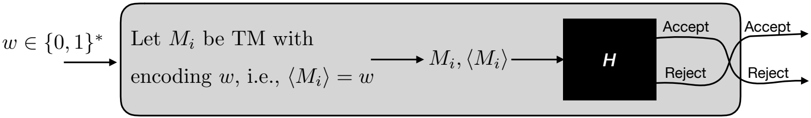


Thm: $A_{TM} = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ accepts } w\}$ is undecidable

Proof by contradiction Assume on the contrary that H is a decider for A_{TM}



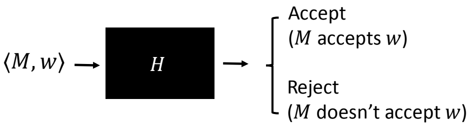
We construct a decider D for $DIAG = \{\langle M_i\rangle : M \text{ doesn't accept } \langle M_i\rangle\}$ using H:



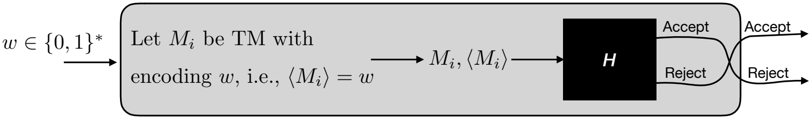
Need to prove that D decides *DIAG*:

Thm: $A_{TM} = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ accepts } w\}$ is undecidable

Proof by contradiction Assume on the contrary that H is a decider for A_{TM}



We construct a decider D for $DIAG = \{\langle M_i\rangle : M \text{ doesn't accept } \langle M_i\rangle\}$ using H:

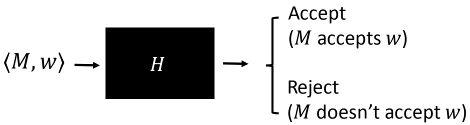


Need to prove that D decides *DIAG*:

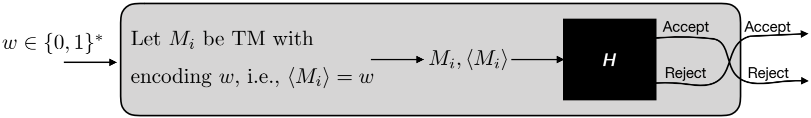
1. D halts on all inputs

Thm: $A_{TM} = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ accepts } w\}$ is undecidable

Proof by contradiction Assume on the contrary that H is a decider for A_{TM}

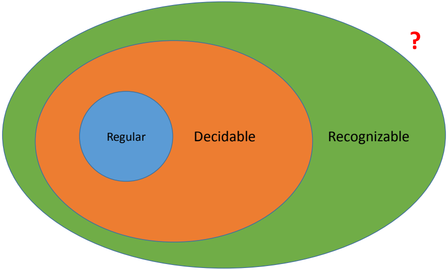


We construct a decider D for $DIAG = \{\langle M_i\rangle : M \text{ doesn't accept } \langle M_i\rangle\}$ using H:



Need to prove that D decides *DIAG*:

1. D halts on all inputs
2. D accepts $\langle M\rangle \iff M$ does not accept $\langle M\rangle \iff \langle M\rangle \in DIAG$



Unrecognizable languages?

Unrecognizable languages exist!

Thm: $\overline{HALT}$ is not recognizable

Step 1

Thm: A language L is decidable iff it is recognizable and its complement is also recognizable

Proof:

Step 1

Thm: A language L is decidable iff it is recognizable and its complement is also recognizable

Proof:

- Easy direction: If L is decidable $\Rightarrow L, \overline{L}$ are recognizable

Step 1

Thm: A language L is decidable iff it is recognizable and its complement is also recognizable

Proof:

- Easy direction: If L is decidable $\Rightarrow L, \overline{L}$ are recognizable
- Harder direction: Assume $L, \overline{L}$ are recognizable

Step 1

Thm: A language L is decidable iff it is recognizable and its complement is also recognizable

Proof:

- Easy direction: If L is decidable $\Rightarrow L, \overline{L}$ are recognizable
- Harder direction: Assume $L, \overline{L}$ are recognizable
- Let M_1, M_2 be the corresponding recognizers

Step 1

Thm: A language L is decidable iff it is recognizable and its complement is also recognizable

Proof:

- Easy direction: If L is decidable $\Rightarrow L, \overline{L}$ are recognizable
- Harder direction: Assume $L, \overline{L}$ are recognizable
- Let M_1, M_2 be the corresponding recognizers

Formally: we alternate between running 1 step of $\boldsymbol{M_1}$ and 1 step of $\boldsymbol{M_2}$

$\boldsymbol{M}$ = "On input **w**:
1- Run **both** $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$ on input $\boldsymbol{w}$, **in parallel**
2- If $\boldsymbol{M_1}$ accepts, ***accept***; if $\boldsymbol{M_2}$ accepts, ***reject***"

Step 1

Thm: A language L is decidable iff it is recognizable and its complement is also recognizable

Proof:

- Easy direction: If L is decidable $\Rightarrow L, \overline{L}$ are recognizable
- Harder direction: Assume $L, \overline{L}$ are recognizable
- Let M_1, M_2 be the corresponding recognizers

$\boldsymbol{M}$ = "On input **w**:
1- Run **both** $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$ on input $\boldsymbol{w}$, **in parallel**
2- If $\boldsymbol{M_1}$ accepts, ***accept***; if $\boldsymbol{M_2}$ accepts, ***reject***"

Formally: we alternate between running 1 step of $\boldsymbol{M_1}$ and 1 step of $\boldsymbol{M_2}$

- M decides L
  - For all w, either $w \in L$ or $w \in \bar{L}$
  - Either M_1 accepts w or M_2 accepts w
  - M halts once one of them stops $\Rightarrow$ M must always halt!

Step 2

Corollary: Language $\overline{HALT}$ is unrecognizable

- Recall that
  - $HALT$ is undecidable
  - $HALT$ is recognizable
- So by previous Thm we must have that $\overline{HALT}$ is unrecognizable

Next week: **Reductions**