重新审视具有少量存在变量的 d-QBF

AndreasGrigorjew LAMSADE，法国国家科学研究中心 UMR7243，巴黎多菲纳—PSL大学，法国巴黎 75775**Michael Lampis** LAMSADE，法国国家科学研究中心 UMR7243，巴黎多菲纳—PSL大学，法国巴黎 75775

arXiv:2603.08826v1 [cs.CC] 9 Mar 2026

摘要

量化布尔公式（QBF）是可满足性问题（SAT）一个众所周知的困难推广，尤其从参数化复杂性的角度来看更是如此：对于大多数标准参数，该问题仍然难以处理。Eriksson 等人最近的一项研究 [IJCAI 24] 通过考察公式的命题部分为合取范式（CNF）的情形，并以存在量化变量的数量k 作为参数，探讨了这一问题。他们的主要结果之一表明，这个自然但迄今被忽视的参数确实能带来固定参数可解性，前提是同时限制给定合取范式中子句的最大元数 d 。遗憾的是，即使 d 是一个绝对常数，他们的算法对 k (2^{2^k})仍具有双指数 依赖。Eriksson 等人的研究仅补充了一个基于 SETH 的下界，说明不可能达到 $2^{O(k)}$ 依赖，因此这一巨大差距仍是一个悬而未决的问题。

本文的主要结果是通过证明双指数依赖是最优的来弥合这一差距，前提是 ETH 成立：即使对于元数为 4 的合取范式，含有 k 个存在变量的量化布尔公式也无法在 $2^{2^{o(k)}}|\varphi|^{O(1)}$ 时间内求解。与此相对，我们还考虑了进一步受限的情形，即仅含两个量词块的量化布尔公式（∀∃-QBF）。我们证明，在这种情形下，情况会显著改善：对于每个 $d \geq 3$ ，我们给出一个运行时间为 $k^{O_d\,(k^{d-1})}\,|\varphi|^{O(1)}$ 的算法，并在 ETH 假设下证明了一个下界，表明我们的算法几乎是最优的。

2012 年 ACM 学科分类 将 ccsdesc 宏替换为有效的宏

关键词与短语 量化布尔公式、固定参数算法、ETH

经费支持 本研究获得法国国家研究署项目 ANR-21-CE48-0022（S-EX-AP-PE-AL）的资助。

1 引言

量化布尔公式问题（QBF）如下：给定一个形如 $Qx_1Qx_2\ldots Qx_n\varphi(x_1,\ x_2,\ \ldots,\ x_n)$ 的公式，其中 $x_1,\ \ldots,\ x_n$ 是布尔变量，每个 Q 都是一个量词（$\exists$ 或 $\forall$），而 φ 是命题公式；我们要判断该公式的求值结果是否为真。QBF 是典型的 PSPACE 完全问题，显然是 SAT 的推广；SAT 是所有量词均为存在量词时的特例。由于其高度一般性，QBF 涵盖了一大类重要问题 [2, 3, 10, 18, 19]，但与此同时，无论在理论上还是实践中，它都比 SAT 难得多 [15]。

本文从参数化算法[1]的视角关注求解量化布尔公式的复杂性。从这一视角来看，量化布尔公式同样出了名地困难，大多数能使可满足性问题具有可处理算法的标准参数（例如树宽）对量化布尔公式并不适用（我们将在下文回顾部分相关文献）。尽管如此，Eriksson、Lagerkvist、Ordyniak、Osipov、Panolan 与 Rychlicki [7] 的一项值得注意的近期工作，通过考虑一个自然但迄今被忽视的参数——存在量化变量的数量——成功发现了该问题罕见的可处理性孤岛。其主要结果可概括如下：若一个量化布尔公式实例具有以合取范式给出的 φ ，且至多只包含 k 个存在变量，则该实例可在大约 m^{2^k}的时间内判定，其中

[1] 我们假定读者熟悉标准教材 [5] 中介绍的参数化复杂性基础知识。

m 是子句的数量；而若 φ 的所有子句的元数也至多为 $d = O(1)$，则该运行时间可改进到大约 $2^{2^k}|\varphi|^{O(1)}$。因此，前一结果是关于参数 k （存在变量的数量）的XP算法，而后一结果则是量化布尔公式关于一个自然参数的（罕见的！）FPT算法。

尽管 [7] 的结果令人耳目一新，因为它们揭示了一个罕见的自然参数，使得针对量化布尔公式可以获得非平凡的算法结果，但其双指数 运行时间是一个主要缺点。不过，对于无界元数公式而言，这是不可避免的，因为 [7] 给出了一种归约，证明在指数时间假设（ETH）成立的条件下， m^{2^k} 算法已是最优的。尽管如此，对于有界元数情形，同一项工作仅确立了一个弱得多的下界：除非强 指数时间假设为假，否则对于任何常数 c，都不存在能在 $c^k|\varphi|^{O(1)}$时间内解决元数为 3 的量化布尔公式的算法。这个下界相当不令人满意，因为它不仅与上界之间存在巨大的复杂度差距，而且还依赖于一个比无界元数合取范式下界所依赖的复杂性假设强得多（也不那么广为接受[20]）的假设。

我们的结果关于具有 k 个存在变量且其合取范式元数为常数的量化布尔公式能否在双指数依赖之外以更快的速度求解，这一问题在[7], 中被明确提出，作者指出，目前尚不清楚应当从哪个方向着手改进。本文的主要贡献是通过证明以下结论来解决这一问题：假设 ETH 为真，则不存在能在 $2^{2^{o(k)}}|\varphi|^{O(1)}$ 时间内求解量化布尔公式的算法，即使命题部分以合取范式给出且所有子句的元数至多为 4。由此，我们解决了 [7] 的主要问题，证明其算法在本质上已是最优的，并且在依赖更弱假设的同时，显著改进了其下界。

作为对这些结果的补充，我们还考虑量化布尔公式的一个有趣特例：输入公式只有两个量词块，我们将其记为 $\forall\exists$-量化布尔公式。考虑这一限制的动机是， [7] 给出的两个下界都已适用于这一特例，因此值得探究我们的下界是否也适用。

我们发现，当 d 有界时， $\forall\exists$-量化布尔公式与一般量化布尔公式的复杂度相当不同：我们证明，对所有 $d \geq 3$ ，我们可以在时间 $k^{O_d(k^{d-1})}$内求解 $\forall\exists$-量化布尔公式，其中 O_d 隐藏了依赖于 d的因子，从而显著改进了一般情形的双指数依赖。此外，我们证明我们的算法在本质上是最优的，至多在指数中相差一个对数因子：对所有 $d \geq 3$，若 $\forall\exists$-量化布尔公式能在时间 $2^{o(k^{d-1})}|\varphi|^{O(1)}$内求解，则ETH为假。请注意，这一下界也显著改进了 [7], 基于SETH的下界，后者证明元数为3的 $\forall\exists$-量化布尔公式不能在时间 c^k内求解，而我们在此情形下的下界为 $2^{o(k^2)}|\varphi|^{O(1)}$。这些结果与无界 d的情形形成了鲜明对比，在该情形中 [7] 给出了针对 $\forall\exists$-量化布尔公式的下界，基本上与他们针对一般量化布尔公式的算法相匹配。我们的结果表明，对于有界的 d，就参数 k而言，量化布尔公式严格难于 $\forall\exists$-量化布尔公式。

技巧 对于我们的主要（双指数）下界，直觉始于 [14], 中给出的一个归约：该归约以含有 n 个变量和 m 个子句的析取范式公式 φ 为输入，并通过添加 $\log m$ 个新的存在变量[2]，生成一个含有 $n + \log m$ 个变量的 $\forall\exists$-量化布尔公式实例 φ'。该归约保证以下强等价关系：对于 φ 和 φ' 的公共变量上的所有赋值 σ，当且仅当 φ 在 σ 下的求值为真时， φ'（在 σ 下化简后）可以由对这 $\log m$ 个新的存在变量的某种赋值满足。

[2] [14] 的归约实际上是从合取范式公式到 $\exists\forall$-量化布尔公式的，但如果我们对所有公式取补，此处的描述也是等价的。

这个归约几乎 适用于我们的目的，因为它生成的存在变量数量 k 为 $\log m$，因此它证明了任何算法都必须具有关于 k 的双指数复杂度（否则，我们的归约就能让我们在亚指数时间内判定 φ 是否有效；这对于 $\neg\varphi$ 等价于可满足性问题，从而推翻 ETH）。问题在于，[14] 的归约生成的合取范式具有非恒定 元数。更具体地说，该归约用一组子句表示 φ 中的每一项，这组子句包含所有 存在变量，但每一项对应的否定模式各不相同。因此，存在玩家的策略是选取这样一组子句，使其由全称量化文字满足；也就是说，选取 φ 中一个求值为真的项。

因此，扩展 [14] 的归约所面临的主要技术障碍，是设法降低元数。我们通过向公式 $O(\sqrt{m})$ 中添加新的全称变量来实现这一点，这些变量的取值应当完全由我们所添加的 $\log m$ 个存在变量的取值决定。具体而言，全称玩家“应当”将其中恰好两个变量设为真，而她所选择的这一对变量必须编码存在变量的赋值。如果确实如此，那么就可以很容易地将子句中每组 $\log m$ 个变量替换为一对新的全称变量。当然，问题在于，全称玩家可能会决定为这些变量赋予与我们的设想不一致的取值。因此，我们构造一个新的析取范式公式 ψ，用来表达“全称玩家在 $O(\sqrt{m})$ 个新变量中的某一个上作弊”。现在的关键思想是，我们可以对这个新的析取范式递归地应用我们的构造。由于 ψ 的大小为 $O(\sqrt{m})$，所以我们使用的新存在变量数量将以一个常数因子递减；因此，直到公式变得足够小、可以进行蛮力搜索为止，我们引入的存在变量总数是一个几何级数，其和为 $O(\log m)$。这意味着，在 ETH 假设下，我们得到了双指数下界。

对于 $\forall\exists$-量化布尔公式的算法结果，我们从一种类似于 [7], 将输入公式的子句划分为若干类的方法入手：同一类中的子句在所有存在变量上的取值一致（因此，对于至少包含一个全称变量的子句，共有$O_d(k^{d-1})$类）。然而，与 [7]试图在这类中寻找向日葵不同，我们试图寻找一个不共享任何全称变量的大型子句集合。我们的动机在于，不同于 [7], ，我们并不试图利用这种结构删除单个子句，而是希望通过概率方法论证：如果所有类中都存在这样的大型集合，那么全称玩家就拥有一种极佳的策略——她可以为自己的变量赋值，使得对于所有类，存在文字都保留在其中一个子句中。如果我们无法在某一类中找到足够大的集合，这就为该类的子句提供了一个小型击中集，我们将在其上执行一个简单的分支算法。

最后，对于证明上述算法几乎最优的下界，我们再次采用 [14], 的归约策略，但这次不引入 $\log m$ 个存在变量，而是引入 $m^{\frac{1}{d-1}}$ 个这样的变量。其思路是，一组 $d-1$ 个这样的变量将标识原公式中的一个项，再结合该项中的一个文字，我们便得到一个元数为 d 的子句。

相关工作 如前所述，从参数化复杂性的角度来看，量化布尔公式比可满足性问题难得多。值得注意的是，对于路径宽度为常数的公式，量化布尔公式是 PSPACE 完全的 [1]。如果限制量词交替的次数，那么以树宽为参数的问题确实属于固定参数可解 [4], ，但参数依赖是高度等于交替次数的指数塔 [9, 13, 14, 16]。这促使人们研究参数受到极严格限制的量化布尔公式，例如原始顶点覆盖；对于该参数，问题确实属于固定参数可解 [6], ，尽管参数依赖是双指数级的，而这一复杂度

在只有一次量词交替时已是最优的 [14]。这也促使人们研究考虑量词排序的结构参数 [6], 、后门类型参数 [17],, 以及介于顶点覆盖与树宽之间的参数 [8]。

2 预备知识

我们采用标准记号，并回顾一些基本概念。 d-QBF 是形如$\varphi = (Q_1 x_1) \dots (Q_n x_n)\psi$的公式，其中 $Q_i \in \{\exists, \forall\}$，且 ψ属于 d-合取范式。回顾一下，命题布尔公式属于 d-合取范式，是指它是由文字子句合取而成，其中每个子句至多包含 d 个文字。相反，公式属于 d-析取范式，是指它是由项析取而成，其中每个项是至多包含 d 个文字的合取。我们有时将子句（或项）视为文字集合，并将合取范式（或析取范式）公式视为子句（或项）的集合。我们用 $\forall\exists d$-QBF表示一种 d-QBF φ，其中 $Q_1, \dots, Q_{n-k} = \forall$ 以及 $Q_{n-k+1}, \dots, Q_n = \exists$。我们将简写 $\varphi = \forall \mathbf{y}\, \exists \mathbf{x} \psi$，其中记号 $\mathbf{x}$ 表示 $\mathbf{x}$ 是一个布尔变量元组（或向量）。对于 φ的（部分）变量的赋值 σ，我们用 $\varphi(\sigma)$表示将该赋值应用于 φ后得到的公式：若某个文字的取值为假，则将它从子句中删除；若某个文字的取值为真，则将该子句从公式中删除。我们还用 $\varphi(\mathbf{x})$表示 φ 是一个变量取自 $\mathbf{x}$ 的公式。我们有时将量化布尔公式视为两个玩家——全称玩家和存在玩家——之间的游戏。在第 i回合，相应的玩家根据 Q_i 为变量 x_i赋值，而存在玩家的目标是得到一个使公式取值为真的赋值。不难看出，当且仅当量化公式的取值为真时，存在玩家才有获胜策略。

指数时间假设（ETH）指出，存在 $c > 1$ 使得 3-SAT 无法在 c^n [12]时间内求解；而其强变体（SETH）指出，对于所有 $\varepsilon > 0$，都存在 k使得 k-SAT 无法在 $(2-\varepsilon)^n$ [11]时间内求解。本文中的所有下界都依赖于 ETH（人们认为 ETH 比 SETH 更可靠）。为简化记号，我们将采用 ETH 的一个稍微不那么正式的变体，即 3-SAT 无法在 $2^{o(n)}$时间内求解。此外，[12]的稀疏化引理意味着，若 ETH 为真，那么即使对于具有 $m = O(n)$ 个子句的实例，3-SAT 也无法在 $2^{o(n)}$时间内求解。

3 下界

本节将介绍两个基于 ETH 的下界。第一个（定理 1）表明，即使将范围限制为元数为 4 的合取范式， [7] 的双指数算法仍基本上是最优的。第二个（定理 2）表明，我们在第 4 节中介绍的算法也基本上是最优的，指数部分至多相差多对数因子。

3.1 双指数下界

在继续证明之前，我们先介绍一个有用的定义。我们定义函数B，其参数为一个由 n 个布尔变量组成的元组（$x_1, \dots, x_n$）和一个整数$i \in \{0, \dots, 2^n - 1\}$，并输出一个关于给定变量的子句，其否定模式表示整数 i。特别地，我们令 $B(0, \emptyset) = \emptyset$ 作为基本情形，并令$B(i, (x_1, \dots, x_n)) = B(\lfloor i/2 \rfloor, (x_1, \dots, x_{n-1})) \vee l_n$，其中 $l_n = x_n$在 i 为偶数时成立，否则为 $l_n = \neg x_n$。注意， B的定义方式使得子句 $B(i, (x_1, \dots, x_n))$ 为假，当且仅当我们赋给这些变量的值对应于整数 i的二进制表示。

▶ **定理 1.** 除非 *ETH* 为假，否则不存在这样的算法：给定一个含有 n 个变量和 k 个存在变量的 4-量化布尔公式实例 φ，能够在 $2^{2^{o(k)}} \cdot n^{O(1)}$ 时间内判定 φ 是否为真。

证明。 我们构造一个多项式时间算法，它以一个析取范式 $\psi(\mathbf{x})$ 为输入，该公式具有 n个变量和 m个子句，并输出一个 4-合取范式 $\varphi(\mathbf{x},\mathbf{y})$ 以及一个元组 $Q \in \{\exists,\forall\}^{|\mathbf{y}|}$，使得以下两个条件成立：

1. 等价公式：对于所有赋值 $\sigma : \mathbf{x} \to \{0,1\}$，我们有 $\psi(\sigma) \iff Q(\mathbf{y})\varphi(\sigma)$，2. $\mathbf{y}$ 中存在变量的数量有界：$|\{q_i \in Q \mid q_i = \exists\}| \leq 6\log_2(n+m) + O(1)$。

在继续之前，让我们解释为何这样的算法足以确立该定理。假设我们以一个含 n 个变量的 3-DNF 公式 ψ 作为输入，并希望判定 ψ 是否有效。一种做法是将其交给我们的算法作为输入，并生成一个等价的 4-QBF 实例，其中 ψ 的 n 个变量在排序中位于最前且被全称量化。该算法的运行时间为 $2^{2^{o(k)}} \cdot n^{O(1)}$，并且由于 $k = O(\log n)$，运行时间将为$2^{o(n)}$。然而，在 ETH 下，这样的算法是不可能的。

现在证明这样的算法确实存在。我们对 $n+m$ 使用归纳法，或者更准确地说，我们描述一种递归算法：当 $n+m$ 很小时（例如 $n+m < 2^{60}$，这里选取这个常数只是为了具体说明和便于表述，并未试图进行优化），算法有一个基本情形；否则，它会执行一个构造，并在规模更小的析取范式上调用同一个算法。

对于基本情形，假设 $n+m \leq 2^{60}$。由于 ψ 很“小”，我们可以通过蛮力法将其转换为 4-CNF。具体而言，我们按如下方式为每个使公式为假的赋值构造一个子句。枚举所有 2^n 个赋值 $\sigma : \mathbf{x} \to \{0,1\}$，如果 $\varphi(\sigma)$ 为假，则将其编码为合取范式公式：

$$\mathcal{C}_{\mathbf{x}} = \left\{ (l_1 \vee \cdots \vee l_n) \mid \exists \sigma : \mathbf{x} \to \{0,1\} : \ \varphi(\sigma) \text{ is False, and } l_i = \begin{cases} x_i, & \text{if } \sigma(x_i) = 0, \\ \neg x_i, & \text{else.} \end{cases} \right\}$$

换句话说，对于 ψ 的每个使公式为假的赋值，我们构造一个子句，使得在采用该赋值时，其中的所有文字都取假。令 $\varphi(\mathbf{x}) = \bigwedge_{c \in \mathcal{C}_{\mathbf{x}}} c$。条件 1 得到满足，因为当且仅当 $\varphi(\sigma)$ 为真时，$\psi(\sigma)$ 才为真；但新公式尚未属于 4-CNF，因为它至多包含 2^n 个子句，且每个子句的元数为 n。我们引入至多 $n2^n$ 个新的存在变量，以便将这些子句分解为元数为 4 的子句——特别地，当且仅当原公式为真时，存在一个新变量赋值使该公式得到满足。由于 $n < 2^{60}$，新公式现在具有 $O(1)$ 个存在变量，因此所有条件均得到满足。

现在假设 $n+m > 2^{60}$，并假设该算法对所有满足 $n'+m' < n+m$ 的对 $n', m' \in \mathbb{N}$ 都能正确运行。我们进一步假设 $m = 4^\ell$，否则我们可以重复子句，使 m 至多增加 4 倍（下面会解释为什么这不是问题）。

假设 $\psi(\mathbf{x})$ 的项编号为 $\mathbf{T} = \{T_0, \ldots, T_{m-1}\}$。Lampis 和 Mitsou [14]定义了一种算法，给定 $\psi(\mathbf{x})$，可构造一个与之等价、采用二进制编码的合取范式量化布尔公式 $\psi'(\mathbf{x}, \mathbf{y})$。具体而言，我们通过将每个项$T_i = (l_1 \wedge \cdots \wedge l_d)$ 替换为 d子句 $\mathbf{C}_i = (l_1 \vee B(i, \mathbf{y})) \wedge (l_2 \vee B(i, \mathbf{y})) \wedge \ldots \wedge (l_d \vee B(i, \mathbf{y}))$，来构造 ψ'。当所有变量 $\mathbf{y}$ 都经过存在量化时，公式 $\psi'(\mathbf{x}, \mathbf{y})$ 满足我们的条件（这一点不难看出，且已在 [14] 中详细证明）。然而，ψ' 的元数至少为 $\log_2 m + 1 > 4$。我们将修改这一构造以缩减元数。我们按以下顺序引入这些变量，并对其进行量化：

6 具有少量存在变量的 d-QBF 再探

- $\log_2 m$ 个存在变量 $\mathbf{y} = \{y_1, \ldots, y_{\log_2 m}\}$（与 [14] 中相同），$2\sqrt{m}$ 个全称变量 $\mathbf{z}^1 = \{z^1_1, \ldots, z^1_{\sqrt{m}}\}$，$\mathbf{z}^2 = \{z^2_1, \ldots, z^2_{\sqrt{m}}\}$，以及一个存在变量 w。

考虑一个双射 $\lambda: \{0, \ldots, m-1\} \to \{0, \ldots, \sqrt{m}-1\} \times \{0, \ldots, \sqrt{m}-1\}$。具体而言，我们使用自然双射 $\lambda(i) = (\lfloor \frac{i}{\sqrt{m}} \rfloor, i \bmod \sqrt{m})$。我们用 $\lambda(i)_1, \lambda(i)_2$ 来指代由 λ 返回的两个整数。我们定义一个量化布尔公式 $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{z}^1, \mathbf{z}^2, w)$，它由以下两部分组成。在将采用 4-CNF 的第一部分中，对于每一项 $T_i = (l_1 \wedge \cdots \wedge l_d)$，我们引入 d 个子句 $\mathsf{C}_i(\mathbf{x}, \mathbf{z}^1, \mathbf{z}^2, w) = (l_1 \vee \neg z^1_{\lambda(i)_1} \vee \neg z^2_{\lambda(i)_2} \vee w) \wedge \cdots \wedge (l_d \vee \neg z^1_{\lambda(i)_1} \vee \neg z^2_{\lambda(i)_2} \vee w)$。这里的想法是，变量向量 $\mathbf{z}^1$ 表示 $y_1, \ldots, y_{\log_2 m/2}$，而 $\mathbf{z}^2$ 表示 $y_{\log_2 m/2+1}, \ldots, y_{\log_2 m}$，其含义是：全称玩家“应当”仅当子句 $B(i, \mathbf{y})$ 的求值为假时，才将 $z^1_{\lambda(i)_1}$、$z^2_{\lambda(i)_2}$ 都设为真，也就是说，当对 $\mathbf{y}$ 的赋值是 i 的二进制表示时。在此假设下，我们在每个子句中使用一对全称变量，以替换 [14] 的构造中的 $\log m$ 个存在变量。我们还在各处加入新的存在变量 w，以便当全称玩家未按上述方式进行时，存在玩家仍能作出回应。

在第二部分中，我们要精确检查全称玩家是否按预期行棋，并定义一个析取范式公式如下：

$$\mathcal{D}'(\mathbf{y}, \mathbf{z}^1, \mathbf{z}^2, w) = (\neg w) \vee \bigvee_{i \in [\log_2 m/2], j \in \{0, \ldots, \sqrt{m}-1\}} (z^1_j \wedge l_{i,j}) \quad \bigvee_{i \in [\log_2 m/2], j \in \{0, \ldots, \sqrt{m}-1\}} (z^2_j \wedge l'_{i,j}),$$

其中，我们令 $\{l_{1,\ j\ \cdots\ j\ y_1\ \cdots\ y_{log_2 m/2}\ j\ \cdots\ j} = B(j, \cdot)\}$，并且令 $\{l'_{1,\cdot}, \ldots, l'_{\log_2 m/2,\cdot}\} = \{B(j, (y_{\log_2 m/2+1}, \ldots, y_{log_2 m}))\}$。

这里的思路如下：如果 $z^1_{j_1}$ 和 $z^2_{j_2}$ 都赋值为真，我们就考虑子句 $B(\sqrt{m} \cdot j_1 + j_2, \mathbf{y})$，其中 $\sqrt{m} \cdot j_1 + j_2$ 是通过拼接 j_1 和 j_2 的二进制表示得到的数。如果这个子句包含一个真文字，那么 $\mathcal{D}'$ 就包含一个真项，且该项不是 $\neg w$。因此，如果全称玩家想避免将 $\mathcal{D}'$ 中的某一项赋值为真，她至多可以将一对 $z^1_{j_1}, z^2_{j_2}$ 变量赋值为真，也就是使 $B(\sqrt{m} \cdot j_1 + j_2, \mathbf{y})$ 的取值为假的那些变量。

利用这两部分，我们定义

$$\varphi'(\mathbf{x}, \mathbf{y}, \mathbf{z}^1, \mathbf{z}^2, w) = \bigwedge_{i \in [m]} \mathcal{C}_i(\mathbf{x}, \mathbf{z}^1, \mathbf{z}^2, w) \wedge \mathcal{D}'(\mathbf{y}, \mathbf{z}^1, \mathbf{z}^2, w).$$

注意，φ' 尚未为合取范式。不过，在将其归约为合取范式之前，我们证明 φ' 满足条件 1。我们证明，对于所有 $\sigma: \mathbf{x} \to \{0, 1\}$ 我们有 $\psi(\sigma) \iff \exists \mathbf{y} \forall \mathbf{z}^1 \forall \mathbf{z}^2 \exists w \varphi'(\mathbf{x}, \mathbf{y}, \mathbf{z}^1, \mathbf{z}^2, w)$。

设 $\psi(\sigma)$ 被满足。也就是说，存在 $T_i = (l_1 \wedge \cdots \wedge l_d) \in \psi$，使得 $T_i(\sigma)$ 的所有文字均为真。选择 $\sigma': \mathbf{y} \to \{0,1\}$，若 $\neg y_j \in B(i, \mathbf{y})$，则 $\sigma'(y_j) = 1$，否则 $\sigma'(y_j) = 0$。换言之，我们选择 σ'，使得 $B(i, \mathbf{y})$ 为假，并且对于所有 $i' \neq i$，我们有 $B(i', \mathbf{y})$ 为真。我们分析 $\mathbf{z}^1$ 与 $\mathbf{z}^2$ 的所有可能选择的结果。

- 如果 $z^1_j = 0$ 对于所有 $j \in \{0, \ldots, \sqrt{m} - 1\}$ 或 $z^2_j = 0$ 对于所有 $j \in \{0, \ldots, \sqrt{m} - 1\}$，那么 C_i 中的每个子句对于所有 $j \in \{0, \ldots, m - 1\}$ 都满足。随后存在玩家选择 $w = 0$，D 也就满足了。如果 $z^1_{j_1} = z^2_{j_2} = 1$ 对于某个 $j_1, j_2 \in \{0, \ldots, \sqrt{m} - 1\}$，使得 $\lambda(i) \neq (j_1, j_2)$，那么 D' 中有一项（除 $(\neg w)$ 外）为真。特别是，如果 $\lambda(i)_1 \neq j_1$，则对于某个 $t \in [\log_2 m/2]$，项 $(z^1_{j_1} \wedge l_{t,j_1})$ 为真（对于 $\lambda(i)_2 \neq j_2$ 的情况也类似）。因此，存在玩家可以选择 $w = 1$，使所有 C_i 都满足。

- 如果 $z^1_{j_1} = z^2_{j_2} = 1$ 仅针对某个特定的变量对 j_1，$j_2 \in \{0, \dots, \sqrt{m}-1\}$，使得 $\lambda(i) = (j_1, j_2)$（且所有其他 z 变量均为假），那么 $\mathcal{C}_i$ 为真，因为 T_i 的所有文字都为真。对于所有 $i' \neq i$，$\mathcal{C}_{i'}$ 为真，因为其他 z 变量均为假。如果存在玩家将 $w = 0$ 设为真，我们也能满足 $\mathcal{D}$。

因此，$\exists \mathbf{y} \forall \mathbf{z}^1 \forall \mathbf{z}^2 \exists w \varphi'(\sigma)$ 为真，我们由此证明了一个方向。

现在设 $\exists \mathbf{y} \forall \mathbf{z}^1 \forall \mathbf{z}^2 \exists w \varphi'(\sigma)$ 为真。对于给定的 $\mathbf{y}$，考虑 $i = \sum_{j=1}^{\log_2 m} y_j 2^{j-1}$。我们进一步为数对 $j_1, j_2 \in \{0, \dots, \sqrt{m}-1\}$ 选择 $z^1_{j_1} = z^2_{j_2} = 1$，使得 $\lambda(i) = (j_1, j_2)$，（特别是 $j_1 = \lfloor \frac{i}{\sqrt{m}} \rfloor$ 和 $j_2 = i$ 模 $\sqrt{m}$），并将所有其他 z 变量设为 0。由于 $\varphi'(\sigma)$ 为真，对于 $\mathbf{z}^1$，$\mathbf{z}^2$ 的这一赋值，它也必须为真。但对于这一赋值，存在玩家被迫将 w 设为假，否则 $\mathbf{D}'$ 就没有真项。因此，对于 $\mathbf{z}^1, \mathbf{z}^2$，的这一选择，$\mathbf{C}_i$ 必须为真，所以项 T_i 中的所有文字都必须为真。因此，$\varphi(\sigma)$ 得到满足。

公式 φ' 现在由一个 4-CNF 部分 $\mathbf{C}_i(\mathbf{x}, \mathbf{z}^1, \mathbf{z}^2, w)$ ($i \in \{0, \dots, m-1\}$) 和一个 DNF 部分 $\mathbf{D}'(\mathbf{y}, \mathbf{z}^1, \mathbf{z}^2, w)$ 组成。注意，$\mathbf{D}'$ 至多包含 $n' \leq 2\sqrt{m} + \log_2 m + 1$ 个变量和 $m' \leq 2\sqrt{m}\log_2 m + 1$ 个项。如果我们一开始就增大 m，使其成为 4 的幂，这些界就变为 $n' \leq 4\sqrt{m} + \log_2 m + 3$ 和 $m' \leq 4\sqrt{m}\log_2 m + 4\sqrt{m} + 1$。因此，我们有 $n' + m' \leq 4\sqrt{m}\log_2 m + 8\sqrt{m} + log_2 m + 3 \leq 16\sqrt{m} + n\log_2(m+n)$。然而，当 $m + n > 2^{60}$ 时，我们有 $16\sqrt{m} + n\log_2(m+n) < (m+n)^{2/3}$。因此，利用 $\mathbf{D}'$ 上的归纳法（即递归调用我们的算法），我们可以将其替换为一个等价的 4-CNF 量化布尔公式 $\mathbf{D}(\mathbf{y}, \mathbf{z}^1, \mathbf{z}^2, w, \mathbf{y}')$，其中至多包含 $6\log_2((m+n)^{2/3}) + O(1) = 4\log_2(m+n) + O(1)$ 个存在变量。最后，我们定义

$$\varphi(\mathbf{x}, \mathbf{y}, \mathbf{z}^1, \mathbf{z}^2, w, \mathbf{y}') = \bigwedge_{i \in [m]} \mathcal{C}_i(\mathbf{x}, \mathbf{z}^1, \mathbf{z}^2, w) \wedge \mathcal{D}(\mathbf{y}, \mathbf{z}^1, \mathbf{z}^2, w, \mathbf{y}')$$

公式 φ 是一个 4-QBF-CNF，且最多使用 $\log_2 m + (4\log_2(m+n) + O(1)) \leq 6\log_2(m+n) + O(1)$ 个额外的存在变量。因此，条件 1 和条件 2 均得到满足。◀

3.2 双量词块的近乎紧下界

与上一节类似，我们定义一个有用的函数，将整数转换为子句。由于我们将构造一个归约族，每个 $d \geq 3$ 对应一个归约，因此需要一族这样的函数。设 $t \geq 1$ 为一个整数，$\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^t$ 为 t 个布尔变量元组，编号为 $x^j_0, \dots, x^j_{n-1}$，其中 $j \in [t]$，且 i 是 $\{0, \dots, n^t - 1\}$ 中的一个整数。我们定义 $B_1(i, \mathbf{x}^1) = \neg x^1_i$，以及 $B_t(i, \mathbf{x}^1, \dots, \mathbf{x}^t) = B_{t-1}(\lfloor \frac{i}{n} \rfloor, \mathbf{x}^1, \dots, \mathbf{x}^{t-1}) \vee \neg x^t_i \bmod n$。理解这个函数的另一种方式是：如果我们想用一个子句表示整数 $i < n^t$，并使用来自 t 个、每个含有 n 个变量的元组中的变量，那么我们就将 i 写成 n 进制数，每个数位给出一个元组中变量的索引，然后将该变量取反后加入子句。直观地说，这个函数很有用，因为随着元数增加，它能让我们用更少的变量来表示相同的整数。

▶ **定理 2.** 对于所有 $d \geq 3$，除非 *ETH* 为假，否则不存在能在 $2^{o(k^{d-1})} \cdot n^{O(1)}$ 时间内求解含有 k 存在变量的 $\forall\exists$ d-量化布尔公式的算法。

证明。 我们描述一种多项式时间算法，其输入为析取范式 $\psi(\mathbf{x})$，含有 n个变量和 $m = O(n)$ 个项，并生成一个 d-合取范式 $\varphi(\mathbf{x}, \mathbf{y})$，使其满足以下两个条件：

1. 等价公式：对于所有赋值 σ： $\mathbf{x} \to \{0, 1\}$ 我们有 $\psi(\sigma) \iff \exists \mathbf{y} \varphi(\sigma)$,

2. y 中存在变量的数量 的上界为 $O(m^{1/(d-1)})$。

继续之前，我们先解释为什么这样的算法能够证明该定理。假设给定一个含有 n 个变量和 $m = O(n)$ 个子句的 3-DNF 公式 ψ，并且要判断 ψ 是否有效。利用上述归约，存在一个等价的 $\forall\exists d$-量化布尔公式 φ，其中有 $O(n^{1/(d-1)})$ 个存在变量。因此，针对 $\forall\exists d$-量化布尔公式、运行时间为 $2^{o(k^{d-1})}$ 的算法，可以在时间 $2^{o(n)}$ 内判断给定的 3-DNF 公式是否有效，这与 ETH 矛盾。

不失一般性，我们可以假设 ψ 的项数 m 满足 $m^{\frac{1}{d-1}}$ 为整数，因为通过添加重复子句，最多进行 $O(n)$ 次，就能达到这一点。我们引入 $(d-1)\cdot m^{1/(d-1)}$ 个存在变量 $\mathbf{y}^1, \ldots \mathbf{y}^{d-1}$，其中每个 $\mathbf{y}^i$ 的大小为 $m^{1/(d-1)}$。将 ψ 中的项编号为 $\mathbf{T} = \{T_0, \ldots, T_{m-1}\}$。考虑自然双射 $\lambda\colon \{0, \ldots, m-1\} \to \{0, \ldots, m^{1/(d-1)-1}\}^{(d-1)}$，它对每个 i 的定义方式是将 i 写成 $m^{1/(d-1)}$ 进制。对于每个项 $T_i = (l_{j_1} \wedge \cdots \wedge l_{j_d})$，我们定义子句 $\mathsf{C}_i(\mathbf{x}, \mathbf{y}^1, \ldots, \mathbf{y}^{d-1}) = (l_{j_1} \vee B_{d-1}(i, \mathbf{y}^1, \ldots, \mathbf{y}^{d-1})) \wedge \cdots \wedge (l_{j_{d'}} \vee B_{d-1}(i, \mathbf{y}^1, \ldots, \mathbf{y}^{d-1}))$。注意，这些子句的元数均为 d。

此外，我们还加入子句 $\mathcal{D}(\mathbf{y}^1, \ldots, \mathbf{y}^{d-1}) = (y_1^1 \vee \cdots \vee y_{m^{1/(d-1)}}^1) \wedge \cdots \wedge (y_1^{d-1} \vee \cdots \vee y_{m^{1/(d-1)}}^{d-1})$。注意，这些子句的元数为 $m^{1/(d-1)}$，过高了，但我们将在最后解决这个问题。

我们定义

$$\varphi(\mathbf{x}, \mathbf{y}^1, \ldots, \mathbf{y}^{d-1}) = \bigwedge_{j \in \{0,\ldots,m^{1/(d-1)}-1\}} \mathcal{C}_j(\mathbf{x}, \mathbf{y}^1, \ldots, \mathbf{y}^{d-1}) \wedge \mathcal{D}(\mathbf{y}^1, \ldots, \mathbf{y}^{d-1}).$$

注意，φ 满足条件 2，因为它使用了 $(d-1)\cdot m^{1/(d-1)} = O(m^{1/(d-1)})$ 个变量。现在我们证明 φ 满足条件 1。设 $\psi(\sigma)$ 为真，也就是说，项 $T_i \in \mathcal{C}$ 为真。此时，我们可以选取 $y^1_{\lambda(i)_1} = \cdots = y^{d-1}_{\lambda\ i\ ()_{d-1}} = 1$，并将所有其他变量 y 赋值为 0。于是，$\mathcal{D}$ 中的每个子句都得到满足。此外，$\mathcal{C}_j$ 中带有 $j \neq i$ 的每个子句也都得到满足，因为我们的赋值使 $B_{d-1}(j, \mathbf{y}^1, \ldots, \mathbf{y}^{d-1})$ 对所有 $j \neq i$ 都得到满足。另外，$\mathcal{C}_i$ 也得到满足，因为 T_i 得到满足。因此，$\varphi(\sigma)$ 为真。

令 $\varphi(\sigma)$ 为真。由于 $\mathcal{D}(\mathbf{y}^1, \ldots\ldots, \mathbf{y}^{d-1})$ 为真，因此每个 $\mathbf{y}^i$ 中至少有一个变量为真。令 $y^1_{i_1}, \ldots\ldots, y^{d-1}_{i_{d-1}}$ 为这些变量，并令 i 为将 $i_1, \ldots\ldots, i_{d-1}$ 拼接起来后，按 $m^{1/(d-1)}$ 进制解释所得的数。注意到子句$B_{d-1}(i, \mathbf{y}^1, \ldots\ldots, \mathbf{y}^{d-1})$ 被 σ 赋值所满足，因此若 φ 被满足，$\mathcal{C}_i$ 中的子句就会因 T_i中的文字而为真。因此，原析取范式 ψ 中的一个项对 σ 为真。

最后，我们来说明如何解决$\mathcal{D}$ 中子句元数过大的问题。我们有d 个元数为 $m^{1/(d-1)}$ 的子句。可以添加新的存在变量，并使用标准技巧，将一个长子句（$l_1 \vee l_2 \vee l_3 \vee \ldots \vee l_t$）替换为两个子句（$l_1 \vee l_2 \vee z$）$\wedge (\neg z \vee l_3 \vee \ldots \vee l_t)$。反复应用这一方法所需的新变量数量至多为 $dm^{1/(d-1)}$，因此存在变量总数仍为 $O(m^{1/(d-1)})$。◀

4 几乎紧的 $\forall\exists d$-量化布尔公式算法

本节将证明以下定理。

▶ **定理 3.** 存在一种算法，给定一个 $\forall\exists d$-量化布尔公式 $\varphi = \forall\mathbf{y}\exists\mathbf{x}\ \psi$，可在时间 $O\left(k^{O_d(k^{d-1})}\right) \cdot |\varphi|^{O(1)}$内验证 φ 是真还是假，其中 $k = |\mathbf{x}|$ 是存在变量的数量。

根据定理 2，该运行时间几乎是紧的（即，指数中仅差一个k 的对数因子）。我们给出一种递归算法，并通过归纳法证明其正确性和时间复杂度。

我们的算法如下开始：给定 d-合取范式 ψ，我们根据存在文字将子句划分为若干组。更准确地说，对所有集合 $C \subseteq \{x, \neg x \mid x \in \mathbf{x}\}$，定义以下两个集合。

$$G_C^\psi = \left\{ c \in \psi \mid c \cap \{x, \neg x \mid x \in \mathbf{x}\} = C \right\},$$
$$\mathcal{S}_C^\psi = \left\{ c \setminus C \mid c \in G_C^\psi \right\}.$$

我们将集合 G_C^ψ 称为子句组。它们构成子句的一个划分 $\mathcal{P}_\psi$，也就是说，G_C^ψ 恰好包含存在部分为子句 C 的那些子句。集合 $\mathcal{S}_C^\psi$ 由 G_C^ψ 中限制在其全称变量上的子句组成。我们将 C 称为子句 $c \in \psi$ 的**存在核心**（或简称核心），其中 $c \in G_C$。

在算法中，我们使用一个“大小阈值” $X = X(k, d) = 2^d d \ln k$，其取值的选择将在下面的证明中说明。我们的算法是递归的。基本情形定义如下：对于每个核心 C，集合族 $\mathcal{S}_C^\psi$ 要么由空子句组成（当 ψ 有一个仅含存在变量的子句时，就会出现这种情况），要么包含大量两两不相交的子句（即至少有 X 个彼此不共享全称变量的子句）。算法的第一步是在每个 $\mathcal{S}_C^\psi$ 中，尝试用贪心方式构造一个极大的两两不相交子句集合。如果成功，也就是说，我们为所有 $\mathcal{S}_C^\psi$ 都找到了大小为 X 的此类集合，那么就处于基本情形。接着，我们从所有子句中删除全部全称变量（也就是用每个子句的核心替换它），并且仅当所得的（存在量化）公式可满足时返回真。这个基本情形可在时间 $2^k |\psi|^{O(1)}$ 内求解。我们将基于概率方法给出论证，说明这种归约为何正确：全称玩家有策略将公式化简为其存在核心，而这显然是对她最优的。

如果对某个$\mathcal{S}_C^\psi$，互不相交集合的贪心构造未能成功，我们就处于递归步骤中，并观察到我们有一个小的击中集 H，即一个大小至多为 dX 的全称变量集合，它与$\mathcal{S}_C^\psi$ 的所有子句相交（该集合仅由互不相交的子句的极大集合之并构成）。在这种情况下，我们考虑所有可能的赋值σ： $H \to \{0,1\}$，并对所有公式 $\psi(\sigma)$运行该算法。如果某次递归调用返回该公式为假，我们就返回假。关键的观察是：由于我们有一个击中集，在每次递归调用中，我们都降低了$\mathcal{S}_C^\psi$ 的所有子句的元数，从而能够界定递归深度。

在整个过程中，我们将假设 ψ 不包含仅由全称变量构成的子句，因为这样的子句要么是重言式，可以安全删除，要么全称玩家可以找到获胜策略并使该公式为假。由此可知，所有核均为非空。

4.1 基本情形与概率论证

如前所述，我们算法的第一步是针对每个 $\mathcal{S}_C^\psi$，贪心地构造一个极大子句集，其中的子句不共享任何全称变量。对于基本情形，假设这一步成功地针对每个集合 $\mathcal{S}_C^\psi \neq \{\emptyset\}$，构造出这样一个至少包含 X 个子句的集合，且这些子句仅共享其核。

利用概率方法，我们为全称玩家提供一种最优策略。给定公式 $\psi(\mathbf{y}, \mathbf{x})$，我们通过将 ψ 的每个子句替换为一个新子句，构造一个新公式 $\psi|_{\mathbf{x}}(\mathbf{x})$：

$$\psi \ni (l_{\mathbf{x},1} \vee \cdots \vee l_{\mathbf{x},i} \vee l_{\mathbf{y},1} \vee \cdots \vee l_{\mathbf{y},j}) \mapsto (l_{\mathbf{x},1} \vee \cdots \vee l_{\mathbf{x},i}) \in \psi|_{\mathbf{x}},$$

其中 $l_{\mathbf{x},\,1}$，……，$l_{\mathbf{x},\,i} \in \{x,\ \neg x \mid x \in \mathbf{x}\}$ 以及 $l_{\mathbf{y},\,1}$，……，$l_{\mathbf{y},\,j} \in \{y,\ \neg y \mid y \in \mathbf{y}\}$。也就是说，我们将 ψ 的所有子句简化为其核心文字（然后当然只保留每个结果子句的一份副本）。

▶ **引理 4.** 设 $\psi(\mathbf{y},\mathbf{x})$ 满足如下性质：对所有 $S_C^\psi \neq \{\emptyset\}$，在 S_C^ψ 中存在一个至少包含 X 个互不相交的子句、且这些子句不共享任何全称变量的集合。则我们有

$$\forall \mathbf{y} \exists \mathbf{x} \psi \iff \exists \mathbf{x} \psi|_{\mathbf{x}}.$$

证明。我们注意到其中一个方向是平凡的，即 $\exists \mathbf{x}\psi|_{\mathbf{x}} \Rightarrow \forall \mathbf{y}\exists \mathbf{x}\psi$，因为如果存在玩家能够满足所有子句的核，她显然就能满足原公式。因此，我们想要证明 $\neg\exists \mathbf{x}\psi|_{\mathbf{x}} \Rightarrow \neg\forall \mathbf{y}\exists \mathbf{x}\psi$ 或等价地 $\forall \mathbf{x}\neg\psi|_{\mathbf{x}} \Rightarrow \exists \mathbf{y}\forall \mathbf{x}\neg\psi$。于是假设 $\psi|_{\mathbf{x}}$ 不可满足，并且我们想为全称玩家找到一个获胜策略。我们证明存在一种策略，其结果是将公式化简为 $\psi|_{\mathbf{x}}$，其方法是证明：若我们选取一个随机赋值，则该结果具有正概率。

均匀随机地选择一个赋值 $\sigma : \mathbf{y} \to \{0,1\}$。也就是说，每个 $y \in \mathbf{y}$ 独立地以 1/2 的概率被赋值为真。令 $C \subseteq \{x.\neg x \mid x \in \mathbf{x}\}$，并令$\mathcal{S}_C^\psi \neq \{\emptyset\}$。根据假设，族 $\mathcal{S}_C^\psi$ 至少包含 X 个两两不相交的集合。也就是说，组 G_C^ψ 包含 X 个子句 $C_1, \ldots, C_{\lceil X \rceil}$，它们的全称变量两两不相交。

给定一个赋值 σ，若赋值不满足 c，则称 $c \in \psi(\sigma)$ 为活跃。若赋值满足 c，则称子句非活跃。如果 c 非活跃，就将其从公式中删除；如果它是活跃的，则将其化简为它的核，因为我们已经为所有全称变量赋了值。

对于每个 C_j，都有 $\Pr[C_j(\sigma) \text{ 为活跃}] \geq (1/2)^d$，因为要使子句处于活跃状态，只需让 σ 将所有（至多 d 个）全称文字设为假。因此，我们有 $\Pr[C_j(\sigma) \text{ 为非活跃}] \leq (1 - (1/2)^d)$，并且

$$\begin{aligned}\Pr[\text{all } C_j(\sigma) \text{ are inactive}] &\leq (1-(1/2)^d)^{2^d d \ln k} \\ &< e^{-d\ln k} \\ &= \frac{1}{k^d},\end{aligned}$$

其中我们代入了 $X = 2^d d\ln\, k$，并利用了这样一个事实：如果子句之间没有共享变量，那么一个子句处于活跃状态的事件与其他子句相互独立。

现在，我们通过并界将这个不等式应用于所有组 $\mathcal{S}_C^\psi$。如果某个组的核大小为 d，那么整个子句由存在变量构成。在这种情况下，集合族 $\mathcal{S}_C^\psi$ 由空子句构成。因此，非平凡组的数量至多为 $\binom{k}{d-1} \leq k^{d-1}$，因为每个核至多包含 k 个存在变量中的 $d-1$ 个。利用这个界，我们得到

$$\begin{aligned}&\Pr\left[\bigvee_{C\subseteq\{x,\neg x \mid x\in\mathbf{x}\},\mathcal{S}_C^\psi\neq\{\emptyset\}} \text{max. disjoint set of } \mathcal{S}_C^\psi \text{ has only inactive members}\right] \\ &\leq \sum_{C\subseteq\{x,\neg x \mid x\in\mathbf{x}\},\mathcal{S}_C^\psi\neq\{\emptyset\}} \Pr[\text{max. disjoint set of } \mathcal{S}_C^\psi \text{ has only inactive members}] \\ &< k^{d-1}\cdot\left(\frac{1}{k^d}\right) = \frac{1}{k}.\end{aligned}$$

因此，根据概率方法，存在一种全称玩家的策略，可将 ψ totheir 中的所有子句缩减为其核。◀

4.2 递归步骤与分析

在算法的递归步骤中，存在一个组 G_C^ψ，其集合族 $\mathcal{S}_C^\psi$ 包含一个大小为 $|H| \leq dX$ 的击中集 H，也就是说，$\mathcal{S}_C^\psi$ 中的每个子句都包含 H 中的一个全称变量。这是因为，如果我们不处于基本情形，那么我们为 $\mathcal{S}_C^\psi$ 构造的极大互不相交子句集合大小至多为 X，并且与 $\mathcal{S}_C^\psi$ 中的所有其他子句都相交。

算法现在枚举所有赋值 σ： $H \to \{0, 1\}$，并针对每个 $\psi(\sigma)$ 递归调用自身。

▶**引理 5.** 上述算法的递归树至多有 $k^{O_d(k^{d-1})}$ 个叶节点。

证明。 我们将公式 ψ 的权重定义为每个组的子句中全称变量的最大数量，作为进展度量：

$$w(\psi) = \sum_{C \subseteq \{x, \neg x | x \in \mathbf{x}\}} \max_{V \in \mathcal{S}_C^\psi} |V|.$$

运行算法之前，所有子句的元数至多为 d。权重非零的组数至多为 k^{d-1}，因为在这样的组中，所有子句都包含至少一个全称变量，因此至多包含 k 个存在变量中的 $d-1$ 个。所以，初始时$w(\psi) \leq d \cdot k^{d-1}$。

令 H 为 $\mathcal{S}_C^\psi$ 和 σ 的击中集： $H \to \{0, 1\}$ 是任意赋值。于是有$w(\psi(\sigma)) < w(\psi)$。显然，为击中集中的每个变量赋一个布尔值后，组中所有子句的元数都会降低。

总体而言，在递归树中，我们以至多 $2^{|H|} \leq 2^{dX}$ 的分支度进行分支，且树的深度至多为 $w(\psi) \leq$ ( )，因此树的叶节点数至多为 $2^{d^2 X k d k^{d-1}} = k^{O_d(k^{d-1})}$ ( )(.)()。◀

综合以上内容，我们便得到本节定理的证明。

定理 3 的证明。算法如下：首先，我们将ψ 的子句划分为 G_C^ψ 组，并计算全称部分$\mathcal{S}_C^\psi$。对于每个全称部分（若它并非仅由空子句组成），我们通过从空集合开始并任意选取子句加入集合，只要仍有可能就继续选取，从而贪心地构造一个变量两两互不相交的子句集合。如果在某个时刻，该集合中至少包含 X 个子句，我们就转到下一个组$\mathcal{S}_C^\psi$。此过程可在多项式时间内执行，并且可能以两种方式终止：

1. 算法成功地为每个组$\mathcal{S}_C^\psi$ 构造了一个包含 X 个互不相交子句的集合。在这种情况下，我们构造 $\psi|_{\mathbf{x}}$ 并检查它是否可满足（遍历所有 2^k种赋值）。如果可满足，就返回真；否则返回假。引理 4 证明了这一步的正确性。

2. 该算法找到一个组 $\mathcal{S}_C^\psi$ 以及一个由少于 X 个子句组成的集合，其中这些子句属于 $\mathcal{S}_C^\psi$，并且 $\mathcal{S}_C^\psi$ 中的所有其他子句都与该集合中的某个子句共享一个全称变量。于是，该集合中子句的至多dX 个变量构成一个击中集 H，也就是说，$\mathcal{S}_C^\psi$ 中的所有子句都至少包含 H中的一个变量。该算法对 H 中的变量的所有赋值进行分支，并且当且仅当某个分支返回假时，才返回假。该分支过程是正确的，且产生的叶节点数量由引理 5 所界定。

综合以上结果，运行时间由递归树的叶节点数量所主导，从而得到所述运行时间。◀

5 结论与未解决问题

本文的主要贡献是通过证明即使对于元数为 4 的量化布尔公式，运行时间对 k 的双指数依赖也是必要的，从而解决了 [7] 的未解决问题。这留下了一个引人关注的未解决问题：元数为 3 的量化布尔公式情形。我们猜想，可以将我们的下界推广到这一情形。

此外，我们证明了，以 k 为参数的 d-量化布尔公式的复杂度在只允许两个量词块（$\forall\exists$-量化布尔公式）时确实会显著降低，因为在这种情况下，我们有一个算法，其对参数的依赖是关于 k^{d-1} 的指数级，而非关于 k 的双指数级。另一方面，我们的双指数下界需要数量较少但无界的量词交替才能成立（大约为 $\log\log n$ 次，因为构造的每一步都会将大小为 m 的公式替换为大小为 $O(\sqrt{m})$ 的公式）。这留下了一个问题：少量但固定次数的量词交替会产生什么结果？也就是说，求解 $\forall\exists\forall\exists$-量化布尔公式的复杂度是否已经对 k 呈双指数级？

参考文献

1 Albert Atserias 和 Sergi Oliva。有界宽度量化布尔公式是 PSPACE 完全的。《计算机与系统科学杂志》，80(7):1415–1429，2014年。网址：https://doi.org/10.1016/j.jcss.2014.04.014, doi:10.1016/J.JCSS.2014.04.014。 2 Olaf Beyersdorff、Mikolás Janota、Florian Lonsing 和 Martina Seidl。量化布尔公式。载于 Biere等人 [3], ，第1177–1221页。 doi:10.3233/FAIA201015。 3 Armin Biere 、 Marijn Heule 、 Hans van Maaren 和 Toby Walsh （编）。《可满足性手册》第二版，人工智能与应用前沿第336卷。IOS Press，2021年。 doi:10.3233/FAIA336。 4 Hubie Chen。量化约束满足与有界树宽。载于 Ramón López de Mántaras 和 Lorenza Saitta（编），《第16届欧洲人工智能会议论文集》，ECAI 2004，包括智能系统杰出应用会议 PAIS 2004，西班牙瓦伦西亚，2004年8月22日至27日，第161–165页。IOS Press，2004年。 5 Marek Cygan、Fedor V. Fomin、Lukasz Kowalik、Daniel Lokshtanov、Dániel Marx、Marcin Pilipczuk、Michal Pilipczuk 和 Saket Saurabh。《参数化算法》。施普林格，2015年。 doi:10.1007/978-3-319-21275-3。 6 Eduard Eiben、Robert Ganian 和 Sebastian Ordyniak。将分解参数用于量化布尔公式：注意前缀！《计算机与系统科学杂志》，110:1–21，2020年。网址：https://doi.org/10.1016/j.jcss.2019.12.005, doi:10.1016/J.JCSS.2019.12.005。 7 Leif Eriksson、Victor Lagerkvist、Sebastian Ordyniak、George Osipov、Fahad Panolan 和 Mateusz Rychlicki。用少量存在变量求解量化布尔公式。载于第三十三届国际人工智能联合会议论文集，IJCAI 2024，韩国济州，2024年8月3日至9日，第1889–1897页。ijcai.org，2024年。网址：https://www.ijcai.org/proceedings/2024/209。 8 Johannes Klaus Fichte、Robert Ganian、Markus Hecher、Friedrich Slivovsky 和 Sebastian Ordyniak。面向结构的下界与拓宽量化布尔公式可处理性的边界。载于第38届ACM/IEEE计算机科学逻辑年度研讨会，LICS 2023，美国马萨诸塞州波士顿，2023年6月26日至29日，第1–14页。IEEE，2023年。 doi:10.1109/LICS56636.2023.10175675。 9 Johannes Klaus Fichte、Markus Hecher 和 Andreas Pfandler。有界树宽量化布尔公式的下界。载于 Holger Hermanns、Lijun Zhang、Naoki Kobayashi 和 Dale Miller（编），LICS 2020：第35届ACM/IEEE计算机科学逻辑年度研讨会，

德国萨尔布吕肯，2020年7月8日至11日，第410–424页。美国计算机协会，2020年。doi:10.1145/3373718.3394756。 10 Enrico Giunchiglia、Paolo Marin 和 Massimo Narizzano。量化布尔公式推理。载于 Biere等人 [3], ，第1157–1176页。 doi:10.3233/FAIA201014。 11 Russell Impagliazzo 和 Ramamohan Paturi。关于 k-SAT 的复杂性。《计算机与系统科学杂志》，62(2):367–375，2001年。网址： https://doi.org/10.1006/jcss.2000.1727, doi:10.1006/JCSS.2000.1727。 12 Russell Impagliazzo、Ramamohan Paturi 和 Francis Zane。哪些问题具有强指数级复杂性? 《计算机与系统科学杂志》，63(4):512–530，2001年。网址： https://doi.org/10.1006/jcss.2001.1774, doi:10.1006/JCSS.2001.1774。 13 Michael Lampis、Stefan Mengel 和 Valia Mitsou。将量化布尔公式作为 Courcelle定理的替代方案。载于 Olaf Beyersdorff 和 Christoph M. Wintersteiger（编），《可满足性测试理论与应用——SAT 2018——第21届国际会议》，SAT 2018，作为逻辑联合会议 FloC 2018 的一部分举行，英国牛津，2018年7月9日至12日，论文集，《计算机科学讲义》第10929卷，第235–252页。施普林格，2018年。 doi:10.1007/978-3-319-94144-8_15。 14 Michael Lampis 和 Valia Mitsou。重新探讨带有一次量词交替的树宽。载于 Daniel Lokshtanov 和 Naomi Nishimura（编），第12届参数化与精确计算国际研讨会，IPEC 2017，2017年9月6日至8日，奥地利维也纳，《信息学论文集》第89卷，第26:1–26:12页。达格斯图尔城堡—莱布尼茨信息学中心，2017年。网址: https://doi.org/10.4230/LIPIcs.IPEC.2017.26, doi:10.4230/LIPICS.IPEC.2017.26。 15 Paolo Marin、Massimo Narizzano、Luca Pulina、Armando Tacchella 和 Enrico Giunchiglia。QBF评测十二年：QSAT是 PSPACE 困难的，事实也确实如此。《基础信息学》，149(1-2):133–158，2016年。 doi:10.3233/FI-2016-1445。 16 Guoqiang Pan 和 Moshe Y. Vardi。PSPACE 内部的固定参数层级。载于第21届IEEE计算机科学逻辑研讨会（LICS 2006），2006年8月12日至15日，美国华盛顿州西雅图，论文集，第27–36页。IEEE计算机学会，2006年。doi:10.1109/LICS.2006.25。 17 Marko Samer 和 Stefan Szeider。量化布尔公式的后门集。《自动推理杂志》，42(1):77–97，2009年。网址： https://doi.org/10.1007/s10817-008-9114-5, doi:10.1007/S10817-008-9114-5。 18 Roberto Sebastiani 和 Armando Tacchella。模态逻辑与描述逻辑的 SAT技术。载于 Biere等人 [3], ，第1223–1266页。 doi:10.3233/FAIA201016。 19 Ankit Shukla、Armin Biere、Luca Pulina 和 Martina Seidl。量化布尔公式应用综述。载于第31届IEEE人工智能工具国际会议，ICTAI 2019，美国俄勒冈州波特兰，2019年11月4日至6日，第78–84页。IEEE，2019年。doi:10.1109/ICTAI.2019.00020。 20 R. Ryan Williams。计算复杂性的若干估计概率。载于 Bernhard Steffen 和 Gerhard J. Woeginger（编），《计算与软件科学：现状与展望》，《计算机科学讲义》第10000卷，第9–26页。施普林格，2019年。doi:10.1007/978-3-319-91908-9_2。