Finite State Transducers

Eric Gribkoff

May 29, 2013

Original Slides by Thomas Hanneforth (Universitat Potsdam)

Outline

1 Definition of Finite State Transducer

A Non-Deterministic **Finite State Transducer** (FST) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, \omega, q_0, F)$

1. Q is a finite set called the **states**
2. Σ is a finite set called the **alphabet**
3. Γ is a finite set called the **output alphabet**
4. $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$ is the **transition function**
5. $\omega : Q \times \Sigma \cup \{\varepsilon\} \times Q \to \Gamma^*$ is the **output function**
6. $q_0 \in Q$ is the **start state**
7. $F \subseteq Q$ is the **set of accept states**

A Deterministic **Finite State Transducer** (FST) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, \omega, q_0, F)$

1. Q is a finite set called the **states**
2. Σ is a finite set called the **alphabet**
3. Γ is a finite set called the **output alphabet**
4. $\delta : Q \times \Sigma \to Q$ is the **transition function**
5. $\omega : Q \times \Sigma \to \Gamma$ is the **output function**
6. $q_0 \in Q$ is the **start state**
7. $F \subseteq Q$ is the **set of accept states**

2 Examples of FSTs

The action of a **Finite State Transducer** can be viewed as computing a relation between two sets.

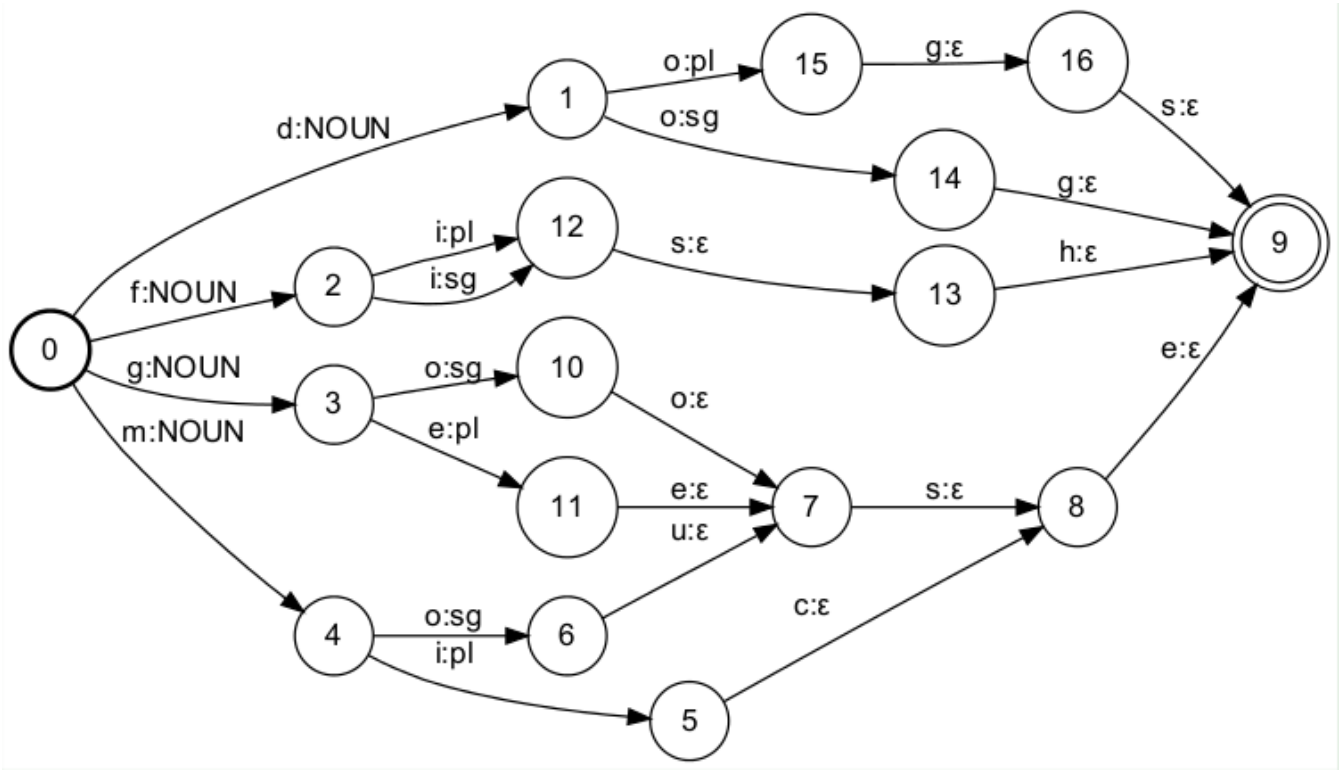


Figure 1: T_{lex} maps words to morphological features

dog $\rightarrow$ NOUN sg (singular)

dogs $\rightarrow$ NOUN pl (plural)

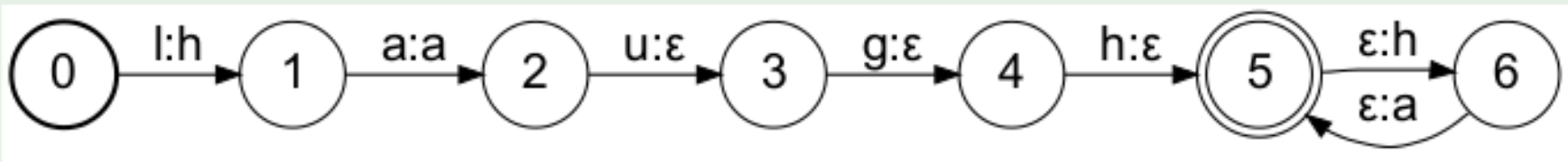


Figure 2: Laughing machine T_{laugh}

The input string laugh is mapped to the infinite set $\{ha^n | n \geq 1\}$

(laugh, ha)

(laugh, haa)

(laugh, haaa)

$\vdots$

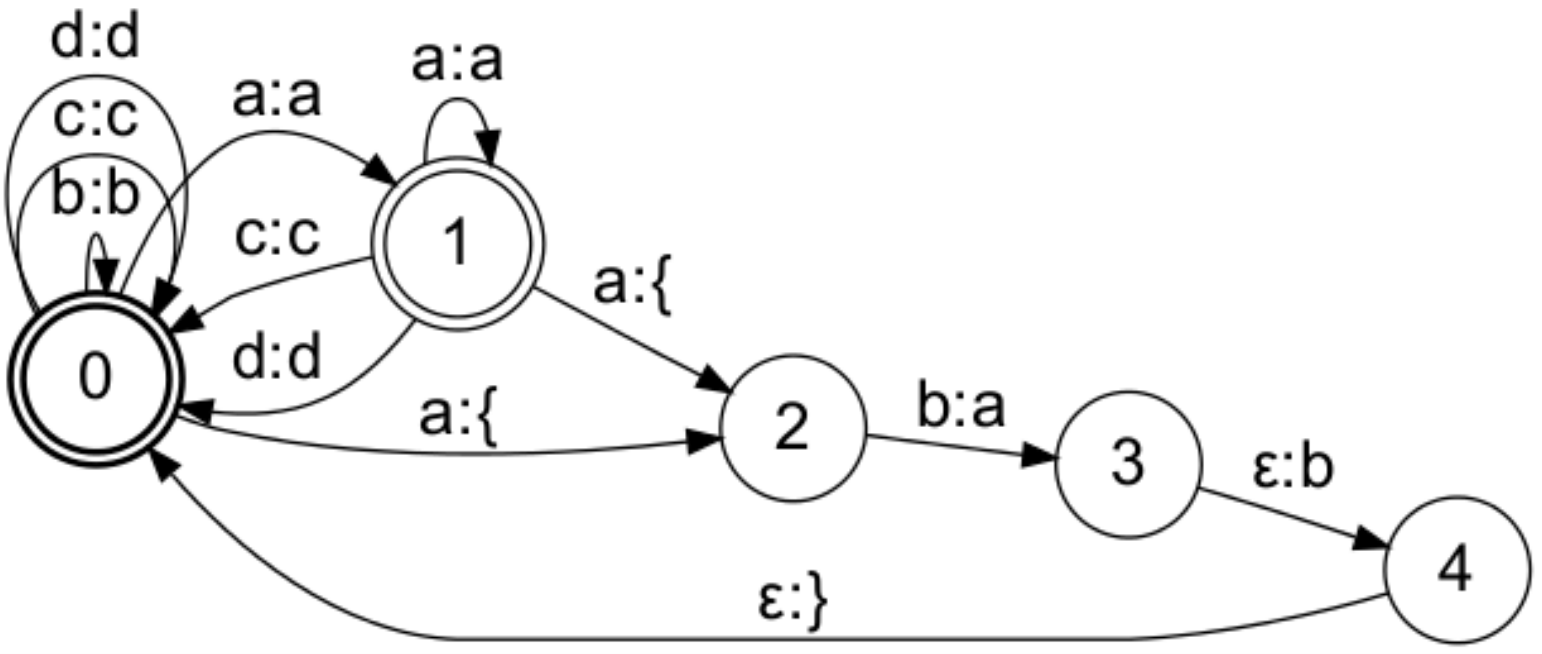


Figure 3: Bracketing machine $T_{bracket}$

Every occurrence of ab is enclosed within brackets.

$$cabbabc \rightarrow c\{ab\}b\{ab\}c$$

3 Definition of Regular Relations

A **relation** over the sets A and B is a subset of the Cartesian product, $A \times B$.

Ex: Let $A = \{dog, cat, cow\}$, $B = \{seven, \pi, octopus\}$. Then $R = \{(dog, seven), (cow, \pi), (cow, octopus)\}$ is a relation.

A **regular (or rational) relation** over the alphabets Σ, Γ is formed from a finite combination of the following rules:

1. $\forall\ (x, y) \in \Sigma \cup \{\varepsilon\} \times \Gamma \cup \{\varepsilon\}$
2. $\emptyset$ is a regular relation
3. If R, S are regular relations, then so are $R \circ S$, $R \cup S$, and R^*

4 Language of an FST

The **language** $L(T)$ of a non-deterministic FST $T = (Q, \Sigma, \Gamma, \delta, \omega, q_0, F)$ is defined using the extended transition and output functions δ^*, σ^*:

$$L(T) = \{(u, v) \mid \delta^*(q_0, u) \cap F \neq \emptyset \ \wedge \ v \in \sigma^*(q_0, u)\}$$

$$\delta^*(q, \varepsilon) = \{q\}$$

$$\delta^*(q, wa) = \bigcup_{q' \in \delta^*(q,w)} \delta(q', a)$$

$$\sigma^*(q, \varepsilon) = \{\varepsilon\}$$

$$\sigma^*(q, wa) = \sigma^*(q, w) \circ \bigcup_{q' \in \delta^*(q,w)} \bigcup_{p \in \delta(q', a)} \sigma(q', a, p)$$

Explanation of δ^*

δ^* is the extended transition function. It returns the set of possible states after processing a state and input string pair.

$\delta^*(q, \varepsilon) = \{q\}$ means that if the input is the empty string, we will remain at the current state

$\delta^*(q, wa) = \bigcup_{q' \in \delta^*(q,w)} \delta(q', a)$

$\bigcup_{q' \in \delta^*(q,w)}$ is a union over all states reachable on input string w from state q

$\delta(q', a)$ is the non-deterministic transition function, returning the set of possible states given state q and input character a

Explanation of σ^*

σ^* is the extended output function. It returns the set of possible output strings after processing a state and input string pair.

$\sigma^*(q, \varepsilon) = \{\varepsilon\}$ (Missing output from ε-transitions?)

$\sigma^*(q, wa) = \sigma^*(q, w) \circ \bigcup_{q' \in \delta^*(q,w)} \sigma(q', a, p),\ p \in \delta(q', a)$

$\sigma^*(q, w)$ is the set of possible outputs after processing w from state q

$\bigcup_{q' \in \delta^*(q,w)}$ is a union over all states q' reachable on input string w from state q

$\bigcup_{p \in \delta(q',a)}$ is a union over all states p reachable from q'

Explanation of σ^*, continued

$\sigma(q', a, p)$ is the non-deterministic output function, returning the set of possible output strings given the transition from state q' to state p on input a.

Parallel with DFAs

A Deterministic Finite Automaton recognizes a **regular language**

Similarly, a Finite State Transducer recognizes (or encodes) a **regular relation**

5 Closure Properties of Regular Relations

Regular relations are closed under the following operations:

- Concatenation
- Union
- Closure (Star)
- Composition

5.1 Composition

Composing an FST T_1 with an FST T_2 means taking an input u for T_1, collecting the output v of T_1 , feeding v as input to T_2, and collecting the output w of T_2.

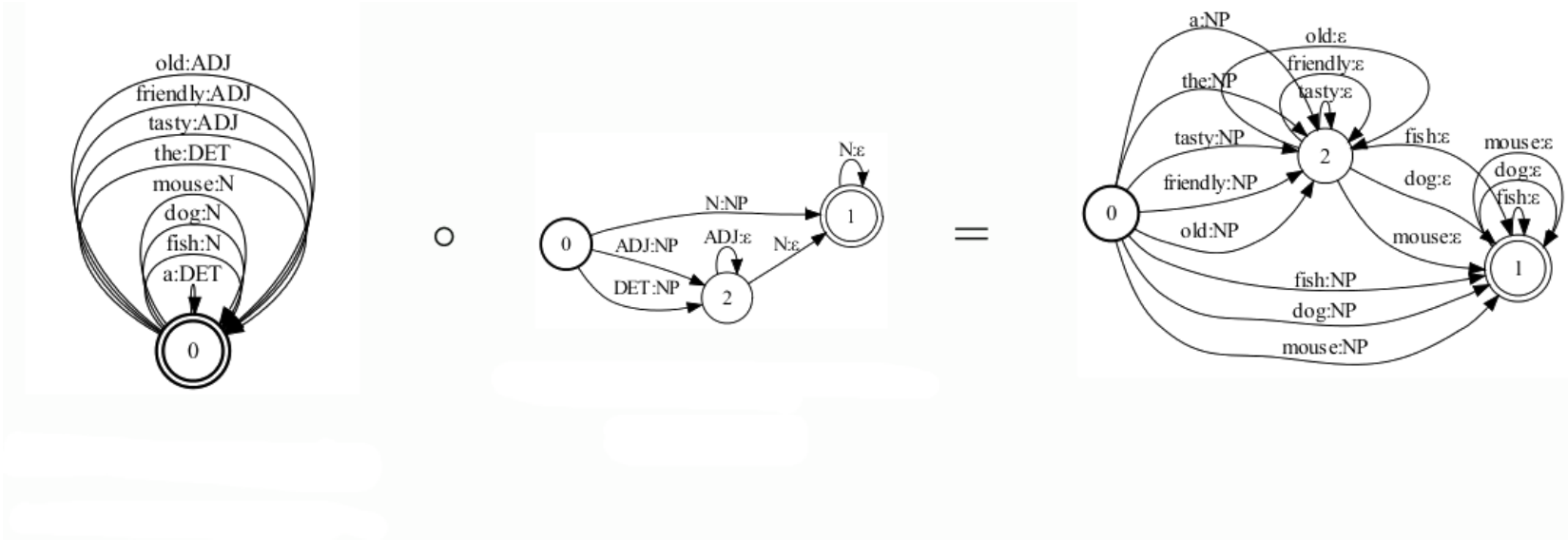


Figure 4: Compositions of FSTs

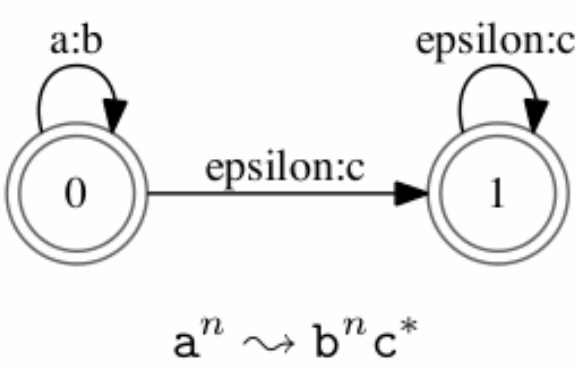
The leftmost FST maps words to their categories. The middle FST maps noun phrase-patterns to noun-phrase categories. Their composition accepts valid noun-phrases.

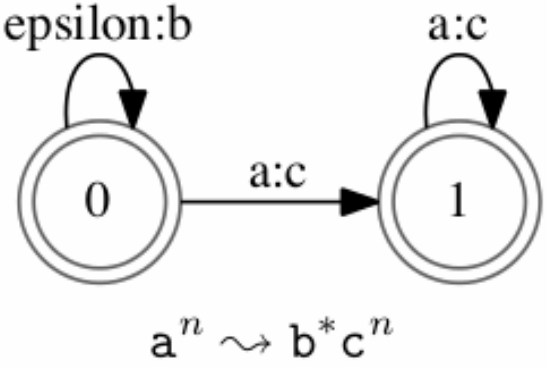
Closure Properties of Regular Relations

Unlike regular languages, regular relations are not closed under:

- Intersection
- Complementation
- Difference

5.2 Intersection





The regular relation $(a^n, b^n c^*)$ can be formed by the concatenation of $(a, b)^*$ and $(\varepsilon, c)^*$. Likewise, $(a^n, b^* c^n) = (\varepsilon, b)^* \circ (a, c)^*$.

Their intersection, $(a^n, b^n c^n)$, is not a regular relation.

The lack of closure under complementation and difference follows from De Morgan's law.

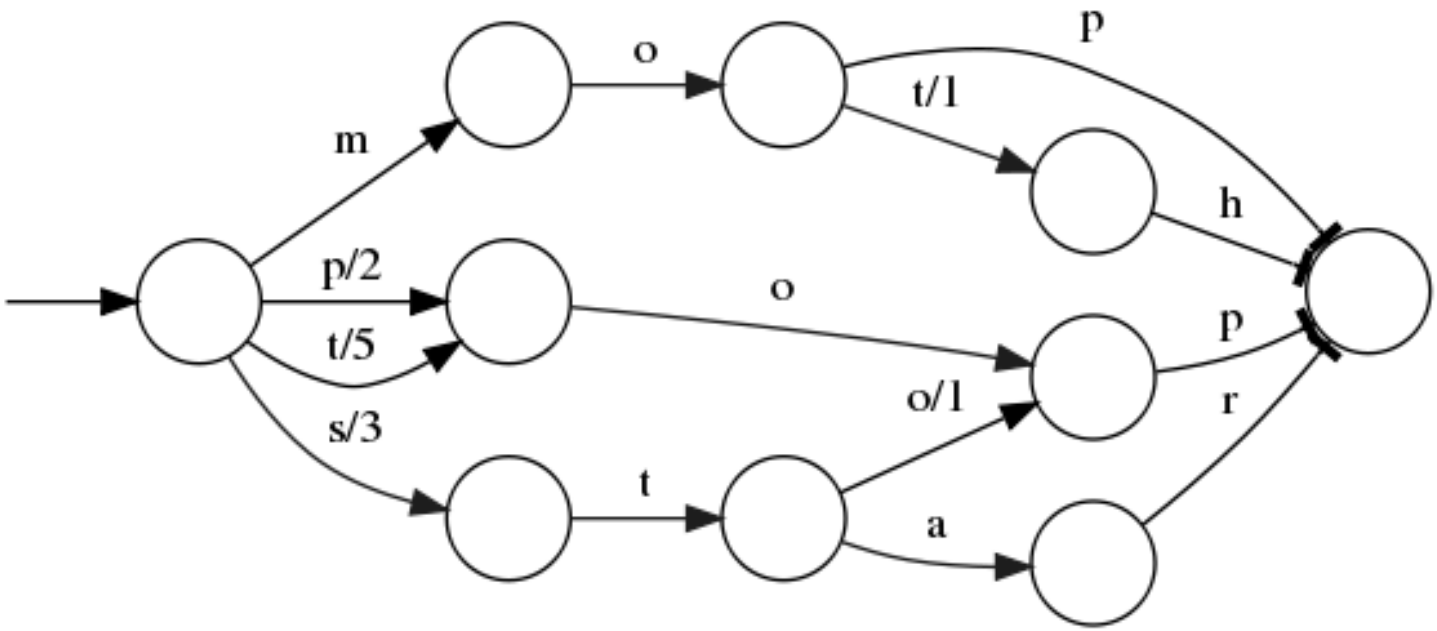
6 FSTs Applied to String Indexing

FSTs can be used to map between a set of words and the index of each word in a sorted array

Define a regular relation between the set of words and their numbering within the alphabetically sorted set

With an FST encoding this relation, words are matched to their index term

But its more space efficient to construct an FST with shared paths for common prefixes and suffixes



The words {mop, moth, pop, star, stop, top} are matched to their ordinal index by summing the digits of the output string.

Efficient algorithms exist to generate these FSTs for lists of millions of words.

Q & A