Formal Languages

Progression of Concepts

- Symbol
- Alphabet
- String
- Language
Symbols

• Letters or numerals:

    a b c ... z          0 1 2 ... 9

• Bits:

    0 1

• English words:

    fox dog jobs

• Syntactic components of a programming language:

    for begin end while ; :=
Alphabet

An alphabet $\Sigma$ is a finite set of symbols.

**EXAMPLE.**

$$\Sigma_1 = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$$

**EXAMPLE.**

$$\Sigma_2 = \{0, 1\}$$

**EXAMPLE.**

The set of lexical elements of a programming language (keywords, syntax, identifiers, etc.)
String

A string over $\Sigma$ is a finite sequence of symbols from $\Sigma$.

EXAMPLE.

Strings over $\Sigma_1$ are sequences of letters:

- $r, a, b, b, i, t$
- $f, o, x$
- $e, a, g, l, e$

Generally just omit the commas:

- rabbit
- fox
- eagle

EXAMPLE.

Strings over $\Sigma_2$ are sequences of binary digits:

$$
\begin{align*}
1110011 \\
10001011101010010010010010101011 \\
01110100010101101101101101101010100
\end{align*}
$$
String Notions

Every string \( v \) has a \textbf{length}, denoted \( \text{length}(v) \), that is the length of the sequence of symbols.

The string with no symbols is the \textbf{null string} or \textbf{empty string}, which has length 0. It is denoted \( \lambda \), \( e \), or \( \epsilon \).

If \( u \) and \( v \) are two strings, then another string results if \( u \) is followed by \( v \). This is the \textbf{concatenation} of \( u \) and \( v \), denoted \( u \cdot v \) or just \( uv \).

The empty string is an identity for the concatenation operation. For every string \( u \),

\[
ul = u = \lambda u.
\]
Other String Notions

String $u$ is a **substring** of $v$ if $v = xuy$ for some strings $x$ and $y$.

String $u$ is a **prefix** of $v$ if $v = uy$ for some string $y$.

String $u$ is a **suffix** of $v$ if $v = xu$ for some string $x$.

String $v^R$ is the **reversal** of $v$ if it is the sequence $v$ in last-to-first order.

If $v = v^R$, then $v$ is a **palindrome**.

**EXERCISE.**

Apply the definitions to $v = bacb$.

How many palindromes of length 4 are there over $\{a, b, c\}$? Length 5?
Languages

Start with an alphabet $\Sigma$.

The set of all strings over $\Sigma$ is denoted $\Sigma^*$.

A language over $\Sigma$ is any subset of $\Sigma^*$.

**EXERCISE.** What can you say about the number of languages over $\Sigma$?
Sample Languages

Example languages over $\Sigma = \{a, b, c\}$:

- $\emptyset$, the empty language

- $\{aaab, aabb, abab, abbb, aacb, acab, accb, abcb, acbb\}$, the set of strings of length 4 that begin with $a$ and end with $b$

- $\{u \in \Sigma^* \mid \text{length}(u) \geq 7\}$, the set of all strings of length at least 7
Operations on Languages

- Union: \( L_1 \cup L_2 \)
- Intersection: \( L_1 \cap L_2 \)
- Complementation: \( \overline{L} = \Sigma^* - L \)
- Concatenation:

\[
L_1 L_2 = \{uv \mid u \in L_1, v \in L_2\}
\]

**EXAMPLE.**

\[
\{b, ba\}\{\lambda, a, ab\} = \{b, ba, bab, baa, baab\}
\]
Powers of Languages

- \( L^0 = \{\lambda\} \)

- For \( i > 0 \),
  \[ L^i = L^{i-1}L. \]

**EXERCISE.**

\[ \emptyset^0 = \square ? \]

\[ \{a, ba\}^3 = \square ? \]
Kleene Closure

The **Kleene closure** (or **Kleene star**) of a language $L$ is

$$L^* = \bigcup_{i=0}^{\infty} L^i.$$

The **Kleene plus** of a language $L$ is

$$L^+ = \bigcup_{i=1}^{\infty} L^i.$$

Some Facts

$$L^+ = LL^*$$
$$L^+ = L^* - \{\lambda\} \quad \text{if } \lambda \notin L$$
$$L^+ = L^* \quad \text{if } \lambda \in L$$
Exercises

1. \( \emptyset^* = \) ?

2. \( \emptyset^+ = \) ?

3. What language is described by

\[ \{a, b\}^* \{cab\}\{b, a\}^* \]

4. Give a recursive definition of \( L^* \).
Regular Languages

Fix an alphabet $\Sigma$. The set of **regular languages** or **regular sets** over $\Sigma$ is defined recursively:

1. **Basis:** The sets $\emptyset$, $\{\lambda\}$, and $\{a\}$, where $a \in \Sigma$, are regular sets.

2. **Recursive step:** If $L_1$ and $L_2$ are regular sets, then
   
   $L_1 \cup L_2,$
   $L_1L_2,$ and
   $L_1^*$

   are regular sets.

3. **Closure:** Only sets attainable by a finite number of applications of the recursive step to the basis are regular sets.
Regular Expressions

A regular expression over $\Sigma$ is defined recursively:

1. **Basis:** The expressions $\emptyset, \lambda,$ and $a$, where $a \in \Sigma$, are regular expressions representing, respectively, $\emptyset$, $\{\lambda\}$, and $\{a\}$.

2. **Recursive step:** If $u_1$ and $u_2$ are regular expressions representing, respectively, languages $L_1$ and $L_2$, then $(u_1 \cup u_2)$, $(u_1 u_2)$, and $(u_1)^*$ are regular expressions representing, respectively, $L_1 \cup L_2$, $L_1 L_2$, and $L_1^*$.

3. **Closure:** Only expressions attainable by a finite number of applications of the recursive step to the basis are regular expressions.
Examples

An algebraic notation for representing regular languages.

EXAMPLE.

The regular expression

\[ (((b \cup (ba)) (\lambda \cup (a \cup (ab)))) \] represents the regular language

\[ \{b, ba\} \{\lambda, a, ab\} = \{b, ba, bab, baa, baab\} \]

Precedence: Kleene closure, concatenation, union (highest to lowest). Allows dropping unnecessary parentheses.

EXAMPLE.

The revised regular expression

\[ (b \cup ba)(\lambda \cup a \cup ab) \] also represents the regular language

\[ \{b, ba\} \{\lambda, a, ab\} \]
Further Examples

Abusing notation, we often write

$$(b \cup ba)(\lambda \cup a \cup ab) = \{b, ba\}\{\lambda, a, ab\}.$$ 

Fix $\Sigma = \{a, b, c\}$. Then

$$\overline{(a \cup b \cup c)^*} = \Sigma^*.$$ 

As shorthand, let $u^+$ represent the same language as $uu^*$. 

Two regular expressions can represent the same language:

$$\Sigma^* = \overline{(a \cup b \cup c)^*} = \overline{(a^*b^*c^*)^*}.$$
Representation Exercises

Problem 12. The set of strings over \( \{a, b, c\} \) in which all the \( a \)'s precede all the \( b \)'s, which in turn precede all the \( c \)'s

Problem 13. The same except excluding the empty string

Problem 21. The set of strings over \( \{a, b\} \) in which the substring \( aa \) occurs exactly once
Regular Expression Identities

Table 2.3.1. Show these identities:

7. \[ u \cup u = u \]

10. \[ (u \cup v)w = uw \cup vw \]

11. \[ (uv)^*u = u(vu)^* \]
Regular Expression Identities

Problem 38 (d). Use the identities in Table 2.3.1 to establish this identity:

\[(a \cup b)^* = (a^* \cup ba^*)^*\.

Answer with explanations:

\[(a \cup b)^* = (a \cup ba^*)^*
\]

12. \[(u \cup v)^* = (u \cup vu^*)^*\]

= \[(a^* \cup ba^*)^*
\]

12. \[(u \cup v)^* = (u^* \cup v)^*\]