Pushdown Automata

Lin Chen

Email: Lin.Chen@ttu.edu

Grader: zulfi.khan@ttu.edu

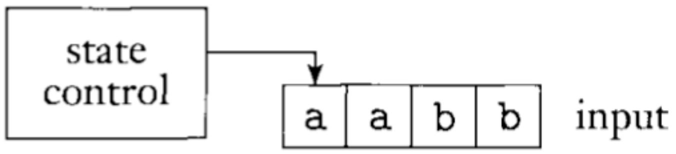
TEXAS TECH
UNIVERSITY

Pushdown automata

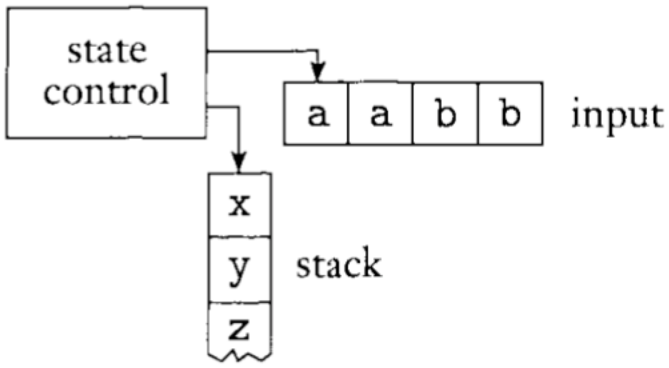
- Regular expressions are string generators
- Finite Automata (DFA, NFA) are string acceptors of REG
- CFGs are string generators
- What is the string acceptor of CFG?
  - Pushdown automata

Pushdown automata

Finite automata:

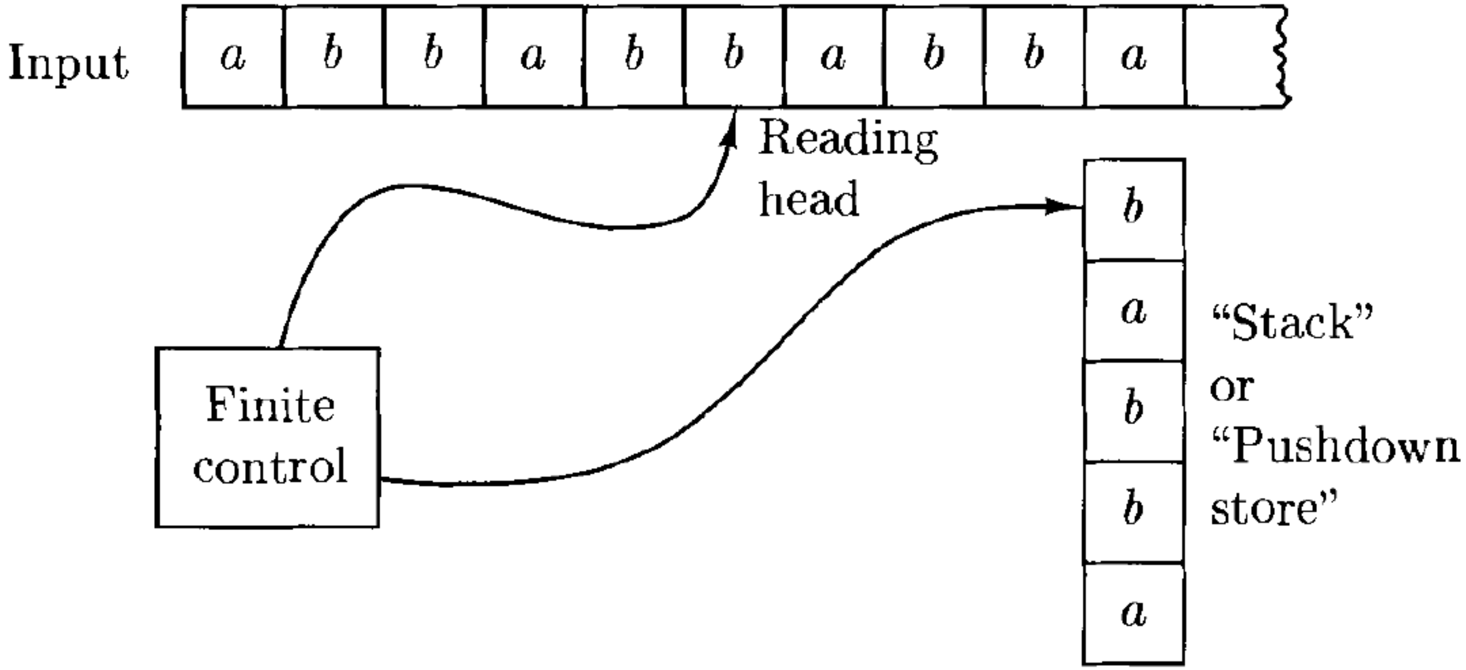


Pushdown automata:



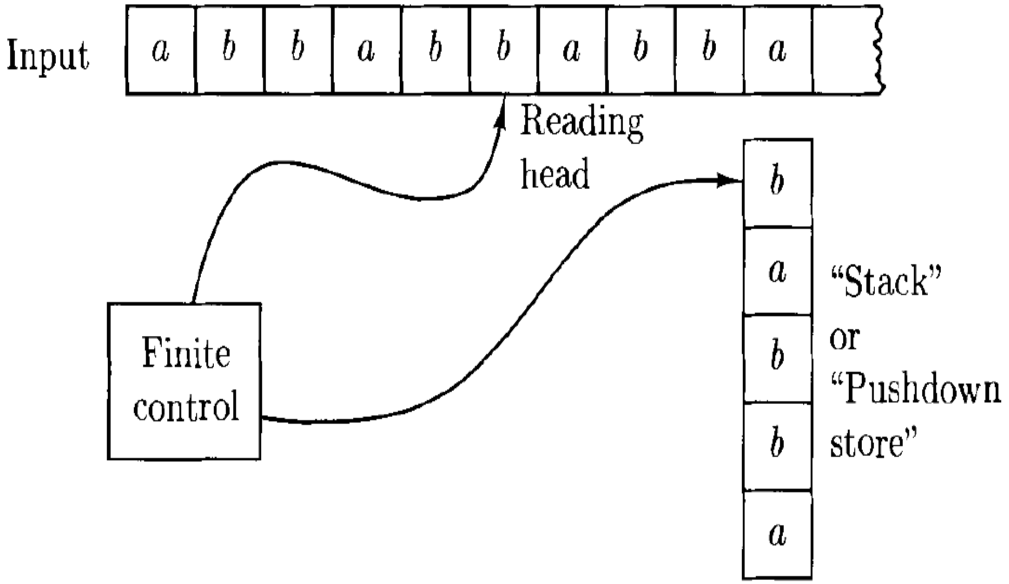
Pushdown automata

- Finite automata cannot accept $\{ww^R : w \in \{a, b\}^*\}$ because it requires some memory
- We can use a stack as memory



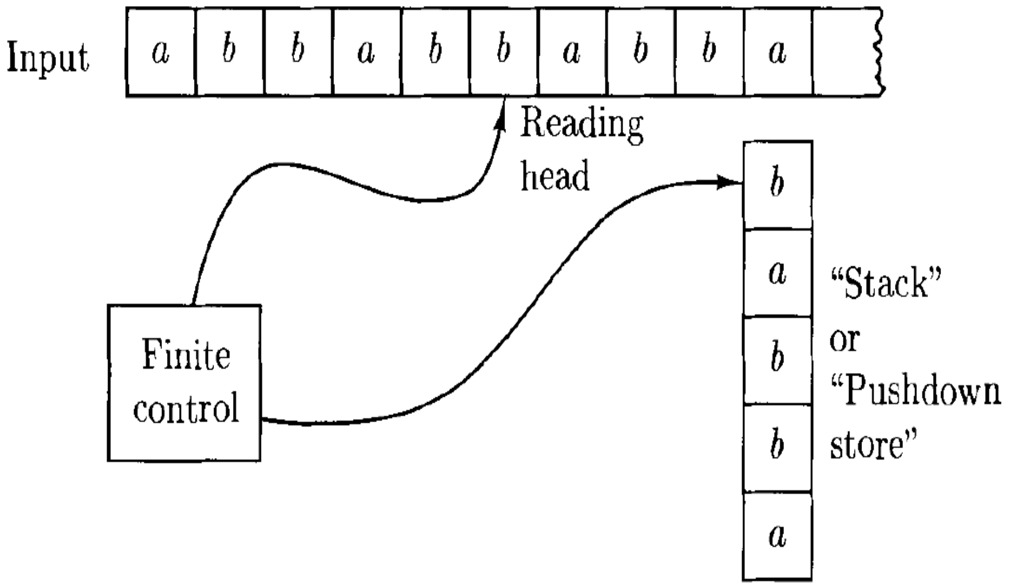
Pushdown automata

- Finite automata cannot accept $\{ww^R : w \in \{a,b\}^*\}$ because it requires some memory
- We can use a stack as memory



Pushdown automata

- Finite automata cannot accept $\{ww^R : w \in \{a, b\}^*\}$ because it requires some memory
- We can use a stack as memory

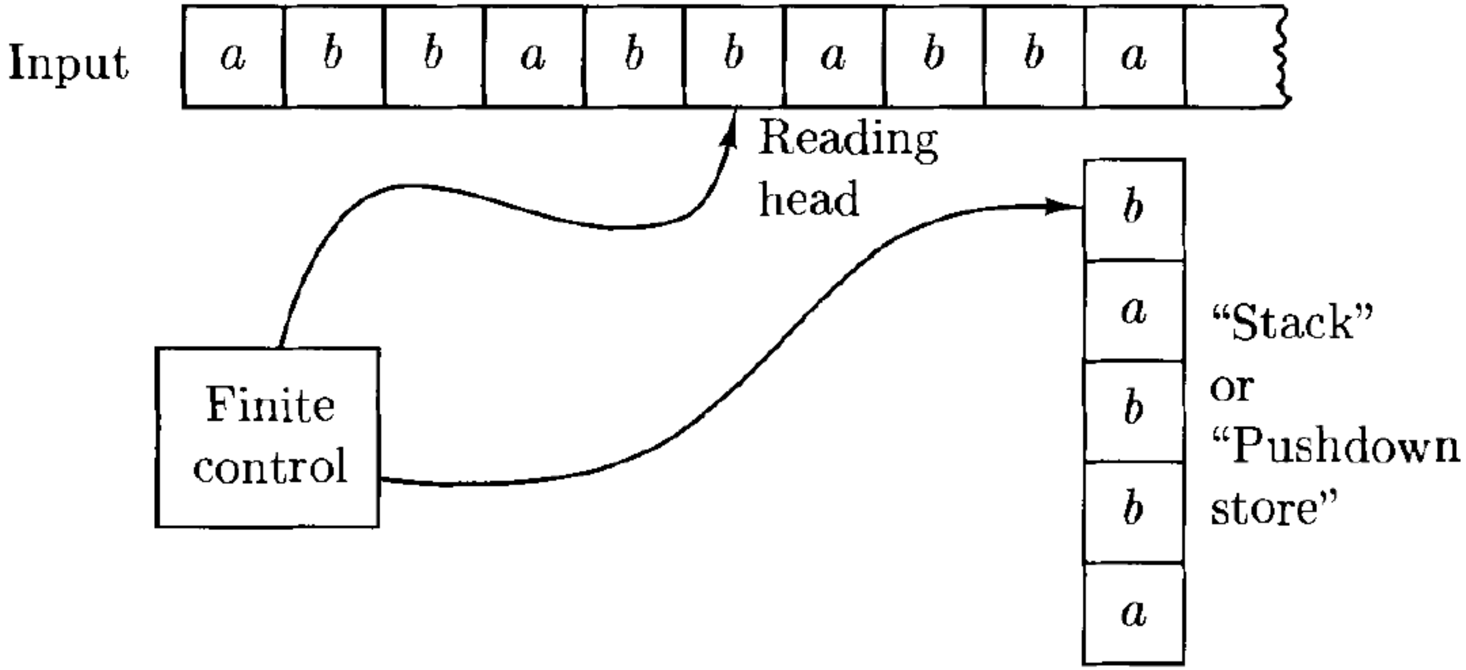


Writing a symbol on stack: Push
Removing a symbol from stack: pop

Pushdown automata

- How a stack is used?

  - We continue pushing symbols into stack, and then let it pop out at a suitable time.

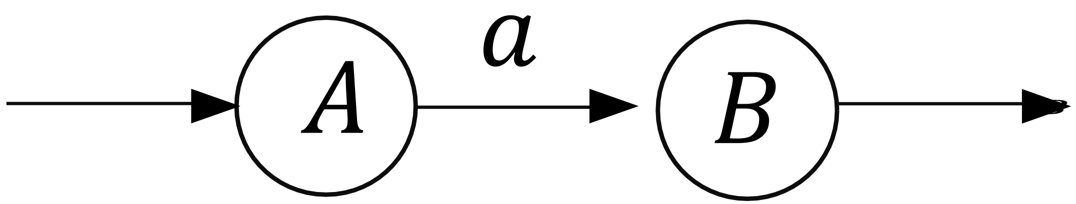


Pushdown automata

- How a stack is used?

  - We continue pushing symbols into stack, and then let it pop out at a suitable time.

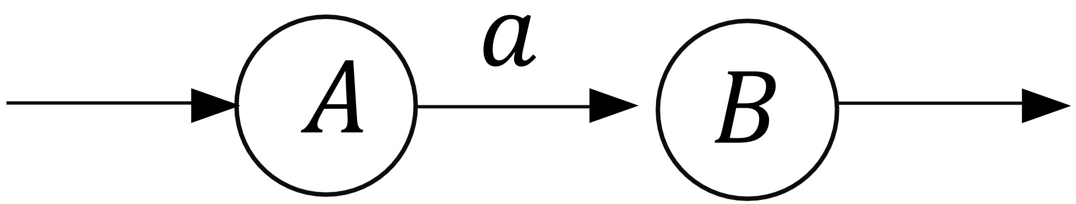
  - $A \to aB$ can be easily simulated by a DFA



Pushdown automata

- How a stack is used?

- $A \to aB$ can be easily simulated by a DFA



- What about $A \to aBb$
  - After we reach the final state from B, we need to "remember" to append an a
  - We push b to the stack, and eventually b will pop up

Pushdown automata

Definition 3.3.1: Let us define a **pushdown automaton** to be a sextuple $M = (K, \Sigma, \Gamma, \Delta, s, F)$, where

K is a finite set of **states**,
Σ is an alphabet (the **input symbols**),
Γ is an alphabet (the **stack symbols**),
$s \in K$ is the **initial state**,
$F \subseteq K$ is the set of **final states**, and
Δ, the **transition relation**, is a finite subset of $(K \times (\Sigma \cup \{e\}) \times \Gamma^*) \times (K \times \Gamma^*)$.

Pushdown automata

- How PDA (Pushdown automata) works?
- If $\big((p, a, \beta), (q, \gamma)\big) \in \Delta$, then the PDA M, once it is in state p with β at the top of the stack, may read a from he input tape, replace β by γ on top of the stack, and enter state q.
  - $\big((p, a, e), (q, \gamma)\big)$ reads a and pushes γ
  - $\big((p, a, \gamma), (q, e)\big)$ reads a and pops γ

Pushdown automata

Definition 3.3.1: Let us define a **pushdown automaton** to be a sextuple $M = (K, \Sigma, \Gamma, \Delta, s, F)$, where

K is a finite set of **states**,
Σ is an alphabet (the **input symbols**),
Γ is an alphabet (the **stack symbols**),
$s \in K$ is the **initial state**,
$F \subseteq K$ is the set of **final states**, and
Δ, the **transition relation**, is a finite subset of $(K \times (\Sigma \cup \{e\}) \times \Gamma^*) \times (K \times \Gamma^*)$.

You read input symbols one by one, but replace stack symbols a bunch by a bunch

Pushdown automata

- Configuration
  - $(q, w, \lambda) \in K \times \Sigma^* \times \Gamma^*$
  - current state q
  - the remainder of the string w
  - strings consisting of the stack symbol in the stack, top-down

Pushdown automata

- Yields (in one step)
  - $(p, x, \alpha) \vdash_M (q, y, \zeta)$ if exists transition $\big((p, a, \beta), (q, \gamma)\big) \in \Delta$
    - $x = ay$
    - $\alpha = \beta\eta$ and $\zeta = \gamma\eta$ for some $\eta \in \Gamma^*$
  - $\vdash_M^*$ indicates a sequence of $\vdash_M$ (reflexive and transitive closure)

Pushdown automata

- Acceptance

  \- PDA M accepts a string $w \in \Sigma^*$ if and only if $(s, w, e) \vdash_M^* (f, e, e)$ for some final state $f \in F$

Pushdown automata

Example 3.3.1: Let us design a pushdown automaton M to accept the language $L = \{wcw^R : w \in \{a,b\}^*\}$. For example, $ababcbaba \in L$, but $abcab \notin L$, and $cbc \notin L$. We let $M = (K, \Sigma, \Gamma, \Delta, s, F)$, where $K = \{s, f\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{a, b\}$, $F = \{f\}$, and Δ contains the following five transitions.

(1) $((s, a, e), (s, a))$
(2) $((s, b, e), (s, b))$
(3) $((s, c, e), (f, e))$
(4) $((f, a, a), (f, e))$
(5) $((f, b, b), (f, e))$

- State diagram on blackboard

Pushdown automata

State	Unread Input	Stack	Transition Used
s	$abbcbba$	e	-
s	$bbcbba$	a	1
s	$bcbba$	ba	2
s	$cbba$	bba	2
f	bba	bba	3
f	ba	ba	5
f	a	a	5
f	e	e	4

- State diagram walk on blackboard

Pushdown automata

Example 3.3.3: This pushdown automaton accepts the language $\{w \in \{a,b\}^* : w \text{ has the same number of } a\text{'s and } b\text{'s}\}$. Either a string of a's or a string of b's is kept by M on its stack. A stack of a's indicates the excess of a's over b's thus far read, if in fact M has read more a's than b's; a stack of b's indicates the excess of b's over a's. In either case, M keeps a special symbol c on the bottom of the stack as a marker. Let $M = (K, \Sigma, \Gamma, \Delta, s, F)$, where $K = \{s, q, f\}$, $\Sigma = \{a, b\}$, $\Gamma = \{a, b, c\}$, $F = \{f\}$, and Δ is listed below.

(1) $((s, e, e), (q, c))$
(2) $((q, a, c), (q, ac))$
(3) $((q, a, a), (q, aa))$
(4) $((q, a, b), (q, e))$
(5) $((q, b, c), (q, bc))$
(6) $((q, b, b), (q, bb))$
(7) $((q, b, a), (q, e))$
(8) $((q, e, c), (f, e))$

- State diagram on blackboard

Pushdown automata

State	Unread Input	Stack	Transition	Comments
s	$abbbabaa$	e	-	Initial configuration.
q	$abbbabaa$	c	1	Bottom marker.
q	$bbbabaa$	ac	2	Start a stack of a's.
q	$bbabaa$	c	7	Remove one a.
q	$babaa$	bc	5	Start a stack of b's.
q	$abaa$	bbc	6	
q	baa	bc	4	
q	aa	bbc	6	
q	a	bc	4	
q	e	c	4	
f	e	e	8	Accepts.

- State diagram walk on blackboard

PDA and CFG

- PDA serves as a checker for context free language

Theorem 3.4.1: *The class of languages accepted by pushdown automata is exactly the class of context-free languages.*

PDA and CFG

Lemma 3.4.1: *Each context-free language is accepted by some pushdown automaton.*

$M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$

- Two states, $\{p, q\}$
- Stack alphabet = terminals + nonterminals
- Transitions:

(1) $((p, e, e), (q, S))$
(2) $((q, e, A), (q, x))$ for each rule $A \rightarrow x$ in R.
(3) $((q, a, a), (q, e))$ for each $a \in \Sigma$.

PDA and CFG

Example 3.4.1: Consider the grammar $G = (V, \Sigma, R, S)$ with $V = \{S, a, b, \mathrm{c}\}$, $\Sigma = \{a, b, c\}$, and $R = \{S \to aSa, S \to bSb, S \to c)$, which generates the language $\{wcw^R : w \in \{a, b\}^*\}$. The corresponding pushdown automaton, according to the construction above, is $M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$, with

$$\begin{aligned}
\Delta = \{&((p, e, e), (q, S)), && (T1)\\
&((q, e, S), (q, aSa)), && (T2)\\
&((q, e, S), (q, bSb)), && (T3)\\
&((q, e, S), (q, c)), && (T4)\\
&((q, a, a), (q, e)), && (T5)\\
&((q, b, b), (q, e)), && (T6)\\
&((q, \mathrm{c}, \mathrm{c}), (q, e))\} && (T7).
\end{aligned}$$

PDA and CFG

Lemma 3.4.2: *If a language is accepted by a pushdown automaton, it is a context-free language.*

Given a PDA, modify it such that:

1. It has a single accept state, q_{accept}.
2. It empties its stack before accepting.
3. Each transition either pushes a symbol onto the stack (a *push* move) or pops one off the stack (a *pop* move), but it does not do both at the same time.

PDA and CFG

Lemma 3.4.2: *If a language is accepted by a pushdown automaton, it is a context-free language.*

Given a PDA, modify it such that:

It has a single accept state, q_{accept}.

Each transition either pushes a symbol onto the stack (a *push* move) or pops one off the stack (a *pop* move), but it does not do both at the same time.

If $\big((p, a, \beta), (q, \gamma)\big) \in \Delta$, replace it with $\big((p, a, \beta), (p', e)\big)$ and $\big((p', e, e), (q, \gamma)\big)$

PDA and CFG

Lemma 3.4.2: *If a language is accepted by a pushdown automaton, it is a context-free language.*

We design a CFG such that A_{pq} generates all strings that take M from state p to state q, starting and ending with empty stack.

PDA and CFG

Lemma 3.4.2: *If a language is accepted by a pushdown automaton, it is a context-free language.*

We design a CFG such that A_{pq} generates all strings that take M from state p to state q, starting and ending with empty stack.

- When M reads any string of A_{pq}, the first move is push, the last move is pop.

PDA and CFG

Lemma 3.4.2: *If a language is accepted by a pushdown automaton, it is a context-free language.*

We design a CFG such that A_{pq} generates all strings that take M from state p to state q, starting and ending with empty stack.

- When M reads any string of A_{pq}, the first move is push, the last move is pop.
- If the first push and last pop is the same symbol, add $A_{pq} \rightarrow aA_{rs}b$

PDA and CFG

Lemma 3.4.2: *If a language is accepted by a pushdown automaton, it is a context-free language.*

We design a CFG such that A_{pq} generates all strings that take M from state p to state q, starting and ending with empty stack.

- When M reads any string of A_{pq}, the first move is push, the last move is pop.
- If the first push and last pop is the same symbol, add $A_{pq} \rightarrow aA_{rs}b$
- If the first push and last pop is different, add $A_{pq} \rightarrow A_{pr}A_{rq}$

PDA and CFG

Lemma 3.4.2: *If a language is accepted by a pushdown automaton, it is a context-free language.*

Formal construction:

- If $\big((p,a,e),(r,\beta)\big)$, $((s,b,\beta),(q,e)) \in \Delta$, add rule $A_{pq} \to aA_{rs}b$
- For all states p,r,q, add $A_{pq} \to A_{pr}A_{rq}$
- For all state p, add $A_{pp} \to e$