Computability and Complexity
COSC 4200

Undecidability

Undecidability

Some decision problems do not have an algorithmic solutions. Such problems are called *undecidable*. To prove that undecidable problems exist, we will use the technique of *diagonalization*.

First we review the original use of the technique to prove the uncountability of the real numbers.

Countable Sets

The natural numbers is the set $\mathbb{N} = \{0, 1, 2, \ldots\}$.
A function is a bijection if it is both one-to-one and onto.

Definition

A set X is *countable* if there is a bijection $f : \mathbb{N} \to X$.

Then $f(0)$, $f(1)$, $f(2)$,. . . , is a listing of the elements of X.

Countable Sets

The natural numbers is the set $\mathbb{N} = \{0, 1, 2, \ldots\}$.
A function is a bijection if it is both one-to-one and onto.

Definition

A set X is *countable* if there is a bijection $f : \mathbb{N} \to X$.

Then $f(0)$, $f(1)$, $f(2)$,..., is a listing of the elements of X.

Some countable sets:

- $\mathbb{N}$

Countable Sets

The natural numbers is the set $\mathbb{N} = \{0, 1, 2, \ldots\}$.
A function is a bijection if it is both one-to-one and onto.

Definition

A set X is *countable* if there is a bijection $f : \mathbb{N} \to X$.

Then $f(0)$, $f(1)$, $f(2)$,..., is a listing of the elements of X.

Some countable sets:

- $\mathbb{N}$
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$

Countable Sets

The natural numbers is the set $\mathbb{N} = \{0, 1, 2, \ldots\}$.
A function is a bijection if it is both one-to-one and onto.

Definition

A set X is *countable* if there is a bijection $f : \mathbb{N} \to X$.

Then $f(0)$, $f(1)$, $f(2)$,..., is a listing of the elements of X.

Some countable sets:

- $\mathbb{N}$
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\mathbb{Q} = \left\{ \frac{a}{b} \,|\, a, b \in \mathbb{Z} \text{ and } b \neq 0 \right\}$

Countable Sets

The natural numbers is the set $\mathbb{N} = \{0, 1, 2, \ldots\}$.
A function is a bijection if it is both one-to-one and onto.

Definition

A set X is *countable* if there is a bijection $f : \mathbb{N} \to X$.

Then $f(0)$, $f(1)$, $f(2)$,. . . , is a listing of the elements of X.

Some countable sets:

- $\mathbb{N}$
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\mathbb{Q} = \left\{ \frac{a}{b} \,|a, b \in \mathbb{Z} \text{ and } b \neq 0 \right\}$
- Σ^* for any finite alphabet Σ

Countable Sets

The natural numbers is the set $\mathbb{N} = \{0, 1, 2, \ldots\}$.
A function is a bijection if it is both one-to-one and onto.

Definition

A set X is *countable* if there is a bijection $f : \mathbb{N} \to X$.

Then $f(0)$, $f(1)$, $f(2)$,…, is a listing of the elements of X.

Some countable sets:

- $\mathbb{N}$
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\mathbb{Q} = \left\{ \frac{a}{b} \,|\, a, b \in \mathbb{Z} \text{ and } b \neq 0 \right\}$
- Σ^* for any finite alphabet Σ
- $L(M)$ for any Turing machine M

Countable Sets

The natural numbers is the set $\mathbb{N} = \{0, 1, 2, \ldots\}$.
A function is a bijection if it is both one-to-one and onto.

Definition

A set X is *countable* if there is a bijection $f : \mathbb{N} \to X$.

Then $f(0)$, $f(1)$, $f(2)$,..., is a listing of the elements of X.

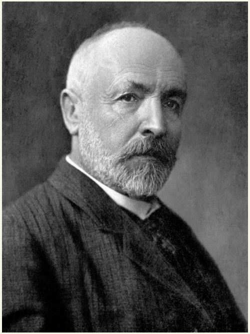
Some countable sets:

- $\mathbb{N}$
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\mathbb{Q} = \left\{ \frac{a}{b} \,|\, a, b \in \mathbb{Z} \text{ and } b \neq 0 \right\}$
- Σ^* for any finite alphabet Σ
- $L(M)$ for any Turing machine M
- $\{M \mid M \text{ is a Turing machine}\}$

The Real Numbers are Uncountable

Theorem (Cantor, 1874)

The set of real numbers $\mathbb{R}$ is uncountable.



Georg Cantor (1845-1918)

Theorem (Cantor, 1874)

The set of real numbers $\mathbb{R}$ is uncountable.

Proof.

Theorem (Cantor, 1874)

The set of real numbers $\mathbb{R}$ is uncountable.

Proof.

Let f by any function mapping $\mathbb{N} \to \mathbb{R}$. We will show that f is not onto.

Theorem (Cantor, 1874)

The set of real numbers $\mathbb{R}$ is uncountable.

Proof.

Let f by any function mapping $\mathbb{N} \to \mathbb{R}$. We will show that f is not onto. Consider the following table, writing the fractional part of each number in decimal:

$$f(0) \quad = \quad r_0.d_0^0 d_0^1 d_0^2 d_0^3 \ldots$$

Theorem (Cantor, 1874)

The set of real numbers $\mathbb{R}$ is uncountable.

Proof.

Let f by any function mapping $\mathbb{N} \to \mathbb{R}$. We will show that f is not onto. Consider the following table, writing the fractional part of each number in decimal:

$$\begin{array}{rcl} f(0) & = & r_0.d_0^0 d_0^1 d_0^2 d_0^3 \ldots \\ f(1) & = & r_1.d_1^0 d_1^1 d_1^2 d_1^3 \ldots \end{array}$$

Theorem (Cantor, 1874)

The set of real numbers $\mathbb{R}$ is uncountable.

Proof.

Let f by any function mapping $\mathbb{N} \to \mathbb{R}$. We will show that f is not onto. Consider the following table, writing the fractional part of each number in decimal:

$$\begin{array}{rcl} f(0) & = & r_0.d_0^0 d_0^1 d_0^2 d_0^3 \ldots \\ f(1) & = & r_1.d_1^0 d_1^1 d_1^2 d_1^3 \ldots \\ f(2) & = & r_2.d_2^0 d_2^1 d_2^2 d_2^3 \ldots \end{array}$$

Theorem (Cantor, 1874)

The set of real numbers $\mathbb{R}$ is uncountable.

Proof.

Let f by any function mapping $\mathbb{N} \to \mathbb{R}$. We will show that f is not onto. Consider the following table, writing the fractional part of each number in decimal:

$$\begin{array}{ccc} f(0) & = & r_0.d_0^0 d_0^1 d_0^2 d_0^3 \ldots \\ f(1) & = & r_1.d_1^0 d_1^1 d_1^2 d_1^3 \ldots \\ f(2) & = & r_2.d_2^0 d_2^1 d_2^2 d_2^3 \ldots \\ \vdots & & \vdots \end{array}$$

Theorem (Cantor, 1874)

The set of real numbers $\mathbb{R}$ is uncountable.

Proof.

Let f by any function mapping $\mathbb{N} \to \mathbb{R}$. We will show that f is not onto. Consider the following table, writing the fractional part of each number in decimal:

$$\begin{array}{ccc} f(0) & = & r_0.d_0^0 d_0^1 d_0^2 d_0^3 \ldots \\ f(1) & = & r_1.d_1^0 d_1^1 d_1^2 d_1^3 \ldots \\ f(2) & = & r_2.d_2^0 d_2^1 d_2^2 d_2^3 \ldots \\ \vdots & & \vdots \end{array}$$

We define a new number x by $x = 0.e_0 e_1 e_2 e_3 \ldots$, where

$$e_i = \begin{cases} 0 & \text{if } d_i^i \neq 0 \\ 1 & \text{if } d_i^i = 0 \end{cases}$$

Theorem (Cantor, 1874)

The set of real numbers $\mathbb{R}$ is uncountable.

Proof.

Let f by any function mapping $\mathbb{N} \to \mathbb{R}$. We will show that f is not onto. Consider the following table, writing the fractional part of each number in decimal:

$$\begin{array}{ccl} f(0) & = & r_0.d_0^0 d_0^1 d_0^2 d_0^3 \ldots \\ f(1) & = & r_1.d_1^0 d_1^1 d_1^2 d_1^3 \ldots \\ f(2) & = & r_2.d_2^0 d_2^1 d_2^2 d_2^3 \ldots \\ \vdots & & \qquad \vdots \end{array}$$

We define a new number x by $x = 0.e_0 e_1 e_2 e_3 \ldots$, where

$$e_i = \begin{cases} 0 & \text{if } d_i^i \neq 0 \\ 1 & \text{if } d_i^i = 0 \end{cases}$$

Then for all i, $e_i \neq d_i^i$. Therefore $x \neq f(i)$ for all i, so f is not onto. $\square$

Acceptance Problem for TMs

The *acceptance problem* for TMs:

$$A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts input string } w\ \}.$$

Acceptance Problem for TMs

The *acceptance problem* for TMs:

$$A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts input string } w\ \}.$$

Theorem

A_{TM} is Turing-recognizable.

Acceptance Problem for TMs

The *acceptance problem* for TMs:

$$A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts input string } w\ \}.$$

Theorem

A_{TM} is Turing-recognizable.

Proof. The following algorithm recognizes A_{TM}.

Algorithm U:
On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on w.

Acceptance Problem for TMs

The *acceptance problem* for TMs:

$$A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts input string } w\ \}.$$

Theorem

A_{TM} is Turing-recognizable.

Proof. The following algorithm recognizes A_{TM}.

Algorithm U:
On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on w.
2. If M ever enters an accept state, accept.
   If M ever enters a reject state, reject.

□

Acceptance Problem for TMs

The *acceptance problem* for TMs:

$$A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts input string } w\ \}.$$

Theorem

A_{TM} is Turing-recognizable.

Proof. The following algorithm recognizes A_{TM}.

Algorithm U:
On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on w.
2. If M ever enters an accept state, accept.
   If M ever enters a reject state, reject.

□

If M does not halt on w, then U will not halt. This is why U does not decide A_{TM}.

Theorem

A_{TM} *is not decidable.*

Theorem

A_{TM} is not decidable.

Proof. Assume that A_{TM} is decidable. Suppose that H is a decider for A_{TM}. Then

$$H(\langle M, w\rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w \\ \text{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

Theorem

A_{TM} is not decidable.

Proof. Assume that A_{TM} is decidable. Suppose that H is a decider for A_{TM}. Then

$$H(\langle M, w\rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w \\ \text{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

Algorithm D: On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. If H accepts, reject.
   If H rejects, accept.

For any M,

$$D(\langle M\rangle) = \begin{cases} \text{accept} & \text{if } M \text{ does not accept } \langle M\rangle \\ \text{reject} & \text{if } M \text{ accepts } \langle M\rangle. \end{cases}$$

For any M,

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ does not accept } \langle M \rangle \\ \text{reject} & \text{if } M \text{ accepts } \langle M \rangle. \end{cases}$$

What if we run D on its own code $\langle D \rangle$?

$$D(\langle D \rangle) = \begin{cases} \text{accept} & \text{if } D \text{ does not accept } \langle D \rangle \\ \text{reject} & \text{if } D \text{ accepts } \langle D \rangle. \end{cases}$$

For any M,

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ does not accept } \langle M \rangle \\ \text{reject} & \text{if } M \text{ accepts } \langle M \rangle. \end{cases}$$

What if we run D on its own code $\langle D \rangle$?

$$D(\langle D \rangle) = \begin{cases} \text{accept} & \text{if } D \text{ does not accept } \langle D \rangle \\ \text{reject} & \text{if } D \text{ accepts } \langle D \rangle. \end{cases}$$

Whatever D does, this says that D does the opposite, a contradiction. Therefore neither D or H can exist, so A_{TM} is undecidable. $\square$

To make the diagonalization argument more explicit, we define the *diagonal halting problem*

$$K = \{\langle M \rangle \mid M \text{ does not accept } \langle M \rangle\}.$$

Theorem

K is undecidable.

To make the diagonalization argument more explicit, we define the *diagonal halting problem*

$$K = \{\langle M\rangle \mid M \text{ does not accept } \langle M\rangle\}.$$

Theorem

K is undecidable.

Proof. Let M be any TM. We will show that M does not decide K.

To make the diagonalization argument more explicit, we define the *diagonal halting problem*

$$K = \{\langle M \rangle \mid M \text{ does not accept } \langle M \rangle\}.$$

Theorem

K is undecidable.

Proof. Let M be any TM. We will show that M does not decide K. We have two cases:

1. $\langle M \rangle \in K$. Then M does not accept $\langle M \rangle$ by definition of K.

To make the diagonalization argument more explicit, we define the *diagonal halting problem*

$$K = \{\langle M \rangle \mid M \text{ does not accept } \langle M \rangle\}.$$

Theorem

K is undecidable.

Proof. Let M be any TM. We will show that M does not decide K. We have two cases:

1. $\langle M \rangle \in K$. Then M does not accept $\langle M \rangle$ by definition of K.
2. $\langle M \rangle \notin K$. Then M accepts $\langle M \rangle$ by definition of K.

To make the diagonalization argument more explicit, we define the *diagonal halting problem*

$$K = \{\langle M \rangle \mid M \text{ does not accept } \langle M \rangle\}.$$

Theorem

K is undecidable.

Proof. Let M be any TM. We will show that M does not decide K. We have two cases:

1. $\langle M \rangle \in K$. Then M does not accept $\langle M \rangle$ by definition of K.
2. $\langle M \rangle \notin K$. Then M accepts $\langle M \rangle$ by definition of K.

In either case, M does not decide K correctly on input $\langle M \rangle$. Therefore M does not decide K. $\square$

Theorem

A_{TM} is undecidable.

Second proof. Suppose that A_{TM} is decidable by some TM N. We design a TM D as follows:

Theorem

A_{TM} is undecidable.

Second proof. Suppose that A_{TM} is decidable by some TM N. We design a TM D as follows:

D: On input $\langle M \rangle$, where M is a TM:

1. Run N on input $\langle M, \langle M \rangle \rangle$.
2. If N accepts, reject.
   If N rejects, accept.

Theorem

A_{TM} is undecidable.

Second proof. Suppose that A_{TM} is decidable by some TM N. We design a TM D as follows:

D: On input $\langle M \rangle$, where M is a TM:

1. Run N on input $\langle M, \langle M \rangle \rangle$.
2. If N accepts, reject.
   If N rejects, accept.

Then D decides K:

- If $\langle M \rangle \in K$, then $\langle M, \langle M \rangle \rangle \notin A_{\mathrm{TM}}$, so N rejects $\langle M, \langle M \rangle \rangle$ and D accepts $\langle M \rangle$.

Theorem

A_{TM} is undecidable.

Second proof. Suppose that A_{TM} is decidable by some TM N. We design a TM D as follows:

D: On input $\langle M \rangle$, where M is a TM:

1. Run N on input $\langle M, \langle M \rangle \rangle$.
2. If N accepts, reject.
   If N rejects, accept.

Then D decides K:

- If $\langle M \rangle \in K$, then $\langle M, \langle M \rangle \rangle \notin A_{\mathrm{TM}}$, so N rejects $\langle M, \langle M \rangle \rangle$ and D accepts $\langle M \rangle$.
- If $\langle M \rangle \notin K$, then $\langle M, \langle M \rangle \rangle \in A_{\mathrm{TM}}$, so N accepts $\langle M, \langle M \rangle \rangle$ and D rejects $\langle M \rangle$.

Theorem

A_{TM} is undecidable.

Second proof. Suppose that A_{TM} is decidable by some TM N. We design a TM D as follows:

D: On input $\langle M\rangle$, where M is a TM:

1. Run N on input $\langle M, \langle M\rangle\rangle$.
2. If N accepts, reject.
   If N rejects, accept.

Then D decides K:

- If $\langle M\rangle \in K$, then $\langle M, \langle M\rangle\rangle \notin A_{\mathrm{TM}}$, so N rejects $\langle M, \langle M\rangle\rangle$ and D accepts $\langle M\rangle$.
- If $\langle M\rangle \notin K$, then $\langle M, \langle M\rangle\rangle \in A_{\mathrm{TM}}$, so N accepts $\langle M, \langle M\rangle\rangle$ and D rejects $\langle M\rangle$.

But K is undecidable, a contradiction. Therefore A_{TM} is undecidable. $\square$

Co-Turing-Recognizability

Definition

A language is *co-Turing-recognizable* if it is the complement of a Turing-recognizable language.

Co-Turing-Recognizability

Definition

A language is *co-Turing-recognizable* if it is the complement of a Turing-recognizable language.

If A is co-Turing-recognizable, then A^c is Turing-recognizable. This means there is a Turing machine M such that for all inputs w,

- $w \in A^c \Rightarrow M$ accepts w.
- $w \notin A^c \Rightarrow M$ does not accept w.

Co-Turing-Recognizability

Definition

A language is *co-Turing-recognizable* if it is the complement of a Turing-recognizable language.

If A is co-Turing-recognizable, then A^c is Turing-recognizable. This means there is a Turing machine M such that for all inputs w,

- $w \in A^c \Rightarrow M$ accepts w.
- $w \notin A^c \Rightarrow M$ does not accept w.

Equivalently,

- $w \notin A \Rightarrow M$ accepts w.
- $w \in A \Rightarrow M$ does not accept w.

A is *decidable* if there is a TM M such that for all w,

- $w \in A \Rightarrow M$ accepts w.
- $w \notin A \Rightarrow M$ rejects w.

A is *Turing-recognizable* if there is a TM M such that for all w,

- $w \in A \Rightarrow M$ accepts w.
- $w \notin A \Rightarrow M$ does not accept w.

A is *co-Turing-recognizable* if there is a TM M such that for all w,

- $w \in A \Rightarrow M$ does not accept w.
- $w \notin A \Rightarrow M$ accepts w.

A is *decidable* if there is a TM M such that for all w,

- $w \in A \Rightarrow M$ accepts w.
- $w \notin A \Rightarrow M$ rejects w.

A is *Turing-recognizable* if there is a TM M such that for all w,

- $w \in A \Rightarrow M$ accepts w.
- $w \notin A \Rightarrow M$ does not accept w.

A is *co-Turing-recognizable* if there is a TM M such that for all w,

- $w \in A \Rightarrow M$ does not accept w.
- $w \notin A \Rightarrow M$ accepts w.

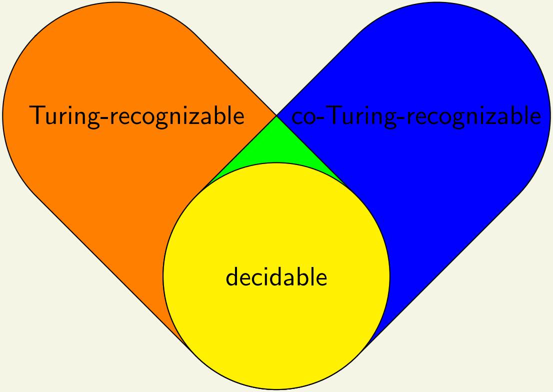
Theorem

A language is decidable if and only if both it is both Turing-recognizable and co-Turing-recognizable.

Decidable = Turing-recognizable $\cap$ co-Turing-recognizable

Theorem

A language is decidable if and only if both it is both Turing-recognizable and co-Turing-recognizable.

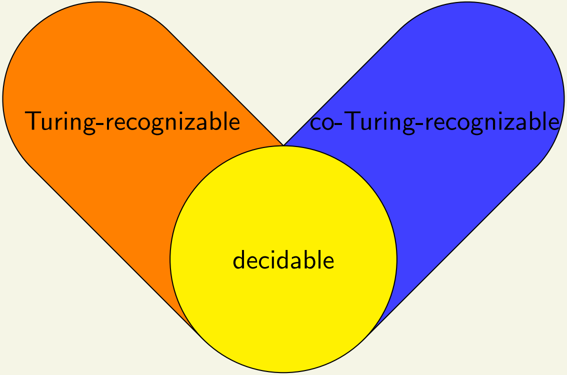


The theorem says the green region is empty.

Decidable = Turing-recognizable ∩ co-Turing-recognizable

Theorem

A language is decidable if and only if both it is both Turing-recognizable and co-Turing-recognizable.



Decidable = Turing-recognizable $\cap$ co-Turing-recognizable

Theorem

A language A is decidable if and only if A is both Turing-recognizable and co-Turing-recognizable.

Decidable = Turing-recognizable $\cap$ co-Turing-recognizable

Theorem

A language A is decidable if and only if A is both Turing-recognizable and co-Turing-recognizable.

Proof. Let A be a language.

$(\Rightarrow)$ If A is decidable, then A^c is also decidable.

Since every decidable language is Turing-recognizable, both A and A^c are Turing-recognizable.

($\Leftarrow$) Suppose that A and A^c are Turing-recognizable. Let M_1 be a recognizer for A and let M_2 be a recognizer for A^c.

Then

- $w \in A \Rightarrow M_1$ accepts w.
- $w \notin A \Rightarrow M_1$ does not accept w.

and

- $w \in A \Rightarrow M_2$ does not accept w.
- $w \notin A \Rightarrow M_2$ accepts w.

When M_1 or M_2 does not accept, they may run forever.

Let M be the following TM:

M: On input w:

1. Run both M_1 and M_2 on w in parallel.
2. If M_1 accepts, accept.
   If M_2 accepts, reject.

Let M be the following TM:

M: On input w:

1. Run both M_1 and M_2 on w in parallel.
2. If M_1 accepts, accept.
   If M_2 accepts, reject.

Running in "parallel" means using two tapes (which can be simulated by a one-tape TM).

- Copy the input from the first tape to the second tape. Move both tape heads to the beginning.
- Run M_1 on the first tape and M_2 on the second tape simultaneously.
- Accept if M_1 accepts. Reject if M_2 accepts.

We have

$$\begin{aligned} w \in A &\Rightarrow M_1 \text{ accepts } w \\ &\Rightarrow M \text{ accepts } w \end{aligned}$$

and

$$\begin{aligned} w \notin A &\Rightarrow M_2 \text{ accepts } w \\ &\Rightarrow M \text{ rejects } w. \end{aligned}$$

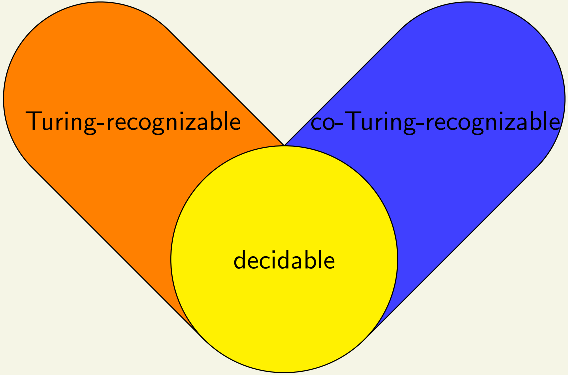
We have

$$\begin{aligned} w \in A &\Rightarrow M_1 \text{ accepts } w \\ &\Rightarrow M \text{ accepts } w \end{aligned}$$

and

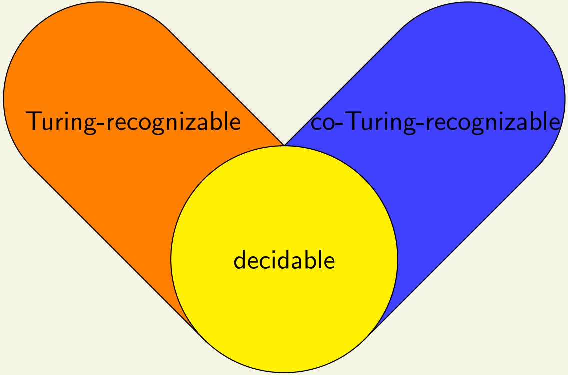
$$\begin{aligned} w \notin A &\Rightarrow M_2 \text{ accepts } w \\ &\Rightarrow M \text{ rejects } w. \end{aligned}$$

Then M decides A, so A is decidable. $\square$



Theorem

A language A is decidable if and only if A is Turing-recognizable and co-Turing-recognizable.



Theorem

A language A is decidable if and only if A is Turing-recognizable and co-Turing-recognizable.

Corollary

1. *If A is Turing-recognizable but not decidable, then A is not co-Turing-recognizable.*
2. *If A is co-Turing-recognizable but not decidable, then A is not Turing-recognizable.*

Corollary

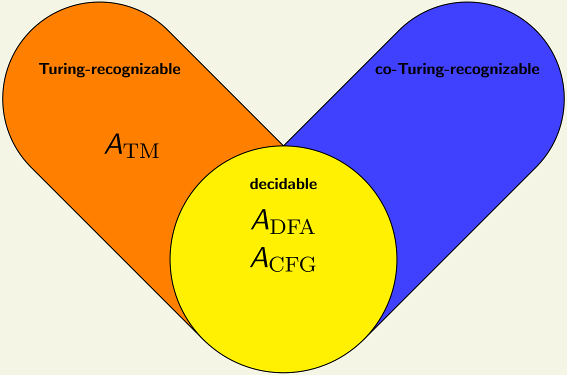
A_{TM} is not co-Turing-recognizable.

Corollary

A_{TM} is not co-Turing-recognizable.

Proof.

We know that A_{TM} is Turing-recognizable. If A_{TM} is also co-Turing-recognizable, then A_{TM} is decidable, but it is not. $\square$


Turing-recognizable
co-Turing-recognizable
A_{TM}
decidable
A_{DFA}
A_{CFG}

$$K = \{\langle M \rangle \mid M \text{ does not accept } \langle M \rangle\}$$

Corollary

K is not Turing-recognizable.

$$K = \{\langle M \rangle \mid M \text{ does not accept } \langle M \rangle\}$$

Corollary

K is not Turing-recognizable.

Proof. This is because K is co-Turing-recognizable but not decidable.

$$K = \{\langle M \rangle \mid M \text{ does not accept } \langle M \rangle\}$$

Corollary

K is not Turing-recognizable.

Proof. This is because K is co-Turing-recognizable but not decidable.

To see that K is co-Turing-recognizable, consider the following TM R:

R: On input $\langle M \rangle$:

1. Run M on input $\langle M \rangle$.
2. If M accepts, accept.
   If M rejects, reject.

$$K = \{\langle M \rangle \mid M \text{ does not accept } \langle M \rangle\}$$

Corollary

K is not Turing-recognizable.

Proof. This is because K is co-Turing-recognizable but not decidable.

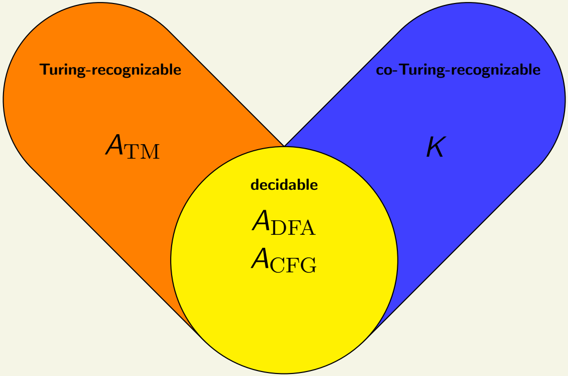
To see that K is co-Turing-recognizable, consider the following TM R:

R: On input $\langle M \rangle$:

1. Run M on input $\langle M \rangle$.
2. If M accepts, accept.
   If M rejects, reject.

Then R recognizes K^c:

- If $\langle M \rangle \in K$, then M does not accept $\langle M \rangle$, so R will not accept $\langle M \rangle$.
- If $\langle M \rangle \notin K$, then M accepts $\langle M \rangle$, so R will accept $\langle M \rangle$. $\square$


Turing-recognizable
co-Turing-recognizable
A_{TM}
K
decidable
A_{DFA}
A_{CFG}

Halting Problem for TMs

The *halting problem* for TMs:

$$HALT_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that halts on input string } w\ \}.$$

Halting Problem for TMs

The *halting problem* for TMs:

$$HALT_{\mathrm{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM that halts on input string } w\ \}.$$

Theorem

$HALT_{\mathrm{TM}}$ is Turing-recognizable.

Halting Problem for TMs

The *halting problem* for TMs:

$$HALT_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that halts on input string } w\ \}.$$

Theorem

$HALT_{\mathrm{TM}}$ is Turing-recognizable.

Proof. The following algorithm recognizes $HALT_{\mathrm{TM}}$.

Algorithm U:
On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on w.

Halting Problem for TMs

The *halting problem* for TMs:

$$HALT_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that halts on input string } w\ \}.$$

Theorem

$HALT_{\mathrm{TM}}$ is Turing-recognizable.

Proof. The following algorithm recognizes $HALT_{\mathrm{TM}}$.

Algorithm U:
On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on w.
2. If M halts, accept.

$\square$

Halting Problem for TMs

The *halting problem* for TMs:

$$HALT_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that halts on input string } w \}.$$

Theorem

$HALT_{\mathrm{TM}}$ is Turing-recognizable.

Proof. The following algorithm recognizes $HALT_{\mathrm{TM}}$.

Algorithm U:
On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on w.
2. If M halts, accept.

□

If M does not halt on w, then U will not halt. This is why U does not decide $HALT_{\mathrm{TM}}$.

Theorem

$HALT_{\mathrm{TM}}$ is undecidable.

Proof. Suppose that $HALT_{\mathrm{TM}}$ is decidable. We will show that A_{TM} is also decidable, obtaining a contradiction.

Theorem

$HALT_{\mathrm{TM}}$ is undecidable.

Proof. Suppose that $HALT_{\mathrm{TM}}$ is decidable. We will show that A_{TM} is also decidable, obtaining a contradiction.

Let R be a decider for $HALT_{\mathrm{TM}}$. We construct the following TM S to decide A_{TM}.

Theorem

$HALT_{\mathrm{TM}}$ is undecidable.

Proof. Suppose that $HALT_{\mathrm{TM}}$ is decidable. We will show that A_{TM} is also decidable, obtaining a contradiction.

Let R be a decider for $HALT_{\mathrm{TM}}$. We construct the following TM S to decide A_{TM}.

S: On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Run TM R on input $\langle M, w\rangle$.
2. If R rejects, reject.

Theorem

$HALT_{\mathrm{TM}}$ is undecidable.

Proof. Suppose that $HALT_{\mathrm{TM}}$ is decidable. We will show that A_{TM} is also decidable, obtaining a contradiction.

Let R be a decider for $HALT_{\mathrm{TM}}$. We construct the following TM S to decide A_{TM}.

S: On input $\langle M, w \rangle$, where M is a TM and w is a string:

1. Run TM R on input $\langle M, w \rangle$.
2. If R rejects, reject.
3. If R accepts, simulate M on w until it halts.
   - If M accepts, accept.
   - If M rejects, reject.

We now verify that S decides A_{TM}. First suppose $\langle M, w \rangle \in A_{\mathrm{TM}}$:

$$\langle M, w \rangle \in A_{\mathrm{TM}} \quad \Rightarrow \quad M \text{ accepts } w$$

We now verify that S decides A_{TM}. First suppose $\langle M, w\rangle \in A_{\mathrm{TM}}$:

$$\begin{aligned} \langle M, w\rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\ &\Rightarrow M \text{ halts on } w \end{aligned}$$

We now verify that S decides A_{TM}. First suppose $\langle M, w \rangle \in A_{\mathrm{TM}}$:

$$\begin{aligned}
\langle M, w \rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\
&\Rightarrow M \text{ halts on } w \\
&\Rightarrow R \text{ accepts } \langle M, w \rangle
\end{aligned}$$

We now verify that S decides A_{TM}. First suppose $\langle M, w \rangle \in A_{\mathrm{TM}}$:

$$\begin{aligned} \langle M, w \rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\ &\Rightarrow M \text{ halts on } w \\ &\Rightarrow R \text{ accepts } \langle M, w \rangle \\ &\Rightarrow S \text{ simulates } M \text{ on } w \end{aligned}$$

We now verify that S decides A_{TM}. First suppose $\langle M, w\rangle \in A_{\mathrm{TM}}$:

$$
\begin{aligned}
\langle M, w\rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\
&\Rightarrow M \text{ halts on } w \\
&\Rightarrow R \text{ accepts } \langle M, w\rangle \\
&\Rightarrow S \text{ simulates } M \text{ on } w \\
&\Rightarrow S \text{ finds that } M \text{ accepts } w
\end{aligned}
$$

We now verify that S decides A_{TM}. First suppose $\langle M, w \rangle \in A_{\mathrm{TM}}$:

$$\begin{aligned}
\langle M, w \rangle \in A_{\mathrm{TM}} \quad &\Rightarrow \quad M \text{ accepts } w \\
&\Rightarrow \quad M \text{ halts on } w \\
&\Rightarrow \quad R \text{ accepts } \langle M, w \rangle \\
&\Rightarrow \quad S \text{ simulates } M \text{ on } w \\
&\Rightarrow \quad S \text{ finds that } M \text{ accepts } w \\
&\Rightarrow \quad S \text{ accepts } \langle M, w \rangle
\end{aligned}$$

Now suppose $\langle M, w \rangle \notin A_{\mathrm{TM}}$. Then M does not accept w, so either M rejects w or M does not halt on w. We consider these two cases separately:

$$M \text{ rejects } w \quad \Rightarrow \quad M \text{ halts on } w$$

Now suppose $\langle M, w\rangle \notin A_{\mathrm{TM}}$. Then M does not accept w, so either M rejects w or M does not halt on w. We consider these two cases separately:

$$\begin{aligned} M \text{ rejects } w \quad &\Rightarrow \quad M \text{ halts on } w \\ &\Rightarrow \quad R \text{ accepts } \langle M, w\rangle \end{aligned}$$

Now suppose $\langle M, w\rangle \notin A_{\mathrm{TM}}$. Then M does not accept w, so either M rejects w or M does not halt on w. We consider these two cases separately:

$$\begin{aligned} M \text{ rejects } w \quad &\Rightarrow \quad M \text{ halts on } w \\ &\Rightarrow \quad R \text{ accepts } \langle M, w\rangle \\ &\Rightarrow \quad S \text{ simulates } M \text{ on } w \end{aligned}$$

Now suppose $\langle M, w \rangle \notin A_{\mathrm{TM}}$. Then M does not accept w, so either M rejects w or M does not halt on w. We consider these two cases separately:

$$
\begin{aligned}
M \text{ rejects } w \quad &\Rightarrow \quad M \text{ halts on } w \\
&\Rightarrow \quad R \text{ accepts } \langle M, w \rangle \\
&\Rightarrow \quad S \text{ simulates } M \text{ on } w \\
&\Rightarrow \quad S \text{ finds that } M \text{ rejects } w
\end{aligned}
$$

Now suppose $\langle M, w\rangle \notin A_{\mathrm{TM}}$. Then M does not accept w, so either M rejects w or M does not halt on w. We consider these two cases separately:

$$\begin{aligned}
M \text{ rejects } w \quad &\Rightarrow \quad M \text{ halts on } w \\
&\Rightarrow \quad R \text{ accepts } \langle M, w\rangle \\
&\Rightarrow \quad S \text{ simulates } M \text{ on } w \\
&\Rightarrow \quad S \text{ finds that } M \text{ rejects } w \\
&\Rightarrow \quad S \text{ rejects } \langle M, w\rangle
\end{aligned}$$

Now suppose $\langle M, w \rangle \notin A_{\mathrm{TM}}$. Then M does not accept w, so either M rejects w or M does not halt on w. We consider these two cases separately:

$$\begin{aligned} M \text{ rejects } w &\Rightarrow M \text{ halts on } w \\ &\Rightarrow R \text{ accepts } \langle M, w \rangle \\ &\Rightarrow S \text{ simulates } M \text{ on } w \\ &\Rightarrow S \text{ finds that } M \text{ rejects } w \\ &\Rightarrow S \text{ rejects } \langle M, w \rangle \end{aligned}$$

$$M \text{ does not halt on } w \quad \Rightarrow \quad R \text{ rejects } \langle M, w \rangle$$

Now suppose $\langle M, w\rangle \notin A_{\mathrm{TM}}$. Then M does not accept w, so either M rejects w or M does not halt on w. We consider these two cases separately:

$$\begin{aligned}
M \text{ rejects } w &\Rightarrow M \text{ halts on } w \\
&\Rightarrow R \text{ accepts } \langle M, w\rangle \\
&\Rightarrow S \text{ simulates } M \text{ on } w \\
&\Rightarrow S \text{ finds that } M \text{ rejects } w \\
&\Rightarrow S \text{ rejects } \langle M, w\rangle
\end{aligned}$$

$$\begin{aligned}
M \text{ does not halt on } w &\Rightarrow R \text{ rejects } \langle M, w\rangle \\
&\Rightarrow S \text{ rejects } \langle M, w\rangle
\end{aligned}$$

Now suppose $\langle M, w \rangle \notin A_{\mathrm{TM}}$. Then M does not accept w, so either M rejects w or M does not halt on w. We consider these two cases separately:

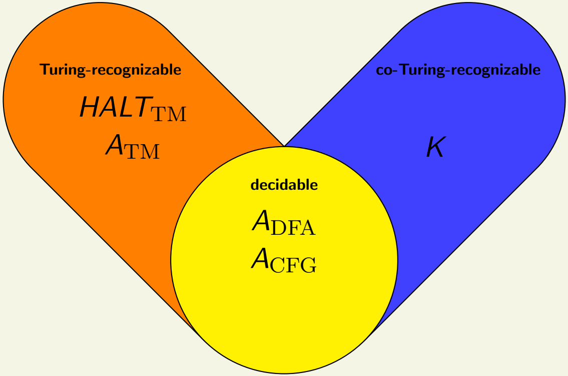
$$
\begin{aligned}
M \text{ rejects } w &\Rightarrow M \text{ halts on } w \\
&\Rightarrow R \text{ accepts } \langle M, w \rangle \\
&\Rightarrow S \text{ simulates } M \text{ on } w \\
&\Rightarrow S \text{ finds that } M \text{ rejects } w \\
&\Rightarrow S \text{ rejects } \langle M, w \rangle
\end{aligned}
$$

$$
\begin{aligned}
M \text{ does not halt on } w &\Rightarrow R \text{ rejects } \langle M, w \rangle \\
&\Rightarrow S \text{ rejects } \langle M, w \rangle
\end{aligned}
$$

In either case, S rejects $\langle M, w \rangle$.

Since A_{TM} is undecidable, decider S does not exist.

Therefore decider R for $HALT_{\mathrm{TM}}$ does not exist and $HALT_{\mathrm{TM}}$ must be undecidable. $\square$


Turing-recognizable
$HALT_{\mathrm{TM}}$
A_{TM}
co-Turing-recognizable
K
decidable
A_{DFA}
A_{CFG}

Emptiness Problem for TMs

The *emptiness problem* for TMs:

$$E_{\mathrm{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \}.$$

Emptiness Problem for TMs

The *emptiness problem* for TMs:

$$E_{\mathrm{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\ \}.$$

Theorem

E_{TM} is undecidable.

Emptiness Problem for TMs

The *emptiness problem* for TMs:

$$E_{\mathrm{TM}} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \}.$$

Theorem

E_{TM} is undecidable.

Proof. Suppose that E_{TM} is decidable. We will show that A_{TM} is also decidable.

Emptiness Problem for TMs

The *emptiness problem* for TMs:

$$E_{\mathrm{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \}.$$

Theorem

E_{TM} is undecidable.

Proof. Suppose that E_{TM} is decidable. We will show that A_{TM} is also decidable.

The idea of the proof is that given any instance $\langle M, w \rangle$ of A_{TM}, we can construct an instance $\langle M_{(w)} \rangle$ of E_{TM} so that:

Emptiness Problem for TMs

The *emptiness problem* for TMs:

$$E_{\mathrm{TM}} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \ \}.$$

Theorem

E_{TM} is undecidable.

Proof. Suppose that E_{TM} is decidable. We will show that A_{TM} is also decidable.

The idea of the proof is that given any instance $\langle M, w \rangle$ of A_{TM}, we can construct an instance $\langle M_{(w)} \rangle$ of E_{TM} so that:

- If M accepts w, then $L(M_{(w)}) \neq \emptyset$.

Emptiness Problem for TMs

The *emptiness problem* for TMs:

$$E_{\mathrm{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\ \}.$$

Theorem

E_{TM} is undecidable.

Proof. Suppose that E_{TM} is decidable. We will show that A_{TM} is also decidable.

The idea of the proof is that given any instance $\langle M, w \rangle$ of A_{TM}, we can construct an instance $\langle M_{(w)} \rangle$ of E_{TM} so that:

- If M accepts w, then $L(M_{(w)}) \neq \emptyset$.
- If M does not accept w, then $L(M_{(w)}) = \emptyset$.

Let M be a TM and w be an input for M. Here is the description of $M_{(w)}$. Note that w is hardcoded into $M_{(w)}$.

$M_{(w)}$: on any input x:

1. If $x \neq w$, reject.
2. If $x = w$, then run M on input w and accept if M does.

Let M be a TM and w be an input for M. Here is the description of $M_{(w)}$. Note that w is hardcoded into $M_{(w)}$.

$M_{(w)}$: on any input x:

1. If $x \neq w$, reject.
2. If $x = w$, then run M on input w and accept if M does.

Then if M accepts w, $L(M_{(w)}) = \{w\}$.

If M does not accept w, $L(M_{(w)}) = \emptyset$.

Now, assuming that E_{TM} is decidable, we can use an algorithm for E_{TM} to solve A_{TM}. Suppose R is a TM that decides E_{TM}.

Now, assuming that E_{TM} is decidable, we can use an algorithm for E_{TM} to solve A_{TM}. Suppose R is a TM that decides E_{TM}.

S: on input $\langle M, w \rangle$, where M is a TM and w is a string:

1. Use the description of M and w to construct the TM $M_{(w)}$ as described above.

Now, assuming that E_{TM} is decidable, we can use an algorithm for E_{TM} to solve A_{TM}. Suppose R is a TM that decides E_{TM}.

S: on input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Use the description of M and w to construct the TM $M_{(w)}$ as described above.
2. Run R on input $\langle M_{(w)}\rangle$.

Now, assuming that E_{TM} is decidable, we can use an algorithm for E_{TM} to solve A_{TM}. Suppose R is a TM that decides E_{TM}.

S: on input $\langle M, w \rangle$, where M is a TM and w is a string:

1. Use the description of M and w to construct the TM $M_{(w)}$ as described above.
2. Run R on input $\langle M_{(w)} \rangle$.
    - If R accepts, reject.
    - If R rejects, accept.

We verify that S decides A_{TM}.

$$\langle M, w\rangle \in A_{\mathrm{TM}} \quad \Rightarrow \quad M \text{ accepts } w$$

We verify that S decides A_{TM}.

$$\begin{aligned} \langle M, w\rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\ &\Rightarrow L(M_{(w)}) = \{w\} \end{aligned}$$

We verify that S decides A_{TM}.

$$\begin{aligned} \langle M, w \rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\ &\Rightarrow L(M_{(w)}) = \{w\} \\ &\Rightarrow L(M_{(w)}) \neq \emptyset \end{aligned}$$

We verify that S decides A_{TM}.

$$\begin{aligned} \langle M, w \rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\ &\Rightarrow L(M_{(w)}) = \{w\} \\ &\Rightarrow L(M_{(w)}) \neq \emptyset \\ &\Rightarrow \langle M_{(w)} \rangle \notin E_{\mathrm{TM}} \end{aligned}$$

We verify that S decides A_{TM}.

$$\begin{aligned}
\langle M, w\rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\
&\Rightarrow L(M_{(w)}) = \{w\} \\
&\Rightarrow L(M_{(w)}) \neq \emptyset \\
&\Rightarrow \langle M_{(w)}\rangle \notin E_{\mathrm{TM}} \\
&\Rightarrow R \text{ rejects } \langle M_{(w)}\rangle
\end{aligned}$$

We verify that S decides A_{TM}.

$$\begin{aligned}
\langle M, w\rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\
&\Rightarrow L(M_{(w)}) = \{w\} \\
&\Rightarrow L(M_{(w)}) \neq \emptyset \\
&\Rightarrow \langle M_{(w)}\rangle \notin E_{\mathrm{TM}} \\
&\Rightarrow R \text{ rejects } \langle M_{(w)}\rangle \\
&\Rightarrow S \text{ accepts } \langle M, w\rangle
\end{aligned}$$

We verify that S decides A_{TM}.

$$\begin{aligned}
\langle M, w\rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\
&\Rightarrow L(M_{(w)}) = \{w\} \\
&\Rightarrow L(M_{(w)}) \neq \emptyset \\
&\Rightarrow \langle M_{(w)}\rangle \notin E_{\mathrm{TM}} \\
&\Rightarrow R \text{ rejects } \langle M_{(w)}\rangle \\
&\Rightarrow S \text{ accepts } \langle M, w\rangle
\end{aligned}$$

$$\langle M, w\rangle \notin A_{\mathrm{TM}} \Rightarrow M \text{ does not accept } w$$

We verify that S decides A_{TM}.

$$\begin{aligned}
\langle M, w\rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\
&\Rightarrow L(M_{(w)}) = \{w\} \\
&\Rightarrow L(M_{(w)}) \neq \emptyset \\
&\Rightarrow \langle M_{(w)}\rangle \notin E_{\mathrm{TM}} \\
&\Rightarrow R \text{ rejects } \langle M_{(w)}\rangle \\
&\Rightarrow S \text{ accepts } \langle M, w\rangle
\end{aligned}$$

$$\begin{aligned}
\langle M, w\rangle \notin A_{\mathrm{TM}} &\Rightarrow M \text{ does not accept } w \\
&\Rightarrow L(M_{(w)}) = \emptyset
\end{aligned}$$

We verify that S decides A_{TM}.

$$\begin{aligned}
\langle M, w\rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\
&\Rightarrow L(M_{(w)}) = \{w\} \\
&\Rightarrow L(M_{(w)}) \neq \emptyset \\
&\Rightarrow \langle M_{(w)}\rangle \notin E_{\mathrm{TM}} \\
&\Rightarrow R \text{ rejects } \langle M_{(w)}\rangle \\
&\Rightarrow S \text{ accepts } \langle M, w\rangle
\end{aligned}$$

$$\begin{aligned}
\langle M, w\rangle \notin A_{\mathrm{TM}} &\Rightarrow M \text{ does not accept } w \\
&\Rightarrow L(M_{(w)}) = \emptyset \\
&\Rightarrow \langle M_{(w)}\rangle \in E_{\mathrm{TM}}
\end{aligned}$$

We verify that S decides A_{TM}.

$$\begin{aligned}
\langle M, w\rangle \in A_{\mathrm{TM}} \quad &\Rightarrow \quad M \text{ accepts } w \\
&\Rightarrow \quad L(M_{(w)}) = \{w\} \\
&\Rightarrow \quad L(M_{(w)}) \neq \emptyset \\
&\Rightarrow \quad \langle M_{(w)}\rangle \notin E_{\mathrm{TM}} \\
&\Rightarrow \quad R \text{ rejects } \langle M_{(w)}\rangle \\
&\Rightarrow \quad S \text{ accepts } \langle M, w\rangle
\end{aligned}$$

$$\begin{aligned}
\langle M, w\rangle \notin A_{\mathrm{TM}} \quad &\Rightarrow \quad M \text{ does not accept } w \\
&\Rightarrow \quad L(M_{(w)}) = \emptyset \\
&\Rightarrow \quad \langle M_{(w)}\rangle \in E_{\mathrm{TM}} \\
&\Rightarrow \quad R \text{ accepts } \langle M_{(w)}\rangle
\end{aligned}$$

We verify that S decides A_{TM}.

$$\begin{aligned}
\langle M, w\rangle \in A_{\mathrm{TM}} \quad & \Rightarrow \quad M \text{ accepts } w \\
& \Rightarrow \quad L(M_{(w)}) = \{w\} \\
& \Rightarrow \quad L(M_{(w)}) \neq \emptyset \\
& \Rightarrow \quad \langle M_{(w)}\rangle \notin E_{\mathrm{TM}} \\
& \Rightarrow \quad R \text{ rejects } \langle M_{(w)}\rangle \\
& \Rightarrow \quad S \text{ accepts } \langle M, w\rangle
\end{aligned}$$

$$\begin{aligned}
\langle M, w\rangle \notin A_{\mathrm{TM}} \quad & \Rightarrow \quad M \text{ does not accept } w \\
& \Rightarrow \quad L(M_{(w)}) = \emptyset \\
& \Rightarrow \quad \langle M_{(w)}\rangle \in E_{\mathrm{TM}} \\
& \Rightarrow \quad R \text{ accepts } \langle M_{(w)}\rangle \\
& \Rightarrow \quad S \text{ rejects } \langle M, w\rangle
\end{aligned}$$

We verify that S decides A_{TM}.

$$\begin{aligned}
\langle M, w\rangle \in A_{\mathrm{TM}} &\Rightarrow M \text{ accepts } w \\
&\Rightarrow L(M_{(w)}) = \{w\} \\
&\Rightarrow L(M_{(w)}) \neq \emptyset \\
&\Rightarrow \langle M_{(w)}\rangle \notin E_{\mathrm{TM}} \\
&\Rightarrow R \text{ rejects } \langle M_{(w)}\rangle \\
&\Rightarrow S \text{ accepts } \langle M, w\rangle
\end{aligned}$$

$$\begin{aligned}
\langle M, w\rangle \notin A_{\mathrm{TM}} &\Rightarrow M \text{ does not accept } w \\
&\Rightarrow L(M_{(w)}) = \emptyset \\
&\Rightarrow \langle M_{(w)}\rangle \in E_{\mathrm{TM}} \\
&\Rightarrow R \text{ accepts } \langle M_{(w)}\rangle \\
&\Rightarrow S \text{ rejects } \langle M, w\rangle
\end{aligned}$$

Therefore S decides A_{TM}, a contradiction, so E_{TM} is undecidable. $\square$

We just proved E_{TM} is undecidable. Is it Turing-recognizable?

Theorem

E_{TM} is co-Turing-recognizable.

We just proved E_{TM} is undecidable. Is it Turing-recognizable?

Theorem

E_{TM} is co-Turing-recognizable.

Proof. Let $s_1, s_2, \ldots$ be an enumeration of all strings in Σ^*.

Algorithm A: On input $\langle M \rangle$:

for $i = 1, 2, \ldots$
  for $j = 1$ to i
    Run M on input s_j for i steps.
    If M accepts, accept.

We just proved E_{TM} is undecidable. Is it Turing-recognizable?

Theorem

E_{TM} is co-Turing-recognizable.

Proof. Let $s_1, s_2, \ldots$ be an enumeration of all strings in Σ^*.

Algorithm A: On input $\langle M \rangle$:

for $i = 1, 2, \ldots$
    for $j = 1$ to i
        Run M on input s_j for i steps.
        If M accepts, accept.

- If $L(M) \neq \emptyset$, then some string is accepted by M, so A will accept $\langle M \rangle$.

We just proved E_{TM} is undecidable. Is it Turing-recognizable?

Theorem

E_{TM} is co-Turing-recognizable.

Proof. Let $s_1, s_2, \ldots$ be an enumeration of all strings in Σ^*.

Algorithm A: On input $\langle M \rangle$:

for $i = 1, 2, \ldots$
    for $j = 1$ to i
        Run M on input s_j for i steps.
        If M accepts, accept.

- If $L(M) \neq \emptyset$, then some string is accepted by M, so A will accept $\langle M \rangle$.
- If $L(M) = \emptyset$, then no string is accepted by M, and A will run forever on $\langle M \rangle$.

We just proved E_{TM} is undecidable. Is it Turing-recognizable?

Theorem

E_{TM} is co-Turing-recognizable.

Proof. Let $s_1, s_2, \ldots$ be an enumeration of all strings in Σ^*.

Algorithm A: On input $\langle M \rangle$:

for $i = 1, 2, \ldots$
    for $j = 1$ to i
        Run M on input s_j for i steps.
        If M accepts, accept.

- If $L(M) \neq \emptyset$, then some string is accepted by M, so A will accept $\langle M \rangle$.
- If $L(M) = \emptyset$, then no string is accepted by M, and A will run forever on $\langle M \rangle$.

Therefore E^c_{TM} is Turing-recognizable. $\square$

Theorem

E_{TM} is undecidable.

Theorem

E_{TM} is co-Turing-recognizable.

Theorem

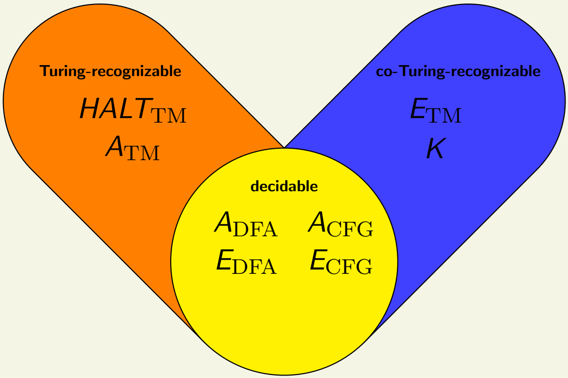
E_{TM} is undecidable.

Theorem

E_{TM} is co-Turing-recognizable.

Corollary

E_{TM} is not Turing-recognizable.


Turing-recognizable
$HALT_{\mathrm{TM}}$
A_{TM}
co-Turing-recognizable
E_{TM}
K
decidable
A_{DFA}
A_{CFG}
E_{DFA}
E_{CFG}