Finite state transducer (FST)

LING 570

Fei Xia

Week 3: 10/12/2009

Applications of FSTs

- ASR
- Tokenization
- Stemmer
- Text normalization
- Parsing
- …

Outline

- Regular relation
- Finite-state transducer (FST)
- Hw3

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
    $R_1 \cdot R_2 = \{(x_1 x_2, y_1 y_2) \mid (x_1, y_1) \in R_1, (x_2, y_2) \in R_2\}$, $R_1 \bigcup R_2$, 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 <u>NOT</u> 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)

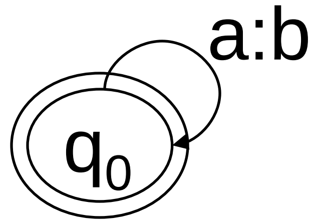
- New operations for regular relations:
  - Composition: $\{(x,z) \mid \exists\, y, (x,y) \in R_1 \text{ and } (y,z) \in R_2\}$
  - Projection: $\{x \mid \exists\, y, (x,y) \in R\}$
  - Inversion: $\{(y,x) \mid (x,y) \in R\}$
  - Take a regular language and create the identity regular relation: $\{(x,x)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

Finite state transducer

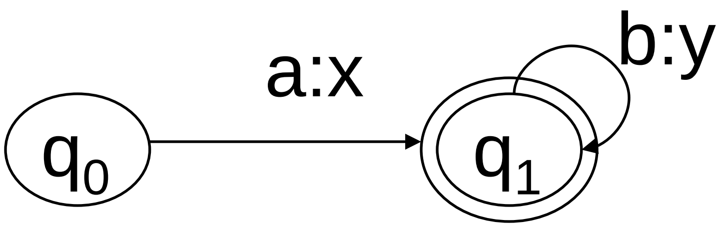
Finite-state transducers

- 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) | n >= 0} is a regular relation. It maps a string of a's into an equal length string of b's

FST examples



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



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

Definition of FST

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

- Q: a finite set of states
- Σ: a finite set of input symbols
- **Γ: 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 δ^* is the smallest set such that

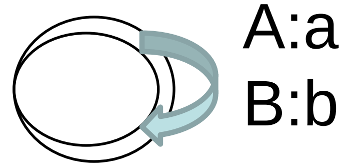
$$\delta \subseteq \delta^*$$
$$(q,x,y,r) \in \delta^* \wedge (r,a,b,s) \in \delta \Rightarrow (q,xa,yb,s) \in \delta^*$$

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

$$x[T]y \quad (a.k.a. \quad (x,y) \in R(T))$$
$$iff \quad \exists q \in I\ \exists f \in F\ s.t.\ (q,x,y,f) \in \delta^*$$

More FST examples

- Lowercase a string of any length
  “Go away” ➔ “go away”



- Tokenize a string
  he said: ”Go away.” ➔ he said : “ Go away . ”

- Convert a word to its morpheme sequence
  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:

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

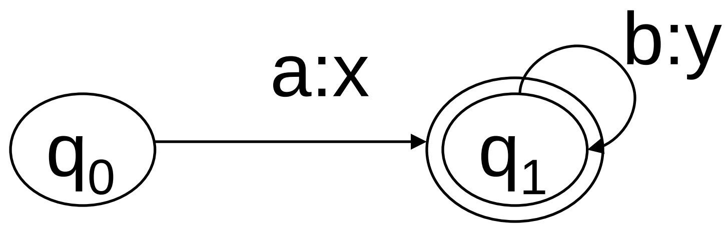
- Concatenation:

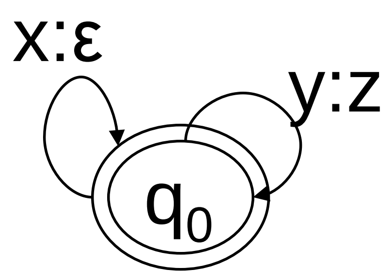
$$(wx, yz) \in R(T_1 \bullet T_2) \; iff \;\; (w, y) \in R(T_1) \; and \; (x, z) \in R(T_2)$$

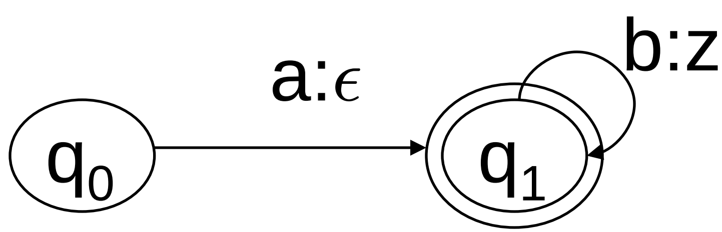
- Composition:

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

An example of composition operation







FST Algorithms

- **Recognition**: Is a given pair of strings accepted by an FST?
  - (x,y) ➔ 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.
  - R1={(x,y)}, R2={(y,z)} ➔ $\{(x,z) \mid (x,y) \in R_1, (y,z) \in R_2\}$

- **Transduction**: given an input string and an FST, provide the output as defined by the regular relation?
  - x ➔ y

Weighted FSTs

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

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

An example: build a unigram tagger

$P(t_1 \ldots t_n \mid w_1 \ldots w_n)$
$\approx P(t_1|w_1) * \ldots * P(t_n \mid w_n)$

Training time: Collect (word, tag) counts, and store P(t | w) in an FST.

Test time: in order to choose the best tag sequence,

1. create an FSA for the input sentence
2. compose it with the FST.
3. choose the best path in the new FST

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.

Hw3

Task: Creating a unigram POS tagger using FSTs

- POS tagger:
  - Input: w1 w2 … w_n
  - Output: w1/t1 w2/t2 … w_n/t_n

- Training data: w1/t1 w2/t2 … w_n/t_n
- Test data: w1 w2/ … w_n