Solution to Writing Assignment I

Released: Sunday, 06/07/99

Due: Sunday, 13/07/99 at 11:59pm

This assignment covers lectures 1 to 2 about regular languages, finite automata and lexical analysis. You can discuss and work out this assignment with other students. However, your write-up must be your own individual work. Solutions must be typed (i.e., hand written solutions will not be corrected) except for Figures and Diagrams that can be drawn by hand. A PDF format of the solution (with your full name and student number) should be uploaded in the course page in Quera before 11:59 PM, Sunday 13/07/99 (https://quera.ir/course/6364/). Submissions with more than 50 hours delay will not be graded. Submissions with less than 50 hours delay will be penalized by the following rule:

Penalized mark = M * (100 - 2 * D) / 100

Where M = the mark achieved from your solution and D is number of hours passed the deadline.

1. Consider the following set of token types:

<table>
<thead>
<tr>
<th>Token Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM</td>
<td>Any string matching: [0-9]*</td>
</tr>
<tr>
<td>ID</td>
<td>Any string matching: [A-Za-z][A-Za-z0-9]*</td>
</tr>
<tr>
<td>KEYWORD</td>
<td>if, else, void, int, while, break, switch, default, case, return</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>; : [ ] { } ( ) + * = &lt;=</td>
</tr>
<tr>
<td>COMMENT</td>
<td>Any string between a /* and a */ OR any string after a // and before a \n</td>
</tr>
<tr>
<td>WHITESPACE</td>
<td>blank (ASCII 32), \n (ASCII 10), \r (ASCII 13), \t (ASCII 9), \v (ASCII 11), \f (ASCII 12)</td>
</tr>
</tbody>
</table>

- Draw appropriate DFAs (i.e., similar to the DFAs in pages 56-59 in Lecture note 3) for recognizing these tokens. Note that ID and KEYWORD are recognized by the same DFA, which is almost identical to the one on Page 58 of Lecture note 3!
- In each of these DFAs, specify exactly what characters should be considered as compatible to the 'other' label. Note label 'other' in different DFAs are not necessarily referring to the same set of symbols.
- Then combine these DFAs into a single DFA. Note that you can use the resultant DFA (provided that the DFA is correct and complete) as a flowchart for implementing your scanner in Programming Assignment 1.
**NUM**

- Start state: 0
- Transition: digit \(\rightarrow\) 3
- Transition: other \(\rightarrow\) 4
- Transition: digit \(\rightarrow\) 3

**Symbol**

- Start state: 0
- Transition: \{\};\[]()\{\}+\-< \(\rightarrow\) 5
- Transition: = \(\rightarrow\) 6
- Transition: = \(\rightarrow\) 7
- Transition: other1 \(\rightarrow\) 6
- Transition: other2 \(\rightarrow\) 8
- Transition: * \(\rightarrow\) 9

**Comment**

- Start state: 0
- Transition: / \(\rightarrow\) a
- Transition: / \(\rightarrow\) b
- Transition: \n \(\rightarrow\) c
- Transition: * \(\rightarrow\) d
- Transition: * \(\rightarrow\) e
- Transition: other3
- Transition: other4
- Transition: other5

**Other**

- Other = any valid character except for letter and digit
- Other1 = any valid character except for '='
- Other2 = any valid character except for '/'
- Other3 = any character except for '\n'
- Other4 = any character except for '\'*
- Other5 = any character except for '\'* and '/'
WHITESPACE
other = any valid character except for letter and digit

other1 = any valid character except for '='

other2 = any valid character except for '/'

other3 = any character except for '\n'

other4 = any character except for '*'

other5 = any character except for '*\n' and '/'
2. Write a regular expression for the following language over the alphabet $\Sigma = \{0-9, a-z\}$:

The set of all identifiers with at most four characters that start with a letter and continue with more letters or digits. In an identifier, the number of digits cannot be more than the number of letters. Note that in this assignment you can ONLY use those operators that have been defined on page 3 in Lecture note 3. In other words, you cannot use other operators that may be available in the regex function of Python. Note that as a general rule, the solution to all writing assignments should be based solely on the theoretical concepts introduced in the Lecture notes.

\[
L \leftarrow [a-z] \\
D \leftarrow [0-9] \\
ID \leftarrow L (L (L | D)? (L | D)? | D (L (L | D)? | D L)?)?
\]

3. Draw a DFA for the language defined in the previous question. Note that DFA should recognize the input string as a correct ID only if the whole input string would be a correct ID. For instance if the input string is 'ab12', it is a correct ID but 'ab12(' should be rejected because it has an extra character '('.

4. In a programming language, comments start with /* and end with */. Comments can contain any character strings except for */ unless it is surrounded by two double quote character ("'). The followings are example of valid and invalid comments. Note that each */ inside a comment must have its own surrounding double quotes and two adjacent */ cannot share a double quote (see the 3rd invalid case). Also, */ must be immediately surrounded by two double quotes (i.e., there cannot be any extra blank character; see the 2nd invalid case).

<table>
<thead>
<tr>
<th>Valid Comment</th>
<th>Invalid Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>/<em>abc</em>/&quot;de*/</td>
<td>/<em>abc</em>/de*/</td>
</tr>
<tr>
<td>/**/</td>
<td>/<em>ab&quot;</em>/&quot;  &quot;*/</td>
</tr>
<tr>
<td>/<em>21</em>/&quot;&quot;43*/</td>
<td>/<em>21</em>/&quot;&quot;43*/</td>
</tr>
<tr>
<td>/<em>a2</em>/</td>
<td>/<em>a2&quot;</em>/</td>
</tr>
</tbody>
</table>

Write a regular expression that defines the set of comments in the above language.
any ← {all characters except / and .}
comment ← /\ (."\"/" | / | ."any")\ ./\\n
In all above questions, try to find the most compact answer (i.e., the shortest regular expression and the smallest DFA).

Ghassem Sani, Gholamreza

16/07/2020