Language and Computation

week 2, Thursday, January 23, 2014

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

Practical matters

- Sections
- Slight change in program (moving "automata" earlier)
- Post-reading after next Tuesday: JM 3 and JM 16 (t.b.a.)
- Pre-reading by Thursday 01/30: JM 4.1-4.3, then 5.1-5.3.
- Project-based **term paper** in lieu of final exam (undergrads): approx. 10-15 pp, details on classes v2 and website.

Automata and transducers

Computation

- Map input onto output
  (e.g., meaning to utterance, sound to text)

  We need a mechanism to compute that mapping.

- Define the set of grammatical forms/sentences/utterances:

  We need a mechanism to compute that set, or to compute whether a form/sentence/utterance $\in$ that set.

We focus on "text-like" information!

Defining a formal language

Formal language $\mathcal{L}$ of alphabet Σ defined as $\mathcal{L} \subseteq \Sigma^*$.
How to define an $\mathcal{L}$?

- using prose and/or human intuition
- enumeration
- regular expressions
- automata
- formal grammars

Language classes

Given a mechanism,

and given an alphabet Σ

what is the class of languages (a subset of $\mathcal{P}(\Sigma^*)$)

that can be defined using that mechanism?

Regular languages

Regular languages over Σ: language that can be defined using regular expressions / regular grammars / finite-state automata.

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

Can we prove these facts using regular expressions?

Automata and transducers

Automaton:

- Input: string $\in \Sigma^*$
- Output: accept or reject.

Transducer:

- Input: string $\in \Sigma^*$
- Output: string $\in \Delta^*$

Finite state automaton

- Q finite set of states
- Σ (input) alphabet
- $q_0 \in Q$ start state
- $F \subseteq Q$ set of final states (can be empty)
- $\delta(q, i)$ transition function $Q \times \Sigma \cup \{\epsilon\} \rightarrow Q$

Finite state automaton

First approximation:

Automaton accepts string $i_1 i_2 \ldots i_n$ iff

there is a series of states $q_1, q_2, \ldots q_n$ such that

$\delta(q_j, i_j) = q_{j+1}$ for all j.

Finite state transducers

- Q finite set of states
- Σ input alphabet and Δ output alphabet
- $q_0 \in Q$ start state
- $F \subseteq Q$ set of final states (can be empty)
- $\delta(q, i)$ transition function $Q \times \Sigma \cup \{\epsilon\} \rightarrow Q$
- $\sigma(q, i)$ output function $Q \times \Sigma \cup \{\epsilon\} \rightarrow \Delta \cup \{\epsilon\}$

See you next week!