Теория Формальных Языков.

Лабораторная работа №1.

Масягин М. М.

ИУ9-52Б

Приложение 2. Список не формально заданных языков.

Задание-31.

L - множество слов в которых слово ab встречается не более двух раз. Слово abaaa.

Будем использовать лишь следующие операции: конкатенация, итерация, дизъюнцкция.

Общий алгоритм решения такой:

1) X — искомое множество. X = ((Y0 ab Y0 ab Y0)|(Y0 ab Y0)|(Y0)), где Y0 — множество всех слов, включая пустое, не содержащих в себе подстроку «ab» (хочу заметить, что порядок регулярок в дизъюнкции должен быть строго таким, как я указал, так как в противном случае, реальные регулярки на Python, C++ и тд будут находить не максимальное совпадение, то есть если раньше других будет стоять Y0, и будет задана строка «abab», то совпадение будет не «abab», а «a»);

2) Y0 - множество слов, включая пустое, не содержащих в себе подстроку «ab».

- Z0 = [c-z]\* - множество всех слов, которые можно составить из символов [c-z], включая пустое слово;

- Z = [c-z][c-z]\* = [c-z]+ - множество всех слов, которые можно составить из символов [c-z], **НЕ** включая пустое слово;

- Ya = Z0 a (a|Z0)\* - множество всех слов, которые содержат хотя бы одну «a», но не содержат «b»;

- Yb = Z0 b (b|Z0)\* - множество всех слов, которые содержат хотя бы одну «b», но не содержат «a»;

- Y0 = (Yb|Z0)(Ya Z Yb)\*(Ya|Z0) — как уже писалось ранее, множество всех слов, включая пустое, не содержащих в себе подстроку «ab»;

Реализация на Python (проверить можно на <https://regex101.com/>):

(((([c-z]\*b(b|([c-z]\*))\*)|([c-z]\*))(([c-z]\*a(a|([c-z]\*))\*)([c-z]+)([c-z]\*b(b|([c-z]\*))\*))\*(([c-z]\*a(a|([c-z]\*))\*)|([c-z]\*)))ab((([c-z]\*b(b|([c-z]\*))\*)|([c-z]\*))(([c-z]\*a(a|([c-z]\*))\*)([c-z]+)([c-z]\*b(b|([c-z]\*))\*))\*(([c-z]\*a(a|([c-z]\*))\*)|([c-z]\*)))ab((([c-z]\*b(b|([c-z]\*))\*)|([c-z]\*))(([c-z]\*a(a|([c-z]\*))\*)([c-z]+)([c-z]\*b(b|([c-z]\*))\*))\*(([c-z]\*a(a|([c-z]\*))\*)|([c-z]\*))))|(((([c-z]\*b(b|([c-z]\*))\*)|([c-z]\*))(([c-z]\*a(a|([c-z]\*))\*)([c-z]+)([c-z]\*b(b|([c-z]\*))\*))\*(([c-z]\*a(a|([c-z]\*))\*)|([c-z]\*)))ab((([c-z]\*b(b|([c-z]\*))\*)|([c-z]\*))(([c-z]\*a(a|([c-z]\*))\*)([c-z]+)([c-z]\*b(b|([c-z]\*))\*))\*(([c-z]\*a(a|([c-z]\*))\*)|([c-z]\*))))|((([c-z]\*b(b|([c-z]\*))\*)|([c-z]\*))(([c-z]\*a(a|([c-z]\*))\*)([c-z]+)([c-z]\*b(b|([c-z]\*))\*))\*(([c-z]\*a(a|([c-z]\*))\*)|([c-z]\*)))

Задание-11.

L - множество паролей со следующей политикой: в пароле должна быть только заглавные буквы, две цифры, длинна пароля не менее 3 в нем должно отсутствовать слова MBD.

Будем использовать лишь следующие операции: конкатенация, итерация, дизъюнцкция.

Общий алгоритм решения такой:

1) X — искомое множество. X = (Y [0..9] Y0 [0..9] Y0) | (Y0 [0..9] Y [0..9] Y0) | (Y0 [0..9] Y0 [0..9] Y), где Y0 — множество всех слов, включая пустое, не содержащих в себе подстроку «mbd», Y - множество всех слов, **НЕ** включая пустое, не содержащих в себе подстроку «mbd»;

2) Y0 - множество всех слов, включая пустое, не содержащих в себе подстроку «mbd».

- P0 - множество всех слов, включая пустое, не содержащих в себе «m». P = [a-ln-z]\*;

- Q0 - множество всех слов, включая пустое, не содержащих в себе «m» и не начинающихся с «bd». Q = (([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*));

- Y0 = P0(m Q0)\* - как уже писалось ранее, множество всех слов, включая пустое, не содержащих в себе подстроку «mbd»;

3) Y - множество всех слов, **НЕ** включая пустое, не содержащих в себе подстроку «mbd».

- P0 - множество всех слов, включая пустое, не содержащих в себе «m». P = [a-ln-z]\*;

- P — множество всех слов, **НЕ** включая пустое, не содержащих в себе «m». P = [a-ln-z][a-ln-z]\* = [a-ln-z]+;

- Q0 - множество всех слов, включая пустое, не содержащих в себе «m» и не начинающихся с «bd». Q0 = (([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*));

- Y = ((P0(m Q0)+)|(P(m Q0)\*)) - как уже писалось ранее, множество всех слов, **НЕ** включая пустое, не содержащих в себе подстроку «mbd»;

Реализация на Python (проверить можно на <https://regex101.com/>):

((([a-ln-z]\*(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*)|([a-ln-z]+(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*))[0-9]([a-ln-z]\*(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*)[0-9]([a-ln-z]\*(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*))|(([a-ln-z]\*(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*)[0-9](([a-ln-z]\*(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*)|([a-ln-z]+(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*))[0-9]([a-ln-z]\*(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*))|(([a-ln-z]\*(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*)[0-9]([a-ln-z]\*(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*)[0-9](([a-ln-z]\*(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*)|([a-ln-z]+(m(([ac-ln-z][a-ln-z]\*)|(b[a-ce-ln-z][a-ln-z]\*)|(b\*)))\*)))

Реализация на Python с использованием операции разности (?!):

1) X — искомое множество. X = (Y [0..9] Y0 [0..9] Y0) | (Y0 [0..9] Y [0..9] Y0) | (Y0 [0..9] Y0 [0..9] Y), где Y0 — множество всех слов, включая пустое, не содержащих в себе подстроку «mbd», Y - множество всех слов, **НЕ** включая пустое, не содержащих в себе подстроку «mbd»;

2) Y0 = (?:(?!mbd)[a-z])\* - множество всех слов, включая пустое, не содержащих mbd.

3) Y = (?:(?!mbd)[a-z])+ - множество всех слов, **НЕ** включая пустое, не содержащих mbd.

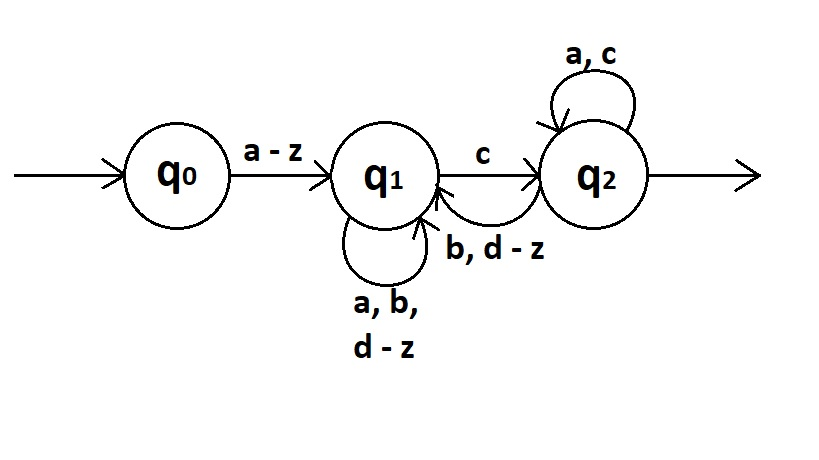
Реализация на Python (проверить можно на <https://regex101.com/>):

(((?:(?!mbd)[a-z])+[0-9](?:(?!mbd)[a-z])\*[0-9](?:(?!mbd)[a-z])\*)|((?:(?!mbd)[a-z])\*[0-9](?:(?!mbd)[a-z])+[0-9](?:(?!mbd)[a-z])\*)|((?:(?!mbd)[a-z])\*[0-9](?:(?!mbd)[a-z])\*[0-9](?:(?!mbd)[a-z])+))

Приложение 3. Список языков заданных регулярными выражениями.

Задание-11.

L - множество слов E+ca\*; Слово caa. E=[a-z]

Пояснение, почему данные автомат коректен:

E+ca\* = EE\*ca\*

1) Из q0 в q1 мы попадаем, если встречаем любой символ из нашего алфавита E=[a-z].

2) Мы находимся в q1 до тех пор пока не встретим ‘c’. Как только мы встречаем ‘c’, мы переходим в q2.

3) В q2 мы крутимся до тех пор, пока не встретим что-то отличное от ‘a’, ‘c’. Как только мы встречаем, что-то не из этих 2 символов, мы снова переходим в q1. Это “валидно”, так как E включает в себя все символы, и мы просто оказались в E.