PUSHDOWN AUTOMATA

Q : states

Σ : input alphabet $\qquad \Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

Γ : stack alphabet $\qquad \Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$

$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}_f(Q \times \Gamma_\varepsilon)$ transition function

$q_0 \in Q$ start state

$F \subseteq Q$ accept states

Here $\mathcal{P}_f$ means finite powerset

start in q_0, stack is empty, looking at the first input symbol

at each step the automaton may

(i) look at input symbol & top of stack and then

(a) change state (b) pop or push the stack (c) move to next symbol

$$a, b \longrightarrow c$$

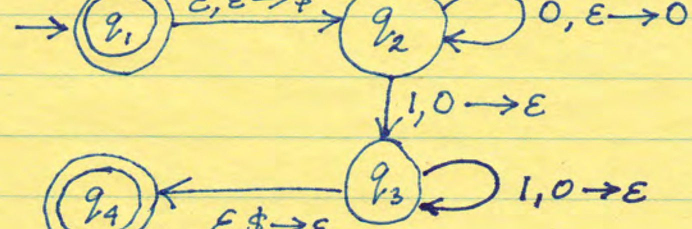
you see a in the input

you see b on top of the stack & replace it with c

a may be ε : don't read input

b may be ε : just push c onto stack

c may be ε : just pop the stack

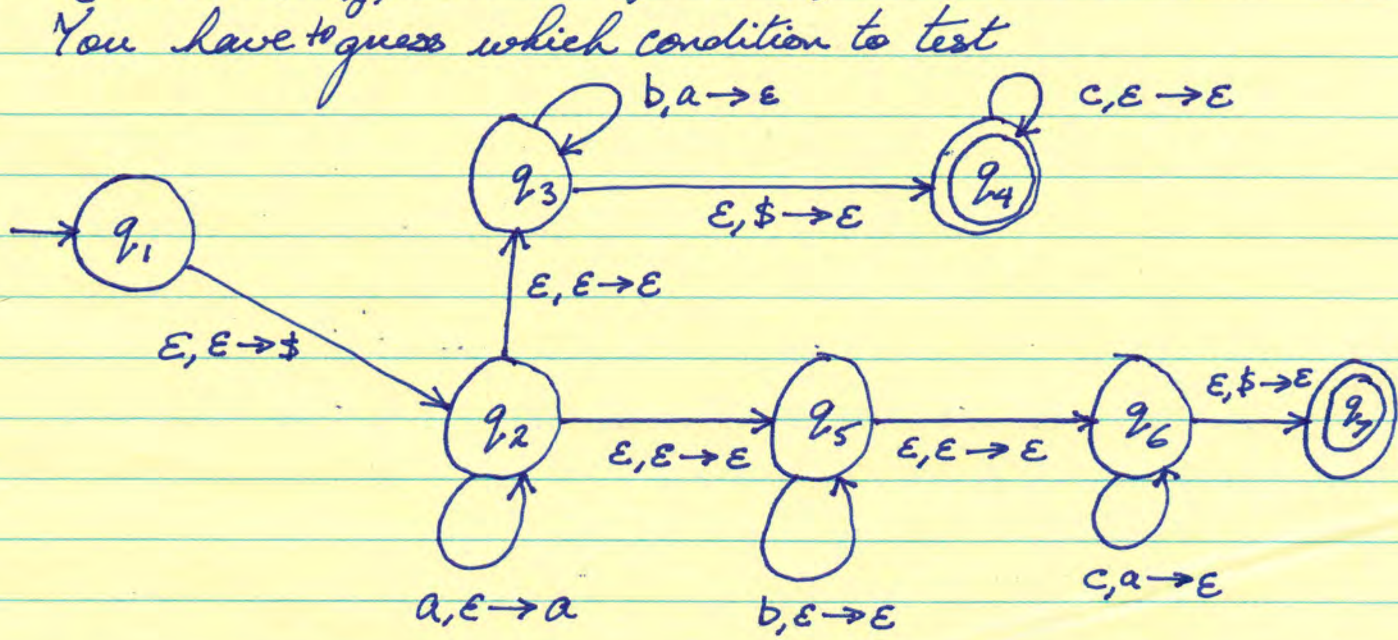


$\{0^n 1^n \mid n \geq 0\}$

What happens if you see an unexpected symbol?
The machine jams, this counts as reject. This happens for example if there are more 1's than 0's. This machine is deterministic.

$L = \{ a^i b^j c^k \mid i,j,k \geq 0 \ \& \ i=j \text{ or } i=k \}$

You have to guess which condition to test



Informal description: push \$ on the stack before you start reading. Then stack all the a's. <u>Guess</u> whether you should match b's or c's. The top branch is for matching b's: this happens in q_3. When all b's have been matched with a's you see the \$ symbol on the stack; jump to q_3 and read any c's. These c's are ignored. The bottom branch corresponds to matching a's & c's. In q_5 b's are just ignored. In q_6 c's & a's are matched.

§ Some important general points:

(α) Acceptance only happens at the end of the input. If the PDA is in an accept state with input still left to read it cannot say "I will stop here and accept"; it has to get to the end of the input in order to accept.

(β) A PDA cannot decide to "jam" when it has a possible move to make.

(v) If there is a state with two or more moves and one of them is $\varepsilon, \varepsilon \to \varepsilon$ it can choose to do this at any point even if one of the other moves is possible.

Let us look back at the example to explore some possibilities. It should be easy to trace an accepted string through the PDA with choices leading to acceptance. For example aabcc will work as follows:

	INPUT	STATE	STACK	
1	aabcc	q_1	ε	
2	aabcc	q_2	\$	
3	abcc	q_2	a\$	
4	bcc	q_2	aa\$	($\varepsilon, \varepsilon \to \varepsilon$ move)
5	bcc	q_5	aa\$	
6	cc	q_5	aa\$	($\varepsilon, \varepsilon \to \varepsilon$ move)
7	cc	q_6	aa\$	
8	c	q_6	a\$	
9	—	q_6	\$	
10	—	q_7	—	ACCEPT

What if it jumped from q_2 to q_5 after step 3? In q_5 it has an $\varepsilon, \varepsilon \to \varepsilon$ move possible but the $b, \varepsilon \to \varepsilon$ move is not possible because the current input letter is a, not b. So it has to go to q_6. Now it is stuck & it rejects. Suppose after line 3 it jumps to q_3 & now it jams. Suppose after line 2 it jumps to q_3. Now input = aabcc, state = q_3, stack = \$. The only move it can do takes it to q_4 where it will jam.

FUNDAMENTAL THEOREM

Every CFL is recognized by a PDA. Every language recognized by a PDA is a CFL.

Idea 1 To show that every CFL is accepted by a PDA we use the stack to keep track of partial derivations. We use nondeterminism to guess which rule to use. Every time we guess a rule we pop the nonterminal on top of the stack & push the RHS of the rule.

Idea 2 For every pair of states we introduce a new nonterminal. We design a grammar to ~~keep~~ generate all the strings that take you from the first state to the second state.

FACT The intersection of a regular language & a CFL is always a CFL.

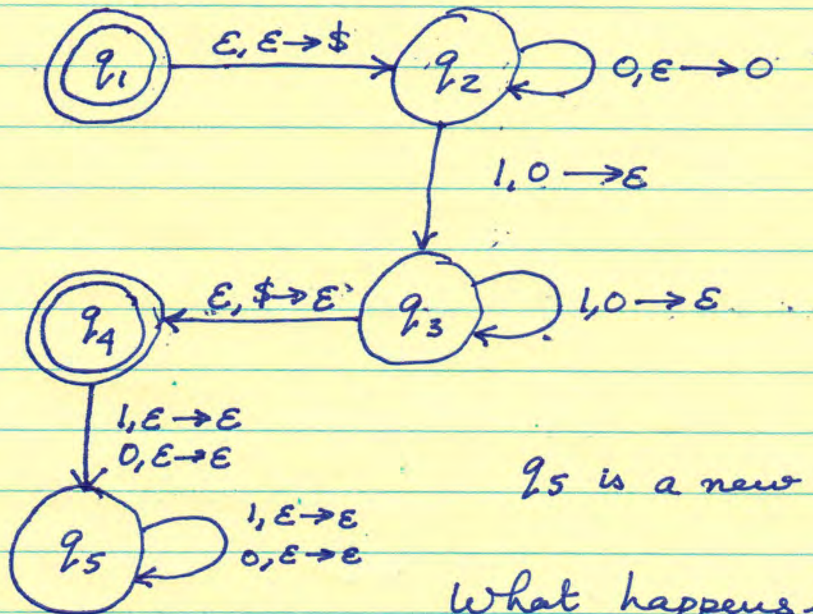
FACT In PDA's nondeterminism cannot always be eliminated.

Deterministic PDA (DPDA): The transition function is

$$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow (Q \times \Gamma_\varepsilon) \cup \phi$$

so we don't have a set of possibilities. Further For every $q \in Q$, $a \in \Sigma$, $x \in \Gamma$ exactly one of $\delta(q,a,x)$, $\delta(q,a,\varepsilon)$, $\delta(q,\varepsilon,x)$ & $\delta(q,\varepsilon,\varepsilon)$ is non-empty. Thus there is never any choice. Every the $\varepsilon, \varepsilon \to \varepsilon$ moves can only happen because there is no enabled action otherwise. The automaton on page 1 needs some dead states to make it a proper DPDA.

Here is the automaton for $\{0^n 1^n \mid n \geq 0\}$ written as a DPDA. There were no transitions coming out of q_4 before



q_5 is a new dead state.

What happens if we try to process 00111? After matching the first 2 0's & the first two 1's we have

INPUT	STATE	STACK
1	q_3	\$

The only action possible is the $\varepsilon, \$ \to \varepsilon$ move to q_4

INPUT	STATE	STACK
1	q_4	—

Now it has only 1 move, to q_5 via $1, \varepsilon \to \varepsilon$

INPUT	STATE	STACK
—	q_5	—

Here the string is rejected.
The PDA cannot stop in q_4 & say "I accept" when there is input still to be read.

A language is a DCFL if it is recognized by a DPDA. Every DCFL has an unambiguous grammar but not every language with an unambiguous grammar is a DCFL. We can define acceptance by empty stack: if the stack is empty at the end of the input we accept. This is an alternate but equivalent notion of acceptance.