Recitation - Week 5
Module 6 + 7

CSE 355: Theory of Computation

Today’s Agenda

1. CFGs
2. Converting CFG into CNF
3. Introduction to PDAs
4. PDA Python Machine Design

Context Free Grammar

A CFG is a 4-tuple (V, Σ, R, S) where,

- V : Set of Variables / Non-Terminals
- Σ : Set of alphabet / Terminals
- R : Set of Rules (V -> (V U Σ U ε)*)
- S : Start Variable

Example CFG

A -> BAB | CD
B -> bB | ε
C -> c
D -> dB

- What are some example strings that can be generated?
- Can ‘cdbbbb’ be derived from this CFG? What is its parse tree?
- Do I have multiple ways to derive ‘cdbbb’?
- Does that say anything about the “ambiguity” of the CFG?

CNF Rules

A CFG is said to be in Chomsky-Normal Form (CNF) if :

1. Only the start symbol generates epsilon or no variable does.
2. A Non-Terminal generates two Non-Terminals (S -> AB)
3. A Non-Terminal generates a terminal (S -> a)

Steps to convert a CFG into CNF

1. Eliminate epsilon rules, if any until it only appears on the start state or does not.
2. Eliminate unit rules by substitution.
3. Group Non-Terminals to transform rules of form (S -> ABA) to (S -> AA1, A1 -> BA)
4. Ensure that the terminal rule is satisfied. If not, fix by adding rules like A -> a & B -> b to convert S -> ab to S -> AB.

Example CFG to CNF

Convert the following CFG into CNF.

S -> AB | C
A -> aA | Aa | ε
B -> bB | BC
C -> cd | ε

Introduction to Push Down Automata

PDA is a class of Finite Automata that recognizes CFGs.

A PDA is like an NFA (with a Stack!). Similar to how we can make transitions on reading an alphabet in NFA, we can make transitions while manipulating the stack.

A PDA transition is of the form **a,b -> c** where,

- a : input to read / alphabet to consume
- b : top stack symbol (what to pop)
- c : push symbol (what to push)

Introduction to Push Down Automata

A PDA is defined as a 6-tuple (Q, Σ, Γ, δ, q0, F)

- Q : Set of States
- Σ : set of input symbols (alphabet)
- Γ : Set of stack symbols
- δ : Transition Function : (Q X (Σ U ε) X (Γ U ε)) -> P(Q X (Γ U ε))
- q0 : Start state
- F : Set of final/accept states

For transition function, remember the form **a, b -> c**

Common transition types

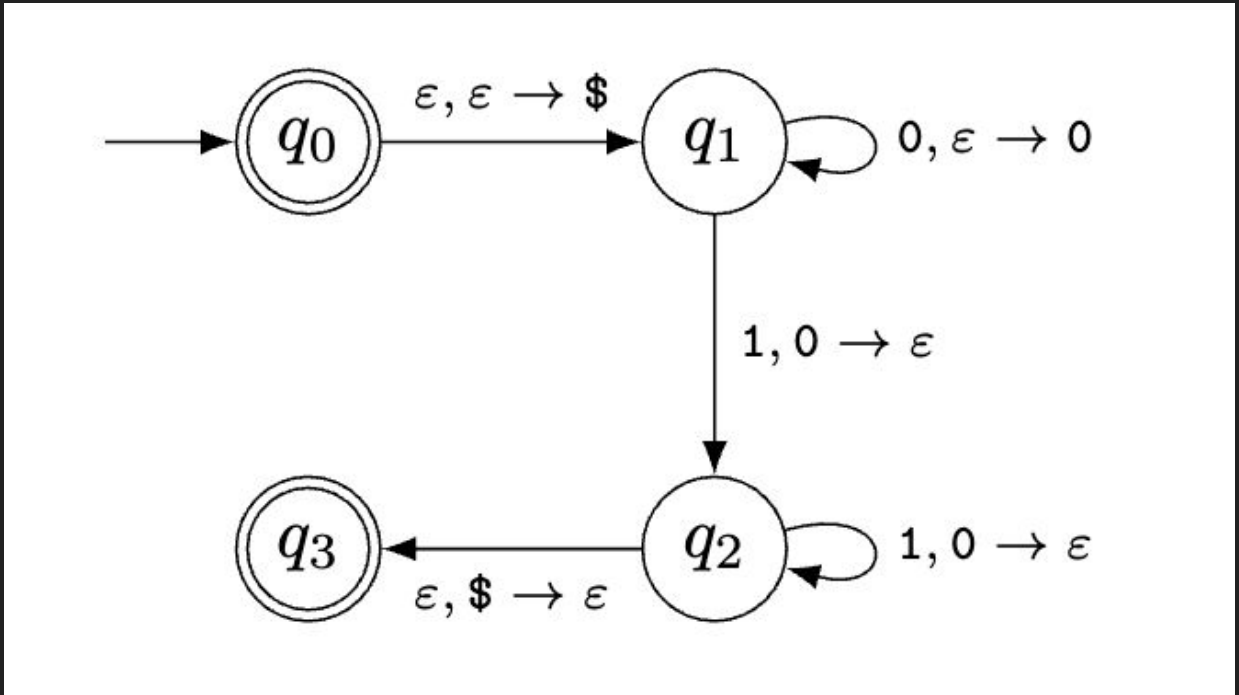
Just like NFAs, we can have epsilon in our transitions. But unlike NFA we have three places where we can fit epsilon (remember a,b -> c).

1. ε,ε -> ε : The “triple epsilon” transition works the same as ε transition in an NFA. It means “read nothing, pop nothing, push nothing”.
2. a,ε -> a : This means “Read ‘a’, push ‘a’”.
3. a,a -> ε : This means “Read ‘a’, pop ‘a’”.
4. ε,a -> b: This means “Read nothing, pop ‘a’ and push ‘b’ in its place”.
5. a,b -> c: This means “Read ‘a’, pop ‘b’ and push ‘c’ in its place”.

Q. What do ε, a -> ε and ε, ε -> a do?

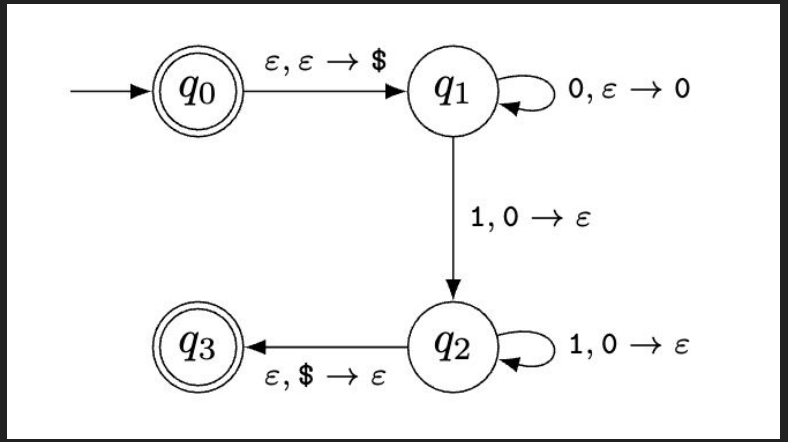
Example PDA

Consider CFG : S -> 0S1 | ε

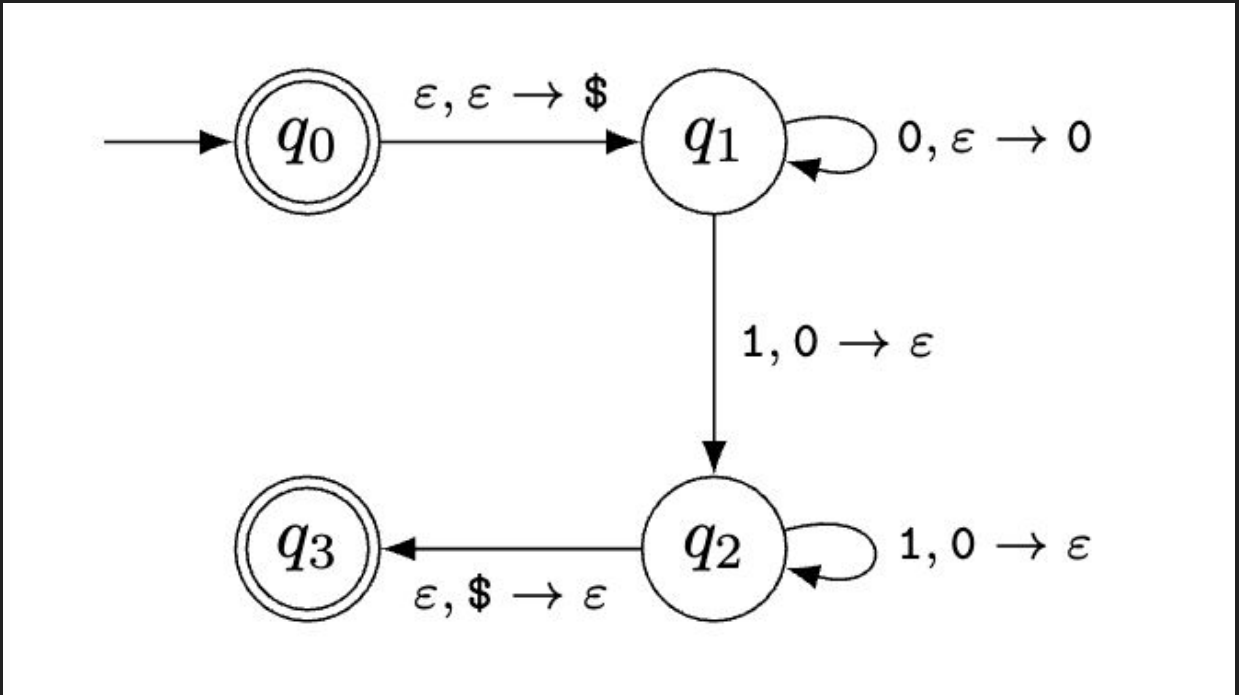


Step-by-step explanation

1. Push ‘$’ to know where the base of the stack is
2. Repeat : Read ‘0’ and push ‘0’
3. Repeat : Read ‘1’ and pop ‘0’
4. If the stack base pointer ($) is found, ACCEPT



Let’s create the PDA in python now!



Any questions, concerns, queries?