(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 112860262 B
(45)授权公告日 2024.06.07

(21)申请号 202110175002.4

(22)申请日 2021.02.09

(65)同一申请的已公布的文献号
申请公布号 CN 112860262 A

(43)申请公布日 2021.05.28

(73)专利权人 上海商汤智能科技有限公司
地址 200233 上海市徐汇区桂平路391号3号楼1605A室

(72)发明人 李秀红 陈仁泽 李懋林 颜深根

(74)专利代理机构 北京中知恒瑞知识产权代理有限公司 11889
专利代理师 吴迪

(51)Int.Cl.
G06F 8/41(2018.01)
G06N 20/00(2019.01)

(56)对比文件
KR 20190043850 A,2019.04.29
US 2010306751 A1,2010.12.02
CN 101689232 A,2010.03.31
CN 102323772 A,2012.01.18
CN 110610423 A,2019.12.24
CN 111767055 A,2020.10.13
US 10628584 B1,2020.04.21
US 2001011371 A1,2001.08.02
US 2013073523 A1,2013.03.21
WO 2019102786 A1,2019.05.31
Cunsheng Ding.Bent Vectorial Functions, Codes and Designs.IEEE.2019,第65卷(第11期),7533 - 7541.
王学瑞.函数式编程语言发展及应用.计算机光盘软件与应用.2012,(第23期),181-182.
袁华强,孙永强.基于Monad的纯函数式程序设计.软件学报.1996,(第11期),8.

审查员 刘晓洋

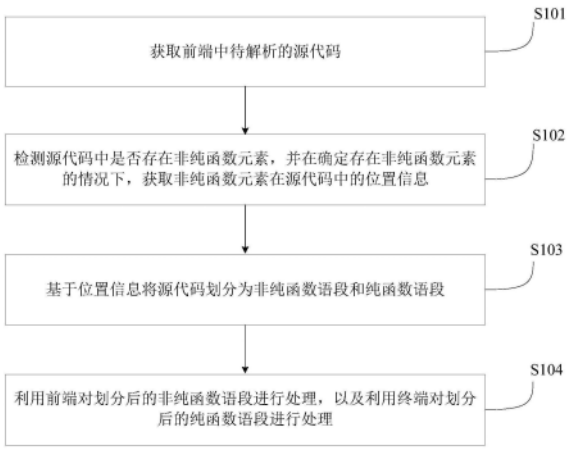
权利要求书2页 说明书11页 附图3页

(54)发明名称

一种代码解析的方法、装置、电子设备及存储介质

(57)摘要

本公开提供了一种代码解析的方法、装置、电子设备及存储介质,其中,该方法包括:获取前端中待解析的源代码;检测源代码中是否存在非纯函数元素,并在确定存在非纯函数元素的情况下,获取非纯函数元素在源代码中的位置信息;基于位置信息将源代码划分为非纯函数语段和纯函数语段;利用前端对划分后的非纯函数语段进行处理,以及利用终端对划分后的纯函数语段进行处理。本公开可以基于非纯函数元素进行语段的划分,这使得有关源代码中的纯函数语段依然是可以在终端运行的,整体的计算性能更佳。



CN 112860262 B

1.一种代码解析的方法,其特征在于,所述方法包括:

获取前端中待解析的源代码;

检测所述源代码中是否存在非纯函数元素,并在确定存在非纯函数元素的情况下,获取所述非纯函数元素在所述源代码中的位置信息;

基于所述位置信息将所述源代码划分为非纯函数语段和纯函数语段;

利用前端对划分后的所述非纯函数语段进行处理,以及利用终端对划分后的所述纯函数语段进行处理;

其中,所述纯函数语段至少为一个;所述利用前端对划分后的所述非纯函数语段进行处理,以及利用终端对划分后的所述纯函数语段进行处理,包括:

依次将至少一个所述纯函数语段进行封装,得到每个纯函数语段对应的封装语句;

构造包含至少一个所述封装语句和所述非纯函数语段的复合函数;

在调用所述复合函数的过程中,利用前端对所述复合函数中的所述非纯函数语段进行处理,以及利用终端调用所述复合函数中包含的每个所述封装语句,并对该封装语句指示的纯函数语段进行处理。

2.根据权利要求1所述的方法,其特征在于,所述检测所述源代码中是否存在非纯函数元素,包括:

将获取的所述源代码对应的字符序列进行词法分析,得到至少一个单词;

对所述至少一个单词进行语法分析,得到不同单词之间的语法关系;

针对所述至少一个单词中的每个单词,基于与该单词存在语法关系的关联单词,确定该单词是否为非纯函数元素。

3.根据权利要求2所述的方法,其特征在于,所述基于与该单词存在语法关系的关联单词,确定该单词是否为非纯函数元素,包括:

针对所述至少一个单词中的每个单词,在确定与该单词存在语法关系的关联单词指示该单词为非局部变量、局部静态变量、可修改的引用参数、以及输入输出流中的任一种的情况下,确定该单词为非纯函数元素。

4.根据权利要求1-3任一所述的方法,其特征在于,检测所述源代码中是否存在非纯函数元素,包括:

对获取的前端中待解析的源代码包括的语句进行拆分处理,得到拆分处理后的源代码;

检测所述拆分处理后的源代码中是否存在非纯函数元素。

5.根据权利要求4所述的方法,其特征在于,所述对获取的前端中待解析的源代码包括的语句进行拆分处理,得到拆分处理后的源代码,包括:

针对所述前端中待解析的源代码中的函数调用语句,从所述函数调用语句中拆分出函数体,将拆分出的函数体作为所述拆分处理后的源代码;和/或,

针对所述前端中待解析的源代码中的目标表达式语句,对该目标表达式语句进行拆分,将得到的拆分后的多个子目标表达式语句作为所述拆分处理后的源代码。

6.根据权利要求1-5任一所述的方法,其特征在于,所述基于所述位置信息将所述源代码划分为非纯函数语段和纯函数语段,包括:

将所述位置信息所指示位置处的语段,确定为所述非纯函数元素对应的非纯函数语

段;以及,

基于所述非纯函数语段,以及所述源代码,确定所述源代码中的纯函数语段。

7.根据权利要求6所述的方法,其特征在于,在确定存在多个所述非纯函数元素的情况下,所述基于所述非纯函数语段,以及所述源代码,确定所述源代码中的纯函数语段,包括:

针对确定的所述源代码中的非纯函数语段,将位于不同的非纯函数语段之间、第一个非纯函数语段之前、以及最后一个非纯函数语段之后的语段,作为所述源代码中的纯函数语段。

8.一种代码解析的装置,其特征在于,所述装置包括:

获取模块,用于获取前端中待解析的源代码;

检测模块,用于检测所述源代码中是否存在非纯函数元素,并在确定存在非纯函数元素的情况下,获取所述非纯函数元素在所述源代码中的位置信息;

划分模块,用于基于所述位置信息将所述源代码划分为非纯函数语段和纯函数语段;

处理模块,用于利用前端对划分后的所述非纯函数语段进行处理,以及利用终端对划分后的所述纯函数语段进行处理;

其中,所述纯函数语段至少为一个;所述处理模块用于按照以下步骤利用前端对划分后的非纯函数语段进行处理,以及利用终端对划分后的纯函数语段进行处理:

依次将至少一个纯函数语段进行封装,得到每个纯函数语段对应的封装语句;

构造包含至少一个封装语句和非纯函数语段的复合函数;

在调用复合函数的过程中,利用前端对复合函数中的非纯函数语段进行处理,以及利用终端调用复合函数中包含的每个封装语句,并对该封装语句指示的纯函数语段进行处理。

9.一种电子设备,其特征在于,包括:处理器、存储器和总线,所述存储器存储有所述处理器可执行的机器可读指令,当电子设备运行时,所述处理器与所述存储器之间通过总线通信,所述机器可读指令被所述处理器执行时执行如权利要求1至7任一所述的代码解析的方法的步骤。

10.一种计算机可读存储介质,其特征在于,该计算机可读存储介质上存储有计算机程序,该计算机程序被处理器运行时执行如权利要求1至7任一所述的代码解析的方法的步骤。

一种代码解析的方法、装置、电子设备及存储介质

技术领域

[0001] 本公开涉及深度学习技术领域,具体而言,涉及一种代码解析的方法、装置、电子设备及存储介质。

背景技术

[0002] 目前,深度学习网络广泛应用在各个领域,基于深度学习网络的各种深度学习框架和终端也越来越多。这里的深度学习框架包括TensorFlow、MXNet、Keras和PyTorch等,这里的终端包括中央处理器(Central Processing Unit,CPU)、图形处理器(Graphics Processing Unit,GPU)等深度学习处理器。为了适应不同端的应用需求,上述深度学习框架和终端之间所采用的编程语言往往不同,因此一款支持前端各种深度学习框架和终端各种终端的编译器显得尤为重要。

[0003] 现有的深度学习编译器可以将算法研究员编写的高层次语言代码(例如Python)自动生成为终端代码(例如CUDA)来提高计算执行性能。然而对于非纯函数而言,由于其具有的特殊性,深度学习编译器无法处理,这将导致包含非纯函数的所有源代码均无法完成编译,这使得终端的计算性能大大减低。

发明内容

[0004] 本公开实施例至少提供一种代码解析的方法、装置、电子设备及存储介质。

[0005] 第一方面,本公开实施例提供了一种代码解析的方法,所述方法包括:

[0006] 获取前端中待解析的源代码;

[0007] 检测所述源代码中是否存在非纯函数元素,并在确定存在非纯函数元素的情况下,获取所述非纯函数元素在所述源代码中的位置信息;

[0008] 基于所述位置信息将所述源代码划分为非纯函数语段和纯函数语段;

[0009] 利用前端对划分后的所述非纯函数语段进行处理,以及利用终端对划分后的所述纯函数语段进行处理。

[0010] 这里,在获取到前端中待解析的源代码的情况下,首先检测源代码中是否存在非纯函数元素,并能够在确定存在非纯函数元素的情况下,获取非纯函数元素在源代码中的位置信息,该位置信息可以将源代码划分为非纯函数语段和纯函数语段,对于划分后的非纯函数语段可以由前端来处理,对于划分后的纯函数语段可以由终端来处理。

[0011] 可见,对于包含有非纯函数元素的源代码而言,本公开实施例可以基于非纯函数元素进行语段的划分,这使得有关源代码中的纯函数语段依然是可以在终端运行的,相比相关技术中由于包含有非纯函数元素而导致整个源代码无法编译进而无法在终端运行而导致计算性能较差相比,整体的计算性能更佳。

[0012] 在一种可能实施方式中,所述检测所述源代码中是否存在非纯函数元素,包括:

[0013] 将获取的所述源代码对应的字符序列进行词法分析,得到至少一个单词;

[0014] 对所述至少一个单词进行语法分析,得到不同单词之间的语法关系;

[0015] 针对所述至少一个单词中的每个单词,基于与该单词存在语法关系的关联单词,确定该单词是否为非纯函数元素。

[0016] 这里,可以结合词法分析和语法分析,确定与源代码对应的每个单词之间存在语法关系的关联单词,利用关联单词所指向的属性信息,即可以确定单词是否为非纯函数元素,通过源代码解析可以进一步提升后续的计算性能。

[0017] 在一种可能的实施方式中,所述基于与该单词存在语法关系的关联单词,确定该单词是否为非纯函数元素,包括:

[0018] 针对所述至少一个单词中的每个单词,在确定与该单词存在语法关系的关联单词指示该单词为非局部变量、局部静态变量、可修改的引用参数、以及输入输出流中的任一种的情况下,确定该单词为非纯函数元素。

[0019] 在一种可能的实施方式中,检测所述源代码中是否存在非纯函数元素,包括:

[0020] 对获取的前端中待解析的源代码包括的语句进行拆分处理,得到拆分处理后的源代码;

[0021] 检测所述拆分处理后的源代码中是否存在非纯函数元素。

[0022] 为了提升非纯函数元素检测的准确性,这里,可以预先对源代码包括的语句进行拆分处理,拆分处理后的源代码去除了复杂操作,以进一步定位非纯函数元素。

[0023] 在一种可能的实施方式中,所述对获取的前端中待解析的源代码包括的语句进行拆分处理,得到拆分处理后的源代码,包括:

[0024] 针对所述前端中待解析的源代码中的函数调用语句,从所述函数调用语句中拆分出函数体,将拆分出的函数体作为所述拆分处理后的源代码;和/或,

[0025] 针对所述前端中待解析的源代码中的目标表达式语句,对该目标表达式语句进行拆分,将得到的拆分后的多个子目标表达式语句作为所述拆分处理后的源代码。

[0026] 在一种可能的实施方式中,所述基于所述位置信息将所述源代码划分为非纯函数语段和纯函数语段,包括:

[0027] 将所述位置信息所指示位置处的语段,确定为所述非纯函数元素对应的非纯函数语段;以及,

[0028] 基于所述非纯函数语段,以及所述源代码,确定所述源代码中的纯函数语段。

[0029] 在一种可能的实施方式中,在确定存在多个所述非纯函数元素的情况下,所述基于所述非纯函数语段,以及所述源代码,确定所述源代码中的纯函数语段,包括:

[0030] 针对确定的所述源代码中的非纯函数语段,将位于不同的非纯函数语段之间、第一个非纯函数语段之前、以及最后一个非纯函数语段之后的语段,作为所述源代码中的纯函数语段。

[0031] 在一种可能的实施方式中,所述纯函数语段至少为一个;所述利用前端对划分后的所述非纯函数语段进行处理,以及利用终端对划分后的所述纯函数语段进行处理,包括:

[0032] 依次将至少一个所述纯函数语段进行封装,得到每个纯函数语段对应的封装语句;

[0033] 构造包含至少一个所述封装语句和所述非纯函数语段的复合函数;

[0034] 在调用所述复合函数的过程中,利用前端对所述复合函数中的所述非纯函数语段进行处理,以及利用终端调用所述复合函数中包含的每个所述封装语句,并对该封装语句

指示的纯函数语段进行处理。

[0035] 这里,为了方便前段和终端进行相应语段的处理,这里,可以对纯函数语段进行封装,通过封装得到的封装语句和非纯函数语段所构造的复合函数,可以快速实现前端对复合函数中的非纯函数语段的处理以及终端对封装语句指示的纯函数语段的处理,进一步提升整体的计算性能。

[0036] 第二方面,本公开实施例还提供了一种代码解析的装置,所述装置包括:

[0037] 获取模块,用于获取前端中待解析的源代码;

[0038] 检测模块,用于检测所述源代码中是否存在非纯函数元素,并在确定存在非纯函数元素的情况下,获取所述非纯函数元素在所述源代码中的位置信息;

[0039] 划分模块,用于基于所述位置信息将所述源代码划分为非纯函数语段和纯函数语段;

[0040] 处理模块,用于利用前端对划分后的所述非纯函数语段进行处理,以及利用终端对划分后的所述纯函数语段进行处理。

[0041] 第三方面,本公开实施例还提供了一种电子设备,包括:处理器、存储器和总线,所述存储器存储有所述处理器可执行的机器可读指令,当电子设备运行时,所述处理器与所述存储器之间通过总线通信,所述机器可读指令被所述处理器执行时执行如第一方面及其各种实施方式任一所述的代码解析的方法的步骤。

[0042] 第四方面,本公开实施例还提供了一种计算机可读存储介质,该计算机可读存储介质上存储有计算机程序,该计算机程序被处理器运行时执行如第一方面及其各种实施方式任一所述的代码解析的方法的步骤。

[0043] 关于上述代码解析的装置、电子设备、及计算机可读存储介质的效果描述参见上述代码解析的方法的说明,这里不再赘述。

[0044] 为使本公开的上述目的、特征和优点能更明显易懂,下文特举较佳实施例,并配合所附附图,作详细说明如下。

附图说明

[0045] 为了更清楚地说明本公开实施例的技术方案,下面将对实施例中所需要使用的附图作简单地介绍,此处的附图被并入说明书中并构成本说明书中的一部分,这些附图示出了符合本公开的实施例,并与说明书一起用于说明本公开的技术方案。应当理解,以下附图仅示出了本公开的某些实施例,因此不应被看作是对范围的限定,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他相关的附图。

[0046] 图1示出了本公开实施例所提供的一种代码解析的方法的流程图;

[0047] 图2示出了本公开实施例所提供的代码解析的方法中,代码划分的示意图;

[0048] 图3示出了本公开实施例所提供的一种代码解析的装置的示意图;

[0049] 图4示出了本公开实施例所提供的一种电子设备的示意图。

具体实施方式

[0050] 为使本公开实施例的目的、技术方案和优点更加清楚,下面将结合本公开实施例中附图,对本公开实施例中的技术方案进行清楚、完整地描述,显然,所描述的实施例仅仅

是本公开一部分实施例,而不是全部的实施例。通常在此处附图中描述和示出的本公开实施例的组件可以以各种不同的配置来布置和设计。因此,以下对在附图中提供的本公开的实施例的详细描述并非旨在限制要求保护的本公开的范围,而是仅仅表示本公开的选定实施例。基于本公开的实施例,本领域技术人员在没有做出创造性劳动的前提下所获得的所有其他实施例,都属于本公开保护的范围。

[0051] 应注意到:相似的标号和字母在下面的附图中表示类似项,因此,一旦某一项在一个附图中被定义,则在随后的附图中不需要对其进行进一步定义和解释。

[0052] 本文中术语“和/或”,仅仅是描述一种关联关系,表示可以存在三种关系,例如,A和/或B,可以表示:单独存在A,同时存在A和B,单独存在B这三种情况。另外,本文中术语“至少一种”表示多种中的任意一种或多种中的至少两种的任意组合,例如,包括A、B、C中的至少一种,可以表示包括从A、B和C构成的集合中选择的任意一个或多个元素。

[0053] 经研究发现,现有的深度学习编译器可以将算法研究员编写的高层次语言代码(例如Python)自动生成为终端代码(例如CUDA)来提高计算执行性能。然而对于非纯函数而言,由于其具有的特殊性,深度学习编译器无法处理,这将导致包含非纯函数的所有源代码均无法完成编译,这使得终端的计算性能大大减低。

[0054] 基于上述研究,本公开提供了一种代码解析的方法、装置、电子设备及存储介质,以通过代码解析提升终端的计算性能。

[0055] 为便于对本实施例进行理解,首先对本公开实施例所公开的一种代码解析的方法进行详细介绍,本公开实施例所提供的代码解析的方法的执行主体一般为具有一定计算能力的计算机设备,该计算机设备例如包括:终端设备或服务器或其它处理设备,终端设备可以为用户设备(User Equipment,UE)、移动设备、用户终端、终端、蜂窝电话、无绳电话、个人数字助理(Personal Digital Assistant,PDA)、手持设备、计算设备、车载设备、可穿戴设备等。在一些可能的实现方式中,该代码解析的方法可以通过处理器调用存储器中存储的计算机可读指令的方式来实现。

[0056] 参见图1所示,为本公开实施例提供的代码解析的方法的流程图,方法包括步骤S101~S104,其中:

[0057] S101:获取前端中待解析的源代码;

[0058] S102:检测源代码中是否存在非纯函数元素,并在确定存在非纯函数元素的情况下,获取非纯函数元素在源代码中的位置信息;

[0059] S103:基于位置信息将源代码划分为非纯函数语段和纯函数语段;

[0060] S104:利用前端对划分后的非纯函数语段进行处理,以及利用终端对划分后的纯函数语段进行处理。

[0061] 为了便于理解本公开实施例提供的代码解析的方法,首先对该方法的应用场景进行详细说明。上述代码解析的方法主要可以应用于利用编译器实现前端各种深度学习框架到各种终端之间的编译过程,在编译过程中,需要进行的是利用前端编写的高层次语言代码(如Python代码)到适用于终端的代码(如CUDA代码)。其中,这里的前端可以包括TensorFlow、MXNet、Keras和PyTorch等深度学习框架,这里的终端可以包括中央处理器(Central Processing Unit,CPU)、图形处理器(Graphics Processing Unit,GPU)等深度学习处理器。

[0062] 然而,相关技术中,若利用前端编写的高层次语言代码中包含有非纯函数,由于非纯函数具有相同输入情况下,多次执行有可能运行结果不同,或者执行过程会有副作用等特殊性,这导致编译器无法直接处理,进而导致无法提升终端的计算性能。

[0063] 正是为了解决上述问题,本公开实施例才提供了一种代码解析的方法,该方法利用检测到的非纯函数元素对源代码(对应上述高层次语言代码)进行拆分,拆分后的函数语段可以包括非纯函数语段和纯函数语段,这里的纯函数语段不受约束,可以被编译器处理,进而可以转换到终端所适用的代码,这可以提升终端的计算性能。

[0064] 本公开实施例提供的代码解析的方法中待解析的源代码可以是从前端获取的。在前端具有解析需求的情况下,可以将待解析的源代码发送给编译器进行代码解析。

[0065] 以Python代码作为源代码为例,这里待解析的源代码可以是加了装饰器的代码。其中,装饰器可以放在一个函数开始定义的地方,以@符号为语法标识,和这个函数绑定在一起。在意图调用这个函数的时候,第一件事并不是执行这个函数,而是将这个函数作为参数传入装饰器。

[0066] 除此之外,本公开实施例中待解析的源代码还可以是其它需要进行解析的代码,在此不做具体的限制。

[0067] 为了检测源代码中是否存在非纯函数元素,本公开实施例可以结合词法分析和语法分析来实现。这里的非纯函数元素可以指的是导致产生具有相同输入情况下,多次执行有可能运行结果不同,和/或,执行过程会有副作用等影响的变量、参数等等。

[0068] 为了便于进行代码划分,本公开实施例还可以在确定存在非纯函数元素的情况下,获取非纯函数元素的位置信息,以基于这一位置信息将源代码划分为非纯函数语段和纯函数语段。

[0069] 其中,这里检测到的非纯函数元素可以有一个,也可以有多个。针对每个非纯函数元素可以对应有相应的位置信息。与非纯函数元素对应的位置信息指向的可以是非纯函数语段,这样,可以结合源代码以及确定的非纯函数语段来确定源代码中的非纯函数语段。例如,可以将非纯函数之间的语段确定为纯函数语段。

[0070] 在划分完成后,可以利用前端对划分后的非纯函数语段进行处理,以及利用终端对划分后的纯函数语段进行处理。这里,由于纯函数语段中不存在非纯函数元素,从而不会受到编译约束,这样,在被编译器执行对应的代码转换操作的情况下,经过转换操作得到的代码可以在终端执行。

[0071] 考虑到非纯函数元素检测对于实现代码解析的关键作用,接下来可以对检测非纯函数元素的过程进行具体描述。

[0072] 本公开实施例中,可以按照如下步骤确定检测源代码中是否存在非纯函数元素:

[0073] 步骤一、将获取的源代码对应的字符序列进行词法分析,得到至少一个单词;

[0074] 步骤二、对至少一个单词进行语法分析,得到不同单词之间的语法关系;

[0075] 步骤三、针对至少一个单词中的每个单词,基于与该单词存在语法关系的关联单词,确定该单词是否为非纯函数元素。

[0076] 这里,首先可以进行的词法分析,得到分析后的至少一个单词,进而可以对至少一个单词进行语法分析,以确定不同单词之间的语法关系,也即,本公开实施例可以确定不同单词在语法上是否存在关系。

[0077] 本公开实施例中的语法关系可以指示的是多个单词之间满足某一些特征或者某一种条件的关系。

[0078] 这里有关特征的关系可以是一个单词对应指示其后的另一个单词的属性信息(包括数据/变量类型信息、附加参数信息等),例如两个单词对应为“int a”,这里的int可以指示a是整形数据,这可以认为存在语法关系;这里有关条件的关系可以是单词之间存在等式关系,例如f＝a+b,这也可以认为存在语法关系。

[0079] 本公开实施例中,针对每个单词而言,可以基于与该单词存在语法关系的关联单词,确定该单词是否为非纯函数元素。

[0080] 这里,可以依赖于上述实例内容描述的第一种语法关系来确定。也即,在确定与该单词存在语法关系的关联单词指示该单词为非局部变量、局部静态变量、可修改的引用参数、以及输入输出流中的任一种的情况下,确定该单词为非纯函数元素。

[0081] 本公开实施例中,在一个关联单词指示一个单词为非局部变量、局部静态变量、可修改的引用参数、以及输入输出流中的任一种的情况下,可以说明的是,该关联单词可以对应指向的单词的附加属性,这一附加属性说明了单词本身是非纯函数元素。

[0082] 这里的非局部变量可以是指不是在局部作用范围内定义的一个变量,局部静态变量可以指的是在局部变量前,加上关键字static所定义成的局部变量,引用参数可以是由调用部位传入实参的地址的形参,输入输出流可以对应的是代码入口和出口。

[0083] 这里,不管非纯函数元素指向的是哪种情况,在具体执行的过程中,要么是相同输入情况下,多次执行有可能运行结果不同,要么是执行过程会有副作用,要么是两者兼顾,因而这里将导致上述结果的代码元素确定为非纯函数元素。

[0084] 本公开实施例中,为了实现上述词法分析和语法分析的具体过程,以Python代码作为源代码为例,可以通过Python提供的抽象语法树(Abstract Syntax Tree,AST)工具包,进行以上操作,从而可以快速的检测出非纯函数元素。

[0085] 这里,考虑到语句作为源代码的主要组成部分,这里,可以基于语句查找方式来确定是否存在非纯函数元素。对于复杂语句而言,如果直接进行非纯函数元素的检测,一旦确定出存在一个非纯函数元素,将导致这个复杂语句对应的语段被判定为非纯函数语段。然而,在实际应用中,上述复杂语句对应的语段往往存在有纯函数语段,采用上述直接判定方式将导致检测结果不够精准。

[0086] 为了提升检测精准度,这里,可以先对待解析的源代码包括的语句进行拆分处理,再检测拆分处理后的源代码中是否存在非纯函数元素。

[0087] 本公开实施例中可以对源代码中的函数调用语句进行拆分,也可以对目标表达式语句进行拆分。针对函数调用语句而言,可以从函数调用语句中拆分出函数体,将拆分出的函数体作为拆分处理后的源代码,例如,针对函数调用语句f＝and(a+b)而言,可以将a+b拆分出来。针对目标表达式语句而言,可以将对该目标表达式语句进行拆分,将得到的拆分后的多个子目标表达式语句作为拆分处理后的源代码,例如,针对目标表达式语句f＝a+b+c,可以拆分为d＝a+b,f＝d+c。

[0088] 基于上述拆分处理后的源代码可以按照上述非纯函数元素的检测方法从中检测出非纯函数元素,在此不再赘述。

[0089] 本公开实施例提供的代码解析的方法可以基于非纯函数元素在源代码中的位置

信息对源代码进行语段划分。这里,可以将位置信息所指示位置处的语段,确定为非纯函数元素对应的非纯函数语段,与此同时,可以基于非纯函数语段,以及源代码,确定源代码中的纯函数语段。

[0090] 本公开实施例中的非纯函数元素可以有一个,也可以有多个。在存在多个非纯函数语段的情况下,可以将位于不同的非纯函数语段之间、第一个非纯函数语段之前、以及最后一个非纯函数语段之后的语段,作为源代码中的纯函数语段。也即,本公开实施例利用非纯函数元素可以将源代码划分为若干语段,进而可以确定出哪些语段对应纯函数语段,哪些语段对应非纯函数语段。

[0091] 需要说明的是,在实际应用中,上述非纯函数元素可以是同时被检测出来的,也可以是依次被检测出现的,具体采用哪种检测方式,需要依赖于源代码的具体执行方式。

[0092] 这里,以Python代码作为源代码为例,该代码采用的是顺序读取机制,因而,这里,可以采用的是顺序检测非纯函数元素的方式。也即,在检测到第一个非纯函数元素的情况下,可以将第一个非纯函数元素所对应的语段确定为第一个非纯函数语段,将第一个非纯函数元素所在位置之前的语段确定为第一个纯函数语段。针对第一个纯函数语段可以先进行函数封装。进而可以对第一个非纯函数语段再次执行非纯函数元素的检测方法,这样,在检测到第二个非纯函数元素的情况下,可以按照上述方式确定出第二个纯函数语段和第二个非纯函数语段,针对第二个纯函数语段也可以进行函数封装,依次进行,直至遍历完所有的源代码。如图2所示,为语段划分后的示意图。

[0093] 为了便于进行后续的函数调用,本公开实施例可以进行函数封装以及函数重组,具体可以通过如下步骤来实现:

[0094] 步骤一、依次将至少一个纯函数语段进行封装,得到每个纯函数语段对应的封装语句;

[0095] 步骤二、构造包含至少一个封装语句和非纯函数语段的复合函数;

[0096] 步骤三、在调用复合函数的过程中,利用前端对复合函数中的非纯函数语段进行处理,以及利用终端调用复合函数中包含的每个封装语句,并对该封装语句指示的纯函数语段进行处理。

[0097] 这里,针对构造的包含至少一个封装语句和非纯函数语段的复合函数,为了实现有关代码功能,在复合函数被调用的过程中,可以依次对复合函数进行函数展开,以利用前端或终端对对应的语段进行处理。

[0098] 不管是前端处理,还是终端处理,最终的处理结果可以反馈到编译器,以通过编译器实现代码转换。

[0099] 本领域技术人员可以理解,在具体实施方式的上述方法中,各步骤的撰写顺序并不意味着严格的执行顺序而对实施过程构成任何限定,各步骤的具体执行顺序应当以其功能和可能的内在逻辑确定。

[0100] 基于同一发明构思,本公开实施例中还提供了与代码解析的方法对应的代码解析的装置,由于本公开实施例中的装置解决问题的原理与本公开实施例上述代码解析的方法相似,因此装置的实施可以参见方法的实施,重复之处不再赘述。

[0101] 参照图3所示,为本公开实施例提供的一种代码解析的装置的示意图,装置包括:获取模块301、检测模块302、划分模块303和处理模块304;其中,

[0102] 获取模块301,用于获取前端中待解析的源代码;

[0103] 检测模块302,用于检测源代码中是否存在非纯函数元素,并在确定存在非纯函数元素的情况下,获取非纯函数元素在源代码中的位置信息;

[0104] 划分模块303,用于基于位置信息将源代码划分为非纯函数语段和纯函数语段;

[0105] 处理模块304,用于利用前端对划分后的非纯函数语段进行处理,以及利用终端对划分后的纯函数语段进行处理。

[0106] 本公开实施例中,在获取到前端中待解析的源代码的情况下,首先检测源代码中是否存在非纯函数元素,并能够在确定存在非纯函数元素的情况下,获取非纯函数元素在源代码中的位置信息,该位置信息可以将源代码划分为非纯函数语段和纯函数语段,对于划分后的非纯函数语段可以由前端来处理,对于划分后的纯函数语段可以由终端来处理。

[0107] 可见,对于包含有非纯函数元素的源代码而言,本公开实施例可以基于非纯函数元素进行语段的划分,这使得有关源代码中的纯函数语段依然是可以在终端运行的,相比相关技术中由于包含有非纯函数元素而导致整个源代码无法编译进而无法在终端运行而导致计算性能较差相比,整体的计算性能更佳。

[0108] 在一种可能实施方式中,检测模块302,用于按照以下步骤检测源代码中是否存在非纯函数元素:

[0109] 将获取的源代码对应的字符序列进行词法分析,得到至少一个单词;

[0110] 对至少一个单词进行语法分析,得到不同单词之间的语法关系;

[0111] 针对至少一个单词中的每个单词,基于与该单词存在语法关系的关联单词,确定该单词是否为非纯函数元素。

[0112] 在一种可能的实施方式中,检测模块302,用于按照以下步骤基于与该单词存在语法关系的关联单词,确定该单词是否为非纯函数元素:

[0113] 针对至少一个单词中的每个单词,在确定与该单词存在语法关系的关联单词指示该单词为非局部变量、局部静态变量、可修改的引用参数、以及输入输出流中的任一种的情况下,确定该单词为非纯函数元素。

[0114] 在一种可能的实施方式中,检测模块302,用于按照以下步骤检测源代码中是否存在非纯函数元素:

[0115] 对获取的前端中待解析的源代码包括的语句进行拆分处理,得到拆分处理后的源代码;

[0116] 检测拆分处理后的源代码中是否存在非纯函数元素。

[0117] 在一种可能的实施方式中,检测模块302,用于按照以下步骤对获取的前端中待解析的源代码包括的语句进行拆分处理,得到拆分处理后的源代码:

[0118] 针对前端中待解析的源代码中的函数调用语句,从函数调用语句中拆分出函数体,将拆分出的函数体作为拆分处理后的源代码;和/或,

[0119] 针对前端中待解析的源代码中的目标表达式语句,对该目标表达式语句进行拆分,将得到的拆分后的多个子目标表达式语句作为拆分处理后的源代码。

[0120] 在一种可能的实施方式中,划分模块303,用于按照以下步骤基于位置信息将源代码划分为非纯函数语段和纯函数语段:

[0121] 将位置信息所指示位置处的语段,确定为非纯函数元素对应的非纯函数语段;以

及，

[0122] 基于非纯函数语段，以及源代码，确定源代码中的纯函数语段。

[0123] 在一种可能的实施方式中，在确定存在多个非纯函数元素的情况下，划分模块303，用于按照以下步骤基于非纯函数语段，以及源代码，确定源代码中的纯函数语段：

[0124] 针对确定的源代码中的非纯函数语段，将位于不同的非纯函数语段之间、第一个非纯函数语段之前、以及最后一个非纯函数语段之后的语段，作为源代码中的纯函数语段。

[0125] 在一种可能的实施方式中，纯函数语段至少为一个；处理模块304，用于按照以下步骤利用前端对划分后的非纯函数语段进行处理，以及利用终端对划分后的纯函数语段进行处理：

[0126] 依次将至少一个纯函数语段进行封装，得到每个纯函数语段对应的封装语句；

[0127] 构造包含至少一个封装语句和非纯函数语段的复合函数；

[0128] 在调用复合函数的过程中，利用前端对复合函数中的非纯函数语段进行处理，以及利用终端调用复合函数中包含的每个封装语句，并对该封装语句指示的纯函数语段进行处理。

[0129] 关于装置中的各模块的处理流程、以及各模块之间的交互流程的描述可以参照上述方法实施例中的相关说明，这里不再详述。

[0130] 本公开实施例还提供了一种电子设备，如图4所示，为本公开实施例提供的电子设备结构示意图，包括：处理器401、存储器402、和总线403。存储器402存储有处理器401可执行的机器可读指令（比如，图3中的代码解析的装置中获取模块301、检测模块302、划分模块303、处理模块304对应的执行指令等），当电子设备运行时，处理器401与存储器402之间通过总线403通信，机器可读指令被处理器401执行时执行如下处理：

[0131] 获取前端中待解析的源代码；

[0132] 检测源代码中是否存在非纯函数元素，并在确定存在非纯函数元素的情况下，获取非纯函数元素在源代码中的位置信息；

[0133] 基于位置信息将源代码划分为非纯函数语段和纯函数语段；

[0134] 利用前端对划分后的非纯函数语段进行处理，以及利用终端对划分后的纯函数语段进行处理。

[0135] 在一种可能实施方式中，上述处理器401执行的指令中，检测源代码中是否存在非纯函数元素，包括：

[0136] 将获取的源代码对应的字符序列进行词法分析，得到至少一个单词；

[0137] 对至少一个单词进行语法分析，得到不同单词之间的语法关系；

[0138] 针对至少一个单词中的每个单词，基于与该单词存在语法关系的关联单词，确定该单词是否为非纯函数元素。

[0139] 在一种可能的实施方式中，上述处理器401执行的指令中，基于与该单词存在语法关系的关联单词，确定该单词是否为非纯函数元素，包括：

[0140] 针对至少一个单词中的每个单词，在确定与该单词存在语法关系的关联单词指示该单词为非局部变量、局部静态变量、可修改的引用参数、以及输入输出流中的任一种的情况下，确定该单词为非纯函数元素。

[0141] 在一种可能的实施方式中，上述处理器401执行的指令中，检测源代码中是否存在

非纯函数元素,包括:

[0142] 对获取的前端中待解析的源代码包括的语句进行拆分处理,得到拆分处理后的源代码;

[0143] 检测拆分处理后的源代码中是否存在非纯函数元素。

[0144] 在一种可能的实施方式中,上述处理器401执行的指令中,对获取的前端中待解析的源代码包括的语句进行拆分处理,得到拆分处理后的源代码,包括:

[0145] 针对前端中待解析的源代码中的函数调用语句,从函数调用语句中拆分出函数体,将拆分出的函数体作为拆分处理后的源代码;和/或,

[0146] 针对前端中待解析的源代码中的目标表达式语句,对该目标表达式语句进行拆分,将得到的拆分后的多个子目标表达式语句作为拆分处理后的源代码。

[0147] 在一种可能的实施方式中,上述处理器401执行的指令中,基于位置信息将源代码划分为非纯函数语段和纯函数语段,包括:

[0148] 将位置信息所指示位置处的语段,确定为非纯函数元素对应的非纯函数语段;以及,

[0149] 基于非纯函数语段,以及源代码,确定源代码中的纯函数语段。

[0150] 在一种可能的实施方式中,在确定存在多个非纯函数元素的情况下,上述处理器401执行的指令中,基于非纯函数语段,以及源代码,确定源代码中的纯函数语段,包括:

[0151] 针对确定的源代码中的非纯函数语段,将位于不同的非纯函数语段之间、第一个非纯函数语段之前、以及最后一个非纯函数语段之后的语段,作为源代码中的纯函数语段。

[0152] 在一种可能的实施方式中,纯函数语段至少为一个;上述处理器401执行的指令中,利用前端对划分后的非纯函数语段进行处理,以及利用终端对划分后的纯函数语段进行处理,包括:

[0153] 依次将至少一个纯函数语段进行封装,得到每个纯函数语段对应的封装语句;

[0154] 构造包含至少一个封装语句和非纯函数语段的复合函数;

[0155] 在调用复合函数的过程中,利用前端对复合函数中的非纯函数语段进行处理,以及利用终端调用复合函数中包含的每个封装语句,并对该封装语句指示的纯函数语段进行处理。

[0156] 本公开实施例还提供一种计算机可读存储介质,该计算机可读存储介质上存储有计算机程序,该计算机程序被处理器运行时执行上述方法实施例中所述的代码解析的方法的步骤。其中,该存储介质可以是易失性或非易失的计算机可读取存储介质。

[0157] 本公开实施例还提供一种计算机程序产品,该计算机程序产品承载有程序代码,所述程序代码包括的指令可用于执行上述方法实施例中所述的代码解析的方法的步骤,具体可参见上述方法实施例,在此不再赘述。

[0158] 其中,上述计算机程序产品可以具体通过硬件、软件或其结合的方式实现。在一个可选实施例中,所述计算机程序产品具体体现为计算机存储介质,在另一个可选实施例中,计算机程序产品具体体现为软件产品,例如软件开发包(Software Development Kit,SDK)等等。

[0159] 所属领域的技术人员可以清楚地了解到,为描述的方便和简洁,上述描述的系统和装置的具体工作过程,可以参考前述方法实施例中的对应过程,在此不再赘述。在本公开

所提供的几个实施例中,应该理解到,所揭露的系统、装置和方法,可以通过其它的方式实现。以上所描述的装置实施例仅仅是示意性的,例如,所述单元的划分,仅仅为一种逻辑功能划分,实际实现时可以有另外的划分方式,又例如,多个单元或组件可以结合或者可以集成到另一个系统,或一些特征可以忽略,或不执行。另一点,所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些通信接口,装置或单元的间接耦合或通信连接,可以是电性,机械或其它的形式。

[0160] 所述作为分离部件说明的单元可以是或者也可以不是物理上分开的,作为单元显示的部件可以是或者也可以不是物理单元,即可以位于一个地方,或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

[0161] 另外,在本公开各个实施例中的各功能单元可以集成在一个处理单元中,也可以是各个单元单独物理存在,也可以两个或两个以上单元集成在一个单元中。

[0162] 所述功能如果以软件功能单元的形式实现并作为独立的产品销售或使用时,可以存储在一个处理器可执行的非易失的计算机可读取存储介质中。基于这样的理解,本公开的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的部分可以以软件产品的形式体现出来,该计算机软件产品存储在一个存储介质中,包括若干指令用以使得一台计算机设备(可以是个人计算机,服务器,或者网络设备等)执行本公开各个实施例所述方法的全部或部分步骤。而前述的存储介质包括:U盘、移动硬盘、只读存储器(Read-Only Memory,ROM)、随机存取存储器(Random Access Memory,RAM)、磁碟或者光盘等各种可以存储程序代码的介质。

[0163] 最后应说明的是:以上所述实施例,仅为本公开的具体实施方式,用以说明本公开的技术方案,而非对其限制,本公开的保护范围并不局限于此,尽管参照前述实施例对本公开进行了详细的说明,本领域的普通技术人员应当理解:任何熟悉本技术领域的技术人员在本公开揭露的技术范围内,其依然可以对前述实施例所记载的技术方案进行修改或可轻易想到变化,或者对其中部分技术特征进行等同替换;而这些修改、变化或者替换,并不使相应技术方案的本质脱离本公开实施例技术方案的精神和范围,都应涵盖在本公开的保护范围之内。因此,本公开的保护范围应所述以权利要求的保护范围为准。

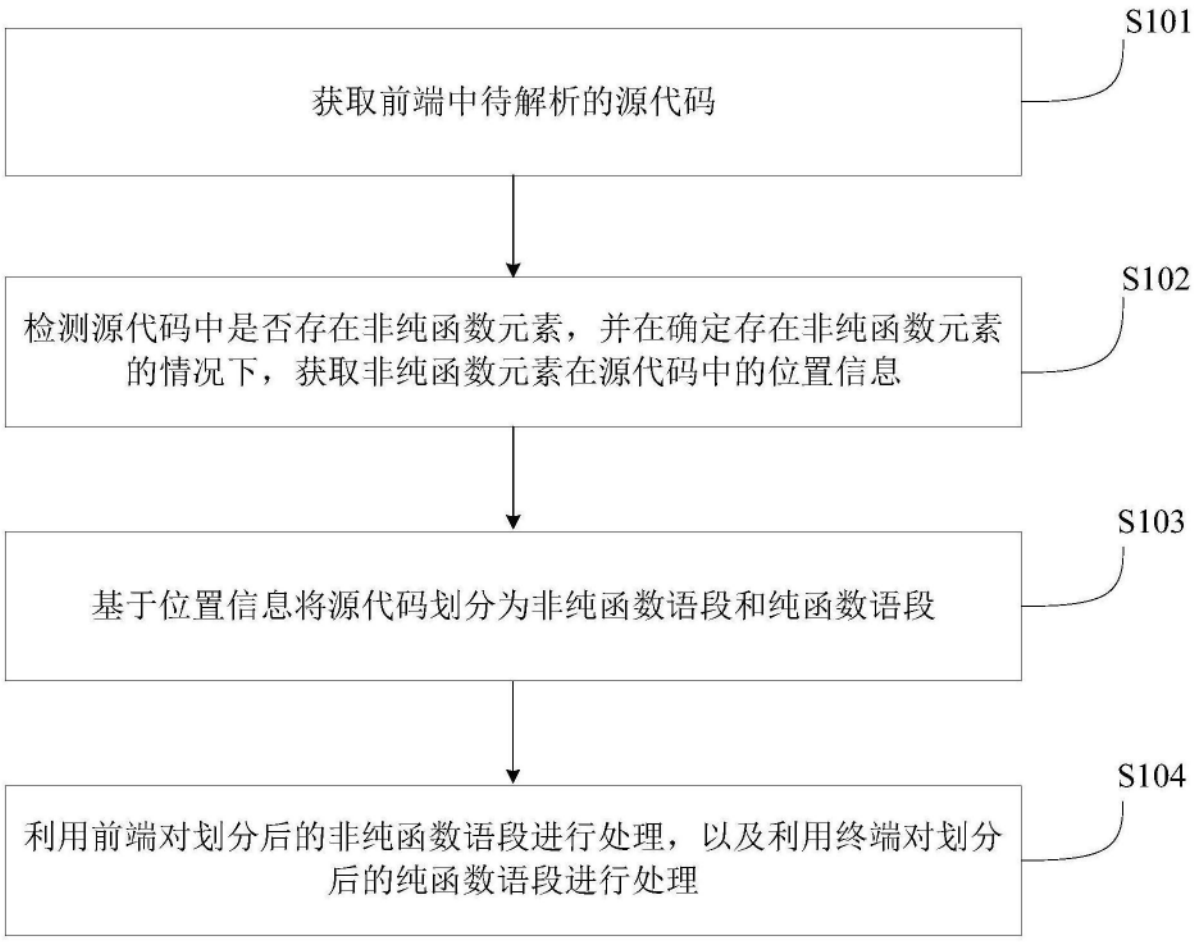

S101
获取前端中待解析的源代码
S102
检测源代码中是否存在非纯函数元素，并在确定存在非纯函数元素的情况下，获取非纯函数元素在源代码中的位置信息
S103
基于位置信息将源代码划分为非纯函数语段和纯函数语段
S104
利用前端对划分后的非纯函数语段进行处理，以及利用终端对划分后的纯函数语段进行处理

图1

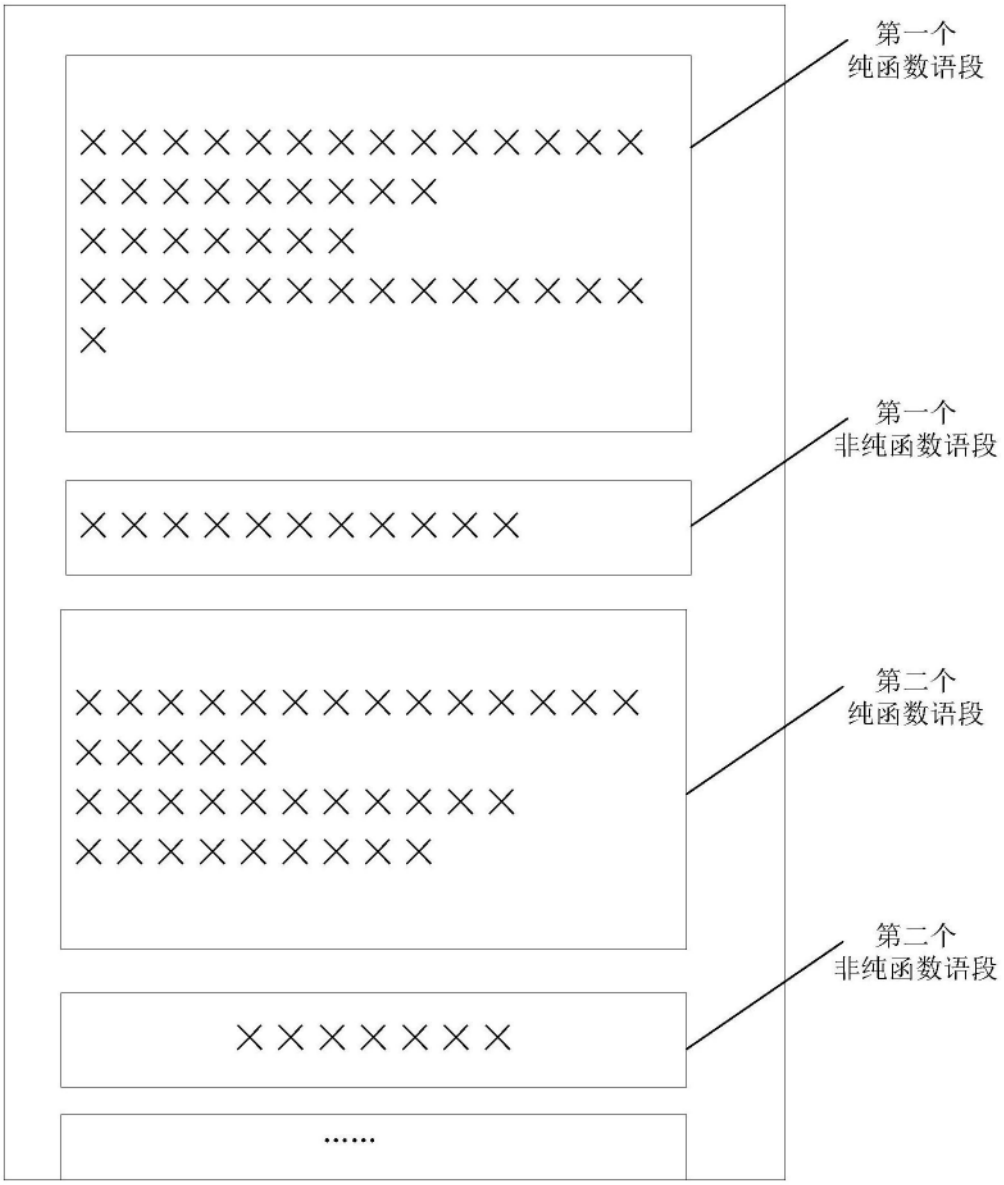


图2

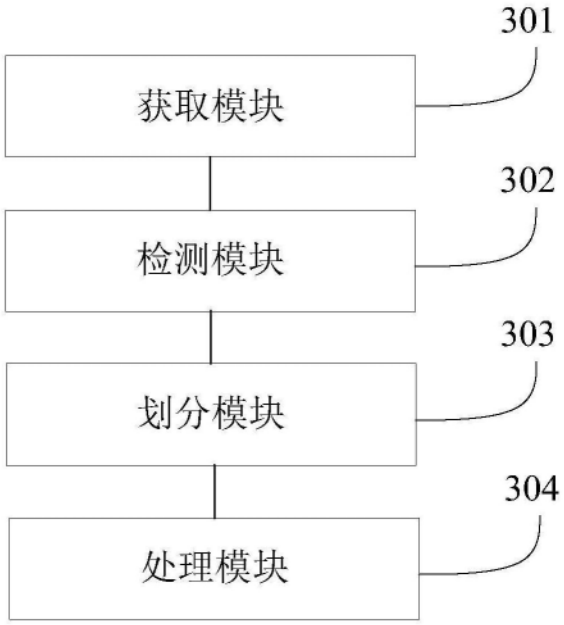

301
获取模块
302
检测模块
303
划分模块
304
处理模块

图3

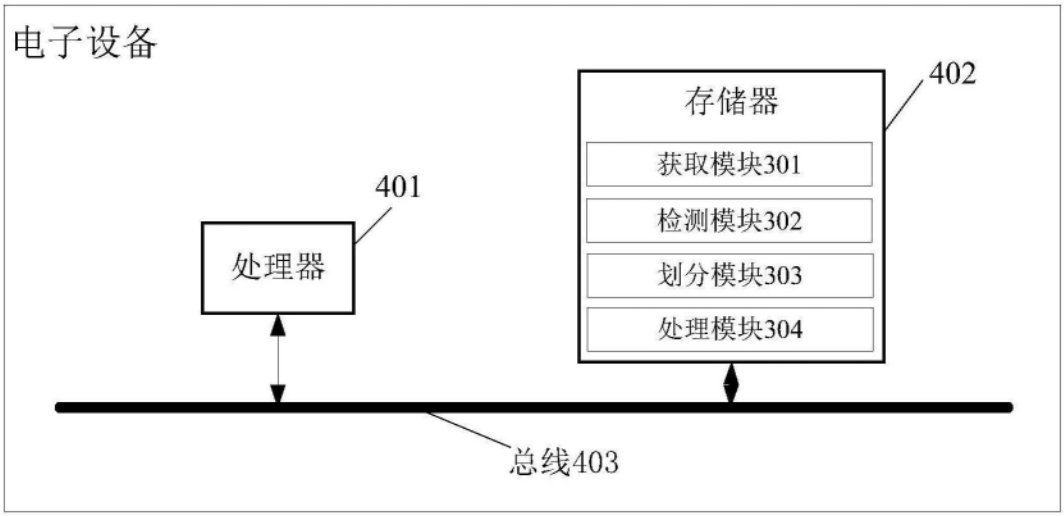

电子设备
存储器
402
获取模块301
检测模块302
划分模块303
处理模块304
401
处理器
总线403

图4