CSE 135: Introduction to Theory of Computation
Rice's Theorem and Closure Properties

Sungjin Im

University of California, Merced

04-21-2015

Mapping Reductions

Definition

A function $f : \Sigma^* \to \Sigma^*$ is computable if there is some Turing Machine M that on every input w halts with $f(w)$ on the tape.

Definition

A reduction (a.k.a. mapping reduction/many-one reduction) from a language A to a language B is a computable function $f : \Sigma^* \to \Sigma^*$ such that

$$w \in A \text{ if and only if } f(w) \in B$$

In this case, we say A is reducible to B, and we denote it by $A \leq_m B$.

Reductions and Recursive Enumerability

Proposition

If $A \leq_m B$ and B is r.e., then A is r.e.

Proof.

Let f be a reduction from A to B and let M_B be a Turing Machine recognizing B. Then the Turing machine recognizing A is

```
On input w
    Compute f(w)
    Run M_B on f(w)
    Accept if M_B accepts, and reject if M_B rejects
```

$\square$

Corollary

If $A \leq_m B$ and A is not r.e., then B is not r.e.

Reductions and Decidability

Proposition

If $A \leq_m B$ and B is decidable, then A is decidable.

Proof.

Let f be a reduction from A to B and let M_B be a Turing Machine *deciding* B. Then a Turing machine that decides A is

On input w
    Compute $f(w)$
    Run M_B on $f(w)$
    Accept if M_B accepts, and reject if M_B rejects $\square$

Corollary

If $A \leq_m B$ and A is undecidable, then B is undecidable.

The Halting Problem

Proposition

The language $HALT = \{\langle M, w\rangle \mid M \text{ halts on input } w\}$ *is undecidable.*

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. Will give reduction f to show $A_{\mathrm{TM}} \leq_m \text{HALT} \implies \text{HALT}$ undecidable.
Let $f(\langle M, w\rangle) = \langle N, w\rangle$ where N is a TM that behaves as follows:

```
On input x
    Run M on x
    If M accepts then halt and accept
    If M rejects then go into an infinite loop
```

N halts on input w if and only if M accepts w.

The Halting Problem

Proposition

The language $HALT = \{\langle M, w\rangle \mid M \text{ halts on input } w\}$ *is undecidable.*

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. Will give reduction f to show $A_{\mathrm{TM}} \leq_m \text{HALT} \implies \text{HALT}$ undecidable.
Let $f(\langle M, w\rangle) = \langle N, w\rangle$ where N is a TM that behaves as follows:

```
On input x
    Run M on x
    If M accepts then halt and accept
    If M rejects then go into an infinite loop
```

N halts on input w if and only if M accepts w. i.e., $\langle M, w\rangle \in A_{\mathrm{TM}}$ iff $f(\langle M, w\rangle) \in \text{HALT}$ □

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable.

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. For the sake of contradiction, suppose there is a decider B for E_{TM}.

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. For the sake of contradiction, suppose there is a decider B for E_{TM}. Then we first transform $\langle M, w\rangle$ to $\langle M_1\rangle$ which is the following:

On input x

  If $x \neq w$, reject

  else run M on w , and accept if M accepts w

, and accept if B rejects $\langle M_1\rangle$, and rejects if B accepts $\langle M_1\rangle$.

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. For the sake of contradiction, suppose there is a decider B for E_{TM}. Then we first transform $\langle M, w\rangle$ to $\langle M_1\rangle$ which is the following:

```
On input x
    If x ≠ w, reject
    else run M on w , and accept if M accepts w
```

, and accept if B rejects $\langle M_1\rangle$, and rejects if B accepts $\langle M_1\rangle$.

Then we show that (1) if $\langle M, w\rangle \in A_{\mathrm{TM}}$, then accept, and (2) $\langle M, w\rangle \in A_{\mathrm{TM}}$, then reject. (how?)

Emptiness of Turing Machines

Proposition

The language $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$ *is not decidable.*

Note: in fact, E_{TM} is not recognizable.

Proof.

Recall $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid w \in L(M)\}$ is undecidable. For the sake of contradiction, suppose there is a decider B for E_{TM}. Then we first transform $\langle M, w\rangle$ to $\langle M_1\rangle$ which is the following:

```
On input x
    If x ≠ w, reject
    else run M on w , and accept if M accepts w
```

, and accept if B rejects $\langle M_1\rangle$, and rejects if B accepts $\langle M_1\rangle$.

Then we show that (1) if $\langle M, w\rangle \in A_{\mathrm{TM}}$, then accept, and (2) $\langle M, w\rangle \in A_{\mathrm{TM}}$, then reject. (how?) This implies A_{TM} is decidable, which is a contradiction. $\square$

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ *is undecidable.*

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ *is undecidable.*

Proof.

We give a reduction f from A_{TM} to REGULAR.

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ *is undecidable.*

Proof.

We give a reduction f from A_{TM} to REGULAR. Let $f(\langle M, w \rangle) = N$, where N is a TM that works as follows:

```
On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does
```

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M)$ *is regular*$\}$ *is undecidable.*

Proof.

We give a reduction f from A_{TM} to REGULAR. Let $f(\langle M, w\rangle) = N$, where N is a TM that works as follows:

```
On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does
```

If $w \in L(M)$ then $L(N) =$

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ is undecidable.

Proof.

We give a reduction f from A_{TM} to REGULAR. Let $f(\langle M, w\rangle) = N$, where N is a TM that works as follows:

```
On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does
```

If $w \in L(M)$ then $L(N) = \Sigma^*$.

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M)$ *is regular*$\}$ *is undecidable.*

Proof.

We give a reduction f from A_{TM} to REGULAR. Let $f(\langle M, w\rangle) = N$, where N is a TM that works as follows:

```
On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does
```

If $w \in L(M)$ then $L(N) = \Sigma^*$. If $w \notin L(M)$ then $L(N) =$

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ *is undecidable.*

Proof.

We give a reduction f from A_{TM} to REGULAR. Let $f(\langle M, w\rangle) = N$, where N is a TM that works as follows:

```
On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does
```

If $w \in L(M)$ then $L(N) = \Sigma^*$. If $w \notin L(M)$ then $L(N) = \{0^n1^n \mid n \geq 0\}$.

Checking Regularity

Proposition

The language $REGULAR = \{M \mid L(M) \text{ is regular}\}$ *is undecidable.*

Proof.

We give a reduction f from A_{TM} to REGULAR. Let $f(\langle M, w\rangle) = N$, where N is a TM that works as follows:

On input x
    If x is of the form 0^n1^n then accept x
    else run M on w and accept x only if M does

If $w \in L(M)$ then $L(N) = \Sigma^*$. If $w \notin L(M)$ then $L(N) = \{0^n1^n \mid n \geq 0\}$. Thus, $\langle N\rangle \in$ REGULAR if and only if $\langle M, w\rangle \in A_{\mathrm{TM}}$ $\square$

Checking Equality

Proposition

$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$ *is not r.e.*

Checking Equality

Proposition

$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$ *is not r.e.*

Proof.

We will give a reduction f from E_{TM} (assume that we know E_{TM} is R.E.) to $\mathrm{EQ}_{\mathrm{TM}}$.

Checking Equality

Proposition

$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$ is not r.e.

Proof.

We will give a reduction f from E_{TM} (assume that we know E_{TM} is R.E.) to $\mathrm{EQ}_{\mathrm{TM}}$. Let M_1 be the Turing machine that on any input, halts and rejects

Checking Equality

Proposition

$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$ is not r.e.

Proof.

We will give a reduction f from E_{TM} (assume that we know E_{TM} is R.E.) to $\mathrm{EQ}_{\mathrm{TM}}$. Let M_1 be the Turing machine that on any input, halts and rejects i.e., $L(M_1) = \emptyset$. Take $f(M) = \langle M, M_1\rangle$.

Checking Equality

Proposition

$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$ *is not r.e.*

Proof.

We will give a reduction f from E_{TM} (assume that we know E_{TM} is R.E.) to $\mathrm{EQ}_{\mathrm{TM}}$. Let M_1 be the Turing machine that on any input, halts and rejects i.e., $L(M_1) = \emptyset$. Take $f(M) = \langle M, M_1\rangle$. Observe $M \in E_{\mathrm{TM}}$ iff $L(M) = \emptyset$ iff $L(M) = L(M_1)$ iff $\langle M, M_1\rangle \in \mathrm{EQ}_{\mathrm{TM}}$. □

Checking Properties

Given M

$$\left.\begin{array}{l}\text{Does } L(M) \text{ contain } M?\\ \text{Is } L(M) \text{ non-empty?}\\ \text{Is } L(M) \text{ empty?}\end{array}\right\}\ \text{Undecidable}$$

Is $L(M)$ infinite?
Is $L(M)$ finite?
Is $L(M)$ co-finite (i.e., is $\overline{L(M)}$ finite)?
Is $L(M) = \Sigma^*$?

Which of these properties can be decided?

Checking Properties

Given M

$$\left.\begin{array}{l}\text{Does } L(M) \text{ contain } M?\\ \text{Is } L(M) \text{ non-empty?}\\ \text{Is } L(M) \text{ empty?}\end{array}\right\}\ \text{Undecidable}$$

$$\left.\begin{array}{l}\text{Is } L(M) \text{ infinite?}\\ \text{Is } L(M) \text{ finite?}\\ \text{Is } L(M) \text{ co-finite (i.e., is } \overline{L(M)} \text{ finite)?}\\ \text{Is } L(M) = \Sigma^*?\end{array}\right\}\ \text{Undecidable}$$

Which of these properties can be decided? None!

Checking Properties

Given M

$$\left.\begin{array}{l}\text{Does } L(M) \text{ contain } M?\\ \text{Is } L(M) \text{ non-empty?}\\ \text{Is } L(M) \text{ empty?}\end{array}\right\} \text{Undecidable}$$

$$\left.\begin{array}{l}\text{Is } L(M) \text{ infinite?}\\ \text{Is } L(M) \text{ finite?}\\ \text{Is } L(M) \text{ co-finite (i.e., is } \overline{L(M)} \text{ finite)?}\\ \text{Is } L(M) = \Sigma^*?\end{array}\right\} \text{Undecidable}$$

Which of these properties can be decided? None! By Rice's Theorem

Properties

Definition
A *property of languages* is simply a set of languages.

Properties

Definition

A *property of languages* is simply a set of languages. We say L *satisfies* the property $\mathbb{P}$ if $L \in \mathbb{P}$.

Properties

Definition

A *property of languages* is simply a set of languages. We say L *satisfies* the property $\mathbb{P}$ if $L \in \mathbb{P}$.

Definition

For any property $\mathbb{P}$, define language $L_{\mathbb{P}}$ to consist of Turing Machines which accept a language in $\mathbb{P}$:

$$L_{\mathbb{P}} = \{M \mid L(M) \in \mathbb{P}\}$$

Properties

Definition

A *property of languages* is simply a set of languages. We say L *satisfies* the property $\mathbb{P}$ if $L \in \mathbb{P}$.

Definition

For any property $\mathbb{P}$, define language $L_{\mathbb{P}}$ to consist of Turing Machines which accept a language in $\mathbb{P}$:

$$L_{\mathbb{P}} = \{M \mid L(M) \in \mathbb{P}\}$$

Deciding $L_{\mathbb{P}}$: deciding if a language represented as a TM satisfies the property $\mathbb{P}$.

- Example: $\{M \mid L(M) \text{ is infinite}\}$

Properties

Definition

A *property of languages* is simply a set of languages. We say L *satisfies* the property $\mathbb{P}$ if $L \in \mathbb{P}$.

Definition

For any property $\mathbb{P}$, define language $L_{\mathbb{P}}$ to consist of Turing Machines which accept a language in $\mathbb{P}$:

$$L_{\mathbb{P}} = \{M \mid L(M) \in \mathbb{P}\}$$

Deciding $L_{\mathbb{P}}$: deciding if a language represented as a TM satisfies the property $\mathbb{P}$.

- Example: $\{M \mid L(M) \text{ is infinite}\}$; $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$

Properties

Definition

A *property of languages* is simply a set of languages. We say L *satisfies* the property $\mathbb{P}$ if $L \in \mathbb{P}$.

Definition

For any property $\mathbb{P}$, define language $L_{\mathbb{P}}$ to consist of Turing Machines which accept a language in $\mathbb{P}$:

$$L_{\mathbb{P}} = \{M \mid L(M) \in \mathbb{P}\}$$

Deciding $L_{\mathbb{P}}$: deciding if a language represented as a TM satisfies the property $\mathbb{P}$.

- Example: $\{M \mid L(M) \text{ is infinite}\}$; $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$
- Non-example: $\{M \mid M \text{ has 15 states}\}$

Properties

Definition

A *property of languages* is simply a set of languages. We say L *satisfies* the property $\mathbb{P}$ if $L \in \mathbb{P}$.

Definition

For any property $\mathbb{P}$, define language $L_{\mathbb{P}}$ to consist of Turing Machines which accept a language in $\mathbb{P}$:

$$L_{\mathbb{P}} = \{M \mid L(M) \in \mathbb{P}\}$$

Deciding $L_{\mathbb{P}}$: deciding if a language represented as a TM satisfies the property $\mathbb{P}$.

- Example: $\{M \mid L(M) \text{ is infinite}\}$; $E_{\mathrm{TM}} = \{M \mid L(M) = \emptyset\}$
- Non-example: $\{M \mid M \text{ has 15 states}\}$ $\longleftarrow$ This is a property of TMs, and not languages!

Trivial Properties

Definition

A property is *trivial* if either it is not satisfied by any r.e. language, or if it is satisfied by all r.e. languages.

Trivial Properties

Definition

A property is *trivial* if either it is not satisfied by any r.e. language, or if it is satisfied by all r.e. languages. Otherwise it is *non-trivial*.

Trivial Properties

Definition

A property is *trivial* if either it is not satisfied by any r.e. language, or if it is satisfied by all r.e. languages. Otherwise it is *non-trivial*.

Example

Some trivial properties:

- $\mathbb{P}_{\text{ALL}}$ = set of all languages
- $\mathbb{P}_{\text{R.E.}}$ = set of all r.e. languages
- $\overline{\mathbb{P}}$ where $\mathbb{P}$ is trivial

Trivial Properties

Definition

A property is *trivial* if either it is not satisfied by any r.e. language, or if it is satisfied by all r.e. languages. Otherwise it is *non-trivial*.

Example

Some trivial properties:

- $\mathbb{P}_{\mathrm{ALL}}$ = set of all languages
- $\mathbb{P}_{\mathrm{R.E.}}$ = set of all r.e. languages
- $\overline{\mathbb{P}}$ where $\mathbb{P}$ is trivial
- $\mathbb{P} = \{L \mid L$ is recognized by a TM with an even number of states$\}$

Trivial Properties

Definition

A property is *trivial* if either it is not satisfied by any r.e. language, or if it is satisfied by all r.e. languages. Otherwise it is *non-trivial*.

Example

Some trivial properties:

- $\mathbb{P}_{\mathrm{ALL}}$ = set of all languages
- $\mathbb{P}_{\mathrm{R.E.}}$ = set of all r.e. languages
- $\overline{\mathbb{P}}$ where $\mathbb{P}$ is trivial
- $\mathbb{P} = \{L \mid L$ is recognized by a TM with an even number of states$\} = \mathbb{P}_{\mathrm{R.E.}}$

Trivial Properties

Definition

A property is *trivial* if either it is not satisfied by any r.e. language, or if it is satisfied by all r.e. languages. Otherwise it is *non-trivial*.

Example

Some trivial properties:

- $\mathbb{P}_{\mathrm{ALL}}$ = set of all languages
- $\mathbb{P}_{\mathrm{R.E.}}$ = set of all r.e. languages
- $\overline{\mathbb{P}}$ where $\mathbb{P}$ is trivial
- $\mathbb{P} = \{L \mid L$ is recognized by a TM with an even number of states$\} = \mathbb{P}_{\mathrm{R.E.}}$

Observation. For any trivial property $\mathbb{P}$, $L_{\mathbb{P}}$ is decidable. (Why?)

Trivial Properties

Definition

A property is *trivial* if either it is not satisfied by any r.e. language, or if it is satisfied by all r.e. languages. Otherwise it is *non-trivial*.

Example

Some trivial properties:

- $\mathbb{P}_{\text{ALL}}$ = set of all languages
- $\mathbb{P}_{\text{R.E.}}$ = set of all r.e. languages
- $\overline{\mathbb{P}}$ where $\mathbb{P}$ is trivial
- $\mathbb{P} = \{L \mid L$ is recognized by a TM with an even number of states$\} = \mathbb{P}_{\text{R.E.}}$

Observation. For any trivial property $\mathbb{P}$, $L_{\mathbb{P}}$ is decidable. (Why?) Then $L_{\mathbb{P}} = \Sigma^*$ or $L_{\mathbb{P}} = \emptyset$.

Rice's Theorem

Proposition

If $\mathbb{P}$ *is a non-trivial property, then* $L_{\mathbb{P}}$ *is undecidable.*

Rice's Theorem

Proposition

If $\mathbb{P}$ is a non-trivial property, then $L_{\mathbb{P}}$ is undecidable.

- Thus $\{M \mid L(M) \in \mathbb{P}\}$ is not decidable (unless $\mathbb{P}$ is trivial)

Rice's Theorem

Proposition

If $\mathbb{P}$ is a non-trivial property, then $L_{\mathbb{P}}$ is undecidable.

- Thus $\{M \mid L(M) \in \mathbb{P}\}$ is not decidable (unless $\mathbb{P}$ is trivial)

We cannot algorithmically determine any interesting property of languages represented as Turing Machines!

Properties of TMs

Note. Properties of TMs, as opposed to those of languages they accept, may or may not be decidable.

Properties of TMs

Note. Properties of TMs, as opposed to those of languages they accept, may or may not be decidable.

Example

$$\left.\begin{array}{l}\{\langle M\rangle \mid M \text{ has 193 states}\}\\ \{\langle M\rangle \mid M \text{ uses at most 32 tape cells on blank input}\}\end{array}\right\}\ \text{Decidable}$$

$$\left.\begin{array}{l}\{\langle M\rangle \mid M \text{ halts on blank input}\}\\ \{\langle M\rangle \mid \text{ on input 0011 } M \text{ at some point writes the symbol \$ on its tape}\}\end{array}\right\}\ \text{Undecidable}$$

Proof of Rice's Theorem

Rice's Theorem

If $\mathbb{P}$ is a non-trivial property, then $L_{\mathbb{P}}$ is undecidable.

Proof.

Proof of Rice's Theorem

Rice's Theorem

If $\mathbb{P}$ is a non-trivial property, then $L_{\mathbb{P}}$ is undecidable.

Proof.

- Suppose $\mathbb{P}$ non-trivial and $\emptyset \notin \mathbb{P}$.

Proof of Rice's Theorem

Rice's Theorem

If $\mathbb{P}$ is a non-trivial property, then $L_{\mathbb{P}}$ is undecidable.

Proof.

- Suppose $\mathbb{P}$ non-trivial and $\emptyset \notin \mathbb{P}$.
  - (If $\emptyset \in \mathbb{P}$, then in the following we will be showing $L_{\overline{\mathbb{P}}}$ is undecidable. Then $L_{\mathbb{P}} = \overline{L_{\overline{\mathbb{P}}}}$ is also undecidable.)

Proof of Rice's Theorem

Rice's Theorem

If $\mathbb{P}$ is a non-trivial property, then $L_{\mathbb{P}}$ is undecidable.

Proof.

- Suppose $\mathbb{P}$ non-trivial and $\emptyset \notin \mathbb{P}$.
  - (If $\emptyset \in \mathbb{P}$, then in the following we will be showing $L_{\overline{\mathbb{P}}}$ is undecidable. Then $L_{\mathbb{P}} = \overline{L_{\overline{\mathbb{P}}}}$ is also undecidable.)
- Recall $L_{\mathbb{P}} = \{\langle M \rangle \mid L(M) \text{ satisfies } \mathbb{P}\}$. We'll reduce A_{TM} to $L_{\mathbb{P}}$.

Proof of Rice's Theorem

Rice's Theorem

If $\mathbb{P}$ is a non-trivial property, then $L_{\mathbb{P}}$ is undecidable.

Proof.

- Suppose $\mathbb{P}$ non-trivial and $\emptyset \notin \mathbb{P}$.
  - (If $\emptyset \in \mathbb{P}$, then in the following we will be showing $L_{\overline{\mathbb{P}}}$ is undecidable. Then $L_{\mathbb{P}} = \overline{L_{\overline{\mathbb{P}}}}$ is also undecidable.)
- Recall $L_{\mathbb{P}} = \{\langle M \rangle \mid L(M) \text{ satisfies } \mathbb{P}\}$. We'll reduce A_{TM} to $L_{\mathbb{P}}$.
- Then, since A_{TM} is undecidable, $L_{\mathbb{P}}$ is also undecidable.

Proof of Rice's Theorem

Rice's Theorem

If $\mathbb{P}$ is a non-trivial property, then $L_{\mathbb{P}}$ is undecidable.

Proof.

- Suppose $\mathbb{P}$ non-trivial and $\emptyset \notin \mathbb{P}$.
  - (If $\emptyset \in \mathbb{P}$, then in the following we will be showing $L_{\overline{\mathbb{P}}}$ is undecidable. Then $L_{\mathbb{P}} = \overline{L_{\overline{\mathbb{P}}}}$ is also undecidable.)
- Recall $L_{\mathbb{P}} = \{\langle M \rangle \mid L(M) \text{ satisfies } \mathbb{P}\}$. We'll reduce A_{TM} to $L_{\mathbb{P}}$.
- Then, since A_{TM} is undecidable, $L_{\mathbb{P}}$ is also undecidable. $\cdot\!\cdot\!\longrightarrow$

Proof of Rice's Theorem

Proof (contd).

Since $\mathbb{P}$ is non-trivial, at least one r.e. language satisfies $\mathbb{P}$.

Proof of Rice's Theorem

Proof (contd).

Since $\mathbb{P}$ is non-trivial, at least one r.e. language satisfies $\mathbb{P}$. i.e., $L(M_0) \in \mathbb{P}$ for some TM M_0.

Proof of Rice's Theorem

Proof (contd).

Since $\mathbb{P}$ is non-trivial, at least one r.e. language satisfies $\mathbb{P}$. i.e., $L(M_0) \in \mathbb{P}$ for some TM M_0.
Will show a reduction f that maps an instance $\langle M, w \rangle$ for A_{TM}, to N such that

- If M accepts w then N accepts the same language as M_0.
  - Then $L(N) = L(M_0) \in \mathbb{P}$
- If M does not accept w then N accepts $\emptyset$.
  - Then $L(N) = \emptyset \notin \mathbb{P}$

Proof of Rice's Theorem

Proof (contd).

Since $\mathbb{P}$ is non-trivial, at least one r.e. language satisfies $\mathbb{P}$. i.e., $L(M_0) \in \mathbb{P}$ for some TM M_0.
Will show a reduction f that maps an instance $\langle M, w \rangle$ for A_{TM}, to N such that

- If M accepts w then N accepts the same language as M_0.
  - Then $L(N) = L(M_0) \in \mathbb{P}$
- If M does not accept w then N accepts $\emptyset$.
  - Then $L(N) = \emptyset \notin \mathbb{P}$

Thus, $\langle M, w \rangle \in A_{\mathrm{TM}}$ iff $N \in L_{\mathbb{P}}$.

Proof of Rice's Theorem

Proof (contd).

Since $\mathbb{P}$ is non-trivial, at least one r.e. language satisfies $\mathbb{P}$. i.e., $L(M_0) \in \mathbb{P}$ for some TM M_0.

Will show a reduction f that maps an instance $\langle M, w \rangle$ for A_{TM}, to N such that

- If M accepts w then N accepts the same language as M_0.
  - Then $L(N) = L(M_0) \in \mathbb{P}$
- If M does not accept w then N accepts $\emptyset$.
  - Then $L(N) = \emptyset \notin \mathbb{P}$

Thus, $\langle M, w \rangle \in A_{\mathrm{TM}}$ iff $N \in L_{\mathbb{P}}$. $\cdots\!\!\longrightarrow$

Proof of Rice's Theorem

Proof (contd).

The reduction f maps $\langle M, w\rangle$ to N, where N is a TM that behaves as follows:

```
On input x
    Ignore the input and run M on w
    If M does not accept (or doesn't halt)
        then do not accept x (or do not halt)
    If M does accept w
        then run M_0 on x and accept x iff M_0 does.
```

Notice that indeed if M accepts w then $L(N) = L(M_0)$. Otherwise $L(N) = \emptyset$. $\square$

Rice's Theorem

Recap

Every non-trivial property of r.e. languages is undecidable

Rice's Theorem
Recap

Every non-trivial property of r.e. languages is undecidable

- Rice's theorem says nothing about properties of Turing machines

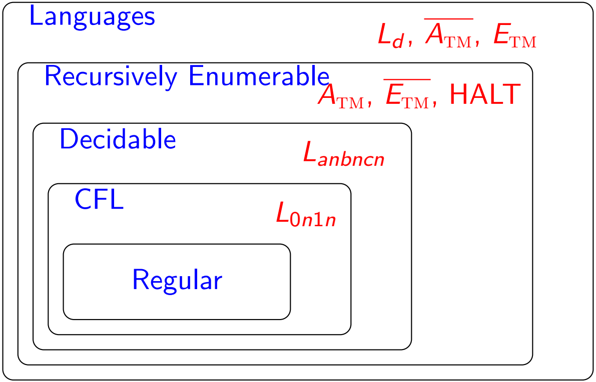
Rice's Theorem

Recap

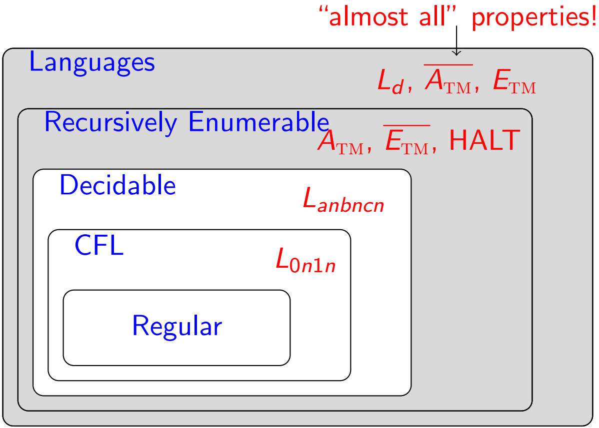
Every non-trivial property of r.e. languages is undecidable

- Rice's theorem says nothing about properties of Turing machines
- Rice's theorem says nothing about whether a property of languages is recurisvely enumerable or not.

Big Picture . . . again



Big Picture . . . again



Boolean Operators

Boolean Operators

Proposition

Decidable languages are closed under union, intersection, and complementation.

Boolean Operators

Proposition

Decidable languages are closed under union, intersection, and complementation.

Proof.

Given TMs M_1, M_2 that decide languages L_1, and L_2

- A TM that decides $L_1 \cup L_2$: on input x, run M_1 and M_2 on x, and accept iff either accepts.

Boolean Operators

Proposition

Decidable languages are closed under union, intersection, and complementation.

Proof.

Given TMs M_1, M_2 that decide languages L_1, and L_2

- A TM that decides $L_1 \cup L_2$: on input x, run M_1 and M_2 on x, and accept iff either accepts. (Similarly for intersection.)

Boolean Operators

Proposition

Decidable languages are closed under union, intersection, and complementation.

Proof.

Given TMs M_1, M_2 that decide languages L_1, and L_2

- A TM that decides $L_1 \cup L_2$: on input x, run M_1 and M_2 on x, and accept iff either accepts. (Similarly for intersection.)
- A TM that decides $\overline{L_1}$: On input x, run M_1 on x, and accept if M_1 rejects, and reject if M_1 accepts. $\square$

Regular Operators

Proposition

Decidable languages are closed under concatenation and Kleene Closure.

Proof.

Given TMs M_1 and M_2 that decide languages L_1 and L_2.

- A TM to decide $L_1 L_2$:

Regular Operators

Proposition

Decidable languages are closed under concatenation and Kleene Closure.

Proof.

Given TMs M_1 and M_2 that decide languages L_1 and L_2.

- A TM to decide $L_1 L_2$: On input x, for each of the $|x| + 1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.

Regular Operators

Proposition

Decidable languages are closed under concatenation and Kleene Closure.

Proof.

Given TMs M_1 and M_2 that decide languages L_1 and L_2.

- A TM to decide $L_1 L_2$: On input x, for each of the $|x| + 1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to decide L_1^*:

Regular Operators

Proposition

Decidable languages are closed under concatenation and Kleene Closure.

Proof.

Given TMs M_1 and M_2 that decide languages L_1 and L_2.

- A TM to decide $L_1 L_2$: On input x, for each of the $|x| + 1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to decide L_1^*: On input x, if $x = \epsilon$ accept. Else, for each of the $2^{|x|-1}$ ways to divide x as $w_1 \dots w_k$ $(w_i \neq \epsilon)$: run M_1 on each w_i and accept if M_1 accepts all. Else reject. $\square$

Boolean Operators

Proposition

R.E. languages are closed under union, and intersection.

Boolean Operators

Proposition

R.E. languages are closed under union, and intersection.

Proof.

Given TMs M_1, M_2 that recognize languages L_1, L_2

Boolean Operators

Proposition

R.E. languages are closed under union, and intersection.

Proof.

Given TMs M_1, M_2 that recognize languages L_1, L_2

- A TM that recognizes $L_1 \cup L_2$: on input x, run M_1 and M_2 on x in parallel, and accept iff either accepts.

Boolean Operators

Proposition

R.E. languages are closed under union, and intersection.

Proof.

Given TMs M_1, M_2 that recognize languages L_1, L_2

- A TM that recognizes $L_1 \cup L_2$: on input x, run M_1 and M_2 on x in parallel, and accept iff either accepts. (Similarly for intersection; but no need for parallel simulation) □

Complementation

Proposition

R.E. languages are not closed under complementation.

Proof.

A_{TM} is r.e. but $\overline{A_{\mathrm{TM}}}$ is not. $\square$

Regular Operations

Proposition

R.E languages are closed under concatenation and Kleene closure.

Proof.

Given TMs M_1 and M_2 recognizing L_1 and L_2

- A TM to recognize $L_1 L_2$:

Regular Operations

Proposition

R.E languages are closed under concatenation and Kleene closure.

Proof.

Given TMs M_1 and M_2 recognizing L_1 and L_2

- A TM to recognize $L_1 L_2$: On input x, do in parallel, for each of the $|x| + 1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.

Regular Operations

Proposition

R.E languages are closed under concatenation and Kleene closure.

Proof.

Given TMs M_1 and M_2 recognizing L_1 and L_2

- A TM to recognize $L_1 L_2$: On input x, do in parallel, for each of the $|x|+1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to recognize L_1^*:

Regular Operations

Proposition

R.E languages are closed under concatenation and Kleene closure.

Proof.

Given TMs M_1 and M_2 recognizing L_1 and L_2

- A TM to recognize $L_1 L_2$: On input x, do in parallel, for each of the $|x| + 1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to recognize L_1^*: On input x, if $x = \epsilon$ accept. Else, do in parallel, for each of the $2^{|x|-1}$ ways to divide x as $w_1 \ldots w_k$ ($w_i \neq \epsilon$): run M_1 on each w_i and accept if M_1 accepts all. Else reject. $\square$