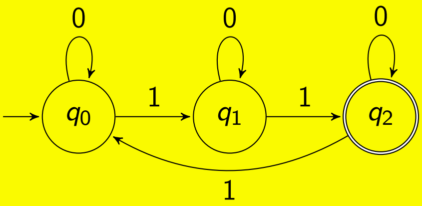
Computability and Complexity
COSC 4200

Deterministic Finite Automata

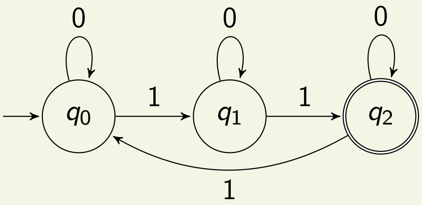
Finite Automata

Example

state diagram of a finite automaton M:



- the circles are *states*
- q_0 is the *initial state* (it has an unlabeled arrow pointint to it)
- a state with two circles is an *accepting state*
- labeled arrows are *transitions*



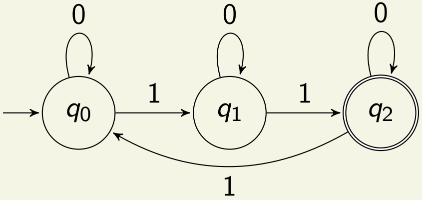
M processes an input string $w \in \{0, 1\}^*$ from left to right.

- It begins in the initial state, reading each bit in succession and taking the corresponding transitions.
- If M is in an accepting state after reading the last bit of w, then M *accepts* w.
- Otherwise, M *rejects* w.

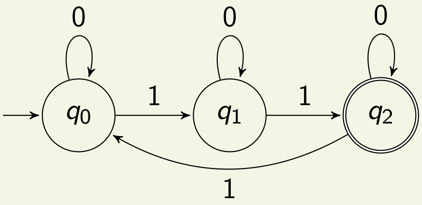
Definition

A *(deterministic) finite automaton* (often abbreviated *DFA*) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of *states*
- Σ is an *alphabet*
- $\delta : Q \times \Sigma \to Q$ is the *transition function*
- $q_0 \in Q$ is the *initial state*
- $F \subseteq Q$ is the set of *accepting states*



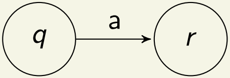
- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0, 1\}$
- q_0 is the initial state
- $F = \{q_2\}$



- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0, 1\}$
- q_0 is the initial state
- $F = \{q_2\}$
- $\delta : Q \times \Sigma \to Q$ is given by

$$\begin{array}{ll} \delta(q_0, 0) = q_0 & \delta(q_0, 1) = q_1 \\ \delta(q_1, 0) = q_1 & \delta(q_1, 1) = q_2 \\ \delta(q_2, 0) = q_2 & \delta(q_2, 1) = q_0 \end{array}$$

$q \in Q, a \in \Sigma, r \in Q$



$\delta(q, a) = r$ means the DFA transitions from state q to state r when it reads symbol a

Strings and Languages

Σ^* = all finite strings of symbols from the alphabet Σ
$|w|$ = length of the string w

Strings and Languages

Σ^* = all finite strings of symbols from the alphabet Σ
$|w|$ = length of the string w

If $\Sigma = \{0, 1\}$, then

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \ldots, 111, 0000, \ldots\}$$

Strings and Languages

Σ^* = all finite strings of symbols from the alphabet Σ
$|w|$ = length of the string w

If $\Sigma = \{0, 1\}$, then

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \ldots, 111, 0000, \ldots\}$$

ϵ is the *empty string* of length 0.
$|\epsilon| = 0$.

Strings and Languages

Σ^* = all finite strings of symbols from the alphabet Σ
$|w|$ = length of the string w

If $\Sigma = \{0, 1\}$, then

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \ldots, 111, 0000, \ldots\}$$

ϵ is the *empty string* of length 0.
$|\epsilon| = 0$.

A *language* is a subset $B \subseteq \Sigma^*$.
Languages are also called *decision problems*, primarily in the areas covered in the second half of the course.

For a string $w \in \Sigma^*$ and any number $i \leq |w|$, w_i is the i^{th} symbol of w.

$$w = w_1 w_2 \cdots w_{|w|}$$

Example

$w = 01001$

$|w| = 5$

$w_1 = 0 \quad w_2 = 1 \quad w_3 = 0 \quad w_4 = 0 \quad w_5 = 1$

Definition

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.

1. Let $w \in \Sigma^*$. We say that M *accepts* w if there is a sequence of states

   $$r_0, r_1, \ldots, r_{|w|} \in Q$$

   such that the following three conditions hold:

   1. $r_0 = q_0$
   2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for all i, $0 \leq i < |w|$
   3. $r_{|w|} \in F$.
2. The *language accepted by M* (or the *language that M recognizes*) is

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

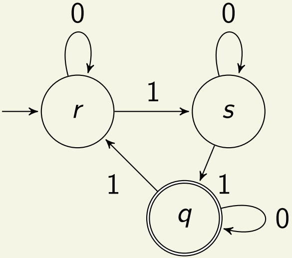
A useful, equivalent way to define acceptance uses induction.
Recall the transition function δ maps $Q \times \Sigma$ into Q.
We define an *extension* δ^* that maps $Q \times \Sigma^*$ into Q as follows:

$\delta^* : Q \times \Sigma^* \to Q$

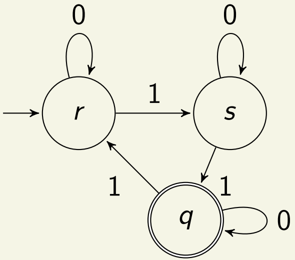
- For all $q \in Q$, $\delta^*(q, \epsilon) = q$.
- For all $q \in Q$, $w \in \Sigma^*$, and $a \in \Sigma$,

$$\delta^*(q, wa) = \delta(\delta^*(q, w), a).$$

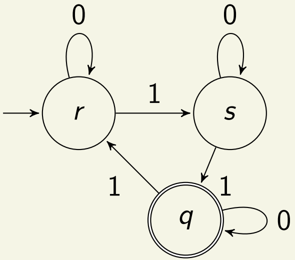
For any state q and string w, $\delta^*(q, w)$ is the state M will finish in if it starts processing w in state q.



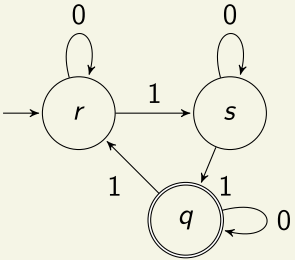
$$\delta^*(s, 1011) \quad = \quad \delta(\delta^*(s, 101), 1)$$



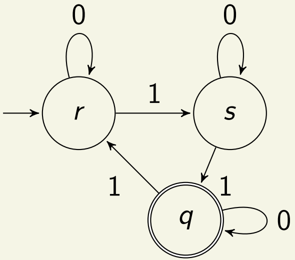
$$\begin{aligned}\delta^*(s, 1011) &= \delta(\delta^*(s, 101), 1)\\ &= \delta(\delta(\delta^*(s, 10), 1), 1)\end{aligned}$$



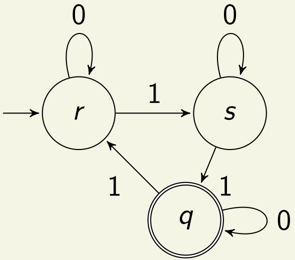
$$\begin{aligned}
\delta^*(s, 1011) &= \delta(\delta^*(s, 101), 1) \\
&= \delta(\delta(\delta^*(s, 10), 1), 1) \\
&= \delta(\delta(\delta(\delta^*(s, 1), 0), 1), 1)
\end{aligned}$$



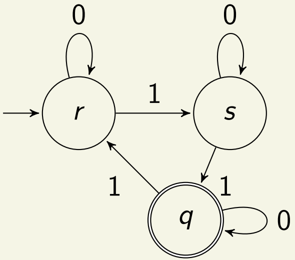
$$
\begin{aligned}
\delta^*(s,1011) &= \delta(\delta^*(s,101),1)\\
&= \delta(\delta(\delta^*(s,10),1),1)\\
&= \delta(\delta(\delta(\delta^*(s,1),0),1),1)\\
&= \delta(\delta(\delta(\delta(\delta^*(s,\epsilon),1),0),1),1)
\end{aligned}
$$



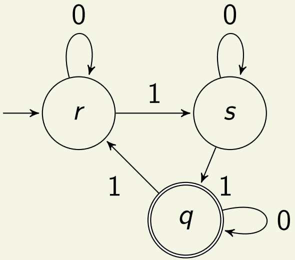
$$\begin{aligned}
\delta^*(s,1011) &= \delta(\delta^*(s,101),1)\\
&= \delta(\delta(\delta^*(s,10),1),1)\\
&= \delta(\delta(\delta(\delta^*(s,1),0),1),1)\\
&= \delta(\delta(\delta(\delta(\delta^*(s,\epsilon),1),0),1),1)\\
&= \delta(\delta(\delta(\delta(s,1),0),1),1)
\end{aligned}$$



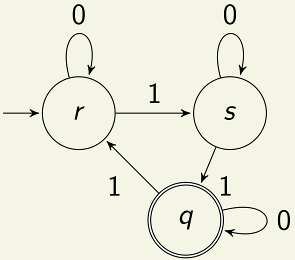
$$
\begin{aligned}
\delta^*(s, 1011) &= \delta(\delta^*(s, 101), 1) \\
&= \delta(\delta(\delta^*(s, 10), 1), 1) \\
&= \delta(\delta(\delta(\delta^*(s, 1), 0), 1), 1) \\
&= \delta(\delta(\delta(\delta(\delta^*(s, \epsilon), 1), 0), 1), 1) \\
&= \delta(\delta(\delta(\delta(s, 1), 0), 1), 1) \\
&= \delta(\delta(\delta(q, 0), 1), 1)
\end{aligned}
$$



$$
\begin{aligned}
\delta^*(s,1011) &= \delta(\delta^*(s,101),1)\\
&= \delta(\delta(\delta^*(s,10),1),1)\\
&= \delta(\delta(\delta(\delta^*(s,1),0),1),1)\\
&= \delta(\delta(\delta(\delta(\delta^*(s,\epsilon),1),0),1),1)\\
&= \delta(\delta(\delta(\delta(s,1),0),1),1)\\
&= \delta(\delta(\delta(q,0),1),1)\\
&= \delta(\delta(q,1),1)
\end{aligned}
$$



$$
\begin{aligned}
\delta^*(s,1011) &= \delta(\delta^*(s,101),1)\\
&= \delta(\delta(\delta^*(s,10),1),1)\\
&= \delta(\delta(\delta(\delta^*(s,1),0),1),1)\\
&= \delta(\delta(\delta(\delta(\delta^*(s,\epsilon),1),0),1),1)\\
&= \delta(\delta(\delta(\delta(s,1),0),1),1)\\
&= \delta(\delta(\delta(q,0),1),1)\\
&= \delta(\delta(q,1),1)\\
&= \delta(r,1)
\end{aligned}
$$



$$
\begin{aligned}
\delta^*(s, 1011) &= \delta(\delta^*(s, 101), 1) \\
&= \delta(\delta(\delta^*(s, 10), 1), 1) \\
&= \delta(\delta(\delta(\delta^*(s, 1), 0), 1), 1) \\
&= \delta(\delta(\delta(\delta(\delta^*(s, \epsilon), 1), 0), 1), 1) \\
&= \delta(\delta(\delta(\delta(s, 1), 0), 1), 1) \\
&= \delta(\delta(\delta(q, 0), 1), 1) \\
&= \delta(\delta(q, 1), 1) \\
&= \delta(r, 1) \\
&= s
\end{aligned}
$$

Alternative Definitions

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Let $w \in \Sigma^*$. We say that M accepts w if

$$\delta^*(q_0, w) \in F.$$

The *language accepted by M* is

$$L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}.$$

Regular Languages

Definition

A language $B \subseteq \Sigma^*$ is *regular* if $B = L(M)$ for some DFA M.

$B = L(M)$ means that:

$$
\begin{aligned}
x \in B &\quad\Rightarrow\quad M \text{ accepts } x \\
x \notin B &\quad\Rightarrow\quad M \text{ rejects } x
\end{aligned}
$$

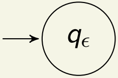
Designing DFAs

Show that the following languages are regular.

1. $\{w \in \{a, b\}^* \mid w \text{ begins with } ab\}$.
2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$.
3. $\{x010y \mid x, y \in \{0, 1\}^*\}$.
4. $\left\{w \in \{a, b\}^* \;\middle|\; \begin{array}{l} w \text{ has an odd number of } a\text{'s} \\ \text{and an even number of } b\text{'s} \end{array} \right\}$.
5. $\left\{w \in \{0, 1\}^* \;\middle|\; \text{the first bit of } w \text{ equals the last bit} \right\}$
6. $\{a^n \mid n \text{ is a multiple of } 6\}$.

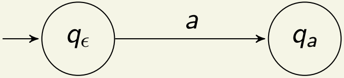
Designing DFAs

1. $\{w \in \{a, b\}^* \mid w \text{ begins with } ab\}$



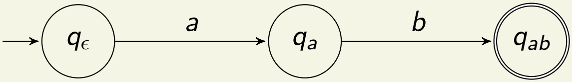
Designing DFAs

1. $\{w \in \{a, b\}^* \mid w \text{ begins with } ab\}$



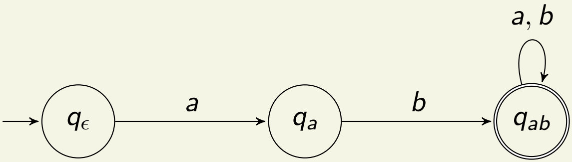
Designing DFAs

1. $\{w \in \{a, b\}^* \mid w \text{ begins with } ab\}$



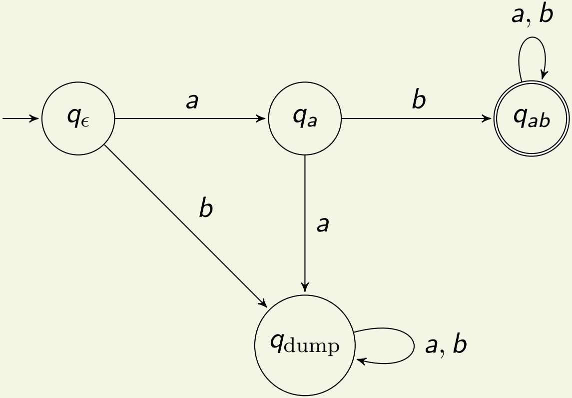
Designing DFAs

1. $\{w \in \{a, b\}^* \mid w \text{ begins with } ab\}$



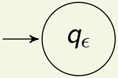
Designing DFAs

1. $\{w \in \{a, b\}^* \mid w \text{ begins with } ab\}$



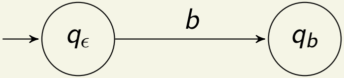
Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$



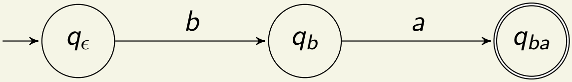
Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$



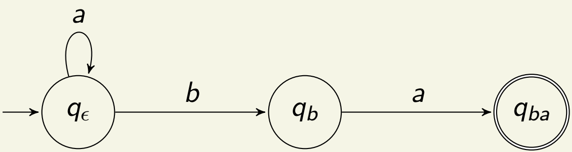
Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$



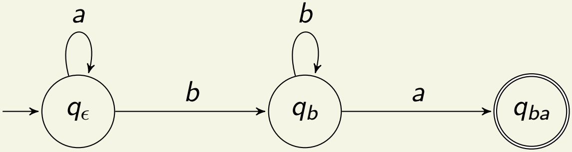
Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$



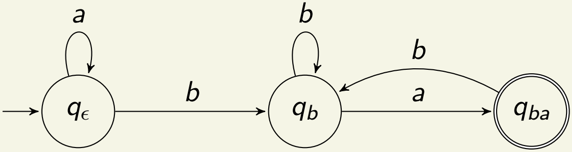
Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$



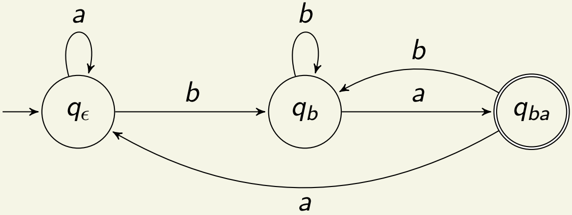
Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$



Designing DFAs

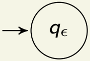
2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$



Designing DFAs

2. $\{w \in \{a,b\}^* \mid w \text{ ends with } ba\}$

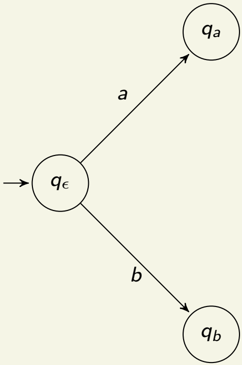
A simpler way (but with more states): use a tree to remember the last two symbols.



Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$

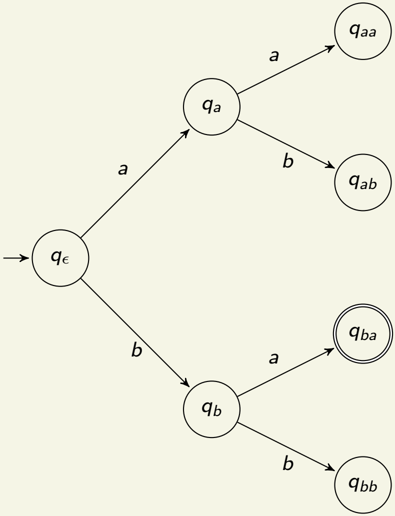
A simpler way (but with more states): use a tree to remember the last two symbols.



Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$

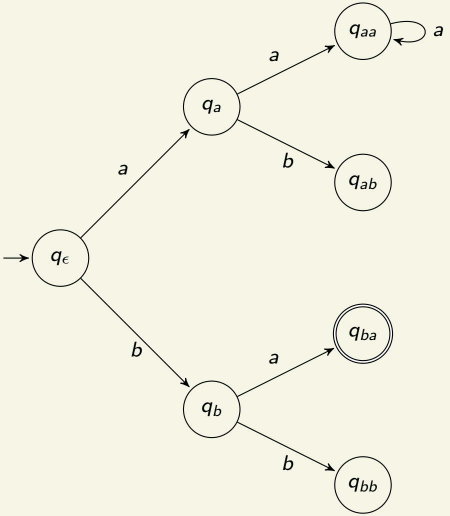
A simpler way (but with more states): use a tree to remember the last two symbols.



Designing DFAs

2. $\{w \in \{a,b\}^* \mid w \text{ ends with } ba\}$

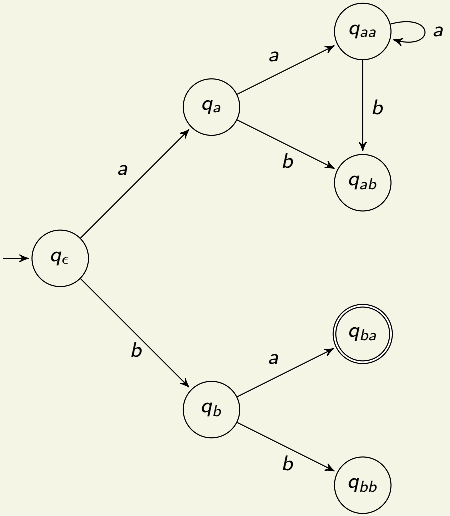
A simpler way (but with more states): use a tree to remember the last two symbols.



Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$

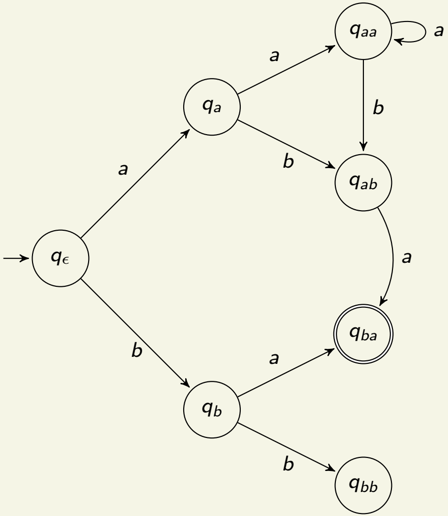
A simpler way (but with more states): use a tree to remember the last two symbols.



Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$

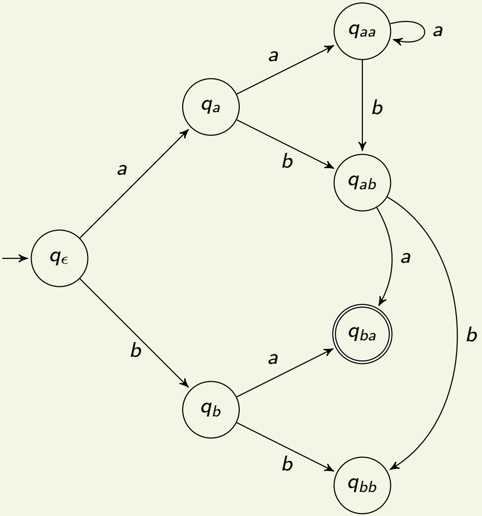
A simpler way (but with more states): use a tree to remember the last two symbols.



Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$

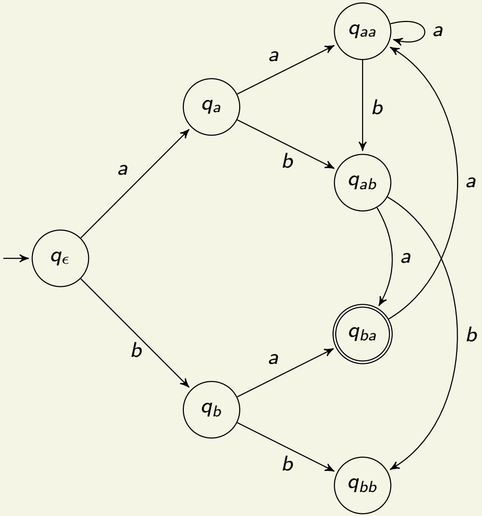
A simpler way (but with more states): use a tree to remember the last two symbols.



Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$

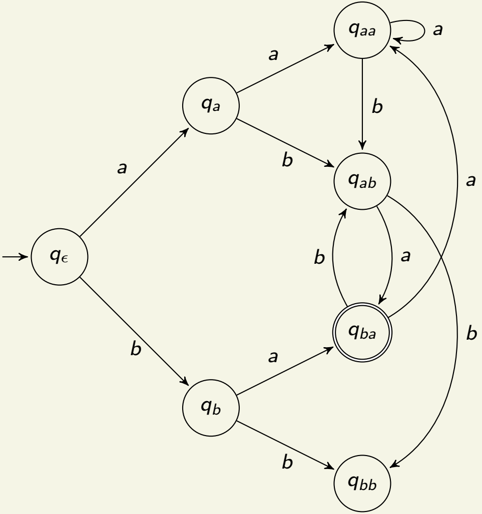
A simpler way (but with more states): use a tree to remember the last two symbols.



Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$

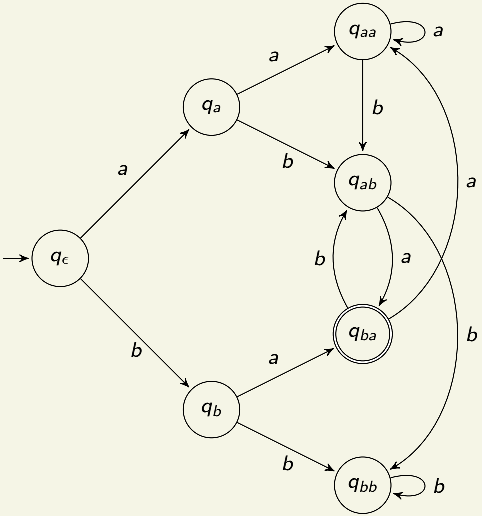
A simpler way (but with more states): use a tree to remember the last two symbols.



Designing DFAs

2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$

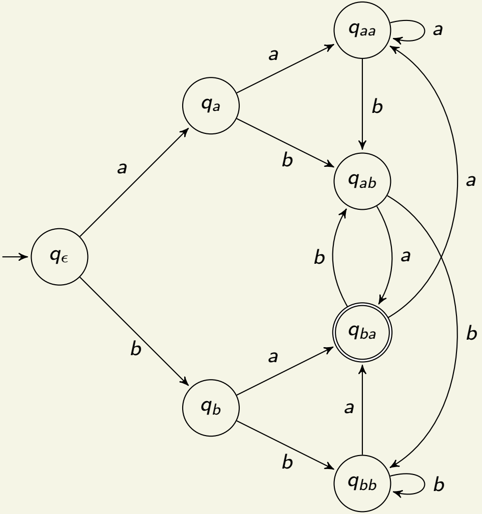
A simpler way (but with more states): use a tree to remember the last two symbols.



Designing DFAs

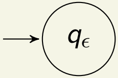
2. $\{w \in \{a, b\}^* \mid w \text{ ends with } ba\}$

A simpler way (but with more states): use a tree to remember the last two symbols.



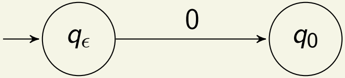
Designing DFAs

3. $\{x010y \mid x, y \in \{0,1\}^*\}$
$= \{w \in \{0,1\}^* \mid w \text{ contains } 010 \text{ as a substring}\}$



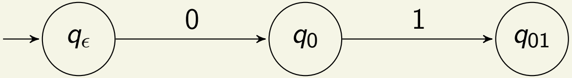
Designing DFAs

3. $\{x010y \mid x, y \in \{0,1\}^*\}$
   $= \{w \in \{0,1\}^* \mid w \text{ contains } 010 \text{ as a substring}\}$



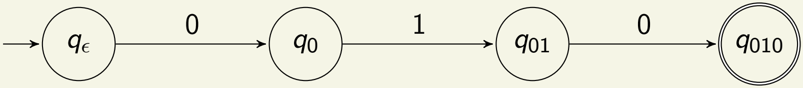
Designing DFAs

3. $\{x010y \mid x, y \in \{0,1\}^*\}$
   $= \{w \in \{0,1\}^* \mid w \text{ contains } 010 \text{ as a substring}\}$



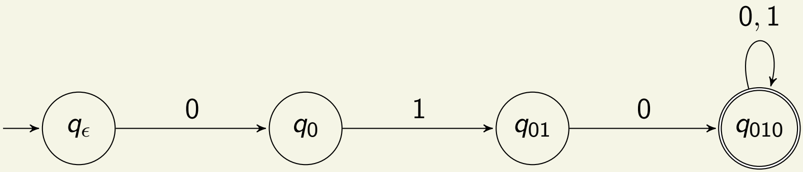
Designing DFAs

3. $\{x010y \mid x, y \in \{0,1\}^*\}$
   $= \{w \in \{0,1\}^* \mid w \text{ contains } 010 \text{ as a substring}\}$



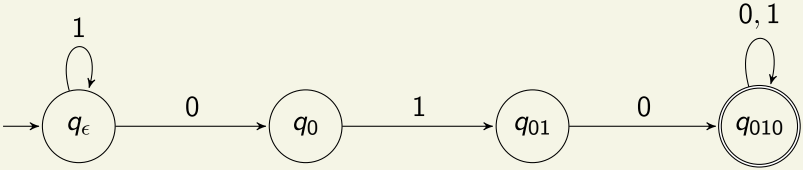
Designing DFAs

3. $\{x010y \mid x, y \in \{0,1\}^*\}$
   $= \{w \in \{0,1\}^* \mid w \text{ contains } 010 \text{ as a substring}\}$



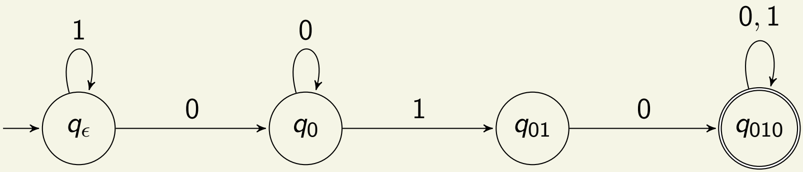
Designing DFAs

3. $\{x010y \mid x, y \in \{0,1\}^*\}$
   $= \{w \in \{0,1\}^* \mid w \text{ contains } 010 \text{ as a substring}\}$



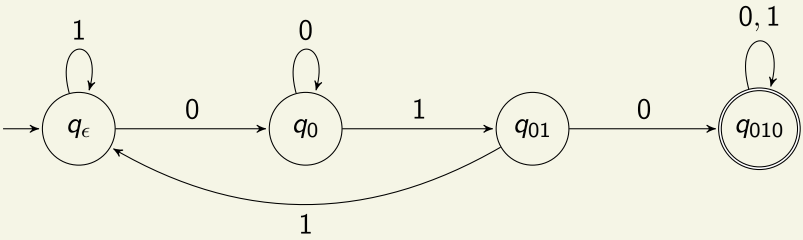
Designing DFAs

3. $\{x010y \mid x, y \in \{0,1\}^*\}$
   $= \{w \in \{0,1\}^* \mid w \text{ contains } 010 \text{ as a substring}\}$



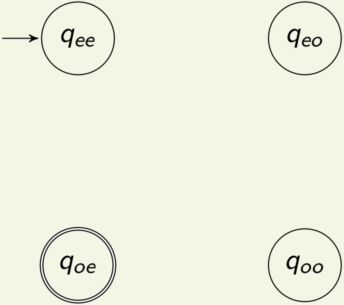
Designing DFAs

3. $\{x010y \mid x, y \in \{0,1\}^*\}$
   $= \{w \in \{0,1\}^* \mid w \text{ contains } 010 \text{ as a substring}\}$



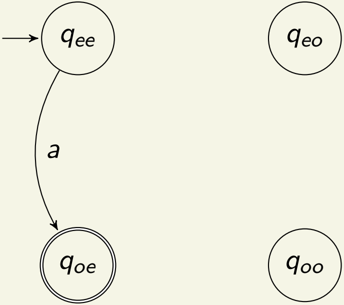
Designing DFAs

4. $\left\{ w \in \{a, b\}^* \;\middle|\; \begin{array}{l} w \text{ has an odd number of } a\text{'s} \\ \text{and an even number of } b\text{'s} \end{array} \right\}$



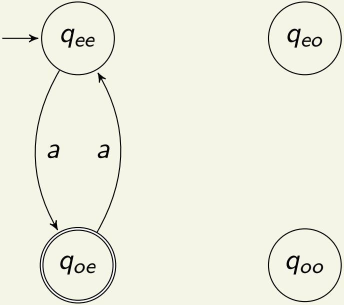
Designing DFAs

4. $\left\{ w \in \{a, b\}^* \;\middle|\; \begin{array}{l} w \text{ has an odd number of } a\text{'s} \\ \text{and an even number of } b\text{'s} \end{array} \right\}$



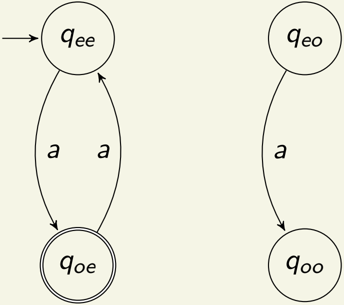
Designing DFAs

4. $\left\{ w \in \{a, b\}^* \;\middle|\; \begin{array}{l} w \text{ has an odd number of } a\text{'s} \\ \text{and an even number of } b\text{'s} \end{array} \right\}$



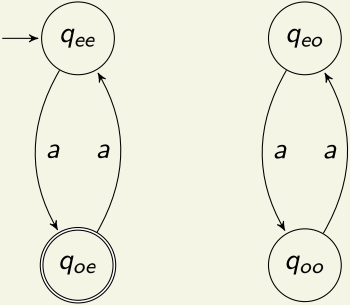
Designing DFAs

4. $\left\{ w \in \{a, b\}^* \;\middle|\; \begin{array}{l} w \text{ has an odd number of } a\text{'s} \\ \text{and an even number of } b\text{'s} \end{array} \right\}$



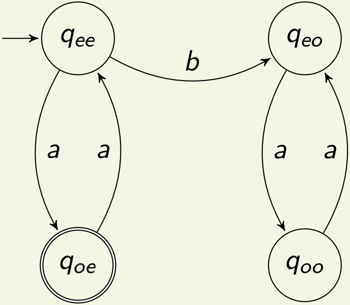
Designing DFAs

4. $\left\{ w \in \{a,b\}^* \;\middle|\; \begin{array}{l} w \text{ has an odd number of } a\text{'s} \\ \text{and an even number of } b\text{'s} \end{array} \right\}$



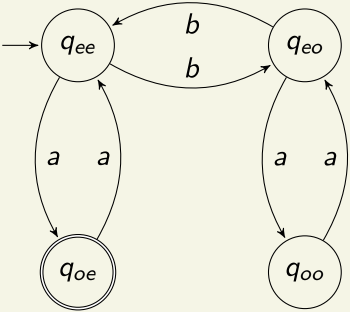
Designing DFAs

4. $\left\{ w \in \{a, b\}^* \;\middle|\; \begin{array}{l} w \text{ has an odd number of } a\text{'s} \\ \text{and an even number of } b\text{'s} \end{array} \right\}$



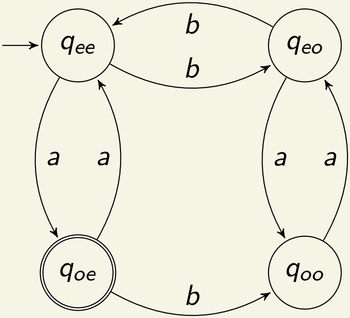
Designing DFAs

4. $\left\{ w \in \{a, b\}^* \;\middle|\; \begin{array}{l} w \text{ has an odd number of } a\text{'s} \\ \text{and an even number of } b\text{'s} \end{array} \right\}$



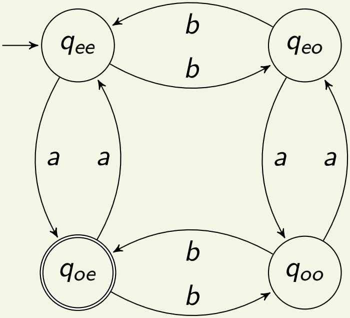
Designing DFAs

4. $\left\{ w \in \{a, b\}^* \;\middle|\; \begin{array}{l} w \text{ has an odd number of } a\text{'s} \\ \text{and an even number of } b\text{'s} \end{array} \right\}$



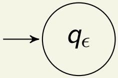
Designing DFAs

4. $\left\{ w \in \{a, b\}^* \;\middle|\; \begin{array}{l} w \text{ has an odd number of } a\text{'s} \\ \text{and an even number of } b\text{'s} \end{array} \right\}$



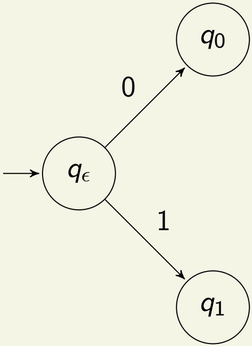
Designing DFAs

5. $\{w \in \{0,1\}^* \mid |w| \geq 2 \text{ and the first bit of } w \text{ equals the last bit } \}$



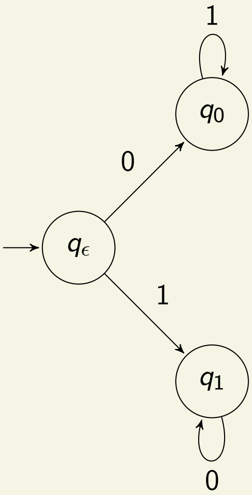
Designing DFAs

5. $\{w \in \{0,1\}^* \mid |w| \geq 2 \text{ and the first bit of } w \text{ equals the last bit } \}$



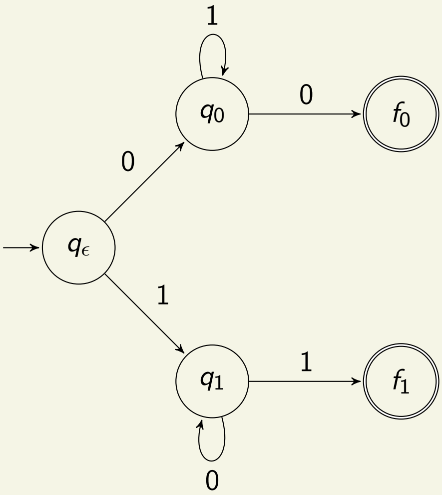
Designing DFAs

5. $\{w \in \{0,1\}^* \mid |w| \geq 2 \text{ and the first bit of } w \text{ equals the last bit } \}$



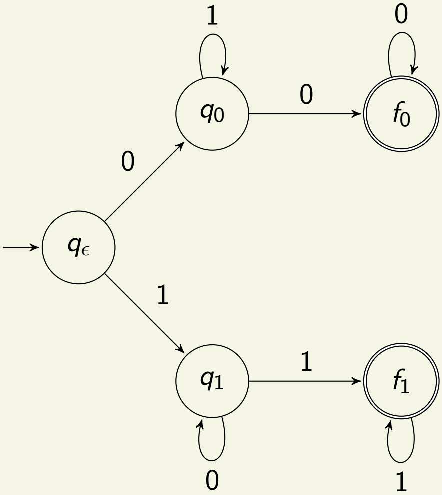
Designing DFAs

5. $\{w \in \{0,1\}^* \mid |w| \geq 2 \text{ and the first bit of } w \text{ equals the last bit } \}$



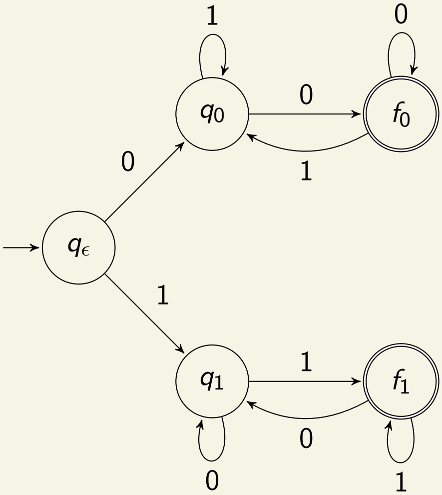
Designing DFAs

5. $\{w \in \{0,1\}^* \mid |w| \geq 2 \text{ and the first bit of } w \text{ equals the last bit } \}$



Designing DFAs

5. $\{w \in \{0,1\}^* \mid |w| \geq 2 \text{ and the first bit of } w \text{ equals the last bit } \}$

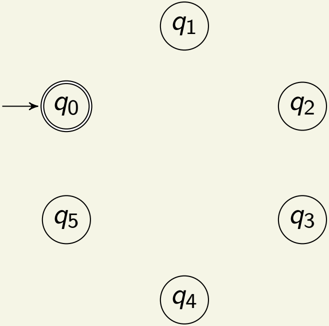


Designing DFAs

6. $\{a^n \mid n \text{ is a multiple of } 6\}$

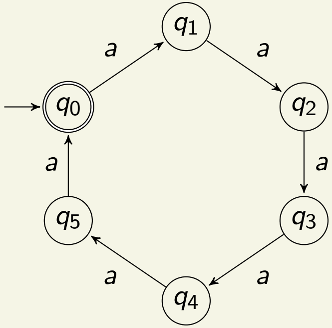
Designing DFAs

6. $\{a^n \mid n \text{ is a multiple of } 6\}$



Designing DFAs

6. $\{a^n \mid n \text{ is a multiple of } 6\}$



Designing DFAs - Summary

1. Each state in a DFA “remembers” or represents something about the input string.

Designing DFAs - Summary

1. Each state in a DFA “remembers” or represents something about the input string.
2. Design each state with a clear idea of what it represents.

Designing DFAs - Summary

1. Each state in a DFA "remembers" or represents something about the input string.
2. Design each state with a clear idea of what it represents.
3. Each state in a DFA should have exactly one transition for each alphabet symbol.

Designing DFAs - Summary

1. Each state in a DFA "remembers" or represents something about the input string.
2. Design each state with a clear idea of what it represents.
3. Each state in a DFA should have exactly one transition for each alphabet symbol.
4. Transitions move the DFA from one state to another as an input symbol is read. Think about how to use these transitions to update what the DFA remembers as it reads each symbol.

Designing DFAs - Summary

1. Each state in a DFA "remembers" or represents something about the input string.
2. Design each state with a clear idea of what it represents.
3. Each state in a DFA should have exactly one transition for each alphabet symbol.
4. Transitions move the DFA from one state to another as an input symbol is read. Think about how to use these transitions to update what the DFA remembers as it reads each symbol.
5. Include comments explaining how your DFA works. (Analogous to commenting code.)

Designing DFAs - Summary

1. Each state in a DFA "remembers" or represents something about the input string.
2. Design each state with a clear idea of what it represents.
3. Each state in a DFA should have exactly one transition for each alphabet symbol.
4. Transitions move the DFA from one state to another as an input symbol is read. Think about how to use these transitions to update what the DFA remembers as it reads each symbol.
5. Include comments explaining how your DFA works. (Analogous to commenting code.)
6. Test your DFA on sample inputs both in and out of the language. Check whether it operates as expected. Try to find examples that make it fail. (Analogous to software testing.)