INT201 Decision, Computation and Language

Lecture 11 –Reducibility and Rice’s Theorem

Dr Yushi Li and Dr Chunchuan Lyu

Xi’an Jiaotong-Liverpool University
西交利物浦大学

Recap

- Cantor’s Diagonalization Method
- Church-Turing Thesis and Universal Turing Machine
- Examples of decidable languages
- Existence of undecidable language and non-Turing recognizable language

Today

- Reducibility
- The Halting Problem and other undecidable languages
- Rice’s Theorem

John von Neumann 1903-1957

Hungarian-American mathematician, physicist, computer scientist, engineer and polymath,

- The axiomatic construction of general set theory (PhD thesis, 1925)
- Private communication to Kurt Gödel about his independent discovery of the second theorem of incompleteness (1930)
- Stopped working foundation of mathematics after 1931
- Axiomatization of Quantum Physics and Functional Analysis
- Mean Ergodic Theorem
- Minimax Theorem and Duality in Linear Programming
- Von Neumann Architecture, Merge Sort, Cellular Automata
- The Computer and the Brain (1958) and Singularity

"Young man, in mathematics you don't understand things. You just get used to them."

Manhattan Project

Die of Cancer

On Computable Numbers, with an Application to the Entscheidungs-problem by Alan Turing

Reviewer:

This is a bizarre paper. It begins by defining a computing device absolutely unlike anything I have seen, then proceeds to show—I haven't quite followed the needlessly complicated formalism—that there are numbers that it can't compute. As I see it, there are two alternatives that apply to any machine that will ever be built: Either these numbers are too big to be represented in the machine, in which case the conclusion is obvious, or they are not; in that case, a machine that can't compute them is simply broken!

"Young man, in mathematics you don't understand things. You just get used to them."

What we are learning is the kind of mathematics that you need to get used to.

Undecidable Problems

Definition

Undecidable problem. The associated language of a problem cannot be recognized by a TM that halts for all inputs.

Unrecognizable problem. The associated language of a problem cannot be recognized by a TM.

The languages of TMs are undecidable but recognizable

$$L_{TM} = \{\langle M, w\rangle : M \text{ is a Turing machine that accepts the string } w\}$$

We consider undecidable problems unsolvable (informal).

Say the TM has run 6 weeks without giving a response. It is possible that it is not in the language, but it is also possible it is in the language and we only need to wait longer.
For decidable languages, there is an upper bound of the waiting time.
For undecidable languages, the waiting has no time limit.

Reducibility

Definition

Reduction is a way of converting one problem to another problem, so that the solution to the second problem can be used to solve the first problem.

If A **reduces** to B, then any solution of B solves A (Reduction always involves two problems, A and B).

- If A is reducible to B, then A cannot be harder than B.
- If A is reducible to B and B is decidable, then A is also decidable.
- If A is reducible to B and A is undecidable, then B is also undecidable.

Reducibility

A common strategy for proving that a language L is undecidable is by reduction method, proceeding as follows:

Typical approach to show L is undecidable via reduction from A to L:

- Find a problem A known to be undecidable
- Suppose L is decidable.
- Let R be a TM that decides L.
- Using R as subroutine to construct another TM S that decides A.
- But A is not decidable.
- Conclusion: L is not decidable.

Mapping Reduction

Definition

Suppose that A and B are two languages

- A is defined over alphabet Σ_1^*, so $A \subseteq \Sigma_1^*$
- B is defined over alphabet Σ_2^*, so $B \subseteq \Sigma_2^*$

Then A is **mapping reducible** to B, written

$$A \leq_m B$$

if there is a computable (Turing Machine can simulate through the tape) function

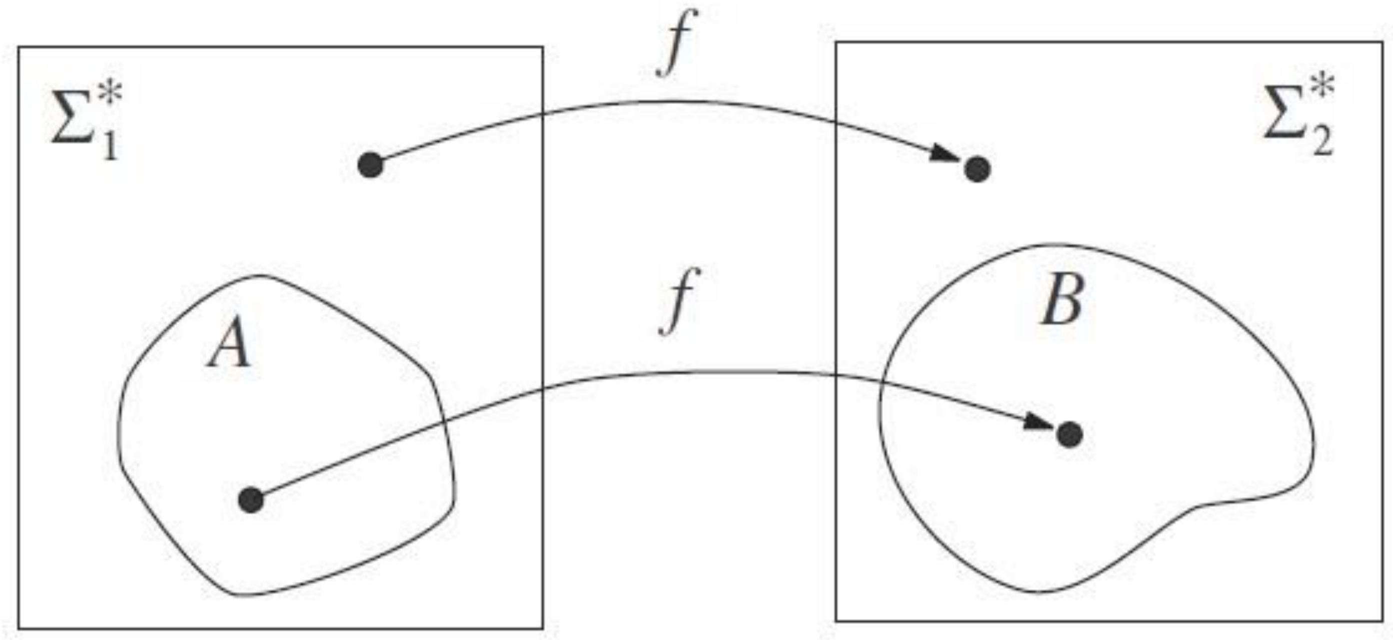
$$f : \Sigma_1^* \rightarrow \Sigma_2^*$$

such that, for every $w \in \Sigma_1^*$

$$w \in A \Longleftrightarrow f(w) \in B$$

The function f is called a **reduction** of A to B.

Mapping Reduction

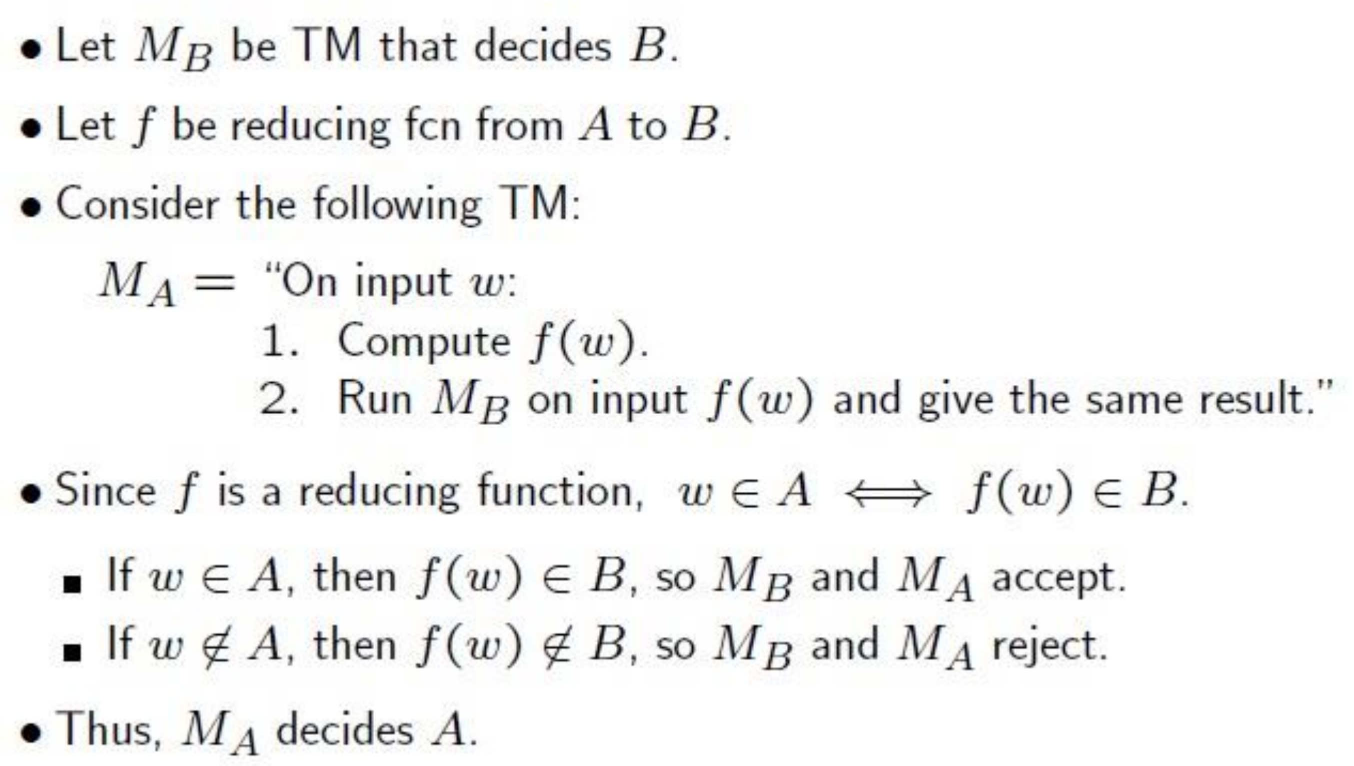


$$w \in A \Longleftrightarrow f(w) \in B$$

Theorem

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

Proof



- Let M_B be TM that decides B.
- Let f be reducing fcn from A to B.
- Consider the following TM:

```
M_A = "On input w:
        1. Compute f(w).
        2. Run M_B on input f(w) and give the same result."
```

- Since f is a reducing function, $w \in A \iff f(w) \in B$.
  - If $w \in A$, then $f(w) \in B$, so M_B and M_A accept.
  - If $w \notin A$, then $f(w) \notin B$, so M_B and M_A reject.
- Thus, M_A decides A.

Corollary

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

Theorem

If $A \leq_m B$ and B is Turing-recognizable, then A is Turing-recognizable.

Corollary

If $A \leq_m B$ and A is not Turing-recognizable, then B is not Turing-recognizable.

Theorem

If $A \leq_m B$, then $\bar{A} \leq_m \bar{B}$.

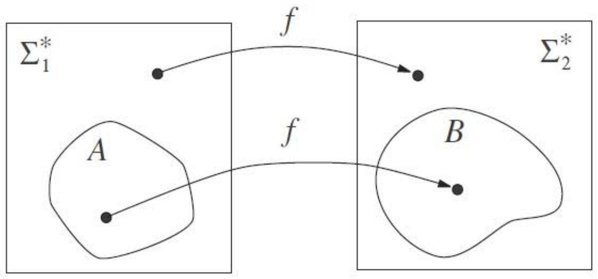
Corollary

L_{TM} is not mapping reducible to $E_{TM}=\{< M > \mid M \text{ is a } TM \text{ and } L(M) = \emptyset\}$.

But is mapping reduction all we have?

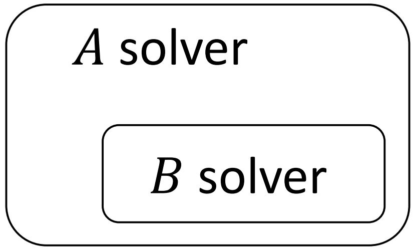
Mapping Reduction v.s. Turing (General) Reduction

Mapping Reducibility of A to B:
Translate A-questions to B-questions.



$$w \in A \iff f(w) \in B$$

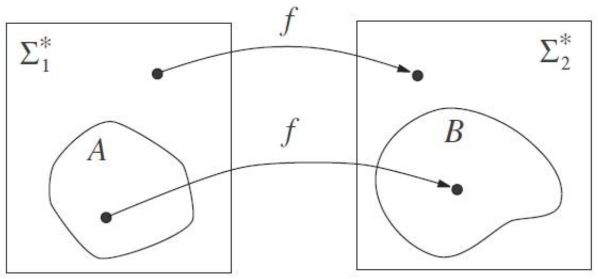
Turing Reducibility of A to B: Use B solver to solve A.



Clearly, we can use mapping reduction to construct general reduction.

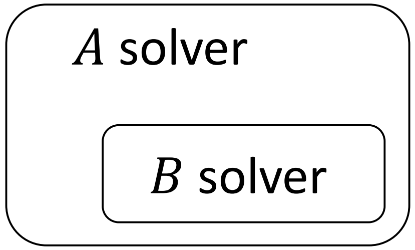
Mapping Reduction v.s. (General) Reduction

Mapping Reducibility of A to B:
Translate A-questions to B-questions.



$$w \in A \Longleftrightarrow f(w) \in B$$

(General) Reducibility of A to B:
Use B solver to solve A.



However, we have other options such as A accepts when B rejects, calling B multiple times or computing while conditioning on the B solution.

Emptiness of TMs is undecidable

$E_{TM} = \{< M > \mid M\ is\ a\ TM\ and\ L(M) = \emptyset\}.$

Intuitively, we need to show the Turing machine won't enter the accept state for any strings. However, unlike the CFG/DFA cases, we find it hard to enumerate over all possible derivations.

Theorem: E_{TM} is undecidable.

Proof by reduction from $\mathbf{L_{TM}}$:
For a given TM M and input w, we construct a TM M_w s.t. M accepts w iff L(M_w) $\neq \emptyset$.
We claim M_w = "On input x:

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

If M accepts w, in step 1 M_w rejects all strings other than w, and in step 2, M_w accepts it, and hence L(M_w) $\neq \emptyset$.
If L(M_w) $\neq \emptyset$, then as it rejects all strings other than w, and accepts w only when M accepts w. Therefore M accepts w.

Emptiness of TMs is undecidable

$E_{TM} = \{< M > \mid M \text{ is a } TM \text{ and } L(M) = \emptyset\}.$

This is a Turing reduction that is not mapping reducible.

Theorem: E_{TM} is undecidable.

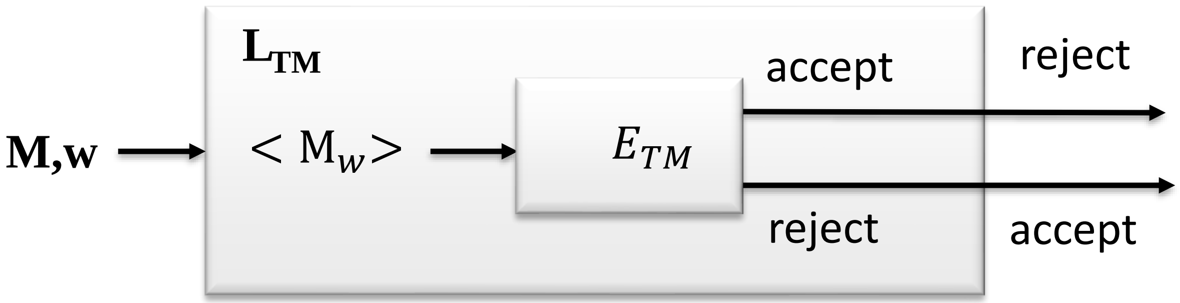
Proof by reduction from $\mathbf{L_{TM}}$:

M_w = "On input x:

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

M accepts w iff $L(M_w) \neq \emptyset$

Assume a decider for E_{TM} exists



But $\mathbf{L_{TM}}$ is undecidable, so must E_{TM}.

Halting problem for TMs is undecidable

L_{TM} (acceptance problem for TMs) is undecidable, where

$$L_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts string } w\}$$

Define related problem:

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on string } w\}$$

L_{TM} and $HALT_{TM}$ has same universe:

$$\Omega=\{\langle M, w\rangle \mid M \text{ is TM, } w \text{ is string}\}$$

Given a $\langle M, w\rangle \in \Omega$:

- if M halts on input w, then $\langle M, w\rangle \in HALT_{TM}$,
- if M doesn't halt on input w, then $\langle M, w\rangle \notin HALT_{TM}$.

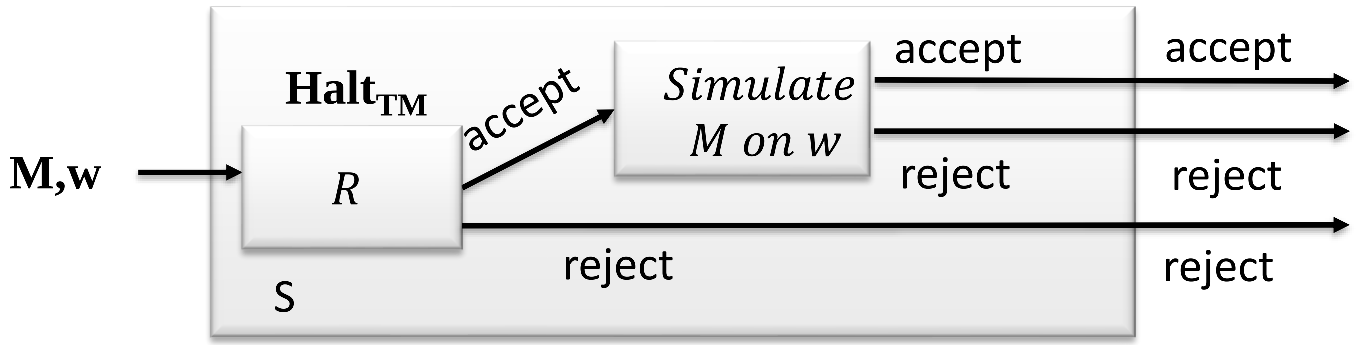
Halting problem for TMs is undecidable

Proof by general reduction from $\mathrm{L_{TM}}$

We reduce $\mathrm{L_{TM}}$ to $HALT_{TM}$. We claim the following machine S decides $\mathrm{L_{TM}}$

```
S = "On input ⟨M, w⟩, an encoding of a TM M and a string w:
    1. Run TM R on input ⟨M, w⟩.
    2. If R rejects, reject.
    3. If R accepts, simulate M on w until it halts.
    4. If M has accepted, accept; if M has rejected, reject."
```

Clearly, S accepts <M,w> iff M halts on w and M accept w.
S rejects, iff M loops on w or M halts and rejects w.



Now, as $\mathrm{L_{TM}}$ is undecidable, so must $HALT_{TM}$

Halting problem for TMs is undecidable

Proof by contradiction

Assume halting is decidable, we have that TM H, where

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

Now, we use exactly the same strategy as in L_{TM} except we loop when we should reject, we have the following TM D:

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

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs. That is, if H accepts, *loop*; and if H rejects, *accept*."

We then ask, does D halts on <D>?
Clearly, if it halts, H accepts, then D loop.
If it does not halt, H rejects, then D accept (hence halt).
We have a contradiction.

Other Undecidable Problems

Define T to be the set of all Turing machines descriptions, i.e.,

$\mathrm{T} = \{\langle \mathrm{M} \rangle$: M is a Turing machine$\}$

- $INFINITE_{TM} = \{< M > | M \; is \; a \; TM \; and \; |L(M)| = \infty\}$.
- $LT_{TM} = \{< M > | M \; is \; a \; TM \; and \; L(M) = L(T)\}$.
- $FINITE_{TM} = \{< M > | M \; is \; a \; TM \; and \; \exists n \in N, |L(M)| = n\}$.
- $ALL_{TM} = \{< M > | M \; is \; a \; TM \; and \; L(M) = \Sigma^*\}$

Is it possible to prove them all at once?

Non-trivial Properties and Rice’s Theorem

- $E_{TM} = \{< M > \mid M \text{ is a } TM \text{ and } L(M) = \emptyset\}$.
- $INFINITE_{TM} = \{< M > | M \text{ is a } TM \text{ and } |L(M)| = \infty\}$.
- $LT_{TM} = \{< M > | M \text{ is a } TM \text{ and } L(M) = L(T)\}$.
- $FINITE_{TM} = \{< M > | M \text{ is a } TM \text{ and } \exists n \in N, |L(M)| = n\}$.
- $ALL_{TM} = \{< M > | M \text{ is a } TM \text{ and } L(M) = \Sigma^*\}$

(Informal) Non-trivial properties in common:

1. none of them are empty set
2. none of them includes all Turing machines.
3. <M> ∈ P iff L(M) satisfy some properties,

   i.e., $P = \{< M > | M \text{ is a } TM$ and p(L(M))==1$\}$ $where\ \ p: \{L(M) | M \in T\} \rightarrow \{1,0\}$

(Informal) Rice’s Theorem:

Any non-trivial property of Turing machines is undecidable

Rice's Theorem

Define T to be the set of all Turing machines descriptions, i.e.,

$$\mathrm{T} = \{\langle \mathrm{M} \rangle : \mathrm{M} \text{ is a Turing machine}\}$$

Let P be a subset of T such that

1. $P \neq \emptyset$, i.e., there exists a Turing machine M such that $\langle \mathrm{M} \rangle \in P$,

2. P is a proper subset of T , i.e., there exists a Turing machine N such that $\langle \mathrm{N} \rangle \notin P$,

3. for any two Turing machines M_1 and M_2 with $\mathrm{L}(\mathrm{M}_1) = \mathrm{L}(\mathrm{M}_2)$,

   (a) either both $\langle \mathrm{M}_1 \rangle$ and $\langle \mathrm{M}_2 \rangle$ are in P or

   (b) none of $\langle \mathrm{M}_1 \rangle$ and $\langle \mathrm{M}_2 \rangle$ is in P.

Then the language P is undecidable.

This is a more operational for checking the condition than $\exists\, p : \{L(M) | M \in T\} \rightarrow \{1,0\}$, such that $\langle \mathrm{M} \rangle \in P$ iff $p(L(M)) \ = 1$.

Rice's Theorem and Applications

Let P be a subset of T such that

1. $P \neq \emptyset$,

2. P is a proper subset of T ,

3. for any two Turing machines M_1 and M_2 with $\mathrm{L}(\mathrm{M}_1) = \mathrm{L}(\mathrm{M}_2)$,

   (a) either both $\langle \mathrm{M}_1 \rangle$ and $\langle \mathrm{M}_2 \rangle$ are in P or

   (b) none of $\langle \mathrm{M}_1 \rangle$ and $\langle \mathrm{M}_2 \rangle$ is in P.

$E_{TM} = \{< M > | M \text{ } is \text{ } a \text{ } TM \text{ } and \text{ } L(M) = \emptyset\}$.

1. A TM rejects all inputs will suffice is in the set.
2. A TM accepts all inputs is not in the set.
3. If $\mathrm{L}(\mathrm{M}_1) = \mathrm{L}(\mathrm{M}_2)$, they are both either $\emptyset$ or non-empty, either both in the set or not in the set respectively.

Therefore E_{TM} is undecidable by Rice's theorem

Rice's Theorem and Applications

Let P be a subset of T such that

1. $P \neq \emptyset$,

2. P is a proper subset of T ,

3. for any two Turing machines M_1 and M_2 with $\mathrm{L}(\mathrm{M}_1) = \mathrm{L}(\mathrm{M}_2)$,

   (a) either both $\langle \mathrm{M}_1 \rangle$ and $\langle \mathrm{M}_2 \rangle$ are in P or

   (b) none of $\langle \mathrm{M}_1 \rangle$ and $\langle \mathrm{M}_2 \rangle$ is in P.

$INFINITE_{TM} = \{< M >| M\ is\ a\ TM\ and\ |L(M)| = \infty\}$.

1. A TM accepts all inputs will suffice is in the set.
2. A TM rejects all inputs is not in the set.
3. If $\mathrm{L}(\mathrm{M}_1) = \mathrm{L}(\mathrm{M}_2)$, $|\mathrm{L}(\mathrm{M}_1)| = |\mathrm{L}(\mathrm{M}_2)|$ so they are both either infinite or finite, either both in the set or not in the set respectively.

Therefore $INFINITE_{TM}$ is undecidable by Rice's theorem

Rice's Theorem and Applications

Let P be a subset of T such that

1. $P \neq \emptyset$,

2. P is a proper subset of T ,

3. for any two Turing machines M_1 and M_2 with $\mathrm{L}(\mathrm{M}_1) = \mathrm{L}(\mathrm{M}_2)$,

(a) either both $\langle \mathrm{M}_1 \rangle$ and $\langle \mathrm{M}_2 \rangle$ are in P or

(b) none of $\langle \mathrm{M}_1 \rangle$ and $\langle \mathrm{M}_2 \rangle$ is in P.

$$INFINITE_{TM} = \{< M > | M \ is \ a \ TM \ and \ |L(M)| = \infty\}.$$

Note that there might be $\boldsymbol{M_1, M_2} \in \boldsymbol{INFINITE_{TM}}$ but $\mathbf{L}(\boldsymbol{M_1}) \neq \mathbf{L}(\boldsymbol{M_2})$

Rice's Theorem and Non-Applications

Let P be a subset of T such that

1. $P \neq \emptyset$,

2. P is a proper subset of T ,

3. for any two Turing machines M_1 and M_2 with $\mathrm{L}(\mathrm{M}_1) = \mathrm{L}(\mathrm{M}_2)$,

   (a) either both $\langle \mathrm{M}_1 \rangle$ and $\langle \mathrm{M}_2 \rangle$ are in P or

   (b) none of $\langle \mathrm{M}_1 \rangle$ and $\langle \mathrm{M}_2 \rangle$ is in P.

$FIVE_{TM} = \{< M > | M\ is\ a\ TM\ and\ M\ has\ 5\ states\}$.

We cannot apply Rice's theorem here, as clearly we can have a UTM with 3 states recognize the same language as a given 5 states TM by simulating that machine.

In general, a property is about the language not about the TMs

Rice’s Theorem and Implications

Rice's Theorem: All non-trivial properties of a program's function are undecidable.

• LeetCode Testing Cases: We **cannot** build a program (e.g., a LeetCode checker) that always terminates and guarantees the submitted function works correctly for **all** possible inputs for nontrivial functions.

•LLM Correctness: we cannot have an oracle machine that always terminates and guarantees the answer/reasoning from LLM is correct or even logically consistent.

Rice’s Theorem and Non-Implications

Rice's Theorem: All non-trivial properties of a program's function are undecidable.

- We can still formally prove some functions/code do its’ job.

- We can still prove some statements are true or can be logically proved.

The point of Rice’s theorem is that you do not have a program that works for all and terminates.

All provable statements are TM-recognizable, that is if you do exhaustive search on provable statement, you will find a proof sooner or later. However, if the program keeps running, you don’t know should you run longer or it is actually unprovable.

Rice's Theorem and Proof

Rice's Theorem

Let P be a proper non-empty subset of TM descriptions such that for M_1 and M_2 with $\mathrm{L}(\mathrm{M}_1) = \mathrm{L}(\mathrm{M}_2)$, $\langle \mathrm{M}_1 \rangle \in P$ iff $\langle \mathrm{M}_2 \rangle \in P$. Then, P is undecidable.

Proof attempt by reduction from $\mathbf{L_{TM}}$:

We will reduce $\mathrm{L_{TM}}$ to all non-trivial property problems. Denote any given problem P, we have a **decider $\mathbf{R}_{\boldsymbol{P}}$** and a Turing machine **T such that <T>** $\in \mathbf{P}$. For a given TM M and input w, ideally we construct a TM M_w s.t. M accepts w iff $< \mathrm{M}_w > \in P$.

$M_w^{(T)} =$ "On input x:

1. Simulate M on w. If it halts and rejects, *reject*. If it accepts, proceed to stage 2.
2. Simulate T on x. If it accepts, *accept*."

Rice's Theorem and Proof

Proof attempt by reduction from $\mathbf{L_{TM}}$:

$\boxed{M \text{ accepts w} \rightarrow \ < \mathrm{M}_w(T) > \in P.}$

```
M_w(T) = "On input x:
    1. Simulate M on w. If it halts and rejects, reject.
       If it accepts, proceed to stage 2.
    2. Simulate T on x. If it accepts, accept."
```

If M accepts w, we have L(M_w(T))=L(T). As <T> ∈ P, we have $< \mathrm{M}_w(T) > \in P$.

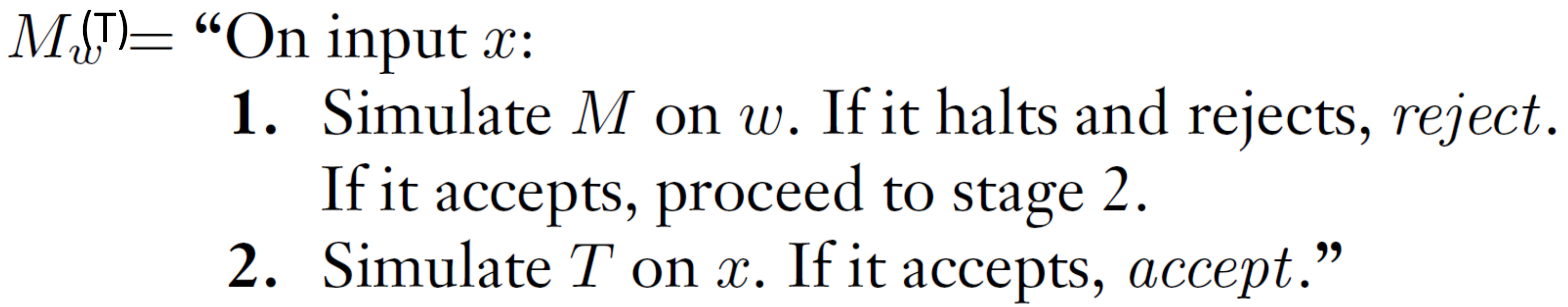
If M rejects or loops on w, we have L(M_w(T))= $\emptyset$. Now < M_w (T)> $\notin P$ iff Turing machines with empty language is not in the property.

Obviously, this is not true for all property.

Can we do something differently when empty language is not in the property?

Rice’s Theorem and Proof

Proof by reduction from L_{TM} :



$M_w(T)$ = “On input x:
1. Simulate M on w. If it halts and rejects, *reject*. If it accepts, proceed to stage 2.
2. Simulate T on x. If it accepts, *accept*.”

Take any P, let $M_\emptyset$ be a TM such that $L(M_\emptyset) = \emptyset$. If $< M_\emptyset >\in P$, we use its’ complement $\bar{P}$.

Obviously, P is a decider iff $\bar{P}$ is a decider.

Moreover, there always exists $<\bar{T}> \in \bar{P}$ as P is a proper subset.

If M accepts w, we have L($M_w(\bar{T})$)=L($\bar{T}$). As $<\bar{T}> \in P$, we have $< M_w > \in \bar{P}$.

If M rejects or loops w, we have L($M_w(\bar{T})$)= $\emptyset$. As we have $< M_\emptyset >\in P$, so $< M_w(\bar{T}) >\notin \bar{P}$

So, we have if $< M_\emptyset >\in P$, M accepts w iff $< M_w(\bar{T}) > \in \bar{P}$.

Rice's Theorem and Proof

Proof by reduction from $\mathbf{L_{TM}}$:

```
M_w(T) = "On input x:
    1. Simulate M on w. If it halts and rejects, reject.
       If it accepts, proceed to stage 2.
    2. Simulate T on x. If it accepts, accept."
```

In all, we have the following results:

If $< \mathrm{M}_\emptyset > \notin P$, M accepts w iff $< \mathrm{M}_w(T) > \in P$.

If $< \mathrm{M}_\emptyset > \in P$, M accepts w iff $< \mathrm{M}_w(\bar{T}) > \in \bar{P}$.

Rice's Theorem and Proof

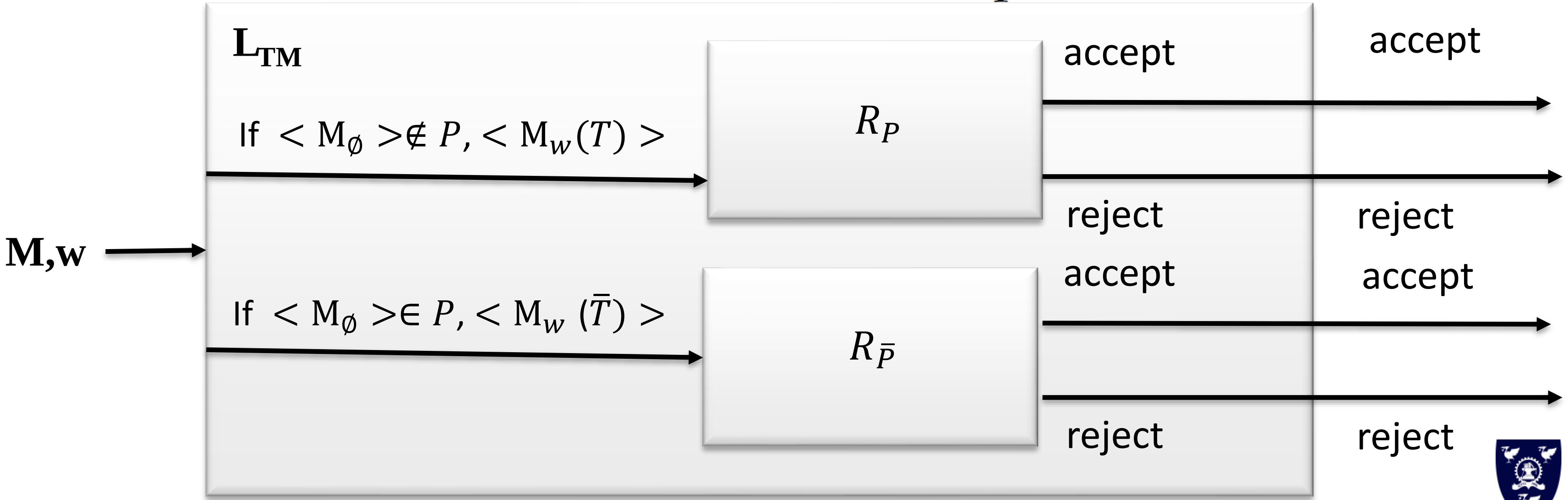
If $< \mathrm{M}_{\emptyset} > \notin P$, M accepts w iff $< \mathrm{M}_w(T) > \in P$.

If $< \mathrm{M}_{\emptyset} > \in P$, M accepts w iff $< \mathrm{M}_w(\bar{T}) > \in \bar{P}$.

Proof by reduction from L_{TM} :

M_w(T)= "On input x:

```
    1. Simulate $M$ on $w$. If it halts and rejects, *reject*.
       If it accepts, proceed to stage 2.
    2. Simulate $T$ on $x$. If it accepts, *accept*."
```

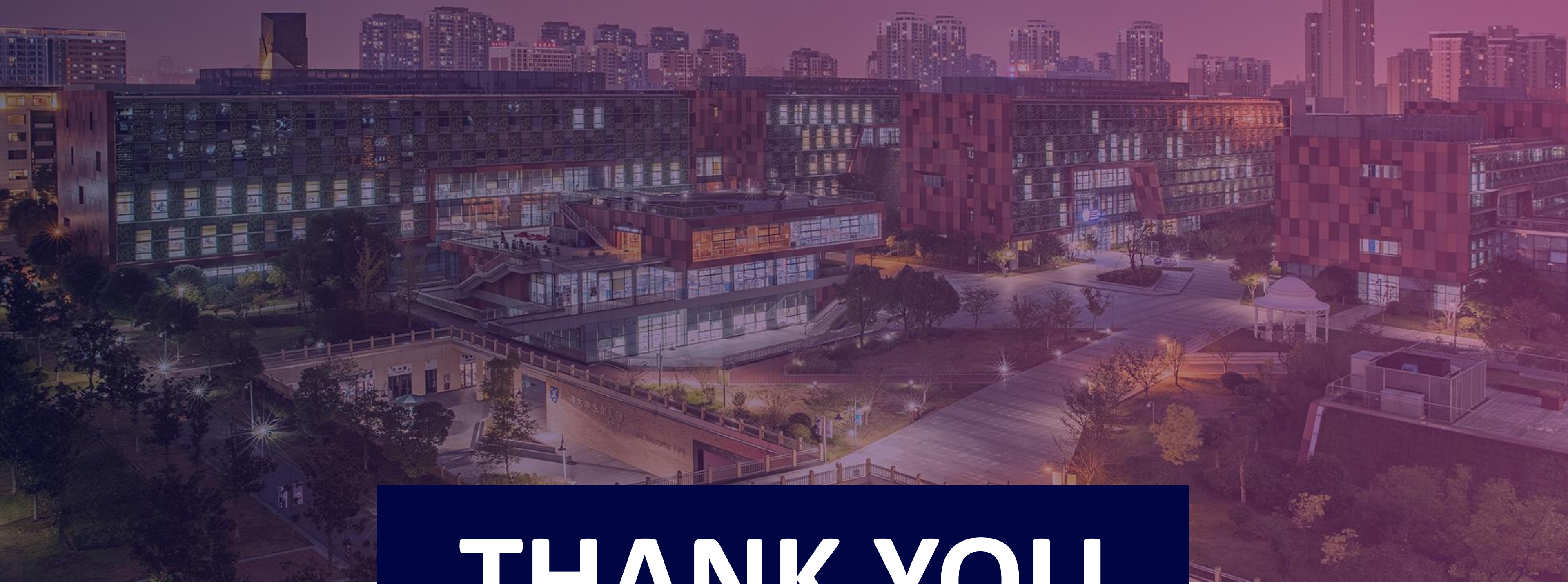


L_{TM}
M,w
If $< \mathrm{M}_{\emptyset} > \notin P, < \mathrm{M}_w(T) >$
R_P
accept
reject
accept
reject
If $< \mathrm{M}_{\emptyset} > \in P, < \mathrm{M}_w(\bar{T}) >$
$R_{\bar{P}}$
accept
reject
accept
reject

Now, as L_{TM} is undecidable, so must all the non-trivial properties.

Quick review

- Reducibility shows a problem can only be reduced to problems that is at least as hard as itself.
- Mapping reduction is powerful but can not cover all cases.
- The Halting Problem and other undecidable languages
- Rice’s Theorem states non-trivial properties is undecidable



THANK YOU

Xi'an Jiaotong-Liverpool University
西交利物浦大學

XJTLU | SCHOOL OF FILM AND TV ARTS