实验四：死代码检测

2024.4.18

中山大學
SUN YAT-SEN UNIVERSITY

实验四：死代码检测

- 实验平台配置
- 实验内容
- API介绍
- 作业提交

实验平台配置——Tai-e

- **按照 https://tai-e.pascal-lab.net/intro/overview.html 配置**
  - Download assignments at https://github.com/pascal-lab/Tai-e-assignments

实验平台配置——Tai-e

运行测试

```
void deadAssign() {
    int x = 1;
    int y = x + 2; // dead assignment
    int z = x + 3;
    use(z);
    int a = x; // dead assignment
}
```

配置完成后，运行Assignment测试是否配置成功，若配置成功将得到图中输出

```
-------------------- <DeadAssignment: void deadAssign()> (livevar) --------------------
[0@L4] x = 1; null
[1@L5] %intconst0 = 2; null
[2@L5] y = x + %intconst0; null
[3@L6] %intconst1 = 3; null
[4@L6] z = x + %intconst1; null
[5@L7] invokevirtual %this.<DeadAssignment: void use(int)>(z); null
[6@L8] a = x; null
[7@L8] return; null

-------------------- <DeadAssignment: void deadAssign()> (constprop) --------------------
[0@L4] x = 1; null
[1@L5] %intconst0 = 2; null
[2@L5] y = x + %intconst0; null
[3@L6] %intconst1 = 3; null
[4@L6] z = x + %intconst1; null
[5@L7] invokevirtual %this.<DeadAssignment: void use(int)>(z); null
[6@L8] a = x; null
[7@L8] return; null

-------------------- <DeadAssignment: void deadAssign()> (deadcode) --------------------
```

实验内容

死代码检测

- 不可达代码：
  - 控制流不可达代码：此类不可达代码可以利用CFG检测出来——遍历CFG并标记可达语句，遍历结束未标记的即为不可达的代码

```
1  int controlFlowUnreachable() {
2      int x = 1;
3      return x;
4      int z = 42; // control-flow unreachable code
5      foo(z); // control-flow unreachable code
6  }
```

实验内容

死代码检测

- 不可达代码：
  - 分支不可达代码：Java中if语句和switch语句皆会导致此类不可达代码的出现，当这两类语句的条件值是常数时，那不符合条件的分支就可能不可达。检测此类不可达代码时，需要预先进行常量传播分析，由此判断条件值是否为常量，再通过遍历CFG的方式标记代码

```java
int unreachableIfBranch() {
    int a = 1, b = 0, c;
    if (a > b)
        c = 2333;
    else
        c = 6666; // unreachable branch
    return c;
}
```

```java
int unreachableSwitchBranch() {
    int x = 2, y;
    switch (x) {
        case 1: y = 100; break; // unreachable branch
        case 2: y = 200;
        case 3: y = 300; break; // fall through
        default: y = 666; // unreachable branch
    }
    return y;
}
```

实验内容

死代码检测

- 无用赋值：一个局部变量在一条语句中被赋值后却再没有被后面的语句读取即为无用赋值。检测无用赋值需要先进行活跃变量分析，对一个赋值语句如果复制号左侧变量是无用变量则这个语句标记为无用赋值

```
1  int deadAssign() {
2      int a, b, c;
3      a = 0; // dead assignment
4      a = 1;
5      b = a * 2; // dead assignment
6      c = 3;
7      return c;
8  }
```

实验内容

具体任务

- 补全LiveVariableAnalysis.java 和 ConstantPropagation.java，可以直接copy前面实验内容
- 完成一个同时支持前向分析和后向分析的 worklist 求解器，这部分也可以直接将前两个实验直接copy
- 实现DeadCodeDetection中的一个API：Set<Stmt> analyze(IR)，这个方法将IR作为输入，经过一系列分析后返回IR中死代码的集合

API介绍

pascal.taie.analysis.dataflow.analysis.DeadCodeDetection

```java
public Set<Stmt> analyze(IR ir) {
    // obtain CFG
    CFG<Stmt> cfg = ir.getResult(CFGBuilder.ID);
    // obtain result of constant propagation
    DataflowResult<Stmt, CPFact> constants =
            ir.getResult(ConstantPropagation.ID);
    // obtain result of live variable analysis
    DataflowResult<Stmt, SetFact<Var>> liveVars =
            ir.getResult(LiveVariableAnalysis.ID);
    // keep statements (dead code) sorted in the resulting set
    Set<Stmt> deadCode = new TreeSet<>(Comparator.comparing(Stmt::getIndex));
    // TODO - finish me
    // Your task is to recognize dead code in ir and add it to deadCode
    return deadCode;
}
```

- 这个类是用于死代码检测，也就是本次实验需要补全的类

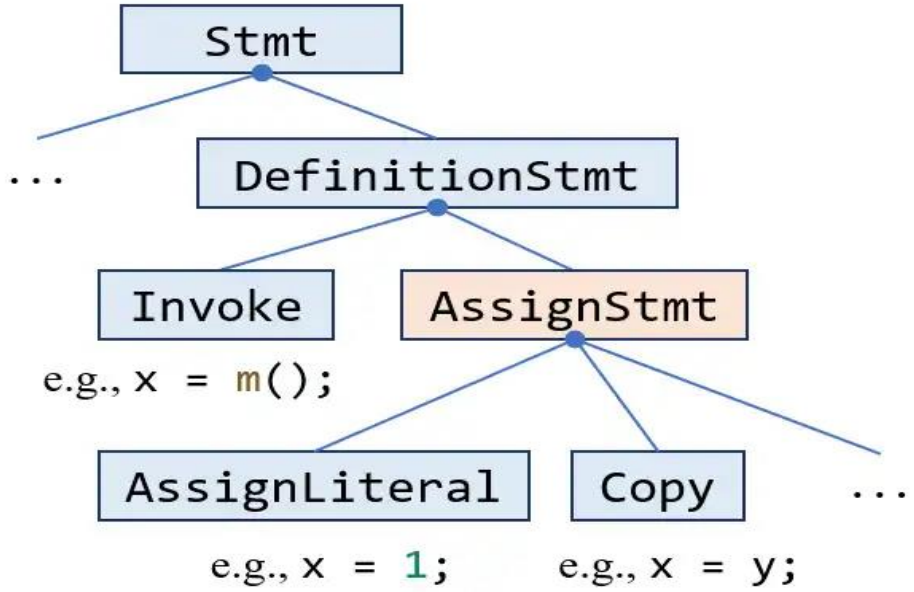
```java
     * @return true if given RValue has no side effect, otherwise false.
     */
    private static boolean hasNoSideEffect(RValue rvalue) {...}
```

API介绍

pascal.taie.ir.stmt

```
/**
 * @return true if execution after this statement could continue at
 * the following statement, otherwise false.
 */
boolean canFallThrough();


/**
 * @return the index of this Stmt in the container IR.
 */
@Override
int getIndex();
```



API介绍

pascal.taie.ir.stmt.AssignStmt

```java
/**
 * Representation of assign statements.
 *
 * @param <L> type of lvalue.
 * @param <R> type of rvalue.
 */
public abstract class AssignStmt<L extends LValue, R extends RValue>
        extends DefinitionStmt<L, R> {

    private final L lvalue;

    private final R rvalue;

    @Override
    public @Nonnull L getLValue() { return lvalue; }

    @Override
    public R getRValue() { return rvalue; }
```

- 这个类表示程序中的赋值语句（比如x = ...;）

API介绍

pascal.taie.ir.stmt.If

```
/**
 * @return the condition expression of the if-statement.
 */
public ConditionExp getCondition() { return condition; }

/**
 * @return the jump target (when the condition expression is evaluated
 * to true) of the if-statement.
 */
public Stmt getTarget() { return target; }
```

- 这个类表示程序中的 if 语句

API介绍

- **pascal.taie.ir.stmt.If**

```java
while (a > b) {
    x = 233;
}
y = 666;
```

```java
0:  if (a > b) goto 2;
1:  goto 4;
2:  x = 233;
3:  goto 0;
4:  y = 666;
```

API介绍

pascal.taie.ir.stmt.SwitchStmt

```
/**
 * @return the variable holding the condition value of the switch-statement.
 */
public Var getVar() { return var; }

/**
 * @return the i-th jump target (for i-th case) of the switch-statement.
 * The indexes start from 0. Target for default case is excluded.
 */
public Stmt getTarget(int i) { return targets.get(i); }
```

public Stmt getDefaultTarget() {return defaultTarget;}

public abstract List<Pair<Integer, Stmt>> getCaseTargets();

public abstract List<Integer> getCaseValues();

- 表示程序中的 switch 语句。大家需要阅读它的源代码和注释来决定如何使用它。

API介绍

pascal.taie.analysis.graph.cfg.Edge

需要考虑四种类型的边：

IF_TRUE、IF_FALSE、

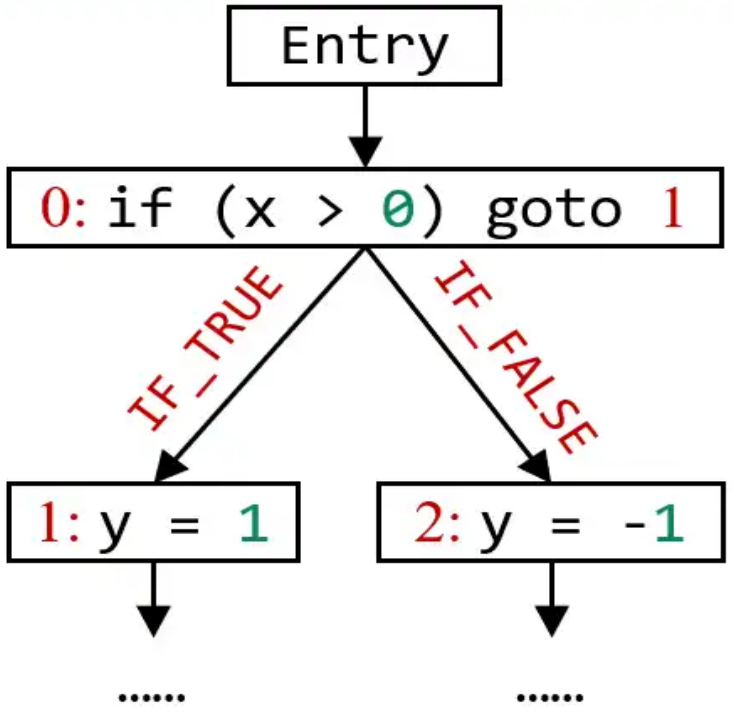
SWITCH_CASE、

SWITCH_DEFAULT

```java
/**
 * @return the kind of the edge.
 * @see Edge.Kind
 */
public Kind getKind() { return kind; }
```

```java
void ifBranch(int x) {
    int y;
    if (x > 0) {
        y = 1;
        …
    } else {
        y = -1;
        …
```

Code

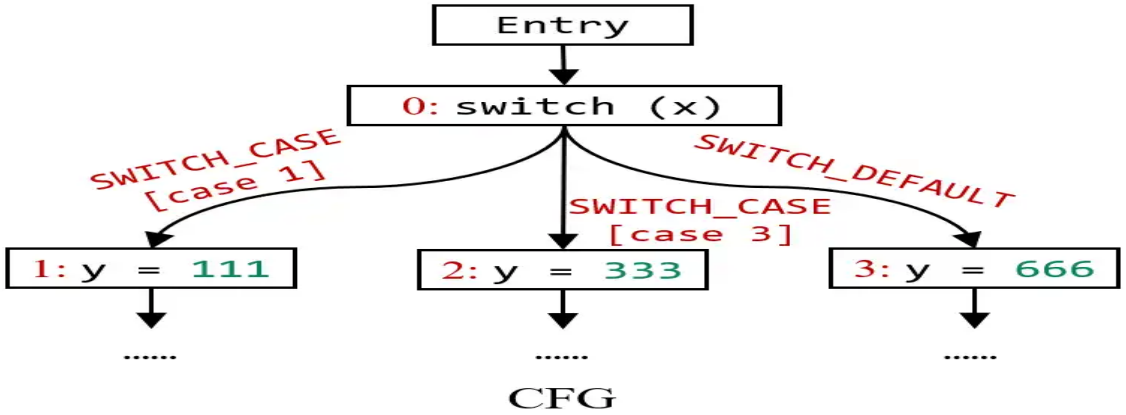


CFG

API介绍

- pascal.taie.analysis.graph.cfg.Edge

```
void switchBranch(int x) {
    int y;
    switch (x) {
        case 1: y = 111; …
        case 3: y = 333; …
        default: y = 666; …
    }
}
```



CFG

```
/**
 * If this edge is a switch-case edge, then returns the case value.
 * The client code should call {@link #isSwitchCase()} to check if
 * this edge is switch-case edge before calling this method.
 *
 * @throws AnalysisException if this edge is not a switch-case edge.
 */
public int getCaseValue() {
    // SwitchCaseEdge overrides this method, thus this method
    // should NOT be reachable
    throw new AnalysisException(this + " is not a switch-case edge," +
            " please call isSwitchCase() before calling this method");
}
```

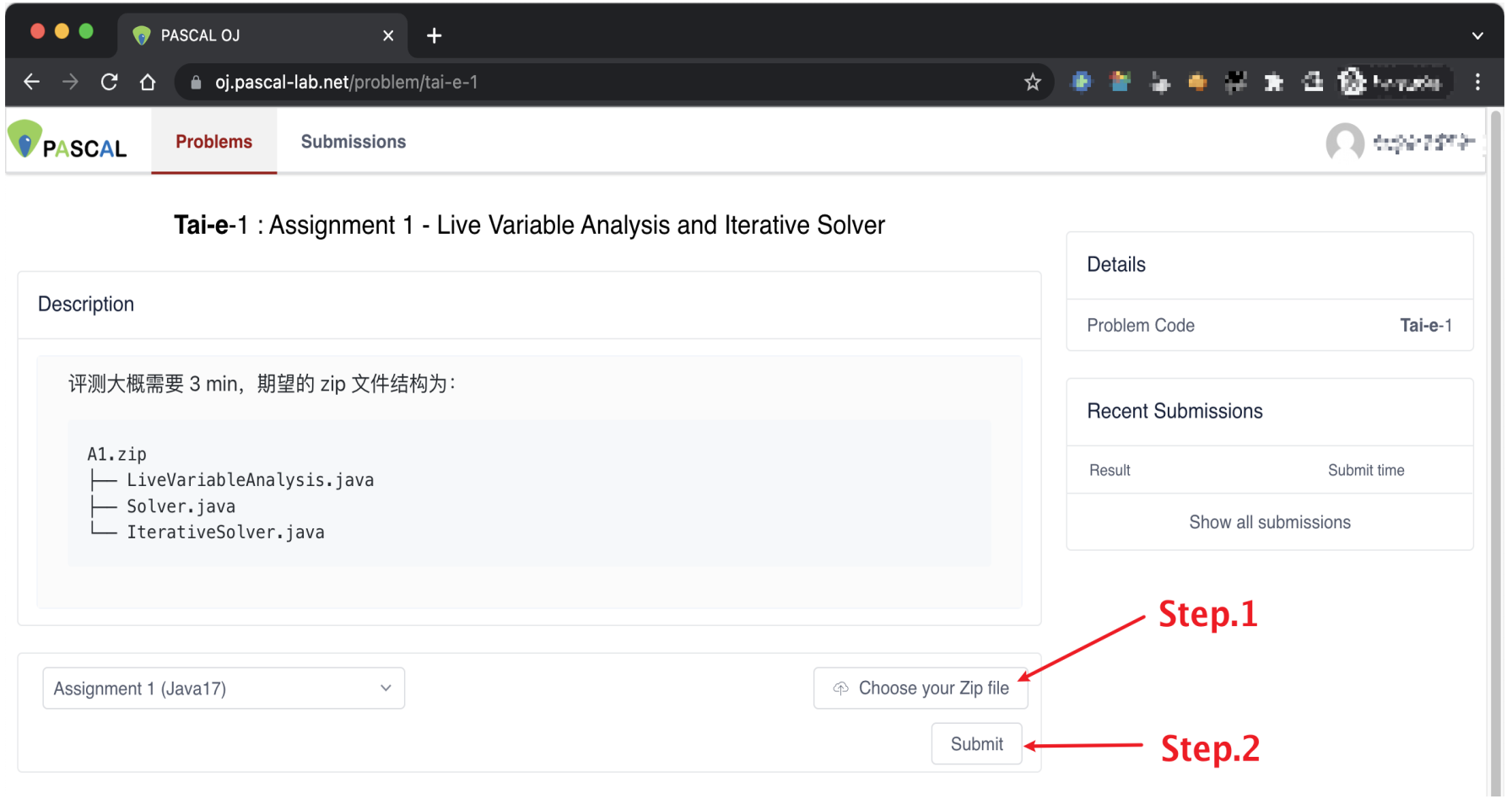
```
/**
 * @return true if this edge is a switch-case edge, otherwise false.
 */
public boolean isSwitchCase() { return kind == Kind.SWITCH_CASE; }
```

作业提交

在线测试平台：https://oj.pascal-lab.net/problem

提交一个**zip文件**，包括实现好的一个类：

- DeadCodeDetection.java



作业测试与提交

最终提交内容

- 实验报告
- 代码（zip文件）

将实验报告与A2.zip文件放在同一个文件夹下，再将其压缩为一个文件后后上传

截止时间：2024-4-30 23:59

提交地址：

https://send2me.cn/gOCemcBP/SBqqNHKC9fNAzw