(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 103886376 B
(45)授权公告日 2017.08.25

(21)申请号 201410117632.6
(22)申请日 2008.11.07
(65)同一申请的已公布的文献号
申请公布号 CN 103886376 A
(43)申请公布日 2014.06.25
(30)优先权数据
60/986,835 2007.11.09 US
12/266,353 2008.11.06 US
12/266,362 2008.11.06 US
(62)分案原申请数据
200880114945.9 2008.11.07
(73)专利权人 万特里克斯公司
地址 加拿大魁北克
(72)发明人 R·E·诺顿
L-R·普瓦里耶-博舍曼 R·埃鲁
J·L·C·M·拉瓦利埃
(74)专利代理机构 北京市中咨律师事务所 11247
代理人 宛丽宏 杨晓光
(51)Int.Cl.
G06N 5/02(2006.01)
(56)对比文件
US 2003031260 A1,2003.02.13,
US 2005108176 A1,2005.05.19,
CN 1294712 A,2001.05.09,
WO 0144933 A2,2001.06.21,
审查员 安飞

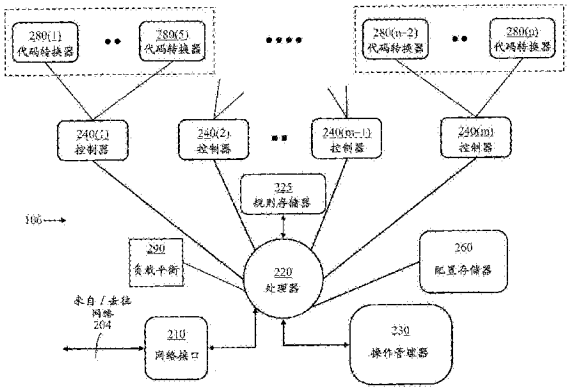
权利要求书2页 说明书25页 附图28页

(54)发明名称

用于基于规则的内容过滤的系统和方法

(57)摘要

公开了一种用于在支持多用途通信服务的网络中的数据容器的内容过滤的系统和方法。该内容过滤系统与内容适配系统集成。若干个服务器单元根据不同的协议来处理在源处规划的5个容器。内容过滤过程取决于表征容器的每个组件以产生内容描述符的集合以及根据每个描述符的预设准则来产生每个组件的二元条件的集合。规则的集合被设计,其中每个规则将各自的布尔表达应用于条件的子集以规定各自的内容编辑动作。公开了指定和估计规则的布尔表达的方法。使用形式图定义规则的相互依赖性。



CN 103886376 B

1.一种用于编辑多媒体容器的内容的引擎,所述引擎包括:

存储设备,用于存储:

布尔变量的阵列,每个布尔变量是根据各自的运算符来确定的,所述各自的运算符可应用于表征多媒体容器内容的第一运算数和指定所述第一运算数的目标值的第二运算数;以及

编码规则的阵列,每个规则指定各自的内容编辑动作,所述各自的内容编辑动作由所述布尔变量各自的子集的各自的布尔表达的值激活;

多个服务器单元,每一服务器单元托管控制器和各自的代码转换器子集,每一代码转换器用于将所述编码规则中的至少一个应用于所述多媒体容器的子集,所述服务器单元包括具有多个核心的处理器,每个核心排他性地分配到所述代码转换器子集中的代码转换器;以及

网络接口,用于从客户端接收所述多媒体容器并且将每个多媒体容器导向用于各自的服务器单元。

2.根据权利要求1所述的引擎,进一步包括:

操作管理器,其包括存储在用于执行操作、管理和维护功能的计算机可读存储介质中的计算机可读指令;以及

图形用户接口,包括存储在计算机可读存储介质中的计算机可读指令,耦合到所述操作管理器以用于输入所述编码规则的阵列。

3.根据权利要求1所述的引擎,进一步包括:代码转换器负载平衡模块,其耦合到所述控制器,用于向所述各自的代码转换器子集中的代码转换器公平地分配编辑请求。

4.根据权利要求1至3中任意一个所述的引擎,进一步包括:

代码转换器服务模块,耦合到所述各自的代码转换器子集中的每个代码转换器,用于从外部源接收新程序;

程序存储介质,用于存储可接入所述每个代码转换器的多个程序;以及

程序寄存器,包括存储在计算机可读存储介质中的计算机可读指令,用于标识每个新程序的功能以及根据各自的功能组织程序。

5.根据权利要求4所述的引擎,其中所述代码转换器服务模块进一步包括存储在计算机可读存储介质中的计算机可读指令,当执行所述计算机可读指令时,引起所述处理器的核心:

接收特定多媒体容器和编辑请求;以及

选择和执行与所述编辑请求有关的程序。

6.根据权利要求4所述的引擎,其中所述程序寄存器包括计算机可读指令,当执行所述计算机可读指令时,引起所述处理器的核心:

根据各自的功能来组织程序;以及

用各自的新程序来取代现有程序。

7.根据权利要求4所述的引擎,其中所述程序存储介质包括:

存储在所述计算机可读存储介质中的驻留程序;以及

存储在所述计算机可读存储介质中的动态加载的外部程序。

8.一种编辑多媒体容器的方法,包括:

规划所述多媒体容器的描述符的集合；

指定与所述描述符相对应的准则的集合；

指定运算符的集合；

确定布尔变量的集合，每个布尔变量是根据从所述描述符的集合选择的描述符和从所述准则的集合选择的准则来定义的；

定义规则的集合，每个规则执行所述布尔变量的子集的布尔表达；

将多个代码转换器布置为代码转换器组，每个代码转换器组耦合到控制器集合中的各自的控制器，每个控制器包括多核处理器并且访问所述规则的集合中的各自的子集；

接收多媒体容器；

在耦合到选择的控制器的代码转换器处估计至少一个可应用于所述多媒体容器的内容的规则；以及

根据估计的结果编辑所述多媒体容器。

9.根据权利要求8所述的方法，其中所述描述符的集合包括以下中的至少一个：

所述内容的大小；

来自预定义的内容族集合当中的内容族；

编码方法；以及

优先级指定。

10.根据权利要求8所述的方法，其中所述运算符的集合包括以下中的一个或多个：

比较运算符；

逻辑运算符；

集合运算符；以及

用户定义的运算符。

11.根据权利要求8所述的方法，进一步包括：通过图形用户接口来获取所述描述符的集合、所述准则的集合、所述运算符的集合以及所述布尔表达的集合，所述图形用户接口包括存储在计算设备的存储器中的计算机可读指令。

12.根据权利要求8所述的方法，其中所述多媒体容器包括多个组件，并且所述内容对应于所述组件中的一个。

13.根据权利要求8所述的方法，进一步包括：基于来自所述布尔变量的集合中的单个布尔变量执行进一步的编辑功能。

14.根据权利要求8至13中的任意一个所述的方法，进一步包括：以由于执行布尔表达而产生的编辑功能有条件地排除至少一个后续布尔表达的执行所根据的顺序，来布置所述布尔表达。

15.根据权利要求13所述的方法，其中执行所述编辑功能的步骤包括以下中的一个：

扫描所述内容以检测恶意插入，并且移除检测到的恶意插入；以及

扫描所述内容以检测恶意插入，并且一旦检测到恶意插入，则删除所述内容。

用于基于规则的内容过滤的系统和方法

[0001] 本申请是于2008年11月7日提交的、题为“用于基于规则的内容过滤的系统和方法”的中国专利申请200880114945.9的分案申请。

[0002] 相关申请的交叉引用

[0003] 本申请要求2007年11月09日提交的序列号为60/986,835的NORTON等人的标题为“A Method and System for Ruled-Based Content Filtering”的美国临时申请、2008年11月06日提交的序列号为12/266,353的NORTON等人的标题为“System and Method for Rule Based Content Filtering”的美国专利申请以及2008年11月06日提交的序列号为12/266,362的NORTON等人的标题为“An Engine for Rule Based Content Filtering”的美国专利申请的权益,所有的申请通过引用的方式合并于此。

技术领域

[0004] 本发明涉及多媒体消息服务,并且更具体地涉及用于内容过滤的方法和系统。

背景技术

[0005] 新兴多用途电信服务使得通信设备能够交换每个具有多个不同类型的组件的数据容器。例如,组件可以包括表示文本、图像、音频信号或视频信号的数据。用于处理这样的服务的复杂终端设备正在快速演进,导致通过相对短时间分隔的多“代”终端设备。

[0006] 由于多代终端设备的共存,所以出现了非兼容性问题。传送数据容器的终端设备通常不知道接收终端设备的特点和能力(或在多播通信的情况下的接收终端设备)。这要求提供用于确保正确地完整检测数据容器的内容或者正确地检测其特定组件以及通知接收方终端设备对原始容器做出的任何改变的设施,如在2008年9月25日提交的序列号为12/238,390的共同待决的美国申请,通过引用的方式将其内容合并于此。

[0007] 除了兼容性问题之外,传送每个可选地具有各自的附件的多个组件进一步增加了遭受诸如病毒的恶意的插入。容器遭受侵入需要提供一种内容过滤设施。

[0008] 因此,需要一种用于在提供多用途服务的网络中提供除了内容适配之外的内容过滤的设施。

发明内容

[0009] 本发明的一个目的是在提供多媒体服务的网络中提供内容过滤设施。另一目的是将内容过滤设施与现有内容适配设施合并,以便实现全面、有效和经济的系统。

[0010] 根据本发明的另一方面,提供了一种用于编辑容器的内容的引擎,所述引擎包括:

[0011] 网络接口,用于从客户端接收所述容器并且将每个容器导向用于标识和解析所述容器的多个控制器当中的各自的控制器;

[0012] 存储设备,用于:

[0013] 将布尔变量的阵列存储到所述容器的子集,根据可应用于表征所述内容的第一运算数和指定所述第一运算数的目标值的第二运算数来确定每个所述布尔变量;

[0014] 所述引擎进一步包括：

[0015] 操作管理器，包括存储在用于执行操作、管理和维护功能的计算机可读存储介质中的计算机可读指令；

[0016] 图形用户接口，包括存储在计算机可读存储介质中的计算机可读指令，耦合到用于输入编码规则的所述阵列和用于向每个控制器分配所述代码转换器的各自的子集的所述操作管理器；以及

[0017] 配置存储设备，用于存储分配到每个控制器的代码转换器的标识符。

[0018] 引擎进一步包括分类模块，包括存储在计算机可读存储介质中的计算机可读指令，耦合到所述网络接口，用于将容器分类成容器类型，每个容器类型对应于规划容器所根据的协议，并且将特定类型的容器从所述多个容器当中导向特定控制器。

[0019] 引擎进一步包括控制器负载平衡模块，包括存储在计算机可读存储介质中的计算机可读指令，耦合到所述操作管理器，所述控制器负载平衡模块包括用于根据所述容器的时变流率将代码转换器动态地分配到控制器的部件。

[0020] 引擎进一步包括代码转换器负载平衡模块，耦合到所述每个控制器，用于在分配到所述每个控制器的代码转换器当中公平地分配容器编辑请求。

[0021] 引擎进一步包括刀片服务器，其托管所述每个控制器以及所述代码转换器的所述各自的子集，所述刀片服务器包括：

[0022] 处理器，其具有多个核心，每个核心排他性地分配到所述代码转换器的子集的代码转换器；

[0023] 多个存储设备；

[0024] 输入接口；以及

[0025] 输出接口。

[0026] 引擎进一步包括：

[0027] 代码转换器服务模块，耦合到所述多个代码转换器的每个代码转换器，用于从外部源接收新程序；

[0028] 程序存储介质，用于存储可接入所述每个代码转换器的多个程序；以及

[0029] 程序寄存器，包括存储在计算机可读存储介质中的计算机可读指令，用于标识每个新程序的功能以及根据各自的功能组织程序。

[0030] 在上述引擎中，代码转换器服务模块进一步包括存储在计算机可读存储介质中的计算机可读指令，当执行所述计算机可读指令时，引起所述处理器的核心：

[0031] 从所述多个控制器中的特定控制器接收特定容器和编辑请求；

[0032] 选择和执行与所述编辑请求有关的程序；以及

[0033] 将结果返回所述特定控制器。

[0034] 所述程序寄存器包括计算机可读指令，当执行所述计算机可读指令时，引起所述处理器的核心：

[0035] 根据各自的功能来组织程序；以及

[0036] 用各自的新程序来取代现有程序。

[0037] 在上述引擎中，所述程序存储介质包括：

[0038] 存储在计算机可读存储介质中的驻留程序；以及

[0039] 存储在计算机可读存储介质中的动态加载的外部程序。

[0040] 根据本发明的另一方面，提供了一种过滤具有多个组件的数据容器的方法，包括：

[0041] 选择组件；

[0042] 确定表征所述组件的内容的多个二元条件；

[0043] 规划布尔表达的集合，每个布尔表达包括布尔运算符和从所述二元条件的集合中选择的至少两个运算数；

[0044] 执行所述布尔表达的集合中的每个布尔表达，以确定内容指示符的集合，所述指示符与所述布尔表达具有一对一的对应关系；以及

[0045] 执行与所述指示符中的每个相对应的所述内容有关的编辑功能。

[0046] 所述方法进一步包括基于单个二元条件执行进一步的编辑功能。

[0047] 所述方法进一步包括以由于执行布尔表达而产生的指示符有条件地排除至少一个后续布尔表达的执行所根据的顺序来布置所述布尔表达。

[0048] 如上所述的方法进一步包括根据以下来确定所述布尔表达：

[0049] 所述内容的指定描述符；以及

[0050] 预设与所述描述符相对应的准则。

[0051] 在上述方法中，执行所述编辑功能的步骤包括以下中的一个：

[0052] 扫描所述内容以检测恶意插入，并且移除检测到的恶意插入；以及

[0053] 扫描所述内容以检测恶意插入，并且一旦检测到恶意插入，则删除所述内容。

[0054] 根据本发明的又一方面，提供了一种使用计算设备来编辑内容的方法，包括：

[0055] 规划所述内容的描述符的集合；

[0056] 指定与所述描述符相对应的准则的集合；

[0057] 执行运算符的集合；

[0058] 定义布尔变量的集合，每个布尔变量是将来自所述运算符的集合当中的运算符应用到第一运算数和第二运算数的结果，其中所述第一运算数是从所述描述符的集合当中选择的描述符，并且所述第二运算数是从所述准则的集合当中选择的准则；以及

[0059] 定义规则的集合，每个规则执行所述布尔变量的子集的从布尔表达的集合当中选择的布尔表达，并且根据所述执行的结果来执行从与所述内容相关的动作的集合当中选择的动作。

[0060] 在上述方法中，描述符的集合包括以下中的至少一个：所述内容的大小；来自预定义的内容族当中的内容族；编码方法；以及优先级指定。

[0061] 在上述方法中，所述运算符的集合包括以下中的一个或多个：一元运算符；二元运算符；算术运算符；比较运算符；逻辑运算符；集合运算符；以及用户定义的运算符。

[0062] 所述方法进一步包括通过包括存储在所述计算设备的存储器中的计算机可读指令的图形用户接口来输入所述描述符的集合、所述准则的集合、所述运算符的集合以及所述布尔表达的集合。

[0063] 方便地，所述方法进一步包括选择所述内容以包括所述数据容器的一部分。

[0064] 根据本发明的另一方面，提供了一种用于过滤多媒体数据容器的内容的系统，包括：

[0065] 网络接口，用于从网络接收所述多媒体数据容器；以及

[0066] 多个服务器单元，每个服务器单元包括处理器的集合和存储设备的集合，在其上安装了：

[0067] 过滤器定义模块，具有存储在计算机可读存储介质中的计算机可读指令，用于从用户获取过滤器的集合的定义，每个过滤器指定内容描述符的定义、描述符准则以及运算符；

[0068] 规则构造模块，具有存储在计算机可读存储介质中的计算机可读指令，用于从所述用户获取内容过滤规则的集合，每个规则指定所述过滤器的子集和过滤动作的布尔表达；

[0069] 用于表征多媒体数据容器的每个组件的内容，确定所述内容描述符，应用所述运算符并且确定所述每个过滤器的状态的模块；

[0070] 用于确定每个所述规则的二元输出的模块；以及

[0071] 用于执行与服从所述二元输出的预设值的所述内容相关的过滤动作的模块。

[0072] 所述系统进一步包括以下模块中的至少一个，每个模块包括存储在计算机可读存储介质中的计算机可读指令：

[0073] (1)用于配置服务器单元以接受根据从已知协议的集合中选择的任何协议形成的多媒体数据容器的模块；

[0074] (2)用于在同样配置的服务器单元当中公平地分发多媒体数据容器的模块；

[0075] (3)用于使得用户能够根据代数句法来提供布尔表达的模块；

[0076] (4)用于使得用户能够以树结构的形式来提供布尔表达的模块；

[0077] (5)用于使得用户能够通过编辑和修整树的图形来输入布尔表达的模块，该树的每个节点表示运算符和各自的运算数的集合；

[0078] (6)用于验证布尔表达的正确性的模块；

[0079] (7)用于以由规则执行的特定过滤动作排除至少一个后续规则的执行所根据的顺序来布置规则的模块；

[0080] (8)用于提示用户指定在布尔表达的值上的规则条件句中的至少一个的连续规则的模块；

[0081] (9)用于以形式图表的形式来呈现内容过滤的集合的模块；以及

[0082] (10)用于优化每个规则的布尔表达的模块。

[0083] 所述系统进一步包括用于执行以下操作的模块，包括存储在计算机可读存储介质中的计算机可读指令：选择特定规则，每个指定包括最多预设数目的过滤器的过滤器子集；对于特定规则中的每个，估计用于过滤器子集的所有的值的过滤器子集的布尔表达，以产生作为在过滤器子集中的过滤器的数目的2μ个比特的阵列，其中$\mu>1$；以及在存储设备中存储所述比特的阵列。

[0084] 根据另一方面，本发明提供了一种过滤数据容器的内容的方法。所述方法包括指定二元条件的集合，指定运算符的集合，形成叶向量以及形成节点向量。

[0085] 二元条件的集合表征内容。运算符中的一个被指定空后继者，而每个其它运算符被指定来自运算符的集合内的后继者。叶向量包括$N>1$个叶记录。每个叶记录具有来自运算符的集合当中的叶运算符以及二元条件的各自的子集。节点向量具有N个节点记录，每个具有节点运算符字段和节点状态字段。

[0086] 每个叶运算符被应用于各自的二元条件，并且结果被放置在节点记录的节点状态字段中。然后，每个叶运算符的后继者被放置在节点记录的节点运算符字段中。

[0087] 在处理了每个叶记录之后，处理了节点向量。具有共同运算符的节点记录被标识并且被组合的记录取代。共同运算符被应用于所标识的节点记录的节点状态字段的条目，并且结果状态被放置在所组合的记录的节点状态字段中。共同运算符的后继者被放置在所组合的记录的节点运算符字段中。

[0088] 内容索引被确定为应用服从共同运算符的后继者是空后继者的条件的共同运算符的结果状态。替代地，所述方法可以在组合了共同运算符的节点记录之后保持跟踪节点向量的节点记录的数目，并且内容索引被确定为与等于一的节点记录的数目相对应的结果状态。所述方法进一步包括根据内容索引的值来执行指定编辑动作的步骤。

[0089] 根据进一步的方面，本发明提供了一种基于预定决定向量来过滤数据容器的内容的方法。所述方法包括定义二元条件的集合，其中每个二元条件是选择的内容的描述符以及描述符的各自准则；定义布尔表达的集合以及相对应的内容过滤动作；选择表示为μ比特，$\mu>1$的字符串的指定的μ个二元条件的布尔表达；以及对于该字符串的2^{μ}个值中的每个估计布尔表达，以产生2^{μ}个条目的决定向量，每个条目是与该字符串的2^{μ}个值中的一个相对应的内容度量的状态。

[0090] 一旦接收到数据容器，根据数据容器的内容来确定所指定的μ个二元条件的值。然后使用μ比特的结果字符串的值来对决定向量编索引，以获取确定是否需要应用内容过滤动作的布尔表达的值。

[0091] 布尔表达可以以具有布尔运算符、运算数和分隔符的代数格式来获取。然后，通过检查布尔表达来估计布尔表达以标识简单模式，该简单模式封闭在两个分隔符之间的布尔运算符和两个运算数。只要发现了简单模式，就将布尔运算符以及两个分隔符应用到两个运算数以确定模式的二元值以及简单模式，用二元值来取代布尔运算符。检查布尔表达以检测简单模式的过程被重复，直到布尔表达被减少到确定是否应用编辑动作的单个二元值（“真”或“假”）。

[0092] 替代地，布尔表达可以以具有多个节点的树结构的形式来获取。然后，接着估计节点来估计布尔表达。具有多个记录的树模板被创建。每个记录对应于各自的节点，并且具有用于第一运算数、第二运算数、当前运算符和后继者记录的四个字段。从第一记录开始并且后续继续进行到最后的记录，当前记录的运算数被应用于从字符串的当前值确定的各自的二元值，以产生新的二元值。如果当前记录是最后的记录，则新的二元值是布尔表达的值。如果当前记录是中间的记录，则新的二元值被放置在后继者记录的运算数字段中。

附图说明

[0093] 现在将参考附图举例描述本发明的实施例，在附图中：

[0094] 图1图示了根据本发明的实施例的用于过滤和适配通过网络传送的数据容器的网络支持服务控制器和编辑引擎；

[0095] 图2图示了根据本发明的实施例的包括控制器和代码转换器的编辑引擎；

[0096] 图3图示了根据本发明的实施例的多媒体容器、容器组件的内容描述符和内容过滤器；

[0097] 图4图示了根据本发明的实施例的与可应用于数据容器的规则的集合相关联的布尔表达的代数形式；

[0098] 图5图示了根据本发明的实施例的内容过滤过程的基本组件；

[0099] 图6图示了根据本发明的实施例的导出容器内容的描述符的过程；

[0100] 图7图示了根据本发明的实施例的用于内容过滤的系统；

[0101] 图8图示了根据本发明的实施例的与图7的系统类似的适于具有多个组件的容器的系统；

[0102] 图9详述了根据本发明的实施例的内容过滤过程；

[0103] 图10图示了根据本发明的实施例的估计规则的布尔表达的方法；

[0104] 图11图示了根据本发明的实施例的用于编码过滤规则的第一示例性规则树结构；

[0105] 图12图示了用于编码图11的规则树结构的第一数据结构；

[0106] 图13图示了根据本发明的实施例的用于编码过滤规则的第二示例性规则树结构；

[0107] 图14图示了应用于图13的第二示例性规则树的图12的第一数据结构；

[0108] 图15图示了根据本发明的实施例的应用图12和图14例示的第一数据结构的过程；

[0109] 图16图示了根据本发明的实施例的用于编码规则树的第二数据结构；

[0110] 图17图示了根据本发明的实施例的应用图16中例示的第二数据结构的过程；

[0111] 图18图示了根据本发明的实施例的使用规则树来确定规则结果的过程；

[0112] 图19详述了根据本发明的实施例的形成与图18的过程相关联的叶向量的步骤；

[0113] 图20图示了根据本发明的实施例的预处理用于快速运行时间规则执行的布尔表达的方法；

[0114] 图21图示了图20的方法的示例性实现；

[0115] 图22图示了根据本发明的实施例的与过滤器定义和规则定义相关的数据的处理；

[0116] 图23图示了顺序过滤多媒体容器的组件的过程；

[0117] 图24图示了根据本发明的实施例的应用用于内容过滤的多个规则的过程；

[0118] 图25图示了根据本发明的实施例的图示确定内容过滤动作的规则的互相依赖性的图；

[0119] 图26图示了根据本发明的实施例的图2的引擎的操作管理器的模块；

[0120] 图27图示了根据本发明的实施例的图2的代码转换器的模块；以及

[0121] 图28图示了根据本发明的实施例的图2的引擎的代码转换器的模块。

具体实施方式

[0122] 术语

[0123] 多媒体服务(MMS)：该术语用于通俗地表示多内容通信服务，其中通过网络在终端当中交换诸如文本、音频信号、视频信号、图像、呈现等的不同形式的信息内容。从一个终端传输到另一终端的编码信息通常被布置在单个数据流中，并且时间交织段对应于不同的信息内容。

[0124] 容器：容器是存储在计算机可读介质上并且通过计算机网络传送的计算机文件。容器被构造成包含各种类型的数据。容器可以支持具有同步信息的多个文本、音频和视频流以使得能够协调回放各种流。

[0125] 容器组件：容器包括分段，每个分段包括以特定形式编码的数据，诸如文本、音频数据、图像数据或视频数据。术语“容器组件”指的是在一个分段中的数据。容器组件可以被简称为“组件”。在多媒体消息系统中，组件也被称为“媒体”。

[0126] 容器筛选：“容器筛选”指的是检查容器的内容的过程，包括所有组件，以确保没有任何不想要的插入，尤其是有害的插入。

[0127] 容器适配：“容器适配”指的是修改发现的容器组件的形式以与各自的接收器的解码能力兼容的过程。如果呈现容器组件以适合接收器不可行，则可以删除容器组件。容器适配过程是接收器特定的，而容器筛选的过程独立于希望接收器的类型。

[0128] 容器编辑：术语“容器编辑”指的是容器筛选和容器适配的组合过程。

[0129] 容器调节：该术语可以与“容器编辑”同义地使用。然而，即使当没有修改容器时，容器调节也向容器附加了适当的通知。

[0130] 代码转换器：代码转换器是执行编码信息的直接数字到数字转换的设备，以使得能够以适合特定接收器的不同格式来再现一个格式的信息记录。

[0131] 图1图示了提供从传送设备120到接收设备160的路径的网络140，在下文中称为传送器120和接收器160。除了许多不同类型的其它硬件终端设备，网络140支持服务控制器103和编辑引擎106。传送器120发送容器到接收器160，其中容器可以包括诸如编码文本、音频信号、静止图像、动画(图像的快速显示)以及视频信号的不同内容类型的数据。容器可以被导向服务控制器103，其进而将容器导向用于检查该容器的编辑引擎106以及在需要时编辑该容器的内容。编辑过程包括数据筛选以确保没有任何不想要的插入，尤其是有害插入，满足具体要求的内容修改，以及与各自接收器的解码能力兼容的内容适配。

[0132] 图2图示了编辑引擎106。网络接口210通过链路204从客户端接收容器。容器被导向到分别标识为240(1)，240(2)，…，240(m)的m>1个控制器240中的一个。控制器204可以被实现为不同的硬件实体或将如下所述的共享主管多个代码转换器280的计算设备。控制器240是协议特定的，每个被编程以处理根据相应协议形成的容器。处理特定协议的控制器被认为具有相同控制器类型。控制器可以分组成控制器组，每个控制器组处理根据相同协议规划的容器。编辑引擎106可以具有不同类型的控制器。然而，整个编辑引擎106可以被配置成具有相同类型的控制器。编辑引擎106也可以具有负载平衡模块290。

[0133] 编辑引擎106包括多个代码转换器280，分别标识为280(1)、280(2)，…，280(n)。代码转换器280的主要功能是执行编码信息的直接数字到数字转换，以使得能够以适于特定接收器的不同格式来再现一个格式的信息记录。然而，代码转换器还可以执行内容过滤以及内容适配的过程。多个选择的代码转换器280被分配到每个控制器240，一起形成控制配件。例如，在图2中，控制器240(1)和代码转换器280(1)至280(5)形成了安装在各自的计算设备上的一个控制配件。控制器240(m)和代码转换器280(n-2)至280(n)形成了安装在另一计算设备上的另一控制配件。控制配件优选地安装在也被称为“刀片服务器”服务器单元上，它是支持处理器和存储设备的单个电路板。

[0134] 处理器220托管网络接口210，并且操作管理器230还被称为操作控制器。网络接口210从通信地耦合到网络140的客户端接收容器(图1)。操作管理器230包括存储在计算机可读存储介质中的计算机可读指令，用于执行操作、管理和维护功能。

[0135] 服务控制器103可以接收容器，并且向编辑引擎106中的一个发送容器编辑请求。

[0136] 处理器220还托管图形用户接口(未示出),它包括存储在计算机可读存储介质中的计算机可读指令,耦合到操作管理器230,用于输入编码的规则的阵列并且用于向每个控制器240分配代码转换器的各自的子集。配置存储设备260存储分配到每个控制器的代码转换器的标识符。

[0137] 控制内容过滤过程的规则可以被存储为存储在每个代码转换器280可访问的规则存储器225中的共同规则文件。替代地,托管控制器和关联代码转换器的每个计算设备(服务器单元)可以存储规则文件的相关子集。

[0138] 图3图示了具有分别标识为320(1)、320(2)等的多个组件320的容器。组件320可以包含文本、音频记录、编码图像、视频记录和任何其它内容类型。使用分别标识为332(1)、332(2)、…332(j)和332(D)的描述符332的集合330来表征组件的内容。D是描述符的总数目。根据诸如内容类型、标识符(名称)、扩展、数字签名、密码功能、优先级和文件大小的若干属性来定义内容描述符。因为内容过滤要求随着多媒体电信的演进技术而变化,所以可以添加或删除描述符。

[0139] 根据本发明的实施例,容器组件的内容的特性被表示为二元变量的集合,每个二元变量确定内容是否满足特定准则。通过检查容器的内容来确定接收到的容器的内容描述符的值。因此,执行该功能的代码转换器知道所接收到的容器的格式以及规划容器所根据的协议的暗示。通过将运算符344应用于两个运算数来确定表征二元变量的值;内容描述符(342)的值和相对应的准则346由内容过滤系统的安装者(用户)输入。运算符344和两个运算数342和346据称形成过滤器340(也被称为内容条件或简化条件)。因此,容器组件的内容的特征在于过滤器的集合,每个过滤器具有值“真”或“假”。

[0140] 在为适合接收器而适配之前编辑容器是基于规则的集合的,每个规则确定编辑动作,诸如删除整个内容、删除在内容中发现的恶意插入、或移除内容的附加。规则是过滤器的各自子集的函数。已经选择过滤器是二元变量,定义规则的函数优选被规划为过滤器子集的布尔表达。因此,用于内容过滤的系统的安装者(用户)(如将针对图7和图8更详细描述的)通过过滤器的子集、布尔表达以及根据执行布尔表达的结果而执行的动作定义了规则。

[0141] 图4图示了与可应用于数据容器的组件的存储在规则阵列420中的四个规则集合相关联的布尔表达的代数形式。相同的规则集合还可应用于规则过滤器的至少一个其它组件。

[0142] 通过一个过滤器(大小>30000)来定义第一规则,规则-1,其中内容描述符是组件的大小,运算符是“大于”并且准则是30000(附图标记440)。如果规则结果是布尔“真”,则各自的动作是放弃组件并且终止用于考虑中的组件的剩余规则的处理。

[0143] 通过两个过滤器(大小>5000)并且(族≠消息)的布尔表达450来定义第二规则,规则-2。第一过滤器的描述符是“大小”,准则是“5000”,并且运算符是“大于”。第二过滤器的描述符是“族”,准则是“消息”,并且运算符是“不等于”。布尔表达包含单个运算符“或”。与规则-2相关联的动作与规则-1的动作相同。

[0144] 通过三个过滤器:(大小>25000)、(内容-类型=图像/wbmp)并且(内容-类型=图像/png)的布尔表达460来定义第三规则,规则-3。第一过滤器的描述符是“大小”,准则是“25000”,并且运算符是“大于”。第二过滤器的描述符是“内容-类型”,准则是“图像/wbmp”,并且运算符是“等于”。第三过滤器的描述符是“内容-类型”,准则是“图像/png”,并且运算

符是“等于”。布尔表达包含两个运算符“与”和“或”。规则-3的动作与规则-1的动作相同。注意到，“wbmp”指的是无线位图(无线应用协议、WAP、图形格式)，并且“png”指的是“便携式网络图形”。

[0145] 通过一个过滤器(族=消息)来定义第四规则，规则-4，其中描述符的内容是“族”，准则是“消息”并且运算符是“等于”(附图标记470)。

[0146] 图5图示了合并在图2的操作管理器230中并且包括用户接口520、用于获取过滤器定义的过滤器创建模块530、用于获取规则定义的规则构造模块540、用于存储过滤器定义的存储分隔550以及用于存储规则定义(规则结构)的存储分隔560的数据获取子系统的基本组件。用户接口520使得安装者(用户)能够提供输入数据512，以定义过滤器的集合和规则的集合。过滤器创建模块530包括存储在计算机可读存储介质中的计算机可读指令，计算机可读指令当被执行时使处理器提示安装者输入内容描述符定义、用于每个定义的准则以及运算符。根据接收到的容器的内容来确定描述符的值。

[0147] 规则构造模块540包含存储在计算机可读存储介质中的计算机可读指令，当执行计算机可读指令时，计算机可读指令当被执行时使处理器提示安装者输入用于每个规则的布尔表达，并且从预定义的动作的集合中选择动作。计算机可读指令还使处理器解析布尔表达并且确定表达的执行方面的序列。

[0148] 过滤器定义被存储在存储设备的存储分隔550中，并且规则定义被存储在同一存储设备或另一其它存储设备的存储分隔560中。

[0149] 图6图示了使用存储在存储分隔550中的内容描述符的定义来确定接收到的容器612的内容描述符的值的过程。在步骤620中解析所接收到的容器612，以标识容器的组件。在步骤640中分析每个组件，并且在步骤650中将结果与从存储分隔550中读取的内容描述符定义配对。

[0150] 图7图示了根据本发明的实施例的用于内容过滤的系统的整体组织。存储器710存储用于所有相关过滤器的数据。通过来自由图5的模块530确定的运算符的集合740的运算符来定义每个过滤器，在图6的步骤650中确定来自内容描述符720的内容描述符，并且在图5的模块530中确定来自描述符的集合730的描述符准则。每个过滤器的二元值(“真”或“假”)被存储在存储设备750中，用于在执行图5的模块540中定义的规则的集合中使用。

[0151] 通过过滤器子集、布尔表达和动作的来定义每个规则。存储器770存储根据系统安装者(用户)输入在图5的模块540中确定的编码的布尔表达。存储器760存储在每个规则和各自的布尔表达中使用的过滤器的标识符。存储器780存储由于估计布尔表达的各自的结果而将执行的编辑动作的指示。每个布尔表达的执行产生二元结果以及各自的编辑动作。一旦完成了编辑动作，则将已编辑的内容放置在存储器790中。

[0152] 尽管图7图示了根据本发明的实施例的当应用于一个组件时用于内容过滤的系统，但是图8图示了当应用于k>1容器组件时用于图7的内容过滤的系统。存储设备810存储k个容器组件中的每个的所有相关过滤器的数据。与每个过滤器有关的数据分别被标识为812(1)至812(k)。容器的组件被顺序地处理。对于考虑中的组件，应用每个过滤器的运算符的结果被保持在存储器850中。N>1个编码的布尔表达的集合被存储在存储器864中。布尔表达被分别标识为870(1)至870(N)，每个与来自被分别标识为880(1)至880(N)的N个编辑动作的各自的编辑动作相关联。

[0153] 图9图示了内容过滤的示例性过程。用于图7或图8的内容过滤的系统的安装者(用户)已经初始定义了分别标识为920(1)至920(5)的五个内容描述符,分别标识为922(1)至922(8)的八个描述符准则、以及分别标识为924(1)至924(4)的四个运算符。安装者已经定义了分别标识为930(1)至930(12)的十二个过滤器,每个过滤器指定内容描述符920中的一个、准则922中的一个以及运算符924中的一个。如在图6的步骤650中描述的,一旦确定了五个内容描述符920(1)至920(5)的值,确定了十二个过滤器的二元值。

[0154] 安装者已经定义了分别标识为950(1)至950(6)的六个布尔表达,其中每个布尔表达与12个过滤器的子集相关联。例如,布尔表达950(2)与过滤器930(2)、930(8)和930(11)相关联。安装者定义了分别标识为960(1)至960(4)的四个动作。然后,安装者使用图5的规则构造模块540来定义分别标识为940(1)至940(9)的九个规则。每个规则与单个布尔表达950和单个动作960相关联。例如,规则940(1)指定布尔表达950(2)和动作960(2),而规则940(9)指定布尔表达950(5)和动作960(4)。

[0155] 规则可以基于一个过滤器,其中该规则的结果是过滤器的二元值。例如,规则940(7)仅取决于过滤器930(11)。

[0156] 布尔表达表示

[0157] 用于图7或图8的内容过滤的系统的安装者可以根据常规代数句法或根据树结构来提供布尔表达950。图5的用户接口520包括用于以代数形式编码布尔表达的第一模块(未示出)以及用于编码作为树结构呈现的布尔表达的第二模块(未示出)。两个模块中的每个提供了各自的模板,以使得安装者能够正确地指定布尔表达。

[0158] 布尔表达包括简单操作、复合操作和复杂操作。简单操作被展现为运算符和两个运算数,该运算符和运算数由两个分隔符(诸如两个括号)来定界。运算符和运算数可以以任何顺序列出,并且两个分隔符不需要彼此区分。两个运算符是表示两个过滤器的布尔变量。复合操作包括运算符和两个简单操作,该运算符和两个简单操作由两个分隔符来定界。复杂操作包括运算符和两个操作,所有都由两个分隔符来定界,其中两个操作中的任一个可以是简单操作或复合操作。构成复杂操作的两个操作还可以是复杂操作。简单操作、复合操作或复杂操作的分隔符可以相同。

[0159] 图10图示了根据本发明的实施例的估计布尔表达的方法,这要求仅识别和执行简单操作。根据该方法,解析编码的布尔表达以标识简单操作。标识的简单操作的运算符被应用于各自的运算数(过滤器)以产生“真”或“假”的二元值(例如,表示为“1”或“0”)。已标识的简单操作、因此处理的运算符、运算数和两个分隔符被删除并且被运算的结果取代。该过程递归地继续,直到已编码的布尔表达减少到单个简单操作,其结果变为布尔表达的结果。

[0160] 在图10的步骤1012中,检查布尔表达以标识简单操作。如果发现了简单操作(步骤1014),则步骤1016执行简单操作并产生二元值。步骤1018用二元值来取代简单操作的运算符、运算数和分隔符。然后,再次访问步骤1012以在减少的布尔结构中寻找另一简单操作。如果步骤1014确定在布尔表达的当前形式中没有发现进一步的简单操作,则步骤1020检查当前形式以确定它是否已经真正减少到单个二元值(“真”、“假”或“1”、“0”)。如果是,则步骤1022报告单个二元值作为执行布尔表达的结果。如果步骤1020确定已处理的布尔表达包含不止单个二元值,则步骤1024报告尚未正确地形成布尔表达的指示。

[0161] 图10的过程优选地在数据输入期间执行,使得用户(安装者)可以纠正布尔表达。

用户接口520或编辑引擎的某个其它组件可以被提供有计算机指令以分析错误地形成的布尔表达并分析错误。

[0162] 根据本发明的实施例的编码和估计布尔表达的替代方法以来布尔表达的图形树表示。图11中图示了示例性规则树1100，图11图示了标识为过滤器1至过滤器6的六个运算数(六个过滤器)的布尔表达，每个是树的叶。标记为θ_1、θ_2和θ_3的三个运算符定义了三个操作{过滤器1，θ_1，过滤器2}、{过滤器3，θ_2，过滤器4}和{过滤器4，θ_3，过滤器6}。每个运算符的中间后继者被定义。例如，θ_1、θ_2和θ_3的后继者分别是运算符θ_5、θ_4和θ_4，并且运算符θ_4和θ_5的后继者分别是θ_5和“空”。具有“空”后继者的运算符产生布尔表达的结果。

[0163] 运算符θ_1产生二元输出B1，它是运算符θ_5的运算数。运算符θ_2产生二元输出B2，它是运算符θ_4的运算数。运算符θ_3产生二元输出B3，它是运算符θ_4的另一运算数。运算符θ_4产生二元输出B4，它是运算符θ_5的另一运算数。运算符θ_5产生二元输出B*，它是由树表示的布尔表达的结果。

[0164] 图12图示了用于表示图11的规则树1100的模板阵列1230。模板阵列1230的索引1220如图12所示从0到19变化。模板阵列1230被划分成等于运算符总数目的多个记录(在图11的示例性树中为五个)，每个记录对应于运算符并且表示具有两个运算数的简单操作。在处理了考虑中的容器之后已知过滤器的二元值。因此，记录包括各自的过滤器、运算符定义和针对与中间后继运算符相对应的另一记录的指针的索引。“空”指针指示当前记录是要处理的最后记录。用户可以以任何顺序输入记录，并且在图5的用户接口520内的模块(未示出)重新组织记录，使得可以顺序地处理记录，并且当处理任何记录时，已经确定了各自的运算数的值。

[0165] 如图12所示，前三个记录对应于可应用于形成树的叶的六个过滤器的运算数θ_1、θ_2和θ_3。第一记录的指针π(1)指向保持运算符θ_1的二元结果B(1)的阵列的索引16。第二记录的指针π(2)指向保持运算符θ_2的二元结果B(2)的阵列的索引12。第三记录的指针π(3)指向保持运算符θ_3的二元结果B(3)的阵列的索引13。因此，当到达第四记录时，已经计算了两个运算数B(2)和B(3)。运算符θ_4的二元结果B(4)被写入位置π(4)=17。因此，当到达第五记录时，已经知道各自的两个运算数B(1)和B(4)。运算符θ_4的二元输出是布尔表达的结果，因为运算数θ_4没有后继者(即，空后继者)。

[0166] 图12中还图示了模板阵列1230的示例性激活。根据图6的过程确定的过滤器1至过滤器6的值分别是“真”、“假”、“真”、“假”“真”和“真”。布尔运算符θ_1至θ_5由用户分别指定为“与”、“或”、“与”、“与”、和“异或”。由于运算符θ_4(“异或”)具有空后继者，所以运算符“XOR”产生二元输出，该二元输出是布尔表达的结果。

[0167] 图13图示了与标记为L1至L11的十一个叶(过滤器)的布尔表达相对应的第二示例性规则树1300，并且图14图示了应用于图13的规则树的模板阵列1430，类似于图12的模板阵列1230，索引1420的范围从0至43。规则树1300包括标记为θ_2至θ_{11}的十个运算符。第一叶L1是运算符θ_{11}的运算数，它没有后继者。对于一致性，表示规则树1300的图14的模板阵列1430的第一记录(在图14中标记为记录1)概念上被查看，以包括运算数L1以及“不关注”运算数Φ和许可非存在运算符θ_1，它传送L1的值作为后继者运算符θ_{11}的运算数。如本领域所已知的，分配给运算数Φ的“不关注”值可以方便地是“真”状态或“假”状态。剩余的十个记录，模板阵列1430的记录2至记录11，对应于运算符θ_2至θ_{11}。在图14的模板阵列1430中每个

条目L1、L2至L11是过滤器的索引。如上所述，参照图12，与图5的用户接口相关联的输入组织模块522布置记录，使得可以顺序地处理记录，每个记录已经确定了运算数。

[0168] 图15总结了使用模板阵列1230(图12)或1430(图14)的树编码方法。在步骤1520中，创建了具有多个记录的模板，每个记录对应于树中的节点。每个记录包括含有两个过滤器的索引、当前运算符和指向与当前运算符的后继者相对应的后继者记录的指针的四个字段。在步骤1530中，根据图6的过程确定的过滤器的列表准备用于考虑中的容器。在步骤1540中，顺序地处理树模板的记录。每个记录的运算数通过对过滤器的列表编索引来获取。各自的运算符被应用于运算数，并且二元值被放置在后继者记录的运算数字段中。在步骤1550中，最后的记录的运算符的结果被呈现为由树表示的布尔表达。

[0169] 图16图示了根据本发明的实施例的表示规则树结构的替代方法。与规则相关的过滤器(条件)集合基于如先前所述的内容描述符、描述符准则和过滤器运算符来定义。过滤器集合的定义被存储在过滤器定义阵列。布尔运算符的集合被定义具有被指定空后继者的一个运算符以及被指定来自运算符的集合的后继者的每个其它运算符。过滤器形成树的叶并且被划分成过滤器的子集，其中子集的过滤器形成了来自布尔运算符的集合当中的布尔运算符的运算数。注意到，如果过滤器的每个子集包括两个过滤器，则布尔运算符的总数目等于过滤器的总数目减去1。

[0170] 考虑M个过滤器的集合，M>1，形成了包括N个叶记录的叶向量的模板；1<N<M。每个叶记录包括来自运算符的集合当中的叶运算符和过滤器的各自的子集。在安装阶段，每个叶记录包括运算符和过滤器定义阵列中的运算数(过滤器)的索引。每个叶记录的过滤器的值被确定用于各个容器组件。

[0171] 形成了具有等于N个数目的叶记录的多个节点记录的节点向量。每个节点记录具有节点运算符字段和节点状态字段。在安装阶段，节点记录为空，不包含数据。在叶记录的处理期间初始地确定节点记录的布尔运算符和节点状态。N个节点记录可以以任何顺序来布置。然而，方便的是，假设节点记录初始具有与叶记录的一对一对应关系。因此，节点记录j的布尔运算符是叶向量j，1≤j≤N的后继者布尔运算符。

[0172] 一旦确定了过滤器的值，就将每个叶运算符应用于各自的过滤器(各自的二元条件)，并且将结果放置在节点记录的节点状态字段中。将每个叶运算符的后继者放置在节点记录的节点运算符字段中。

[0173] 在处理了所有叶记录之后，处理节点记录。然后标识在此被称为连接节点记录的具有共同运算符的节点记录。然后，将共同运算符应用于所有连接节点记录的节点状态以产生新的状态。从连接的节点记录中选择的节点记录的运算符字段被共同运算符的后继者取代，并且所选择的节点记录的节点状态字段被刚确定的新的状态取代。从节点向量中删除剩余的连接节点记录。因此，通过用组合的节点记录取代连接节点记录的每个集合，减少了在节点向量中的节点记录的数目。标识连接节点记录的过程继续递归，直到节点向量仅包含一个节点记录。在剩余的一个节点记录的运算符字段中应用布尔运算符的结果是估计布尔表达的结果。最后的节点记录的布尔运算符具有空后继者。

[0174] 图16图示了用于编码图13的示例性树的规则树的递归规则构造1600，它表示形成树的叶的十一个过滤器的布尔表达。叶(过滤器)被标记为L1至L11。在图13的树中，叶L1与任何其它叶都不相关联。为了一致性，叶L1人工地与叶Φ相关联，它与其共享被动运算符

θ_1。插入的叶Φ被分配了“不关注”值。如本领域中已知的，“不关注”值可以方便地分配“真”状态或“假”状态。

[0175] 由用户来定义布尔运算符θ_2至θ_{11}的集合。运算符θ_2至θ_6与叶记录相关联，而运算符θ_7至θ_{11}与节点记录相关联。运算符θ_{11}具有空后继者，并且每个其它运算符θ_2至θ_{10}具有来自运算符θ_7至θ_{11}的集合的后继者，如图16的列表1610所图示的。

[0176] 叶向量1620包括分配了放置在运算符字段1624(1)至1624(6)中的运算符θ_1至θ_6的六个记录1622，分别被标识为1622(1)至1622(6)，相对应运算数的索引被放置在运算数字段1626(1)至1626(12)中。在六个叶记录的运算数字段中的运算数是{Φ，L1}、{L2，L3}、{L4，L5}、{L6，L7}、{L8，L9}和{L10，L11}。

[0177] 当确定了二元值时(图5和图6)，处理叶记录1622。从叶记录1622(1)开始，人工被动运算符θ_1仅将L1的值传递到节点记录1642(1)的节点状态字段。作为θ_1的后继者的运算符θ_{11}被放置在节点记录1642(1)的运算符字段中。然后，处理第二叶记录1622(2)，其中将运算符θ_2应用于级别L2和L3(过滤器L2和L3)以产生要放置在节点记录1642(2)的节点状态字段中的二元值B2。作为运算符θ_9的运算符θ_2后继者被放置在节点记录1622(2)的运算符字段中。该过程继续直到确定了所有节点记录1642(1)至1642(6)。

[0178] 该过程仅使用节点向量1640来递归地继续；不再需要叶向量1620。在节点1642(1)中的运算符θ_{11}在节点向量1640(1)中不配对。因此，节点记录1642(1)保持不变。类似地，节点记录1642(2)保持不变，因为节点记录1642中的任何一个都不包括运算符θ_9。节点记录1642(3)和1642(4)具有取代B3的共同运算符θ_7，它被应用于运算数B3和B4以产生要在节点记录1642(3)的节点状态字段中放置的二元结果B7。作为θ_9的运算符θ_7的后继者运算符被放置在记录1642(3)的运算符字段中，取代θ_7。删除了现在在新的组合记录1642(3)中占用的节点记录1642(4)。类似地，节点记录1642(5)和1642(6)被组合在具有运算符θ_8的后继者运算符θ_{10}以及通过将共同运算符θ_8应用于运算数B5和B6确定的节点状态B8的新的节点记录中。节点向量1640现在缩减成通过附图标记1640(2)标识的四个节点记录。节点记录1640(2)简单地重写节点记录1640(1)。

[0179] 该过程递归地继续，组合了节点记录1642(2)和1642(3)以产生新组合的节点记录1642(2)同时节点记录1642(1)和1642(4)保持不变；两个不变的节点记录现在是在减少的节点向量1640(3)中的记录1642(1)和1642(3)。

[0180] 节点记录1642(2)和1642(3)具有共同运算符θ_{10}。运算符θ_{10}被应用于运算数B9和B8以产生放置在节点记录1642(2)的节点状态字段中的新状态B10。运算符θ_{10}的后继者运算符θ_{11}被放置在节点记录1642(2)的节点运算符字段中。节点记录1642(1)和1642(2)的节点记录的共同运算符θ_{11}被应用于运算数B1和B10以产生布尔表达的输出B*。

[0181] 图17图示了使用图16的叶向量模板1620和节点向量模板1640来确定规则树的输出的过程。在步骤1720中，确定了表征考虑中的数据内容的布尔条件(布尔过滤器L2至L11)的集合。在步骤1722中，形成了具有N>1个叶记录1622的叶向量1620。每个叶记录1622包括布尔运算符字段1624和用于布尔条件(过滤器L2至L11的子集)的子集的字段。在步骤1724中，形成了N个节点记录1642的节点向量1640。每个节点记录1642包括布尔运算符字段1644和节点状态字段1648。在步骤1726中，每个叶运算符被应用于从如上参照图5和图6描述的内容数据的特性确定的布尔条件(布尔过滤器)的各自的子集。二元结果被放置在选择的节

点记录1642的节点状态字段中。在步骤1728中，每个叶运算符的后继者被放置在所选择的节点记录的运算符字段中。在步骤1730中，用组合的记录来取代具有共同运算符的节点记录，因此减少了节点向量1640的节点记录1642的数目。在步骤1732中，共同运算符被应用于所取代的节点记录的节点状态，并且二元结果被放置在所组合的记录的运算符字段中。在步骤1734中，从图16的列表1610确定的共同运算符的后继者被放置在所组合的节点记录的运算符字段中。在步骤1736中，如果剩余节点记录的数目大于1，则重新访问步骤1730以继续组合共同运算符的节点记录的过程。如果剩余记录的数目是1，则剩余节点记录的运算符被应用于节点记录的节点状态，并且该结果确定是否需要执行编辑动作(步骤1740)。

[0182] 图18是详述图17的过程的流程图。在步骤1820中，为了在图19中详细描述，规划了叶向量1620。顺序地考虑叶记录1622(1)至1622(N)。在步骤1824中，索引j被设置成等于0。如果步骤1826确定要处理更多的叶记录，则步骤1828将索引j增加1并且获取与当前叶记录的叶索引相对应的叶集合(过滤器集合)，并且步骤1830获取当前叶记录的运算符θ(θ_1至θ_6中的一个)。步骤1832将该运算符应用于所获取的叶集合，从而产生二元输出B。在步骤1834中，从图16的列表1610确定后继者S(θ)。

[0183] 节点向量1640的节点状态字段和运算符字段在此被标记为U(j)、V(j)，1≤j≤N，即，U(j)和V(j)定义节点记录1642(j)，1≤j≤N。在步骤1836中，B的值被放置在节点向量1640的节点状态字段U(j)中，并且S(θ)的值被放置在节点向量1640的运算符字段V(j)中。当处理了所有叶记录1622(1)至1622(N)时，索引j等于叶记录N的数目，并且节点向量1640的每个节点记录1642具有各自的节点运算符和节点状态。在步骤1840中，节点向量1640的节点记录1642的当前数目v被设置成等于j(它等于N)。在步骤1842中，如果节点记录v的当前数目大于1，则节点向量被扫描以收集具有相同运算符的所有节点记录1642并且组合这样的记录。在扫描之前，节点记录的当前数目v*=v被注意(步骤1843)以使得能够检测节点记录的数目的变化。在步骤1844中，索引k被设置成等于零，并且步骤1846记录节点记录1642(k)的运算符θ=V(k)。步骤1848检查节点向量1640的后续节点记录以标识具有相同运算符θ的后续节点记录的数目μ。如果所标识的后续节点记录的数目μ是零(步骤1850)，则在步骤1852中索引k增加1，并且如果索引k小于节点记录的当前数目v，则重新访问步骤1846。否则，步骤1856收集相同运算符θ的节点记录的(μ+1)个运算数，并且将运算符θ应用到(μ+1)个运算数以确定组合的节点记录的新状态B。在步骤1860中，删除了后续μ个标识的节点记录，并且步骤1862在节点记录1642(k)的节点状态字段U(k)中插入新状态B，并且在节点记录1642(k)的运算符字段V(k)中插入后继者运算符S(θ)。在步骤1864中，剩余节点记录的数目被确定为(v-μ)。在步骤1864之后应用了步骤1852和1854，以确定节点向量1640是否包含进一步的共同运算符的节点记录。如果步骤1854确定k小于v，则继续从步骤1846扫描节点向量。否则，如果步骤1854确定k=v(k无法超过v)，则步骤1855确保v的当前值(在步骤1864中最后更新的)小于先前的值v*。否则，在步骤1880中报告错误。注意到，如果用户提供的布尔表达的表示不正确，则不满足步骤1855的要求v<v*。如果v<v*，则步骤1855之后是步骤1842。如果步骤1842确定剩余节点记录的数目是1，则剩余节点记录的运算符被应用于各自的运算数，以确定状态B*(步骤1890)，其确定各自的编辑动作。

[0184] 图19详述了规划图16的叶向量1620的图18的步骤1820。在步骤1920中，规划了过滤器(条件)的集合，并且在步骤1922中，基于如上参照图5和图6描述的用户输入确定了叶

运算符。叶运算符被顺序应用以生成相对应的叶记录1622。如果步骤1924确定还没有应用所述至少一个运算符，则步骤1926向叶向量添加新的叶记录1622。步骤1928选择剩余运算符中的一个，并且步骤1930向叶记录的运算数字段1624添加相关联的过滤器。重复步骤1930，直到步骤1932确定属于所选择的运算符的所有过滤器已经被包括在当前叶记录1622中。当完成了当前叶记录1622时，如在步骤1932中确定的，重新访问步骤1924。当步骤1924确定已经考虑了所有的叶运算符，则将完成的叶向量1620呈现给图18的步骤1824。

[0185] 图20图示了预先计算过滤器集合的每个值的布尔表达的二元值的方法。过滤器的集合通过具有含有与过滤器的一对一对应关系的多个比特的比特字符串来表示，使得在该字符串中的每个比特对应于一个过滤器。利用$\mu>1$个过滤器，字符串包含μ个比特，并且假设值的范围从0到$2^{\mu}-1$。在步骤2012中，设置了开始字符串值0(μ个比特都被设置成零)，并且具有2^{μ}个条目的规则向量的每个条目被初始化成“0”。在步骤2014中，使用如参照图10、15或17描述的方法中的一个来估计布尔表达。在步骤2016中，二元结果(“真”、“假”)被存储在与字符串(0到$2^{\mu}-1$)的当前值相对应的位置处的规则向量中。在步骤2018中，字符串值通过添加1而增加。当步骤2030确定字符串的μ个比特中的每个具有值0时，完成了规则向量的生成(步骤2040)。注意到，μ个比特中的每个具有值“1”的字符串对应于规则向量的第$2\mu-1$条目，并且在步骤2018中添加1将字符串重新设置成μ个零。替代地，字符串可以具有($\mu+1$)个比特，最高有效位用于指示完成了规则向量生成。然后，规则向量可以用于直接确定运行时布尔表达的二元值，因此增加了内容过滤系统的吞吐量。

[0186] 总之，过滤数据容器的方法然后包括以下步骤：

[0187] (1)定义过滤器(二元条件)的集合，其中每个过滤器是选择的内容的描述符和各自的描述符的准则的函数。

[0188] (2)定义规则的集合，每个规则指定布尔表达以及相应的内容过滤动作。

[0189] (3)一次考虑一个布尔表达。

[0190] (4)考虑μ个过滤器(二元条件)的布尔表达。过滤器被表示为μ个比特的字符串，$\mu>1$。

[0191] (5)估计该字符串的2^{μ}个值中的每个的布尔表达，以产生2^{μ}个条目的规则向量，每个条目是与该字符串的2^{μ}个值中的一个相对应的内容度量的状态。

[0192] (6)对所有布尔表达重复步骤(5)。

[0193] (7)接收和解析数据容器。

[0194] (8)选择规则和根据数据容器的内容确定所指定的已选择的规则的μ个过滤器的值。

[0195] (9)对于所选择的规则相对应的规则向量编索引，并且确定与由μ个比特的字符串确定的索引相对应的规则向量中的条目的值。

[0196] (10)根据条目的值执行内容过滤动作。

[0197] (11)如果需要将新规则应用于所接收到的容器，则重复步骤(8)至(10)。

[0198] 图21图示了用于执行标记为L1、L2、L3和L4的四个过滤器的集合($\mu=4$)的布尔表达的规则的规则向量。过滤器的集合通过四个比特的字符串来表示。布尔表达被估计用于范围从“0000”至“1111”的该字符串的16个值2112中的每个，以产生与该字符串的字符串值j，$0\leq j\leq\mu$相对应的标记为“真”或“假”的二元输出2114(j)。

[0199] 一旦接收到容器，则检查容器组件的内容以确定图21中考虑的用于该规则的四个过滤器的集合。如果例如四个过滤器的集合具有值“1”、“0”、“0”和“1”，产生“1001”的字符串2140，则直接从二元规则向量2114的位置9（二元数1001）读取布尔表达的值。

[0200] 通过图21的示例性图示，图20的方法适合采用适度数目的运算数（过滤器）的布尔表达的规则。例如，利用8个过滤器，二元规则向量2114将相对短，仅具有256比特。如果布尔表达具有不止16个运算数，则例如，它可以优选地每当需要时估计布尔表达而不是存储大的二元规则向量。超过16的每个布尔表达的运算数的数目可以不同。

[0201] 图22图示了与过滤器定义和规则定义相关的数据条目的过程。该过程开始于确定是否已经创建了规则文件（步骤2220）。如果还没有创建规则文件，则步骤2222使用本领域已知的常规方法来创建文件。下一步骤是向规则文件添加规则。在步骤2224中开始填充或更新规则文件。步骤2224打开规则文件并且将该过程导向步骤2226，它提示用户指示是否要编码新规则并将新规则添加到规则文件。填充或更新规则文件由用户来终止（步骤2228）。如果要添加更多的规则，则放置在用户接口520（图5）中的数据获取模块（未示出）或在操作、管理和维护模块230中创建规则模板（步骤2230）。规则模板可以可选地采取多种形式中的一种，这由用户来决定。规则模板的形式取决于：（1）要顺序地还是根据等级顺序来应用由用户指定的规则；以及（2）要以代数分析形式还是以树结构格式来输入规则的布尔表达，在树结构中，树的节点表示运算符和各自的运算数。在任一情况下，数据获取模块可以提供具有方便数据输入的指令的各自的模板。例如，数据获取模块可以通过提示用户输入简单的操作来引导用户构造代数形式的布尔表达，每个操作包括运算符和运算数的集合，然后进行到想要的表达。可以通过分别添加新运算符来验证所构造的表达的有效性。如果布尔表达被呈现为树结构，则数据获取模块可以显示通用树结构，它可以当用户输入与选择的树的节点相关的数据时被修整和验证。

[0202] 在步骤2232中，提供了以任何合适格式编码的规则标识符。在步骤2234中，指定了规则动作，并且步骤2240定义了与规则相关联的布尔表达。根据相关联的布尔表达的值来应用特定规则的规则动作。步骤2240包括步骤2242、2244、2246、2248、2250、2252和2260。步骤2242如图3所示创建了附图标记为340的过滤器模板。步骤2244设置了过滤器的类型，它可以是考虑中的容器的内容的多个描述符中的一个。步骤2246设置过滤器的运算符，它可以从一元运算符、二元运算符、算术运算符、比较运算符、逻辑运算符、设置运算符和用户定义的运算符的菜单中选择。步骤2248设置了过滤器的准则，它是与在步骤2244中选择的描述符相关的目标值或阈值。步骤2250提示用户定义用于该规则的新的过滤器或继续定义要应用于目前为止指定的过滤器的集合的布尔表达。为了添加另一过滤器，重新访问步骤2242至2248，直到在步骤2250中用户确定所有相关的过滤器都存在。步骤2252提示用户输入根据上述格式中的一个的布尔表达。注意到，规则可以仅基于一个过滤器，如图9所示，在这种情况下，布尔表达缩减成被动运算符，它仅适用一个过滤器的值以确定是否要应用在步骤2234中指定的规则工作。

[0203] 步骤2260将刚构造的规则附加到在步骤2224中打开的规则文件。注意到，一旦处理了接收到的容器，则在“运行时间”要确定因此构造的每个规则的过滤器的值。编码的规则包括过滤器标识符，它仅是存储过滤器的阵列（未示出）的索引。

[0204] 在系统安装或更新期间执行图22的过程。编码和存储在规则文件中的规则被“实

时”地激活。

[0205] 图23图示了顺序过滤接收到的具有多个组件的容器的组件的过程。处理用于内容过滤的容器的组件的顺序是任意的，并且可以由用户来设置。如果由于一些操作原因而对整个容器施加了整体约束，则作为结果将具有处理组件的顺序。

[0206] 当控制器提示时，内容过滤的过程在步骤2320中开始(图2)。如果已经选择了要处理组件的顺序，则步骤2340确定在步骤2350是否要处理至少一个组件。否则，步骤2380结束该处理并报告结果。在步骤2360中，执行可应用于考虑中的组件的规则设置的所有规则，并且步骤2340被重新访问以确定是否需要处理另一组件。模块插入指示应用于组件的任何过滤动作的通知。

[0207] 图24详述了步骤2360(图23)，其中规则的集合被应用于容器的内容。步骤2360应用于容器的组件。步骤2424确定是否已经应用了整个规则集合。如果是，则步骤2480将指示由于指定规则的集合而引起的任何内容过滤动作的通知附加到容器。否则，步骤2428选择当前规则，并获取与所选择的当前规则相关联的所有相关过滤器的定义。注意到，如果一个规则的结果影响了另一规则的选择，则可以以特定顺序来布置规则。另外，如将参照图25描述的，可以通过形式图而不是简单的阵列来表示规则相互依赖性。

[0208] 步骤2430执行所选择的当前规则。步骤2430包括步骤2432、2436、2440和2444。步骤2432确定是否已经激活了在步骤2428中标识的所有过滤器以确定每个过滤器的二元值。当将过滤器的运算符应用于各自的运算数以产生该过滤器的二元值时就称为激活了过滤器。如果已经激活了与当前规则相关的所有过滤器，则步骤2432将控制传输到步骤2460。否则，执行步骤2346、2440和2444以产生考虑中的过滤器的值。如参照图5和图6描述的，步骤2436基于考虑中的容器内容的特性来获取运算符和各自的运算数的值。步骤2440将运算符应用于运算数，并且步骤2444记录当前过滤器的值用于在估计当前规则的布尔表达中使用。

[0209] 步骤2460根据图10、图15或图17的编码方法中的一个来获取布尔表达。步骤2464估计布尔表达。步骤2468可以将与当前规则相关联的内容过滤动作应用于服从如在步骤2464中确定的布尔表达的值的考虑中的内容的内容。在步骤2470中，如果当前规则的内容过滤动作导致删除整个容器组件，则不需要执行后续规则(如果有的话)，并且步骤2360向所删除的组件附加各自的通知。如果没有编辑内容，或编辑但没有删除，则重新访问步骤2424以确定是否需要将更多的规则应用于考虑中的内容。注意到，如果具有超过特定阈值的附件，或者如果具有无法移除的恶意插入，则可以删除整个组件。

[0210] 规则相互依赖性

[0211] 通常，可应用于特定内容的规则可以具有补充动作、冲突动作或互斥动作。利用补充动作，内容过滤结果可以与实现规则的顺序无关。利用冲突动作或互斥动作，一个动作代替另一个。根据本发明的实施例，可以提示用户使用图形来定义规则的相互关系。

[0212] 图25图示了指示标记为规则1至规则5的五个规则的等级布置的图形。规则的状态在此被定义为由于执行规则的布尔表达而产生的二元值。

[0213] 规则1的“真”状态导致标记为“动作1”的动作，之后，认为完成了步骤2360。“动作1”可以需要两个相反极性中的一个；第一个是因为它太大或不能修复而删除整个组件，或因为它太短以致不能包含恶意插入而确定组件是可接受的。规则1的“假”状态指示内容通

过第一测试并且应当进行规则2的第二测试。

[0214] 规则2的“真”状态导致后面是实现规则5的标记为“动作2”的动作。规则2的“假”状态指示内容通过第二测试并且应当进行规则3的第三测试等等。如果规则4的状态是“假”，则该过程在没有编辑内容的情况下结束。该过程还可以在(仅)实现一个的情况下结束：{动作1}、{动作2和动作5}、{动作3}和{动作5}。

[0215] 图26图示了由操作管理器230使用的在下面列出的模块。每个模块包括存储在计算机可读存储介质中的计算机可读指令。

[0216] (1)服务器单元配置模块2610，用于配置服务器单元以接受根据指定的协议形成的多媒体数据容器。

[0217] (2)负载平衡模块2612，用于在同样配置的服务器单元当中公平地分发多媒体数据容器以处理共同类型的数据容器。

[0218] (3)过滤器定义模块2614，用于从用户获取过滤器集合的定义，每个过滤器指定内容描述符、描述符准则和运算符的定义。

[0219] (4)布尔表达获取模块2616，用于使得用户能够根据代数句法来提供布尔表达。

[0220] (5)布尔表达获取模块2618，用于使得用户能够以树结构的形式来提供布尔表达。

[0221] (6)布尔表达获取模块2620，用于使得用户能够通过编辑和修整通用树的图形来输入布尔表达，其中树的每个节点表示运算符和各自的运算数集合。

[0222] (7)规则构造模块2622，用于从用户获取内容过滤规则的集合，每个规则指定过滤器和过滤动作的子集的布尔表达。

[0223] (8)规则验证模块2624，用于验证针对规则指定的布尔表达的正确性。

[0224] (9)规则布置模块2626，用于以由规则执行的特定过滤动作排除至少一个后续规则的执行所根据的顺序来布置规则。

[0225] (10)规则互依赖性模块2628，用于提示用户以给定规则的各自的布尔表达的值为条件来指定给定规则的连续规则。

[0226] (11)规则图形定义模块2630，用于以形式图的形式来呈现内容过滤规则的集合(图25)。

[0227] (12)规则优化模块2632，用于使用常规逻辑优化技术来优化每个规则的布尔表达，以最小化处理努力。

[0228] (13)规则预处理模块2634，用于选择特定规则，每个规则实行最多包括预设数目的过滤器的过滤器子集；对特定规则中的每个估计用于过滤器的子集的所有的值的过滤器子集的布尔表达，以产生2^m个比特的阵列，$m>1$是过滤器子集中的过滤器的数目；以及将比特的阵列存储在存储设备中(图20和21)。

[0229] (14)分类模块2636，用于将容器分类成容器类型，每个容器类型对应于在源处规划的容器所根据的协议；以及从多个容器当中将特定类型的容器导向特定控制器。分类模块2636可以与网络接口210或操作管理器230相关联。

[0230] 图27图示了根据本发明的实施例的由代码转换器280使用的在下面列出的模块。每个模块包括存储在计算机可读存储介质中的计算机可读指令。

[0231] (a)模块2710，用于表征多媒体数据容器的每个组件的内容，确定内容描述符，应用运算符以及确定过滤器的状态。

[0232] (b)模块2720，用于布尔表达的运行时间估计并且确定规则的二元输出。可以根据代数句法或作为树结构来呈现布尔表达。

[0233] (c)模块2730，用于执行与服从于各自规则的布尔表达的预设值的给定容器内容相关的过滤动作。

[0234] 图28图示了根据本发明的实施例的包括代码转换器服务模块2810、程序寄存器2820和程序存储器2840的代码转换器280。代码转换器服务模块包括存储在计算机可读存储介质中的计算机可读指令，当执行计算机可读指令时，使得处理器的核心：从多个控制器中的特定控制器接收特定容器和编辑请求；选择和执行与编辑请求相关的程序；以及将结果返回特定控制器。程序寄存器包括计算机可读指令，当执行计算机可读指令时，使得处理器根据各自的功能来组织程序；以及用各自的新的程序来取代现有程序。

[0235] 控制器240(图2)将编辑请求转发到代码转换器280。一旦接收到编辑请求2850，则代码转换器服务模块2810标识使用包含在编辑请求中的信息执行的插件程序。代码转换器服务模块2810执行所选择的插件程序并且将结果返回各自的控制器240。

[0236] “插件”在此被定义为设计成执行特定任务的自包含的模块。程序存储器2840包括存储在计算机可读存储介质中的计算机可读指令，并且包括两种类型的插件：

[0237] (a)初始加载的驻留插件2842；以及

[0238] (b)动态加载的外部插件2844，外部插件可以取代驻留插件。

[0239] 驻留插件提供了基本的功能，并且外部插件提供了附加功能，内容过滤和病毒扫描时这样的功能的两个例子。

[0240] 插件被注册到程序寄存器2820，程序寄存器2820管理插件注册和接入。程序寄存器2820基于它们的特性来组织插件。插件可以布置在插件组中。

[0241] 插件程序以预定义的方式来组织插件的执行。从确定用于预定义的具有特定目标的插件集合的执行逻辑的简单的指令集合来构建插件程序。

[0242] 接下来呈现使用插件的简单程序的指令的例子。

[0243] (01)OnErrorGoto TERMINATION_PLUGIN

[0244] (02)Execute DEFAULT_SETUP_INITIAL_PROFILE

[0245] (03)Execute DEFAULT_SETUP_SESSION

[0246] (04)Execute DEFAULT_PRE_PROCESS_DECODE_PLUGIN

[0247] (05)ExecuteGroup GROUP_HOT_PIPELINE_DUAL_LOAD_AROUND_CREATE

[0248] (06)Execute DEFAULT_TRANSFORMER_PLUGIN

[0249] (07)Execute DEFAULT_CREATOR_PLUGIN

[0250] (08)ExecuteGroup GROUP_HOT_PIPELINE_CREATOR

[0251] (09)ExecuteGroup GROUP_HOT_PIPELINE_DUAL_LOAD_AROUND_CREATE

[0252] (10)Execute DEFAULT_CHARGE_DATA_RECORD_PLUGIN

[0253] (11)Execute DEFAULT_OPTIMISER_PLUGIN

[0254] (12)ExecuteGroup GROUP_HOT_PIPELINE_ANALYSER

[0255] (13)Execute DEFAULT_ENCODE_PLUGIN

[0256] (14)Label TERMINATION_PLUGIN

[0257] (15)Execute DEFAULT_CHARACTERIZE_PLUGIN

[0258] (16)ExecuteGroup GROUP_HOT_PIPELINE_TERMINATOR

[0259] (17)Execute DEFAULT_UNSETUP_SESSION

[0260] (18)Execute DEFAULT_CHARGE_DATA_RECORD_PLUGIN

[0261] 注意到,引入左边的数目仅是便于参考并且不一定构成指令的一部分。

[0262] 每个“执行(Excute)”命令具有作为总是表示驻留插件名称的变量的插件的名称。因为外部插件是可选的,所以从未通过名称来直接参考外部插件,并且因此仅当存在时才执行外部插件。每个“ExecuteGroup”命令具有作为变量的插件组的名称。命令“Execute Group”执行属于该组的所有插件。

[0263] 行1声明在任何错误时程序跳转到行14并且继续执行行15至18。行2和3执行要完成的适配的设置;如果需要,行4执行输入的解码,例如,如果输入是电子邮件,则将它分解成它的子组件;行5和行9执行内容过滤插件所属于的插件组。因此,如果存在,则它在行5开始执行并且在行9终止;行6和行7分别用于执行创建适配管线所必需的设置操作并且实际创建它。适配管线包含要执行的操作的集合以执行所要求的适配;行8意指执行在执行之前对适配管线有影响的外部插件;行10提供将参与适配的输入组件的详情。行18对输出组件执行类似的任务。这样的信息可以被分析用于报告、付费和与适配功能不一定相关的其它目的;行11执行适配管线优化;行12执行在它的执行之前执行适配管线的分析和优化的任何外部插件;行13执行适配管线;行15表征由于执行适配管线而生成的输出组件;行16执行对生成的输出组件具有影响的任何外部插件;以及行17执行完成适配的附加步骤(诸如提供详细的适配记录)。

[0264] 规则被永久存储在“规则文件”中。规则文件可以应用到不止一个控制器。使用规则文件的内容过滤器将包含在规则文件中的规则应用于媒体(内容)。如果给定规则估计成“真”,则执行相对应的动作。动作可以包括移除诸如病毒(包括移动特定的病毒)的不想要的内容;移除特定类型的媒体(诸如游戏);使用第三方应用对媒体执行动作(诸如扫描病毒媒体)。然而,特定类型的内容可以未经处理地通过。

[0265] 在操作管理器230中完成规则的定义(图2)。在代码转换器280中完成用于影响内容编辑(过滤和适配)的规则的应用。一旦创建了规则文件,则用户可选地配置控制器240中的一个以将规则文件以及任何适配请求发送到它选择的代码转换器280。

[0266] 动作“放弃”确保媒体不是内容适配过程的输出的一部分。动作“扫描保持”导致扫描媒体病毒。这假设安装了反病毒外部插件。媒体实际上被“标记”为“扫描病毒”,使得在执行适当插件程序时,其中执行了反病毒插件,扫描了标记为“扫描病毒”的所有媒体以寻找病毒。

[0267] 下面给出了称为规则1的规则的例子。

[0268] Rule Name=″MaxFileSize50000″Action=″Drop″

[0269] Filter Type=″MaxFileSize″Operaror=″Greater Than″Value=″50000″

[0270] FilterFilterOperator=″AND″Type=″Family″Operator=″NotEqual″

[0271] Value=″MESSAGE″

[0272] 与规则1相关联的名称是“MaxFileSize50000”,而与该规则相对应的动作是用于移除匹配包含在规则中的过滤器(多个)的任何媒体的“放弃(Drop)”。该规则指定了两个过滤器。第一过滤器是针对文件的大小应用的“MaxFileSize”类型。过滤器运算符是“大于”,

其中作为值是“50000”。第二过滤器特征在于称为“族(Family)”的类型。该过滤器针对媒体族来应用(例如,图像、音频等)。与过滤器相关联的运算符不是“不等于”并且该值是“消息”。通过使用布尔运算符“与”来组合过滤器。因此,如果文件具有大于50000的大小,并且不是族“消息”,则执行所指定的动作。

[0273] 接下来描述称为规则2的另一规则:

[0274]
```
Rule Name="MaxFileSize25000AndContentTypes"
Action="Drop"
        Filter Type="MaxFileSize"
        Operator="GreaterThan"  Value="25000"
              BracketOpen
        FilterOperator="AND"
        Filter  Type="ContentType"
        Operator="Equals" Value="image/wbmp"
        Filter FilterOperator="OR"
        Type="ContentType" Operator="Equals"
                    Value="image/png"
              BracketClose
```

[0275] 该规则的名称是“MaxFileSize25000AndContentTypes”并且相应的动作是“放弃(Drop)”。规则2的目的是移除匹配包含在该规则中的过滤器(多个)的任何媒体。接下来呈现规则2的结构的详细描述。

[0276] ●仅指定了下列过滤器

[0277] ○第一过滤器

[0278] ■过滤器类型是“MaxFileSize”并且针对文件的大小来应用过滤器;

[0279] ■运算符是“大于”;

[0280] ■值是“25000”;

[0281] ○以“与”布尔过滤器运算符开始的括号

[0282] ○第二过滤器

[0283] ■过滤器类型是“ContentType”并且针对媒体的内容类型(等同于mimetype)来应用过滤器;

[0284] ■运算符是“等于”;

[0285] ■值是“图像/wbmp”

[0286] ○第三过滤器:

[0287] ■具有布尔过滤器运算符“或”;

[0288] ■过滤器类型是“ContentType”并且针对媒体的内容类型(等同于mimetype)来应用过滤器;

[0289] ■运算符是“等于”;

[0290] ■值是“图像/png”

[0291] ○括号关闭

[0292] 因此,如果文件具有大于“25000”的大小且(具有等于“图像/wbmp”的内容类型或具有等于图像/png的内容类型),则执行在该规则中指定的动作。如果媒体是诸如电子邮件

或MMS的容器，则动作可以影响媒体或任何它的附件。动作可以包括：保持(适配媒体)；保持和扫描(在适配之前对媒体进行病毒扫描)；放弃(在最后的消息中不包括附件)；通过(没有适配媒体，使其保持未改变)。

[0293] 接下来描述在共同规则文件中定义的例子规则3。

[0294] Rule Name＝"MaxFileSize300000"Action＝"Drop"

[0295] Filter Type＝"MaxFileSize"Operator＝"GreaterThan"Value＝"300000"

[0296] 呈现了完成该共同规则文件的另一例子规则4。

[0297] Rule Name＝"VirusScan"Action＝"ScanKeep"

[0298] Filter Type＝"Family"Operator="Equals"Value＝"MESSAGE"

[0299] 在这种情况下，共同规则文件包含：

[0300] ●规则3，该规则3"放弃"具有大于300000的大小的所有文件；以及

[0301] ●规则4，该规则4对作为消息的任何媒体执行病毒扫描。

[0302] 考虑将共同规则文件应用于所有控制器并且一些控制器"X"已经定义了包含规则1和2的规则文件的情况。当向选择的代码转换器发送适配请求时，该控制器将发送包含规则1至4的被称为"RuleFile1"的规则文件。呈现了"RuleFile1"的结构。

[0303]

```
Rule Name="MaxFileSize50000" Action="Drop"
        Filter Type="MaxFileSize" Operator="GreaterThan" Value="50000"
        Filter FilterOperator="AND" Type="Family" Operator="NotEqual"
                Value="MESSAGE"
Rule Name="MaxFileSize25000AndContentTypes" Action="Drop"
        Filter Type="MaxFileSize" Operator="GreaterThan" Value="25000"
        BracketOpen FilterOperator="AND"
                Filter Type="ContentType" Operator="Equals" Value="image/wbmp"
```

[0304]

```
                Filter FilterOperator="OR" Type="ContentType" Operator="Equals"
                        Value="image/png"
        BracketClose
Rule Name="MaxFileSize300000" Action="Drop"
        Filter Type="MaxFileSize" Operator="GreaterThan" Value="300000"
Rule Name="VirusScan" Action="ScanKeep"
        Filter Type="Family" Operator="Equals" Value="MESSAGE"
```

[0305] 在提供该特征的实施例的背景下，XML用于管理在规则文件内部的规则的结构。这确保了便携性和缩放性。接下来呈现RuleFile1的XML版本。

[0306]

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentFiltering xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:type="ContentFiltering">
    <CompatibilityVersion>7.0</CompatibilityVersion>
    <Version>7.0</Version>
    <Name>RuleFile1</Name>
    <FilterRule Name="MaxFileSize50000" Action="Drop">
        <Filter Type="MaxFileSize" Operator="GreaterThan" Value="50000"/>
        <Filter FilterOperator="AND" Type="Family" Operator="NotEqual"
                Value="MESSAGE"/>
    </FilterRule>
    <FilterRule Name="MaxFileSize25000AndContentTypes" Action="Drop">
        <Filter Type="MaxFileSize" Operator="GreaterThan" Value="25000"/>
        <Bracket FilterOperator="AND">
            <Filter  Type="ContentType" Operator="Equals"
                Value="image/wbmp"/>
            <Filter FilterOperator="OR" Type="ContentType"
Operator="Equals"                   Value="image/png"/>
        </Bracket>
    </FilterRule>
    <FilterRule Name="MaxFileSize300000" Action="Drop">
        <Filter Type="MaxFileSize" Operator="GreaterThan" Value="300000"/
>
    </FilterRule>
    <FilterRule Name="VirusScan" Action="ScanKeep">
        <Filter Type="Family" Operator="Equals" Value="MESSAGE"/>
    </FilterRule>
</ContentFiltering>
```

[0307] 下面呈现了内容过滤的例子。考虑多媒体容器：

[0308] MMS特性：

[0309] 名称：mms1.mms

[0310] 族：消息

[0311] 大小：171100

[0312] 内容类型：应用/vnd.wap.multipart.mixed

[0313] 附件的数目：3

[0314] MMS附件特性：

[0315] 名称：image.jpg

[0316] 族：图像

[0317] 大小：75000

[0318] 内容类型：图像/jpg

[0319] 名称：image2.jpg

[0320] 族：图像

[0321] 大小：45000

[0322] 内容类型：图像/jpg

[0323] 名称：image.png

[0324] 族：图像

[0325] 大小:50000

[0326] 内容类型:图像/png

[0327] 根据下面的步骤来执行内容过滤:

[0328] ●MMS通过内容过滤;

[0329] ■由于媒体属于族“消息”,所以规则“VirusScan”估计成“真”;

[0330] ■媒体被标记为“扫描病毒”。

[0331] ●附件image.jpg通过内容过滤;

[0332] ■由于媒体不是消息并且它的大小大于50000,所以规则“MaxFileSize50000”估计成“真”;

[0333] ■媒体被标记为“放弃”。

[0334] ●附件image2.jpg通过内容过滤;

[0335] ■对于该媒体没有任何规则估计成“真”;

[0336] ●第二附件image.png通过内容过滤;

[0337] ■由于媒体具有大于25000的大小并且具有内容类型“图像/png”,所以对于该媒体规则“MaxFileSize25000AndContentTypes”估计成“真”;

[0338] ■媒体被标记为“放弃”。

[0339] 在执行了内容过滤插件之后继续插件程序。这导致通过反病毒插件来病毒扫描MMS媒体及其内容。然后,开始适配过程。考虑适配和内容过滤产生具有下列形式的输出MMS的情况。

[0340] MMS特性:

[0341] 名称:mmslout.mms

[0342] 族:消息

[0343] 大小:25175

[0344] 内容类型:应用/vnd.wap.multipart.mixed

[0345] 附件的数目:2

[0346] MMS附件特性:

[0347] 名称:image2.gif

[0348] 族:图像

[0349] 大小:24000

[0350] 内容类型:图像/gif

[0351] 名称:removal_notification.txt

[0352] 族:文本

[0353] 大小:75

[0354] 内容类型:文本/明文(plain)

[0355] 假设,作为内容适配的结果,适配了“image2.jpg”以输出“image2.gif”。注意到,“image.jpg”和“image.png”两者都被“放弃”并且由于应用了内容过滤动作而不是输出MMS的一部分。新媒体“removal_notification.txt”被添加到输出消息。这由于移除了“image.jpg”和“image.png”。代码转换器被设计成,一旦移除了媒体,则附接解释性文本通知。该通知意在向MMS的接收器提供对没有适配并且移除了原来在MMS中的一些媒体的解

释。

[0356] 以外部插件的形式在代码转换器中呈现了反病毒扫描。在这种情况下，插件架构用于向诸如McAffee或Kaspersky的第三方反病毒扫描引擎提供接口。由于具有任何外部插件，反病毒插件的存在是可选的。在插件程序级别，意指执行反病毒插件的插件程序将包含执行反病毒插件所属于的插件组的命令。

[0357] 反病毒插件的执行并没有自动暗示将对媒体病毒扫描。仅在通过内容过滤标记为“扫描病毒”的媒体上执行病毒扫描。一些第三方反病毒引擎可以被安装为单独的库。其它第三方反病毒引擎可以被安装为客户端服务器。反病毒插件将以正确地与第三方反病毒引擎对接这样的方式来写入。在任何情况下，反病毒插件是代码转换器进入点以在通过内容适配的媒体上执行病毒扫描。

[0358] 因此，在上述实施例中，已经提供了下列特征：(1)解析消息的能力，以便检查附件；(2)表征附件的能力，以便根据内容类型来对其过滤；以及(3)调节用户定义的、可扩展的并分级的规则集合，以确定是否需要媒体元素。

[0359] 编辑动作确定怎样处理媒体附件。编辑动作可以包括下列中的一个或多个：将附件呈现给适配过程；保持附件并在呈现给适配过程之前对媒体进行病毒扫描；以及放弃附件。编辑动作可以包括调用反病毒和入侵防护软件程序。

[0360] 尽管已经详细描述了本发明的具体实施例，但是应当理解，所描述的实施例意在是说明性的而非限制性的。在不背离本发明的最广义方面的范围的情况下，可以在下面的权利要求的范围内做出在附图中示出并在说明书中描述的实施例的各种变化和修改。

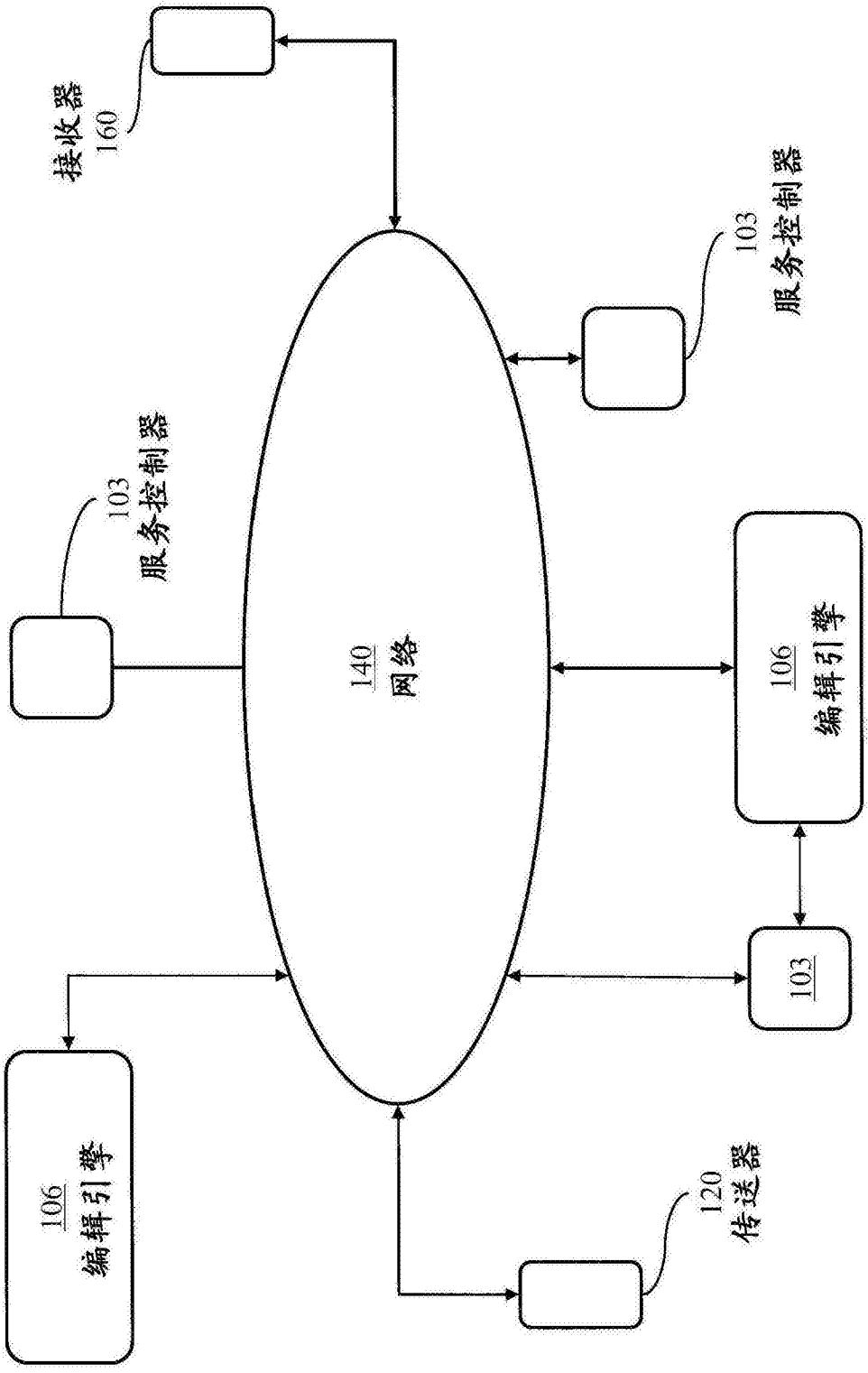

接收器
160
103
服务控制器
103
服务控制器
140
网络
106
编辑引擎
103
106
编辑引擎
120
传送器

图1

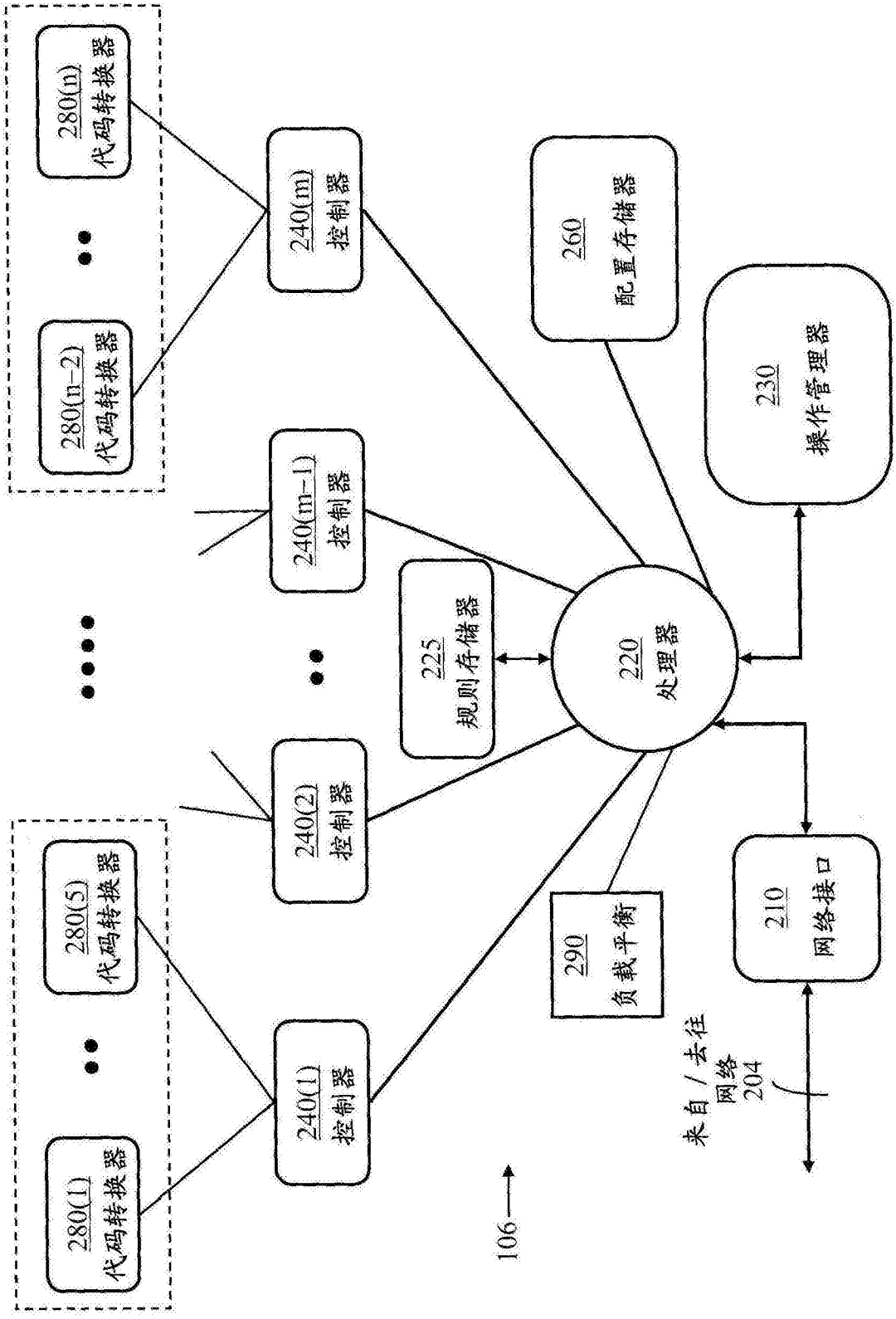

280(n)
代码转换器
280(n-2)
代码转换器
240(m)
控制器
260
配置存储器
230
操作管理器
240(m-1)
控制器
225
规则存储器
220
处理器
240(2)
控制器
280(5)
代码转换器
280(1)
代码转换器
240(1)
控制器
290
负载平衡
210
网络接口
106
来自/去往
网络
204

图2

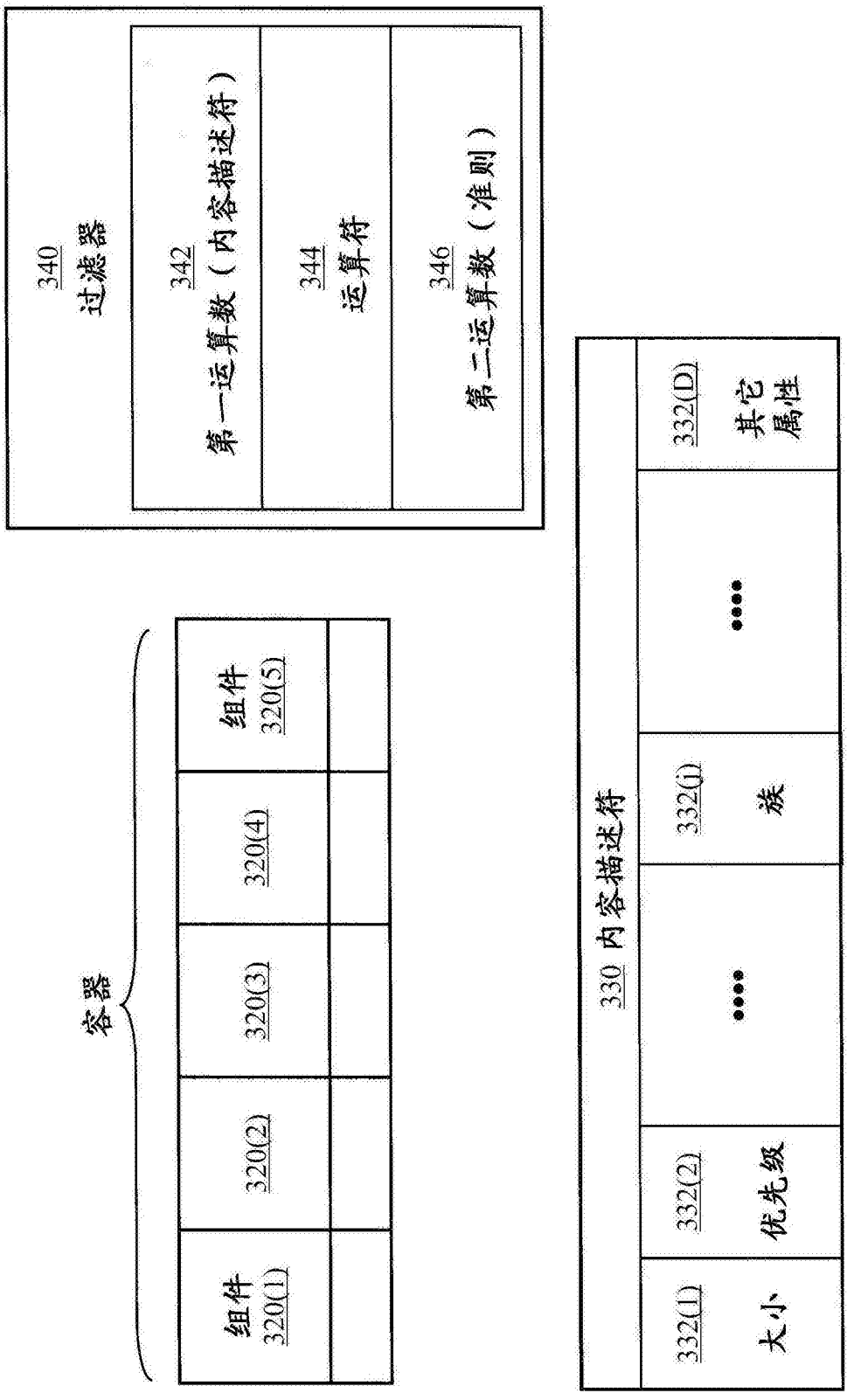

340
过滤器
342
第一运算数（内容描述符）
344
运算符
346
第二运算数（准则）
容器
组件 320(1)
320(2)
320(3)
320(4)
组件 320(5)
330 内容描述符
332(1)
大小
332(2)
优先级
332(j)
族
332(D)
其它属性

图3

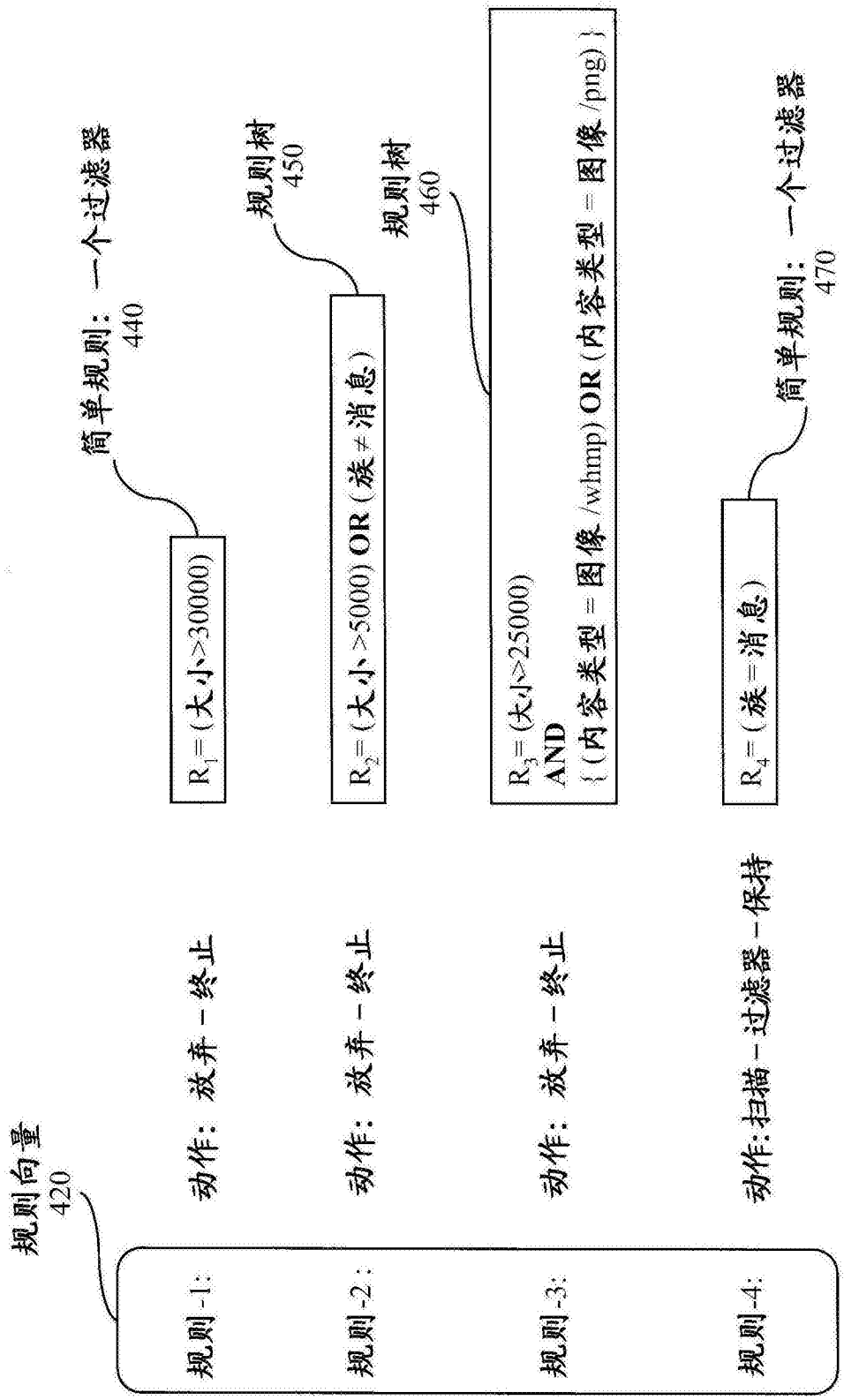

规则向量
420
规则-1:
动作: 放弃 – 终止
规则-2:
动作: 放弃 – 终止
规则-3:
动作: 放弃 – 终止
规则-4:
动作: 扫描 – 过滤器 – 保持
简单规则: 一个过滤器
440
R_1=(大小>30000)
规则树
450
R_2=(大小>5000) OR (族≠消息)
规则树
460
R_3=(大小>25000)
AND
{(内容类型 = 图像 /whmp) OR (内容类型 = 图像 /png)}
R_4=(族=消息)
简单规则: 一个过滤器
470

图4

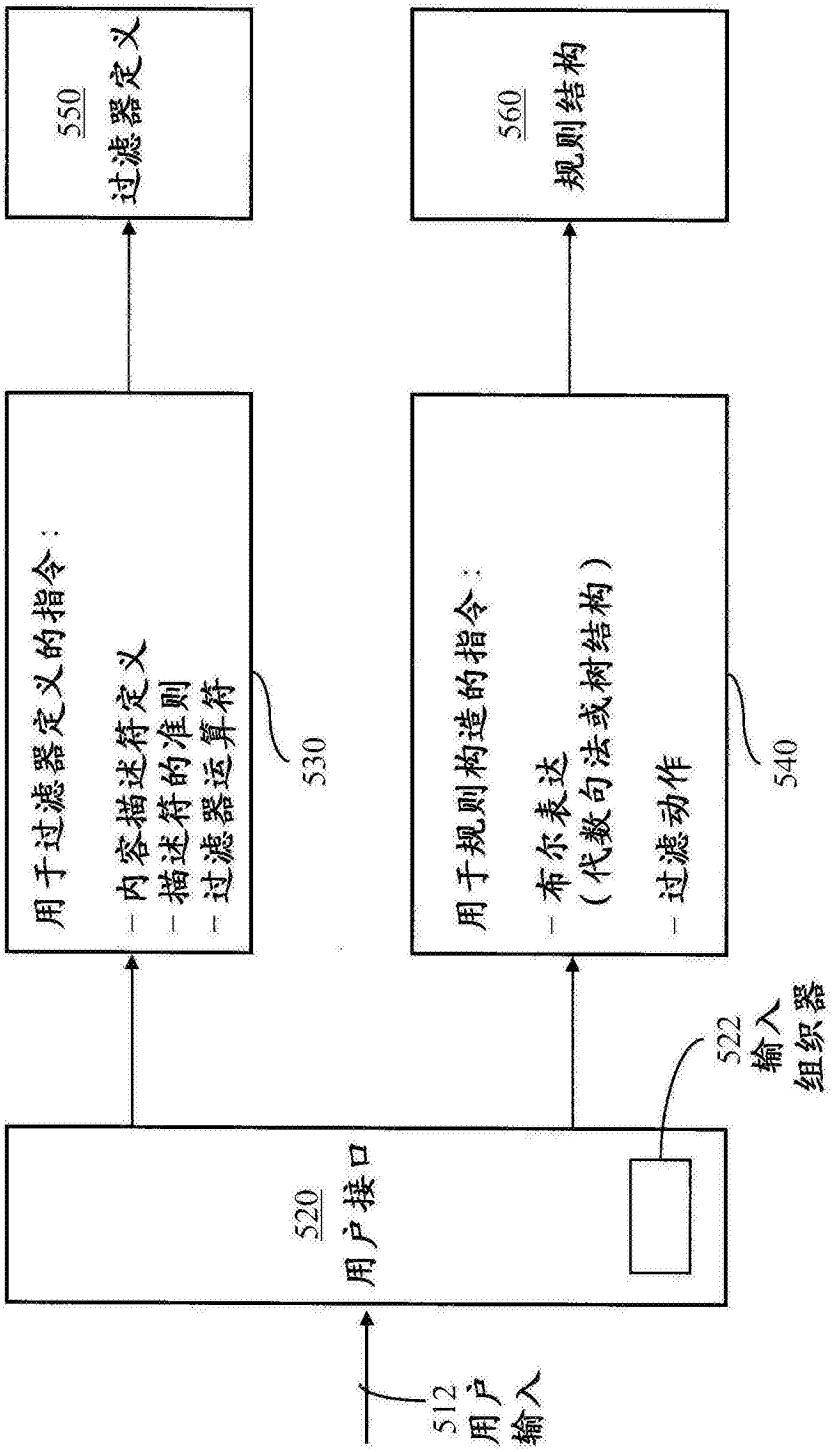

550
过滤器定义
560
规则结构
用于过滤器定义的指令：
– 内容描述符定义
– 描述符的准则
– 过滤器运算符
530
用于规则构造的指令：
– 布尔表达
（代数句法或树结构）
– 过滤动作
540
522
输入
组织器
520
用户接口
512
用户
输入

图5

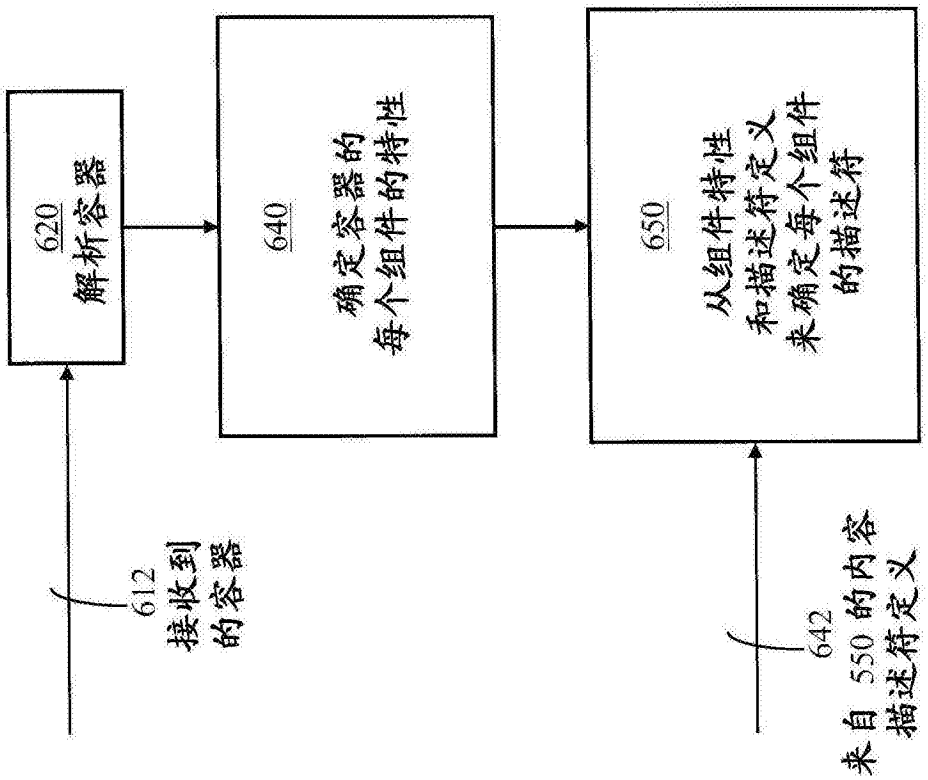

620
解析容器
640
确定容器的
每个组件的特性
650
从组件特性
和描述符定义
来确定每个组件
的描述符
612
接收到
的容器
642
来自 550 的内容
描述符定义

图6

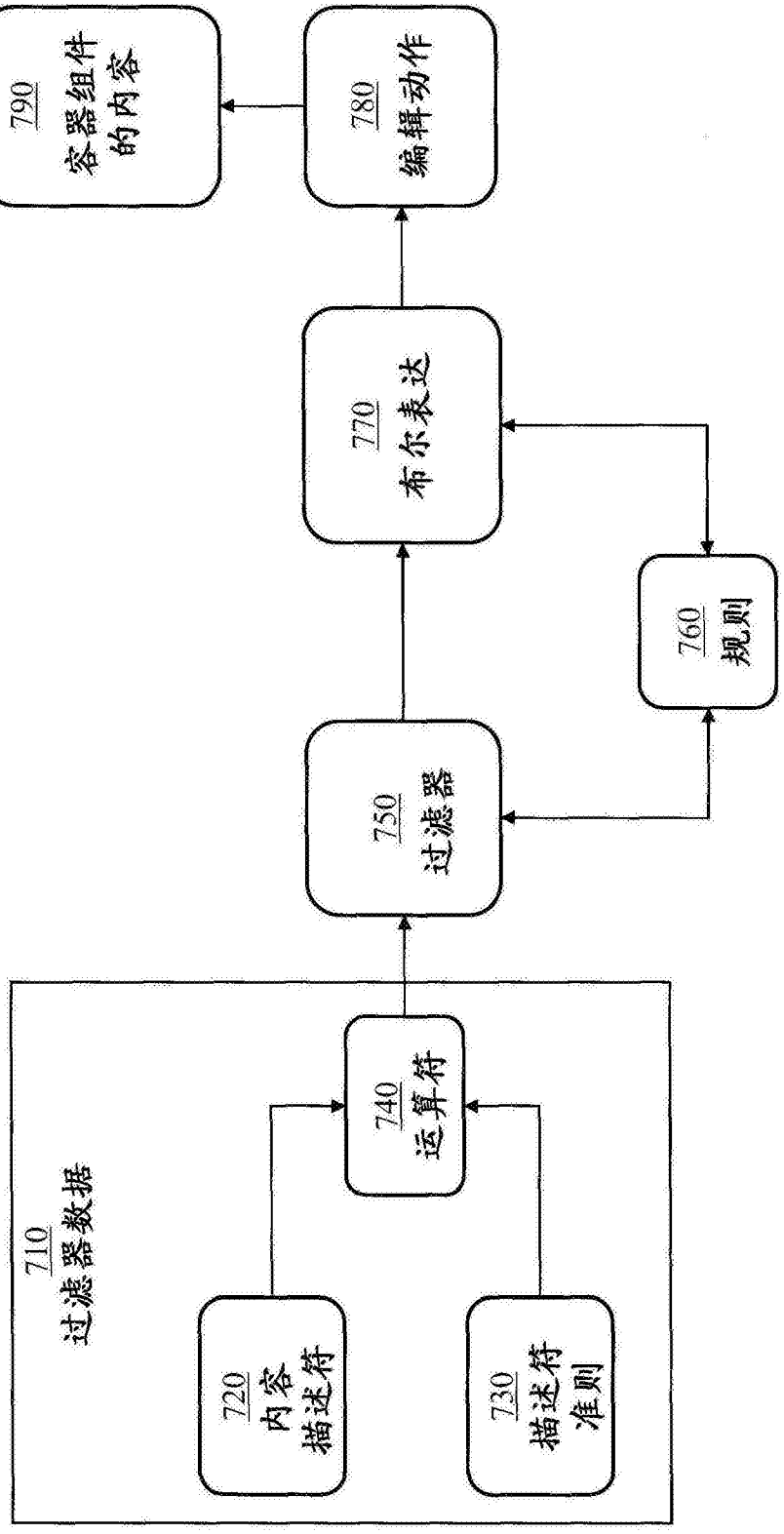

790
容器组件的内容
780
编辑动作
770
布尔表达
760
规则
750
过滤器
740
运算符
710
过滤器数据
720
内容描述符
730
描述符准则

图7

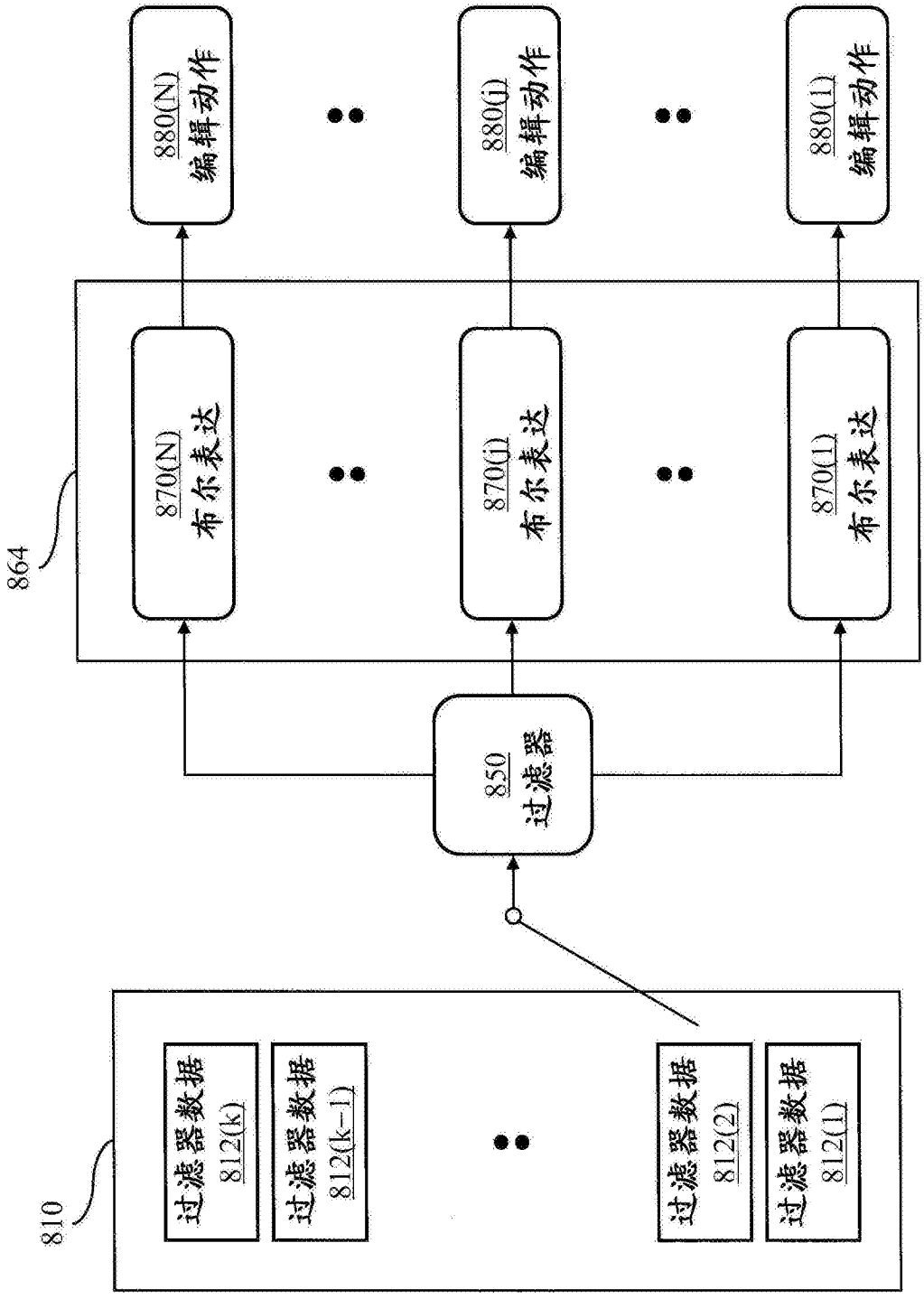

880(N)
编辑动作
880(j)
编辑动作
880(1)
编辑动作
870(N)
布尔表达
870(j)
布尔表达
870(1)
布尔表达
864
850
过滤器
810
过滤器数据
812(k)
过滤器数据
812(k-1)
过滤器数据
812(2)
过滤器数据
812(1)

图8

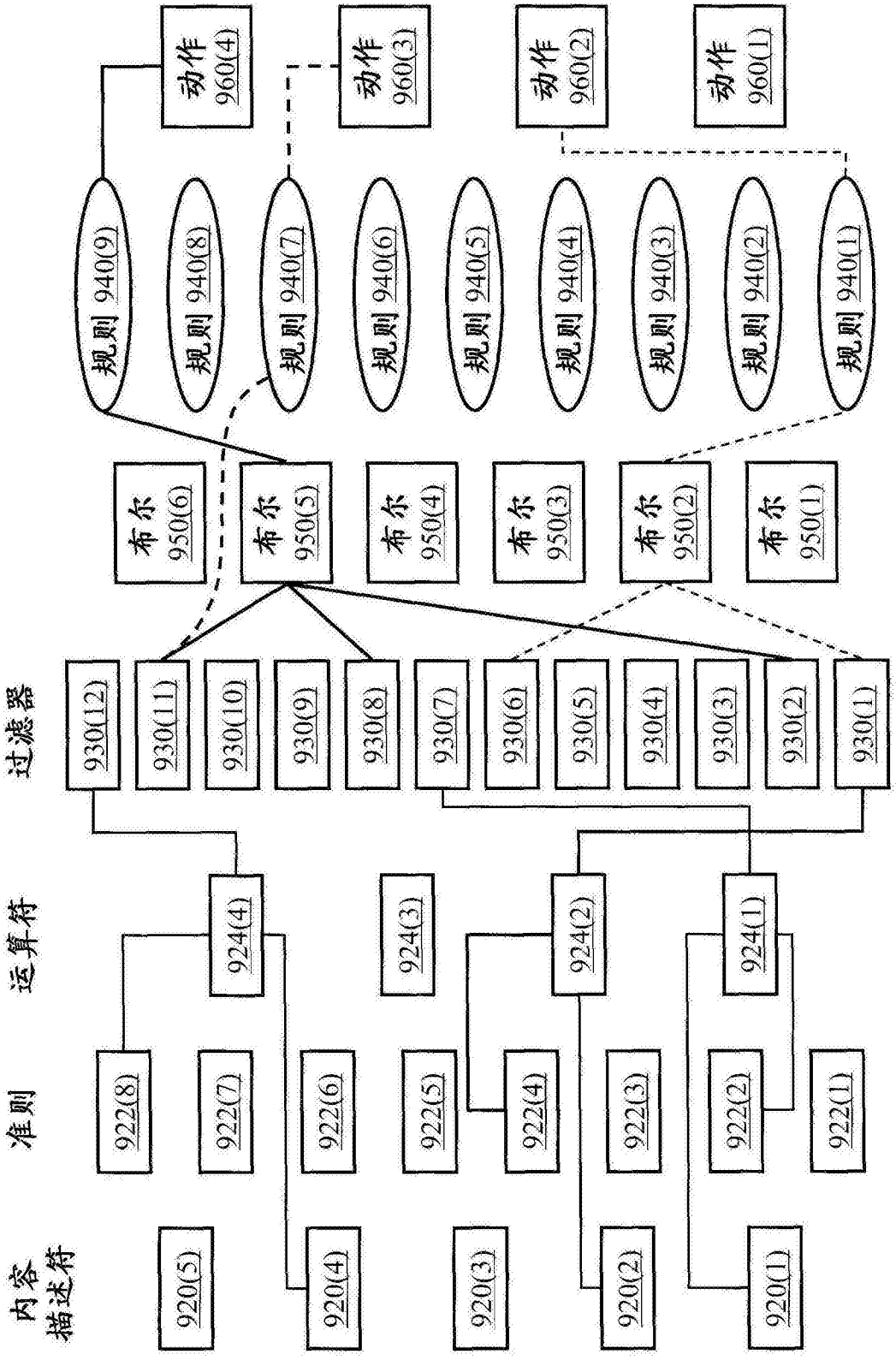

动作 960(4)
动作 960(3)
动作 960(2)
动作 960(1)
规则 940(9)
规则 940(8)
规则 940(7)
规则 940(6)
规则 940(5)
规则 940(4)
规则 940(3)
规则 940(2)
规则 940(1)
布尔 950(6)
布尔 950(5)
布尔 950(4)
布尔 950(3)
布尔 950(2)
布尔 950(1)
过滤器
930(12)
930(11)
930(10)
930(9)
930(8)
930(7)
930(6)
930(5)
930(4)
930(3)
930(2)
930(1)
运算符
924(4)
924(3)
924(2)
924(1)
准则
922(8)
922(7)
922(6)
922(5)
922(4)
922(3)
922(2)
922(1)
内容描述符
920(5)
920(4)
920(3)
920(2)
920(1)

图9

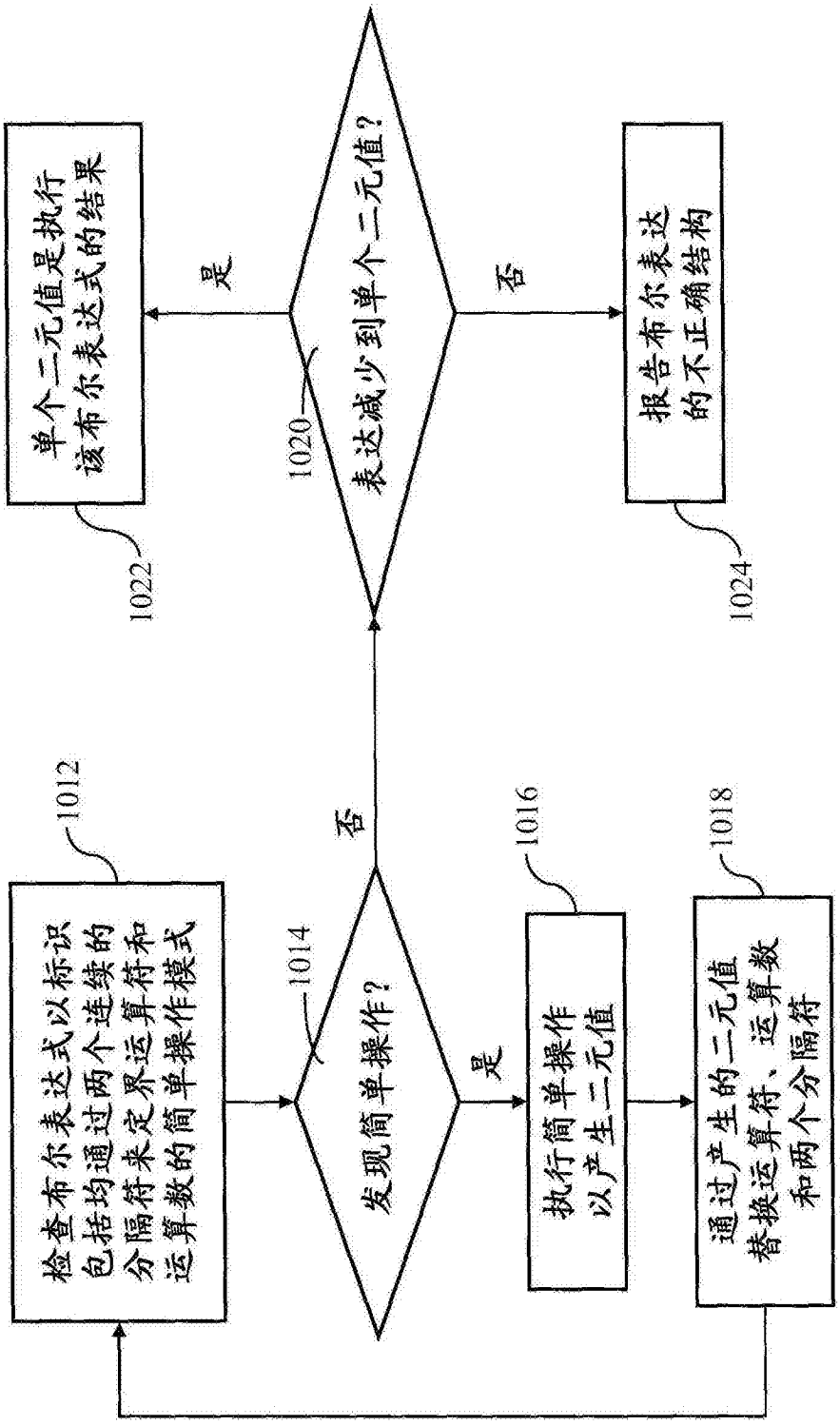

检查布尔表达式以标识包括均通过两个连续的分隔符来定界运算符和运算数的简单操作模式
1012
发现简单操作?
1014
否
是
执行简单操作以产生二元值
1016
通过产生的二元值替换运算符、运算数和两个分隔符
1018
表达减少到单个二元值?
1020
否
是
单个二元值是执行该布尔表达式的结果
1022
报告布尔表达的不正确结构
1024

图10

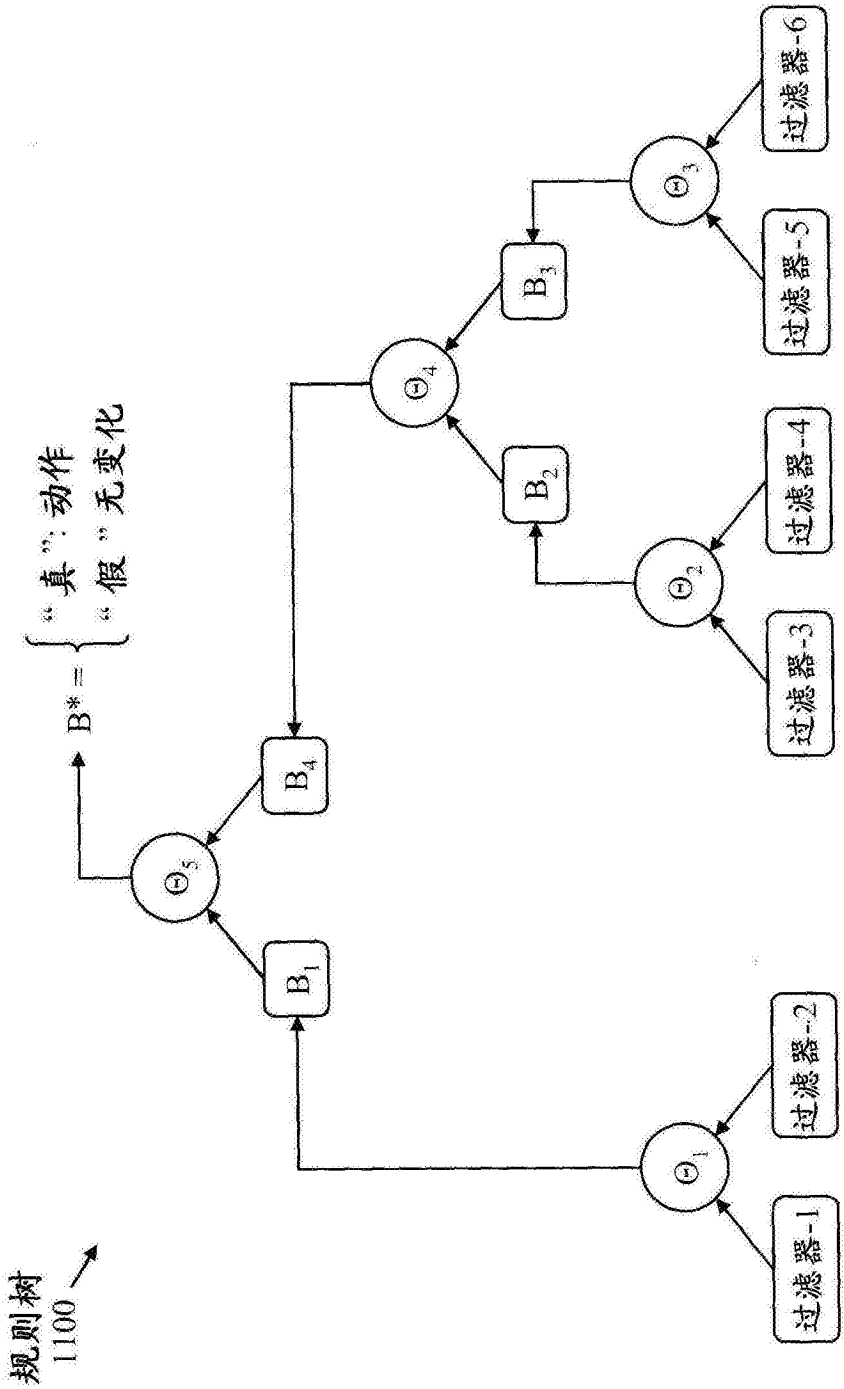


图11

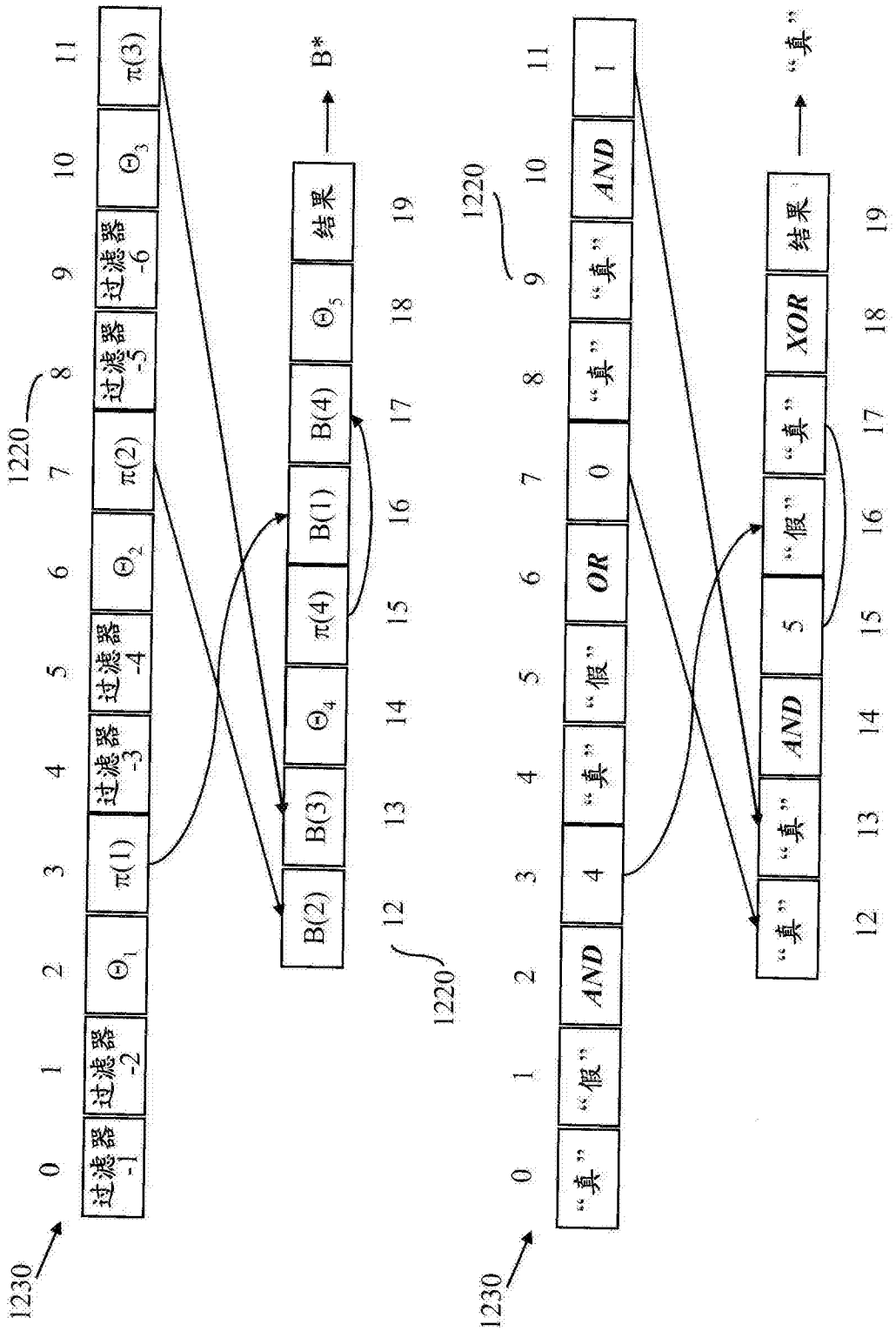

过滤器 -1
过滤器 -2
Θ1
π(1)
过滤器 -3
过滤器 -4
Θ2
π(2)
过滤器 -5
过滤器 -6
Θ3
π(3)
B(2)
B(3)
Θ4
π(4)
B(1)
B(4)
Θ5
结果
B*
1220
1230
"真"
"假"
AND
4
"真"
"假"
OR
0
"真"
"真"
AND
1
"真"
"真"
AND
5
"假"
"真"
XOR
结果
"真"

图12

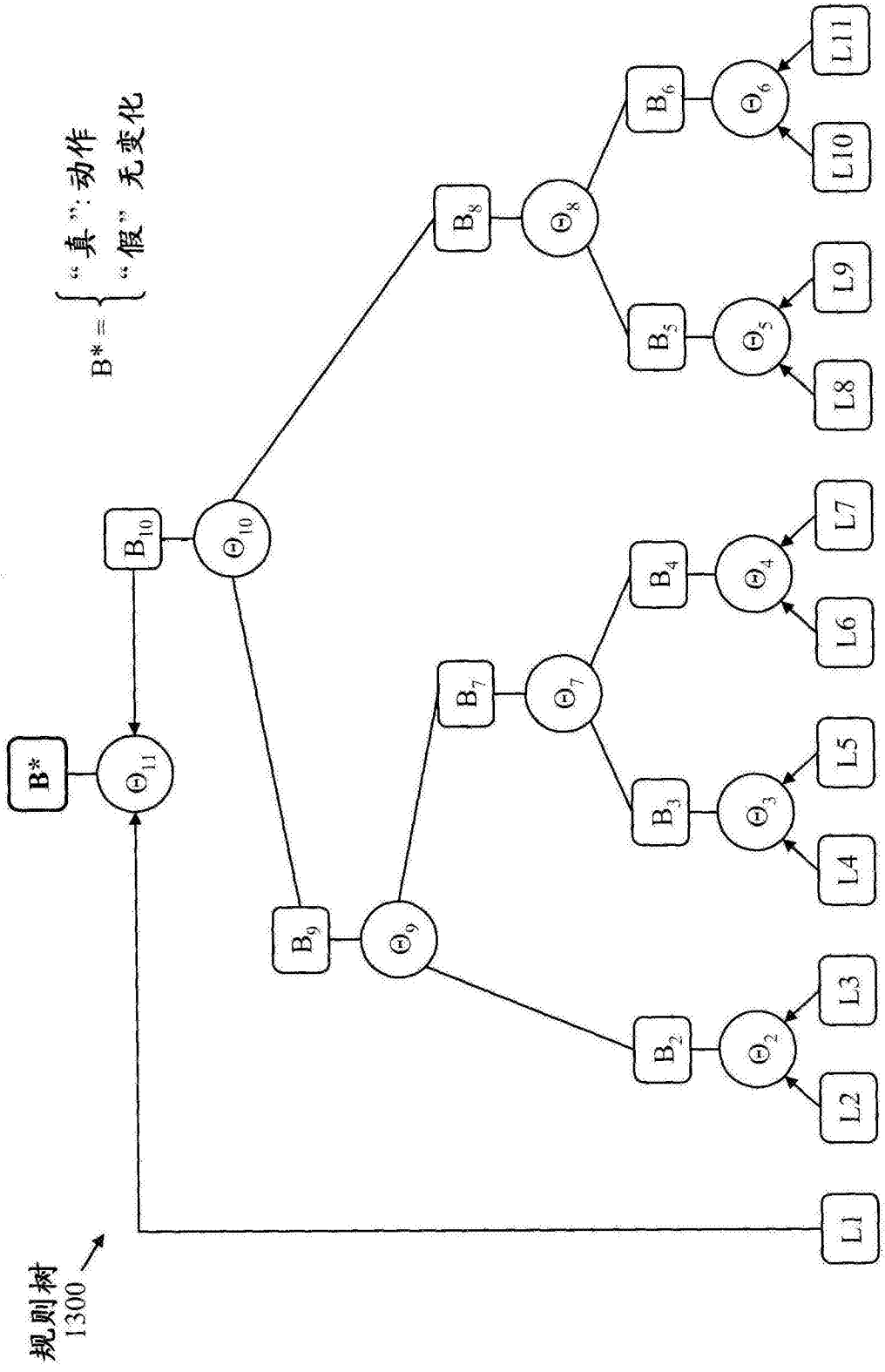

B* = { "真": 动作
"假" 无变化
规则树
1300
B*
B10
B8
B6
B5
B9
B7
B4
B3
B2
L1
L2
L3
L4
L5
L6
L7
L8
L9
L10
L11

图13

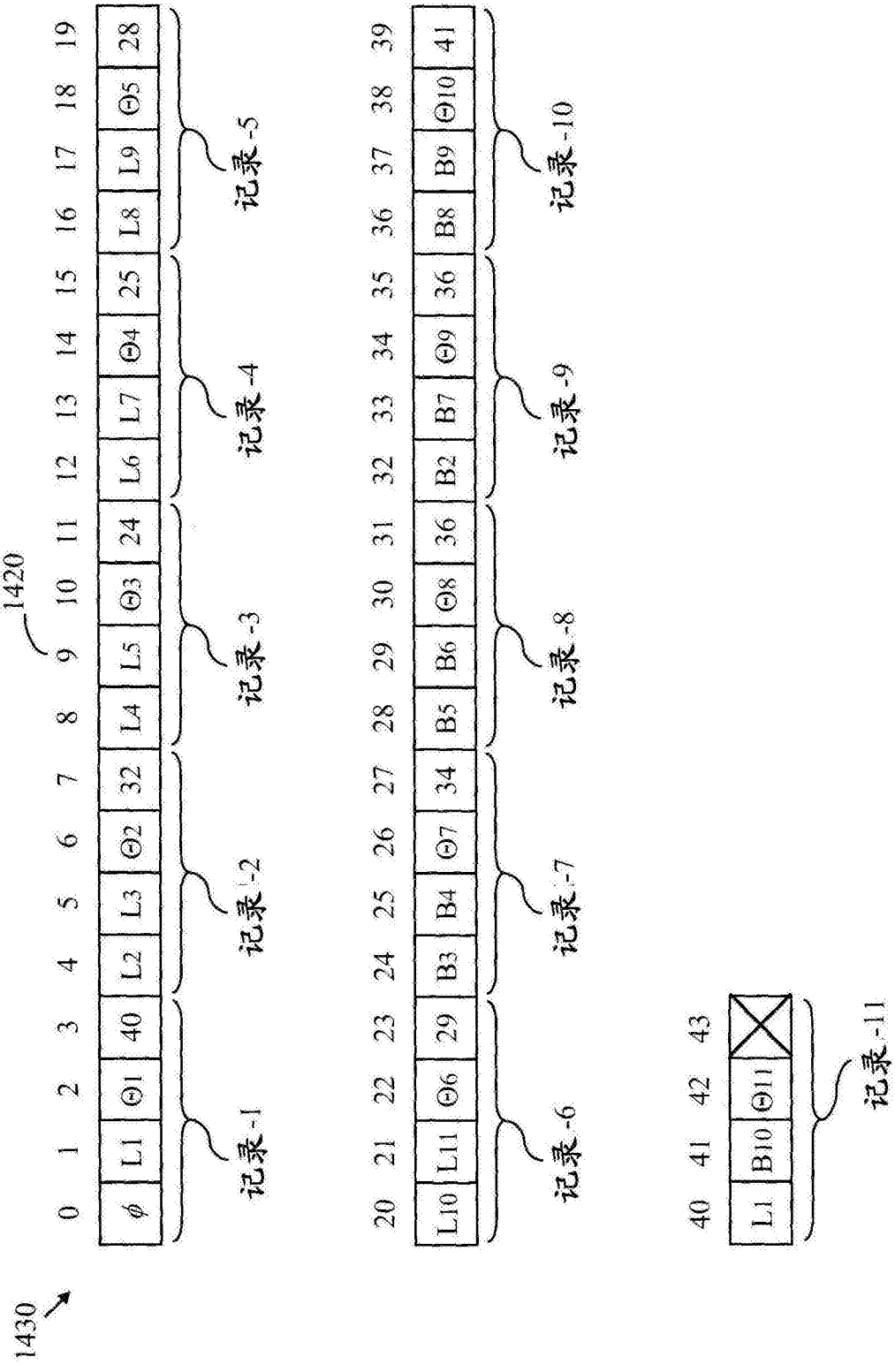

1420
1430
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
∅ L1 Θ1 40 L2 L3 Θ2 32 L4 L5 Θ3 24 L6 L7 Θ4 25 L8 L9 Θ5 28
记录-1
记录-2
记录-3
记录-4
记录-5
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
L10 L11 Θ6 29 B3 B4 Θ7 34 B5 B6 Θ8 36 B2 B7 Θ9 36 B8 B9 Θ10 41
记录-6
记录-7
记录-8
记录-9
记录-10
40 41 42 43
L1 B10 Θ11
记录-11

图14

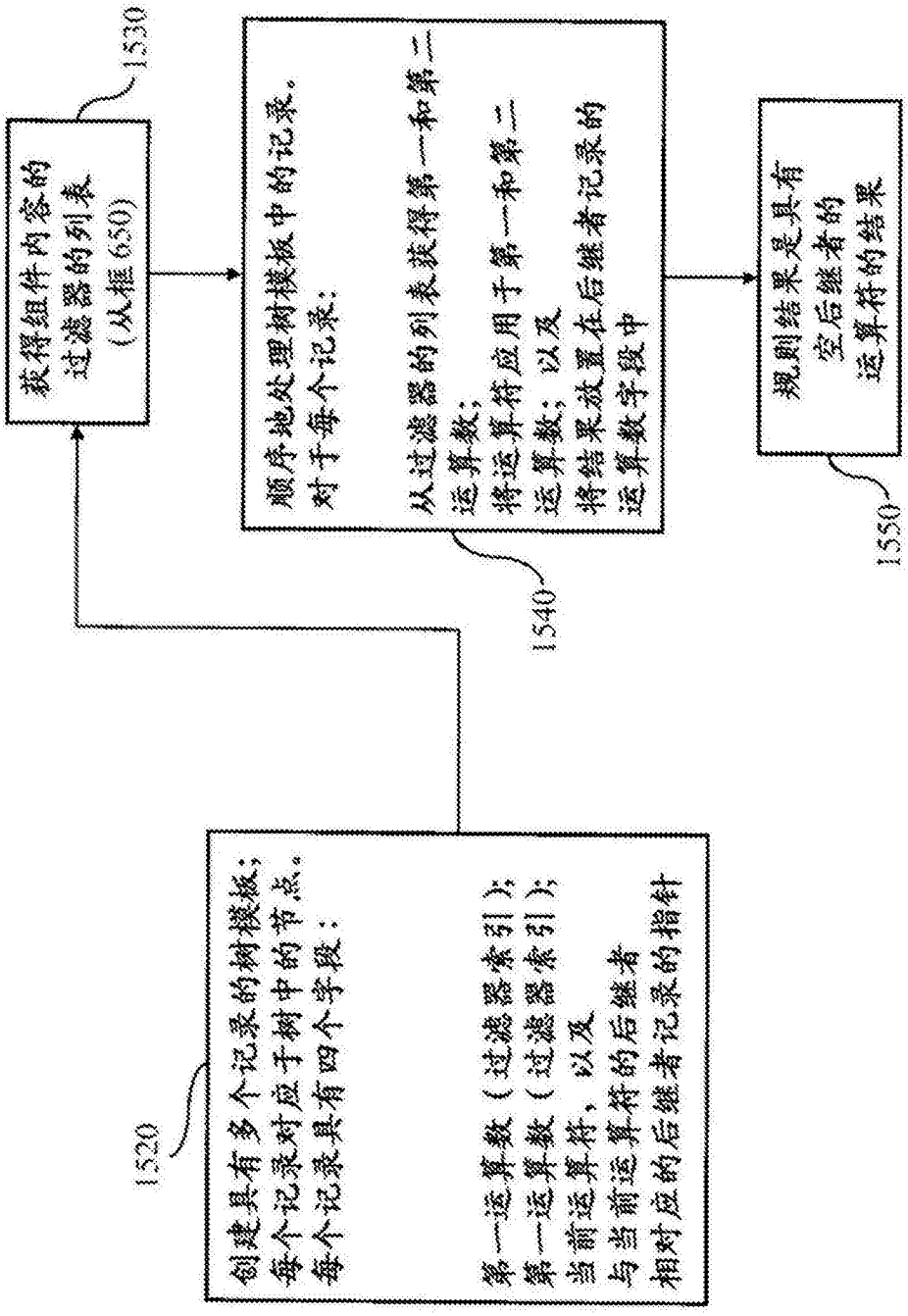

1520
创建具有多个记录的树模板；
每个记录对应于树中的节点。
每个记录具有四个字段：
第一运算数（过滤器索引）；
第二运算数（过滤器索引）；
当前运算符，以及
与当前运算符的后继者
相对应的后继者记录的指针
1530
获得组件内容的
过滤器的列表
（从框650）
顺序地处理树模板中的记录。
对于每个记录：
从过滤器的列表获得第一和第二
运算数；
将运算符应用于第一和第二
运算数；以及
将结果放置在后继者记录的
运算数字段中
1540
规则结果是具有
空后继者的
运算符的结果
1550

图15

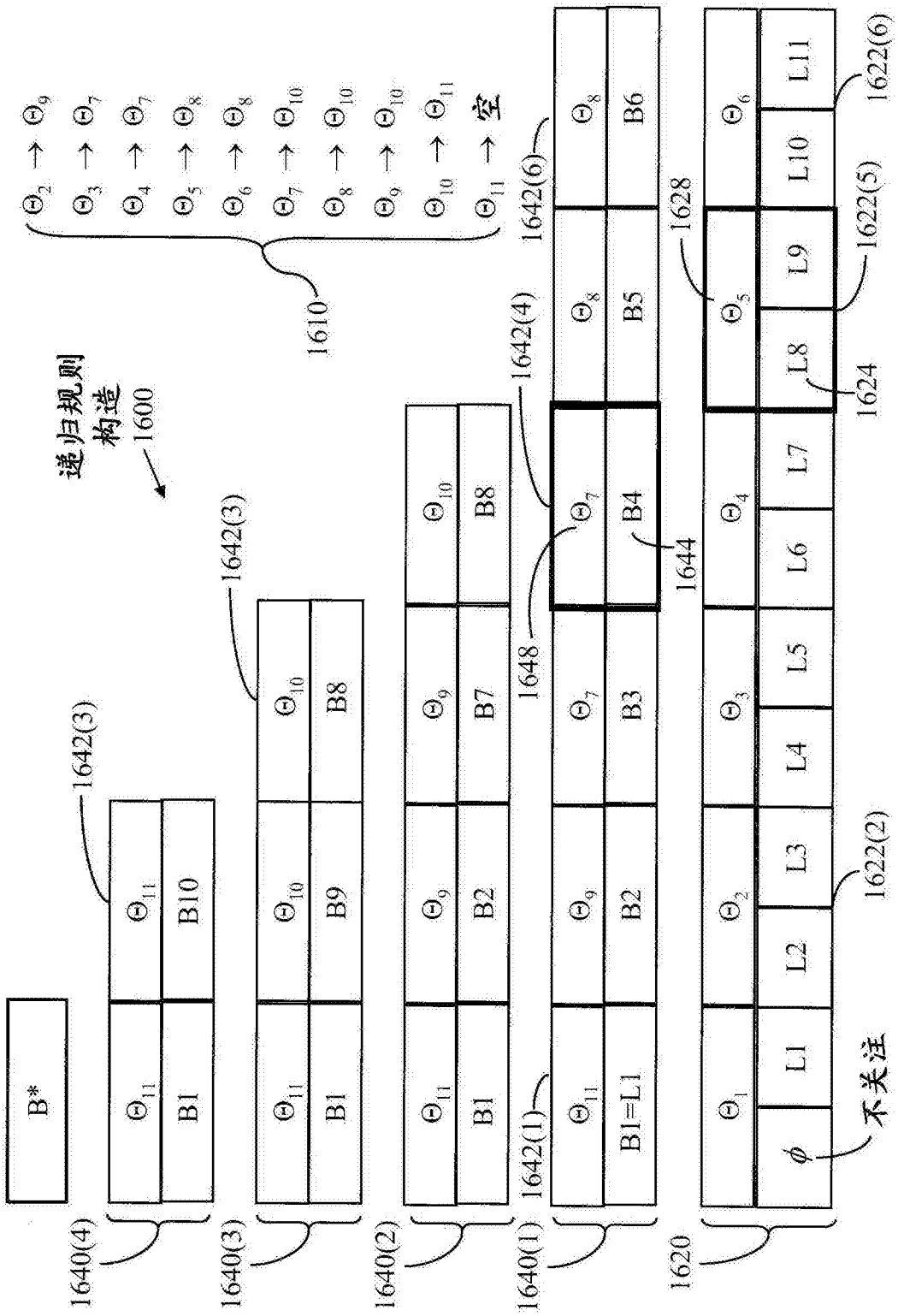

递归规则构造 1600
1610
1620
1622(2)
1622(5)
1622(6)
1624
1628
1642(1)
1642(2)
1642(3)
1642(4)
1642(6)
1644
1648
1640(1)
1640(2)
1640(3)
1640(4)
B*
B1
B10
B9
B8
B7
B6
B5
B4
B3
B2
B1=L1
L1
L2
L3
L4
L5
L6
L7
L8
L9
L10
L11
不关注
空

图16

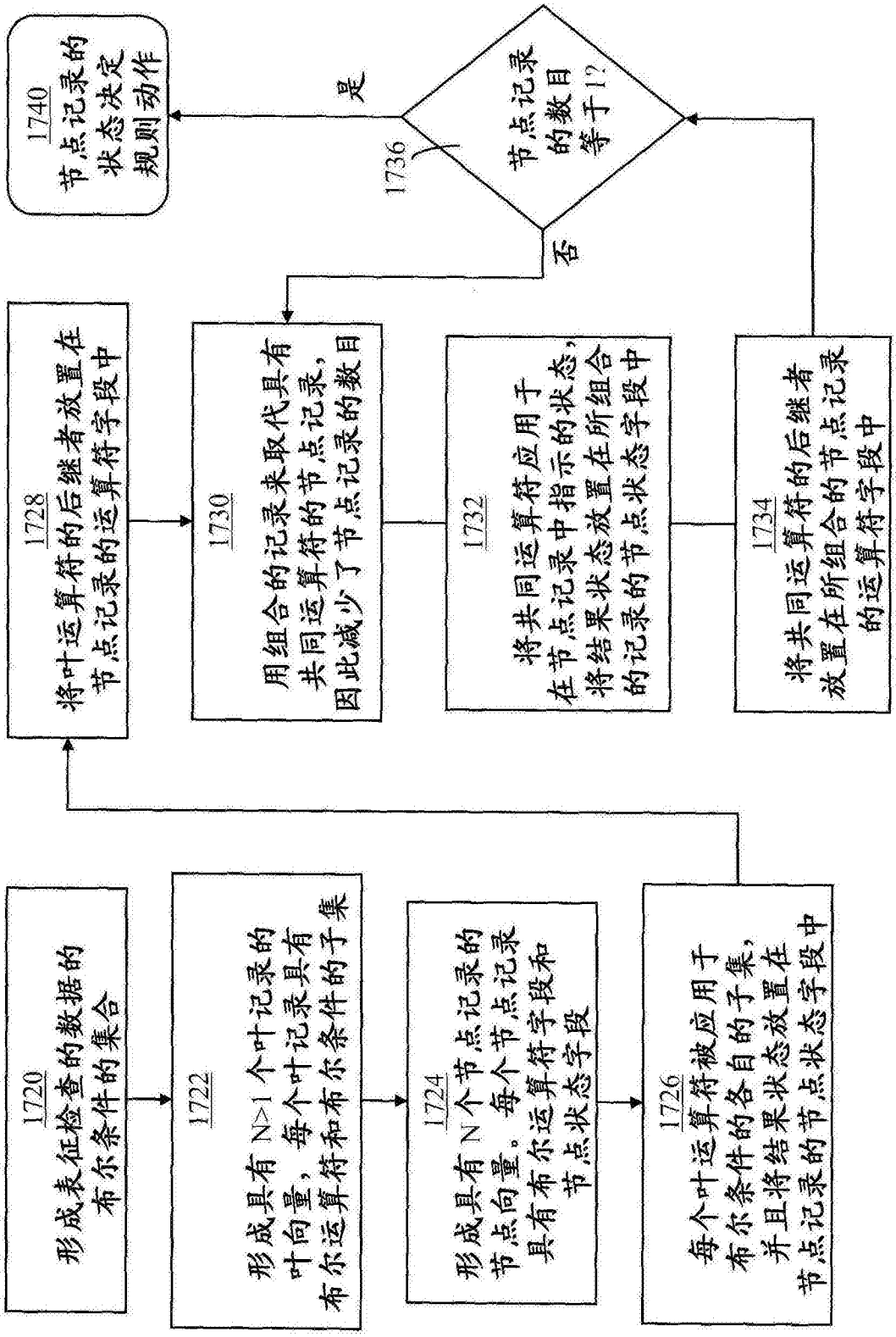

1720
形成表征检查的数据的布尔条件的集合
1722
形成具有 N>1 个叶记录的叶向量，每个叶记录具有布尔运算符和布尔条件的子集
1724
形成具有 N 个节点记录的节点向量。每个节点记录具有布尔运算符字段和节点状态字段
1726
每个叶运算符被应用于布尔条件的各自的子集，并且将结果状态放置在节点记录的节点状态字段中
1728
将叶运算符的后继者放置在节点记录的运算符字段中
1730
用组合的记录来取代具有共同运算符的节点记录，因此减少了节点记录的数目
1732
将共同运算符应用于在节点记录中指示的状态，将结果状态放置在所组合的记录的节点状态字段中
1734
将共同运算符的后继者放置在所组合的节点记录的运算符字段中
1736
节点记录的数目等于1?
是
否
1740
节点记录的状态决定规则动作

图17

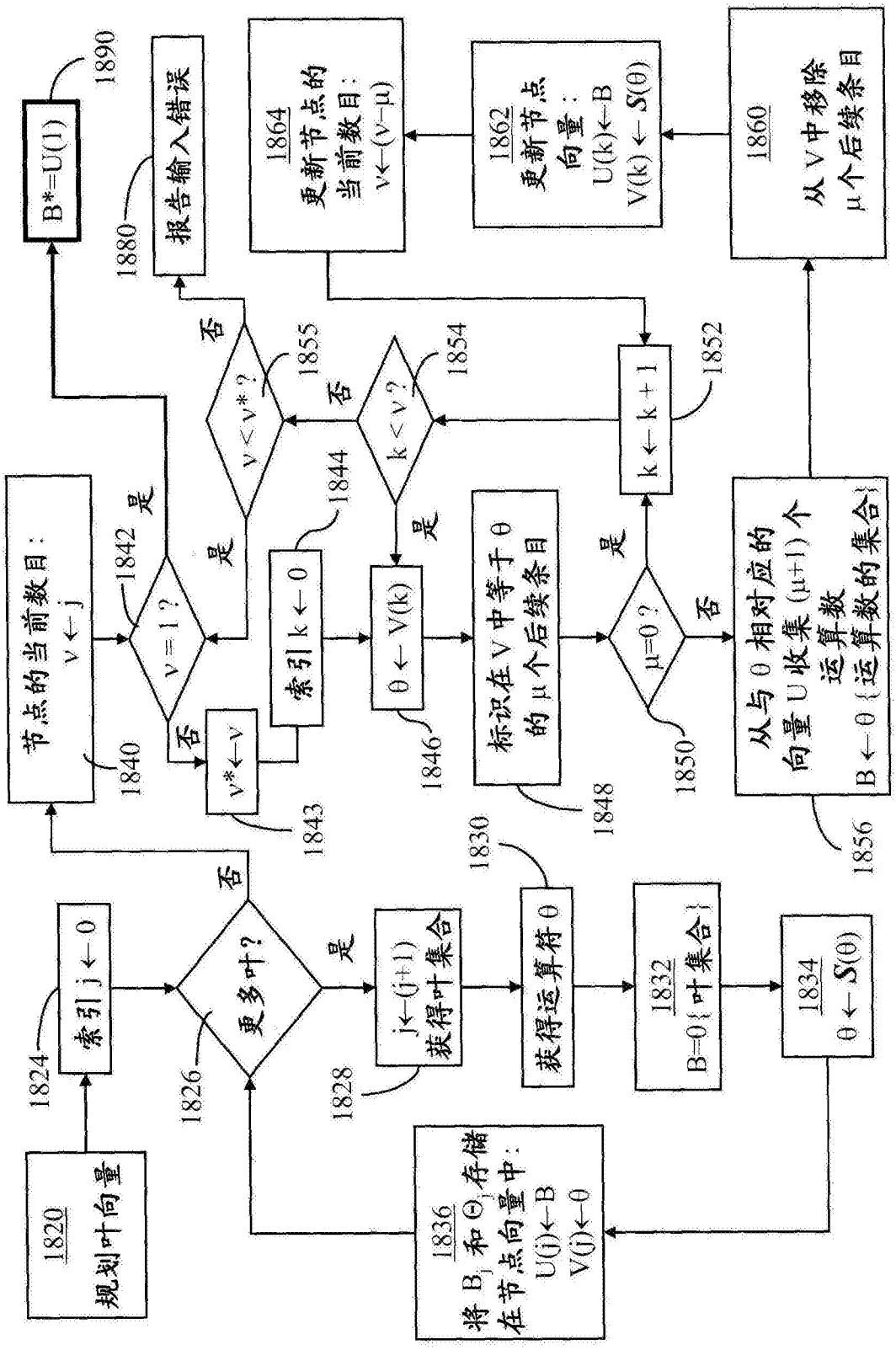

1820
规划叶向量
索引 j ← 0
更多叶?
是
否
j←(j+1)
获得叶集合
获得运算符 θ
1832
B=0{叶集合}
1834
θ ← S(θ)
1836
将 Bj 和 Θj 存储
在节点向量中:
U(j)←B
V(j)←θ
1824
1826
1828
1830
节点的当前数目:
v ← j
v = 1 ?
是
否
v*←v
索引 k ← 0
θ ← V(k)
标识在 V 中等于 θ
的 μ 个后续条目
μ=0 ?
是
否
从与 θ 相对应的
向量 U 收集 (μ+1) 个
运算数
B ← θ{运算数的集合}
k ← k + 1
k < v ?
是
否
v < v* ?
是
否
报告输入错误
B*=U(1)
1840
1842
1843
1844
1846
1848
1850
1852
1854
1855
1856
1860
从 V 中移除
μ 个后续条目
1862
更新节点
向量:
U(k)←B
V(k) ← S(θ)
1864
更新节点的
当前数目:
v←(v−μ)
1880
1890

图18

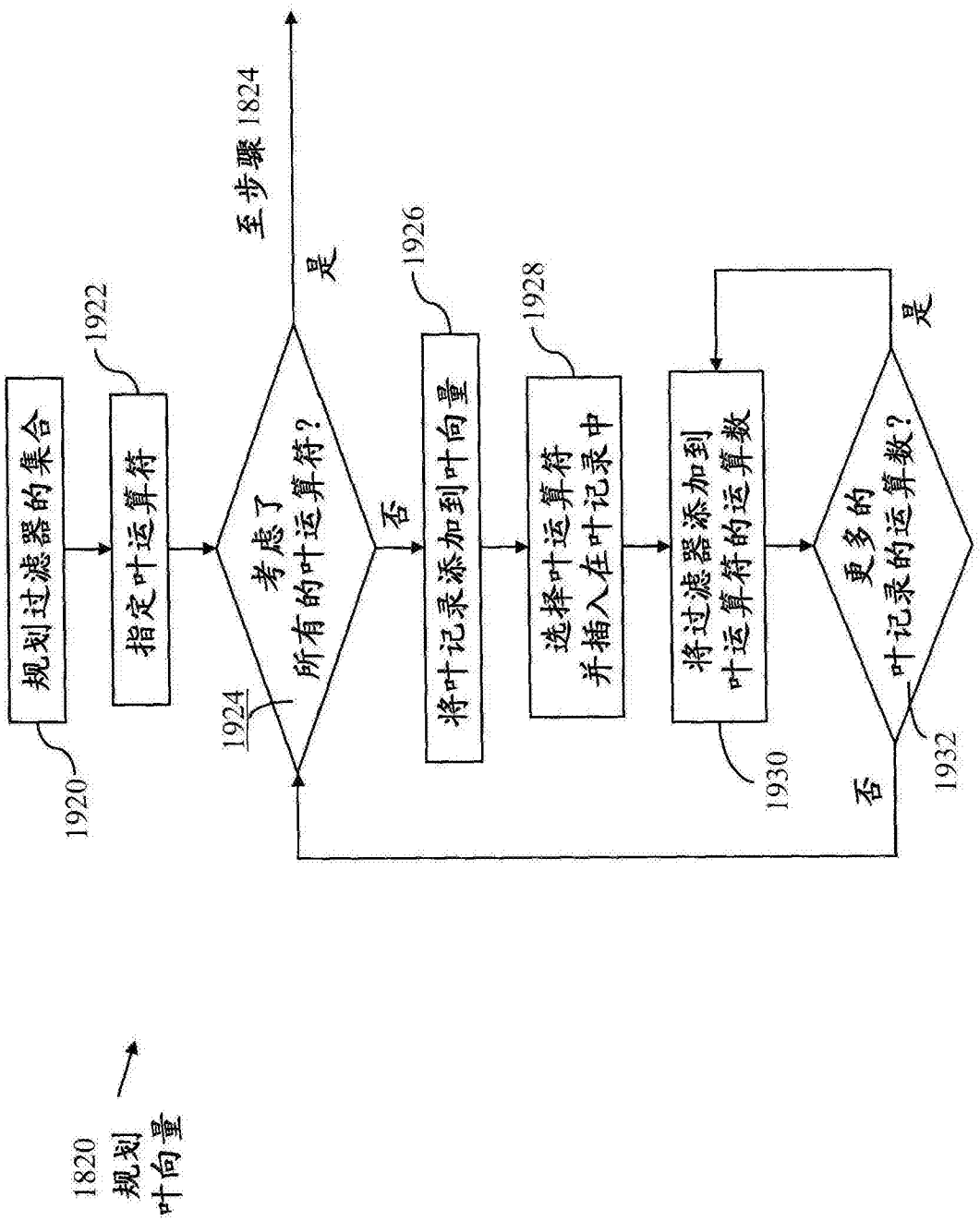

1820
规划
叶向量
规划过滤器的集合
1920
指定叶运算符
1922
考虑了
所有的叶运算符?
1924
至步骤 1824
是
否
将叶记录添加到叶向量
1926
选择叶运算符
并插入在叶记录中
1928
将过滤器添加到
叶运算符的运算数
1930
更多的
叶记录的运算数?
1932
是
否

图19

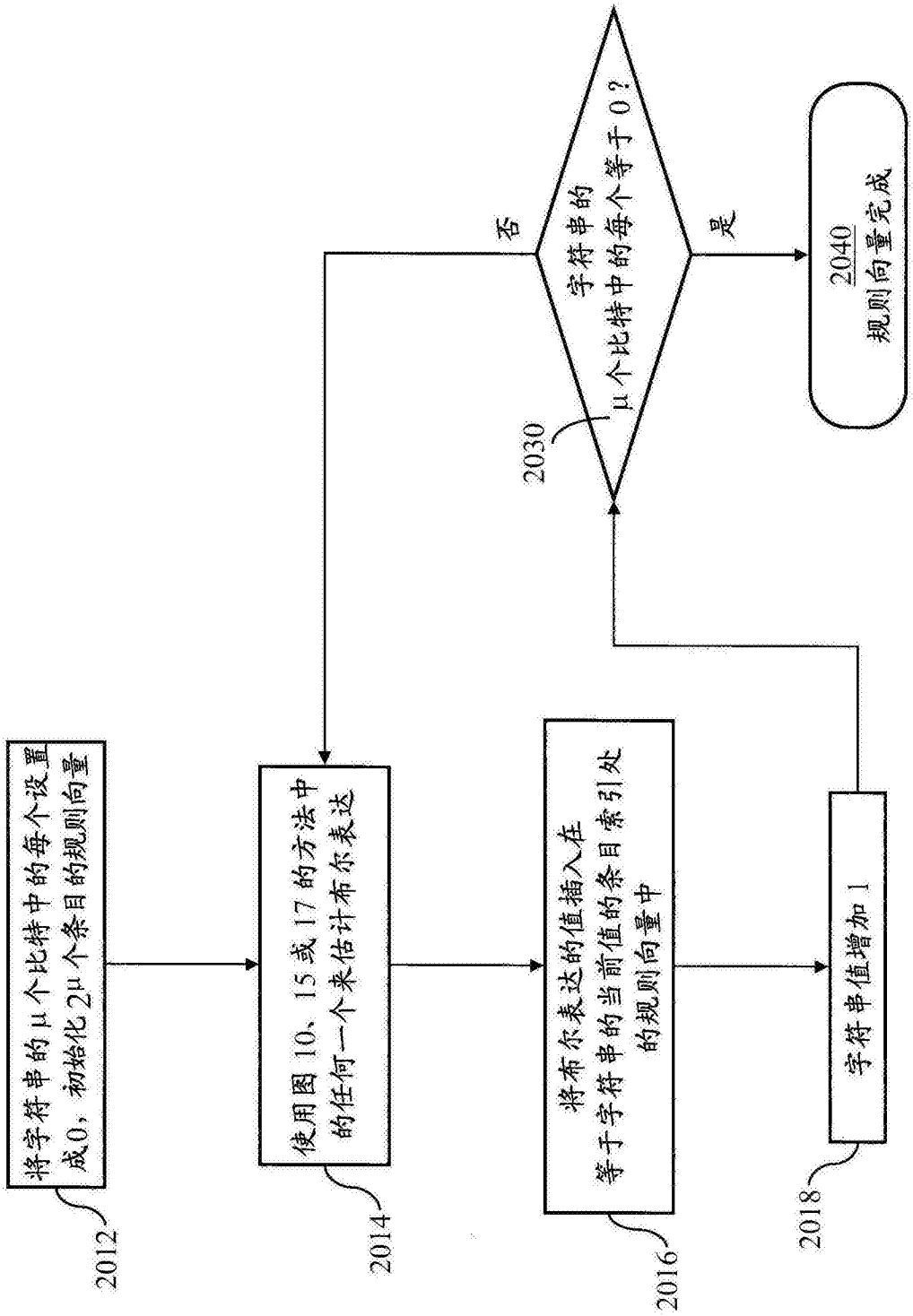

将字符串的μ个比特中的每个设置成0，初始化2^{μ}个条目的规则向量
2012
使用图10、15或17的方法中的任何一个来估计布尔表达
2014
将布尔表达的值插入在等于字符串的当前值的条目索引处的规则向量中
2016
字符串值增加1
2018
字符串的μ个比特中的每个等于0？
2030
否
是
2040
规则向量完成

图20

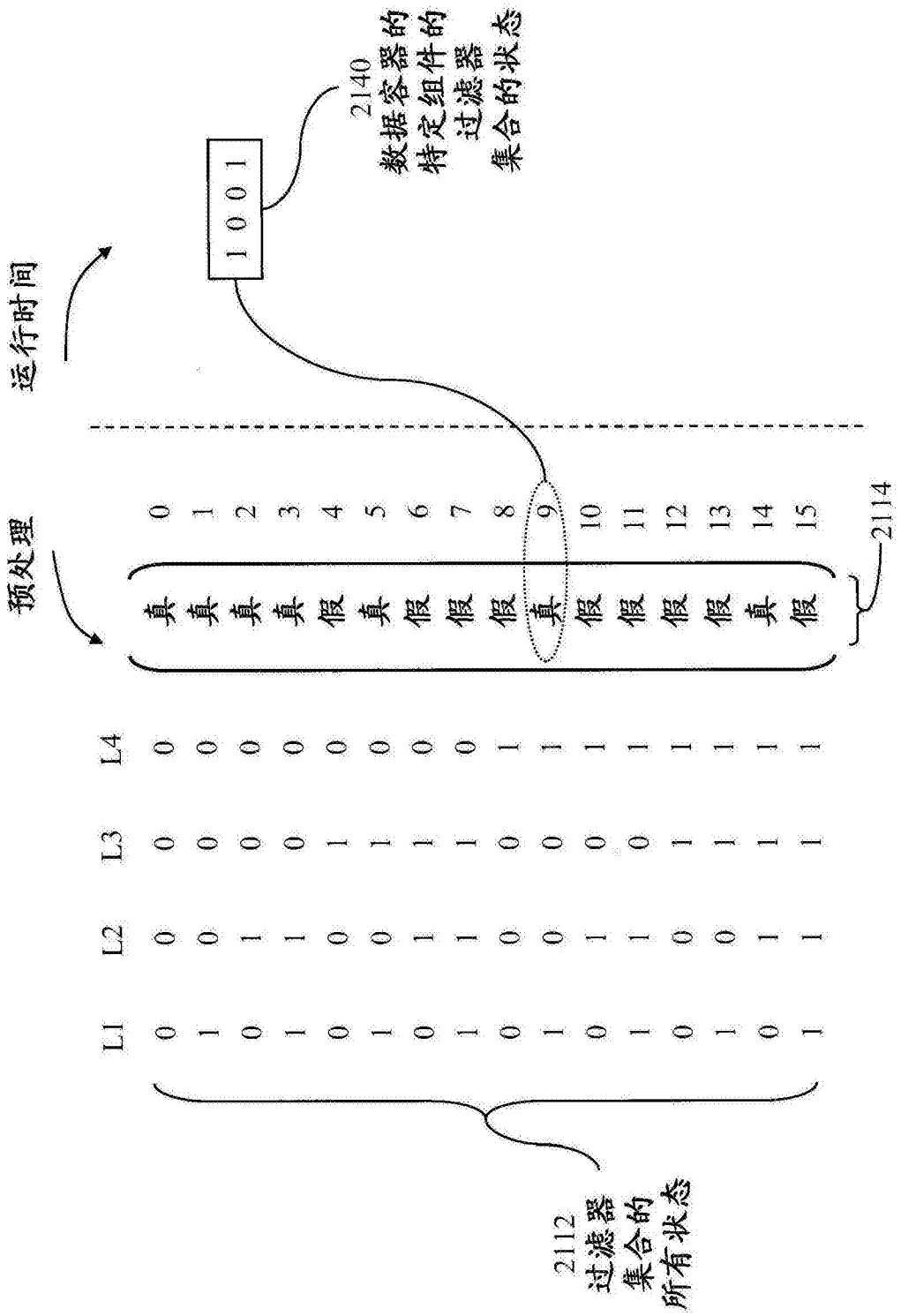

运行时间
预处理
1 0 0 1
2140
数据容器的
特定组件的
过滤器
集合的状态
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
真 真 真 真 假 真 假 假 假 真 假 假 假 假 真 假
2114
L4 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
L3 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
L2 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
L1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
2112
过滤器
集合的
所有状态

图21

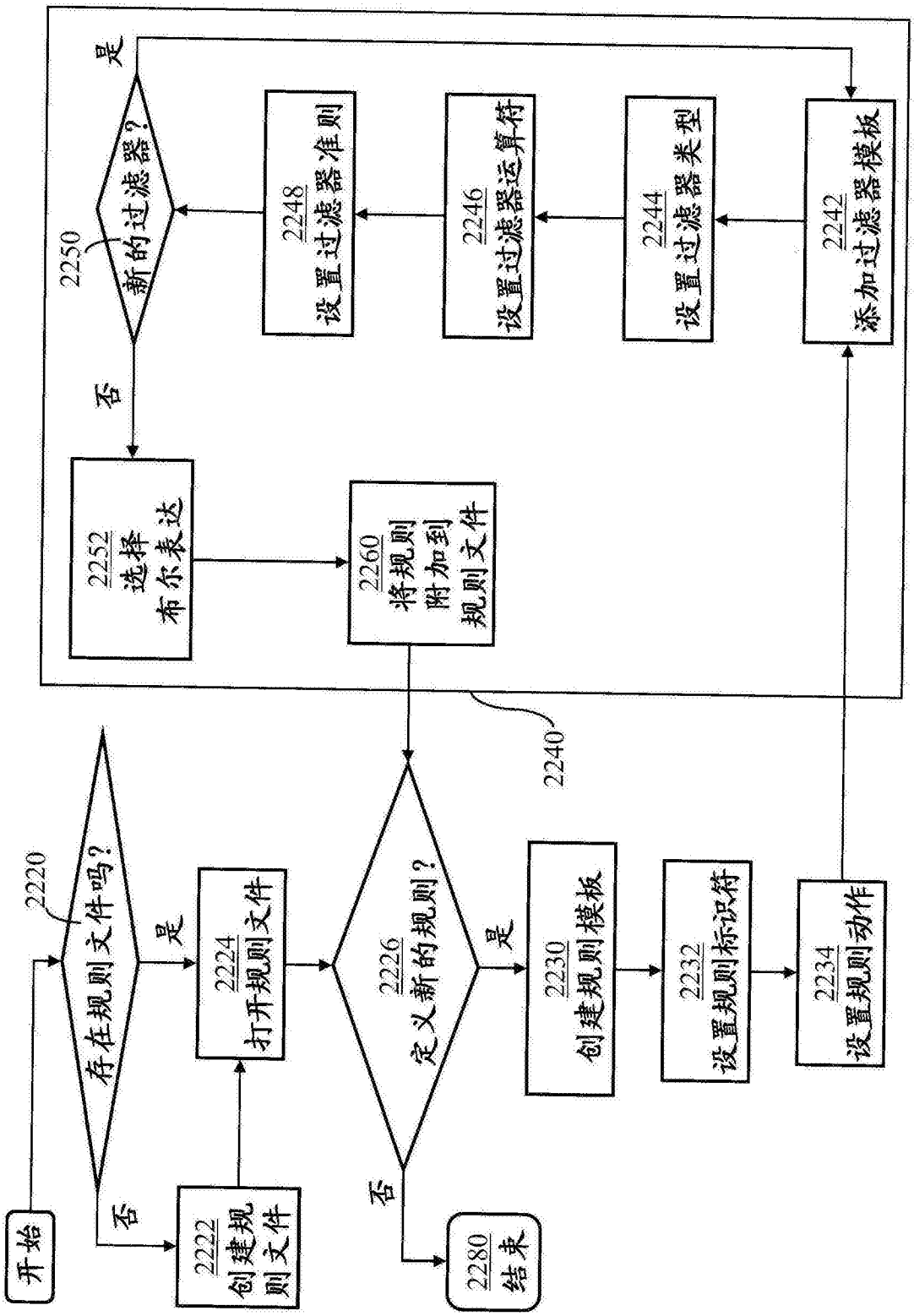

开始
2220
存在规则文件吗?
是
否
2222
创建规则文件
2224
打开规则文件
2226
定义新的规则?
是
否
2280
结束
2230
创建规则模板
2232
设置规则标识符
2234
设置规则动作
2240
2242
添加过滤器模板
2244
设置过滤器类型
2246
设置过滤器运算符
2248
设置过滤器准则
2250
新的过滤器?
是
否
2252
选择
布尔表达
2260
将规则
附加到
规则文件

图22

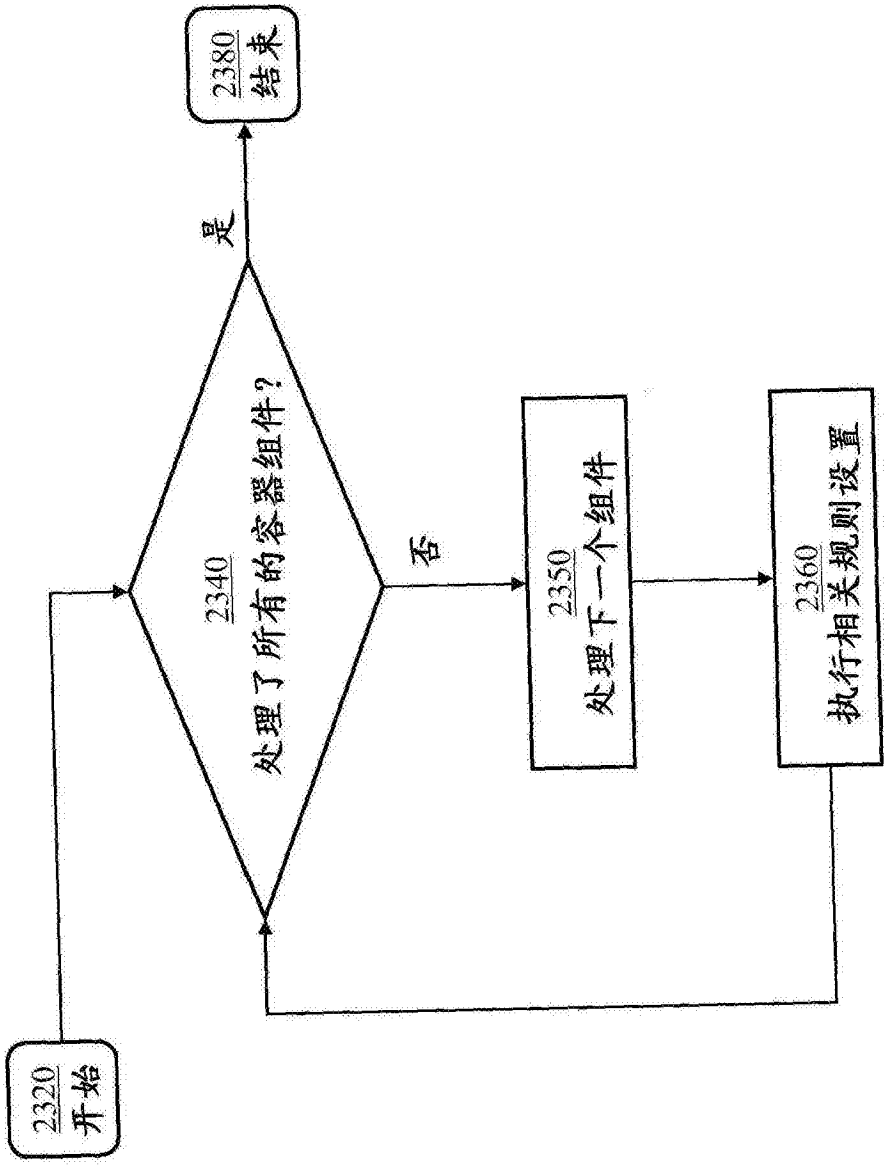

2320
开始
2340
处理了所有的容器组件?
是
2380
结束
否
2350
处理下一个组件
2360
执行相关规则设置

图23

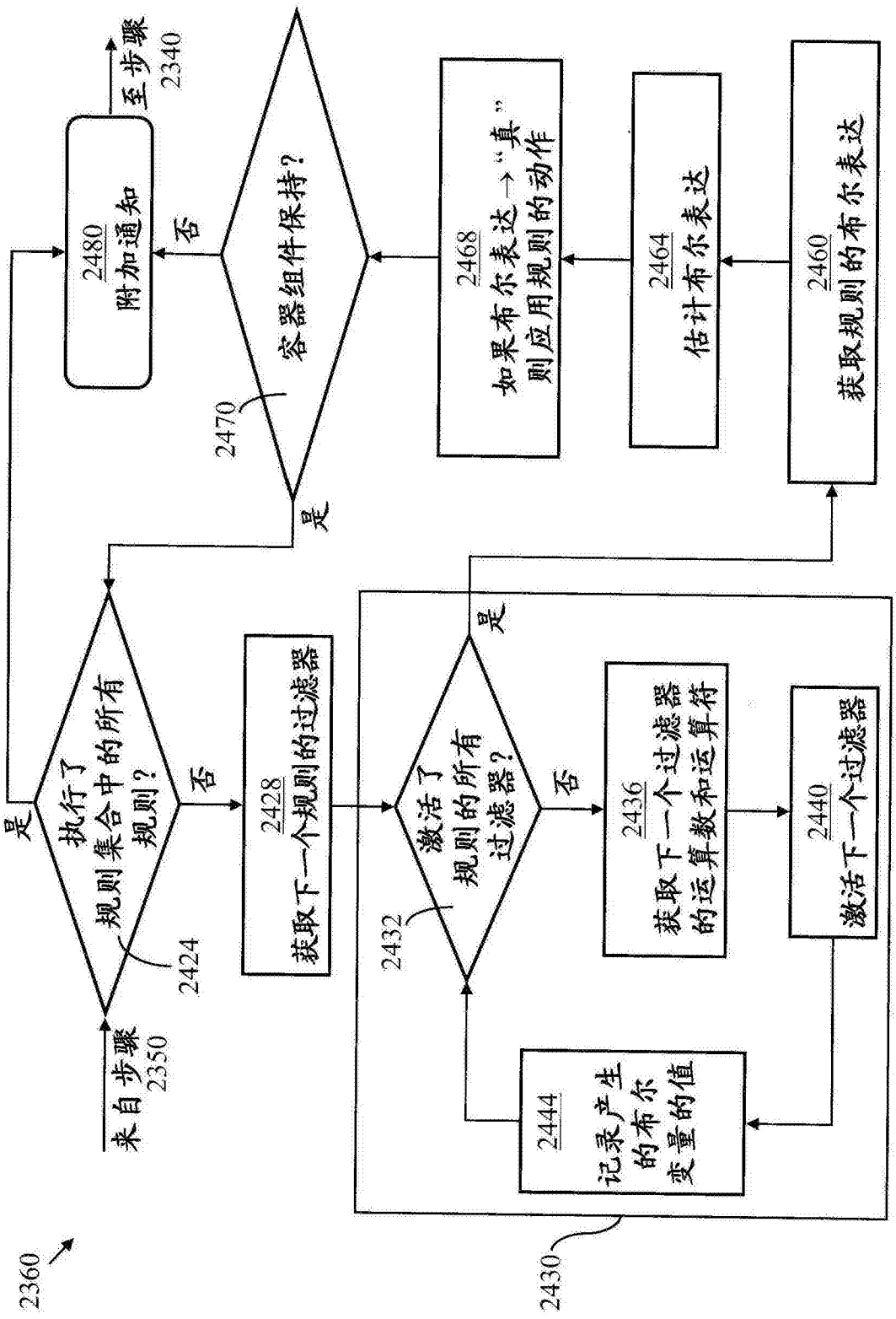

2360
来自步骤 2350
执行了规则集合中的所有规则？
2424
是
否
2428 获取下一个规则的过滤器
激活了规则的所有过滤器？
2432
是
否
2430
2436 获取下一个过滤器的运算数和运算符
2440 激活下一个过滤器
2444 记录产生的布尔变量的值
2460 获取规则的布尔表达
2464 估计布尔表达
2468 如果布尔表达→“真”则应用规则的动作
容器组件保持？
2470
是
否
2480 附加通知
至步骤 2340

图24

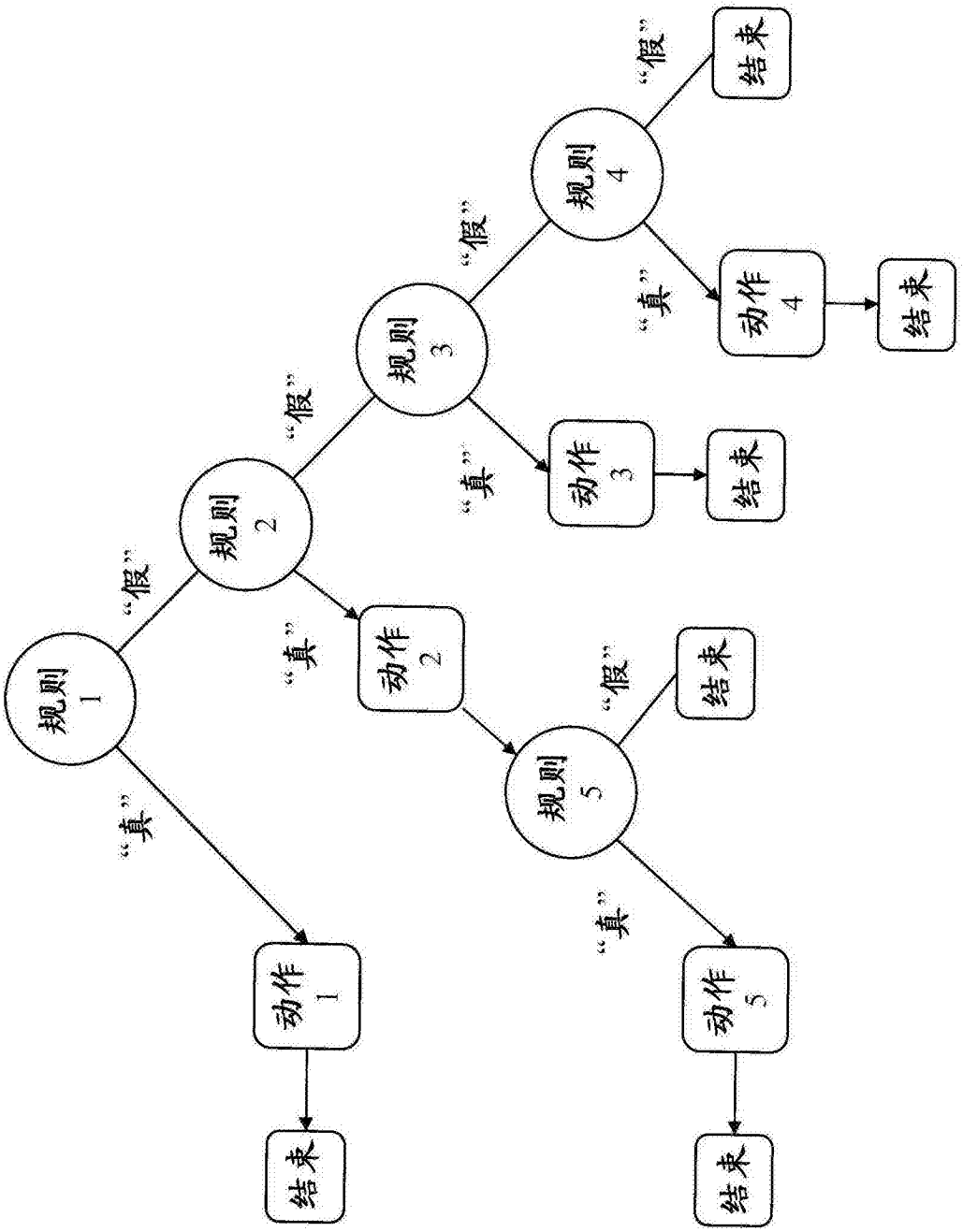

规则 1
“假”
“真”
规则 2
“假”
“真”
规则 3
“假”
“真”
规则 4
“假”
“真”
规则 5
“假”
“真”
动作 1
动作 2
动作 3
动作 4
动作 5
结束
结束
结束
结束
结束
结束

图25

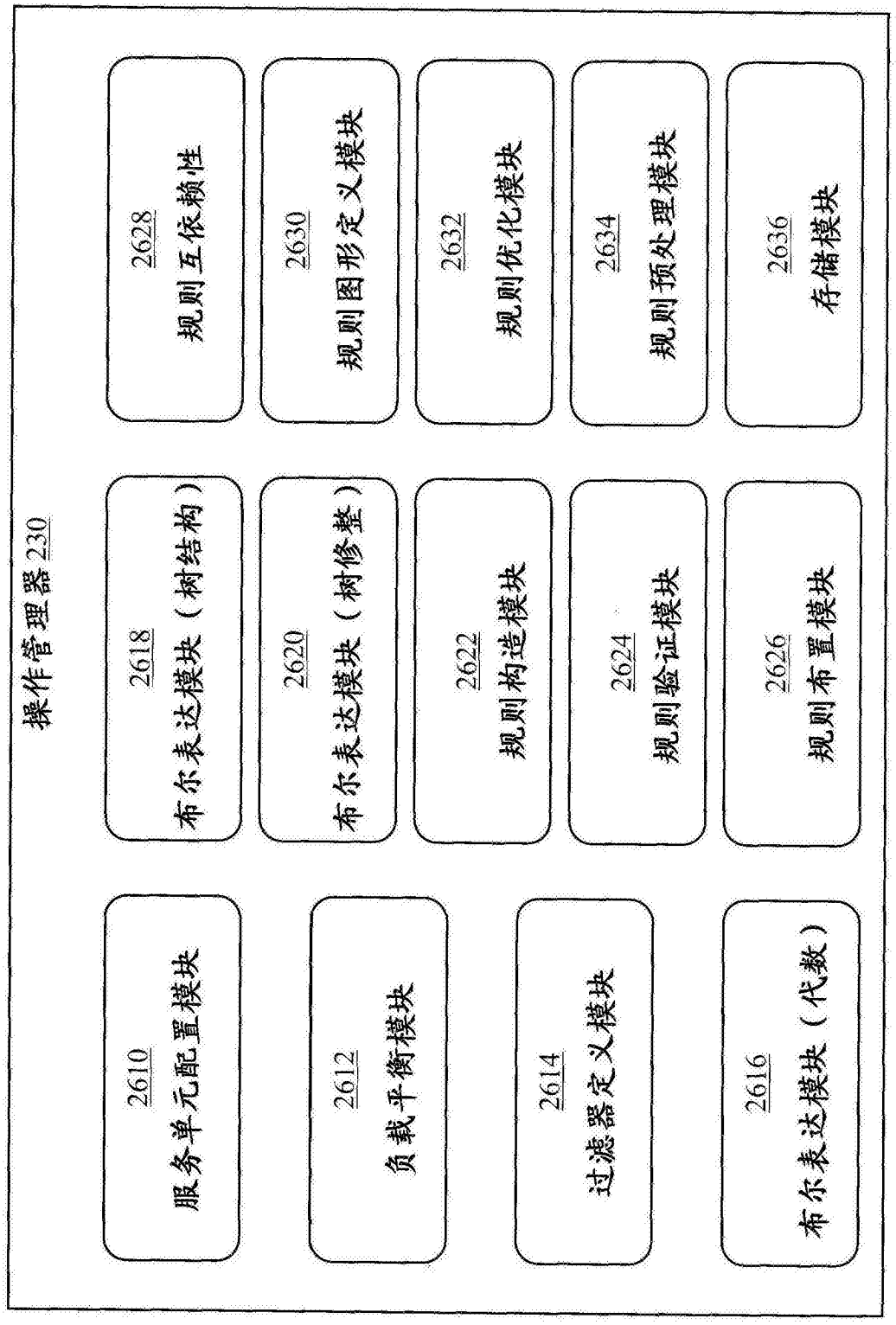

操作管理器 230
2610
服务单元配置模块
2612
负载平衡模块
2614
过滤器定义模块
2616
布尔表达模块（代数）
2618
布尔表达模块（树结构）
2620
布尔表达模块（树修整）
2622
规则构造模块
2624
规则验证模块
2626
规则布置模块
2628
规则互依赖性
2630
规则图形定义模块
2632
规则优化模块
2634
规则预处理模块
2636
存储模块

图26

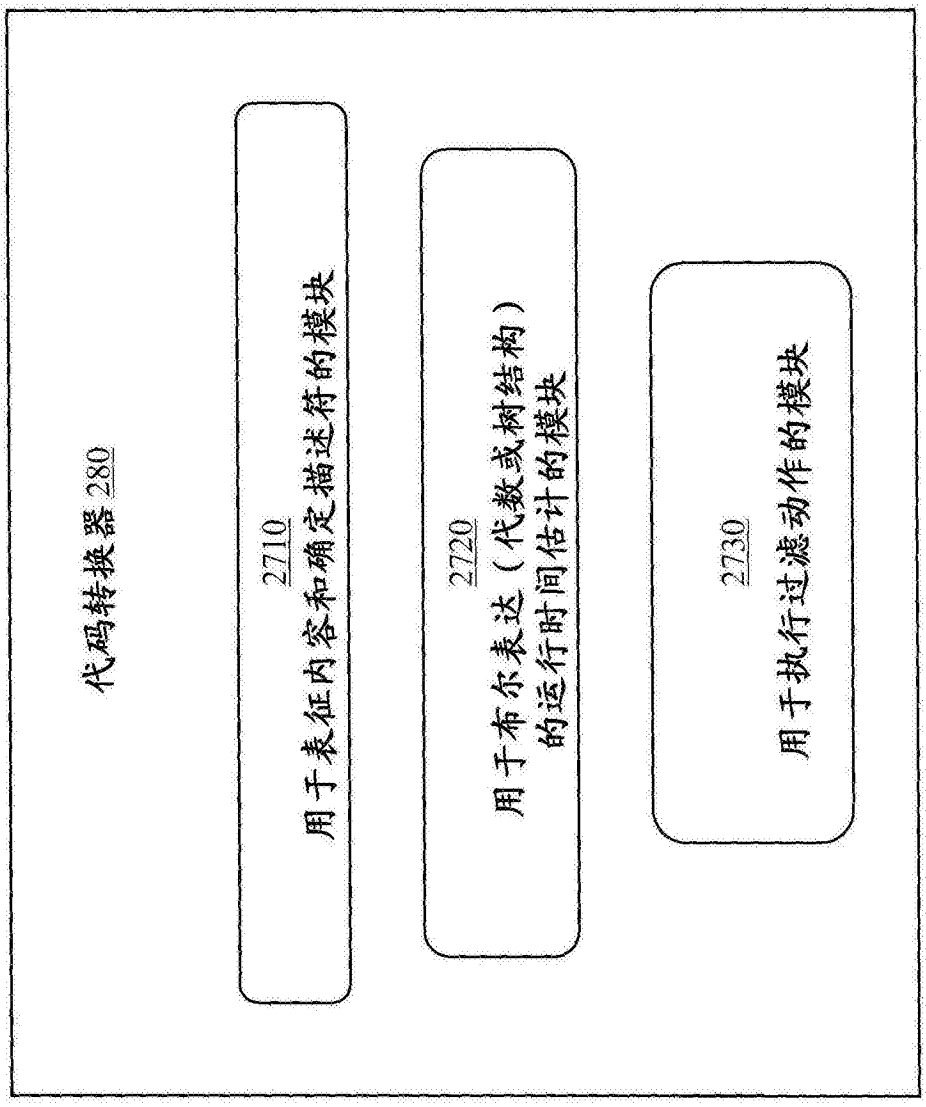

代码转换器 280
2710
用于表征内容和确定描述符的模块
2720
用于布尔表达（代数或树结构）的运行时间估计的模块
2730
用于执行过滤动作的模块

图27

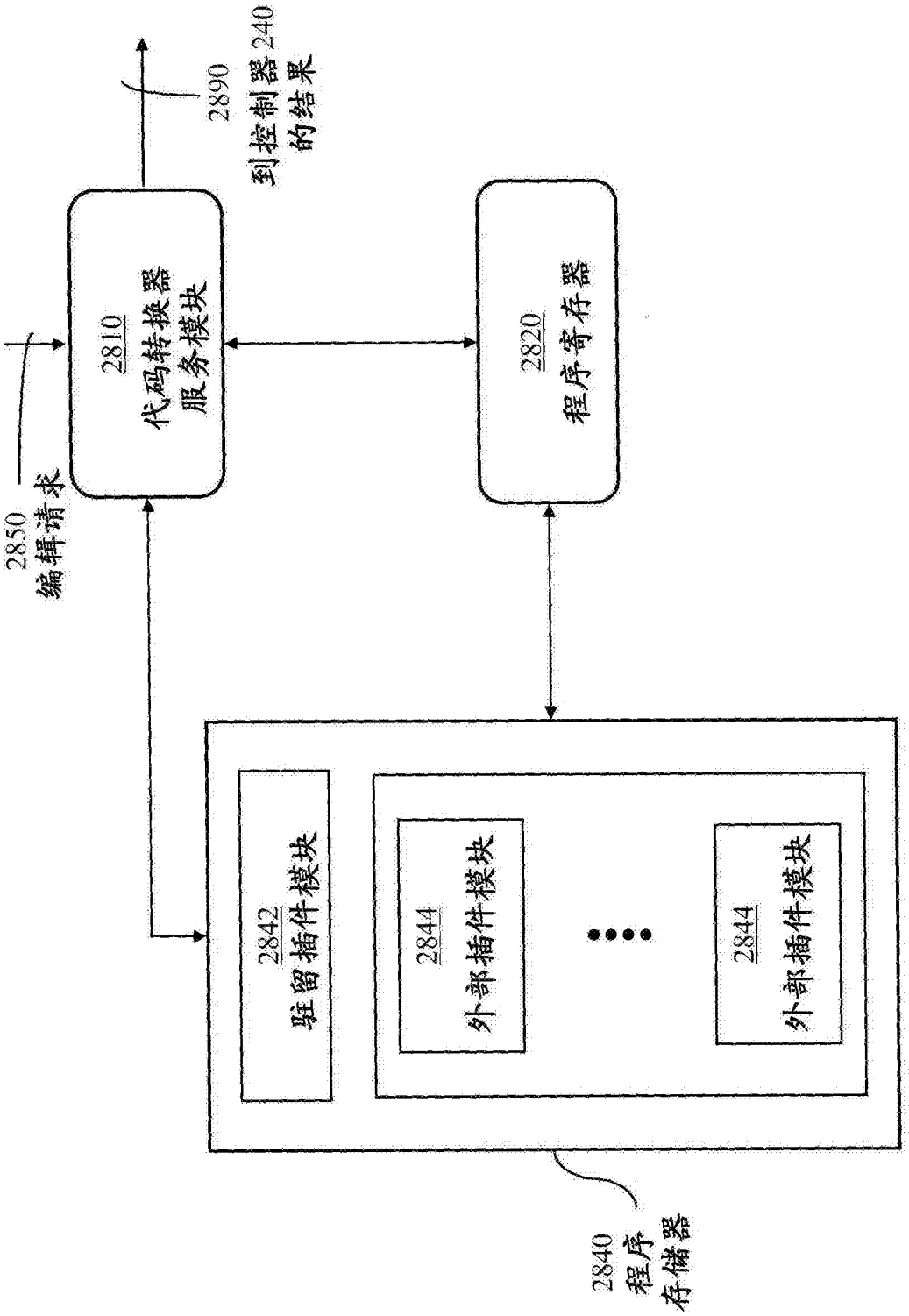

2890
到控制器 240
的结果
2810
代码转换器
服务模块
2820
程序寄存器
2850
编辑请求
2842
驻留插件模块
2844
外部插件模块
2844
外部插件模块
2840
程序
存储器

图28