T065001: Introduction to Formal Languages

Lecture 13: Reducibility (2)

Chapters 5.2 - 5.3 in Sipser's textbook

2025-07-14

(Lecture slides by Yih-Kuen Tsay)

(From Chapter 5.1)

- A *reduction* is a way of converting one problem into another problem in such a way that a solution to the second problem can be used to solve the first problem.
- If a problem A reduces (is reducible) to another problem B, we can use a solution to B to solve A.
- *Reducibility* says nothing about solving A or B alone, but only about the solvability of A in the presence of a solution to B.
- Reducibility is the primary method for proving that problems are computationally unsolvable.
- Suppose that A is reducible to B. If B is decidable, then A is decidable; equivalently, if A is undecidable, then B is undecidable.

The Post Correspondence Problem

- Consider a collection of dominos such as:

$$\left\{ \left[\frac{b}{ca} \right], \left[\frac{a}{ab} \right], \left[\frac{ca}{a} \right], \left[\frac{abc}{c} \right] \right\}$$

The Post Correspondence Problem

- Consider a collection of dominos such as:

$$\left\{ \left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right] \right\}$$

- A *match* is a list of these dominos (repetitions permitted) where the string of symbols on the top is the same as that on the bottom. Below is a match:

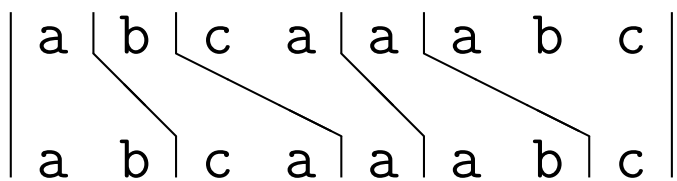
The Post Correspondence Problem

- Consider a collection of dominos such as:

$$\left\{ \left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right] \right\}$$

- A *match* is a list of these dominos (repetitions permitted) where the string of symbols on the top is the same as that on the bottom. Below is a match:

$$\left[\frac{a}{ab}\right] \left[\frac{b}{ca}\right] \left[\frac{ca}{a}\right] \left[\frac{a}{ab}\right] \left[\frac{abc}{c}\right]$$



The Post Correspondence Problem (cont.)

- The Post correspondence problem (PCP) is to determine whether a collection of dominos has a match.
- More formally, an instance of the PCP is a collection of dominos:

$$P = \left\{ \left[\frac{t_1}{b_1} \right], \left[\frac{t_2}{b_2} \right], \cdots, \left[\frac{t_k}{b_k} \right] \right\}$$

- A *match* is a sequence $i_1, i_2, \cdots, i_l$ such that $t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$.
- $PCP = \{\langle P \rangle \mid P$ is an instance of the Post correspondence problem with a match$\}$.

Undecidability of the PCP

From Chapter 4:

- $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$.

Theorem (4.11)

A_{TM} is undecidable.

Undecidability of the PCP

From Chapter 4:

- $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$.

Theorem (4.11)

A_{TM} is undecidable.

We will now use Theorem 4.11 to prove:

Theorem (5.15)

PCP is undecidable

Undecidability of the PCP

From Chapter 4:

- $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$.

Theorem (4.11)

A_{TM} is undecidable.

We will now use Theorem 4.11 to prove:

Theorem (5.15)

PCP is undecidable

- The proof is by reduction from A_{TM} via accepting computation histories.

(From Chapter 3.1)

- As a TM computes, changes occur in
  1. the current state,
  2. the current tape contents, and
  3. the current head location.
- A setting of these three items is called a **configuration** of the TM.
- We write uqv to denote the configuration where
  1. the current state is q,
  2. the current tape contents is uv, and
  3. the current head location is the first symbol of v.

  (The tape contains only blanks following the last symbol of v.)

(From Chapter 3.1)

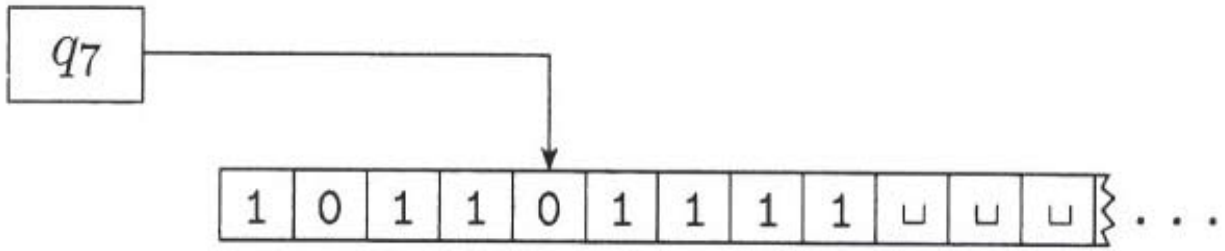


FIGURE **3.4**
A Turing machine with configuration $1011q_701111$

(From Chapter 5.1)

Definition (5.5)

An *accepting computation history* for M on w is a sequence of configurations $C_1, C_2, \cdots, C_l$, where

1. C_1 is the start configuration,
2. C_l is an accepting configuration, and
3. C_i yields C_{i+1}, $1 \leq i \leq l-1$.

Note that any accepting computation history is a *finite sequence*.
$\Rightarrow$ We can represent accepting computation histories by *strings*.

Undecidability of the PCP (cont.)

Theorem (5.15)

PCP is undecidable

Undecidability of the PCP (cont.)

Theorem (5.15)

PCP is undecidable

- Assume that a TM R decides PCP.

  We will show that then there would exist a TM S that could decide A_{TM}, which is impossible.

  Main idea of the proof:
  For any instance $\langle M, w \rangle$ of A_{TM}, let S construct an instance P of PCP in which a match is an accepting computation history for M on w.

  To do so, we need to define the dominos in P carefully so that making a match forces a simulation of M on w to occur.

Undecidability of the PCP (cont.)

The set P consists of various types of dominos that link positions in one configuration with the next configuration.

The next slides describe how to define P for any given $\langle M, w \rangle$.

For now, assume that a match always begins with $\left[\frac{t_1}{b_1}\right]$.

(We'll see how to enforce this constraint later.)

Undecidability of the PCP (cont.)

1. Add $\left[\frac{\#}{\#q_0 w_1 w_2 \cdots w_n \#}\right]$ as $\left[\frac{t_1}{b_1}\right]$.
2. For every $a, b \in \Gamma$ and every $q, r \in Q$ where $q \neq q_{\text{reject}}$,

   $$\text{if } \delta(q, a) = (r, b, R), \text{ add } \left[\frac{qa}{br}\right].$$

3. For every $a, b, c \in \Gamma$ and every $q, r \in Q$ where $q \neq q_{\text{reject}}$,

   $$\text{if } \delta(q, a) = (r, b, L), \text{ add } \left[\frac{cqa}{rcb}\right].$$

4. For every $a \in \Gamma$, add $\left[\frac{a}{a}\right]$.
5. Add $\left[\frac{\#}{\#}\right]$ and $\left[\frac{\#}{\sqcup\#}\right]$.

Undecidability of the PCP (cont.)

Example: Let $\Gamma = \{0, 1, 2, \sqcup\}$ and $w = 0100$.

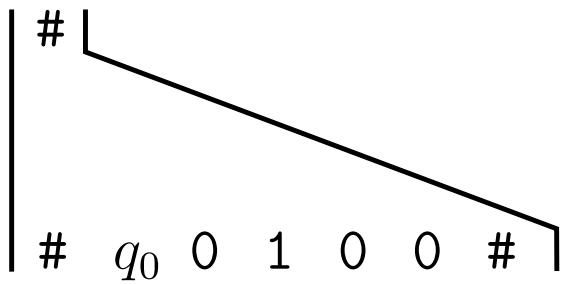
Undecidability of the PCP (cont.)

Example: Let $\Gamma = \{0, 1, 2, \sqcup\}$ and $w = 0100$.

Part 1 places the domino

$$\left[\frac{\#}{\#q_0 0100\#}\right] = \left[\frac{t_1}{b_1}\right]$$

in P, and the match begins



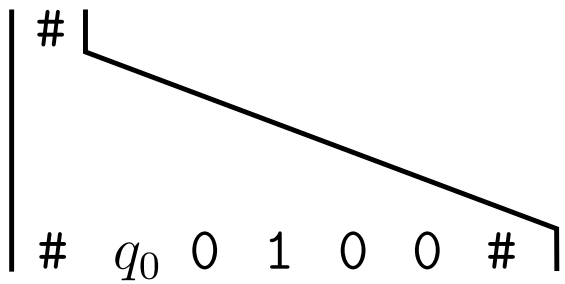
Undecidability of the PCP (cont.)

Example: Let $\Gamma = \{0, 1, 2, \sqcup\}$ and $w = 0100$.

Part 1 places the domino

$$\left[\frac{\#}{\#q_0 0100\#}\right] = \left[\frac{t_1}{b_1}\right]$$

in P, and the match begins



$\Rightarrow$ The bottom row contains the start configuration for M on w.

Undecidability of the PCP (cont.)

Example: (continued)
Next, suppose $\delta(q_0, 0) = (q_7, 2, \mathrm{R})$. Then parts 2 & 4 place the dominos

$$\left[\frac{q_0 0}{2 q_7}\right]$$

and

$$\left[\frac{0}{0}\right], \left[\frac{1}{1}\right], \left[\frac{2}{2}\right], \text{ and } \left[\frac{\sqcup}{\sqcup}\right]$$

in P. Together with part 5, that allows us to extend the match to:

Undecidability of the PCP (cont.)

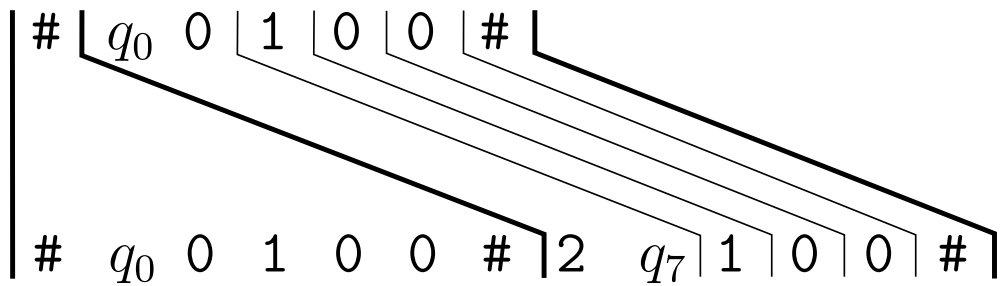
Example: (continued)
Next, suppose $\delta(q_0, 0) = (q_7, 2, \mathrm{R})$. Then parts 2 & 4 place the dominos

$$\left[\frac{q_0 0}{2 q_7}\right]$$

and

$$\left[\frac{0}{0}\right], \left[\frac{1}{1}\right], \left[\frac{2}{2}\right], \text{ and } \left[\frac{\sqcup}{\sqcup}\right]$$

in P. Together with part 5, that allows us to extend the match to:



$\Rightarrow$ The bottom row contains the second configuration after the first one.

Undecidability of the PCP (cont.)

Example: (continued)
Furthermore, suppose that $\delta(q_7, 1) = (q_5, 0, \mathrm{R})$ so that we get

$$\left[\frac{q_7 1}{0 q_5}\right] \text{ in } P.$$

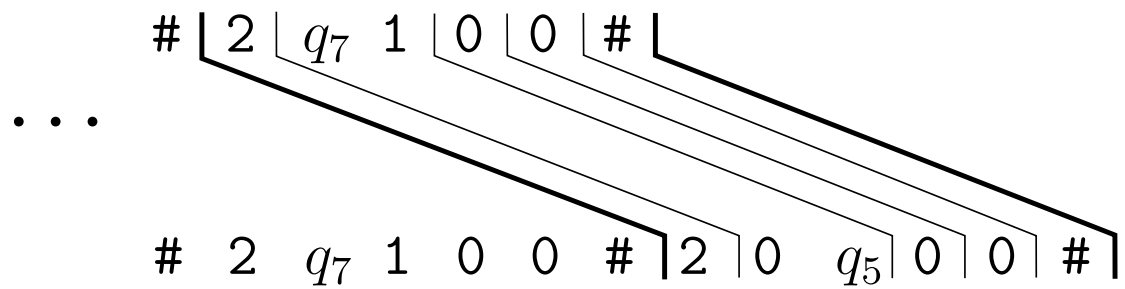
Then the latest partial match extends to:

Undecidability of the PCP (cont.)

Example: (continued)
Furthermore, suppose that $\delta(q_7, 1) = (q_5, 0, \mathrm{R})$ so that we get

$$\left[\frac{q_7 1}{0 q_5}\right] \text{ in } P.$$

Then the latest partial match extends to:



$\Rightarrow$ The bottom row contains the third configuration after the second one.

Undecidability of the PCP (cont.)

Example: (continued)

Also, suppose that $\delta(q_5, 0) = (q_9, 2, \mathsf{L})$. Then part 3 gives the dominos

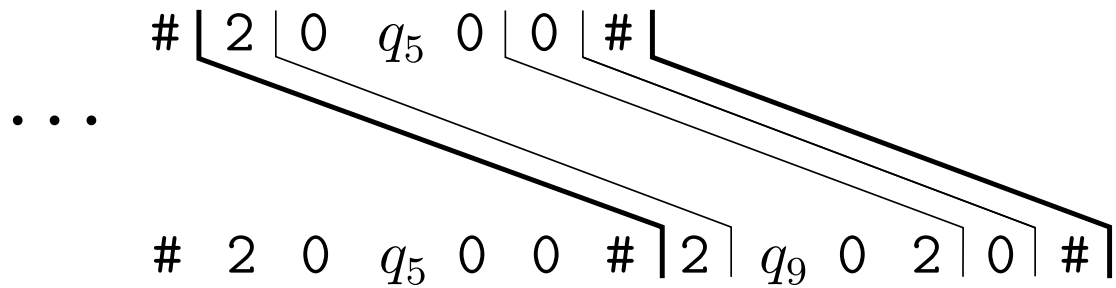
$$\left[\frac{0q_50}{q_902}\right], \left[\frac{1q_50}{q_912}\right], \left[\frac{2q_50}{q_922}\right], \text{ and } \left[\frac{\sqcup q_50}{q_9\sqcup 2}\right].$$

Undecidability of the PCP (cont.)

Example: (continued)
Also, suppose that $\delta(q_5, 0) = (q_9, 2, \mathrm{L})$. Then part 3 gives the dominos

$$\left[\frac{0q_50}{q_902}\right], \left[\frac{1q_50}{q_912}\right], \left[\frac{2q_50}{q_922}\right], \text{ and } \left[\frac{\sqcup q_50}{q_9\sqcup 2}\right].$$

Using the first domino above, the partial match from before is extended to:



$\Rightarrow$ The bottom row contains the fourth configuration after the third one.

Undecidability of the PCP (cont.)

Note that as we construct a match, we are forced to simulate M on w.

The process continues until M reaches a halting state.

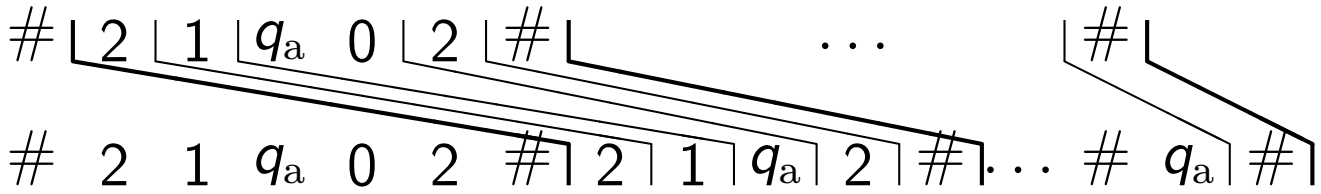
However, at that point, we're not yet done...

If M's accept state is reached, we want the top of the partial match to catch up with the bottom to make the match complete.

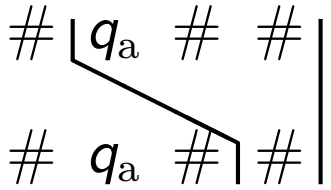
This can be accomplished by adding some additional dominos that "eat" symbols adjacent to q_{accept}, as described next.

Undecidability of the PCP (cont.)

6. For every $a \in \Gamma$, add $\left[\frac{aq_{\text{accept}}}{q_{\text{accept}}}\right]$ and $\left[\frac{q_{\text{accept}}a}{q_{\text{accept}}}\right]$.

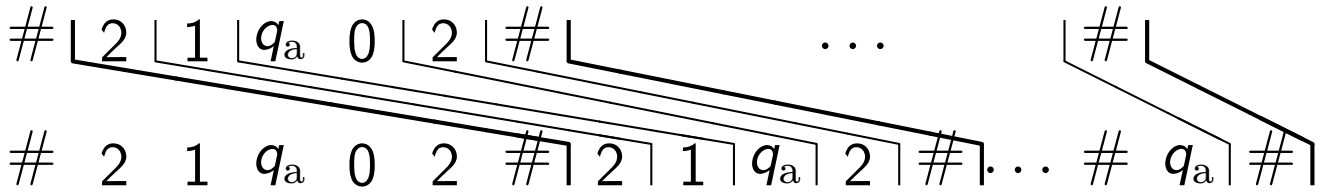


7. Add $\left[\frac{q_{\text{accept}}\#\#}{\#}\right]$.

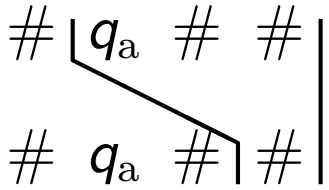


Undecidability of the PCP (cont.)

6. For every $a \in \Gamma$, add $\left[\frac{aq_{\text{accept}}}{q_{\text{accept}}}\right]$ and $\left[\frac{q_{\text{accept}}a}{q_{\text{accept}}}\right]$.



7. Add $\left[\frac{q_{\text{accept}}\#\#}{\#}\right]$.



But how do we force matches to always begin with the domino $\left[\frac{t_1}{b_1}\right]$?

Undecidability of the PCP (cont.)

To ensure that a match starts with $\left[\frac{t_1}{b_1}\right]$,

S converts the collection $\left\{\left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \cdots, \left[\frac{t_k}{b_k}\right]\right\}$ to

$$\left\{\left[\frac{\star t_1}{\star b_1 \star}\right], \left[\frac{\star t_1}{b_1 \star}\right], \left[\frac{\star t_2}{b_2 \star}\right], \cdots, \left[\frac{\star t_k}{b_k \star}\right], \left[\frac{* \diamond}{\diamond}\right]\right\}$$

where

$$\begin{aligned} \star u &= *u_1 * u_2 * u_3 * \cdots * u_n \\ u\star &= u_1 * u_2 * u_3 * \cdots * u_n * \\ \star u \star &= *u_1 * u_2 * u_3 * \cdots * u_n * \end{aligned} .$$

Undecidability of the PCP (cont.)

To ensure that a match starts with $\left[\frac{t_1}{b_1}\right]$,

S converts the collection $\left\{\left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \cdots, \left[\frac{t_k}{b_k}\right]\right\}$ to

$$\left\{\left[\frac{\star t_1}{\star b_1 \star}\right], \left[\frac{\star t_1}{b_1 \star}\right], \left[\frac{\star t_2}{b_2 \star}\right], \cdots, \left[\frac{\star t_k}{b_k \star}\right], \left[\frac{* \diamond}{\diamond}\right]\right\}$$

where

$$\begin{aligned} \star u &= * u_1 * u_2 * u_3 * \cdots * u_n \\ u \star &= u_1 * u_2 * u_3 * \cdots * u_n * \\ \star u \star &= * u_1 * u_2 * u_3 * \cdots * u_n * \end{aligned} .$$

Now, M accepts w if and only if P has a match.

Undecidability of the PCP (cont.)

To ensure that a match starts with $\left[\frac{t_1}{b_1}\right]$,

S converts the collection $\left\{\left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \cdots, \left[\frac{t_k}{b_k}\right]\right\}$ to

$$\left\{\left[\frac{\star t_1}{\star b_1 \star}\right], \left[\frac{\star t_1}{b_1 \star}\right], \left[\frac{\star t_2}{b_2 \star}\right], \cdots, \left[\frac{\star t_k}{b_k \star}\right], \left[\frac{* \diamond}{\diamond}\right]\right\}$$

where

$$\begin{aligned} \star u &= * u_1 * u_2 * u_3 * \cdots * u_n \\ u \star &= u_1 * u_2 * u_3 * \cdots * u_n * \\ \star u \star &= * u_1 * u_2 * u_3 * \cdots * u_n * \end{aligned} .$$

Now, M accepts w if and only if P has a match.

In summary, S could decide A_{TM} by first constructing an instance P of PCP as above from the input $\langle M, w \rangle$, and then applying R to P. This contradicts Theorem 4.11. Thus, R cannot exist, i.e., PCP is undecidable.

Mapping (Many-One) Reducibility

So far, we have used the reducibility method to prove that many languages are undecidable.

Next: A special type of reducibility called mapping reducibility (in most other textbooks, many-one reducibility).

This technique is more powerful than general reducibility and can be used to prove stronger results in some situations because it preserves more information about Turing-recognizability.

Computable Functions

- A Turing machine computes a function by starting with the input to the function on the tape and halting with the output of the function on the tape.

Definition (5.17)

A function $f : \Sigma^* \longrightarrow \Sigma^*$ is a **computable function** if some Turing machine M, on every input w, halts with just $f(w)$ on its tape.

- For example, all usual arithmetic operations on integers are computable functions.
- Computable functions may be transformations of machine descriptions.

Mapping (Many-One) Reducibility (cont.)

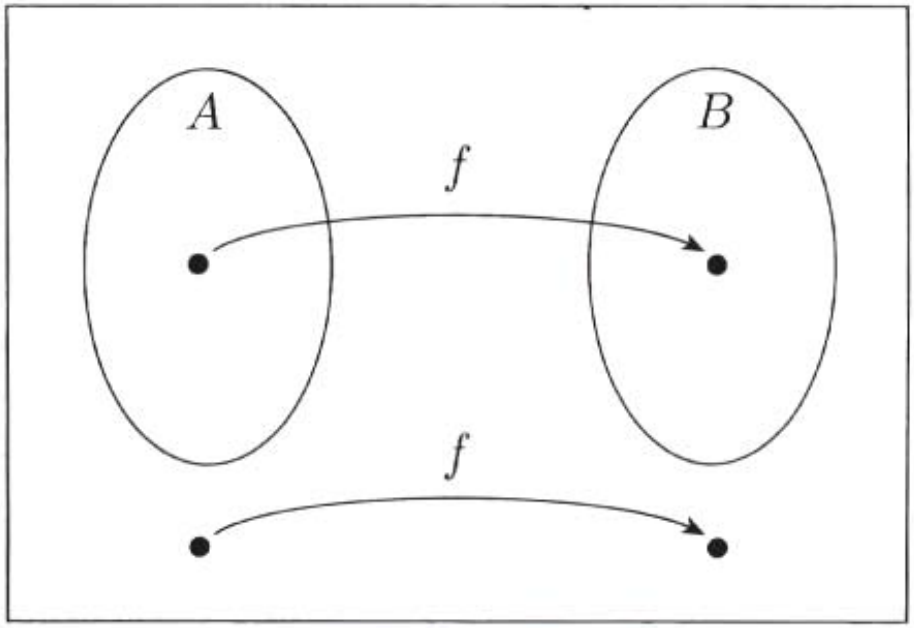
Definition (5.20)

Language A is **mapping reducible** (many-one reducible) to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \longrightarrow \Sigma^*$, where for every w, $w \in A \iff f(w) \in B$.

Mapping (Many-One) Reducibility (cont.)

Definition (5.20)

Language A is **mapping reducible** (many-one reducible) to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \longrightarrow \Sigma^*$, where for every w, $w \in A \iff f(w) \in B$.



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

Reducibility and Decidability

Theorem (5.22)

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

- Let M be a decider for B and f a reduction from A to B. A decider N for A works as follows.
- $N =$ "On input w:
  1. Compute $f(w)$.
  2. Run M on input $f(w)$ and output whatever M outputs."

Reducibility and Decidability

Theorem (5.22)

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

- Let M be a decider for B and f a reduction from A to B. A decider N for A works as follows.
- $N =$ "On input w:
  1. Compute $f(w)$.
  2. Run M on input $f(w)$ and output whatever M outputs."

Corollary (5.23)

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

Note: $(P \wedge Q) \rightarrow R \equiv P \rightarrow (Q \rightarrow R) \equiv P \rightarrow (\neg R \rightarrow \neg Q) \equiv (P \wedge \neg R) \rightarrow \neg Q$

Reducibility and Recognizability

Theorem (5.28)

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

Corollary (5.29)

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

Corollary

If $A \leq_m B$ (i.e., $\overline{A} \leq_m \overline{B}$) and A is not co-Turing-recognizable, then B is not co-Turing-recognizable.

Note: "A is not co-Turing-recognizable" is the same as "$\overline{A}$ is not Turing-recognizable".

The Halting Problem

Recall from the previous lecture that this language is undecidable:

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

Theorem (5.1)

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

In the previous lecture, Theorem 5.1 was proved using the reducibility method.

Next, we give an alternative proof of Theorem 5.1, using mapping reducibility instead.

The Halting Problem (cont.)

- We show that $A_{\mathrm{TM}} \leq_m HALT_{\mathrm{TM}}$, i.e., a computable function f exists (as defined by F below) such that

$$\langle M, w \rangle \in A_{\mathrm{TM}} \iff f(\langle M, w \rangle) \in HALT_{\mathrm{TM}}.$$

The Halting Problem (cont.)

- We show that $A_{\mathrm{TM}} \leq_m HALT_{\mathrm{TM}}$, i.e., a computable function f exists (as defined by F below) such that

$$\langle M, w \rangle \in A_{\mathrm{TM}} \iff f(\langle M, w \rangle) \in HALT_{\mathrm{TM}}.$$

- $F =$ "On input $\langle M, w \rangle$:
  1. Construct the following machine M'.
     $M' =$ "On input x:
     - 1.1 Run M on x.
     - 1.2 If M accepts, *accept*.
     - 1.3 If M rejects, enter a loop.
  2. Output $\langle M', w \rangle$."

  (Improperly formed inputs are assumed to map to strings not in $HALT_{\mathrm{TM}}$.)

The Halting Problem (cont.)

- We show that $A_{\mathrm{TM}} \leq_m HALT_{\mathrm{TM}}$, i.e., a computable function f exists (as defined by F below) such that

$$\langle M, w \rangle \in A_{\mathrm{TM}} \iff f(\langle M, w \rangle) \in HALT_{\mathrm{TM}}.$$

- $F =$ "On input $\langle M, w \rangle$:
  1. Construct the following machine M'.
     $M' =$ "On input x:
     - 1.1 Run M on x.
     - 1.2 If M accepts, *accept*.
     - 1.3 If M rejects, enter a loop.
  2. Output $\langle M', w \rangle$."

  (Improperly formed inputs are assumed to map to strings not in $HALT_{\mathrm{TM}}$.)

Finally, by applying "A_{TM} is undecidable." [Theorem 4.11] and "If $A \leq_m B$ and A is undecidable, then B is undecidable." [Corollary 5.23], we get "$HALT_{\mathrm{TM}}$ is undecidable.".

(From Chapter 4)

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

Remark: Although A_{TM} is undecidable by Theorem 4.11, A_{TM} is Turing-recognizable because there exists a TM U with $L(U) = A_{\mathsf{TM}}$:

- $U =$ "On input $\langle M, w\rangle$, where M is a TM and w is a string:
  1. Simulate M on input w.
  2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *reject*."

Note that U recognizes A_{TM}, but U does not decide A_{TM}.
The reason is that if M loops on w then U loops on input $\langle M, w\rangle$.

U is a **universal Turing machine**, first proposed by Alan Turing in 1936. It is called "universal" because it can simulate any other Turing machine from the description of that machine.

Now how about $HALT_{TM}$?

- $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$.
- $HALT_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on } w\}$.

Note the difference in the definitions of A_{TM} and $HALT_{\mathrm{TM}}$:

- If M accepts w then $\langle M, w\rangle \in A_{\mathrm{TM}}$ and $\langle M, w\rangle \in HALT_{\mathrm{TM}}$.
- If M rejects w then $\langle M, w\rangle \notin A_{\mathrm{TM}}$ but $\langle M, w\rangle \in HALT_{\mathrm{TM}}$.
- If M **loops** on w then $\langle M, w\rangle \notin A_{\mathrm{TM}}$ and $\langle M, w\rangle \notin HALT_{\mathrm{TM}}$.

Now how about $HALT_{TM}$?

- $A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$.
- $HALT_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on } w\}$.

Note the difference in the definitions of A_{TM} and $HALT_{\text{TM}}$:

- If M accepts w then $\langle M, w\rangle \in A_{\text{TM}}$ and $\langle M, w\rangle \in HALT_{\text{TM}}$.
- If M rejects w then $\langle M, w\rangle \notin A_{\text{TM}}$ but $\langle M, w\rangle \in HALT_{\text{TM}}$.
- If M **loops** on w then $\langle M, w\rangle \notin A_{\text{TM}}$ and $\langle M, w\rangle \notin HALT_{\text{TM}}$.

Very similar! Indeed, we proved that $HALT_{\text{TM}}$ is undecidable by reducing A_{TM} to it. Next, let's modify the universal Turing machine U from chapter 4:

- U'= "On input $\langle M, w\rangle$, where M is a TM and w is a string:
  1. Simulate M on input w.
  2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *accept*."

The language recognized by U' is $HALT_{\text{TM}}$. $\Rightarrow$ $HALT_{\text{TM}}$ is Turing-recognizable.

Illustrating the power of mapping reducibility

From Chapter 5.1:

- $EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$.

Theorem (5.4)

EQ_{TM} is undecidable.

We will now use the corollaries of Theorem 5.28 to prove something even stronger about EQ_{TM}...

Illustrating the power of mapping reducibility (cont.)

- $EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$.

Theorem (5.30 Part One)

EQ_{TM} is not Turing-recognizable.

Illustrating the power of mapping reducibility (cont.)

- $EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$.

Theorem (5.30 Part One)

EQ_{TM} is not Turing-recognizable.

- We show that A_{TM} reduces to $\overline{EQ_{\mathrm{TM}}}$, i.e., $\overline{A_{\mathrm{TM}}}$ reduces to EQ_{TM}.
- Since $\overline{A_{\mathrm{TM}}}$ is not Turing-recognizable, EQ_{TM} is not Turing-recognizable.

Illustrating the power of mapping reducibility (cont.)

- $EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$.

Theorem (5.30 Part One)

EQ_{TM} is not Turing-recognizable.

- We show that A_{TM} reduces to $\overline{EQ_{\mathrm{TM}}}$, i.e., $\overline{A_{\mathrm{TM}}}$ reduces to EQ_{TM}.
- Since $\overline{A_{\mathrm{TM}}}$ is not Turing-recognizable, EQ_{TM} is not Turing-recognizable.
- $F =$ "On input $\langle M, w\rangle$:
  1. Construct the following two machines M_1 and M_2.
     $M1 =$"On any input: *reject*."
     $M2 =$"On any input: Run M on w. If it accepts, *accept*."
  2. Output $\langle M_1, M_2\rangle$."

Illustrating the power of mapping reducibility (cont.)

- $EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$.

Theorem (5.30 Part Two)

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

Illustrating the power of mapping reducibility (cont.)

- $EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$.

Theorem (5.30 Part Two)

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

- We show that A_{TM} reduces to EQ_{TM}.
- Since A_{TM} is not co-Turing-recognizable, EQ_{TM} is not co-Turing-recognizable.

Illustrating the power of mapping reducibility (cont.)

- $EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$.

Theorem (5.30 Part Two)

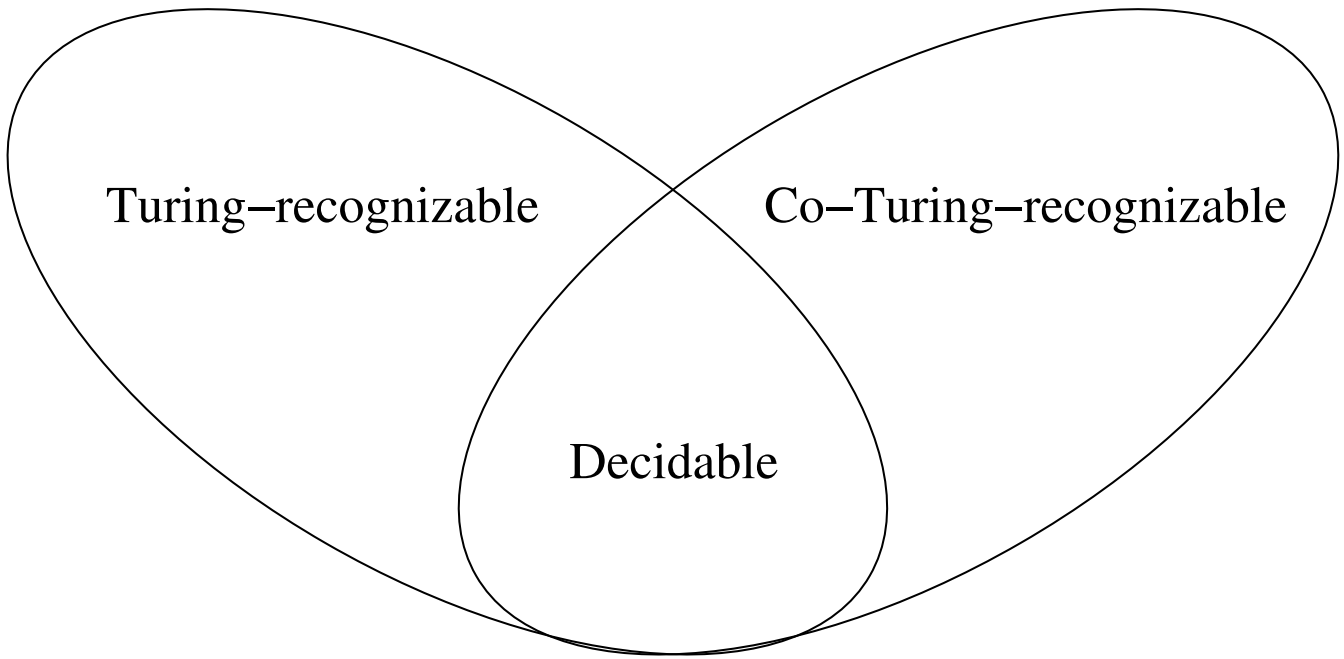
EQ_{TM} is not co-Turing-recognizable.

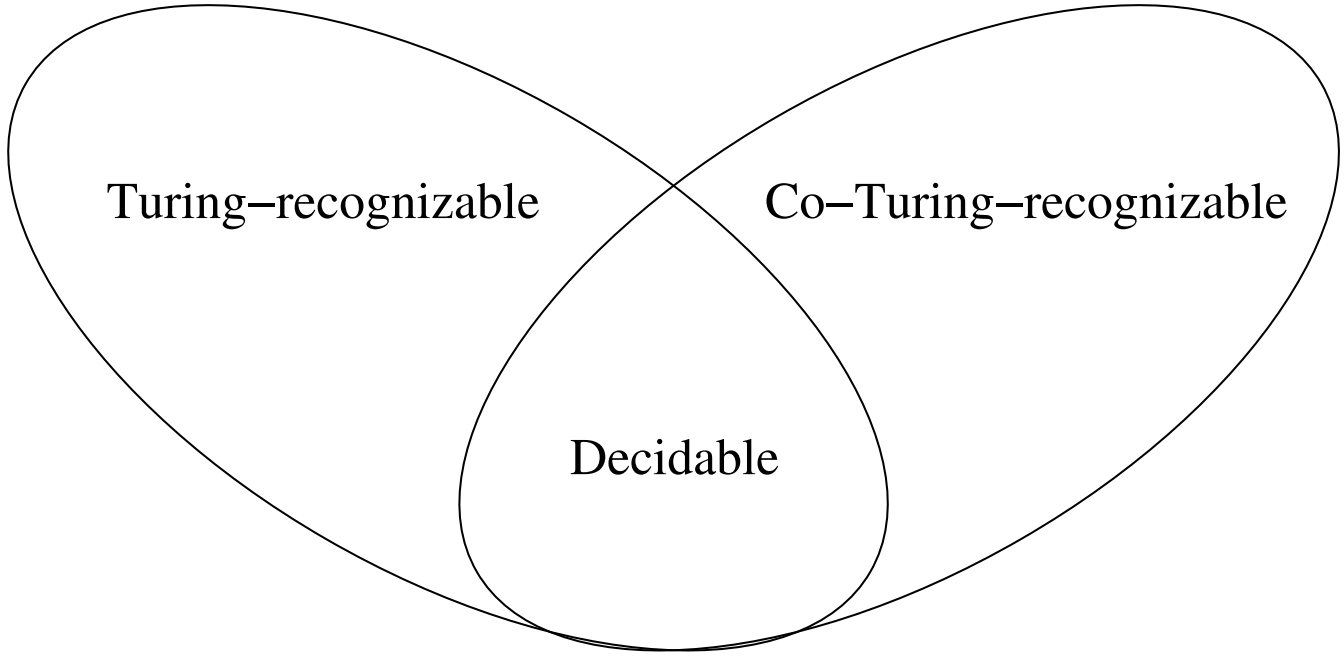
- We show that A_{TM} reduces to EQ_{TM}.
- Since A_{TM} is not co-Turing-recognizable, EQ_{TM} is not co-Turing-recognizable.
- $G =$ "On input $\langle M, w\rangle$:
  1. Construct the following two machines M_1 and M_2.
     $M1 =$"On any input: *accept*."
     $M2 =$"On any input: Run M on w. If it accepts, *accept*."
  2. Output $\langle M_1, M_2\rangle$."

- *Decidable language:*
  There exists a TM that, for any input string w, tells us whether w **is or is not** in the language.

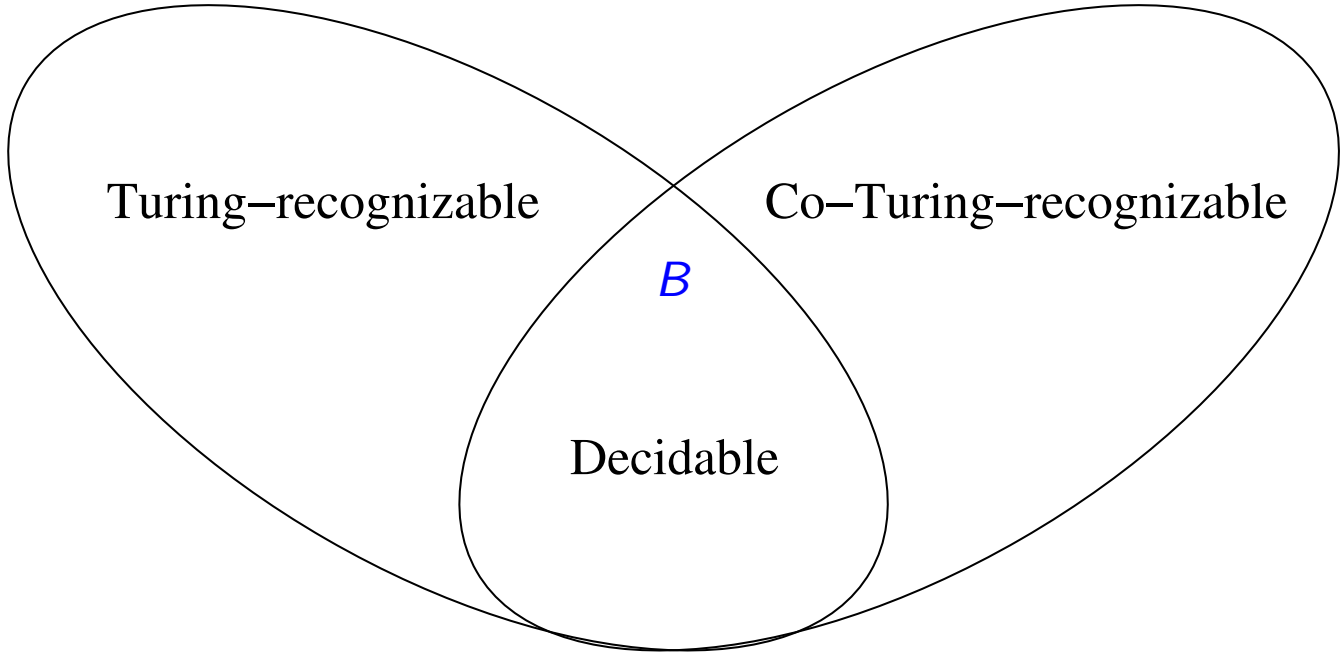
- *Turing-recognizable language*:
  There exists a TM that, for any input string w, verifies if w is in the language.

- *Co-Turing-recognizable language*:
  There exists a TM that, for any input string w, verifies if w is not in the language.

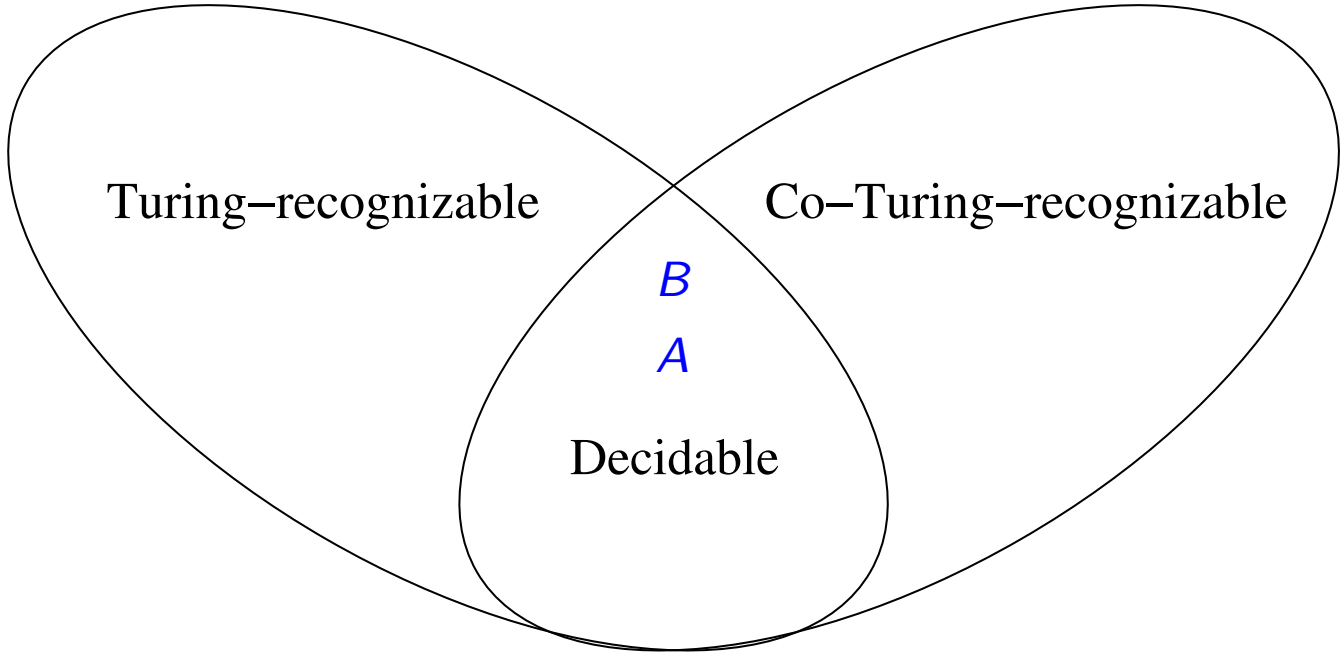




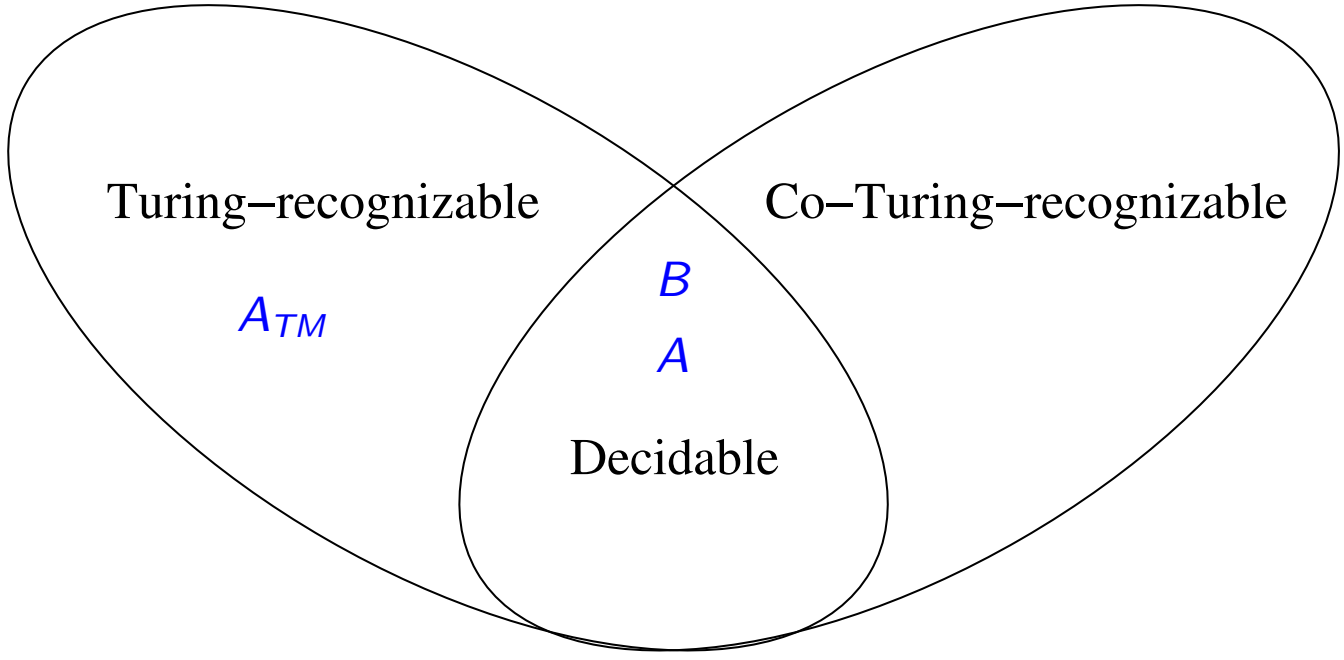
- $B = \{w\#w \mid w \in \{0,1\}^*\}$
- $A = \{\langle G \rangle \mid G \text{ is a connected, undirected graph}\}$
- $A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$
- $HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$
- $E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } M \text{ does not accept any strings at all}\}$
- $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs that accept the same set of strings}\}$



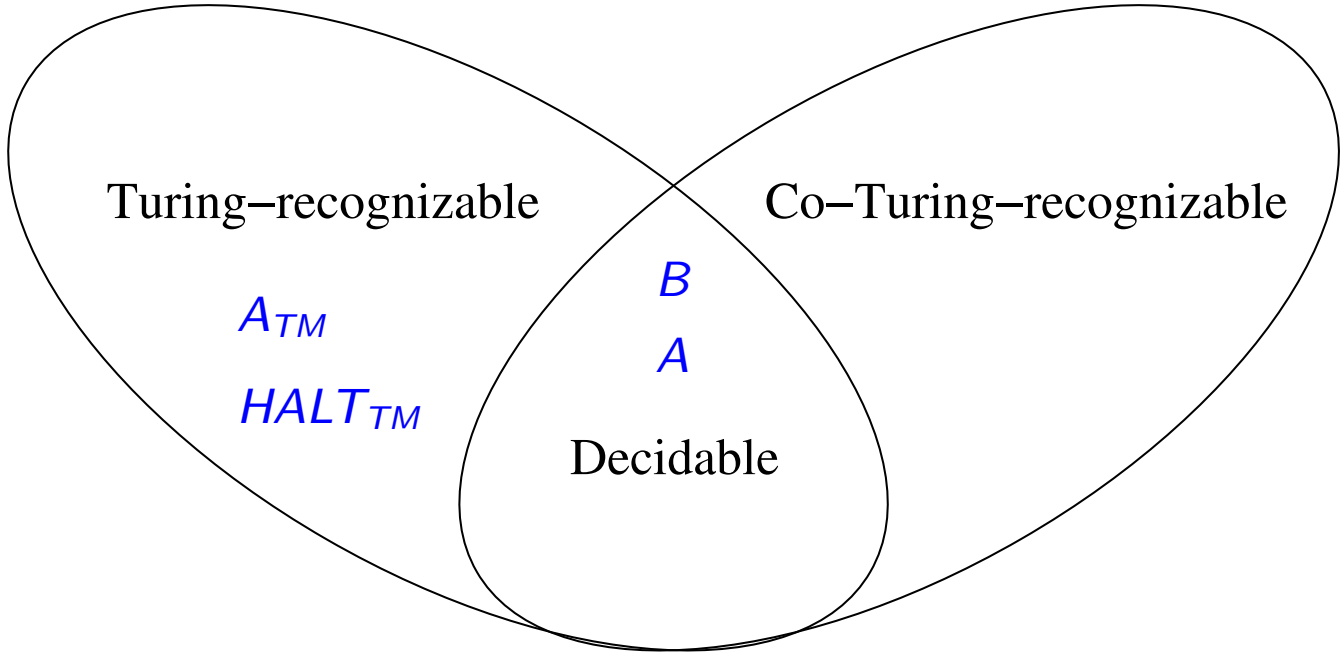
- $B = \{w\#w \mid w \in \{0,1\}^*\}$
- $A = \{\langle G \rangle \mid G \text{ is a connected, undirected graph}\}$
- $A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$
- $HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$
- $E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } M \text{ does not accept any strings at all}\}$
- $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs that accept the same set of strings}\}$



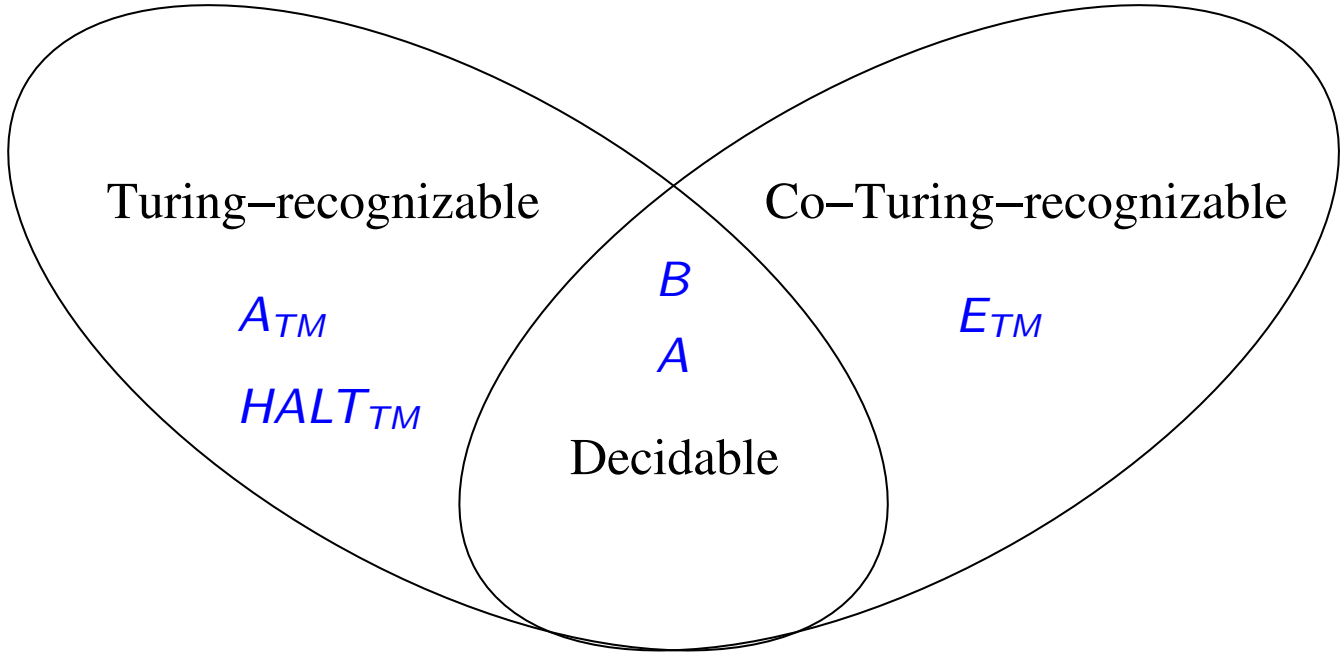
- $B = \{w\#w \mid w \in \{0,1\}^*\}$
- $A = \{\langle G \rangle \mid G \text{ is a connected, undirected graph}\}$
- $A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$
- $HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$
- $E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } M \text{ does not accept any strings at all}\}$
- $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs that accept the same set of strings}\}$



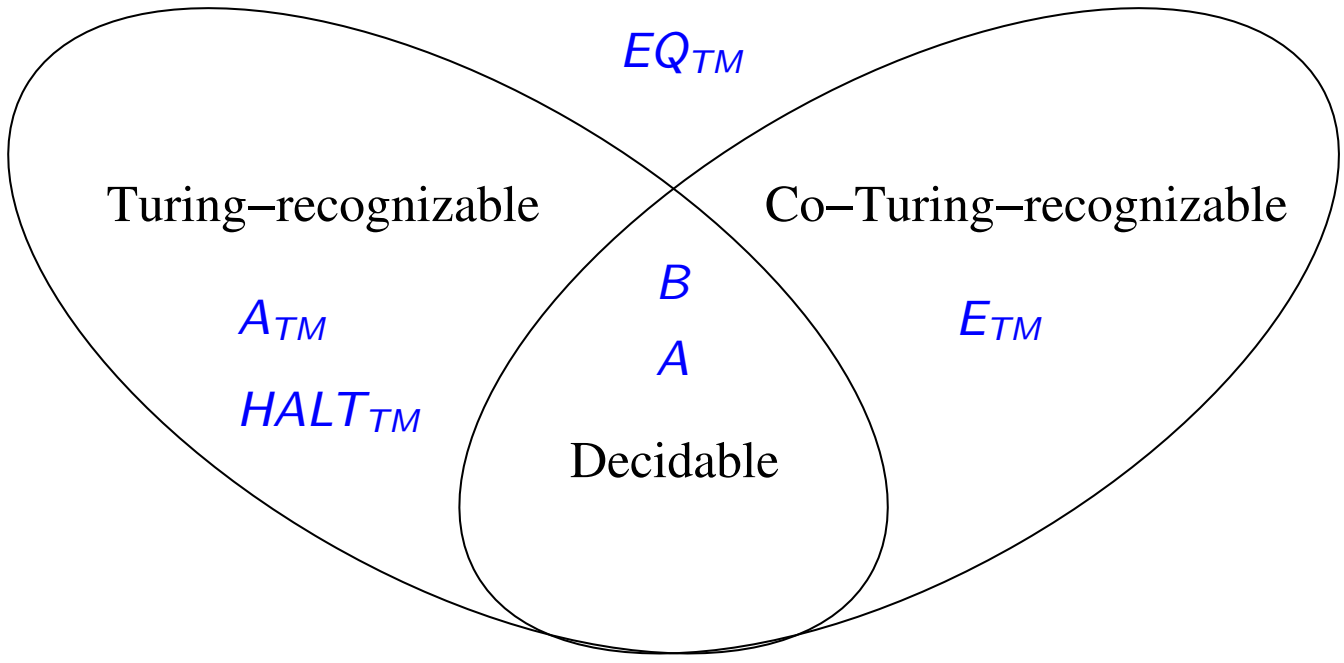
- $B = \{w\#w \mid w \in \{0,1\}^*\}$
- $A = \{\langle G \rangle \mid G$ is a connected, undirected graph$\}$
- $A_{TM} = \{\langle M, w \rangle \mid M$ is a TM and M accepts $w\}$
- $HALT_{TM} = \{\langle M, w \rangle \mid M$ is a TM and M halts on input $w\}$
- $E_{TM} = \{\langle M \rangle \mid M$ is a TM and M does not accept any strings at all$\}$
- $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1$ and M_2 are TMs that accept the same set of strings$\}$



- $B = \{w\#w \mid w \in \{0,1\}^*\}$
- $A = \{\langle G \rangle \mid G$ is a connected, undirected graph$\}$
- $A_{TM} = \{\langle M, w \rangle \mid M$ is a TM and M accepts $w\}$
- $HALT_{TM} = \{\langle M, w \rangle \mid M$ is a TM and M halts on input $w\}$
- $E_{TM} = \{\langle M \rangle \mid M$ is a TM and M does not accept any strings at all$\}$
- $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1$ and M_2 are TMs that accept the same set of strings$\}$



- $B = \{w\#w \mid w \in \{0,1\}^*\}$
- $A = \{\langle G \rangle \mid G$ is a connected, undirected graph$\}$
- $A_{TM} = \{\langle M, w \rangle \mid M$ is a TM and M accepts $w\}$
- $HALT_{TM} = \{\langle M, w \rangle \mid M$ is a TM and M halts on input $w\}$
- $E_{TM} = \{\langle M \rangle \mid M$ is a TM and M does not accept any strings at all$\}$
- $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1$ and M_2 are TMs that accept the same set of strings$\}$



- $B = \{w\#w \mid w \in \{0,1\}^*\}$
- $A = \{\langle G \rangle \mid G$ is a connected, undirected graph$\}$
- $A_{TM} = \{\langle M, w \rangle \mid M$ is a TM and M accepts $w\}$
- $HALT_{TM} = \{\langle M, w \rangle \mid M$ is a TM and M halts on input $w\}$
- $E_{TM} = \{\langle M \rangle \mid M$ is a TM and M does not accept any strings at all$\}$
- $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1$ and M_2 are TMs that accept the same set of strings$\}$