Strings and Languages

Assylbek Issakhov,
Ph.D., professor of School of Math and Cybernetics

Strings and Languages

- The basic object in automata and language theory is a *string*. A string is a finite sequence of *symbols*. For example, the following are three strings and the corresponding sets of symbols in the strings:
- $strings$ $\{s, t, r, i, n, g\}$
- $CS5400$ $\{C, S, 5, 4, 0\}$
- 1001 $\{1, 0\}$

Strings and Languages

- In a formal theory, it is necessary to fix the set of symbols used to form strings. Such a finite set of symbols is called an *alphabet*. For example, the following are three alphabets:
- $\{a, b, c, \dots, x, y, z\}$ (Roman alphabet)
- $\{0, 1, \dots, 9\}$ (Arabic digits)
- $\{0, 1\}$ (binary alphabet)
- A string over the binary alphabet is called a *binary string*.

Strings and Languages

- In general, an alphabet may be defined by a finite set of strings instead of symbols, as long as it satisfies the property that two different finite sequences of its elements form two different strings. For instance, the set $\{00,01,11\}$ is an alphabet, but $\{00,0,1\}$ is not an alphabet because both sequences $(0,0)$ and (00) form the same string 00. Usually, we do not consider this general type of alphabets, and will only work with alphabets whose elements are single symbols.

Strings and Languages

- The *length* of a string x, denoted by $|x|$, is the number of symbols contained in the string. For example,

$$|strings| = 7,$$

$$|CS5400| = 6,$$

$$|1001| = 4.$$

- The *empty string*, denoted by ε, is a string having no symbol. Clearly, $|\varepsilon| = 0$.

Strings and Languages

- **Example 1.1**. How many strings over the alphabet

$$A = \{a_1, a_2, \dots, a_k\}$$

- are there which are of length n, where n is a nonnegative integer?

- **Solution**. There are n positions in such a string, and each position can hold one of k possible symbols. Therefore, there are k^n strings of length exactly n.

Strings and Languages

- Let x and y be two strings, and write $x = x_1 x_2 \dots x_n$ and $y = y_1 y_2 \dots y_m$, where each x_i and each y_j is a single symbol. Then, x and y are equal if and only if (1) $n = m$ and (2) $x_i = y_i$ for all $i = 1, 2, \dots, n$. For example, $01 \neq 010$ and $1010 \neq 0101$.
- The basic operation on strings is *concatenation*. The concatenation $x \cdot y$ of two strings x and y is the string xy, that is, x followed by y.

Strings and Languages

- For example, $CS5400$ is the concatenation of CS and 5400. In particular, we denote

$$x = x^1, xx = x^2, \dots, xx \dots x = x^k,$$

- and define $x^0 = \varepsilon$. (Why is $x^0 = \varepsilon$? The reason is that ε is the identity for the operation of concatenation, and so x^0 satisfies the relation $x^0 x^k = x^{0+k} = x^k$.) For example, $10101010 = (10)^4 = (1010)^2$, $(10)^0 = \varepsilon$. It is obvious that $x^i x^j = x^{i+j}$ for $i, j > 0$.

Strings and Languages

- Let x be a string. A string s is a *substring* of x if there exist strings y and z such that

$$x = ysz.$$

- In particular, when $x = sz$ $(y = \varepsilon)$, s is called a *prefix* of x; and when $x = ys$ $(z = \varepsilon)$, s is called a *suffix* of x.
- For example, CS is a prefix of $CS5400$ and 5400 is a suffix of $CS5400$.

Strings and Languages

- For a string x over alphabet Σ, the *reversal* of x, denoted by x^R, is defined by

$$x^R = \begin{cases} \varepsilon & , if\ x = \varepsilon \\ x_n \dots x_2 x_1 & , if\ x = x_1 \dots x_n \end{cases}$$

- for $x_1, x_2, \dots x_n \in \Sigma$.

Strings and Languages

- **Example 1.2**. For strings x and y, $(xy)^R = y^R x^R$.
- **Proof.** If $x = \varepsilon$, then $x^R = \varepsilon$ and hence $(xy)^R = y^R = y^R x^R$. If $y = \varepsilon$, then $y^R = \varepsilon$ and hence $(xy)^R = x^R = y^R x^R$. Now, suppose $x = x_1 \dots x_m$ and $y = y_1 \dots y_n$ with $m, n \geq 1$. Then

$$(xy)^R = (x_1 \dots x_m y_1 \dots y_n)^R =$$

$$= y_n \dots y_1 x_m \dots x_1 = y^R x^R.$$

Strings and Languages

- Strings are also called *words*. Relations between strings form a theory, called *word theory*.
- For instance, in word theory, we may be given an equation of strings and are asked to find the solution strings for the variables in the equation.

Strings and Languages

- **Example 1.3**. Solve the word equation
$$x011 = 011x$$
- over the alphabet $\{0, 1\}$, that is, find the set of strings x over $\{0, 1\}$ which satisfy the equation.
- **Solution**. For the equation to hold, either x is the empty string or the string 011 is both a prefix and a suffix of x:

$$011[\ldots x \ldots] = [\ldots x \ldots]011$$

Strings and Languages

- (It is obvious that x cannot be of length 1 or 2.) Let $x = 011y$. Now, remove the first occurrence of 011 from both $011x$ and $x011$, we get $x = y011$. It follows that

$$011y = y011.$$

- This gives us a recursive solution for x: x is either ε or $x = 011y$ for some other solution y of the equation. It is not hard to see now that $(011)^n$ is a solution to the equation for each $n \geq 0$, and they are the only solutions.

Strings and Languages

- A *language* is a set of strings. For example,

$$\{0, 1\}, \{0^0, 0^1, 0^2, \dots\},$$

- and the set of all English words are languages. Let Σ be an alphabet. We write Σ^* to denote the set of all strings over Σ. Thus, a language L over Σ is just a subset of Σ^*. For any finite language $A \subseteq \Sigma^*$, we write $|A|$ to denote the size (i.e., the number of strings) in A.

Strings and Languages

- The following are some basic operations on languages. (The first four are just set operations. See Figure 1.1.)
- *Union*: If A and B are two languages, then $A \cup B = \{x|x \in A\ or\ x \in B\}$.
- *Intersection*: If A and B are two languages, then $A \cap B = \{x|x \in A\ and\ x \in B\}$.
- Subtraction: If A and B are two languages, then $A\backslash B = \{x|x \in A\ and\ x \notin B\}$. ($A\backslash B$ is also denoted by A-B when $B \subseteq A$.)

Strings and Languages

- *Complementation*: If A is a language over the alphabet Σ, then $\bar{A} = \Sigma^* - A$.
- *Concatenation*: If A and B are two languages, then their concatenation is $A \cdot B = \{ab | a \in A \; and \; b \in B\}$. We also write AB for $A \cdot B$.
- It is clear that concatenation satisfies the associativity law, and so we do not need parentheses when we write the concatenation of more than two languages: $A_1 A_2 \dots A_k$.

Strings and Languages

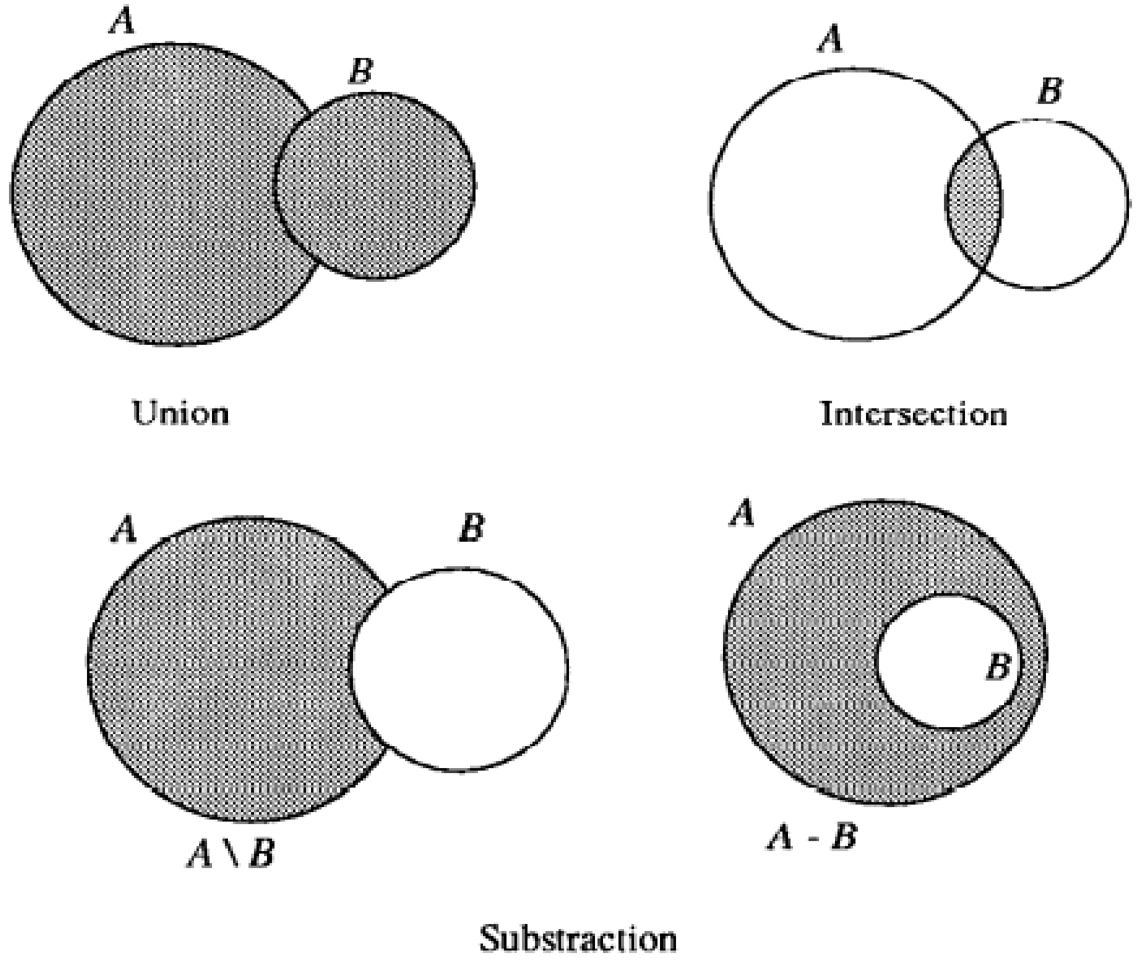

Figure 1.1: Set operations.

Strings and Languages

- **Example 1.4.** (a) If $A = \{0,1\}$ and $B = \{1, 2\}$, then $AB = \{01, 02, 11, 12\}$.
- (b) Is it true that if A is of size $n \geq 0$ and B is of size $m \geq 0$, then AB must be of size nm?
- The answer is no. For instance, if $A = \{0,01\}$ and $B = \{1,11\}$, then $AB = \{01,011,0111\}$ has only three elements.

Strings and Languages

- (c) Let $A = \{(01)^n | n \geq 0\}$ and $B = \{01, 010\}$. Then

$$AB = \{(01)^n, (01)^n 0 \mid n \geq 1\}$$

$$ABA = \{(01)^n \mid n \geq 1\} \cup \{(01)^n 0 (01)^m \mid m \geq 0, n \geq 1\}$$

- For any language A, we define

$$A^1 = A, A^2 = AA, \ldots, A^k = AA^{k-1}$$

- for $k \geq 2$. We also define $A^0 = \{\varepsilon\}$.

Strings and Languages

- Note that $\emptyset$ and $\{\varepsilon\}$ are two different languages: $\emptyset A = \emptyset$, and $\{\varepsilon\}A = A\{\varepsilon\} = A$.
- For example, for $\Sigma = \{0,1\}$ we have $\Sigma^2 = \{00, 01, 10, 11\}$, and, in general, for $k \geq 0$, Σ^k is the set of all strings of length k over Σ. Therefore,

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots.$$

- The following is the more general star operation based on this formula:

Strings and Languages

- *Kleene closure* (or *star closure*): For any language A, define
$$A^* = A^0 \cup A^1 \cup A^2 \cup \cdots$$
$$= \{w | w \; is \; the \; concatenation \; of \; 0 \; or \; more \; strings \; from \; A\}$$
- **Example 1.5.** The language $\{0, 10\}^*$ is the set of all binary strings having no substring 11 and ending with 0.

Strings and Languages

- **Proof**. It is clear that the concatenation of any number of 0 and 10 must end with 0. Furthermore, it cannot produce a substring 11, since the ending 0's of both strings 0 and 10 separate any two 1's in the concatenated string.
- Conversely, let x be a string over $\{0,1\}$ having no substring 11 and ending with 0. If x contains no occurrence of 1, then x is the concatenation of $|x|$ many 0's, and so $x \in \{0,10\}^{|x|} \subseteq \{0,10\}^*$. Suppose x contains $n \geq 1$ occurrences of 1's.

Strings and Languages

- Then, each occurrence of 1 in x must be followed by a 0, for otherwise that symbol 1 is either followed by a 1 or is the last symbol of x, violating the assumption on x. So, we can write x as

$$0 \ldots 0(10)0 \ldots 0(10)0 \ldots 0(10)0 \ldots 0,$$

- where $0 \ldots 0$ means zero or more 0's. Thus, x is the concatenation of strings 0 and 10, or, $x \in \{0,10\}^*$.

Strings and Languages

- **Example 1.6**. Show that for any languages A and B,

$$(A \cup B)^* = A^*(BA^*)^*$$

- **Proof**. We observe that every string in $A^*(BA^*)^*$ can be written as the concatenation of strings in $A \cup B$. Indeed, a string x in $A^*(BA^*)^*$ must be in $A^n(BA^*)^m$ for some $n, m \geq 0$. Thus, x can be decomposed into

$$x = x_1 x_2 \dots x_n y_1 y_2 \dots y_m$$

- where $x_1, x_2, \dots, x_n \in A$ and $y_1, y_2, \dots, y_m \in BA^*$.

Strings and Languages

- Similarly, each $y_j, j = 1, \dots, m$, can be decomposed into

$$y_j = y_{j,0}y_{j,1}y_{j,2} \cdots y_{j,k_j}$$

- with $k_j \geq 0$ and $y_{j,1}, y_{j,2}, \dots, y_{j,k_j} \in A$. Therefore,

$$x = x_1x_2 \dots x_n y_{1,0}y_{1,1} \cdots y_{1,k_1}y_{2,0} \cdots y_{2,k_2} \cdots y_{m,k_m}$$

- is the concatenation of strings in $A \cup B$. It follows that $A^*(BA^*)^* \subseteq (A \cup B)^*$.

Strings and Languages

- Next, we show that $(A \cup B)^* \subseteq A^*(BA^*)^*$. To do so, consider a general string $x \in (A \cup B)^*$. Again, we can see that $x \in (A \cup B)^n$ for some $n \geq 0$. Thus, we may write

$$x = x_1 x_2 \dots x_n,$$

- for some $x_1, x_2, \dots, x_n \in A \cup B$. Now, assume that $x_{i_1}, x_{i_2}, \dots, x_{i_k} \in B$, for some $k \geq 0$ and $1 \leq i_1 < i_2 < \ \dots < i_k \leq n$, and the other strings x_j, with $j \neq i_1, i_2, \dots, i_k$, are in A.

Strings and Languages

- Then, we can write

$$x = y_{i_1} x_{i_1} y_{i_2} x_{i_2} \ \dots y_{i_k} x_{i_k} y_{i_{k+1}}$$

- where each $y_{i_j} \in A^*$. Thus, $x \in A^*(BA^*)^k \subseteq A^*(BA^*)^*$. It follows that $(A \cup B)^* \subseteq A^*(BA^*)^*$.

- Define the *positive closure* of a language A to be

$$A^+ = A^*A = A \cup A^2 \cup A^3 \cup \ \dots .$$

Strings and Languages

- **Example 1.7.** Prove that $A^{+} = A^{*}$ if and only if $\varepsilon \in A$.
- **Proof**. Clearly, $A^{+} \subseteq A^{*}$. If $\varepsilon \in A$, then

$$\{\varepsilon\} = A^{0} \subseteq A \subseteq A^{+}.$$

- Thus, $A^{*} = A^{+}$.
- Conversely, if $\varepsilon \notin A$, then every string in A^{+} has positive length. Thus, A^{+} does not contains ε. But, $\varepsilon \in A^{*}$. Hence, $A^{*} \neq A^{+}$.

Strings and Languages

- For a language A, define the *reversal language* of A to be

$$A^R = \{x^R \mid x \in A\}$$

- **Example 1.8**. For languages A and B,

$$(AB)^R = B^R A^R,$$

$$(A \cup B)^R = A^R \cup B^R.$$

Strings and Languages

- **Proof**.

$$
\begin{aligned}
(AB)^R &= \{x^R \mid x \in AB\} \\
&= \{(yz)^R \mid y \in A, z \in B\} \\
&= \{z^R y^R \mid y \in A, z \in B\} \\
&= \{z^R \mid z \in B\} \cdot \{y^R \mid y \in A\} = B^R A^R,
\end{aligned}
$$

$$
\begin{aligned}
(A \cup B)^R &= \{x^R \mid x \in A \cup B\} \\
&= \{x^R \mid x \in A\} \cup \{x^R \mid x \in B\} \\
&= A^R \cup B^R
\end{aligned}
$$

Strings and Languages

- **Example 1.9 (Arden's Lemma)**. Assume that A, B are two languages with $\varepsilon \notin A$, and X is a language satisfying the relation $X = AX \cup B$. Then, $X = A^*B$.
- **Proof.** We use induction to show $X \subseteq A^*B$. First, consider $x = \varepsilon$. If $x \in X$, then $x \in AX \cup B$. Since $\varepsilon \notin A$, we must have $x \in B$ and, hence, $x \in A^*B$.
- Next, assume that for all strings w of length less than or equal to n, if $w \in X$ then $w \in A^*B$, and consider a string x of length $n + 1$.

Strings and Languages

- If $x \in X = AX \cup B$, then either $x \in B \subseteq A^*B$ or $x = yw$ for $y \in A$ and $w \in X$. In the second case, we must have $y \neq \varepsilon$ and, hence, $|w| < |x|$. So, by the inductive hypothesis, $w \in A^*B$ and $x \in AA^*B \subseteq A^*B$. This completes the induction step, and it follows that $X \subseteq A^*B$.

- Conversely, we use induction to show that $A^nB \subseteq X$ for all $n \geq 0$. For $n = 0$, we have $A^0B = B \subseteq AX \cup B = X$. For $n > 0$, we have, by the inductive hypothesis, $A^nB = A(A^{n-1}B) \subseteq AX$. Thus $A^nB \subseteq AX \subseteq AX \cup B = X$.

Strings and Languages

- **Example 1.10**. Assume that languages $A, B \subseteq \{a, b\}^*$ satisfy the following two equations:

$$A = \{\varepsilon\} \cup \{a\}A \cup \{b\}B,$$

$$B = \{\varepsilon\} \cup \{b\}B.$$

- Find simple representations for A and B.

Strings and Languages

- **Proof**. We apply Arden's lemma to the second equation, and we get $B = \{b\}^* \cdot \{\varepsilon\} = \{b\}^*$. Then, we apply Arden's lemma to the first equation, and we get

$$A = \{a\}^*(\{\varepsilon\} \cup \{b\}B).$$

- Now, substitute $\{b\}^*$ for B, we have

$$A = \{a\}^*(\{\varepsilon\} \cup \{b\}\{b\}^*) = \{a\}^*\{b\}^*.$$

Regular Languages and Regular Expressions

- The concept of *regular languages* (or, *regular sets*) over an alphabet Σ is defined recursively as follows:
- (1) The empty set $\emptyset$ is a regular language.
- (2) For every symbol $a \in \Sigma, \{\mathrm{a}\}$ is a regular language.
- (3) If A and B are regular languages, then $A \cup B, AB$ and A^* are all regular languages.
- (4) Nothing else is a regular language.

Regular Languages and Regular Expressions

- **Example 1.11**. (a) The set $\{\varepsilon\}$ is a regular language, because $\{\varepsilon\} = \emptyset^*$.
- (b) The set $\{001,110\}$ is a regular language over the binary alphabet:

$$\{001,110\} = (\{0\}\{0\}\{1\}) \cup (\{1\}\{1\}\{0\}).$$

- (c) From (b) above, we can generalize that every finite language is a regular language.

Regular Languages and Regular Expressions

- When a regular language is obtained through a long sequence of operations of union, concatenation and Kleene closure, its representation becomes cumbersome. For example, it may look like this:

$$(\{0\}^* \cup (\{1\}\{0\}\{0\}^*))\{1\}\{0\}^*(\{0\}\{1\}^* \cup \{1\}^*) \tag{1.1}$$

- To simplify the representations for regular languages, we define the notion of *regular expressions* over alphabet Σ as follows:

Regular Languages and Regular Expressions

- (1) $\emptyset$ is a regular expression which represents the empty set.
- 2) ε is a regular expression which represents language $\{\varepsilon\}$.
- (3) For $a \in \Sigma$, a is a regular expression which represents language $\{a\}$.

Regular Languages and Regular Expressions

- (4) If r_A and r_B are regular expressions representing languages A and B, respectively, then $(r_A) + (r_B), (r_A)(r_B), (r_A)^*$ are regular expressions representing $A \cup B, AB$ and A^*, respectively.
- (5) Nothing else is a regular expression over Σ.
- For example, language $A = \{0\}^*$ has a regular expression $r_A = (0)^*$ and language $B = \{00\}^* \cup \{0\}$ has a regular expression $r_B = \left(\left((0)(0)\right)^*\right) + (0)$.

Regular Languages and Regular Expressions

- For any regular expression r, we let $L(r)$ denote the regular language represented by r.
- To further reduce the number of parentheses in a regular expression, we apply the following preference rules to a non-fully parenthesized regular expression:
- (1) Kleene closure has the higher preference over union and concatenation.
- (2) Concatenation has the higher preference over union.

Regular Languages and Regular Expressions

- In other words, we interpret a regular expression like an arithmetic expression, treating union like addition, concatenation like multiplication, and Kleene closure like exponentiation. (This is exactly why we use the symbol $+$ for union, the symbol $\cdot$ for concatenation, and the symbol $*$ for Kleene closure.)
- Using these rules, we can simplify the above two regular expressions to $r_A = 0^*$ and $r_B = (00)^* + 0$, respectively.

Regular Languages and Regular Expressions

- The regular expression (1.1) can also be simplified to

$$(0^* + 100^*)10^*(01^* + 1^*).$$

- In addition, like the operations $+$ and $\cdot$ in an arithmetic expression, the operations $+$ and $\cdot$ in a regular expression satisfy the *distributive law*: For any regular expressions r, s and t,

$$r(s+t) = rs + rt$$

$$(r+s)t = rt + st.$$

Regular Languages and Regular Expressions

- A regular language may have several regular expressions. For example, both $0^*1 + \emptyset$ and 0^*1 represent the same regular set $\{0\}^*\{1\}$.
- The following are some examples of identities about regular expressions.
- (When there is no risk of confusion, we use the Roman letter a to denote both the symbol a in the alphabet of the language and the regular expression representing the set $\{a\}$).

Regular Languages and Regular Expressions

- **Example 1.12**. $a^*(a+b)^* = (a+ba^*)^*$.
- **Proof**. We show that both sides are equal to $(a+b)^*$.
- Clearly, both sides are subsets of $(a+b)^*$ since $(a+b)^*$ contains all strings over alphabet $\{a, b\}$. Thus, it suffices to show that both sides contain $(a+b)^*$. Since $\varepsilon \in a^*$, we have $a^*(a+b)^* \supseteq (a+b)^*$. Also, $b \in ba^*$ and it follows that $(a+b)^* \subseteq (a+ba^*)^*$.

Regular Languages and Regular Expressions

- For convenience, we define an additional notation:
$$r^+ = rr^*.$$
- **Example 1.13**. $(ba)^+(a^*b^* + a^*) = (ba)^*ba^+b^*$.
- **Proof**. $(ba)^+(a^*b^* + a^*) = (ba)^*(ba)a^*(b^* + \varepsilon) = (ba)^*ba^+b^*$.
- Regular expressions can be a convenient notation to represent regular languages, if one knows how to construct them. The following examples demonstrate some ideas.

Regular Languages and Regular Expressions

- **Example 1.14**. Find a regular expression for the set of binary expansions of integers which are the power of 4.
- **Solution**. The binary expansion of the integer 4^n is $100 \dots 0$, where 0 is repeated $2n$ times, can be represented by $1(00)^*$.

Regular Languages and Regular Expressions

- **Example 1.15**. Find a regular expression for the set of binary strings which have at least one occurrence of the substring 001.
- **Solution**. Such a string can be written as $x011y$, where x and y could be any binary strings. So, we get a regular expression for this set:

$$(0+1)^*001(0+1)^*.$$

Regular Languages and Regular Expressions

- **Example 1.16**. Find a regular expression for the set A of binary strings which have no substring 001.
- **Solution**. A string x in this set has no substring 00, except that it may have a suffix 0^k for $k \geq 2$. The set of strings with no substring 00 can be represented by the regular expression

$$(01 + 1)^*(\varepsilon + 0)$$

- Therefore, set A has a regular expression

$$(01 + 1)^*(\varepsilon + 0 + 000^*) = (01 + 1)^*0^*$$

Regular Languages and Regular Expressions

- **Example 1.17**. Find a regular expression for the set B of all binary strings with at most one pair of consecutive 0's and at most one pair of consecutive 1's.
- **Solution**. A string x in B may have one of the following forms:

$$(1)\ \varepsilon,\ (2)\ u_1 0,\ (3)\ u_0 1,\ (4)\ u_1 00 v_1,$$

$$(5)\ u_0 11 v_0,\ (6)\ u_1 00 w_1 11 v_0,\ (7)\ u_0 11 w_0 00 v_1$$

Regular Languages and Regular Expressions

- where $u_0, u_1, v_0, v_1, w_0, w_1$ are strings with no substring 00 or 11, and u_0 ends with 0, u_1 ends with 1, v_0 begins with 0, v_1 begins with 1, w_0 begins with 0 and ends with 1, and w_1 begins with 1 and ends with 0.

- Now, observe that these types of strings can be represented by simple regular expressions:

$$u_1 0: (\varepsilon + 0)(10)^*$$
$$0 v_1: (01)^*(\varepsilon + 0)$$
$$0 w_1 1: (01)^*$$

Regular Languages and Regular Expressions

- (For convenience, we added ε to each case. Note that ε is in B.)
- Now, we can combine cases (1), (2), (4), (6) and use the distributive law to simplify it into the following regular expression:

$$
\begin{aligned}
&(\varepsilon + 0)(10)^*\big(\varepsilon + (01)^*(\varepsilon + 0) \\
&\qquad + (01)^*(10)^*(\varepsilon + 1)\big) \\
&\qquad = (\varepsilon + 0)(10)^*(01)^*\big(0 + (10)^*(\varepsilon + 1)\big)
\end{aligned}
$$

Regular Languages and Regular Expressions

- Note that

$$\varepsilon + (01)^*\varepsilon = (01)^*.$$

- cases (1), (3), (5), (7) have a symmetric form, and set B has the following regular expression:

$$(\varepsilon + 0)(10)^*(01)^*\big(0 + (10)^*(\varepsilon + 1)\big) + \\ (\varepsilon + 1)(01)^*(10)^*\big(1 + (01)^*(\varepsilon + 0)\big).$$

Regular Languages and Regular Expressions

- **Example 1.18**. Find a regular expression for the set of all binary strings with the property that none of its prefixes has two more 0's than 1's nor two more 1's than 0's.
- **Solution**. Consider a string $x = x_1 x_2 \dots x_n$ in the language, where each x_i is a bit 0 or 1. The given property implies that for any positive integer $i \leq n/2, x_{2i-1} \neq x_{2i}$. To see this, we assume, for the sake of contradiction, that there exists a positive integer $i \leq n/2$ such that $x_{2i-1} = x_{2i}$.

Regular Languages and Regular Expressions

- Let i^* be the smallest such i. Without loss of generality, assume $x_{2i^*-1} = x_{2i^*} = 0$. Then, each pair of

$$x_1 x_2, x_3 x_4, \ldots, x_{2i^*-3} x_{2i^*-2}$$

- is either 01 or 10 and, hence, the prefix

$$x_1 x_2 \ldots x_{2i^*-2}$$

- has an equal number of 0's and 1's. It follows that the prefix $x_1 x_2 \ldots x_{2i^*-1} x_{2i^*}$ contains two more 0's than 1's, a contradiction.

Regular Languages and Regular Expressions

- Conversely, any string x satisfying that, for all positive integers $i \le n/2$, $x_{2i-1} \ne x_{2i}$ belongs to this language, since each pair $x_{2i-1}x_{2i}$ is either 10 or 01.
- From this characterization, it is now easy to see that this language can be represented by the regular expression

$$(01 + 10)^*(0 + 1 + \varepsilon).$$