

编译原理笔记

陈鸿峥

2020.04*

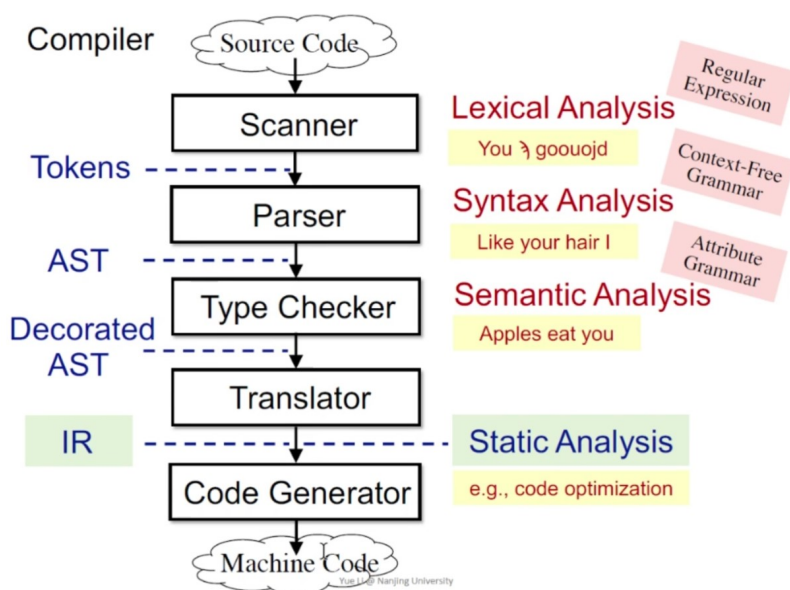
目录

1 简介	1
2 词法分析	2
2.1 基本定义	2
2.2 正则表达式	2
2.3 有限自动机	3

本课程采用书目Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, *Compilers: Principles, Techniques & Tools (2nd ed)*, 即大名鼎鼎的龙书。

1 简介

编译器的几个阶段如下，前端包括词法(lexical)、语法(syntax)、语义(semantic)分析，中端IR生成、优化，后端代码生成。



2 词法分析

分离词法分析和语法分析可以简化这两个任务，同时提升编译器的性能与兼容性。

2.1 基本定义

定义 1. 令牌(*token*)是一个令牌名字与可选属性值构成的对；模式(*pattern*)描述了每个词素(*lexeme*)要遵循什么规则；而词素（最小意义单位）则是源程序中一连串满足模式的字母，作为令牌的实例化。

例 1. 考虑C语句

```
printf("Total = %d\n", score);
```

其中printf和score是匹配(*match*)上令牌*id*模式的词素，而"Total = %d\n"是匹配上字面值*literal*的词素。

简单来讲，令牌是一个更大的概念，是同类词素的集合。比如一个令牌*comparison*的样例词素可以有<=和!=。

定义 2 (字母表与语言). 字母表(*alphabet*) Σ 是有限符号(*symbol*)的集合，如ASCII就是一个字母表。字符串(*string*) s 是从字母表中抽取的有限符号的序列， $|s|$ 为字符串长度， ϵ 为空串。语言(*language*)是字符串的可数集合。

例 2. 字母表 $\Sigma = \{0, 1\}$ ，则 $\{001, 1001\}$ 和 $\{\}$ 都是定义在 Σ 上的语言。

定义 3 (字符串术语). 前缀(*prefix*)和后缀(*suffix*)都可以包括 ϵ 。子串(*substring*)可通过删除任意前缀和任意后缀（包括零个）获得。真(*proper*)子串则不包含 ϵ 。子序列(*subsequence*)是删除零个或多个不一定连续的字母得到的字符串。

语言是一种集合，故集合运算也适用于语言。

并集(union)	$L \cup M$
连接(concatenation)/交集	LM
柯林闭包(Kleene closure)	$L^* = \bigcup_{i=0}^{\infty} L^i$
正闭包(positive)	$L^+ = \bigcup_{i=1}^{\infty} L^i$

2.2 正则表达式

定义 4 (正则表达式(regular expression, regex)). 正则表达式 r 定义了语言 $L(r)$ ，以递归形式定义：

1. 奠基：

- ϵ 是正则表达式，即 $L(\epsilon) = \{\epsilon\}$
- $a \in \Sigma$ 是正则表达式，即 $L(a) = \{a\}$ （这里用斜体代表符号，粗体代表符号对应的正则表达式）

2. 推论(*induction*): 若 r 和 s 都是正则表达式给出了语言 $L(r)$ 和 $L(s)$ ，则

- $(r)|(s)$ 是正则表达式, 表示 $L(r) \cup L(s)$
- $(r)(s)$ 是正则表达式, 表示 $L(r)L(s)$
- $(r)^*$ 是正则表达式, 表示 $(L(r))^*$
- (r) 是正则表达式, 表示 $L(r)$

正则表达式表示的语言叫做正规集。

有以下运算规定:

- 一元运算符 $*$ 有最高优先级, 左结合
- 连接优先级次之, 左结合
- $|$ 优先级最低, 左结合

定义 5 (正则定义). $d_i \rightarrow r_i$, 其中 d_i 都是名字, 且各不相同。每个 r_i 是 $\Sigma \cup \{d_1, \dots, d_{i-1}\}$ 中符号上的正则表达式。

例 3. 比如 C 语言的标识符可记为

$$letter_ \rightarrow A|B|\dots|Z|a|b|\dots|z|_$$

$$digit \rightarrow 0|1|\dots|9$$

$$id \rightarrow letter_ (letter_ | digit)^*$$

正则表达式的拓展¹:

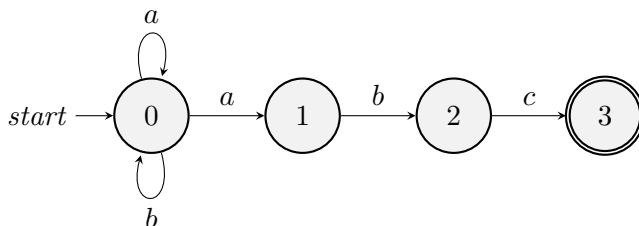
- r^+ 代表一个或多个
- $r^?$ 代表零或一个
- $[a-z]$ 字母类

2.3 有限自动机

2.3.1 确定性/非确定性有限自动机

确定有限自动机(DFA)不可对 ϵ 进行移动, 而且对于每一状态 s , 输入符号 a , 只有唯一一条出边标记为 a ; 而非确定性有限自动机(NFA)可能有多种转换路径。有限状态集 S , 状态 $s_0 \in S$ 为初始状态(start/initial), $F \subset S$ 为终止状态(accepting/final)。

例 4. 识别语言 $L((a|b)^*abb)$, 下面为一个 NFA



¹更多可参见[Regex101](#)

Algorithm 1 基于DFA的识别算法

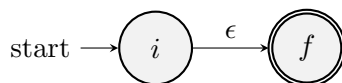
```
1:  $s = s_0$ 
2:  $c = nextChar()$ 
3: while ( $c \neq eof$ ) do
4:    $s = move(s, c)$ 
5:    $c = nextChar()$ 
6: if  $s \in F$  then
7:   return “yes”
8: elsereturn “no”
```

时间复杂度为 $O(|str|)$ 。

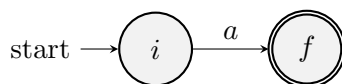
2.3.2 正则表达式转NFA

1. 奠基

- 对于表达式 ϵ , 构建NFA

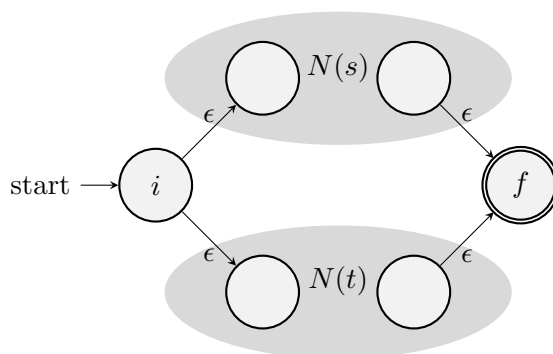


- 对于任意子表达式 $a \in \Sigma$, 构建NFA

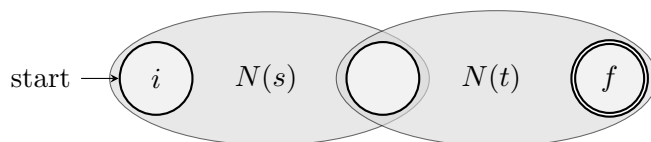


2. 推论

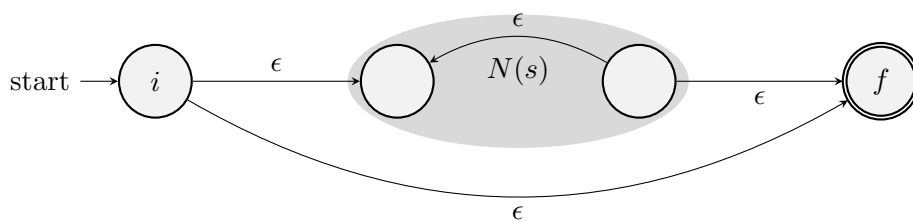
- $r = s|t$, 取并集



- $r = st$, 取连接



- $r = s^*$, Kleene闭包



2.3.3 NFA转DFA

定义 6 (ϵ 闭包及 $move$). ϵ 闭包是可通过NFA的 ϵ 边转换的状态。 $move(T, a)$ 为状态 $s \in T$ 通过输入符号 a 可到达的新的状态。

Algorithm 2 子集构造

Require: NFA N

Ensure: DFA D (与 N 接受相同的语言)

- 1: $\epsilon - closure(s_0)$ 是 $Dstates$ 的唯一状态, 且未被标记(unmarked)
 - 2: **while** 在 $Dstates$ 中还有未被标记的状态 T **do**
 - 3: 标记 T
 - 4: **for** 每一个输入符号 a **do**
 - 5: $U = \epsilon - closure(move(T, a))$
 - 6: **if** $U \notin Dstates$ **then**
 - 7: 将 U 作为未标记的状态加入 $Dstates$
 - 8: $Dtran[T, a] = U$
-