Computer Science 351

Introduction to Deterministic Finite Automata

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #2

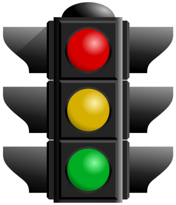
Goals for Today

Goal for Today:

- Introduction of ***deterministic finite automata*** — the simple (abstract) computational devices that will be considered at the beginning of this course

Finite State Machines — An Example

Consider how a simple ***traffic light*** works. For now, let's just worry about the light being shone in *one* direction.



Suppose this light receives (as input) a series of *tick*'s from a timer.

Finite State Machines — An Example

At any point of time (that is, after any input string of *tick*'s has been received and processed), the traffic light is in exactly one of three ***states***:

- q_r: The traffic light is currently *red*.
- q_g: The traffic light is currently *green*.
- q_y: The traffic light is currently *yellow*.

Let us suppose that the traffic light starts by shining *red* — so that its ***initial state*** is the state q_r.

In other words, if the traffic light has not received any *tick*'s yet, at all, then the light is red.

Finite State Machines — An Example

Suppose that we are interested in the input sequences of *tick*'s whose processing allow traffic to move — that is, whose processing cause the traffic light to be in state q_g.

Then q_g should be an ***accepting state***, and q_r and q_y should not — because the sequence of *tick*'s received only allows traffic to move when the light is green.

Finite State Machines — An Example

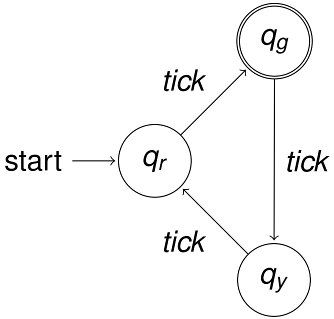
Note that, whenever another *tick* is received and processed, the traffic light ***changes state*** in a well-defined way:

- If the traffic light was in state q_r before this, then it is in state q_g after.
- If the traffic light was in state q_g before this, then it is in state q_y after.
- If the traffic light was in state q_y before this, then it is in state q_r after.

Simplifying Assumption: State changes are assumed to be ***instantaneous***, that is, they do not require any time at all.

Finite State Machines — An Example

The traffic light's behaviour might be represented as follows.



A double circle is used to draw the state q_g in this picture because it is an accepting state.

Fundamental Definitions: Alphabet, Strings, and Languages

Recall that...

- an ***alphabet*** is a finite non-empty set,
- a ***string*** over an alphabet Σ is a finite sequence of symbols in Σ, and
- a ***language*** over Σ is a set of some of the strings over Σ — that is, a subset of the set, $\Sigma^\star$, of all such strings.

Fundamental Definitions: Decision Problems

Definition: A ***decision problem*** is a computational problem for which the output is always either "Yes" or "No."

Every ***alphabet*** Σ and ***language*** L over Σ defines a ***decision problem*** such that

- The set of valid ***instances*** (or "inputs") for the problem is the set $\Sigma^\star$ of all strings over the alphabet Σ.
- The set of "Yes-instances," that is, instances for which the correct output is "Yes," is the language L.
- The set of "No-instances," that is, instances for which the correct output is "No," is the set of strings in $\Sigma^\star$ that *do not* belong to L.

This kind of computational problem will be studied in most of CPSC 351.

Formal Definition

As described above a traffic light is an example of a ***deterministic finite automaton***. This is formally (i.e. mathematically) described as follows.

Definition: A ***deterministic finite automaton (DFA)*** is (formally modelled as) a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite, nonempty, set of ***states***;
- Σ is an ***alphabet*** such that $Q \cap \Sigma = \emptyset$;
- $\delta : Q \times \Sigma \to Q$ is a ***transition function***;
- $q_0 \in Q$ is the ***start state***; and
- $F \subseteq Q$ is the set of ***accept states***.

In particular, δ is a ***total*** function from $Q \times \Sigma$ to Q.

DFA's — Example Continued

Example: The above *traffic light* can be modelled as a deterministic finite automaton

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

where

- $Q = \{q_r, q_g, q_y\}$;
- $\Sigma = \{\mathit{tick}\}$ — that is, the alphabet previously called Σ_0;
- δ is as shown on the following slide;
- q_r is the initial state (as shown by the 5-tuple given above as M); and
- $F = \{q_g\}$ — so that q_g is the only accepting state.

DFA's — Example Continued

As noted above, the ***transition function*** $\delta : Q \times \Sigma \to Q$ is a *total* function from the set of ordered pairs of states and elements of the alphabet to the set of states. For $q, r \in Q$ and $\sigma \in \Sigma$,

$$\delta(q, \sigma) = r$$

if and only if the following holds: If the automaton is in state q and the symbol σ is received, then the automaton moves to state r.

Thus, in the *traffic light* example,

$$\delta(q_r, \mathit{tick}) = q_g, \quad \delta(q_g, \mathit{tick}) = q_y, \quad \text{and} \quad \delta(q_y, \mathit{tick}) = q_r.$$

DFA's — Example Continued

A transition function can be shown by a ***transition table*** in which

- rows are indexed by *states* in Q,
- columns are indexed by *symbols* in Σ, and
- for every state $q \in Q$ and symbol $\sigma \in \Sigma$, the state r in row q and column σ of the table is $\delta(q, r)$.

Thus the transition function for the *traffic light* is given by the following transition table:

	tick
q_r	q_g
q_g	q_y
q_y	q_r

Extended Transition Functions

The ***extended transition function*** is a total function

$$\delta^\star : Q \times \Sigma^\star \to Q$$

such that if the automaton M is in state $q \in Q$, and the (sequence of symbols in) the string $\omega \in \Sigma^\star$ is received, then M is in state $\delta^\star(q, \omega)$ after that.

Thus

$$\delta^\star(q, \lambda) = q$$

for every state $q \in Q$.

Extended Transition Functions

Suppose that

$$\omega = a_1 a_2 \dots a_n \in \Sigma^\star$$

is a string with length $n \geq 0$ in $\Sigma^\star$ — so that, for $1 \leq i \leq n$, the i^{th} symbol in ω is a_i.

Let $q \in Q$. Then a *sequence* of states $r_0, r_1, \dots, r_n$, with length $n + 1$, can be defined as follows:

- $r_0 = q$, and
- $r_{i+1} = \delta(r_i, a_{i+1})$ for every integer i such that $0 \leq i < n$.

Easily Proved by Induction on i: If the automaton M is in state q, and the string $a_1 a_2 \dots a_i$ is received, then M is in state r_i immediately after that.

Thus $\delta^\star(q, \omega) = r_n$, that is, $\delta^\star(q, \omega)$ is the *last* state in the above sequence.

Extended Transition Function

Useful Observation: If $\omega = a_1 a_2 \dots a_n$ is a string with length $n \geq 1$ in $\Sigma^\star$ then

$$\omega = \mu \cdot \tau$$

for the string

$$\mu = a_1 a_2 \dots a_{n-1}$$

with length $n - 1$ in $\Sigma^\star$ and for the symbol $\tau = a_n \in \Sigma$.

Another Equivalent Definition of the Extended Transition Function: For every state $q \in Q$ and string $\omega \in \Sigma^\star$,

$$\delta^\star(q, \omega) = \begin{cases} q & \text{if } \omega = \lambda, \\ \delta(\delta^\star(q, \mu), \tau) & \text{if } \omega = \mu \cdot \tau \text{ for } \mu \in \Sigma^\star \text{ and } \tau \in \Sigma. \end{cases}$$

A ***proof*** of the equivalence of these definitions is given in supplemental material for this lecture.

Extended Application Function

Application of Second Definition: Evaluation of the function!

Example:

$$\begin{aligned}\delta^\star(q_r, \mathit{tick}\ \ \mathit{tick}\ \ \mathit{tick}) &= \delta(\delta^\star(q_r, \mathit{tick}\ \ \mathit{tick}), \mathit{tick}) \\ &= \delta(\delta(\delta^\star(q_r, \mathit{tick}), \mathit{tick}), \mathit{tick}) \\ &= \delta(\delta(\delta(\delta^\star(q_r, \lambda), \mathit{tick}), \mathit{tick}), \mathit{tick}) \\ &= \delta(\delta(\delta(q_r, \mathit{tick}), \mathit{tick}), \mathit{tick}) \\ &= \delta(\delta(q_g, \mathit{tick}), \mathit{tick}) \\ &= \delta(q_y, \mathit{tick}) \\ &= q_r.\end{aligned}$$

In the first three lines, the *second* part of the definition is used (with various choices of ω, μ and τ). The *first* part of the definition is used in the line after that, and then the definition of the (regular) transition function δ is repeatedly used to finish the evaluation.

The Language of an Automaton

Suppose again that M is the deterministic finite automaton

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

Then, for every string $\omega \in \Sigma^\star$, M ***accepts*** ω if and only if

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

and M ***rejects*** ω otherwise.

The ***language*** $L(M)$ of an automaton M is the set of strings $\omega \in \Sigma^\star$ such that M accepts ω.

The Language of an Automaton

For the automaton M modelling the *traffic light*, described above, M accepts the string

$$\omega_i = \textit{tick}\,\textit{tick}\,\ldots\,\textit{tick}$$

with length i if and only if $i = 3 \times k + 1$ for some integer $k \geq 0$ — because

$$\delta^\star(q_r, \omega_i) = q_g$$

if and only if $i = 3 \times k + 1$ for some integer $k \geq 0$.

Thus, if M is the *traffic light* automaton, then $\Sigma = \{\textit{tick}\}$ and

$$L(M) = \{\omega \in \Sigma^\star \mid |\omega| = 3 \times k + 1 \text{ for an integer } k \geq 0\}.$$

Regular Languages

Definition: For every alphabet Σ, a language $L \subseteq \Sigma^\star$ is a ***regular language*** if (and only if) $L = L(M)$ for some deterministic finite automaton M with alphabet Σ.

Properties and applications of regular languages will be considered at the beginning of this course.

Why are DFA's Important?

- ***Easy Exercise:*** Suppose you are given a DFA whose language, Σ, is a subset of Java or Python *characters*. Write a Python or Java program that takes a string $\omega \in \Sigma^{\star}$ as input, and reports whether this string is accepted by the given DFA.
- As noted in the preface to *Introduction to the Theory of Computation*, ***regular expressions*** are extremely useful for string searching and pattern matching.

  We will see, by the end of this unit, that DFA's (and "NFA's," which will be defined shortly) are extremely useful when you are using regular expressions.