Regular Grammars

Definition

- A Regular Grammar is a quadruple

  G= (V,T,P,S), where

  1. V is a finite set of *variables* (*nonterminals*, *syntactic categories*)
  2. T is a finite set of *terminals (alphabet)*
  3. P is a finite set of *productions :* rules of the forms
     1. V -> Λ (λ)
     2. V -> w (β)
     3. V -> V (γ rules)
     4. V -> w V (δ rules) where $w \in T^*$
  4. S, the *start symbol*, is an element of V

Example 1

Non-terminals = [S,B]

Terminals = [a,b]

Start = S

S ->

S -> a S

S -> B

B -> b

B -> b B

Example 2

Non-terminals = [S,C]

Terminals = [a,b,c]

Start = S

S -> a S

S -> b C

C ->

C -> c C

Example 3

Non-terminals = [A,B,C]

Terminals = [a,b]

Start = A

A -> a A

A -> a C

A -> b B

B -> a B

C -> b B

B ->

Derivation

- We say a grammar derives a string if
- Start with any rule whose LHS is the start symbol. Write down the RHS.
- Repeatedly, replace any Non-terminal, X, in the written down term, with rhs, where (X -> rhs) is one of the productions.
- When there are no more Non-terminals, written down term is the derived string.

Example

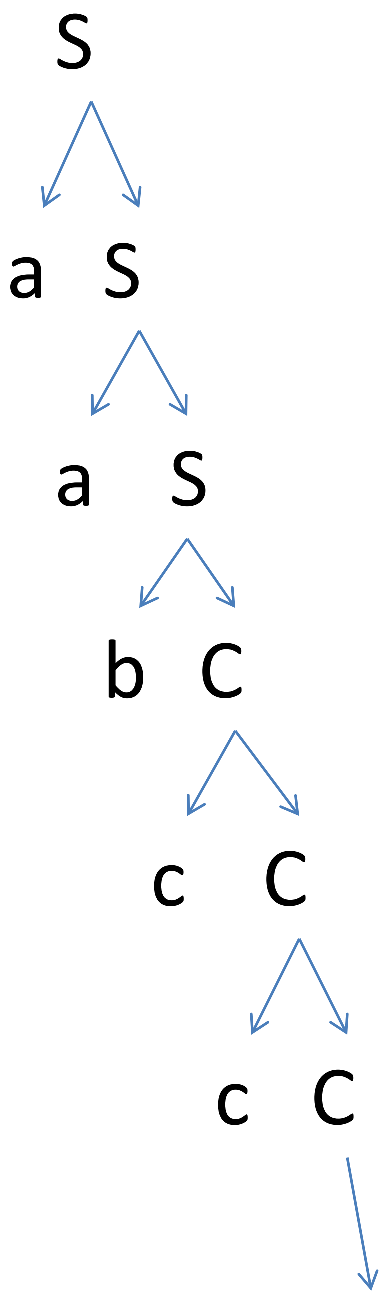
Non-terminals = [S,C]
Terminals = [a,b,c]
Start = S
S -> a S
S -> b C
C ->
C -> c C

- Right-Hand-Side
- a S
- a a S
- a a b C
- a a b c C
- a a b c c C
- a a b c c

- Rule
- S -> a S
- S -> a S
- S -> b C
- C -> c C
- C -> c C
- C ->

Tree Derivation

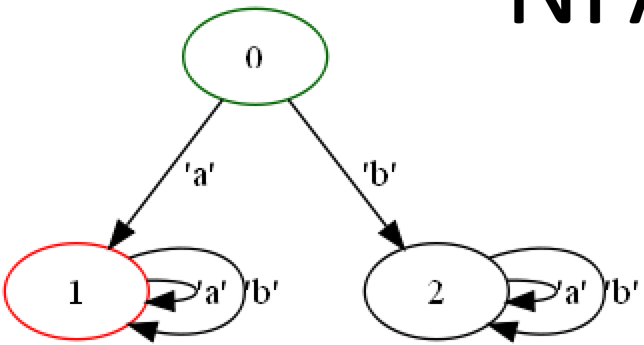
Non-terminals = [S,C]
Terminals = [a,b,c]
Start = S
S -> a S
S -> b C
C ->
C -> c C



Derived string

aabcc

NFA to RegGram



Non-terminals = [S0,S1,S2]

Terminals = [a,b]

Start = S0

S0 -> a S1

S0 -> b S2

S1 -> a S1

S1 -> b S1

S2 -> a S2

S2 -> b S2

S1 ->

For every transition

I –a-> J

Add a production

SI -> a SJ

For every transition

I -Λ-> J

Add a production

SJ ->

For every final state K

Add a production

Sj ->

RegGram to GenNFA

Terminals = [a,b,c,d]

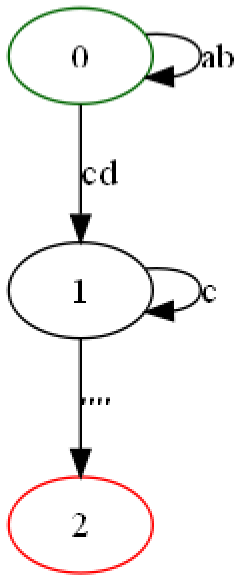
Start = S0

S0 -> a b S0

S0 -> c d S1

S1 ->

S1 -> c S1



The non-terminal become the states, but also invent a new final state F

For each kind of prod

1. V -> Λ (λ)
2. V -> w (β)
3. V -> V (γ rules)
4. V -> w V (δ rules)

Add a transition

1. I -> Λ add I –Λ–> F
2. I -> w add I –w-> F
3. I -> w J add I –w-> J
4. I -> J add I –Λ–> J

Simplify GenNFA