[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H03M 7/30

[12] 发明专利申请公开说明书

[21] 申请号 03821724.4

[43] 公开日 2005 年 10 月 12 日

[11] 公开号 CN 1682448A

[22] 申请日 2003.7.11 [21] 申请号 03821724.4

[30] 优先权

[32] 2002. 7.12 [33] US [31] 60/395,481

[32] 2003. 5.14 [33] US [31] 10/438,357

[86] 国际申请 PCT/CA2003/001058 2003.7.11

[87] 国际公布 WO2004/008644 英 2004.1.22

[85] 进入国家阶段日期 2005.3.14

[71] 申请人 斯利普斯特里姆数据公司

地址 加拿大安大略

[72] 发明人 杨恩辉 何大可

[74] 专利代理机构 永新专利商标代理有限公司

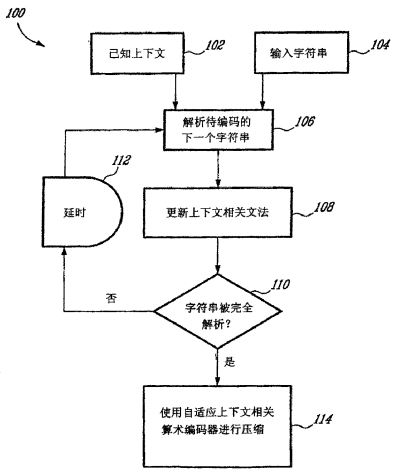
代理人 韩 宏

权利要求书 18 页 说明书 25 页 附图 13 页

[54] 发明名称 使用贪婪的顺序上下文相关文法变换的改进的无损耗数据压缩方法

[57] 摘要

本发明提供了一种无损耗数据压缩方法，它使用文法变换顺序地构造一个贪婪的上下文相关文法的序列，可以从该序列中逐步恢复原始数据序列。使用顺序上下文相关方法、改进的顺序上下文相关方法和等级式上下文相关方法中的任意一种，对该数据序列进行编码。



ISSN1008-4274

知识产权出版社出版

1、一种将原始数据序列顺序地变换为不可约上下文相关文法的方法，所述原始数据序列包括多个符号且与已知上下文模型相关联，根据所述不可约上下文相关文法可完全恢复所述原始数据，其中，所述不可约上下文相关文法是由产生规则集合表示的，这些产生规则是使用表示所述数据序列中的不重叠重复模式和上下文的变量和上下文的对的集合所形成的，所述方法适用于任何可数上下文模型，并包括以下步骤：

(a) 从所述序列中解析子串，其中，如果所述序列以前未解析的符号的字符串的最长前缀存在，则所述子串是所述最长前缀，否则，所述子串是所述序列以前未解析的符号的字符串的第一个符号，其中，在当前时间点的上下文时，所述序列以前未解析的符号的字符串用前一个不可约上下文相关文法的变量集合中的变量、而不是变量和上下文对的集合的初始对中的初始变量来表示；

(b) 根据所述子串、当前上下文和所述前一个不可约上下文相关文法，产生容许的上下文相关文法；

(c) 将至少一个约简规则集合应用于所述容许的上下文相关文法，以产生新的不可约上下文相关文法；

(d) 重复步骤(a)至(c)，直到用最终不可约上下文相关文法表示所述序列的所有符号。

2、如权利要求 1 所述的方法，其中，所述应用步骤还包括执行约

简规则的步骤，该约简规则包括以下步骤：

(a) 将 s 定义为在容许的上下文相关文法 G 的范围内在某上下文 C 时出现一次的 G 的变量；

(b) 定义产生规则$s'|\hat{C}\rightarrow\alpha s\beta$，其中，在上下文 C 时，s 出现在一组符号的右边；

(c) 定义与对(C,s)相对应的产生规则 s|C→γ；以及

(d) 通过从 G 中删除所述产生规则 s|C→γ，并将所述产生规则$s'|\hat{C}\rightarrow\alpha s\beta$替换为产生规则$s'|\hat{C}\rightarrow\alpha\gamma\beta$，来将 G 约简为容许的上下文相关文法 G′，所得的容许的上下文相关文法 G′和 G 表示等价的数据序列。

3、如权利要求 1 所述的方法，其中，所述应用步骤还包括执行约简规则的步骤，该约简规则包括以下步骤：

(a) 将 G 定义为具有形式为$s|C\rightarrow\alpha_1\beta\alpha_2\beta\alpha_3$的产生规则的容许的上下文相关文法，其中，β 的长度至少为 2，并且 β 在这两个位置的主要上下文等于相同的上下文 C′；

(b) 定义新的产生规则$s'|C'\rightarrow\beta$，其中，(C′,s′)是 G 的域中没有的上下文和变量的新对；以及

(c) 通过将 G 的产生规则$s|C\rightarrow\alpha_1\beta\alpha_2\beta\alpha_3$替换为$s|C\rightarrow\alpha_1 s'\alpha_2 s'\alpha_3$，并添加产生规则$s'|C'\rightarrow\beta$，来将 G 约简为新的上下文相关文法 G′，所得的文法 G′具有上下文和变量的新对（C′,s′），并表示与 G 等价的数据序列 x。

4、如权利要求 3 所述的方法，其中，当在 G'的范围内，G'具有所述符号和上下文的不重叠重复模式时，执行所述约简步骤，以替代所述产生规则集合中的其他规则。

5、如权利要求 1 所述的方法，其中，所述应用步骤还包括执行约简规则的步骤，该约简规则包括以下步骤：

(a) 将 G 定义为具有形式为 $s|C \rightarrow \alpha_1\beta\alpha_2$ 和 $s'|C' \rightarrow \alpha_3\beta\alpha_4$ 的两个产生规则的容许的上下文相关文法，其中，β 的长度至少为 2，β 在这两个位置的主要上下文等于相同的上下文 C''，α_1 和 α_2 之一不为空，并且 α_3 和 α_4 之一不为空；

(b) 定义新的产生规则 $s''|C'' \rightarrow \beta$，其中，(C'',s'')是 G 的域中没有的上下文和变量的新对；以及

(c) 通过将产生规则 $s|C \rightarrow \alpha_1\beta\alpha_2$ 替换为 $s|C \rightarrow \alpha_1 s''\alpha_2$，并将产生规则 $s'|C' \rightarrow \alpha_3\beta\alpha_4$ 替换为 $s'|C' \rightarrow \alpha_3 s''\alpha_4$，并添加新规则 $s''|C'' \rightarrow \beta$，来将 G 约简为新的上下文相关文法 G'。

6、如权利要求 1 所述的方法，其中，所述应用步骤还包括执行约简规则的步骤，该约简规则包括以下步骤：

(a) 将 G 定义为具有形式为 $s|C \rightarrow \alpha_1\beta\alpha_2$ 和 $s'|C' \rightarrow \beta$ 的两个产生规则的容许的上下文相关文法，其中，β 的长度至少为 2，$s|C \rightarrow \alpha_1\beta\alpha_2$ 中的 β 的上下文也等于上下文 C'，并且 α_1 和 α_2 之一不为空；

(b) 通过将所述产生规则 $s|C \rightarrow \alpha_1\beta\alpha_2$ 替换为新的产生规则

$s|C \to \alpha_1 s' \alpha_2$，来将 G 约简为上下文相关文法 G'。

7、如权利要求 1 所述的方法，其中，所述应用步骤还包括执行约简规则的步骤，该约简规则包括以下步骤：

(a) 将 G 定义为容许的上下文相关文法，其中，在给定的上下文 C 时的两个变量 s 和 s'表示由 G 表示的 A 序列的相同的子串；

(b) 通过在 G 的范围内，当 s'的上下文等于 C 时，将 s'的每次出现替换为 s，并删除(C, s')所对应的产生规则，来将 G 约简为新的上下文相关文法 G'。

8、如权利要求 7 所述的方法，其中，所述约简步骤还包括以下步骤：如果所述文法 G'是不容许的，则通过将 G'的连续替换过程中未涉及到的 G'的上下文和变量对所对应的各所述产生规则删除，来将 G'进一步约简为容许的上下文相关文法 G"。

9、一种将原始数据序列 $x=x_1x_2\cdots\cdots x_n$ 顺序地变换为不可约上下文相关文法 $\{G_i\}_{i=1}^{t}$ 的序列的方法，所述原始数据序列包括多个符号并与已知上下文模型相关联，所述已知上下文模型由可数上下文集合 C 和下一个上下文函数 f 给出，可从所述不可约上下文相关文法的序列中递增地完全恢复所述原始数据序列 x，其中，每个上下文相关文法 G_i 是由产生规则 $s_i|C_i \to G_i(s_i|C_i)$ 的集合表示的，所述产生规则是由变量集合 $S(j_i)=\{s_0,s_1,\ldots,s_{j_i-1}\}$ 以及上下文和变量对的集合 $\Omega_G(K_i,j_i)$ 形成的，$j_1=1$，

$K_1=1$，并且$\Omega_G(K_1,j_1)=\{(C_1,s_0)\}$，所述方法包括以下步骤：

(a) 将序列 $x=x_1x_2\cdots\cdots x_n$ 解析为 t 个不重叠的子串 $\{x_1,x_2\cdots x_{n_2},\ldots,x_{n_{i-1}+1}\cdots x_{n_i}\}$，其中 $n_1=1$，$n_t=n$，并且对于每个 $1<i\leqslant t$，如果以前未解析的符号的剩余序列 $x_{n_{i-1}+1}\cdots x_n$ 存在最长前缀，则子串 $x_{n_{i-1}+1}\cdots x_{n_i}$（用 β_i 表示）是所述最长前缀，否则，所述子串就是以前未解析的符号的剩余序列的第一个符号 $x_{n_{i-1}+1}$，其中在上下文为 $C_{n_{i-1}+1}$ 时，该以前未解析的符号的剩余序列可以由 $\{s:s\neq s_0\ \&\ (C_{n_{i-1}+1},s)\in\Omega_G(K_{i-1},j_{i-1})\}$ 给出的变量子集中的变量 s_j 来表示，其中，$\Omega_G(K_{i-1},j_{i-1})$ 是前一个不可约上下文相关文法 G_{i-1} 的上下文和变量对的集合；以及

(b) 根据当前解析的子串 $\beta_i=x_{n_{i-1}+1}\cdots x_{n_i}$、从所述上下文模型确定的当前上下文 $C_{n_{i-1}+1}$ 和所述前一个不可约上下文相关文法 G_{i-1}，来为每个 $x_1\cdots x_{n_i}$ 产生不可约上下文相关文法 G_i，其中，G_1 只包括一个产生规则 $\{s_0|C_1\rightarrow x_1\}$，并且 (C_1,s_0) 是初始上下文和变量对。

10、如权利要求 9 所述的方法，其中，将所述序列 $x=x_1x_2\cdots\cdots x_n$ 解析为多个不重叠子串 $\{x_1,x_2\cdots x_{n_2},\ldots,x_{n_{i-1}+1}\cdots x_{n_i}\}$ 的步骤还包括以下步骤：

(a) 对于每个 $1<i\leqslant t$，确定在时间点 $n_{i-1}+1$ 的上下文 $C_{n_{i-1}+1}$；

(b) 判断在当前上下文 $C_{n_{i-1}+1}$ 时，剩余未解析的序列 $x_{n_{i-1}+1}\cdots\cdots x_n$ 的前缀能否由 $\{s:s\neq s_0\ \&\ (C_{n_{i-1}+1},s)\in\Omega_G(K_{i-1},j_{i-1})\}$ 给出的变量子集中的变量 s_j 表示，其中，$\Omega_G(K_{i-1},j_{i-1})$ 是所述前一个不可约上下文相关文法 G_{i-1} 的上

下文和变量对的集合；

(c) 如果$x_{n_{i-1}+1}\cdots x_{n_i}$在上下文$C_{n_{i-1}+1}$时用变量 s_j 表示，则设置当前解析的子串 β_i 等于最长前缀$x_{n_{i-1}+1}\cdots x_{n_i}$；

(d)如果在当前上下文$C_{n_{i-1}+1}$时，没有$x_{n_{i-1}+1}\cdots\cdots x_n$的前缀可以用$\{s: s\neq s_0 \,\&\, (C_{n_{i-1}+1}, s)\in \Omega_G(K_{i-1}, j_{i-1})\}$给出的变量子集中的变量 s_j 表示，则设置当前解析的子串 β_i 等于$x_{n_{i-1}+1}$，其中，$n_i = n_{i-1}+1$。

11、如权利要求 9 所述的方法，其中，根据当前解析的子串$\beta_i = x_{n_{i-1}+1}\cdots x_{n_i}$、当前上下文$C_{n_{i-1}+1}$和所述前一个不可约上下文相关文法 G_{i-1}，为每个$x_i\cdots x_{n_i}$产生不可约上下文相关文法 G_i 的步骤还包括以下步骤：

(a) 如果 $n_i-n_{i-1}>1$ 并且$\beta_i = x_{n_{i-1}+1}\cdots x_{n_i}$在上下文$C_{n_{i-1}+1}$时用 s_j 表示，则将变量 s_j 添加到 $G_{i-1}(s_0|C_1)$的右端，以产生容许的上下文相关文法G'_{i-1}；

(b) 如果 $n_i-n_{i-1}=1$，则将符号$x_{n_{i-1}+1}$添加到 $G_{i-1}(s_0|C_1)$的右端，以产生容许的上下文相关文法G'_{i-1}；以及

(c) 将约简规则集合应用于G'_{i-1}，从而为每个$x_i\cdots x_{n_i}$产生不可约上下文相关文法 G_i。

12、如权利要求 11 所述的方法，其中，所述将约简规则集合应用于G'_{i-1}的步骤还包括以下步骤：

(a) 如果所述容许的上下文相关文法$G_{i-2}^{'}$是可约的，则将与第(i-i)个解析的子串相关联的文法约简比特 I(i-1)设置等于 1，如果$G_{i-2}^{'}$是不可约的，则将 I(i-1)设置为 0；

(b) 将α定义为$G_{i-1}(s_0|C_1)$的最后一个符号；

(c) 将 C 定义为$G_{i-1}(s_0|C_1)$中的α的上下文；

(d) 将β定义为$G_{i-1}'(s_0|C_1)$的最后一个符号；

(e) 判断α β在上下文 C 时是否出现在$G_{i-1}^{'}$的范围内的两个不重叠的位置；

(f) 如果α β在上下文 C 时没有出现在$G_{i-1}^{'}$的范围内的两个不重叠的位置，则设置 G_i 等于$G_{i-1}^{'}$；

(g) 如果 I(i-1)=0 并且α β在相同的上下文 C 时在$G_{i-1}'(s_0|C_1)$中的两个不重叠的位置重复自身，则对$G_{i-1}^{'}$应用第一约简规则，以产生 G_i，其中，所述第一约简规则包括以下步骤：

(1) 将 G 定义为具有形式为$s|\hat{C}\rightarrow\alpha_1\gamma\alpha_2\gamma\alpha_3$的产生规则的容许的上下文相关文法，其中，γ的长度至少为 2，并且γ在这两个位置的主要上下文等于相同的上下文 C′；

(2) 定义新的产生规则 s′|C′→γ，其中，(C′,s′)是 G 的域中没有的上下文和变量的新对；以及

(3) 通过将 G 的产生规则$s|\hat{C}\rightarrow\alpha_1\gamma\alpha_2\gamma\alpha_3$替换为$s|\hat{C}\rightarrow\alpha_1 s'\alpha_2 s'\alpha_3$，并添加所述产生规则 s′|C′→γ，来将 G 约简为新的上下文相关文法$\hat{G}$，所得的文法$\hat{G}$具有上下文和变量的新对（C′,s′），且表示与 G 等价的数据序列 x；

(h) 如果 I(i-1)=0 并且 α β 在相同的上下文 C 时在 G'_{i-1} 而非 $G'_{i-1}(s_0|C_1)$ 的范围内的两个不重叠的位置重复自身，则对 G'_{i-1} 应用第二约简规则，以产生 G_i，其中，所述第二约简规则包括以下步骤：

(1) 将G定义为具有形式为 $s|\hat{C}\rightarrow\alpha_1\gamma\alpha_2$ 和 $s'|C'\rightarrow\alpha_3\gamma\alpha_4$ 的两个产生规则的容许的上下文相关文法，其中，γ 的长度至少为 2，γ 在这两个位置的主要上下文等于相同的上下文 C''，α_1 和 α_2 之一不为空，并且 α_3 和 α_4 之一不为空；

(2) 定义新的产生规则 $s''|C''\rightarrow\gamma$，其中，(C'',s'')是 G 的域中没有的上下文和变量的新对；以及

(3) 通过将产生规则 $s|\hat{C}\rightarrow\alpha_1\gamma\alpha_2$ 替换为 $s|\hat{C}\rightarrow\alpha_1 s''\alpha_2$，并将所述产生规则 $s'|C'\rightarrow\alpha_3\gamma\alpha_4$ 替换为 $s'|C'\rightarrow\alpha_3 s''\alpha_4$，并添加新规则 $s''|C''\rightarrow\gamma$，来将 G 约简为新的上下文相关文法 $\hat{G}$；

(i) 如果I(i-1)=1并且 α β 在相同的上下文C时在 G'_{i-1} 的范围内的两个不重叠的位置重复自身，则应用所述第一约简规则和第二约简规则之一，然后应用第三约简规则，其中，所述第三约简规则包括以下步骤：

(1) 将 s 定义为在 G 的范围内在某上下文 $\hat{C}$ 时出现一次的容许的上下文相关文法 G 的变量；

(2) 定义产生规则 $s'|C'\rightarrow\alpha_1 s\beta_1$，其中，在上下文 $\hat{C}$ 时 s 出现在一组符号的右边；

(3) 定义与对（$\hat{C}$,s）相对应的产生规则 $s|\hat{C}\rightarrow\gamma$，并通过从 G 中删除所述产生规则 $s|\hat{C}\rightarrow\gamma$，并将所述产生规则 $s'|C'\rightarrow\alpha_1 s\beta_1$ 替换为所

述产生规则 $s'|C' \rightarrow \alpha_1 \gamma \beta_1$，来将 G 约简为容许的上下文相关文法 G，所得的容许的上下文相关文法 $\hat{G}$ 和 G 表示等价的数据序列。

13、如权利要求 12 所述的方法，其中，所述判断步骤(e)还包括以下步骤：

(a) 对于除初始对（C_1,s_0）之外的每对 $(\hat{C},\gamma) \in \Omega_G(K_{i-1}, j_{i-1})$，定义两个列表 $L_1(\gamma|\hat{C})$ 和 $L_2(\gamma|\hat{C})$，其中，所述列表 $L_1(\gamma|\hat{C})$ 包括具有如下性质的所有符号 $\eta \in A \bigcup S(j_{i-1})$：

(1) 在上下文 $\hat{C}$ 时，模式 γ η 出现在 G_{i-1} 的范围内；

(2) 模式 γ η 不是 $G_{i-1}(s_0|C_1)$ 的最后两个符号；以及

(3) 没有 G_{i-1} 的变量 s_j，使得 $G_{i-1}(s_j|\hat{C})$ 等于 γ η，并且，其中所述列表 $L_2(\gamma|\hat{C})$ 包括具有性质（1）和（2）的所有符号 $\eta \in A \bigcup S(j_{i-1})$；

(b) 如果 β 出现在列表 $L_1(\alpha|C)$ 中，则表示在上下文 C 时，α β 出现在 G'_{i-1} 的范围内的两个不重叠的位置。

14、一种通过使用自适应上下文相关算术编码来编码原始数据序列以编码不可约上下文相关文法的方法，所述原始数据序列包括多个符号且与由可数上下文集合 C 和下一个上下文函数 f 给出的已知上下文模型相关联，根据所述不可约上下文相关文法可完全恢复所述原始数据序列，其中，所述不可约上下文相关文法是由产生规则集合表示的，所述产生规则是由表示所述数据序列中的不重叠重复模式和上下文的变量和上下文对的集合形成的，所述方法包括以下步骤：

(a) 将所述数据序列变换为所述不可约上下文相关文法，可从该不可约上下文相关文法中完全恢复所述原始数据序列；

(b) 将所述不可约上下文相关文法转换为其分类形式，产生分类的不可约上下文相关文法；

(c) 从所述分类的不可约上下文相关文法构造一个产生的序列；以及

(d) 使用具有动态字母表的自适应上下文相关算术编码，对该产生的序列进行编码。

15、如权利要求 14 所述的方法，其中，所述变换步骤(a)还包括以下步骤：

(1) 从所述序列中解析子串，其中，如果所述序列中以前未解析的符号的字符串的最长前缀存在，则所述子串是所述最长前缀，否则，所述子串是所述序列以前未解析的符号的字符串的第一个符号，其中，在当前时间点的上下文时，所述序列中以前未解析的符号的字符串可用前一个不可约上下文相关文法的变量集合中的一变量，而不是变量和上下文对的集合的初始对中的初始变量来表示；

(2) 根据所述子串、当前上下文和所述前一个不可约上下文相关文法，来产生容许的上下文相关文法；

(3) 将至少一个约简规则集合应用于所述容许的上下文相关文法，以产生新的不可约上下文相关文法；以及

(4) 重复步骤(1)至(3)，直到所述序列的所有符号被处理，并获取

了所期望的最终不可约上下文相关文法。

16、如权利要求 14 所述的方法，其中，如果对于每个合法的上下文 C，具有变量集合 $S(j)=\{s_0,s_1,\cdots s_{j-1}\}$ 以及上下文和变量对的集合$\Omega_G(K,j)$的不可约上下文相关文法 G 是分类形式，则对于任何有意义的 i>0，(C,s_i)的替换总是按照以下递归替换过程发生在(C,s_{i+1})的替换之前：

(1) 令Ω_G表示$\Omega_G(K,j)$的动态变化的子集，开始时，Ω_G为空；

(2) 设置$u^{(0)}=G(s_0|C_1)$以及 i=0；

(3) 对于 i≥0，从左到右读取 $u^{(i)}$，将Ω_G中没有的第一对（C,s）替换为 G(s|C)，其中，C 是由所述上下文模型确定的 s 的上下文；

(4) 通过将对(C,s)插入到Ω_G来更新Ω_G；

(5) 对于 i=0,1,……，$|\Omega_G(K,j)|$-1，重复步骤(3)和(4)，从而使得除(C_1,s_0)之外的每对 $(C,s)\in\Omega_G(K,j)$ 正好被替换为 G(s|C)一次；以及

所述转换步骤(b)还包括以下步骤：在每个合法的上下文 C 时，重新命名变量 s_i，从而使得对于重命名的变量，对于任何有意义的 i>0，(C,s_i)的替换总是按照所述递归替换过程发生在(C,s_{i+1})的替换之前。

17、如权利要求 14 所述的方法，其中，所述构造步骤(c)还包括以下步骤：

(1) 定义具有变量集合 $S(j)=\{s_0,s_1,\cdots s_{j-1}\}$ 以及上下文和变量对的集合$\Omega_G(K,j)$的分类的不可约上下文相关文法 G；

(2) 将Ω_G初始化为空集；

(3) 设置 $u^{(0)}=G(s_0|C_1)$以及 i=0；

(4) 对于 i≥0，从左向右读取 $u^{(i)}$，并将 Ω_G 中没有的第一对(C,s)替换为[G(s|C)]，其中，C 是由上下文模型确定的 s 的上下文；

(5) 将对(C,s)插入到 Ω_G，以更新 Ω_G；

(6) 对于 i=0,1,…… $|\Omega_G(K_j)|$-1，重复步骤(4)和(5)，从而使得除(C_1,s_0)之外的每对 $(C,s)\in\Omega_G(K,j)$ 正好被替换为[G(s|C)]一次，最终序列 $u^{(|\Omega_G(K,j)|-1)}$ 是从 G 产生的序列。

18、如权利要求 14 所述的方法，其中，所述编码步骤(d)还包括以下步骤：

(1) 对于任何上下文 C，将下一个上下文函数 f 扩展以包括 C=f(C,[)和 C=f(C,])；

(2) 将每对(C, α)与计数器 c(C, α)相关联，其中，C 是上下文，并且 α 是符号、变量或括号([或])；

(3) 如果 α 是所述原始数据序列中的符号或者是括号([或])，则将 c(C, α)初始化为 1，否则，将其初始化为 0；

(4) 在其上下文 C 的情况下，根据被所有满足 c(C, α)≠0 的 α 的所有计数器 c(C, α)的和所除的计数器 c(C, β)，对所述产生的序列的每个符号 β 进行编码；

(5) 将 c(C, β)加 1；

(6) 只要 β 等于括号]并且与 β 相对应的合法的括号对是在上下文 C′时从所述产生的序列的左边计数的第 i 个括号对，则将 c(C′, s_i)从 0

增加到 1，其中，C'是该括号对的左括号[的上下文；

(7) 确定所述产生的序列中的下一个符号的上下文；以及

(8) 重复步骤(4)至(7)，直到所述产生的序列中的所有符号都被编码。

19、一种将原始数据序列 $x=x_1x_2\cdots\cdots x_n$ 顺序地变换为不可约上下文相关文法 $\{G_i\}_{i=1}^{t}$ 的序列，并通过使用自适应上下文相关算术编码对所述数据序列进行编码以编码最终不可约上下文相关文法 G_t 的方法，其中，所述数据序列 $x=x_1x_2\cdots\cdots x_n$ 包括多个符号且与已知上下文模型相关联，所述已知上下文模型由可数上下文集合 C 和下一个上下文函数 f 给出，并且每个上下文相关文法 G_i 是由产生规则 $s_i|C_i \to G_i(s_i|C_i)$ 的集合表示的，所述产生规则是由变量集合 $S(j_i)=\{s_0,s_1,\ldots,s_{j_i-1}\}$ 和上下文和变量对的集合 $\Omega_G(K_i,j_i)$ 形成的，$j_1=1$，$K_1=1$，并且 $\Omega_G(K_1,j_1)=\{(C_1,s_0)\}$，所述方法包括以下步骤：

(a) 将所述序列 $x=x_1x_2\cdots\cdots x_n$ 解析为 t 个不重叠的子串 $\{x_1,x_2\cdots x_{n_2},\ldots,x_{n_{i-1}+1}\cdots x_{n_i}\}$，其中 $n_1=1$，$n_i=n$，对于各 $1<i\leqslant t$，如果以前未解析的符号的剩余序列 $x_{n_{i-1}+1}\cdots x_n$ 的最长前缀存在，则子串 $x_{n_{i-1}+1}\cdots x_{n_i}$（用 β_i 表示）是所述最长前缀，否则，该子串就是以前未解析的符号的剩余序列的第一个符号 $x_{n_{i-1}+1}$，在上下文 $C_{n_{i-1}+1}$ 时，所述以前未解析的符号的剩余序列可以用 $\{s:s\neq s_0\ \&\ (C_{n_{i-1}+1},s)\in\Omega_G(K_{i-1},j_{i-1})\}$ 给出的变量子集中的变量 s_j 来表示，其中，$\Omega_G(K_{i-1},j_{i-1})$ 是前一个不可约上下文相关文法 G_{i-1} 的上下文和变量的对的集合；以及

(b) 根据当前解析的子串 $\beta_i = x_{n_{i-1}+1} \cdots x_{n_i}$、从所述上下文模型确定的当前上下文 $C_{n_{i-1}+1}$ 和所述前一个不可约上下文相关文法 G_{i-1}，为每个 $x_1 \cdots x_{n_i}$ 产生不可约上下文相关文法 G_i，其中，G_i 只包括一个产生规则 $\{s_0|C_1 \rightarrow x_1\}$，并且$(C_1,s_0)$是初始的上下文和变量对；

(c) 将所述最终不可约上下文相关文法 G_t 转换为其分类形式 G_l^s；

(d) 从所述分类的不可约上下文相关文法 G_l^s，构造一个产生的序列；以及

(e) 使用具有动态字母表的自适应上下文相关算术编码，对所述产生的序列进行编码。

20、一种将数据序列顺序地变换为不可约上下文相关文法的序列，并通过使用自适应上下文相关算术编码对基于每个不可约上下文相关文法的所述数据序列进行编码的方法，其中，所述数据序列包括多个符号且与已知上下文模型相关联，所述已知上下文模型由可数上下文集合 C 和下一个上下文函数 f 给出，并且每个不可约上下文相关文法是由产生规则集合表示的，所述产生规则是使用表示所述数据序列中不重叠重复模式和上下文的变量和上下文对的集合所形成的，所述方法包括以下步骤：

(a) 从所述序列中解析子串，其中，如果所述序列以前未解析的符号的字符串的最长前缀存在，则所述子串是所述最长前缀，否则，所述子串是所述序列以前未解析的符号的字符串的第一个符号，在当前时间点的上下文时，所述序列以前未解析的符号的字符串可用前一个

不可约上下文相关文法的变量的集合中的变量，而不是变量和上下文对的集合的初始对中的初始变量来表示；

(b) 通过利用所述前一个不可约上下文相关文法的结构并使用自适应上下文相关算术编码，对所述子串进行编码；

(c) 根据所述子串、当前上下文和所述前一个不可约上下文相关文法，产生容许的上下文相关文法；

(d) 将至少一个产生规则集合应用于所述容许的上下文相关文法，以产生新的不可约上下文相关文法；以及

(e) 重复步骤(a)至(d)，直到所述序列中的所有符号都被解析和编码。

21、一种用于将原始数据序列 $x=x_1x_2\cdots\cdots x_n$ 顺序地变换为不可约上下文相关文法 $\{G_i\}_{i=1}^{t}$ 的序列，并通过使用自适应上下文相关算术编码，基于每个所述不可约上下文相关文法 $\{G_i\}_{i=1}^{t}$ 对所述数据序列进行编码的方法，其中，所述数据序列 $x=x_1x_2\cdots\cdots x_n$ 包括多个符号且与已知上下文模型相关联，所述已知上下文模型是由可数上下文集合 C 和下一个上下文函数 f 给出的，并且每个上下文相关文法 G_i 由产生规则 $s_i|C_i \rightarrow G_i(s_i|C_i)$ 的集合表示，所述产生规则是由变量集合 $S(j_i)=\{s_0,s_1,\ldots,s_{j_i-1}\}$ 和上下文和变量对的集合 $\Omega_G(K_i,j_i)$ 形成的，$j_1=1$，$K_1=1$，并且 $\Omega_G(K_1,j_1)=\{(C_1,s_0)\}$，所述方法包括以下步骤：

(a) 将各对(C, α)与两个计数器 c(C, α)和 $\hat{c}$(C, α)相关联，其中，C 是上下文，并且 α 是符号或变量；

(b) 如果α是所述原始数据序列中的符号，则将 c(C,α)和$c(C,\alpha)$初始化为 1，否则，将其初始化为 0；

(c) 对于每个 1<i≤t，从所述序列中解析子串$x_{n_{i-1}+1}\cdots x_{n_i}$，其中 n_1=1，n_i=n，并且对于各 1<i≤t，如果以前未解析的符号的剩余序列$x_{n_{i-1}+1}\cdots x_n$的最长前缀存在，则子串$x_{n_{i-1}+1}\cdots x_{n_i}$（用$\beta_i$表示）是所述最长前缀，否则，该子串是以前未解析的符号的剩余序列的第一个符号$x_{n_{i-1}+1}$，在上下文$C_{n_{i-1}+1}$时，所述以前未解析的符号的剩余序列可用$\{s: s\neq s_0 \,\&\, (C_{n_{i-1}+1}, s)\in \Omega_G(K_{i-1}, j_{i-1})\}$给出的变量子集中的变量 s_j 来表示，其中，$\Omega_G(K_{i-1}, j_{i-1})$是前一个不可约上下文相关文法 G_{i-1} 的上下文和变量的对的集合，并且其中，G_1只包括一个产生规则$\{s_0|C_1\rightarrow x_1\}$，$(C_1,s_0)$是初始的上下文和变量对；

(d) 根据所述前一个不可约上下文相关文法 G_{i-1}、当前子串β_i和当前上下文$C_{n_{i-1}+1}$，产生容许的上下文相关文法G'_{i-1}；

(e) 如果所述容许的上下文相关文法G'_{i-1}是可约的，则设置当前文法约简比特 I(i)为 1，如果G'_{i-1}是不可约的，则设置当前文法约简比特 I(i)为 0；

(f) 使用一阶或更高阶自适应算术编码，对所述当前文法约简比特 I(i)进行编码；

(g) 使用自适应上下文相关算术编码，根据所述前一个不可约上下文相关文法 G_{i-1}、文法约简比特 I(i)和 I(i-1)以及当前上下文$C_{n_{i-1}+1}$，对当前子串β_i进行编码；

(h) 将至少一个约简规则集合应用于所述容许的上下文相关文法

G'_{i-1}，以产生新的不可约上下文相关文法 G_i；以及

(i) 重复步骤(c)至(h)，直到所述序列 x 的所有符号都被解析和编码。

22、如权利要求 21 所述的方法，其中，所述编码步骤(g)还包括以下步骤：

(1) 将 α 定义为 $G_{i-1}(s_0|C_1)$ 的最后一个符号；

(2) 将 $\hat{C}$ 定义为 $G_{i-1}(s_0|C_1)$ 中 α 的上下文；

(3) 如果 $n_i-n_{i-1}>1$ 并且 $\beta_i = x_{n_{i-1}+1}\cdots x_{n_i}$ 在上下文 $C_{n_{i-1}+1}$ 时用 s_j 表示，则将 β 定义为变量 s_j；如果 $n_i-n_{i-1}=1$，则将其简单定义为符号 $x_{n_{i-1}+1}$；

(4) 将 $\tilde{C}$ 定义为上下文 $C_{n_{i-1}+1}$；

(5) 对于除初始对（C_1,s_0）之外的每对 $(\hat{C},\gamma)\in\Omega_G(K_{i-1},j_{i-1})$，定义两个列表 $L_1(\gamma|C)$和 $L_2(\gamma|C)$，其中，所述列表 $L_1(\gamma|C)$包括具有如下性质的所有符号 $\eta\in A\bigcup S(j_{i-1})$：

a) 在上下文 C 时，模式 γ η 出现在 G_{i-1} 的范围内；

b) 在上下文 C 时，模式 γ η 不是 $G_{i-1}(s_0|C_1)$ 的最后两个符号；

c) 没有 G_{i-1} 的变量 s_j，使得 G_{i-1}（$s_j|C$）等于 γ η，并且，其中所述列表 $L_2(\gamma|C)$包括具有性质 a）和 b）的所有符号 $\eta\in A\bigcup S(j_{i-1})$；

(6) 如果 I(i)=0，则在其上下文 $\tilde{C}$ 的情况下，根据被满足 $c(\tilde{C},\gamma)\neq 0$ 的所有 $\gamma\in A\bigcup S(j_{i-1})\setminus L_2(\alpha|\hat{C})$ 的所有计数 $c(\tilde{C},\gamma)$的和所除的计数器 $c(\tilde{C},\beta)$，对 β 进行编码，然后将 $c(\tilde{C},\beta)$加 1；

(7) 如果 I(i-1)=0 且 I(i)=1，则在其上下文 $\tilde{C}$ 的情况下，根据被所有

$\gamma \in L_1(\alpha|\hat{C})$的所有计数器$\hat{c}(\tilde{C},\gamma)$的和所除的计数$\hat{c}(\tilde{C},\beta)$，对β进行编码，然后将$\hat{c}(\tilde{C},\beta)$加1；

23、如权利要求21所述的方法，其中，所述步骤(h)还包括相应地更新列表$L_1(\gamma|C)$和$L_2(\gamma|C)$的步骤。

使用贪婪的顺序上下文相关文法变换的改进的无损耗数据压缩方法

技术领域

本发明一般涉及数据压缩领域，尤其涉及使用顺序文法变换（sequential grammar transform）和编码方法压缩具有已知上下文的数据。

背景技术

通用数据压缩方法可以分为两类：通用无损耗数据压缩和通用有损耗数据压缩。传统的通用无损耗数据压缩方法通常采用算术编码算法、Lempel-Ziv 算法及其变体。算术编码算法以待压缩数据的统计模型为基础。为了使用算术编码算法对数据序列进行编码，需要在编码过程中动态地建立一个统计模型，或者，假设该模型已经事先存在。现有一些方法可以动态地建立统计模型，包括通过部分匹配算法、动态马尔可夫建模、上下文收集算法和上下文树加权的预测。这些方法各使用合适的上下文，预测数据序列中的下一个符号，并使用其估计出的相应条件概率，对这些符号进行编码。

大多数算术编码算法及其变体仅对于马尔可夫阶数低于一些设计参数值的马尔可夫信源类型而言是通用的。并且，算术编码

算法对数据序列进行逐字符编码。

相比之下，Lempel-Ziv 算法及其变体不采用统计模型。Lempel-Ziv 算法根据字符串匹配机制，将原始数据序列解析为不重叠的变长短语，然后逐短语地将其编码。此外，Lempel-Ziv 算法对于比有界阶数马尔可夫信源类型要宽广的信源类型而言是通用的，有界阶数马尔可夫信源是平稳遍历信源。

其他传统的通用压缩方法包括动态霍夫曼（Huffman）算法、移至前端编码机制和一些利用代码本传输的两级压缩算法。这些传统方法要么次于算术编码和 Lempel-Ziv 算法，要么太复杂而无法实现。最近提出了一种基于替换表的新型无损耗数据压缩算法，它包括一个新的编码框架，但却没有引入明确的数据压缩算法。但是，该方法的缺点在于：编码过程中使用的贪婪顺序变换（greedy sequential transformation）很难实现，并且不利于进行高效的编码，因为解析过程中涉及到了初始符号 s_0。

此外，这些算法并不需要对压缩的数据序列具备任何先验知识。在使它们适合通用数据压缩需要的同时，这些算法对于数据压缩特定应用的效率不高。在很多情况下，如压缩 web 页面、java 小程序或文本文件时，对于压缩的数据序列通常要有先验知识，该知识通常采用所谓的“上下文模型”的形式。

我们需要的是一种充分利用压缩的数据序列的上下文的先验知识、并同时克服现有压缩方法的上述缺点的通用无损耗数据压缩方法。

发明内容

根据本发明，提供了一种通用无损耗数据压缩方法。该方法

采用信源编码构造在线的、可调的、上下文相关的压缩规则。不像以前使用的压缩单个目标的其他方法，该方法压缩规则的数字集合。例如，为了实现基于 web 的数据的在线压缩，该方法接收数据，动态地构造规则，然后对这些规则进行编码。因为规则集合是动态的，所以当 web 文本数据的结构发生改变时，相应的规则也被更新。同样，当 web 对象的内容更新时，对应的规则也相应地被更新。该方法对于 web 页面编码尤其高效，因为 web 页面的内容经常发生改变，而 web 页面的底层结构基本保持恒定。压缩数据时，该底层结构的相对一致性提供了数据的可预测上下文。

本发明的一个方面涉及一种方法，它将与关联已知上下文相关联的原始数据序列顺序地转换为一个不可约的上下文相关文法，并从该文法中恢复原始数据序列。该方法包括以下步骤：从该序列中解析子串；根据解析的子串，产生容许的上下文相关文法；将一个约简规则集合应用于该容许的上下文相关文法，从而产生一个新的不可约上下文相关文法；重复这些步骤，直到将整个序列编码。此外，基于变量和上下文对的约简规则集合表示该不可约上下文相关文法，从而使得这些对表示数据序列的不重叠的重复模式和上下文。

根据本发明的另一个方面，该方法涉及使用自适应上下文相关算术编码，对与来自可数上下文模型集合的已知上下文模型相关联的不可约上下文相关文法进行编码。此外，应用一个基于变量和上下文对的约简规则集合，以表示不可约上下文相关文法，从而使得这些对表示数据序列的不重叠的重复模式和上下文。

根据本发明的另一方面，提供了一种将具有已知上下文模型的数据序列进行编码的方法，包括：将该数据序列变换为不可约

上下文相关文法；将该不可约上下文相关文法转换为其分类形式；从该分类的不可约上下文相关文法中，构造一个产生的序列；并使用自适应上下文相关算术编码，将产生的序列进行编码。

本发明还涉及一种方法，它将与已知上下文模型相关联的原始数据序列顺序地变换为不可约上下文相关文法的序列；并且根据每个不可约上下文相关文法，使用自适应上下文相关算术编码，对该数据序列进行编码。该方法包括以下步骤：从该序列中解析子串；通过利用前一个不可约上下文相关文法的结构和使用自适应上下文相关算术编码，对该子串进行编码；根据该子串、当前上下文和前一个不可约上下文相关文法，产生容许的上下文相关文法；将一个约简规则集合应用于该容许的上下文相关文法，从而产生一个新的不可约上下文相关文法；重复这些步骤，直到该系列所有的符号都被解析和编码。

附图说明

通过参考以下结合附图的描述，可以更好地理解本发明的上述和更多优点：

图 1 是根据本发明实施例的上下文相关文法变换的一系列操作的流程图；

图 2 是根据本发明的实施例，在已知上下文情况下解析输入字符串的子串的一系列操作的流程图；

图 3 是根据本发明的实施例，更新不可约上下文相关文法的一系列操作的流程图；

图 4 是根据本发明的实施例，将约简规则集合应用于顺序压

缩具有已知上下文和前一个已知上下文相关文法的输入字符串的一系列操作的流程图；

图 5、图 6、图 7、图 8 和图 9 是根据本发明的实施例，将约简规则应用于压缩具有已知上下文和前一个已知上下文相关文法的输入字符串的一系列操作的流程图；

图 10 是根据本发明的实施例，将产生的序列进行编码的等级式方法的流程图；

图 11 是使用顺序上下文相关方法的一个实施例对具有已知上下文的输入字符串进行编码的流程图；

图 12 是根据本发明的实施例，使用上下文相关文法变换方法的数据压缩装置的框图。

具体实施方式

本发明的描述中采用了以下符号：A 用于表示一个基数大于或等于 2 的信源字母表；A^*用于表示从 A 中提取的所有有限字符串（包括空字符串 λ）的集合；A^+ 用于表示从 A 中提取的所有具有正长度的有限字符串的集合；符号 | A | 表示 A 的基数，并且对于任意 $x \in A^*$，| x | 表示 x 的长度；对于任意正整数 n，A^n表示 A 中长度为 n 的所有序列的集合；由 A 构成的序列也可以被称为 A 序列；$x \in A^+$用于表示待压缩的序列。信源字母表 A 是跟应用相关的。定义一个可数的符号集合 $S = \{s_0, s_1, s_2, \cdots\}$，它与 $A \cup C$ 不相交。S 中的符号被称为变量；A 中的符号被称为终结符。对于任意 $j \geq 1$，令 $S(j) = \{s_0, s_1, s_2, \cdots, s_{j-1}\}$。

C 用于定义任意上下文集合，集合 C 可以是有限的或无限的。在通常的上下文模型中，对于从字母表 A 提取的任意序列 $x=x_1x_2\cdots x_n$，存在一个来自 C 的上下文序列 $C_1C_2\cdots C_n$。在这个上下文序列中，每一个 C_i 表示 x_i 的上下文。对 x_i 进行编码之前，可通过某种方式，根据所有历史 $x_1x_2\cdots x_{i-1}$ 和 C_1 确定 C_i。此关系的表达式为：

$$C_{i+1}=f(x_1, \cdots, x_i, C_1), \quad i=1, 2, \cdots, \qquad (1)$$

其中 $C_1 \in C$ 是初始的固定上下文，f 是一种映射关系，在这里指“下一个上下文函数”。可以将上述性质进一步扩展到所谓的可数上下文模型，其中式（1）式仍成立，并且上下文集合 C 是可数无限的。

G 用来表示从 $C\times S(j)$ 的子集到 $(S(j)\cup A)^+$ 的上下文相关文法的映射关系，其中，$j\geq 1$。集合 S (j)表示 G 的变量集合，S (j)中的元素被称为 G 变量。于是，G 的域可表示为 $\Omega_G(K_i, j_i)\subset C\times S(j)$，其中 K 是域中不同上下文的个数。在 $\Omega_G(K_i, j_i)$ 中，有一个特殊对 $(C_1, s_0) \in \Omega_G(K_i, j_i)$，它表示初始对，其中 C_1 用于标识初始上下文。

描述变量、其上下文与上下文相关文法之间映射关系的一种方法是：对于每对 $(C^k, s_i) \in \Omega_G(K_i, j_i)$，将关系 $(s_i \mid C^k, G(s_i \mid C^k))$ 定义为 $s_i \mid C^k \rightarrow G(s_i \mid C^k)$，作为产生规则（production rule）。然后，就可以用一个产生规则集合完全描述上下文相关文法 G，即 $\{s_i \mid C^k \rightarrow G(s_i \mid C^k) : (C^k, s_i) \in \Omega_G(K_i, j_i), 0\leq i<j, 0<k<\infty\}$。

在一个示例性的实施例中，参照图 1，根据本发明，方法 100 可用于执行数据压缩。方法 100 的输入为已知的上下文 102 和输入字符串 104。解析器接收上下文 102 和输入字符串 104，并将输入字符串 104

解析为不重叠子串的集合（步骤106）。图2中的方法200进一步示出了解析步骤（步骤202）。解析步骤（步骤202）包括：确定当前子串的上下文（步骤202），以及确定是否能在当前上下文下表示当前子串（步骤204）。如果不能由当前上下文下表示当前子串，则设置当前子串等于当前上下文下的下一个符号（步骤206）。或者，如果当前子串能由当前上下文表示，则设置当前子串等于表示该子串的最长前缀（步骤208）。回到图1，解析器将该子串传输到上下文相关文法更新设备，然后其更新上下文相关文法G（步骤108）。图3中的方法300进一步示出了更新步骤（步骤108）。如果当前解析的字符串的长度大于1（步骤302），则在前一个不可约的上下文相关文法的末尾添加当前变量（步骤304），然后应用如下所述的约简规则集合（步骤308）。或者，如果当前解析的字符串的长度不大于1，则在前一个不可约的上下文相关文法的末尾添加当前符号（步骤306），然后，应用如下所述的约简规则集合（步骤308）。再回到图1，如果输入字符串没有被完全解析（步骤110），则经过一个延时之后重复该方法（步骤112）。然后，上下文相关算术编码器将该文法压缩成二进制码字（步骤114）。

示例1：在此示例中，信源字母表被定义为A＝｛0，1｝。此外，可定义一个可数的上下文模型，使得C＝｛0，1｝，其下一个上下文函数f为$C_{i+1}=x_i$。变量集合为｛s_0，s_1，s_2｝且初始对为（0，s_0）的上下文相关文法G的一个可能例子是：

$$s_0 \mid 0 \rightarrow s_1\ s_1\ 1 s_2\ 0 s_1\ s_2\ s_1\ 0$$

$$s_1 \mid 0 \rightarrow 001$$

$$s_1 \mid 1 \rightarrow 01$$

$$s_2 \mid 1 \rightarrow 1s_1$$

在上面的示例 1 中，可以用变量集合 S（j）和初始对（C_1，s_0）定义 G。通过从 G（$s_0 \mid C_1$）中的左边重复替换第一个变量 s 的 G（s | C）（其中 C 是根据上面的公式（1）由 C_1 及 G（$s_0 \mid C_1$）中直到 s 的前缀进行确定的），将满足下列条件之一：

（1）在有限次数的替换步骤之后，从 A 中获得一个序列；

（2）因为这样获得的每个字符串都包含一个 G 变量的入口，所以替换过程永不结束；

（3）因为 s | C 对应的产生规则未在 G 中定义，所以替换过程不能继续。

继续来看示例 1，如果满足条件（1）－（3），可做以下替换：

$$\begin{aligned}
s_0|0 &\xrightarrow{G} s_1s_11s_20s_1s_2s_10\\
&\xrightarrow{G} 001s_11s_20s_1s_2s_10\\
&\xrightarrow{G} 001011s_20s_1s_2s_10\\
&\xrightarrow{G} 0010111s_10s_1s_2s_10\\
&\xrightarrow{G} 00101110s_1s_2s_10\\
&\xrightarrow{G} 001011101s_2s_10\\
&\xrightarrow{G} 0010111010s_10\\
&\xrightarrow{G} 00101110100s_1s_10\\
&\xrightarrow{G} 001011101001s_10\\
&\xrightarrow{G} 0010111010011010
\end{aligned}$$

在以上示例 1 中，通过执行以下过程，可以从 A 中获得一个序列。从 $s_0 \mid 0 \rightarrow G(s_0 \mid 0)$ 开始，重复应用连续替换过程。九个步骤之后（每次出现符号 $\xrightarrow{G}$ 就表示替换过程的一步），连续替换过程结束。此外，在整个替换过程中，使得 $s_i \mid C^k$ 的产生规则

定义于 G 中的上下文 C^k（$1 \leqslant k \leqslant 2$）下的每一个变量 s_i（$0 \leqslant i < 3$）至少被 G（$s_i \mid C^k$）替换一次。因此，此例中，在上下文 0（或 G）下，s_0 表示序列 x = 0010111010001101010。每一个其他 G 变量表示 x 的一个或更多的子串：当上下文 0 时，s_1 表示 001；当上下文为 1 时，s_1 表示 01；当上下文为 1 时，s_2 表示 101。为了进一步说明此方法的上下文相关特征，应该注意的是：子串“01”在上下文相关文法 G 的范围内出现了两次。然而，在上下文相关的意义上来说，这两个子串并不是重复的字符串，因为每一个子串都有不同的上下文。

如果满足下列条件，一种容许的上下文相关文法 G 就可以被定义为不可约的：

（1）对于任意上下文 C，使得 $(C, s) \in \Omega_G (K, j) \setminus \{(C_1, s_0)\}$ 的每一个 G 变量 s 在上下文 C 下在范围 G 内至少出现两次；

（2）就上下文相关意义来说，在范围 G 内不存在长度大于或等于 2 的不重叠的重复模式；

（3）在相同上下文 C 下，每一个不同的 G 变量 $s \in \{s: (C, s) \in \Omega_G (K, j)\}$ 表示不同的 A 序列。

因此，上述示例 1 中容许的上下文相关文法是不可约的。

文法变换是将任意的数据序列 $x \in A^+$ 变换为表示 x 的容许的上下文相关文法 G_x 的变换。如果对于所有的 $x \in A^+$，G_x 都是不可约的，则该文法变换就是不可约的。从表示 x 的一个容许的上下文相关文法 G 开始，可以重复应用一个约简规则集合，从而生成另一个容许的上下文相关文法 G′，它表示相同的 x 并满足以下性质：

（1）与 x 的长度相比，G 的大小 | G | 应该足够小；

（2）当上下文相同时，不同的 G 变量所表示的 A 字符串是不同的；

（3）在 G 的范围内，G 的变量和终结符的上下文相关频率分布应该使得能够应用有效的上下文相关算术编码。

图 4 示出了应用一个约简规则集合的示例性方法 400，它可以顺序构造一个不可约上下文相关文法的序列，通过此序列，可以递增地恢复原始数据序列。提取出前一个文法约简比特 I（步骤 402）。将 α 定义为前一个不可约的上下文相关文法中的初始变量和上下文对 $s_0 | C_l$ 所对应的产生规则的最后一个符号（步骤 408）。将 C 定义为 α 的上下文（步骤 410）。将 β 定义为中间文法 G'_{i-1} 中的初始变量和上下文对 $s_0 | C_l$ 所对应的产生规则的最后一个符号（步骤 412）。判断上下文 C 时 α β 是否出现在 G'_{i-1} 中的两个不重叠位置（步骤 414）。如果否，则设置当前的不可约文法为 G'_{i-1}（步骤 416），并设置当前的文法比特 I 为 0（步骤 418）。如果 α β 在上下文 C 时出现于 G'_{i-1} 中的两个不重叠位置，则进一步判断前一个文法比特是否等于 1（步骤 420）。

如果前一个文法比特不等于 1，则进一步判断在相同上下文情况下 α β 是否重复自身出现在 G'_{i-1}（$s_0 | C_l$）中两个不重叠位置（步骤 422）。如果重复，则对 G'_{i-1} 应用一次约简规则 2（步骤 428），将当前不可约上下文相关文法设置为所得的文法（步骤 426），并将当前的文法比特 I 设置为 1（步骤 428）。如果不重复，则对 G'_{i-1} 应用一次约简规则 3（步骤 430），将当前不可约上下文相关文法设置为所得的文法（步骤 426），并将当前的文法比特 I 设置为 1（步骤 428）。

回到前一个文法比特等于 1 的情况（步骤 420），进一步判断 α β 是否在相同上下文情况下重复自身出现在 G'_{i-1} （s_0 | C_1）中两个不重叠位置（步骤 432）。如果重复，则对 G'_{i-1} 先应用约简规则 2 一次，然后再应用约简规则 1（步骤 434），将当前不可约的上下文相关文法设置为所得的文法（步骤 426），并将当前的文法比特 I 设置为 1（步骤 428）。如果不重复，则对 G'_{i-1} 应用约简规则 3 一次，然后再应用约简规则 1（步骤 436），将当前不可约的上下文相关文法设置为所得的文法（步骤 426），并将当前的文法比特 I 设置为 1（步骤 428）。

图 5 示出了应用约简规则 1 的方法 500。将 s 定义为容许的上下文相关文法 G 的一个变量，在 G 范围内上下文为 C 时它仅出现一次（步骤 502）。定义产生规则 $s'|\hat{C} \rightarrow \alpha s \beta$，其中，上下文为 C 时 s 出现在右边（步骤 504）。定义对（C，s）所对应的产生规则 s | C→γ（步骤 506）。通过从 G 中删除产生规则 s | C→γ，并将产生规则 $s'|\hat{C} \rightarrow \alpha s \beta$ 替换为产生规则 $s'|\hat{C} \rightarrow \alpha \gamma \beta$，来将 G 约简为容许的上下文相关文法 G′，所得的上下文相关文法 G′ 与 G 表示相同的序列（步骤 508）。

图 6 示出了应用约简规则 2 的方法 600。将 G 定义为容许的上下文相关文法，其具有形式为 $s|C \rightarrow \alpha_1 \beta \alpha_2 \beta \alpha_3$ 的产生规则，其中 β 的长度至少为 2，并且 β 在这两个位置的上下文等于相同的上下文 C′（步骤 602）。定义新的产生规则 $s'|C' \rightarrow \beta$，其中（C′，s′）是 G 域中没有的新的上下文和变量的对（步骤 604）。通过将 G 的产生规则 $s|C \rightarrow \alpha_1 \beta \alpha_2 \beta \alpha_3$ 替换为 $s|C \rightarrow \alpha_1 s' \alpha_2 s' \alpha_3$，并添加产生规则 $s'|C' \rightarrow \beta$，来将 G 约简为新的上下文相关文法 G′，所得的文法 G′ 包括新的上下

文和变量的对（C′，s′），并且，并且与 G 表示相同的序列 x（步骤 608）。

图 7 示出了应用约简规则 3 的方法 700。定义 G 为容许的上下文相关文法，其具有形式分别为 $s|C\rightarrow\alpha_1\beta\alpha_2$ 和 $s'|C'\rightarrow\alpha_3\beta\alpha_4$ 的两个不同的产生规则，其中 β 的长度至少为 2，这两个位置的 β 的上下文等于相同的上下文 C″，α_1 或 α_2 不为空，并且 α_3 或 α_4 不为空（步骤 702）。定义一个新的产生规则 $s''|C''\rightarrow\beta$，其中（C″，s″）是不在 G 的域中的新的上下文和变量的对（步骤 704）。通过用 $s|C\rightarrow\alpha_1 s''\alpha_2$ 替换产生规则 $s|C\rightarrow\alpha_1\beta\alpha_2$，并用 s′ | C′ → α_3 s″ α_4 替换产生规则 $s'|C'\rightarrow\alpha_3\beta\alpha_4$，并附加新的规则 $s''|C''\rightarrow\beta$，来将 G 约简成新的上下文相关文法 G′（步骤 706）。

图 8 示出了应用约简规则 4 的方法 800。定义 G 为容许的上下文相关文法，其具有形式为 $s|C\rightarrow\alpha_1\beta\alpha_2$ 和 $s'|C'\rightarrow\beta$ 的两个产生规则，其中 β 的长度至少为 2，$s|C\rightarrow\alpha_1\beta\alpha_2$ 中 β 的上下文也等于 C′，并且 α_1 和 α_2 中之一不为空（步骤 802）。通过用新的产生规则 $s|C\rightarrow\alpha_1 s'\alpha_2$ 替换所述产生规则 $s|C\rightarrow\alpha_1\beta\alpha_2$，将 G 约简为上下文相关文法 G′（步骤 804）。

图 9 示出了应用约简规则 5 的方法 900。定义 G 为容许的上下文相关文法，其中，给定上下文 C 情况下两个变量 s 和 s′ 表示由 G 所表示的 A 序列的相同子串（步骤 902）。在 G 范围内，只要 s′ 的上下文等于 C，就用 s 替换每一个 s′ 的出现，并删除（C，s′）所对应的产生规则，从而将 G 约简成新的上下文相关文法 G′（步骤 904）。判断文法 G′ 是否为容许的上下文相关文法（步骤 906）。

文法 G′ 可能不是容许的，因为在 G′ 的连续替换过程中可能没有包括一些 G′ 变量。如果这是真的，则通过删除连续约简过程中不包括的变量 G′ 所对应的所有产生规则，来进一步把 G′ 约简为上下文相关文法 G″（步骤 908），否则，解析下一字符串（步骤 910）。

本发明的一个方面包括一个贪婪的上下文相关文法变换，在一个实施例中，此文法变换将压缩 A 中的数据序列 $x=x_1, x_2, \cdots x_n$。不可约的上下文相关文法变换被描述为贪婪的，它把序列 x 解析成不重叠的子串 $\{x_1 x_2 \cdots x_{n_2}, \ldots, x_{n_{i-1}+1} \cdots x_{n_i}\}$ 并顺序为每个 $x_1 \cdots x_{ni}$ 建立不可约的上下文相关文法，其中 $1 \leqslant i \leqslant t$，$n_1=1$ 且 $n_i=n$。在这种情况下，第一个子串 x_1 和相应的不可约上下文相关文法 G_1 只包含一个产生规则 $s_0 \mid C_1 \rightarrow x_1$，其中 C_1 是初始上下文。假设 $\{x_1 x_2 \cdots x_{n_2}, \ldots, x_{n_{i-1}+1} \cdots x_{n_i}\}$ 已经被解析且 $x_1 \cdots x_{ni}$ 相应的不可约上下文相关文法 G_i 已经建立。假设 G_1 的变量集合是 $S(j_i)=\{s_0, s_1, \cdots s_{k-1}\}$，其中 $j_1=1$，G_i 的域是 $\Omega_G(K_i, j_i)$。令 C_{ni+1} 是由可数上下文模型生成的 x_{ni+1} 的上下文。如果 $x_{n_i+1} \cdots x_n$ 存在最长前缀，其在上下文 Cn_{i+1} 时能由 $s_j \in \{s:(C_{n_i+1}, s) \in \Omega_G(K_i, j_i)\}$ 表示，则下一子串 $x_{n_i+1} \cdots x_{n_{i+1}}$ 就是该最长前缀。否则，$n_{i+1}=n_i+1$ 时，$x_{n_i+1} \cdots x_{n_{i+1}} = x_{n_i+1}$。如果 $n_{i+1}-n_i>1$ 且在上下文 Cn_{i+1} 时 $x_{n_i+1} \cdots x_{n_{i+1}}$ 由 s_j 表示，那么在 G_i（$s_0 \mid C_1$）右端添加 s_j；否则，在 G_i（$s_0 \mid C_1$）右端添加符号 x_{ni+1}。所得的上下文相关文法是容许的，但未必是不可约的。应用约简规则 1 至 5，将该上下文相关文法约简成不可约的上下文相关文法 G_{i+1}。约简之后，G_{i+1} 表示

序列 $x_1 \cdots x_{n_{i+1}}$。重复该过程，直到整个序列 x 被处理得到表示序列 x 的最终不可约上下文相关文法 G_t。

示例 2：下面的示例说明上述方法。定义 A＝｛0，1｝且 x＝0001001111000011111001111。使用可数上下文模型 C＝｛0，1｝以及下一个上下文函数 f 为 $C_{i+1}=x_i$ 对 x 应用上述不可约的上下文相关文法变换。前三个解析的子串是 0，0 和 0。假设初始上下文 $C_1=0$，相应的不可约上下文相关文法 G_1，G_2 和 G_3 分别用｛s_0｜0→0｝，｛s_0｜0→00｝和｛s_0｜0→000｝表示。由于 $j_3=1$，第四个解析的子串为 $x_4=1$。在 G_3（s_0｜0）的末尾添加符号 1，得到由｛s_0｜0→0001｝表示的容许的上下文相关文法 G_3'。G_3' 本身是不可约的，因此没有约简规则能被应用，并且 G_4 等于 G_3'。

继续上面的示例 2，第五和第六个解析短语分别是 $x_5=0$ 和 $x_6=0$。G_6 表示为｛s_0｜0→000100｝。第七个解析短语是 $x_7=1$。在 G_6（s_0｜0）的末尾添加符号 1，所得容许的上下文相关文法 G_6' 表示为 s_0｜0→0001001。G_7' 不是不可约的，因为在 G_6' 范围内同样的上下文 0 时有不重叠的重复模式 01。应用约简规则 2 一次，我们可得到不可约的上下文相关文法 G_7，表示为：

$$s_0 \mid 0 \rightarrow 00 s_1 0 s_1$$

$$s_1 \mid 0 \rightarrow 01$$

根据 G_7，接下来的五个解析短语分别是 $x_8=1$，$x_9=1$，$x_{10}=1$，$x_{11}=0$ 和 $x_{12}=0$。在 G_7（s_0｜0）的末端添加解析的短语，依次产生不可约的上下文相关文法 G_9 至 G_{12}。第十三个解析的短语是 $x_{13}=0$。在 G_{12}（s_0

| 0）的末尾添加符号 0 并应用一次约简规则 2，得到以下不可约的上下文相关文法 G_{13}：

$$s_0 \mid 0 \rightarrow s_2 s_1 0 s_1\ 1110 s_2$$

$$s_1 \mid 0 \rightarrow 01$$

$$s_2 \mid 0 \rightarrow 00$$

当前上下文为 0 的情况下，剩下未解析的字符串为 01111001111。在此例中，由于上下文为 0 时由 s_1 表示的来自 A 的序列匹配上下文同样为 0 情况下 x 的剩余部分的前缀，所以，下一个解析短语是 $x_{14}x_{15}$ ＝01。在 G_{13}（$s_0 \mid 0$）的末端添加符号 s_1，得到容许的上下文相关文法 G_{13}'：

$$s_0 \mid 0 \rightarrow s_2 s_1 0 s_1\ 1110 s_2\ s_1$$

$$s_1 \mid 0 \rightarrow 01$$

$$s_2 \mid 0 \rightarrow 00$$

因为上下文为 0 时 $s_2 s_1$ 重复，所以 G_{13}' 不是不可约的。应用约简规则 2，得到不可约的上下文相关文法 G_{13}''：

$$s_0 \mid 0 \rightarrow s_3 0 s_1\ 1110 s_3$$

$$s_1 \mid 0 \rightarrow 01$$

$$s_2 \mid 0 \rightarrow 00$$

$$s_3 \mid 0 \rightarrow s_2 s_1$$

在上面的文法中，在 G_{13}'' 的范围内上下文为 0 时，变量 s_2 只出现了一次，因此应用一次约简规则 1，得到不可约的上下文相关文法 G_{14}：

$$s_0 \mid 0 \rightarrow s_2 0 s_1\ 1110 s_2$$

$$s_1 \mid 0 \rightarrow 01$$

$$s_2 \mid 0 \rightarrow 00\ s_1$$

继续该方法，产生最终不可约的上下文相关文法 G_{22}：

$$s_0 \mid 0 \rightarrow s_2 0 s_1\ s_1\ s_2\ s_1\ s_1 s_2$$

$$s_1 \mid 0 \rightarrow 01$$

$$s_1 \mid 1 \rightarrow s_2 0$$

$$s_2 \mid 0 \rightarrow 00\ s_1$$

$$s_2 \mid 1 \rightarrow 111$$

这样，不可约的上下文相关文法变换将 x 解析成 {0，0，0，1，0，0，1，1，1，1，0，0，0，0，1，1，1，1，0，0，1，1，1，1}。

本发明的另一方面（表示为 1000）如图 10 所示，在这里表示为“等级式上下文相关方法”。此方法包括一个贪婪的不可约上下文相关变换，然后对最终不可约上下文相关文法 G_t 进行上下文相关算术编码。数据序列首先被变换为不可约的上下文相关文法（步骤 1002），然后再被转换成其分类形式（步骤 1004）。从该分类文法构造产生的序列（步骤 1006），并使用具有动态字母表的自适应上下文相关算术编码对该产生的序列进行编码（步骤 1008）。

作为一个示例，令 χ 为待编码的 A 序列，令 G_1 为表示由我们的不可约上下文相关文法变换提供的 χ 的最终不可约上下文相关文法。在等级式上下文相关方法中，具有动态字母表的上下文相关算术编码用于对 G_1（或其等价形式）进行编码。接收到二进制

码字之后，解码器恢复 G_1（或其等价形式），然后执行如示例 1 所示的连续替换过程，从而得到 χ。

一个实施例首先示出了容许的上下文相关文法 G 如何推导出序列 $\chi \in A^n$ 的一部分，然后进一步描述了分类的上下文相关文法的概念。令 Ω_G 表示 $\Omega_G(K, j)$ 动态变化的子集；开始时，Ω_G 为空。令 $u^{(0)} = G(s_0 \mid C_1)$；其中 $u^{(0)}$ 是来自（S（j）∪A）的序列。如果 j＝1，或等价为如果 $u^{(0)}$ 中没有变量，那么，$u^{(0)}$ 本身指由 G 所推导出的部分序列。否则，执行下面的步骤：

步骤 1：设置 i＝0；

步骤 2：对于 i≥0，从左向右读取 $u^{(i)}$，用 G（s｜C）替换不在 Ω_G 中的第一对（C，s），其中 C 是由上下文模型确定的 s 的上下文，所得的序列用 $u^{(i+1)}$ 表示；

步骤 3：将对（C，s）插入 Ω_G，从而更新 Ω_G；

步骤 4：对于 i＝0，1，…，｜$\Omega_G(K, j)$｜－1，重复步骤 2 和步骤 3，以使每一对（C，s）∈ $\Omega_G(K, j) \setminus \{(C_1, s_0)\}$ 正好被 G（s｜C）替换一次。

最终的序列 $u^{(|\Omega_G(K,j)|-1)}$ 被称为由 G 导出的部分序列。如果在上面的过程中，动态集合 Ω_G 以这样的方式扩增：对每个出现在 $\Omega_G(K, j)$ 中的上下文 C，对于任何有意义的 i>0，（C，s_i）出现于（C，s_{i+1}）之前，那么，上下文相关文法 G 被称为分类形式。

在示例 2 所示的 G_{22} 中，｜$\Omega_G(K, j)$｜＝5。五个序列 $u^{(0)}, \cdots, u^{(4)}$ 为：

$u^{(0)}=s_20s_1s_1s_2s_1s_1s_2$，

$u^{(1)}=00s_10s_1s_1s_2s_1s_1s_2$，

$u^{(2)}=00010s_1s_1s_2s_1s_1s_2$，

$u^{(3)}=00010\ s_1s_20s_2s_1s_1s_2$，

$u^{(4)}=00010s_11110s_2s_1s_1s_2$。

部分序列 $u^{(4)}$ 将序列 x 解析为 0，0，0，1，0，01，1，1，1，0，0001，1110，01，111。将以上子串连接，重建原始序列 x。动态集合 Ω_G 按照（0，s_2），（0，s_1）（1，s_1）和（1，s_2）的顺序扩增。因而，在此例中，G_{22} 不是分类形式。

现在，可以使用等级式上下文相关方法，对序列进行编码，其如例 2 所示。如上所述，最终不可约的上下文相关文法 G_{22} 不是分类形式。通过下式，将 G_{22} 转换成分类形式 G_{22}^s：

$s_0|0\rightarrow s_10s_2s_1s_1s_1s_2s_2$

$s_1|0\rightarrow 00s_2$

$s_1|1\rightarrow s_20$

$s_2|0\rightarrow 01$

$s_2|1\rightarrow 111.$

在例 2 中，为了从 G_{22} 获得 G_{22}^s，可将 G_{22} 中上下文 0 时的变量 s_1、s_2 分别重命名为 G_{22}^s 中上下文 0 时的 s_2、s_1。需要注意的是，G_{22} 和 G_{22}^s 表示相同的序列 x。现在，将 G_{22}^s 而非 G_{22} 进行编码和传输。对 G_{22}^s 应用上面步骤 1－4 中所述的过程，得到由 G_{22}^s 导出的部分序列：

$u^{(4)}=00010S_21110S_1S_1S_2S_2$

令 α 是 $G_i(s_0\mid C_l)$的最后一个符号，$\hat{C}$ 是 α 的上下文。为了解决 G_{22}^s

无法从部分序列恢复的事实，将该过程稍做修改。具体为，在步骤 2 中，用[G(s | C)]替换不在Ω_G中的一对的第一个变量 s 及其上下文 C，并且相应的序列被表示为$^{(i)}, i=0,\ldots,|\Omega_G(K,j)|-1$。将修改的过程应用于$G_{22}^s$，得到来自（S（j）∪A）∪｛[,]｝的以下序列：

$u^{(4)}$ = [00 [01]] 0S_2 [[lll]0]$S_1S_1S_2S_2$

该序列被称为由 G_{22} 或其分类形式 G_{22}^s 产生的序列。由于引入了符号[和]，对关联的可数上下文模型的下一个上下文函数 f 稍做修改，使得对于每一个 C∈C，C=f（C，[）和 C=f（C，]），即：[和]仅仅接替（relay）上下文。因此，可以从由 G_{22}^s 所产生的序列$\tilde{u}^{(4)}$中恢复 G_{22}^s 和序列 x。$\tilde{u}^{(4)}$中合法的括号对[和]的个数等于除（C_1，s_0）之外 G_{22}^s 的域中的变量和上下文对的个数；上下文 C 时第 i 个合法的括号对[和]内的内容产生对应于 s_i | C 的产生规则。因此，可以从$\tilde{u}^{(4)}$推断出每一个变量 s 在给定的上下文时表示什么，因此使用分类的 CDG G_{22}^s 而非原始的 CDG G_{22}。为了压缩 G_{22}^s 或序列 x，使用具有动态字母表的上下文相关算术编码对由 G_{22}^s 生成的序列进行编码。需要注意的是：编码器和解码器都知道关联的可数上下文模型。每一对$(C,\beta)\in(C\times S)\cup(C\times(A\cup\{[,]\}))$与一个计数器 c（C，β）相关联，如果$(C,\beta)\in C\times(A\cup\{[,]\})$，设置 c（C，β）为 1，否则设置 c（C，β）为 0。按照下列步骤，对根据 G_{22}^s 所生成的序列中的每一个符号 β 进行编码并更新相关的计数器：

步骤 1： 在上下文 C（初始时 C=C_1）的情况下，使用概率

$c(C,\beta)/\sum c(C,\alpha)$对β进行编码，其中求和符号$\sum_{\alpha}$针对使c（C，α）≠0的所有符号$\alpha \in S \cup A \cup \{[,]\}$进行求和；

步骤2： 将c（C，β）加1；

步骤3： 在由G生成的序列中，如果β=]且对应于β的合法的括号对是上下文C′时的第i个括号对（其中，C′是此括号对中[的上下文],则将计数器c（C′，s_i）由0增加到1；

步骤4： 确定产生的序列中下一短语的下一个上下文C。

重复以上过程直到将整个产生的序列编码。对于上面所示的产生的序列$\tilde{u}^{(4)}$，算术编码处理中所使用的概率积是：

$$\frac{1}{4}\frac{1}{5}\frac{2}{6}\frac{2}{7}\frac{3}{8}\frac{1}{9}\frac{1}{4}\frac{2}{5}\frac{1}{6}\frac{1}{12}\frac{1}{7}\frac{2}{8}\frac{1}{9}\frac{2}{10}\frac{3}{11}\frac{3}{12}\frac{2}{14}\frac{1}{13}\frac{1}{14}\frac{1}{16}\frac{2}{15}\frac{1}{17}$$

总之，要对最终的不可约CDG G_i编码，首先要把G_i转换成它的分类形式G_i^s，然后根据G_i构造生成的序列，最后使用具有动态字母表的上下文相关算术编码，对生成的序列进行编码。

图11还示出了本发明的另一方面，即方法1100，在这里表示为“顺序上下文相关方法”，其包括：使用具有动态字母表的上下文相关算术编码对解析短语序列$x_1, x_2 \cdots x_{n_2}, \ldots, x_{n_{k-1}+1} \cdots x_{n_k}$进行编码。已知的上下文102和输入字符串104作为输入，并被解析为不重叠的字符串（步骤106）。使用前一个不可约上下文相关文法的结构并使用具有动态字母表的自适应上下文相关算术编码，对每一个解析的子串进行编码（步骤1101）。生成和更新上下文相关文法（步骤108），直到该字符串被完全解析（步骤110）。

与等级式上下文相关方法一样，各对$(C,\beta)\in(C\times S)\cup(C\times A)$都与计数器 c（C，β）相关联，如果$(C,\beta)\in C\times A$，此计数器初始值被设置为 1，否则被设置为 0。每个短语在给定的上下文条件下被编码。假设$x_1,x_2\cdots x_{n_2},\ldots,x_{n_{i-1}+1}\cdots x_{n_i}$已被编码且所有相应的计数器已被更新。进一步假设 G_i 是$x_1\cdots x_{n_i}$的相应不可约上下文相关文法且 G_i 的变量集合等于$S(j_1)=\{s_0,s_1,\ldots,s_{j_i-1}\}$。使用贪婪的上下文相关文法变换解析$x_{n_i+1}\cdots x_{n_{i+1}}$并用$\beta\in\{s_1,\ldots,s_{j_i-1}\}\cup A$表示解析的字符串。按照下列步骤对 β 进行编码，并更新相关计数器：

步骤 1： 确定 β 的上下文 C；

步骤 2：在给定上下文 C 的条件下使用概率$c(C,\beta)/\sum_{\alpha}c(C,\alpha)$对 β 进行编码，其中求和符号$\sum_{\alpha}$针对使 c（C，α）≠0 的所有符号$\alpha\in S(j_i)\cup A$进行求和；

步骤 3： 将 c（C，β）加 1；

步骤 4： 根据添加的 G_i 得到 G_{i+1}；

步骤 5： 如果 G_{i+1} 引入了新对（C′，S）所对应的新产生规则，则将计数器 c（C′，S）从 0 增加到 1。

对所有的解析短语重复上面的过程，直到整个序列被处理并编码。

本发明的另外一方面，在这里被称作“改进的顺序上下文相关方法”，它包括：使用一阶算术编码对序列$\{I(i)\}_{i=1}^{t}$进行编码，并使用序列$\{I(i)\}_{i=1}^{t}$和$\{G_i\}_{i=1}^{t}$的结构改进解析短语$x_1,x_2\cdots x_{n2},\cdots,x_{n_{i-1}+1}\cdots x_{n_i}$的序列的编码。除了上面所定义的计数器$c(C,\gamma),(C,\gamma)\in C\times(S\cup A)$外，还定义了计数器 $c_I(0,0)$、$c_I(0,1)$、$c_I(1,0)$、$c_I(1,1)$和$\hat{C}(C,\gamma)$。计数器 $c_I(0,0)$、$c_I(0,1)$、

$c_I(1,0)$和 $c_I(1,1)$用于对序列$\{I(i)\}_{i=1}^{\infty}$进行编码，并且初始时都被设置为等于 1。在 I（i）＝0 且 I（i＋1）＝1 时，计数器$\hat{C}(C,\gamma)$对第（i＋1）个解析的短语进行编码；在 I（i＋1）＝0 时，则用计数器 c（C，γ）来编码。与 c（C，γ）一样，如果（C，γ）∈C×A，则初始时设置$\hat{C}(C,\gamma)$＝1，否则设置为 0。与顺序上下文相关方法中一样，开始的三个解析短语 x_1，x_2和 x_3被编码。而且，因为 I（1），I（2）和 I（3）全是 0，没有必要对它们进行编码。从第四个短语开始，首先对 I（i＋1）进行编码，然后将其与 G_i的结构一起用作附加上下文信息，对第（i+1）个解析短语进行编码。

作为进一步的说明，假设$x_1, x_2 \cdots x_{n_2}, \ldots, x_{n_{i-1}+1} \cdots x_{n_i}$和 I(4)……I(i)已经被编码并且所有相应的计数器已经被更新。进一步假设 G_i是$x_1 \cdots x_{n_i}$相应的不可约上下文相关文法，且 G_i的变量集合等于 $S(j_i)=\{s_0, s_1, \cdots, s_{j-1}\}$。$G_i$的域是$\Omega_G(K_i, j_i)$，令 j（c，i）为与$\Omega_G(K_i, j_i)$中任意 C 相关联的变量（除去 s_0）的数目。此外，令 α 是 $G_i(s_0|C_1)$的最后一个符号，$\hat{C}$是 α 的上下文。使用贪婪的上下文相关文法变换解析$x_{n_i+1} \cdots x_{n_{i+1}}$，并用$\beta \in \{s_1, \ldots, s_{j_i-1}\} \cup A$表示解析的字符串。使用下面的步骤对 I（i＋1）和 β 进行编码并更新相关计数器：

步骤 1：使用概率$c_I(I(i), I(i+1))/(c_I(I(i),0)+c_I(I(i),1))$对 I（i＋1）进行编码；

步骤 2：将 c_I（I（i），I（i＋1））加 1；

步骤 3：确定 β 的上下文$\hat{C}$；

步骤 4：如果 I（i＋1）＝0，使用概率$c(\hat{C},\beta)/\sum_{\gamma} c(\hat{C},\gamma)$对上下文$\hat{C}$的

β 进行编码，其中求和符号 $\sum_{\gamma}$ 对所有使得 $c(\hat{C},\gamma)\neq 0$ 的符号 $\gamma\in S(j_i)\cup A-L_2(\alpha|\hat{C})$ 求和，然后将 $c(\hat{C},\beta)$ 加 1。否则，如果 I（i）=0 且 I（i+1）=1，则使用概率 $c(\tilde{C},\beta)/\sum_{\gamma\in L_1(\alpha|\hat{C})}c(\tilde{C},\gamma)$ 对上下文 $\hat{C}$ 的 β 进行编码，然后将 $c(\hat{C},\beta)$ 加 1。相反，如果 I（i）=1 且 I（i+1）=1，则不发送信息，因为 $L_1(\alpha|\hat{C})$ 仅包含一个元素，解码器知道 β 是什么。这里所说的列表 $L_1(\alpha|\hat{C})$ 包含具有以下性质的所有符号 $\gamma\in S(j_i)\cup A$：

a）在上下文 $\hat{C}$ 时，模式 αγ 出现于 G_i 范围内；

b）上下文 $\hat{C}$ 时的模式 αγ 不是 $G_i(s_0 | C_l)$ 的两个符号；

c）G_i 中没有变量 s_j 使得 $G_i(s_j | \hat{C})$ 等于 αγ；以及

d）所述列表 $L_2(\alpha|\hat{C})$ 包括具有 a）和 b）性质的所有符号 $\gamma\in S(j_i)\cup A$。

步骤 5：从附加的 G_i 获得 G_{i+1}，相应地更新所有列表 $L_1(\gamma|C)$ 和 $L_2(\gamma|C)$；

步骤 6：如果 $j_{(\hat{C},j+1)}>j_{(\hat{C},j)}$，这说明 G_{i+1} 引入了对应于 $s_{j_{(\hat{C},j)}+1}|\hat{C}$ 的新的产生规则，因而将 $c(\hat{C},s_{j_{(\hat{C},j)}+1})$ 和 $\hat{c}(\hat{C},s_{j_{(\hat{C},j)}+1})$ 都加 1。

重复此过程直到整个序列 x 被处理和编码。

示例 3：对示例 2 中序列被解析为{0，0，0，1，0，0，1，1，1，1，0，0，0，0，1，1，1，1，0，0，1，1，1，1}的序列 x，应用上述改进的顺序上下文相关方法，进行压缩，其中，相应的序列{ I（i）}是{000000100000110111001 1}。用于对序列{ I（i）}$^{22}_{i=4}$ 进行编码的概

率积是 $\frac{1\,2\,3\,1\,1\,4\,5\,6\,7\,2\,1\,2\,3\,2\,3\,3\,8\,4\,4}{2\,3\,4\,5\,2\,6\,7\,8\,9\,10\,3\,4\,11\,5\,6\,7\,12\,13\,8}$，用于对解析短语进行编码的概率积是 $\frac{1\,2\,3\,1\,1\,4\,1\,1\,2\,3\,2\,5\,1\,4\,1\,1\,5\,2}{2\,3\,4\,1\,2\,6\,2\,1\,4\,5\,2\,7\,2\,4\,2\,9\,8\,3}$。

对于那些将本发明提供为软件程序的实施例，该程序可以用多种高级语言中的任意一种进行编写，如 FORTRAN、PASCAL、JAVA、C、C++、C#或 BASIC。此外，该软件可以用汇编语言实现，汇编语言专用于驻留在目标计算机上的微处理器，例如，如果该软件被配置为运行于 IBM PC 或其它类似 PC 上，该软件可用 Intel 80x86 汇编语言来实现。该软件可以被收录在制造的产品中，包括、但不限于软盘、硬盘、光盘、磁带、PROM、EPROM、EEPROM、现场可编程门阵列（FPGA）或 CD-ROM。

在这些实施例中，该软件可以被安装运行在任何个人类型的计算机或工作站上，例如 PC 或 PC 兼容机、Apple Macintosh、Sun 工作站等等。总之，可以使用任何设备，只要它能够实现这里描述的所有功能和能力。对于本发明来说，计算机或工作站的具体类型不是重点。

图 12 示出了一个实施例，其中本发明为执行上述方法的设备 1200。设备 1200 包括：解析器 1206，上下文相关文法更新设备 1210 和上下文相关文法编码器 1214。解析器 1206 接收输入字符串 1204 和已知的上下文 1202 作为输入，并把输入字符串 1204 解析成一系列不重叠的子串 1208。解析器将子串 1208 发送给上下文相关文法更新设备 1210，上下文相关文法更新设备 1210 产生更新的上下文相关文法 G。上下文相关文法更新设备 1210 将更新的文法 G 发送到上下文相关文法

编码器 1214，然后，上下文相关文法编码器 1214 将文法 G 编码成压缩的二进制码字 1216。

例如，设备 1200 可以是一个或多个专用集成电路（ASIC）、现场可编程门阵列（FPGA）、电可擦除可编程只读存储器（EEPROM）、可编程只读存储器（PROM）、可编程逻辑电路（PLD）或只读存储设备（ROM）。在一些实施例中，可以用一个或多个微处理器实现设备 1200，例如，加利福尼亚州圣克拉拉的英特尔（Intel）公司制造的奔腾系列芯片，或者，伊利诺斯州绍姆堡（Schaumburg）的摩托罗拉（Motorola）公司制造的 PowerPC 系列芯片。

上面结合具体的优选实施例对本发明进行了详细显示和描述，但是，本领域技术人员应该理解，在不脱离所附权利要求书定义的本发明精神和保护范围内，可以对本发明进行各种形式和细节上的改变。

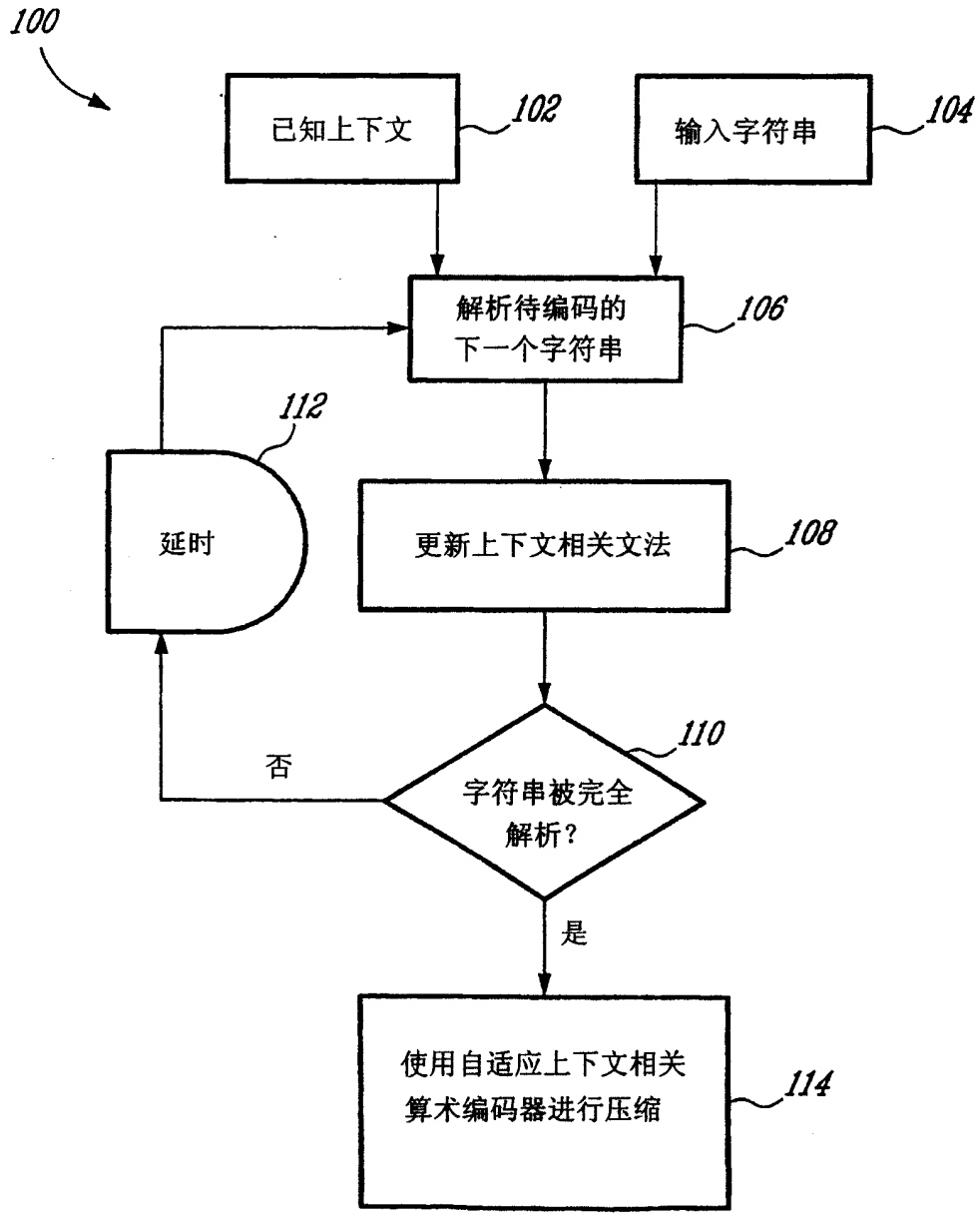

100
已知上下文
102
输入字符串
104
解析待编码的
下一个字符串
106
112
延时
更新上下文相关文法
108
110
否
字符串被完全
解析?
是
使用自适应上下文相关
算术编码器进行压缩
114

图1

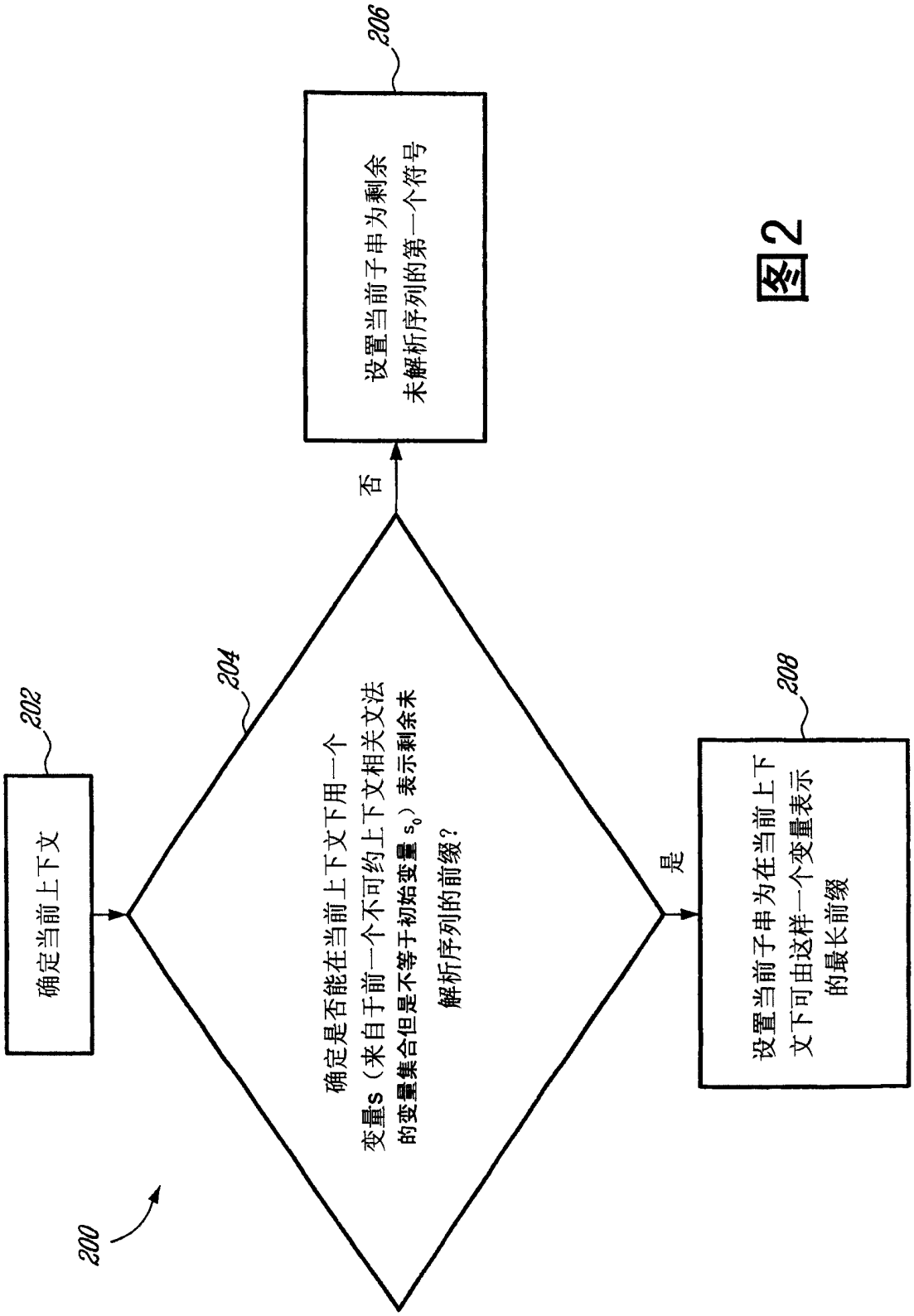

200
202
确定当前上下文
204
确定是否能在当前上下文下用一个变量s（来自于前一个不可约上下文相关文法的变量集合但是不等于初始变量 s_0）表示剩余未解析序列的前缀?
否
206
设置当前子串为剩余未解析序列的第一个符号
是
208
设置当前子串为在当前上下文下可由这样一个变量表示的最长前缀

图2

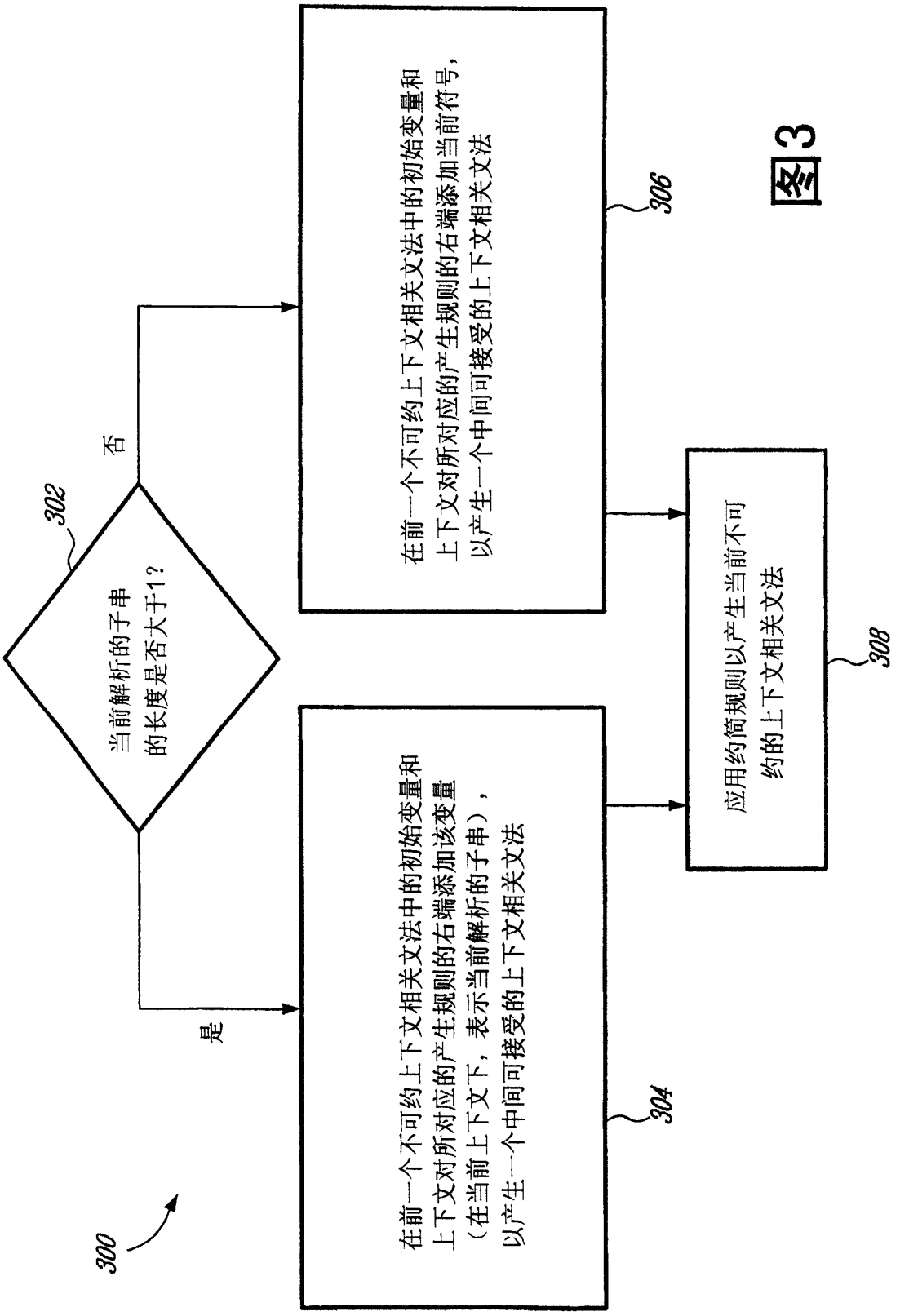

300
302
当前解析的子串
的长度是否大于1?
否
是
在前一个不可约上下文相关文法中的初始变量和
上下文对所对应的产生规则的右端添加该变量
（在当前上下文下，表示当前解析的子串），
以产生一个中间可接受的上下文相关文法
304
在前一个不可约上下文相关文法中的初始变量和
上下文对所对应的产生规则的右端添加当前符号，
以产生一个中间可接受的上下文相关文法
306
应用约简规则以产生当前不可
约的上下文相关文法
308

图3

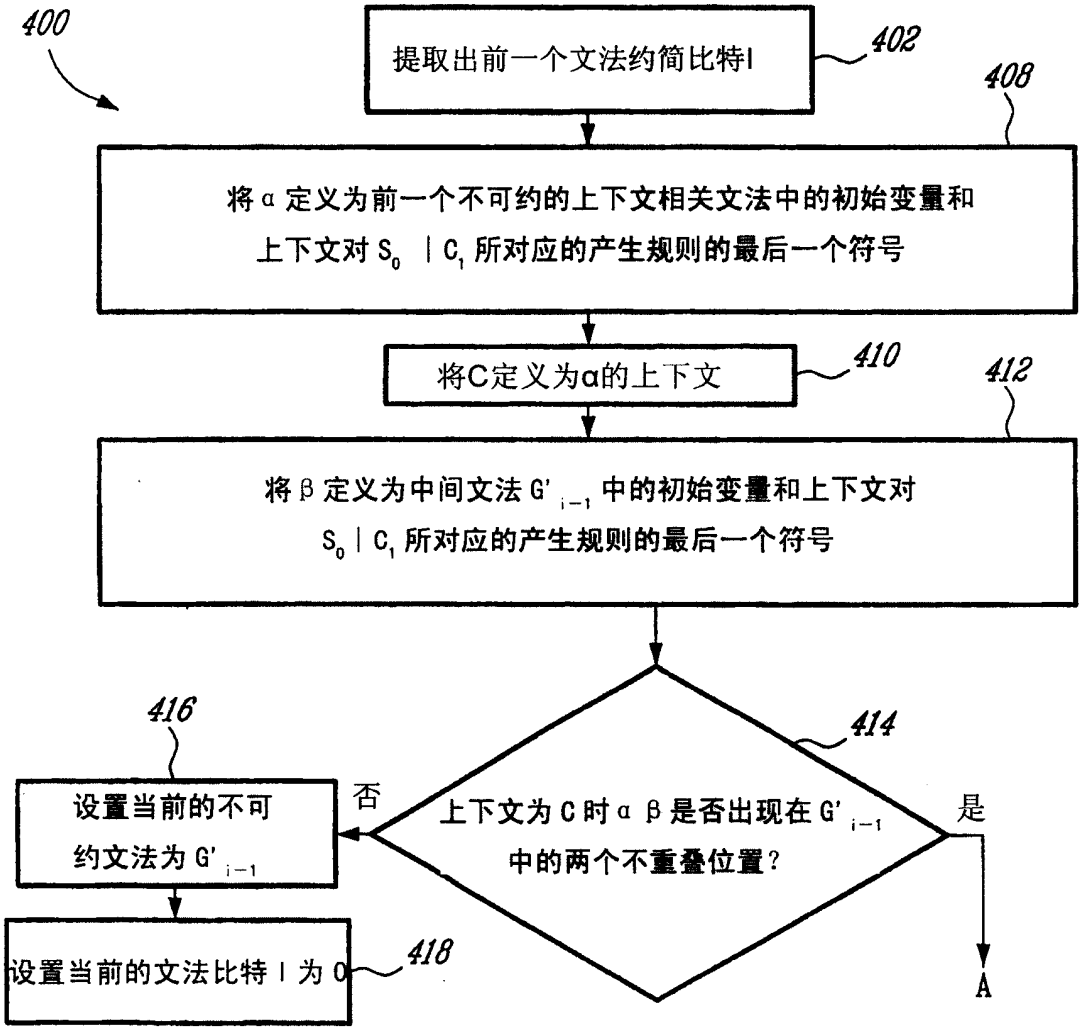

400
提取出前一个文法约简比特I
402
408
将α定义为前一个不可约的上下文相关文法中的初始变量和
上下文对 $S_0 \mid C_1$ 所对应的产生规则的最后一个符号
将C定义为α的上下文
410
412
将β定义为中间文法 G'_{i-1} 中的初始变量和上下文对
$S_0 \mid C_1$ 所对应的产生规则的最后一个符号
414
上下文为C时α β是否出现在 G'_{i-1}
中的两个不重叠位置?
否
是
416
设置当前的不可
约文法为 G'_{i-1}
设置当前的文法比特 I 为 0
418
A

图4

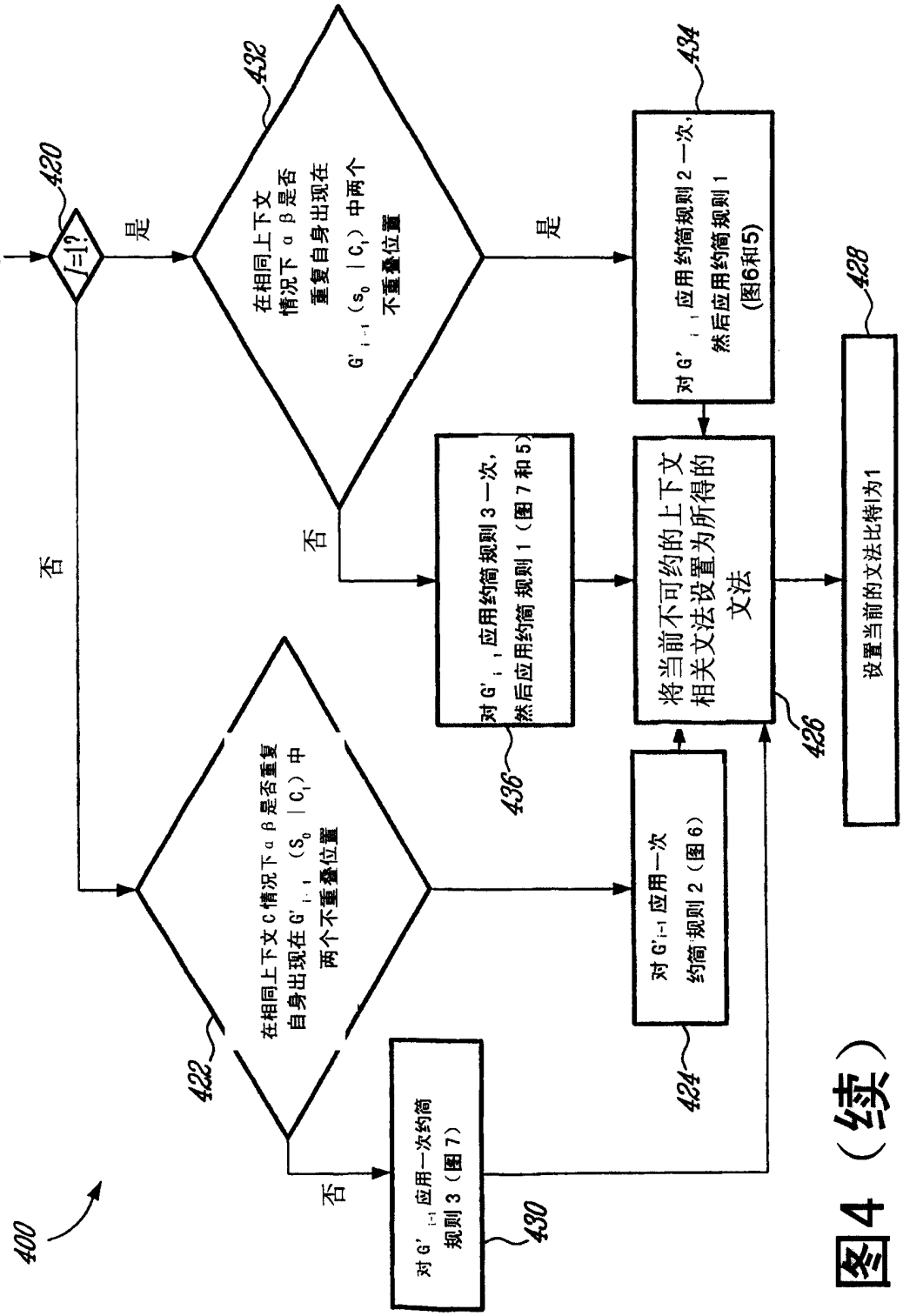

400
A
420
I=1?
是
否
432
在相同上下文情况下αβ是否重复自身出现在$G'_{i-1}(s_0 \mid c_1)$中两个不重叠位置
是
否
434
对G'_{i-1}应用约简规则2一次，然后应用约简规则1(图6和5)
436
对G'_{i-1}应用约简规则3一次，然后应用约简规则1（图7和5）
422
在相同上下文C情况下αβ是否重复自身出现在$G'_{i-1}(S_0 \mid c_1)$中两个不重叠位置
否
是
424
对G'_{i-1}应用一次约简规则2（图6）
430
对G'_{i-1}应用一次约简规则3（图7）
426
将当前不可约的上下文相关文法设置为所得的文法
428
设置当前的文法比特I为1

图4（续）

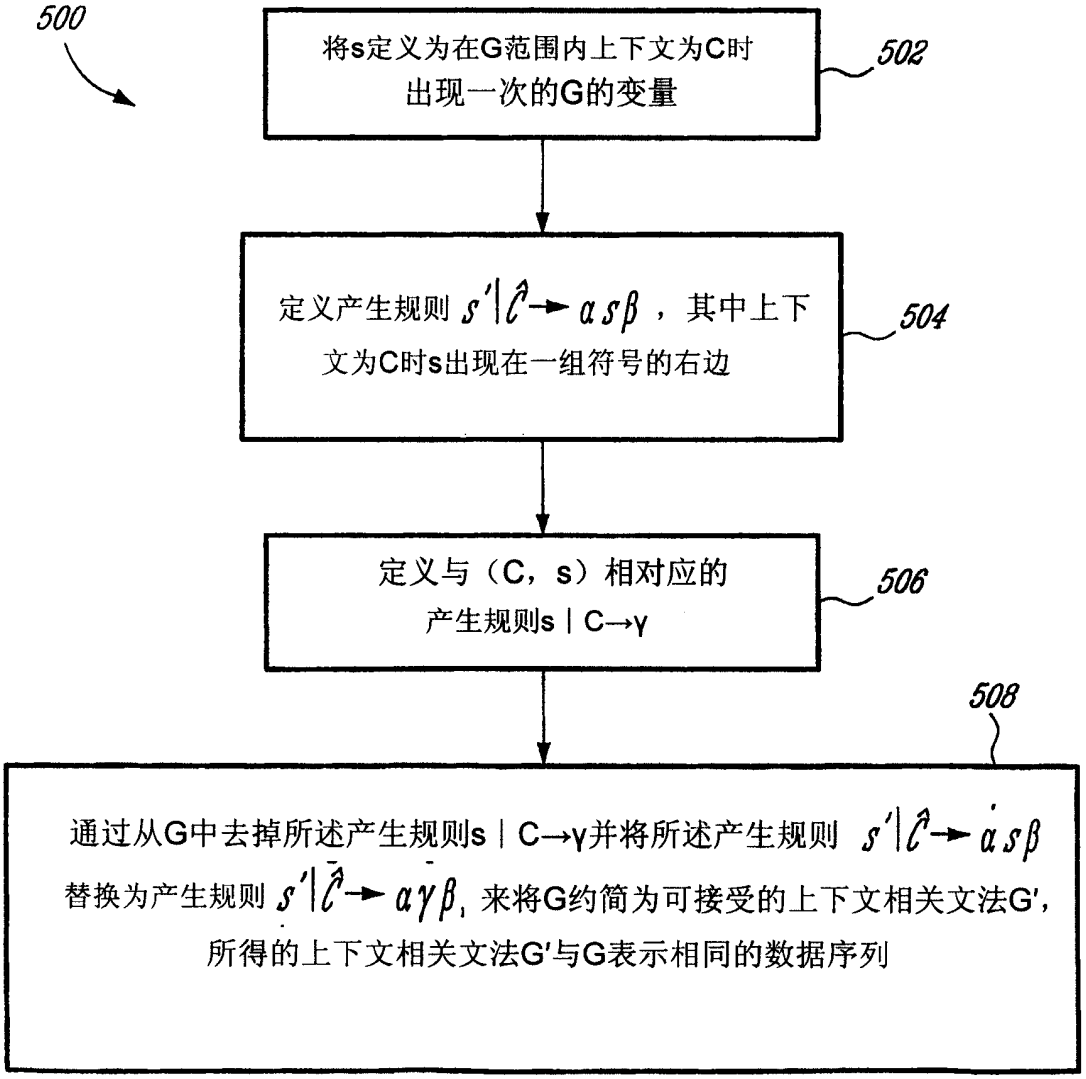

500
将s定义为在G范围内上下文为C时出现一次的G的变量
502
定义产生规则 $s'|\hat{C}\rightarrow\alpha s\beta$ ，其中上下文为C时s出现在一组符号的右边
504
定义与（C，s）相对应的产生规则s | C→γ
506
508
通过从G中去掉所述产生规则s | C→γ并将所述产生规则 $s'|\hat{C}\rightarrow\dot{\alpha}s\beta$ 替换为产生规则 $s'|\hat{C}\rightarrow\alpha\gamma\beta$，来将G约简为可接受的上下文相关文法G'，所得的上下文相关文法G'与G表示相同的数据序列

图5

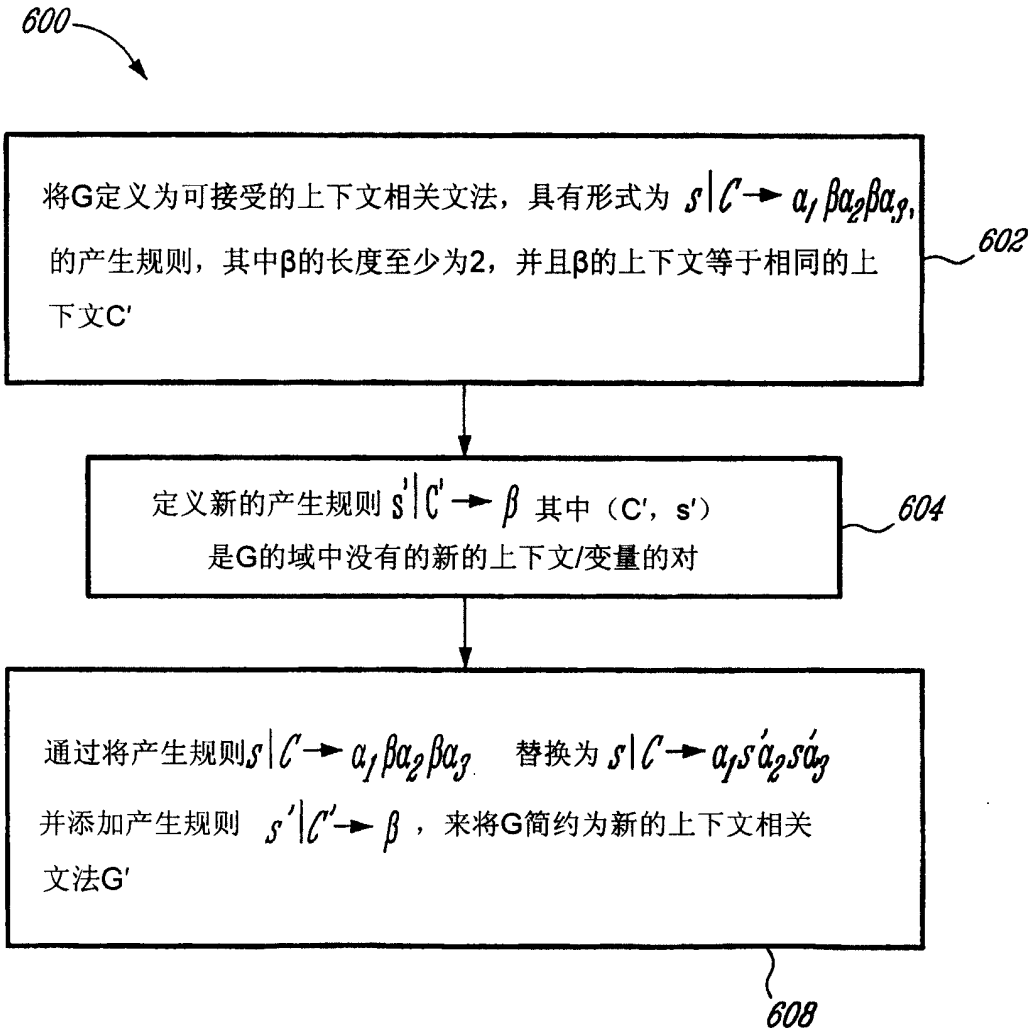

600
将G定义为可接受的上下文相关文法，具有形式为 $s|C \rightarrow a_1 \beta a_2 \beta a_3$，的产生规则，其中β的长度至少为2，并且β的上下文等于相同的上下文C'
602
定义新的产生规则 $s'|C' \rightarrow \beta$ 其中（C'，s'）是G的域中没有的新的上下文/变量的对
604
通过将产生规则 $s|C \rightarrow a_1 \beta a_2 \beta a_3$ 替换为 $s|C \rightarrow a_1 s' a_2 s' a_3$ 并添加产生规则 $s'|C' \rightarrow \beta$，来将G简约为新的上下文相关文法G'
608

图6

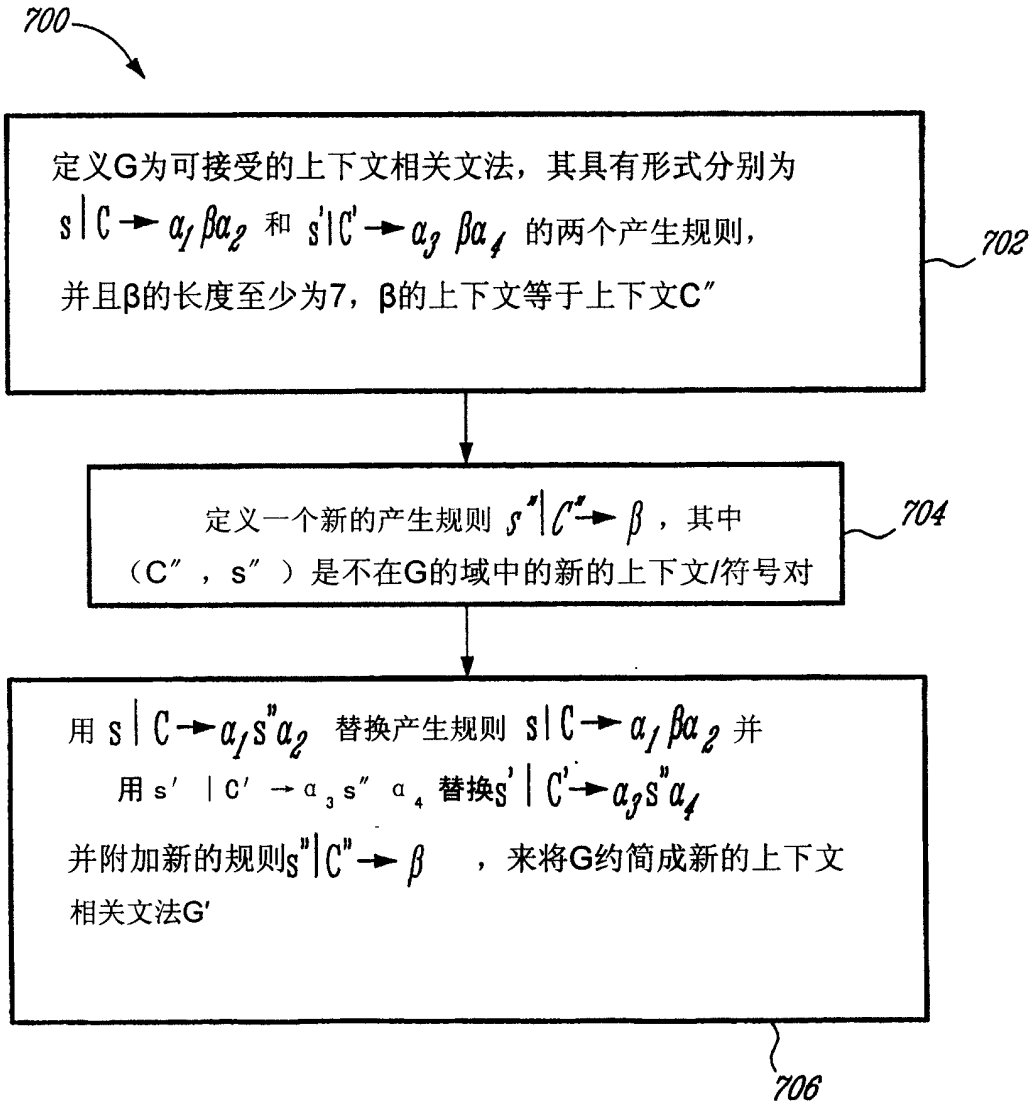

700
定义G为可接受的上下文相关文法，其具有形式分别为
$s|C \rightarrow \alpha_1 \beta \alpha_2$ 和 $s'|C' \rightarrow \alpha_3 \beta \alpha_4$ 的两个产生规则，
并且β的长度至少为7，β的上下文等于上下文C″
702
定义一个新的产生规则 $s''|C'' \rightarrow \beta$ ，其中
（C″，s″）是不在G的域中的新的上下文/符号对
704
用 $s|C \rightarrow \alpha_1 s'' \alpha_2$ 替换产生规则 $s|C \rightarrow \alpha_1 \beta \alpha_2$ 并
用 $s'|C' \rightarrow \alpha_3 s'' \alpha_4$ 替换 $s'|C' \rightarrow \alpha_3 s'' \alpha_4$
并附加新的规则 $s''|C'' \rightarrow \beta$ ，来将G约简成新的上下文
相关文法G′
706

图7

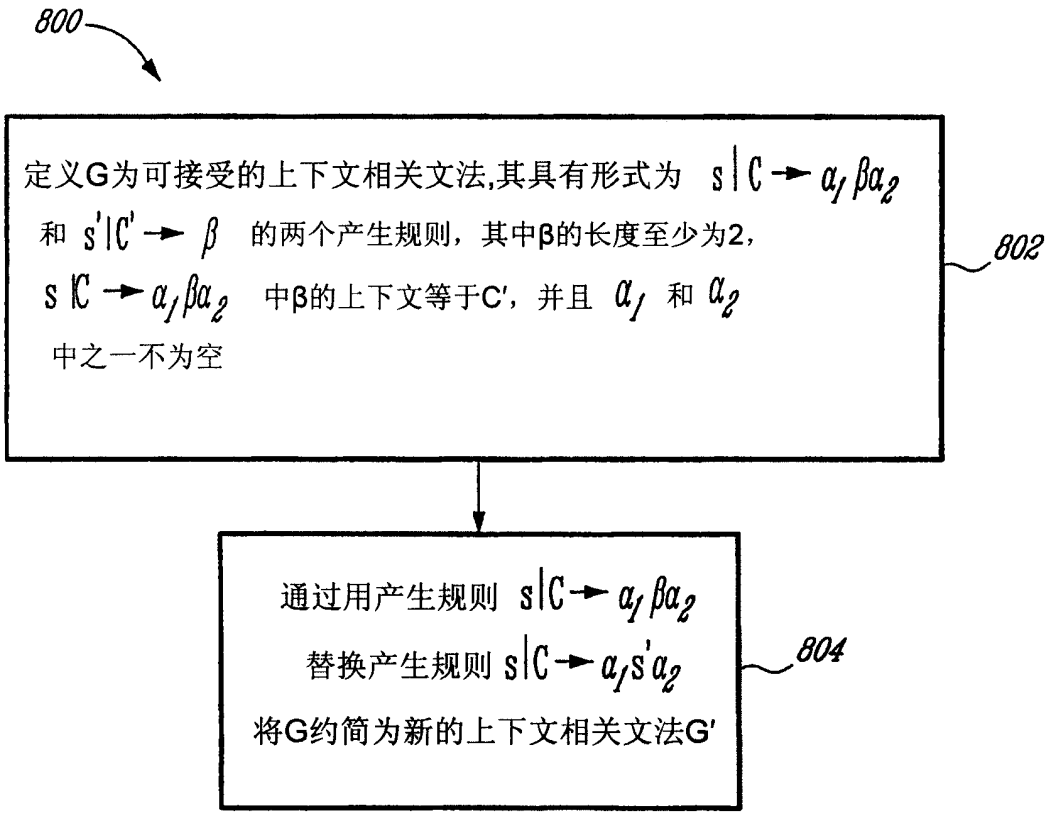

800
定义G为可接受的上下文相关文法,其具有形式为 $s|C \rightarrow \alpha_1 \beta \alpha_2$
和 $s'|C' \rightarrow \beta$ 的两个产生规则，其中β的长度至少为2，
$s|C \rightarrow \alpha_1 \beta \alpha_2$ 中β的上下文等于C'，并且 α_1 和 α_2
中之一不为空
802
通过用产生规则 $s|C \rightarrow \alpha_1 \beta \alpha_2$
替换产生规则 $s|C \rightarrow \alpha_1 s' \alpha_2$
将G约简为新的上下文相关文法G'
804

图8

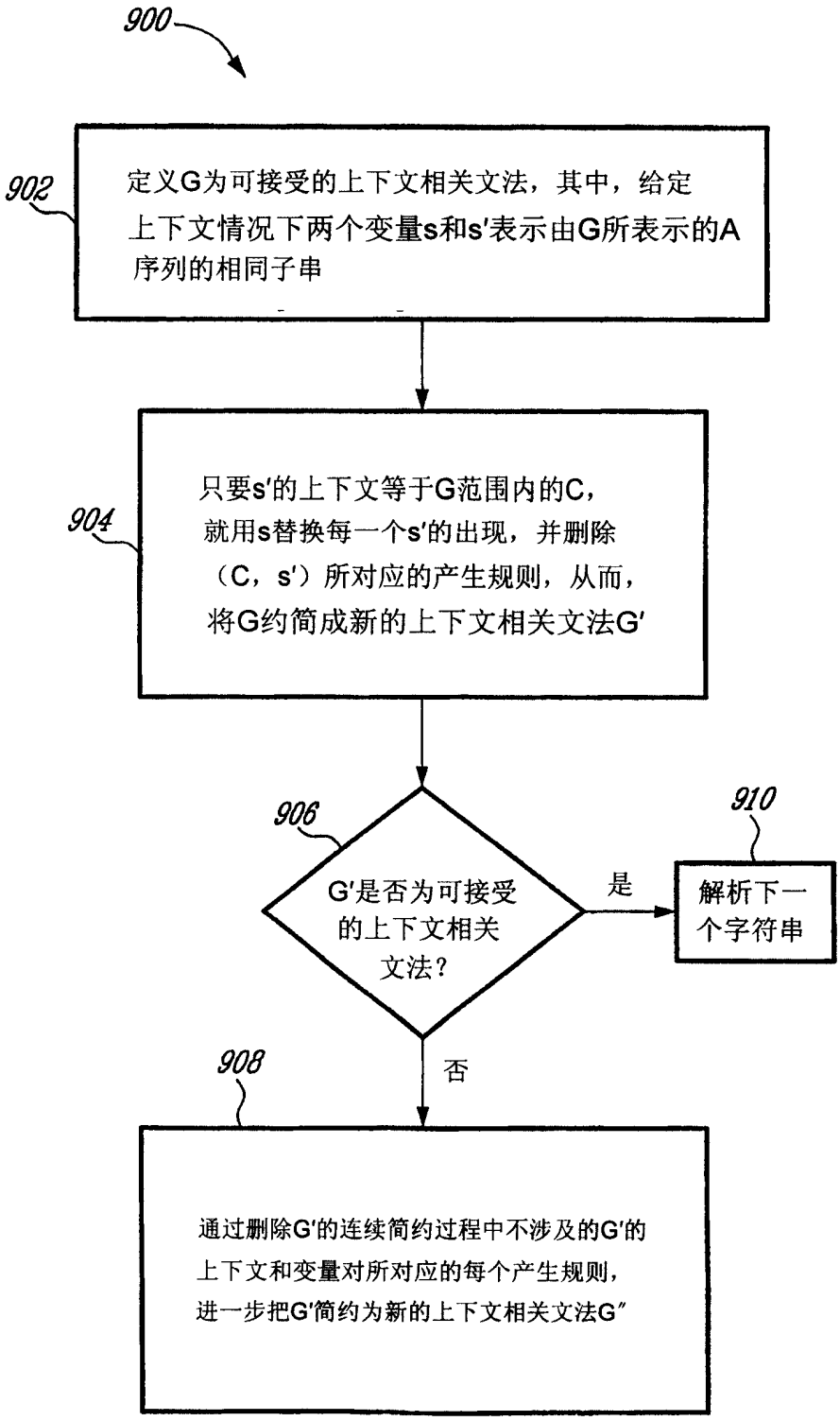

900
902
定义G为可接受的上下文相关文法，其中，给定上下文情况下两个变量s和s′表示由G所表示的A序列的相同子串
904
只要s′的上下文等于G范围内的C，就用s替换每一个s′的出现，并删除（C，s′）所对应的产生规则，从而，将G约简成新的上下文相关文法G′
906
G′是否为可接受的上下文相关文法？
是
910
解析下一个字符串
否
908
通过删除G′的连续简约过程中不涉及的G′的上下文和变量对所对应的每个产生规则，进一步把G′简约为新的上下文相关文法G″

图9

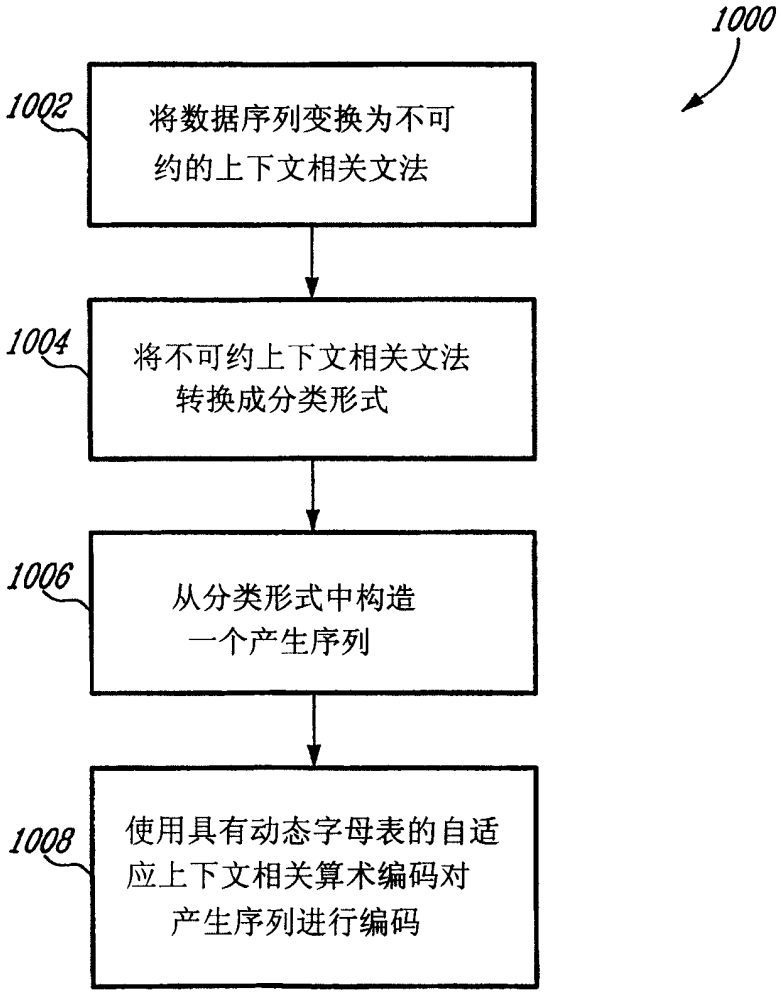

1000
1002
将数据序列变换为不可约的上下文相关文法
1004
将不可约上下文相关文法转换成分类形式
1006
从分类形式中构造一个产生序列
1008
使用具有动态字母表的自适应上下文相关算术编码对产生序列进行编码

图10

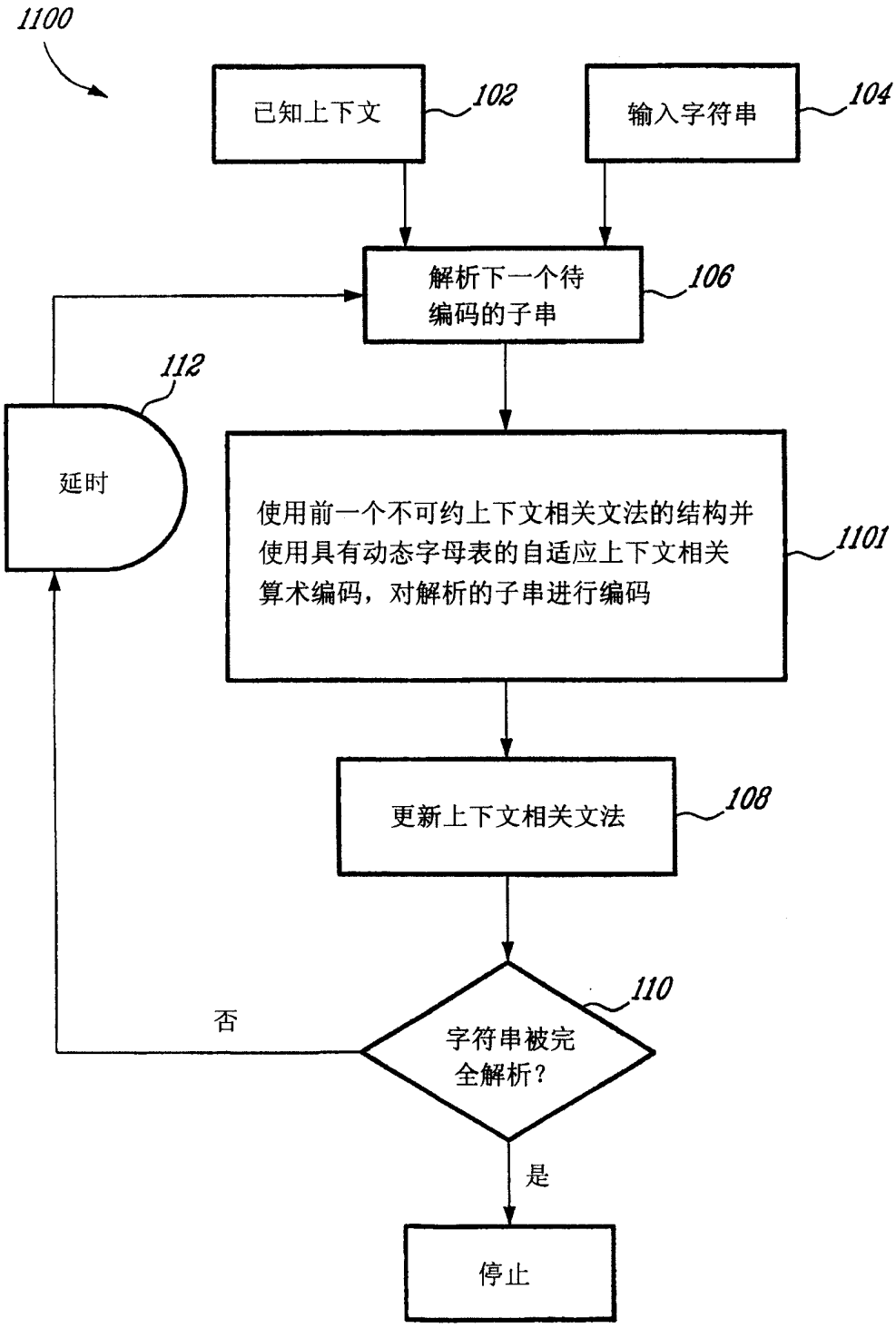

1100
已知上下文
102
输入字符串
104
解析下一个待编码的子串
106
112
延时
使用前一个不可约上下文相关文法的结构并使用具有动态字母表的自适应上下文相关算术编码，对解析的子串进行编码
1101
更新上下文相关文法
108
110
字符串被完全解析？
否
是
停止

图11

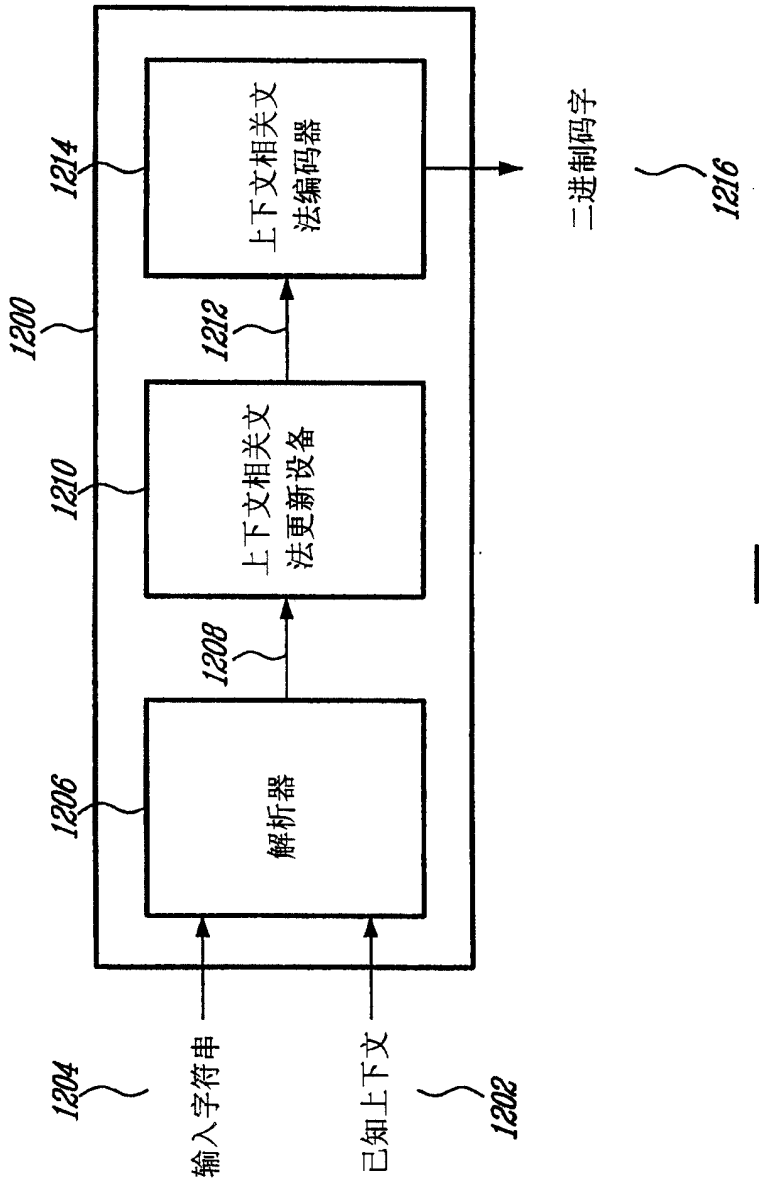

1200
1202
已知上下文
1204
输入字符串
1206
解析器
1208
1210
上下文相关文法更新设备
1212
1214
上下文相关文法编码器
1216
二进制码字

图12