Finite-state algorithms in text processing

This chapter describes a simple technique often used to process input, change text encodings etc. We consider two examples of "lexical analysis" that can occur during the first pass of a compiler. In section 5.1 we show how to process multi-character symbols. Then in section 5.2 we show how a finite automaton can convert string representation of a number to its numeric value. More advanced applications of finite-state machines are described in chapter 10.

5.1 Compound symbols, comments, etc.

5.1.1. Throughout a program text the operation $x^y$ was denoted by $x**y$. It was decided that notation should be changed to $x^y$. How do we do that? The input text is read character-by-character; the output text should be produced in the same manner.

Solution. At any time, the program is in one of two states: "basic" state and "after" state (after an asterisk):

<table>
<thead>
<tr>
<th>State</th>
<th>Next symbol</th>
<th>New state</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>*</td>
<td>after</td>
<td>none</td>
</tr>
<tr>
<td>basic</td>
<td>$x \neq *$</td>
<td>basic</td>
<td>print $x$</td>
</tr>
<tr>
<td>after</td>
<td>*</td>
<td>basic</td>
<td>print $^*$</td>
</tr>
<tr>
<td>after</td>
<td>$x \neq *$</td>
<td>basic</td>
<td>print $^*$, $x$</td>
</tr>
</tbody>
</table>

If after reading all the text, the program is in the "after" state, it should print an asterisk (and quit).

Remark. Our program replaces *** by $^*$ (and not by $^*$). We did not specify the behavior of the program in this case, assuming (as is often done) that some "reasonable" behavior is expected. In this example, the simplest way to describe the required behavior is to list the states and the corresponding actions.

Please note also that if two asterisks appear in other parts of the program (say, comments), they will be also replaced.
5.1.2. Write a program that deletes all occurrences of the substring abc.

5.1.3. In Pascal, comments are surrounded by curly braces like this:

```
begin {here a block begins}
i:=i+1; {increase i by one}
```

Write a program that removes all comments and puts a space character in the place of a removed comment. (According to Pascal rules, 1{one}2 is equivalent to 1 2, not 12).

**Solution.** The program has two states: a “basic” state and an “inside” state (inside a comment).

<table>
<thead>
<tr>
<th>State</th>
<th>Next symbol</th>
<th>New state</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>{</td>
<td>inside</td>
<td>none</td>
</tr>
<tr>
<td>basic</td>
<td>x ≠ (</td>
<td>basic</td>
<td>print x</td>
</tr>
<tr>
<td>inside</td>
<td>}</td>
<td>basic</td>
<td>print a space</td>
</tr>
<tr>
<td>inside</td>
<td>x ≠ }</td>
<td>inside</td>
<td>none</td>
</tr>
</tbody>
</table>

This program cannot deal with nested comments: the string

```
{{comment inside a} comment}
```

is transformed into

```
comment}
```

(the latter string starts with two spaces). It is impossible to deal with nested comments using a finite automaton (a program that has finite number of internal states); roughly speaking, we have to remember the number of opening braces and a finite automaton cannot do that.

Please note that after reading all the text, the program may still be in the “inside” state. Most probably, we would like to consider this as an error.

5.1.4. Pascal programs also contain quoted strings. If a curly brace appears inside a string, it does not mean the start of a comment. Similarly, a quote symbol inside a comment does not signify a string. How do we modify the above program to take this into account?

*[Hint. We need three states: “basic”, “inside a comment”, “inside a string”]*

(Note that actual Pascal conventions are more complicated allowing a quote to appear inside a quoted string, etc.)

5.1.5. One more feature that exists in many Pascal implementations is a comment of the type

```
i := i+1; (* here i is increased by 1 *)
```

A closing comment symbol must be paired with an opening comment symbol of the same type (e.g., {...*} is not permitted). How do we deal with these types of comments?