Q1.

For each statement below, state if it is true or false, and explain why. The explanation does not need to be a formal proof, but the argument should be sound.

> ### Statement a
>
> If L_1 is regular and $|L_1| = k$ and L_2 is non-regular, then $L_1 \cap L_2$ is regular.

This statement is **false**.

All finite languages are regular. $|L_1| = k$ implies that L_1 is finite, and therefore regular. The intersection of a regular language and a non-regular language is not guaranteed to be regular.

Note that all string under $L_1 \cap L_2$ must be a subset of L_1, and a subset of a finite language is finite, therefore regular.

For example, let L_1 be a regular language that contains a string $a^n b^n$ and $L_2 = \{a^n b^n\}$.

The intersection of $L_1 \cap L_2$ is non-regular.

> ### Statement b

If L_1 and L_2 are non-regular, then $L_1 \cup L_2$ is regular.

This statement is **false**.

The union of a regular and a non-regular language is not guaranteed to be regular.

A language is regular if there is an finite automaton that accepts it.

Note that L_1 is a regular language, therefore finite, and L_2 is non-regular, therefore there does not exist a finite automaton that accepts it.

If $L_1 \cup L_2$ is regular, then there must exist a finite automaton that accepts it. However, such automaton would also accept L_2 since $L_2 \subseteq L_1 \cup L_2$, therefore meet contradiction.

Which renders the statement **false**.

Statement c

$\forall L_1 \mid L_1$:non-regular, $\exists L_2 \mid L_2$:regular $\wedge L_1 \subseteq L_2$

This statement is **true**.

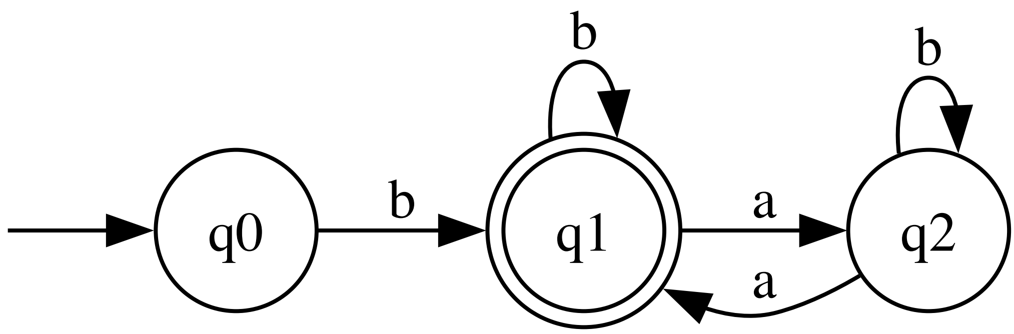
Let Σ be the alphabet of L_1., choose $L_2 = \Sigma^*$, which is regular.

Since Σ^* is the set of all strings formed from Σ plus empty string, it is guaranteed to contain L_1. Therefore $L_1 \subseteq \Sigma^*$. Therefore, $L_1 \subseteq L_2$

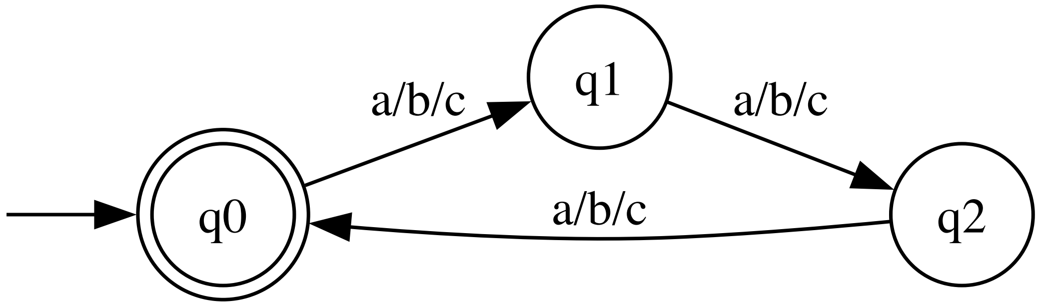
Q2.

Create a DFA M such that:

> **Statement a**
>
> M accepts all strings which begin with b but do not contain the substring bab.

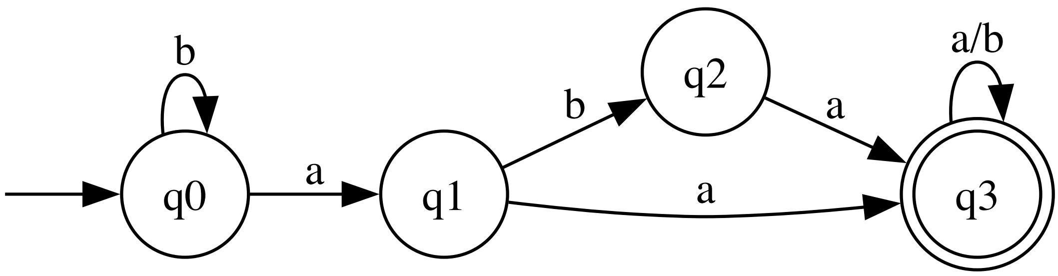


> **Statement b**
>
> $\mathcal{L}(M) = \{a^i b^j c^k \mid i + j + k \text{ is a multiple of } 3\}$,
> $\Sigma = \{a, b, c\}$



Statement c

$$\mathcal{L}(M) = \{x \mid \text{at least two a's in last three characters of x}\}$$



Q3.

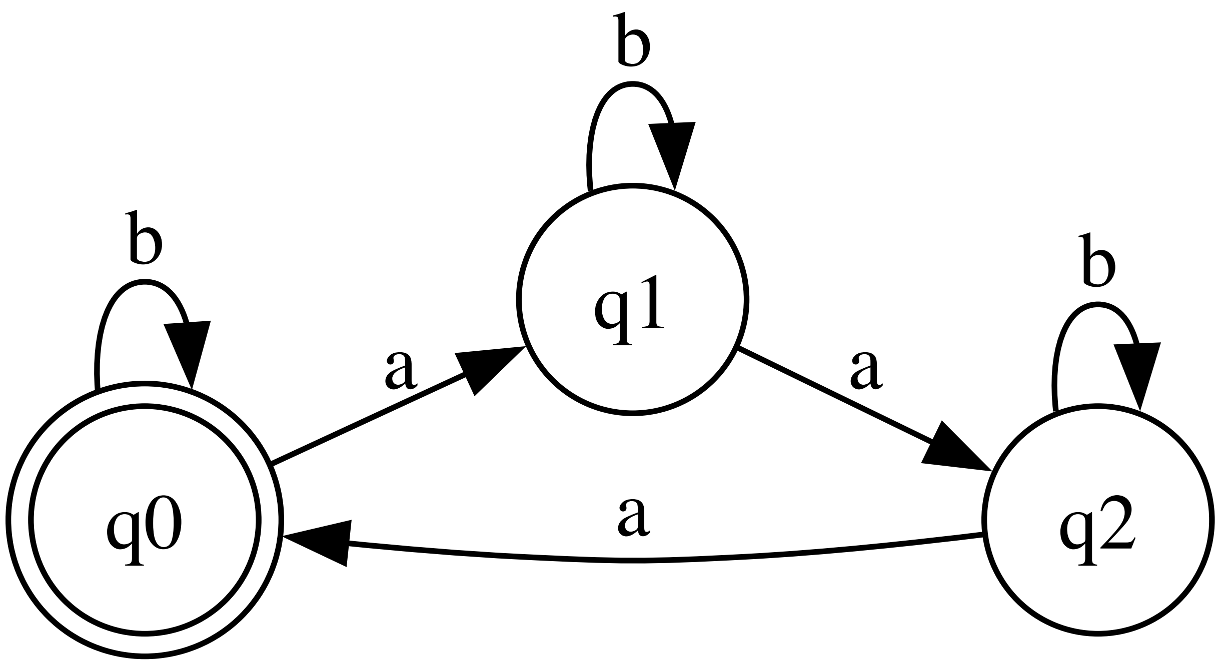
Via product construction, create a DFA M such that

$$\mathcal{L}(M) = \{a^n b^m \mid n \vee m \text{ is a multiple of } 3\}$$

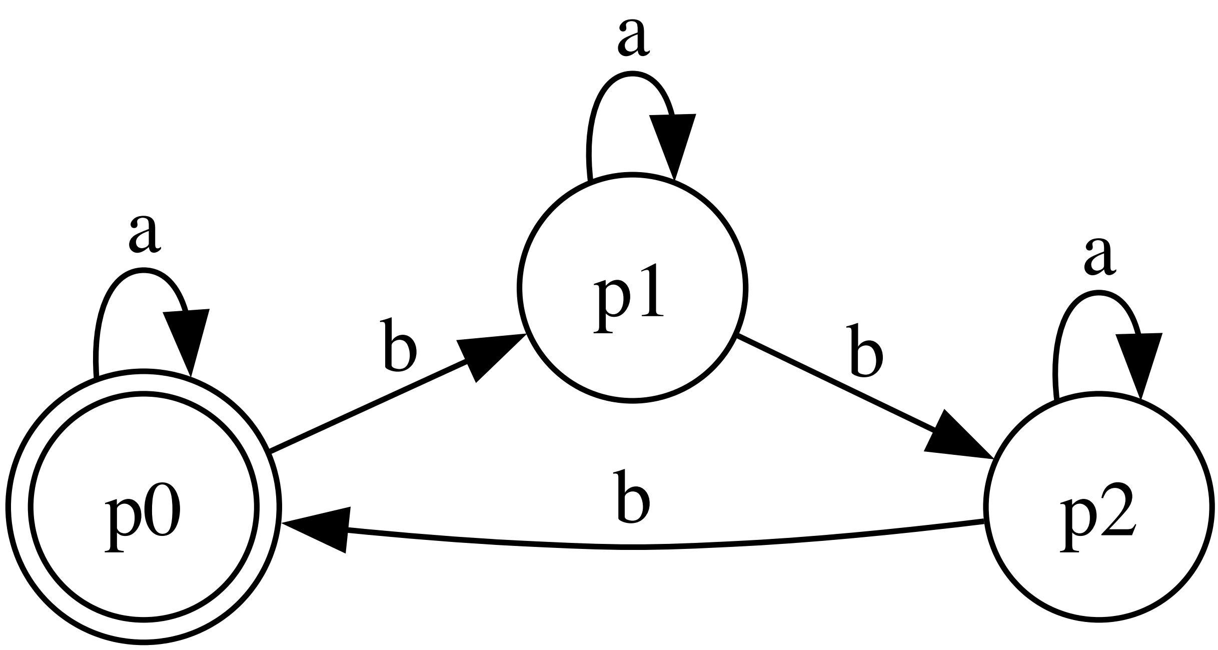
First create two machine: one where n is a multiple and one where m is a multiple of 3. Then create the "union" machine:

$$\mathcal{L}(M_1) = \{a^n b^m \mid n \text{ is a multiple of } 3$$
$$\mathcal{L}(M_2) = \{a^n b^m \mid m \text{ is a multiple of } 3$$

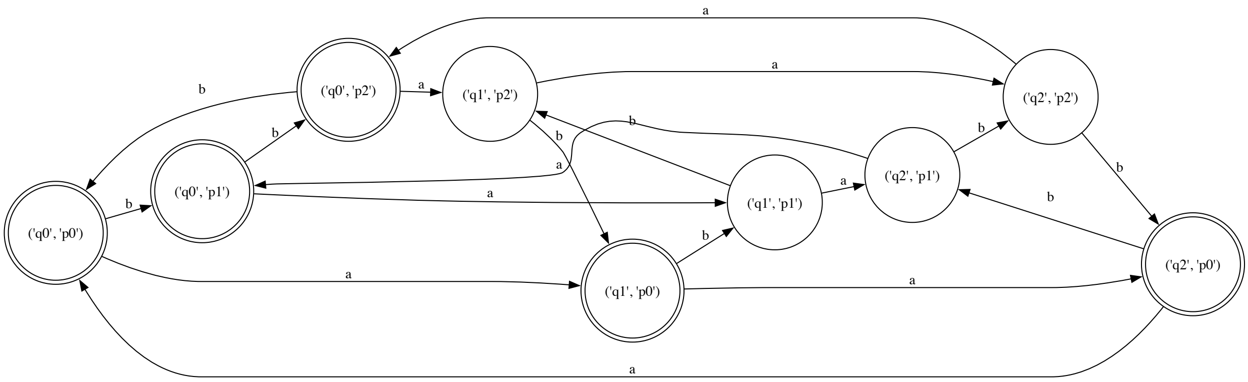
First, we will construct M_1:



Then, we will construct M_2:



From product construction, we will create M based on M_1 and M_2:



Q4.

Create an NFA which accepts all string in which the third last character is an a. Then via subset construction, create an equivalent DFA. Show all your work

Solution

We define the following NFA $(Q, \Sigma, \delta, q_0, F)$ with:

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{a, b\}$
- Start state q_0
- Accept state q_3
- Transition function δ as follows:

$$\delta(q_0, a) = \{q_0, q_1\}$$
$$\delta(q_0, b) = \{q_0\}$$
$$\delta(q_1, a) = \{q_2\}$$
$$\delta(q_1, b) = \{q_2\}$$
$$\delta(q_2, a) = \{q_3\}$$
$$\delta(q_2, b) = \{q_3\}$$
$$\delta(q_3, a) = \{q_4\}$$
$$\delta(q_3, b) = \{q_4\}$$
$$\delta(q_4, a) = \{q_4\}$$
$$\delta(q_4, b) = \{q_4\}$$

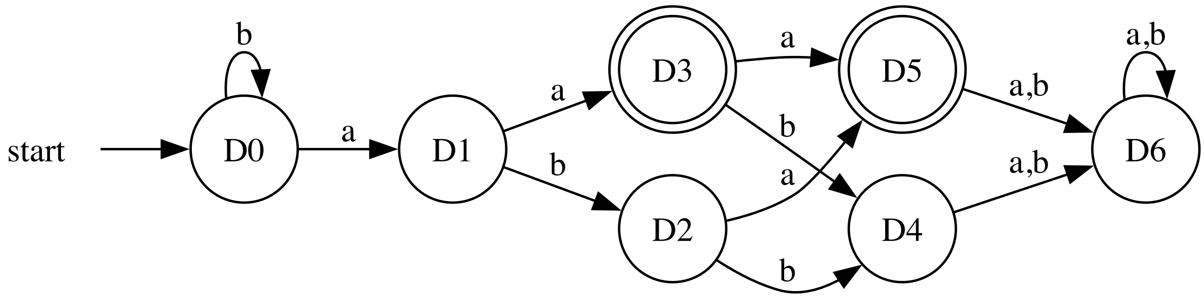
Via subset construction, we can create the following DFA:

Start state of DFA is $\{q_0\}$, as it is the epsilon closure of the start state of the NFA

Transition table:

DFA state	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_3\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_2, q_3\}$
$\{q_0, q_3\}$	$\{q_4\}$	$\{q_4\}$
$\{q_0, q_1, q_2, q_3\}$	$\{q_4\}$	$\{q_4\}$
$\{q_4\}$	$\{q_4\}$	$\{q_4\}$

The final state are any states that include q_3, which are $\{q_0, q_1, q_2, q_3\}$ and $\{q_0, q_3\}$.



Where

```
dfa_states = {
    'D0': '{q0}',
    'D1': '{q0, q1}',
    'D2': '{q0, q2}',
    'D3': '{q0, q1, q2}',
    'D4': '{q0, q3}',
    'D5': '{q0, q1, q2, q3}',
    'D6': '{q4}'
}
```