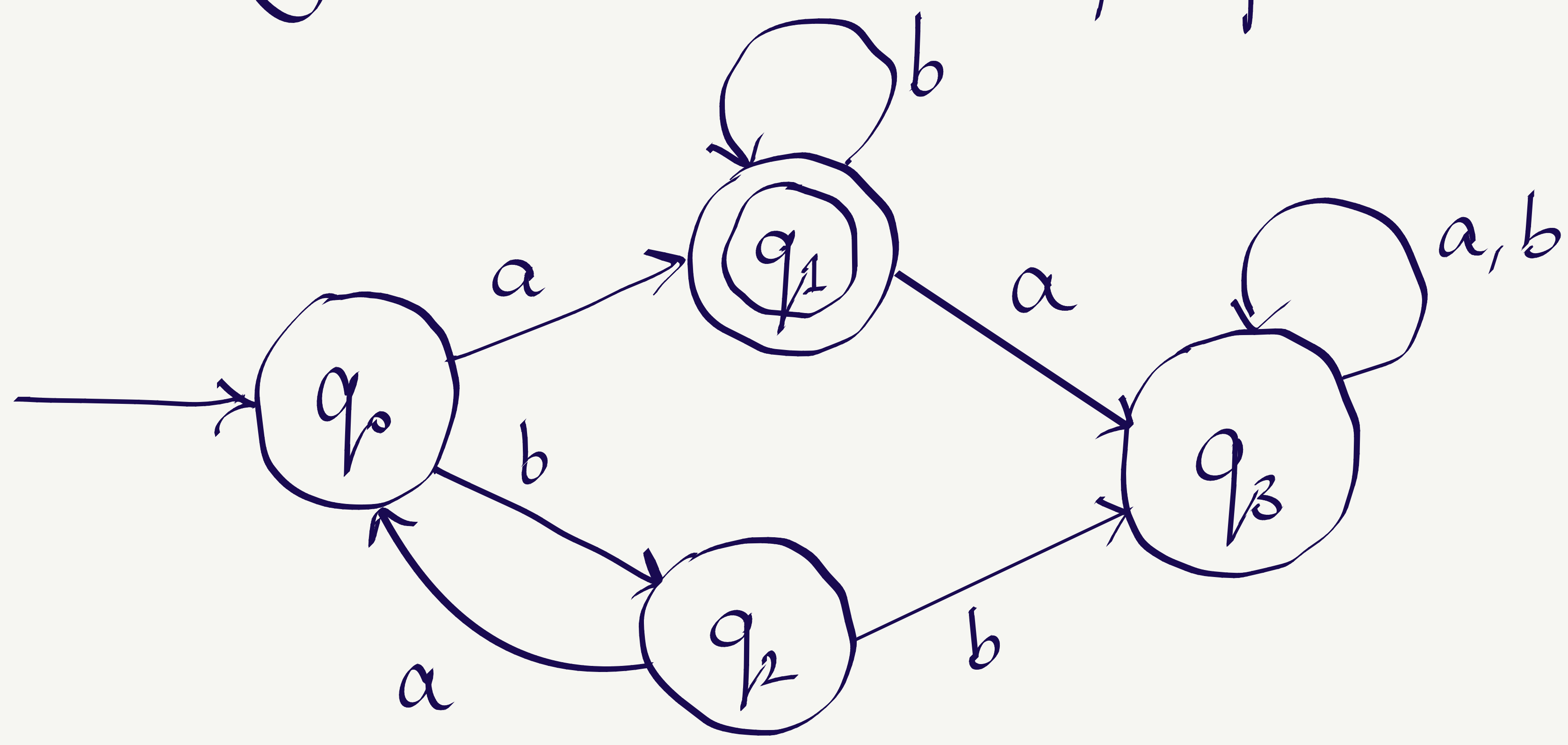
Machines As Input

Recall: Saw some equivalent models of TMs

Today: TMs can accept other machines as input; how do we use this?

We already saw some example problems for which we described algorithms



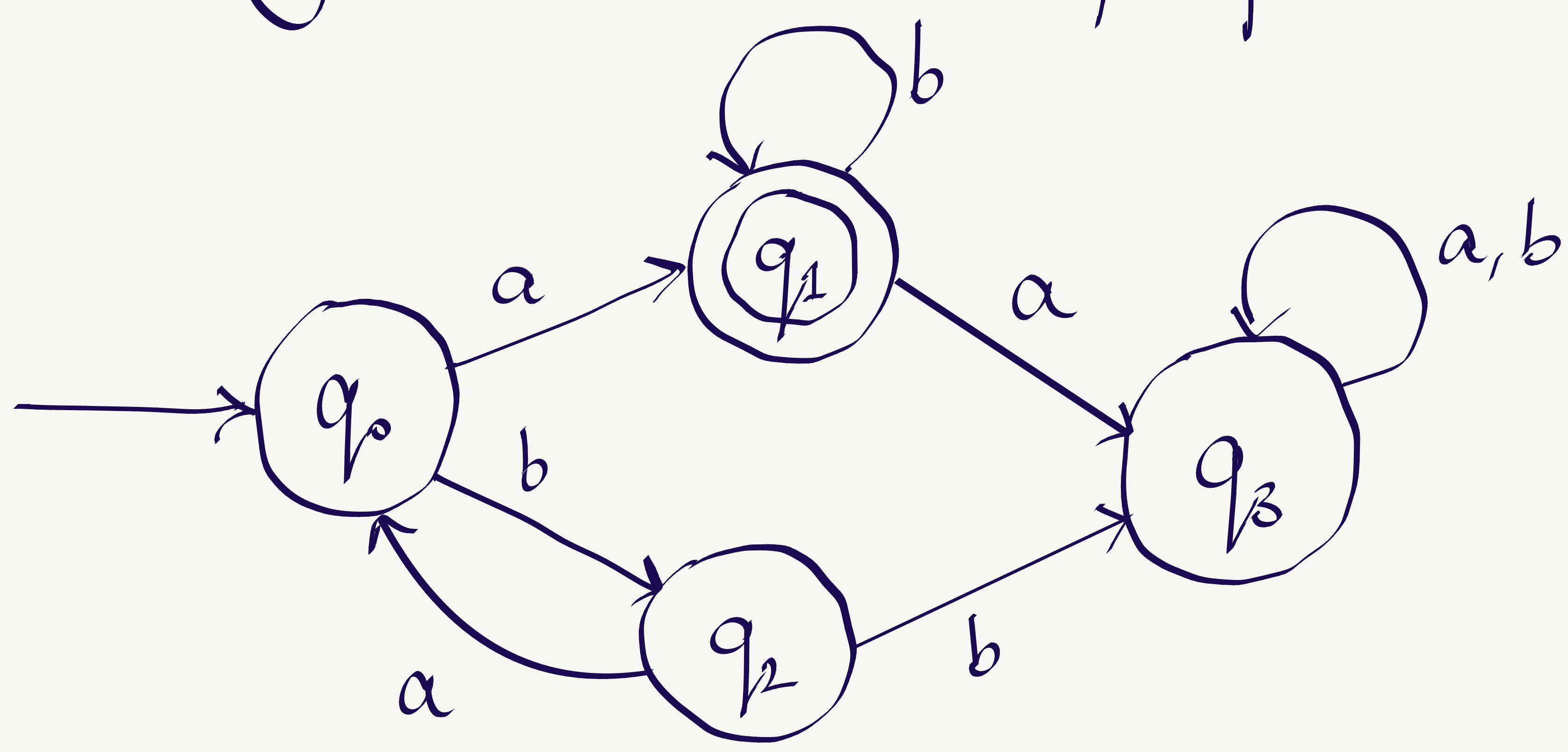
Does this machine accept abbbaaabb?

What is an algorithm to check if this machine accepts this string?

Recall: Saw some equivalent models of TMs

Today: TMs can accept other machines as input; how do we use this?

We already saw some example problems for which we described algorithms



Does this machine accept abbbaaabb?

What is an algorithm to check if this machine accepts this string?

What about this algorithm is specific to this machine or this string?

Essentially, the algorithm decides the following language:

$$L_{DFA} = \{ D\#\omega \mid D \text{ is a DFA which accepts } \omega \}$$

D has to be of the form $(Q, \Sigma, \delta, q_0, F)$

ω is a string over Σ

can be expressed as a finite string

Suppose we give $D\#\omega$ as input to a Turing Machine M

→ Check if D is of the form $(Q, \Sigma, \delta, q_0, F)$

→ Run D on ω

→ See if D ends up in an accepting state

how do we implement this?

Use the tape to keep track of state and symbol currently being read

Is $\overline{L_{DFA}} = \{ D\#\omega \mid D$ is a DFA which does not accept $\omega \}$ decidable? [In general, complement of decidable languages...]

$L_{NFA} = \{ N\#\omega \mid N$ is an NFA which accepts $\omega \}$

→ Check if N is of the form $(Q, \Sigma, \Delta, Q_0, F)$

→ Run N on ω

→ See if N ends up in an accepting state

Another option is to use the earlier M.

→ Convert N to its equivalent DFA D_N

→ Run M on D_N → Incorporate the functionality of M

→ Accept if M accepts

Is the following language decidable?

$$\mathcal{L}_{CFG} = \{ G \# w \mid G \text{ is a CFG which generates } w \}$$

What about the following?

$$\mathcal{L}_{TM} = \{ M\#w \mid M \text{ is a Turing Machine which accepts } w \}$$

Whether one has an algorithm for deciding this language or not, one needs a Turing Machine which can simulate an arbitrary Turing Machine on arbitrary input.

Such a TM is called a Universal Turing Machine (UTM).

$\mathcal{L}_{TM}$ is Turing-recognizable, or recursively enumerable (r.e.)

Use a UTM to do the following on input $M\#w$:

- Simulate M on w — might loop forever, in which case this UTM also loops forever
- If M accepts, accept.

We said that if $\mathcal{L}$ is decidable, so is $\overline{\mathcal{L}}$.

What if both $\mathcal{L}$ and $\overline{\mathcal{L}}$ are r.e.?

Can we say something more?

M_1 : Y if $x \in \mathcal{L}$, $\longrightarrow$ if $x \notin \mathcal{L}$

M_2 : Y if $x \in \overline{\mathcal{L}}$, $\longrightarrow$ if $x \notin \overline{\mathcal{L}}$

Then, $\mathcal{L}$ is decidable