(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 105608003 A
(43)申请公布日 2016.05.25

(21)申请号 201510953537.4

(22)申请日 2015.12.17

(71)申请人 西安电子科技大学
地址 710071 陕西省西安市太白南路2号

(72)发明人 王旭 杨超 孙聪 马建峰 纪倩 张邦元 金方圆 张鹏

(74)专利代理机构 陕西电子工业专利中心 61205
代理人 王品华 王喜媛

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

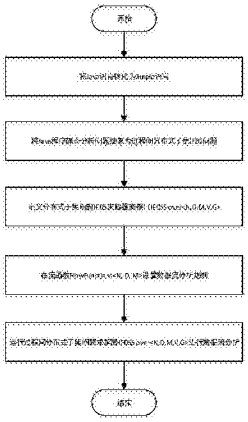
权利要求书3页 说明书6页 附图4页

(54)发明名称

基于控制流分析和数据流分析的Java程序静态分析方法

(57)摘要

本发明公开了一种基于控制流分析和数据流分析的Java程序静态分析方法，主要解决现有静态分析方法分析准确率低的问题。其实现步骤是：1.将Java源程序转化为Jimple语言的中间表示形式；2.将Java程序静态分析问题抽象为过程间分布式子集问题IFDS；3.定义过程间分布式子集问题IFDS求解器的接口类；4.通过求解器类接口中的数据流功能函数FlowFunctions<N,D,M>，设置数据流分析规则；5.运行过程间分布式子集问题IFDS，在控制流图的基础上进行数据流分析，得出分析结果。本发明提高了静态分析的完整性和准确性，分析效率高，扩展性强，可用对较大系统规模的程序分析。



CN 105608003 A

1.一种基于控制流分析和数据流分析的Java程序静态分析方法，包括：

(1)使用开源工具Soot，将Java源程序转化为Jimple语言的中间表示形式，并对中间表示形式进行数据结构的拆分，建立抽象语法树，生成控制流图；

(2)将Java程序静态分析问题抽象为过程间分布式子集问题IFDS，并以过程间分布式子集问题IFDS求解器的方式表示该过程间分布式子集问题IFDS；

(3)定义过程间分布式子集问题IFDS求解器的接口类：

3a)规定过程间分布式子集问题IFDS求解器接口类参数的形式，即用字母N表示节点参数，用字母D表示数据参数，用字母M表示方法参数，用字母V表示值参数，用字母G表示控制流图参数；

3b)规定过程间分布式子集问题IFDS求解器的类接口的名称为：

IFDSSlover<N,D,M,V,G>；

3c)规定过程间分布式子集问题IFDS求解器的类接口IFDSSlover<N,D,M,V,G>中包含有如下四个功能函数：数据流函数FlowFunctions<N,D,M>，控制流图函数interproceduralCFG()，初始化种子函数initialSeeds()，零值表示函数zeroValue()；

(4)在数据流函数FlowFunctions<N,D,M>中设置数据流分析规则：

4a)规定控制流图上节点的类型包括：普通节点，调用节点及返回节点；

4b)根据节点的类型规定节点之间边的类型：

若当前节点为普通节点，且当前节点的下一个节点为普通节点，则规定这两个节点之间边的类型为普通边；

若当前节点为调用节点，且当前节点的下一个节点为普通节点，则规定这两个节点之间边的类型为调用边；

若当前节点为返回节点，且当前节点的下一个节点为普通节点，则规定这两个节点之间边的类型为返回边；

若当前节点为调用节点，且当前节点的下一个节点为返回节点，则规定这两个节点之间边的类型为调用返回边；

4c)根据边的类型确定边上数据流分析的不同操作规程：

对于普通边的操作是：先执行对应的控制流图边上的程序，再以数据等量赋值或数据空值截流的方式，将当前节点的数据传递给当前普通节点的下一个普通节点；

对于调用边的操作是：先执行对应的控制流图边上的程序，再以变量名称替换和数据等量赋值的方式，将当前节点的数据传递给当前调用节点的下一个普通节点；

对于返回边的操作是：先执行对应的控制流图边上的程序，再以变量名称替回和数据等量赋值或数据空值截流的方式，将当前节点的数据传递给当前返回节点的下一个普通节点；

对于调用返回边的操作是：先执行对应的控制流图边上的程序，再以数据等量赋值或数据空值截流的方式，将当前节点的数据传递给返回节点。

(5)运行过程间分布式子集问题IFDS求解器接口类IFDSSlover<N,D,M,V,G>，在控制流图的基础上进行数据流分析：

5a)将控制流图传递给过程间分布式子集问题IFDS求解器接口类IFDSSlover<N,D,M,V,G>中的控制流图参数G，再通过控制流图参数G将控制流图填充到控制流图函数

interproceduralCFG()中；

5b)初始化种子函数initialSeeds()，即以控制流图的第一个节点为起始节点，生成初始化种子，从初始化种子节点开始按照控制流图的执行逻辑，应用数据流分析规则，进行数据流分析；

5c)执行(4)数据流规则，根据步骤4c)数据流操作规程中的数据传递方式，确定具体的值传递操作方式：

若传递方式为数据等量赋值，则直接将该值传递给下一个节点；

若传递方式为数据空值截流，则先调用零值表示函数zeroValue()将空值转化为空值符号NULL，再将该空值符号NULL传递给下一个节点；

5d)判断当前节点传递给下一个节点是否成功：

若当前节点的下一个节点正确接收当前节点传递的等量值或空值符号NULL，则当前节点传递给下一个节点成功，跳转到步骤5e)；

若当前节点的下一个节点不能正确接收当前节点传递的等量值或空值符号NULL，则当前节点传递给下一个节点失败，退出数据流分析；

5e)判断当前节点的下一个节点是否为最后一个节点：

若当前节点的下一个节点不是最后一个节点，则从当前节点的下一个节点开始，返回5c)；

若果当前节点的下一个节点是最后一个节点，则数据流分析完成，输出最终结果。

2.根据权利要求1在数据流函数FlowFunctions<N,D,M>中设置数据流分析规则，其特征在于，步骤4c)中的数据等量赋值，按如下操作进行：

假设赋值的变量为x，被赋值的变量为y；

第1步，操作系统在内存中为变量x分配一块内存空间，在这块内存空间中存储着变量x的实际值；

第2步，将变量x赋值给变量y，操作系统在内存中不给变量y分配空间，直接将变量y指向变量x的内存空间；

第3步，变量x和变量y指向同一块内存空间，改变变量x的实际值，使变量y的实际值随着变量x的实际值的改变而改变，即变量x对变量y的数据等量赋值。

3.根据权利要求1在数据流函数FlowFunctions<N,D,M>中设置数据流分析规则，其特征在于，步骤4c)中的数据空值截流，按如下操作进行：

假设赋值的变量为x，被赋值的变量为y；

第一步，操作系统在内存中为变量x分配一块内存空间，在这块内存空间中存储着变量x的实际值；

第二步，将变量x赋值给变量y，操作系统在内存中不给变量y分配空间，直接将变量y指向变量x的内存空间；

第三步，将变量x的内存空间收回，变量x将不具有内存空间，即x为空值；

第四步，变量x和变量y指向同一块内存空间，变量x为空值，变量y也为空值，即变量x对变量y的数据空值截流。

4.根据权利要求1所述的基于控制流分析和数据流分析的Java程序静态分析方法，其特征在于，步骤5a)中通过控制流图参数G将控制流图填充到控制流图函数

interproceduralCFG()中，是先根据程序的执行逻辑，将控制流图节点中包含的数据以集合的形式传递给控制流图参数G；再调用控制流图函数interproceduralCFG()中的构造函数接收控制流图参数G，并对控制流图参数G进行拆分和自填充。

基于控制流分析和数据流分析的Java程序静态分析方法

技术领域

[0001] 本发明涉及计算机安全领域，更进一步涉及一种检测分析方法，可用于对Java语言环境中的程序静态检测，实现对Java程序的分析。

背景技术

[0002] Java语言，作为软件开发的代表性语言，以它独特的优势占据着市场的主要份额。Java语言是由美国SUN公司开发的一种面向对象的程序设计语言，它比C和C++语言有着更好的通用性、高效性、跨平台性以及安全性。开发者使用Java语言可大大提高软件开发的效率，但同时，随着软件程序代码规模的逐步增加，复杂度的提高，程序代码的检测就会变得越来越困难，这也使得人们在查找漏洞过程中面临更大的挑战。

[0003] 基于Java源程序的漏洞分析和检测现有的方法有很多种，从大的分类来看，其可分为程序执行过程中进行的检测即动态检测和程序源代码的分析检测即静态检测。国内外对Java程序的静态分析和动态分析已经有相当的经验和成果。其中主要针对Java程序语言的几种相对典型的问题，例如内存溢出、Java并行程序、Java applets、数组越界等都是人们在编写Java程序时常常出现或者遗漏在代码中的问题。

[0004] 动态分析最常用的是插桩技术，在程序运行过程中对被检测的代码段插入具有一定功能的检测代码，然后运行程序，收集程序运行过程中的相关信息，以发现程序运行的错误，从而达到检测代码中存在的漏洞的目标。动态分析的难点是有一定的盲目性，在无法预计程序功能块的前提下，盲目的插入检测代码到程序的逻辑块中，影响代码的时间复杂度和空间复杂度。

[0005] 静态分析不同于动态分析，它是在不运行程序的前提下，对程序的执行逻辑逐步分析，生成语法树和流程图，在对程序中某个功能块的分析已经完备的前提下，对程序的运行结果的可能性进行分析，得出结论。但这种静态分析有一定的限制，它只是在分析中去推测程序运行的结果中可能出现的问题，会出现误报或者漏报的情况，影响分析结果的准确性。

发明内容

[0006] 本发明在目的在于针对上述现有静态分析的不足，提出一种基于控制流分析和数据流分析的Java程序静态检测方法，减小误报或者漏报，提高分析结果的准确性。

[0007] 为实现上述目的，本发明的技术方案包括：

[0008] (1)使用开源工具Soot，将Java源程序转化为Jimple语言的中间表示形式，并对中间表示形式进行数据结构的拆分，建立抽象语法树，生成控制流图；

[0009] (2)将Java程序静态分析问题抽象为过程间分布式子集问题IFDS，并以过程间分布式子集问题IFDS求解器的方式表示该过程间分布式子集问题IFDS；

[0010] (3)定义过程间分布式子集问题IFDS求解器的接口类：

[0011] 3a)规定过程间分布式子集问题IFDS求解器接口类参数的形式，即用字母N表示节

点参数，用字母D表示数据参数，用字母M表示方法参数，用字母V表示值参数，用字母G表示控制流图参数；

[0012] 3b)规定过程间分布式子集问题IFDS求解器的类接口的名称为：IFDSSlover<N,D,M,V,G>；

[0013] 3c)规定过程间分布式子集问题IFDS求解器的类接口IFDSSlover<N,D,M,V,G>中包含有如下四个功能函数：数据流函数FlowFunctions<N,D,M>，控制流图函数interproceduralCFG()，初始化种子函数initialSeeds()，零值表示函数zeroValue()；

[0014] (4)在数据流函数FlowFunctions<N,D,M>中设置数据流分析规则：

[0015] 4a)规定控制流图上节点的类型包括：普通节点，调用节点及返回节点；

[0016] 4b)根据节点的类型规定节点之间边的类型：

[0017] 若当前节点为普通节点，且当前节点的下一个节点为普通节点，则规定这两个节点之间边的类型为普通边；

[0018] 若当前节点为调用节点，且当前节点的下一个节点为普通节点，则规定这两个节点之间边的类型为调用边；

[0019] 若当前节点为返回节点，且当前节点的下一个节点为普通节点，则规定这两个节点之间边的类型为返回边；

[0020] 若当前节点为调用节点，且当前节点的下一个节点为返回节点，则规定这两个节点之间边的类型为调用返回边；

[0021] 4c)根据边的类型确定边上数据流分析的不同操作规程：

[0022] 对于普通边的操作是：先执行对应的控制流图边上的程序，再以数据等量赋值或数据空值截流的方式，将当前节点的数据传递给当前普通节点的下一个普通节点；

[0023] 对于调用边的操作是：先执行对应的控制流图边上的程序，再以变量名称替换和数据等量赋值的方式，将当前节点的数据传递给当前调用节点的下一个普通节点；

[0024] 对于返回边的操作是：先执行对应的控制流图边上的程序，再以变量名称替回和数据等量赋值或数据空值截流的方式，将当前节点的数据传递给当前返回节点的下一个普通节点；

[0025] 对于调用返回边的操作是：先执行对应的控制流图边上的程序，再以数据等量赋值或数据空值截流的方式，将当前节点的数据传递给返回节点。

[0026] (5)运行过程间分布式子集问题IFDS求解器接口类IFDSSlover<N,D,M,V,G>，在控制流图的基础上进行数据流分析：

[0027] 5a)将控制流图传递给过程间分布式子集问题IFDS求解器接口类IFDSSlover<N,D,M,V,G>中的控制流图参数G，再通过控制流图参数G将控制流图填充到控制流图函数interproceduralCFG()中；

[0028] 5b)初始化种子函数initialSeeds()，即以控制流图的第一个节点为起始节点，生成初始化种子，从初始化种子节点开始按照控制流图的执行逻辑，应用数据流分析规则，进行数据流分析；

[0029] 5c)执行(4)数据流规则，根据步骤4c)数据流操作规程中的数据传递方式，确定具体的值传递操作方式：

[0030] 若传递方式为数据等量赋值，则直接将该值传递给下一个节点；

[0031] 若传递方式为数据空值截流，则先调用零值表示函数zeroValue()将空值转化为空值符号NULL，再将该空值符号NULL传递给下一个节点；

[0032] 5d)判断当前节点传递给下一个节点是否成功：

[0033] 若当前节点的下一个节点正确接收当前节点传递的等量值或空值符号NULL，则当前节点传递给下一个节点成功，跳转到步骤5e)；

[0034] 若当前节点的下一个节点不能正确接收当前节点传递的等量值或空值符号NULL，则当前节点传递给下一个节点失败，退出数据流分析；

[0035] 5e)判断当前节点的下一个节点是否为最后一个节点：

[0036] 若当前节点的下一个节点不是最后一个节点，则从当前节点的下一个节点开始，返回5c)；

[0037] 若果当前节点的下一个节点是最后一个节点，则数据流分析完成，输出最终结果。

[0038] 本发明与现有技术相比具有以下优点：

[0039] 1.分析效率高。

[0040] 本发明将Java源程序转化为Jimple语言形式，既没有破坏Java语言基本结构特点，又直接对类Java字节码执行了分析，跳过了对Java源码的解析成Java字节码的阶段，执行速度快，分析效率高。

[0041] 2.提高了数据流分析的完整性和准确性。

[0042] 本发明在控制流图的基础上，采用功能函数的方式定义数据流分析规则，依次规定了控制流图中的节点类型、边类型和对应不同边类型的数据流操作方式，克服了现有技术数据流分析单一化的模式，使数据流的分析更加完整和准确。

[0043] 3.扩展性强，易于应用。

[0044] 本发明采用功能函数的定义方式定义数据流分析规则，提供给用户一套完整的模板，用户可以将该模板扩展成针对具体问题的分析，在模板的基础上进行代码的填充，达到特定的分析效果，易于用户应用。

附图说明

[0045] 图1为本发明的实现总流程图；

[0046] 图2为本发明过程间分布式子集问题IFDS接口类定义流程图；

[0047] 图3为本发明数据流分析规则定义流程图；

[0048] 图4为本发明数据流分析过程流程图。

具体实施方式

[0049] 下面结合附图对本发明作进一步的描述。

[0050] 参照图1，本发明的实现步骤如下：

[0051] 步骤1，将Java语言转化为Jimple语言。

[0052] 目前，Java语言的中间表示形式有很多种，包括:Baf，Shimple，Grimp和Jimple，其中Jimple语言是最接近Java字节码的中间表示形式，本实例使用当前唯一的中间语言转化工具Soot，将Java源码转换为Jimple语言的表示形式，接着基于Jimple语言数据规则的拆分，并按照程序的执行逻辑，建立抽象语法树，生成控制流图。

[0053] 步骤2，将Java程序分析问题抽象为过程间分布式子集问题IFDS。

[0054] 根据分布式子集问题IFDS的种类，将实际程序分析问题，转化为分布式子集问题，例如：程序分析问题是分析程序中是否含有未初始化变量，转化为分布式子集问题IFDS为是否存在一条数据流，这条数据流的起始点是未初始化变量，终结点是其它已初始化变量。

[0055] 步骤3，定义过程间分布式子集的接口类。

[0056] 参照图2，本步骤的具体实现如下：

[0057] 3a)规定过程间分布式子集问题IFDS求解器接口类参数的形式，即用字母N表示节点参数，用字母D表示数据参数，用字母M表示方法参数，用字母V表示值参数，用字母G表示控制流图参数；

[0058] 3b)规定过程间分布式子集问题IFDS求解器的类接口的名称为：IFDSSlover<N,D,M,V,G>；

[0059] 3c)规定过程间分布式子集问题IFDS求解器的类接口IFDSSlover<N,D,M,V,G>中包含有如下四个功能函数：数据流函数FlowFunctions<N,D,M>，控制流图函数interproceduralCFG()，初始化种子函数initialSeeds()，零值表示函数zeroValue()。

[0060] 步骤4，在数据流函数FlowFunctions<N,D,M>中，设置数据流分析规则。

[0061] 参照图3，本步骤的具体实现如下：

[0062] 4a)规定控制流图上节点的类型包括：普通节点，调用节点及返回节点；

[0063] 4b)根据节点的类型规定节点之间边的类型：

[0064] 若当前节点为普通节点，且当前节点的下一个节点为普通节点，则规定这两个节点之间边的类型为普通边；

[0065] 若当前节点为调用节点，且当前节点的下一个节点为普通节点，则规定这两个节点之间边的类型为调用边；

[0066] 若当前节点为返回节点，且当前节点的下一个节点为普通节点，则规定这两个节点之间边的类型为返回边；

[0067] 若当前节点为调用节点，且当前节点的下一个节点为返回节点，则规定这两个节点之间边的类型为调用返回边；

[0068] 4c)根据边的类型确定边上数据流分析的不同操作规程，包括四种操作规程：对于普通边的操作、对于调用边的操作、对于返回边的操作和对于调用返回边的操作；具体的操作规程如下：

[0069] 4c1)对于普通边的操作是：先执行对应的控制流图边上的程序，再以数据等量赋值或数据空值截流的方式，将当前节点的数据传递给当前普通节点的下一个普通节点；

[0070] 4c2)对于调用边的操作是：先执行对应的控制流图边上的程序，再以变量名称替换和数据等量赋值的方式，将当前节点的数据传递给当前调用节点的下一个普通节点；

[0071] 4c3)对于返回边的操作是：先执行对应的控制流图边上的程序，再以变量名称替回和数据等量赋值或数据空值截流的方式，将当前节点的数据传递给当前返回节点的下一个普通节点；

[0072] 4c4)对于调用返回边的操作是：先执行对应的控制流图边上的程序，再以数据等量赋值或数据空值截流的方式，将当前节点的数据传递给返回节点。

[0073] 所述的数据等量赋值是：假设赋值的变量为x，被赋值的变量为y，按如下操作进

行：

[0074] 第1步，操作系统在内存中为变量x分配一块内存空间，在这块内存空间中存储着变量x的实际值；

[0075] 第2步，将变量x赋值给变量y，操作系统在内存中不给变量y分配空间，直接将变量y指向变量x的内存空间；

[0076] 第3步，变量x和变量y指向同一块内存空间，改变变量x的实际值，使变量y的实际值随着变量x的实际值的改变而改变，即变量x对变量y的数据等量赋值。

[0077] 所述的数据空值截流是:假设赋值的变量为x，被赋值的变量为y，按如下的操作进行：

[0078] 第一步，操作系统在内存中为变量x分配一块内存空间，在这块内存空间中存储着变量x的实际值；

[0079] 第二步，将变量x赋值给变量y，操作系统在内存中不给变量y分配空间，直接将变量y指向变量x的内存空间；

[0080] 第三步，将变量x的内存空间收回，变量x将不具有内存空间，即变量x为空值；

[0081] 第四步，变量x和变量y指向同一块内存空间，变量x为空值，变量y也为空值，即变量x对变量y的数据空值截流。

[0082] 步骤5，运行过程间分布式子集问题IFDS求解器接口类IFDSSlover<N,D,M,V,G>，在控制流图的基础上进行数据流分析。

[0083] 参照图4，本步骤的具体实现如下：

[0084] 5a)将控制流图传递给过程间分布式子集问题IFDS求解器接口类IFDSSlover<N,D,M,V,G>中的控制流图参数G，再通过控制流图参数G将控制流图填充到控制流图函数interproceduralCFG()中，其中通过控制流图参数G将控制流图填充到控制流图函数interproceduralCFG()中的步骤为：

[0085] 5a1)根据程序的执行逻辑，将控制流图节点中包含的数据以集合的形式传递给控制流图参数G；

[0086] 5a2)调用控制流图函数interproceduralCFG()中的构造函数接收控制流图参数G，并对控制流图参数G进行拆分和自填充；

[0087] 5b)初始化种子函数initialSeeds()，即以控制流图的第一个节点为起始节点，生成初始化种子，从初始化种子节点开始按照控制流图的执行逻辑，应用数据流分析规则，进行数据流分析；

[0088] 5c)执行步骤4数据流规则，根据步骤4c)数据流操作规程中的数据传递方式，确定具体的值传递操作方式：

[0089] 若传递方式为数据等量赋值，则直接将该值传递给下一个节点；

[0090] 若传递方式为数据空值截流，则先调用零值表示函数zeroValue()将空值转化为空值符号NULL，再将该空值符号NULL传递给下一个节点；

[0091] 5d)判断当前节点传递给下一个节点是否成功：

[0092] 若当前节点的下一个节点正确接收当前节点传递的等量值或空值符号NULL，则当前节点传递给下一个节点成功，跳转到步骤5e)；

[0093] 若当前节点的下一个节点不能正确接收当前节点传递的等量值或空值符号NULL，

则当前节点传递给下一个节点失败，退出数据流分析；

[0094] 5e)判断当前节点的下一个节点是否为最后一个节点：

[0095] 若当前节点的下一个节点不是最后一个节点，则从当前节点的下一个节点开始，返回5c)；

[0096] 若果当前节点的下一个节点是最后一个节点，则数据流分析完成，输出最终结果。

[0097] 符号说明

[0098] Baf:基于栈的Java语言中间表示形式；

[0099] Jimple:三地址的基于语句的带类型的Java语言中间表示形式；

[0100] Shimple:基于Jimple，在Jimple语言上添加了静态单点任务分配功能的Java语言中间表示形式；

[0101] Grimp:基于Jimple，在Jimple语言上添加了允许树形态表达和new指令的Java语言中间表示形式；

[0102] Soot:Java到Jimple语言转化工具；

[0103] IFDS:过程间分布式子集问题；

[0104] N:IFDS求解器的节点参数；

[0105] D:IFDS求解器的数据参数；

[0106] M:IFDS求解器的方法参数；

[0107] V:IFDS求解器的值参数；

[0108] G:IFDS求解器的控制流参数；

[0109] IFDSSlover<N,D,M,V,G>:IFDS求解器类接口；

[0110] FlowFunctions<N,D,M>:IFDS求解器类接口中的数据流功能函数；

[0111] interproceduralCFG():IFDS求解器类接口中的控制流图功能参数；

[0112] initialSeeds():IFDS求解器类接口中的出初始化种子功能函数；

[0113] zeroValue():IFDS求解器类接口中的零值表示功能函数；

[0114] NULL:空值表示符号。

[0115] 以上描述仅是本发明的一个具体实例，不构成对本发明的任何限制，显然对于本领域的专业人员来说，在了解了本发明的内容和原理后，都可能在不背离本发明原理、结构的情况下，进行形式和细节上的各种修正和改变，但是这些基于本发明的思想修正和改变仍在本发明的权利要求保护范围之内。

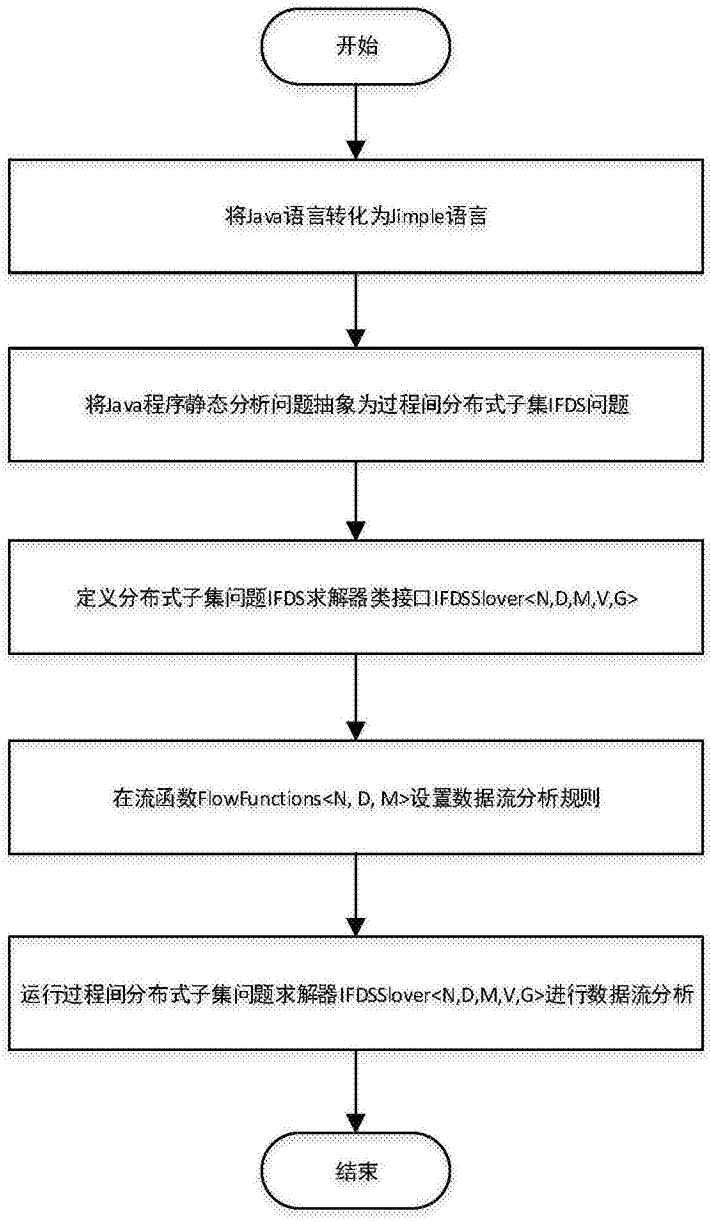

开始
将Java语言转化为Jimple语言
将Java程序静态分析问题抽象为过程间分布式子集IFDS问题
定义分布式子集问题IFDS求解器类接口IFDSSlover<N,D,M,V,G>
在流函数FlowFunctions<N, D, M>设置数据流分析规则
运行过程间分布式子集问题求解器IFDSSlover<N,D,M,V,G>进行数据流分析
结束

图1

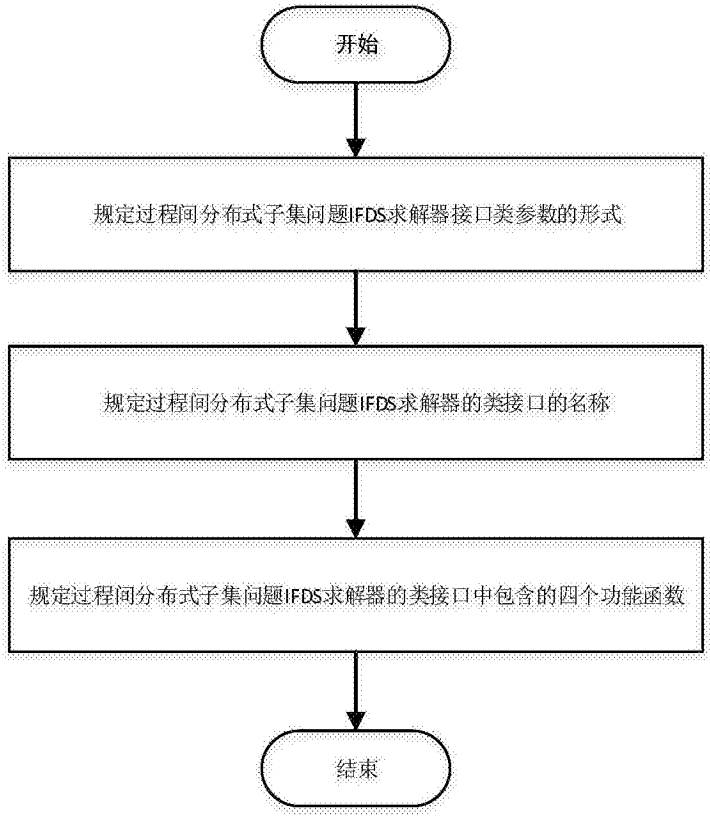

开始
规定过程间分布式子集问题IFDS求解器接口类参数的形式
规定过程间分布式子集问题IFDS求解器的类接口的名称
规定过程间分布式子集问题IFDS求解器的类接口中包含的四个功能函数
结束

图2

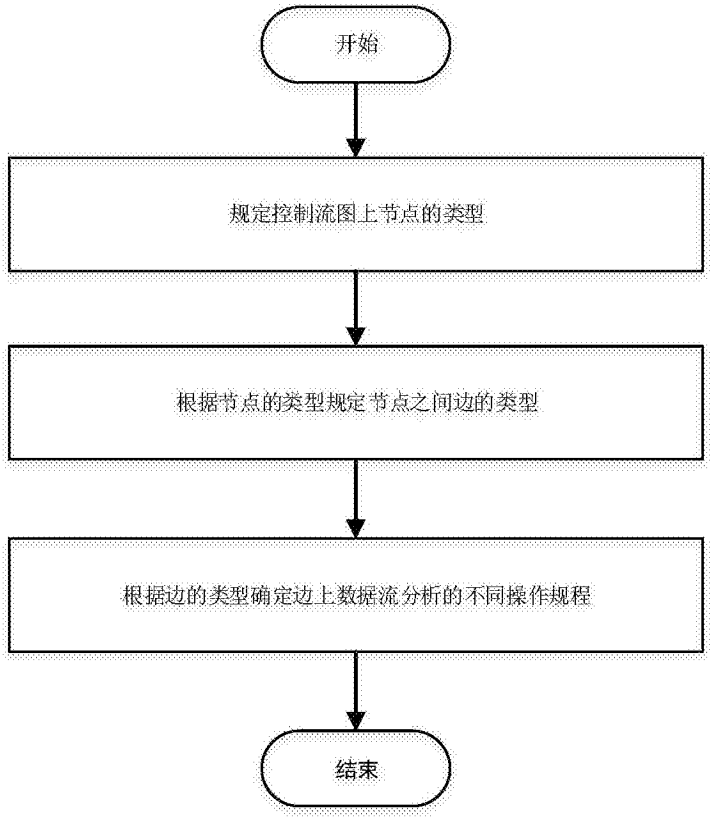

开始
规定控制流图上节点的类型
根据节点的类型规定节点之间边的类型
根据边的类型确定边上数据流分析的不同操作规程
结束

图3

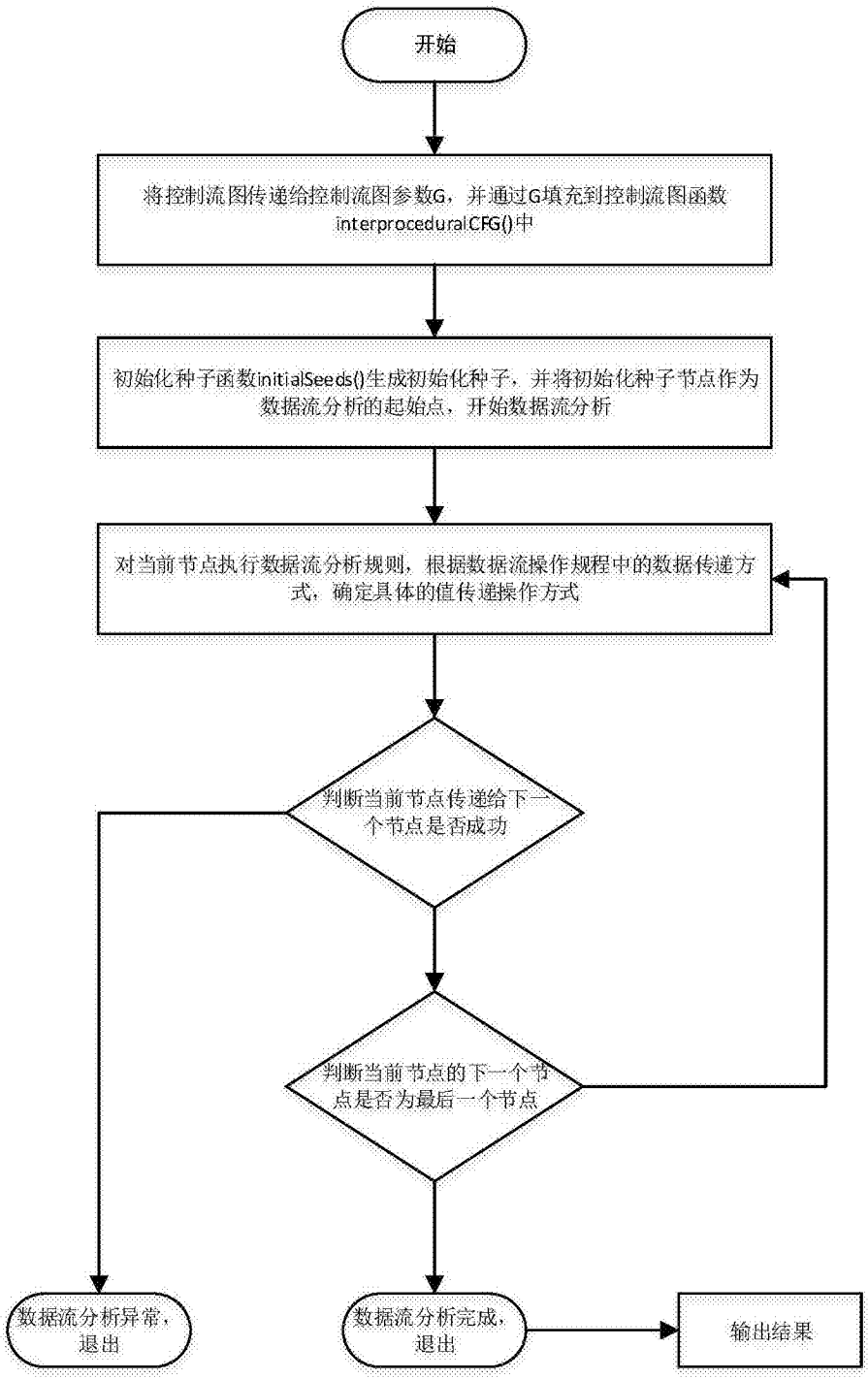

开始
将控制流图传递给控制流图参数G，并通过G填充到控制流图函数interproceduralCFG()中
初始化种子函数initialSeeds()生成初始化种子，并将初始化种子节点作为数据流分析的起始点，开始数据流分析
对当前节点执行数据流分析规则，根据数据流操作规程中的数据传递方式，确定具体的值传递操作方式
判断当前节点传递给下一个节点是否成功
判断当前节点的下一个节点是否为最后一个节点
数据流分析异常，退出
数据流分析完成，退出
输出结果

图4