A language is **regular** over $\Sigma$ if it can be built from $\emptyset$, { $\Lambda$ }, and { $a$ } for every $a \in \Sigma$, using operators union ( $\cup$ ), concatenation ( . ) and Kleene star ( * ).

Example:  

\[
(\{a\} \cup \{b\})^* \{a\}\{a\} (\{a\} \cup \{b\})^* \cup \\
(\{a\} \cup \{b\})^* \{b\}\{b\} (\{a\} \cup \{b\})^*
\]

<table>
<thead>
<tr>
<th>String</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>aabb 4B</td>
<td>YES</td>
</tr>
<tr>
<td>baab YES</td>
<td></td>
</tr>
<tr>
<td>baba not there</td>
<td></td>
</tr>
</tbody>
</table>
A language is **regular** over $\Sigma$ if it can be built from $\emptyset$, $\{ \Lambda \}$, and $\{ a \}$ for every $a \in \Sigma$, using operators union ($\cup$), concatenation ($.$) and Kleene star ($^*$).

Example:

\[
( \{ a \} \cup \{ b \})^* \{ a \} \{ a \} ( \{ a \} \cup \{ b \})^* \cup
( \{ a \} \cup \{ b \})^* \{ b \} \{ b \} ( \{ a \} \cup \{ b \})^* 
\]

Simplify as:

\[
(a+b)^* aa (a+b)^* +
(c+a)^* bb (a+b)^* 
\]
Regular Languages

A language is **regular** over $\Sigma$ if it can be built from $\emptyset$, $\{ \Lambda \}$, and $\{ a \}$ for every $a \in \Sigma$, using operators union ($\cup$), concatenation ($\cdot$) and Kleene star ($^*$).

Recursive definition of **regular languages over** $\Sigma$:

1) $\emptyset, \{ \Lambda \}, \{ a \}$ for every $a \in \Sigma$ are regular.
2) if $L_1, L_2$ are regular then $L_1 \cup L_2$, $L_1 \cdot L_2$, $L_1^*$ are all regular.
3) nothing else.
Recursive definition of regular expressions over $\Sigma$:

1) $\emptyset$, $\{\Lambda\}$, and $\{a\}$ for every $a \in \Sigma$ are regular languages represented by regular expressions $\emptyset$, $\Lambda$, $a$, respectively.

2) If $L_1$, $L_2$ are regular languages represented by regular expressions $r_1$ and $r_2$, then $L_1 \cup L_2$, $L_1L_2$, and $L_1^*$ are regular languages represented by regular expressions $(r_1 + r_2)$, $(r_1 \cdot r_2)$, $(r_1^*)$, respectively.

3) No other expressions are regular expressions.
Regular Expressions

Comments:
- For convenience, we will also use $r^+$ and $r^i$ for $i \in \mathbb{N}$.
- The priority of operators in decreasing order: Kleene's star, concatenation, union.

Parenthesize:
\[
\left( a + (b^*c) \right) \quad (b^*) \quad \text{then} \quad ((b^*)c) \quad \text{then} \quad (a + ((b^*)c))
\]

\[
\left( a + (b(c^*)) \right)
\]
Regular Expressions

Simplify the following regular expressions:

- $a^*(a+\Lambda) = a^*$
- $a^*a^* = a^*$
- $(a^*b^*)^* = (a+b)^*$
- $(a+b)^*ba(a+b)^* + a^*b^* = (a+b)^*$

[Section 3.1]
Give a regular expression for each of the following:

- the language of all strings over \{a,b\} of even length,
  \[(aa + ab + ba + bb)^* = ((a+b)(a+b))^*\]
- the language of all strings over \{a,b,c\} in which all a’s precede all b’s,
  \[(c+a)^* (b+c)^* = (c^*a^+)^*(a^+c^*)^*\]
- the language of all strings over \{a,b\} with length greater than 3,
  \[((a+b)^+)^4 = (a+b)^4 \cdot (a+b)^*\]
- the language of all odd-length strings over \{a,b\} containing the string bb,
  \[((a+b)^2)^* bb (a+b)^* (a+b) + (a+b)(a+b)^2)^* bb (a+b)^2)^*\]
- the language of all strings over \{a,b\} that do not contain the string aaa.

Is the language \{ a^k b^k \mid k \in \mathbb{N} \} regular? Not regular, we’ll see why.
Finite Automata

Our simplest model of computation:
- has a tape with input, read once from left to right
- has only a single memory register with the state
Finite Automata

Example: the language of all strings over \{a, b\} with an even number of a's.

Diagram:

- States: even a, odd a
- Edges:
  - a between even a and odd a
  - b between odd a and even a
- Initial state: even a
- Accepting state: odd a

Transition diagram:
- States ↔ bubbles
- Exactly one arrow per symbol of the alphabet out of each state
- Initial arrow
- Double bubble = accepting
More examples:
- Language of strings over \{a,b\} with at least 3 a’s.
- Language of strings over \{a,b\} containing substring aaba.
- Language of strings over \{a,b\} not containing substring aaba.
- Language of strings over \{a,b\} with even number of a’s and odd number of b’s.
A (deterministic) finite automaton (FA, in some books also abbreviated as DFA) is a 5-tuple \((Q, \Sigma, q_0, A, \delta)\) where

- \(Q\) is a finite set of states
- \(\Sigma\) is a finite alphabet (input symbols)
- \(q_0 \in Q\) is the initial state
- \(A \subseteq Q\) is a set of accepting states
- \(\delta : Q \times \Sigma \rightarrow Q\) is the transition function
Finite Automata

Comment: We saw that we can represent a finite automaton by a **transition diagram**.

Draw the transition diagram for $$(\{q_a, q_b\}, \{a, b\}, q_a, \{q_b\}, \delta)$$ where $\delta$ is given by the following table:

- $(q_a, a) \rightarrow q_a$
- $(q_a, b) \rightarrow q_b$
- $(q_b, a) \rightarrow q_a$
- $(q_b, b) \rightarrow q_b$

Which strings does this FA accept?
Finite Automata

Defining the computation of an FA $M = (Q, \Sigma, q_0, A, \delta)$.

**Extended transition function** $\delta^* : Q \times \Sigma^* \rightarrow Q$:

1) For every $q \in Q$, let $\delta^*(q, \Lambda) = q$

2) For every $q \in Q, y \in \Sigma^*$, and $\sigma \in \Sigma$, let $\delta^*(q, y\sigma) = \delta(\delta^*(q, y), \sigma)$

We say that a string $x \in \Sigma^*$ is **accepted by** $M$ if $\delta^*(q_0, x) \in A$.

A string which is not accepted by $M$ is **rejected by** $M$.

The **language accepted by** $M$, denoted by $L(M)$, is the set of all strings accepted by $M$. 

[Sections 3.2, 3.3]
**Finite Automata**

**Kleene's Thm:** A language $L$ over $\Sigma$ is regular iff there exists a finite automaton that accepts $L$.

Recall $L = \{ a^k b^k | k \geq 0 \}$. Is it regular?

Suppose there is an FA $M = (Q, \Sigma, q_0, A, \delta)$ s.t. $L(M) = L$.

Consider $a^i$ - infinitely many but $|Q|$ is finite.

Thus, $\exists i < j$ s.t. $\delta^*(q_0, a^i) = \delta^*(q_0, a^j)$.

What happens if the input $a^i b^i$? Has to accept, i.e.

$\delta^*(q_0, a^i b^i) \in A$.

Let's look at $a^i b^i$ - not supposed to accept.

But $\delta^*(q_0, a^i b^i) = \delta^*(q_0, a^j b^i) \in A$
Let \( M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1) \) and \( M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2) \) be two FA's.

**Union:**

We know that \( L(M_1) \cup L(M_2) \) is regular. Construct an FA \( M \) such that \( L(M) = L(M_1) \cup L(M_2) \).

\[
M = (Q, \Sigma, q_0, A, \delta)
\]

where

\[
Q = Q_1 \times Q_2
\]

\[
q_0 = (q_1, q_2)
\]

\[
A = A_1 \times A_2 \cup A_1 \times (Q_2 - A_2) \cup (Q_1 - A_1) \times A_2 = A_1 \times Q_2 \cup Q_1 \times A_2
\]

\[
\delta((p, q), \sigma) = (\delta_1(p, \sigma), \delta_2(q, \sigma)) \quad \forall p \in Q_1, \forall q \in Q_2, \forall \sigma \in \Sigma
\]
Let $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$ be two FA's.

**Intersection:**

Is $L(M_1) \cap L(M_2)$ is regular? Yes but not from reg. expr.

Construct an FA $M$ such that $L(M) = L(M_1) \cap L(M_2)$.

\[ M \text{ same as before except } \]
\[ A = A_1 \times A_2 \]
Let $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ be an FA.

**Complement:**

Is $L(M_1)'$ regular?

Construct an FA $M$ such that $L(M) = L(M_1)'$.

$$M = (Q_1, \Sigma, q_1, Q_1 - A_1, \delta_1)$$
Let $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$ be two FA's.

**Difference:**

Construct an FA $M$ such that $L(M) = L(M_1) - L(M_2)$.

lazy approach: $L(M_1) - L(M_2) = L(M_1) \cap L(M_2)'$

other approach: $M$: same as for $\cup$ or $\cap$

except

$A = A_1 \times (Q_2 - A_2)$
Closure Properties

We just saw that the class of regular languages is closed under union, intersection, complement, and difference.

Let \( L = \{ x \in \{a,b\}^* \mid x \text{ has the same number of } a\text{'s and } b\text{'s } \} \).

Is \( L \) regular? NO.

Recall \( L_0 = \{ a^k b^k \mid k \geq 0 \} \) is not regular.

Outline of the proof:

Suppose \( L \) is regular.

Is it possible to modify \( L \) (using operations w. regular lang.) into \( L_0 \)?

\[ L \cap a^* b^* = L_0 \]

\( a^* b^* \) regular \( \Rightarrow \) assumed reg.

\( L \) regular \( \Rightarrow \) but \( L_0 \) is not regular.

\( \therefore \) thus, \( L \) not reg.