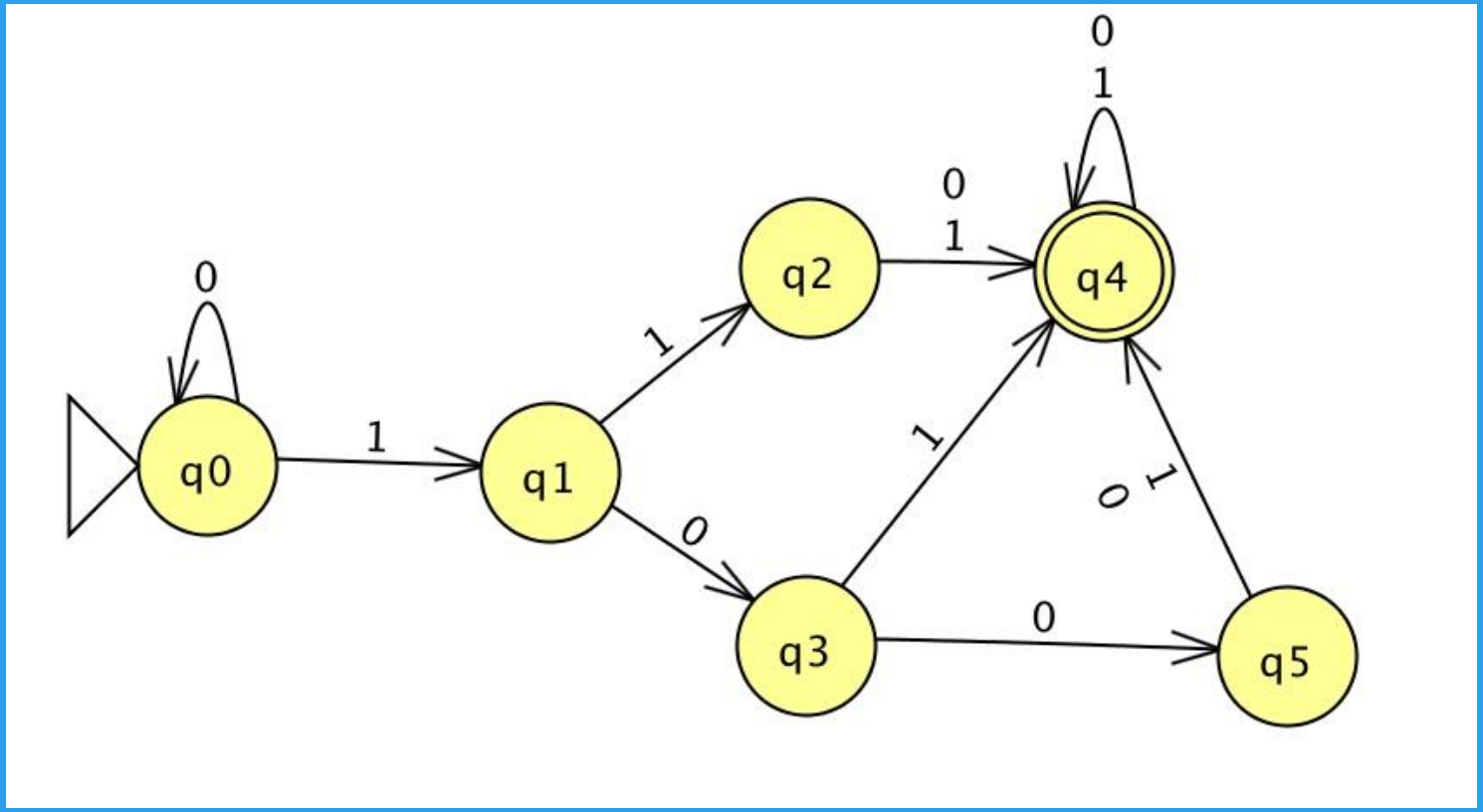
Theory of Computation and Programming Languages



DFAs

CS51 – Spring 2026

Models of computation

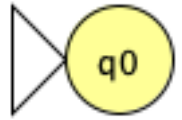
- We will see some *very* simple models of computation.
  - Much simpler than modern computers.
- Why study simple models?
  - They are much easier to understand.
  - They allow us to reason and think about them.
  - Even if simple, many have similar capabilities as modern computers.
    - Just not as fast.
- Reasoning about simple models can allow us to reason about computational power of computers.
- Answer questions like, what are the capabilities and limitations of computers?
  - Are there problems that we could NEVER (even given an infinite amount of time, memory, etc.) answer with a computer?

Languages

- A language L is a set of strings.
- Strings in that language are built up from an alphabet of characters, denoted as Σ.
- We want to ask a simple question, like:
  - Given a string of characters in Σ does it belong to the language L?

Deterministic Finite Automata (DFAs)

- We have a set of **states**, indicated by circles. 

- We have a **start state** where computation begins (indicated by a triangle) 

- We have a collection of **final states** (indicated by a concentric circle) 

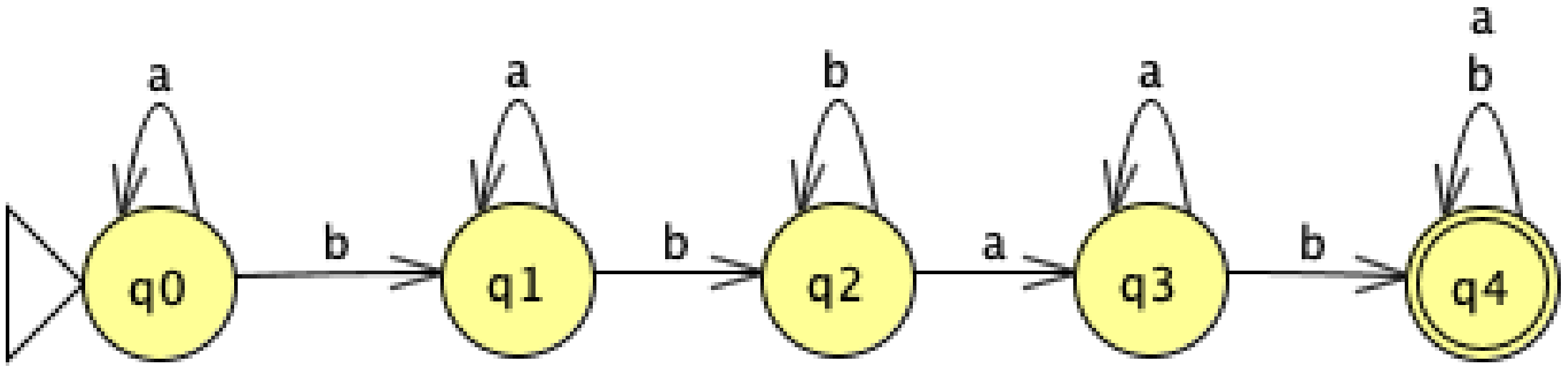
- For **each** state and **each** letter in our alphabet, we have a transition to another state
- Note it's critical that for each state there is **exactly one** transition for each letter in the alphabet.
- If not, then if we encountered that letter we wouldn't know what to do!

Computing on a DFA

- We have a string as input on a tape.
- We start at the beginning of the string.
- We read a symbol from the tape and transition to the state indicated by the model.
- If we end in a final state (i.e. when we get to the end of the string we are in a final state), we accept the string.
- Otherwise, if when we get to the end of the string on the tape and we're in a non-final state we reject.
- All strings accepted by a DFA define a language.
- We will use JFLAP https://www.jflap.org/ to create DFAs and evaluate whether a DFA recognizes a specific string.

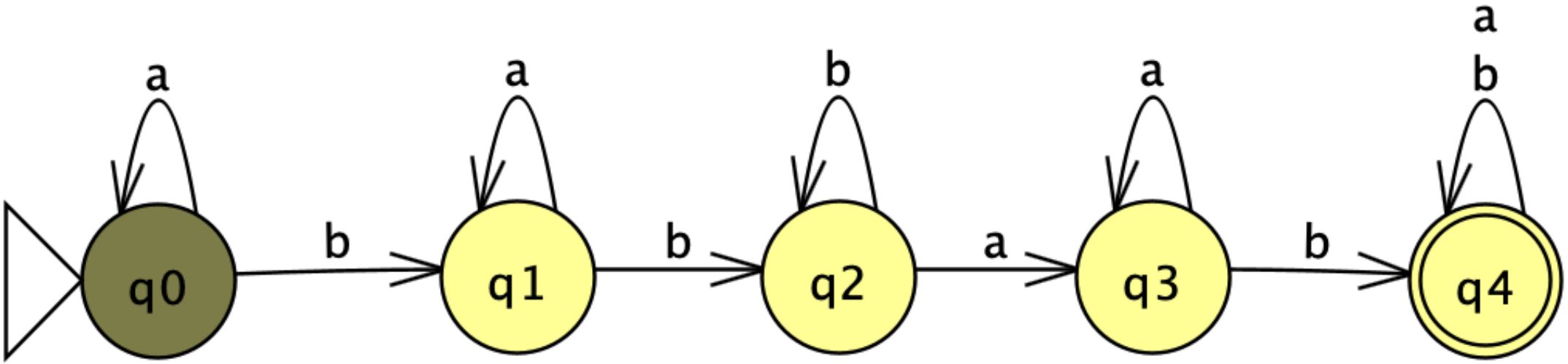
Example

- You are given the following DFA over the alphabet {a, b} and the string abbbaabab.
- Start in q0.



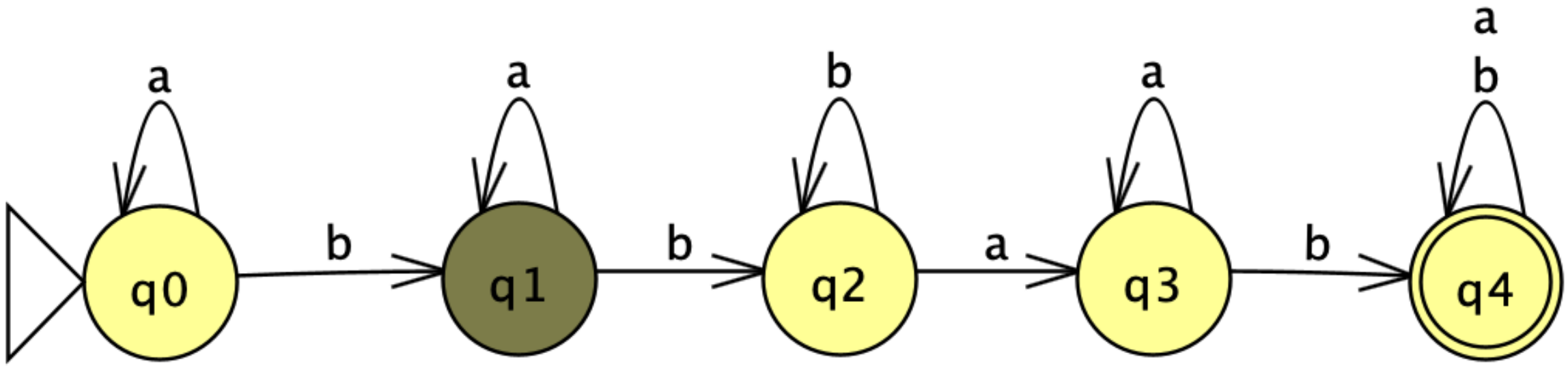
Example

- **a**bbbaabab
- a -> q0



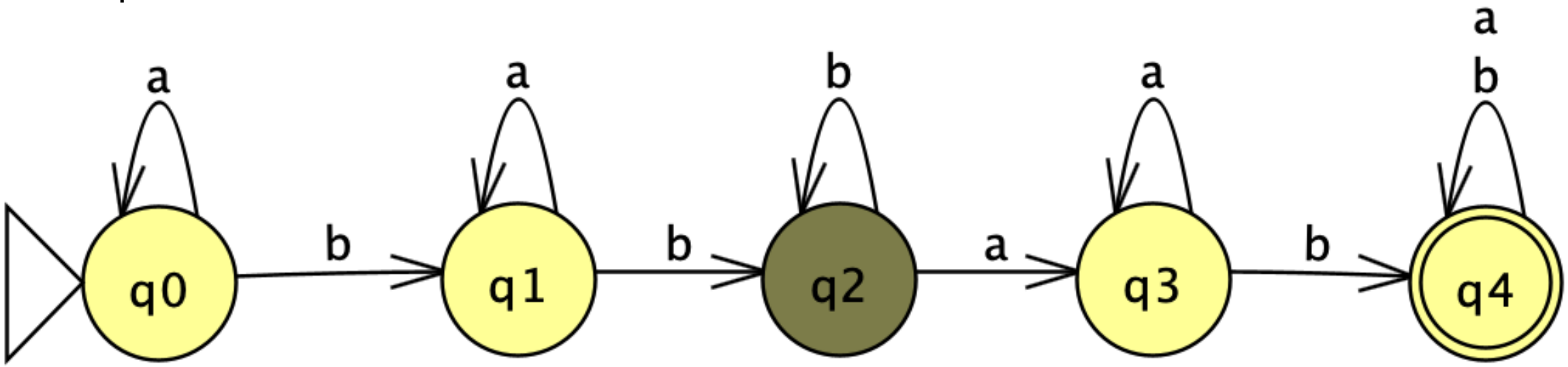
Example

- a**b**bbaabab
- b -> q1



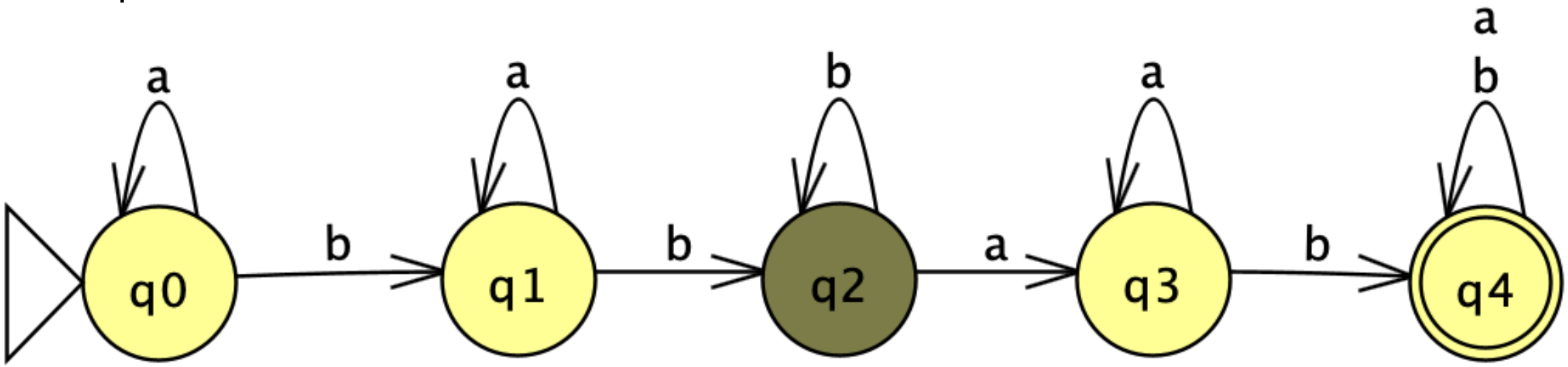
Example

- ab**b**baabab
- b -> q2



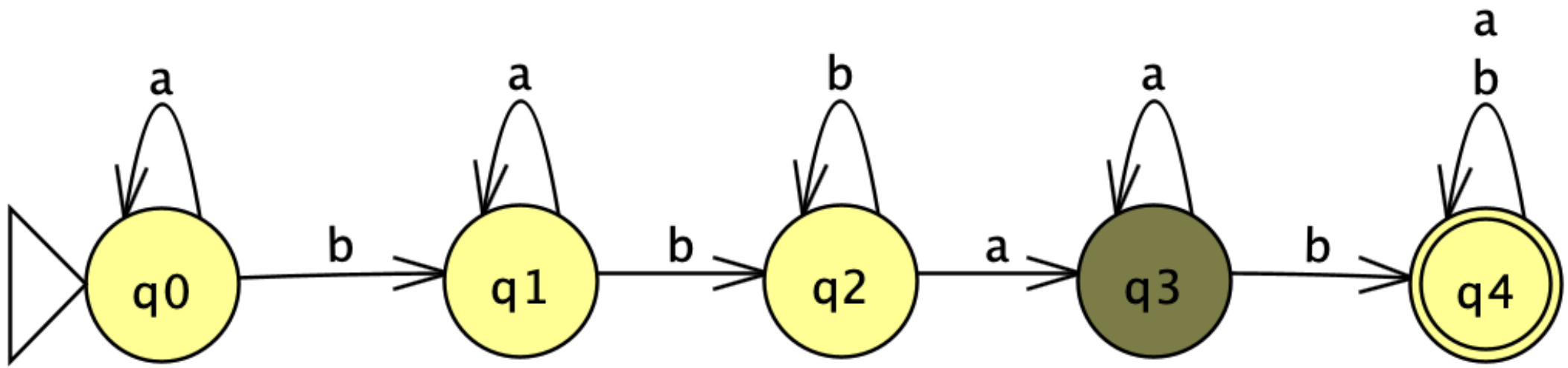
Example

- abb**b**aabab
- b -> q2



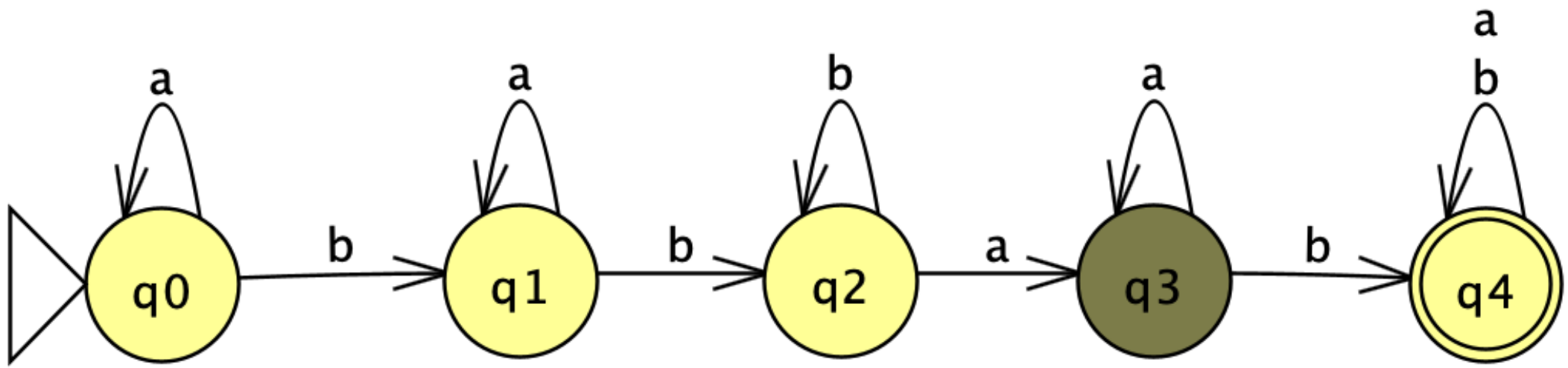
Example

- abbb**a**abab
- a -> q3



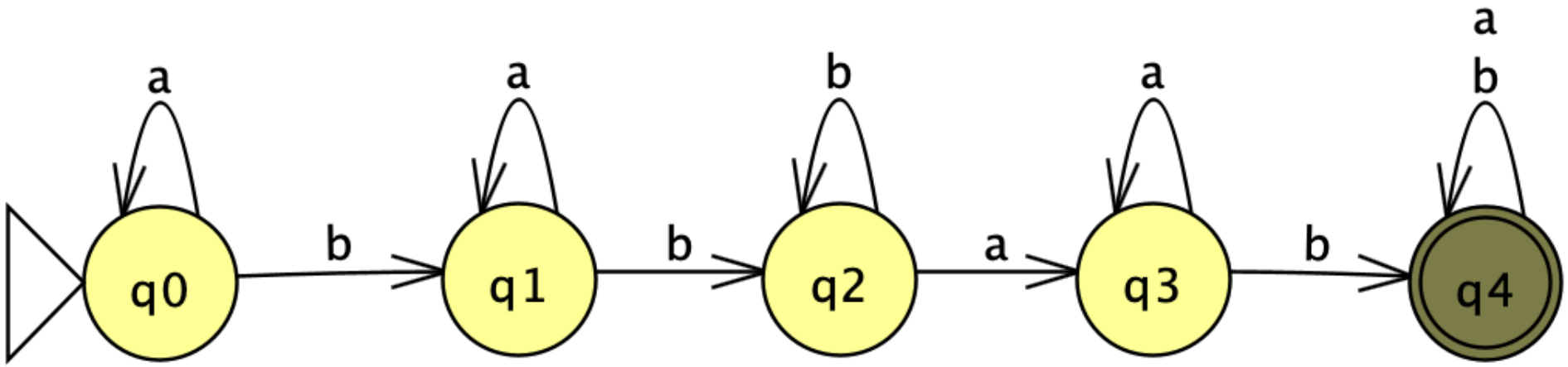
Example

- abbba**a**bab
- a -> q3



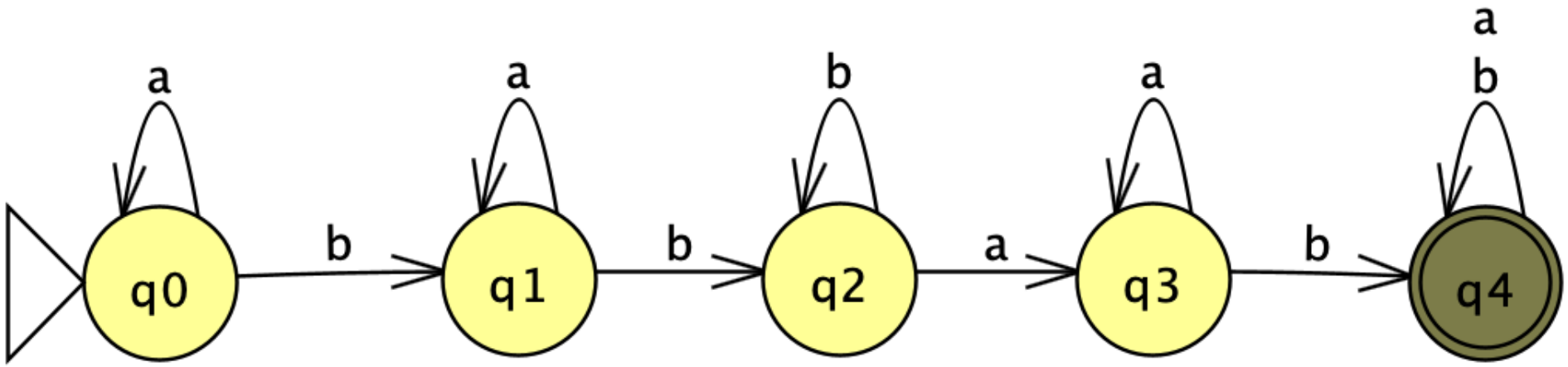
Example

- abbbaa**b**ab
- b -> q4



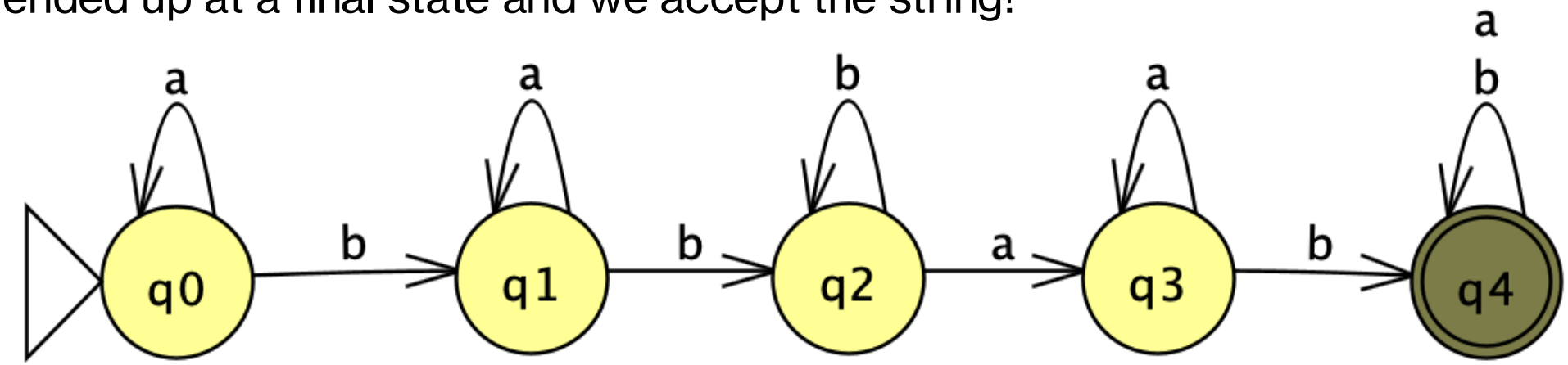
Example

- abbbaab**a**b
- a -> q4



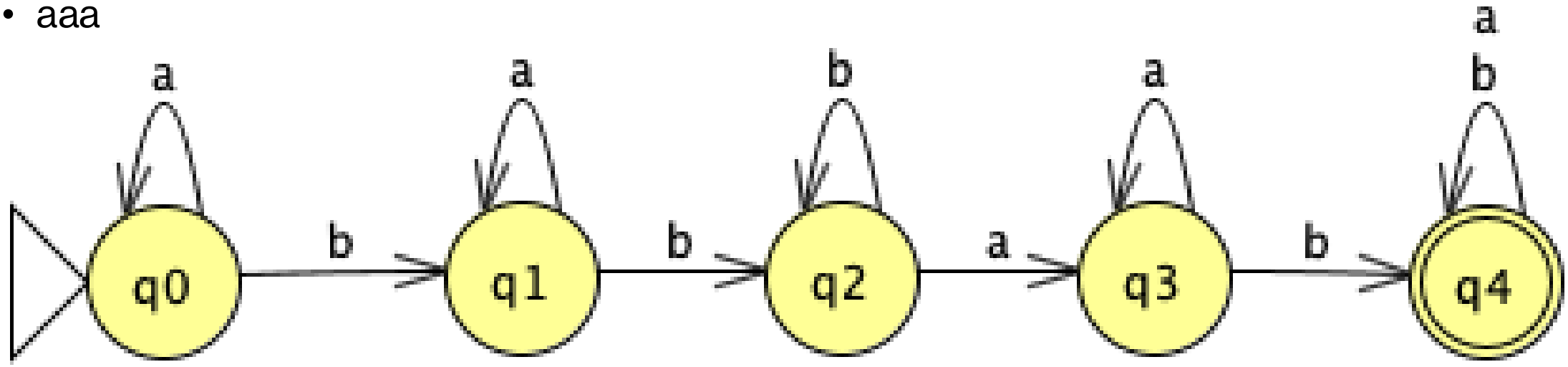
Example

- abbbaaba**b**
- b -> q4
- We ended up at a final state and we accept the string!



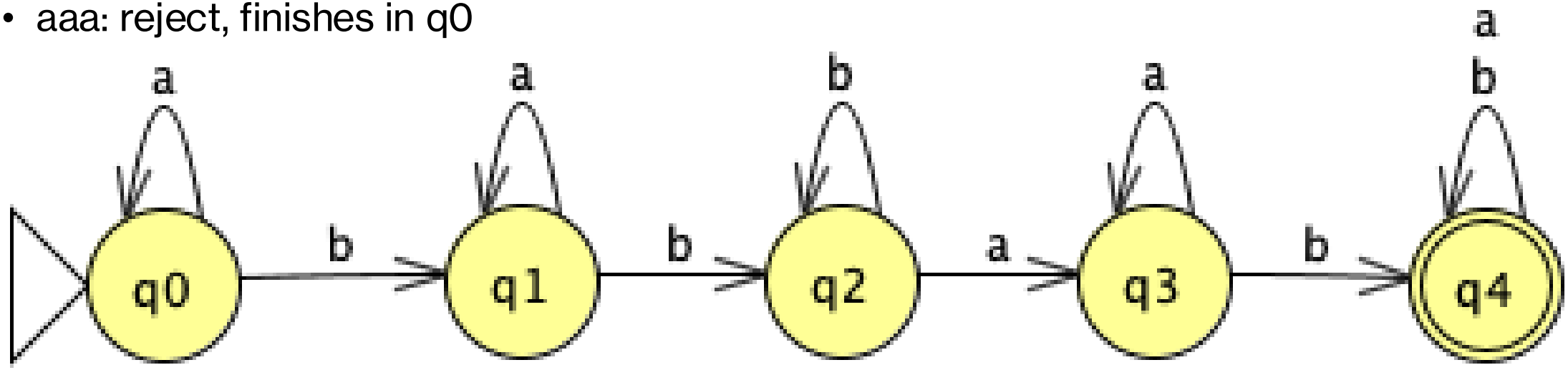
Practice time

- What about the following strings?
- aabab
- bbb
- babbab
- aabbaab
- aaa



Answer

- What about the following strings?
- aabab: reject, finishes in q2
- bbb: reject, finishes in q2
- babbab: accept
- aabbaab: accept
- aaa: reject, finishes in q0

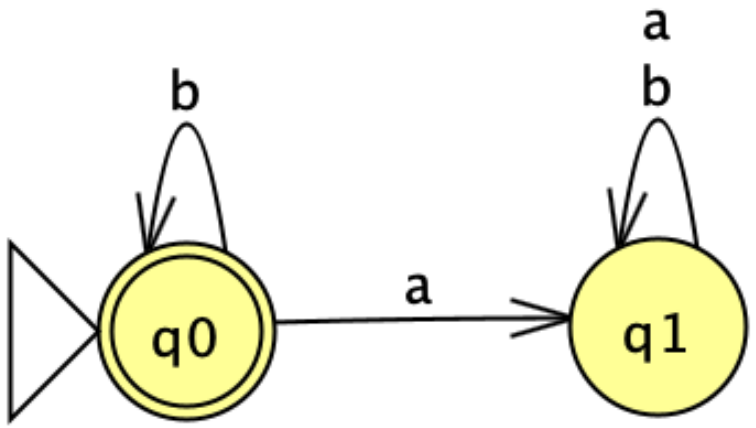


Formal definition of DFAs

- A Deterministic Finite Automaton (DFA) is a finite state machine that accepts or rejects finite strings of symbols and produces the same unique computation for each unique input string. For any given finite input string, the DFA will halt and either accept or reject the string. A DFA, M, is said to recognize a language, $L(M)$, which is the set of all strings that M accepts.
- Formally, a DFA is described by a 5-tuple $(Q, \Sigma, \delta, q0, F)$
  - Q is a finite set of states
  - Σ is a finite set of input symbols also known as the alphabet
  - δ is a state transition function ($\delta : Q \times \Sigma \rightarrow Q$)
  - $q0$ is the start state ($q0 \in Q$)
  - F is a set of accept states ($F \subseteq Q$).

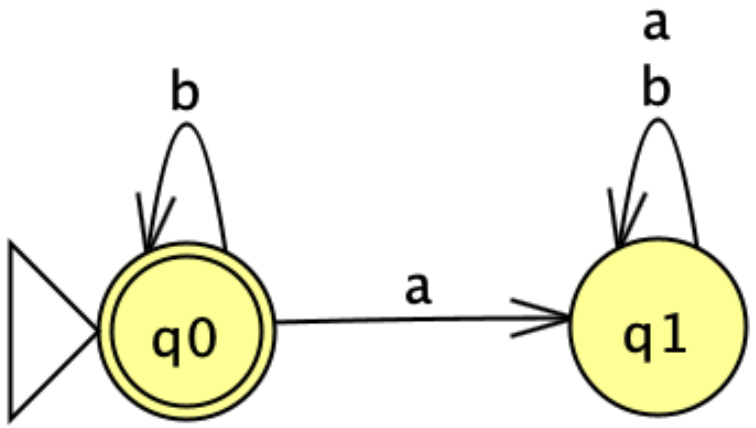
Practice Time

- What language is described by this DFA?



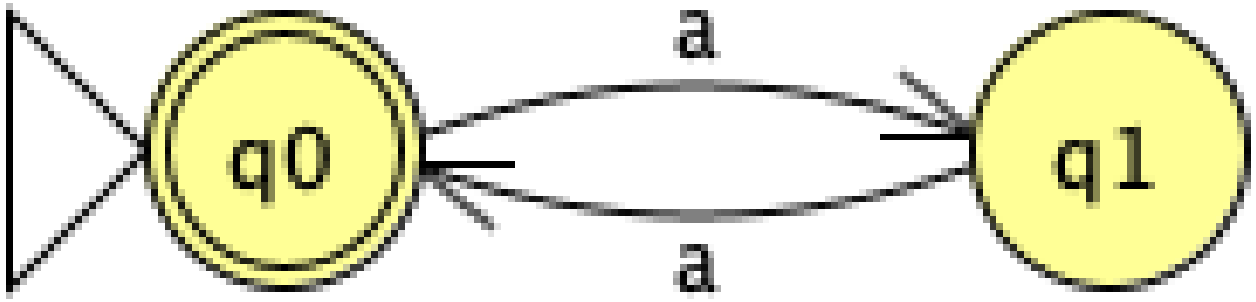
Answer

- All strings of a's and b's that don't contain any a.



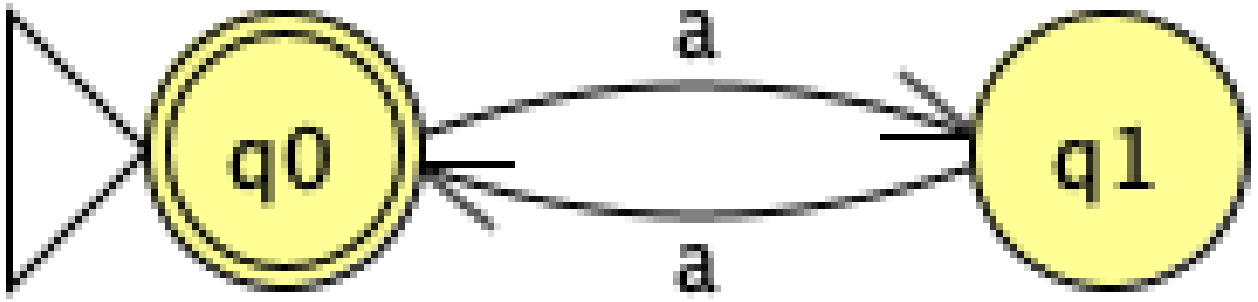
Practice Time

- What language is described by this DFA?



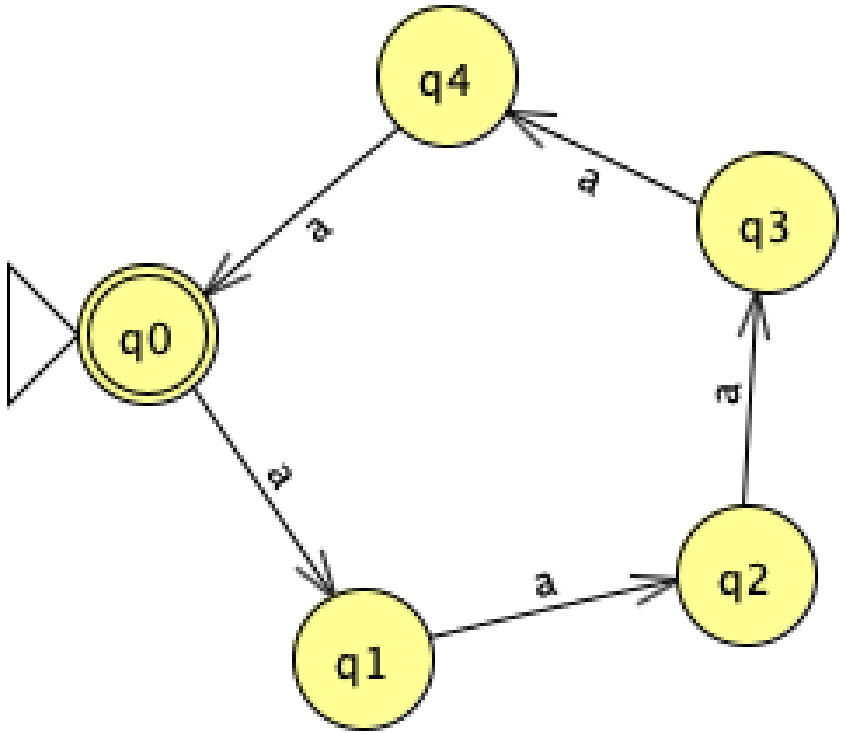
Answer

- All strings of a's that contain an even number of a's



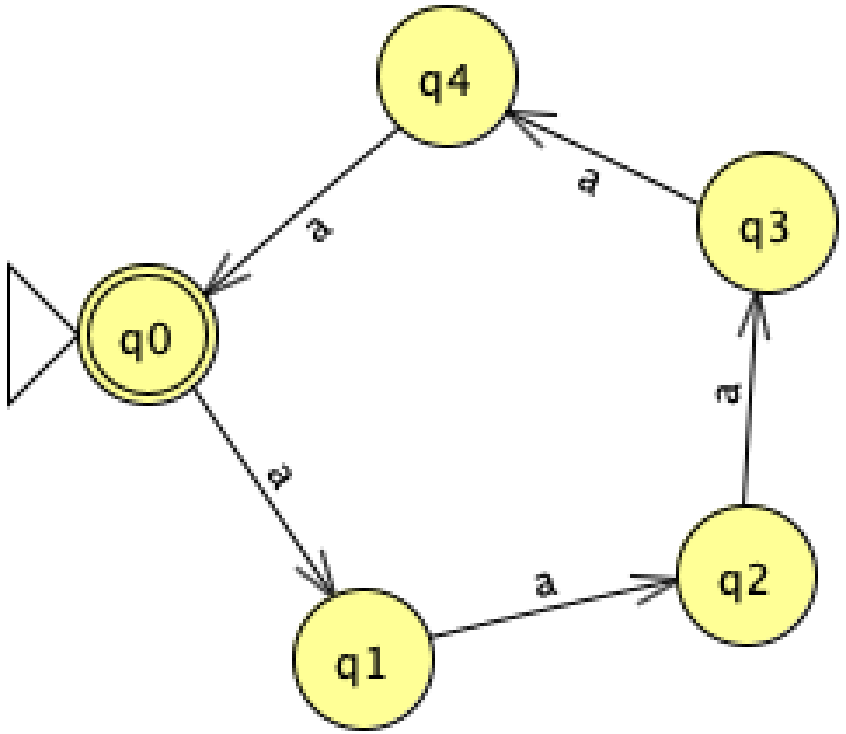
Practice Time

- What language is described by this DFA?



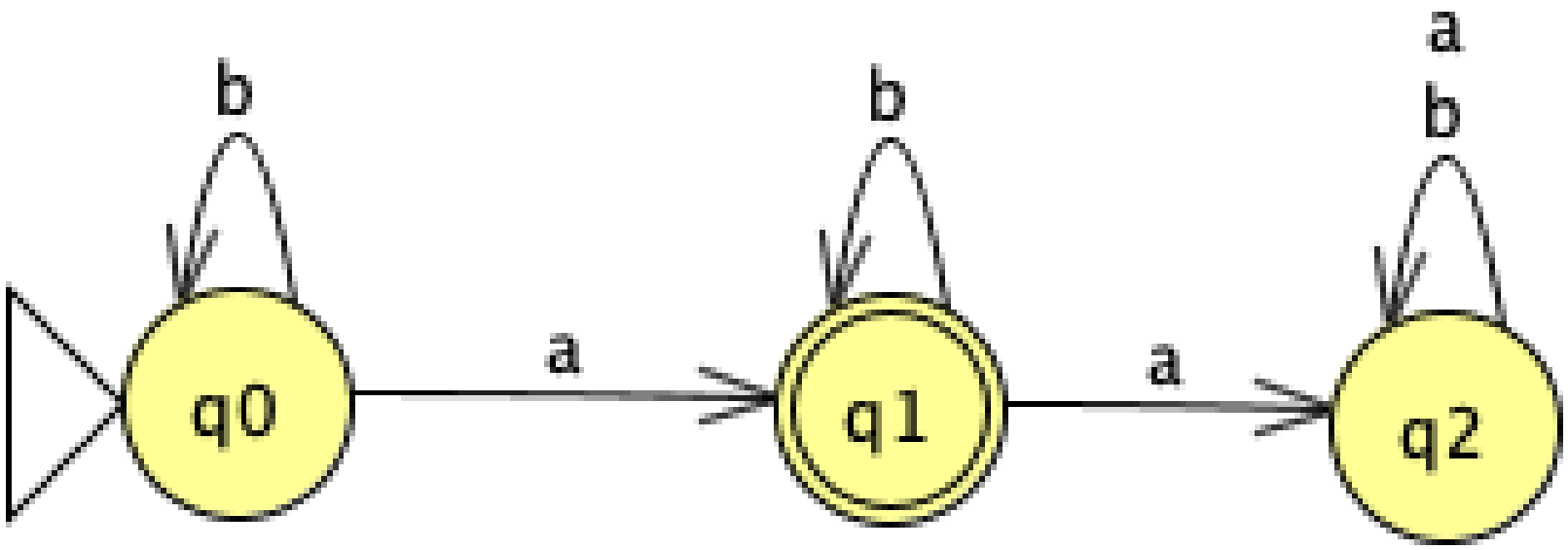
Answer

- All strings of a's that are multiples of 5



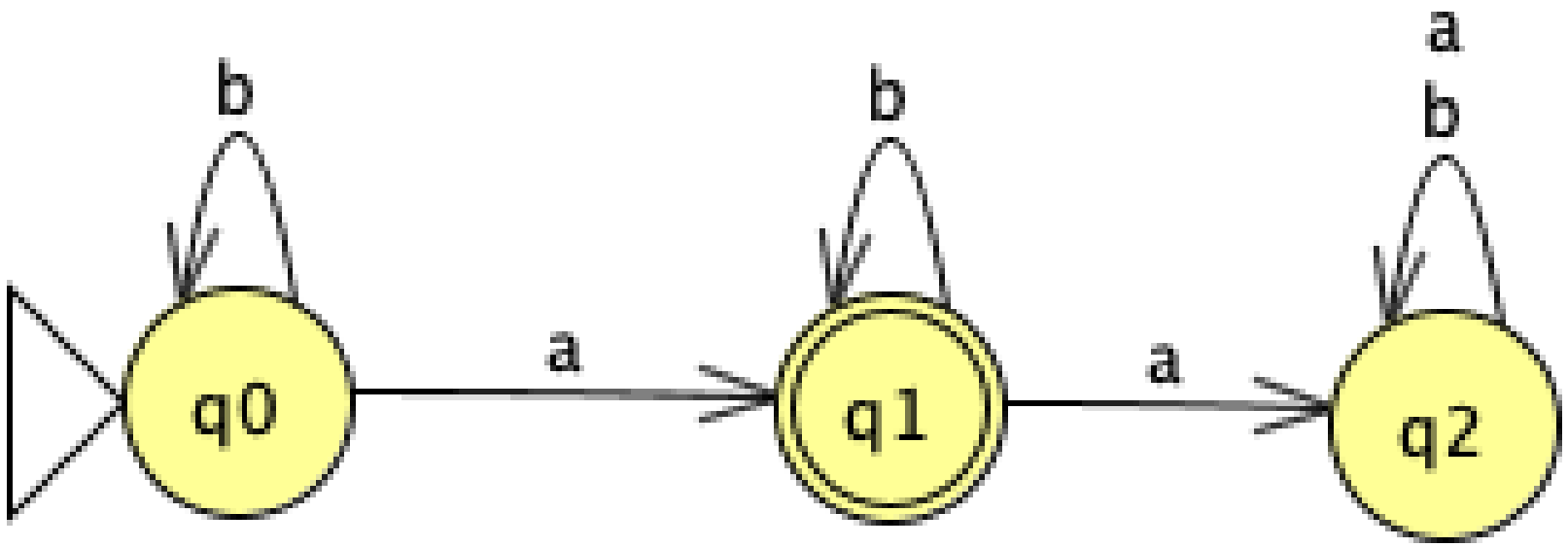
Practice Time

- What language is described by this DFA?



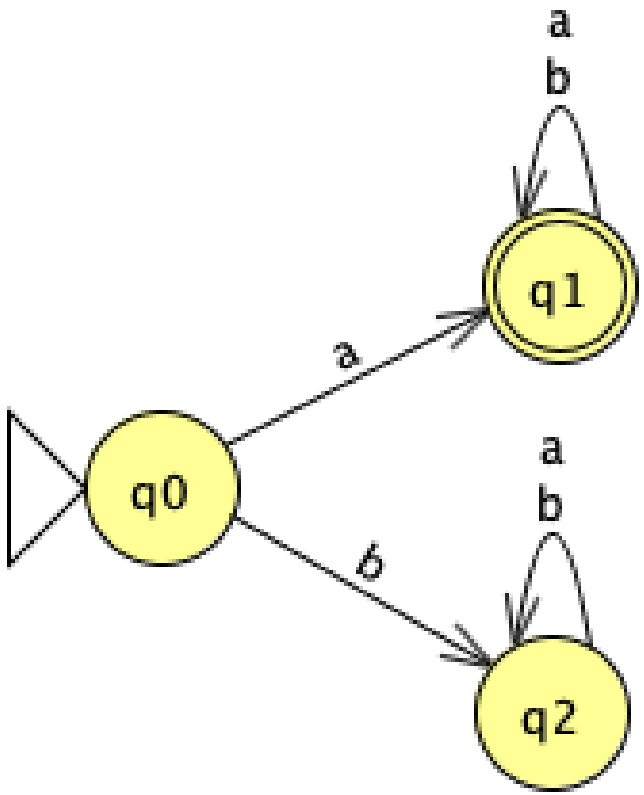
Answer

- All strings of a's and b's that contain a single a.



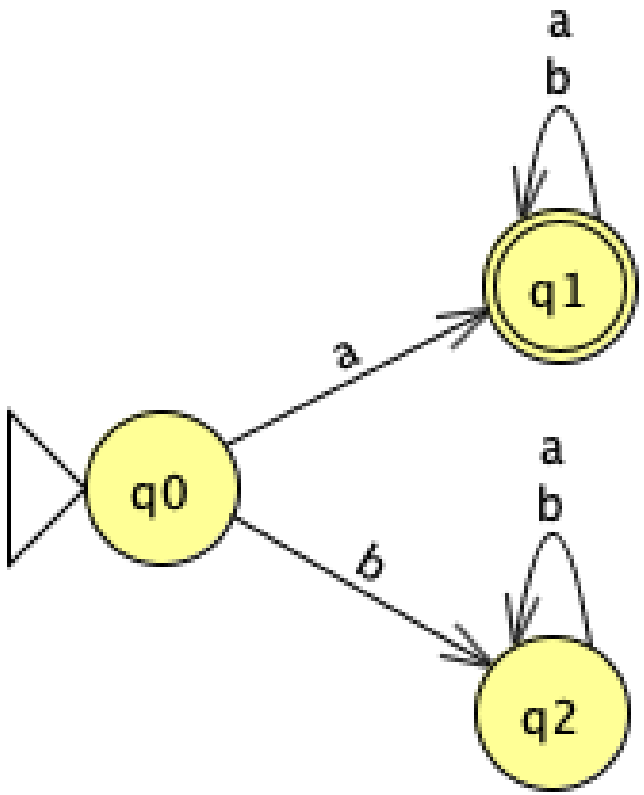
Practice Time

- What language is described by this DFA?



Answer

- All strings of a's and b's that start with a.

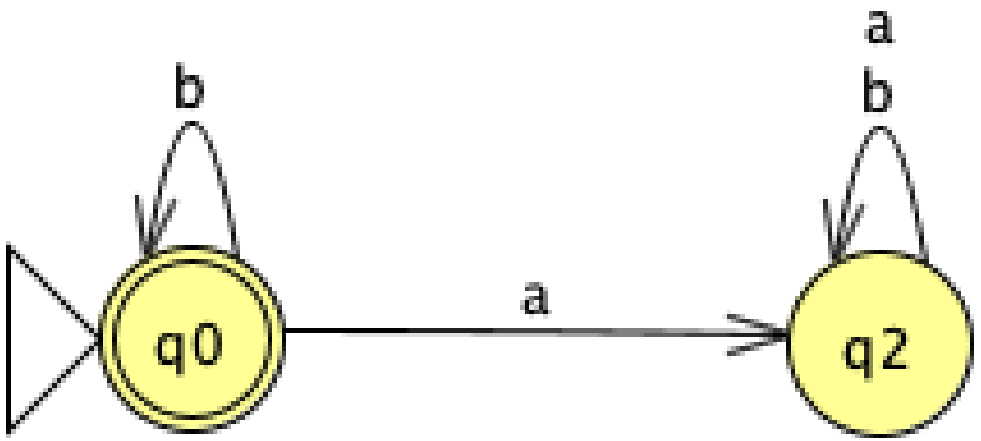


Regular Expressions

- Another way of expressing a language but *not* a computational model
- Regular expressions are built from :
  - * (Kleene star): 0 or more repetitions (highest precedence)
  - concatenation
  - | : union or "or" (lowest precedence)
  - () : used to enforce precedence
- For example,
  - a|b* denotes: a, b, bb, bbb, and so on
  - (a|b)* denotes: a, b, aa, ab, ba, bb, aaa, and so on, that is the set of all strings with no symbols other than "a" and "b"
- Let's revisit the examples we just saw and describe them using a regular expression.

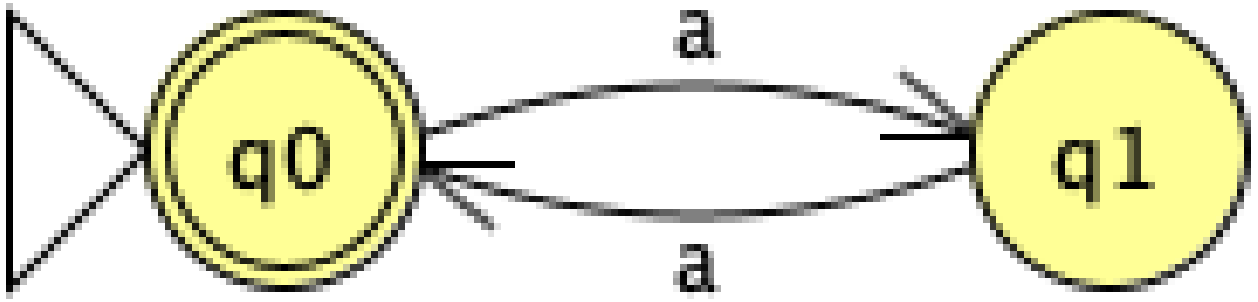
Example

- All strings of a's and b's that don't contain any a.
- Regular expression: b*



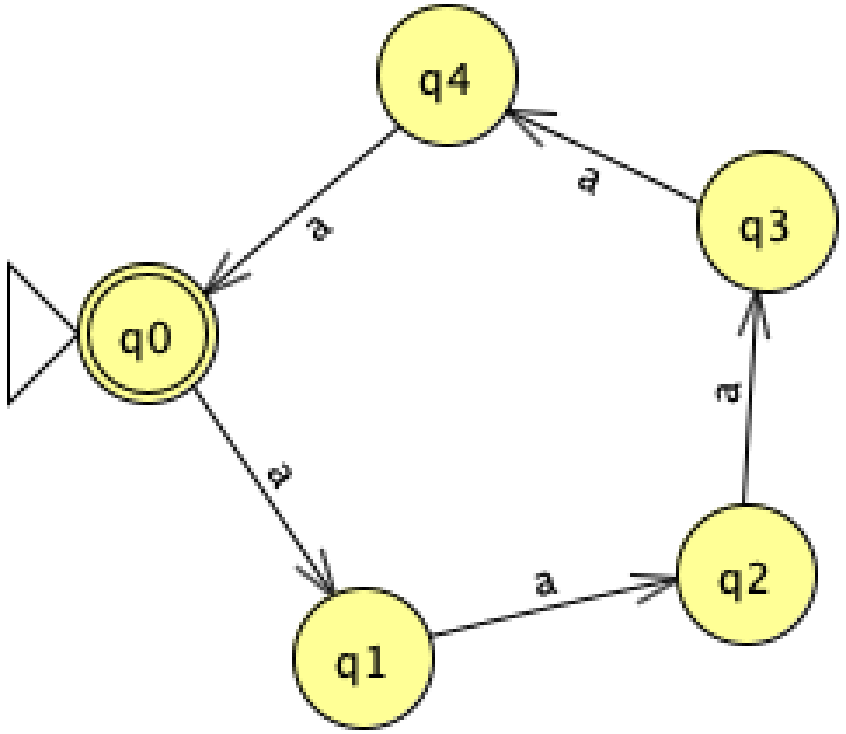
Example

- All strings of a's that contain an even number of a's
- Regular expression: (aa)*



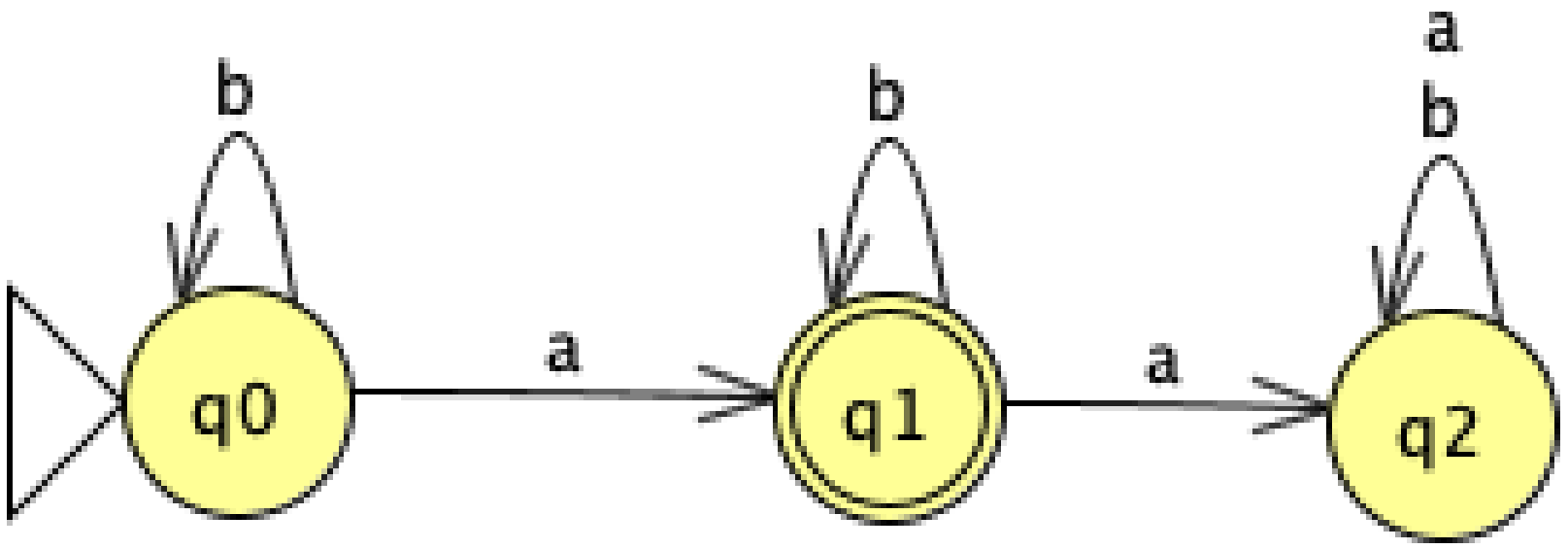
Example

- All strings of a's that are multiples of 5
- Regular expression: (aaaaa)*



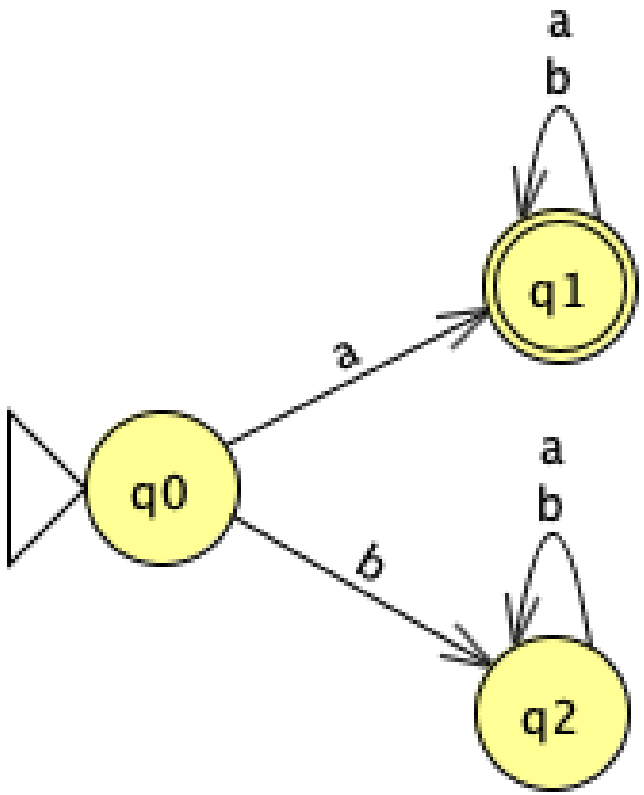
Example

- All strings of a's and b's that contain a single a.
- Regular expression: b*ab*



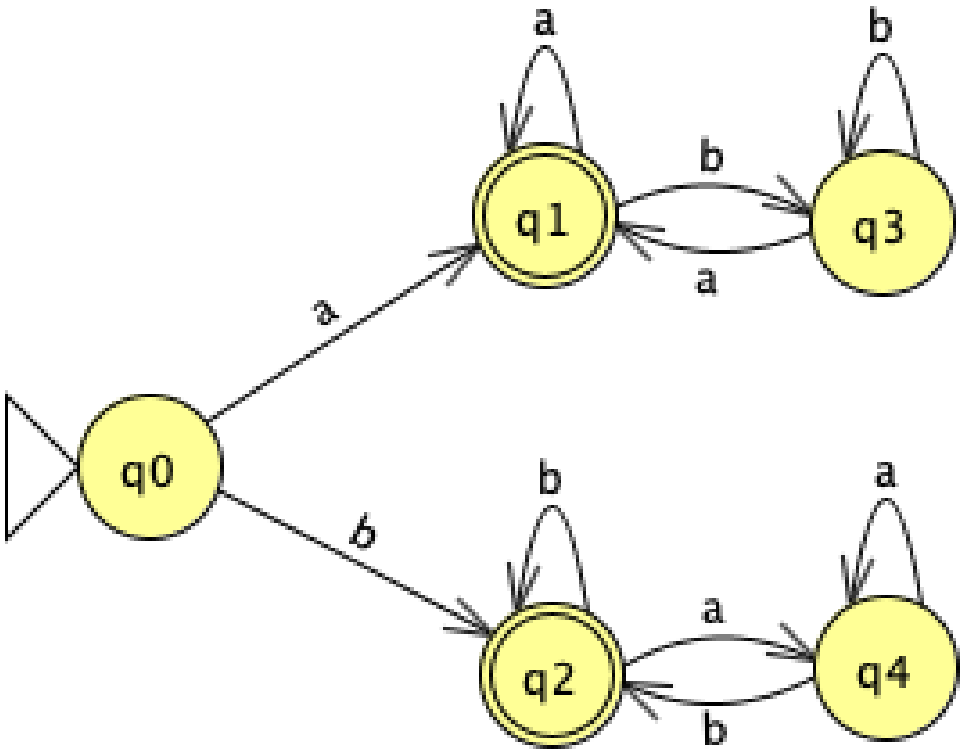
Example

- All strings of a's and b's that start with a.
- a(a|b)*



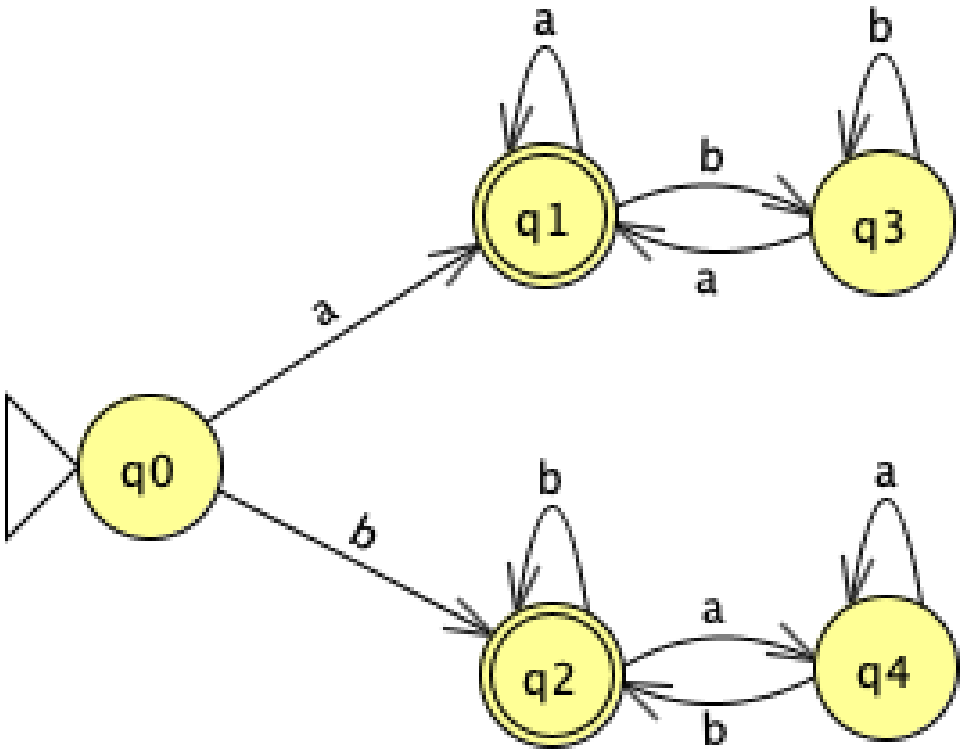
Practice Time

- What language is described by this DFA?
- What regular expression describes this language?



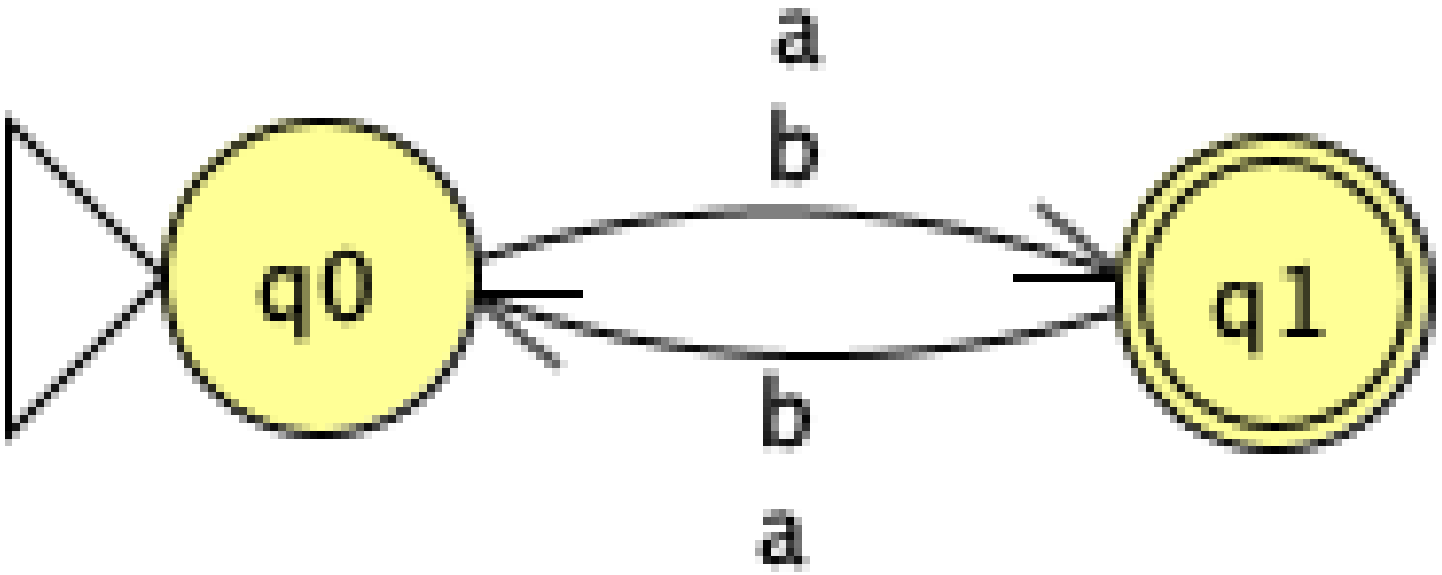
Answer

- Strings of a's and b's that start and end with the same letter
- (a(a|b)*a)|(b(a|b)*b)|a|b



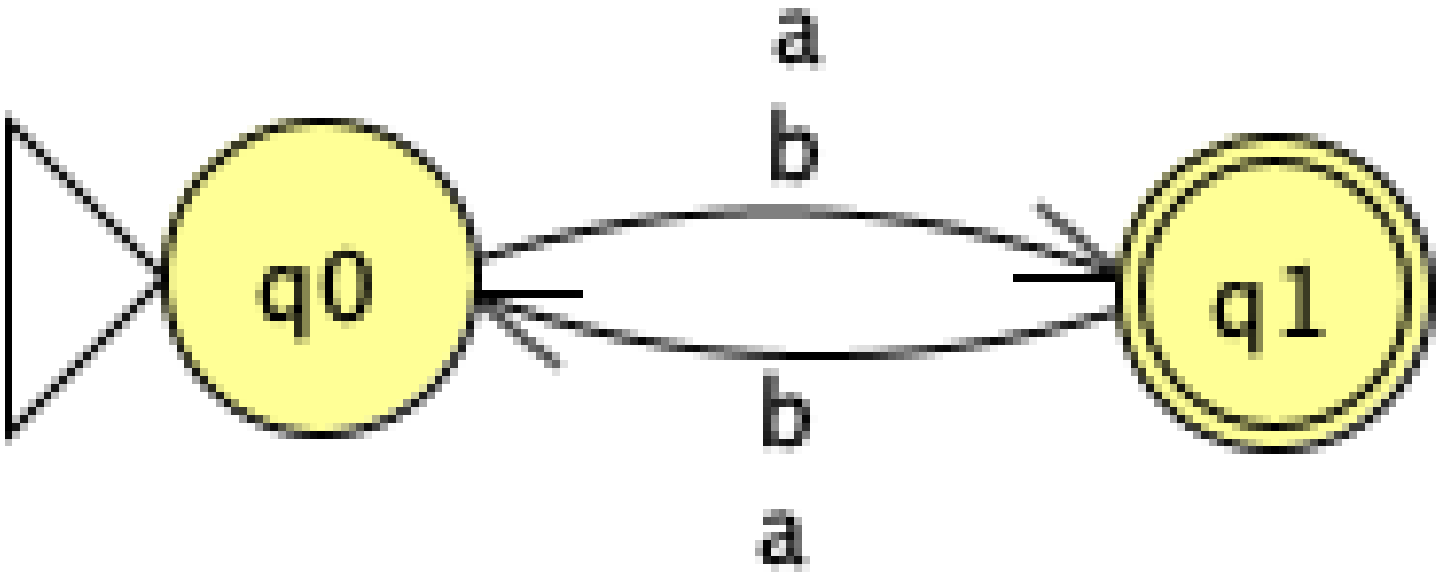
Practice Time

- What language is described by this DFA?
- What regular expression describes this language?



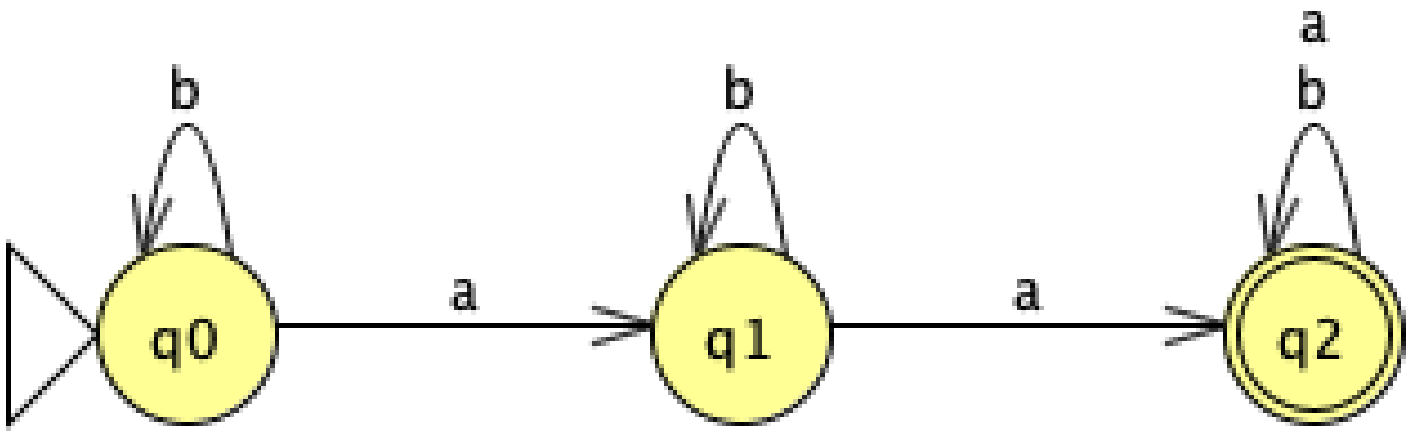
Answer

- Strings of a's and b's that have an odd length
- (a|b)((a|b)(a|b))*



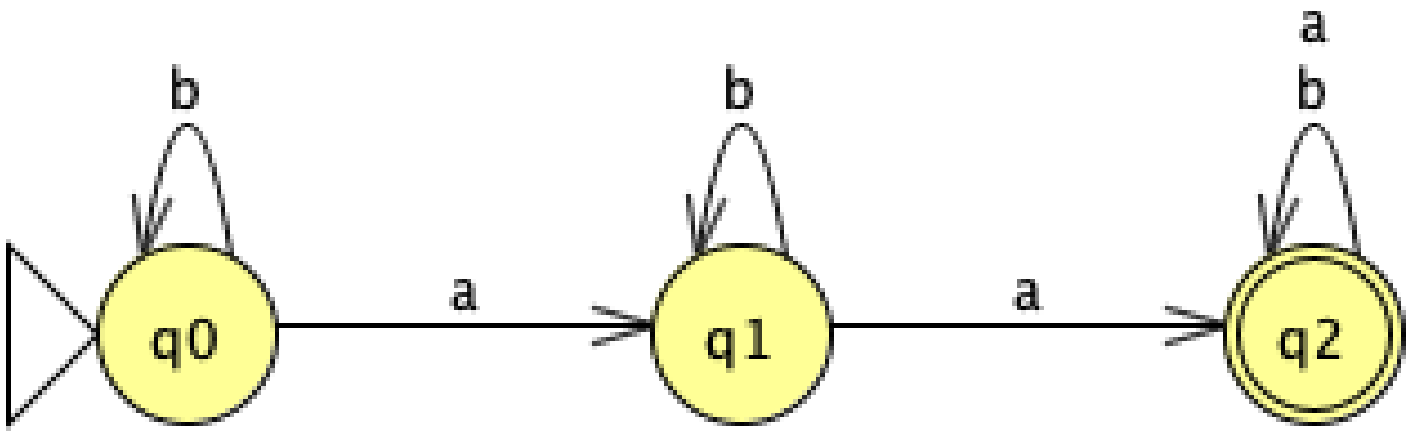
Practice Time

- What language is described by this DFA?
- What regular expression describes this language?



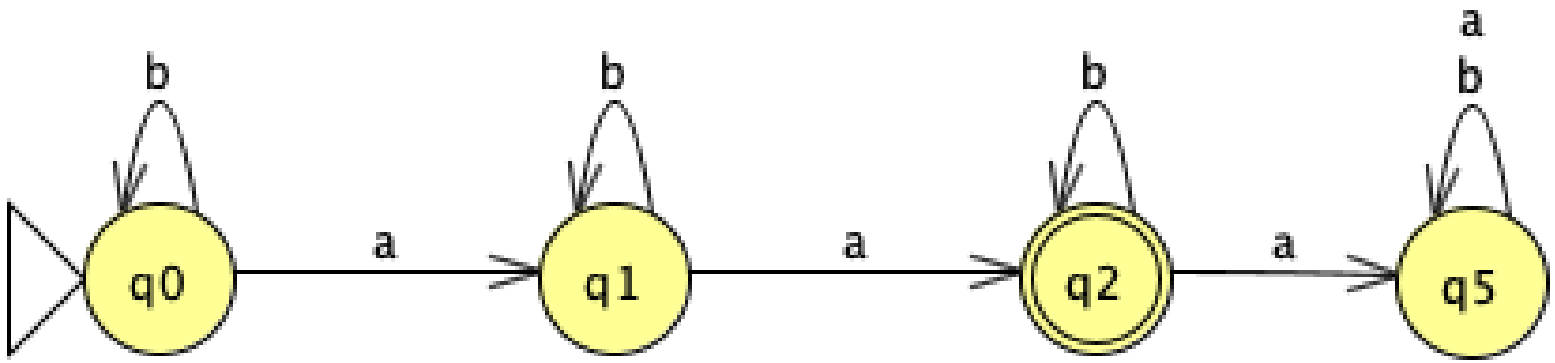
Answer

- Strings of a's and b's that have two or more a's
- (a|b)*a(a|b)*a(a|b)*
- b*ab* a(a|b)*



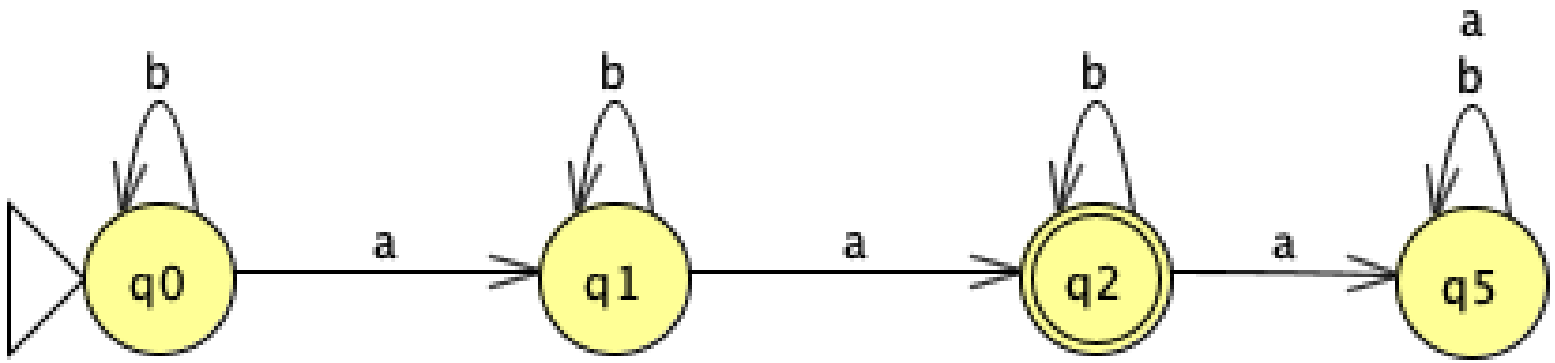
Practice Time

- What language is described by this DFA?
- What regular expression describes this language?



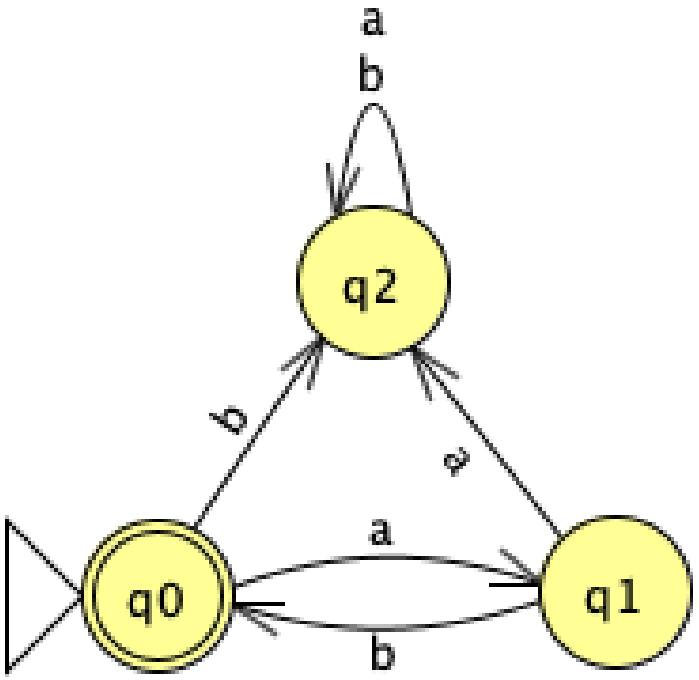
Answer

- Strings of a's and b's that have exactly two a's
- b*ab*ab*



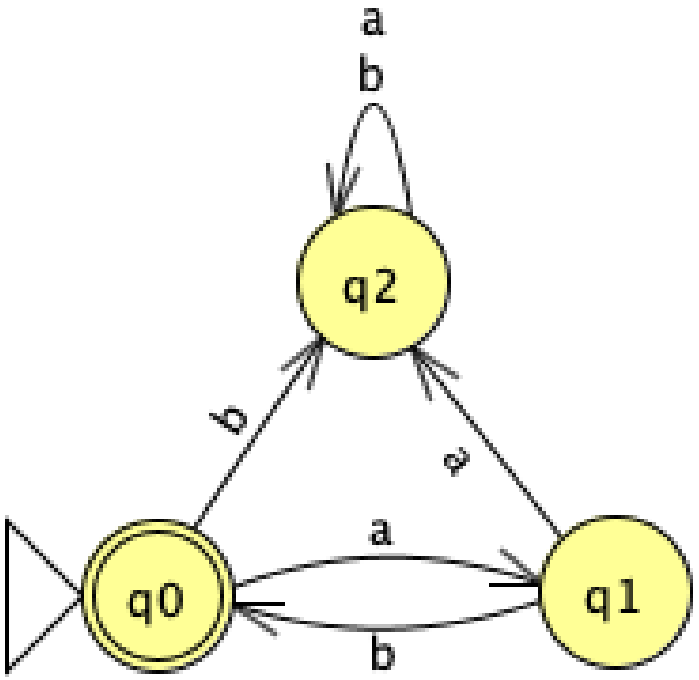
Practice Time

- What language is described by this DFA?
- What regular expression describes this language?



Answer

- Strings of a's and b's that have some number of repetitions of the sequence ab
- (ab)*



JFLAP examples:

- DFA examples