Undecidability

Lecture 21

Today

Undecidable Problems

Proving undecidability

Using reductions to prove more undecidability

Language of Universal TM

Language recognized by U:

$$L(U) = \{ (z,w) \mid U \text{ accepts } (z,w) \}$$
$$= \{ (z,w) \mid M_z \text{ accepts } w \}$$

pair of binary strings encoded as a binary string

M_z is the TM encoded by the string M_z

We will call $L(U) = ACCEPT$

Today:

$ACCEPT$ is undecidable!

No matter what encoding schemes are used

Cantor's Diagonal Slash

Is the set of all infinitely long binary strings countable?

Suppose it was: consider *enumerating* them in a table

Consider the string corresponding to the "flipped diagonal"

> It doesn't appear in this table!

0	1	0	0	1	1	1	.	.

S_i									
S_1 =	1	0	0	1	0	0	0	0	1
S_2 =	0	0	1	0	1	0	0	1	1
S_3 =	1	1	1	1	1	1	1	0	0
S_4 =	1	1	0	1	0	1	0	1	1
S_5 =	1	1	0	0	0	0	1	0	0
S_6 =	0	0	0	0	0	0	1	1	0
S_7 =	0	1	0	1	0	1	0	1	1

Undecidability

Table of languages <u>recognized</u> by TMs

$T(z,w) = 1$ iff M_z accepts w

D = "diagonal language"
  $= \{ w \mid M_w \text{ accepts } w \}$

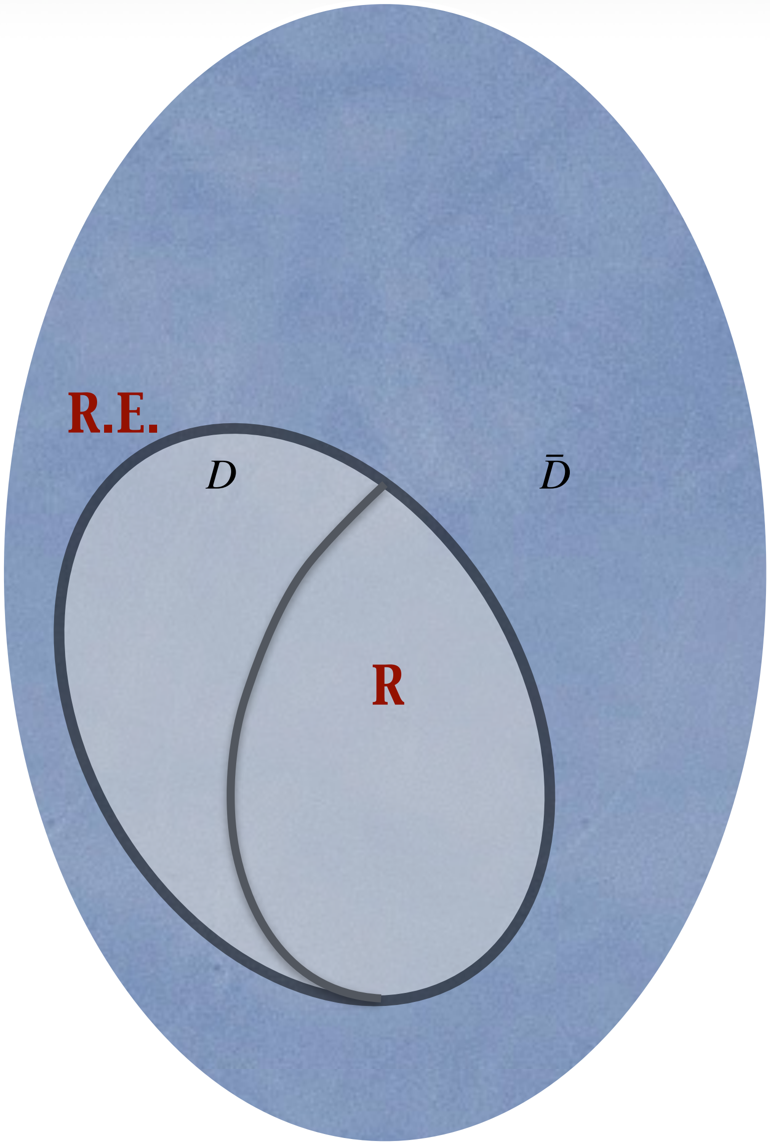
$\bar{D} = \{ w \mid M_w \text{ doesn't accept } w \}$

$\bar{D}$ does not appear as a row in this table. Hence not recognizable!

0	1	0	0	1	1	1	.	.

z \ w	0	1	00	01	10	11	000	001	010
0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	1	0	0	1	1
00	1	1	1	1	1	1	1	0	0
01	1	1	0	1	0	1	0	1	1
10	1	1	0	0	0	0	1	0	0
11	0	0	0	0	0	0	1	1	0
000	0	1	0	1	0	1	0	1	1

Map



Undecidability

Table of languages <u>recognized</u> by TMs

Entries indicate if $(z,w) \in ACCEPT$

0	1	0	0	1	1	1	.	.

z \ w	0	1	00	01	10	11	000	001	010
0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	1	0	0	1	1
00	1	1	1	1	1	1	1	0	0
01	1	1	0	1	0	1	0	1	1
10	1	1	0	0	0	0	1	0	0
11	0	0	0	0	0	0	1	1	0
000	0	1	0	1	0	1	0	1	1

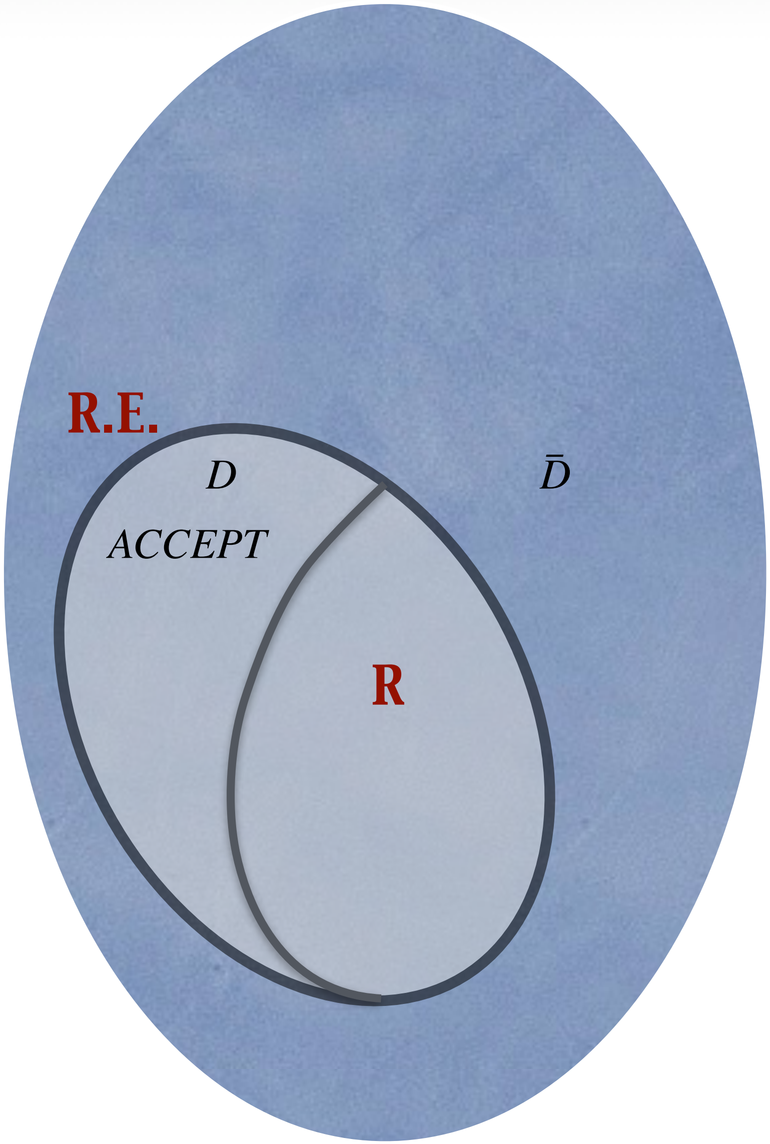
$T(z,w) = 1$ iff M_z accepts w

If $ACCEPT$ decidable, can compute $T(z,w)$ using a TM that halts on every input

Then $\bar{D}$ would be decidable too: On input w, compute $T(w,w)$ and accept iff it is 0

Hence $ACCEPT$ undecidable!

Map



Reduction

We just saw how a “reduction” can show impossibility

1. Showed that if $ACCEPT$ is decidable, then $\bar{D}$ decidable (using a *“reduction” from $\bar{D}$ to* $ACCEPT$)

2. We already saw $\bar{D}$ not decidable

3. Hence $ACCEPT$ not decidable

w / z	0	1	00	01	10	11	000	001	010
0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	1	0	0	1	1
00	1	1	1	1	1	1	1	0	0
01	1	1	0	1	0	1	0	1	1
10	1	1	0	0	0	0	1	0	0
11	0	0	0	0	0	0	1	1	0
000	0	1	0	1	0	1	0	1	1

Reduction

Reduction *from* L_1 *to* L_2 ($L_1 \leq L_2$):

Any instance of L_1 can be solved by solving an instance of L_2
(and there is an algorithm to change the L_1-instance to the L_2-instance)

The task of solving L_1 is *reduced* to the task of solving L_2

Positive implication:
If we can solve L_2, then we can solve L_1

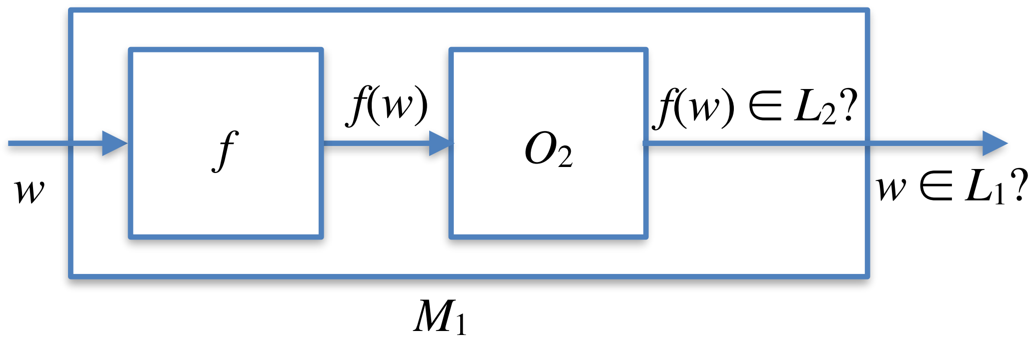
Negative implication:
If we can't solve L_1, then we can't solve L_2

Reduction

Our "reduction" of $\bar{D}$ to $ACCEPT$ does not fit this. It was from $\bar{D}$ to $ACCEPT^C$

We use a simple notion of reduction (for most part).
Algorithm for solving L_1 should behave as follows:

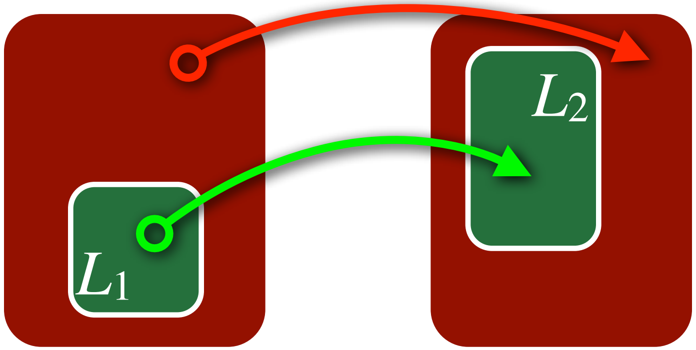
On input w, compute $f(w)$
Accept iff $f(w) \in L_2$



A (mapping) reduction from L_1 to L_2:
a computable function f s.t. $\forall w, w \in L_1 \Leftrightarrow f(w) \in L_2$

Reduction

A (mapping) reduction from L_1 to L_2:
a computable function f s.t. $\forall w, w \in L_1 \Leftrightarrow f(w) \in L_2$



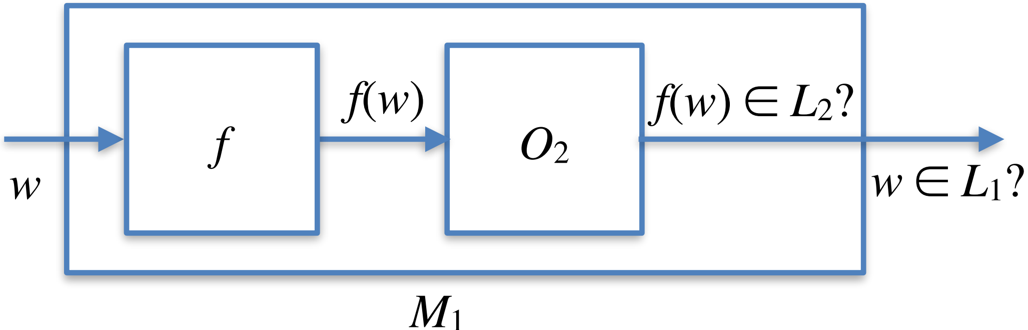
Note: a reduction from L_1 to L_2
is also a reduction from $\bar{L}_1$ to $\bar{L}_2$

$$L_1 \leq L_2 \Leftrightarrow \bar{L}_1 \leq \bar{L}_2$$

Reduction

A (mapping) reduction from L_1 to L_2:
a computable function f s.t. $\forall w, w \in L_1 \Leftrightarrow f(w) \in L_2$

On input w, compute $f(w)$
Accept iff $f(w) \in L_2$



Positive implication:
If $L_1 \leq L_2$ then: can "solve" $L_2 \Rightarrow$ can "solve" L_1

L_2 decidable $\Rightarrow$ L_1 decidable
L_2 recognizable $\Rightarrow$ L_1 recognizable

Negative implication: If $L_1 \leq L_2$ then:
L_1 undecidable $\Rightarrow$ L_2 undecidable
L_1 unrecognizable $\Rightarrow$ L_2 unrecognizable

Halting Problem

$HALT = \{ (z,w) \mid M_z$ halts on input $w \}$

Claim: $ACCEPT \leq HALT$

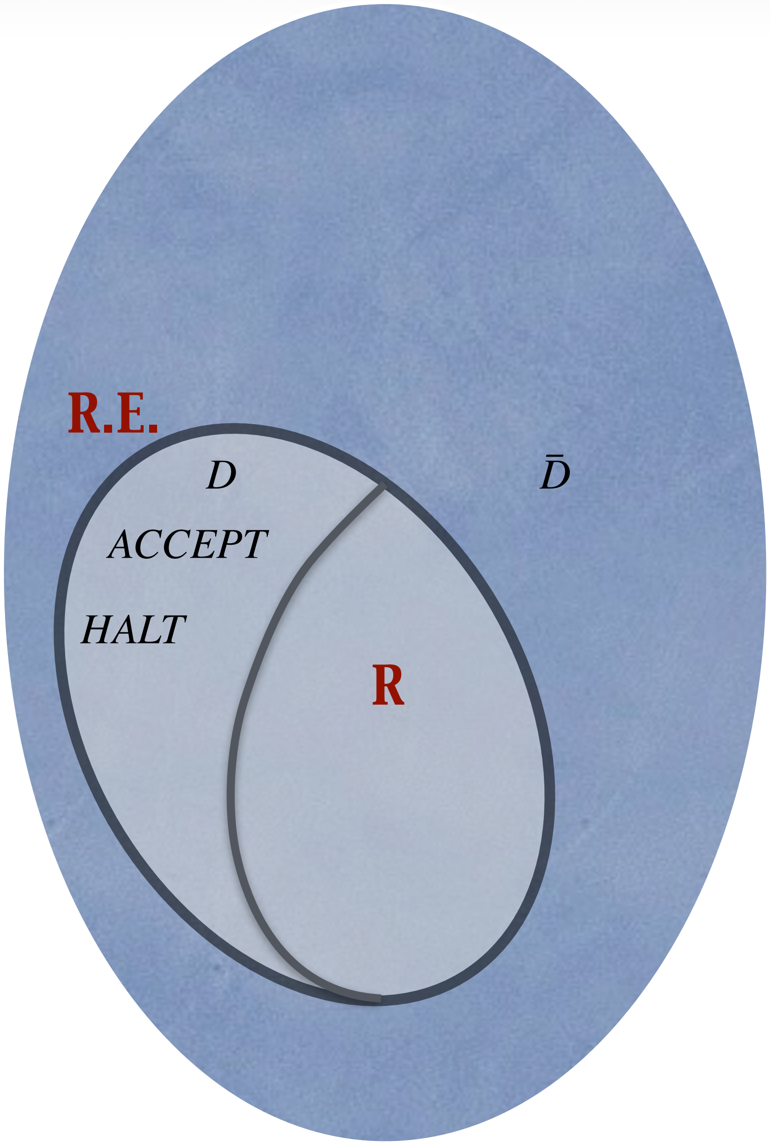
$f(z,w) = (z',w)$ where $M_{z'}$ behaves as follows:

On input x, run M_z on x.
If M_z halts rejecting x, go into an infinite loop.
If M_z halts accepting x, halt (and say, accept).

$(z',w) \in HALT \Leftrightarrow (z,w) \in ACCEPT$

$ACCEPT$ undecidable $\Rightarrow$ $HALT$ undecidable

Map



Complement & Undecidability

$ACCEPT$ is undecidable, but is recognizable (why?)

$ACCEPT^C$ is undecidable too (why?)

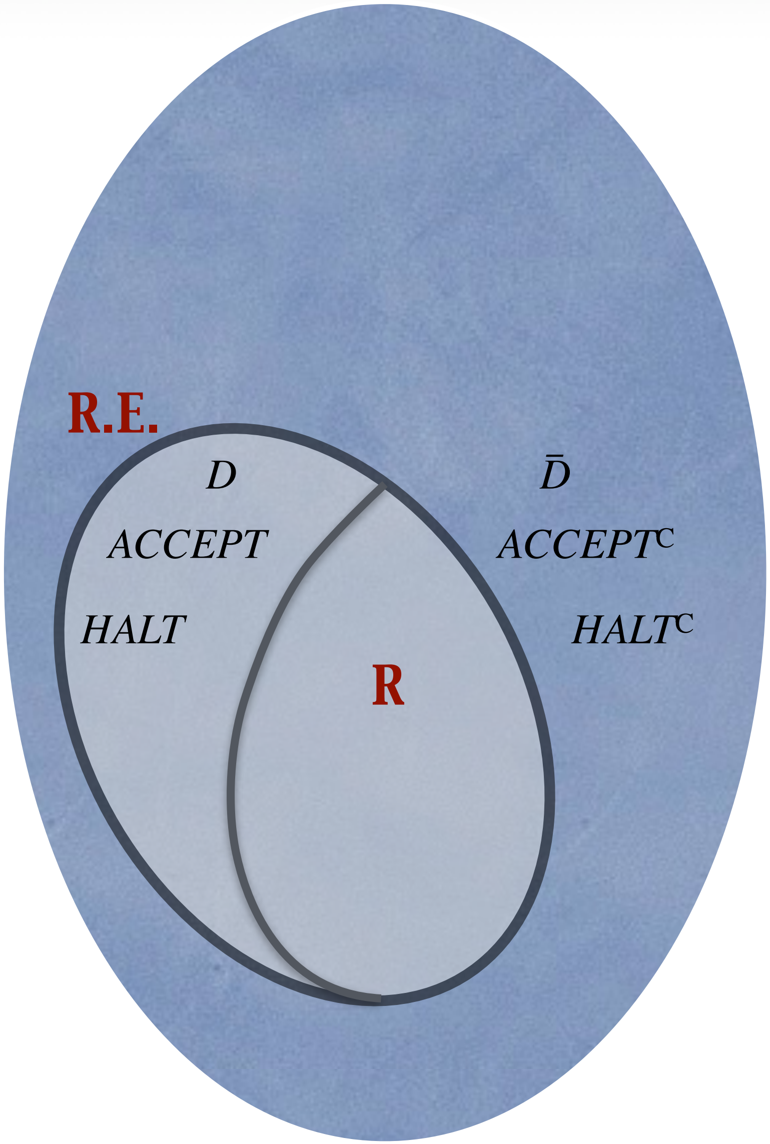
L^C stands for $\bar{L}$

Is $ACCEPT^C$ recognizable?

Claim: $ACCEPT^C$ is not recognizable

If not, $ACCEPT$ and $ACCEPT^C$ both recognizable,
Then $ACCEPT$ would be decidable! (why?)

Map



Empty Language Problem

$$EMPTY = \{ \ z \mid L(M_z) = \emptyset \ \}$$

Claim: $ACCEPT^C \leq EMPTY$

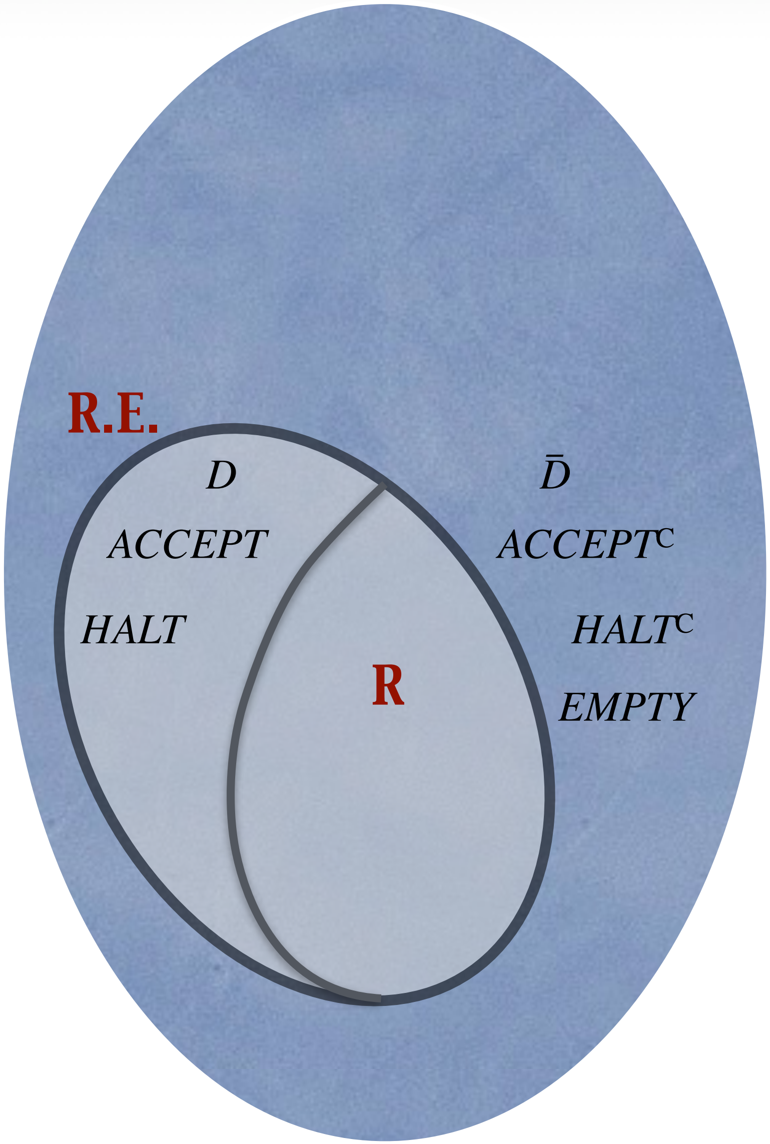
$f(z,w) = z'$ where $M_{z'}$ behaves as follows:

On input x, run M_z on w.
If M_z halts rejecting w, reject x.
If M_z halts accepting w, accept x.

$$z' \in EMPTY \Leftrightarrow (z,w) \notin ACCEPT$$

$ACCEPT^C$ unrecognizable $\Rightarrow$ $EMPTY$ is unrecognizable

Map



Dovetailing

Claim: $EMPTY^C = \{ z \mid L(M_z) \neq \emptyset \}$ is recognizable

$EMPTY^C = \{ z \mid \exists w \; M_z \text{ accepts } w \}$.

Given z, how to check if there is some w that M_z accepts?

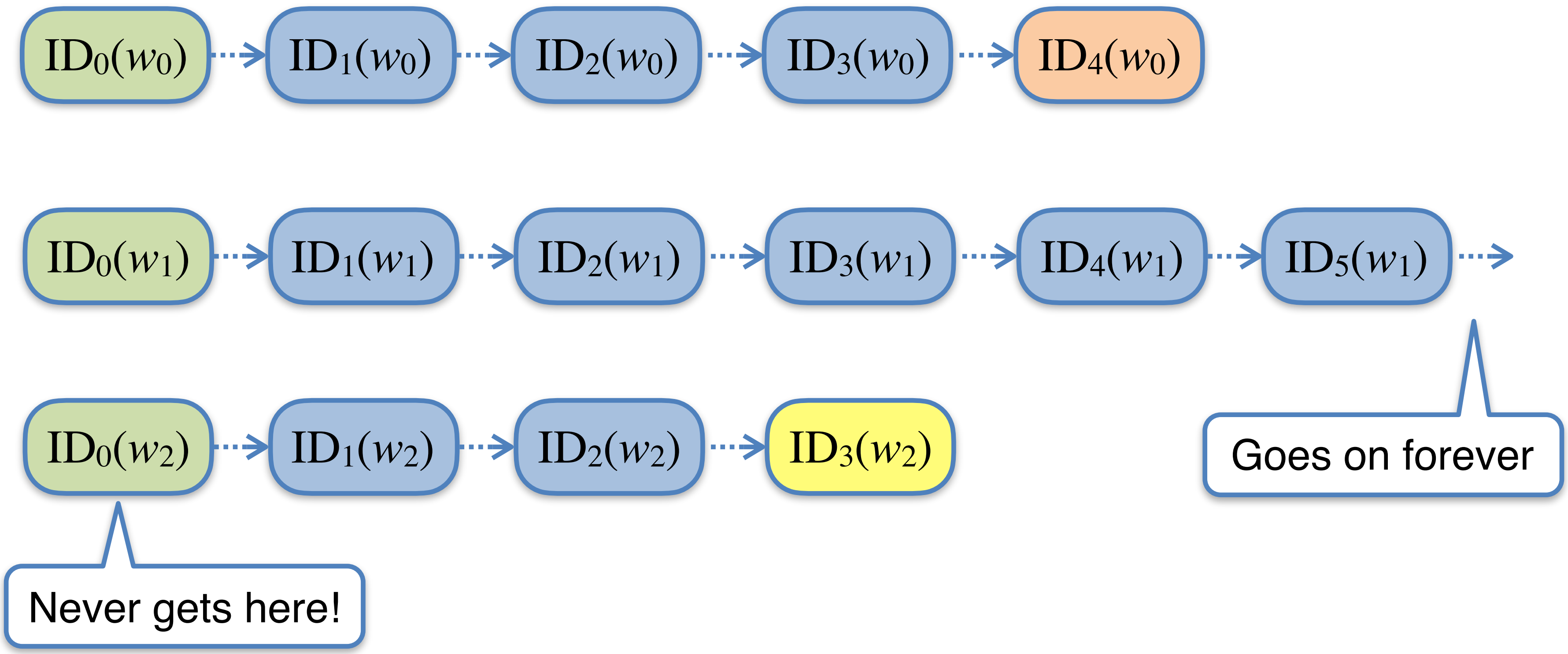
Run M_z on all w, and if it accepts any, accept (if not keep trying)

In "parallel"? Can't run infinitely many executions in parallel!

Solution: increasingly more
executions in parallel

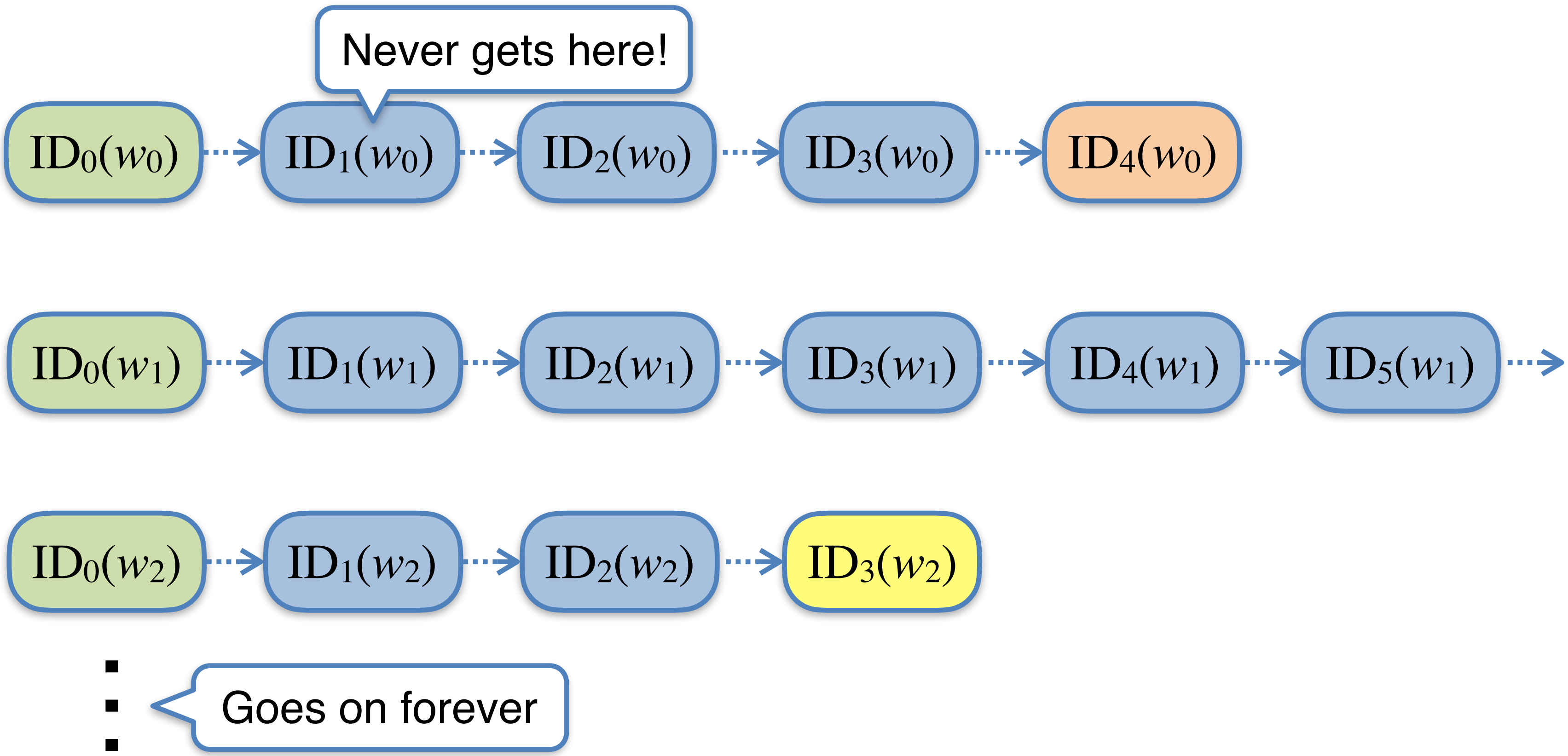
Exploring the ID Graph

Sequential Simulation: Depth first



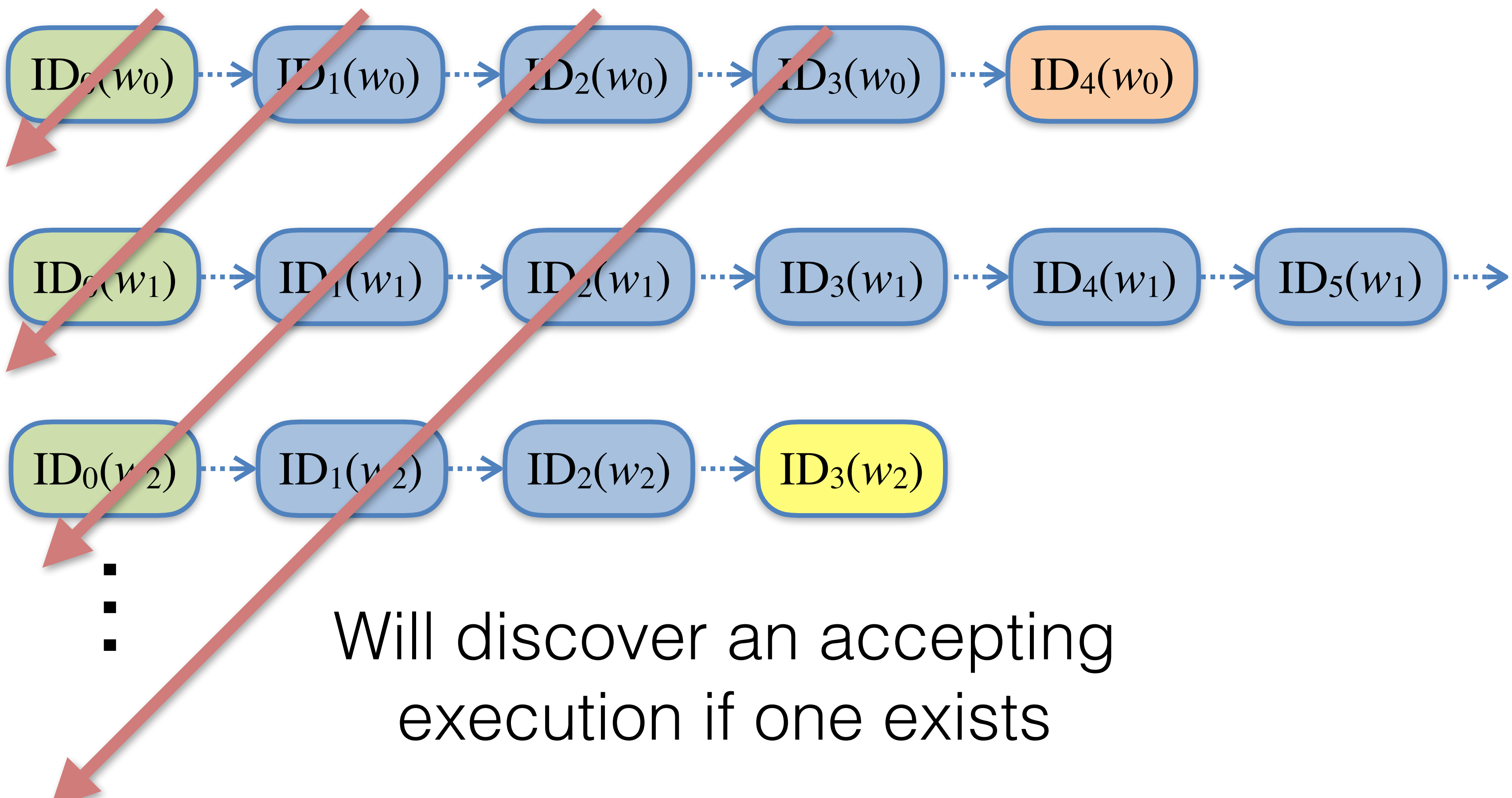
Exploring the ID Graph

Parallel Simulation: Breadth first



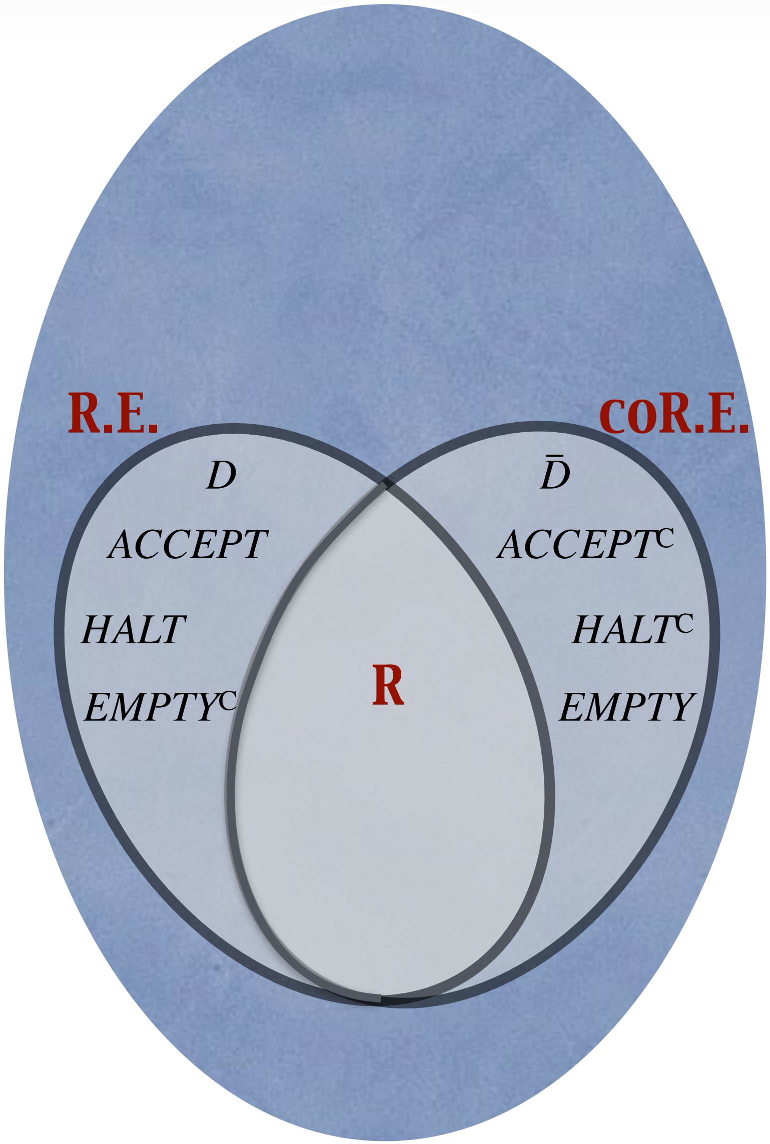
Dovetailing

Explore increasingly more executions for increasingly more steps



Will discover an accepting execution if one exists

Map



Language Equality Problem

$$EQUAL = \{\ (z, z') \mid L(M_z) = L(M_{z'})\ \ \}$$

Claim: $EMPTY \leq EQUAL$

$f(z) = (z, z')$ where $M_{z'}$ rejects all inputs

$(z, z') \in EQUAL \Leftrightarrow z \in EMPTY$

$EMPTY$ unrecognizable $\Rightarrow$ $EQUAL$ unrecognizable

Language Equality Problem

$$EQUAL = \{ (z, z') \mid L(M_z) = L(M_{z'}) \}$$

Claim: $ACCEPT \leq EQUAL$

$ACCEPT^C \leq EQUAL^C$

$f(z,w) = (z_1,z_2)$ where M_{z_1} & M_{z_2} behave as follows:

M_{z_1} accepts all strings. i.e., $L(M_{z_1}) = \Sigma^*$

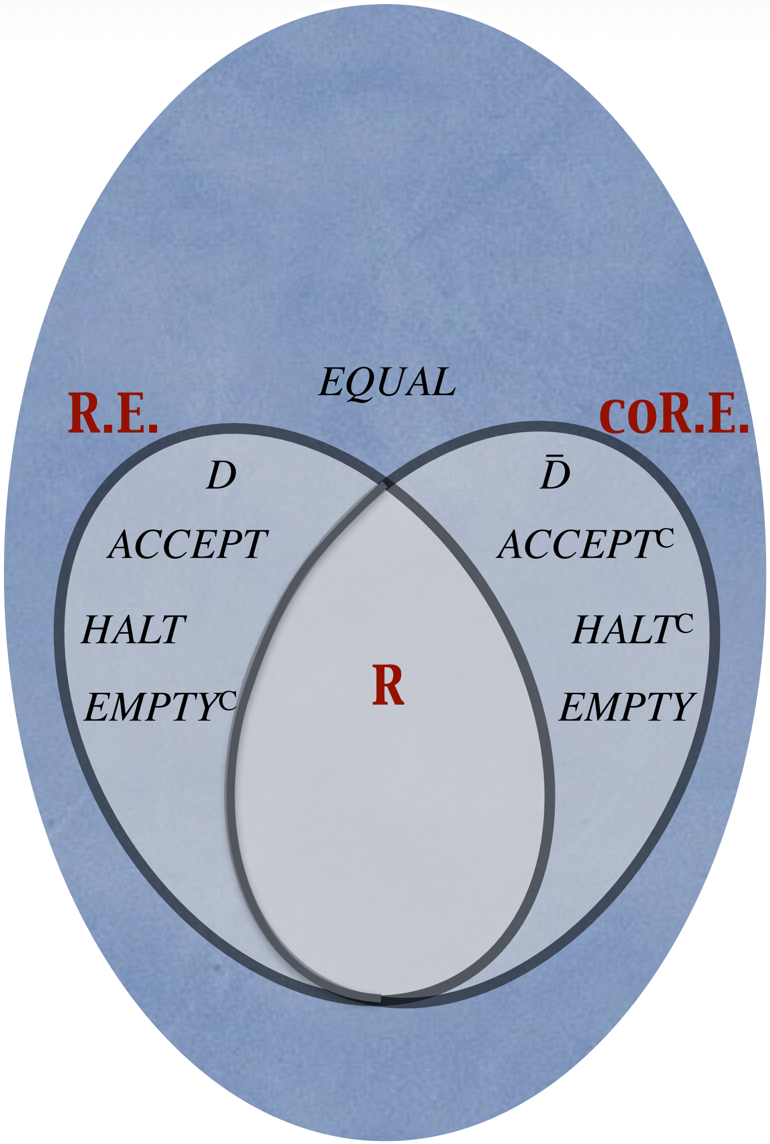
M_{z_2} runs M_z on w and if it accepts, accepts its input

$$(z_1,z_2) \in EQUAL \Leftrightarrow (z,w) \in ACCEPT$$

Hence $EQUAL$ is not decidable.

Also, $EQUAL^C$ is not recognizable. (Why?)

Map


EQUAL
R.E.
coR.E.
D
ACCEPT
HALT
EMPTY^C
$\bar{D}$
ACCEPT^C
HALT^C
EMPTY
R

Post Correspondence Problem

<u>Theorem</u> [Post'46]: *HALT* reduces to *PostCP*

— a "combinatorial" problem

<u>*PostCP* is undecidable.</u>

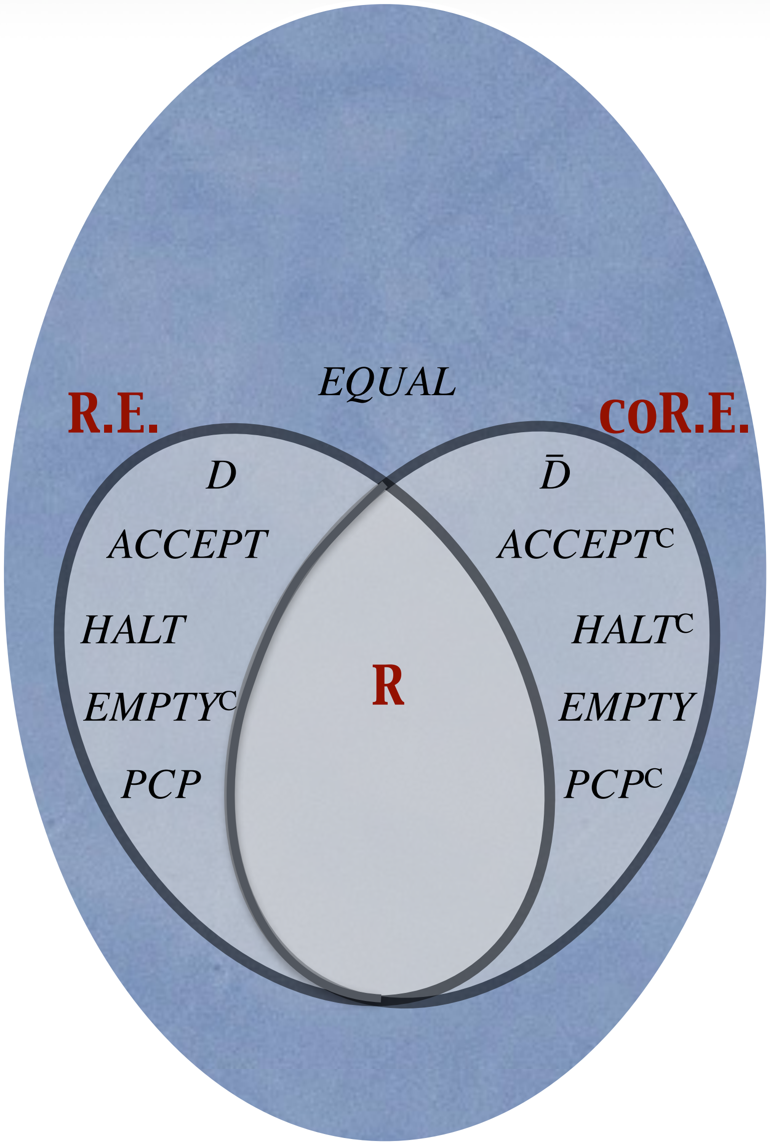
Given: Dominoes, each with a top-word and a bottom-word

b	**ba**	**abb**	**abb**	**a**
bbb	**bbb**	**a**	**baa**	**ab**

Can one arrange them (using <u>any number of copies of each type</u>) so that the top and bottom strings are identical?

abb	**ba**	**abb**	**a**	**abb**	**b**
a	**bbb**	**a**	**ab**	**baa**	**bbb**

Map



EQUAL
R.E.
coR.E.
D
ACCEPT
HALT
EMPTY^C
PCP
D̄
ACCEPT^C
HALT^C
EMPTY
PCP^C
R

Recap

- If $L_1 \le L_2$ then:
  - If L_1 is undecidable, so is L_2
  - If L_1 is unrecognizable, so is L_2
  - $\bar{L}_1 \le \bar{L}_2$
- L and $\bar{L}$ recognizable $\Leftrightarrow$ L and $\bar{L}$ decidable $\Leftrightarrow$ L decidable
  - Corollary: If L recognizable but undecidable, then $\bar{L}$ not recognizable
  - e.g., $ACCEPT^C$ is not recognizable
- e.g.: If $ACCEPT \le L$, then $\bar{L}$ not recognizable (Why?)
- If L is recognizable, then so is $L' = \{ x \mid \exists w, (x,w) \in L \}$ (via dovetailing)