Regular Languages

October 31, 2011
CS 81: Computability & Logic
Recall

1. Computability questions reduce to decision problems.

2. Every decision problem can be rephrased as a question about membership in a language.
Determining What’s Computable

The Plan:

✓ Define a class of abstract “machines” that accept or reject strings $w$

✓ See what languages this class of machines can recognize (i.e., what decision problems it can solve).

Note:

✓ Every machine corresponds to a language (its accepted strings)
✓ There may be many different machines accepting the same language
✓ It’s unlikely that every language has a corresponding machine. (Why?)
One Possible Class of Machines

“When the term ‘machine’ is used in ordinary discourse, it tends to evoke an unattractive picture. It brings to mind a big, heavy, complicated object which is noisy, greasy, and metallic; performs jerky repetitive, and monotonous motions; and has sharp edges that may hurt one if he does not maintain sufficient distance…”

Marvin Minsky, *Computation: Finite and Infinite Machines*

But, can we generalize from this?
**Our First Class of Machines: State Machines**

Mathematically, a state machine consists of:

1. an alphabet $\Sigma$
2. a collection of states $K$
3. a transition relation $\rightarrow \subseteq K \times (\Sigma \cup \{\varepsilon\}) \times K$ (where $q \xrightarrow{\sigma} q'$ means that $(q, \sigma, q')$ is in the relation)
4. one initial/starting state $s \in K$
5. a set of final/accepting states $A \subseteq K$

**Finite State Machine:**
$K$ is finite

**Deterministic State Machine:**
a transition function $\delta : K \times \Sigma \rightarrow K$. 
Machine Behavior

✓ The machine starts in state $q_0$.
✓ It can change from state $q$ to state $q'$ on input $\sigma$ provided that $q \xrightarrow{\sigma} q'$.
✓ It can change from state $q$ to state $q'$ spontaneously provided that $q \xrightarrow{\varepsilon} q'$.
✓ The machine accepts a string $w \in \Sigma^*$ if there is at least one path spelling out $w$, that starts at $q_0$ and ends at a state in $F$.
What's Accepted?
WHAT’S ACCEPTED?
**What's Accepted?**

(  bb?  b?  aaab?  ba?  )
Finite State Machines

We care mostly about finite state machines, also known as “Finite Automata”

Terminology:

✓ DFA = Deterministic Finite Automaton = Deterministic FSM = DFSM
✓ NFA = Nondeterministic Finite Automaton = Nondeterministic FSM = NDFSM
The following are equivalent:

✓ There is a DFA accepting the language $L$
✓ [Rabin and Scott] There is an NFA accepting $L$
✓ [Kleene] $L$ is a regular set.
Can a DFA predict this machine’s behavior?
Digression: “Scotland Yard,” the game
FROM NFA TO DFA: THE **Subset Construction**
**Regular Languages**

An *inductively-defined collection of sets*!

- ✓ ∅ is a regular language.
- ✓ \{a\} is regular for any \(a \in \Sigma\).
- ✓ If \(L\) and \(M\) are regular, then so is \(LM\) and \(L \cup M\).
- ✓ If \(L\) is regular, then so is \(L^*\).

True or False?

1. \(\Sigma^*\) is regular.
2. \(\{\epsilon\}\) is regular.
3. If \(w \in \Sigma^*\), then \(\{w\}\) is regular.
4. Every finite language is regular.
5. Every set is regular (since \(\{w_1, w_2, \ldots\} = \{w_1\} \cup \{w_2\} \cup \cdots\).
Regular Expressions

An inductively-defined collection of expressions!

✓ ∅ is a regexp
✓ ε is a regexp
✓ a is a regexp for any $a \in \Sigma$.
✓ If $r_1$ and $r_2$ are regexps, then so is $(r_1 r_2)$ and $(r_1 | r_2)$.
✓ If $r$ is a regexp, then so is $(r^*)$.

Parenthesis Convention:

$$ab^*|c^* = (a(b^*)) | (c^*)$$
**Regexp Interpretations**

Regular expressions abbreviate regular languages.

- ✓ $L(\emptyset) = \emptyset$
- ✓ $L(\varepsilon) = \{\varepsilon\}$
- ✓ $L(a) = \{a\}$
- ✓ $L(r_1 r_2) = L(r_1) L(r_2)$
- ✓ $L(r_1 | r_2) = L(r_1) \cup L(r_2)$
- ✓ $L(r^*) = L(r)^*$

We say that “$r$ matches $w$” if $w \in L(r)$. True or False?

- ✓ $L(r_1) = L(r_2) \rightarrow r_1 = r_2$
- ✓ There is a regular expression $r$ with $L(r) = \Sigma^*$
Regular Expression Examples ($\Sigma = \{0, 1\}$)

Describe the Language

1. $0 \mid 1$
2. $(0|1)^*$
3. $(0|1)\ 0^*\ 1^*$
4. $0^*110^*|1^*001^*$

Find the regular expression

1. Strings where every 1 is followed by a 0.
2. Strings where no 1 is followed by a 0.
3. Strings where every 1 is preceded by and followed by 0.