CSC236 fall 2018

languages: definitions

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

Using Introduction to the Theory of Computation,
Chapter 7
Outline

formal languages

regular expressions

NFSAs

notes
some definitions

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

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

language: Subset of \(\Sigma^*\) for some alphabet \(\Sigma\). 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 operations

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 \(0 \leq i < |s|\).

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

\(st\) or \(s \circ t\): concatenation 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., \(ab^3 = ababab\), \(101^0 = \varepsilon\).

\(\Sigma^n\): all strings of length \(n\) over \(\Sigma\), \(\Sigma^*\) denotes all strings over \(\Sigma\).


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

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

concatenation: \( LL' \) or \( L \circ L' = \{ rt | r \in L, t \in L' \} \). 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: \( L^* = L^0 \cup L^1 \cup L^2 \cup \ldots \)
another way to define languages
In addition to the language accepted by DFSA $L(M)$
and set description $L = \{\ldots\}$.

Definition: The regular expressions (regexps or REs) over alphabet $\Sigma$ is the smallest set such that:

1. $\emptyset$, $\varepsilon$, and $x$, for every $x \in \Sigma$ are REs over $\Sigma$
2. if $T$ and $S$ are REs over $\Sigma$, then so are:
   - $(T + S)$ (union) — lowest precedence operator
   - $(TS)$ (concatenation) — middle precedence operator
   - $T^*$ (star) — highest precedence
regular expression to language:

The \( L(R) \), the language denoted (or described) by \( R \) is defined by structural induction:

**Basis:** If \( R \) is a regular expression by the basis of the definition of regular expressions, then define \( L(R) \):

- \( L(\emptyset) = \emptyset \) (the empty language — no strings!)
- \( L(\varepsilon) = \{\varepsilon\} \) (the language consisting of just the empty string)
- \( L(x) = \{x\} \) (the language consisting of the one-symbol string)

**Induction step:** If \( R \) is a regular expression by the induction step of the definition, then define \( L(R) \):

- \( L(S + T) = L(S) \cup L(T) \)
- \( L(ST) = L(S)L(T) \)
- \( L(T^*) = L(T)^* \)
regexp examples

- csc207 regex practice
- regex crosswords
- command-line REs

- \( L(0 + 1) = \{0, 1\} \)
- \( L((0 + 1)^*) \) All binary strings over \( \{0, 1\} \)
- \( L((01)^*) = \{\epsilon, 01, 0101, 010101, \ldots\} \)
- \( L(0^*1^*) \) 0 or more 0s followed by 0 or more 1s.
- \( L(0^* + 1^*) \) 0 or more 0s or 0 or more 1s.
- \( L((0 + 1)(0 + 1)^*) \) Non-empty binary strings over \( \{0, 1\} \).
example

$L = \{ x \in \{0, 1\}^* \mid x \text{ begins and ends with a different bit} \}$
RE identities
some of these follow from set properties... others require some proof (see 7.2.5 example)

- communitativity of union: \( R + S \equiv S + R \)
- associativity of union: \( (R + S) + T \equiv R + (S + T) \)
- associativity of concatenation: \((RS)T \equiv R(ST)\)
- left distributivity: \(R(S + T) \equiv RS + RT\)
- right distributivity: \((S + T)R \equiv SR + TR\)
- identity for union: \(R + \emptyset \equiv R\)
- identity for concatenation: \(R\epsilon \equiv R \equiv \epsilon R\)
- annihilator for concatenation: \(\emptyset R \equiv \emptyset \equiv R\emptyset\)
- idempotence of Kleene star: \((R^*)^* \equiv R^*\)
non-deterministic FSA (NFSA) example

FSA that accepts $L((010 + 01)^*)$

convenient!
non-deterministic FSA (NFSA) example

FSA that accepts $L((010 + 01)^*)$
NFSAs are real
...you can always convert them to DFSA

Use subset construction, notes page 219