11/18

Regular Expressions

Formal definitions

regex

A **regular expression** over an alphabet Σ is any of the following:

- ∅ (the empty regular expression)
- ε
- *a* (for any $a \in \Sigma$)

base case or foundation case

Moreover, if R_1 and R_2 are regular expressions over Σ, then so are: $R_1 \mid R_2$, $R_1 \cdot R_2$, $R_1{}^*$

inductive rules

Every regex R has a language L(R) associated with it.

regular language (or regular set) : A language (or set) is regular if and only if it can be defined using a regular expression.

i.e if it can be written L(R) for regex R.

like arithmetic logic, sets.

Regular expressions as an expression language

regular expression = pattern describing a set of strings

→ regular set.

Operands $a \in \Sigma$, ε

Operators

precedence

low

$a \mid b$ alternation ("or") also called choice, union.

$a \cdot b$ concatentation ("followed by") or catenation

also written as ab $a \cdot a \cdot a = a^3$ only for actual numeric values

a^* iteration (0 or more) also called closure, Kleene closure.

high.

parens are not operators but used for grouping overriding precedence.

Examples

Express each of these using a regular expression (where Σ = {0, 1}):

1. the set consisting of the strings 0, 11, and 010

   0 | 11 | 010

2. the set of strings that contain exactly one 1

$0^* 1 0^*$

ε ✗ 0110 ✗

0100 ✓ 1000 ✓

3. the set of strings that contain two consecutive 0s

$(0|1)^* 00 (0|1)^*$

4. the set of strings that end with two 0s

$(0|1)^* 00$

exactly 2 0s $(0|1)^* 100 | 00$

Connection between finite automata and regular expressions

Theorem: A language *L* is regular if and only if it is accepted by some finite automaton.

1) For every regex R, $\exists$ finite automaton M s.t. $L(R) = L(M)$

2) For every language L decidable by some finite automaton, $\exists$ regex R s.t. $L = L(R)$

* But finite automaton might not be deterministic

Theorem: Let *N* be a nondeterministic finite automaton. Then there exists a deterministic finite automaton *M* such that L(*N*) = L(*M*).

So we do DFA → regex

and regex → NFA

But DFA might have exponential # of states

(compare to original NFA)