多个程序依赖图的融合问题

一、问题简介

1. 程序融合问题

程序融合是指将一组程序当中语义相似的语句合并在一起。其中，语义相似的语句指：在不同程序中，对程序的功能实现起同样作用的语句。

程序融合的一个应用场景是在协同式编程中实现错误程序代码的自动化修复：一组编程人员（初学者）各自独立的实现同一个编程任务，并提交代码。这些代码可能存在这样或那样的错误（假设没有语法错误），导致程序无法得到理想的结果。

为自动发现和纠正这些错误，可以借助程序融合-反馈的方法。

（1） 程序融合：由于这些代码求解同一问题，它们必定会存在一定的相似性。借助程序融合，可以发现程序中与问题求解起相同作用的语句（语义相似的语句），然后将它们合并在一起。

（2） 错误反馈：在融合结果中寻找差异，差异表现在语句表达式上的，也包含执行的关系上。通过识别差异，发现程序融合语句中少数人的表达方式和多数人的表达方式，把多数人的表达反馈给少数人中，进而修正少数程序语句中的错误。

2. 案例

有三个学生分别提交了三个代码求解 Fibonacci 问题：输入一个整数，输出该整数位置的 Fibonacci 的值。如图 1 所示：这三个代码中分别存在不同情况的错误，使用基于程序融合-反馈的方法，修正这些错误。

程序融合过程：

- 先将这三个代码转换成程序依赖图（PDG）
- 程序依赖图融合生成程序依赖融合图（CPDG）
- 检查 CPDG 并反馈程序修复信息

二、图的表示

1. 程序依赖图的含义

程序依赖图常用于表达程序语义。一个程序依赖图用节点表达语句，用边表达语句之间的依赖关系。

节点分为 6 种类型：

- Entry：程序的入口
- 赋值（assignment）：赋值语句，例如：i=i+1;
- 声明(declaration)：对某个变量的声明语句，例如：int i;
- 谓词(predicate)：控制程序执行流程的语句，一般是条件表达式，例如：i>10
- 控制转换(control transfer)：改变程序控制流程的语句，如 break, continue, return
- 限定性函数(restricted function)：为了简化问题，仅处理输入/输出函数 scanf, printf)。

边有四种类型：

- 控制依赖(Control dependence，C)：对于两个节点 v1 和 v2 之间存在控制依赖，当 v1 是谓词节点（或 entry）节点，只有当 v1 成立（条件为真），v2 执行。例如：v1: while(i<1){v2:i=i+1;}
- 声明依赖(Declaration dependence，DE)：对于两个节点 v1 和 v2 之间存在声明依赖，当 v1 声明一个变量，而 v2 对这个变量进行赋值。例如：v1:int i; v2:i=1;
- 数据依赖(Data dependence，D)：对于两个节点 v1 和 v2 之间存在数据依赖，如果 v1 对某个变量进行赋值，v2 使用了这个变量。例如：v1:i=1;v2:i=i+1;
- 循环数据依赖(Loop Data dependence，LD)：对于两个节点 v1 和 v2 之间存在循环数

据依赖，v1 和 v2 在一个循环体内，并且 v1 对某个变量的赋值会影响到下一次循环执行时 v2 的计算。例如：while(i<10){v1:sum=sum+i; v2:i=i+1;}

例如：求解 Fibonacci 的一个代码和它的 PDG 如下图 2 所示：

```
1  int main(){
2      int f1,f2,f3;
3      int i,n;
4      f1=1;
5      f2=1;
6      f3=1;
7      i=2;
8      scanf("%d",&n);
9      while(i<=n)
10     {
11         f3=f1+f2;
12         f1=f2;
13         f2=f3;
14         i=i+1;
15     }
16     printf("%d",f3);
17     return 0;
18 }
```

P_0

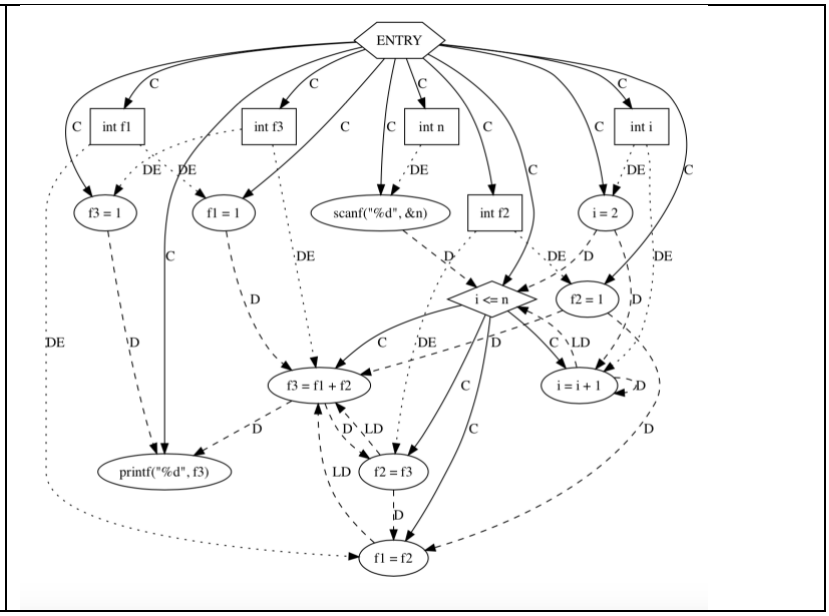


图 2 程序代码和程序依赖图

3．程序融合图的含义

程序融合图包括 融合节点 和 融合边。**融合节点** 是一个包含一组语义相似的，且来自于不同程序依赖图的节点的集合。其中语义相似是指：类型相同，语句表达式相似，且在 PDG 中的依赖关系也相似。 **融合边**也是一个集合，对于一个从融合节点 V1 到融合节点 V2 的融合边 E,它包含一个或多条来自于 PDG 的边，这些边具有同一类型，起点属于 V1,终点属于 V2. 如图 1 中 CPDG 图下方的表列出了所有融合边的细节。

三、输入文件格式说明

程序融合的输入数据是一组程序依赖图 PDG，输出结果是一个程序依赖融合图 CPDG。

PDG 和 CPDG 都使用.dot 的格式进行存储，它是一种开源图形化表达方式。详见 http://www.graphviz.org。

例如，图 2 中的 PDG 图的.dot 存储格式如下：

```
digraph PDG_Graph{
    Node2 [label="ENTRY",shape="hexagon", type="CONTROL",coord="1"];
    Node4 [label="int f1",shape="box",type="DECL",coord="2",varname="f1",vartype="int"];
    Node5 [label="int f2",shape="box",type="DECL",coord="2",varname="f2",vartype="int"];
    Node6 [label="int f3",type="DECL",shape="box",coord="2",varname="f3",vartype="int"];
    Node7 [label="int i",shape="box",type="DECL",coord="3",varname="i",vartype="int"];
    Node8 [label="int n",shape="box",type="DECL",coord="3",varname="n",vartype="int"];
    Node9 [label="f1 = 1",type="ASSIGN",coord="4"];
    Node13 [label="f2 = 1",type="ASSIGN",coord="5"];
    Node17 [label="f3 = 1",type="ASSIGN",coord="6"];
    Node21 [label="i = 2",type="ASSIGN",coord="7"];
    Node25 [label="scanf(\"%d\", &n)",type="CALL",coord="8"];
    Node31 [label="i <= n",shape="diamond",type="CONTROL",coord="9"];
    Node33 [label="f3 = f1 + f2",type="ASSIGN",coord="11"];
    Node39 [label="f1 = f2",type="ASSIGN",coord="12"];
    Node43 [label="f2 = f3",type="ASSIGN",coord="13"];
    Node47 [label="i = i + 1",type="ASSIGN",coord="14"];
    Node55 [label="printf(\"%d\", f3)",type="CALL",coord="16"];
    Node2->Node4 [label = "C" ];
    Node2->Node5 [label = "C" ];
    Node2->Node6 [label = "C" ];
```

```
Node2->Node7 [label = "C" ];
Node2->Node8 [label = "C" ];
Node2->Node9 [label = "C" ];
Node2->Node13 [label = "C" ];
Node2->Node17 [label = "C" ];
Node2->Node21 [label = "C" ];
Node2->Node25 [label = "C" ];
Node2->Node31 [label = "C" ];
Node2->Node55 [label = "C" ];
Node31->Node33 [label = "C" ];
Node31->Node39 [label = "C" ];
Node31->Node43 [label = "C" ];
Node31->Node47 [label = "C" ];
Node9->Node33 [label = "D",style=dashed ]
Node13->Node33 [label = "D",style=dashed]
Node13->Node39 [label = "D",style=dashed]
Node17->Node55 [label = "D",style=dashed]
Node21->Node31 [label = "D",style=dashed]
Node21->Node47 [label = "D",style=dashed ]
Node25->Node31 [label = "D" ,style=dashed]
Node33->Node43 [label = "D",style=dashed]
Node33->Node55 [label = "D",style=dashed]
Node39->Node33 [label = "LD",style=dashed]
Node43->Node33 [label = "LD",style=dashed]
Node43->Node39 [label = "D",style=dashed]
Node47->Node31 [label = "LD",style=dashed]
Node47->Node47 [label = "D",style=dashed]
Node4->Node9 [label = "DE",style=dotted]
Node4->Node39 [label = "DE",style=dotted]
Node5->Node43 [label = "DE",style=dotted]
Node5->Node13 [label = "DE",style=dotted]
Node6->Node33 [label = "DE",style=dotted]
Node6->Node17 [label = "DE",style=dotted]
Node7->Node21 [label = "DE",style=dotted]
Node7->Node47 [label = "DE",style=dotted]
Node8->Node25 [label = "DE",style=dotted]
}
```

四、评价

程序依赖图的效果从以下两个方面体现：

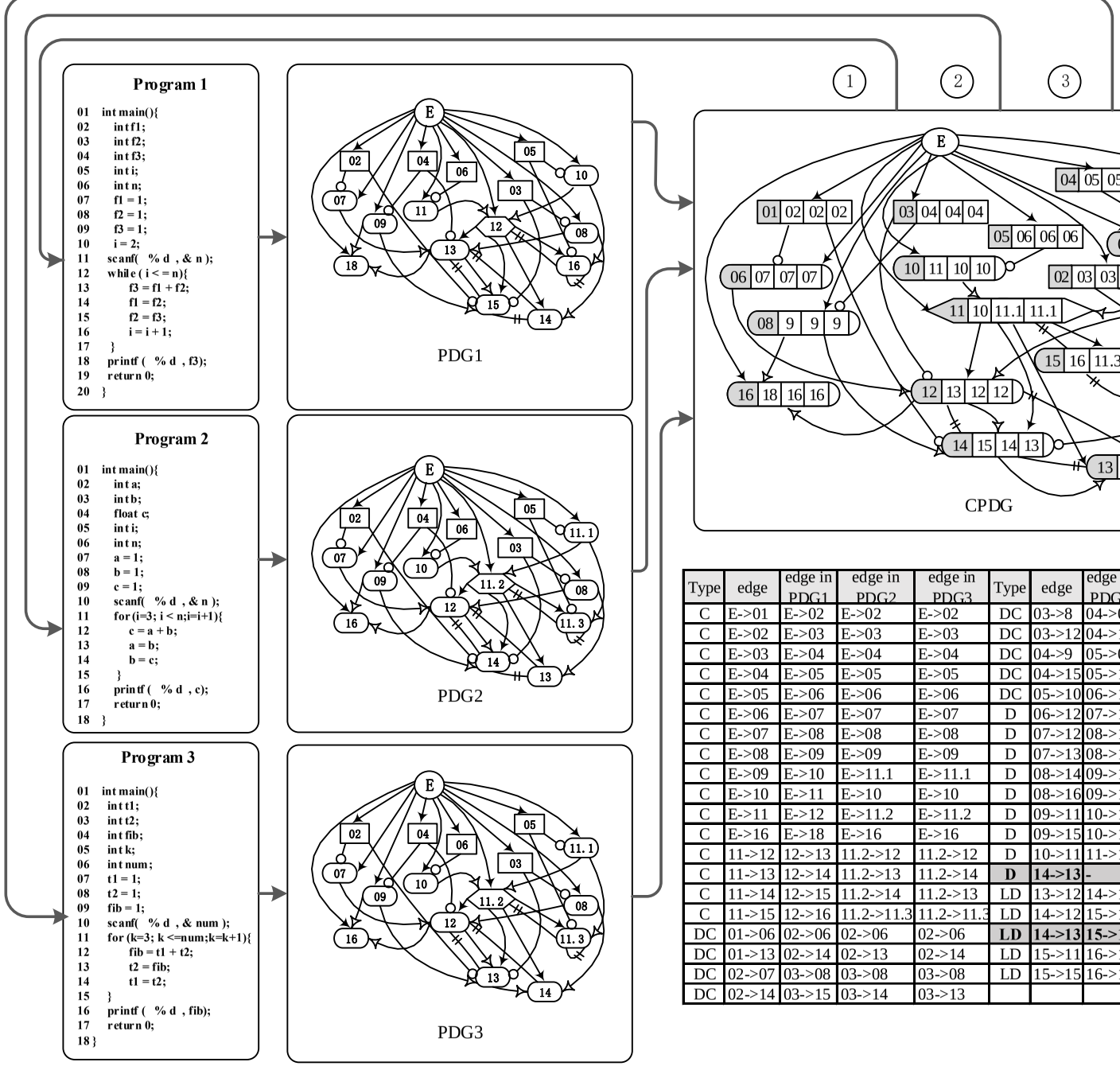
- 融合图中的节点和边的数量都尽可能的少；
- 每个融合节点中来自于不同 PDG 的语句的语义相似性近可能的高。语义相似性不仅要考虑语句表达式上的相似，还应该尽可能的考虑语句与其他语句之间的关系。

五、测试案例说明

一共包含三个用于测试程序依赖图融合方法案例:

1. 其中一个是图 1 中的 Fibonacci 案例，另外两个案例每个包括 10 个 PDG，问题描述如下：

- 统计药费（drug price）：依据给定分段计费规则计算总费用。
- 出现奇数次的数（odd number）：输入一个数组，寻找其中唯一一个出现奇数次的数。



Feedback Information

①	Replace i=2 at line 10 with i=3
②	Replace i<n at line 11 with i<=n
③	Adjust the order of t2=fib and t1=t2 in line 13 and 14

Type	edge	edge in PDG1	edge in PDG2	edge in PDG3	Type	edge	edge in PDG1	edge in PDG2	edge in PDG3
C	E->01	E->02	E->02	E->02	DC	03->8	04->09	04->09	04->09
C	E->02	E->03	E->03	E->03	DC	03->12	04->13	04->12	04->12
C	E->03	E->04	E->04	E->04	DC	04->9	05->09	05->09	05->09
C	E->04	E->05	E->05	E->05	DC	04->15	05->16	05->11.3	05->11.3
C	E->05	E->06	E->06	E->06	DC	05->10	06->11	06->10	05->10
C	E->06	E->07	E->07	E->07	D	06->12	07->13	07->12	07->12
C	E->07	E->08	E->08	E->08	D	07->12	08->13	08->12	08->12
C	E->08	E->09	E->09	E->09	D	07->13	08->14	08->13	08->14
C	E->09	E->10	E->11.1	E->11.1	D	08->14	09->15	09->14	09->13
C	E->10	E->11	E->10	E->10	D	08->16	09->18	09->16	09->16
C	E->11	E->12	E->11.2	E->11.2	D	09->11	10->12	11.1->11.2	11.1->11.2
C	E->16	E->18	E->16	E->16	D	09->15	10->16	11.1->11.3	11.1->11.3
C	11->12	12->13	11.2->12	11.2->12	D	10->11	11->12	10->11.2	10->11.2
C	11->13	12->14	11.2->13	11.2->14	**D**	**14->13**	**-**	**-**	**13->14**
C	11->14	12->15	11.2->14	11.2->13	LD	13->12	14->13	13->>12	14->12
C	11->15	12->16	11.2->11.3	11.2->11.3	LD	14->12	15->13	14->12	13->12
DC	01->06	02->06	02->06	02->06	**LD**	**14->13**	**15->14**	**14->13**	**-**
DC	01->13	02->14	02->13	02->14	LD	15->11	16->12	11.3->11.2	11.3->11.2
DC	02->07	03->08	03->08	03->08	LD	15->15	16->16	11.3->11.3	11.3->11.3
DC	02->14	03->15	03->14	03->13					

ID	P1	P2	P3
01	int f1	int a	int t1
02	int f2	int b	int t2
03	int f3	float c	int fib
04	int i	int i	int k
05	int n	int n	int num
06	f1=1	a=1	b=1
07	f2=1	b=1	t2=1
08	f3=1	c=1	fib=1
09	**i=2**	**i=3**	**k=3**
10	scanf…	scanf…	scanf…
11	**i<=n**	**i<n**	**k<=num**
12	f3=f1+f2	c=a+b	fib=t1+t2
13	f1=f2	a=b	t1=t2
14	f2=f3	b=c	t2=fib
15	i=i+1	i=i+1	k=k+1
16	printf…	printf…	printf…

图 1 程序融合示例