Jewels of Computer Science

CS 81: Computability and Logic

October 14, 2010
Languages vs. State Machines (Recognizers)

- Every state machine specifies a language
- For every language, there are state machines that specify it.
- For every language, there is a minimal state machine

What languages have finite state 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...
Abstract Machines

✓ Abstraction of computation.
✓ Simplest form: Recognizers

abbaabb... Black Box

(Light goes on or off after each symbol)
Language of a Machine

“w is accepted”: the light is on when w is fully entered
“w is rejected”: the light is off when w is fully entered

Every machine corresponds to a language (accepted strings)
Mathematically, a state machine consists of:

1. an alphabet $\Sigma$
2. a collection of states $Q$;
3. a transition relation $\to \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$
   where $q \xrightarrow{\sigma} q'$ means that $(q, \sigma, q')$ is in the relation;
4. one initial state $q_0 \in Q$;
5. a set of final/accepting states $F \subseteq Q$.

finite: $Q$ is finite
deterministic: transition function $\delta: Q \times \Sigma \to Q$
Finite State Machines also called Finite Automata

Hence, you may see references to

NFSM vs. DFSM

NFA vs. DFA
Machine Behavior

A machine starts in state $q_0$.

From a current state $q$ it can change state to $q'$ with input $\sigma$ provided that $q \xrightarrow{\sigma} q'$

From a current state $q$ it can change state to $q'$ spontaneously provided that $q \xrightarrow{\varepsilon} q'$

The machine accepts a string $w \in \Sigma^*$ provided there is some path, spelling out $w$, that starts at $q_0$ and ends at some $q \in F$. 
What’s Accepted?
Regular Languages

The following are equivalent:

✓ There is an NFA accepting the language L
✓ [Rabin and Scott]: There is a DFA accepting L
✓ [Kleene] L is regular.
✓ [Myhill-Nerode] \( \{ L_w \mid w \in \Sigma^* \} \) is finite.
Digression:
“Scotland Yard” (the game)
From NFA to DFA: the **Subset Construction**
Regular Languages
(an inductively defined collection of sets!)

✓ $\emptyset$ is regular

✓ $\{a\}$ is regular for every $a \in \Sigma$.

✓ If $L$ is a regular language, then so is $L^*$

✓ If $L$ and $M$ are regular, then so are $LM$ and $L \cup M$
Regular Expressions
(an inductive definition!)

- ∅ is a regexp
- a is a regexp for every a ∈ Σ
- If r is a regexp, so is (r*)
- If r₁ and r₂ are regexps, so is (r₁r₂) and (r₁|r₂)

Convention:  ab*|c* = (a(b*)) | (c*)
Regular Expressions

Regular expressions abbreviate regular languages

- \( L(\emptyset) = \emptyset \)
- \( L(\varepsilon) = \{ \varepsilon \} \)
- \( L(a) = \{ a \} \)
- \( L(r^*) = (L(r))^* \)
- \( L(r_1 r_2) = L(r_1) \cdot L(r_2) \)
- \( L(r_1 | r_2) = L(r_1) \lor L(r_2) \)

Defined by induction/recursion on the regular expression!

- \( L(r_1) = L(r_2) \implies r_1 = r_2 \)
- We say that “\( w \) matches \( r \)” if \( w \in L(r) \).
Examples

0 | 1

(0|1)*

(0|1) 0* 1*

0*110* | 1*001*
Regexp Exercise

\[ \Sigma = \{0, 1\} \]

- Strings where every 1 is followed by a 0.
- Strings where no 1 is followed by a 0.
- Strings where every 1 is preceded by and followed by a 0.
From Regexp to NFA
(by induction/recursion on the regexp)

✓ ∅ is a regexp
✓ a is a regexp for every a ∈ Σ
✓ If r is a regexp, so is (r*)
✓ If r₁ and r₂ are regexps, so is (r₁r₂) and (r₁|r₂)