Matching patterns - regular expressions

- Finding patterns in text/files is a very common requirement.
- Patterns are specified by **regular expressions**. A regular expression (RE) is a simple finite representation for a (possibly infinite) set of strings.
- Let R be a RE and S_R be the set of strings that correspond to R. If s is a string then R **matches** s if $s \in S_R$ - that is s is a string in the set S_R.

Specifying regular expressions

First let us define REs more precisely then we will see how one particular tool called **egrep** realizes REs. Most programming languages have libraries that implement REs and the syntax is very similar to egrep. Let $\mathcal{A}$ be any alphabet - that is set of characters. Then REs are defined by the following:

- The empty string ϵ is an RE.
- Any $c \in \mathcal{A}$ is an RE. That is any character c in $\mathcal{A}$ represents the set $\{c\}$.
- If r_1, r_2 are REs then the following expressions are REs (where '(' and ')' are meta symbols for grouping):
  a) $r_1 r_2$ - concatenation.
  b) $(r_1 + r_2)$ -union of r_1 and r_2. That is set of strings represented by r_1 or r_2.
  c) $r_1^\star$ - 0 or more repetitions of r_1, called Kleene closure.
  d) r_1^+ - 1 or more repetitions of r_1.
  e) $r_1?$ - 0 or 1 occurrence of r_1.

Note that just the first 3 clauses are enough but the last two clauses are often needed and so it is better to have explicit syntax for them.

grep/egrep - 1: Character class, bracket expression

`grep -options RE file`

will print all lines in the file where there the RE matches. While matching (e)grep removes the newline character.

- All characters are REs; '.' stands for any character; the escape character is '\'.
- Bracket expression: $[c_1 c_2 \ldots c_m]$ will match any character $c_i, i = 1..m$; $[\hat{c}_1 c_2 \ldots c_m]$ will match characters other than $c_i, i = 1..m$; $[a - g]$ called range expression represents $[abcdefg]$ same is true for capital letters and digits - so $[0 - 9]$ stands for [0123456789] - meta characters lose their special meaning bracket or range expressions.
- The following named character classes are useful [:alnum:], [:alpha:], [:cntrl:], [:digit:], [:graph:], [:lower:], [:print:], [:punct:], [:space:], [:upper:], and [:xdigit:]. Their meanings are clear from the names, [:graph:] is set of visible characters, [:xdigit:] is hexadecimal digit characters.
  Named character classes may not be available in all flavours of REs.

grep/egrep-2: Alternation, repetition

Again let r, r_1, r_2 be REs then the following are also REs where the meta-characters act as alternation or quantification operators. Parentheses '(' and ')' act as meta characters for grouping.

Operator	**Example**	**Meaning**
$\vert$	$r_1 \vert r_2$	occurrence of r_1 or r_2
$\star$	$r^\star$	0 or more occurrences of r
$+$	$r+$	1 or more occurrences of r
?	$r?$	0 or 1 occurrence
$\{m\}$	$r\{m\}$	Exactly m occurrences.
$\{m,\}$	$r\{m,\}$	m or more occurrences.
$\{m,n\}$	$r\{m,n\}$	At least m occurrences and at most n occurrences
$\{,m\}$	$r\{,m\}$	At most m occurrences. Only GNU grep.

Start, end and word markers

Operator	Example	Meaning
^	^(r)	Match r only at the beginning of a line.
$	(r)$	Match r only at the end of a line.
\<		Position at the beginning of a word, where word=alphanumeric sequence.
\>		Position at the end of a word, where word=alphanumeric sequence.

The beginning and end of word meta sequences may not be available in all versions of (e)grep.

Repeating earlier matches

- There is a limited facility for matching a previously matched sub-REs.
- This is possible only for parenthesized sub-REs.
- The syntax is: $\backslash n$, where $n = 1..9$. So, $\backslash 1$ will match the first parenthesized sub-RE.

Precedence of operators

- Repetition precedes concatenation which precedes alternation. Or repetition first then concatenation and last alternation.
- Parentheses ‘(’ and ‘)’ override the precedence relations above.