

–contents of Regular Expressions–

Rationale  
Definitions

–requires–

Notation Supporting Grammars

## Regular Expressions

Regular expressions serve the same purpose as CFGs: they syntactically define strings of symbols. One advantage is conciseness and ease of implementation. One disadvantage is that they can describe fewer things than CFGs. Three issues favor a discussion of regular expressions.

The first issue is that regular expression expressiveness can be mixed in with CFGs. The new grammars are called regular expression grammars (REGs). See the discussion on Regular Expression Grammars. The second issue is that REGs are an essential step in building recursive parsers. See the discussion on Top-down Parsers. The third issue is a close relation between regular expressions and finite automata. Finite automata, in turn, form the basis for automatic parsers.

**Note:** This is *not* a discussion about the glob form of regular expressions used in command-line utilities such as

```
ls *. [ch]*
```

### Definitions

A regular expression is a combination of input symbols ( $V_I$ ) and meta-symbols. The meta-symbols are the following meta-operators:

- () meta-parentheses
- regular expression ‘catenate’
- | regular expression ‘or’
- & regular expression ‘and’
- regular expression difference
- \* zero or more repetitions
- + one or more repetitions
- ? zero or one (optional item)
- $n$  exactly  $n$  repetitions
- ~ regular expression complement

Table 1: Regular Expression Operators

When the meaning is clear from context, the catenation operator  $a \circ b$  is often implied by the juxtaposition  $ab$ . Applied to sets  $A \circ B$  means the catenation of all pairs, the first taken from  $A$  and the second from  $B$ .

$\mathcal{L}(\lambda)$	$\stackrel{\text{def}}{=}$	$\{\lambda\}$
$\mathcal{L}(a)$	$\stackrel{\text{def}}{=}$	$\{a\}$
$\mathcal{L}(A)$	$\stackrel{\text{def}}{=}$	$\mathcal{L}(A)$
$\mathcal{L}(AB)$	$\stackrel{\text{def}}{=}$	$\mathcal{L}(A) \circ \mathcal{L}(B)$
$\mathcal{L}(A B)$	$\stackrel{\text{def}}{=}$	$\mathcal{L}(A) \cup \mathcal{L}(B)$
$\mathcal{L}(A\&B)$	$\stackrel{\text{def}}{=}$	$\mathcal{L}(A) \cap \mathcal{L}(B)$
$\mathcal{L}(A - B)$	$\stackrel{\text{def}}{=}$	$\mathcal{L}(A) - \mathcal{L}(B)$
$\mathcal{L}(\sim A)$	$\stackrel{\text{def}}{=}$	$V_I^* - \mathcal{L}(A)$
$\mathcal{L}(A^0)$	$\stackrel{\text{def}}{=}$	$\{\lambda\}$
$\mathcal{L}(A^i)$	$\stackrel{\text{def}}{=}$	$\mathcal{L}(A^{i-1}A)$
$\mathcal{L}(A_j^k)$	$\stackrel{\text{def}}{=}$	$\bigcup_{i=j}^k \mathcal{L}(A^i)$
$\mathcal{L}(A^*)$	$\stackrel{\text{def}}{=}$	$\mathcal{L}(A_0^\infty)$
$\mathcal{L}(A^+)$	$\stackrel{\text{def}}{=}$	$\mathcal{L}(A_1^\infty)$
$\mathcal{L}(A^?)$	$\stackrel{\text{def}}{=}$	$\mathcal{L}(A_0^1)$

Table 2: Regular Expression Languages

Let  $a \in V_I$  and  $A, B$  designate regular expressions. Given a regular expression  $A$ , the set of strings,  $\mathcal{L}(A)$ , defined by it can be built up by repetitive application of the rules in Table 2.

The regular expression operators  $\&$ ,  $\sim$  and  $-$  are useful for describing things but are harder to implement for recognizing things, so they are often omitted.

### Exercises

1. What are  $\mathcal{L}(\sim())$ ,  $\mathcal{L}(a|b)$ ,  $\mathcal{L}(a|a)$ ,  $\mathcal{L}(a\&b)$ ,  $\mathcal{L}(a\&a)$ ,  $\mathcal{L}(a^*)$ ,  $\mathcal{L}(a^+)$ ,  $\mathcal{L}(a^9)$ ,  $\mathcal{L}(a_3^9)$ , and  $\mathcal{L}(cr^?a(zy|t))$ ?
2. Using Letter, Digit and Any, describe the following C programming language constructs as regular expressions.
  - (a) identifier
  - (b) real number literal (e.g. 0.314E+1)
  - (c) integer constant (e.g 123)
  - (d) all X reserved words (e.g. if, fi...)
  - (e) X comment
  - (f) C comment (hard)
3. The body of a telegram is a sequence of words. It is also a sequence of lines of at most 80 characters. Write a REG using  $\&$ , and phrase names tele, line, word, alnum and ch that helps define the telegram layout.