

# Finite state transducer (FST)

LING 570

Fei Xia

# Applications of FSTs

- ASR
- Tokenization
- Stemmer
- Text normalization
- Parsing
- ...

# Outline

- Regular relation
- Finite-state transducer (FST)

# Regular relation

# Definition of regular relation

- The set of regular relations is defined as follows:
  - For all  $(x, y) \in \Sigma_1 \times \Sigma_2$ ,  $\{(x, y)\}$  is a regular relation
  - The empty set is a regular relation
  - If  $R_1, R_2$  are regular relations, so are  
 $R_1 \cdot R_2 = \{(x_1 x_2, y_1 y_2) \mid (x_1, y_1) \in R_1, (x_2, y_2) \in R_2\}$ ,  
 $R_1 \cup R_2$ , and  $R^*$ .
  - Nothing else is a regular relation.

# Closure properties

- Like regular languages, regular relations are closed under
  - union
  - concatenation
  - Kleene closure
- Unlike regular languages, regular relations are NOT closed under
  - Intersection:  $R1=\{(a^n b^*, c^n)\}$ ,  $R2=\{(a^* b^n, c^n)\}$ ,  
the intersection is  $\{(a^n b^n, c^n)\}$  and it is not regular
  - difference:
  - complementation

# Closure properties (cont)

- New operations for regular relations:
  - Composition:  $\{ (x,z) \mid \exists y, (x,y) \in R_1 \text{ and } (y,z) \in R_2 \}$
  - Projection:  $\{ x \mid \exists y, (x,y) \in R \}$
  - Inversion:  $\{ (y,x) \mid (x,y) \in R \}$
  - Take a regular language and create the identity regular relation:  $\{ (x,x) \mid x \in L \}$
  - Take two regular languages and create the cross product relation:  $\{ (x,y) \mid x \in L_1, y \in L_2 \}$

- A question about the notation: If you have something like  $(x, yz)$ , where it seems like there's nothing in the input that corresponds to the output  $z$ , is there an implicit  $*e*$  in the input?
- In other words  $(x*e*, yz)$ ?

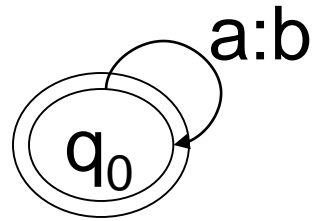


# Finite state transducer

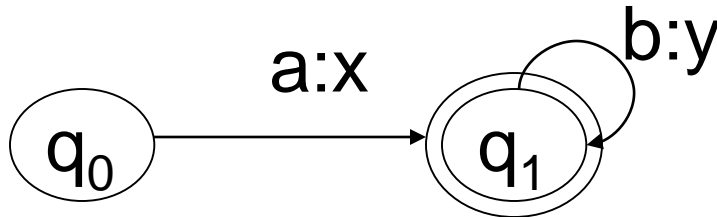
# Finite-state transducers

- $x:y$  is a notation for a mapping between two alphabets:  $x \in \Sigma_1, y \in \Sigma_2$
- An FST processes an input string, and outputs another string as the output.
- Finite-state automata equate to regular languages, and FSTs equate to **regular relations**.
  - Ex:  $R = \{ (a^n, b^n) \mid n \geq 0 \}$  is a regular relation.  
It maps a string of a's into an equal length string of b's.

# FST examples



$$R(T) = \{ (\epsilon, \epsilon), (a, b), (aa, bb), \dots \}$$



$$R(T) = \{ (a, x), (ab, xy), (abb, xyy), \dots \}$$

# Definition of FST

A FST is  $(Q, \Sigma, \Gamma, I, F, \delta)$

- $Q$ : a finite set of states
- $\Sigma$ : a finite set of input symbols
- **$\Gamma$ : a finite set of output symbols**
- $I$ : the set of initial states
- $F$ : the set of final states
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$ : **the transition relation between states.**

→ FSA can be seen as a special case of FST.

# Definition of transduction

- The extended transition relation  $\delta^*$  is the smallest set such that

$$\delta \subseteq \delta^*$$

$$(q, x, y, r) \in \delta^* \wedge (r, a, b, s) \in \delta \Rightarrow (q, xa, yb, s) \in \delta^*$$

- T transduces a string x into a string y if there exists a path from an initial state to a final state whose input is x and whose output is y:

$$x[T]y \quad (a.k.a. \quad (x, y) \in R(T))$$

$$\text{iff} \quad \exists q \in I \exists f \in F \text{ s.t. } (q, x, y, f) \in \delta^*$$

# More FST examples

- Case folding:  
“Go away” → “go away”



- Tokenization:  
he said: "Go away." → he said : “ Go away . ”

- Morphological analysis:  
cats → cat s
- POS tagging:  
He called Mary → PN V N
- Map Arabic numbers to words  
123 → one hundred and twenty three

# Operations on FSTs

- Union:

$(x, y) \in R(T_1 \cup T_2)$  iff  $(x, y) \in R(T_1)$  or  $(x, y) \in R(T_2)$

- Concatenation:

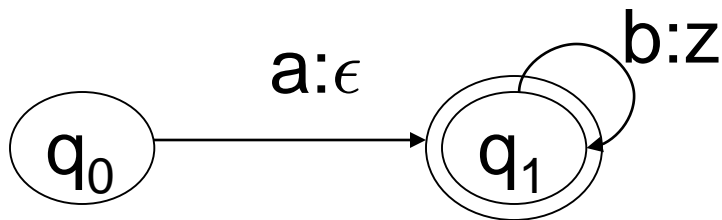
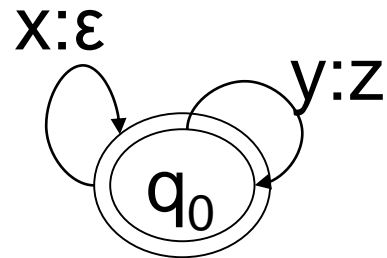
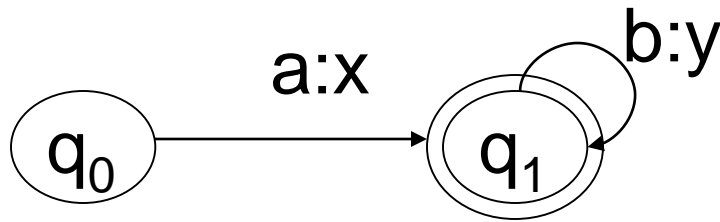
$(wx, yz) \in R(T_1 \bullet T_2)$  iff  $(w, y) \in R(T_1)$  and  $(x, z) \in R(T_2)$

- Composition:

$(x, z) \in R(T_1 \circ T_2)$  iff  $\exists y$  s.t.  $(x, y) \in R(T_1)$  and  $(y, z) \in R(T_2)$



# An example of composition operation



# FST Algorithms

- **Recognition:** Is a given pair of strings accepted by an FST?
  - $(x,y) \rightarrow \text{yes/no}$
- **Composition:** Given two FSTs  $T_1$  and  $T_2$  defining regular relations  $R_1$  and  $R_2$ , create the FST that computes the composition of  $R_1$  and  $R_2$ .
  - $R_1=\{(x,y)\}, R_2=\{(y,z)\} \rightarrow \{(x,z) \mid (x,y) \in R_1, (y,z) \in R_2\}$
- **Transduction:** given an input string and an FST, provide the output as defined by the regular relation?
  - $x \rightarrow y$

# Weighted FSTs

A FST is  $(Q, \Sigma, \Gamma, I, F, \delta, P)$

- $Q$ : a finite set of states
- $\Sigma$ : a finite set of input symbols
- $\Gamma$ : a finite set of output symbols
- $I: Q \rightarrow \mathbb{R}^+$  (initial-state **probabilities**)
- $F: Q \rightarrow \mathbb{R}^+$  (final-state **probabilities**)
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$ : the transition relation between states.
- **$P: \delta \rightarrow \mathbb{R}^+$  (transition probabilities)**

# Summary

- Finite state transducers specify regular relations
- FST closure properties: union, concatenation, composition
- FST special operations:
  - creating regular relations from regular languages (Id, crossproduct);
  - creating regular languages from regular relations (projection)
- FST algorithms
  - Recognition
  - Transduction
  - Composition
  - ...
- Not all FSTs can be determinized.
- Weighted FSTs are used often in NLP.