The grammar of a (natural or formal) language tells us how to combine a finite number of discrete units to form a potentially infinite set of complex strings. Some grammars can be modeled as finite state automata (FSA). The languages whose grammar can be modeled as a FSA are called regular languages. We will study those in this part of the course (topics 3 and 4 in syllabus).

1 String, Alphabet, Language

- **ALPHABET** is any finite set. The members of the alphabet are the SYMBOLS of the alphabet.
  - $\Sigma_1 = \{0, 1\}$
  - $\Sigma_2 = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$
  - $\Gamma = \{0, 1, x, y, z\}$
- A **STRING OVER AN ALPHABET** is a finite sequence of symbols from that alphabet.
  - If $\Sigma_1 = \{0, 1\}$, then 011001 is a string over $\Sigma_1$.
  - If $\Sigma_2 = \{a, b, c, ..., z\}$, then abracadabra is a string over $\Sigma_2$.
  - If $w$ is a string over some alphabet, the **LENGTH** of $w$, written $|w|$, is the number of symbols that it contains.
  - $|011001| = 6$
  - $|abracadabra| = 11$
- The string of length zero is called the **EMPTY STRING** and is written $\epsilon$.
- The reverse of string $w$, written $w^R$, is the string obtained by writing $w$ in the opposite order.
  - $011001^R = 100110$
- String $z$ is a **SUBSTRING** of $w$ if $z$ appears consecutively within $w$.
  - 11 is a substring of 011001.
- Given two strings $x$ and $y$, the **CONCATENATION** of $x$ and $y$ is the string obtained by appending $y$ to the end of $x$.
  - Concatenation of 011001 and abracadabra is 011001abracadabra.
- A **LANGUAGE** is a (possibly infinite) set of strings.
2 An example: the cola machine

- Characteristics of Cola machine
  
  2. The only coins accepted by the machine are quarter (Q), dime (D), and nickel (N).
  3. The machine accepts any combination of these coins, in any order, that add up to 25 cents.
  4. The machine requires exact change.

- This cola machine is a finite state automaton. Before we put any money into it, it will be in a start state. Our job is to add appropriate coins that change the state of the machine until it is in a special final state. The machine will reach the final state (or accept state) when the total amount of money you inserted into the machine adds exactly up to 25 cents. By convention, we will use a double circle to mark such a final state.

- We can get to the final state in one step by inserting a quarter. But there are other ways of getting to the final state by inserting various combinations and permutations of nickels and dimes.

- List of all the possible sequences of coins that the cola machine will accept, where acceptance means reaching the final state and allowing us to get a cola:

- Let us make a transition from a mechanical machine to a linguistic one. Think of the inputs to the machine not as coins but as symbols like Q, D, and N. The set of valid symbols that the machine will accept is its alphabet. The sequence of symbols that the machine will accept are strings. The entire set of words that the machine accepts or recognizes is its language.
Questions:

1. What is the alphabet of the cola machine?
2. What are the strings that the cola machine accepts?
3. What is the language of the cola machine?

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 that string from left to right, and produces an output. The output is either accept or reject. The processing begins in the machine’s start state, and after reading each symbol, it moves from one state to another along the transition that has that symbol as its label. When the machine reads the last symbol, the output is accept 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. 01010101
   f. 10
   g. 010100000
   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 any state $q$, you can choose any item from the alphabet and it will take you to exactly one state $q'$.

• In a non-deterministic FSA, not all moves are uniquely determined. That is, at some state $q$, some alphabet item may not send you to any state at all, or it may send you to more than one state.

5 Formal Definition of a Deterministic Finite State 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. $\Sigma$ 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 transition arrow exiting every state for each possible input symbol?

We can describe $M_1$ formally by writing $M_1 = <Q, \Sigma, \delta, q_0, F>$, where

1. $Q = \{q_0, q_1, q_2\}$;
2. $\Sigma = \{0, 1\}$;
3. $\delta$ is defined as

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q_0$</td>
<td>$q_1$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_2$</td>
<td>$q_1$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_1$</td>
<td>$q_1$</td>
</tr>
</tbody>
</table>

4. $q_0$ is the start state;
5. $F = \{q_1\}$.

Question

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

$M_2 = <Q, \Sigma, \delta, q_0, F>$, where

1. $Q = \{q_0, q_1\}$;
2. $\Sigma = \{0, 1\}$;
3. $\delta$ is defined as

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q_0$</td>
<td>$q_1$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_0$</td>
<td>$q_1$</td>
</tr>
</tbody>
</table>

4. $q_0$ is the start state;
5. $F = \{q_1\}$.

Which of the following strings are accepted by $M_2$?

a. 0
b. 1
c. 00
d. 11111
e. 1000000
f. 1010011
g. $\epsilon$
• Question
Consider the state diagram for finite state automaton $M_3$, and give a formal description of $M_3$.

$M_3 = \langle Q, \Sigma, \delta, q_0, F \rangle$, where
1. $Q =$
2. $\Sigma =$
3. $\delta$ is defined as

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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. 10000000
f. 1010011
g. $\epsilon$

• Question for recitation
Consider the state diagram for finite state automaton $M_4$, and give a formal description of $M_4$. 
Which of the following strings does $M_4$ accept?

a. aabb
b. abaab
c. bbbba
d. baaba
e. aabaa
f. $\epsilon$
g. ba
h. a
i. aaaaa
j. abaa

6 The language of a FSA

Let $M$ denote a finite state automaton and $w$ denote a string of symbols:

- The language of machine $M$, written as $L(M)$, is the set of all strings that machine $M$ accepts.
  
  For example, $L(M_4) = \{w \mid w$ ends with a 1 or $w$ ends with an even number of 0s following the last 1\}.
  
  Let $A$ be $L(M)$. We say that $M$ recognizes $A$ or that $M$ accepts $A$.

- The string $w$ is in the language accepted by $M$, $w \in L(M)$, if and only if $M$ reads $w$ entirely and halts in a final state of $M$.

- A machine may accept several strings, but it always recognizes only one language.

- If the machine accepts no strings, it still recognizes one language, namely, the empty language, written as $\emptyset$.

- A language is regular if and only if there exists a finite state automaton that recognizes it.

  That is, if we claim that a language is a regular language, we must be able to back up our claim by producing a finite state automaton that recognizes the language.

- Question

  Show that the following languages are regular. That is, for each of these languages, give a FSA diagram that accepts all and only the strings in that language. Assume $\Sigma = \{1, 0\}$.

  a. $\{w : w$ contains at least three 1s$\}$
  b. $\{w : w$ begins with 1 and ends with 0$\}$
• Question for recitation
For each of the following languages, give a FSA that recognizes it. Assume \( \Sigma = \{ 1, 0 \} \).
  a. \( \{ w : w \text{ contains exactly three 1s (and any number of 0s)} \} \)
  b. \( \{ w : w \text{ has an odd length} \} \)
  c. \( \{ 1 \} \)
  d. \( \{ w : w \text{ contains the sub-string 111} \} \)
  e. \( \{ w : \text{the length of } w \text{ is a multiple of 3} \} \)

7 The Regular Operations

7.1 Definition of regular operations

• Definition 1.10
  Let \( A \) and \( B \) be languages (with the same alphabet \( \Sigma \)). We define the regular operations \( \text{union} \), \( \text{intersection} \), \( \text{concatenation} \) and \( \text{star} \) as follows.

  - Union: \( A \cup B = \{ x | x \in A \text{ or } x \in B \} \).
  - Intersection: \( A \cap B = \{ x | x \in A \text{ and } x \in B \} \).
  - Concatenation: \( A \circ B = \{ xy | x \in A \text{ and } x \in B \} \).
  - Star: \( A^* = \{ x_1x_2x_3...x_k | k \geq 0 \text{ and each } x_i \in A \} \).

• Examples
  Let the alphabet \( \Sigma \) be the standard 26 letters \( \{ a, b, c, ..., z \} \). If \( A = \{ \text{good, bad} \} \) and \( B = \{ \text{boy, girl} \} \), then
  \( A \cup B = \{ \text{good, bad, girl, boy} \} \).
  \( A \cap B = \emptyset \).
  \( A \circ B = \{ \text{goodboy, goodgirl, badboy, badgirl} \} \).
  \( A^* = \{ \epsilon, \text{good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad, badgoodgood, badbadbadgood, ...} \} \)
  Note that the empty string \( \epsilon \) is always a member of \( A^* \), no matter what \( A \) is!!

• Question
  Let \( C = \{ \text{apple, banana, pear} \} \) and \( D = \{ \text{grape, banana} \} \).
  \( C \cup D = \)
  \( C \cap D = \)
  \( C \circ D = \)
  \( D^* = \)
7.2 Properties of regular operations

- Theorem 1.12
  The class of regular languages is closed under the union operation.
  That is, if $A$ and $B$ are regular languages, then $A \cup B$ is also a regular language.

- Theorem
  The class of regular languages is closed under the intersection operation.
  That is, if $A$ and $B$ are regular languages, then $A \cap B$ is also a regular language.

- Theorem 1.13
  The class of regular languages is closed under the concatenation operation.
  That is, if $A$ and $B$ are regular languages, then $A \circ B$ is also a regular language.

- Theorem
  The class of regular languages is closed under the star operation.
  That is, if $A$ is a regular language, then $A^*$ is also a regular language.

7.3 Proof of Theorem 1.12

Theorem 1.12
The class of regular languages is closed under the union operation.
That is, if $A$ and $B$ are regular languages, then $A \cup B$ is also a regular language.

- Remember that if $A$ is a regular language, then there is a finite state automaton, call it $M_A$, such that $A = L(M_A)$. That is, there must be a finite state automaton that accepts $A$.

  So, to show that $A \cup B$ is a regular language if $A$ and $B$ are regular languages, we have to show that there is a finite state automaton, $M_{A\cup B}$ that accepts $A \cup B$.

  In particular, suppose $M_A$ is the automaton that recognized $A$ and $M_B$ is the automaton that recognized $B$, we want to show how to automatically build a finite state automaton, $M_{A\cup B}$, from $M_A$ and $M_B$ such that $A \cup B$ is accepted by $M_{A\cup B}$.

  This is a proof by construction. We construct a machine $M'$ that simulates $M_A$ and $M_B$ for any input. In a sense, $M'$ would take the input and run $M_A$ and $M_B$ on it in parallel. It would accept the input if and only if either $M_A$ or $M_B$ accepts the input.
• What does it mean for a finite state machine to simulate two other machines in parallel?

At each step, $M'$ would look at an input symbol from the string and enter a new state that is a function of what $M_A$ does on the string and what $M_B$ does on the string. That is, $M'$ needs to remember a pair of states.

This suggests that we should construct $M'$’s states from the Cartesian product of $M_A$’s states and $M_B$’s states. The transitions of $M'$ go from pair to pair, in a way updating the current state for both $M_A$ and $M_B$.

• Here is how to construct such a machine.

Let $M_A$ recognize $A$, where $M_A = < Q_A, \Sigma, \delta_A, q_0, F_A >$, and $M_B$ recognize $B$, where $M_B = < Q_B, \Sigma, \delta_B, q_2, F_B >$.

Construct $M'$ to recognize $A \cup B$, where $M' = < Q', \Sigma, \delta', q_0, F' >$.

1. $Q' = \{ < q_i, q_j > \mid q_i \in Q_A \text{ and } q_j \in Q_B \}$
   This set is the Cartesian product of $Q_A$ and $Q_B$, and is written $Q_A \times Q_B$. It is the set of all pairs of states, the first from $Q_A$ and the second from $Q_B$.

2. $\Sigma$, the alphabet, is the same as in $M_A$ and $M_B$.

3. $\delta'$, the transition function, is defined as follows.
   For each $< q_i, q_j > \in Q'$ and each $a \in \Sigma$,
   $\delta'( < q_i, q_j >, a ) = < \delta_A(q_i, a), \delta_B(q_j, a) >$
   That is, $\delta'$ gets a state of $M'$ (which is a pair of states from $M_A$ and $M_B$), together with an input symbol, and returns $M'$’s next state, which is the pair of states returned by $\delta_A$ on $a$ and $\delta_B$ on $a$.

4. $q_0$ is the pair $< q_0, q_2 >$.

5. $F' = \{ < q_i, q_j > \mid q_i \in F_A \text{ or } q_j \in F_B \}$.
   That is, $F'$ is the set of pairs in which either member is a final state of $M_A$ or $M_B$.

If $M_A$ and $M_B$ are finite state automata, then so is $M'$ since it is built from the Cartesian product of the states in $M_A$ and $M_B$, which must be finite.
Example

Let $M_A$ be defined as follows:

1. $Q = \{q_0, q_1, q_2\}$;
2. $\Sigma = \{0, 1\}$;
3. $\delta$ is defined as

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q_0$</td>
<td>$q_1$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_2$</td>
<td>$q_1$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_1$</td>
<td>$q_1$</td>
</tr>
</tbody>
</table>

4. The start state is $q_0$;
5. $F = \{q_1\}$.

And let $M_B$ be defined as follows:

1. $Q = \{q_3, q_4\}$;
2. $\Sigma = \{0, 1\}$;
3. $\delta$ is defined as

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_3$</td>
<td>$q_4$</td>
<td>$q_3$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>$q_4$</td>
<td>$q_3$</td>
</tr>
</tbody>
</table>

4. The start state is $q_3$;
5. $F = \{q_3\}$.
Construct $M'$ that accepts $L(M_A) \cup L(M_B)$.

1. $Q = \{ <q_0,q_3> >, <q_0,q_4>, <q_1,q_3>, <q_1,q_4>, <q_2,q_3>, <q_2,q_4> \}$

2. $\Sigma = \{0,1\}$

3. $\delta$ is

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;q_0,q_3&gt;$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;q_0,q_4&gt;$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;q_1,q_3&gt;$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;q_1,q_4&gt;$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;q_2,q_3&gt;$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;q_2,q_4&gt;$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. The start state is $<q_0,q_3>$

5. $F = \{ <q_0,q_3>, <q_1,q_3>, <q_2,q_3>, <q_2,q_4> \}$
7.4 Proof of closure under intersection

The class of regular languages is closed under the intersection operation. That is, if $A$ and $B$ are regular languages, then $A \cap B$ is also a regular language.

- Let $M_A$ recognize $A$, where $M_A = \langle Q_A, \Sigma, \delta_A, q_1, F_A \rangle$, and
  $M_B$ recognize $B$, where $M_B = \langle Q_B, \Sigma, \delta_B, q_2, F_B \rangle$.

- We construct $M' = \langle Q', \Sigma, \delta', q_0, F' \rangle$ that accepts only strings in the intersection of $A$ and $B$ according to the following rules:

  1. $Q' = Q_A \times Q_B = \{ <q_i, q_j> \mid q_i \in Q_A \text{ and } q_j \in Q_B \}$
  2. $\Sigma$ is the alphabet;
  3. $\delta'(q_i, q_j, a) = \langle \delta_A(q_i, a), \delta_B(q_j, a) \rangle$;
  4. The start state $q_0$ is the pair $(q_1, q_2)$;
  5. $F' = \{ <q_i, q_j> \mid q_i \in F_A \text{ and } q_j \in F_B \}$