arXiv:2601.16076v1 [cs.CC] 22 Jan 2026

DNF 公式可以通过相对误差进行高效测试

Xi Chen 哥伦比亚大学

William Pires Toniann Pitassi 哥伦比亚大学

哥伦比亚大学

Rocco A. Servedio 哥伦比亚大学

2026年1月23日

摘要

我们给出一个$\mathrm{poly}(s, 1/\varepsilon)$次查询算法，用于测试一个未知且任意的函数 $f : \{0,1\}^n \to \{0,1\}$ 是否为一个 s项析取范式（DNF）。我们采用了布尔函数性质测试中颇具挑战性的相对误差框架；该框架近期已在多项研究中被提出并加以探讨 [CDH+25b、CPPS25a、CPPS25b、CDH+25a]。这给出了第一个例子，展示了一类丰富且自然的函数类：它们可以依赖超常数个变量，却仍能在相对误差模型下以常数查询复杂度高效地进行性质测试。

使我们的方法成为可能的一个关键新要素，是将任意 s 项析取范式公式分解为由各项构成的“局部簇”的一种新颖分解。我们的结果表明，即使没有明确给出 s 项析取范式，也可以有效地利用这种新分解来设计算法；我们相信，这种分解在其他情境中也可能有应用。

目录

1 引言 **1**

1.1 我们的主要结果：DNF 的相对误差测试.2 1.2 背景与相关工作.2

2 技术概述 **4**

2.1 高层思路.4 2.2 寻找因式分解析取范式.6 2.3 测试因式分解析取范式.8

3 预备知识 **9**

3.1 相对误差测试.9 3.2 预言机和模拟器.10 3.3 字符串和函数的记号.11 3.4 项、立方体与距离.12

4 析取范式项的聚类 **13**

4.1 K-聚类 s-项析取范式.13 4.2 簇与因子化析取范式相对应.16

5 主算法 及其分析 17

5.1 以相对距离测试因子化析取范式.17 5.2 过程**Find-Factored-DNFs** .17 定理 1 的证明，假设 定理 20 和 定理 23 .20

6 创建预言机： Find-Factored-DNFs 与定理 23 21 的证明

6.1 **Test-Factored-DNF**的关键方面概述：类型 1 和类型 2 查询 . . . . . 21 6.2 当 f 为析取范式时的两个有用子程序及其分析.22 6.3 **Find-Factored-DNFs**过程.24 6.4 在**Find-Factored-DNFs**的肯定情形下进行分析.27 6.5 在**Find-Factored-DNFs** 的否情形下进行分析.36

7 测试因子化析取范式38

7.1 参数.38 7.2 初始化.38 7.3 **DNFLearner** 的性能保证.39 7.4 **ConsCheck** 的性能保证.40 7.5 定理 20 的证明.41

8 DNFLearner算法 41

8.1 符号与设置.41 8.2 阶段 1： **Approximate** .43 8.3 学习器阶段 2： **FindCandidate** .48 8.4 学习器阶段 3：构建预言机.51 8.5 综合起来.54

9 一致性检查 **54**

9.1 ConjTest 的鲁棒性**ConjTest** .56 9.2 肯定情形： 定理 52 的证明 .57 9.3 否情形： 定理53 的证明 . 58

A A $n^{O(\log(s/\varepsilon))}$相对误差测试器的查询 s项析取范式 65 B 无分布测试与相对误差测试不可比较 66 C ConjTest 概述，来自 [CPPS25a] **67**

简要描述的算法列表（每种算法的详细描述请参见其标题）。

1 Sim-SAMP.11 2 聚类算法.14 3 Test-DNF：我们用于对析取范式进行相对误差测试的主要算法 . . . . . 17 4 Test-Equivalence.23 5 In-Pool.23 6 Find-Factored-DNFs：一种算法，用于生成至多包含 s 对预言机模拟器（成员查询$_i^*$、SAMP$_i^*$）的列表，这些模拟器对应于因子化析取范式.26 7 成员查询* .27 8 SAMP* .27 9 Test-Factored-DNF：一种算法，用于相对误差测试一个函数是否为 r 项因子化析取范式.38 10 DNFLearner：给定一个因子化 DNF f : $= H \wedge t$，该算法返回函数 g 的成员查询和 SAMP 预言机，其中该函数与 t 的相对误差很小.42 11 Approximate.45 12 Extract.48 13 FindCandidate.49 14 CheckLit.52 15 Sim-MQ-J.53 16 Sim-SAMP-J.53 17 ConsCheck：给定函数 f 和 g，该算法检查对于某个合取 C , f 是否与 $C \wedge g$ 相对误差很小.55 18 Sim-MQ-Γ .56

1 引言

性质测试始于 [BLR93、RS96、GGR98], 等人的工作，提供了一种通用的算法框架：通过对给定对象进行少量查询，判断该对象是否满足某项性质，或者是否与满足该性质的对象相距甚远。其核心思想和动机是，对于许多组合结构（如图、函数或分布），通常只需检查输入中极小的一部分，就能近似测试该结构的全局性质。研究最深入的性质测试领域之一是布尔函数类；针对范围广泛的函数类，人们已设计出高效的测试器。此外，相关技术还带来了新的见解，并与其他领域建立了联系，包括布尔函数的傅里叶分析和学习理论。

在标准布尔函数类的测试模型中，测试器可以查询输入函数 f，并接收参数 ε，其任务是区分 f 该函数属于此类，还是 f 它与此类中的每个函数的汉明距离至少为 ε 。（更多背景见第 1.2 节。）尽管标准模型下的性质测试取得了极大成功，但该模型有一个显著缺陷：它不适合测试稀疏的布尔函数；这类函数仅在所有 2^n 个点中有极小一部分 $\{0,1\}^n$ 是满足赋值，因此任何这类函数都与常数 0 函数平凡地接近。稀疏函数在均匀分布下易于测试这一局限，可以说绕开了若干有趣的布尔函数类中丰富而复杂的许多方面，其中包括本文研究的基础类—— s项 *DNF* 公式。例如， s项 DNF 公式在高效 PAC 学习中向来难以处理，困难恰恰源于 DNF 公式可能包含“宽”（含有许多变量）的项。然而， s仅含“宽”项的项 DNF 公式是稀疏的，因此在标准模型中很容易测试，因为它们与常数 0 函数接近。

为了有意义地扩展性质测试的范围，使其涵盖有关测试稀疏布尔函数的问题， [CDH+25b] 引入了布尔函数性质测试的一种新变体，称为相对误差测试。相对误差测试模型将待测试的未知 $f\colon \{0,1\}^n \to \{0,1\}$ 与另一个函数之间的距离定义为 $g\colon \{0,1\}^n \to \{0,1\}$ 满足

$$\mathsf{reldist}(f,g) := \frac{|f^{-1}(1) \triangle g^{-1}(1)|}{|f^{-1}(1)|}.$$

与标准模型不同，在相对误差模型中， f 与 g 之间的距离是根据 f的支撑集大小 $f^{-1}(1)$来度量的，而不是仅在所有 2^n输入上进行全局度量。这一区别对于特征选择或故障检测等应用至关重要，因为其中的底层函数通常描述罕见但重要的事件，或者存在于规模庞大但稀疏的数据域中。相对误差性质测试还将性质测试与其他几个以相对意义度量距离的领域联系起来，例如一般分布下的PAC学习、噪声稳定性和近似计数。

请注意，如果只允许黑盒 $\mathrm{MQ}(f)$ 查询（如标准模型中那样），那么对于非常稀疏的 f，可能需要数量庞大的查询才能获得关于 f的任何信息。因此，该模型还允许测试算法从采样预言机 $f^{-1}(1)\mathrm{SAMP}(f)$中获取 第 3.1 节 所述的独立均匀元素。（有关相对误差测试模型的详细描述，请参见第 3.1 节。）

直观上显然且容易证明（见 [CDH+25b]中的事实 3.1）相对误差测试至少与标准模型测试一样困难。此外，一个简单的（人为构造的）函数类示例（见 [CDH+25b]）表明，即使对于在标准模型中使用 $2^{\Omega(n)}$ 次 $\mathrm{MQ}(f)$ 或 $\mathrm{SAMP}(f)$查询即可轻松测试的函数类，相对误差测试也可能很困难，需要调用 $O(1)$。然而，对于所关注的自然函数类，进行相对误差测试究竟有多容易（或多困难），先验上 并不清楚。

1.1 我们的主要结果：DNF 的相对误差测试

本文聚焦于 s 项 DNF 公式这一函数类；该类包含所有函数 $f\colon \{0,1\}^n \to \{0,1\}$，这些函数都可以表示为 $f = T_1 \vee \cdots \vee T_{s'}$，其中某个 $s' \le s$ 满足：每个 T_i 都是布尔文字的合取。DNF 公式是一类丰富且受到深入研究的函数，在逻辑和理论计算机科学领域有着悠久的历史。作为一种能够表示任意布尔函数的通用模型，DNF 公式是最重要的布尔函数类之一，在布尔函数分析、学习理论、计算复杂性和测试领域都占据核心地位。

从相对误差性质测试的角度来看，析取范式也特别有趣。一方面， s项 DNF 公式具有很强的表达能力，与 junta函数（及其子类）不同，它可以依赖于超常数个变量。另一方面，与标准模型相比，它们在相对误差测试模型中似乎颇具挑战性。在标准误差模型中， s项析取范式仅需 $\tilde{O}(s/\varepsilon)$ 次查询（ [Bsh20 、 DLM+07 、 CGM11]）。相比之下，学习理论中先前工作的一个直接推论给出了一个针对 s项析取范式的相对误差测试器，所需查询次数为 $n^{O(\log(s/\varepsilon))}$ 。（详见附录 A 。）

改进析取范式的相对误差测试面临的一项关键挑战是其缺乏平滑性：对析取范式进行微小改动，例如从某个项中删除一个变量，都可能导致满足赋值的数量发生巨大变化[1], ，从而使测试和近似都变得复杂。因此，理解相对误差模型下析取范式测试的复杂度，有助于洞察组合结构如何与乘法精度保证相互作用，以及在绝对距离和相对距离这两种概念出现分歧的情况下，是否仍能构造高效的测试器。

我们的主要结果是一个用于 DNF 公式的相对误差测试算法，其查询复杂度与环境维度 无关：

定理 1 （ s项析取范式的相对误差测试）。 对于 $0 < \varepsilon \le 1/2$， 存在一个针对 s项 DNF 公式类的 ε-相对误差测试算法，该算法在 $\{0,1\}^n$ 上进行 poly$(s, 1/\varepsilon)$ 次查询。

这对上文提到的 $n^{O(\log(s/\varepsilon))}$ 查询相对误差 DNF 测试器是一次重大改进。值得注意的是，我们的新测试器的查询复杂度与标准模型下最先进的 DNF 测试器（[Bsh20，DLM+07，CGM11]）相当，至多相差多项式因子。

1.2 背景与相关工作

标准误差测试。 在过去三十年左右的时间里，研究者投入了大量精力，以了解哪些布尔函数的性质（等价地，哪些函数类）在标准模型中可以高效测试。从[BLR93、GGR98], 的开创性工作开始，人们研究了范围广泛的不同函数类。令人惊讶的是，在相当多的情况下，常数查询可测试性（即查询复杂度与

[1]例如，从析取范式的一个项中删除单个变量，可能会使其满足赋值的数量翻倍。

环境维度 n 无关）确实是可能的。目前已知具有常数查询可测试性的函数类，部分包括文字 [PRS02]; 合取式 [PRS02, GR20]; 奇偶函数 [BLR93]; 决策列表 [DLM+07, Bsh20]; 线性阈值函数 [MORS10]; s-相关函数[FKR+04, Bla08, Bla09]; s-相关函数的子类（其中涵盖许多自然的函数类，包括大小为s 的决策树、大小为s 的分支程序、大小为s的布尔公式，以及傅里叶次数至多为 d [的函数 +07DLM, CGM11, Bsh20]）；以及这样一些函数类，其中每个函数都能在均匀分布下由某个相关函数良好近似（其中还涵盖许多其他函数类，包括s-项单调析取范式 [PRS02]、 s-项析取范式、定义在 $\mathbb{F}_2$上的 s-稀疏多项式，以及具有无界扇入的大小为s 的布尔电路 [DLM+07, CGM11, Bsh20]）。

相对误差测试。 回顾一下，在相对误差测试模型中，从 f 到 g 的距离是在 $|f^{-1}(1)|$ 这一“相对”尺度下测量的，而不是在 $2^n = |\{0,1\}^n|$这一“绝对”尺度下测量的。该模型的思想与图性质测试中研究充分的模型类似（参见 [GR02、PR02]以及 [Gol10]的第 9 章和第 10 章），这些模型适用于测试具有 N个顶点且拥有$o(N^2)$ 条边的稀疏图。

近期有几项工作为多种经过深入研究的类给出了高效的相对误差测试器。[CPPS25b] 表明， s-junta函数以及在变量重命名下封闭的 s-junta函数的子类（并满足一个额外的温和技术条件，详见 [CPPS25b] ）可以在相对误差模型中进行高效测试，仅使用 poly$(s, 1/\varepsilon)$ 次预言机调用。基于截然不同的技术，另一项近期工作 [CPPS25a]表明，任意长度的布尔合取式类和˜1-决策列表类同样可以进行高效的相对误差测试，仅使用 $O(1/\varepsilon)$ 次查询。 [CPPS25a] 的结果与 [CPPS25b]形成了有趣的对比，因为合取式和决策列表都不是相关函数的子类；不过，合取式和决策列表都是相当简单且表达能力有限的布尔函数类型。从负面结果来看，一个同时推广了合取式和 1-决策列表的稍丰富类，即线性阈值函数类，最近被证明无法通过常数次查询进行测试： [CDH+25a]证明，针对线性阈值函数的任何相对误差测试˜算法都必须具有 $\Omega(\log n)$ 的查询复杂度。

与无分布测试的关系。

我们指出，相对误差测试并非第一个采用不同于统计函数 $\{0,1\}^n$ 上两者不一致的点所占比例的距离概念，来衡量函数 f 和 g 之间距离的布尔函数性质测试模型。在无分布测试模型中，两个函数 f 和 g 之间的距离根据 $\mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$, 来衡量，其中 $\mathcal{D}$ 是定义域 $\{0,1\}^n$上的未知任意分布。除了进行黑盒预言机查询外，无分布测试算法还可以从未知分布 $\mathcal{D}$中抽取独立同分布样本。然而，事实证明，在标准模型中可通过常数次查询进行测试的许多函数类，例如决策列表、线性阈值函数和合取式（对于 $s = 1$而言，它们是 s-项析取范式），在无分布设置中却需要超常数次查询——事实上，需要多项式数量的查询 n ；参见例如 [GS09、CX16、CP22、CFP24]。

我们进一步指出，相对误差测试模型与无分布测试不可比较；其中一个原因是，在这两种设置中，函数之间的距离度量方式不同。在分布无关模型中，两个函数 f, g 之间的距离是相对于分布 $\mathcal{D}$ 来度量的，而测试算法可以从该分布中进行采样；这些样本通常会同时包含所测试函数 f 的正例和负例。相比之下，在相对误差设置中，测试器可以从中采样的分布是 $f^{-1}(1)$ 上的均匀分布，也就是说，该分布完全由 f的正例构成。

此外，相对距离 $\mathrm{reldist}(f,g)$不等于 $\mathbf{Pr}_{\boldsymbol{x}\sim f^{-1}(1)}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$：函数 g与 f之间可能具有较大的相对距离，因为 g在 $f^{-1}(1)$之外的许多点上不同于 f；而在 $f^{-1}(1)$上的均匀分布下，这些点的权重为零。

为具体说明两个函数类，从而展示这两种模型不可比较，我们指出，已知在分布无关设置下，合取式类需要 $\mathrm{poly}(n)$ 次查询 [GS09, CX16], ，而在相对误差模型中，它可以通过 $O(1/\varepsilon)$ 次查询进行测试 [CPPS25a]。反过来，在 B 节 中，我们给出一个简单的函数类示例：它显然可以进行分布无关测试，但任何相对误差测试器都需要 $2^{\Omega(n)}$ 次查询。

2 技术概述

本节介绍我们用于析取范式相对误差测试的算法 **Test-DNF** 背后的主要思路。在 第 2.1 节中，我们首先从概念层面概述该方法的总体思路，并介绍其两个算法组件： **Find-Factored-DNFs** 和 **Test-Factored-DNF**。随后，我们将在 第 2.2 节 和 2.3 节中，讨论要使该方法奏效所需克服的一些最重要的难题。有关测试算法详细结构的示意图，请参见 图 2 。

2.1 高层思路

因子化析取范式。 为了说明我们的方法，一个关键的预备概念是因子化析取范式。粗略地说，因子化析取范式是如下形式的函数 $H \wedge f_1$，其中 H是一个长度任意的合取项，而 f_1 是一个仅依赖于$(\leq s)$个变量的项析取范式$\mathrm{poly}(s)$ 个变量；更精确的定义见定义 18 。请注意，我们可以将合取项 H （其长度同样可以任意）视为从初始析取范式的每一项中“提取出来”的公共部分。我们有时将 H称为“头部 ”，将 f_1称为因子化析取范式的“尾部”。

有了因子化析取范式这一概念，我们就可以介绍我们新的分解方法了。

析取范式的一种新分解。 我们的方法关键在于采用了一项新的技术性和概念性要素，即针对任意 s 项析取范式的一项新颖结构分解结果。粗略地说，这一结构分解结果表明：

> 给定任意 s 项析取范式 f，存在一个由 f 删除若干项（因此 $f'^{-1}(1) \subseteq f^{-1}(1)$）得到的子析取范式 f'，它具有以下性质：**(i)** f 几乎所有满足赋值也都是 f'的满足赋值；**(ii)** f' 可以划分为不超过 s 个子析取范式 h_1、.....、$h_{\leq s}$ 的析取（因此，每个 h_i 中的每一项都是 f的原始 s 项），其中每个子析取范式都是前文所述的因子化析取范式；**(iii)** 对 f' 进行均匀随机满足赋值（满足某个子析取范式 h_i）时，它极有可能在大量坐标上与其他每个子析取范式 h_j、$j \neq i$中的每一项都不一致。

图1：结构分解

以上只是一个粗略的表述；更多细节请参见第4节，特别是推论17。既然存在这样的分解，我们现在概述算法的第一阶段。

给定一个输入函数 f：$\{0,1\}^n \to \{0,1\}$（可访问预言机 MQ(f) 和 SAMP(f)），Test-DNF 的第一阶段运行一个名为 Find-Factored-DNFs 的过程：当 f 是一个 s-项析取范式时，该过程针对上述分解中的每个因子化析取范式 h_i，生成一对预言机 SAMP(h_i) 和 MQ(h_i)。另一方面，当 f 在相对误差距离意义下远离 s-项析取范式时，Find-Factored-DNFs 要么拒绝，要么返回一系列函数 h_1、……、$h_{\le s}$ 所对应的预言机对；这些函数同样满足 $\bigcup h_i^{-1}(1) \subseteq f^{-1}(1)$，并且 $\bigcup h_i^{-1}(1)$ 包含 f 的几乎所有满足赋值（不过，与肯定情形不同，这里的 h_i 不一定是因子化析取范式）。Find-Factored-DNFs 的查询复杂度为 poly(s/ε)。

我们强调，以上描述的 Find-Factored-DNFs 的性能保证已大幅简化，旨在帮助读者直观理解 Test-DNF 的工作原理。特别是，能够为分解中的因子化析取范式 h_i 返回完美预言机，听起来似乎好得令人难以置信。事实上，正如第2.2节所述，在第二阶段使用这些预言机时需要克服的障碍之一是，Find-Factored-DNFs 返回的是 MQ(h_i) 和 SAMP(h_i) 的“近似模拟器”：它们是由 Find-Factored-DNFs 构建的随机算法，通过调用 MQ(f) 和 SAMP(f) 多项式次（poly(s/ε）来模拟一次对 MQ(h_i) 或 SAMP(h_i) 的调用；为清楚起见，后文将分别称它们为 MQ*(h_i) 和 SAMP*(h_i)，以强调它们是模拟器，并且可能出错。不过，在本小节的概述中，读者可以假设它们是完美模拟器，因此我们仍将它们记作 MQ(h_i) 和 SAMP(h_i)。

测试因子化析取范式。已知 h_1、……、$h_{\le s}$ 构成 s 项析取范式 f 的一个子析取范式 f' 的划分，则存在正整数 r_i，使得 $\sum_i r_i \le s$，且每个 h_i 都是尾部含有 r_i 项的因子化析取范式。另一方面，当 f 与 s 项析取范式的相对距离很远时，利用 Find-Factored-DNFs 返回的 h_i 满足 $\bigcup h_i^{-1}(1) \subseteq f^{-1}(1)$，且 $\bigcup h_i^{-1}(1)$ 包含 f 的几乎所有满足赋值，不难证明，不存在整数 r_i' 使得 $\sum_i r_i' \le s$，并且每个 h_i 在相对距离上都非常接近一个尾部含有 r_i' 项的因子化析取范式。该算法的第二阶段旨在区分这两种情况，因此自然的做法是在每个 h_i 上运行一个因子化析取范式测试器。

这由我们的第二个算法组件 **Test-Factored-DNF** 实现。给定一个函数h 和一个参数 $r \le s$，它使用 $\mathrm{poly}(s/\varepsilon)$ 次查询来调用 $\mathrm{MQ}(h)$ 和 $\mathrm{SAMP}(h)$，以区分以下两种情况：h 是尾部至多包含 r 项的因子化析取范式，以及 h 与任何尾部至多包含 r 项的因子化析取范式之间的相对距离都很大。我们将在稍后的 第2.3节中讨论设计这种高效测试器所面临的挑战。值得注意的是，鉴于 **Find-Factored-DNFs** 只生成“近似模拟器”，**Test-Factored-DNF**需要具备足够的鲁棒性，以应对它们的不准确之处。我们将在稍后的 第2.3节中回到这一点。

主要算法概述。根据以上讨论，我们现在可以对用于 s项析取范式的相对误差测试算法 **Test-DNF** 作一个高层次的概述：

> **阶段 1：** 给定输入函数 f，运行 **Find-Factored-DNFs**；它要么拒绝，要么为一系列函数 h_1、……、$h_{\le s}$，生成预言机对 $\mathrm{SAMP}(h_i)$ 和 $\mathrm{MQ}(h_i)$。
>
> **第二阶段：** 对于每个 h_i，运行 **Test-Factored-DNF**，在 h_i 上使用参数 r，其值依次设为 1、2、……、s（使用由 **Find-Factored-DNFs**生成的预言机）。令 r_i' 表示使 **Test-Factored-DNF** 接受 h_i 的最小整数，其中 r_i'。若 $\sum_i r_i' \le s$，则接受；否则拒绝。

Test-DNF 对 $\mathrm{MQ}(f)$ 和 $\mathrm{SAMP}(f)$ 的总体查询复杂度显然为 $\mathrm{poly}(s/\varepsilon)$。

当 f 是一个 s项析取范式时，**Find-Factored-DNFs** 会返回 h_i在 f的分解中给出的预言机。设 r_i 为每个 h_i尾部中的项数，且满足 $\sum_i r_i \leq s$。我们知道，在第二阶段中，对于每个 $r'_i \leq r_i$，i （注意， h_i 可能接近一个尾部项数少于 r_i 项的因子化析取范式），因此，**Test-DNF** 接受。

当 f 与每一个 s项析取范式在相对距离上都相距甚远时，**Find-Factored-DNFs** 要么拒绝，要么返回函数 h_i 如前所述，不存在整数r'_i使得$\sum_i r'_i \leq s$ 且每个 h_i 在相对距离上都接近于一个尾部含有r'_i 项的因子化析取范式。由此可以得出结论： f 在**Test-DNF**的第二阶段被拒绝。

2.2 寻找因子化析取范式

现在我们进一步说明**Find-Factored-DNFs** 如何为因子化析取范式生成预言机模拟器 h_1、……、 $h_{\leq s}$，它们构成了前文所述 f的一个分解，其中 f是一个 s项析取范式。

首先从第4节开始，我们将在该节介绍一个简单的贪心聚类过程：给定 f的各项 T_1、……、 T_s，将它们划分为簇 D_1、……、 $D_{\leq s}$，使得如果令 g_i为由 D_i中的项定义的子析取范式，就能将 f分解为 g_1、……、 $g_{\leq s}$，且这些函数满足前文图1中给出的全部三个性质。（这里我们使用了 g_i而不是 h_i，因为它们并非 Find-Factored-DNFs 返回的确切函数。）具体做法是确保大致而言：(1) 同一簇中的任意两项都“接近”；(2) 不同簇中的任意两项都“相距甚远”。读者可以将两项之间的距离视为由它们的对称差度量（不过在第4节中需要采用更谨慎的处理方式）。自然地，(1) 导致图1所示分解中的性质 (ii)，而 (2) 导致性质 (iii)。后者是因为同一簇 D_ℓ 中的任意两项都彼此接近，因此可以将它们提取公因子，形成一个因子化析取范式（见推论17）。我们强调，我们的算法并未也无法执行这一过程，因为它并“不知道”这些项。应将这一聚类 D_1、……、 $D_{\leq s}$ 视为 Find-Factored-DNFs 的理想目标，这将在其设计与分析中发挥至关重要的概念作用。为方便起见，在本节余下部分中，我们将D_1、……、 $D_{\leq s}$ 称为规范簇。

从较高的层面来看，**Find-Factored-DNFs** 首先从 $\mathrm{poly}(s/\varepsilon)$ 中抽取 $f^{-1}(1)$ 个样本。为便于讨论本节内容，我们假设每个被抽取的点 $\boldsymbol{x}$ 都满足析取范式 f中的唯一一项；我们用 $T_{\boldsymbol{x}}$ 表示该项，并用 $T_{\boldsymbol{x}}$ 表示 $D_{\boldsymbol{x}}$所在的规范簇。利用 g_1、……、 $g_{\leq s}$ (或者等价地，利用规范聚类 D_1、……、 $D_{\leq s}$) 上的性质 (iii)，可以证明，对于任意两个被抽取的点 $\boldsymbol{x}$、$\boldsymbol{y}$，以下结论以高概率成立：

(a) 如果 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 来自同一项 $T_{\boldsymbol{x}} = T_{\boldsymbol{y}}$，那么由 $\boldsymbol{x}$和 $\boldsymbol{y}$[2] 定义的立方体中的每个点 z 都满足 T，因此也是 f 的满足赋值（这显然总是成立）；

(b) 如果 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 来自两个不同的规范簇（即 $D_{\boldsymbol{x}} \neq D_{\boldsymbol{y}}$），那么由 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 定义的立方体中的大多数点都不是 f的满足赋值。直观地说，之所以如此，是因为根据分解的性质 (iii)， $\boldsymbol{x}$ 很可能与满足 $D_{\boldsymbol{x}}$ 之外的每一项相距甚远，而 $\boldsymbol{y}$ 也很可能与满足 $D_{\boldsymbol{y}}$之外的每一项相距甚远；由此可以推知，它们所定义的立方体中的大多数点都不满足 f中的任何一项。

只需从立方体中查询少量随机点，就能区分这两种情况。我们对每对样本运行这个子程序，**Test-Equivalence**， $\boldsymbol{x}$， $\boldsymbol{y}$ 并添加一条边

[2]由 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 定义的立方体，是由满足以下条件的点构成的集合： z，且对每个 $i \in [n]$，都有 $z_i \in \{\boldsymbol{x}_i, \boldsymbol{y}_i\}$。

$(\boldsymbol{x},\ \boldsymbol{y})$ 当**Test-Equivalence** 确信由 $\boldsymbol{x},\ \boldsymbol{y}$ 定义的立方体中充满了 f的满足赋值时，就在它们之间添加边。以高概率，对于每对 $\boldsymbol{x},\ \boldsymbol{y}$，都有：(a) 若 $T_{\boldsymbol{x}} = T_{\boldsymbol{y}}$，则$(\boldsymbol{x},\ \boldsymbol{y})$ 是一条边；(b) 若 $D_{\boldsymbol{x}} \neq D_{\boldsymbol{y}}$，则$(\boldsymbol{x},\ \boldsymbol{y})$ 不是一条边。该图将我们的样本划分为多个"池" $\mathcal{P}_1$，.....，$\mathcal{P}_{\leq s}$，其中每个池 $\mathcal{P}_i$ 都是图的一个连通分量。

我们知道，由于 (a) 以及 f 的项数不超过 s，池的数量至多为 s。令 C_i 为 f 的项的集合，其中每一项都至少由 $\mathcal{P}_i$中的一个点满足。由 (b) 可知，$C_1,\ \ldots,\ C_{\leq s}$ 是 $D_1,\ \ldots,\ D_{\leq s}$ 的规范聚类的一种细化，因此每个 C_i 所定义的子析取范式也必定是因子化析取范式，我们将其记为 h_i。我们希望 $h_1,\ \ldots,\ h_{\leq s}$ 能为我们提供所需的 f 分解。然而，不同于作为规范聚类的子析取范式的 g_i，现在尚不清楚图1中的性质 (iii) 是否仍然成立：C_i 和 C_j可能来自同一个 D_ℓ，此时它们的项可能彼此接近，因此，从 $f^{-1}(1)$ 中抽取的样本若满足 C_i 中的某个项，也很可能满足 C_j 中的某个项。

为克服这一障碍，我们执行一个关键过程：利用来自 $f^{-1}(1)$ 的一批新样本合并池，并在这些样本与每个池中的点之间运行 Test-Equivalence。我们观察到，如果从 $f^{-1}(1)$ 抽取的样本满足 C_i 中的某个项的可能性很高，那么它也很可能满足C_j 中的某个项；因此，我们很可能会在新抽取的一批样本中得到这样一个点 $\boldsymbol{z}$，而等价性测试子程序会确信，在 $\boldsymbol{z}$ 与 C_i 中的某个点之间定义的立方体，以及在 $\boldsymbol{z}$ 与 C_j 中的某个点之间定义的立方体，都充满了满足赋值。在这种情况下，我们将 $\mathcal{P}_i$ 和 $\mathcal{P}_j$ 合并为一个池 $\mathcal{P}_i \cup \mathcal{P}_j$，这样一来，$C_i$ 和 C_j 也随之合并。另一方面，如果 C_i 和 C_j 来自不同的规范簇，那么由于规范聚类满足条件 (iii)，它们不太可能被 Test-Equivalence 合并。

最终，我们得到池 $\mathcal{P}_1,\ \ldots,\ \mathcal{P}_{\leq s}$，使得它们定义的 h_i 构成了将 f 分解为因子化析取范式所需的分解。Find-Factored-DNFs 的其余工作是使用池 $\mathcal{P}_i$ 为每个 h_i 的预言机构建模拟器，下面将它们分别记为 MQ*(h_i) 和 SAMP*(h_i)。我们重点讨论以下情形：f 是一个 s-项析取范式，并且每个 h_i 都是由池 $\mathcal{P}_i$ 定义的因子化析取范式。理想情况下，我们希望 MQ*(h_i) 和 SAMP*(h_i) 的行为与 MQ(h_i) 和 SAMP(h_i) 类似，从而 Test-Factored-DNF 在 MQ*(h_i) 和 SAMP*(h_i) 上接受 h_i 的概率，至少等于它在 MQ(h_i) 和 SAMP(h_i) 上接受 h_i 的概率（该概率至少为 $2/3$）减去 0.01。为此，基于耦合的标准分析表明，我们需要具备以下性能保证：

1. 每次调用 SAMP*(h_i)，都会以至少 $1-\delta$ 的概率返回一个从 $h_i^{-1}(1)$ 中抽取的样本；
2. 每次针对点 $x \in \{0,1\}^n$ 调用 MQ*(h_i)，都会以至少 $1-\delta.$ 的概率返回 $h_i(x)$

We w如果 $1/\delta$ 大于 **Test-Factored-DN** 的查询复杂度，则应这样做 **F**.

好消息是，SAMP*(h_i) 恰好实现了上述功能。它首先从 $\boldsymbol{x} \sim f^{-1}(1)$ 中抽取一个样本，并对 $\boldsymbol{x}$ 与 $\mathcal{P}_i$ 中的每个点运行 Test-Equivalence；如果 Test-Equivalence 确信由 $\boldsymbol{x}$ 和 $\mathcal{P}_i$ 中某个点定义的立方体中，f 的满足赋值已填满该立方体，则返回 $\boldsymbol{x}$；否则，它会从 $f^{-1}(1)$ 中抽取一个新样本并重复上述过程。只要 $\boldsymbol{x} \in h_i^{-1}(1)$，Test-Equivalence 总能判断出这一点；另一方面，根据关于 h_i 的分解的性质 (iii)，抽到一个能够欺骗 Test-Equivalence 的 $\boldsymbol{x} \notin h^{-1}(1)$ 的可能性极低。[3]

坏消息是，要实现 MQ*(h_i) 并达到上述（最坏情况类型的）性能保证，似乎不可行。考虑一个 $x \in f^{-1}(1)$，使得 $h_i(x) = 0$，但 $= 0$ 与

[3]为了限制 SAMP*(h_i) 所需的样本数量，我们需要额外执行一个池修剪预处理步骤，丢弃"轻"的池，只保留"重"的池，也就是说，$\boldsymbol{z} \sim f^{-1}(1)$ 满足 $h_i(\boldsymbol{z}) = 1$ 的概率至少为 $1/\text{poly}(s/\varepsilon)$。我们略过细节，但要指出，修剪轻池是可行的，因为分解中的性质 (i) 只要求 f' 覆盖 $f^{-1}(1)$ 的大部分。

x 之间的立方体包含的点大多属于 $f^{-1}(1)$。算法不太可能区分这种情况与 $x \in h_i^{-1}(1)$ 的情况。另一方面，正如图1中分解的性质 (iii) 所示，对于一个“近似随机”的点 $\boldsymbol{x}$，正确回答 $h_i(\boldsymbol{x})$ 很容易。举例来说，考虑 $\boldsymbol{x} \sim f^{-1}(1)$。根据性质 (iii)，很可能出现以下两种情况之一：$h_i(\boldsymbol{x}) = 1$（且 $\boldsymbol{x}$ 满足 C_i 中至少一个项），或者 $h_i(\boldsymbol{x}) = 0$，且 $\boldsymbol{x}$ 远未满足 C_i 中的任何项。使用 Test-Equivalence 可以高概率地区分这两种情况。剩下的问题是，这是否足以达到我们的目的？具体而言，在肯定情形下，如果在 $\mathrm{MQ}^*(h_i)$ 上运行，而不是在 $\mathrm{MQ}(h_i)$ 上运行，Test-Factored-DNF 是否仍会以高概率接受 h_i？

为此，我们现在来看看 **Test-Factored-DNF** 是如何工作的，先从它可以访问 完美 预言机 成员查询(h_i) 和 $\mathrm{SAMP}(h_i)$这一较简单的情形开始。

2.3 测试因子化析取范式

如何设计一个高效的因子化析取范式相对误差测试器？近期˜的工作 [CPPS25a], 对此提供了一条线索：该工作给出了一个针对 $O(1/\varepsilon)$ 查询的决策列表相对误差测试器决策列表。决策列表与因子化析取范式之间的联系如下：稍加思考便可看出，任何非平凡的决策列表都可以视为如下形式：$H \wedge L$，其中 H 是一个（长度任意的）合取，而 L 是另一个“稠密”决策列表：L 在 1 上的输出占所有输入赋值的常数比例。从高层次来看，[CPPS25a] 算法的基本思路是，对该列表运行一个标准模型的决策列表测试算法 L (请注意，由于头部的存在，这么做并非易事 H；不过，本概述中我们省略对这些问题的讨论。

在当前背景下，我们考虑的不是以下形式的决策列表：$H \wedge L$ 其中 L 是一个“稠密”决策列表，而是以下形式的因子化析取范式：$H \wedge f_1$，其中 f_1 是一个依赖于 s个变量的 $\mathrm{poly}(s)$项析取范式。对于 s 项析取范式 [Bsh20], ，已有标准模型测试算法；因此很自然会想到，将 [CPPS25a], 中使用的标准模型决策列表测试算法替换为这类算法，或许就能得到一种针对因子化析取范式的测试算法。然而，标准模型的 s项析取范式测试器并不足以满足我们的需求，因为一个含 s项、涉及 $\mathrm{poly}(s)$个变量的析取范式 f_1 未必是“稠密的”；它可能只有 $2^{-\mathrm{poly}(s)}$ 比例的满足赋值（例如，每个项的长度都是 s时，就会出现这种情况）。

相反，我们测试因子化析取范式的方法借鉴了另一篇论文[CPPS25b]中的思想和工具。[CPPS25b] 的主要结果是，为任何在输入变量置换下封闭的 $\mathrm{poly}(k/\varepsilon)$算法，提供对其任意子类进行相对误差测试的算法；注意，只依赖于 k个输入变量的 s项析取范式类，显然满足此条件。在较高层次上，我们测试因子化析取范式的方法，是用一个针对含 相对误差测试器来处理 s项、涉及 $\mathrm{poly}(s)$个变量的析取范式，以替代（标准模型）决策列表测试器，从而在整体 隐式学习对 f_1 的近似（相对误差较小），并将其纳入 [CPPS25a] 测试器的框架中。

我们稍后会在第 7 节更详细地说明这一切是如何完成的。不过，在继续之前，我们先进一步解释一下“隐式学习”或“学习到相关变量的名称为止”这一概念。所谓对尾部的近似进行“隐式学习” f_1（即一个 s项、$\mathrm{poly}(s)$变量的析取范式），是指如果 f_1 依赖于来自 k 的 $x_1, \ldots, x_n$这些变量，那么隐式学习算法会识别出一个函数 g：$\{0,1\}^k \to \{0,1\}$，它依赖于我们称为 $y_1, \ldots, y_k$的变量；经过某种（未知的）映射，将这些 y_i 变量映射到 $\{x_1, \ldots, x_n\}$后，函数 g 就会接近 f。[4] 在

[4]我们提到，“隐式学习”方法已用于不少先前的研究中，例如：[CPPS25b、Bsh20、DLM+07]。

特别是，这种“隐式学习”基本上能让我们确定尾部析取范式中有多少项 f_1 （如前所述，获得这个数值对我们至关重要）。将这个尾部的近似称为 f_1'后，我们的因子化析取范式测试算法会检查 f 是否确实与 $H' \wedge f_1'$ 相对误差接近，其中 H'是某个合取。此项检查由**ConsCheck** 子程序执行；值得指出的是，这项检查比 [CPPS25a],框架中的类似检查更复杂，因为函数 f_1' 可能是稀疏的（而 L 在 [CPPS25a] 框架中的对应函数是稠密的）。完整细节见第9节。

最后，我们转向实际场景：Test-Factored-DNF 运行在 h_i ，且只能访问近似模拟器预言机 MQ*(h_i) 和 SAMP*(h_i)。我们关注 h_i 为因子化析取范式的情况，此时需要 Test-Factored-DNF 仍以接近 $2/3$ 的概率接受（就像在 MQ(h_i) 和 SAMP(h_i) 上运行时一样）。正如第2.2节中所述，主要问题在于 MQ*(h_i)：MQ*(h_i) 只能在“近似随机”的查询点上良好工作的这一事实，是否会显著降低 Test-Factored-DNF 接受 h_i 的概率?

令人惊讶的是，尽管**Test-Factored-DNF** 如上所述是一个包含许多算法组件的高度复杂的测试器（例如参见 图 2），但它发出的所有成员查询都属于以下两种类型：

1. “近似随机”的查询。粗略地说，这类查询针对的是一个点 z ，该点位于点 $\boldsymbol{x}$ （从 h_i^{-1}（1） 中抽取）与点 $\boldsymbol{y}$ （从 $\{0,1\}^n$ 中均匀随机抽取）之间的立方体内；但在一小组随机选取的坐标 i上， z_i 可以任意设定而无需随机性。关于这些“类型 *1*”查询的详细定义，我们将推迟到 第 6.1 节。将这一表述从 **Test-Factored-DNF** 的内部运作中抽象出来，同时确保 MQ*（h_i） 能够正确处理此类查询，是我们需要克服的主要挑战。

2. 查询不一定是随机的，但返回的值必须为1，否则**Test-Factored-DNF** 会立即拒绝。事实证明，这类（所谓的“类型-2”）查询很容易处理，因为 MQ*(h_i) 的实现是单侧的：对于查询，它总是返回 1，当 z 如果 $h_i(z) = 1$ （大致来说，因为当立方体仅包含满足赋值时，**Test-Equivalence** 绝不会出错）。因此，对于类型-2查询，运行**Test-Factored-DNF** 于 MQ*(h_i) 只会提高其接受概率。

总之，虽然 MQ*(h_i) 并非通用的成员关系预言机，适用于 h_i，但它专为与我们的因子化析取范式相对误差测试器协同工作而设计。因此，使用模拟器 **Test-Factored-DNF**接受 h_i 的概率，借助模拟器 MQ*(h_i)和 SAMP(h_i)* 运行时，至少等于使用完美预言机 MQ(h_i) 和 SAMP(h_i) 运行时的概率减去 0。01。

3 预备知识

3.1 相对误差测试

回顾一下，在针对一类 $\mathcal{C}$ 个 n 变量布尔函数的标准测试模型中，测试算法可以通过成员查询 MQ(f) 访问一个未知的任意函数 $f\colon \{0,1\}^n \to \{0,1\}$ 的预言机。如果 $f \in \mathcal{C}$，算法必须以高概率（例如至少 2/3；这可以通过标准技术放大）输出“是”；而且必须以高概率（同样例如至少 2/3）

如果 f 与每个函数 $g \in \mathcal{C}$ 之间的距离至少为 ε，其中 f 和 g 之间的距离定义为

$$\frac{|f^{-1}(1) \triangle g^{-1}(1)|}{2^n}.$$

相比之下，针对 $\mathcal{C}$ 的相对误差测试算法可以访问 MQ(f) 预言机，也可以访问 SAMP(f) 预言机；调用该预言机时，它会返回一个均匀随机元素 $\boldsymbol{x} \sim f^{-1}(1)$。针对 $\mathcal{C}$ 的相对误差测试算法必须在 $f \in \mathcal{C}$ 时以高概率（例如至少 2/3；同样，这可以轻松地放大）输出“是”；而在 $\mathsf{reldist}(f, \mathcal{C}) \geq \varepsilon$ 时，必须以高概率（同样，例如至少 2/3）输出“否”，其中

$$\mathsf{reldist}(f, \mathcal{C}) := \min_{g \in \mathcal{C}} \mathsf{reldist}(f, g) \quad \text{and} \quad \mathsf{reldist}(f, g) := \frac{|f^{-1}(1) \triangle g^{-1}(1)|}{|f^{-1}(1)|}.$$

如上所述，相对误差测试算法可以调用 MQ(f) 预言机和 SAMP(f) 预言机。我们用“查询复杂度”来指调用任一预言机的总次数。

注 2。 由相对误差测试的定义可直接得出：如果某算法是针对某个类 $\mathcal{C}$ 的ε-相对误差测试算法，那么对于任意参数 $\varepsilon' \in [\varepsilon, 1/2].$，它也是针对 $\mathcal{C}$ 的 ε'-相对误差测试算法。这个观察对我们很有用，因为它使我们可以不失一般性地假设，我们是在对 s-项析取范式类进行 ε 测试，且 ε 的取值不超过某个足够小的绝对常数（这也意味着 $\log(s/\varepsilon)$ 等量至少是某个足够大的绝对常数）。

我们回顾相对距离的以下“近似对称性”和“近似三角不等式”性质（参见 [CPPS25b] 中的引理 9 和 10，其证明直接明了）：

引理 3 （相对距离的近似对称性）。 设 $f, g : \{0,1\}^n \to \{0,1\}$ 是函数，满足 $\mathsf{reldist}(f, g) \leq \varepsilon$，其中 $\varepsilon \leq 1/2$。则 $\mathsf{reldist}(g, f) \leq 2\varepsilon$。

引理 4 （相对距离的近似三角不等式）。 设 $f, g, h : \{0,1\}^n \to \{0,1\}$ 满足 $\mathsf{reldist}(f, g) \leq \varepsilon$，且 $\mathsf{reldist}(g, h) \leq \varepsilon'$。则 $\mathsf{reldist}(f, h) \leq \varepsilon + (1+\varepsilon)\varepsilon'$。

3.2 预言机与模拟器

在整个 **Test-DNF**算法中，我们将实例化不同的算法，用于模拟 成员查询(g) 或 SAMP(g)，它们针对各种函数 g。具体而言，用于模拟 成员查询 预言机行为的算法实例记为 成员查询*；类似地，用于模拟 SAMP 预言机的算法实例记为 SAMP*。

我们稍后会用到的这类模拟器的一个简单示例如下：

然后，算法可以执行如下形式的语句：“$\mathrm{SAMP}^* \leftarrow$ **Sim-SAMP**($\mathrm{SAMP}(h)$，Y)”（其中 $\mathrm{SAMP}(h)$，Y 是适当选取的）。然后，算法可以调用 SAMP^*，以获取$\boldsymbol{x} \in \{0,1\}^{\mathsf{Y}}$中的某个字符串；所期望的是，$\mathrm{SAMP}^*$ 的行为与某个函数的$\mathrm{SAMP}(g)$ 预言机完全相同$g : \{0,1\}^{\mathsf{Y}} \to \{0,1\}$。

算法1Sim-SAMP

输入： SAMP(h)，某个函数 h 的：$\{0,1\}^n \to \{0,1\}$ 以及 $\emptyset \neq \mathsf{Y} \subseteq [n]$**输**

出： $\{0,1\}^{\mathsf{Y}}$ 中的一个点。

1: 抽取 $\boldsymbol{z} \sim$SAMP(h)。

2: 返回 $\boldsymbol{z}_{\mathsf{Y}}$。

算法1： **Sim-SAMP**。一个用于模拟 SAMP 预言机的算法示例。稍后将在 第9节中使用该算法，来模拟函数 Γ 在 **ConsCheck**中定义的SAMP 预言机。

定义5。 令 $g\colon \{0,1\}^{\mathsf{Y}} \to \{0,1\}$ 为任意函数。

1. 如果对于任意 x，算法 ALG 以至少 $1-\delta$ 的概率输出 $g(x)$，则称算法 ALG 是 MQ(g) 的 δ-精度模拟器。2. 类似地，如果每次被调用时，算法 ALG 都以至少 $1-\delta$ 的概率输出 $z \sim g^{-1}(1)$，则称算法 ALG 是 SAMP(g) 的 δ-精度模拟器。

如果某个算法是 MQ(g)（SAMP(g）的 0-精度模拟器，我们称其完美模拟MQ(g)（SAMP(g)）。

3.3 字符串和函数的记号

我们用 $[n]$ 表示集合 $\{1,\ldots,n\}$.。对于子集 $\mathsf{S} \subseteq [n]$，我们用 S 表示 $\overline{[n]} \setminus \mathsf{S}$。我们用 $\circ$ 表示拼接。

给定 $\mathsf{S} \subseteq [n]$ 和 $z \in \{0,1\}^n$，我们用 z_{S} 表示 $\{0,1\}^{\mathsf{S}}$ 中的字符串，该字符串在 S 中的每个坐标上都与 z 一致。我们还用 $\bar{z}$ 表示将 z 的每一位取反后得到的字符串，因此对于每个 $i \in [n]$，都有 $\bar{z}_i = 1 - z_i$。

我们使用函数限制的标准记号。对于 $f\colon \{0,1\}^n \to \{0,1\}$ 和 $u \in \{0,1\}^{\mathsf{S}}$，其中某个 S 满足 $\subseteq [n]$，函数 $f\restriction_u\colon \{0,1\}^{[n]\setminus\mathsf{S}} \to \{0,1\}$ 定义为函数

$$f\restriction_u(x) = f(z), \quad \text{where} \quad z_i = \begin{cases} u_i & \text{if } i \in \mathsf{S}, \\ x_i & \text{if } i \in [n] \setminus \mathsf{S}. \end{cases}$$

对于有限集合 A，我们用“$\boldsymbol{x} \sim A$”表示 $\boldsymbol{x}$ 在 A 上均匀随机。

给定双射 $\pi\colon \mathsf{X} \to \mathsf{Y}$，其中 $\mathsf{X},\mathsf{Y} \subseteq [n]$ 且 $x \in \{0,1\}^{\mathsf{X}}$，我们用 $\pi(x)$ 表示 $\{0,1\}^{\mathsf{Y}}$ 中满足对每个 $j \in Y$ 都有 $(\pi(x))_j = x_{\pi^{-1}(j)}$ 的字符串。给定 $f\colon \{0,1\}^{\mathsf{Y}} \to \{0,1\}$，我们用 $f_\pi\colon \{0,1\}^{\mathsf{X}} \to \{0,1\}$ 表示函数 $f_\pi(x) = f(\pi(x))$。

给定 $n \geq r \geq 1$、单射映射 $\sigma\colon [r] \to [n]$ 和赋值 $y \in \{0,1\}^n$，我们用 $\sigma^{-1}(y)$ 表示满足对每个 $i \in [r]$ 都有 $x_i = y_{\sigma(i)}$ 的字符串 $x \in \{0,1\}^r$。给定一个 $z \in \{0,1\}^r$，我们用 $\sigma(z)$ 表示字符串 $u \in \{0,1\}^{\mathsf{S}}$，其中 $\mathsf{S} = \{\sigma(i)\colon i \in [r]\}$ 是 σ 的像集，且对每个 $i \in [r]$ 都有 $u_{\sigma(i)} = z_i$。给定一个关于 h 个变量的函数 $g\colon \{0,1\}^r \to \{0,1\}$，以及单射映射 $\sigma\colon [r] \to [n]$，我们用 g_σ 表示关于 $\{0,1\}^n$ 的布尔函数，满足 $g_\sigma(x) = g(\sigma^{-1}(x))$。

最后，我们指出，仅仅重新映射坐标并不会改变相对距离：

事实 6. 考虑函数 $f, g\colon \{0,1\}^{\mathsf{X}} \to \{0,1\}$ 以及双射 $\pi\colon \mathsf{Y} \to \mathsf{X}$。我们有

$$\mathsf{reldist}(f_\pi, g_\pi) = \mathsf{reldist}(f, g).$$

3.4 项、立方体与距离

一个项 $T = \ell_1 \wedge \cdots \wedge \ell_k$ 是变量 $x_1, \ldots, x_n$ 上的布尔文字的合取。项的宽度是其中包含的文字数量。不失一般性，我们假设任何析取范式中的项都不同时包含某个变量 x_i 及其否定 $\overline{x_i}$。

对我们而言，将项视为文字集合 $T = \{\ell_1, \ldots, \ell_k\}$ 通常会很方便；根据上下文，总能看出我们将 T 视为合取还是文字集合。因此，给定两个项 T, T'，我们用“$T \cap T'$”表示由文字 $\wedge_{\ell \in T \cap T'} \ell$ 合取而成的项（注意，这不同于两个项 $T \wedge T'$ 的合取，后者是项 $\wedge_{\ell \in T \cup T'} \ell$）。类似地，我们用“$T \setminus T'$”表示合取 $\wedge_{\ell \in T \setminus T'} \ell$。最后，对于文字集合 A，我们用 $\mathsf{var}(A)$ 表示与 A 中的文字相对应的变量集合（例如，若 $A = \{x_2, \overline{x_2}, x_3\}$，则有 $\mathsf{var}(A) = \{x_2, x_3\}$）。

这将由我们来做。eful to have notation for the subcube of $\{0,1\}^n$ that is “spanned” by two 点 $:$

定义7（立方体）。给定两个 n 位字符串 $a,\ b \in \{0,1\}^n$ 我们用 $\mathsf{Cube}(a,\ b)$ 表示 $\{0,1\}^n$ 的子立方体，它以 a 和 b 为对踵点，即，

$$\mathsf{Cube}(a,b) := \left\{z \in \{0,1\}^n : z_i = a_i = b_i \text{ for all } i \text{ such that } a_i = b_i\right\}$$

很容易看出，如果 $a,\ b \in \{0,1\}^n$ 满足析取范式 f 的同一个项 T，那么 $\mathsf{Cube}(a,\ b) \subseteq f^{-1}(1)$。不过，反方向未必成立；作为一个简单例子，考虑析取范式 $f = (x_2 \wedge x_3) \vee (x_1 \wedge x_3)$ 和点 $a = 000,\ b = \overline{001}.$。我们有 $\mathsf{Cube}(a,\ b) = \{a, b\} \subseteq f^{-1}(1)$，但 a 和 b 不满足 f 的同一个项。

有时，我们将项 T 到项 T' 的距离定义为属于 T' 但不属于 T. 的所有文字构成的集合 $|T' \setminus T|$ 的大小。我们还将使用以下定义来描述一个点距离满足某个项有多远：

定义 8（点到项的距离 T）。给定 $x \in \{0,1\}^n$ 和项 T，我们用

$$\mathrm{dist}(x,T) := \left|\{\ell : \ell \text{ is a literal in } T \text{ with } \ell(x) = 0\}\right|.$$

如果 $\mathsf{dist}(x,T) \geq k$，我们称 x 与 T 相距至少 k；等价地，也可以说“x 在至少 k 个坐标上与 T 不一致”。

引理 9。设 $a,\ b \in \{0,1\}$ 和 T 是一个项，且 a, b 中至少一个与 T 相距 k。该 n

$$\mathop{\mathbf{Pr}}_{z \sim \mathsf{Cube}(a,b)}\left[T(z) = 1\right] \leq 2^{-k}.$$

证明。不失一般性，假设 a 与 T 在坐标集合 S 上不一致，其中 $|\mathsf{S}| \geq k$。如果 $a_i = b_i$ 对某个坐标 $i \in S$ 成立，那么立方体 $\mathsf{Cube}(a,\ b)$ 中没有任何点能满足 T，因此所讨论的概率为零。所以我们可以假设 b 与 a 在 S 中的所有坐标上都不一致。为了使点 $z \in \mathsf{Cube}(a,\ b)$ 满足 T，它必须在 S 中的所有坐标上都取正确值。因此 $\mathbf{Pr}_{z \in \mathsf{Cube}(a,b)}[T(z) = 1] \leq 1/2^{|\mathsf{S}|} \leq 2^{-k}$. □

4 析取范式项的聚类

本节旨在确立关于 s 项析取范式的结构性结果。该结果表明，对于任意一个 s 项析取范式 f 的表示，以及任意正整数 K，都存在一种唯一的划分（“K-聚类”），将 f 的项划分为互不相交的“簇”（子析取范式），记作 Clustering(f, K)。这种划分具有一些有用的性质，我们将在之后对析取范式相对误差测试算法的“是”情形进行分析时用到这些性质。粗略地说，我们对 Clustering(f, K) 所需的性质如下：

1. 对于 Clustering(f, K) 中任意一个由项构成的簇 C，所有项彼此“看起来相似”（见引理 13）。更形式化地说，C 的子析取范式可以写成 $T \wedge (T_1 \vee \cdots \vee T_{|C|})$，使得涉及 T_1、……、$T_{|C|}$ 的变量数量有界于 $O(s^2K)$。注意，这个界关于 s 和 K 是多项式的，但与 n 无关。2. 对于 Clustering(f, K) 中两个不同簇内的任意 T 和 T'，若 T 不太宽，则有 $|T' \setminus T| = \Omega(K)$（见推论 15）。这意味着，对于均匀随机满足赋值 $\boldsymbol{a} \sim$ SAMP(f)，若它满足项 T，那么它与任何不属于 T 所在簇的项 T' 之间距离 dist($\boldsymbol{a}, T'$) 很小的概率极低。（精确表述见推论17。）

在本节中，我们固定 f 为一个 s 项析取范式，并固定 f 的一个特定表示形式，即一个 s 项析取范式。

4.1 s 项析取范式的 K 聚类

在本小节中，我们定义并解释什么是 s 项析取范式 f 的 K 聚类 。

注记10。如上所述，给定 f 的一个 s 项析取范式表示，f 的 K 聚类是唯一的：正如下文所述，它是通过对 f 的项集合执行一个确定性过程得到的。我们强调，这一确定性过程并非由我们的测试算法执行；它仅用于从概念上定义并分析 f 的 K 聚类。

定义11。f 的一个簇 C 是 f 的项的一个非空子集。每个簇 C 都有一个相应的标签 (T^*, S^*)，其中 T^* 和 S^* 都是文字集合，定义如下：

1. T^* 是 C 中所有项共有的文字集合（等价地，如果我们将 C 中的项视为文字集合，那么 T^* 就是它们的交集）；并且 2. S^* 由文字 x_i 和 x_i 组成，其中对所有满足以下条件之一的 $i \in [n]$：(1) x_i 出现在 C 的某些项中但并非所有项中；或 (2) x_i 出现在 C的某些项中但并非所有项中。（将簇的标签（T^*,S^*）与本节开头提到的关于 Clustering(f, K)的第一个性质联系起来，可以注意到，簇 C 的子析取范式可写为$T^* \wedge (T_1 \vee \cdots \vee T_{|C|})$，其中 T_1、……、$T_{|C|}$ 中的文字都来自 S^*。这就是为什么引理 13 要对 Clustering(f, K)中的每个簇，将 $|S^*|$ 上界为 $O(s^2K)$。）

由上述定义可知：（1） S^* 是一个在否定运算下封闭的文字集合；（2） T^* 与 S^* 不相交。

最后，对 f 进行聚类，得到一组互不相交的簇，其项划分了 s 的项f。正如算法 2所示，每个簇的标签在创建该簇时生成。

算法 2 构造析取范式 f 的 K-聚类过程

输入： 一个 s-项析取范式 f 和一个正整数 K。
输出： f 的项的 K-聚类 Clustering(f, K)。

1: 令 L 为 f 的项集合，并令 Clustering $= \emptyset$ 初始为空。
2: **当** $L \neq \emptyset$ **时，执行**
3: 在 L 中宽度最小的项里，选取字典序最小的项 T。
4: 从 L 中移除 T，并创建一个新的簇 $C = \{T\}$，其标签为（$T^* = T$， $S^* = \emptyset$）。
5: **当** 存在一个项 $T' \in L$ 满足以下条件时：

$$\left|\{\ell \in T' \text{和 } \ell \notin T^* \cup S^*\}\right| \leq 2K$$

do
6: 从 L中选取这样一项中，字典序最小的项 T'。
7: 从 L 中移除 T'，并将其添加到 C中。
8: 按如下方式更新 C 的标签： $T^* \leftarrow T^* \cap T'$ 和 $S^* \leftarrow S^* \cup \{\ell, \ell : \ell \in (T^* \triangle T')\}$
9: **结束 while 循环**
10: 将 C 添加到聚类中。
11: **结束 while 循环**
12: 输出聚类。

算法 2：聚类算法

算法 2 给出一个确定性过程，该过程以一个 s 项析取范式 f 以及一个正整数参数 K 作为输入，并输出一种特定的聚类，称为 f 的 K-聚类，记作 Clustering(f, K)。粗略地说，该聚类的构造过程如下：通过将最窄项作为初始簇，并迭代地向其中添加其他“与该簇相距不太远”的项，形成一个簇 C（详情见算法 2 第 5–9 行的 while 循环）。当不存在这样的项时，簇 C 即告完成，并被添加到 Clustering(f, K) 中；随后重复该过程（即以剩余最窄项为起点建立新簇），直至 f 中的所有项都被聚类。我们还指出，通过检查第 4 行和第 8 行（这是仅有的设置 T^* 或 S^* 的行）可轻易看出，每个簇的标签（T^*, S^*）都按定义 11 中的规定得到维护。

令 C 为 Clustering(f, K) 中的一个簇，并令 T 为添加到 C 中的第一个项（因此它必定是 C 中最窄的项之一）。令 (T^*, S^*) 为 C 的标签。关于 f 的 K-聚类，我们的第一个引理指出，$|T^* \cap S^*|$ 相对于 $|T|$ 不能太大：

引理 12（簇标签大小界）。令 C 为 Clustering(f, K) 中任意一个带有标签 (T^*, S^*) 的簇，并令 T 为添加到 C 中的第一个项。那么

$$|T^* \cup S^*| \leq |T| + 4sK.$$

证明。当簇 C首次在 算法 2的第 5 行创建时，初始 $T^* \cup S^*$ 包含 $|T|$个文字，因为 $T^* = T$且 $S^* = \emptyset$。每当簇 C通过添加新项 T'而增长时

（在第 6—8 行中），$T^* \cup S^*$ 中出现的每个新文字必须属于 $T' \setminus (T^* \cup S^*)$，或者是此类文字的否定。因此，根据第 6 行中的条件，每当簇增长时，$T^* \cup S^*$ 中的新文字数量最多增加 $4K$。由于簇 C 最多可以增长 s 次，引理得证。

□

引理 13. 设 C 是 Clustering(f, K) 中一个标记为 (T^*, S^*) 的簇。那么 $|S^*| \leq 16s^2K$。

证明。设 T 是创建 C 时加入的第一项。对于之后加入 C的任意其他项 T'，在其加入时，根据第 5 行的条件，必有 $|T' \setminus (T^* \cup S^*)| \leq 2K$。另一方面，$|T^* \cup S^*| \leq |T| + 4sK$根据 引理 12。由于 $|T'| \geq |T|$，我们有

$$|T' \cap (T^* \cup S^*)| \geq |T| - 2K$$

因此，

$$|(T^* \cup S^*) \setminus T'| \leq (|T| + 4sK) - (|T| - 2K) \leq 6sK$$

由此可知，$(T^* \triangle T') \setminus S^*$ 中的文字数至多为 $6sK + 2K \leq 8sK$；回顾第 8 行可知，每当将项 T' 加入 C时，S^* 的大小最多增加 $16sK$。

□

引理 14. 如果任意两个项 A, B of f满足 $|A| \leq |B| + K$ and $|B \setminus A| \leq K$，那么它们必定属于 Clustering(f, K) 中的同一个簇。

证明。在执行Algorithm2的某个时刻，必定会出现这种情况：A, B 中有一个被加入当前簇 C，而另一个仍在列表中 L。

首先，假设 A被加入当前簇 C，而 B仍在 L 中。设 (T^*, S^*) 为 C 在 A 被加入后的标签。根据簇的标签定义，我们有 $A \subseteq T^* \cup S^*$，且在跨越第 5 至 9 行的循环执行期间，这一关系将始终成立。由于 $|B \setminus A| \leq K$,B 最终会在该循环中被加入 C。

另一种可能是，B 被加入当前簇 C，而 A 仍在 L 中。由于$|B \setminus A| \leq K$，我们有 $|A \setminus B| \leq K + (|A| - |B|) \leq 2K$。类似的论证表明，$A$ 最终会在 while 循环期间被加入 C。

□

回忆一下，$w_{\min}$ 是 f中任意项的最小宽度。

推论 15. 设 T, T'是 f的两个项，其中 T满足 $|T| \leq w_{\min} + K$，且 T, T'分别属于 Clustering(f, K) 中的两个不同簇。那么 $|T' \setminus T| > K$。

证明. 由 $|T| \leq w_{\min} + K$，可得 $|T| \leq |T'| + K$。于是 $|T' \setminus T| \leq K$将意味着T, T'属于 Clustering(f, K) 中的同一个簇，这由引理 14可得。

□

展望后文，在论文的其余部分，我们将令 K 为 $\log^2(s/\varepsilon)$。采用这一K后，我们可以利用下面的引理证明：如果 $\boldsymbol{a} \sim f^{-1}(1)$，且 $\boldsymbol{a}$ 满足一个宽度为$\leq w_{\min} + K$的项 T，那么对于任何属于不同簇的 T'，$\boldsymbol{a}$ 都极不可能（即其概率关于 s/ε呈准多项式级小）与 T'“接近”。

引理 16。 设 g 是一个析取范式，T 是 g 的一项，T' 是满足 $|T' \setminus T| = k$ 的任意一项。那么，

$$\mathop{\mathbf{Pr}}_{\boldsymbol{a} \sim g^{-1}(1)} \left[\operatorname{dist}(\boldsymbol{a}, T') \leq (k/4) \,\middle|\, T(\boldsymbol{a}) = 1 \right] \leq e^{-k/8}.$$

证明。 令 S 为集合 var($T' \setminus T$)。在给定 $\boldsymbol{a} \sim g^{-1}(1)$ 满足 T 的条件下，对于每个 $i \in$S，彼此独立地， $\boldsymbol{a}_i$ 要么在 $\boldsymbol{a}_i$ 中均匀随机分布，要么 $\boldsymbol{a}_i$ 对于 T' 取值错误的概率为 1。对于每个 $i \in$ S，令 $\mathbf{X}_i \in \{0,1\}$ 为二元随机变量：若 $\boldsymbol{a}_i$ 对于 T' 取值正确，则取值 0；否则取值 1。注意，$\mathbf{X}_i$ 彼此独立。

我们有 dist($\boldsymbol{a}$, T') $\geq \sum_{i \in \mathsf{S}} \mathbf{X}_i$，并且 $\mathbf{E}$[dist($\boldsymbol{a}$, T')] $\geq \mathbf{E}\left[\sum_{i \in \mathsf{S}} \mathbf{X}_i\right] \geq |\mathsf{S}|/2 = k/2$。因此，根据标准的 Chernoff 界，我们有

$$\Pr_{\boldsymbol{a} \sim g^{-1}(1)}\left[\operatorname{dist}(a, T') \leq k/4 \,\middle|\, T(\boldsymbol{a}) = 1\right] \leq e^{-k/8}. \quad \square$$

特化 引理 16 在我们的上下文中，其中 f 是一个 s-项析取范式，我们有如下结论：

推论17。 令 T为 f的一个项，满足 $|T| \leq w_{\min} + K$，并令 T'为 f的任意一个项，且该项位于 Clustering(f, K) 中与 T 不同的簇中。则

$$\Pr_{\boldsymbol{a} \sim f^{-1}(1)}\left[\operatorname{dist}(\boldsymbol{a}, T') \leq (K/4) \,\middle|\, T(\boldsymbol{a}) = 1\right] \leq e^{-K/8}.$$

证明。 由于 T、T'位于不同的簇中，且 T的宽度为 $\leq w_{\min} + K$，根据 推论 15 我们有 $|T' \setminus T| \geq K$。该结论现在由 引理 16得出。 $\square$

4.2 簇对应于因子化析取范式

如引言中所述，这种聚类产生了一种表示方式：将 s 项析取范式 f表示为若干个 因子化析取范式的析取。我们接下来对其进行形式化定义：

定义 18。 函数 $h : \{0,1\}^n \to \{0,1\}$ 是一个（r, μ）-因子化析取范式，当且仅当其形式为 $H \wedge (T_1 \vee \cdots \vee T_{\leq r})$，其中

1. H 是一个项，称为因子化析取范式的头部。（注意， H中的宽度（文字的数量）没有上界。）
2. $T_1 \vee \ldots \vee T_{\leq r}$是若干项的析取，称为因子化析取范式的尾部 F.
3. H 中的变量与尾部中的变量互不相同，且尾部中出现的变量总数至多为 μ。

在非正式的直观讨论中，我们有时会提到“因子化析取范式”，而不指定 r 和 μ的取值。容易验证，对于每个簇，簇中各项的析取都可以等价地写成一个（r, μ）-因子化析取范式，其中 T^* 中的文字构成头部， r是簇的大小，并且 $\mu \leq |S^*|$:

断言 19。 令 $h : \{0,1\}^n \to \{0,1\}$ 是一个由 k 个项组成的析取范式，这些项全部属于 Clustering(f、K) 的同一个簇 C。那么 h 是 $\mu = 16s^2K$ 的（k、μ）因子化析取范式。

证明。 令 (T^*, S^*)为 C 的标签。我们有 h可以写成 $T^* \wedge (T'_1 \vee \cdots \vee T'_k)$，其中$T'_1$，……，$T'_k$均由 S^* 中的文字构成，并且根据 引理 13 我们有 $|S^*| \leq \mu$。根据 定义11我们有 var(T^*) $\cap$ var(S^*) $= \emptyset$，因此结论得证。 $\square$

算法3Test-DNF

输入： 成员查询(f) 和 SAMP(f)，针对某个函数 f：$\{0,1\}^n \to \{0,1\}$, s和 ε

输出： 接受或拒绝。

1：令 K =$\log^2(s/\varepsilon)$ 和 $\mu = 16s^2K$。2：运行 Find-Factored-DNFs(MQ(f), SAMP(f) s, ε)。3：如果其拒绝，则拒绝；否则，令 (MQ_i^*, SAMP_i^*)，$i \in [s']$ 表示返回的 $s' \le s$ 对。4：对于每个 $i \in [s']$，执行 5：对于每个 $r \in [s]$，执行 6：运行 Test-Factored-DNF(MQ_i^*, SAMP_i^*, $r, \mu, \varepsilon/(2s)$)，共 $O(\log\ s)$ 次。7：结束循环。8：令 r_i 为使多数运行接受的最小 r；如果不存在这样的 r，则拒绝。9：结束循环。10：如果 $\sum_{i\in[s']} r_i \le s$，则接受；否则拒绝。

算法3： **Test-DNF**。一种用于测试函数是否 ε 在相对距离意义下接近于一个s项析取范式的算法

5 主算法及其分析

本节介绍用于在相对距离下测试 **析取范式** 的主算法 s项析取范式，并利用它证明 定理 1。为此，我们将陈述并假设两个定理（定理 20 和 23），这些定理将在论文的其余部分中得到证明。

5.1 在相对距离下测试因子化析取范式

展望后文，第 7 至 9 节合起来得到的主要结果，是下面关于在相对距离下测试（r, μ）-因子化析取范式的定理。

定理 20。 存在一个随机化算法 **Test-Factored-DNF**，它接收以下输入：(1) 成员查询(h)和 SAMP(h)，即某个函数 h：$\{0,1\}^n \to \{0,1\}$；(2) 正整数 r和 μ；以及 (3) 距离参数 ε。其查询复杂度为 poly(r/ε, μ)，并满足以下性质：

1. I若 h 是 (r, μ)-因子化析取范式，则其接受概率至少为 0。9；一个 d
2. 若对于每个 (r, μ)-因子化析取范式 h'，reldist(h, h') $> \varepsilon$，则其拒绝概率至少为 0。9。

5.2 过程 Find-Factored-DNFs

假设定理 20 成立，理想情况下，如果我们能设计一个过程来“划分”输入函数 f： $\{0,1\}^n \to \{0,1\}$，那么就可以使用 Test-Factored-DNF 在相对距离下测试一般的 s-项析取范式，具体如下：给定对 MQ(f) 和 SAMP(f) 的访问权限，我们希望这个“划分”过程（将其命名为 Find-Factored-DNFs）能够使用 poly(s/ε) 次查询，创建一个至多包含 s 对 (MQ_i^*,SAMP_i^*) 的列表，$i \in [s']$ 其中某个 $s' \le s$ 满足 MQ_i^* 和 SAMP_i^* 是一对

随机算法（可访问 $\mathrm{MQ}(f)$和 $\mathrm{SAMP}(f)$，且具有 $\mathrm{poly}(s/\varepsilon)$查询复杂度），用于实现某个函数 h_i 的成员查询和采样预言机：$\{0,1\}^n \to \{0,1\}$，[5]分别满足

1. 当 f是一个 s-项析取范式时，数值 s'对（成员查询$_i^*$、SAMP$_i^*$）的数量与聚类(f、 K)中的簇数相同，其数量为 $K = \log^2(s/\varepsilon)$；并且每个簇 C 都对应一对（成员查询$_i^*$、SAMP$_i^*$），使得函数 h_i 是由簇 C中的项构成的子析取范式。

2. 当 f 与 s-项析取范式的相对距离至少为 ε时，函数 h_i 满足$h_1^{-1}(1)\ldots,h_{s'}^{-1}(1) \subseteq f^{-1}(1)$，且它们的并集为 $f^{-1}(1)$。

在以下理想的性能保证下，**Find-Factored-DNFs**我们将在下文说明，**Test-DNF**（见算法3）是一个$\mathrm{poly}(s/\varepsilon)$-复杂度测试器，用于测试相对距离下的 s-项析取范式。

为说明这一点，我们首先注意到，**Test-DNF** 的查询复杂度为 $\mathrm{poly}(s/\varepsilon)$，这是基于**Test-Factored-DNF**的查询复杂度为 $\mathrm{poly}(s/\varepsilon)$ ，且每次调用 成员查询$_i^*$ 和 SAMP$_i^*$会向 $\mathrm{poly}(s/\varepsilon)$ 个查询 $\mathrm{MQ}(f)$和 $\mathrm{SAMP}(f)$。为证明 **Test-DNF** 的正确性，我们考虑以下两种情况：

I. 当 f <style id='4}>是一个 s项析取范式时，根据引理 13以及**Find-Factored-DNFs** 的性能保证可知，每个 h_i 都是一个（$|C_i|$，μ）-因子化析取范式，其中我们记 C_i 为聚类（f，K）中对应于 h_i的簇。根据定理 20、标准的 Chernoff 界和并集界可知，以高概率，$r_i \le |C_i|$ 对所有$i \in [s']$均成立。当此情况发生时，**Test-DNF** 接受，因为 $\sum_i r_i \le \sum_i |C_i| \le s$。

II. 当 f 在相对距离意义下与每个 s-项析取范式都相差 ε时，令 t_i 为满足以下条件的最小整数：h_i 在相对距离意义下与一个（t_i，μ）-因子化析取范式的距离为 $\varepsilon/(2s)$。根据定理 20、标准的 Chernoff 界和并集界可知，以高概率，对于所有 $i \in [s']$，我们都有 $r_i \ge t_i$，因此 $\sum_i r_i \ge \sum_i t_i$。另一方面，下面这个简单的引理表明，鉴于 f 在相对距离意义下与 s-项析取范式相差 ε，必有 $\sum_i t_i > s$。因此，Test-DNF 以高概率拒绝。

引理 21。设 $h_1,\ldots,h_{s'}$为函数，使得 $h_1^{-1}(1)\ldots,h_{s'}^{-1}(1) \subseteq f^{-1}(1)$，且它们的并集为$f^{-1}(1)$。若对于每个 $i \in [s']$ 和 $\sum_i t_i \le s$，h_i 在相对距离上与某个 t_i-项析取范式的距离至多为 $(\varepsilon/(2s))$，则 f 必定在相对距离上与某个 s-项析取范式的距离至多为 ε。

证明。设 h_i' 是一个 t_i-项析取范式，使得 $\mathrm{reldist}(h_i, h_i') \le \varepsilon/(2s)$。令 g 为 h_i'与 $i \in [s']$的析取。由于 $\sum_i t_i \le s$，g 是一个 s-项析取范式，并且我们有

$$\mathrm{reldist}(f,g) = \frac{|f^{-1}(1) \triangle g^{-1}(1)|}{|f^{-1}(1)|} \le \frac{\sum_i |h_i^{-1}(1) \triangle h_i'^{-1}(1)|}{|f^{-1}(1)|} \le \sum_i \frac{|h_i^{-1}(1) \triangle h_i'^{-1}(1)|}{|h_i^{-1}(1)|} \le s' \cdot \frac{\varepsilon}{2s} < \varepsilon,$$

使用 $s' \le s$。至此，引理的证明完成。 □

[5]正如读者可能猜到的那样，MQ$_i^*$ 和 SAMP$_i^*$ 背后的随机算法无法完美实现所需的预言机 MQ(h_i) 和 SAMP(h_i)；我们稍后会详细讨论这个问题，并且始终使用 $*$ 标记由随机算法模拟的不完美预言机，例如 MQ$_i^*$ 和 SAMP$_i^*$。在当前的概述中，读者可以假定 MQ$_i^*$ 和 SAMP$_i^*$ 确实是针对 h_i的忠实预言机。

遗憾的是，我们未能得到一个过程**Find-Factored-DNFs** ，能够精确实现上述条目 (1.) 和 (2.) 所述的理想性能保证。在介绍 **Find-Factored-DNFs** 的定理之前，我们先讨论 h_i 函数 上实际保证与理想保证之间的差异：

1′。当 f 是一个 s-项析取范式时， h_i （即第 i 个二元组（成员查询 MQ_i^*，SAMP_i^*）所实现的函数）仍然是一个析取范式，但并不严格等于 Clustering(f, K) 中某个簇的子析取范式。相反，令 D_i 为构成 h_i 的项的集合，则对于 Clustering(f, K) 中的某个簇 C，有 $D_i \subseteq C$，并且 D_i 两两不相交，由此可得 $\sum_i |D_i| \leq s$。（注意， D_i 的并集不一定覆盖 f 中的所有项。直观地说，这是因为 f 可能包含一些“非常宽”的项，它们只有极少的满足赋值，因此我们无法找到它们。）

2′。当 f 在相对距离意义下与 s-项析取范式相差 ε 时，我们仍有 $h_i^{-1}(1) \subseteq f^{-1}(1)$，但 $h_i^{-1}(1)$ 的并集只能近似覆盖 $f^{-1}(1)$，即，与 $|f^{-1}(1)|$ 相比， $|f^{-1}(1) \setminus (\cup_i h_i^{-1}(1))|$ 很小。

很容易验证，如果这些是唯一的差异，那么 **Test-DNF** 的正确性就可以通过与上文类似的论证得出。（特别是，引理 21 的证明足够稳健，可以容忍上文第 2 项中的近似。）

然而，主要的挑战在于，$(\mathrm{MQ}_i^*, \mathrm{SAMP}_i^*)$ 背后的随机算法并不是 h_i 的预言机的完美实现。为此，我们引入以下定义：

定义 22。设 ALG 是一个可以访问某个函数$h : \{0,1\}^n \to \{0,1\}$ 的 MQ(h) 和 SAMP(h) 的算法，该算法最终会接受或拒绝。若满足以下条件，我们称 $(\mathrm{MQ}^*, \mathrm{SAMP}^*)$ 是 $h : \{0,1\}^n \to \{0,1\}$ 关于 ALG 的预言机的充分实现：

1. MQ^* 是一个随机化算法，对于任意给定的 $x \in \{0,1\}^n$，都会返回 0 或 1；
2. SAMP^* 是一种随机化算法，始终返回一个点 $x \in \{0,1\}^n$；并且
3. 我们有

$$\left| \mathbf{Pr}\left[\mathsf{ALG}\big(\mathrm{MQ}(h), \mathrm{SAMP}(h)\big) \text{ accepts}\right] - \mathbf{Pr}\left[\mathsf{ALG}\big(\mathrm{MQ}^*, \mathrm{SAMP}^*\big) \text{ accepts}\right] \right| \leq 0.1.$$

令 $K = \log^2(s/\varepsilon)$ 和 $\mu = 16s^2K$。有了 定义 22 之后，我们陈述关于 **Find-Factored-DNFs** 的主定理，并将在第 6 节中证明。

定理 23. Find-Factored-DNFs 的输入为：(1) 对某个函数 $f : \{0,1\}^n \to \{0,1\}$ 的 MQ(f) 和 SAMP(f) 的访问权限；以及 (2) 两个参数 s 和 ε。它对 MQ(f) 和 SAMP(f) 进行 poly(s/ε) 次查询，最后要么拒绝，要么返回一个对列表 $(\mathrm{MQ}_1^*, \mathrm{SAMP}_1^*), \ldots, (\mathrm{MQ}_{s'}^*, \mathrm{SAMP}_{s'}^*)$，使其满足以下条件：

1. 对于每个 $i \in [s']$，MQ_i^* 和 SAMP_i^* 是两个随机算法，它们可以访问 MQ(f) 和 SAMP(f)，且查询复杂度为 poly(s/ε)；
2. 当 f 是一个 s项析取范式时，Find-Factored-DNFs 以至少 0.9 的概率不会拒绝，并且存在 $s' \leq s$ 个布尔函数 $h_1, \ldots, h_{s'} : \{0,1\}^n \to \{0,1\}$ 使得 (a) 每个 h_i 都是一个 k_i项析取范式，且满足 $\sum_{i=1}^{s'} k_i \leq s$；(b) 每个 h_i 都是一个（k_i, μ）因子化析取范式；以及 (c) 对每个 $i \in [s']$，（MQ_i^*, SAMP_i^*）都是针对 h_i的预言机的充分实现，关于 Test-Factored-DNF($\cdot$, $\cdot$, k_i, μ, $\varepsilon/(2s)$)。

3. 当 f 是 ε与每个 s项析取范式的相对距离至少为 0.9 时，**Find-Factored-DNFs** 要么拒绝，要么存在 $s' \le s$ 个函数$h_1, \ldots, h_{s'}$: $\{0,1\}^n \to \{0,1\}$ 使得 (a′) 对每个 $i \in [s']$，都有 $h_i^{-1}(1) \subseteq f^{-1}(1)$，并且

$$\frac{|f^{-1}(1) \setminus (\cup_i h_i^{-1}(1))|}{|f^{-1}(1)|} \le \frac{\varepsilon}{2}; \tag{1}$$

且（b′）对于每个 $i \in [s']$ 以及任意 $r \in [s]$，（成员查询$_i^*$，SAMP$_i^*$）都是针对 h_i 关于 **Test-Factored-DNF**（$\cdot$，$\cdot$，r，μ，ε的预言机的充分实现 /(2s)).

5.3 在假设定理1 成立的情况下证明定理 20 和定理 23

首先注意到，由于根据定理 20（以及对 $\mu = 16s^2K$ 和 $K = \log^2(s/\varepsilon)$ 的选择），Test-Factored-DNF 的查询复杂度为 poly(s/ε)，且根据定理 23，每次调用 MQ$_i^*$和 SAMP$_i^*$都会向 MQ(f) 和 SAMP(f) 发出 poly(s/ε) 次查询，因此 Test-DNF 的查询复杂度显然为 poly(s/ε)。为了证明 Test-DNF 的正确性，我们考虑以下两种情况:

- 当 f 是一个 s-项析取范式时，由定理 23 和引理 13 可知，每个 h_i 都是一个（k，μ）- 因子化析取范式。进而，由定理 20 、定理 23 （充分实现部分）、Chernoff 界和并集界可知，以高概率，对所有 $r_i \le k_i$， $i \in [s']$均成立。当这种情况发生时，**Test-DNF** 会接受，因为 $\sum_i r_i \le \sum_i k_i \le s$，依据定理 23 给出的保证 $\sum_{i=1}^{s'} k_i \le s$。

- 当 f 在相对距离意义下与所有 s-项析取范式都相距至少 ε时，根据定理 23, Find-Factored-DNFs 以至少 0.9 的概率要么拒绝（此时 Test-DNF 也会拒绝），要么返回一个预言机对列表；这些预言机对满足定理 23 针对这种情况所详述的条件，并对应于函数 $h_1, \ldots, h_{s'}$。令 t_i 为最小整数，使得 h_i 在相对距离意义下与某个 (t_i, μ)- 因子化析取范式的距离至多为 (ε/(2s))。由此，根据定理 20、Chernoff 界和并集界可知，以高概率，对所有 $i \in [s']$ 都有 $r_i \ge t_i$，因此 $\sum_i r_i \ge \sum_i t_i$。另一方面，下面对引理 21 的推广表明，若已知 f 在相对距离意义下与 s-项析取范式相距至少 ε，那么在定理 23 中给出的关于 h_i^{-1}(1) 与 f^{-1}(1) 的条件下，必有 $\sum_i t_i > s$。因此，Test-DNF 以高概率拒绝。

引理 24。设 $h_1, \ldots, h_{s'}$是满足 h_i^{-1}（1）$\subseteq f^{-1}$（1）对每个 i以及 公式（1） 的函数。如果 h_i 在 (ε/(2s))相对距离上与一个 t_i 项析取范式相差不超过 $i \in [s']$，且 $\sum_i t_i \le s$，那么 f必定在相对距离上与一个 ε项析取范式相差不超过 s。

证明。对每个 $i \in [s']$，令 h_i' 为一个 t_i 项析取范式，使得相对距离(h_i, h_i') $\le \varepsilon/(2s)$。令 $g = h_1' \vee \cdots \vee h_{s'}$。那么 g是一个 s 项析取范式，因为 $\sum_i t_i \le s$，并且我们有

$$\text{reldist}(f,g) = \frac{|f^{-1}(1) \triangle g^{-1}(1)|}{|f^{-1}(1)|} \le \frac{|(\cup_i h_i^{-1}(1)) \triangle g^{-1}(1)|}{|f^{-1}(1)|} + \frac{\varepsilon}{2},$$

分别使用公式（1）的第一部分和第二部分。另一方面，我们有

$$\frac{|(\cup_i h_i^{-1}(1)) \triangle g^{-1}(1)|}{|f^{-1}(1)|} \le \frac{\sum_i |h_i^{-1}(1) \triangle h_i'^{-1}(1)|}{|h_i^{-1}(1)|} \le \frac{\varepsilon}{2}$$

使用 h_i^{-1}(1) $\subseteq f^{-1}$(1) 和 $s' \le s$。因此我们有 reldist(f, g) $\le$(ε/2) +(ε/2) = ε。 □

6 创建预言机：Find-Factored-DNFs 及定理 23

在本节 节中，我们介绍过程 Find-Factored-DNFs 并证明定理 23 .

设 $f:\{0,1\}^n \to \{0,1\}$ 是我们希望检验的未知函数，检验其是否为一个 s 项析取范式，所用的相对误差参数为 ε。与前几节一样，我们可以假设 s/ε至少是某个足够大的绝对常数。

在本节余下部分中，我们使用以下参数：

- 如前所述，我们有 $K := \log^2(s/\varepsilon)$ 和 $\mu := 16s^2K$。
- 设 $Q = \mathrm{poly}(s/\varepsilon)$ 为查询复杂度（调用 成员查询(f) 和SAMP(f) 的总次数）的上界，其中 Test-Factored-DNF$(\cdot,\cdot,r \le s,\mu,\varepsilon/(2s))$。
- 令 N 为关于 s/ε的一个充分大的多项式，使得

$$N \ge Q^{10} \quad \text{and} \quad N \ge (s/\varepsilon)^{10}. \tag{2}$$

- 最后，令 $\alpha = \log N = O(\log(s/\varepsilon))$。

6.1 Test-Factored-DNF： 类型 1 和类型 2查询的关键方面概述

为简化 Test-Factored-DNF与 Find-Factored-DNFs之间的联系，让我们总结一下为证明 Test-Factored-DNF所输出的预言机对于 Test-Factored-DNF 是充分的，所需了解的 Test-Factored-DNF性质。在后续论证中（特别是为了证明 ′的第 2 项 (c) 部分和第 3 项 (b定理 23) 部分），只需考虑这些性质。

如上所述，对于任意函数 $h:\{0,1\}^n \to \{0,1\}$， Q =poly(s/ε) 是 Test-Factored-DNF(MQ(h), SAMP(h) $r \le s,\mu,\varepsilon/(2s)$) 的最大总查询复杂度的上界。通过检查 Test-Factored-DNF 及其所包含的子程序，可以验证在整个执行过程中，Test-Factored-DNF 会向 MQ(h) 发起两种不同类型的查询，我们称之为类型 1 查询和类型 2 查询。在 Test-Factored-DNF 的每个子程序代码中，每次出现这两类查询之一时，都会用相应的颜色标示（类型 1 为红色，类型 2 为蓝色）。

在进行任何类型 1 或 类型 2查询之前，Test-Factored-DNF 会随机划分$[n]$ 为 $\tau := \Theta(\mu^2)$ 个块 $\mathbf{X}_1,\ldots,\mathbf{X}_m$；这是在 Approximate的阶段 1 中完成的。现在我们可以描述这两种查询：

类型 1 查询：类型 1 查询具有以下结构：Test-Factored-DNF 一次性地（在发出任何类型 1 查询之前）从 $\{0,1\}^n$ 中抽取至多 Q 个点 $\boldsymbol{y} \sim$ SAMP(h)，并从 $\{0,1\}^n$ 中均匀抽取至多 Q 个点 $\boldsymbol{w}$。分别用 Y 和 W 表示这两组点。类型 1 查询是对 MQ(h) 发出的查询，查询的字符串 z 具有以下结构：存在 $\boldsymbol{y} \in$Y、 $\boldsymbol{w} \in$ W 和一个块 $\mathbf{X}_\ell$，使得对于任意$i \in [n] \setminus \mathbf{X}_\ell$，都有 $z_i \in \{\boldsymbol{y}_i, \boldsymbol{w}_i\}$。位于 $\mathbf{X}_\ell$ 中的 z 位可以任意设置。类型 1 查询是在 Approximate（ DNFLearner 的一个子程序）中发出的。此外，

Approximate 返回一个预言机列表，供整个过程中使用 **Test-Factored-DNF**；这些预言机仅使用类型 1 查询。 [6]

类型 2 查询： 类型 2 查询是对 MQ(h) 在任意字符串 $\{0,1\}^n$上的查询。 类型 2 查询的关键性质是，**Test-Factored-DNF** 会在 $h(z) = 0$ 的任意一个 类型 2 查询下拒绝。 类型 2 查询在 **ConsCheck**中进行。

更具体地说，在运行 **Approximate** 之后，算法只通过 MQ(h)返回的预言机进行查询，**Approximate** 或者使用 类型 2 查询。

为了证明我们的模拟器相对于 Test-Factored-DNF 是 h_1、……、$h_{s'}$ 的充分实现，我们将进行如下设置：在“否情形”下，每个 MQ_i^* 都完美实现 MQ(h_i)（见第6.5节）。然而，在“肯定情形”下，我们必须分别分析两种查询。如果在 Find-Factored-DNFs 的执行过程中“一切顺利”，那么我们将证明，对于满足 $h(z) = 1$ 的所有类型 2 查询，MQ_i^* 均回答 1；并且以高概率（关于随机划分 X1、……、X$_m$ 以及 Y 和 W 的抽取），MQ_i^* 能够正确回答所有类型 1 查询（见第6.4.6节）。

6.2 当 f 是析取范式时的两个实用子程序及其分析

本小节将介绍并分析 Find-Factored-DNFs 算法中大量使用的两个子程序。在介绍这两个子程序之前，先来说明一些直观认识。直观而言，给定一次抽取 $\boldsymbol{a} \sim$ SAMP(f)，在是情形（即 f 是一个 s-项析取范式时），若能确定 $\boldsymbol{a}$满足 f 中的哪些项，将会很有帮助。遗憾的是，这个目标难以实现，因为我们不应指望能通过一个$O_{s,\varepsilon}$(1)-复杂度算法识别 f中的任何项。然而，如前所述（参见定义7及其后续讨论），如果 $a, b \in \{0,1\}^n$ 满足 f中的同一项，那么 Cube(a, b)必定是 f^{-1}(1)的子集。基于这一点，我们定义并分析两个简单的子程序：Test-Equivalence 和 In-Pool。它们能为我们提供一些部分信息，说明 a 和 b 满足 f中同一项的程度。

首先，我们需要以下简单定义：

定义25。 给定两个点 a，$b \in \{0,1\}^n$，如果立方体(a，b)中至少有 $(1/N^2)$-比例的点是 f 的满足赋值，则称立方体(a，b) 在 f 中是稠密的 。

我们将使用 以下关于**Test-Equivalence**的简单引理，其中 f 是一个 s-项析取范式 ：

引理 26。 设 f是一个 s-项析取范式，且 a，$b \in \{0,1\}^n$ 是两个点。

1. 如果 a, b满足 $T(a) = T(b) = 1$，对于 f 的某个项 T，则 Test-Equivalence(a, b, MQ(f)) 总是返回 True。2. 如果立方体(a, b) 在 f 中是稠密的，则 Test-Equivalence(a, b, MQ(f)) 以至少 $1 - 2^{-\Omega(N)}$ 的概率返回 True。3. 如果对于 f 的每个项 T，max{dist(a, T)、dist(b, T)} $\geq K/4$，那么，

$$\mathbf{Pr}\left[\textbf{Test-Equivalence}(a, b, \mathrm{MQ}(f)) \text{ returns True}\right] \leq 2^{-K/10}.$$

[6]读者可能会注意到，**Approximate** 只将 R $\subseteq [n]$ 划分为块，而不是将全部 $[n]$都划分进去。事实上，对于一个坐标 $i \in$ R，我们总是有 $z_i = \boldsymbol{y}_i$。因此，很容易看出这确实是 类型 1 查询的一种特殊情况。

算法 4~~Test-Equivalence~~

输入： a，$b \in \{0,1\}^n$，对 f 进行成员查询访问：$\{0,1\}^n \to \{0,1\}$。**输出：** 真或假。

1：重复 N^3 次：从 $z \sim$立方体(a, b) 中采样，并查询 $f(z)$。2：如果对某个采样得到的 z，$f(z) = 1$，则返回真；否则返回假。

算法 4：Test-Equivalence。算法的输入是两个点 a，$b \in \{0,1\}^n$。如果立方体(a, b) 是稠密的，算法将以高概率返回真。当 f 是析取范式时，如果存在一个项 T属于 f，使得 $T(a) = T(b) = 1$，那么算法总是返回真。

算法 5In-Pool

输入： $a \in \{0,1\}^n$，一个池 $\mathcal{P} \subseteq f^{-1}(1)$，对 f 的成员查询访问：$\{0,1\}^n \to \{0,1\}$。**输出：** 真或假。

1：**对于**每个 $b \in \mathcal{P}$ **执行**2：重复 10α 次：从 $z \sim$立方体(a, b) 中采样，并查询 $f(z)$。3：如果对所有采样得到的 z，$f(z) = 1$，则返回真。4：**结束循环**5：返回假。

算法 5：In-Pool。该算法以一个点 $a \in \{0,1\}^n$ 和一个池 $\mathcal{P}$作为输入。当 f 是析取范式时，如果存在一个项 T 属于 f 且 $b \in \mathcal{P}$ 满足 $T(a) = T(b) = 1$，算法返回真。如果对于每个 $b \in \mathcal{P}$ 立方体(a, b)，对于每个 $b \in \mathcal{P}$都不稠密，则算法以高概率返回 False。

证明。I 项 (1) 成立，因为每个点 $z \in$立方体(a, b) 都满足 T，因此 $f(z) = 1$。

项 (2) 可直接由以下事实推出：Test-Equivalence只要有一个 N^3样本落在 $f^{-1}(1)$中，就会返回真，且 $(1 - 1/N^2)^{N^3} \leq 2^{-\Omega(N)}$。

F对于项 (3)，根据 引理 9 并对 s 的 f个项应用并集界，我们有 t

$$\Pr_{z \sim \mathsf{Cube}(a,b)} \left[f(z) = 1 \right] \leq s 2^{-K/4} \leq 2^{-K/5}$$

使用 $K \gg \log(s)$。第（3）项通过对 N^3 由以下方式抽取的样本取并集界：Test-Equivalence 抽取，并使用 $K \gg \log(N)$。 □

我们将关注一些非空子集 $\mathcal{P} \subseteq f^{-1}(1)$，它们由满足赋值构成，我们称之为 池。先给出一个有用的直观理解：可以将一个池 $\mathcal{P}$ 视为由一组点 $z \sim \mathrm{SAMP}(f)$ 构成，这些点所满足的项彼此“接近”，也就是说，它们都落在 聚类(f, K) 的同一个簇中（参见 第4节）。

定义 27。 设 f 是一个 s-项析取范式。给定 $x \in f^{-1}(1)$，我们用 terms(x) 表示 f 中满足 x 的项的集合；有时我们也说这些项被 x覆盖。给定一个池$\mathcal{P} \subseteq f^{-1}(1)$，我们用 terms($\mathcal{P}$) 表示 f 中至少被一个$x \in \mathcal{P}$满足的项的集合，即它是 terms(x) 的并集，$x \in \mathcal{P}$；有时我们也说这些项被 $\mathcal{P}$覆盖。

我们证明关于子程序**In-Pool**的一个简单引理：

引理 28. 设 f 是一个 s-项析取范式。设 $\mathcal{P} \subseteq f^{-1}(1)$ 是一个池，且 $a \in \{0,1\}^n$。

1. 如果 $T(a) = 1$ 对于某个项 $T \in$ 项($\mathcal{P}$)，那么 **In-Pool**(a, $\mathcal{P}$, 成员查询(f)) 总是返回真。
2. 如果对于每个 $b \in \mathcal{P}$，Cube(a, b) 在 f 中都不是稠密的，那么

$$\mathbf{Pr}\left[\textbf{In-Pool}(a, \mathcal{P}, \mathrm{MQ}(f)) \text{ returns True}\right] \leq \frac{|\mathcal{P}|}{N^{20\alpha}}.$$

证明。第 (1) 项显然成立，因为 $T(a) = 1$ 对某些 $T \in$ 项($\mathcal{P}$) 成立，意味着 $T(b) = 1$ 对某些 $b \in \mathcal{P}$ 也成立，因此 $f(z) = 1$ 对所有 $z \in$ Cube(a, b) 都成立。第 (2) 项由定义25、In-Pool 抽取 10α 个相互独立的点这一事实，以及对 $\mathcal{P}$中的元素应用并集界得出。 □

6.3 Find-Factored-DNFs 过程

现在我们可以介绍 **Find-Factored-DNFs** 算法，算法 6。该算法主要分为五个阶段；本小节将先对这些阶段进行高层次概述，再详细介绍该算法，并简要证明其查询复杂度。

第一阶段“构建池”会抽取一组 $\mathcal{S}$ 个 N^3 赋值，每个赋值都从集合 $f^{-1}(1)$ 中均匀随机抽取，该集合由满足赋值构成。 $\mathcal{S}$ 随后根据对每一对 $\mathcal{G}$ 运行Test-Weak-Equivalence(a, b, MQ(f)) 的结果，将其划分为一组池 a。$b \in \mathcal{S}$. （请注意，在整个 $\mathcal{G}$ 算法过程中， $\mathcal{S}$始终是**Find-Factored-DNFs**的一个划分。）直观地说，如果 f 是一个 s项析取范式，那么这一阶段产生的不同池将包含某种表示方式中不同因子化析取范式的（部分）满足赋值，其中 f 该表示方式将其表示为一族因子化析取范式的析取。

第二阶段“合理性检查”会在第一阶段形成的池严格多于 s 个时停止并拒绝（如上所述，如果 f 实际上是一个 s 项析取范式，这种情况就不应该发生）。

第三阶段“合并池”使用 **Test-Equivalence**以及来自 $f^{-1}(1)$ 的少量新鲜样本，来合并 $\mathcal{G}$中的一些对池。正如 第2.2节 中所讨论的，执行这一步是为了确保 图1 中的性质 (iii) 得以满足。

第四阶段“移除低权重池”使用 In-Pool 以及来自 $f^{-1}(1)$ 的另一小批新鲜样本，从 $\mathcal{G}$ 中找出一个子集，其中的池是“重”池。这里的直观理解是，对于重池，我们可以非常确信，从 SAMP(f) 中进行随机抽样时，算法后续步骤得到的样本会满足该池所涵盖的某个项。我们指出，在阶段 4 中，“足够长”的随机字符串 $\mathbf{str}_{\mathcal{P}}$ 应被视为一个巨大的随机字符串，其中包含足够的随机性，使得本阶段及之后对 **In-Pool**的所有调用都完全确定。特别地，对于每个 $z \in f^{-1}(1)$，$\mathbf{str}_{\mathcal{P}}$ 都包含所需的随机比特，从而能够对每个可能的输入 **In-Pool** 进行确定性模拟 z。（在分析中，使用一个全局随机字符串会很有用；证明 定理 23时将用到这一点。）

最后，在第五阶段，对于每个“heavy”池 $\mathcal{P}_i$ ，我们运行算法 7 和算法 8，构造成员查询* 和SAMP* 模拟器，用于函数 h_i ，该函数对应于该“heavy”池。

请回想一下，这些模拟器随后将在主析取范式测试算法的第 6 行中使用**Test-DNF**。

第 第 6 节 的其余部分将用于证明定理 23。以下简单论断证明了该定理的第 1 项，该项给出了查询复杂度的上界：

算法 6 **Find-Factored-DNFs**

输入： 成员查询(f)，SAMP(f)，一个正整数 s 和 $\varepsilon > 0$。**输出：** 要么拒绝，要么返回一个列表，其中包含至多 s 对预言机模拟器（成员查询$_i^*$，SAMP$_i^*$）。

阶段 1：构建池

1: 抽取一组 $\mathcal{S}$ 个点 $\boldsymbol{z} \sim$ SAMP(f)。2: 按如下方式创建一个顶点集为 $\mathcal{S}$ 的无向图 $\mathcal{G}$。3: **对于**每一对不同的样本 $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{S}$ **执行**4: 运行 **Test-Equivalence**($\boldsymbol{a}, \boldsymbol{b}$, MQ($f$))；如果返回真，则向 $\mathcal{G}$ 添加边 $\{\boldsymbol{a}, \boldsymbol{b}\}$。5: **结束循环**6: 令 $\{\mathcal{P}_i\}$ 为 $\mathcal{G}$ 的连通分量，我们称它们为 $\mathcal{G}$ 的池。

阶段 2：完整性检查

7: 如果 $\mathcal{G}$ 包含多于 s 个池，则停止并拒绝。

阶段 3：合并池

8: 抽取 N^5 个样本 $\boldsymbol{z} \sim$SAMP(f)。9: 对每个 $\boldsymbol{a} \in \mathcal{S}$ 以及每个样本 $\boldsymbol{z}$，运行 **Test-Equivalence**($\boldsymbol{a}, \boldsymbol{z}$, 成员查询($f$))。10: **如果**存在一个样本 $\boldsymbol{z}$，以及来自两个不同池 $\mathcal{P}$, $\mathcal{P}'$ 的 $\boldsymbol{a} \in \mathcal{P}$ 和 $\boldsymbol{a}' \in \mathcal{P}'$，使得**Test-Equivalence** 在 ($\boldsymbol{a}, \boldsymbol{z}$, 成员查询($f$)) 和 ($\boldsymbol{a}', \boldsymbol{z}$, 成员查询($f$)) 上都返回真，**那么** 11: 移除 $\mathcal{P}, \mathcal{P}'$，并用 $\mathcal{P} \cup \mathcal{P}'$ 替换它们。12: 返回本阶段的开头（第 8 行）。

13: **结束 if**

阶段 4：移除低权重池

14: 对每个池 $\mathcal{P}$，抽取一个足够长的随机字符串 str$_\mathcal{P}$，并令计数器$_\mathcal{P} = 0$。15: 抽取 $\omega := O(s/\varepsilon)$ 个 $\boldsymbol{z} \sim$ SAMP(f)。 16: 对于抽取的每个样本 $\boldsymbol{z}$，执行 17: 对于 $\mathcal{G}$ 中的每个池 $\mathcal{P}$，执行 18: 使用来自 str$_\mathcal{P}$($\boldsymbol{z}$) 的随机性运行 In-Pool($\boldsymbol{z}, \mathcal{P}$, 成员查询($f$))。 19: 如果 In-Pool 返回真，则递增计数器$_\mathcal{P}$。 20: 结束 for 21: 如果对 $\boldsymbol{z}$ 的所有 In-Pool 调用都返回假，则停止并拒绝 f。 22: 结束 for 23: 令 Heavy 为满足计数器$_\mathcal{P}$ 至少为 $\frac{\varepsilon}{20s} \cdot \omega$ 的那些 $\mathcal{P}$ 组成的集合。

—

阶段 5：返回预言机

24: 返回包含以下对的列表，其中 $\mathcal{P} \in$ 为 Heavy:

$$\Big(\mathrm{MQ}^*\big(\cdot, \mathcal{P}, \mathrm{MQ}(f), \mathbf{str}_\mathcal{P}\big),\ \mathrm{SAMP}^*\big(\mathcal{P}, \mathrm{MQ}(f), \mathrm{SAMP}(f), \mathbf{str}_\mathcal{P}\big)\Big),$$

其中 成员查询* 和 SAMP* 分别在 算法 7 和 算法 8中给出。

算法 6：**Find-Factored-DNFs**。该算法以以下内容作为输入：成员查询和SAMP 预言机，针对 f、大小参数 s以及 $\varepsilon > 0$。该算法尝试构造从以下对象中抽取样本的“重池”：SAMP。它要么拒绝，要么为每个重池返回一对模拟预言机成员查询*、SAMP*，用于由该池诱导的 f 的一个子函数。

算法 7 成员查询*(x，$\mathcal{P}$，成员查询(f)，字符串)

algo:simulator-for-MQ **输入：** $x \in \{0,1\}^n$，一个池 $\mathcal{P} \subseteq f^{-1}(1)$，对 f 的成员查询访问，以及一个足够长的字符串 str。
输出： 0 或 1。

1: 如果 $f(x) = 0$，则返回 0；否则，使用来自 **In-Pool**(x，$\mathcal{P}$，成员查询(f)) 的随机性运行 In-Pool(x)。2: 如果 **In-Pool** 返回真，则返回 1；否则返回 0 。

算法 7：成员查询*。该算法的输入为 x、一个池 $\mathcal{P}$ 以及一个足够长的随机比特字符串 str。该算法的目标是模拟 成员查询预言机 $h_{\mathcal{P}}$ (如 公式 (3) 中所定义)。

算法 8 SAMP*($\mathcal{P}$，成员查询(f)，SAMP(f)，str)

algo:simulator-for-samp **输入：** 一个池 $\mathcal{P} \subseteq f^{-1}(1)$，对 f 的成员查询和 SAMP 访问权限，以及一个足够长的字符串 str。
输出： $\{0,1\}^n$中的一个点。

1: **重复**100(s/ε)$\alpha = O(s/\varepsilon)$ **次**2: 抽取 $z \sim$SAMP(f)；如果 MQ*($z, \mathcal{P}$, MQ(f), str) $= 1$，则停止并返回 z。3: **结束**4: 返回 0^n。▷ 默认字符串：算法未能获得“良好”的 z。

算法 8：SAMP*。算法以池 $\mathcal{P}$ 和一个由随机比特组成的长字符串 str 作为输入。算法的目标是模拟一个 SAMP 预言机，用于 $h_{\mathcal{P}}$ （定义见 公式 (3)）。

命题 29 （**Find-Factored-DNFs**的复杂度）。Find-Factored-DNFs 对 SAMP(f) 和MQ(f) 进行的 **Find-Factored-DNFs** 调用总次数至多为 poly(s/ε)，并且它返回的每个预言机模拟器的查询复杂度为 poly(s/ε) 。

证明。首先注意，由于只有在 $|\mathcal{G}| \leq s$ 时才会进入阶段 3，因此阶段 3 中的循环最多执行 s 次。所以，合计在第 1 行和第 8 行抽取的样本数不超过 $O(sN^5 + s^2/\varepsilon)$；根据我们对参数的设定，这个数量是 poly($s/\varepsilon$)。其次，成员查询调用总数至多为阶段 1 和阶段 3 中 Test-Equivalence 调用次数的 N^3 倍，而该调用次数至多为 $O(sN^6)$。对于阶段 4，容易验证总查询复杂度为 poly(s/ε)：首先，我们从 SAMP(h) 获取 poly(s/ε) 个样本；接着，外层循环重复 poly(s/ε) 次，内层循环最多重复 s 次，而内层循环的每次迭代都会使用 poly(s/ε) 次成员查询(h) 调用。最后，关于预言机模拟器查询复杂度的断言，通过直接检查算法 7 和算法 8 即可轻松验证。

□

6.4 分析Find-Factored-DNFs中的肯定情形

在本小节中，我们始终假设 $f\colon \{0,1\}^n \to \{0,1\}$ 是一个 s 项析取范式，并固定 $f = T_1 \vee \ldots \vee T_{\leq s}$ 的某种表示 —— 特别是采用与聚类过程中 f 相同的表示

过程（第 4.1 节）。本小节的目标是证明定理 23的第二项，现为方便起见重述如下：

> **第 定理 23 （Find-Factored-DNFs，肯定情形）：**
> 当 f 是一个 s 项析取范式时，Find-Factored-DNFs 以至少 0.9 的概率不会拒绝，并且存在 $s' \le s$ 个布尔函数 $h_1, \ldots, h_{s'}: \{0,1\}^n \to \{0,1\}$ 满足以下条件：(a) 每个 h_i 都是一个 k_i 项析取范式，且 $\sum_{i=1}^{s'} k_i \le s$；(b) 每个 h_i 都是一个 (k_i, μ)-因子化析取范式；(c) 对每个 $i \in [s']$，(成员查询$_i^*$，SAMP$_i^*$) 都是关于 h_i 的预言机的充分实现，其对应于 Test-Factored-DNF$(\cdot,\cdot,k_i,\mu,\varepsilon/(2s))$.

6.4.1 高层思路

给定一个池 $\mathcal{P}$，令

$$h_{\mathcal{P}} := \bigvee_{a \in \mathcal{P}} \bigvee_{T \in \mathsf{terms}(a)} T. \tag{3}$$

换言之，回想定义 27，$h_{\mathcal{P}}$ 是由 terms($\mathcal{P}$) 中所有项组成的析取范式。定理 23 第 2 项中的函数 $h_1, \ldots, h_{s'}$ 将对应于 Find-Factored-DNFs 执行至第 24 行时的函数 $h_{\mathcal{P}}$。有鉴于此，注意当 $h_i = h_{\mathcal{P}}$时，我们有$k_i = |\mathsf{terms}(\mathcal{P})|$。为证明 (a)，我们证明 f 中的每个项至多被 $\mathcal{S}$中的一个赋值覆盖。这又意味着 $h_{\mathcal{P}}$、$\mathcal{P} \in$ Heavy 中的项构成我们对 f的析取范式表示中的（一个子集的）项的划分，因此每个 $h_{\mathcal{P}}$ 中的项互不相同，并且对所有 $h_{\mathcal{P}}$、$\mathcal{P} \in$ Heavy 求和得到的项总数至多为 s。为证明 (b)，我们通过证明 $h_{\mathcal{P}}$ 的所有项都必须属于聚类 Clustering(f, K) 中同一个簇 C，来说明每个 $h_{\mathcal{P}}$ 都是因子化析取范式。由于每个簇 C 都对应一个因子化析取范式（回想第 4.2 节），因此每个 $h_{\mathcal{P}}$ 也都是因子化析取范式。最后，为证明 (c)，我们证明我们的预言机模拟的函数与函数 $h_{\mathcal{P}}$非常接近。

6.4.2 初步定义与结果

在开始详细证明定理 23的第 2 项之前，我们先介绍一些初步定义和基本引理。

引理 30。 每个项 $T \in f$ 至多被一个池覆盖。

证明。固定 f的一个项 T，并假设有 $\boldsymbol{a}$、$\boldsymbol{b} \in \mathcal{S}$，使得 $T(\boldsymbol{a}) = T(\boldsymbol{b}) = 1$。根据引理 26 的第 (1) 部分，在 Find-Factored-DNFs 的第 4 行中，$\boldsymbol{a}$ 与 $\boldsymbol{b}$之间添加了一条边。因此，根据第 6 行，所有满足 $T(\boldsymbol{a}) = 1$ 的 $\boldsymbol{a} \in \mathcal{S}$ 都位于 $\mathcal{G}$的同一个连通分量中。这意味着在第 6 行，T 只能被一个池覆盖 $\mathcal{P}$。第 6 行之后，改变池的唯一方式是在阶段 3 中将它们合并，因此 T 仍然只能被一个池覆盖。 □

定义 31. 令 $w_{\min}$ 为 f 的析取范式表示中任一项的最小宽度。我们用 $w_{\min}$ 和 α 根据 f 中各项的宽度对其进行分类如下：如果 f 的某一项的宽度介于 $w_{\min}$ 和 $w_{\min} + 2\alpha$ 之间，我们称该项为中等项；如果其宽度介于 $w_{\min}$ 和 $w_{\min} + \alpha$ 之间，则称其为窄项。（请注意，长度介于 $w_{\min}$ 和 $w_{\min} + \alpha$ 之间的项既是中等项，也是窄项。）最后，如果 f 的某一项的宽度至少为 $w_{\min} + K$，我们称其为宽项。

引理 32. 以至少 0.99 的概率，f 的每个中等项都至少被一个$a \in \mathcal{S}$覆盖。

证明. 固定 f的一个中等项 T。一次抽取 $\boldsymbol{z} \sim f^{-1}(1)$满足 T的概率至少为

$$\frac{2^{n-(w_{\min}+2\alpha)}}{s \cdot 2^{n-w_{\min}}} \geq \frac{1}{sN^2},$$

回忆 $\alpha = \log N$。由于 $\mathcal{S}$有 N^3个样本，$\mathcal{S}$中没有一次抽取满足 T的概率至多为 $(1 - \frac{1}{sN^2})^{N^3} \leq e^{-N/s}$；由于 $N \gg s$，对中等项 T的所有（至多 s种）可能性应用并集界，即可得出该引理。□

定义 33. 如果集合 $A \subseteq f^{-1}(1)$ 满足以下条件，我们就称其为良好的：

(i) terms(a) 不包含任何宽项；

(ii) 设 $T \in$terms(a) 满足 $|T| \leq w_{\min} + K$。那么对于所有满足 $|T' \setminus T| \geq K$ 的 $T' \in f$，都有 dist(a, T') $\geq K/4$。

引理 34. 以至少 0.99 的概率，集合 $\mathcal{S}$ 在 **Find-Factored-DNFs** 第 1 行抽取，是良好的。

证明。 $\boldsymbol{a} \sim f^{-1}(1)$ 满足固定宽项 T 的概率至多为

$$\frac{|T^{-1}(1)|}{|f^{-1}(1)|} \leq \frac{1}{2^K}.$$

因此，根据并集界，任何 $a \in \mathcal{S}$ 满足任何宽项的概率至多为$s \cdot N^3 2^{-K} < .005$。对于第 (ii) 部分，假设每个 $a \in \mathcal{S}$ 都满足性质 (i)。令 $T \in$ 项(a)，并令 $T' \in f$满足 $|T' \setminus T| \geq K$。根据引理 16，对于均匀随机$\boldsymbol{a} \in f^{-1}(1)$，dist($\boldsymbol{a}, T'$) $\leq K/4$的概率至多为 $e^{-K/8}$。因此，对 $\mathcal{S}$的所有 N^3 个元素，以及满足 T非宽且 $|T' \setminus T| \geq K$的项 T、$T' \in f$的所有 s^2对应用并集界可得，集合 $\mathcal{S}$ 使点 (ii) 不成立的概率至多为 $s^2 N^3 e^{-K/8} < .005$。合并两个 0.005 的失败概率，可得 $\mathcal{S}$ 不满足 good 条件的总概率至多为 .01。□

6.4.3 Find-Factored-DNFs 以高概率接受

以上预备定义和引理介绍完毕后，我们转而证明定理 23 的第 2 项。我们首先在本小节中证明，Find-Factored-DNFs 以高概率接受。为了给 Find-Factored-DNFs 拒绝的概率设定上界，我们分别对它可能拒绝的两个位置设定上界：阶段 2（第 7 行）和阶段 4（第 21 行）。首先，根据引理 30，每个项 $T \in f$ 至多被一个池覆盖；因此 $\mathcal{G}$ 至多包含 s 个池，所以阶段 2 永远不会拒绝。对于阶段 4，第 21 行拒绝的一个必要条件是：第 15 行抽取的样本 $\boldsymbol{z}$ 中有一个，使得 In-Pool($\boldsymbol{z}, \mathcal{P}$, 成员查询($f$)) 对每个 $\mathcal{G}$ 的池 $\mathcal{P}$ 都返回 False。鉴于（以至少 0.99 的概率）每个中等项都被一个池覆盖（引理 32），再回顾引理 28，这种情况发生的一个必要条件是：对于某个并非中等的 f 中的 T，$T(\boldsymbol{z}) = 1$。然而，$f^{-1}(1)$ 中满足某个非中等项的点所占比例至多为

$$\frac{s \cdot 2^{n-w_{\min}-2\alpha}}{2^{n-w_{\min}}} = \frac{s}{N^2}.$$

˜因此，对第 15 行中的 $O(s/\varepsilon)$ 多次抽样应用并集界可知，**Find-Factored- 析取范式**拒绝的概率至多为.02。

接下来需要证明条目 2 的 (a)、(b) 和 (c) 部分；我们将在以下各小节中完成证明。

6.4.4 断言 2（a）的证明

我们的目标是证明，如果 **Find-Factored-DNFs** 到达阶段 5，那么必然有$\sum_{\mathcal{P}\in\text{Heavy}} |\text{项}(\mathcal{P})| \le s$。根据 引理 30，每个项至多被一个池覆盖；又因为项数至多为 s，所以所有池中 $|\text{项}(\mathcal{P})|$ 的总和，在 $\mathcal{P}$ 阶段 1 创建的所有池中至多为 s。因此，由于阶段 3 会合并池，而阶段 4 只处理部分池，这一性质在算法运行过程中始终保持不变。

6.4.5 断言 2（b）的证明

为证明这一点，我们将说明，阶段 3 之后，对于每个池 $\mathcal{P}$，都有 terms($\mathcal{P}$) $\subseteq C$，其中某个簇 C 属于 Clustering(f, K)。这足以证明断言 2（b），因为每个 $h_{\mathcal{P}}$ 都由 |个 terms($\mathcal{P}$)|项组成；根据断言 19，由同一簇 C （属于 Clustering(f, K)）中的 k 项组成的析取范式是一个 (k, μ)-因子化析取范式。

为此，假设 $\mathcal{S}$ 满足引理 32 和 34 中的所有先前性质；特别是， $\mathcal{S}$ 是良好的，并且每个中等项至少被一个 $a \in \mathcal{S}$ 覆盖。我们首先证明，对于每个 $a \in \mathcal{S}$，都存在一个簇 C，使得 terms(a) $\subseteq C$。令 $a \in \mathcal{S}$，并假设存在 $T, T' \in$ 个 terms(a)，使得 T, T' 位于不同的簇中。由于 $\mathcal{S}$是良好的，根据定义 33 的性质（i），有 $|T|, |T'| \le w_{\min} + K$；因此，根据推论 15，有 $|T' \setminus T| > K$。接着，根据定义 33 的性质（ii），有 dist(a, T') $\ge K/4$。但这与以下事实矛盾：对于每个满足 a 的 T'，都有 dist(a, T') $= 0$。因此，terms(a) 中的所有项都位于同一个簇中。

我们称点 $\boldsymbol{a} \in \mathcal{S}$ 位于簇 C 中，如果 terms(a) 中的所有项都位于 C 中。接下来，我们论证，在阶段 1 结束时，对于每个池 $\mathcal{P}$， $\mathcal{P}$ 中的所有点都位于同一个簇中。下一个引理表明，以高概率我们绝不会在位于不同簇中的点之间添加边：

引理 35。 以至少 0.99 的概率，对于任意 a， $b \in \mathcal{S}$ 这两个点位于聚类(f, K) 的不同簇中，**Test-Equivalence**$(a, b,$ 成员查询$(f))$ 返回 False，并且 (a, b) 不是 $\mathcal{G}$ 中的一条边。

证明。令 C 为聚类(f, K) 中满足 terms(a) $\subseteq C$ 的簇，并令 C' 为满足 terms(b) $\subseteq C'$ 的簇。给定 $\mathcal{S}$ 良好，且 a、 $b \in \mathcal{S}$ 位于不同簇中，则对于 f 的任意项 T，要么 dist(a, T) $\ge K/4$，要么 dist(b, T) $\ge K/4$。（这是因为，如上面两段所述，当 $T \notin C$ 时前一种情况成立，而当 $T \notin C'$ 时后一种情况成立。）根据引理 26 的第 (3) 项，并对属于 $\mathcal{S}$ 的 a、 b 的至多 N^6 对应用并集界，即可得到该引理。 □

从现在起，我们固定 $\mathcal{S}$ 和 $\mathcal{G}$（因此也固定阶段 1 结束时的 $\mathcal{G}$ 池），并假设它们满足目前为止的所有条件；特别地， $\mathcal{S}$ 是良好的，并且引理 35 的结论成立。在这些假设下，到阶段 1 结束时，对于每个池 $\mathcal{P}$， $\mathcal{P}$ 中的所有点都位于 Clustering(f, K) 中的同一个簇 $C_{\mathcal{P}}$ 内，因此 terms($\mathcal{P}$) $\subseteq C$。接下来需要证明的是，在阶段 3 合并池时，对于每个池 $\mathcal{P}$， terms($\mathcal{P}$) 仍然是 Clustering(f, K) 中某个簇 C 的子集。

我们先给出一个与 引理 35类似的引理：

引理 36。 以至少 0.99 的概率，阶段 3 中采样得到的所有点的集合 $\boldsymbol{z} \sim \text{SAMP}(f)$ 是一个良好集合，并且阶段 3 中对 **Test-Equivalence**$(a, z, \text{MQ}(f))$ 的每次调用在 a, z位于不同簇中时都返回 False。

证明。这可由与以下证明类似的论证得出：引理 34和35。 □

考虑阶段 3 的一次迭代：到达第 8 行时，对于每个池 $\mathcal{P}$，其中所有点都位于同一个簇中。假设对于两个不同的池 $\boldsymbol{a} \in \mathcal{P}$、 $\boldsymbol{a}' \in \mathcal{P}'$ ， $\mathcal{P}'$ 满足以下条件：

在第 10 行，Test-Equivalence($\boldsymbol{a}, \boldsymbol{z}$, MQ($f$)) 和 Test-Equivalence($\boldsymbol{a}', \boldsymbol{z}$, MQ($f$)) 均返回真。根据引理 36，$\boldsymbol{a}$、$\boldsymbol{a}'$ 和 $\boldsymbol{z}$ 必定都位于同一个簇 C 中。根据假设，$\mathcal{P}$ 和 $\mathcal{P}'$ 中的所有点也都位于 C 中。因此，创建池 $\mathcal{P} \cup \mathcal{P}'$ 时，这个新池中的每个点也都位于簇 C 中。于是回到第 8 行时，关于这些池的假设仍然成立。因此，当算法到达阶段 4 时，对于 $\mathcal{G}$ 的每个池 $\mathcal{P}$，必有 terms($\mathcal{P}$) $\subseteq C$，其中 C 是 Clustering(f, K) 中的某个簇。由此，根据断言 19，$h_{\mathcal{P}}$ 是一个 ($|\text{terms}(\mathcal{P})|, \mu$)-因子化析取范式，因此条目 2 的 (b) 部分以至少 0.97 的概率成立。

6.4.6 条目 2（部分 c）的证明

条目 2 中这部分需要最复杂的论证。给定一个池 $\mathcal{P}$ 和一个不属于 terms($\mathcal{P}$) 的项 T，如果至少有 $(1/N^2)$ 比例的点 $b \in T^{-1}(1)$ 满足：对于某个 $a \in \mathcal{P}$，Cube(a, b) 是稠密的，那么我们称 T被 $\mathcal{P}$ 吸引。（回顾一下，如果 Cube(a, b) 中至少有 $(1/N^2)$ 比例的点属于 $f^{-1}(1)$，那么它就是稠密的。）

我们先给出两个预备引理，它们说明了 $\mathcal{S}$ 和 $\mathcal{G}$的一些有用性质：

引理 37。 在条件 $\mathcal{S}$ 是良好的情况下，以至少 0.97 的概率，阶段 3 结束时以下情况不会发生：存在一个池 $\mathcal{P}$ 和一个属于 f 的中等项 $T \notin$terms($\mathcal{P}$)，使得T 被 $\mathcal{P}$吸引。

证明。 我们首先回顾一下，根据 引理 34，$\mathcal{S}$ 以至少 0.99 的概率是良好的；在接下来的论证中，我们假设 $\mathcal{S}$是良好的。根据 引理 32，以至少 $1 - 0.01 \cdot \frac{100}{99}$ 的概率，在阶段 3 开始时，每个中等一项都由唯一一个池覆盖（概率界如上所述，是因为我们已假设 $\mathcal{S}$ 是良好的，这一事件发生的概率至少为 0.99）。经过任意轮次的合并池后，这一性质仍然成立，因此在下面余下的论证中，我们假设该性质始终成立。

现在考虑以下事件：执行第 i 轮第 8–12 行之初，$\mathcal{P}$ 是一个当前的池，它吸引了某个中等项 $T \notin$ 项（$\mathcal{P}$），但在这一轮中，$\mathcal{P}$ 没有与覆盖 T 的池合并。我们证明这种情况发生的概率至多为 $e^{-\Omega(N/s)}$；由此，对 i，的所有（至多 s 种）选择、$\mathcal{P}$ 的所有（至多 s 种）选择以及 T 的所有（至多 s 种）选择应用并集界，即可得出该引理。

为此，由于每个中等项都被覆盖，令 $\mathcal{P}'$ 为第 i 轮开始时含有 $T \in$项（$\mathcal{P}'$）的池，并令 b 为 $\mathcal{P}'$ 中含有 $T \in$ 项（b）的任意一个点。已知 T 是中等项且被 $\mathcal{P}$ 吸引，则满足以下条件的点 $b \in T^{-1}(1)$ 的数量至少为：对于某个 $a \in \mathcal{P}$，Cube(a, b) 是稠密的

$$\frac{1}{N^2} \cdot 2^{n - w_{\min} - 2\alpha},$$

这至少是 $1/(sN^4)$-fraction of $|f^{-1}(1)|$。鉴于我们在这一轮中抽取了 N^5samples $\boldsymbol{z}$，至少有一个 $\boldsymbol{z}$ 满足 $T(\boldsymbol{z}) = 1$，并且对于某个 $a \in \mathcal{P}$，Cube($a, \boldsymbol{z}$) 是稠密的，其概率至少为 $1 - e^{-\Omega(N/s)}$。对于这个 $\boldsymbol{z}$，根据引理 26，Test-Equivalence($b, \boldsymbol{z}$, 成员查询(f)) 始终返回 True，而 Test-Equivalence($a, \boldsymbol{z}$, 成员查询(f)) 以至少 $1 - 2^{-\Omega(N)}$ 的概率返回 True。因此，$\mathcal{P}$ 在这一轮中不与 $\mathcal{P}'$ 合并的概率至多为 $e^{-\Omega(N/s)}$。

□

引理 38. 固定 $\mathcal{S}$，$\mathcal{G}$ 使得本小节前面各个引理的所有结论均成立。固定一个池 $\mathcal{P}$，其中包含 $\mathcal{G}$，位于 **Find-Factored-DNFs**的阶段 4 开始时。对于任意窄$T \in$项($\mathcal{P}$) 以及任意项 $T' \notin$项($\mathcal{P}$)，有 $|T' \setminus T| \geq \alpha$。

证明。若 T' 不是中等的，鉴于 $|T' \setminus T| \geq |T'| - |T| \geq \alpha$，结论显然成立。当 T' 是中等的时，我们声称有 $|T \setminus T'| \geq 2\alpha$；否则， T' 会被 $\mathcal{P}$ 吸引，这与引理 37 的结论矛盾。为说明这一点，注意对于任意满足 $T(a) = 1$ 的 a 以及满足 $T'(b) = 1$ 的 b，假设 $|T \setminus T'| < 2\alpha$ 意味着 $\mathsf{Cube}(a, b)$ 中满足 T 的点所占比例至少为 $1/2^{2\alpha} = 1/N^2$。因此，令 $a \in$ 项$(\mathcal{P})$ 为一个满足 $T(a) = 1$ 的点，则每个 $b \in T'^{-1}(1)$ 都满足 $\mathsf{Cube}(a, b)$ 是稠密的，所以 T' 确实会被 $\mathcal{P}$ 吸引。

现在，利用 $|T \setminus T'| \geq 2\alpha$，并且因为 T 是窄的，我们有

$$|T' \setminus T| = |T'| - |T \cap T'| = |T'| - (|T| - |T \setminus T'|) \geq \alpha,$$

引理得证。 □

接下来，我们假设前面引理中的所有结论均成立。首先注意，在阶段 4 开始时抽取 $\mathrm{str}_{\mathcal{P}}$ 后，$\mathsf{MQ}^*(\cdot, \mathcal{P}, \mathsf{MQ}(f), \mathrm{str}_{\mathcal{P}})$（简记为 $\mathsf{MQ}^*_{\mathcal{P}}$，类似地，将 $\mathsf{SAMP}^*(\mathcal{P}, \mathsf{MQ}(f), \mathsf{SAMP}(f), \mathrm{str}_{\mathcal{P}})$ 简记为 $\mathsf{SAMP}^*_{\mathcal{P}}$）就成为一个确定性算法，并对应于 $\{0,1\}^n$ 上的一个布尔函数。我们记作$g_{\mathcal{P}}$：$\{0,1\}^n \to \{0,1\}$（在抽取 $\mathrm{str}_{\mathcal{P}}$ 后），表示 $\mathsf{MQ}^*_{\mathcal{P}}$ 所对应的函数。根据算法 7，显然我们总有

$$h_{\mathcal{P}}^{-1}(1) \subseteq g_{\mathcal{P}}^{-1}(1) \subseteq f^{-1}(1).$$

特别地，$\mathsf{MQ}^*_{\mathcal{P}}$ 恰好是 $g_{\mathcal{P}}$ 的成员关系预言机，因此当在满足 $h_{\mathcal{P}}(x) = 1$ 的点上进行查询时，它总是回答 1。

观察 39. 成员查询$^*_{\mathcal{P}}$ 在 类型 2 查询中始终回答 1 $h_{\mathcal{P}}(z) = 1$。

下一个引理表明，对于$\mathbf{str}_{\mathcal{P}}$，以高概率有 $g_{\mathcal{P}}^{-1}(1)$ 与 $h_{\mathcal{P}}^{-1}(1)$ 非常接近：

引理 40. 对于$\mathbf{str}_{\mathcal{P}}$，以至少.99 的概率，对每个池 $\mathcal{P}$，以下不等式均成立：

$$\frac{|g_{\mathcal{P}}^{-1}(1) \setminus h_{\mathcal{P}}^{-1}(1)|}{|f^{-1}(1)|} \leq O\left(\frac{s^2}{N}\right).$$

证明。 我们考虑一个固定的池 $\mathcal{P}$，并在论证最后对至多 s 个池应用并集界。

回顾 公式 (3) 和 算法 7，对于一个点 $x \in \{0,1\}^n$ 满足 $g_{\mathcal{P}}(x) = 1$ 但 $h_{\mathcal{P}}(x) = 0$，必定有 $T(x) = 1$，其中某一项 T 属于 f，且 $T \notin$ 项$(\mathcal{P})$[7]。因此我们有

$$|g_{\mathcal{P}}^{-1}(1) \setminus h_{\mathcal{P}}^{-1}(1)| \leq \sum_{T \notin \mathsf{terms}(\mathcal{P})} \left|g_{\mathcal{P}}^{-1}(1) \cap T^{-1}(1)\right|. \tag{4}$$

对非中等的项 $T \notin \mathsf{terms}(\mathcal{P})$ 求和，可以轻易地将其上界估计为

$$\frac{s}{N^2} \cdot |f^{-1}(1)|.$$

给定 $|T^{-1}(1)| \leq |f^{-1}(1)|/N^2$，对于每个非中等项。

[7]由于 $g_{\mathcal{P}}(x) = 1$，因此必有 $f(x) = 1$，所以对于某个属于 f 的项 T， $T(x) = 1$。但由于 $h_{\mathcal{P}}(x) = 0$，必有 $T \notin \mathsf{terms}(\mathcal{P})$。

另一方面，考虑任意一个中等项 $T \notin$项$(\mathcal{P})$。根据 引理 37，我们有 T不会被 $\mathcal{P}$吸引，因此 $b \in T^{-1}$中的点至多有 $1/N^2\,(1)$ 满足 立方体$(a,\ b)$在 f 中稠密 $a \in \mathcal{P}$。因此，利用 引理 28，我们有 $|g_{\mathcal{P}}^{-1}\,(1) \cap T^{-1}(1)|$ 的期望大小至多为

$$|T^{-1}(1)| \cdot \left(\frac{1}{N^2} + \left(1 - \frac{1}{N^2}\right) \frac{|\mathcal{P}|}{N^{20\alpha}} \right).$$

利用 $|\mathcal{P}| \leq N^3$ 并对所有中等项 $T \notin$ 项$(\mathcal{P})$求和，中等项$T \notin$项$(\mathcal{P})$的总和至多为

$$\frac{2}{N^2} \cdot \sum_{\text{medium terms } T \notin \mathsf{terms}(\mathcal{P})} |T^{-1}(1)| \leq \frac{2s}{N^2} \cdot |f^{-1}(1)|.$$

因此，公式（4）右侧的期望至多为

$$O(s/N^2) \cdot |f^{-1}(1)|.$$

根据马尔可夫不等式，我们有 $|g_{\mathcal{P}}^{-1}(1) \setminus h_{\mathcal{P}}^{-1}(1)| > O(s^2/N) \cdot |f^{-1}(1)|$的概率至多为 $1/(100s)$。对至多 s 个池应用并集界即可得出该引理。 □

下一个引理表明，以高概率，对于阶段 4 结束时处于 Heavy 中的每个池 $\mathcal{P}$，$|g_{\mathcal{P}}^{-1}\,(1)|$ 都相当大。

引理 41。 以至少 0.99 的概率，阶段 4 结束时每个处于 Heavy 中的池 $\mathcal{P} \in$都满足

$$\mathop{\mathbf{Pr}}_{\boldsymbol{z} \sim f^{-1}(1)} \left[g_{\mathcal{P}}(\boldsymbol{z}) = 1 \right] \geq \varepsilon/(30s). \tag{5}$$

证明。 设 $\mathcal{P}$ 是一个池，其“真实频率”$\mathbf{Pr}_{\boldsymbol{z} \sim f^{-1}(1)} \left[g_{\mathcal{P}}(\boldsymbol{z}) = 1 \right]$ 至多为 $\varepsilon/(30s)$。根据标准的切尔诺夫乘法界，经过 $O((s/\varepsilon)\log(s/\varepsilon))$个样本后，经验频率至少为 $\varepsilon/(20s)$的概率至多为 $1/(100s)$。由并集界即可得出该引理。

□

我们假设前面所有引理的结论都成立，这一事件发生的概率至少为 0.9。因此，**Find-Factored-DNFs** 进入阶段 5 时拥有一个包含至多 s 个池的Heavy集合，并为每个 $\mathcal{P} \in$ Heavy返回预言机模拟器 $(\mathrm{MQ}^*_{\mathcal{P}}、\mathrm{SAMP}^*_{\mathcal{P}})$。由于**Test-Factored-DNF**$(\mathrm{SAMP}(h_{\mathcal{P}})、\mathrm{MQ}(h_{\mathcal{P}}))$ 以至少 0.9 的概率接受，

要完成第 2 部分第 (c) 项的证明，只需证明

$$\begin{aligned} &\mathbf{Pr}\left[\textbf{Test-Factored-DNF}\left(\mathrm{SAMP}^*_{\mathcal{P}_i}, \mathrm{MQ}^*_{\mathcal{P}_i}\right) \text{ accepts}\right] \\ &\quad \geq \mathbf{Pr}\left[\textbf{Test-Factored-DNF}\left(\mathrm{SAMP}(h_{\mathcal{P}_i}), \mathrm{MQ}(h_{\mathcal{P}_i})\right) \text{ accepts}\right] - 0.1 \end{aligned} \tag{6}$$

对于每个池 $\mathcal{P}_i \in$ Heavy，在本句中，为了排版方便，我们省略了 Test-Factored-DNF 中的参数 $k_i \leq s$、μ和 $\varepsilon/(2s)$，其中 $k_i = |$项$(\mathcal{P}_i)|$。我们将把公式（6）的证明分成两个步骤。首先，我们将 $\mathrm{SAMP}^*_{\mathcal{P}}$ 替换为 $\mathrm{SAMP}(h_{\mathcal{P}})$，然后将 $\mathrm{MQ}^*_{\mathcal{P}}$ 替换为 $\mathrm{MQ}(h_{\mathcal{P}})$；在这两个步骤中，我们分别致力于界定 Test-Factored-DNF 接受概率的变化。

回顾一下，根据 $g_{\mathcal{P}}$ 的定义，$\mathrm{MQ}^*_{\mathcal{P}}$ 正是 $g_{\mathcal{P}}$ 的成员关系预言机。另一方面，我们将在下文证明，$\mathrm{SAMP}^*_{\mathcal{P}}$ 与 $g_{\mathcal{P}}$ 的采样预言机非常接近，而后者又与针对 $h_{\mathcal{P}}$ 的采样预言机 $\mathrm{SAMP}(h_{\mathcal{P}})$ 非常接近：

引理 42。 以 $1 - O(s^3/(\varepsilon N))$ 的概率，$\mathrm{SAMP}^*_{\mathcal{P}}$ 从 $h^{-1}_{\mathcal{P}}(1)$ 中均匀抽取一个样本。

证明。观察算法 8可知，$\mathrm{SAMP}^*_{\mathcal{P}}$ 会返回它收到的第一个样本$\boldsymbol{z} \in g^{-1}_{\mathcal{P}}(1)$；如果不存在这样的样本，它就会返回默认值 0^n。由于 $\mathcal{P} \in$ Heavy 根据引理 41，$\mathrm{SAMP}^*_{\mathcal{P}}$ 未能抽取到“好”的 $\boldsymbol{z}$ 并默认返回 0^n 的概率至多为

$$(1 - \varepsilon/(30s))^{100(s/\varepsilon)\alpha} \leq 1/N.$$

另一方面，对算法 $8s/\varepsilon$次重复α 应用并集界，并结合算法 8 以及引理 40，该算法收到点 $\boldsymbol{z} \in \left(g^{-1}_{\mathcal{P}}(1) \setminus h^{-1}_{\mathcal{P}}(1)\right)$ 的概率至多为

$$O\big((s/\varepsilon)\alpha\big) \cdot O(s^2/N) = O\left(\frac{s^3}{\varepsilon N}\right).$$

当这两个低概率事件都未发生时，$\mathrm{SAMP}^*_{\mathcal{P}}$ 必定返回从 $h^{-1}_{\mathcal{P}}(1)$ 中均匀抽取的结果。

□

现在回想一下，Q 是Test-Factored-DNF从其获得的采样预言机中抽取的样本数的上界。因此，我们有

$$\begin{aligned}
&\mathbf{Pr}\left[\text{Test-Factored-DNF}\left(\mathrm{SAMP}^*_{\mathcal{P}}, \mathrm{MQ}^*_{\mathcal{P}}\right) \text{ accepts}\right] \\
&\geq \left(1 - O\left(\frac{s^5 Q}{\varepsilon^3 N}\right)\right) \cdot \mathbf{Pr}\left[\text{Test-Factored-DNF}\left(\mathrm{SAMP}(h_{\mathcal{P}}), \mathrm{MQ}^*_{\mathcal{P}}\right) \text{ accepts}\right] \\
&\geq \mathbf{Pr}\left[\text{Test-Factored-DNF}\left(\mathrm{SAMP}(h_{\mathcal{P}}), \mathrm{MQ}^*_{\mathcal{P}}\right) \text{ accepts}\right] - 0.01, \qquad (7)
\end{aligned}$$

采用 N 使得 $N \gg Q(s/\varepsilon)^5$ 的选择。

我们还需要将 成员查询$^*_{\mathcal{P}}$ 替换为 成员查询$(h_{\mathcal{P}})$。为此，我们需要处理 类型 1 查询，下面的新记号将会很有帮助：

定义 43。 给定 a，$b \in \{0,1\}^n$ 和 $\mathsf{X} \subseteq [n]$，我们将 $\mathsf{Cube}_{\mathsf{X}}(a, b)$ 定义为满足以下条件的点集合 $y \in \{0,1\}^n$：对所有 $i \notin \mathsf{X}$，$y_i \in \{a_i, b_i\}$。

（注意，与 $\mathsf{Cube}(a, b)$ 的原始定义不同，这里对 y_i for $i \in \mathsf{X}$ 没有任何条件。）

接下来，受 Test-Factored-DNF 在 第 6.1 节中的工作原理总结的启发，我们证明以下引理。该引理大致表明，以高概率（关于 $\boldsymbol{y}$，$\boldsymbol{w}$ 以及 $[n]$的随机划分），当算法进行一次 类型 1 查询时，无论位 $i \in \mathsf{X}_\ell$ 如何设置，对于被查询的点 z 我们并没有 $h_i(z) = 0$，而是 $f(z) = 1$。

引理 44。 设X_1、.....、X_τ是 $[n]$ 的一个随机 τ路划分，其中 $\tau = \Theta(\mu^2)$。设 $\boldsymbol{y} \sim h^{-1}_{\mathcal{P}}(1)$和 $\boldsymbol{w} \sim \{0,1\}^n$。那么，以至少 $1 - e^{-\Omega(\alpha)}$的概率，对于每个块X_ℓ，都有

$$\mathsf{Cube}_{\mathsf{X}_\ell}(\boldsymbol{y}, \boldsymbol{w}) \cap \left(f^{-1}(1) \setminus h^{-1}_{\mathcal{P}}(1)\right) = \emptyset.$$

*Pr*证明。 固定一个 $\ell \in [\tau]$ 和一个项 $T' \notin$ 项$(\mathcal{P})$。我们将概率 o 上界为 f

$$\mathsf{Cube}_{\mathsf{X}_\ell}(\boldsymbol{y}, \boldsymbol{w}) \cap T'^{-1}(1) \neq \emptyset$$

通过 $e^{-\Omega(\alpha)}$，然后对 ℓ 和 T' 至多 τs 种选择应用并集界。首先，我们证明，以至少 $1 - O(s^2/(\varepsilon N)) - e^{-\alpha/8}$ 的概率，抽取一次 $\boldsymbol{y} \sim h^{-1}_{\mathcal{P}}(1)$ 会满足以下两个条件：

1. $\boldsymbol{y}$ 在 terms($\mathcal{P}$) 中满足一个窄项 T；

2. dist($\boldsymbol{y}, T'$) $\geq \alpha/4$。

由于 $\mathcal{P} \in$ Heavy，由 引理 41 可知 $\mathbf{Pr}_{z\sim f^{-1}(1)}[g_{\mathcal{P}}(\boldsymbol{z}) = 1] \geq \varepsilon/30s$。另一方面，引理 40给出

$$\Pr_{\boldsymbol{z}\sim f^{-1}(1)}[g_{\mathcal{P}}(\boldsymbol{z}) = 1 \wedge h_{\mathcal{P}}(\boldsymbol{z}) = 0] \leq O(s^2/N).$$

由于 $O(s^2/N) \ll \varepsilon/(60s)$，结合上面的两个等式可得

$$\Pr_{z\sim f^{-1}(1)}[h_{\mathcal{P}}(\boldsymbol{z}) = 1] \geq \varepsilon/60s.$$

现在，对于 $\boldsymbol{z} \sim f^{-1}(1)$，$\boldsymbol{z}$ 不满足 f 的任何窄项的概率至多为$s \times 2^{-\alpha} = s/N$。结合前面的不等式，可得

$$\Pr_{\boldsymbol{z}\sim f^{-1}(1)}[T(\boldsymbol{z}) = 1 \text{ for some narrow } T \in \mathsf{terms}(\mathcal{P})] \geq \Pr_{z\sim f^{-1}(1)}[h_{\mathcal{P}}(\boldsymbol{z}) = 1] - s/N.$$

现在结合前面的两个不等式，可得

$$\Pr_{\boldsymbol{y}\sim h_{\mathcal{P}}^{-1}(1)}[T(\boldsymbol{y}) = 1 \text{ for some narrow } T \in \mathsf{terms}(\mathcal{P})] \geq 1 - 60s^2/(\varepsilon N),$$

正如上面第 (1.) 项所需。

因此，假设 $\boldsymbol{y} \sim h_{\mathcal{P}}^{-1}(1)$ 满足 T中的某个窄项 $h_{\mathcal{P}}$。根据 引理 38，我们有 $|T' \setminus T| \geq \alpha$。因此，利用 引理 16，以至少 $1 - e^{-\alpha/8}$ 的概率，有 $\mathrm{dist}(\boldsymbol{y}, T') \geq \alpha/4$，从而得到 (2.)

根据上面的条件 (2.)，由一个简单的 Chernoff 界（现在针对 $\mathbf{X}_1, \ldots, \mathbf{X}_\tau$ 划分中的随机性），取 τ充分大[8]，我们有至少以$1 - e^{-\alpha/8}$ 的概率，以下结论成立：

3. 随机划分的部分 $\mathbf{X}_\ell$ 中，贡献于 dist($\boldsymbol{y}, T'$) 的索引不超过总数的一半。将这些索引记为 Y。

当 (1.)、(2.) 和 (3.) 全部成立时，我们有

$$\mathsf{Cube}_{\mathbf{X}_\ell}(\boldsymbol{y}, \boldsymbol{w}) \cap T'^{-1}(1) = \emptyset$$

只要对于某个文字 $\ell \in T'$，$\ell(\boldsymbol{w}) = 0$ 且 $\ell(\boldsymbol{y}) = 0$，并且 ℓ 对应的变量位于 $\mathbf{X}_\ell$ 之外。因此，在抽取 $\boldsymbol{w}$ 时，这种情况发生的概率至少为 $1 - 2^{-\alpha/8}$。

综合这些界，$\mathsf{Cube}_{\mathbf{X}_\ell}(\boldsymbol{y}, \boldsymbol{w}) \cap T'^{-1}(1) \neq \emptyset$的概率至多为 $O(s^2/(\varepsilon N)) + 2e^{-\alpha/8} \leq e^{-\Omega(\alpha)}$。

□

现在我们解释引理 44 与类型 1 查询之间的联系。回顾类型 1 查询的工作方式：算法首先将 $[n]$ 随机划分为若干块 X1，...，X$_\tau$，并抽取一个至多包含 Q 个字符串的集合 $\boldsymbol{y} \sim$ SAMP(h)，以及一个至多包含 Q 个字符串的集合 $\boldsymbol{w} \sim \{0,1\}^n$。随后，一个类型 1 查询对应于一个点 z，其中对于某个 $\boldsymbol{y} \in$ Y 和 $\boldsymbol{w} \in$ W，有 $z \in \mathsf{Cube}_{\mathbf{X}_\ell}(\boldsymbol{y}, \boldsymbol{w})$。

[8]由于 τ 取决于 ε，正如我们先前所讨论的，这不失一般性。

因此，如果 $\mathsf{Cube}^{\mathsf{x}}_{\ell}(\boldsymbol{y},\boldsymbol{w}) \cap (f^{-1}(1) \setminus h_{\mathcal{P}}^{-1}(1)) = \emptyset$，我们可以得到以下结论：要么 $f(z) = h_{\mathcal{P}}(z) = 0$，要么 $h_{\mathcal{P}}(z) = 1$。在第一种情况下，$\mathrm{MQ}^*_{\mathcal{P}}$ 始终会回答 0；在后一种情况下，$\mathrm{MQ}^*_{\mathcal{P}}$ 始终会回答 1。特别地，这个类型 1 查询将由 $\mathrm{MQ}^*_{\mathcal{P}}$正确回答。

因此，根据引理 44，并对所有 τQ^2 对 $(\boldsymbol{y},\boldsymbol{w}) \in [\tau] \times \mathsf{Y} \times \mathsf{W}$ 应用并集界，每个类型 1 查询都能以至少 $1 - \tau Q^2 e^{-\Omega(\alpha)}$ 的概率得到正确回答。我们已经论证过，当 $h_{\mathcal{P}}(z) = 1$ 时，类型 2 查询总是得到 1 的回答（见观察 39）。由于 Test-Factored-DNF 在类型 2 查询得到 0 的回答时会立即拒绝，因此，用 $\mathrm{MQ}^*_{\mathcal{P}}$ 代替 $\mathrm{MQ}_{\mathcal{P}}$ 来回答类型 2 查询，不会降低 Test-Factored-DNF 接受的概率。

$$\begin{aligned}&\mathbf{Pr}\left[\textbf{Test-Factored-DNF}\left(\mathrm{SAMP}(h_{\mathcal{P}}), \mathrm{MQ}^*_{\mathcal{P}}\right) \text{ accepts}\right] \\ &\quad \geq \mathbf{Pr}\left[\textbf{Test-Factored-DNF}\left(\mathrm{SAMP}(h_{\mathcal{P}}), \mathrm{MQ}(\mathcal{P})\right) \text{ accepts}\right] - \tau Q^2 \cdot e^{-\Omega(\alpha)}. \end{aligned} \tag{8}$$

通过令 N（即 2^α）成为 Q 的一个充分大的多项式，

我们可以令 $\tau Q^2 \cdot e^{-\Omega(\alpha)} \leq 0.01$。公式（6）于是可由公式（7）和（8）推出。这就完成了定理 23中条目 2（部分 c）的证明，也就完成了条目 2（**Find-Factored-DNFs**的肯定情形）的证明。

6.5 分析Find-Factored-DNFs否情形

在本小节中，我们令 f为任意一个函数 $f\colon \{0,1\}^n \to \{0,1\}$ 该函数 ε与s项析取范式的相对距离至少为s，并证明定理 23的第 3 项。为方便起见，我们在此重述第 3 项：

定理 23 的第 3 项 (Find-Factored-DNFs，否情形)：

当 f 为 ε与任意 s项析取范式的相对距离至少为时，以至少 0.9 的概率，**Find-Factored-DNFs** 要么拒绝，要么存在 $s' \leq s$ 个函数 $h_1, \ldots\ldots, h_{s'} : \{0,1\}^n \to \{0,1\}$ 使得 (a′) 对于每个 $i \in [s']$，我们都有 $h_i^{-1}(1) \subseteq f^{-1}(1)$，并且

$$\frac{|f^{-1}(1) \setminus (\cup_i h_i^{-1}(1))|}{|f^{-1}(1)|} \leq \frac{\varepsilon}{2}; \tag{1}$$

并且 (b′) 对于每个 $i \in [s']$，以及每个 $r \in [s]$，$(\mathrm{MQ}^*_i, \mathrm{SAMP}^*_i)$ 都是一个合适的实现 用于 h_i相对于**Test-Factored-DNF**$(\cdot, \cdot, r, \mu, \varepsilon/(2s))$的预言机实现。

首先我们注意到，抽取**字符串**$_{\mathcal{P}}$后，$\mathrm{MQ}^*(\cdot, \mathcal{P}, \mathrm{MQ}(f),$ **字符串**$_{\mathcal{P}})$ 会变成一个确定性函数，我们将其记为 $h_{\mathcal{P}}\colon \{0,1\}^n \to \{0,1\}$。（因此，在此处考虑的否情形中，$\mathrm{MQ}^*(\cdot, \mathcal{P}, \mathrm{MQ}(f),$ **字符串**$_{\mathcal{P}})$ 恰好是 $h_{\mathcal{P}}$ 的成员关系预言机 $\mathrm{MQ}(h_{\mathcal{P}})$。）显然（根据 MQ^* 的工作方式），对于每个池，我们都有 $h_{\mathcal{P}}^{-1}(1) \subseteq f^{-1}(1)$。

下面的引理表明，除非 **Find-Factored-DNFs** 的并集几乎覆盖了阶段 5（未拒绝）的所有 $h_{\mathcal{P}}^{-1}(1)$、$\mathcal{P} \in \mathsf{Heavy}$ 和 $f^{-1}(1)$，否则

引理 45. 概率至少为 0.9，**Find-Factored-DNFs** 要么拒绝，要么进入阶段 5，且其中的Heavy 满足以下两项条件

$$\frac{|f^{-1}(1) \setminus (\cup_{\mathcal{P} \in \mathsf{Heavy}} h_{\mathcal{P}}^{-1}(1))|}{|f^{-1}(1)|} \leq \frac{\varepsilon}{2},$$

此外，每个 $\mathcal{P} \in$Heavy 都满足

$$\mathbf{Pr}_{\boldsymbol{z}\sim f^{-1}(1)}\left[h_{\mathcal{P}}(\boldsymbol{z})=1\right] \geq \frac{\varepsilon}{30s}. \tag{9}$$

Pr唉。 我们假设阶段 4 开始时的池集合满足

$$\frac{|f^{-1}(1) \setminus (\cup_{\mathcal{P}\in} h_{\mathcal{P}}^{-1}(1))|}{|f^{-1}(1)|} \leq \frac{\varepsilon}{10s}; \tag{10}$$

˜否则，鉴于我们抽取了 $O(s/\varepsilon)$ 个样本，阶段 4 以至少 0.99 的概率拒绝，因此我们就完成了。

另一方面，根据 Chernoff 界以及对阶段 4 抽取的样本随机性应用并集界，以至少 0.99 的概率，我们有

$$\mathbf{Pr}_{\boldsymbol{z}\sim f^{-1}(1)}\left[h_{\mathcal{P}}(\boldsymbol{z})=1\right] \geq \frac{\varepsilon}{30s}.$$

对所有 $\mathcal{P} \in$Heavy 均成立，且

$$\mathbf{Pr}_{\boldsymbol{z}\sim f^{-1}(1)}\left[h_{\mathcal{P}}(\boldsymbol{z})=1\right] \leq \frac{\varepsilon}{10s}. \tag{11}$$

对所有未被选入 Heavy 的 $\mathcal{P}$ 均成立。

结合 公式（10）和 公式（11），我们有

$$\frac{|f^{-1}(1) \setminus (\cup_{\mathcal{P}\in \mathsf{Heavy}} h_{\mathcal{P}}^{-1}(1))|}{|f^{-1}(1)|} \leq \frac{\varepsilon}{10s} + s \cdot \frac{\varepsilon}{10s} \leq \frac{\varepsilon}{2}.$$

引理的证明到此结束。 □

还需证明：如果算法在否情形下完成阶段 5，那么对于每个$\mathcal{P} \in$Heavy，我们有

$$\begin{aligned}\Big|\mathbf{Pr}\left[\textbf{Test-Factored-DNF}(\mathrm{MQ}^*_{\mathcal{P}}, \mathrm{SAMP}^*_{\mathcal{P}}) \text{ accepts}\right] \\ - \mathbf{Pr}\left[\textbf{Test-Factored-DNF}(\mathrm{MQ}(h_{\mathcal{P}}), \mathrm{SAMP}(h_{\mathcal{P}})) \text{ accepts}\right]\Big| \leq 0.1,\end{aligned} \tag{12}$$

为方便排版，我们省略了 r、μ 和 $\varepsilon/(2s)$ 在 **Test-Factored-DNF** 中的参数。回顾一下，成员查询$^*_{\mathcal{P}}$ 与 $\mathrm{MQ}(h_{\mathcal{P}})$ 完全相同，因此我们只需证明 $\mathrm{SAMP}^*_{\mathcal{P}}$的表现与 $\mathrm{SAMP}(h_{\mathcal{P}})$类似。下面的引理给出了这一结论：

引理46。满足 概率至少为 $1-1/N$ 时，$\mathrm{SAMP}^*_{\mathcal{P}}$返回从 $\mathrm{SAMP}(h_{\mathcal{P}})$ 中均匀抽取的样本 .

证明。 由于每个 $\mathcal{P} \in$ Heavy 都满足 方程（9），$\mathrm{SAMP}^*_{\mathcal{P}}$ 返回默认值 0^n 的概率至多为 $(1-\varepsilon/(30s))^{100(s/\varepsilon)\alpha} \leq 1/N$。若未发生这种情况，$\mathrm{SAMP}^*_{\mathcal{P}}$ 则返回从 $\mathrm{SAMP}(h_{\mathcal{P}})$中均匀抽取的样本。 □

鉴于 **Test-Factored-DNF** 至多抽取 Q 个样本自 $\mathrm{SAMP}^*_{\mathcal{P}}$，根据并集界，它获得 Q个均匀样本自 $\mathrm{SAMP}(h_{\mathcal{P}})$的概率至少为 $1-Q/N$。因此，对于每个$\mathcal{P} \in$ Heavy （为方便起见，令 $A_{\mathcal{P}}$ 表示来自成员查询$^*_{\mathcal{P}}$ 和 $\mathrm{SAMP}^*_{\mathcal{P}}$ 的概率，并令 $B_{\mathcal{P}}$表示来自 $\mathrm{MQ}(h_{\mathcal{P}})$和 $\mathrm{SAMP}(h_{\mathcal{P}})$的概率），我们有

$$(1-Q/N)B_{\mathcal{P}} \leq A_{\mathcal{P}} \leq (Q/N) + (1-Q/N)B_{\mathcal{P}} \leq (Q/N) + B_{\mathcal{P}}, \tag{13}$$

这意味着 $|A_{\mathcal{P}} - B_{\mathcal{P}}| \leq Q/N$ 因此，方程（12）只要 $N \gg Q$就成立。至此，定理 23的证明。

算法 9 ~~Test-Factored-DNF~~

输入： 某个函数 h 的成员查询(h)和 SAMP(h)：$\{0,1\}^n \to \{0,1\}$、 r、 μ 和 ε。
输出： 接受或拒绝。

1: 抽取取 $O((\mu\log\ \mu)/\xi)$ 个样本 $\boldsymbol{z} \sim$ SAMP(h)，并将这些样本的集合称为 Samples。L et

$$\mathsf{R} = \left\{ i \in [n] \mid \exists \boldsymbol{z}, \boldsymbol{z}' \in \text{Samples with } \boldsymbol{z}_i = 1, \boldsymbol{z}'_i = 0 \right\}$$

若 $\mathsf{R} = \emptyset$，则接受。2：运行 DNFLearner(R, MQ(h), SAMP(h))。3：如果它拒绝，则拒绝；否则，令 (MQ*, SAMP*) 为 DNFLearner 返回的结果。4：运行 ConsCheck(R, MQ(h), SAMP(h), MQ*, SAMP*)，并返回相同的输出。

算法 9：**Test-Factored-DNF**。该算法的输入是对函数 h 的预言机访问，r，$\mu \in \mathbb{N}$ 以及 $\varepsilon > 0$，并测试 h 是否 ε-接近于某个 $(r,\ \mu)$-因子化析取范式。

7 测试因子化析取范式

在本节中，我们介绍针对 $(r,\ \mu)$-因子化析取范式的测试器，并证明 定理 20。该测试器 **Test-Factored-DNF**见 算法 9。

Test-Factored-DNF 从初始化阶段（第 1–2 行）开始。初始化完成后，**Test-Factored-DNF** 执行的两个主要步骤是 **DNFLearner** 和**ConsCheck**，我们将在第8节和第9节中分别介绍并分析它们。本节中，我们证明一个关于初始化阶段在是情形下所达到效果的简单引理（下文的 引理 47 ）。之后，我们解释 **DNFLearner** 和 **ConsCheck** 过程所实现的功能，并正式陈述它们的性能保证。我们将证明推迟到 第8节和第9节，利用它们的性能保证，在本节末尾证明定理 20 。

7.1 参数

为清晰起见，我们假设 r、 μ 和 ε 这三个 **Test-Factored-DNF**的输入参数，由 **DNFLearner** 和 **ConsCheck** 过程及其子程序共享。因此，我们无需将它们作为输入参数逐层传递。我们还将在本节以及 κ 和 ξ 中使用以下两个参数第8节 和 第9节：

$$\kappa := \varepsilon^2 \quad \text{and} \quad \xi := \frac{\kappa}{4000} \cdot \frac{1}{\log(100 \cdot 2^{2\mu r})}. \tag{14}$$

直观地说，ξ是一个“相当小”的量，而 $1/\xi$ 仍然是关于r，μ，$1/\varepsilon$的多项式。我们还假设所有算法共享 κ 和 ξ 。

7.2 初始化

在初始化过程中，**Test-Factored-DNF** 从 $O((\mu\log\ \mu)/\xi)$ 中抽取 SAMP(h) 个样本，并令R 为这些样本意见不一致的坐标 $i \in [n]$ 。注意，当 $|h^{-1}(1)| = 1$

（此时 h 是一个平凡的（r, μ）-因子化析取范式），算法总是在第 2 行正确接受；当 $|h^{-1}(1)| > 1$ 时，初始化结束时得到 $\mathsf{R} = \emptyset$ 的可能性极低。不失一般性，在证明的其余部分中，我们始终假设 $|h^{-1}(1)| > 1$ 且 $\mathsf{R} \neq \emptyset$。

我们证明关于 R 的以下引理，其中 h 是一个 (r, μ)-因子化析取范式。回顾一下，当 h 是一个 (r, μ)-因子化析取范式时，我们可以将其写成 $h = H \wedge (T_1 \vee \ldots \vee T_{\leq r})$，其中：

- $H, T_1, \ldots, T_{\leq r}$ 都是项。
- 在 $T_1, \ldots, T_{\leq r}$ 中至多有 μ 个变量。我们将这组变量称为 S。
- 我们有 $\mathrm{var}(H) \cap \mathsf{S} = \emptyset$。

给定一个集合 $\mathsf{R} \subseteq [n]$，若 $\mathrm{var}(H) \subseteq \mathsf{R}$ 且存在一个字符串 $u \in \{0,1\}^{\mathsf{V}}$，其中 $\overline{\mathsf{V}} := \mathsf{R} \cap (\mathsf{S} \cup \mathrm{var}(H))$，使得 R 关于 h 是稳定的，则称 R 为稳定的。

$$\mathop{\mathbf{Pr}}_{\boldsymbol{z} \sim h^{-1}(1)} \left[\boldsymbol{z}_{\mathsf{V}} \neq u \right] \leq \xi. \tag{15}$$

当 R 稳定时，字符串 $u \in \{0,1\}^{\mathsf{V}}$ 显然是唯一的。

引理 47. 当 h 是一个 (r, μ)-因子化析取范式时，R 以至少 0.99 的概率是稳定的。

证明。首先注意，每个 $\boldsymbol{z} \in h^{-1}(1)$ 都必须满足 $H(\boldsymbol{z}) = 1$。因此，所有样本在 $\mathrm{var}(H)$ 中的坐标上必须一致，这意味着 $\mathrm{var}(H) \subseteq \mathsf{R}$。

对于引理的第二部分，令 $\boldsymbol{u} \in \{0,1\}^{\mathsf{R}}$ 为唯一的字符串，使得对所有 $i \in \mathsf{R}$ 和 $\boldsymbol{z} \in$ 样本都有 $\boldsymbol{u}_i = \boldsymbol{z}_i$。固定任意坐标 $i \in \mathsf{S}$ 和位 $b \in \{0,1\}$，使得

$$\mathop{\mathbf{Pr}}_{\boldsymbol{z} \sim h^{-1}(1)} \left[\boldsymbol{z}_i = b \right] \geq \xi / \mu.$$

存在满足 $\boldsymbol{u}_i = 1 - b$ 的 $i \in \overline{\mathsf{R}}$ 的概率至多为 $(1 - \xi/\mu)^{\Delta} \leq 1/200\mu$，其中 $\Delta = O((\mu \log \mu)/\xi)$ 是抽取的样本数。因此，通过对 S 中至多 μ 个坐标以及 $b \in \{0,1\}$ 应用并集界，我们以至少 0.99 的概率得到：每个 $i \in \mathsf{S} \cap \mathsf{R}$ 都满足 $\mathbf{Pr}_{\boldsymbol{z} \sim h^{-1}(1)}[\boldsymbol{z}_i = \boldsymbol{u}_i] \geq 1 - \xi/\mu$。鉴于样本在 $\mathrm{var}(H)$ 中的坐标上始终一致，通过对 $i \in \mathsf{S} \cap \mathsf{R}$ 应用并集界，我们可以得出

$$\mathop{\mathbf{Pr}}_{\boldsymbol{z} \sim h^{-1}(1)} \left[\boldsymbol{z}_{\mathsf{V}} \neq \boldsymbol{u}_{\mathsf{V}} \right] \leq |\mathsf{S}| \cdot \frac{\xi}{\mu} \leq \xi.$$

这就完成了该引理的证明。 □

7.3 DNFLearner的性能保证

接下来我们将详细说明 **DNFLearner** 旨在实现的目标，并陈述其性能保证 s.

为此，考虑肯定情形，即我们假设 h 是一个 (r, μ)-因子化析取范式，且R是稳定的。令 $u \in \{0,1\}^{\mathsf{V}}$ 为满足公式（15）的唯一字符串。根据V的定义，$\mathsf{R} \setminus \mathsf{V}$ 中的每个变量都是无关变量。由公式（15）可直接推出，存在唯一的函数 $\mathfrak{h}: \{0,1\}^{\mathsf{R}} \to \{0,1\}$ 使得

$$\mathbf{Pr}_{\boldsymbol{z} \sim h^{-1}(1)} \left[h|_{\boldsymbol{z}_{\overline{\mathsf{R}}}} \neq \mathfrak{h} \right] \leq \xi.$$

函数 H 可由 h 得到：按照 u 设置 V 中的变量，并任意设置 R \V 中的变量。我们将 H 称为 h 在 R 上的支配函数；当 R 稳定时，它在肯定情形下定义良好。注意，由于 h 是一个 (r, μ)- 因子化析取范式，H 必须是一个至多涉及 R 中 μ 个变量的 r-项析取范式。

特别地，给定一个 r项析取范式 D ，其至多依赖于 μ 个变量，我们将 D称为 r项、 μ相关函数析取范式。

DNFLearner 的目标是学习支配函数 H。它将 R、 h 的预言机（MQ(h)、SAMP(h)）以及参数 r、μ 和 ε 作为输入。终止时，它要么拒绝，要么输出一对针对某个函数 J : $\{0,1\}^R \to \{0,1\}$的预言机模拟器（MQ*、SAMP*）；其中，MQ* 和 SAMP* 都是在 MQ(h) 上具有低查询复杂度的随机化算法。

我们总结如下 **DNFLearner**的性能保证（稍后将予以证明）：

引理 48。DNFLearner(R, MQ(h), SAMP(h)) 对 MQ(h)、SAMP(h) 进行 poly($r, \mu, 1/\varepsilon$) 次调用。它要么拒绝，要么返回一对（MQ*、SAMP*），其中 MQ* 和 SAMP* 是随机化算法，每次调用时分别对 MQ(h) 发出 $O(\mu \log(\mu/\varepsilon))$ 次查询。

定理 49。（DNFLearner 在是情形下的性能）假设 h 是 (r, μ)-因子化析取范式，且 R $\subseteq [n]$ 关于 h 稳定。那么，以至少 $19/20$ 的概率，DNFLearner 不会拒绝，并且返回的二元组 (MQ*, SAMP*) 是某个函数 J : $\{0,1\}^R \to \{0,1\}$ 的完美预言机模拟器，且 reldist(H, J) $\leq \kappa$，其中 H 是 h 在 R 上的支配函数。

定理 50。（DNFLearner 在否情形下的性能）假设 h 与 (r, μ)-因子化析取范式的相对距离至少为 ε，且 R $\subseteq [n]$。那么，以至少 $19/20$ 的概率，DNFLearner 要么拒绝，要么返回 (MQ*, SAMP*)，它们是某个函数 J :$\{0,1\}^R \to \{0,1\}$ 的 κ-准确预言机模拟器，且对于某个 r-项 μ-相关函数析取范式 D，有 reldist(J, D) $\leq \kappa$。

7.4 ConsCheck 的性能保证

粗略地说，当 h 是一个（r， μ）-因子化析取范式，R 稳定，并且 J 满足 reldist(H，J) $\leq \kappa$ 时，可以证明 h 与 $C(x_R) \cdot J(x_R)$ 接近，其中 C 是 $\{0,1\}^R$ 上的某个合取。另一方面，在否情形中，如果 J 在相对距离意义下接近某个 r-项、 μ-相关函数析取范式，那么对于任意$\{0,1\}^R$ 上的合取 C，h 必定与 $C(x_R) \cdot J(x_R)$ 相距甚远，因为在否情形中， h 在相对距离意义下与（r， μ）-因子化析取范式相距甚远。

基于上述直觉，**ConsCheck** 的目标是使用 （MQ*，SAMP*）预言机模拟器 J ，这些模拟器由 **DNFLearner**传递下来，用以区分这两种情形。下面总结其性能保证：

引理51。 **ConsCheck**(R，成员查询(h)，SAMP(h)，成员查询*，SAMP*) 调用成员查询($O(1/\varepsilon)$ 的次数不超过 次，调用 成员查询(h)、SAMP(h)、成员查询* 和 SAMP*。最后，它要么接受，要么拒绝。

定理 52。 (**ConsCheck** 在“是”情形下的性能）假设 h 是一个 (r, μ)-因子化析取范式，R 是稳定的，并且 (成员查询*，SAMP*) 是某个函数 J : $\{0,1\}^R \to \{0,1\}$的完美模拟器，其中 相对距离(J，J) $\leq \kappa$. 那么 **ConsCheck** 以至少 $39/40$的概率接受。

定理53。（ConsCheck 在“否”情形下的性能）假设 h 与 (r, μ)-因子化析取范式的相对距离至少为 ε，且 R $\subseteq [n]$；对于某个函数 J : $\{0,1\}^R \to \{0,1\}$，(成员查询*, SAMP*) 是 κ-准确的预言机模拟器，并且对于某个 r-项、 μ-相关函数析取范式 D，有 reldist(J, D) $\leq \kappa$。那么 ConsCheck 以至少 $19/20$ 的概率拒绝。

7.5 定理 20 的证明

上述各部分就位后，定理 20 的证明很简单。

定理 20 的证明。为了界定 Test-Factored-DNF 的复杂度，我们注意到：(1) 在初始化期间，它调用 SAMP(h) poly$(r,\mu,1/\varepsilon)$ 次；(2) DNFLearner 调用 MQ(h) 和 SAMP(h) poly$(r,\mu,1/\varepsilon)$˜次；(3) ConsCheck 调用 MQ(h)、SAMP(h)、MQ*和 SAMP*共 $O(1/\varepsilon)$ 次。鉴于每次调用 MQ* 和 SAMP* 最多会调用 MQ(h) $O(\mu\log(\mu/\varepsilon))$ 次，Test-Factored-DNF 的总体复杂度为 poly$(r,\mu,1/\varepsilon)$。

关于正确性，当 h 是一个 (r,μ)-因子化析取范式时，根据引理 47、定理 49 和定理 52，再结合并集界可知，Test-Factored-DNF 以至少 0.9 的概率接受。当 h 与 (r,μ)-因子化析取范式的相对距离至少为 ε时，根据定理 50 和定理 53，再结合并集界可知，Test-Factored-DNF 以至少 0.9 的概率拒绝。□

8 DNFLearner 算法

该 DNFLearner 见 算法 10。我们先对它的三个阶段作一些高层次的直观说明。我们的概述侧重于是情形，其中 h 是一个 (r,μ)-因子化析取范式，并且$\mathsf{R} \subseteq [n]$ 是一个稳定集（相对于 h）。回顾一下，在这种情况下，我们记 $\mathfrak{H}:\{0,1\}^{\mathsf{R}} \to \{0,1\}$ 为支配函数，它是一个 r-项、 μ-相关函数析取范式，并满足

$$\Pr_{z\sim h^{-1}(1)}\left[h\restriction_{z_{\bar{\mathsf{R}}}} \neq \mathfrak{H}\right] \le \xi.$$

我们先介绍一些符号与设置，然后依次分析该算法的三个阶段，最后证明 定理 49、定理 50 和 引理 48。显然，本节的大部分工作都集中在是情形。关于否情形的论证（定理 50）仅出现在第 第 8.4 节中的阶段 3 分析里。

8.1 符号与设置

我们首先定义变量预言机列表这一有用概念：

定义 54. 我们称有序元组 $L = ($(成员查询(g^1)，X_1)$\ldots$，(成员查询(g^ℓ)，X_ℓ)（其中 $\ell \ge 0$）为定义域不相交的预言机列表，如果每个成员查询(g^i)都是某个函数 $g^i:\{0,1\}^{\mathsf{X}_i} \to \{0,1\}$的成员关系预言机，$\mathsf{X}_i \subseteq \mathsf{R}$，且各个 X_i两两不相交。

此外，如果每个函数 g^i 在均匀分布下都与某个文字 $x_{\tau(i)}$ 或 $x_{\tau(i)}$¯相差至多 (1/30)，其中存在某个 $\tau(i) \in \mathsf{X}_i$，我们就称 L 为变量预言机列表。进一步地，如果每个 g^i 都恰好是一个文字，我们就称 L 为完美变量预言机列表。（注意，空列表 L显然是完美变量预言机列表。）

给定变量预言机列表 L （使得 $\tau(i)$ 唯一定义），我们将单射映射 σ_L $\sigma_L:[\ell] \to \mathsf{R}$ 记为 $\sigma_L(i) = \tau(i)$，其中对每个 $i \in [\ell]$ 均成立。

展望后文，在阶段 1 构建变量预言机列表 L 时，每个函数 g^i 都会被选为 $h\restriction_v$，其中 $v \in \{0,1\}^{\mathsf{X}_i}$ 是算法在过程中找到的某个字符串，因此可以使用 MQ(h) 按查询逐一精确模拟 MQ(g^i)。

回想一下，在肯定情形中，我们要学习的支配函数 H 始终是 R 上一个最多涉及 μ 个变量的 r 项析取范式（因此是一个 μ-相关函数）。这一算法的动机来自这样一个事实：该算法不会

算法 10 DNFLearner

输入： $\emptyset \neq \mathsf{R} \subseteq [n]$、某个函数 $h : \{0,1\}^n \to \{0,1\}$ 的成员查询（h）和 SAMP（h）。**输出：** 拒绝，或为某个函数 $\mathsf{J} : \{0,1\}^{\mathsf{R}} \to \{0,1\}$ 提供预言机模拟器（SAMP*，MQ*）。

阶段1：预处理

1: 运行 Approximate(成员查询(h)，SAMP(h)，R)。2: 如果拒绝，则拒绝；否则令 $\boldsymbol{L}$ 为返回的局部预言机列表。3: 如果 $|\boldsymbol{L}| = 0$，则返回 (成员查询($\mathbf{1}_{\mathsf{R}}$)，SAMP($\mathbf{1}_{\mathsf{R}}$))。▷ $\mathbf{1}_{\mathsf{R}}$ 是从 $\{0,1\}^{\mathsf{R}}$ 到 $\{0,1\}$ 的全 1 函数

阶段 2：寻找良好近似函数

4: 运行 FindCandidate(SAMP(h) $\boldsymbol{L}$，R)。5: 如果拒绝，则拒绝；否则令 J 为返回的函数（以真值表形式表示）。

阶段 3：为近似函数构造预言机

6: 运行 CheckLit($\boldsymbol{L}$)；如果它拒绝，则拒绝。7: 令 SAMP* ← Sim-SAMP-J(R $\boldsymbol{L}$, J)，并令 MQ* ← Sim-MQ-J(·,R $\boldsymbol{L}$, J)8: 返回 (SAMP*,MQ*)。

算法 10：**DNFLearner**。该算法以对 h 的预言机访问和集合 $\mathsf{R} \subseteq [n]$ 为输入。算法的目标是返回函数 MQ 和 SAMP 预言机的模拟器J：$\{0,1\}^{\mathsf{R}} \to \{0,1\}$，其中 J 与某个 r-项、μ-junta 析取范式 D接近。在肯定情形下，回顾一下，h 表示定义在 R 上的支配函数；以高概率，h 与 J 的相对距离非常小。

为了能够保证找到所有相关变量（因为其中一些变量的影响度可能较低），我们针对每个 $\ell \in [\mu]$ 引入一个由定义在 $\{0,1\}^{\ell}$ 上的函数构成的集合 DNF-Approx(ℓ)。粗略地说，DNF-Approx(ℓ) 中的函数以相对误差 $\kappa/2$ 近似 r-项、μ-相关函数析取范式。

形式化地说，给定任意 r-项析取范式 D：$\{0,1\}^{\mu} \to \{0,1\}$，定义 D'：$\{0,1\}^{\mu} \to \{0,1\}$ 如下：

$$\mathsf{D}'(x) = \arg\max_{b \in \{0,1\}} \left\{ \mathop{\mathbf{Pr}}_{\boldsymbol{w} \sim \{0,1\}^{\mu-\ell}} \left[\mathsf{D}\left(x_{[\ell]} \circ \boldsymbol{w} \right) = b \right] \right\}.$$

显然，D' 仅　　取决于前 ℓ 个变量。令 J_{D}：$\{0,1\}^{\ell} \to \{0,1\}$ 为满足以下条件的函数

$$\mathsf{J}_{\mathsf{D}}(z) = \mathsf{D}'\left(z \circ 0^{r-\ell} \right).$$

当且仅当 reldist(D，D′) $\leq \kappa/2$ 时，将函数 J_{D} 加入 DNF-Approx(ℓ)，也就是说，DNF-Approx(ℓ) 包含所有函数 J_{D}：当 D 遍历所有关于 $\{0,1\}^{\mu}$ 的 r-项析取范式且满足 reldist(D，D′) $\leq \kappa/2$ 时所得到的函数。

请注意，算法无需进行任何查询即可构建集合 DNF-Approx(ℓ)。

展望未来，（成员查询*，SAMP*）对**DNFLearner** 最终输出的这对对象始终是J_{σ}[9] 的预言机模拟器，其中 $\ell \in [\mu]$，$\mathsf{J} \in$ DNF-Approx（ℓ）以及单射映射 $\sigma : [\ell] \to \mathsf{R}$（DNFLearner**第 3 行的平凡情况除外**）。我们记录关于DNF-Approx（ℓ）的两个引理：

[9]有关记号 J_{σ}.含义的说明，请参见 第 3.3 节末尾

引理 55。 对于任意 $\ell \in [\mu]$，我们有 $|\text{DNF-Approx}(\ell)| \le 2^{2\mu r}$。

证明。 根据定义， $|\text{DNF-Approx}(\ell)|$ 的上界为 $\{0,1\}^{\mu}$ 上 r 项析取范式 D 的数量。后者可由 $(2^{2\mu})^r$ 上界限制。 □

引理 56。 给定任意函数 $J \in \text{DNF-Approx}(\ell)$ 和任意单射映射 $\sigma : [\ell] \to R$，存在某个 r 项、 μ-相关函数析取范式 D，使得：$\{0,1\}^R \to \{0,1\}$。

证明。 根据 DNF-Approx(ℓ) 的定义，对于某个 r 项析取范式 D，有 $J = J_D$： $\{0,1\}^{\mu} \to \{0,1\}$。考虑任意满足以下条件的单射映射 $\pi : [\mu] \to R$：对每个 $i \in [\ell]$， $\pi(i) = \sigma(i)$。于是我们有

$$\text{reldist}\left(\mathcal{D}_\pi, J_\sigma\right) = \text{reldist}(\mathcal{D}, \mathcal{D}') \le \kappa/2.$$

根据对称性（见 引理 3），我们有 相对距离$(J_\sigma, \mathcal{D}_\pi) \le \kappa$。显然， $\mathcal{D}_\pi$ 是一个 r项、 μ相关函数析取范式，定义在 $\{0,1\}^R$上。 □

最后，给定集合 R，我们用 $\mathbf{1}_R$ 表示从 $\{0,1\}^R \to \{0,1\}$ 到全 1 函数。注意，算法可以轻易构造 MQ($\mathbf{1}_R$)（始终返回 1）和 SAMP($\mathbf{1}_R$)（返回 $z \sim \{0,1\}^R$）。

8.2 阶段 1: Approximate

考虑肯定情形。Approximate 以 MQ(h)、SAMP(h) 以及集合 $\emptyset \ne R \subseteq [n]$ 作为输入，我们假设该集合稳定。令 $H : \{0,1\}^R \to \{0,1\}$ 为支配函数，并令 S′ 为 h 在 R 中的相关变量集合。由于 var(H) ⊆R（因为 R 稳定），我们有 $|S'| \le \mu$。我们还知道 H 的所有相关变量都在 S′ 中。

在肯定情形下，Approximate 的目标是尽可能多地找出 H 至多 μ 个相关变量。注意，我们无法负担使用 O_r、μ、ε 次查询来显式找出其中任何一个变量。相反，Approximate 首先将 R 随机划分为 $\tau = O(\mu^2)$ 个互不相交的块 $X_1, \ldots, X_\tau$。根据生日悖论，只需付出较小的常数失败概率，我们就可以假设对于所有 $i \in [\tau]$，都有 $|S' \cap X_i| \le 1$。随后，Approximate 反复对各块执行二分搜索过程，以找出 (i) 一个新的“相关块”，即满足 $|S \cap X_i| = 1$ 的某个 X_i；以及 (ii) 对于找到的每个新“相关块”X_i，找出函数 h 的一个限制 g^i：$\{0,1\}^{X_i} \to \{0,1\}$，该限制恰好是文字 $x_{\tau(i)}$ 或 $\overline{x_{\tau(i)}}$，对应于 $S' \cap X_i$ 中唯一的 $x_{\tau(i)}$。随后将这一对 (MQ(g^i), X_i) 加入列表，从而得到一个完美变量预言机列表，供 Approximate 在最后返回。

如上所述，我们的算法所使用的一个关键子程序是在变量“块”上进行二分搜索，以查找新的相关块。更具体地，设 $X_1, \ldots X_\tau$ 是 R 的一个划分，并固定某个 $I \subseteq [\tau]$，满足 $X = \cup_{\ell \in I} X_\ell$。给定 $u \in \{0,1\}^n$ 和 $w \in \{0,1\}^{R \setminus X}$，使得 $h(u) \ne h(z_R \circ z_X \circ w)$，利用二分搜索可以直接确定一个 $X_\ell, \ell \notin I$，以及一个字符串 $v \in \{0,1\}^n$，使得

$$h(v) \ne h\left(w_{X_\ell} \circ v_{\overline{X_\ell}}\right),$$

使用 $O(\log \tau) = O(\log \mu)$ 次查询来 h。其（简单且标准的）思路如下：令 $U = [\tau] \setminus I$为 X之外的块的索引，并令 $a = z, b = z_R \circ z_X \circ w$。重复执行以下步骤，直到 $|U| = 1$:

令 U′ 为 U 的前半部分。查询 c，它是 a，其块位于 U′ 集合中，依据 b。如果$f(a) \ne f(c)$，我们令 $U = U'$ 并 $b = c$；否则我们令 $U = U \setminus U'$ 并 $a = c$。

我们的目标是证明以下两个关于 Approximate 的引理:

引理 57。 Approximate(成员查询(h)，SAMP(h)， R) 进行 poly(μ， $1/\xi$) 次调用 SAMP(h)，成员查询(h)。

证明。 重复执行第 3–9 行，直到 $|\boldsymbol{L}| > \mu$ 或 $t = t^*$。在循环的一次迭代中，Approximate在第 4 行从 h 抽取一个样本。如果第 5 行的"if"条件为真，则执行第 6 行，进行 $O(\log \tau) = O(\log \mu)$ 次查询，并且 $|\boldsymbol{L}|$ 加一，同时将 t 设为 0。如果第 5 行的"if"条件为假，则第 5 行只进行一次查询，且 $|\boldsymbol{L}|$ 不增加，但 t加一。因此， Approximate 的查询复杂度为

$$O\big(\mu t^* + \mu \log \mu\big) = \mathrm{poly}(\mu/\xi). \quad \square$$

引理 58。假设 h 是一个 (r, μ)-因子化析取范式，且 R 稳定。以至少 $99/100$ 的概率，Approximate 返回一个完美列表 $\boldsymbol{L}$，其中包含满足 $|\boldsymbol{L}| \le \mu$ 的局部预言机，并具有以下性质：存在一个函数 $G \in$ DNF-Approx($|\boldsymbol{L}|$)，使得 reldist(H, $G^{\sigma_{\boldsymbol{L}}}$) $\le \xi$；对于特殊情况，即 $\boldsymbol{L}$ 为空时，我们有 reldist(H $\mathbf{1}_R$) $\le \xi$，其中 $\mathbf{1}_R$ 是定义在 $\{0,1\}^R$ 上的全 1 函数。

我们在本小节余下部分证明 引理 58 。我们始终假设 h 是一个 $(r,\ \mu)$-因子析取范式，且 R 是稳定的。我们先证明一个关于随机划分的简单引理：

Le引理 59。 我们有 $|\mathbf{X}_i \cap S'| \le 1$ 对所有 $i \in [\tau]$ 均成立，且其概率至少为 $199/200$.

证明。 任取 j_1， $j_2 \in S'$。如果它们都属于 R，则它们落在同一块中的概率为 $1/\tau$；否则概率为 0。由并集界以及 $|S'| \le \mu$这一假设，存在某个 i 使得$|\mathbf{X}_i \cap S'| > 1$ 的概率至多为

$$\binom{\mu}{2} \cdot \frac{1}{\tau} \le 1/200. \quad \square$$

从现在起，我们固定 R 的划分 X_1，……，X_τ，并假设对每个 $i \in [\tau]$都有 $|X_i \cap S'| \le 1$。我们用 $\boldsymbol{L} = ((\mathrm{MQ}(g^1), X_1) \ldots\ldots, (\mathrm{MQ}(g^\ell), X_\ell))$ 表示返回的列表，其中 $\ell = |\boldsymbol{L}|$。

接下来我们证明一个关于 $\boldsymbol{L}$ 和 $\mathbf{X}$ 在 Approximate 终止时的引理：

引理 60。 假设 $|X_i \cap S'| \le 1$ 对所有 $i \in [\tau]$都成立。那么 Approximate 绝不会拒绝，并且总会返回一个完美列表 $\boldsymbol{L}$，其中包含满足 $|\boldsymbol{L}| \le \mu$的变量预言机。此外，概率至少为199/200时，当Approximate终止时，我们有

$$\Pr_{\substack{\boldsymbol{z} \sim h^{-1}(1) \\ \boldsymbol{w} \sim \{0,1\}^{R \setminus \mathbf{X}}}} \Big[h(\boldsymbol{z}_{\overline{R} \cup \mathbf{X}} \circ \boldsymbol{w}) = 0 \Big] \le \frac{\xi}{5}. \tag{16}$$

证明。显然， $\boldsymbol{L}$ 中的 X_i 彼此不相交。对于加入 $\boldsymbol{L}$ 的每一对 $(\mathrm{MQ}(g^i), X_i)$，Approximate 都找到了 h 的一个限制 g^i，使得 $g^i : \{0,1\}^{X_i} \to \{0,1\}$ 不是常值函数。由于在 R 中， h 只依赖于 S' 中的变量以及 $|S' \cap X_i| \le 1$，因此必有 $|S' \cap X_i| = 1$，且 g^i 是 $S' \cap X_i$ 中该变量的一个文字。因此， $\boldsymbol{L}$ 是一个完美变量预言机列表。鉴于 $|S'| \le \mu$， $\boldsymbol{L}$ 不可能大于 μ，因此 Approximate 绝不会拒绝。

若要违反公式（16），则返回时必须满足Approximate 已经抽取了$\boldsymbol{z} \sim h^{-1}$（1）和 $\boldsymbol{w} \sim \{0,1\}^{R \setminus \mathbf{X}}$，针对 t^* 多次，并且 h（$\boldsymbol{z}_{R \cup X} \circ \boldsymbol{w}$）$= 1$ 每次都如此。然而，鉴于违反了公式（16），这种情况发生的概率至多为

$$\big(1 - \xi/5\big)^{t^*} \le e^{-\ln(200\mu)} \le 1/200\mu.$$

根据并集界（对第 5 行的所有执行过程中涉及的至多 μ 个不同的 $\mathbf{X}$），Approximate 在违反 公式（16）的情况下返回的概率至多为 $1/200$。 $\square$

算法 11 Approximate

输入： 函数 h 的成员查询(h) 和 SAMP(h)：$\{0,1\}^n \to \{0,1\}$ 和 $\emptyset \neq \mathrm{R} \subseteq [n]$。**输出：** 要么拒绝，要么返回一个由定义域不相交的预言机组成的列表 $\boldsymbol{L}$ 。

阶段 1：将R划分为块

令 $\tau = \theta(\mu^2)$；将 R 随机划分为块 $\mathbf{X}_1, \ldots, \mathbf{X}_\tau$。

阶段 2：查找相关块

设置 $\mathbf{X} = \emptyset$； $\boldsymbol{L}$ 为空列表； $t = 0$ 并且 $t^* = 5\ln(200\mu)/\xi$。**当** $t \le t^*$ **时** 抽取 $\boldsymbol{z} \sim \mathrm{SAMP}(h)$ 并 $\boldsymbol{w} \sim \{0,1\}^{\mathrm{R}\setminus\mathbf{X}}$，并设置 $t \leftarrow t+1$。**如果** $h(\boldsymbol{z}_{\mathrm{R}\cup\mathbf{X}} \circ \boldsymbol{w}) = 0$ **则** ▷ 类型 1 查询

通过二分搜索找到一个新的相关块 $\mathbf{X}_i$ 以及一个字符串 $\boldsymbol{v} \in \{0,1\}^n$，使得

$$1 = h(\boldsymbol{v}) \neq h\left(\boldsymbol{w}_{\mathbf{X}_i} \circ \boldsymbol{v}_{\overline{\mathbf{X}}_i}\right) = 0.$$

▷ 二分搜索仅使用类型 1 查询

为方便记号起见，不失一般性地假设找到的第 i 个新块的索引为 $\mathbf{X}_i$。将以下函数记为 g^i：$\{0,1\}^{\mathbf{X}_i} \to \{0,1\}$

$$g^i(z) = h\left(z \circ \boldsymbol{v}_{\overline{\mathbf{X}}_i}\right).$$

▷ 对 g^i 的每次查询都是类型 1 查询

设置 $t \leftarrow 0$、$\mathbf{X} \leftarrow \mathbf{X} \sqcup \mathbf{X}_i$，并将 (MQ($g^i$)，$\mathbf{X}_i$) 添加到列表 $\boldsymbol{L}$ 中。如果 $|\boldsymbol{L}| > \mu$，则输出“拒绝”并停止。**结束 if** **结束 while** 输出 $\boldsymbol{L}$。

算法 11：Approximate。该算法以对 h 的预言机访问以及集合 $\mathrm{R} \subseteq [n]$ 作为输入。在肯定情形下，回顾一下，H 是 R 上的支配函数，并且它至多依赖于 μ 个变量。我们无法使用 O_r、μ、$\varepsilon(1)$ 次查询显式地找到这些变量。因此，该算法的目标是生成一个完美变量预言机列表 $\boldsymbol{L}$，其中包含 H 所依赖的“最相关”变量。具体而言，当算法返回时，以高概率存在某个函数 G $\in$DNF-Approx($|\boldsymbol{L}|$)，使得在由$\sigma_{\boldsymbol{L}}$ 给出的变量重新映射下，H 与 G 很接近（ $\sigma_{\boldsymbol{L}}$ 的定义见定义 54）。

本小节余下的内容都围绕着如何将公式（16）与我们的目标联系起来展开：存在一个函数$\mathfrak{G} \in \mathrm{DNF\text{-}Approx}(|\boldsymbol{L}|)$ 使得相对距离$(\mathfrak{H},\mathfrak{G}_{\sigma_{\boldsymbol{L}}}) \le \xi$ （以及 $\boldsymbol{L}$为空这一特殊且更简单的情形）。为此，鉴于 R 是稳定的，且$\mathfrak{H}:\{0,1\}^{\mathrm{R}} \to \{0,1\}$ 是支配函数，我们首先利用公式（16）得到关于$\mathfrak{H}$的不等式。

引理 61。公式（16）意味着

$$\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\mathsf{X}}}}\Big[\mathfrak{H}(\boldsymbol{y}_{\mathsf{X}}\circ\boldsymbol{w})=0\Big]\le\xi/4, \tag{17}$$

证明。 由于 R 是稳定的，我们有

$$\begin{aligned}\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{z}\sim h^{-1}(1),\\ \boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\mathsf{X}}}}\Big[h(\boldsymbol{z}_{\overline{\mathsf{R}}\cup\mathsf{X}}\circ\boldsymbol{w})=0\Big]&\ge\mathop{\mathbf{Pr}}_{\boldsymbol{z},\boldsymbol{w}}\Big[h(\boldsymbol{z}_{\overline{\mathsf{R}}\cup\mathsf{X}}\circ\boldsymbol{w})=0\,\Big|\,h\restriction_{\boldsymbol{z}_{\overline{\mathsf{R}}}}\equiv\mathfrak{H}\Big]\times\mathop{\mathbf{Pr}}_{\boldsymbol{z}\sim h^{-1}(1)}\big[h\restriction_{\boldsymbol{z}_{\overline{\mathsf{R}}}}\equiv\mathfrak{H}\big]\\&\ge(1-\xi)\cdot\mathop{\mathbf{Pr}}_{\boldsymbol{z},\boldsymbol{w}}\Big[\mathfrak{H}(\boldsymbol{z}_{\mathsf{X}}\circ\boldsymbol{w})=0\,\Big|\,h\restriction_{\boldsymbol{z}_{\overline{\mathsf{R}}}}=\mathfrak{H}\Big].\end{aligned}$$

但 c 在 $h\restriction_{\boldsymbol{z}_{\mathsf{R}}}\equiv\mathsf{H}$ 的条件下，我们有 $\boldsymbol{z}_{\mathsf{R}}\sim\mathsf{H}^{-1}(1)$。因此，上式变为 :

$$\xi/5\ge\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{z}\sim h^{-1}(1),\\ \boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\mathsf{X}}}}\Big[h(\boldsymbol{z}_{\overline{\mathsf{R}}\cup\mathsf{X}}\circ\boldsymbol{w})=0\Big]\ge(1-\xi)\cdot\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\mathsf{X}}}}\Big[\mathfrak{H}(\boldsymbol{y}\circ\boldsymbol{w})=0\Big],$$

由此 引理 61得证。 □

注意， $\boldsymbol{L}$ 为空这一特殊情形可由引理 61 得出，因此我们假设列表 $\boldsymbol{L}=$((成员查询(g^1), X_1) . . . , (成员查询 (g^ℓ), X_ℓ)) 非空，且 $\ell=|\boldsymbol{L}|>0$。对每个 $i\in[\ell]$， 令 $\tau(i)$ 为 $\mathsf{X}_i\cap\mathsf{S}'$ 中唯一满足 g^i 是 $x_{\tau(i)}$ 或 $\overline{x_{\tau(i)}}$ 的变量，并令 $\sigma=\sigma_{\boldsymbol{L}}$ 为满足对每个 $i\in[\ell]$ 均有 $\sigma(i)=\tau(i)$ 的单射映射。令

$$\Sigma:=\big\{\sigma(i):i\in[\ell]\big\}\quad\text{with}\quad\Sigma=\mathsf{S}'\cap\mathsf{X}$$

已知 $\mathsf{X}=\mathsf{X}_1\cup\cdots\cup\mathsf{X}_\ell$。令 $\mathsf{G}:\{0,1\}^\ell\to\{0,1\}$ 为以下 ℓ-变量函数:

$$\mathfrak{G}(z):=\arg\max_{b\in\{0,1\}}\left\{\mathop{\mathbf{Pr}}_{\boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\Sigma}}\Big[\mathfrak{H}\left(\sigma(z)\circ\boldsymbol{w}\right)=b\Big]\right\}. \tag{18}$$

引理 62。 公式（17） 意味着 相对距离$(\mathfrak{H},\mathfrak{G}_\sigma)\le\xi$。

证明。 注意到我们有:

$$\mathfrak{G}_\sigma(z)=\arg\max_{b\in\{0,1\}}\left\{\mathop{\mathbf{Pr}}_{\boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\Sigma}}\Big[\mathfrak{H}\left(z_\Sigma\circ\boldsymbol{w}\right)=b\Big]\right\}.$$

反设相对距离$(\mathsf{H},\mathsf{G}_\sigma)>\xi$。令 $F=\mathsf{H}^{-1}(1)$ 和 $G=\mathsf{G}_\sigma^{-1}(1)$。已知 $|F\triangle G|\ge\xi|F|$，我们考虑 $|F\setminus G|>\xi|F|/2$ 或 $|G\setminus F|>\xi|F|/2$. 这两种情况。为此，我们考虑一种等价的方式来抽取 $\boldsymbol{y}_{\mathsf{X}}\circ\boldsymbol{w}$，其中 $\boldsymbol{y}\sim\mathsf{H}^{-1}(1)$ 和 $\boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\mathsf{X}}$: 抽取 $\boldsymbol{y}\sim\mathsf{H}^{-1}(1)$ 和 $\boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\Sigma}$，并返回 $\boldsymbol{y}_\Sigma\circ\boldsymbol{w}$。这两个分布相同，因为$\Sigma=\mathsf{X}\cap\mathsf{S}'$ 是 X 中 H 的唯一相关变量。

$$\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\mathsf{X}}}}\Big[\mathfrak{H}\left(\boldsymbol{y}_{\mathsf{X}}\circ\boldsymbol{w}\right)=0\Big]=\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\Sigma}}}\Big[\mathfrak{H}\left(\boldsymbol{y}_{\Sigma}\circ\boldsymbol{w}\right)=0\Big]. \tag{19}$$

对于 $|F \setminus G| > \xi|F|/2$ 的情况，我们有

$$\Pr_{\substack{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\Sigma}}}\left[\mathfrak{H}\left(\boldsymbol{y}_\Sigma\circ\boldsymbol{w}\right)=0\right]\geq\Pr_{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)}\left[\boldsymbol{y}\in F\setminus G\right]\cdot\Pr_{\substack{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\Sigma}}}\left[\mathfrak{H}\left(\boldsymbol{y}_\Sigma\circ\boldsymbol{w}\right)=0\,\middle|\,\boldsymbol{y}\in F\setminus G\right].$$

右侧的第一个概率至少为 $\xi/2$；右侧的第二个概率至少为 $1/2$根据 $\mathfrak{G}_\sigma$ 在 $\mathfrak{H}$ 中的定义以及 $\boldsymbol{y}\in F\setminus G$ 这一条件，因此，$\mathfrak{G}_\sigma(\boldsymbol{y})=0$。结合 公式（17） 和 公式（19），即可得到矛盾。

对于 $|G\setminus F|\geq\xi|F|/2$ 的第二种情况，考虑任意一个固定的 $a\in G\setminus F$。由于 $a\in G$，我们有

$$\Pr_{\boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\Sigma}}\left[f\left(a_\Sigma\circ\boldsymbol{w}\right)=1\right]\geq 1/2$$

因此，必定至少存在 $2^{|\mathsf{R}\setminus\Sigma|}/2$ 个满足 $y_\Sigma=a_\Sigma$ 的 $y\in F$。所以

$$\Pr_{\substack{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\Sigma}}}\left[\boldsymbol{y}_\Sigma\circ\boldsymbol{w}=a\right]\geq\Pr_{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)}\left[\boldsymbol{y}_\Sigma=a_\Sigma\right]\cdot\Pr_{\boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\Sigma}}\left[\boldsymbol{w}=a_{\mathsf{R}\setminus\Sigma}\right]\geq\frac{2^{|\mathsf{R}\setminus\Sigma|}}{2|F|}\times 2^{-|\mathsf{R}\setminus\Sigma|}=\frac{1}{2|F|}.$$

因此，我们有

$$\Pr_{\substack{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\Sigma}}}\left[\mathfrak{H}\left(\boldsymbol{y}_\Sigma\circ\boldsymbol{w}\right)=0\right]\geq\sum_{a\in G\setminus F}\left(\Pr_{\substack{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\mathsf{R}\setminus\Sigma}}}\left[\boldsymbol{y}_\Sigma\circ\boldsymbol{w}=a\right]\right)\geq\frac{\xi|F|}{2}\cdot\frac{1}{2|F|}=\frac{\xi}{4}.$$

这就完成了该引理的证明。 □

引理 63. 如果 相对距离$(\mathfrak{H},\mathfrak{G}_\sigma)\leq\xi$ 那么 $\mathfrak{G}\in\text{DNF-}\textsc{Approx}(|\boldsymbol{L}|)$。

证明。 这直接由 DNF-Approx(|$\boldsymbol{L}$|) 的定义得出。给定 H，它是一个 r-项、 μ-相关函数析取范式，我们可以对其相关变量进行置换，从而得到定义 DNF-Approx 时所述的 D，使 G 成为其中的 D′ 。该引理随即由我们对 κ 和 ξ 的选择得出，即 $\xi\ll\kappa$。 □

最后，我们结合所有引理来证明 引理 58。

引理 *58* 的证明。 由 引理 59 和 引理 60 的并集界可知， **Approximate**不会拒绝，并会返回一个由变量预言机构成的完美列表 $\boldsymbol{L}$ ，其满足 $|\boldsymbol{L}|\leq\mu$ ，且 公式（16） 成立。当 $\boldsymbol{L}$ 为空时，这一特殊情形由 引理 61得出； $\boldsymbol{L}$ 为空意味着 $\mathbf{X}=\emptyset$，因此我们有：

$$\mathsf{reldist}(\mathfrak{H},\mathbf{1}_{\mathsf{R}})=\Pr_{\boldsymbol{w}\sim\{0,1\}^{\mathsf{R}}}[\mathfrak{H}(\boldsymbol{w})=0]\leq\xi/4.$$

当 $\boldsymbol{L}$ 非空时，一般情形可由引理 61 至 63推出。 □

算法 12 Extract

输入： 定义域不相交的预言机列表 $L = ((\text{成员查询}(g^1), \mathrm{X}_1) \ldots, (\text{成员查询}(g^\ell), \mathrm{X}_\ell))$，$w \in \{0,1\}^{\mathsf{R}}$ 和 δ。**输出：** 一个字符串 $z \in \{0,1\}^\ell$。

1: 对于 i从 1 到 ℓ 执行 2: 令 $Y_b = \{j \in \mathrm{X}_i : w_j = b\}$ 对于 $b \in \{0,1\}$ 二者，并设置计数器 $G_0 = G_1 = 0$。 3: 重复 $O(\log(\mu/\delta))$ 次 4: 为二者抽取 $a^b \sim \{0,1\}^{Y_b}$ $b \in \{0,1\}$。 5: 如果 $g^i(a^0 \circ a^1) \neq g^i(a^0 \circ a^1)$，则 $G_0 \leftarrow G_0 + 1$；否则， $G_1 \leftarrow G_1 + 1$。 ▷ 类型 1 查询 6: 结束 7: 如果 $G_0 > G_1$，则设置 $z_i = 0$；否则，设置 $z_i = 1$。 8: 结束循环 9: 返回 z。

算法 12： Extract。该算法的输入是一个定义域不相交的预言机列表 L、一个字符串$w \in \{0,1\}^{\mathsf{R}}$以及一个失败概率 δ。其目标是在 $\sigma_L^{-1}(w) \in \{0,1\}^\ell$时返回 L是一个变量预言机列表。（参见定义 54 以回顾 σ_L的定义。）

8.3 学习者阶段 2：FindCandidate

现在我们将注意力转向阶段 2。在介绍FindCandidate之前，我们先介绍一个简单的辅助子程序Extract。（该子程序将在稍后的阶段 3 中，对否情形的分析中发挥关键作用；在该情形下， L 可能只是一个变量预言机列表，而非完美预言机。因此，下面我们分析它在这两种情形下的表现，尽管在肯定情形下我们可以假设 L 是完美的。）

Extract 的输入是一个定义域不相交的预言机列表 $L = ((\text{成员查询}(g^1), \mathrm{X}_1) \ldots, (\text{成员查询}(g^\ell), \mathrm{X}_\ell))$，其中 $\ell \in [\mu]$；一个字符串 $w \in \{0,1\}^{\mathsf{R}}$；以及一个失败概率 δ。它的目标是在 L 为变量预言机列表时，返回 $\sigma_L^{-1}(w) \in \{0,1\}^\ell$。（关于 σ_L 的定义，请参见定义 54；并请注意，当 L 为变量预言机列表时，σ_L 是定义良好的。）

引理 64. Extract(L, w, δ) 需要 $O(\mu\log(\mu/\delta))$ 次对 L 中预言机的查询，并满足 s:
1. 当 L 是变量预言机列表时，Extract 以至少 $\sigma_L^{-1}(w)$ 的概率返回 $1-\delta$；2. 当 L 是完美的，Extract 总是返回 $\sigma_L^{-1}(w)$。

证明。 通过检查即可立即得出 Extract 的查询复杂度，并且 $\ell \le \mu$。

当 L 是完美的，每个 g^i 都是文字 $x_{\tau(i)}$ 或 $\overline{x_{\tau(i)}}$，且 $\sigma_L(i) = \tau(i)$。在第 3–6 行的每一轮中，其 $G_{w_{\tau(i)}}$ 都会递增。因此，我们总有 $z_i = w_{\tau(i)}$，且 $z = \sigma_L^{-1}(w)$。

现在，考虑仅承诺 L是变量预言机列表的情形。固定一个 g^i，它与 $x_{\tau(i)}$ 或 $\overline{x_{\tau(i)}}$ 之一的距离为 (1/30)，且 $w_{\tau(i)} = 0$ （即 $\overline{\tau(i)} \in Y_0$； $w_{\tau(i)} = 1$ 的情形是对称的）。那么

$$\mathbf{Pr}\left[g^i(a^0 \circ a^1) = g^i(\overline{a^0} \circ a^1)\right] \le \mathbf{Pr}\left[g^i(a^0 \circ a^1) \neq a_i^0\right] + \mathbf{Pr}\left[g^i(\overline{a^0} \circ a^1) \neq \overline{a_i^0}\right] \le 1/15,$$

最后一个不等式是由 $a^0 \circ a^1$ 和 $a^0 \circ a^1$ 的边缘分布均为均匀分布这一事实得出的。因此，每一轮中， G_0 以至少 $14/15$ 的概率增加 1

算法 13 **FindCandidate**

输入： SAMP(h)，一个长度为 ℓ 的定义域不相交的预言机列表 L，以及 R $\subseteq [n]$。
输出： 要么拒绝，要么返回函数 J $\in$DNF-Approx(ℓ) 的真值表。

令 $A =$ DNF-Approx({v4)
重复 $\eta = 1/(200\xi)$ **次**
    抽取 $\boldsymbol{w} \sim \mathrm{SAMP}(h)$ 并令 $\boldsymbol{z} = \mathbf{Extract}(L, \boldsymbol{w}_{\mathsf{R}}, 0.5)$。
    从 A 中移除所有满足 J($\boldsymbol{z}$) $= 0$ 的 J。 ▷ 这不会查询 MQ(h)
结束
如果 $A = \emptyset$，则拒绝；否则，返回满足 $|\mathsf{J}^{-1}(1)|$ 最小的 J $\in A$。

算法 13： **FindCandidate**。该算法的输入是对以下对象的样本访问： h一份预言机定义域互不相交的列表 L 以及一个集合 R $\subseteq [n]$。在肯定情形下，回顾一下，$\mathfrak{h}$ 是支配函数，定义在R上。该算法的目标是找到一个函数 $\mathfrak{J} \in$ DNF-Approx($|L|$) 使得 $\mathfrak{J}$ 与 $\mathfrak{h}$ 的相对距离很小。

并且 $\boldsymbol{G}_1$ 以至多 $1/15$ 的概率增加 1。在这种情况下，根据标准的 Chernoff 界，并将隐藏常数取足够大，我们可以得到 $\boldsymbol{z}_i = 0$ 以至少 $1 - \delta/\mu$ 的概率成立。于是由并集界可得该引理。 □

FindCandidate 的输入为 SAMP(h)、定义域互不相交的预言机列表以及 R。同样，我们聚焦于肯定情形，并证明当 R 稳定且 L 是满足引理 58条件的**FindCandidate** 会返回一个满足 $\mathfrak{J} \in$ DNF-Approx(ℓ) 的相对距离 $(\mathfrak{h},\mathfrak{J}^{\sigma_L}) \leq \kappa$。

形式上，我们的目标是证明以下结论：

引理 65. 假设 h 是一个 (r, μ)-因子化 DNF，R 是稳定的，L 是一个满足 $\ell = |L| > 0$ 的完美变量预言机列表，并且存在 G $\in$ DNF-Approx(ℓ)，使得 reldist(H, G$^{\sigma_L}$) $\leq \xi$。以至少 $98/100$ 的概率，FindCandidate(SAMP(h) L, R) 返回一个函数 J $\in$DNF-Approx(ℓ)，且满足 reldist(H, J$^{\sigma_L}$) $\leq \kappa$。

FindCandidate(SAMP(h) L，R) 首先将 A 设为由所有候选DNF-Approx(ℓ) 函数构成的集合。随后，它从 SAMP(h)中抽取样本，使用 **Extract** 对其重新映射，并丢弃所有重新映射后的样本不是满足赋值的候选函数。**FindCandidate** 会在 A 变为空集时拒绝；否则，关键的是，它会返回留在 A中的、满足赋值数量最少的候选函数（的真值表）。

我们将证明分为两个引理。首先，我们证明很可能 $\mathfrak{G}$ 会在 A 中存留至最后，因此 **FindCandidate**不会拒绝：

引理 66。在 引理 65的假设下，$\mathfrak{G}$以至少 A 的概率留在 $99/10$ 0.

证明。 已知 reldist(H,G$^{\sigma_L}$) $\leq \xi$，我们有

$$\Pr_{\boldsymbol{y} \sim \mathfrak{h}^{-1}(1)}\left[\mathfrak{G}\left(\sigma_L^{-1}(\boldsymbol{y})\right) = 0\right] = \Pr_{\boldsymbol{y} \sim \mathfrak{h}^{-1}(1)}\left[\mathfrak{G}_{\sigma_L}(\boldsymbol{y}) = 0\right] \leq \xi.$$

已知 R 是稳定的，我们有

$$\Pr_{\boldsymbol{x}\sim h^{-1}(1)}\left[\mathfrak{G}\left(\sigma_L^{-1}(\boldsymbol{x}_{\mathsf{R}})\right)=0\right]\le\xi+(1-\xi)\cdot\Pr_{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)}\left[\mathfrak{G}\left(\sigma_L^{-1}(\boldsymbol{y})\right)=0\right]\le 2\xi.$$

鉴于**Extract** 在 L 完美时总能正确运行，每次循环都会以至多$\mathfrak{G}$的概率从 A 中移除 2ξ。因此，根据针对所有 η 个循环的并集界，$\mathfrak{G}$ 被移除的概率至多为 $2\eta\xi\le 1/100$。

□

接下来我们证明，最有可能的是，在 A 中存活下来的每个函数 J 都满足以下条件：

引理 67。 在引理 65的假设下，以至少 99/100的概率，每个J 在 A 中存活下来的函数都满足

$$\Pr_{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)}\left[\mathsf{J}_{\sigma_L}(\boldsymbol{y})=0\right]\le\frac{\kappa}{10}.$$

证明。 任取一个 J $\in$DNF-Approx(ℓ)，它不满足上述不等式。鉴于 R 是稳定的，

$$\Pr_{\boldsymbol{x}\sim h^{-1}(1)}\left[\mathsf{J}\left(\sigma_L^{-1}(\boldsymbol{x}_{\mathsf{R}})\right)=0\right]\ge(1-\xi)\cdot\Pr_{\boldsymbol{y}\sim\mathfrak{H}^{-1}(1)}\left[\mathsf{J}_{\sigma_L}(\boldsymbol{y})=0\right]\ge\frac{\kappa}{20}.$$

鉴于 **Extract** 总能正确运行，每次循环都会以至少J 的概率移除 $\kappa/20$。因此，J存活的概率至多为

$$\left(1-\frac{\kappa}{20}\right)^{\eta}\le\frac{1}{100\cdot 2^{2\mu r}},$$

使用 $\eta=1/(200\xi)$，以及我们对 ξ 的选择。该引理由对至多 $2^{2\mu r}$ 个函数J 在DNF-Approx（ℓ）上的并集界得出（见 引理 55）。

□

结合以上结果，现在可以证明引理 65：

引理 *65*的证明。 以至少 98/100 的概率，引理 66 和 67的结论同时成立。假设这一情况发生，并令 J 为返回的函数。由于 $\mathfrak{G}\in A$，我们有 $|\mathsf{J}^{-1}(1)|\le|\mathfrak{G}^{-1}(1)|$。因此

$$\left|\mathsf{J}_{\sigma_L}^{-1}(1)\right|\le\left|\mathfrak{G}_{\sigma_L}^{-1}(1)\right|\quad\text{and}\quad\frac{|\mathfrak{H}^{-1}(1)\setminus\mathsf{J}_{\sigma_L}^{-1}(1)|}{|\mathfrak{H}^{-1}(1)|}\le\frac{\kappa}{10}.\tag{20}$$

已知 reldist(H，G_{σ_L}) $\le\xi$，我们有

$$\left|\mathsf{J}_{\sigma_L}^{-1}(1)\right|\le\left|\mathfrak{G}_{\sigma_L}^{-1}(1)\right|\le(1+\xi)|\mathfrak{H}^{-1}(1)|.$$

结合上述结果与公式（20）的第一个不等式，我们有

$$\left|\mathsf{J}_{\sigma_L}^{-1}(1)\setminus\mathfrak{H}^{-1}(1)\right|=\left|\mathsf{J}_{\sigma_L}^{-1}(1)\right|-\left|\mathfrak{H}^{-1}(1)\right|+\left|\mathfrak{H}^{-1}(1)\setminus\mathsf{J}_{\sigma_L}^{-1}(1)\right|\le\left(\xi+\frac{\kappa}{10}\right)|\mathfrak{H}^{-1}(1)|.\tag{21}$$

C结合 公式（20）的第二个不等式和公式（21），我们得到 t

$$\mathsf{reldist}(\mathfrak{H},\mathsf{J}_{\sigma_L})\le\kappa/10+\xi+\kappa/10\le\kappa$$

根据我们对这两个参数的选择使用 $\xi\ll\kappa$。至此，引理得证。

□

最后，我们还需要以下关于**FindCandidate**查询复杂度的简单引理。

引理 68。 **FindCandidate**(SAMP(h) L，R) 进行 poly($(\mu/\xi)\log\mu$) 次调用 SAMP(h)、MQ(h)。

证明。 **FindCandidate** 抽取 $\eta = O(1/\xi)$ 个 SAMP(h) 样本，并对每个样本调用 **Extract** 。根据 引理 64 ，每次调用 **Extract** 会使用 $O(\mu\log(\mu))$ 次对 L中的预言机进行调用。注意，每次这样的调用都可以通过一次 MQ(h)调用来模拟。 □

8.4 学习器阶段3：构建预言机

现在我们终于来考察 **DNFLearner** 的阶段 3。阶段 3 首先运行一个简单的子程序 **CheckLit**，检查 L 是否为变量预言机列表（即，其中每个 g^i 都是 L 的 $(1/30)$-近似文字）。注意，肯定情形下会轻易通过此测试，因为我们假设传递到该阶段的列表 L 是完美的。很快就会清楚，运行 **CheckLit** 实际上是为否情形提供的保障，因为在这种情况下，我们无法对列表 L 的内容作出任何保证。（回想一下，到目前为止对 **DNFLearner** 的所有分析都针对肯定情形；对于否情形，我们只知道 **DNFLearner** 要么已经拒绝，要么已经向阶段 3 传递了一个定义域互不相交的预言机列表 L 。）假设 L 通过了测试，阶段 3 就会构建一对预言机 (MQ*，SAMP*) 并将它们返回给 **DNFLearner**。很快就会清楚，构建这两个预言机不需要任何查询。只需界定每次调用这两个预言机时对 h 的查询次数，并证明它们确实会在两种情形下模拟具有所需性质的函数。

8.4.1 CheckLit子程序

在介绍 **CheckLit** 之前，我们先回顾以下关于 [Bla09], 均匀分布下的合取项测试的结果：

定理 69（ [Bla09] 的定理 1.1）。存在一个单边自适应算法 UniformJunta，它在标准模型下对 k-相关函数进行 ε 测试。UniformJunta(MQ(f) k, ε, δ) 进行 $O((k/\varepsilon + k\log(k))\log(1/\delta))$ 次查询；若 f 是 k-相关函数，则总是接受；若 f 相对于均匀分布与每个 k-相关函数的距离都至少为 ε，则以至少 $1-\delta$ 的概率拒绝 f。

我们证明以下关于**CheckLit**的引理：

引理 70。 **CheckLit**(L)使用 $O(\mu)$次预言机查询，查询的预言机位于 L，并满足以下条件：

1. 若 L 是一个完美变量预言机列表，则 **CheckLit** 总是接受。
2. 若存在某个 g^i 属于 L ，使得 g^i 在均匀分布下与每个文字都相距 $(1/30)$，则 **CheckLit** 以至少 $99/100$ 的概率拒绝。

证明。 查询复杂度由定理 69 得出，因为每次调用 **UniformJunta** 使用$O(1)$ 次查询。如果每个 g^i 属于 L 的都是文字，**UniformJunta** 就绝不会拒绝，因为根据 定理 69，它是单侧的。而且我们总有 $g^i(a) \neq \bar{g}^i(a)$。因此 **CheckLit** 总是接受。

另一方面，假设某个 g^i 属于 L ，且在均匀分布下与每个文字都相距 $(1/30)$。那么它要么

1. 在均匀分布下，与每个 1-串联函数都相差至少 $(1/30)$；或者

算法 14 CheckLit

输入： 对于某个 $\ell \in [\mu]$，一份定义域不相交的预言机列表 $L =$(成员查询(g^1)，X_1)...，(成员查询(g^ℓ)，X_ℓ)。**输出：** 接受或拒绝。

1: **对于** 每个 $i \in [\ell]$ **执行**2: 如果 UniformJunta(成员查询(g^i) 1， 1/30， 1/100) 拒绝，则拒绝。▷ 类型 1 查询 3: 重复 10 次：抽取 $\boldsymbol{a} \sim \{0,1\}^{\mathsf{X}_i}$，如果 $g^i(\boldsymbol{a}) = g^i(\underline{\boldsymbol{a}})$，则拒绝。▷ 类型 1 查询 4: **结束循环**5: 接受。

算法 14： CheckLit。该算法以定义域互不相交的预言机列表 L作为输入，并检查 L是否为变量预言机列表（即，每个 g^i 在 L 中都（1/30）-接近于某个文字）。

2. 与每个文字的距离为 (1/30)，并且与 0-析取子（即常值函数）的距离为 (1/30)。

在情况1下，根据定理 69， UniformJunta以至少 99/100的概率拒绝。在情况 2下，如果 g^i与（1/30）-接近于某个常值函数（不失一般性地，设为全0 函数），那么

$$\begin{aligned}\Pr_{\boldsymbol{a}\sim\{0,1\}^{\mathsf{X}_i}}\left[g^i(\boldsymbol{a}) \neq g^i(\overline{\boldsymbol{a}})\right] &= \Pr_{\boldsymbol{a}\sim\{0,1\}^{\mathsf{X}_i}}\left[g^i(\boldsymbol{a}) = 0 \wedge g^i(\overline{\boldsymbol{a}}) = 1\right] + \Pr_{\boldsymbol{a}\sim\{0,1\}^{\mathsf{X}_i}}\left[g^i(\boldsymbol{a}) = 1 \wedge g^i(\overline{\boldsymbol{a}}) = 0\right] \\ &\leq \Pr_{\boldsymbol{a}\sim\{0,1\}^{\mathsf{X}_i}}\left[g^i(\overline{\boldsymbol{a}}) = 1\right] + \Pr_{\boldsymbol{a}\sim\{0,1\}^{\mathsf{X}_i}}\left[g^i(\boldsymbol{a}) = 1\right] \leq 1/15,\end{aligned}$$

其中最后一行是根据以下事实得出的：在均匀分布下， g^i 与全零函数的距离为 (1/30)，并且 $\boldsymbol{a}$和 $\boldsymbol{a}$ 都是均匀随机抽样得到的。因此，第 3 行未能拒绝 g^i 的概率至多为 $(1/15)^{10} \leq 1/100$。

□

8.4.2 预言机模拟器

我们先介绍一个较简单的 成员查询 模拟器，称为 Sim-MQ-J。给定 R，一个定义域互不相交的预言机列表 $L =$((成员查询(g^1)，X_1).....，(成员查询(g^ℓ)，X_ℓ)) 以及某个 J 的真值表：$\{0,1\}^\ell \to \{0,1\}$，当收到关于 $x \in \{0,1\}^n$ 的成员查询时，它会运行 Extract 以获得 $z \in \{0,1\}^\ell$，并返回$\mathsf{J}(z)$。

我们证明关于Sim-MQ-J的以下引理：

引理 71。 每次调用 Sim-MQ-J(x，R L，J) 都会向 $O(\mu\log(\mu/\kappa))$ 中的预言机发起 L 次查询，并满足以下条件：

1. 当 L 是一个变量预言机列表时，Sim-MQ-J 是一个 κ-精确模拟器，用于成员查询(J^{σ_L})；并且

2. 当 L 是完美的，Sim-MQ-J 模拟成员查询(J^{σ_L}) 时完全正确。

证明。 当 L 是一个变量预言机列表时，根据引理 64 可知，Extract 以至少 $\sigma_L^{-1}(x)$的概率返回 $1-\kappa$。当 L 是完美的，Extract 总是能正确运行。

查询复杂度源于以下事实：Extract 需要 $O(\mu\log(\mu/\kappa))$ 次查询。 □

算法 15 Sim-MQ-J

输入： $x \in \{0,1\}^n$, $\mathsf{R} \subseteq [n]$, 定义域不相交的预言机列表 L 以及定义在 $\{0,1\}^{|L|}$ 上的 J 的真值表。**输出：** 0 或 1。

1: 令 $z = \mathbf{Extract}(L, x, \kappa)$2: 返回 $\mathsf{J}(z)$。

算法 15： Sim-MQ-J。该算法将定义域互不相交的预言机列表 L、函数 J 的真值表 $\{0,1\}^{|L|} \to \{0,1\}$ 作为输入。该算法的目标是为 J 模拟一个 成员查询 σ_L （假设 L 是变量预言机列表）。

算法 16 Sim-SAMP-J

输入： $\mathsf{R} \subseteq [n]$、定义域互不相交的预言机列表 L 以及 J 在 $\{0,1\}^{|L|}$上的真值表。**输出：** $\{0,1\}^n$中的一个字符串。

1: 绘制 $\boldsymbol{y} \sim \{0,1\}^{\mathsf{R}}$ 和 $\boldsymbol{z} \sim \mathsf{J}^{-1}(1)$。2: 令 $\boldsymbol{x} = \mathbf{Extract}(L, w, \kappa)$。3: **对于** $i = 1$ 至 $|L|$ **执行**4: 如果 $\boldsymbol{x}_i \neq \boldsymbol{z}_i$5: **结束循环**6: 返回 $\boldsymbol{y}$。

算法 16： Sim-SAMP-J。该算法以定义域互不相交的预言机列表 L、函数 $\mathsf{J}:\{0,1\}^{|L|} \to \{0,1\}$ 的真值表为输入。该算法的目标是模拟一个 SAMP 预言机，以访问 J^{σ_L} （假设 L 是变量预言机列表）。

对 J^{σ_L} 进行样本访问的模拟 Sim-SAMP-J更为复杂。假设对每个$i \in [\ell]$ g^i，其在均匀分布下与 $\tau(i)$ 或 $\tau(i)$ 相差 $(1/30)$。并令 $\Sigma := \{\sigma(i) : i \in [\ell]\}$。给定 $y \in \{0,1\}^{\mathsf{R}}$和 $z \in \{0,1\}^{\ell}$，令 $y_{\sigma_L, z} \in \{0,1\}^{\mathsf{R}}$表示以下字符串：对于 $j \notin \cup_{i\in[\ell]}\mathsf{X}_i$，设 $(y_{\sigma_L, z})_j = y_j$;

$$\text{the } \mathsf{X}_i \text{ block of } y_{\sigma_L,z} = \begin{cases} y_{\mathsf{R}_i} & \text{if } y_{\tau(i)} = z_i \\ \overline{y_{\mathsf{R}_i}} & \text{if } y_{\tau(i)} \neq z_i \end{cases}$$

对每个 $i \in [\ell]$。关键观察是，当 $\boldsymbol{y} \sim \{0,1\}^{\mathsf{R}}$ and $\boldsymbol{z} \sim \mathsf{J}^{-1}(1)$ 时，则 $\boldsymbol{y}_{\sigma_L, z}$如上所定义的分布与 $\sigma_L(\boldsymbol{z}) \circ \boldsymbol{w}$ with $\boldsymbol{w} \sim \{0,1\}^{\mathsf{R}\setminus\Sigma}$ 完全相同，这恰好对应于从 $\mathsf{J}_{\sigma_L}^{-1}(1)$ 中抽取的样本。

引理 72. 每次调用 Sim-SAMP-J$(\mathsf{R}, L, \mathsf{J})$ 使用 $O(\mu \log(\mu/\kappa))$ 次查询，以访问 L 中的预言机，并满足以下条件：

1. 当 L 是变量预言机列表时，Sim-SAMP-J 是 κ-准确的模拟器，用于 $\mathrm{SAMP}(\mathsf{J}^{\sigma_L})$;

2. 当 L 是完美的，**Sim-SAMP-J** 可以完美地模拟SAMP(J_{σ_L}) 。

证明。查询复杂度由 **Extract** 的查询复杂度以及我们对 $\delta = \kappa$ 的选择得出。

每当 $\boldsymbol{x} = \sigma_L^{-1}(\boldsymbol{y})$ 时，我们知道在第 6 行，算法返回 $\boldsymbol{y}_{\sigma_L,\boldsymbol{z}}$，它对应于 $\mathrm{J}_{\sigma_L}^{-1}(1)$ 中一个均匀随机的字符串（根据上文讨论）。根据 引理 64，若 L 是完美的，则总有 $\boldsymbol{x} = \sigma_L^{-1}(\boldsymbol{y})$；若 L 是变量预言机列表，则 $\boldsymbol{x} = \sigma_L^{-1}(\boldsymbol{y})$ 的概率为 $1-\kappa$。 □

8.5 综合起来

现在我们证明 定理 49 和 定理 50。下面重述这两个定理:

定理 49。（DNFLearner 在是情形下的性能）假设 h 是一个（r, μ）-因子化析取范式，且 R $\subseteq [n]$ 是稳定的（相对于 h）。那么，DNFLearner 以至少 $19/20$ 的概率不会拒绝，并且返回的数对 (MQ*, SAMP*) 是某个函数 J : $\{0,1\}^{\mathrm{R}} \to \{0,1\}$ 的完美预言机模拟器，且 reldist(H, J) $\leq \kappa$，其中 H 是 h 在 R 上的支配函数。

证明。该 t 定理可通过结合引理 58、引理 65、引理 71和引理 72得出 . □

定理 50。（DNFLearner 在否情形下的性能）假设 h 在相对距离意义下与（r, μ）-因子化析取范式相差 ε, 且 R $\subseteq [n]$。那么，以至少 $19/20$ 的概率，DNFLearner 要么拒绝，要么返回（MQ*，SAMP*），它们对于某个函数 J: $\{0,1\}^{\mathrm{R}} \to \{0,1\}$ 是 κ-准确的预言机模拟器，并且对于某个 r-项、 μ-相关函数析取范式 D，有 reldist(J, D) $\leq \kappa$。

证明。DNFLearner 要么（1）在第 3 行返回（MQ($\mathbf{1}_{\mathrm{R}}$)，SAMP($\mathbf{1}_{\mathrm{R}}$)）（此时定理条件成立），要么（2）在阶段 1 或阶段 2 期间拒绝，要么（3）带着一个定义域互不相交的预言机列表 L 和一个函数 J $\in$DNF-Approx($|L|$) 进入阶段 3。根据引理 56，对于任意单射映射 $\sigma : [|L|] \to$ R，有 reldist(J$_\sigma$, D) $\leq \kappa$，其中 D 是某个 r-项、 μ-相关函数析取范式。

根据 引理 70，如果其中任何一个 L 与均匀分布下的任意文字相距 g^i $(1/30)$，那么它通过测试的可能性很低。另一方面，当 L 是一个变量预言机列表时，根据引理 71 和72 ，**Sim-MQ-J** 和 **Sim-SAMP-J** 是 κ-精确模拟器，适用于 J_{σ_L}。 □

剩下的只是界定 **DNFLearner** 的查询复杂度。回顾一下 引理 48:

引理 48。DNFLearner(R, MQ(h), SAMP(h)) 会对 MQ(h)、SAMP(h) 进行 poly($r, \mu, 1/\varepsilon$) 次调用。它要么拒绝，要么返回一对 (MQ*, SAMP*)，其中 MQ* 和 SAMP* 是随机算法，每次调用时都会对 MQ(h) 进行 $O(\mu \log(\mu/\varepsilon))$ 次查询。

证明。 阶段 1 和阶段 2 中对MQ(h)和SAMP(h) 的调用次数，可以利用 引理 57、引理 68 以及我们对参数的选择来界定。在阶段 3 中，**CheckLit** 会对$O(\mu)$个 L中的预言机进行查询，每次查询都通过调用MQ(h)来模拟。

每次调用 **Sim-MQ-J** 和 **Sim-SAMP-J** 的查询复杂度分别由 引理71和 72给出。注意， $\kappa = \varepsilon^2$ 列表中的每个预言机调用都由一次 MQ(h)调用来模拟。 □

9 一致性检查

本节旨在确立我们关于 **ConsCheck**所需的结果，即 引理51、定理52和定理53。

ConsCheck 以集合 R $\subseteq [n]$、预言机 MQ(h) 和 SAMP(h) 作为输入，输入函数 h以及一对用于某个函数的预言机模拟器 MQ* 和 SAMP* J： $\{0,1\}^{\mathrm{R}} \to \{0,1\}$。（回想一下

算法17 ConsCheck

输入： $\emptyset \neq \mathsf{R} \subseteq [n]$，$\varepsilon$，MQ($h$)，SAMP($h$)，MQ* 和 SAMP*。**输出：** 接受或拒绝。

阶段 1：验证 h

1: 重复 $O(1/\varepsilon)$次：2: 从 $\boldsymbol{z} \sim$SAMP(h)中抽取；若MQ*($\boldsymbol{z}_{\mathsf{R}}$) $= 0$，则拒绝。

3: 从 $\boldsymbol{v} \sim$SAMP*中抽取100个点。4: 若对于第3行抽取的任意点 $\boldsymbol{v}$，MQ(h)($\boldsymbol{z}_{\mathsf{R}} \circ \boldsymbol{v}$) $= 0$，则拒绝。

▷ 2类查询

▷ Type 2 Query

5: 重复 $O(1/\varepsilon)$次：6: 从 $\boldsymbol{z} \sim$ SAMP(h)和 $\boldsymbol{v} \sim$ SAMP*中抽取；若MQ(h)($\boldsymbol{z}_{\mathsf{R}} \circ \boldsymbol{v}$) $= 0$，则拒绝。

阶段 2：这是一个合取吗?

令 $\Gamma : \{0,1\}^{\mathsf{R}} \to \{0,1\}$ 为如下函数：

$$\Gamma(\alpha) = \begin{cases} 1 & \text{if } \mathbf{Pr}_{\boldsymbol{z} \sim \mathsf{J}^{-1}(1)}[h(\alpha \circ \boldsymbol{z})] \geq 0.9 \\ 0 & \text{otherwise} \end{cases}$$

7: 令 MQ$' \leftarrow$ **Sim-MQ-Γ**（$\cdot$，MQ（h）、SAMP*）和 SAMP$' \leftarrow$ **Sim-SAMP**（SAMP（h）、R）。8: 运行 **ConjTest**（MQ$'$，SAMP$'$，$\varepsilon/10$）并返回相同的答案。

算法17：ConsCheck。该算法以 $\mathsf{R} \subseteq [n]$、$\varepsilon > 0$作为输入，并获得对 h 以及预言机 MQ*、SAMP* 的预言机访问，其中这些预言机对应某个函数 $\mathsf{J}: \{0,1\}^{\mathsf{R}} \to \{0,1\}$。该算法检查是否存在某个合取 C，使得 f 与 $C(x_R) \wedge \mathsf{J}(x_{\mathsf{R}})$之间的相对误差较小。特别地，该算法将始终用于 J 接近某个 DNF r-项、μ-相关函数 DNF 的情形；因此，如果 f 与每个 (r、μ)–因式分解析取范式之间的相对距离都至少为 ε，则该算法将以高概率拒绝。

在肯定情形下，它们是J的完美模拟器；在否情形下，它们是 κ-准确模拟器，适用于J，且误差为 $\kappa := \varepsilon^2$。）回顾一下本节将要证明的关于**ConsCheck** 的结果：

算法 1Sim-SAMP

输入： 某个函数 h 的 SAMP(h)：$\{0,1\}^n \to \{0,1\}$ 和 $\emptyset \neq \mathsf{Y} \subseteq [n]$。**输出：** $\{0,1\}^{\mathsf{Y}}$中的一个点。

1: 抽取 $\boldsymbol{z} \sim$SAMP(h)。

2: 返回 $\boldsymbol{z}_{\mathsf{Y}}$。

引理51。 **ConsCheck**(R,成员查询(h),SAMP(h),成员查询*,SAMP*) 至多调用 $O(1/\varepsilon)$ 次成员查询(h),SAMP(h),成员查询* 和SAMP*。最终，它要么接受，要么拒绝。

定理 52。 (ConsCheck 的 **ConsCheck** 在“是”情况下的性能）假设 h 是一个 (r,μ)-因子化析取范式，R 是稳定的，并且 (成员查询*，SAMP*) 是某个函数的完美模拟器 $\mathsf{J}: \{0,1\}^{\mathsf{R}} \to \{0,1\}$

算法 18 模拟成员查询-Γ

输入： $\alpha \in \{0,1\}^{\overline{\mathsf{R}}}$、成员查询($h$) 和 SAMP*。**输出：** 拒绝或 1。

1: 重复抽取 $\boldsymbol{w}$ ~SAMP* 次，共 $O\log(1/\varepsilon)$ 次。
2: 如果对上一步抽取的所有 $\boldsymbol{w}$，MQ(h)($\alpha \circ \boldsymbol{w}$) $= 1$，则返回 1。 ⊳2类查询
3: 否则，停止并拒绝。

算法 18: **模拟成员查询-Γ**。一种用于模拟函数 Γ 的**ConsCheck** 中定义的 成员查询预言机。

当 相对距离($\mathfrak{H}$,$\mathfrak{J}$) $\leq \kappa$. 那么 **ConsCheck** 以至少 $39/40$ 的概率接受。

定理53。（ConsCheck 在“否”情形下的性能）假设 h 与 (r,μ)-因子化析取范式的相对距离至少为 ε，R $\subseteq [n]$，并且 (成员查询*, SAMP*) 是某个函数 J : $\{0,1\}^{\mathsf{R}} \to \{0,1\}$ 的 κ-准确预言机模拟器，其中 reldist(J, D) $\leq \kappa$，且 D 是某个 r-项、 μ-相关函数析取范式。那么，ConsCheck 以至少 $19/20$的概率拒绝。

ConsCheck 见 算法17。我们简要给出引理51的证明。为此，我们回顾**ConjTest** 在 [CPPS25a] 中的查询复杂度：

定理 73。 **ConjTest**(成员查询(f)、SAMP(f) ε)进行 $O(1/\varepsilon)$次查询，查询对象为 成员查询(f)和 SAMP(f)。

引理51的证明。阶段 1 进行 $O(1/\varepsilon)$次查询。在阶段 2，ConjTest 使用 $O(1/\varepsilon)$ 次成员查询′和 SAMP′调用。SAMP′ 使用 Sim-SAMP(SAMP(h), R)，因此每次调用它都会使用一次 SAMP(h)调用。成员查询′ 使用 Sim-MQ-Γ(·, 成员查询(h), SAMP*)，因此每次成员查询′ 调用分别会使用 $O(\log(1/\varepsilon))$ 次成员查询(h)和 SAMP*调用。因此，预言机调用总次数为 $O(\log(1/\varepsilon)/\varepsilon)$。 □

接下来要证明定理 52和定理53。在此之前，我们先记录**ConjTest** 算法的一个“鲁棒性”性质，后面会用到。

9.1 ConjTest的鲁棒性

我们回顾一下**ConjTest** 在 [CPPS25a]中的一些结果。第一个结果指出，如果**ConjTest** 在一个合取上运行，即使采样预言机存在缺陷，它也会以概率 1 接受（只要它返回的都是 f的满足赋值）：

定理 74 ([CPPS25a]中的定理 27）。设 f： $\{0,1\}^n \to \{0,1\}$ 是一个合取。设SAMP*(f) 是一个预言机，每次调用时总会返回 $f^{-1}(1)$ 中的某个字符串。于是，**Con-jTest**在 成员查询(f) 和 SAMP*(f) 上运行时，以概率 1 接受 f。

第二个结果表明，若 **ConjTest**在函数 f 上运行，而该函数相对误差远离每一个合取，那么只要使用 MQ(f) 预言机，以及一个“精度尚可”的 SAMP(f) 预言机（也就是说，它对每个满足赋值都赋予非平凡的权重），就会以高概率拒绝：

定理 75 （[CPPS25a]的定理 28）。 假设 $f\colon \{0,1\}^n \to \{0,1\}$ 对每个合取 f'都满足 相对距离$(f, f') > \varepsilon$。令 SAMP*(f)为 f的采样预言机，使其底层分布 $\mathcal{D}$ （即 SAMP*(f) 生成样本所依据的分布）满足

$$\Pr_{\boldsymbol{x}\sim\mathcal{D}}\left[\boldsymbol{x} = z\right] \geq \frac{1}{20|f^{-1}(1)|}, \quad \text{for every } z \in f^{-1}(1).$$

此时 **ConjTest**在 MQ(f) 和 SAMP*(f)上运行时，以至少 f 的概率拒绝 $99/100$。

对于我们的目的，还需要关于 **ConjTest**性能的“鲁棒性”保证：当其在采样不完美且成员查询预言机不完美的合取上运行时，该保证超越了 定理 74所提供的保证。我们将在引理 76中确立所需的保证；为完整起见，我们在**ConjTest** 算法的 附录 C中重述该算法：

引理 76。 设 $C\colon \{0,1\}^n \to \{0,1\}$ 为一个合取。假设预言机 SAMP*(C) 总是返回 $C^{-1}(1)$中的某个字符串。那么，**ConjTest** 在 SAMP*(C) 和 MQ*(C)上的运行具有以下性质：(1) 它仅在具有 $C(x) = 1$的点上调用 MQ*(C)

。(2) 如果所有这些查询均由 1回答，则 **ConjTest** 以概率1接受

证明。 通过检查 **ConjTest** (参见 [CPPS25a], 的算法 2 和注记10，或参见 附录 C），我们可以看到，**ConjTest** 当且仅当它进行的每次 成员查询*(C) 都得到 1时才接受；这就证明了（2）。

现在我们转而证明（1）。首先，注意到（再次通过检查 ConjTest）ConjTest 是非自适应的，因此它进行的成员查询*(C) 不取决于它是在成员查询(C) 还是成员查询*(C) 下运行。因此，只需证明：给定预言机成员查询(C) 和一个仅返回 C^{-1}(1) 中字符串的预言机 SAMP*(C)，ConjTest 只会在满足 $C(x) = 1$ 的点 x 上调用成员查询(C)。

反设存在某个合取 g，使得 ConjTest 在成员查询(g) 和 SAMP*(g)（其仅返回 g^{-1}(1) 中的字符串）下运行时，以非零概率在某个满足 $g(z) = 0$ 的点 z 上调用成员查询(g)。由于 ConjTest 当且仅当它进行的每次成员查询(g) 都得到 1 时才接受，这意味着 ConjTest 以非零概率拒绝。这与定理 74 矛盾，因此引理得证。 □

9.2 肯定情形：证明定理 52

回顾一下定理 52的假设：R 是稳定的，并且 (MQ*，SAMP*) 是某个函数的完美模拟器 J：$\{0,1\}^{\mathsf{R}} \to \{0,1\}$ 且 $\mathsf{reldist}(\mathfrak{H}, \mathsf{J}) \leq \kappa$.

由于 $\kappa \ll 1/2$，根据对称性（见 引理 3），我们有 $\mathsf{reldist}(\mathsf{J}, \mathfrak{H}) \leq 2\kappa$。根据定义，由于 R是稳定的，我们有存在 $u \in \{0,1\}^{\mathsf{V}}$，其中 $\mathsf{V} := \mathsf{R} \cap (\mathsf{S} \cup \mathsf{var}(H))$，使得

$$\Pr_{\boldsymbol{z}\sim h^{-1}(1)}\left[\boldsymbol{z}_{\mathsf{V}} \neq u\right] \leq \xi,$$

这意味着

$$\Pr_{\boldsymbol{z}\sim h^{-1}(1)}\left[h|_{\boldsymbol{z}_{\mathsf{R}}} \neq \mathfrak{H}\right] \leq \xi.$$

特别地，每当我们以至少 $1-\xi$ 的概率对 $\boldsymbol{z} \sim h^{-1}$(1) 进行采样时，就有 $h|_{\boldsymbol{z}_{\mathsf{R}}} \equiv \mathsf{H}$ 和$\boldsymbol{z}_{\mathsf{R}} \backsim \mathsf{H}^{-1}$(1)。根据 reldist(H，J) 和 reldist(J，H) 的界，我们将使用以下结论：

$$\Pr_{\boldsymbol{w}\sim\mathsf{J}^{-1}(1)}\left[\mathfrak{H}(\boldsymbol{w}) = 1\right] = O(\kappa) \quad \text{and} \quad \Pr_{\boldsymbol{z}\sim\mathfrak{H}^{-1}(1)}\left[\mathsf{J}(\boldsymbol{z}) = 1\right] = O(\kappa).$$

牢记这些后，我们首先证明以下结论：

引理 77. ConsCheck在第 1–6 行拒绝的概率至多为 $1/80$.

证明。 在第 1–6 行之间，ConsCheck 抽取 $O(1/\varepsilon)$ 个样本 $z \sim \mathrm{SAMP}(h)$。根据并集界，以至少 $1-O(\xi/\varepsilon)$ 的概率，每个 z 都满足 $h|_{z_R} = \mathfrak{h}$，因此 z_R 是从 $\mathfrak{h}^{-1}(1)$ 中抽取的。

因此，假设上述条件成立。现在考察在第 2、4 和 6 行拒绝的概率。每次执行第 2 行时，如果 $J(z_R) = 0$ 其中 $z_R \sim H^{-1}(1)$，算法就会拒绝，因此这种情况发生的概率至多为 $O(\kappa)$。在第 4 行，如果 $H(v) = 0$ 而 $v \sim J^{-1}(1)$，算法就会拒绝，这种情况发生的概率也至多为 $O(\kappa)$。同理，每次执行第 6 行时，算法拒绝的概率至多为 $O(\kappa)$。因此，根据并集界，算法在这些行拒绝的概率至多为 $O(\kappa/\varepsilon)$。

总的来说，算法在第 1–6 行拒绝的概率为 $O(\kappa/\varepsilon) + O(\xi/\varepsilon) \le 1/80$。 □

现在，令 $C^* : \{0,1\}^R \to \{0,1\}$ 为由 $C^*(\alpha) = 1$ 当且仅当 $\alpha_V = u$ 定义的合取。现在我们证明，第 8 行以较小概率拒绝。

Lem引理 78。 第 12 行调用 ConjTest以至多 $1/80$ 的概率拒绝。

证明。 在此阶段，ConjTest 将调用 $O(1/\varepsilon)$ 次 成员查询′ 和 SAMP′ (参见定理 73）。由于我们使用 **模拟成员查询-Γ** 和 **Sim-SAMP** 来模拟预言机，运行 ConjTest将使用 $O(\log(1/\varepsilon)/\varepsilon)$ 个来自 $\mathrm{SAMP}(h)$ 和 SAMP^*的样本。

因此，以 $O(\xi \cdot \log(1/\varepsilon)/\varepsilon) \le 1/160$ 的失败概率，每个样本 $w \sim \mathrm{SAMP}(h)$ 都满足$w_V = u$。此时，我们有 $\mathrm{SAMP}^*(\Gamma) = \text{Sim-SAMP}(\mathrm{SAMP}(h), R)$ 只会返回 $(C^*)^{-1}(1)$ 中的字符串。因此，根据引理 76，ConjTest 只查询满足 $C^*(\alpha) = 1$ 的点 α；如果成员查询′ 对这些点的回答都为 1，ConjTest 就接受。

现在注意，对于任意满足 $C^*(\alpha) = 1$ 的 α，MQ′ 都会返回 1，除非存在某个$w \sim \mathrm{SAMP}^*$（这与 SAMP(J) 相同），使得 $h(\alpha \circ w) = 0$。由于 $\alpha_V = u$，这等价于 $H(w) = 0$，发生的概率为 $O(\kappa)$。因此，对至多 $O(\log(1/\varepsilon)/\varepsilon)$ 个样本 $w \sim \mathrm{SAMP}^*$ 应用并集界可知，概率至少为 $1 - 1/160$ 时，MQ′的每次查询都会得到 1，且 ConjTest 接受。

因此，ConjTest 以至少 $1 - 1/80$ 的概率接受。 □

最后，我们可以证明定理 52：

定理 *52* 的证明。 根据 引理 77，ConsCheck 在第 1–6 行拒绝的概率至多为$1/80$；根据 引理 78，ConsCheck 在第 8 行拒绝的概率至多为 $1/80$。因此，应用并集界可得，ConsCheck 拒绝的概率至多为 $1/40$. □

9.3 否情形：定理定理53

回顾一下，在否情形中，h 与每个（r, μ）因子化 DNF 的相对距离至少为 ε，并且对于某个函数 $J : \{0,1\}^R \to \{0,1\}$，$(\mathrm{MQ}^*, \mathrm{SAMP}^*)$ 是精度为 κ 的预言机模拟器，其中对于某个 r- 项、 μ- 相关函数析取范式 D[10]，$\mathrm{reldist}(J, \mathcal{D}) \le \kappa$。鉴于 $\kappa = \varepsilon^2$ 远小于 ConsCheck 在 MQ^* 和 SAMP^* 上的调用次数（引理51），不失一般性，我们可以假设它们是 MQ(J) 和 SAMP(J) 的完美模拟器，并在此假设下证明 ConsCheck 以至少 0.96 的概率拒绝。

[10]请注意，这意味着 D 至多依赖于 μ 个变量。

为了分析 ConsCheck，我们引入一个新函数 $\gamma\colon \{0,1\}^n \to \{0,1\}$ 定义如下

$$\gamma(z) = \begin{cases} 1 & \text{if } \Gamma(z_{\overline{\mathsf{R}}}) = 1 \text{ and } \mathsf{J}(z_{\mathsf{R}}) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

wh在 $\Gamma\colon \{0,1\}^{\overline{\mathsf{R}}} \to \{0,1\}$ 阶段 2 开始时定义于 ConsCheck .

现在进入形式证明。我们首先证明，如果 h 有“许多”并非 γ的满足赋值，那么 ConsCheck 以高概率拒绝:

引理 79。 假设

$$|h^{-1}(1) \setminus \gamma^{-1}(1)| \geq \frac{\varepsilon |h^{-1}(1)|}{20}.$$

那么 ConsCheck在第 1–4 行拒绝的概率至少为 $99/100$。

证明。由于 $|h^{-1}(1) \setminus \gamma^{-1}(1)| \geq \varepsilon |h^{-1}(1)|/20$，每当我们在第 2 行抽取 $z \sim h^{-1}(1)$ 时，以至少 $\varepsilon/20$ 的概率，我们有 $\gamma(z) = 0$。因此，失败概率至多为 $(1-\varepsilon/20)^{O(1/\varepsilon)} \leq 1/200$，我们便能在第 1–4 行的某次迭代中获得这样的样本 z^*。对于这个 z^*，如果 $\mathsf{J}(z^*_{\mathsf{R}}) = 0$，那么算法会在第 2 行拒绝。否则，如果 $\mathsf{J}(z^*_{\mathsf{R}}) = 1$，根据公式（22），我们必有 $\Gamma(z^*\,\mathsf{R}) = 0$；但在此情况下，ConsCheck 以至少 $1 - (0.9)^{O(1)} \geq 1 - 1/200$ 的概率在第 4 行拒绝。因此，算法在第 1–4 行期间拒绝的概率至少为 $1 - 1/100$。 □

接下来，我们证明：如果 h 的几乎所有满足赋值都是 γ 的满足赋值，那么 Γ^{-1} 的每个满足赋值都“很有可能”作为 h 的均匀随机满足赋值的 R 部分出现:

引理 80。假设 设 $|h^{-1}(1) \setminus \gamma^{-1}(1)| \leq \varepsilon |h^{-1}(1)|/20.$。对于任意满足 $\alpha \in \Gamma^{-1}(1)$ 的 $\alpha \in \{0,1\}^{\mathsf{R}}$ ，

$$\Pr_{z \sim h^{-1}(1)} \left[z_{\overline{\mathsf{R}}} = \alpha \right] \geq \frac{1}{2|\Gamma^{-1}(1)|}.$$

证明。根据假设，我们有 $|\gamma^{-1}(1)| \geq |h^{-1}(1)|(1 - \frac{\varepsilon}{20})$，并且由于 $|\gamma^{-1}(1)| = |h^{-1}(1)| \setminus |\Gamma^{-1}(1)|$，我们有 $1.1 \cdot |\mathsf{J}^{-1}(1)| \cdot |\Gamma^{-1}(1)| \geq |h^{-1}(1)|$。因此，考虑任意满足 $\Gamma(\alpha) = 1$ 的 $\alpha \in \{0,1\}^{\overline{\mathsf{R}}}$。根据 Γ 的定义，我们有

$$|\{z \in \{0,1\}^n \mid \mathsf{J}(z_{\mathsf{R}}) = 1 \wedge z_{\overline{\mathsf{R}}} = \alpha\} \cap h^{-1}(1)| \geq 0.9 |\mathsf{J}^{-1}(1)|.$$

这意味着

$$\Pr_{z \sim h^{-1}(1)} [z_{\overline{\mathsf{R}}} = \alpha] \geq \frac{0.9 |\mathsf{J}^{-1}(1)|}{|h^{-1}(1)|} \geq \frac{0.9}{1.1 |\Gamma^{-1}(1)|} \geq \frac{1}{2|\Gamma^{-1}(1)|}. \quad \square$$

接下来，我们证明，即使 h 的几乎所有满足赋值也都是 γ 的满足赋值，但如果 γ 中有“很多”满足赋值不是 h 的满足赋值，那么ConsCheck 很可能会拒绝:

引理 81。 假设 ConsCheck 执行到第 5 行时具有 $|h^{-1}(1) \setminus \gamma^{-1}(1)| \leq \varepsilon |h^{-1}(1)|/20.$ 如果

$$|\gamma^{-1}(1) \setminus h^{-1}(1)| \geq \frac{\varepsilon |h^{-1}(1)|}{20},$$

那么， h 会在第 6 行被拒绝，概率至少为 $99/100$。

证明。 假设 $|\gamma^{-1}(1) \setminus h^{-1}(1)| \geq \varepsilon |h^{-1}(1)|/20$。这可以改写为

$$\Pr_{\boldsymbol{z} \sim \gamma^{-1}(1)}[h(\boldsymbol{z}) = 0] \geq \frac{\varepsilon |h^{-1}(1)|}{20 |\gamma^{-1}(1)|}.$$

如果 $|\gamma^{-1}(1)| \geq 2|h^{-1}(1)|$，那么显然有 $\Pr_{\boldsymbol{z} \sim \gamma^{-1}(1)}[h(\boldsymbol{z}) = 0] \geq 1/2 \geq \varepsilon/40.$。否则，

$$\Pr_{\boldsymbol{z} \sim \gamma^{-1}(1)}[h(\boldsymbol{z}) = 0] \geq \frac{\varepsilon |\gamma^{-1}(1)|}{40 |\gamma^{-1}(1)|} = \varepsilon/40,$$

因此，两种情况下都有 $\Pr_{\boldsymbol{z} \sim \gamma^{-1}(1)}[h(\boldsymbol{z}) = 0] \geq \varepsilon/40$.

现在注意，以下过程会生成一个均匀随机样本 $\boldsymbol{w} \sim \gamma^{-1}(1)$：独立地抽取 $\boldsymbol{\alpha} \sim \Gamma^{-1}(1)$ 和 $\boldsymbol{v} \sim \mathsf{J}^{-1}(1)$，并返回 $\boldsymbol{w} = \boldsymbol{\alpha} \circ \boldsymbol{v}$。因此，根据引理 80，我们可以得出，对于任意固定的 $w \in \gamma^{-1}(1)$，以下过程以至少 $1/(2|\gamma^{-1}(1)|)$ 的概率返回 w：独立地抽取 $\boldsymbol{z} \sim h^{-1}(1)$ 和 $\boldsymbol{v} \sim \mathsf{J}^{-1}(1)$，并返回 $\boldsymbol{w} = \boldsymbol{z}_{\mathsf{R}} \circ \boldsymbol{v}$。由此可知，每次执行第 8 行时，都有 $\Pr[h(\boldsymbol{z}_{\mathsf{R}} \circ \boldsymbol{v}) = 0] \geq \varepsilon/80$。因此，算法在第 8 行拒绝的概率至少为

$$1 - (1 - \varepsilon/80)^{O(1/\varepsilon)} \geq 1 - 1/100. \quad \square$$

下面这个引理大致说明，如果 h 以高概率到达第 7 行，那么 Γ 必定与每个合取都相距甚远（稍后将用它来证明，**ConjTest** 以高概率在第 8 行拒绝）：

引理 82。 假设对于每个(r, μ)-因子化析取范式 g，$\mathrm{reldist}(h, g) > \varepsilon$，并且以下条件也都成立：

(a) 对某个 r-项、 μ-相关函数析取范式 D，有 $\mathrm{reldist}(\mathsf{J},\mathsf{D}) \leq \varepsilon/10$。

(b) $|h^{-1}(1) \setminus \gamma^{-1}(1)| \leq \varepsilon |h^{-1}(1)|/20.$

(c) $|\gamma^{-1}(1) \setminus h^{-1}(1)| \leq \varepsilon |h^{-1}(1)|/20.$

那么，对于每个合取 C，$\mathrm{reldist}(\Gamma, C) > \varepsilon/10$。

证明。 反设对于某个定义在 $\{0,1\}^{\mathsf{R}}$ 上的合取 C，$\mathrm{reldist}(\Gamma, C) \leq \varepsilon/10$。我们利用条件 (a)—(c) 论证，存在一个$(r, \mu)$-因子化析取范式 g，使得 $\mathrm{reldist}(h, g) < \varepsilon$，这便与引理中 $\mathrm{reldist}(h, g) > \varepsilon.$ 的假设矛盾。

首先，利用 (b) 和 (c)，我们有相对距离$(h, \gamma) \leq \varepsilon/10$。

证明分两步进行：定义关于 $\{0,1\}^n$ 的两个布尔函数 ψ 和 g，其中 g将是一个 k-项析取范式。我们将证明以下不等式：

$$\text{(i) } \mathrm{reldist}(\gamma, \psi) \leq \varepsilon/10, \quad \text{and} \quad \text{(ii) } \mathrm{reldist}(\psi, g) \leq \varepsilon/10.$$

结合 相对距离$(h, \gamma) \leq \varepsilon/10$，由相对距离的近似三角不等式（引理 4）可得 相对距离$(h, g) < \varepsilon$；因此，只需证明 (i) 和 (ii)。

我们先证明 (i)。根据条件 (a)，存在一个 r-项、 μ-相关的析取范式 D，使得 $\mathrm{reldist}(\mathsf{J},\mathsf{D}) \leq \varepsilon/10$。我们的第一个函数 ψ 定义在 $\{0,1\}^n$ 上，如下：

$$\psi(z) := \Gamma(z_{\overline{\mathsf{R}}}) \wedge \mathfrak{D}(z_{\mathsf{R}}).$$

注意，对于任意 $\alpha \in \{0,1\}^{\overline{\mathsf{R}}}$，若 $\Gamma(\alpha) = 0$，则 $\gamma\restriction_\alpha \equiv \psi\restriction_\alpha \equiv 0$。此外，若 $\Gamma(\alpha) = 1$，则 $\gamma\restriction_\alpha \equiv \mathsf{J}$，而 $\psi\restriction_\alpha \equiv \mathsf{D}$；此时 $\mathsf{reldist}(\gamma\restriction_\alpha, \psi\restriction_\alpha) = \mathsf{reldist}(\mathsf{J},\mathsf{D}) \le \varepsilon/10$。因此我们有：

$$\begin{aligned}\mathsf{reldist}(\gamma,\psi) &= \frac{1}{|\gamma^{-1}(1)|}\sum_{\alpha:\Gamma(\alpha)=1}\left|\left\{w \in \{0,1\}^{\mathsf{R}} : \gamma(\alpha\circ w) \neq \psi(\alpha\circ w)\right\}\right| \\ &= \frac{1}{|\gamma^{-1}(1)|}\sum_{\alpha:\Gamma(\alpha)=1}\mathsf{reldist}(\gamma\restriction_\alpha,\psi\restriction_\alpha)\cdot\left|(\gamma\restriction_\alpha)^{-1}(1)\right| \\ &\le \frac{\varepsilon}{10}\cdot\frac{1}{|\gamma^{-1}(1)|}\sum_{\alpha:\Gamma(\alpha)=1}\left|\gamma\restriction_\alpha^{-1}(1)\right|.\end{aligned}$$

由于 $|\gamma^{-1}(1)| = \sum_\alpha |(\gamma\restriction_\alpha)^{-1}(1)|$，我们得到相对距离 $\mathsf{reldist}(\gamma,\psi) \le \varepsilon/10$，从而得到 (i)。

接下来讨论 (ii)。首先回顾一下，C 是 $\{0,1\}^{\mathsf{R}}$ 上的一个合取，满足 $\mathsf{reldist}(\Gamma, C) \le \varepsilon/10$。最后一个函数 $g\colon\{0,1\}^n \to \{0,1\}$ 定义如下：

$$g(z) := C(z_{\overline{\mathsf{R}}}) \wedge \mathfrak{D}(z_{\mathsf{R}})$$

（即在 ψ 的定义中，用 $C(z_{\mathsf{R}})$ 替换 $\Gamma(z_{\mathsf{R}})$）。注意，由于 D 是一个 r-项、μ-相关函数析取范式，g 是一个 (r,μ)-因子化析取范式。下面我们对 $\mathsf{reldist}(\psi, g)$ 作出界定。

注意，对于每个 $\alpha \in \{0,1\}^{\overline{\mathsf{R}}}$，如果 $\Gamma(\alpha) = C(\alpha)$，那么 $g\restriction_\alpha \equiv \psi\restriction_\alpha$。否则，$g\restriction_\alpha$ 或 $\psi\restriction_\alpha$ 中有一个是 D，而另一个是全 0 函数。因此我们有

$$\mathsf{reldist}(\psi,g) = \frac{1}{\left|\psi^{-1}(1)\right|}\cdot\left|\left\{\alpha \in \{0,1\}^{\overline{\mathsf{R}}} : \Gamma(\alpha) \neq C(\alpha)\right\}\right|\cdot\left|\mathfrak{D}^{-1}(1)\right|.$$

由于 $|\psi^{-1}(1)| = |\Gamma^{-1}(1)| \times |\mathfrak{D}^{-1}(1)|$，我们得到

$$\mathsf{reldist}(\psi,g) = \frac{|\{\alpha \in \{0,1\}^{\overline{\mathsf{R}}} : \Gamma(\alpha) \neq C(\alpha)\}|}{|\Gamma^{-1}(1)|} = \mathsf{reldist}(\Gamma, C) \le \frac{\varepsilon}{10}, \quad \text{giving (ii).}$$

引理的证明至此完成。 □

根据定理 73，当**ConsCheck** 在阶段 2 中以距离 $\varepsilon/10$运行时，它将进行 $O(1/\varepsilon)$ 次查询，查询对象为 MQ′，SAMP′。令 c_0/ε 为查询次数的一个上界。我们需要以下简单引理来确立MQ′的正确性：

引理 83。对于任意 $\alpha \in \{0,1\}^{\overline{\mathsf{R}}}$，MQ′(α) 满足

$$\mathbf{Pr}\left[\mathrm{MQ}'(\alpha) \text{ does not reject and returns } 1-\Gamma(\alpha)\right] \le \frac{\varepsilon}{100c_0}.$$

证明。回顾可知，我们有 MQ′ ← Sim-MQ-Γ(·, MQ(h), SAMP*, ε)，其中 SAMP* 仅仅是 SAMP(J)。我们首先假设 $\Gamma(\alpha) = 0$；这意味着 $\mathbf{Pr}_{z\sim \mathsf{J}^{-1}(1)}[h(\alpha\circ z)] \le 0.9$。因此，通过选取足够大的隐藏常数，算法 18 在不拒绝 h 的情况下返回 1 的概率至多为 $(0.9)^{O(\log(1/\varepsilon))} \le \frac{\varepsilon}{100c_0}$。另一种可能性是 $\Gamma(\alpha) = 1$；在这种情况下，算法要么返回 1，要么拒绝。 □

引理 84。假设对于每个合取 相对距离$(\Gamma,\ C) \ge \varepsilon/10$都成立，且 C满足 $|h^{-1}(1) \setminus \gamma^{-1}(1)| \le \varepsilon|h^{-1}(1)|/20$。那么，如果 **ConsCheck** 执行到第 7 行，则第 8 行对**ConjTest**的调用以至少 $49/50$的概率拒绝。

证明。令 $\mathrm{SAMP}' = \textbf{Sim-SAMP}(\mathrm{SAMP}(h), \mathsf{R}), \neg$ 因此根据 引理 80，对于每个 $\alpha \in \Gamma^{-1}(1)$，有

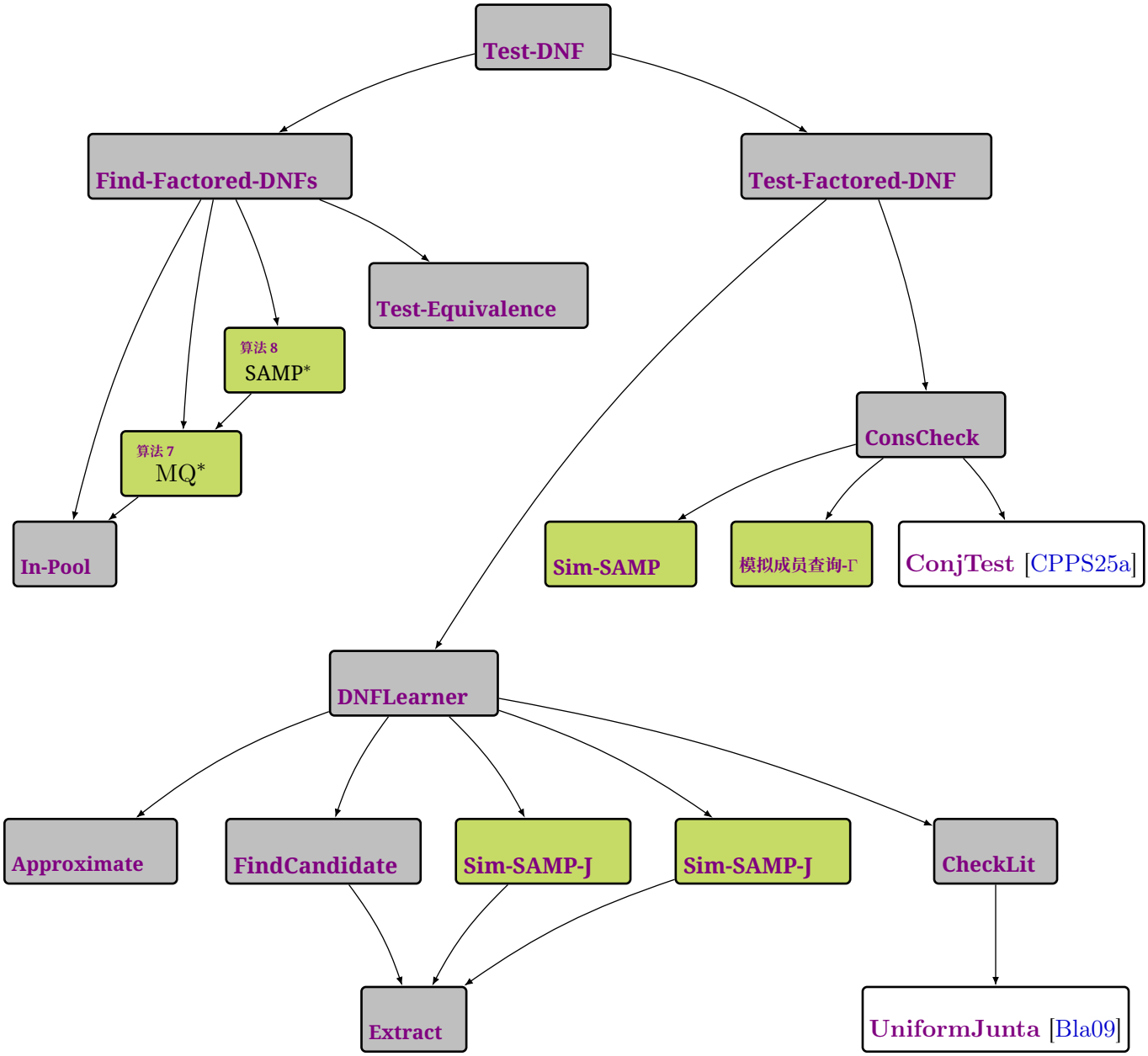
$$\mathbf{Pr}\left[\mathrm{SAMP}' \text{ returns } \alpha\right] \geq \frac{1}{2|\Gamma^{-1}(1)|}.$$

如果所有由 c_0/ε 个（至多）成员查询′ ConjTest 所执行的 **查询**都得到了正确回答，那么根据 定理 75 **ConjTest** 以至少 Γ 的概率拒绝 $99/100$。根据 引理 83和并集界，以至少 $99/100$的概率，成员查询′ 会决定拒绝，或返回正确答案。因此，**ConsCheck**第 8 行以至少 $49/50$的概率拒绝。□

最后，我们可以把所有部分整合起来。

定理53的证明。若 $|h^{-1}(1) \setminus \gamma^{-1}(1)| \geq \varepsilon|h^{-1}(1)|/20$，则根据引理 79，ConsCheck 在第 1–4 行以至少 $99/100$ 的概率拒绝，因此我们可以假设 ConsCheck 在 $|h^{-1}(1) \setminus \gamma^{-1}(1)| \leq \varepsilon|h^{-1}(1)|/20.$ 的情况下到达第 5 行。在这种情况下，若 $|\gamma^{-1}(1) \setminus h^{-1}(1)| \geq \varepsilon|h^{-1}(1)|/20$，则根据引理 81，ConsCheck 在第 6 行以至少 $99/100$ 的概率拒绝。因此我们可以假设同时有 $|h^{-1}(1) \setminus \gamma^{-1}(1)|, |h^{-1}(1) \setminus \gamma^{-1}(1)| \leq \varepsilon|h^{-1}(1)|/20$。于是，根据引理 82，对于每个合取 C，都有 $\mathrm{reldist}(\Gamma, C) > \varepsilon/10$，因此根据引理 84，$h$ 以至少 $49/50$ 的概率在第 9 行被拒绝。综合所有失败概率可得，ConsCheck 以至少 $1 - (1/100 + 1/100 + 1/50) = 0.96.$ 的概率拒绝。□

图 2：我们算法的结构Test-Factored-DNF。从算法A指向算法B的箭头表示A正在调用B。绿色算法用于模拟整个SAMP和成员查询预言机Test-Factored-DNF。白色算法取自先前的文献工作。



参考文献

[Bla08] Eric Blais. Junta测试的改进界。载于 RANDOM 会议录，第 317–330 页，2008年。3[Bla09] Eric Blais. 近乎最优地测试junta函数。载于第 41 届 ACM计算理论研讨会（STOC）年会论文集，第 151–158 页，2009年。3, 51, 63[BLR93] Manuel Blum、Michael Luby 和 Ronitt Rubinfeld. 自测试/自校正及其在数值问题中的应用。《计算机与系统科学杂志》，47:549–595，1993年。早期版本发表于 STOC'90。1, 2, 3[Bsh20] Nader H. Bshouty. 简洁表示的近乎最优测试器。载于《近似、随机化与组合优化：算法与技术》，APPROX/RANDOM，第 5:1–5:20 页，2020年。2, 3, 8[CDH+25a] X. Chen、A. De、Y. Huang、S. Nadimpalli、R.A. Servedio 和 T. Yang. 半空间难以进行相对误差测试。即将发表于 SODA 2026，2025年。, 3[CDH+25b] Xi Chen、Anindya De、Yizhi Huang、Yuhao Li、Shivam Nadimpalli、Rocco A. Servedio 和 Tianqi Yang. 单调性的相对误差测试。2025年。, 1, 2, 67[CFP24] Xi Chen、Yumou Fei 和 Shyamal Patel. 以亚线性数量的查询对决策列表进行无分布测试。载于第 56 届 ACM计算理论研讨会（STOC）年会论文集，第 1051–1062 页，2024年。3[CGM11] Sourav Chakraborty、David García-Soriano 和 Arie Matsliah. 高效的junta函数样本提取器及其应用。载于《自动机、语言与程序设计——第 38 届国际学术研讨会》（ICALP），第 545–556 页，2011年。2, 3[CP22] Xi Chen 和 Shyamal Patel. 对半空间进行无分布测试（几乎）需要 PAC学习。载于 2022年 ACM-SIAM离散算法研讨会（SODA）论文集，第 1715–1743 页，2022年。3[CPPS25a] Xi Chen、William Pires、Toniann Pitassi 和 Rocco A. Servedio. 合取式和决策列表的相对误差测试。载于第 52 届自动机、语言与程序设计国际学术研讨会（ICALP 2025），第 52:1–52:18 页，2025年。, 3, 4, 8, 9, 56, 57, 63, 66, 67[CPPS25b] Xi Chen、William Pires、Toniann Pitassi 和 Rocco A. Servedio. 以相对误差测试junta函数及其子类。载于第 38 届学习理论年会论文集（COLT 2025），2025年。, 3, 8, 10[CX16] Xi Chen 和 Jinyu Xie. 单调合取式无分布测试的紧确界。载于第二十七届 ACM-SIAM离散算法研讨会（SODA）年会论文集，第 54–71 页，2016年。3, 4, 66[DDS15] A. De、I. Diakonikolas 和 R. A. Servedio. 从满足赋值中学习。载于第二十六届 ACM-SIAM离散算法研讨会论文集（SODA 2015），第 478–497 页，2015年。65

[DLM+07] I. Diakonikolas、H. Lee、K. Matulef、K. Onak、R. Rubinfeld、R. Servedio 和 A. Wan. 简洁表示测试。载于第 48 届计算机科学年会研讨会（FOCS）论文集，第 549–558 页，2007年。2, 3, 8[FKR+04] E. Fischer、G. Kindler、D. Ron、S. Safra 和 A. Samorodnitsky. Junta测试。《计算机与系统科学杂志》，68(4):753–787，2004年。3[GGR98] Oded Goldreich、Shafi Goldwasser 和 Dana Ron. 性质测试及其与学习和近似的联系。《美国计算机学会杂志》，45:653–750，1998年。1, 2[Gol10] O. Goldreich，编。《性质测试：当前研究与综述》。施普林格，2010年。《计算机科学讲义》6390。3[GR02] Oded Goldreich 和 Dana Ron. 有界度图上的性质测试。《算法学》，32(2):302–343，2002年。3[GR20] Oded Goldreich 和 Dana Ron. 单侧错误的单项式和仿射子空间测试。《计算复杂性电子研讨会》，TR20-068，2020年。3[GS09] Dana Glasner 和 Rocco A. Servedio. 基本布尔函数的无分布测试下界。《计算理论》，5(1):191–216，2009年。3, 4[MORS10] Kevin Matulef、Ryan O'Donnell、Ronitt Rubinfeld 和 Rocco A. Servedio. 半空间测试。《SIAM计算杂志》，39(5):2004–2047，2010年。3[PR02] Michal Parnas 和 Dana Ron. 图直径测试。《随机结构与算法》，20(2):165–183，2002年。3[PRS02] M. Parnas、D. Ron 和 A. Samorodnitsky. 基本布尔公式测试。《SIAM离散数学杂志》，16:20–46，2002年。3[RS96] R. Rubinfeld 和 M. Sudan. 多项式的鲁棒刻画及其在程序测试中的应用。《SIAM计算杂志》，25:252–271，1996年。1

一个使用 A $n^{O(\log(s/\varepsilon))}$ 次查询、针对 s 项析取范式的相对误差测试器

观察85. 对于 $0 < \varepsilon \leq 1/2$，存在一种针对 $\{0,1\}^n$ 上 s 项 DNF 公式类的 ε 相对误差测试算法，它调用采样预言机 SAMP(f) $n^{O(\log(s/\varepsilon))}$ 次，并调用成员查询预言机 MQ(f) $O(1/\varepsilon)$ 次。

证明概要。 在 [DDS15] De 等人给出了一种算法，它使用 $n^{O(\log(s/\varepsilon))}$ 个来自 $f^{-1}(1)$ 的独立样本，学习任意未知的 s项 DNF 公式 f在 $\{0,1\}^n$ 上的 ε相对误差精度为（见 [DDS15]的定理 1.4）。这样的学习算法很容易按如下方式导出我们相对误差模型中的测试算法：

1. 首先，使用准确度参数 $\varepsilon/10000$ 运行学习算法，得到假设函数 $h' : \{0,1\}^n \to \{0,1\}$。（注意，这个假设 h'不一定是一个 s 项析取范式。）

2. 接下来，对所有 s 项 DNF 公式 h 进行穷举搜索，找出使 max{相对距离(h, h')，相对距离(h', h)}最小的公式。如果没有 s 项 DNF h 满足 max{相对距离(h, h')，相对距离(h', h)}$\leq \varepsilon/1000$ ，则停止并拒绝；否则继续。

3. 最后，从 $f^{-1}(1)$（分别从 $h^{-1}(1)$）抽取 $10/\varepsilon$ 个随机样本，并检查它们是否也都是 h（分别是 f）的满足赋值；如果是，则接受，否则拒绝。

显然，调用 SAMP(f) 和 MQ(f) 的次数符合所述；下面简要说明正确性的论证。

首先，假设 f 确实是一个 s 项析取范式公式。那么，学习算法以高概率（例如，至少 $99/100$）构造出一个假设 h' ，使得 reldist(f, h') $\leq \varepsilon/10000.$ 。在步骤 2 中，穷举搜索会找到一个 s 项析取范式 h ，使得 reldist(h, h')，reldist(h', h) $\leq \varepsilon/1000$ （因为 f 本身就是这样的 s 项析取范式），因此过程会继续到步骤 3。由于 reldist(f, h') $\leq \varepsilon/10000$ 且 reldist(h', h)，reldist(h, h') $\leq \varepsilon/1000$，根据引理 3 和 4 可得 reldist(f, h)，reldist(h, f) $\leq \varepsilon/100$。这意味着在步骤 3 中，任意一个来自 $f^{-1}(1)$ 的 $10/\varepsilon$ 个随机样本不满足 h 的概率至多为 $1/10$；反过来交换 f 和 h 的角色时也同样如此。因此，算法以至少 $79/100$ 的概率接受。

现在，假设对于每个 s-项析取范式公式 g， f 的相对距离满足 reldist(f, g) $> \varepsilon$ 。如果该过程在步骤 2 中拒绝，我们就完成了，因此假设步骤 2 成功找到了一个 s-项析取范式h ，满足 reldist(h, h'),reldist(h', h) $\leq \varepsilon/1000$。现在，由于 h 是一个 s-项析取范式，我们必有 reldist(f, h) $> \varepsilon$。由此容易证明，以高概率（至少为 $2/3$），要么来自 $f^{-1}(1)$ 的 $10/\varepsilon$ 个随机样本中有某个样本不满足 h ，要么来自 $h^{-1}(1)$ 的 $10/\varepsilon$ 个随机样本中有某个样本不满足 f。因此，该过程以至少 $2/3$ 的概率拒绝，我们就完成了。 □

B 无分布测试与相对误差测试不可比较

本节指出，无分布测试与相对误差测试是不可比较的模型：对于这两种模型中的每一种，都存在一个函数类，它在第一种模型中易于测试（即只需常数次查询即可测试），但在另一种模型中难以测试（即查询复杂度必须随维度 n 增长）。下面的观察将这一点定量地明确指出：

观察 **第 86 节**（无分布测试与相对误差测试不可比较） .

1. 存在一个函数类 $\mathcal{F}_1$ ，在相对误差模型中，只需使用 $O(1/\varepsilon)$ 个样本和成员查询即可进行 ε 测试；但在分布无关模型中进行 ε_0 测试，则需要 $\Omega(n^{1/3})$ 个样本和成员查询，其中 $\tilde{}$ 为某个绝对常数。2. 存在一个函数类 $\mathcal{F}_2$ ，对于任意 $\varepsilon > 2^{-n/3}$，在分布无关模型中进行 ε 测试无需样本和成员查询；但对于任意 $\varepsilon < 1$，在相对误差模型中进行 ε 测试则需要 $\Omega(2^{n/3})$ 个样本和成员查询。

对于第 1 部分，我们可以将 $\mathcal{F}_1$ 设为 $\{0,1\}^n$上的所有合取式类。相对误差测试的正面结果由 [CPPS25a], 的定理 1 给出，负面结果则是 [CX16]的定理 1.4。

算法 19 相对误差反单调合取测试器。

输入： 某个函数 f 的成员查询 MQ(f) 和 SAMP(f)：$\{0,1\}^n \to \{0,1\}$和 ε。

输出： “拒绝”或“接受”。

阶段 1：

1: 重复以下操作 $O(1/\varepsilon)$ 次：2: 抽取 $\boldsymbol{x}, \boldsymbol{y} \sim$ SAMP(f)。

3: 如果 MQ(f)($\boldsymbol{x} \oplus \boldsymbol{y}$) $= 0$，则停止并拒绝 f。

阶段 2：

4: 重复以下操作 $O(1)$ 次：5: 抽取 $\boldsymbol{x} \sim$ SAMP(f)。

6: 抽取一个均匀随机的 $\boldsymbol{y} \preceq \boldsymbol{x}$。7: 抽取 $\boldsymbol{u} \sim$ SAMP(f)。8: 如果 MQ(f)($\boldsymbol{y} \oplus \boldsymbol{u}$) $= 0$，则停止并拒绝 f。9: 接受 f。

对于第 2 部分，我们令 $\mathcal{F}_2$ 为所有满足以下条件的 g 类：$\{0,1\}^n \to \{0,1\}$ 具有 $|g^{-1}(1)| \geq 2^{2n/3}$. 对于下界，附录 A 中给出的一个简单的生日悖论论证 [CDH+25b] 表明，对于任意 $\varepsilon < 1$，任何针对 $\mathcal{F}_2$ 的 ε-相对误差测试算法，都必须进行 $\Omega(2^{n/3})$ 次黑盒查询，或从 $f^{-1}(1)$ 中抽取样本。对于上界，固定任意 $\varepsilon = \omega(2^{-n/3})$ 以及一个定义在 $\{0,1\}^n$上的分布 $\mathcal{D}$。我们声称，每个 $f\colon \{0,1\}^n \to \{0,1\}$ 在 $\mathcal{D}$ 下都与 $\mathcal{F}_2$ 中的某个元素 ε-接近（因此，测试算法可以直接接受，无需进行任何黑盒查询或从 $f^{-1}(1)$ 中抽取样本）。

为了说明该结论为何成立，令 $S \subset \{0,1\}^n$ 为 $\{0,1\}^n$ 中在 $\mathcal{D}$ 下权重最小的 $2^{2n/3}$ 个点（若有并列，则任意打破平局）。我们有 $\mathcal{D}(S) \leq 2^{2n/3}/2^n = 2^{-n/3} < \varepsilon$。因此，令 g 为将 f 在 S中的每个点处的值固定为 1 后得到的函数，我们有$\mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] < \varepsilon$。由于 g 具有 $|g^{-1}(1)| \geq 2^{2n/3}$，我们得到 $g \in \mathcal{F}_2$，结论得证。

C ConjTest概述，来自 [CPPS25a]

本节介绍 CPPS25a 的合取相对误差测试器 [CPPS25a]，称为**ConjTest**。作者首先指出，对于任意合取 f 以及任意 $y \in f^{-1}(1)$，函数

$$f_y(x) := f(x \oplus y)$$

是一个反单调合取式 [11]。同样不难看出，如果 f 与每个合取式都相差 ε 相对误差，那么 f_y 必然与每个反单调合取式都相差 ε 相对误差。

因此，作者并未明确给出 **ConjTest**，而是给出了如下用于反单调合取式的测试器。要得到 **ConjTest**，只需简单地抽取一个 $y \sim f^{-1}(1)$，然后在 f_y上运行反单调合取式测试器（算法 19），其中 MQ(f_y)和SAMP(f_y) 可以分别使用 MQ(f) 和 SAMP(f)轻松模拟。

由上可知，很容易验证 **ConjTest**确实当且仅当它所进行的每个 成员查询 都得到 1时才接受。

[11]如果一个合取式只包含否定文字，那么它就是反单调合取式。