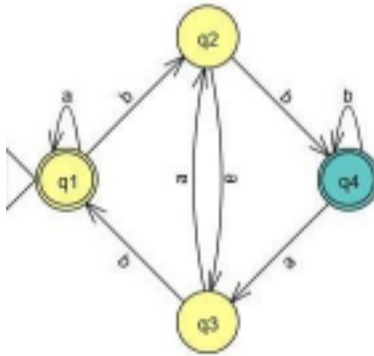


## CSE 355: Intro to Theoretical Computer Science

### Recitation #2 (20 pts)

• Due: **Tuesday, Jan. 26, 2021 at 11:59pm** Arizona time on Canvas.

1. [5 pts] For the following DFA, answer questions.



1) Which state is the initial/start state?

Answer: q1

2) Which state(s) is(are) accepting state(s)?

Answer: q1 and q4

3) Write state sequence for input string *aabb*, does the DFA accept string *aabb*?

Answer: q1(start) → q1(a, stays) → q1(a, stays) → q2(b) → q4(b)

Yes, the DFA accept string *aabb*.

4) Write state sequence for input string *aabaaba*, does the DFA accept string *aabaaba*?

Answer: q1(start) → q1(a, stays) → q1(a, stays) → q2(b) → q3(a) → q2(a) → q4(b) → q3(a)

No, the DFA does not accept string *aabaaba*, since it is not in the accepting stage(q1 or q4)

5) Does above DFA accept the empty string  $\epsilon$ ?

Answer: q1(start). Yes, the DFA accept empty string since it starts at stage q1

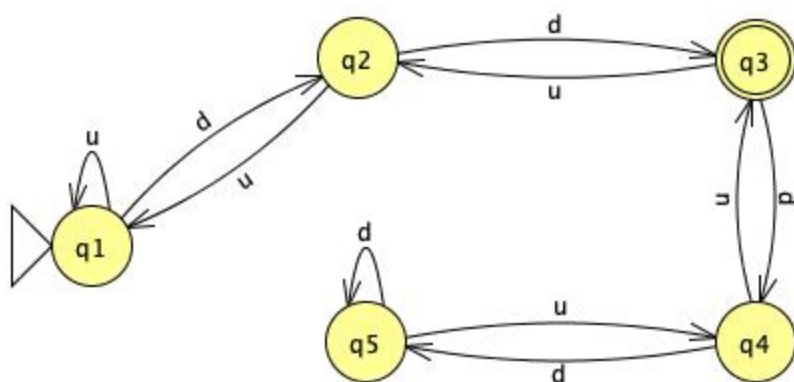
and it is a accepting stage

2. [2 pts] Given the formal description of a DFA  $M$  defined as below, using [JFLAP](http://www.jflap.org) (<http://www.jflap.org>) to draw its state diagram (save it as .jpg file and paste it here)

$$M = (\{q1, q2, q3, q4, q5\}, \{u, d\}, \delta, q1, \{q3\})$$

where the transition function  $\delta$  is defined as:

$\delta$	$u$	$d$
q1	q1	q2
q2	q1	q3
q3	q2	q4
q4	q3	q5
q5	q4	q5



1

3. [5 pts] For the following language  $L$ , assume alphabet  $\Sigma = \{0, 1\}$ . Use [JFLAP](http://www.jflap.org) (<http://www.jflap.org>) to draw its state diagram (save it as .jpg file and paste it here).

$L = \{\omega \mid \omega \text{ begins with a 1 and ends with a 0}\}$

```

graph LR
    start(( )) --> q0((q0))
    q0 -- 0 --> q1((q1))
    q1 -- 1 --> q0
    q0 -- 1 --> q2((q2))
    q2 -- 0 --> q1
    q2 -- 0 --> q3(((q3)))
    q3 -- 1 --> q2
    q1 -- 0 --> q1
    q2 -- 1 --> q2
    q3 -- 0 --> q3
  
```

Table Text Size

Slider

Input	Result
11001	Reject
11000	Accept
0111	Reject
0110	Reject

Load Inputs
Run Inputs
Clear
Enter Lambda
View Trace

4. [4 pts] The following language is the intersection of two simpler languages. First construct DFAs for the simpler languages, then combine them using the product construction described in Theorem 1.25 (textbook pp.66) to draw the state diagram (Use JFLAP) for the language given. Assume alphabet  $\Sigma = \{a, b\}$ .

$L = \{\omega \mid \omega \text{ has even length and an odd number of } a\text{'s}\}$

```

graph LR
    start(( )) --> q0((q0))
    q0 -- a --> q1(((q1)))
    q1 -- b --> q0
    q0 -- b --> q1
    q1 -- a --> q0
  
```

Table Text Size

Input	Result
a	Reject
b	Reject
aa	Accept
bb	Accept
aba	Reject
bab	Reject
aaa	Accept
bbb	Accept

||

Load Inputs
Run Inputs
Clear
Enter Lambda
View Trace

```

graph LR
    start(( )) --> q0((q0))
    q0 -- a --> q1(((q1)))
    q1 -- a --> q0
    q0 -- b --> q0
    q1 -- b --> q1
  
```

Table Text Size

Input	Result
a	Accept
b	Reject
aa	Reject
bb	Reject
aba	Reject
bab	Accept
aaa	Reject
bbb	Reject
aaa	Accept
ababa	Accept
baaaa	Accept
baabb	Reject

||

Load Inputs
Run Inputs
Clear
Enter Lambda
View Trace

Table Text Size

Input	Result
aaaa	Reject
bbbb	Reject
ab	Accept
abaa	Accept
abbb	Accept
baab	Reject
aba	Reject
aaa	Reject
bbb	Reject
aabbba	Accept
ababababa	Reject
<b>baaaabbb</b>	<b>Reject</b>

||
Load Inputs
Run Inputs
Clear
Enter Lambda
View Trace

5. [4 pts] The following language is the complement of a simpler language. First construct DFA for the simpler languages, then use it to give the state diagram of the language (use JFLAP). Assume alphabet  $\Sigma = \{a, b\}$ .

$$L = \{\omega \mid \omega \text{ does not contain the substring } ab\}$$

**Image is on next page**

(contains *ab*)

Table Text Size

Input	Result
abba	Accept
bbaa	Reject
baba	Accept
bbab	Accept
aaab	Accept
aaaa	Reject
bbbb	Reject
baba	Accept

Load Inputs Run Inputs Clear Enter Lambda View Trace

Ans : (Does not contain *ab*)

Table Text Size

Input	Result
abba	Reject
bbaa	Accept
baba	Reject
bbab	Reject
aaab	Reject
aaaa	Accept
bbbb	Accept
baba	Reject

Load Inputs Run Inputs Clear Enter Lambda View Trace