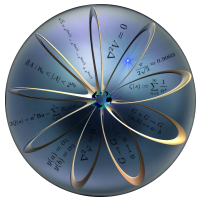

planetmath.org

Math for the people, by the people.

regular expression

Canonical name	RegularExpression
Date of creation	2013-03-22 12:26:56
Last modified on	2013-03-22 12:26:56
Owner	CWoo (3771)
Last modified by	CWoo (3771)
Numerical id	17
Author	CWoo (3771)
Entry type	Definition
Classification	msc 20M35
Classification	msc 68Q70
Related topic	RegularLanguage
Related topic	KleeneStar
Related topic	KleeneAlgebra

A *regular expression* is a particular meta-syntax for specifying regular grammars, which has many useful applications.

While variations abound, fundamentally a regular expression consists of the following pieces:

- Parentheses can be used for grouping and nesting, and must contain a fully-formed regular expression.
- The `|` symbol can be used for denoting alternatives. Some specifications do not provide nesting or alternatives.
- There are also a number of postfix operators. The `?` operator means that the preceding element can either be present or non-present, and corresponds to a rule of the form $A \to B \,|\, \lambda$.
- The `*` operator means that the preceding element can be present zero or more times, and corresponds to a rule of the form $A \to BA \,|\, \lambda$.
- The `+` operator means that the preceding element can be present one or more times, and corresponds to a rule of the form $A \to BA \,|\, B$.

Note that while these rules are not immediately in regular form, they can be transformed so that they are.

Formally, let $S = \{\varnothing, \cup, ^*, (,)\}$ and Σ an alphabet disjoint from S. Consider the language $L(\Sigma)$ over $\Sigma \cup S$ specified below

1. $\varnothing \in L(\Sigma)$,
2. $a \in L(\Sigma)$ for each $a \in \Sigma$,
3. if $u \in L(\Sigma)$, then $u^* \in L(\Sigma)$,
4. if $u_1, u_2 \in L(\Sigma)$, then $(u_1 \cup u_2)$ and $(u_1 u_2)$ are both in $L(\Sigma)$, and
5. among all languages over $\Sigma \cup S$ satisfying conditions 1-4, $L(\Sigma)$ is the smallest.

Then any element $u \in L(\Sigma)$ is called a *regular expression* over Σ.

Here is an example of a regular expression that specifies a grammar that generates the binary representation of all multiples of 3 (and only multiples of 3).

$$(0^*(1(01^*0)^*1)^*)^*0^*$$

This specifies the context-free grammar (in BNF):

$$\begin{array}{lll} S & ::= & AB \\ A & ::= & CD \\ B & ::= & \mathtt{0}B|\lambda \\ C & ::= & \mathtt{0}C|\lambda \\ D & ::= & \mathtt{1}E\mathtt{1} \\ E & ::= & FE|\lambda \\ F & ::= & \mathtt{0}G\mathtt{0} \\ G & ::= & \mathtt{1}G|\lambda \end{array}$$

A little further work is required to transform this grammar into an acceptable form for regular grammars, but it can be shown that this grammar (and any grammar specified by a regular expression) is equivalent to some regular grammar.

One can understand the language described by a regular expression in another way, by viewing the regular expression operators as shorthand for various set-theoretic operations. Formally, the *language* $L(u)$ *over* Σ *associated with a regular expression* u over Σ is inductively defined as follows:

- $L(\varnothing) = \varnothing$,
- $L(a) = \{a\}$ whenever $a \in \Sigma$,
- $L(u^*) = L(u)^*$, where the * on the right side is the Kleene star operation on sets,
- $L((u_1 u_2)) = L(u_1)L(u_2)$, where the right side denotes the concatenation of two sets, and
- $L((u_1 \cup u_2)) = L(u_1) \cup L(u_2)$, where $\cup$ on the right side is the union operation on sets.

A language L over Σ is regular iff there is a regular expression u over Σ such that $L = L(u)$.

With this interpretation, it is quite straightforward to design a nondeterministic finite automaton that recognizes the language described by a regular expression. Of course, for computer implementations, one must transform this into a deterministic finite automaton, but there are various algorithms for doing this efficiently. This process, production of a nondeterministic automaton and conversion to an equivalent deterministic automaton is approximately what is done in software packages implementing regular expression searching. In fact, most such packages implement operations impossible for a finite automaton, such as requiring a later part of the string to be the same as a previous part (the language $\{A^n B A^n \text{ for } n \geq 0\}$ is not regular but can be matched by most "regular expression" software; such capabilities are called "extended regular expressions". None of these systems are powerful enough to recognize the language of balanced parentheses.

Regular expressions have many applications. Quite often they are used for powerful string matching and substitution features in many text editors and programming languages.