Lecture #2: DFA Design and Verification
Key Concepts

When designing a deterministic finite automaton for a given language one is given the following:

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

The objective is to produce the following:

- A deterministic finite automaton

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

  whose alphabet is the alphabet, Σ, that has been given.

- A ***proof*** that $L = L(M)$ — or enough information so that it is clear that this proof could be written.

DFA design begins by answering the following question: What does a DFA, with the given language, need to remember about the part of the string that has been seen so far?

- This is used to describe a set of *cases* that might apply, for the part of the string that has been seen so far.
- This is used to develop a collection of subsets $S_0, S_1, S_2, \ldots, S_{n-1}$ of subsets of $\Sigma^\star$ — with each subset corresponding to one of the cases that has been identified.[1]

Additional steps (and "sanity checks") include the following.

1. Confirm that only *finitely* many cases — with corresponding sets $S_0, S_1, S_2, \ldots, S_{n-1}$ (for a fixed positive integer n) — have been identified. These will correspond to states $q_0, q_1, q_2, \ldots, q_{n-1}$ in the deterministic finite automaton that is, eventually, designed.

[1] These subsets should be *nonempty*: Any of the sets that are empty can be removed from this collection, to simplify the process and the deterministic finite automaton that is designed using it.

2. Confirm that every string $\omega \in \Sigma^\star$ belongs to *exactly one* of the subsets $S_0, S_1, S_2, \ldots, S_{n-1}$ of $\Sigma^\star$.

   Re-order these states, if necessary, so that $\lambda \in S_0$. Then q_0 can be used as the *start state*.

   The deterministic finite automaton, to be designed, will then include states $q_0, q_1, \ldots, q_{n-1}$.

   The ***desired relationship*** between sets $S_0, S_1, S_2, \ldots, S_{n-1}$ and states $q_0, q_1, q_2, \ldots, q_{n-1}$ can now be described as follows: For every integer i such that $0 \leq i \leq n-1$,

   $$S_i = \{\omega \in \Sigma^\star \mid \delta^\star(q_0, \omega) = q_i\}. \tag{1}$$

3. Confirm that, for every integer i such that $0 \leq i \leq n-1$, either $S_i \subseteq L$ or $S_i \cap L = \emptyset$.

   The set, F, of accepting states can now be defined as follows. For every integer i such that $0 \leq i \leq n-1$, include q_i in F if $S_i \subseteq L$, and do not include q_i otherwise. That is, define F so that

   $$F = \{q_i \mid 0 \leq i \leq n-1 \text{ and } S_i \subseteq L\}.$$

4. Confirm that, 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 $0 \leq j \leq n-1$ such that

   $$\{\omega \cdot \sigma \mid \omega \in S_i\} \subseteq S_j \tag{2}$$

   — in which case, one can set $\delta(q_i, \sigma)$ to be q_j.

Note that, if all of the above conditions have been verified, and q_0, F and the transition function $\delta : Q \times \Sigma \to Q$ have been defined as described above, then a deterministic finite automaton

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

has been completely described.

- Unfortunately, this process is *not* guaranteed to succeed. One way for it to fail involves a discovery that one or more transitions, for the DFA being developed, would not be well-defined.
- If this happens then the ***reason*** for this kind of failure should be carefully examined, to see whether *additional information* about the string processed, so far, was missed on a previous attempt — and should now be added to the information that is being used to design a DFA.
- At this point the design process should be "rolled back" to the beginning and repeated.

- Since this involves additional information, or detail, this can be (arguably) be seen as one form of ***refinement*** — something that is also useful for other design processes.

Students should be able to understand — and use — the following result, to prove that a given DFA has a given language, after following the design process that has been given. It is not necessary for students to understand (or describe) this result's proof, even though students might be able to do that:

Theorem 1 (Correctness of a DFA)**.** *Let $L \subseteq \Sigma^\star$, for an alphabet Σ, and let*

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

be a deterministic finite automaton 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) 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 $L(M) = L$.