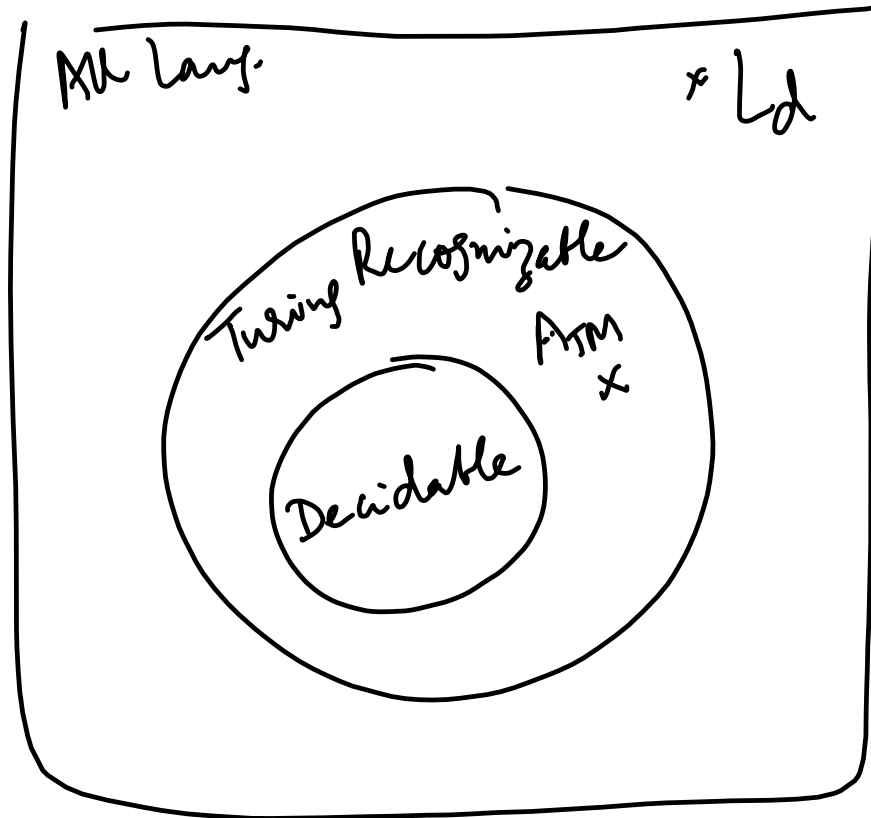


CS 273

4/12/07

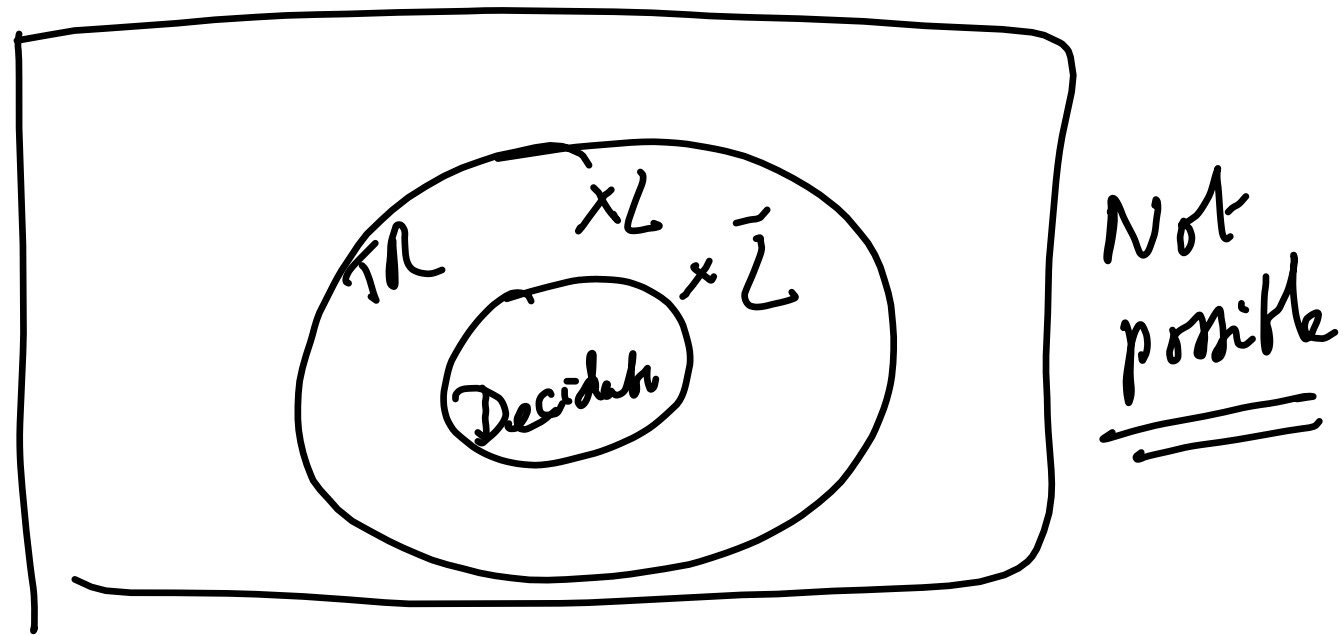
Undecidability via Reductions

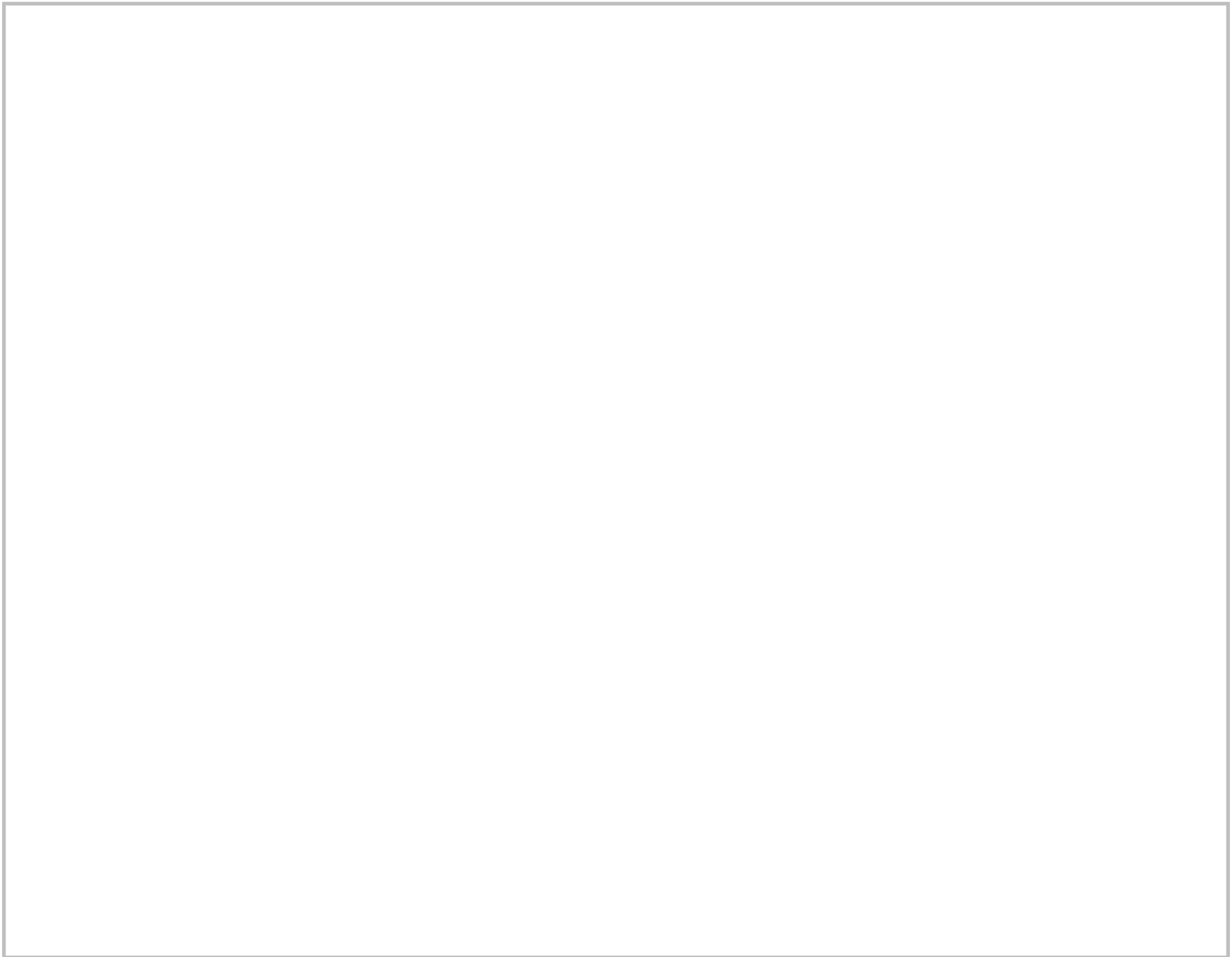


$$L_d = \{ w \mid \text{TM } w \text{ does not accept string } w \}$$

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

Theorem: A language L is
decidable if and only if
 L and \bar{L} are Turing-recognizable





L is decidable implies

L and \bar{L} are T.R

(Turing recognizable)

(L is decidable implies \bar{L} is
decidable) (HW)

L and \bar{L} are TR
implies L (and hence \bar{L})
are decidable

M_1 accepts L

M_2 accepts \bar{L}

create M that decides L

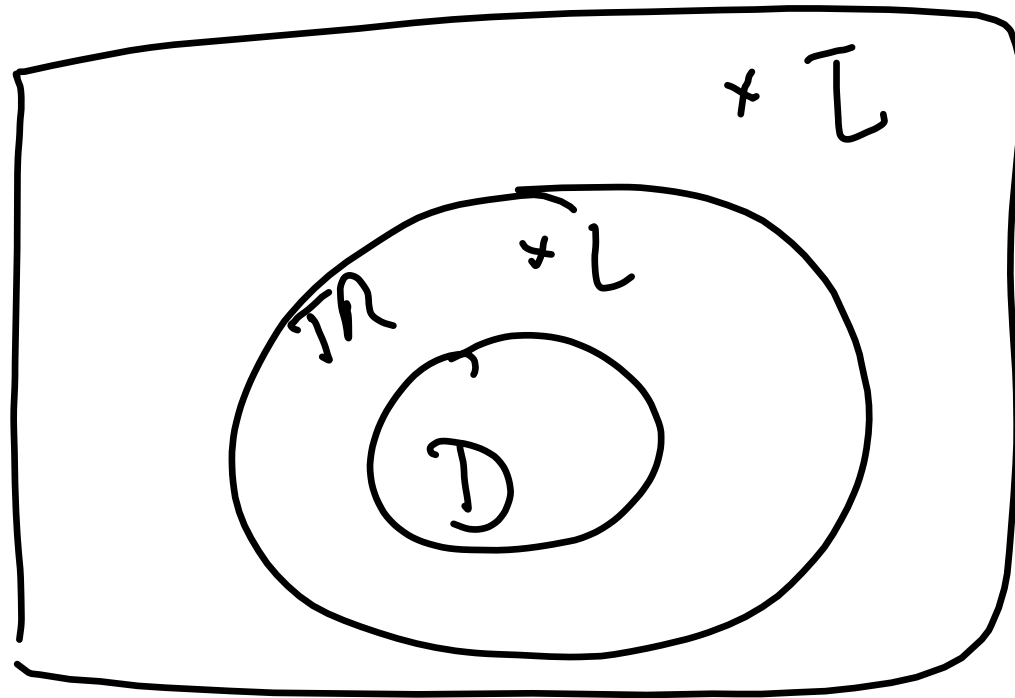
$w \in \Sigma^*$

M : on input w

- Simulate M_1 and M_2 on w in parallel
(1 step of M_1 and 1 step of M_2)
- if M_1 halts first, halt and output answer of M_1
(accept or reject)
- if M_2 halts first, halt and output negation of the answer of M_2 .

If L is T.R and undecidable

then \bar{L} is not TR



A_{TM} is TR but not
decidable

$\Rightarrow \overline{A_{TM}}$ is not TR

$\overline{A_{TM}} = \{ \langle M, w \rangle \mid \exists M \text{ is a TM}$
and w is a
string
and M does not
accept w }

$\cup \{ x \mid x \text{ is not of the}$
form $\langle M, w \rangle \}$
 $\xrightarrow{\text{"easy"}}$ decidable

Reductions

If a task A is easy

\Rightarrow a task B is easy

} reduction
from
 B to A

Given two tasks A and B

and a reduction from B to A

① positive side: solving A implies
solving B

② negative side: if B is hard
then A is hard.

Reduction for Proving Undecidability

A reduction from L_B to L_A

is a TM M s.t. if

L_A is decidable then shows

L_B is decidable.

Know A_{TM} is undecidable

Want to prove

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

is undecidable

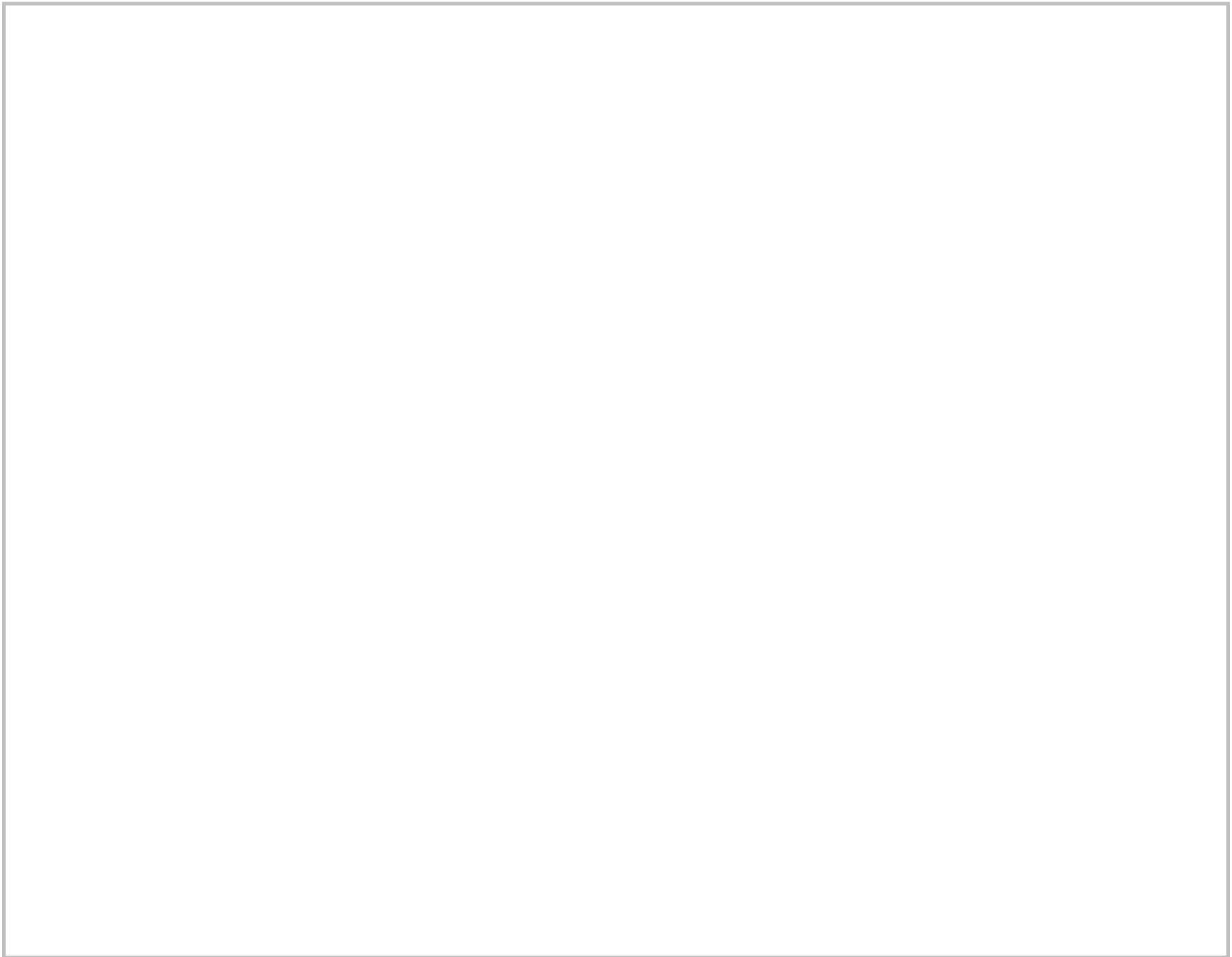
Suppose $HALT_{TM}$ is decidable
then there is a decider N for it
create a decider S for A_{TM}

S : on input $\langle M, w \rangle$

1. Simulate N on $\langle M, w \rangle$
 2. If N accepts $\langle M, w \rangle$
 - 2.a. Simulate M on w
 - 2.b. Output answer of M on w
 3. If N rejects, reject $\langle M, w \rangle$
-

S decides A_{TM} if N decides
 $HALT_{TM}$

$\Rightarrow HALT_{TM}$ is undecidable



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

To prove that

E_{TM} is not decidable
by a reduction from A_{TM}

Machines that rewrite code

Given a string $\langle M, w \rangle$

We will create a machine

$S_{\langle M, w \rangle}$: on input x

1. ignore x and simulate M on w
2. If M accepts w , accept x
3. If M halts and rejects w , reject x

$$\begin{aligned} L(S_{\langle M, w \rangle}) &= \Sigma^* \text{ if } M \text{ accepts } w \\ &= \emptyset \text{ otherwise.} \end{aligned}$$

① $S_{\langle M, w \rangle}$ stores $\langle M, w \rangle$ in its internal state

② Given $\langle M, w \rangle$ there is
a TM H which outputs

$\langle S_{\langle M, w \rangle} \rangle$

(think about this!)

E_{TM} is undecidable
via A_{TM}

Given a decider R for E_{TM}

give a decider N for A_{TM}

What is N ?

N : on input $\langle M, w \rangle$

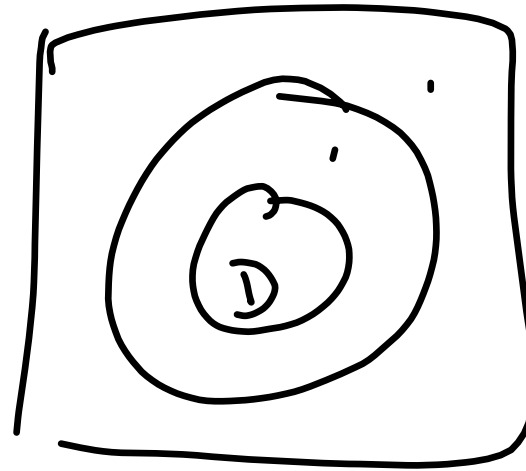
1. Use H on $\langle M, w \rangle$ to generate
 $\langle S_{\langle M, w \rangle} \rangle$

2. give $\langle S_{\langle M, w \rangle} \rangle$ to R as input

3. If R accepts $\langle S_{\langle M, w \rangle} \rangle$
reject $\langle M, w \rangle$

4. If R rejects $\langle S_{\langle M, w \rangle} \rangle$
accept $\langle M, w \rangle$.

E_{TM} is undecidable



$$\overline{E_{TM}} = \left\{ \langle M \rangle \mid \begin{array}{l} M \text{ is a TM} \\ L(M) \neq \emptyset \end{array} \right\}$$

∪ junk

$$\bar{E}_{TM} = \{ \langle M \rangle \mid \langle M \rangle \text{ is a } \\ \text{TM and } L(M) \neq \emptyset \}$$

is TR

A: on input $\langle M \rangle$

for $i = 0$ to ∞ do

for $j = 0$ to i do

- Simulate M on string $\langle j \rangle$ for
 $i-j$ steps

- If M halts and accepts
 halt and ~~say~~ accept $\langle M \rangle$

Why does N work properly?

If M accepts w
then $L(S_{\langle M, w \rangle}) = \Sigma^*$

Else $L(S_{\langle M, w \rangle}) = \emptyset$