实验报告

实验题目：合一算法

实验目的：加深对合一算法的理解和熟悉求解最一般合一的过程

实验内容：对两个原子谓词公式进行判断是否存在最一般合一，存在，求解，不存在，输出不存在最一般合一信息

实验代码及结果：

#include <iostream>

#include <string>

#include <vector>

using namespace std;

struct transform //一组置换

{

string t\_f1;

string t\_f2;

};

//函数声明

bool same(const string f1,const string f2) ;

transform dif(const string f1,const string f2);

string change(string f,transform q);

string change2(string f,transform q);

bool syncretism(const string f1,const string f2, vector<transform> & );

int legal(transform &);

bool var(const string s);

string varData(string s);

int main()

{

cout<<"const:capital\t"<<"varible:lowercase."<<endl;

//输入两个谓词公式

string f1,f2;

cout<<"intput F1:";

cin>>f1;

cout<<"intput F2:";

cin>>f2;

vector <transform> mgu;

if(syncretism(f1,f2,mgu))//存在最一般合一，并输出结果

{

cout<<"mgu={ ";

int i=0;

for(i=0;i<mgu.size()-1;i++)

cout<<mgu[i].t\_f1<<"/"<<mgu[i].t\_f2<<", ";

cout<<mgu[i].t\_f1<<"/"<<mgu[i].t\_f2<<" }"<<endl;

}

else //不存在最一般合一

{

cout<<"cannot be syncretized"<<endl;

}

return 0;

}

bool syncretism (const string tf1,const string tf2,vector<transform> &mgu)//合一方法，判断是否可进行合一

{

string f1=tf1, f2=tf2;

while(!same(f1,f2))//f1与f2中的符号不完全相同时才进入while循环

{

transform t=dif(f1,f2);//得到f1和f2的一个差异集，并把它赋给t

int flag=legal(t);

if(flag==0)

return false;

else

{

mgu.push\_back(t);

f1=change(f1,mgu.back());

f2=change(f2,mgu.back());

cout<<"after change:"<<endl;

cout<<"f1:"<<f1<<endl;

cout<<"f2:"<<f2<<endl;

if(same(f1,f2)) break;//f1和f2相同时就停止循环

}

}

return true;

}

bool same(const string f1, const string f2) //判断两个谓词f1和f2是否相同

{

if(f1.length()==f2.length())

{

int i=0;

while(i<f1.length()&&f1.at(i)==f2.at(i))

i++;

if(i==f1.length())

return true;

else

{

return false;

}

}

else return false;

}

transform dif(const string f1,const string f2)//求解f1和f2的差异集

{

int i=0;

transform t;

while(f1.at(i)==f2.at(i))//第i个相等时就转向比较i+1，直到遇到不相等时就跳出while循环

i++;

int j1=i;

while(j1<f1.length()-1&&f1.at(j1)!=',')//从不相等的部分开始，直到遇到‘，’或到达结尾时跳出while循环

j1++;

if(j1-i==0) return t;

t.t\_f1=f1.substr(i,j1-i);//将f1中的不相同的项截取出来放入变量t.t\_f1中

int j2=i;

while(j2<f2.length()-1&&f2.at(j2)!=',')

j2++;

if(j2-i==0) return t;

t.t\_f2=f2.substr(i,j2-i);//将f2中的不相同的项截取出来放入变量t.t\_f2中

while(t.t\_f1[j1-i-1]==t.t\_f2[j2-i-1])//去除相同的部分

{

t.t\_f1.erase(j1-1-i);

t.t\_f2.erase(j2-i-1);

j1--;

j2--;

}

return t;

}

int legal(transform &t)//判断置换t是否合法

{

if(t.t\_f1.length()==0||t.t\_f2.length()==0)

return 0;

if(var(t.t\_f2))

{

if(var(t.t\_f1)&&(varData(t.t\_f1)==varData(t.t\_f2)))//不能代换合一

return 0;

else

return 2;

}

if(!var(t.t\_f1))//若t\_f1和t\_f2都不是变量，也不能合一

return 0;

string temp;

temp=t.t\_f1;

t.t\_f1=t.t\_f2;

t.t\_f2=temp;//在t\_f1是变量而t\_f2不是变量的情况下，交换t\_f1和t\_f2

return 1;

}

string varData(string s)//该函数是剥去外层括号

{

if(s.length()==1||s.length()==0)

return s;

if(s.length()>1)

{

int i=0;

while(i<s.length()&&s.at(i)!='(')

i++;

int j=i;

while(j<s.length()&&s.at(j)!=')')

j++;

string ss=s.substr(i+1,j-i-1);

return varData(ss);

}

else

{

return false;

}

}

bool var(const string s)

{

if(s.length()==0) return false;

if(s.length()==1&&s[0]>='A'&&s[0]<='Z')

return false;

if(s.length()>1)

{

int i=0;

while(i<s.length()&&s.at(i)!='(')//略过不是'('的字符

i++;

int j=i;

while(j<s.length()&&s.at(j)!=')')//略过')'前的字符

j++;

string ss=s.substr(i+1,j-i-1);//取出'('和')'之间的串

return var(ss);//递归调用var

}

else

{

return true;

}

}

string change(string f,transform q)//该函数查找t\_f2在f中的位置并用t\_f1替代f中相应的t\_f2

{

int i=f.find(q.t\_f2);

while(i<f.length())

{

i=f.find(q.t\_f2);

if(i<f.length())

f=f.replace(i,q.t\_f2.length(),q.t\_f1);

}

return f;

}

