بسم اللّه الرّحمن الرّحیم

دانشگاه صنعتی اصفهان ـ دانشکدهٔ مهندسی برق و کامپیوتر
(نیم‌سال تحصیلی ۴۰۰۱)

نظریهٔ زبان‌ها و ماشین‌ها

حسین فلسفین

Regular Expressions

One way of describing regular languages is via the notation of regular expressions. This notation involves a combination of strings of symbols from some alphabet Σ, parentheses, and the operators $\cup$, $\circ$, and $*$.

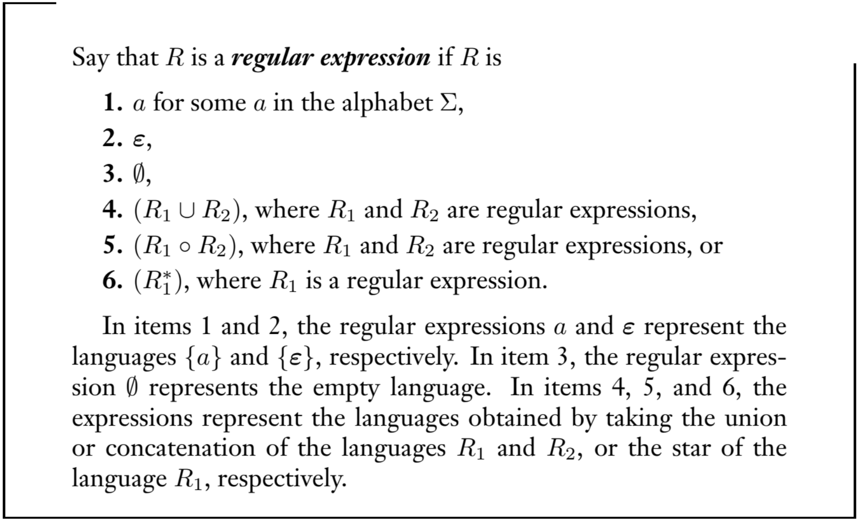
In arithmetic, we can use the operations $+$ and $\times$ to build up expressions such as $(5+3)\times 4$. Similarly, we can use the regular operations to build up expressions describing languages, which are called regular expressions. An example is: $(0\cup 1)0^*$. The value of the arithmetic expression is the number 32. The value of a regular expression is a language.

دربارهٔ عبارت منظم $(0 \cup 1)0^*$

In this case, the value is the language consisting of all strings starting with a 0 or a 1 followed by any number of 0s. We get this result by dissecting the expression into its parts. First, the symbols 0 and 1 are shorthand for the sets $\{0\}$ and $\{1\}$. So $(0 \cup 1)$ means $(\{0\} \cup \{1\})$. The value of this part is the language $\{0, 1\}$. The part 0^* means $\{0\}^*$, and its value is the language consisting of all strings containing any number of 0s. Second, like the $\times$ symbol in algebra, the concatenation symbol $\circ$ often is implicit in regular expressions. Thus $(0 \cup 1)0^*$ actually is shorthand for $(0 \cup 1) \circ 0^*$. The concatenation attaches the strings from the two parts to obtain the value of the entire expression.

دربارهٔ تقدّم و تأخّر عملگرها در عبارات منظم

In arithmetic, we say that $\times$ has precedence over $+$ to mean that when there is a choice, we do the $\times$ operation first. Thus in $2+3\times4$, the 3×4 is done before the addition. To have the addition done first, we must add parentheses to obtain $(2+3)\times4$. In regular expressions, the star operation is done first, followed by concatenation, and finally union, unless parentheses change the usual order.



Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. (R_1^*), where R_1 is a regular expression.

In items 1 and 2, the regular expressions a and ε represent the languages $\{a\}$ and $\{\varepsilon\}$, respectively. In item 3, the regular expression $\emptyset$ represents the empty language. In items 4, 5, and 6, the expressions represent the languages obtained by taking the union or concatenation of the languages R_1 and R_2, or the star of the language R_1, respectively.

A definition of this type is called an inductive definition.

Let Σ be a given alphabet. Then

1. $\varnothing$, λ, 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 $r_1 + r_2$, $r_1 \cdot r_2$, $\boldsymbol{r_1^*}$, 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).

دربارهٔ زبانی که توسط یک عبارت منظم توصیف می‌شود

☞ *When we want to distinguish between a regular expression R and the language that it describes, we write $L(R)$ to be the language of R.*
☞ *If R is a regular expression, we will let $L(R)$ denote the language associated with R.*

زبان نظیر یک عبارت منظم

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. $L(\boldsymbol{r_1^*}) = (L(r_1))^*$.

به بیان دیگر:

For each regular expression E, we'll associate a regular language $L(E)$ as follows, where A is an alphabet and R and S are any regular expressions:

$$\begin{aligned}
&L(\varnothing) = \varnothing, && \\
&L(\Lambda) = \{\Lambda\}, && \\
&L(a) = \{a\} && \text{for each } a \in A, \\
&L(R+S) = L(R) \cup L(S), && \\
&L(R \cdot S) = L(R)\, L(S) && \text{(language product)}, \\
&L(R^*) = L(R)^* && \text{(language closure)}.
\end{aligned}$$

چند مثال ساده

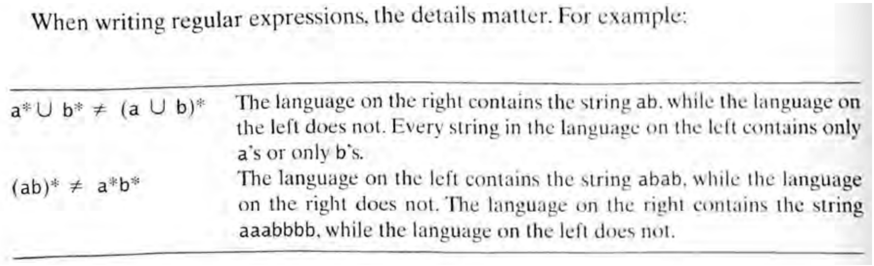
☞ *An example of a regular expression is* $(0 \cup 1)^*$. *It starts with the language* $(0 \cup 1)$ *and applies the* $*$ *operation. The value of this expression is the language consisting of all possible strings of* 0*s and* 1*s.*

☞ *If* $\Sigma = \{0, 1\}$, *we can write* Σ *as shorthand for the regular expression* $(0 \cup 1)$. *More generally, if* Σ *is any alphabet, the regular expression* Σ *describes the language consisting of all strings of length* 1 *over this alphabet, and* Σ^* *describes the language consisting of all strings over that alphabet.*

☞ *Similarly,* Σ^*1 *is the language that contains all strings that end in a* 1.

☞ *The language* $(0\Sigma^*) \cup (\Sigma^*1)$ *consists of all strings that start with a* 0 *or end with a* 1.

ضرورت دقت هنگام مواجهه با عبارات منظم



When writing regular expressions, the details matter. For example:

$a^* \cup b^* \neq (a \cup b)^*$	The language on the right contains the string ab, while the language on the left does not. Every string in the language on the left contains only a's or only b's.
$(ab)^* \neq a^*b^*$	The language on the left contains the string abab, while the language on the right does not. The language on the right contains the string aaabbbb, while the language on the left does not.

ضرورت دقت هنگام مواجهه با عبارات منظم

If we let R be any regular expression, we have the following identities. They are good tests of whether you understand the definition.

☞ $R \cup \emptyset = R$. *Adding the empty language to any other language will not change it.*

☞ $R \circ \varepsilon = R$. *Joining the empty string to any string will not change it.*

☞ $R \cup \varepsilon$ *may not equal* R. *For example, if* $R = 0$, *then* $L(R) = \{0\}$ *but* $L(R \cup \varepsilon) = \{0, \varepsilon\}$.

☞ $R \circ \emptyset$ *may not equal* R. *For example, if* $R = 0$, *then* $L(R) = \{0\}$ *but* $L(R \circ \emptyset) = \emptyset$.

مفهوم R^+ و R^k چیست؟

For convenience, we let R^+ be shorthand for RR^*. In other words, whereas R^* has all strings that are 0 or more concatenations of strings from R, the language R^+ has all strings that are 1 or more concatenations of strings from R. So $R^+ \cup \varepsilon = R^*$. In addition, we let R^k be shorthand for the concatenation of k R's with each other.

مثال‌ها

In the following instances, we assume that the alphabet Σ is $\{0,1\}$.

1. $0^*10^* = \{w|\ w \text{ contains a single } 1\}$.
2. $\Sigma^*1\Sigma^* = \{w|\ w \text{ has at least one } 1\}$.
3. $\Sigma^*001\Sigma^* = \{w|\ w \text{ contains the string } 001 \text{ as a substring}\}$.
4. $1^*(01^+)^* = \{w|\text{ every } 0 \text{ in } w \text{ is followed by at least one } 1\}$.
5. $(\Sigma\Sigma)^* = \{w|\ w \text{ is a string of even length}\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w|\text{ the length of } w \text{ is a multiple of } 3\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w|\ w \text{ starts and ends with the same symbol}\}$.
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$.
   The expression $0 \cup \varepsilon$ describes the language $\{0, \varepsilon\}$, so the concatenation operation adds either 0 or ε before every string in 1^*.
10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$.
11. $1^*\emptyset = \emptyset$.
    Concatenating the empty set to any set yields the empty set.
12. $\emptyset^* = \{\varepsilon\}$.
    The star operation puts together any number of strings from the language to get a string in the result. If the language is empty, the star operation can put together 0 strings, giving only the empty string.

مثال‌ها

$$(0+1)^*110(0+1)^* = \{w \in \{0,1\}^* \mid w \text{ includes 110 as a substring}\},$$

$$(0+1)(0+1)(0+1)(0+1)^*$$

$$= \{w \in \{0,1\}^* \mid \text{the length of } w \text{ is greater than or equal to 3}\},$$

$$0+1+0(0+1)^*0+1(0+1)^*1$$

$$= \{w \in \{0,1\}^* \mid \text{the first and the last symbols of } w \text{ are the same}\},$$

$$0^*1^*2^* = \{0^i1^j2^k \mid i \geq 0, j \geq 0, k \geq 0\}.$$

مثال

Regular expressions are useful tools in the design of compilers for programming languages. For example, a numerical constant that may include a fractional part and/or a sign may be described as a member of the language

$$(+ \cup - \cup \varepsilon)(D^+ \cup D^+.D^* \cup D^*.D^+)$$

where $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ *is the alphabet of decimal digits. Examples of generated strings are:* 72, 3.14159, $+7.$, *and* $-.01$.

مثال

The Language of Strings in $\{a, b\}^*$ with an Odd Number of a's

A string with an odd number of a's has at least one a, and the additional a's can be grouped into pairs. There can be arbitrarily many b's before the first a, between any two consecutive a's, and after the last a. The expression

$$b^*ab^*(ab^*a)^*b^*$$

is not correct, because it doesn't allow b's between the second a in one of the repeating pairs ab^*a and the first a in the next pair. One correct regular expression describing the language is

$$b^*ab^*(ab^*ab^*)^*$$

The expression

$$b^*a(b^*ab^*ab^*)^*$$

is also not correct, because it doesn't allow strings with just one a to end with b, and the expression

$$b^*a(b^*ab^*a)^*b^*$$

corrects the mistake. Another correct expression is

$$b^*a(b + ab^*a)^*$$

All of these could also be written with the single a on the right, as in

$$(b + ab^*a)^*ab^*$$

مثال

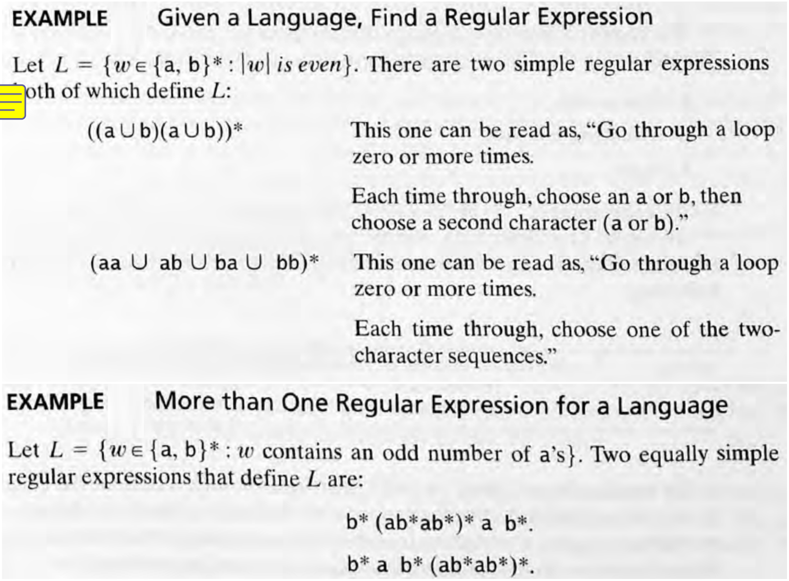
The Language of Strings in $\{a, b\}^*$ Ending with b and Not Containing aa

If a string does not contain the substring aa, then every a in the string either is followed immediately by b or is the last symbol in the string. If the string ends with b, then every a is followed immediately by b. Therefore, every string in the language L of strings that end with b and do not contain aa matches the regular expression $(b + ab)^*$. This regular expression does not describe L, however, because it allows the null string, which does not end with b. At least one of the two strings b and ab must occur, and so a regular expression for L is

$$(b + ab)^*(b + ab)$$

به عبارت دیگر: $(b \cup ab)^+$

مثال



EXAMPLE **Given a Language, Find a Regular Expression**

Let $L = \{w \in \{a, b\}^* : |w| \textit{ is even}\}$. There are two simple regular expressions both of which define L:

$((a \cup b)(a \cup b))^*$	This one can be read as, "Go through a loop zero or more times. Each time through, choose an a or b, then choose a second character (a or b)."
$(aa \cup ab \cup ba \cup bb)^*$	This one can be read as, "Go through a loop zero or more times. Each time through, choose one of the two-character sequences."

EXAMPLE **More than One Regular Expression for a Language**

Let $L = \{w \in \{a, b\}^* : w$ contains an odd number of a's$\}$. Two equally simple regular expressions that define L are:

$$b^* (ab^*ab^*)^* a\ b^*.$$

$$b^* a\ b^* (ab^*ab^*)^*.$$

EXAMPLE

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

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

Example: Find a regular expression for the language

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

Even though this looks similar to the previous example, 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 $1\cdots101\cdots1$*, 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*

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

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

Note that this language is the complement of the language in the previous example. However, the regular expressions are not very similar and do not suggest clearly the close relationship between the languages.