Pushdown Automata

Chapter 12

Recognizing Context-Free Languages

We need a device similar to an FSM except that it needs more power.

The insight: Precisely what it needs is a stack, which gives it an unlimited amount of memory with a restricted structure.

Example: Bal (the balanced parentheses language)

(((())))

Definition of a Pushdown Automaton

$M = (K, \Sigma, \Gamma, \Delta, s, A)$, where:

- K is a finite set of states
- Σ is the input alphabet
- Γ is the stack alphabet
- $s \in K$ is the initial state
- $A \subseteq K$ is the set of accepting states, and
- Δ is the transition relation. It is a finite subset of

$$\underbrace{(K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^*)} \times \underbrace{(K \times \Gamma^*)}$$

state — input or ε — string to pop from top of stack — state — string of to push on top of stack

Definition of a Pushdown Automaton

A configuration of M is an element of $K \times \Sigma^* \times \Gamma^*$.

The initial configuration of M is (s, w, ε).

Yields

Let $c \in \Sigma \cup \{\varepsilon\}$, $\gamma_1, \gamma_2, \gamma \in \Gamma^*$, and $w \in \Sigma^*$.

Then:
$(q_1, cw, \gamma_1\gamma) \vdash_M (q_2, w, \gamma_2\gamma)$ iff $((q_1, c, \gamma_1), (q_2, \gamma_2)) \in \Delta$.

Let $\vdash_M^*$ be the reflexive, transitive closure of $\vdash_M$.

C_1 ***yields*** configuration C_2 iff $C_1 \vdash_M^* C_2$

Computations

A ***computation*** by M is a finite sequence of configurations $C_0, C_1, \ldots, C_n$ for some $n \geq 0$ such that:

- C_0 is an initial configuration,
- C_n is of the form (q, ε, γ), for some state $q \in K_M$ and some string γ in Γ^*, and
- $C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_n$.

Nondeterminism

If M is in some configuration (q_1, s, γ) it is possible that:

- Δ contains exactly one transition that matches.
- Δ contains more than one transition that matches.
- Δ contains no transition that matches.

Accepting

A computation C of M is an ***accepting computation*** iff:

- $C = (s, w, \varepsilon) \vdash_M^* (q, \varepsilon, \varepsilon)$, and
- $q \in A$.

M ***accepts*** a string w iff at least one of its computations accepts.

Other paths may:
- Read all the input and halt in a nonaccepting state,
- Read all the input and halt in an accepting state with the stack not empty,
- Loop forever and never finish reading the input, or
- Reach a dead end where no more input can be read.

The ***language accepted by*** M, denoted $L(M)$, is the set of all strings accepted by M.

Rejecting

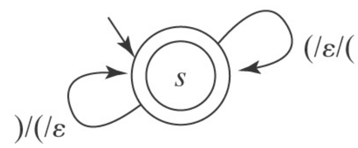
A computation C of M is a ***rejecting computation*** iff:

- $C = (s, w, \varepsilon) \vdash_M^* (q, w', \alpha)$,
- C is not an accepting computation, and
- M has no moves that it can make from (q, ε, α).

M ***rejects*** a string *w* iff all of its computations reject.

So note that it is possible that, on input *w*, *M* neither accepts nor rejects.

A PDA for Balanced Parentheses



$M = (K, \Sigma, \Gamma, \Delta, s, A)$, where:

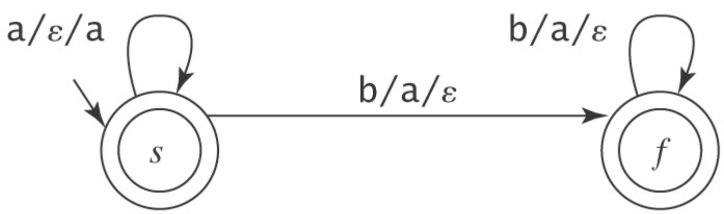
$K = \{s\}$	the states
$\Sigma = \{ (,) \}$	the input alphabet
$\Gamma = \{ (\}$	the stack alphabet
$A = \{s\}$	

Δ contains:

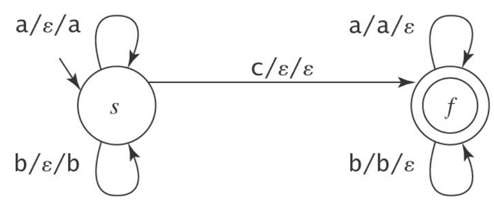
$((s, (, \varepsilon^{¶}), (s, ( ))$
$((s,), ( ), (s, \varepsilon))$

¶ This does not mean that the stack is empty

A PDA for $A^nB^n = \{a^nb^n: n \geq 0\}$



A PDA for $\{wcw^R: w \in \{a, b\}^*\}$

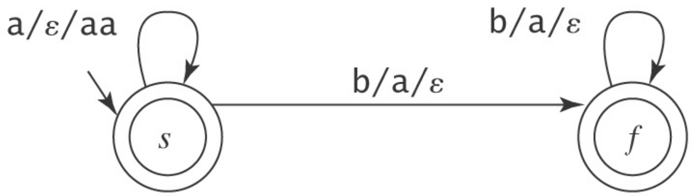


$M = (K, \Sigma, \Gamma, \Delta, s, A)$, where:

$K = \{s, f\}$	the states
$\Sigma = \{a, b, c\}$	the input alphabet
$\Gamma = \{a, b\}$	the stack alphabet
$A = \{f\}$	the accepting states

Δ contains:
$((s, a, \varepsilon), (s, a))$
$((s, b, \varepsilon), (s, b))$
$((s, c, \varepsilon), (f, \varepsilon))$
$((f, a, a), (f, \varepsilon))$
$((f, b, b), (f, \varepsilon))$

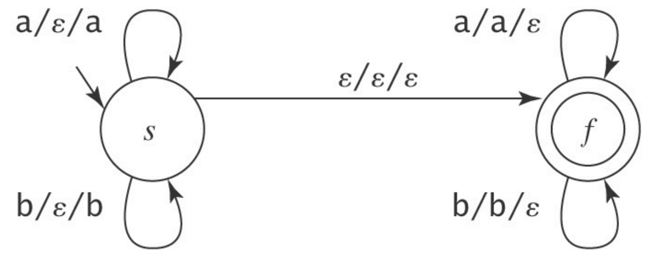
A PDA for $\{a^n b^{2n}: n \geq 0\}$



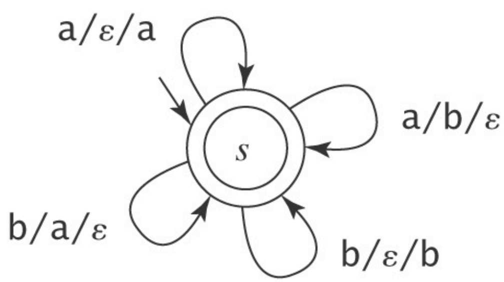
A PDA for PalEven $=\{ww^R: w \in \{a, b\}^*\}$

$S \rightarrow \varepsilon$
$S \rightarrow aSa$
$S \rightarrow bSb$

A PDA:



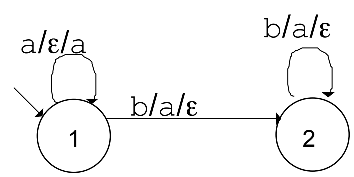
A PDA for $\{w \in \{a, b\}^* : \#_a(w) = \#_b(w)\}$



Accepting Mismatches

$L = \{a^m b^n : m \neq n; m, n > 0\}$

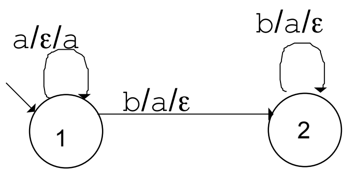
Start with the case where $n = m$:



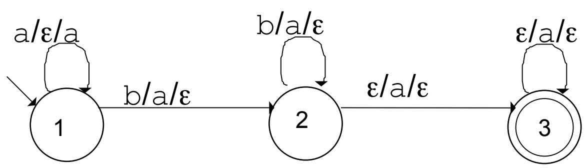
- If stack and input are empty, halt and reject.
- If input is empty but stack is not ($m > n$) (accept):
- If stack is empty but input is not ($m < n$) (accept):

Accepting Mismatches

$L = \{a^m b^n : m \neq n;\ m, n > 0\}$

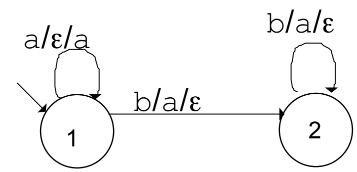


- If input is empty but stack is not ($m > n$) (accept):

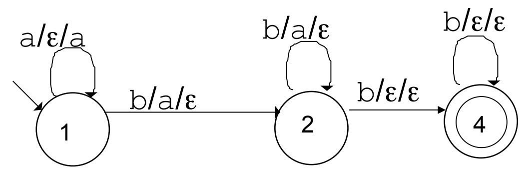


Accepting Mismatches

$L = \{a^m b^n : m \neq n;\ m, n > 0\}$

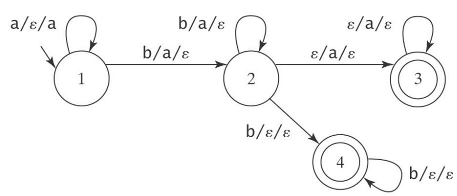


- If stack is empty but input is not ($m < n$) (accept):



Putting It Together

$L = \{a^m b^n : m \neq n;\ m, n > 0\}$



- Jumping to the input clearing state 4:
  Need to detect bottom of stack.

- Jumping to the stack clearing state 3:
  Need to detect end of input.

$A^nB^nC^n$ vs $\neg A^nB^nC^n$

Consider $A^nB^nC^n = \{a^n b^n c^n : n \geq 0\}$.

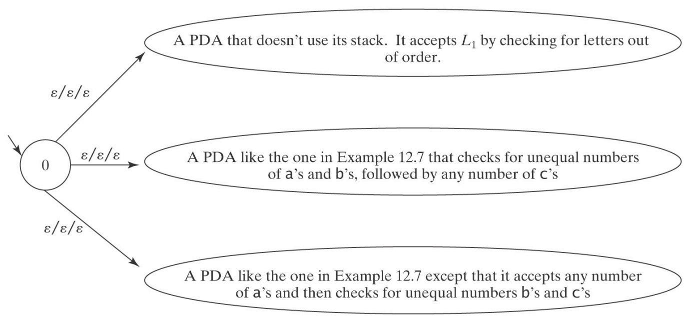
PDA for $A^nB^nC^n$?

Now consider $L = \neg A^nB^nC^n$. L is the union of two languages:

1. $\{w \in \{a, b, c\}^* :$ the letters are out of order$\}$, and

2. $\{a^i b^j c^k : i, j, k \geq 0$ and $(i \neq j$ or $j \neq k)\}$ (in other words, unequal numbers of a's, b's, and c's).

A PDA for $L = \neg A^nB^nC^n$



Are the Context-Free Languages Closed Under Complement?

$\neg A^nB^nC^n$ is context free.

If the CF languages were closed under complement, then

$$\neg\neg A^nB^nC^n = A^nB^nC^n$$

would also be context-free.

But we will prove that it is not.

$L = \{a^nb^mc^p : n, m, p \geq 0 \text{ and } n \neq m \text{ or } m \neq p\}$

$S \rightarrow NC$	/* $n \neq m$, then arbitrary c's
$S \rightarrow QP$	/* arbitrary a's, then $p \neq m$
$N \rightarrow A$	/* more a's than b's
$N \rightarrow B$	/* more b's than a's
$A \rightarrow$ a	
$A \rightarrow$ aA	
$A \rightarrow$ aAb	
$B \rightarrow$ b	
$B \rightarrow B$b	
$B \rightarrow$ aBb	
$C \rightarrow \varepsilon$ | cC	/* add any number of c's
$P \rightarrow B'$	/* more b's than c's
$P \rightarrow C'$	/* more c's than b's
$B' \rightarrow$ b	
$B' \rightarrow$ bB'	
$B' \rightarrow$ bB'c	
$C' \rightarrow$ c | C'c	
$C' \rightarrow C'$c	
$C' \rightarrow$ bC'c	
$Q \rightarrow \varepsilon$ | aQ	/* prefix with any number of a's

PDAs and Context-Free Grammars

Theorem 12.3: The class of languages accepted by PDAs is exactly the class of context-free languages.

Recall: context-free languages are languages that can be defined with context-free grammars.

Restate theorem:

Can describe with context-free grammar

$=$

Can accept by PDA

Going One Way

Theorem 12.1: Each context-free language is accepted by some PDA.

Proof (by construction) (not required for midterm !!)

The idea: Let the stack do the work.

Two approaches:

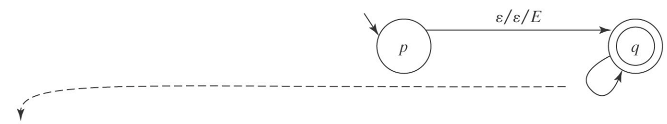
- Top down
- Bottom up

Top Down

The idea: Let the stack keep track of expectations.

Example: Arithmetic expressions

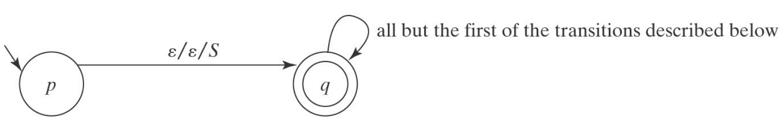
$E \to E + T$
$E \to T$
$T \to T * F$
$T \to F$
$F \to (E)$
$F \to id$



(1) $(q, \varepsilon, E), (q, E+T)$
(2) $(q, \varepsilon, E), (q, T)$
(3) $(q, \varepsilon, T), (q, T*F)$
(4) $(q, \varepsilon, T), (q, F)$
(5) $(q, \varepsilon, F), (q, (E))$
(6) $(q, \varepsilon, F), (q, id)$
(7) $(q, id, id), (q, \varepsilon)$
(8) $(q, (, (\ ), (q, \varepsilon)$
(9) $(q,),)\ ), (q, \varepsilon)$
(10) $(q, +, +), (q, \varepsilon)$
(11) $(q, *, *), (q, \varepsilon)$

A Top-Down Parser

The construction in general:



$M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$, where Δ contains:

- The start-up transition $((p, \varepsilon, \varepsilon), (q, S))$.
- For each rule $X \to s_1 s_2 \ldots s_n$. in R, the transition: $((q, \varepsilon, X), (q, s_1 s_2 \ldots s_n))$.
- For each character $c \in \Sigma$, the transition: $((q, c, c), (q, \varepsilon))$.

Example: $L = \{a^n b^* a^n\}$

(p, ε, ε), (q, S)
(q, a, a), (q, ε)
(q, b, b), (q, ε)

(q, ε, S), (q, ε)
(q, ε, S), (q, B)
(q, ε, S), (q, aSa)
(q, ε, B), (q, ε)
(q, ε, B), (q, bB)

CF grammar:
S → ε
S → B
S → aSa
B → ε
B → bB

input = a a b b a a

Trans	*state*	*unread input*	*stack*
	p	a a b b a a	ε
0	q	a a b b a a	S
3	q	a a b b a a	aSa
6	q	a b b a a	Sa
3	q	a b b a a	aSaa
6	q	b b a a	Saa
2	q	b b a a	Baa
5	q	b b a a	bBaa
7	q	b a a	Baa
5	q	b a a	bBaa
7	q	a a	Baa
4	q	a a	aa
6	q	a	a
6	q	ε	ε

Going The Other Way

Theorem 12.2: If a language is accepted by a pushdown automaton *M*, it is context-free (i.e., it can be described by a context-free grammar).

The proof is by construction - very complicated, not required !!

Nondeterminism, minimality

A PDA *M* is ***deterministic*** iff:

- Δ_M contains no pairs of transitions that compete with each other
- Whenever *M* is in an accepting configuration it has no available moves.

1. Determinism is strictly less powerful: There are context-free languages for which no deterministic PDA exists.

2. It is possible that a PDA may
   - not halt,
   - not ever finish reading its input.

3. There exists no algorithm to minimize a PDA. It is undecidable whether a PDA is minimal.