Strings and Language Operations

- Concatenation
- Exponentiation
- Kleene star
- Regular expressions

Strings and Languages Operations

- Concatenation
- Exponentiation
- Kleene Star
- Regular Expressions

String Concatenation

- If x and y are strings over alphabet Σ, the concatenation of x and y is the string xy formed by writing the symbols of x and the symbols of y consecutively.
- Suppose x = abb and y = ba
  - xy = abbb
  - yx = baabb

Properties of String Concatenation

- Suppose x, y, and z are strings.
- Concatenation is not commutative.
- xy is not guaranteed to be equal to yx
- Concatenation is associative
  - (xy)z = x(yz) = xyz
- The empty string is the identity for concatenation
  - x\ = /x = x

Language Concatenation

- Suppose L₁ and L₂ are languages (sets of strings).
- The concatenation of L₁ and L₂, denoted L₁L₂, is defined as
  - L₁L₂ = { xy | x ∈ L₁ and y ∈ L₂ }
- Example,
  - Let L₁ = { ab, bba } and L₂ = { aa, bb, ba }
  - What is L₁L₂?
    - Solution
      - Let x₁ = ab, x₂ = bba, y₁ = aa, y₂ = b, y₃ = ba
      - L₁L₂ = { x₁y₁, x₁y₂, x₁y₃, x₂y₁, x₂y₂, x₂y₃, x₃y₁, x₃y₂, x₃y₃ }
      - = { abaa, abb, abba, bbaa, bbb, bbaa }
ASSOCIATIVITY OF LANGUAGE CONCATENATION

- \((L_1 L_2)L_3 = L_1 (L_2 L_3)\)
- **Example**
  - Let \(L_1 = \{a, b\}\), \(L_2 = \{c, d\}\), and \(L_3 = \{e, f\}\)
  - \(L_1 L_2 L_3 = (a, b)(c, d)(e, f) = \{ace, acf, ade, aef, bce, bdf\}\)

SPECIAL CASES

- What language is the identity for language concatenation?
  - The set containing only the empty string \(\Lambda\): \(\{\Lambda\}\)
  - **Example**
    - \((\{aab, aba, abc\}\{\Lambda\}) = (\Lambda)\{aab, aba, abc\} = \{aab, aba, abc\}\)
- What about \(\{\}\)?
  - For any language \(L\), \(L\{\}\ = \{\}\ L = \{\}\)
    - Thus \(\{\}\) for concatenation is like 0 for multiplication
    - **Example**
      - \((\{aab, aba, abc\}\{\}\) = \(\{aab, aba, abc\}\{\}\)
      - The intuitive reason is that we must choose a string from both sets that are being concatenated, but there is nothing to choose from \(\{\}\).

EXPONENTIATION

- We use exponentiation to indicate the number of items being concatenated
  - **Symbols**
  - **Strings**
  - **Set of symbols (Σ for example)**
  - **Set of strings (languages)**
  - \(a^3 = aaa\)
  - \(x^3 = xxx\)
  - \(Σ^3 = ΣΣΣ = \{ x ∈ Σ^* | |x| = 3 \}\)
  - \(L^3 = LLL\)

EXAMPLES OF EXPONENTIATION

- Let \(x = abb\), \(Σ = \{a, b\}\), \(L = \{ab, b\}\)
  - \(a^4 = aaaa\)
  - \(x^3 = (abb)(abb)(abb) = abbbabb\)
  - \(Σ^3 = ΣΣΣ = \{a, b\}[a, b]\)
    - \(\{aaa, aab, aba, abb, baa, bab, bba,bbb\}\)
  - \(L^3 = LLL = \{ab, b\}[a, b]\)
    - \(\{abab, ababb, abbab, abb\}, babab, abbabb, bbb\)
**Kleene Star**

- Kleene * is a unary operation on languages.
- Kleene * is not an operation on strings
  - However, see the pages on regular expressions.
- $L^*$ represents any finite number of concatenations of $L$.
  
  $$L^* = \bigcup_{k \geq 0} L^k = L^0 \cup L^1 \cup L^2 \cup \ldots$$

- For any $L$, $\Lambda$ is always an element of $L^*$
  - because $L^0 = \{ \Lambda \}$
- Thus, for any $L$, $L^* \neq \emptyset$

**Example of Kleene Star**

- Let $L=\{aa\}$
- $L^0=\{\Lambda\}$
- $L^1=\{aa\}$
- $L^2=\{aaaa\}$
- $L^3=\ldots$
- $L^* = L_0 \cup L_1 \cup L_2 \cup L_3 \ldots$
- $= \{\Lambda, aa, aaaa, aaaaaaa, \ldots\}$
- $= \text{set of all strings that can be obtained by concatenating 0 or more copies of aa}$

**Regular Languages**

- Regular languages are languages that can be obtained from the very simple languages over $\Sigma$, using only
  - Union
  - Concatenation
  - Kleene Star

**Examples of Regular Languages**

- $\{aab\}$ (i.e. $\{a\} \{a\} \{b\}$)
- $\{aa,b\}$ (i.e. $\{a\} \{a\} \cup \{b\}$)
- $\{a,b\}^*$ language of strings that can be obtained by concatenating any number of a’s and b’s
- $\{bb\} \{a,b\}^*$ language of strings that begin with bb (followed by any number of a’s and b’s)
- $\{a\}^* \{bb,\Lambda\}$ language of strings that begin with any number of a’s and end with an optional bb.
- $\{a\}^* \cup \{b\}^*$ language of strings that consist of only a’s or only b’s and $\Lambda$.

**Regular Expressions**

- We can simplify the formula for regular languages slightly by
  - leaving out the set brackets $\{\}$ and
  - replacing $\cup$ with $+$
- The results are called *regular expressions.*
### Examples of Regular Expressions

<table>
<thead>
<tr>
<th>Set notation</th>
<th>Regular Expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>{aab}</td>
<td>aab</td>
</tr>
<tr>
<td>{aa,b}</td>
<td>(aa) ∪ (b)</td>
</tr>
<tr>
<td>(a,b)*</td>
<td>(a) ∪ (b)</td>
</tr>
<tr>
<td>{bb}{a,b}*</td>
<td>{bb}(a+b)</td>
</tr>
<tr>
<td>(a)*{bb,ʌ}</td>
<td>(a)∗{(bb) ∪ (ʌ)}</td>
</tr>
<tr>
<td>{a})<em>{b}</em></td>
<td>a∗+b*</td>
</tr>
</tbody>
</table>

### String or Language?

- Consider the regular expression $a^*(bb+ʌ)$
- $a^*(bb+ʌ)$ is a **string** over alphabet {a, b, *, +, /, (, ʌ, ɸ}
- $a^*(bb+ʌ)$ represents a **language** over alphabet {a, b}
  - It represents the language of strings over (a,b) that begin with any number of a’s and end with an optional bb.
- Some regular expressions look just like strings over alphabet {a,b}
  - Regular expression aaba represents the language {aaba}
  - Regular expression / represents the language (/)
- It should be clear from the context whether a sequence of symbols is a regular expression or just a string.