FST Morphology

Based on Beesley and Karttunen 2002

Miriam Butt
October 2002

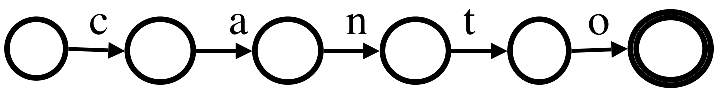
Recap

Last Time: Finite State Automata can model most anything that involes a finite amount of states.

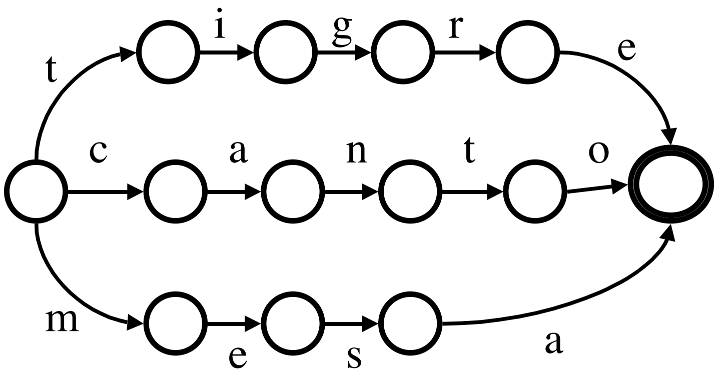
We modeled a Coke Machine and saw that it could also be thought of as defining a *language*.

We will now look at the extension to natural language more closely.

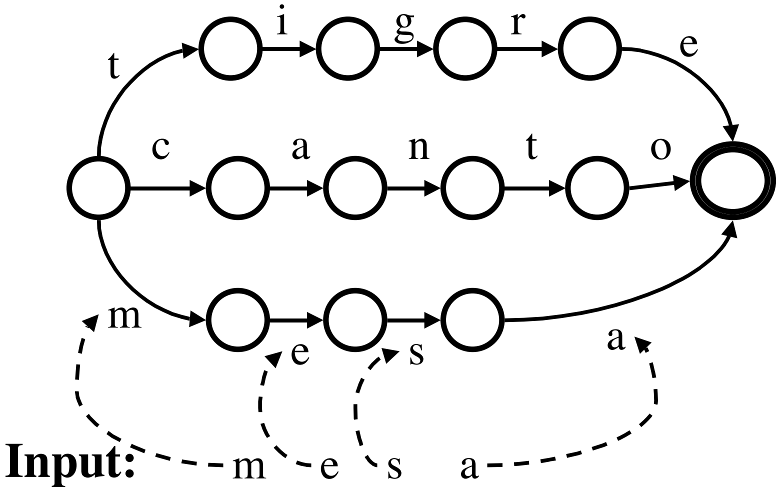
A One-Word Language



A Three-Word Language



Analysis: A Successful Match

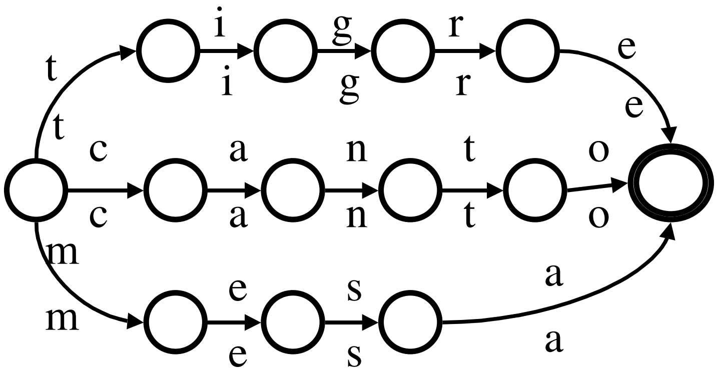


Rejects

The analysis of *libro, tigra, cant, mesas* will fail.

Why?

Transducers: Beyond Accept and Reject

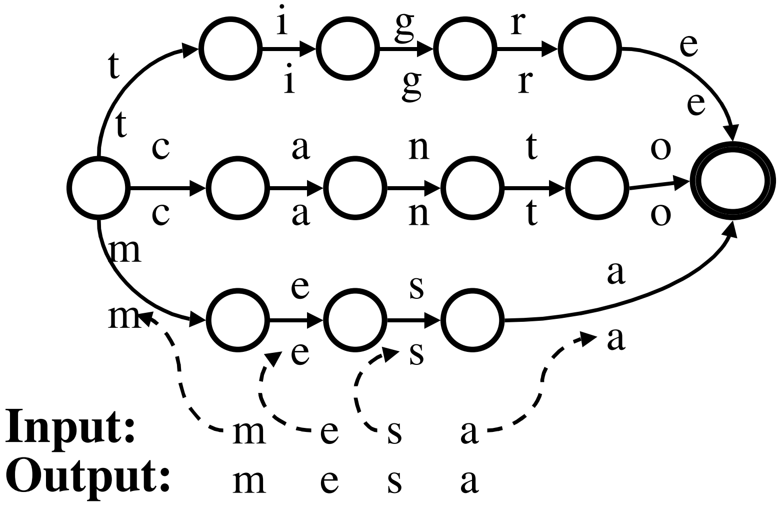


Transducers: Beyond Accept and Reject

Analysis Process:

- Start at the Start State
- Match the input symbols of string against the *lower-side* symbol on the arcs, consuming the input symbols and finding a path to a final state.
- If successful, return the string of *upper-side* symbols on the path as the result.
- If unsucessful, return nothing.

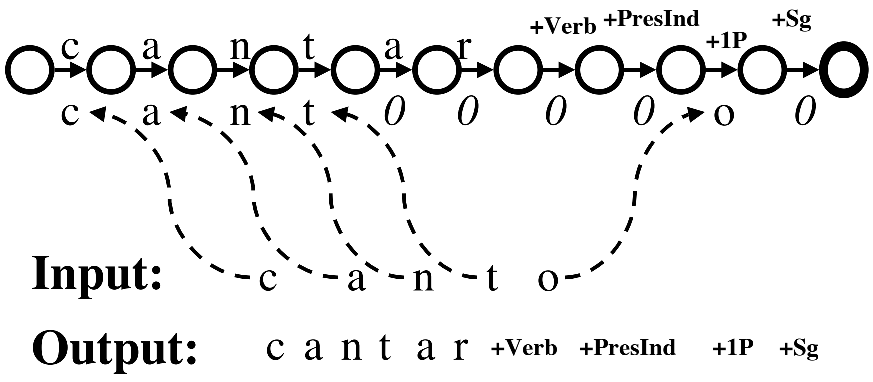
A Two-Level Transducer



Input: m e s a

Output: m e s a

A Lexical Transducer

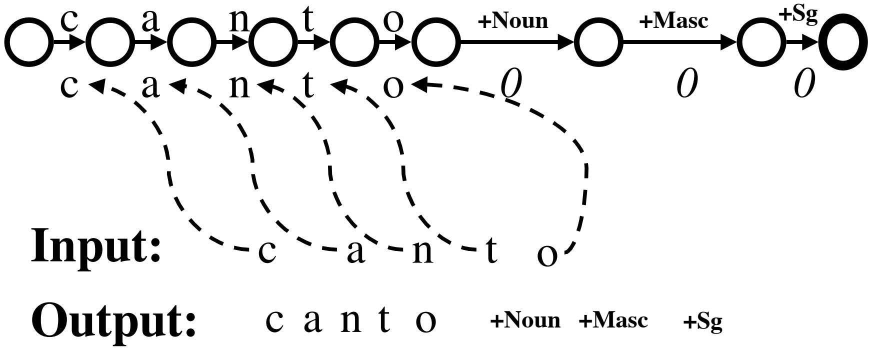


Input: c a n t o

Output: c a n t a r +Verb +PresInd +1P +Sg

One Possible Path through the Network

A Lexical Transducer



Input: c a n t o

Output: c a n t o +Noun +Masc +Sg

Another Possible Path through the Network

The Tags

Tags or Symbols like +Noun or +Verb are *arbitrary*: the naming convention is determined by the (computational) linguist and depends on the larger picture (type of theory/type of application).

One very successful tagging/naming convention is the Penn Treebank Tag Set

The Tags

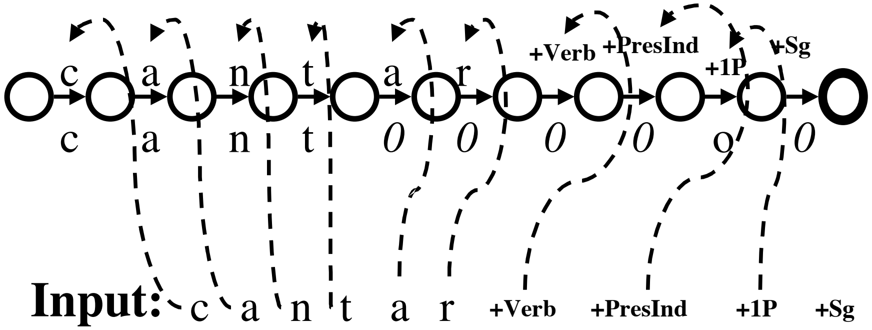
What kind of Tags might be useful?

Generation vs. Analysis

The same finite state transducers we have been using for the *analysis* of a given surface string can also be used in reverse: for *generation*.

The XRCE people think of analysis as *lookup*, of generation of *lookdown*.

Generation --- Lookdown



Input: c a n t a r +Verb +PresInd +1P +Sg

Output: c a n t o

Generation --- Lookdown

Analysis Process:

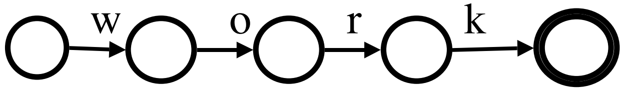
- Start at the Start State and the beginning of the input string
  - Match the input symbols of string against the *upper-side* symbols on the arcs, consuming the input symbols and finding a path to a final state.
  - If successful, return the string of *lower-side* symbols on the path as the result.
  - If generation is unsucessful, return nothing.

Concatenation

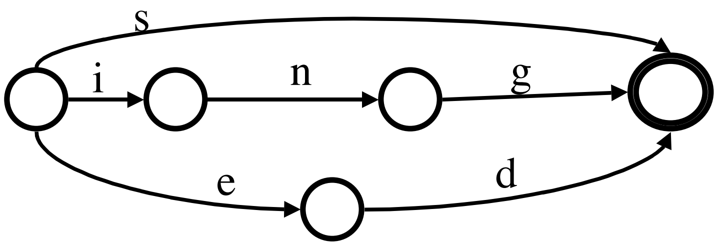
One can also concatenate two existing languages (finite state networks with one another to build up new words productively/dynamically.

This works nicely, but one has to write extra rules to avoid things like: **trys*, **tryed*, though *trying* is okay.

Concatenation

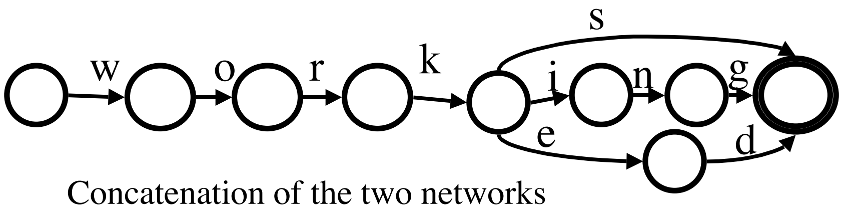


Network for the Language {"work"}



Network for the Language {"s", "ed", "ing"}

Concatenation



Concatenation of the two networks

What strings/language does this result in?

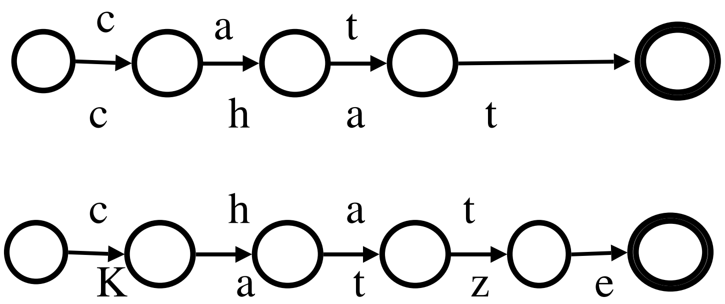
Composition

Composition is an operation on two relations.

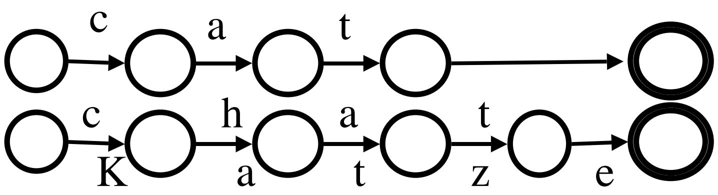
Composition of the two relations <x,y> and <y,z> yields <x, z>

Example: <"cat", "chat"> with <"chat", "Katze"> gives <"cat", "Katze">

Composition

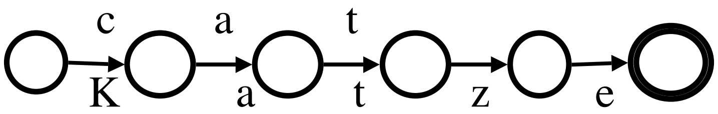


Composition



Merging the two networks

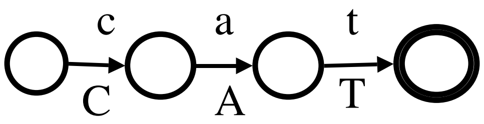
Composition



The Composition of the Networks

What is this reminiscent of?

Other Uses for the Transducers



Upper/Lower Casing