Finite State Transducers

Maha Alkhairy

May 6, 2024

1 Introduction

A finite state machine is a machine which consists of states and transitions and can either be deterministic or non-deterministic. In a deterministic finite state machine each state has exactly one transition for every symbol in the alphabet. A finite state machine can either be a finite state automata (FSA), which accepts strings in the language it models, or a finite state transducer which coverts an input string to an output string using contextual or non-contextual replacement, insertion, or deletion.

FSAs and FSTs can be written using regular expressions and are closed under operations such as concatenation and union. FST is bidirectional and hence input and output can be inverted for the same FST[1].

A Finite State Transducer (FST) is a finite state machine which transforms an input in one language to an output in another language. FSTs and their variations [17, 18, 15] have been used to model morphology [16, 1, 6, 5, 12, 2, 13, 11, 10] and phonology [14, 1, 8, 19, 3].

There are two types of Finite State Machines that are similar to Finite State Transducers and often categorized as type of FSTs: Moore machines [9] and Mealy machines. The formal definitions are defined in Sections 2 and 3. A Moore machine [21, 4] is a machine where the output only depends on the states, whereas a Mealy machine [20] is machine where the output depends on both the states and the transitions.

A Moore machine can always be transformed to a Mealy machine but not the other way around [9]. Mealy and Moore machines do not have accept states.

The main difference between Mealy machines and FSTs are that FSTs have accept states whereas Mealy machines do not. In addition, Moore and Mealy machines are deterministic whereas FSTs can either be determinisitic or nondeterministic.

[1] if it is non-deterministic it might not always be be a one to one correspondence as with non-determinism we can have transitions

2 Moore Machine

A **Moore machine** [21] can be defined formally as a 6-tuple $(Q, s_0, \Sigma, \Gamma, \delta, \Pi)$ where:

$Q :$ the set of states

$s_0 \in Q :$ the start

Σ: the input alphabet

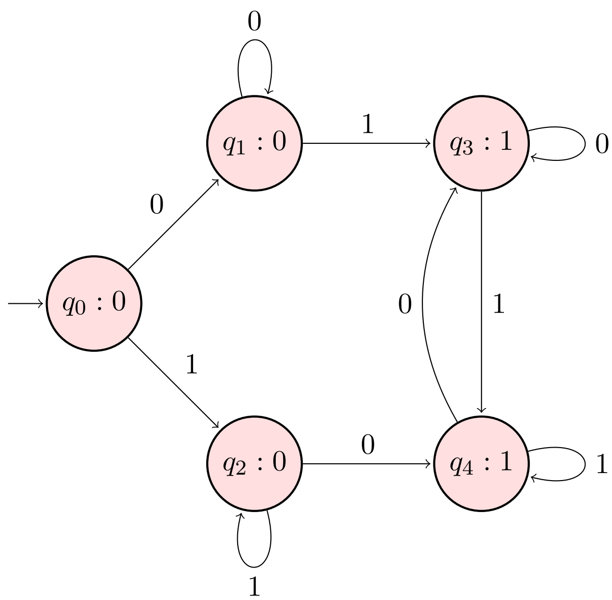
Γ: the output alphabet

$\delta : Q \times \Sigma \to Q :$ transition function from states and Σ to the states

$\Pi : Q \to \Gamma :$ output function from states to the output alphabet.

2.1 Example of a Moore Machine

A Deterministic Moore Machine [7]:



In this example:

$Q = \{q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1\}$

δ :

State	0	1
q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_4	q_2
q_3	q_3	q_4
q_4	q_3	q_4

$\Pi = \{(q_0, 0), (q_1, 0), (q_2, 0), (q_3, 1), (q_4, 1)\}$

Lets demonstrate input $\rightarrow$ output examples on this machine:
11 $\rightarrow$ 000
01 $\rightarrow$ 001
10 $\rightarrow$ 011
00 $\rightarrow$ 000

3 Mealy Machine Formal Definition

A Mealy machine[20] can be formally define as a 6-tuple $(Q, s_0, \Sigma, \Gamma, \delta, \Pi)$ where:

$Q :$ the set of states

$s_0 \in Q :$ the start state

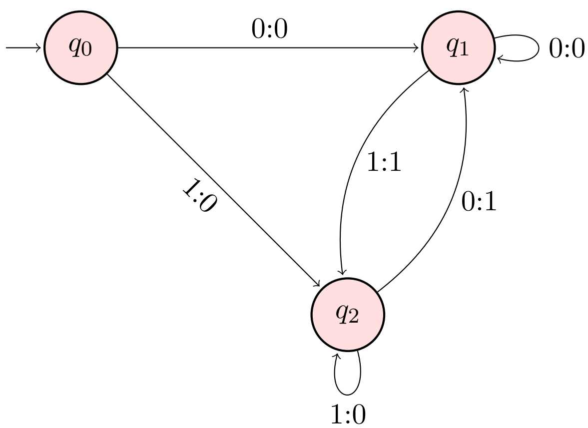
$\Sigma :$ the input alphabet

$\Gamma :$ the output alphabet

$\delta : Q \times \Sigma \rightarrow Q :$ transition function from states and Σ to the states

$\Pi : Q \times \Sigma \rightarrow \Gamma :$ output function from states and Σ to the output alphabet

3.1 Example of a Mealy Machine



In this example:

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1\}$

$\delta :$

State	**0**	**1**
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_1	q_2

$\Pi =$

State	**0**	**1**
q_0	0	0
q_1	0	1
q_2	1	0

Lets demonstrate input $\rightarrow$ output examples on this machine:
$11 \rightarrow 00$
$01 \rightarrow 01$
$10 \rightarrow 01$
$00 \rightarrow 00$
$001100 \rightarrow 001010$

4 Finite State Transducer

A **Deterministic Finite State Transducer** (FST) is represented as a 7-tuple $(Q, s_0, \Sigma, \Gamma, \delta, \Pi, F)$ where:

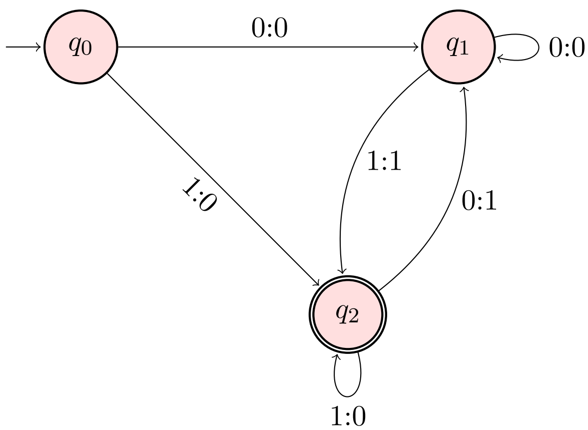
1. $Q :$ the set of states
2. $\Sigma :$ the input alphabet
3. $\Gamma :$ the output alphabet.
4. $\delta : Q \times \Sigma \rightarrow Q :$ the transition function from states and Σ to the states
5. $\Pi : Q \times \Sigma \rightarrow \Gamma :$ output function from states and Σ to the output alphabet
6. $s_0 \in Q$ is the start state.
7. $F \subseteq Q$ is the set of accept states.

A **Nondeterministic Finite State Transducer** (FST) is represented as a 7-tuple $(Q, s_0, \Sigma, \Gamma, \delta, \Pi, F)$ where:

1. Q : the set of states
2. Σ : the input alphabet
3. Γ : the output alphabet.
4. $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$: the transition function from states and Σ to the states
5. $\Pi : Q \times \Sigma \cup \{\varepsilon\} \to \Gamma^*$: output function from states and Σ to the output alphabet
6. $s_0 \in Q$ is the start state.
7. $F \subseteq Q$ is the set of accept states.

4.1 Examples of Finite State Transducers

4.1.1 Deteministic



In this example:

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1\}$

δ :

State	**0**	**1**
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_1	q_2

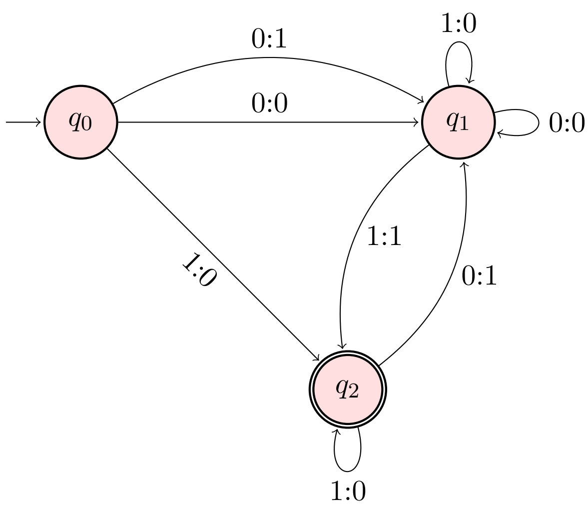
State	0	1
q_0	0	0
q_1	0	1
q_2	1	0

$\Pi =$ (table above)

$F = \{q_2\}$

Lets demonstrate input → output examples on this machine:
11 → 00
01 → 01
0011001 → 0010101

4.1.2 Nondeterministic



In this example:

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1\}$

$\delta :$

State	0	1
q_0	$\{q_1\}$, $\{q_1\}$	$\{q_2\}$
q_1	$\{q_1\}$, $\{\}$	$\{q_2\}$, $\{q_1\}$
q_2	$\{q_1\}$	$\{q_2\}$

$\Pi =$

State	0	1
q_0	0, 1	0
q_1	0	1, 0
q_2	1	0

$F = \{q_2\}$

Let's demonstrate input $\rightarrow$ output examples on this machine:
$11 \rightarrow 00$
$01 \rightarrow 01$ or 11 or 10
$0011001 \rightarrow 0010101$

4.2 Comparisons with Finite State Automata (FSAs)

- A Finite State Automata recognizes a **regular language**.
- A Finite State Transducer encodes a **regular relation**.

As with regular relations, FSTs are closed under:

- Concatenation
- Union
- Kleene star (M^*)
- Composition
- Projection (projects to FSAs)
- Inversion

Concatenation, Union, Kleene star constructions of FSTs is the same as that for FSAs an example of Kleene star in in Figure 3.

As with regular relations, FSTs are **NOT** closed under:

- Intersection [2]
- Complementation
- Difference

4.3 Illustration of Closure Properties

The definitions and examples below are adapted from `https://core.ac.uk/download/pdf/24059977.pdf`, `https://dsacl3-2019.github.io/slides/fst.pdf`, and `https://www.cs.sfu.ca/~anoop/courses/MACM-300-Spring-2006/fst.pdf` which illustrate the closure properties.

4.3.1 Composition

Composing an FST M_1 with another FST M_2 [3] produces an FST.

Let $M_1 = (Q_1, s_0^1, \Sigma_1, \Gamma_1, \delta_1, \Pi_1, F_1)$ and $M_2 = (Q_2, s_0^2, \Sigma_2, \Gamma_2, \delta_2, \Pi_2, F_2)$
$M_3 = M_1 \circ M_2$ $M_3 = (Q_1 \times Q_2, (s_0^1, s_0^2), \Sigma_1, \Gamma_2, F_1 \times F_2, \Pi_3)$
where $\Pi_3 = \{((p, q), a, b, (p', q')) \mid \exists c \in \Gamma_1 \cap \Sigma_2 : ((p, a, c, p') \in \Pi_1) \wedge ((q, c, b, q') \in \Pi_2)\}$

[2] unless they are acyclic and ε-free transducers
[3] under the assumption that the output alphabet of M_1 is a subset as the input alphabet of M_2

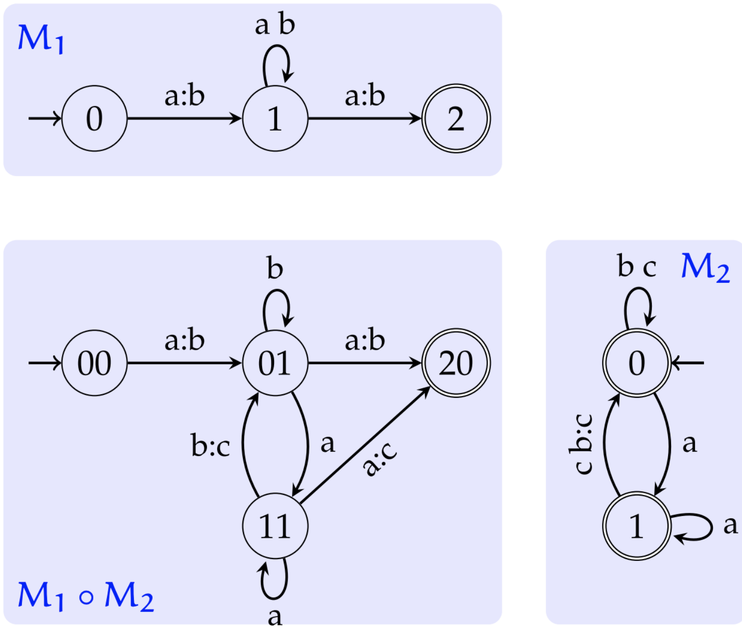


Figure 1: Example of FST composition by `https://dsacl3-2019.github.io/slides/fst.pdf`

Figure 1 illustrates an example of composition of two FSTs.

4.3.2 Inversion

Figure 2 illustrates an example of inversion of an FST.

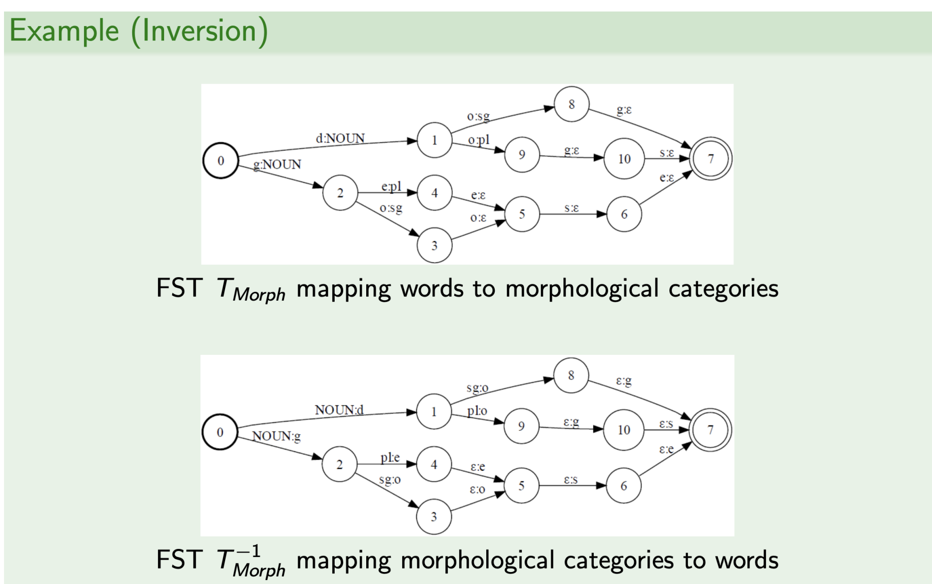


Figure 2: Example of FST inversion by `https://core.ac.uk/download/pdf/24059977.pdf`

4.3.3 Kleene Closure

Figure 3 illustrates an example of Kleene star on an FST.

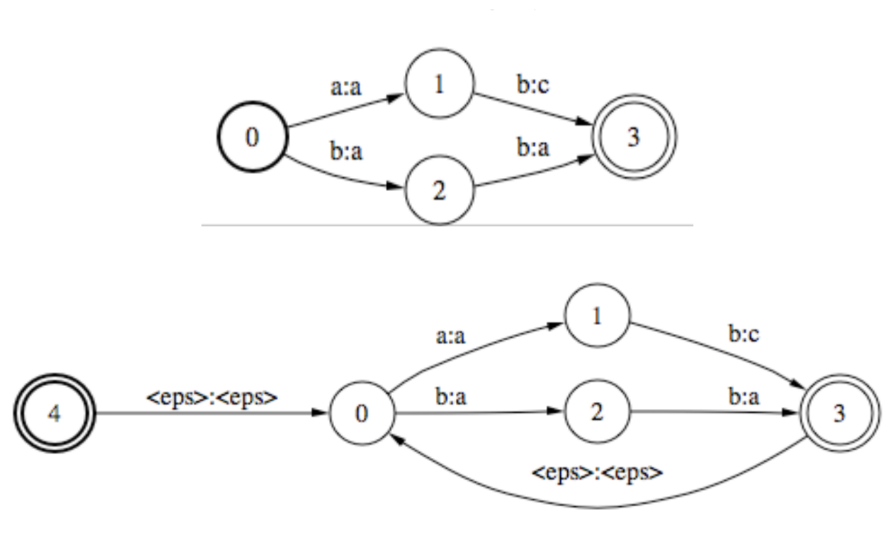


Figure 3: Example of FST Kleene Closure by `https://www.cs.sfu.ca/~anoop/courses/MACM-300-Spring-2006/fst.pdf`

4.3.4 Projection

Figure 4 illustrates an example of a projection on an FST. In projection, we transform an FST to an FSA using either the input alphabet or the output alphabet

Figure 4: Example of FST composition by `https://dsacl3-2019.github.io/slides/fst.pdf`

5 Live Demo

I show examples of Finite State Transducer implemented for Arabic from [1].

References

[1] Maha Alkhairy, Afshan Jafri, and Adam Cooper. An integrated, bidirectional pronunciation, morphology, and diacritics finite-state system. *International Journal of Advanced Computer Science and Applications*, 14(10), 2023.

[2] Evan L Antworth. Pc-kimmo: a two-level processor for morphological analysis. *Summer Institute of Linguistics*, 1990.

[3] Steven Bird and T Mark Ellison. One-level phonology: Autosegmental representations and rules as finite automata. *Computational Linguistics*, 20(1):55–90, 1994.

[4] Alonzo Church. Edward f. moore. gedanken-experiments on sequential machines. automata studies, edited by c. e. shannon and j. mccarthy, annals of mathematics studies no. 34, lithoprinted, princeton university press, princeton1956, pp. 129–153. *Journal of Symbolic Logic*, 23 (1):60–60, 1958. doi: 10.2307/2964500.

[5] Daiga Deksne. Finite state morphology tool for latvian. In *Proceedings of the 11th International Conference on Finite State Methods and Natural Language Processing*, pages 49–53, 2013.

[6] Elaine Uí Dhonnchadha. A two-level morphological analyser and generator for irish using finite-state transducers. In *LREC*. Citeseer, 2002.

[7] Geeks for Geeks. Mealy and moore machines in toc. `https://www.geeksforgeeks.org/mealy-and-moore-machines-in-toc/`, 2022. Accessed: 2024-05-05.

[8] Daniel Gildea and Dan Jurafsky. Automatic induction of finite state transducers for simple phonological rules. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 9–15, 1995.

[9] Arthur Gill. Comparison of finite-state models. *IRE Transactions on Circuit Theory*, 7(2): 178–179, 1960.

[10] Bakyt M Kairakbay and David L Zaurbekov. Finite state approach to the kazakh nominal paradigm. In *Proceedings of the 11th International Conference on Finite State Methods and Natural Language Processing*, pages 108–112, 2013.

[11] Laura Kataja and Kimmo Koskenniemi. Finite-state description of semitic morphology: A case study of ancient accadian. In *Coling Budapest 1988 Volume 1: International Conference on Computational Linguistics*, 1988.

[12] George Kiraz. Multi-tape two-level morphology: a case study in semitic non-linear morphology. *arXiv preprint cmp-lg/9407023*, 1994.

[13] George Anton Kiraz. Multitiered nonlinear morphology using multitape finite automata: a case study on syriac and arabic. *Computational Linguistics*, 26:77–105, 2000.

[14] András Kornai. Formal phonology. 2018.

[15] Kimmo Koskenniemi. *Two-level morphology.* PhD thesis, Ph. D. thesis, University of Helsinki, 1983.

[16] Vipul Mittal. Automatic sanskrit segmentizer using finite state transducers. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 85–90, 2010.

[17] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311, 1997.

[18] Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. *Springer Handbook of Speech Processing*, pages 559–584, 2008.

[19] I Trancoso, D Caseiro, C Viana, F Silva, and I Mascarenhas. Pronunciation modeling using finite state transducers. In *Proc. 15th International Congress of Phonetic Sciences (ICPhS'2003)*, 2003.

[20] Wikipedia contributors. Mealy machine. `https://en.wikipedia.org/wiki/Mealy_machine`, 2024. Accessed: 2024-05-05.

[21] Wikipedia contributors. Moore machine. `https://en.wikipedia.org/wiki/Moore_machine`, 2024. Accessed: 2024-05-05.