FST Technology

(based on Beesley and Karttunen 2003, Jurafsky and Martin 2000)

Miriam Butt

October 2003

Regular Expression

- Formula for a special language that can specify simple classes of strings (Kleene 1956).
- Most popular use: search for patterns in a corpus (e.g., Webbrowser and the Internet).
- Examples:

/a/	any "a" in a text
/[A-Z]/	any upper case letter
/[0-9]/	any single digit

Regular Expressions

- The "sheep language": /baa+!/
- Produces:

  baa!
  baaa!
  baaaa!
  baaaaa!
  baaaaaa!
  baaaaaaa!
  ...

- Kleene *: 0 to infinitely many
- Kleene +: 1 to infinitely many

Regular Expressions+FSA

Finite-State Automata/Machines can be used to implement regular expressions.

FSA: Background

Finite State Machines and their properties are well known mathematical objects.

However, their uses for natural language processing have only really been explored since the mid 80s.

The current effort is pioneered by

- Kenneth Beesley
- Ronald Kaplan
- Lauri Karttunen
- Martin Kay
- Kimmo Koskenniemi

Background

Finite Automata go back to Alan Turing’s (1936) model of algorithmic computation.

Turing Machine:

Abstract machine with a finite control and an input/output tape. In one move, it could read a symbol on the tape, change state and move left or right.

Things FST can be used for

Finite-State Morphological Analyzers/ Generators (Lexical Transducers).

Finite-State Tokenizers (divide an input string into tokens)

Finite-State POS Taggers (noun, verb, etc.)

Finite-State shallow syntactic parsers

Things FST can be used for

Morphological Analysis:

In Word Formation *morphemes* can only appear in a certain order and in certain combinations:

piti-less-ness	**piti-ness-less*
un-guard-ed-ly	**un-elephant-ed-ly*

Things FST can be used for

Phonological/Orthographical Alternation

pity → *piti* when followed by *less*
fly → *flie* when followed by *s*
swim → *swimm* when followed by *ing*

Things FST can be used for

Basic Insight (work developed at Xerox):

1. The legal combination of morphemes (morphotactics) can be encoded as a finite-state network.
2. The rules that determine the form of each morpheme (alternation) can be implemented as finite-state transducers.
3. A lexicon network and the rule transducers can be composed together into a single network (lexical transducer). This contains all the morphological information about a language (morphemes, derivation, inflection, compounding, etc.)

Efficiency

Finite State Lexical Transducers are mathematically well understood and therefore highly efficient.

Example: Xerox Spanish Morphology (1996 version)

46 0000 base forms

3 400 000 inflected word forms (generated/analyzed)

3349 kbytes of memory

can be further compressed for commerical applications

FST Basics

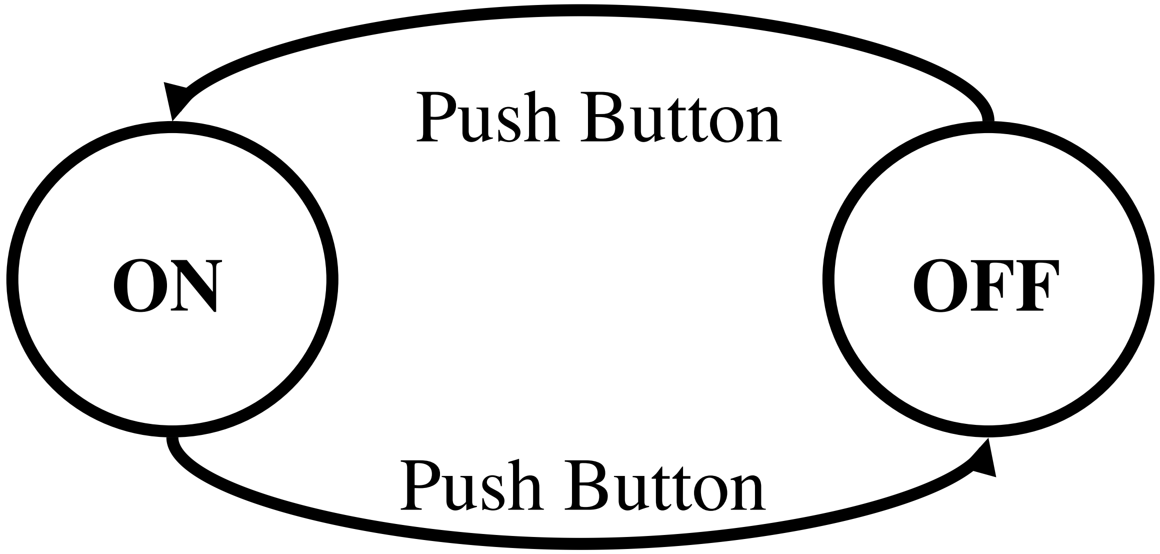
In order to understand how to build the linguistic applications, we first need to get acquainted with the basics of how finite-state machines work.

The basic concepts are very simple, but the possibilities allowed by composing differing transducers become extremely complex (and versatile) very quickly.

States

Two States of an On-Off Light Switch

A Network: States and Transitions



On-Off Toggling Switch

Beyond the Binary

$$\begin{tikzcd}
\mathbf{OFF} \arrow[r, bend right, "\text{Right}"'] & \mathbf{LOW} \arrow[l, bend right, "\text{Left}"'] \arrow[r, bend right, "\text{Right}"'] & \mathbf{MED} \arrow[l, bend right, "\text{Left}"'] \arrow[r, bend right, "\text{Right}"'] & \mathbf{HIGH} \arrow[l, bend right, "\text{Left}"']
\end{tikzcd}$$

Intermediate Settings Between Off and High

Beyond the Binary

$$\begin{tikzcd}
\textbf{OFF} \arrow[r, bend right, "\text{Right}"'] & \textbf{LOW} \arrow[r, bend right, "\text{Right}"'] & \textbf{MED} \arrow[r, bend right, "\text{Right}"'] & \textbf{HIGH} \arrow[lll, bend right, "\text{Left}"']
\end{tikzcd}$$

A Three-Way Light Switch

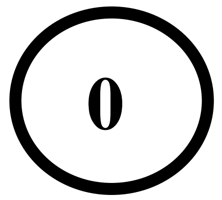
Finiteness

FSTs are finite in the sense that they are not infinite.

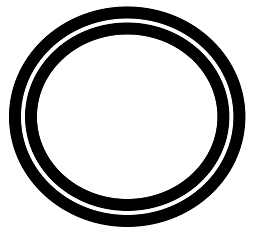
They can model any finite number of states, but not an infinite gradation of states.

Example: While an FST can model a light switch, it could not model a light dimmer.

Some Basic Concepts/ Representations

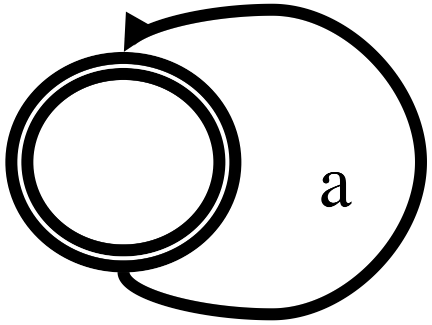


Initial State

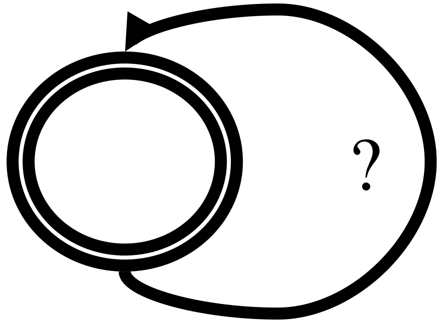


Final State

Some Basic Concepts/ Representations

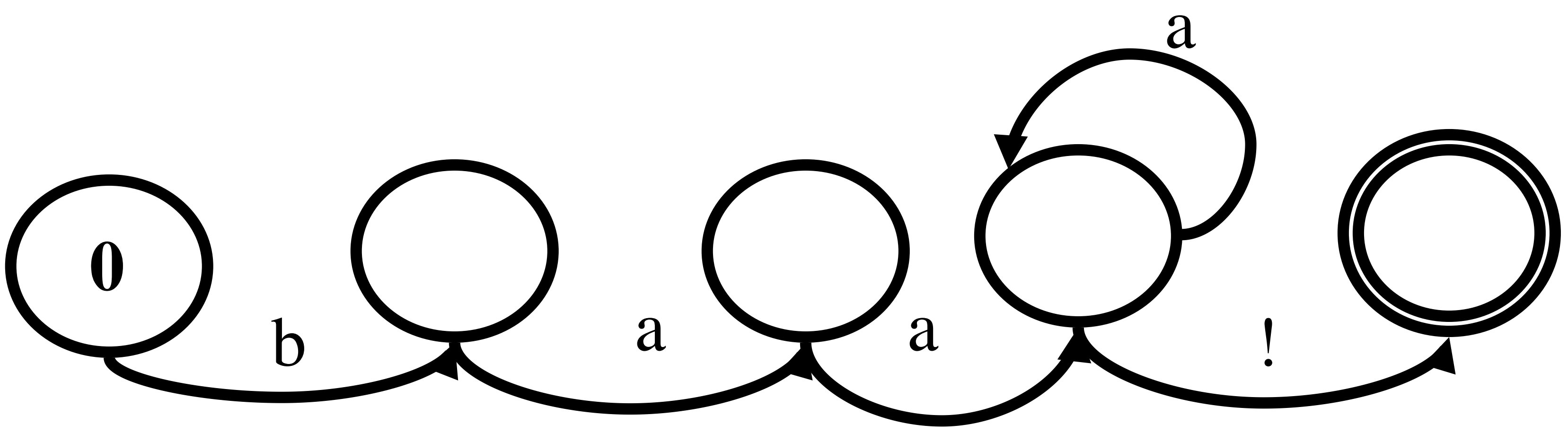


A Network for an Infinite Language



A Network for the Universal Language

The Sheep Language



Coke Automaton

In a sense, FSTs are like little robots, who can go off and do things for you. Hence the term Automoton.

Example: A Coke Machine

A coke costs 65 cents. Build a machine that will accept nickels (N), dimes (D) and quarters (Q) in any order in order to return a coke.

(A version of this can be found on the XRCE Finite-State Web page)

The Connection to Language

If the inputs to the coke machine are taken to be letter SYMBOLS, then:

- The set of valid symbols the machine accepts is its ALPHABET.
- The sequences of symbols are WORDS.
- The entire set of words is the LANGUAGE.