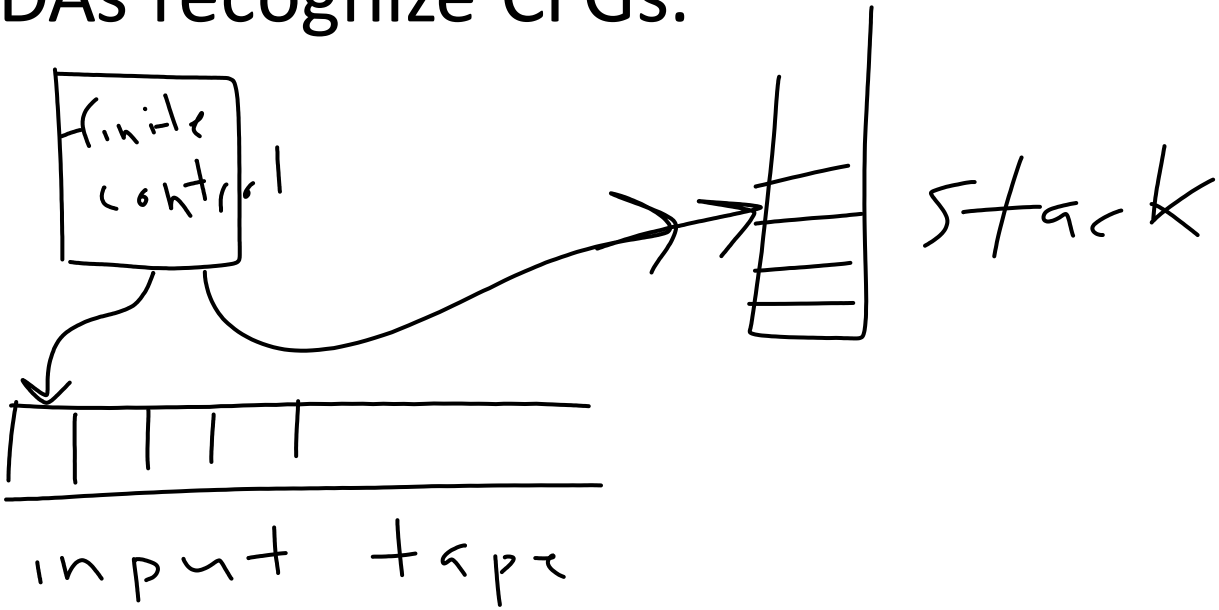
Modeling Computation

Introduction to Formal Languages and Automata

Pushdown Automata

Pushdown Automata (PDAs)

PDAs recognize CFGs.



Pushdown Automata (PDAs)

$A = (Q, \Sigma, \Gamma, \Delta, s_0, Z_0, F)$

Q = states

Σ = input alphabet

Γ = stack symbols

$\Delta \subseteq (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma^*) \times (Q \times \Gamma^*)$

s_0 initial state

Z_0 empty/bottom of stack symbol

F set of favorable

$$\big((s, a, \beta), (q, \gamma)\big) \in \Delta$$

If PDA

- is in state s
- reads a from tape $a \in \Sigma \cup \{\varepsilon\}$
- pop β from stack

Then

- move to state q
- push γ to the stack

PDA Acceptance

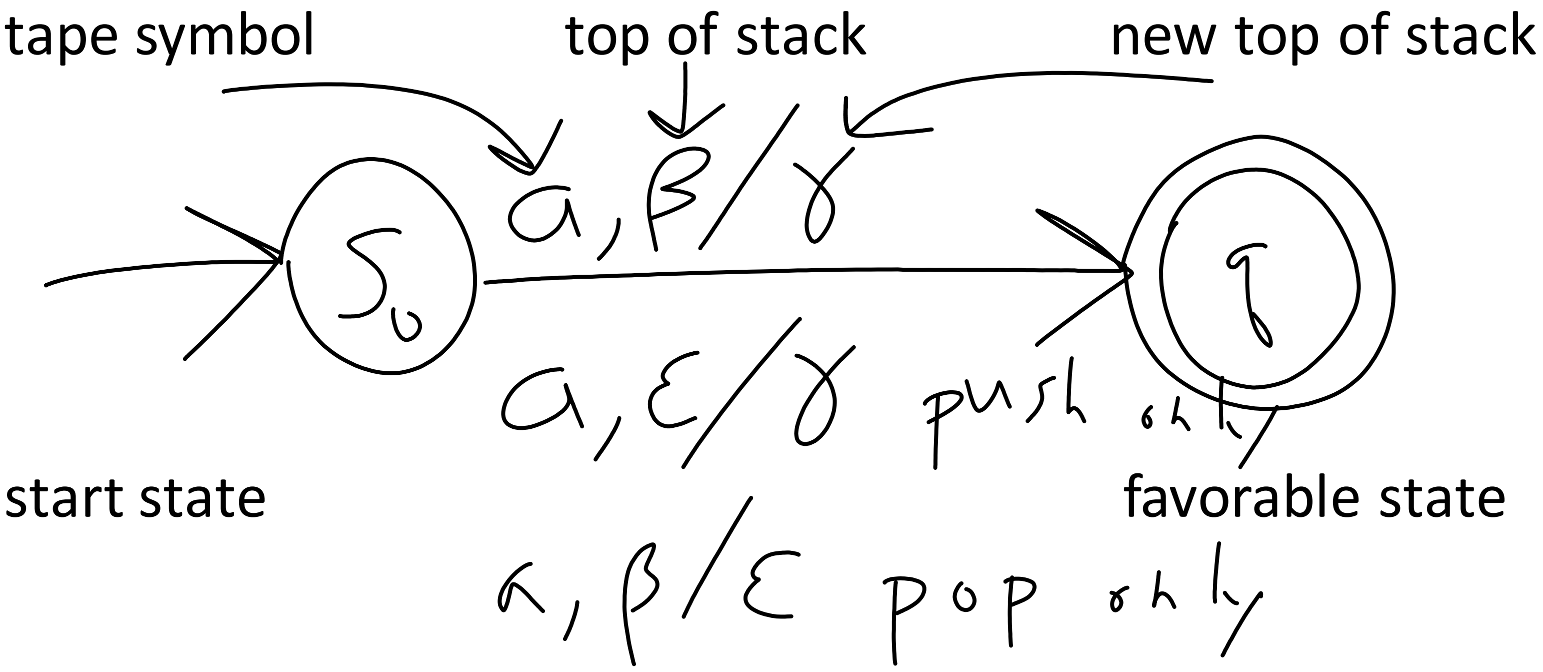
PDA A accepts a string w if there exists a sequence of transitions such that

all symbols of input have been read

AND (the stack is empty

OR A is in a favorable state)

Graphical Representation of PDA



Edge label $a, \beta / \gamma$

$a \in (\Sigma \cup \{\epsilon\})$ is read from tape

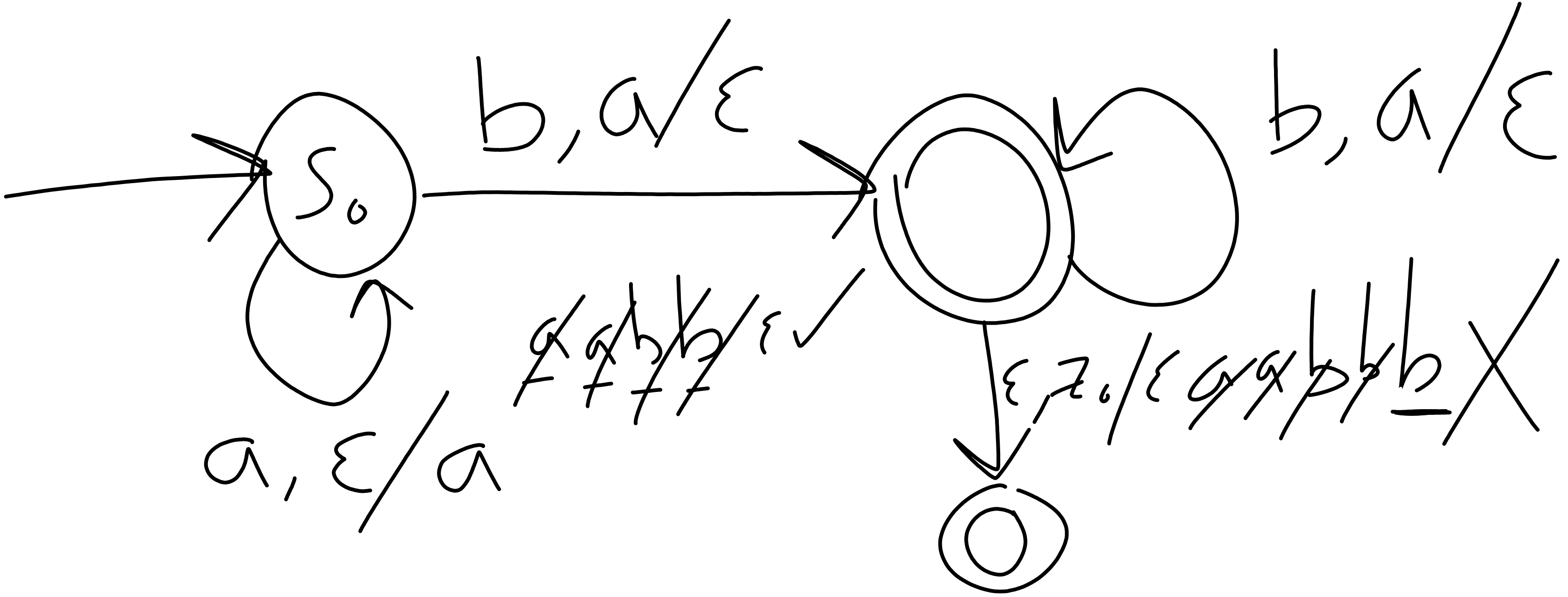
$\beta \in \Gamma^*$ is popped from stack

$\gamma \in \Gamma^*$ pushed to stack

Example: PDA that accepts

$$L = \{a^n b^n \mid n > 0\}$$

Idea: push "a"s to stack then, for each "b", pop "a" from stack.



Nondeterministic PDAs

$((s, a, \beta), (q, \gamma))$ and $((s, a, \beta), (p, \delta))$

Or $((s, \varepsilon, \beta), (q, \gamma))$

Some languages can be accepted by a nondeterministic PDA, but not by a deterministic PDA.

Ex: $L = \{ww^R \mid w \in \Sigma^*\}$

Some languages are not context-free, so no PDA can recognize them.

Ex: $L = \{a^n b^n c^n \mid n > 0\}$

Construct PDA to recognize $L = \{ww^R \mid w \in \Sigma^*\}$

Construct a PDA that recognizes strings that have an equal number of 0s and 1s.

Construct a CFG that generates strings that have an equal number of 0s and 1s.