[19]中华人民共和国国家知识产权局

[51]Int.Cl[7]
G06F 7/00

[12]发明专利申请公开说明书

[21]申请号 97199793.4

[43]公开日 2000年3月29日

[11]公开号 CN 1249042A

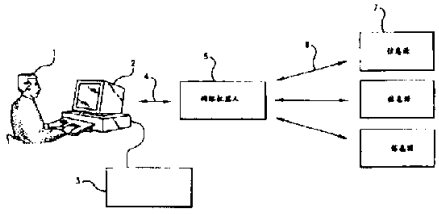
[22]申请日 1997.9.22 [21]申请号 97199793.4
[30]优先权
[32]1996.9.20 [33]US [31]60/025,304
[86]国际申请 PCT/US97/17132 1997.9.22
[87]国际公布 WO98/12881 英 1998.3.26
[85]进入国家阶段日期 1999.5.17
[71]申请人 耐特波特股份有限公司
地址 美国加利福尼亚州
[72]发明人 D·克里斯蒂安森 R·B·多伦波斯
O·埃茨奥尼 C·夸克 G·劳克哈特
E·泽尔贝格
D·S·韦尔德
[74]专利代理机构 上海专利商标事务所
代理人 张政权

权利要求书2页 说明书42页 附图页数5页

[54]发明名称 用于网络信息访问的方法和系统

[57]摘要

本发明为用户访问连网的信息源提供帮助。在一个方面,本发明 是这样一种 方法,它智能化地把用户查询路由选择到与该查询相关的信息源,从接收到的响 应中提取相关数据字段。并按照所估计的相关性的顺序把所提取的 数据智能化地 显示给用户。本发明的系统在一个或多个连网计算机上以集中或分 散的方式实现 该方法的一个或多个步骤。此外,本发明提供了一种新的语言,它有助于容易地 编写和维持信息源查询和响应格式的描述。



ISSN1008-4274

权 利 要 求 书

1. 一种有助于用户查询可从连到网络的信息源中获得的信息的方法，其特征在于所述方法包括:

a. 选择与用户查询最相关的一个或多个信息源;

b. 依据以包装器描述语言编写的每个所述相关信息源的描述，对每个所述相关信息源的所述用户查询进行格式化;

c. 把所述初始化的查询传送到所述相关信息源中的每一个;

d. 从所述相关信息源所返还的响应中提取与所述用户查询相关的数据字段，所述的提取是依据返还每个所述响应的所述相关信息源的所述描述进行的;以及

e. 把所述相关数据字段显示给所述用户。

2. 如权利要求 1 所述的方法，其特征在于还包括，在所述提取步骤后，估计所述响应与所述用户查询的相关性的步骤。

3. 如权利要求 1 所述的方法，其特征在于所述选择步骤还包括以下步骤:

a. 确定与所述用户查询相关的一个或多个概念; 以及

b. 如果一信息源与所述概念相关，则选择所述信息源是相关的。

4. 如权利要求 1 所述的方法，其特征在于所述包装器描述语言包括用于就成分规范表达式来规定整个规范表达式的功能，从而在识别所述整个规范表达式中的一个成分规范表达式时，执行动作是使用在识别所述成分规范表达式时候联系的变量，实际上只在也识别所述整个规范表达式时执行所述动作.

5. 一种有助于用户查询可从连到网络的信息源中获得的信息的系统，其特征在于所述系统包括:

a. 第一处理器装置，用于选择与用户查询最相关的一个或多个信息源;

b. 第二处理器装置，用于依据以包装器描述语言编写的每个所述相关信息源的描述，格式化所述用户对每个所述相关信息源的查询，并把所述格式化的查询传送到所述相关信息源中的每一个;

c. 第三处理器装置，用于从所述相关信息源所返还的响应中提取与所述用户查询相关的数据字段，所述的提取是依据返还每个所述响应的所述相关信息源的所述描述进行的; 以及

d. 第四处理器装置，用于把所述相关数据字段显示给所述用户。

6. 如权利要求 5 所述的系统，其特征在于所述第一、所述第二、所述第三和所述第四处理器装置位于连网的计算机中，所述计算机包括一 CPU 和存储器。

7. 如权利要求 5 所述的系统，其特征在于所述第一、所述第二、所述第三和所述第四处理器装置位于分开的连网的计算机中，每个所述计算机包括一 CPU 和存储器。

8. 如权利要求 5 所述的系统，其特征在于所述第四处理器装置通过执行万维 Web 浏览器来显示所述相关数据字段。

9. 如权利要求 4 所述的系统，其特征在于还包括第一个连网的存储器装置，所述存储器装置用于存储所述信息源的描述并在所述第二或第三处理器装置请求时提供所述描述。

10. 一种包装器描述语言方法，其特征在于包括通过依据以规范表达式语言所编写的规范表达式扫描一个串来识别所述串，所述规范表达式语言能就成分规范表达式来规定整个规范表达式，从而在识别所述整个规范表达式的一个成分规范表达式时，执行动作是使用在识别所述成分规范表达式时联系的变量，实际上只在也识别所述整个规范表达式时执行所述动作。

11. 如权利要求 10 所述的包装器描述语言方法，其特征在于还包括执行以动作语言编写的一个或多个语句的步骤，所述动作语言能规定变量上算符的动作、变量上函数以及语句顺序的动作，所述动作语言语句还能嵌入所述规范表达式语言，用以规定在识别所述成分规范表达式时所发生的所述动作。

12. 一种计算机可读媒体，其特征在于包含使一个或多个计算机实现权利要求 1 的方法的指令。

13. 一种计算机可读媒体，其特征在于包含使一个或多个计算机进行权利要求 10 的方法的指令。

说　明　书

用于网络信息访问的方法和系统

技术领域

本发明涉及网络上的信息访问，尤其涉及响应于用户的查询对公共或专用网络上可获得的来自信息源的相关信息进行自动定位和估计.

背景技术

专用企业网和公共因特网的飞速发展使得大量文档、数据库和实用程序(utilities)不断增长，而成为可怕的非常复杂的事．现在几乎任何类型的信息均存在于某个地方，但大多数用户不能找到他们要找的信息，甚至老练的用户也需要浪费许多时间和精力来寻找适当的信息源．一个问题是简单的，即不断增长的大量可获得的信息源，这超出了单个用户的理解力．第二个问题是，随着可获得的信息及信息源的这种增长，用来管理、访问和显示(presentation)该信息的方法和软件实用程序也相应增加．每个实用程序具有不同且通常是独有的界面以及命令和能力的集合，并且适应于不同的用户集合和不同的信息类型与信息源的集合．于是，可获得的实用程序的迥异给用户造成的问题可与信息爆炸所产生的问题相比拟．现在，用户面临着使用哪个工具在哪个信息源进行查询的双重问题.

过去，为给用户提供有助于解决网络革命所带来的这种双重问题的自动计算机辅助服务，已进行了许多努力．例如， AI 的研究人员已创造了几个帮助用户进行电子邮件和网络新闻筛选(见 Pattie Maes 等人的 Learning interface agents, *Proceedings of AAAI-93* 一文, 1993)的原型软件工具、有助于万维网(World Wide Web)浏览的工具(见 H. Lieberman 的 Letizia: An agent that assists web browsing 一文, *proc. 15th Int. Joint Conf. on A.I.* pp. 924-929, 1995 ; Robert Armstrong 等人的 Webwatcher: A learning apprentice for the world wide web 一文, *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments,* pp. 6-12, Stanford University, AAAI Press, 1992)、计划会议的工具(见 Lisa Dent 等人的 A personal learning apprentice 一文, *Proc. 10th Nat. Conf. on A.I.*, pp. 96-103, 1992 ; Pattie Meas 的 Agents that reduce work and

information overload —文， *Comm. of the ACM* 37(7):31-40, 146, 1994； Tom Mitchell 等人的 Experience with a learning personal assistant —文， *Comm. of the ACM* 37(7):81-91, 1994)以及进行有关互连网的任务的工具(O. Etzioni 等人的 A softbot-based interface to the internet, *CACM* 37(7):72-75, 1994). 这些工具需要访问的不断增长的信息可在万维网上获得. 不幸的是，甚至与 WWW 同样标准化的领域也会对自动软件工具造成严重的问题. 例如，虽然通常可以用超文本标记语言("HTML")来编写网页，但这种语言仅定义了信息显示的格式，而未尝试暗示其含义或语义上的内容. 目前，对于 Web，不存在可接受的"语义标记语言"，也不存在可能被广泛采纳的语言. 因特网可能会造成更大的问题.

因而，企业网、因特网以及万维网的出现已对旨在帮助用户找到相关信息的自动服务或工具造成几个基本的问题. 第一，迄今为止没有一个服务能提供足够的附加值来替换使用诸如 Yahoo 或 Lycos 等可访问现有目录和索引的 Web 浏览器. 第二，这种服务还不能被理解，也不能胜任地对从各种因特网和 Web 信息源返回的响应中的相关信息进行分析. 第三，现有的服务和工具还不便于适应数目不断增长的信息源及其不断变化的响应格式. 这是由于现有的工具所利用的每个因特网服务和 Web 方面的个性化的手工编码的界面所引起的(见 Yigal Arens 等人的 Retrieving and integrating data from multiple information sources, *International Journal on Intelligent and Cooperative Information Systems* 2(2):127-158, 1993； O. Etzioni 等人的 A softbot-based interface to the internet, *CACM* 37(7):72-75, 1994； B. Krulwich 的 Bargain finder agent prototype, Technical report, Anderson Consulting, 1995； Alon Y. Levy 等人的 Data model and query evaluation in global information systems, *Journal of Intelligent Information Systems, Special Issue on Networked Information Discovery and Retrieval* 5(2), 1995； Mile Perkowitz 等人的 Category translation: Learning to understand information on the internet, *Proc. 15th Int. Joint Conf. on A.I.*, 1995). 一个服务或工具最好能访问一个新的或变化的因特网信息源，以便自动地学习如何检索来自该信息源的相关信息. 这样，即使在这种设备限于具有依据某些强制原理而选中的响应格式的信息源组时也是有利的.

发明内容

本发明的主要目的是通过提供个人化网络机器人叫做"netbot.(网络机器人)"的方法和系统来解决这些基本问题．网络机器人通过跟踪可获得的的网络信息源，了解每个特定源的相关信息和特征，并在用户请求时确定哪个源与给定的查询相关，把此查询传送到最相关的信息源，理解从每个源返回的响应，为用户综合并灵活地提供查询结果，从而担当用户的智能化助理的角色．

本发明的网络机器人具有下列优点．第一，网络机器人只向用户返回最相关的信息．一方面，只将每个用户的查询传送到被确定为最相关的主要信息源．另一方面，如此分析和理解信息源的响应，从而只提取相关的数据项给用户显示．丢弃复制、陈旧和及不相关的信息项．第二，网络机器人是快速的．由于它可自动地以并行方式搜索主要相关源，所以它可在最快的主要源返还响应时就立即提供信息．尽管改变条件将使不同信息源的速度发生波动，但网络机器人综合器(integrator)可保持与最快的信息源一样快．没有信息要返还查询的源不会使用户减缓，这是因为网络机器人简单地忽略了这些源．第三，网络机器人容易地适应其数目不断增长的网络信息源及不断变化的响应格式．网络机器人利用新型的说明性语言来描述信息源．源的描述简短而便于理解，因此容易对其进行编写和维护．

因此，本发明的一个方面包括一种有效地访问网络上的信息源的方法，该方法最好包括以下的一个或多个步骤：接收用户对信息的查询；确定与此查询最相关的信息；检索每个信息源的描述；以适用于每个信息源的方式依据此描述对该查询进行格式化并把经格式化的查询传输到该源；检索来自该信息源的响应；对于每个源，依据检索到的描述来理解和提取相关数据字段；把来自每个信息源的相关数据按照其相关估计来排列的智能化方式显示给用户．有利的是，尽可能以最大限度平行地进行这些步骤．尤其是，至少平行地把所有查询传输到所有的相关信息源，而不必等待中间的响应．

本发明的另一个方面包括一种用于进行本发明方法的一个或多个步骤的计算机系统和设备．用户具有接在一网络上的显示装置，该网络还接有多个信息源．显示装置接收用户查询并显示网络机器人的响应．此外，显示装置进行本发明方法的一个或多个步骤．有利的是，可在接到网络机器人服务器计算机的网络上进行不在该装置上进行的一个或多个步骤，网络机器人响应于来自用户装置的功能请求．或者，用户装置的范围可从无盘手提式终端到 PC 、到工作站等．

本发明的再一个方面包括一种便于创造和维护信息源的描述的新语言和语言工具．重要的是，这种语言识别从信息源返还的响应中的相关数据字段，并能提取所有这些字段．在较佳实施例中，该语言具有动作(action)语句成分和规范(regular)表达式成分．规范表达式成分具有以下这些新特征，即产生规范表达式的模块化分层描述，把变量连接到在与信息源的响应的模式匹配期间所识别的正确子串，以多个变量连接进行任意动作语言表达，以及在可能的地方指定无回溯地识别子串。

附图概述

参考附图、以下的描述和所附的权利要求将更好地理解本发明的这些和其它特征、方面和优点，其中:

图 1 从总体上示出本发明的网络机器人;

图 2A-B 示出图 1 的网络机器人实施例的示例用户界面;

图 3 示出图 1 的网络机器人的示例功能性组成；以及

图 4 示出图 1 的网络机器人的另一个硬件实施例.

本发明的较佳实施方式

为了清楚地揭示而不是限制，以一种用于网络信息访问的方法或处理、一种用于进行该方法的系统或设备以及一种设计成有助于实行该方法和系统的新语言来详细地描述本发明的网络机器人.

以下，首先对本发明的这些方面进行概述，接着对各个组成进行详细讨论.

5.1. 网络机器人构造的概述

本发明的网络机器人方法或系统包括在连有计算机的一个或多个网络中共同起动作以帮助用户访问连有服务器的网络(以下叫做"信息源")中存在的信息的软件和硬件设备．图 1 从总体上示出网络机器人与用户以及与网络化信息源的关系．例如，用户 1 通过诸如监视器 2 等标准接口装置来访问用户计算机 3．在工作过程中，用户需要来自信息源 7 的信息，信息源 7 通过诸如网络链路 4 和 6 等各种网络链路接到用户计算机．由于信息源有许多，所以用户可从寻找来自相关信息源的所需信息的协助中受益．由网络机器人 5 来提供此协助，它保持了解可获得的信息源并代表用户或作为用户的工具通过链路 6 对这些信息源进行查询.

或者，网络机器人 5 可部分或整体地位于用户计算机 3 上，或者，网络机器人可部分或整体地分布在网络上并被用户通过链路 4 进行访问.

具有类似信息种类的信息源组 7 分成概念上的种类叫做信息域. 例如，一个域可以是用于特定产品的电子存储器的域；另一个域可以包括因特网索引，这些索引包含有关各种万维网("WWW")页的关键字内容的信息.

在一较佳实施例中，网络机器人包括三个主要的功能性模块：用户界面、综合器和 I/O 管理器. 简而言之，用户界面模块与用户进行交互动作，以接收用户对信息的查询并对从信息源的网络接收到的信息响应进行格式化和显示. 有利的是，用户界面适用于被访问的特定信息域. 综合器模块接受来自用户界面模块的用户查询，选择相关的信息源，对其进行格式化以从网络传输到每个相关的信息源，接收来自这些信息源的响应，理解这些响应，以及把这些响应的相关部分传回用户界面模块以显示给用户. I/O 管理器模块为用户界面和综合器模块完成硬件、操作系统和网络特定对接，从而把网络机器人安置(port)到不同的硬件平台、操作系统或网络只需改变完全模块化的代码.

在另一个特殊实施例中，可以不需要用户界面或 I/O 管理器模块. 例如，这些模块的功能可能已由其它操作系统成分完成. 网络机器人可只设有所揭示的一个或多个设备，而不必提供其它的设备. 例如，在某些实施例中，网络机器人可只用于进行查询路由选择，或者还具有相关性排列能力. 在其它实施例中，网络机器人可简单地对查询进行格式化并理解响应. 此外，在网络机器人中还可存在其它的模块或这些模块被网络机器人所访问. 例如，可提供使网络机器人自动地获取新网络信息源的特性的学习模块. 最后，如本领域内的技术人员所公知的，在数目更多或更少的模块中，还可以其它方式来分割或组合由所述模块所进行的功能.

在没有特别提出时，可在所揭示的硬件结构上以诸如 C 等完全过程编程语言或者诸如 C++等完全面向对象的编程语言来实现本发明的处理.

用户界面模块

更具体来说，用户界面模块既具有对网络机器人用户界面通用的重要功能(无论信息域指向哪个网络机器人)，也具有对特定网络机器人的特定信息域的适用性. 首先转向较佳的通用功能，一个这样的功能是记住与网络机器人进行交互动作的用户优先级的能力. 例如，要记住的优先级包括：网络机器人是否要取得页

面来计算其本身的相关性、数 N 或待查询的相关信息源、显示特性等．第二个这样的通用功能是不断地显示从查询接收到的信息．由于每个查询通常都可使网络机器人查阅许多代理的信息源，它们是通常在各不相同的时间接收到结果．立即显示不在同一时间接收到的结果会引起一些不想要的后果，诸如屏幕闪烁或使已显示的数据杂乱地重新排列．另一方面，不断的显示一方面适应于用户快速观察信息的愿望，另一方面也适应于用户全面地观察所有接收到的信息并依据相关性进行排序的愿望．

不断显示的策略最好在用户屏幕上的一个或多个窗口上设置用于查询满意处理以及某些通用用户控制(诸如用于操纵这些窗口的屏幕按钮)的几个确定的视图．在一个窗口中，用户界面模块显示用符号表示为例如网络地址、图标或其它简洁的屏幕表象的每个被查阅的源的信息源清单．然后，当从特定源接收到回答时，其相关的屏幕表象通过例如改变强度、改变色彩等方式来改变外观．还显示了当前接收到的独特信息项总数的计数．或者，通过单击信息源的屏幕表象而打开另一个窗口，该窗口具有有关该信息源的信息，或者从该信息源接收到的响应的显示，或者对该网络上的信息源的访问等．

最好把一个通用控制用一个通用的显示按钮实现，该按钮按下时使得显示另一个窗口，其中依据对所有当前接收到的响应所估计的与查询的相关性的排列来显示这些响应的清单．最好把另一个通用控制后一窗口中的 more 按钮实现，该钮按下时，使得重新显示与自最后一个窗口显示以来新近接收到的项目合并的所有先前的数据项．把新近接收到的项目按次序或按相关性合并到显示清单中，例如由其色彩或强度把它们与先前的项目区分开来，从而用户可避免再次扫描先前的项目．或者，通过单击数据项而打开又一个信息窗口，从而给出该项目的源、包含该项目的响应的显示或者该网络上对该项目的源的访问等．

除了这些通用功能和控制以外，网络机器人的用户界面模块最好还实现适用于它所计划的信息域的特定设计、格式化和字段．例如，用于因特网上电子商店的信息域的比较购买的网络机器人可具有包含产品名称、型号和价格的标记字段的特定界面显示．另一方面，用于访问联机因特网索引的信息域的网络机器人可具有这样的界面，该界面具有标有从查询中获得的元素的字段．

在一个网络机器人的实施例中，大多数功能和模块位于连接有用户可远程访问的服务器的网络上．例如，用户可利用诸如 Netscape 等 Web 浏览器以万维网

协议来访问因特网上的一个网络机器人．在此情况下，用户界面建立了由 I/O 管理器在网络上传输的经 HTML 格式化的页面．图 2A 从总体上示出来自该实施例的一个例子的用户显示，该显示进一步指向联机电子软件商店的信息域．图 2A 的网络机器人显示被分割成三部分．部分 11 是标题部分，通常表示来源于用于购买的网络机器人"ShopBot"的显示．网络机器人最好还具有特定输入查询屏幕．部分 12 表示以其 WWW 地址 15 所代表的当前正被查阅的联机商店的清单，该地址是可选择的以提供进一步的信息或指导 WWW 访问．在部分 12 中的 16 处，类似地表示已返还查询结果的那些源．部分 13 表示迄今为止依据此特定信息域格式化成为用于大部分 PC 操作系统的段的结果．对起始的信息源用产品名称、价格和地址对所返还的每个独立项目例如项目 16 进行格式化．在此实施例中，以窗口滚动和 Web 浏览器中所固有的控制设备来控制信息显示．把此用户界面实现为在网络机器人服务器处产生的并传输到 Web 浏览器的 HTML 格式化的页面．

图 2B 从总体上示出来自基于 Web 浏览器的用户界面实施例的另一个例子的用户显示，在此情况下，该显示指向 WWW 索引或搜索引擎的一个信息域．通常把此显示分割成三个部分．部分 71 显示了网络机器人的标题；部分 72 显示了当前搜索的状态；部分 73 显示了搜索结果．更具体来说，部分 71 的显示包括该网络机器人的标识，“MC”是“MetaCrawler”的缩写，选此名是因为 WWW 搜索引擎常称为“Web Crawler”，71 的显示还控制对某些系统层显示特征的访问，诸如 MetaCrawler 主页和用户反馈页面．部分 72 的显示包括以其通用名识别的、当前查询的搜索引擎总的和每个搜索引擎的当前查询状态，以及共同的用户控制．一般，饼图图标 78 归纳了在被查询的搜索引擎中已有八分之七响应于该查询．在公知为"Lycos"的搜索引擎 75 处，检查(check)标志表示已接收到包含信息项的响应．在公知为"Inktomi"的搜索引擎 76 处，交叉(cross)标志表示已接收到没有任何信息项的响应．另一方面，可从视觉上把公知为"Galaxy"的搜索引擎 77 与其它搜索引擎区分开来，以表示它还未响应于此查询．部分 72 的共同控制包括请求显示新近到达的搜索结果的 more-按钮 79 以及请求发送新的或经改变的查询的 modify-search-按钮 80．最后，部分 73 的显示包括从搜索引擎返回的信息项．每个信息项被分开地显示且包括标题 81、可获得的描述性文本 82 以及具有该信息项网页的 URL 以及所估计的该信息项与查询的相关性(在这里为"1000.")

的行 83．通过所估计的相关性的降序排列来分类显示项目．使用 Web 浏览器提供的控制来滚动所显示的项目．该用户界面由 Java 小应用程序实现，它从网络机器人服务器下载并由 Web 浏览器执行．这样，图 2B 的界面比图 2A 的界面具有更大的互操作性．例如，它可轮询网络机器人服务器的当前搜索状态且无需用户的动作而相应地更新状态显示．

虽然主要借助窗口和按钮来描述用户界面，但本领域内的技术人员应知道本发明可适用于为显示信息和输入用户命令而设的其它显示范例．例如，用户界面模块可控制整个屏幕和当前的图形显示，而无需窗口系统的介入．

最好以面向对象的编程语言来实现用户界面模块，该语言补充了提供开窗口功能的类库．较佳的实施例是使用带有 java.awt 软件包的 Java 语言．例如，见 Flanagan, 1996, Java In A Nutshell, O'Reilly & Associates, 第 5 和 19 部分．

综合器模块

图 3 示出较佳的功能模块、数据库以及它们在功能上与网络机器人 30 的大致相互关系以及特别是与综合器模块 37 的功能上的相互关系．综合器最好由三个功能构成：查询路由器 39、包装器(wrapper)数据库 40 和聚合(aggregation)引擎 38．以下将详细地描述这里所引入的元件．如果给出从用户界面模块 34 所传来的用户查询 31，则综合器首先呼叫查询路由器 39 按照相关性来排列网络机器人所知的网络信息源并返还 N 个较相关的源．接着，综合器检索来自包装器数据库 40 的 N 个最相关源的 N 个包装器．以本发明的包装器描述语言来编写这些包装器，它们是对信息源及其要求的描述．聚合引擎 38 使用所检索的包装器，首先把此查询格式化成为被每个信息源所识别的形式，接着理解被每个信息源 33 所返还的信息，从而消除无关的格式化成分并使接收到的信息构成通用格式．然后，依据较佳的增量显示方法 36 来聚合此格式化信息并传送到为用户显示 32 的用户界面模块．还可由所存储的用户的喜好 35 来控制用户显示．

查询路由器

性能良好的外来机器人最好应以足够有效和节约的方式使用宝贵的网络带宽和信息源处理资源，同时最好地答复每个用户查询．这种性能把资源的使用量减到最少，从而对各个网络机器人和在一网络上同时起动作的所有网络机器人实现最好的整体性能．

查询路由器对实现此性能来说是重要的，因为它允许网络机器人只向可能具

有与查询相关的信息的信息源发送请求．查询路由器从用户查询中确定每个信息源与给定查询的相关性并返还 N 个最相关的源． N 是可由用户通过喜好来控制的参数，它可以小到 1．此相关性确定宁可多算几个而不要少算几个．宁可偶尔包括无关信息源而不要丢失重要的相关信息源．此外，最好快速地计算此相关性确定，而无需开销大的处理技术．

在一较佳实施例中，查询路由器计算每个信息源的数值相关性等级值，该值用于估计源的相关性．此计算是基于"概念类(conceptual class)"这一概念．因而，预先给每个信息源添加与其相关的概念类的标记．然后，查询路由器把每个查询映射到与其相关的概念类，并找到具有与该查询共享概念类的信息源．最好以散列函数来把查询映射到其概念类．

聚合引擎

聚合引擎是综合器模块的协调函数．它接收来自用户界面模块的用户查询并请求查询路由器来提供与给出的查询最相关的 N 个信息源的清单．然后，它检索来自包装器数据库的 N 个信息源的 N 个包装器．由 N 个包装器所引导，聚合引擎把此查询翻译成可被 N 个信息源中每一个所接受的请求格式，并把这 N 个请求转移到 I/O 管理器以备网络传输．对于某些源，请求可以是待返还的格式．当从一信息源接收到一响应时，适当的包装器再次引导聚合引擎从该响应中提取数据，并把该数据置于与特定信息域相关的数据字段(叫做元组格式 - tuple format)的清单中．或者，可使用适用于特定信息域的方法给每个元组赋予优先级．最后，当增量显示管理器请求把数据显示给用户时，聚合引擎在已确定优先级时按照优先级的顺序把这些元组传递到用户界面模块．

例如，如果信息域与因特网联机软件供应商相关，则元组可选择地包含诸如产品名称、制造商、软件版本号、所需的操作系统、价格等相关字段．可以根据用户的喜好，由价格、交付延迟或其它因素来决定元组的示例优先级．

对于进一步的例子，如果信息域与万维网("WWW")搜索引擎(一般可在 WWW 上获得其索引信息页面)相关，则元组可选择地包含诸如索引页面的标题、该页面的通用资源定位器("URL")、估计索引页面与该查询的关系的相关性分值、描述性文本等相关字段．示例的优先级可根据网络机器人对索引页面与查询的相关性归一化估计．如果网络机器人不检索索引页面，则它对从每个搜索引擎返还的该响应的归一化相关性估计值求和．如果搜索引擎不返还一相关估计值，

则使用默认值．然后，通过对所返还的分值进行线性调节到具有例如 1000 的公共最大值，然后把经调节的分值与可信度因子相乘，从而对获得的相关性估计值进行归一化．此可信度因子的范围从 0 到 1，它是特定信息源自身相关性估计值的可靠性的预定估计值．例如，可通过对信息源的相关性估计值进行实验来确定该因子．或者，用户可请求网络机器人来检索该页面，以进行其自身的相关性估计．在一示例实施例中，对于请求出现所有查询字或任何查询字的查询，通过扫描页面并计算实际出现的查询字的数目，然后对此计数进行定标，使出现所有的字导致公共的最大相关值，这样来确定此估计值．对于请求出现短语的查询，例如通过从公共最大值中减去在该页面中短语的每个字离开该短语中后续字的距离平方的归一化总数来确定此估计值．从而，如果该短语在页面中连续出现，则相关性高，而如果该短语的字在页面中散布得很开，则相关性低．

总之，对于大多数信息域，从例如在 WWW 搜索引擎的例子中所进行的相关性计算来确定优先级．然而，对于诸如联机软件供应商等某些领域，可简单地从响应元组的一个或多个数值字段的值中来确定优先级．

包装器数据库

描述信息源及其能力，尤其是在其查询格式和响应格式中，较佳方式是以称为包装器的压缩的模块化说明性描述．由于机器人可访问几百到上千个信息源，所以对这些源的描述最好是压缩的，使其只需要最少的存储量．此外，由于新的信息源频繁地产生且现有的源也频繁地改变其格式，所以方便地维持源的描述是重要的．模块化的说明性描述而非复杂的过程化描述有助于这种维护．在本发明的一个实施例中，可由用于具有足够规范格式的信息源的独立模块来学习包装器．

对于每个信息源，在示例实施例中，每个包装器最好包括以下信息:

1．信息源的通用资源定位器("URL")地址;

2．源的概念类;

3．对从例如字或短语等查询自变量映射到用于询问该源(包括对任何、所有、短语或邻近查询的部分(site)支持)的查询字段或 HTML 所定义的形式的描述;

4．对能从其它信息和外来格式化素材中分析相关信息的查询响应或 HTML 页面布局的格式的描述．

最好至少以本发明的包装器描述语言来编写第 3 和第 4 项.

网络机器人可以各种方式来检索包装器. 在一个实施例中, 信息源本身可在网络机器人请求时提供该信息源自己的包装器. 在另一个实施例中, 网络机器人可以各种方式来提供了它自己的包装器. 例如, 可在网络机器人内建立包装器, 尤其是在网络机器人只访问少量信息源时. 此外, 可把包装器存储到本地数据库中或可根据来自中央数据库的命令而被下载.

本发明的包装器描述语言(以下叫做"WDL")有利于通过使用组合了来自文法和规范表达式的特征的说明性描述格式对查询、形式和页面进行语义上的描述. 这里示出这种描述语义的一个例子. 在后面部分对其进行详细描述. 所使用的语法遵循本领域内的技术人员所公知的用于规定文法(包括规范表达式)的约定. 例如, 见 Schwartz, 1993, Learning Perl, O'Reilly & Associates, Inc., chapter 7 ; Aho 等人, 1986, Compilers Principles, Techniques, and Tools , Addison Wesley Publishing Co., section 3.3 .

以下是从 WWW 搜索引擎返还的典型页面的示例 WDL 描述. WDL 的解释程序使用页面描述来分析页面并执行任何指定的动作语句. 注意, "stuff(要素)"是 WDL 中所保留的一个字, 它使任何字符串与第一次出现的强制下列串文字匹配.

```
<page>::=stuff "<dl>" <item>* .*$
<item>::=stuff "<dt>" stuff "=\"" (stuff) "\"><strong>"
        (stuff) "</strong/></a>" stuff "<dd>" (stuff)
        "<br>" {output ($0, $1, $2, 500)}
```

它描述了页面除了其它数据以外还由零个或多个项目的序列构成. 更具体来说, 规定页面由 stuff 、串<dl>、零个或多个项目、零个或多个字符(".")以及最后由页面的结尾("$")来组成. 一般, 一个项目包括以(stuff)所标记并依次叫做$0 、$1 和$2 的三个相关性字段, 当一个项目被识别时以"output( )"语句来输出这些字段. 具体来说, 规定一个项目由 stuff 、串<dt>、多个 stuff 、串'="'、以下以$0 所引用的第一重要字段、串'">'、以下以$1 所引用的第二重要字段、串""、多个 stuff 、串"<dd>"、以下以$2 所引用的第三重要字段、串"
"所组成. 当一<item>匹配时用联编(bindings)来规定变量$0 、$1 和$2 的输出的情况下, 在识别每个项目时执行括号内的动作语句.

网络机器人系统

可把一个网络机器人的功能结构以各种方法分配给系统硬件部件．在任何情况下，较佳的选择方法取决于哪一个功能定位实现快速的响应和合理的成本．图4针对先前的总体描述从总体上示出示例网络机器人的硬件实施例和选项．它示出用户计算机单元51-56、网络57、信息源58和网络机器人服务器计算机59-61的相互关系．计算机51是一用户计算机，它包括处理器、存储器和各种相连的外围设备．这些外围设备包括显示器件52或与用户进行交互动作的其它器件、网络附属件54、可选的硬盘存储器53等．计算机51还可以是没有永久存储器的网络器件、PC、工作站或更强大的计算机．计算机51最好是运行Windows操作系统、Macintosh操作系统或UNIX之一的PC或工作站．用户计算机51的存储器中除了有其它软件以外，还有本地网络机器人软件55和本地系统部件56．本地网络机器人软件实现一个或多个网络机器人功能．本地系统部件例如可包括Web浏览器．

网络57可以是具有多个与其相连接的信息源58的任何网络，可以根据主题在概念上对这些信息源58进行任意分类而成为信息域．在一较佳实施例中，网络57是支持包括诸如FTP、HTTP等用户层协议的TCP/IP协议的公共因特网或专用企业网．这些信息源是服务器计算机，它们通过使用网络57所支持的协议而获得它们所存储的信息．这些信息可包括期刊、报纸等的数据库、与特定商业、教学或其它类型的组织有关或产生的信息，用于电子商务的工具等．

在这种网络中，网络机器人可具有各种实施例．在一完全本地的实施例中，所有的网络机器人功能放在用户计算机51上的本地网络机器人软件55中，在该实施例中，用户计算机必须具有足够的处理和存储能力．在另一个实施例中，一个或多个所揭示的网络机器人功能可分布于其它挂在网络上的计算机中．

例如，计算机59是接受用于从其包装器数据库下载包装器的请求的包装器服务器．可使用能存储和检索压缩的文本描述的任何数据管理系统来把包装器数据库存储在存储器或盘片上．计算机60是通过接受查询并从存储有与其有关的信息的几万或几十万个信息源中返还N个最相关的信息源从而来查询路由选择的查询服务器．计算机61是通过接受用户查询并返还搜索结果以进行综合器模块功能的网络机器人服务器，它多半要使用包装器服务器59和查询服务器60的设备．由这些网络服务器，本地网络机器人软件最好只支持可被完全委托给Web

浏览器的用户界面．或者，它还可包括向查询服务器 59 进行查询路由选择请求并向包装器服务器 60 进行包装器请求的聚合引擎．此外，它可包括后面两个功能中的一个功能或这两个功能．

可给网络机器人系统的各种计算机提供来自计算机可读媒体或通过网络安装的用于实现本发明的方法的软件．本发明适用于诸如磁盘、磁带和 CD-ROM 等公知的磁性和光学媒体．

5.2. I/O 管理器

I/O 管理器模块为综合器模块实行硬件、操作系统和网络专用连接．网络连接包括发送请求并接收来自与网络连结的信息源的响应的任务．本发明的网络机器人的一个重要应用是在因特网上进行信息检索．在此应用中，I/O 管理器负责实行 WWW 、 Gopher 、 FTP 、因特网工具等的相关协议．或者，为了提高响应时间，I/O 管理器可暂时高速缓存页面和其它数据．

操作系统连接包括对用户界面模块进行窗口管理并访问可能存在的包装器数据库的任务．

I/O 管理器最好由诸如 Java.awt 软件包等市售的协议堆栈、窗口库和其它工具构成．在某些实施例中，可由挂在网络上的计算机的其它系统部件来进行或多或少的 I/O 管理器功能．或者，把 I/O 管理器设计成可被定标成多个装置，且不需要多线程或可重入代码，以及设计成为跨平台的和持久的．

5.3. 聚合引擎

在较佳实施例中，通过以下流程来进行先前对本发明的聚合引擎元件所述的功能．平行地进行对信息源的搜索，这是因为传输所有的请求而无需任何响应．

1．接收来自用户界面模块的用户查询;

2．进行查询路由选择，以确定与此用户查询最相关的 N 个信息源;

3．对于 N 个相关信息源中的每个信息源，实行:

A．检索该信息源的包装器(例如，从该信息源本身或从本地或远程包装器数据库);

B．由包装器来引导，把用户查询格式化成为该信息源所需的的形式或格式;

C．把经翻译的命令转移到 I/O 模块以传送到此信息源;

4.将响应清单进行初始化成空清单;

5. 直至达到用户指定时限，实行:

A. 当信息源响应已被 I/O 模块接收到并转移到综合器时，则:

i. 由该信息源的包装器所引导，分析该响应来理解所返还的信息，丢弃专用于地点(site-specific)的格式化文本和其它无关的素材，并使相关字段集合成元组;

ii. 把每个元组加到元组清单中，或者进行优先级排列、复制、除去等;

B. 等待下一个响应;

6. 可按请求把所产生的元组的清单交付给用户界面模块，例如该请求可通过用户的 show-me-button 或 more-button 控制键而启动.

在查询多个信息源时，最好在步骤 6 给用户界面模块继而给用户显示从这些响应中所提取的元组的单个合并清单，该清单是依据每个元组对用户的重要性或相关性的估计来排序的. 最好依据专用于网络机器人所指向的信息域的方法来进行这样的估计. 对于某些领域，可从元组中一个或多个数据字段的值中直接进行此重要性估计. 例如，在电子购物的领域中，依据用户的喜好，重要性只与价格或交付日期相关. 然而，对于大多数领域，依据所返还信息的相关性来进行此重要性估计，这必须通过检查来自每个源的响应来确定.

在此相关性确定的一个较佳实施例中，用户有权选择是否让网络机器人自己检查所有的信息页面. 如果用户如此选择，则由网络机器人依据领域专用 Analyze 函数来确定相关性. 在信息查询的领域中，示例的 Analyze 函数在所返还的响应中找到查询字的数目和文字. 对于关键字查询，存在频率较大的较多关键字的响应较相关. 对于基于短语的查询，短语的字较靠近的响应有更大的相关，例如这些字在一个句子中或者甚至是连续的. 在其它领域中，提供适当的 Analyze 函数.

如果用户选择不让网络机器人来检查这些响应，则网络机器人依赖从信息源返还的相关性估计值. 如果特定的源未返还相关性估计值，则使用默认值. 然后把这些估计值归一化到如 0 和 1000 之间，并与可信度排列因子相乘. 此可信度因子的范围从 0 到 1，它是对特定信息源本身的相关性估计值的可靠性的预定估计. 例如，可通过对信息源的估计值进行实验来确定该因子. 当从两个或多个源返还相同的元组时，把来自所有这些源的相关值相组合. 或者，调节从每个源返还的相关性估计值而使它们在归一化范围内具有均匀的分布.

在一特定的详细实施例中，依据以下流程来进行此确定. 这里，查询路由选

择已确定 K 个信息源 source_k 构成的清单并已返还其可信度等级 crank_k， k 从 1 到 K．这些源中的每个源已被查询并返还了响应，且已从这些响应中提取了信息源元组 tuples_j 构成的 K 个清单，这里 j 从 1 到 length_k．以验证标志 V 来记录用户对网络机器人分析的喜好．变量 t.score 代表元组 t 的合成相关性分值；变量 t.sourcescore_k 代表对于从中提取元组 t 的响应，从：信息 source_k 所返还的相关性估计值．

输入：从查询路由选择系统获得的 K 个源与其可信度等级对(source_k, crank_k)的清单；从源 source_k 获得的长度 length_k 的元组 tuples_k 的 K 个排序清单；以及从用户喜好获得的验证标志 V(布尔)

输出：以相关性排序的所有元组的合并清单．

```
/*主程序*/
IF (V is true) THEN
  FOREACH k=1..K
    FOREACH tuple t in tuples_k
      page=the HTML page that tuple t points to,
        download if necessary
      t.score=Analyze(页面);
ELSE /*Vis false*/
  FOREACH k=1..K
    NormalizeSores(tuples_k)
    AdjustByHeight(tuples_k)
    AdjustByServiceRanking(tuples_k)
  /*把结果元组 t 合并到 MERGED_LIST 中，并从信息源所返还的分值 t.sourcescore_k 中确定合成的相关性分值 t.score；把从多个源返还的相同元组的合成分值递增来自每个源的源分值*/
  FOREACH k=1..K
    FOREACH tuple t in tuples_k
      IF t is not in the MERGED_LIST THEN
        Add t to the MERGED_LIST
        t.score=t.sourcescore_k
```

```
            ELSE
                t.score=t.score+t.sourcescore_k
            ENDIF
        ENDIF
        SORT all tuples t by t.score and discard duplicates
        OUTPUT sorted tuples
        EXIT /*完成相关性排列*/
        /*子程序*/
        SUBROUTINE NormalizeScores (tuples_k)
        /*如果 information source_k 返还相关性估计值，则进行归一化而使它们落在
0 到 1000 的范围内；否则，使用默认的相关性估计值*/
        /*"s"是来自 source_k 的元组清单上第一个元组的第 K 个信息源的相关性
分值*/
        s=tuples_k[1].sourcescore_k
        IF (s==0) THEN
            /*此信息源不返还分值，因此使用默认值*/
            FOREACH each tuple t in tuples_k
                t.sourcescore_k=1000
        ELSE
            scaling_factor=1000.0/s ;
            FOREACH tuple t in tuples_k
                t.sourcescore_k=t.sourcescore_k*
            scaling_factor ;
        ENDIF
    ENDSUB
    SUBROUTINE AdjustByHeight(tuples_k)
    /*把源分值调节到具有均匀的百等分分布；对于 10 个元组的例子，把第一元
组的源分值调节到其源分值的 100%, 把第二元组的源分值调节到其源分值的 90%
等*/
        percent_step=100/Length_k ;
```

```
    percent_off=0 ;
    FOREACH tuples_k 中的元组 t
        t.sourcescore_k=t.sourcescore_k*(100-percent_off)/100 ;
        percent_off=percent_off+percent_step ;
ENDSUB
SUBROUTINE AdjustByServiceRanking(tuples_k)
```

/*给每个服务分配表示对其返还的源分值给出的可信度的百等分排列；此排列用于对所返还的每个元组的源分值进行定标；例如，90%的可信度排列意味着每个源分值被定标 0.9*/

```
    FOREACH tuple t in tuples_k
        t.sourcescore_k=t.sourcescore_k*crank_k
ENDSUB
```

本领域内的技术人员应知道，可把这些流程变为以相同的方式实行相同的功能的常规变化和改进．尤其是，可使用归一化范围的其它值和信息源相关性估计的默认值．本发明包括这种程序上的变化．最好以 C++来实现这些流程，但它们也可以用任何过程编程语言或面向对象的编程语言来实现．

5.4. 查询路由器

查询路由器接收表示为字或关键字清单的用户查询作为输入，并返还以与输入查询的相关性来排序的 N 个信息源的清单作为输出．以速度和广泛性来优化这些信息源的确定．宁可偶尔包括无关信息源向不要忽略相关的源．

较佳的查询路由器是基于把相关概念分配给信息源和查询字的原理．预先选择一组概念来描述一个或多个网络机器人所指向的一个或多个信息域的信息源．对于这些域中的每个信息源，判断该源是否与每个选中的概念相关．此外，检查在可能的查询中出现的每个字，以确定哪一个选中的概念与该字相关．然后，在接收到一查询的字或关键字时，确定与这些字相关的概念，然后找到与这些概念相关的信息源．通过组合该源与该查询的所有概念的各个相关性来确定每个源的排序的相关性．最好通过对此查询类型产生分开的数据来处理以短语为基础的查询的情况．

此流程的较佳实施例利用包含相关性信息的四个表．以下，选择 W 比可能的查询中所出现的字的数目大一些，例如 10%； C 为选中概念的数目； S 为信息

域中信息源的数目． WORD2CONCEPT[ ]是 W 个 C 位(bit)矢量的表，这里一个字的矢量的 C 位表示哪 C 个概念与该字相关． CONCEPT2SOURCE[ ][ ]是一 C 乘 S 的表．对于 C 个概念和 S 个源中的每一个，该表的相应表目包含此源与此概念的相关值．例如，如果表目<i,j>等于 5，则第 j 个信息源相对于第 i 个概念的相关权重为 5．在以字进行搜索时使用 CONCEPT2SOURCE[ ][ ]．同样，对于以短语进行搜索，表 CONCEPT2PHRASE[ ][ ]也使概念与源相关．最后，DEFAULT-RELEVANCE[ ][ ]具有对于 S 个信息源中每一个的默认相关权重．

本较佳实施例实行以下流程．

1．对于 S 个信息源中的每一个，设定 RELEVANCE[j]=DEFAULT-RELEVANCE[j];

2．对于用户查询中的每个字，实行:

A．对该字计算散列函数，从而获得位于零和 W 之间的数目 M．任何适当的散列函数都适用于此流程．在 Sedgewick, 1990, Algorithms in C, Addison-Wesley PublishingCo., Chapter 16 中找到一示例的散列函数．

B．令 C 位矢量 V 等于 WORD2CONCEPT[M];

C．通过进行以下程序而把 V 中所有概念的相关性与这些信息源的相关性相组合:

```
For i from 0 to C, do :
If (i-th bit of V is '1') THEN
    FOR j=1 to S DO
    RELEVANCE[j]=RELEVANCE[j]+CONCEPT2SOURCE[i,j]
```

还可使用不同于"+"的单调递增函数来把各个概念相关组合成最终的相关;

D．例如通过把用户查询中所有字的相关加在一起来组合这些相关．

3．在以短语进行搜索的情况下，另外要实行:

A．串连用户查询短语中所有的字;

B．对该短语计算散列函数并获得 M，设定 C 位矢量 V 等于 WORD2 CONCEPT[M];

C．通过进行以下程序而把 V 中所有概念的相关与这些信息源的相关相组合:

For i from 0 to C，实行:

If (i-th bit of V is '1') THEN

FOR j=1 to S DO

RELEVANCE[j]=CONCEPT2PHRASE[i,j]

4. 根据这些信息源的 RELEVANCE 对它们进行排序，并返还 N 个最相关的源.

本领域内的技术人员应知道，此过程适合于以相同的方式实行相同的功能的程序的变化和改进．本发明包括这种程序上的变化.

最好以 C++来实现此流程，但它也可以任何过程编程语言或面向对象的编程语言来实现．在查询路由器保存有与大量(例如几万)个信息源有关的信息的情况下，最好以服务器计算机上的服务流程来实现此查询路由器，以适应查询路由选择所需的数据结构和处理要求.

表 WORD2CONCEPT 的示例构造以选择表示所考虑的信息域特征的概念并确定可能在用户查询中产生的字和短语开始．对于每个概念，进行以下操作．把与该概念有关或此概念所涉及的字和短语赋给串数组 KEYS[ ]以及 PHRASES [ ]. 然后，实行以下流程.

FOR i equals 1 to the number of element in KEYS[ ], DO

Apply the previously used hash function to KEYS[i] to obtain a number between 0 and W

SET the bit matching the current concept in WORD2CONCEPT[M].

FOR j equals 1 to the number of elements in PHRASES[ ], DO

Apply the previously used hash function to KEYS[i] to obtain a number between 0 and W

SET the bit matching the current concept in WORD2CONCEPT[M].

对每个选中的概念重复这些操作．或者，为了给概念匹配保持正确的串，可使用开放式或封闭式散列来存储概念信息以及串信息.

5.5. 包装器定义语言

这一部分首先提出使用包装器中的 WDL 时的引导性资料．接着，详细地描述 WDL 的两个主要成分-动作语言和规范表达式语言．最后，提出 WDL 的示例实施例.

包装器是对信息源的描述，它描述了网络机器人应如何与信息源进行交互动

作，尤其是如何对该信息源的请求进行格式化以及如何理解来自信息源的响应．网络机器人需要与网络上的成百上千个信息地点进行有效的交互动作．此交互动作提出了两个要求：第一，压缩地存储使用以 WDL 编码的表示对信息源所进行的描述；第二，使用此描述来理解该信息源．例如，目前网络机器人需要把对一源的请求进行格式化并从该源所返还的页面中分析有用的信息，同时忽略无关的格式化信息．随着信息源变得更加功能化，网络机器人将需要处理比简单的请求-响应对更为复杂的交互动作．因此，包装器和 WDL 最好具有以下特征：

1) WDL 描述比例如 C++中的描述更容易编写．这是重要的，因为新的信息源频繁地产生且现有的服务频繁地改变其格式．或者，也可使用信息域(其中的响应具有规范的预定格式)中的机器学习技术而自动地产生包装器描述．

2) 包装器描述很少．这是重要的，因此，例如可把它们存储在本地的数据库中，或者即使经由一慢的网络连接也可把它们从一服务器快速地传送到在一客户端运行的网络机器人处．或者，信息源可被请求时提供它们自己的包装器．

3) 可自动地把 WDL 描述编译成快速的有限状态自动机，该自动机可快速地分析由信息源所返还的信息．

4) 使用具有 WDL 的包装器使得网络机器人可适应于未来的新型信息格式和新型的信息服务器交互动作．

包装器描述最好规定至少两个流程：第一，向信息源请求信息，例如，如何从一源中取得适当的经 HTML 格式化的页面；第二，如何分析返还的页面来提取相关数据．为了进行第一个流程， WDL 包括动作语言成分，它是可扩展的表达式和语句的语言．为了进行第二个流程， WDL 包括规范表达式语言成分，它是规定规范表达式模式匹配工具的广泛的新手段．

在另一个实施例中，网络机器人可利用为本领域内的技术人员所公知的其它模式匹配工具．例如，可以用上下文无关语言规约("CFL")来替换规范表达式成分．在此情况下， WDL 的实现可遵循构成编译器的结构的公知技术，例如 YACC．然而，在可能的情况下，规范表达式模式匹配是较佳的，这是因为其规约的明确性以及快速的执行．

规范表达式成分的一个例子

规范表达式有利于描述从许多信息源返还的信息的格式． WDL 的规范表达式成分以几个新的方式把以前的规范表达式匹配工具予以扩大．第一，它使得可

在识别规范表达式时执行例如语句和表达式等动作语言的编程语言工具，并且具有以整个识别期间所识别的局部匹配来确定的变量联编．第二，此较佳实施例把规范表达式编译成为压缩而有效的有限状态自动机．第三，它鼓励以嵌套的方式来有效而直观地表达复杂的规范表达式．第四，它具有有效的无回溯(backtrack-free)填充方法．

示例的包装器适用于因特网信息搜索引擎并以 WDL 来编写．以备注来说明每个部分的重要性，备注用"/*"和"*/."括起来．

/*把输入的查询字的清单从聚合引擎传递到包装器，该聚合引擎执行该信息源的包装器．动作语言的 argv( )函数提取该输入查询字的清单．*/

{$keywords=argv(2)};

/*通过串连(concatenating)(以动作语言的算符"."来表示)三个分离的串来计算待发送到信息源的请求：一，信息源的因特网 URL 地址以及最初的查询格式串；二，要搜索的查询字；三，该信息源的查询格式串的其余部分；以及*/

$url="http://searcher.source.com/searcher.cgi?query=".$keywords."&onlyrr=0"
;

/*fetch 动作语句将该查询送到用于网络传输的 I/O 模块，然后等待经 HTML 格式化的响应．*/

$page=fetch(0, $url, ' ');

/*使用以下的规范表达式文法来分析经 HTML 格式化响应文本．*/}

$result=parse($page, <page>); }

/*来自此示例源的响应由包含零个或多个信息项的页构成．这是用<page>和<item>的表达式以规范表达式语言分层方式来描述的，这里<page>指若干<item>．尤其是，一页包括后接串"results returned... "然后后接零个或多个项目的大量文本．*/

<page>::=stuff 'results returned, ranked by' <item> *END

/*每个项目由无关文本、HTML 格式化代码和相关数据字段构成．相关数据字段包含在括号中并依次以$0 、$1 和$2 来表示．当在确认一<item>时执行动作语言的"output"语句时，把这些字段以及数目 500 作为元组输出．<item>的定义的特定意义将在下一部分中变得十分明显．*/

<item>::=stuff '<hr>' stuff ' <b><a href="'

```
(stuff) '"' stuff '>' (stuff) '</a>'
stuff '<font size=-1>' (stuff) '</font>'
{output ($0, $1, $2, 500); } END
```

5.5.1. WDL 成分的描述

这一部分描述 WDL 的动作语言和规范表达式成分的较佳特征．本发明不限于所提出的描述．这些描述可利用用于描述文法包括规范表达式的公知方法，本领域内的技术人员应知道有许多基本上等效的描述．这种等效的描述包括通过对所描述的语法元素进行更名或者通过把公知的文法变换应用于所提出的语法而获得的那些，但不限于此．本发明也包括这些基本上等效的描述．

动作语言成分

动作语言包括某些较佳的基础级特征：赋值语句；诸如"if"和"while"构成的顺序语句；元组输出语句；串和数值表达式；串、数值和布尔算符；以及某些内部的函数．对本领域内的技术人员很明显的是，可以用与这些特征稍稍不同的选择来构成具有等效的语义表达能力的基础级动作语言．例如，可以 goto 语句来替换 while 语句．本发明的动作语言成分包括这些与所揭示的语义等效的组成．此外，在可选的实施例中，可以用这些附加的特征来扩大较佳的基础级特征，诸如："for"或"repeat"循环等附加的顺序语句；用户所定义的函数；诸如在 C 或 C++中出现的附加的串、数值和布尔表达式和算符；附加的内部函数；以及数组变量．可把这些附加的特征以公知的方式加到较佳的基本语言中．例如，见 Aho 等人在 1986, Compilers Principles, Techniques, and Tools, Addison Wesley Publishing Co. 中的描述．

通过以下由标准表示法来表示的文法来给出较佳的基本动作语言的语法．

```
Statement::=
    $VariableName "=" Expression ";"
    | "if" Expression Statement [ "else" Statement ]
    | "while" Expression Statement
    | "{" Statement* "}"
    | "output" "(" [ Expression {","Expression}* ] ")" ";"
Expression::=
    string-constant
```

| float-constant

| $variableName

| Expression op Expression

| "(" Expression ")"

| functionName "(" [Expression {","Expression]*] ")"

于是，根据语句和表达式来定义动作语言成分．一个语句可以是：一赋值语句，它给标量变量赋新的值；一 if 语句；一 while 循环；一复合语句，它是由"{"和"}"所包围的一系列其它语句；或者一输出语句．除了输出语句以外，语句函数的方式都与例如 C 、 C++或 Pascal 等其它过程编程语言中的方式相同．对于 if 和 while 语句，如果条件自变量是一非零数字或非空串(即不是空串""), 则认为该自变量为真． OUTPUT 语句使得包装器把在来自信息源的响应中匹配的信息返还到执行该包装器的网络机器人模块．例如，执行语句"output (arg_1, ..., arg_n) "使得将把元组<arg_1, ..., arg_n>从包装器返还网络机器人．

一个表达式可以是：一个串常数，它是由引号所包围的符号串；一浮点数；一变量名，它是前面有美元符号的一个名称或一个整数；一加到周围的两个子表达式的插入算符；一子表达式，它被包围在"("和")"中；或者对内部函数的一个调用。

此外，该语言还提供了过程编程语言中的算符标准，包括以下：算术运算符("+"、"-"、"*"、"/")；数值比较运算符("<"、"="、">"、"<="、">="、"!=")；串比较运算符("lt"、"eq"、"gt"、"le"、"ge"、"ne")；串并置运算符(".")；以及布尔算符("&&"、"||"、"!")．这些运算符具有以下语义：

1) +、-、*、/：对两边的表达式进行所指定的算术运算；

2) .：并置周围的串；

3) ==、<=、>=、<、>、!=：对两边的数字进行指定的数值比较并相应地返还浮点值 0.0 或 1.0；

4) eq 、 le 、 ge 、 lt 、 gt 、 ne：对两边串的 ASCII 码进行字符对字符的比较，并返回浮点值 0.0 或 1.0；

5) !：如果自变量为数字 0 或串""则返还 1 ，否则返还 0；

6) &&：估计第一自变量，如果第一自变量为零或""，则停止并返还该值，否则估计第二自变量并返还后者的值；

7) ||: 估计第一自变量，如果第一自变量不是零也不是""，则立即停止并返还该值；否则估计第二自变量并返还后者的值.

这些运算符从高到低依次具有以下优先级:

1. 括号内的任何表达式(最高优先级)

2. *、/

3. +、-

4. ==、<=、>=、<、>、!=、 ne、 eq、 le、 lt、 gt 、 ge

5. !(布尔非)

6. && (布尔和)

7. || (布尔或)(最低优先级)

所有的算符都是左相连的.

不需要变量说明. 第一，通过在前面加上"$"来区分动作语言中的所有变量. 代表以规范表达式来匹配的子串的那些特定变量由"$"后接一个整数来构成，例如，$0 或$1 . 第二，在运行时，以动态方式在串和浮点类型之间进行自动的类型转换. 如果算符希望数字而得到了串，则通过通用 C 库函数 atof( )把串转换成数字，该函数把数字的 ASCII 表示转换成其内部浮点表示. 如果算符希望串而得到了数字，则它使用 C 库函数 sprintf(...,"%f")把它转换成串. 第三，给还未赋值的参考变量设定适当的默认值 0 或"".

动作语言成分具有几个内部的函数. 以下是较佳的基本内部函数.

1) argc( )、 argv( ): 当网络机器人执行包装器时，它可给包装器传递一个或多个自变量. 这些自变量通常代表由用户所提供的查询参数、查询字或查询关键字. 通过返还传递到包装器的自变量数目的函数 argc( )以及返还第 n 个自变量的函数 argv(n)来访问包装器中的这些自变量.

2) fetch( ): 此函数最好与 I/O 模块相连，以依据适当的协议经由网络把包含信息源的网络地址和可能的查询参数的串转移到被寻址的信息源，并返还包含该信息源的响应的串. 包装器使用该函数来查询信息源并检索页.

3) parse(<string>,<nonterminal>): 该函数采用一个串并尝试把它与相应于给出的<nonterminal>(如 WDL 的规范表达式语言成分所定义)的规范表达式相匹配. 如果匹配成功，则该函数返还"1"，如果该串与规范表达式不匹配则返还"0". 包装器使用该函数来分析信息源的响应.

示例的包装器包括以下的一系列动作语言语句．首先是使用 argc( )和 argv( )的一系列语句，以获得用户查询参数并以对例如$url 等串变量进行初始化，使该串变量包含信息源中适当页的 URL 以及经适当格式化的查询．接着，赋值语句"$html_text = fetch ($url)"把查询-响应页引入例如$html_text 等另一个串变量中．最后，函数 parse ($html_text,<page>)尝试把返还的 html 文本与描述被搜索页的规范表达式<page>相匹配．

规范表达式语言

WDL 的规范表达式("reg-exp")成分把串与规范表达式相匹配．已发现，规范表达式便于描述从大量信息源返还的响应的格式．然而，本发明的 reg-exp 语言能如此匹配该信息，从而可以比诸如 AWK 或 Perl 等以前的语言和系统更快而且更方便的方式来提取相关字段．规范表达式成分包括解决以前匹配系统的问题的新工具从而使它可被网络机器人所使用．

第一个新工具使得以类似于上下文无关文法(它通过用于文法中非终结点的一组规则来规定)的方式来分解规范表达式的规约．例如，见 Aho 等人在 1986, Compilers Principles, Techniques, and Tools, Addison Wesley Publishing Co., section 4.2 中的描述．在以前的系统中，需要编写单个规范表达式而表示来自信息源的页的格式，这样常导致该表达式非常巨大而复杂，这样的表达式难于编写、理解和维护．为了解决现有系统的这个问题， reg-exp 成分通过用于规范表达式成分的一组规则来规定一规范表达式．以非终结点来标注这些成分．然而，与上下文无关文法相反，不允许 reg-exp 成分中的这组规则是递归的或者相互递归的．换句话说，用于一特定非终结点的规则不能直接或间接地涉及与该特定非终结点有关的其它规则．

以下示例使用一组规则和非终结点．定义信息响应的一个顶层非终结点可以是:

<page>::=<head> <item>* <tail> END.

它规定了该响应是由标题(head)，接着是零或多个项目，然后尾部所构成的．关键字"END"代表规则的结尾．该规则的右边项(“ RHS ”)的第二层非终结点<head>、<item>和<tail>由它们自己的规则来定义:

<head>::="Results of your search:\n" END ;

<item>::="Data: ...\n" END ;

<tail>::="No more results\n" END.

为了执行这些规则， reg-exp 成分编译器把用于<head>、<item>和<tail>的规则的 RHS 代入<page>的 RHS．这个结果就好象包装器包含巨大而复杂的合成顶层规则:

<page>::="Results of your search:\n" ("Data: ...\n")* "No more results\n" END

如果第二层规则的 RHS 部分包含更多的非终结点，则编译器将继续进行适当的代换，直到顶层规则的组合规则的 RHS 上不再有非终结点．由于没有递归或相互递归，所以该代换终结．

第二个新的功能是允许跳过串中的字符组而不进行回溯．在以前的规范表达式匹配系统中，这种普遍要求的实现方式需要在一堆栈上存储许多回溯点，因而可能导致广泛的回溯．例如，已有技术中与任何字符的任何次出现匹配的 Perl 表达方式(idiom)".*"或者与所有的非数字字符直至第一次出现的数字匹配的 Perl 表达方式"[^\d]*\d 可在匹配期间引起广泛的回溯．由于应对信息响应进行快速地分析，所以它并不是高效的而且也不是较佳的．例如，见 Schwartz 在 1993, Learning Perl, O'Reilly & Associates, Inc., chapter 7 中的描述．为了解决这个问题， reg-exp 成分对此通用表达方式引入了简单而直接的语法:

stuff "literal-string"

这里"stuff"是一保留的字．把 stuff 定义为与当前字符中的所有字符匹配，直至(但不包括)第一次出现的串"literal-string"，它是一串文字．该结构使得可以紧凑、高效而与回溯无关地来实现此通用且重要的表达方式．

第三个功能是允许从任意复杂的规范表达式中提取相关数据字段．可把动作语义语句嵌入非终结点的 RHS，以便在与该非终结点匹配且每当出现该非终结点的匹配时现行的变量联编时执行．在先前的例子中，可如下扩展<item>的定义:

<item>::="Data:" (stuff) "\n" {output($0);} END.

每当<item>被匹配时，把$0 设定为与括号中的 stuff 匹配的串，然后执行输出语句．在此情况下，$0 被联编到后接"Data:"的任何字符并下一行字符之前．

例如 AWK 和 Perl 等以前的系统没有这种功能．虽然它们也设定诸如$0 、$1 等变量，但它们只在与定义<page>的单个合成的规范表达式匹配后才这么做．换句话说，整个页是在变量设定前匹配的．因此，在 AWK 和 Perl 中，如果在<page>上没有一个以上的<item>，则除了最后的一个<item>以外将丢失所有的相关数据

字段. WDL 的 reg-exp 成分通过使执行多次(每次对应于每个非终结点匹配)的动作的规约，每次带有不同的变量联编，来解决这个问题.

现在转到描述 reg-exp 语言，它包括某些较佳的基础级特征：对非终结点规范表达式的定义；在规范表达式规则中嵌入动作语言语句；用于表达交替和重复的算符；以及文字串匹配. 对本领域内的技术人员很明显的是，可以用稍稍不同的特征选择来构成具有等效的语法表达能力的较佳的基础级 reg-exp 语言，且 WDL 的 reg-exp 语言成分包括这些公知的等价物. 尤其是，reg-exp 语言包括应用于以下规则的变量更名和公知的文法变换.

在可选的实施例中，可以用诸如附加的特征来扩大较佳的基本特征，诸如：特定逻辑加(disjunction)("||")；重复任意整数；匹配到字符类；本地串存储；以及固定(anchor)字符. 这些附加的特征都是标准的，就除了特定逻辑加以外，它使得所列出的选择对象(alternative)的匹配与串分析同步，返还第一选择对象. 相反，规范逻辑加以列出的顺序匹配每个选择对象，进行回溯，直到成功. 可把附加的特征以公知的方式应用于基础级语言. 例如，见 Schwartz 在 1993, Learning Perl, O'Reilly & Associates, Inc.; Aho 等人在 1986, Compilers Principles, Techniques, and Tools, Addison Wesley Publishing Co.; Hopcroft 等人在 Introduction to Automata Theory, Languages, and Computation, Addison Wesley Publishing Co.中的描述.

在另一个实施例中，为了提高规范表达式匹配的性能，可加上说明来控制回溯. 如果已知规则的某些部分不需要回溯，则可把该部分与说明一起用括号括起来，从而告诉编译器和解释器来产生有限状态机，而不提供任何回溯. 这使得能对规则的这一部分进行更有效地匹配.

通过以下由标准表示法所表达的文法来给出较佳的基础级动作语言的语法.

```
Rule        ::= <nonterminal> "::=" Regexp "END"
Regexp      ::= Sequence ("|" Sequence ) *
Sequence    ::= Repetition+
Repetition  ::= Term "?"
              | Term "*"
              | Term "+"
```

```
                | Term
Term            ::= String_In_Double_Quotes
                | "stuff"
                | "(" Regexp ")"
                | <nonterminal>
                | Action
Action          ::= "{" Statement* "}"
```

概括地说，规则规定了识别特定规范表达式的特定<nonterminal>．一个规范表达式可包括：逻辑加("|")；顺序；零个或多个重复("+")；一个或多个重复("+")；以及零个或一个重复("?")．检索词(term)可以是：由双引号所包围的文字串；特定符号要素；加括号的规范表达；必须以另一个规则来定义的非终结点；或者以动作语言编写的语句．

包装器描述语言

一完整的包装器描述可以是：

Wrapperdescription ::= Statement Rule*

于是，较佳的 WDL 实体包括动作语言的语句，它通常是一系列动作语言语句，然后是 reg-exp 语言的可选的一组规则，它定义了用于匹配从信息源返还的响应的规范表达式．

为了执行完整的包装器描述，如后面的描述来执行该语句．一般，该语句还包含对内部函数 fetch ( )和 parse( )的调用．parse ( )内部函数尝试通过引用由所附的规则(每个规则定义了一个规范表达式)所定义的非终结点来匹配由 fetch ( )返还的响应．如果规范表达式匹配成功，则执行一般被嵌入规范表达式中的所有的动作语言语句，执行多次嵌入较低级非终结点的某些动作语句，在每出现一个匹配时有操作对象(operand)联编，parse( )函数返还值"1"．如果匹配不成功，则不执行嵌入的语句，且 parse( )函数返还值"0"．

5.5.2. WDL 成分的实施

在本部分中在以下标题下来描述 WDL 的较佳实施例：(1)分析规范表达式规则、(2)规范表达式的中间代码产生、(3)规范表达式的运行时间解释、(4)动作语言代码产生和运行时间解释．可以理解，WDL 的这个较佳实施例是一个示例．本领域内的技术人员可知道，这里所述的实施例存在其它选择的变化．例如，可

以不同的方式来实施所述的流程，例如以不同的变量和各个步骤的不同次序．此外，这些流程可实施其它的算法而对例如串匹配实现相同的效果。对于另一个例子，所述的实施例揭示了解释各种中间代码的处理过程．可使用不同的中间代码．对于规范表达式语言，不同类型的节点可包括例如具有后继状态的变量清单的节点，从而避免保持后继状态的回溯堆栈．对于动作语言，替换所揭示的基于地址的中间代码，等效地还可使用基于堆栈的逆波兰(Polish)中间代码．最后，替代解释，还可使用所揭示的语法指向的方法来直接编译成机器语言．

在这一部分的其余部分，标为"1"的分析树节点包含数据"d"，且具有以"1<d;c_1,...,c_n>"所表示的子节点"c_1,...,c_n"．可以本领域内公知的方式来构成和引用这些分析树节点，例如通过指向一包含节点数据的数据区的指针．

分析规范表达式规则

编译包装器描述的较佳实施例的第一步是分析 reg-exp 语言的规则和动作语言的语句．该步骤的输入是定义要匹配的规范表达式的 reg-exp 语言的规则．从该步骤输出的是具有一组分析树形式的中间代码，有一个代码用于顶层规则，其余的分析树用于每个较低层的规则．

在一较佳实施例中，通过依据递归递降分析程序进行分析并依据语法指向的翻译程序发出分析树来进行该步骤的流程．用于先前所述 reg-exp 语言的语法的递归递降编译器的构造对本领域内的技术人员来说是公知的．例如，在诸如 Aho 等人的 1986, Compilers Principles, Techniques, and Tools, Addison Wesley Publishing Co. 的 2.4 和 4.4 节等书中清楚地揭示了几个例子． 5.2 节中的例子包含了分析树的语法指向的构造．本发明不限于本较佳实施例．其它的分析技术在本领域内是公知的，本发明包括使用诸如 LL 或 LR 分析等这些技术的实施例．见 Aho 等人的文章的第 4 章．本发明还包括本领域内所公知的产生中间代码的其它技术．一般，见 Aho 等人的文章的第 5 章．

语法指向的分析规约包括在用分析程序识别每个语法规则时所产生的分析树节点的规则．对于每一个先前的 reg-exp 语法规则，产生以下由该规则的非终结点来标注的下列节点:

1. Rule：产生节点"Rule <nonterminal-name; node-for-regexp>."．这个节点用于以其非终结点名称来标注的整个规则．

2. Regexp：产生节点"Alternatives <node_for_sequence_1, ...,

node_for_sequence_n>.". 这是用于其它规范表达式的逻辑和("|")的节点，这些表达式按照所列顺序测试匹配，返还第一个成功的匹配.

3. Sequence : 产生节点 "Sequence <node_for_repetition_1 ..., node_for_repetition_n>.". 这是用于一系列规范表达式模式的节点，每个模式必须依次匹配.

4. Repetition : 产生节点"Repetition <type; node_for_term>.". 这是用于重复的节点，其类型是"?"(0 或 1 次重复)、"+"(1 或多次重复)或者"*"(0 或多次重复)中的一个.

5. String_In_Double_Quotes : 产生节点"String <literal_string>.". 这是匹配文字串的节点.

6. Stuff : 产生用于保留字"stuff"的节点"Stuff<>".

7. (Regexp): 产生节点"Sequence <OpenParenthesis<i>, node_for_Regexp, CloseParenthesis<i>>.". 这是使得在"Regexp. "进行匹配时给依次编号的变量赋值的节点. 节点"OpenParenthesis<n>"是代表在依次分析当前规则的 RHS 时碰到的第 n 个开括号的节点. 规则"CloseParenthesis<n>"是代表所碰到的相应第 n 个闭括号的节点.

8. <nonterminal>: 产生用于非终结点的一个实例的节点"Nonterminal <nonterminal_name>,".

9. Action : 产生节点"Action <node_for_action_language_statement>.". 这是用于动作语言语句的一个实例的节点，其中 node_for_action_language_statement 代表该动作语言语句的中间代码.

在分析程序确认相应的 reg-exp 语言语法规则时所执行的这些规则使得产生一分析树，它由以真实结构链接的所列节点类型而形成的. 把此分析树输入到该流程的随后步骤中.

产生用于规范表达式的中间代码

较佳实施例中的代码产生包括以下三个步骤:

1. 对要素节点进行特定的预处理;

2. 消除在顶层非终结点的 RHS 中出现的非终结点; 以及

3. 把顶层规则所获得的 RHS 转换成有限状态自动机("FSA").

首先将从总体上描述这三个步骤，然后对其进行详细描述.

首先，必须对要素节点进行预处理，以便在今后使用产生 FSA 的较佳后续遍历方法．由于不能单独对一个要素元建立 FSA，所以必须对要素节点和必须接在每个要素元后的文字串节点的组合建立 FSA，只能把开闭括号插入一要素节点及其后面的文字串之间．为此，对分析树进行遍历，且通过以新的 Stuff_And_String 节点替换两个节点每个要素节点与随后的文字串被合并．新的节点还考虑到任何插入的括号．

产生代码的第二步骤用相应的诸 RHS 来替换顶层非终结点的 RHS 中所有的非终结点，以对整个规范表达式产生单个组合的规则．在此替换期间，必须对动作语言语句中出现的变量重新编号．

例如，考虑包装器

<page> ::= ("dave") ("bill") <a> ("dan") { output($2); } END

<a>::= "oren" (stuff "cody") { output($0); } END

这里，<page>的输出语句中的$2 指("dan")，而<a>的输出语句中的$0 指(stuff "cody")．在替换后，如果不对输出语句中的变量重新编号，则<page>的 RHS 将扩展为:

<page> ::= ("dave") ("bill") "oren" (stuff "cody")

output($0); } ("dan") { output($2); }

然而，这里$0 指("dave")而不是(stuff "cody")，而$2 指(stuff "cody") 而不是("dan")．为了保持正确的变量赋值，必须如下所示对变量号码重新编号:

<page> ::= ("dave") ("bill") "oren" (stuff "cody")

output($2); } ("dan") { output($3); }

代码产生流程的较佳实施例的第三个步骤通过对代表顶层规则的 RHS 的分析树进行后续遍历(在此期间，依次把每个节点转换成 FSA)，把经处理和扩展的 RHS 转换成 FSA． FSA 具有几种类型的状态，最重要的是:

1. Char_branch： 依据当前字符转移至(branch)256 个可能的后继状态中的一个，并进到当前读取位置;

2. Action： 执行一批动作语言语句;

3. Marker： 在标记堆栈上记录加上或减去一定偏移的当前输入串位置;

4. Sussess： 在到达时，分析已成功;

5. Fail： 在到达时，当前的分析尝试已失败，必须把 FSA 回溯到先前的回溯

点，或者，如果不能获得回溯点则失败;

6. Push：把包括 FSA 状态和当前输入串位置的配置压入用于今后回溯的回溯堆栈.

代码产生步骤 1：预处理

通过扩展或弄平任何嵌套的序,(即嵌套在其它 Sequence<>元素中的 Sequence<>元素)，开始对要素节点进行较佳的预处理．这是依据以下流程进行的:

```
FOR nonterminal PERFORM a POSTORDER traversal of its parse tree
  At each parse tree node v
    IF (v=Sequence<c_1,...,c_n>) THEN
      WHILE (there is an i, 1<=i<=n, such that
      c_i=Sequence<d_1,...,d_m>) DO
        Replace node v with a new node
        Sequence <c_1,...,c_i-1, d_1,...,d_m, c_i+1,...,c_n>
```

接着，以同样考虑到以下文字串和任何插入括号的 Stuff_And_String 节点来替换出现的每个要素节点．在包装器中每出现一个"stuff"，后面必须接一个文字串，其语义结果是"stuff"与从当前位置直到(但不包括)第一次出现随后文字串的所有字符匹配．只有开闭括号可插入诸如要素和文字串之间．在较佳实施例中，最好依据以下流程来进行此替换:

```
FOR nonterminal PERFORM a POSTORDER traversal of its parse tree.
  At each node v of the parse tree
    IF (v=Sequence<c_1,...,c_n>) THEN
    WHILE (there is an i, 1<=i<=n, such that
    c_i=Stuff<>) DO
      Let j be the smallest j>i such that
      c_j=String<s> for some string s.
      IF there is no such j THEN signal an error.
      ELSE IF (any element ck (where i<k<j) is
            NOT either an OpenParenthesis or
            CloseParenthesis) THEN signal an error.
```

/*cj 是该串的节点*/

ELSE replace node v with a new node=

Sequence <c_1,...,c_i-1,

Stuff_And_String<c_j; c_i+1,...,

c_j-1>, c_j+1,..., c_n>

此流程把要素节点、文字串节点、 c_j 以及任何插入的括号节点、...c_j-1 合并成单个节点 Stuff_And_String<the-string ; c_i+1,...,c_j-1>.

代码产生步骤 2： 替换和变量更名

在完成预处理后，最好对顶层规则的 RHS 上的非终结点进行替换，并对任何动作语句的变量进行重新编号．替换和变量重新编号最好使用具有两个函数 EXPAND_REGEXP 和 EXPAND_ACTION 和一全局变量 PAREN_COUNT 的流程．从调用顶层非终结点开始，递归地调用 EXPAND_REGEXP 来替换 RHS 非终结点．把 PAREN_COUNT 初始化到 0，在替换进行时，每碰到一个"("时该变量就递增．这样， PAREN_COUNT 具有迄今为止碰到的"("总数的计数．然后，在替换期间，把碰到的括号重新编号，从其先前号码变到 PAREN_COUNT 的当前值，并以相应的方式把动作语言中的变量重新编号．例如，如果 PAREN_COUNT 目前为 8，则以 OpenParenthesis<8>节点来替换 OpenParenthesis<2>，并以输出($8)语句来替换输出($2)语句． EXPAND_ACTION 实行动作语句变量重新编号．最好依据如下所示来进行这些流程:

GLOBAL VARIABLE integer PAREN_COUNT ;

FUNCTION EXPAND_REGEXP(<nt>: nonterminal): RETURNS a parse tree with renumbered variables

/*整数数组具有新的变量号码*/

LOCAL VARIABLE new_names [ ]

LET T=parse tree for the RHS of rule for <nt>

PERFORM a PREORDER traversal of T.

At each node v of tree T

IF (v=OpenParenthesis<i>) THEN

new_names[i]=PAREN_COUNT ;

REPLACE node v with OpenParenthesis<PAREN_COUNT>

PAREN_COUNT=PAREN_COUNT+1

ELSE IF (v=CloseParenthesis<i>) THEN

REPLACE node v with CloseParenthesis<new_names[i]>

ELSE IF (v=Nonterminal<x>) THEN

REPLACE node v with EXPAND_REGEXP (x)

ELSE IF (v=Action<x>) THEN

REPLACE node v with EXPAND_ACTION (x, new_names[ ])

FUNTION EXPAND_ACTION (T: parse tree T for action language statement ; new_names [ ]): RETURNSa parse tree with renumbered variables

PERFORM a PREORDER traversal of the parse tree T.

At each node v,

IF (v denotes variable name $i(i an integer)

THEN

REPLACE node v with a new node with $i

replaced by $(new_names[i]).

代码产生步骤 3：产生有限状态自动机

产生代码的较佳实施例中的最后一个步骤产生代表经替换和处理的规范表达式的 FSA．虽然产生这种 FSA 的算法是公知的，但在过去它们未提供例如把动作语言语句嵌入这些 FSA 的功能．相应地，以下描述集中在本流程较佳实施例中旨在支持动作语言语句的那些新特征以及 reg-exp 语言的其它新特征．

来自本流程的 FSA 输出在初始状态下开始其执行，输入指针设定为输入串的起始，然后依据其当前状态来执行六个基本过程中的一个．这些过程设定了新的当前状态，且一般实行由 FSA 所访问的一个或多个数据结构．这六种类型的状态及其过程如下：

(1) Char_branch<next_state_0, ..., next_state_255>：当前状态为 char_branch 状态时，依据当前输入字符的 ASCII 值 i 来选择下一状态，如 next_state_i，并且输入指针进到输入串中的下一字符．

(2) Marker<num, open/close_flag, offset, next_state>：当前状态为标志器状态时，把一记录压入公知为标志堆栈的运行时间堆栈，该堆栈包含输入串中碰到的括号的字符位置加上指定的偏移量，以及该括号是开还是闭的指示，把下一状态

设定为 next_state ．示例的标志可压入表示在当前输入位置处出现的第五个开括号的记录．

(3) Action<compiled_action_statements, next_state>: 当前状态为动作状态时，执行由 compiled_action_statements 所代表的那些动作语言语句，把下一状态设定为 next_state ．

(4) Success<>: 当前状态为成功状态时，则已成功地匹配了一个串，且此 FSA 终结．

(5) Push<state, next_state>: 当前状态为压入状态时，则把 FSA 的当前配置压入公知为回溯堆栈的运行时间堆栈，把下一状态设定为 next_state ． FSA 配置是包含至少当前状态的标识符以及输入串中当前位置的记录．使用压入来支持规范表达式语句中的非确定性结构．示例的非确定性结构是被 FSA 匹配的逻辑和 "reg-expr1 | reg-expr2"，从而首先把一回溯状态压入回溯堆栈；然后第二步尝试运行相应于 reg-expr1 的 FSA ；如果失败，则回溯，并且(c)尝试运行相应于 reg-expr2 的有限状态机．

(6) Failure<>: 当前状态为失败状态时，把回溯记录弹出回溯堆栈，并把下一状态和当前输入指针设定到回溯记录的内容中．如果回溯堆栈为空，则 FSA 不能成功地匹配输入串且终结．

使用对先前产生的分析树的后续遍历而对输入规范表达式产生 FSA ．在对分析树进行遍历时，把一有限状态机附加到每个节点 v ，该状态机与以 v 为根的子树所代表的子表达式匹配．此附加的有限状态机是变量 v.machine 的值．最终的 FSA 输出是在产生流程结束时附加到分析树的根部的 FSA ．

在遍历期间，依据本领域内的技术人员已公知的方法来产生某些有限状态机．这些状态机识别标准规范表达式的结构并可通过公知的方法来构成．产生这种标准有限状态机的一般考虑参考包括如下: Aho 等人， 1974, The Design And Analysis Of Computer Algorithms, Addison Wesley Publishing Co., chapter 9 ; Aho 等人， 1986, Compilers Principles, Techniques, and Tools, Addison Wesley Publishing Co., chapter 3 ; Hopcroft 等人， 1979, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley Publishing Co., Section 2.5 ; Sedgewick, 1990, Algorithms in C, Addison-Wesley Publishing Co., chapter 16 ．

此外，所揭示的流程还采用某些构成方法，它们可以在其它实施例改变．例

如，可依据 Boyer-Moore 或其它串匹配算法来构成代表文字串的状态机．例如，见 Sedgewick 的文章的第 19 章。

较佳的流程为:

```
PERFORM a POSTORDER traversal of the tree.
  At each node v, DO:
  IF (v=OpenParenthesis<n>) THEN
     SET v.machine=Marker<n, "open", 0, Success<>>
  ELSE IF (v=CloseParenthesis<n>) THEN
     SET v.machine=Marker<n, "close", 0, Success<>>
  ELSE IF (v=Action<node_for_the_action_language_statements>) THEN
     SET v.machine=Action<compiled_action_statements, Success<>>
  ELSE IF (v=String<literal_string>) THEN
     SET len=length (the-string)
     FOR i=1 to len DO
        branch_state[i]=Char_branch <s_0, s_1, ..., s_255>, where all the s_j
           are Failure<> EXCEPT s_j where j is ASCII code for the i'th
           character of the-string which is Success<>
        SET v.machine=MAKE_SEQUENCE
           (branch_state[1], ..., branch_state[len]).
  ELSE IF (v=Stuff_And_String<the-string,par_1,...,par_k>) THEN
     /*m_1 可以是依据先前所述参考文献中所揭示的公知方法构成的*/
     SETm_1=a finite state machine that scans the input string for the first
     occurrence of the end of this first occurrence.
     FOR i=1 to k DO
        SET par_i.machine=marker state appropriate to parenthesis par_iand
           having an offset of -length(the-string).
     SET v.machine= MAKE_SEQUENCE (m_1, par_1.machine,...,
        par_k.machine).
  ELSE IF (v=Sequence<element_1, ..., element_k>) THEN
     FOR i=1 to k DO
```

SET element_i.machine=machine constructed according to this process for element_i

SET v.machine=MAKE_SEQUENCE

(element_1.machine, ..., element_k.machine)

/*逻辑和，"|"*/

ELSE IF (v=Alternatives<element_1, ..., element_k>) THEN

FOR i=1 to k DO

SET element_i.machine=machine constructed according to this precess for element_i

SET v.machine=element_k.machine

FOR i=k-1 to 1 DO

v.machine=Push<v.machine, element_i.machine>

/*类型为"?"、"+"或"*"*/

ELSE IF (v=Repetition<type, element>)) THEN

SET element.machine=machine constructed according to this process for element

SET v.machine=a finite state machine constructed according to methods disclosed in the previously cited references for this type of repetition element.machine

此流程使用从一系列子机中建立复合机的 MAKE_SEQUENCE 函数. 复合机依次执行每个子机，直到一个子机失败(在此情况下复合机也失败)，或者所有的子机都成功(在从情况下复合机成功). 换句话说，复合机运行第一子机；如果成功，则它运行第二子机；如果成功，则它运行第三子机等等. 如果任何子机失败，即达到 Failure<>状态，则复合机也失败.

FUNCTION MAKE_SEQUENCE (machine m_1, m_2, ..., m_k)

: RETURNS machine.

new_m=m_1;

FOR i=2 to k DO

Traverse the states of new_m and replace any Success<> states with m_i

RETURN new_m

这样总结了表示来自本发明的 reg-exp 语言的规范表达式的 FSA 的较佳结构．接着，描述 FSA 的执行．

规范表达式的运行时间解释

在较佳实施例中，通过解释在先前的代码产生步骤中所产生的结构来执行代表规范表达式的 FSA ．解释程序最好访问包括 FSA 的当前状态、对输入串的当前字符位置的指针、回溯堆栈和标志器堆栈的几个数据结构．回溯堆栈包含以解释 FSA 的状态为特征的记录，当尝试对输入串所进行的局部匹配失败时，则可以用本领域内公知的实行回溯的方式来使用该堆栈．配置记录包括当前状态和当前输入位置．在另一个实施例，通过实行具有潜在的下一个状态的清单的状态，可避免压入状态．

本发明的 reg-exp 语言的新变量联编和动作语句特征的较佳实施例利用附加堆栈即标志堆栈．这些特征的语义需要直到整个规范表达式匹配流程已成功才执行标志动作．在成功时，使用在分析期间所发生的变量联编．执行所有的动作这意味着，例如$1 等单个变量在匹配成功时执行多次的同一动作语句中具有不同的联编．

按照以下方式在较佳实施例中使用标志堆栈来实现该语义．在当前解释的状态为 Action<>状态时，不立即执行动作语言语句．而是把动作语句代码压到标志堆栈上．同样，当前状态为标志状态时，(它表示规范表达式定义中的一个括号)，则把出现在标志状态的信息压到标志堆栈上．该信息使能找出输入串中当前变量联编的开始或结束位置．为了防止在失败的局部匹配尝试期间执行所碰到的动作语言语句，在 FSA 回溯时把标志堆栈弹到适当的层次．这样，回溯堆栈上的机器配置和配置记录还包括标志堆栈中的当前位置．

在最后的匹配成功时，通过从头到尾地扫描标志堆栈来执行动作语言语句．在该扫描中，按照标志状态记录的指示来设定变量联编，并按照动作状态记录的指示来执行动作．在匹配失败的情况下，不执行动作语言语句．

更具体地说，较佳实施函数的一个实施例如下:

```
GLOBAL VARIABLE current_state=initialized to the initial machine state
GLOBAL VARIABLE input_pos=initialized to the address of the beginning of
    the input string
GLOBAL VARIABLE BT_STACK=initialized to an empty stack
```

```
GLOBAL VARIABLE MARK_STACK=initialized to an empty stack
REPEAT UNTIL one of the following clauses exits
CASE (current_state) IS
  Char_branch<state_0, ..., state_255>:
      current_state=state_(ASCII value of the current input character)
      input_pos=input_pos+1
  Marker<num,open/close_flag,offset,nextstate>:
      push ("mark",num,open/close_flag,input_pos+offset)
          onto MARK_STACK
      current_state=nextstate
  Action<compiled_action_code,nextstate>:
      push ("action", compiled_action_code) onto MARK_STACK
      current_state=nextstate
  Push<state,nextstate>:
      push (state, input_pos, height(MARK_STACK)) onto MARK_STACK
      current_state=nextstate
  /*在最后成功时退出 REPEAT 语句*/
  Success<>: GOTO done
  Failure<>:
      /*如果尝试回溯且回溯堆栈为空则匹配最终失败，*/
      IF BT_STACK is empty THEN
          RETURN "fail"
      ELSE
          pop BT_STACK record into variables st, ip, msh
          current_state=st
          input_pos=ip
          pop MARK_STACK down to height msh
END CASE
/*在进行分析时，扫描 MARK_STACK 并执行具有由标志所指示的变量联编的动作*/
```

```
done:
LOCAL VARIABLE open_marks[ ], close_marks[ ]
FOR element=bottom of MARK_STACK up to top of MARK_STACK DO :
  IF (element=("mark",num,"open",pos)) THEN
    SET open_marks[num]=pos
  IF (element=("mark",num,"close",pos)) THEN
    SET close_marks[num]=pos
  IF (element=("action",compiled_action_code)) THEN
    SET values of $0, $1, ..., $n to the positions in the imput string indicated
by      open_marks[n] and close_marks[n]
    RUN compiled_action_statement using these bindings of $0, $1, ..., $n
ENDFOR
/*匹配成功并执行所有的动作*/
RETURN "succeed"
```

例如，在前面的 FOR 循环中，把$2 的联编设定为以 open_marks[2]开始并以 close_marks[2]结束的输入串的子串.

动作语言代码产生以及运行时间解释

在较佳实施例中，动作语言语句被编译成可变长度字符编码类型的中间语言，在运行时间由字符编码解释程序解释执行. 通过任何适当的公知分析技术来分析先前所述的动作语言，例如通过 LL 或 LR 分析. 在例如 Aho 等人， 1986, Compilers Principles, Techniques, and Tools, Addison Wesley Publishing Co., 第 4 章中的例子提出了足够的分析技术. 在分析期间，给包装器中出现的所有变量赋值独有的数. 这种较佳的赋值按照碰到的顺序把连续正整数分配给例如$x 、$abc 等指定的变量，并把连续负整数分配给例如$0 、$1 、$2 等代表匹配子串的数值变量，给数值变量$i 分配号码-(i+1).

通过公知的面向语法的翻译技术来产生中间代码. 在 Aho 等人第 8 章的例子中提出足够的面向语法的翻译技术. 以下提出较佳的中间代码. 在此介绍中，中间语言指令具有指令代码的格式，它具有随意修改的信息，接着是零个或单个自变量. 以<var>来代表变量，这里 var 是变量的 2 字节整数编码. 相对转移的偏移量编码为 4 字节整数. 所有的转移对应于当前指令地址即转移指令本身的地址，

从而可对字节代码进行重新定位.

1. Function Calls:

   Encoding: - funcode numargs <var_0> <var_1> ... <var_numargs>

   Meaning: 给 var0 赋予由号码"numargs"的自变量 var_1, ..., var_numargs 上的"funcode"所识别的函数的结果. 该语言的每个内部函数(例如, agrc 、 argv 等)以及每个算符(例如, "+"、"-"、"*"等)具有其自身独有的整数"funcode".

2. Branch Instructions:

   Encoding: 1 <offset> <var0>

   Meaning: 如果 var0 为真, 则转移相对转移指令的偏移量

   Encoding: 2 <offset> <var0>

   Meaning: 如果 var0 为假, 则转移相对转移指令的偏移量

   Encoding: 3 <offset>

   Meaning: 总是转移相对转移指令的偏移量

3. Load Constant Instructions:

   Encoding: 4 <var0> <null-terminated_string>

   Meaning: 给 var0 赋值 null-terminated-string 常数

   Encding: 5 <var0> <floating-point-constant>

   Meaning: 给 var0 赋值 floating-point-constant

4. Move Instructions:

   Encoding: 6 <var0> <var1>

   Meaning: var0 从 var1 拷贝

5. Output Instructions:

   Encoding: 7 <var0>

   Meaning: 把一个项 var0 输出到当前元组

   Encoding: 8

   Meaning: 终结当前元组, 使它返还到网络机器人

6. Parsing Instructions:

   Encoding: 9 <var0> <var1> <address of the finite state machine for <nt>>

   Meaning: 给 var0 赋予依据以<nt>所标记的规范表达式对 var1 所进行的分析结果

7. Exit Instructions:

Encoding: 10

Meaning: 退出当前动作语言块

在另一个实施例中，可使用不同的中间代码，或者甚至可不使用中间代码．例如，替代先前所揭示的基于地址的中间代码，还可等效地使用基于堆栈的逆波兰中间代码．或者，不使用中间代码，可把动作语言语句直接编译成机器语言．使用所揭示的面向语法的方法可实现这两个选择．例如，见 Aho 等人．

在一较佳实施例中以几个步骤来进行包装器动作的执行．首先，分配存储器来存储所有的变量和变量值．在分析了动作语句后知道变量的数目．把所有的变量初始化到未赋的值．接着，把指针 CURRENT_INSTRUCTION 初始化到指向待执行的字节代码的第一指令．然后，解释程序输入一循环，它检索 CURRENT_INSTRUCTION 指向的字节代码，依据先前所述的字节代码语言来完成经编码的简单动作，并把 CURRENT_INSTRUCTION 更新到指向下一指令或对于转移指令，通过偏移值来修改它的值．解释程序在碰到退出指令时结束执行．

通过调用 FSA 解释程序来解释规定了有限状态机的编译的规范表达式代码，在运行时间执行分析内部函数．翻译输出(a_1,...,a_n)语句，然后通过对输出清单中的每个变量执行字节代码指令代码 7 来执行该输出语句，接着是终结当前元组即标志输出语句结束的字节代码指令 8．

6. 特定实施例，参考文献的引用

本发明不限于这里所述的特定实施例的范围．实际上，除了这里所述的实施例以外，从上述描述和附图中，本发明的各种修改对本领域内的技术人员来说将变得明显起来．这些修改将落在后面的权利要求书的范围内．

这里引用了各种出版物，在这里通过引用而引入其整个内容．

说 明 书 附 图

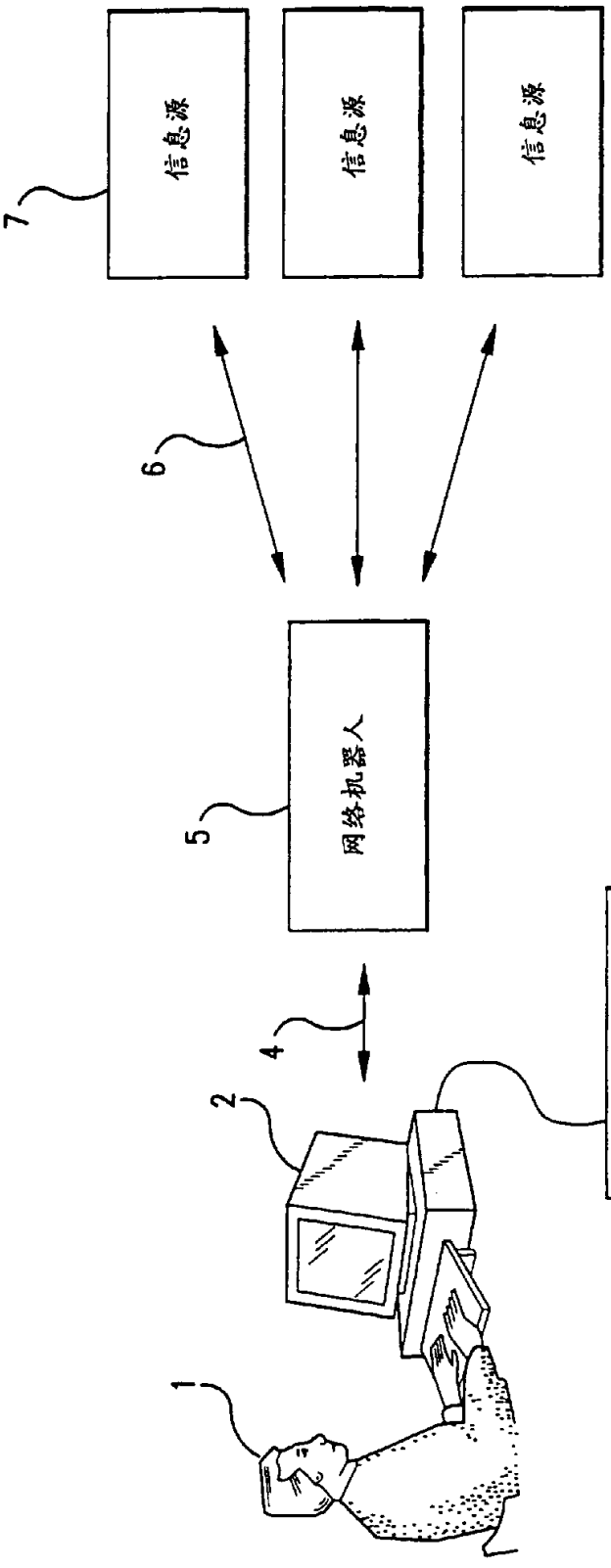


图 1

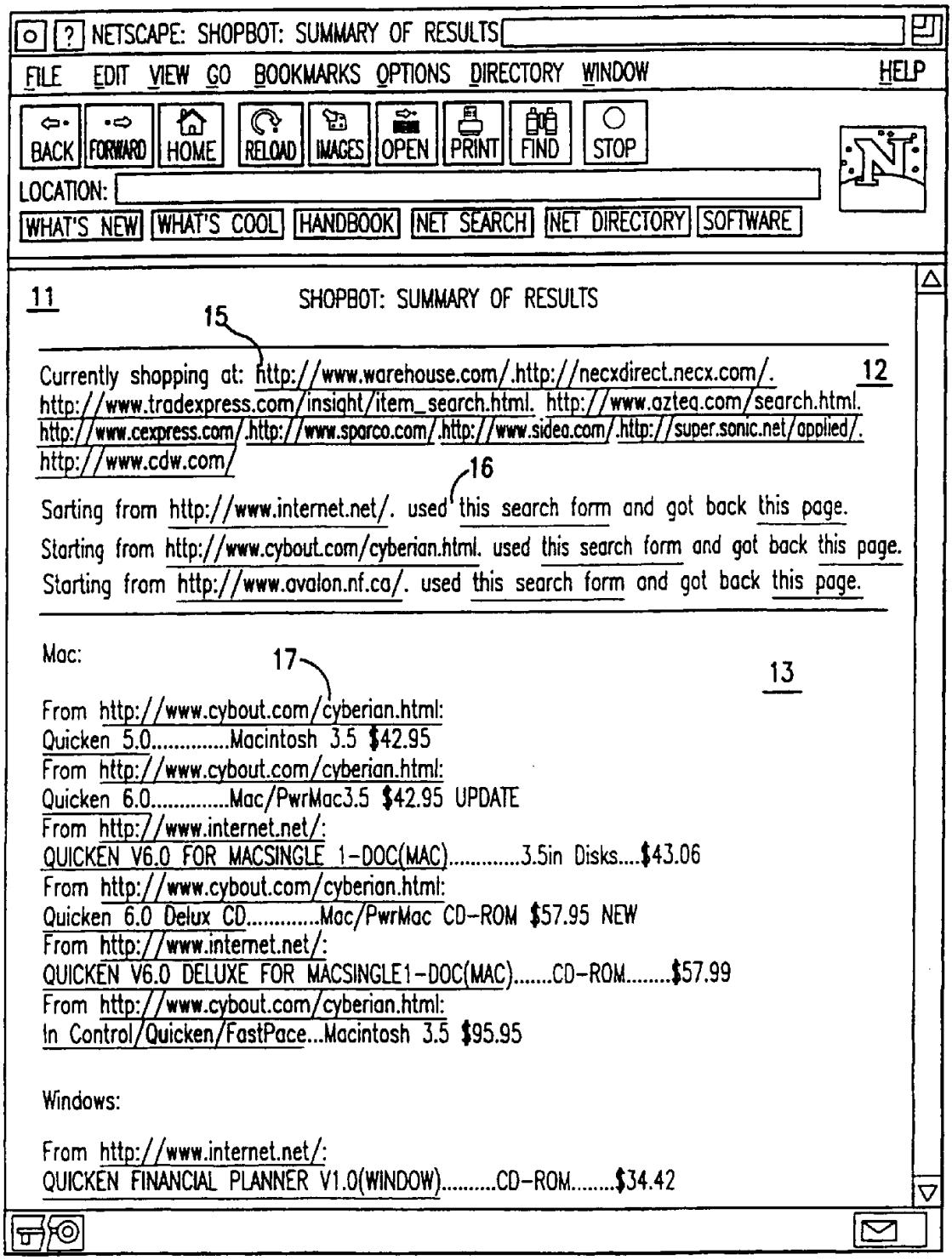

NETSCAPE: SHOPBOT: SUMMARY OF RESULTS
FILE EDIT VIEW GO BOOKMARKS OPTIONS DIRECTORY WINDOW HELP
BACK FORWARD HOME RELOAD IMAGES OPEN PRINT FIND STOP
LOCATION:
WHAT'S NEW WHAT'S COOL HANDBOOK NET SEARCH NET DIRECTORY SOFTWARE
11
SHOPBOT: SUMMARY OF RESULTS
15
Currently shopping at: http://www.warehouse.com/.http://necxdirect.necx.com/. 12
http://www.tradexpress.com/insight/item_search.html. http://www.azteq.com/search.html.
http://www.cexpress.com/.http://www.sparco.com/.http://www.sidea.com/.http://super.sonic.net/applied/.
http://www.cdw.com/
16
Sarting from http://www.internet.net/. used this search form and got back this page.
Starting from http://www.cybout.com/cyberian.html. used this search form and got back this page.
Starting from http://www.avalon.nf.ca/. used this search form and got back this page.
Mac:
17
13
From http://www.cybout.com/cyberian.html:
Quicken 5.0.....Macintosh 3.5 $42.95
From http://www.cybout.com/cyberian.html:
Quicken 6.0.....Mac/PwrMac3.5 $42.95 UPDATE
From http://www.internet.net/:
QUICKEN V6.0 FOR MACSINGLE 1-DOC(MAC).....3.5in Disks....$43.06
From http://www.cybout.com/cyberian.html:
Quicken 6.0 Delux CD.....Mac/PwrMac CD-ROM $57.95 NEW
From http://www.internet.net/:
QUICKEN V6.0 DELUXE FOR MACSINGLE1-DOC(MAC).....CD-ROM.....$57.99
From http://www.cybout.com/cyberian.html:
In Control/Quicken/FastPace...Macintosh 3.5 $95.95
Windows:
From http://www.internet.net/:
QUICKEN FINANCIAL PLANNER V1.0(WINDOW).....CD-ROM.....$34.42

图 2A

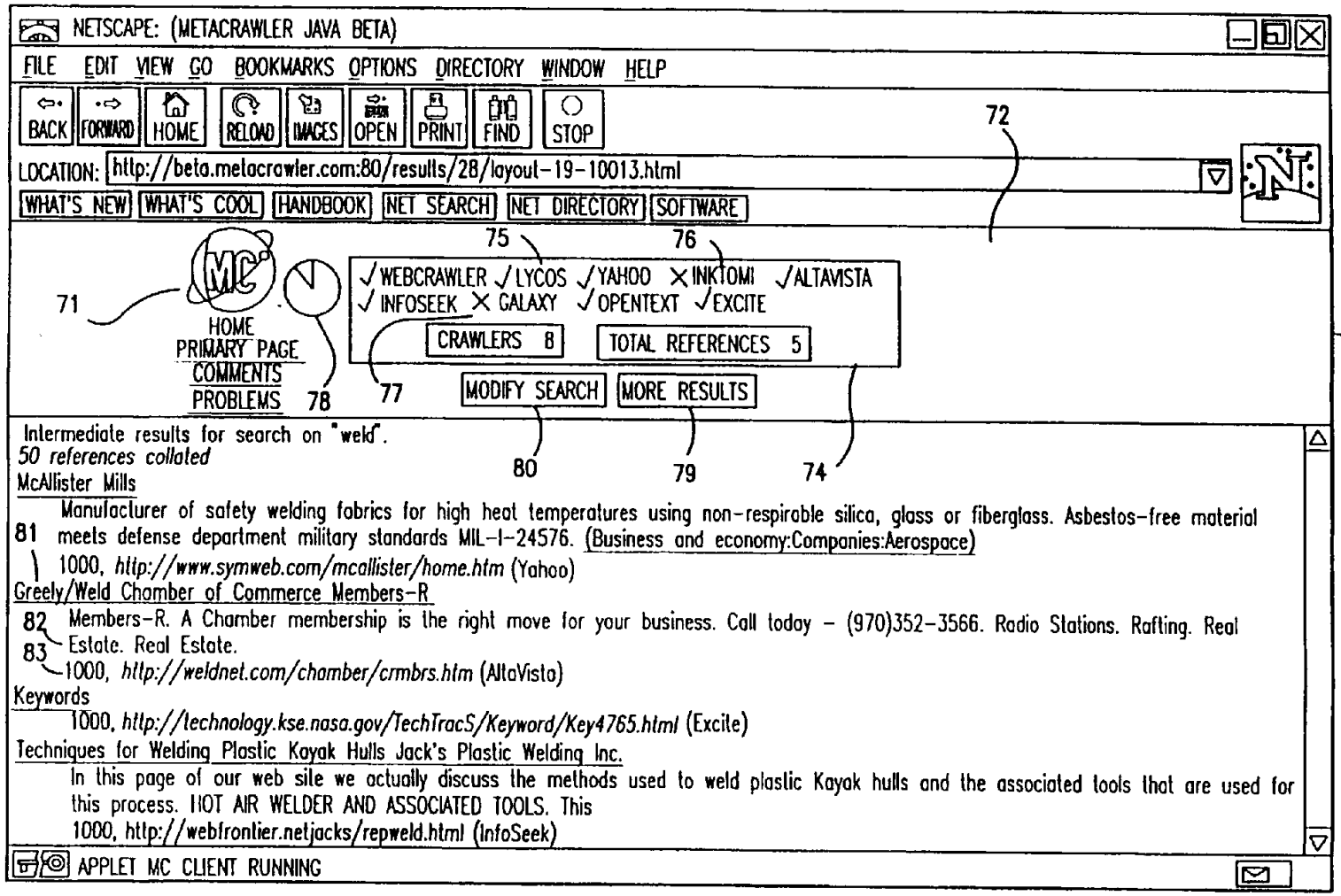

NETSCAPE: (METACRAWLER JAVA BETA)
FILE EDIT VIEW GO BOOKMARKS OPTIONS DIRECTORY WINDOW HELP
BACK FORWARD HOME RELOAD IMAGES OPEN PRINT FIND STOP
LOCATION: http://beta.metacrawler.com:80/results/28/layout-19-10013.html
WHAT'S NEW WHAT'S COOL HANDBOOK NET SEARCH NET DIRECTORY SOFTWARE
72
71
HOME
PRIMARY PAGE
COMMENTS
PROBLEMS
78
75
76
✓WEBCRAWLER ✓LYCOS ✓YAHOO ✕INKTOMI ✓ALTAVISTA
✓INFOSEEK ✕GALAXY ✓OPENTEXT ✓EXCITE
CRAWLERS 8
TOTAL REFERENCES 5
73
77
MODIFY SEARCH
MORE RESULTS
80
79
74
Intermediate results for search on "weld".
50 references collated
McAllister Mills
Manufacturer of safety welding fabrics for high heat temperatures using non-respirable silica, glass or fiberglass. Asbestos-free material meets defense department military standards MIL-I-24576. (Business and economy:Companies:Aerospace)
81
1000, http://www.symweb.com/mcallister/home.htm (Yahoo)
Greely/Weld Chamber of Commerce Members-R
82
Members-R. A Chamber membership is the right move for your business. Call today - (970)352-3566. Radio Stations. Rafting. Real Estate. Real Estate.
83
1000, http://weldnet.com/chamber/crmbrs.htm (AltaVista)
Keywords
1000, http://technology.kse.nasa.gov/TechTracS/Keyword/Key4765.html (Excite)
Techniques for Welding Plastic Kayak Hulls Jack's Plastic Welding Inc.
In this page of our web site we actually discuss the methods used to weld plastic Kayak hulls and the associated tools that are used for this process. HOT AIR WELDER AND ASSOCIATED TOOLS. This
1000, http://webfrontier.netjacks/repweld.html (InfoSeek)
APPLET MC CLIENT RUNNING

图 2B

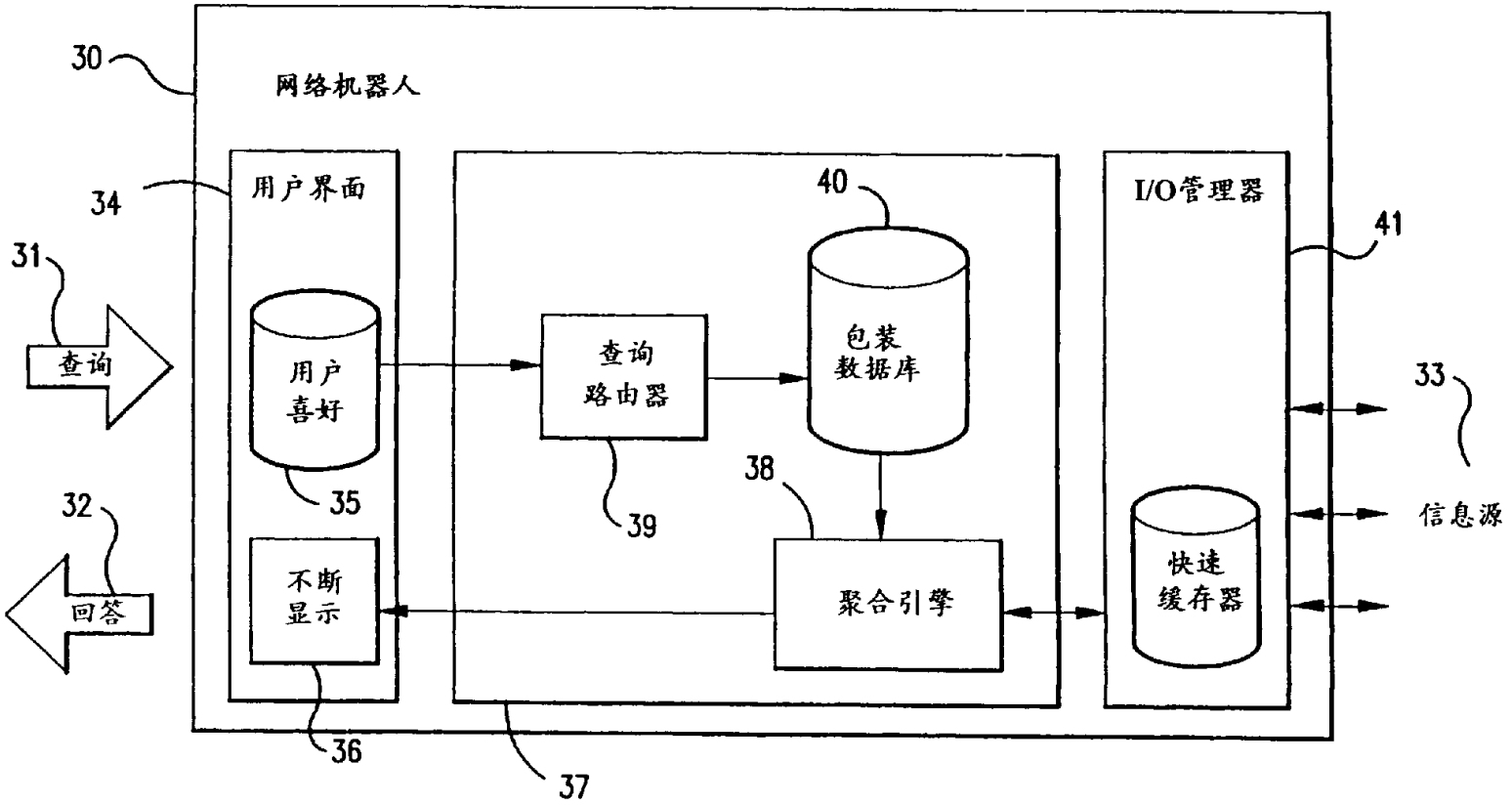

30
网络机器人
34
用户界面
31
查询
用户
喜好
35
32
回答
不断
显示
36
37
查询
路由器
39
40
包装
数据库
38
聚合引擎
I/O管理器
41
快速
缓存器
33
信息源

图 3

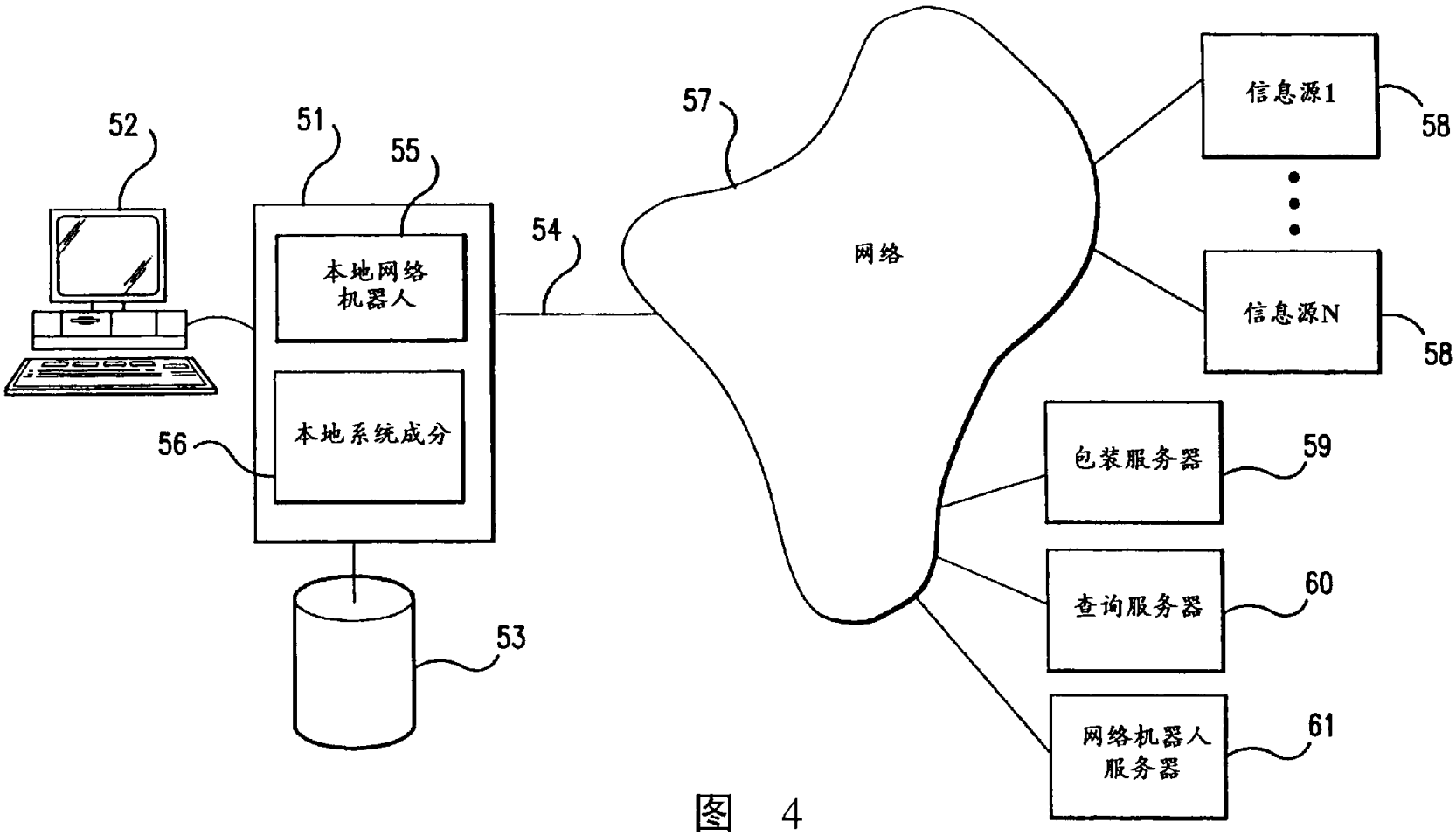

52
51
55
57
信息源1
58
本地网络
机器人
54
网络
信息源N
58
本地系统成分
56
包装服务器
59
查询服务器
60
53
网络机器人
服务器
61

图 4