Computer Science 351

DFA Design and Verification — Part One

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #3

Goals for Today

Goals for Today:

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

DFA Design — Objectives

Given at the Beginning:

- An ***alphabet*** Σ.
- The ***language*** $L \subseteq \Sigma^\star$ of the DFA that is to be designed.

What We Want to Have a the End:

- A deterministic finite automaton

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

  whose alphabet is the alphabet, Σ, that we were given.
- A ***proof*** that $L = L(M)$ — or enough information so that it is clear that this proof could be written.

DFA Design — Advice

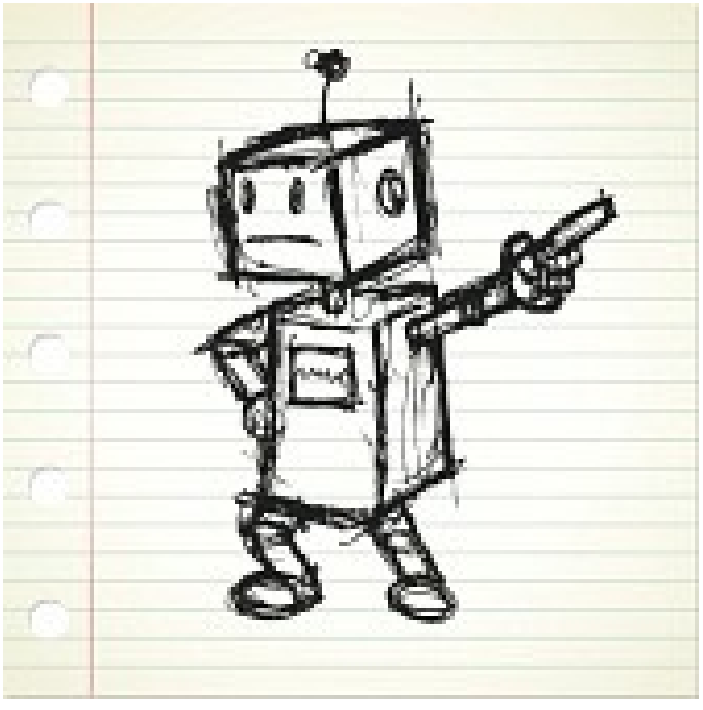
Good Advice from "Introduction to the Theory of Computation:"

> "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.***"

DFA Design - Advice

That is:

Be the Machine!



Ongoing Example

Goal: 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 — including the empty string.

Key Points about DFAs

Key Points about DFAs:

- One symbol of input string is processed at a time.
- Current state is the only information about what has been processed so far.
- A DFA only has a finite number of states.

Conclusion:

- Only a finite amount of information is remembered about the part of the string processed so far — no matter how long that part of the string might be.

First Question To Try To Answer

First Question To Try To Answer:

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

Ongoing Example

First Guess for Answer: We need to remember whether the string, so far, belongs to L.

- The string, seen so far, might be the whole string — so it is necessary to remember at least this much information.
- By definition of L, this would mean that that we must remember whether the string, seen so far, either does not include an "a" or does not include a "b" (or both).

Stating This More Formally

The answer for the first question must be precise enough so that it defines a *collection*

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

of subsets of $\Sigma^\star$, for some positive integer n — such that the information you are remembering, about a string $\omega \in \Sigma^\star$, is the same as remembering which, of these subsets, ω belongs to.

- These sets can be given more descriptive names, instead.

Ongoing Example

The answer for the example corresponds to the integer $n = 2$ and a pair of sets

$$S_0 = L = \{\omega \in \Sigma^\star \mid \text{either } \omega \text{ does not include an “a” or } \omega \text{ does not include a “b”}\}$$

and

$$S_1 = \Sigma^\star \setminus L = \{\omega \in \Sigma^\star \mid \omega \text{ includes both an “a” and a “b”}\}.$$

- One might also call S_0 “S_{yes}” and call S_1 “S_{no}” (reflecting whether the string, seen so far, should be accepted by the DFA).

Continuing the Process: A First "Sanity Check"

First condition that must be satisfied:

1. Only a *finite* number of subsets of $\Sigma^\star$ have been identified — so that these really can be listed as $S_0, S_1, S_2, \ldots, S_{n-1}$ for some positive integer n.

About This Condition:

- It is necessary because each of the sets, that have been identified, will correspond to one of the states in the DFA being designed.
- This condition is satisfied, for the example problem, with $n = 2$.

Continuing the Process: A Second "Sanity Check"

Second condition that must be satisfied:

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.

About This is Condition:

- The DFA will include states $q_0, q_1, \ldots, q_n$ such that, for every integer i such that $0 \leq i \leq n-1$ and for every string $\omega \in \Sigma^\star$, q_i is reached after processing ω *if and only if* $\omega \in S_i$. More formally: For every integer i such that $0 \leq i \leq n-1$ and for all $\omega \in \Sigma^\star$,

$$\delta^\star(q_0, \omega) = q_i \quad \text{if and only if} \quad \omega \in S_i.$$

The condition ensures that that the state reached, by processing a string $\omega \in \Sigma^\star$ is always well-defined.

Avoid This Mistake!

- Sometimes, students are trying to do this *after* a DFA, based on this information has been designed.
- It can be tempting to think about “what the DFA does” and try to use this to prove that this condition holds.
- ***It is a mistake to do that*** because correctness of the DFA has not been proved yet!
- The same kind of mistake can be made — and should be watched for and avoided — in later steps too.

Continuing the Process: Identifying the Start State

- If the second condition is satisfied then the empty string, λ, belongs to exactly one of the sets $S_0, S_1, \ldots, S_{n-1}$.
- *Reordering and renumbering sets, if needed*, we may now require that $\lambda \in S_0$.
- Then the the corresponding state, q_0, will be the ***start state*** of the DFA being designed — because this is the state reached after processing the empty string.

Ongoing Example

Ongoing Example:

- The condition is satisfied, for the example, because $S_1 = \Sigma^\star \setminus S_0$: For all $\omega \in \Sigma^\star$, $\omega \in S_1$ if and only if $\omega \notin S_0$.
- It follows from that that every string $\omega \in \Sigma^\star$ belongs to *exactly one* of S_0 or S_1, as required.
- Since the empty string does not include an "a" (or a "b"), $\lambda \in S_0 = L$ — and the state, q_0, that corresponds to S_0 can be chosen to be the start state.

The Example, So Far

Desired Properties:

- $\delta^\star(q_0, \omega) = q_0$ for every string $\omega \in S_0$, that is, for all ω such that either ω does not include an "a" or ω does not include a "b" (or both);
- $\delta^\star(q_0, \omega) = q_1$ for every string $\omega \in S_1$, that is, for all ω such that ω includes both an "a" and a "b".

Continuing the Process: A Third "Sanity Check"

Third condition that must be satisfied:

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

About This Condition:

- Since the set F of accept states is well defined, either $q_i \in F$ (and it should be true that $S_i \subseteq L$) or $q_i \notin F$ (and it should be true that $S_i \cap L = \emptyset$).

Application to the Example:

- Recall that $S_0 = L$ — so that $S_0 \subseteq L$ (and $q_0 \in F$) — while $S_1 \cap L = S_1 \cap S_0 = \emptyset$ (and $q_1 \notin F$).

The Example, So Far

Desired Properties:

- $\delta^\star(q_0, \omega) = q_0$ for every string $\omega \in S_0$, that is, for all ω such that either ω does not include an "a" or ω does not include a "b" (or both);
- $\delta^\star(q_0, \omega) = q_1$ for every string $\omega \in S_1$, that is, for all ω such that ω includes both an "a" and a "b".

Continuing the Process: A Fourth "Sanity Check"

Let i be an integer such that $0 \leq i \leq n-1$ and let $\sigma \in \Sigma$.

- There must also exist an integer j such that $0 \leq j \leq n-1$ and

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

- Thus, if $\omega \in \Sigma^\star$ such that $\omega \in S_i$, then

$$\begin{aligned}
\delta^\star(q_0, \omega \cdot \sigma) &= \delta(\delta^\star(q_0, \omega), \sigma) \quad \text{(as shown in Lecture \#2)} \\
&= \delta(q_i, \sigma) \quad \text{(since } \omega \in S_i \text{, so } \delta^\star(q_0, \omega) = q_i\text{)} \\
&= q_j
\end{aligned}$$

— implying that $\omega \cdot \sigma \in S_j$.

Continuing the Process: A Fourth "Sanity Check"

- Since ω was arbitrarily chosen from $\Sigma^\star$ such that $\omega \in S_i$,

$$\omega \cdot \sigma \in S_j$$

for all strings $\omega \in S_i$ (where $j = \delta(q_i, \sigma)$). That is,

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

Continuing the Process: A Fourth "Sanity Check"

Fourth condition that must be satisfied:

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

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

Then $\delta(q_i, \sigma) = q_j$, when q_i is the state corresponding to S_i and when q_j is the state corresponding to S_j.

Application to the Example

- Easily checked: $\{\omega \cdot \sigma \mid \omega \in S_1\} \subseteq S_1$ for all $\sigma \in \Sigma$ — so that
$$\delta(q_1, \mathtt{a}) = \delta(q_1, \mathtt{b}) = \delta(q_1, \mathtt{c}) = q_1.$$
- Adding the transitions that have been discovered, we now have the following.

Application to the Example

- Also easily checked: $\{\omega \cdot \mathtt{c} \mid \omega \in S_0\} \subseteq S_0$ — so that

$$\delta(q_0, \mathtt{c}) = q_0.$$

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

Application to the Example

Now consider $\{\omega \cdot \mathtt{a} \mid \omega \in S_0\}$.

- Let $\omega \in S_0$. Then three cases might arise.
  (a) *Case:* ω does not include any a's *or* any b's.
  (b) *Case:* ω includes at least one "a" but does not include any b's.
  (c) *Case:* ω includes at least one "b" but does not include any a's.

Application to the Example

Consider case (a): ω does not include any $\mathtt{a}$'s *or* any $\mathtt{b}$'s.

- Then $\omega \cdot \mathtt{a}$ *also* does not include any $\mathtt{b}$'s, so $\omega \cdot \mathtt{a} \in S_0$.
- This case is possible, since this case arises when $\omega = \lambda$.
- Thus
$$\{\omega \cdot \mathtt{a} \mid \omega \in S_0\} \cap S_0 \neq \emptyset.$$
- *Conclusion:* The fourth property can *only* be satisfied if
$$\{\omega \cdot \mathtt{a} \mid \omega \in S_0\} \subseteq S_0,$$
so we need to try to prove this property in the other two cases.

Application to the Example

Consider case (b): ω includes at least one "a" but does not include any b's.

- Then $\omega \cdot \mathtt{a}$ *also* does not include any b's, so $\omega \cdot \mathtt{a} \in S_0$,
- This is consistent with what was discovered in the first case.

Application to the Example

Consider case (c): ω includes at least one "b" but does not include any a's.

- Then $\omega \cdot \mathtt{a}$ includes at least one "a", namely, the copy of "a" at the end.
- $\omega \cdot \mathtt{a}$ also includes at least one "b" because ω does.
- Thus $\omega \cdot \mathtt{a} \in S_1$.
- This case is possible — because $\omega = \mathtt{b}$ is a string in S_0 that includes at least one "b" but does not include any a's.
- *Conclusion:*

$$\{\omega \cdot \mathtt{a} \mid \omega \in S_0\} \cap S_1 \neq \emptyset$$

so it is ***not*** possible that $\{\omega \cdot \mathtt{a} \mid \omega \in S_0\} \subseteq S_0$.

Application to the Example

What This Means:

The fourth “sanity check” has failed.

Application to the Example

- In particular, we have confirmed that it is ***not*** possible to identify a well-defined transition out of the state q_0 for the symbol "a"".
- ***Conclusion (For Now):*** The automaton must remember ***different*** information — or, possibly, ***additional*** information — in order to recognize the language L.

To Be Continued. . .

- The set $\{\omega \cdot \mathtt{b} \mid \omega \in S_0\}$ will be considered in the lecture presentation to obtain more information that might initially be disappointing but that will be useful.
- We will see, next time, that ***all that time and effort was worthwhile*** and that there is a way to make use of what we have done, so far, to solve this problem.