Set operations provide a convenient way of specifying (certain) formal languages.

Let $\Sigma$ be an alphabet not containing the symbols $(, )$, $\emptyset$, $+$, and $\ast$. Regular expressions are strings built from these special symbols and the symbols in $\Sigma$.

Formally, the set of all regular expression over $\Sigma$ is defined recursively:

1. The string $\emptyset$ is a regular expression.
2. If $a \in \Sigma$, then the string $a$ is a regular expression.
3. If $\alpha$ and $\beta$ are regular expressions, so are
   
   (a) $(\alpha \beta)$,
   
   (b) $(\alpha + \beta)$, and
   
   (c) $\alpha^*$. 
4. Only strings constructed according to these rules are regular expressions.

For example, $\emptyset$, $a$, $b$, $(ab)$, $(a + b)$, $b^*$, and $((a + b)^*(ab))$ are all regular expressions over $\{a, b\}.$
**Language Operations**

- Regular expressions specify languages, an interpretation that requires the following operations on languages.

- **Union**
  
  If $L_1$ and $L_2$ are languages, then $L_1 \cup L_2$ denotes the (set) union of the two languages.

- **Concatenation**
  
  If $L_1$ and $L_2$ are languages, then
  
  $$L_1L_2 = \{xy : x \in L_1 \text{ and } y \in L_2\}.$$ 

- **Kleene Star**
  
  If $L$ is a language, then
  
  $$L^* = \{x_1 \cdots x_k : k \geq 0 \text{ and } x_1, \ldots, x_k \in L\}.$$
Every regular expression defines a language, as specified by the following (recursive) rules:

1. $L(\emptyset) = \emptyset$.
2. If $a \in \Sigma$, then $L(a) = \{a\}$.
3. If $\alpha$ and $\beta$ are regular expressions, then
   (a) $L((\alpha\beta)) = L(\alpha)L(\beta)$,
   (b) $L((\alpha + \beta)) = L(\alpha) \cup L(\beta)$, and
   (c) $L(\alpha^*) = L(\alpha)^*$.

For example,

\[
\begin{align*}
L(a) &= \{a\} \\
L(b) &= \{b\} \\
L((ab)) &= \{ab\} \\
L((a+b)) &= \{a, b\} \\
L((b^*)^*) &= \{\epsilon, b, bb, bbb, \ldots\} \\
L(((a+b)^*(ab))) &= \{wab : w \in \{a, b\}^*\}
\end{align*}
\]
Regular Languages

- A language $L$ is called regular if $L = L(\alpha)$ for some regular expression $\alpha$.

- **Examples**
  - The set $L_1 = \{wa : w \in \{a, b\}^*\}$ is regular:
    \[ L_1 = L((((a + b)^*a)) \]
  - The set $L_2$ of all strings over $\{a, b, c\}$ that do not contain the substring $ac$ is regular:
    \[ L_2 = L((c^*(a + (bc^*))^*)) \]
  - The set $L_3$ of all binary strings that do not contain the substring 111 is regular:
    \[ L_3 = L(0^* + (((0^*(1+(11)))(00^*)(1+(11)))^*)0^*)) \]
Closure Properties

• The class of regular languages is evidently closed under union, concatenation, and Kleene star.

• **Theorem**

  The class of regular languages over the alphabet $\Sigma$ is the closure of the set

  $$\{\{\sigma\} : \sigma \in \Sigma\} \cup \{\emptyset\}$$

  with respect to union, concatenation, and Kleene star.

• It is also not difficult to prove that any *finite* set of strings is regular.
Finite Automata

• Regular expressions are *language generators*, i.e., a formalism for *generating* elements of a formal language.

• Finite automata are the corresponding *language recognizers*.

• **Theorem**

  A language is generated by a regular expression if, and only if, it is recognized by a finite automaton.

• **Proof sketch.**

  – It can easily be seen that every regular language is accepted by a finite automaton: (i) the empty set and all singletons \( \{ \sigma \} \) are accepted by finite automata and (ii) the class of languages accepted by finite automata is closed under union, concatenation, and Kleene star.

  – Suppose a language \( L \) is accepted by a finite automaton \( M = (Q, \Sigma, \Delta, s, T') \). Let us assume that the states are \( q_1, \ldots, q_n \).

  – For all integers \( i \) and \( j \) with \( 1 \leq i, j \leq n \), and all integers \( k \) with \( 0 \leq k \leq n \) we denote by \( R(i, j, k) \) the set of all strings that drive \( M \) from state \( q_i \) to state \( q_j \) without passing through any *intermediate* state numbered \( k + 1 \) or higher.
The sets $R(i,j,k)$ can be defined recursively:

- $R(i,i,0) = \{ \epsilon \} \cup \{ a \in \Sigma : (q_i, a, q_i) \in \Delta \}$;
- $R(i,j,0) = \{ a \in \Sigma \cup \{ \epsilon \} : (q_i, a, q_j) \in \Delta \}$ if $i \neq j$;
- $R(i,j,k) = R(i,j,k-1) \cup R(i,k,k-1)R(k,k,k-1) \ast R(k,j,k-1)$ if $k > 0$.

- Evidently, all sets $R(i,j,k)$ are regular.

- Furthermore,

$$L(M) = \bigcup \{ R(i,j,n) : s = q_i \text{ and } q_j \in T \},$$

which shows that $L = L(M)$ is a regular set.

- Regular expressions for the sets $R(i,j,k)$ can be computed via so-called generalized finite automata, which are represented by state diagrams where the edges are labeled by regular expressions.
Pumping Lemma

- Regular expressions that do not contain the symbol \( * \) define finite languages.

- Regular expressions that define infinite languages must contain the symbol \( * \) (though some expressions with \( * \) define a finite language).

- The interpretation of \( * \) implies a certain repetitive structure in corresponding languages.

- In finite automata this repetition manifests itself in cycles in the transition diagram.

**Theorem** [Pumping Lemma]

If \( L \) is a regular language then there is an integer \( n_L \geq 1 \) (the “pumping length”) such that any string \( w \in L \) with \( |w| \geq n_L \) can be written as \( w = xyz \), where

1. \( y \neq \epsilon \),
2. \( |xy| \leq n_L \), and
3. \( xy^iz \in L \) for all \( i \) with \( i \geq 0 \).
• **Proof sketch.**

  – If $L$ is regular there is a *deterministic* finite automaton $M$ such that $L = L(M)$.

  – Let $n_L$, or simply $p$, be the number of states of $M$ and $w \in L$ be a string of length $p$ at least.

  – There is a sequence of transitions,
    \[
    (s_0, a_1 a_2 \ldots a_p) \vdash_M (s_1, a_2 \ldots a_p) \vdash_M \cdots \vdash_M (s_p, \epsilon)
    \]
    where $a_1 a_2 \ldots a_p$ is a prefix of $w$ and $s_0$ is the start state of $M$.

  – Since $M$ contains only $p$ states, there are indices $i$ and $j$ such that $s_i = s_j$.

  – Let $x$ be the string $a_1 \ldots a_i$, $y$ be the string $w a_{i+1} \ldots a_j$, and $z$ be a suffix of $w$ such that $w = xyz$. Then
    1. $y \neq \epsilon$,
    2. $|xy| \leq n_L$, and
    3. $M$ accepts all strings $xy^kz$, where $k \geq 0$.

  – This completes the proof.
Nonregular Languages

• The Pumping Lemma can be used to prove that a variety of languages are nonregular.

• Examples
  – The set \( \{a^i b^i : i \geq 0\} \) is not regular.
  – The set of all strings over \( \{a, b\}^* \) that have an equal number of \( a \)'s and \( b \)'s is not regular.
  – The set of all palindromes (over a given alphabet) is not regular.