Lecture eight

Topics that must be covered in this lecture:

- ***The regular grammars and regular languages.***

Regular Grammars

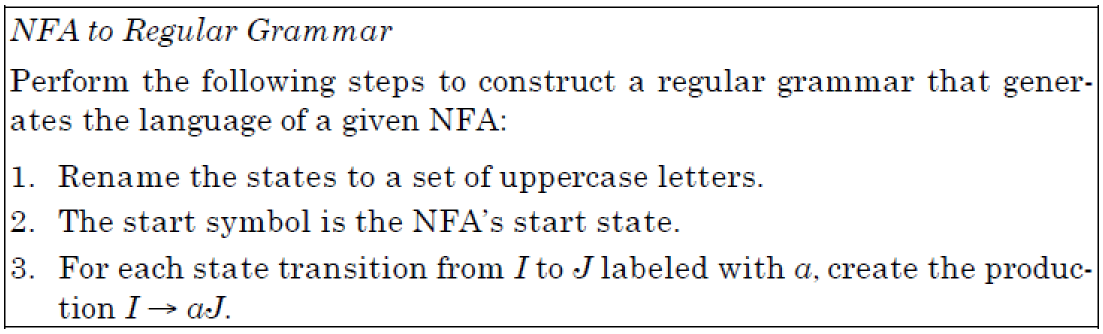
A regular grammar is any right-linear or left-linear grammar

Examples:

G_1	G_2
$S \to abS$	$S \to Aab$
$S \to a$	$A \to Aab \mid B$
	$B \to a$

Convert NFA to a regular grammar:

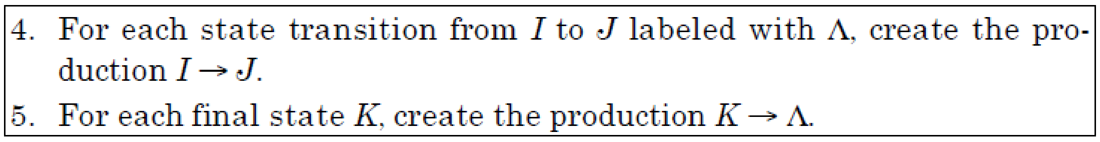
Any regular language has a regular grammar; conversely, any regular grammar generates a regular language. To see this, we'll give two algorithms: one to transform an NFA to a regular grammar and the other to transform a regular grammar to an NFA, where the language accepted by the NFA is identical to the language generated by the regular grammar.



NFA to Regular Grammar

Perform the following steps to construct a regular grammar that generates the language of a given NFA:

1. Rename the states to a set of uppercase letters.
2. The start symbol is the NFA's start state.
3. For each state transition from I to J labeled with a, create the production $I \to aJ$.

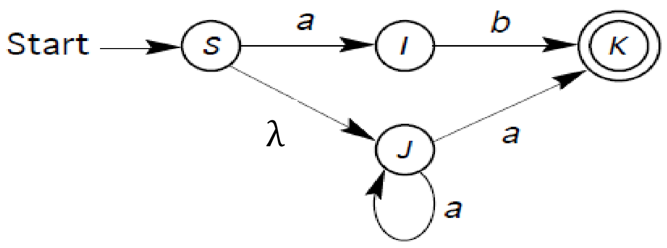


4. For each state transition from I to J labeled with Λ, create the production $I \to J$.
5. For each final state K, create the production $K \to \Lambda$.

It's easy to see that the language of the NFA and the language of the constructed grammar are the same. Just notice that each state transition in the NFA corresponds exactly with a production in the grammar so that the acceptance path in the NFA for some string corresponds to a derivation by the grammar for the same string.

EXAMPLE 1 convert from NFA to Regular Grammar

Let's see how transforms the following NFA into a regular grammar:



The algorithm takes this NFA and constructs the following regular grammar with start symbol *S*:

V={S,I,J,K}

t={a,b} , start symbol=S

production rules:

S→aI | J

I→bK

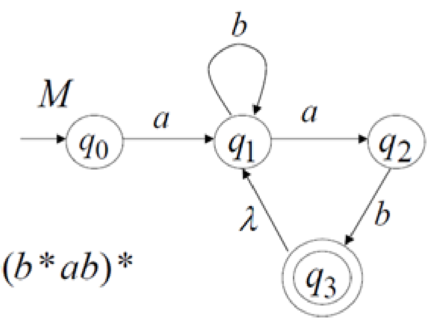
J→aJ | aK

K→λ

For example, to accept the string *aa*, the NFA follows the path *S*, *J*, *J*, *K* with edges labeled *λ*, *a*, *a*, respectively. The grammar derives this string with the following sequence of productions:

$$S\to J\ ,\quad J\to aJ\quad ,\ J\to aK\qquad ,\quad K\to\lambda$$

EXAMPLE 2 :From NFA to Regular Grammar



$$L = ab^*ab(b^*ab)^*$$

$$L = L(M)$$

Solution :

A0=q0

A1=q1

A2=q2

A3=q3

V={A0,A1,A2,A3}, t={a,b}

Start symbol=A0

Production rule:

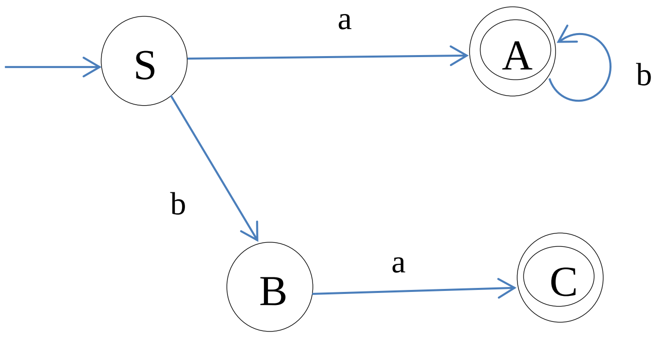
A0→A1

A1→bA1 | aA2

A2→bA3

A3→A1 | λ

EXAMPLE 3 :From NFA to Regular Grammar



Solution:

V={S,A,B,C} t={a,b} start symbol=S

Production rules:

S→aA | bB

A→bA | λ

B→aC

C→ λ

Convert regular grammar to NFA:

Now let's look at an algorithm that does the job of transforming a regular grammar into an NFA.

Regular Grammar to NFA

Perform the following steps to construct an NFA that accepts the language of a given regular grammar:

1. If necessary, transform the grammar so that all productions have the form $A \to x$ or $A \to xB$, where x is either a single letter or Λ.
2. The start state of the NFA is the grammar's start symbol.
3. For each production $I \to aJ$, construct a state transition from I to J labeled with the letter a.
4. For each production $I \to J$, construct a state transition from I to J labeled with Λ.
5. If there are productions of the form $I \to a$ for some letter a, then create a single new state symbol F. For each production $I \to a$, construct a state transition from I to F labeled with a.
6. The final states of the NFA are F together with all I for which there is a production $I \to \Lambda$.

It's easy to see that the language of the NFA is the same as the language of the given regular grammar because the productions used in the derivation of any string correspond exactly with the state transitions on the path of acceptance for the string. Here's an example.

EXAMPLE 1 From Regular Grammar to NFA

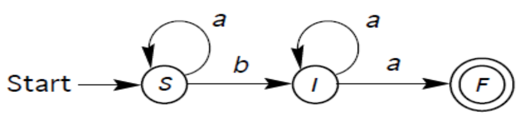
Let's use to transform the following regular grammar into an NFA:

$$S \to aS \mid bI$$
$$I \to a \mid aI.$$

Since there is a production $I \to a$, we need to introduce a new state F, which then gives us the following NFA:

Solution:

Q={S,I}, Σ={a,b} , start state=S , final state={F}



EXAMPLE 2 From Regular Grammar to NFA

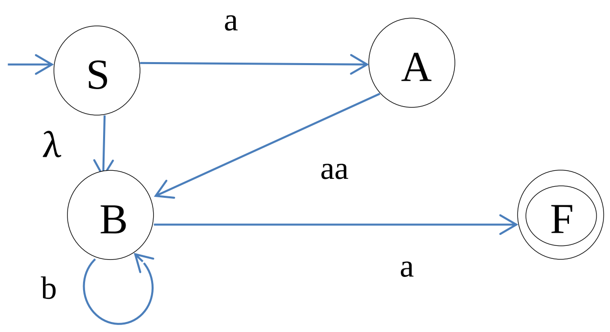
$S \rightarrow aA \mid B$

$A \rightarrow aa\ B$

$B \rightarrow b\ B \mid a$

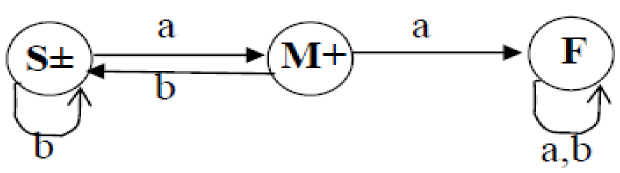
Solution:

Q={S,A,B,F}, Σ={a,b} , start state=S , final state={F}



Example

Consider the following FA with two final states:



So the production rules of our example will be:

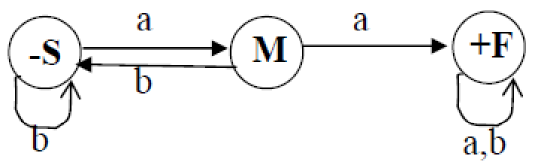
$S \rightarrow aM \mid bS \mid \lambda$

$M \rightarrow aF \mid bS \mid \lambda$

$F \rightarrow aF \mid bF$

Example

Consider the FA below, which accepts the language of all words with a double a:



So the production rules of our example will be:

$S \rightarrow aM \mid bS$
$M \rightarrow aF \mid bS$
$F \rightarrow aF \mid bF \mid \lambda$