arXiv:2512.25014v1 [cs.LG] 31 Dec 2025

扩散语言模型是可证明最优的并行采样器

HaozheJiang Nika Haghtalab Lijie Chen
ericjiang@berkeley.edu nika@berkeley.edu lijiechen@berkeley.edu

2026年1月1日

摘要

扩散语言模型（DLM）已成为自回归模型的一种有前景的替代方案，可通过并行词元生成加快推理。我们通过形式化定义并行采样模型，并证明经多项式长度思维链（CoT）增强的扩散语言模型能够以最优数量的顺序步骤模拟任意并行采样算法，从而为这一优势奠定了严格的理论基础。因此，只要某个目标分布能够通过少量顺序步骤生成，就可以使用扩散语言模型以相同数量的最优顺序步骤生成该分布。然而，如果无法修改先前已揭示的词元，带思维链的扩散语言模型仍可能产生庞大的中间占用空间。我们证明，启用重新掩码（将未掩码词元转换为掩码）或修订（将未掩码词元转换为其他未掩码词元），并结合思维链，可以进一步使扩散语言模型以最优空间复杂度模拟任意并行采样算法。我们还通过确立严格的表达能力差距，进一步论证修订的优势：支持修订或重新掩码的扩散语言模型，其表达能力严格强于不支持这些操作的模型。我们的研究结果不仅从理论上论证了扩散语言模型作为最高效并行采样器的潜力，也倡导在扩散语言模型中启用修订。

1 引言

扩散语言模型（DLM，Austin 等人 [2021], Nie 等人 [2025],Song 等人 [2025]）近来已成为自回归（AR）生成的一种颇具前景的替代方案，其潜力在于能够实现更快的生成和更高效的推理。与 AR 模型不同，后者即使运行在大规模并行硬件上，仍必须逐词元顺序生成文本；扩散语言模型则通过迭代地对掩码序列进行去噪来解码，从而允许并行解除多个位置的掩码。这一根本性的架构差异有望显著降低生成和推理延迟，也使扩散语言模型在日益重视推理时扩展的当下受到广泛关注 Song 等人 [2025],Khanna 等人 [2025]。

然而，与高度优化的 AR 模型相比，扩散语言模型仍处于起步阶段，其承诺的优势何时（或者是否）能转化为有理论依据的增益，远未明朗。因此，理论对于评估扩散语言模型的真正潜力尤为重要：不同于 AR 模型——其在思维链（CoT；Wei 等人 [2022]）推理下的能力已得到充分理解——扩散语言模型的基本极限，例如它们究竟能节省多少顺序计算，以及这种并行性需要多少空间和内存资源，仍鲜有探索。

在这项工作中，我们通过电路复杂度的视角对 DLMs 进行形式化，并以电路深度和宽度作为抽象概念，分别表示并行计算一个函数所需的时间和空间资源，从而为研究 DLMs 提供了这样的理论基础。正如我们将在下文详述的那样，我们的结果表明，扩散语言模型属于最高效的并行采样器之列。

我们的首个结果（定理 3.1）表明，配备足够长思维链的 DLM 可以用最少的顺序计算步骤模拟任意采样过程。具体而言，任何可由深度为 d 的电路实现的分布，都可以由 DLM 在 d 轮解码中生成，

从而达到目标计算所需的最少顺序步骤数。这与自回归解码形成鲜明对比。具体来说，尽管已有研究指出，配备思维链的自回归模型可以模拟任意布尔电路 [Li 等人， 2024], 但在这种情况下，顺序步骤数（对于自回归模型而言，即思维链的长度）随电路规模线性增长，也就是说，它取决于节点总数，而非深度。

除了生成样本所需的步数外，我们还分析了扩散语言模型的工作内存占用（定理 3.2），并表明它在很大程度上取决于这类模型特有的设计选择——尤其是推理时机制，例如 重新掩码 [Nie 等人， 2025] 以及修订 Song 等人[2025]。重新掩码是一种推理时机制，允许已解除掩码的词元重新被掩码（加噪）并重新采样（去噪）。另一方面，修订允许将未掩码词元直接更改为其他词元。

最后，我们通过零奇偶性的字符串均匀采样问题，证明重新掩码或修订能够生成严格更丰富的分布。具体而言，采用重新掩码或修订的 DLM 可以用常数步数生成该分布（定理 4.1 和 4.2），而既不采用重新掩码也不采用修订的 DLM 则无法做到这一点（定理 4.5）。我们的结果清楚地表明，源自 DLM 特殊结构的重新掩码和修订，在推理时间受限时有望显著增强模型的能力。这表明未来可以探索更多重新掩码策略和更灵活的前向过程。

1.1 相关工作

扩散语言模型。 扩散模型最初用于连续域中的生成建模 [Ho et al., 2020, Sohl-Dickstein et al., 2015], , 之后被扩展到离散域。D3PM 的开创性工作 [Austin 等人，2021] 将类别前向核形式化，并推广了吸收掩码; 随后，一系列语言建模工作相继出现 [He et al., 2023, Lou et al., 2024, Sahoo et al., 2024]。此外，近期的工作 [Song et al., 2025, Nie et al., 2025, Yang et al., 2025, Ye et al., 2025, Xie et al., 2025, Khanna 等人，2025] 扩大了扩散语言模型的规模，并取得了可与自回归模型媲美的性能，同时生成速度更快。

解码顺序、重新掩码与修订。 扩散语言模型可以按任意顺序进行推理，但研究发现，解码顺序会显著影响扩散语言模型的性能 [Kim et al., 2025, Yang et al., 2025, Xie et al., 2025]。重新掩码 Nie 等人 [2025], Yang 等人 [2025] 是另一种推理时策略，它引入了更丰富的解码路径，以进一步提升性能。近期，人们提出了带修订的扩散语言模型 [Song 等人，2025] ，并展现出卓越的能力。然而，这些策略之间的根本差异尚不明确。

2 预备知识

本节将说明本文使用的符号和约定，并介绍用于分析问题的理论框架。

记号。 令 $\mathcal{V}$ 为词汇表。本文默认始终考虑二元词汇表 $\mathcal{V} = \{a, b\}$ ，但扩散语言模型适用于任何其他离散词汇表。我们还考虑一个掩码词元M。我们考虑定义在 $\mathcal{V} \cup \{$M$\}$上的长度为 L 的序列。对于序列$x_t \in (\mathcal{V} \cup \{M\})^L$， $t \in [0, 1]$ 表示噪声级别，即序列中被掩码词元所占的比例。为简单起见，我们假设 $t \in \{0, 1/L, 2/L, \ldots, (L-1)/L,\ 1\}$，因此 $t \cdot L$ 是一个整数。这是文献中的标准记号，我们默认采用这一记法。本文也用 x 泛指词元序列。对于正整数 n，令 $[n] = \{1, \ldots, n\}$。不含掩码的序列称为无噪声序列。我们用x_t^i表示 $i \in [L]$ ，以指代 x_t的第 i个元素；用 $x_t^{i:j}$ 表示 x_t 中从第 i个元素到第 j个元素（包含两端）的子序列。由于下标用于表示噪声级别，我们默认使用上标表示序列的索引。

对于序列的索引，我们默认使用上标。我们将支撑集为 S 的分布集合记为$\Delta(S)$。

2.1 扩散语言模型

扩散语言模型（DLM）的核心是预测器 $p(\cdot|x_t) \in \Delta\left(\mathcal{V}^L\right)$，它根据噪声序列 $x_t \in (\mathcal{V} \cup \{\mathrm{M}\})^L$预测无噪声序列。令 $p^i(\cdot|x_t)$ 表示位置i处词元的分布。该预测器满足：给定 x_t 后，不同位置词元的预测条件独立，并且它只会改变掩码词元。用数学形式表示，对于所有 $x \in \mathcal{V}^L$，$x_t \in (\mathcal{V} \cup \{\mathrm{M}\})^L$，我们有

$$p(x|x_t) = \prod_{i=1}^{L} p^i\left(x^i|x_t\right), \text{ and } p^i\left(x^i = x_t^i|x_t\right) = 1 \text{ if } x_t^i \in \mathcal{V}.$$

这些约束源于前向过程的定义：不同位置的词元会被独立地掩码为 M。

在推理时，我们指定希望输出序列达到的噪声级别 s，并反掩码一组位置，使其与该噪声级别相匹配。按照 Kim 等人 [2025], 的惯例，我们用 $\mathcal{S} = \mathcal{F}(x_t) \subseteq \left\{i|x_t^i = \mathrm{M}\right\}$ 表示依赖于 x_t 的待反掩码位置集合。简单的反掩码策略$\mathcal{F}(x_t)$只需知道哪些位置被掩码，并从中均匀随机地选择一个集合$\mathcal{S} \subseteq \left\{i|x_t^i = \mathrm{M}\right\}$ 进行反掩码，使得 $|\mathcal{S}| = L(t-s)$。本文大部分内容中，我们将 $\mathcal{F}$ 限制为确定性函数。

我们还考虑训练 DLM 生成在输入 $q \in \mathcal{V}^n$条件下的输出 $o \in \mathcal{V}^m$ 这一一般情形。在推理时，我们从 $x_t = q\mathrm{M}^{L-n}$ 开始，即在输入序列中填入掩码，并采用迭代生成过程。每次迭代中，我们首先计算要解码的位置$\mathcal{S} = \mathcal{F}(x_t)$，然后使用 $p^i(\cdot|x_t)$ 为 $i \in \mathcal{S}$解码相应位置。迭代在预先设定的步数 D 结束，此时得到一个无噪声序列 x_0。最后，我们提取输出 $o = x_0^{L-m+1:L}$。生成过程总结于算法 1 中。

思维链（CoT）是一种语言模型技术，使提示 q 的输出 o 能够在生成一些中间词元之后出现。在我们的框架中，这意味着 $L > m + n$，而 x_0 的形式为 $q|\mathrm{CoT}|o$。为简单起见，本文始终假设 L 与 $q|\mathrm{CoT}|o$长度相同；但在实践中，L是预先设定的，通常大于 $q|\mathrm{CoT}|o$的长度。我们也允许 q 为空字符串，这对应于生成无条件分布。

重新掩码。 重新掩码是一种用于提升输出质量的推理策略。它允许在生成过程中将未掩码的词元重新掩码。我们通过函数 $\mathcal{G}(\cdot)$ 对此进行建模，该函数在每次迭代结束时生成一个集合 $\mathcal{T} = \mathcal{G}(x_t) \subseteq [L]$，表示需要重新掩码并设为 M 的位置。默认情况下，我们考虑 $\mathcal{G}$，它是一个随机函数。重新掩码过程也纳入了算法 1。

修订。 近期的研究（例如 Song 等人 [2025] ）提出了这样的前向过程：词元在完全被掩码之前可以变为另一个词元。我们将此称为带修订的扩散语言模型。考虑带修订的扩散语言模型时，我们放宽了未掩码词元必须保持不变这一约束。不过，我们仍然施加约束，要求不同位置上的词元分布相互独立。形式化地，对于所有 $x \in \mathcal{V}^L$，$x_t \in (\mathcal{V} \cup \{\mathrm{M}\})^L$，我们有

$$p(x|x_t) = \prod_{i=1}^{L} p^i\left(x^i|x_t\right), \text{but it is possible that } p^i\left(x^i = x_t^i|x_t\right) < 1 \text{ for } x_t^i \in \mathcal{V}.$$

除非另有说明，本文中的 DLM 均指不进行重新掩码或修订的标准模型。

算法 1DLM 推理

输入： 提示 $q \in \mathcal{V}^n$，一个包含以下内容的 DLM：预测器 p、长度 L、轮次 D、解掩码位置策略 $\mathcal{F}$，可选的重新掩码位置策略 $\mathcal{G}$。

输出： 输出 $o \in \mathcal{V}^m$

$x \leftarrow q\mathrm{M}^{L-n}$

对于 $j = 1$ 到 $D-1$ **执行**

$\mathcal{S} \leftarrow \mathcal{F}(x)$。

对每个 $i \in \mathcal{S}$ 执行 $x^i \sim p^i(\cdot|x)$。

$\mathcal{T} \leftarrow \mathcal{G}(x)$。 // 如果启用了重新掩码

对每个 $i \in \mathcal{T}$ 执行 $x^i \leftarrow \mathrm{M}$。 // 如果启用了重新掩码

循环结束

$x^i \sim p^i(\cdot|x)$，用于每个剩余的掩码位置。

返回 $x^{L-m+1:L}$

2.2 布尔电路

布尔电路是一种用于刻画执行特定并行计算复杂度的模型。电路 C 定义为分层有向无环图（DAG）。形式上，有向无环图 $C = (V, E)$ 由一组顶点 V和一组有向边 $E \subseteq V \times V$组成，且不含有向环。顶点 $|V|$的数量称为电路的规模。此外，存在一个函数 $l: V \to \mathbb{Z}^+$，它将每个顶点分配到某一层，并满足：对于所有的 $(v_1, v_2) \in E$，$l(v_2) = l(v_1) + 1$。也就是说，没有边会跳过某一层。每个顶点表示一个输入、一个逻辑门或一个输出。输入顶点和输出顶点的入度和出度分别为零，并且它们从 $\{0,1\}$中取布尔值。逻辑门对其输入执行各种运算，我们将在接下来的段落中具体说明。因此，我们可以沿着图中的逻辑门，从输入计算出输出。

电路的输入为 $(x, R) \in \{0,1\}^n \times \{0,1\}^r$，由两部分组成：$\chi \in \{0,1\}^n$ 表示实际输入，$R \in \{0,1\}^r$ 是随机比特，从 $U^r = \text{Uniform}(\{0,1\}^r)$ 中采样，即在 $\{0,1\}^r$ 上的均匀分布。我们将电路的输出记为 $C(\chi, R)$。电路的深度 d和宽度 w分别定义为

$$d = \max_{v \in \mathcal{V}} l(v), w = \max_{\ell \in [d]} |\{v | v \in \mathcal{V}, l(v) = \ell\}|.$$

也就是说，深度是层的总数，宽度是单层中顶点数量的最大值。深度表示执行由 C 表示的计算所需的最少顺序步骤数，宽度表示执行该计算所需的最少存储比特数。最后，我们称 C模拟随机函数 $f: \{0,1\}^n \to \{0,1\}$，如果

$$\Pr[f(\chi) = \psi] = \Pr_{R \sim U^r}[C(\chi, R) = \psi] \text{ for all } \chi \in \{0,1\}^n, \psi \in \{0,1\}^m.$$

在本文中，我们用拉丁字母 x，y表示词元序列，用希腊字母 χ，ψ表示布尔序列。我们使用布尔电路来建模生成分布的计算复杂度。本文中的所有分布均假设由某个电路生成。此外，我们还将 DLM（包括预测器 p、$\mathcal{F}$ 和 $\mathcal{G}$）建模为电路。由于电路模型是现代计算机的标准模型，因此使用电路来建模计算机能够采样的分布以及 DLM 中的计算是很自然的。此外，使用电路模型研究神经网络的表达能力已成为惯例 [Li et al., 2024, Merrill 和 Sabharwal, 2023]。

复杂度类。 一个问题 是一个随机映射 $\mathcal{L}: \{0,1\}^* \to \{0,1\}^m$。一组电路 $\{C^n\}_{n \in \mathbb{N}}$模拟问题 $\mathcal{L}$，如果

$$\Pr[\mathcal{L}(\chi) = \psi] = \Pr_{R \sim U^{r|\chi|}}\left[C^{|\chi|}(\chi, R) = \psi\right] \text{ for all } \chi \in \{0,1\}^*, \psi \in \{0,1\}^m.$$

这里，r^n 表示 C^n中的随机比特数。

电路的一组基本逻辑门是 {与门、或门、非门、恒等门}。与门、或门的入度为 2，而非门、恒等门的入度为 1。它们的出度均为 1。与门、或门、非门实现了名称所示的逻辑计算，而恒等门则原样输出输入。若存在一组电路 $\{C_n\}_{n\in\mathbb{N}}$，使其能够计算问题 $\mathcal{L}$，且任意一个 C_n 都具有 $O(\text{poly}(n))$ 个逻辑门和 $O\left((\log n)^k\right)$深度，则问题 $\mathcal{L}$ 属于 NC^k，其中 $k\in\mathbb{N}$。现在我们将复杂度类推广到更强大的类。如果允许与门、或门具有无界入度，相应的复杂度类称为 AC^k。如果进一步允许使用具有无界入度的多数门，当且仅当输入中 1 的数量多于 0 的数量时，多数门才输出 1；相应的复杂度类称为 TC^k。多数门可以看作是允许电路进行计数，因为要判断哪种数字占多数，电路必须统计它们各自出现的次数。已知对于所有 $i\in\mathbb{N}$，$\text{NC}^i\subseteq\text{AC}^i\subseteq\text{TC}^i\subseteq\text{NC}^{i+1}$ [（Vollmer，1999]）。

DLM 的电路模型。 回想一下，我们将词汇表限制为二进制。为了对词汇表以及掩码词元 M 进行编码，我们将使用固定的 2 位编码

$$a\to 00,\quad b\to 01,\quad \mathrm{M}\to 10,$$

未使用的编码 11 无效且永远不会生成。对于序列 $x\in(V\cup\{M\})^L$，设其编码为 $\chi\in\{0,1\}^{2L}$，其中数对 $(\chi_{2i-1},\ \chi_{2i})$ 对 x^i 进行编码。为方便起见，我们允许电路接收 a，b，将 a 和 b 的数值分别视为 0 和 1，而非其编码值 00 和 01。

DLM 的预测器 p 是一组输入来自编码空间的电路。对于每个位置 $i\in[L]$，我们有一个电路 C^i：$\{0,1\}^{2L}\times\{0,1\}^{r_i}\to\{00,01,10\}$，它将共享的编码序列 χ_t 作为输入，并使用 r^i 个随机比特 $R^i\sim U^{r^i}$ 来模拟 $p_i(\cdot|x_t)$。与前面对 DLM 的定义类似，我们要求不同 i 的随机比特相互独立，以便 $p(x|x_t)=\prod_{i=1}^L p^i\left(x^i\middle|x_t\right)$。此外，每个电路 C_i 只能解除掩码词元的掩码，即如果 $\chi_t^{2i-1:2i}\neq 10$，则 i 处的输出数对等于输入数对。

掩码和重新掩码的处理方式类似。具体而言，函数 $\mathcal{F}$ 由确定性布尔电路 $f\colon\{0,1\}^{2L}\to\{0,1\}^L$表示，其中 $f(\chi_t)^i$ 表示第 i 个位置是否应取消掩码。同样，函数 $\mathcal{G}$ 由可能随机化的电路函数 $g\colon\{0,1\}^{2L}\times\{0,1\}^r\to\{0,1\}^L$表示，其中 $r\sim U^r$ 和 $g(\chi_t,\ r)^i$ 表示第 i 个位置是否应重新掩码。

在本文中，我们假设所有扩散语言模型都由常深度电路构成。这样，我们可以用 *DLM* 推理期间的轮次数 D 来反映扩散语言模型采样分布所需的顺序步骤。

3 扩散语言模型是高效的采样器

本节将证明，DLM 能以尽可能少的并行计算步骤和内存模拟电路。如第 2.2 节所述，电路的深度和宽度自然地形式化了从分布中采样所需的最少顺序步骤和内存。本节将 DLM 记为 M，并用 $M(q)$表示使用算法 1 构建的 DLM 在给定输入 q 时产生的输出。

3.1 带思维链的 DLM

以下定理形式化地表明，扩散语言模型能够以可能的最少顺序步骤生成任意分布。

定理 3.1。 设 C 是一个深度为 d、包含 m 个输出位、n 个输入位、$R\sim U^r$ 个随机输入位和 N个顶点的电路。存在一个带有思维链且长度为 $L=N$的 ***DLM*** M，使得对于每个输入 q，M（q）生成

[1]对于更大的词汇表，可采用标准归约方法：将词元编码为二进制码字，并相应扩展序列长度。

与 C（q、 R）相同的分布。此外， M 的解码步数为 d，相应的预测器 p 和未掩码位置策略 $\mathcal{F}$ 均可由常深度电路表示。

在给出证明之前，我们先定义一个对我们的构造非常有用的函数。

定义 1. 布尔函数 ShiftR： $\{0,1\}^n \to \{0,1\}^n$ 定义如下。对于任意 $\chi \in \{0,1\}^n$,

$$\mathsf{ShiftR}(\chi)^i = \begin{cases} \chi^{i-1} & if\, 1 < i \le n \\ 1 & if\, i = 1 \end{cases}$$

模拟该函数属于 NC^0。换言之，该电路可以通过 O 个具有 n 的 poly(n) AND、OR（入度为 2）以及非门，在 O(1)中实现。对于任意布尔变量 η，1 可以由 $1 = \eta \vee \neg\eta$ 实现，其余部分则直接来自输入。

定理 *3.1* 的证明。 我们先介绍一些记号。

记号。 不失一般性，我们可以假设所有输入顶点都位于第一层，所有输出顶点都位于最后一层，并且第一层中没有随机位。我们定义以下变量:

- v^1，.....， v^N 表示电路中的顶点列表；这些顶点按照层号递增的顺序排列 s 在该序列中，
- $u^1, \ldots, u^N$ 为顶点输出的相应变量;
- $l^1, \ldots, l^N$ 为每个顶点的层号;
- $s^1, \ldots, s^d$ 为每一层中第一个顶点的索引（为简便起见， $s^0 = 0, s^{d+1} = L+1$）；
- $w^1, \cdots, w^d$ 为 C中每一层的宽度，即对于所有 $i = 1, \ldots, d$， $w^i = s^{i+1} - s^i$。

在此证明中，为方便起见，我们使用 x 的下标来标记正在进行的迭代，而非噪声级别。例如，模型的初始序列为 $x_1 = q\mathrm{M}^{L-n}$，经过第一次迭代后变为 x_2。

生成过程。 现在我们来描述希望实现的生成过程。我们的构造会在迭代 i期间计算第 i 层电路，将输出追加到已有输出的末尾，并据此计算第 $i+1$ 层。用数学形式表示，我们实现以下中间序列:

$$x_i^j = \begin{cases} u^j & \text{if } j < s^{i+1} \\ \mathrm{M} & \text{if } j \ge s^{i+1} \end{cases}.$$

因此，生成过程会在序列 $x_d = u^1 \cdots u^N$ 处停止，并且 $x_d^{L-m+1:L}$ 等于 o。下面我们将介绍如何构造实现这一生成过程的 DLM M （具体而言，即函数 $\mathcal{F}$ 和预测器 p）。

构造 $\mathcal{F}$。 为了实现上述过程，函数 $\mathcal{F}$ 需要根据 x_i识别当前的迭代索引 i，并据此生成 $\mathcal{S} = \{s^{i+1}, \ldots, s^{i+2} - 1\}$。由于 χ_i^{2j-1} 根据词元编码表示 x_i^j 是否为掩码，因此，序列 $\left\{\chi_i^{2s^j-1}\right\}_{j=1,\ldots,d}$ 表示电路中的每一层是否在 x_i中被解码。设 $\chi_i^{2s-1} \in \{0,1\}^d$ 为该序列的拼接，则$(\chi_i^{2s-1})^j$ 和 $\mathsf{ShiftR}\left(\chi_i^{2s-1}\right)^j$ 分别表示电路中的第 j 层是否在 x_i 和 x_{i+1}中被解码。因此， $\left(\mathsf{ShiftR}\left(\chi_i^{2s-1}\right) \wedge \neg\chi_i^{2s-1}\right)^j$ 表示第 j 层是否应在迭代 i 中根据 x_i被解码。换言之，仅当 $j = i+1$时，它才等于 1。综上，我们设定

$$f(\chi_i)^j = \left(\mathsf{ShiftR}\left(\chi_i^{2s-1}\right) \wedge \neg\chi_i^{2s-1}\right)^{l^j},$$

这可通过一个使用入度为 2 的或门和非门的常深度电路实现。

构造 $p(\cdot|x_t)$。为了实现上述过程，我们的 $p(\cdot|x_t)$ 只需复现从第 i 层到第 $(i+1)$ 层的计算。请注意，$p(\cdot|x_t)$ 的构造在所有迭代中都相同。要模拟的电路层由 $\mathcal{F}$ 控制；根据上一段所述，它从 x_t 中的掩码模式提取 i 的信息。

将计算从第 i 层到第 $(i+1)$ 层输出的电路记为 C_i：$\mathcal{V}^w \times \{0,1\}^{r^{i+1}} \to \{0,1\}^{w^{i+1}}$。[2] 这里，$r^{i+1}$ 表示第 $i+1$ 层中的随机比特数。那么，$p(\cdot|x)$ 的输出为：对于 $s^{i+1} \le j \le s^{i+2}-1$

$$\chi_{i+1}^{2j-1} = 0, \chi_{i+1}^{2j} = C_i\left(x_i^{s^i:s^{i+1}-1}, R_i\right)^j, R_i \sim U^{r^{i+1}}.$$

未指定其他 j 的输出位，因为其他位置未被解除掩码，因此无关紧要。最后，我们需要确认 $p(\cdot|x_t)$ 在不同位置的输出彼此独立。请注意，在 C_i 中，随机比特 R_i 仅用于生成第 $(i+1)$ 层中的随机比特。因此，R 的每一位在第 $(i+1)$ 层中都会变成随机位，且不会参与任何其他输出位的计算。因此，构造的 $p(\cdot|x_t)$ 确实实现了逐项独立输出分布。 □

在该构造中，我们必须记录中间计算结果（序列 u），因为每次应用 p 只能模拟一个常深度电路。同时，尽管只需使用一次，我们仍无法在不进行重新掩码或修订的情况下清除先前的中间结果。因此，虽然时间复杂度易于实现，该构造仍会占用大量内存。

这一结果表明，原则上，扩散语言模型能够比其自回归模型对应物更快地生成分布，并且在快速并行生成方面具有巨大潜力。然而在实践中，能够在 $O(d)$ 轮而非 $O(N)$ 轮内完成采样，并不意味着我们在实际应用中就能降低计算量或缩短相较于自回归模型的实际运行延迟。举例来说，我们比较了采用 KV缓存的自回归 Transformer 与由相同规模 Transformer 实现的 DLM，在一个解码步骤中的计算成本和实际运行延迟。在每个注意力层中，自回归模型只需计算一个位置的注意力输出，而 DLM 则需要计算所有位置的注意力输出。因此，DLM 所需的浮点运算次数是自回归模型的 $O(L)$ 倍。DLM 对 L 词元的计算可以并行进行，因此如果硬件支持足够高程度的并行，两者的实际运行延迟相同。然而，如果硬件不支持足够的并行计算，DLM 的实际运行延迟可能会更长。

3.2 带思维链和重新掩码或修订的 DLM

通过重新掩码或修订，我们可以擦除不再需要的中间结果。在本部分中，我们将展示，借助重新掩码或修订，我们不仅能够以可能的最少顺序步骤从任意分布中进行采样，还能将内存用量降至最低。

定理 3.2。设 C 是一个深度为 d、宽度为 w、具有 m 个输出位、n 个输入位和 $R \sim U^r$ 个随机输入位的电路。存在一个带有思维链和重新掩码的 ***DLM*** M，其长度为 $L = 2w + 2\lceil \log(d+1) \rceil$，使得 $M(q)$ 的分布与 $C(q, R)$相同。此外，解码步数为 $d+1$，而 p、$\mathcal{F}$、$\mathcal{G}$ 均可由深度为 $O(\log d)$ 的电路表示。

一些有用的函数。同样地，我们首先介绍一些对构造有用的函数，然后展示如何在常数深度下实现它们。

- 函数 bi n: $\mathbb{N} \to \{a,b\}^{\lceil \log(d+1) \rceil}$ 是一个将自然数转换为二进制形式的函数。

[2]为简化记号 ，这里假设电路接收来自 $\mathcal{V}$ 的词元，并将其映射为二进制表示 n
然后再传递给 t 实际电路

- 布尔函数 ADD： $\mathcal{V}^d \to \{0,1\}^n$ 定义为输出等于输入所表示的二进制数加一的函数。当输入为 b^d时，输出为 0^d。
- 布尔函数 识别$_i$： $\{a,b\}^d \to \{0,1\}$ 其中，整数 $i < 2^d$ 定义为：当且仅当输入为 bin(i) 时输出 1。

注意，由电路复杂度中的一个著名结果 [Brent 和 Kung,1982] 可知，ADD 属于 NC^1, 。这意味着，具有 d 个输入的与逻辑可以通过入度为 2 的与门，在深度为 $O(\log d)$ 的电路中实现。IDENTIFY$_i$ 也同样如此，这可由其另一种定义推出：

$$\mathsf{IDENTIFY}_i(x) = \bigwedge_{j=1}^{d} \left(\chi^j \wedge \mathsf{bin}(i)^j\right) \vee \left(\neg\chi^j \wedge \neg\mathsf{bin}(i)^j\right).$$

注意， $O(\log d)$ 相对于 n 仍然是常数，因此使用这两个电路不会违反我们对常深度电路的限制。

定理 *3.2* 的证明。 在此证明中，我们采用与定理 3.1 相同的记号，因为证明策略类似。同样，我们假设所有输入顶点都位于第一层，所有输出顶点都位于最后一层，并且第一层中没有随机位。此外，不失一般性，我们假设 d 是偶数。

在描述生成过程之前，我们首先将 x_i 划分为两个块，每个块的长度为 $w + d^*$。其中 $d^* = \lceil \log(d+1) \rceil$。使用 d^* 位，我们可以记录层索引。每个块用于记录某一层的输出，以及该层索引的二进制编码。这两个块交替使用，用于存储当前层的输出和下一层的输出。与定理 3.1 中的构造类似，我们在每次迭代中模拟单层计算。

生成过程。 现在，我们正式描述希望实现的生成过程。与之前一样，我们使用 x 的下标表示迭代索引。开始时，我们有 $x_0 = q\mathrm{M}^{L-n}$。对于$0 < i \le d$，我们设

$$\begin{aligned} x_i &= u^{s^i:s^{i+1}-1}\mathrm{M}^{w-w^i}\mathsf{bin}(i)\mathrm{M}^{w+d^*} && \text{当 } i \text{ 为奇数时,} \\ x_i &= \mathrm{M}^{w+d^*}\mathsf{bin}(i)\mathrm{M}^{w-w^i}u^{s^i:s^{i+1}-1} && \text{当 } i \text{ 为偶数时,} \end{aligned}$$

在每次迭代中，我们根据当前层的输出计算下一层的输出，并将其写入掩码块。随后，我们重新对该块进行掩码处理，记录当前层的输出，为下一次迭代的输出预留空间。最后，DLM 输出 $o = x_d^{L-m+1:L}$ 。

一种实用的多路复用器构造。 在深入探讨构造细节之前，先介绍一种反复出现的结构：我们需要构造一个电路，根据当前层索引 i选择使用哪个电路，即一个多路复用器。形式化地说，给定具有单个输出的电路 $D_1, \ldots, D_d$，我们希望构造一个电路 D，使其输出 $D(x,\mathsf{bin}(i)) = D_i(x)$。由于我们可以构造.....，因此可以通过增加电路深度 $O(\log d)$ 和 $O(\mathrm{poly}(d))$ 个逻辑门来实现这一点

$$D(x, \mathsf{bin}(i)) = \bigvee_{k=1}^{d} \mathsf{IDENTIFY}_k(i) \wedge D_i(x). \qquad (\star)$$

这里，具有 d 个输入的或逻辑可以通过入度为 2 的或门实现，深度为 $O(\log d)$；恒等门的深度为 $O(\log d)$。在此证明中，bin(i) 是 $x^{w+1:w+d^*}$ 或 $x^{w+d^*+1:\ w+2d^*}$，取未被掩码的那个。形式化地，对于 $j = 1,\ldots,d^*$，

$$\mathsf{bin}(i)^j = \left(\chi^1 \wedge \chi^{2(w+d+j)}\right) \vee \left(\neg\chi^1 \wedge \chi^{2(w+j)}\right).$$

因此，在 证明的其余部分中，当我们需要多路复用器时，只需构造 D_i 个独立的 *ly.*

构造 $\mathcal{F}$，$\mathcal{G}$ **和** $p(\cdot|x_t)$。我们首先构造 $\mathcal{F}$。当 $i=0$ 时，我们需要未掩码位置集合$\mathcal{S}_0=\{w+1,\cdots,w+d^*\}$。对于 $0<i<d$，我们需要未掩码位置集合

$$\begin{aligned}&\mathcal{S}_i=\{w+d^*+1,\ldots,w+2d^*,L-w^{i+1}+1,\ldots,L\} && \text{当 } i\text{为奇数时，}\\&\mathcal{S}_i=\{1,\ldots,w^{i+1},w+1,\ldots,w+d^*\} && \text{when } i \text{ is even.}\end{aligned}$$

形式化地，设 $D_i(x)=\mathbb{1}\{x\in\mathcal{S}_i\}$，那么我们可以使用上文构造的多路复用器来实现所需的$\mathcal{F}$。在构造 $\mathcal{G}$ 和 p时，多路复用器的用法也类似。

构造 $\mathcal{G}$ 与构造 $\mathcal{F}$类似。当 $i=0$时，$\mathcal{T}=\varnothing$。对于 $0<i<d$，

$$\begin{aligned}&\mathcal{T}=\{1,\ldots,w+d^*\} && \text{如果 } i\text{为奇数，}\\&\mathcal{T}=\{w+d^*+1,\ldots,L\} && \text{如果 } i\text{为偶数。}\end{aligned}$$

最后，我们构造 p。与定理 3.1 的证明一样，我们只需构造 $\mathcal{S}$内部的输出位置。同样，我们用 C_i 表示从第 i 层的词元输出到第 $(i+1)$ 层比特输出的电路。当 $i=0$时，我们有

$$x^{w+1:w+d}=\mathsf{bin}(1).$$

对于 $0<i\le d$ 且 i 为奇数，我们有

$$\chi_{i+1}^{2j-1}=0,\chi_{i+1}^{2j}=\begin{cases}C_i\left(x_i^{1:w^i},R_i\right)^{j-L+w^{i+1}}, & \text{if } L-w^{i+1}+1\le j\le L\\ \mathsf{ADD}\left(x_i^{w+1:w+d^*}\right)^{j-w-d^*}, & \text{if } w+d^*+1\le j\le w+2d^*\end{cases},R_i\sim U^{w^{i+1}}.$$

这模拟了从第 i 层到第 $(i+1)$ 层的电路计算，并递增迭代索引。类似地，对于 $0<i\le d$ 且 i 为偶数，我们有

$$\chi_{i+1}^{2j-1}=0,\chi_{i+1}^{2j}=\begin{cases}C_i\left(x_i^{L-w^i+1:L},R_i\right)^{j}, & \text{if } 1\le j\le w^{i+1}\\ \mathsf{ADD}\left(x_i^{w+d+1:w+d+d^*}\right)^{j-w}, & \text{if } w+1\le j\le w+d^*\end{cases},R_i\sim U^{w^{i+1}}.$$

□

长度 L 中的 $\lceil\log(d+1)\rceil$ 项源于这样一个事实：我们不再能从掩码模式推断出迭代步，因此需要额外的空位来记录当前迭代步。对于带修订的扩散语言模型，证明更简单，因为我们可以直接用 x_{i+1} 重写 x_i，而不必交替使用两块内存。定理陈述如下：

定理 3.3. 设 C 是一个深度为 d、宽度为 w 的电路，具有 m 个输出位、n 个输入位和 $R\sim U^r$ 个随机输入位。存在一个带思维链和修订的 ***DLM*** M，其长度为 $L=w+\lceil\log(d+1)\rceil$，使得 $M(q)$ 的分布与 $C(q,R)$ 相同。此外，解码步数为 $d+1$，且 p、$\mathcal{F}$ 均可表示为深度为 $O(\log d)$ 的电路。

4 D通过重新掩码或修订，语言模型的表达能力更强 n

在上一节中，我们展示了尺寸适当的扩散语言模型能够高效模拟布尔电路的生成过程，并且启用重新掩码或修订可以显著减少所需的序列长度 L。自然会引出这样一个问题：对于固定 L的扩散语言模型，重新掩码或修订能否使我们通过常数次生成步数生成新的分布？本节将通过展示在生成下一段中引入的一个特定分布方面存在的分离，对这个问题给出肯定的回答。

为了证明分离，我们只需考虑更简单的情形：DLM 不接收输入（$q=\varnothing$），而是从无条件分布中采样。设 $\oplus$ 为模2加法。也就是说，

当且仅当 $z_1 + z_2 = b$（模 2）时，$z_1 \oplus z_2 = b$ 对应于 $b \in \{0,1\}$。众所周知，$\oplus$ 可以表示为与门和或门的组合：

$$\chi \oplus \psi = (\chi \vee \psi) \wedge \neg(\chi \wedge \psi) \quad \text{for any } \chi, \psi \in \{0,1\}$$

我们将奇偶函数定义如下。

定义 2。 布尔函数 奇偶函数： $\{0,1\}^n \to \{0,1\}$ 定义如下。对于任意 $\chi \in \{0,1\}^n$ ，

$$\mathsf{PARITY}(\chi) = \bigoplus_{i=1}^{n} \chi_i.$$

众所周知，奇偶函数不属于 AC^0 [Furst 等人，1984年，H˚astad， 1986]。我们使用以下分布证明这一分离结果。

定义 3。 分布 $\mathcal{D}_n^{\oplus}$ 定义为所有偶校验（即 $\{(z_1, \ldots, z_n) | \text{奇偶函数}(z) = 0\}$）的 n 位字符串上的均匀分布。

本节中，我们假设扩散语言模型由 AC^0 电路构成。请注意，嵌入大小为多项式级别的常数精度、常数深度 Transformer 可以由 AC^0 电路模拟 Li 等人 [2024], ，而当前大多数 DLM 实现都基于 Transformer Nie 等人 [2025], Yang 等人 [2025]。此外，采用硬注意力机制的 Transformer（其中每个注意力块只能关注得分最高的位置）也可以由 AC^0 电路模拟 Hao 等人 [2022]。因此，我们认为这些结果反映了当前架构的真实表达能力。特别是，对于从$\mathcal{D}_n^{\oplus}$中采样的任务，上文描述的采样算法非常简单，可以轻松地由 Transformer 模拟。

4.1 上界

在本节中，我们将展示带修订的扩散语言模型可以在两步内从 $\mathcal{D}_n^{\oplus}$ 中采样字符串。

定理 4.1。 带修订的扩散语言模型和 $L = n$ 可以在两步内生成 $\mathcal{D}_n^{\oplus}$，使得预测器 p 和函数 $\mathcal{F}$ 均由NC^0 电路表示。

证明。 与定理 3.1 的证明类似，我们首先描述希望实现的生成过程，然后展示电路构造。我们还沿用以下记号： x_i 表示第 i 个生成步骤之后的序列。

我们的目标是生成 $(z^1, \ldots, z^n) \sim \mathcal{D}_n^{\oplus}$。令 $(y^1, \ldots, y^n) \in \{0,1\}^n$ 定义为 $y^i = \bigoplus_{j=1}^{i} z^j$。容易验证，$(y^1, \ldots, y^{n-1}) \sim U^{n-1}$，且 $y^n = 0$。此外，根据定义，对于$i = 2, \ldots, n$， $z^1 = y^1$ 和 $z^i = y^i \oplus y^{i-1}$，因此我们可以从 y 轻松恢复 z。于是，我们可以通过以下过程生成 $\mathcal{D}^{\oplus}$：

$$x_0 = \mathrm{M}^n, \quad x_1 = y, \quad x_2 = z.$$

上述过程很容易通过带修订的扩散语言模型实现：第一步，我们生成 $(y^1$、.....、$y^{n-1}) \sim U^{n-1}$，以及 $y^n = 0$。第二步，我们计算 $z^1 = y^1$ 和 $z^i = y^i \oplus y^{i-1}$ 对于$i = 2$、.....、n。下文将详细介绍电路构造。

函数 $\mathcal{F}$ 始终输出长度为 n 的全 1 向量。形式上，对所有 $\chi \in \{0,1\}^{2n}$， $f(\chi)^j = 1$。

现在我们来描述 $p(\cdot|x_t)$ 的电路。我们首先需要根据第一个词元是否被掩码（χ^1）来识别当前处于 $i = 0$ 还是 $i = 1$，然后执行相应的计算。因此，对于每个 $i \in \{0,1\}$，

$$R \sim U^{n-1},$$

$$\chi_{i+1}^{2j-1} = 0, \chi_{i+1}^{2j} = \begin{cases} \left(\chi_i^1 \wedge R^1\right) \vee \left(\neg\chi_i^1 \wedge \chi_i^2\right) & \text{if } j = 1 \\ \left(\chi_i^1 \wedge R^j\right) \vee \left(\neg\chi_i^1 \wedge \left(\chi_i^{2j} \oplus \chi_i^{2j-2}\right)\right) & \text{if } 1 < j < n \\ \left(\chi_i^1 \wedge 0\right) \vee \left(\neg\chi_i^1 \wedge \left(\chi_i^{2j} \oplus \chi_i^{2j-2}\right)\right) & \text{if } j = n \end{cases}.$$

上述内容表明，$p(\cdot|x_t)$ 可以通过常深度电路轻松实现（上述常深度电路将 χ_i 映射到 χ_{i+1}，适用于 $i \in \{0,1\}$）。此外，请注意，χ 的每个位置最多依赖一个随机输入比特，因此在给定 χ_i 的条件下，不同位置的输出相互独立。至此，证明完成。

□

对于带重新掩码的扩散语言模型，也有类似结果，如下所述。

定理 4.2。带重新掩码的扩散语言模型和 $L = n$ 可以在 O步内采样 $\mathcal{D}_n^{\oplus}$，使得相应的电路属于AC^0。这里，我们还允许解掩码位置预测器 $\mathcal{F}$ 依赖于生成过程中的步数。

证明。下面我们将描述该采样算法，同时说明如何使用 AC^0 预测器 p 和解掩码位置预测器 $\mathcal{F}$来实现它。由于这里的预测器 p 和 $\mathcal{F}$ 也可以依赖于生成过程中的步数，因此每一步我们只需使用一个新的 p 和 $\mathcal{F}$。

为简单起见，假设 n 是偶数。我们分两个阶段进行，每个阶段包含 $O(1)$ 步。首先，将 n位输入依次划分为 $n/2$ 2位的块。在第一阶段的每个偶数位置$2i$，我们将采样一位 x^i，使得位置 $2i-1$ 和 $2i$ 上的位的奇偶性应为 x^i。注意，所有这些 x^i 都应服从分布 $\mathcal{D}_{n/2}^{\oplus}$。接着，在第二阶段，我们将对每个 2 位块（即 $(2i-1,\ 2i)$，其中 $i = 1,\ .....,\ n/2$）进行均匀随机采样，使其奇偶性为 x^i。由此得到所需的分布 $\mathcal{D}_n^{\oplus}$。

阶段 1：对 $n/2$ 对的奇偶函数进行采样。在这一阶段，我们分两步进行。第一步，在所有奇数位置生成 $y^1, y^2, \ldots, y^{n/2}$，使得 $(y^1, \ldots, y^{n/2-1})$ 均匀随机且$y^{n/2} = 0$。然后，第二步，在所有偶数位置计算 $x^i = y^i \oplus y^{i-1}$，其中 $i \in \{2, \ldots, n/2\}$，并且 $x^1 = y^1$。完成这两步后，我们生成了 $(y^1, x^1, y^2, x^2, \ldots, y^{n/2}, x^{n/2})$。前两步可以轻松地使用 AC^0 预测器 p 和 $\mathcal{F}$ 实现。

现在，按照定理 4.1 的证明，$(x^1, \ldots, x^{n/2})$ 服从分布 $\mathcal{D}_{n/2}^{\oplus}$，也就是说，在奇偶函数为偶数的条件下，它们是均匀随机的。

阶段 2：采样所有 $n/2$ 已知奇偶性的数对。接下来，我们重新掩码所有奇数位置（所有的y^i）。回想一下，我们将 x^i 解释为第 i个 2 位块的奇偶性。从现在起，我们的目标是对第 i个 2 位块（$2i-1,\ 2i$）进行均匀随机采样，使其奇偶性为 x^i，从而得到所需的分布 $\mathcal{D}_n^{\oplus}$。

具体而言，对于每个形如 M0 的块，我们将以相同的概率将其替换为来自 $\{00, 11\}$ 的新字符串。对于每个形如 M1 的块，我们将以相同的概率将其替换为来自 $\{01, 10\}$ 的新字符串。我们将分两个子阶段完成，每个子阶段包含若干步骤。

阶段 2.a：处理形如 M0的块。在第一个子阶段中，我们对所有形如 M0 的块随机采样第一个比特。然后，对于每个形如 10 的块，我们先重新掩码 0，得到 1M，再将最后一个 M 改为 1，得到 11。这样，每个 M0 块都会以相同的概率被替换为 00 或 11，正如所需。

阶段 2.b：处理形如 M1 的块。在第二个子阶段中，对于所有形如 M1 的块，我们随机采样第一个比特。然后，对于每个形如 11 的块，我们先重新掩码第二个 1，得到 1M，再将最后一个 M 改为 0，得到 10。这样，每个 M1 块就会以相同的概率被替换为 01 或 10，符合预期。

不难看出，如果预测器和解掩码位置预测器也能依赖生成过程中的步数，那么这两个子阶段就可以轻松地使用 AC^0 预测器 p 与门 $\mathcal{F}$ 来实现。至此，证明完成。

□

4.2 下界

在本节中，我们将展示，即使允许预测器 p 与 $\mathcal{F}$ 为 AC^0 电路，不进行重新掩码或修订的扩散语言模型也无法在 $\mathcal{D}_n^{\oplus}$ 中于 $O(1)$步生成。在此基础上，我们确立了进行修订/重新掩码与不进行修订/重新掩码的扩散语言模型之间的强分离。

鉴于如前所述，奇偶函数不属于 AC^0，这一结果应当是直观的。在扩散语言模型生成过程中，只能修改掩码词元。当只剩下一个掩码词元时，扩散语言模型必须计算所有其他比特的奇偶性，而这是不可能的。然而，由于扩散语言模型可以在每次迭代中同时解码多个词元，要精确地形式化这一直觉颇具挑战。因此，我们需要如下所述的针对 AC^0 的奇偶函数平均情形下界。

定理 4.3 (H˚astad [2014])。 设 d 为常数，且 $C\colon \{0,1\}^n \to \{0,1\}$ 是一个深度为 d、规模为 S的 AC^0 电路。存在一个仅依赖于 d 的常数 c_d，使得

$$\Pr_{\chi\sim U^n}[C(\chi) = \mathsf{PARITY}(\chi)] < 1/2 + 2^{-c_d\cdot n/(\log S)^{d-1}}.$$

通俗地说，该定理表明，任何 AC^0 电路计算奇偶函数的能力都不会比随机猜测好太多。此外，该定理还可以直接推广到概率 AC^0 电路，如下所示。

推论 4.4。 设 d 为常数。设 $C\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ 为一个深度为d 、规模为 S的概率 AC^0 电路。存在一个仅依赖于 d 的常数 c_d，使得

$$\Pr_{\chi\sim U^n, R\sim U^m}[C(\chi, R) = \mathsf{PARITY}(\chi)] < 1/2 + 2^{-c_d\cdot n/(\log S)^{d-1}}.$$

现在，我们可以给出扩散语言模型生成的下界了。

定理 4.5。 当预测器 $p(\cdot|x)$ 和 $\mathcal{F}$ 均属于 AC^0时，具有 $L = n$ 的扩散语言模型（不进行重新掩码或修订）无法在 O 个步骤内采样 $\mathcal{D}_n^{\oplus}(1)$。

证明。 反设存在矛盾：对于任意整数 n，都存在一个 DLM，能在ℓ 步内采样 $\mathcal{D}_n^{\oplus}$，并且预测器 p 和 $\mathcal{F}$ 都属于 AC^0。

我们先引入一些记号。令 $\ell = \Theta(1)$为任意常数。我们将 DLM 的规模定义为预测器 p （对所有 i）和 $\mathcal{F}$的电路规模的最大值。由于预测器 p 和 $\mathcal{F}$ 都属于 AC^0,，只要 n 足够大，DLM 的规模就至多为 n^c，其中 $c > 1$为某个常数。

归约到一个辅助断言。 在上述假设下，我们可以证明以下断言，它对于证明我们的定理至关重要:

对于每个 $i \in \{0, 1, \ldots, \ell\}$，都存在一个 $O(n^c)$规模的 *DLM*，满足 $L = n_i$，并能在 $\ell - i$ 步内采样 $\mathcal{D}_{n_i}^{\oplus}$，
$$where\ n_i \geq n^{2^{-i}}.$$

在证明该断言之前，我们先说明它如何帮助我们完成证明。该陈述意味着一个 n^c规模的 DLM 能在 0 步内采样 $\mathcal{D}_{n^{2^{-\ell}}}^{\oplus}$，这显然会导致矛盾，因为 0 步内不可能采样任何词元。

命题证明的记号。 现在我们通过归纳法来证明该命题。当$i = 0$ 时，结论直接由我们的假设得出。现在，假设该命题对 i成立，并证明其对$i + 1$成立。这里我们使用与定理 3.1的证明类似的记号。令 $\mathcal{S}_1 = \mathcal{F}(\mathrm{M}^{n_i})$ 表示第一次迭代中要取消掩码的位置集合，并令 $x_2^{\mathcal{S}_1} \in \{a, b\}^{|\mathcal{S}_1|}$ 表示第一次迭代中采样得到的词元序列。由于 $\mathcal{D}_{n_i}^{\oplus}$ 在 $\mathcal{S}_1$ 上的边缘分布为 $U^{|\mathcal{S}_1|}$，且没有修订或重新掩码的 DLM 无法改变已取消掩码的词元，因此必有 $x_2^{\mathcal{S}_1} \sim U^{|\mathcal{S}_1|}$。此外，令 $m = n_i - |\mathcal{S}_1|$ 表示 x_2中的掩码词元数量。由于 $U^{n_i} \neq \mathcal{D}_{n_i}^{\oplus}$，必有 $m > 0$。

命题证明的高层思路。在深入细节之前，我们先介绍整体思路。核心论证是证明，对于足够大的 n_i，$m \geq \sqrt{n_i}$。为此，我们将证明可以构造一个 $\overline{\mathsf{AC}}^0$ 电路，以 $\chi \in \{0,1\}^{|\mathcal{S}_1|}$的精度近似奇偶函数 (χ)，并且 m 越小，近似就越准确。定理 4.4 中的电路下界便能给出关于 m 的下界，因为 AC^0 电路无法很好地近似奇偶函数。此外，我们可以构造一个 DLM

使用一个在 $\ell - i$ 步内采样 $\mathcal{D}^{\oplus}_{n_i}$ 的 DLM，在 $\ell - i - 1$ 步内采样 $\mathcal{D}^{\oplus}_{m}$。结合 $m \geq \sqrt{n_i}$，我们就完成了归纳步骤。

主张证明中的关键构造。</style>近似奇偶函数 (χ) 的电路 C 的构造如下：

- 使用对应关系 $0 \mapsto a, 1 \mapsto b$，通过 χ 将集合 $\mathcal{S}_1$ 中的位置设置在 x_2 中。将 x_2 的其他位置设为 M。
- 从 x_2 开始模拟 DLM，运行 $\ell - i - 1$ 步，并获取 $[n_i] \setminus \mathcal{S}_1$ 上的输出词元。
- DLM 输出序列 $x_{\ell-i}$。电路 C：如果 $x_{\ell-i}^{[n_i]\setminus\mathcal{S}_1}$ 全部为 a，则输出 0；否则输出一个随机位。

该电路属于 AC^0，因为每次迭代我们都会在 x 和 $\ell = O(1)$ 上应用一个 AC^0 电路。此外，判断输出是否全部为 a 可以通过一个具有无界扇入的与门来实现。根据 $\mathcal{D}^{\oplus}$ 的定义，输出序列的奇偶函数 $\mathrm{PARITY}(x_{l-i}) = 0$，因此我们知道 $\mathrm{PARITY}\left(x_{\ell-i}^{\mathcal{S}_1}\right) = \mathrm{PARITY}\left(x_{\ell-i}^{[n_i]\setminus\mathcal{S}_1}\right)$。因此，所构造的电路平均而言计算奇偶函数的效果略优于随机猜测：当 $x_{\ell-i}^{[n_i]\setminus\mathcal{S}_1}$ 全部为 a 时，它能准确预测奇偶函数，因为我们必有 $\mathrm{PARITY}\left(x^{\mathcal{S}_1}\right) = 0$。

现在我们计算 C 在输入 $\chi \sim U^{|\mathcal{S}_1|}$上正确计算奇偶函数的概率。容易验证，在 $x^{\mathcal{S}_1}$ 的奇偶函数值为零的条件下，$x^{[n_i]\setminus\mathcal{S}_1}$ 的分布为 $\mathcal{D}^{\oplus}_{n_i-|\mathcal{S}_1|}$。因此，$x_{\ell-i}^{[n_i]\setminus\mathcal{S}_1}$ 为全 a 的概率是 2^{-m}。所以，C 正确输出奇偶函数值的概率为

$$2^{-m} \cdot 1 + (1 - 2^{-m}) \cdot \frac{1}{2} = \frac{1}{2} + 2^{-(m+1)}.$$

注意，C 具有常数深度和大小 n_i^c，应用定理 4.4 可知，我们必须有

$$\frac{1}{2} + 2^{-(m+1)} \leq \frac{1}{2} + 2^{-\frac{c_d n_i}{(\log n_i^c)^{d-1}}} \Leftrightarrow m > \frac{c_d(n_i - m)}{c^{d-1}\left(\log(n_i - m)\right)^{d-1}} - 1.$$

对于足够大的 n_i，我们有 $m > \sqrt{n_i}$。[3] 最后，由于 DLM 从 $\mathcal{D}^{\oplus}_{n_i}$中完美采样，当将 w_1 设为全零时，它会在 $[n_i] \setminus S_1$中的位置上采样 $\mathcal{D}^{\oplus}_{m}$。这样，我们得到一个在 $\ell - i - 1$ 步内从 $\mathcal{D}^{\oplus}_{m}$ 中采样的 DLM，从而完成归纳法。□

注。该定理的证明包含一个过程，可将 AC^0 DLM 的常数轮生成转换为 AC^0 电路。人们可能会提出以下问题：为什么不能使用相同的过程，将定理 4.1 和 4.2 的证明中的构造转换为 AC^0 电路，使其在单步内采样 $\mathcal{D}^{\oplus}_{n}$？这难道不说明无需修订/重新掩码的 DLM 可以在单步内采样 $\mathcal{D}^{\oplus}_{n}$ 吗？事实上，我们确实可以得到这样的 AC^0 电路，但这样的 AC^0 电路不是有效的 DLM。有效的 DLM 预测器 p 满足 $p(x|x_t) = \prod_{i=1}^{L} p^i\left(x^i|x_t\right)$，因此不可能在单步内采样具有相关性的分布。

我们的理论分析表明，设计包含修订的前向过程，并收集包含修订/重新掩码的训练数据，是扩散语言模型的一项非常重要的设计要素，其潜力远大于不包含修订或重新掩码的扩散语言模型。

5 结论

本文为扩散语言模型（DLMs）作为并行采样器奠定了理论基础，表明借助思维链，它们能够达到最优顺序步骤数；相比之下，自回归模型的成本会随电路规模增长。此外，通过重新掩码或修订，它们还能同时

[3] 由于归纳法在 ℓ 步后结束，因此总能通过足够大的 n 实现足够大的 n_i。

达到最优空间需求。此外，我们证明，重新掩码和修订不仅能降低内存需求，使其随电路宽度增长，还能严格提升表达能力，从而实现对奇偶函数等分布的常数步采样，而标准 DLM 无法实现这些分布。这些结果表明，DLM 是最高效的并行采样器，并凸显了重新掩码和修订是释放其全部潜力的关键机制。

参考文献

Jacob Austin、Daniel D. Johnson、Jonathan Ho、Daniel Tarlow 和 Rianne van den Berg。离散状态空间中的结构化去噪扩散模型。载于 M. Ranzato、A. Beygelzimer、Y. Dauphin、P.S.Liang 和 J. Wortman Vaughan 编，《神经信息处理系统进展》，第 34 卷，17981–17993 页。Curran Associates, Inc.，2021 年。网址：https://proceedings.neurips.cc/paper_files/ paper/2021/file/958c530554f78bcd8e97125b70e6973d-Paper.pdf。

Brent 和 Kung。并行加法器的一种规则布局。《*IEEE* 计算机学报》，C-31(3):260–264，1982 年。doi: 10.1109/TC.1982.1675982。

Merrick Furst、James B. Saxe 和 Michael Sipser。奇偶函数、电路与多项式时间层次。数学系统理论，17(1):13–27，1984年。

Yiding Hao、Dana Angluin 和 Robert Frank。硬注意力变换器的形式语言识别：电路复杂度视角。计算语言学协会会刊，10：800–810，2022年。doi：10.1162/tacl a 00490。URL https://aclanthology.org/2022.tacl-1.46/。

J H˚astad。小深度电路的近乎最优下界。载于第十八届 *ACM* 计算理论研讨会论文集，STOC ’86，第 6–20 页，美国纽约州纽约，1986年。计算机协会。ISBN 0897911938。doi：10.1145/12130.12132。URL https://doi.org/10. 1145/12130.12132。

Johan H˚astad。奇偶函数与小深度电路的相关性。*SIAM* 计算杂志，43(5)：1699–1708，2014年。doi: 10.1137/120897432。URL https://doi.org/10.1137/120897432。

Zhengfu He、Tianxiang Sun、Qiong Tang、Kuanning Wang、Xuan-Jing Huang 和 Xipeng Qiu。DiffusionBERT：使用扩散模型改进生成式掩码语言模型。载于 计算语言学协会第*61*届年会论文集（第*1*卷：长篇论文），第4521–4534页，2023年。

Jonathan Ho、Ajay Jain 和 Pieter Abbeel。去噪扩散概率模型。载于 H. Larochelle、M. Ranzato、R. Hadsell、M.F. Balcan 和 H. Lin（编），神经信息处理系统进展，第33卷，第6840–6851页。Curran Associates, Inc.，2020年。网址：https://proceedings. neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf。

Samar Khanna、Siddhant Kharbanda、Shufan Li、Harshit Varma、Eric Wang、Sawyer Birnbaum、Ziyang Luo、Yanis Miraoui、Akash Palrecha、Stefano Ermon 等。Mercury：基于扩散的超高速语言模型。*arXiv* 预印本 *arXiv:2506.17298*，2025年。

Jaeyeon Kim、Kulin Shah、Vasilis Kontonis、Sham Kakade 和 Sitan Chen。以最坏情况训练，为最佳情况规划：理解掩码扩散中的词元排序。*arXiv* 预印本 *arXiv:2502.06768*，2025年。

Zhiyuan Li、Hong Liu、Denny Zhou 和 Tengyu Ma。思维链赋能 Transformer 解决本质上的串行问题。载于第十二届国际学习表征会议，2024年。网址：https://openreview.net/forum?id=3EWTEy9MTM。

Aaron Lou、Chenlin Meng 和 Stefano Ermon。通过估计数据分布的比率进行离散扩散建模。载于第 *41* 届国际机器学习会议论文集，第 32819–32848 页，2024年。

William Merrill 和 Ashish Sabharwal。并行性权衡：对数精度 Transformer 的局限性。计算语言学协会会刊，11:531–545，2023年。doi：10.1162/tacl a 00562。网址：https://aclanthology.org/2023.tacl-1.31/。 - -

Shen Nie、Fengqi Zhu、Zebin You、Xiaolu Zhang、Jingyang Ou、Jun Hu、Jun Zhou、Yankai Lin、Ji-Rong Wen 和 Chongxuan Li。大型语言扩散模型。*arXiv* 预印本 *arXiv:2502.09992*，2025年。

Subham Sekhar Sahoo、Marianne Arriola、Yair Schiff、Aaron Gokaslan、Edgar Marroquin、Justin T Chiu、Alexander Rush 和 Volodymyr Kuleshov。简单且有效的掩码扩散语言模型。载于 A. Globerson、L. Mackey、D. Belgrave、A. Fan、U. Paquet、J. Tomczak 和 C. Zhang 编，神经信息处理系统进展，第 37 卷，第 130136–130184 页。Curran Associates, Inc.，2024年。URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ eb0b13cc515724ab8015bc978fdde0ad-Paper-Conference.pdf。

Jascha Sohl-Dickstein、Eric Weiss、Niru Maheswaranathan 和 Surya Ganguli。使用非平衡热力学进行深度无监督学习。载于 Francis Bach 和 David Blei（编），第 *32* 届国际机器学习会议论文集，机器学习研究论文集第 37 卷，第 2256–2265 页，法国里尔，2015年7月7日至9日。机器学习研究论文集。网址：https://proceedings.mlr.press/v37/ sohl-dickstein15.html。

Yuxuan Song、Zheng Zhang、Cheng Luo、Pengyang Gao、Fan Xia、Hao Luo、Zheng Li、Yuehang Yang、Hongli Yu、Xingwei Qu 等。Seed 扩散：一种支持高速推理的大规模扩散语言模型。*arXiv* 预印本 *arXiv:2508.02193*，2025年。

H. Vollmer。电路复杂度导论：一种统一方法。理论计算机科学丛书。EATCS 丛书。Springer Berlin Heidelberg，1999年。ISBN 9783540643104。网址：https:// books.google.com/books?id=55ZTgOJs8bsC。

Jason Wei、Xuezhi Wang、Dale Schuurmans、MaartenBosma、Fei Xia、Ed Chi、Quoc V Le、Denny Zhou 等。思维链提示能够激发大型语言模型的推理能力。神经信息处理系统进展，35:24824–24837，2022年。

Zhihui Xie、Jiacheng Ye、Lin Zheng、Jiahui Gao、Jingwei Dong、Zirui Wu、Xueliang Zhao、Shansan Gong、Xin Jiang、Zhenguo Li 和 Lingpeng Kong。Dream-coder 7b：一种面向代码的开放式扩散语言模型，2025年。URL https://arxiv.org/abs/2509.01142。

Ling Yang、Ye Tian、Bowen Li、Xinchen Zhang、Ke Shen、Yunhai Tong 和 Mengdi Wang。Mmada：多模态大型扩散语言模型。*arXiv* 预印本 *arXiv:2505.15809*，2025年。

Jiacheng Ye、Zhihui Xie、Lin Zheng、Jiahui Gao、Zirui Wu、Xin Jiang、Zhenguo Li 和 Lingpeng Kong。Dream 7b：扩散大型语言模型。*arXiv* 预印本 *arXiv:2508.15487*，2025年。