- **Right-Linear Grammar**

A grammar is **right-linear**, if all productions have one of the two forms:

$$V \rightarrow T^* V \text{ or}$$

$$V \rightarrow T^*$$

We can have only one variable-symbol on the left-hand side and on the right-hand side, we have at most one variable, and this is at the far right.

- **Left-Linear Grammar**

A grammar is **left-linear**, if all productions have one of the two forms:

$$V \rightarrow V\ T^* \text{ or}$$

$$V \rightarrow T^*$$

We can have only one variable-symbol on the left-hand side and on the right-hand side, we have at most one variable, and this is at the far left.

- **Regular Grammar**

A grammar is **regular**, if it is either right-linear or left-linear.

This means, all productions in the grammar have to be completely left-linear or completely right-linear but not mixed left-linear and right-linear.

- ## Linear Grammar

Grammars, in which each rule is in right-linear or left-linear form, i.e. left-linear and right-linear rules can be mixed, is called **linear**.

Linear grammars are a more general class of grammars than regular grammars.

Example 1 : The grammar with the following productions:

$S \rightarrow a X$

$X \rightarrow S b$

$S \rightarrow \lambda$

is linear but neither right-linear nor left-linear, and thus not a regular grammar.

Which language does this grammar describe? H.W

Example 2 :What languages do the following grammars generate? H.W

$G_1 = (V, T, P_1, S)$ with **V** = {A, B} **T** = {a, b} and Productions $\mathbf{P_1}$:

$S \rightarrow A \mid B$

$A \rightarrow a A \mid a$

$B \rightarrow b B \mid b$

$G_2 = (V, T, P_2, S)$ with **V**, **T** as above and Productions $\mathbf{P_2}$:

$S \rightarrow A$

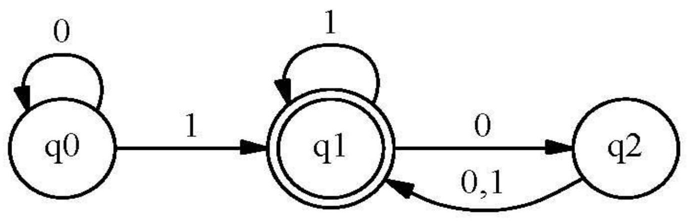
$A \rightarrow a A \mid B \mid \lambda$

$B \rightarrow b B \mid \lambda$

Finite State Automata

3 How to Read State Diagrams

The following figure depicts a finite state automaton called M_1:



- M_1 has three STATES, labeled $q0$, $q1$, $q2$.
- The START STATE is $q0$.
- The FINAL STATE (or ACCEPT STATE) is the one with a double circle, $q1$.
- The arrows going from one state to another are called TRANSITIONS.
- When this automaton receives an input string, it processes each symbol in tha string from left to right, and produces an output. The output is either ACCEP or REJECT. The processing begins in the machine's start state, and after readin each symbol, it moves from one state to another along the transition that has tha symbol as its label. When the machine reads the last symbol, the output is ACCEP if the machine is in a final (accept) state, and REJECT if it is not.

  When we feed the input string `1101` to machine M_1:

  1. start in state $q0$;
  2. read 1, follow transition from $q0$ to $q1$;
  3. read 1, follow transition from $q1$ to $q1$;
  4. read 0, follow transition from $q1$ to $q2$;
  5. read 1, follow transition from $q2$ to $q1$;
  6. *accept* because M_1 is in an accept state $q1$ at the end of the input.

- Question:

  Which of the following strings are accepted by the machine M_1?

  a. `1`
  b. `01`
  c. `101000`
  d. `11`
  e. `0101010101`
  f. `10`
  g. `0101000000`
  h. `100`
  i. `0`
  j. `0100`
  k. `110000`
  l. `111010100000`

4 Deterministic vs. Non-Deterministic FSA

- In an deterministic FSA, all moves are always uniquely determined. That is, at an state q, you can choose any item from the alphabet and it will take you to exactl one state q'.
- In a non-deterministic FSA, not all moves are uniquely determined. That is, a some state q, some alphabet item may not send you to any state at all, or it ma send you to more than one state.

5 Formal Definition of a Deterministic Finite Stat Automaton

- Definition 1.1

  A finite state automaton is a 5-tuple $< Q, \Sigma, \delta, q0, F >$, where:

  1. Q is a finite set of states;
  2. Σ is a finite set called the alphabet;
  3. $\delta : Q \times \Sigma \rightarrow Q$ is the transition function;
  4. $q0 \in Q$ is the start state;
  5. $F \subseteq Q$ is the set of accept states.

- Questions:

  Can finite state machines have more than one accept states?

  Can they have zero number of accept states?

  Given the definition 1.1 of deterministic FSA, must there be exactly one transitio arrow exiting every state for each possible input symbol?

- We can describe M_1 formally by writing $M_1 =< Q, \Sigma, \delta, qo, F >$, where

  1. $Q = \{q0, q1, q2\}$;
  2. $\Sigma = \{0, 1\}$;
  3. δ is defined as

	0	1
q0	q0	q1
q1	q2	q1
q2	q1	q1

  4. $q0$ is the start state;
  5. $F = \{q1\}$.

- Question

  Given the formal description of finite state automaton M_2 below, draw a corre sponding state diagram for M_2.

  $M_2 =< Q, \Sigma, \delta, qo, F >$, where

  1. $Q = \{q0, q1\}$;
  2. $\Sigma = \{0, 1\}$;
  3. δ is defined as

	0	1
q0	q0	q1
q1	q0	q1

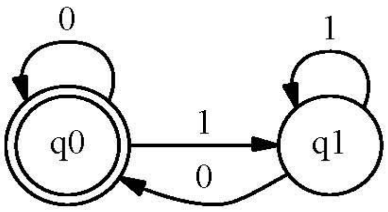
  4. $q0$ is the start state;
  5. $F = \{q1\}$.

  Which of the following strings are accepted by M_2?

  a. 0
  b. 1
  c. 00
  d. 11111
  e. 1000000
  f. 1010011
  g. ϵ

- Question

Consider the state diagram for finite state automaton M_3, and give a formal de scription of M_3.



$M_3 =< Q, \Sigma, \delta, qo, F >$, where

1. $Q =$
2. $\Sigma =$
3. δ is defined as

	0	1
q0		
q1		

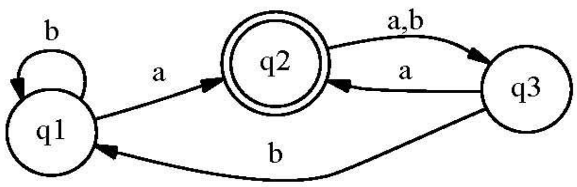
4. The start state is
5. $F =$

Which of the following strings are accepted by M$_3$?

a. 0
b. 1
c. 00
d. 11111
e. 1000000
f. 1010011
g. ϵ

- Question for RECITATION

Consider the state diagram for finite state automaton M_4, and give a formal de scription of M_4.



Which of the following strings does M_4 accept?

a. `aabb`
b. `abaab`
c. `bbbba`
d. `baaba`
e. `aabaa`
f. ϵ
g. `ba`
h. `a`
i. `aaaaa`
j. `abaa`