Regular Expressions

Reading: Chapter 3
Regular Expressions vs. Finite Automata

- Offers a declarative way to express the pattern of any string we want to accept
  - E.g., 01*+ 10*

- Automata => more machine-like
  - < input: string , output: [accept/reject] >

- Regular expressions => more program syntax-like

- Unix environments heavily use regular expressions
  - E.g., bash shell, grep, vi & other editors, sed

- Perl scripting – good for string processing
- Lexical analyzers such as Lex or Flex
Regular Expressions

Regular expressions

Finite Automata (DFA, NFA, ε-NFA)

Syntactical expressions

Automata/machines

Regular Languages

Formal language classes
Language Operators

- **Union** of two languages:
  - \( L \cup M \) = all strings that are either in \( L \) or \( M \)
  - **Note:** A union of two languages produces a third language

- **Concatenation** of two languages:
  - \( L \cdot M \) = all strings that are of the form \( xy \)
    s.t., \( x \in L \) and \( y \in M \)
  - The *dot* operator is usually omitted
    - i.e., \( LM \) is same as \( L \cdot M \)
“$i$” here refers to how many strings to concatenate from the parent language $L$ to produce strings in the language $L^i$.

**Kleene Closure (the * operator)**

- **Kleene Closure** of a given language $L$:
  - $L^0 = \{ \epsilon \}$
  - $L^1 = \{ w | \text{for some } w \in L \}$
  - $L^2 = \{ w_1w_2 | w_1 \in L, w_2 \in L \text{ (duplicates allowed)} \}$
  - $L^i = \{ w_1w_2...w_i | \text{all } w\text{'s chosen are } \in L \text{ (duplicates allowed)} \}$
  - (Note: the choice of each $w_i$ is independent)
  - $L^* = \bigcup_{i=0}^{\infty} L^i$ (arbitrary number of concatenations)

**Example:**
- Let $L = \{ 1, 00 \}$
  - $L^0 = \{ \epsilon \}$
  - $L^1 = \{ 1,00 \}$
  - $L^2 = \{ 11,100,001,0000 \}$
  - $L^3 = \{ 111,1100,1001,10000,000000,00001,00100,0011 \}$
  - $L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup ...$
Kleene Closure (special notes)

- $L^*$ is an infinite set iff $|L| \geq 1$ and $L \neq \{\varepsilon\}$  
  Why?
- If $L = \{\varepsilon\}$, then $L^* = \{\varepsilon\}$  
  Why?
- If $L = \emptyset$, then $L^* = \{\varepsilon\}$  
  Why?

$\Sigma^*$ denotes the set of all words over an alphabet $\Sigma$

- Therefore, an abbreviated way of saying there is an arbitrary language $L$ over an alphabet $\Sigma$ is:
  - $L \subseteq \Sigma^*$
Building Regular Expressions

- Let $E$ be a regular expression and the language represented by $E$ is $L(E)$.

- Then:
  - $(E) = E$
  - $L(E + F) = L(E) \cup L(F)$
  - $L(E \cdot F) = L(E) \cdot L(F)$
  - $L(E^*) = (L(E))^*$
Example: how to use these regular expression properties and language operators?

- \( L = \{ w \mid w \text{ is a binary string which does not contain two consecutive } 0\text{s or two consecutive } 1\text{s anywhere} \} \)
  - E.g., \( w = 01010101 \) is in \( L \), while \( w = 10010 \) is not in \( L \)
- **Goal:** Build a regular expression for \( L \)
- Four cases for \( w \):
  - Case A: \( w \) starts with 0 and \(|w|\) is even
  - Case B: \( w \) starts with 1 and \(|w|\) is even
  - Case C: \( w \) starts with 0 and \(|w|\) is odd
  - Case D: \( w \) starts with 1 and \(|w|\) is odd
- Regular expression for the four cases:
  - Case A: \((01)^*\)
  - Case B: \((10)^*\)
  - Case C: \(0(10)^*\)
  - Case D: \(1(01)^*\)
- Since \( L \) is the union of all 4 cases:
  - Reg Exp for \( L = (01)^* + (10)^* + 0(10)^* + 1(01)^* \)
- If we introduce \( \varepsilon \) then the regular expression can be simplified to:
  - Reg Exp for \( L = (\varepsilon +1)(01)^*(\varepsilon +0) \)
Precedence of Operators

- Highest to lowest
  - * operator (star)
  - . (concatenation)
  - + operator

Example:

- $01^* + 1 = (0 . ((1)^*)) + 1$
Finite Automata (FA) & Regular Expressions (Reg Ex)

To show that they are interchangeable, consider the following theorems:

- **Theorem 1**: For every DFA $A$ there exists a regular expression $R$ such that $L(R)=L(A)$

- **Theorem 2**: For every regular expression $R$ there exists an $\varepsilon$-NFA $E$ such that $L(E)=L(R)$

Proofs in the book
DFA to RE construction

Informally, trace all distinct paths (traversing cycles only once) from the start state to *each of the* final states and enumerate all the expressions along the way.

Example:

```
<table>
<thead>
<tr>
<th>State</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>q1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>q2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

Expressions:

- 1* (1*)
- 0 (0*)
- 1 (0+1)*

Q) What is the language?

```
1* 00* 1(0+1)*
```
Theorem 2

RE to ε-NFA construction

Example: \((0+1)^*01(0+1)^*\)
Algebraic Laws of Regular Expressions

- **Commutative:**
  - $E + F = F + E$

- **Associative:**
  - $(E + F) + G = E + (F + G)$
  - $(EF)G = E(FG)$

- **Identity:**
  - $E + \Phi = E$
  - $\varepsilon E = E \varepsilon = E$

- **Annihilator:**
  - $\Phi E = E \Phi = \Phi$
Algebraic Laws…

- **Distributive:**
  - \( E(F+G) = EF + EG \)
  - \( (F+G)E = FE+GE \)

- **Idempotent:** \( E + E = E \)

- **Involving Kleene closures:**
  - \( (E^*)^* = E^* \)
  - \( \Phi^* = \varepsilon \)
  - \( \varepsilon^* = \varepsilon \)
  - \( E^+ = EE^* \)
  - \( E? = \varepsilon + E \)
True or False?

Let $R$ and $S$ be two regular expressions. Then:

1. $((R^*)^*)^* = R^*$ ?
2. $(R+S)^* = R^* + S^*$ ?
3. $(RS + R)^* RS = (RR*S)^*$ ?
Summary

- Regular expressions
- Equivalence to finite automata
- DFA to regular expression conversion
- Regular expression to $\varepsilon$-NFA conversion
- Algebraic laws of regular expressions
- Unix regular expressions and Lexical Analyzer