Lecture 3: Finite State Technology

CSA3202 Human Language Technology

Mike Rosner, Dept ICS
October 2011
Acknowledgements

- Shuly Wintner (University of Haifa), Lecture Notes, 2008
Outline

1. Computational Morphology
2. Revision of Formal Language Theory
3. Regular Expressions
Morphology versus Computational Morphology

- Morphology studies *relations* between two domains:
  - word forms enlargement
  - morphemes en + large + ment

- **Computational Morphology** is the design of algorithms which perform computations over that relation.
  - Analysis
  - Synthesis

- Computational Morphology is not just about strings, but is also about meanings.
Challenges for Computational Morphology

- Handling Segmentation: what are the parts into which the word is broken.
- Handling Morphotactics:
  - handling the order in which the parts combine together
  - computing the result
- Handling Phonological Alternations
  - pity is realized as piti in pitilessness
  - die becomes dy in dying
- Handling the lexicon: how to represent it.
Finite-state automata are a good model for representing the lexicon.

Also good for representing dictionaries (lexicons + additional information)

They are also for describing morphological processes that involve concatenation etc.

A natural extension of finite-state automata - finite-state transducers - are a perfect model for most processes known in morphology and phonology including non-segmental ones.
Intuition: A formal language is a set of strings over an alphabet.

Definition of alphabet and string
- An alphabet (often denoted by $\Sigma$) is a finite set of symbols called letters.

Definition of String
- A string is a finite sequence of letters.
Operations on Strings

- **String Length** \( |w| \)
- **Concatenation** \( w_1 . w_2 \)
- **Exponent** \( w^n = w_1 \ldots w_{n-1} . w_n \)
- **Reversal** \( w^R \). If \( w = \langle w_1, w_2 \ldots w_n \rangle \) then
  \[ w^R = \langle w_n, w_{n-1} \ldots w_1 \rangle \]
- **Substring**: If \( w = \langle x_1 \ldots x_n \rangle \) then for any \( i, j \) such that
  \[ 1 \leq i \leq j \leq n \]
  \[ \langle x_i \ldots x_j \rangle \] is a substring of \( w \).
- Two special cases of substring are *prefix* and *suffix*.
- If \( w = w_l . w_c . w_r \) then
  - \( w_l \) is a prefix of \( w \) and
  - \( w_r \) is a suffix of \( w \)
Prefix and Suffix

- Let $\Sigma = a, b, c, \ldots y, z$ be an alphabet.
- Let $w = \text{indistinguishable}$ be a string over $\Sigma$.
- Then $\epsilon, \text{in, indis, indistinguish} \text{and indistinguishable}$ are prefixes of $w$.
- $\epsilon, \text{e, able, distinguishable and indistinguishable}$ are suffixes of $w$.
- Substrings that are neither prefixes nor suffixes include $\text{distinguish, gui and is.}$
Let $\Sigma = a, b, c, \ldots y, z$ be an alphabet.

Let $w = \text{indistinguishable}$ be a string over $\Sigma$.

Then $\epsilon$, $\text{in}$, $\text{indis}$, $\text{indistinguish}$ and $\text{indistinguishable}$ are prefixes of $w$.

$\epsilon$, $\text{e}$, $\text{able}$, $\text{distinguishable}$ and $\text{indistinguishable}$ are suffixes of $w$.

Substrings that are neither prefixes nor suffixes include $\text{distinguish}$, $\text{gui}$ and $\text{is}$. 
Prefix and Suffix

- Let $\Sigma = a, b, c, \ldots, y, z$ be an alphabet.
- Let $w = \text{indistinguishable}$ be a string over $\Sigma$.
- Then $\epsilon$, in, indis, indistinguish and indistinguishable are prefixes of $w$.
- $\epsilon$, e, able, distinguishable and indistinguishable are suffixes of $w$.
- Substrings that are neither prefixes nor suffixes include distinguish, gui and is.
Given an alphabet $\Sigma$, the set of all strings over $\Sigma$, is denoted by $\Sigma^*$.

A formal language over $\Sigma$ is a subset of $\Sigma^*$.

Examples

Let $\Sigma = a, b, c, \ldots y, z$ be an alphabet.

The following are formal languages over $\sigma$

- $\Sigma^*$
- the set of strings consisting of consonants only;
- the set of strings consisting of vowels only;
- the set of strings each of which contains at least one vowel and at least one consonant;
- the set of palindromes:
  - a man a plan a canal panama
  - as I pee sir I see pisa
Lifting String Operations to Languages

- In general string operations can be lifted to languages. We use the same notation as for strings.
- If \( L \) is a language then the *reversal* of \( L \), denoted \( L^R \), is the language
  \[
  \{ w \mid w^R \in L \}
  \]
- If \( L_1 \) and \( L_2 \) are languages, then the concatenation of \( L_1 \) and \( L_2 \),
  \[
  L_1 . L_2 = \{ w_1 . w_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2 \}
  \]
Examples

- $L_1 = \{\text{re, un}\}$
- $L_2 = \{\text{mark, talk}\}$
- $L_3 = \{\epsilon, s, \text{ed, ing}\}$
- $L_2^R = \{\text{karm, klat}\}$
- $L_1 . L_2 . L_3 = \{\text{remark, remarked, remarking, ... retalking}\}$
Kleene Closure

- The Kleene Closure of L is denoted $L^*$ and defined as
  $$
  \bigcup_{i=0}^{\infty} L^i.
  $$

- Note also that
  $$
  L^+ = \bigcup_{i=1}^{\infty} L^i.
  $$

Example

Let $L = \{\text{dog, cat}\}$.

- $L^0 = \{\epsilon\}$.
- $L^1 = \{\text{dog, cat}\}$,
- $L^2 = \{\text{dogdog, catcat}\}$,
Regular expressions are a formalism for defining (formal) languages.

Their “syntax” is formally defined and is relatively simple.

Their “semantics" is sets of strings

The denotation of a regular expression is a set of strings in some formal language.
Regular Expressions

Syntax

- 0 is an RE
- $\epsilon$ is an RE
- if $a \in \Sigma$ is a letter then $a$ is an RE
- if $r_1$ and $r_2$ are REs, then so are $r_1 + r_2$ and $r_1.r_2$
- if $r$ is an RE then so is $(r)^*$
- nothing else is an RE over $\Sigma$
Regular Expressions

Examples

- Let $\Sigma = a, b, c, \ldots y, z$ be an alphabet. Some REs over $\Sigma$ include
- $\epsilon$
- $\epsilon$
- $((c.a).t)$
- $((m.e).(o))^{*}.w$)
- $(a + (e + (i + (o + u))))$
- $(a + (e + (i + (o + u))))^{*}$
Regular Expressions
Semantics

For every RE $r$ its *denotation* $[r]$ is defined as follows:

- $[0] = 0$
- $[\epsilon] = \{\epsilon\}$
- if $a \in \Sigma$ is a letter then $[a] = a$
- if $r_1$ and $r_2$ are REs whose denotations are $[r_1]$ and $[r_2]$, then
  - $[r_1 + r_2] = [r_1] \cup [r_2]$
  - $[r_1 \cdot r_2] = [r_1] \cdot [r_2]$
  - $[(r_1)^*] = [r_1]$
Example

<table>
<thead>
<tr>
<th>REGULAR EXPRESSIONS</th>
<th>SEMANTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>{a}</td>
</tr>
<tr>
<td>((c.a).t)</td>
<td>{c.a.t}</td>
</tr>
<tr>
<td>(((m.e).(0)*).w)</td>
<td>{mew, meow, meoow, meooow ...}</td>
</tr>
<tr>
<td>(a + (e + (i + (o + u))))</td>
<td>{a, e, i, o, u}</td>
</tr>
<tr>
<td>(a + (e + (i + (o + u))))*</td>
<td>all strings of vowels</td>
</tr>
</tbody>
</table>
Definition

A Language is **regular** if it is the denotation of some regular expression.
Not all formal languages are regular

Closure

A class of languages is said to be **closed** under some operation if and only if whenever two languages are in the class, the result of performing the operation on the two languages is also in this class.
Regular expressions closure properties:

Regular languages are closed under:

- Union
- Intersection
- Complementation
- Difference
- Concatenation
- Kleene-star
Some Things that are Regular Languages

- Zero or more a’s followed by zero or more b’s
- The set of words in an English dictionary
- Dates
- URLs
- English?
Some Things that are not Regular Languages

- Zero or more a’s followed by exactly the same number of b’s
- The set of all English palindromes
- The set that includes all noun phrases of the form
  - the cat slept
  - the cat the dog bit slept
  - the cat the dog the man fed bit slept
Regular Languages
Equivalent Ways of Describing Regular Languages

finite automata

regular languages

regular grammars

regular expressions
Exercise 3.1

Write regular expressions, including definition of alphabet, for

1. The language \{walk, walks, walking, walked\}
2. The same language for the verb “sleep”
3. The same language for the verb “run”
4. The language \{nikteb, tikteb, jikteb, niktbu, tiktbu, jiktbbu\}
Summary

- What is computational morphology?
- Answer in terms of computations over relations between (a) domain of forms and (b) domain of morphemes
- Investigate the domain of forms (words on paper) consider as a formal language
- Revise notion formal language
- Regular expression
- Next - Finite State Automata