

# 词法分析

## (2. 正则表达式)

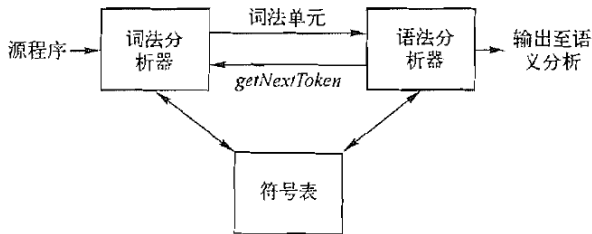
魏恒峰

hfwei@nju.edu.cn

2023 年 03 月 08 日 (周三)



**输入:** 程序文本/字符串  $s$  + **词法单元 (token) 的规约**



**输出:** 词法单元流



```
LPAREN : '(' ;
```

```
RPAREN : ')' ;
```

```
ID : (LETTER | '_' ) WORD* ;
```

```
INT : '0' | ([1-9] DIGIT*) ;
```

```
FLOAT : INT '.' DIGIT*
```

```
      | '.' DIGIT+
```

```
      ;
```

```
WS : [ \t\r\n]+ -> skip ;
```

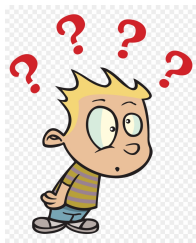
```
//SL_COMMENT : '//' .*? '\n' -> skip ;
```

```
SL_COMMENT2 : '//' ~[\n]* '\n' -> skip;
```

```
DOC_COMMENT : '/*' .*? '*/' -> skip ;
```

```
ML_COMMENT : '/*' .*? '*/' -> skip ;
```

## 词法单元的规约



词法单元	非正式描述	词素示例
if	字符 i, f	if
else	字符 e, l, s, e	else
comparison	< 或 > 或 <= 或 >= 或 == 或 !=	<=, !=
id	字母开头的字母 / 数字串	pi, score, D2
number	任何数字常量	3.14159, 0, 6.02e23
literal	在两个 "之间, 除" 以外的任何字符	"core dumped"

我们需要词法单元的**形式化**规约

id: 字母开头的字母/数字串

id 定义了一个集合, 我们称之为**语言 (Language)**

id: 字母开头的字母/数字串

id 定义了一个集合, 我们称之为**语言 (Language)**

它使用了字母与数字等符号集合, 我们称之为**字母表 (Alphabet)**

id: 字母开头的字母/数字串

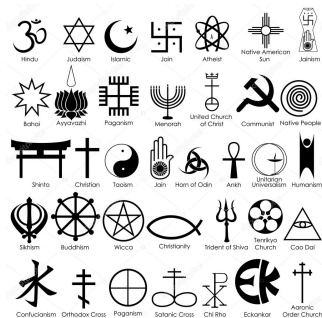
id 定义了一个集合, 我们称之为**语言 (Language)**

它使用了字母与数字等符号集合, 我们称之为**字母表 (Alphabet)**

该语言中的每个元素 (即, 标识符) 称为**串 (String)**

## Definition (字母表)

**字母表**  $\Sigma$  是一个有限的符号集合。





## Definition (串)

字母表  $\Sigma$  上的**串** ( $s$ ) 是由  $\Sigma$  中符号构成的一个**有穷**序列。

$\epsilon$

空串 :  $|\epsilon| = 0$

Definition (串上的“连接”运算)

$$x = \text{dog}, y = \text{house} \quad xy = \text{doghouse}$$

$$s\epsilon = \epsilon s = s$$

Definition (串上的“连接”运算)

$$x = \text{dog}, y = \text{house} \quad xy = \text{doghouse}$$

$$s\epsilon = \epsilon s = s$$

Definition (串上的“指数”运算)

$$s^0 \triangleq \epsilon$$

$$s^i \triangleq ss^{i-1}, i > 0$$

## Definition (语言)

**语言**是给定字母表  $\Sigma$  上一个任意的**可数的**串集合。

$\emptyset$

$\{\epsilon\}$

## Definition (语言)

**语言**是给定字母表  $\Sigma$  上一个任意的**可数的**串集合。

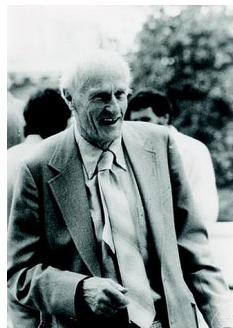
$$\emptyset$$
$$\{\epsilon\}$$
$$\text{id} : \{a, b, c, a1, a2, \dots\}$$
$$\text{ws} : \{\text{blank}, \text{tab}, \text{newline}\}$$
$$\text{if} : \{\text{if}\}$$

# 语言是串的集合

因此, 我们可以通过集合操作**构造**新的语言。

运算	定义和表示
$L$ 和 $M$ 的并	$L \cup M = \{s \mid s \text{ 属于 } L \text{ 或者 } s \text{ 属于 } M\}$
$L$ 和 $M$ 的连接	$LM = \{st \mid s \text{ 属于 } L \text{ 且 } t \text{ 属于 } M\}$
$L$ 的 Kleene 闭包	$L^* = \bigcup_{i=0}^{\infty} L^i$
$L$ 的正闭包	$L^+ = \bigcup_{i=1}^{\infty} L^i$

$L^*$  ( $L^+$ ) 允许我们构造**无穷**集合



Stephen Kleene  
(1909 ~ 1994)

$$L = \{A, B, \dots, Z, a, b, \dots, z\}$$

$$D = \{0, 1, \dots, 9\}$$

运算	定义和表示
$L$ 和 $M$ 的并	$L \cup M = \{s \mid s \text{ 属于 } L \text{ 或者 } s \text{ 属于 } M\}$
$L$ 和 $M$ 的连接	$LM = \{st \mid s \text{ 属于 } L \text{ 且 } t \text{ 属于 } M\}$
$L$ 的Kleene 闭包	$L^* = \bigcup_{i=0}^{\infty} L^i$
$L$ 的正闭包	$L^+ = \bigcup_{i=1}^{\infty} L^i$



$$L = \{A, B, \dots, Z, a, b, \dots, z\}$$

$$D = \{0, 1, \dots, 9\}$$

运算	定义和表示
$L$ 和 $M$ 的并	$L \cup M = \{s \mid s \text{ 属于 } L \text{ 或者 } s \text{ 属于 } M\}$
$L$ 和 $M$ 的连接	$LM = \{st \mid s \text{ 属于 } L \text{ 且 } t \text{ 属于 } M\}$
$L$ 的Kleene 闭包	$L^* = \bigcup_{i=0}^{\infty} L^i$
$L$ 的正闭包	$L^+ = \bigcup_{i=1}^{\infty} L^i$

$$L \cup D \quad LD \quad L^4 \quad L^* \quad D^+ \quad L(L \cup D)^*$$

**id** :  $L(L \cup D)^*$

该如何告诉 ANTLR v4 : 这个集合就是 **id** 呢?

$\text{id} : L(L \cup D)^*$

该如何告诉 ANTLR v4 : 这个集合就是 **id** 呢?



下面向大家隆重介绍简洁、优雅、强大的**正则表达式**

每个正则表达式  $r$  对应一个正则语言  $L(r)$



正则表达式是**语法**, 正则语言是**语义**

## Definition (正则表达式)

给定字母表  $\Sigma$ ,  $\Sigma$  上的正则表达式由且仅由以下规则定义:

- (1)  $\epsilon$  是正则表达式;
- (2)  $\forall a \in \Sigma$ ,  $a$  是正则表达式;
- (3) 如果  $r$  是正则表达式, 则  $(r)$  是正则表达式;
- (4) 如果  $r$  与  $s$  是正则表达式, 则  $r|s$ ,  $rs$ ,  $r^*$  也是正则表达式。

运算优先级:  $() \succ * \succ \text{连接} \succ |$

$$(a)|((b)^*(c)) \equiv a|b^*c$$

每个正则表达式  $r$  对应一个正则语言  $L(r)$

Definition (正则表达式对应的正则语言)

$$L(\epsilon) = \{\epsilon\} \quad (1)$$

$$L(a) = \{a\}, \forall a \in \Sigma \quad (2)$$

$$L((r)) = L(r) \quad (3)$$

$$L(r|s) = L(r) \cup L(s) \quad L(rs) = L(r)L(s) \quad L(r^*) = (L(r))^* \quad (4)$$

$$\Sigma = \{a, b\}$$

$$L(a|b) = \{a, b\}$$

$$\Sigma = \{a, b\}$$

$$L(a|b) = \{a, b\}$$

$$L((a|b)(a|b))$$



$$\Sigma = \{a, b\}$$

$$L(a|b) = \{a, b\}$$

$$L((a|b)(a|b))$$

$$L(a^*)$$

$$\Sigma = \{a, b\}$$

$$L(a|b) = \{a, b\}$$

$$L((a|b)(a|b))$$

$$L(a^*)$$

$$L((a|b)^*)$$

$$\Sigma = \{a, b\}$$

$$L(a|b) = \{a, b\}$$

$$L((a|b)(a|b))$$

$$L(a^*)$$

$$L((a|b)^*)$$

$$L(a|a^*b)$$

So  
EASY

表达式	匹配	例子
<code>c</code>	单个非运算符字符 $c$	<code>a</code>
<code>\c</code>	字符 $c$ 的字面值	<code>\*</code>
<code>"s"</code>	串 $s$ 的字面值	<code>"**"</code>
<code>.</code>	除换行符以外的任何字符	<code>a.*b</code>
<code>^</code>	一行的开始	<code>^abc</code>
<code>\$</code>	行的结尾	<code>abc\$</code>
<code>[s]</code>	字符串 $s$ 中的任何一个字符	<code>[abc]</code>
<code>[^s]</code>	不在串 $s$ 中的任何一个字符	<code>[^abc]</code>
<code>r*</code>	和 $r$ 匹配的零个或多个串连接成的串	<code>a*</code>
<code>r+</code>	和 $r$ 匹配的一个或多个串连接成的串	<code>a+</code>
<code>r?</code>	零个或一个 $r$	<code>a?</code>
<code>r{m,n}</code>	最少 $m$ 个, 最多 $n$ 个 $r$ 的重复出现	<code>a{1,5}</code>
<code>r<sub>1</sub>r<sub>2</sub></code>	$r_1$ 后加上 $r_2$	<code>ab</code>
<code>r<sub>1</sub>   r<sub>2</sub></code>	$r_1$ 或 $r_2$	<code>a b</code>
<code>(r)</code>	与 $r$ 相同	<code>(a b)</code>
<code>r<sub>1</sub>/r<sub>2</sub></code>	后面跟有 $r_2$ 时的 $r_1$	<code>abc/123</code>

表达式	匹配	例子
<code>c</code>	单个非运算符字符 $c$	<code>a</code>
<code>\c</code>	字符 $c$ 的字面值	<code>\*</code>
<code>"s"</code>	串 $s$ 的字面值	<code>"**"</code>
<code>.</code>	除换行符以外的任何字符	<code>a.*b</code>
<code>^</code>	一行的开始	<code>^abc</code>
<code>\$</code>	行的结尾	<code>abc\$</code>
<code>[s]</code>	字符串 $s$ 中的任何一个字符	<code>[abc]</code>
<code>[^s]</code>	不在串 $s$ 中的任何一个字符	<code>[^abc]</code>
<code>r*</code>	和 $r$ 匹配的零个或多个串连接成的串	<code>a*</code>
<code>r+</code>	和 $r$ 匹配的一个或多个串连接成的串	<code>a+</code>
<code>r?</code>	零个或一个 $r$	<code>a?</code>
<code>r{m,n}</code>	最少 $m$ 个, 最多 $n$ 个 $r$ 的重复出现	<code>a{1,5}</code>
<code>r<sub>1</sub>r<sub>2</sub></code>	$r_1$ 后加上 $r_2$	<code>ab</code>
<code>r<sub>1</sub>   r<sub>2</sub></code>	$r_1$ 或 $r_2$	<code>a b</code>
<code>(r)</code>	与 $r$ 相同	<code>(a b)</code>
<code>r<sub>1</sub>/r<sub>2</sub></code>	后面跟有 $r_2$ 时的 $r_1$	<code>abc/123</code>

`[0 - 9]` `[a - zA - Z]`

$\wedge$   $\$$

# 正则表达式简记法

Vim ↕	Java ↕	ASCII ↕	Description ↕
	\p{ASCII}	[\x00-\x7F]	ASCII characters
	\p{Alnum}	[A-Za-z0-9]	Alphanumeric characters
\w	\w	[A-Za-z0-9_]	Alphanumeric characters plus "_"
\W	\W	^[A-Za-z0-9_]	Non-word characters
\a	\p{Alpha}	[A-Za-z]	Alphabetic characters
\s	\p{Blank}	[ \t]	Space and tab
\< \>	\b	(?<=\W) (?=\w)   (?<=\w) (?=\W)	Word boundaries
	\B	(?<=\W) (?=\W)   (?<=\w) (?=\w)	Non-word boundaries
	\p{Cntrl}	[\x00-\x1F\x7F]	Control characters
\d	\p{Digit} or \d	[0-9]	Digits
\D	\D	^[0-9]	Non-digits
	\p{Graph}	[\x21-\x7E]	Visible characters
\l	\p{Lower}	[a-z]	Lowercase letters
\p	\p{Print}	[\x20-\x7E]	Visible characters and the space character
	\p{Punct}	[ ] [ ! " # \$ % & ' ( ) * + , . / : ; < = > ? @ \ ^ _ ` {   } ~ - ]	Punctuation characters
\s	\p{Space} or \s	[ \t\r\n\v\f]	Whitespace characters
\S	\S	^[ \t\r\n\v\f]	Non-whitespace characters
\u	\p{Upper}	[A-Z]	Uppercase letters
\x	\p{XDigit}	[A-Fa-f0-9]	Hexadecimal digits







$$(0|(1(01^*0)^*1))^*$$



REGULAR EXPRESSION v1

/ ^((0|(1(01\*0)\*1))\*\$

TEST STRING

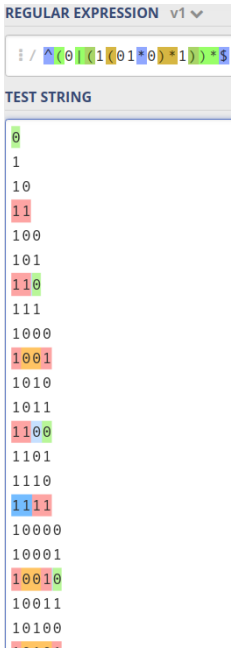
0  
1  
10  
11  
100  
101  
110  
111  
1000  
1001  
1010  
1011  
1100  
1101  
1110  
1111  
10000  
10001  
10010  
10011  
10100

<https://regex101.com/r/ED4qgC/1>

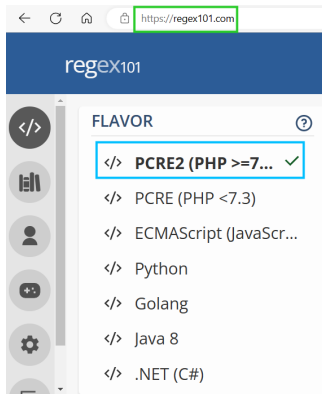
regex101

FLAVOR

- </> PCRE2 (PHP >=7... ✓
- </> PCRE (PHP <7.3)
- </> ECMAScript (JavaScr...
- </> Python
- </> Golang
- </> Java 8
- </> .NET (C#)



<https://regex101.com/r/ED4qgC/1>



3 的倍数 (二进制表示)



Let Us Ask ChatGPT Again

```
body {  
  background-color: #fefbd8;  
}  
  
h1 {  
  background-color: #0000ff;  
}  
  
div {  
  background-color: #d0f4e6;  
}  
  
span {  
  background-color: #f08970;  
}
```

<https://regex101.com/r/Z6IWGL/1> (regex/bg-color)

3/8/23

3-8-2023

2/2/2←

03-08-2023

<https://regex101.com/r/X1Q6DG/1> (regex/date)

1001: • \$496.80↵  
1002: • \$1290.89↵  
1003: • \$26.43↵  
1004: • 613.42↵  
1005: • 7.61↵  
1006: • \$414.90↵  
1007: • \$25.00

<https://regex101.com/r/dSKeBr/1> (regex/dollar)

The cat scattered his food all over the room.

<https://regex101.com/r/K5MCMZ/1> (regex/cat)



```
<body>␣
•• <H1>Welcome•to•my•Homepage</H1>␣
•• Content•is•divided•into•two•sections:<br/>␣
•• <h2>SQL</h2>␣
•• Information•about•SQL.␣
•• <h2>RegEx</h2>␣
•• Information•about•Regular•Expressions.␣
•• <h3>This•is•not•a•valid•HTML</h4>␣
</body>
```

<https://regex101.com/r/2ZSx0a/1> (regex/html-head)

<body>↵

••<H1>Welcome to my Homepage</H1>↵

••Content is divided into two sections:<br/>

••<h2>SQL</h2>↵

••Information about SQL.↵

••<h2>Regex</h2>↵

••Information about Regular Expressions.↵

••<h3>This is not a valid HTML</h4>↵

</body>

<https://regex101.com/r/mFt85G/1>  
(regex/html-head-lookaround)

LEARNING REGULAR EXPRESSIONS

# 正则表达式 必知必会 (修订版)

[美] 本·福塔◎著 门佳 杨涛等◎译

紧贴实战需求，化繁为简，高效解决编程难题



中国工信出版集团



人民邮电出版社  
POSTS & TELECOM PRESS

## (Ir)Regular expressions

"Some people, when confronted with a problem, think 'I know, I'll use regular expressions.' Now they have two problems."

— Jamie Zawinski

Thank  
You!



Office 926

hfwei@nju.edu.cn