Lecture #2: DFA Design and Verification

What To Do Before the Lecture

1. Watch the videos for Lecture #2 —noting that they will probably be understandable if you play them at double speed. If you do not have time for this then look at the "Key Concepts" document that is found, immediately after the videos for this lecture, on the course web site, instead.

2. ***Print*** and read through the rest of this document and — if you have time — try to solve the problems! These should help you to begin (but, not necessarily complete the process, for design of a deterministic finite automaton, that is presented.

   Note that ***supplemental documents*** include another design example, along with a completion of the example discussed in the lecture slides. These might be helpful if you are having trouble understanding the design process that you are to use.

   The supplemental material also includes material that is for interest only, namely, a proof of the technical result, which can be used to prove the correctness of deterministic finite automata, that is introduced at the end of the preparatory material for this lecture.

More About the Example from the Lecture Presentation

Once again, let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and let

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

Recall that, when designing a deterministic finite automaton for L, we considered 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

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

We were trying to design a deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$, whose language is L, such that $Q = \{q_0, q_1\}$ — and such that

$$S_0 = \{\omega \in \Sigma^\star \mid \delta^\star(q_0, \omega) = q_0\} \qquad \text{and} \qquad S_1 = \{\omega \in \Sigma^\star \mid \delta^\star(q_0, \omega) = q_1\}.$$

Unfortunately, this attempt was not successful — because we discovered that the transition out of state q_0 for the symbol "a" was not well-defined.

In order to consider a transition out of state q_0 for the symbol "b", we must consider the set

$$\{\omega \cdot \mathtt{b} \mid \omega \in S_0\}.$$

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 incudes at least one "b" but does not include any a's.

(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:

Conclusion:

Another Design Problem

The Language To Be Considered

Once again, let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and consider the language

$$L = \{\omega \in \Sigma^\star \mid \omega \text{ ends with } \mathtt{abb}\}.$$

Let us try to use the design process, introduced in the preparatory material, to design a deterministic finite automaton (with alphabet Σ) whose language is this language, L

Designing a DFA

First Attempt

As the initial example, suppose we begin by considering a situation where the DFA remembers as little as possible.

What Must the DFA Remember?

Representation Using Subsets of $\Sigma^\star$:

Initial Sanity Checks:

Identifying Transitions:

What Went Wrong: Which Transition was not Well-Defined? Why?

What Could Be Learned From This:

A Second Attempt

What Must the DFA Remember?

Representation Using Subsets of $\Sigma^\star$:

Initial Sanity Checks:

Identifying Transitions:

A Third Attempt

What Went Wrong — Which Transition was not Well Defined? Why?

What Must the DFA Remember?

Representation Using Subsets of $\Sigma^\star$:

Initial Sanity Checks:

Identifying Transitions:

What We Have Now

Writing a Proof Correctness

What Result, from the Lecture Notes, is Useful?

What Things Must Be Established? A detailed list of the properties that should be established is as follows.

Other Things to Think About, or To Do:

Verification vs. Testing