CS 6410: Compilers

Fall 2019

HW 1 – Regular Expressions and Scanners
Solutions

Assigned: Thursday, September 27, 2023, Due: Saturday, October 14, 2023

Instructor: Tamara Bonaci
Khoury College of Computer Sciences
Northeastern University – Seattle

Submission Guidelines

- Please submit your homework as a single .pdf file through Canvas.
- You do not have to type in your submission - hand-written and then scanned, or photographed documents are fine, as long as your document is readable.
- This assignment is meant to be worked on individually, and you should submit it by **11:59pm on Saturday, October 13, 2023**.

Hints on Regular Expressions

For questions that ask for regular expressions, you **must** restrict yourself to the basic operations and abbreviations, including:

- Concatenation,
- Alternation, |
- Kleene closure, $*$
- Positive closure, $+$,
- Question mark, ?, and
- Character classes, [. . .]

Additionally, you may specify character classes containing all characters except for specific characters, e.g., $[\wedge abc]$. You can also use abbreviations, and be somewhat informal when the meaning is clear, e.g., (all except x) $= a|b|c|...|w|y|z$. You should not use additional "regular expression" operations found in libraries or languages like perl, python, or ruby, or in unix tools like grep and sed. In particular, you cannot use $not(re)$ as a regular expression that matches all other regular expressions except for re.

Problem 1

For each of the following regular expressions, please give:

1. An example of two strings that can be generated by the given regular expression,
2. An example of two strings that use the same alphabet, but cannot be generated, and
3. An English description of the set of strings generated (for example, "all strings consisting of the word 'cow' followed by 1 or more 'x's and 'o's in any order", not just a transliteration of the regular expression operations into English)

Regular expressions are as follows:

- $(\text{to}(\text{m}|\text{y}))^+$
- $\text{h}(\text{i}|\text{o})(\text{pity})^*$
- $((\epsilon|206)299)^?$

Problem 2

Please state regular expressions that generate the following sets of strings:

- All strings of d's and e's with at least 3 d's.
- All strings of d's and e's where e's only appear in sequences whose length is a multiple of 2 (a few examples: deed, eeeedeeddd, d and ϵ are in this set; ded, e, deded, and deede are not).
- All strings of lower-case letters that contain the 5 vowels (aeiou) exactly once and in that order, with all other possible sequences of lower-case letters before, after, or in between the individual vowels.

Problem 3 (Cooper and Torczon, Problem 2.2.1)

Please informally describe the languages accepted by the finite automata (FAs) depicted in Figure 1.

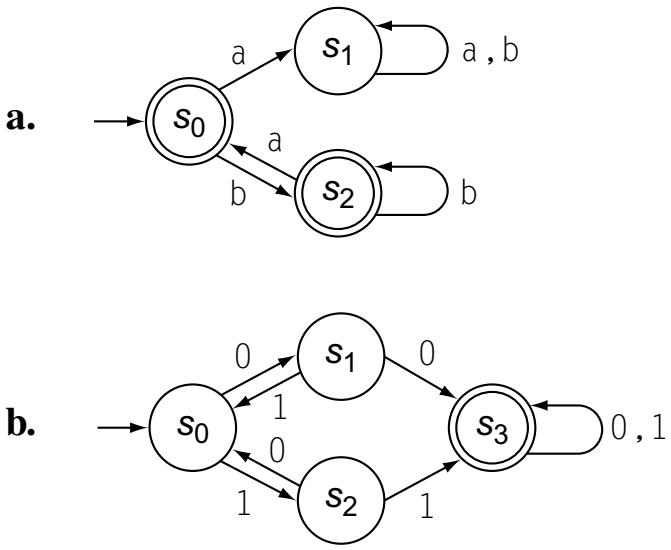

Fig. 1. Finite automate used in Problem 3.

Problem 4 (Cooper and Torczon, Problem 2.2.2)

Please construct a DFA accepting each of the following languages:

1. {w in {a,b} * | w starts with 'a' and contains 'baba' as a substring}
2. {w in {0,1}* | w contains '111' as a substring and does not contain '00' as a substring}

Problem 5

In The C Programming Language (Kernighan and Ritchie), an integer constant is defined as follows:
An integer constant consisting of a sequence of digits is taken to be octal if it begins with 0 (digit zero), decimal otherwise. Octal constants do not contain the digits 8 or 9. A sequence of digits preceded by $0x$ *or* $0X$ *(digit zero) is taken to be a hexadecimal integer. The hexadecimal digits include a or A through f or F with values 10 through 15. An integer constant may be suffixed with the letter u or U, to specify that it is unsigned. It may also be suffixed by the letter l or L to specify that it is long.*

1. Write a regular expression that generate C integer constants as described above.
   Hint 1: you may want to break the solution down into a regular expression with several named parts, if it makes things easier to write and read – which it probably will.
2. Draw a DFA that recognizes integer constants as defined by your solution to part (a). You may draw this directly; you don't need to formally trace through an algorithm for converting a regular expression to a NFA, and then constructing a DFA from that. However, you might find it useful to do so at least partially.
   Hint 2: You might find it helpful to alternate between designing the DFA and writing the regular expressions as you work on your solution.

Problem 6

In C programming language, a comment is a sequence of characters between characters $/*...*/$. Write a set of regular expressions that generate C-style comments. You can restrict the alphabet to lower-case letters, digits, spaces, newlines ($\backslash n$), carriage returns ($\backslash r$) and the characters $*$ and $/$. Also, remember that in C comments do not nest (i.e., a $*/$ marks the end of a comment no matter how many times $/*$ appears before it.)
Hint 3: be careful about what is included in the ... between $/*$ and $*/$.