Decidability

Lin Chen

Email: Lin.Chen@ttu.edu

TEXAS TECH UNIVERSITY

Decidability

We have learned the language that can and cannot be recognized by DFA/NFA and PDA, it is time to learn the possibility and limitations of a Turing machine

Decidability

$A_{\mathsf{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}.$

Is this language decidable?

```
M = "On input ⟨B, w⟩, where B is a DFA and w is a string:
        1. Simulate B on input w.
        2. If the simulation ends in an accept state, accept. If it ends in a
           nonaccepting state, reject."
```

Decidability

$A_{\mathsf{NFA}} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts input string } w\}.$

Is this language decidable?

Transform NFA to an equivalent DFA and then simulate the DFA.

Decidability

$A_{\text{REX}} = \{\langle R, w\rangle \mid R \text{ is a regular expression that generates string } w\}.$

Is this language decidable?

Transform a regular expression to an equivalent NFA.

Decidability

$E_{\mathsf{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}.$

Is this language decidable?

```
T = "On input ⟨A⟩ where A is a DFA:
    1. Mark the start state of A.
    2. Repeat until no new states get marked:
    3.     Mark any state that has a transition coming into it from any
           state that is already marked.
    4. If no accept state is marked, accept; otherwise, reject."
```

Decidability

$A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}.$

Is this language decidable?

$S =$ "On input $\langle G, w\rangle$, where G is a CFG and w is a string:

1. Convert G to an equivalent grammar in Chomsky normal form.
2. List all derivations with $2n - 1$ steps, where n is the length of w, except if $n = 0$, then instead list all derivations with 1 step.
3. If any of these derivations generate w, *accept*; if not, *reject*."

Decidability

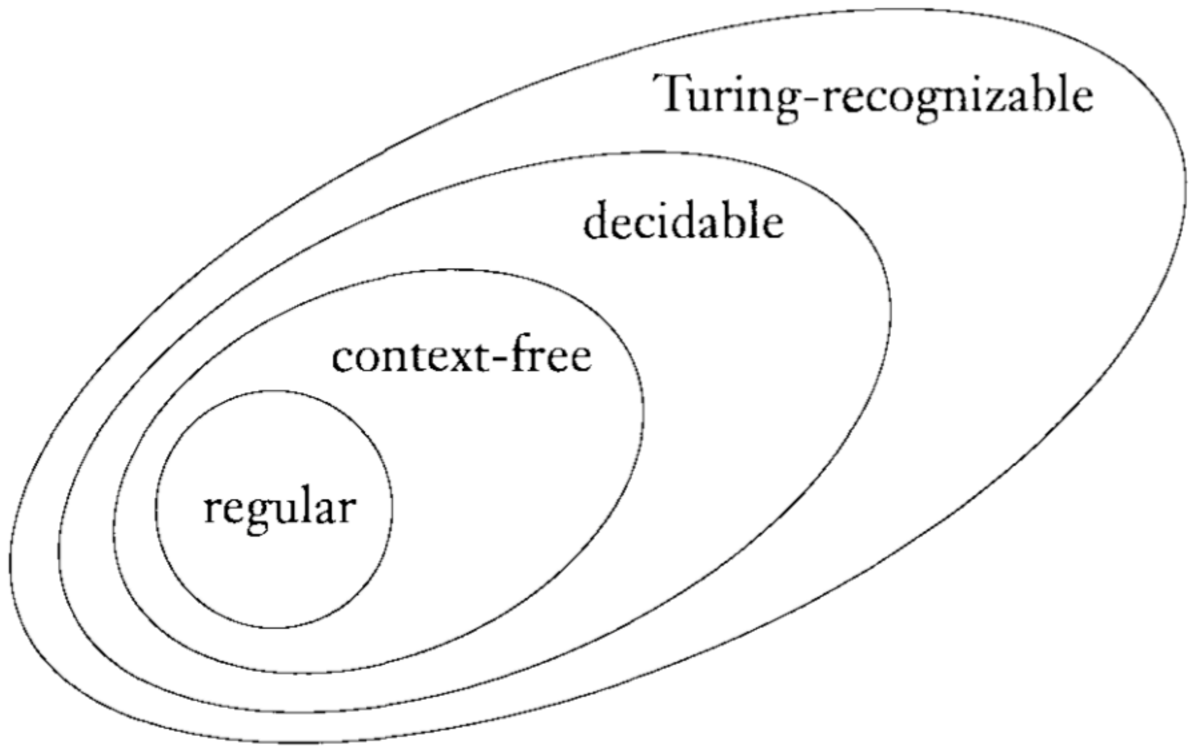
Every context-free language is decidable.

Is this language decidable?

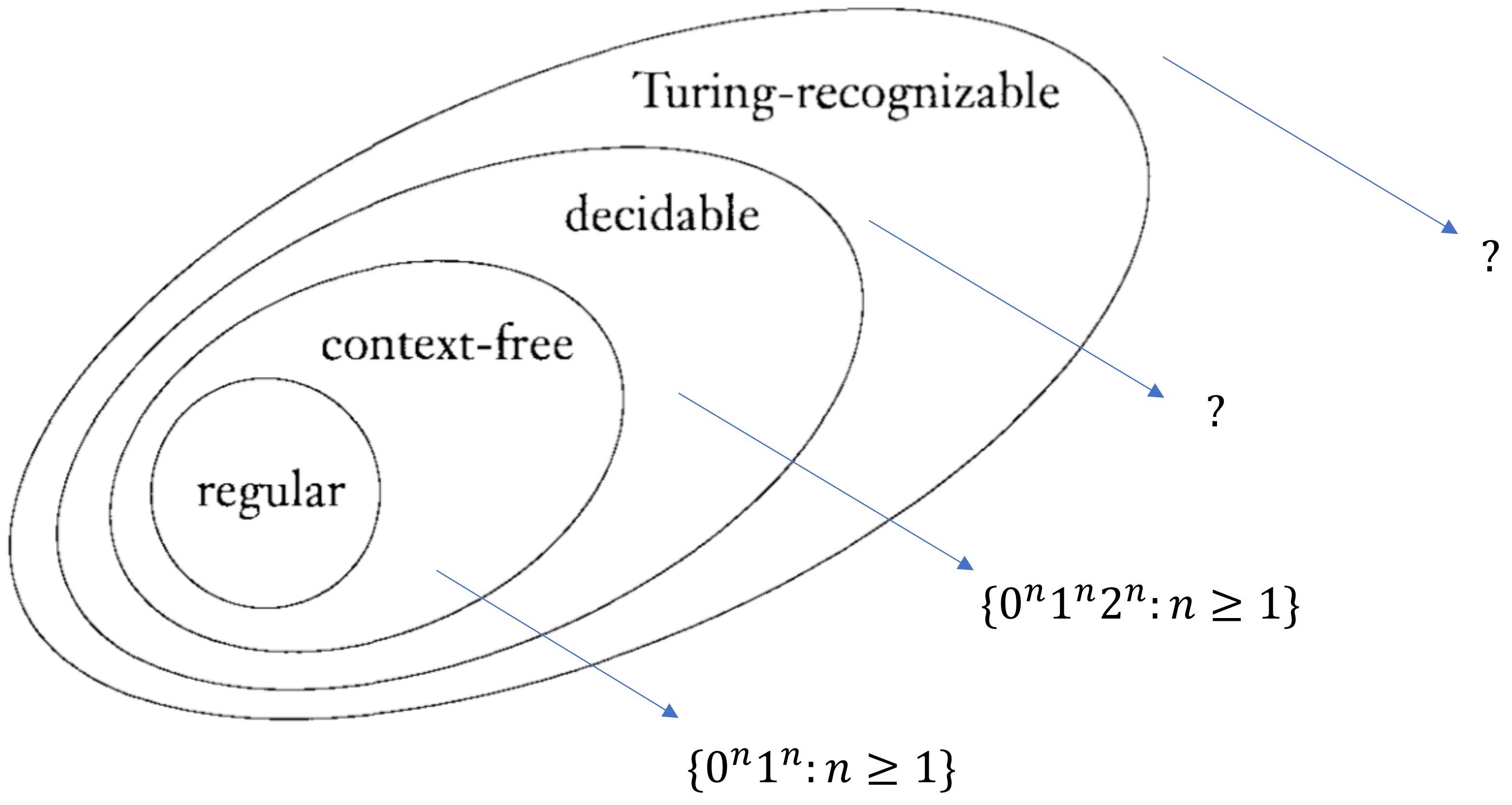
Every context-free language has a context-free grammar to generate it.

```
$M_G$ = "On input $w$:
    1. Run TM $S$ on input $\langle G, w \rangle$
    2. If this machine accepts, *accept*; if it rejects, *reject*."
```

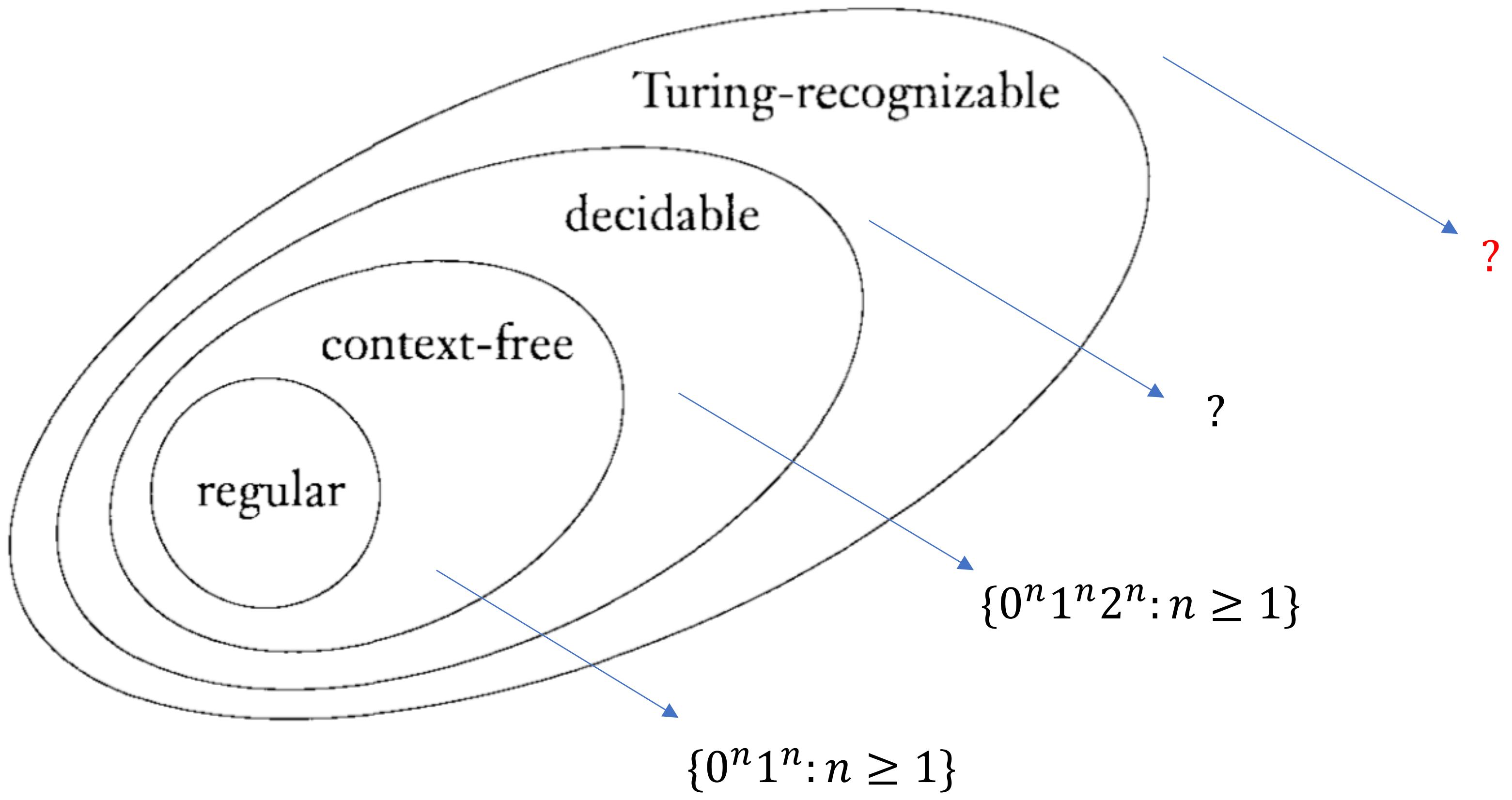
Decidability



Decidability



Decidability



Decidability

Recall: the proof that R the set of real numbers is uncountable

The set of all Turing machines is countable.

The set of all languages is uncountable.

There are some (infinitely manly) languages that are not Turing recognizable.

Decidability

Recall: the proof that R the set of real numbers is uncountable

The set of all Turing machines is countable.

- The set of all strings is countable (why?).
- Each Turing machine, by the 7-tuple representation, is a string.

The set of all languages is uncountable.

There are some (infinitely manly) languages that are not Turing recognizable.

Decidability

Recall: the proof that R the set of real numbers is uncountable

The set of all Turing machines is countable.

The set of all languages is uncountable.

List of all strings

$$
\begin{array}{rcccccccccccl}
\Sigma^* = \{ & \varepsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & \cdots & \} ; \\
A = \{ & & 0, & & 00, & 01, & & & 000, & 001, & \cdots & \} ; \\
\chi_A = & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \cdots & .
\end{array}
$$

Characteristic sequence of a language

The set of all infinite binary sequence is uncountable (why?)

Decidability

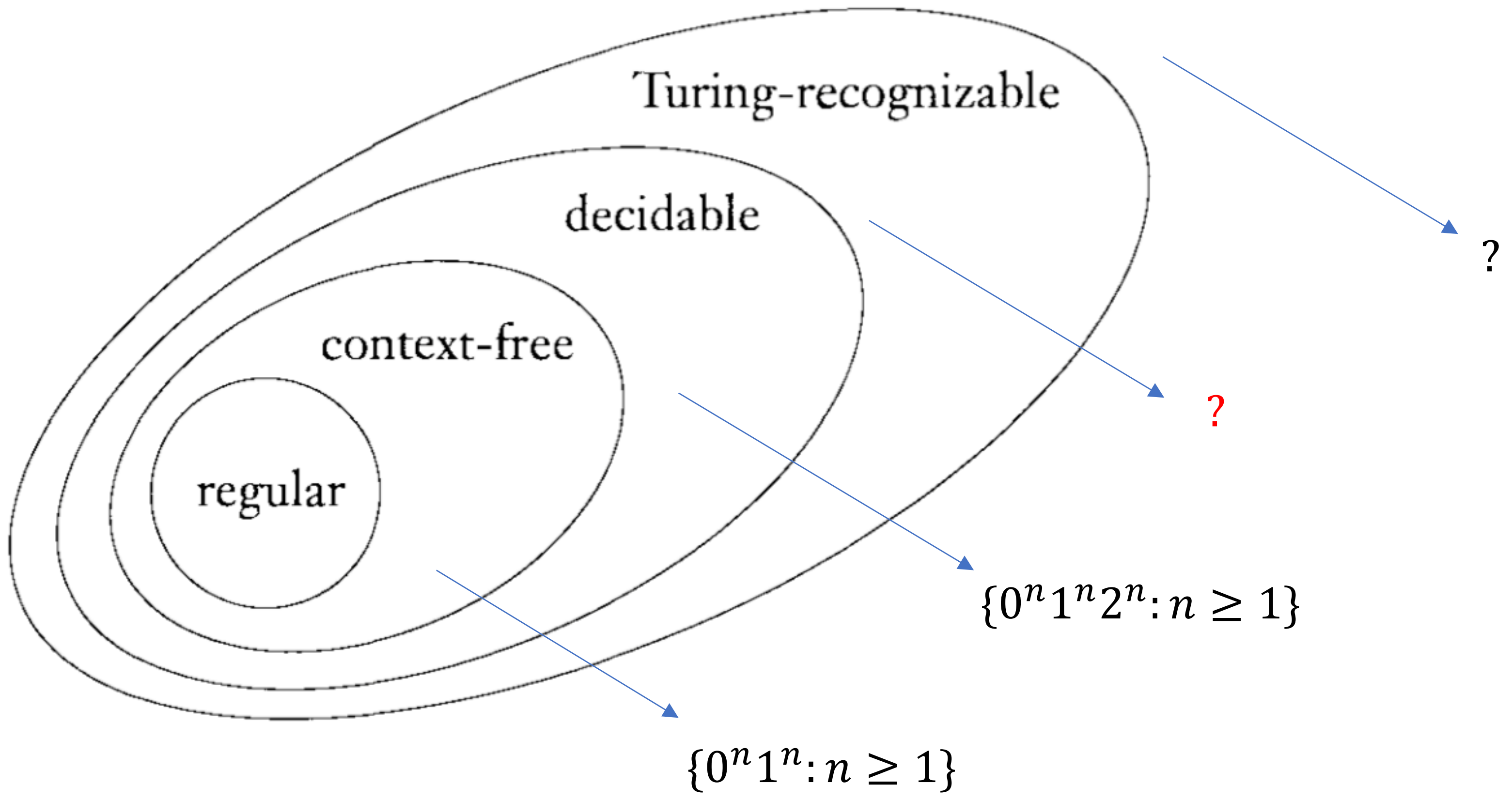
Recall: the proof that R the set of real numbers is uncountable

The set of all Turing machines is countable.

The set of all languages is uncountable.

There are some (infinitely manly) languages that are not Turing recognizable.

Decidability



Decidability

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

Is this language decidable?

```
U = "On input ⟨M, w⟩, 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."
```

The TM machine U only recognizes but not decides (why?)

Decidability

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

Is this language decidable?

Suppose it is decidable, let H be decider.

$$H(\langle M, w\rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ accepts } w \\ \textit{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

Decidability

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

Is this language decidable?

Suppose it is decidable, let H be decider.

$$H(\langle M, w\rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ accepts } w \\ \textit{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

```
D = "On input ⟨M⟩, where M is a TM:
    1. Run H on input ⟨M, ⟨M⟩⟩.
    2. Output the opposite of what H outputs; that is, if H accepts,
       reject and if H rejects, accept."
```

What happens if the input is D?

Decidability

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

Suppose it is decidable, let H be decider.

$$H(\langle M, w\rangle) = \begin{cases} \mathit{accept} & \text{if } M \text{ accepts } w \\ \mathit{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

D = "On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs; that is, if H accepts, reject and if H rejects, accept."

What happens if the input is D?

$$D(\langle D\rangle) = \begin{cases} \mathit{accept} & \text{if } D \text{ does not accept } \langle D\rangle \\ \mathit{reject} & \text{if } D \text{ accepts } \langle D\rangle. \end{cases}$$

Decidability

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

Suppose it is decidable, let H be decider.

$$H(\langle M, w\rangle) = \begin{cases} \mathit{accept} & \text{if } M \text{ accepts } w \\ \mathit{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

$D =$ "On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs; that is, if H accepts, *reject* and if H rejects, *accept*."

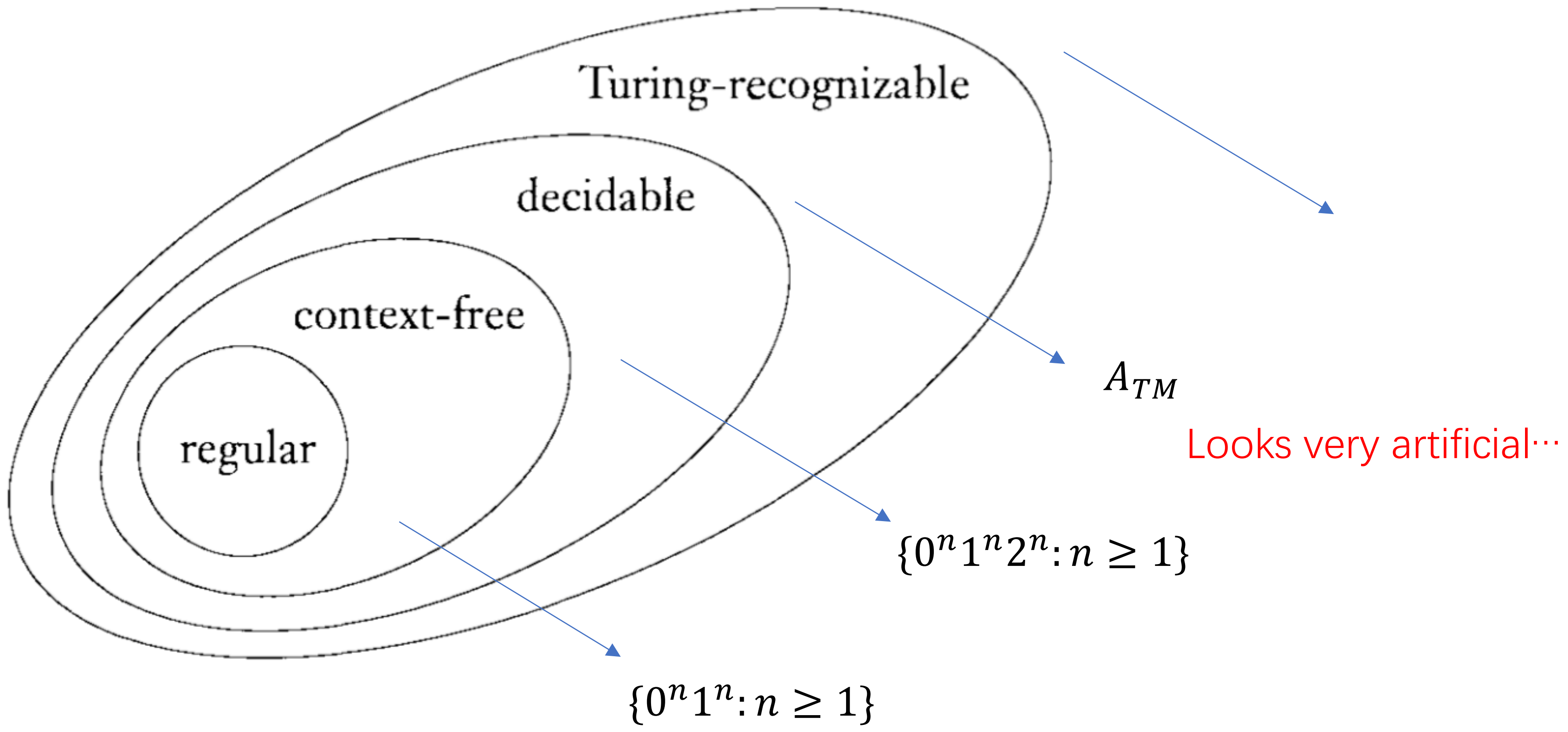
What happens if the input is D?

$$D(\langle D\rangle) = \begin{cases} \mathit{accept} & \text{if } D \text{ does not accept } \langle D\rangle \\ \mathit{reject} & \text{if } D \text{ accepts } \langle D\rangle. \end{cases}$$

But, what does it mean by a Turing machine taking itself as an input?

It is not really something very surprising, e.g., there is a program for searching keyword. The program itself is a sequence of code (strings), and can be used as an input to itself.

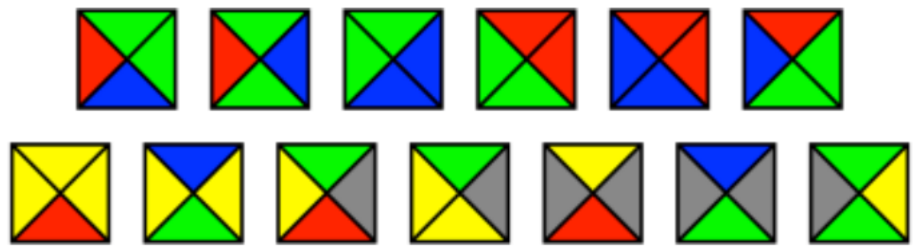
Decidability



Decidability

Wang Tile Problem (by Hao Wang '61)

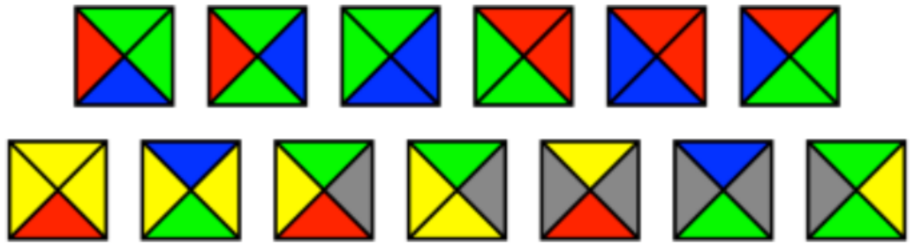
A set of square tiles, with each tile edge of a fixed color, are arranged side by side in a rectangular grid. All four edges of each tile must 'match' (have the same color as) their adjoining neighbor.

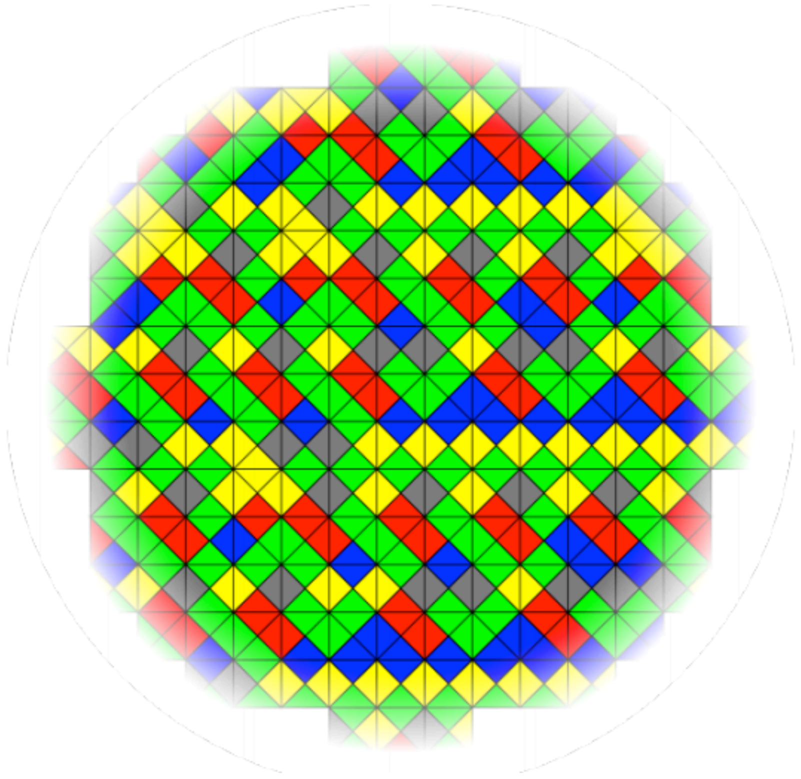


Decidability

Wang Tile Problem (by Hao Wang '61)

Given a set of Wang tiles, is it possible to tile the infinite plane with them ?





Decidability

Wang Tile Problem (by Hao Wang '61)

Given a set of Wang tiles, is it possible to tile the infinite plane with them ?

Have you played similar games?

Decidability

Wang Tile Problem (by Hao Wang '61)

Conjecture (1961, Wang)

If a finite set of Wang tiles can tile the plane, then it can tile the plane periodically.

Decidability

Wang Tile Problem (by Hao Wang '61)

Conjecture (1961, Wang)

If a finite set of Wang tiles can tile the plane, then it can tile the plane periodically.

If Wang's conjecture is true, there exists an algorithm to decide whether a given finite set of Wang tiles can tile the plane. (By Konig's infinity lemma)

Decidability

Wang Tile Problem (by Hao Wang '61)

Conjecture (1961, Wang)

If a finite set of Wang tiles can tile the plane, then it can tile the plane periodically.

Wang's student, Robert Berger, gave a negative answer to the Domino Problem, by reduction from the Halting Problem. It was also Berger who coined the term "Wang Tiles".

Theorem (1966, Robert Berger, Memoris of AMS)

Domino problem is undecidable.

Decidability

Wang Tile Problem (by Hao Wang '61)

Theorem (2015, Emmanuel Jeandel and Michael Rao)

There exists an aperiodic set containing 11 Wang tiles. Moreover, there is no aperiodic set of Wang tiles with 10 tiles or less, and there is no aperiodic set of Wang tiles with less than 4 colors.

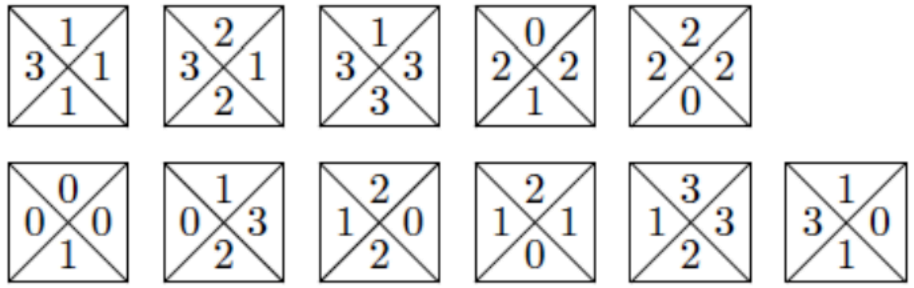
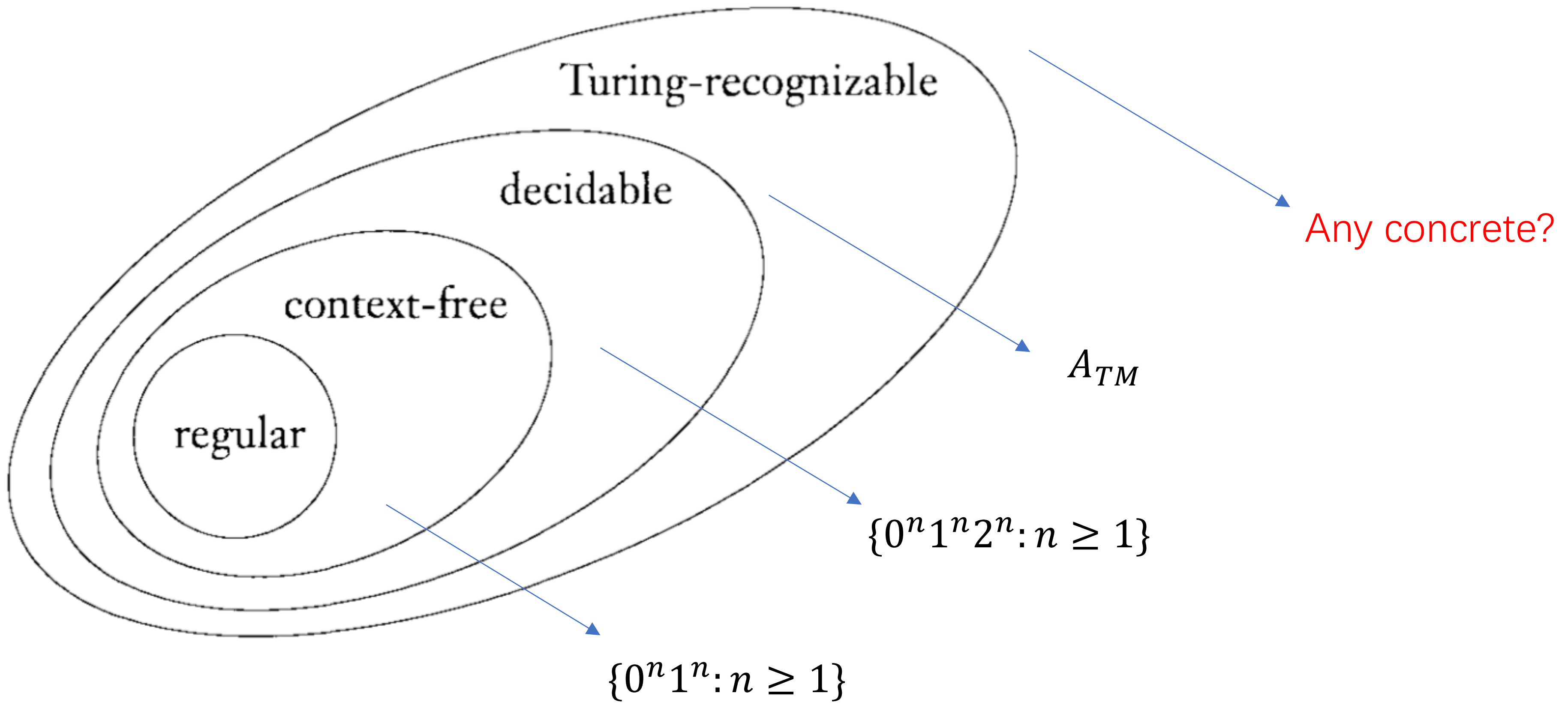
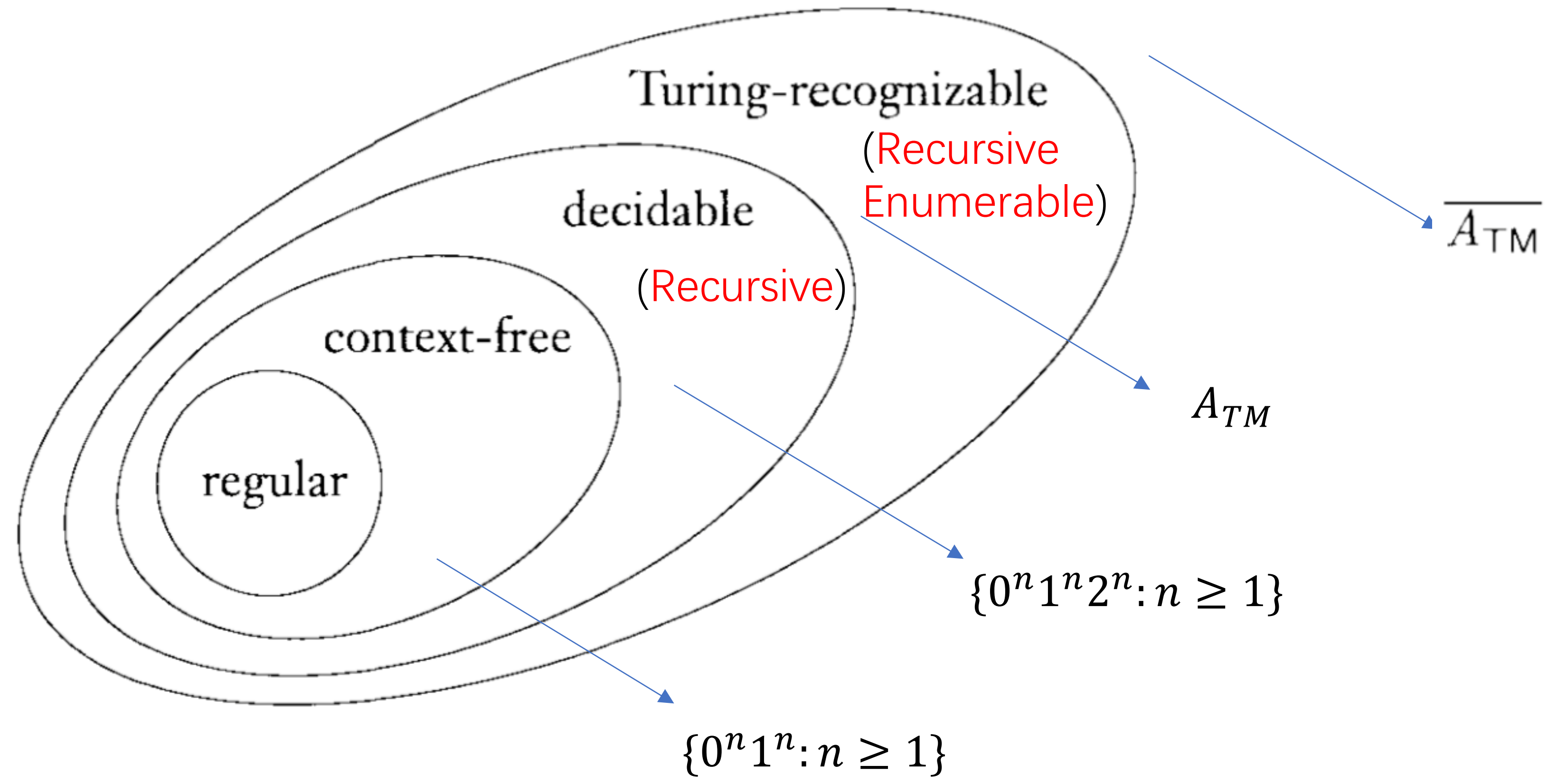


Figure 4: Wang set $\mathcal{T}'$.

Decidability



Decidability



Decidability

A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.

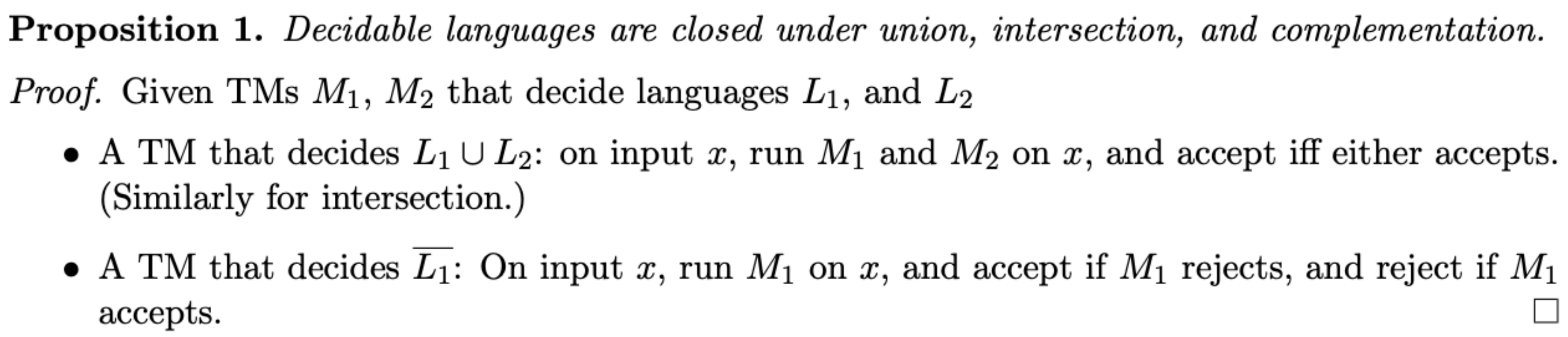
Let A be Turing-recognizable (M_1) and $\bar{A}$ be also Turing-recognizable (M_2)

```
$M$ = "On input $w$:
    1. Run both $M_1$ and $M_2$ on input $w$ in parallel.
    2. If $M_1$ accepts, *accept*; if $M_2$ accepts, *reject*."
```

Closure property

Recursive (Turing decidable) language is closed under:

Concatenation
Kleene-star
Union
Intersection
Complement



Proposition 1. *Decidable languages are closed under union, intersection, and complementation.*

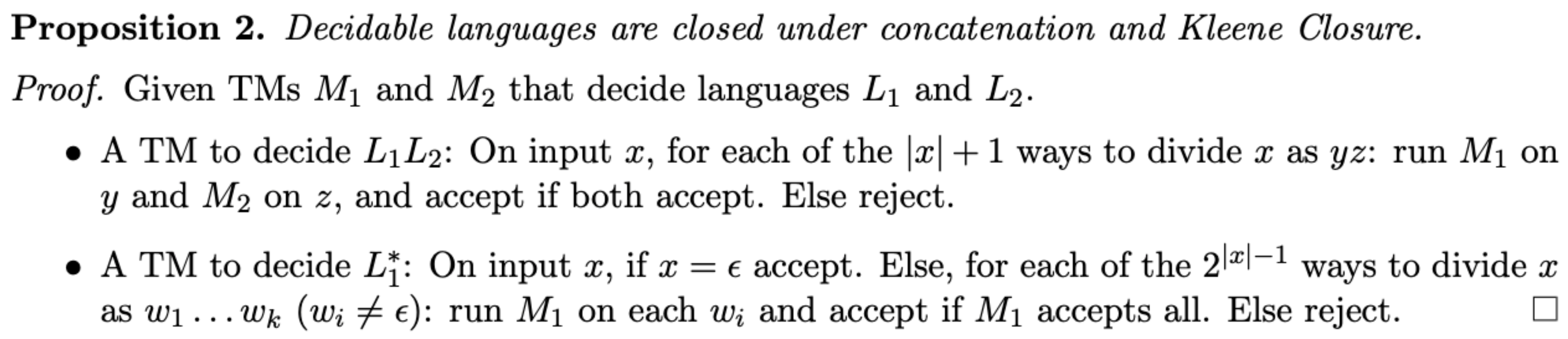
Proof. Given TMs M_1, M_2 that decide languages L_1, and L_2

- A TM that decides $L_1 \cup L_2$: on input x, run M_1 and M_2 on x, and accept iff either accepts. (Similarly for intersection.)
- A TM that decides $\overline{L_1}$: On input x, run M_1 on x, and accept if M_1 rejects, and reject if M_1 accepts. $\square$

Closure property

Recursive (Turing decidable) language is closed under:

Concatenation
Kleene-star
Union
Intersection
Complement



Proposition 2. *Decidable languages are closed under concatenation and Kleene Closure.*

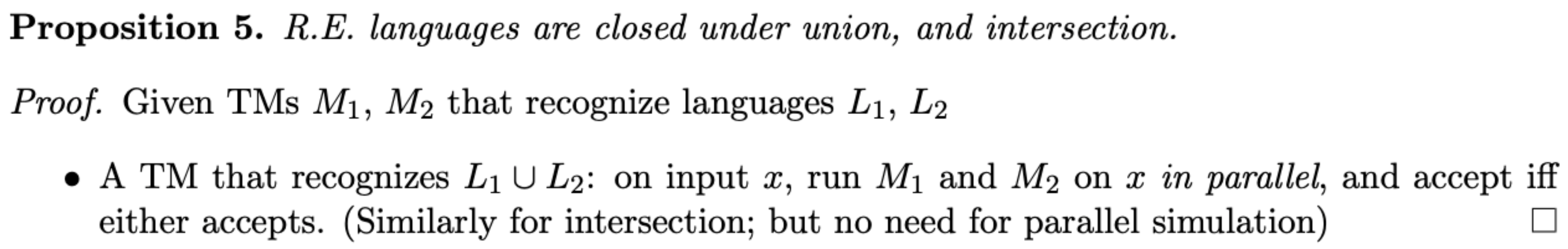
Proof. Given TMs M_1 and M_2 that decide languages L_1 and L_2.

- A TM to decide L_1L_2: On input x, for each of the $|x|+1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to decide L_1^*: On input x, if $x = \epsilon$ accept. Else, for each of the $2^{|x|-1}$ ways to divide x as $w_1 \dots w_k$ $(w_i \neq \epsilon)$: run M_1 on each w_i and accept if M_1 accepts all. Else reject. □

Closure property

Recursively enumerable (Turing semi-decidable) language is closed under:

Concatenation
Kleene-star
Union
Intersection



Proposition 5. *R.E. languages are closed under union, and intersection.*

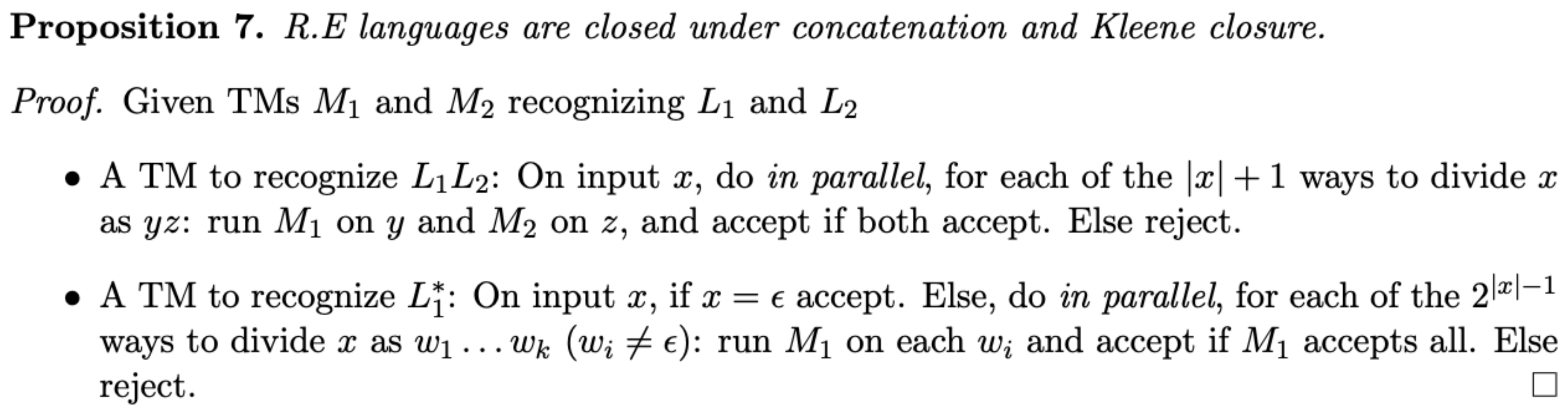
Proof. Given TMs M_1, M_2 that recognize languages L_1, L_2

- A TM that recognizes $L_1 \cup L_2$: on input x, run M_1 and M_2 on x *in parallel*, and accept iff either accepts. (Similarly for intersection; but no need for parallel simulation) $\square$

Closure property

Recursively enumerable (Turing semi-decidable) language is closed under:

Concatenation
Kleene-star
Union
Intersection



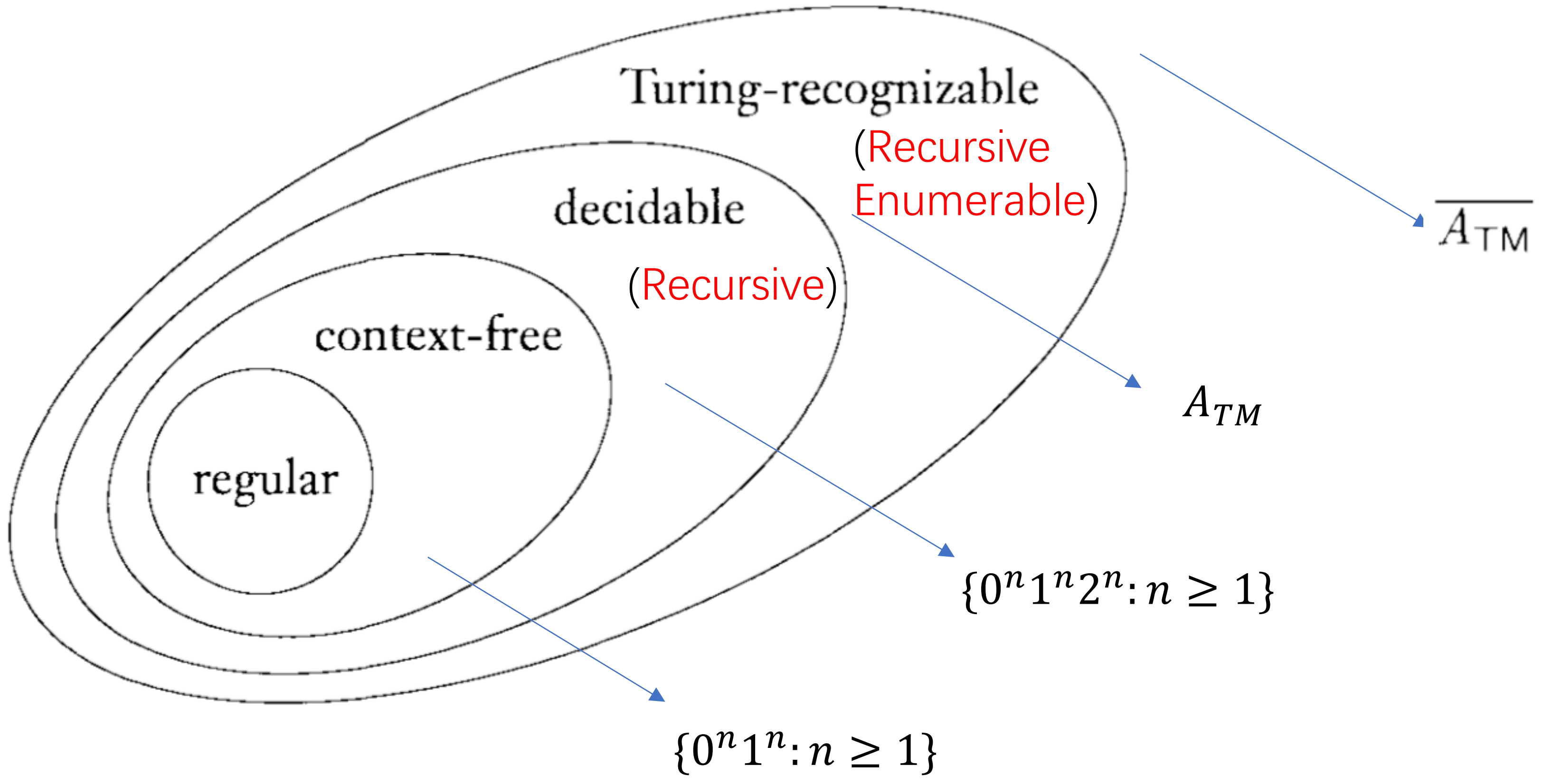
Proposition 7. *R.E languages are closed under concatenation and Kleene closure.*

Proof. Given TMs M_1 and M_2 recognizing L_1 and L_2

- A TM to recognize L_1L_2: On input x, do *in parallel*, for each of the $|x|+1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to recognize L_1^*: On input x, if $x = \epsilon$ accept. Else, do *in parallel*, for each of the $2^{|x|-1}$ ways to divide x as $w_1 \dots w_k$ ($w_i \neq \epsilon$): run M_1 on each w_i and accept if M_1 accepts all. Else reject. $\square$

Closure property

Recursively enumerable (Turing semi-decidable) language is not closed under complement


Turing-recognizable
(Recursive Enumerable)
decidable
(Recursive)
context-free
regular
$\overline{A_{TM}}$
A_{TM}
$\{0^n1^n2^n : n \geq 1\}$
$\{0^n1^n : n \geq 1\}$