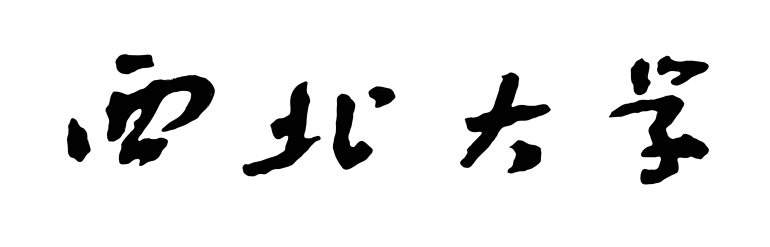
****

****

**人工智能实践课实验报告**

**题 目： 合一算法**

**年 级： 2019**

**学 号： 2019111137**

**专 业： 　软件工程**

**姓 名： 杨润林**

1. **实验内容**

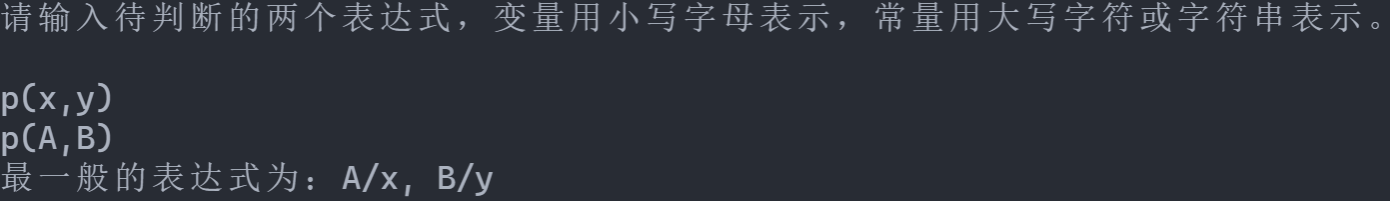
**用java实现合一算法**

1. **解决思路**

**合一函数unify(String E1,String E2)用来判断两个表达式是否可以合一，可以则返回一般合一式，不行则返回Failed。**

**合一算法的关键是实现element这个数据结构，可以是列表，可以是变量、常量、函数，可以是空。我首先定义了函数getElements(String E)，来获取表达式中的元素。还需要判断一个元素是常量还是变量，我定义了isVariable(String E),isConstants(String E).一个比较重要的函数是apply(String subs1, String E, List<String> list),用来把一个字符串中一个子串换为另一个字符串。**

1. **实验结果**



1. **总结**

**在编程求解问题时，模块化的思想比较实用，通过把功能分解为许多个比较容易实现的函数，可以使程序结构更清晰，编码难度降低。**

**附录（代码）：**

package unify;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Unify {

    private static String EMPTY = " ";

    private static String FAIL = "Failed";

    private static List<String> list1 = new ArrayList<String>();

    private static List<String> list2 = new ArrayList<String>();

    @SuppressWarnings("resource")

    public static void main(String[] args){

        //输入两个谓词公式

        Scanner input = new Scanner(System.in);

        while(true) {

        System.out.println("请输入待判断的两个表达式，变量用小写字母表示，常量用大写字符或字符串表示。\n");

        String E1 = input.nextLine();

        list1 = Unify.getElements(E1);

        String E2 = input.nextLine();

        list2 = Unify.getElements(E2);

//        System.out.println("list1="+list1.toString());

        String res=Unify.unify(E1, E2);

//        if(res.contains("Failed")) {

//          System.out.println("Failed");

//        }

//        else

        System.out.println("最一般的表达式为：" +res);}

    }

    static List<String> getElements(String E){

        List<String> list = new ArrayList<String>();

        for(int i = 0; i < E.length(); i++){

            if(E.charAt(i) == ')')

                list.add("()");

            if((E.charAt(i) >= 'a' && E.charAt(i) <= 'z') || (E.charAt(i) >= 'A' && E.charAt(i) <= 'Z')){

                int j = i+1;

                while(j < E.length() && ((E.charAt(j) >= 'a' && E.charAt(j) <= 'z') || (E.charAt(j) >= 'A' && E.charAt(j) <= 'Z')))

                    j++;

                list.add(E.substring(i,j));//取出'('和')'之间的串

                i = j - 1;

            }

        }

        return list;

    }

    private static String unify(String E1, String E2){

         if((E1.isEmpty() && E2.isEmpty())||(Unify.isConstants(E1) && Unify.isConstants(E2)))

         {

             if(E1.equals(E2))

                 return Unify.EMPTY;

             else

                 return Unify.FAIL;

         }

         else if(Unify.isVariable(E1)){

             //contains()方法,当且仅当此字符串包含指定的char值序列时,返回true

             if(!E2.equals(E1) && E2.contains(E1))

                 return Unify.FAIL;

             else{

                 if(E2.equals(E1))//判断是否相等

                     return Unify.EMPTY;

                 else

                     return E2 + "/" + E1;

             }

         }

         else if(Unify.isVariable(E2)){

             if(!E1.equals(E2) && E1.contains(E2))

                 return Unify.FAIL;

             else{

                 if(E2.equals(E1))

                     return Unify.EMPTY;

                 else

                     return E1 + "/" + E2;

             }

         }

         else if(E1.isEmpty() || E2.isEmpty())

             return Unify.FAIL;

         else{

             String he1 = Unify.getFirstElement(list1);

             E1 = Unify.getRest(E1, he1);

             String he2 = Unify.getFirstElement(list2);

             E2 = Unify.getRest(E2, he2);

             System.out.println("he1="+he1);

             System.out.println("he2="+he2);

             String subs1 = Unify.unify(he1, he2);

             System.out.println("subs1="+subs1);

             if(subs1.equals(Unify.FAIL))

                 return Unify.FAIL;

             String te1 = Unify.apply(subs1, E1, list1);

             String te2 = Unify.apply(subs1, E2, list2);

             String subs2 = Unify.unify(te1, te2);

             System.out.println("subs2="+subs2);

             if(subs2.equals(Unify.FAIL))

                 return Unify.FAIL;

             else

                 return Unify.composition(subs1, subs2);

         }

    }

    //判断常量

    private static boolean isConstants(String E){

        if(!E.isEmpty()){

            if(E.length() > 1){

                if(E.equals("()"))

                    return true;

                if(E.contains(" ") || E.contains("(") || E.contains(")") || E.contains(","))

                    return false;

                else

                    return true;

            }

            else{

                if(E.charAt(0)>='A' && E.charAt(0) <= 'Z')

                    return true;

                else

                    return false;

            }

        }

        return false;

    }

    //判断变量

    private static boolean isVariable(String E){

        if(!E.isEmpty()){

            if(E.length() > 1){

                if(E.contains(" ") || E.contains("(") || E.contains(")") || E.contains(","))

                    return false;

                else

                    return true;

            }

            else{

                if(E.charAt(0)>='a' && E.charAt(0) <= 'z')

                    return true;

                else

                    return false;

            }

        }

        return false;

    }

    private static String getFirstElement(List<String> list){

        String element = list.get(0);

        list.remove(0);

        return element;

    }

    //该函数查找s2在E中的位置并用s1替代E中相应的s2

    private static String apply(String subs1, String E, List<String> list){

        if(subs1.equals(Unify.EMPTY))

            return E;

        String sub1 = subs1.substring(0, subs1.indexOf('/'));

        String sub2 = subs1.substring(subs1.indexOf('/') + 1, subs1.length());

        E = E.replaceAll(sub2, sub1);

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

            if(list.get(i).equals(sub2))

                list.set(i, sub1);

        }

        return E;

    }

    private static String composition(String subs1, String subs2){

        if(subs1.equals(Unify.EMPTY) && subs2.equals(Unify.EMPTY))

            return Unify.EMPTY;

        else if(subs1.equals(Unify.EMPTY))

            return subs2;

        else if(subs2.equals(Unify.EMPTY))

            return subs1;

        else

            return subs1 + ", " + subs2;

    }

    private static String getRest(String E, String firstElement){

        if(!E.isEmpty()){

            String e;

            if(firstElement.equals("()"))

                e = E.replaceFirst("\\(\\)","0");

            else

                e = E.replaceFirst(firstElement,"0");

            if(e.length() == 1)

                return "";

            int i = e.indexOf('0');

            int j = i + 1;

            while(e.charAt(j) == ' ' || e.charAt(j) == ',')

                j++;

            e = e.substring(0, i) + e.substring(j);

            return e;

        }

        return null;

    }

}