Context-free languages and Pushdown Automata

Sipser 2 (pages 99-115)

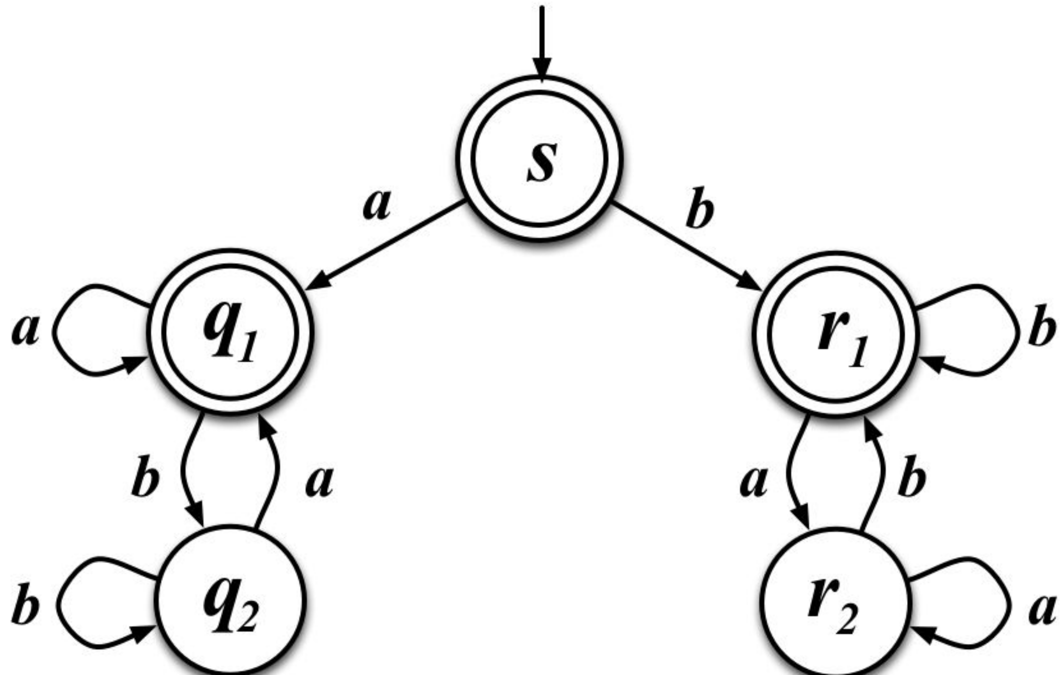
Last time…

Context-free grammars

- A context-free grammar G is a quadruple (V, Σ, R, S), where
  - V is a finite set called the **variables**
  - **Σ is a finite set, disjoint from V, called the terminals**
  - **R is a finite subset of $V \times (V \cup \Sigma)^*$ called the rules**
  - **$S \in V$ is called the start symbol**
- **For any $A \in V$ and $u \in (V \cup \Sigma)^*$, we write $A \to_G u$ whenever $(A, u) \in R$**

Context Free Languages
vs
Regular Languages

Regular languages are context-free



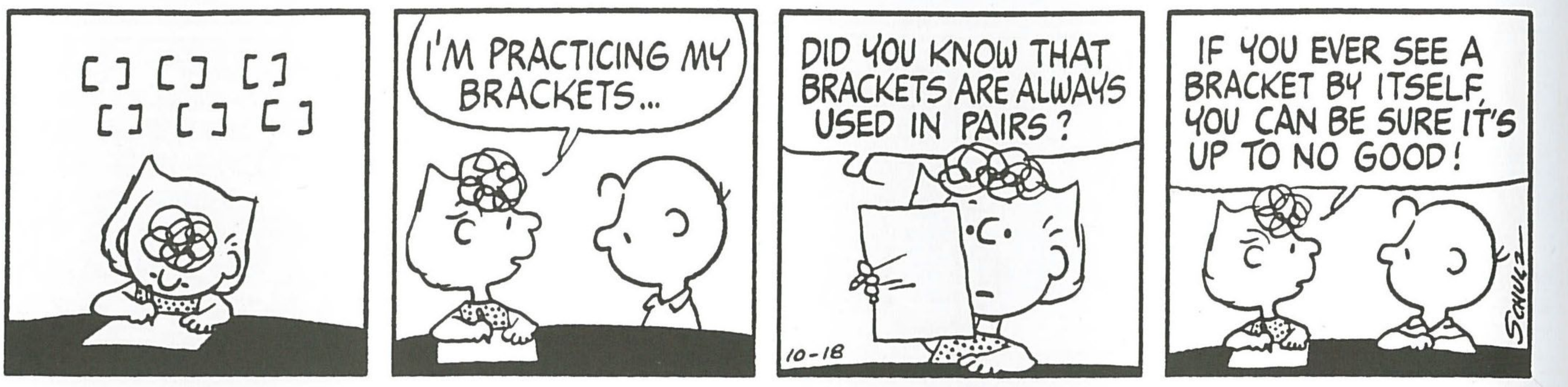
$G = (V, \Sigma, R, S)$, where

$V = \{S, Q_1, Q_2, R_1, R_2\}$

$\Sigma = \{a, b\}$

$R = \{S \to aQ_1,\ Q_1 \to aQ_1,\ Q_1 \to bQ_2,\ Q_2 \to bQ_2,\ Q_2 \to aQ_1,$
$S \to bR_1,\ R_1 \to bR_1,\ R_1 \to aR_2,\ R_2 \to aR_2,\ R_2 \to bR_1,$
$S \to \varepsilon,\ Q_1 \to \varepsilon,\ R_1 \to \varepsilon\}$

What about...?


[ ] [ ] [ ]
[ ] [ ] [ ]
I'M PRACTICING MY BRACKETS...
DID YOU KNOW THAT BRACKETS ARE ALWAYS USED IN PAIRS?
IF YOU EVER SEE A BRACKET BY ITSELF, YOU CAN BE SURE IT'S UP TO NO GOOD!
10-18
SCHULZ

Balanced Brackets

- The grammar $G = (V, \Sigma, R, S)$, where

  $V = \{S\}$

  $\Sigma = \{[,]\}$

  $R = \{ \; S \rightarrow_G \varepsilon,$

  $\qquad S \rightarrow_G SS,$

  $\qquad S \rightarrow_G [S] \}$

  generates all strings of balanced brackets

- Is the language L(G) is regular?
  - Why/Why not?

Recognizing Context-Free Languages

- Grammars are language generators. It is not immediately clear how they might be used as language recognizers.

Recognizing Context-Free Languages

- Grammars are language generators. It is not immediately clear how they might be used as language recognizers.
- The language L(G) of balanced brackets is not regular. It cannot be recognized by a finite state automaton.

Recognizing Context-Free Languages

- Grammars are language generators. It is not immediately clear how they might be used as language recognizers.
- The language L(G) of balanced brackets is not regular. It cannot be recognized by a finite state automaton.
- However, it is very similar to the

  { }

  blocks of some programming languages and, therefore, must be recognizable by some compiler or interpreter!

Auxiliary storage

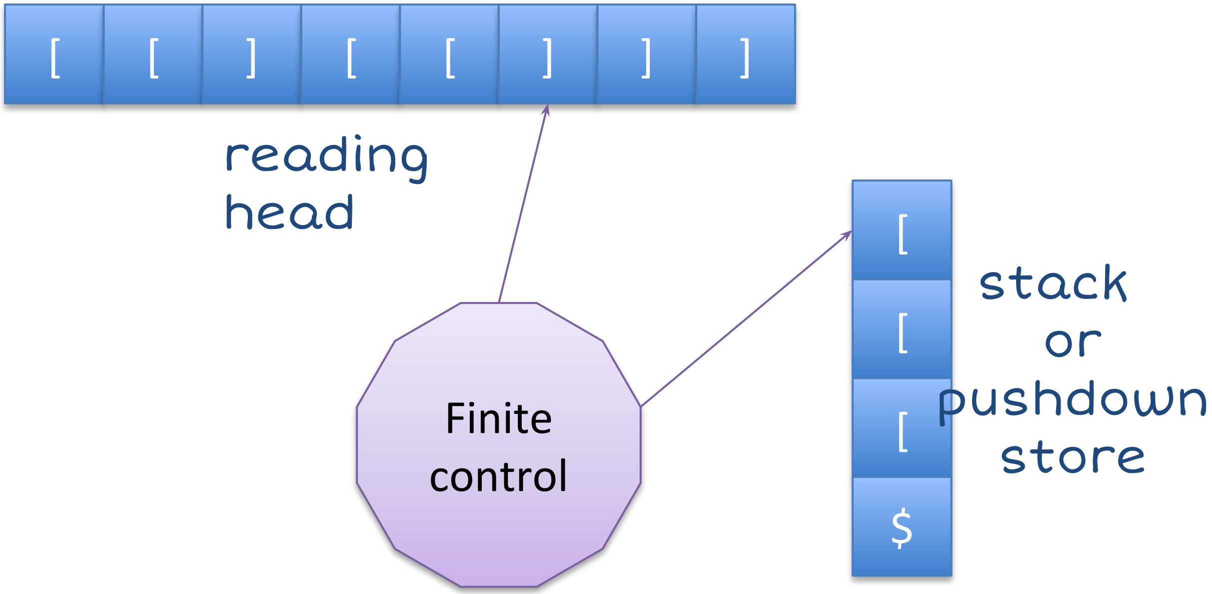
- We could recognize the language L(G) of balanced brackets by reading left to right, if we could remember left brackets along the way.

[[ ] [[ ]]]

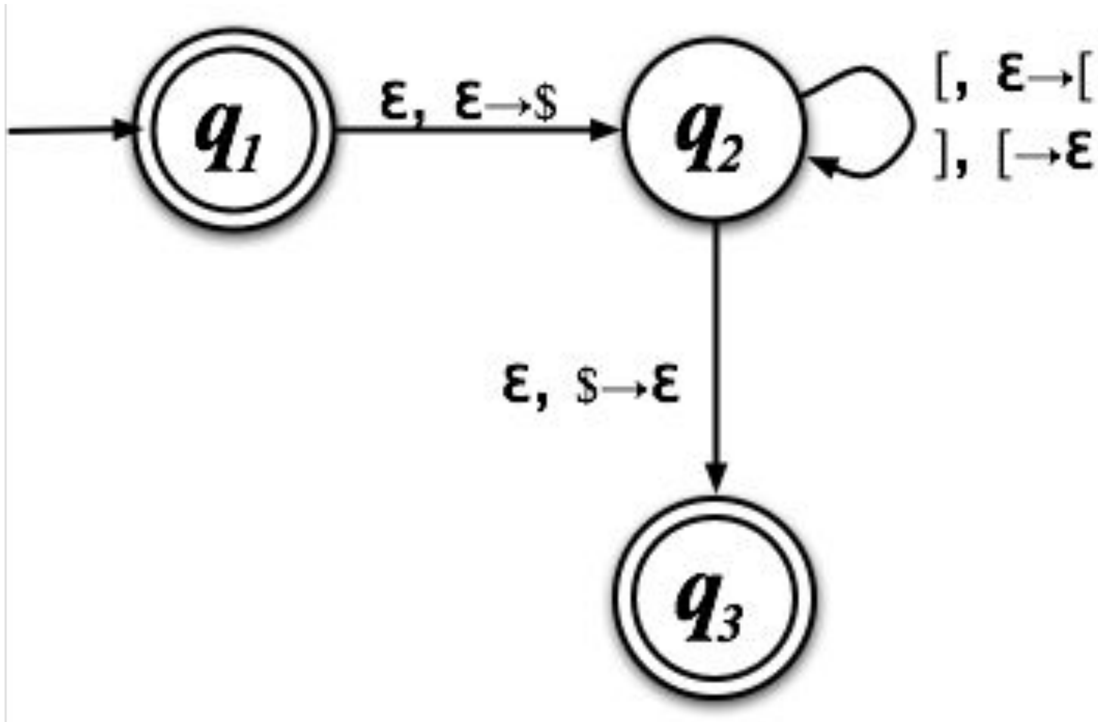
Must match some left bracket along the way

Pushdown Automata

- The last left bracket seen matches the first right bracket. This suggests a stack storage mechanism.



Describing a pushdown machine

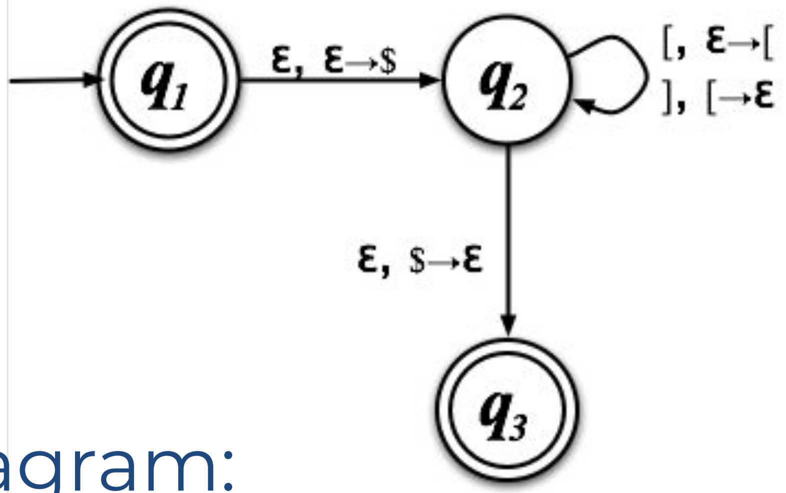


Formally…

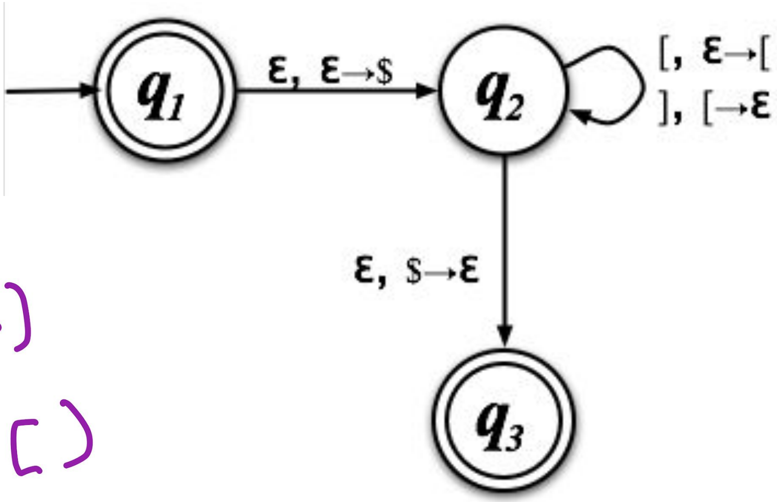
- A pushdown automaton is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where
  - Q is a finite set of states
  - Σ is a finite alphabet (the input symbols)
  - Γ is a finite alphabet (the stack symbols)
  - $\delta: (Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon) \rightarrow P(Q \times \Gamma_\varepsilon)$ is the transition function
  - $q_0 \in Q$ is the initial state, and
  - $F \subseteq Q$ is the set of accept states

Balanced brackets

- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where
  - $Q = \{q_1, q_2, q_3\}$
  - $\Sigma = \{[,]\}$
  - $\Gamma = \{[, \$\}$
  - $q_0 = q_1$
  - $F = \{q_1, q_3\}$
  - δ is given by state diagram:



PDA computation



[ ]

stack tos

$(q_1, [\,], \varepsilon) \xrightarrow{\varepsilon} (q_2, [\,], \$)$

state

input left to read

$\xrightarrow{[} (q_2,], \$[)$

$\xrightarrow{]} (q_2, \varepsilon, \$)$

$\xrightarrow{\varepsilon} (q_3, \varepsilon, \varepsilon)$

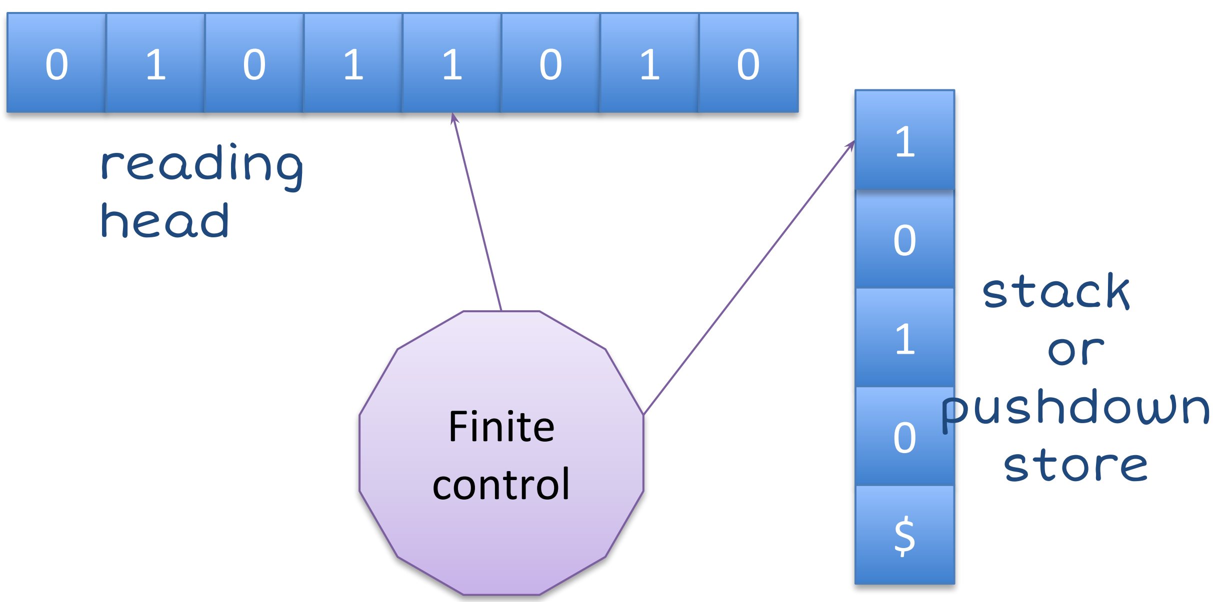
$\in F$ $\quad == \varepsilon$

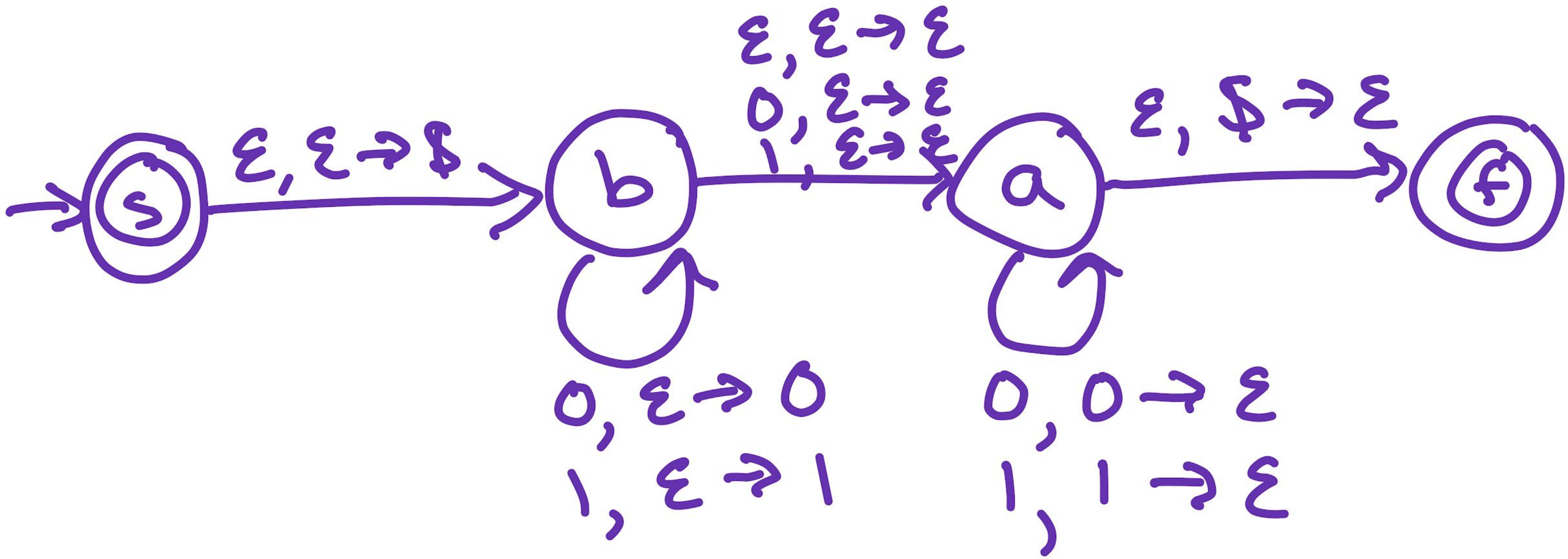
accept

Pushdown automata are nondeterministic

- Build a machine to recognize

$$PAL = \{w \mid w \in \{0, 1\}^* \text{ and } w = w^R\}$$




s
ε, ε → $
b
ε, ε → ε
0, ε → ε
1, ε → ε
a
ε, $ → ε
f
0, ε → 0
1, ε → 1
0, 0 → ε
1, 1 → ε