Different Kinds of Turing Machines

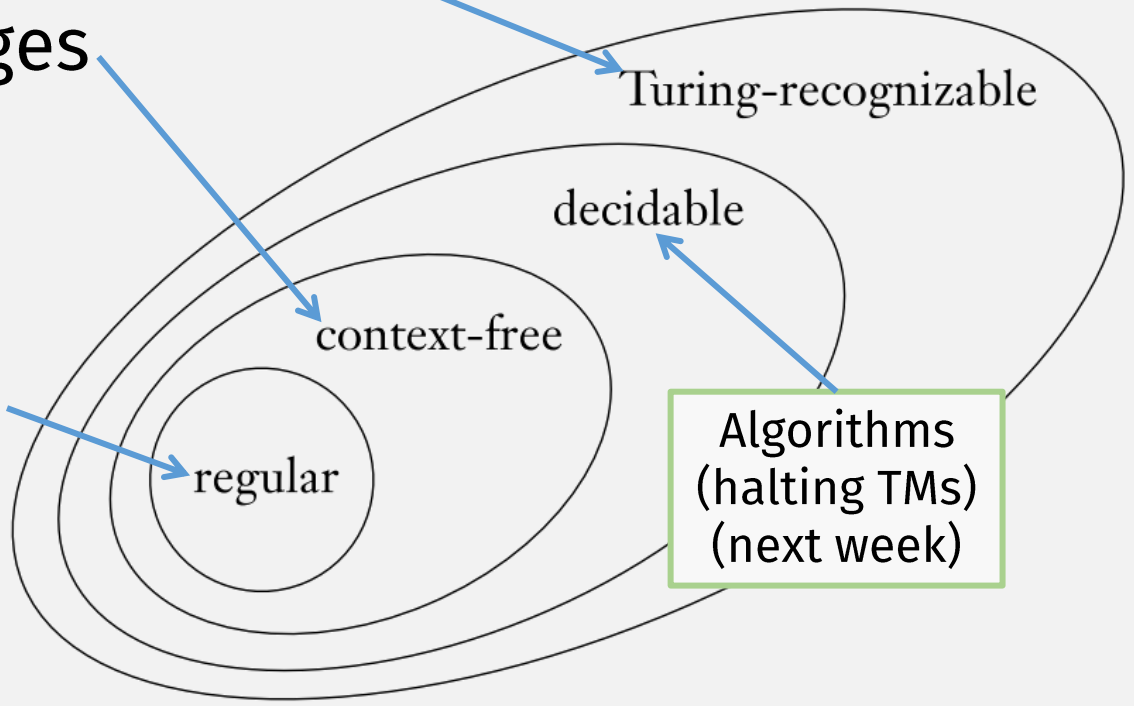
Wed October 21, 2020

HW5 out

- The README requirement is back
  - Time spent
  - Who you discussed hw with
  - Other sources consulted

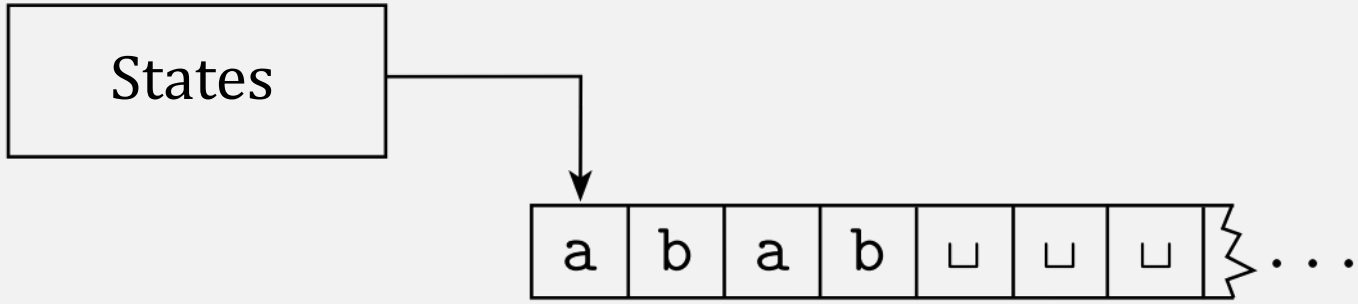
CS420, So Far

- Turing Machines (TMs)
  - Infinite tape (memory), arbitrary read/write
  - Models "computers"
- PDAs: recognize context-free languages
  - Infinite stack (memory), push/pop only
  - Can't recognize langs w. arbitrary dependency, e.g., $\{ww | \ w \in \{0,1\}^*\}$
- DFAs / NFAs: recognize regular langs
  - Finite states (memory)
  - Can't recognize langs w. dependency e.g., $\{0^n 1^n | \ n \geq 0\}$



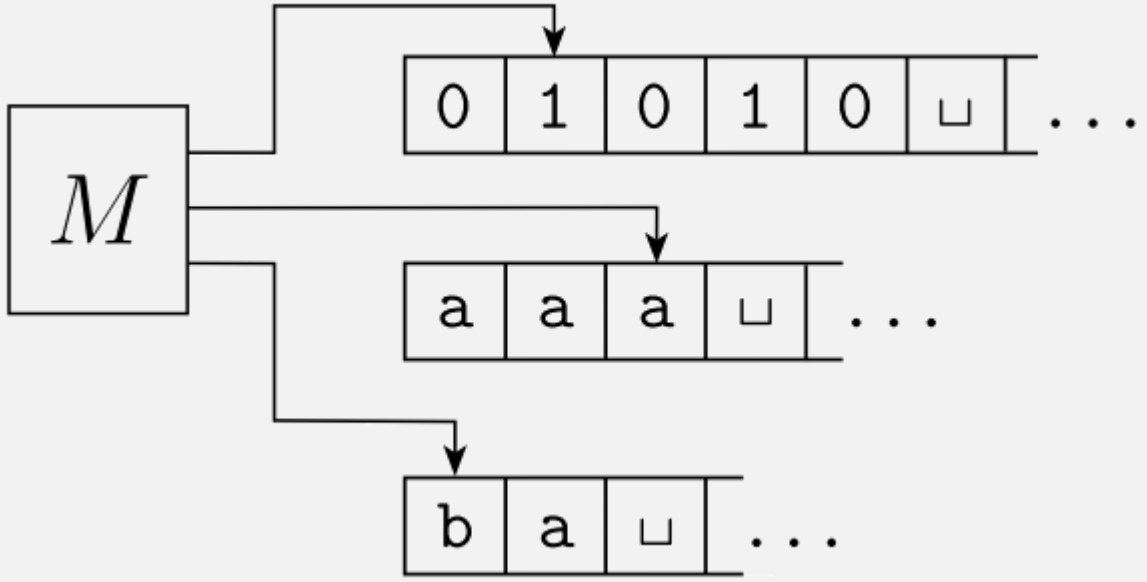
Last time: Turing Machines

- Turing Machines can read and write 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

Multi-Tape Turing Machines



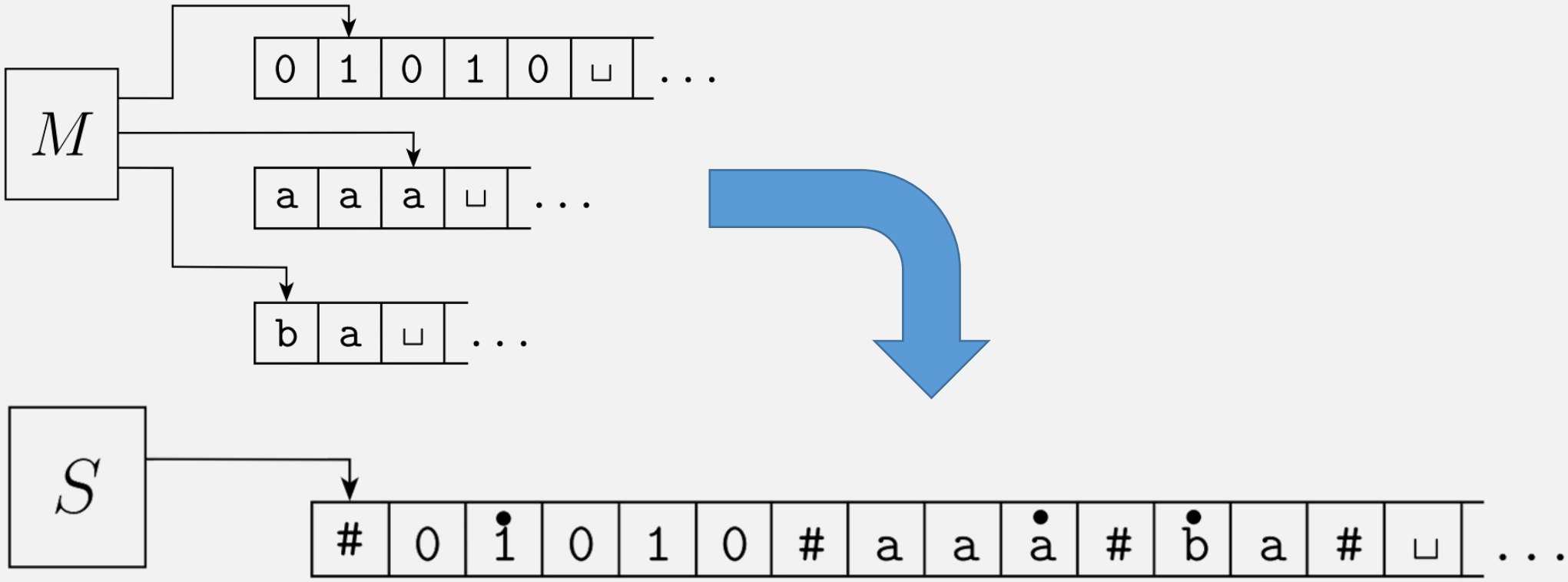
Single-tape TM ⇔ Multi-tape TM

- => If a single-tape TM recognizes a language,
  then a multi-tape TM recognizes the language
  - A single-tape TM is a multi-tape TM that does not use other tapes

- <= If a multi-tape TM recognizes a language,
  then a single-tape TM recognizes the language
  - Convert multi-tape TM to single-tape TM

Multi-tape TM ➔ Single-tape TM

- Use delimiter (#) on single-tape to simulate multiple tapes
- Add “dotted” version of every char to simulate multiple heads



Single-tape TM ⇔ Multi-tape TM

- => If a single-tape TM recognizes a language,
  then a multi-tape TM recognizes the language
  - A single-tape TM is a multi-tape TM that does not use other tapes

- <= If a multi-tape TM recognizes a language,
  then a single-tape TM recognizes the language
  - Convert multi-tape TM to single-tape TM **(DONE!)**

Non-determistic Turing Machines (NTMs)

Last time: Turing Machine formal def

DEFINITION 3.3

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

1. Q is the set of states,
2. Σ is the input alphabet not containing the ***blank symbol*** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{L}, \text{R}\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$.

Non-deterministic Turing Machine formal def

DEFINITION 3.3

Nondeterministic Turing Machine

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

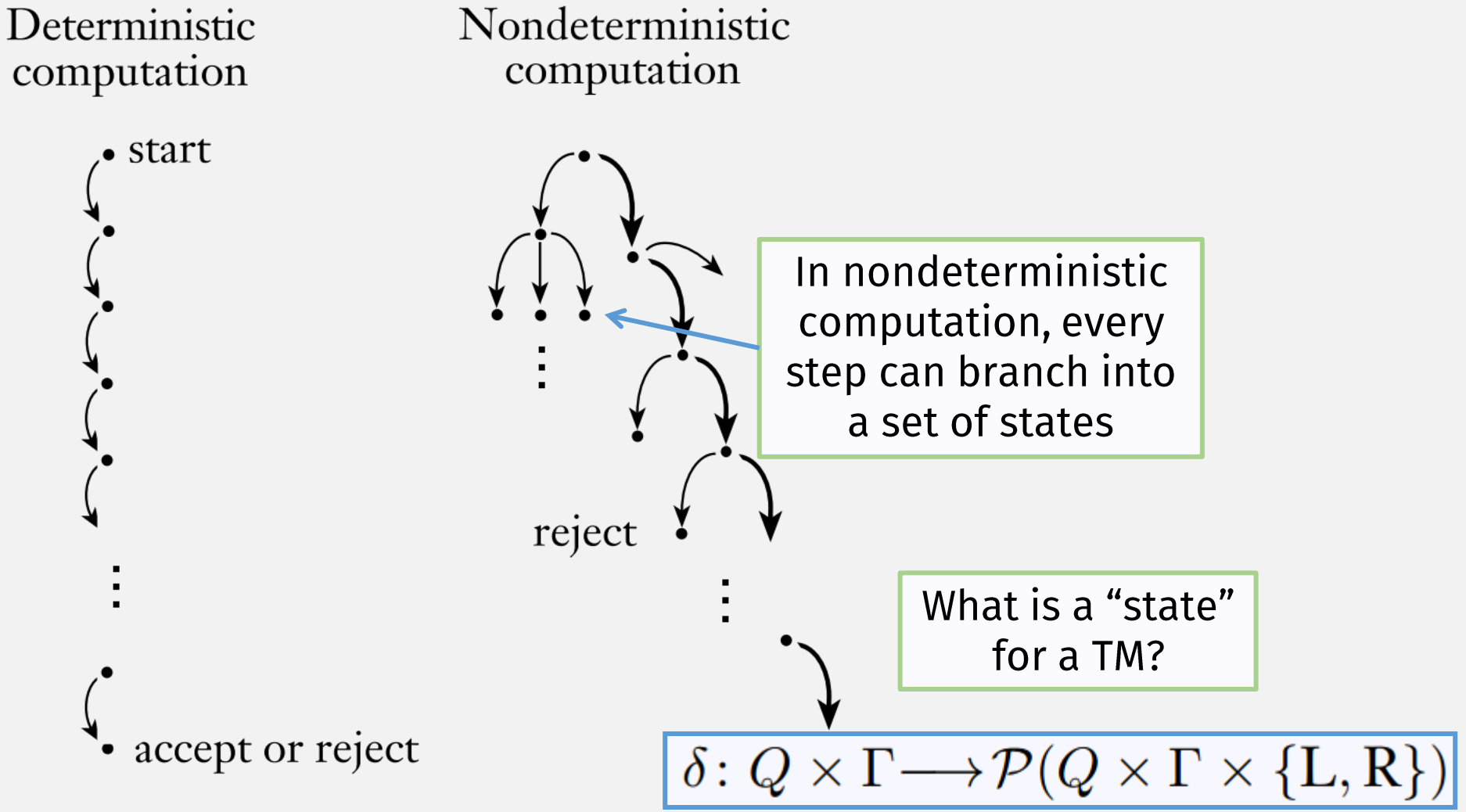
1. Q is the set of states,
2. Σ is the input alphabet not containing the ***blank symbol*** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. ~~$\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{L}, \text{R}\}$~~ $\delta: Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{\text{L}, \text{R}\})$
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$.

Deterministic TM ⇔ Nondeterministic TM

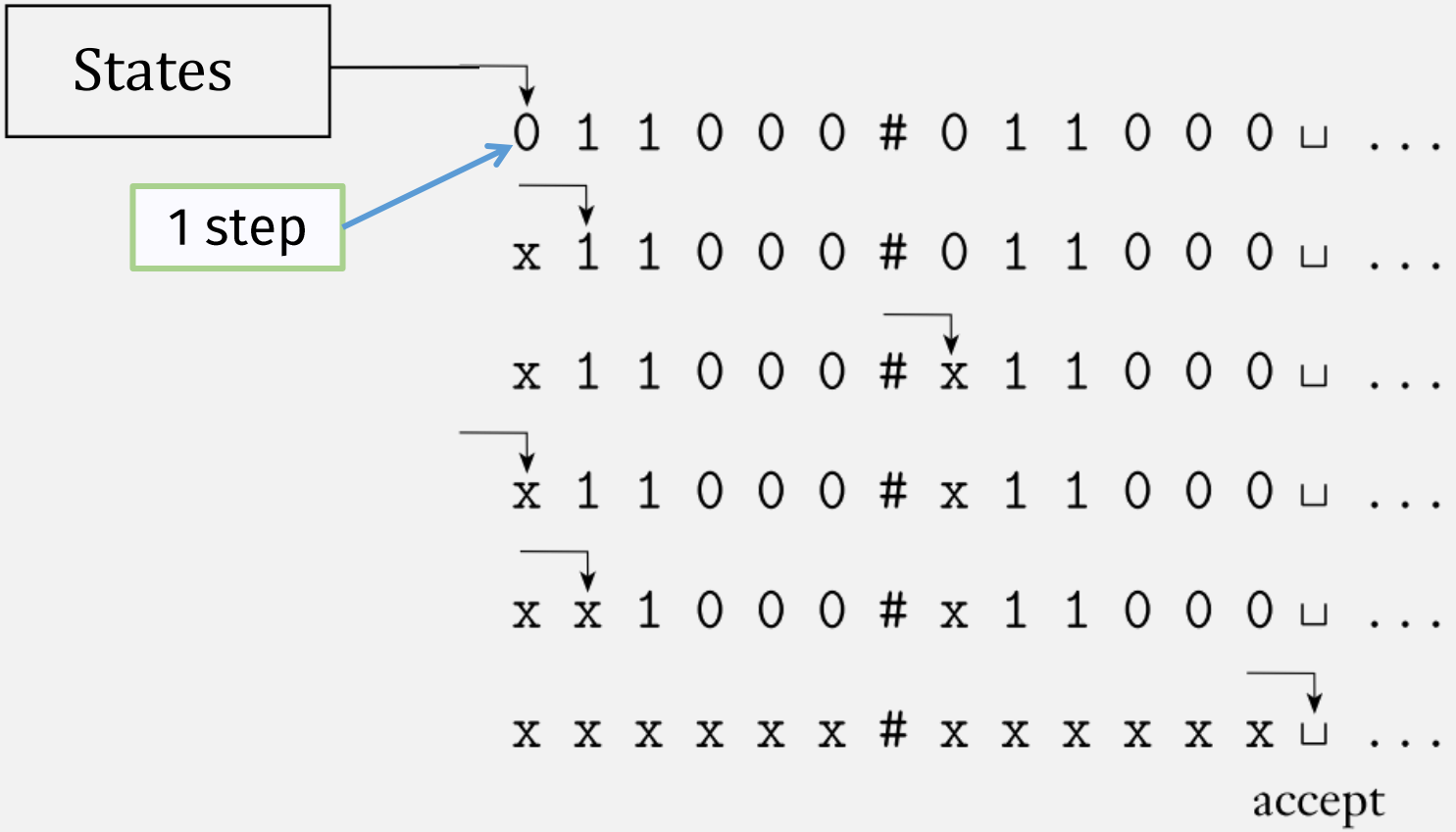
- => If a deterministic TM recognizes a language, then a nondeterministic TM recognizes the language
  - Deterministic TM → nondeterministic TM
  - Wrap output of delta in a set

- <= If a nondeterministic TM recognizes a language, then a deterministic TM recognizes the language
  - Nondeterministic TM → deterministic TM
  - ???

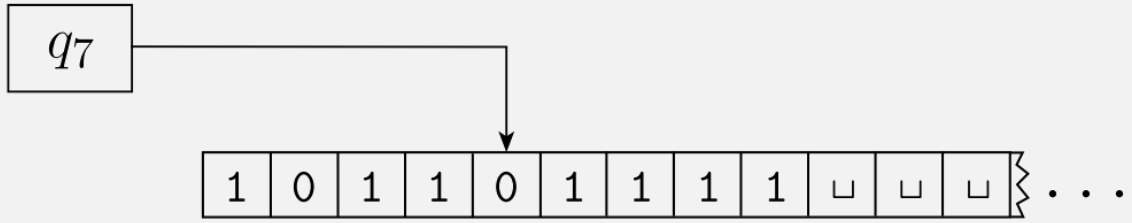
Nondeterminism



TM Configuration = Representation of 1 step



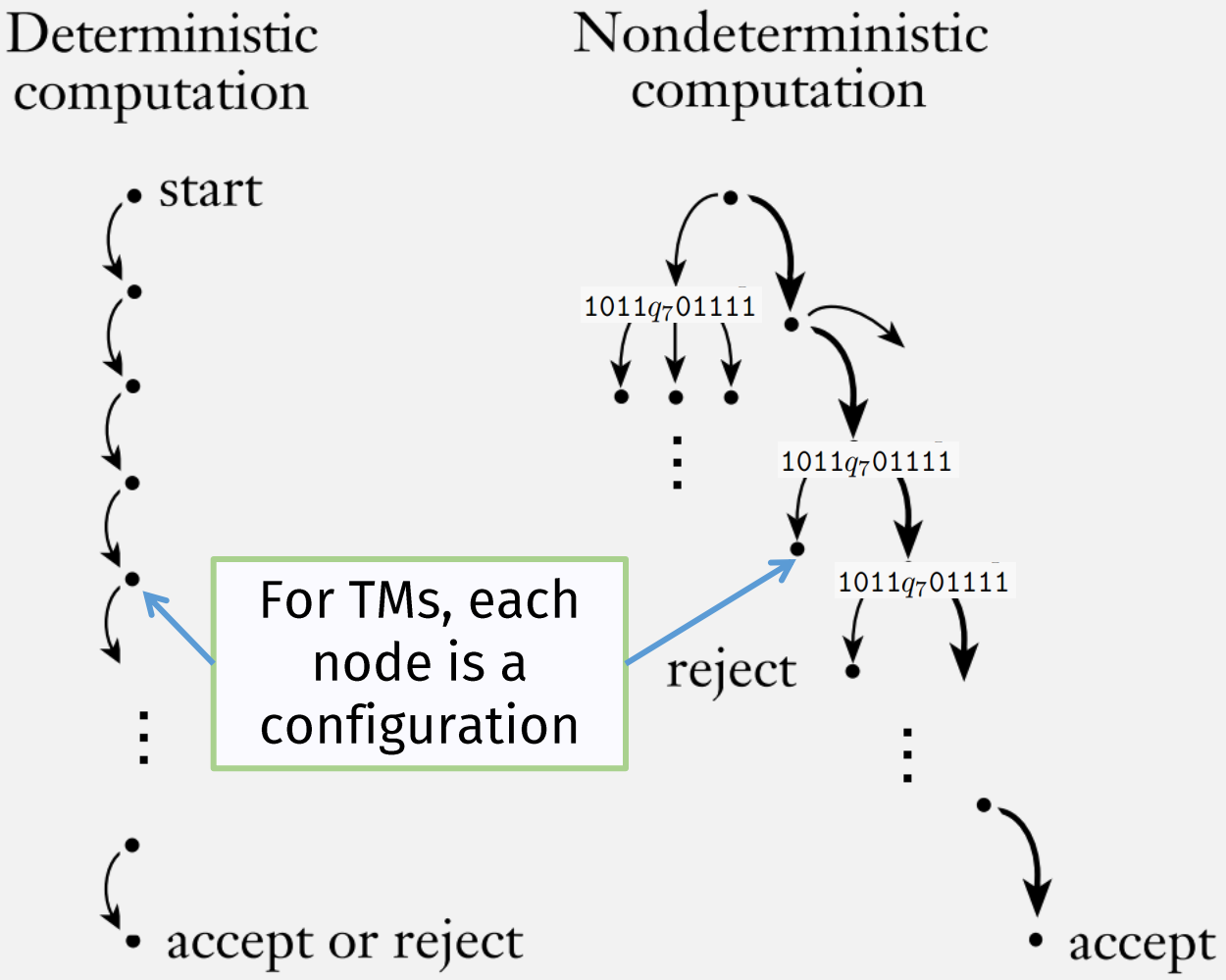
TM Configuration = State + Head + Tape



$$1011\underline{q_7}01111$$

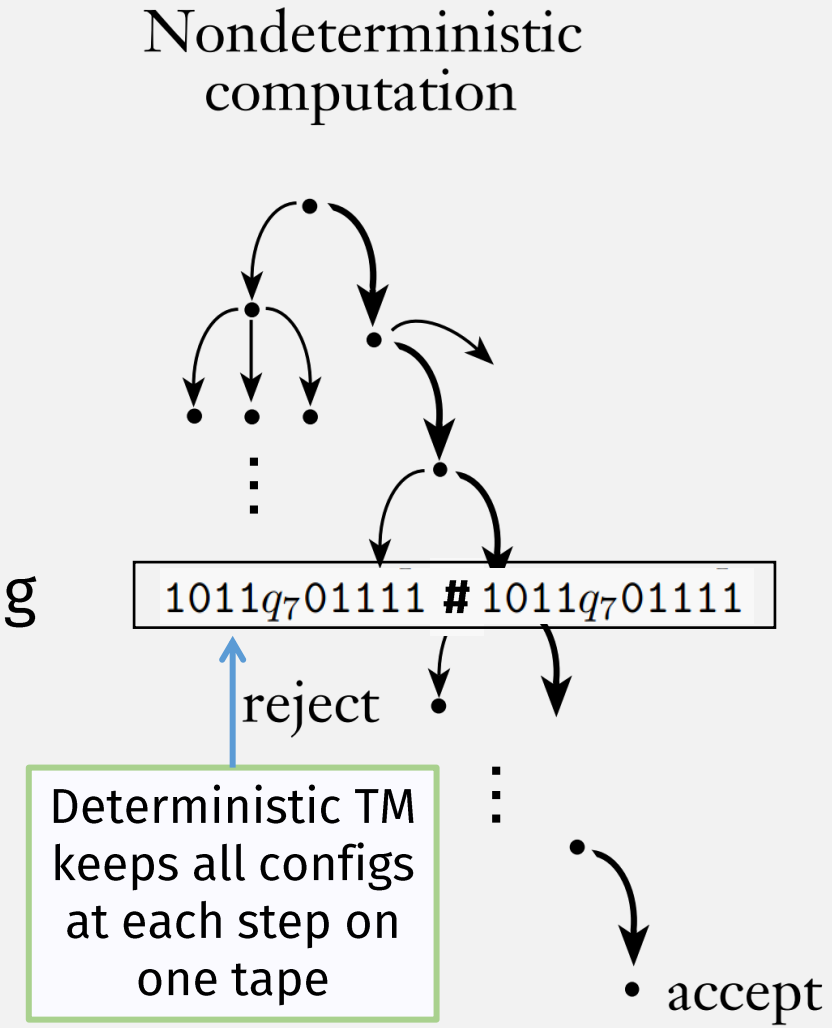
Textual representation of “configuration”

Nondeterminism



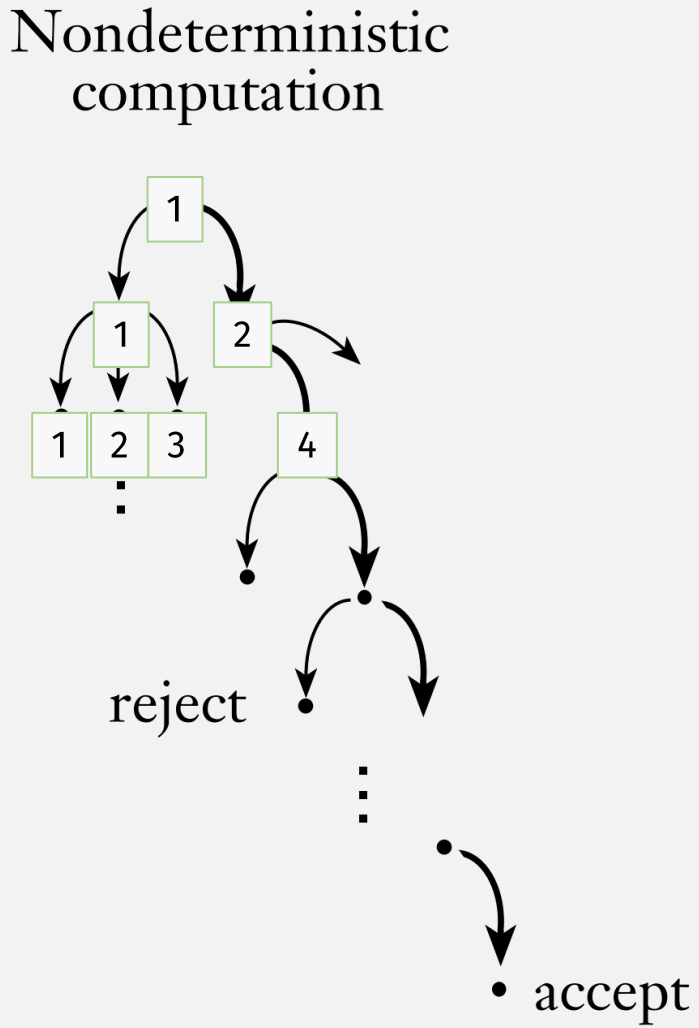
Nondeterministic TM → Deterministic: 1st way

- Deterministic TM simulates nondeterministic TM:
  - When computation branches, deterministic TM keeps multiple configurations on its (single) tape
  - (Similar to how a single-tape TM simulates multitapes)
  - It steps each config one-by-one, adding or removing when needed
  - Accept if accepting config is found
  - **Note**: Must step configs breadth-first (why?)



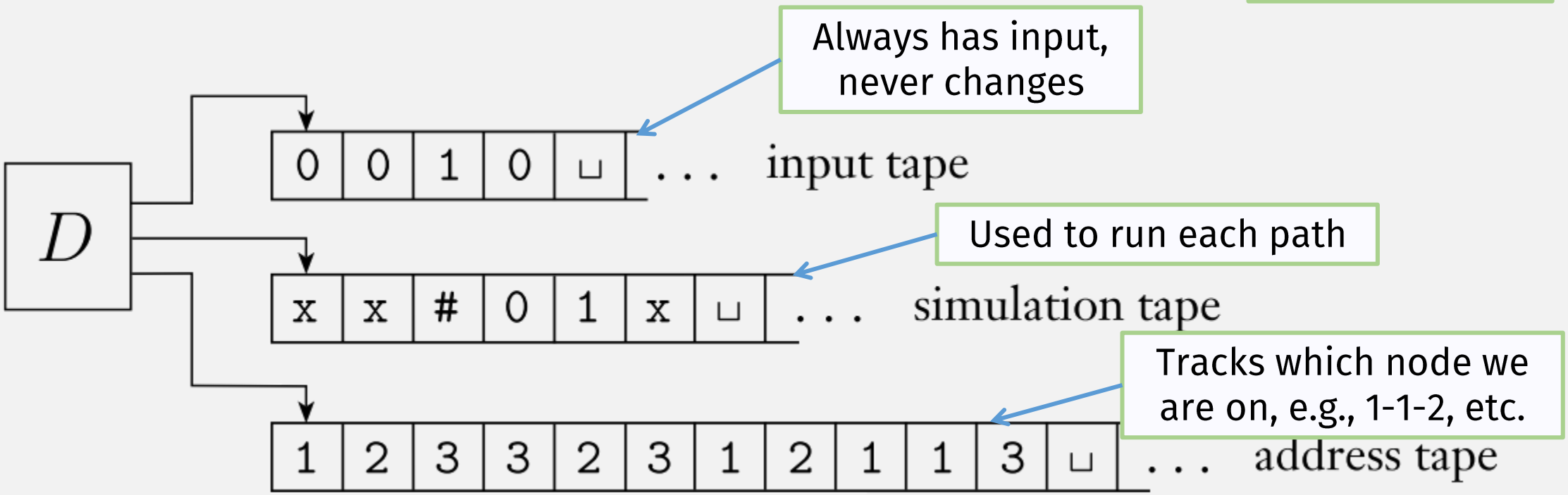
Nondeterministic TM → Deterministic: 2nd way

- Deterministic TM simulating nondeterministic TM:
  - Number the nodes at each step
  - Deterministically check every tree path, in breadth-first order
    - 1
    - 1-1
    - 1-2
    - 1-1-1
    - 1-1-2
    - and so on
  - Accept if accepting config found



Nondeterministic TM → Deterministic: 2nd way

Needs 3 tapes



Nondeterministic TM ⇔ Deterministic TM

- => If a deterministic TM recognizes a language, then a nondeterministic TM recognizes the language
  - Deterministic TM → nondeterministic TM
  - Wrap output of delta in a set

- <= If a nondeterministic TM recognizes a language, then a deterministic TM recognizes the language
  - Nondeterministic TM → deterministic TM (**DONE!**)

All Equivalent TMs!

- Single-tape Turing Machine
- Multi-tape Turing Machine
- Nondeterministic Turing Machine

Check-in Quiz 10/21

On gradescope

End of Class Survey 10/21

See course website