Computational Morphology: Finate State Methods

Yulia Zinova

09 April 2014 – 16 July 2014

Finite state approach

- Finite state approach to morphology is by far the most popular one;
- References: Johnson (1972); Kaplan and Kay (1994); Karttunen (2003)
- Two-level morphology: Koskenniemi (1984)

What is a language?

- A *language* is a set of expressions that are built from a set of symbols from an *alphabet*.
- An *alphabet* is a set of letters (or other symbols from a writing system), phones, or words.
- *Regular language* is a language that can be constructed out of a finite alphabet (denoted Σ) using ore or more of the following operations:
  - set union $\cup$
    $\{a, b, c\} \cup \{c, d\} = \{a, b, c, d\}$
  - concatenation $\cdot$
    $abc \cdot cd = abccd$
  - transitive closure *
    $a*$ denotes the set of sequences consisting of 0 or more a's

Regular language

- Any finite set of strings from a finite alphabet is a regular language.
- Regular languages can be used to describe a large number of phenomena in natural language.
- There are morphological constructions that cannot be described by regular languages: phrasal reduplication in Bambara, a language of West Africa (Culy, 1985).

Bambara example

(1) a. wulu o wulu
dog MARKER dog
'whichever dog'

b. wulunuinina o wulunuinina
dog searcher MARKER dog searcher
'whichever dog searcher'

c. manolunyininafilèla o
rice searcher watcher MARKER
manolunyininafilèla
rice searcher watcher
'whichever rice searcher watcher'

Bambara example

- Phrasal reduplication: X-o-X pattern.
- Why is this a problem for a regular language?

Bambara example

- Phrasal reduplication: X-o-X pattern.
- Why is this a problem for a regular language?
- Because the nominal phrase is in principle unbounded, so the construction involves unbounded copying.
- Unbounded copying can be described neither by regular nor by contex-free languages.

Regular languages

- Σ^* – universal language; consists of all strings that can be constructed out of the alphabet Σ;
- ϵ – the empty string; Σ^* contains ϵ;
- $\emptyset$ – consists of no strings;
- **Question:**
  Does $\emptyset$ include ϵ?

Regular languages

- Σ^* – universal language; consists of all strings that can be constructed out of the alphabet Σ;
- ϵ – the empty string; Σ^* contains ϵ;
- $\emptyset$ – consists of no strings;
- **Question:**
  Does $\emptyset$ include ϵ?
- **Answer:**
  No: ϵ is a string and $\emptyset$ contains **no** strings.

Regular languages: more operations

- Regular languages are also closed under the following operations:
  - intersection $\cap$
    $\{a, b, c\} \cap \{c, d\} = \{c\}$
  - difference $-$
    $\{a, b, c\} - \{c, d\} = \{a, b\}$
  - complementation $\overline{X}$
    $\overline{A} = \Sigma^* - A$
  - string reversal X^R
    $(abc)^R = cba$

Regular languages: regular expressions

- Regular languages are commonly denoted via *regular expressions*.
- Regular expressions involve a set of reserved symbols as notation:
  - *: zero or more;
  - ?: zero or one;
  - $+$: one or more;
  - $|$ or $\cup$: disjunction
  - $\neg$: negation
- **Question:**
  Which language is denoted by
  - $(abc)?$

Regular languages: regular expressions

- Regular languages are commonly denoted via *regular expressions*.
- Regular expressions involve a set of reserved symbols as notation:
  - *: zero or more;
  - ?: zero or one;
  - $+$: one or more;
  - $|$ or $\cup$: disjunction
  - $\neg$: negation
- **Question:**
  Which language is denoted by
  - (abc)? **Answer:** $\{\epsilon, abc\}$

Regular languages: regular expressions

- Regular languages are commonly denoted via *regular expressions*.
- Regular expressions involve a set of reserved symbols as notation:
  - *: zero or more;
  - ?: zero or one;
  - $+$: one or more;
  - $|$ or $\cup$: disjunction
  - $\neg$: negation
- **Question:**
  Which language is denoted by
  - $(abc)?$ **Answer:** $\{\epsilon, abc\}$
  - $(a|b)$

Regular languages: regular expressions

- Regular languages are commonly denoted via *regular expressions*.
- Regular expressions involve a set of reserved symbols as notation:
  - *: zero or more;
  - ?: zero or one;
  - $+$: one or more;
  - $|$ or $\cup$: disjunction
  - $\neg$: negation
- **Question:**
  Which language is denoted by
  - $(abc)?$ **Answer:** $\{\epsilon, abc\}$
  - $(a|b)$ **Answer:** $\{a, b\}$

Regular languages: regular expressions

- Regular languages are commonly denoted via *regular expressions*.
- Regular expressions involve a set of reserved symbols as notation:
  - *: zero or more;
  - ?: zero or one;
  - $+$: one or more;
  - $|$ or $\cup$: disjunction
  - $\neg$: negation
- **Question:**
  Which language is denoted by
  - $(abc)?$ **Answer:** $\{\epsilon, abc\}$
  - $(a|b)$ **Answer:** $\{a, b\}$
  - $(\neg a)^*$

Regular languages: regular expressions

- Regular languages are commonly denoted via *regular expressions*.
- Regular expressions involve a set of reserved symbols as notation:
  - *: zero or more;
  - ?: zero or one;
  - $+$: one or more;
  - $|$ or $\cup$: disjunction
  - $\neg$: negation
- **Question:**
  Which language is denoted by
  - $(abc)?$ **Answer:** $\{\epsilon, abc\}$
  - $(a|b)$ **Answer:** $\{a, b\}$
  - $(\neg a)^*$ **Answer:** the set of strings with zero or more occurences of anything rather than a

Exercise

- Find regular expressions over $\{0, 1\}$ that determine the following languages:
    1. all strings that contain an even number of 1's;
    2. all strings that contain an odd number of 0's.

Finite state automaton

- *Finite-state automata* are computational devices that compute regular languages.
- A finite-state automaton is a quintuple $M = (Q, s, F, \Sigma, \delta)$ where:
  1. Q is a finite set of states;
  2. s is a designated initial state;
  3. F is a designated set of final states;
  4. Σ is an alphabet of symbols;
  5. δ is a transition relation from $Q \times (\Sigma \cup \epsilon)$ to Q (from state/symbol pairs to states).
- $A \times B$ denotes the cross-product of sets A and B
  $\{a, b\} \times \{c, d\} = \{< a, c >, < b, c >, < a, d >, < b, d >\}$

FSA: Kleene's theorem

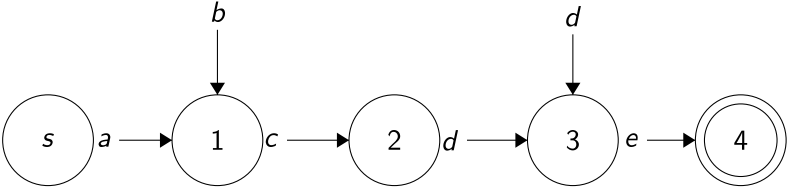
- Kleene's theorem states that every regular language can be recognized by a finite-state automaton.
- Similarly, every finite state automaton recognizes a regular language.

FSA: simple example

- **Task:** draw an automaton that accepts the language ab^*cd^+e

FSA: simple example

- **Task:** draw an automaton that accepts the language ab^*cd^+e



Regular relations

- Regular relations express relations between sets of strings.
- A regular n-relation is defined as follows:
  1. $\emptyset$ is a regular n-relation;
  2. For all symbols $a \in [(\Sigma \cup \epsilon) \times \ldots \times (\Sigma \cup \epsilon)]$, $\{a\}$ is a regular n-relation;
  3. If R_1, R_2, and R are regular n-relations, then so are
     - 3.1 $R_1 \cdot R_2$, the *n-way concatenation* of R_1 and R_2: for every $r_1 \in R_1 and r_2 \in R_2, r_1 r_2 \in R_1 \cdot R_2$
     - 3.2 $R_1 \cup R_2$
     - 3.3 R^*, the *n-way transitive (Kleene) closure* of R.
- For most applications in speech and language processing $n = 2$.

Finite state transducer

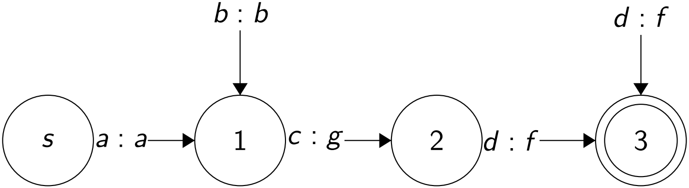
- A 2-way finite-state transducer is a quintuple $M = (Q, s, F, \Sigma \times \Sigma, \delta)$ where:
    1. Q is a finite set of states;
    2. s is a designated initial state;
    3. F is a designated set of final states;
    4. Σ is an alphabet of symbols;
    5. δ is a transition relation from $Q \times (\Sigma \cup \epsilon \times \Sigma \cup \epsilon)$ to Q.

FST example

- With a transducer, a string matches against the input symbols on the arcs, while at the same time the machine is outputting the corresponding output symbols.
- **Task:** draw a FST that computes the relation $(a:a)(b:b)^*(c:g)(d:f)^+$

FST example

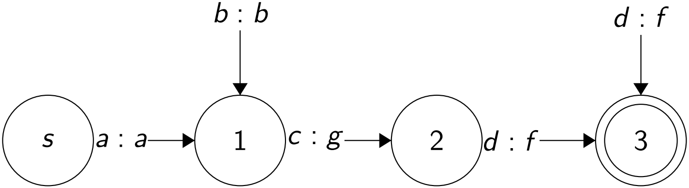
- With a transducer, a string matches against the input symbols on the arcs, while at the same time the machine is outputting the corresponding output symbols.
- **Task:** draw a FST that computes the relation $(a:a)(b:b)^*(c:g)(d:f)^+$



- **Question:** What will it produce for the string *abbcddd*?

FST example

- With a transducer, a string matches against the input symbols on the arcs, while at the same time the machine is outputting the corresponding output symbols.
- **Task:** draw a FST that computes the relation $(a:a)(b:b)^*(c:g)(d:f)^+$



- **Question:** What will it produce for the string *abbcddd*?
  Answer: *abbgfff* .

Closure properties of regular languages and relations

Property	Languages	Relations
concatenation	yes	yes
Kleene closure	yes	yes
union	yes	yes
intersection	yes	no
difference	yes	no
composition	–	yes
inversion	–	no

- Composition: if f and g are two regular relations and x a string, then $[f \circ g](x) = f(g(x))$
- Inversion: swapping the input and the output symbols on the arcs

Culy, C. (1985). The complexity of the vocabulary of bambara. *Linguistics and Philosophy*, pages 345–351.

Johnson, C. D. (1972). *Formal aspects of phonological description*. Mouton The Hague.

Kaplan, R. M. and Kay, M. (1994). Regular models of phonological rule systems. *Computational linguistics*, **20**(3), 331–378.

Karttunen, L. (2003). Finite-state morphology.

Koskenniemi, K. (1984). A general computational model for word-form recognition and production. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics*, pages 178–181. Association for Computational Linguistics.