



(12) 发明专利申请

(10) 申请公布号 CN 116257555 A

(43) 申请公布日 2023. 06. 13

(21) 申请号 202211610024.X

(22) 申请日 2022.12.12

(30) 优先权数据

21306752.3 2021.12.12 EP

(71) 申请人 达索系统公司

地址 法国韦利济-维拉库布莱

(72) 发明人 S·C·德科克 F·M·拉贝特

E·L·瓦莱特格莱尼森

J-P·L·M·萨胡特迪扎恩

(74) 专利代理机构 永新专利商标代理有限公司

72002

专利代理师 赵磊

(51) Int.Cl.

G06F 16/2455 (2019.01)

G06F 16/901 (2019.01)

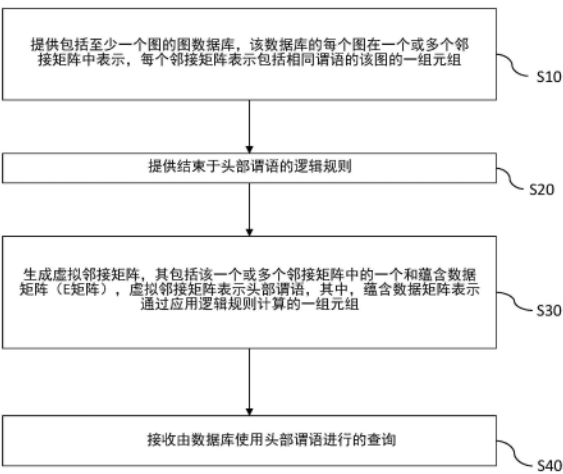
权利要求书1页 说明书11页 附图4页

(54) 发明名称

在SPARQL查询引擎中处理逻辑规则

(57) 摘要

本公开尤其涉及一种用于处理图数据库中的逻辑规则的计算机实现的方法。所述方法包括：提供包括至少一个图的图数据库，所述数据库的每个图在一个或多个邻接矩阵(R矩阵)中表示，每个邻接矩阵表示包括相同谓语的所述图的一组元组；提供结束于头部谓语的逻辑规则；生成虚拟邻接矩阵，其包括所述一个或多个邻接矩阵中的一个和蕴含数据矩阵(E矩阵)，所述虚拟邻接矩阵表示头部谓语，其中，所述蕴含数据矩阵表示通过应用所述逻辑规则计算的一组元组；以及接收由所述数据库使用所述头部谓语进行的查询。



1. 一种计算机实现的用于处理图数据库中的逻辑规则的方法,所述方法包括:
 - 提供 (S10) 包括至少一个图的图数据库,所述数据库的每个图在一个或多个邻接矩阵 (R矩阵) 中表示,每个邻接矩阵表示包括相同谓语的图的一组元组;
 - 提供 (S20) 结束于头部谓语的所述逻辑规则;
 - 生成 (S30) 虚拟邻接矩阵,其包括所述一个或多个邻接矩阵 (R矩阵) 中的一个和蕴含数据矩阵 (E矩阵),所述虚拟邻接矩阵表示所述头部谓语,其中,所述蕴含数据矩阵表示通过应用所述逻辑规则计算的一组元组;以及
 - 接收 (S40) 由所述数据库使用所述头部谓语进行的查询。
2. 根据权利要求1所述的方法,其中,所述虚拟邻接矩阵还包括更新处理器,所述更新处理器被配置为:当所述头部谓语被修改时,更新所述蕴含数据矩阵。
3. 根据权利要求1至2中任一项所述的方法,其中,所述逻辑规则包括一组一个或多个逻辑规则,每个逻辑规则至多为一阶逻辑规则。
4. 根据权利要求3所述的方法,其中,所述逻辑规则表示为线性递归查询。
5. 根据权利要求1至4中任一项所述的方法,其中,所述蕴含数据矩阵的生成包括前向链接技术。
6. 根据权利要求1至5中任一项所述的方法,其中,由各自的邻接矩阵表示的所述图的所述一组元组是非推断元组。
7. 根据权利要求1至6中任一项所述的方法,其中,所述图数据库为DSY.02931.CN1 RDF图数据库,每个元组为:
 - RDF三元组,其包括主语、谓语和宾语;或者
 - RDF四元组,其包括主语、谓语、宾语和图名称。
8. 根据权利要求7所述的方法,其中,所述提供的逻辑规则为RDFS规则。
9. 根据权利要求7所述的方法,其中,所述提供的逻辑规则是被配置为进行以下操作的默认属性规则:返回至少一个RDF元组的宾语或主语的默认值。
10. 根据权利要求6至9中任一项所述的方法,其中,所述接收由所述数据库使用所述头部谓语进行的查询包括:接收由SPARQL查询引擎进行的查询。
11. 根据权利要求10所述的方法,其中,所述方法还包括:
 - 由SPARQL查询引擎回答 (S50) 接收到的查询。
12. 一种计算机程序,其包括用于执行根据权利要求1至11中任一项权利要求所述的方法的指令。
13. 一种计算机可读存储介质,其上记录有根据权利要求12所述的计算机程序。
14. 一种系统,其包括耦合至存储器的处理器,所述存储器上记录有根据权利要求12所述的计算机程序。

在SPARQL查询引擎中处理逻辑规则

技术领域

[0001] 本公开涉及计算机程序和系统领域,更具体地说,本公开涉及用于处理图数据库中的逻辑规则的方法、系统和程序。

背景技术

[0002] 市场上提供了大量用于对象的设计、工程和制造的系统 and 程序。CAD是计算机辅助设计的缩写,例如,它涉及用于设计对象的软件解决方案。CAE是计算机辅助工程的缩写,例如,它涉及用于对未来产品的物理行为进行模拟的软件解决方案。CAM是计算机辅助制造的缩写,例如,它涉及用于定义制造过程和操作的软件解决方案。在这些计算机辅助设计系统中,图用户界面在技术效率方面起着重要作用。这些技术可以嵌入产品生命周期管理 (PLM) 系统中。PLM指的是一种业务战略,其帮助企业在扩展型企业的整个概念中共享产品数据、应用通用流程以及利用企业知识从概念到使用寿命结束来开发产品。Dassault Systèmes (商标为CATIA、ENOVIA和DELMIA) 提供的PLM解决方案提供了工程中心(其组织产品工程知识)、制造中心(其管理制造工程知识) 以及企业中心(其允许将企业集成和连接成工程和制造中心)。该系统共同提供了一种开放对象模型,该模型将产品、过程、资源链接在一起,以实现动态的、基于知识的产品创建和决策支持,从而推动优化的产品定义、制造准备、生产和服务。

[0003] 图数据库特别适用于与数据库交互的应用,例如许多“创意创作应用”,例如CAD软件。此类应用通常在存储器内数据库(即主要依靠存储器进行数据存储的专用数据库)中实现,与将数据存储于磁盘或SSD上的数据库形成对比。

[0004] 对以图式表示和/或存储的知识进行一些推理更有用,即从现有事实中推断出新事实。这种推理通过在由推理引擎表示为图的知识库上应用逻辑规则来推导出新信息。推理过程可以进一步迭代知识库中的新事实以触发推理引擎中的附加规则。推理引擎用于处理知识规则的自动管理。使用知识规则可以分解(即压缩)存储在图中的一些信息,从而不再需要重复该信息,从而使系统的设计更易于理解。经分解后的信息不再在图中明确表示,而是可以通过推理得出。

[0005] 在数据库领域,数据库理论的基础是基于—阶逻辑。—阶逻辑组成关系演算(然后是SQL)但没有递归。在20世纪90年代后期,由于性能成本较高,数据库研究逐渐远离递归查询。Abiteboul等人的文章“Foundations of databases”,Vol.8.Reading:Addison-Wesley,1995,以及Green等人的文章“Datalog and recursive query processing”,Now Publishers,2013,讨论了使用Datalog的研究,尝试将递归添加到关系演算的正(即,没有否定)部分。

[0006] 近年来,递归查询在数据集成与交换、信息抽取、网络、程序分析等各种应用领域广泛兴起。然而,这些方法在查询层面很复杂,难以实现,性能成本高昂,或需要专用的查询引擎。

[0007] 在这种情况下,仍然需要用于处理图数据库中的逻辑规则的改进的方法。

发明内容

[0008] 因此,提供了一种用于处理图数据库中的逻辑规则的计算机实现的方法。所述方法包括:

[0009] -提供包括至少一个图的图数据库,所述数据库的每个图在一个或多个邻接矩阵(R矩阵)中表示,每个邻接矩阵表示包括相同谓语的所述图的一组元组;

[0010] -提供结束于头部谓语的逻辑规则;

[0011] -生成虚拟邻接矩阵,其包括所述一个或多个邻接矩阵(R矩阵)中的一个和蕴含数据矩阵(E矩阵),所述虚拟邻接矩阵表示头部谓语,其中,所述蕴含数据矩阵表示通过应用所述逻辑规则计算的一组元组;以及

[0012] -接收由所述数据库使用所述头部谓语进行的查询。

[0013] 所述方法可以包括下列各项中的一项或多项:

[0014] -所述虚拟邻接矩阵还包括更新处理器,所述更新处理器被配置为:当所述头部谓语被修改时,更新所述蕴含数据矩阵;

[0015] -所述逻辑规则包括一组一个或多个逻辑规则,每个逻辑规则至多为一阶逻辑规则;

[0016] -所述逻辑规则表示为线性递归查询;

[0017] -所述蕴含数据矩阵的生成包括前向链接技术;

[0018] -由各自的邻接矩阵表示的所述图的一组元组是非推断元组;

[0019] -所述图数据库为RDF图数据库,每个元组为:RDF三元组,其包括主语、谓语和宾语;或者RDF四元组,其包括主语、谓语、宾语和图名称;

[0020] -所述提供的逻辑规则为RDFS规则;

[0021] -所述提供的逻辑规则是被配置为进行以下操作的默认属性规则:返回至少一个RDF元组的宾语或主语的默认值;

[0022] -所述接收由所述数据库使用所述头部谓语进行的查询包括:接收由SPARQL查询引擎进行的查询;

[0023] -由SPARQL查询引擎回答接收到的查询。

[0024] 还提供了一种计算机程序,其包括用于执行所述方法的指令。

[0025] 还提供了一种计算机可读存储介质,其上记录有所述计算机程序。

[0026] 还提供了一种系统,其包括耦合至存储器的处理器,所述存储器上记录有所述计算机程序。

附图说明

[0027] 现在将参考附图描述非限制性示例,其中:

[0028] -图1示出了该方法的示例的流程图;

[0029] -图2示出了推理的示例。

[0030] -图3示出了RDFS蕴含模式的示例;

[0031] -图4示出了虚拟邻接矩阵的示意表示;

[0032] -图5示出了前向链接技术的示例;以及

[0033] -图6示出了系统的示例。

具体实施方式

[0034] 参考图1的流程图,提出了一种用于处理图数据库中的逻辑规则的计算机实现的方法。该方法包括:提供S10包括至少一个图的图数据库,数据库的每个图在一个或多个邻接矩阵(也被称为R矩阵)中表示。每个邻接矩阵表示包括相同谓语的图的一组元组。该方法还包括:提供S20得出头部谓语的逻辑规则,并生成S30虚拟邻接矩阵,该虚拟邻接矩阵包括一个或多个邻接矩阵中的一个(即R矩阵)和蕴含数据矩阵(或E矩阵)。虚拟邻接矩阵表示头部谓语。蕴含数据矩阵表示一组通过应用逻辑规则计算的元组。该方法还包括:接收S40数据库使用头部谓语的查询。

[0035] 通过考虑表示数据库的R矩阵和表示通过应用逻辑规则计算的一组元组的E矩阵,这种方法构成了处理图空间中的逻辑规则的改进的解决方案。换句话说,该方法通过以下操作来改进处理:分离已经以数据库的邻接矩阵的形式可用的所需信息的部分和在推理过程中需要通过应用逻辑规则来计算和更新的信息。因此,数据库上的查询引擎能够使用虚拟邻接矩阵作为标准邻接矩阵来回答查询,甚至在不知道逻辑规则的情况下也不会增加复杂度。除了众所周知的规则,例如基于万维网联盟(W3C)设置的语义定义的RDFS(即RDF模式)规则之外,该方法是另外有益的,因为它能够使用自定义逻辑规则来从数据库中推断知识。这些规则在RDF 1.1语义、2014年2月25日的W3C推荐(或另外的草案版本RDF-star)中提出。众所周知,RDFS是使用RDF可扩展知识表示数据模型的具有某些属性的类的集合,其为本体的描述提供基本元素。

[0036] 值得注意的是,该方法在处理的计算时间方面优化了在SPARQL查询引擎中表达为线性递归查询的逻辑规则的处理,而没有增加查询引擎中实施方式的高复杂度。

[0037] “数据库”是指为搜索和检索而组织的任何数据(即信息)集合(例如,面向图的数据库)。如本领域已知的,面向图的数据库是使用图论的面向对象的数据库,因此具有顶点(即节点)和边(即弧),从而能够表示和存储数据。图将存储中的数据项与顶点和边的集合相关联,边表示节点之间的关系。这些关系允许存储中的数据直接链接在一起,并且在许多情况下,可以通过一次操作进行检索。图数据库优先考虑数据之间的关系;与通过隐式连接链接数据的其他数据库模型(例如关系数据库)相反。当存储在存储器中时,图数据库允许计算机快速搜索和检索。特别是,图数据库的结构是为了结合各种数据处理操作快速检索、修改和删除关系。面向图的数据库也被称为图数据库;“面向图的数据库”和“图数据库”是同义词。

[0038] 在示例中,图数据库可以是RDF图数据库。RDF图是一种传统的数据模型,用于图的存储和检索。RDF图是一种有向标记图数据格式。这种格式广泛用于表示Web中的信息。W3C已发布了标准规范,将信息的RDF表示指定为图,例如参见“RDF 1.1 Concepts and Abstract Syntax”,W3C推荐,2014年2月25日。使用的抽象语法的核心结构是元组的集合,每个元组包括谓语。这样的RDF元组的集合被称为RDF图。在示例中,RDF元组可以包括三个或四个包括节点和边的元素。在示例中,每个RDF元组(或每个RDF元组的元素)可以是包括主语、谓语和宾语的三元组。在此类示例中,RDF图可以可视化为节点和有向标记图,其中每个三元组表示为节点-弧-节点链接。或者,RDF三元组可以由两个节点(它们是主语和宾语)以及连接它们的弧(其是谓语)可视化。在示例中,RDF元组可以是RDF四元组。W3C已发布了标准规范来指定RDF四元组(也被称为N-Quads),例如参见“RDF 1.1 N-Quads, A line-based

syntax for RDF datasets”, W3C推荐, 2014年2月25日。因为RDF四元组可以通过将图名称添加到RDF三元组来获得。图名称可以是空的(即, 对于默认或未命名的图)或IRI(例如, 谓语)。每个四元组的图名称是该四元组在数据集中所属的图。在下文中, 术语RDF元组(或元组)无差别地指代RDF三元组或RDF四元组, 除非明确提及其中一个或另一个的使用。

[0039] RDF图数据库可以具有数十亿个元组; 例如, Uniprot数据集是蛋白质序列和功能信息的资源。

[0040] “逻辑规则”(可以等同地被称为“推理规则”或“知识规则”)是指这样一种演绎机制: 当其前提(等同地被称为规则主体)为真时, 其结论(等同地被称为规则的“头部”)为真。逻辑规则的头部是谓语, 等价地被称为“头部谓语”。换句话说, 头部谓语定义了对数据库的相应查询的感兴趣输出。“处理图数据库中的逻辑规则”是指将逻辑规则应用于存储在数据库中的图以推断新信息。

[0041] 参考图1, 在步骤S10中, 该方法包括: 提供包括至少一个图的图数据库。“提供图数据库”是指获得图数据库。计算机系统因此可以对获得的图数据库执行读取和/或写入和/或执行操作。获得图数据库可以从远程或本地存储器、持久性或非持久性存储器中检索图数据库。数据库中的每个图都用一个或多个邻接矩阵(也被称为R矩阵)表示。邻接矩阵是用于表示有限图的方阵。矩阵的元素指示图中的顶点对是否相邻。因此, 每个邻接矩阵表示包括相同谓语的图的一组元组。

[0042] 在示例中, 由相应的邻接矩阵表示的图的这种元组的组可以是非推断元组, 或者(等同地被称为)基本元组。基本元组是显式存在(即存储)在数据库中的元组。换言之, 每个R矩阵(邻接矩阵)表示所提供的图数据库的图的一组元组, 这些元组不是通过对数据库的推测(即推理)获得的。每个R矩阵的元组的组包括(即共享)相同的谓语。换言之, 元组的组根据图的谓语对提供的图数据库进行成组(即, 分区)。这样的分区是有益的, 因为谓语的可变性在图数据库(例如在RDF图数据库中)中很低, 并且特别适合基于垂直分区标准图表示(参见例如, Abadi等人的“Scalable semantic web data management using vertical partitioning”, 发表于Proceedings of the 33rd international conference on Very large data bases, 2007, pp.411-422)。仍然参考图1, 在步骤S20中, 该方法还包括: 提供结束于头部谓语的逻辑规则。“提供逻辑规则”是指例如由用户设置逻辑规则或从库(例如, W3C推荐)导入逻辑规则。“结束于头部谓语的逻辑规则”是指该逻辑规则在其应用于提供的图数据库时推断出信息, 即新图推导出头部谓语的信息, 即提供的逻辑规则的头部。换言之, 逻辑规则在图数据上的应用推导(即推断)头部谓语类型的新信息。头部谓语也可以等价地被称为“区分的查询谓语”, 是头部谓语对应查询的感兴趣输出。

[0043] 在示例中, 所提供的逻辑规则可以包括一组一个或多个逻辑规则, 每个逻辑规则至多是一阶逻辑规则。换言之, 每个提供的逻辑规则可以被重写为一组一个或多个一阶逻辑规则。

[0044] 在示例中, 提供的逻辑规则被表示为线性递归查询, 即, 头部谓语在提供的逻辑规则的主体中仅出现一次。

[0045] 在示例中, 提供的逻辑规则是RDFS规则。例如, 提供的规则可以是W3C语义中的RDFS9规则, 定义如下:

[0046] $\text{Type}(z, y) \leftarrow \text{Type}(z, x), \text{SubClassOf}(x, y)$

[0047] 读作:如果前提“Type(z,x)” (即,z是x类型) 并且“SubClassOf(x,y)” (即类x是y的子类) 为真,则结论“Type(z,y)” (即z是y类型) 为真。在符号中,逗号“,”表示逻辑与。这样的规则是线性递归的,因为“类型”在规则的头部和规则的主体中。

[0048] 在示例中,提供的逻辑规则可以是具有默认属性的类,或者安全规则的评估。具体而言,提供的逻辑规则可以是被配置为进行以下操作的默认属性规则:返回至少一个RDF元组的宾语或主语的默认值。根据本发明的在SPARQL查询引擎中表示为线性递归查询的逻辑规则的处理因此不限于RDFS规则。

[0049] 回到图1,在步骤S30中,该方法生成(计算)虚拟邻接矩阵,该虚拟邻接矩阵包括一个或多个邻接矩阵之一和蕴含数据矩阵,其中蕴含数据矩阵或E矩阵表示头部谓语。“表示头部谓语”是指E矩阵将计算所提供的逻辑规则的结论,即头部谓语。换言之,每个E矩阵是对应于逻辑规则的结论计算的。E矩阵表示一组通过应用逻辑规则计算的元组。换言之,E矩阵表示图数据库的推理结果,用于通过应用所提供的逻辑规则来推断新信息,即基本元组之外的新元组。虚拟邻接矩阵表示区分查询谓语的邻接矩阵,即感兴趣的输出。

[0050] 在示例中,蕴含数据矩阵的生成可以包括前向链接技术。众所周知,“前向链接技术”(或等效的自下而上方法或半朴素算法)是一种从非推断元组中导出新元组的技术。这种前向链接技术可以通过任何已知的方法来执行。这改进了该方法的实施的便利性和响应时间。

[0051] 在生成蕴含数据矩阵的示例中,该方法可以将查询的结果实体化在头谓语的E矩阵中。在这样的示例中,蕴含数据矩阵的生成可以与计算关系数据库中的实体化视图相比较。众所周知,“实体化视图”是包含查询结果的数据库对象。这样的查询可以是使用头部谓语的任何查询。

[0052] 在示例中,虚拟邻接矩阵还可以包括更新处理器。更新处理器被配置为:当头部谓语被修改时更新蕴含数据矩阵。“对头部谓语的修改”是指对头部谓语的相应查询的修改。对头部谓语的这种修改是对规则主体的谓语进行修改的结果,即,不直接修改头部谓语(这导致对R矩阵的修改)。这种修改类似于EDB(即,扩展数据库谓语)的修改,对应于针对SQL数据库从中计算出实体化视图的数据库的源表。在示例中,虚拟邻接矩阵的更新处理器在头部谓语改变时更新虚拟邻接矩阵的E矩阵。

[0053] 回到图1,在步骤S40中,该方法还包括:通过数据库接收数据库使用头部谓语的查询。在图数据库是RDF图数据库的示例中,接收查询可以是SPARQL查询。SPARQL是一种用于RDF数据的查询语言,它能够表达跨不同数据源的查询,无论数据是本地存储为RDF还是经由中间件作为RDF查看。SPARQL包含用于查询所需的和可选的图模式及其合取和析取的功能。SPARQL还支持聚合、子查询、否定、通过表达式创建值、可扩展值测试以及通过源RDF图约束查询。SPARQL查询的结果可以是结果集合或RDF图。在示例中,数据库使用头部谓语接收查询,接收由SPARQL查询引擎进行的查询。接收到的查询可以是任何标准SPARQL查询。

[0054] 在示例中,该方法还可以包括:由SPARQL查询引擎回答接收到的查询。

[0055] 所述方法是计算机实现的。这意味着方法的步骤(或基本上所有步骤)由至少一台计算机或任何类似系统执行。因此,方法的步骤由计算机可能完全自动或半自动地执行。在示例中,可以通过用户-计算机交互来执行方法的至少一些步骤的触发。所需的用户-计算机交互级别可能取决于预见的自动级别,并与实现用户意愿的需求保持平衡。在示例中,该

级别可以是用户定义的和/或预定义的。

[0056] 方法的计算机实现的典型示例是使用适用于此目的的系统来执行该方法。该系统可以包括耦合至存储器和图用户界面 (GUI) 的处理器, 存储器上记录有包括用于执行该方法的指令的计算机程序。存储器还可以存储数据库。存储器是适合于这种存储的任何硬件, 可能包括几个物理上不同的部分 (例如, 一个用于程序, 而一个可能用于数据库)。

[0057] 图6示出了系统的示例, 其中该系统是客户端计算机系统, 例如, 用户的工作站。

[0058] 该示例的客户端计算机包括连接至内部通信总线1000的中央处理单元 (CPU) 1010, 也连接至总线的随机存取存储器 (RAM) 1070。客户端计算机还设有图处理单元 (GPU) 1110, 其与连接到总线的视频随机存取存储器1100相关联。视频RAM 1100在本领域中也被称为帧缓冲器。大容量存储设备控制器1020管理对大容量存储设备 (例如硬盘驱动器1030) 的访问。适用于有形地体现计算机程序指令和数据的大容量存储设备包括所有形式的非易失性存储器, 通过举例的形式, 包括半导体存储设备, 例如EPROM、EEPROM和闪存设备; 磁盘, 例如内部硬盘和可移动磁盘; 磁光盘; 以及CD-ROM盘1040。前述内容中的任何一项都可以由专门设计的ASIC (专用集成电路) 补充或并入其中。网络适配器1050管理对网络1060的访问。客户端计算机还可以包括触觉设备1090, 例如光标控制设备、键盘等。在客户端计算机中使用光标控制设备, 以允许用户有选择地将光标定位在显示器1080上的任何所需位置。此外, 光标控制设备允许用户选择各种命令并输入控制信号。光标控制设备包括多个信号生成设备, 用于将控制信号输入到系统。通常, 光标控制设备可以是鼠标, 该鼠标的按钮用于生成信号。替换地或附加地, 客户端计算机系统可以包括敏垫和/或敏屏。

[0059] 计算机程序可以包括可由计算机执行的指令, 这些指令包括用于使上述系统执行方法的单元。程序可以记录在任何数据存储介质上, 包括系统的存储器。程序可以例如以数字电子电路或计算机硬件、固件、软件或它们的组合来实现。程序可以被实现为装置, 例如, 有形地体现在机器可读存储设备中以供可编程处理器执行的产品。方法步骤可以通过执行指令程序的可编程处理器来执行, 以通过对输入数据进行操作并生成输出来执行方法的功能。因此, 处理器可以是可编程的并且耦合以从数据存储系统、至少一个输入设备和至少一个输出设备接收数据和指令, 以及向数据存储系统、至少一个输入设备和至少一个输出设备发送数据和指令。如果需要, 可以以高级过程或面向对象的编程语言或汇编或机器语言来实现应用。在任何情况下, 语言都可以是编译语言或解释语言。程序可以是完整的安装程序或更新程序。在任何情况下, 程序在系统上的应用都会导致执行该方法的指令。

[0060] 现在讨论上述方法示例的实施方式示例。

[0061] 本发明的实施方式示例处理表示为SPARQL查询引擎中的线性递归查询的逻辑规则。应当理解, 逻辑规则不限于RDFS (RDF模式) 规则 (例如, 如下所述的“默认属性”规则)。

[0062] 实施方式示例还与SPARQL查询相关。如前所述, SPARQL是用于查询RDF的W3C推荐, 是一种建立在RDF三元组的三元组模式之上的图匹配语言。W3C规范定义了用于RDF的SPARQL查询语言的语法和语义。SPARQL可以用于表达跨不同数据源的查询, 无论数据是本地存储为RDF还是经由中间件作为RDF查看。SPARQL包含用于查询所需的和可选的图模式及其合取和析取的功能。SPARQL还支持聚合、子查询、否定、通过表达式创建值、可扩展值测试以及通过源RDF图约束查询。SPARQL查询的结果可以是结果集合或RDF图。

[0063] 实施方式的示例还涉及对数据库中的知识进行推理, 即使用逻辑规则从现有事实

中推断出新事实。这种推断可以使用RDFS推断来执行,如图2所示。在图2中,RDFS规则210表示类Man(男人)215(指资源集)是类Person(人)225(指另一资源集)的子类(子集),而RDF三元组220表示Fabian 235是(的类型是)Man(男人)240。使用RDFS规则210,没有必要说Fabian 235是Man(男人)240和Person(人)225,因为这个事实可以从Fabian235是Man(男人)240,并且Man(男人)240是Person(人)225的子类推导出,因此Fabian 235是Person(人)225。从而推理和RDFS规则消除了数据库中的重复。这样的规则可以被视为一阶逻辑规则。

[0064] 根据W3C语义中的RDFS9规则,RDFS蕴含模式如图3所示。该规则表示:如果前提xxx rdfs:subClassOf yyy(即xxx是yyy的子类)和zzz rdf:type xxx(即zzz是xxx类型)为真,则结论zzz rdf:type yyy(即zzz是yyy类型)因此为真。RDFS9可以表示为线性递归

[0065] $\text{Type}(z, y) \leftarrow \text{Type}(z, x), \text{SubClassOf}(x, y)$

[0066] 读作:如果前提“Type(z,x)”(即,z是x类型)并且“SubClassOf(x,y)”(即类x是y的子类)为真,则结论“Type(z,y)”(即z是y类型)为真。在符号中,逗号“,”表示逻辑与。这样的规则是线性递归的,因为“类型”在规则的头部和规则的主体中。

[0067] 现在讨论“默认属性”示例。

[0068] 考虑以下RDF数据集(示例A):

[0069] @prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

[0070] @prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>.

[0071] @prefix ex:<http://www.example.org#>.

[0072] @prefix er:<https://www.3ds.com/er/>.

[0073] er:ProductAbstract a rdfs:Class.

[0074] er:VPMReference a rdfs:Class.

[0075] er:DomainSpecificRef a rdfs:Class.

[0076] er:VPMReference rdfs:subClassOf er:ProductAbstract.

[0077] er:DomainSpecificRef rdfs:subClassOf er:VPMReference.

[0078] ex:attr10 rdfs:domain er:ProductAbstract.

[0079] ex:attr10 rdfs:subPropertyOf er:Attribute.

[0080] ex:attr10 er:defaultValue"attr10DefaultValue".

[0081] ex:instance1 a er:DomainSpecificRef.

[0082] ex:instance1 ex:attr11"specificValue4".

[0083] 作为SPARQL查询,使用了解RDFS和“默认属性”逻辑规则的查询引擎,如

select ? attr ? value where {

ex:instance1 ? attr ? value.

[0084] *? attr rdfs:subPropertyOf er:Attribute.*

}

[0085] SPARQL查询返回以下结果:

[0086] <http://www.example.org#attr10>"attr10DefaultValue"

[0087] <http://www.example.org#attr11>"specificValue4"

[0088] 因为attr10的值是在ProductAbstract类上获取的。除了RDFS规则之外,该实施方

式可以使用以下“默认属性”规则,以RDFS规则为先决条件:

er:Attribute.# a constant

DefaultAttr(A,v)

[0089]

$\leftarrow \text{SubProperty}(A, \text{er:Attribute}), \text{DefaultValue}(A, v) .$

$A(i, v) \leftarrow \text{DefaultAttr}(A, v), \text{Domain}(A, p), \text{Type}(i, P) .$

[0090] 因此,应用默认属性规则,将A与http://www.example.org#attr10>匹配并将i与ex:instance1匹配,给出输出“attr10DefaultValue”作为v。特别是,RDFS规则也适用于类型(如在RDF 1.1概念和抽象语法中)。谓语DefaultAttr可能未被实体化,在这种情况下,规则可以被重写为:

[0091] $A(i, v)$

[0092] $\leftarrow \text{SubProperty}(A, \text{er:Attribute}), \text{DefaultValue}(A, v), \text{Domain}(A, p), \text{Type}(i, P) .$

[0093] 实施方式可以只考虑线性递归规则,即递归谓语在规则主体中只出现一次。如本领域已知的,关系演算的线性递归查询可以使用Datalog。现在讨论线性递归查询处理的一些已知方法。

[0094] 自下而上的方法可以使用半朴素的算法从基本元组中推导出新的元组。这种方法很容易在查询引擎中实施,但不会最大限度地减少在回答查询时不需要的元组的推导。由于进行了不必要的工作,这会带来性能成本。在此示例中,数据库中带有谓语“链接”的元组(即基本元组)是:

[0095] $\text{link}(a, b), \text{link}(b, c), \text{link}(c, c), \text{link}(c, d)$

[0096] 并且查询被定义为:

[0097] $r1 \text{ reachable}(X, Y) :- \text{link}(X, Y) .$

[0098] $r2 \text{ reachable}(X, Y) :- \text{reachable}(X, Z), \text{link}(Z, Y) .$

[0099] $\text{query}(Y) :- \text{reachable}(b, Y) .$

[0100] 区分查询谓语(即头部谓语或感兴趣的输出)是可达表。输出以常量b为界,表明查询只对节点b的可达元组感兴趣。限制输出集会影影响使用自下而上与自上而下评估技术的相对开销。如图5所示,半朴素计算在三次迭代中完成。

[0101] 自下而上的方法易于实现,但需要大量计算并因此存在性能问题。

[0102] 自上而下的方法旨在通过从查询本身开始评估并将选择标准(即常量)从查询推入规则来仅推导出与查询相关的元组。自上而下的评估是为查询搜索证明树。这种方法的代表性方法被称为查询/子查询(QSQ)。

[0103] 自上而下的方法是高效的,因为它只推导与查询相关的元组,但它很复杂并且需要专用的查询引擎。

[0104] 在文献中,自下而上的方法也被称为前向链接,自上而下的方法也被称为反向链接。术语前向/反向链接通常用于RDF推理引擎。

[0105] 标准的线性递归查询处理方法有一些缺点。在这样的处理中,可以使用简单的策略,如上文讨论的自下而上的方法,但它们效率不高并且会推导出太多元组。另一方面,处理可以使用复杂的策略,如自上而下的方法,这会增加查询引擎的复杂度(或需要专用的查

询引擎)。自上而下方法在查询执行时增加的复杂度限制了它的使用,并可能使其无法在交互式应用中使用。

[0106] SQL数据库中的一个常见解决方案是使用实体化视图。实体化视图是包含查询结果的数据库对象。实体化视图可用于提高执行频繁但复杂查询的Datalog程序的性能。如果EDB(即扩展数据库谓语句)对应于数据库的源表,则实体化视图是根据改变而计算的,则需要维护实体化视图以反映此类改变。因此,实体化视图的高效维护成为维持性能优势的一个重要方面。然而,现有技术中的实体化视图是特定于表格方法的,并且在将数据表示为图方面没有实体化视图。

[0107] 已知的RDF蕴涵引擎实施方式遵循与上文讨论的自下而上和自上而下方法类似的模式。“蕴涵引擎”是指RDFS蕴涵推理引擎。正如RDF 1.1语义、W3C推荐中所讨论的,用于RDF和RDFS的模型理论包含许多蕴涵,它们应该由RDF推理引擎生成。现有系统以几种方式生成这些,但最常见的方法采用生产规则系统,该系统通过从断言事实的前向链接或从呈现给系统的查询的反向链接生成所需的蕴涵。

[0108] 这些方法各有利弊:纯粹的前向链接系统将模型理论中的蕴涵规则详尽地应用于所断言的事实,以生成这些事实的RDF闭包(演绎闭包)。这种对演绎闭包的急切评估(由Sesame等系统执行)具有降低评估查询的处理成本的效果。由查询返回的蕴涵(除了任何基本事实之外)是提前生成的,这在某些情况下可以减少评估查询所花费的时间。这对于快速响应时间至关重要的交互式应用来说是一个优势。这种方法的缺点是RDF闭包的大小可能是断言事实大小的许多倍,并且包含许多很少与查询匹配的蕴涵。

[0109] 相反,纯反向链接系统在查询处理时间方面评估查询所匹配的蕴涵。这降低了存储蕴涵的成本,因为只有那些需要的蕴涵才会以更高贵的查询处理为代价生成。评估蕴涵的反向链接系统可能会导致时间损失,使其无法在交互式应用中使用。尽管通过对生成的蕴涵进行高速缓存可以减少重复查询的成本,但在查询时间方面仍然存在大量成本。

[0110] 典型的查询引擎采用混合方法,在断言新事实时使用前向链接规则对其中的一些蕴涵规则(通常是那些生成较少蕴涵的规则)进行评估。另一方面,具有较大存储成本和较低评估成本的蕴涵在查询时间方面通过反向链接和查询重写的组合根据需要进行评估。这种混合方法有一些缺点。使用结合前向链接、反向链接和查询重写的混合策略,SPARQL查询引擎可以非常有效地评估RDFS规则。然而,它是一种专用于RDFS规则的自定义策略,它不能按原样用于解决更通用的规则,例如上文讨论的“默认属性”或“可达”规则。

[0111] 本发明的实施例可以被设想为受SQL查询引擎中使用的实体化视图方法的启发,而不是在SPARQL查询引擎中增加额外的复杂度,例如使用本领域已知的查询/子查询策略,参见已经引用的文档:Green等人的“Datalog and recursive query processing”,Now Publishers,2013。然而,实体化视图是表格对象,不存在于SPARQL和RDF的图范例中。实施方式的示例表示使用垂直分区的RDF图,其中每个谓语句都是一个邻接矩阵。因此,本着表格数据视图的相同精神,该实施方式定义了一个虚拟邻接矩阵,它表示区分查询谓语句(即感兴趣的输出)的邻接矩阵。

[0112] 例如,在默认属性规则中:

[0113] `DefaultAttr(A,v)`

[0114] `<-SubProperty(A,er:Attribute),DefaultValue(A,v) .`

[0115] $A(i,v) < -\text{DefaultAttr}(A,v), \text{Domain}(A,p), \text{Type}(i,P)$.

[0116] 除了subPropertyOf、DefaultValue等的“具体”邻接矩阵之外,还存在根据实施方式的用于A的虚拟邻接矩阵。虚拟邻接矩阵的作用就好像所有三元组都是在半朴素方法中推导出的,并且矩阵的内容能够回答上文讨论的规则,包括RDFS规则、“默认属性”和“可达”规则。

[0117] 现在针对如上所述的“默认属性”的示例来讨论实施方式的示例。

[0118] 在RDF三元组中,每个主语(S)、谓语(P)和/或宾语(O)都可以是查询变量。这意味着SPARQL查询需要回答SPARQL中可能的八个不同的三元组模式。这八个三元组模式包括(S,P,O)、(S,?P,O)、(S,P,?O)、(S,?P,?O)、(?S,P,O)、(?S,?P,O)、(?S,P,?O)和(?S,?P,?O),其中变量在模式中以符号?开头。变量是SPARQL查询的输出,用作中间结果(例如,以处理连接),或者没有特殊用途但有助于形成查询。在一些示例中,SPARQL可以通过连接三元组模式的集合来构建更复杂的查询(通常被称为基本图模式或BGP)。

[0119] 在实施方式的示例中,查询引擎可以使用上述八种模式作为基本构建块。RDF图可以表示为邻接矩阵的集合,即每个谓语一个邻接矩阵。然后,对三重模式的回答可以被视为一个或多个邻接矩阵上的模式匹配,其中每个邻接矩阵代表一个谓语。给定“默认属性”规则定义为:

[0120] $\text{DefaultAttr}(A,v)$

[0121] $< -\text{SubProperty}(A, \text{er:Attribute}), \text{DefaultValue}(A,v)$.

[0122] $A(i,v) < -\text{DefaultAttr}(A,v), \text{Domain}(A,p), \text{Type}(i,P)$

[0123] 有以下谓语:DefaultAttr、SubProperty、DefaultValue、A、Domain和Type,其中Domain代表rdf:domain,Type代表rdf:type,SubProperty代表rdfs:subProperty。区分查询谓语,即感兴趣的输出,是规则中的谓语A。在上文讨论的示例A中,区分查询谓语可以在RDF中被翻译为ex:attr10。

[0124] 因此,实施方式的目标的示例是能够高效地回答以下模式:

[0125] ex:instance1 ex:attr10?v由于“默认属性”规则,输出“attr10DefaultValue”(参见示例A)。模式ex:instance1 ex:attr10?v是应用于邻接矩阵P=ex:attr10的模式(S,P,?O),主语S设置(即绑定)为ex:instance1。

[0126] 谓语ex:attr10可以具有基础三元组(即非推断的三元组)。根据实施方式的示例,谓语的基础三元组可以形成自己的邻接矩阵,即所谓的“原始数据矩阵”(或R矩阵)。与SQL数据库相比,R矩阵是扩展数据库谓语(EDB)的一部分,即数据库的源表。

[0127] SQL数据库需要规则结果的实体化视图。当(SQL数据库的)EDB改变时,此类规则的结果被更新。实施方式的示例将“蕴含数据矩阵”(或E矩阵)定义为谓语ex:attr10的邻接矩阵,该谓语由应用“默认属性”规则产生的所有三元组组成。实施方式的示例使用前向链接技术来构建E矩阵。众所周知的前向链接技术将模型理论中的规则详尽地应用于(数据库的)断言事实,以便生成所述事实的RDF闭包(演绎闭包)。

[0128] 将这种前向链接技术应用于示例“默认属性”规则,如下所示:

[0129] $\text{DefaultAttr}(A,v)$

[0130] $< -\text{SubProperty}(A, \text{er:Attribute}), \text{DefaultValue}(A,v)$,

[0131] $A(i,v) < -\text{DefaultAttr}(A,v), \text{Domain}(A,p), \text{SubClassOf}(i,P)$,

[0132] 意味着实施方式的示例执行以下SPARQL查询,该查询是上述规则的翻译:

[0133] `select ?i ?attr ?v`

[0134] `where {`

[0135] `?attr er:DefaultValue ?v.`

[0136] `?attr rdfs:subPropertyOf er:Attribute.`

[0137] `?attr rdfs:domain ?p.`

[0138] `?irdfs:subClassOf ?p.`

[0139] `}`

[0140] 并在每个谓语?p的E矩阵中实体化SPARQL查询的结果(具体而言使用?p==ex:attr10)。上文讨论的查询成本太高,无法用于查询重写。

[0141] 当修改上述查询的谓语,即er:DefaultValue,rdfs:subPropertyOf,rdfs:domain和rdfs:subClassOf,它们在SQL中是EDB,实施例的示例相应地将E-Matrix更新为SQL中的实体化视图。实施例的示例通过所谓的“更新处理器”来执行此类更新。

[0142] R矩阵、E矩阵和更新处理器一起形成虚拟邻接矩阵,该矩阵被配置为根据定义的规则来回答SPARQL的八个三元组模式。图4中示出了虚拟邻接矩阵的示意图,其中虚拟邻接矩阵410包括R矩阵420、E矩阵430和更新处理器440以便回答SPARQL 450的八个三元组模式。

[0143] 实施方式的示例利用虚拟邻接矩阵,就好像所有三元组都是通过上文讨论的自下而上方法在半朴素方法中导出的。换言之,采用上文讨论的示例数据,查询:

```
select ? attr ? value where {  
  ? attr rdfs:subPropertyOf er:Attribute.  
[0144]   ex:instance1 ? attr ? value.  
}
```

[0145] 返回以下结果:

[0146] `<http://www.example.org#attr10>"attr10DefaultValue",并且`

[0147] `<http://www.example.org#attr11>"specificValue4".`

[0148] 第一个结果来自ex:attr10的E矩阵,实施方式的示例中的SPARQL查询引擎只询问谓语ex:attr10的虚拟邻接矩阵上的模式ex:instance1?attr?value。

[0149] 因此,实施方式的示例中的SPARQL查询引擎能够使用虚拟邻接矩阵作为标准邻接矩阵来回答查询,而无需知道“默认属性”规则,因此不会增加复杂度。因此,除了众所周知的RDFS规则之外,实施方式的示例还能够使用自定义逻辑规则来推断知识。

[0150] 实施方式的示例是有利的,因为推理由数据结构管理,独立于查询引擎,从而避免增加查询引擎的复杂度并简化E矩阵的更新。换句话说,为了更新E矩阵,不需要经过查询引擎层。

[0151] RDFS规则的确涉及递归,但是,上文讨论的“默认属性”规则不涉及递归。将本发明限制为线性递归规则(即,其中递归谓语在规则主体中仅出现一次)避免了在实施方式的示例中更新E矩阵的高昂成本。

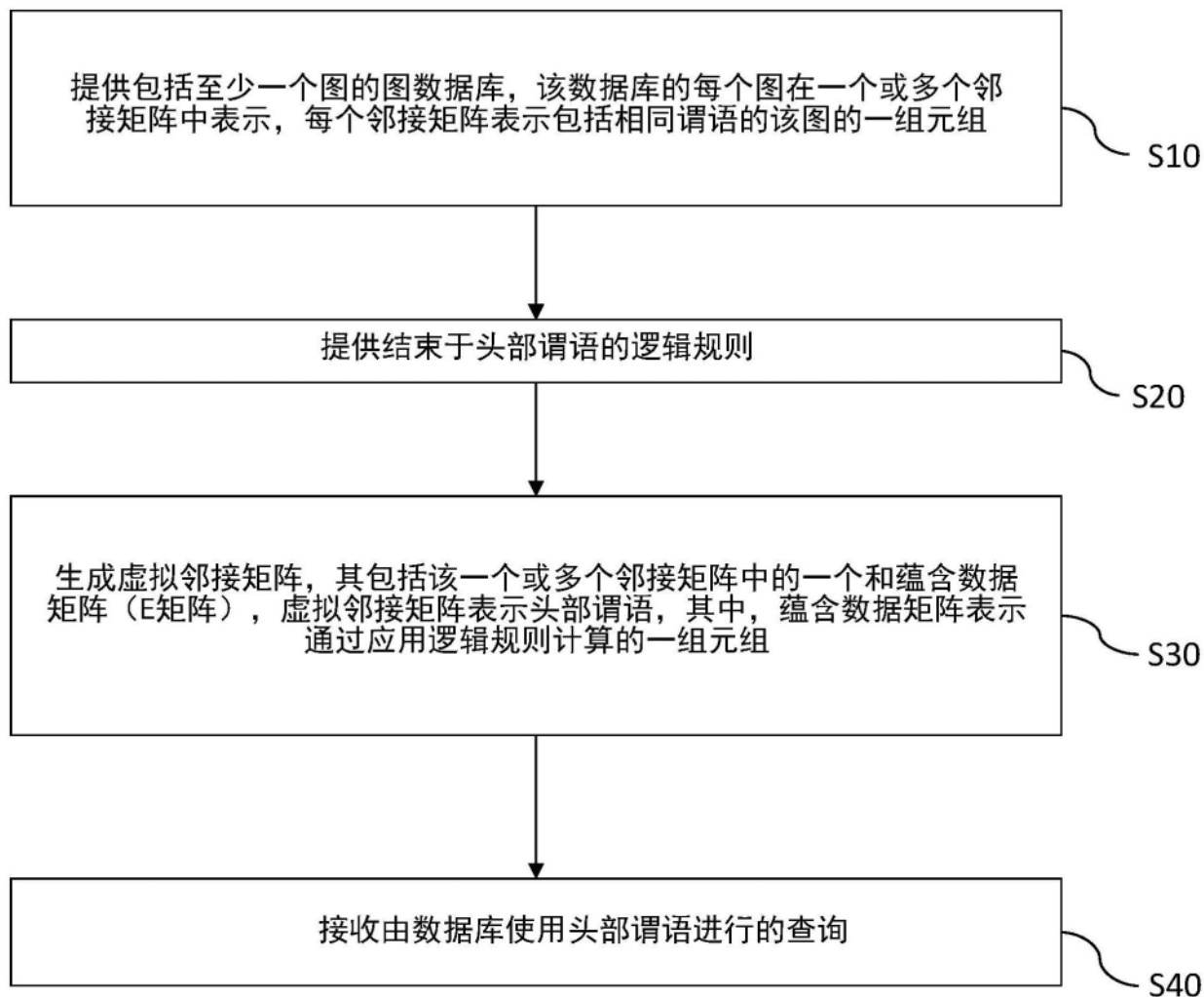


图1

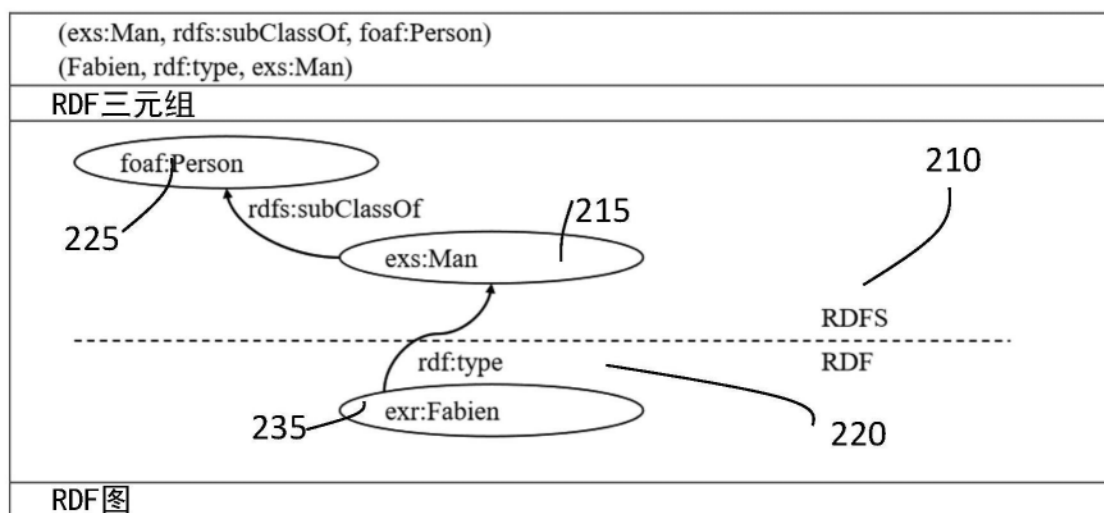


图2

RDFS蕴含模式		
	如果S包含:	则S RDFS蕴含识别D:
<i>rdfs9</i>	xxx rdfs:subClassOf yyy . zzz rdf:type xxx .	zzz rdf:type yyy .

图3

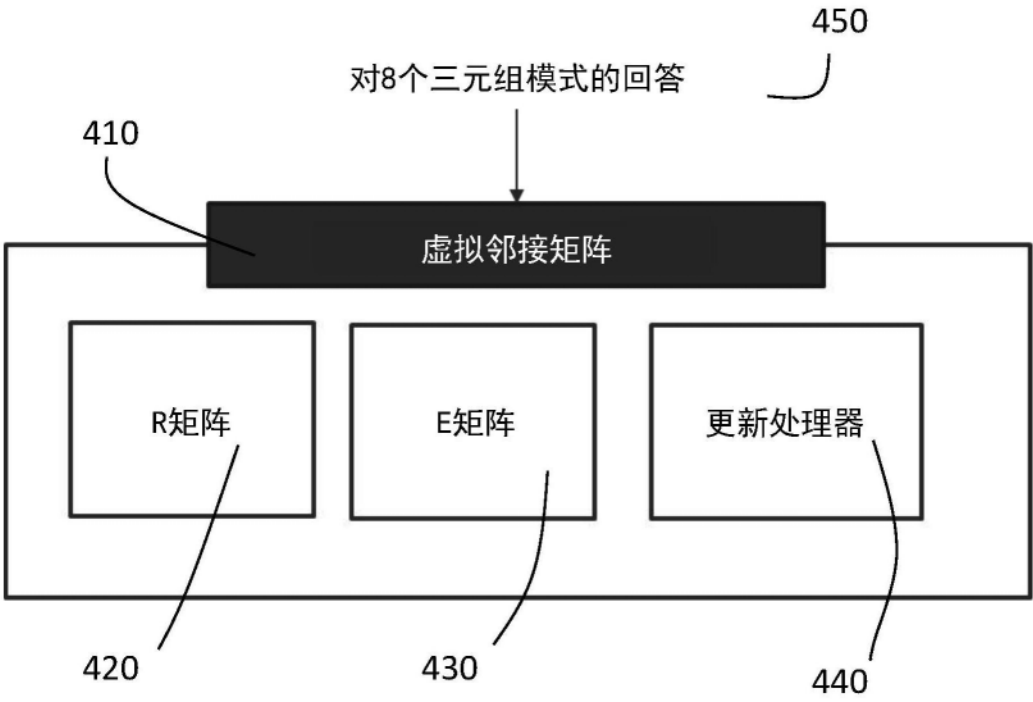


图4

初始化

$$\begin{aligned}\text{reachable}^{[0]} &:= \emptyset \\ \delta(\text{reachable})^{[0]} &:= \{(a, b), (b, c), (c, d), (c, c)\}\end{aligned}$$

迭代1:

$$\begin{aligned}\text{reachable}^{[1]} &:= \{(a, b), (b, c), (c, d), (c, c)\} \\ \Delta(\text{reachable})^{[1]} &:= \{(a, c), (b, d), (b, c), (c, c)\} \\ \delta(\text{reachable})^{[1]} &:= \{(a, c), (b, d)\}\end{aligned}$$

迭代2:

$$\begin{aligned}\text{reachable}^{[2]} &:= \{(a, b), (b, c), (c, d), (c, c), (a, c), (b, d)\} \\ \Delta(\text{reachable})^{[2]} &:= \{(a, d)\} \\ \delta(\text{reachable})^{[2]} &:= \{(a, d)\}\end{aligned}$$

迭代3:

$$\begin{aligned}\text{reachable}^{[3]} &:= \{(a, b), (b, c), (c, d), (c, c), (a, c), (b, d), (a, d)\} \\ \Delta(\text{reachable})^{[3]} &:= \emptyset \\ \delta(\text{reachable})^{[3]} &:= \emptyset\end{aligned}$$

图5

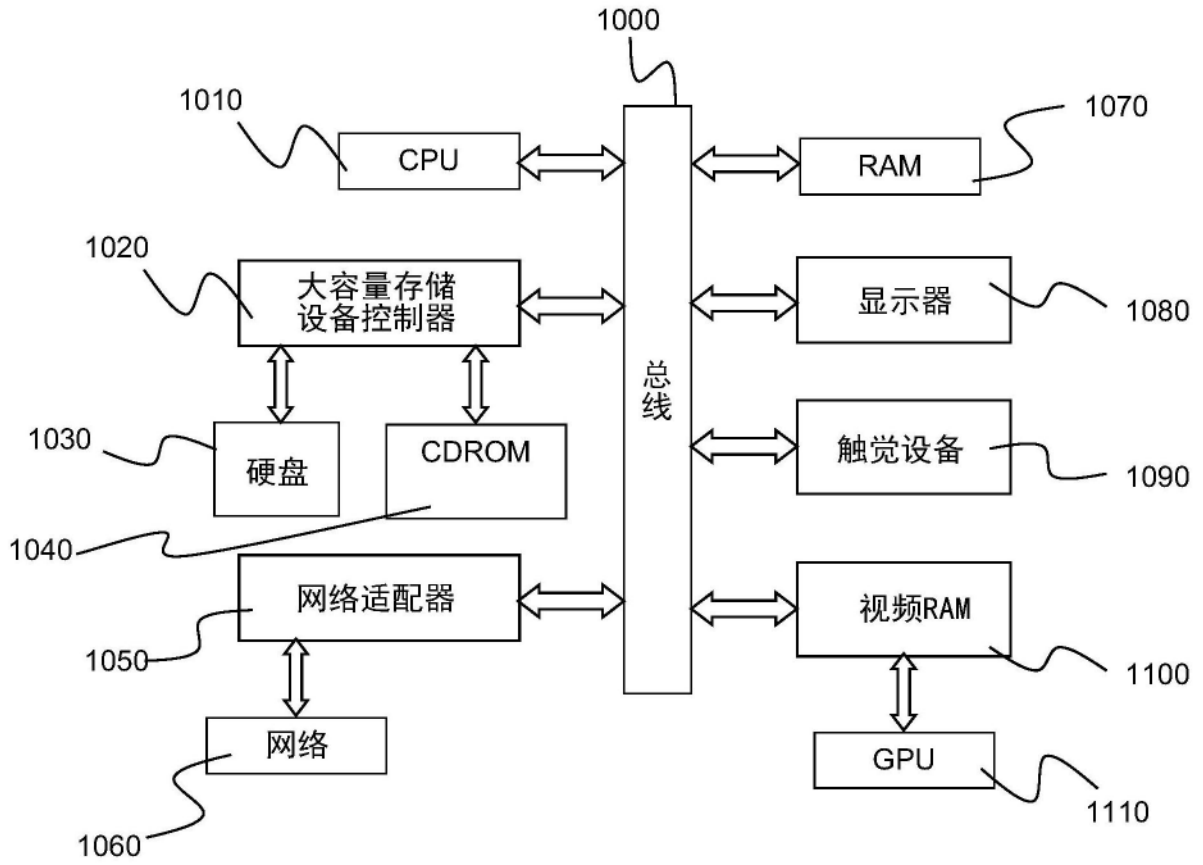


图6