Computer Science 351

DFA Design and Verification — Part Two

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #4

Goals for Today

Goals for Today:

- *Conclusion* of a presentation of a process that can be used to ***design*** a deterministic finite automaton that has a given language.
- Description of a process to ***prove*** that a given deterministic finite automaton has a given language

Ongoing Example

The process being developed will be used to design a deterministic finite automata for the following language over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$:

$$L = \{\omega \in \Sigma^\star \mid \text{either } \omega \text{ does not include an "}\mathtt{a}\text{"} \\ \text{or } \omega \text{ does not include a "}\mathtt{b}\text{"}\}.$$

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

Following a Design Process — So Far

We started by considering ***what a DFA must remember about the string that has been processed, so far***.

- *Our Answer, So Far:* We *guessed* that the DFA must remember whether both an “`la`” and a “`lb`” have already been seen: The string *is* in the desired the language if this is not true yet — but it it is *not* in the language if this *is* true.

Following a Design Process — So Far

This was used to participation the set, $\Sigma^\star$, of all strings over Σ, in to a pair of subsets, namely

$$S_1 = S_{\text{yes}} = \{\omega \in \Sigma^\star \mid \text{either } \omega \text{ does not include an “a”} \\ \text{or } \omega \text{ does not include a “b”}\}$$

and

$$S_0 = S_{\text{no}} = \{\omega \in \Sigma^\star \mid \omega \text{ includes both an “a” and a “b”}\}.$$

Following a Design Process — So Far

Several ***necessary conditions*** — or “sanity checks” — were considered.

1. We checked that only ***finitely many*** subsets of $\Sigma^\star$ had been identified.

   This test passed — because two subsets (S_0 and S_1) were identified. Let n be the number of subsets identified, so that $n = 2$.

2. We checked that ***every string belongs to exactly one of these subsets***.

   This test passed too — because S_0 is the desired language, L, and S_1 is the set of strings that *do not* belong to L… so these sets correspond to the two possible answers for a “Yes/No” question.

Following a Design Process — So Far

- The empty string, λ, belongs to S_0. We would generally *renumber* the states, at this point, if necessary — because we generally want this condition, "$\lambda \in S_0$" to be true.
- We can now associate a ***state*** with each of our subsets:
  - State q_{yes} corresponds to $S_0 = S_{\text{yes}}$. Since $\lambda \in S_0$ q_{yes} should be the start state for the DFA we wish to design.
  - State q_{no} corresponds to $S_1 = S_{\text{no}}$.

Following a Design Process — So Far

- The desired relationship between subsets and states can now be given more precisely:

  For each state q and subset S of $\Sigma^\star$ corresponding to q, we want it to be the case that

  $$\delta^\star(q_0, \omega) = q \quad \textit{if and only if} \quad \omega \in S$$

  for every string $\omega \in \Sigma^\star$ — where q_0 is the start state (so that $q_0 = q_{yes}$, for this example).

- In particular, we want the following conditions to be satisfied, for every string $\omega \in \Sigma^\star$:
  - $\delta^\star(q_{\mathrm{yes}}, \omega) = q_{\mathrm{yes}}$ if and only if $\omega \in S_{\mathrm{yes}}$, and
  - $\delta^\star(q_{\mathrm{yes}}, \omega) = q_{\mathrm{no}}$ if and only if $\omega \in S_{\mathrm{no}}$.

Following a Design Process — So Far

Another necessary condition — or "sanity check" was checked after that. Suppose subsets

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

of $\Sigma^\star$ have been identified — so that $n = 2$, $S_0 = S_{\text{yes}}$ and $S_1 = S_{\text{no}}$ in this example. Suppose state q_i corresponds to subset S_i for $0 \le i \le n-1$ — so that $q_0 = q_{\text{yes}}$ and $q_1 = q_{\text{no}}$ in this example.

3. For every integer i such that $0 \le i \le n-1$ either $S_i \subseteq L$ — and q_i is an accepting state — or $S_i \cap L = \emptyset$ — and q_i is *not* an accepting state.

   This test was also passed, because $S_{\text{yes}} = L$ (so q_{yes} is an accepting state) and $S_{\text{no}} \cap L = \emptyset$ (and q_{no} is not an accepting state).

Following a Design Process — So Far

A final necessary condition — or "sanity check" was also checked:

4. ***Transitions must be well defined:*** For every integer i such that $0 \leq i \leq n-1$ and symbol $\sigma \in \Sigma$ there must exist an integer j such that $0 \leq j \leq n-1$ and $\delta(q_i, \sigma) = q_j$. Then it must also be true that

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

Good News: Since $\omega \cdot \sigma \in S_{\text{no}}$, whenever $\omega \in S_{\text{no}}$ — so that

$$\{\omega \cdot \sigma \mid \omega \in S_{\text{no}}\} \subseteq S_{\text{no}}$$

for all $\sigma \in \Sigma$, *the transitions out of S_{no} were well-defined*.

Following a Design Process — So Far

Not-So-Good News: While

$$\{\omega \cdot \mathrm{c} \mid \omega \in S_{\mathrm{yes}}\} \subseteq S_{\mathrm{yes}},$$

so that the transition for q_{yes} and "c" is well-defined (and $\delta(q_{\mathrm{yes}}, \mathrm{c}) = q_{\mathrm{yes}}$, it was also discovered that

$$\{\omega \cdot \mathrm{a} \mid \omega \in S_{\mathrm{yes}}\} \not\subseteq S_{\mathrm{yes}} \quad \text{and} \quad \{\omega \cdot \mathrm{a} \mid \omega \in S_{\mathrm{yes}}\} \not\subseteq S_{\mathrm{no}}$$

— so that the transition for q_{yes} and "a" is not well-defined — and that

$$\{\omega \cdot \mathrm{b} \mid \omega \in S_{\mathrm{yes}}\} \not\subseteq S_{\mathrm{yes}} \quad \text{and} \quad \{\omega \cdot \mathrm{b} \mid \omega \in S_{\mathrm{yes}}\} \not\subseteq S_{\mathrm{no}}$$

— so that the transition for q_{yes} and "b" is not well-defined, either.

Following a Design Process —- So Far

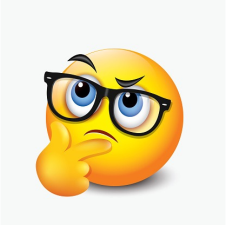
What We Concluded:

The fourth “sanity check” has failed.

Following a Design Process — So Far

- ***Further Conclusion:*** The automaton must remember ***different*** information — or, possibly, ***additional*** information — in order to recognize the language L.

All is Not Lost!



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 Example — Starting a Second Attempt

Hypothesis: In order to recognize L you must remember the 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, 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 but a "b" is the next symbol that is seen.

Wait a Minute! What Just Happened Here?

- Notice that we are ***rolling back***, and ***starting this design process all over again***.
- In particular: The above information is a new answer for the “question” that was considered at the *beginning* of this design process.
- However, ***we have learned something from the first attempt*** — and we are making use of new information that has been discovered — so that answers for questions will be different, and there is a chance that the process will complete, successfully, this time!
- This is one form of a process that is called ***refinement***.

Proceeding with the "Rolled Back" Process

We continued by describing a finite collection of *subsets of* $\Sigma^\star$ — such that the information we are remembering, about a string $\omega \in \Sigma^\star$, is the same as remembering which, of these subsets, ω belongs to.

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

Application to the Example — A Second Attempt

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

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

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

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

Application to the Example — A Second Attempt

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

$$S_{\text{no}} = \{\omega \in \Sigma^\star \mid \omega \text{ includes at least one “a” and at least one “b”}\}.$$

Application to the Example — Comparing the Attempts

- ***Note:*** The final set, S_{no}, is the same as the set that was called S_{no} before this.
- On the other hand, $S_{\emptyset}$, S_a and S_b are all ***subsets*** of the set S_{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.

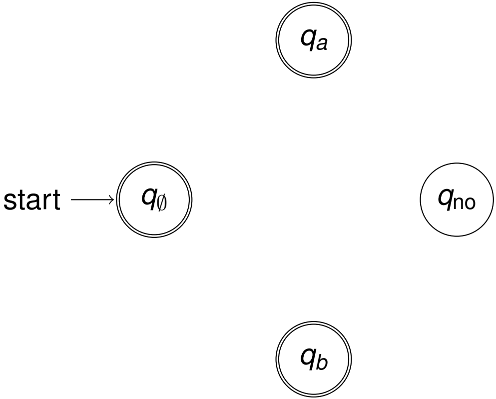
Proceeding with the "Rolled Back" Process

- 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 S_{no} that every string in $\Sigma^\star$ belongs to *exactly one* of these sets.

  Let $q_\emptyset$, q_a, q_b and q_{no} be states corresponding to the subsets $S_\emptyset$, S_a, S_b and S_{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 $S_\emptyset \subseteq L$, $S_a \subseteq L$, $S_b \subseteq L$, and $S_{\text{no}} \cap L = \emptyset$. It follows from this that $q_\emptyset$, q_a and q_b are all accepting states, and q_{no} is not.

The Example, So Far

Here is what we have, so far.



Note: We are now back at the step where the first attempt failed.

Continuing the 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 $\omega \cdot \mathrm{a}$ includes at least one "a", but not a "b", so that $\omega \cdot \mathrm{a} \in S_a$. Thus $\{\omega \cdot \mathrm{a} \mid \omega \in S_\emptyset\} \subseteq S_a$, and

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

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

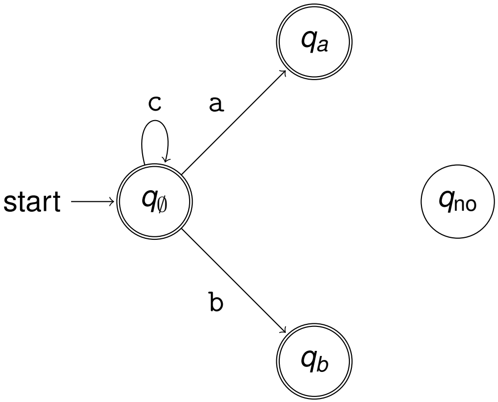
$$\delta(q_\emptyset, \mathrm{b}) = q_b.$$

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

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

The Example, So Far

Here is what we have, so far.



Continuing the Example: Discovering Transactions

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

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

- 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 $\{\omega \cdot \mathtt{a} \mid \omega \in S_a\} \subseteq S_a$, and

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

- If $\omega \in S_a$ then $\omega \cdot \mathtt{b}$ includes both an "a" and a "b", so that $\omega \cdot \mathtt{b} \in S_{\text{no}}$. Thus $\{\omega \cdot \mathtt{b} \mid \omega \in S_a\} \subseteq S_{\text{no}}$, and

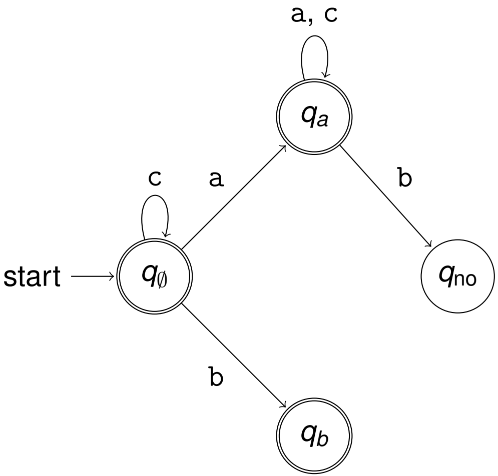
$$\delta(q_a, \mathtt{b}) = q_{\text{no}}.$$

- 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 $\{\omega \cdot \mathtt{c} \mid \omega \in S_a\} \subseteq S_a$, and

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

The Example, So Far

Here is what we have, so far.



Continuing the Example: Discovering Transactions

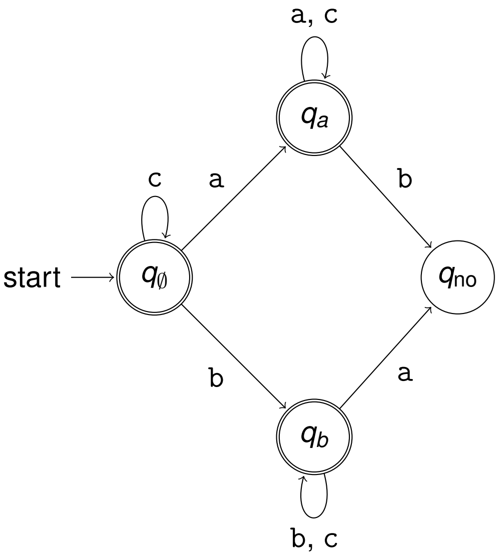
Consider transactions out of the state q_b, which corresponds to the set

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

- If $\omega \in S_b$ then $\omega \cdot \mathtt{a}$ includes both an “a” and a “b”, so that $\omega \cdot \mathtt{a} \in S_{\text{no}}$. Thus $\{\omega \cdot \mathtt{b} \mid \omega \in S_a\} \subseteq S_{\text{no}}$, and
$$\delta(q_b, \mathtt{a}) = q_{\text{no}}.$$
- 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 $\{\omega \cdot \mathtt{b} \mid \omega \in S_b\} \subseteq S_b$, and
$$\delta(q_b, \mathtt{b}) = q_b.$$
- 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 $\{\omega \cdot \mathtt{c} \mid \omega \in S_b\} \subseteq S_b$, and
$$\delta(q_b, \mathtt{c}) = q_b.$$

The Example, So Far

Here is what we have, so far.



Continuing the Example: Discovering Transactions

Consider transactions out of the state q_{no}, which corresponds to the set

$$S_{\mathrm{no}} = \{\omega \in \Sigma^\star \mid \omega \text{ includes both an "}\mathtt{a}\text{" and a "}\mathtt{b}\text{"}\}.$$

- If $\omega \in S_{\mathrm{no}}$ then ω includes both an "$\mathtt{a}$" and a "$\mathtt{b}$", so that $\omega \cdot \mathtt{a}$, $\omega \cdot \mathtt{b}$, and $\omega \cdot \mathtt{c}$ each contain both an "$\mathtt{a}$" and a "$\mathtt{b}$" as well. Thus $\omega \cdot \mathtt{a}, \omega \cdot \mathtt{b}, \omega \cdot \mathtt{c} \in S_{\mathrm{no}}$, implying that

  $$\{\omega \cdot \sigma \mid \omega \in S_{\mathrm{no}}\} \subseteq S_{\mathrm{no}}$$

  for $\sigma = \mathtt{a}$, for $\sigma = \mathtt{b}$, and for $\sigma = \mathtt{c}$, so that

  $$\delta(q_{\mathrm{no}}, \mathtt{a}) = \delta(q_{\mathrm{no}}, \mathtt{b}) = \delta(q_{\mathrm{no}}, \mathtt{c}) = q_{\mathrm{no}}.$$

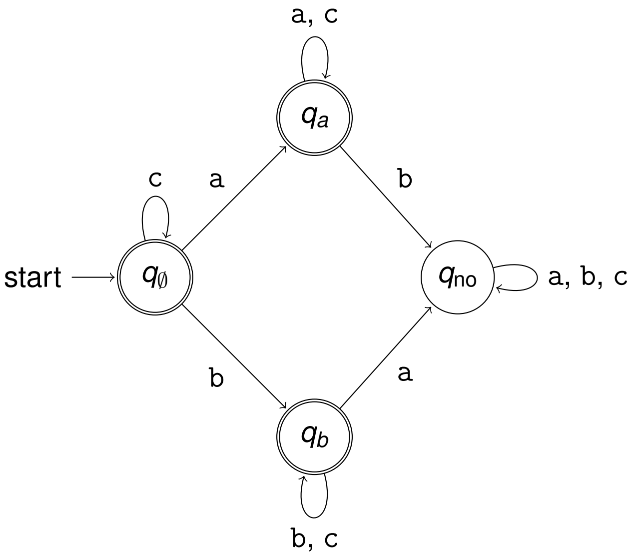
Continuing the Example: Discovering Transitions

Hurray!

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

The 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!
- Study Examples — and ***Practice!*** As you consider more DFA's for languages you will begin to recognize common situations and patterns.

Verification

We're not done yet!

- Why should you — or anyone else — believe that ***this process is correct***, that is, that this process is guaranteed to provide what is claimed (a DFA for the given language) whenever you completed the process, without making mistakes?
- How could you *convince someone else* that a given DFA has a given language?

A Useful Result

Theorem (Correctness of a DFA)

Let $L \subseteq \Sigma^\star$, for an alphabet Σ, and let

$$M = (Q, \Sigma, \delta, q_0, F)$$

with the same alphabet Σ. Suppose that (after renaming states, if needed)

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

where $n = |Q| \geq 1$. Suppose, as well, that

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

are subsets of $\Sigma^\star$ such that the following properties are satisfied.

A Useful Result

(a) *Every string in* $\Sigma^\star$ *belongs to **exactly one** of*

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

(b) $\lambda \in S_0$.

(c) $S_i \subseteq L$ *for every integer* i *such that* $0 \leq i \leq n-1$ *and* $q_i \in F$.

(d) $S_i \cap L = \emptyset$ *for every integer* i *such that* $0 \leq i \leq n-1$ *and* $q_i \notin F$.

(e) *The following property is satisfied, for every integer* i *such that* $0 \leq i \leq n-1$ *and for every symbol* $\sigma \in \Sigma$:

"Suppose that $q_j = \delta(q_i, \sigma)$ *(so that* $0 \leq j \leq n-1$*). Then*

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

Then $L(M) = L$.

Using This Result

- A ***proof*** of this result is given in a supplement for this lecture — and this can be used if you — or someone else — is not convinced that *the claim* is correct, either.
- You do not need to know how to prove that this result is correct — but you can ***use*** it to prove that a given deterministic finite automaton has a given language.
- ***Note:*** If you used the "design process" that was presented in these notes, to design a DFA for the language L — and M is the DFA you obtained — then (if you carried out the design process completely) you have already discovered proofs of everything you need. That is: ***proving correctness of your DFA*** only involves re-organizing the information you have already discovered.

Using This Result

- Proving correctness of your DFA, when you *did not* use a design process like this one to develop it, might be considerably more difficult. Indeed, it might not be clear that your DFA is correct, at all!
- A completion of this example — that is, and application of the above theorem to prove that the above deterministic finite automaton has the language L, is included in the supplement to this lecture.