1.1 Alphabets, Words, and Languages

A finite nonempty set of symbols is called an alphabet, typically denoted by $\Sigma$.

A finite sequence of symbols from the alphabet is called a word or string over the alphabet.

Note: there is a special word, denoted by $\epsilon$ or $e$ called the empty word.

It is a word over any alphabet.

Any collection of words is called a language.

Note: there is a special language consisting of no words, the empty language $\emptyset$.

Note: the empty language is not the same as the language consisting of the single string $\{\epsilon\}$.

We denote the set of all possible words over an alphabet by $\Sigma^*$, the so-called star closure or universal language over $\Sigma$.

Examples.

(1) All strings that contain a given substring, say, $aaba$. 

(2) All strings that begin and end with the string \textit{abba}.

(3) The set of all balanced parentheses: for example, (((()())()())())

(4) The collection of balanced parentheses and brackets: for example, ([])([])([])

(5) All strings of the form $a^n b^m c^{n+m}$ — this example shows how addition can be treated as a language.

(6) All strings of the form $a^n b^m c^{n*m}$ — this example shows how multiplication can be treated as a language problem.

\section*{1.2 Operations on Strings}

If \( w \) is a word over an alphabet \( \Sigma \), we let \(|w|\), denote its length, which is the number of alphabet symbols in \( w \).

\underline{Note:} the empty string \( \epsilon \) has length 0.

There is a natural binary operation on strings, called \textit{concatenation}.

In particular, if \( w \) and \( z \) are two strings, we let \( wz \) be the word formed by appending the word \( z \) after the word \( w \).

Sometimes we write this operation as \( w \cdot z \).

\underline{Note:} \(|w \cdot z| = |w| + |z|\).
The exponentiation of a word \( w \) is given as \( w^0 = \varepsilon \), while \( w^n = w \cdot w^{n-1} \), if \( n > 0 \).

The reversal or transpose of a word \( w \) is just the mirror image of \( w \).

Formally, we define it as follows:

\[
    w^R = \begin{cases} 
        w, & \text{if } w = \varepsilon, \\
        y^R a, & \text{if } w = ay, \text{ for some } a \in \Sigma \text{ and } y \in \Sigma^* 
    \end{cases}
\]

1.3 Operations on Languages

Let \( A \) and \( B \) be two languages. Then we define their concatenation, written as \( A \cdot B \) or \( A \circ B \), as

\[
    A \cdot B = \{ w \cdot z : w \in A \text{ and } z \in B \}.
\]

The exponentiation of \( A \) is simply given as \( A^0 = \{ \varepsilon \} \),

while \( A^n = A \cdot A^{n-1} \), for \( n > 0 \).

We define the Kleene closure or star closure \( A^* \) of \( A \) to be the infinite union

\[
    \bigcup_{n=0}^{\infty} A^n,
\]
while the positive or plus closure $A^+$ of $A$ as $\bigcup_{n=1}^{\infty} A^n$.

Note: if $A = \emptyset$, then $A^* = \{\epsilon\}$.

The reversal $A^R$ of a language $A$ is given as: $A^R = \{w^R : w \in A\}$.

We have a series of set identities about languages:

1. $A \cup B = B \cup A; \quad A \cap B = B \cap A$.
2. $A \cup \emptyset = \emptyset \cup A = A; \quad A \cap \emptyset = \emptyset$.
3. $A \cup A = A; \quad A \cap A = A$.
4. $A \cdot \{\epsilon\} = \{\epsilon\} \cdot A = A$.
5. $A \cdot \emptyset = \emptyset \cdot A = \emptyset$.
6. $A \cdot (B \cdot C) = (A \cdot B) \cdot C$.
8. $\emptyset^* = \{\epsilon\}^* = \{\epsilon\}$.
10. $A \cdot (B \cup B) = A \cdot B \cup A \cdot C$.
11. $(A \cup B) \cdot C = A \cdot C \cup B \cdot C$.

Note: in general, $A \cdot (B \cap C) \neq A \cdot B \cap A \cdot C$.  

4
As an example of how these identities may be established, we now consider the following identity.

SHOW: $(L_1 \cup L_2)^* = (L_1^* \circ L_2^*)^*$

Step 1: We show that

$(L_1 \cup L_2)^* \subseteq (L_1^* \circ L_2^*)^*$

Let $w \in (L_1 \cup L_2)^*$. If $w = e$, then $w \in (L_1^* \circ L_2^*)^*$.

So, we take $w \neq e$. Then $w = v_1v_2\ldots v_n$, where each $v_j \in L_1 \cup L_2$.

But $v_1v_2\ldots v_n \in (L_1^*L_2^*)^n$.

We conclude: $(L_1 \cup L_2)^* \subseteq (L_1^* \circ L_2^*)^*$.

Step 2: We show that: $(L_1^* \circ L_2^*)^* \subseteq (L_1 \cup L_2)^*$

Let $w \in (L_1^* \circ L_2^*)^*$.

Then either $w = e$, or $w = v_1 \circ v_2 \circ \ldots v_n$, where $v_j \in L_1^* \circ L_2^*$.

So, each $v_j \in (L_1 \cup L_2)^*$ (Why?)

Hence, $w \in (L_1 \cup L_2)^*$, as well, since $(L_1 \cup L_2)^*$ is closed under concatenation.

**Definition.** If $L \subseteq \Sigma^*$, we let $L^R$ be the language:
\[ L^R = \{ w^R : w \in L \}. \]

**Proposition.** \((L_1 \circ L_2)^R = L_2^R \circ L_1^R.\)

**Proof.** Let \( w \in L_1 \circ L_2. \) Then \( w = w_1 \circ w_2, \) for some \( w_1 \in L_1 \) and \( w_2 \in L_2. \)

So, \( w^R = (w_1 \circ w_2)^R = w_2^R \circ w_1^R, \) which is in \( L_2^R \circ L_1^R. \)

We conclude: \((L_1 \circ L_2)^R \subseteq L_2^R \circ L_1^R.\)

Conversely, we let \( w \in L_2^R \circ L_1^R. \) Then \( w = v_2 \circ v_1, \) where \( v_2 \in L_2^R \) and \( v_1 \in L_1^R. \)

But \( v_2 = w_2^R \) and \( v_1 = w_1^R, \) for some strings \( w_1 \in L_1 \) and \( w_2 \in L_2. \)

Hence, \( w = w_2^R \circ w_1^R = (w_1 \circ w_2)^R. \)

We conclude: \( L_2^R \circ L_1^R \subseteq (L_1 \circ L_2)^R, \) so \( L_2^R \circ L_1^R = (L_1 \circ L_2)^R. \)

**Section Two: Regular Languages**

**2.1 Languages over Alphabets**

There is a fundamental problem that there are uncountably many distinct languages over any alphabet. (Note: we shall discuss the cardinality or size of infinite sets soon — so don’t worry if you do
not know what this term is for now.) Hence, there is no way that we can give distinct finite labels to denote any possible languages.

**Definition.** Let $\Sigma$ be any alphabet. The collection of *regular languages* over $\Sigma$ is given as follows:

1. $\emptyset$ is regular;
2. $\{\varepsilon\}$ is regular;
3. $\{\sigma\}$ is regular, for any $\sigma \in \Sigma$;
4. if $A$ and $B$ are regular languages, so are their union, concatenation, and Kleene star closure.

That is, $A \cup B, A \cdot B, A^*$.

2.2 Regular Languages and Regular Expressions

**Definition.** The regular expressions over an alphabet $\Sigma$ are all words over the extended alphabet $\Sigma \cup \{\emptyset, \cup, \ast\}$ such that

1. $\epsilon, \emptyset$, and $\sigma \in \Sigma$ are all regular expressions;
2. if $\alpha$ and $\beta$ are regular expressions, so is $\alpha \beta$.

(We sometimes write this as $\alpha \circ \beta$ or $\alpha \cdot \beta$);
3. if $\alpha$ and $\beta$ are regular expressions, so is $\alpha \cup \beta$;
4. if $\alpha$ is a regular expression, so is $\alpha^*$.
(5) Nothing else is regular.

In working with regular expressions, we shall use the usual order of precedence.

So, * has highest order,

○ has second highest order,

and \( \cup \) has third highest.

EXAMPLES OF REGULAR EXPRESSIONS

Let \( \Sigma \) be the two letter alphabet \( \{0, 1\} \).

1. \((0 \cup 1)^* = \Sigma^*\)

2. \(0^*1^*\)

gives the set of all binary strings with exactly ONE occurrence of 1

3. \((0 \cup 1)^*1(0 \cup 1)^*\)

gives the set of all binary strings with at least ONE occurrence of the symbol 1

4. \((0 \cup 1)^*001(0 \cup 1)^*\)

gives the set of all binary strings that contain 001 as a substring
5. \(((0 \cup 1)(0 \cup 1))^*\)
gives the set of all binary strings of EVEN length

6. \(((0 \cup 1)(0 \cup 1)(0 \cup 1))^*\)
gives the set of all binary strings whose length is a multiple of 3

7. \(0(0 \cup 1)^*0 \cup 1(0 \cup 1)^*1\)
gives the set of all binary strings that start and end in the same symbol

8. \(0^*10^*010^*(10^* \cup e)\)
is the set of all binary strings that have two or three occurrences of the symbol 1 such that the first and second occurrences of 1 are NOT consecutive.

9. \(0^*\)
is the set of all binary strings with no substring of the form 1

10. \(0^* \cup 0^*1(00^*1)^*0^* \cup (00^*100^*)^*\)
gives the set of all binary strings that do NOT have the substring 11

11. \((1^*01^*01^*)^*\)
gives the set of all binary strings with an EVEN number of occurrences of the symbol 0

12. \((00 \cup 11 \cup (01 \cup 10)(00 \cup 11)^*(01 \cup 10))^*\)

gives the set of all binary strings with an EVEN number of both symbols 0 and 1

We can define a formal map \(L\) from the collection of regular expressions to languages such that \(L\) has the following properties:

1. \(L(\emptyset) = \emptyset; L(a) = \{a\}\), for \(a \in \Sigma\).

2. \(L(\alpha \circ \beta) = L(\alpha) \circ L(\beta)\), where \(\alpha\) and \(\beta\) are regular expressions.

3. \(L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)\), where \(\alpha\) and \(\beta\) are regular expressions.

4. \(L(\alpha^*) = L(\alpha)^*\), where \(\alpha\) is a regular expression.

We call two regular expressions \(\alpha\) and \(\beta\) equivalent if \(L(\alpha) = L(\beta)\). We write equivalent regular expressions as \(\alpha \equiv \beta\) or \(\alpha \equiv \beta\).

**Proposition.** Let \(r\) denote a regular expression.

If \(L = L(r)\), then its reversal \(L(r)^R\) is also regular.
We shall prove this result later by induction on the number of operators in the regular expression \( r \).

Useful Identities for Regular Expressions:

1. \( \alpha \cup (\beta \cup \gamma) \equiv (\alpha \cup \beta) \cup \gamma \). “associativity of \( \cup \)”
2. \( \alpha \cup \beta \equiv \beta \cup \alpha \). “commutativity of \( \cup \)”
3. \( \alpha \cup \emptyset \equiv \alpha \). “\( \emptyset \) is an identity for \( \cup \)”
4. \( \alpha \cup \alpha \equiv \alpha \). “idempotence of \( \cup \)”
5. \( \alpha \circ (\beta \circ \gamma) \equiv (\alpha \circ \beta) \circ \gamma \). “associativity of \( \circ \)”
6. \( \epsilon \circ \alpha \equiv \alpha \epsilon \equiv \alpha \). “\( \epsilon \) is an identity for \( \circ \)”
7. \( \alpha \circ (\beta \cup \gamma) \equiv \alpha \circ \beta \cup \alpha \circ \beta \). “distributivity”
8. \( \emptyset \circ \alpha \equiv \alpha \circ \emptyset \equiv \emptyset \). “\( \emptyset \) is a zero for \( \circ \)”
9. \( \epsilon \cup \alpha \circ \alpha^* \equiv \alpha^* \).
10. \( \epsilon \cup \alpha^* \circ \alpha \equiv \alpha^* \).
11. \( \beta \cup \alpha \circ \gamma \leq \gamma \Rightarrow \alpha^* \circ \beta \leq \gamma \).
12. \( \beta \cup \gamma \circ \alpha \leq \gamma \Rightarrow \beta \circ \alpha^* \leq \gamma \).

Other useful identities:

1. \( (\alpha \circ \beta)^* \alpha \equiv \alpha \circ (\beta \circ \alpha)^* \).
(2) \((\alpha^* \circ \beta)^* \circ \alpha^* \equiv (\alpha \cup \beta)^*\). "denesting rule"

(3) \(\alpha^* \circ (\beta \circ \alpha^*)^* \equiv (\alpha \cup \beta)^*\). "shifting rule"

(4) \((\epsilon \cup \alpha)^* \equiv \alpha^*\).

(5) \(\alpha \circ \alpha^* \equiv \alpha^* \circ \alpha\).