CHAPTER 1

REGULAR LANGUAGES AND EXPRESSIONS

1.1 NOTATION

We begin with a brief introduction to the basic mathematical terminology and notation.

Sets will be denoted by capital letters, e.g. X, Y and Z. The cardinality of a set X will be denoted by $\#X$. If all the sets under consideration are subsets of some set I, then I is called a universal set. The empty set is ϕ. We use the following notation, where X, Y are subsets of I:

$X \subseteq Y$	X is a subset of Y
$X \subset Y$	X is a proper subset of Y
$X \cup Y$	the union of X and Y
$X \cap Y$	the intersection of X and Y
$X - Y$	the difference of X and Y
$\bar{X} = I-X$	the complement of X in I
$X \Delta Y$	the symmetric difference of X and Y $X \Delta Y = (X-Y) \cup (Y-X)$
$P(X)$	the power set of X, i.e. the set of all subsets of X.

A family $\mathcal{B}$ of subsets of a set I is a boolean algebra iff:

1) $X \in \mathcal{B}$ implies $\bar{X} \in \mathcal{B}$, and

2) $X,Y \in \mathcal{B}$ implies $X \cup Y \in \mathcal{B}$.

In view of the fact $X \cap Y = \overline{(\bar{X} \cup \bar{Y})}$, 1) and 2) imply that $X \cap Y \in \mathcal{B}$. Also $X-Y$ and $X \Delta Y$ are in $\mathcal{B}$ if 1) and 2) hold, since $X-Y = X \cap \bar{Y}$ and $X \Delta Y = (X-Y) \cup (Y-X)$.

The cartesian product $X \times Y$ of two sets X and Y is

$$X \times Y = \{(x, y) \mid x \in X, y \in Y\} ,$$

where (x, y) is an ordered pair of elements. A binary relation ρ between sets X and Y is any subset of $X \times Y$. If $(x, y) \in \rho$ we also write $x\rho y$ to indicate that x is related by ρ to y. A binary relation $\rho \subseteq X \times X$ is called a binary relation on X. The converse ρ^{-1} of a relation is

$$\rho^{-1} = \{(x, y) \mid (y, x) \in \rho\}.$$

We will usually consider relations on a set X. A binary relation ρ on X is reflexive iff $x\rho x$ for all $x \in X$. It is symmetric iff $x\rho y$ implies $y\rho x$ for all $x,y \in X$, and antisymmetric iff $x\rho y$ and $y\rho x$ implies $x = y$ for all $x,y \in X$.

If $\rho_1 \subseteq X \times Y$ and $\rho_2 \subseteq Y \times Z$ are binary relations, their composition $\rho_1 \circ \rho_2$ is a binary relation between X and Z,

$$\rho_1 \circ \rho_2 = \{(x, z) \mid x\rho_1 y \text{ and } y\rho_2 z \text{ for some } y \in Y\} .$$

One verifies that the composition of binary relations on a set X is associative, i.e. that $\rho_1 \circ (\rho_2 \circ \rho_3) = (\rho_1 \circ \rho_2) \circ \rho_3$ for all ρ_1, ρ_2, and ρ_3. We write ρ^2 as a shorthand for $\rho\circ\rho$ and $\rho^n = \rho^{n-1} \circ \rho$, in general.

A partial order on a set X is a binary relation on X that is reflexive, antisymmetric, and transitive. An equivalence relation on

Mathematical terminology

set, subset, cardinality, universal set

subset operations:
    union, intersection, difference,
    complement, symmetric difference

Power set, Boolean Algebra, De Morgan's law

Binary relation
Relations with special properties: converse relation
    reflexive, symmetric, antisymmetric relations

composition of relations

partial order relation, equivalence relation, equivalence classes,

induced quotient sets, index of an equivalence relation

X is a binary relation on X that is reflexive, symmetric, and transitive. Given an equivalence relation ρ on X and any $x \in X$, let $[x]_\rho = \{y \in X \mid x\rho y\}$. Then $[x]_\rho$ is called the ρ-equivalence class of x. Note that $[x]_\rho = [y]_\rho$ iff $x\rho y$, and if $[x]_\rho \neq [y]_\rho$ then $[x]_\rho$ and $[y]_\rho$ are disjoint. Let $X/\rho = \{[x]_\rho \mid x \in X\}$; X/ρ is called the quotient set of X by ρ. A set π is a partition of X iff the elements of π are subsets of X and there exists for each $x \in X$ exactly one $S \in \pi$ such that $x \in S$. Clearly X/ρ is a partition of X, if ρ is an equivalence relation on X. Conversely, for any partition π on X we can define the equivalence relation

$$\rho_\pi = \{(x, y) \in X \times X \mid x \in S \text{ and } y \in S \text{ for some } S \in \pi\},$$

and $\pi = X/\rho_\pi$. The index of an equivalence relation ρ on X is the number of equivalence classes, i.e. the cardinality of X/ρ.

A function f from set X to set Y is a binary relation between X and Y such that for every $x \in X$ there exists $y \in Y$ such that xfy, and xfy_1 and xfy_2 implies $y_1 = y_2$. We write $f : X \to Y$ to indicate that f is a function from X to Y. For any $x \in X$ the unique element y such that xfy is denoted $f(x)$. The set $Y' = \{y \in Y \mid y = f(x) \text{ for some } x \in X\}$ is the image of X under f and is denoted by $f(X)$. We say that f is surjective or onto iff $Y = f(X)$; f is injective or one-to-one whenever $x_1 \neq x_2$ implies $f(x_1) \neq f(x_2)$; f is bijective or one-to-one and onto iff it is both injective and surjective. By a restriction of a function $f : X \to Y$ to $X' \subseteq X$ we mean the function $f' : X' \to Y$ such that $f'(x) = f(x)$ for all $x \in X'$. Usually f' is also denoted by f

A semigroup is an algebraic system $S = \langle X, \cdot \rangle$, where X is a set and $\cdot$ is a binary operation on X that satisfies the associative law:

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z, \qquad (1)$$

for all $x,y,z \in X$. The binary operation is often called multiplication and the dot is usually omitted.

A monoid is an algebraic system $M = \langle X, \cdot, 1_M \rangle$, where $\langle X, \cdot \rangle$ is a semigroup and $1_M \in X$ is a unit element satisfying:

$$x 1_M = 1_M x = x, \qquad (2)$$

for all $x \in X$. It is easily verified that a semigroup can have only one element satisfying (2); i.e. the unit element is unique. If there is no risk of ambiguity we simply write 1 for 1_M.

As an example, consider any set X. Then the systems $\langle P(X), \cup, \phi \rangle$, $\langle P(X), \cap, X \rangle$ and $\langle P(X), \Delta, \phi \rangle$ are all monoids. Also, if N is the set of non-negative integers, then $\langle N, +, 0 \rangle$ and $\langle N, \cdot, 1 \rangle$ are monoids, where + and $\cdot$ denote ordinary addition and multiplication.

Mathematical terminology cont.

Function relation, image,
 onto (surjective) functions
 one-to-one (injective) functions,
 bijective functions
 restriction of a function

Semi groups and monoids.

Alphabet A, letters, word, word length, empty word,
 (concatenation) product, free monoid generated by A, A*

Languages (subsets of A*)

Language operations :
 word operations extended to the elements of the power set,
 multiplication (product) associative ,
 the empty-word language is the multiplicative identity.

1.2 LANGUAGES

Let A be a finite, non-empty set. We will refer to A as an alphabet, and to the elements $a \in A$ as letters. Any finite sequence or string of letters is called a word. For example, if $A = \{a,b\}$, then a and babb are words. Note that we make no distinction between a letter and the word consisting of that letter; the meaning is clear from the context. The number of letters in a word x is called its length and is denoted by $|x|$. Thus $|a| = 1$ and $|babb| = 4$. A word of length n may be viewed as an ordered n-tuple of letters. It is convenient to introduce also the 0-tuple of letters, called the empty word and denoted by 1. Note that $|1| = 0$.

Given two words $x = a_1 \ldots a_n$ and $y = b_1 \ldots b_m$ we define the concatenation or product of x and y to be the word $x \cdot y = a_1 \ldots a_n b_1 \ldots b_m$. The dot is usually omitted and we write xy. It is clear that concatenation is associative, and that the empty word 1 acts as a unit, since $1x = x1 = x$ for any word x. Let A^* be the set of all words (including the empty word 1) over the alphabet A. It follows that $\langle A^*, \cdot, 1\rangle$ is a monoid under concatenation, with the empty word as the unit. This monoid of all the words over an alphabet A is called the free monoid generated by A.

If $w \in A^*$ is any word we shall denote the concatenation of n copies of w by w^n, i.e.

$$w^n = \underbrace{w \ldots w}_{n \text{ times}} .$$

If $n = 0$, we have $w^0 = 1$ for all w. Thus, for example, the infinite set of words

$$X = \{b, ab, aab, \ldots, a^n b, \ldots\},$$

with $a,b \in A$, is conveniently denoted by

$$X = \{a^n b \mid n \geq 0\},$$

i.e. it is the set of words of the form "zero or more a's followed by b".

Note the following property of the length of the product of two words:

$$|xy| = |x| + |y|. \tag{3}$$

In particular, this equation is satisfied if $x = 1$ since $1y = y$ and $|1| = 0$.

A language over an alphabet A is any subset of A^*, i.e. any set of words. Given a family of languages, we can form new languages by applyin certain operations. To begin with, we have the usual set operations such as union, intersection, symmetric difference and complement. However, othe operations arise naturally because languages are subsets of a special universal set A^*, which is a monoid. Thus we can extend the operation of concatenation from A^* to $P(A^*)$, the set of all languages over A^*, as follows: For $X, Y \subseteq A^*$,

$$XY = \{w \mid w = xy,\ x \in X,\ y \in Y\}. \tag{4}$$

It is easily verified that this multiplication of languages is associative and that the language $\{1\}$ satisfies

$$\{1\}X = X\{1\} = X, \tag{5}$$

for all $X \subseteq A^*$. Thus the system $\langle P(A^*), \cdot, \{1\}\rangle$ is again a monoid.

To illustrate multiplication of languages let $A = \{a,b\}$, and let $X = \{bab,baba\}$ and $Y = \{1,a,bb\}$ be two languages over A. Then

$$XY = \{bab,baba,babaa,babbb,bababb\}.$$

Note that a word in the product may be generated in more than one way; for example, baba = (baba)(1) = (bab)(a).

Some basic properties of concatenation are shown in Table 1. They are all easily derived from the definition of concatenation. Note that C4 and C4' are special cases of C5 and C5', respectively. The latter laws state that concatenation distributes also over infinite unions. For example, in C5 both sides denote the set of all words of the form xy, where $x \in X$ and $y \in Y_i$ for some $i \in I$.

Note that the distributive laws are in general false, if union is replaced by intersection. Thus, although C6 holds, the reverse containment

TABLE 1 Properties of Concatenation

C1 $X(YZ) = (XY)Z$

C2	$X\{1\} = X$	C2'	$\{1\}X = X$
C3	$X\phi = \phi$	C3'	$\phi X = \phi$
C4	$X(Y\cup Z) = XY \cup XZ$	C4'	$(Y\cup Z)X = YX \cup ZX$
C5	$X(\bigcup_{i\in I} Y_i) = \bigcup_{i\in I}(XY_i)$	C5'	$(\bigcup_{i\in I} Y_i)X = \bigcup_{i\in I}(Y_iX)$
C6	$X(Y\cap Z) \subseteq XY \cap XZ$	C6'	$(Y\cap Z)X \subseteq YX \cap ZX$
C7	$XY - XZ \subseteq X(Y-Z)$	C7'	$YX - ZX \subseteq (Y-Z)X$
C8	$XY \Delta XZ \subseteq X(Y\Delta Z)$	C8'	$YX \Delta ZX \subseteq (Y\Delta Z)X$

Note: We assume that concatenation has precedence over Boolean operations.

does not hold. For example, let $X = \{1,a\}$, $Y = \{aa\}$ and $Z = \{a\}$. Then $X(Y\cap Z) = \phi$, but $(XY) \cap (XZ) = \{aa\}$. Similarly, the reverse containment does not hold in C7 and C8.

We next define two closely related unary operations on languages. For $X \subseteq A^*$ the language

$$X^+ = \bigcup_{n\geq 1} X^n, \qquad (6)$$

is the subsemigroup of A^* generated by X. Thus X^+ is the set of all the words of the form $w = x_1 \ldots x_n$, $n \geq 1$, $x_i \in X$, $i = 1,\ldots,n$. Similarly, the language

$$X^* = \bigcup_{n\geq 0} X^n, \qquad (7)$$

is the submonoid of A^* generated by X. We refer to these operations as plus and star operations, respectively. Some properties of these operations are listed in Table 2.

TABLE 2 Properties of Plus and Star

P1	$X^+ = XX^*$	S1	$X^* = X^+ \cup \{1\}$
P2	$(X^+)^+ = X^+$	S2	$(X^*)^* = X^*$
P3	$\phi^+ = \phi$	S3	$\phi^* = \{1\}$
P4	$\{1\}^+ = \{1\}$	S4	$\{1\}^* = \{1\}$
P5	$X^+X = XX^+$	S5	$X^*X = XX^*$
P6	$X \subseteq Y$ implies $X^+ \subseteq Y^+$	S6	$X \subseteq Y$ implies $X^* \subseteq Y^*$

Product identities, subsemigroup, submonoid, and star identities

Product identities

C1	$X(YZ) = (XY)Z$		
C2	$X\{1\} = X$	C2'	$\{1\}X = X$
C3	$X\emptyset = \emptyset$	C3'	$\emptyset X = \emptyset$
C4	$X(Y \cup Z) = XY \cup XZ$	C4'	$(Y \cup Z)X = YX \cup ZX$
C5	$X(\bigcup_{i\in I} Y_i) = \bigcup_{i\in I} (XY_i)$	C5'	$(\bigcup_{i\in I} Y_i)X = \bigcup_{i\in I} (Y_iX)$

Plus and star identities

P1	$X^+ = XX^*$	S1	$X^* = X^+ \cup \{1\}$
P2	$(X^+)^+ = X^+$	S2	$(X^*)^* = X^*$
P3	$\emptyset^+ = \emptyset$	S3	$\emptyset^* = \{1\}$
P4	$\{1\}^+ = \{1\}$	S4	$\{1\}^* = \{1\}$
P5	$X^+X = XX^+$	S5	$X^*X = XX^*$

Monotonicity

P6	$X \subseteq Y$ implies $X^+ \subseteq Y^+$	S6	$X \subseteq Y$ implies $X^* \subseteq Y^*$

no claim of independence P4 follows P2 and P3, S4 from S2 and S3 and P5 from P1 and S5

Another simple unary operation on languages is reversal. First we define the reverse, x^ρ, of a word x by induction on $|x|$, the length of x:

$$1^\rho = 1 \text{ and } (xa)^\rho = ax^\rho.$$

Thus x^ρ is the word x read backwards. We extend this to languages as follows:

$$X^\rho = \{x^\rho | x \in X\}. \qquad (8)$$

Some properties of the reversal operation are given in Table 3.

TABLE 3 Properties of Reversal

R1	$(X \cup Y)^\rho = X^\rho \cup Y^\rho$
R2	$(X \cap Y)^\rho = X^\rho \cap Y^\rho$
R3	$(\bar{X})^\rho = (\overline{X^\rho})$
R4	$(XY)^\rho = Y^\rho X^\rho$
R5	$(X^+)^\rho = (X^\rho)^+$
R6	$(X^*)^\rho = (X^\rho)^*$

1.3 REGULAR LANGUAGES

We now restrict our attention to a particular family R of languages; this turns out to be precisely the family of languages recognizable by finite automata.

Given a finite alphabet A with #A = n, define the family L of letter languages:

$$L = \{\{a\} | a \in A\}. \qquad (9)$$

These are the n languages each consisting of a word of length 1.

Definition 1 The family R of regular languages is defined inductively as follows:

Basis: $L \subseteq R$.

Induction Step: If X and Y are in R, then so are

$$X \cup Y, \bar{X}, XY \text{ and } X^*.$$

Thus R is the smallest family of languages containing all the letter languages and closed under boolean operations, concatenation and star.

Consider some examples of regular languages. Since $\{a\} \cup \overline{\{a\}} = A^*$, we have $A^* \in R$. Next $\phi = \overline{A^*}$ and $\{1\} = \phi^*$, showing that ϕ and $\{1\}$ are in R. Let A = {a,b} for the examples below. The language

$$X = (\{a\} \cup \{b\})^*\{a\}\{a\} = A^*\{aa\},$$

is the set of all words over {a,b} that end in two a's. The language

Definition and properties of Reversal

$1^\rho = 1$, $(xa)^\rho = ax^\rho$, $a \in A$, $x \in A^*$, $X^\rho = \{x^\rho \mid x \in X\}$, $X \subseteq A^*$

R1 $(X \cup Y)^\rho = X^\rho \cup Y^\rho$

R2 $(X \cap Y)^\rho = X^\rho \cap Y^\rho$

R3 $(\neg X)^\rho = \neg(X^\rho)$

R4 $(XY)^\rho = Y^\rho X^\rho$

R5 $(X^+)^\rho = (X^\rho)^+$

R6 $(X^*)^\rho = (X^\rho)^*$

R7 $(X^\rho)^\rho = X$

Definition of Regular Languages, $\mathbb{R}$

Letter languages: $\mathbb{L} = \{\{a\} \mid a \in \mathrm{A}\}$

A nonempty finite alphabet A is assumed to be given.

1. $\mathbb{L} \subseteq \mathbb{R}$

2. If X and Y are in $\mathbb{R}$, then so are $X \cup Y$, $\neg X$, XY, and X^*

3. Nothing else is in $\mathbb{R}$, unless it is followed from a finite number of application of 1. and 2.

$$Y = (\{a\} \cup \{b\}\{a\}^*\{b\})^*$$

can be verified to consist of all the words over $\{a,b\}$ that have an even number of b's. The language

$$Z = \{a\}A^* \cap A^*\{b\} \cap \overline{A^*\{b\}\{b\}A^*}$$

is a regular language consisting of all the words over A that begin with the letter a, end with the letter b and do not contain two consecutive b's.

We next introduce a notion that will provide a characterization of regular languages.

Definition 2 Let $X \subseteq A^*$ be a language and $w \in A^*$ a word. The (left) quotient of X by w is denoted by $w\backslash X$, and is defined by

$$w\backslash X = \{y \mid wy \in X\}.$$

Similarly the right quotient of X by w is defined by

$$X/w = \{y \mid yw \in X\}.$$

The quotient $w\backslash X$ can be viewed as the set of all words which "can follow w in X". To construct $w\backslash X$, first take the set Y of all words in X that begin with w, where it is understood that w begins with w since $w = w1$. If w is then "removed" from each word wx of Y leaving the word x, the resulting set of words is precisely $w\backslash X$. In a sense this is a division of X by w on the left; hence the name left quotient.

As an illustration consider $X = \{ba, abb, aba\}$. Then $a\backslash X = \{ba, bb\}$, $aa\backslash X = \phi$, $ab\backslash X = \{a,b\}$, $ba\backslash X = \{1\}$, etc.

It will be often necessary to determine whether a given language contains the empty word. This will be done with the aid of the δ operator defined by:

$$\delta(X) = \{1\}, \text{ if } 1 \in X$$
$$\delta(X) = \phi, \text{ if } 1 \notin X \qquad (10)$$

Now the use of left quotient and the δ operator permits us to determine whether an arbitrary word w is in X, for we have

$$w \in X \text{ iff } 1 \in w\backslash X \text{ iff } \delta(w\backslash X) = \{1\}. \qquad (11)$$

Of course the right quotient can also be used for this purpose. However, we will often omit such statements about left-right symmetry, in the interest of conciseness. Also the left quotients will turn out to be more appropriate in Chapter 2. Unless otherwise stated the word quotient will mean left quotient.

Some basic properties of left quotients with respect to letters are given below.

Proposition 1 For all $a \in A$, and $X, Y \subseteq A^*$,

(a) $a\backslash\phi = \phi$; $a\backslash\{1\} = \phi$; $a\backslash\{b\} = \phi$, for all $b \in A$, $b \neq a$; $a\backslash\{a\} = \{1\}$; $a\backslash A^* = A^*$.

(b) $a\backslash(X \cup Y) = (a\backslash X) \cup (a\backslash Y)$

(c) $a\backslash(\bar{X}) = \overline{a\backslash X}$

(d) $a\backslash(XY) = (a\backslash X)Y \cup \delta(X)(a\backslash Y)$

(e) $a\backslash(X^*) = (a\backslash X)X^*$

A characterization of Regular Languages, (Left) Quotients

Definition: ***The quotient of X by w***, $w\backslash X = \{y | wy \in X\}$, $X \subseteq A^*$, $w \in A^*$
Division of X by *w* on the left.

The δ operator to check the presence of the empty word in a language.

$\delta(X) = \{1\}$ if $1 \in X$, otherwise $\delta(X) = \emptyset$, in short $\delta(X) = \{1\} \cap X$

Properties of quotients resulting from divisions by a letter *a*, $a, b \in A$, $a \neq b$, $X, Y \subseteq A^*$

(a) $a\backslash\emptyset = a\backslash\{1\} = a\backslash\{b\} = \emptyset$, $a\backslash\{a\} = \{1\}$, $a\backslash A^* = A^*$

(b) $a\backslash(X \cup Y) = (a\backslash X) \cup (a\backslash Y)$

(c) $a\backslash(\neg X) = \neg(a\backslash X)$

(d) $a\backslash(XY) = (a\backslash X)Y \cup \delta(X)(a\backslash Y)$

(e) $a\backslash(X^*) = (a\backslash X)X^*$

(f) $1\backslash X = X$

(g) for words of length > 1, $(wa)\backslash X = a\backslash(w\backslash X)$

Proof

(a) This is easily verified.

(b) $z \in a\backslash(X \cup Y)$ iff $az \in X \cup Y$ iff $az \in X$ or $az \in Y$ iff $z \in a\backslash X$ or $z \in a\backslash Y$ iff $z \in a\backslash X \cup a\backslash Y$.

(c) $z \in a\backslash(\bar{X})$ iff $az \in \bar{X}$ iff $az \notin X$ iff $z \notin a\backslash X$ iff $z \in \overline{a\backslash X}$

(d) $z \in a\backslash(XY)$ iff $az \in XY$.

If $az \in XY$, we have two cases. Either there exist $z_1, z_2 \in A^*$ such that $z = z_1z_2$, $az_1 \in X$ and $z_2 \in Y$, or $1 \in X$ and $az \in Y$. In the first case, $z_1 \in a\backslash X$, $z_2 \in Y$ and $z = z_1z_2 \in (a\backslash X)Y$. In the second case $z \in a\backslash Y$ and $\delta(X) = 1$; i.e. $z \in \delta(X)(a\backslash Y)$. Hence $a\backslash(XY) \subseteq (a\backslash X)Y \cup \delta(X)(a\backslash Y)$. The converse containment follows similarly.

(e) Note that

$$X^* = (X - \{1\})^* = \{1\} \cup (X - \{1\})(X - \{1\})^* = \{1\} \cup (X - \{1\})X^*$$

Thus $a\backslash(X^*) = (a\backslash(X-\{1\}))X^*$ since $\delta(X-\{1\}) = \phi$. Finally $a\backslash(X-\{1\}) = a\backslash X$ and (e) follows. □

Note also the following property of quotients with respect to words of length ≥ 1:

$$(wa)\backslash X = a\backslash(w\backslash X), \qquad (12)$$

for all $w \in A^*$, $a \in A$.

We will show that a language is regular iff it has a finite number of distinct quotients. However, before we do this, we will simplify our notation for regular languages by introducing certain expressions to denote languages.

1.4 REGULAR EXPRESSIONS

In view of Definition 1, the only way we have for representing a regular language X consists of specifying precisely how X is formed from L by the application of boolean operations, concatenation and star. This amounts to writing an expression for X. We formalize this as follows.

Definition 3 The set E_A of regular expressions over A is defined inductively as follows.

Basis: ϕ, 1, I, and each $a \in A$ is a regular expression.

Induction Step: If F and G are regular expressions then so are $(F \cup G)$, $\bar{F}$, (FG) and F^*. It is also convenient to include $(F \cap G)$, $(F - G)$, $(F \Delta G)$ and F^+ as regular expressions.

At this point regular expressions are strings of symbols formed by starting with some basis symbols and combining them by means of a finite number of operations as in the induction step. For example, $(((0 \cup 1^*)(00))^* \cup \bar{1})$ is a regular expression, whereas $0 \cup 1)$ is not. Note that union and concatenation are treated here as binary operators. This is somewhat awkward notationally, and will be remedied after we relate regular expressions to languages.

Definition 4 Let $L: E_A \to P(A^*)$ be the mapping defined inductively:

Basis: $L(\phi) = \phi$, $L(1) = \{1\}$, $L(I) = A^*$,

$L(a) = \{a\}$, for all $a \in A$.

Regular Expressions over A, $\mathbb{E}_A$

1. Ø, 1, I and each $a \in$ A is a regular expression.
2. If F and G are regular expressions then so are (F **U** G), ⌐F, (FG) and F*
Other Boolean operators may also be included: (F ∩ G), (F – G), (F Δ G).

The **language denoted by a regular expression** is defined recursively by the mapping

$$L: \mathbb{E}_A \rightarrow P(A^*)$$

1. L(Ø) = Ø, L(1) = {1}, L(I) = A*, L(a) = {a} for each $a \in$ A

2. L(F **U** G) = L(F) **U** L(G), L(⌐F) = ⌐L(F), L(FG) = L(F)L(G) and L(F*) = (L(F))*

Equivalence of Regular Expressions

Two regular expressions, F and G, are equivalent if they denote the same language.

$$F \equiv G \text{ iff } L(F) = L(G)$$

Induction Step: $L(F \cup G) = L(F) \cup L(G)$

$$L(\bar{F}) = \overline{L(F)}$$

$$L(FG) = L(F)L(G)$$

$$L(F^*) = (L(F))^*$$

The mapping L associates with each regular expression E a language L(E). For example, let $E = ((a \cup b)^*b)$. Then

$$L(E) = L((a \cup b)^*)L(b) = (L(a \cup b))^*L(b) =$$

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

Note that each language can be represented by many expressions. We will say that expressions F and G are equivalent iff $L(F) = L(G)$, i.e. iff they denote the same language. To simplify the notation we will denote this equivalence relation by equality, and write $F = G$ if $L(F) = L(G)$. Now all the laws applicable to languages are also extended to expressions. Thus we will write $E \cup F \cup G$ instead of $(E \cup (F \cup G))$ since union is associative for languages. Also we will write $x \in E$ instead of $x \in L(E)$, $E \supseteq F$ instead of $L(E) \supseteq L(F)$ etc. This simplifies the notation considerably. For example, for the languages X, Y and Z introduced after Definition 1, we now write:

$$X = (a \cup b)^*aa = Iaa,$$

$$Y = (a \cup ba^*b)^*,$$

$$Z = aI \cap Ib \cap \overline{IbbI}.$$

Note that the expressions ϕ, 1 and I can be omitted from the basis in Definition 3, without changing the family of languages defined. However, these expressions occur often and it is convenient to treat them as basic. Also we usually omit $F \cap G$, $F - G$, $F \Delta G$ and F^+ for their properties can be derived by noting that $F \cap G = \overline{\bar{F} \cup \bar{G}}$, $F-G = F \cap \bar{G}$, $F \Delta G = (F-G) \cup (G-F)$ and $F^+ = FF^*$. Thus $L(F \cap G) = L(F) \cap L(G)$, etc.

In summary, we may redefine a language X to be regular iff there exists a regular expression E such that $X = L(E)$.

1.5 DERIVATIVES OF REGULAR EXPRESSIONS

We now define a set of expressions derived from a given regular expression and called "derivatives" of the expression. It will be shown that derivatives correspond to the left quotients of the language denoted by the given expression and can be used to characterize regular languages.

First we need an effective method for computing $\delta(L(E))$ for a given regular expression E. We define the δ operator for regular expressions as follows:

Definition 5 If E is a regular expression, $\delta(E)$ is defined recursively as follows:

Basis: $\delta(\phi) = \phi$, $\delta(1) = 1$, $\delta(I) = 1$, $\delta(a) = \phi$ for all $a \in A$.

Induction Step: If F and G are regular expressions then

$$\begin{aligned} \delta(F \cup G) &= \delta(F) \cup \delta(G) \\ \delta(\bar{F}) &= 1 - \delta(F) \\ \delta(FG) &= \delta(F) \cap \delta(G) \\ \delta(F^*) &= 1 \end{aligned}$$

Proposition 2 $L(\delta(E)) = \delta(L(E))$.

Proof The verification of this is straightforward and is left as an exercise for the reader. □

Definition 6 The derivative of a regular expression E with respect to a letter $a \in A$ is the regular expression E_a defined as follows:

Basis: $\phi_a = \phi$, $1_a = \phi$, $a_a = 1$, $I_a = I$ and

$b_a = \phi$ for all $b \in A$, $b \neq a$.

Induction Step: If F and G are regular expressions

$$\begin{aligned} (F \cup G)_a &= F_a \cup G_a \\ (\bar{F})_a &= \overline{F_a} \\ (FG)_a &= F_aG \cup \delta(F)G_a \\ (F^*)_a &= F_aF^* \end{aligned}$$

Definition 7 The derivative of a regular expression E with respect to a word $w \in A^*$ is the regular expression E_w defined by induction on the length of w

Basis: $|w| = 0$. Then $w = 1$ and $E_1 = E$ for all E.

$|w| = 1$. Then $w \in A$ and we use Definition 6.

Induction Step: $|w| > 1$. Then $w = xa$ for some $x \in A^+$, $a \in A$. Define $E_{xa} = (E_x)_a$.

Example 1 Let $A = \{a,b\}$ and $E = (a \cup b)^*b$. Then

$$\begin{aligned} E_a &= [(a \cup b)^*]_a b \cup 1b_a \\ &= [(a \cup b)_a(a \cup b)^*]b \cup 1\phi \\ &= [(a_a \cup b_a)(a \cup b)^*]b \cup 1\phi \\ &= [(1 \cup \phi)(a \cup b)^*]b \cup 1\phi \end{aligned}$$

Since we are interested in $L(E_a)$ rather than the expression E_a itself, we will simplify E_a if possible. In this case E_a simplifies to E. Similarly,

$$\begin{aligned} E_b &= [(\phi \cup 1)(a \cup b)^*]b \cup 1 \cdot 1 \\ &= (a \cup b)^*b \cup 1 = E \cup 1. \end{aligned}$$

Derivatives of Regular Expressions

Computation of $\delta(L(E))$ for a given regular expression E.

Recall the definition of δ **for languages**
$\delta(X) = \{1\}$ if $1 \in X$, otherwise $\delta(X) = \emptyset$, in short $\delta(X) = \{1\} \cap X$

For regular expressions we define $\boldsymbol{\delta} : \mathbb{E}_{\mathbf{A}} \to \mathbb{E}_{\emptyset}$

1. $\boldsymbol{\delta}(\emptyset) = \emptyset$, $\boldsymbol{\delta}(1) = 1$, $\boldsymbol{\delta}(I) = 1$, $\boldsymbol{\delta}(a) = \emptyset$ for all $a \in A$

2. $\boldsymbol{\delta}(F \cup G) = \boldsymbol{\delta}(F) \cup \boldsymbol{\delta}(G)$, $\boldsymbol{\delta}(\neg F) = 1 - \boldsymbol{\delta}(F)$, $\boldsymbol{\delta}(FG) = \boldsymbol{\delta}(F)\boldsymbol{\delta}(G) = \boldsymbol{\delta}(F) \cap \boldsymbol{\delta}(G)$, $\boldsymbol{\delta}(F^*) = 1$

Note: $L(\boldsymbol{\delta}(E)) = \delta(L(E))$

The derivative of a regular expression E with respect to a letter $a \in$ A, E_a

$a, b \in A$, $a \neq b$, $F, G \in \mathbb{E}_{\mathbf{A}}$

1. $\emptyset_a = 1_a = b_a = \emptyset$, $a_a = 1$, $I_a = I$

2. $(F \cup G)_a = F_a \cup G_a$, $(\neg F)_a = \neg F_a$, $(FG)_a = F_a G \cup \delta(F) G_a$, $(F^*)_a = F_a F^*$

The derivative of a regular expression E with respect to a word $w \in A^*$

If $w = 1$ then $E_1 = E$, if $w \in A$ then the definition of the derivative is already covered, finally if $w \in A^+A$ then $w = xa$ where $x \in A^+$ then the derivative E_w is defined as follows:

$$E_w = E_{xa} = (E_x)_a$$

Since $A^* = \{1\} \cup A \cup A^+A$ the definition is complete for all words in A^*.

Because of the recursive correspondence between derivatives and quotients

$$\mathbf{L(E_{\mathit{w}}) = \mathit{w}\backslash L(E)}$$

Next we compute further derivatives using Definition 7.

$$E_{aa} = (E_a)_a = (E)_a = E_a = E$$
$$E_{ab} = (E_a)_b = E_b$$
$$E_{ba} = (E_b)_a = (E \cup 1)_a = E_a \cup \phi = E_a = E$$
$$E_{bb} = (E_b)_b = (E \cup 1)_b = E_b, \text{ etc.}$$

We now prove that the expression E_w denotes the quotient $w\backslash E$.

Proposition 3 For all $a \in A$, $E \in E_A$,

$$L(E_a) = a\backslash L(E).$$

Proof The claim is easily verified if E is a regular expression having no operators, i.e. one of ϕ, 1, I or $a \in A$. See Proposition 1(a) and the basis of Definition 6. Now assume that F and G are expressions having n or fewer operators and that $L(F_a) = a\backslash L(F)$ and $L(G_a) = a\backslash L(G)$. If $E = F \cup G$ we have

$$\begin{aligned} L(E_a) &= L(F_a \cup G_a) = L(F_a) \cup L(G_a) \\ &= a\backslash L(F) \cup a\backslash L(G) = a\backslash(L(F) \cup L(G)) \\ &= a\backslash L(F \cup G) = a\backslash L(E), \end{aligned}$$

where we have used Proposition 1(b).

The cases $E = \bar{F}$, $E = FG$ and $E = F^*$ are similarly verified. Hence the induction step goes through and the claim holds. □

This result is now generalized to arbitrary words.

Proposition 4 For all $w \in A^*$ and $E \in E_A$,

$$L(E_w) = w\backslash L(E).$$

Proof We proceed by induction on the length of w.

Basis: $|w| = 0$, i.e. $w = 1$. Since $E_1 = E$ and $1\backslash L(E) = L(E)$, the proposition holds.

$|w| = 1$. This is precisely Proposition 3.

Induction Step: Let $n \geq 1$ and assume that the proposition holds for all w with $|w| \leq n$. Now let $w = xa$, $a \in A$, $|x| = n$. Then

$$L(E_w) = L(E_{xa}) = L((E_x)_a) = a\backslash L(E_x)$$

by the induction hypothesis, since $|a| = 1 \leq n$. Again by the induction hypothesis, since $|x| \leq n$,

$$L(E_x) = x\backslash L(E).$$

Altogether, $L(E_w) = a\backslash(x\backslash L(E)) = xa\backslash L(E) = w\backslash L(E)$, where we have used (12). □

1.6 CONSTRUCTION OF DERIVATIVES

We will show in this section that every regular language has a finite number of distinct left quotients. If we are dealing with regular expressions we encounter the problem of recognizing when two expressions F and G denote the same language, i.e. when L(F) = L(G). In general, this is a difficult problem. For example, we have such equivalences as

$L(\overline{a^*}) = L(IbI)$ for $A = \{a,b\}$, and

$L(Ia) = (b \cup aa^*b)^*aa^*$,

which may be hard to see. On the other hand, it is sufficient to recognize only certain simple equivalences and still prove the desired result.

Definition 8 Two regular expressions E_1 and E_2 are similar (written $E_1 \sim E_2$) iff E_2 can be obtained from E_1 by using only the following rules:

$$F\phi = \phi F = \phi \qquad (13)$$

$$F \cup \phi = \phi \cup F = F \qquad (14)$$

$$F1 = 1F = F \qquad (15)$$

$$F \cup F = F \qquad (16)$$

$$F \cup G = G \cup F \qquad (17)$$

$$E \cup (F \cup G) = (E \cup F) \cup G \qquad (18)$$

where E, F and G are any regular expressions.

It is easily verified that $\sim$ is an equivalence relation on E_A and that $F \sim G$ implies $L(F) = L(G)$. On the other hand $L(F) = L(G)$ does not necessarily imply $F \sim G$, as can be seen from the examples above.

We will show that every regular expression has a finite number of dissimilar derivatives. First we need some preliminary results.

Lemma 1

For all $w \in A^*$, $(FG)_w \sim F_wG \cup \bigcup_{i \in J_w} G_i$, where J_w is a finite index set depending on w and G_i is a derivative of G for all $i \in J_w$.

Proof

We proceed by induction on $|w|$.

Basis: $|w| = 0$

$(FG)_1 = FG \sim FG = F_1G \cup \phi$

Induction Step: $|w| > 0$.

Assume the claim holds for $|w| \leq n$ for $n \geq 0$, and consider wa. Note that we use the associative law (18) for union without explicitly mentioning this, but implying it by the notation.

$$(FG)_{wa} = ((FG)_w)_a \sim (F_wG \cup \bigcup_{i \in J_w} G_i)_a \sim$$
$$(F_wG)_a \cup \bigcup_{i \in J_w} (G_i)_a \sim F_{wa}G \cup \delta(F_w)G_a \cup \bigcup_{i \in J_w} (G_i)_a.$$

Now if $\delta(F_w) = \phi$, use (13) and (14) to obtain $(FG)_{wa} \sim F_{wa}G \cup \bigcup_{i \in J_w} (G_i)_a$. Since each $(G_i)_a$ is a derivative of G, this is in the required form. If $\delta(F_w) = 1$, then $\delta(F_w)G_a \sim G_a$ by (15). Now rewrite $G_a \cup \bigcup_{i \in J_w} (G_i)_a$ as $\bigcup_{j \in J_{wa}} G_j$, where G_j is either G_a or it is one of the $(G_i)_a$. In any case, the result follows. □

Lemma 2

For all $w \in A^+$, $(F^*)_w \sim (\bigcup_{i \in J_w} F_i)F^*$, where J_w is a finite non-empty index set depending on w and F_i is a derivative of F for all $i \in J_w$.

Each regular language has a finite number of (left) quotients

The similarity equivalence

Regular expressions X, Y are **similar**, $X \sim Y$, iff Y is obtainable from X by the following rules:

$F\emptyset \equiv \emptyset F \equiv \emptyset$, $F \cup \emptyset \equiv F$, $F1 \equiv 1F \equiv F$, $F \cup F \equiv F$,
$F \cup G \equiv G \cup F$, $(E \cup F) \cup G \equiv E \cup (F \cup G)$, where $E, F, G \in \mathbb{E}_{\mathbf{A}}$

Observe the finite form of derivatives

For all $w \in A^*$ $(FG)_w \sim F_wG \cup G_1 \cup G_2 \ldots \cup G_n$, where each G_i is a derivative of G.
$(F^*)_w \sim (F_1 \cup F_2 \ldots \cup F_n)F^*$, where each F_i is a derivative of F.

$\boldsymbol{\Delta}(E)$ = the number of dissimilar derivatives of E

1. $\boldsymbol{\Delta}(\emptyset) = \boldsymbol{\Delta}(I) = 1$, $\boldsymbol{\Delta}(1) = 2$, $\boldsymbol{\Delta}(a) = 3$ for each $a \in A^*$
2. $\boldsymbol{\Delta}(F \cup G) \le \boldsymbol{\Delta}(F)\boldsymbol{\Delta}(G)$, $\boldsymbol{\Delta}(\neg F) = \boldsymbol{\Delta}(F)$, $\boldsymbol{\Delta}(FG) \le \boldsymbol{\Delta}(F)2^{\boldsymbol{\Delta}(G)}$, $\boldsymbol{\Delta}(F^*) \le 2^{\boldsymbol{\Delta}(F)}$

By induction, we see that E has a finite no. of dissimilar derivatives,

since $X \sim Y$ implies $L(X) = L(Y)$, L(E) has a finite no. of quotients.

Proof We proceed by induction on $|w|$.

Basis: $|w| = 1$, i.e. $w = a \in A$.

We have $(F^*)_a = F_a F^*$, which is of the required form, by Definition 6.

Induction Step: $|w| > 1$. Consider wa, $a \in A$.

$$(F^*)_{wa} = ((F^*)_w)_a \sim [(\bigcup_{i\in J_w} F_i)F^*]_a \sim$$

$$(\bigcup_{i\in J_w} (F_i)_a)F^* \cup \delta(\bigcup_{i\in J_w} F_i)(F^*)_a.$$

As in Lemma 1, this is easily reduced to the required form. □

Theorem 1 (Brzozowski) The number of dissimilar derivatives of any regular expression E is finite.

Proof The proof is by induction on the number n of operators in E. Let d(E) be the number of dissimilar derivatives of E.

Basis: $n = 0$.

(a) If $E = \phi$ then $E_w = \phi$ for all $w \in A^*$, i.e. $d(\phi) = 1$.

(b) If $E = I$ then $E_w = I$ for all $w \in A^*$, i.e. $d(I) = 1$.

(c) If $E = 1$ then $E_1 = 1$ and $E_w = \phi$ for all $w \in A^+$, i.e. $d(1) = 2$.

(d) If $E = a$ then $E_1 = a$, $E_a = 1$ and $E_w = \phi$ for $w \in A^*-\{1,a\}$. Thus $d(A) = 3$.

Induction Step: $n > 0$.

(e) If $E = F \cup G$, then $E_w = F_w \cup G_w$ for all $w \in A^*$. Thus $d(F \cup G) \le d(F)d(G)$.

(f) If $E = \bar{F}$, then $E_w = \overline{F_w}$. Thus $d(\bar{F}) = d(F)$.

(g) If $E = FG$, by Lemma 1

$$(FG)_w \sim F_w G \cup \bigcup_{i\in J_w} G_i.$$

By using the similarity transformations (16), (17) and (18) we can ensure that $\bigcup_{i\in J_w} G_i$ contains no repetitions, and the G_i can be arranged in some standard order. By the inductive assumption d(G) is finite, and there are at most $2^{d(G)}$ expressions of the form $\bigcup_{i\in J_w} G_i$, up to similarity. From the form of $(FG)_w$,

$$d(FG) \le d(F)2^{d(G)}.$$

(h) If $E = F^*$, by Lemma 2, for $w \in A^+$

$$(F^*)_w \sim (\bigcup_{i\in J_w} F_i)F^*,$$

where J_w is nonempty. There are $2^{d(F)}-1$ nonempty subsets of the set of derivatives of F. Adding to this the derivative with respect to 1, i.e. $(F^*)_1 = F^*$, we find

$$d(F^*) \le 2^{d(F)}.$$ □

The following are easy consequences of the theorem.

Corollary 1 If E has d(E) dissimilar derivatives they can all be found by taking derivatives with respect to words of length $\le d(E)-1$.

Proof Arrange the words of A in order of increasing length. (For example, if $A = \{a,b\}$, consider 1; a,b; aa, ab, ba, bb; aaa, aab,... etc.). Find the derivatives in that order. If for some n no new derivatives are found with respect to words of length n, then no new derivatives will ever be found and the process terminates. Thus at least one new derivative must be found for n or the process stops. In the worst case, only one derivative is found for $n = 0,1,\dots,d(E)-1$.

<u>Corollary 2</u> Every regular language has a finite number of distinct left quotients.

<u>Proof</u> This follows since similarity implies equivalence.

<u>Example 2</u> Let $I = (a \cup b)^*$ and $E = a(aI) \cap \overline{Ia}$. Then

$$\begin{aligned} E_a &= (a(aI))_a \cap (\overline{Ia})_a \\ &= [(a)_a(aI) \cup \delta(a)(aI)_a] \cap \overline{(I)_a a \cup \delta(I)(a)_a} \\ &= [1(aI) \cup \phi(aI)_a] \cap \overline{Ia \cup 11} \\ &= aI \cap \overline{Ia \cup 1}\,. \end{aligned}$$

Such details of the calculation are omitted below. We find:

$$\begin{aligned} E_1 &= E = a(aI) \cap \overline{Ia} \\ E_a &= aI \cap \overline{Ia \cup 1} \\ E_b &= \phi \cap \overline{Ia} \\ E_{aa} &= I \cap \overline{Ia \cup 1} \\ E_{ab} &= \phi \cap \overline{Ia} = E_b \\ E_{ba} &= \phi \cap \overline{Ia \cup 1} \\ E_{bb} &= \phi \cap \overline{Ia} = E_b \\ E_{aaa} &= I \cap \overline{Ia \cup 1} = E_{aa} \\ E_{aab} &= I \cap \overline{Ia} \\ E_{baa} &= \phi \cap \overline{Ia \cup 1} = E_{ba} \\ E_{bab} &= \phi \cap \overline{Ia} = E_b \\ E_{aaba} &= I \cap \overline{Ia \cup 1} = E_{aa} \\ E_{aabb} &= I \cap \overline{Ia} = E_{aab} \end{aligned}$$

Since no new derivatives are found for words of length 4, the process terminates. Using only the similarity laws of Definition 10 we failed to

recognize that $E_{ba} = E_b$, for example. In practice, it is convenient to augment the set of similarity laws by other equivalences. Using the laws $\phi \cap F = \phi$ and $I \cap F = F$, we find the following simplified computation:

$$\begin{aligned} E &= aaI \cap \overline{Ia} \\ E_a &= aI \cap \overline{Ia \cup 1} \\ E_b &= \phi = E_{ab} = E_{ba} = E_{bb} \\ E_{aa} &= \overline{Ia \cup 1} = E_{aaa} = E_{aaba} \\ E_{aab} &= \overline{Ia} = E_{aabb} \end{aligned}$$

There are 5 derivatives implying that L(E) has at most 5 distinct quotients. The question of deciding whether these quotients are indeed distinct will be discussed later.

<u>Example 3</u> Let $X = \{a^n b^n | n \geq 1\}$. We are now in a position to prove that this language is not regular. We have

$$\begin{aligned} a\backslash X &= \{a^{n-1}b^n | n \geq 1\} \\ a^2\backslash X &= \{a^{n-2}b^n | n \geq 2\} \\ &\ldots\ldots \\ a^i\backslash X &= \{a^{n-i}b^i | n \geq i\} \\ &\ldots\ldots \end{aligned}$$

It is easily seen that all these quotients are distinct, ($a^i\backslash X$ contains b^i, but no other word consisting of b's only.) By Corollary 2, X is not regular.

An example of finding all dissimilar derivatives of $\mathbf{E} = a(a\mathrm{I}) \cap \neg(\mathrm{I}a)$, where $\mathrm{L(I)} = \{a, b\}^*$

We use the lexical tree of L(I) with nodes $\mathbf{E}_w$, $w \in \mathrm{L(I)}$, where the spelling of w is the path to $\mathbf{E}_w$.
We cut off all branches emanating from $\mathbf{E}_{wx}$ if $\mathbf{E}_{wx}$ has already been encountered, $x \in \{a, b\}$.
The finiteness of the no. of dissimilar derivatives guarantees finite termination.

$$\mathbf{E_1} = \mathbf{E} = a(a\mathrm{I}) \cap \neg(\mathrm{I}a)$$

$/a$ $\qquad$ $\backslash b$

$\mathbf{E}_{a} = a\mathrm{I} \cap \neg(\mathrm{I}a \cup 1)$ $\qquad$ $\mathbf{E}_{b} = \varnothing \cap \neg(\mathrm{I}a)$

$/a$ $\qquad$ $\backslash b$ $\qquad$ $/a$ $\qquad$ $\backslash b$

$\mathbf{E}_{aa} = \mathrm{I} \cap \neg(\mathrm{I}a \cup 1)$ $\qquad$ $\mathrm{E}_{ab} = \varnothing \cap \neg(\mathrm{I}a) = \mathbf{E}_{b}$ $\qquad$ $\mathbf{E}_{ba} = \varnothing \cap \neg(\mathrm{I}a \cup 1)$ $\qquad$ $\mathrm{E}_{bb} = \varnothing \cap \neg(\mathrm{I}a) = \mathbf{E}_{b}$

$/a$ $\qquad$ $\backslash b$ $\qquad$ $/a$ $\qquad$ $\backslash b$

$\mathrm{E}_{aaa} = \mathrm{I} \cap \neg(\mathrm{I}a \cup 1) = \mathbf{E}_{aa}$ $\qquad$ $\mathbf{E}_{aab} = \mathrm{I} \cap \neg(\mathrm{I}a)$ $\qquad$ $\mathrm{E}_{baa} = \varnothing \cap \neg(\mathrm{I}a \cup 1) = \mathbf{E}_{ba}$ $\qquad$ $\mathrm{E}_{bab} = \varnothing \cap \neg(\mathrm{I}a \cup 1) = \mathbf{E}_{b}$

$/a$ $\qquad$ $\backslash b$

$\mathrm{E}_{aaba} = \mathrm{I} \cap \neg(\mathrm{I}a \cup 1) = \mathbf{E}_{aa}$ $\qquad$ $\mathrm{E}_{aabb} = \mathrm{I} \cap \neg(\mathrm{I}a) = \mathbf{E}_{aab}$

We obtain the following set of dissimilar derivatives: $\{\mathbf{E_1}, \mathbf{E}_a, \mathbf{E}_b, \mathbf{E}_{aa}, \mathbf{E}_{ba}, \mathbf{E}_{aab}\}$

Note: $\mathbf{E}_b \equiv \mathbf{E}_{ba} \equiv \varnothing$, but they are dissimilar.

1.7 DERIVATIVE EQUATIONS

A word in any language X is either empty or it must begin with a letter of the alphabet. The set of all words in X that begin with letter a is clearly $\{a\}(a\backslash X)$. Hence we have the following disjoint decomposition of any language:

$$X = \bigcup_{a\in A} \{a\}(a\backslash X) \cup \delta(X). \tag{19}$$

Each quotient in turn can be expressed in the same form:

$$w\backslash X = \bigcup_{a\in A} \{a\}(wa\backslash X) \cup \delta(w\backslash X) \quad . \tag{20}$$

If X is regular and is denoted by E, then the dissimilar derivatives of E satisfy a finite set of equations derived from (20).

Let the dissimilar derivatives of E be $E_1 = E, E_2, \dots, E_n$. Then the derivatives satisfy:

$$E_i = a_1 E_{i,1} \cup a_2 E_{i,2} \cup \dots \cup a_k E_{i,k} \cup \delta(E_i), \tag{21}$$

for $i = 1,\dots,n$, where $A = \{a_1, a_2, \dots, a_k\}$, and $E_{i,j} = (E_i)_{a_j}$.

Example 4 For $E = aaI \cap \overline{Ia}$ of Example 2, we have

$$\begin{aligned}
E &= aE_a \cup bE_b\\
E_a &= aE_{aa} \cup bE_b\\
E_b &= aE_b \cup bE_b\\
E_{aa} &= aE_{aa} \cup bE_{aab}\\
E_{aab} &= aE_{aa} \cup bE_{aab} \cup 1
\end{aligned}$$

Note that $E_b = \phi$. It is sometimes convenient to remove the empty derivative. This leads to the simplified equations:

$$\begin{aligned}
E &= aE_a\\
E_a &= aE_{aa}\\
E_{aa} &= aE_{aa} \cup bE_{aab}\\
E_{aab} &= aE_{aa} \cup bE_{aab} \cup 1
\end{aligned}$$

It is useful to represent the derivative equations in different, but equivalent, forms. We show three such forms below.

Matrix Notation

$$\begin{pmatrix} E \\ E_a \\ E_b \\ E_{aa} \\ E_{aab} \end{pmatrix} = \begin{pmatrix} \phi & a & b & \phi & \phi \\ \phi & \phi & b & a & \phi \\ \phi & \phi & a{\cup}b & \phi & \phi \\ \phi & \phi & \phi & a & b \\ \phi & \phi & \phi & a & b \end{pmatrix} \begin{pmatrix} E \\ E_a \\ E_b \\ E_{aa} \\ E_{aab} \end{pmatrix} \cup \begin{pmatrix} \phi \\ \phi \\ \phi \\ \phi \\ 1 \end{pmatrix}$$

Tabular Form

	a	b	
E	E_a	E_b	ϕ
E_a	E_{aa}	E_b	ϕ
E_b	E_b	E_b	ϕ
E_{aa}	E_{aa}	E_{aab}	ϕ
E_{aab}	E_{aa}	E_{aab}	1

Figure 1-1 Tabular representation of derivative equations

Derivative Equations

For a given alphabet $A = \{a, b, \dots, z\}$ a language $X \subseteq A^*$
is a linear combination of its letter quotients and $\delta(X)$ as follows.

$$X = \{a\}(a\backslash X) \cup \{b\}(b\backslash X) \cup \dots \cup \{z\}(z\backslash X) \cup \delta(X)$$

and each quotient $w\backslash X$ in turn is a linear combination of its letter quotients and $\delta(w\backslash X)$

$$w\backslash X = \{a\}(wa\backslash X) \cup \{b\}(wb\backslash X) \cup \dots \cup \{z\}(wz\backslash X) \cup \delta(w\backslash X)$$

If X is regular, $X = L(E)$, then the finite set of dissimilar derivatives of E, $\Delta(E)$,
$\Delta(E) = \{E, F, \dots, Q\}$, satisfies a finite system of linear equations.

$E = aE_a \cup bE_b \cup \dots \cup zE_z \cup \boldsymbol{\delta}(E)$, where $E_a, E_b, \dots, E_z \in \Delta(E)$
$F = aF_a \cup bF_b \cup \dots \cup zF_z \cup \boldsymbol{\delta}(F)$, where $F_a, F_b, \dots, F_z \in \Delta(E)$
::
$Q = aQ_a \cup bQ_b \cup \dots \cup zQ_z \cup \boldsymbol{\delta}(Q)$, where $Q_a, Q_b, \dots, Q_z \in \Delta(E)$

For $\mathbf{E} = a(aI) \cap \neg(Ia)$, we have $\Delta(\mathbf{E}) = \{\mathbf{E}, \mathbf{E}_a, \mathbf{E}_b, \mathbf{E}_{aa}, \mathbf{E}_{aab}\}$

$$\begin{aligned}
\mathbf{E} &= a\mathbf{E}_a \cup b\mathbf{E}_b \\
\mathbf{E}_a &= a\mathbf{E}_{aa} \cup b\mathbf{E}_b \\
\mathbf{E}_b &= a\mathbf{E}_b \cup b\mathbf{E}_b \\
\mathbf{E}_{aa} &= a\mathbf{E}_{aa} \cup b\mathbf{E}_{aab} \\
\mathbf{E}_{aab} &= a\mathbf{E}_{aa} \cup b\mathbf{E}_{aab} \cup 1
\end{aligned}$$

Graphical Form

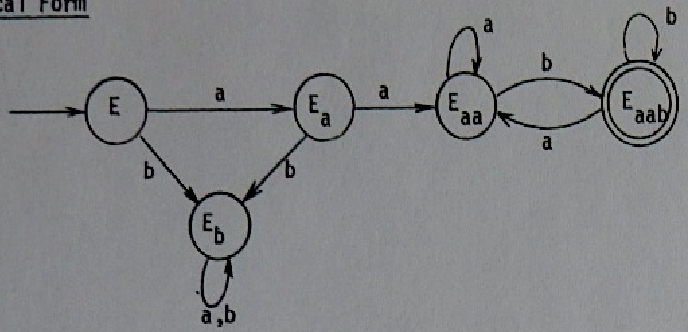

Figure 1-2 Graphical representation of derivative equations

The matrix representation is self-explanatory; here concatenation plays the role of multiplication and union plays the role of addition. The tabular form will be used in Chapter 2. In the graphical form the incoming arrow designates the given expression E whose derivatives are represented, and the double circles denote derivatives containing the empty word. We return to these representations again in Chapter 2.

Up to this point we have shown that every regular expression has a finite number of derivatives which satisfy the set of equations (21). We will now show that every such set of equations can be solved and that the solution is always a regular language.

Theorem 2 (Arden, Bodnarčuk) Let X, Y, Z be arbitrary languages over A , with the restriction $1 \notin Y$. Then the equation

$$X = YX \cup Z \tag{22}$$

has the unique solution $X = Y^*Z$.

Under the same conditions, the equation $X = XY \cup Z$ has the unique solution $X = ZY^*$.

Proof First we show that Y^*Z satisfies (22). We have

$$Y(Y^*Z) \cup Z = (YY^* \cup 1)Z = (Y^+ \cup 1)Z = Y^*Z.$$

Note that every solution of (22) contains Y^*Z .

Now suppose that $Y^*Z \cup W$ is a solution. Without loss of generality we can assume that $Y^*Z \cap W = \phi$. Then

$$Y^*Z \cup W = Y(Y^*Z \cup W) \cup Z = Y^*Z \cup YW.$$

Let w be a shortest word of W and let $|w| = r$. We must have $w \in Y^*Z \cup YW$. Also $w \notin Y^*Z$ because $Y^*Z \cap W = \phi$. Thus $w \in YW$. This is impossible, for $1 \notin Y$ and all words in YW are of length at least r+1. Hence $W = \phi$ and $X = Y^*Z$ is the unique solution. The second claim in the theorem follows analogously. □

Corollary Any set of n equations in the form (21) can be solved for all the E_i.

Proof Consider (21). If $E_{i,j} \neq E_i$ for all $j = 1,\ldots,k$, then E_i does not depend on itself and can be eliminated from the set of equations by substitution. If $E_{i,j_1} = \ldots = E_{i,j_r} = E_i$, $r \geq 1$, then (21) can be rewritten as:

$$E_i = (a_{j_1} \cup \ldots \cup a_{j_r})E_i \cup (\bigcup_{p \in J} a_p E_{i,p} \cup \delta(E_i))$$

where $p \in J$ iff $E_{i,p} \neq E_i$. We solve this equation for E_i using Theorem 2,

$$E_i = (a_{j_1} \cup \ldots \cup a_{j_r})^*(\bigcup_{p \in J} a_p E_{ip} \cup \delta(E_i)).$$

Now E_i can be eliminated from the original set of n equations by substitution.

In any case, the set of n equations can be reduced to a set of n-1 equations. In the new set of equations one easily verifies that if $E_j = YE_j \cup Z$

Arden's Theorem:

Let $X, Z \subseteq A^*$ and $Y \subseteq A^+$, then the language equation $X = YX \cup Z$ has a unique solution $X = Y^*Z$

Let A be an alphabet $A = \{a, b, \ldots, z\}$ and
let $\Delta = \{E, F, \ldots, Q\}$ be a finite set of unknown languages over A,
then the following finite system of linear equations has a unique solution vector of regular languages (E, F, . . . , Q) which can be obtained by Arden's Theorem and successive elimination of the unknowns.

$$
\begin{array}{ll}
E = aE_a \cup bE_b \cup \ldots \cup zE_z \cup \boldsymbol{\delta}(E), & \text{where } E_a,\ E_b, \ldots, E_z \in \Delta \\
F = aF_a \cup bF_b \cup \ldots \cup zF_z \cup \boldsymbol{\delta}(F), & \text{where } F_a,\ F_b, \ldots, F_z \in \Delta \\
\vdots & \\
Q = aQ_a \cup bQ_b \cup \ldots \cup zQ_z \cup \boldsymbol{\delta}(Q), & \text{where } Q_a,\ Q_b, \ldots, Q_z \in \Delta
\end{array}
$$

Example:
For $A = \{a, b\}$ $\Delta = \{\mathbf{E}, \mathbf{E}_a, \mathbf{E}_b, \mathbf{E}_{aa}, \mathbf{E}_{aab}\}$

and linear system a solution in terms of regular expressions is given below

$$
\begin{array}{ll}
\mathbf{E} = a\mathbf{E}_a \cup b\mathbf{E}_b & \mathbf{E} = aa(a \cup bb^*a)^*bb^* \\
\mathbf{E}_a = a\mathbf{E}_{aa} \cup b\mathbf{E}_b & \mathbf{E}_a = a(a \cup bb^*a)^*bb^* \\
\mathbf{E}_b = a\mathbf{E}_b \cup b\mathbf{E}_b & \mathbf{E}_b = \emptyset \\
\mathbf{E}_{aa} = a\mathbf{E}_{aa} \cup b\mathbf{E}_{aab} & \mathbf{E}_{aa} = (a \cup bb^*a)^*bb^* \\
\mathbf{E}_{aab} = a\mathbf{E}_{aa} \cup b\mathbf{E}_{aab} \cup 1 & \mathbf{E}_{aab} = b^*a(a \cup bb^*a)^*bb^* \cup b^*
\end{array}
$$

then Y satisfies the condition $1 \notin Y$. Hence the process can be repeated until we are left with one equation which can be solved by Theorem 2. □

Example 5 Consider the derivative equations from Example 4. Solving for E_b first we have

$$E_b = (a \cup b)E_b \cup \phi = (a \cup b)^*\phi = \phi.$$

Substitution of $E_b = \phi$ yields the second set of equations in Example 4. Solve next for E_{aab}:

$$\begin{aligned} E_{aab} &= (b)E_{aab} \cup (aE_{aa} \cup 1) \\ &= b^*(aE_{aa} \cup 1) = b^*aE_{aa} \cup b^*. \end{aligned}$$

Next,

$$\begin{aligned} E_{aa} &= aE_{aa} \cup b(b^*aE_{aa} \cup b^*) \\ &= (a \cup bb^*a)E_{aa} \cup bb^* \\ &= (a \cup bb^*a)^*bb^*, \\ E_a &= a(a \cup bb^*a)^*bb^*, \text{ and} \\ E &= aa(a \cup bb^*a)^*bb^*. \end{aligned}$$

In summary,

$$\begin{aligned} E &= aa(a \cup bb^*a)^*bb^* \\ E_a &= a(a \cup bb^*a)^*bb^* \\ E_b &= \phi \\ E_{aa} &= (a \cup bb^*a)^*bb^* \\ E_{aab} &= b^*a(a \cup bb^*a)^*bb^* \cup b^*. \end{aligned}$$

Notice that the expression found for E by solving the equations is different from the original expression $aaI \cap \overline{Ia}$ (see Example 2). In general, the form of the expression may depend on the order in which the derivatives are eliminated. For instance, note that

$$E_{aab} = E_{aa} \cup 1$$

in Example 4. Then

$$\begin{aligned} E_{aa} &= aE_{aa} \cup b(E_{aa} \cup 1) \\ &= (a \cup b)E_{aa} \cup b = (a \cup b)^*b = Ib, \text{ and} \\ E &= aaE_{aa} = aaIb. \end{aligned}$$

However, the language denoted by the solution is always the same.

We have shown that each regular expression E has a finite number of derivatives which satisfy the derivative equations. Conversely, the derivative equations can be solved to recover L(E), although the new expression may be different from E. This provides a converse to Corollary 2 of Theorem 1. Altogether we can now state:

Theorem 3 A language is regular iff it has a finite number of left quotients (right quotients).

Proof We need to show only that, if X has a finite number of left quotients, then it is regular. This follows because we can write a finite number of derivative equations in the form (21) for X. These can be solved for X. The solution is a regular expression, showing explicitly that X is regular.

The statement about the right quotients follows similarly, since we can solve $X = XY \cup Z$. □

Important Consequences:

A language is regular iff it has a finite number of (left) quotients.

A language X is regular iff for some positive expression E, X = L(E),
where ***positive expressions*** are regular expressions without any Boolean operators except **U**.

For regular expressions E, F

$E \equiv \emptyset$ iff no quotient of L(E) contains 1

$E \equiv I$ iff all quotient of L(E) contain 1

$E \subseteq F$ iff $E \cap \neg F \equiv \emptyset$ iff $F \cup \neg E \equiv I$

$E \equiv F$ iff $E \Delta F \equiv \emptyset$

Effective testing for equivalence and containment of regular expressions
is performed by testing of appropriate derivatives for the presence or absence of 1.

Theorem 2 and its Corollary have another important consequence as we now show.

Definition 9 The set $\mathcal{P}_A$ of positive expressions over alphabet A is defined as follows.

Basis: ϕ, 1, I and each $a \in A$ are positive expressions.

Induction Step: If F and G are positive expressions then so are $(F \cup G)$, (FG) and F^*.

Analogously, a language X is positive iff there exists a positive expression E such that $X = L(E)$. Clearly, each positive language is regular. Furthermore, we have:

Theorem 4 A language is regular iff it is positive.

Proof Let X be regular and E any regular expression such that $X = L(E)$. Construct the derivative equations for E, and solve for E. The solution involves only the operations of union, concatenation and star. Hence the solution is positive. □

As a consequence, for any regular expression E involving intersections and complements, there exists a positive expression equivalent to E.

1.8 EQUIVALENCE OF REGULAR EXPRESSIONS

We now consider the problem of deciding the equivalence of two regular expressions. The solution of this problem allows us to reduce the derivative equations to a set in which all derivatives are distinct.

Proposition 5 Let E, F and G be regular expressions. Then

a) $E = \phi$ iff no derivative of E contains 1.

b) $E = I$ iff all derivatives of E contain 1.

c) $F \supseteq G$ iff $F \cup \bar{G} = I$.

d) $F = G$ iff $F \Delta G = \phi$.

The verification of the proposition is trivial. We have now the following procedures.

Testing whether $E = \phi$ or $E = I$

Given a regular expression E, construct its derivatives. The equivalence of two derivatives may not always be recognized, but we are assured that the number of dissimilar derivatives is finite. Find $\delta(E_w)$ for each derivative E_w.

$E = \phi$ iff $\delta(E_w) = \phi$, for all $w \in A^*$,

$E = I$ iff $\delta(E_w) = 1$, for all $w \in A^*$.

The solution of the containment and equivalence problems follow immediately.

Testing whether $F \supseteq G$

To test whether $F \supseteq G$, test whether $F \cup \bar{G} = I$. This can be done by constructing the derivatives of $F \cup \bar{G}$ and checking whether 1 is present in all cases.

Testing whether $F = G$

Construct the derivatives of $F \Delta G$ and determine whether 1 is absent in all cases.

Example 6 Let $F = (a \cup b)^*b$ and $G = (a \cup ba^*b)^*ba^*b$. Does $F \supseteq G$? Let $E = F \cup \bar{G}$. The derivatives of E are:

$$
\begin{aligned}
E &= F \cup \bar{G} \\
E_a &= F \cup \bar{G} = E \\
E_b &= (F \cup 1) \cup \overline{a^*bG \cup a^*b} = F \cup \overline{a^*bG \cup a^*b} \\
E_{ba} &= F \cup \overline{a^*bG \cup a^*b} = E_b \\
E_{bb} &= (F \cup 1) \cup \overline{G \cup 1} = F \cup \bar{G} = E,
\end{aligned}
$$

where we recognize that $1 \cup \overline{G \cup 1} = 1 \cup (\bar{G}-1) = \bar{G}$. One verifies that $1 \in E$ and $1 \in E_b$. Hence $F \cup \bar{G} = I$ and $F \supseteq G$. This fact is obvious in this simple example, since F consists of all the words ending in b and all the words in G end in b. In general, the containment problem is far from obvious. Note also that $E = I$ since all derivatives of E contain 1. This means that $F \cup \overline{a^*bG \cup a^*b} = I$, which is not obvious.

<u>Example 7</u> Let $A = \{a,b\}$, $I = A^*$, $F = Ib$, $G = (a \cup bb^*a)^*bb^*$. Is $F = G$?

Let $E = F \Delta G$. The derivatives of E are:

$$
\begin{aligned}
E &= F \Delta G \\
E_a &= F \Delta G = E \\
E_b &= (F \cup 1) \Delta (b^*aG \cup b^*) \\
E_{ba} &= F \Delta G = E \\
E_{bb} &= (F \cup 1) \Delta (b^*aG \cup b^*) = E_b
\end{aligned}
$$

Neither E nor E_b contain 1. Hence $E = F \Delta G = \phi$ and $F = G$.

Since we now in a position to test whether two regular expressions are equivalent, we can test for the equivalence of the derivatives of a given expression E, and reduce the derivative equations so that each derivative expression E, and reduce the derivative equations so that each derivative corresponds to a distinct left quotient of L(E). This problem will be re-examined in more detail in Chapter 2.

PROBLEMS

1. For languages $X,Y,Z \subseteq A^*$, show that $XY - XZ \supseteq X(Y-Z)$ is false, in general. Repeat for $XY \Delta XZ \supseteq X(Y \Delta Z)$.

2. A language $X \subseteq A^*$ is said to be <u>prefix-free</u> iff $x \in X$ and $xy \in X$ implies $y = 1$; i.e. no word x is a prefix of another word y. Similarly, X is <u>suffix-free</u> iff $x,yx \in X$ implies $y = 1$. Prove the following for all $X,Y,Z \subseteq A^*$:

   (a) If X is suffix-free then

   $(Y \cap Z)X = YX \cap ZX$

   $(Y - Z)X = YX - ZX$

   $(Y \Delta Z)X = YX \Delta ZX$

   (b) If X is prefix-free then

   $X(Y \cap Z) = XY \cap XZ$

   $X(Y - Z) = XY - XZ$

   $X(Y \Delta Z) = XY \Delta XZ$

3. For any $a \in A$ and $X \subseteq A^*$ prove

   (a) $\overline{\{a\}X} = \{a\}\bar{X} \cup (A-\{a\})A^* \cup \{1\}$

   (b) $\overline{AX} = A\bar{X} \cup \{1\}$.

   (c) For all $w \in A^*$, $(w\backslash X)^\rho = X^\rho / w^\rho$.

4. Construct the derivatives of the following expressions:

   (a) $b^*a(b \cup ab^*a)^*$

   (b) $(b \cup aa^*b)^*aa\ b^*$

   (c) $a^*b \cup ba^* \cup a^*ba^*$

   (d) $a \cup bb \cup (a \cup b)^*abb$

   (e) $IaaI \cap \overline{(a \cup ba^*b)^*}$, where $I = (a \cup b)^*$

(f) $\overline{bb^*} \,\Delta\, (1 \cup b^*aI)$, $I = (a \cup b)^*$

(g) Iab^*, $I = (a \cup b \cup c)^*$

(h) b^*aI, $I = (a \cup b \cup c)^*$.

5. Write the derivative equations for the expressions of Problem 4.

6. (a) Solve the following equations for X.

$$X = Yb \cup Za \cup 1$$

$$Y = Xa \cup Ya \cup Zb$$

$$Z = Xb$$

(b) Repeat for the equations:

$$X = aY \cup bZ \cup 1$$

$$Y = aY \cup bX$$

$$Z = aX \cup bY \;.$$

(c) Show that the solutions in (a) and (b) are equal. Compare the matrix form of the derivative equations in (a) and (b).

7. Let $E = F \cup IG$ where F and G are finite languages over A, and let $x \in A^*$. Show that $E_x = H \cup IG$, where H is finite.

8. In the following equations show that $E_a \supseteq E$ and find an expression for $E_a - E$.

$$E = aE_a \cup bE_b \cup 1$$

$$E_a = aE_a \cup bE \cup 1$$

$$E_b = aE_a.$$

9. Solve each equation below for E. Show that L(E) is the same in each case.

(a) $\bar{E} = aE \cup b\bar{E} \cup 1$

(b) $E = EA \,\Delta\, Ib$, $A = \{a,b\}$

(c) $E = Ea \cup \bar{E}b$

10. Show that for $A = \{a,b\}$, $\bar{b} = (a \cup ab \cup ba \cup bb \cup bab \cup bbb)^*$.