CHAPTER 1 – Formal languages & Regular Expressions

(19 pages)

§ 1. Alphabets, strings, and operations on strings

Def: An alphabet is a finite, non-empty set V of symbols such that no string can be made up in more than one way from symbols of V. The elements of V are called characters (or letters). (For def. of string see below)

Ex. 1 (a) $V_1 = \{0,1\}$, $V_2 = \{a,b,c\}$, $V_3 = \{a\}$, $V_4 = \{\$, \#\}$ are all examples of alphabets.

(b) $V_5 = \{a, ab, b\}$, $V_6 = \{0, 11, 0110\}$, $V_7 = \{00, 000\}$ are not examples of alphabets.

(c) $V_8 = \{ab, ac\}$, $V_9 = \{cab, ba\}$, $V_{10} = \{00, 101\}$ are examples of alphabets – but strange looking ones.

Definition: A string on the alphabet V is a finite sequence of characters of V. (A string on V is also called a word on V.) The characters may be repeated in the sequence. Instead of writing a string as $\langle c_1, c_2, \ldots, c_k \rangle$, we usually write it as $c_1 c_2 \cdots c_k$ with the characters placed one after the other with no commas or spaces.

Ex. 2 (a) Let $V = \{a,b,c\}$. Then $\langle b,a,c \rangle = bac$, $\langle c,a,c \rangle = cac$, and $\langle a,a \rangle = aa$ are all examples of strings on V.

(b) The empty sequence, $\langle\ \rangle$, is also a string on any alphabet. We call $\langle\ \rangle$ the empty string and use the Greek letter lambda, λ, to denote it.

(2) <u>Notation</u> We will use lower case Greek letters such as α, β, γ to denote strings

<u>Def.</u> The length of a string φ is roughly defined to be the number of terms in the sequence φ and is denoted by $|\varphi|$

<u>Ex. 3</u> $|bac| = 3$, $|aba| = 3$, $|aa| = 2$, $|b| = 1$, $|\lambda| = 0$

<u>Def.</u> The <u>reverse</u> of a string $\varphi = \langle c_1, c_2, \ldots, c_n \rangle$ is defined by $\varphi = \langle c_n, \ldots, c_2, c_1 \rangle$. (In other words φ^R is the sequence φ in reverse order.)

The <u>concatenation</u> (or product) of the string $\alpha = \langle a_1, \ldots, a_k \rangle$ and $\beta = \langle b_1, \ldots, b_n \rangle$ is defined by

$$\alpha . \beta = \langle a_1, a_2, \ldots, a_k, b_1, b_2, \ldots, b_n \rangle .$$

(In other words $\alpha . \beta$ is the sequence of terms in α followed by the sequence of terms in β.)

<u>Def.</u> If φ is a string, we define φ^k recursively as follows: $\varphi^0 = \lambda$ and $\varphi^{k+1} = \varphi^k . \varphi$.

<u>Ex 4</u> (a) $(\langle b, a, c \rangle)^R = \langle c, a, b \rangle$, $(\lambda)^R = (\langle\ \rangle)^R = \langle\ \rangle = \lambda$

$(baa)^R = aab$, $(bab)^R = bab$

(b) $\langle b, a \rangle . \langle b, c \rangle = \langle b, a, b, c \rangle$, $\langle b \rangle . \langle a, b \rangle = \langle b, a, b \rangle$

$ba . \lambda = \langle b, a \rangle . \langle\ \rangle = \langle b, a \rangle = ba$

(c) $(ba)^0 = \lambda$, $(ba)^2 = baba$, $(ba)^3 = bababa$.

$\lambda^0 = \lambda$, $\lambda^2 = \lambda . \lambda = \lambda$, $\lambda^3 = \lambda^2 . \lambda = \lambda . \lambda = \lambda$

Note: The definitions of the reverse of a string and of the concatenation of one string with another are not very suitable to give nice proofs about results on strings. We really should define $\alpha.\beta$ recursively as follows:

$\alpha.\lambda = \alpha$ and $\alpha.(\beta.c) = (\alpha.\beta).c$ for any $c \in V$

Similarly we define the reverse of α recursively by

$\lambda^R = \lambda$ and $(\alpha.c)^R = c.(\alpha)^R$ for any $c \in V$.

Finally we define the length of α recursively by:

$|\lambda| = 0$ and $|\alpha.c| = |\alpha| + 1$ for any $c \in V$.

Using these definitions we can then nicely prove the following results by using induction

Prop. 1 Let α, β, and γ be strings on an alphabet V. Then

(a) $\alpha.\lambda = \alpha$ and $\lambda.\alpha = \alpha$

(b) $(\alpha.\beta).\gamma = \alpha.(\beta.\gamma)$

(c) $(\alpha.\beta)^R = \beta^R.\alpha^R$

(d) $(\alpha^R)^R = \alpha$

(e) $|\alpha.\beta| = |\alpha| + |\beta|$.

(f) $|\alpha^n| = n|\alpha|$ for each $n \in \mathbb{N}$

Note: In general $\alpha.\beta \neq \beta.\alpha$.

Ex. 5(a) Let $\alpha = aa$ and $\beta = baa$. Then

$\alpha.\beta = aabaa$

but $\beta.\alpha = baaaa \neq aabaa = \alpha\beta$

(b) However $\alpha.\beta = \beta.\alpha$ in many special cases. For example $\lambda.aa = aa.\lambda$, $\lambda.abb = abb.\lambda$, $aa.aaa = aaa.aa$, $ab.abab = abab.ab$.

(4)

<u>Prop. 1</u>: Let α, β, γ be strings on an alphabet V. Then

(a) $\alpha.\lambda = \alpha$ and $\lambda.\alpha = \alpha$

(b) $(\alpha.\beta).\gamma = \alpha.(\beta.\gamma)$

(c) $(\alpha.\beta)^R = \beta^R.\alpha^R$

(d) $(\alpha^R)^R = \alpha$

(e) $|\alpha.\beta| = |\alpha| + |\beta|$

(f) $|\alpha^n| = n.|\alpha|$

<u>Proof</u>:

(a) First $\alpha.\lambda = \alpha$ by definition of concatenation. [Remember concatenation was defined recursively as follows: $\alpha.\lambda = \alpha$ and $\alpha.(\beta.c) = (\alpha.\beta).c$ for each character $c \in V$.]

We will now prove that $\lambda.\alpha = \alpha$ by induction on α. For $\alpha = \lambda$, we have

$$\lambda.\alpha = \lambda.\lambda = \lambda = \alpha,$$

So the result is true for λ.

Now suppose the result is true for any α. Let c be any character in V. Then

$$\begin{aligned} \lambda.(\alpha.c) &= (\lambda.\alpha).c && \text{by def. of concatenation} \\ &= (\alpha).c && \text{because } \lambda.\alpha = \alpha \\ &= \alpha.c \end{aligned}$$

So if the result is true for α, it will be true for $\alpha.c$ for each $c \in V$. Hence by the Principle of Math Induction on strings, the result is true for any string α. Thus $\lambda.\alpha = \alpha$.

[Recall also that the reverse of α was recursively defined by: $\lambda^R = \lambda$ and $(\alpha.c)^R = c.(\alpha)^R$ for each $c \in V$.]

5.

(b) We will prove the result by parametric induction on γ. (α and β will be parameters — this means that they will be arbitrary, but fixed.) For $\gamma = \lambda$, we have

$$\begin{aligned}(\alpha.\beta).\gamma &= (\alpha.\beta).\lambda \\ &= (\alpha.\beta) && \text{because } (\alpha.\beta).\lambda = \alpha.\beta \\ &= \alpha.(\beta.\lambda) && \text{because } \beta = \beta.\lambda \\ &= \alpha.(\beta.\gamma)\end{aligned}$$

So the result is true for λ.

Now suppose that the result is true for any γ. Let c be any character in V. Then

$$\begin{aligned}(\alpha.\beta).(\gamma.c) &= ((\alpha.\beta).\gamma).c && \text{by def. of concat.} \\ &= (\alpha.(\beta.\gamma)).c && \text{bec. result was true for } \gamma \\ &= \alpha.((\beta.\gamma).c) && \text{by def. of concat.} \\ &= \alpha.(\beta.(\gamma.c)) && \text{by def of concat. again}\end{aligned}$$

So if the result is true for γ, then it will be true for $\gamma.c$ for any $c \in V$. Hence the result is true for all γ by the Principle of Math Induction on strings.

(c) We will prove the result by parametric induction on β. (α will be the arbitrary but fixed parameter.) For $\beta = \lambda$, we have

$$\begin{aligned}(\alpha.\beta)^R &= (\alpha.\lambda)^R \\ &= (\alpha)^R = \lambda.(\alpha)^R && \text{by part (a)} \\ &= \lambda^R.\alpha^R && \text{bec. } \lambda^R = \lambda \\ &= \beta^R.\alpha^R\end{aligned}$$

So the result is true for λ.

Now suppose that the result is true for any β. Let $c \in V$. Then

$$\begin{aligned}(\alpha.(\beta.c))^R &= ((\alpha.\beta).c)^R && \text{by part (b)} \\ &= c.(\alpha.\beta)^R && \text{by def. of reverse}\end{aligned}$$

(c) $= c.(\beta^R.\alpha^R)$ because result is true for β.

$= (c.\beta^R).\alpha^R$ by part (b)

$= (\beta.c)^R.\alpha^R$ by def. of reverse

So if the result is true for β, it will be true for $\beta.c$ for each $c \in V$. Hence the result is true for all β by the Principle of Math Induction for strings.

(d) We will prove the result by induction on α. For $\alpha = \lambda$, we have $(\alpha^R)^R = (\lambda^R)^R = (\lambda)^R = \lambda = \alpha$. So the result is true for λ.

Now suppose the result is true for any α. Then for any $c \in V$, we have

$((\alpha.c)^R)^R = (c.\alpha^R)^R$ by def. of $(\alpha.c)^R$

$= (\alpha^R)^R.c^R$ by part (c)

$= \alpha.c^R$ because result is true for α

$= \alpha.(\lambda.c)^R$ because $c = \lambda.c$

$= \alpha.(c.\lambda^R)$ by def of reverse

$= \alpha.(c.\lambda)$ because $\lambda^R = \lambda$

$= \alpha.c$

So if the result is true for α, then it will be true for $\alpha.c$ for any $c \in V$. Hence by the Principle of Math Induction for strings, the result is true for all α.

(e) We will prove the result by parametric induction on β. (α will be the fixed parameter). For $\beta = \lambda$, we have

$|\alpha.\beta| = |\alpha.\lambda| = |\alpha| = |\alpha| + 0$

$= |\alpha| + |\lambda| = |\alpha| + |\beta|$. So result is true for λ.

[Recall that $|\alpha|$ was defined recursively as follows. $|\lambda| = 0$ and $|\alpha.c| = |\alpha| + 1$ for each $c \in V$.]

(e) Now suppose the result is true for β. Then $|\alpha.\beta| = |\alpha| + |\beta|$.
So for each $c \in V$,

$$
\begin{aligned}
|\alpha.(\beta.c)| &= |(\alpha.\beta).c| && \text{by def. of concatenation} \\
&= |\alpha.\beta| + 1 && \text{by def. of length} \\
&= (|\alpha| + |\beta|) + 1 && \text{because result is true for } \beta \\
&= |\alpha| + (|\beta| + 1) \\
&= |\alpha| + |\beta.c| && \text{by def. of length}
\end{aligned}
$$

So if the result is true for β, then it will be true for $\beta.c$ for each $c \in V$. Hence the result is true for all strings β by the Princ. of Math. Induction on strings.

(f) We will prove the result by Mathematical induction on n. (Recall that α^n was defined recursively as follows: $\alpha^0 = \lambda$ and $\alpha^{n+1} = \alpha^n.\alpha$ for $n \geq 0$.)

For $n = 0$, then $|\alpha^n| = |\alpha^0| = |\lambda| = 0 = 0.|\alpha| = n.|\alpha|$. So the result is true for 0.
Now suppose the result is true for n. Then for any $\alpha \in V^*$, we have

$$
\begin{aligned}
|\alpha^{n+1}| &= |\alpha^n.\alpha| && \text{by definition of } \alpha^{n+1} \\
&= |\alpha^n| + |\alpha| && \text{by part (e)} \\
&= (n.|\alpha|) + |\alpha| && \text{bec. we assumed } |\alpha^n| = n.|\alpha| \\
&= (n+1).|\alpha|
\end{aligned}
$$

So if the result is true for n, it will be true for $n+1$. By the Principle of Mathematical Induction on $\mathbb{N}$, it follows that the result is true for all $\alpha \in V^*$ and all $n \in \mathbb{N}$.

§2. Languages & operations on languages

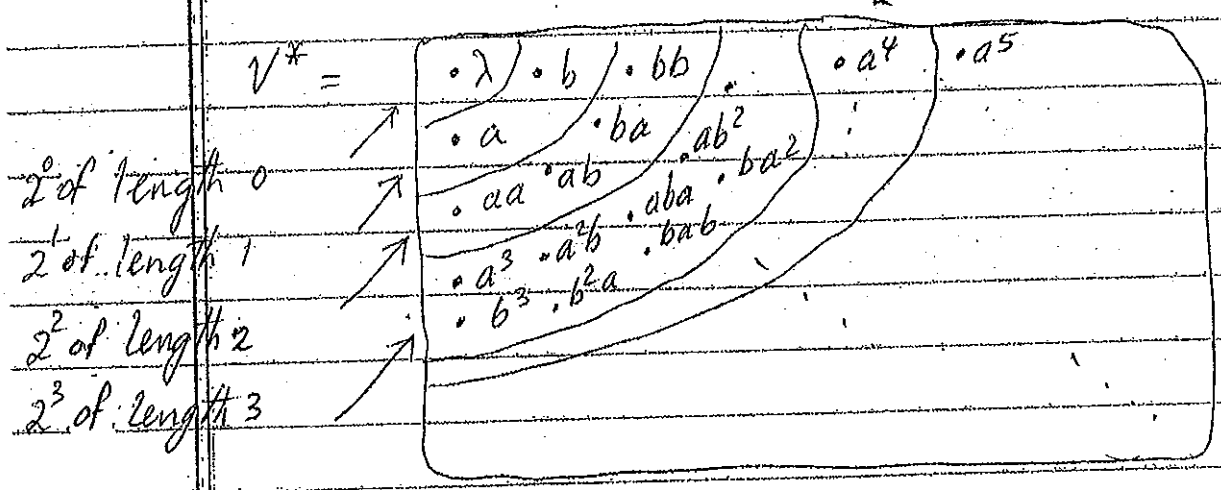
A language on V is just a set of strings on V. We will denote the set of all strings on V by V^*.

Ex.1. Let $V = \{a, b\}$. (a) Then $L_1 = \{a, ab, abb\}$, $L_2 = \{a, aa\}$, $L_3 = \{\lambda\}$, $L_4 = \{\lambda, a, b, ba, ab\}$ are all languages on V.

(b) $L_5 = \{a^n : n \geq 0\} = \{\lambda, a, aa, aaa, \ldots\}$ and $L_6 = \{a.b^{2n} : n \geq 0\} = \{a.\lambda, a.bb, a.b^4, a.b^6, \ldots\}$ are also languages on V.

Question: What is the cardinality of V^*?

Ex.2 Take $V = \{a, b\}$. Then V^* looks as shown below.



Proposition 2 Let V be any alphabet. Then V^* is denumerable and consequently any language on V is countable.

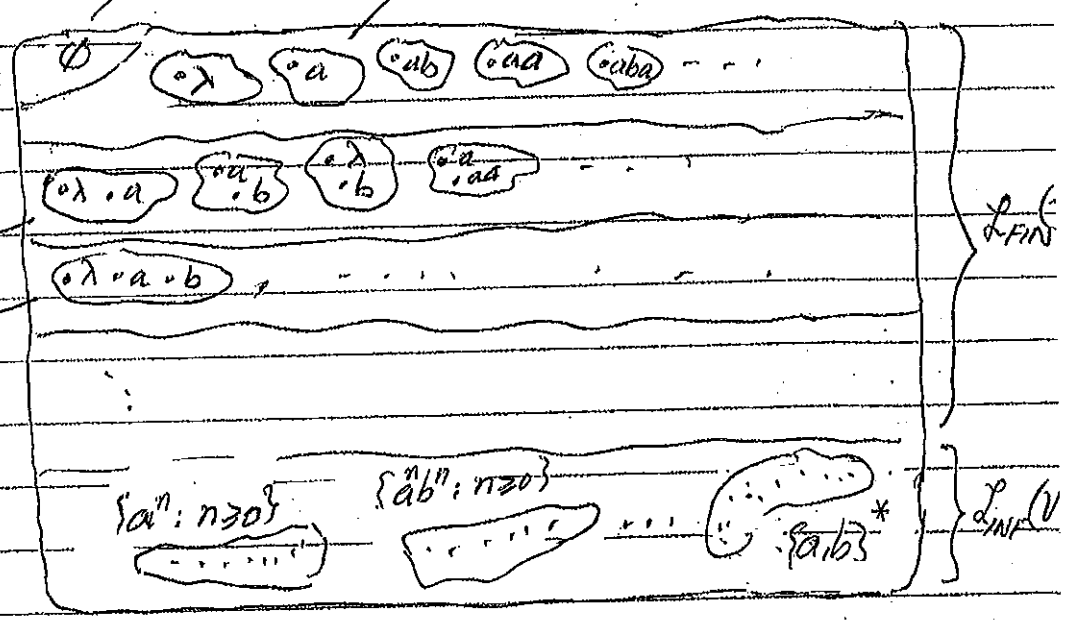
Proof: Let W_k = the set of all strings on V of length k. Then $|W_k| = (|V|)^k$ and $V^* = \bigcup_{k \in \mathbb{N}} W_k$. So V^* is a countable union of finite sets. By a standard theorem in Discrete Math (a countable union of countable sets is countable), it follows that V^* is countable. Also if c is any character

in V, then $\{c^n : n \geq 0\} \subseteq V^*$. So V^* will always be infinite. Hence V^* is infinite and countable, and thus is denumerable. Since any subset of a countable set is countable (another theorem from Discrete Math), it follows that any language on V is countable. $\square$

We will denote the set of all languages on V by $\mathcal{L}(V)$ and the set of all finite languages on V by $\mathcal{L}_{FIN}(V)$. (A language L is finite if the cardinality of $|L|$ of L is finite.)

Questions: What are the cardinalities of $\mathcal{L}(V)$ and $\mathcal{L}_{FIN}(V)$?

$V = \{a, b\}$



Prop. 3 Let V be any alphabet. Then
(a) $\mathcal{L}_{FIN}(V)$ is denumerable
(b) $\mathcal{L}(V)$ is uncountable.

Proof: (a) Let $\mathcal{L}_n(V)$ = set of all languages on V of cardinality n,

Then $\mathcal{L}_0(V) = \{\emptyset\}$ and for each $n \geq 1$, $\mathcal{L}_n(V)$ is denumerable. Also $\mathcal{L}_{FIN}(V) = \bigcup_{k \in \mathbb{N}} \mathcal{L}_n(V)$. So $\mathcal{L}_{FIN}(V)$ is denumerable.

(b) $\mathcal{L}(V)$ = set of all subsets of $V^* = \mathcal{P}(V^*)$. Since V^* is denumerable, $|\mathcal{P}(V^*)| = |\mathcal{P}(\mathbb{N})|$. But $|\mathcal{P}(\mathbb{N})| = |\mathbb{R}|$ = an uncountable set. (Here $\mathbb{R}$ is the set of real numbers.) So $\mathcal{P}(V^*)$ is is uncountable. Hence $\mathcal{L}(V)$ is uncountable.

Since a language is a set, all the operations on sets can be applied to languages. Thus if L_1 & L_2 are languages on V, then

$$L_1 \cup L_2 = \{\alpha : \alpha \in L_1 \text{ or } \alpha \in L_2\}$$
$$L_1 \cap L_2 = \{\alpha : \alpha \in L_1 \text{ and } \alpha \in L_2\}$$
$$L_1 - L_2 = \{\alpha : \alpha \in L_1 \text{ and } \alpha \notin L_2\}$$
$$L_1^c = \{\alpha \in V^* : \alpha \notin L_1\} = V^* - L_1.$$

All the theorems on sets will also be theorems about languages. So we will have

$$L_1 \cup L_2 = L_2 \cup L_1, \qquad L_1 \cap L_2 = L_2 \cap L_1$$
$$(L_1 \cup L_2)^c = L_1^c \cap L_2^c \quad \text{and so on.}$$

We also have three more operations on languages.

<u>Def.</u> The <u>reverse</u> of a language on V is defined by $L^R = \{\alpha^R : \alpha \in L\}$.

<u>Def.</u> The <u>concatenation</u> of L_1 with L_2 is defined $L_1 . L_2 = \{\alpha . \beta : \alpha \in L_1 \text{ and } \beta \in L_2\}$.

We also define L^k recursively as follows:

$$L^0 = \{\lambda\} \quad \text{and} \quad L^{k+1} = L^k . L .$$

<u>Def.</u> Finally, we define the third operation, the <u>Kleene star</u> of L by

$$L^* = \bigcup_{k \in \mathbb{N}} L^k .$$

<u>Note</u>: L^* = set of all strings that can be formed from a finite no. of strings in L

<u>Ex.3</u> Let $L_1 = \{a, ab\}$ and $L_2 = \{c, bc\}$. Then

$$L_1^R = \{(a)^R, (ab)^R\} = \{a, ba\}$$

$$L_1 . L_2 = \{a, ab\} . \{c, bc\} = \{a.c, a.bc, ab.c, ab.bc\}$$
$$= \{ac, abc, abbc\}$$

$$L_2 . L_1 = \{c, bc\} . \{a, ab\} = \{c.a, c.ab, bc.a, bc.ab\}$$
$$= \{ca, cab, bca, bcab\}$$

<u>Ex.4</u> Let $L = \{a\}$. Then $L^0 = \{\lambda\}$

$$L^1 = L^0 . L = \{\lambda\} . \{a\} = \{a\}$$
$$L^2 = L^1 . L = \{a\} . \{a\} = \{aa\}$$
$$L^3 = L^2 . L = \{aa\} . \{a\} = \{aaa\}$$
$$\vdots$$
$$L^k = L^{k-1} . L = \{a^{k-1}\} . \{a\} = \{a^k\}$$

So $L^* = \bigcup_{k \in \mathbb{N}} L^k = \bigcup_{k \in \mathbb{N}} \{a^k\} = \{a^k : k \geq 0\}$.

<u>Ex.5</u> a) Let $L = \{\lambda\}$. Then $L^* = \{\lambda\}$

b) Let $L = \emptyset$. Then $L^* = \{\lambda\}$ also.

<u>Ex.6</u> Let $L = \{a, b\}$. Then $L^0 = \{\lambda\}$, $L^1 = \{a, b\}$

$L^2 = \{aa, ab, ba, bb\}$, ... and L^k = set of all strings of a's & b's of length k

$L^* = \bigcup_{k \in \mathbb{N}} L^k$ = set of all strings of a's and b's.

<u>Proposition 4</u>: Let L_1, L_2, L_3 be languages on V. Then

a) $(L_1 \cdot L_2) \cdot L_3 = L_1 \cdot (L_2 \cdot L_3)$

b) $L_1 \cdot (L_2 \cup L_3) = (L_1 \cdot L_2) \cup (L_1 \cdot L_3)$

c) $(L_1 \cup L_2) \cdot L_3 = (L_1 \cdot L_3) \cup (L_2 \cdot L_3)$

d) $(L_1^R)^R = L_1$ (e) $(L_1 - L_2)^R = L_1^R - L_2^R$

(f) $(L_1 \cdot L_2)^R = (L_2^R) \cdot (L_1^R)$ (g) $(L_1^c)^R = (L_1^R)^c$

<u>Warning</u>: In general $L_1 \cdot L_2 \neq L_2 \cdot L_1$
and $L_1 \cdot (L_2 \cap L_3) \neq L_1 \cdot L_2 \cap L_1 \cdot L_3$

<u>Proof</u>:

(a) $(L_1 \cdot L_2) \cdot L_3 = \{\varphi \cdot \gamma : \varphi \in L_1 \cdot L_2 \text{ and } \gamma \in L_3\}$

$= \{(\alpha \cdot \beta) \cdot \gamma : \alpha \in L_1, \beta \in L_2, \text{ and } \gamma \in L_3\}$

$= \{\alpha \cdot (\beta \cdot \gamma) : \alpha \in L_1, \beta \in L_2 \text{ and } \gamma \in L_3\}$ by Prop. 3(a)

$= \{\alpha \cdot \psi : \alpha \in L_1 \text{ and } \psi \in L_2 \cdot L_3\}$

$= L_1 \cdot (L_2 \cdot L_3)$

(b) Let $\varphi \in L_1 \cdot (L_2 \cup L_3)$. Then $\varphi = \alpha \cdot \beta$ with $\alpha \in L_1$ & $\beta \in L_2 \cup L_3$. Since $\beta \in L_2 \cup L_3$, we know that $\beta \in L_2$ or $\beta \in L_3$. Now if $\beta \in L_2$, then $\varphi = \alpha \cdot \beta \in L_1 \cdot L_2$. And if $\beta \in L_3$, then $\varphi = \alpha \cdot \beta \in L_1 \cdot L_3$. So $\varphi \in L_1 \cdot L_2$ or $\varphi \in L_1 \cdot L_3$. Hence $\varphi \in (L_1 \cdot L_2) \cup (L_1 \cdot L_3)$. Since φ was arbitrary, $L_1 \cdot (L_2 \cup L_3) \subseteq (L_1 \cdot L_2) \cup (L_1 \cdot L_3)$

Now suppose $\varphi \in (L_1 \cdot L_2) \cup (L_1 \cdot L_3)$. Then $\varphi \in L_1 \cdot L_2$ or $\varphi \in L_1 \cdot L_3$. So $\varphi = \alpha \cdot \beta$ with $\alpha \in L_1$ & $\beta \in L_2$ or $\varphi = \gamma \cdot \delta$ with $\gamma \in L_1$ & $\delta \in L_3$. In the first case $\varphi = \alpha \cdot \beta \in L_1 \cdot (L_2 \cup L_3)$ because $\beta \in L_2 \cup L_3$ and in the second case $\varphi = \gamma \cdot \delta \in L_1 \cdot (L_2 \cup L_3)$ because

(13)

(b) $\delta \in L_2 \cup L_3$. So in either case $\varphi \in L_1 . (L_2 \cup L_3)$. Since φ was arbitrary $(L_1 . L_2) \cup (L_1 . L_3) \subseteq L_1 . (L_2 \cup L_3)$.
Hence $L_1 . (L_2 \cup L_3) = (L_1 . L_2) \cup (L_1 . L_3)$.

(c) Proof is very similar to that in part (b). Do for H.W.

(d) $(L_1^R)^R = \{\varphi^R : \varphi \in L_1^R\}$ bec. $L_1^R = \{\alpha^R : \alpha \in L_1\}$ (Take $L = L^R$)
$= \{\varphi^R : \varphi^R \in L_1\}$ bec. $\varphi \in L^R \Leftrightarrow \varphi^R \in L$
$= \{\psi : \psi \in L_1\} = L_1$ (Take $\psi = \varphi^R$)

(e) $\varphi \in (L_1 - L_2)^R \Leftrightarrow \varphi^R \in L_1 - L_2$
$\Leftrightarrow \varphi^R \in L_1$ and $\varphi^R \notin L_2$
$\Leftrightarrow \varphi \in L_1^R$ and $\varphi \notin L_2^R$
$\Leftrightarrow \varphi \in L_1^R - L_2^R$

(f) $(L_1 . L_2)^R = \{\varphi^R : \varphi \in L_1 . L_2\}$ let $\varphi = \alpha . \beta$ with $\alpha \in L_1$ & $\beta \in L_2$
$= \{(\alpha . \beta)^R : \alpha \in L_1 \ \& \ \beta \in L_2\}$
$= \{\beta^R . \alpha^R : \beta \in L_2 \ \& \ \alpha \in L_1\}$
$= \{\beta^R : \beta \in L_2\} . \{\alpha^R : \alpha \in L_1\}$
$= (L_2^R) . (L_1)^R$

(g) $(L_1^c)^R = (V^* - L_1)^R = (V^*)^R - L_1^R$ by part (e)
$= V^* - L_1^R = (L_1^R)^c$

<u>Note</u> (i) $\{a\} . \{ab, b\} = \{aab, ab\}$
& $\{ab, b\} . \{a\} = \{aba, ba\}$. So $L_1 . L_2 \neq L_2 . L_1$ in general

(ii) $\{a, ab\} . (\{b\} \cap \{bb\}) = \{a, ab\} . \emptyset = \emptyset$ but
$(\{a, ab\} . \{b\}) \cap \{a, ab\} . \{bb\} = \{ab, abb\} \cap \{abb, abbb\} = \{abb\}$.

(14) <u>Proposition 5</u> Let L, L_1, L_2 be languages on V. Then

a) $L^* L^* = L^*$

b) $(L^*)^* = L^*$

c) $(L^R)^* = (L^*)^R$

<u>Warning</u>: In general

(i) $(L_1 \cup L_2)^* \neq L_1^* \cup L_2^*$ (ii) $(L_1 \cap L_2)^* \neq L_1^* \cap L_2^*$

(iii) $(L_1 - L_2)^* \neq L_1^* - L_2^*$ (iv) $(L^c)^* \neq (L^*)^c$

<u>Proof</u>:

(a) $L^* = \bigcup_{k \in \mathbb{N}} L^k = \{\alpha_1 \alpha_2 \cdots \alpha_k : \alpha_i \in L \text{ and } k \geq 0\}$

So $L^* . L^* = \{\alpha_1 \cdots \alpha_k : \alpha_i \in L, k \geq 0\} . \{\beta_1 \cdots \beta_n : \beta_j \in L, n \geq 0\}$

$= \{\alpha_1 \cdots \alpha_k \beta_1 \cdots \beta_n : \alpha_i \in L, \beta_j \in L, k \& n \geq 0\}$

$\subseteq L^*$

Also $L^* . L^* = L^* . (\{\lambda\} \cup \{L\} \cup \{L^2\} \cup \cdots)$

$\supseteq L^* . \{\lambda\} = L^*$

$\therefore L^* . L^* = L^*$.

(b) $(L^*)^* = \bigcup_{k \in \mathbb{N}} (L^*)^k = (L^*)^0 \cup (L^*)^1 \cup (L^*)^2 \cup \cdots$

$= \{\lambda\} \cup L^* \cup L^* \cup L^* \cup \cdots$

$= L^*$ bec. $(L^*)^k = L^*$ for $k \geq 1$

(c) $(L^R)^* = \bigcup_{k \in \mathbb{N}} (L^R)^k = \bigcup_{k \in \mathbb{N}} (L^k)^R = \left(\bigcup_{k \in \mathbb{N}} L^k\right)^R = (L^*)^R$,

Because $(L^k)^R = (L^R)^k$ for $k \geq 0$

<u>Warning</u> (i) Take $L_1 = \{a\}$ & $L_2 = \{b\}$

(ii) Take $L_1 = \{aa\}$ & $L_2 = \{aaa\}$

(iii) Take $L_1 = \{a\}$ & $L_2 = \{aa\}$

(iv) Take any L. $\lambda \in (L^c)^*$ always

and $\lambda \notin (L^*)^c$ because $\lambda \in L^*$

§3. Regular Expressions & the languages they describe

Def A regular expression over the alphabet $V = \{c_1, \dots, c_k\}$ is a string on the auxiliary alphabet $\{\underline{c}_1, \dots, \underline{c}_k, \underline{\lambda}, \underline{\emptyset}, +, \cdot, *,), (\}$ that is defined recursively as follows:

a) Basis: $\underline{c}_1, \underline{c}_2, \dots, \underline{c}_k$, $\underline{\lambda}$ and $\underline{\emptyset}$ are regular expressions

b) Rec. Step: If E_1 and E_2 are regular expressions, then so are $(E_1 + E_2)$, $(E_1 . E_2)$ and (E_1^*).

Ex.1 Let $V = \{a, b, c\}$. Then a regular expression is a string on $\{\underline{a}, \underline{b}, \underline{c}, \underline{\lambda}, \underline{\emptyset}, +, \cdot, *,), (\}$ which follows the construction rules above

a) $(\underline{a}.((\underline{c} + (\underline{b}.\underline{a}))^*))$ is a regular expr. over V

b) $((\underline{b}.\underline{a}).(\underline{\lambda} + (\underline{c}(\underline{a})^*)))$ is a regular expr. over V.

a) $\underline{b}$ & $\underline{a}$ are regular expr., so $(\underline{b}.\underline{a})$ is a reg. expr. $\underline{c}$ & $(\underline{b}.\underline{a})$ are reg. expr., so $(\underline{c} + (\underline{b}.\underline{a}))$ is one, so $((\underline{c} + (\underline{b}.\underline{a}))^*)$ is also one. Finally $\underline{a}$ and $((\underline{c} + (\underline{b}.\underline{a}))^*)$ are reg. expr., so we get $(\underline{a} . ((\underline{c} + (\underline{b}.\underline{a}))^*))$ is a reg. expr.

To reduce the number of parentheses, we always leave out the outermost pair, specify that $*$ should be performed before "$\cdot$" and "$+$" (when they are no parentheses, specify that "$\cdot$" should be performed before "$+$", and when an operation such as "$\cdot$" or "$+$" is repeated, it is done from left to right. Using these rules the regular expressions in (a) & (b) can be simplified to:

(a') $\underline{a} \cdot (\underline{c} + (\underline{b}\underline{a})^*)$

(b') $\underline{b} \cdot \underline{a} \cdot (\underline{\lambda} + \underline{c} \cdot \underline{a}^*)$

We often leave out the dots "$\cdot$", so that (a') & (b') can be written as

(a'') $\underline{a}(\underline{c} + (\underline{b}\underline{a})^*)$

(b'') $\underline{b}\underline{a}(\lambda + \underline{c}\underline{a}^*)$

Note the alphabet on which the reg. expr. is based is totally different from the alphabet from which the characters of the reg. expression are taken. If $V = \{a, b, c\}$, then $V_{REG} = \{\underline{a}, \underline{b}, \underline{c}, \underline{\lambda}, \phi, +, \cdot, *,), (\}$. Note also that a, b, c and λ are underlined in V_{REG} but the other characters $\phi, +, \cdot, *,)$, and $($ are not.

<u>Def.</u> Let E be a regular expression over $V = \{c_1, \ldots, c_k\}$. If we interpret $\underline{c_1}, \ldots, \underline{c_k}$ and $\underline{\lambda}$ as $\{c_1\}, \ldots \{c_k\}$, & $\{\lambda\}$, "+" as "$\cup$", and ϕ, $\cdot$, and $*$ as themselves, then a regular expression will describe a language, $L(E)$, on V.

<u>Ex.</u> 1 (a) Let $E_1 = \underline{a} \cdot \underline{b}^*$. Then

$$L(E_1) = \{a\} \cdot \{b\}^* = \{a\} \cdot \{\lambda, b, bb, \ldots, b^n, \ldots\}$$
$$= \{a\} \cdot \{b^n : n \geq 0\} = \{ab^n : n \geq 0\}$$

(b) Let $E_2 = (\underline{a} \cdot \underline{b})^*$. Then

$$L(E_2) = (\{a\} \cdot \{b\})^* = \{ab\}^* = \{(ab)^n : n \geq 0\}$$
$$= \{\lambda, ab, abab, ababab, \ldots, (ab)^n, \ldots\}$$

Ex.1(c) Let $E_3 = (\underline{a}+\underline{b})^*$. Then

$$L(E_3) = (\{a\} \cup \{b\})^* = \{a,b\}^*$$

$$= \text{set of all possible strings of } a\text{'s } \& \ b\text{'s}.$$

<u>Def.</u> A language L_1 is said to be <u>regular</u> if we can find a regular expression E_1 such that $L(E_1) = L_1$.

<u>Ex.2</u> Show that the following languages are regular.

(a) $L_1 = \{a^{2n} : n \geq 0\}$

(b) $L_2 = \{\varphi \in \{a,b\}^* : |\varphi| \text{ is even}\}$

(c) $L_3 = \{a^n : n \neq 1\} = \{\lambda\} \cup \{a^n : n \geq 2\}$

(a) Let $E_1 = (\underline{aa})^*$. Then

$$L(E_1) = (\{a\}.\{a\})^* = \{aa\}^* = \{(aa)^n : n \geq 0\}$$

$$= \{a^{2n} : n \geq 0\} = L_1$$

$\therefore$ L_1 is a regular language

(b) Let $E_2 = (\underline{aa} + \underline{ab} + \underline{ba} + \underline{bb})^*$. Then

$L(E_2) = \{aa, ab, ba, bb\}^*$. So every string in $L(E_2)$ will be of even length because $L(E_2)$ = set of all strings that can be made by concatenating a finite no. of strings from $\{aa, ab, ba, bb\}$. So $L(E_2) \subseteq L_2$.

Now let φ be any string in L_2. Then

$$\varphi = \underbrace{\_\_}\underbrace{\_\_}\cdot\underbrace{\_\_}\underbrace{\_\_}\cdot \ \cdots \ \underbrace{\_\_}\underbrace{\_\_}$$

where each group of two is aa, ab, ba or bb (because $|\varphi|$ is even). So φ can be obtained from joining a finite no. of strings from $\{aa, ab, ba, bb\}$. So $L_2 \subseteq L(E_1)$. Hence $L(E_2) = L_2$.

Ex. 2 (c) We are not allowed to use "−" in a regular expression, so we cannot say that

$L_3 = L(a^* - a)$ (So this is wrong).

But if we take $E_3 = (aa + aaa)^*$, then we can see that

$$L(E_3) = \{(aa)^n.(aaa)^k : n \geq 0, k \geq 0\}$$
$$= \{a^{2n}.a^{3k} : n \geq 0, k \geq 0\}$$
$$= \{a^0\} \cup \{a^l : l \geq 2\} = L_3$$

because any number l can be written as a multiple of 2 plus a multiple of 3.

Ex. 3 Find regular expressions which describe the following languages

(a) $L_1 = \{\varphi \in \{0,1\}^* : \varphi$ contains 100 as a substring$\}$

(b) $L_2 = \{\varphi \in \{0,1\}^* : \varphi$ contains 100 or 01 as a substring$\}$

(c) $L_3 = \{\varphi \in \{0,1\}^* : \varphi$ contains both 100 & 01 as substrings$\}$

(a) $\underbrace{\ldots\ldots}_{\text{anything}}100\underbrace{\ldots\ldots}_{\text{anything}}$ Ans: $E_1 = (0+1)^*.100.(0+1)^*$.

(b) $\ldots 100 \ldots$ or $\ldots 01 \ldots$

Ans: $E_2 = (0+1)^*.100.(0+1)^* + (0+1)^*.01.(0+1)^*$.

Another answer is: $E_2' = (0+1)^*.(100+01).(0+1)^*$.

(c) $\ldots 100 \ldots 01 \ldots$, or

$\ldots 01 \ldots 100 \ldots$, or

$\ldots 1001 \ldots$, or $\ldots 0100 \ldots$.

Ans: $E_3 = (0+1)^*.100.(0+1)^*.01.(0+1)^*$
$+ (0+1)^*.01.(0+1)^*.100.(0+1)^*$
$+ (0+1)^*.1001.(0+1)^* + (0+1)^*.0100.(0+1)^*$.

It might appear that finding a regular expr. for a given language is rather easy – but this is not always so. The following examples are harder. In finding an E_i such that $L(E_i) = L_i$, you have make sure that everything E_i describes is in L_i (i.e., $L(E_i) \subseteq L_i$) and everything that is in L_i can be described by E_i (i.e., $L_i \subseteq L(E_i)$).

Ex. 4 Find regular expressions which describe the languages

(a) $L_1 = \{\varphi \in \{0,1\}^* : \varphi$ has no occurrences of $00\}$

(b) $L_2 = \{\varphi \in \{0,1\}^* : \varphi$ has exactly one occurrence of $00\}$

(c) $L_3 = \{\varphi \in \{0,1\}^* : \varphi$ has at most one occurrences of $00\}$

(d) $L_4 = \{\varphi \in \{0,1\}^* : \varphi$ has exactly two occurrences of $00\}$

(e) $L_5 = \{\varphi \in \{0,1\}^* : \varphi$ has no occurrence of $101\}$

(a) $L_1 = \underbrace{\overbrace{\cdots\cdots 1}^{\text{can end in } 0}}_{\text{no } 00\text{'s}}$ or $\underbrace{\overbrace{\cdots\cdots 0}^{\leftarrow \text{ cannot end in } 0}}_{\text{no } 00\text{'s}}$

$E_1 = (1+01)^* + (1+01)^*0$

This can be abbreviated to $(1+01)^*.(\lambda+0)$

(b) $L_2 = \underbrace{\cdots\cdots}_{\text{cannot end in } 0} 00 \underbrace{\cdots\cdots}_{\text{cannot begin with } 0}$ $\quad E_2 = (1+01)^*.00.(1+10)^*$

(c) $E_3 = \underbrace{(1+01)^*.(\lambda+0)}_{\text{no } 00\text{'s}} + \underbrace{(1+01)^*.00.(1+10)^*}_{\text{one } 00}$

(d) $L_4 = \underbrace{\cdots\cdots}_{\text{cannot end in } 0} 000 \underbrace{\cdots\cdots}_{\text{cannot begin with } 0}$ or $\underbrace{\cdots 001 \cdots 00 \cdots}_{\text{no occurrences of } 000}$

$E_4 = (1+01)^*.000.(1+10)^* + (1+01)^*.001.(1+01)^*.00.(1+10)^*$

(e) $E_5 = 0^*.(1+00+000)^*.0^*$.

END OF Ch. 1

(c') $E_3' = (1+01)^*(\lambda+00)(1+10)^*$ is almost correct – but you can't get 0.