编译器设计专题实验
实验报告

实验 4 语法分析

班级：计算机 2102
姓名：李芝塬
学号：2216113163

目录

实验要求

根据输入的文法输出对应的 SLR(1)分析表

题目分析

构造 SLR(1)分析表的步骤：
1.写出拓广文法
2.画出项目集规范族
3.求该非终结符的 FOLLOW 集
4.判断是否是 SLR(1)文法
5.构造 SLR(1)分析表

首先需要从用户输入读取所有的文法，保存在一个数组中。随后在产生式中添加圆点，以求出所有的项目并拓广文法，即加入 S'->S.的项目。下一步是求项目集规范族，需要先求 GO 表，再划分 CLOSURE()闭包。下一步是求非终结符的 FOLLOW 集，因此需要先求 FIRST 表，再求 FOLLOW 表。最后在构造 SLR(1)分析表时，需要结合 GO，action, GOTO, FOLLOW 表，按照规则填写 action 和 GOTO 表。

实验过程

根据已有的 slr1-add.cpp 进行改写。

首先需要从终端中读取用户输入的产生式，并记录在数组 wf 中。随后构造项目集，即在产生式中添加圆点并拓广文法。随后使用 dfs 构造 FIRST 表，再构造 FOLLOW 表。下一步构造 GO 表，最后根据 GO，FOLLOW 的值遍历每个项目，填写对应的表格，即可构造出 SLR 分析表。

在这个程序中，全局变量与数据类型为：

vector<WF> wf	记录所有的文法，每个文法都有一个序号。顺序就是用户输入的顺序
map<string, vector<int>> dic	记录每个左部对应的全部项目的编号。编号就是这个项目在 items 中的下标
map<string, vector<int>> VN_set	key(string)是文法左侧是字符，value 是这个字符对应的原始的文法的编号。一个符号可能有多个文法，所以需要用 vector 记录所有的文法的编号。编号在 wf 作为下标中使用
map<string, bool> vis	在 dfs 中记录是否已经被遍历过
char start	文法的开始符号 S
vector<Closure> collection	记录所有的闭包，每个元素是一个闭包
vector<WF> items	记录所有的项目，即加入了'.'的文法
int go[MAX][MAX]	GO 表

int to[max]	to[i]记录从项目 i-1 到项目 i 的弧
vector<char> V	记录文法中包含的字符的合集，包括终结符和非终结符
bool used[MAX]	在 make_V 中的变量
Content action[MAX][MAX]	action 表
int Goto[MAX][MAX]	GOTO 表
map<string, set<char>> first	FIRST 表
map<string, set<char>> follow	FOLLOW 表
struct Content	记录 action 表的一个单元格，其中 type 是行为，0,1,2 分别代表移进，归约，出错。num 是值，即 sj,rj 中的 j 值。
WF.left	产生式的左边
WF.right	产生式的右边
WF.back	记录当前产生式在所有产生式中的序号，作为在 make_table 的第 2 条中的 j
WF.id	项目集序号
Closure	记录一个闭包

实验中需要完成 make_table()函数。规则如下：

1. if A->α.aβ 属于 Ik 且 GO(Ik,a)=Ij，a 为终结符，则 ACTION[k,a]=sj
2. if A->α.属于 Ik,则对任何终结符 a，a in FOLLOW(A)，则 ACTION[k,a]=rj
3. if S'->S.属于 Ik,则 ACTION[k,#]为 accept
4. if GO(Ik,A)=Ij,A 为非终结符， 则 GOTO[k,A]=j;
5. 其余格子为错误

对于任何项目，圆点要么在最末尾，要么不在末尾。如果在末尾，则可能符合第 2,3 条，否则可能符合第 1,4 条。表格的纵坐标是闭包 Ik，横坐标是在产生式中出现过的所有字符，而每个闭包中又有多个项目，因此需要 3 层 for 循环，最外层是遍历每个闭包，第 2 层是遍历这个闭包中的每个项目，第 3 层是遍历字符集 V 中的每个字符。

每次遍历时，首先需要寻找圆点'.'的位置，如果在最后，则使用规则 2,3 处理，否则使用规则 1,4 处理。对于规则 2,3，需要首先检查是否是接受状态，否则检查 FOLLOW 表，如果当前字符在 FOLLOW 表中，则 action 设为 rj。 对于其他规则，则先找到圆点的位置，对 A->α.aβ，检查 a 是否与当前遍历到的字符一致，一致则检查 GO 表，设定 action 和 GOTO 为 sj。

所以 make_table 程序如下：

```
void make_table()
{
    memset(Goto, -1, sizeof(Goto));
    // sj 对应条目1和4
    //     | -- action -------| -----GOTO---- |
    //     | i   | +    |*    | E   |T   |F   |
    //  I0 |
    //  I1 |
    // ...
```

```
    for (int k = 0; k < collection.size(); k++)
    {
        for (int i = 0; i < collection[k].element.size(); i++)
        {
            WF &t = collection[k].element[i];        // 闭包 row 的项目 col
            if (t.right[t.right.length() - 1] == CH) // 项目最右侧是'.',
即匹配选项 2 和 3
            {
                // 此时 t 是一个满足 2 或 3 的项目
                if (t.left[0] == start) // 选项 3，即匹配成功
                    action[k]['#'] = Content(2, -1);
                else // 满足选项 2，需要按照产生式 j 进行归约
                    for (auto col = V.begin(); col != V.end(); col++)
                    {
                        if (0 == follow[t.left].count(*col))
                            continue;
                        else
                        {
                            int j = t.back;
                            action[k][*col] = Content(1, j);
                        }
                    }
            }
            else // 圆点不在最右，即可能匹配选项 1 和 4
            {
                int index; // 圆点的位置
                for (index = 0; index < t.right.length(); index++)
                {
                    if (t.right[index] == CH)
                    {
                        break;
                    }
                }
                if (index >= t.right.length() - 1)
                {
                    exit(EXIT_FAILURE);
                }
                else
                {
                    for (auto col = V.begin(); col != V.end(); col++)
                    {
                        char a = *col;
                        if (t.right[index + 1] == a)
                        {
```

```
                            int j = go[k][a];
                            if (j != -1)
                            {
                                if (!isupper(a))
                                    action[k][a] = Content(0, j); //
action[Ik][a]=sj
                                else
                                    Goto[k][a] = j;
                            }
                            else
                            {
                                continue;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

实验结果

运行 run.sh，结果如下：

```
现在的日期:
Sat Jun  1 10:52:51 AM CST 2024
开始编译
开始运行
输入的内容:
S
7
S->E
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
i*i+i输出的内容:
输入开始符号:
输入文法个数和文法: ------------------------项目表------------------------
S->.E back:0 id:0
S->E. back:0 id:1
E->.E+T back:1 id:2
E->E.+T back:1 id:3
E->E+.T back:1 id:4
E->E+T. back:1 id:5
E->.T back:2 id:6
E->T. back:2 id:7
T->.T*F back:3 id:8
T->T.*F back:3 id:9
T->T*.F back:3 id:10
T->T*F. back:3 id:11
T->.F back:4 id:12
T->F. back:4 id:13
F->.(E) back:5 id:14
F->(.E) back:5 id:15
F->(E.) back:5 id:16
F->(E). back:5 id:17
F->.i back:6 id:18
F->i. back:6 id:19
```

```
------------------------------------------------------
***************FIRST集**************************
FIRST(E)={(,i}
FIRST(F)={(,i}
FIRST(S)={(,i}
FIRST(T)={(,i}
-------------CLOSURE---------------------
              closure-I0
E->.E+T
E->.T
F->.(E)
F->.i
S->.E
T->.F
T->.T*F
              closure-I1
E->.E+T
E->.T
F->(.E)
F->.(E)
F->.i
T->.F
T->.T*F
              closure-I2
E->E.+T
S->E.
              closure-I3
T->F.
              closure-I4
E->T.
T->T.*F
              closure-I5
F->i.
              closure-I6
E->E.+T
F->(E.)
```

```
              closure-I7
E->E+.T
F->.(E)
F->.i
T->.F
T->.T*F
              closure-I8
F->.(E)
F->.i
T->T*.F
              closure-I9
F->(E).
              closure-I10
E->E+T.
T->T.*F
              closure-I11
T->T*F.

---------------EDGE----------------------
I0--(--I1
I0--E--I2
I0--F--I3
I0--T--I4
I0--i--I5
I1--(--I1
I1--F--I3
I1--T--I4
I1--i--I5
I1--E--I6
I2--+--I7
I4--*--I8
I6--+--I7
I6--)--I9
I7--(--I1
I7--F--I3
I7--i--I5
I7--T--I10
I8--(--I1
I8--i--I5
I8--F--I11
I10--*--I8
```

LR(0)分析表

	(	)	*	+	E	F	S	T	i	#
0	S1				2	3		4	S5	
1	S1				6	3		4	S5	
2				S7						acc
3		R4	R4	R4						R4
4		R2	S8	R2						R2
5		R6	R6	R6						R6
6		S9		S7						
7	S1					3		10	S5	
8	S1					11			S5	
9		R5	R5	R5						R5
10		R1	S8	R1						R1
11		R3	R3	R3						R3

输入待分析的字符串:

steps	op-stack	input	operation	state-stack	ACTION	GOTO
1	#	i*i+i#	shift	0	S5	
2	#i	*i+i#	reduce(F->i)	05	R6	3
3	#F	*i+i#	reduce(T->F)	03	R4	4
4	#T	*i+i#	shift	04	S8	
5	#T*	i+i#	shift	048	S5	
6	#T*i	+i#	reduce(F->i)	0485	R6	11
7	#T*F	+i#	reduce(T->T*F)	04811	R3	4
8	#T	+i#	reduce(E->T)	04	R2	2
9	#E	+i#	shift	02	S7	
10	#E+	i#	shift	027	S5	
11	#E+i	#	reduce(F->i)	0275	R6	3
12	#E+F	#	reduce(T->F)	0273	R4	10
13	#E+T	#	reduce(E->E+T)	02710	R1	2
14	#E	#	accept	02	acc	

如图所示，输入的产生式为：

S->E

E->E+T

E->T

T->T*F

T->F

F->(E)

F->i

要判断的表达式为：

i*i+i

即可生成 SLR(1)表并进行自下而上的语法分析。

实验总结

本次实验中所给的代码已经完成了大部分的工作，只需要自己完成 make_table()函数即可。完成本次实验需要先看懂示例代码，而读懂示例代码需要熟悉相关的理论知识，才能明白函数与变量名的具体含义。因此，本次实验看似需要写的代码不多，但是要求熟悉相关理论，还是具有一定的难度。

实验的过程有效地增强了我对 SLR 中各种函数与表之间关系的理解与记忆，帮助我更好地理解课本上的知识。

附录

源代码：https://github.com/KnownCircle/compileExperiment/tree/main/4

ans.cpp：

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <cctype>
#include <vector>
#include <string>
#include <queue>
#include <map>
#include <set>
#include <sstream>
#include <fstream>
#define MAX 507
#define DEBUG

/*计算机2102 李芝源 2216113163 zhiyuanli0122@outlook.com
 * 参考的教材：《编译原理 第三版》陈火旺 9787118022070
 * 使用的命名结合了slr1-add.cpp与教材
 * 参考链接：https://blog.csdn.net/GJ_007/article/details/79587693
*/

using namespace std;

// 文法类 每个WF就是一个产生式
class WF
{
public:
    string left, right; // 产生式左边和右边
    int back;           // 记录当前产生式在所有产生式中的序号，作为在make_table的第2条中的j
    int id;             // 项目集序号
    WF(char s1[], char s2[], int x, int y)
    {
        left = s1;
        right = s2;
        back = x;
        id = y;
    }
    WF(const string &s1, const string &s2, int x, int y)
    {
```

```
        left = s1;
        right = s2;
        back = x;
        id = y;
    }
    // 重载<运算符
    bool operator<(const WF &a) const
    {
        if (left == a.left)
            return right < a.right;
        return left < a.left;
    }
    // 重载==
    bool operator==(const WF &a) const
    {
        return (left == a.left) && (right == a.right);
    }
    void print()
    {
        printf("%s->%s\n", left.c_str(), right.c_str());
    }
};
// 闭包类
class Closure
{
public:
    vector<WF> element;
    void print(string str)
    {
        printf("%-15s%-15s\n", "", str.c_str());
        for (int i = 0; i < element.size(); i++)
            element[i].print();
    }
    // 重载==
    bool operator==(const Closure &a) const
    {
        if (a.element.size() != element.size())
            return false;
        for (int i = 0; i < a.element.size(); i++)
            if (element[i] == a.element[i])
                continue;
            else
                return false;
        return true;
```

```
    }
};

/*每个content就是一个GOTO表格的结构，代表了操作与值
type 0,1,2代表shift, reduce, accept
num是具体的值
*/
struct Content
{
    int type;
    int num;
    string out;
    Content() { type = -1; }
    Content(int a, int b)
        : type(a), num(b) {}
};

vector<WF> wf;                  // 记录所有的文法，每个文法都有一个序号。顺序就是用户输入的顺序
map<string, vector<int>> dic;    // 记录每个左部对应的全部项目的编号。编号就是这个项目在items中的下标
map<string, vector<int>> VN_set; // key(string)是文法左侧是字符，value是这个字符对应的原始的文法的编号。一个符号可能有多个文法，所以需要用vector记录所有的文法的编号。编号在wf作为下标中使用
map<string, bool> vis;           // 在dfs中记录是否已经被遍历过
char start;                      // 文法的开始符号S
vector<Closure> collection;      // 记录所有的闭包，每个元素是一个闭包
vector<WF> items;                // 记录所有的项目，即加入了'.'的文法。
char CH = '.';
int go[MAX][MAX];
int to[MAX];    // to[i]记录从项目i-1到项目i的弧
vector<char> V; // 记录文法中包含的字符的合集，包括终结符和非终结符
bool used[MAX];
Content action[MAX][MAX];
int Goto[MAX][MAX];
map<string, set<char>> first; // first[X] 就是 FIRST(X)的值
map<string, set<char>> follow;

// 在读取所有的文法之后统一生成项目（即圆点在不同位置的文法）
void make_item()
{
    // ??? WTF 这里原作者写错了
    // memset(to, -1, sizeof(-1));
    memset(to, -1, sizeof(to));
```

```
    for (int i = 0; i < wf.size(); i++)
        VN_set[wf[i].left].push_back(i);
    for (int i = 0; i < wf.size(); i++)
    {
        for (int j = 0; j <= wf[i].right.length(); j++)
        {
            string temp = wf[i].right; // 记录加入'.'后的右部
            temp.insert(temp.begin() + j, CH);
            dic[wf[i].left].push_back(items.size()); // item.size 即为当
前项目的编号
            if (j > 0)
                to[items.size() - 1] = items.size();
            items.push_back(WF(wf[i].left, temp, i, items.size()));
        }
    }
#ifdef DEBUG
    puts("-------------------------项目表-------------------------");
    for (int i = 0; i < items.size(); i++)
        printf("%s->%s back:%d id:%d\n", items[i].left.c_str(),
items[i].right.c_str(), items[i].back, items[i].id);
    puts("--------------------------------------------------------");
#endif
}

// 遍历求符号X的FIRST，结果存储在map first中
void dfs(const string &x)
{
    if (vis[x])
        return;
    vis[x] = 1;
    vector<int> &id = VN_set[x];        // 获取符号X的所有文法的编号
    for (int i = 0; i < id.size(); i++) // 每次循环都只分析一条文法
    {
        string &left = wf[id[i]].left;
        string &right = wf[id[i]].right;
        for (int j = 0; j < right.length(); j++)
            if (isupper(right[j])) // 大写字母是非终结项， 小写字母与符号是
终结项
            {
                dfs(right.substr(j, 1));
                set<char> &temp = first[right.substr(j, 1)];
                set<char>::iterator it = temp.begin();
                bool flag = true;
                for (; it != temp.end(); it++)
```

```
                {
                    if (*it == '~')
                        flag = false;
                    first[left].insert(*it);
                }
                if (flag)
                    break;
            }
            else
            {
                // 符合规则1
                first[left].insert(right[j]);
                break;
            }
    }
}

/*求FIRST()函数
原理：教材P78
对于FIRST(X):
1. X in VT: FIRST(X)={X}
2. x in VN, and X->a... : 把a加入FIRST(X)中。若存在X->e，则加入e
3. X->Y and Y in VN: 加入FIRST(Y)到FIRST(X)中。 对X->Y1 Y2 ... Yi ...
Yk，检查e
*/
void make_first()
{
    vis.clear();
    map<string, vector<int>>::iterator it2 = dic.begin();
    for (; it2 != dic.end(); it2++)
    {
        if (vis[it2->first]) // it2->first 是项目的左部。 vis[]表示这个项目
是否被检查过
            continue;
        else
            dfs(it2->first);
    }
#ifdef DEBUG
    puts("****************FIRST集**************************");
    map<string, set<char>>::iterator it = first.begin();
    for (; it != first.end(); it++)
    {
        printf("FIRST(%s)={", it->first.c_str());
        set<char> &temp = it->second;
```

```
        set<char>::iterator it1 = temp.begin();
        bool flag = false;
        for (; it1 != temp.end(); it1++)
        {
            if (flag)
                printf(",");
            printf("%c", *it1);
            flag = true;
        }
        puts("}");
    }
#endif
}

void append(const string &str1, const string &str2)
{
    set<char> &from = follow[str1];
    set<char> &to = follow[str2];
    set<char>::iterator it = from.begin();
    for (; it != from.end(); it++)
        to.insert(*it);
}

bool _check(const vector<int> &id, const string str)
{
    for (int i = 0; i < id.size(); i++)
    {
        int x = id[i];
        if (wf[x].right == str)
            return true;
    }
    return false;
}

/*求FOLLOW()函数
原理：书P79
1. 对于开始符号，加入#
2. 若A-> αBβ 是一个产生式，则FOLLOW(B)中加入FIRST(β)\{ε}
3. 若A->αB 是产生式，或A->αBβ 是产生式而β-->ε，则把FOLLOW(A)加入FOLLOW(B)
中
*/
void make_follow()
{
    while (true)
```

```
    {
        bool goon = false;
        map<string, vector<int>>::iterator it2 = VN_set.begin();
        for (; it2 != VN_set.end(); it2++)
        {
            vector<int> &id = it2->second;
            for (int i = 0; i < id.size(); i++)
            {
                bool flag = true;
                WF &tt = wf[id[i]];
                string &left = tt.left;
                const string &right = tt.right;
                for (int j = right.length() - 1; j >= 0; j--)
                    if (isupper(right[j]))
                    {
                        if (flag)
                        {
                            int tx = follow[right.substr(j, 1)].size();
                            append(left, right.substr(j, 1));
                            int tx1 = follow[right.substr(j, 1)].size();
                            if (tx1 > tx)
                                goon = true;
                            if (_check(id, "~"))
                                flag = false;
                        }
                        for (int k = j + 1; k < right.length(); k++)
                            if (isupper(right[k]))
                            {
                                string idd = right.substr(k, 1);
                                set<char> &from = first[idd];
                                set<char> &to = follow[right.substr(j,
1)];
                                set<char>::iterator it1 = from.begin();
                                int tx = follow[right.substr(j,
1)].size();
                                for (; it1 != from.end(); it1++)
                                    if (*it1 != '~')
                                        to.insert(*it1);
                                int tx1 = follow[right.substr(j,
1)].size();
                                if (tx1 > tx)
                                    goon = true;
                                if (_check(id, "~"))
                                    break;
```

```
                            }
                            else
                            {
                                int tx = follow[right.substr(j,
1)].size();
                                follow[right.substr(j,
1)].insert(right[k]);
                                int tx1 = follow[right.substr(j,
1)].size();
                                if (tx1 > tx)
                                    goon = true;
                                break;
                            }
                        }
                        else
                            flag = false;
                    }
                }
                if (!goon)
                    break;
            }
#ifdef DEBUG
            // puts ("***************FOLLOW 集*******************");
            map<string, set<char>>::iterator it = follow.begin();
            for (; it != follow.end(); it++)
            {
                //  printf ( "FOLLOW(%s)={" , it->first.c_str() );不是我写的
                set<char> &temp = it->second;
                // if ( it->first[0] == 'S' )不是我写的
                temp.insert('#');
                set<char>::iterator it1 = temp.begin();
                bool flag = false;
                for (; it1 != temp.end(); it1++)
                {
                    /*  if ( flag ) printf ( "," );
                      printf ( "%c" , *it1 );不是我写的*/
                    flag = true;
                }
                // puts ("}");
            }
#endif
}

// 划分闭包 CLOSURE
```

```
void make_set()
{
    bool has[MAX];
    for (int i = 0; i < items.size(); i++)
        if (items[i].left[0] == start && items[i].right[0] == CH)
        {
            Closure temp;
            string &str = items[i].right;
            vector<WF> &element = temp.element;
            element.push_back(items[i]);
            int x = 0;
            for (x = 0; x < str.length(); x++)
                if (str[x] == CH)
                    break;

            memset(has, 0, sizeof(has));
            has[i] = 1;
            if (x != str.length() - 1)
            {
                queue<string> q;
                q.push(str.substr(x + 1, 1));
                while (!q.empty())
                {
                    string u = q.front();
                    q.pop();
                    vector<int> &id = dic[u];
                    for (int j = 0; j < id.size(); j++)
                    {
                        int tx = id[j];
                        if (items[tx].right[0] == CH)
                        {
                            if (has[tx])
                                continue;
                            has[tx] = 1;
                            if (isupper(items[tx].right[1]))
                                q.push(items[tx].right.substr(1, 1));
                            element.push_back(items[tx]);
                        }
                    }
                }
            }
            collection.push_back(temp);
        }
    for (int i = 0; i < collection.size(); i++)
```

```
{
    map<int, Closure> temp;
    for (int j = 0; j < collection[i].element.size(); j++)
    {
        string str = collection[i].element[j].right;
        int x = 0;
        for (; x < str.length(); x++)
            if (str[x] == CH)
                break;
        if (x == str.length() - 1)
            continue;
        int y = str[x + 1];
        int ii;
        // cout << i << "previous: " << str << endl;
        str.erase(str.begin() + x);
        str.insert(str.begin() + x + 1, CH);
        // cout << i <<"after: " << str << endl;
        WF cmp = WF(collection[i].element[j].left, str, -1, -1);
        for (int k = 0; k < items.size(); k++)
            if (items[k] == cmp)
            {
                ii = k;
                break;
            }
        // string& str1 = items[ii].right;
        memset(has, 0, sizeof(has));
        vector<WF> &element = temp[y].element;
        element.push_back(items[ii]);
        has[ii] = 1;
        x++;

        if (x != str.length() - 1)
        {
            queue<string> q;
            q.push(str.substr(x + 1, 1));
            while (!q.empty())
            {
                string u = q.front();
                q.pop();
                vector<int> &id = dic[u];
                for (int j = 0; j < id.size(); j++)
                {
                    int tx = id[j];
                    if (items[tx].right[0] == CH)
```

```
                        {
                            if (has[tx])
                                continue;
                            has[tx] = 1;
                            if (isupper(items[tx].right[1]))
                                q.push(items[tx].right.substr(1, 1));
                            element.push_back(items[tx]);
                        }
                    }
                }
            }
        }
        map<int, Closure>::iterator it = temp.begin();
        for (; it != temp.end(); it++)
            collection.push_back(it->second);
        for (int i = 0; i < collection.size(); i++)
            sort(collection[i].element.begin(),
collection[i].element.end());
        for (int i = 0; i < collection.size(); i++)
            for (int j = i + 1; j < collection.size(); j++)
                if (collection[i] == collection[j])
                    collection.erase(collection.begin() + j);
    }
#ifdef DEBUG
    puts("-------------CLOSURE---------------------");
    stringstream sin;
    for (int i = 0; i < collection.size(); i++)
    {
        sin.clear();
        string out;
        sin << "closure-I" << i;
        sin >> out;
        collection[i].print(out);
    }
    puts("");
#endif
}

// 记录所有产生式中的所有符号，保存在V中
void make_V()
{
    memset(used, 0, sizeof(used));
    for (int i = 0; i < wf.size(); i++)
    {
```

```
        string &str = wf[i].left;
        for (int j = 0; j < str.length(); j++)
        {
            if (used[str[j]])
                continue;
            used[str[j]] = 1;
            V.push_back(str[j]);
        }
        string &str1 = wf[i].right;
        for (int j = 0; j < str1.length(); j++)
        {
            if (used[str1[j]])
                continue;
            used[str1[j]] = 1;
            V.push_back(str1[j]);
        }
    }
    sort(V.begin(), V.end());
    V.push_back('#');
}

void make_cmp(vector<WF> &cmp1, int i, char ch)
{
    for (int j = 0; j < collection[i].element.size(); j++) //
collection[i].element 就是这个闭包中的所有项目构成的 vector
    {
        string str = collection[i].element[j].right;
        int k;
        for (k = 0; k < str.length(); k++)
            if (str[k] == CH)
                break;
        if (k != str.length() - 1 && str[k + 1] == ch)
        {
            str.erase(str.begin() + k);
            str.insert(str.begin() + k + 1, CH);
            cmp1.push_back(WF(collection[i].element[j].left, str, -1, -
1));
        }
    }
    sort(cmp1.begin(), cmp1.end());
}

void make_go()
{
```

```
    memset(go, -1, sizeof(go));
    int m = collection.size();

    for (int t = 0; t < V.size(); t++)
    {
        char ch = V[t];
        for (int i = 0; i < m; i++)
        {
            vector<WF> cmp1;
            make_cmp(cmp1, i, ch);
            if (cmp1.size() == 0)
                continue;
            for (int j = 0; j < m; j++)
            {
                vector<WF> cmp2;
                for (int k = 0; k < collection[j].element.size(); k++)
                {
                    string &str = collection[j].element[k].right;
                    int x;
                    for (x = 0; x < str.length(); x++)
                        if (str[x] == CH)
                            break;
                    if (x && str[x - 1] == ch)
                        cmp2.push_back(WF(collection[j].element[k].left,
str, -1, -1));
                }
                sort(cmp2.begin(), cmp2.end());
                bool flag = true;
                if (cmp2.size() != cmp1.size())
                    continue;
                for (int k = 0; k < cmp1.size(); k++)
                    if (cmp1[k] == cmp2[k])
                        continue;
                    else
                        flag = false;
                if (flag)
                    go[i][ch] = j;
            }
        }
    }
#ifdef DEBUG
    puts("---------------EDGE----------------------");
    stringstream sin;
    string out;
```

```
    for (int i = 0; i < m; i++)
        for (int j = 0; j < m; j++)
            for (int k = 0; k < MAX; k++)
                if (go[i][k] == j)
                {
                    sin.clear();
                    sin << "I" << i << "--" << (char)(k) << "--I" << j;
                    sin >> out;
                    printf("%s\n", out.c_str());
                }
#endif
}

/* 完成action 表和GOTO 表
原理：书P112
1. if A->α.aβ 属于Ik 且GO(Ik,a)=Ij，a 为终结符，则ACTION[k,a]=sj
2. if A->α.属于Ik,则对任何终结符a，a in FOLLOW(A)，则ACTION[k,a]=rj
3. if S'->S.属于Ik,则ACTION[k,#]为accept
4. if GO(Ik,A)=Ij,A 为非终结符，  则GOTO[k,A]=j;
5. 其余格子为错误

实现方式:
任一项目要么属于2，要么属于1（即圆点要么在最后，要么不在最后）
*/
void make_table()
{
    memset(Goto, -1, sizeof(Goto));
    // sj 对应条目1 和4
    //     | -- action -------| -----GOTO---- |
    //     | i   | +    |*    | E   |T   |F   |
    //  I0 |
    //  I1 |
    // ...
    for (int k = 0; k < collection.size(); k++)
    {
        for (int i = 0; i < collection[k].element.size(); i++)
        {
            WF &t = collection[k].element[i];        // 闭包row 的项目col
            if (t.right[t.right.length() - 1] == CH) // 项目最右侧是'.',
即匹配选项2 和3
            {
                // 此时t 是一个满足2 或3 的项目
                if (t.left[0] == start) // 选项3，即匹配成功
                    action[k]['#'] = Content(2, -1);
```

```
                else // 满足选项2，需要按照产生式j进行归约
                    for (auto col = V.begin(); col != V.end(); col++)
                    {
                        if (0 == follow[t.left].count(*col))
                            continue;
                        else
                        {
                            int j = t.back;
                            action[k][*col] = Content(1, j);
                        }
                    }
            }
            else // 圆点不在最右，即可能匹配选项1和4
            {
                int index; // 圆点的位置
                for (index = 0; index < t.right.length(); index++)
                {
                    if (t.right[index] == CH)
                    {
                        break;
                    }
                }
                if (index >= t.right.length() - 1)
                {
                    exit(EXIT_FAILURE);
                }
                else
                {
                    for (auto col = V.begin(); col != V.end(); col++)
                    {
                        char a = *col;
                        if (t.right[index + 1] == a)
                        {
                            int j = go[k][a];
                            if (j != -1)
                            {
                                if (!isupper(a))
                                    action[k][a] = Content(0, j); //
action[Ik][a]=sj
                                else
                                    Goto[k][a] = j;
                            }
                            else
                            {
```

```
                                continue;
                            }
                        }
                    }
                }
            }
        }
    }
#ifdef DEBUG
    cout << "_____________________________LR(0)分析表
_____________________________" << endl;
    printf("%8s%5c%5s", "|", V[0], "|");
    for (int i = 1; i < V.size(); i++)
        printf("%5c%5s", V[i], "|");
    puts("");
    for (int i = 0; i < (V.size() + 1) * 10; i++)
        printf("_");
    puts("");
    stringstream sin;
    for (int i = 0; i < collection.size(); i++)
    {
        printf("%5d%5s", i, "|");
        for (int col = 0; col < V.size(); col++)
        {
            char ch = V[col];
            if (isupper(ch))
            {
                if (Goto[i][ch] == -1)
                    printf("%8s", "|");
                else
                    printf("%5d%3s", Goto[i][ch], "|");
            }
            else
            {
                sin.clear();
                if (action[i][ch].type == -1)
                    printf("%8s", "|");
                else
                {
                    Content &temp = action[i][ch];
                    if (temp.type == 0)
                        sin << "S";
                    if (temp.type == 1)
                        sin << "R";
```

```
                    if (temp.type == 2)
                        sin << "acc";
                    if (temp.num != -1)
                        sin << temp.num;
                    sin >> temp.out;
                    printf("%6s%1s", temp.out.c_str(), "|");
                }
            }
        }
        puts("");
    }
    for (int i = 0; i < (V.size() + 1) * 10; i++)
        printf("_");
    puts("");
#endif
}

void print(string s1, string s2, string s3, string s4, string s5,
string s6, string s7)
{
    printf("%-15s%-15s%-15s%-20s%-15s%-15s%-15s\n", s1.c_str(),
s2.c_str(), s3.c_str(), s4.c_str(), s5.c_str(),
           s6.c_str(), s7.c_str());
}

string get_steps(int x)
{
    stringstream sin;
    sin << x;
    string ret;
    sin >> ret;
    return ret;
}

template <class T>
string get_stk(vector<T> stk)
{
    stringstream sin;
    for (int i = 0; i < stk.size(); i++)
        sin << stk[i];
    string ret;
    sin >> ret;
    return ret;
}
```

```
string get_shift(WF &temp)
{
    stringstream sin;
    sin << "reduce(" << temp.left << "->" << temp.right << ")";
    string out;
    sin >> out;
    return out;
}

void analyse(string src)
{
    print("steps", "op-stack", "input", "operation", "state-stack",
"ACTION", "GOTO");
    vector<char> op_stack;
    vector<int> st_stack;
    src += "#";
    op_stack.push_back('#');
    st_stack.push_back(0);
    int steps = 1;
    for (int i = 0; i < src.length(); i++)
    {
        char u = src[i];
        int top = st_stack[st_stack.size() - 1];
        Content &act = action[top][u];
        if (act.type == 0)
        {
            print(get_steps(steps++), get_stk(op_stack), src.substr(i),
"shift", get_stk(st_stack), act.out, "");
            op_stack.push_back(u);
            st_stack.push_back(act.num);
        }
        else if (act.type == 1)
        {
            WF &tt = wf[act.num];
            int y = st_stack[st_stack.size() - tt.right.length() - 1];
            int x = Goto[y][tt.left[0]];
            print(get_steps(steps++), get_stk(op_stack), src.substr(i),
get_shift(tt), get_stk(st_stack), act.out, get_steps(x));
            for (int j = 0; j < tt.right.length(); j++)
            {
                st_stack.pop_back();
                op_stack.pop_back();
            }
```

```
            op_stack.push_back(tt.left[0]);
            st_stack.push_back(x);
            i--;
        }
        else if (act.type == 2)
        {
            print(get_steps(steps++), get_stk(op_stack), src.substr(i),
"accept", get_stk(st_stack), act.out, "");
        }
        else
            continue;
    }
}

int main()
{
    int n;
    char s[MAX];
    cout << "输入开始符号: \n";
    cin >> start;
    cout << "输入文法个数和文法: ";

    if (scanf("%d", &n) > 0 && n > 0)
    {
        for (int i = 0; i < n; i++)
        {
            cin >> s;
            int len = strlen(s);
            int j; // 记录文法中箭头的位置
            for (j = 0; j < len; j++)
                if (s[j] == '-')
                    break;
            s[j] = 0;
            wf.push_back(WF(s, s + j + 2, -1, -1));
        }
        make_item();
        make_first();
        make_follow();
        make_set();
        make_V();
        make_go();
        make_table();
        string s1;
        cout << "输入待分析的字符串: \n";
```

```
        cin >> s1;
        analyse(s1);
    }
    else
    {
        cout << "请输入合法的正整数!\n";
    }
}
```

input.txt:

```
S
7
S->E
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
i*i+i
```

run.sh:

```
clear
echo 现在的日期:
date
echo 开始编译
g++ ans.cpp -o b
echo 开始运行
echo 输入的内容:
cat input.txt
echo 输出的内容:
./b < input.txt > output.txt
cat output.txt
```