Decidable Languages

Recall that:

A language L is **Turing-Acceptable** if there is a Turing machine M that accepts L

Also known as: *Turing-Recognizable* or *Recursively-enumerable* languages

Turing-Acceptable

For any input string w :

$w \in L \Rightarrow M$ halts in an accept state

$w \notin L \Rightarrow M$ halts in a non-accept state

or loops forever

Definition:

A language L is **decidable**
if there is a Turing machine (decider) M
which accepts L
and halts on every input string

Also known as *recursive languages*

Turing-Decidable

For any input string w :

$w \in L \Rightarrow M$ halts in an accept state

$w \notin L \Rightarrow M$ halts in a non-accept state

Observation:

Every decidable language is Turing-Acceptable

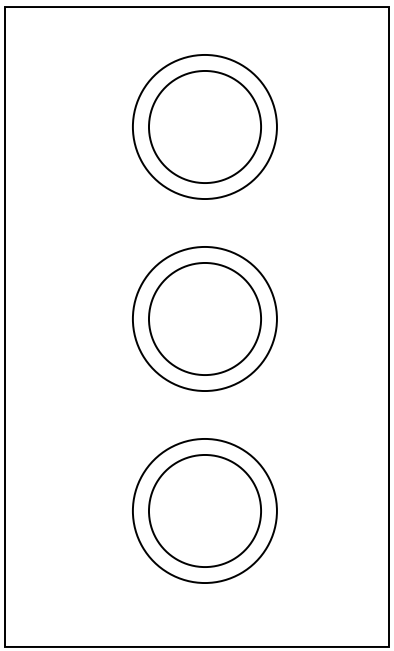
Sometimes, it is convenient to have Turing machines with single accept and reject states

These are the only halting states

That result to possible halting configurations

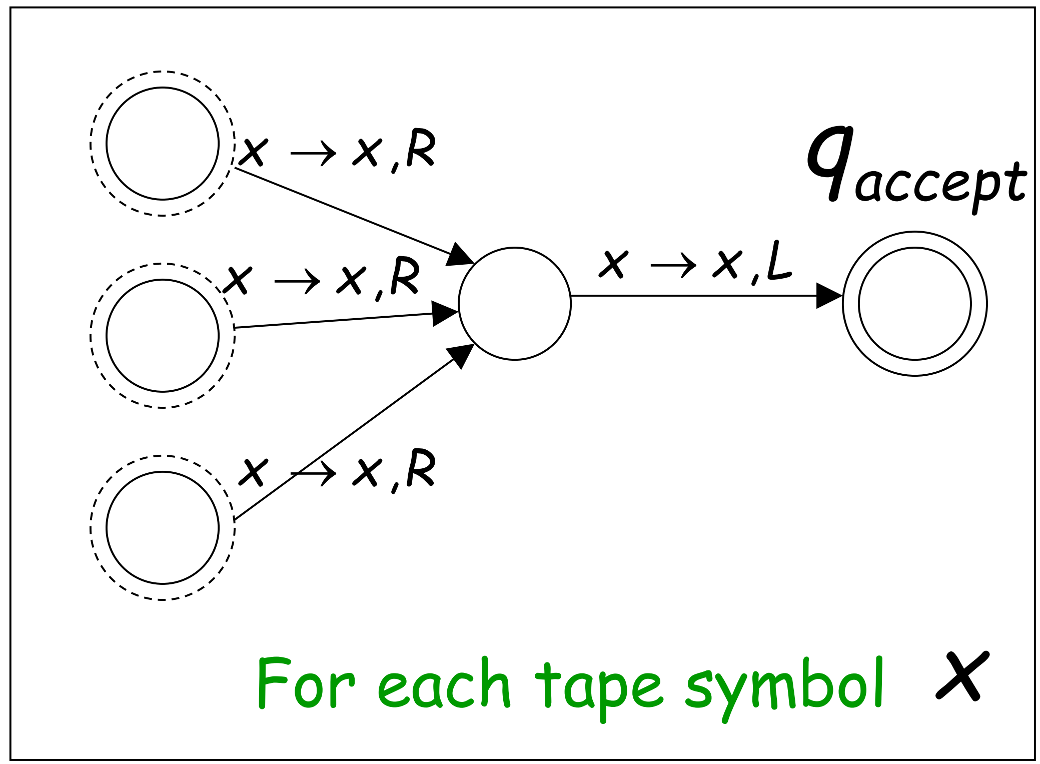
We can convert any Turing machine to have single accept and reject states

Old machine



Multiple accept states

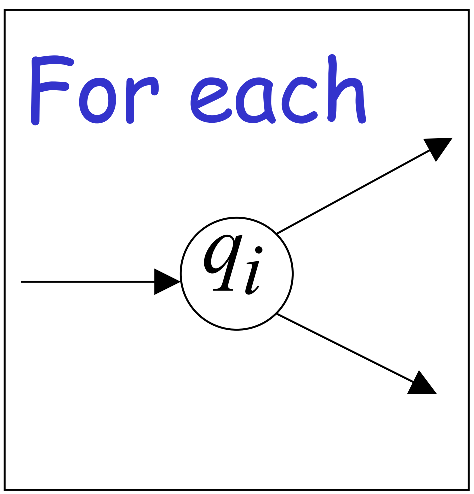
New machine



One accept state

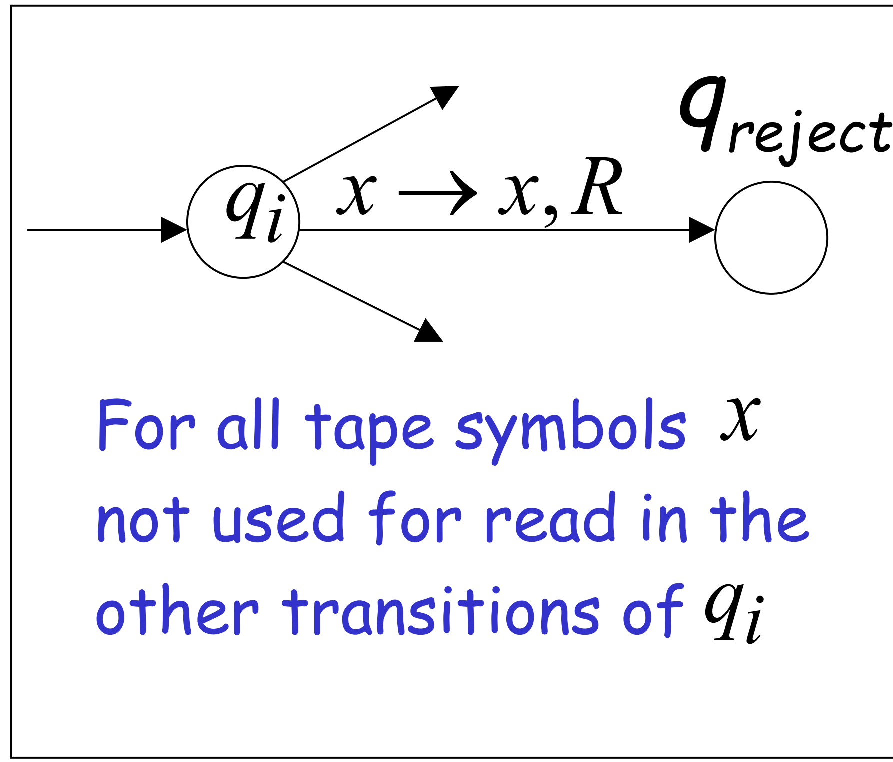
Do the following for each possible halting state:

Old machine



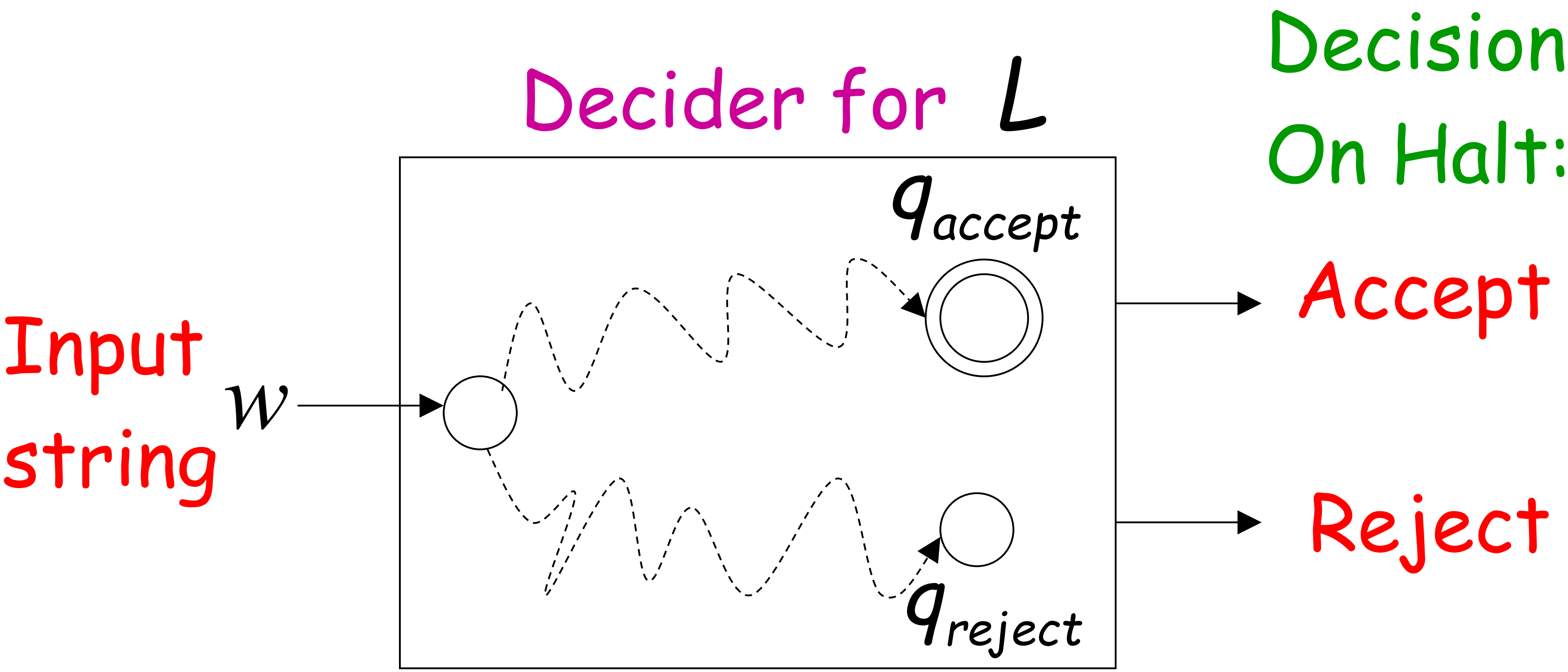
Multiple reject states

New machine



One reject state

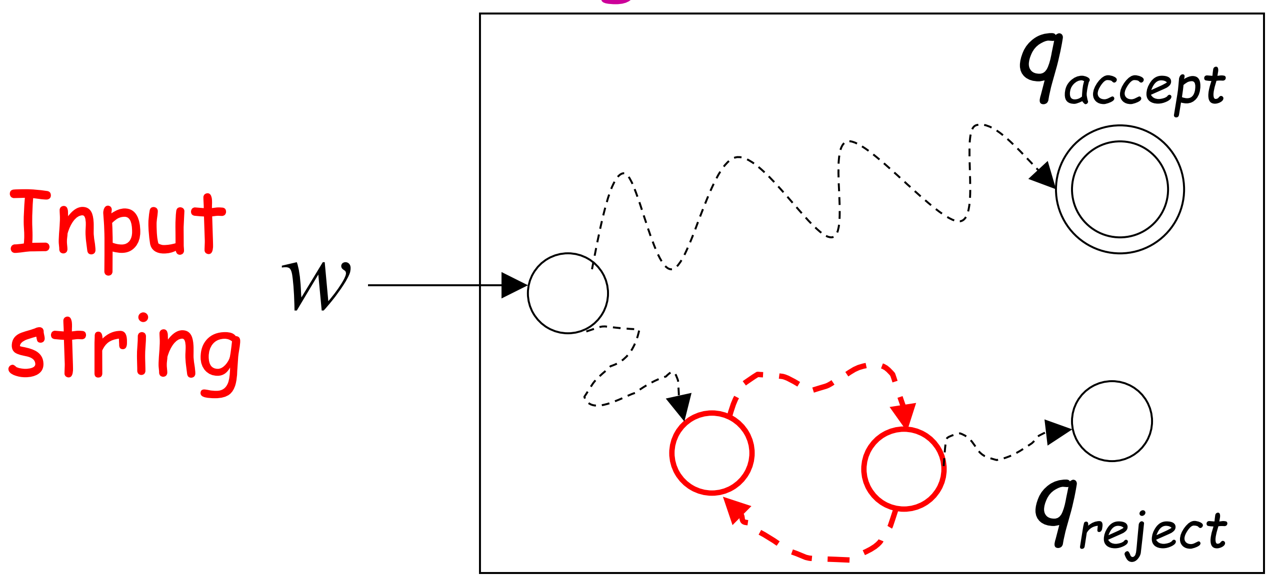
For a decidable language L:



For each input string, the computation halts in the accept or reject state

For a Turing-Acceptable language L :

Turing Machine for L



It is possible that for some input string the machine enters an infinite loop

A computational problem is decidable if the corresponding language is decidable

We also say that the problem is solvable

Problem: Is number x prime?

Corresponding language:

$$PRIMES = \{1, 2, 3, 5, 7, \ldots\}$$

We will show it is decidable

Decider for $PRIMES$:

On input number x :

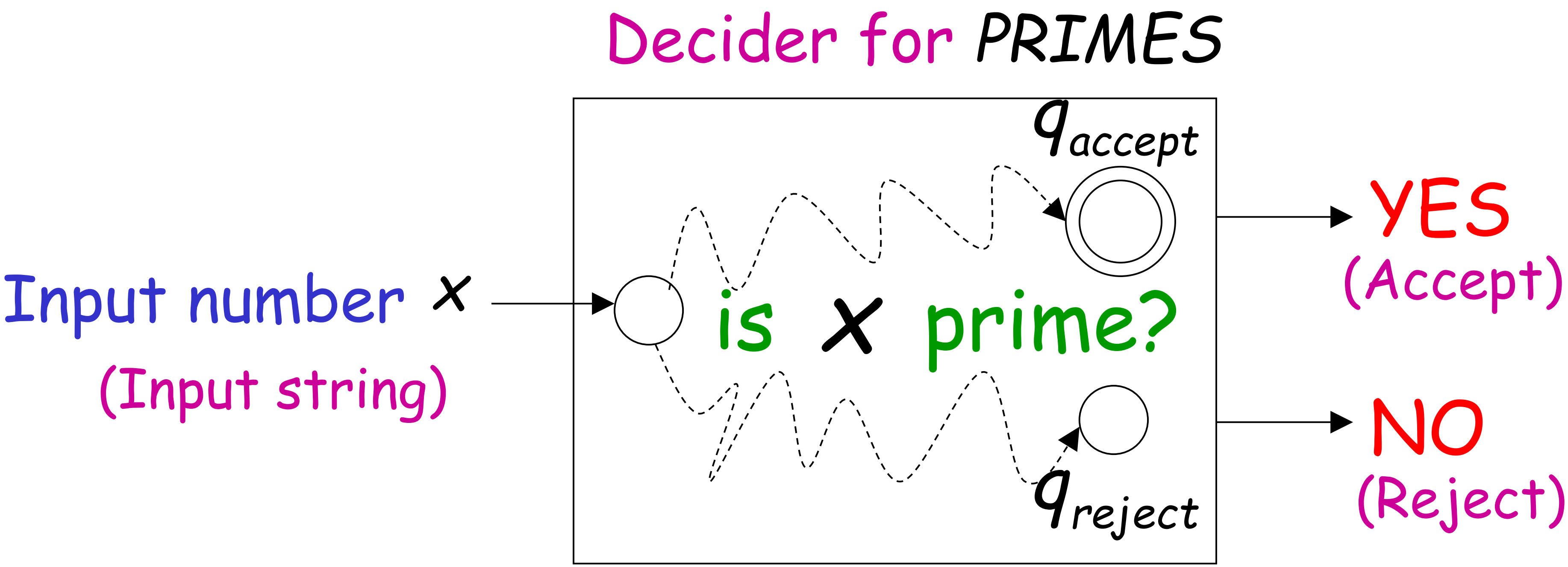
Divide x with all possible numbers between 2 and $\sqrt{x}$

If any of them divides x

Then reject

Else accept

the decider Turing machine can be designed based on the algorithm



Problem: Does DFA M accept the empty language $L(M) = \emptyset$?

Corresponding Language: (Decidable)

$EMPTY_{DFA} =$

$\{\langle M \rangle : M \text{ is a DFA that accepts empty language } \emptyset\}$

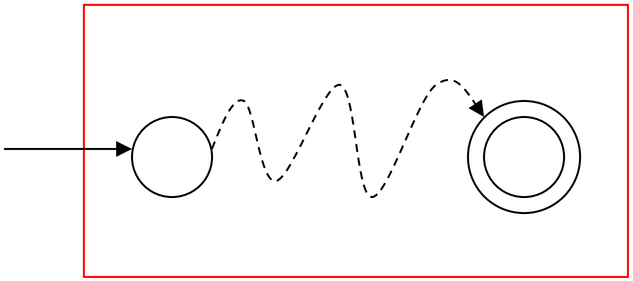
Description of DFA M as a string
(For example, we can represent M as a binary string, as we did for Turing machines)

Decider for $EMPTY_{DFA}$:

On input $\langle M \rangle$:

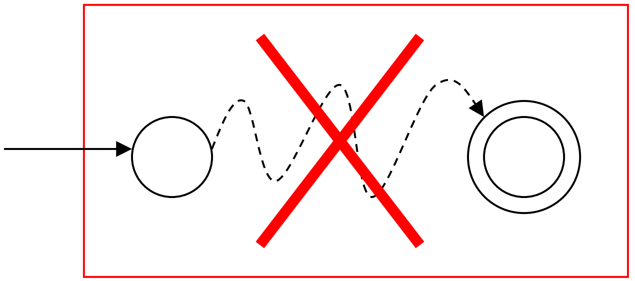
Determine whether there is a path from the initial state to any accepting state

DFA M



$L(M) \neq \emptyset$

DFA M



$L(M) = \emptyset$

Decision: Reject $\langle M \rangle$ Accept $\langle M \rangle$

Problem: Does DFA M accept a finite language?

Corresponding Language: (Decidable)

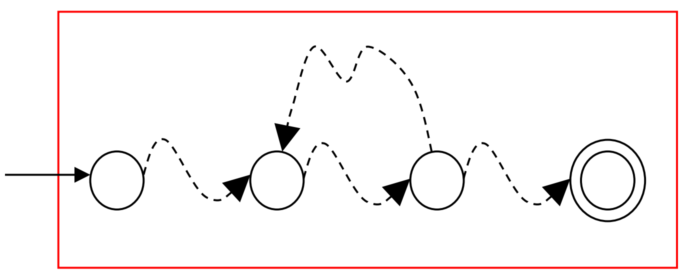
$$FINITE_{DFA} =$$
$$\{\langle M \rangle : M \text{ is a DFA that accepts a finite language}\}$$

Decider for $FINITE_{DFA}$:

On input $\langle M \rangle$:

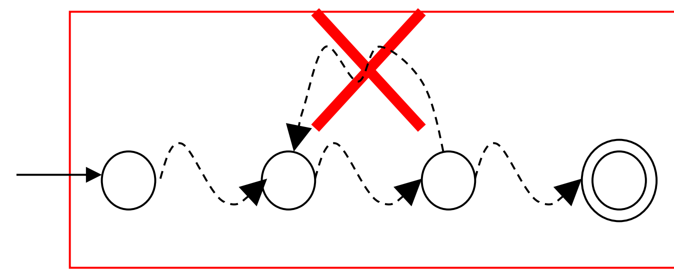
Check if there is a walk with a cycle from the initial state to an accepting state

DFA M



infinite

DFA M



finite

Decision: Reject $\langle M \rangle$ (NO)

Accept $\langle M \rangle$ (YES)

Problem: Does DFA M accept string w ?

Corresponding Language: (Decidable)

$$A_{DFA} =$$
$$\{\langle M, w \rangle : M \text{ is a DFA that accepts string } w\}$$

Decider for A_{DFA} :

On input string $\langle M, w \rangle$:

Run DFA M on input string w

If M accepts w

Then accept $\langle M, w \rangle$ (and halt)

Else reject $\langle M, w \rangle$ (and halt)

Problem: Do DFAs M_1 and M_2 accept the same language?

Corresponding Language: (Decidable)

$EQUAL_{DFA} =$

$\{\langle M_1, M_2\rangle : M_1$ and M_2 are DFAs that accept the same languages$\}$

Decider for $EQUAL_{DFA}$:

On input $\langle M_1, M_2 \rangle$:

Let L_1 be the language of DFA M_1

Let L_2 be the language of DFA M_2

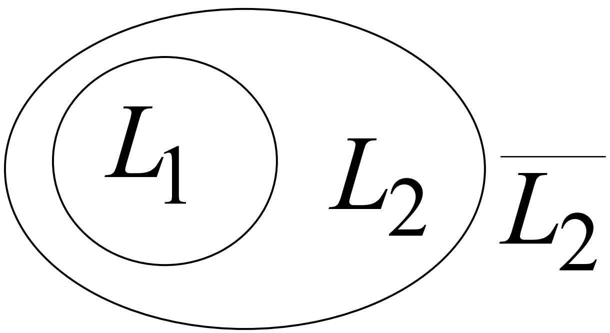
Construct DFA M such that:

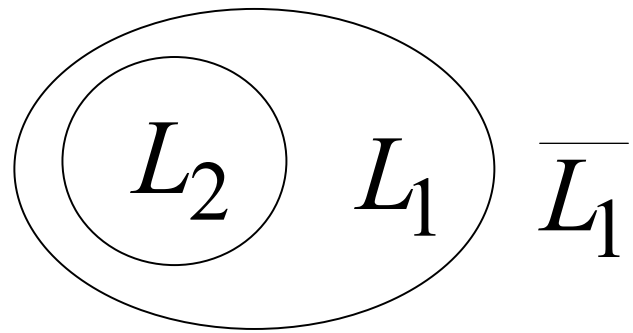
$$L(M) = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$$

(combination of DFAs)

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$$

$$L_1 \cap \overline{L_2} = \emptyset$$ and $$\overline{L_1} \cap L_2 = \emptyset$$



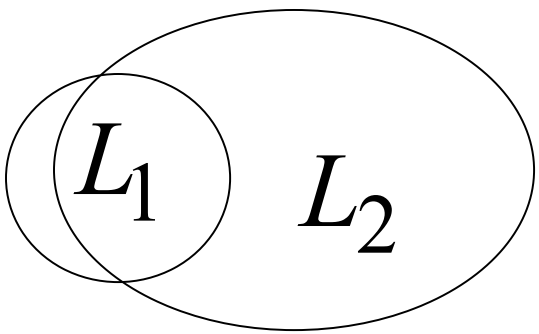


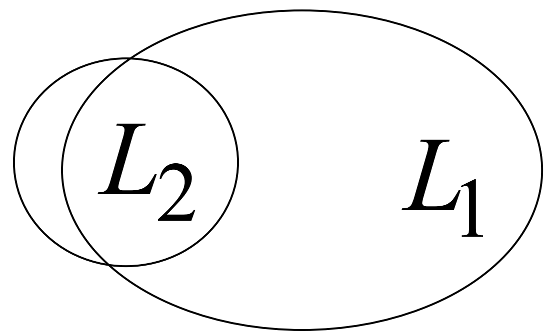
$$L_1 \subseteq L_2 \qquad L_2 \subseteq L_1$$

$$L_1 = L_2$$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) \neq \emptyset$$

$L_1 \cap \overline{L_2} \neq \emptyset$ or $\overline{L_1} \cap L_2 \neq \emptyset$





$L_1 \not\subset L_2$ $\qquad$ $L_2 \not\subset L_1$

$$L_1 \neq L_2$$

Therefore, we only need to determine whether

$$L(M) = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$$

which is a solvable problem for DFAs:

$EMPTY_{DFA}$

Theorem:

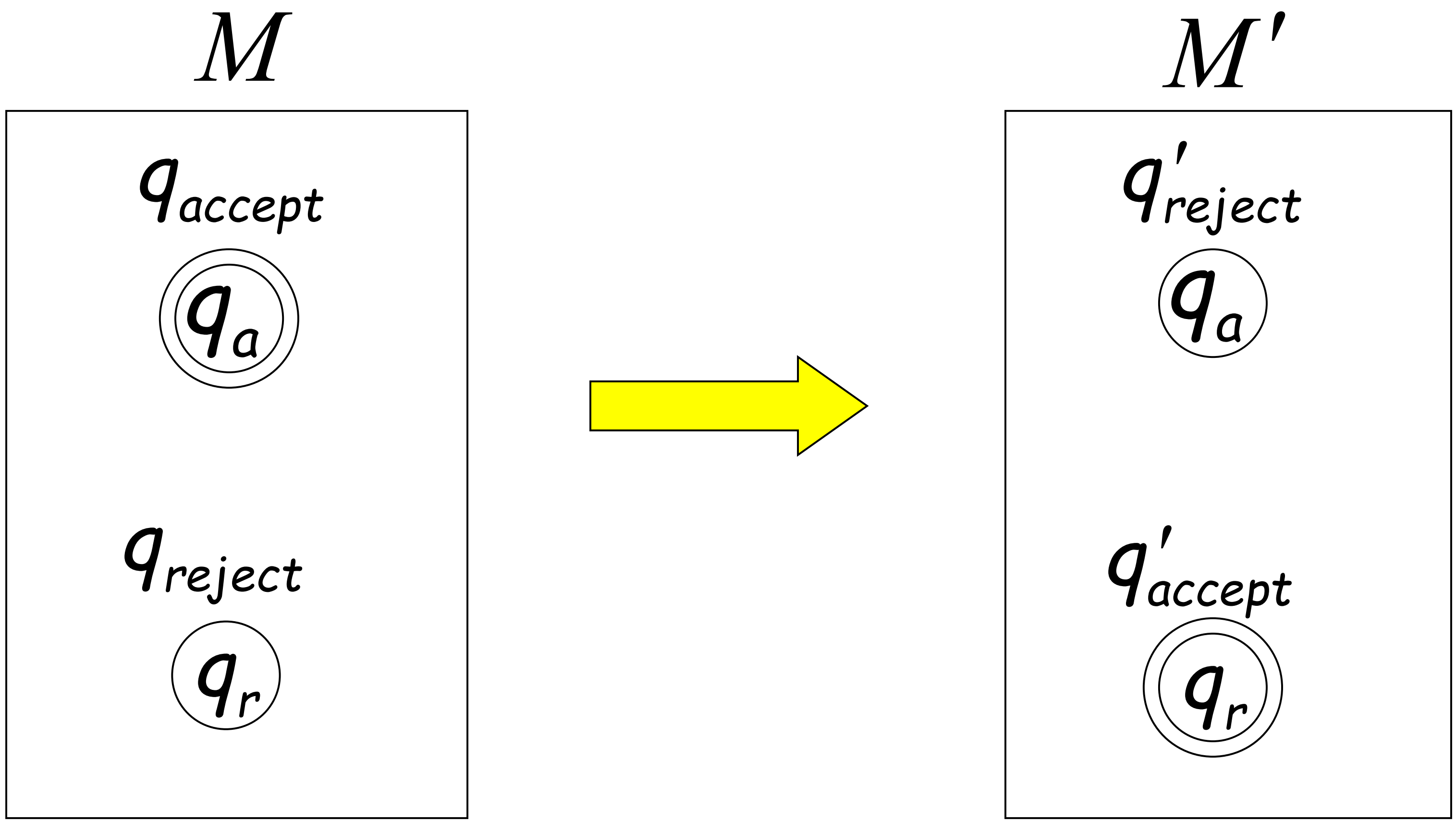
If a language L is decidable, then its complement $\overline{L}$ is decidable too

Proof:

Build a Turing machine M' that accepts $\overline{L}$ and halts on every input string

(M' is decider for $\overline{L}$)

Transform accept state to reject and vice-versa



Turing Machine M'

On each input string w do:

1. Let M be the decider for L

2. Run M with input string w

   If M accepts then reject

   If M rejects then accept

Accepts $\overline{L}$ and halts on every input string

END OF PROOF

Undecidable Languages

Undecidable Languages

An undecidable language has no decider:

Any Turing machine that accepts L does not halt on some input string

We will show that:

There is a language which is Turing-Acceptable and undecidable

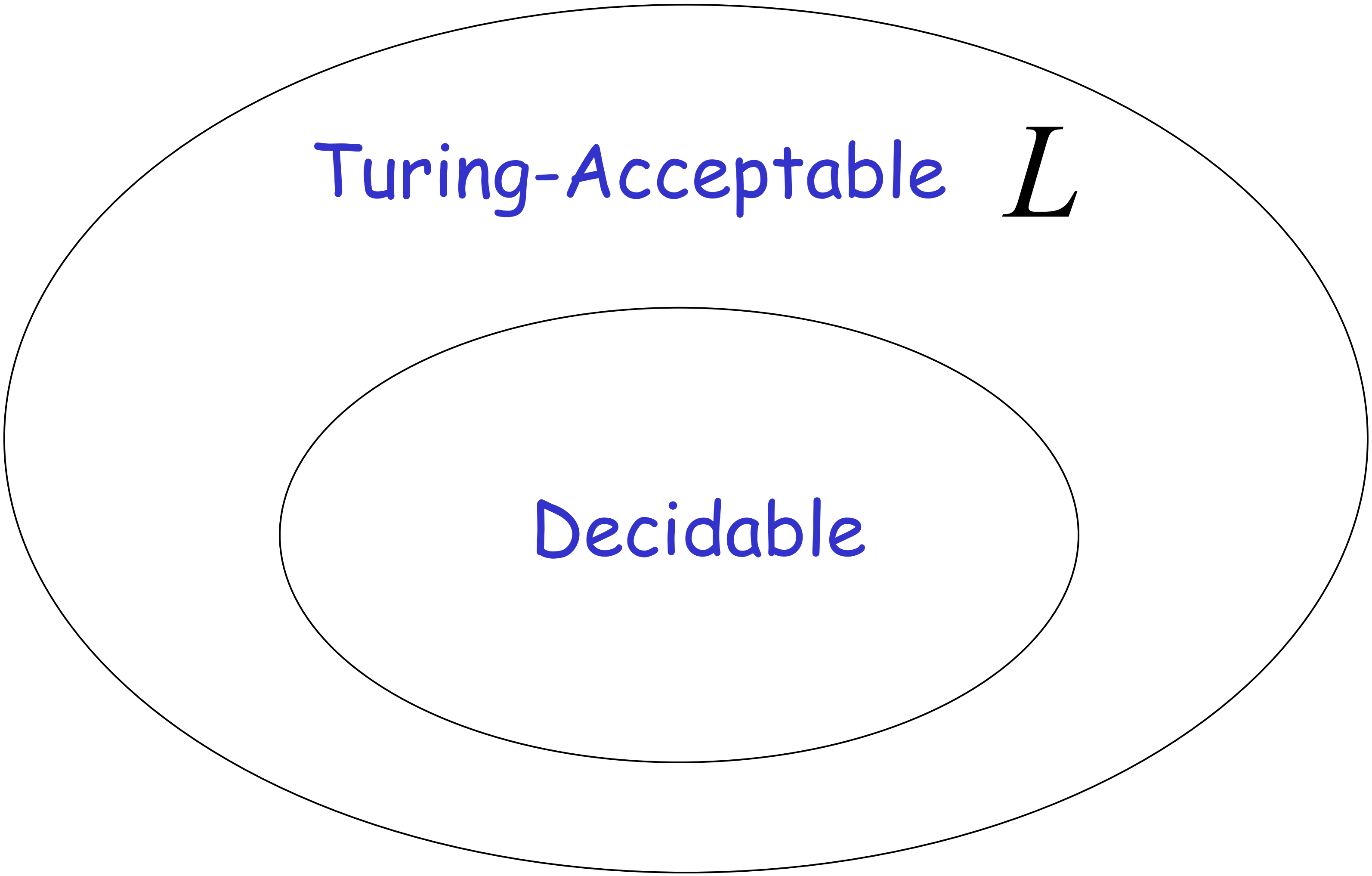
We will prove that there is a language L:

- L is Turing-acceptable
- $\overline{L}$ is **not** Turing-acceptable
  (not accepted by any Turing Machine)

the complement of a
decidable language is decidable

Therefore, L is undecidable

Non Turing-Acceptable $\overline{L}$



Consider alphabet $\{a\}$

Strings of $\{a\}^+$:

$$a, \ aa, \ aaa, \ aaaa, \ \ldots$$

$$a^1 \quad a^2 \quad a^3 \quad a^4 \quad \ldots$$

Consider Turing Machines
that accept languages over alphabet $\{a\}$

They are countable:

$$M_1,\ M_2,\ M_3,\ M_4,\ \ldots$$

(There is an enumerator that generates them)

Each machine accepts some language over $\{a\}$

$$M_1,\ M_2,\ M_3,\ M_4,\ \ldots$$

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

$$L(M_1),\ L(M_2),\ L(M_3),\ L(M_4),\ \ldots$$

Note that it is possible to have

$$L(M_i) = L(M_j) \quad \text{for} \quad i \neq j$$

Since, a language could be accepted by more than one Turing machine

Example language accepted by M_i

$$L(M_i) = \{aa, aaaa, aaaaaa\}$$

$$L(M_i) = \{a^2, a^4, a^6\}$$

Binary representation

	a^1	a^2	a^3	a^4	a^5	a^6	a^7	$\cdots$
$L(M_i)$	0	1	0	1	0	1	0	$\cdots$

Example of binary representations

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

Consider the language

$$L = \{a^i : a^i \in L(M_i)\}$$

L consists of the 1's in the diagonal

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$L = \{a^3, a^4, \ldots\}$$

Consider the language $\overline{L}$

$$\overline{L} = \{a^i : a^i \notin L(M_i)\}$$

$$L = \{a^i : a^i \in L(M_i)\}$$

$\overline{L}$ consists of the 0's in the diagonal

$$
\begin{array}{c|ccccc}
 & a^1 & a^2 & a^3 & a^4 & \cdots \\
\hline
L(M_1) & 0 & 1 & 0 & 1 & \cdots \\
L(M_2) & 1 & 0 & 0 & 1 & \cdots \\
L(M_3) & 0 & 1 & 1 & 1 & \cdots \\
L(M_4) & 0 & 0 & 0 & 1 & \cdots
\end{array}
$$

$$\overline{L} = \{a^1, a^2, \ldots\}$$

Theorem:

Language $\overline{L}$ is not Turing-Acceptable

Proof:

Assume for contradiction that

$\overline{L}$ is Turing-Acceptable

Let M_k be the Turing machine that accepts $\overline{L}$: $L(M_k) = \overline{L}$

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$L(M_k) = \overline{L} = 1100\cdots$$

Question: $M_k = M_1$?

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$L(M_k) = \overline{L} = 1100\cdots$$

$$a^1 \in L(M_k)$$

Answer: $M_k \neq M_1$

$$a^1 \notin L(M_1)$$

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$L(M_k) = \overline{L} = 1100\cdots$$

Question: $M_k = M_2$?

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$L(M_k) = \overline{L} = 1100\cdots$$

$$a^2 \in L(M_k)$$

Answer: $M_k \neq M_2$

$$a^2 \notin L(M_2)$$

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$L(M_k) = \overline{L} = 1100\cdots$$

Question: $M_k = M_3$?

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$L(M_k) = \overline{L} = 1100\cdots$$

$$a^3 \notin L(M_k)$$

Answer: $M_k \neq M_3$

$$a^3 \in L(M_3)$$

Similarly: $M_k \neq M_i$ for any i

Because either:

$a^i \in L(M_k)$ or $a^i \notin L(M_k)$

$a^i \notin L(M_i)$ $\quad$ $a^i \in L(M_i)$

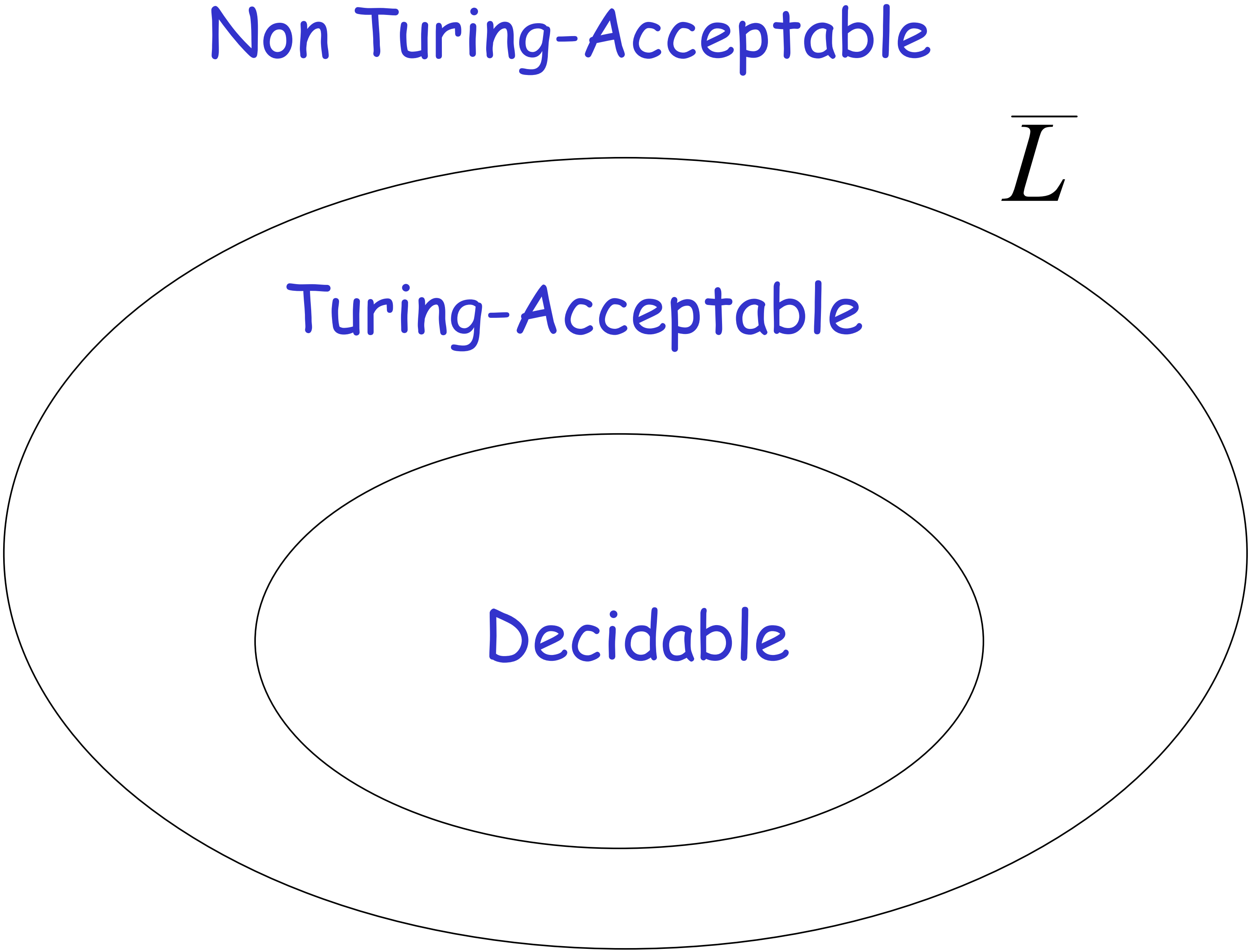
$\Downarrow$

the machine M_k cannot exist

$\Downarrow$

$\overline{L}$ is not Turing-Acceptable

End of Proof


Non Turing-Acceptable
$\overline{L}$
Turing-Acceptable
Decidable

We will prove that the language

$$L = \{a^i : a^i \in L(M_i)\}$$

Is Turing-Acceptable

Undecidable

There is a Turing machine that accepts L

Each machine that accepts L doesn't halt on some input string

Theorem: The language

$$L = \{a^i : a^i \in L(M_i)\}$$

Is Turing-Acceptable

Proof: We will give a Turing Machine that accepts L

Turing Machine that accepts L

For any input string w

- Suppose $w = a^i$

- Find Turing machine M_i

  (using the enumerator for Turing Machines)

- Simulate M_i on input string a^i

- If M_i accepts, then accept w

End of Proof

Therefore:

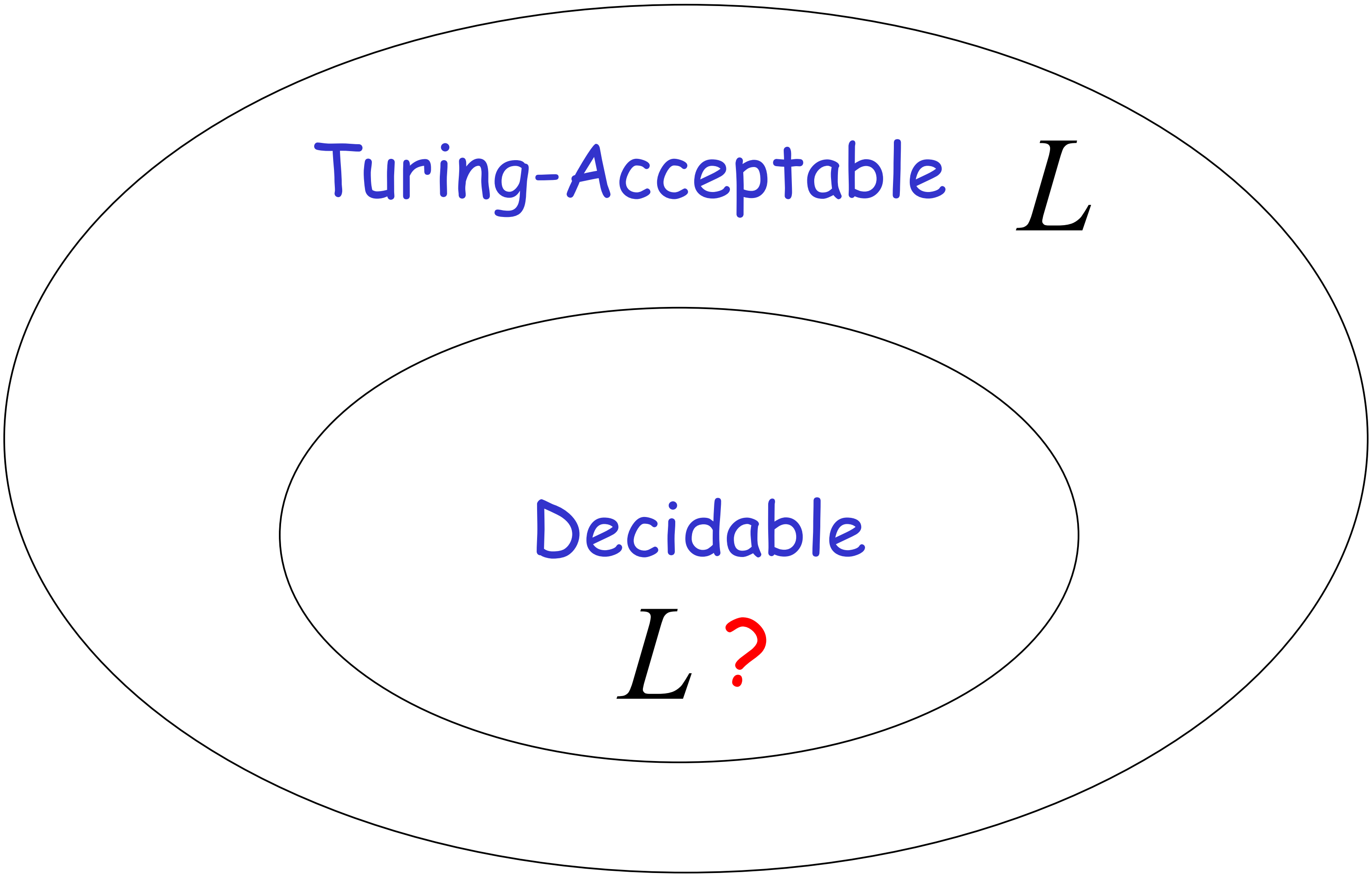
Turing-Acceptable

$$L = \{a^i : a^i \in L(M_i)\}$$

Not Turing-acceptable

$$\overline{L} = \{a^i : a^i \notin L(M_i)\}$$

Non Turing-Acceptable $\overline{L}$



Theorem: $L = \{a^i : a^i \in L(M_i)\}$

is undecidable

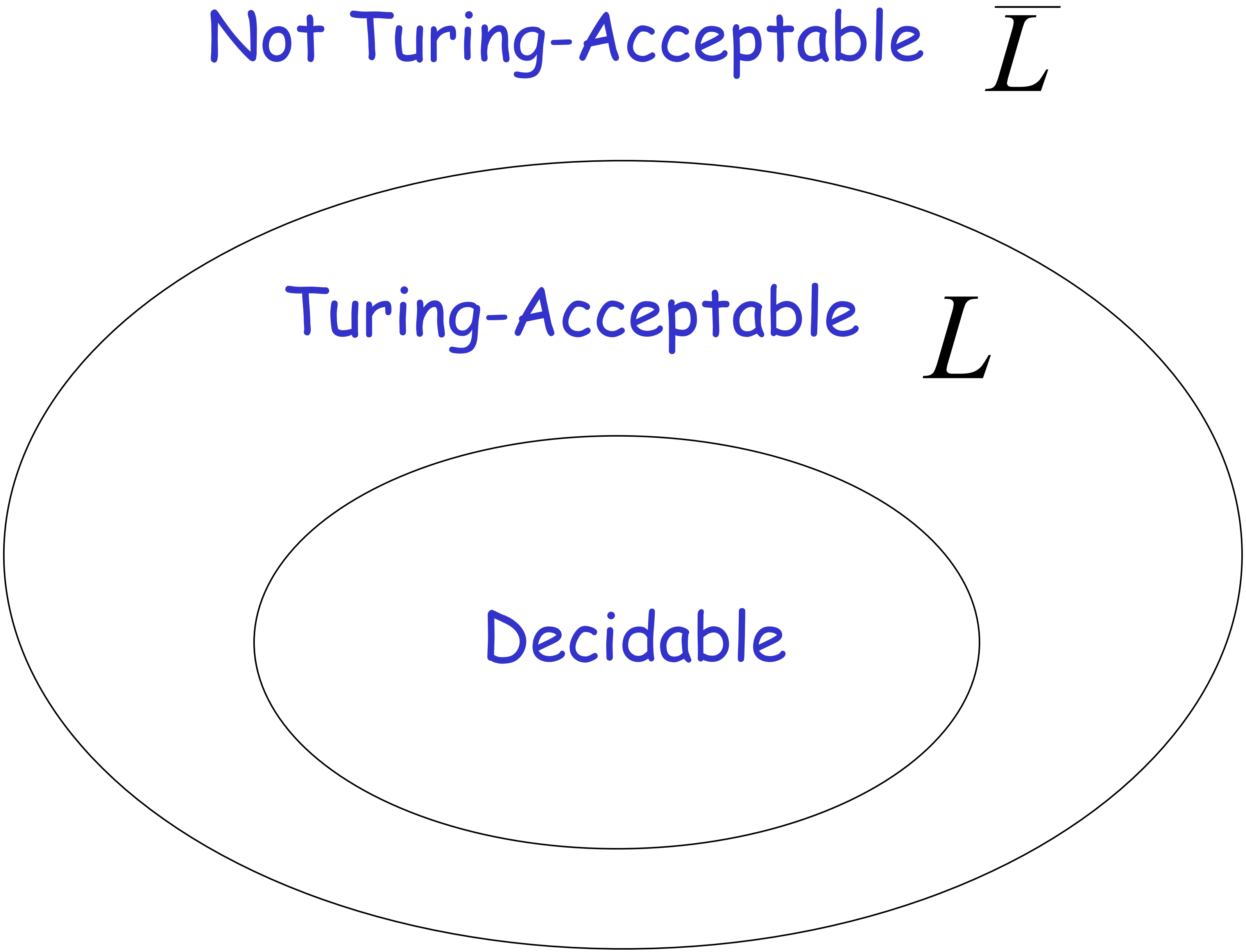
Proof: If L is decidable

the complement of a decidable language is decidable

Then $\overline{L}$ is decidable

However, $\overline{L}$ is not Turing-Acceptable!

Contradiction!!!!


Not Turing-Acceptable $\overline{L}$
Turing-Acceptable L
Decidable