arXiv:2603.21645v1 [cs.FL] 23 Mar 2026

斐波那契自动序列的线性子序列复杂度

Delaram Moradi*滑铁卢大学计算机科学学院 滑铁卢，安大略省 N2L 3G1 加拿大
delaram.moradi@uwaterloo.ca

Narad Rampersad†温尼伯大学数学与统计系 Portage 大道 515 号，温尼伯，马尼托巴省 R3B 2E9 加拿大
n.rampersad@uwinnipeg.ca

Jeffrey Shallit*滑铁卢大学计算机科学学院 滑铁卢，安大略省 N2L 3G1 加拿大
shallit@uwaterloo.ca

2026年3月24日

摘要

我们构造输入为斐波那契表示（也称为 Zeckendorf 表示）的自动机，用以识别一些基本的算术关系，并研究其状态数。我们还考察斐波那契自动序列的一些基本运算，并讨论其状态复杂度。此外，作为研究结果的推论，我们改进了 Bosma 和 Don 最近一篇论文中的一个界限。我们还讨论了使用对 B¨uchi 算术的合理解释来实际构造一些用于识别关系的自动机时的状态复杂度和运行时间复杂度。

*本研究获得加拿大自然科学与工程研究委员会资助，项目编号为 RGPIN-2024-03725。
†本研究获得加拿大自然科学与工程研究委员会资助，项目编号为 RGPIN-2019-04111 和 RGPIN-2025-04076。

1 引言

本文讨论输入采用斐波那契表示（也称为 Zeckendorf 表示）时，特定算术关系（例如加法和乘法）的状态复杂度。我们还研究斐波那契自动序列的移位和线性子序列的状态复杂度，以及使用对 B¨uchi 算术的一种解释来构造自动机的计算复杂度。我们在先前的一篇论文中 [12] 研究了 k-自动序列的相同问题，但斐波那契表示这一情形带来了许多新的、不同的挑战。

第2节介绍本文后续部分所需的背景知识。第3节中，我们构造了状态数为 $O(\log\ c)$ 的自动机，用于识别 $Y = X + c$ 和 $Y = X - c$，其中 X、Y 为输入，$c \geq 0$ 为常数。我们还构造了状态数为 $O(n^2)$ 的自动机，用于识别关系 $Y = nX + c$，其中 $n \geq 1$ 是自然数，$0 \leq c < n$，且数值 X 和 Y 为输入。第4节中，我们证明：若 $(h(i))_{i\geq 0}$ 是由一个含 m 个状态的自动机生成的固定斐波那契自动序列，则移位子序列 $(h(i+c))_{i\geq 0}$ 可由一个含 $O(m^2c^2)$ 个状态的确定性有限自动机输出机生成；Thue-Morse 序列的线性子序列 $(t(i+c))_{i\geq 0}$ 可由一个含 $O(c)$ 个状态的确定性有限自动机输出机生成；对于 $n \geq 1$ 和 $0 \leq c < n$，线性子序列 $(h(ni+c))_{i\geq 0}$ 可由一个含 $O(m^2n^4)$ 个状态的确定性有限自动机输出机生成。因此，我们改进了 Bosma 和 Don 最近一篇论文中关于斐波那契词线性子序列的态射大小的上界 [3]。

一个使用 B¨uchi 算术解释的系统示例是 `Walnut`，这是一个自由软件包，可用于对自动序列进行计算，并评估用一阶逻辑表述的相关命题。有关 `Walnut`的更多信息，请参阅例如 [13, 18]。在第 5节中，我们将展示如何使用 B$h(ni+c))_{i\geq 0}$生成的自动机（对于固定的 $n \geq 1$, ） $0 \leq c < n$ 可通过 B¨uchi 算术解释在多项式时间内构造出来。

本文中，我们偶尔会提到 $O(\log i)$，其中 i 可以为 0 或 1。在这些情况下，我们采用通常的约定，即对于 $i \in \{0, 1\}$，$O(\log\ i)\ = O(1)$。

2 背景

2.1 有限自动机

本文采用常见的确定性有限自动机模型，相关讨论见[8], ，并介绍了该模型的一些变体。确定性有限自动机（DFA）由有限状态集 Q、输入字母表 Σ、初始状态 q_0、接受状态集 A 以及转移函数 δ 组成：$Q \times \Sigma \to Q$。该转移函数根据自动机的当前状态和当前输入字母，指定下一个状态。我们按照通常方式将 δ 的定义域扩展为 $Q \times \Sigma^*$。输入被完全处理后，DFA 是否进入接受状态决定了该输入是否被接受。通常，转移函数 δ 被视为从 $Q \times \Sigma$ 到 Q的全函数；但在本文中，为了

妥善处理死状态，我们允许转移函数为部分函数。若无法通过一条（可以为空的）路径从 DFA 的状态 q 到达任何接受状态，则称该状态为死状态。显然，语言 L 的最小确定有限自动机至多有一个死状态。按照惯例，本文不计入或展示死状态。若一个状态可从初始状态到达，则称其为可达的；若可从该状态到达某个接受状态，则称其为共可达的。

我们还需要介绍另一种模型：非确定有限自动机（NFA）。它与确定有限自动机类似，但转移函数现在允许在同一个输入字母下转移到 0 个或多个状态，并根据是否存在至少一条路径，在完全处理完输入后进入接受状态来定义接受条件。

我们还使用带输出的确定有限自动机（DFAO）这一概念。它与确定有限自动机类似，不同之处在于输出与状态相关联。与某个输入对应的输出，是到达的最后一个状态所关联的输出。如果其输入采用斐波那契表示（见第 2.2 节），则称该 DFAO 为斐波那契带输出确定有限自动机。斐波那契带输出确定有限自动机$M = (Q, \Sigma, \Delta, \delta, q_0, \tau)$按以下方式生成一个斐波那契自动序列 $(h(i))_{i\geq 0}$：输入是斐波那契计数系统中 i 的最高位优先表示 x，输出 $h(i)$ 为 $\tau(\delta(q_0, x))$。我们假设，这种带输出的确定有限自动机在输入包含无效斐波那契表示时，不会到达任何状态。

对于给定的自动序列 $(h(i))_{i\geq 0}$，存在唯一与之关联的内部序列$(h'(i))_{i\geq 0}$，其取值属于 Q。该序列由最小确定性有限自动机输出机 $(Q, \Sigma, \Delta, \delta, q_0, \tau)$ 定义：将 τ 替换为 Q 上的恒等映射。这里，“唯一”指的是除字母重命名外唯一。因此，$(h(i))_{i\geq 0}$ 是 $(h'(i))_{i\geq 0}$ 在编码 $q \to \tau(q)$ 下的像。

形式语言的状态复杂度是识别该语言的最小确定有限自动机中的状态数；自动序列的状态复杂度则是生成该序列的最小确定性有限自动机输出机中的状态数。

我们需要一个关于由斐波那契自动序列生成的无限词$\mathbf{x}$ 的子词复杂度（也称“因子复杂度”）的引理。$\mathbf{x}$ 的子词复杂度是一个函数 $\rho_{\mathbf{x}}(n)$，用于统计$\mathbf{x}$中出现的长度为 n 的不同子词的数量。

引理 1. 我们有 $\rho_{\mathbf{x}}(n) = O(n)$。设 $\mathbf{x}$ 是一个由具有 *msd* 优先输入的 m 状态确定性有限自动机输出机生成的斐波那契自动序列。那么对所有 $n \geq 1$，都有 $\rho_{\mathbf{x}}(n) = O(nm^2)$。

证明. 参见 [1, 定理 10.3.1]。该证明针对底数-k 给出，但同样适用于斐波那契自动序列。□

最后，我们还需要介绍一个不太熟悉的概念：无歧义输出有限自动机（UFAO）。它结合了无歧义有限自动机（UFA）与确定性有限自动机输出机的概念。UFAO 是一种非确定性机器，对每个输入至多产生一个输出。与某个输入关联的输出，就是到达的唯一接受状态所关联的输出。通常的子集构造可以将 UFAO 转换为确定性有限自动机输出机，但状态数可能因此发生指数级膨胀。

DFA 和 NFA 模型在标准参考著作中有详细介绍，例如 Hopcroft 和 Ullman [8, 第 2 章和第 3]章。关于 DFAO，可参见例如 [1,第 4]章。UFA 除了在许多其他地方有所讨论外，也见 Jir´asek 等人 [9]的著作。

在本文中，我们经常将自动机视为以自然数 i 作为输入进行处理，并用某种计数系统表示这些数。输入字母表为 $\Sigma_k = \{0, 1, \ldots, k-1\}$，其中 $k \geq 2$是某个整数，它表示 i的表示形式中的各位数字。

这引出了一个微妙的问题：我们该如何处理输入中的前导零？例如，如果自动机对输入 00101 和输入 101 给出不同的结果，该怎么办？为避免这个问题，我们设计的自动机满足以下条件：首先，无论在输入前面添加多少个前导零，我们讨论的自动机都表现相同（接受/拒绝或给出相同输出）。其次，我们假设自动机在初始状态 q_0 遇到输入 0 时有一条自循环边（如果有多个输入，则在输入 $[0, 0, \ldots, 0]$ 时有一条自循环边）。当处理长度不同的多个整数输入时，这些约定至关重要。

2.2 斐波那契表示

我们令 $\mathbb{N} = \{0, 1, 2, \ldots\}$。斐波那契数按通常方式定义为 $F_0 = 0$、 $F_1 = 1$ 和$F_n = F_{n-1} + F_{n-2}$，其中 $n \geq 0$。

斐波那契表示由 Lekkerkerker [10] 和 Zeckendorf [21], 引入，尽管在 Ostrowski [15] 的一篇论文中，可以找到一种更早且一般得多的形式。 $n \in \mathbb{N}$ 的斐波那契表示将 n 写成不同斐波那契数 F_i之和，其中 $i \geq 2$；也就是说， $n = \sum_{2 \leq i \leq t} e_i F_i$ 对于 $e_i \in \{0, 1\}$成立。为确保表示的唯一性，不能使用两个连续的斐波那契数；也就是说，在有效表示中，对于 $1 \leq i \leq t$, 我们有$e_i \cdot e_{i+1} = 0$。表示可以与其系数对应的二进制字 $e_t \cdots e_2$相关联；注意，表示的最高有效位写在左侧。我们需要一种方法，将任意二进制字 $x = e_t \cdots e_2$ 转换为它所表示的整数。为此，我们记作 $[x] = \sum_{1 \leq i \leq t} e_i F_i$。例如，我们有 $[01001] = 6$。整数 i 的规范斐波那契表示是不含前导零的字 x，使得 $[x] = i$。为此，我们使用记号（i）。如果 x 是 z 的前缀，且 z 不含 11，那么我们称字 z 是字 x 的有效右扩展。

有时我们需要表示自然数对。为此，我们使用字母表 $\Sigma_2 \times \Sigma_2$。这可能需要用前导零填充较短的表示。例如，字 $[0, 1][0, 0][1, 1][0, 0][1, 0]$ 表示数对 $[4, 11]$。如果 w和 x 是字母表 Σ上的等长字，那么 $w \times x$ 表示元素$y \in (\Sigma^2)^*$，其第一分量为 w，第二分量为 x。这些概念也可类似地扩展，以表示由两个以上自然数组成的元组。

除非另有说明，本文余下部分均假设自动机的输入采用斐波那契表示，并以最高有效位在前的格式给出。

斐波那契词 $\mathbf{f} = f_0 f_1 f_2 \cdots = 01001010 \cdots$是一个无限二进制字，可以用多种等价方式定义 [2]。对我们而言，最简单的定义是 f_i 是 i的斐波那契表示的最后一位。它也被定义为态射 $0 \to 01, 1 \to 0$ 的不动点。它可以由一个包含 2 个状态的确定性有限自动机输出机生成，如图 1所示。

现在我们陈述一些关于斐波那契表示的事实。第一个很容易，第二个由 Reble [16], 提及，第三个则是斐波那契表示

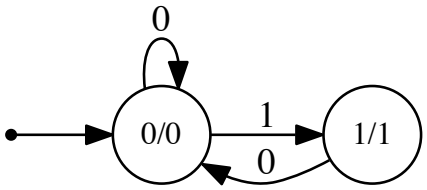


图1：斐波那契词的自动机 f。

由贪心算法给出这一点的推论。设 $\alpha = (1+\sqrt{5})/2$ 和 $\overline{\beta} = (1-\sqrt{5})/2$为多项式 $X^2 - X + 1$的零点。

引理 2。

(a) 设 $x \in \Sigma_2^*$。则 $[x00] = [x] + [x0]$。*(b)* 设 x 为有效斐波那契表示。则 $-\beta^2 < [x0] - \alpha[x] < -\beta$。

(c) 设 x, y 为有效斐波那契表示。当且仅当 $x < y$ 按字典序排列时，$[x] < [y]$。

接下来我们陈述一个引理，将在第 4节中使用。

引理3。 假设 x 是一个词， $a \in \{0,1\}$ 且 xa 是有效斐波那契表示。则$[x] = \lfloor ([xa]+2)/\alpha \rfloor - 1$。

证明。
情况 1： $a = 0$。因此根据引理 2 (b)，可得

$$\alpha[x] - \beta^2 < [xa] < \alpha[x] - \beta.$$

加上 2 并除以 α ，得到

$$[x] + 1 < \frac{[xa]+2}{\alpha} < [x] + \alpha < [x] + 2.$$

结果由此得出。

情况 2： $a = 1$。那么，由于 xa 是有效斐波那契表示，我们必有 $x = x'0$。按照 Reble [16], 中的论证，令 $x' = e_t \cdots e_2$，使得 $[x'] = \sum_{2 \le i \le t} e_i F_i$。于是

$$\begin{aligned}
[x'00] &= \sum_{2 \le i \le t} e_i F_{i+2} \\
&= \sum_{2 \le i \le t} e_i (\alpha F_{i+1} + \beta^{i+1}) \\
&= \alpha \sum_{2 \le i \le t} e_i F_{i+1} + \sum_{2 \le i \le t} \beta^{i+1} \\
&= \alpha [x'0] + \sum_{2 \le i \le t} \beta^{i+1}.
\end{aligned}$$

因此

$$-\beta^2 = \beta^3 + \beta^5 + \beta^7 + \cdots < [x'00] - \alpha[x'0] < \beta^4 + \beta^6 + \beta^8 + \cdots = \sqrt{5} - 2.$$

现在，在两边加上 $3 + \alpha[x'0]$，得到 $\alpha[x'0] + \alpha^2 < [x'01] + 2 < \alpha[x'0] + \sqrt{5} + 1$，换句话说，$\alpha[x] + \alpha^2 < [xa] + 2 < \alpha[x] + \sqrt{5} + 1$。两边除以 α，得到 $[x] + 1 < [x] + \alpha < \frac{[xa]+2}{\alpha} < [x] + 2$，这就证明了这种情形下的结论。 □

我们还需要以下引理。

引理 4. 假设 x, y 是两个长度相同的字，并且 $a, a', b, b' \in \Sigma_2$ 和 $D(x, y) = [y] - n[x]$ 对于某个 $n \ge 1$成立。那么

$$D(xa'a, yb'b) = D(x, y) + D(xa', yb') + b - na + b' - na'.$$

证明. 引理2 (a) 给出 $[x00] = [x] + [x0]$。因此 $[xa'a] = [x] + [x0] + 2a' + a$ 且 $[yb'b] = [y] + [y0] + 2b' + b$。因此

$$\begin{aligned}
D(xa'a, yb'b) &= [yb'b] - n[xa'a] \\
&= [y] + [y0] + 2b' + b - n([x] + [x0] + 2a' + a) \\
&= ([y] - n[x]) + ([yb'] - n[xa']) + b' - na' + b - na \\
&= D(x, y) + D(xa', yb') + b' - na' + b - na.
\end{aligned}$$

□

3 识别关系

本节研究识别一些简单关系所需的状态复杂度；例如，接受当且仅当 $[x] + c = [y]$ 时 $x \times y$ 的自动机的状态复杂度。

3.1 加法

定理 5。 对于所有整数 $c \geq 0$，存在一个最高位优先斐波那契自动机，其状态数至多为 $O(\log c)$，接受 $x \times y$，其中 $[x] + c = [y]$。

证明。我们假设 $c \geq 1$，因为情形 $c = 0$ 是平凡的。我们设计一个斐波那契自动机M_c，并行读取输入 $x \times y$，当且仅当 $[y] = [x] + c$ 时接受。我们对之前处理底数为k 表示法 [12]时所用的方法作相同修改。也就是说，我们首先找到一个值的范围 I，使其包含 c，并且如果 $[y] - [x]$ 曾经落到 I之外，那么 $[y'] - [x']$ 对 I 的所有右扩展 x' 以及 x 对 y' 的所有右扩展 y，它都始终处于该范围之外。

设 $[y] < [x]$。由引理 2 (c) 可知 $y < x$ 在字典序下。因此，若 y' 是 y 的右扩展，且 x' 是 x的右扩展，则有 $y' < x'$ 在字典序下，因此 $[y'] < [x']$。

现在假设 $[y] > [x] + c$。我们有

$$\begin{aligned}[yb] - [xa] &= [y0] - [x0] + b - a > (\alpha[y] - \beta^2) - (\alpha[x] - \beta) - 1 \\ &= \alpha([y] - [x]) - \beta^2 + \beta - 1 \geq \alpha(c+1) - \beta^2 + \beta - 1 = \alpha c - \beta^2 > c.\end{aligned}$$

因此，我们可以取范围 I 为 $[0, c]$。

我们的自动机 $M_c = (Q,\ \Sigma_2,\ q_0,\ \delta,\ A)$ 构造如下。

$$\begin{aligned}Q &\subseteq \{[a, b, d', d] \ :\ a, b \in \{0, 1\} \text{ and } d, d' \in I\}, \\ q_0 &= [0, 0, 0, 0], \\ \delta([a', b', d'', d'], [a, b]) &= [a, b, d', d] \text{ where } a, b, a', b \in \{0, 1\}, aa', bb' \neq 11, \text{ and} \\ &\qquad d = d' + d'' + b' - a' + b - a, \\ A &= \{[a, b, d', d] \ :\ a, b \in \{0, 1\} \text{ and } d = c\}.\end{aligned}$$

转移规则的正确性现在由引理 4 （取 $n = 1$）以及对输入长度的简单归纳法得出。

接下来，我们要更精确地定义 Q。定义 $d' = [y] - [x]$、$d = [yb] - [xa]$、$\varepsilon_x = [x0] - \alpha[x]$ 和 $\varepsilon_y = [y0] - \alpha[y]$。因此

$$\begin{aligned}d &= [yb] - [xa] = [y0] - [x0] + b - a \\ &= \alpha[y] + \varepsilon_y - \alpha[x] - \varepsilon_x + b - a = \alpha d' + \varepsilon_y - \varepsilon_x + b - a.\end{aligned}$$

改写后，我们得到

$$d' - d/\alpha = \frac{\varepsilon_x - \varepsilon_y + a - b}{\alpha}. \tag{1}$$

现在需要考虑三种情况。由引理 2 (b) 可知 $-\beta^2 < \varepsilon_x,\ \varepsilon_y < -\beta$。因此，情况如下。

情况 *1*：$a = b$。那么 $d/\alpha - 1 < d' < d/\alpha + 1$。

情况 *2*：$a = 0$、$b = 1$。然后 $d/\alpha - 2 < d' < d/\alpha$。

情形3： $a=1$、 $b=0$。然后 $d/\alpha < d' < d/\alpha + 2$。

这给出了与特定 d 对应的状态的八种可能形式：

$$\{[0,0,\lfloor d/\alpha\rfloor, d], [0,0,\lceil d/\alpha\rceil, d], [1,1,\lfloor d/\alpha\rfloor, d], [1,1,\lceil d/\alpha\rceil, d],$$
$$[0,1,\lfloor d/\alpha\rfloor, d], [0,1,\lfloor d/\alpha\rfloor - 1, d], [1,0,\lceil d/\alpha\rceil, d], [1,0,\lceil d/\alpha\rceil + 1, d]\}.$$

从一个形式为 $[a,b,d',c]$ 的终状态出发，我们确定该状态可能的前驱状态，以及这些前驱状态可能的前驱状态，依此类推。注意，根据方程（1），如果$[y]-[x]\geq 4$，那么对于 x 的任意右扩展，y 差值只会增大。我们将前驱状态分组为不同层级。形式为 $[a,b,d',c]$ 的状态处于第 0 层，这些状态的前驱状态处于第 1 层，依此类推。现在，我们需要给出每一层状态数以及层级数的上界。

第 i 层的状态形式为 $[a,b,d',d]$ 和 $d\in[D_i, D_i+r_i]$，其中某些 D_i， r_i。注意， $D_0=c$ 且 $r_0=0$。由于第 i 层状态中， d' 的最小可能值为 $\lfloor D_i/\alpha\rfloor - 1$，最大可能值为 $\lceil (D_i+r_i)/\alpha\rceil+1$，因此可得 $r_{i+1} < r_i/\alpha + 4$。从 $r_0=0$ 开始，连续使用该不等式四次可得，对于所有 i，都有 $r_i\leq 8$。因此，任意给定层级的所有状态中， d 至多有 $8+1=9$ 个可能取值。

为了估计可能的层级数，注意到方程（1）给出 $d' < (d+2)/\alpha$因此 $D_{i+1}+r_{i+1} < (D_i+10)/\alpha < D_i/1.1$ 只要 $D_i\geq 22$成立即可。因此，直到 $D_i\leq 22$为止，层级数至多为 $\log_{1.1} c = O(\log c$。当某一层级中的最大 d 小于 22 时，我们便停止计算层级，并直接纳入所有满足 $[a,b,d',d]$ 的状态 $d\leq 22$（这是一个常数数量）。

这表明共可达状态只有 $O(\log c)$ 个，我们的证明也就完成了。 □

对于 $[x]-c=[y]$，也可以给出类似的证明。

定理6。对于所有 $c\geq 0$，都存在一个自动机，其状态数至多为 $O(\log c)$，它接受$x\times y$，其中 $[x]-c=[y]$。

证明。考虑定理5证明中给出的自动机。这里唯一的区别是，我们跟踪的是 $[x]-[y]$ 而不是 $[y]-[x]$。因此，我们只需将转移函数改为如下形式：

$$\delta([a',b',d,d'],\ [a,b]) = [a,b,g,d],\ \text{若}\ g\in I,\ \text{其中}\ g = d+d'+a-b+a'-b'\ \text{且}\ aa',\ bb'\neq 11.$$

□

3.2 乘法

设 $n\geq 1$ 是一个自然数，且 $0\leq c<n$。我们给出了自动机识别关系 $Y=nX+c$ 所需状态数的上界。

我们令 $L_{n,c}$ 表示所有由 $\Sigma_2\times\Sigma_2$ 上的字组成的语言，其中第一个分量表示某个整数 $[x]$，第二个分量表示 $n[x]+c$。更准确地说，

$$L_{n,c} := \{x\times y\in(\Sigma_2\times\Sigma_2)^*\ :\ [y]=n[x]+c\}.$$

例如，图 2 展示了一个识别 $L_{2,0}$ 的最小确定有限自动机，由 Walnut 计算得出。编号为 0 的状态是初始状态（以无箭头的线段表示），接受状态 0、3、6 和 8 则以双圆圈标出。

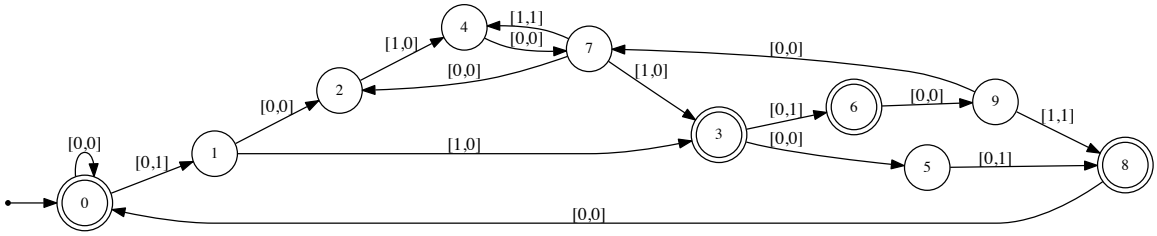


图2：斐波那契表示中的乘以 2。

定理 7。 存在一个具有 $O(n^2)$ 个状态、可接受 $L_{n,c}$ 的确定有限自动机。

Charlier等人证明了一个与定理 7 相关的结果。 [6]他们证明， $2n^2$对于输入属于斐波那契计数系统的常数 n 的倍数，接受这些倍数的确定有限自动机所需的状态数既是必要的，也是充分的。然而，尽管他们的结果与我们的结果密切相关，我们仍看不出如何直接从他们的结果推导出我们的结果。

对于斐波那契表示中的输入，我们采用与在基数为 k from [12] 的系统中识别这一关系时所概述的相同策略。具体而言，我们使用自动机 $M_{n,c}$ 来跟踪差函数 $D(x, y)$: $= [y] - n[x]$。这里有四个方面会带来复杂情况:

- 在基数为 k 的系统中，读取输入中的一个字母 a 会使读出的整数从 n 变为 $kn + a$。但对于斐波那契表示，公式要复杂得多。
- 仅维护一个 d（当前差值）已不够。我们还需要维护另一个相关量： d'（前一个差值）。
- 我们必须确保输入是有效的斐波那契表示。这意味着我们必须在状态中存储两个输入词各自读到的末字母。
- 要确定 I_n d 和 d'的区间变得不再那么简单；在该区间之外，没有任何右扩展会被接受。

正如我们所提到的，这个想法是，在输入 $xa' \times yb'$时，自动机应当跟踪$d = D(xa', yb')$。然而，考虑到引理 4，这还不够；我们还需要跟踪 $d' = D(x, y)$。此外，我们需要确定区间 I_n ，以便如果 d

永远落在此区间之外，我们就知道输入的任何右扩展都不可能落入L_n。因此，状态是形如 $[a', b', d, d'] \in \Sigma_2 \times \Sigma_2 \times I_n \times I_n$ 的 4 元组。

假设 I_n存在，转移函数的定义便可由引理 4 立即得出。具体而言，$M_{n,\ c} = (Q,\ \Sigma_2,\ q_0,\ \delta,\ A)$ 定义如下：

$$\begin{aligned}
Q &= \{[a', b', d, d'] \ :\ d, d' \in I_n \text{ and } a', b' \in \Sigma_2\},\\
\Sigma &= \Sigma_2 \times \Sigma_2,\\
q_0 &= [0, 0, 0, 0],\\
\delta([a', b', d, d'], [a, b]) &= [a, b, g, d],\ \text{if } g \in I_n \text{ where } g = d + d' + b - na + b' - na'\\
&\qquad \text{and } aa', bb' \neq 11;\\
A &= \{[a', b', d, d'] \ :\ d = c\}.
\end{aligned}$$

因此，剩下的任务是确定区间 I_n。我们证明可以取 I_n 为闭区间 $[-n, 2n-1]$。

引理 8。 设 $D(x, y) \leq -n-1$。则 $D(xa, yb) \leq -n-1$。

证明。我们有

$$\begin{aligned}
D(xa, yb) &= [yb] - n[xa]\\
&= [y0] - n[x0] + b - na\\
&< (\alpha[y] - \beta) - n(\alpha[x] - \beta^2) + 1\\
&= \alpha D(x, y) - \beta + n\beta^2 + 1\\
&\leq \alpha(-n-1) - \beta + n\beta^2 + 1\\
&= -n(\alpha - \beta^2) + 1 - \alpha - \beta\\
&= -n(\alpha - \beta^2)\\
&< -n\\
&\leq -n - 1.
\end{aligned}$$

□

引理 9. 令 a、$b \in \Sigma_2$，并假设 $D(x、\ y) \geq 2n$。那么 $D(xa、\ yb) \geq n$。

证明。我们有

$$\begin{aligned} D(xa, yb) &= [yb] - n[xa] \\ &= ([y0] + b) - n([x0] + a) \\ &\geq [y0] - n[x0] - n \\ &> (\alpha[y] - \beta^2) - n(\alpha[x] - \beta) - n \\ &= \alpha([y] - n[x]) - \beta^2 + n\beta - n \\ &= \alpha D(x, y) - \beta^2 + n\beta - n \\ &\geq 2n\alpha - \beta^2 + n\beta - n \\ &= (2\alpha + \beta - 1)n - \beta^2 \\ &> n - 1. \end{aligned}$$

□

引理 10。 令 $a, b, a', b' \in \Sigma_2$ 满足 $a = 0$ 或 $a' = 0$。假设 $D(x, y) \geq n$ 且$D(xa', yb') \geq n$。那么 $D(xa'a, yb'b) \geq n$。

证明。 根据引理 4可得

$$\begin{aligned} D(xa'a, yb'b) &= D(x, y) + D(xa', yb') + b - na + b' - na' \\ &\geq 2n - n(a + a') \\ &\geq n. \end{aligned}$$

注意，我们知道 a, a' 中至少有一个等于 0，否则就不是有效斐波那契表示。

□

定理 11。 我们可以取 $I_n = [-n, 2n - 1]$。

证明。 引理 8 表明，如果 $D(x, y) \leq -n - 1$，那么对于 x 和 y 的每一个右扩展，同一不等式都成立。

引理 9 和 10 结合起来表明，如果 $D(x, y) \geq 2n$，那么对于 $D(x', y') \geq n$，对 x 和 y' 的每个右扩展 x' y都成立。 □

现在，通过对 $|x| = |y|$ 进行简单的归纳法可得

$$\delta(q_0, xa' \times yb') = [a', b', D(xa', yb'), D(x, y)],$$

前提是 $xa' \times yb'$ 的任何前缀都不含有 $D(x', y') \notin I_n$。

现在我们可以证明定理7。

定理的证明 7. 我们构造的自动机 $M_{n,c}$ 由以下形式的状态组成$[a', b', d, d']$ 其中 $d, d' \in I_n$ 且 $a', b' \in \Sigma_2$。 因此总共有 $2 \cdot 2 \cdot 3n \cdot 3n$ 个状态，即 $O(n^2)$。

□

备注 12. 我们指出，在我们的构造中，确定有限自动机 $M_{n,c}$ 通常并非最小。不过，我们可以在二次时间内构造自动机，然后应用 Hopcroft [8] 或 Valmari [19] 提出的著名确定有限自动机最小化算法，在 $Y = nX + c$ 的 $O(n^2$ 对数 $n)$ 时间内得到识别

识别语言 $L_{n,0}$ 所构成的最小确定有限自动机的状态数形成数列 A372846，收录于在线整数数列百科全书（OEIS）。使用自由软件 Walnut计算得到的$1 \le n \le 10$ 值（不计死状态）为 2、10、23、40、59、85、114、146、181、224。数值证据表明，这个量大约为 $2n^2 + \alpha^2 n$，但我们尚不知如何证明。更深入地理解 $M_{n,c}$ 的可达状态，或许有助于缩小这个界。

4 线性子序列

若有限字母表上的数列 $(h(i))_{i\ge 0}$ 由确定性有限自动机输出机 M_h 生成，则称其为斐波那契自动数列：该自动机读取以斐波那契表示的整数输入 i，并返回 $h(i)$ 的值作为到达的最后一个状态所关联的输出。更准确地说，若 $M_h = (Q_h, \Sigma_2, \Delta_h, q_{0,h}, \delta_h, \tau_h)$，则对于所有有效表示 $x \in \Sigma_2^*$ 且满足$[x] = i$，都有 $h(i) = \tau_h(\delta_h(q_{0,h}, x))$。我们假设不存在对应于连续两个输入 1 的转移；换言之，若状态 q 可由某个其他状态经输入 1 到达，则 $\delta_h(q, 1)$ 未定义。有关此类数列的更多信息，请参见 [14]。

定理 13. 设$(h(i))_{i\ge 0}$ 是由具有 m个状态的确定性有限自动机输出机生成的斐波那契自动序列。对于常数 $c \ge 0$，存在一个具有 $O(m^2c^2)$ 个状态的确定性有限自动机输出机，生成$(h(i + c))_{i\ge 0}$。

证明。设 $(h'(i))_{i\ge 0}$ 是由 $M = (Q, \Sigma_2, q_0, \delta, \Delta, \tau)$ 生成的 $(h(i))_{i\ge 0}$ 的内部序列。我们采用与在基数 k 的情形中 [12] 所用方法类似的方法，并构造一个自动机 M'，使其在读入 x 时到达形如 $[[h'([x]) \ a'_0], \ldots, [h'([x] + c) \ a'_c]]$ 的状态，其中 a'_i 是 $([x] + i)$ 的末字母；读入字母 a 后，它会转移到下一个状态 $[[h'([xa]) \ a_0], \ldots, [h'([xa] + c) \ a_c]]$。

自动机 $M' = (Q', \Sigma_2, q'_0, \delta', \Delta, \tau')$ 的构造如下。Q' 中的状态具有如下形式：$[[p_0, a_0], \ldots, [p_c, a_c]]$，其中 $p_i \in Q$ 且 $a_i \in \Sigma_2$。初始状态 q'_0 为 $[[h'(0) \ 0], \ldots, [h'(c) \ a_c]]$，其中 a_c 是 (c) 的末字母。此外，我们有 $\tau'([[p_0, a_0], \ldots, [p_c, a_c]]) = \tau(p_c)$。

转移函数 δ' 更为复杂，需要加以说明。如果输入 δ' 的内容为 $[[p_0, a_0], \ldots, [p_c, a_c]]$ 和 a，它会按如下方式计算输出。首先，它会创建一个列表。然后依次遍历每个 $[p_i, a_i]$；若 $a_i = 0$，则将 $[\delta(p_i, 0), 0]$ 和 $[\delta(p_i, 1), 1]$ 加入列表；否则，只将 $[\delta(p_i, 0), 0]$ 加入列表。最后，若 $a = 0$，该函数输出第 0 个到第 c 个元素；否则输出第 1 个到第 $(c+1)$ 个元素。

现在来求 M' 中的状态数。由于 p_0、……、p_c 构成了 $(h'(i))_{i\ge 0}$ 的一个长度为 $c + 1$ 的子序列，因此共有 $O(m^2c)$ 种可能。此外， a_0、……、a_c 构成了无限斐波那契词的一个长度为 $c+1$ 的子序列，它们共有 $O(c)$ 种可能。因此，识别 $(h(i + c))_{i\ge 0}$ 的自动机 M' 有 $O(m^2c^2)$ 个状态。

□

定理 14。 设 $(t(i))_{i\geq 0}$ 为斐波那契-图厄-莫尔斯序列；换言之， $t(i)$ 是 i 的斐波那契表示中数字 1 的个数对 2 取模的结果。对于某个常数 $c \geq 0$，生成 $(t(i+c))_{i\geq 0}$ 的最小确定性有限自动机输出机中的状态数为 $O(c)$。

证明。根据定理 13 中给出的构造，我们来构造用于生成 $(t(i+c))_{i\geq 0}$ 的自动机 M_c。输入 x 后，自动机 M_c 会到达形如 $[[t'([x])\ a'_0],\ \ldots\ldots,\ [t'([x]+c)\ a'_c]$ 的状态，其中 a'_j 是 $([x]+j)$ 的末字母。考虑从 [17] 生成 $(t(i))_{i\geq 0}$ 的自动机。我们可以看到， $t'([x]+j)$ 决定了 a'_j。因此，根据引理 1，我们得出 M_c 中有 $O(c)$ 个状态。 □

我们猜想，生成 $(t(i+c))_{i\geq 0}$ 的最小自动机的状态数也有线性下界。以下问题留作开放问题。

问题 1. 证明，生成 $(t(i+c))_{i\geq 0}$ 的最小自动机的状态数为 (c)。

在一篇即将发表的论文中， [11], 我们研究了生成（f（$i+c$））$_{i\geq 0}$的状态复杂度，其中（f（i））$_{i\geq 0}$是斐波那契词$\mathbf{f}$，并证明其为 $O(\log\ c)$。

定理15。 设 $n \geq 1$ 和 $0 \leq c < n$，并设$(h(i))_{i\geq 0}$ 是一个由具有 m 个状态的确定性有限自动机输出机生成的斐波那契自动序列。存在一个具有 $O(m^2n^4)$ 个状态的确定性有限自动机输出机，可识别线性子序列$(h(ni+c))_{i\geq 0}$。

令（h（n）$_{n\geq 0}$ 是一个由确定性有限自动机输出机计算的任意斐波那契自动序列

$$M_h = (Q_h, \Sigma_2, \Delta, \delta_h, q_{0,h}, \tau_h),$$

其中 Δ 是一个有限字母表， τ_h 是与 M_h相关联的输出函数。固定 n和 c。现在我们如下构造一个 UFAO $M_u = (Q_u, \Sigma_2, \Delta, \delta_u, q_{0,u}, A_u, \tau_u)$:

$$\begin{aligned}
Q &= \{[a', b', d, d', q'] \ :\ a', b' \in \Sigma_2 \text{ and } d, d' \in I_n \text{ and } q' \in Q_h\},\\
\delta([a', b', d, d', q'], a) &= \{[a, b, g, d, q] \ :\ b \in \Sigma_2,\ g = d + d' + b' - na' + b - na,\ d \in I_n,\\
&\qquad aa', bb' \neq 11, \text{ and } q = \delta_h(q', b)\},\\
q_{0,u} &= \{[0, b', 0, i, p] \ :\ 0 \leq i \leq 2n-1,\ b' = \mathbf{f}[i],\ p = \delta_h(q_{0,h}, (i))\},\\
A_u &= \{[a', b', d, d', q'] \ :\ d = c\},\\
\tau_u([a', b', d, d', q']) &= \tau_h(q').
\end{aligned}$$

从概念上讲， M_u 通过取$M_{n,c}$ 与 M_h 的笛卡尔积构成，其中 M_h 接收输入 x，“猜测” y，并在 $x \times y$ 上模拟$M_{n,c}$ ，在 y上模拟 M_h 。UFAO 的输出是 y 上与 $d = c$对应的 M_h 的唯一输出。

初始状态 $q_{0,u}$ 需要作一些说明。它对应于所有满足 $[x] = 0$ （换言之，对于某个 j）的输入 x ，以及所有满足 $[y] - n[x] \in I_n$ 的可能 y。这是必要的，因为我们希望构造的确定性有限自动机输出机无论在输入前添加多少个前导零，都能返回正确结果。

接下来，我们对 M_u 使用子集构造，得到一个确定性有限自动机输出机 M。对于输入 x，这个确定性有限自动机输出机计算 $h(n[x]+c)$。M 的状态是 M_u 的状态子集。我们先证明一个引理，用以刻画哪些状态子集可能出现在 M 的一个状态中。

引理16。对于 M 中的每个状态 p，都存在 $a \in \Sigma_2$ 以及整数 S、T，使得 $[S, S+T] \subseteq [-n, 2n-1]$，并且 p 的每个状态都具有 $[a, b_i, d_i, d'_i, s_i]$ 的形式，其中 $0 \le i \le T$ 且 $d_i = S + i$。

换言之，p 中各状态的第三个分量互不相同，并且恰好一次地遍历范围 $[S, S+T]$ 内的所有整数。

证明。考虑一个可由输入 xa 到达的状态 p。子集构造生成 M_u 中那些可由输入 $xa \times yb$ 到达的状态，其中 $r = |y| = |x|$。因此，对应的整数 $[yb]$ 恰好是区间 $[0, F_{r+2} - 1]$ 内的整数。对应的 d_i 具有 $[yb] - n[xa]$ 的形式。然而，并非所有这些都能构成有效状态，因为 d'_i, d_i 可能落在区间 $[-n, 2n-1]$ 之外。因此，允许的 d_i 的范围是若干区间的交集，因而自身也是 $[-n, 2n-1]$ 的一个子区间。 □

定理15的证明。考虑从 xa 到 M的输入。由引理16可知，对于该输入，M会到达一组状态，这些状态对应于 M_u中的输入 $xa \times y_i b_i$，其中 $d_i = [y_i b_i] - n[xa] = S + i$，对 $0 \le i \le T$、某个词 y_i 和字母 b_i均成立。令 $[y_0 b_0] = Y$。那么对于$0 \le i \le T$，有 $[y_i b_i] = Y + i$。注意到 $d'_i = [y_i] - n[x]$，定义 $e_i = d'_{i+1} - d'_i$，对 $0 \le i < T$成立。令 g为将 t的斐波那契表示去掉最后一位后所得整数映射到 t的函数；引理3表明，$g(t) = \lfloor (t+2)/\alpha \rfloor - 1$。于是 $d'_i = g(Y+i) - n[x]$。三元组 (b_i, e_i, s_i) 由 $0 \le i \le T$对应的 $(\mathbf{f}[Y+i], g(Y+i+1) - g(Y+i)\ h'(Y+i))$ 给出；一旦选定 d'_0，这个公式就完全确定了所有的 b_i、d'_i 和 s_i。具体来说，$d'_i = d'_0 + \sum_{0 \le j < i} e_j$。

在输入 x时，生成 $(g(j+1) - g(j))_{j \ge 0}$ 的自动机只需跟踪最近读入的两个字母；若其中任一个是 1，则 $g([x]) = 1$，否则 $g([x]) = 0$。现在将三元组 (b_i, e_i, s_i) 看作无限三元组序列 $\mathbf{x} := (\mathbf{f}[j], g(j+1) - g(j)\ h'(j))_{j \ge 0}$ 的一个子序列。该序列是自动序列，因为 $(\mathbf{f}[j])_{j \ge 0}$、$(g(j+1) - g(j))_{j \ge 0}$ 和内部序列 $(h'(j))_{j \ge 0}$ 都是自动序列。此外，序列 $\mathbf{x}$ 由一个有 $O(m)$ 个状态的确定性有限自动机输出机生成。于是，只需说明有限序列 $(\mathbf{f}[Y+i], g(Y+i+1) - g(Y+i)\ h'(Y+i))_{0 \le i \le T}$ 是 $\mathbf{x}$ 的哪一个长度为 $T+1$ 的因子，就能完全确定该序列；根据引理 1，这样的因子有 $O(m^2 n)$ 个。

综上，共有

(a) $O(n)$ 种 $d_0 \in [-n, 2n-1]$ 的选择，

(b) $O(n)$ 种 $d'_0 \in [-n, 2n-1]$ 的选择，

(c) $O(n)$ 种 T 的选择，

(d) $O(m^2 n)$ 种选择，决定自动序列 $\mathbf{x}$ 中的哪个特定子词给出 $0 \le i \le T$ 的 $(b_i, d'_{i+1} - d'_i, s_i)$，

(e) $O(1)$ 种选择，适用于 a。

因此共有 $O(m^2n^4)$ 个可能的状态。 □

例 17. 让我们求出 $\mathbf{f}[2i]$ 的最小确定性有限自动机输出机。为此，我们可以使用免费软件 `Walnut`，并执行以下命令

```
def f2 "?msd_fib F[2*i]=@1":
combine F2 f2=1:
```

这将得到图3所示的确定性有限自动机输出机。

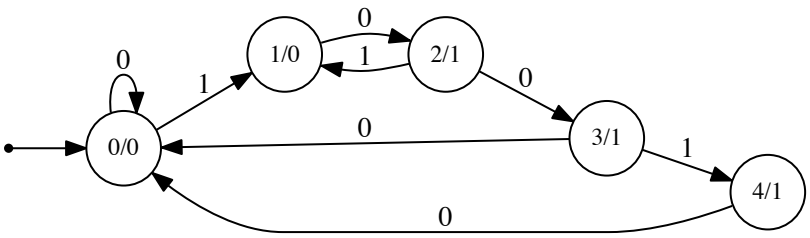


图3：$\mathbf{f}[2i]$ 的一个斐波那契DFAO。

注意， `Walnut` 的输出是一个最小化自动机。因此，图3是本文所给构造方法生成的自动机的最小化版本。

注记 18。 我们指出，为 $(h(ni+c))_{i\geq 0}$ 构造的自动机并不是最小的。不过，我们可以通过最小化算法在多项式时间内找到最小自动机。Hopcroft [7] 的最小化算法可在 $O(n\log\ n)$ 时间内运行，并且可以按照 van Spaendonck [20] 的描述将其调整用于确定性有限自动机输出机。

对于 $c=0$ 和 $h(i)=\mathbf{f}[i]$（即无限斐波那契词）这一特殊情形，线性子序列$\mathbf{f}[ni]$ 对应的最小确定性有限自动机输出机的状态数由在线整数数列百科全书中的数列A385021给出。前 10 项对应于 $1\leq n\leq 10$，分别为 2、5、10、17、27、36、52、65、78、103。

4.1 Bosma 和 Don 的一个问题的应用

众所周知，一个 k自动序列也可以表示为某个不动点 $\gamma^{\omega}(0)$ 在编码下的像，该不动点由一个可延长态射 γ 生成；该态射将某个字母表 Σ 中的每个字母映射为一个长度为 Σ^* 的 k [1] 词。这里，“可延长”是指字母 0 的像以 0 开头。事实上，适当的态射可以直接从

自动机推导出来，方法如下：字母表中的字母就是自动机的状态 q；值 $\gamma(q)$ 由输入 0、1、……、$k-1$ 时到达的状态给出。编码由与每个状态相关联的输出定义。斐波那契自动序列大致也是如此，不过现在 $\gamma(q)$ 的值由输入 0 和 1 时的输出给出（如果计数系统规则允许后者）。

例19。 从图中所示的自动机3我们可以直接读出子序列 $(\mathbf{f}[2i])_{i\geq 0}$ 可由态射 γ 和编码 ρ表示:

$$\begin{aligned}
\gamma(0) &= 01, & \tau(0) &= 0,\\
\gamma(1) &= 2, & \tau(1) &= 0,\\
\gamma(2) &= 31, & \tau(2) &= 1,\\
\gamma(3) &= 04, & \tau(3) &= 1,\\
\gamma(4) &= 0, & \tau(4) &= 1.
\end{aligned}$$

迭代态射 γ后，我们得到序列 $01231040\cdots$；应用编码 τ后，我们得到 $00110000\cdots$。

Bosma 和 Don [3] 研究了斐波那契词 $\mathbf{f}$ 的线性子序列。他们主要关注生成该序列所需的最小态射的大小 $(\mathbf{f}[ni+c])_{i\geq 0}$，并证明了该态射大小的一个指数上界。这里，态射的大小是指写出它所需符号的总数，即 $\sum_{a\in\Sigma}|\gamma(a)|$。

作为我们研究结果的推论，所需态射的大小可以从指数级降至多项式级。

推论20。 设 $n\geq 1$ 和 $0\leq c<n$，并设 $(h(i))_{i\geq 0}$ 是一个斐波那契自动序列。存在一个大小为 $O(n^4)$ 的态射，生成子序列 $(h(ni+c))_{i\geq 0}$。

证明。 正如我们在定理 15中所看到的，所得的自动机有 $O(m^2n^4)$ 个状态。因此，相应的态射大小为 $O(m^2n^4)$。 □

5 通过 B¨uchi 算术进行构造

与此前讨论的底数为 k [12] 的情形类似，给定一个自动序列，我们可以使用免费软件 `Walnut` [13, 18] 计算其线性子序列对应的自动机。 `Walnut`采用 B¨uchi 算术的解释（自然数连同加法以及函数 $V_k(n)$ 的逻辑理论；对于某个固定的 $k\geq 2$，该函数表示能整除 n 的最大 k 次幂），并实现了 B¨uchi [5, 4]提出的判定过程：它接收一阶逻辑陈述，并将其转换为相应的自动机。此类软件的用户关心以算法方式构造相应自动机需要多长时间；本节将研究这一时间复杂度。我们使用

自动机的转移总数 M 作为构造 M 的计算复杂度的替代指标。

特别是，在当前版本中，我们没有使用前几节讨论过的用于构造自动机的算法，而是采用了适用于任何可加数系的更通用技术。这意味着中间自动机的规模可能不同于我们构造的最小自动机，运行时间也可能相应地更长。有时，本节中的复杂度界会包含 $O(\log i)$ 或 $O(i^j)$ 项，其中某个常数为 i；与 $O(\log i)$ 或 $O(i^j)$ 项（其中某个常数为 i）中的乘法相比，这些加法项并不显著。

此外，对于一个 n状态自动机，实现中的最小化步骤可以使用运行时间为 $O(n\log n$ 的算法，Hopcroft [7] 或 Valmari [19]提出了此类算法。Hopcroft 的算法稍作修改后，也可用于确定性有限自动机输出机（DFAO），具体如 van Spaendonck[20]所述。注意，对于一个 n状态的确定有限自动机（DFA），转移数量为 $O(n)$。最小化会应用于所创建的每个 DFA 或 DFAO。本文有时只进行部分最小化，并忽略自动机完全最小化可能产生的影响，以便分析运行时间复杂度并得到上界；无论如何，状态复杂度和运行时间复杂度的上界仍然成立，而且所得的运行时间复杂度反映了完成一次完整最小化的计算成本。

考虑 Mousavi 等人给出的自动机 M_{add} [14]，它在输入 $x\times y\times z$ 时接受 $[x]+[y]=[z]$。在输入 $x\times y\times z$ 时，该自动机到达由序列 $([x0^n]+[y0^n]-[z0^n])_{n\geq 0}$ 表示的状态。我们对这个自动机进行修改，以便跟踪一些额外信息。考虑由 M_{add} 创建的自动机 $M'_{\text{新增}}$，它在输入 $xa\times yb\times ze$ 时到达由 $[s,s',e]$ 表示的状态，其中 $s=([xa0^n]+[yb0^n]-[ze0^n])_{n\geq 0}$，且 $s'=([x0^n]+[y0^n]-[z0^n])_{n\geq 0}$。

如果我们有 $M_{\text{add}}=(Q,\ \Sigma_2^3,\ \delta,\ q_0,\ A)$，则按如下方式创建 $M'_{\text{add}}=(Q',\ \Sigma_2^3,\ \delta',\ q'_0,\ A')$。

$$\begin{aligned} Q' &= Q\times Q\times \Sigma_2,\\ \delta'([s,s',e],[a,b,e]) &= [\delta(s,[a,b,e]),s,e],\\ q'_0 &= [q_0,q_0,0],\\ A' &= \{[s,s',e]\ :\ s\in A\}. \end{aligned}$$

我们使用 $M'_{\text{add}}(x,\ y,\ z)[i]$ 来表示状态 M'_{add} 在输入 $x\times y\times z$ 时到达的第 i 个分量（对于 $i\in\{0,1,2\}$）；例如，$M'_{\text{add}}(x,\ y,\ z)[0]$ 是 $([x0^n]+[y0^n]-[z0^n])_{n\geq 0}$。自动机 M'_{add} 有 $O(1)$ 个状态。

引理 21。 用于识别关系 $[x]_k=c$ 的最小自动机有 $O(\log c)$ 个状态。

证明。 该自动机由一个节点组成的线性链构成，节点标记为 c 的斐波那契表示，其长度为 $O(\log c)$。 □

定理 22。 设 $c\geq 0$ 为一个固定常数。识别关系 $[x]=c$ 的自动机可以在 $O((\log^2 c)(\log\log c))$ 时间内计算得到。

证明。 我们递归地构造自动机 $M_{=c}$。

基础情形为 $c = 0$，且自动机有 1 个状态。

如果 c 是偶数，我们递归地计算一个最小确定有限自动机 $M_{=(c/2)}$，使其在输入 y 上识别 $[y] = c/2$，然后使用一阶表达式

$$\exists y\ M_{=(c/2)}(y) \wedge M_{\text{add}}(y, y, x)$$

并将其转换为自动机。我们知道，M_{add} 的自动机有 $O(1)$ 个状态；根据引理 21，自动机 $M_{=(c/2)}$ 有 $O(\log c)$ 个状态。首先，我们对 $M_{=(c/2)}$ 和 M_{add} 应用乘积构造，使 $M_{=(c/2)}$ 部分处理 y，而 M_{add} 部分处理 $y \times y \times x$；所得自动机有 $O(\log c)$ 个状态。乘积自动机中的每个状态都形如 $[q, p]$：在输入 $x \times y$上，q是最小自动机$M_{=(c/2)}$ 中的一个状态，它表示根据引理 21，$(c/2)$ 中与 y 匹配的字母数；p 则是 M_{add} 中的一个状态，使得 $p(0) = 2[y] - [x]$。然后，我们通过从转移中移除对应于 y 的分量来应用 $\exists y$ 量词，从而得到一个非确定有限自动机。该非确定有限自动机中，不存在从同一状态出发、指向不同状态的两条相同转移。因此，经过子集构造后，所得确定有限自动机与该非确定有限自动机相同。接着进行最小化，所得确定有限自动机就是引理 21 中有 $O(\log c)$ 个状态的自动机。

若 c为奇数，我们递归地计算一个确定有限自动机 $M_{=(c-1)}$，使其在输入 y时识别 $[y] = c - 1$，然后使用一阶表达式

$$\exists y\ M_{(c-1)}(y) \wedge M_{\text{add}}(y, 1, x)$$

它被转换为一个自动机。按照与 c 为偶数时所述类似的构造，我们得到引理 21 中针对常数 c 的确定有限自动机，其具有 $O(\log c)$ 个状态。

将最小化步骤考虑在内，我们得到如下时间复杂度递归公式：

$$T(c) = \begin{cases} O((\log c)(\log\log c)) + T(c/2), & \text{if } c \bmod 2 = 0; \\ O((\log c)(\log\log c)) + T(c-1), & \text{if } c \bmod 2 = 1. \end{cases}$$

因此，总时间为 $O((\log^2 c)(\log\log c))$。 □

定理 23。 设 $c \geq 0$ 为一个固定常数。识别关系 $[x]+c=[z]$ 的自动机可在 $O((\log^2 c)(\log\log c))$ 时间内计算得到。

证明。 假设我们再次修改了 M'_{add}，使其在输入 $xa \times yb \times ze$时，除了跟踪 e之外，还跟踪 a，且其状态形式为 $[s, s', a, e]$。我们可以使用以下表达式来构造自动机 M_c，使其在输入 $x \times z$时识别 $[x] + c = [z]$。此外，我们还有一个自动机 $M_{=c}$，用于识别定理 $[y] = c$ 中的关系 22。

$$\exists y\ M'_{\text{add}}(x, y, z) \wedge M_{=c}(y).$$

首先，我们对 M'加和 $M_{=c}$应用乘积构造，使得 M'_{add}部分处理$x \times y \times z$，而 $M_{=c}$部分处理 y。所得自动机 M'_c有 $O(\log\ c)$ 个状态。

令 Pre(c,i) 表示 (c) 中的前 i 个字母。对于输入 $xa\times yb\times ze$，其中 yb 是 $0^*y'b$，且 $y'b$ 是 (c) 的一个前缀，自动机 M'_c 会到达状态 $[q,p]$，其中 q 来自 M'加，$p=|y'b|$ 来自 $M_{=c}$，表示 y 中有多少个字母已与 (c) 匹配。因此，我们有 $q=[s,s',a,e]$，其中 $s(0)=[xa]+[yb]-[ze]$，$s'(0)=[x]+[y]-[z]$，且 $[$ Pre$(c,p)]=[yb]$。所以 $[ze]-[xa]=[$Pre$(c,p)]-s(0)$，且 $[z]-[x]=[$Pre$(c,p-1)]-s'(0)$。因此，对 M'_c 中的每个状态 $[q,p]$，都可以赋予一个元组 $[d,d',a,e]$，其中 $d=[ze]-[xa]$ 且 $d'=[z]-[x]$，并且由相同的 xa,ze 可达的两个状态，不可能被赋予不同的 $[d,d',a,e]$。

然后我们从转移中移除 y 分量，并将自动机确定化以获得一个确定有限自动机。考虑确定有限自动机中通过输入 $xa\times ze$ 可到达的某个（非死）状态。该状态由一些 $[q,p]$ 组成，使得 $[ze]-[xa]=[$Pre$(c,p)]-s(0)$ 且 $[z]-[x]=[$Pre$(c,p-1)]-s'(0)$。在非确定有限自动机中，不存在从同一状态出发并通向不同状态的两个相同转移。因此，经过子集构造后，所得的确定有限自动机与非确定有限自动机相同。

根据定理 5的证明，我们知道，识别 $[x]+c=[z]$ 在输入$x\times z$时的自动机只需记录一个差值 $[z]-[x]$，其范围为 $[0,c]$，具有 O（log c）个状态。因此，经过某种最小化后，自动机 $M_c=$（Q，Σ_2，q_0，δ，A）可以形式化定义如下。令 $I_c=[0,c]$。

$$\begin{aligned}
Q &= \{[a,b,d,d'] \ : \ d,d'\in I_c\},\\
q_0 &= [0,0,0,0],\\
\delta([a',b',d,d'],[a,b]) &= [a,b,g,d] \text{ where } g=d+d'+b'-a'+b-a,\\
A &= \{[a,b,d,d'] \ : \ d=c\}.
\end{aligned}$$

注意，上述形式化定义中的状态不一定都可达。

考虑到创建 $M_{=c}$ 和执行最小化步骤所花费的时间，创建识别 $[x]+c=[z]$ 的自动机 M_c 所需时间为$O((\log^2 c)(\log\log c))$，其输入为$x\times z$。

□

接下来讨论识别关系 $[z]=n[x]$ 的自动机，其中 n 是一个固定常数。实现该自动机有几种可能的递归方式。

- $\exists y\ M_{n-1}(x,\ y)\ \wedge\ M$加$(x,\ y,\ z)$。
- $\exists y_1,\ y_2\ M_{\lfloor n/2\rfloor}(x,\ y_1)\ \wedge\ M_{\lceil n/2\rceil}(x,\ y_2)\ \wedge\ M$加$(y_1,\ y_2,\ z)$。

然而，一种 hird 递归实现　　l方法　　hat foll比上述两种方法都更高效。

若 n 为偶数，我们递归地计算一个确定有限自动机 $M_{n/2}(x,y)$，它识别 $[y]=(n/2)[x]$，然后使用一阶表达式

$$\exists y\ M_{n/2}(x,y)\ \wedge\ M'_{\text{add}}(y,y,z),$$

该表达式通过针对 $\wedge$ 的直积构造转换为自动机，并通过投影去除 y 上的转移（即投影第 2 个坐标）。

如果 n 为奇数，我们递归计算一个识别 $[y]_k=(n-1)[x]_k$ 的确定有限自动机 $M_{n-1}(x,\ y)$，并使用表达式

$$\exists y\ M_{n-1}(x,y)\ \wedge\ M'_{\text{add}}(x,y,z).$$

在这两种情况下，转换理论上都可能生成一个非确定性自动机（通过投影），而对其进行确定化可能需要指数级时间。不过，我们证明事实并非如此。

定理 24. 设 $n \geq 1$ 为一个固定常数。可以在 $O(n^6 \log^2 n)$ 时间内，计算出一个用于在输入 $x \times y$ 时识别 $[y] = n[x]$ 的自动机。

证明。我们首先对 n 进行归纳，证明可以递归地构造自动机 M_n，其状态具有 $[a, b, d, d']$ 的形式，使得在输入 $xa \times yb$ 时，自动机到达一个满足 $d = [yb] - n[xa]$ 且 $d' = [y] - [x]$ 的状态，并且 d, $d' \in I_n = [-n, 2n - 1]$。

基础情形：$n = 1$。

在这种情况下，一个简单的单状态自动机即可识别 $[y] = [x]$。这个唯一的状态对应于 $[a, b, d, d'] = [0, 0, 0, 0]$。

对于归纳步骤，假设对所有 $n' < n$ 结论都成立；我们证明它对 n 也成立。分两种情况。

情况 1： n 模 $2 = 0$。

根据归纳假设，我们构造了一个确定有限自动机 $M_{n/2}$，它有 $O(n^2)$ 个状态，分别对应所有可能的 $[a, b, d, d']$，其中 a, $b \in \Sigma_2$ 以及 d, $d' \in I_{n/2} = [-(n/2), 2(n/2) - 1]$。在这种情况下，我们使用公式

$$\exists y \; M_{n/2}(x, y) \wedge M'_{\text{add}}(y, y, z).$$

首先，我们创建一个自动机 M'_n，它是 $M_{n/2}$ 和 M'_{add} 的乘积构造。因此，这个自动机有 $O(n^2)$ 个状态。考虑一个输入 $xa \times yb \times ze$，对于 M'_n 会将其引向某个状态 $[q, p]$，其中 q 来自 $M_{n/2}$，而 p 来自 M'。根据 M'_n 的构造，我们知道 $q = [a, b, d, d']$，其中 $d = [yb] - (n/2)[xa]$、 $d' = [y] - (n/2)[x]$，并且 $p = [s, s', e]$，其中 $s = (2[yb0^n] - [ze0^n])_{n \geq 0}$、 $s' = (2[y0^n] - [z0^n])_{n \geq 0}$。因此，我们有 $2d - s(0) = [ze] - n[xa]$ 以及 $2d' - s'(0) = [z] - n[x]$。所以，对于通过 $xa \times yb \times ze$ 在 M'_n 中可达的每个状态，都可以为其赋予一个元组 $[a, e, d, d']$，其中 $d = [ze] - n[xa]$ 且 $d' = [z] - n[x]$。此外，具有相同 xa 的两个输入不能到达被赋予不同 $[a, e, d, d']$ 的状态。

接下来，我们消去 y 分量。所得自动机是一个非确定有限自动机，我们使用子集构造来创建一个确定有限自动机。考虑一个可通过 $xa \times ze$ 到达的确定有限自动机状态。该状态由非确定有限自动机中的一些数对 $[q, p]$ 组成，使得 $q = [a, b, d, d']$、 $p = [s, s', e]$，并且我们可以从 $[q, p]$ 构造出 $[a, e, 2d - s(0)\ 2d' - s'(0)]$，其中 $[ze] - n[xa] = 2d - s(0)$ 和 $[z] - n[x] = 2d' - s'(0)$。

在确定有限自动机的每个状态中，使用与定理15证明中类似的论证，部件有 $O(n^4)$ 种可能。此状态中的所有 $[q, p]$ 都应具有相同的 $2d - s(0)$ 和 $2d' - s'(0)$（有 $O(n^2)$ 种可能）。查看Hamoon等人的转移表 [14]，并考虑所有 $\delta_{\text{add}}(s', [b, b, e]) = s$，其中 δ_{add} 是 M_{add} 的转移函数，那么对于每个 b，从转移中得到的 $(s'(0)\ s(0))$ 数对都是唯一的。因此，鉴于我们已经确定了每个 $[q, p]$ 的 b、 d、 d'，并且固定了 $2d - s(0)$ 和 $2d' - s'(0)$，那么 s、 s' 也就随之确定（有 $O(1)$ 种可能）。所以，确定有限自动机中有 $O(n^6)$ 个状态。

根据定理7的证明，我们知道，识别 $n[x] = [z]$ 在输入$x \times z$时的自动机只需跟踪一个差值 $[z] - n[x]$，其范围为 $I_n = [-n, 2n-1]$。因此，经过一定的最小化后，确定有限自动机 $M_n = (Q, \Sigma_k^2, \delta, q_0, A)$可形式化定义如下。

$$\begin{aligned} Q &= \{[a, e, d, d'] \ : \ a, e \in \Sigma_2, \ d, d \in I_n]\}, \\ q_0 &= [0, 0, 0, 0], \\ \delta([a', e', d, d'], [a, e]) &= [a, e, g, d] \text{ where } g = d + d' + e' - na' + e - na, \\ A &= \{[a, e, d, d'] \ : \ d = 0\}. \end{aligned}$$

情况 2： n 模 $2 = 1$。

根据归纳假设，我们有一个确定有限自动机 M_{n-1}，它有 $O(n^2)$ 个状态，对应于所有可能的 $[a, b, d, d']$，其中 $a, \ b \in \Sigma_2$且 $d, \ d' \in I_{n-1} = [-(n-1), \ 2(n-1) - 1]$。在这种情况下，我们使用公式

$$\exists y \ M_{n-1}(x, y) \ \wedge \ M'_{\text{add}}(x, y, z)$$

通过通常的乘积构造，为 M_{n-1}和 M'加法构造自动机 M'_n。因此，这个自动机有 $O(n^2)$ 个状态。考虑一个输入 $xa \times yb \times ze$，它使 M'_n 到达某个状态 $[q, p]$，其中 q来自 M_{n-1}，而 p来自 M'加法。根据 M'_n的构造，我们知道$q = [a, b, d, d']$，其中 $d = [yb] - (n-1)[xa]$、 $d' = [y] - (n-1)[x]$，并且 $p = [s, s', e]$，其中 $s = ([xa0^n] + [yb0^n] - [ze0^n])_{n \geq 0}$、 $s' = ([x0^n] + [y0^n] - [z0^n])_{n \geq 0}$。因此，我们有 $d - s(0) = [ze] - n[xa]$且 $d' - s'(0) = [z] - n[x]$。所以，对于 M'_n中可由 $xa \times yb \times ze$到达的每个状态，都可以为其赋予一个元组 $[a, e, d, d']$，其中 $d = [ze] - n[xa]$且 $d' = [z] - n[x]$。此外，两个输入若具有相同的 xa和 ze，就不可能到达被赋予不同 $[a, e, d, d']$的状态。

接下来，我们将 y 分量投影消去。由此得到的自动机是一个非确定有限自动机，我们使用子集构造来创建一个确定有限自动机。考虑确定有限自动机中一个可由 $xa \times ze$到达的状态。该状态由非确定有限自动机中的一些数对 $[q, p]$组成，使得 $q = [a, b, d, d']$， $p = [s, s', e]$，并且我们可以从 $[q, p]$构造 $[a, e, d - s(0) \ d' - s'(0)]$，其中 $[ze] - n[xa] = d - s(0)$且$[z] - n[x] = d' - s'(0)$。

在确定有限自动机的每个状态中，使用与定理15的证明中类似的论证， $q = [a, b, d, d']$部分有 $O(n^4)$ 种可能，而 p中的 e有 $O(1)$ 种可能。该状态中的所有 $[q, p]$都应具有相同的 $d - s(0)$和 $d' - s'(0)$（有 $O(n^2)$ 种可能）。如果我们查看 Hamoon等人的转移表 [14]，并考虑所有 $\delta_{\text{add}}(s', [a, b, e]) = s$，其中 δ_{add}是 M_{add}的转移函数，那么对于每个 a, b, e，由转移得到的 $(s'(0) \ s(0))$ 数对都是唯一的。因此，鉴于我们已经确定了每个 $[q, p]$的 a, b, e, d, d'，并且固定了 $d - s(0)$和 $d' - s'(0)$，s, s' 也就随之确定了（有 $O(1)$ 种可能）。因此，确定有限自动机中有 $O(n^6)$ 个状态。

从定理 7的证明中，我们知道，识别 $n[x] = [z]$ 在输入$x \times z$时的自动机只需跟踪一个差值 $[z] - n[x]$，其范围为 $I_n = [-n, 2n-1]$。因此，经过

一些小型 最小化后，确定有限自动机 $M_n = (Q,\ \Sigma_k^2,\ \delta,\ q_0,\ A)$ 可形式化定义如下 s.

$$\begin{aligned} Q &= \{[a, e, d, d'] \ :\ a, e \in \Sigma_2,\ d, d \in I_n]\}, \\ q_0 &= [0, 0, 0, 0], \\ \delta([a', e', d, d'], [a, e]) &= [a, e, g, d] \text{ where } g = d + d' + e' - na' + e - na, \\ A &= \{[a, e, d, d'] \ :\ d = 0\}. \end{aligned}$$

接下来，我们分析创建上述自动机的时间复杂度。我们使用 $T(n)$ 表示递归创建 M_n 所需的时间。我们使用所遍历的转移数量作为所需时间的度量。

在情况 1 中，为获得 M_n 所遍历的转移数量为：M'_{add}对应的 $O(1)$ 个转移，$M_{n/2}$ 和 $M'_{\text{加法}}$的乘积构造对应的 $O(n^2)$ 个转移，投影去除 y 分量对应的 $O(n^2)$ 个转移，确定化自动机对应的 $O(n^6)$ 个转移；总计为 $O(n^6)$ 个转移，此外还要加上创建 $M_{n/2}$ 以及进行最小化时所遍历的转移数量。

在情况 2 中，为获得 M_n 所遍历的转移数量为：M_{n-1}对应的 $O(n^2)$ 个转移，M'_{add} 对应的 $O(1)$ 个转移，M_{n-1} 和M'_{add}的乘积构造对应的 $O(n^2)$ 个转移，投影去除 y 分量对应的 $O(n^2)$ 个转移，确定化自动机对应的 $O(n^6)$ 个转移；总计为 $O(n^6)$ 个转移，此外还要加上创建 M_{n-1}以及进行最小化时所遍历的转移数量。

综合以上内容，我们得到以下递归公式：

$$\begin{cases} T(n) = O(n^6 \log n) + T(n/2), \\ T(n) = O(n^6 \log n) + T(n-1). \end{cases}$$

因此，我们有 $T(n) = O(n^6 \log^2 n)$。 □

定理 25. 设 $n \geq 1, 0 \leq c < n$ 为固定常数。识别输入为 $x \times z$ 时的 $n[x]+c = [z]$ 的自动机，可在 $O(n^6 \log^2 n)$ 时间内计算得到。

证明. 设 M_c 为定理23 中的自动机，并设 M_n 为定理 24中的自动机。当且仅当 $[y] + c = [z]$时，我们称 $M_c(y, z)$ 的值为 1。因此，我们可以使用以下表达式来构造 $M_{n,c}$:

$$\exists y\ M_n(x, y)\ \wedge M_c(y, z).$$

我们首先对 M_n 和 M_c 应用乘积构造。所得自动机 $M'_{n,c}$ 有 $O(n^2 \log\ c)$ 个状态。

考虑一个输入 $xa \times yb \times ze$ for $M'_{n,\ c}$它会导向某个状态 $[q, p]$，其中 q来自M_n，而 p来自 M_c。根据 $M'_{n,\ c}$的构造，我们知道 $q = [a, b, d_q, d'_q]$，其中$d_q = [yb] - n[xa]$、$d'_q = [y] - n[x]$，且 $p = [b, e, d_p, d'_p]$，其中 $d_p = [ze] - [yb]$、$d'_p = [z] - [y]$。因此我们有 $d_q + d_p = [ze] - n[xa]$ 和 $d'_q + d'_p = [z] - n[x]$。所以，对于自动机 $M'_{n,\ c}$中的每个状态，都可以赋予一个元组 $[a, e, d, d']$，其中 $d = [ze]-n[xa]$ 且 $d' = [z]-n[x]$

而且，不存在两个由相同的 $xa,\ ze$ 可达的状态，被赋予不同的 $[a,e,d,d']$。

接下来，我们从 $M'_{n,c}$ 中投影掉 y 分量。所得的非确定有限自动机需要通过子集构造进行确定化，才能得到确定有限自动机。考虑确定有限自动机中可由 $xa\times ze$ 到达的一个状态。该状态由非确定有限自动机中的若干个 $[q,p]$ 组成，满足 $q=[a,b,d_q,d'_q]$,$p=[b,e,d_p,d'_p]$ 和 $d_q+d_p=[ze]-n[xa]$, $d'_q+d'_p=[z]-n[x]$。对于确定有限自动机的每个状态，使用类似于证明定理15时的论证， $q=[a,b,d_q,d'_q]$ 部分有 $O(n^4)$ 种可能。然后，我们为这个确定有限自动机的状态固定 e,d,d'，共有 $O(n^2)$ 种可能。由于 $d_q+d_p=d$ 和 $d'_q+d'_p=d'$，我们现在也可以确定 d_p,d'_p。因此，确定有限自动机有 $O(n^6)$ 个状态。

根据定理 7 的证明，我们知道，识别 $n[x]+c=[z]$ 在输入 $x\times z$ 时只需跟踪一个差值 $[z]-n[x]$，其范围为 $I_n=[-n,2n-1]$。因此，经过若干最小化步骤后，确定有限自动机 $M_{n,c}=(Q,\Sigma_k^2,\delta,q_0,A)$ 可形式化定义如下。

$$\begin{aligned}
Q &= \{[a,e,d,d'] \ :\ a,e\in\Sigma_2,\ d,d'\in I_n\},\\
q_0 &= [0,0,0,0],\\
\delta([a',e',d,d'],[a,e]) &= [a,e,g,d] \text{ where } g=d+d'+e'-na'+e-na,\\
A &= \{[a,e,d,d'] \ :\ d=c\}.
\end{aligned}$$

考虑到最小化步骤以及创建 M_n 所花费的时间，创建 $M_{n,c}$ 所花费的时间为 $O(n^6\log^2 n)$。□

定理 26. 设 $n\geq 1$, $0\leq c<n$ 为固定常数。给定一个生成$(h(i))_{i\geq 0}$ 的 m 状态确定性有限自动机输出机，我们可以在 $O(n^6\log^2 n+m^2n^4\log(m^2n^4))$ 时间内计算出 $(h(ni+c))_{i\geq 0}$ 的确定性有限自动机输出机。

证明。给定一个确定性有限自动机输出机 $M_h=(Q_h,\Sigma_k,\delta_h,q_{0,h},\Delta_h,\tau_h)$，它生成 $(h(i))_{i\geq 0}$，且有 m 个状态；我们要构造一个用于 $(h(ni+c))_{i\geq 0}$ 的自动机。我们将 M_h 在 y 上的输出记为 $M_h(y)$，并且有 $M_h(y)=h([y])$。此外，根据上文有 $M_{n,c}$，使得 $M_{n,c}(x,y)$ 当且仅当 $n[x]+c=[y]$ 时为 1。

首先，我们对以 $x\times y$ 为输入的 $M_{n,\ c}$ 和以 y 为输入的 M_h 使用乘积构造，得到自动机 M'。考虑在 M' 中通过输入 $xa\times yb$ 可达的状态 $[q,p]$，其中 q 来自$M_{n,\ c}$，而 p 来自 M_h。我们有 $q=[a,b,d,d']$，其中 $d=[yb]-n[xa]$、 $d'=[y]-n[x]$ 和 $p=\delta$（$q_{0,\ h}$，yb）。该自动机有 O（mn^2）个状态。

接下来，我们投影掉 y 分量，并使用子集构造从非确定性自动机得到一个确定性有限自动机输出机。以 xa 为输入时，自动机到达一个由若干 $[q,p]$ 组成的状态，其中 $q=[a,b,d,d']$ 满足存在 yb，其中 $d=[yb]-n[xa]$, $d'=[y]-n[x]$, d, $d'\in I_n$ 和 $p=\delta$（$q_{0,\ h}$，yb）$\in Q_h$。对于确定性有限自动机输出机中的每个状态，利用与定理 15 的证明中类似的论证，有 O（m^2n^4）种可能。因此，该确定性有限自动机输出机有 O（m^2n^4）个状态。

考虑到构造 $M_{n,c}$ 以及最小化步骤所需的时间，创建给定的 $M\,M_{n,c}$ 需要 $O(n^6\log^2 n+m^2n^4\log(m^2n^4))$ 时间。□

6 结论

在此前的一篇论文中，[12]，我们研究了与本文类似的主题，但采用的是以 k 为基数的输入，而非斐波那契表示中的输入。我们可以尝试在其他计数系统中研究这些主题并解决类似问题。例如，我们提出以下开放问题。

问题 2. $r\left(r\left(i+c\right)\right)_{i\geq 0}$，其中 r 是特里波那契词?

我们猜想，我们的结果可以推广到任意定义在次数为 d 的 Pisot 计数系统上的广义自动序列。在这种情况下，识别关系 $Y=nX+c$ 可以使用 $O(n^d)$ 个状态，而计算线性子序列的确定性有限自动机输出机可以使用 $O(n^{d+2})$ 个状态；对这些问题的研究留待未来开展。

参考文献

[1] Jean-Paul Allouche 和 Jeffrey Shallit。《自动序列：理论、应用与推广》。剑桥大学出版社，2003年。[2] Jean Berstel。斐波那契词——综述。载于 G. Rozenberg 和 A. Salomaa 编，《L 之书》，第13–27页。施普林格出版社，1986年。[3] Wieb Bosma 和 Henk Don。为斐波那契数的算术子序列构造态射。载于 Venanzio Capretta、Robbert Krebbers 和 Freek Wiedijk 编，《理论与实践中的逻辑和类型系统》，《计算机科学讲义》第14560卷，第100–110页。施普林格，2024年。[4] V´eronique Bruy`ere、Georges Hansel、Christian Michaux 和 Roger Villemaire。逻辑与 p-可识别的整数集。《比利时数学学会通报——西蒙·斯蒂文》，1(2):191–238，1994年。[5] J. Richard B¨uchi。弱二阶算术与有限自动机。《Zeitschrift f》¨《ur Mathematische Logik und Grundlagen der Mathematik》，6:66–92，1960年。重印于 Saunders Mac Lane 和 Dirk Siefkes 编，《J. Richard B》¨《uchi 文集》，第398–424页，施普林格出版社，1990年。[6] ´Emilie Charlier、Narad Rampersad、Michel Rigo 和 Laurent Waxweiler。在一种线性计数系统中识别 $m\mathbb{N}$ 的最小自动机。《整数：组合数论电子期刊》，11B:#A4，2011年。[7] John Hopcroft。一种用于最小化有限自动机状态的 $n\log n$ 算法。载于 Zvi Kohavi 和 Azaria Paz 编，《机器与计算理论》，第189–196页。学术出版社，1971年。

[8] John E. Hopcroft 和 Jeffrey D. Ullman。《自动机理论、语言与计算导论》。Addison-Wesley计算机科学丛书。Addison-Wesley出版公司，1979年。[9] Jozef Jir´asek, Jr.、Galina Jir´askov´a 和 Juraj ˇSebej。无歧义有限自动机上的运算。《国际计算机科学基础杂志》，29(05):861–876，2018年。[10] C. G. Lekkerkerker。用斐波那契数之和表示自然数。Simon Stevin，29:190–195，1951/52年。[11] Delaram Moradi、Pierre Popoli、Jeffrey Shallit 和 Ingrid Vukusic。斐波那契词移位的状态复杂度。准备中，2026年。[12] Delaram Moradi、Narad Rampersad 和 Jeffrey Shallit。斐波那契 k-自动序列的线性子序列复杂度，2026年。arXiv:2512.10017 [cs.FL]。[13] Hamoon Mousavi。使用 Walnut 进行自动定理证明，2021年。arXiv:1603.06017[cs.FL]。[14] Hamoon Mousavi、Luke Schaeffer 和 Jeffrey Shallit。斐波那契自动字的判定算法，I：基本结果。《RAIRO理论信息学与应用》，50(1):39–66，2016年。[15] Alexander Ostrowski。关于丢番图逼近理论的评注。《汉堡大学数学研讨会论文集》，1(1):77–98，1922年。重印于《数学论文集》第3卷，第57–80页。Birkh¨auser出版社，1984年。[16] Don Reble。Zeckendorf表示与Wythoff表示：关于A007895的评注，2008年。可见于 https://oeis.org/A007895。[17] Jeffrey Shallit。斐波那契–Thue–Morse序列的子词复杂度：Dekking猜想的证明。《数学研究》，32(3):729–735，2021年。[18] Jeffrey Shallit。《自动序列的逻辑方法：使用Walnut探索词组合学》，伦敦数学学会讲义系列第482卷。剑桥大学出版社，2023年。[19] Antti Valmari。快速、简明、实用的DFA最小化。《信息处理快报》，112(6):213–217，2012年。[20] Flip van Spaendonck。《自动序列：局部变化对复杂度的影响》。硕士论文，拉德堡德大学，荷兰，2020年。

[21] E. Zeckendorf. 用斐波那契数或卢卡斯数之和表示自然数。列日皇家科学院公报 ´ ` , 41(3-4):179–182，1972。