arXiv:2603.05540v1 [cs.CL] 4 Mar 2026

注意力遇上可达性：语法约束的大语言模型解码中的结构等价与效率

Faruk Alpay，Bahçeşehir University 计算机工程系，土耳其伊斯坦布尔

Bilge Senturk 工业工程系，Bahçeşehir University，伊斯坦布尔，土耳其 bilge.senturk@bahcesehir.edu.tr

faruk.alpay@bahcesehir.edu.tr

2026年3月9日

摘要

语法约束解码（GCD）要求自回归语言模型生成符合上下文无关文法（CFG）所规定形式语言的输出。我们将GCD形式化为Transformer式下一词元分布与可达性预言机之间的耦合；该预言机运行于由CFG编译而来的下推系统之上，并构建统一的理论框架，同时探讨约束引擎的复杂度结构与受约束生成器的概率保真度。我们证明了预言机不变性定理：语言等价文法对任意前缀都会产生相同的下一词元可接受集合（因此也会产生相同的logit掩码），但其编译状态空间和在线歧义成本可能存在可证明的差异。我们给出了冗余非终结符委托下典范 $a^n b^n$ 语言的精确状态数膨胀，并引入一种从左至右的结构歧义成本（SAC），用于衡量每个词元带来的压缩解析森林增量。对于两个等价的 Σ^* 文法，我们通过对压缩解析森林密度给出紧确界证明：在右递归下，SAC每个词元为 $O(1)$；而在连接下，SAC每个词元为$\Theta(t^2)$，累计为 $\Theta(n^3)$。除渐近复杂度外，我们还提出另外三项理论贡献。第一，我们将SAC提升为与引擎无关的下界：对于特定的常量规模CFG族，任何可靠、检索高效且保持解析的在线掩码引擎，每个词元都必须付出$\Omega(t^2)$的计算量（累计为$\Omega(n^3)$）；这与经典的将CFG句法分析和矩阵乘法联系起来的复杂度结果 [34, 18]相呼应。第二，我们定义了文法的解码成本等价类，并证明在有界重写族中存在最小SAC代表，从而阐明典范低SAC范式的含义及其局限。第三，我们形式化了文法条件自回归过程：真实条件采样器 $p(\cdot \mid \tau(y) \in L)$ 可由Doob h-变换刻画；我们还证明了硬掩码解码的紧确失真界，该界以允许的后续词元之间的生存概率差异表示。最后，我们将这些结果与Transformer和混合专家（MoE）架构相结合，根据词汇表大小、活跃状态集合和束宽推导出解码时延包络，并将SAC扩展为经测量校准的预测性能模型，以支持实证验证和自动文法优化。

1 引言

自回归大语言模型（LLM）定义了如下形式的分布

$$p_\theta(y_{1:T}) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}), \tag{1}$$

其中 $y_t \in \mathcal{V}$ 是一个词汇标记，而 θ 是模型参数。然而，在许多部署场景中，所需的输出并非任意文本，而是结构化产物（例如 SQL、JSON 或编程语言片段）。语法约束解码（GCD）通过限制生成过程，使其仅生成能够令当前前缀 在形式语言规范下保持可完成的 词元（通常该规范是上下文无关文法），并即时拒绝不可接受的词元，从而解决这一问题 [28, 2]。结构化生成领域中广受欢迎的开源工具链也采用了相同的概念接口 [19, 13, 8, 36, 15]。

本文探讨 GCD 核心处的一种张力：语言等价是语义性的（从用户视角看，生成相同字符串的两种文法可以互换），但受约束解码执行的是一个特定编译后的识别器，其运行时取决于文法结构。我们的核心主张是：即使两个上下文无关文法生成完全相同的语言，它们也可能为从左到右解码引擎诱导出截然不同的内部搜索空间。这种差异并非语义性的，而纯粹是结构性的，并通过下推可达性表现出来。

贡献。我们做出六项主要贡献。

1. **下推可达性形式化**。我们将 GCD 形式化为神经模型与下推可达性预言机之间的耦合，把解码步骤建立在上下文无关语言解析与可达性理论 [16, 9, 27, 4] 之上。我们证明语言等价下的预言机不变性，并给出硬掩码解码的精确代数刻画。2. **状态空间膨胀界**。我们证明在冗余非终结符委托下 $a^n b^n$语言的精确控制状态膨胀计数，从而给出“引擎状态空间膨胀”这一具体、可机械检验的概念。3. **结构歧义成本（SAC）与紧致增长界**。我们将 SAC 引入为压缩解析森林增长的从左到右逐词元度量。我们证明：对于拼接式 Σ^* 文法，SAC 为每个词元 $\Theta(t^2)$（累积为 $\Theta(n^3)$），而对于等价的右递归文法则为 $O(1)$。4. **与引擎无关的下界**。我们定义一种解析保持、检索高效屏蔽引擎模型，并证明任何此类引擎在常数规模的上下文无关文法族上，每个词元都必须付出 $\Omega(t^2)$ 的工作量，从而使 SAC下界在该语义接口内成为无条件的。这与 Valiant/Lee 风格归约 [34,18] 互补，但并不相同。5. **解码成本等价类与规范低SAC形式**。我们将 $\equiv_{\text{dec}}$定义为语言等价与 SAC 等价的联合，并证明在任何有限有界重写族中都存在最小 SAC 文法代表。6. **语法条件自回归过程**。我们通过 Doob h-变换刻画真正的条件采样器 $p(\cdot \mid \tau(y) \in L)$，并针对硬掩码解码，依据可接受令牌之间的存活概率散布，导出尖锐的单步 KL 与全变差失真界。

我们进一步推导代数约束，描述可达性有效转移子集如何调制 Transformer 对数几率与 MoE路由 [35, 30, 10], ，并展示如何将 SAC 落地为一种预测性能模型，该模型依据真实推理栈的经验轨迹进行校准 [12, 6, 5]。

2 解码作为下推可达性

我们固定以下三项：(i) 定义在抽象终结符字母表上的上下文无关文法，(ii) 从模型词元到终结符串的分词器映射，以及 (iii) 将该上下文无关文法显式编译为下推自动机，其可达配置集决定允许的后续词元。

2.1 语言、前缀与分词

定义 1（上下文无关文法）。 上下文无关文法是一个元组 $G = (N, \Sigma, P, S)$，其中 N 是有限的非终结符集合， Σ 是有限的终结符集合， $P \subseteq N \times (N \cup \Sigma)^*$ 是写成 $A \to \alpha$ 的有限产生式集合， $S \in N$ 是开始符号。由 G 生成的语言记为 $\mathcal{L}(G) \subseteq \Sigma^*$。

定义 2（前缀闭包与单步扩展）。 对于任意语言 $L \subseteq \Sigma^*$，定义其前缀闭包

$$\mathrm{Pref}(L) \coloneqq \{u \in \Sigma^* \ : \ \exists v \in \Sigma^* \ \textit{such that} \ uv \in L\}.$$

将前缀 u 之后的后续终结符集 定义为

$$\mathrm{Next}_L(u) \coloneqq \{a \in \Sigma \ : \ ua \in \mathrm{Pref}(L)\}.$$

将结束信号 $\langle \mathrm{eos} \rangle$ 定义为在 u 之后可接受，当且仅当 $u \in L$。

定义 3 (分词器同态**)**。 令 $\mathcal{V}$ 为大语言模型词汇表。分词器同态是一个映射 $\tau : \mathcal{V} \to \Sigma^*$。对于词元序列 $y_{1:T} \in \mathcal{V}^T$，将其实现的终结符串定义为

$$\tau(y_{1:T}) \coloneqq \tau(y_1)\tau(y_2)\cdots\tau(y_T) \in \Sigma^*,$$

其中，并置表示 Σ^*中的连接。我们令 $\tau(\langle \mathrm{eos} \rangle) = \varepsilon$。

约束“模型输出属于 $\mathcal{L}(G)$”解释为 $\tau(y_{1:T}) \in \mathcal{L}(G)$。

2.2 下推自动机与上下文无关文法编译

定义 4 (非确定性下推自动机**)**。 非确定性下推自动机（*NPDA*）是一个元组

$$\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, \bot, F),$$

其中， Q 是有限控制状态集合， Γ 是有限栈字母表，其底部带有栈底标记 $\bot$； $q_0 \in Q$是初始状态， $F \subseteq Q$ 是接受状态集合，并且

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$$

是转移函数。配置是一个三元组 (q, u, γ)，其中 $q \in Q$, $u \in \Sigma^*$ 是未读输入，而 $\gamma \in \Gamma^*$ 是栈内容， $\bot$ 位于栈底。接受条件是到达终态，且输入为空、栈恰好为 $\bot$。

定义 5 (递归转移网络编译**)**。 给定 $G = (N, \Sigma, P, S)$，按如下方式定义 $\mathcal{A}_G$。对于每个非终结符 $A \in N$，创建控制状态 q_A^{in} 和 q_A^{out}。对于每个产生式$p = (A \to X_1 \cdots X_m) \in P$以及每个点位置 $i \in \{0, 1, \ldots, m\}$，创建一个控制状态 $q_{p,\ i}$。令

$$Q \coloneqq \{q_{\mathrm{start}}\} \cup \{q_A^{\mathrm{in}}, q_A^{\mathrm{out}} : A \in N\} \cup \{q_{p,i} : p \in P,\, i\}.$$

令栈字母表为 $\Gamma : = \{\bot\} \cup \{q_{p,i} : \ \ p \in P,\, i\}$。定义转移：

- 起始符： $\delta(q_{\mathrm{start}} \ \varepsilon, \bot) \ni (q_{\mathrm{in}} S, \bot)$。

- 产生式选择：对于每个产生式 $p = (A \to \cdots)$， $\delta(q_{\text{in}A}, \varepsilon, \gamma) \ni (q_{p,0}, \gamma)$ 对所有$\gamma \in \Gamma$。
- 退出： $\delta(q_{p,m}, \varepsilon, \gamma) \ni (q_{\text{输出}A}, \gamma)$。
- 终结符：若 $X_i \in \Sigma$，则 $\delta(q_{p,i-1}, X_i, \gamma) \ni (q_{p,i}, \gamma)$。
- 非终结符调用：若 $X_i = B \in N$，则 $\delta(q_{p,i-1}, \varepsilon, \gamma) \ni (q_{\text{输入}B}, q_{p,i}\gamma)$ 并且，对于每个返回地址 $r = q_{p,i}$, $\delta(q_{\text{输出}B}, \varepsilon, r) \ni (r, \varepsilon)$。

集合 $q_0 = q_{\text{start}}$ 和 $F = \{q_S^{\text{out}}\}$。我们称 引擎控制状态空间。

命题 1 (编译正确性**)。** 对于每个上下文无关文法 G，编译得到的非确定性下推自动机 $\mathcal{A}_G$ 恰好接受 $\mathcal{L}(G)$。

证明。 每次调用非终结符时，都会压入一个返回地址，并转移到被调用者的入口状态；完成一个非终结符时，会弹出返回地址，并恢复到调用者的点位置状态。终结符转移恰好消耗所需的终结符；非确定性则表示产生式选择。一个接受运行对应一棵最左推导树，反之亦然。详细证明可通过将经典的上下文无关文法→转下推自动机的转换（见 [16, 第 6] 章）改编为点位置控制状态表示得出。 □

2.3 从左到右掩码的引擎语义

定义 6 （可达性与活性）。 固定一个已编译的非确定性下推自动机 $\mathcal{A}_G$。对于终结符前缀 $u \in \Sigma^*$，将$\text{Reach}_G(u)$ 定义为从初始配置出发、恰好消耗 u 后可达的配置集合，并允许在终结符转移之前、之间及之后进行任意次数的 ε 转移。若存在某个 $v \in \Sigma^*$，使得 c 在消耗 v 后到达接受配置，则配置 c 是活的 。令可共同到达集合 $_G(u) \subseteq \text{Reach}_G(u)$ 为具有活性的可达配置子集。

定义 7 （可接受的下一个词汇词元）。 给定前缀 $u \in \Sigma^*$，定义允许词元集合

$$\Omega_G(u) \coloneqq \{v \in \mathcal{V} \,:\, \text{CoReach}_G(u\,\tau(v)) \neq \emptyset\}.$$

注记 1（可达性作为计算瓶颈）。 计算可共同到达集合$_G$ 是下推系统中的可达性问题，与CFL可达性和过程间分析 [27, 1, 4]紧密相关。现代GCD 实现通过离线预处理、记忆化、确定性下推编译以及细致的分词器对齐 [2, 23, 6, 33, 5] 来降低这一开销。

3 结构等价性、掩码解码与概率约束

3.1 语言等价下的预言机不变性

定义 8 （语言等价）。 如果 G 和 G' 是（强）等价的上下文无关文法，则 $\mathcal{L}$（G）$=\mathcal{L}$（G'）。

Th定理 1 （预言机不变性）。 如果 $\mathcal{L}$（G）$= \mathcal{L}$（G'），那么对于每个终结符前缀 $u \in \Sigma^*$,

$$\Omega_G(u) = \Omega_{G'}(u).$$

因此，任何仅使用 $(p_\theta(\cdot \mid y_{<t}), \Omega_G(\tau(y_{<t})))$ 的解码过程，在将 G 替换为 G'时，其行为保持不变。

证明。 固定 $u \in \Sigma^*$ 和 $v \in \mathcal{V}$。根据定义， $v \in \Omega_G(u)$ 当且仅当 可共同到达集合$_G(u\tau(v)) \neq \emptyset$。根据编译正确性（命题 1），可共同到达集合$_G(w) \neq \emptyset$ 当且仅当 $w \in \mathrm{Pref}(\mathcal{L}(G))$ (即， w 可完成）。因此

$$v \in \Omega_G(u) \iff u\tau(v) \in \mathrm{Pref}(\mathcal{L}(G)).$$

由于 $\mathcal{L}(G) = \mathcal{L}(G')$ 蕴含 $\mathrm{Pref}(\mathcal{L}(G)) = \mathrm{Pref}(\mathcal{L}(G'))$，因此该条件对于 G' 是相同的。 □

3.2 将掩码解码视为受约束的随机过程

令 $\ell_t \in \mathbb{R}^{|\mathcal{V}|}$ 表示步骤 t 的模型对数几率，其中

$$p_\theta(v \mid y_{<t}) = \frac{\exp(\ell_t(v))}{\sum_{w \in \mathcal{V}} \exp(\ell_t(w))}.$$

定义 9 (硬掩码算子)。 给定 $A \subseteq \mathcal{V}$，定义 $m_A \in (\mathbb{R} \cup \{-\infty\})^{|\mathcal{V}|}$ 为

$$m_A(v) \coloneqq \begin{cases} 0, & v \in A, \\ -\infty, & v \notin A. \end{cases}$$

定义掩码下一词元分布

$$q_{\theta,A}(v \mid y_{<t}) \coloneqq \mathrm{softmax}(\ell_t + m_A)(v) = \frac{\exp(\ell_t(v))\mathbf{1}[v \in A]}{\sum_{w \in A} \exp(\ell_t(w))}.$$

在 GCD 中， $A = \Omega_G(\tau(y_{<t}))$。根据定理 1，这仅取决于语言，而不取决于语法表示。

命题 2 (硬掩码解码的可靠性)。 令 q_θ^G 表示自回归采样，其中每一步$y_t \sim q_{\theta,\ \Omega_G(\tau(y_{<t}))}(\cdot \mid y_{<t})$。那么，每个以 $\langle \mathrm{eos} \rangle$ 终止的生成样本 $y_{1:T}$ 都满足 $\tau(y_{1:T}) \in \mathcal{L}(G)$。

证明。 每一步采样得到的词元都会使实际前缀保持在 $\mathrm{Pref}(\mathcal{L}(G))$ 中。此外，只有当当前实际前缀属于 $\mathcal{L}(G)$ 时， $\langle \mathrm{eos} \rangle$ 才是允许的。因此，终止意味着满足归属关系。

□

备注 2（屏蔽并非真实条件分布）。 采用硬掩码进行采样，得到的分布以 $\mathcal{L}(G)$为支撑集，但通常并不等同于将基础模型条件化于 $\tau(y) \in \mathcal{L}(G)$所得的分布。精确条件化需要根据模型从每个候选后续词元出发最终成功完成的概率进行加权。我们将在第 8 节通过 Doob h 变换精确刻画这种差异。

引理 1 (与全局条件化的区分)。 存在一个基础自回归模型 p_θ 和一种语言 L ，使得掩码过程 q_θ^L 不同于全局条件分布$p_\theta(\cdot \mid \tau(y) \in L)$。

证明。 令 $\Sigma = \{a, b\}$、 $L = \{a,\, ba\}$ 和 $\mathcal{V} = \{a, b, \langle \mathrm{eos} \rangle\}$ 满足 $\tau(a) = a$、 $\tau(b) = b$、 $\tau(\langle \mathrm{eos} \rangle) = \varepsilon$。设

$$\begin{gathered} p_\theta(a \mid \varepsilon) = 0.6, \quad p_\theta(b \mid \varepsilon) = 0.4, \quad p_\theta(\langle \mathrm{eos} \rangle \mid \varepsilon) = 0, \\ p_\theta(\langle \mathrm{eos} \rangle \mid a) = 0.1, \quad p_\theta(a \mid a) = 0.9, \quad p_\theta(b \mid a) = 0, \\ p_\theta(a \mid b) = 0.01, \quad p_\theta(b \mid b) = 0.99, \quad p_\theta(\langle \mathrm{eos} \rangle \mid b) = 0. \end{gathered}$$

在掩码过程下， $q(b \mid \varepsilon) = 0.4$ （两个词元均可接受；第 1 步的掩码保持不变）。在全局条件化下，总接受质量为

$$p_\theta(\text{accept via } a) = 0.6 \cdot 0.1 = 0.06, \qquad p_\theta(\text{accept via } ba) = 0.4 \cdot 0.01 \cdot 1 = 0.004,$$

因此 $p_\theta(b \mid \tau(y) \in L) = 0.004/(0.06 + 0.004) \approx 0.0625 \neq 0.4$。 □

4　非终结符委托导致的形式状态空间膨胀

4.1 静态控制状态计数

考虑典范语言 $L = \{a^n b^n : \ n \geq 0\}$ 以及文法:

$$G_1 : S \to aSb \mid \varepsilon$$
$$G_2 : S \to aAb \mid \varepsilon, \quad A \to aAb \mid \varepsilon.$$

二者都满足 $\mathcal{L}(G_1) = \mathcal{L}(G_2) = L$。

定义 10 (文法大小泛函)。 对于上下文无关文法 $G = (N, \Sigma, P, S)$，定义 $p = (A \to \alpha)$ 的右部长度为 $|\alpha|$。定义编译控制状态数

$$\kappa(G) := 1 + 2|N| + \sum_{p \in P} (|\mathrm{rhs}(p)| + 1),$$

与定义5中的构造相匹配。

引理 2 (精确的编译后控制状态数)。 对于每个上下文无关文法 G，编译后的非确定性下推自动机 $\mathcal{A}_G$恰好有 $\kappa(G)$ 个控制状态。

证明。 根据定义5, Q 包含: (i) 一个初始状态; (ii) 每个非终结符对应两个状态（进入/退出）; 以及 (iii) 对于每个产生式 p，其右部长度为 m，包含一个大小为 $m+1$的点位置链。求和得到 $\kappa(G)$。 □

Lemma 3 （ G_2 与 G_1相比的状态空间膨胀）。 在定义的编译下 5,

$$\kappa(G_1) = 8, \qquad \kappa(G_2) = 15.$$

具体而言 地， $\kappa(G_2) > \kappa(G_1)$，编译后的控制状态空间膨胀了 $15/8$ 倍 .

证明。 对于 G_1: $|N| = 1$，产生式 $S \to aSb$ （右部长度为 3）和 $S \to \varepsilon$ （长度为 0）:

$$\kappa(G_1) = 1 + 2 \cdot 1 + ((3 + 1) + (0 + 1)) = 8.$$

对于 G_2: $|N| = 2$，共有四个产生式（两个长度为 3，两个长度为 0）:

$$\kappa(G_2) = 1 + 2 \cdot 2 + 2 \cdot (3 + 1) + 2 \cdot (0 + 1) = 15.$$

□

4.2 屏蔽引擎的在线后果

定义 11 (位集式活动集引擎)。 活动集引擎将活跃控制状态集合表示为关于 Q 的位集，并通过应用针对 ε-转移和终结符转移的预计算邻接表来更新该位集。

命题 3 (控制状态膨胀导致的下界)。 任取一个位集式活动集引擎和一个文法 G。如果该引擎每个解码步骤都执行一次扫描整个位集全集的操作（例如，计算 ε-闭包、规范化或回收非活跃状态），那么每步开销为 $\Omega(\kappa(G))$。因此，将 G_1 替换为 G_2 会使该开销至少增加 $15/8$ 倍。

证明。 在 RAM模型中，扫描长度为 $\kappa(G)$ 的位集需要 $\Omega(\kappa(G))$ 的代价。乘法因子由引理 3得出。 □

注释 3. 这是针对一种常见表示策略的引擎层面论述，并非对所有实现都适用的普遍下界。它提出了一项可检验的不变量：冗余非终结符结构会扩大编译后的状态全集、增加内存占用，并通常会降低缓存局部性，即使语言接受保持不变。

5 结构歧义成本与解析森林密度

5.1 Σ^* 的两个等价文法

令 $\Sigma = \{a, b\}$。定义：

$$G_3 : S \to aS \mid bS \mid \varepsilon,$$
$$G_4 : S_0 \to S \mid \varepsilon, \quad S \to SS \mid a \mid b.$$

引理 4（语言等价）。 $\mathcal{L}(G_3) = \mathcal{L}(G_4) = \Sigma^*$。

证明。 G_3 通过反复发出 a 或 b，并以 ε 情形结束，可以生成任意字符串；因此 $\mathcal{L}(G_3) = \Sigma^*$。在 G_4 中，S 通过 $S \to a$、$S \to b$ 以及对 $S \to SS$ 的封闭性生成所有非空字符串，而 S_0 还额外允许空字符串。因此 $\mathcal{L}(G_4) = \Sigma^*$。 □

备注 4。在 G_4 中使用辅助开始符号 S_0，可以避免由递归非终结符 S 直接允许 ε 时产生的虚假 ε 展开歧义（这会导致空字符串有无限多个推导）。

5.2 解析树与卡特兰歧义

定义 12 (解析树数量**)**。对于文法 G 和字符串 $w \in \Sigma^*$，令 $A_G(w) := |\mathrm{PT}_G(w)|$表示从开始符号推导出 w 的不同解析树的数量。

Le引理 5（G_4的卡特兰解析爆炸）。对于任意非空 $w \in \{a, b\}^n$ 且满足 $n \geq 1$ ，

$$A_{G_4}(w) = C_{n-1},$$

其中 C_k 是第 k个卡特兰数，而 $A_{G_4}(w)$ 因此关于 n呈指数增长。

证明。在 G_4中，每个终结符都由一条叶规则（$S \to a$ 或 $S \to b$）生成。因此，长度为 $n \geq 1$ 的字符串的任意解析树都是一棵有 n 个叶子的满二叉树，其内部节点对应于 $S \to SS$ 的应用。叶标签由 w决定。因此，$A_{G_4}(w)$等于有 n 个叶子的满二叉树的数量，即 C_{n-1}。 □

5.3 打包森林密度

我们使用位置 0、1、.....、n上的半开区间 $[i, j)$。

定义 13（打包森林分割数）。在 G_4 *[29,17]*，的自然 *SPPF* 风格表示中，对于 S 可推导的每个跨度，都存在一个 符号节点 $S[i, j]$，并且一个 打包节点 记录每个不同的分割 $S[i, j] \Rightarrow S[i, k]\, S[k, j]$，其中 $i < k < j$。

引理 6（针对 G_4 的紧确打包森林大小）。固定任意非空的 $w \in \{a, b\}^n$，且满足 $n \geq 2$。对于 S，符号节点的数量为 $\binom{n+1}{2} = \Theta(n^2)$，而打包节点总数为

$$P_n = \sum_{0 \leq i < j \leq n} (j - i - 1) = \binom{n+1}{3} = \Theta(n^3).$$

证明。对于每个 $0 \leq i < j \leq n$，都存在一个符号节点 $S[i, j]$，因为 S 可以推导任意非空子串，从而得到 $\binom{n+1}{2}$ 个符号节点。对于一个固定的跨度 $[i, j)$（长度为 $L \geq 2$），规则 $S \to SS$ 可以在任意 $k \in$ 处（i，j）进行分割，从而产生 $L - 1 =$ （$j - i - 1$）个打包节点。对所有跨度求和，

$$P_n = \sum_{0 \leq i < j \leq n} (j - i - 1).$$

三元组（i，k，j）与 $0 \leq i < k < j \leq n$的一个 3 元子集一一对应，因此 $P_n = \binom{n+1}{3}$。

□

5.4 结构歧义成本

定义 14 （结构歧义成本（SAC））。 固定一个文法 G 以及一种在线策略，该策略为 w 的长度为 t 的前缀维护一个打包结构 F_t 。定义每步增量

$$\Delta_G(t; w) \coloneqq |F_t(w)| - |F_{t-1}(w)|,$$

以及最坏情况下的结构歧义成本

$$\mathrm{SAC}_G(t) \coloneqq \max_{w \in \Sigma^t} \Delta_G(t; w).$$

定理 2 (SAC 分离： G_3 与 G_4)。 对于 G_4，$\mathrm{SAC}_{G_4}(t) = \Theta(t^2)$。对于 G_3，存在一种确定性在线策略，可实现 $\mathrm{SAC}_{G_3}(t) = O(1)$。

证明。 对于 G_4，当第 t 个符号到达时，所有新的跨度 $[i, t)$ 对于 $0 \le i < t$ 均可用。每个长度为 $t - i \ge 2$ 的跨度 $[i, t)$ 都允许 $(t - i - 1)$ 种新的二元切分。新引入的打包备选项总数为

$$\sum_{i=0}^{t-2}(t - i - 1) = \sum_{m=1}^{t-1} m = \frac{t(t-1)}{2} = \Omega(t^2).$$

引理 6 给出了相匹配的 $O(t^2)$ 第 t 步新增打包节点数的上界，因此 $\mathrm{SAC}_{G_4}(t) = \Theta(t^2)$。

对于 G_3，该文法是右线性的（因此是正则的）且无歧义；在线策略只需跟踪当前无歧义状态，其大小为常数。因此 $\mathrm{SAC}_{G_3}(t) = O(1)$。 □

推论 1 （累积在线成本）。 对于 G_4，长度为 n 的字符串上打包结构的累积增长为 $\Theta(n^3)$ 。对于 G_3，累积代价为 $O(n)$ 。

证明。 将 $\Theta(t^2)$ 在 $t = 1, \ldots, n$ 上求和，得到 $\Theta(n^3)$ ；引理 6 给出了相匹配的上界。对于 G_3，对 $O(1)$求和得 $O(n)$ 。 □

备注 5（对引擎而非语言的压力测试）。 对于 Σ^*，理想的可接受性判定器十分简单：所有词元始终可接受，且 ⟨eos⟩始终可接受。然而，当文法引擎被视为黑盒上下文无关文法执行器时，内部歧义可能迫使它维护庞大的打包结构，仅仅为了重新发现这个简单的判定器。SAC 量化了这种开销。

6 解析保持屏蔽的引擎无关下界

本节通过形式化定义一大类引擎，将 SAC 分析提升为引擎无关下界；这类引擎同时满足：(i) 在屏蔽方面可靠；(ii) 以检索高效的方式保持解析。

6.1 一种解析保持的屏蔽模型

定义 15 （在线屏蔽引擎）。 用于文法 G 的在线上下文无关文法屏蔽引擎，是一种在读取每个前缀 $u \in \Sigma^*$ 后输出 $\Omega_G(u)$ 并更新内部状态$\sigma(u)$ ，以供后续步骤使用的算法。

定义 *16*（解析保持、检索高效引擎）。固定一个文法 G 和一个前缀 u。令 $\mathcal{P}_G(u)$ 表示与 u 一致的所有部分解析结构的集合（即，对于 u 的某种补全，其解析森林中可能出现的所有打包分裂备选）。若在线引擎能够精确地由 $\sigma(u)$确定 $\mathcal{P}_G(u)$（即存在一个确定性提取器，可从 $\sigma(u)$ 恢复 $\mathcal{P}_G(u)$），则称其具有解析保持性。若该提取器能在 $O(|\mathcal{P}_G(u)|)$ 时间内枚举 $\mathcal{P}_G(u)$ 的所有元素，则称其检索高效。这是广义解析 *[29, 17]* 中采用的标准输出敏感检索方式，也是 *[18]* 中研究的检索高效解析模型。

注记 6（检索效率为何具有原则性）。 若无检索要求，引擎可以存储二元组（G， u）并按需重新计算解析信息，从而空洞地“保留所有解析”。检索效率要求所维护的状态以输出敏感的方式真正编码解析集，从而排除了这一退化漏洞。

6.2 普适的$\Omega(t^2)$ 每词元下界

定理 3 （与引擎无关的 SAC 下界）。 存在一个常数规模的上下文无关文法 $G^\star$ 以及一族长度为 t 的前缀，使得任何对 $G^\star$ 既可靠又保持解析且检索高效的在线掩码引擎，在最坏情况下都必须产生 $\Omega(t^2)$的每词元更新工作量。因此，在长度为 n 的输入上，累积工作量为 $\Omega(n^3)$。

证明。 取 $G^\star = G_4$。考虑任意长度为t 的前缀 u （字母表为 $\{a, b\}$）。由引理 6，集合 $\mathcal{P}_{G^\star}(u)$包含所有三元组 $(i,\ k,\ j)$，其中 $0 \le i < k < j \le t$，累计共 $\binom{t+1}{3}$ 个元素。由定理 2的论证，那些新引入的、位于步骤 t 的三元组数量为 $\binom{t}{2} = \Omega(t^2)$。

对于检索高效且保持解析的引擎， $\sigma(u_{1:t})$ 必须能够在线性输出时间内枚举 $\mathcal{P}_{G^\star}(u_{1:t})$ 的所有元素。编码 $\Omega(t^2)$ 个全新的不同对象，需要在 RAM模型 中进行 $\Omega(t^2)$ 次基本内存写入；否则，提取器无法在 $O(|\mathcal{P}|)$ 时间内枚举它们。因此，每个词元的更新成本为 $\Omega(t^2)$，求和后得到 $\Omega(n^3)$。□

备注 7（与经典解析复杂度的关系）。定理 3 在指定的语义接口内是无条件成立的（可靠性 + 检索高效的解析保持性）。它补充了 Valiant [34] 和 Lee [18], 的经典结果：这些结果通过算法归约，将上下文无关文法句法分析的复杂度与布尔矩阵乘法联系起来。我们的论证由输出结构驱动——基于打包分裂集的增长——而非从另一问题归约而来。

7 解码成本等价类与规范低SAC形式

预言机不变性（定理 1）表明，语言等价会保留可接受词元预言机。本节将探讨一个更细致的问题：两个等价文法何时在解码器的成本视角下也等价?

7.1 内在 SAC

定义 17 （内在解析保持 SAC）。 固定一个文法 G。令 $\mathcal{E}_{\mathrm{pp}}(G)$ 表示适用于 G 的可靠、检索高效的解析保持型在线掩码引擎类。将内在*SAC* 定义为

$$\mathrm{SAC}^\star_G(t) \coloneqq \inf_{E \in \mathcal{E}_{\mathrm{pp}}(G)} \max_{w \in \Sigma^t} \Delta^E_G(t; w),$$

其中， $\Delta^E_G(t;\ w)$ 表示其维护的解析表示每一步由引擎诱导的增长量（以基本解析对象计量）。

注释 8。 下确界刻画了在能够检索高效地保留所有解析的引擎中可实现的最佳渐近行为。它抽象掉了实现细节，同时仍受到定理 3的实质性约束。

7.2 成本等价关系

定义 18（成本等价）。 对于共享终结符字母表的文法 G 和 H，定义：

- 语言等价： $G \equiv_L H$ 当且仅当 $\mathcal{L}(G) = \mathcal{L}(H)$。
- SAC 等价： $G \equiv_{\mathrm{SAC}} H$ 当且仅当存在常数 c_1、$c_2 > 0$ 和 t_0，使得对所有$t \geq t_0$,

$$c_1 \, \mathrm{SAC}^{\star}_G(t) \leq \mathrm{SAC}^{\star}_H(t) \leq c_2 \, \mathrm{SAC}^{\star}_G(t).$$

- 解码成本等价： $G \equiv_{\text{解码}} H$ 当且仅当 $G \equiv_L H$ 并且 $G \equiv_{\mathrm{SAC}} H$。

备注 9（解释）。 $G \equiv_L H$ 意味着可接受词元集相同（定理1）。 $G \equiv_{\mathrm{dec}} H$ 对此作了进一步加强：最优的 保留解析的在线成本在渐近意义上相同，因此这些文法不仅在语义上等价，从解码器成本视角来看也等价。

7.3 有界重写下的规范低 SAC 代表

即使在受限情形下，完全最小化下推模型也是 NP 困难的 [11], ，因此任何关于规范最小 SAC 代表元的主张都必须谨慎界定范围。

定义 19 (有界重写族)。 固定一个有限的局部文法重写集合 $\mathcal{R}$，其中每个重写都会将一个文法映射为与其语言等价的文法。对于整数 $k \geq 0$，令 可达集$_{\mathcal{R},\, k}(G)$ 表示从 G出发，至多应用 k 次来自 $\mathcal{R}$的重写后可达的文法有限集合。

定理 4 (最小 SAC 代表元的存在性)。 固定 $\mathcal{R}$、k 和一个文法 G，使得 $\mathcal{R}$ 中的每个重写都保持 $\mathcal{L}(G)$。那么，存在 $G_{\min} \in$可达集$_{\mathcal{R},\, k}(G)$，使得

$$\mathrm{SAC}^{\star}_{G_{\min}}(t) \leq \mathrm{SAC}^{\star}_H(t) \quad \textit{for all } H \in \mathsf{Reach}_{\mathcal{R},k}(G) \textit{ and all } t.$$

此外，还存在这样一个 $G_{\min}$，它对于可达集$_{\mathcal{R},\, k}(G)$上的任意有限决胜函数（例如文法大小）也同时是最小的。

证明。 Reach$_{\mathcal{R},\, k}(G)$ 是有限的，因为 $\mathcal{R}$ 是有限的，且 k 有界。由 SAC* 在此有限集合上诱导的逐点偏序至少存在一个极小元素。应用任意有限决胜函数即可选出一个规范代表元。 □

备注 10（范围）。定理4是一个存在性结果：一旦固定有界变换族，规范的低 SAC 对象在数学上就定义明确。它并未断言通常具有可计算性，也未声称不受限制的全局最小化是可行的。

8 文法条件化的自回归过程

现在，我们将视角从句法分析机制拓展到把受约束解码视为与形式语言约束交互的随机过程。由此，我们建立理论基础，用以量化硬掩码相较于真实条件采样所引入的偏差。

8.1 通过 Doob h 变换实现真实条件化

固定一个定义在以 $\langle \mathrm{eos} \rangle$ 结束的词元序列上的基础自回归模型 $p(\cdot)$。令 $L \subseteq \Sigma^*$ 为目标语言，其接受事件为

$$E := \{\tau(y_{1:T}) \in L \text{ and } y_T = \langle \mathrm{eos} \rangle\}.$$

定义 20 (生存函数**)。** 对于任意前缀词元序列 $y_{<t}$，定义生存概率

$$h(y_{<t}) := \Pr_p(E \mid y_{<t}).$$

定理5 (Doob h-变换，用于条件解码)。 假设 $h(y_{<t}) > 0$。那么，真实的条件下一个词元分布满足

$$p_E(v \mid y_{<t}) := \Pr_p(y_t = v \mid y_{<t}, E) = p(v \mid y_{<t}) \cdot \frac{h(y_{<t}v)}{h(y_{<t})}.$$

证明。 根据贝叶斯法则和塔性质，

$$\Pr(y_t = v \mid y_{<t}, E) = \frac{\Pr(y_t = v \mid y_{<t}) \Pr(E \mid y_{<t}v)}{h(y_{<t})} = p(v \mid y_{<t}) \frac{h(y_{<t}v)}{h(y_{<t})}.$$

这是马尔可夫链 [7, 20] 上 Doob h-变换的离散时间类比 . □

定义21 (硬掩码过程**)。** 令 $A(y_{<t}) := \Omega_G(\tau(y_{<t}))$ 为允许词元集合。定义硬掩码下一个词元分布

$$q(v \mid y_{<t}) := \frac{p(v \mid y_{<t}) \, \mathbf{1}[v \in A(y_{<t})]}{\sum_{w \in A(y_{<t})} p(w \mid y_{<t})}.$$

8.2 精确刻画与失真界

由于 $h(y_{<t}v) = 0$ 只要 $v \notin A(y_{<t})$，当$h(y_{<t}) > 0$时， p_E 和 q 的支持集都包含于 $A(y_{<t})$。这些分布对允许分支的赋权方式不同。

定理6 （硬屏蔽等同于真实条件化）。 对于固定前缀 $y_{<t}$，若 $h(y_{<t}) > 0$，则以下条件等价：

1. $q(\cdot \mid y_{<t}) = p_E(\cdot \mid y_{<t})$，
2. $h(y_{<t}v)$ 在所有可容许的 $v \in A(y_{<t})$ 上为常数。

证明。 限制到 $A = A(y_{<t})$：

$$p_E(v \mid y_{<t}) \propto p(v \mid y_{<t}) \, h(y_{<t}v), \qquad q(v \mid y_{<t}) \propto p(v \mid y_{<t}).$$

当且仅当乘法因子 $h(y_{<t}v)$ 在 A 上为常数时，两者相同。 □

定义 22 （生存扩散）。 对于前缀 $y_{<t}$ 及其可容许集合 $A = A(y_{<t})$，定义

$$h_{\min}(y_{<t}) := \min_{v \in A} h(y_{<t}v), \qquad h_{\max}(y_{<t}) := \max_{v \in A} h(y_{<t}v),$$

以及扩散比 $\Gamma(y_{<t}) := h_{\max}(y_{<t}) / h_{\min}(y_{<t})$ （其中 $\Gamma = \infty$ 若 $h_{\min} = 0$）。

命题 4 （单步 KL 界）。 假设 $0 < h_{\min}(y_{<t}) \le h_{\max}(y_{<t}) < \infty$。*Th en*

$$\mathrm{KL}\big(q(\cdot \mid y_{<t}) \,\|\, p_E(\cdot \mid y_{<t})\big) \le \log \Gamma(y_{<t}).$$

证明。 记 $A = A(y_{<t})$，$p_v = p(v \mid y_{<t})$，$h_v = h(y_{<t}v)$，$Z = \sum_{w \in A} p_w$，$H = \sum_{w \in A} p_w h_w$。那么 $q_v = p_v / Z$ 和 $p_v^E = p_v h_v / H$，由此得到

$$\mathrm{KL}(q \| p_E) = \mathbb{E}_q\left[\log \frac{q_v}{p_v^E}\right] = \log \frac{H}{Z} - \mathbb{E}_q[\log h_v].$$

由于 $h_{\min} \le h_v \le h_{\max}$，我们有 $\log(H/Z) \le \log h_{\max}$ 和 $-\mathbb{E}_q[\log h_v] \le -\log h_{\min}$，因此 $\mathrm{KL}(q \| p_E) \le \log h_{\max} - \log h_{\min} = \log \Gamma(y_{<t})$。 □

Cor推论 2 (单步总变差界)。 在命题 4 ,

$$\mathrm{TV}\big(q(\cdot \mid y_{<t}),\, p_E(\cdot \mid y_{<t})\big) \;\le\; \sqrt{\tfrac{1}{2}\log \Gamma(y_{<t})}.$$

证明。 对命题 $\mathrm{TV}(P,\ Q) \le \sqrt{\mathrm{KL}(P\|Q)/2}$ 应用 Pinsker不等式4。 □

注释 11（解释）。 当允许的后续词元的存活概率几乎相等（Γ（$y_{<t}$） ≈ 1）时，硬掩码能够近似真实的条件采样。这将一种直觉形式化：硬掩码忽略了“完成难易程度”的影响，并将失真完全量化为完成概率变异性。若不同的允许词元会导致最终接受概率相差悬殊，则这种差异 Γ 很大；若约束在允许分支上近乎均匀，则差异很小。

9 神经集成：注意力与专家混合路由

我们将可达性预言机调制 Transformer 中对数几率的代数表达出来，并将该表述扩展到 MoE 路由概率。

9.1 语法调制的可接受性下的 Transformer 对数几率

设因果Transformer [35] 根据前缀 $y_{<t}$ 计算隐藏状态 $h_t \in \mathbb{R}^d$ （假设使用 KV 缓存）。未掩码的对数几率为

$$\ell_t = W_{\text{out}} h_t + b_{\text{out}} \in \mathbb{R}^{|\mathcal{V}|}.$$

令 $A_t := \Omega_G(\tau(y_{<t}))$ 表示允许词元集合。掩码后的对数几率为

$$\ell_t^{(G)} := \ell_t + m_{A_t}, \tag{2}$$

受约束的下一令牌分布为

$$q_\theta^G(y_t = v \mid y_{<t}) = \mathrm{softmax}(\ell_t^{(G)})(v) = \frac{\exp(\ell_t(v))\mathbf{1}[v \in A_t]}{\sum_{w \in A_t} \exp(\ell_t(w))}. \tag{3}$$

这是 GCD引擎 [28, 2, 23, 6, 33]执行的代数对数几率调制。

9.2 语法状态下的 MoE路由

考虑一个包含 MoE 前馈块 [30, 10] 的 Transformer。在第 ℓ 层和位置 t，门控网络会在 E 个专家上生成预激活值 $g_{t,\ell} \in \mathbb{R}^E$ 。标准路由计算

$$r_{t,\ell} = \mathrm{softmax}(g_{t,\ell}), \qquad \mathcal{E}_{t,\ell} = \mathrm{TopK}(r_{t,\ell}), \qquad h'_{t,\ell} = \sum_{e \in \mathcal{E}_{t,\ell}} \alpha_{t,\ell,e}\, \mathrm{FFN}_e(h_{t,\ell}),$$

归一化权重为 $\alpha_{t,\ell,e}$。令 $\mathcal{C}_t :=$ 可共同到达集合$_G(\tau(y_{<t}))$ 表示活动配置集。语法感知 MoE 可以通过特征嵌入 $\phi(\mathcal{C}_t) \in \mathbb{R}^k$ 来纳入 $\mathcal{C}_t$ ：

$$r_{t,\ell}^{(G)} \;=\; \mathrm{softmax}\big(g_{t,\ell} + U_\ell\, \phi(\mathcal{C}_t)\big), \tag{4}$$

其中 $U_\ell \in \mathbb{R}^{E \times k}$ 被学习。这定义了一个将句法状态与条件计算耦合起来的联合约束。

9.3 延迟包络

¯设 $V := |\mathcal{V}|$ （词汇表大小）， B 束宽， t 当前步， $\ell := \mathbb{E}[|\tau(y_t)|]$ 每个令牌的平均终止长度， M_t 每个假设的活动配置表示大小，以及$K_t := |\Omega_G(\tau(y_{<t}))|$ 可接受令牌的数量。

¯**定义 23 (**抽象引擎步原语**)**。 设 $\mathrm{Update}_G(M_t, \ell)$ 表示消耗一个词元后更新文法引擎状态的成本，$\mathrm{Enumerate}_G(M_t)$ 表示枚举可接受令牌的成本。

命题 5 (每步复杂度：密集词汇表掩码**)**。 如果模型针对每个假设 B ，都对全部 V 个词元计算密集 *logits*，则每步的时间为

$$O\big(B \cdot \big(T_{\mathrm{model}}(t) + \mathrm{Update}_G(M_t, \bar{\ell}) + \mathrm{Enumerate}_G(M_t) + V\big)\big),$$

其中 $T_{\mathrm{model}}(t)$ 表示使用 *KV* 缓存时 *Transformer* 的前向计算成本。

证明。 对于每个假设：计算 logits（$T_{\mathrm{model}}(t)$），更新并枚举掩码（Update_G +$\mathrm{Enumerate}_G$），然后扫描长度为 V 的掩码。再乘以 B。 □

命题 6 (每步复杂度：稀疏可接受集评分**)**。 如果推理栈只对可接受令牌评分（例如，通过 *Trie* 限制投影），那么 V 项可以替换为 K_t:

$$O\big(B \cdot \big(T_{\mathrm{model}}(t) + \mathrm{Update}_G(M_t, \bar{\ell}) + \mathrm{Enumerate}_G(M_t) + K_t\big)\big).$$

推论 3 (SAC 导致的掩码瓶颈**)**。 对于 G_4 以及可靠的打包结构引擎，$M_t = \Omega(t^2)$ *(*定理 *3)*，因此，任何涉及新建打包结构的更新例程都满足 更新$_{G_4}(M_t, \bar{\ell}) = \Omega(t^2)$。对于 G_3，可以维护 $M_t = O(1)$ ，得到 更新$_{G_3}$ ¯$O(1)$。

9.4 具有可靠屏蔽的语法条件 logits

定义 24 (语法条件 logits**)**。 令 $e_t = \psi(\mathcal{C}_t) \in \mathbb{R}^k$ 嵌入活动配置集。定义

$$\tilde{\ell}_t(v) := \ell_t(v) + u(v)^\top e_t$$

对于学习得到的向量 $u(v) \in \mathbb{R}^k$，使用 A_t 进行硬掩码：

$$q_\theta^{G,\psi}(v \mid y_{<t}) := \mathrm{softmax}(\tilde{\ell}_t + m_{A_t})(v).$$

命题 7 (语法条件 logits 下的可靠性**)**。 对于任意嵌入 ψ 和权重 $u(\cdot)$，从 $q_{\cdot,\theta}^{G\ \psi}$ 采样并终止后，只会得到 $\mathcal{L}(G)$ 中的字符串。

证明。 在每一步， $q_{\cdot,\theta}^{G\ \psi}$ 的支持集 A_t 都包含于其中，这是由硬掩码的构造保证的。命题 2 的论证可以原样应用。 □

注释 12. 硬掩码可确保正确性，而语法条件 logits 则能降低掩码截断高概率质量的频率——命题 4量化了这一关键失真问题。在混合专家模型中，同一嵌入 e_t 可通过（4）输入路由器，从而按句法区域促进专家专门化。实证验证需要在 JSONSchemaBench [12]等结构化基准上开展受控消融实验。

10 从 SAC 到运行时预测、语法优化与混合控制

本节将 SAC 分析扩展为：(i) 可通过经验校准的性能模型；(ii) 保持等价性的文法优化器；以及 (iii) 通过插桩引擎实现更深入的符号 – 神经耦合。

10.1 插桩引擎与 SAC 代理

生产引擎不会直接公开 $|F_t(w)|$；它们公开的是引擎原生结构（Earley 项目、LR 项目、字典树节点、持久栈节点等）。LLGuidance 使用基于 Earley 的识别器，并结合词元字典树遍历 [15]；XGrammar 则采用词汇表拆分和持久栈表示 [6]。我们引入通用插桩接口，以统一这些结构。

定义 25（插桩解码引擎）。固定文法 G 与分词器 τ。插桩解码引擎是一个三元组 $E = (\mathrm{Step}、\mathrm{Mask}、\mathrm{Ctr})$，其中在每一步 t：

- Step 为每个假设推进引擎状态；
- Mask 返回 $\Omega_G(\tau(y_{<t}))$；
- Ctr 返回一个由非负计数器组成的向量，用于记录符号工作量（创建的项、遍历的边、访问的字典树节点、触及的栈节点等）。

定义 26（SAC 代理）。令 $c_t(E; w) \in \mathbb{R}^d_{\geq 0}$ 为第 t 步的计数器向量。标量 SAC 代理 为

$$S_t^E(w) \coloneqq \langle \alpha, c_t(E; w) \rangle$$

对于某个固定的 $\alpha \in \mathbb{R}^d_{\geq 0}$，选取它来估算每步边际符号工作量。

引理 7（拆分完备性下的代理下界 SAC）。考虑带规则 $S \to SS$ 的 G_4。假设引擎 E 是拆分完备的：对于其维护结构中每个长度为 ≥ 2、满足 $[i, t)$ 的跨度，它都会表示每个满足 $i < k < t$ 的拆分点 k。那么存在 $\beta > 0$，使得对于所有 t，

$$\max_{w \in \Sigma^t} S_t^E(w) \;\geq\; \beta \cdot \mathrm{SAC}_{G_4}(t).$$

证明。在步骤 t，拆分完备性要求引擎为每个以 t 结尾的新跨度计入每个新引入的拆分备选项。根据定理2，最坏情况下，这需要 $\Omega(t^2)$ 的工作量。任何为此类备选项赋予正权重的代理，其计数都有一个线性下界，二者相差至多一个常数因子。 □

10.2 预测性能模型

令 $T_{\mathrm{NN}}(t)$ 表示使用 KV 缓存时每个假设对应的 GPU 模型前向计算时间，令 $T_{\mathrm{mask}}(t)$ 表示用于状态推进和掩码计算的 CPU 端（或混合）引擎时间。

定义 27（关键路径步时间）。在理想重叠下，

$$T_{\mathrm{step}}(t) \coloneqq \max\{T_{\mathrm{NN}}(t),\, T_{\mathrm{mask}}(t)\} + T_{\mathrm{sync}}(t) + T_{\mathrm{sel}}(t),$$

其中 $T_{\mathrm{sync}}(t)$ 计入同步，而 $T_{\mathrm{sel}}(t)$计入采样/*top-k* 选择。

XGrammar明确将“使语法计算与GPU执行重叠”作为延迟优化策略 [6]。一项实际研究应测量重叠，并识别屏蔽何时成为关键路径；Nsight Systems为此提供NVTX范围标注和时间线可视化 [22]。

定义 28（仿射代理到时间模型）。仿射代理模型是

$$T_{\mathrm{mask}}(t) \approx a \cdot S_t^E(w) + b,$$

对于通过经验轨迹（例如采用最小二乘拟合）估算出的常数 $a \geq 0,\ b \geq 0$。

备注 13（校准工具链）。 校准通常在仅包含掩码操作的工作负载（例如 MaskBench）上进行 [14]。完整的工具链包括：用于时间线/重叠分析的 Nsight Systems [22]；Linux `perf` 用于获取 CPU 性能监控单元计数器 [24]；用于 GPU 缓存/内存分析的 Nsight Compute [21]；用于算子级归因分析的 PyTorch Profiler [25]；以及用于监测内存压力的 CUDA 分配器快照 [26]。

命题 8 （符号工作量的波束放大）。 假设波束假设不共享语法引擎状态，

$$S^E_{t,\text{total}} \;=\; \sum_{i=1}^{B} S^E_t(w^{(i)}) \;=\; \Theta(B \cdot \overline{S}_t),$$

其中 $\overline{S}_t$ 是每个假设的平均代理量。在仿射模型下，$T_{\text{mask, total}}(t) \;\approx a \cdot \Theta\,(B \cdot \overline{S}_t) \;+ b'$。

证明。 计数器由彼此独立的引擎更新生成；代理量在相互独立的假设之间具有可加性。代入仿射模型后，常数项并入 b'。 □

备注 14（DPDA 编译与批处理效率）。 当非确定性导致持久栈分支时，即使前缀长度固定，S^E_t 也会随波束增大。Pre^3 通过将 LR(1) 结构编译为确定性下推自动机，并预先计算前缀条件边来减少运行时路径探索，从而解决这一问题 [5]。在我们的框架中，这相当于减少活跃配置的扇出，并降低各步之间 S^E_t的方差。

10.3 自动化文法优化

Oracle 不变性（定理 1）允许编译器在等价文法中进行搜索，以最小化依赖引擎的成本模型。

定义 29 （语法优化问题）。 固定引擎族 E、分词器 τ 和工作负载 $\mathcal{D}$。定义可行集 $\mathcal{G}(G)$: $= \{G' :\; \mathcal{L}(G') \;= \mathcal{L}(G)\}$ 以及目标函数

$$\min_{G' \in \mathcal{G}(G)} \;\; \mathbb{E}_{w \sim \mathcal{D}} \Big[\text{Cost}_E(G', \tau; w) \Big],$$

其中 成本$_E$ 可以综合静态项（例如 $\kappa(G')$）、动态项（预期 *SAC* 代理）以及分词器对齐开销。

备注 15（分词器对齐）。 DOMINO 指出，子词词汇表与语法终结符之间的不当对齐会扭曲模型分布，并产生“桥接”处理开销 [2]。Park 等人将依赖词法分析器状态的词元到终结符映射形式化，以避免运行时出现低效组合 [23]。因此，对齐不当的成本是 Cost$_E$ 的首要组成部分。

定义 30 （内联改写）。 设 $A \in N$ 具有产生式 $A \to \gamma_1 \mid \cdots \mid \gamma_m$。内联改写会将产生式 $B \to \alpha A \beta$ 替换为集合 $\{B \to \alpha\gamma_i\beta :\; i = 1, \ldots, m\}$，并保持其他产生式不变。

引理 8 (内联保持语言不变)。 假设 A 不是开始符号，且内联改写没有引入新的非终结符。那么改写后的语法 G' 满足 $\mathcal{L}(G') = \mathcal{L}(G)$。

证明。 在 G 中先应用 $B \to \alpha A \beta$ 再应用 $A \Rightarrow^* \gamma_i$ 的每个推导，都对应于在G' 中使用 $B \to \alpha\gamma_i\beta$得到的一个推导；因此 $\mathcal{L}(G) \subseteq \mathcal{L}(G')$。反向包含关系同理可证。 □

备注 16（面向文法优化的等式饱和）。 即使采用局部重写，等价文法的搜索空间仍然很大。借助 e-图的等式饱和 [32, 37]提供了一种有原则的方法：在 e-图中表示文法片段，应用保持等价性的

重写（内联、委托消除、递归规范化），附加代价分析以估算 κ、SAC 代理替代和分词器对齐，然后提取最小代价等价文法。困难性结果（VPA最小化是 NP完全问题 [11]）表明，应采用启发式或类受限优化，而非追求全局最优。

11 相关工作

句法分析理论与可达性。 经典的上下文无关文法句法分析算法为我们的形式化提供了基础 [9, 38]。GLR句法分析和歧义复杂度见 [17], ，基于SPPF的打包森林表示见 [29]。Valiant 将上下文无关文法识别归约为布尔矩阵乘法 [34]，Lee 的反向归约则见 [18] ，二者建立了通用上下文无关文法句法分析复杂度与矩阵乘法之间的紧密联系；我们的与引擎无关的下界（第6节）则与之互补，并由输出结构驱动。CFL可达性与下推模型检测见 [27, 4, 1]。

语法约束解码。 PICARD 使用增量句法分析，在解码过程中拒绝无效的子词词元 [28]。DOMINO 强调子词对齐约束和非侵入式掩码 [2]。Park 等人分析了预处理与在线处理之间的权衡，以及依赖词法分析器状态的词元映射 [23]。XGrammar 引入了词汇表拆分、持久栈、文法变换以及 CPU/GPU重叠 [6]。SynCode 提供了完备性与可靠性的框架 [33]。Pre^3 编译确定性下推自动机，以减少大批量场景下对运行时路径的探索 [5]。JSONSchemaBench 提供了大规模结构化输出基准测试，其中包含用于隔离掩码计算性能的 MaskBench 组件 [12, 14]。Guidance、Outlines 和 LLGuidance 等库提供了广泛使用的约束接口[19, 13, 8, 15]。

随机过程与 h-变换。Doob h-变换刻画了条件马尔可夫过程，并将概率分析与调和函数理论联系起来[7, 20]。我们使用 h-变换将 GCD 刻画为真正的条件解码（第8节），由此将约束生成与经典的稀有事件条件化理论联系起来。

语法优化。等式饱和与 e-图 [32, 37]为等价条件下的优化提供了系统化框架，我们提出将其作为自动化文法重构的基础。可见下推自动机最小化的 NP完全性 [11] 促使我们采用启发式方法和受限类别方法。

神经架构耦合。Transformer [35] 和混合专家架构 [30, 10] 是我们神经集成研究结果所针对的推理对象。文献 [31], 研究了语义解析中的约束生成，而 LMQL 则为约束大语言模型解码提供了编程接口[3]。

12 结论

我们将语法约束解码形式化为一种耦合：将神经下一词元模型与基于 CFG 推导的下推可达性预言机相结合。我们的贡献如下。

预言机不变性 （定理 1）：语言等价文法对每个前缀都会产生相同的允许词元集合，因此也会产生相同的 logit掩码。这种不变性与引擎状态空间大小及在线更新成本存在可证明的差异并存。

状态空间膨胀 （引理 3）：对于 $a^n b^n$ 语言，我们在一种具体的下推自动机编译下给出了精确的控制状态计数，并证明存在由冗余非终结符委托导致的 $15/8$ 膨胀因子。

*SAC*界 （定理2，推论 1）：连接文法会造成 $\Theta(t^2)$ 增量打包结构增长和 $\Theta(n^3)$ 累积代价；右递归文法分别具有 $O(1)$和$O(n)$。

与引擎无关的下界 （定理3）：任何可靠、检索高效且解析保持的屏蔽引擎，在 $\Omega(t^2)$ 上都必须付出逐词元工作量 G_4，因此在此模型中，SAC下界是无条件的。

解码成本等价类 （定理4）：在任意有界重写族中，都存在最小SAC文法代表，从而为规范的低SAC范式提供了数学上精确的定义。

文法条件过程 （定理5，命题 4，推论 2）：真实的条件采样器可由Doob h-变换刻画，而硬掩码会造成单步KL失真，其上界为 $\log \Gamma(y_{<t})$，其中 Γ 衡量了允许的下一个词元之间的生存概率差异。

神经集成与性能建模：我们推导了 Transformer 和混合专家架构的延迟包络，展示了SAC引起的瓶颈会影响束搜索下的掩码关键路径时间，并将这些结果与基于插桩的预测模型以及通过等价保持重写和等式饱和实现的自动文法优化联系起来[12, 5, 32, 37]。

综上，这些结果为文法重构作为延迟优化问题奠定了严谨的理论基础，并明确了其语义不变量和可测量的代价代理。

参考文献

[1] Rajeev Alur、Michael Benedikt、Kousha Etessami、Patrice Godefroid、Thomas W. Reps 和 Mihalis Yannakakis。《递归状态机分析》。*ACM*编程语言与系统汇刊，27(4)：786–818，2005年。

[2] Luca Beurer-Kellner、Marc Fischer 和 Martin Vechev。《以正确方式引导大语言模型：快速、非侵入式约束生成》。收录于 Ruslan Salakhutdinov、Zico Kolter、Kather- ine Heller、Adrian Weller、Nuria Oliver、Jonathan Scarlett 和 Felix Berkenkamp 编，第*41*届国际机器学习会议论文集，第235卷，机器学习研究论文集，第3658–3673页。PMLR，2024年7月。

[3] Luca Beurer-Kellner、Marc Fischer 和 Martin T. Vechev。提示即编程：面向大语言模型的查询语言。*ACM*编程语言汇刊，7(PLDI):1946–1969，2023年。

[4] Ahmed Bouajjani、Javier Esparza 和 Oded Maler。下推自动机的可达性分析：在模型检测中的应用。载于*CONCUR'97*：并发理论，第 1243 卷，计算机科学讲义，第 135–150 页。Springer，1997年。

[5] Junyi Chen、Shihao Bai、Zaijun Wang、Siyu Wu、Chuheng Du、Hailong Yang、Ruihao Gong、Shengzhong Liu、Fan Wu 和 Guihai Chen。《Pre^3：支持确定性下推自动机，实现更快速的结构化大语言模型生成》。载于《计算语言学协会第*63*届年会论文集（第*1*卷：长论文）》，第11253–11267页，奥地利维也纳，2025年7月。计算语言学协会。

[6] Yixin Dong、Charlie F. Ruan、Yaxing Cai、Ruihang Lai、Ziyi Xu、Yilong Zhao 和 Tianqi Chen。《XGrammar：面向大语言模型的灵活高效结构化生成引擎》。arXiv:2411.15100，2024年。[7] J. L. Doob。《条件布朗运动与调和函数的边界极限》。法国数学会通报，85:431–458，1957年。[8]dottxt-ai。《Outlines：面向大语言模型的结构化输出》。https://github.com/dottxt-ai/ outlines，2026年。访问于2026年3月4日。[9] Jay Earley。《一种高效的上下文无关解析算法》。ACM通讯，13(2):94–102，1970年。[10] William Fedus、Barret Zoph 和 Noam Shazeer。《Switch Transformer：以简单高效的稀疏性扩展至万亿参数模型》。机器学习研究期刊，23(120):120:1–120:39，2022年。[11] Olivier Gauwin、Anca Muscholl 和 Michael Raskin。《可见下推自动机的最小化是NP完全问题》。计算机科学逻辑方法，16(1)，2020年。[12] Saibo Geng、Hudson Cooper、Michał Moskal、Samuel Jenkins、Julian Berman、Nathan Ranchin、Robert West、Eric Horvitz 和 Harsha Nori。《JSONSchemaBench：面向语言模型结构化输出的严格基准测试》。arXiv:2501.10868，2025年。[13] guidance-ai。《guidance：用于控制大语言模型的引导语言》。https: //github.com/guidance-ai/guidance，2026年。访问于2026年3月4日。[14] guidance-ai。《jsonschemabench：JSONSchemaBench 和 MaskBench》。https://github. com/guidance-ai/jsonschemabench，2026年。访问于2026年3月4日。[15] guidance-ai。《llguidance：超高速结构化输出》。https://github.com/guidance-ai/llguidance，2026年。访问于2026年3月4日。[16] John E. Hopcroft 和 J effrey D. Ullman。《自动机理论、语言与计算导论》。Addison-Wesley，1979年。[17] Mark Johnson。《GLR句法分析的计算复杂性》。载于 Masaru Tomita 编，《广义LR解析》，第35–42页。Springer，1991年。[18] Lillian Lee。《快速上下文无关文法解析需要快速布尔矩阵乘法》。ACM学报，49(1):1–15，2002年。[19] 微软研究院。《Guidance：控制语言模型输出》。https://www.microsoft.com/en-us/research/project/guidance-control-lm-output/，2026年。访问于2026年3月4日。[20] J. R. Norris。《马尔可夫链》。剑桥统计与概率数学丛书。剑桥大学出版社，1997年。[21] 英伟达。《Nsight Compute 性能分析指南》。https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html，2026年。访问于2026年3月4日。[22]英伟达。《Nsight Systems 用户指南》。https://docs.nvidia.com/nsight-systems/ UserGuide/index.html，2026年。访问于2026年3月4日。[23]Kanghee Park、Timothy Zhou 和 Loris D'Antoni。《灵活高效的文法约束解码》。arXiv:2502.05111，2025年。2025年国际机器学习大会海报。

[24] perf wiki 贡献者。《Linux 性能计数器分析》。https://perfwiki.github.io/main/tutorial/，2026年。访问日期：2026年3月4日。[25]PyTorch 贡献者。《PyTorch Profiler》。https://docs.pytorch.org/tutorials/ recipes/recipes/profiler_recipe.html，2026年。访问日期：2026年3月4日。[26] PyTorch 贡献者。《了解 CUDA 内存使用情况》。https://docs.pytorch. org/docs/stable/torch_cuda_memory.html，2026年。访问日期：2026年3月4日。[27] Thomas W. Reps、Susan Horwitz 和 Shmuel Sagiv。《通过图可达性进行精确的过程间数据流分析》。载于《第22届 ACM SIGPLAN-SIGACT 编程语言原理研讨会（POPL）论文集》，第49–61页，1995年。[28] Torsten Scholak、Nathan Schucher 和 Dzmitry Bahdanau。《PICARD：对语言模型的受约束的自回归解码进行增量句法分析》。载于 Marie-Francine Moens、Xuanjing Huang、Lucia Specia 和 Scott Wen-tau Yih（编），《2021年自然语言处理经验方法会议论文集》，第9895–9901页，线上及多米尼加共和国蓬塔卡纳，2021年11月。计算语言学协会。[29] Elizabeth Scott 和 Adrian Johnstone。《识别并非句法分析——从三次时间识别器进行 SPPF 风格解析》。《计算机程序科学》，75(1-2)：55–70，2010年。[30] Noam Shazeer、Azalia Mirhoseini、Krzysztof Maziarz、Andy Davis、Quoc V. Le、Geoffrey E. Hinton 和 Jeff Dean。《规模异常庞大的神经网络：稀疏门控混合专家层》。载于《国际学习表征会议》，2017年。[31] Richard Shin、Christopher Lin、Sam Thomson、Charles Chen、Subhro Roy、Emmanouil Antonios Platanios、Adam Pauls、Dan Klein、Jason Eisner 和 Benjamin Van Durme。《受约束的语言模型产生少样本语义解析器》。载于 Marie-Francine Moens、Xuanjing Huang、Lucia Specia 和 Scott Wen-tau Yih（编），《2021年自然语言处理经验方法会议论文集》，第7699–7715页，线上及多米尼加共和国蓬塔卡纳，2021年11月。计算语言学协会。[32] Ross Tate、Michael Stepp、Zachary Tatlock 和 Sorin Lerner。《等式饱和：一种新的优化方法》。载于《第36届 ACM SIGPLAN-SIGACT 编程语言原理年会（POPL）论文集》，第264–276页，2009年。[33] Shubham Ugare、Tarun Suresh、Hangoo Kang、Sasa Misailovic 和 Gagandeep Singh。《SynCode：通过语法增强实现大语言模型生成》。arXiv:2403.01632，2024年。[34] Leslie G. Valiant。《不到三次时间的通用上下文无关识别》。《《计算机与系统科学杂志》》，10(2)：308–315，1975年。[35] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Łukasz Kaiser 和 Illia Polosukhin。《注意力就是你所需要的一切》。载于《神经信息处理系统进展》，第30卷，2017年。[36] Brandon T. Willard 和 Rémi Louf。《面向大语言模型的高效引导式生成》。arXiv 预印本：arXiv:2307.09702，2023年。[37] Max Willsey、Chandrakana Nandi、Yisu Remy Wang、Oliver Flatt、Zachary Tatlock 和 Pavel Panchekha。《egg：快速且可扩展的等式饱和》。《ACM编程语言汇刊》，5(POPL)：1–29，2021年。

[38] Daniel H. Younger。在 n^3时间内对上下文无关语言的识别与句法分析。信息与控制，10(2)：189–208，1967年。