Tutorial 7: Finite State Morphology

Alok Debnath

22 February 2019

1 Introduction

Given the basic introduction to computational morphology, we now venture into the domain of finite state machinery in order to computationally represent the morphological rules and patterns we have discussed so far. The machinery used for this representation comes from a basic understanding of states and processes, and how a process leads to change in the state of the system. By modelling a machine as a set of changes in states by valid processes, we verify that:

- Every state that is reached is valid according to the machine. Even if there is an error state, it has been precomputed.
- There is a final state. Sure, loops can be formed, but a final state exists which can be reached.
- There is no process possible which has not been predetermined.

These simple properties are very useful to understand the role of finite state machines in computational linguistics, especially morphology. But first, let us take a simple example of a finite state machine to demonstrate how a real world machine can be broken down into a set of states and processes.

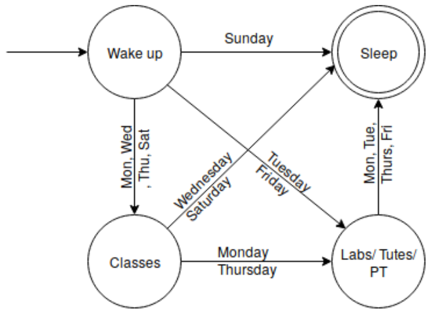

Figure 1: FSA of IIIT First Years

Figure 1 is a representative diagram of the college machinery (time table) for the first year students. Note that there are four basic states, with the processes being the days of the week. This should give you a basic idea of how to map a system to its states or processes. A state table can be made which shows the events during each day.

2 What are FSAs and FSTs?

In this section, we define finite state automata and finite state transducers. These are important terminology, so bear in mind the difference.

A finite state automata (FSA) is defined as a quintuple (Σ, Q, i, F, E) where:

- Σ is the alphabet
- Q is a finite set of states
- $i \in Q$ is a single, initial state
- $F \subseteq Q$ is a set of final states
- $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$ is the set of valid arc transitions from one state to the other.

Note that there is only one alphabet, Σ, which makes the applications into computational linguistics a little harder, as we would like to provide the information of the actual morphological changes taking place in the words if there is an addition of an alphabet. This is where we can use FSTs.

Unlike FSAs which only have one input alphabet, FSTs have an input alphabet Σ_i and an output alphabet Σ_o. None of the other properties undergo a change, other E, which is now defined as $E \subseteq Q \times (\Sigma_i \cup \{\epsilon\} \times \Sigma_o \cup \{\epsilon\}) \times Q$.

3 Applications in Computational Morphology

In computational morphology, we can associate certain morphological properties to the addition or removal of certain words (suffixes). This can be done using FSAs, FSTs or tries. A trie is a data structure with edge labels are the letters, therefore you can read the word out from the edges.

The rest of this tute is going to be taught on blackboard and from the Jurafsky and Martin book uploaded on Moodle. I do not want you all to become dependent on my notes for this course.