Atitit.词法分析的理论原理 part2

[1.  转换图 1](#_Toc23709)

[1.1. 转换图是由程序流程图改进而成的。同样，转换图也可以等价地转换为程序流程图 3](#_Toc10648)

[1.2. 2.2.3  构造词法分析器（2）流程程序2-1虽然只有26行，却是词法分析器的核心 4](#_Toc5075)

[1.3. 单词存储形式就是三元组（单词ID，单词备注，单词行号）。 4](#_Toc3153)

[1.4. 单词流是如何传递给语法分析器的。 5](#_Toc15674)

[1.5. 词法定义 5](#_Toc23432)

[1.6. 词法分析器主要包括：构造转换图与转换表、设计词法分析器算法。 6](#_Toc11507)

[1.7. 超前搜索几个字符与词法定义有关，有些设计不精良的语言可能需要超前搜索三个甚至四个字符才能正确识别单词 7](#_Toc13667)

# ****转换图****

从图2-2中，不难发现，其中只有"搜索指针后移一个字符"一种处理动作（方框）。那么，读者不妨想象一下，词法分析器是否就只有这种处理动作呢？仔细分析手工识别单词的过程后，就可以发现事实确实如此。既然词法分析器的流程中条件判断（菱形框）比较复杂，而处理动作非常单一，因此，可以将普通流程图改造成一种专门用于描述条件判断的流程图。具体改造步骤如下：

1）把图2-2中所有上、下菱形（判断）之间的箭头用圆表示。

2）把图2-2中所有的菱形直接用箭头线表示，箭头上写上原菱形的判断成立与否的条件，即可得到图2-3。

|  |
| --- |
| [IMG_256](http://images.51cto.com/files/uploadimg/20120601/131150577.jpg) |
| 图2-3  识别Pascal标识符的状态转换图 |

这里省略了出错处理，由于词法分析器的出错处理可以统一完成，所以无需重复考虑。

那么，将图2-2改造成图2-3的意义是什么呢？实际上，主要针对词法分析器的流程特点，突出了流程中最为复杂的部分，省略或简化相对简单的部分，这样的改造便于更直观地分析问题的本质。这种分析问题的方法在软件系统分析与设计过程中较为常用，读者应该善于改进一些已有的工具，使之更有利于分析问题。

图2-3 对于词法分析器设计具有非常重要的作用。在形式语言学中将这种图称为转换图（transition diagrams），亦称为状态转换图。

转换图是一个有向图。在转换图中，圆表示状态，状态之间用箭弧连接。箭弧上的标记表示在箭弧始结点的状态下读入给定字符或字符集时，当前状态转到箭弧终结点所示状态。例如，图2-3在开始状态下，若输入字母，则转换到状态1。一张转换图只包含有限个状态（即有限个结点），其中，有一个是初始状态（图2-3中的"开始"状态），而且必须至少有一个终止状态（用双圈表示，读者可以将终止状态理解为可终止状态，而其他状态即为不可终止状态）。那么，如何运用转换图完成单词的识别呢？实际上，识别一个单词就是从初始状态经过数次转换到达终止状态的过程。如果某一个字符串的识别过程不能到达终止状态，说明输入字符串不能被该转换图接收。

下面，应用转换图来分析一个复杂的实例。在Pascal词法分析器中最复杂的就是识别实型常量的逻辑流程。Pascal的实型常量有两种形式：普通小数形式、科学记数法的小数形式，例如，2.34、23.4E32、46.1e-43、30.e12等都是合法的实型常量，而诸如 .32、31.3e、32.3E13.2等形式都是非法的实型常量。读者可以参考图2-4。

|  |
| --- |
| [IMG_257](http://images.51cto.com/files/uploadimg/20120601/131212335.jpg) |
| 图2-4  识别Pascal实型常量的转换图 |

读者可能会有疑惑，为什么图2-4没有考虑实型常量的+/-号？实际上，在词法分析时，只能将"-"识别成运算符，但无法确定其到底是用作负号还是减号。这个工作将在语法分析阶段完成。因此，词法分析器无法确定是否将"-"直接与后续常量组合。仅从识别实型常量的角度而言，图2-4已经足够完整了。但从实际编译器设计的角度而言，可能还需作少量改进。这里，值得注意的是标准Pascal中的数组声明形式，例如：

下面，再来看看如何应用图2-4识别字符串"92.3E1"。这是一个合法的实数常量表示形式。首先，从开始箭头进入状态1。在状态1时，读取到数字即转入状态2。在状态2时，读取第二个字符"2"，根据圆弧箭头指示，状态2遇到数字时仍然停留在状态2。在状态2时，读取第三个字符"."，状态2即转入状态3。在状态3时，读取第四个字符"3"，状态3遇到数字时仍然停留在状态3。在状态3时，读取第五个字符"E"，状态3即转入状态4。在状态4时，读取第六个字符"1"，状态4转换为状态6。根据超前搜索识别原则，词法分析器继续读取下一个字符，且下一个字符必定不是数字。在状态6时，读取一个非数字字符即转入状态7。至此，状态转换停留在状态7上，状态7是终止状态（即双圈状态），故判定输入字符串合法有效。表2-3给出几个实例分析，请读者仔细理解。

表2-3  识别实型常量的实例分析

|  |  |  |
| --- | --- | --- |
| 输入字符串 | 状 态 转 换 | 识 别 结 果 |
| 1234 | 1 , 2 , 2 , 2 , 2 , 7 | 有效（到达终止态7） |
| 192.32 | 1 , 2 , 2 , 2 , 3 , 3 , 7 | 有效（到达终止态7） |
| 192a2 | 1 , 2 , 2 , 2 | 无效（到达非终止态2） |
| A12s | 1 | 无效（到达非终止态1） |
| 1.104E19 | 1 , 2 , 3 , 3 , 3 , 3 , 4 , 6 , 6 , 7 | 有效（到达终止态7） |
| 12E12.98 |  |  |
| -12.12 |  |  |
| +32.E23 |  |  |
| 43.1E3.9 |  |  |

请读者自行补充表2-3中空白区域，以加深对转换图使用方法的理解。只有理解了转换图与普通流程图之间的联系，才能进行词法分析器的设计。Neo Pascal语言的完整转换图将在2.3节详细分析。

作者::  ★(attilax)>>>  绰号:老哇的爪子 （ 全名：：Attilax Akbar Al Rapanui 阿提拉克斯 阿克巴 阿尔 拉帕努伊 ） 汉字名：艾龙，  EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

## 转换图是由程序流程图改进而成的。同样，转换图也可以等价地转换为程序流程图

完整的转换图就是词法分析器的程序流程图，是实现词法分析器的依据。早期的编译器都是采用这种方式手工编码实现的。这种实现方式对编译器设计者的编程技巧要求非常高。如果非常机械地完全按照转换图的描述编码，所得到的词法分析器的执行效率可能比较低，必须运用编程技巧使词法分析器尽可能精悍高效。因此，这种编码实现方式存在一个严重的不足：程序的耦合度高。当词法定义发生改变时，词法分析器的改动将非常大，甚至是颠覆性的。然而，为什么早期的编译器大多采用这种方式编码实现呢？最主要的原因就是这种方式实现的词法分析器运行时所需的存储空间的代价相对较小

显然，转换图的形式是不利于输入计算机的，众所周知，最便于程序处理的数据结构就是表格。如果能将转换图转化为表格或者类似的数据结构，那么，词法分析器就可以识别并处理转换图了。事实上，将转换图等价转换为二维表格的过程可以参照以下四个步骤：

## ****2.2.3  构造词法分析器（2）流程****程序2-1虽然只有26行，却是词法分析器的核心

第11行：查询当前状态下读取当前字符时的转换状态，这是程序的核心部分。

第12行：如果查询得到的转换状态为-1，表示当前状态下不允许读取当前字符。

第13行：置出错标志bTag为true。

第15行：调用IsTerminal函数判断转换状态是否为终止状态。

第16行：将当前状态设为查询得到的转换状态。

第17行：当前状态下读取当前字符时，转换状态为终止状态，即得到一个单词。

第19行：由于超前搜索识别，必须将当前读取位置回退一个字符。

第20行：将单词缓存区回退一个字符。

第21行：调用ProcessToken处理单词。

第22行：将单词缓存区清空。

第23行：当前状态置为0。

## 单词存储形式就是三元组（单词ID，单词备注，单词行号）。

词法分析器识别得到的单词流是语法分析器的输入，那么，词法分析器需要提供哪些信息给语法分析器呢？以什么方式传递给语法分析器呢？这些问题是需要设计者回答的。

在编译器设计中，最常见的单词存储形式就是三元组（单词ID，单词备注，单词行号）。

单词行号一般包含两部分信息：单词所在的源程序文件名、单词在源程序文件中的行号。这些信息主要用于出错处理，可以提供错误的位置信息，便于用户定位修改源程序。行号信息

## 单词流是如何传递给语法分析器的。

这主要取决于词法分析是作为一个独立阶段还是将它设计为独立的一遍。词法分析作为一个独立阶段就是指将词法分析器作为一个函数，由语法分析器调用。每次调用词法分析器只识别一个单词，然后将单词直接传递给语法分析器处理。整个过程由语法分析器控制，在语法分析过程中，进行单词识别。词法分析作为独立的一遍就是指词法分析器将对整个源程序文件扫描一次，识别出所有的单词，然后将源程序以单词流的方式传递给语法分析器处理。这两种方式各有利弊，且各有成功的应用案例

## ****词法定义****

设计词法分析器之前，必须明确程序设计语言的词法定义。词法定义是一门程序设计语言的必要部分。同时，词法定义也直接关系着词法分析器的复杂程度，甚至并非所有的词法定义都可以实现。例如，假设C语言的词法定义规定标识符中可以出现"\*"，则这样的词法定义是很难实现的。

表2-5详细描述了Neo Pascal的词法定义。

表2-5  Neo Pascal词法定义

|  |  |
| --- | --- |
| 标识符 | 以字母开头，后跟字母与数字的任意组合的字符串  ID：001 |
| 运算符 | Neo Pascal一共包含13个运算符，分别是：   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | ID | 单词 | ID | 单词 | ID | 单词 | ID | 单词 | ID | 单词 | | 016 | = | 017 | := | 018 | < | 019 | <> | 020 | <= | | 021 | > | 022 | >= | 023 | + | 024 | - | 025 | \* | | 026 | / | 027 | ^ | 028 | @ |  |  |  |  | |

（续）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 界符 | Neo Pascal一共包含9个界符，分别是：   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | ID | 单词 | ID | 单词 | ID | 单词 | ID | 单词 | ID | 单词 | | 007 | . | 008 | ( | 009 | ) | 010 | ; | 011 | , | | 012 | : | 013 | .. | 014 | [ | 015 | ] |  |  | |
| 关键字 | 关键字是特殊的标识符，必须符合标识符的定义。Neo Pascal一共包含54个关键字，分别是：   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | ID | 单词 | ID | 单词 | ID | 单词 | ID | 单词 | ID | 单词 | | 040 | ASM | 041 | AND | 042 | ARRAY | 043 | BEGIN | 044 | BOOLEAN | | 045 | BYTE | 046 | BREAK | 047 | CARDINAL | 048 | CASE | 049 | CHAR | | 050 | CONST | 051 | CONTINUE | 052 | DIV | 053 | DO | 054 | DOWNTO | | 055 | ELSE | 056 | END | 057 | FILE | 058 | FOR | 059 | FORWARD | | 060 | FUNCTION | 061 | GOTO | 062 | IF | 063 | IN | 064 | INTEGER | | 065 | LABEL | 066 | LONGWORD | 067 | MOD | 068 | NIL | 069 | NOT | | 070 | OF | 071 | OR | 072 | PROCEDURE | 073 | PROGRAM | 074 | REAL | | 075 | RECORD | 076 | REPEAT | 077 | SET | 078 | SHL | 079 | SHORTINT | | 080 | SHR | 081 | SINGLE | 082 | SMALLINT | 083 | STRING | 084 | THEN | | 085 | TO | 086 | TYPE | 087 | UNTIL | 088 | USES | 089 | VAR | | 090 | WITH | 091 | WHILE | 092 | WORD | 093 | XOR |  |  | |
| 常  量 | 字符串常量，                     ID：002。 例：'I am a boy.'  整数常量，                       ID：003。 例：324、32  普通形式实型常量，               ID：004。 例：43.53、62.034  带+/-号的科学记数形式实型常量，  ID：005。 例：43.12E-23  不带+/-号的科学记数形式实型常量，ID：006。 例：41.09E12 |

## 词法分析器主要包括：构造转换图与转换表、设计词法分析器算法。

词法分析器的核心就是依据转换图识别单词。不过，事实并非完全如此。由于某些程序设计语言的词法定义缘故，仅仅依据程序2-1的算法是不足以完成词法分析的

## 超前搜索几个字符与词法定义有关，有些设计不精良的语言可能需要超前搜索三个甚至四个字符才能正确识别单词

。反复回溯搜索会大大降低词法分析器的执行效率，故在设计一门程序设计语言时，应尽可能做到只需超前搜索一个字符就可以完成所有单词的识别。

参考

2.2.2 转换图 - 51CTO.COM.html