

**杭州电子科技大学**

**《编译原理课程实践》**

**实验报告**

题 目：编译理论知识验证

学 院：计算机

专 业：计算机科学与技术

班 级：21052314

学 号：21051408

姓 名：蔡龙祥

完成日期：2023-11-7

1. **实验目的**

学习如何消除文法的左递归

1. **实验内容与实验要求**

文法左递归消去和左公共因子提取方法

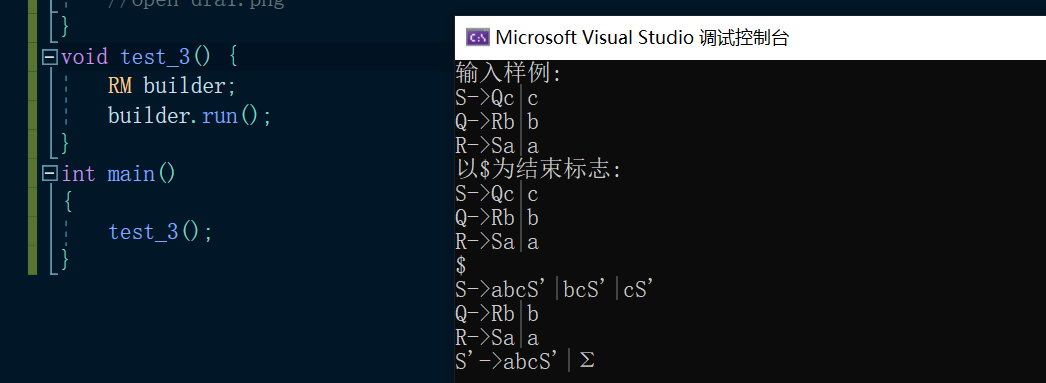
1. **设计方案与算法描述**

消除左递归是为了构造不带回溯的自上而下的分析算法

首先考虑直接消除产生式中的左递归,这是比较容易的。假定关于非终结符P的规则为  
P→Pα | β  
其中，β不以P开头。那么，我们可以把P的规则改写为如下的非直接左递归形式：  
P→βP′  
P′→αP′ | ε （ε为空字）

而对于文法的非终结符有这样的要求,希望对于每一个非终结符,当有新的输入到达时,它仅可以转换到一个确定的状态,这需要提取公共左因子,经过反复提取左因子，就能够把每个非终结符的所有候选集互不重叠。

1. **测试结果**



1. **源代码**

rm\_recurse.h

#pragma once

#include<bits/stdc++.h>

using namespace std;

struct kvs{

string key;

set<string>projects;

};

class RM {

public:

RM() = default;

public:

vector<kvs>kvs\_;

vector<string>not\_end\_keys\_;

void get\_kvs();

void run();

void clean();

kvs split(string str);

void get\_not\_end();

bool check\_all\_end(string str);

void rm\_direct\_recurse(int pos);

string replace(string str, string ori, string another) {

int pos = -1;

while (pos=str.find(ori) != -1) {

str.replace(pos, ori.size(), another);

}

return str;

}

};

#include"rm\_recurse.h"

void RM::clean() {

}

void RM::run() {

get\_kvs();

//for (int i = 0; i < kvs\_.size(); i++) {

// cout << "lval:" << kvs\_[i].key<<" ";

// cout << "projects:";

// for (auto project : kvs\_[i].projects) {

// cout << " " << project ;

// }

// cout<< "\n";

//}

get\_not\_end();

int n = not\_end\_keys\_.size();

cout << "n:" << n << endl;

for (int i = 0; i < n; i++) {

for (int j = 0; j < i; j++) {

int pi = -1, pj = -1;

for (int k = 0; k < n; k++) {

if (not\_end\_keys\_[k] == kvs\_[i].key) {

pi = k;

break;

}

}

for (int k = 0; k < n; k++) {

if (not\_end\_keys\_[k] == kvs\_[j].key) {

pj = k;

break;

}

}

if (pi == -1 || pj == -1)continue;

auto it = kvs\_[pi].projects.begin();

set<string>::iterator it2;

set<string>tempset;

for (it; it != kvs\_[pi].projects.end(); it++)

{

string tempstr = \*it;

string ss;

if (tempstr.find(not\_end\_keys\_[j]) != -1)

{

for (it2 = kvs\_[pj].projects.begin(); it2 != kvs\_[pj].projects.end(); it2++)

{

ss = tempstr;

tempset.insert(replace(ss, not\_end\_keys\_[j], \*it2));

}

}

}

vector<string>temv;

for (it = kvs\_[pi].projects.begin(); it != kvs\_[pi].projects.end(); it++)

{

string str = \*it;

if (str.find(not\_end\_keys\_[j]) != -1)

temv.push\_back(str);

}

for (int x = 0; x < temv.size(); x++)

{

it = kvs\_[pi].projects.find(temv[x]);

if (it != kvs\_[pi].projects.end())

kvs\_[pi].projects.erase(it);

}

for (it = tempset.begin(); it != tempset.end(); it++)

kvs\_[pi].projects.insert(\*it);

rm\_direct\_recurse(pi);

}

}

for (int i = 0; i < kvs\_.size(); i++){

cout << kvs\_[i].key << "->";

set<string>::iterator it = kvs\_[i].projects.begin();

int cnt = 0;

for (it; it != kvs\_[i].projects.end(); it++)

{

cout << \*it;

cnt++;

if (cnt != kvs\_[i].projects.size())

cout << '|';

}

cout << endl;

}

}

void RM::get\_kvs() {

cout<<"输入样例:\nS->Qc|c\nQ->Rb|b\nR->Sa|a\n";

cout<<"以$为结束标志:\n";

string str;

while (cin >> str && str[0] != '$')

{

kvs\_.push\_back(split(str));

}

}

kvs RM::split(string str)

{

string key;

set<string>projects;

string tmp = "";

for (int i = 0; i < str.size(); i++)

{

if (str[i] == '>')continue;

if (str[i] == '-')

{

key = tmp;

tmp = "";

continue;

}

if (str[i] == '|')

{

projects.insert(tmp);

tmp = "";

continue;

}

tmp += str[i];

}

if (tmp != "")projects.insert(tmp);

vector<kvs>ret;

return { key,projects };

}

void RM:: get\_not\_end() {

set<string>keys;

for (auto kv : kvs\_) {

keys.insert(kv.key);

}

for (auto k : keys) {

not\_end\_keys\_.push\_back(k);

cout << k << endl;

}

}

bool RM::check\_all\_end(string str){

for (auto c : str) {

cout << c << endl;

if (c >= 65 && c<=90)return false;

}

return true;

}

void RM::rm\_direct\_recurse(int pos)

{

set<string>::iterator it = kvs\_[pos].projects.begin();

bool flag = true;

for (it; it != kvs\_[pos].projects.end(); it++)

{

string str = \*it;

if ((str.find(kvs\_[pos].key) == str.rfind(kvs\_[pos].key) && str.find(kvs\_[pos].key) == 0) || check\_all\_end(str))

flag = true;

else

{

flag = false;

break;

}

}

if (!flag)return;

vector<string>va, vb;

for (it = kvs\_[pos].projects.begin(); it != kvs\_[pos].projects.end(); it++)

{

string str = \*it;

if (str.find(kvs\_[pos].key) == -1)

{

vb.push\_back(str);

}

else

{

str.erase(0, kvs\_[pos].key.size());

va.push\_back(str);

}

}

kvs\_[pos].projects.clear();

for (int i = 0; i < vb.size(); i++)

{

kvs\_[pos].projects.insert(vb[i] + kvs\_[pos].key + "'");

}

set<string>ans;

for (int i = 0; i < va.size(); i++)

{

ans.insert(va[i] + kvs\_[pos].key + "'");

}

ans.insert("Σ");

struct kvs N = { kvs\_[pos].key + "'",ans };

kvs\_.push\_back({ kvs\_[pos].key + "'",ans });

}