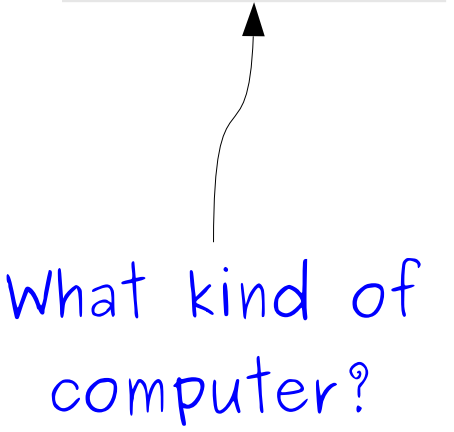
Finite Automata

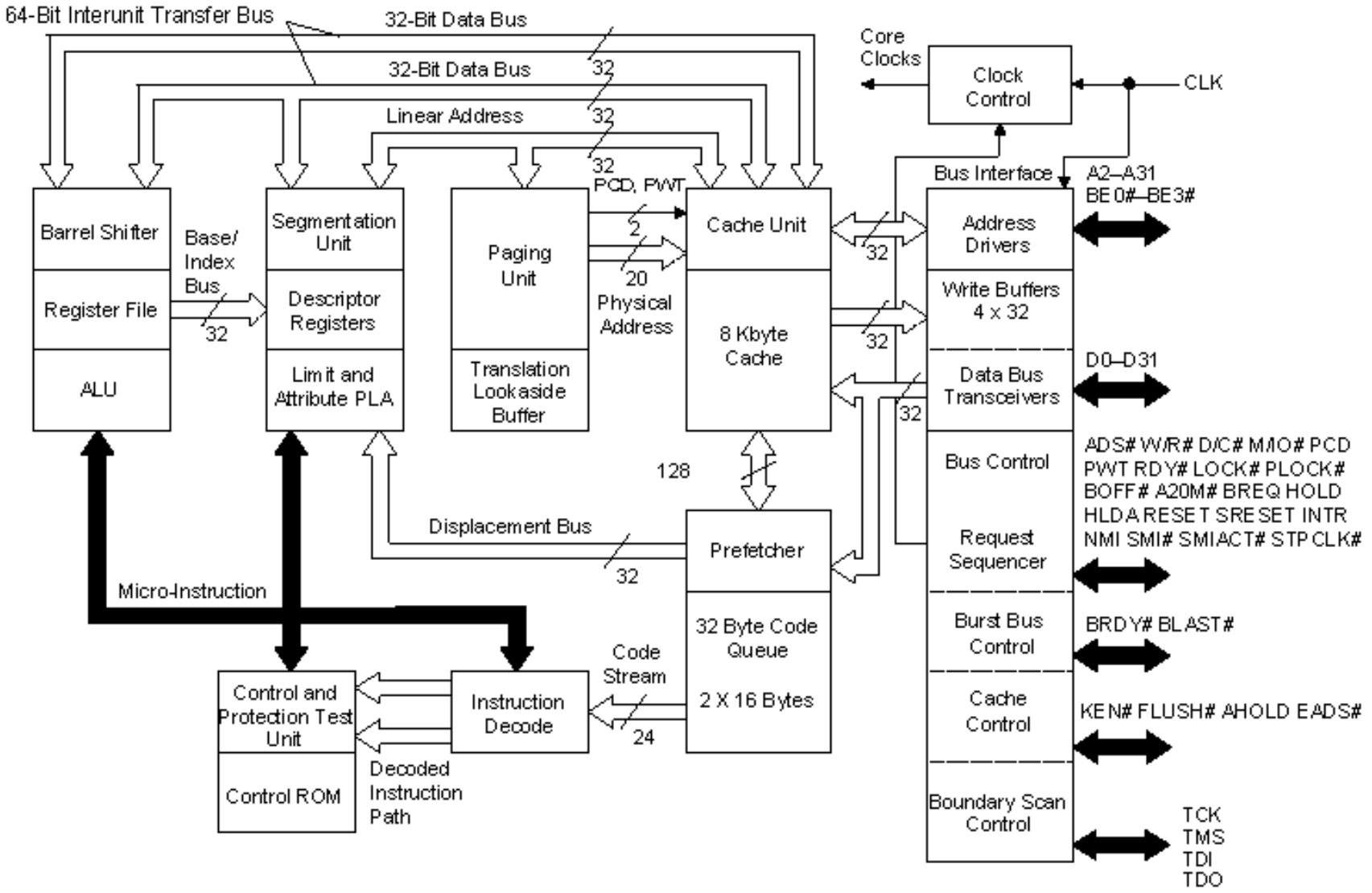
Part One

Computability Theory

What problems can we solve with a computer?



Computers are Messy



http://www.intel.com/design/intarch/prodbref/272713.htm

We need a simpler way of discussing computing machines.

An ***automaton*** (plural: **automata**) is a mathematical model of a computing device.

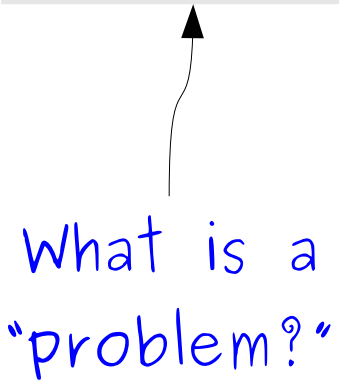
Why Build Models?

- **Mathematical simplicity.**
  - It is significantly easier to manipulate our abstract models of computers than it is to manipulate actual computers.
- **Intellectual robustness.**
  - If we pick our models correctly, we can make broad, sweeping claims about huge classes of real computers by arguing that they're just special cases of our more general models.

Why Build Models?

- The models of computation we will explore in this class correspond to different conceptions of what a computer could do.
- ***Finite automata*** (next two weeks) are an abstraction of computers with finite resource constraints.
  - Provide upper bounds for the computing machines that we can actually build.
- ***Turing machines*** (later) are an abstraction of computers with unbounded resources.
  - Provide upper bounds for what we could ever hope to accomplish.

What problems can we solve with a computer?



Problems with Problems

- Before we can talk about what problems we can solve, we need a formal definition of a “problem.”
- We want a definition that
  - corresponds to the problems we want to solve,
  - captures a large class of problems, and
  - is mathematically simple to reason about.
- No one definition has all three properties.

Formal Language Theory

Strings

- An ***alphabet*** is a finite set of symbols called ***characters***.
  - Typically, we use the symbol **Σ** to refer to an alphabet.
- A ***string over an alphabet Σ*** is a finite sequence of characters drawn from Σ.
- Example: If Σ = {**a**, **b**}, some valid strings over Σ include

  a

  aabaaabbabaaabaaaabbb

  abbababba

- The ***empty string*** contains no characters and is denoted **ε**.

Languages

- A ***formal language*** is a set of strings.
- We say that L is a ***language over Σ*** if it is a set of strings over Σ.
- Example: The language of palindromes over $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ is the set

  $\{\varepsilon, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{aa}, \mathbf{bb}, \mathbf{cc}, \mathbf{aaa}, \mathbf{aba}, \mathbf{aca}, \mathbf{bab}, \ldots\}$
- The set of all strings composed from letters in Σ is denoted **Σ***.
- Formally, we say that L is a language over Σ if $L \subseteq \Sigma^*$.

The Model

- ***Fundamental Question:*** Given an alphabet Σ and a language L over Σ, in what cases can we build an automaton that determines which strings are in L?
- The answer depends on both the choice of L and the choice of automaton.
- The entire rest of the quarter will be dedicated to answering these questions.

To Summarize

- An ***automaton*** is an idealized mathematical computing machine.
- A ***language*** is a set of strings.
- The automata we will study will accept as input a string and (attempt to) determine whether that string is contained in a particular language.

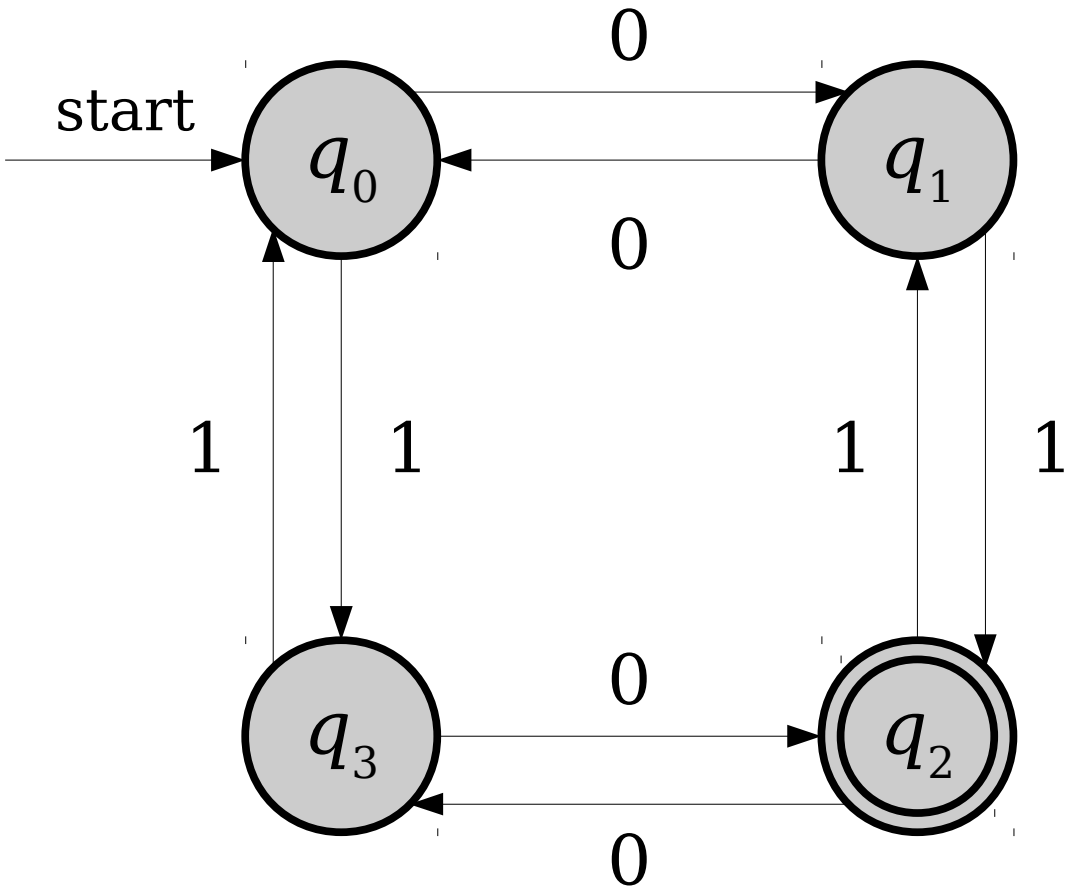
What problems can we solve with a computer?

Finite Automata

A ***finite automaton*** is a simple type of mathematical machine for determining whether a string is contained within some language.

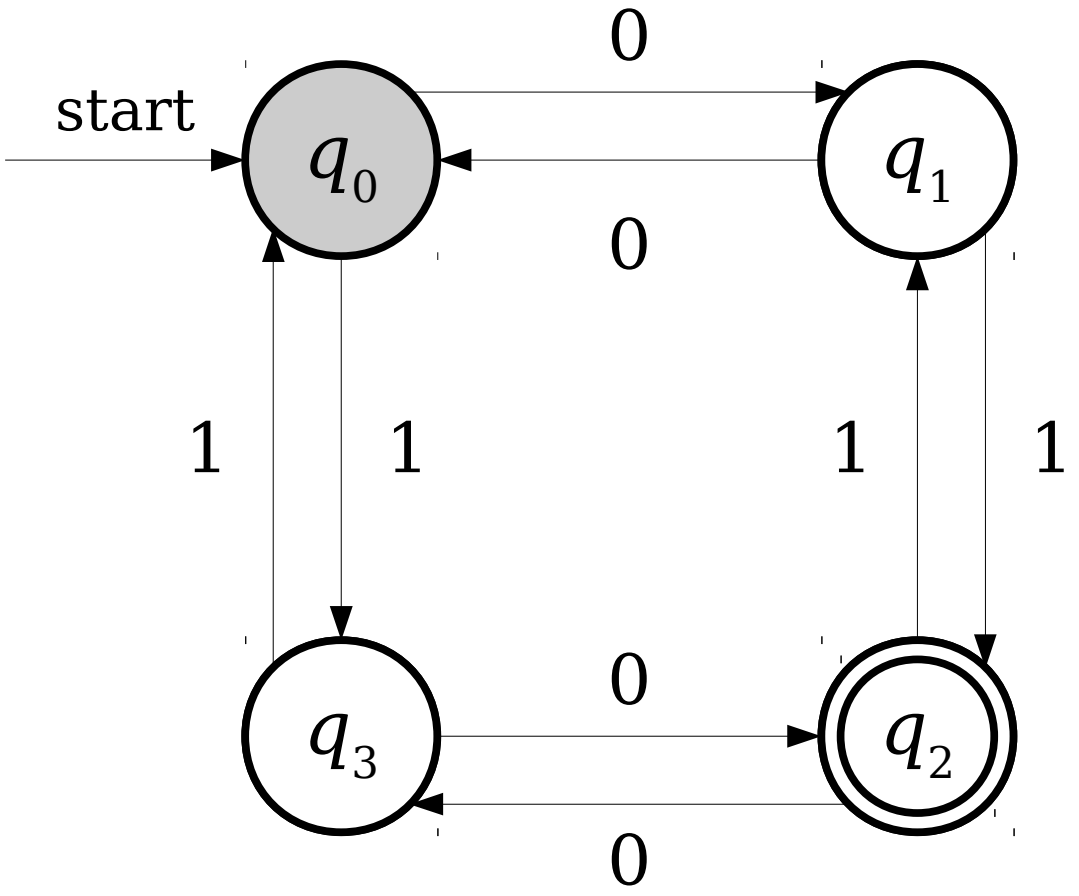
Each finite automaton consists of a set of ***states*** connected by ***transitions***.

A Simple Finite Automaton



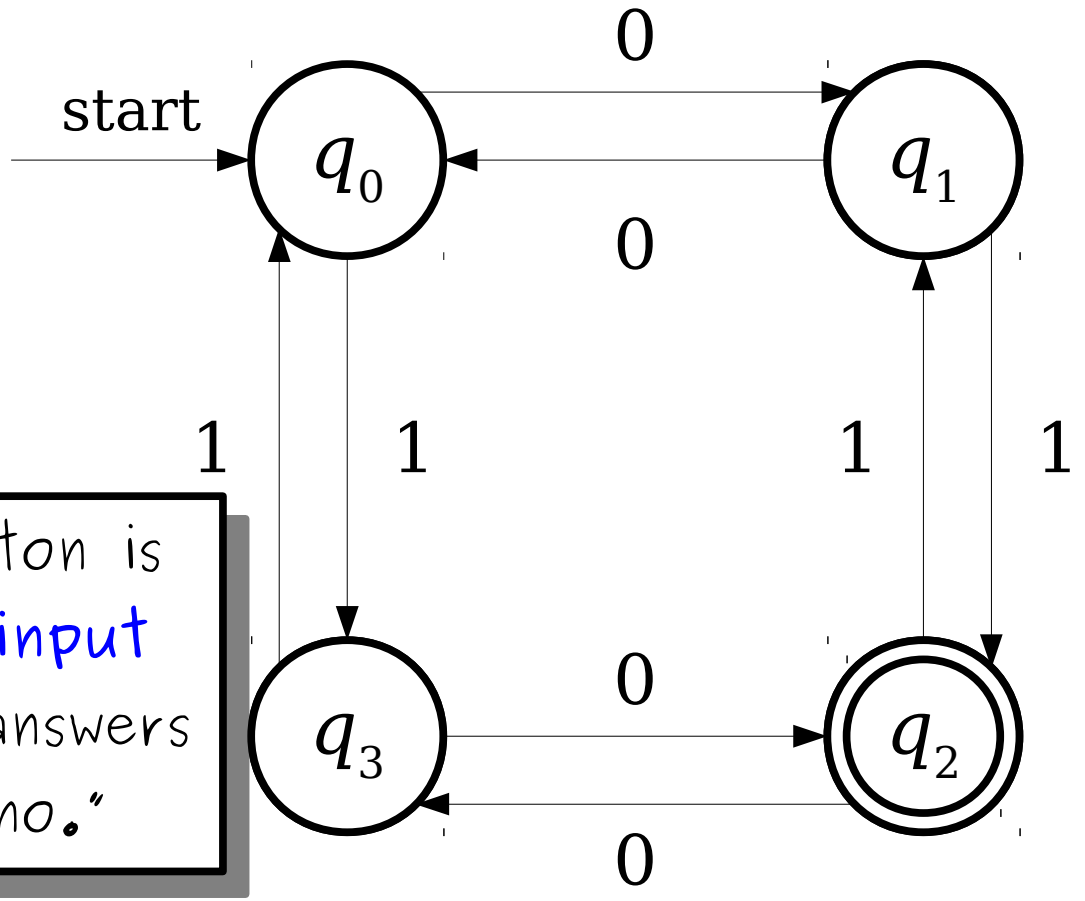
Each circle represents a state of the automaton.

A Simple Finite Automaton



One special state is designated as the start state.

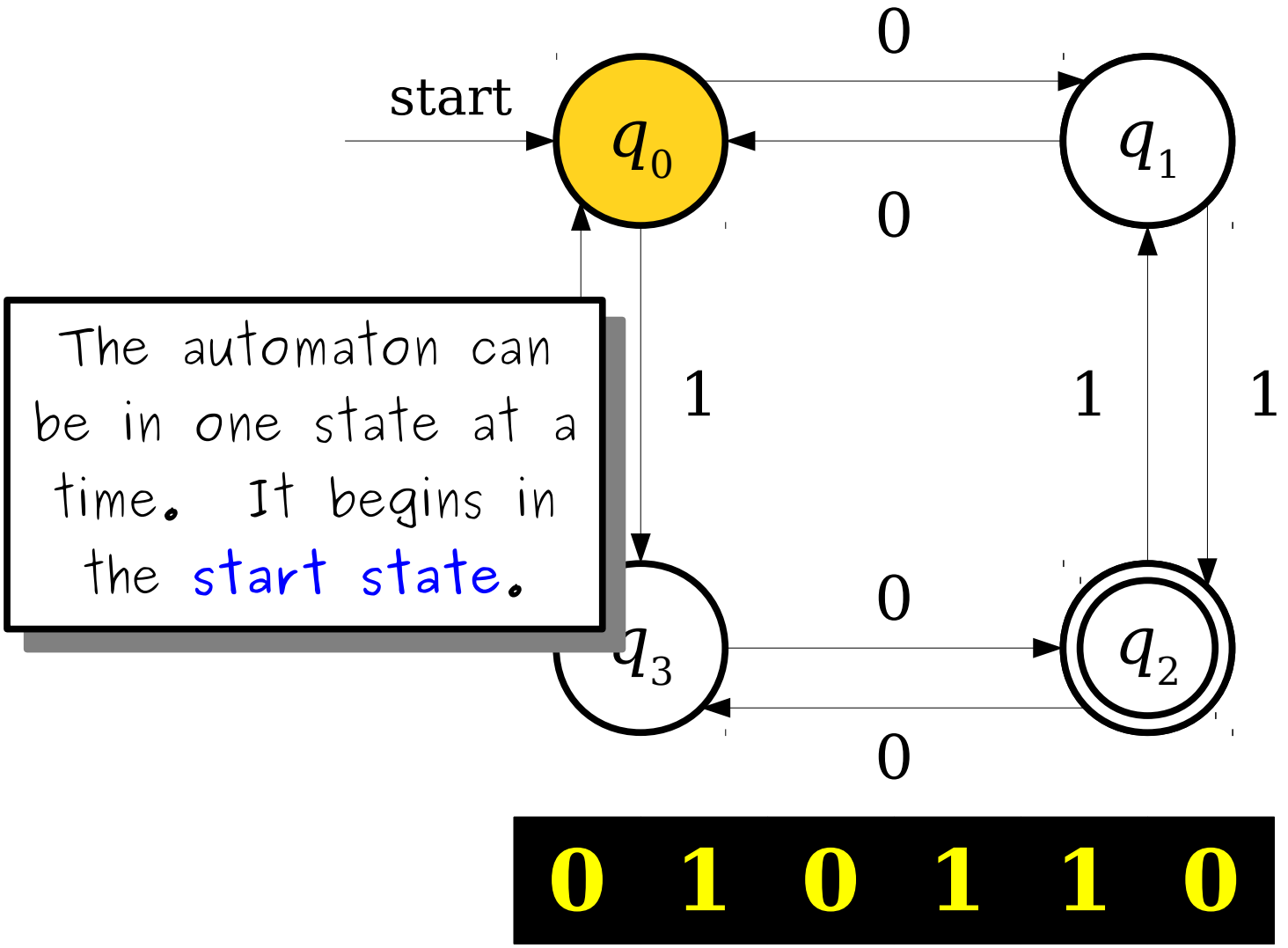
A Simple Finite Automaton



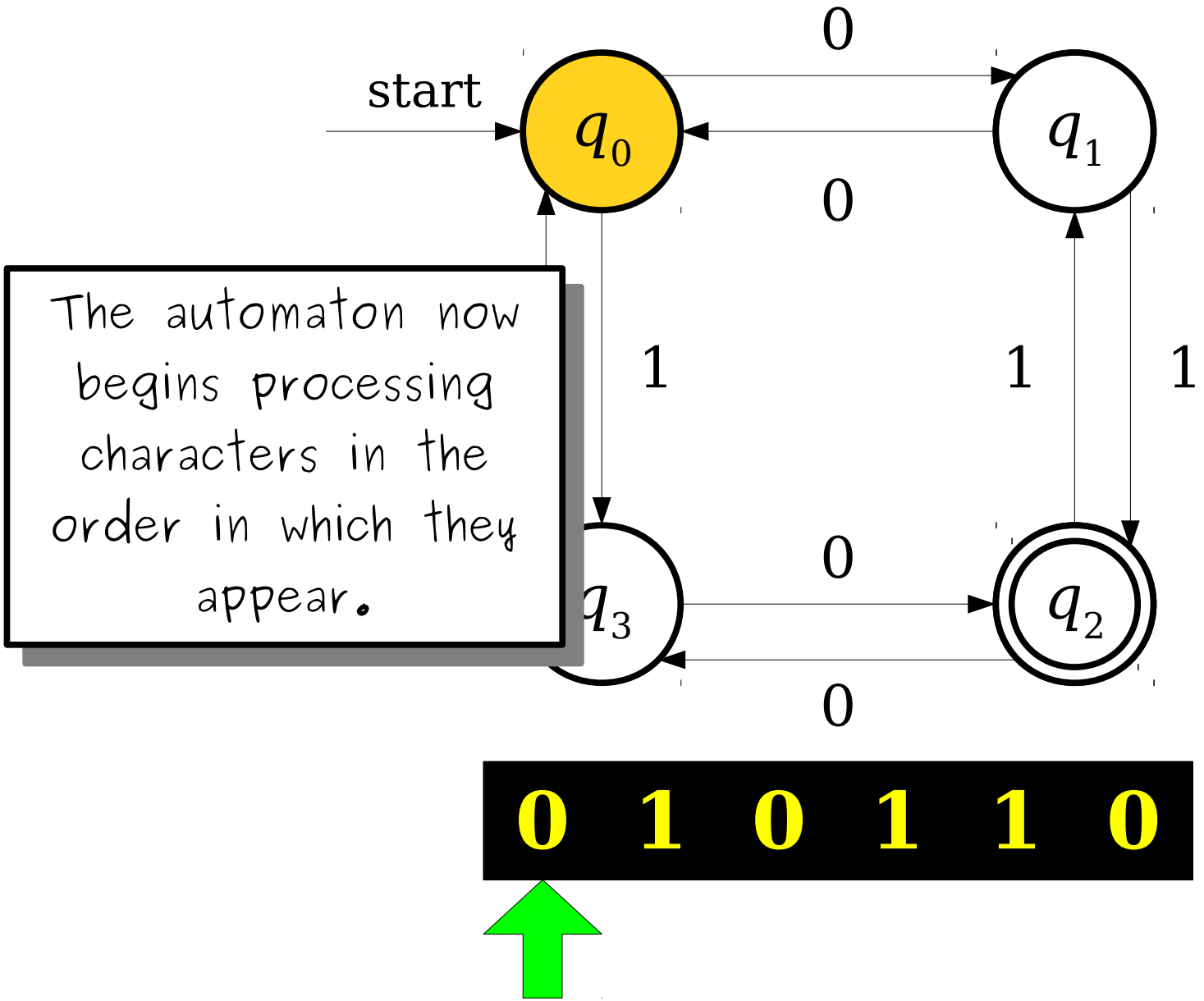
The automaton is run on an input string and answers "yes" or "no."

0 1 0 1 1 0

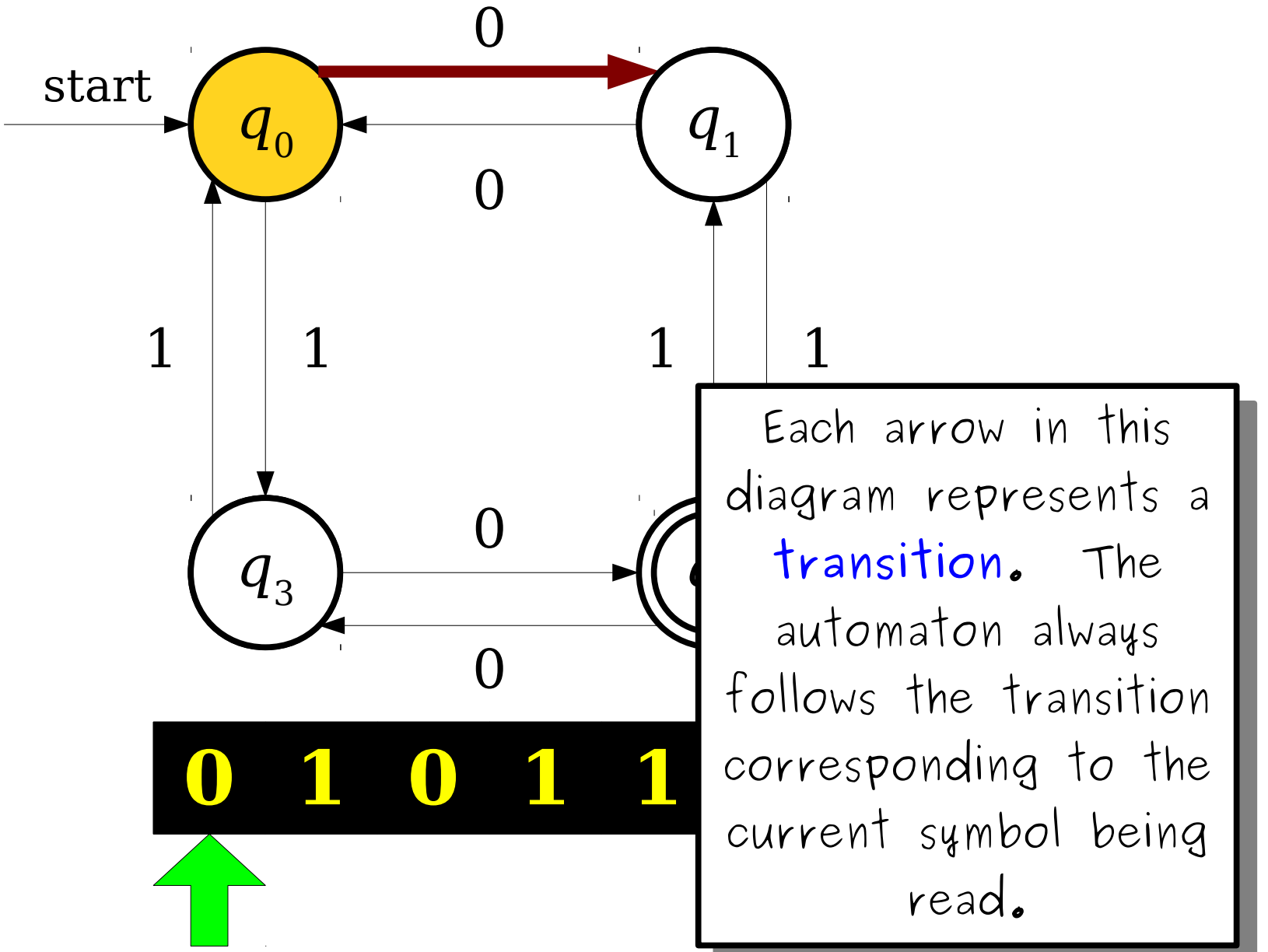
A Simple Finite Automaton



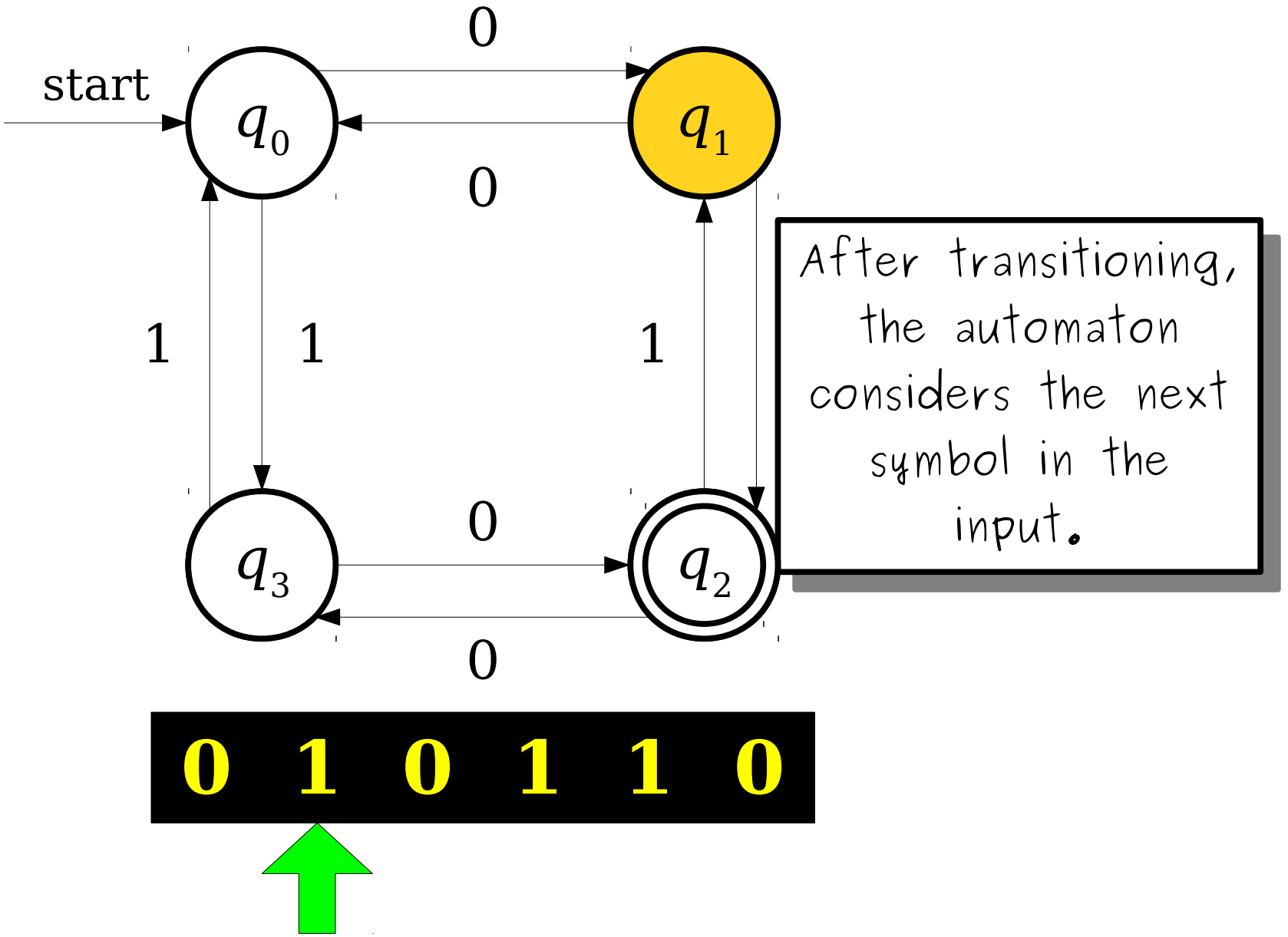
A Simple Finite Automaton



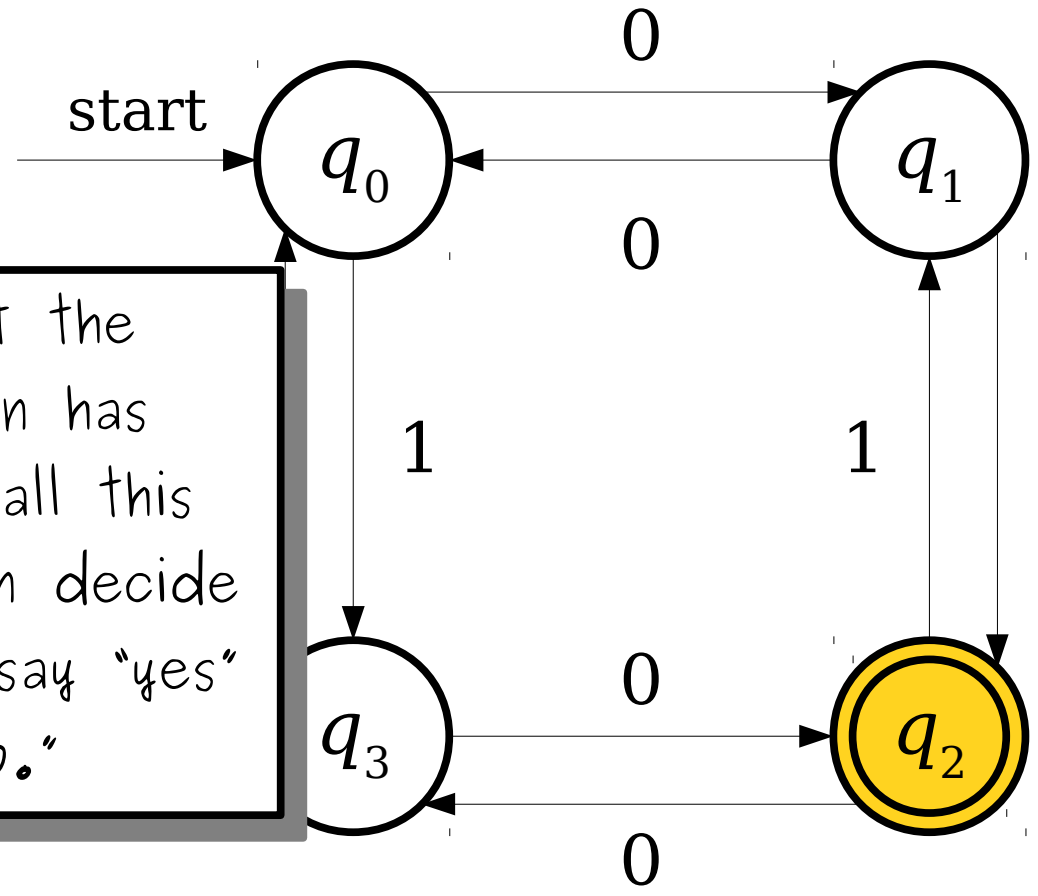
A Simple Finite Automaton



A Simple Finite Automaton



A Simple Finite Automaton

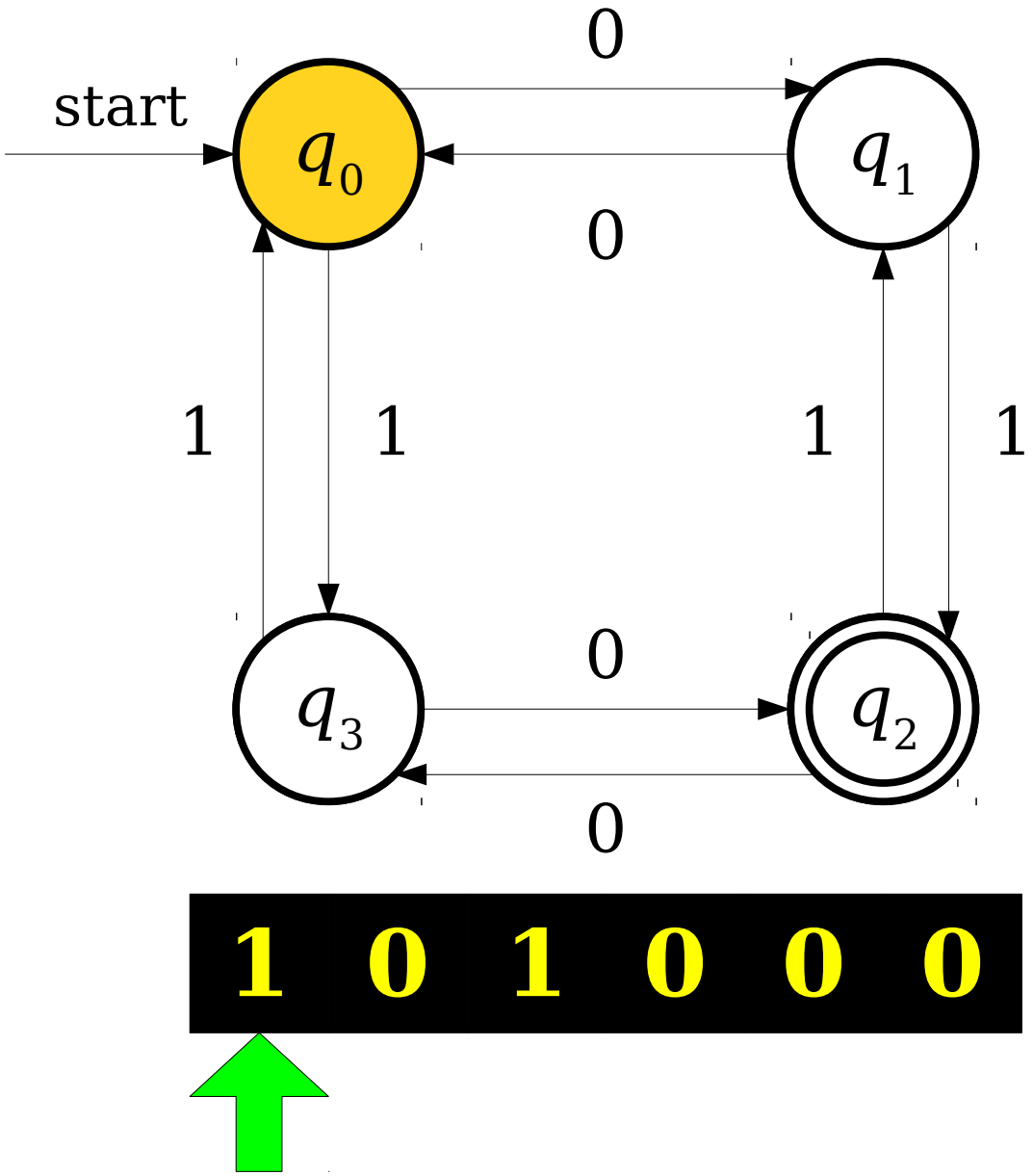


Now that the automaton has looked at all this input, it can decide whether to say "yes" or "no."

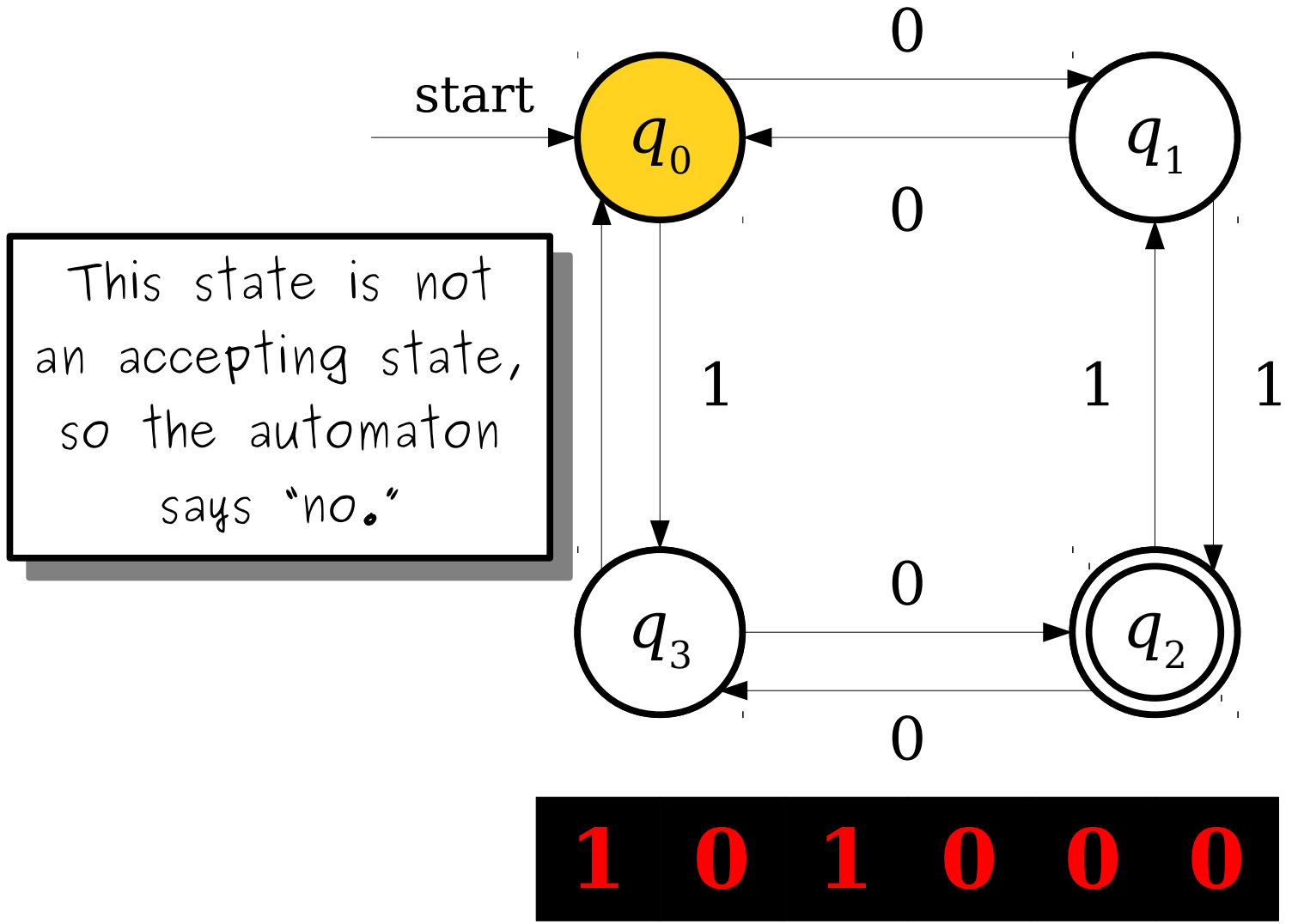
The double circle indicates that this state is an accepting state, so the automaton outputs "yes."

0 1 0 1 1 0

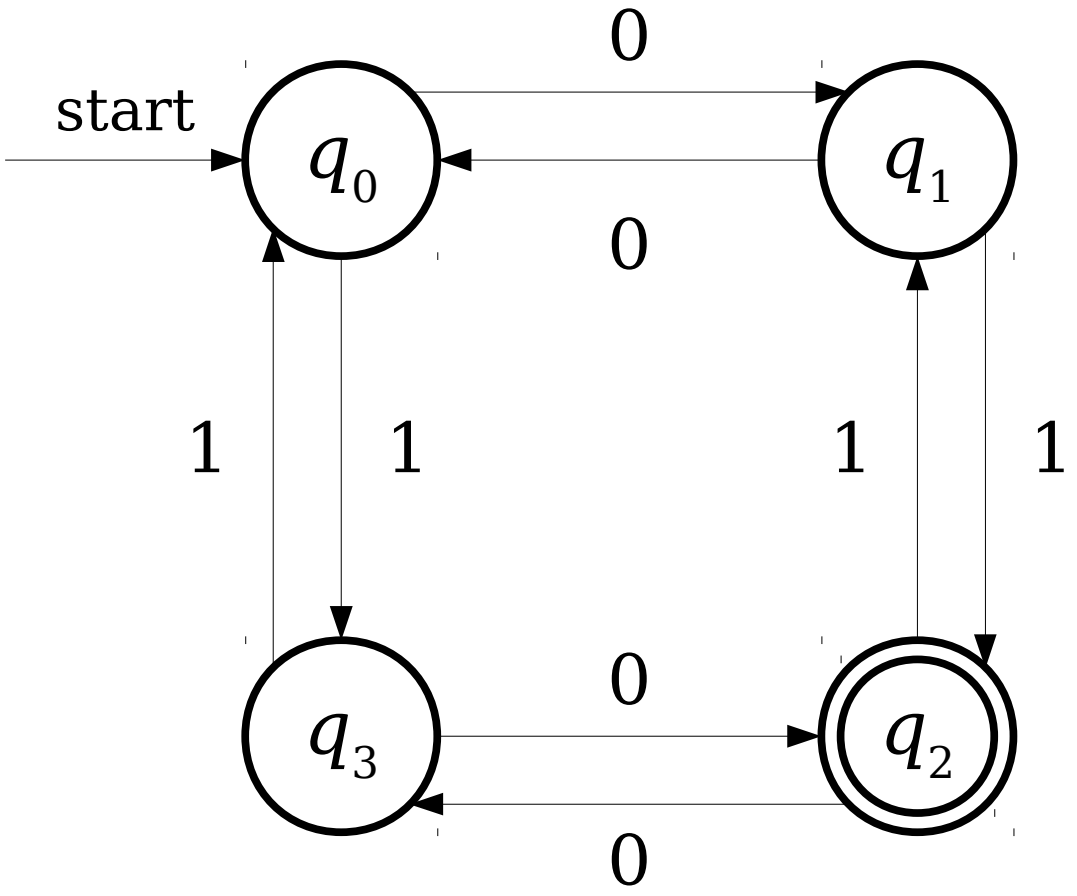
A Simple Finite Automaton



A Simple Finite Automaton



A Simple Finite Automaton



Try it yourself! Does the automaton **accept** (say yes) or **reject** (say no)?

1 1 0 1 1 1 0 0

The Story So Far

- A ***finite automaton*** is a collection of ***states*** joined by ***transitions***.
- Some state is designated as the ***start state***.
- Some states are designated as ***accepting states***.
- The automaton processes a string by beginning in the start state and following the indicated transitions.
- If the automaton ends in an accepting state, it ***accepts*** the input.
- Otherwise, the automaton ***rejects*** the input.

Time-Out For Announcements!

Problem Set Logistics

- Problem Set Four checkpoint was due at 12:50PM today.
- We'll return graded PS3's and graded PS4 checkpoints by Wednesday at 12:50PM.
- Remainder of PS4 is due a week from today at the start of class.

Midterm Logistics

- Midterm is this Thursday from 7PM – 10PM. Locations divvied up by last (family) name:
  - Aba – Mes: Go to **Annenberg Auditorium**.
  - Mex – Zoc: Go to **Cubberly Auditorium**.
- Closed-book, closed-computer, open one double-sided 8.5” × 11” sheet of notes.
- Covers material up through and including graphs (Lectures 00 – 08) and material from PS1 – PS3.

Practice Midterm Exam

- Practice midterm exam is ***tonight*** from 7PM – 10PM in Annenberg Auditorium.
- *Highly recommended!*
- Can't make it? No worries! We'll post the practice exam on the course website.
- Solutions will be available at the practice exam and in hardcopy in the solutions filing cabinet.

Even More Practice Problems

- We've released a new set of practice problems (based on the popular topics from Google Moderator) on the course website.
- Solutions will be released on Wednesday.
- Solutions to the practice problems from Friday are now available in hardcopy.

Your Questions

“What would be the most useful study strategy for our midterm? Crank through a ton of practice problems? Read the course reader? Review the lecture slides? I know "all of the above" is best, but if we had to prioritize?”

My recommendation is to quickly figure out what your strengths and weaknesses are and focus your studying efforts on the areas in which you need improvement. Take the practice exam to identify where you could use some extra practice. Review the slides to make sure you get the definitions, then do lots of practice problems. If you need more practice, read the course notes.

“What sorts of proof-formatting, formulas, definitions, etc. would you suggest we put on our cheat-sheet for the midterm Thursday?”

Everyone's different. Again, figure out what your own strengths and weaknesses are. Is there something you keep forgetting? Are there terms that you get confused? Are there particular proofs you like that you're worried you're going to forget? Put that kind of information on your notes sheet.

“Why are the answers to problem sets and checkpoint problems only available in hard copy? There have been many times when I know I and many others could have found the answers useful but didn't get a copy in lecture and couldn't make it to Gates.”

This one has a long answer.

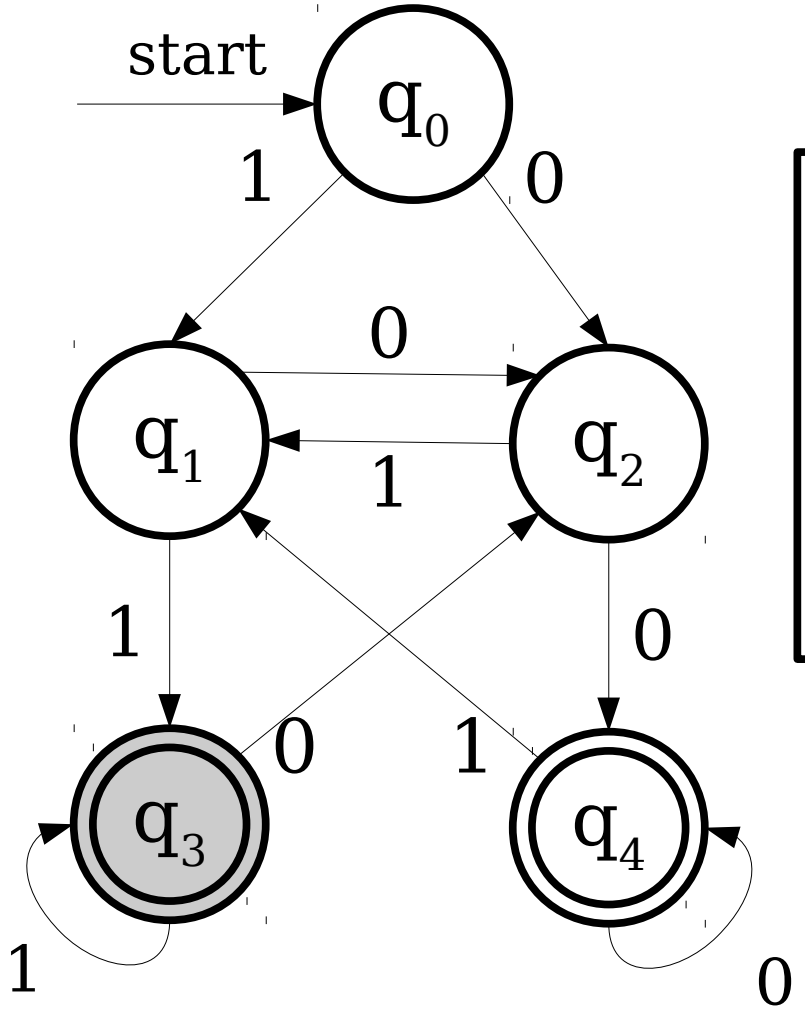
“why”

Back to CS103!

A finite automaton does ***not*** accept as soon as it enters an accepting state.

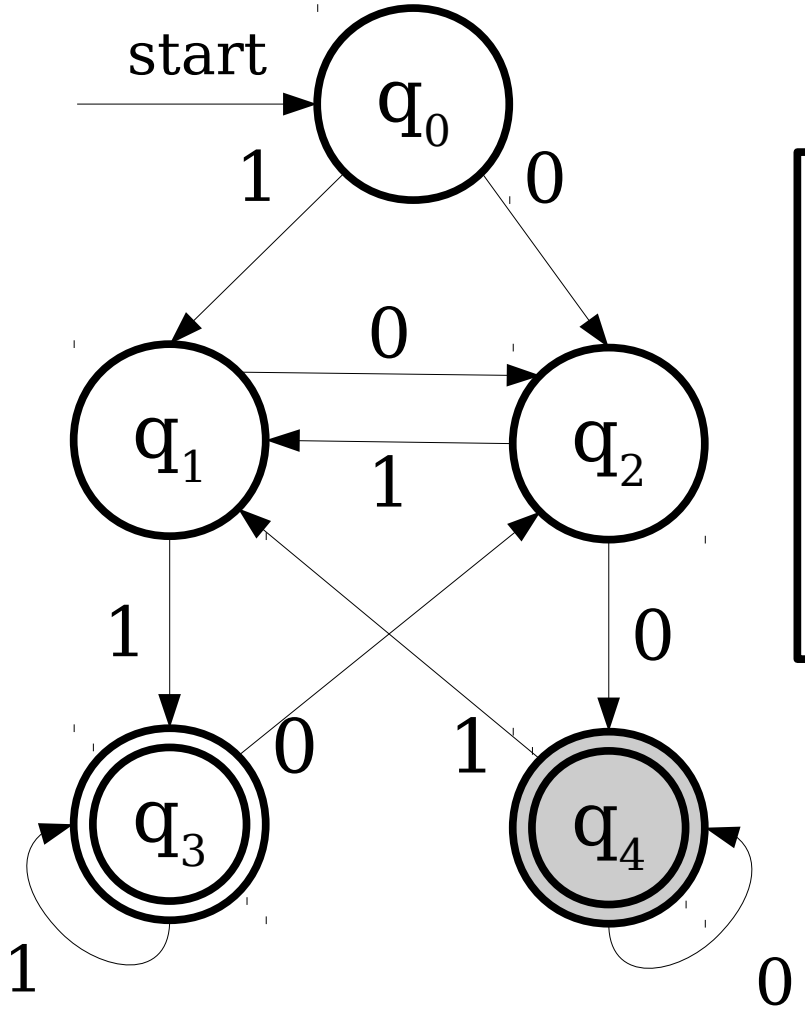
A finite automaton accepts if it ***ends*** in an accepting state.

What Does This Accept?



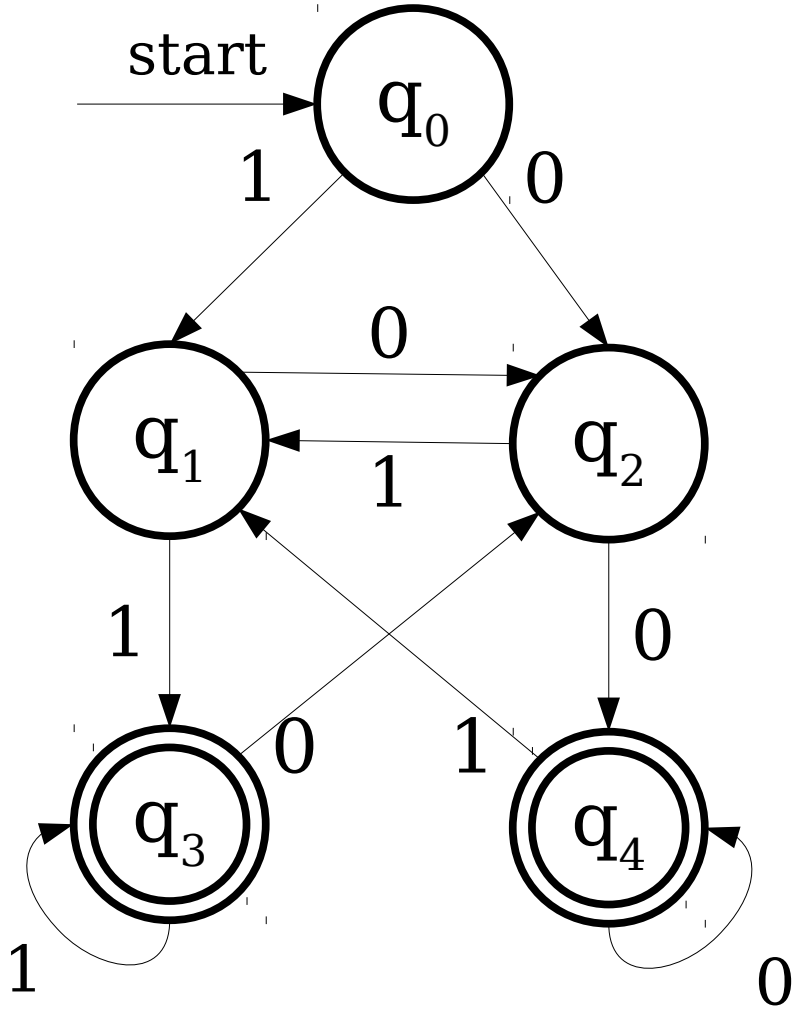
No matter where we start in the automaton, after seeing two 1's, we end up in accepting state q_3.

What Does This Accept?



No matter where we start in the automaton, after seeing two 0's, we end up in accepting state q_5.

What Does This Accept?



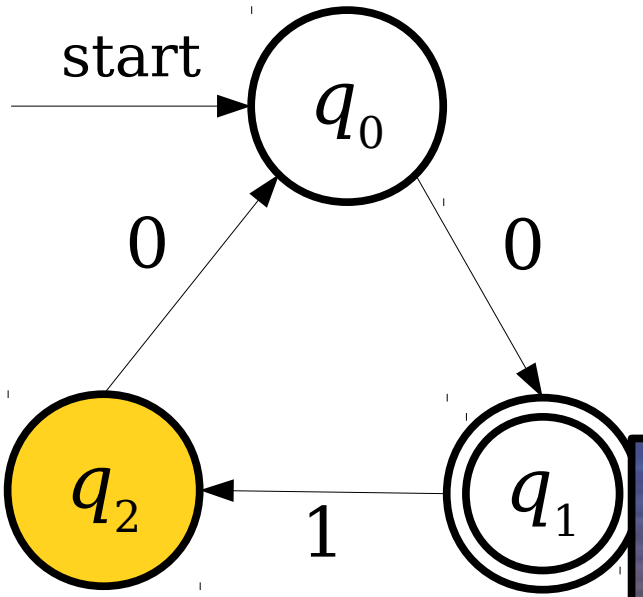
This automaton accepts a string iff the string ends in *00* or *11*.

The ***language of an automaton*** is the set of strings that it accepts.

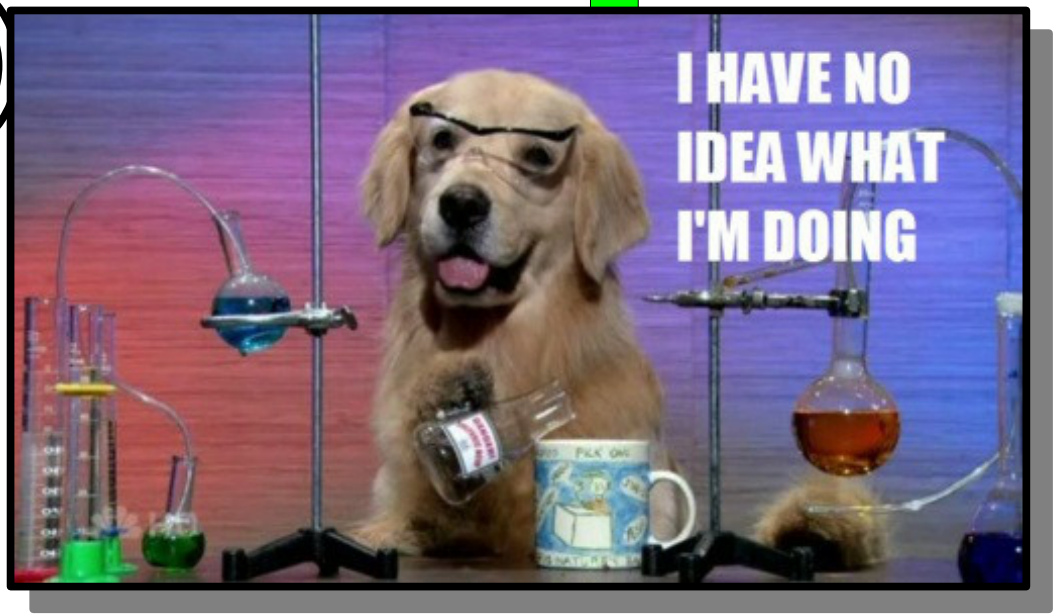
If D is an automaton, we denote the language of D as $\mathscr{L}(D)$.

$$\mathscr{L}(D) = \{\ w \in \Sigma^* \mid D \text{ accepts } w\ \}$$

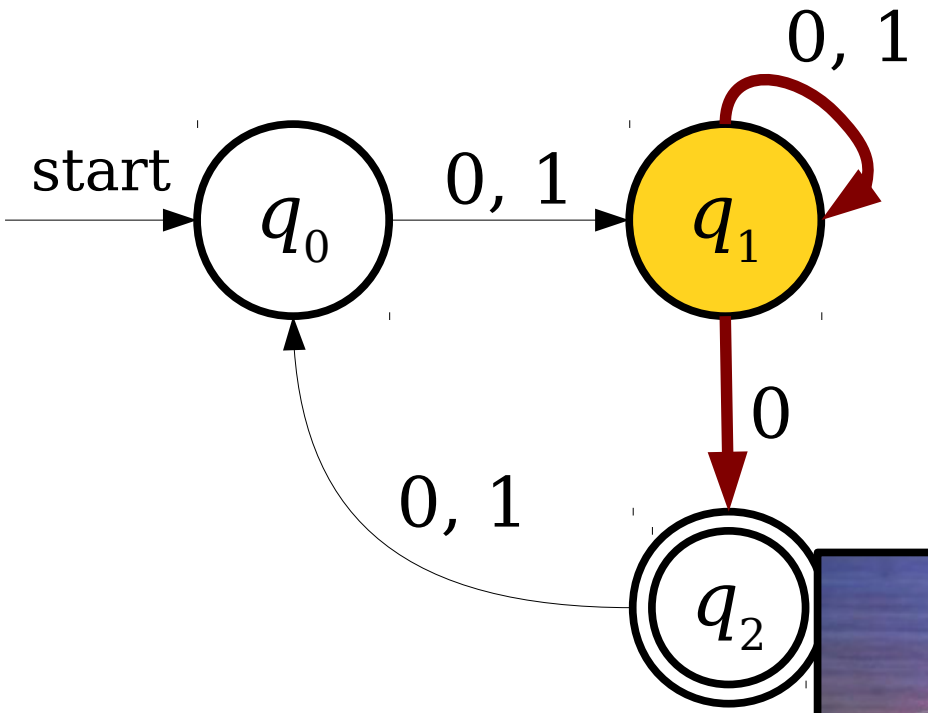
A Small Problem



0 1 1 0



Another Small Problem



The Need for Formalism

- In order to reason about the limits of what finite automata can and cannot do, we need to formally specify their behavior in *all* cases.
- All of the following need to be defined or disallowed:
  - What happens if there is no transition out of a state on some input?
  - What happens if there are *multiple* transitions out of a state on some input?

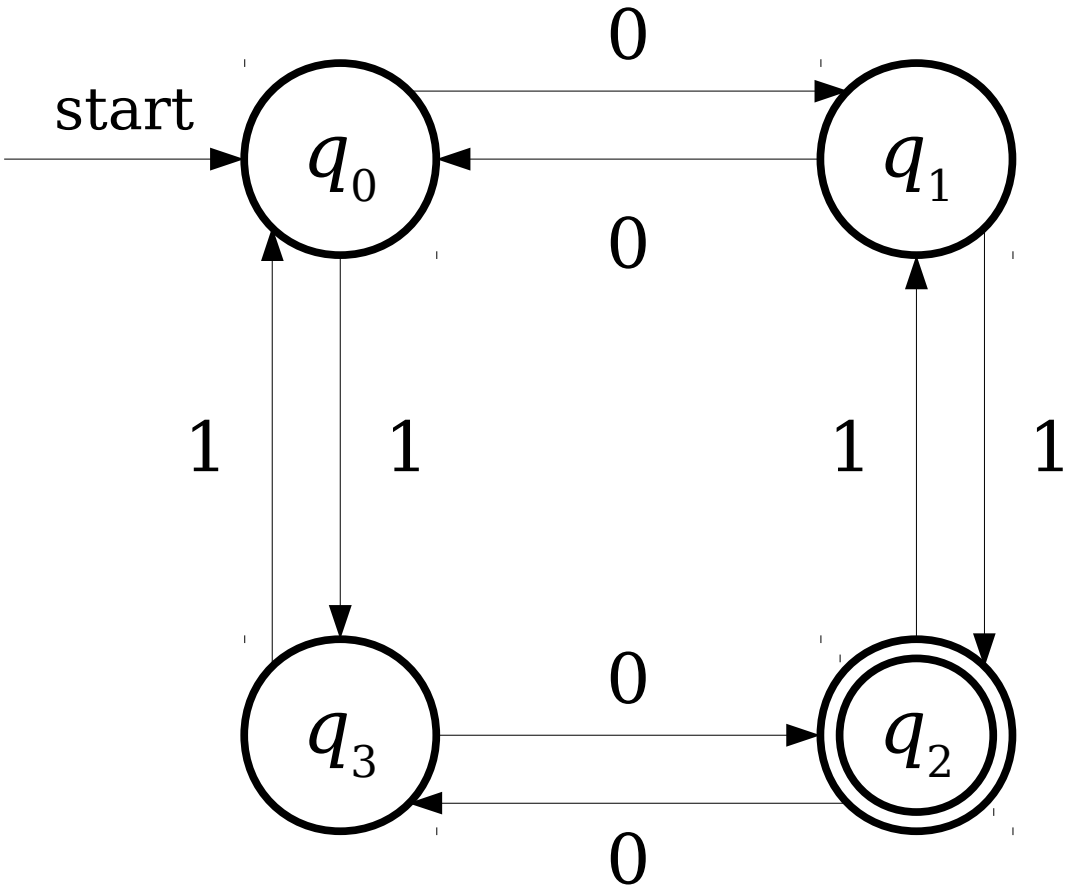
DFAs

- A **DFA** is a
  - **D**eterministic
  - **F**inite
  - **A**utomaton
- DFAs are the simplest type of automaton that we will see in this course.

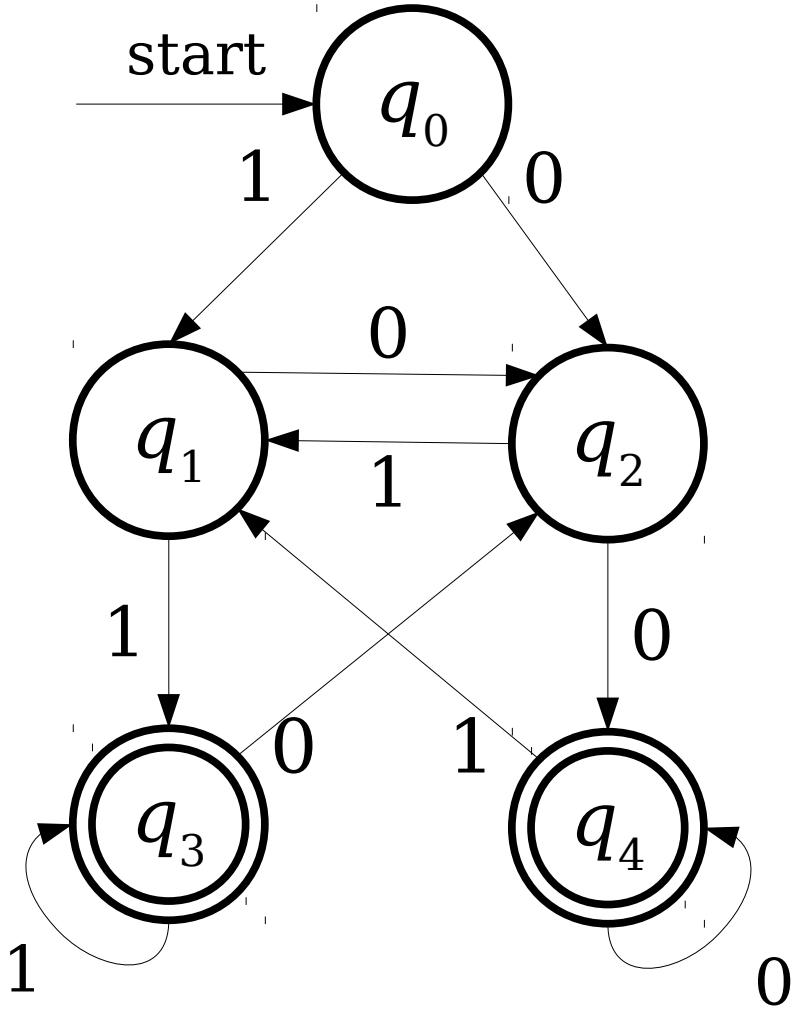
DFAs, Informally

- A DFA is defined relative to some alphabet Σ.
- For each state in the DFA, there must be ***exactly one*** transition defined for each symbol in Σ.
  - This is the “deterministic” part of DFA.
- There is a unique start state.
- There are zero or more accepting states.

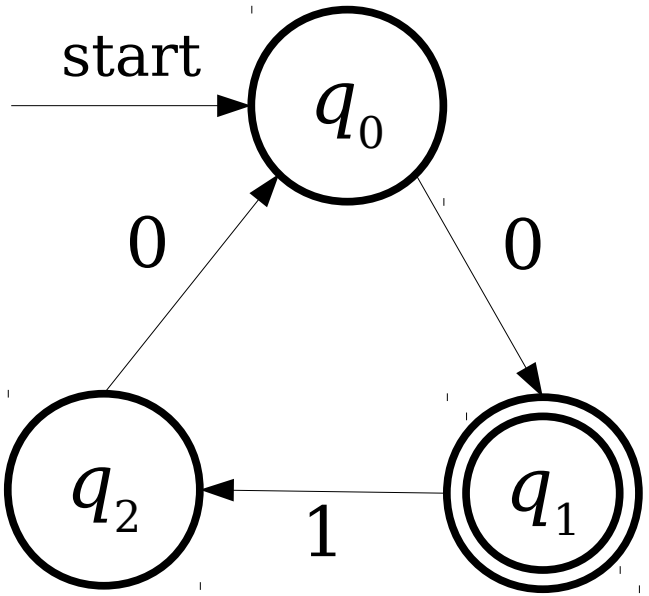
Is this a DFA over {0, 1}?



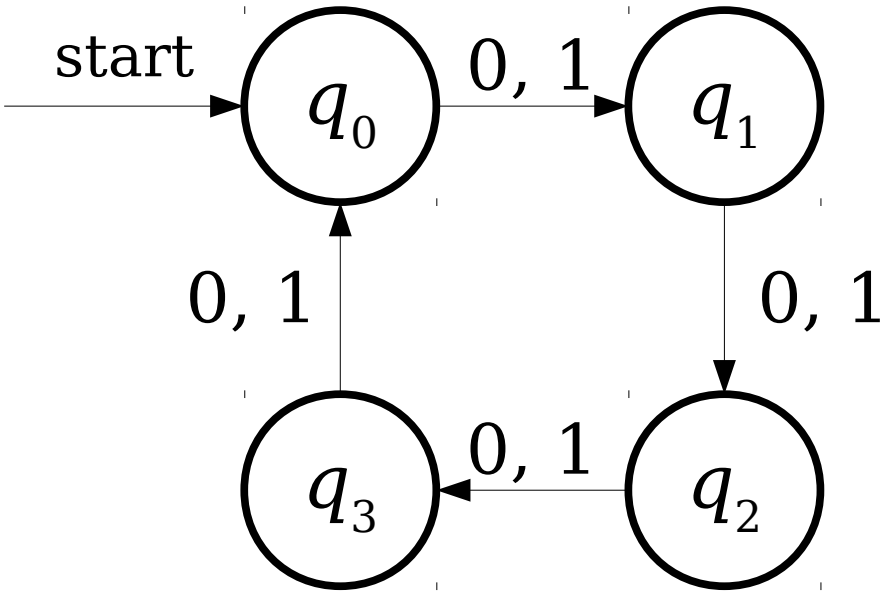
Is this a DFA over {0, 1}?



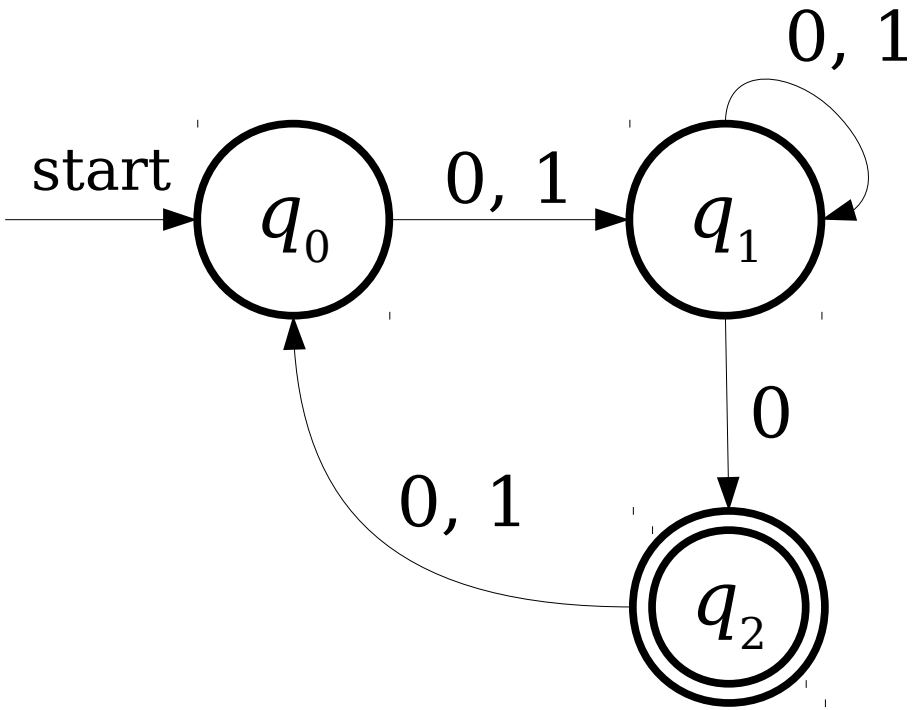
Is this a DFA over {0, 1}?



Is this a DFA over {0, 1}?



Is this a DFA over {0, 1}?

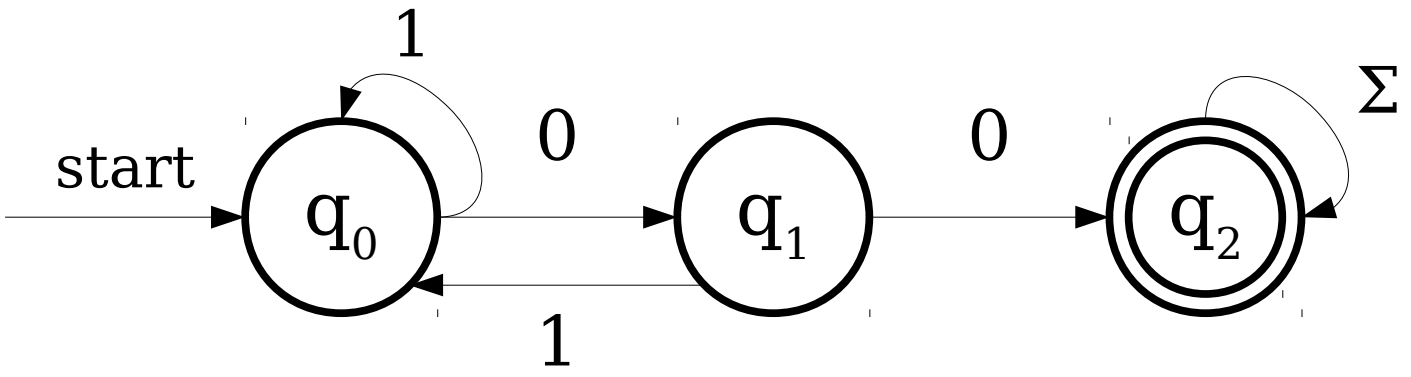


Designing DFAs

- At each point in its execution, the DFA can only remember what state it is in.
- **DFA Design Tip:** Build each state to correspond to some piece of information you need to remember.
  - Each state acts as a “memento” of what you're supposed to do next.
  - Only finitely many different states ≈ only finitely many different things the machine can remember.

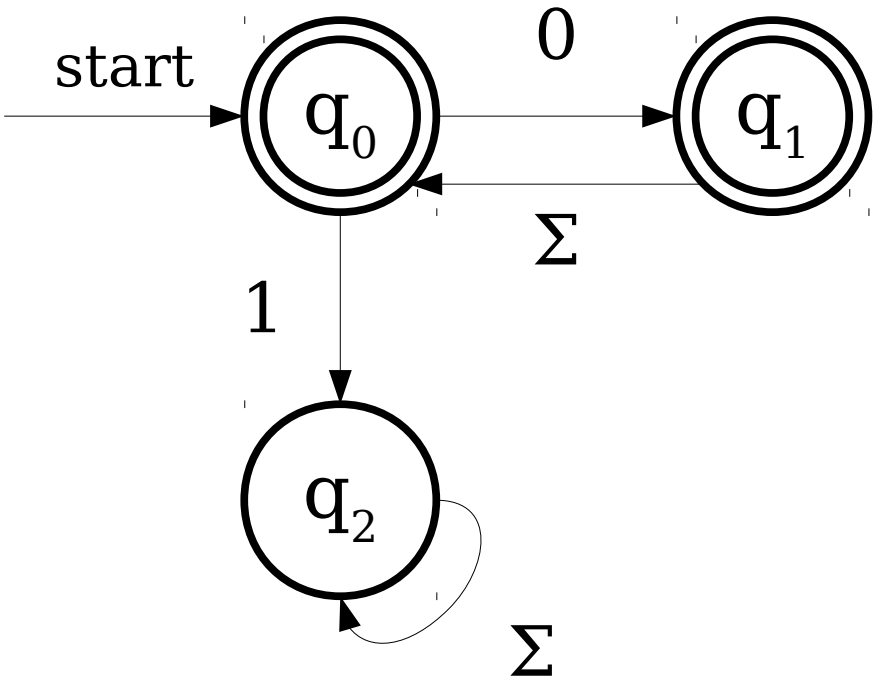
Recognizing Languages with DFAs

$L = \{ w \in \{\texttt{0}, \texttt{1}\}^* \mid w \text{ contains } \texttt{00} \text{ as a substring} \}$



Recognizing Languages with DFAs

$L = \{ w \in \{0, 1\}^* |$ every other character of w, starting with the first character, is 0 $\}$



More Elaborate DFAs

$L = \{ w \mid w \text{ is a C-style comment } \}$

Suppose the alphabet is

$\Sigma = \{$ a, *, / $\}$

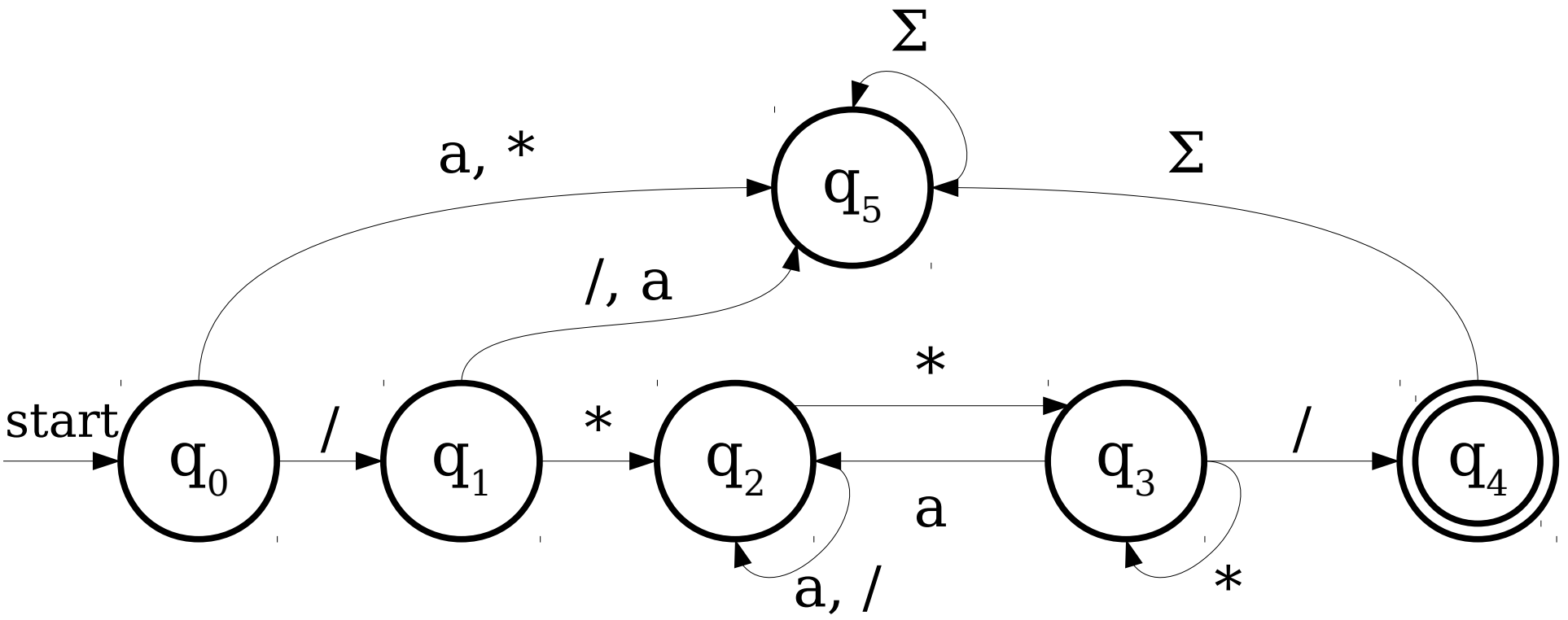
Try designing a DFA for comments!

Some test cases:

ACCEPTED	REJECTED
/*a*/	/**
/**/	/**/a/*aa*/
/***/	aaa/**/
/*aaa*aaa*/	/*/

More Elaborate DFAs

$L = \{ w \mid w \text{ is a C-style comment } \}$



Next Time

- **Regular Languages**
  - What is the expressive power of DFAs?
- **NFAs**
  - Automata with Magic Superpowers!
- **Nondeterminism**
  - Nondeterminisic computation.
  - Intuitions for nondeterminism.
  - Programming with nondeterminism.