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 grammars
- By means of recognizer automata

a ***regular expression*** (r.e.) is a ***language*** espression built by applying the operators

union	‘|’ (also denoted as ‘$\cup$’)
concatenation	‘$\cdot$’ (NB: ‘$\cdot$’ is often omitted, i.e., left understood)
Kleen star	‘*’
Cross	‘+’ (NB: this is derived , i.e., not fundamental : $e^+ = e \cdot e^*$)

starting from the «building blocks»

any $a \in \Sigma$	(Σ is the alphabet of the language)
$\varnothing$	empty language
ε	(NB: this is derived: $\varepsilon = \varnothing^*$)

OPERATOR PRECEDENCE : star ‘*’, concatenation ‘.’, union ‘|’

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

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

$$e = (111)^*$$

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

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

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

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

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

$$\boxed{e = (+ \mid - \mid \varepsilon) d d^*}$$

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

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

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 INTRODUCE THE NOTION OF **CHOICE**:

The union and repetition operators correspond to possible choices

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

expression *r* of type	**choices of *r***
$e_1 \mid \ldots \mid e_n$	e_k for every $1 \leq k \leq n$
e^*	ε or e^n for every $n \geq 1$
e^+	e^n for every $n \geq 1$

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

by replacing any «top level» subexpression with another that is a choice of it

DERIVATION RELATION among two r.e. e' and e'': $e' \Rightarrow e''$

$\alpha\beta\gamma \Rightarrow \alpha\delta\gamma$ (NB: α and γ may be missing)

where δ is a choice of β

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

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

$e_0 \overset{n}{\Rightarrow} e_n$ iff $e_0 \Rightarrow e_1,\ \ e_1 \Rightarrow e_2,\ \ \ldots,\ \ e_{n-1} \Rightarrow e_n$

e_0 derives e_n (or e_n is derived from e_0) 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:

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

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

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

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.***

A 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 : an operator chosen for a derivation step must be «top level»

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

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

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

these can ***differ in the order*** of the choices used in the derivation (next we underline the subexpression chosen for the derivation step)

$a\underline{(ba \mid c)^*}d \Rightarrow a\underline{(ba \mid c)}(ba \mid c)d \Rightarrow ac\underline{(ba \mid c)}d \Rightarrow acbad$

$a\underline{(ba \mid c)^*}d \Rightarrow a(ba \mid c)\underline{(ba \mid c)}d \Rightarrow a\underline{(ba \mid c)}bad \Rightarrow acbad$

AMBIGUITY OF REGULAR EXPRESSIONS

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

$$(a \mid b)^* a (a \mid b)^*$$

$$(a \mid b)^* a (a \mid b)^* \Rightarrow (a \mid b) a (a \mid b)^* \Rightarrow aa (a \mid b)^* \Rightarrow aa\varepsilon \Rightarrow aa$$

$$(a \mid b)^* a (a \mid b)^* \Rightarrow \varepsilon a (a \mid b)^* \Rightarrow \varepsilon a (a \mid b) \Rightarrow \varepsilon aa \Rightarrow aa$$

here a terminal symbol can derive from distinct subexpressions of the original r.e.

This is the notion of ambiguity. To define precisely ambiguity …

… we introduce the numbered version e_N of a reg.exp. e

Example: $\boldsymbol{e = (a \mid (bb))^* (c^+ \mid (a \mid (bb)))}$

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

in e, just add a distinct integer subscript to all occurrences of symbols $a \in \Sigma$
(typically, for simplicity, but not necessarily, from 1 up, from left to right)

sufficient condition for ambiguity :

a r.e. e is ambiguous if the language of its numbered version e_N

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

Example

$e = (a \mid b)^* \, a \, (a \mid b)^*$

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

$a_1 a_3$ and $a_3 a_4$ prove (witness) the ambiguity of the r.e.

Another 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

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

$$r = s.c.e$$

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

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

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

$$(+ \mid - \mid \varepsilon)(d^{+} \bullet d^{*} \mid d^{*} \bullet d^{+})(\varepsilon \mid E(+ \mid - \mid \varepsilon)d^{+})$$

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

integer and fractional parts can be empty, but not both

NB: the above r.e. is ambiguous for numbers having both integer and fractional parts: 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 modeled 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: $[\ a\ ]$ equivalent to $(\varepsilon \mid a)$

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

Set theoretic operators: INTERSECTION, DIFFERENCE, COMPLEMENT

It can be shown (by 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: $\neg L = \{\, x \in (a \mid b)^* \mid x \text{ contains substring } aa \,\}$

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

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 under *op* iff …
every language obtained by applying *op* to languages of the family is also in the family

property: the *REG* family is closed under

concatenation, union, star

(and hence also under the derived operators of cross '+' and power)

this 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 under INTERSECTION and COMPLEMENT

(this is not so obvious; we will use finite automata to show that)

APPLICATION OF R.E.: REPRESENTATION OF LISTS

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 languages

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

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

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

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 ***bounded***

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: $begin\ instr_1;\ instr_2;\ \ldots\ instr_n\ \ end$
level 2: $WRITE\ (var_1,\ var_2,\ \ldots\ var_n\ )$

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

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