Hirsch 猜想的形式化反证

Xavier Allamigeon
Inria CMAP、CNRS、
École polytechnique、巴黎综合理工学院，法国
xavier.allamigeon@inria.fr

Quentin Canu
Inria CMAP、CNRS、
École polytechnique、巴黎综合理工学院，法国
quentin.canu@inria.fr

Pierre-Yves Strub
Meta 法国
strubpy@meta.com

arXiv:2301.04060v1 [cs.LO] 10 Jan 2023

摘要

本文旨在对 Santos 等人针对所谓的多胞体直径（有界凸多面体）赫希猜想提出的一个反例进行形式化验证。与纸笔证明不同，我们的方法完全以计算为基础：我们在 Coq 中实现并证明一个算法的正确性，该算法能在证明助手中显式计算多胞体的顶点-边图及其直径。这种基于证书的算法的创新之处在于，在简洁性与效率之间取得了平衡。

简洁性对于获得算法的正确性证明至关重要。该证明分为两部分：一是证明一个基于面向证明的数据类型表述的抽象算法正确，二是证明它与一个基于面向计算的数据类型的底层实现之间存在对应关系。我们特别致力于将算法化简为一小组基本操作（例如矩阵乘法、集合和图上的基本例程），以便更清晰地推导出底层实现的正确性。

高效性使我们能够将方法扩展到组合结构颇具挑战性的多胞体。例如，我们形式化地验证了 Matschke、Santos 和 Weibel 给出的两个违反 Hirsch 猜想的反例：它们分别是 20 维和 23 维多胞体，拥有 36 425 个和 73 224 个顶点，其有理系数的分子和分母最多有 40 位数字。我们还通过计算一些知名多胞体类的顶点列表或直径，展示了该方法的性能，例如 McMullen 上界定理中涉及的循环多胞体（及其极对偶）。

关键词：多面体、多胞体、Hirsch 猜想、证明助手、经验证计算

1 引言

1.1 研究动机

多面体直径的研究是优化领域以下重大问题的核心：单纯形法能否在多项式时间内终止？自乔治·B·丹齐格在 40 年代末提出单纯形法以来，这个问题一直悬而未决，并启发菲尔兹奖得主 Steve Smale 将关于存在一种用于线性规划的强多项式算法的问题列为他的第九个问题，21st 作为世纪难题 [41]。单纯形法 [17] 无疑是求解线性规划问题最标准的方法，即求解如下形式的问题

$$\begin{aligned} &\text{Minimize} && \langle c, x\rangle \\ &\text{subject to} && \langle a_1, x\rangle \geq b_1\,,\ \ldots\,,\ \langle a_m, x\rangle \geq b_m\,, \\ & && x \in \mathbb{R}^n \end{aligned}$$

对于一些向量 a_1、……、a_m、$c \in \mathbb{R}^n$ 和实数 b_1、……、b_m，其中 $y, z := \sum_{i=1}^n y_i z_i$ 表示 y、$z \in \mathbb{R}^n$ 的欧几里得内积。该问题是在由满足约束条件 $a_i, x \geq b_i$（对所有 $i = 1$、……、m）的点 $x \in \mathbb{R}^n$ 构成的凸多面体上，最小化目标函数 $x \mapsto c, x$。从几何角度看，单纯形法的基本原理是通过遍历多面体的部分顶点，逐步减小目标函数值。具体而言，在每次迭代中，该方法都会从与当前顶点相邻（即由一条边相连）的顶点中，选取一个函数值较小的顶点。多面体顶点和边的数学定义见第 2 节，图 1 展示了相关示意图。每次迭代中下一个顶点的选择由所谓的枢轴规则决定。因此，每种枢轴规则都会使单纯形法沿多面体的顶点-边图行进，也就是沿由该多面体的顶点和边构成的图行进（见图 1）。文献中已有大量枢轴规则，但所有经过数学分析的规则都已被证明在最坏情况下呈现（次）指数级行为（近期综述见 [3], 和 [20] ）。

回顾一下，在图中，两个顶点之间的距离 是它们之间任意一条最短路径的长度（即边的数量）。图的直径 定义为任意两个顶点之间的最大距离。因此，多面体的（组合）直径，也就是其图的直径，构成了采用任何枢轴规则的单纯形法所执行迭代次数的下界。基于这一动机，赫希于 1957 年在致 Dantzig 的一封信中提出了以下猜想：

猜想 1（多胞体上的赫希猜想）。任何具有 d 维和 p 个面的多胞体，其直径不超过 $p - d$。

在此表述中，多胞体 指有限多个点的凸包，或者等价地说，有界多面体。d维多面体的面 是维数为 $d - 1$ 的面。（最初，猜想的表述

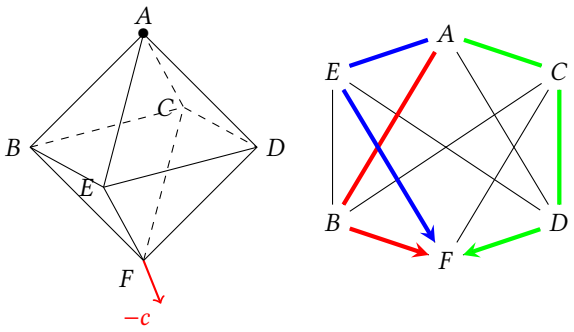


图1。 单纯形法在 3 维交叉多胞体上的几种可能执行轨迹（说明见第 6 节）。单纯形法的执行过程会在多胞体的图中，从起始顶点到最优顶点画出一条路径（图中最右侧部分）。

针对的是多面体而非多胞体，但人们很快意识到，该界限对无界多面体并不成立 [35]。）多年来，多胞体和多面体直径的研究备受关注；例如，参见 [46] 中的相关综述。尽管如此，这个猜想仍有五十多年未获解决，直到Santos给出了一个反例：

定理1 ([39])。 存在一个43维多胞体，它有86 个面，且直径大于43。

在与 Matschke 和 Weibel 合作开展的后续研究中，[36], Santos 给出了另外两个反例，分别位于 20 维和 23 维，均反驳了该猜想。所有这些构造都关键依赖于计算论证。更具体地说，在这些构造中，非Hirsch多胞体都是由一个维数较低、具有特殊组合结构的多胞体——即一个纺锤体——构造而成；并且必须证明该纺锤体中两个指定顶点之间的距离大于其维数。对于最初的反例，这个纺锤体有 5 维、48 个面和 322 个顶点。Santos 指出，这一性质已借助非正式软件 Polymake [24], 进行了验证，并提供了两种彼此独立的证明，称其“无需计算机（但并非无需计算）”。幸运的是，利用这个纺锤体的对称群，可以手工完成这些计算。

自 Santos 取得突破以来，人们对多胞体与多面体直径的兴趣始终不减。首先，目前仍不清楚是否能在更低维数下找到违反赫希猜想的反例。此外，从单纯形法复杂度的角度来看，最重要的问题是多面体的直径能否由维数和面数的多项式函数所界定；这就是多项式*Hirsch*猜想。计算很可能会在解决这两个问题的过程中发挥关键作用，正如它们在 Santos 的构造中所起的作用一样。

因此，我们有充分的理由开发一个框架，使多面体上的计算能够在证明助手中进行。这样既能大幅拓展病态多胞体的研究范围（例如，使研究能够扩展到顶点数量更多的情况），又能保持（甚至提高）相较于纸笔计算的可信度。多胞体的组合性质并非此类贡献唯一受益的研究主题。例如，多面体是软件编译、验证 [16],或动力系统分析中的不变量计算 [31]等关键应用的核心工具。在这些应用中，计算多面体顶点是核心的基本操作，而开发一个能够执行此操作且经过形式化证明的算法，是一项值得追求的目标。

1.2 贡献

我们提出了一种算法，用于计算顶点集合以及多胞体图，并介绍了该算法在证明助手 Coq 中的实现及其正确性证明。我们的贡献独到之处在于兼具简洁的设计与实用的效率。简洁性是该算法的关键特征，使其能够切实地在证明助手中实现并 得到证明。简洁性在可维护性以及向其他证明助手移植方面也具有显著优势。

我们的算法基于证书：它以某种非正式软件计算出的图为输入，并对该图确实正确这一事实进行形式化认证。具体而言，证书是多胞体经过扰动后所得的图，该图具有连通、正则等显著性质。得益于这些性质，认证过程由一系列基本步骤构成，涉及矩阵乘法、集合运算等基本操作。随后，可以通过某种映射将原多胞体的图推导为前述图的像。

首先，使用一些面向证明的类型实现算法并证明其正确性，例如表示矩阵、凸多面体等的依赖类型。然而，这些类型用于计算时的代价高得难以接受：它们的定义旨在简化数学理论的形式化开发，因此优先采用朴素的数据结构和算法，而非计算性能良好的实现。因此，我们使用面向计算的类型实现第二个算法，例如持久化数组、原生整数和任意精度有理数。该算法结构简单还有一项额外好处：这使得低层实现与高层实现之间的等价性证明更容易。

为展示我们方法的实际效率，我们对若干类值得关注的多胞体进行了实验。我们成功计算出违反 [36] Hirsch 猜想的 20 维和 23 维反例的图，它们分别有 36425 个顶点和 73224 个

364250 条边，由此我们得出该猜想的形式化反证。我们还研究了超立方体、交叉多胞体以及循环多胞体的对偶多胞体等经典多胞体。根据麦克马伦上界定理，后者在维数和面数固定时已知具有最多的顶点 [37]。

1.3 相关工作

计算多胞体与多面体的顶点是一项出了名的难题。由于顶点数量 v 可能随定义不等式的数量 m 和维数 n呈指数级增长，因此必须将其复杂度视为 m、 n 和 v的函数。该问题的最坏情况复杂度尚未完全明了：对于无界多面体，这是一类 NP 困难的枚举问题 [34], ，但对于多胞体，其复杂度现状仍未解决。两种主要的顶点枚举算法是双描述法 [23, 38] 和反向搜索法 [5, 6]。双描述法的复杂度难以用m、 n 和 v来界定；而对于“非退化”的多胞体表示（定义见第 3 节），反向搜索法的复杂度属于 O（poly(m, n) v）。这与我们的算法属于同一复杂度类，因为这两种方法都依赖于枚举单纯形基。双描述法和反向搜索法都有标准实现 [4, 22] ，广泛用于计算数学 [24, 43]和软件验证 [8]领域的专用软件中。

计算证明（或反射证明）如今已成为 Coq 证明助手中的常用技术。例如，它被用于实现依赖符号计算的策略 [29], ，也广泛用于四色定理的形式化证明 [25]。反射证明也是小尺度反射 （SSReflect [26]）证明方法的核心；在这种方法中，人们广泛运用计算来解决涉及可判定谓词的目标。其他基于计算的 Coq 证明助手证明实例，请参见 [11, 13, 32, 33] 。反射证明也已应用于许多其他系统：Agda [44], 、Isabelle/HOL [12], 、Lean [7], 、PVS [45]。关于多面体计算，受静态分析应用启发的 Farkas认证技术见于 [10, 21]。与我们的工作不同，该工作仅涵盖使用不等式进行多面体计算，不涉及顶点或顶点-边图的计算。最后，我们提及 [42] 已在 Isabelle 证明助手中实现了一个基于单纯形的可满足性判定程序。

最后，我们的论文采用了程序和数据结构细化技术。这是一种证明方法，其核心是将高层程序或数据结构转换为较低层、效率更高的实现，同时保留其主要性质（程序规约和/或数据结构不变量）。细化技术早已得到应用于 Coq 证明助手中，尤其是 Co-qEAL [14, 19], ——一个旨在简化数据结构细化定义的 Coq 框架。我们的数据精化形式化工作紧密遵循 CoqEAL 的方法。不同之处在于，作为初步实验，我们选择不利用 CoqEAL 提供的等价性证明自动演绎功能（该功能基于 Coq 类型类）。这样，我们就能更细致地控制算法的底层实现。我们指出，由于算法本身简单，手动证明等价性证明所需的工作量并不大。

1.4 论文结构

在第 2 节中，我们回顾多面体的基本概念并介绍相关记号。在第 3 节中，我们定义一种基于证书的算法，用于计算给定多胞体的图。第 4 节介绍使用面向计算的数据类型实现该算法，并给出此实现的正确性证明。在第 5 节中，我们给出对赫希猜想的形式化反证。最后，在第 6 节中，我们报告该方法在若干类多胞体上的实验结果。

本文所提交版本的源文件可在 Coq-Polyhedra 库的 Git 仓库中找到[1]。本文经常引用这些源文件。除 Coq 外，我们还依赖 MathComp 库 [27] 以及`finmap`和`bignums` 库。[2,3]

我们指出，本文是发表于 CPP'23 会议论文集中的会议论文 [1] 的一个（略微）扩展版本。

2 预备知识与记号

如引言中所述，（凸）多面体定义为满足有限个仿射不等式 $a_i, x \geq b_i$ 的点集 $x \in \mathbb{R}^n$ ，其中 $a_1, \ldots, a_m \in \mathbb{R}^n$ 以及$b_1, \ldots, b_m \in \mathbb{R}$。我们也将约束系统写作 $Ax \geq b$，其中 $A \in \mathbb{R}^{m \times n}$ 是行向量为 a_i^T, $b = (b_i)_i \in \mathbb{R}^m$ 的矩阵，而 $\geq$ 表示 $\mathbb{R}^m$ 上的逐项序。多面体中一个值得注意的子类是多胞体。多胞体是有限个点 $v^1, \ldots, v^p \in \mathbb{R}^n$ 的凸包 $\left\{ \sum_{i=1}^p \lambda_i v^i : \lambda_1, \ldots, \lambda_p \geq 0, \sum_{i=1}^p \lambda_i = 1 \right\}$。闵可夫斯基定理指出，多胞体恰好是有界多面体。

多面体的维数 定义为其仿射包的维数，也就是包含该多面体的最小（按包含关系而言）仿射子空间。例如，点的维数为 0，线段的维数为 1，等等。设 P 为一个多面体。P 的一个（非空）面 是使线性函数 $x \mapsto c, x$ 在 P 上取最小值的点集，其中 $c \in \mathbb{R}^n$。等价地，一个面是某个定义在 P 上的线性规划的最优解集。顶点 和边

[1] https://github.com/Coq-Polyhedra/Coq-Polyhedra/tree/CPP-23

[2] https://github.com/math-comp/finmap [3] https://github.com/coq-community/bignums

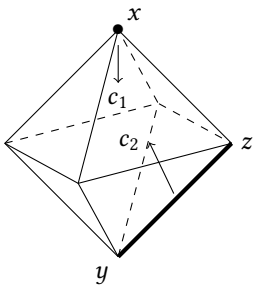


图 2。 三维交叉多胞体。目标向量 c_1 仅在一个顶点 x处取最小值，而目标向量 c_2 在一条边 $[y,z]$ 上取最小值，该边的端点为 y和 z。

是 $\mathcal{P}$ 的维数分别为0和1的面；见图2。任意有界边都可表示为两个顶点之间的线段 v,v'，它们属于 $\mathcal{P}$。在这种情况下，顶点 v和v' 称为相邻。如引言所述，$\mathcal{P}$的图记作 $G_{\text{vert}}(\mathcal{P})$（或在上下文明确时简记为 G_{vert}），是由顶点间的邻接关系所诱导的组合图。

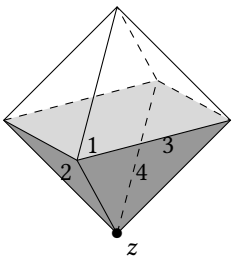
在本文其余部分中，对于所有整数 $p \geq 1$，我们定义 $[p] := \{1,\dots,p\}$。有限集合 S 的基数记为 $\#S$。给定矩阵 $M \in \mathbb{R}^{p\times q}$，我们将其第 i行（$i \in [p]$）记为 M_i；对于所有子集 $I \subset [p]$，将行 M_i 对应于 $i \in I$的子矩阵记为 $M_I \in \mathbb{R}^{\#I\times q}$。大小为 $p\times p$的单位矩阵记为 Id_p。给定一个（无向）图$G = (V,E)$（其中 $E \subset V\times V$）和一个顶点 $v \in V$，我们将 $v \in V$在 G中的邻域（v）记为 N_G，即顶点 w的集合，其中的顶点与 v 相邻。给定一个函数$f\colon V \to V'$，G在 f下的像记为 $f(G)$，其定义为如下图：顶点为 $f(V) := \{f(v)\colon v \in V\}$，边为$\{(f(v)\ f(w)) : (v,w) \in E,\ f(v) \neq f(w)\}$。

3 图认证算法

3.1 非退化情形下的首个算法

我们的方法高度依赖基和基本点的概念，它们是单纯形法的核心要素。设 $\mathrm{P} = \{x \in \mathbb{R}^n\colon Ax \geq b\}$ 为一个多面体，其中$A \in \mathbb{R}^{m\times n}$ 和 $b \in \mathbb{R}^m$。回顾一下，基是一个基数为 n 的子集$I \subset [m]$，使得子矩阵 A_I 非奇异。在这种情况下，等式组 $A_I x = b_I$ 有唯一解 x^I，我们称其为与 I相关联的基本点。当 x^I 属于多面体 P 时，称基 I 为可行。进一步地，称点 x^I 为可行基本点。

这是一个标准性质：$\mathcal{P}$ 的顶点恰好是可行基点；参见 [40, 第 11]章。然而，一般来说，它们之间的对应关系并非双射：每个可行基点都是一个顶点，但一个顶点可能对应多个可行基；参见图3 以了解示例。这些基称为退化。当处于非退化情形时，



$$\begin{cases} 1 & x_1+x_2+x_3 & \geq -1 \\ 2 & -x_1+x_2+x_3 & \geq -1 \\ 3 & x_1-x_2+x_3 & \geq -1 \\ 4 & -x_1-x_2+x_3 & \geq -1 \\ 5 & x_1+x_2-x_3 & \geq -1 \\ 6 & -x_1+x_2-x_3 & \geq -1 \\ 7 & x_1-x_2-x_3 & \geq -1 \\ 8 & -x_1-x_2-x_3 & \geq -1 \end{cases}$$

图3. 三维交叉多胞体存在退化基。例如，底部点 z ($x_3 = -1$)是一个顶点，与四个可行基相关联：(2，3，4)、(1，3，4)、(1，2，4) 和 (1，2，3)。

不存在这样的基，即可行基与顶点之间的对应关系是一一对应的。

给定一个形式如下的线性规划

$$\text{Minimize} \quad \langle c,x\rangle \quad \text{subject to} \quad Ax \geq b,\ x \in \mathbb{R}^n, \quad (1)$$

其中 $c \in \mathbb{R}^n$，单纯形法会依次遍历可行基，直到找到使函数 $x \mapsto c$，x取得最小值的（可行）基本点。在这一方案中，任意两个连续的基 I，I' 都满足 $\#(I \cap I_\prime) = n-1$，也就是说，它们只相差一个元素。这样的基称为相邻。这一邻接关系产生了（可行）基图，我们将其记为 G_{bases}。基本点与顶点之间的关系可推广到多面体的图与基图，如下所示：

命题2。 顶点-边图 G_{vert} 是图 G_{bases} 在函数 $I \mapsto x^I$ 下的像；该函数将任意可行基映射到其基本点。

此外，在非退化情形下，后一个函数是这两个图之间的同构。

这一结果进一步阐明了我们在引言中对单纯形所作的几何描述，即单纯形法在多面体的顶点-边图中诱导出一条路径。

本节仅讨论非退化情形下多胞体顶点-边图的计算。介绍这一特殊情形，是为了便于理解第 3.2 节中给出的一般算法。我们说明，除非另有明确说明，否则我们无需将下文呈现的结果形式化。

鉴于命题 2 的第二部分，我们利用 G_{vert} 与 G_{bases}之间的同构，并概述一种算法，用以验证给定的图（通过某种非正式过程 事先计算得到）是否与可行基图一致。为此，我们利用以下两个基本性质：

命题 3. 图 G_{bases} 是连通的。

算法 1 图认证算法（非退化情形以及 P $=\{x \in \mathbb{R}^n\colon\ Ax \geq b\}$ 有界）

要求： $A \in \mathbb{R}^{m\times n}$， $b \in \mathbb{R}^m$ 和 $G =$ （V， E）

断言 G 非空
对所有 $I \in V$ **执行断言** I 是一个可行基
对所有 $I \in V$ **执行**
    对所有 $J \in N_G(I)$ **执行断言** $\#(I \cap J) = n - 1$
    断言 $\#N_G(I) = n$
完成
返回 *true*

命题 4. 在非退化情形下，且当$\mathcal{P}$是一个多胞体时，图 G_{bases} 是 n-正则的，也就是说，每个可行基都恰好与 n 个可行基相邻。

命题 3 在一般情形下成立（即使存在退化基也是如此）。这是以下事实的推论：对于任意可行基 $I^\star$，我们都能找到一个向量 $c \in \mathbb{R}^n$ 使得 $x^{I^\star}$ 是线性规划（1）的唯一最优解。然后，以任意可行基 I 为初始基的单纯形法会在图 G_{bases} 中沿路径到达 。

涵盖非退化情形的算法，以及假设$\mathcal{P}$ 是一个多胞体的条件，是算法1。我们使用命令**断言** *<cond>*作为以下代码块的一种语法糖：**如果非** *<cond>* **则返回***false*。输入图的顶点 $G = (V, E)$ 被假定为整数的集合 I。该算法分为四个步骤：

(i) 检查该图是否非空（第1行）；(ii) 检查每个顶点 I 是否为可行基（第2行）；(iii) 对每个顶点 $I \in V$，检查其邻域是否由相邻基构成（第4行）；(iv) 对每个顶点 $I \in V$，检查其邻域的基数 n （第5行）。

第二步和第三步确保 G 是G_{bases}的子图。此外，由于 G_{bases} 是 n-正则的（命题4），第四步实际上确保 $N_G(I) = N_{G_{\text{bases}}}(I)$ 对所有$I \in V$都成立。因此， G 是 G_{bases} 的一个子图，并且 G 中每个顶点的邻域都与G_{bases}中的相同。如下述引理所示， G的非空性和 G_{bases} 的连通性进而确保 G和G_{bases} 相同：

引理 5. 设 G和 H是两个图，且 G是 H的一个非空子图。假设 H是连通的，并且对于 G的所有顶点 v，都有 $N_G(v) = N_H(v)$。那么 $G = H$。

这个引理由引理 sub_图同构谓词 (参见high_图。v) 项目源代码中给出。（该结果实际上表明了一个稍微更一般但等价的命题，其中G 被一个同构图替代。）接下来便可由引理5推出以下结果，并证明算法 1是正确的：

定理 6. 假设我们处于非退化情形，且 $\mathcal{P} = \{x \in \mathbb{R}^n\colon\ Ax \geq b\}$ 是一个多胞体。若算法 *1* 返回 *true*，则 $G = G_{\text{bases}}$。

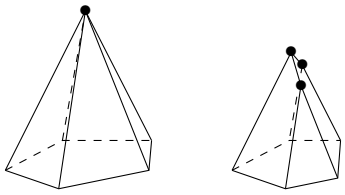


图 4。字典序扰动方法。左侧顶部的顶点在右侧被分裂成三个不同的顶点。

读者也可以验证，若算法 1 中的某个断言不成立，则图 G不可能等于 G_{bases}。

3.2 处理一般情形

3.2.1 扰动方法。 一般情况下，非退化情形并不成立，但我们可以通过对多胞体进行轻微扰动将问题化归于此，同时仍能以某种方式恢复主要的组合结构，例如顶点集合或顶点-边图。我们的方法源自 Dantzig 等人提出的所谓字典序枢轴规则 [18], , 后来 Avis [5] 将其用于顶点枚举算法 [4]。该方法通过扰动向量 b 来实现：将每个分量 $b_i (i \in [m])$ 替换为 $b_i - \varepsilon^i$，其中 $\varepsilon > 0$ 是一个充分小的实数。从几何上看，虽然界定多面体的超平面的法向量保持不变，但对向量 b 的扰动会将每个与多个（退化）基相关联的顶点分解为不同的（非退化）基本点；示意见图 4 。

不通过赋予 ε 某些数值（这会带来一个问题：究竟该取多小），而是以符号方式实现扰动：将每个 b_i 视为关于 ε 的次数至多为 m的多项式。由此，我们考虑一个“多面体”，其中各点的坐标也是关于 ε 的次数至多为 m 的多项式。这类多项式 $\sum_{k=0}^{m} \alpha_k \varepsilon^k$ 可以编码为大小为 $1 + m$的行向量（α_0， . . .， α_m）。在这种情况下，实数上的标准序被$\leq_{\text{lex}}$元组上的字典序所取代。事实上，（α_0， . . .， α_m） $\leq_{\text{lex}}$ (β_0， . . .， β_m) 当且仅当 $\sum_{k=0}^{m} \alpha_k \varepsilon^k \leq \sum_{i=0}^{m} \beta_k \varepsilon^k$ 对所有$0 < \varepsilon \ll 1$都成立。由此得到符号扰动多面体

$$\widetilde{\mathcal{P}} := \left\{ X \in \mathbb{R}^{n\times(1+m)} : AX \geq_{\text{lex}} \widetilde{b} \right\} \tag{2}$$

其中矩阵 $\widetilde{b} := \begin{bmatrix} b & -\mathrm{Id}_m \end{bmatrix} \in \mathbb{R}^{m\times(1+m)}$ 对应于上述向量的扰动 b： i第~行 b 编码多项式 $b_i - \varepsilon^i$。（2）中，关系 $\geq_{\text{lex}}$ 表示字典序的逐项扩展：若两个矩阵 X、$Y \in \mathbb{R}^{p\times(1+m)}$ 满足 $X \geq_{\text{lex}} Y$，则$X_i \geq_{\text{lex}} Y_i$ 对所有 $i \in [p]$均成立。矩阵 $X \in \mathbb{R}^{n\times(1+m)}$ 中的 ~$\mathcal{P}$对应于分量经过扰动的向量（用 ε 中次数至多为 m的多项式编码）。

对于这种符号化多面体，可行基的概念仍然适用。为避免与~原多面体的基混淆，P 的可行基称为字典序可行基。形式上，集合 $I \subset [m]$ 是一个字典序可行基，当且仅当 I的基数为 n，矩阵 A_I 非~奇异，且基本点 $X^I := A_I^{-1} b_I$满足 $AX^I \geq_{\text{lex}} b$。我们回顾一下，字典序可行基并不退化：

引理 7（见 [2]，其中给出了形式化实现）。设 I，I' 是两个不同的字典序可行基。那么 $X^I \neq X^{I'}$。

3.2.2 字典序图性质的形式化。在下文中，我们假设 $\mathcal{P}$ 是一个多胞体。

定义在可行基上的邻接关系可以直接推广到字典序可行基上。这由此定义了字典序可行基图，简称字典序图，记作 G_{lex}。

字典序可行基是 Allamigeon 和 Katz [2],在 Coq-Polyhedra 库中实现的单纯形法形式化的基石。更具体地说，后者的工作对字典序单纯形法进行了形式化；该方法遍历字典序可行基，以避免在退化基上发生循环。Coq-Polyhedra 引入了字典序可行基的类型`lex_feasible_basis A b`，其中 `A : 'M_(m,n)` 和 `b : 'cV_m` 分别对应矩阵 A 和（未扰动的）向量 b。（回顾一下，`'M_(m,n)` 和 `'cV_m`分别是 MathComp 库 [27]提供的 `m ×n` 阶矩阵和 `m` 维向量的类型。）我们以此为基础，首先定义字典序可行基的图，记作`lex_graph`：

```
定义集合_邻接 := (* 邻接关系 *)
  函数 I I'   : {集合 'I_m}  => #| I :&: I'   | == n.-1.
定义字典序_图 := 构造_图
  [fset x  |  x: 单纯形法.lex_可行_基 A b]集合_邻
  接关系。
```

接下来我们证明，正如预期，命题 3和 4对于字典序可行基同样成立：

```
引理字典序_图_连通 : 连通的字典序_图。
引理字典序_图_正则 : 正则的字典序_图。
```

其证明直接源于 Coq-Polyhedra 中提供的字典序单纯形法形式化。

字典序可行基的一个基本性质是，它们构成 $\mathcal{P}$ 的可行基的子集。更具体地说，我们用 π 表示将一个矩阵 $X \in \mathbb{R}^{n\times(1+m)}$（即一个扰动点）映射到其第一列的函数。后者对应于 X的未扰动部分，即扰动点在 $\varepsilon = 0$时的值。以下结果是众所周知的结果（形式化内容请参见 [2]）：

命题 8。设 I 是一个字典序可行基。那么 I 是 $\mathcal{P}$ 的一个可行基，而 $\pi(X^I)$ 是与之关联的基本点。

我们将后一结果推广为字典序图与顶点-边图之间的如下对应关系。

定理 9. $\mathcal{P}$ 的顶点-边图 G_{vert} 是 G_{lex} 在函数 $\phi\colon I \mapsto \pi(X^I)$下的像。

在 Coq 中，该陈述写作如下形式（参见模块`enum_proof`）：

```
定理 im_lex_graph_vert_graph:

  poly_图 P  =
    (Simplex.point_of_basis b) @/ 字典序_图。
```

其中，项 `poly_graph P` 是顶点-边图 G_{vert}，函数 `Simplex.point_of_basis b` 对应于函数 ϕ，而形如 `f @/ G` 的项表示图 `G` 在函数 `f`下的像。

我们简要说明定理 9 的形式化证明，因为文献中通常没有详细阐述其非形式化证明。证明中最困难的部分是说明，对于 P 的每条边 $[v, w]$，都存在两个相邻的字典序可行基 I、J，使得 $v = \phi(I)$ 且 $w = \phi(J)$。我们利用 Coq-Polyhedra 的字典序单纯形法构造这两个基。更确切地说，由于 $[v, w]$ 是 P 的一条边，因此存在一个向量 c，使得 $[v, w]$ 恰好是 P 中使函数 $x \mapsto c, x$ 取得最小值的点集。以目标向量 c 调用字典序单纯形法，可得到一个字典序可行基 I_0，使得点 $\phi(I_0)$ 取得该最小值。由于 $\phi(I_0)$ 是 P 的一个顶点，因此它必定是 v 或 w。不失一般性，我们假设 $\phi(I_0) = v$。现在考虑一个目标向量 c'，使得w 是 P 上唯一使 $x \mapsto c', x$ 取最小值的点（根据顶点的定义，这是可能的），并引入第三个目标向量 $d := c + \delta c'$，其中 $\delta > 0$ 是一个足够小的量。直观地说，以这种方式将 c 扰动为 d，应当能确保 w 是使 $x \mapsto d, x$ 在 P 上取最小值的唯一点，并且 v 是仅次于 w的第二“优”顶点，也就是说，对每个顶点 $z \notin \{v, w\}$，都有 $d, w < d, v < d, z$。[4]最后，我们从字典序可行基 I_0 出发，以目标函数$x \mapsto d, x$应用字典序单纯形法。由于沿途目标函数不会增加，字典序单纯形法会生成一个由相邻字典序可行基组成的序列 $I_0, \ldots, I_{p-1}, I_p, \ldots$，使得对所有 $k < p$，都有 $\phi(I_k) = v$，且 $\phi(I_p) = w$。于是，只需取 $I = I_{p-1}$ 和 $J = I_p$。

3.2.3 字典序图和顶点-边图的认证。根据引理字典_图_连通 和 引理 字典_图_正则中形式化的性质，我们可以从算法 1 推导出一种认证方法，用以确认非正式计算得到的图与字典序图一致。随后，定理9 将使我们能够通过计算$\mathcal{P}$ 的像来恢复其顶点-边图，像是通过 ϕ计算得到的。

我们描述 G_{lex}的认证过程的形式化。其输入为一个图 `G`，其顶点是形如 `(I, X)` 的有序对，其中 `I : {集合 'I_m}` 是整数子集（小于 m），而 `X : 'M_(n,1+m)` 是一个 $(n \times (1+m))$ 矩阵。这两个分量分别表示一个字典序可行基 I及其对应的基本点 X^I</output>

[4]这些关系实际上表明，δ需要选得多小。

算法被形式化为一个返回布尔值的程序（请注意，&&、 [&& ...] 和[对所有 ..., ...]表示布尔合取）： </output>

定义high_enum_算法 G : bool :=(G != graph0) (*G 非空 *) &&[对所有 u : G 的顶点，(* u 是一个有序对 (I,X) *)[&& 基数_验证 u，bas_验证 u，feas_验证 u，reg_验证u & 子集_验证 u]]。

其中我们定义

定义 card_verificationu := #|u.1| == n. 定义 bas_verification u :=(row_submx A u.1) *m u.2 == row_submx b_pert u.1. 定义 feas_verificationu :=u.2 \in Simplex.lex_polyhedron A b_pert. 定义 reg_verification u :=#|`successors G u| == n. 定义 subset_verification u :=[对所有 u' : successors G u, set_adjacence u.1 u'.1]。

回顾一下，u.1 : T 和 u.2 : T' 分别对应有序对 u : T * T' 的第一和第二个分量。high_enum_算法的实现遵循算法 1 的结构。具体而言，它首先检查图 G 是否非空，然后对每个顶点 u 依次进行五项测试。最后两项测试针对 G 中 u 的邻域 successors G u，分别检查其基数 n (reg_verification)以及是否由相邻基构成 (subset_verification)。与算法 1 的主要区别在于，验证 G 中每个顶点 u 的第一分量 I 是否为字典序可行基的方式。前面三项测试 card_verification、bas_verification 和 feas_verification 正是为此而设。第一项检查其 I 基数为 n，而最后两项则分别确保$A_I X = \widetilde{b_I}$和 $AX \geq_{\text{lex}} \widetilde{b}$。注意，由于 $\widetilde{b} = \begin{bmatrix} b & -\mathrm{Id}_m \end{bmatrix}^{\sim}$等式 $A_I X = b_I$确保 A_I是一个非奇异矩阵。事实上，若 Y是由 X的第 $(1+i)$ 列（其中 $i \in I$）构成的子矩阵，可以验证 $A_I Y = -\mathrm{Id}_n$。换言之，矩阵 X 本身就带有一份证书，证明 A_I是$\widetilde{\ }$非奇异的。因此， I是一个基。由于 $A_I X = b_I^{\sim}$，所以 $X = X^I$，而条件 $AX \geq_{\text{lex}} b$最终确保 I 是一个字典序可行基。

由此我们得以证明high_enum_algo的正确性：

定理 表示_词法_图G:
高_枚举_算法 G ->
图同构谓词 G 词法_图（函数 u => u.1）。

蕴含关系最右侧的图同构谓词 表示函数 $(I, X) \mapsto I$是G 与字典序图之间的同构。

以下陈述由以下定理结合得到：定理im_词法_图_顶点_图 和定理 表示_词法_图:

定理 表示_多项式_图G:
高_枚举_算法 G -> 多项式_图 P = (phi @/ G)。

其中函数 phi 定义为 phi u = col 0 u.2，也就是说，如果 u = (I, X)，那么 phi u 就是矩阵 X的第一列。

4 高效实现

The Coq 函数 high_enum_algo 在第3节中介绍，它使用依赖类型（例如 MathComp 类型），这些类型经过调整以便于证明，而非计算。例如，自然数采用一元表示，有理数则附带证明项，证明其分子和分母互素。类似地，有限集、图或矩阵的实现基于 MathComp 序列（即通过归纳构造的基本列表），这些序列并非为快速计算而设计，且附带多个证明项来保证其良构性。因此，即便对于最简单的实例，函数high_enum_algo 也无法在合理时间内返回。为解决这一问题，我们采用更接近机器表示、因而在实践中效率更高的数据类型。基于这些类型，我们实现了函数high_enum_algo的“低层”版本；见第4.1节。在第4.2节中，我们介绍如何通过结合精化，将高层数据结构与低层数据结构关联起来。最后，在第4.3节中，我们证明函数high_enum_algo的低层实现与其高层实现等价。

4.1 底层实现

认证算法底层实现中使用的主要数据类型如下：(i) 基于 OCaml 整数构建的 63 位整数类型整数（Coq 中的模块 Int63 ）；(ii) 基于 OCaml 数组构建、并基于 [15, 第 2] 节中的思想的持久化数组类型数组（Coq 中的模块PArray ）；(iii) 表示任意大的有理数的类型bigQ （Coq 库bignums中的模块BigQ ），它基于由 63 位整数构成的字序列 [30]。

类型 bigQ 适用于处理这样的多面体：其不等式系统中的数值项或顶点中的数值项可能是非常大的有理数，例如我们在第 5 节讨论的赫希猜想反例。如下文所述，底层实现中的多种数据结构都用到了持久化数组。我们选择持久化数组而非 Coq AVL 树，因为早期实验表明，后者存在性能问题。事实上，每棵树都附带一个用于平衡的证明项，而在大型实例中，这个证明项可能会变得非常庞大。

具有有理数项的向量和矩阵分别使用类型 `array bigQ` 和 `array (array bigQ)`实现。定义多面体的不等式系统由类型为polyType：`= array (arraybigQ) * array bigQ` 的项表示。基（即行索引的集合）使用类型 `array int`编码。更准确地说，基的元素按顺序存储在一个数组中。这样便可以在线性时间内计算两个基的交集。

顶点带有某种类型t标签的图，也就是类型为`graph t`的图的低层对应项，是通过`(g, lbl)`形式的有序对实现的，其中`g :array (array int)` 和`lbl : array t`。具体而言，底层图的顶点由整数（类型为`int`）编入索引，项`lbl.[i]` 对应于索引为i的顶点的标签。项g 表示图的邻接数组，也就是说，`g.[i]` 是一个数组，包含索引为i的顶点的邻居索引。我们采用记法`graph_struct := array (array int)`。顶点的索引方式确保数组`lbl` 中的标签按非递减顺序排列（为此，我们在标签上引入一种全序关系）。这样可以确保图中每个标签只出现一次。

利用这些数据结构，我们构建了一个底层实现，称为`enum_algo`，用于实现函数`high_enum_algo`；参见模块`enum_algo.v`。函数`enum_algo` 是对`high_enum_algo` 在底层数据结构上的直接适配。具体而言，`high_enum_algo` 中的每项测试，在底层实现中都有对应项。与基于依赖类型的高层数据结构不同，我们的底层数据结构并不天然具备良构性不变量。因此，必须通过额外的函数检查这些不变量。例如，我们需要验证表示向量和矩阵的数组是否具有特定大小（如环境空间的维数或不等式的数量）、图中出现的所有顶点索引是否都属于正确的范围，以及表示集合（如基或图标签）的数组是否已排序。我们的认证算法的优势在于，无需证明这些不变量在函数执行过程中得以保持。事实上，由于算法简单，数据结构只会被读取，不会产生任何新结构。因此，数据结构一致性的检查只需在调用函数`enum_algo`之前进行一次。

一旦`enum_algo` 验证某个底层图是该多胞体的字典序图（的底层表示），就还需要处理通过函数 `phi` 在 定理 `repr_多胞体_图` （参见第 3.2.3 节末尾）。这是得到该多胞体顶点-边图的最后一步。图的像是高层图的一种基本构造，但在底层处理时，问题会稍微复杂一些。我们再次依靠证书。具体而言，我们定义一个函数 `img_lex_图`，它以两个图 `g_lex g_vert : 图_struct` 及其各自的标记 `lbl_lex` 和 `lbl_vert`，并检查 `(g_vert,lbl_vert)` 是否是`(g_lex, lbl_lex)` 通过底层对应项 `low_phi := 函数 (u.2)`。`[0]` 函数`phi`所得的像。我们简要介绍该函数的实现。

函数 img_lex_graph 还需提供证书 morph、morph_inv 和 edge_inv。项 morph : array int 对应两个图的顶点索引之间的映射。函数 img_lex_graph 首先检查该映射是否与函数 low_phi 在标签上的作用一致，也就是说，对于 g_lex 的每个索引 i，low_phi lbl_lex.[i] 等于 lbl_vert.[morph.[i]]。img_lex_graph 的其余部分用于验证 morph 是否在 g_lex 和 g_vert 所描述的关联结构之间诱导出图态射。为此，算法检查对于所有索引 i，morph.[i] < 的长度是否等于 g_vert 的长度，以确保 morph 格式正确。然后，算法确保映射 morph 是满射。为此，算法利用证书 morph_inv : array int，该证书旨在作为 morph 的右逆，并检查对于 g_vert 的所有索引 i，是否有 morph.[morph_inv.[i]] == i。接着，算法处理边。它检查：如果 i j : int 在 g_lex 中相邻，那么 morph.[i] 和 morph.[j] 在 g_vert 中也相邻，除非 morph.[i] == morph.[j]（根据像的定义）。反过来，算法利用第三个证书 edge_inv，验证对于 g_vert 中任意两个相邻顶点 i'、j'，都存在 g_lex 中两个相邻顶点 i、j，使得 morph.[i'] == i 且 morph.[j'] == j。[5]

4.2 数据细化

为了证明底层实现相对于高层实现是正确的，我们采用项目 CoqEAL [14, 19] 中提出的方法，并使用数据细化。给定一个高层类型T 及其对应的低层类型 t，细化关系 *r* : t -> T -> 命题 是项之间的一种关系，这些项分别对应于同一对象的低层表示和高层表示。在这种设定下，如果两个函数在输入相关时返回的输出也相关，则这两个函数等价。形式化定义如下：

```
定义 rel_func
  (r1 : t -> T -> Prop) (r2 : u -> U -> Prop)
  (f : t -> u) (g : T -> U) :=
  (forall x y, r1 x y -> r2 (f x) (g y)).
```

我们使用记号 (r1 =~> r2) f g 表示 rel_func r1 r2 f g。

图 5 展示了为关联字典序图的低层表示与高层表示而执行的细化关系的组合，即基和矩阵上的图

[5]需要注意的是，`i'` 和 `j'` 在`morph_inv` 下的原像在`g_lex`中可能并不相邻，因此我们需要额外的证书`edge_inv`。

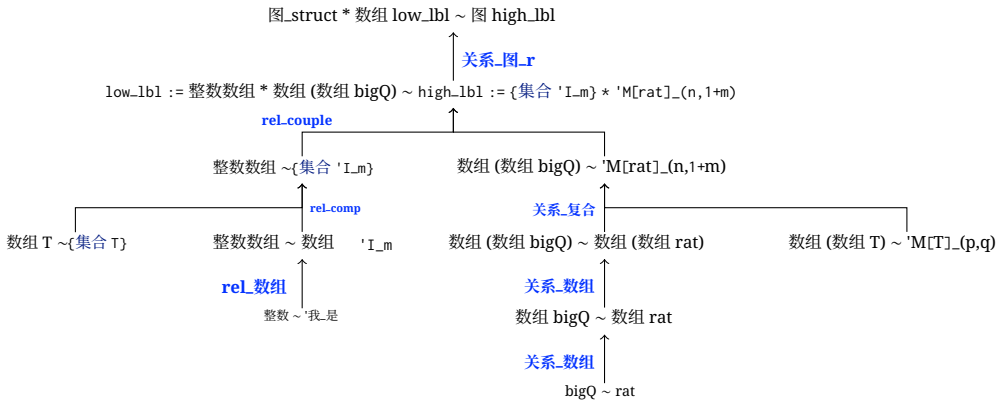


图5. 用于字典序图低层与高层表示之间的精化关系的构造。我们使用记号 `t ~ T` 表示低层类型 `t` 与高层类型 `T`之间存在精化关系。

的有序对。（这些关系的形式化在模块`refinement.v`中完成。）我们来谈谈这棵树。

树的叶子对应于原子类型之间的精化关系。对于每个这样的关系，我们需要证明低层类型和高层类型上基本操作的实现是等价的。例如，精化关系 `rel_整数_ord : int -> 'I_m -> 命题`将计算效率高的整数与类型为 `'I_m` 的 MathComp 序数（即小于固定整数 `m`的一元整数）联系起来，并且我们证明低层与高层序关系的等价性。该结论表述如下：

```
引理rel_整数_ord_lt :
  (rel_整数_ord  =~>rel_整数_ord =~>
  eq)(fun i j : int => (i  <? j)%
  int63)(fun i j :'I_m  => (i  < j)%nat)
```

其中 `eq` 是恒等关系。其他基本精化还包括关系 `rel_数组_set : array T -> {set T} -> 命题` 它连接了基于数组实现的有限集与对应的 MathComp 类型 `{set T}`，或者关系 rel_mx_col : array (array T) -> 'M[T]_(p,q) ->

命题 基于数组的矩阵与大小为 `p ×q` 的 MathComp 矩阵之间。最后，我们使用`rat_bigQ : bigQ -> rat -> Prop` CoqEAL 在类型 `bigQ` （计算效率高的有理数）与 MathComp 类型 `rat`（有理数）之间实现的精化关系。

树的边对应于函子。后者使我们能够组合精化，从而构造更复杂的关系。例如，函子 `rel_数组` lifts a refinement `r : t -> T -> 命题` to another between the types `array t` and`array T`:

```
定义rel_数组 r a A :=
  长度 a = 长度 A /\ 对于所有 i, i <长度 a -> r a。
  [i] A。[i]。
```

它附带了若干基本原语的数组_map `f a` 等价性证明，例如，使用函数`f`计算某个数组`a` 的像：

```
引理rel_数组_map r r' f F :(r =~>
r') fF ->(rel_array r  =~> rel_
arrayr')(arr_mapf) (arr_map F).
```

类似地，函子 rel_couple 将两个精化关系 r : t -> T -> 命题 和 r' : t' -> T' -> 命题 合并为乘积类型 t * t' 与 T * T' 之间的精化关系。第三个函子 rel_comp 使我们能够通过两个精化关系 r : t -> tT -> 命题 和 r' : tT -> T -> 命题，合成 t 与 T 之间的精化关系。例如，低级有理数矩阵的类型 array (array bigQ) 与高层方阵的类型 'M[rat]_n 之间的关系，可通过合成 bigQ 与 rat 的低级矩阵之间的关系，以及此前讨论过的、将 T 实例化为 := rat 的关系 rel_mx_col 得到。

如图 5 所示，低层与高层字典序图之间的精化关系，是将函子 rel_graph_r 应用于类型 low_lbl 与 high_lbl 之间的精化关系得到的（它们分别对应于字典序图标签的低层表示和高层表示）。函子 rel_graph_r 如图6所示。给定低层类型 t 与高层类型 T，它构造出类型 graph_struct 之间的精化关系

* 数组 t 由标记为 t 的低层图和标记为 T的高层图组成。它是将前面描述的函子应用于两个原子精化关系而构建的。第一个称为 rel_结构，将低层类型图_struct: = 关联数组的数组 与低层整数上的高层图类型整数图 关联起来。第二个将类型 整数图 * 数组 T（对应于一种混合表示，其中图的关联结构与到标签的映射分开存储）与

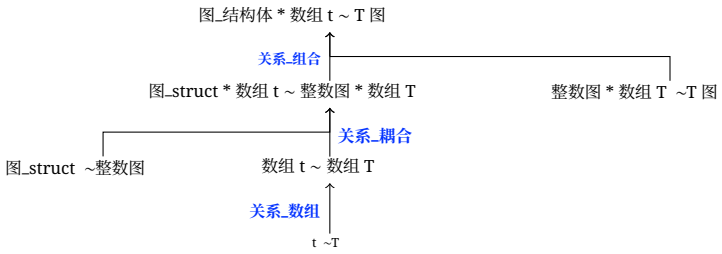


图6. 从细化关系t ~ T构造连接低层图和高层图的函子关系_图_r 。

高层图的类型 图 T关联起来。

4.3 等价性证明

现在，我们可以证明字典序图认证算法的低层实现与高层实现之间的等价性，即分别为 enum_algo 和 high_enum_algo；参见模块 enum_equiv.v。这涉及不等式系统定义的多面体的低层表示与高层表示之间的精化关系 rel_poly（在 polyType 与类型 'M[rat]_(m,n) * 'cV[rat]_m 的有序对 (A,b) 之间），以及第4.2节所述低层与高层字典序图之间的精化关系 rel_lex_graph。该等价性表述如下：

引理 **字典序_认证_等价性**:
(rel_多项式 =~>关系_词法_图 =~> 等式)
enum_algo high_enum_algo。

如果暂且不考虑直接明了的程序变换，那么证明实质上包括展开函数 enum_algo 和high_enum_algo中每个测试的定义，并利用低层与高层实现中相关基本操作的等价性，例如行向量的字典序比较、矩阵向量乘法或集合交。

类似地，函数img_lex_graph用于认证函数phi作用于字典序图所得像的底层函数的正确性，由以下陈述给出：

引理**img_lex_graph_等价性** (...) :
rel_lex_图 (g_lex, lbl_lex) G_lex ->rel_vert_图
(g_vert,lbl_vert) G_vert ->img_lex_图 morph
morph_inv 边_inv g_lex lbl_lex g_vert lbl_
vert ->G_vert =phi @/ G_lex.

其中rel_vert_graph 是针对带向量标签的图对rel_graph_r函子的一种实例化。（我们指出，由于morph、morph_inv 和edge_inv 提供给img_lex_graph的额外证书，这一陈述并非等价性。）

实际上，提供给算法的证书是用低层类型编写的。正如第 6 节所解释的，这些证书可能非常庞大，因此不可能使用高层类型对其进行编译。为了

利用我们先前讨论过的正确性陈述，每个精化关系 rel_t_T : t -> T ->Prop 都附带一个额外的函数 spec_t_T : t -> T，它能根据一个格式良好的低层项构造出一个高层项，因此我们有 r x (spec_t_T x)。由此，我们得到以下陈述，它

定理 **验证**Po (...) :
(...) 枚举_algo Po g_lex lbl_lex ->img_lex_图
morph morph_inv 边_inv g_lex lbl_lex g_
vert lbl_vert ->poly_图 (poly_of_syst (spec_
poly Po)) =spec_vert_图 (g_vert, lbl_vert).

输入包括底层证书，以及这些证书由算法的底层实现所认证的假设。它证明，对于由底层不等式系统 多面体_的_系统（规格_多面体 Po）所描述的高层多面体 Po，其（高层）顶点-边图与底层图 (g_vert, lbl_vert)相对应。该证明结合了高层算法的正确性（定理 repr_poly_graph）与底层实现的正确性（即，引理 lex_certif_equiv 以及引理 img_lex_graph_equiv）。

5 赫希猜想的形式化反证

本节旨在说明，我们如何利用第 4 节介绍的认证算法及其底层实现，得出对赫希猜想的形式化反证。回顾一下，赫希猜想涉及三个概念：顶点-边图的直径、面数以及多胞体的维数。我们通过计算直径和维数的下界，以及面数的上界来反驳该猜想，最后利用传递性得出结论。我们先说明如何计算每个量。

考虑多面体 $P = \{x \in \mathbb{R}^n : Ax \geq b\}$，其中 $A \in \mathbb{R}^{m \times n}$ 且 $b \in \mathbb{R}^m$。众所周知，面数受不等式数量 m 的限制。这一结论已加入 Coq-Polyhedra 中已确立的多面体性质（参见模块poly_base.v），并表述如下：

引理**面_le** {base : base_t[R,n]}:

(#|`多面体 'P(base) | <= #|` base)%nat.
项 base 对应于一组定义该多面体的不等式，而 'P(base) 是表示该多面体的项。

为处理维数问题，我们证明，Hirsch 猜想的（非形式化）反例的维数等于 n（即环境空间的维数）。为此，我们展示多胞体的一组 $n+1$ 顶点 $v^0, \ldots, v^n \in \mathbb{R}^n$，它们是仿射无关的，也就是说，矩阵 $M = \begin{bmatrix} v^1 - v^0 & \ldots & v^n - v^0 \end{bmatrix}$ 是非奇异的。这些点通过（低层级）证书映射 _lbl : 整数数组 和 origin : 整数 给出，它们表示这些点在低层级顶点-边图中的索引集合；此外，还给出了一个非形式化计算得到的低层级矩阵 inv_lbl，我们将验证它是矩阵 M 的逆矩阵。在低层级一侧，这些证书的验证由函数 dim_full_test （位于 enum_algo.v 中）执行，其正确性陈述如下（参见 enum_high_algo.v）：

```
引理 high_维_h :
  维_全_测试映射_lbl 原点 inv_lbl ->
  \pdim P = n.+1。
```

其中 \pdim P 表示多胞体 P 的维数加一（Coq-Polyhedra 采用空集的维数为 0、点的维数为 1 等约定）。

最后，通过给出图中的一个顶点 v 来计算直径的下界。该顶点的离心率，即它与 v 任意其他顶点 w 之间的最大距离，达到直径的值。为计算离心率，我们利用广度优先搜索（BFS）算法，并在模块 diameter.v 中通过底层和高层数据结构实现该算法。底层实现由函数 Low.BFS : graph_struct -> int -> NArith.BinNat 提供，该函数以底层关联数组和一个顶点的索引作为输入，并返回从该顶点出发的最短路径的最大长度。类似地，高层实现是函数 High.BFS ，它使用类型 graph T。与底层实现的对应关系如下所示：

```
引理rel_struct_广度优先搜索 (g : 图_struct) (G : 图
    整数):
  关系_结构 g G ->
    对所有 x，mem_顶点 g x->
      Low.BFS g x = High.BFS G x :> 自然数.
```

现在，我们可以通过对赫希猜想的两个反例运行认证算法，正式证伪这一猜想；这两个反例见 [36]。这分别在 test/data/poly20dim21 和 test/data/poly23dim24 目录中完成（我们参阅 README 文件，从 test/archives 中的压缩档案提取它们）。我们利用 Weibel 提供的这些反例的显式不等式表示。[6] 它们的形式为

[6] https://sites.google.com/site/christopheweibel/research/hirsch-conjecture

供该库使用的输入文件 lrslib [4]。我们将此库与额外的 Python 脚本结合使用，为我们的算法生成非形式化证书（它们存储在 test/data/polyXXdimYY/coq/）。如第 4.3 节所述，这些证书采用低层类型提供。下面简要介绍每个证书：

- poly.v 包含用不等式描述多胞体的内容；
- g_lex.v 和 lbl_lex.v（分别对应 g_vert.v 和 lbl_vert.v）旨在表示字典序图（分别对应顶点-边图）；
- morph.v，morph_inv.v 和 edge_inv.v 是提供给函数 img_lex_graph （参见第 4.1 节）；
- map_lbl.v、origin.v 和 inv_lbl.v 是函数 dim_full_test 所需维数的证书；
- cert.v 提供用于检查输入多面体是否为多胞体的证书。这些证书用于证明，定义该多面体的不等式的非负组合可得出形如 $-K \le x_i \le K$ 的不等式，其中 K 是一个足够大的常数。验证由函数 bounded_Po_test（模块 enum_algo.v）执行。
- start.v 是用于获得直径下界的顶点索引。

每个文件都使用 coqc 编译，然后导入，以获得 test/data/polyXXdimYY/coq_Hirsch/Hirsch.v 中对该猜想的形式化反证：

```
定理 赫希_是_错误的:
  存在 (d : 自然数) (P :'poly[rat]_
  d),(High.diameter (poly_图 P)> #|`面 P| -
  (\pdimP).-1)%nat.
证明。
设 P: = poly_由_syst(A, b)).
存在 n'。+1, P。
应用/反证_赫希。
- exact: well_formedness_ok。-
exact: enum_algo_ok。- exact:
img_图_ok。- exact: bounded_
Po_test_ok。- exact: dim_full_
test_ok。- exact: 直径_check_
ok。
Qed.
```

（注意，由于 Coq-Polyhedra 中采用了维数加一的约定，陈述使用 (pdim P).-1 表示维数。）证明首先给出存在性陈述的见证，即 poly.v 中提供的多面体。随后应用 定理 disprove_赫希 ，该定理证明，只要所有认证测试均返回真，猜想便不成立（见文件 theories/enum_equiv.v）。证明最后六行验证了这些假设，分别对应于：(i) 所有输入证书的良构性；(ii) 字典序图；

(iii) 顶点-边图；(iv) 多面体的有界性；(v) 多面体的维数；(vi) 直径的下界。每项测试都使用 Coq 的策略`vm_compute`完成，该策略会计算每项测试的（布尔）结果。

正如引言中所述，[39], 也采用了 [36], 的方法，通过将特殊的低维纺锤体提升到更高维数，构造非 Hirsch 多胞体。这只能给出非 Hirsch 多胞体直径的下界；参见 [39, 定理 1。5]。除了验证下界外，我们还正式证明了20维和23维反例的直径恰为 [36]:

定理**poly20dim21_直径**：
直径 (poly_图 poly20dim21) = 21%nat.
定理**poly23dim24_直径**：
直径 (poly_图 poly23dim24) = 24%nat.
由于这两个多胞体分别有 40 个和 46 个面，因此它们的直径恰好达到下界。据我们所知，这是首次证明这一事实。

6 实际实验

为了确保实验结果可复现，我们在两种不同的架构上进行了实验：(i) 一台搭载 Apple M1 处理器、配有 32 GB 内存并运行 Mac OS 11.0.1 的机器（架构 A）；(ii) 一台搭载 2.3GHz Intel Core 处理器、配有 64 GB 内存并运行 Linux 5.4 的机器（架构 B）。两者均使用 Coq 8.16.1。

如引言中所述，[36]的两个反例是非常复杂的多胞体，包含数万个顶点和数十万条边。此外，顶点的有理系数很长，分子和分母最长可达 40 位。因此，若将证书以纯文本形式显式写入文件，会生成非常大的 .v 文件，编译这些文件对 Coq 来说是一项艰巨（甚至不切实际）的任务。例如，对于 23 维反例，我们需要约 600 MB 来存储纯文本 .v 文件中的项 lbl_lex，该项包含字典序图的标签。在这种情况下，coqc 在编译文件时的内存使用量成为主要限制因素。在架构 B 上，编译因内存耗尽而失败。得益于更好的内存管理（内存压缩和使用交换内存），在架构 A 上编译成功了，但耗时超过 8 000 秒。为解决这些编译问题，我们实现了一个名为 BinReader[7] 的 Coq 插件，它提供命令 LoadData <filename.bin> As <term_id>，用于根据从二进制文件 <filename.bin>读取的描述构建 Coq 项 <term_id>。该插件支持第 4.1 节中描述的数据类型的所有可能组合。实际使用中，构建项所需的时间微不足道，主要耗时步骤是由 Coq 内核对该项进行类型检查。举例来说，我们成功构建了此前讨论的项 `lbl_lex`，在架构 A 上耗时约 8 分钟，在架构 B 上耗时 13 分钟，总内存使用量限制在 50 GB。

表 1 列出了在架构 A 上处理各个实例所用的总 CPU 时间（架构 B 的速度约慢一倍）。实例 `cube-n` 和 `cross-n` 分别对应 n维立方体和交叉多胞体。实例 `cyclic-p-n` 是 n 维、由 p 个面定义的循环多胞体的对偶多胞体；正式描述见下文。实例 `spindle-n` 对应于 [36, 39] 中用于构造更高维赫希猜想反例的原始小维数纺锤体。最后，`poly20dim21` 和`poly23dim24` 分别是 [36]给出的 20 维和 23 维赫希猜想反例。表1 的第二列给出了 `coqc` 使用插件`BinReader`载入证书所需的时间（包括根据二进制文件构造项所需的时间，以及内核对这些项进行类型检查所用的时间）。第三列给出了使用第4节中的算法检查证书所需的时间。最后一列给出了在 Coq 中计算多胞体精确直径所需的时间。对于`polyXXdimYY`实例，我们还以斜体列出了按照第5节所述计算直径下界所需的时间。需要指出的是，实际中可以通过并行化（例如使用构建系统`dune`）缩短编译时间，因为证书检查和直径计算可以独立进行（直径是在非形式化图上计算的）。

正如我们之前提到的，证书测试结果的计算是通过 Coq 中的策略 `vm_compute` 完成的 [28].[8] 后者依赖于一种基于优化字节码的虚拟机；在完整求值计算量大的函数应用时，它的性能优于 Coq 内核的抽象归约机制。我们未能使用效率更高的策略`native_compute` [9]（该策略会将代码提取到某个 OCaml 文件中，再进行编译并以本机代码执行）。事实上，由于项的规模过大，该策略执行失败并引发了异常。使用 `vm_compute`，针对 20 维反例的赫希猜想形式化反证耗时约 1 小时 22 分钟，针对 23 维反例则耗时约 1 小时 55 分钟。由于这些耗时相对较短，我们还通过将 `vm_compute` 替换为 Coq 的标准按值调用转换策略 `cbv`，重复了相同的实验。我们成功对这两个反例进行了形式化验证，耗时约增加了 20 倍。

最后，我们给出其他实例的数学描述。 n维立方体由满足以下条件的点组成

[7] https://github.com/Coq-Polyhedra/coq-binreader

[8] 更准确地说，我们使用策略 `vm_cast_no_check`，它依赖于`vm_compute`，但只有在内核对证明项进行类型检查时才执行计算，也就是说，在执行命令 `Qed` 时。

表1。 ArchitectureA 上认证过程主要步骤的 CPU 时间总计（秒）。

实例	证书加载	证书检查	直径计算
cube_2	7.85	6.40	1.07
cube_3	7.90	6.30	1.08
cube_4	7.93	6.31	1.09
cube_5	9.18	6.31	1.16
cube_6	8.32	6.36	1.15
cube_7	8.33	6.45	1.18
交叉_2	8.54	6.27	1.21
交叉_3	8.11	6.73	1.10
交叉_4	8.17	6.36	1.14
交叉_5	8.65	6.99	1.11
交叉_6	14.97	17.33	1.19
交叉_7	113.32	265.15	1.14
循环_12_6	8.37	6.52	1.22
循环_20_10	25.71	45.87	36.77
纺锤体_25	8.93	7.06	1.16
纺锤体_28	10.74	6.90	1.21
纺锤体_48	10.88	8.70	1.22
poly20dim21	404.01	4490.89	5295.91 *(17.57)*
poly23dim24	1147.00	5736.19	24 551。46 *(14.75)*

$x \in \mathbb{R}^n$ 对所有 $i \in [n]$均满足 $-1 \le x_i \le 1$。它有 2^n 个顶点。 n维交叉多胞体是满足 $\sum_{i=1}^{n} |x_i| \le 1$的点集$x \in \mathbb{R}^n$。它可以用 2^n个形如 $\sum_{i=1}^{n} \pm x_i \ge -1$的不等式来描述。挑战在于它有许多退化基：每个顶点都对应 2^{n-1} 个可行基。这种重要的退化现象反映在计算字典序图所需的时间上。给定$p \ge 1$，即循环多胞体的极对偶，记作 $C(n,p)$，其构造方法是选取 p 个实数 $t_1 < \cdots < t_p$，并考虑对所有 $i \in [p]$均成立的不等式$c^i - \bar{c}, x \le 1$，其中$c^i := (t_i, t_i^2, \ldots, t_i^n)$ 且 $\bar{c} := \frac{1}{p}\sum_{i=1}^{p} c^i$。McMullen 上界定理 [37] 指出，在具有 p 个面的 n维多胞体中，多胞体 $C(n,p)$的顶点数最多。这个数量可明确写为：

$$\binom{p - \lceil \frac{n}{2} \rceil}{\lfloor \frac{n}{2} \rfloor} + \binom{p - 1 - \lceil \frac{n-1}{2} \rceil}{\lfloor \frac{n-1}{2} \rfloor}$$

我们已确认，经认证的图恰好包含预期数量的顶点。

7 结论

我们开发了实用高效的算法，通过检查非正式计算得到的证书，使我们能够计算多胞体的顶点-边图。设计的简洁性是实现这一目标的关键特征。我们通过形式化验证 [36] 的两个反例，给出了对著名 Hirsch 猜想的形式化否证。尽管这些反例具有非常复杂的组合结构，我们的方法仍能在数小时内借助证明助手 Coq 完成这一否证。据我们所知，这是首项展示如何在证明助手中实际完成关于多面体的计算要求较高的证明的工作。

作为未来的工作，我们旨在大幅提升我们的方法的性能，使其更接近非形式化软件的性能。为此，我们计划利用矩阵的秩一更新和整数主元变换（见 [5]及其中的参考文献），以缩小证书的规模，并降低我们算法中矩阵-向量乘法的计算复杂度。我们还希望，本文所介绍的这类计算密集型方法能让更多人关注证明助手的计算性能。为此，我们计划在解决大型项处理问题后，利用效率更高的策略 `native_compute` [9]。目前，我们的方法仅适用于多胞体。为求简便，我们作了这一限制；我们计划将多面体在邻接图中的无界射线纳入考虑，以扩展我们的技术。最后，我们计划研究本文 [14] 所述的等价性证明自动演绎技术在多大程度上有助于改善我们算法的底层实现与高层实现之间的联系。

致谢

第一作者衷心感谢 Ricardo D. Katz 就这一主题进行的有益讨论。作者感谢 CPP'23 的匿名审稿人提出的意见和建议。

参考文献

[1] Xavier Allamigeon、Quentin Canu 和 Pierre-Yves Strub。2023年。Hirsch猜想的形式化反证。载于第十二届 *ACM SIGPLAN* 国际认证程序与证明会议（美国宾夕法尼亚州费城）（*CPP 2023*）。计算机协会，纽约，美国，13–26页。https://doi.org/10.1145/3573105.3575678[2] Xavier Allamigeon 和 Ricardo D. Katz。2019年。基于单纯形法的凸多面体形式化。《自动推理期刊》63卷第2期（2019年），323–345页。https://doi.org/10.1007/s10817-018- 9477-1[3] Nina Amenta 和 Gunter M Ziegler。1999年。多胞体的变形积与最大阴影。《当代数学》223卷（1999年），57–90页。[4] D. Avis。 [无日期]。lrslib 库。http://cgm.cs.mcgill.ca/~avis/C/lrs. html。[5] David Avis。2000年。反向搜索顶点枚举算法的修订实现。载于《多胞体——组合学与计算》，Gil Kalai 和 Günter M. Ziegler（编）。Birkhäuser Basel，巴塞尔。https://doi.org/10.1007/978-3-0348-8438-9_9

[6] David Avis 和 Komei Fukuda。1992年。用于凸包以及超平面配置和多面体的顶点枚举的枢轴算法。《离散与计算几何》8卷第3期（1992年）。https://doi.org/10.1007/ BF02293050[7] Seul Baek。2019年。Lean 中的反射式判定过程。硕士论文。卡内基梅隆大学。[8] Roberto Bagnara、Patricia M. Hill 和 Enea Zaffanella。2008年。Parma 多面体库：面向硬件和软件系统分析与验证的完备数值抽象集。《计算机编程科学》72卷第1–2期（2008年），3–21页。[9] Mathieu Boespflug、Maxime Dénès 和 Benjamin Grégoire。2011年。全速完全归约。载于《认证程序与证明》，Jean-Pierre Jouannaud 和 Zhong Shao（编）。Springer Berlin Heidelberg，柏林、海德堡，362–377页。[10] Sylvain Boulmé、Alexandre Maréchal、David Monniaux、Michaël Périn 和 Hang Yu。2018年。验证多面体库：概述。载于《2018年第20届科学计算符号与数值算法国际研讨会》（SYNASC）。9–17页。https://doi.org/10.1109/ SYNASC.2018.00014[11] Thomas Braibant 和 Damien Pous。2010年。在 Coq 中判定克莱尼代数。载于 ITP（LNCS，第6172卷）。Springer，爱丁堡，英国，163 178 10 1007 978 3 642 14052 5_13 – 。2008
[12] https://doi.org/ 。/ - - - - Amine Chaieb 和 Tobias Nipkow。 。线性算术的证明综合与反射。《自动推理期刊》41卷第1期（2008年），33–59页。https: //doi.org/10.1007/s10817-008-9101-x [13] Adam Chlipala。2013年。依赖类型的认证编程：Coq 证明助手的实用入门。麻省理工学院出版社。http: //adam.chlipala.net/cpdt/ [14] Cyril Cohen、Maxime Dénès 和 Anders Mörtberg。2013年。免费的精化！载于《CPP 2013论文集》，Georges Gonthier 和 Michael Norrish（编）。Springer。[15] Sylvain Conchon 和 Jean-Christophe Filliâtre。2007年。一种持久化并查集数据结构。载于《2007年ML研讨会工作坊论文集》（德国弗赖堡）（ML '07）。计算机协会，纽约，美国，37–46页。https://doi.org/10.1145/1292535.1292541[16] P. Cousot 和 N. Halbwachs。1978年。自动发现程序变量之间的线性约束。载于《POPL 1978论文集》。ACM出版社，亚利桑那州图森。[17] G. B. Dantzig。1951年。在《生产与分配的活动分析》中，最大化满足线性不等式约束的变量线性函数。Wiley。[18] George B. Dantzig、Alex Orden 和 Philip Wolfe。1955年。在线性不等式约束下最小化线性形式的广义单纯形法。《太平洋数学杂志》5卷第2期（1955年）。[19] Maxime Dénès、Anders Mörtberg 和 Vincent Siles。2012年。Coq 中计算代数的基于精化的方法。载于《交互式定理证明》，Lennart Beringer 和 Amy Felty（编）。Springer Berlin Heidelberg，柏林、海德堡，83–98页。[20] Yann Disser、Oliver Friedmann 和 Alexander V. Hopp。2022年。Zadeh 枢轴规则的指数下界。《数学规划》（2022年）。[21] Alexis Fouilhe 和 Sylvain Boulmé。2014年。（次）多面体抽象域的认证前端。载于《VSTTE 2014》，Dimitra Giannakopoulou 和 Daniel Kroening（编）。Springer，200–215页。[22] K. Fukuda。[无日期。]。CDD 库。https://github.com/cddlib/cddlib。[23] K. Fukuda 和 A. Prodon。1996年。双描述法再探。载于第八届法日组合学与计算机科学会议及第四届法中组合学与计算机科学会议精选论文集。施普林格出版社，英国伦敦，91–111页。[24] Ewgenij Gawrilow 和 Michael Joswig。2000年。polymake：用于分析凸多胞体的框架。载于《多胞体——组合学与计算》（奥伯沃尔法赫，1997年）。DMV 研讨会丛书，第29卷。Birkhäuser，巴塞尔，43–73。[25] Georges Gonthier。2008年。《四色定理：形式化证明的构建》。载于《计算机数学》，Deepak Kapur（编）。Springer Berlin Heidelberg，柏林，海德堡，333–333。[26] Georges Gonthier 和 Assia Mahboubi。2010年。《Coq 中小规模反射简介》。《形式化推理期刊》3，2（2010年），95 152 00515548 – 。
[27] https://hal.inria.fr/inria- Georges Gonthier、Assia Mahboubi 和 Enrico Tassi。2016年。《Coq 系统的小规模反射扩展》。研究报告 RR-6455。Inria Saclay Ile de France。[28] Benjamin Grégoire 和 Xavier Leroy。2002年。《强归约的编译实现》。《SIGPLAN 通知》37，9（2002年9月），235–246。https://doi.org/10 1145 583852 581501 。 / 。Benjamin Gregoire 和
[29] Assia Mahboubi。2005年。《在 Coq 中正确证明交换环中的等式》。载于 TPHOLs 2005（《计算机科学讲义》，第3603卷），Joe Hurd 和 Tom Melham（编）。Springer，牛津，英国，98–113。https://doi.org/10.1007/ 11541868 7_[30] Benjamin Grégoire 和 Laurent Théry。2006年。《用于模算术及其在大素数认证中应用的纯函数式库》。载于《自动推理》，第三届国际联合会议 IJCAR 2006，西雅图，美国，2006年8月17—20日，会议论文集（《计算机科学讲义》，第4130卷）。Springer，423–437。[31] Nicola Guglielmi、Linda Laglia 和 Vladimir Protasov。2017年。《稳定和可稳定化 LSS 的多胞体李雅普诺夫函数》。《计算数学基础》17，2（2017年）。https://doi.org/10.1007/ s10208-015-9301-9[32] Dimitri Hendriks。2002年。《Coq 中的证明反射》。《自动推理期刊》29（2002年），277–307。[33] Daniel W. H. James 和 Ralf Hinze。2009年。《Coq 中格的基于反射的证明策略》。载于第十届函数式编程趋势研讨会论文集，TFP 2009，斯洛伐克 Komárno，2009年6月2—4日（《函数式编程趋势》，第10卷），Zoltán Horváth、Viktória Zsók、Peter Achten 和 Pieter W. M. Koopman（编）。Intellect，97–112。[34] Leonid Khachiyan、Endre Boros、Konrad Borys、Khaled Elbassioni 和 Vladimir Gurvich。2008年。《生成多面体的所有顶点是困难的》。《离散与计算几何》39，1（2008年3月1日），174–190。https://doi.org/10 1007 00454 008 9050 5 。 /s - - - Victor Klee 和
[35] David W. Walkup。 .《维数为 d 的多面体的 d 步猜想》<6。《数学学报》117，none（1967年），53 – 1967
78 10 1007 02395040 。https://doi.org/ 。 /BF
[36] Benjamin Matschke、Francisco Santos 和 Christophe Weibel。2015年。《五维棱柱拟多面体的宽度》。《伦敦数学会会刊》110，3（2015年），647–672。https://doi.org/10.1112/ plms/pdu 054
[37] P. McMullen。1970年。《凸多面体面的最大数量》。《Mathematika》17，2（1970年），179–184。https://doi.org/10.1112/ S0025579300002850[38] T.S. Motzkin、H. Raiffa、G.L. Thompson 和 R.M. Thrall。1953年。《双描述法》。载于《博弈论文集》，H.W. Kuhn 和 A.W. Tucker（编），第二卷。51–73。[39] Francisco Santos。2012年。《赫希猜想的一个反例》。《数学年刊》176，1（2012年）。[40] Alexander Schrijver。1986年。《线性与整数规划理论》。John Wiley & Sons, Inc.，美国纽约州纽约市。[41] Steve Smale。1998年。《下个世纪的数学问题》。《数学智者》20（1998年）。[42] Mirko Spasić 和 Filip Marić。2012年。《通过逐步精化形式化增量单纯形算法》。载于 FM 2012 论文集，Dimitra Giannakopoulou 和 Dominique Méry（编）。Springer。[43] William A. Stein 等。2020年。《Sage 数学软件》（版本 9.0）。Sage 开发团队。http://www.sagemath.org。

[44] Paul van Der Walt 和 Wouter Swierstra。2012年。《在 Agda 中通过反射构建证明》。载于 *IFL*——第*24*届函数式语言实现与应用国际研讨会（《计算机科学讲义》，第*8241*卷），Ralf Hinze（编）。Springer，牛津，英国，157–173。 https://doi.org/10.1007/978-3-642-41582-1_10

[45] Friedrich W. von Henke、Stephan Pfab、Holger Pfeifer 和 Harald Rue。1998年。元层定理证明中的案例研究。载于 高阶逻辑中的定理证明，Jim Grundy 和 Malcolm Newey（编）。Springer Berlin Heidelberg，柏林，海德堡，461–478。[46] G. M. Ziegler。2012年。谁证明了赫希猜想？《数学文献》 (2012)，75—85。增刊“优化故事”。