程序执行序列的获取方法及装置

申请号：201410001275.7
申请日：2014-01-02

申请(专利权)人 北京百度网讯科技有限公司 南京大学

地址 100085 北京市海淀区上地十街10号百度大厦2层

发明(设计)人 陈振宇 田亮 张旭辉 黄明明 曾卫 汪亚斌 周锴 高则宝 沈毅 房春荣

主分类号 G06F11/36(2006.01)I

分类号 G06F11/36(2006.01)I

公开(公告)号 103744782A

公开(公告)日 2014-04-23

专利代理机构 北京清亦华知识产权代理事务所(普通合伙) 11201

代理人 宋合成

(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103744782 A
(43)申请公布日 2014.04.23

(21)申请号 201410001275.7

(22)申请日 2014.01.02

(71)申请人 北京百度网讯科技有限公司
地址 100085 北京市海淀区上地十街 10 号百度大厦 2 层
申请人 南京大学

(72)发明人 陈振宇 田亮 张旭辉 黄明明 曾卫 汪亚斌 周锴 高则宝 沈毅 房春荣

(74)专利代理机构 北京清亦华知识产权代理事务所（普通合伙）11201
代理人 宋合成

(51)Int.Cl.
G06F 11/36(2006.01)

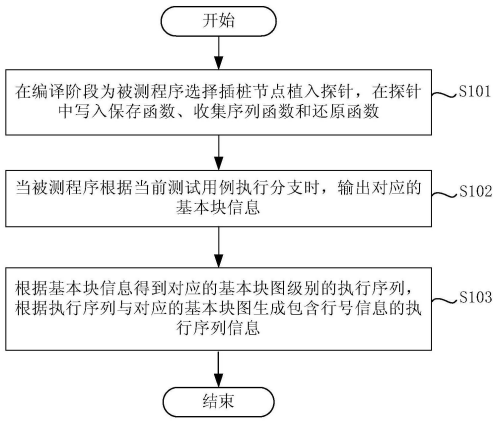
权利要求书1页 说明书7页 附图3页

(54)发明名称

程序执行序列的获取方法及装置

(57)摘要

本发明提出一种程序执行序列的获取方法及装置，其中，程序执行序列的获取方法包括：在编译阶段为被测程序选择插桩节点植入探针，在探针中写入保存函数、收集序列函数和还原函数；当被测程序根据当前测试用例执行分支时，输出对应的基本块信息；以及根据所述基本块信息得到基本块图级别的执行序列，根据执行序列与对应的基本块图生成包含行号信息的执行序列信息。本发明实施例，通过插桩模块在探针中写入保存函数和还原函数，可以在不修改程序源代码的情况下，实现目标代码的插桩，通过执行测试用例后可以获得程序的多个执行序列，进而可以帮助测试人员实现软件工程任务，比如错误定位等。



CN 103744782 A

1. 一种程序执行序列的获取方法，其特征在于，包括：

在编译阶段为被测程序选择插桩节点植入探针，在所述探针中写入保存函数、收集序列函数和还原函数；

当所述被测程序根据当前测试用例执行分支时，输出对应的基本块信息；以及

根据所述基本块信息得到基本块图级别的执行序列，根据所述执行序列与对应的基本块图生成包含行号信息的执行序列信息。

2. 如权利要求 1 所述的方法，其特征在于，所述保存函数用于在调用所述收集序列函数之前，将寄存器数据保存到内存中；所述还原函数用于在调用所述收集序列函数之后，将保存在内存中的寄存器数据进行还原。

3. 如权利要求 1 所述的方法，其特征在于，在所述在编译阶段为被测程序选择插桩节点植入探针之前，所述方法还包括：

预处理所述被测程序的源文件，对所述源文件划分基本块和分支，并绘制基本块图。

4. 如权利要求 3 所述的方法，其特征在于，所述为被测程序选择插桩节点植入探针包括：

根据所述基本块图确定需要插桩的分支，根据所述需要插桩的分支确定所述插桩节点。

5. 如权利要求 1-4 任一权利要求所述的方法，其特征在于，在所述根据所述执行序列与对应的基本块图生成包含行号信息的执行序列信息之后，所述方法还包括：

获得多个执行序列信息，根据所述执行序列信息进行所述被测程序的错误定位。

6. 一种程序执行序列的获取装置，其特征在于，包括：

插桩模块，用于在编译阶段为被测程序选择插桩节点植入探针，在所述探针中写入保存函数、收集序列函数和还原函数；

执行模块，用于当所述被测程序根据当前测试用例执行分支时，输出对应的基本块信息；以及

生成模块，用于根据所述基本块信息得到基本块图级别的执行序列，根据所述执行序列与对应的基本块图生成包含行号信息的执行序列信息。

7. 如权利要求 6 所述的装置，其特征在于，所述保存函数用于在调用所述收集序列函数之前，将寄存器数据保存到内存中；所述还原函数用于在调用所述收集序列函数之后，将保存在内存中的寄存器数据进行还原。

8. 如权利要求 6 所述的装置，其特征在于，所述装置还包括：

绘制模块，用于在所述插桩模块为被测程序选择插桩节点植入探针之前，预处理所述被测程序的源文件，对所述源文件划分基本块和分支，并绘制基本块图。

9. 如权利要求 8 所述的装置，其特征在于，所述插桩模块，具体用于：

根据所述绘制模块绘制的基本块图确定需要插桩的分支，根据所述需要插桩的分支确定所述插桩节点。

10. 如权利要求 6-9 任一权利要求所述的装置，其特征在于，所述装置还包括：

定位模块，用于在所述生成模块生成包含行号信息的执行序列信息之后，获得多个执行序列信息，根据所述执行序列信息进行所述被测程序的错误定位。

程序执行序列的获取方法及装置

技术领域

[0001] 本发明涉及计算机技术领域，尤其涉及一种程序执行序列的获取方法及装置。

背景技术

[0002] 程序测试是指对一个完成了全部或部分功能、模块的计算机程序在正式使用前的检测，以确保该程序能按预定的方式正确地运行。目前采用的测试方法包括白盒测试和黑盒测试等。

[0003] 其中，白盒测试也被称作结构测试或者逻辑驱动测试，它主要测试程序内部的结构。测试人员根据程序的内部逻辑结构信息，设计测试用例并测试程序的逻辑路径，验证执行结果。与白盒测试相对的是黑盒测试，黑盒测试又称为功能测试。黑盒测试不考虑软件内部结构和内部特性，而只关注接口输入输出的正确性。

[0004] 覆盖技术是白盒测试中常用的技术之一，逻辑覆盖是一种标准，表示测试用例在程序执行时对程序逻辑的覆盖程度。代码覆盖主要关注程序运行时内部逻辑的覆盖程度，按照粒度的不同可以分为：语句覆盖、判定覆盖、条件覆盖、判定 / 条件覆盖、条件组合覆盖等。覆盖测试可以通过检测是否对程序的关键路径都已经覆盖完全来检查测试用例设计的合理性。程序插桩是覆盖测试所依赖的关键技术。插桩技术在不影响程序逻辑特性的前提下，使得运行时覆盖信息的收集成为可能。按照插桩时机的不同，程序插桩技术主要分为两种：(1)源代码插桩：源代码插桩就是直接对程序源文件进行修改，插入相应的桩代码。源代码插桩需要对源代码进行完整的词法分析和语法分析，从而确定插桩的位置。(2)目标代码插桩：相比源代码插桩，目标代码插桩不会影响源代码，但是会影响生成的目标代码。但是，目标代码中通常不具有相应的程序结构。基于程序插桩技术，就可以在覆盖测试中获得各个粒度级别的覆盖信息。

[0005] 在科研工作中通常使用源代码插桩的方式获取执行序列，这样能够获得精确的结果并且易于操作，但在工业应用中，通常不允许修改程序的源代码，因此，在工业应用中不能使用源代码插桩的方式来获取执行序列。

发明内容

[0006] 本发明旨在至少解决上述技术问题之一。

[0007] 为此，本发明的第一个目的在于提出一种程序执行序列的获取方法。该方法，在不修改程序源代码的情况下，实现了目标代码的插桩，执行测试用例后可以获得程序的执行序列，进而可以帮助测试人员实现软件工程任务，比如错误定位。

[0008] 为了实现上述目的，本发明第一方面实施例的程序执行序列的获取方法，包括以下步骤：

[0009] 在编译阶段为被测程序选择插桩节点植入探针，在探针中写入保存函数、收集序列函数和还原函数；

[0010] 当被测程序根据当前测试用例执行分支时，输出对应的基本块信息；以及

[0011] 根据基本块信息得到基本块图级别的执行序列，根据执行序列与对应的基本块图生成包含行号信息的执行序列信息。

[0012] 本发明实施例的程序执行序列的获取方法，通过在探针中写入保存函数和还原函数，可以在不修改程序源代码的情况下，实现目标代码的插桩，通过执行测试用例后可以获得程序的多个执行序列，进而可以帮助测试人员实现软件工程任务，比如错误定位。

[0013] 为了实现上述目的，本发明第二方面实施例的程序执行序列的获取装置，包括：插桩模块、执行模块以及生成模块。

[0014] 本发明实施例的程序执行序列的获取装置，通过插桩模块在探针中写入保存函数和还原函数，可以在不修改程序源代码的情况下，实现目标代码的插桩，通过执行测试用例后可以获得程序的多个执行序列，进而可以帮助测试人员实现软件工程任务，比如错误定位等。

[0015] 本发明附加的方面和优点将在下面的描述中部分给出，部分将从下面的描述中变得明显，或通过本发明的实践了解到。

附图说明

[0016] 本发明上述的和 / 或附加的方面和优点从下面结合附图对实施例的描述中将变得明显和容易理解，其中，

[0017] 图 1 是根据本发明一个实施例的程序执行序列的获取方法的流程图；

[0018] 图 2 是根据本发明另一个实施例的程序执行序列的获取方法的流程图；

[0019] 图 3 是根据本发明一个实施例的基本块图的示意图；

[0020] 图 4 是根据本发明一个实施例的程序执行序列的获取装置的结构示意图；

[0021] 图 5 是根据本发明另一个实施例的程序执行序列的获取装置的结构示意图。

具体实施方式

[0022] 下面详细描述本发明的实施例，实施例的示例在附图中示出，其中自始至终相同或类似的标号表示相同或类似的元件或具有相同或类似功能的元件。下面通过参考附图描述的实施例是示例性的，仅用于解释本发明，而不能理解为对本发明的限制。相反，本发明的实施例包括落入所附加权利要求书的精神和内涵范围内的所有变化、修改和等同物。

[0023] 在本发明的描述中，术语“第一”、“第二”等仅用于描述目的，而不能理解为指示或暗示相对重要性。在本发明的描述中，除非另有明确的规定和限定，术语“相连”、“连接”应做广义理解，例如，可以是固定连接，也可以是可拆卸连接，或一体地连接；可以是机械连接，也可以是电连接；可以是直接相连，也可以通过中间媒介间接相连。对于本领域的普通技术人员而言，可以具体情况理解上述术语在本发明中的具体含义。此外，在本发明的描述中，除非另有说明，“多个”的含义是两个或两个以上。

[0024] 流程图中或在此以其他方式描述的任何过程或方法描述可以被理解为，表示包括一个或更多个用于实现特定逻辑功能或过程的步骤的可执行指令的代码的模块、片段或部分，并且本发明的优选实施方式的范围包括另外的实现，其中可以不按所示出或讨论的顺序，包括根据所涉及的功能按基本同时的方式或按相反的顺序，来执行功能，这应被本发明的实施例所属技术领域的技术人员所理解。

[0025] 下面参考附图描述本发明实施例的程序执行序列的获取方法及装置。

[0026] 图 1 是根据本发明一个实施例的程序执行序列的获取方法的流程图。

[0027] 如图 1 所示，程序执行序列的获取方法包括以下步骤：

[0028] S101，在编译阶段为被测程序选择插桩节点植入探针，在探针中写入保存函数、收集序列函数和还原函数。

[0029] 其中，保存函数用于在调用收集序列函数之前，将寄存器数据保存到内存中；还原函数用于在调用收集序列函数之后，将保存在内存中的寄存器数据进行还原。

[0030] 在探针中写入保存函数的目的是保存收集序列函数的上下文场景；在探针中写入还原函数的目的是恢复收集序列函数上下文场景；在探针中写入收集序列函数的目的是收集程序执行序列。

[0031] 另外，在为被测程序选择插桩节点植入探针之前，该方法还可以包括：预处理被测程序的源文件，对源文件划分基本块和分支，并绘制基本块图。

[0032] 绘制完基本块图之后，根据基本块图确定需要插桩的分支，然后根据需要插桩的分支确定插桩节点。

[0033] S102，当被测程序根据当前测试用例执行分支时，输出对应的基本块信息。

[0034] 测试用例是由测试数据和预期结果构成的，在本发明实施例中可以采用多个测试用例，这些测试用例包括执行成功的测试用例和执行失败的测试用例。执行不同测试用例可以获得不同的分支信息。

[0035] 当被测程序根据当前测试用例执行分支时，会执行分支中的基本块。

[0036] S103，根据基本块信息得到基本块图级别的执行序列，根据执行序列与对应的基本块图生成包含行号信息的执行序列信息。

[0037] 在本实施例中，在执行分支中的基本块时会触发收集序列函数收集基本块的执行序列；然后可以通过面向对象、直译式计算机程序设计语言处理包含执行序列的执行文件以及存放程序的覆盖信息和基本块图信息的文件，得到易读的执行序列信息即包含行号信息的执行序列信息。

[0038] 上述程序执行序列的获取方法，在不修改程序源代码的情况下，实现了目标代码的插桩，执行测试用例后可以获得程序的执行序列，进而可以帮助测试人员实现软件工程任务，比如错误定位。

[0039] 图 2 是根据本发明另一个实施例的程序执行序列的获取方法的流程图。

[0040] 如图 2 所示，程序执行序列的获取方法包括以下步骤：

[0041] S200，预处理被测程序的源文件，对源文件划分基本块和分支，并绘制基本块图。

[0042] 假定，本实施例中的待测程序的源文件如下：

[0043]

```
1:    int main()
2:    {
3:        int i, j, k;
4:        j = 0;
5:        for(i = 0; i != 3; ++i)
6:        {
7:            ++j;
8:        }
9:
10:       k = 0;
11:       if(j)
12:       {
13:           ++k;
14:       }
15:       else
16:       {
17:           --k;
18:       }
19:   }
```

[0044]

[0045] 通过代码覆盖工具绘制得到的基本块(BB)图如图 3 所示,从图 3 可以看出该程序被划分成为 9 个基本块,即 BB0-BB8。其中 BB0、BB2、BB7、BB8 没有对应的源代码信息,这些是编译工具插入的辅助基本块,相应的分支(arc)也是虚拟的分支。根据有向图的基本概念,每一个顶点的出度和等于其入度和。其中,BB0 是虚拟的开始入口节点,BB8 是虚拟的出口节点,BB0 的入度和为 0,且 BB8 的出度和为 0。

[0046] S201,在编译阶段为被测程序选择插桩节点植入探针,在探针中写入保存函数、收集序列函数和还原函数。

[0047] 在本实施例中,通过步骤 S200 绘制完如图 3 所示的基本块图之后,可以根据基本块图确定需要插桩的分支,然后根据需要插桩的分支确定插桩节点。

[0048] 在程序编译阶段在所确定的插桩节点植入探针,在该实施例中要为 arc0-arc9 的所有边都插桩。在探针中写入保存函数的目的是保存收集序列函数的上下文场景 ;在探针中写入还原函数的目的是恢复收集序列函数上下文场景。

[0049] 另外,在本实施例中,保存函数和还原函数采用内联汇编(即在高级语言例如 C 语言中插入汇编语言),可以直接访问寄存器,提高效率。

[0050] S202,当被测程序根据当前测试用例执行分支时,输出对应的基本块信息。

[0051] 测试用例是由测试数据和预期结果构成的,在本发明实施例中可以采用多个测试

用例，这些测试用例包括执行成功的测试用例和执行失败的测试用例。执行不同测试用例可以获得不同的分支信息。

[0052] S203，根据基本块信息得到基本块图级别的执行序列，根据执行序列与对应的基本块图生成包含行号信息的执行序列信息。

[0053] 执行编译后的程序，输出的包含基本块图级别的执行序列的执行文件为：

[0054] demo.gcda0

[0055] demo.gcda1

[0056] demo.gcda2

[0057] demo.gcda4

[0058] demo.gcda2

[0059] demo.gcda4

[0060] demo.gcda2

[0061] demo.gcda4

[0062] demo.gcda3

[0063] demo.gcda5

[0064] demo.gcda7

[0065] demo.gcda9

[0066] 其中，上述执行文件的第一列和第二列用空格隔开，第一列为分支所属的文件，第二列为分支对应于基本块图中的 arc 号，即 0 表示图 3 中的 arc0，1 表示图 3 中的 arc1，2 表示图 3 中的 arc2，以此类推。

[0067] 在本实施例中，可以通过面向对象、直译式计算机程序设计语言例如 Python 语言处理执行文件以及存放程序的覆盖信息和基本块图信息的文件，得到易读的执行序列信息即包含行号信息的执行序列信息为：

[0068] [2,3,4,5]

[0069] [5,7]

[0070] [5,7]

[0071] [5,7]

[0072] [10,11]

[0073] [13][19]

[0074] 其中，该执行序列信息按照序列的执行顺序记录，中括号表示一个块(block)，内部数字表示该 block 所包含的语句。

[0075] S204，获得多个执行序列信息，根据执行序列信息进行被测程序的错误定位。

[0076] 通过步骤 S201-203 可以获得多个测试用例对应的执行序列信息，根据上述执行序列信息生成多个图，使用子图挖掘算法对生成的图进行挖掘，进而根据挖掘结果对被测程序进行错误定位。

[0077] 上述程序执行序列的获取方法，通过在探针中写入保存函数和还原函数，可以在不修改程序源代码的情况下，实现目标代码的插桩，通过执行测试用例后可以获得程序的多个执行序列，进而可以帮助测试人员实现软件工程任务，比如错误定位。

[0078] 图 4 是根据本发明一个实施例的程序执行序列的获取装置的结构示意图。如图 4

所示，程序执行序列的获取装置包括：插桩模块 410、执行模块 420 和生成模块 430。

[0079] 插桩模块 410 用于在编译阶段为被测程序选择插桩节点植入探针，在上述探针中写入保存函数、收集序列函数和还原函数；执行模块 420 用于当上述被测程序根据当前测试用例执行分支时，输出对应的基本块信息；生成模块 430 用于根据基本块信息得到基本块图级别的执行序列，根据上述执行序列与对应的上述基本块图生成包含行号信息的执行序列信息。

[0080] 其中，保存函数用于在调用上述收集序列函数之前，将寄存器数据保存到内存中；上述还原函数用于在调用上述收集序列函数之后，将保存在内存中的寄存器数据进行还原。

[0081] 在探针中写入保存函数的目的是保存收集序列函数的上下文场景；在探针中写入还原函数的目的是恢复收集序列函数上下文场景；在探针中写入收集序列函数的目的是记录程序执行序列。

[0082] 另外，该装置还可以包括：绘制模块 400，该绘制模块 400 用于在插桩模块 410 为被测程序选择插桩节点植入探针之前，预处理被测程序的源文件，对上述源文件划分基本块和分支，并绘制基本块图。在绘制模块 400 绘制完基本块图之后，插桩模块 410 可以根据绘制的基本块图确定需要插桩的分支，然后根据需要插桩的分支确定上述插桩节点。

[0083] 进一步地，该装置还可以包括：定位模块 440，如图 5 所示，该定位模块 440 用于在上述生成模块 430 生成包含行号信息的执行序列信息之后，获得多个执行序列信息，根据上述执行序列信息进行上述被测程序的错误定位。

[0084] 上述程序执行序列的获取装置，通过绘制模块 400、插桩模块 410、执行模块 420 和生成模块 430 之间的配合，可以获得被测程序的执行序列，具体实现过程可参见图 1，此处不赘述；通过绘制模块 400、插桩模块 410、执行模块 420、生成模块 430 和定位模块 440 之间的配合，可以根据获得的被测程序的执行序列对被测程序进行错误定位，具体实现过程可参见图 2，此处不赘述。

[0085] 上述程序执行序列的获取装置，通过插桩模块在探针中写入保存函数和还原函数，可以在不修改程序源代码的情况下，实现目标代码的插桩，通过执行测试用例后可以获得程序的多个执行序列，进而可以帮助测试人员实现软件工程任务，比如错误定位等。

[0086] 应当理解，本发明的各部分可以用硬件、软件、固件或它们的组合来实现。在上述实施方式中，多个步骤或方法可以用存储在存储器中且由合适的指令执行系统执行的软件或固件来实现。例如，如果用硬件来实现，和在另一实施方式中一样，可用本领域公知的下列技术中的任一项或他们的组合来实现：具有用于对数据信号实现逻辑功能的逻辑门电路的离散逻辑电路，具有合适的组合逻辑门电路的专用集成电路，可编程门阵列（PGA），现场可编程门阵列（FPGA）等。

[0087] 在本说明书的描述中，参考术语“一个实施例”、“一些实施例”、“示例”、“具体示例”、或“一些示例”等的描述意指结合该实施例或示例描述的具体特征、结构、材料或者特点包含于本发明的至少一个实施例或示例中。在本说明书中，对上述术语的示意性表述不一定指的是相同的实施例或示例。而且，描述的具体特征、结构、材料或者特点可以在任何的一个或多个实施例或示例中以合适的方式结合。

[0088] 尽管已经示出和描述了本发明的实施例，本领域的普通技术人员可以理解：在不

脱离本发明的原理和宗旨的情况下可以对这些实施例进行多种变化、修改、替换和变型，本发明的范围由权利要求及其等同物限定。

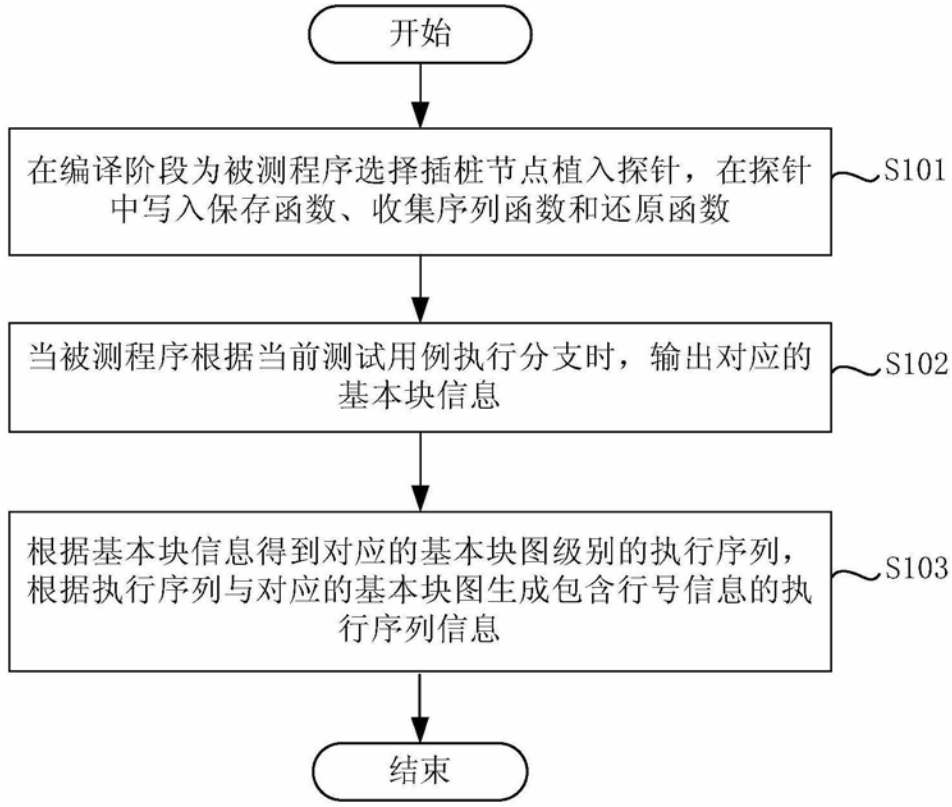

开始
在编译阶段为被测程序选择插桩节点植入探针，在探针中写入保存函数、收集序列函数和还原函数
S101
当被测程序根据当前测试用例执行分支时，输出对应的基本块信息
S102
根据基本块信息得到对应的基本块图级别的执行序列，根据执行序列与对应的基本块图生成包含行号信息的执行序列信息
S103
结束

图 1

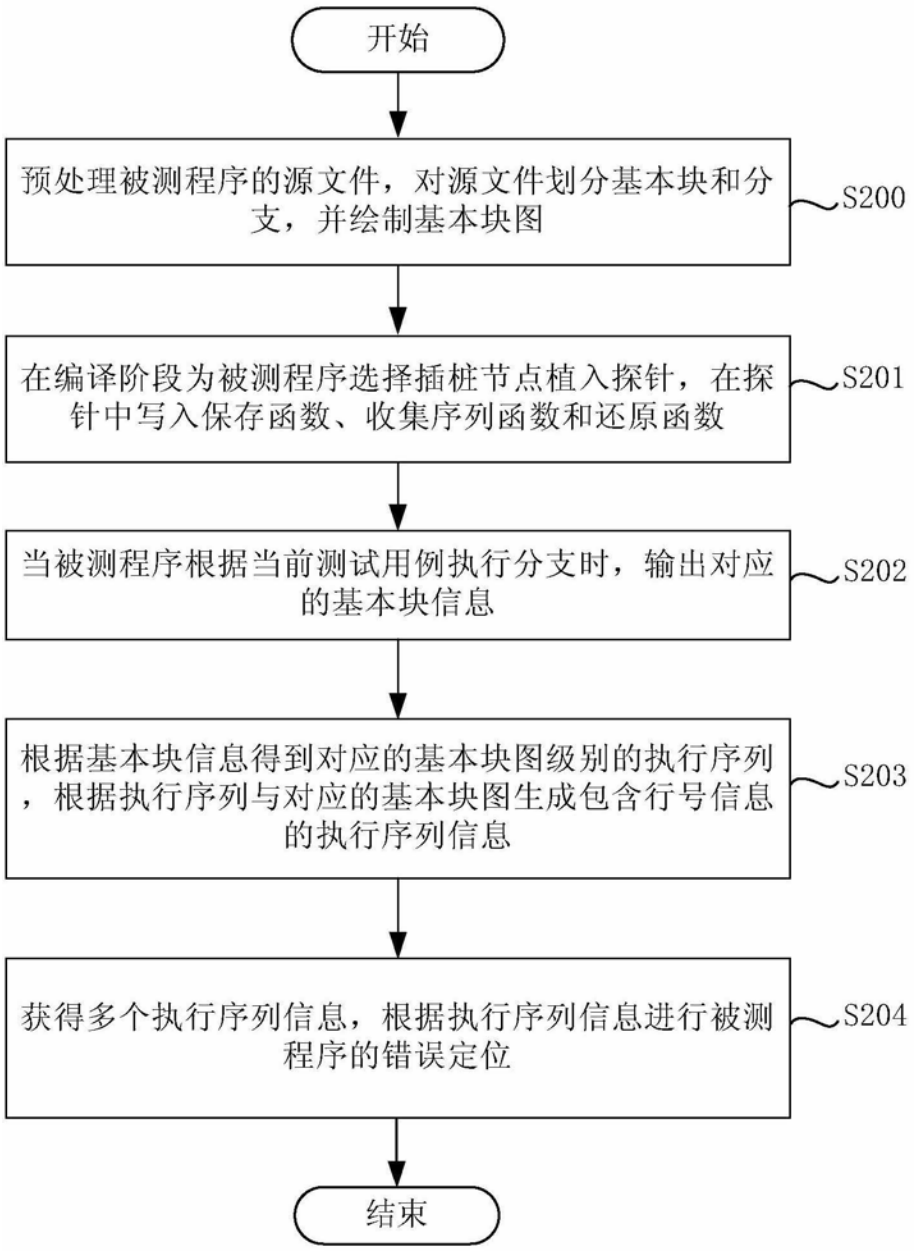

开始
预处理被测程序的源文件，对源文件划分基本块和分支，并绘制基本块图
S200
在编译阶段为被测程序选择插桩节点植入探针，在探针中写入保存函数、收集序列函数和还原函数
S201
当被测程序根据当前测试用例执行分支时，输出对应的基本块信息
S202
根据基本块信息得到对应的基本块图级别的执行序列，根据执行序列与对应的基本块图生成包含行号信息的执行序列信息
S203
获得多个执行序列信息，根据执行序列信息进行被测程序的错误定位
S204
结束

图 2

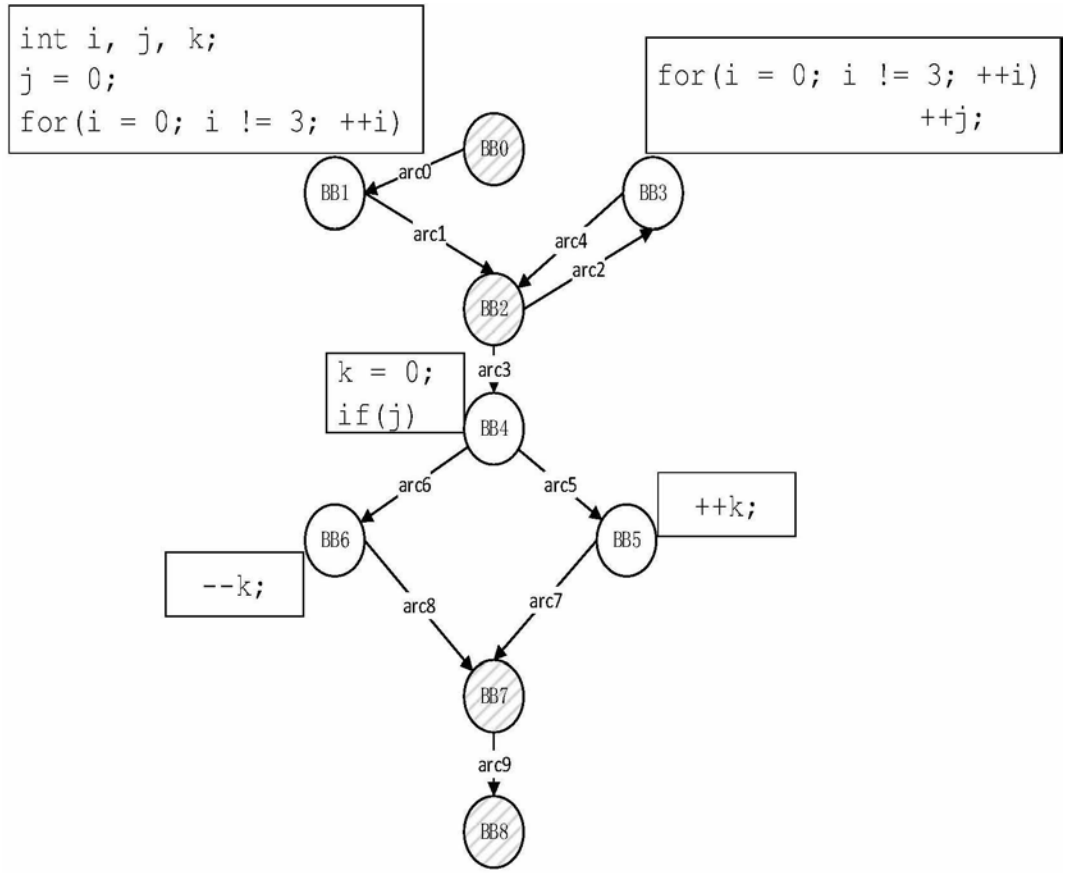

int i, j, k;
j = 0;
for(i = 0; i != 3; ++i)
for(i = 0; i != 3; ++i)
++j;
BB0
arc0
BB1
arc1
BB3
arc4
arc2
BB2
k = 0;
if(j)
arc3
BB4
arc6
arc5
BB6
BB5
++k;
--k;
arc8
arc7
BB7
arc9
BB8

图 3

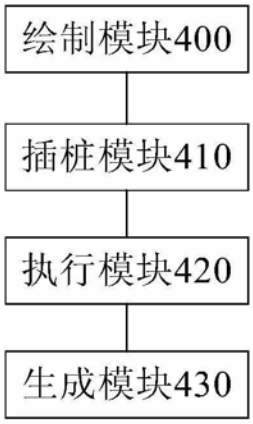

绘制模块400
插桩模块410
执行模块420
生成模块430

图 4


绘制模块400
插桩模块410
执行模块420
生成模块430
定位模块440

图 5