Formal languages and Automata

形式语言与自动机

Chapter3 REGULAR LANGUAGES AND REGULAR GRAMMARS

Beijing University of Posts and Telecommunications

北京邮电大学

刘咏彬 liuyb@bupt.edu.cn

2023.10

Regular Expression

- Regular expression is a concise(简洁的) way to describe regular language.
- This notation involves a combination of strings of symbols
- From some alphabet Σ, parentheses（括号）, and the operators +, ·, and □.
- + union（并） · concatenation（连接）, □ star-closure（星闭包）
- DEFINITION 3.1

Let Σ be a given alphabet. Then

1. $\varnothing, \lambda$, 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^*, and (r_1).
3. A string is a regular expression if and only if it can be derived from the primitive regular expressions by a finite number of applications of the rules in (2).

EXAMPLE of Regular Expressions

- a stands for $\{a\}$
- $a+b+c$ stands for $\{a, b, c\}$
- $(a+(b\cdot c))^*$ the star-closure of $\{a\}\cup\{bc\}$, that is, the language $\{\lambda, a, bc, aa, abc, bca, bcbc, aaa, aabc, ...\}$.
- For $\Sigma = \{a, b, c\}$, the string $(a+b\cdot c)^*\cdot(c+\emptyset)$ is a regular expression
- $(a + b +)$ is not a regular expression

Languages Associated with Regular Expressions

- DEFINITION 3.2

The language $L(r)$ denoted by any regular expression r is defined by the following rules.

1. $\varnothing$ is a regular expression denoting the empty set,
2. λ is a regular expression denoting $\{\lambda\}$,
3. For every $a \in \Sigma$, a is a regular expression denoting $\{a\}$.

   If r_1 and r_2 are regular expressions, then

4. $L(r_1 + r_2) = L(r_1) \cup L(r_2)$,
5. $L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$,
6. $L((r_1)) = L(r_1)$,
7. $L(r_1^*) = (L(r_1))^*$.

EXAMPLES of RE->RL

- Precedence（优先级）rules for the operators
  - star-closure > concatenation > union.

EXAMPLE 3.2

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

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

EXAMPLE 3.3

For $\Sigma = \{a, b\}$, the expression

$$r = (a+b)^* (a+bb)$$

is regular. It denotes the language

$$L(r) = \{a, bb, aa, abb, ba, bbb, \ldots\}.$$

EXAMPLES of RL->RE

EXAMPLE 3.5

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

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

One can arrive at an answer by reasoning something like this: Every string in $L(r)$ must contain 00 somewhere, but what comes before and what goes after is completely arbitrary. An arbitrary string on $\{0, 1\}$ can be denoted by $(0 + 1)^*$. Putting these observations together, we arrive at the solution

$$r = (0 + 1)^* 00 (0 + 1)^*.$$

EXAMPLES of RL->RE

EXAMPLE 3.6

Find a regular expression for the language

$$L = \left\{ w \in \{0,1\}^* : w \text{ has no pair of consecutive zeros} \right\}.$$

Even though this looks similar to Example 3.5, the answer is harder to construct. One helpful observation is that whenever a 0 occurs, it must be followed immediately by a 1. Such a substring may be preceded and followed by an arbitrary number of 1's. This suggests that the answer involves the repetition of strings of the form $1 \cdots 101 \cdots 1$, that is, the language denoted by the regular expression $(1^*011^*)^*$. However, the answer is still incomplete, since the strings ending in 0 or consisting of all 1's are unaccounted for. After taking care of these special cases we arrive at the answer

$$r = (1^*011^*)^* (0 + \lambda) + 1^* (0 + \lambda).$$

If we reason slightly differently, we might come up with another answer. If we see L as the repetition of the strings 1 and 01, the shorter expression

$$r = (1 + 01)^* (0 + \lambda)$$

might be reached. Although the two expressions look different, both answers are correct, as they denote the same language. Generally, there are an unlimited number of regular expressions for any given language.

Equivalence of Regular Expressions.

- Two regular expressions are equivalent if they denote the same language.
- One can derive a variety of rules for simplifying regular expressions (see Exercise 22 in the following exercise section), but since we have little need for such manipulations we will not pursue this.

22. Determine whether or not the following claims are true for all regular expressions r_1 and r_2. The symbol $\equiv$ stands for equivalence of regular expressions in the sense that both expressions denote the same language.

(a) $(r_1^*)^* \equiv r_1^*$

(b) $r_1^*(r_1 + r_2)^* \equiv (r_1 + r_2)^*$

(c) $(r_1 + r_2)^* \equiv (r_1^* r_2^*)^*$

(d) ~~$(r_1 r_2)^* \equiv r_1^* r_2^*$~~

END