[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/30 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200480011576.2

[43] 公开日 2006 年 5 月 31 日

[11] 公开号 CN 1781102A

[22] 申请日 2004.4.22
[21] 申请号 200480011576.2
[30] 优先权
[32] 2003. 4.30 [33] FI [31] 20035055
[86] 国际申请 PCT/FI2004/050044 2004.4.22
[87] 国际公布 WO2004/097673 英 2004.11.11
[85] 进入国家阶段日期 2005.10.28
[71] 申请人 诺基亚有限公司
地址 芬兰埃斯波
[72] 发明人 J·索恩陶斯塔 田继雷
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 刘 红 梁 永

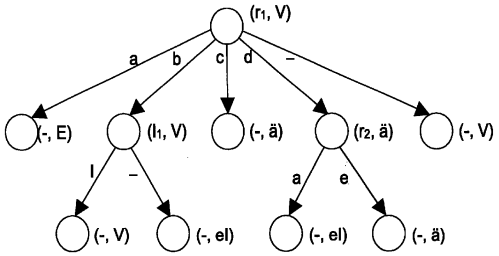
权利要求书 4 页 说明书 16 页 附图 5 页

[54] 发明名称

低速存储器判定树

[57] 摘要

本发明涉及低速存储器树形数据结构的管理。根据本发明的方法包括用于创建由父节点和至少一个叶节点组成的判定树的步骤，和用于从所述节点搜索数据的步骤。 以节点按照存储顺序跟随父节点的这样一种方式顺序地存储判定树的节点，其中没有来自其父节点的链路，也能够到达精选可搜索数据的上下文的节点。 最好能够在语音识别系统中在文本－音素映射中利用该方法。



1、用于管理树形数据结构的一种方法，该方法包括创建包含父节点和至少一个叶节点的判定树的步骤，所述方法还包括用于从所述节点中搜索数据的步骤，其特征在于：通过以节点按照存储顺序跟随父节点的这样一种方式顺序地存储节点来创建判定树，其中能够到达精选可搜索数据的上下文的节点而没有来自其父节点的链路。

2、根据权利要求 1 所述的方法，其特征在于，通过以下方案之一存储判定树的节点：深度优先方案，宽度优先方案，或者所述方案的组合。

3、根据权利要求 1 或 2 所述的方法，其特征在于，在存储之前，利用对准的数据元素集训练判定树，根据所述元素的有效性利用惩罚初始化所述元素，其中通过对于无效元素增加约束，根据所述元素的已知特征，进一步重新初始化该惩罚。

4、根据权利要求 1 或 2 或 3 所述的方法，其特征在于，该方法用于文本-音素映射，其中所述数据包括诸如字母和音素的元素，其中为一个字母创建判定树，并且节点包括所述字母的音素。

5、根据权利要求 4 所述的方法，其特征在于，节点还包括有关诸如周围字母的所述字母的上下文的信息。

6、根据权利要求 3-5 之一所述的方法，其特征在于，检查音素的有效性，检查它是否是允许的音素之一，其中利用零初始化所述惩罚；或者检查它是否不是允许的音素，其中利用大的正值初始化所述惩罚。

7、根据权利要求 3-6 之一所述的方法，其特征在于，进一步检查音素的有效性，检查它是否是语言上允许的，其中利用零重新初始化该惩罚；或者检查该音素是否不是语言上允许的音素，其中利用大的正值重新初始化该惩罚。

8、根据权利要求 1-7 之一所述的方法，其特征在于，以以下格式之一编码节点的数据：具有固定比特的格式，具有可变比特的格式，或者具有霍夫曼码的格式。

9、根据权利要求 8 所述的方法，其特征在于，如果由于可变比特格式带来的节省大于存储由这些元素组成的表的开销，则以可变比特格式提供这些元素；或者如果由于霍夫曼编码带来的节省大于由于可

变比特格式带来的节省并大于由于霍夫曼码引入的开销，则所述元素被霍夫曼编码。

10、根据权利要求 9 所述的方法，其中通过把利用固定比特格式分配给元素的比特数量（Num_bits_fixed）和利用可变比特格式分配给元素的比特数量（Num_bits_varible）之和与元素出现在判定树中的总次数（Count_occurrence）相乘，确定由于可变比特格式带来的节省。

11、根据权利要求 9 所述的方法，其中通过把利用固定比特格式分配给元素的比特数量（Num_bits_fixed）和分配给码字的比特数量（Num_bits_CW）之和与码字出现在霍夫曼编码树中的次数（Count_CW）相乘，计算由于霍夫曼编码带来的节省。

12、根据权利要求 8-11 所述的方法，其特征在于，实质上对于每个元素单独地选择存储格式。

13、用于管理树形数据结构的一种系统，该系统包括判定树创建器，所述判定树由父节点和至少一个叶节点组成，并且该系统还包括从所述节点搜索数据的数据搜索器，其特征在于：该创建器适用于通过以节点按照存储顺序跟随父节点的这样一种方式顺序地存储节点来创建判定树，其中精选可搜索数据的上下文的节点是可达的而没有来自其父节点的链路。

14、根据权利要求 13 所述的系统，其特征在于，存储顺序是以下方案之一：深度优先方案，宽度优先方案，或者所述方案的组合。

15、根据权利要求 13 或 14 所述的系统，其特征在于，该系统还包括用于利用对准数据元素集训练判定树的装置，所述装置能够根据所述元素的有效性利用惩罚来初始化所述元素，并且通过对于无效元素增加约束，根据所述元素的已知特征进一步重新初始化该惩罚。

16、根据权利要求 13 或 14 或 15 所述的系统，其特征在于，该系统应用文本-音素映射，其中所述数据包括诸如字母和音素的元素，其中判定树对应于一个字母，以及所述节点包括所述字母的音素。

17、根据权利要求 16 所述的系统，其特征在于，节点还包括有关诸如周围字母的所述字母的上下文的信息。

18、根据权利要求 13-17 之一所述的系统，其特征在于，该系统适用于检查音素的有效性，检查它是否是允许的音素之一，其中该系

统适用于利用零初始化所述惩罚；或者如果该音素不是允许的音素，则该系统适用于利用大的正值初始化所述惩罚。

19、根据权利要求 13-18 之一所述的系统，其特征在于，该系统还适于检查音素的有效性，检查它是否是语言上允许的，其中该系统适于用零初始化所述惩罚；或者如果该音素不是语言上允许的音素，则该系统适用于利用大的正值初始化所述惩罚。

20、根据权利要求 13-19 之一所述的系统，其特征在于，以以下方式之一表示节点的数据：可变或固定或霍夫曼编码的二进制数字。

21、根据权利要求 20 所述的系统，其特征在于，该系统适用于通过一起比较所述格式来检查所述格式中的哪个格式比其它的格式产生更大的节省，其中该系统适于使用那个二进制格式。

22、根据权利要求 13-20 之一所述的系统，其特征在于，该系统是语音识别系统。

23、一种设备，包括用于在树形数据结构中存储数据的存储媒体，所述树形数据结构由父节点和至少一个叶节点组成，所述设备还包括用于处理所述结构中数据的处理器，该处理器包括判定树创建器和用于来自所述节点的数据的搜索器，其特征在于：该创建器适用于通过以节点按照存储顺序跟随父节点的这样一种方式顺序地存储节点来创建判定树，其中精选可搜索数据的上下文的节点是可达的而没有来自其父节点的链路。

24、根据权利要求 23 所述的设备，其特征在于，存储顺序是以下方案之一：深度优先方案，宽度优先方案，或者所述方案的组合。

25、根据权利要求 23 或 24 所述的设备，其特征在于，该设备应用文本-音素映射，其中所述数据包括诸如字母和音素的元素，其中判定树用于一个字母，以及节点包括所述字母的音素。

26、根据权利要求 23-25 之一所述的设备，其特征在于，该设备还包括语言识别系统。

27、根据权利要求 23-26 之一所述的设备，其特征在于，该设备还包括用于电信的装置。

28、一种树形数据结构，由父节点和至少一个叶节点组成，其中所述节点包括可搜索数据，其特征在于：以节点按照存储顺序跟随父节点的这样一种方式顺序地定位这些节点，其中精选可搜索数据的上

下文的节点是可达的而没有来自其父节点的链路。

29、根据权利要求 28 所述的树形数据结构，其特征在于，以以下方式之一表示节点的数据：可变或固定或霍夫曼编码的二进制数字。

30、根据权利要求 28 或 29 所述的树形数据结构，其特征在于，所述结构由文本-音素映射利用，其中判定树包括一个字母的信息，其中节点包括其音素的信息。

31、根据权利要求 28-30 之一所述的树形数据结构，其特征在于，节点包括有关该字母的周围字母的信息。

32、一种计算机程序产品，包括计算机存储媒体和写在计算机存储媒体上用于利用存储在所述存储媒体中的树形数据结构的计算机可读代码，所述树形数据结构包括父节点和至少一个叶节点，其中所述计算机可读代码包括用于从所述节点搜索内容的指令，其特征在于：该计算机可读代码具有用于以节点按照存储顺序跟随父节点的这样一种方式顺序地安排节点的指令，其中精选可搜索数据的上下文的节点是可达的而没有来自其父节点的链路。

低速存储器判定树

技术领域

本发明涉及根据所附的独立权利要求 1 的前序部分的用于管理树形数据结构的方法，本发明还涉及根据所附的独立权利要求 13 的前序部分的用于实施上述方法的系统。此外，本发明涉及根据所附的独立权利要求 23 的前序部分的设备。本发明还涉及根据所附的独立权利要求 28 的前序部分的树形数据结构，以及涉及根据所附的独立权利要求 32 的前序部分的用于利用上述树形数据结构的计算机程序产品。

背景技术

多语言方面在自动语音识别系统中变得越来越重要。语音识别系统的类型包括语音识别引擎，该语音识别引擎例如可以包括用于自动语言识别、在线发音模型建立（文本-音素）和多语言声音模型建立的单元。语音识别引擎的操作工作在以文本形式给定词汇项的假设上。首先，语言识别模块根据词汇项的书写表示来识别语言。一旦这已被确定，则应用合适的在线文本-音素模型建立方案来获得与该词汇项相关联的音素序列。音素是将一个单词的发音与另一个单词的发音区分开来的最小项。任何语言中的任何词汇项可以被呈现为一组对应于人类语音产生系统中变化的音素。

多语言声音模型被级联，以便为每个词汇项构建识别模型。利用这些基本模型，识别器能够原则上自动地处理多语言词汇项，而不需要用户的任何协助。文本-音素对于在自动语音识别以及文本-语音两者中为词汇项提供精确音素序列具有关键作用。神经网络或判定树方案经常被用作文本-音素映射。在用于语言和说话者无关的语音识别的解决方案中，基于判定树的方案已提供最精确的因素序列。用于安排树结构的方法的一个实例呈现在 US6411957B1 中。

在该判定树方案中，语言的字母表中的每个字母的发音被分别建模，并且为每个字母训练单独的判定树。当找到单词的发音时，一次一个字母地处理该单词，并根据当前字母的判定树文本-音素模型找到当前字母的发音。

判定树的一个实例被显示在图 1 中。该判定树由可以是内部节点 I

或叶 L 的多个节点组成。分支是诸多节点的集合，这些节点从根 R 一起被链接到叶 L。节点可以是父（parent）节点或者是子（child）节点。父节点是能够从中进一步遍历树的节点，换句话说，它具有子节点。树中的子节点是可以从父节点到达的节点。内部节点 I 可以是父节点和子节点，但叶只是子节点。判定树中的每个节点都存储信息。存储的信息根据判定树的上下文而变化。

在语音识别系统中，内部节点 I 通常具有有关被识别的单词和该单词的发音的信息。单词的字母的发音可以由某些上下文中的音素（P_i）来指定。上下文例如指单词中在感兴趣的字母的右边和左边的字母。当在判定树中爬升（climb）时，上下文信息的类型由考虑其上下文的属性（a_i）（也被称作属性类型）指定。可以借助属性值来实现爬升，在假定给定字母的上下文信息时，该属性值定义分支，其中搜索算法应当进行到该分支中。

自根节点 R 开始爬升树结构。在每个节点上，应当检查属性类型（a_i），并且应将对应信息用于确定当前字母的上下文。利用该信息，匹配上下文信息的分支可以向前移动到树中的下一个节点。爬树，直至找到叶节点 L，或者在树中对于当前上下文没有匹配属性值。

在图 2 中示出了基于判定树文本-音素映射的一个简化实例。该图中的判定树用于字母‘a’，其中节点代表字母‘a’的音素。应当注意，该图示被简化并且不包含字母‘a’的所有音素。在根节点中，存在关于属性类型的信息，这是右边的第一字母并利用 r_1 表示。对于两个其它的内部节点，属性类型是左边的由 I_1 表示的第一字母和右边的由 r_2 表示的第二字母。对于叶节点，没有属性类型被分配。

当搜索单词‘Ada’的发音时，可以利用该实例中呈现的判定树和用于字母‘d’的判定树生成用于该单词的音素序列。在该实例中，用于字母‘d’的树仅仅由根节点组成，并且分配给根节点的音素是音素 /d/。

当生成音素序列时，从左到右每次一个字母处理该单词。第一字母是‘a’，因此首先考虑用于字母‘a’的判定树（参见图 2）。属性 r_1 被附加到根节点上。‘a’后面的下一个字母是‘d’，因此我们前进到对应于属性值‘d’的根节点之后的分支。该节点是被附加了属性 r_2 的内部节点。右边的第二字母是‘a’，并且我们前进到相应分支，

而且进一步前进到是叶的相应节点上。对应于叶的音素是/e1/。因此，序列中的第一音素是/e1/。

该实例单词中的下一个字母是‘d’。用于字母‘d’的判定树如上所述由根节点组成，其中最频繁的音素是/d/。因此，该序列中的第二音素是/d/。

该单词中的最后的字母是‘a’，并且再次考虑字母‘a’的判定树（参见图 2）。附加到根节点的属性是 r_1。对于该单词的最后字母，字母‘a’右边的下一个字母是语义图ε’_’。沿着相应的分支爬树直至是叶的节点。附加到叶节点上的音素是/V/，这是序列中的最后音素。

最后，用于单词‘Ada’的完整音素序列是/e1/ /d/ /V/。在对于字母表中的所有字母训练判定树之后，可以以类似方式生成用于任何单词的音素序列。

判定树训练是在包含单词及其发音的发音字典上完成的。判定树的强度在于利用信息理论原理从训练词典中学习紧映射（compact mapping）的能力。

如所述的，基于判定树的实施已经提供最精确的音素序列，但是缺点在于当使用判定树方案作为文本-音素映射时的大存储器消耗。大存储器消耗是由于链接列表判定树方案中使用的众多指针造成的。存储器的量尤其随着诸如英语或类似的其中发音不规则性频繁发生的语言而增加。

对于所述问题的现有技术解决方案可以被分类为有损耗的和无损耗的方法。当试图降低判定树的存储器需求时，大部分使用有损耗方法。这些方案例如是组合判定树的属性值，最佳化判定树训练处理的停止准则，根据错误计数修剪判定树，和其它类似方法。

对于现有技术的低速存储器（low memory）判定树方法，当为了存储器而最佳化系统时，总是降低性能。总是存在精度与存储器消耗之间的折衷。与此相反，由于根据本发明的方案，几乎没有精度的任何恶化并且使存储器消耗最佳化。可以显著降低存储器需求而没有性能的恶化。

发明内容

为了实现这一目的，用于管理树形数据结构的方法包括用于创建

由父节点和至少一个叶节点组成的判定树的步骤，并且也包括用于从所述节点中搜索数据的步骤。所述方法的特征在于判定树，其中通过以节点按照存储顺序跟随父节点的这样一种方式顺序地存储节点来创建该判定树，其中可以到达精选（refine）可搜索数据的上下文的节点而没有来自其父节点的链路。

对于用于管理树形数据结构的系统，其特征在于创建器，其适用于通过以节点按照存储顺序跟随父节点的这样一种方式顺序地存储节点来创建判定树，其中精选可搜索数据的上下文的节点是可达的而没有来自其父节点的链路。

根据本发明的设备包括用于存储树形数据结构中的数据的存储媒体和用于处理所述结构中数据的处理器，所述处理器包括判定树创建器和用于从所述节点中搜索数据的搜索器。该设备的特征在于：创建器适用于通过以诸多节点按照存储顺序跟随父节点的这样一种方式顺序存储节点来创建判定树，其中没有来自其父节点的链路，精选可搜索数据的上下文的节点也是可达的。

所述树形数据结构由父节点和至少一个叶节点组成，所述节点包括可搜索数据，所述树形数据结构的特征在于节点，以这些节点按照存储顺序跟随父节点的这样一种方式来顺序地定位这些节点，其中没有来自父节点的链路，也可以到达精选可搜索数据的上下文的节点。

根据本发明的计算机程序产品包括计算机存储媒体和写在计算机存储媒体上的计算机可读代码，用于利用所述存储媒体中存储的树形数据结构，所述树形数据结构包括父节点和至少一个叶节点。计算机可读代码包括用于从节点搜索数据的指令。计算机程序产品的特征在于：计算机可读代码具有用于以节点按照存储顺序跟随父节点的这样一种方式顺序地安排节点的指令，其中没有来自其父节点的链路，精选可搜索数据的上下文的节点是可达的。

本发明的第一部分描述了在训练判定树时使用的剪切对准（clipped alignment）方法。该方法能够根据语言知识制作高质量对准词典。如现有技术方法中那样，没有剪切方法，就不会充分使用语言知识。此外，由于本发明，可以容易地找出错误项（音素-字母对）。因此，不规则性被减少，并且改善对已对准字典训练的判定树的存储和精度。同样，如果外来词和姓名服从不同于英语的发音规则，本发

明提供了可能性来除去这些外来词和姓名的项目。显然，不规则性也被降低。剪切对准方法在某种程度上还可以检测错误抄录，并且进一步丢弃它们。由于包含不可能映射对的项目被剪切掉，因此可以正确利用具有小概率的可能的映射对。

附图说明

在附图、随后的详细描述以及所附的权利要求中提出了本发明的优选实施例。在描述中还考虑了本发明的其它目的和优点。在权利要求中利用特殊性定义了发明本身。

图 1 显示具有节点和叶的示范性判定树，具有属性和音素；

图 2 显示在文本-音素映射中使用的字母 'a' 的示范性判定树；

图 3 显示语义图隐藏式马尔可夫（Markov）模型的一个实例；

图 4 显示根据本发明方法的原理图；

图 5 显示链接列表方案的一个实例，其中节点包括指针表；

图 6a-6d 显示用于存储判定树的四种方法的实例；和

图 7 显示利用根据本发明方法的设备的非常原理的实例。

具体实施方式

根据本发明的方法对与约束维特比（Viterbi）算法相结合的判定树应用无损耗编码。本发明对于其中一个字母可以对应于零个、一个或两个音素的语言诸如英语是有利的。

图 4 中展现了基于判定树的所建议的发音模型建立方案的高级描述。该方案基于利用剪切对准算法对准的发音字典。判定树的训练基于对准的发音字典，并且训练的结果是基于判定树的发音模型。在训练之后，把树转换成低速存储器格式，以最小化判定树的存储器需求。低速存储器判定树表示包括：把判定树的节点转换成合适的格式，并利用产生最低存储器消耗的压缩方案压缩判定树的变量。本发明提供用于执行剪切训练对准和用于把判定树转换成低速存储器格式的方法。发音字典包含单词及其发音。判定树通过利用信息理论原理能够从训练字典中学习紧映射。

为了阐明本发明，将本发明分成以下部分：

1. 剪切对准方法

2. 低速存储器判定树方法

3. 用于判定树数据元素的比特分配。

根据本发明的剪切对准方法的第一部分被设计用于在部分 2 中进一步描述的低速存储器判定树的训练。在第三部分中提供了用于压缩判定树的数据元素以实现最低存储器需求的方法。但是，首先，在训练判定树之前，对准项目或如说明书中所涉及的发音字典中的表项目，以找到字母与音素之间的对应关系。通过把音素空（null）（称之为音素 ε，用“_”标记）插入不发音的那些字母的音素序列和产生两个音素的那些字母的伪音素（pseudophoneme）中，能够获得对准。通过级联公知为对应于单个字母的两个音素(/eI/, /oU/,...)获得伪音素。

HMM-维特比算法适于供对准使用。HMM-维特比算法的使用确保在统计意义上以最佳方式执行对准，并因此最小化字典项目的剩余熵。此外，使用 HMM-维特比算法用于对准的优点是能够在统计意义上达到最佳对准。在表 1 给出了对准的发音字典的一个实例:

单词	对准的音素序列
aaron	_ E r2 s@ n
abraham	eI b r2 s@ h ä m
accola	A: _ k oU l s@
ackerman	ä _ k _ s@r m s@ n
ada	eI d s@
adelaide	ä d s@ l eI _ d _
ahmanson	A: _ m s@ n s s@ n
aikman	eI _ k m s@ n
alabamas	A: l A: b A: m s@ z

表 1: 对准的发音字典的一个实例。

隐藏式马尔可夫模型（HMM）是公知的并且被广泛用于例如已经在语音识别中应用的统计方法中。该模型还能够被称为马尔可夫源或马尔可夫链的概率函数。HMM 的基础假设是: 信号可以被很好地表征为参量随机处理，并且可以以精确、明确定义的方式确定/估算随机处理的参数。可以根据分配给每个状态的可观测事件是离散（比如码字）的还是连续的，将 HMM 分类成离散模型或连续模型。利用任何一种方式，

观测是概率的。该模型具有基础随机处理，该随机处理不是直接可观测的，而是仅仅通过另一组产生观测序列的随机处理才能够被看。HMM由具有状态之间转换的隐藏状态组成。数学表示包括三项：状态之间的状态转换概率，每个状态的观测概率和初始状态分布。给定 HMM 和观测，则维特比算法用来通过跟随最佳路径给出观测状态对准。

为了对准发音字典，如果在用于字母 1 的允许音素的列表中能够找到音素 f，则用零初始化用于给定的字母-音素对的惩罚（penalty）P(f, 1)，否则利用大的正值来初始化。给定初始惩罚值，则分两步对准字典。在第一步中，为字典中的每个项，生成所有可能的对准。随后根据所有对准的项，重新计算惩罚值。在第二步中，仅为每个项查找单个最佳对准。

对于每个项，在语义图 HMM 上可以利用维特比算法找到最佳对准。图 3 中显示了隐藏马尔可夫模型的一个实例。语义图 HMM 具有表目 E，出口 X 和字母 S1、S2、S3 状态。可以映射到伪音素的字母通过具有持续时间状态 D 来处理的。图 2 中的状态 S1-S3 是对应于单词中发音字母的状态。状态 S2 对应于可以产生伪音素的字母，并且这就是状态 S2 为什么具有持续时间状态 D 的原因。允许从所有以前状态跳跃到当前状态，以支持音素 ε。每个状态和持续时间状态持有权标和对应于累积计分的状态序列，其中权标包含相对语义图 HMM 使音素序列对准的累积惩罚。通过从头到尾每次一个音素地通过（go through）音素序列，使音素序列相对字母对准。权标传递被执行，以查找字母与音素之间的维特比对准。最后，在 HMM 所有的状态上找到具有最低累积惩罚的权标。根据权标的状态序列，能够确定单词的字母和音素之间的对准。

对准的字典可以包含如下列出的项目：

a)外来姓名和外来词，如“Juan, Xiong 等”被包含在英语发音字典中。最好在文本-音素映射中与除英语之外的相应语言一起使用那些姓名和单词。那些词使发音更不规则，并且不规则发音使判定树更大和更不精确。

b)错误抄录。由于打字错误和某些不可预见的原因在字典中具有某些错误抄录是不可避免的。这也使发音更加无规则和不精确。

c)错误对准，例如“‘‘apple - ä p V l _ ”。利用基本语言知

识，知道字母“p”从不映射到元音音素“V”。此外，这使得发音更不规则和不精确。

为了解决上述问题，建议根据本发明利用维特比算法的剪切对准方法。借此，高质量对准发音将更加规则，从而导致根据本发明的基于判定树文本-音素模型的低存储要求。

1.用于训练低速存储器判定树的剪切对准方法

根据本发明，基于上述的从对准的字典中估算的重新估算的惩罚P（f,1）完成对准。显然，所形成的对准产生非常粗的对准，所以惩罚P(f, 1)决不是非常精确的。在语言上不可能的情况下，还将值分配给P(f, 1)。例如，P(“V”, “p”)具有一个值，但这明显违反语言知识。为了避免这一情况并且为了克服上述困难（a-c），把定义为剪切方法的约束应用于维特比解码。

所建议的剪切对准算法需要为目标语言定义字母和语音集。表2中的列表规定了字母可以在语言上对应的音素和伪音素（基于人类专家的语言知识）。下表2包括真实的语言相关信息。

字母	对应(伪)音素
a	V, A, ä, eI, e, 0, _
b	b, _
c	k, s, tS, S, _
…	…
1	1, V, V_1, _
…	…
z	z, s, tS, t, t_s, dZ, S, _

表2：字母表的音素和伪音素定义的一个实例。

表2可以利用不同的方式来实施，但是这些实施用于相同目的。

根据所有对准项，重新计算惩罚值。这样，对于每一项，仅找到单个最佳对准。如果在表2中能够找到音素f，则如常估算P(f, 1)，这意味着用于字母1的音素f在语言上是允许的。如果不能在表2中找到用于给定字母1的音素f，则应用约束将P(f, 1)设置为最高值而

不进行任何估算。

现在，在对准字典中仅允许在上表中找到的字母-音素对用于训练基于判定树的文本-音素映射。

由于该方法，可以相当容易地考虑语言信息。因为对准中的剪切方法，将某些项剪切掉。通过检验剪切的项列表，容易发现或者调谐语言信息，例如定义新的伪音素，把丢失音素添加到字母相关音素集，等等。如果涉及更好的语言信息，可以改善对准并可以降低存储器使用。

2.判定树模型结构

通过最小化所有字母的判定树文本-音素模型的存储器需求，最小化用于给定语言的文本-语音模型的尺寸。因此，考虑单一判定树文本-音素模型的存储器需求的最小化。

最好不使用图 1 和图 2 中展示的判定树模型的直接链接列表实施。这是因为以下事实：在链接列表方案中，树的每个节点将包含指针表作为开销。为了除去该开销，最好以允许将树的节点用于文本-音素转换的这样的顺序把树的节点存储在存储器中。

该顺序必须是：当正在为字母精选正确的上下文时，下一匹配上下文也是在紧接后一级上的上下文。换言之，尽管根据现有技术的链接列表树可以按任何顺序存储到存储器中，但是根据本发明的树却不能。现有技术链接列表树的结构自动注意校正参考：当搜索下一级节点时，算法通过使用节点中存储的链路（指针）找出信息。这些链路使用存储器，并且其目的仅仅是启动树的遍历。

链接列表方案的一个实例可以在图 5 中看到，其中节点（10）包括指针表，从此到达对应于字母的子节点（11，12，13，14，15，16）。除非子节点是叶节点（诸如 12，15），否则该子节点仍然包括指针表。

本发明基于这样一种认识：通过在存储器中以适当顺序存储树节点，可以从节点中省去链路或指针，从而节省存储器。这样的结构是例如图 6a-6d 所示的深度优先（depth-first）和宽度优先（breadth-first）存储方案或其某些组合。换言之，本发明在于以适合于文本-音素转换的特定顺序存储树内容，以便即使在树中没有链路，也可以适当搜索树。

在深度优先存储方案中，通过首先一直跟随树结构到最后的最左边叶来存储树的节点。然后，一直遍历右边的下一个分支直至最后的叶。图 6a 显示了图 1 的判定树结构的一个实例，其中节点和叶被转换成低速存储器深度优先格式。在深度优先存储格式中，最好仅存储每个节点一次。例如，仅存储根节点一次。显然，也可以两次或多次存储每个节点，如根节点（如图 6a 所示）。

在宽度优先存储方案中，首先存储树的根 R，然后到达第一层上的所有节点，再到达第二层上的所有节点，等等。图 6b 显示了图 1 的判定树结构的一个实例，其中节点和叶被转换成低速存储器宽度优先格式。

在混合存储方案中，可以混合深度优先和宽度优先方案。图 6c 和 6d 显示了利用较低节点层（M, N, O）继续的判定树 1 的混合存储方案的实例。例如，如图 6c 所示，宽度优先方案可以被用到层三（L），并从那点开始使用深度优先方案。作为选择，如图 6d 所示，全宽度优先方案可以被用到层三（L），并且然后以宽度优先顺序单独存储层三（L）上的每个节点的子树。这可以进行以允许可能需要的节点的更快速存储器存取。

这些存储方案的主要目的是允许树的存储而在树结构中不使用链路或者指针。为了确保适当操作，存储方案必须使得在链路除去之后，以可以用来精选上下文的节点总是按照存储顺序跟随父节点的方式，顺序地在存储器中安排这些节点。

在上述方式中，逐个分支地存储树的节点。来自判定树的单个内部节点 I 包含具有以下信息的数据元素：

-属性值，比如字母

-区分内部节点 I/叶 L 的一个比特

-属性类型 a_i

-对应于特定上下文的音素 p_i。

单一叶 L 节点包含具有以下信息的数据元素：

-属性值，比如字母

-区分内部节点 I/叶 L 的一个比特

-对应于特定上下文的音素 p_i

-指示这是否为父节点的最后叶的一个比特。

利用所建议的方案，可以最小化判定树模型的存储器需求。为了后面的用途，定义判定树的数据元素为属性类型、属性值或音素。

3.用于表示判定树中数据元素的方法

本发明的这一部分描述用于表示判定树的数据元素以实现最小存储器需求的三种方法。所建议的方法是固定比特分配、用于判定树数据元素的可变比特分配以及判定树数据元素的霍夫曼（Huffman）编码。

用于判定树数据元素的固定比特分配

判定树的尺寸是所有内部节点和叶的尺寸之和。下面分析内部节点和叶的尺寸。这里所述的数量用于英语语言。

a）属性值的数量:

存在 26 个字母，少于 64 个音素以及少于 16 个音素类别。其中的最大值是 64，因此把 6 个比特分配用于属性值。

b）属性类型的数量:

对于 4 的上下文长度，在当前字母左边和右边具有 4 个字母、在左边具有 4 个音素和在当前字母左边具有 4 个音素类别。这使得总数为 16，并因此把 4 个比特分配用于属性类型。

c）音素的数量:

对于英语语言，数量在 32 和 64 之间，所以 6 个比特被分配给音素。

d）表示内部节点/叶的标志仅需要一个比特。

e）表示用于给定内部节点的叶的尾部的标志仅需要一个比特。

上述的比特分配被称作固定比特分配，因为比特数量是预定的和固定的。

内部节点和叶的尺寸可以如下确定:

对于内部节点，尺寸是条目 a）、b）、c）和 d）之和:

Internal_node_size（内部节点尺寸）= 6+4+6+1 = 17 比特

对于叶，尺寸是条目 a）、b）、d）和 e）之和:

Leave_size（叶尺寸）= 6+6+1+1 = 14 比特

用于判定树数据元素的可变比特分配

在基于判定树文本-音素映射中，每个树对应于一个字母。在本发明的部分 1 中，建议剪切方法。对于给定字母，在对准字典中仅允许表 2 中列出的相应音素。因此，对于每个树，由表 2 限定音素的数量。例如，字母“a”具有 7 个可能的音素，其中需要 3 个比特分配用于音素，而不是如上（c）所述把 6 个比特分配给所有音素。这是因为只有 7 个音素用于字母“a”，所有其他的在对准期间被剪切掉。现在，对于内部节点和叶，比特数量都减少 3。当然，分配给音素的比特数量对于不同的叶将是不同的。

在剪切方法中，每个字母 1 仅仅可以映射到字母相关音素集。

1？p，其中 $p \in \{p_1, p_2, \ldots, p_n\}$.

可以根据字母相关音素集，而不是根据整个语言相关音素集，对音素编码。该方法被称作可变比特分配，因为分配给音素的比特数量可以随字母和树而变化。例如，利用固定比特分配的方法，把“a”映射到整个集（40 个音素），当利用可变比特分配的方法时，能够把字母“a”映射到字母相关音素集（英语中的 8 个音素）。这样，利用固定比特分配，需要[log2（40）]=6 比特；而利用可变比特分配，需要[log2(8)]=3 比特。

把字母相关比特分配用于其它数据元素比如属性类型和属性值是可能的。为此，对于给定字母，需要发现所有可能的属性类型和属性值的集合。一旦获知这些集合，就可以计算属性类型和属性值所需的比特数量。

为了把可变比特分配用于判定树数据元素，为每个字母找到允许的音素、属性类型和属性值的集合。一旦获知这些集合，就把它们存储到表中。如果用于数据元素的表的尺寸是 n，则利用可变比特分配存储数据元素所需的比特数量是[log2(n)]比特。该表需要被存储在引入开销的存储器中。因此，只在由于可变比特分配而导致的节省（Saved_var_bits）大于存储表的开销（Overhead_var_bits）时，才使用可变比特分配。

节省比特的数量按以下方式计算:

Saved_var_bits =

(Num_bits_fixed-Num_bits_variable)Count_occurrence

Num_bits_fixed 对应于利用固定比特分配分配给数据元素的比特数量。Num_bits_variable 对应于利用可变比特分配分配给数据元素的比特数量。Count_occurrence 是数据元素出现在判定树中的总次数。

用于数据元素的存储表的开销按以下方式计算:

Overhead_var_bits =(Size_table+1)Bits_in_byte

Size_table 对应于表中元素的数量，以及 Bits_in_byte 为 8。

为每个数据元素(属性类型,属性值和音素)检查 Saved_var_bits 与 Overhead_var_bits 之间的比较，并且如果 Saved_var_bits 大于 Overhead_var_bits，则使用可变比特分配:

if Saved_var_bits > Overhead_var_bits

判定树数据元素的霍夫曼编码

为了把二进制代码分配用于判定树数据元素，可以使用霍夫曼码。如果判定树数据元素的分布具有大变化，则霍夫曼码的使用可以节省存储器。霍夫曼编码的基本思想是把短码字分配给具有高概率的数据元素，并把长码字分配给具有低概率的数据元素。霍夫曼码是最佳的和无损耗的。该码是可变长度码，其中利用对应于特定数据元素的码字的长度给出用于编码数据元素的比特数量。霍夫曼码必须单独地为每个判定树变量导出。

下表 3 显示了用于英语的字母“a”树的音素的霍夫曼编码的一个实例。音素的压缩比是 1.2554。表中的“FLC”代表固定长度码。

音素	A:	ä	aI	E	eI	I	V	_
概率	0.2238	0.277	0.0019	0.0250	0.1211	0.0075	0.2650	0.0780
FLC	000	001	010	011	100	101	110	111
霍夫曼码	10	01	110000	11001	111	110001	00	1101

表 3: 判定树中用于美国英语字母“a”的音素的编码

为了将霍夫曼编码用于判定树数据元素，霍夫曼码字、用于每个

码字的比特数量和相应的字母表需要被存储在引入开销的存储器中。为每个数据元素，单独进行是否使用霍夫曼编码的判定。对于给定的数据元素，只在霍夫曼编码带来的节省(Saved_huff_bits) 大于开销(Overhead_huff_bits)时，才可以使用霍夫曼编码。

霍夫曼编码带来的节省比特的数量可以根据下式计算:

$$Saved\_huff\_bits = \sum_i (Num\_bits\_fixed - Num\_bits\_CW_i) Count\_CW_i$$

Num_bits_fixed 是利用固定比特分配分配给数据元素的比特数量。$Num\_bits\_CW_i$ 对应于分配给出现在霍夫曼编码树的第 i 码字的比特数量。

存储霍夫曼码的开销能够根据下式计算:

Overhead_huff_bits=(3 Num_huff_CW+1)Bits_per_byte

Num_huff_CW 是用于数据元素的霍夫曼码字的数量，而 Bits_per_byte 是 8。假定霍夫曼码字、指示霍夫曼码字中比特数量的变量和字母表的成员存储在单个字节变量中。

为每个数据元素（属性类型，属性值和音素）检查 Saved_huff_bits 与 Overhead_huff_bits 之间的比较，并且如果确定的条件满足为 if Saved_huff_bits > Overhead_huff_bits，则对于数据元素应用霍夫曼编码。

存储判定树数据元素所需的存储器的最小化

如同在部分 2 的开头所解释的那样，存储树的基本结构是固定的，但是能够以各种方式表示树中的数据元素。比特分配可以是固定的，或者可以是可变的，或者可以使用霍夫曼编码。为判定树中的每个数据元素（属性类型，属性值和音素）进行这些编码方法之间的判定。由于对于每个字母具有基于判定树的文本-音素模型，因此可以为每个字母重复选择。

在可替代实施例中，利用使用所谓的截短霍夫曼编码的可能性来辅助是使用固定长度编码还是霍夫曼编码的判定。在该编码方法中，具有非常低概率的数据元素的某些值被组合在一起，并且把公用霍夫

曼前缀码分配给该组。然后利用固定长度码对该组值中数据元素的实际值进行编码。例如，8 个非常不大可能的值的组可以用具有假定 7 比特、其后跟随 3 比特的固定长度码的霍夫曼码进行编码。

对于给定树中的给定数据元素，进行是使用固定比特分配、可变比特分配还是霍夫曼编码的选择，以便最小化用于低速存储器判定树模型的存储器需求。因此，判定基于以下逻辑：

a）初始化：假设固定比特分配用于数据元素；

b）如果 Saved_var_bits > Overhead_var_bits，则使用可变比特分配用于数据元素；

c）如果霍夫曼编码带来的节省大于可变比特分配带来的节省并大于霍夫曼码引入的开销，则将霍夫曼编码用于数据元素：

Saved_huff_bits-Overhead_huff_bits>Saved_var_bits- Overhead_var_bits

以及

Saved_huff_bits > Overhead_huff_bits

本发明利用该最小化方案来自动确定用于所有判定树中的每个判定树数据元素的最小比特分配。

实验：

本发明通过在从 CUM（Carnegie Mellon University 卡内基梅隆大学）字典中提取的美国 Census（人口普查）姓名列表的发音上训练判定树进行实验。发音的总数是 40,529。基本实施利用与判定树数据元素的固定比特分配的原始对准。如下表 4 所示，判定树模型尺寸被明显减小（36％），并且在音素精度和串速率（string rate）方面，文本-音素性能没有任何恶化。这验证本发明的有用性。

方法	存储器 (kB)	音素精度	串速率
现有技术	132.5	99.27 %	97.12 %
本发明	84.9	99.27%	97.17 %

表 4：现有技术方案与本发明方案之间的判定树比较

在本发明中已经介绍了不同的创新技术：对准中的剪切方法、判定树结构、用于判定树数据元素的固定比特分配、用于判定树数据元

素的可变比特分配和用于判定树数据元素的霍夫曼编码比特。显然，所有的技术可以被单独地利用或者以不同方式进行组合，这就是为什么本发明的描述不应被考虑为本发明的限制的原因。

文本-音素系统可以被实施为电子设备中的语音识别系统的一部分，例如被实施为数字信号处理单元。电子设备可以包括其它功能，比如蜂窝电话 T（图 7）中的用于电信的装置。该设备最好包括扬声器 H、麦克风 M。文本-音素系统还有可能在与电子设备一起使用的并发设备内实现。如果把蜂窝电话考虑为电子设备，则并发设备可以是例如耳机或者视频护目镜。

文本-音素系统可以另外在普遍存在的环境中使用，其中该系统可以被实施在住房的各个房间中、各种家用电器（例如，电视，洗衣机）中、家具中或者耐磨附件（例如衣服）中。

显然，上述实施例不应解释为本发明的限制，并且这些实施例可以在以下权利要求中所提出的发明特征的范围内变化。

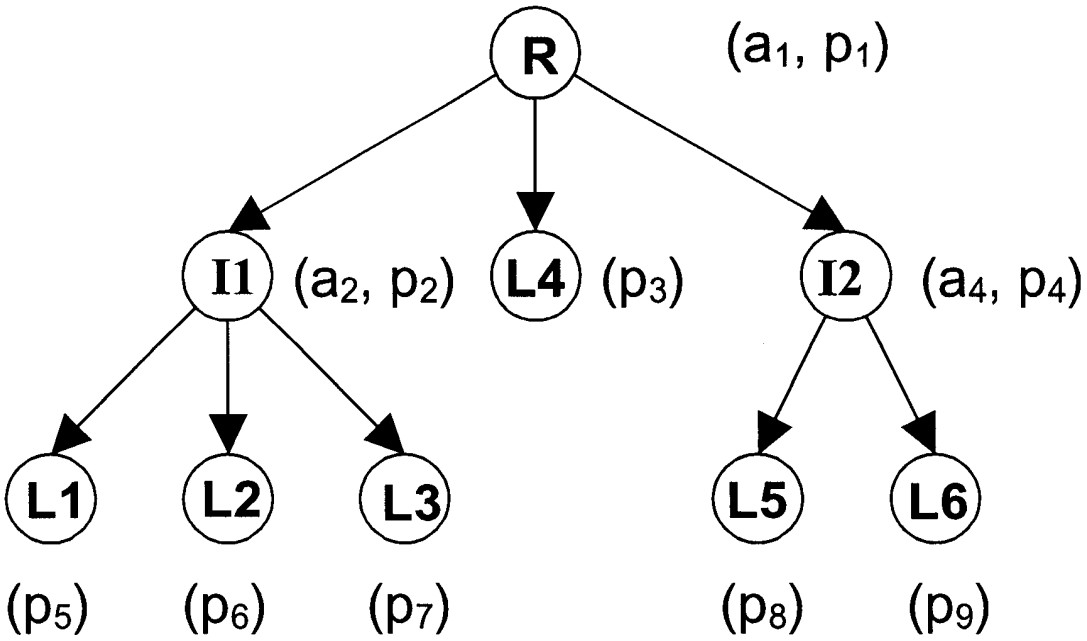

R
(a1, p1)
I1
(a2, p2)
L4
(p3)
I2
(a4, p4)
L1
L2
L3
L5
L6
(p5)
(p6)
(p7)
(p8)
(p9)

图 1

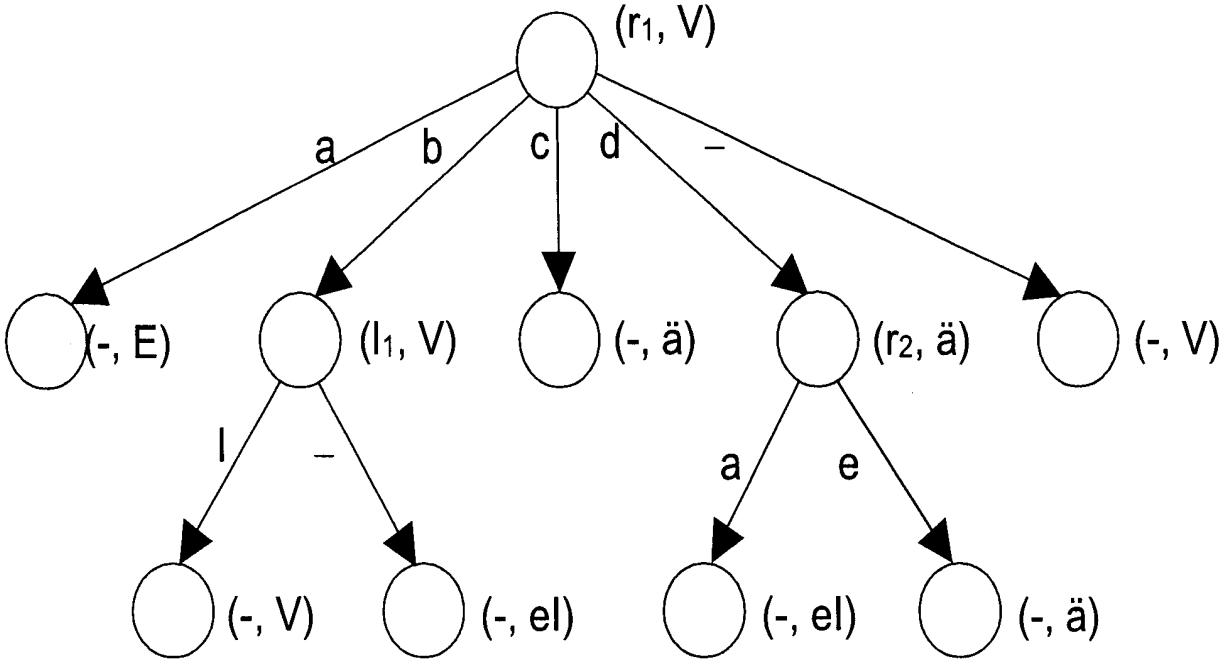

(r1, V)
a
b
c
d
–
(-, E)
(l1, V)
(-, ä)
(r2, ä)
(-, V)
l
–
a
e
(-, V)
(-, el)
(-, el)
(-, ä)

图 2

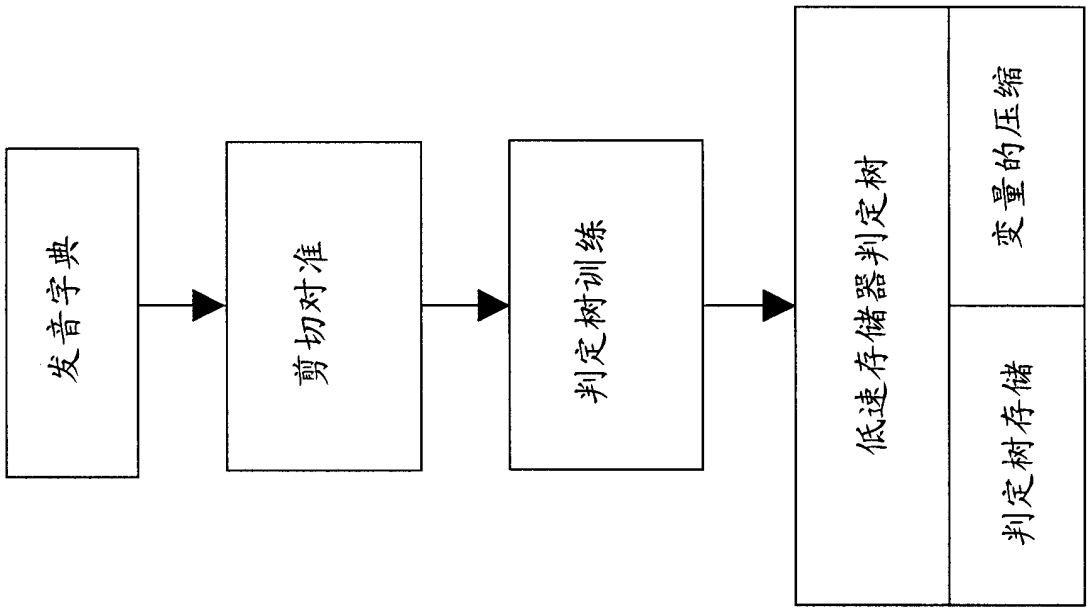


图 4

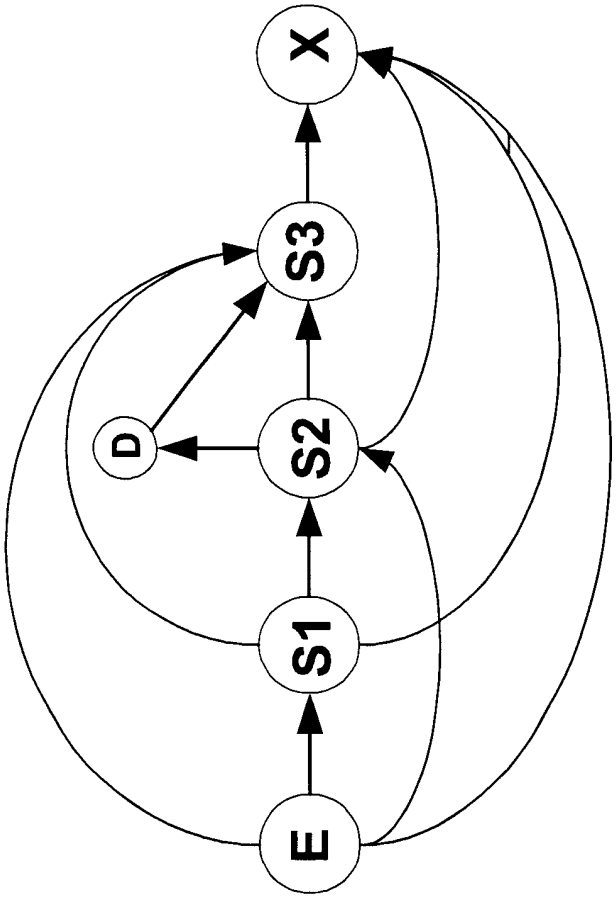


图 3

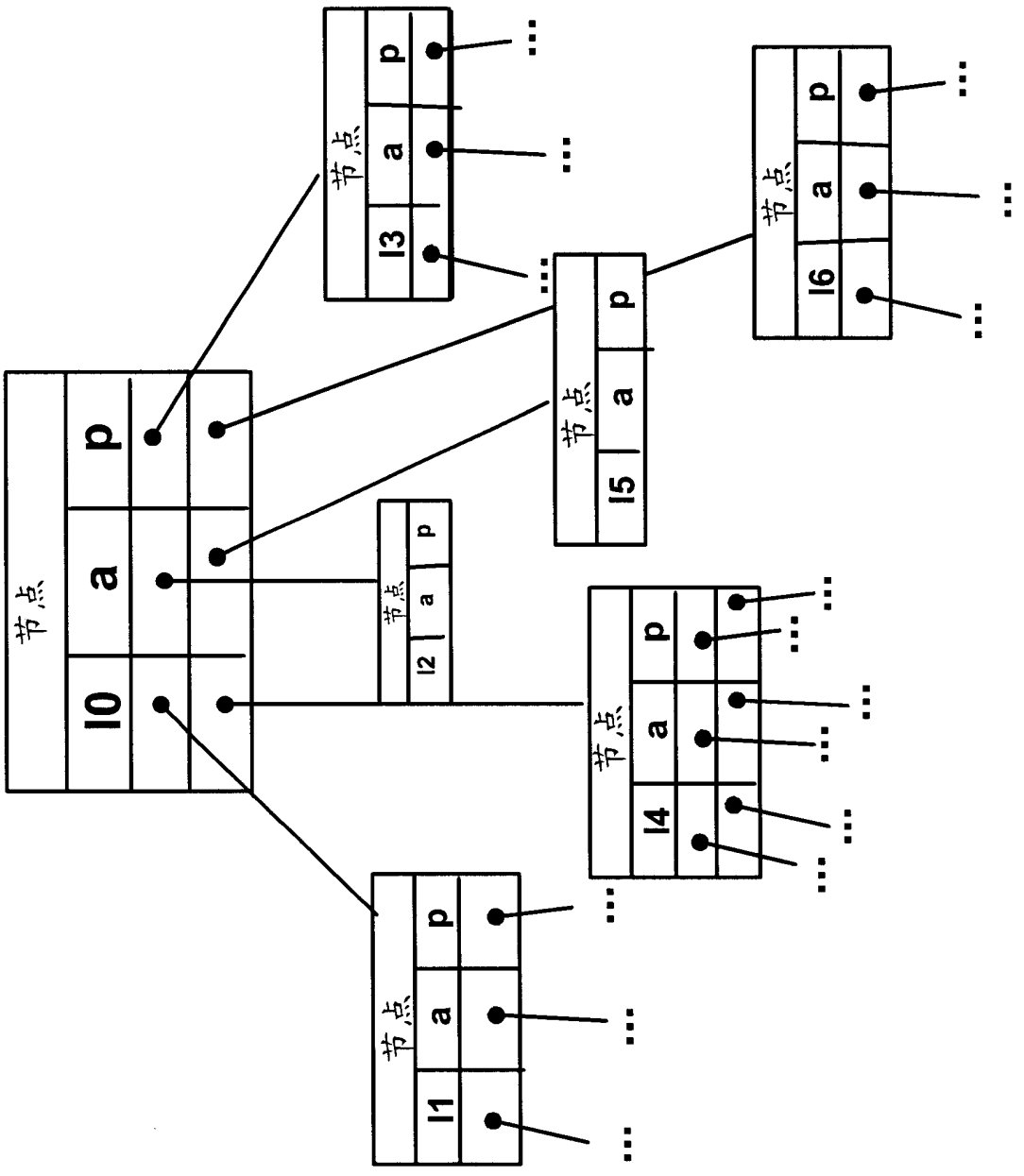


图 5

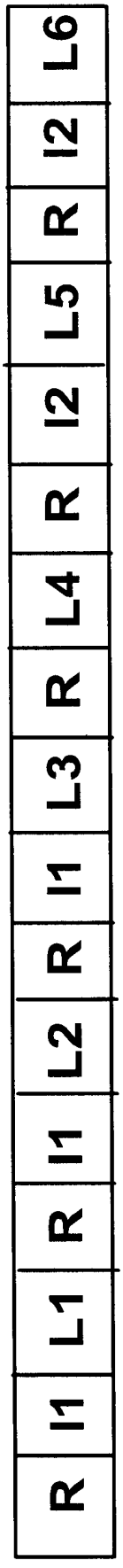


图 6a

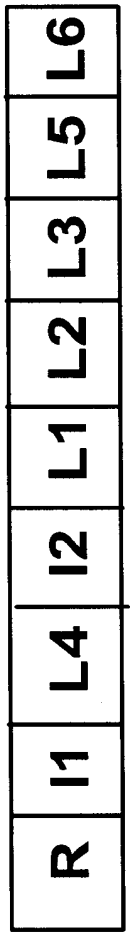


图 6b

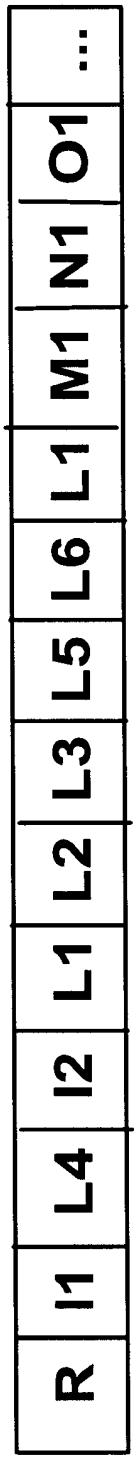


图 6c

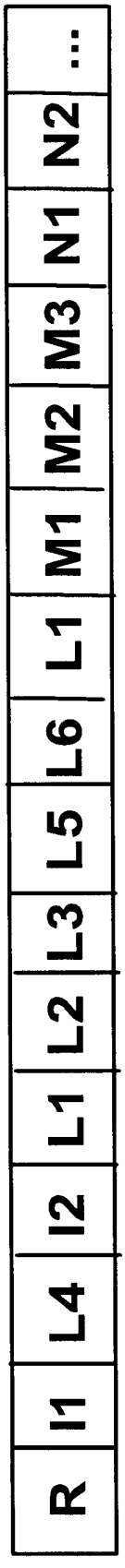


图 6d

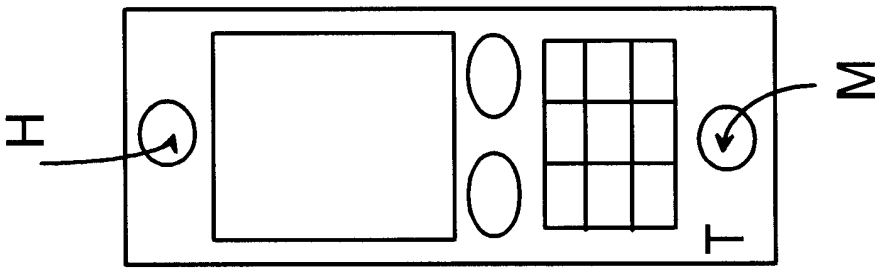


图 7