Harvard CS 121 and CSCI E-207

Lecture 5: Regular Expressions

Salil Vadhan

September 18, 2012

- **Reading:** Sipser, §1.3.

Regular Expressions

- Let $\Sigma = \{a, b\}$. The **regular expressions** over Σ are certain expressions formed using the symbols $\{a, b, (,), \varepsilon, \emptyset, \cup, \circ, {}^*\}$

- We use **red** for the strings under discussion (the **object language**) and **black** for the ordinary notation we are using for doing mathematics (the **metalanguage**).

- Construction Rules (= inductive/recursive definition):

  1. a, b, ε, $\emptyset$ are regular expressions (of size 1)

  2. If R_1 and R_2 are REs (of size s_1 and s_2), then $(R_1 \circ R_2)$, $(R_1 \cup R_2)$, and (R_1^*) are REs (of sizes $s_1 + s_2 + 1$, $s_1 + s_2 + 1$, and $s_1 + 1$, respectively).

- Examples:

$$(a \circ b) \qquad ((((a \circ (b^*)) \circ c) \cup ((b^*) \circ a))^*) \qquad (\emptyset^*)$$

What REs Do

- Regular expressions (which are strings) represent languages (which are sets of strings), via the function L:

$$
\begin{array}{lrcl}
(1) & L(a) & = & \{a\} \\
(2) & L(b) & = & \{b\} \\
(3) & L(\varepsilon) & = & \{\varepsilon\} \\
(3) & L(\emptyset) & = & \emptyset \\
(4) & L((R_1 \circ R_2)) & = & L(R_1) \circ L(R_2) \\
(5) & L((R_1 \cup R_2)) & = & L(R_1) \cup L(R_2) \\
(6) & L((R_1^*)) & = & L(R_1)^*
\end{array}
$$

- Example:

$$L(((a^*) \circ (b^*))) \quad =$$

- $L(\cdot)$ is called the **semantics** of the expression.

Syntactic Shorthand

- Omit many parentheses, because union and concatenation of languages are associative. For example,

  for any languages L_1, L_2, L_3:

  $$(L_1 L_2) L_3 = L_1 (L_2 L_3)$$

  and therefore for any regular expressions R_1, R_2, R_3,

  $$L\big((R_1 \circ (R_2 \circ R_3))\big) = L\big(((R_1 \circ (R_2 \circ R_3)))\big)$$

- Omit $\circ$ symbol

- Drop the distinction between red and black, between object language and metalanguage.

Semantic equivalence

The following are equivalent:

$$((ab)c) \qquad (a(bc)) \qquad abc$$

or strictly speaking

$$((a \circ b) \circ c) \qquad (a \circ (b \circ c))$$

- **Equivalent** means:

  “same semantics—same $L(\cdot)$-value—maybe different syntax”

More syntactic sugar

- By convention, $*$ takes precedence over $\circ$, which takes precedence over $\cup$.

  So $a \cup bc^*$ is equivalent to $(a \cup (b \circ (c^*)))$.

- Σ is shorthand for $a \cup b$ (or the analogous RE for whatever alphabet is in use).

Examples of Regular Languages

Strings ending in $a = \Sigma^* a$

Strings containing the substring $abaab =$?

$(aa \cup ab \cup ba \cup bb)^* =$?

Strings with even # of a's $= (b \cup ab^*a)^*$
$= b^*(ab^*ab^*)^*$

Strings with $\leq$ two a's $=$?

Strings of form $x_1 x_2 \cdots x_k$, $k \geq 0$, each $x_i \in \{aab, aaba, aaa\}$ = ?

Decimal numerals, no leading zeroes
$= 0 \cup ((1 \cup \ldots \cup 9)(0 \cup \ldots \cup 9)^*)$

All strings with an even # of a's and an even # of b's
$= (b \cup ab^*a)^* \cap (a \cup ba^*b)^*$ but this isn't a regular expression

Equivalence of REs and FAs

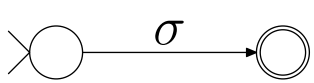
Recall: we call a language **regular** if there is a finite automaton that recognizes it.

Theorem: For every regular expression R, $L(R)$ is regular.

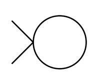
Proof:

Induct on the construction of regular expressions (“structural induction”).

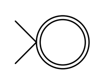
Base Case: R is a, b, ε, or $\emptyset$



accepts $\{\sigma\}$



accepts $\emptyset$



accepts $\{\varepsilon\}$

Equivalence of REs and FAs, continued

Inductive Step: If R_1 and R_2 are REs and $L(R_1)$ and $L(R_2)$ are regular (inductive hyp.), then so are:

$$\begin{aligned} L((R_1 \circ R_2)) &= L(R_1) \circ L(R_2) \\ L((R_1 \cup R_2)) &= L(R_1) \cup L(R_2) \\ L((R_1^*)) &= L(R_1)^* \end{aligned}$$

(By the closure properties of the regular languages).

Proof is constructive (actually produces the equivalent finite automaton, not just proves its existence).

Example Conversion of a RE to a FA

$$(a \cup \varepsilon)(aa \cup bb)^*$$

The Other Direction

Theorem: For every regular language L, there is a regular expression R such that $L(R) = L$.

Proof: Next time.