

Re: Regular expression – what with *

Source: <http://sci.tech-archive.net/Archive/sci.math/2006-11/msg02339.html>

- *From:* Helmut Richter <hhr-m@xxxxxx>
 - *Date:* Wed, 8 Nov 2006 22:03:37 +0100
-

On Wed, 8 Nov 2006, pgn@xxxxxxxxxx wrote:

1. I have a regular expr. like this: $((a+b)a^*)^*$ and I'm trying to describe it, and it's like this in my opinion: at the very beginning language generated by this regular expr. has 'a' or 'b' and then multiple of 'a' (empty, a, aa...). And now my question what about the last '*' sign? How to write it down?

Take any string over the alphabet {a, b}, e.g.

baabbababaabbbababbbabab

Now cut it into pieces just before each occurrence of b:

baa b ba ba baa b b ba ba b b ba ba b

Which of the pieces does not belong to the language $(a+b)a^*$?

What can be concluded for the language $((a+b)a^*)^*$?

2. I'm trying to find regular expression which generates languages where in word i cannot have 'bbb' (there can be ...bb... or ..b...b... or ...b...). So far I have something like this $a^* + (a^*ba^*) + (a^*bba^*) + (a^*ba^*ba^*)$. Is it true or there is something missing?

The following string is missing:

abababa

It does not contain bbb but is not in your regular expression.

The only (moderately) simple way to get a regular expression for $\neg((a+b)^*bbb(a+b)^*)$ where the "¬" denotes complement is to construct a finite automaton and transform it back to a regular expression. It is easy to construct a finite automaton for $(a+b)^*bbb(a+b)^*$. Complete it by introducing a new state catching all missing transitions. Then a finite

Re: Regular expression – what with *

automaton for the complement looks the same with final states becoming non-final and vice versa. The process is described in my Web page <http://www.lrz-muenchen.de/services/schulung/unterlagen/regul/regul-12.html#publish4.1.1.0.0.0>, albeit in German language. But even if you do not understand the text, you can look at the three sketches: the second automaton is the completion of the first one and accepts the same language, and the third accepts the complement.

Another way to do it works with derivatives of regular expressions, invented by your fellow countryman (but now in the U.S.) Janusz Brzozowski. But there is much to explain, and I am afraid that it will be as complex in the end.

--

Helmut Richter

.