CSC 240

REDUCIBILITY

ACCEPTANCE PROBLEMS

We can build a Turing Machine that tells us:

- If a DFA will accept a string.
- If an NFA will accept a string.
- If a Regular Expression will accept a string.
- If a CFG describes a string.
- If a PDA accepts a string.

Decidable

We can't build a Turing Machine that tells us:

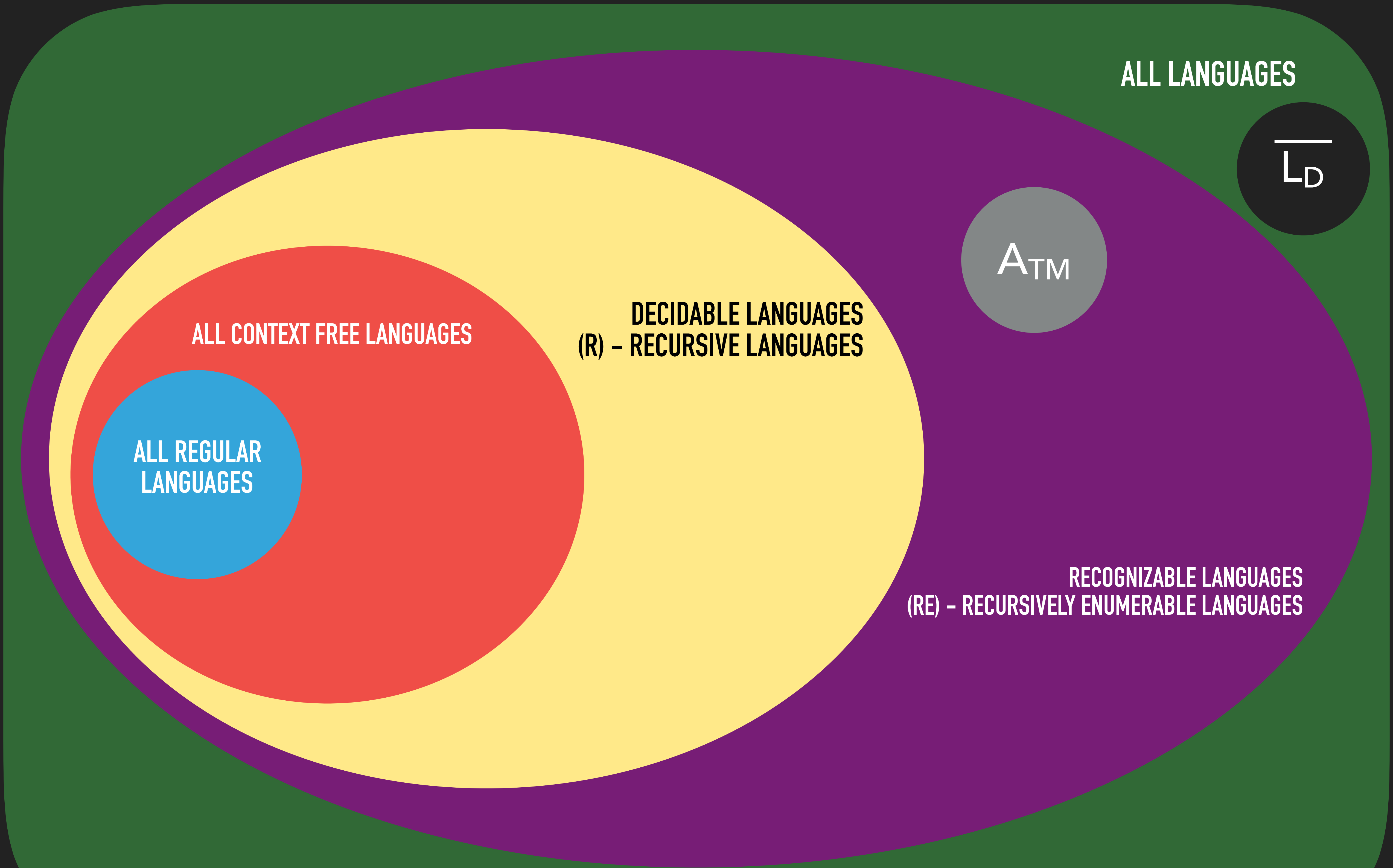
- If a TM will halt (accept or reject) on a string.

Not Decidable

All of these problems (languages) are *recognizable* by at least one TM.

Are there problems (languages) *not recognizable* by any TM?

LANGUAGES



How can we determine where a language fits?

A language is "Turing Recognizable" if a Turing machine recognizes it.

A language is "Turing Decidable" if a Turing machine decides it, that is if the strings of that language cause the Turing machine to end up in an accept or reject state.

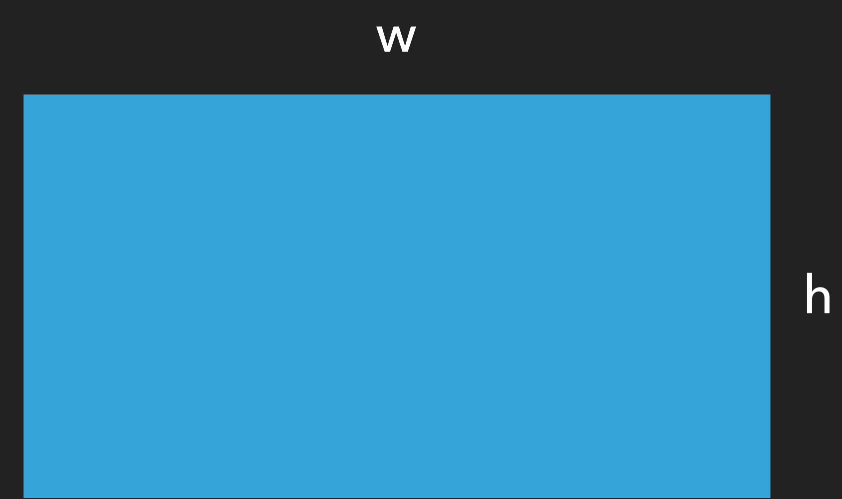
$L \in R$: We can write a computer program that “solves” problems expressed by L. That is, we can decide conclusively whether an arbitrary input in that language should be accepted or rejected.

$L \in RE$: We cannot write a computer program that “solves” problems expressed by L. That is, we can’t decide conclusively whether an arbitrary input in that language should be accepted or rejected.

However, RE languages are recognizable, meaning that given an answer that we know to be correct for a given problem in RE, there is an algorithmic way to prove it.

REDUCIBILITY

Problem A: Calculate the area of this rectangle:



Problem B: Calculate the product of these two numbers:

w x h

Which problem is harder to solve?

If we can take a problem, A, and convert it to another form, B, then if B has a solution, so does A.

1. If A is reducible to B, and B is decidable, then A is decidable.
2. If A is reducible to B, and A is undecidable, then B is undecidable.

$HALT_{TM}$ = { <M, w> | M is a TM and M halts on string w }

Is $HALT_{TM}$ decidable?

NO

Proof by contradiction:

1. Let's assume that $HALT_{TM}$ is decidable.
2. That means we can construct a TM, HALT-DECIDER, which decides it.
3. This implies that we could build another TM, S, such that:

What did we just build?

3a. If you give S the input <M, w>, where M is a Turing Machine and w an input string

3b. S runs HALT-DECIDER on <M,w>, HALT-DECIDER must either accept or reject.

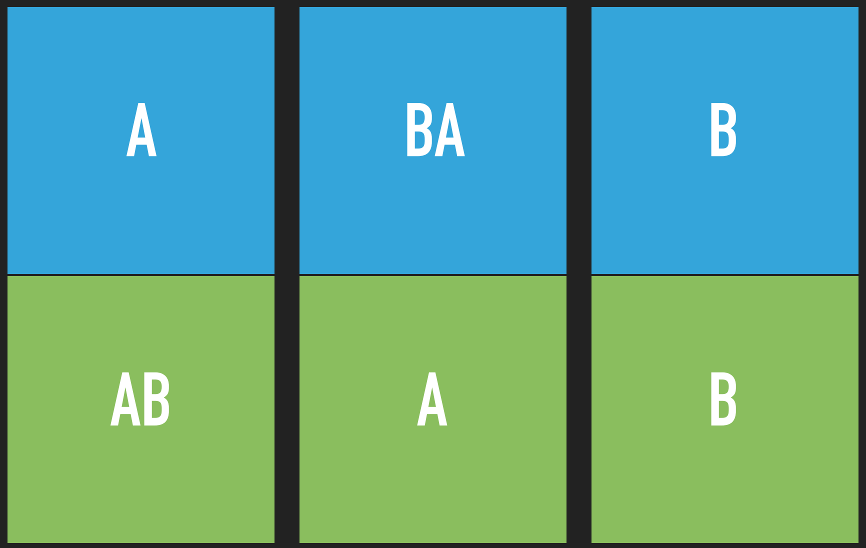
3c. If HALT-DECIDER rejects <M,w>, then S rejects.

3d. If HALT-DECIDER accepts <M,w>, then M must halt, so run M on w until it halts.

3e. If M accepts w, then S accepts. If M rejects w, then S rejects.

4. That would meant that A_{TM} can be solved by $HALT_{TM}$. In other words, A_{TM} is reducible to $HALT_{TM}$.

5. But since A_{TM} is undecidable, $HALT_{TM}$ is also undecidable, which is a contradiction.



Can we arrange this set of dominos in a way (including repeats) so that the string on top is the same as the string on bottom?

What about this set?

THE POST CORRESPONDENCE PROBLEM

A PCP instance is an arbitrary set of dominos, P:

$$P = \left\{ \left[\frac{T_1}{B_1}\right], \left[\frac{T_2}{B_2}\right], \dots, \left[\frac{T_K}{B_K}\right] \right\}$$

Let Match be defined as a sequence of length L: $i_1, i_2, \dots i_L$

where $t_1 t_2 \dots t_L = b_1 b_2 \dots b_L$

Then PCP = { <P> | P is an instance of PCP with a match }.

Is PCP Decidable?

In other words, can we construct a Turing Machine that will tell us if an arbitrary P contains a match? NO (see pages 228 - 233 for the proof)

THE POST CORRESPONDENCE PROBLEM

A PCP instance is an arbitrary set of dominos, P:

$$P = \left\{ \left[\frac{T_1}{B_1}\right], \left[\frac{T_2}{B_2}\right], \ldots, \left[\frac{T_K}{B_K}\right] \right\}$$

Let Match be defined as a sequence of length L: $i_1, i_2, \ldots i_L$

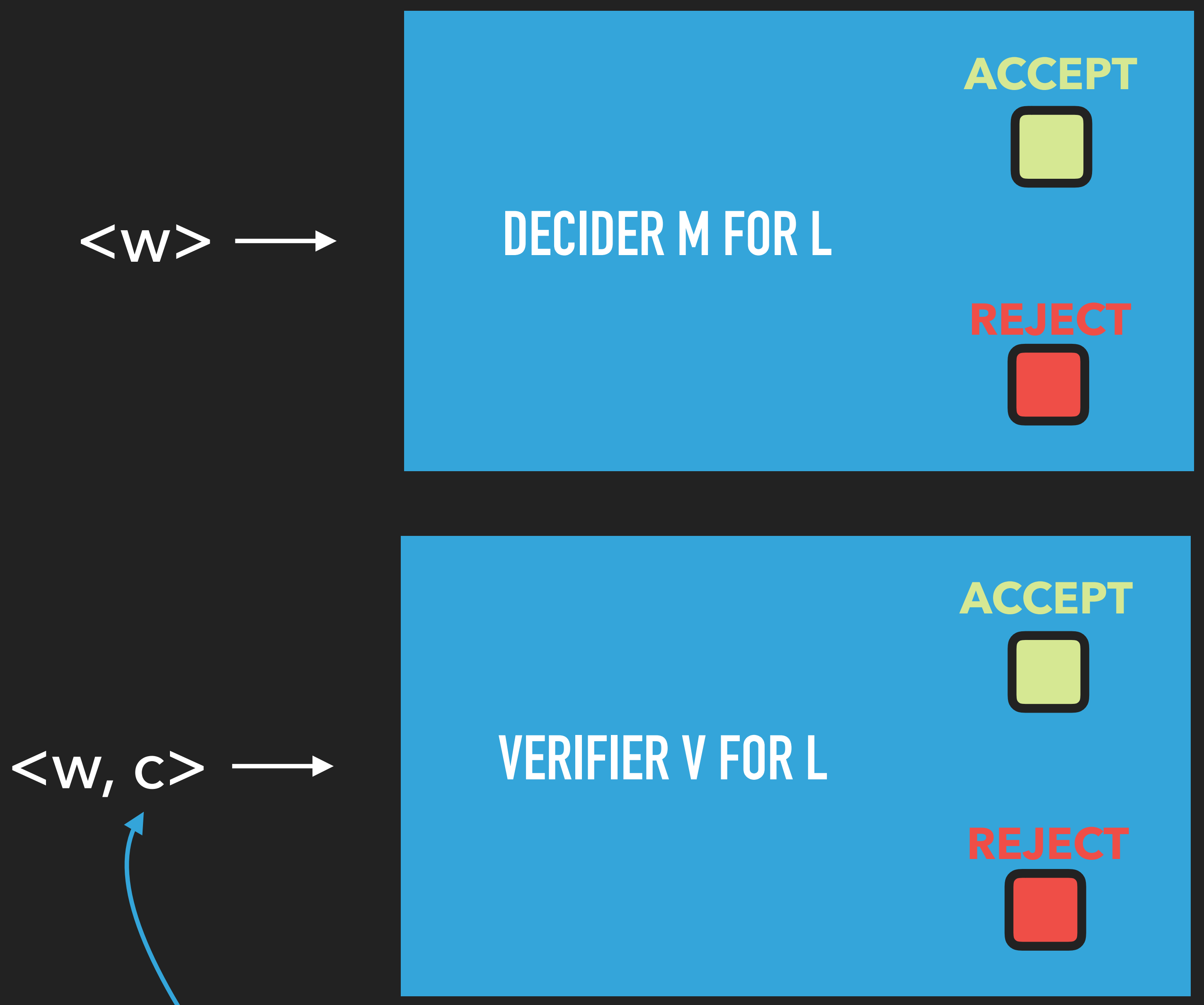
where $t_1 t_2 \ldots t_L = b_1 b_2 \ldots b_L$

Then PCP = { <P> | P is an instance of PCP with a match }.

Is PCP Verifiable?

In other words, can we construct a Turing Machine that given P and an arrangement, w, of the members of P can tell us if w is a valid match?

VERIFIER



If M accepts w, then $w \in L$

If M rejects w, then $w \notin L$

If V accepts <w, c>, then $w \in L$

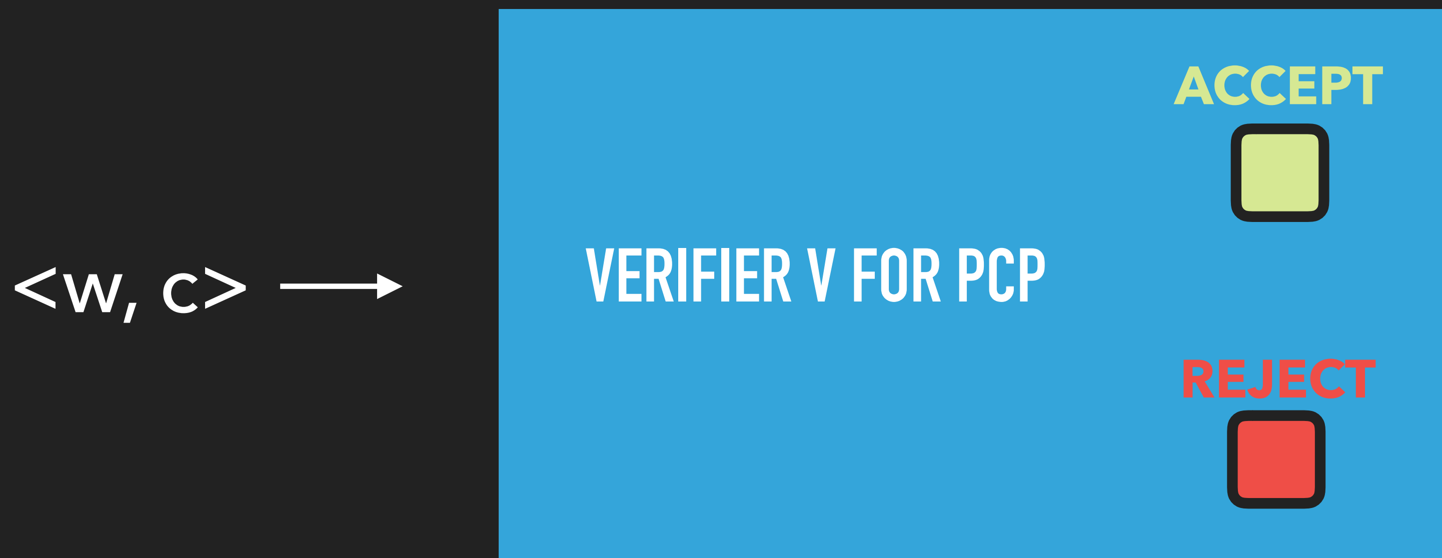
If V rejects <w, c>, then w's membership in L is still unknown.

The "certificate", something that attempts to prove that $w \in L$.

We can build a decider for any language in R.

We can build a verifier for any language in RE.

PCP = { <P> | P is an instance of PCP with a match }.



If V accepts <w, c>, then w ∈ L

If V rejects <w, c>, then w's membership in L is still unknown.

w: <an encoding of dominos>

c: <an encoding of an arrangement of dominos that form a match>

If c is a valid match, we know that w is member of PCP.

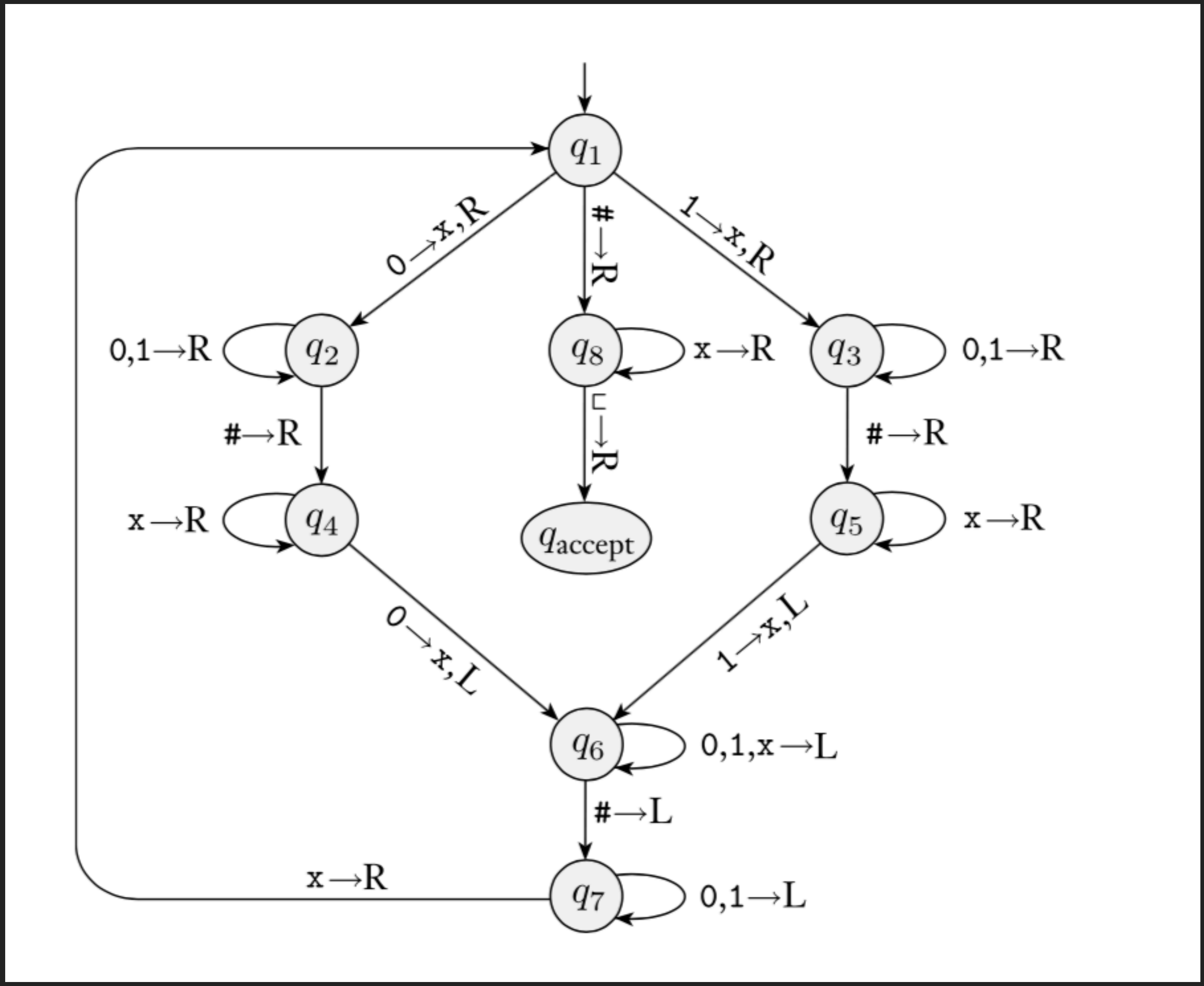
If c is not a valid match, this doesn't tell us anything about w, it might still have a match we don't know about.

A_{TM} = { <M, w> | M is a TM that accepts string w }

Is A_{TM} Verifiable?

In other words, given a Turing Machine, M, an input string, w, and some kind of extra information, c, claiming to prove that M accepts w, can we verify that claim?

VERIFIER



$L(M) = \{ x\#x \mid x \in \{0,1\}^* \}$

w = 001#001

c = ???

What extra information could we use to attempt to prove that w is accepted by M?