[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H04L 12/24
H04Q 3/00 G06F 17/00
//H04L29/06

[12] 发明专利说明书

[21] ZL 专利号 95197181.6

[43] 授权公告日 2003 年 2 月 19 日

[11] 授权公告号 CN 1102001C

[22] 申请日 1995.12.29 [21] 申请号 95197181.6
[30] 优先权
[32] 1994.12.30 [33] FI [31] 946209
[86] 国际申请 PCT/FI95/00719 1995.12.29
[87] 国际公布 WO96/21324 英 1996.7.11
[85] 进入国家阶段日期 1997.6.28
[71] 专利权人 诺基亚电信公司
地址 芬兰埃斯波
[72] 发明人 奥里·芬尼
审查员 焦景梅

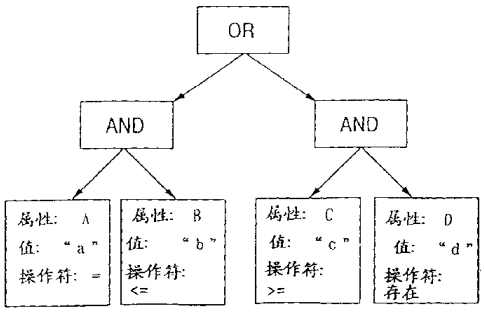
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 李德山

权利要求书 1 页 说明书 12 页 附图 3 页

[54] 发明名称 网络单元中比较可控对象表示式属性值的方法

[57] 摘要

本发明涉及一种为实施于通信网网络单元(NE)的操作选择目标群的方法。这种方法包括以下步骤：(a)从网络管理单元接收关于操作和第一目标群(S)的信息，第一目标群(S)表示为包含在网络单元存储器中的一群对象实例，以及对象实例属性的比较准则，该准则包含有参考值，(b)比较属于第一目标群的对象实例的属性值和所述参考值，(c)从第一目标群(S)中选出属性满足所接收的比较准则的那些对象实例作为最终目标群。为了实现通用的方法，单个对象实例间的属性比较由下列步骤完成：(i)读入事先存储在网络单元存储器中并且与该属性关联的语义数据，(ii)在网络单元中搜索该对象实例的关联属性值，以及(iii)利用读取的语义数据比较属性值和接收到的参考值。



ISSN1008-4274

1. 一种为实施于通信网网络单元(NE)的操作选择目标群的方法，这种方法包括步骤:

— 从网络管理单元获取关于操作和第一目标群（S）的信息，第一目标群（S）表示为包含在网络单元存储器中的一组对象实例，以及对象实例属性的比较准则，该准则包含有参考值，

— 比较属于第一目标群的对象实例的属性值和所述参考值，以及

— 从第一目标群(S)中选出那些属性值满足所收到的比较准则的对象实例作为最终目标群，

其特征在于:

— 对象实例的属性通过下述方式进行比较

— 读入事先存储在网络单元存储器中并且与该属性关联的语义数据，

— 在网络单元中搜索该对象实例的关联属性值，

— 利用读取的语义数据比较属性值和接收到的参考值。

2. 根据权利要求1中所述的方法，其特征在于，语义数据以树状分层结构存储在网络单元的存储器中。

3. 根据权利要求2中所述的方法，其特征在于，语义树节点至少包括:（a）对象实例属性的标识符，和（b）关于在所述属性值比较中要使用什么数据类型的信息。

网络单元中比较可控对象表示式属性值的方法

技术领域

本发明一般涉及通信网的管理系统。本发明尤其涉及一种为实施于通信网网络单元(NE)的操作选择目标群的方法。

背景技术

上面提到的网络管理系统在实际应用中可能是图 1 所示的类型。位于网络操作中心 OC 的网络操作员使用网络管理工作站 WS，这些工作站连接到单独的工作站网络 WSN，例如，Ethernet 网。通常情况下，网络管理系统分布在工作站网络的几台计算机上，它们可以接入到包含有网络管理所需信息的数据库 DB。管理系统通过在国际标准中定义的 Q3 接口连接到网络，例如移动网络 MN ，移动网络的网络单元包括：移动业务交换中心 MSC，基站控制器 BSC，基站 BTS，以及移动站。它们与所管理网络的连接通过数据通信网 DCN 来建立。所管理的网络同样可以是任何通信网，例如，混合的 SDH/PDH 网络。

在网络管理中 Q3 接口的 OSI 通信中使用的 CMISE 应用服务要素 (CMISE = 公共管理信息服务要素）为应用进程提供了 SCOPE/FILTER 功能（参看 ISO/IEC-9596-1 信息技术 —开放系统互连 —公共管理信息协议 — 第一部分：说明书)。该功能的目的是从所管理的对象实例中选出一个子群，对该子群实施象 m-Get 之类的 CMIP 操作（ CMIP=公共管理信息协议）。

网络单元中被管理的对象形成一种树状分级体系结构，这种结构保存在网络单元的存储器中，其节点中包含有对象实例和相关的属性。在 SCOPE/FILTER 功能中，SCOPE 条件用于将 CMIP 操作定界到管理对象实例树（MIT， 管理信息树）的子树。这通过操作中所给的子树树根名，即 DN（可识别名） 来进行。CMIP 操作的 FILTER 条件分别作用于每一个其功能相当于子树节点的对象实例。在对对象实例实施 CMIP 操作之

前，对象实例属性的值必须要满足这一条件。应用 SCOPE 和 FILTER 条件后，CMIP 操作所涉及的对象实例群就建立起来了。

在测试对象实例的属性值是否满足 FILTER 条件时要进行比较，在比较过程中对象实例的属性值将与特定的参考值进行比较。由于没有通用的测试方法，需要为网络单元中的每种属性类型创建单独的特定比较函数。为了实现上述功能，网络单元需要有很大的存储器容量。就大的网络单元而言，通常有数百甚至是数千个被管理的对象，由于进行比较所需要的总存储容量扩展到很大，这样使存储器的管理变得复杂并且降低了它的速度。如果考虑小的网络单元，这种网络单元从商业角度也要强调对存储器的这种要求。因为在尽可能经济地生产小网络单元（例如，移动电话）的竞争中，存储器仅仅扩展一个字节都有意义。

由于上面的原因，在实现 FILTER 条件时，应该采用一种通用的方案，即：如何以通用方式在网络单元中实现 FILTER 条件测试。其目的是实现一种函数，该函数能够根据 CMIP 操作的 FILTER 条件和对象实例的属性值来判定对象实例的属性值是否满足 FILTER 条件。在前面提到的标准 ISO/IEC-9596-1, p.16 中公开了 FILTER 条件的形式，它利用了 ASN.1 数据类型 CMISFilter 中 ASN.1 的表示法(ASN.1 = Abstract Syntax Notation One, 抽象语法表示 1）。（在标准 ISO 8824 中定义了 ASN.1 表示法，对此感兴趣的读者可以从中找到更详细的说明）。实际上，FILTER 条件是正规表达式，（参见程序语言中的表达式），它可能包括逻辑运算符 (AND, OR, NOT)，关系运算符(=，≤，≥)，图形串运算符（initialString, anyString, finalString) 以及集合运算符（present, subsetOf , supersetOf, non_nullIntersection)。

逻辑运算符和图形串运算符在测试通用的 FILTER 条件时不会有问题。而在涉及关系运算符和集合运算符的执行时会产生问题。下面的例子将清楚地显示这些运算符执行时的困难:

假定要比较的属性是属于类型 A，用 ASN.1 表示法表示如下:

```
A::= SEQUENCE {
    B   INTEGER,
    C   REAL,
```

D GraphicString

}

上述的表示法说明类型 A 的属性是三个变量 B, C 和 D 的序列，其中 B 是整数，C 是实数，D 是图形串。在网络单元中，类型 A 可以用 C 语言的下列结构数据类型来实现:

```
typedef struct {
        int   b;
        float c;
        char  *d;
} A;
```

如果属性值比较函数获得的真正的属性值和参考值仅仅是八字节串的形式，而不知道数据类型更详细的结构，那么就很难想象该如何来进行属性值之间通用的相等比较。即便是类型 A 的两个值有完全相同的含义，如果域 D 的图形串指针指向了不同的地方，那么线性八字节串的比较将导致错误的结果。相等比较只有在比较函数知道“数据类型 A 是一记录，它的第三个域 (D) 是一图形串指针，以及第三个域的相等比较要在指针寻址的图形串之间进行”的情况下才能给出正确的结果。

因此，测试 FILTER 条件时出现的问题起源于这样的事实，即：不能对包含在 FILTER 条件中的属性值实施线性八字节串比较，而比较者必须通过某种方式知道要比较的属性值的数据类型。

发明内容

本发明的目的是提供一种方法，通过它可能避免上述问题。这个目的通过依据本发明的方法可以达到，其特征在于：单个对象实例属性之间的比较通过下述过程来完成：（a）读取预先存储在网络单元存储器中，且与该属性相关联的语义数据，（b）在该网络单元中搜索该对象实例所关联的属性值，（c）利用所读取的语义数据比较属性值和收到的参考值。

本发明的思想是在网络单元的存储器中保存语义数据（例如已在使用阶段），该语义数据指示在比较特定对象实例的属性时要使用的数据类型。当网络单元已投入使用，而网络管理系统（操作员）正在为将实施于网络

单元的 CMIP 管理操作搜索目标群时，首先从网络单元读取这些语义数据，接着这些语义数据被用于比较对象实例的属性值和所接收到的来自网络管理系统的参考值。

依据本发明的结果，假定与所使用的描述语言（例如，ASN.1）一致的数据类型将被创建为一种程序语言的数据类型，该程序语言有翻译程序（例如， C 或 C++ 编译器）。这要在网络单元中根据明确的规则来进行。前面给出了 ASN.1 数据类型和从 ASN.1 数据类型导出的 C 语言数据类型的例子。从 ASN.1 表示式导出的数据类型（例如，用 C 语言）的实例此处称之为本地区域(home areas)。这样在网络单元中，属性值被表示为本地区域。

本发明的基础在于：在进行 CMIP 管理操作时，如果在网络单元中知道对应于属性值的本地区域的语义，那么属性值的比较就能够实现。

本发明在网络单元中使用少得多的存储器，并改进存储器管理。虽然必须在网络单元中为每一种类型的属性存储一个不同的语义数据，但是其意义在于：语义数据可以存储在比比较函数小很多的存储空间中，而如果没有依据本发明的通用比较方法存在，就需要有比较函数。

附图说明

下面，参考附图的图 2..5，将详细公开本发明和所涉及的优选实施例，其中：

图 1 示出了一种典型的网络管理系统；

图 2 示出了被管理的对象实例形成的树；

图 3 示出了如上所述用 C 语言描述的对应于数据类型 A 的语义树；

图 4 示出了一个对应于 FILTER 条件的树状数据结构的例子；以及

图 5 说明在示范条件下本发明的操作，在这里网络操作员搜索一群特殊的用户。

具体实现方式

如果要在通信网络中对网络单元实施管理操作，必须要为该管理操作找到最终目标群， 根据 CMIP 数据传送协议，网络管理系统要向网络单元发送关于操作和目标群的信息。目标群表示为一群对象实例（即子树，其根节点标识符要发送给网络单元）以及对象实例属性的比较准则，准则

包括一个或多个参考值和一个或多个比较条件。接下来，在网络单元中，属于目标群的对象实例的属性值将与接收到的参考值利用接收到的比较条件进行比较，最终目标群是从原始目标群中属性值满足所收到的比较准则的对象实例中选出。

图 2 示出了被管理的对象实例的树，即：MIT，以及由 SCOPE 和 FILTER 条件所定界的对象实例群。 由 SCOPE 条件定界的对象实例群（例如，一颗子树）位于虚线 S 之内。箭头表示 FILTER 条件“属性 A 的值为 1”所定界的对象实例。这样，在图 2 的例子中， CMIP 操作的最终目标群将由箭头所示的对象实例构成。

下面将更详细地描述根据所收到的来自网络管理系统的信息如何在网络单元中进行比较。

为了实现通用的比较机制，在网络单元中，数据类型的实例即本地区域语义表示为树状数据结构，它在几个层上有节点。每一节点表示在比较对象实例属性时要使用的数据类型。

树状数据结构对应于一颗分析树，该分析树由所使用的程序语言的翻译程序（如 C 编译器）从本地数据类型中生成。在由 ASN.1 数据类型生成网络单元中使用的程序语言的数据类型的同时，生成本地区域的语义树是有利的。通常在该阶段要使用编译程序，它把 ASN.1 的表达式翻译为程序语言的数据类型。

图 3 所示的语义树相当于上例中类型 A 的 C 语言版本。利用该语义树，可以详尽了解存储器中的本地区域的结构。数据类型 A 的本地区域的语义树说明本地区域属于 C 语言的“结构”类型，并且它包括两个独立的存储器区域。第一个存储器区域顺序包括一个四字节长的整数(ASN.1 的 INTEGER 类型，C 语言的 int 类型)，一个四字节长的浮点数(ASN.1 的 REAL 类型，C 语言的“float”类型)，以及第二个存储器区域的初始地址(ASN.1 的 GraphicString 类型，C 语言的 “Char *” 类型，即：图形串指针)。第二个存储器区域包括图形串。相应地，利用语义树可以知道“类型 A 本地区域的第三个域 D 的相等比较可以在图形串之间进行，图形串通过位于距离本地区域开始位置八个字节处的指针来寻址”。相应地，在比较本地区域的第一和第二个域时也要利用语义树所包含的信息。这样，

利用语义树，通过对应于属性值的本地区域的比较可以完成类型 A 的两个属性值的比较。

假定 CMIP 操作（例如，m-Get）的 FILTER 条件在网络单元（如诺基亚 DX200 交换机）中表示为树状数据结构，其叶节点包括：所比较的属性的对象标识符，参考值的本地区域，以及运算符。至于运算符，有关系运算符（≤，≥，=），图形串运算符（initialString，anyString 或 finalString）或一串集合运算符（present，subsetOf，supersetOf 或 nonNullIntersection）。在树的其它节点中可能有逻辑运算符（AND，OR 或 NOT）。图四示出了一个对应于 FILTER 条件的树状数据结构的例子，在这个例子中条件语句为：（（属性 A=a）AND（属性 B≤b））OR（（属性 C≥c）AND （具有属性 D））。

根据下述的 evaluate_filter 算法可以生成测试 FILTER 条件的程序代码。evalate_filter 迭代算法从根节点开始遍历表示 FILTER 条件的树状数据结构，返回 FILTER 条件的有效性作为结果。该算法要测试除叶节点外所有的树状数据结构。对叶节点中条件的测试方法下面将会更详细地描述。在这种算法中，叶节点的测试被 compare_values 操作所取代。这种算法没有考虑就其结构而言树状数据结构可能有错误的情况。

evaluate_filter 算法可以表示如下：

```
boolean  evaluate_filter（root node）{
        辅助变量: operator，child node，reuslt;
        if （root node 为空）
            return  TRUE;
        if  （树只有 root node）
            return  compare_values(root node);
        operator := root node 中的运算符;
        child node := root node 的左孩子;
        if  (operator  =“ “NOT” ”)
            result := 对 evaluate_filter( child node) 的值取反;
        else
            do {
```

```
                result := evaluate_filter(child node);
                if （child node 非空）
                  child node := child node 的右兄弟;
            } while ((( result = FALSE and operator = “ “OR” ”) or
（result = TRUE and operator =
                “ “AND” ”) ）and  child node 为空);
        return  result;
}
```

evaluate_filter 算法表明测试 FILTER 条件中的逻辑运算符不会有问题，但在测试表示 FILTER 条件的树状数据结构的叶节点中的条件时，即：完成 evaluate_filter 算法中的 compare_value 操作时，会产生问题。

如上所述，用于比较 FILTER 条件所包含的属性值的方法分为两步。根据本发明，第一步包括：为所有在网络单元中存在的属性值的本地区域构造语义树并创建目录。语义树和目录存储在网络单元中。第二步是对要比较的属性值使用 compare_values 算法，该算法的功能以对所存储的语义树的解释为基础。

实现 compare_values 算法的先决条件是语义树的节点要包括下列信息：对应于本地区域的 ASN.1 类型的标识符，本地区域类型的标识符，记录类型本地区域中每个域相对于本地区域起始位置的偏移，以及本地区域的大小。此外还必须有一目录，由属性记录的对象标识符通过目录与该属性的本地区域语义树相关联。图 3 是本地区域语义树的例子。

属性值比较的原则可由 compare_values 算法来说明。这里给出的是算法的简化形式，并且只对相等比较进行处理。举例来说， ASN.1 类型的 SET 和 SET OF 就未作处理。但是，根据这里所公开的说明，本领域技术人员将能够写出完整的算法。图 4 中表示 FILTER 条件的树状数据结构的叶节点将作为输入提供给 compare_values 算法。叶节点中包含有比较中要使用的属性的对象标识符，关系运算符，以及在比较中要使用的属性值。这种算法假定在比较开始时能够检索出对应于属性标识符的属性值。因此，该算法能够提供属性值比较成功或失败的信息。该算法没有考虑任何可能

出错的情况。算法 compare_values 调用了辅助迭代算法 compare_home。

compare_values 算法:

```
boolean compare_values (leaf node) {
  辅助变量: semantic tree_root, attribute_value;
        semantic tree_root := search_semantic tree_root( leafnode.attribute_identifier);
        attribute value := search_attribute_value( leafnode.attribute_identifier);
  return compare_home(semantic tree_root,
                      attribute_value,
                      leaf node.attribute_reference value);
}
```

辅助算法 compare_home:

```
boolean compare_home ( root, attribute_value, attribute_reference value)
{
    辅助变量:  child node, result;

  if (root.home area_type = 指针类型 ) {
      attribute_value := <把 attribute_value 解释为一个地址，并且在该地址所指示的位置搜索新
                          的值>;
      attribute_compare value := <将 attribute reference_value 解释为一个地址，并且在该地址所
                          指示的位置搜索新的值>;
   }
  if ( root.ASN1 类型 = SEQUENCE) {
      child node := 根节点的左孩子;
      do {
          result = compare_home (
```

```
                                child node,
                                attribute_value + child node.OFFSET,
                                attribute reference_value + child
node.OFFSET);
                child node := 子节点的右兄弟;
              } while (result = TRUE and 子节点存在）;
                return result;
        } else {
          if  ( root.ASN1-type = BOOLEAN)
                    return compare_boolean ( attribute_value,
attribute_reference_value);
          else if  (root.ASN1-type = INTEGER)
                    return compare_interger(attribute_value,
attribute_reference_value);
            ...
          }
      }
```

辅助子程序 compare_boolean (比较整数是否相等）：

```
boolean  compare_boolean (value, reference value)
 {
   return value = refernce value;
}
```

正如上面公开的内容，网络单元的比较程序要读入所接收到的属性的标识，参考值和数据类型语义。接着，比较程序要求单独的自适应程序（自适应程序包含有关于何处能真正检索到属性，以及哪个能够用正确的数据类型来返回属性值的信息）提供属性的实际值并进行比较。其结果是得到属性值匹配的对象实例，并对这些对象实例实施操作。

为了清晰地说明上面的内容，下面将分析一个以电话交换机作为网络单元的实际例子。网络单元包含有用户寄存器，它保存着连接到该网络单元的用户的信息。在网络管理接口上，用户被表示为“Subscriber”对象类，具有“Line number”和“Directory number”属性。这个例子是虚构的，但是属性“Directory number” 可以是指，例如信令中用户线的号码表示，而 “Line number” 可以是网络单元内分机用户线的标识符。属性“Line number” 用对象标识符{1，2，3} 来注册，而属性“Directory number”用标识符 {1，2，4}来注册。在 ASN.1 中属性表示如下:

Line number :: = INTERGER;

Directory number ::= GraphicString;

利用 ASN.1 翻译器，属性的 ASN.1 表示可以被翻译成 C 语言中的数据类型表示:

typedef long Line number;

typedef char *Directory number;

假定网络管理工作站（图 1 中 WS 参考标记）想要找出与网络单元相连，并且其电话号码以数字 4 或 5 开始的用户。网络管理工作站向网络单元发送下面的 CMIP 操作 m-Get，它包括第一目标群(S) 和 FILTER 条件， 即: 对象实例属性的比较准则。下面利用 ASN.1 的符号表示法来说明，利用 baseManagedObjectInstance 域的值和该域的范围值，在 CMIP 协议的 PDU（协议数据单元）中确定 S 群，baseManagedObjectInstance 域指出了在网络单元的 MIT 中的子树根节点。与本例无关的域已被忽略并被三个点取代。FILTER 条件由 filter 域的值来表示。至于根节点，本例子有一个表示用户寄存器的对象实例。为了清晰起见，本例子中用户寄存器实例的实际名称被标识符<Subscriber register> 取代。scope 域的值 wholeSubtree 说明群 S 包括了子树的所有对象实例，即: 与网络单元相连的所有用户。 至于 filter 域的值，有一个 ASN.1 表示式，如果用自然语言它可以描述为表达式:“电话号码以数字 4 开始或电话号码以数字 5 开始”。

{

```
...
baseManagedObjectInstance <Subscriber register>
...
scope wholeSubtree,
filter or {
    item {
      substrings {
        initialString {
          attributeId { 1   2   4 }
          string " "4" "
        }
      }
    }
    item {
      substrings {
     initialString {
          attributeId   {1 2 4}
          string " "5" "
          }
        }
      },
     ...
  }
```

在下文中，括号内的数字表示图 5 中相应的数字，它说明在网络单元中代理程序的操作。（但是这些数字未必表示操作执行的顺序）。在网络单元中，负责执行操作的代理程序接收到了 m-Get 操作 (1)。 代理程序将 m-Get 操作的 FILTER 条件组织成树状的结构（2）。代理程序在网络单元的对象实例树（即: MIT）上定界群 S，群 S 根据 m-Get 操作确定

（3）。图 5 中群 S 在 MIT 中用虚线定界。代理程序对属于群 S 的每一对象实例单独处理（4）。通过调用上述的 evaluate_filter 算法实现，代理程序要确定一个特定的对象实例是否属于最终对象实例群（5）。根据 FILTER 条件中的属性标识符{1 2 4}，evaluate_filter 检索对应于属性“Directory number”的本地区域的语义树（6）。在这个示范性的例子中，语义树中的标记意味着标识符为{1 2 4} 的属性将被认为是 ASN.1 类型的 GraphicString，而本地区域的数据类型是 C 语言的 “char *”， 即：图形串指针。包含在语义树中的信息控制 compare_filter 以上述方式去比较包含在 FILTER 条件中的比较值和该对象实例的“Directory number”属性。接下来，代理程序向网络管理工作站返回电话号码为“56789”和“442224”的用户作为 m-Get 操作的结果（7）。这些用户构成了最终目标实例群。在图 5 的 MIT 中，箭头指向所讨论的用户。

正如上面所提到的那样，基于语义树的使用并用于比较属性值的这种比较方法是通用的。通用性这里意味着这种方法与属性的 ASN.1 表示无关。除了上面所述的优点之外，该方法的通用性能够大大地节省网络单元软件开发的费用，这是因为 ASN.1 表示的修改或新 ASN.1 表示的实现都不需要额外的软件开发。由于上述的原因，可以更彻底地测试这种方法的实现，这将会提高比较结果的可靠性和整个网络单元的可靠性。

对本领域的技术人员而言，很显然本发明的不同实施例并不限于上面的例子，这些实施例可能会在所附的权利要求书的范围内变化。

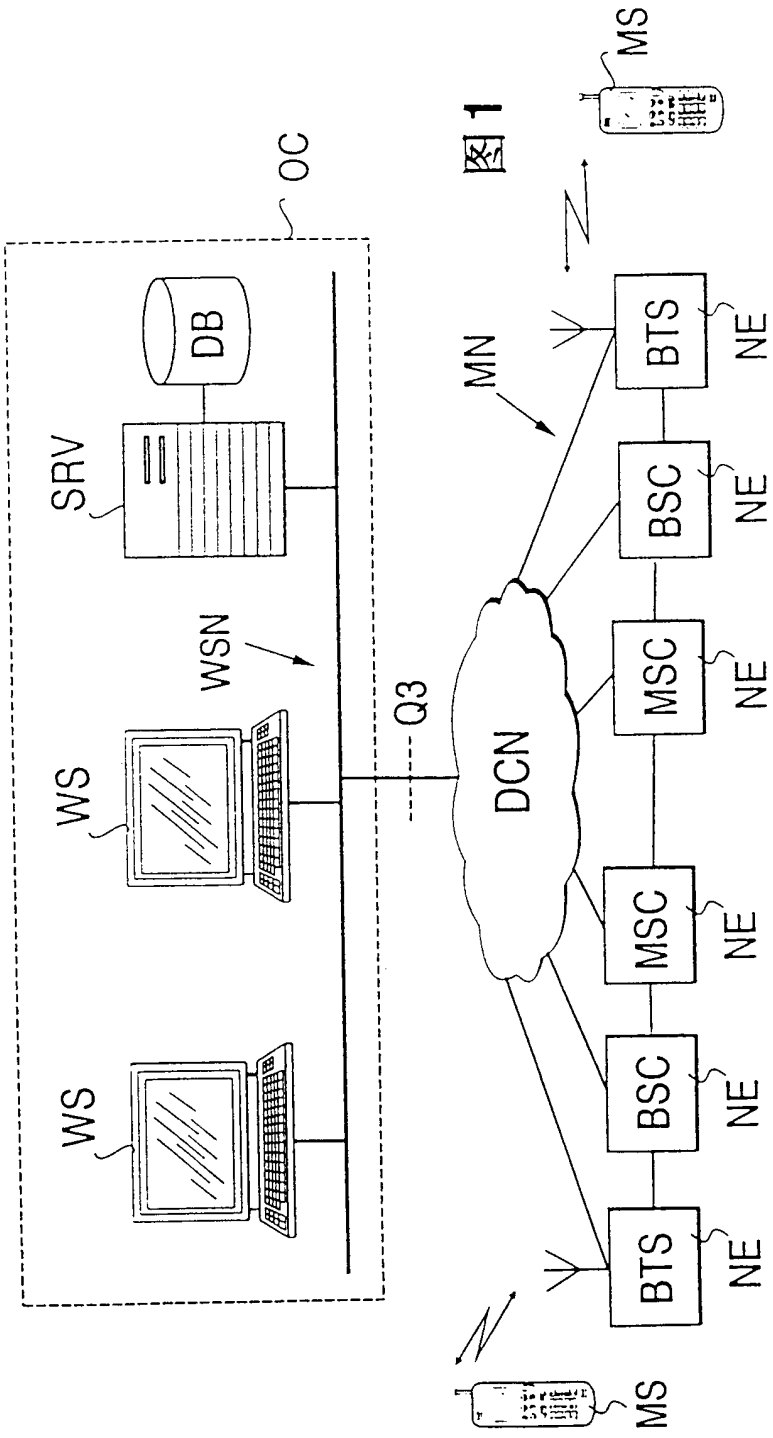

OC
DB
SRV
WSN
WS
WS
Q3
DCN
MN
MS
BTS
BSC
MSC
MSC
BSC
BTS
NE
NE
NE
NE
NE
NE
MS

图 1

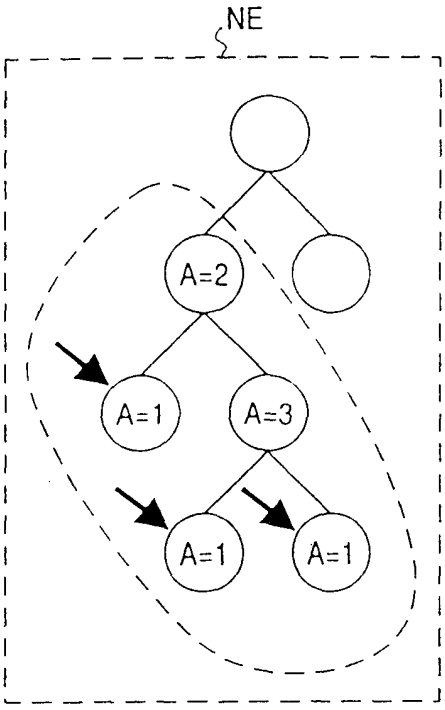

NE
A=2
A=1
A=3
A=1
A=1

图 2

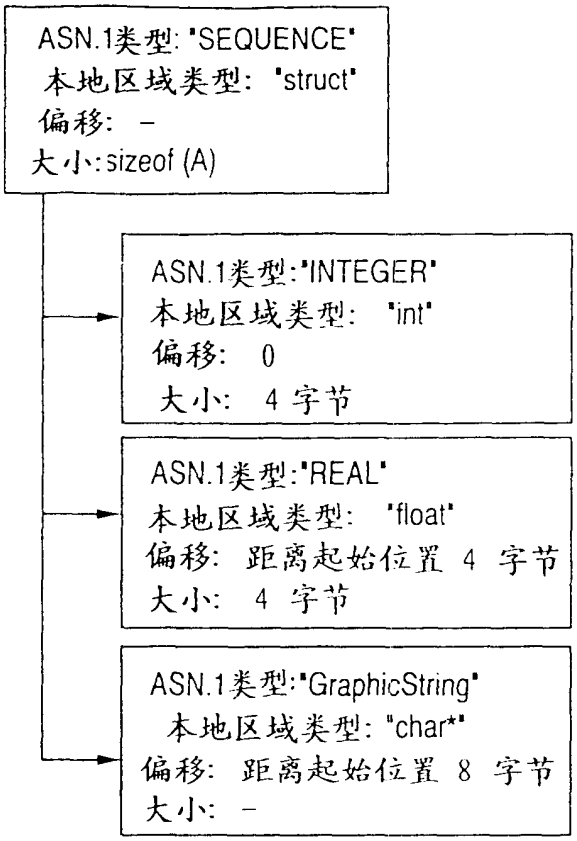

ASN.1类型: "SEQUENCE"
本地区域类型: "struct"
偏移: -
大小: sizeof (A)
ASN.1类型: "INTEGER"
本地区域类型: "int"
偏移: 0
大小: 4 字节
ASN.1类型: "REAL"
本地区域类型: "float"
偏移: 距离起始位置 4 字节
大小: 4 字节
ASN.1类型: "GraphicString"
本地区域类型: "char*"
偏移: 距离起始位置 8 字节
大小: -

图 3

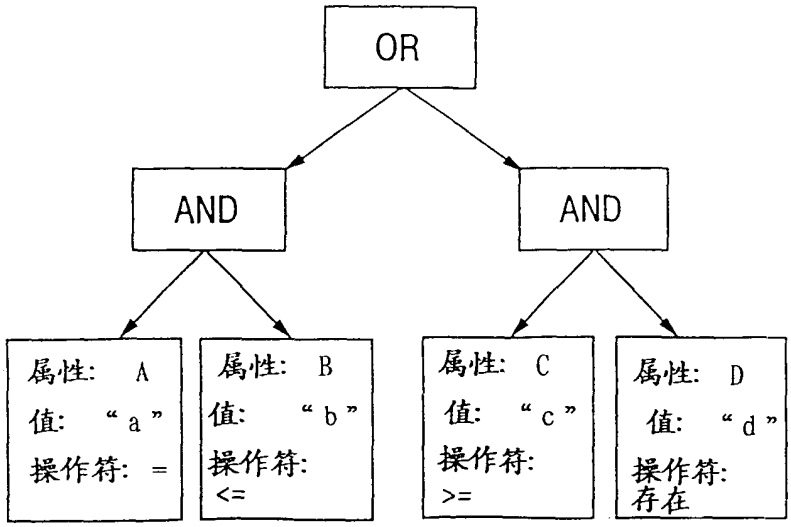

OR
AND
AND
属性: A
值: "a"
操作符: =
属性: B
值: "b"
操作符: <=
属性: C
值: "c"
操作符: >=
属性: D
值: "d"
操作符: 存在

图 4

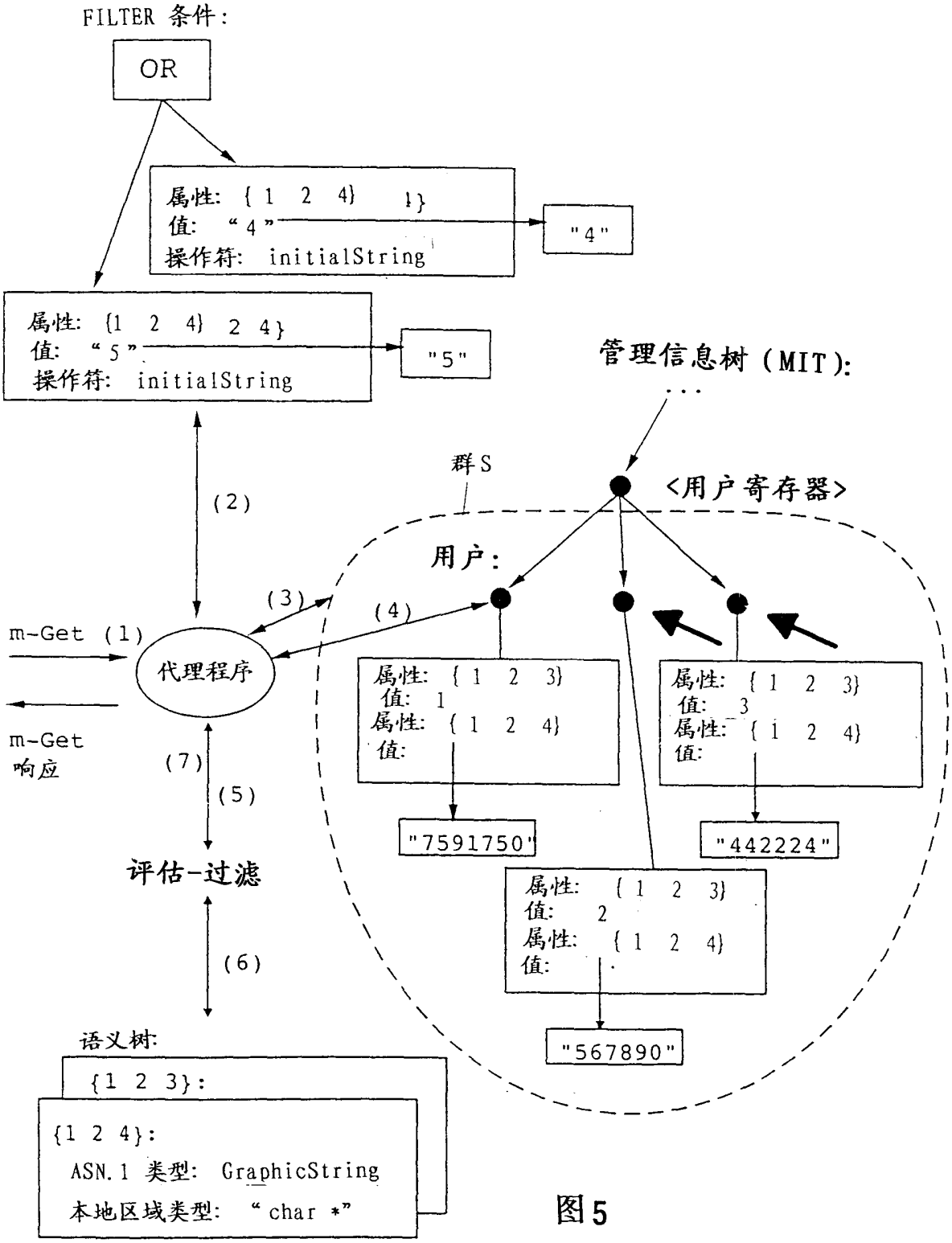

FILTER 条件:
OR
属性: { 1 2 4} 1}
值: "4"
操作符: initialString
"4"
属性: {1 2 4} 2 4}
值: "5"
操作符: initialString
"5"
管理信息树 (MIT):
...
群S
<用户寄存器>
用户:
(2)
(3)
(4)
m-Get (1)
代理程序
m-Get
响应
(7)
(5)
评估-过滤
(6)
属性: { 1 2 3}
值: 1
属性: { 1 2 4}
值:
"7591750"
属性: { 1 2 3}
值: 3
属性: { 1 2 4}
值:
"442224"
属性: { 1 2 3}
值: 2
属性: { 1 2 4}
值:
"567890"
语义树:
{1 2 3}:
{1 2 4}:
ASN.1 类型: GraphicString
本地区域类型: "char *"

图5