编译原理研究性项目

设计个人信息文法

2020 级弘毅 4 班-2020300004062-张姝怡
2020 级弘毅 3 班-2020302111411-解馨迪

分工介绍：
张姝怡：确定选题以及个人信息形式，撰写报告第一，三部分。完成代码，并进行实验。
解馨迪：撰写报告第二，四部分。修改代码，完善报告。

一、题目背景与语法描述

(1) 问题的背景描述
①项目：个人信息收集
②用途：进行大型重要场所/会议的入场身份确认。参会人员需要填写入会报名表。通过识别表格中的内容，可以将参会人员信息转化为二维码。将包含个人信息的二维码打印在入场牌上，工作人员通过扫描二维码即可确认是否有参会资格以及参会人基本信息。如果入场牌遗失，也能通过扫描二维码，获取失主的相关信息和联系方式。上述方法有利于大型重要会议的人员管理和秩序维持。下面，我们将目标聚焦于为“高校交流会”收集参会人员信息。当然，相同的方式也可以应用于各种不同的现实场景。
③要求：
身份证号（Id）和电话号码（Phone）不能重复且格式正确；

(2) 被处理对象的文法定义
①非终结符：
{info,lines,line,prompt,filling,IdNUmber,Tag,PhoneNumber,Date,Name,Id,Unit,Occupation,Phone,Time }
②终结符：{a, b, c, d, e, f……0, 1, 2, 3 …… 8, 9, -, A, B, C, D …… Z, . , :}
③生成式 P：
Info→<Info><Lines>
Lines→<Line><Lines>
Line→<Number><.><Prompt><:><Filling>
Prompt→<Name>|<Id>|<Unit>|<Occupation>|<Phone>|<Time>
Filling→<IdNumber>|<Tag>|<PhoneNumber>|<Date>
IdNumber→$\{< Number >\}^{17}$(<Number>|<X>)
Tag→<Letter><Tag>|<Letter>
Unit→<Letter><Unit>|<Letter>
Occupation→<Letter><Occupation>|<Letter>
PhoneNumber→$\{< Number >\}^{11}$
Number→<0>|<1>|<2>|……<9>
Letter→<a>|<b>|<c>|<d>|……<z>|<A>|<B>|……|<Z>
Date→<Number><Number><Number><Number><-><Number><Number><-><Number><Number>

④开始符号 S：Info

⑤文法：G[Info]: {{ Info, Line, Prompt, Name, Id, Unit, Occupation, Phone, Time, Letter, Number }，{ a, b, c, d, e, f……0, 1, 2, 3 …… 8, 9,-,A,B,C,D……,Z, . , :}，P，Info}

⑥非终结符含义的简要介绍：Info-个人信息，Name-姓名，Id-身份证号，Unit-所属单位，Occupation-职业，Phone-电话号码，Time-有效参会时间(年-月-日)

(3) 给出“单词”列表

提示符:

Info/Name/Id/Unit/Occupation/Phone/Time

标识符:

Mike/Mary……（姓名）

Wuhan University/WHU/Harvard/UCL/……（所属单位）

Principle/professor/teacher/student/……（职业）

常数:

1/2/3/4324/420106222/……

连接符:

-/:/.

(4) 给出被处理对象的“结构”举例（可以用语法树形式）

现有一个参会人员的信息为，如图 1:

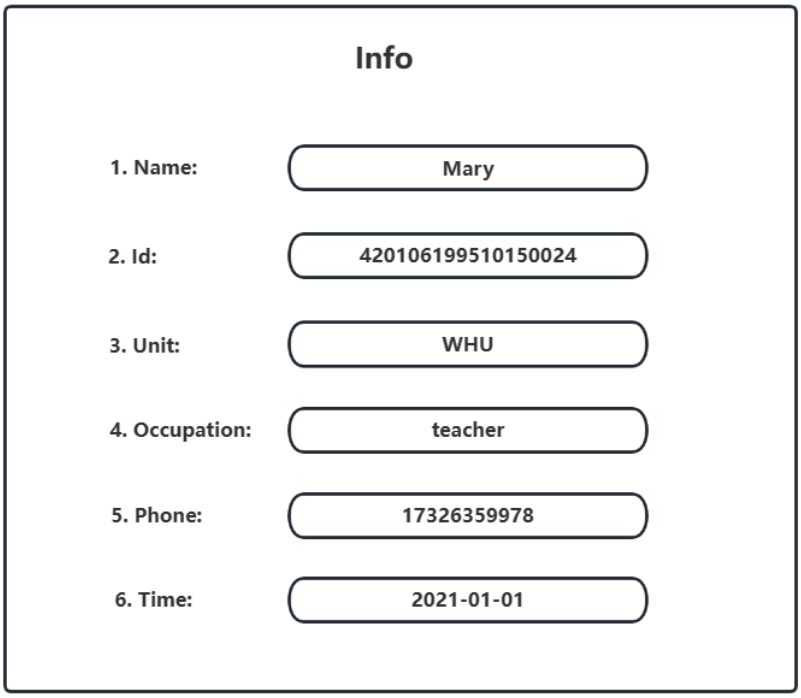


图 1

这里为了简化语法树，我将以上信息简化为：“Info 1. Name：Mary 2. Id：420106199510150024 6. Time：01-01”。我们只选择姓名、身份证号和有效时间来表示个人信息，形成语法树，但依然可以表示出对象的特点。那么可以构建语法树如图 2，其中 L 代表 Letter，N 代表 Number。

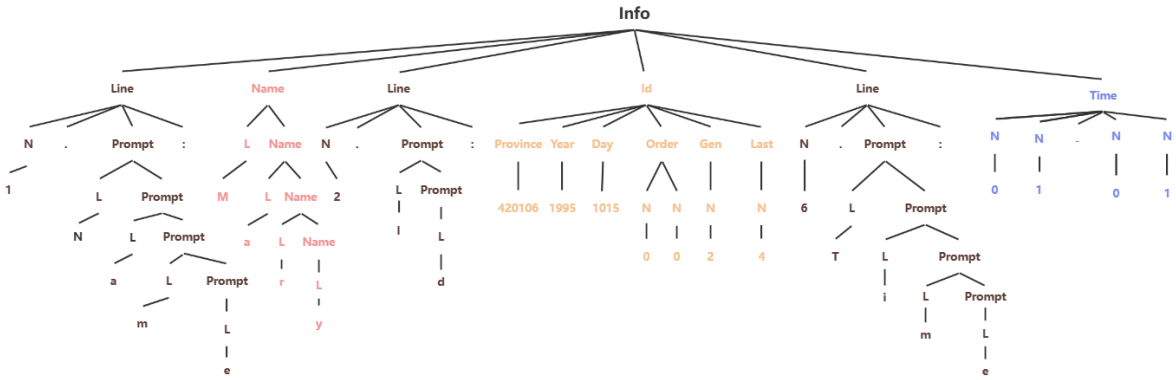


图 2

(5) 明确各部分的任务

①词法分析：识别出待研究符号串中的最基本成分。

根据上述信息收集表，我们个人信息的但此符号分为五大类，分别是提示符、选择符、标识符、常数和连接符。为了便于语义区分，我们又将选择符分为了六小类。我们将根据分类识别出所有单词符号，如表 1。

提示符：

Info/Name/Id/Unit/Occupation/Phone/Time

标识符：

Mike/Mary……（姓名）

Wuhan University/WHU/Harvard/UCL/……（所属单位）

Principle/professor/teacher/student/……（职业）

常数：

1/2/3/4324/420106222/……

连接符：

-/:/.

②语法分析：识别出被处理对象的结构。

如上图 2

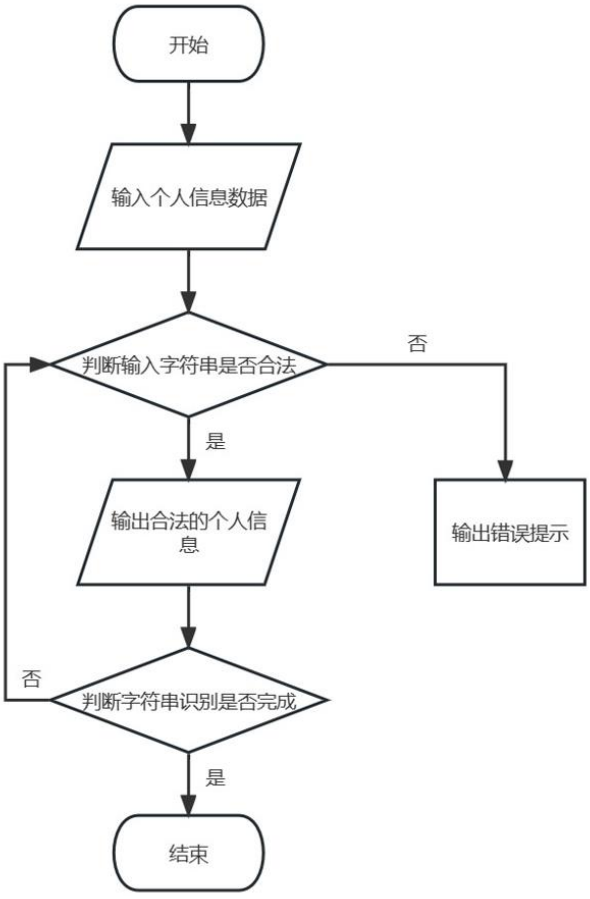
③语义分析：识别出被处理对象后执行的处理/应用目的。

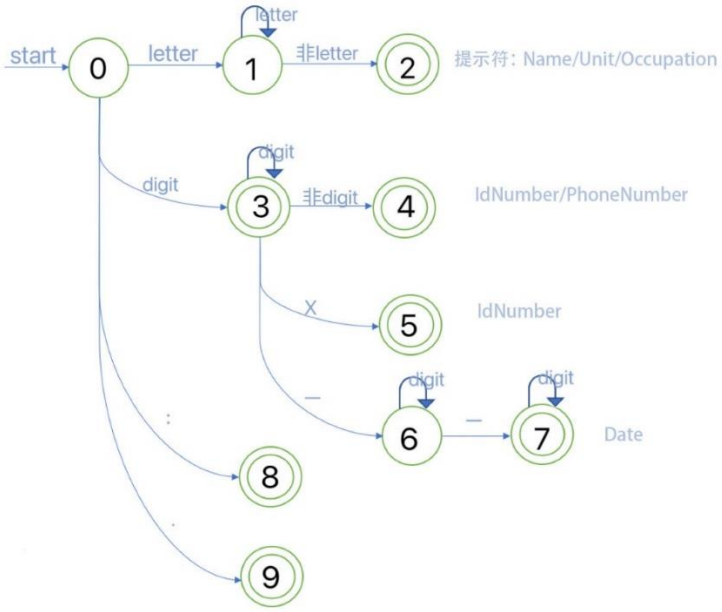
识别出表格中的内容后，可以将参会人员信息转化为二维码。二维码可用于确认参会资格和参会人信息，便于人员管理。

单词符号	类别编码	内码值	单词符号	类别编码	内码值
Info	1	258	Colon(:)	10	267
Name	2	259	Dot(.)	11	268
Id	3	260	IdNumber	12	269
Unit	4	261	Tag	13	270
Occupation	5	262	PhoneNumber	14	271
Phone	6	263	Date	15	272
Time	7	264	IdNumber_ERROR	16	273
No	8	265	PhoneNumber_ERROR	17	274
Connect(-)	9	266			

表 1

二、流程图和 DFA





三、程序源代码

test.l：flex 代码

这部分代码为个人信息编译中的词法分析部分。识别出序号、提示符、特殊符号、标识符这四类单词。此外，还有识别报错的部分。当输入的信息不合法时，输出错误信息提示。例如，身份证位数过多或过少或者包含不恰当的字母。

```
%{
#include <stdio.h>
#include <string.h>
#include "test.tab.h"
void yyerror(char*);
%}
/*记录符号所在行号*/
%option yylineno
/*定义正则表达式*/
letter [A-Za-z]
digit [0-9]
//序号
No [1-9]
//提示符：用于提示用户输入
Info Info
Name Name
Id Id
Unit Unit
Occupation Occupation
Phone Phone
Time Time
//特殊符号
Dot \.
Colon \:
//标识符：用户输入
//姓名、职业等
Tag {letter}+
//身份证号码
IdNumber ({digit}{17})({digit}|[X])
//电话号码
PhoneNumber {digit}{11}
//时间
Date ({digit}{4})("-")({digit}{2})("-")({digit}{2})

/*词法分析出错*/
//身份证号不合法
IdNumber_ERROR [0-9]*[a-zA-Z]+[0-9]*|{digit}{0,16}({digit}|[X])|{digit}{17}({digit}|[X])[0-9]+
//电话号码不合法
PhoneNumber_ERROR [0-9]*[a-zA-Z]+[0-9]*|{digit}{0,10}|{digit}{11}[0-9]+

/*空白符*/
SPACE [ \f\r\t\v]+
/*换行*/
EOL \n
```

```
45.
46.  %%
47.  {SPACE} {}
48.  {EOL} {}
49.
50.  {No} {
51.      fprintf(yyout,"(%d,%s)", No,yytext);//输出token和相应的值
52.      yylval=strdup(yytext); //记录当前的内容
53.      return No;}
54.  {Info} {
55.      fprintf(yyout,"(%d,%s)\n", Info,yytext);
56.      yylval=strdup(yytext);
57.      return Info;}
58.  {Name} {
59.      fprintf(yyout,"(%d,%s)", Name,yytext);
60.      yylval=strdup(yytext);
61.      //printf("Name : %s\n",yytext);
62.      return Name;}
63.  {Id} {
64.      fprintf(yyout,"(%d,%s)", Id,yytext);
65.      yylval=strdup(yytext);
66.      return Id;}
67.  {Unit} {
68.      fprintf(yyout,"(%d,%s)", Unit,yytext);
69.      yylval=strdup(yytext);
70.      return Unit;}
71.  {Occupation} {
72.      fprintf(yyout,"(%d,%s)", Occupation,yytext);
73.      yylval=strdup(yytext);
74.      return Occupation;}
75.  {Phone} {
76.      fprintf(yyout,"(%d,%s)", Phone,yytext);
77.      yylval=strdup(yytext);
78.      return Phone;}
79.  {Time} {
80.      fprintf(yyout,"(%d,%s)", Time,yytext);
81.      yylval=strdup(yytext);
82.      return Time;}
83.
84.  {Dot} {
85.      fprintf(yyout,"(%d,%s)", Dot,yytext);
86.      yylval=strdup(yytext);
87.      return Dot;}
88.  {Colon} {
89.      fprintf(yyout,"(%d,%s)", Colon,yytext);
```

```
90.      yylval=strdup(yytext);
91.      return Colon;}
92.
93.  {Tag} {
94.      fprintf(yyout,"(%d,%s)\n", Tag,yytext);
95.      yylval=strdup(yytext);
96.      return Tag;}
97.  {PhoneNumber} {
98.      fprintf(yyout,"(%d,%s)\n", PhoneNumber,yytext);
99.      yylval=strdup(yytext);
100.     return PhoneNumber;}
101. {Date} {
102.     fprintf(yyout,"(%d,%s)\n", Date,yytext);
103.     yylval=strdup(yytext);
104.     return Date;}
105. {IdNumber} {
106.     fprintf(yyout,"(%d,%s)\n", IdNumber,yytext);
107.     yylval=strdup(yytext);
108.     return IdNumber;}
109.
110. //错误提示
111. {IdNumber_ERROR} {printf("IdNumber_ERROR at line %d: \"%s\"\n",yylineno,yytext);
112. return IdNumber_ERROR;}
113. {PhoneNumber_ERROR} {printf("PhoneNumber_ERROR at line %d: \"%s\"\n",yylineno,yytext);
114. return PhoneNumber_ERROR;}
115.
116.
117. %%
```

test.y：bison 代码

这部分为个人信息编译中的语法分析和中间代码生成部分。如下产生式所示，当一行信息识别完成之后，会生成四元式。以第一行为例，当姓名这一行信息识别完成后，输出（:，Name,Mary,1）。“：”符号表示复制，将 Mary 赋值给 Name；“1”表示信息序号，即第一条信息。

```
%{
#include <stdio.h>
#include <string.h>
int yylex(void);
int infos=0;  //信息的数量
void yyerror(char *);
%}

%debug
%token Info Name Id Unit Occupation Phone Time No Connect Colon Dot
%token IdNumber Tag PhoneNumber Date
%token IdNumber_ERROR PhoneNumber_ERROR
%%

info:
  | Info lines
   {
     //当有6条信息时，说明每一项均填写
    if (infos==6)
    {
        printf("Pass Parser\n");
        printf("There are %d info.",infos);
    }
    //存在项目漏填
    else
    {
        printf("Pass Parser\n");
        printf("You may miss some blanks.");
    }
    }
  ;
lines:
   | line lines
   ;
line:
    No Dot prompt Colon filling
    {
        printf("quaternary:(%s,%s,%s,%s)\n",$4,$3,$5,$1);
        infos=infos+1;
    }
  ;
filling:
  | IdNumber {$$=$1;}
  | Tag {$$=$1;}
  | PhoneNumber {$$=$1;}
  | Date {$$=$1;}
  ;
prompt:
  | Name {$$=$1;}
  | Id  {$$=$1;}
  | Unit  {$$=$1;}
  | Occupation  {$$=$1;}
  | Phone  {$$=$1;}
  | Time  {$$=$1;}
  ;
%%
void yyerror(char *str){
    fprintf(stderr,"error :%s\n",str);
}
int yywrap(void){
    return 1;
}
int main(int argc, char* argv[])
{
    freopen("outSyntax.txt","w",stdout);
    //yydebug=1;
    yyparse();
}
```

四、实验结果

正确输入：

```
Info
1.Name:Mary
2.Id:420106199510150OX
3.Unit:WHU
4.Occupation:teacher
```

```
5.Phone:17326359978
6.Time:2021-01-01
```

正确输出：

```
(258,Info)
(265,1)(268,.)(259,Name)(267,:)(270,Mary)
quaternary:(:,Name,Mary,1)
(265,2)(268,.)(260,Id)(267,:)(269,42010619951015002X)
quaternary:(:,Id,42010619951015002X,2)
(265,3)(268,.)(261,Unit)(267,:)(270,WHU)
quaternary:(:,Unit,WHU,3)
(265,4)(268,.)(262,Occupation)(267,:)(270,teacher)
quaternary:(:,Occupation,teacher,4)
(265,5)(268,.)(263,Phone)(267,:)(271,17326359978)
quaternary:(:,Phone,17326359978,5)
(265,6)(268,.)(264,Time)(267,:)(272,2021-01-01)
quaternary:(:,Time,2021-01-01,6)
Pass Parser
There are 6 info.
```

错误输出-缺少一项输入：

```
(258,Info)
(265,1)(268,.)(259,Name)(267,:)(270,Mary)
quaternary:(:,Name,Mary,1)
(265,2)(268,.)(260,Id)(267,:)(269,42010619951015002X)
quaternary:(:,Id,42010619951015002X,2)
(265,3)(268,.)(261,Unit)(267,:)(270,WHU)
quaternary:(:,Unit,WHU,3)
(265,4)(268,.)(262,Occupation)(267,:)(270,teacher)
quaternary:(:,Occupation,teacher,4)
(265,5)(268,.)(263,Phone)(267,:)(271,17326359978)
quaternary:(:,Phone,17326359978,5)
Pass Parser
You may miss some blanks.
```

错误输出-IdNumber（身份证号码）不合法：

```
(258,Info)
(265,1)(268,.)(259,Name)(267,:)(270,Mary)
quaternary:(:,Name,Mary,1)
(265,2)(268,.)(260,Id)(267,:)IdNumber_ERROR at line 3: "4201061995101500X"
quaternary:(:,Id,(null),2)
Pass Parser
You may miss some blanks.
```

五、参考文献：

[1]编译工程附录：Bison 基础 - 知乎 (zhihu.com)

[2] (47 条消息) 编译原理学习（三）——Flex 实现词法分析器（附 Flex 使用简介）_flex 词法分析_NKU 丨阳的博客-CSDN 博客

[3] (47 条消息) 利用 FLEX & BISON 快速实现简单的 C 语言编译器前端_基于 flex 和 bison 的简单编译器实现_unicxitoiv 的博客-CSDN 博客