Regular Expressions

Seungjin Choi

Department of Computer Science
POSTECH, Korea
seungjin@postech.ac.kr
http://www.postech.ac.kr/∼seungjin

Outline

- Regular expressions: One type of language-defining notation
- Regular expressions and finite automata
- Regular grammars (will be discussed in Lecture 4)

Operators

- Union:
$$L \cup M = \{w \mid w \in L \text{ or } w \in M\}.$$
- Concatenation:
$$LM = \{w \mid w = xy,\ x \in L,\ y \in M\}.$$
- Powers:
$$L^0 = \{\epsilon\},\ L^1 = L,\ L^{n+1} = LL^n.$$
- Kleene Closure (star-closure):
$$L^* = L^0 \cup L^1 \cup L^2 \cdots = \cup_{i=0}^{\infty} L^i.$$

ϕ^i

Question: What are ϕ^0, ϕ^i, ϕ^*?

- $\phi^0 = \{\epsilon\}$.
- $\phi^i, i \geq 1$ is empty since we cannot select any strings from the empty set.
- $\phi^* = \{\epsilon\}$.

Regular Expressions

- A FA (DFA or NFA) is a "blueprint" for constructing a machine recognizing a regular language. (machine-like description)
- A regular expression is a "user-friendly" declarative way of describing a regular language. (algebraic description)
- Involves a combination of:
  - strings of symbols from Σ
  - parentheses
  - operators $(+,\ \cdot,\ *)$. (union, concatenation, star-closure)

Examples

- $\{a, b, c\}$ (regular language) $\Longleftrightarrow a + b + c$ (regular expression).
- $(a + b \cdot c)^*$ represents the star-closure of $\{a\} \cup \{bc\}$, which is $\{\epsilon, a, bc, abc, bca, bcbc, aaa, \ldots\}$.
- $01^* + 10^*$: The language consisting of all strings that are either a single 0 followed by any number of 1's or a single 1 followed by any number of 0's.
- `UNIX grep` command
- `UNIX Lex (lexical analyzer generator)` and `Flex (fast lex)` tools

Inductive Definition of Regular Expressions

Definition 1. *Let Σ be a given alphabet. Then*

1. *ϕ, ϵ, and $a \in \Sigma$ are all regular expressions. These are called primitive regular expressions.*
2. *If r_1 and r_2 are regular expressions, so are $r_1 + r_2,\ r_1 \cdot r_2,\ r_1^*,\ (r_1)$.*
3. *A string is a regular expression if and only if it can be derived from primitive regular expressions by a finite number of applications of the rules described above.*

Language $L(r)$

Definition 2. *The language $L(r)$ denoted by any regular expression r, is defined by the following rules:*

1. *ϕ is a regular expression denoting the empty set.*
2. *ϵ is a regular expression denoting $\{\epsilon\}$.*
3. *For every $a \in \Sigma$, a is a regular expression denoting $\{a\}$.*
4. $L(r_1 + r_2) = L(r_1) \cup L(r_2)$.
5. $L(r_1 r_2) = L(r_1)L(r_2)$.
6. $L((r_1)) = L(r_1)$.
7. $L(r_1^*) = (L(r_1))^*$.

Example

Exhibit the language $L\left(a^* \cdot (a+b)\right)$ in set notation.

Solution.

$$
\begin{aligned}
L\left(a^* \cdot (a+b)\right) &= L(a^*)L(a+b) \\
&= (L(a))^* \left(L(a) \cup L(b)\right) \\
&= \{\epsilon, a, aa, aaa, \ldots\} \cdot \{a, b\} \\
&= \{a, aa, aaa, \ldots, b, ab, aab, \ldots\}.
\end{aligned}
$$

Another Example

Given $\Sigma = \{a, b\}$ and $r = (a+b)^*(a+bb)$, find $L(r)$.

$$
\begin{aligned}
L(r) &= L\left((a+b)^*\right) L(a+bb) \\
&= (L(a+b))^* \left(L(a) \cup L(bb)\right) \\
&= (L(a) \cup L(b))^* \left(L(a) \cup L^2(b)\right) \\
&= \{a, b\}^* \left(\{a\} \cup \{bb\}\right) \\
&= \{a, b\}^* \{a, bb\} \\
&= \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}\{a, bb\} \\
&= \{a, bb, aa, abb, ba, bbb, \ldots\}.
\end{aligned}
$$

$(a+b)^*$ represents any string of a's and b's.

$\{a\}^* = \{\epsilon, a, aa, \ldots\}$.

More Examples

1. Given $r = (aa)^*(bb)^*b$, determine $L(r)$.

2. For $\Sigma = \{0, 1\}$, give a regular expression r such that

$$L(r) = \{w \in \Sigma^* \mid w \text{ has at least one pair of consequtive zeros}\}.$$

3. Find a regular expression for

$$L = \{w \in \{0,1\}^* \mid w \text{ has no pair of consequtive zeros}\}.$$

Connection between Regular Expressions and Regular Languages

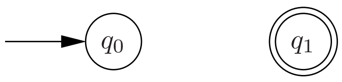
For every regular language, there is a regular expression and vice versa.

Theorem 1. *Let r be a regular expression. Then, there exists some NFA that accepts $L(r)$. Consequently $L(r)$ is a regular language.*

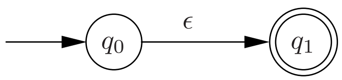
Proof is shown in next several slides!

Proof

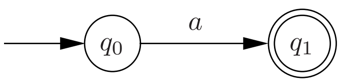
We begin with automata that accept languages for simple regular expressions ϕ, ϵ, and $a \in \Sigma$.



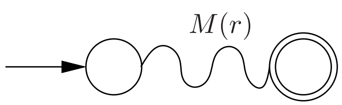
NFA accepts ϕ



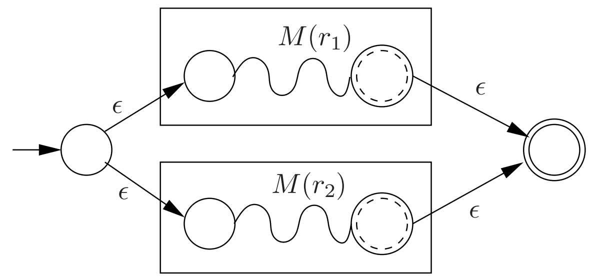
NFA accepts ϵ



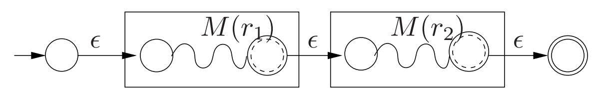
NFA accepts a



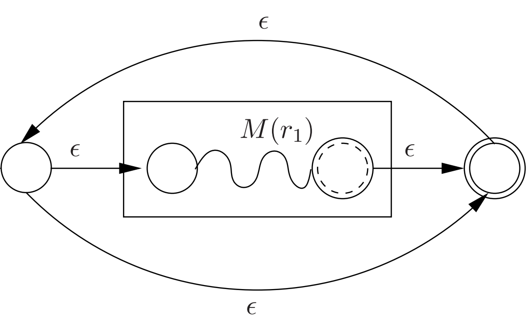
NFA accepts $L(r)$



Automaton for $L(r_1 + r_2)$



Automaton for $L(r_1 r_2)$



Automaton for $L(r_1^*)$

W.l.g we consider a NFA with a single final state.

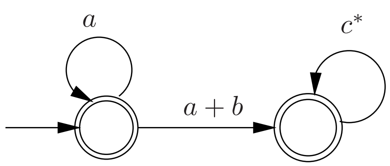
Just like building automata for $L(r_1+r_2)$, $L(r_1r_2)$, $L(r_1^*)$, we can build automata for arbitrary regular expressions. ■

Example

Find an NFA which accepts $L(r)$ where $r = (a + bb)^*(ba^* + \epsilon)$.

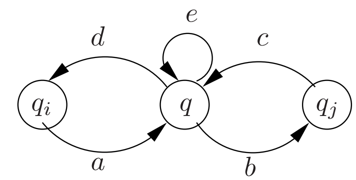
Generalized Transition Graph

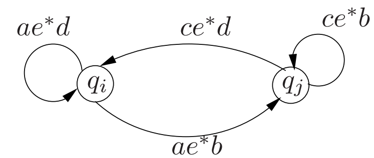
Generalized transition graph is a transition graph whose edges are labeled with regular expressions.



$L\left(a^* + a^*(a+b)c^*\right)$

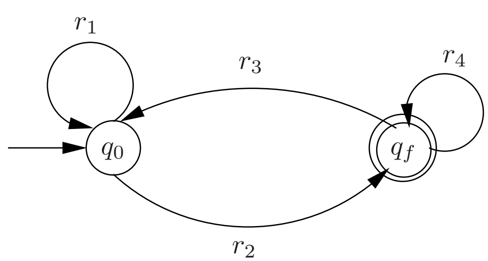
Remove State q





(After removing state q)

Canonical Form



Associated regular expression is

$$r = r_1^* r_2 \left(r_4 + r_3 r_1^* r_2\right)^*.$$

Regular Language and Regular Expression

Theorem 2. *Let L be a regular language. Then there exists a regular expression r such that $L = L(r)$.*

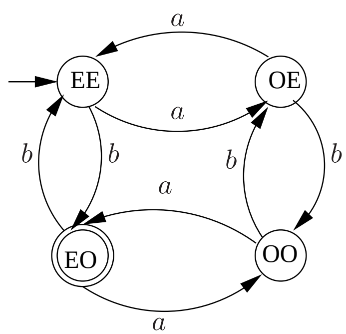
Proof. Let M be an NFA that accepts L. W.l.g. we can assume that M has only one final state and that $q_0 \notin F$. We interpret the graph M as a generalized transition graph and apply the construction (illustrated in previous slides) to it. We use the method of removing state q. We continue this process, removing one state after the other, until we reach the canonical form. Then the regular expression is of the form in the previous slide. ∎

Example

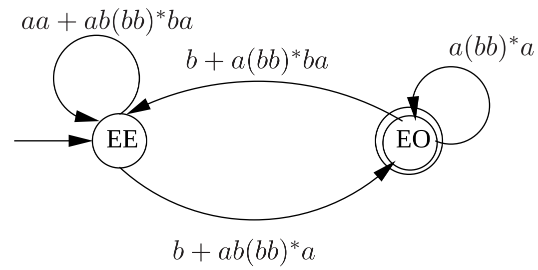
Find a regular expression for

$$L = \{w \in \{a,b\}^* \mid n_a(w) \text{ is even and } n_b(w) \text{ is odd}\}.$$

The associate DFA is given by



The canonical graph is of the form



Algebraic Laws: Commutativity and Associativity

- Commutative law for union: $r_1 + r_2 = r_2 + r_1$
- Commutative law for concatenation: $r_1 r_2 \neq r_2 r_1$
- Associative law for union: $(r_1 + r_2) + r_3 = r_1 + (r_2 + r_3)$
- Associative law for concatenation: $(r_1 r_2) r_3 = r_1 (r_2 r_3)$

Algebraic Laws: Identities and Annihilators

- Identities (ϕ and ϵ)

$$\begin{aligned} \phi + r &= r + \phi = r \\ \epsilon\, r &= r\, \epsilon = r \end{aligned}$$

- Annihilator (ϕ)

$$\phi\, r = r\, \phi = \phi$$

Algebraic Laws: Distributive Laws

- Left distributive lat of concatenation over union

$$r_1(r_2 + r_3) = r_1r_2 + r_1r_3$$

- Right distributive law of concatenation over union

$$(r_2 + r_3)r_1 = r_2r_1 + r_3r_1$$

Algebraic Laws: Idempotent Law

Definition 3. *An operator is said to be idempotent if the result of applying it to two of the same values as arguments is that that value.*

- Common arithmetic operators are not idempotent, i.e.,

$$x + x \neq x, \qquad x \times x \neq x.$$

- In regular expressions, idempotent law for union

$$r + r = r$$

Algebraic Laws: Laws Involving Closures

- $(r^*)^* = r^*$
- $\phi^* = \epsilon$
- $\epsilon^* = \epsilon$
- $r^+ = rr^* = r^*r$
- $r^* = r^+ + \epsilon$