Feedback:

- Many thanks for the feedback!

- Terms: decidable, recognizable, etc. — there is a table in §2.4 of the handout

- Remarks on my enthusiasm and handwriting

- Next reason for enthusiasm:
  regular and non-regular languages over \( \Sigma = \{ a \} \).

New as of 2021, based on a question of Markus de Medeiros
Remark.

\textsc{Non-Python} (in the handout) equals \textsc{Not-Prog-Plus-Input} (in class), equals \[
\{ \text{strings not of the form } p\sigma_0^i \mid \text{where } p \text{ is a valid Python program} \}\]
is decidable (in polynomial time)

Also

\textsc{Groucho-Marx-Self} equals \textsc{Non-Self-Acceptance} equals \[
\{ p \mid p \notin \text{LanguageRecBy}(p) \}\]
Today

- Review the formal definition of a DFA as a tuple \((Q, \Sigma, \delta, q_0, F)\)

- Possible examples:

  \[ L = \{ w \in \{a, b\}^* \mid \text{first letter of } \omega \} \]

  \[ L = \{ w \in \{0, 1\}^* \mid \text{w, in binary, represents an integer divisible by 3} \} \]

  and/or similarly, with \( w \in \{0, 1, \ldots, 9\}^* \), etc.

- Regular and non-regular languages when \( \Sigma = \{a\} \)
Last class: begin with example

\[ \Sigma = \{ a, b, c, \lambda \} \]

\[ \mathcal{L} = \{ s_1 u b c s_2 \mid s_1, s_2 \in \Sigma^* \} \]

\[ Q = \text{set of states} = \{ q_0, q_1, q_2, q_3 \} \]
In the picture of what a DFA is

\[ \xrightarrow{\text{non-accepting state}} \quad \xrightarrow{\text{accepting state}} \]

initial state

input \( \sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_n \)
Formally, a DFA (deterministic finite automaton) is a tuple

$$(Q, \Sigma, \delta, q_0, F)$$

$Q =$ states of DFA

$\Sigma =$ alphabet

$\delta : Q \times \Sigma \rightarrow Q$

meaning

$\delta(q, \sigma) =$ the next state that you move to when in state $q$, next input symbol is $\sigma$
$q_0 = \text{initial state}$

$F = \text{set of accepting states}$

"final"

We say a DFA $(Q, \Sigma, \delta, q_0, F)$ accepts $i$ if

$i \in \Sigma^*$, and when we run $i$ on $M = (Q, \Sigma, \delta, q_0, F)$

we end in a state $\in F$.

Language recognized by $M$
\[ \{ i \in \Sigma^* \mid M \text{ accepts } i \} \]

A language \( L \subseteq \Sigma^* \) is regular if \( L \) is recognized by some DFA, \( M = (Q, \Sigma, \delta, q_0, F) \).

(Otherwise we say \( L \) is non-regular.)

Make the formalities as simple as possible.
Remark

0 \rightarrow "no"

6 \rightarrow "yes"

all DFA's can be viewed as very simple Python programs

Q in a DFA roughly program line or set of lines

"Complexity" of a regular might be the minimum number of states
Another example:

\[ \{ s \in \{0,1,\ldots,9\}^* \mid \text{s represents a string in decimal divisible by 3} \} \]

- we don't allow \( \varepsilon \)
- leading 0's OK

\[ \{ 3, 6, 9 \} \]

\[ \{ 0, 00, 03, 06, 09, 12, 15, \ldots \}

\[ 96, 99, 000, 003, \ldots \} \]

DIV-BY-3-IN-DECIMAL-
LEADING-0'S-OK
Algorithm: an integer in decimal is divisible by 3 iff the sum of its digits is divisible by 3.
DIV-BY-10 - IN-DECIMAL-LEADING-ZEROS

- OK

- So far, not div by 10

- So far div by 10

We can merge $\mathcal{Q}_0$ and $\mathcal{Q}_1$.
So for the sum is divisible by 3.

So far, the sum mod 3 is 1.

So far, the sum mod 3 is 2.