Imperial College London
Department of Computing

Compilers (221)

Exercises - Lexical Analysis

Check your answers with the tutorial helpers during tutorials and with each other on Piazza.

PART 1 - Regular Expressions.

Tips. (1) When devising a regular expression also devise good test cases e.g. shortest strings it should match, strings it shouldn't match. (2) To aid thinking and for clarity use spaces to separate parts of a complicated expression, (3) A good approach for developing regular expressions that involve regular expression meta-characters is to re-cast the question changing the meta-character to standard characters that are easier to work with, developing a solution and then changing the meta-characters back, escaping them when necessary.

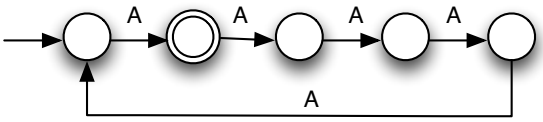
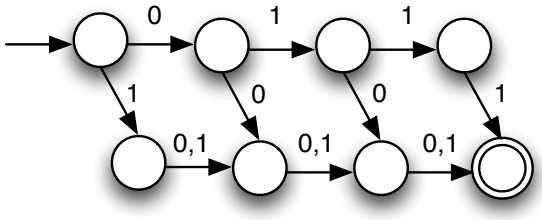
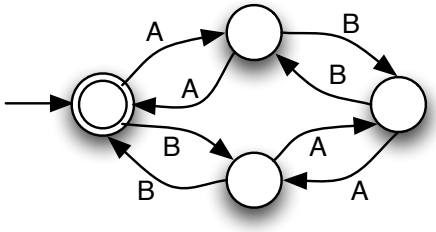
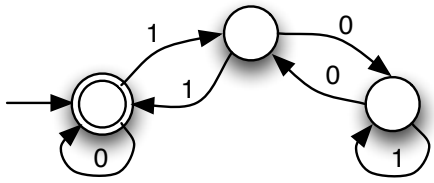
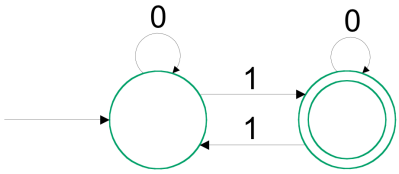
Suggested order to do the questions: 1, 2, 3, 12, 14, 15, 16, 20, 23, 25 and then the remaining questions.

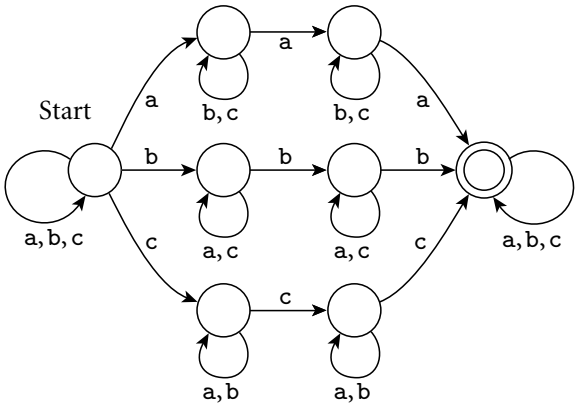
1.	Rewrite each of the following expressions using only the basic regular expression operators *, |, concatenation, grouping () and ε: (i) `[ABCD]` (ii) `M?` (iii) `M+`	L1
2.	Give a regular expression for **optionally signed floating points numbers**. You can assume that the forms `3.` (no fractional part) and `.3` (not integer part) are not permitted. Allow for an optional exponent part also. Use the notation `\X` to escape Regex special characters like dot.	L2
3.	For the alphabet `{A,B,C}` give a regular expression for strings that contain **exactly one `B`**.	L1
4.	For the alphabet `{A,B,C}` give a regular expression for strings that contain **at most one `B`**.	L1
5.	For the alphabet {A, B} give a regular expression for all strings **containing exactly two B's**.	L1
6.	For the alphabet `{A,B,C}` give a regular expression for strings that **do not contain two consecutive `B`'s,** i.e. between any two `B`'s there must be at least one `A` or one `C`.	L2
7.	For the alphabet {A, B} give a regular expression for all strings in which **every A is immediately followed by at least two B's**.	L1
8.	For the alphabet {A. B} give a regular expression for all strings containing **two consecutive `A`'s or two consecutive `B`'s**.	L1
9.	For the alphabet {A. B} give a regular expression for all strings containing an **even number of A's and an even number of B's**.	L4
10.	Devise a regular expression for all binary strings of 0's and 1's with **at most one pair of consecutive 1's**.	L3
11.	Devise a regular expression for all strings of A's, B's and C's **where no letter can appear more than once.**	L4
12.	Give a regular expression for **Pascal-style comments** of the form `{ XXX }` where `XXX` is any sequence of characters except }, i.e. where the comment is enclosed in braces. You can use the shortcut `[^A]` to denote any character except the character `A`, `[^AB]` for any character except the characters `A` and `B`, and so on.	L2
13.	Give a regular expression for **C-style comments** of the form `/* comment */` Heed tip 3 above.	L5
14.	Give a regular expression for the set of strings in the set $\{0^n1^n : n>=1\}$. That is, the regular expression should accept **n** zeroes followed by **n** ones. Do not spend more than 5 minutes on this question!	L2

PART 2 - DFAs.

Tip. When reasoning about DFAs, it can be very useful to list out matching strings, starting with the shortest ones.

15.	Give a short (one sentence) English description for the following DFA.	L2

		
16.	Give a short English description for the following DFA (has a small gotcha). 	L2
17.	Give a short English description for the following DFA. 	L4
18.	Give a short English description for the following DFA. 	L5
19.	Give a regular expression for the following DFA and state what set of strings it accepts: 	L4
20.	Derive the NFA for the regular expression `(A|B)*ABB`. Number your states from 0. You should have 11 states (0 to 10).	L2
21.	Give a DFA for C-style comments i.e. /* comment with no appearances of */ followed by */ Use the label **not X** for any char except X, e.g. **not ***, **not /**	L4
22.	Use the subset construction to derive the DFA for the regular expression `L(L|D)*` from its NFA (see slide with NFA for an identifier in lecture slides). You should have 4 states.	L3
23.	Using the subset construction derive the DFA for the regular expression `(A|B)*ABB` from its NFA (using your solution from question above). You should have 5 states.	L4
24.	Derive the minimal DFA for the regular expression `(A|B)*ABB` from your DFA above. You should have 4 states. OPTIONAL MATERIAL	L4
25.	For the regular expression: `(AB|AC)*` i) Convert the regular expression to a NFA using Thompson's construction (10 states) ii) Convert the NFA to a DFA using the Subset construction (4 states)	L3

	iii) Minimise the DFA (2 states) OPTIONAL	
26.	What strings does the following NFA recognise? 	L5