# Non-Context-Free Languages and Turing Machines

Mon, October 19, 2020

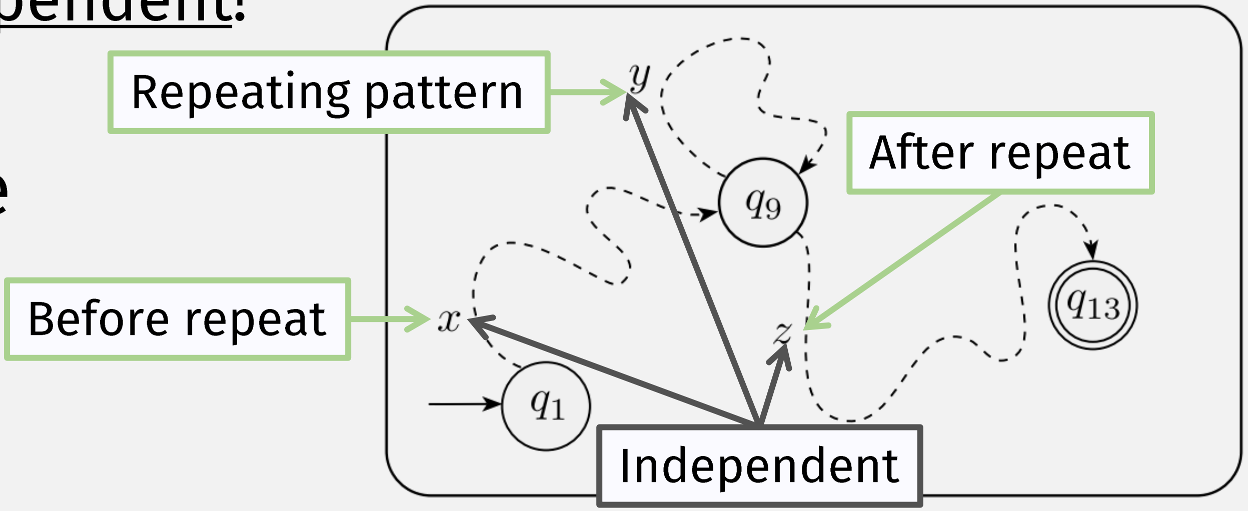
# HW4 Questions?

# Flashback: Pumping Lemma for Reg Langs

- The Pumping Lemma **describes how strings repeat**
- Regular lang strings can only repeat using Kleene pattern
  - But substrings are independent!
- A non-regular language

$$\{0^n 1^n \mid n \geq 0\}$$

Kleene can't express this pattern:
2nd part depends on (length of) 1st part




- What about context-free languages?

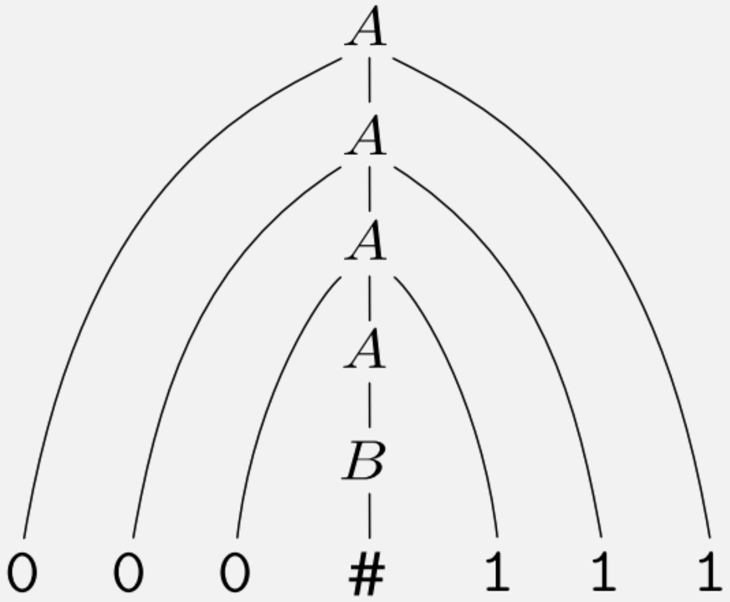
# Repetition and Dependency in CFLs

Grammar links first and second part

Repetition

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

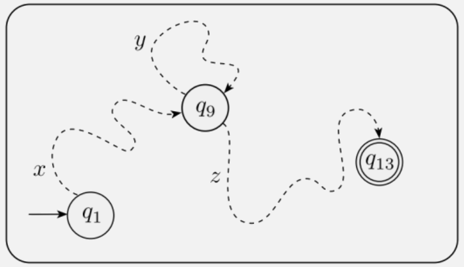
$$\{0^n\#1^n \mid n \geq 0\}$$



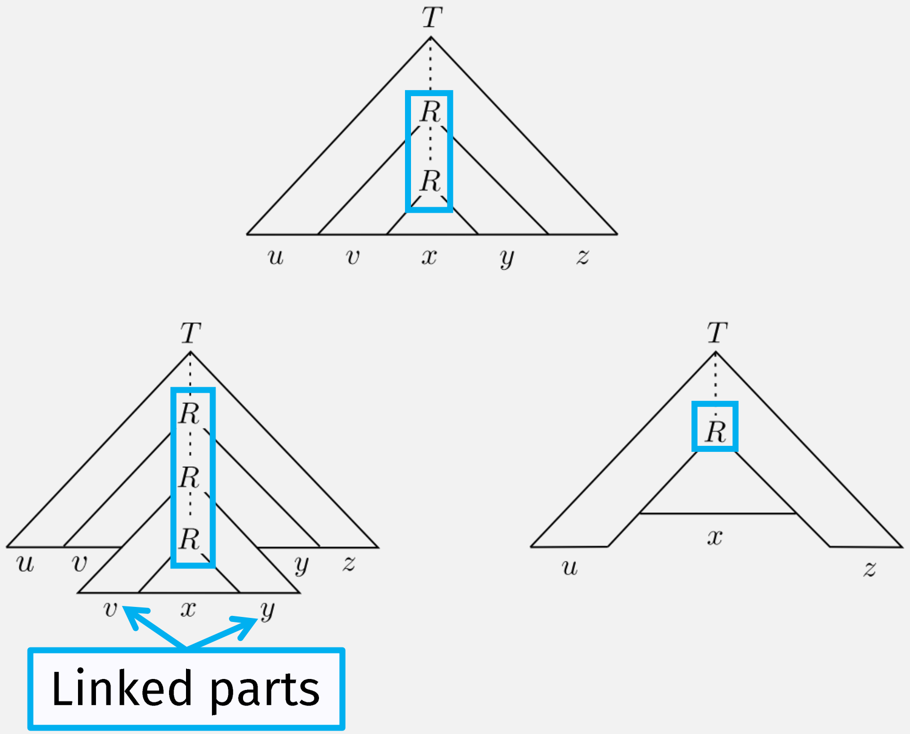

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$$

# How Can Strings in CFLs Repeat?

- Strings in regular languages repeat states




- Strings in CFLs <u>repeat subtrees</u> in the parse tree

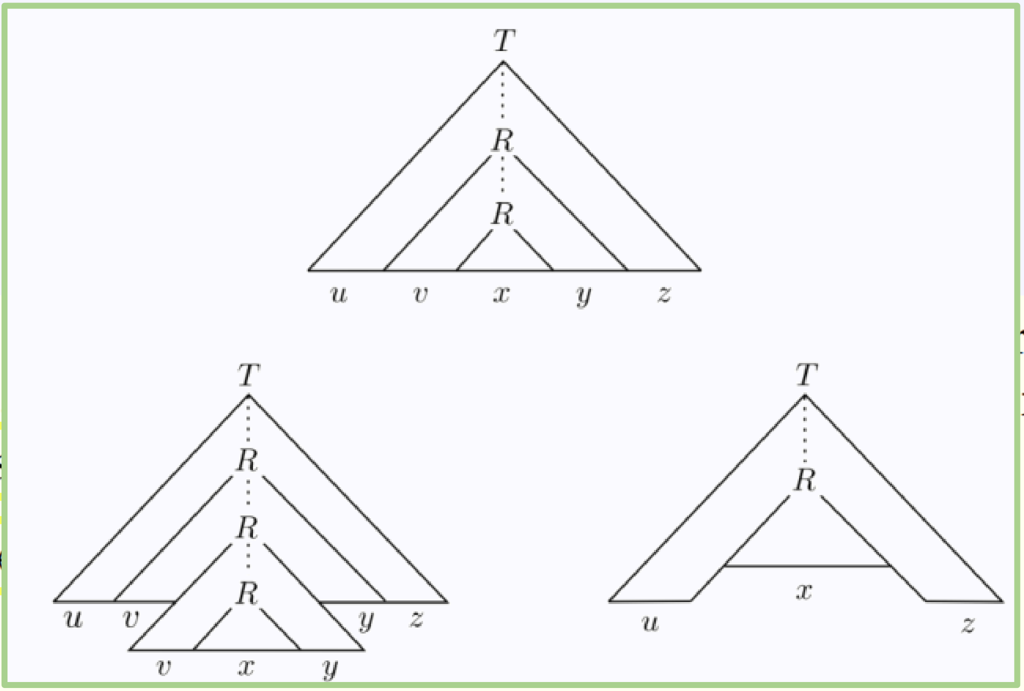


# Pumping Lemma for CFLS

**Pumping lemma for context-free languages** If $A$ is a context-free language, then there is a number $p$ (the pumping length) where, if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into five pieces $s = uvxyz$ satisfying the conditions

Now there are two pumpable parts.
But they must be pumped together!

1. for each $i \geq 0$, $uv^ixy^iz \in A$,
2. $|vy| > 0$, and
3. $|vxy| \leq p$.

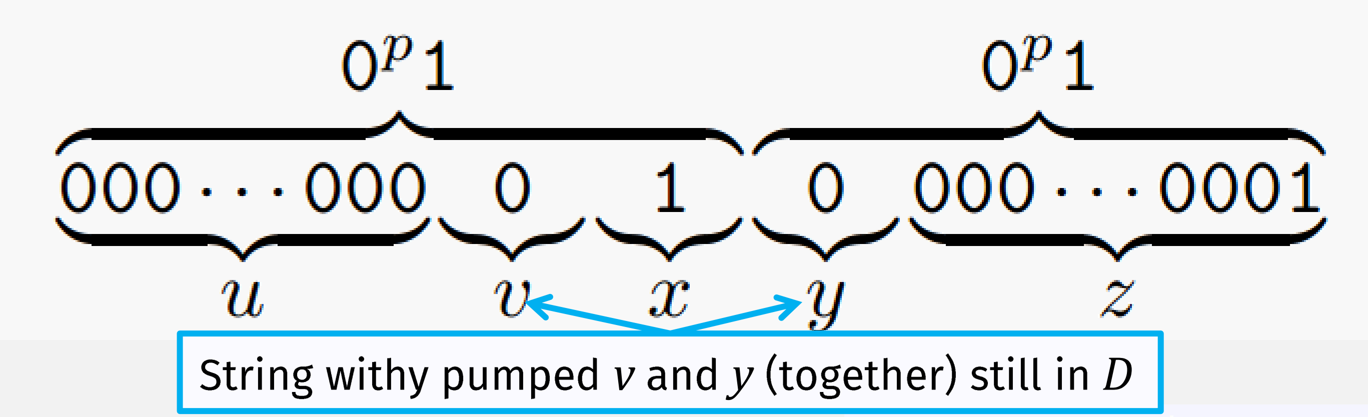



**Pumping lemma** If $A$ … $p$ (the pumping length) where if … may be divided into three pieces, $s$ …

1. for each $i \geq 0$, $xy^iz$ …
2. $|y| > 0$, and
3. $|xy| \leq p$.

# Non CFL example $D = \{ww \mid w \in \{0,1\}^*\}$

- Previous: Showed $D$ nonregular; unpumpable string $s$: $0^p10^p1$
- Now: But $s$ **can** be pumped according to CFL pumping lemma:




- CFL Pumping Lemma conditions: ☑ **1.** for each $i \geq 0$, $uv^ixy^iz \in A$,

  ☑ **2.** $|vy| > 0$, and

  ☑ **3.** $|vxy| \leq p$.

# Non CFL example $D = \{ww \mid w \in \{0,1\}^*\}$

- Choose another string *s*:

If *vyx* is all in first or second half, then any pumping will break the match ✖

$$0^p 1^p 0^p 1^p$$

So *vyx* must straddle the middle
But any pumping still breaks the match ✖

- CFL Pumping Lemma conditions:

**1.** for each $i \geq 0$, $uv^ixy^iz \in A$,

**2.** $|vy| > 0$, and

**3.** $|vxy| \leq p$.

# Non CFL example $D = \{ww \mid w \in \{0,1\}^*\}$

- Previously: Showed $D$ is not regular

- Just Now: $D$ is not context-free either!

# But that means …

- We previously said XML sort of looks context-free:
  - ELEMENT → <TAG>CONTENT</TAG>
  - TAG → any string
  - CONTENT → any string | ELEMENT

But these arbitrary TAG strings must match!

- Meaning XML also looks like: $D = \{ww|\ w \in \{0,1\}^*\}$

- So XML is not context-free either!
  - Note: HTML is context-free because …
  - … there are only a finite number of tags,
  - so we can hardcode them into a finite number of rules.

- In practice:
  - XML is parsed as a CFL, with a CFG
  - Then matching tags checked with a more powerful machine …

# A New Hypothetical Machine

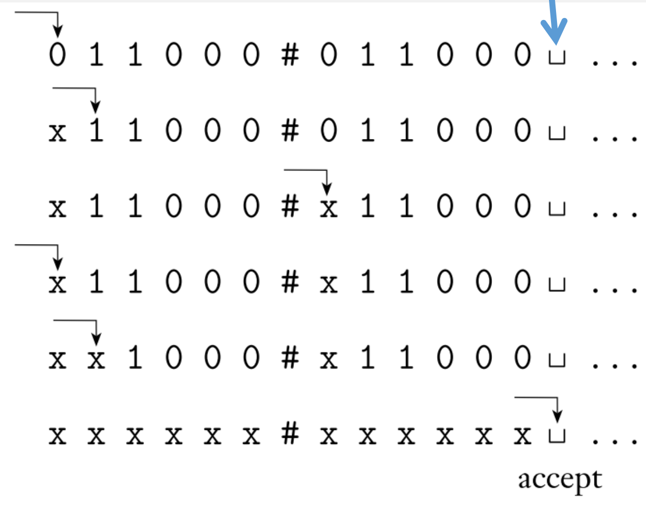
Blank space

$M_1$ accepts if input is in language $B = \{w\#w \mid w \in \{0,1\}^*\}$

memory where input is located

$M_1$ = "On input string $w$:

1. Zig-zag across the tape to corresponding positions on either side of the # symbol to check whether these positions contain the same symbol. If they do not, or if no # is found, *reject*. Cross off symbols as they are checked to keep track of which symbols correspond.

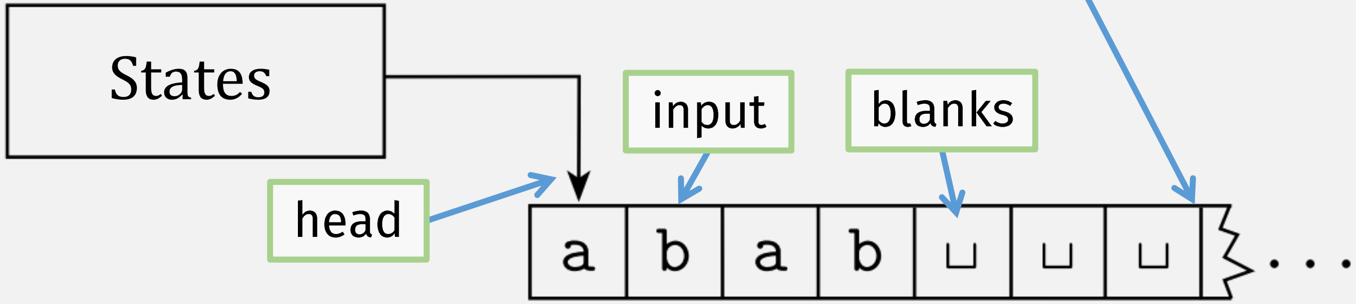



2. When all symbols to the left of the # have been crossed off, check for any remaining symbols to the right of the #. If any symbols remain, *reject*; otherwise, *accept*."

# Turing Machines (TMs)

# Automata vs Turing Machines

- Turing Machines can read <u>and write</u> to input "tape"
- The read-write "head" can move arbitrarily left or right
- The tape is infinite
- A Turing Machine can accept/reject at any time




**DEFINITION 3.5**

Call a language ***Turing-recognizable*** if some Turing machine recognizes it.

# Turing Machines: Formal Definition

**DEFINITION 3.3**

A ***Turing machine*** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where $Q$, $\Sigma$, $\Gamma$ are all finite sets and

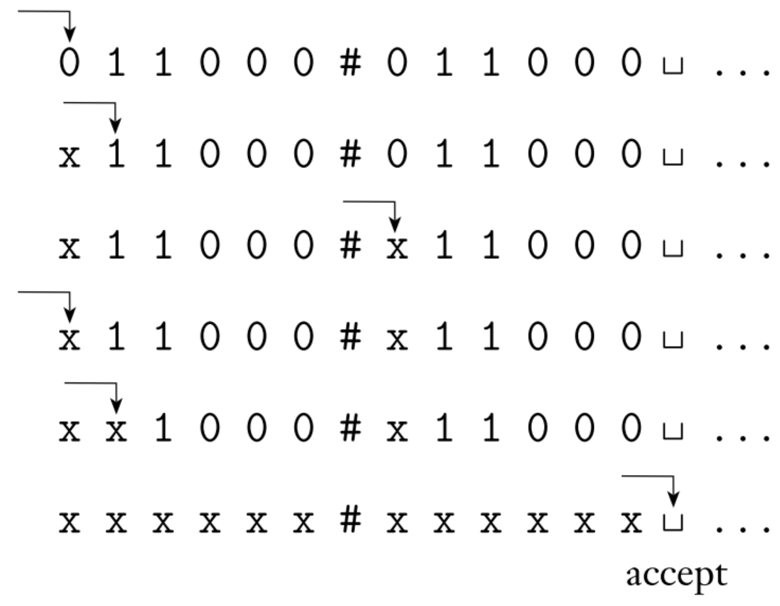
1. $Q$ is the set of states,
2. $\Sigma$ is the input alphabet not containing the ***blank symbol*** $\sqcup$,
3. $\Gamma$ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,

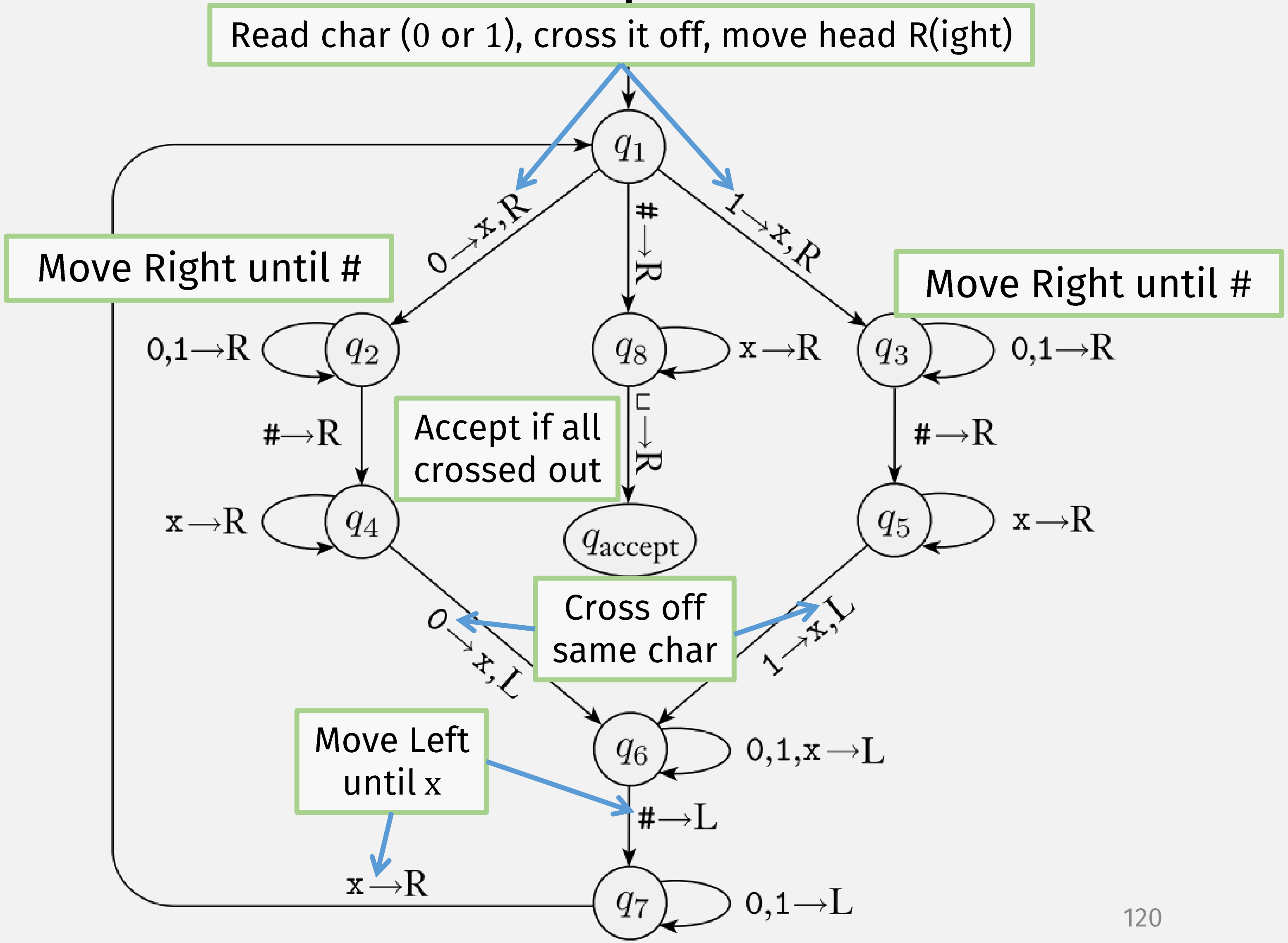
   read write move

5. $q_0 \in$ … he st… te,
6. $q_{accept} \in Q$ is the accept state, and
7. $q_{reject} \in Q$ is the reject state, where $q_{reject} \neq q_{accept}$.

# Formal Turing Machine Example

$$B = \{w\#w \mid w \in \{0,1\}^*\}$$



# Turing Machine: Informal Description

- $M_1$ accepts if input is in language $B = \{w\#w |\ w \in \{0,1\}^*\}$

$M_1 =$ "On input string $w$:

**1.** Zig-zag across the tape to corresponding positions on either side of the # symbol to check whether these positions contain the same symbol. If they do n… if no # is found, *reject*. Cross off symbols as the… track of which symbols correspond.

**2.** When all symbols to … n crossed off, check for any remaining … nt of the #. If any symbols remain, *reject*; otherwise, *accept*."

We will (mostly) stick to informal descriptions of Turing machines, like this one

# TM Informal Description: Caveats

- TM informal descriptions are not a “do whatever” card
  - They must be sufficiently precise to communicate the formal tuple
- Input must be a string, written with chars from finite alphabet
- An informal “step” represents sequence of formal transitions
  - I.e., some **finite** number of transitions
  - It cannot run forever
  - E.g., can’t say “try all numbers” as a “step”

# Non-halting Turing Machines (TMs)

- A DFA, NFA, or PDA always halts
  - Because the (finite) input is always read exactly once
- But a Turing Machine can run forever
  - E.g., the head can move back and forth in a loop
- A <u>decider</u> is a Turing Machine that always halts.

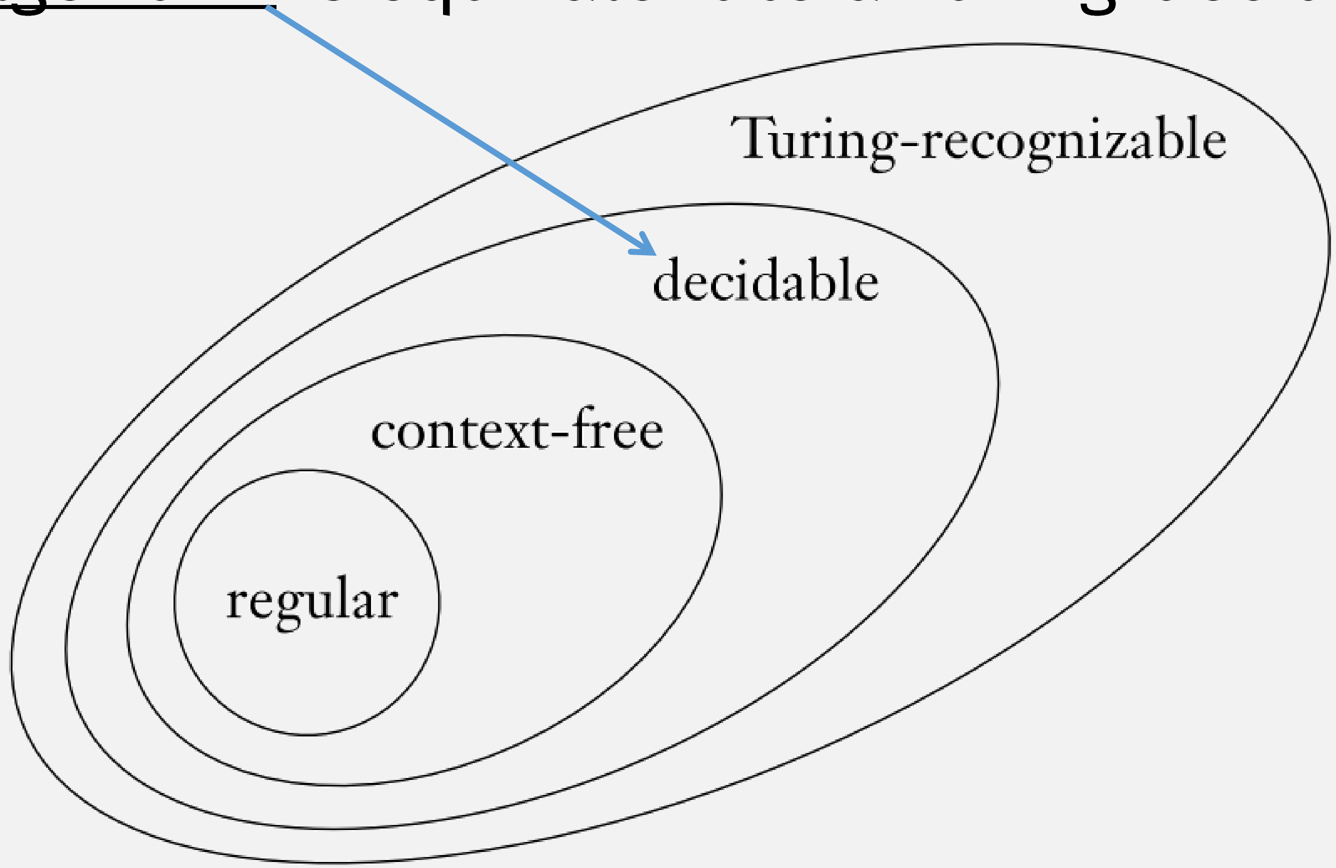
THIS IS A VERY IMPORTANT SLIDE

**DEFINITION 3.6**

Call a language ***Turing-decidable*** or simply ***decidable*** if some Turing machine decides it.

# Formal Definition of an "Algorithm"

- An <u>algorithm</u> is equivalent to a Turing-decidable Language



## Check-in Quiz 10/19

On Gradescope

## End of Class Survey 10/19

See course website