

## 5. Reducibility

- The proof of unsolvability of the halting problem is an example of a *reduction*:
  - a way of converting problem  $A$  to problem  $B$  in such a way that a solution to problem  $B$  can be used to solve problem  $A$
  - If the halting problem were decidable, then the universal language would also be decidable
- Reducibility says nothing about solving either of the problems alone; they just have this connection
  - We know from other sources that the universal language is not decidable
- When problem  $A$  is reducible to problem  $B$ , solving  $A$  cannot be harder than solving  $B$  because a solution to  $B$  gives one to  $A$ 
  - If an unsolvable problem is reducible to another problem, the latter also must be unsolvable



## Non-emptiness Testing for TMs

(Observe that the book deals with  $E_{TM}$ .)

The following decision problem is undecidable:

“Does the given Turing machine accept any inputs?”

$$NE_{TM} = \{ \langle M \rangle \mid M \text{ is a Turing machine and } L(M) \neq \emptyset \}$$

**Theorem (5.2)**  $NE_{TM}$  is Turing-recognizable, but not decidable

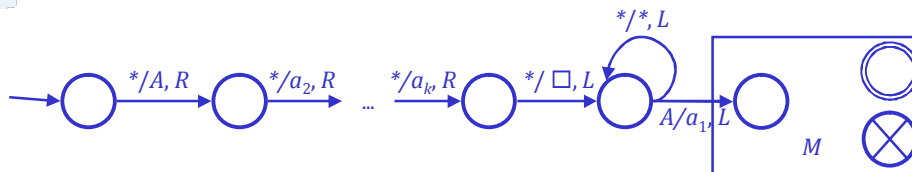
**Proof.** The fact that  $NE_{TM}$  is Turing-recognizable will be shown in the exercises.

- Let us assume that  $NE_{TM}$  has a decider  $M_{NE}^T$
- Using it we can construct a total Turing machine for the language  $U$
- Let  $M$  be an arbitrary Turing machine, whose operation on input  $w$  is under scrutiny



- Let  $M^w$  be a Turing machine that replaces its actual input with the string  $w = a_1 a_2 \dots a_k$  and then works as  $M$
- Operation of  $M^w$  does not depend in any way about the actual input.
  - The TM either accepts or rejects all inputs:

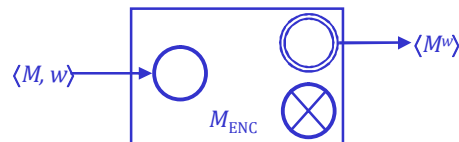
$$L(M^w) = \begin{cases} \{0,1\}^* & \text{if } w \in L(M) \\ \emptyset & \text{if } w \notin L(M) \end{cases}$$



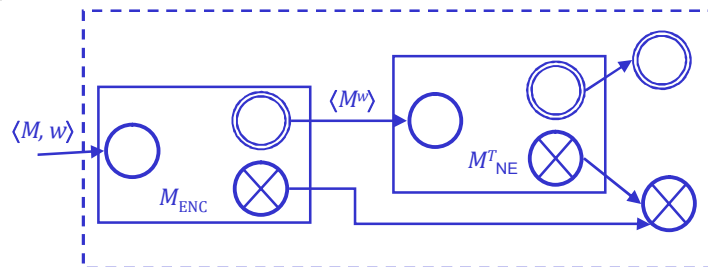
The Turing machine  $M^w$



- Let  $M_{ENC}$  be a TM, which
  - Inputs the concatenation of the code  $\langle M \rangle$  for a Turing machine  $M$  and a binary string  $w$ ,  $\langle M, w \rangle$ , and
  - Leaves to the tape the code  $\langle M^w \rangle$  of the TM  $M^w$



- By combining  $M_{ENC}$  and the decider  $M_{NE}^T$  for the language  $NE_{TM}$  we are now able to construct the following Turing machine  $M_U^T$



A decider  $M_U^T$  for the universal language  $U$



- $M_U^T$  is total whenever  $M_{NE}^T$  is, and  $L(M_U^T) = U$  because

$$\begin{aligned} \langle M, w \rangle \in L(M_U^T) & \\ \Leftrightarrow \langle M^w \rangle \in L(M_{NE}^T) = NE_{TM} & \\ \Leftrightarrow L(M^w) \neq \emptyset & \\ \Leftrightarrow w \in L(M) & \\ \Leftrightarrow \langle M, w \rangle \in U & \end{aligned}$$

- However, by Theorem G  $U$  is not decidable, and the existence of the TM  $M_U^T$  is a contradiction
- Hence, the language  $NE_{TM}$  cannot have a total recognizer  $M_{NE}^T$  and we have, thus, proved that the language  $NE_{TM}$  is not decidable.  $\square$



## TMs Recognizing Regular Languages

- Similarly, we can show that recognizing those Turing machines that accept a regular language is undecidable by reducing the decidability of the universal language into this problem

The decision problem is:

*"Does the given Turing machine accept a regular language?"*

$$REG_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language} \}$$

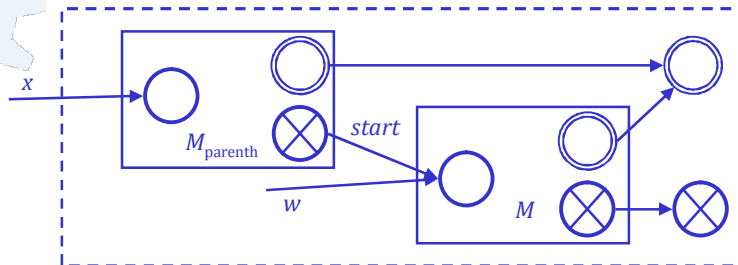
**Theorem 5.3**  $REG_{TM}$  is undecidable.

**Proof.**

- Let us assume that  $REG_{TM}$  has a decider  $M_{REG}^T$



- Using  $M_{REG}^T$  we could construct a decider for the universal language  $U$
- Let  $M$  be an arbitrary Turing machine, whose operation on input  $w$  we are interested in
- The language corresponding to balanced pairs of parenthesis  $\{0^n 1^n \mid n \geq 0\}$  is not regular, but easy to decide using a TM
- Let  $M_{parenth}$  be a decider for the language
- Now, let  $M_{ENC}$  be a TM, which on input  $\langle M, w \rangle$  composes an encoding for a TM  $M^w$ , which on input  $x$ 
  - First works as  $M_{parenth}$  on input  $x$ .
  - If  $M_{parenth}$  rejects  $x$ ,  $M^w$  operates as  $M$  on input  $w$ .
  - Otherwise  $M^w$  accepts  $x$

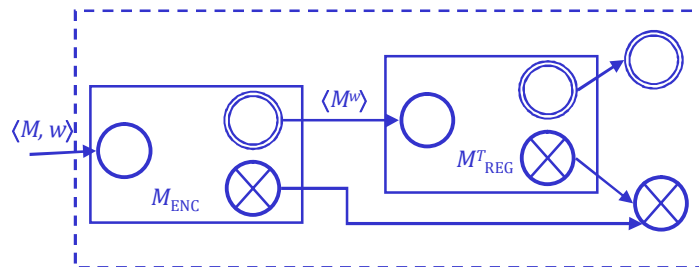


Deciding a regular language: the TM  $M^w$

- Thus,  $M^w$  either accepts the regular language  $\{0,1\}^*$  or non-regular  $\{0^n1^n \mid n \geq 0\}$
- Accepting/rejecting the string  $w$  on  $M$  reduces to the question of the regularity of the language of the TM  $M^w$

$$L(M^w) = \begin{cases} \{0,1\}^* & \text{if } w \in L(M) \\ \{0^n1^n \mid n \geq 0\} & \text{if } w \notin L(M) \end{cases}$$

- Let  $M_{ENC}$  be a TM, which
  - inputs the concatenation of the code  $\langle M \rangle$  for a Turing machine  $M$  and a binary string  $w$ ,  $\langle M, w \rangle$ , and
  - Leaves to the tape the code  $\langle M^w \rangle$  of the TM  $M^w$
- Now by combining  $M_{ENC}$  and  $M_{REG}^T$  would yield the following Turing machine  $M_U^T$



A decider  $M_U^T$  for the universal language  $U$



- $M_U^T$  is total whenever  $M_{REG}^T$  is and  $L(M_U^T) = U$ , because

$$\begin{aligned} \langle M, w \rangle \in L(M_U^T) & \\ \Leftrightarrow \langle M^w \rangle \in L(M_{REG}^T) = \text{REG}_{TM} & \\ \Leftrightarrow L(M^w) \text{ is a regular language} & \\ \Leftrightarrow w \in L(M) & \\ \Leftrightarrow \langle M, w \rangle \in U & \end{aligned}$$

- By Theorem G,  $U$  is not decidable, and the existence of the TM  $M_U^T$  is a contradiction
- Hence, language  $\text{REG}_{TM}$  cannot have a decider  $M_{REG}^T$
- Thus, we have shown that the language  $\text{REG}_{TM}$  is not decidable  $\square$



## Rice's Theorem

- Any property that only depends on the language recognized by a TM, not on its syntactic details, is called a *semantic property* of the Turing machine
- E.g.
  - " $M$  accept the empty string",
  - " $M$  accepts some string" ( $NE$ ),
  - " $M$  accept infinitely many strings",
  - "The language of  $M$  is regular" ( $REG$ ) etc.
- If two Turing machines  $M_1$  and  $M_2$  have  $L(M_1) = L(M_2)$ , then they have exactly the same semantic properties





- More abstractly: a semantic property  $S$  is any collection of Turing-recognizable languages over the alphabet  $\{0, 1\}$
- Turing machine  $M$  has property  $S$  if  $L(M) \in S$ .
- *Trivial* properties are  $S = \emptyset$  and  $S = TR$
- Property  $S$  is *solvable*, if language 
$$\text{codes}(S) = \{\langle M \rangle \mid L(M) \in S\}$$
 is decidable.

**Rice's theorem** *All non-trivial semantic properties of Turing machines are unsolvable*



## Computation Histories

- The **computation history** for a Turing machine on an input is simply the sequence of configurations that the machine goes through as it processes the input.
- An accepting computation history for  $M$  on  $w$  is a sequence of configurations  $C_1, C_2, \dots, C_j$ , where
  - $C_1$  is the start configuration  $q_0 w$ ,
  - $C_j$  an accepting configuration of  $M$ , and
  - each  $C_i$  legally follows from  $C_{i-1}$  according to the rules of  $M$
- Similarly one can define a rejecting computation history
- Computation histories are finite sequences —  
if  $M$  doesn't halt on  $w$ , no accepting or rejecting computation history exists for  $M$  on  $w$



## Linear Bounded Automata

- A linear bounded automaton (LBA) is a Turing machine that cannot use extra working space
- It can only use the space taken up by the input
- Because the tape alphabet can, in any case, be larger than the input alphabet, it allows the available memory to be increased up to a constant factor
- Deciders for problems concerning context-free languages
- If a LBA has
  - $q$  states
  - $g$  symbols in its tape alphabet, and
  - an input of length  $n$ ,
 then the number of its possible configurations is  $q \cdot n \cdot g^n$



### Theorem 5.9

*The acceptance problem for linear bounded automata*

$A_{\text{LBA}} = \{ \langle M, w \rangle \mid M \text{ is an LBA that accepts string } w \}$   
is decidable.

**Proof.** As  $M$  computes on  $w$ , it goes from configuration to configuration. If it ever repeats a configuration, it will go on to repeat this configuration over and over again and thus be in a loop.

Because an LBA has only  $q \cdot n \cdot g^n$  distinct configurations, if the computation of  $M$  has not halted in so many steps, it must be in a loop.

Thus, to decide  $A_{\text{LBA}}$  it is enough to simulate  $M$  on  $w$  for  $q \cdot n \cdot g^n$  steps or until it halts.  $\square$




**Theorem 5.10**

The emptiness problem for linear bounded automata

$$E_{\text{LBA}} = \{ \langle M \rangle \mid M \text{ is an LBA and } L(M) = \emptyset \}$$

is undecidable.

**Proof.** Reduction from the universal language (acceptance problem for general TMs).

Counter-assumption:  $E_{\text{LBA}}$  is decidable; i.e., there exists a decider  $M_E^T$  for  $E_{\text{LBA}}$ .

Let  $M$  be an arbitrary Turing machine, whose operation on input  $w$  is under scrutiny. Let us compose an LBA  $B$  that recognizes all accepting computation histories for  $M$  on  $w$ .



Now we can reduce the acceptance problem for general Turing machines to the emptiness testing for LBAs:

$$\begin{cases} L(B) \neq \emptyset & \text{if } w \in L(M) \\ L(B) = \emptyset & \text{if } w \notin L(M) \end{cases}$$

The LBA  $B$  must accept input string  $x$  if it is an accepting computation history for  $M$  on  $w$ .

Let the input be presented as  $x = C_1 \# C_2 \# \dots \# C_l$ .





$B$  checks that  $x$  satisfies the conditions of an accepting computation history:

- $C_1 = q_0 w$ ,
- $C_i$  is an accepting configuration for  $M$ ; i.e. **accept** is the state in  $C_i$ , and
- $C_{i-1} \Rightarrow_M C_i$ :
  - configurations  $C_{i-1}$  and  $C_i$  are identical except for the position under and adjacent to the head in  $C_{i-1}$ , and
  - the changes correspond to the transition function of  $M$ .

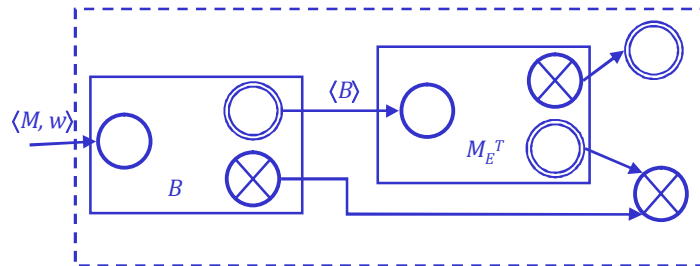
Given  $M$  and  $w$  it is possible to construct LBA  $B$  mechanically.



By combining machines  $B$  and  $M_E^T$  as shown in the following figure, we obtain a decider for the acceptance problem of general Turing machines (universal language).

$$\begin{aligned}
 &\langle M, w \rangle \in L(M_U^T) \\
 &\Leftrightarrow \langle B \rangle \notin L(M_E^T) \\
 &\Leftrightarrow L(B) \neq \emptyset \\
 &\Leftrightarrow w \in L(M) \\
 &\Leftrightarrow \langle M, w \rangle \in U
 \end{aligned}$$

This is a contradiction, and the language  $E_{LBA}$  cannot be decidable.  $\square$



A decider  $M_U^T$  for the universal language  $U$

### 5.3 Mapping Reducibility

- Let  $M = (Q, \Sigma, \Gamma, \delta, q_0, \text{accept}, \text{reject})$  be an arbitrary standard Turing machine
- Let us define the *partial function*  $f_M: \Sigma^* \rightarrow \Gamma^*$  computed by the TM as follows:

$$f_M(w) = \begin{cases} u, & \text{if } q_0 w \xrightarrow{*} u q, \\ & q \in \{\text{accept}, \text{reject}\} \\ \text{undefined} & \text{otherwise} \end{cases}$$

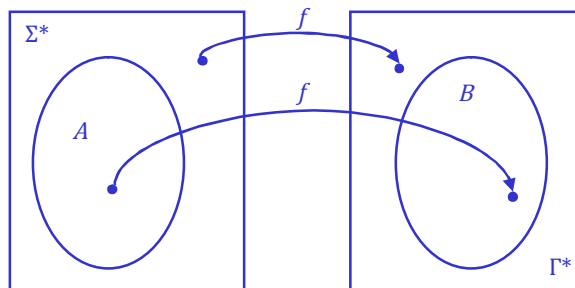
- Thus, the TM  $M$  maps a string  $w \in \Sigma^*$  to the string  $u$ , which is the contents of the tape, if the computation halts on  $w$
- If it does not halt, the value of the function is not defined in  $w$

## Definition 5.20

- Partial function  $f$  is **computable**, if it can be computed with a total Turing machine. I.e. if its value  $f(w)$  is defined for every  $w$
- Let us formulate the idea that problem  $A$  is "at most as difficult as" problem  $B$  as follows
  - Let  $A \subseteq \Sigma^*$ ,  $B \subseteq \Gamma^*$  be two formal languages
  - $A$  is **mapping reducible** to  $B$ , written
 
$$A \leq_m B,$$
 if there is a computable function  $f: \Sigma^* \rightarrow \Gamma^*$  s.t.
 
$$w \in A \Leftrightarrow f(w) \in B \quad \forall w \in \Sigma^*$$
- The function  $f$  is called the **reduction** of  $A$  to  $B$



- Mapping an instance  $w$  of  $A$  computably into an instance  $f(w)$  of  $B$  and
  - "does  $w$  have property  $A$ ?"  $\square$
  - "does  $f(w)$  have property  $B$ ?"





**Lemma K** For all languages  $A, B, C$  the following hold

- i.  $A \leq_m A$ , (reflexive)
- ii. if  $A \leq_m B$  and  $B \leq_m C$ , then  $A \leq_m C$ , (transitive)
- iii. if  $A \leq_m B$  and  $B$  is Turing-recognizable, then so is  $A$ , and
- iv. if  $A \leq_m B$  and  $B$  is decidable, then so is  $A$

**Proof.**

- i. Let us choose  $f(x) = x$  as the reduction.
- ii. Let  $f$  be reduction of  $A$  to  $B$  and  $g$  a reduction of  $B$  to  $C$ .  
In other words,  $f: A \leq_m B$ ,  $g: B \leq_m C$ .

We show that the composite function  $h$ ,  $h(x) = g(f(x))$  is a reduction  $h: A \leq_m C$ .



1.  $h$  is computable: Let  $M_f$  and  $M_g$  be the total Turing machines computing  $f$  and  $g$ .  $M_{\text{REW}}$  replaces all symbols to the right of the tape head with  $\square$  and moves the tape head to the beginning of the tape. The total machine depicted in the following figure computes function  $h$ .

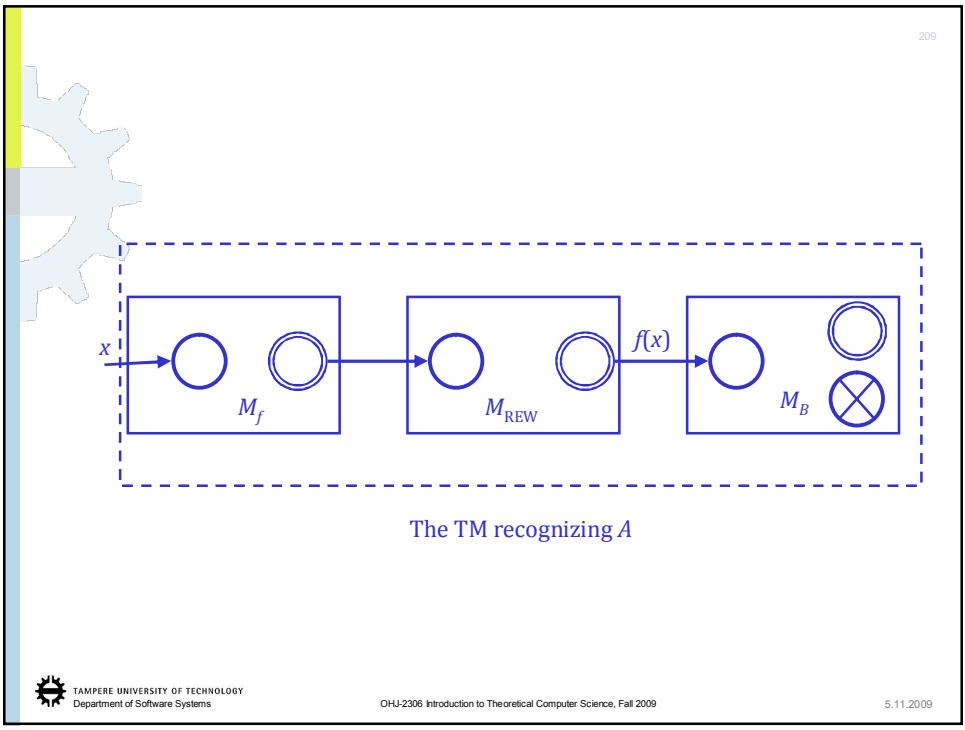
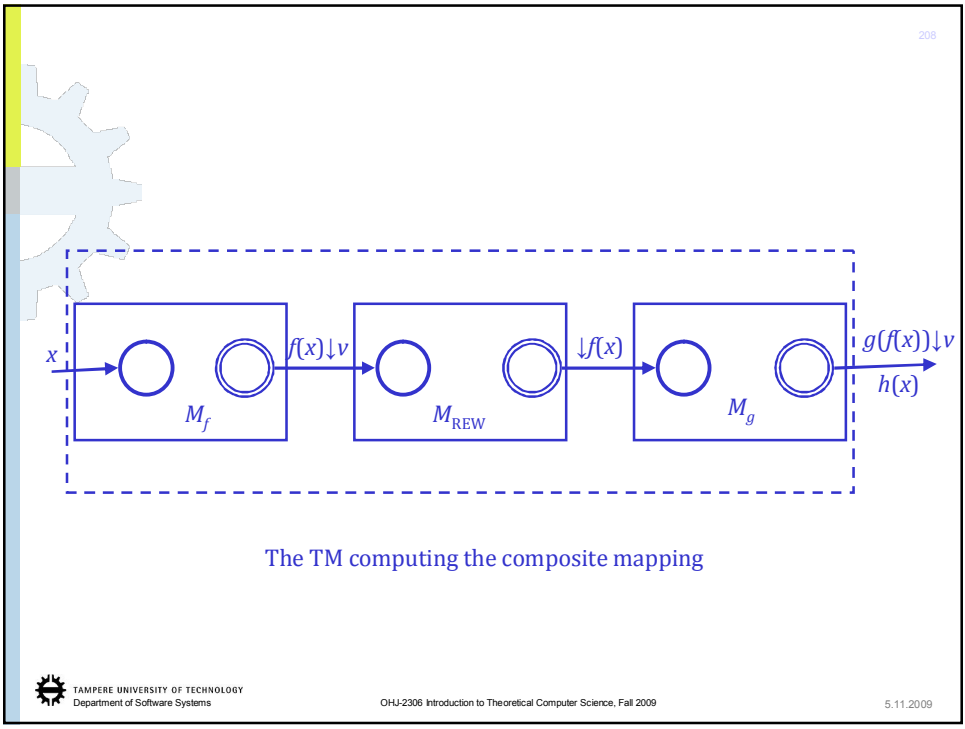
2.  $h$  is a reduction:

$$\begin{aligned} x \in A &\Leftrightarrow f(x) \in B \\ &\Leftrightarrow g(f(x)) = h(x) \in C, \end{aligned}$$

hence,  $h: A \leq_m C$ .

- iii. (and iv.) Let  $f: A \leq_m B$ ,  $M_B$  the recognizer of  $B$  and  $M_f$  the TM computing  $f$ . The TM depicted below recognizes language  $A$  and it is total whenever  $M_B$  is.  $\square$





- We have already used the following consequence of Lemma K to prove undecidability.

**Corollary 5.23** *If  $A \leq_m B$  and  $A$  is undecidable, then  $B$  is undecidable.*

Let us call language  $B \subseteq \{0, 1\}^*$  **TR-complete**, if

1.  $B$  is Turing-recognizable (TR), and
2.  $A \leq_m B$  for all Turing-recognizable languages  $A$

**Theorem L** *The universal language  $U$  is TR-complete.*

**Proof.** We know that  $U$  is Turing-recognizable. Let  $B$  be any Turing-recognizable language. Furthermore, let  $B = L(M_B)$ .

Now,  $B$  can be reduced to  $U$  with the function  $f(x) = \langle M_B, x \rangle$ , which is clearly computable, and for which it holds

$$x \in B = L(M_B) \Leftrightarrow f(x) = \langle M_B, x \rangle \in U. \quad \square$$



**Theorem M** *Let  $A$  be a TR-complete language,  $B$  TR, and  $A \leq_m B$ . Then also  $B$  is a TR-complete language.*

- All "natural" languages belonging to the difference of TR and decidable languages are TR-complete, but it contains also other languages
- The class of TR languages is not closed under complementation, thus it has the dual class

$$\text{co-TR} = \{ \bar{A} \mid A \in \text{TR} \}$$

- $\text{TR} \cap \text{co-TR} = \text{decidable languages}$  (by Theorem C, 4.22)
- $B \subseteq \{0, 1\}^*$  is co-TR-complete, if  $B \in \text{co-TR}$  and  $A \leq_m B$  for all  $A \in \text{co-TR}$
- A language  $A$  is co-TR-complete, if and only if the language  $\bar{A}$  is TR-complete
- Language  $\text{TOT} = \{ \langle M \rangle \mid M \text{ halts on all inputs} \}$  does not belong to either TR or co-TR



