6. Finite State Transducers

So far, we have used finite state machines as language recognizers. All we have cared about, in analyzing a machine M, is whether or not M ends in an accepting state. But it is a simple matter to augment our finite state model to allow for output at each step of a machine's operation. Often, once we do that, we may cease to care about whether M actually accepts any strings. Many finite state transducers are loops that simply run forever, processing inputs.

An automaton that produces outputs based on current input and/or previous state is called a **transducer**. Transducers can be of two types:

- **Moore Machine** The output depends only on the current state.
- **Mealy Machine** The output depends both on the current state and the current input.

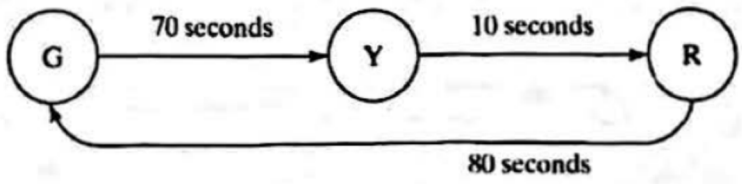
Moore Machine

One simple kind of finite state transducer associates an output with each state of a machine M. That output is generated whenever M enters the associated state. Deterministic finite state transducers of this sort are called Moore machines, after their inventor Edward Moore. A ***Moore machine*** M is a seven-tuple $(K, \Sigma, O, \delta, D, s, A)$, where:

- K is a finite set of states,
- Σ is an input alphabet,
- O is an output alphabet,
- $s \in K$ is the start state,
- $A \subseteq K$ is the set of accepting states (although for some applications this designation is not important),
- δ is the transition function. It is function from $(K \times \Sigma)$ to (K), and
- D is the display or output function. It is a function from (K) to (O^*).

A Moore machine M computes a function $f(w)$ iff, when it reads the input string w, its output sequence is $f(w)$.

Example: Traffic light



Mealy Machine

A different definition for a deterministic finite state transducer permits each machine to output any finite sequence of symbols as it makes each transition (in other words, as it reads each symbol of its input). FSMs that associate outputs with transitions

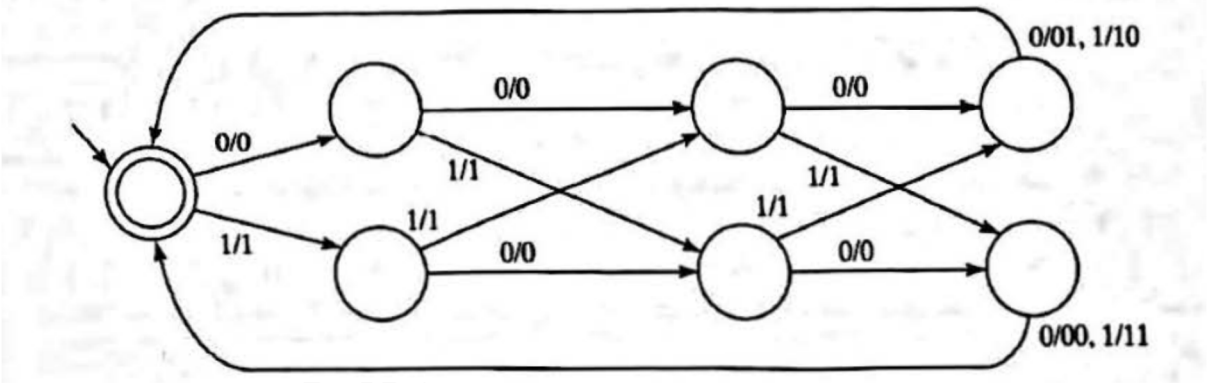
are called Mealy machines, after their inventor George Mealy. A ***Mealy machine*** M is a six-tuple $(K, \Sigma, O, \delta, s, A)$, where:

- K is a finite set of states,
- Σ is an input alphabet,
- O is an output alphabet,
- $s \in K$ is the start state,
- $A \subseteq$ is the set of accepting states, and
- δ is the transition function. It is a function from $(K \times \Sigma)$ to $(K \times O^*)$.

A Mealy machine M computes a function $f(w)$ iff, when it reads the input string w, its output sequence is $f(w)$.

Example: Generating Parity bits

The following Mealy machine adds an odd parity bit after every four binary digits that it reads. We will use the notation a/b on an arc to mean that the transition may be followed if the input character is a. If it is followed, then the string b will be generated.



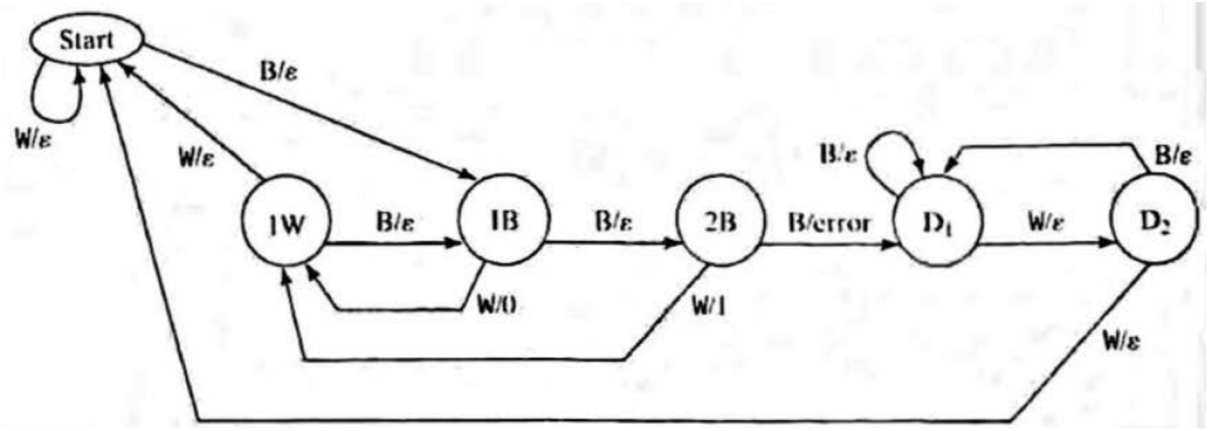
Example: Bar Code reader

Bar codes are ubiquitous. We consider here a Simplification: a bar code system that encodes just binary numbers. Imagine a bar code such as:

It is composed of columns, each of the same width. A column can he either white or black. If two black columns occur next to each other, it will look to us like a single-wide-black column, but the reader will see two adjacent black columns of the standard width. The job of the white columns is to delimit the black ones. A single black column encodes 0. A double black column encodes 1.

We can build a finite state transducer to read such a bar code and output a string of binary digits. We will represent a black bar with the symbol B and a white bar with the symbol W. The input to the transducer will be a sequence of those symbols corresponding to reading the bar code left to right. We'll assume that every correct bar code starts with a black column, so

white space ahead of the first black column is ignored. We will also assume that after every complete bar code there are at least two white columns. So, the reader should, at that point, reset to be ready to read the next code. If the reader sees three or more black columns in a row, it must indicate an error and stay in its error state until it is reset by seeing two while columns.



Interpreters for finite stale transducers can be built using techniques similar to the ones that we used to interpreters for finite state machines.

7. Bidirectional Transducers

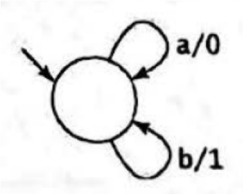
A process that reads an input string and constructs a corresponding output string can be described in a variety of different ways. Why should we choose the finite state transducer model? One reason is that it provides a declarative, rather than a procedural, way to describe the relationship between inputs and outputs, such a declarative model can then be run in two directions.

For example, to read an English text requires transforming a word like "liberties" into the root word "liberty" and the affix PLURAL. To generate an English text requires transforming a root word like "liberty" and the semantic marker “PLURAL” into the surface word "liberties". If we could specify, in a single declarative model, the relationship between surface words (the ones we see in text) and underlying root words and affixes, we could use it for either application.

The facts about English spelling rules and morphological analysis can be described with a bidirectional finite state transducer.

If we expand the definition of a Mealy machine to allow non-determinism, then any of these bidirectional processes can be represented. A nondeterministic Mealy machine can be thought of as defining a relation between one set of strings (for example, English surface words) and a second set of strings (for example. English underlying root words. along with affixes). It is possible that we will need a machine that is nondeterministic in one or both directions because the relationship between the two sets may not be able to be described as a function.

Example: When we define a regular language, it doesn't matter what alphabet we use. Anything that is true of a language L defined over the alphabet {a,b} will also be true of the language L' that contains exactly the strings in L except that every a has been replaced by a 0 and every b has been replaced by a L We can build a simple bidirectional transducer that can convert strings in L to strings in L' and vice-versa.



Of course. the real power of bidirectional finite state transducers comes from their ability to model more complex processes.

*****