Homework 1 – ECS 120, Winter 2020

1 Auto-graded problems

These problems are *not randomized*, so there is no need to first submit a file named `req`. Each problem below appears as a separate "Assignment" in Gradescope, beginning with "HW1:".

1.1 DFAs

For each problem submit to Gradescope a `.dfa` file describing a DFA deciding the given language. Make sure that it is a *plain text file* that ends in `.dfa` (*not* `.txt`). Use the finite automata simulator to test the DFAs: http://web.cs.ucdavis.edu/~doty/automata/. Documentation is available at the help link at the top of that web page.

Do not just submit to Gradescope without testing on the simulator. The purpose of this homework is to develop intuition. You'll develop more intuition by running the DFA in the simulator, trying to come up with some of your own examples and seeing where they fail, than you will by just using the Gradescope autograder as a black box. One you think your solution works, submit to Gradescope. If you fail any test cases, *go back to the simulator* and use it to see **why** those cases fail. During an exam, there's no autograder to help you figure out if your answer is correct. Practice right now how to determine for yourself whether it is correct.

Please test on the simulator *before* submitting to Gradescope. If your file is not formatted properly, the simulator will tell you this. Also, if you lose points on a Gradescope test case, try that test case in the simulator to ensure that your DFA is behaving as you expect.

begin and end: $\{w \in \{0,1\}^* \mid w$ begins with 010 and ends with a 0 $\}$

at most three 1s: $\{w \in \{0,1\}^* \mid w$ contains at most three 1's$\}$.

no substring: $\{w \in \{\texttt{a},\texttt{b},\texttt{c}\}^* \mid w$ does not contain the substring `acab`$\}$.

even odd: $\{w \in \{\texttt{a},\texttt{b}\}^* \mid w$ starts with `a` and has even length, or w starts with `b` and has odd length $\}$.

mod: $\{w \in \{0,1\}^* \mid w$ is the binary expansion of $n \in \mathbb{N}$ and $n \equiv 3 \mod 5\}$. Assume ε represents 0 and that leading 0's are allowed. A number $n \in \mathbb{N}$ is *congruent* to 3 mod 5 (written $n \equiv 3 \mod 5$) if n is 3 greater than a multiple of 5, i.e., $n = 5k + 3$ for some $k \in \mathbb{N}$. For instance, 3, 8, and 13 are congruent to 3 mod 5.

1.2 Regular expressions

For each problem submit to Gradescope a `.regex` file with a regular expression deciding the given language. Use the regular expression evaluator to test each regex: http://web.cs.ucdavis.edu/~doty/automata/. Do *not* test them using the regular expression library of a programming

language; typically these are more powerful and have many more features that are not available in the mathematical definition of regular expressions from the textbook. Only the special symbols `( ) * + |` are allowed, as well as "input alphabet" symbols: alphanumeric, and `.` and `@`.

even/odd/substring:

$$\left\{ x \in \{a,b\}^* \;\middle|\; \begin{array}{l} x \text{ has an even number of } a\text{'s, or } x \text{ has an odd number of } b\text{'s, or} \\ x \text{ contains both the substrings } babb \text{ and } aabaa \end{array} \right\}$$

first appears more:

$$\{x \in \{0,1\}^* \mid |x| \geq 3 \text{ and the first symbol of } x \text{ appears at least three times total in } x\}$$

repeat near end: $\{x \in \{0,1\}^* \mid \; x[|x|-5] = x[|x|-3] \;\}$

Assume we start indexing at 1, so that $x[|x|]$ is the last symbol in x.

email: $\{x \in \Sigma^* \mid x \text{ is a syntactically valid email address}\}$

Definition of "syntactically valid email address": Let Σ = {., @, `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h`, `i`, `j`, `k`, `l`, `m`, `n`, `o`, `p`, `q`, `r`, `s`, `t`, `u`, `v`, `w`, `x`, `y`, `z`, `A`, `B`, `C`, `D`, `E`, `F`, `G`, `H`, `I`, `J`, `K`, `L`, `M`, `N`, `O`, `P`, `Q`, `R`, `S`, `T`, `U`, `V`, `W`, `X`, `Y`, `Z`, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 } contain all alphanumeric symbols, as well as the symbols for period . and ampersand @. Syntactically valid emails are of the form *username*`@`*host*`.`*domain* where *username* and *host* are nonempty and may contain alphanumeric symbols or `.`, but never two `.`'s in a row, nor can either of them begin or end with a `.`, and *domain* must be of length 2 or 3 and contain only alphanumeric symbols. For example, `doty@ucdavis.edu` and `David.S.Doty2@cs.ucdavis.edu` are valid email addresses, but `ucdavis.edu` is not (no `@` symbol), nor is `.smith@ucdavis.edu` (*username* starts with a `.`), nor is `doty@ucdavis.education` (*domain* is too long), nor is `david..doty@ucdavis.edu` (two periods in a row).

Note: You will want to use the ability of the regex simulator to define subexpressions that can be used in the main regex. (See example that loads when you click "Load Default"). But since the input alphabet is the whole alphanumeric alphabet, it is crucial to use variable names for the subexpressions that are longer than one letter; e.g., if you write something like `A = (A|B|C);`, then later when you write `A`, it's not clear whether it refers to the symbol `A` or the subexpression `(A|B|C)`. Instead try something like `alphabet = (A|B|C);` and use `alphabet` in subsequent expressions. See here for more details: https://campuswire.com/c/G214DFB6C/feed/102

sequence design for DNA nanotechnology: We once designed some synthetic DNA strands that self-assembled to execute Boolean circuits: https://web.cs.ucdavis.edu/~doty/papers/#drmaurdsa. We had to be careful designing the DNA sequences to ensure they behaved as we wanted. Every sequence needed to obey all of the following rules:

- starts with a G or C and ends with a G or a C,
- has an A or T within two indices of each end (i.e., the second or third symbol is an A or T, and also the second-to-last or third-to-last symbol was an A or T),
- has at most one appearance of C,

- does not have four G's in a row (this would form something we didn't want, called a *G-quadruplex*: https://en.wikipedia.org/wiki/G-quadruplex)

Write a regex indicating strings that *violate* any of the rules above, i.e., it decides the following language: $\{x \in \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}^* \mid x \text{ violates at least one of the rules}\}$.

1.3 CFGs

For each problem submit to Gradescope a `.cfg` file with a context-free grammar deciding the given language.

mod length: $\{x \in \{a, b\}^* \mid |x| \equiv 3 \mod 5\}$

substring: $\{x \in \{a, b\}^* \mid x \text{ contains the substring } abba\}$

equal 0 and 1: $\{x \in \{0, 1\}^* \mid \#(0, x) = \#(1, x)\}$

palindrome: $\{x \in \{0, 1\}^* \mid x = x^{\mathcal{R}}\}$

Recall that $x^{\mathcal{R}}$ is the reverse of x.

first or last: $\{0^i 1^j 0^k \mid i, j, k \in \mathbb{N} \text{ and } (i = j \text{ or } j = k)\}$

integers: The set of strings that look like nonnegative decimal integers with no leading 0's. For example: 0, 1, 2, 3, 10, 11, 12, 21, 100, 99999

expressions: The set of strings that look like arithmetic expressions using nonnegative integers and the operations `+`, `-`, `*`, `/`, and parentheses to group terms.

For example, the following are properly formatted arithmetic expressions: `0`, `2`, `2+30`, `2+30*401`, `(2+30)*401/(23+0)`, `(((1+2)/3-4)*5+6)*7`

The following are not: `02`, `(2+30`, `2+30*401+`, `(2+30)*401)`, `-4`, `2++3`, `()`, `2*()`, `((((1+2)*3-4)*5+6)*7`

2 Written problems

Please complete the written portion of this homework on Gradescope, in the assignment titled "HW1 written". There, you will find the problem statements for the written portion. Please type solutions directly into Gradescope, using appropriate mathematical notation when appropriate, by typing LaTeX in double dollar signs. For example, type `$$D = (Q,\Sigma,\delta,s,F)$$` to display $D = (Q, \Sigma, \delta, s, F)$. By clicking outside the text entry field, you can see a preview of how the mathematics will render. See the second half of this page for examples: https://hackmd.io/cmThXieERK2AX_VJDqR3IQ?both

Your written solutions will be checked for completeness but not for correctness. To receive credit, you must make a serious attempt at all problems.