# CS133 : Automata Theory and Computability

Lecture Slides

## Turing Machines and Undecidability

Francis George C. Cabarle
fccabarle@up.edu.ph

Algorithms and Complexity
Department of Computer Science
College of Engineering
University of the Philippines Diliman

Day 22

# Undecidability

QUESTION: Are there languages that are **NOT recursive** (i.e. not decidable)?

QUESTION: Are there languages that are **NOT recursive** (i.e. not decidable)? **Yup**.

QUESTION: Are there languages that are **NOT recursive** (i.e. not decidable)? **Yup**.

QUESTION: Are these non-$REC$ problems or languages esoteric, existing only in minds of theoreticians?

QUESTION: Are there languages that are **NOT recursive** (i.e. not decidable)? **Yup**.

QUESTION: Are these non-$REC$ problems or languages esoteric, existing only in minds of theoreticians? **Nope**.

EXAMPLE (*software verification in general*): Given an arbitrary computer program (e.g. sorting program), verify if said program is correct (i.e. performs sorting).

QUESTION: Are there languages that are **NOT recursive** (i.e. not decidable)? **Yup**.

QUESTION: Are these non-$REC$ problems or languages esoteric, existing only in minds of theoreticians? **Nope**.

EXAMPLE (*software verification in general*): Given an arbitrary computer program (e.g. sorting program), verify if said program is correct (i.e. performs sorting). In general, this "simple" task is *not theoretically decidable!*

QUESTION: Are there languages that are **NOT recursive** (i.e. not decidable)? **Yup**.

QUESTION: Are these non-$REC$ problems or languages esoteric, existing only in minds of theoreticians? **Nope**.

EXAMPLE (*software verification in general*): Given an arbitrary computer program (e.g. sorting program), verify if said program is correct (i.e. performs sorting). In general, this "simple" task is *not theoretically decidable!*

QUESTION: *Must* there really exist undecidable problems?

QUESTION: Are there languages that are **NOT recursive** (i.e. not decidable)? **Yup**.

QUESTION: Are these non-$REC$ problems or languages esoteric, existing only in minds of theoreticians? **Nope**.

EXAMPLE (*software verification in general*): Given an arbitrary computer program (e.g. sorting program), verify if said program is correct (i.e. performs sorting). In general, this "simple" task is *not theoretically decidable!*

QUESTION: *Must* there really exist undecidable problems? Are you sure? Are you really, really sure? In fact, a simple C program example can suffice...

The following pseudocode can be easily written in C or some programming language.

## Pseudocode:

Request user input $n$ for some $n \in \mathbb{N}$.
Let $total = 3$ and $solved = 0$, repeat the following until $solved = 1$:

- For $x$ in range $[1, \ldots, total - 2]$:
  - For $y$ in range $[1, \ldots, total - x - 1]$:
    - $z = total - x - y$.
    - If $x^n + y^n = z^n$ then $solved = 1$.
- $total \leftarrow total + 1$.

What does this pseudocode do?

The following pseudocode can be easily written in C or some programming language.

Pseudocode:

Request user input $n$ for some $n \in \mathbb{N}$.
Let $total = 3$ and $solved = 0$, repeat the following until $solved = 1$:

- For $x$ in range $[1, \ldots, total - 2]$:
  - For $y$ in range $[1, \ldots, total - x - 1]$:
    - $z = total - x - y$.
    - If $x^n + y^n = z^n$ then $solved = 1$.
- $total \leftarrow total + 1$.

What does this pseudocode do?

Will your program for this pseudocode halt for $n > 2$?...

## Definition

A set $A$ is **countable** if either it is finite or it has the same size as $\mathbb{N} = \{1, 2, 3, \ldots\}$.

Definition

A set $A$ is **countable** if either it is finite or it has the same size as $\mathbb{N} = \{1, 2, 3, \ldots\}$.

**Theorem:** $\mathbb{N}_e = \{\text{set of even natural numbers}\}$ is **countable**, i.e. same size as $\mathbb{N}$.

Cómo??? How???

Definition

A set $A$ is **countable** if either it is finite or it has the same size as $\mathbb{N} = \{1, 2, 3, \ldots\}$.

**Theorem:** $\mathbb{N}_e = \{\text{set of even natural numbers}\}$ is **countable**, i.e. same size as $\mathbb{N}$.

Cómo??? How???

**Theorem:** $\mathbb{Q}^+$, the set of positive rational numbers is **countable**.

How??? "Classic" counting/pairing technique do not suffice: **diagonalization technique** *al rescate!*

Definition

A set $A$ is **countable** if either it is finite or it has the same size as $\mathbb{N} = \{1, 2, 3, \ldots\}$.

**Theorem:** $\mathbb{N}_e = \{\text{set of even natural numbers}\}$ is **countable**, i.e. same size as $\mathbb{N}$.

Cómo??? How???

**Theorem:** $\mathbb{Q}^+$, the set of positive rational numbers is **countable**.

How??? "Classic" counting/pairing technique do not suffice: **diagonalization technique** *al rescate!*

**Theorem:** $\mathbb{R}$ is uncountable.

Definition
A set $A$ is **countable** if either it is finite or it has the same size as $\mathbb{N} = \{1, 2, 3, \ldots\}$.

**Theorem:** $\mathbb{N}_e = \{\text{set of even natural numbers}\}$ is **countable**, i.e. same size as $\mathbb{N}$.

Cómo??? How???

**Theorem:** $\mathbb{Q}^+$, the set of positive rational numbers is **countable**.

How??? "Classic" counting/pairing technique do not suffice:
**diagonalization technique** *al rescate!*

**Theorem:** $\mathbb{R}$ is uncountable.

Proof idea:
Suppose that a correspondence $f$ exists between $\mathbb{N}$ and $\mathbb{R}$.

Definition

A set $A$ is **countable** if either it is finite or it has the same size as $\mathbb{N} = \{1, 2, 3, \ldots\}$.

**Theorem:** $\mathbb{N}_e = \{\text{set of even natural numbers}\}$ is **countable**, i.e. same size as $\mathbb{N}$.

Cómo??? How???

**Theorem:** $\mathbb{Q}^+$, the set of positive rational numbers is **countable**.

How??? "Classic" counting/pairing technique do not suffice:
**diagonalization technique** *al rescate!*

**Theorem:** $\mathbb{R}$ is uncountable.

Proof idea:
Suppose that a correspondence $f$ exists between $\mathbb{N}$ and $\mathbb{R}$.
Then $f$ must pair all members of $\mathbb{N}$ will all members of $\mathbb{R}$.

Definition
A set $A$ is **countable** if either it is finite or it has the same size as $\mathbb{N} = \{1, 2, 3, \ldots\}$.

**Theorem:** $\mathbb{N}_e = \{\text{set of even natural numbers}\}$ is **countable**, i.e. same size as $\mathbb{N}$.

Cómo??? How???

**Theorem:** $\mathbb{Q}^+$, the set of positive rational numbers is **countable**.

How??? "Classic" counting/pairing technique do not suffice:
**diagonalization technique** *al rescate!*

**Theorem:** $\mathbb{R}$ is uncountable.

Proof idea:
Suppose that a correspondence $f$ exists between $\mathbb{N}$ and $\mathbb{R}$.
Then $f$ must pair all members of $\mathbb{N}$ will all members of $\mathbb{R}$.
We need to find an $x$ in $\mathbb{R}$ that is not paired with anything in $\mathbb{N}$.

Definition
A set $A$ is **countable** if either it is finite or it has the same size as $\mathbb{N} = \{1, 2, 3, \ldots\}$.

**Theorem:** $\mathbb{N}_e = \{\text{set of even natural numbers}\}$ is **countable**, i.e. same size as $\mathbb{N}$.

Cómo??? How???

**Theorem:** $\mathbb{Q}^+$, the set of positive rational numbers is **countable**.

How??? "Classic" counting/pairing technique do not suffice:
**diagonalization technique** *al rescate!*

**Theorem:** $\mathbb{R}$ is uncountable.

Proof idea:
Suppose that a correspondence $f$ exists between $\mathbb{N}$ and $\mathbb{R}$.
Then $f$ must pair all members of $\mathbb{N}$ will all members of $\mathbb{R}$.
We need to find an $x$ in $\mathbb{R}$ that is not paired with anything in $\mathbb{N}$.

**Theorem:** Language $L_d$ is not $RE$. That is, there is no Turing machine that accepts $L_d$.

**Theorem:** Language $L_d$ is not $RE$. That is, there is no Turing machine that accepts $L_d$.

- $\Sigma^*$ is countable, for any alphabet $\Sigma$.

**Theorem:** Language $L_d$ is not $RE$. That is, there is no Turing machine that accepts $L_d$.

- $\Sigma^*$ is countable, for any alphabet $\Sigma$.
  - With only finitely many strings of each length, we may form a list of $\Sigma^*$ by writing down all strings of length 0, length 1, length 2, and so on.

**Theorem:** Language $L_d$ is not $RE$. That is, there is no Turing machine that accepts $L_d$.

- $\Sigma^*$ is countable, for any alphabet $\Sigma$.
  - With only finitely many strings of each length, we may form a list of $\Sigma^*$ by writing down all strings of length 0, length 1, length 2, and so on.
- The set of all Turing machines is countable.

**Theorem:** Language $L_d$ is not $RE$. That is, there is no Turing machine that accepts $L_d$.

- $\Sigma^*$ is countable, for any alphabet $\Sigma$.
  - With only finitely many strings of each length, we may form a list of $\Sigma^*$ by writing down all strings of length 0, length 1, length 2, and so on.
- The set of all Turing machines is countable.
  - Each Turing machine has an encoding into a string $\langle M \rangle$. Simply omit those strings that are not legal encodings of Turing machines, we can obtain a list of all Turing machines.

**Theorem:** Language $L_d$ is not $RE$. That is, there is no Turing machine that accepts $L_d$.

- $\Sigma^*$ is countable, for any alphabet $\Sigma$.
  - With only finitely many strings of each length, we may form a list of $\Sigma^*$ by writing down all strings of length 0, length 1, length 2, and so on.
- The set of all Turing machines is countable.
  - Each Turing machine has an encoding into a string $\langle M \rangle$. Simply omit those strings that are not legal encodings of Turing machines, we can obtain a list of all Turing machines.
- Since we can count TMs, we can say $M_i$ is the $i$th TM, with *binary coding* $w_i$ (recall *standard description* used in Universal TM, for example).

**Theorem:** Language $L_d$ is not $RE$. That is, there is no Turing machine that accepts $L_d$.

- $\Sigma^*$ is countable, for any alphabet $\Sigma$.
  - With only finitely many strings of each length, we may form a list of $\Sigma^*$ by writing down all strings of length 0, length 1, length 2, and so on.
- The set of all Turing machines is countable.
  - Each Turing machine has an encoding into a string $\langle M \rangle$. Simply omit those strings that are not legal encodings of Turing machines, we can obtain a list of all Turing machines.
- Since we can count TMs, we can say $M_i$ is the $i$th TM, with *binary coding* $w_i$ (recall *standard description* used in Universal TM, for example).
- Define the *diagonalization language*

$$L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}.$$

Prove that $L_d$ is not $RE$. (see next slide)

**Theorem:** Language $L_d$ is not $RE$. That is, there is no Turing machine that accepts $L_d$.

- $\Sigma^*$ is countable, for any alphabet $\Sigma$.
  - With only finitely many strings of each length, we may form a list of $\Sigma^*$ by writing down all strings of length 0, length 1, length 2, and so on.
- The set of all Turing machines is countable.
  - Each Turing machine has an encoding into a string $\langle M \rangle$. Simply omit those strings that are not legal encodings of Turing machines, we can obtain a list of all Turing machines.
- Since we can count TMs, we can say $M_i$ is the $i$th TM, with *binary coding* $w_i$ (recall *standard description* used in Universal TM, for example).
- Define the *diagonalization language*

$$L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}.$$

  Prove that $L_d$ is not $RE$. (see next slide)
- Existence of a non-$RE$ language**s** such as $L_d$ (and other languages in $RE$ but not $REC$, more on these later) are some of the most *philosophically important* results in the theory of computation.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$. Thus, we can let $M$ be the $i$th TM, i.e. $M = M_i$.
4. Now we ask if input string $w_i$ is in $L_d$:

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$. Thus, we can let $M$ be the $i$th TM, i.e. $M = M_i$.
4. Now we ask if input string $w_i$ is in $L_d$:
    - If $w_i$ is in $L_d$, then the TM $M_i$ for $L_d$ **accepts** $w_i$.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$. Thus, we can let $M$ be the $i$th TM, i.e. $M = M_i$.
4. Now we ask if input string $w_i$ is in $L_d$:
    - If $w_i$ is in $L_d$, then the TM $M_i$ for $L_d$ **accepts** $w_i$.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$. Thus, we can let $M$ be the $i$th TM, i.e. $M = M_i$.
4. Now we ask if input string $w_i$ is in $L_d$:
   - If $w_i$ is in $L_d$, then the TM $M_i$ for $L_d$ **accepts** $w_i$. But by definition of $L_d$, string $w_i$ **cannot** be in $L_d$ since $L_d$ contains only strings $w_j$ where TM $M_j$ does **not** accept $w_j$.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$. Thus, we can let $M$ be the $i$th TM, i.e. $M = M_i$.
4. Now we ask if input string $w_i$ is in $L_d$:
   - If $w_i$ is in $L_d$, then the TM $M_i$ for $L_d$ **accepts** $w_i$. But by definition of $L_d$, string $w_i$ **cannot** be in $L_d$ since $L_d$ contains only strings $w_j$ where TM $M_j$ does **not** accept $w_j$. *Contradiction!*
   - If $w_i$ is not in $L_d$ then $M_i$ does not accept $w_i$.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$. Thus, we can let $M$ be the $i$th TM, i.e. $M = M_i$.
4. Now we ask if input string $w_i$ is in $L_d$:
   - If $w_i$ is in $L_d$, then the TM $M_i$ for $L_d$ **accepts** $w_i$. But by definition of $L_d$, string $w_i$ **cannot** be in $L_d$ since $L_d$ contains only strings $w_j$ where TM $M_j$ does **not** accept $w_j$. *Contradiction!*
   - If $w_i$ is not in $L_d$ then $M_i$ does not accept $w_i$.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$. Thus, we can let $M$ be the $i$th TM, i.e. $M = M_i$.
4. Now we ask if input string $w_i$ is in $L_d$:
   - If $w_i$ is in $L_d$, then the TM $M_i$ for $L_d$ **accepts** $w_i$. But by definition of $L_d$, string $w_i$ **cannot** be in $L_d$ since $L_d$ contains only strings $w_j$ where TM $M_j$ does **not** accept $w_j$. *Contradiction!*
   - If $w_i$ is not in $L_d$ then $M_i$ does not accept $w_i$. But by definition of $L_d$, string $w_i$ **is in** $L_d$.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$. Thus, we can let $M$ be the $i$th TM, i.e. $M = M_i$.
4. Now we ask if input string $w_i$ is in $L_d$:
   - If $w_i$ is in $L_d$, then the TM $M_i$ for $L_d$ **accepts** $w_i$. But by definition of $L_d$, string $w_i$ **cannot** be in $L_d$ since $L_d$ contains only strings $w_j$ where TM $M_j$ does **not** accept $w_j$. *Contradiction!*
   - If $w_i$ is not in $L_d$ then $M_i$ does not accept $w_i$. But by definition of $L_d$, string $w_i$ **is in** $L_d$. *Contradiction!*
5. One of these two cases must be true, but both cases lead to a contradiction.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$. Thus, we can let $M$ be the $i$th TM, i.e. $M = M_i$.
4. Now we ask if input string $w_i$ is in $L_d$:
   - If $w_i$ is in $L_d$, then the TM $M_i$ for $L_d$ **accepts** $w_i$. But by definition of $L_d$, string $w_i$ **cannot** be in $L_d$ since $L_d$ contains only strings $w_j$ where TM $M_j$ does **not** accept $w_j$. *Contradiction!*
   - If $w_i$ is not in $L_d$ then $M_i$ does not accept $w_i$. But by definition of $L_d$, string $w_i$ **is in** $L_d$. *Contradiction!*
5. One of these two cases must be true, but both cases lead to a contradiction.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$. Thus, we can let $M$ be the $i$th TM, i.e. $M = M_i$.
4. Now we ask if input string $w_i$ is in $L_d$:
   - If $w_i$ is in $L_d$, then the TM $M_i$ for $L_d$ **accepts** $w_i$. But by definition of $L_d$, string $w_i$ **cannot** be in $L_d$ since $L_d$ contains only strings $w_j$ where TM $M_j$ does **not** accept $w_j$. *Contradiction!*
   - If $w_i$ is not in $L_d$ then $M_i$ does not accept $w_i$. But by definition of $L_d$, string $w_i$ **is in** $L_d$. *Contradiction!*
5. One of these two cases must be true, but both cases lead to a contradiction. Our assumption that TM $M_i = M$ for $L_d$ exists is **false**.

**Theorem:** Let $L_d = \{w_i \mid w_i \notin L(M_i) \text{ and } w_i = \langle M_i \rangle\}$.
Then, $L_d \notin RE$.

Proof:

1. N.t.s. that there is no TM that accepts $L_d$.
2. Assume $L_d \in RE$, i.e. $L_d = L(M)$ for some TM $M$.
3. Since $L_d$ is a language over $\Sigma = \{0, 1\}$, $M$ is one of the TMs in our list of all possible TMs over $\Sigma$. Thus, we can let $M$ be the $i$th TM, i.e. $M = M_i$.
4. Now we ask if input string $w_i$ is in $L_d$:
   - If $w_i$ is in $L_d$, then the TM $M_i$ for $L_d$ **accepts** $w_i$. But by definition of $L_d$, string $w_i$ **cannot** be in $L_d$ since $L_d$ contains only strings $w_j$ where TM $M_j$ does **not** accept $w_j$. *Contradiction!*
   - If $w_i$ is not in $L_d$ then $M_i$ does not accept $w_i$. But by definition of $L_d$, string $w_i$ **is in** $L_d$. *Contradiction!*
5. One of these two cases must be true, but both cases lead to a contradiction. Our assumption that TM $M_i = M$ for $L_d$ exists is **false**. Hence, $L_d \notin RE$.

## Recap before we proceed

- There are uncountably many languages.
- Only countably many TMs.
- Each Turing machine can recognize a single language.
- There are more languages than TMs.
- Some languages are not $RE$, i.e. not recognized by any TM, e.g. $L_d$.
- Some languages are recursive (decidable) but not recursively enumerable, e.g. $L_u$ (more later)

## More results on (un)solvability

**Theorem:** $REC$ is closed under complementation.

## More results on (un)solvability

**Theorem:** $REC$ is closed under complementation.

i.e. If $L \in REC$, then so is $\overline{L}$. Proof?

## More results on (un)solvability

**Theorem:** $REC$ is closed under complementation.

i.e. If $L \in REC$, then so is $\overline{L}$. Proof?

**Theorem:** If languages $L, \overline{L} \in RE$ , then $L \in REC$.

## More results on (un)solvability

**Theorem:** $REC$ is closed under complementation.

i.e. If $L \in REC$, then so is $\overline{L}$. Proof?

**Theorem:** If languages $L, \overline{L} \in RE$ , then $L \in REC$.

note: by previous Theorem, $\overline{L} \in REC$ also. Proof?

## More results on (un)solvability

**Theorem:** $REC$ is closed under complementation.

i.e. If $L \in REC$, then so is $\overline{L}$. Proof?

**Theorem:** If languages $L, \overline{L} \in RE$ , then $L \in REC$.

note: by previous Theorem, $\overline{L} \in REC$ also. Proof?

From both Theorems, only four possible placements of $L$ and $\overline{L}$ in $REC$ and $RE$:

- $L, \overline{L} \in REC$;
- $L, \overline{L} \notin RE$;
- $L \in RE, L \notin REC$ and $\overline{L} \notin RE$;
- $\overline{L} \in RE, \overline{L} \notin REC$ and $L \notin RE$.

## Back to UTM

Recall that a Universal TM can be a "stored program computer": TM that takes as input other TMs and their inputs.

## Back to UTM

Recall that a Universal TM can be a "stored program computer": TM that takes as input other TMs and their inputs.
Define $L_u$, the *universal language*, to be the set

$$L_u = \{\langle M, w\rangle \mid M \text{ is a TM and } w \in L(M)\}.$$

## Back to UTM

Recall that a Universal TM can be a "stored program computer": TM that takes as input other TMs and their inputs.
Define $L_u$, the *universal language*, to be the set

$$L_u = \{\langle M, w\rangle \mid M \text{ is a TM and } w \in L(M)\}.$$

$L_u$ is useful when proving another problem $P$ is undecidable, by reduction of $L_u$ to $P$ (more on future lectures).

## Back to UTM

Recall that a Universal TM can be a "stored program computer": TM that takes as input other TMs and their inputs.
Define $L_u$, the *universal language*, to be the set

$$L_u = \{\langle M, w\rangle \mid M \text{ is a TM and } w \in L(M)\}.$$

$L_u$ is useful when proving another problem $P$ is undecidable, by reduction of $L_u$ to $P$ (more on future lectures).
Let $L_u = L(U)$ for a UTM $U$.

## Back to UTM

Recall that a Universal TM can be a "stored program computer": TM that takes as input other TMs and their inputs.
Define $L_u$, the *universal language*, to be the set

$$L_u = \{\langle M, w\rangle \mid M \text{ is a TM and } w \in L(M)\}.$$

$L_u$ is useful when proving another problem $P$ is undecidable, by reduction of $L_u$ to $P$ (more on future lectures).
Let $L_u = L(U)$ for a UTM $U$.

**Theorem:** $L_u$ is in $RE$ but not in $REC$.

## Back to UTM

Recall that a Universal TM can be a "stored program computer": TM that takes as input other TMs and their inputs.
Define $L_u$, the *universal language*, to be the set

$$L_u = \{\langle M, w\rangle \mid M \text{ is a TM and } w \in L(M)\}.$$

$L_u$ is useful when proving another problem $P$ is undecidable, by reduction of $L_u$ to $P$ (more on future lectures).
Let $L_u = L(U)$ for a UTM $U$.

**Theorem:** $L_u$ is in $RE$ but not in $REC$.

Proof idea: Easy to see that $L_u$ is $RE$ (really?).

## Back to UTM

Recall that a Universal TM can be a "stored program computer": TM that takes as input other TMs and their inputs.
Define $L_u$, the *universal language*, to be the set

$$L_u = \{\langle M, w\rangle \mid M \text{ is a TM and } w \in L(M)\}.$$

$L_u$ is useful when proving another problem $P$ is undecidable, by reduction of $L_u$ to $P$ (more on future lectures).
Let $L_u = L(U)$ for a UTM $U$.

**Theorem:** $L_u$ is in $RE$ but not in $REC$.

Proof idea: Easy to see that $L_u$ is $RE$ (really?).
Suppose $L_u$ is $REC$, then $\overline{L_u}$ must be $REC$ as well!

## Back to UTM

Recall that a Universal TM can be a "stored program computer": TM that takes as input other TMs and their inputs.
Define $L_u$, the *universal language*, to be the set

$$L_u = \{\langle M, w\rangle \mid M \text{ is a TM and } w \in L(M)\}.$$

$L_u$ is useful when proving another problem $P$ is undecidable, by reduction of $L_u$ to $P$ (more on future lectures).
Let $L_u = L(U)$ for a UTM $U$.

**Theorem:** $L_u$ is in $RE$ but not in $REC$.

Proof idea: Easy to see that $L_u$ is $RE$ (really?).
Suppose $L_u$ is $REC$, then $\overline{L_u}$ must be $REC$ as well!
How to arrive at a contradiction, i.e. $L_u$ cannot be $REC$?

## Back to UTM

Recall that a Universal TM can be a "stored program computer": TM that takes as input other TMs and their inputs.
Define $L_u$, the *universal language*, to be the set

$$L_u = \{\langle M, w\rangle \mid M \text{ is a TM and } w \in L(M)\}.$$

$L_u$ is useful when proving another problem $P$ is undecidable, by reduction of $L_u$ to $P$ (more on future lectures).
Let $L_u = L(U)$ for a UTM $U$.

**Theorem:** $L_u$ is in $RE$ but not in $REC$.

Proof idea: Easy to see that $L_u$ is $RE$ (really?).
Suppose $L_u$ is $REC$, then $\overline{L_u}$ must be $REC$ as well!
How to arrive at a contradiction, i.e. $L_u$ cannot be $REC$?
Proof involves **reduction** of (the problem) $L_d$ to (the problem) $\overline{L_u}$.

## Solvability of politicians...

Q: What do you call a politician that when given the correct amount of money, is able to finish his/her tasks, but it is unknown whether his/her tasks will finish given the incorrect (more or less) amount of money?

## Solvability of politicians...

Q: What do you call a politician that when given the correct amount of money, is able to finish his/her tasks, but it is unknown whether his/her tasks will finish given the incorrect (more or less) amount of money?

A: Recursively enumerable but not recursive...

# Fin (wakas)

## Thanks for the attention.

## Questions?

## Work on examples and exercises in:

[Sipser 2005] Chapter 4.2 , or

[Hopcroft et al 2001] Chapter 9.1, 9.2

## References:


[Sipser 2005] M. Sipser. *Introduction to the Theory of Computation*: 2ed. PWS Publishing Company, 2005.

[Hopcroft, Ullman 1979] J. Hopcroft, J. Ullman. *Introduction to Automata Theory, Languages, and Computation*: Addison-Wesley, 1979.

[Hopcroft et al 2001] J. Hopcroft, R. Motwani, J. Ullman. *Introduction to Automata Theory, Languages, and Computation*: Addison-Wesley, 2001.