Describing Turing Machines
  - Algorithm descriptions
Testing the Church-Turing Thesis
  - Nondeterministic Turing machines can be simulated by deterministic Turing machines
The method of diagonalization
  - Problems that cannot be solved by computers
Church-Turing thesis

Intuitive notion of computation

equals

Turing machine computation
The details of the computational model are not important.
Consequences of the Church-Turing thesis

- The details of the computational model are not important.
- Opens up the possibility to prove that some problems are not solvable by computers/Turing machines.
The details of the computational model are not important.
Opens up the possibility to prove that some problems are not solvable by computers/Turing machines.
Humans can be simulated by Turing machines!?
The universe is a Turing machine!?
Describing Turing machines

Example

Describe a Turing machine that recognizes the language $L = \{0^n1^n | n \geq 0\}$.

1. Scan the input from left to right and make sure it is of the form $0^*1^*2^*$ (if it is not, then reject)

2. Return the head to the left end of the tape

3. If there is no 0 on the tape, then scan right and check that there are no 1's and 2's on the tape and accept (should a 1 or 2 be on the tape, then reject)

4. Otherwise, cross off the first 0 and continue to the right
   cross off the first 1 and the first 2 that is found (should there be no 1 or no 2 on the tape, then reject)

5. Go to Step 2
Describe a Turing machine that recognizes the language $L = \{0^n 1^n 2^n \mid n \geq 0\}$.

1. Scan the input from left to right and make sure it is of the form $0^* 1^* 2^*$ (if it is not, then reject)
2. Return the head to the left end of the tape
3. If there is no 0 on the tape, then scan right and check that there are no 1's and 2's on the tape and accept (should a 1 or 2 be on the tape, then reject)
4. Otherwise, cross of the first 0 and continue to the right crossing of the first 1 and the first 2 that is found (should there be no 1 or no 2 on the tape, then reject)
5. Go to Step 2
Describing Turing machines

- Formal description (e.g., State diagram)
- Implementation level description
- Algorithm description
An algorithm description is a list of simple instructions for solving/computing some task.
An algorithm description is a list of simple instructions for solving/computing some task.

If the goal of the algorithm description is to convince the reader that the task can be solved/computed, then “simple instructions” means “can be carried out by a Turing machine”.
An algorithm description is a list of simple instructions for solving/computing some task.

If the goal of the algorithm description is to convince the reader that the task can be solved/computed, then “simple instructions” means “can be carried out by a Turing machine”.

Algorithm descriptions are similar to mathematical proofs
- The goal of a mathematical proof is to convince the reader that the truth of a mathematical statement follows from the basic axioms.
- The goal of an algorithm description is to convince the reader that a task/problem can be solved by Turing machines/computers.
Example

Describe an algorithm for recognizing the language
\[ L = \{0^n1^n2^n \mid n \geq 0\}. \]
Example

Describe an algorithm for recognizing the language 
\( L = \{0^n1^n2^n \mid n \geq 0\} \).

1. Check that the input is of the form 0*1*2*. Then count the number of 0’s, 1’s, and 2’s. If they are the same, accept. Otherwise, reject.
Algorithm

Muhammad ibn Musa al-Khwarizmi (780-850)
Theorem

Non-deterministic Turing machines can be simulated by deterministic Turing machines
A nondeterministic Turing machine is a 7-tuple \((Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})\) where

- \(Q\) is the finite set of states
- \(\Sigma\) is the finite input alphabet not containing the blank symbol \(B\)
- \(\Gamma\) is the finite tape alphabet where \(B \in \Gamma\) and \(\Sigma \subseteq \Gamma\)
- \(\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})\) is the transition function
- \(q_0 \in Q\) is the start state
- \(q_{\text{accept}} \in Q\) is the accept state
- \(q_{\text{reject}} \in Q\) is the reject state

A nondeterministic Turing machine accepts its input \(w\) if at least one of the states explored is an accept state.
**Definition**

A **nondeterministic Turing machine** is a 7-tuple

\[(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})\]

where

- \(Q\) is the finite set of states
- \(\Sigma\) is the finite input alphabet not containing the blank symbol \(B\)
- \(\Gamma\) is the finite tape alphabet where \(B \in \Gamma\) and \(\Sigma \subseteq \Gamma\)
- \(\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L,R\})\) is the transition function
- \(q_0 \in Q\) is the start state
- \(q_{\text{accept}} \in Q\) is the accept state
- \(q_{\text{reject}} \in Q\) is the reject state

A nondeterministic Turing machine accepts its input \(w\) if at least one of the states explored is an accept state.
Given a Turing machine $M$ operating on an input $w$: the current state, current tape contents, and current position of the read/write head is the current configuration of $M$. 
Given a Turing machine $M$ operating on an input $w$: the current state, current tape contents, and current position of the read/write head is the current configuration of $M$.

**Example**

$00q_510$ represent the configuration where the tape contents is $0010$, the state is $q_5$, and the position of the head is over the $1$. 
Given a Turing machine $M$ operating on an input $w$: the current state, current tape contents, and current position of the read/write head is the current configuration of $M$.

**Example**

$00q_510$ represent the configuration where the tape contents is $0010$, the state is $q_5$, and the position of the head is over the 1.

- The **start configuration** is $q_0w$
- An **accept configuration** is one where the state is $q_{\text{accept}}$
- A **reject configuration** is one where the state is $q_{\text{reject}}$
Theorem

Nondeterministic Turing machines can be simulated by deterministic Turing machines

Proof.

Given a nondeterministic Turing machine $N$ we construct a deterministic Turing machine $D$ such that $D$ accepts input $w$ if and only if $N$ accepts $w$. $D$ works as follows:
Theorem

Nondeterministic Turing machines can be simulated by deterministic Turing machines

Proof.

Given a nondeterministic Turing machine $N$ we construct a deterministic Turing machine $D$ such that $D$ accepts input $w$ if and only if $N$ accepts $w$. $D$ works as follows:

1. Given input $w$. Beginning with the start configuration of $N$ on input $w$, $D$ explores the computation tree of $N$ on input $w$.
2. If $D$ encounters an accept configuration of $N$, then $D$ accepts $w$.
3. If $D$ has explored the whole computational tree of $N$ without finding an accept configuration, then $D$ rejects $w$. 
Theorem

_Nondeterministic Turing machines can be simulated by deterministic Turing machines_

Proof.

Given a nondeterministic Turing machine $N$ we construct a deterministic Turing machine $D$ such that $D$ accepts input $w$ if and only if $N$ accepts $w$. $D$ works as follows:

1. Given input $w$. Beginning with the start configuration of $N$ on input $w$, $D$ explores the computation tree of $N$ on input $w$ in breadth first manner (i.e., level by level).
2. If $D$ encounters an accept configuration of $N$, then $D$ accepts $w$.
3. If $D$ has explored the whole computational tree of $N$ without finding an accept configuration, then $D$ rejects $w$. 
A_{DFA} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts input string } w \}
Decidable languages

\[ A_{DFA} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts input string } w \} \]

**Theorem**

\[ A_{DFA} \text{ is decidable} \]
Decidable languages

\[ A_{DFA} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts input string } w \} \]

**Theorem**

\( A_{DFA} \) is decidable

**Proof.**

Let \( M \) be a Turing machine that works as follows:

1. Check that the input \( \langle B, w \rangle \) is a legal encoding of a DFA \( B \) and string \( w \) (otherwise reject)
2. Simulate \( B \) on input \( w \)
3. If the simulation ends in an accept state (of \( B \)), then \( M \) accepts the input \( \langle B, w \rangle \)
4. If the simulation ends in a nonaccepting state (of \( B \)), then \( M \) rejects the input \( \langle B, w \rangle \)
Diagonalization

Georg Cantor (1845-1918)
Diagonalization

Georg Cantor (1845-1918)
Diagonalization

Theorem

Some languages are not Turing-recognizable
### Theorem

*Some languages are not Turing-recognizable*

### Proof idea.

1. The set of all Turing machines is countable
2. The set of all languages is uncountable
3. Since each Turing machine recognize exactly one language, there are languages that are not recognized by any Turing machine
Lemma

The set of all Turing machines is countable
Lemma

The set of all Turing machines is countable

Proof.

1. The set of all strings $\Sigma^*$ (for any alphabet $\Sigma$) is countable
   - A list of all strings in $\Sigma^*$ can be written down by listing all strings of length 0, length 1, length 2, ...

2. Each Turing machine $M$ can be encoded as a string $\langle M \rangle$ over $\Sigma$

3. By omitting those strings in $\Sigma^*$ which are not legal encodings of Turing machines, we get a list of all Turing machines
Diagonalization

Lemma

*The set of all languages over \( \Sigma \) is uncountable*

Proof idea.

1. Each language over \( \Sigma \) can be represented by its **characteristic sequence** (an infinite binary sequence). Each infinite binary sequence can be seen as a characteristic sequence for a language over \( \Sigma \). Hence, there is a one-to-one correspondence between infinite binary sequences and languages over \( \Sigma \).
Lemma

*The set of all languages over $\Sigma$ is uncountable*

Proof idea.

1. Each language over $\Sigma$ can be represented by its **characteristic sequence** (an infinite binary sequence). Each infinite binary sequence can be seen as a characteristic sequence for a language over $\Sigma$. Hence, there is a one-to-one correspondence between infinite binary sequences and languages over $\Sigma$.

2. The set of infinite binary sequences is uncountable (by a simple diagonalization proof)
Lemma

*The set of all languages over $\Sigma$ is uncountable*

Proof idea.

1. Each language over $\Sigma$ can be represented by its **characteristic sequence** (an infinite binary sequence). Each infinite binary sequence can be seen as a characteristic sequence for a language over $\Sigma$. Hence, there is a one-to-one correspondence between infinite binary sequences and languages over $\Sigma$.

2. The set of infinite binary sequences is uncountable (by a simple diagonalization proof)

3. Hence, the set of all languages over $\Sigma$ is uncountable
Theorem

Some languages are not Turing-recognizable

Proof.

1. The set of all Turing machines is countable
2. The set of all languages is uncountable
3. Since each Turing machine recognize exactly one language, there are languages that are not recognized by any Turing machine