

(S181 HU#1

1)  $X = \{w, x, y, z\}$

$B = \{0, 1\}$

a)  $(B \times B) \times B$

$B \times B = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$

$(B \times B) \times B = \{(0, 0, 0), (0, 1, 0), (1, 0, 0), (1, 1, 0), (0, 0, 1), (0, 1, 1), (1, 0, 1), (1, 1, 1)\}$

b)  $B \times (B \times B)$

$B \times B = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$

$B \times (B \times B) = \{(0, (0, 0)), (0, (0, 1)), (0, (1, 0)), (0, (1, 1)), (1, (0, 0)), (1, (0, 1)), (1, (1, 0)), (1, (1, 1))\}$

c)  $B \times B \times B$

$B \times B \times B = \{(0, 0, 0), (0, 1, 0), (1, 0, 0), (1, 1, 0), (0, 0, 1), (0, 1, 1), (1, 0, 1), (1, 1, 1)\}$

d)  $|P(N)| = 2^{|N|}$

$|X \times X| = |X| \cdot |X| = 4 \cdot 4$

$|X \times X| = 16$

$|P(X \times X)| = 2^{16} = 65536$

e)  $\{\epsilon, a, ac\} \times \{a, c, aa\}$

$= \{(\epsilon, a), (\epsilon, c), (\epsilon, aa), (a, a), (a, c), (a, aa), (ac, a), (ac, c), (ac, aa)\}$

2)  $\Sigma = \{a, b, c\}$

$L_2 = \text{language over } \Sigma$

a)  $\{\epsilon, a, ac\} \cdot \{a, c, aa\} \rightarrow \epsilon \text{ is concat's identity}$

$= \{a, c, aa, ac, aaa, aca, acc, acaa\}$

b)  $L_2^+ \cdot \{\epsilon\}$

$\emptyset$

c)  $\{\epsilon\} \times \Sigma$

$\{\epsilon\} \times \{a, b, c\}$

$\{(\epsilon, a), (\epsilon, b), (\epsilon, c)\}$



d)  $\{\epsilon\} \times L_2^+$

$\emptyset$

e)  $\{\epsilon\} \cdot L_2^+ \rightarrow \epsilon$  is wrong's identity

$L_2^+$ , no it doesn't contain the empty string, as the Kleene operation on a language excludes the empty string

3)  $\Sigma' = \{0, 1\}$

The language  $L_3$  accepts any odd-length strings beginning with 1 and any even-length strings beginning in 0, with the exception of the empty string, which is rejected.

4) a)  $\Sigma' = \{0, 1, =, +\}$

$L_{4a} = \{w \in \Sigma'^+ \mid w \text{ is of the form } x+y=z, \text{ where } x, y, z \in \{0, 1\}^+\}$

$\hookrightarrow$  correct binary sum

This is a non-FSL language, because, although it would be possible to implement a DFA to check for formatting of strings, FSLs do not have the capability to check if the binary sum is actually correct.

b)  $\Sigma' = \{a, b\}$

$L_{4b} = \{w \in \Sigma'^+ \mid \text{all runs of } a\text{'s in } w \text{ are of even length, and all runs of } b\text{'s in } w \text{ are of odd length}\}$

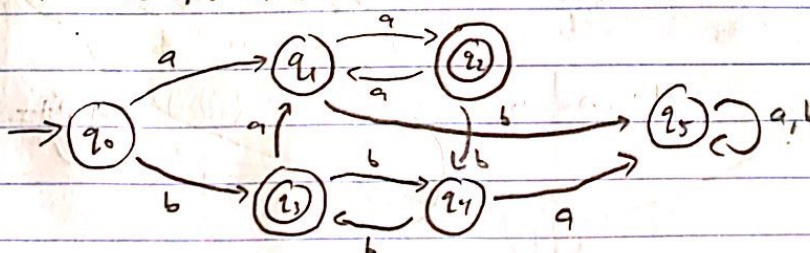
$Q \rightarrow$  states

$\Sigma \rightarrow$  alphabet

$\delta \rightarrow$  transition function

$q_0 \rightarrow$  start state

$F \rightarrow$  accept states





|   |       |       |
|---|-------|-------|
| $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$ |       |       |
| $\Sigma' = \{a, b\}$                        |       |       |
| $\delta =$                                  | a     | b     |
| $q_0$                                       | $q_1$ | $q_3$ |
| $q_1$                                       | $q_2$ | $q_5$ |
| $q_2$                                       | $q_1$ | $q_4$ |
| $q_3$                                       | $q_1$ | $q_4$ |
| $q_4$                                       | $q_5$ | $q_3$ |
| $q_5$                                       | $q_5$ | $q_5$ |
| $q_0 = q_0$                                 |       |       |
| $F = \{q_2, q_3\}$                          |       |       |

Since the language is defined by runs of a and b, it would be trivial or ambiguous to include the  $\epsilon$  in the language, as a run is defined as having 1+ characters.

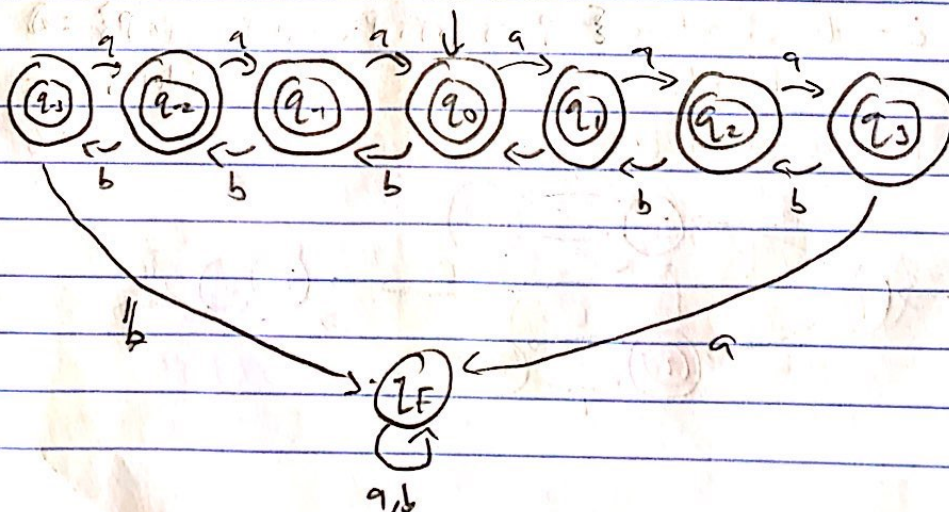
c)  $\Sigma' = \{a, b\}$

$L_{nc} = \{w \in \Sigma^+ \mid |\#(a, w) - \#(b, w)| < 4\}$

This is not possible to implement as a DFA, as there could be an indeterminate amount of 1 letter before a single occurrence of the other, resulting in a need for an indeterminate number of states to keep count of that letter.

d)  $\Sigma' = \{a, b\}$

$L_{ns} = \{w \in \Sigma^+ \mid |\#(a, y) - \#(b, y)| < 4, \text{ for all prefixes } y \text{ of } w\}$





|   |          |          |
|---|----------|----------|
| $Q = \{q_{-3}, q_{-2}, q_{-1}, q_0, q_1, q_2, q_3, q_F\}$ |          |          |
| $\Sigma = \{a, b\}$                                       |          |          |
| $\delta =$  | $q$      | $b$      |
| $q_{-3}$  | $q_{-2}$ | $q_F$    |
| $q_{-2}$  | $q_{-1}$ | $q_{-3}$ |
| $q_{-1}$  | $q_0$    | $q_{-2}$ |
| $q_0$   | $q_1$    | $q_{-1}$ |
| $q_1$   | $q_2$    | $q_0$    |
| $q_2$   | $q_3$    | $q_1$    |
| $q_3$   | $q_F$    | $q_2$    |
| $q_F$   | $q_F$    | $q_F$    |
| $q_0 = q_0$   |          |          |
| $F = \{q_{-3}, q_{-2}, q_{-1}, q_0, q_1, q_2, q_3\}$      |          |          |

5) Base case: Let  $w = \epsilon$

$$|w| = |w| = 0$$

$$\epsilon = xy$$

$$x = \epsilon, y = \epsilon$$

$$w^R = y^R x^R$$

$$\epsilon^R = \epsilon^R \epsilon^R$$

$$\epsilon = \epsilon \epsilon$$

$$\epsilon = \epsilon \checkmark \rightarrow \text{Base case solved}$$

Inductive Step: Assume:  $w = xy$  and  $(xy)^R = y^R x^R$  for all  $|w| \leq n$

Prove: the same holds for all  $|w| = n+1$

$$w' = xy' \text{, where } |y'| = |y| + 1 \rightarrow \text{prove } w'^R = y'^R x^R$$

$$w'^R = (xy')^R$$

$$w'^R = ((xy)c)^R \text{, where } c \in \Sigma$$

$$w'^R = c(xy)^R \text{, by definition of reverse}$$

$$w'^R = c(y^R x^R) \text{, by inductive hypothesis}$$

$$w'^R = (cy^R)x^R$$

$$w'^R = y'^R x^R \checkmark \rightarrow \text{Inductive step solved}$$



6) All sections are numbered from 0-10, with subsections numbered  $S.x$ , where  $S$  is the section and  $x$  is the subsection number (counting from 1). All figures/examples/theorems/etc. are numbered as  $S.y$ , where  $S$  is the section they're in, and  $y$  is the figure/example/theorem/etc. number (using the same counter and starting from 1). Finally, exercises are numbered  $S.z$ , where  $S$  is the section number and  $z$  is the exercise number (counting from 1). Some problems have subproblems, denoted by  $a), b), c),$  etc.