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

For any 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*

For any string w :

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

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

Every decidable language is Turing-Acceptable

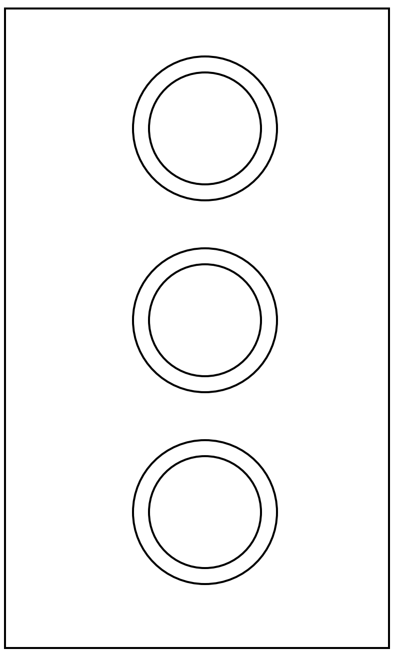
Sometimes, it is convenient to have Turing machines with single accept and reject states

These are the only halting states

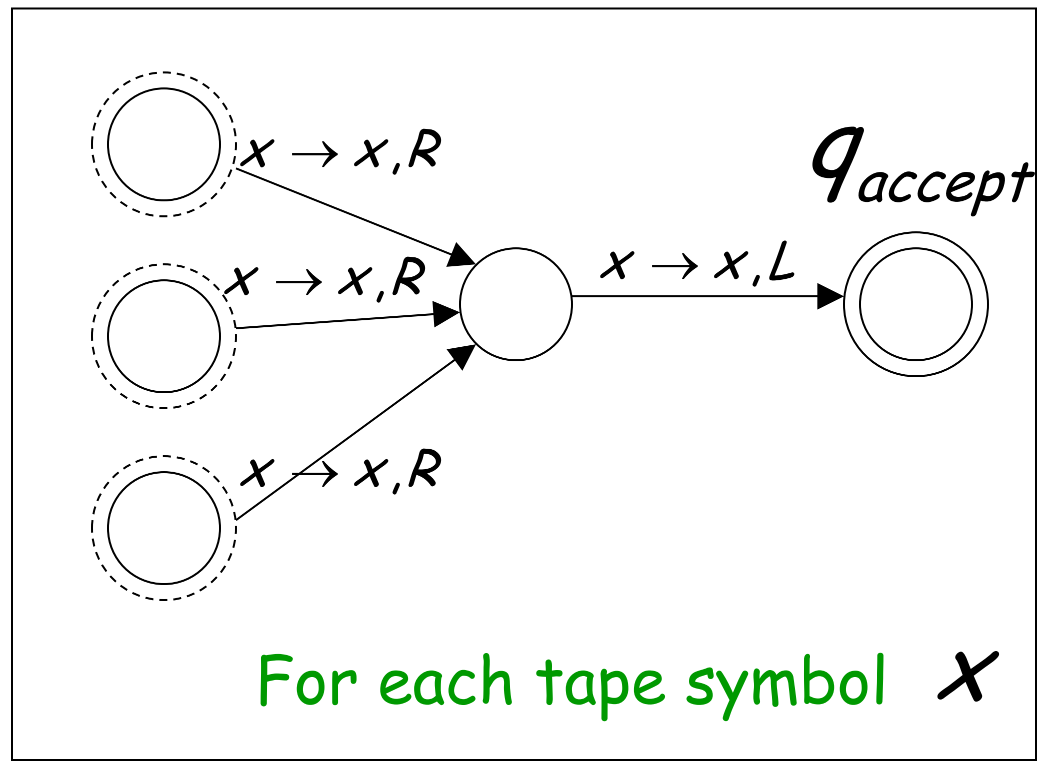
That result to possible halting configurations

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

Old machine



New machine



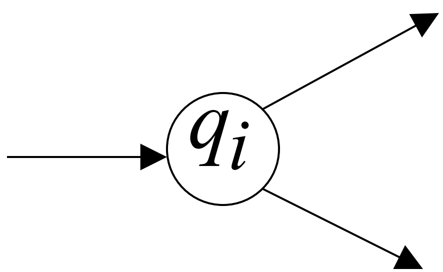
Multiple accept states

One accept state

Do the following for each possible halting state:

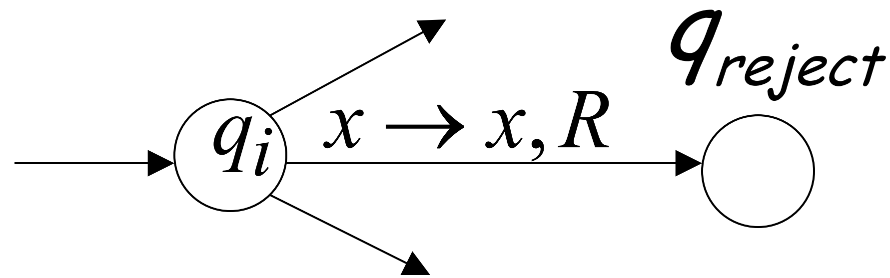
Old machine

For each



Multiple reject states

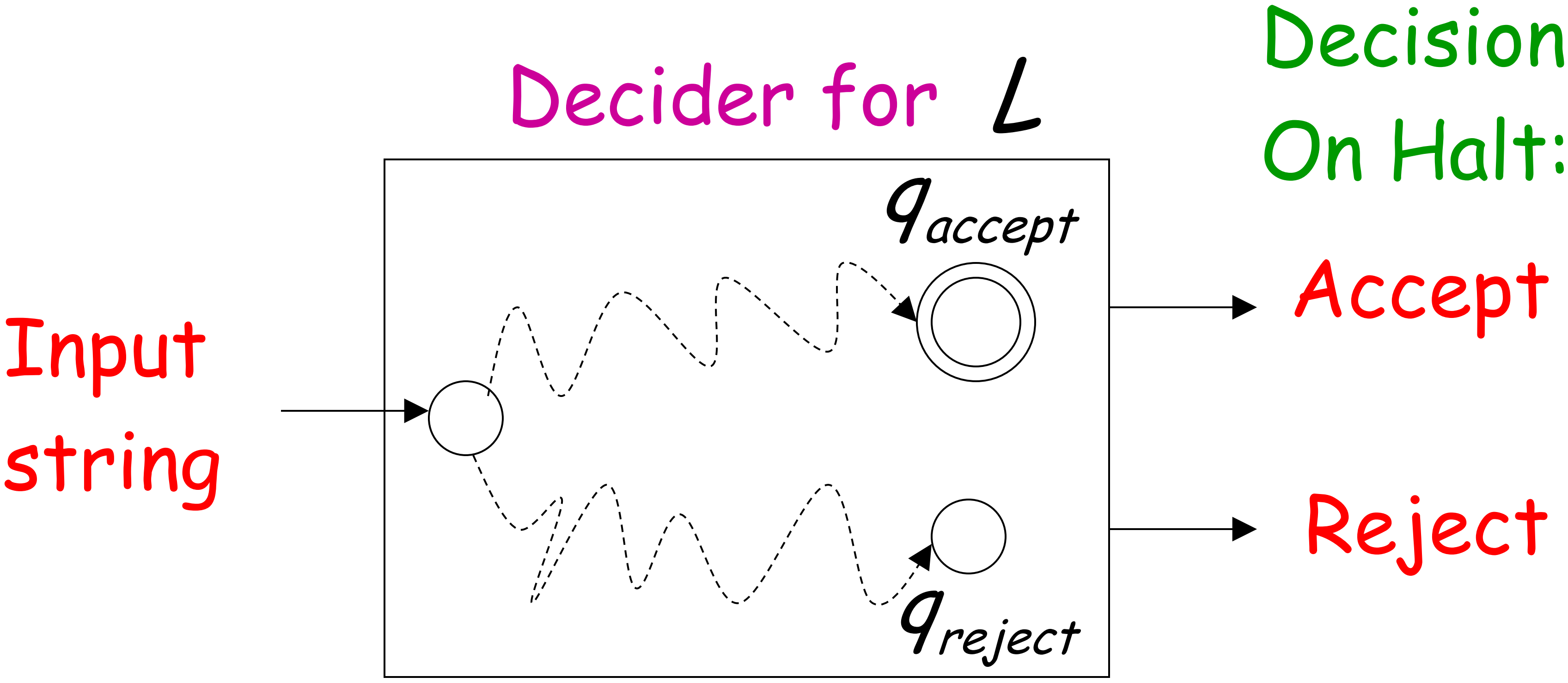
New machine



For all tape symbols x not used for read in the other transitions of q_i

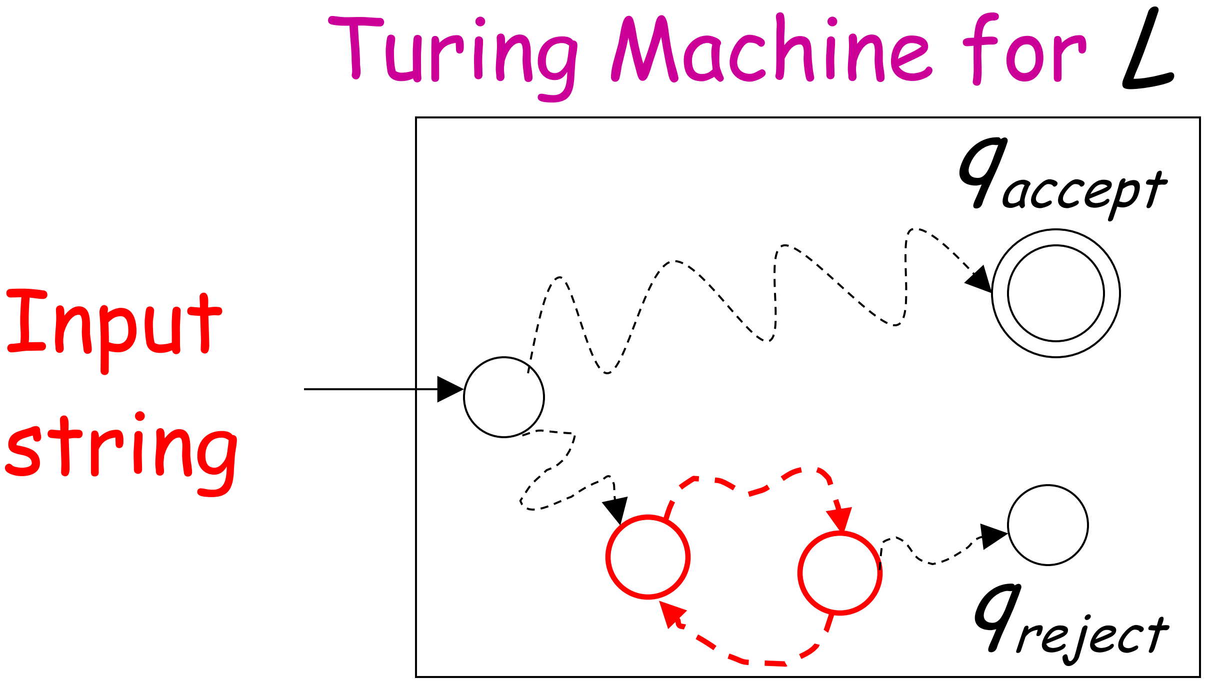
One reject state

For a decidable language L:



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

For a Turing-Acceptable language L :



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

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 :

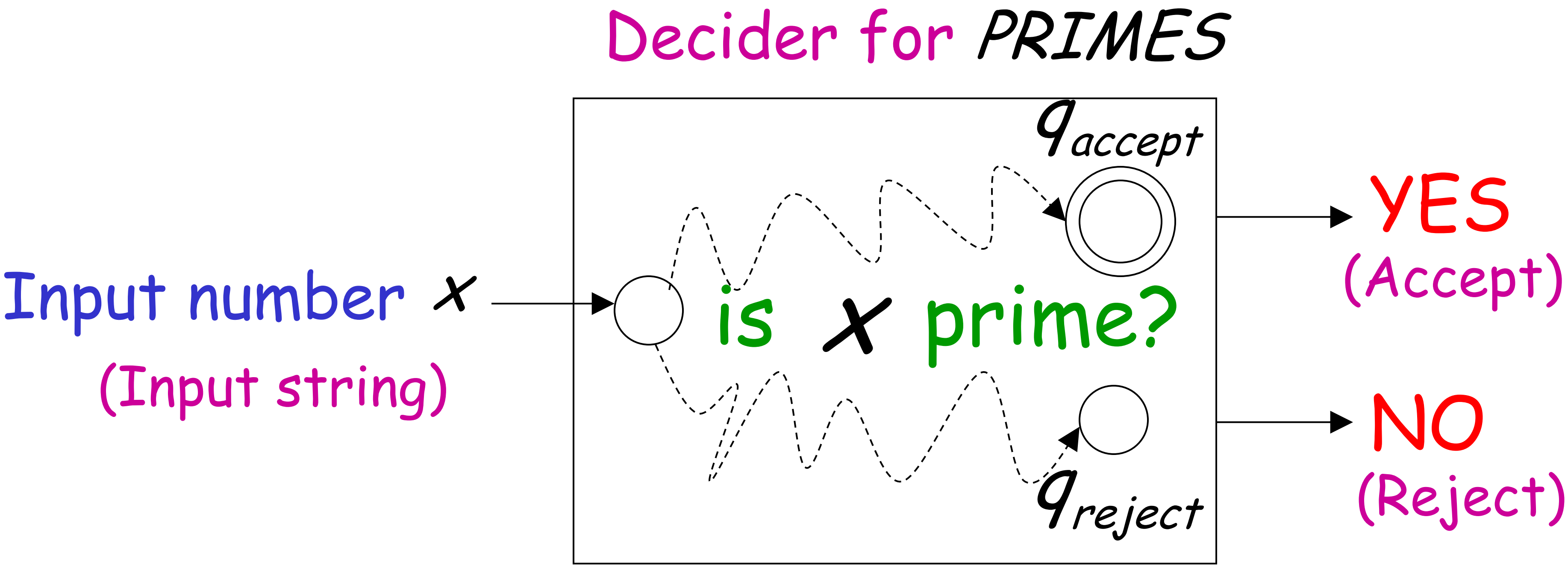
Divide x with all possible numbers between 2 and $\sqrt{x}$

If any of them divides x

Then reject

Else accept

the decider for the language
solves the corresponding problem



Theorem:

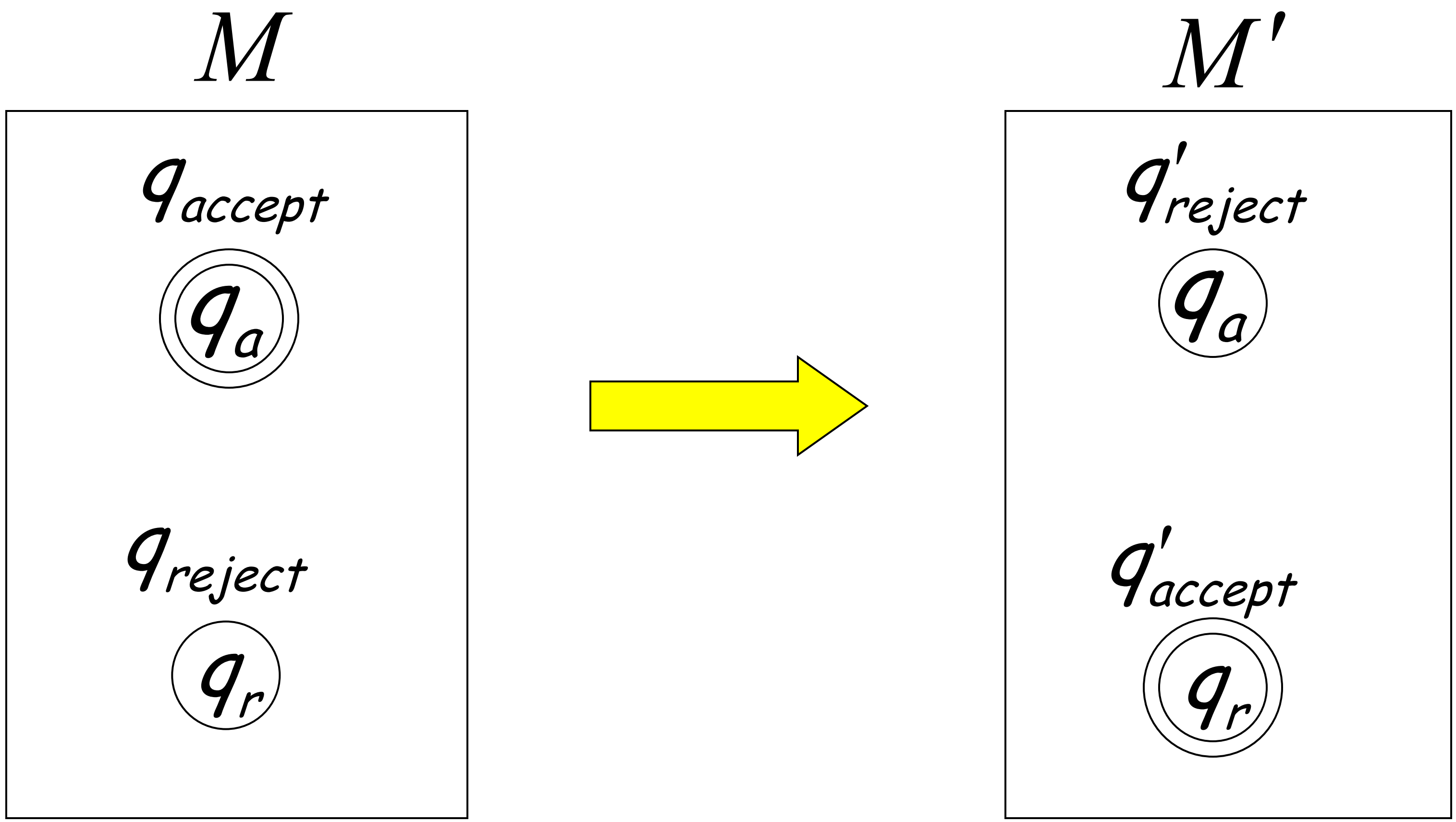
If a language L is decidable, then its complement $\overline{L}$ is decidable too

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

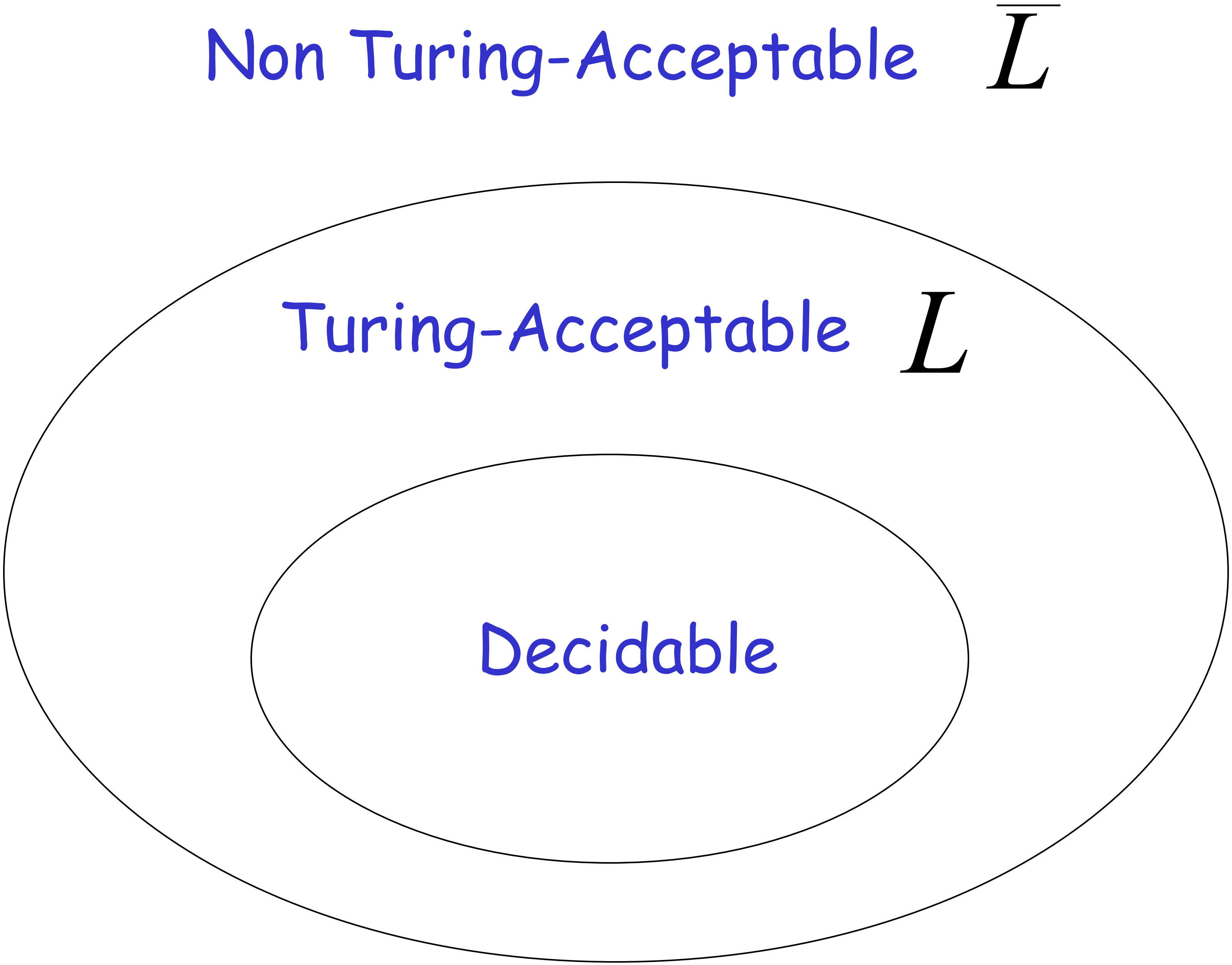
END OF PROOF

Undecidable Languages

An undecidable language has no decider:

each Turing machine that accepts L

does not halt on some input string


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