The preceding expression will match E. B. White, Edgar Allen Poe, and Alan Turing. Since pattern matching is restricted to the form of the strings and not any underlying meaning (that is, pattern matching checks syntax and not semantics), the expression will also match Buckingham Palace and U. S. Mail. Moreover, the pattern will not match Vincent van Gogh, Dr. Watson, or Aristotle. Additional conditions would need to be added to the expression to match these variations of names.

Unlike off-line analysis, search commands in web browsers or word processors interactively find occurrences of strings that match an input pattern. A substring matching a pattern may span several lines. The pattern `m*n` in the Microsoft Word "Find" command searches for substrings beginning with `m` and ending with `n`; any string may separate the `m` and `n`. The search finds and highlights the first substring beginning at or after the current location of the cursor that matches the pattern. Repeating the search by clicking "next" highlights successive matches of the pattern. The substrings identified as matches of `m*n` in the file caesar follow, with the matching substrings highlighted.

Cowards die *many* times before their deaths;

Cowards die many t*imes before their deaths;*
*The valiant n*ever taste of death but once.

It se*ems to me most strang*e that men should fear;

It seems to *me most strang*e that men should fear;

It seems to me *most strang*e that men should fear;

It seems to me most strange that *men* should fear;

Will co*me when* it will come.

Notice that not all matching substrings are highlighted. The pattern `m*n` is matched by any substring that begins with an occurrence of `m` and extends to any subsequent occurrence of `n`. The search only highlights the first matching substring for every `m` in the file.

In Chapter 6 we will see that a regular expression can be converted into a finite-state machine. The computation of the resulting machine will find the strings or substrings that match the pattern described by the expression. The restrictions on the operations used in regular expressions—intersection and set difference are not allowed—facilitate the automatic conversion from the description of a pattern to the implementation of a search algorithm.

Exercises

1. Give a recursive definition of the length of a string over Σ. Use the primitive operation from the definition of string.

2. Using induction on i, prove that $(w^R)^i = (w^i)^R$ for any string w and all $i \geq 0$.
3. Prove, using induction on the length of the string, that $(w^R)^R = w$ for all strings $w \in \Sigma^*$.
4. Let $X = \{aa, bb\}$ and $Y = \{\lambda, b, ab\}$.
   a) List the strings in the set XY.
   b) How many strings of length 6 are there in X^*?
   c) List the strings in the set Y^* of length three or less.
   d) List the strings in the set X^*Y^* of length four or less.
5. Let L be the set of strings over $\{a, b\}$ generated by the recursive definition
   i) Basis: $b \in$L.
   ii) Recursive step: if u is in L then $ub \in L$, $uab \in$L, and $uba \in$L, and $bua \in$L.
   iii) Closure: a string v is in L only if it can be obtained from the basis by a finite number of iterations of the recursive step.
   a) List the elements in the sets L_0, L_1, and L_2.
   b) Is the string *bbaaba* in L? If so, trace how it is produced. If not, explain why not.
   c) Is the string *bbaaaabb* in L? If so, trace how it is produced. If not, explain why not.
6. Give a recursive definition of the set of strings over $\{a, b\}$ that contain at least one b and have an even number of a's before the first b. For example, *bab*, *aab*, and *aaaababababab* are in the set, while *aa*, *abb* are not.
7. Give a recursive definition of the set $\{a^i b^j \mid 0 \leq i \leq j \leq 2i\}$.
8. Give a recursive definition of the set of strings over $\{a, b\}$ that contain twice as many a's as b's.
9. Prove that every string in the language defined in Example 2.2.1 has even length. The proof is by induction on the recursive generation of the strings.
10. Prove that every string in the language defined in Example 2.2.2 has at least as many a's as b's. Let $n_a(u)$ denote the number of a's in the string u and $n_b(u)$ denote the number of b's in u. The inductive proof should establish the inequality $n_a(u) \geq n_b(u)$.
11. Let L be the language over $\{a, b\}$ generated by the recursive definition
    i) Basis: $\lambda \in$ L.
    ii) Recursive step: If $u \in$ L then $aaub \in$ L.
    iii) Closure: A string w is in L only if it can be obtained from the basis by a finite number of applications of the recursive step.
    a) Give the sets L_0, L_1, and L_2 generated by the recursive definition.
    b) Give an implicit definition of the set of strings defined by the recursive definition.
    c) Prove by mathematical induction that for every string u in L, the number of a's in u is twice the number b's in u. Let $n_a(u)$ and $n_b(u)$ denote the number of a's and the number of b's in u, respectively.

* 12. A **palindrome** over an alphabet Σ is a string in Σ^* that is spelled the same forward and backward. The set of palindromes over Σ can be defined recursively as follows:

   i) Basis: λ and a, for all $a \in \Sigma$, are palindromes.

   ii) Recursive step: If w is a palindrome and $a \in \Sigma$, then awa is a palindrome.

   iii) Closure: w is a palindrome only if it can be obtained from the basis elements by a finite number of applications of the recursive step.

   The set of palindromes can also be defined by $\{w \mid w = w^R\}$. Prove that these two definitions generate the same set.

13. Let $L_1 = \{aaa\}^*$, $L_2 = \{a, b\}\{a, b\}\{a, b\}\{a, b\}$, and $L_3 = L_2^*$. Describe the strings that are in the languages L_2, L_3, and $L_1 \cap L_3$.

For Exercises 14 through 38, give a regular expression that represents the described set.

14. The set of strings over $\{a, b, c\}$ in which all the a's precede the b's, which in turn precede the c's. It is possible that there are no a's, b's, or c's.
15. The same set as Exercise 14 without the null string.
16. The set of strings over $\{a, b, c\}$ with length three.
17. The set of strings over $\{a, b, c\}$ with length less than three.
18. The set of strings over $\{a, b, c\}$ with length greater than three.
19. The set of strings over $\{a, b\}$ that contain the substring ab and have length greater than two.
20. The set of strings of length two or more over $\{a, b\}$ in which all the a's precede the b's.
21. The set of strings over $\{a, b\}$ that contain the substring aa and the substring bb.
22. The set of strings over $\{a, b\}$ in which the substring aa occurs at least twice. *Hint:* Beware of the substring aaa.
23. The set of strings over $\{a, b, c\}$ that begin with a, contain exactly two b's, and end with cc.
* 24. The set of strings over $\{a, b\}$ that contain the substring ab and the substring ba.
25. The set of strings over $\{a, b, c\}$ in which every b is immediately followed by at least one c.
26. The set of strings over $\{a, b\}$ in which the number of a's is divisible by three.
27. The set of strings over $\{a, b, c\}$ in which the total number of b's and c's is three.
* 28. The set of strings over $\{a, b\}$ in which every a is either immediately preceded or immediately followed by b, for example, $baab$, aba, and b.
29. The set of strings over $\{a, b, c\}$ that do not contain the substring aa.
30. The set of strings over $\{a, b\}$ that do not begin with the substring aaa.
31. The set of strings over $\{a, b\}$ that do not contain the substring aaa.
* 32. The set of strings over $\{a, b\}$ that do not contain the substring aba.
33. The set of strings over $\{a, b\}$ in which the substring aa occurs exactly once.
34. The set of strings of odd length over $\{a, b\}$ that contain the substring bb.
35. The set of strings of even length over $\{a, b, c\}$ that contain exactly one a.
36. The set of strings of odd length over $\{a, b\}$ that contain exactly two b's.
37. The set of strings over $\{a, b\}$ with an even number of a's or an odd number of b's.
* 38. The set of strings over $\{a, b\}$ with an even number of a's and an even number of b's. This is tricky; a strategy for constructing this expression is presented in Chapter 6.
39. Use the regular expression identities in Table 2.1 to establish the following identities:

   a) $(ba)^+(a^*b^* \cup a^*) = (ba)^*ba^+(b^* \cup \lambda)$

   b) $b^+(a^*b^* \cup \lambda)b = b(b^*a^* \cup \lambda)b^+$

   c) $(a \cup b)^* = (a \cup b)^*b^*$

   d) $(a \cup b)^* = (a^* \cup ba^*)^*$

   e) $(a \cup b)^* = (b^*(a \cup \lambda)b^*)^*$

40. Write the output that would be printed by a search of the file caesar described in Section 2.4 with the following extended regular expressions.

   a) `[Cc]`

   b) `[K-Z]`

   c) `\<[a-z]{6}\>`

   d) `\<[a-z]{6}\>|\<[a-z]{7}\>`

41. Design an extended regular expression to search for addresses. For this exercise, an address will consist of

   i) a number,

   ii) a street name, and

   iii) a street type identifier or abbreviation.

   Your pattern should match addresses of the form `1428 Elm Street`, `51095 Tobacco Rd.`, and `1600 Pennsylvania Avenue`. Do not be concerned if your regular expression does not identify all possible addresses.

Bibliographic Notes

Regular expressions were developed by Kleene [1956] for studying the properties of neural networks. McNaughton and Yamada [1960] proved that the regular sets are closed under the operations of intersection and complementation. An axiomatization of the algebra of regular expressions can be found in Salomaa [1966].