Language and Computation

week 2, Tuesday, January 21, 2014

Tamás Biró
Yale University
tamas.biro@yale.edu
http://www.birot.hu/courses/2014-LC/

Practical matters

- **Sections:** consultation with TA Jen Rund.
  Registration administratively necessary, attendance optional.
  Poll: We 5:30-6:30 and Fr 10:30-11:30.
  Begins next week, rooms t.b.a.

- **Programming sections:** 6 people with Tamas Biro.
  We 4:30-6:00, DOW 314 (CLAY lab, 370 Temple St)

- Optional project-based **term paper** in lieu of final exam:
  Undergrads only. Approx. 10-15 pages. Cover more topics, compare methods and include references. Details t.b.a.

Readings

- Pre-reading by Thursday 01/23: JM 3.1-3.6.
  Pre-reading by Tuesday 01/28: JM 4.1-4.3.

- Post-reading after today's lecture: JM 2.

- Python: H1 this week, H2 next week.
  Programming section: install Python on your laptop by We.

- Recommended: `FSA Utilities` by *Gertjan van Noord*
  `http://odur.let.rug.nl/vannoord/Fsa/` (install it)

Regular expressions for NLP

Language *as* computation

- Data structures, a.k.a. representations

- Operations on these representations

- Overall architectures

Language technology as computation

- Data structures, a.k.a. representations:
  typically bytes, characters and strings.

- Operations on these representations:
  for example: regular expressions.

- Overall architectures

Regular expressions (and transducers)

A useful tool to process "text-like data":

- Text written in natural language X.
- Speech transcribed using IPA, ARPAbet, etc.
- Program code
- DNA sequences
- etc.

“Text-like data”

- **Alphabet** Σ: finite set of atomic symbols ($letters$ or $characters$).
- **String**, aka **word** or **sentence** or **text**: a series of n letters ($n \geq 0$).
  Empty string: string composed of $n = 0$ characters.
- **Concatenation** operation $a + b$ of strings a and b:
  letters in a followed by letters in b, joining the two strings end-to-end.
  Concatenation is associative but not commutative:
  $(a + b) + c = a + (b + c)$, but $a + b \neq b + a$.
- NB: Set Σ^* of all finite strings over alphabet Σ is a $monoid$ with concatenation as operation and the empty string as identity element.

Processing “text-like data”

Processing as **generations / recognition / acceptance**:

- Language = set of “sentences”
- Goal: generate this set
- Goal: accept a sentence if and only if (=iff) in this set
- Cf. Chomsky’s I-language vs. E-language

$\longrightarrow$ automata

Processing as **mapping**:

- Map meaning to utterance
- Map utterance to meaning
- Map text to speech
- Map speech to text

$\longrightarrow$ transducers

Formal language

Definition:

A language $\mathcal{L}$ over a finite alphabet Σ is a subset of the set Σ^* of all finite strings over alphabet Σ:

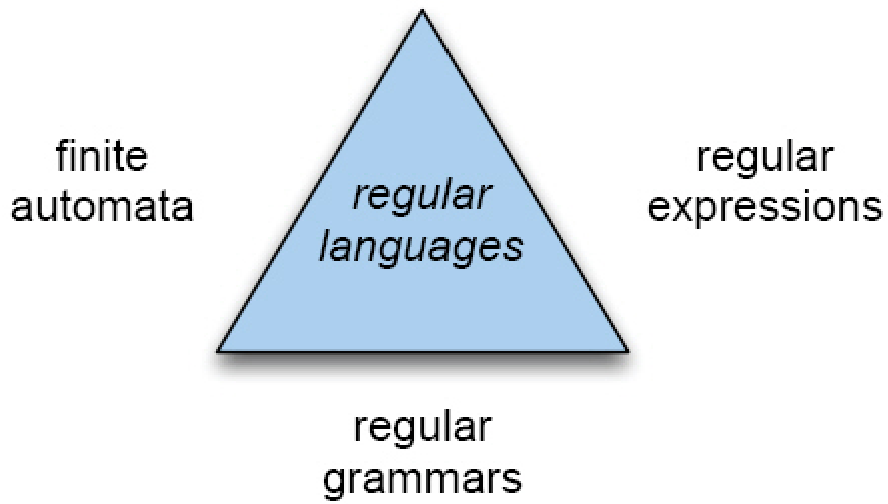
$$\mathcal{L} \subseteq \Sigma^*$$

Our task: find a mechanism that can define such an $\mathcal{L}$.

Answers: regular expressions, automata, formal grammars, etc.

Regular languages

The sets of formal languages accepted / generated by



are the same!

Regular expressions

Pattern matching:

RE	Example Patterns Matched
`/woodchucks/`	"interesting links to <u>woodchucks</u> and lemurs"
`/a/`	"M<u>a</u>ry Ann stopped by Mona's"
`/Claire␣says,/`	" "Dagmar, my gift please," <u>Claire says,</u>"
`/DOROTHY/`	"SURRENDER <u>DOROTHY</u>"
`/!/`	"You've left the burglar behind again<u>!</u>" said Nori

Regular expressions

Pattern matching with sets of characters:

RE	Match	Example Patterns
`/[wW]oodchuck/`	Woodchuck or woodchuck	"Woodchuck"
`/[abc]/`	'a', 'b', *or* 'c'	"In uomini, in soldati"
`/[1234567890]/`	any digit	"plenty of 7 to 5"

Regular expressions

Sets of characters, ranges:

RE	Match	Example Patterns Matched
`/[A-Z]/`	an upper case letter	"we should call it 'Drenched Blossoms' "
`/[a-z]/`	a lower case letter	"my beans were impatient to be hoed!"
`/[0-9]/`	a single digit	"Chapter 1: Down the Rabbit Hole"

Regular expressions

Sets of characters, negation with “caret”:

RE	Match (single characters)	Example Patterns Matched
`[^A-Z]`	not an upper case letter	“Oyfn pripetchik”
`[^Ss]`	neither ‘S’ nor ‘s’	“I have no exquisite reason for’t”
`[^\.]`	not a period	“our resident Djinn”
`[e^]`	either ‘e’ or ‘^’	“look up ^ now”
`a^b`	the pattern ‘`a^b`’	“look up a^ b now”

Regular expressions

Optionality:

RE	Match	Example Patterns Matched
`woodchucks?`	woodchuck or woodchucks	"woodchuck"
`colou?r`	color or colour	"colour"

Regular expressions

Any character:

RE	Match	Example Patterns
`/beg.n/`	any character between *beg* and *n*	begin, beg'n, begun

Regular expressions

Operator precedence:

Parenthesis	`( )`
Counters	`* + ? {}`
Sequences and anchors	`the ^my end$`
Disjunction	`|`

Regular expressions

More character sets:

RE	Expansion	Match	Examples
`\d`	`[0-9]`	any digit	Party␣of␣5
`\D`	`[^0-9]`	any non-digit	Blue␣moon
`\w`	`[a-zA-Z0-9_]`	any alphanumeric/underscore	Daiyu
`\W`	`[^\w]`	a non-alphanumeric	!!!!
`\s`	`[␣\r\t\n\f]`	whitespace (space, tab)	
`\S`	`[^\s]`	Non-whitespace	in␣Concord

Regular expressions

Regex operators for counting:

RE	Match
`*`	zero or more occurrences of the previous char or expression
`+`	one or more occurrences of the previous char or expression
`?`	exactly zero or one occurrence of the previous char or expression
`{n}`	*n* occurrences of the previous char or expression
`{n,m}`	from *n* to *m* occurrences of the previous char or expression
`{n,}`	at least *n* occurrences of the previous char or expression

Regular expressions

Escape characters:

RE	Match	Example Patterns Matched
`\*`	an asterisk “*”	“K*A*P*L*A*N”
`\.`	a period “.”	“Dr. Livingston, I presume”
`\?`	a question mark	“Why don’t they come and lend a hand?”
`\n`	a newline	
`\t`	a tab	

REGEX substitution

ELIZA (Weizenbaum, 1966), the “Rogerian psychologist”:

User$_1$:	Men are all alike.
ELIZA$_1$:	IN WHAT WAY
User$_2$:	They’re always bugging us about something or other.
ELIZA$_2$:	CAN YOU THINK OF A SPECIFIC EXAMPLE
User$_3$:	Well, my boyfriend made me come here.
ELIZA$_3$:	YOUR BOYFRIEND MADE YOU COME HERE
User$_4$:	He says I’m depressed much of the time.
ELIZA$_4$:	I AM SORRY TO HEAR YOU ARE DEPRESSED

REGEX substitution

ELIZA (Weizenbaum, 1966), the “Rogerian psychologist”:

```
s/.* YOU ARE (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/
s/.* YOU ARE (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/
s/.* all .*/IN WHAT WAY/
s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/
```

Regular languages

The set of languages accepted by regular expressions (as well as by finite state automata and regular grammars).

Regular languages are closed under the following operations:

- Intersection
- Difference
- Complementation
- Reversal

Regular languages

intersection	if L_1 and L_2 are regular languages, then so is $L_1 \cap L_2$, the language consisting of the set of strings that are in both L_1 and L_2.
difference	if L_1 and L_2 are regular languages, then so is $L_1 - L_2$, the language consisting of the set of strings that are in L_1 but not L_2.
complementation	If L_1 is a regular language, then so is $\Sigma^* - L_1$, the set of all possible strings that aren't in L_1.
reversal	If L_1 is a regular language, then so is L_1^R, the language consisting of the set of reversals of all the strings in L_1.

Regular languages

1. $\emptyset$ is a regular language
2. $\forall a \in \Sigma \cup \epsilon$, $\{a\}$ is a regular language
3. If L_1 and L_2 are regular languages, then so are:
   (a) $L_1 \cdot L_2 = \{xy | x \in L_1, y \in L_2\}$, the **concatenation** of L_1 and L_2
   (b) $L_1 \cup L_2$, the **union** or **disjunction** of L_1 and L_2
   (c) L_1^*, the **Kleene closure** of L_1

Finite state automata

- Q finite set of states
- Σ (input) alphabet
- q_0 start state
- F set of final states $(F \subseteq Q)$
- $\delta(q, i)$ transition function

Thursday:

- More on finite state automata
- Finite state transducers
- Applications to morphology, spell checking, etc.
- Edit distance

See you on Thursday!