Regular Expressions and Languages

Prof. A. Morzenti

The family of REGULAR LANGUAGES is our simplest formal language family
It can be defined in three ways:

- Algebraically (we start from this)
- By means of generative grammars
- By means of recognizer automata

a ***regular expression*** (r.e.) is a string r
over the alphabet $\Sigma=\{a_1, a_2, \ldots, a_k\}$ and the metasymbols
$\varnothing$ (empty language), $\cup$ (union), $\cdot$ (concatenation), $*$ (star)
according to the following rules (where s and t are regular expressions):

1. $r = \varnothing$ 2. $r = a,\ a \in \Sigma$ 3. $r = (s \cup t)$ or $r = s \mid t$ (alternative notation)

4. $r = (s \cdot t)$ or $r = (s\ t)$ 5. $r = (s)^*$

OPERATOR PRECEDENCE : star ‘*’, concatenation ‘.’, union ‘$\cup$’

Frequently used derived operators:

ε defined by $\varepsilon = \varnothing^*$
e^+ defined by $e \cdot e^*$

The ***meaning*** of a r.e. r is a ***language*** L_r of alphabet Σ according to the table

expression $\boldsymbol{r}$	**language** $\boldsymbol{L_r}$
$\varnothing$	$\varnothing$
ε	$\{\varepsilon\}$
$a \in \Sigma$	$\{a\}$
$s \cup t$ or $s \mid t$	$L_s \cup L_t$
$s \cdot t$ or $s\,t$	$L_s \cdot L_t$
s^*	$L_s^{\,*}$

a ***regular language*** is a language denoted by a regular expression

Example: language that consists of sequences of '1' of length multiple of three

$$e = (111)^*$$

$$L_e = \{\varepsilon, 111, 111111, \ldots.\} = \{1^n \mid n \bmod 3 = 0\}$$

$$e_1 = 11(1)^* \quad \text{NB:} \quad L_{e_1} \neq L_e$$

$$L_{e_1} = \{11, 111, 1111, 11111, \ldots.\} = \{111^n \mid n \geq 0\}$$

Example: let Σ={ +, -, d } with d denoting the decimal digits 0,1,…,9

Let us define the r.e. defining the language of integer numbers with or without sign

$$\boxed{e = (+ \cup - \cup \varepsilon)dd^*}$$

$$\boxed{L_e = \{+, -, \varepsilon\}\{d\}\{d\}^*}$$

Example: The language of alphabet $\{a, b\}$ such that
in any phrase the number of characters a is odd and there is at least one b

let us use two auxiliary r.e. : A_E strings with #a even, A_O , #a odd

$$\boxed{\begin{array}{l} e = A_E b A_O \mid A_O b A_E \text{ , where} \\ A_E = b^*(ab^*ab^*)^* \quad A_O = b^*ab^*(ab^*ab^*)^* \end{array}}$$

THE FAMILY OF REGULAR LANGUAGES (***REG***)

It is the collection of all regular languages

THE FAMILY OF FINITE LANGUAGES (***FIN***)

It is the collection of all languages having a finite cardinality

EVERY FINITE LANGUAGE IS REGULAR (hence $FIN \subseteq REG$)
because it is the union of a finite number of strings
each one being the concatention of a finite number of alphabet symbols

$$\boxed{(x_1 \cup x_2 \cup \ldots \cup x_k) = (a_{1_1} a_{1_2} \ldots a_{1_n} \cup \ldots \cup a_{k_1} a_{k_2} \ldots a_{k_m})}$$

Le family of regular languages also includes languages having infinite cardinality

hence inclusion is strict: $\boldsymbol{FIN \subset REG}$

HOW CAN ONE DERIVE FROM A R.E. THE SENTENCES OF ITS LANGUAGE?

LET US DEFINE THE ***SUBEXPRESSION*** OF A R.E.

1. Consider a r.e. with all possible parentheses
2. Derive a ***numbered*** version of the r.e.
3. Derive the numbered subexpressions

$$e = (a \cup (bb))^* (c^+ \cup (a \cup (bb)))$$

$$e_N = (a_1 \cup (b_2 b_3))^* (c_4{}^+ \cup (a_5 \cup (b_6 b_7)))$$

$$(a_1 \cup (b_2 b_3))^* \qquad c_4{}^+ \cup (a_5 \cup (b_6 b_7))$$

$$a_1 \cup (b_2 b_3) \qquad c_4{}^+ \qquad a_5 \cup (b_6 b_7)$$

$$a_1 \qquad b_2 b_3 \qquad c_4 \qquad a_5 \qquad b_6 b_7$$

$$b_2 \quad b_3 \qquad\qquad b_6 \quad b_7$$

LET US INTRODUCE THE NOTION OF **CHOICE**:

The union and repetition operators correspond to possible choices

One obtains a subexpression by making a choice that defines a sublanguage

expression r	**choice of r**
$e_1 \cup \ldots \cup e_n$ or $e_1 \mid \ldots \mid e_n$	e_k for every $1 \le k \le n$
e^*	ε, e^n for every $n \ge 1$
e^+	e^n for every $n \ge 1$

Given a r.e. one can ***derive*** another one

by replacing any subexpression with another that is a choice of it

in practice always use an «outermost» subexpression

DERIVATION RELATION among two r.e. *e'* and *e''*

$e' \Rightarrow e''$ if the two r.e. can be factorized as

$$e' = \alpha\beta\gamma \qquad e'' = \alpha\delta\gamma$$

where δ is a choice of β

NB: the definition implies that the operator (|, *, or $^+$)
of which a choice in made is «outermost»
(see remark on next slide)

the derivation relation can be applied repeatedly, yielding relation $\overset{n}{\Rightarrow}, \overset{+}{\Rightarrow}, \overset{*}{\Rightarrow}$
(ref. ***power*** and ***reflexive transitive closure*** of a relation)

$e_0 \overset{n}{\Rightarrow} e_n$ iff $e_0 \Rightarrow e_1, \quad e_1 \Rightarrow e_2, \quad \ldots, \quad e_{n-1} \Rightarrow e_n$ (e_0 derives e_n in n steps)

$e_0 \overset{+}{\Rightarrow} e_n$ e_0 derives e_n in $n \geq 1$ steps

$e_0 \overset{*}{\Rightarrow} e_n$ e_0 derives e_n in $n \geq 0$ steps

Examples Some immediate and multi-step derivations:

$$a^* \cup b^+ \Rightarrow a^*, \quad a^* \cup b^+ \Rightarrow b^+$$

$a^* \cup b^+ \Rightarrow a^* \Rightarrow \varepsilon$ that is, $a^* \cup b^+ \overset{2}{\Rightarrow} \varepsilon$ or $a^* \cup b^+ \overset{+}{\Rightarrow} \varepsilon$

$a^* \cup b^+ \Rightarrow b^+ \Rightarrow bbb$ that is, $a^* \cup b^+ \overset{2}{\Rightarrow} bbb$ or $a^* \cup b^+ \overset{+}{\Rightarrow} bbb$

Some of the derived r.e. include metasymbols (operators and parentheses)

other ones only symbols of Σ (also known as ***terminal symbols*** or ***terminals***) and ε

These constitute the ***language defined by the r.e.***

An alternative definition of the language of a r.e. similar to the one we will use for grammars

$$\boxed{L(r) = \left\{ x \in \Sigma^* \mid r \overset{*}{\Rightarrow} x \right\}}$$

NB : in derivations, operators must be chosen from external to internal

otherwise a ***premature*** choice would rule out valid sentences

e.g., $(a^* \mid bb)^* \Rightarrow (a^2 \mid bb)^*$ prevents subsequent derivation of sentence a^2bba^3

(see remark on previous slide)

Further examples:

1. $(ab)^* \Rightarrow abab$	5. $a^*(b \cup c \cup d)f^+ \Rightarrow aaa(b \cup c \cup d)f^+$
2. $(ab \cup c) \Rightarrow ab$	6. $a^*(b \cup c \cup d)f^+ \Rightarrow a^*cf^+$
3. $a(ba \cup c)^*d \Rightarrow ad$	7. $a^*(b \cup c \cup d)f^+ \stackrel{+}{\Rightarrow} aaacf^+$ in 2 steps
4. $a(ba \cup c)^*d \Rightarrow a(ba \cup c)(ba \cup c)d$	8. $a^*(b \cup c \cup d)f^+ \stackrel{+}{\Rightarrow} aaacff$ in 3 steps

Two r.e. are ***equivalent*** if they define the same language

a phrase of a regular language can be obtained through distinct equivalent derivations

these can ***differ in the order*** of the choices used on the derivation

$a(ba \cup c)^*d \Rightarrow a(ba \cup c)(ba \cup c)d \Rightarrow ac(ba \cup c)d \Rightarrow acbad$

$a(ba \cup c)^*d \Rightarrow a(ba \cup c)(ba \cup c)d \Rightarrow a(ba \cup c)bad \Rightarrow acbad$

AMBIGUITY OF REGULAR EXPRESSIONS

a phrase may be obtained through distinct derivations, which **differ not only in the order**

$$
\boxed{
\begin{array}{l}
(a \bigcup b)^* a(a \bigcup b)^* \\
(a \bigcup b)^* a(a \bigcup b)^* \Rightarrow (a \bigcup b)a(a \bigcup b)^* \Rightarrow aa(a \bigcup b)^* \Rightarrow aa\varepsilon \Rightarrow aa \\
(a \bigcup b)^* a(a \bigcup b)^* \Rightarrow \varepsilon a(a \bigcup b)^* \Rightarrow \varepsilon a(a \bigcup b) \Rightarrow \varepsilon aa \Rightarrow aa
\end{array}
}
$$

sufficient condition for ambiguity :

a r.e. f is ambiguous if the language of the numbered version f'

includes two distinct strings x and y that coincide when numbers are erased

Example

$f = (a \cup b)^* a (a \cup b)^*$

$f' = (a_1 \cup b_2)^* a_3 (a_4 \cup b_5)^*$ is a r.e. of alphabet $\Sigma' = \{a_1, b_2, a_3, a_4, b_5\}$

a_1a_3 and a_3a_4 prove (witness) the ambiguity of the r.e. $f = (a \cup b)^* a (a \cup b)^*$

Example (ambiguity)

$(aa \mid ba)^* \, a \mid b(aa|b)^*$ is ambiguous

numbered version: $(a_1a_2 \mid b_3a_4)^* \, a_5 \mid b_6(a_7a_8|b_9)^*$

from which one can derive $b_3a_4a_5$ and $b_6a_7a_8$

both mapped to the string *baa* by erasing the subscript

PAY ATTENTION: ambiguity is often a source of problems

APPLICATION of r.e. and ambiguity: specify floating point numbers with or without sign and exponent

$$\Sigma = \{+, -, \bullet, E, d\}$$

$$r = s.c.e$$

$s = (+ \cup - \cup \varepsilon)$ provides the optional $\pm$ sign

$c = (d^+ \bullet d^* \cup d^* \bullet d^+)$ generates integer or fractional constants with no sign

$e = (\varepsilon \cup E(+ \cup - \cup \varepsilon) d^+)$ generates the optional exponent preceded by E

$$(+ \cup - \cup \varepsilon)(d^+ \bullet d^* \cup d^* \bullet d^+)(\varepsilon \cup E(+ \cup - \cup \varepsilon) d^+)$$

$+dd \bullet E - ddd$ $\quad +12 \bullet E - 341$ represents the number $12.0 \cdot 10^{-341}$

NB: r.e. for numbers with integer and fractional part is ambiguous: why?
because the two r.e. $d^+ \bullet d^*$ and $d^* \bullet d^+$ define non-disjointed languages
REMEDY?
Typical remedy: divide the language in three disjointed parts, each one modelled by a distinct e.r.

Do this as an exercise ;-)

EXTENDED REGULAR EXPRESSIONS

Extended with other operators

POWER: $a^h = aa \ldots a$ (h times): a^n

REPETITION: from k to $n > k$: $[a]_k^n = a^k \cup a^{k+1} \cup \ldots a^n$

OPTIONALITY: $(\varepsilon \cup a)$ or $[\, a \,]$

ORDERED INTERVAL: $(0 \ldots 9)$ $(a \ldots z)$ $(A \ldots Z)$

Set theoretic operators: INTERSECTION, DIFFERENCE, COMPLEMENT

It can be shown (studying the relation with finite automata) that set theoretic operations do ***not*** increase the expressive power of r.e.

(they are only useful abbreviations)

INTERSECTION: useful to define languages through conjunction of conditions
EXAMPLE: the language $L \subset \{a, b\}^*$ of even-length strings which contain *bb*

Easy to define using a r.e. with intersection :

$$e = \underbrace{(\ (a \mid b)^*\ bb\ (a \mid b)^*\ )}_{\text{phrases including } bb} \cap \underbrace{(\ (a \mid b)^2\ )^*}_{\text{even-length phrases}}$$

Without intersection:

***bb* surrounded by two even- or two odd-length strings**

$$\boxed{((a \mid b)^2)^* bb((a \mid b)^2)^* \mid (a \mid b)((a \mid b)^2)^* bb(a \mid b)((a \mid b)^2)^*}$$

Example of extended r.e. with complement operator

Language $L \subset \{a,b\}^*$ of strings ***not*** containing substring aa

Easy to define its complement: $\neg L = \{ x \in (a \mid b)^* \mid x$ contains substring $aa \}$

$$\neg L = ((a \mid b)^* \, aa \, (a \mid b)^*)$$

Therefore L can be defined by a r.e. extended with complement

$$L = \neg((a \mid b)^* \, aa \, (a \mid b)^*)$$

Definition by a r.e. non-extended (*subjectively* less readable)

$$L = \underbrace{((ab) \mid b)}_{\text{of two consecutive symbols, at least one is a } b}{}^* \, \underbrace{(a \mid \varepsilon)}_{\text{possible final } a}$$

CLOSURE PROPERTIES OF THE *REG* FAMILY (family of regular languages)

Let *op* be a unary or binary language operator (e.g., complement, concatenation, etc.)

a family of languages is closed w.r.t. *op* iff …
every language obtained by applying *op* to languages of the family is also in the family

property: the *REG* family is closed w.r.t.

concatenation, union, star

(and hence also w.r.t. the derived operators of cross '+' and power)

it is an obvious consequence of the very definition of regular expression

Therefore regular languages can be combined by these operators without exiting *REG*
(i.e., obtaining languages that are still regular)

REG is also closed w.r.t. INTERSECTION and COMPLEMENT

(we will use finite automata to show that)

APPLICATION: REPRESENTATION OF LISTS BY MEANS OF R.E.

a list contains an unspecified number of elements e of the same type

generated by the r.e. e^+, or e^* if it can be empty

e can be a terminal symbol or any regular subexpression

LISTS WITH SEPARATORS AND OPENING AND CLOSING MARKS

$$\boxed{ie(se)^* f \qquad i[e(se)^*]f}$$

Examples from programming lang.

$$\overbrace{begin}^{i}\,\overbrace{istr_1}^{e}\overbrace{;}^{s}\overbrace{istr_2}^{e}\overbrace{;}^{s}\ldots\overbrace{;}^{s}\overbrace{istr_n}^{e}\,\overbrace{end}^{f}$$

$$\overbrace{procedure\ PRINT(}^{i}\,\overbrace{par_1}^{e}\overbrace{,}^{s}\overbrace{par_2}^{e}\overbrace{,}^{s}\ldots\overbrace{,}^{s}\overbrace{par_n}^{e}\overbrace{)}^{f}$$

$$\overbrace{array\ MATRIX\ '['}^{i}\,\overbrace{\mathrm{int}_1}^{e}\overbrace{,}^{s}\overbrace{\mathrm{int}_2}^{e}\overbrace{,}^{s}\ldots\overbrace{,}^{s}\overbrace{\mathrm{int}_n}^{e}\overbrace{']'}^{f}$$

LISTS WITH PRECEDENCE OR LEVELS

An element in a list can be a list of a lower level

NB: the list can be represented by a r.e. only if the ***number of levels*** is ***limited***

otherwise more powerful notations are needed (grammars)

$list_1 = i_1 \ \ list_2 \ \ (s_1 \ \ list_2)^* \ \ f_1$

$list_2 = i_2 \ \ list_3 \ \ (s_2 \ \ list_3)^* \ \ f_2$

...

$list_{\mathrm{k}} = i_{\mathrm{k}} \ \ e_{\mathrm{k}} \ \ (s_{\mathrm{k}} \ \ e_{\mathrm{k}})^* \ \ f_{\mathrm{k}}$

Examples from progr. lang.

level 1: *begin* $instr_1$; $instr_2$; ... $instr_n$ *end*
level 2: *WRITE* (var_1, var_2, ... var_n)

some arithmetic expressions can be viewed as lists (e.g., sums of terms)

$3 + 5 \times 7 \times 4 - \ 8 \times 2 \ \div 5 + 8 + 3$