Computer Science 313

Designing a Deterministic Finite Automaton

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #3

Goal for Today

Goal for Today: Identification of a ***Process*** that you can use to design a deterministic finite automaton for a given language.

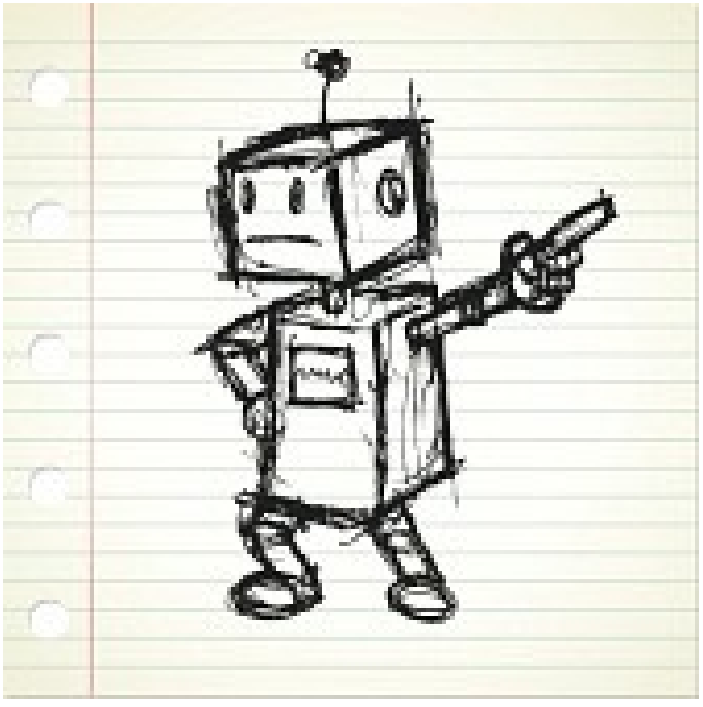
Good Advice from "Introduction to the Theory of Computation:"

> "Whether it be automata or artwork, ***design is a creative process.*** As such it cannot be reduced to a simple recipe or formula. However, you might find a particular approach helpful when designing various types of automata. That is, put *yourself* in the place of the machine you are trying to design and then see how you would go about performing the machine's task. ***Pretending that you are the machine is a psychological trick that helps engage your whole mind in the design process.***"

“Being a DFA”

That is:

Be the Machine!



The Rules for "Being a DFA"

Recall that a ***deterministic finite automaton*** (DFA) is (modelled formally as) a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite, nonempty set of ***states***;
- Σ is a finite, nonempty set of symbols called an ***alphabet***;
- $\delta : Q \times \Sigma \to Q$ is a total function called a ***transition function***;
- $q_0 \in Q$ is the ***start state***;
- $F \subseteq Q$ is the set of ***accepting states***.

The function δ can be used to define an ***extended transition function*** $\delta^\star : Q \times \Sigma^\star \to Q$, and the automaton M ***accepts*** a string $\omega \in \Sigma^\star$ if and only if

$$\delta^\star(q_0, \omega) \in F.$$

The Rules for “Being a DFA”

- As an automaton, you process a string ***one symbol at a time***, and you move ***from state to state*** when you do so.
- The *only* information that you (as the automaton) know about what has seen so far is represented by the ***current state***.
- A deterministic finite automaton has only ***finitely many states***, so only a *finite* amount of information can be remembered, even though the input string can be arbitrarily long.

What Does a DFA Do?

This leads to the following question, if you are going to "be the automaton."

What do you need to remember about the part of the string you have seen so far?

Note: You do *not* need to be sure about the answer in order to start! Answering this question is the step that requires the most imagination and creativity in this design process.

A ***recommendation*** about how to figure out how to do this is given near the end of these notes.

Ongoing Examples

The process being developed will be used to design deterministic finite automata for each of the following languages in $\Sigma^\star$, for $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$:

(a) $L_1 = \{\omega \in \Sigma^\star \mid \omega$ includes at least one "$\mathtt{a}$"$\}$

(b) $L_2 = \{\omega \in \Sigma^\star \mid$ either ω does not include an "$\mathtt{a}$" or ω does not include a "$\mathtt{b}$"$\}$

Clarification: In the definition of L_2, "or" does not mean "exclusive or" — *both* conditions might be satisfied. Thus L_2 includes all the strings that only include $\mathtt{c}$'s.

Application to Examples

First Attempts at Answers for "the big question:"

(a) $L_1 = \{\omega \in \Sigma^\star \mid \omega$ includes at least one "a"$\}$

Answer: We probably need to remember whether an "a" has already been seen. (The string should be ***accepted*** if this is eventually true.)

(b) $L_2 = \{\omega \in \Sigma^\star \mid$ either ω does not include an "a" or ω does not include a "b"$\}$

Answer: We probably need to remember whether both an "a" and "b" have been seen. (The string should ***rejected*** if this is eventually true.)

Identifying Subsets of $\Sigma^\star$ — and Corresponding States

Next, try to use your answer to identify a collection of nonempty subsets

$$S_0, S_1, S_2, \ldots$$

of $\Sigma^\star$. These will (eventually) correspond to the ***states*** of your DFA: They will correspond to the different cases, concerning the part of the string that has been seen so far, that might arise and should be remembered.

Note: It is probably a good idea to give more meaningful names to these states!

Application To Examples

(a) For $L_1 = \{\omega \in \Sigma^\star \mid \omega$ includes at least one "a"$\}$ we decided that we probably need to remember whether an "a" has already been seen. This suggests *two* subsets:

- $S_{\text{no}} = \{\omega \in \Sigma^\star \mid \omega$ does not include any a's$\}$
- $S_{\text{yes}} = \{\omega \in \Sigma^\star \mid \omega$ includes at least one "a"$\}$

(b) For $L_2 = \{\omega \in \Sigma^\star \mid$ either ω does not include an a or ω does not include a b$\}$ we decided that we probably need to remember both an "a" and a "b" have both been seen. This suggests *two* subsets:

- $\widehat{S}_{\text{no}} = \{\omega \in \Sigma^\star \mid \omega$ includes both an "a" and a "b"$\}$
- $\widehat{S}_{\text{yes}} = \{\omega \in \Sigma^\star \mid$ either ω does not include an "a" or ω does not include a "b"$\}$

A First "Sanity Check"

Before doing much more you should confirm that the following properties hold.

1. Only a *finite* number of subsets of $\Sigma^\star$ have been identified — so that these could be numbered

   $$S_0, S_1, \ldots, S_{n-1}$$

   for some positive integer n.

   Explanation: Each of these subsets will eventually correspond to a different ***state*** in the automaton being designed — and a DFA can only have a *finite* number of states.

   Note: This condition is satisfied for both examples.

A Second "Sanity Check"

2. Every string $\omega \in \Sigma^\star$ belongs to ***exactly one*** of the sets

$$S_0, S_1, \ldots, S_{n-1}$$

that have been identified.

Explanation: When our automaton sees (and processes) the string ω it should end up in ***exactly one*** state.

Note: This condition is also satisfied for both examples.

Identifying the Start State

If we have got this far then we can (provisionally) identify the states in our DFA:

$$Q = \{q_0, q_1, \ldots, q_{n-1}\}$$

where (for $0 \leq i \leq n-1$) q_i "corresponds" to the subset S_i of $\Sigma^\star$ that has been identified.

If the second "sanity check," above, was passed, then the empty string λ belongs to ***exactly one*** of the subsets $S_0, S_1, \ldots, S_{n-1}$.

The ***start state*** should now be set to be q_i, for the unique integer i such that $0 \leq i \leq n-1$ and $\lambda \in S_i$.

Simplification: Let us renumber states (and subsets) as needed, so that $\lambda \in S_0$ and q_0 is the start state.

How Subsets Correspond To States

The ***correspondence*** between subsets of $\Sigma^\star$ and states in Q can now be described more precisely.

Desired Property: For every string $\omega \in \Sigma^\star$ and for every integer i such that $0 \leq i \leq n-1$,

$$\omega \in S_i$$

if and only if

$$\delta^\star(q_0, \omega) = q_i$$

in the automaton $M = (Q, \Sigma, \delta, q_0, F)$ that is being designed.

Application To the First Example

In what follows, names of states will resemble names of corresponding sets but will sometimes be simplified to produce a better picture.

(a) The automaton for the language L_1 should include

- a state q_n corresponding to the set

$$S_{\text{no}} = \{\omega \in \Sigma^\star \mid \omega \text{ does not include any a's}\}$$

- a state q_y corresponding to the set

$$S_{\text{yes}} = \{\omega \in \Sigma^\star \mid \omega \text{ includes at least one "a"}\}$$

Since $\lambda \in S_{\text{no}}$, q_n should be the start state.

The First Example, So Far

Desired Properties:

- $\delta^\star(q_n, \omega) = q_n$ for every string $\omega \in S_{\text{no}}$, that is, for all ω such that ω does not include any a's;
- $\delta^\star(q_n, \omega) = q_y$ for every string $\omega \in S_{\text{yes}}$, that is, for all ω such that ω includes at least one "a".

Application To the Second Example

(b) The automaton for the language L_2 should include

- a state q_y corresponding to the set

$$\widehat{S}_{\text{yes}} = \{\omega \in \Sigma^\star \mid \text{either } \omega \text{ does not include an “a”} \\ \text{or } \omega \text{ does not include a “b”}\}$$

- a state q_n corresponding to the set

$$\widehat{S}_{\text{no}} = \{\omega \in \Sigma^\star \mid \omega \text{ includes both an “a” and a “b”}\}$$

Since $\lambda \in \widehat{S}_{\text{yes}}$, q_y should be the start state.

The Second Example, So Far

Desired Properties:

- $\delta^\star(q_y, \omega) = q_y$ for every string $\omega \in \widehat{S}_{\text{yes}}$, that is, for all ω such that either ω does not include an "a" or ω does not include a "b" (or both);
- $\delta^\star(q_y, \omega) = q_n$ for every string $\omega \in \widehat{S}_{\text{no}}$, that is, for all ω such that ω includes both an "a" and a "b".

A Third "Sanity Check"

The following property should also be satisfied by the subsets

$$S_0, S_1, \ldots, S_{n-1}$$

that have been identified, when designing an automaton for the language L.

3. Either $S_i \subseteq L$ or $S_i \cap L = \emptyset$ for each integer i such that $0 \leq i \leq n-1$.

   Explanation: Recall that all strings $\omega \in S_i$ should reach the same state q_i, when processed. These strings should therefore ***all*** belong to L if this state is an accepting state. On the other hand, ***none*** of them should belong to L if this state is *not* an accepting state.

Application to the First Example

The previous "explanation" gives a way to decide whether each state should be an accepting state — it should be one, if and only if the corresponding subset of $\Sigma^\star$ is also a subset of the language L for which an automaton is being designed.

(a) Recall that $S_{\text{no}} = \{\omega \in \Sigma^* \mid \omega \text{ does not include any a's}\}$, and that

$$S_{\text{yes}} = L_1 = \{\omega \in \Sigma^* \mid \omega \text{ includes at least one "a"}\}.$$

Since $S_{\text{no}} \cap L_1 = \emptyset$, the corresponding state q_n is *not* an accepting state. Since $S_{\text{yes}} \subseteq L_1$, the corresponding state q_y *is* an accepting state.

The First Example, So Far

Desired Properties:

- $\delta^\star(q_n, \omega) = q_n$ for every string $\omega \in S_{\text{no}}$, that is, for all ω such that ω does not include any a's;
- $\delta^\star(q_n, \omega) = q_y$ for every string $\omega \in S_{\text{yes}}$, that is, for all ω such that ω includes at least one "a".

Application to the Second Example

(b) Recall that

$$L_2 = \widehat{S}_{\text{yes}} = \{\omega \in \Sigma^\star \mid \text{either } \omega \text{ does not include an “a”} \\ \text{or } \omega \text{ does not include a “b”}\}$$

and that $\widehat{S}_{\text{no}} = \{\omega \in \Sigma^\star \mid \omega \text{ includes both an “a” and a “b”}\}$.

Since $\widehat{S}_{\text{yes}} \subseteq L_2$ the corresponding state, q_y, *is* an accepting state. Since $\widehat{S}_{\text{no}} \cap L_2 = \emptyset$ the corresponding state, q_n, is *not* an accepting state.

The Second Example, So Far

Desired Properties:

- $\delta^\star(q_y, \omega) = q_y$ for every string $\omega \in \widehat{S}_{\text{yes}}$, that is, for all ω such that either ω does not include an "a" or ω does not include a "b" (or both);
- $\delta^\star(q_y, \omega) = q_n$ for every string $\omega \in \widehat{S}_{\text{no}}$, that is, for all ω such that ω includes both an "a" and a "b".

A Fourth "Sanity Check"

One more property should be satisfied by the subsets

$$S_0, S_1, \ldots, S_{n-1}$$

that have been identified when designing an automaton for the language L.

4. For every integer i such that $0 \leq i \leq n-1$ and for every symbol $\sigma \in \Sigma$ there should exist an integer j such that $0 \leq j \leq n-1$ and

$$\{\omega \cdot \sigma \mid \omega \in S_i\} \subseteq S_j.$$

In other words, $\omega \cdot \sigma$ should all belong to the ***same*** subset (the same one of $S_0, S_1, \ldots, S_{n-1}$) for all $\omega \in S_i$ — for each integer i such that $0 \leq i \leq n-1$,and for each $\sigma \in \Sigma$.

A Fourth "Sanity Check"

Explanation: Recall that the "extended transition function" satisfies the following property: For every string $\omega \in \Sigma^\star$ and for every symbol $\sigma \in \Sigma$,

$$\delta^\star(q_0, \omega \cdot \sigma) = \delta(\delta^\star(q_0, \omega), \sigma).$$

Suppose, now, that $\omega \in S_i$ — so that $\delta^\star(q_0, \omega) = q_i$. Then it follows that

$$\delta^\star(q_0, \omega \cdot \sigma) = \delta(q_i, \sigma).$$

Thus $\delta^\star(q_0, \omega \cdot \sigma) = q_j$, for the integer j such that $0 \leq j \leq n-1$ and $\delta(q_i, \sigma) = q_j$.

This implies that $\omega \cdot \sigma \in S_j$. Since ω was arbitrarily chosen from S_i,

$$\{\omega \cdot \sigma \mid \omega \in S_i\} \subseteq S_j,$$

as claimed.

Calculating Transitions

Notice that this tells us how to define the transition function $\delta : Q \times \Sigma \to Q$: For every integer i such that $0 \leq i \leq n-1$ and for every symbol $\sigma \in \Sigma$,

$$\delta(q_i, \sigma) = q_j$$

for the (unique) integer j such that $0 \leq j \leq n-1$ and

$$\{\omega \cdot \sigma \mid \omega \in S_i\} \subseteq S_j.$$

Application to the First Example

Let us begin by considering the state q_n, which corresponds to the set

$$S_{\text{no}} = \{\omega \in \Sigma^\star \mid \omega \text{ does not include any a's}\}.$$

- If $\omega \in S_{\text{no}}$ then $\omega \cdot \mathtt{b} \in S_{\text{no}}$ and $\omega \cdot c \in S_{\text{no}}$ as well. Thus

$$\delta(q_n, \mathtt{b}) = \delta(q_n, \mathtt{c}) = q_n.$$

- On the other hand, if $\omega \in S_{\text{no}}$ then $\omega \cdot \mathtt{a}$ belongs to the set

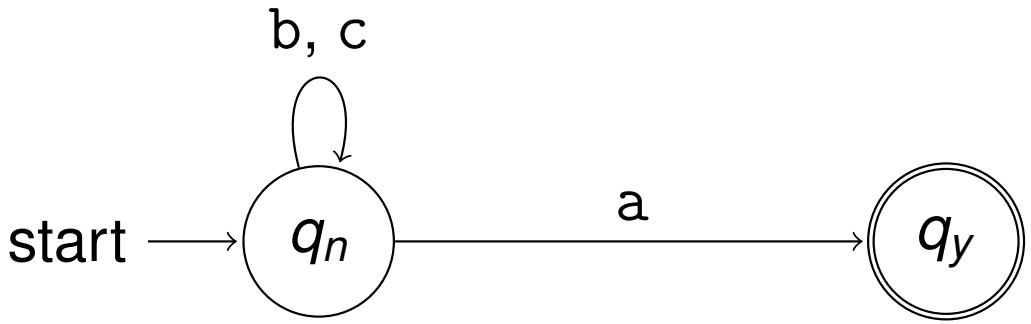
$$S_{\text{yes}} = \{\omega \in \Sigma^\star \mid \omega \text{ includes at least one “a”}\},$$

which corresponds to the state q_y. Thus

$$\delta(q_n, \mathtt{a}) = q_y.$$

The First Example, So Far

Adding the transitions that have been discovered, we now have the following.



Application to the First Example

We must also discover transitions out of the state q_y, which corresponds to the set

$$S_{\text{yes}} = \{\omega \in \Sigma^\star \mid \omega \text{ includes at least one “a”}\}.$$

- If ω includes at least one “a”, then so do $\omega \cdot \mathtt{a}$, $\omega \cdot \mathtt{b}$, and $\omega \cdot \mathtt{c}$. Thus

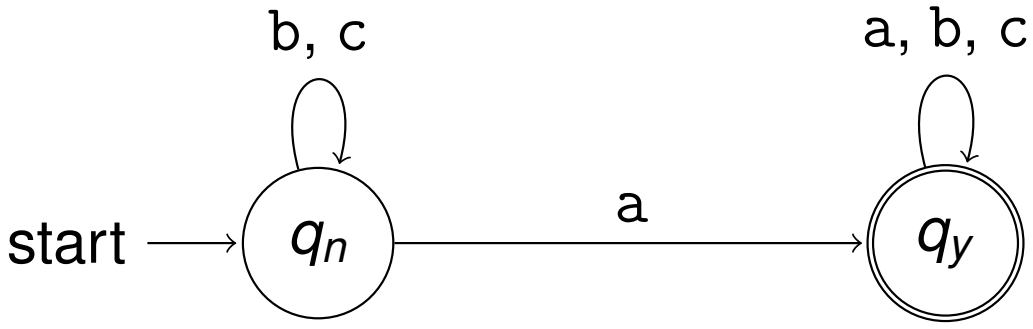
$$\delta(q_y, \mathtt{a}) = \delta(q_y, \mathtt{b}) = \delta(q_y, \mathtt{c}) = q_y.$$

Application to the First Example

Hurray!

The fourth “sanity check” has been passed for this example. Adding in the transitions that have been discovered, we obtain the following deterministic finite automaton for the language L_1.

The First Example, Concluded



Application to the Second Example

Let us begin by considering the state q_n, which corresponds to the set

$$\widehat{S}_{\text{no}} = \{\omega \in \Sigma^\star \mid \omega \text{ includes both an “a” and a “b”}\}.$$

- If $\omega \in \widehat{S}_{\text{no}}$, so that ω includes both an “a” and a “b”, then so do $\omega \cdot \mathtt{a}$, $\omega \cdot \mathtt{b}$, and $\omega \cdot \mathtt{c}$. Thus

$$\delta(q_n, \mathtt{a}) = \delta(q_n, \mathtt{b}) = \delta(q_n, \mathtt{c}) = q_n.$$

The Second Example, So Far

Adding the transitions that have been discovered, we now have the following.

Application to the Second Example

We must also discover transitions out of the state q_y, which corresponds to the set

$$\widehat{S}_{\text{yes}} = \{\omega \in \Sigma^\star \mid \text{either } \omega \text{ does not include an “a”} \\ \text{or } \omega \text{ does not include an “b”}\}$$

- First, Some Good News: Suppose that $\omega \in \widehat{S}_{\text{yes}}$, so that either ω does not include an “a” or ω does not include an “b”. Consider the string $\omega \cdot \mathsf{c}$.

  If ω does not include an “a” then $\omega \cdot \mathsf{c}$ does not include an “a” either, so that $\omega \cdot \mathsf{c} \in \widehat{S}_{\text{yes}}$.

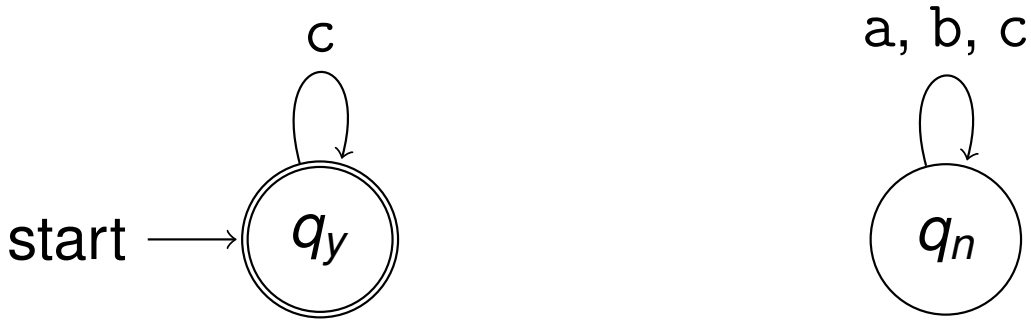
  If ω does not include a “b” then $\omega \cdot \mathsf{c}$ does not include a “b” either, so that $\omega \cdot \mathsf{c} \in \widehat{S}_{\text{yes}}$, once again.

  Since $\omega \cdot \mathsf{c} \in \widehat{S}_{\text{yes}}$ in every possible case, it follows that

  $$\delta(q_y, \mathsf{c}) = q_y.$$

The Second Example, So Far

Adding the transition that has been discovered, we now have the following.



Application to the Second Example

There are two more transitions to be discovered out of the state q_y, which corresponds to the set $\widehat{S}_{\text{yes}}$.

- Suppose that $\omega \in \widehat{S}_{\text{yes}}$, so that either ω does not include an "a" or ω does not include an "b". Consider the string $\omega \cdot \mathtt{a}$.

  If ω does not include an "b" then $\omega \cdot \mathtt{a}$ does not include an "b", either — so that $\omega \cdot \mathtt{a} \in \widehat{S}_{\text{yes}}$.

  Unfortunately, if ω does not include an "a" then $\omega \cdot \mathtt{a}$ *does* include one!

  - If ω did not include a "b" either, then $\omega \cdot \mathtt{a} \in \widehat{S}_{\text{yes}}$, because $\omega \cdot \mathtt{a}$ (still) does not include a "b".
  - However, if ω *does* include a "b" then $\omega \cdot \mathtt{a}$ includes both an "a" and a "b" — so that $\omega \cdot \mathtt{a} \in \widehat{S}_{\text{no}}$, instead.

Application to the Second Example

Something similar happens when one considers a string $\omega \cdot \mathtt{b}$, for $\omega \in \widehat{S}_{\text{yes}}$:

- If ω does not include an "a" then $\omega \cdot \mathtt{b} \in \widehat{S}_{\text{yes}}$.
- On the other hand, if ω does not include a "b" then
  - $\omega \cdot \mathtt{b} \in \widehat{S}_{\text{yes}}$ if ω does not include an "a", either, but
  - $\omega \cdot \mathtt{b} \in \widehat{S}_{\text{no}}$, instead, if ω *does* include an "a".

Question: What has happened here, and what does it mean?

Application to the Second Example

Answer:

The fourth “sanity check” has failed.

Application to the Second Example

In other words, it is ***not*** possible to identify well-defined transitions out of the state q_y for the symbols "a" or "b": The automaton must remember ***different*** information — or, possibly, ***additional*** information — in order to recognize the language L_2.

However, what we have done so far is useful — because it can help us to discover the "different information — or, possibly additional information" that is needed.

Application to the Second Example

Hypothesis: In order to recognize L_2 you must remember following information.

1. You must remember whether an "a" has already been seen.

   Explanation: This is needed to decide what to do if the string seen so far is in L_2 but an "a" is the next symbol that is seen.

2. You must *also* remember whether a "b" has already been seen.

   Explanation: This is needed to decide what to do if the string seen so far is in L_2 but a "b" is the next symbol that is seen.

Application to the Second Example

Taken in combination this leads to $4 = 2 \times 2$ possibilities — with corresponding subsets of $\Sigma^\star$.

1. The string seen so far does not include any a's *or* any b's, so that it belongs to the set

$$S_\emptyset = \{\omega \in \Sigma^\star \mid \omega \text{ only includes c's}\}.$$

2. At least one "a" has been seen but no b's have, so that the string seen so far belongs to the set

$$S_a = \{\omega \in \Sigma^\star \mid \omega \text{ includes at least one "a" but no b's}\}.$$

Application to the Second Example

3. At least one "b" has been seen but no a's have, so that the string seen so far belongs to the set

$$S_b = \{\omega \in \Sigma^\star \mid \omega \text{ includes at least one "b" but no a's}\}.$$

4. The string seen so far includes both an "a" and a "b", so that it belongs to the set

$$\widehat{S}_{\text{no}} = \{\omega \in \Sigma^\star \mid \omega \text{ includes at least one "a" and at least one "b"}\}.$$

Application to the Second Example

- ***Note:*** The final set, $\widehat{S}_{\text{no}}$, is the same as the set that was called $\widehat{S}_{\text{no}}$ before this.
- On the other hand, $S_{\emptyset}$, S_a and S_b are all ***subsets*** of the set $\widehat{S}_{\text{yes}}$ that we were working with before.
- So, we are remembering ***more*** information (instead of *different* information) — and we have ***refined*** the collection of sets (and the collection of states of an automaton) being used.
- We are now ready to “roll back” to (near) the beginning of our design process, using this new collection of sets and states.

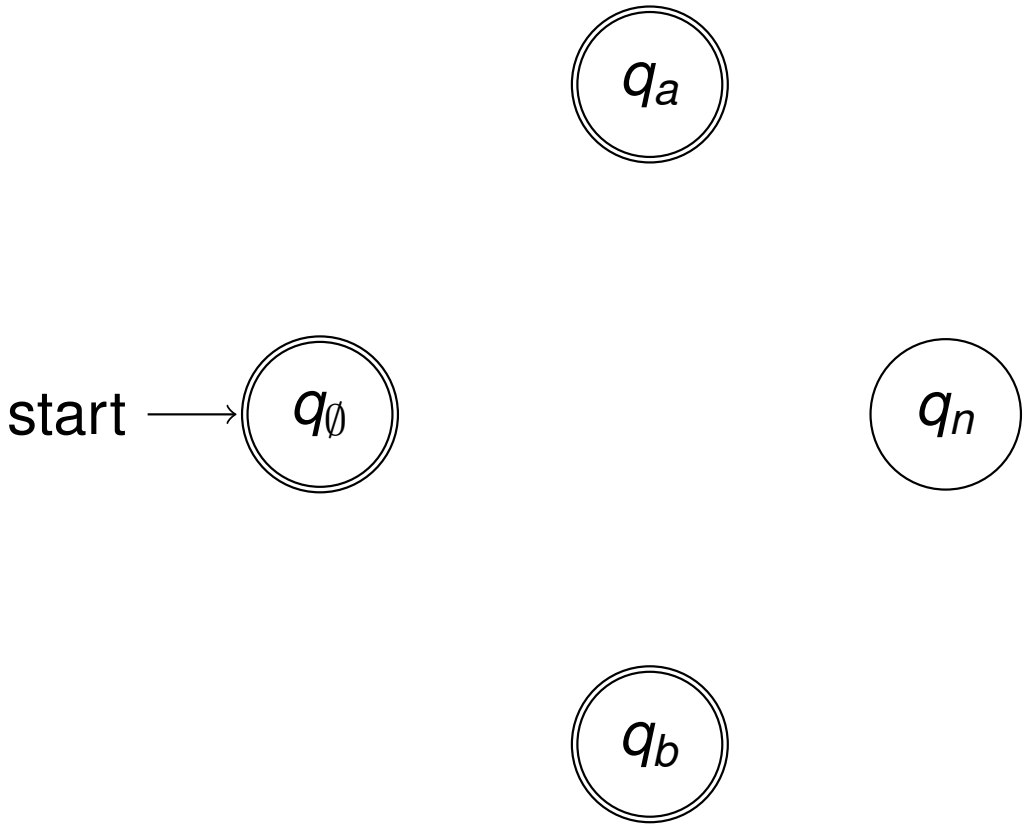
Resuming the Process

- The first "sanity check" is passed, once again: Only a finite number of subsets of $\Sigma^\star$ (and corresponding states) have been identified.
- The second "sanity check" is passed as well: It follows from the descriptions of $S_\emptyset$, S_a, S_b and $\widehat{S}_{\text{no}}$ that every string in $\Sigma^\star$ belongs to *exactly one* of these sets.

  Let $q_\emptyset$, q_a, q_b and q_n be states corresponding to the subsets $S_\emptyset$, S_a, S_b and $\widehat{S}_{\text{no}}$, respectively. Then, since $\lambda \in S_\emptyset$, $q_\emptyset$ is the start state for the automaton being designed.
- The third "sanity check" is also passed because $S_\emptyset \subseteq L_2$, $S_a \subseteq L_2$, $S_b \subseteq L_2$, and $\widehat{S}_{\text{no}} \cap L_2 = \emptyset$. It follows from this that $q_\emptyset$, q_a and q_b are all accepting states, and q_n is not.

The Second Example, So Far

Here is what we have, so far.



The Second Example: Discovering Transactions

Consider transactions out of the state $q_\emptyset$, which corresponds to the set

$$S_\emptyset = \{\omega \in \Sigma^\star \mid \omega \text{ only include c's}\}.$$

- If $\omega \in S_\emptyset$ then $\omega \cdot \mathtt{a}$ includes at least one "a", but not a "b", so that $\omega \cdot \mathtt{a} \in S_a$. Thus

$$\delta(q_\emptyset, \mathtt{a}) = q_a.$$

- If $\omega \in S_\emptyset$ then $\omega \cdot \mathtt{b}$ includes at least one "b", but not an "a", so that $\omega \cdot \mathtt{b} \in S_b$. Thus

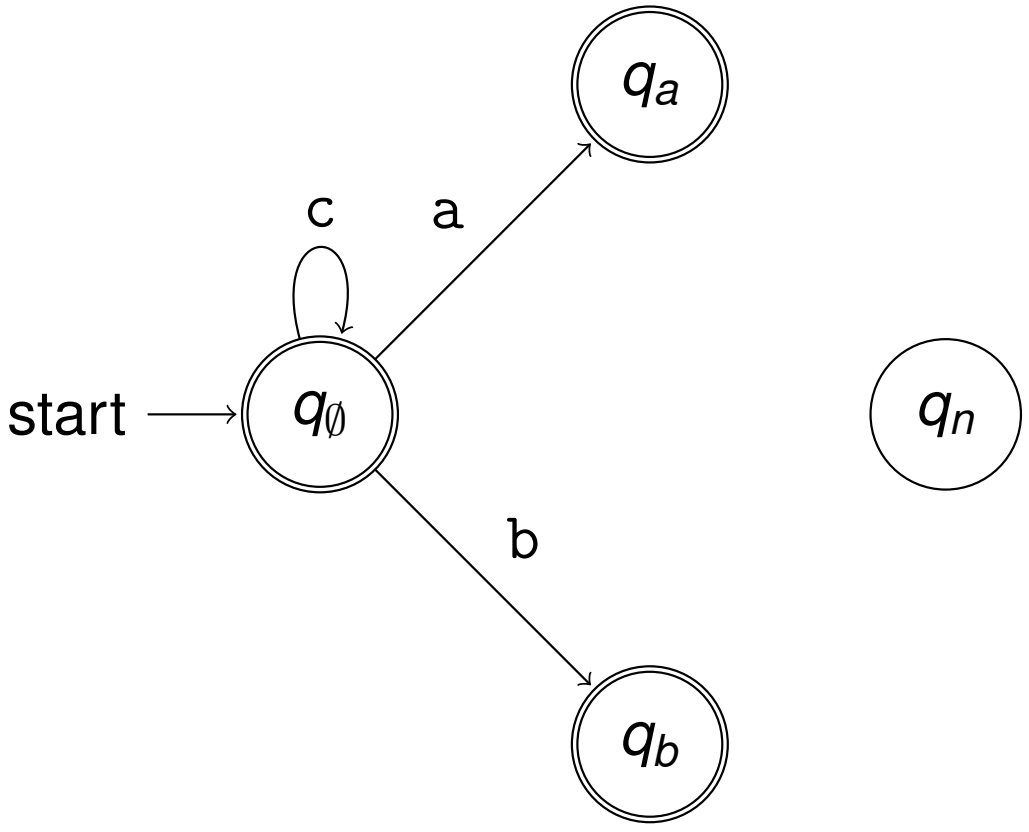
$$\delta(q_\emptyset, \mathtt{b}) = q_b.$$

- If $\omega \in S_\emptyset$ then $\omega \cdot \mathtt{c}$ does not include an "a" or a "b", so that $\omega \cdot \mathtt{c} \in S_\emptyset$. Thus

$$\delta(q_\emptyset, \mathtt{c}) = q_\emptyset.$$

The Second Example, So Far

Here is what we have, so far.



The Second Example: Discovering Transactions

Consider transactions out of the state q_a, which corresponds to the set

$$S_a = \{\omega \in \Sigma^\star \mid \omega \text{ contains at least one “a” but no b's}\}.$$

- If $\omega \in S_a$ then $\omega \cdot \mathtt{a}$ includes at least one “a”, but not a “b”, so that $\omega \cdot \mathtt{a} \in S_a$. Thus

$$\delta(q_a, \mathtt{a}) = q_a.$$

- If $\omega \in S_a$ then $\omega \cdot \mathtt{b}$ includes both an “a” and a “b”, so that $\omega \cdot \mathtt{b} \in \widehat{S}_{\text{no}}$. Thus

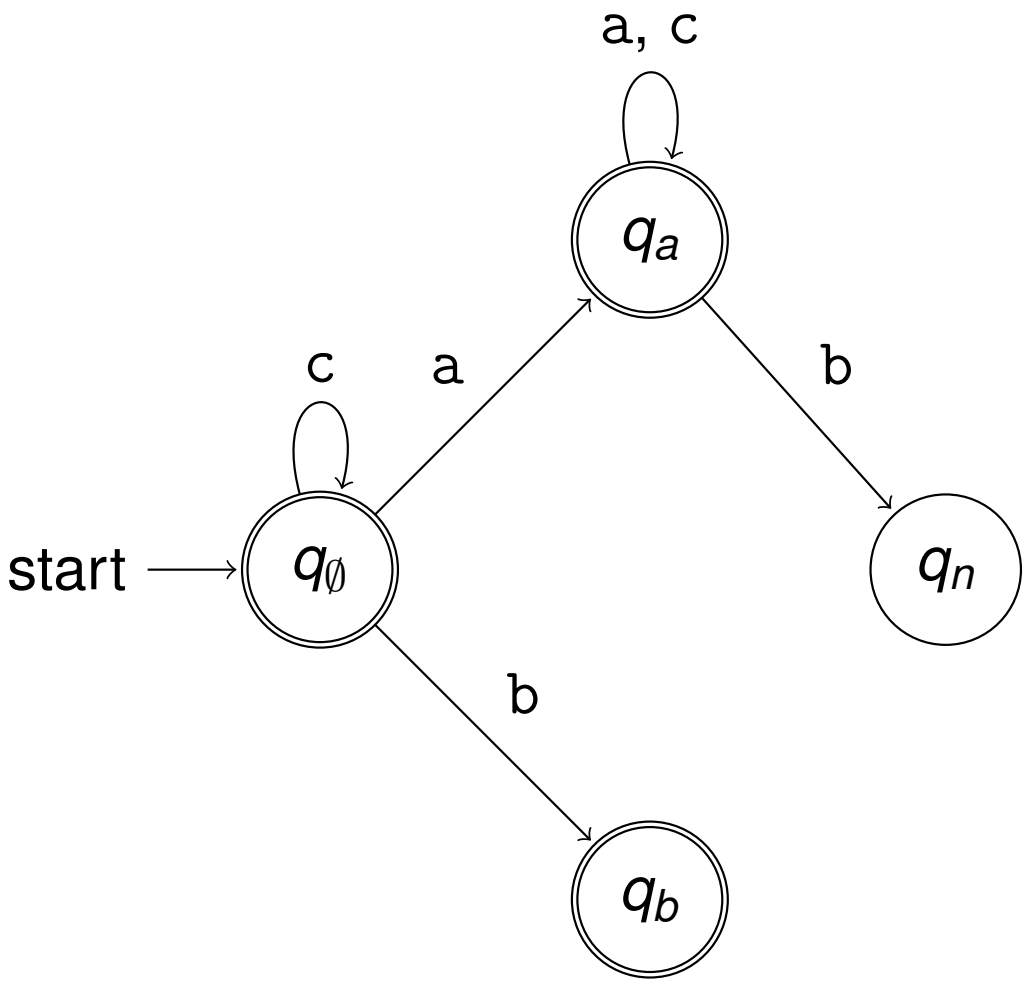
$$\delta(q_a, \mathtt{b}) = q_n.$$

- If $\omega \in S_a$ then $\omega \cdot \mathtt{c}$ includes at least one “a”, but not a “b”, so that $\omega \cdot \mathtt{c} \in S_a$. Thus

$$\delta(q_a, \mathtt{c}) = q_a.$$

The Second Example, So Far

Here is what we have, so far.



The Second Example: Discovering Transactions

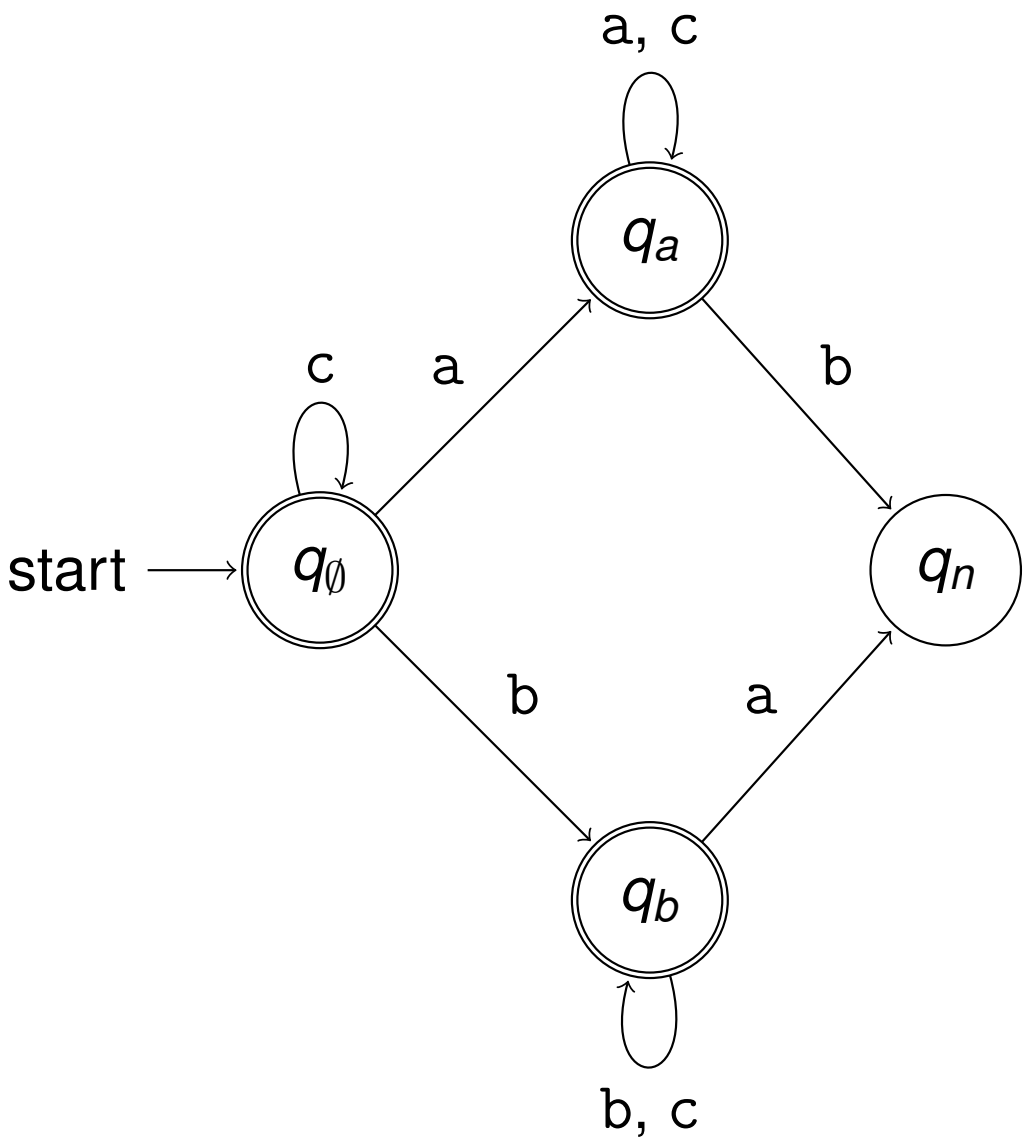
Consider transactions out of the state q_b, which corresponds to the set

$$S_b = \{\omega \in \Sigma^\star \mid \omega \text{ contains at least one “b” but no a's}\}.$$

- If $\omega \in S_b$ then $\omega \cdot \mathtt{a}$ includes both an “a” and a “b”, so that $\omega \cdot \mathtt{a} \in \widehat{S}_{\text{no}}$. Thus
$$\delta(q_b, \mathtt{a}) = q_n.$$
- If $\omega \in S_b$ then $\omega \cdot \mathtt{b}$ includes at least one “b”, but not an “a”, so that $\omega \cdot \mathtt{b} \in S_b$. Thus
$$\delta(q_b, \mathtt{b}) = q_b.$$
- If $\omega \in S_b$ then $\omega \cdot \mathtt{c}$ includes at least one “b”, but not an “a”, so that $\omega \cdot \mathtt{c} \in S_b$. Thus
$$\delta(q_b, \mathtt{c}) = q_b.$$

The Second Example, So Far

Here is what we have, so far.



The Second Example: Discovering Transactions

Consider transactions out of the state q_n, which corresponds to the set

$$\widehat{S}_{\text{no}} = \{\omega \in \Sigma^\star \mid \omega \text{ includes both an “a” and a “b”}\}.$$

- If $\omega \in \widehat{S}_{\text{no}}$ then ω includes both an “a” and a “b”, so that $\omega \cdot \mathtt{a}$, $\omega \cdot \mathtt{b}$, and $\omega \cdot \mathtt{c}$ each contain both an “a” and a “b” as well. Thus $\omega \cdot \mathtt{a}, \omega \cdot \mathtt{b}, \omega \cdot \mathtt{c} \in \widehat{S}_{\text{no}}$, implying that

$$\delta(q_n, \mathtt{a}) = \delta(q_n, \mathtt{b}) = \delta(q_n, \mathtt{c}) = q_n.$$

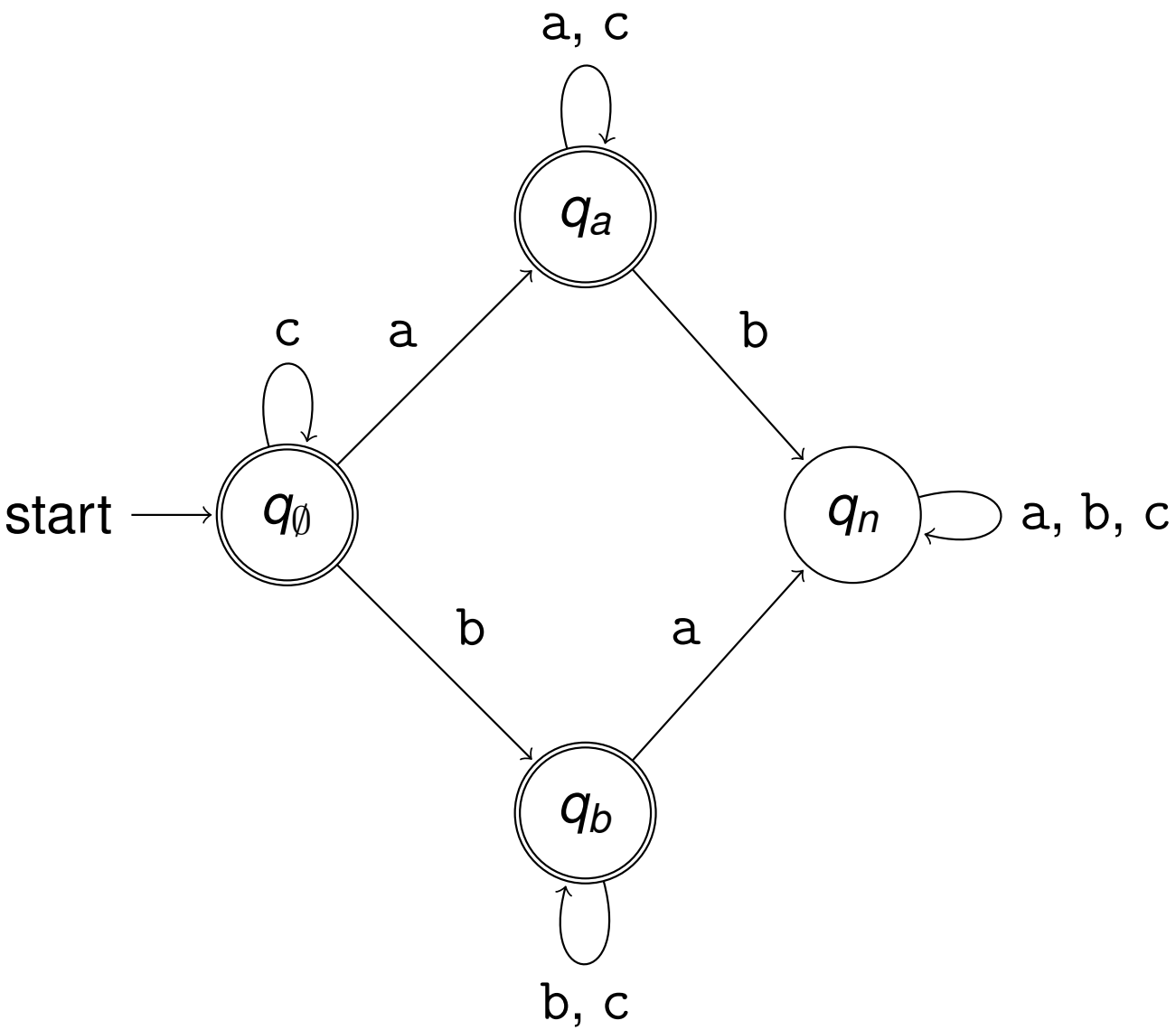
The Second Example: Discovering Transitions

Hurray!

The fourth “sanity check” has now been passed as well. A DFA for the language L_2 — which includes the transitions that have been identified — is as follows.

The Second Example, Concluded

Here is what we now have.



Suggestions

Suggestions about Figuring out "What to Remember"

- Consider the language to be recognized very carefully. This may be enough to figure out what to remember, or it may provide a good start.
- Learn from unsuccessful attempts instead of discarding them. This helped us to design a DFA for L_2!
- Study Examples — and Practice! As you consider more DFA's for languages you will begin to recognize common situations and patterns.

  Completion of the suggested reading from *Introduction to the Theory of Computation,* given at the end of the previous lecture, preparing for and participating in the lecture presentation for *this* lecture, and completion of the next tutorial exercise, will all be helpful.

How Not To Do It

It can be tempting to ignore processes like the one described in today's class. Instead, a student might

- Study and memorize several example DFA's for various languages.
- When asked to design a DFA for another language, L, pick the example whose language seems to be closest to L and try to change the DFA in the example so that L is its language.

Sometimes this works, but

this is often not reliable!

It really *is* better to study, understand, and learn to apply the design process that has been introduced in this lecture.

What's Next?

Coming Up:

- The next lecture introduces a process that can be used to ***prove*** that a given deterministic finite automaton has a given language.