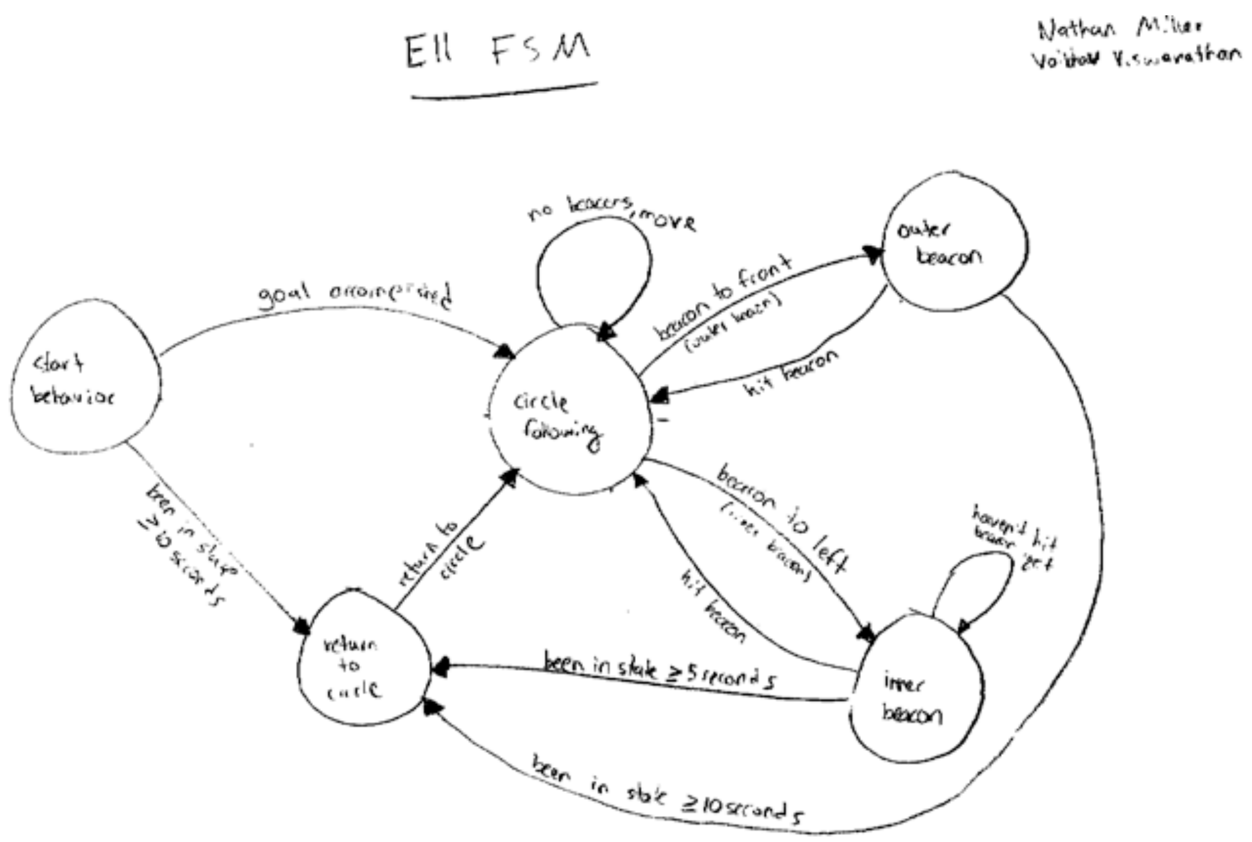
DFAs are useful—they're everywhere!



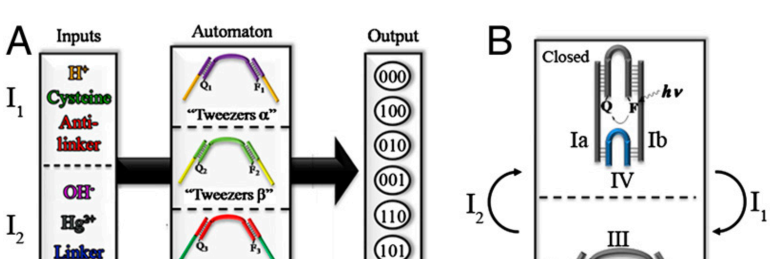
All-DNA finite-state automata with finite memory

Zhen-Gang Wang[a,1], Johann Elbaz[a,1], F. Remacle[b], R. D. Levine[a,c,2], and Itamar Willner[a,2]

[a]Institute of Chemistry, Hebrew University of Jerusalem, Jerusalem 91904, Israel; [b]Chemistry Department, B6c, University of Liège, 4000 Liège, Belgium; and [c]Department of Chemistry and Biochemistry, Crump Institute for Molecular Imaging, and Department of Molecular and Medical Pharmacology, University of California, Los Angeles, CA 90095

Contributed by Raphael D. Levine, October 25, 2010 (sent for review August 6, 2010)

Biomolecular logic devices can be applied for sensing and nano-medicine. We built three DNA tweezers that are activated by the inputs H^+/OH^-; Hg^{2+}/cysteine; nucleic acid linker/complementary antilinker to yield a 16-states finite-state automaton. The outputs of the automata are the configuration of the respective tweezers (opened or closed) determined by observing fluorescence



Draw the DFA for L = {`01`}

Full name

T. 9/11

Deterministic Finite Automaton (DFA)

Has a finite *alphabet* ($\boldsymbol{\Sigma}$) of valid symbols.

Has a finite set of states—one *initial state* and some *accepting states*.

Has a *transition function*.
Each configuration has exactly one transition. ← A DFA's "configuration" is its current state and current input symbol.

Each transition consumes one input symbol.

The machine can be in exactly one state at a time.

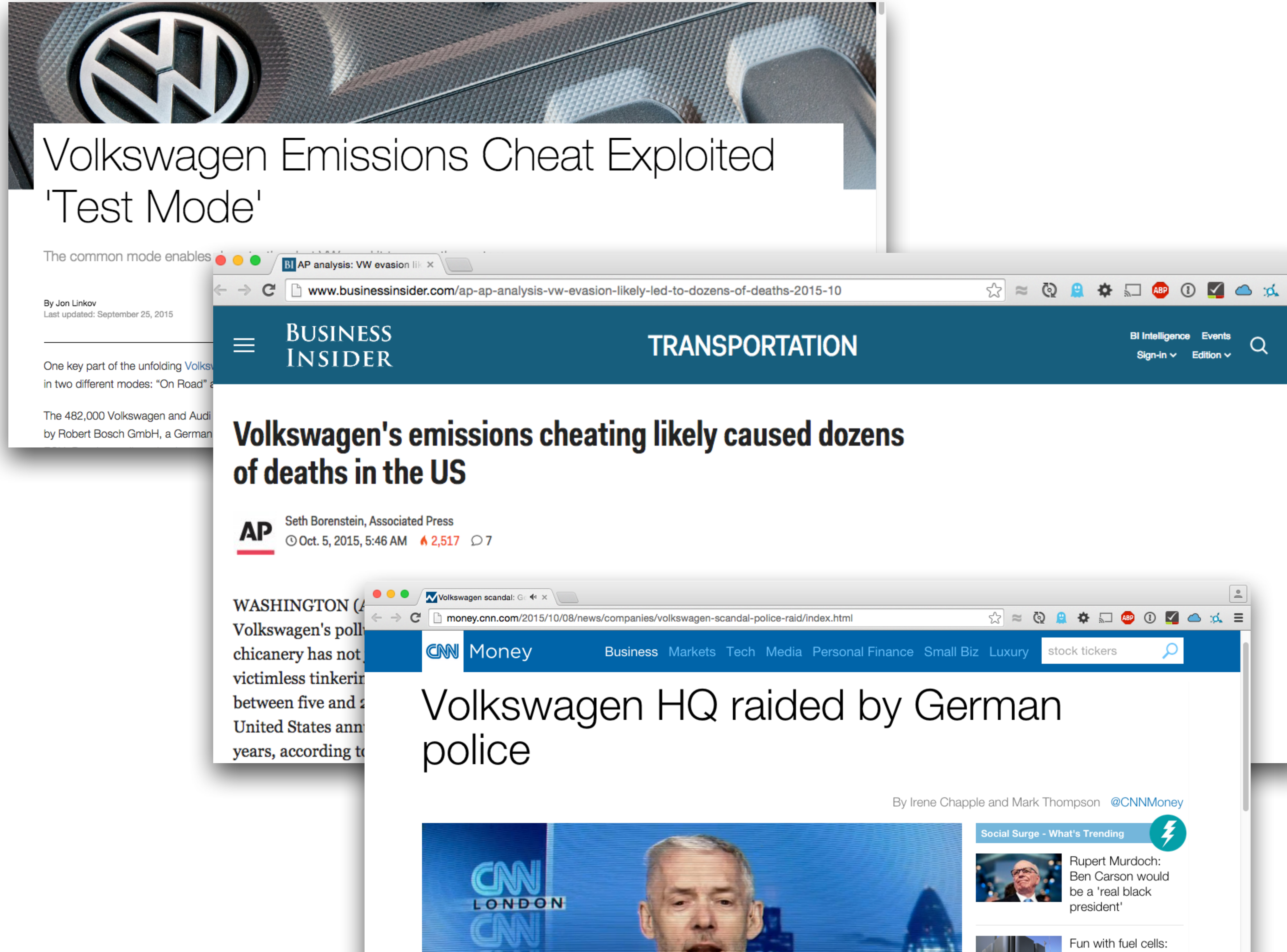
Given an input string, a DFA operates as follows:

The machine starts in the initial state.

The machine takes the only applicable transition (consuming an input symbol) until it has read and responded to the entire input string. When all the input is consumed, the machine halts.

The machine accepts only if it is in an accepting state.

What makes for a "good" DFA?



Volkswagen Emissions Cheat Exploited 'Test Mode'

The common mode enables

By Jon Linkov
Last updated: September 25, 2015

One key part of the unfolding Volksw
in two different modes: "On Road"

The 482,000 Volkswagen and Audi
by Robert Bosch GmbH, a German

AP analysis: VW evasion li ×

www.businessinsider.com/ap-ap-analysis-vw-evasion-likely-led-to-dozens-of-deaths-2015-10

BUSINESS INSIDER

TRANSPORTATION

BI Intelligence Events
Sign-in Edition

Volkswagen's emissions cheating likely caused dozens of deaths in the US

AP
Seth Borenstein, Associated Press
Oct. 5, 2015, 5:46 AM 2,517 7

WASHINGTON (A
Volkswagen's poll
chicanery has not
victimless tinkeri
between five and
United States ann
years, according t

Volkswagen scandal: G ×

money.cnn.com/2015/10/08/news/companies/volkswagen-scandal-police-raid/index.html

CNN Money Business Markets Tech Media Personal Finance Small Biz Luxury stock tickers

Volkswagen HQ raided by German police

By Irene Chapple and Mark Thompson @CNNMoney

Social Surge - What's Trending

Rupert Murdoch: Ben Carson would be a 'real black president'

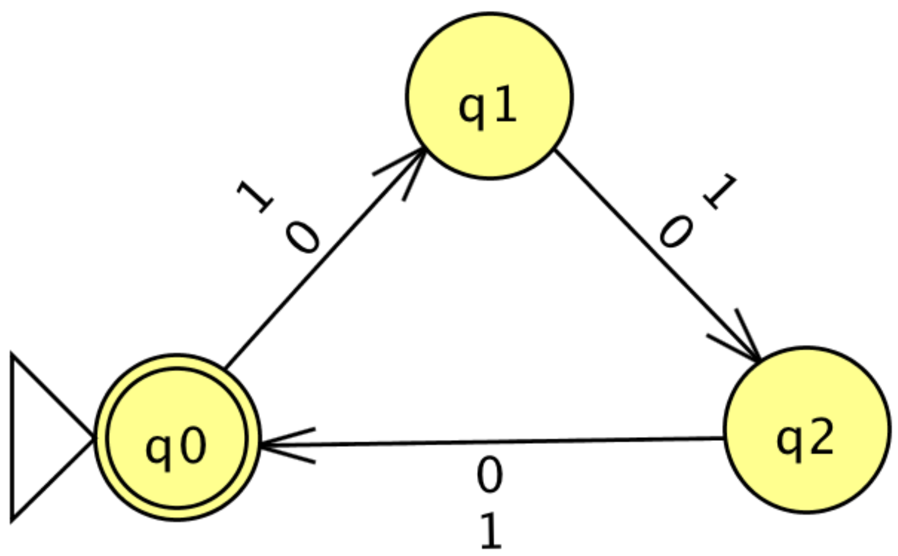
Fun with fuel cells:

Given a language L, can we *prove* that a DFA for L requires at least n states, for some n?

Let's practice!

Show that the DFA for the following language requires at least 3 states:

$$L = \{w \mid \text{the length of } w \text{ is divisible by 3}\}$$



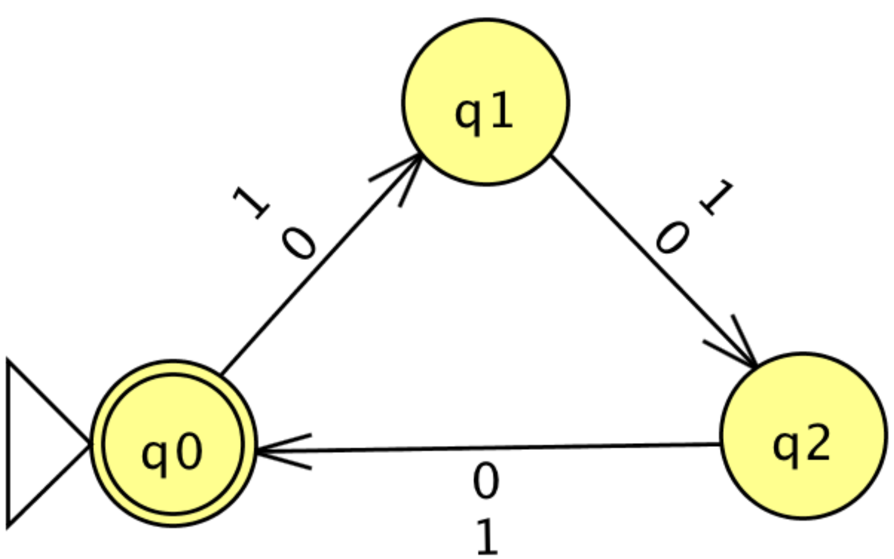
		w_1	w_2	w_3
		λ	**1**	**11**
w_1	**λ**	–	11	1
w_2	**1**	–	–	1
w_3	**11**	–	–	–

Distinguishability

of two strings

Two strings w_1, w_2 are **distinguishable** if there is some other string z such that $w_1 z \in \mathrm{L}$ and $w_2 z \notin L$.

"distinguishing extension"



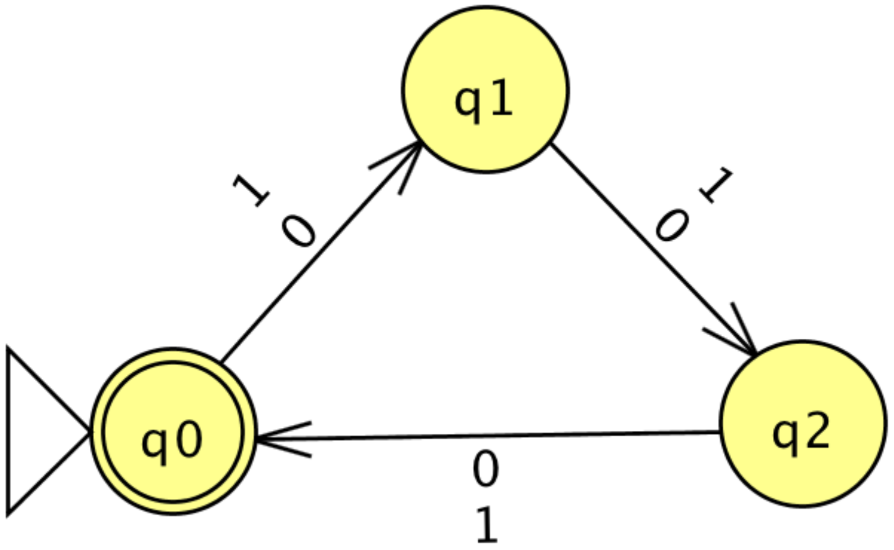
			w_1	w_2	w_3
			λ	**1**	**11**
w_1	**λ**		–	11	1
w_2	**1**		–	–	1
w_3	**11**		–	–	–

λ and 1 are distinguishable.

Pair-wise distinguishability

of a set of strings

A set of strings $S = \{w_1, w_2, \ldots, w_n\}$ is **pairwise distinguishable** for a language L if every pair of strings $w_i \neq w_j$ is distinguishable.

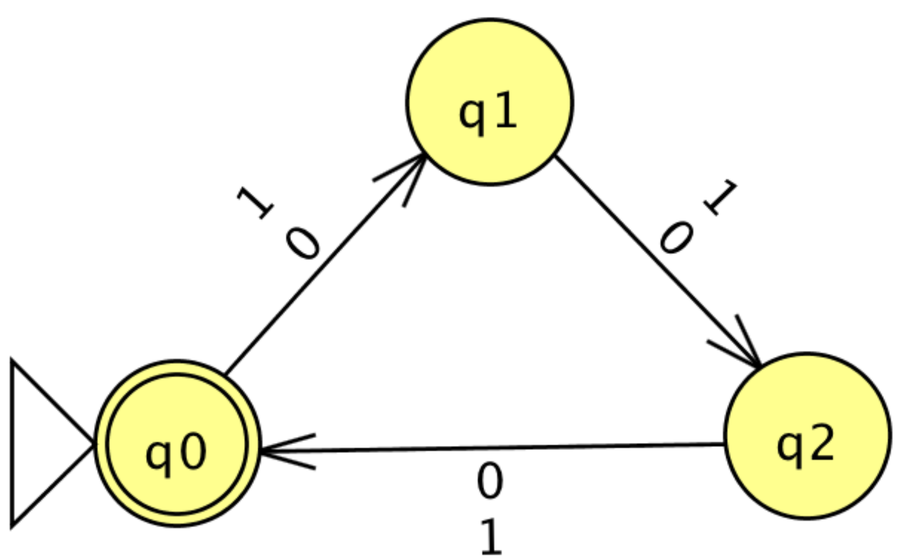


		w_1	w_2	w_3
		λ	**1**	**11**
w_1	**λ**	–	11	1
w_2	**1**	–	–	1
w_3	**11**	–	–	–

The set $\{\lambda, 1, 11\}$ is pairwise distinguishable.

Distinguishability theorem

If a set of strings $S = \{w_1, w_2, \ldots, w_n\}$ is **pairwise distinguishable** for a language L, then any DFA that accepts L must have at least n states.



		w_1	w_2	w_3
		λ	**1**	**11**
w_1	**λ**	–	11	1
w_2	**1**	–	–	1
w_3	**11**	–	–	–

$L = \{w \mid w\text{'s length is divisible by } 3\}$ has a DFA with at least 3 states.

Two unanswered questions

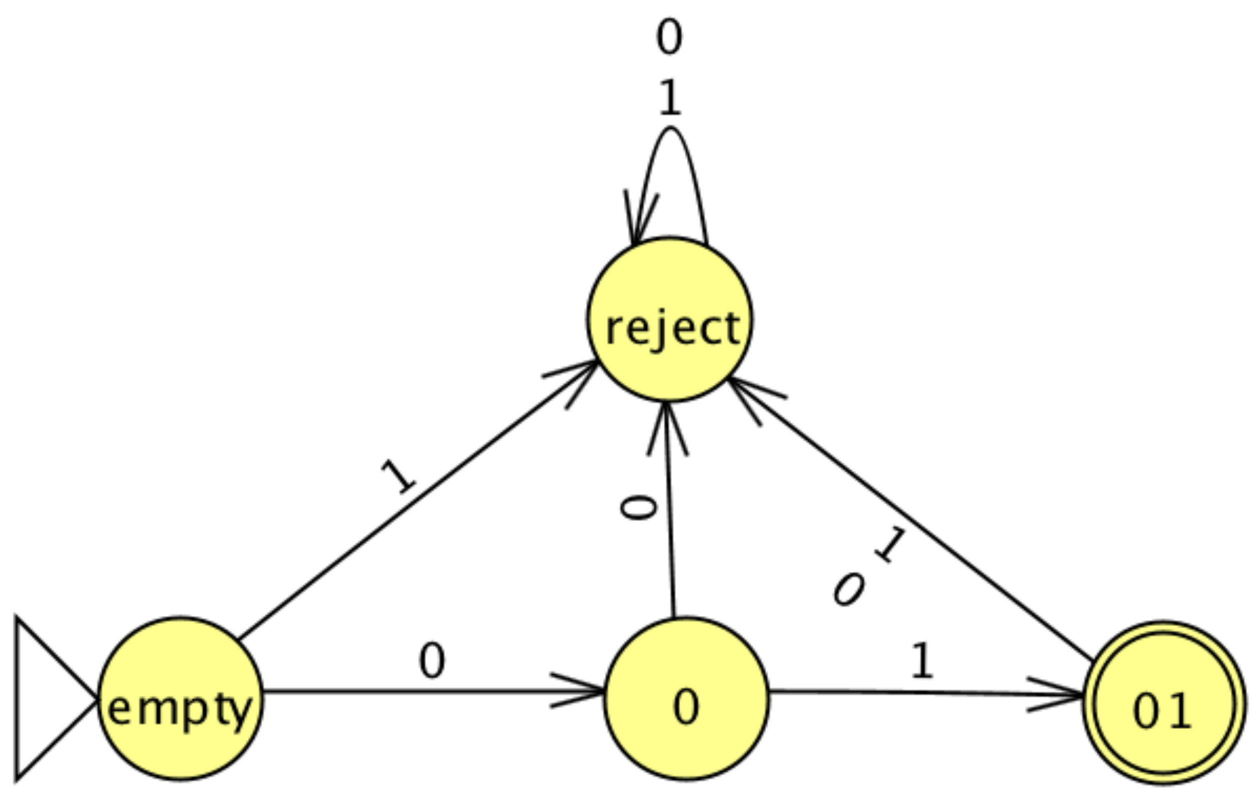
Given a language L…

What is the minimal number of states required for L’s DFA?

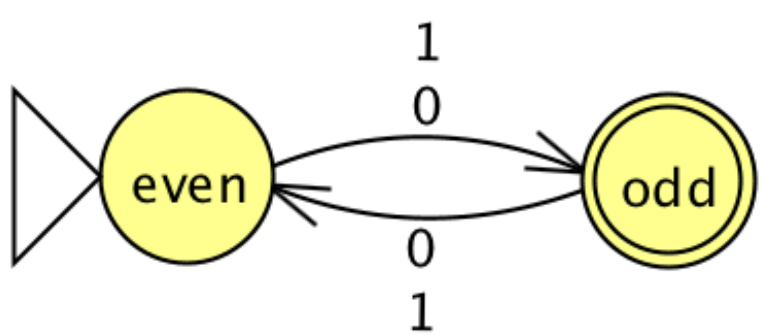
Given the minimal number of states, how do we construct a correct DFA?

Wouldn't this be great?

L = {w | w is `01` or the length of w is odd}



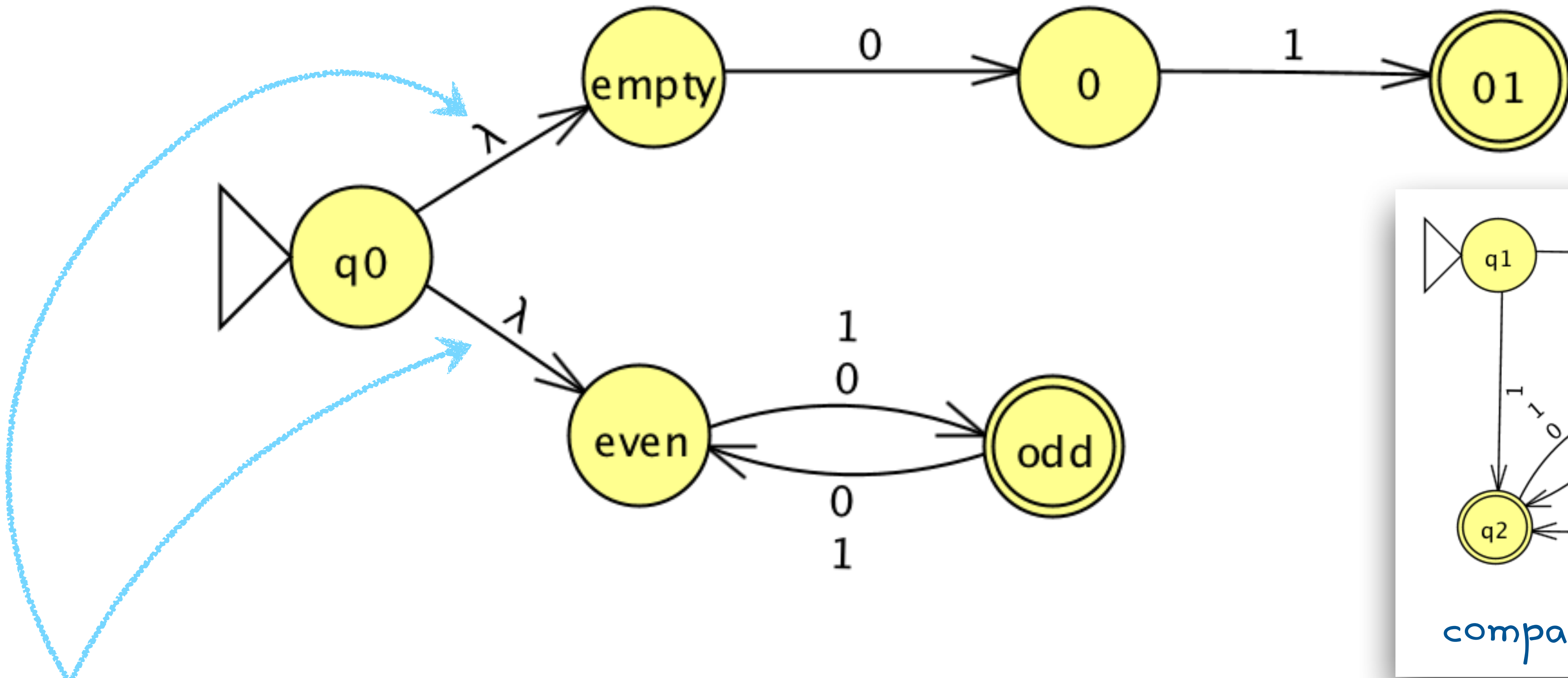
or

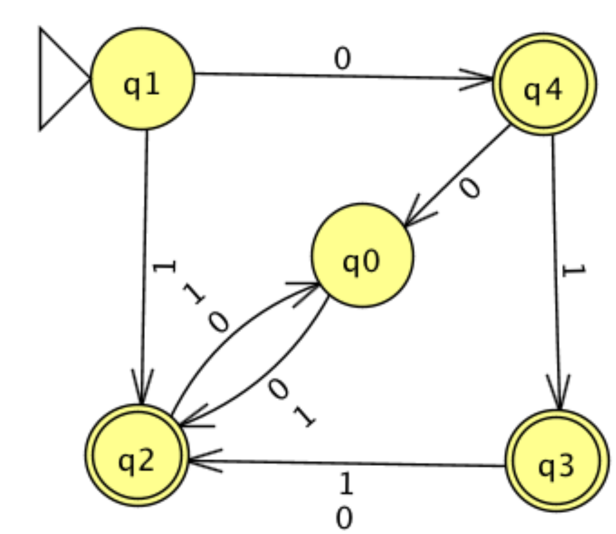


Wouldn't this be great?

$$L = \{w \mid w \text{ is } \texttt{010} \text{ or the length of } w \text{ is even}\}$$

Unspecified configurations transition to an implicitly defined, rejecting state.





compare to DFA

"λ transitions" don't consume input, they just change state.

NFAs can be in more than one state at the same time.

Nondeterministic Finite Automaton (NFA)

Has a finite *alphabet* ($\mathbf{\Sigma}$) of valid symbols.

Has a finite set of states—one *initial state* and some *accepting states*.

Has a *transition* ***relation***.

Each configuration corresponds to **zero or more** transitions.

***λ-transitions* change state but do not consume input.**

Unspecified configurations transition to an implicitly defined, rejecting state.

All other transitions consume one input symbol.

The machine can be in **multiple** states at one time.

Given an input string, a DFA operates as follows:

The machine starts in the initial state.

The machine takes **all applicable transitions** until it has read and responded to the entire input string. When all the input is consumed, the machine halts.

The machine accepts only if it is in **at least one** accepting state.

Draw these NFAs

(1) L = $\{w \mid w$ is `101` or w contains an odd number of `1`s$\}$

(2) L = $\{w \mid w$ starts with `1` and ends with either `00` or `01`$\}$

These are some examples you can try out.

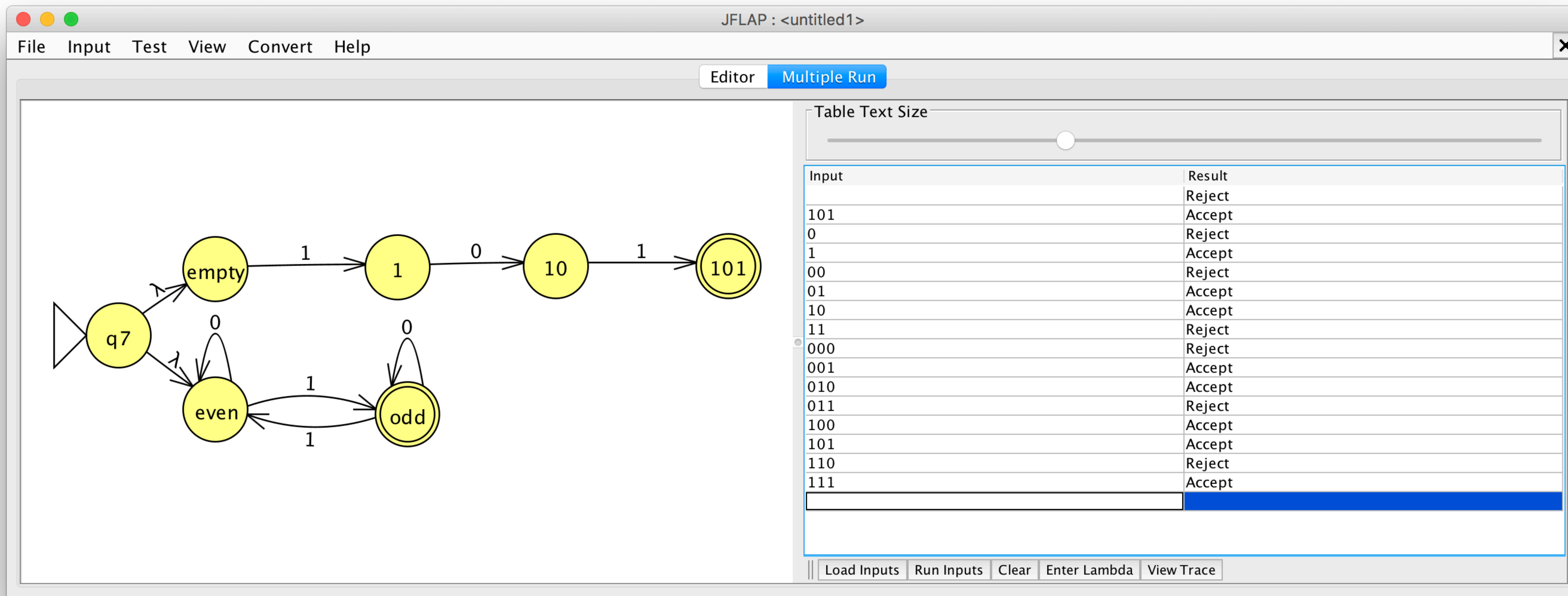
Write test cases first!!!!

At least three strings **accepted** by the NFA.

At least three strings **rejected** by the NFA.

Draw these NFAs

(1) L = {w | w is `101` or w contains an odd number of 1s}



Input	Result
	Reject
101	Accept
0	Reject
1	Accept
00	Reject
01	Accept
10	Accept
11	Reject
000	Reject
001	Accept
010	Accept
011	Reject
100	Accept
101	Accept
110	Reject
111	Accept

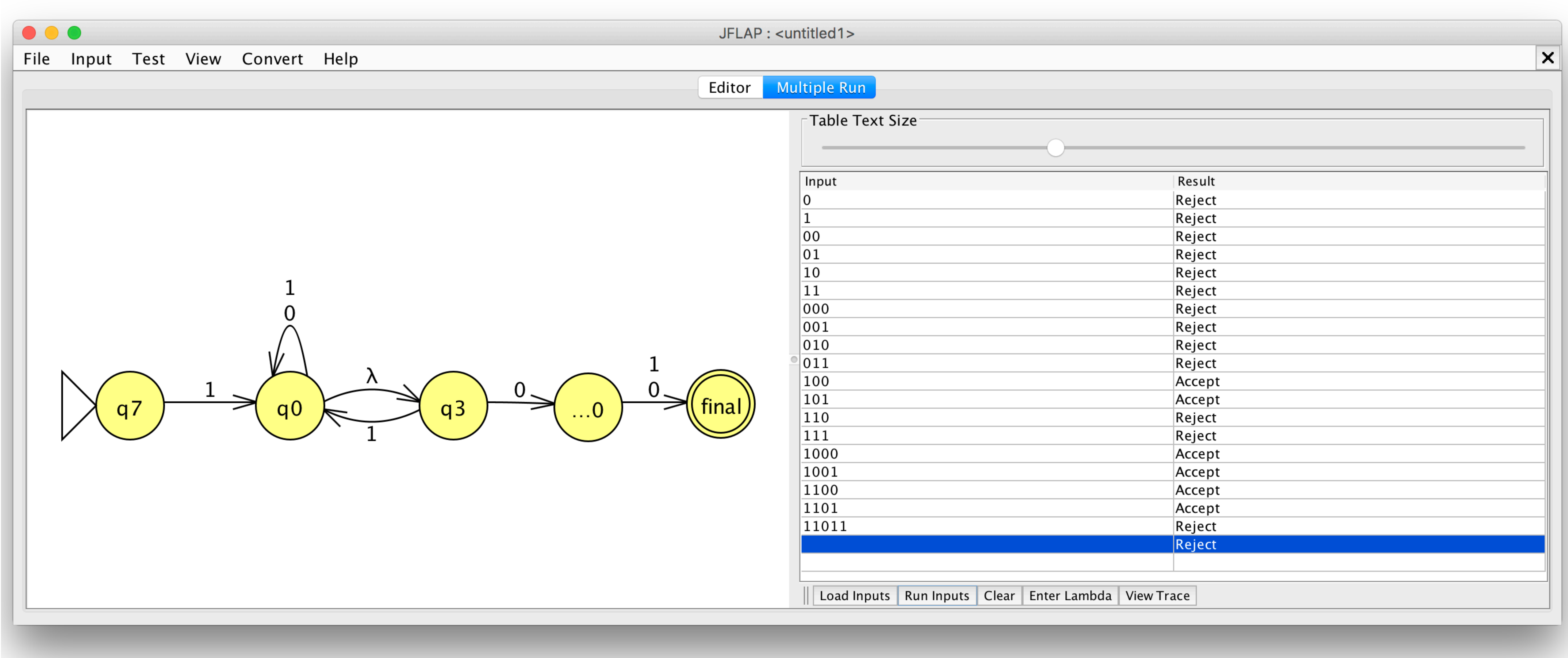
Write test cases first!!!!

At least three strings **accepted** by the NFA.

At least three strings **rejected** by the NFA.

Draw these NFAs

(2) L = {w | w starts with `1` and ends with either `00` or `01`}



Write test cases first!!!!

At least three strings **accepted** by the NFA.

At least three strings **rejected** by the NFA.