Lesson 3: Pumping lemma and regular expressions

Theme: Pumping lemma as a tool for establishing non-regular languages and regular expressions as alternative description of regular languages.

1 Pumping lemma

In the following, for a word $w$ and an integer $n \geq 0$, $w^n$ obtained by repeating $w$ for $n$ number of times, i.e., $\underbrace{w \cdots w}_n$. By default, we define $w^0 = \epsilon$.

Lemma 3.1 (pumping lemma) Let $A = \langle \Sigma, Q, q_0, F, \delta \rangle$ be an NFA. Suppose there is a word $x \in L(A)$ such that $|x| \geq |Q|$. Then, the word $x$ can be divided into three parts $u, v, w$, i.e., $x = uvw$, such that for every positive integer $n \geq 0$, $uv^n w \in L(A)$.

Lemma 3.2 (more refined pumping lemma) Let $A = \langle \Sigma, Q, q_0, F, \delta \rangle$ be an NFA, and let $x, y, z$ be words such that $xyz \in L(A)$ and $|y| \geq |Q|$. Then, the word $y$ can be divided into three parts $u, v, w$, i.e., $y = uvw$, such that for every positive integer $n \geq 0$, $xuv^n wz \in L(A)$.

2 Regular expressions

In the following we fix an alphabet $\Sigma$. Regular expressions (over $\Sigma$) are expressions built inductively as follows.

- $\emptyset$ is a regular expression.
- $a$ is a regular expression, for every symbol $a \in \Sigma$.
- If $e_1, e_2$ are regular expressions, then so are $(e_1 \cdot e_2)$ and $(e_1 \cup e_2)$.
- If $e$ is a regular expression, then so is $(e)^*$.

A regular expression $e$ over $\Sigma$ defines a language, denoted by $L(e)$, over the same alphabet as follows.

- If $e = \emptyset$, then $L(e) = \emptyset$.
- If $e = a$, where $a \in \Sigma$, then $L(e) = \{a\}$.
- If $e$ is of the form $(e_1 \cdot e_2)$, where $e_1$ and $e_2$ are regular expressions, then $L(e) = L(e_1) \cdot L(e_2)$.
- If $e$ is of the form $(e_1 \cup e_2)$, where $e_1$ and $e_2$ are regular expressions, then $L(e) = L(e_1) \cup L(e_2)$.
- If $e$ is of the form $(e_1)^*$, where $e_1$ is a regular expression, then $L(e) = L(e_1)^*$.

Usually, we omit writing $\cdot$ in $(e_1 \cdot e_2)$, and instead, we simply write $(e_1 e_2)$. Also, when there is no ambiguity, we will omit writing the brackets and simply write $e_1 e_2$ and $e_1^*$, instead of $(e_1 \cdot e_2)$ and $(e_1)^*$.

The following theorem states that the class of languages defined by regular expressions is exactly the class of regular languages.

Theorem 3.3 Regular expressions define precisely the class of regular languages. That is, for every regular expression $e$ over $\Sigma$, $L(e)$ is a regular language, and vice versa, for every DFA $A$, there is a regular expression $e$ such that $L(e) = L(A)$. 
Combining what we have learnt so far, we obtain three different, but equivalent, characterizations of regular languages, as stated below.

**Corollary 3.4** Let \( L \) be a language. Then, the following are equivalent.

- \( L \) is accepted by a DFA.
- \( L \) is accepted by an NFA.
- \( L \) is defined by a regular expression.

**Remark 3.5** The term regular expressions are commonly abbreviated as `regex`. In most literatures and websites, the term “regex” are used more often than “regular expression.” Due to its widespread applications, many modern programming languages now include libraries for regex. The following are some of them.

- Java: [https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html](https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html)
- Python: [https://docs.python.org/2/library/re.html](https://docs.python.org/2/library/re.html)