CSC236 fall 2016

languages: definitions and proofs

Danny Heap
heap@cs.toronto.edu / BA4270 (behind elevators)
http://www.cdf.toronto.edu/~csc236h/fall/
416-978-5899

Using Introduction to the Theory of Computation, Chapter 7

Outline

FSAs formally

formal languages

notes

build an automaton with formalities...

quintuple: $(Q, \Sigma, q_0, F, \delta)$

Q is set of states, Σ is finite, non-empty alphabet, q_0 is start state

F is set of accepting states, and $\delta : Q \times \Sigma \mapsto Q$ is transition function

We can extend $\delta : Q \times \Sigma \mapsto Q$ to a transition function that tells us what state a **string** s takes the automaton to:

$$\delta^* : Q \times \Sigma^* \mapsto Q \qquad \delta^*(q, s) = \begin{cases} q & \text{if } s = \varepsilon \\ \delta(\delta^*(q, s'), x) & \text{if } s' \in \Sigma^*, \\ & \quad x \in \Sigma, s = s'x \end{cases}$$

String s is accepted **if and only if** $\delta^*(q_0, s) \in F$, it is rejected otherwise.

example — an odd machine

devise a machine that accepts strings over $\{a, b\}$ with an odd number of as

Formal proof requires inductive proof of state invariant:

$$\delta^*(E, s) = \begin{cases} E & \text{only if } s \text{ has even number of } a\text{s} \\ O & \text{only if } s \text{ has odd number of } a\text{s} \end{cases}$$

more odd/even: intersection

L is the language of binary strings

with an odd number of as, and at least one b

Devise a machine for L

more odd/even: union

L is the language of binary strings
with an odd number of as, **or** at least one b
Devise a machine that accepts L,

some definitions

alphabet: finite, non-empty set of symbols, e.g. $\{a, b\}$ or $\{0, 1, -1\}$. Conventionally denoted Σ.

string: finite (including empty) sequence of symbols over an alphabet: abba is a string over $\{a, b\}$. Convention: ε is the empty string, never an allowed symbol, Σ^* is set of all strings over Σ.

language: Subset of Σ^* for some alphabet Σ. Possibly empty, possibly infinite subset. E.g. $\{\}$, $\{aa, aaa, aaaa, \ldots\}$.

N.B.: $\{\} \neq \{\varepsilon\}$.

Many problems can be reduced to languages: logical formulas, identifiers for compilation, natural language processing. Key question is recognition:

Given language L and string s, is $s \in L$?

Languages may be described either by descriptive generators (for example, regular expressions) or procedurally (e.g. finite state automata)

more notation

string length: denoted $|s|$, is the number of symbols in s, e.g. $|bba| = 3$.

$s = t$: if and only if $|s| = |t|$, and $s_i = t_i$ for $1 \leq i \leq |s|$.

s^R: reversal of s is obtained by reversing symbols of s, e.g. $1011^R = 1101$.

st or $s \circ t$: contcatenation of s and t — all characters of s followed by all those of t, e.g. $bba \circ bb = bbabb$.

s^k: denotes s concatenated with itself k times. E.g., $ab^3 = ababab$, $101^0 = \varepsilon$.

Σ^n: all strings of length n over Σ, Σ^* denotes all strings over Σ.

language operations

$\overline{L}$: Complement of L, i.e. $\Sigma^* - L$. If L is language of strings over $\{0,1\}$ that start with 0, then $\overline{L}$ is the language of strings that begin with 1 plus the empty string.

$L \cup L'$: union

$L \cap L'$: intersection

$L - L'$: difference

$\mathrm{Rev}(L)$: $= \{s^R : s \in L\}$

concatenation: LL' or $L \cdot L' = \{rt | r \in L, t \in L'\}$. Special cases $L\{\varepsilon\} = L = \{\varepsilon\}L$, and $L\{\} = \{\} = \{\}L$.

more language operations

exponentiation: L^k is concatenation of L k times. Special case, $L^0 = \{\varepsilon\}$, including $L = \{\}$ (!)

Kleene star: $L^* = L^0 \cup L^1 \cup L^2 \cup \ldots$.

notes