
# 基于示例演化的学生程序自动修复

王甜甜，许家欢，王克朝, 苏小红

# CONTENT
# 目录

1

# PART ONE

## 研究背景及现状

01 背景和研究意义

02 国内外研究现状

03 基于模板的错误定位

# 课题背景和研究意义

如何自动化地评价学生的学习效果、提供充分的反馈,与学生互动?

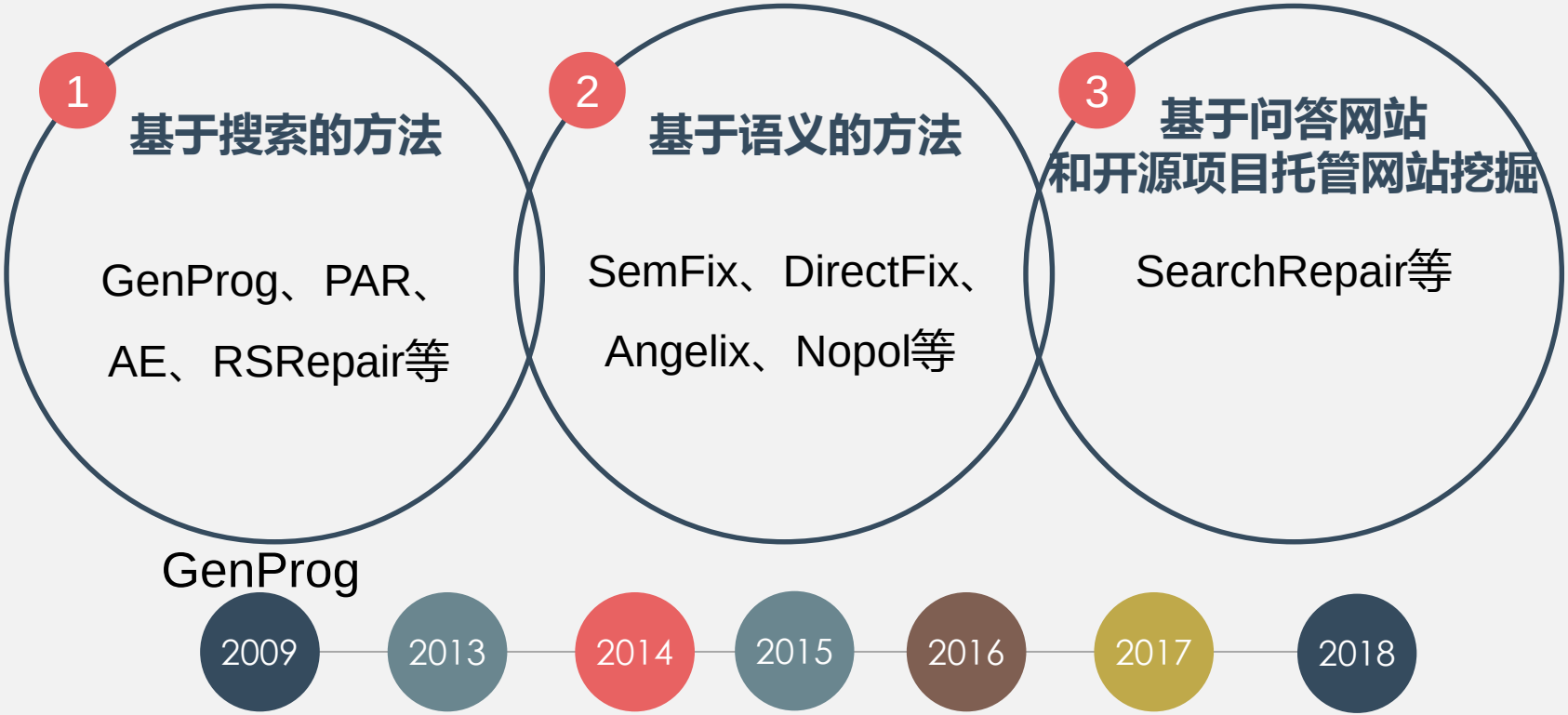
## 课题背景

在我国目前大学生的编程教育中，mooc课堂、acm刷题系统、各学校的**代码评审系统**非常流行。当前该类系统仅告知学生程序不正确，但不能提示错误的**具体位置**和程序的**修改方案**。而大量且多样化的学生程序给教师人工检查带来了一定困难和时间消耗。

## 研究意义

**研究学生程序自动修复方法**，将教师提供的模板程序作为示例，不仅可以对学生的程序自动测试和提供**修复参考**，还可以减少教师的负担，同时为学生提供及时反馈，辅助其调试程序。

# 国内外研究现状-工业软件的自动修复





[1] Zhong H, Su Z. An empirical study on real bug fixes. IEEE/ACM, IEEE International Conference on Software Engineering. Firenze, Italy, NJ: IEEE, 2015: 913-923.
[2]玄跻峰, 任志磊, 王子元, 谢晓园, 江贺. 自动程序修复方法研究进展. 软件学报, 2016, 27(4):771-784.
[3]王赞, 郜健, 陈翔, et al. 自动程序修复方法研究述评[J]. 计算机学报, 2018(3)：588-610

# 国内外研究现状-工业软件修复方法不适合直接应用于修复学生程序

intelligent tutoring system for programming(ITSP)


对于学生在入门程序设计课程中实际编写的程序，这些工具的**修复率为31%。**

Yi J, Ahmed UZ, Karkare A, Tan SH, Roychoudhury A. A feasibility study of using automated program repair for introductory programming assignments, Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, Paderborn, Germany, NJ: IEEE, 2017:740-751.

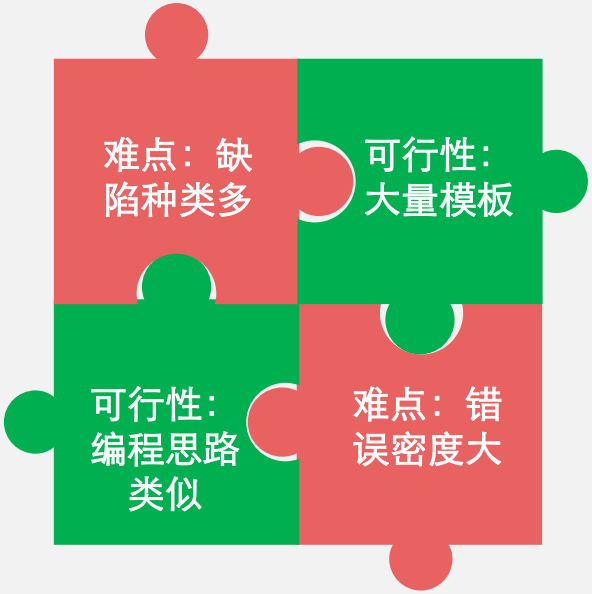
# 学生程序的特点

**难点1：缺陷种类多**

可能含有概念错误,甚至可能缺少关键语句。

**可行性2：编程思路类似**

由于学生参加相同的课程,编写程序的思路甚至所犯的错误通常也相似。




**可行性1：大量模板**

完整的规格说明是已知的。MOOC中学生提交了大量的正确程序。

**难点2：缺陷密度大**

可能对所有的测试用例都执行失效。**工业软件调试中“缺陷程序存在部分成功执行”的假设,不完全适用于学生程序调试。**

2

PART TWO

# 基于示例演化的学生程序自动修复研究框架

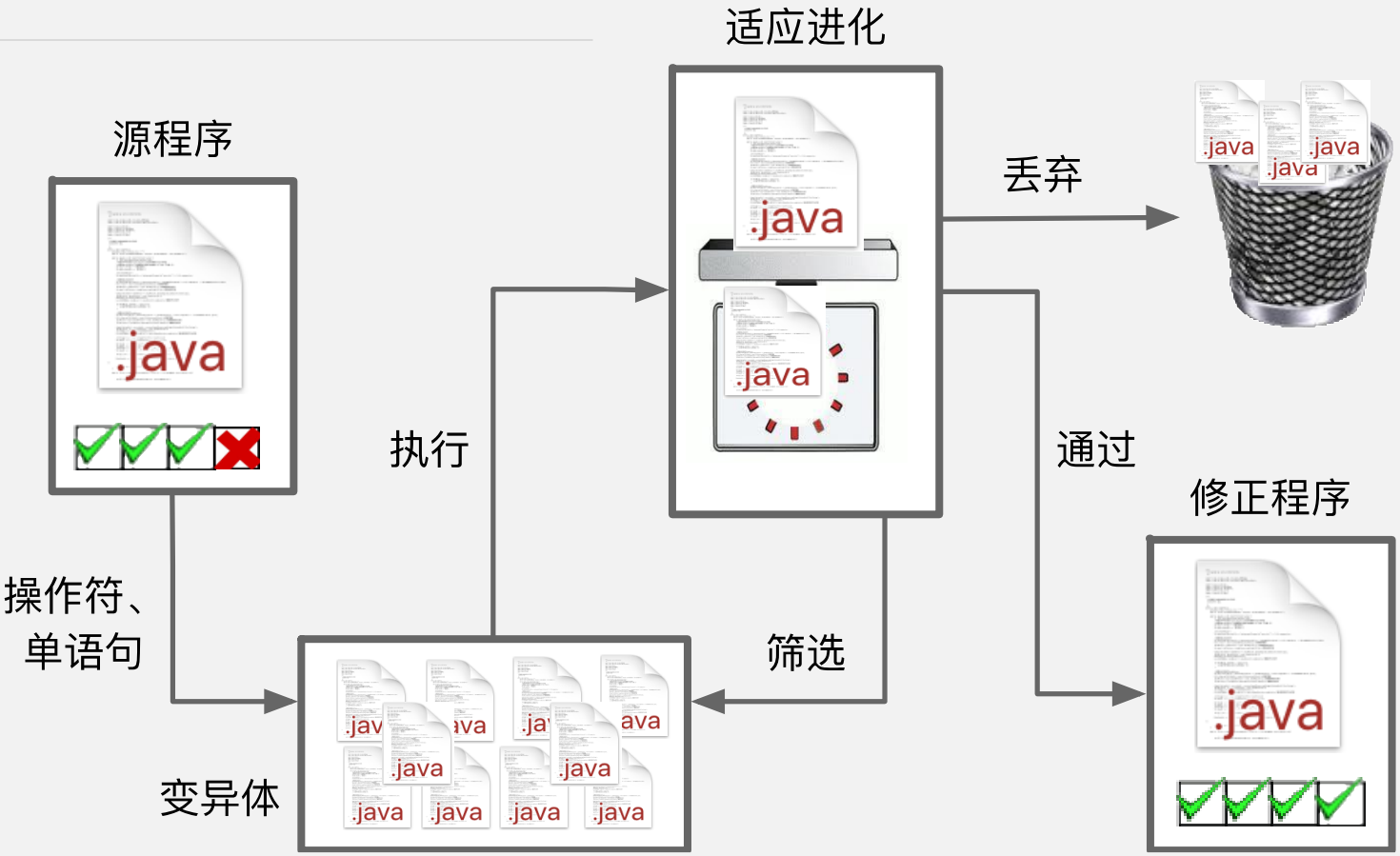
01 遗传编程算法研究

02 针对学生程序对遗传编程算法的改进

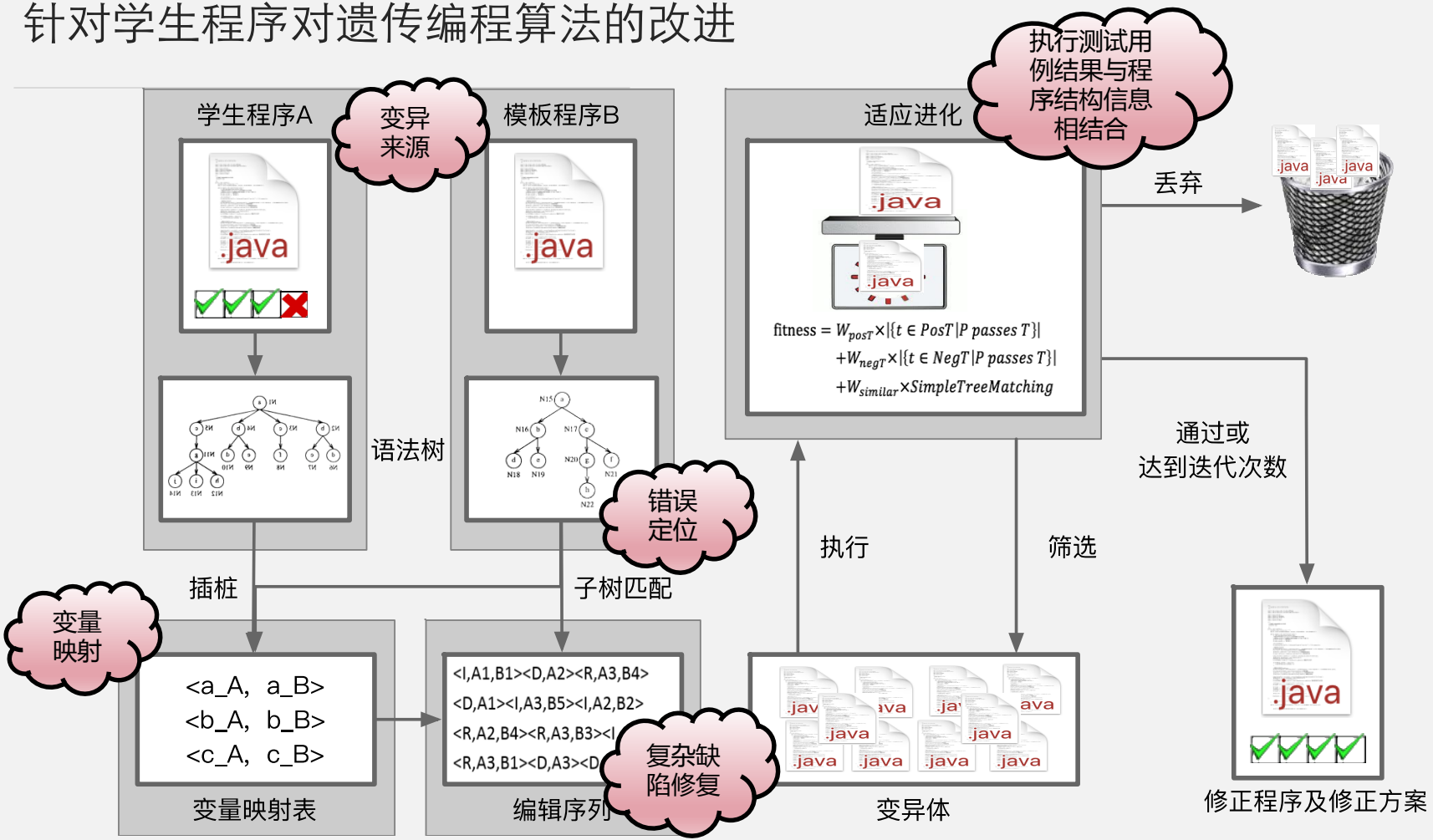
03 总体研究框架

# 遗传编程算法研究

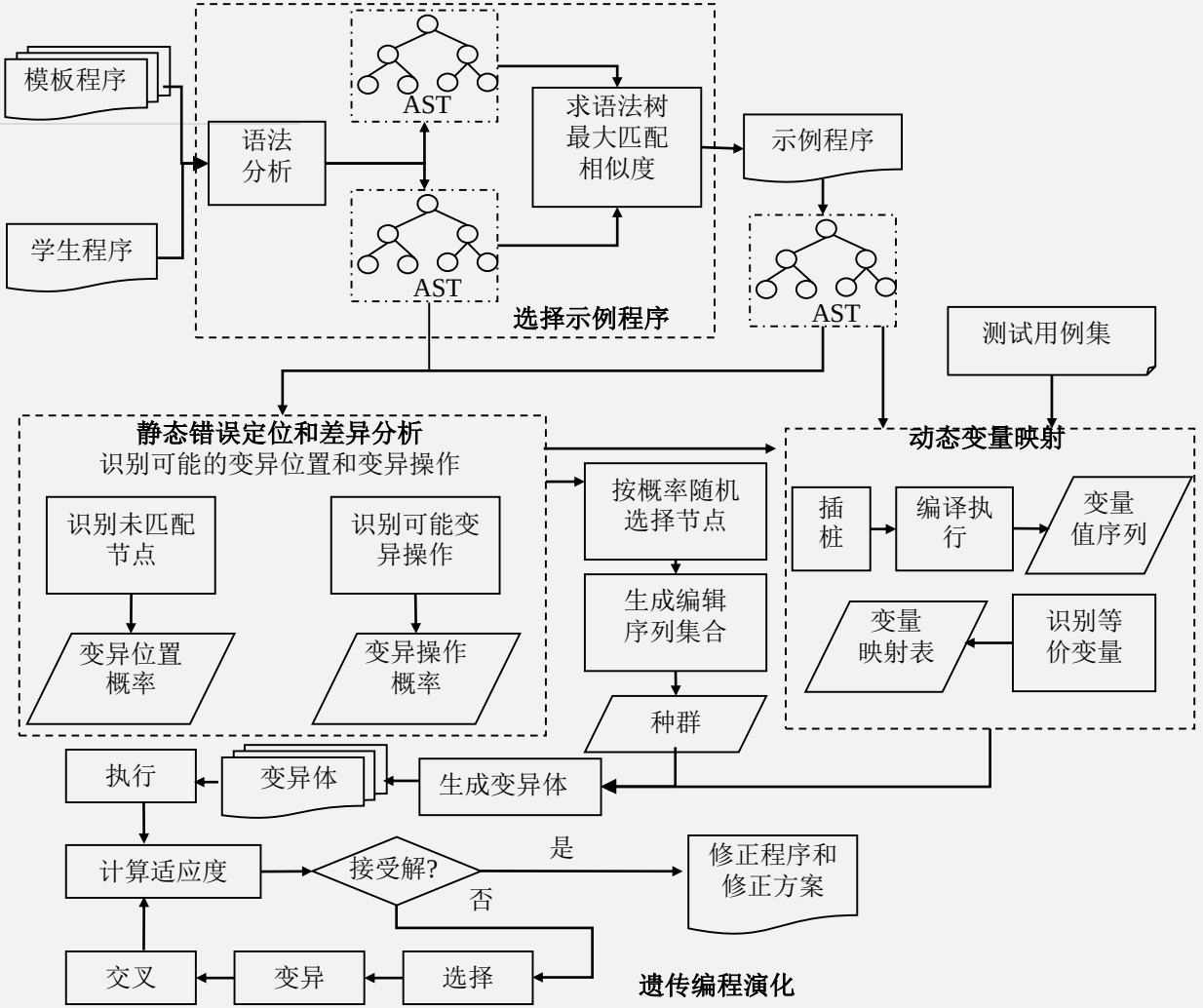
GenProg



# 针对学生程序对遗传编程算法的改进



# 总体研究框架


模板程序
学生程序
语法分析
AST
AST
求语法树最大匹配相似度
选择示例程序
示例程序
AST
测试用例集
静态错误定位和差异分析
识别可能的变异位置和变异操作
识别未匹配节点
识别可能变异操作
变异位置概率
变异操作概率
按概率随机选择节点
生成编辑序列集合
种群
动态变量映射
插桩
编译执行
变量值序列
变量映射表
识别等价变量
执行
变异体
生成变异体
计算适应度
接受解?
是
否
修正程序和修正方案
交叉
变异
选择
遗传编程演化

3

# PART THREE

## 关键技术

01 代码多样化问题

02 缺陷程序和示例程序的结构语义和执行特征值差异识别和变量映射

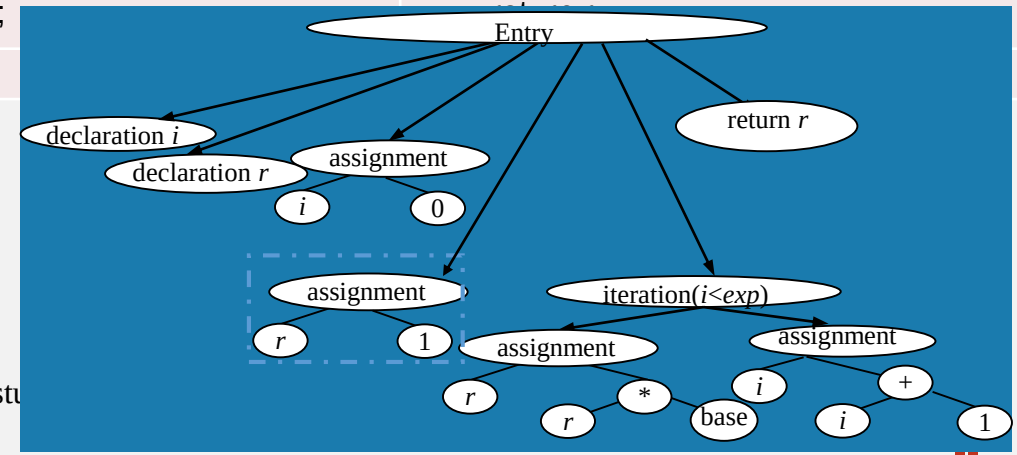
03 编辑序列

04 适应度计算

# 代码多样化问题

**通过结构语义分析，识别等价的语法结构和表达式等，减少所需要提供的模板，提高学生程序和示例程序差异分析的准确性.**

| Code A示例程序 | Code B与示例等价的程序 | Code C缺陷程序 |
|---|---|---|
| static public int fact( int base, int exp){ | static public int fact( int base, int exp){ | static public int fact( int base, int exp){ |
| int i, r=1; | int j, ret; | int j, r; |
| for( i=0; i<exp; i++) | j = 0; | j = 0; |
| r *= base; | ret = 1; | //缺陷,r未初始化 |
| return r; | while( exp> j){ | while( exp> j){ |
| } | j=j+1; | j=j+1; |
| | ret = ret * base;} | r = r * base;} |
| | return ret; | |
| | } | |



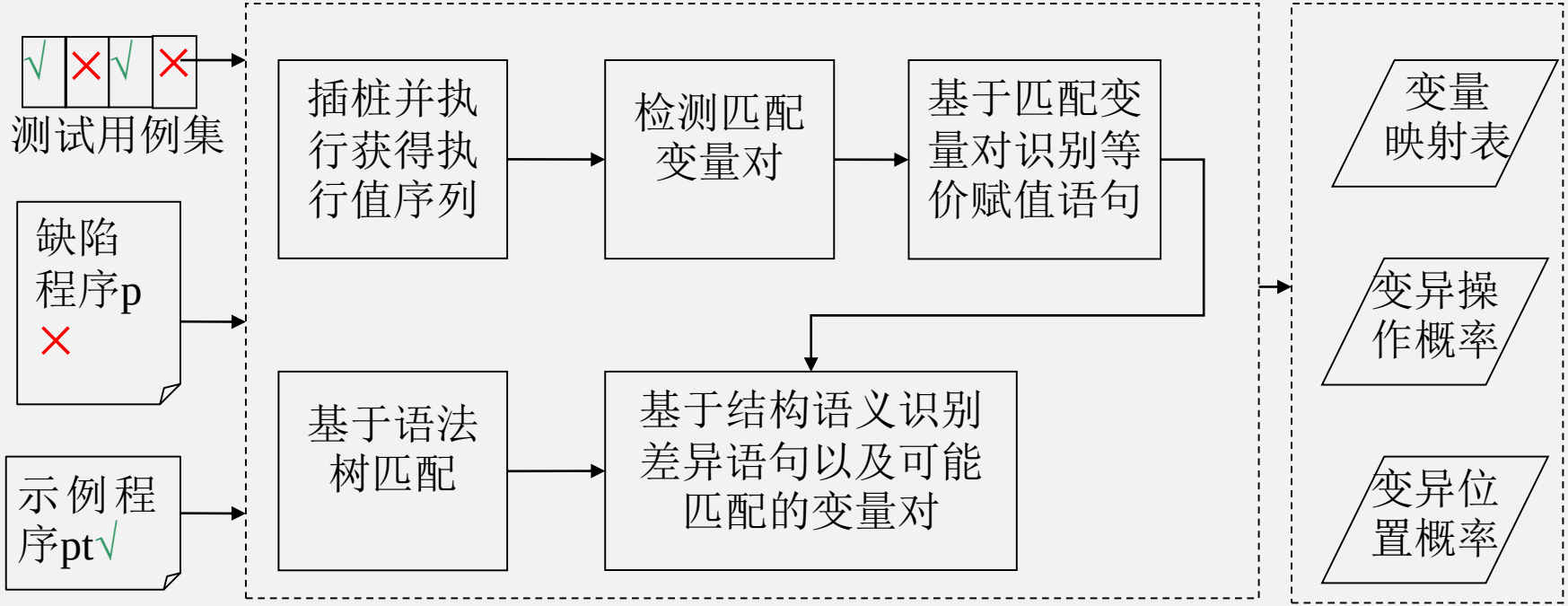

Wang T, Su X, Wang Y, et al. Semantic similarity-based grading of stu

# 代码多样化问题

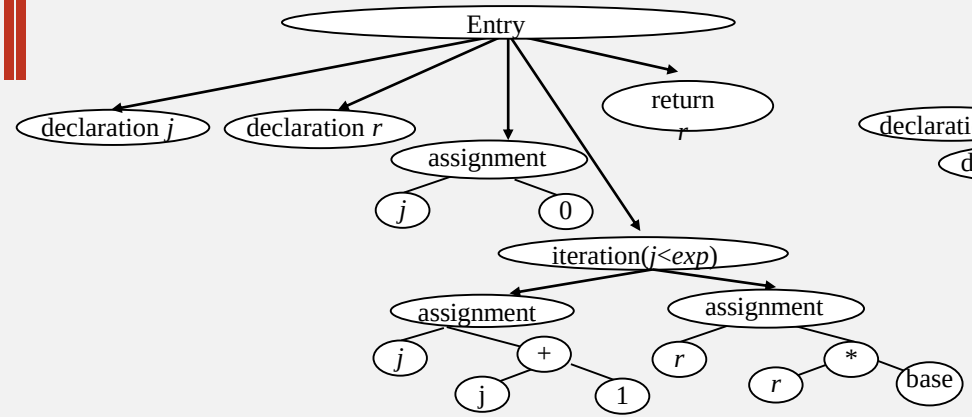
## 相同功能对应变量：执行值序列相似

| Code A示例程序 | Code B与示例等价的程序 | Code C缺陷程序 |
|---|---|---|
| static public int fact( int base, int exp){ | static public int fact( int base, int exp){ | static public int fact( int base, int exp){ |
| int i, r=1; | int j, ret; | int j, r; |
| for( i=0; i<exp; i++) | j = 0; | j = 0; |
| r *= base; | ret = 1; | //缺陷,r未初始化 |
| return r; | while( exp> j){ | while( exp> j){ |
| } | j=j+1; | j=j+1; |
| | ret = ret * base;} | r = r * base;} |
| | return ret; | return r; |
| | } | } |
| 调用fact(3, 4)时的执行值序列 | | |
| base 3 | base 3 | base 3 |
| exp 4 | exp 4 | exp 4 |
| i 0,1,2,3 | j 0,1,2,3 | j 0,1,2,3 |
| r 3, 9, 27, 81 | ret 3, 9, 27, 81 | r 随机数构成的序列 |

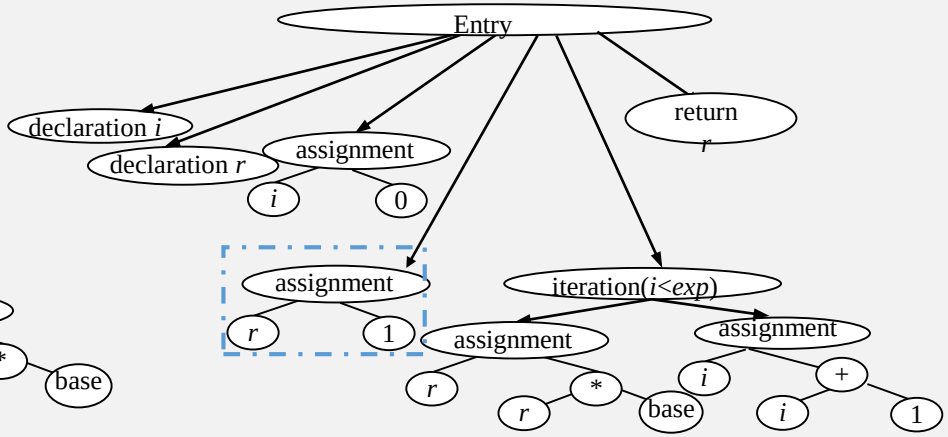
# 基于结构语义和执行特征值的差异识别和变量映射



缺陷学生程序C




示例程序A




节点序列<A1, A2, A3, A4, A5,…., An>

节点信息:
AST
Length
NodeType
Parent
Root
StartPosition
Class

节点序列<B1, B2, B3, B4, B5,…., Bm>

最长公共子序列算法

差异分析:

C: <A1, A2, A3, A4, A5,…., An>

A: <B1, B2, B3, B4, B5,…., Bm>

变异位置概率:

C: <W1, W1, W1, W1, W2,…,W1>

A: <W1, W1, W1, W1, W2,…., W1>

变异操作概率:

C1: < WI=0.33, WR=0.33, WD=0.33>

…

C4: <WI= 0.5, WR=0.25, WD=0.25>…

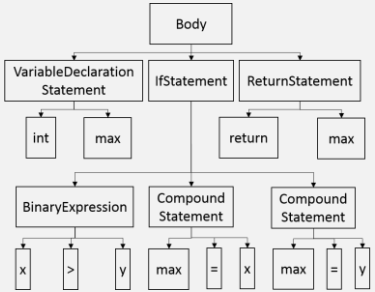
变量映射表：
(base,base)
(exp,exp)
(j,i)
(r,r)

$$\text{相同节点}W1 = \frac{1}{\text{相同节点数} + \text{不同节点数} \times 2}$$

$$\text{不同节点}W2 = \frac{x}{\text{相同节点数} + \text{不同节点数} \times 2}$$

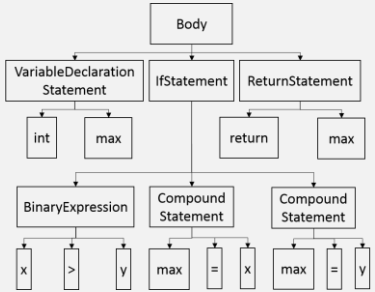
# 编辑序列

## 缺陷学生程序A




<A1,A2,A3,…,An>

## 示例程序B

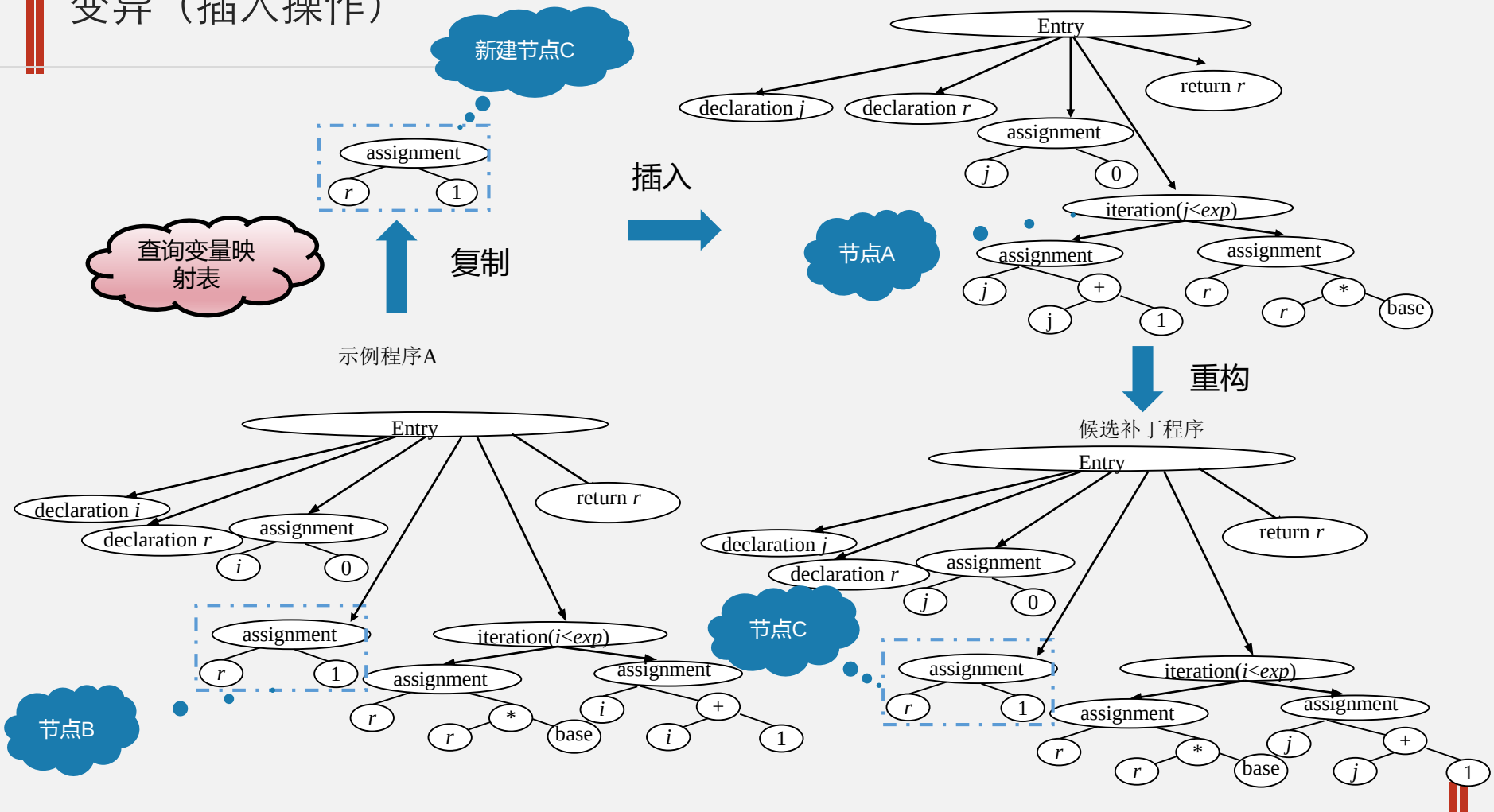



<B1,B2,B3,…,Bm>

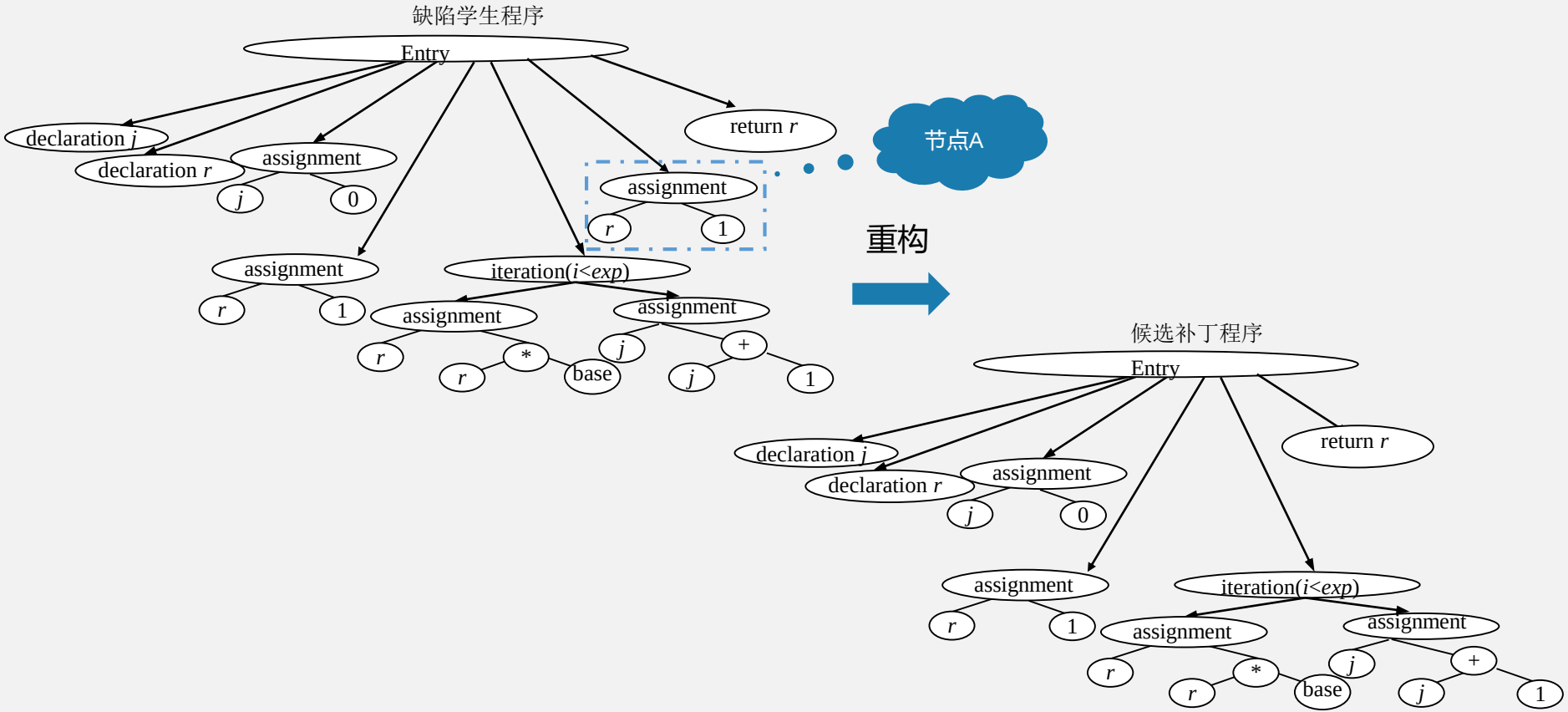
编辑序列：<I(A1,B2),D(A2),R(A3,B2)>

| 变异操作 | 动作 | 缺陷学生程序A | 模板程序B | 含义 |
|---|---|---|---|---|
| I(A1,B2) | Insert | A1 | B2 | 在学生程序A1节点后插入模板程序B2节点 |
| D(A2) | Delete | A2 | - | 删除学生程序中的A2节点 |
| R(A3,B2) | Replace | A3 | B2 | 将学生程序A3节点替换为模板程序B2节点 |

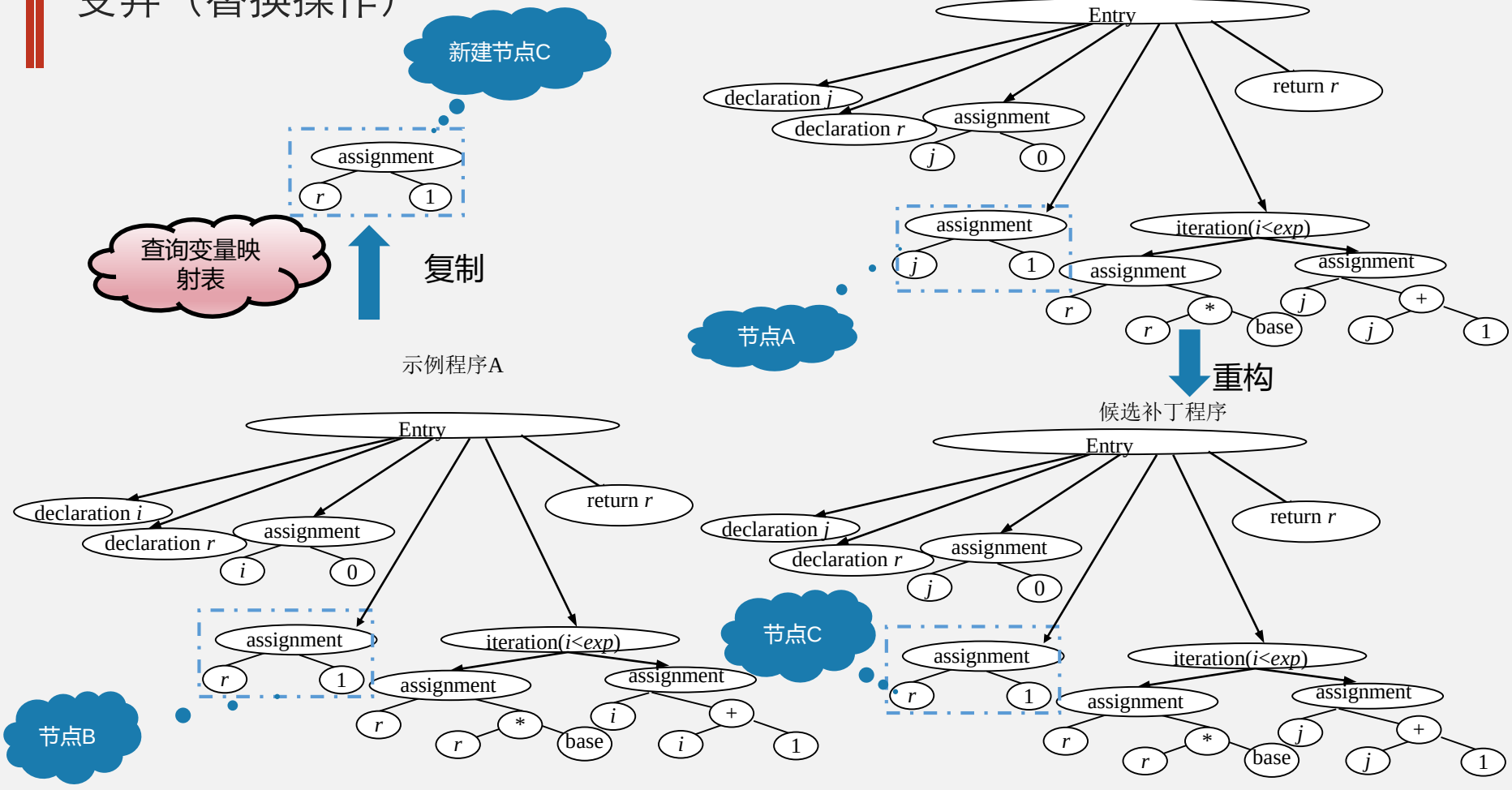
# 变异（插入操作）


新建节点C
assignment
r
1
查询变量映射表
复制
插入
缺陷学生程序C
Entry
declaration j
declaration r
assignment
return r
j
0
iteration(j<exp)
节点A
assignment
assignment
j
+
r
*
j
1
r
base
示例程序A
重构
候选补丁程序
Entry
declaration i
declaration r
assignment
return r
i
0
assignment
iteration(i<exp)
r
1
assignment
assignment
r
*
i
+
r
base
i
1
节点B
Entry
declaration j
declaration r
assignment
return r
j
0
节点C
assignment
iteration(i<exp)
r
1
assignment
assignment
r
*
j
+
r
base
j
1

# 变异（删除操作）


缺陷学生程序
Entry
declaration j
declaration r
assignment
j
0
return r
assignment
r
1
assignment
r
1
iteration(i<exp)
assignment
r
*
base
assignment
j
+
j
1
节点A
重构
候选补丁程序
Entry
declaration j
declaration r
assignment
j
0
return r
assignment
r
1
iteration(i<exp)
assignment
r
*
base
assignment
j
+
j
1

# 变异（替换操作）


新建节点C
assignment
r
1
查询变量映射表
复制
缺陷学生程序
Entry
declaration j
declaration r
assignment
return r
j
0
assignment
iteration(i<exp)
j
1
assignment
assignment
r
*
j
+
r
base
j
1
节点A
重构
示例程序A
候选补丁程序
Entry
declaration i
declaration r
assignment
return r
i
0
assignment
iteration(i<exp)
r
1
assignment
assignment
r
*
i
+
r
base
i
1
节点B
节点C

# 交叉

group：

<I(A1,B2), D(A2), R(A3,B2), I(A4,B2), R(A1,B1)>

<I(A2,B1), D(A2), R(A3,B2), R(A4,B2), D(A1)>

从种群中按适应度概率选择编辑序列

<I(A1,B2), D(A2), | R(A3,B2), I(A4,B2), R(A1,B1)> 序列1

<I(A2,B1), D(A2), R(A3,B2), | R(A4,B2), D(A1)> 序列2

交叉

<I(A1,B2), D(A2), R(A4,B2), D(A1)>

<I(A2,B1), D(A2), R(A3,B2), R(A3,B2), I(A4,B2), R(A1,B1)>

group：

<I(A1,B2), D(A2), R(A3,B2), I(A4,B2), R(A1,B1)>

<I(A2,B1), D(A2), R(A3,B2), R(A4,B2), D(A1)>

<I(A1,B2), D(A2), R(A4,B2), D(A1)>

<I(A2,B1), D(A2), R(A3,B2), R(A3,B2), I(A4,B2), R(A1,B1)>

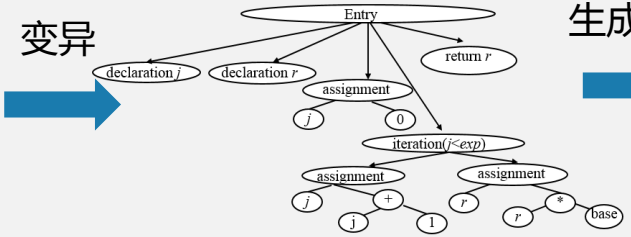
保证种群规模不变和变异体的多样性

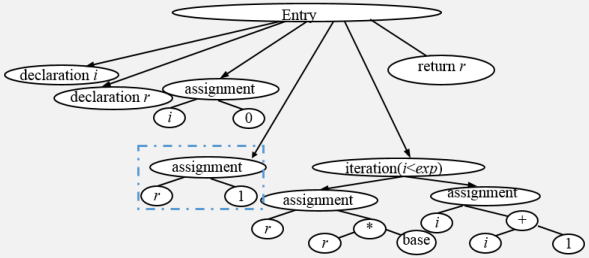
# 适应度计算

变量映射表：
(base,base)
(exp,exp)
(j,i)
(r,r)

group:
<I(A1,B2), D(A2), R(A3,B2), I(A4,B2), R(A1,B1)>
<I(A2,B1), D(A2), R(A3,B2), R(A4,B2), D(A1)>
<I(A1,B2), D(A2), R(A4,B2), D(A1)>
<I(A2,B1), D(A2), R(A3,B2), R(A3,B2), I(A4,B2), R(A1,B1)>

变异




语法结构匹配

生成候选补丁

```
static public int fact( int base, int exp){
        int j, ret;
        j = 0;
        ret = 1;
        while( exp> j){
          j=j+1;
          ret = ret * base;}
        return ret;
    }
```

执行

0 0
2 3
4 3

测试结果

0√
28 √
64 √

计算适应度

$$\text{fitness} = W_{posT} \times |\{t \in PosT | P \text{ passes } T\}| + W_{negT} \times |\{t \in NegT | P \text{ passes } T\}| + W_{similar} \times SimpleTreeMatching$$

4

# PART FOUR

## 自动修正原型系统的实现和测试

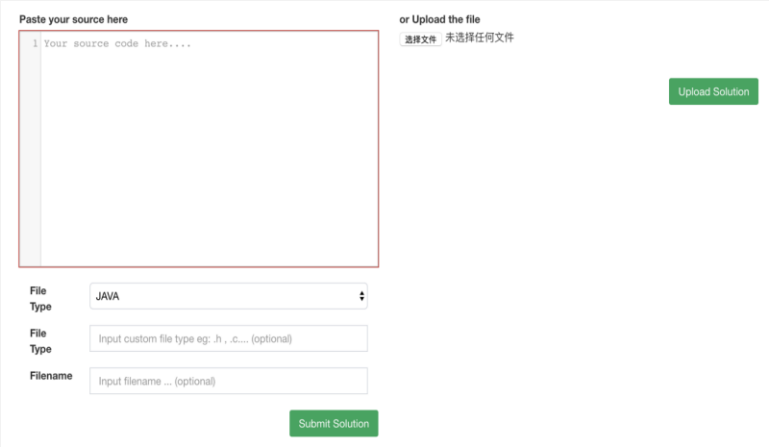
01 功能测试

02 性能测试

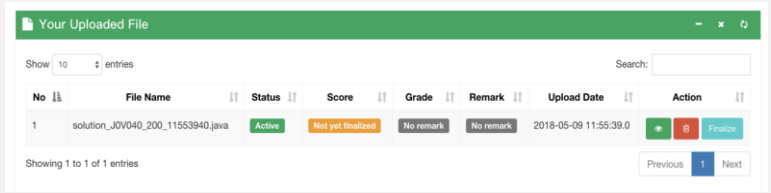
03 结果分析

# 原型系统（学生用户）

上传答案




自动修正




自动修正功能集成到程序评分系统上，学生上传代码进行自动修正

# 原型系统（学生用户）

修正结果

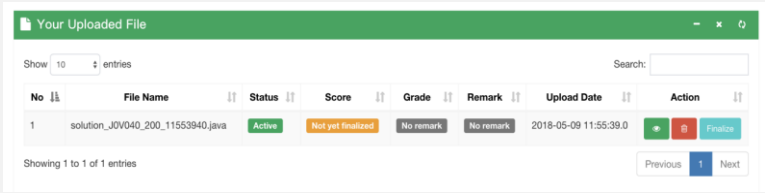
Run Result:

```
import java.text.DecimalFormat;
import java.util.Scanner;

public class sum_ori{

        private static final DecimalFormat df = new DecimalFormat("#.#####");

        public static Double calculate(Double f1, Double f2, Double f3){

                Double d=(30 * f1 + 20 * f2 + 50 * f3) / 100;



                return Double.valueOf(df.format(d));

        }


        public static void main(String[] args){

                Scanner sc = new Scanner(System.in);
                Double a = Double.valueOf(sc.next());
                Double b = Double.valueOf(sc.next());
                Double c = Double.valueOf(sc.next());

                Double ans = calculate(a,b,c);

                System.out.println(ans);
        }
}
model : Double d=(30 * f1 + 20 * f2 + 50 * f3) / 100;

sourseCode : Double d=(30 * f1 + 20 * f2 + 50 * f3) / 10;
```

*Run Time : 0 ms*

*resultGrade*

自动修正




得到修正程序和修正方案

# 可修正的缺陷类型

| 节点类型 | 节点名称 | 描述 |
| --- | --- | --- |
| VariableDeclarationStatement | 初始化 | 单语句 |
| ExpressionStatement | 表达式 | |
| IfStatement.expression | If条件表达式 | 条件表达式 |
| WhileStatement.expression | While循环条件表达式 | |
| ForStatement.expression | For循环条件表达式 | |
| SwitchStatement | Switch语句块 | 语句块 |
| IfStatement.thenStatement | If语句中的Then语句块 | |
| IfStatement.elseStatement | If语句中的Else语句块 | |
| WhileStatement.body | While循环语句块 | |
| ForStatement.body | For循环语句块 | |
| ReturnStatement | 返回值语句 | 返回值 |
| Package | 包的加载 | 函数调用包 |

# 实验数据

从“赛码网”上选取真实学生编程题目

| 题目名称 | 模板数量 | 错误版本数 | 测试用例数 | 描述 |
| --- | --- | --- | --- | --- |
| 回文串 | 10 | 10 | 15 | 判断能否添加字符成为回文串 |
| 研究生考试 | 10 | 10 | 11 | 4科总分判断录取情况 |
| 上台阶 | 10 | 10 | 13 | 上台阶的走法 |
| 公交车乘客 | 10 | 10 | 10 | 上下车多次后公交车乘客数 |
| 分苹果 | 10 | 10 | 15 | 苹果分堆求原始个数 |
| 击鼓传花 | 10 | 10 | 20 | 花n次传回第一个人手中的方法数 |
| 字符判断 | 10 | 10 | 15 | 判断字符串b的所有字符是否都在字符串a中出现过 |
| 股神 | 10 | 10 | 10 | 股票涨跌n天后股值 |
| 刮刮卡兑换 | 10 | 10 | 15 | 刮刮卡最多兑换多少赠品 |
| 日期倒计时 | 10 | 10 | 15 | 计算出今天距离未来的某一天还剩多少天 |

# 排除无法修正的缺陷类型

**函数参数个数不同**

相同功能的函数可能含有不同个数的参数，而程序自动修正只能修正表达式函数调用表达式，所以函数的参数个数不同会导致变异体无法顺利执行

**变量映射不准确**

因为学生程序的错误变量执行值序列和模板程序的变量执行值序列可能出现完全不相同的情况，在学生程序中有而模板程序中没有的变量，此时变量映射会出现偏差，而变量名映射不准确会导致学生程序无法正确执行

**多文件程序修复**

当前只允许学生上传单文件程序代码，其中含有输入输出和main函数，可以直接编译运行，不允许多个文件之间的函数调用

**编译未通过**

无法执行测试用例，如果程序不能执行测试用例则无法对学生程序进行插桩获得变量值序列，进而无法得到变量映射表

# 性能测试

## 学生程序“回文串”测试结果

| 版本 | 缺陷数量 | 程序规模 | 缺陷类型 | 对应模板 | 种群规模 | 适应度(%) | 是否修正 | 运行时间(s) |
|---|---|---|---|---|---|---|---|---|
| V1 | 1 | 43 | If条件语句 | M3 | 50 | 100 | 是 | 11.414 |
| V2 | 1 | 36 | Then语句块 | M2 | 50 | 95.83 | 是 | 14.393 |
| V3 | 1 | 32 | 表达式 | M8 | 50 | 100 | 是 | 13.836 |
| V4 | 2 | 42 | 返回值、表达式 | M9 | 70 | 94.08 | 是 | 117.073 |
| V5 | 2 | 31 | 缺包、初始化 | M1 | 70 | 89.28 | 是 | 134.097 |
| V6 | 2 | 44 | If条件语句、返回值 | M6 | 70 | 83.93 | 是 | 155.473 |
| V7 | 2 | 35 | 缺包、缺包 | M10 | 70 | 90.17 | 是 | 128.329 |
| V8 | 3 | 53 | 初始化、初始化、初始化 | M4 | 100 | 43.97 | 否 | 502.306 |
| V9 | 3 | 47 | 表达式、初始化、then语句块 | M5 | 100 | 85.38 | 是 | 344.753 |
| V10 | 4 | 244 | 条件表达式、返回值、缺包、else语句块 | M7 | 100 | 63.79 | 否 | 643.628 |

# 缺陷个数与修正结果的关系

缺陷数量越少，修正准确率越高，时间越短

| 缺陷数量 | 版本数 | 修正版本数 | 平均种群规模 | 平均适应度(%) | 运行时间(s) |
|---|---|---|---|---|---|
| 1 | 30 | 30 | 50 | 93.37 | 17.863 |
| 2 | 40 | 39 | 70 | 79.42 | 159.722 |
| 3 | 20 | 14 | 100 | 62.58 | 409.216 |
| 4及以上 | 10 | 5 | 100 | 30.29 | 738.927 |

# 缺陷类型与修正结果的关系

排除无法修正的缺陷类型，修正准确性与可分析缺陷类型无关

| 节点类型 | 节点名称 | 缺陷出现次数 | 修正次数 |
| --- | --- | --- | --- |
| VariableDeclarationStatement | 初始化 | 18 | 15 |
| ExpressionStatement | 表达式 | 38 | 30 |
| IfStatement.expression | If条件表达式 | 21 | 18 |
| WhileStatement.expression | While循环条件表达式 | 14 | 10 |
| ForStatement.expression | For循环条件表达式 | 18 | 16 |
| SwitchStatement | Switch语句块 | 2 | 2 |
| IfStatement.thenStatement | If语句中的Then语句块 | 23 | 20 |
| IfStatement.elseStatement | If语句中的Else语句块 | 12 | 10 |
| WhileStatement.body | While循环语句块 | 16 | 13 |
| ForStatement.body | For循环语句块 | 18 | 14 |
| ReturnStatement | 返回值语句 | 15 | 12 |
| Package | 包的加载 | 15 | 13 |

# 种群规模和迭代次数与修正结果的关系

选择有4个缺陷的10个版本程序

增大种群规模，使变异体覆盖的缺陷范围更大，种类更多样，或增加迭代次数或多次执行，增加变异体的多样性

| 版本数量 | 迭代次数 | 种群规模 | 成功修复数量 | 平均适应度(%) | 运行时间(s) |
|---|---|---|---|---|---|
| 10 | 10 | 50 | 0 | 30.47 | 438.285 |
| 10 | 10 | 70 | 3 | 38.39 | 542.043 |
| 10 | 10 | 100 | 5 | 45.58 | 683.941 |
| 10 | 10 | 150 | 6 | 53.25 | 822.294 |
| 10 | 10 | 200 | 6 | 54.84 | 1028.028 |
| 10 | 5 | 100 | 3 | 35.35 | 428.954 |
| 10 | 7 | 100 | 3 | 38.59 | 537.851 |
| 10 | 10 | 100 | 5 | 45.58 | 683.941 |
| 10 | 15 | 100 | 5 | 43.12 | 835.935 |
| 10 | 20 | 100 | 5 | 45.83 | 1129.283 |

# 结果分析

01

## 缺陷数量

缺陷数量越少，修正准确率越高，时间越短

02

## 缺陷类型

排除无法修正的缺陷类型，修正准确性与可分析缺陷类型无关

03

## 种群规模

缺陷数量多可增大种群规模，使变异体覆盖的缺陷范围更大，种类更多样

04

## 迭代次数

缺陷数量多可增加迭代次数或多次执行，增加变异体的多样性

## 05 程序模板

如果学生程序与模板程序差异较大，可能无法生成最小修复，也可能无法生成补丁。

# 结论

## 改进遗传编程算法

包括遗传算法中适应度计算方式、变异体的生成方式和变异体的筛选，更适合学生程序修正。

遗传编程算法的基础上将示例程序中正确的逻辑结构转移到含有缺陷的学生程序中，并不断结合缺陷程序自身的语法和语义特征进行进化。

## 结构语义和执行特征值差异识别和变量映射

静态错误定位将修正范围定位到较小范围内，避免大量盲目变异操作；动态变量映射保证在子树挖掘替换中不会因变量名的改变使编译程序无法正确的编译运行。

# 未来工作研究方向

**针对C语言学生程序**

尝试针对C语言和其它语言程序，完善自动修正系统

**分析更多缺陷类型**

尝试分析更多缺陷类型，扩大修正范围，如函数调用、死循环等

**多文件修正**

扩大程序规模，实现多文件程序修正，应用到更多场景

**变量映射**

寻找使变量映射更加准确的方法

# 谢谢

# 请专家批评与指教