一种用于量词消去的多算法方法

1st James H. Davenport巴斯大学计算机科学系巴斯，英国。电子邮件：J.H.Davenport@bath.ac.uk ORCID 0000-0002-3982-7545

2nd Zak P. Tonks巴斯大学计算机科学系巴斯，英国。电子邮件：Zak.P.Tonks@bath.edu ORCID 0000-0001-6019-1775

3rd Ali K. Uncu*RICAM*、奥地利科学院及巴斯大学奥地利及英国。电子邮件：aku21@bath.ac.uk ORCID 0000-0001-5631-6424

arXiv:2302.06814v2 [cs.SC] 7 Dec 2023

摘要——**柱状代数分解（CAD）是首个用于实数域量词消去（QE）的实用方法，至今仍是一种重要方法。尽管如此，其复杂度相对于变量数量本质上呈双指数增长。在适用的情况下，虚拟项代换（VTS）效率更高，一次应用即可将含有 n 个变量的量词消去问题转化为含有 $n-1$ 个变量的问题，依此类推。因此，混合方法大有可为：先在可能的情况下使用 VTS，再使用 CAD。**

本文介绍了这样一种多算法实现，其基础是第二作者的博士论文。所采用的 CAD 基于 Lazard 方法的一种新实现，并进行了改进以处理等式约束，这与以往柱面代数分解算法中处理等式约束的方法类似。

索引词—**量词消去、柱状代数分解、虚拟项替换**

一、引言

第二作者的学位论文 [Tonks(2021)]为 Maple 开发了一个新软件包 QuantifierElimination，整合了量词消去（QE）研究的各个方面，以探究将它们结合使用时的细微差别。该项目重点全面研究虚拟项代换 [Weispfen- ning(1988)] (VTS)与柱状代数分解[McCallum 等人(2019)] (CAD)之间的多算法方法，并为 QE 提供更高的效率和更多功能。该软件包包含与 Maplesoft 合作开发的首个 Maple 版 VTS 实现、首个 Maple 版 Lazard 风格柱面代数分解实现，以及首个在任何场景下实现的、带有等式约束（EC）优化的 Lazard 风格柱面代数分解。这也包括关于 Lazard 风格柱面代数分解中的幕的最新研究 [Nair(2021)]。图 2 显示，在[Tonks(2020a)], 的基准测试中，多算法方法比单纯使用 CAD 更快；而利用单个 EC（§III-B）的 CAD，也比未对 EC 作特殊处理的 CAD 更快。所幸，基准测试集中的多个 EC 很少见，因为相关理论 [Nair(2021)] 在这方面尚不完善。

第二作者的学位论文获得了英国工程与自然科学研究委员会（EPSRC）资助项目 EP/N509589/1 和 Maplesoft 的支持。第一作者和末位作者获得了英国工程与自然科学研究委员会资助项目 EP/T015713/1 的部分支持。末位作者还感谢奥地利科学基金（FWF）项目 P34501-N 提供的部分资助。

令每个 Q_i 取自 $\forall$，$\exists$。实数域量词消去问题如下：给定一个陈述 Φ_0:=

$$Q_1 x_{1,1},\ldots,x_{1,k_1}\cdots Q_{a+1}x_{a+1,1},\ldots,x_{a+1,k_{a+1}}\Phi(y_i,x_{i,j}), \quad (1)$$

其中 Φ 是 $P_\alpha(y_i,x_{i,j})\rho_\alpha 0$的一个布尔组合，其中$P_\alpha$是多项式，且 $\rho_\alpha\in\{=,\neq,<,\leq,>,\geq\}$，生成由多项式 $P_\beta(y_i)$之间的等式和不等式构成的布尔组合 Ψ，使其与原式等可满足；也就是说， Ψ 为真当且仅当 Φ_0 为真。如果 $\Phi(y_i,x_{i,j})$中的所有多项式 $P_\alpha(y_i,x_{i,j})$均具有整数系数，我们称 $\Phi(y_i,x_{i,j})$为塔斯基公式。

有了 Ψ 就意味着量词消去问题已解决。部分解（即消去一些量词后得到 Ψ

$$\Phi_1 := Q_1 x_{1,1},\ldots,x_{1,k_1}\cdots Q_b x_{b,1},\ldots,x_{b,l_b}\hat{\Phi}(y_i,x_{i,j}), \quad (2)$$

以及 $l_b\leq k_b$）称为中间量词消去结果（IQER） [Tonks(2021), 定义 9]。

假设 $Q_i\neq Q_{i+1}$， a 是量词交替的次数。令 n_0 为自由变量的数量 y_i，令 n为变量 y_i 和$x_{i,\ j}$的总数。

量词消去是否可行，远非显而易见。Collins 首次展示了其实际可行性 [Collins(1975)]，他采用了柱形代数分解方法（见 §III）。

本文概述了 QuantifierElimination 软件包背后的基础知识。§II 和 §III 分别概述 VTS 和柱形代数分解。§IV 定义并讨论多算法方法。§V 讨论基准测试和一些实验结果。§VI 总结全文并展望未来工作。

II. 虚拟项代换

虚拟项代换（VTS）是一种处理塔斯基公式为纯存在量词形式的量词消去问题的方法：参见 [Weispfenning（1988）]。当相应变量的次数较低时，VTS 可以消去量词。线性和二次消去见于 [Weispfenning（1988）], [、Weispfenning（1997）], 以及 [Koˇsta（2016）] ，后者阐述了对以三次形式出现的变量进行消去。

VTS 是针对整系数多项式制定的。因此，本节默认假设我们的多项式具有整数系数，或至少具有有理数系数。

特别是，公式中的所有约束都必须满足这样的条件：相关多项式可分解为次数不超过 3 的多项式，也就是说，次数限制实际上仅适用于不可约因子。由于 VTS 在次数和系数基环方面的使用限制，一些研究者将 VTS 视为其他量词消去（QE）算法之前的预处理步骤（见 [Strzebo´nski(2022)]）。QuantifierElimination 首次在 Maple 中实现了截至并包括三次虚拟项代换的情形 [Davenport et al.(2022)] 。

VTS 围绕从带量词公式中生成的测试点展开，这些测试点描述了多项式（可能是多元多项式）在特定变量上的实根。这类结构测试点可能表示对某个多项式实根的精确代入。或者，这些结构测试点也可能包含 ∞ 或无穷小量 ε，尤其是在存在强关系且 $\rho \in \{<, >, \neq\}$ 的情况下。对于 ε，必须提供多项式的根描述，而测试点隐含地表示“代入一个略小于 $(-\varepsilon)$／略大于 $(+\varepsilon)$ 该实根的值”。

VTS 的另一个要素是守卫条件。守卫条件定义了从结构测试点到塔斯基公式的映射，其中每个守卫条件表示结构测试点要具有关联性必须满足的条件。这使得使用多元多项式的代入点更具可信度，因为这类多项式相对于任一变量所呈现的实根数量可能会发生变化。因此，将测试点虚拟代换到任何公式中，都取决于其守卫条件的结果，并且必须将守卫条件与虚拟代换的结果合取。

对于给定函数 $f(x)$，将 a 代入 x 位置的代数代换记作 $f[x / a] := f(a)$。同理，我们将 a 在 x 中的虚拟代换记作 $f[x // a]$。

定义1 (虚拟项代换 [Kˇsta(2016)，第2节。3])。将结构测试点 t 代入无量词关系 $g(x, \mathbf{y})\ \rho\ 0$ 中的 x，其虚拟项代换为 $F(\mathbf{y}) := g[x // t]$，这是一个无量词公式，满足：对于任意参数值 $\mathbf{a} \in \mathbb{R}^{n-1}$，若 $\mathbf{a}$ 满足 t的守卫条件，则 $\mathbf{a}$ 满足 F 当且仅当 $\mathbb{R} \models (g\ \rho\ 0(\mathbf{a}\ t[\mathbf{y}\ /\mathbf{a}])$。

如果我们有一个非原子塔斯基公式 $\Phi(x, y)$，而非单个关系 $g(x, \mathbf{y})\ \rho\ 0$，那么根据定义1，VTS将递归地进行 [Kˇsta(2016)]。

总之，对于形如 $\exists x \Phi(x, \mathbf{y})$ 的给定存在量化塔斯基公式，VTS 通过显式对应关系，识别有限集 $T = \{(\gamma_i, t_i)\}$，其中包含若干对：γ_i 是守卫条件，t_i 是实轴上每个区间的样本测试点，并据此消去 x 上的量词

$$\exists x\ \Phi(x, \mathbf{y}) \Leftrightarrow \bigvee_{(\gamma_i, t_i) \in E} (\gamma_i \wedge \Phi[x // t_i])\,. \tag{3}$$

对于全称量化问题，可以先对问题取反，再应用 VTS：$\forall x\, \Phi(x) \equiv \neg\exists x\, \neg\Phi(x)$。

多变量虚拟代入围绕着每次对一个变量依次代入结构测试点展开。这意味着，我们可以轻松地从任一量词块内测试点的代入中继承一种规范树结构。这棵树的分支是结构测试点，规范树的每一层对应一个量化变量，而节点则是因虚拟代换的连续结果而形成的公式。这些节点对应中间量词消去结果（IQER）。

III. 柱形代数分解

柱状代数分解（CAD）由 Collins [Collins(1975)] 引入。符号不变 CAD 的关键在于，多项式在每个单元上的符号都相同，因此也与该单元的样本点符号相同。考虑一个位于 $n-1$ 个变量空间中的单元 C，即除 $x_{a+1,\ k_{a+1}}$ 外的所有变量，其样本点为 s。它上方有若干单元 C_1，……，C_m，其样本点为 s_1，……，s_m，这些样本点的前 $n-1$ 个坐标与 s 的相同。根据 $Q = Q_{a+1}$:

$Q = \exists$如果 s_1、……、s_m 中有任何一个满足 Φ，那么显然 $\exists x_{a+1}$、$_{k_{a+1}}\Phi$ 在 s 处为真（并且在整个 C 上都为真）。反过来，如果没有样本点满足 Φ，那么根据符号不变性，C_1、……、C_m 中没有任何一个满足 Φ，因此 $\exists x_{a+1}$、$_{k_{a+1}}\Phi$ 在 s 处为假（并且在整个 C 上都为假）；$Q = \forall$如果 s_1、……、s_m 全都满足 Φ，那么根据每个单元上的符号不变性，Φ 在每个点 $(s, x_{a+1},$ $_{k_{a+1}})$ 处都为真，因此 $\forall x_{a+1}$、$_{k_{a+1}}\Phi$ 在 s 处为真；同样根据符号不变性，它在整个 C 上都为真。反过来，如果某个 s_i 不满足 Φ，那么根据符号不变性，它在整个 C_i 上都不满足；并且由于 C_i 投影到 C 上，$\forall x_{a+1}$、$_{k_{a+1}}\Phi$ 在整个 C 上都为假。

因此，我们可以确定 $Q_{a+1} x_{a+1,\ k_{a+1}}\Phi$ 在 C 上的真值，实际上也能确定其在 $\mathbb{R}^{n-1}$ 中所有单元上的真值。随后我们以归纳方式继续，逐步确定直到 $\mathbb{R}^{n_0}$的真值；此时我们只考虑自由变量上的单元，并且已知每个单元上 Φ_0的真值。如果 $TF(C_i)$ 是定义单元 C_i 的塔斯基公式，那么无量词公式 Ψ 可以取为

$$\bigvee_{i:\Phi_0(C_i)\equiv \text{true}} TF(C_i).$$

A. 变量排序

QE 的定义（1）在 CAD 的变量上施加了偏序 $\prec$：必须先投影 x_{a+1}，i，然后是 x_a，i，依此类推，最后投影 y_i。不过，每个变量块内部的顺序并未指定。然而，CAD 要求一个细化 $\prec$的全序 $\prec$。如何选择最佳细化这一问题历史悠久，至少可以追溯到[Brown（2003）]。有多种策略，可根据其所进行的投影量加以分类。

完全这里我们取所有 $n!$（如果有完全自由度）可能的排序，并将它们全部投影到 $\mathbb{R}^1$，然后选择“最佳”方案。这可以是总次数之和（sotd）[Dolzmann 等人（2004）],，也可以是不同实根数（ndrr） [Bradford 等人（2013a）]。部分这里我们取第一个变量的所有 n （如果有完全自由度）可能选择，执行一次投影到 $\mathbb{R}^{n-1}$，选择最佳方案（通过 sotd），然后重复此过程，因此执行 $O(n^2)$ 次投影：“贪心”方法，见 [Bradford 等人（2013a）]。

无这里仅关注输入。第一种方法在 [Brown（2003）], 中被称为“Brown”；RegularChains [Chen 和 Moreno Maza（2016）] 软件包中有其变体，而 [Tonks（2021）] 中还有一种适用于等式约束的方法。鉴于这一选择，人们采用机器学习来选择策略 [Huang 等人(2019)]。该软件允许用户直接指定 $\prec$（系统会检查 $\prec$ 是否细化 $\prec$），也可以指定用于执行细化的若干策略之一。相关细节见 [Tonks(2021)，§3。8], 实验结果见 [Tonks(2021)，§7.4。2],总结于 §V-C。

B. 柱形代数分解与方程

使用柱形代数分解进行量词消去时，我们只需要对布尔表达式 Φ中的多项式保持符号不变。McCallum [McCallum(1985)] 将符号不变替换为更强的阶不变条件，从而能够使用小得多的投影算子，但其缺点是：当某个多项式发生零化时，无法应用该方法。随后， [Mc- Callum 等人(2019)] 利用字典序最小不变性，并采用不同形式的投影和提升来避免零化问题，从而论证了 Lazard 在 [Lazard(1994)],中的思想。

事实上，我们并不需要处处满足某种形式的不变性：Collins [Collins(1998)]指出，如果 Φ 的形式为 $\widehat{\ }(P_1 = 0) \wedge \Phi$，那么我们只需要一个针对 P_1 的代数簇$\widehat{\ }$的柱形代数分解，使其对 Φ中的多项式符号不变。换言之，子句 $P_1 = 0$ 将分解整个空间的问题简化为分解该代数簇的问题。在这种情况下， P_1 称为等式约束（EC）。在[Bradford 等人(2013b)] 中进一步研究了这一思想，并指出只需要一个柱形代数分解，使 Φ 整体的真值保持不变。

第二作者的学位论文 [Tonks(2021)]中的软件基于 [McCallum 等人(2019)] 版本的 Lazard 方法，并采用了 [Nair(2021)]中的一些等式约束改进。

C. Gr¨obner 基

我们可能有多个等式约束，即我们的问题 Φ 具有如下形式

$$(P_1 = 0) \wedge \cdots \wedge (P_k = 0) \wedge \widehat{\Phi}. \tag{4}$$

在这种情况下，McCallum [McCallum(2001)],建议（假设所有多项式关于待消元变量的次数均非零，且都是本原的），将$P_1 = 0$ 视为 x_n中的等式约束，则$R_{1,\,2}$:=res$_{x_n}$(P_1, P_2) 视为 x_{n-1}中的等式约束； $R_{1,\,2,\,3}$:=res$_{x_{n-1}}$(res$_{x_n}$(P_1, P_2), res$_{x_n}$(P_1, P_3))位于 x_{n-2}中，依此类推。

我们只关注P_i的代数簇上的解，而 P_i 的具体形式并不重要。特别是，我们可以计算 P_i的Gr¨obner基 $\{Q_i\}$ ；如果采用字典序且该基是三角形的（正如我们对“处于一般位置”的情形通常所预期的那样，当然我们未必能满足这一条件），那么我们就可以在 x_n中使用 Q_1 ，在 x_{n-1}中使用 Q_2 ，依此类推：参见 [Wilson等人(2012)]。正如 [Bus´e和Mourrain(2009)]所指出的，$R_{1,\,2,\,3}$ 并不是由 $p_1 = p_2 = p_3 = 0$ 定义的真正的多元结果式（根据 $p_1 = p_2 = p_3 = 0$ ，其次数应为 $\mathcal{O}(d^3)$，由B´ezout定理可知），而是次数为 $\mathcal{O}(d^4)$。因此，一旦有了等式约束 ≥ 3 等式约束，我们预计使用Gr¨obner基将带来显著收益。然而，在软件包 QuantifierElimination随附的基准测试中，我们看到的改进却非常有限（见[Tonks(2021)，图7.6]）。这是由于所使用的基准测试中此类示例十分罕见。

D. 幕

当某些多项式变为零，即在 $\mathbb{R}^k = \langle x_1, \ldots, x_k \rangle$的某个子集上消失时，Collins算法的大多数改进都会遇到困难。Nair等人(2020)]引入了术语“幕”，用来指多项式变为零的代数簇：

定义 2 ([Nair(2021)，定义 43]). **</style}代数簇称为幕，如果对于任意(**$x, x_n) \in C$，都有$(x, y) \in C$对所有 $y \in \mathbb{R}$成立。

C 若是 $\mathbb{R}^n \to \mathbb{R}^{n-1}$ 的纤维之并，则称其为幕。

定义 3 ([Nair(2021)，定义 44])。假设 $f \in \mathbb{R}[x_1, \ldots, x_n]$ 且 $S \subseteq \mathbb{R}^{n-1}$。若 V_f（或 f）在 S 处有一条幕，是指对于所有 $(\alpha_1, \ldots, \alpha_{n-1}) \in S$， $y \in \mathbb{R}$ 都有$f(\alpha_1, \ldots, \alpha_{n-1}, y) = 0$。我们称 S 为这条幕的底。若底 S 是单点，则称这条幕为点幕。

定义 4 (CF [Nair(2021)，定义 45])。假设多项式 $f \in \mathbb{R}[x_1, \ldots, x_n]$ 可因式分解为 $f = gh$，其中 $g \in \mathbb{R}[x_1, \ldots, x_{n-1}]$ 且 $g(\alpha_1, \ldots, \alpha_{n-1}) = 0$。则称 f 的代数簇包含一条显式幕，其底为 g 的零点集。不（按集合意义）包含显式幕的幕称为隐式幕；包含显式幕但本身不是显式幕（因为 h 本身有幕）的幕称为混合幕。

存在幕时，McCallum 构造 [McCallum(1985)]可能无法保证完备性。McCallum 等人(2019) [证明，Lazard 提升不存在这个问题；但正如 Nair(2021) [Nair 等人(2020)], [指出的，尝试将等式约束并入 Lazard 提升时，如果等式约束含有幕，就会产生问题。

[Tonks(2021)] 的算法 30–34 实现了 [Nair(2021) 第 5、6 章中的单等式约束理论，1],当我们有一个等式约束 $f = 0$ ，且 f 在$x_{a+1,k_{a+1}}$中的次数为正时，生成该部分 $\mathbb{R}^n$（即 f的代数簇上 f =0 的部分）的符号不变 CAD。当然，当前实现无法将多个等式约束作为此类约束来处理；当然，也可以忽略它们是等式约束这一事实。

E. 部分 *CAD*

部分 CAD 是 Collins 和 Hong [Collins 和 Hong(1991)] 专为量词消去量身打造的 CAD 算法关键优化。具体而言，与构造每一个单元不同，只要构造出能够解决量词消去问题的满意单元（或单元集），就可以终止 CAD 的构造。例如，对于完全存在量化的问题，部分 CAD 可实现最佳情况：只需构造一个最高层级的单元，其样本点

与该单元相关联的样本点就是见证，将其代入Φ 后即可使其为真。QuantifierElimination 软件包包含一个部分 CAD 实现，其设计紧密遵循 [Collins 和 Hong(1991)] 中的思想。

Collins–Hong [Collins 和 Hong(1991)]通过创建子单元，并在可能的情况下将真值从这些子单元向根传播，描述了部分 CAD。在上述讨论的术语中，子单元是尚未求值的单元，而传播过程则会对其进行求值。

子单元是根据某个单元的下一个规范变量，通过堆栈构造建立的。单元 c 的子单元是空间中 c 的子集，位于 c的圆柱内；由于分解的缘故，就 CAD 中尚未求值单元的规范容器而言，它们在某种意义上“替代”了 c 。其中会出现实代数数和实代数函数；实代数数构成由 c的单变量提升多项式确定的静态局部单元边界，而实代数函数则用于形成单元的局部单元描述。这里还需要谨慎处理从任何提升约束中得到的边界。

真值传播的思想是在子单元中搜索有意义的真值，一旦找到，就尝试将其沿树向上传播。当一个单元具有有意义的真值时，可以将以该单元为根的子树从 CAD 中移除，即其中的单元无需再进行求值或其他处理。真值传播的返回值使我们能够推断出可以移除的最大 CAD 子树，而不是在向上遍历的过程中徒劳地尝试移除该子树的每一个子树。

提升失败虽不常见，但仍应加以处理和避免。在 QuantifierElimination 中，零化现象仅与等式约束的零化相关多项式有关，并且可用幕来刻画。若单元 c 在层级 $2 \leq i \leq n$ 的枢轴多项式上存在幕，那么字典序最小半评价 [Brown 和 McCallum（2020）] 并不足够，因为相关集合中的多项式之间缺少交叉结式。因此，无法找到用于推导更精细几何结构的多项式，以实现字典序最小不变性，甚至符号不变性。这些幕妨碍我们构造字典序最小且符号不变的柱形代数分解，因此，至少识别出这些幕，并在可能时避免或处理它们，都很有意义；例如，对于通过部分 CAD 进行量词消去而言，符号不变性意味着顶层公式上的真值不变性。QuantifierElimination 中的柱形代数分解会检查任何经由堆栈构造处理的子单元 c 是否存在幕，方法是查看导致零化的枢轴多项式的因子。

点幕不会造成问题。下面给出了一个利用等式约束处理 Lazard 风格柱形代数分解的定理。QuantifierElimination 软件包依靠该定理来识别并解决此类幕问题。

定理 1（根据 [Nair(2021)，定理 18]）。设 $f \in \mathbb{R}[x_1, \ldots, x_n]$，并设 $\alpha \in \mathbb{R}^{n-1}$。如果 f是一个等式约束，并且在 α处有一个点幕，那么 $P_L^{\{f\}}$，也就是说，对问题中所有 g，$\{f\}$ 的 ***Lazard*** 投影和 $\{\mathrm{res}_{x_n}(f, g)\}$ 是充分的以获得符号不变 *CAD*。

注 1。定理 *1* 意味着，如果我们构造一个字典序最小柱形代数分解，使得唯一的 ***level*** $n-1$ 幕都是点幕，那么实际上就能实现符号不变性。这并不意味着 ***Lazard*** 不变性，但通常 ***Lazard*** 不变性仅用于推出符号不变性，例如通过实现真值不变性来实现量词消去。

QuantifierElimination 软件包还提供了使用多个等式约束的选项，但无法保证答案正确。不过，幕问题很少见 [Tonks(2021)], 参见第 VI-A 节。

IV. 多算法方法

本节讨论 QuantifierElimination 软件包的一项主要特性。相关思想和方法的初步探索见 [Tonks(2019)]。读者可以先观看一段概述多项式算法方法的视频，该视频可公开获取[Tonks(2020b)]，最初是为 ICMS 2020 录制的。

A. 处理 *VTS IQER*

多算法方法的目的是尽可能使用 VTS。一旦发现 VTS 无法独自完成量词消去（因为它收到至少一个不适用的 IQER，其中包含四次或更高次数的多项式，因而无法使用 VTS），多算法便会将该 IQER 转交给柱形代数分解。此外，在这种多算法方法中，我们希望尽可能重新利用柱形代数分解计算。

由于我们只能将一种类型的量词通过同一个量词块内形成的析取或合取进行分配，因此多算法只用于最后一个量词块。这已经足以解决一大类问题。例如，在理论可满足性（SMT）的量词消去以及许多类似示例中，问题的量词是齐次的，所以我们始终处于最后一个量词块。可以将最内层量词块的（剩余）部分通过 VTS 的规范布尔运算符分配到不适用的 IQER 上。若不加区分地对这些 IQER 使用柱形代数分解，就会生成柱形公式，供剩余的量词块进行量化。然而，柱形代数分解计算会将这些剩余量词块中的变量视为自由变量，并围绕它们构造几何结构；而柱形代数分解复杂度高度取决于变量的数量，因此这可能导致不必要且繁重的计算。因此，在这种情况下使用多算法会导致重复使用后续量词块中的变量。更糟的是，QuantifierElimination 中柱形代数分解的无量词输出通常是扩展塔斯基公式，不适合作为 QuantifierElimination 中任何量词消去函数的输入，因此在当前实现中，我们甚至无法考虑对其进行“重新量化”，并使用后续量词块。

因此，我们要求终止时 VTS 的状态为

$$Qx_{n-m+1} \ldots Qx_{n-t}\ B\left(B_{j=1}^{k} I_j^{'}\ B_{j=1}^{s} I_j\right) \tag{5}$$

对于 t 任意 I_1、.....、 I_s，不合格 IQER 的最低层级，以及 $I_1^{'}$、.....、 $I_k^{'}$ $k \geq 0$，要使用 VTS 传播的叶 IQER，B 是与量词符号 $Q \in \{\exists, \forall\}$ 相对应的规范布尔运算符。

我们通常按以下方式处理 IQER:

1) 根据某种度量，从不符合条件的 IQER（I_1、.....、 I_s）中选择一个 IQER I 。QuantifierElimination 默认使用基于所选 IQER 深度的度量，该深度与其量化变量的数量成反比。通过部分 CAD 执行量词消去 $Qx_{n-m+1} \dots Qx_{n-I \mapsto \text{level}}\ I$。我们保留此 CAD 的数据，$C$。

2) 如果这得出了有意义的真值，我们就完成了。否则，根据所选度量选择下一个要解决的 IQER I 。我们预计此树中的 IQER 具有相似的布尔结构和多项式。通过将 I 中的多项式作为一个集合进行检查，我们可以衡量此 IQER 的多项式与 CAD 中上一个使用的 IQER 公式中的多项式之间的重合比例。上一个 IQER 的公式通过重新利用，存在于该 CAD 的根单元中。如果共同多项式的比例达到某个阈值，我们就增量式地重用该 CAD C 来解决 I。否则，我们可以舍弃 C 并创建一个新的 CAD 来解决I。此外， I 不得包含对C 而言全新的自由变量，以便 C 能够处理 I；否则，我们必须创建一个新的 CAD。不过，新的量化变量是允许的。3) 我们进一步迭代这一过程，并在之后生成完整的量词消去输出时，使用每个不符合条件的 IQER 的等价 CAD 表述。再次说明，如果在这些计算过程中得到有意义的真值，我们就完成了。4) 多项式算法终止后，此量词块的量词消去输出可描述为 $B\left(B_{j=1}^{k} I_j^{'}\ B_{j=1}^{s} I_j\right)$ ，其中每个 I_j 都使用其 CAD 输出 C_j 作为其量词自由等价式，而每个 $I_j^{'}$ 都使用其简化后的公式。

请记住，VTS 假设带量词的输入公式是塔斯基公式，即由整系数多项式约束构成；多项式算法仅适用于塔斯基公式。对于无理数的情形，可以将根 IQER 规范化为不适用，但只要根 IQER 不适用，多项式算法就显然没有用处——量词消去会退化为与部分 CAD 相同的方法。

使用多项式算法的最佳情形，是对恰好一个不适格的 IQER 使用柱形代数分解，以获得有意义的真值。在完全（齐次）量化的公式中，真和假是 IQER 的量词自由等价式仅有的候选项。因此，多项式算法能够很好地处理完全量化公式。将量词消去用于 SMT 求解器时，公式实际上被视为完全存在量化，因此多项式算法通过寻找真（即可满足性）来很好地适配 SMT，而无需使用所有 IQER。

B. 遍历 *VTS* 树的增量式 *CAD*

在多项式算法的语境下，对 VTS 树进行深度优先遍历，意味着所选 IQER 的深度会逐渐减小（不一定是单调递减），以便重新利用已有的柱形代数分解。深度严格小于上一个 IQER 的 IQER，所持有的量化变量严格更多。这是 QuantifierElimination 的默认选择，另一种选择是广度优先遍历。广度优先遍历会尽早固定所有可能纳入柱形代数分解的变量的排序，但代价是柱形代数分解可能尽可能“庞大”。深度优先遍历旨在创建尽可能小的柱形代数分解，以期为总体量词 Q找到有意义的真值；它对后续变量的排序保留了一定的自由度，但并非完全自由。

对于柱形代数分解，我们首先按照变量排序中的最后一个变量进行投影，而提升的方向与投影方向相反，因此根单元的子单元对应于柱形代数分解变量排序中的第一个变量。事实上，VTS 的作用方向与柱形代数分解的投影方向相反，这一点很方便，也就是说，各算法树的构造方向实际上是一致的。如果要在柱形代数分解中使用的 IQER，在一个完全齐次量化问题 $Qx_{n-m+1} \dots Qx_n\ \Phi$($x_1$、.....、 x_n) 中的层级为 j ($0 < j < m$)，那么它涉及变量 x_1、.....、 x_{n-j}。为了适配层级为 $k > j$ 的 IQER，我们现在必须扩展柱形代数分解，使其包含变量 x_{n-j+1}、.....、 x_k。对包含这些变量的多项式进行投影是规范的，由此生成的 x_1、.....、 x_j 中的多项式可以通过缓存，逐步输入投影的其余部分。由于这些新变量可以出现在排序的末尾，提升后的 CAD 树只需扩展，无需“重新提升”。现有几何结构均仍然有效；只需根据待求值的 IQER 中传入的塔斯基公式，重新利用这些单元即可。

需要注意的是，与使用多项式算法不同，另一种做法是在终止时将 VTS 的状态 (5) 作为公式，以“整体/标准”的方式传递给柱形代数分解（如 [Strzebo´nski(2022)]）。这样做会舍弃多项式算法。采用这种非增量方法的一个潜在优点是，柱形代数分解在初始时就能获得全部信息——初始的柱形代数分解调用在变量排序方面不受限制，因此不会出现无法很好地适应后续增量调用的情况。此外，变量策略完全不必与 VTS 中的变量策略相关。

第五节所示的图 2 中的生存图对应用于量词消去问题的这些不同技术进行了比较，结果发现它们相当相似。

C. 多项式共享准则

多项式共享准则是指用于决定是否重复使用某个柱形代数分解来求解传入 IQER 的准则。判断该柱形代数分解的几何结构是否适合传入 IQER，最可靠的方法是检查投影基；然而，结构的所有层级中分布着数量本质上呈指数增长的多项式。由于假设这些多项式在不同 IQER 之间

表 I：钢琴搬运工问题的统计数据。

方法	# 投影多项式	# 等式约束	# 单元	# 叶单元
深度/广度	10	1	154	143
整体	13	0	314	283

相似，我们转而建议对传入 IQER 中的多项式与上一次使用的 IQER 中的多项式求最大公因式。由于这是我们首次检查 IQER 的多项式之处，我们还会确保 IQER 中不包含任何自由变量，以免无法重复使用现有的柱形代数分解。

V. 实验结果

A. 多算法方法的比较

我们将比较 [Yang 和 Zeng（2006）] 中提出的钢琴搬运工问题的多算法方法结果。该问题带有存在量词：

$$\exists a\,\exists b\,\exists c\,\exists d\; a^2+b^2=r^2 \wedge 0\le a \wedge b<0 \wedge 1\le c \wedge d<-1 \wedge c-(1+b)(c-a)=0 \wedge d-(1-a)(d-b)=0.$$

VTS 可以消去 4 个量词中的 3 个，但所有由此产生的第 3 层 IQER 中都会出现同一个多项式 $-a^6+a^4r^2+2\,a^5-2\,a^3r^2-2\,a^4+a^2r^2$，它可分解为不可约因子 $a^2(a^4-a^2r^2-2\,a^3+2\,ar^2+2\,a^2-r^2)$，其中一个因子的次数为 4，因此 VTS 无法遍历该因子。

在这里，考虑到 VTS 树生成的所有 IQER 都处于第 1 层（即待消去的量词数量），并且缺少真/假 量词自由等价式意味着必须遍历每个此类 IQER，因此按深度优先和按广度优先遍历 VTS 树的适用情形相同。

传递给柱形代数分解的第一个 IQER 本身就是一个关于 a 的存在量化量词消去问题，其中含有 6 个多项式和一个自由变量 r。最终得到的柱形代数分解在投影中使用了 10 个多项式，并产生了 135 个叶单元。下一个 IQER 需要新增 4 个投影多项式来扩展该柱形代数分解，但此后不再需要新增多项式，只需重新利用不同的布尔结构。在第一次重新利用时，一个多项式被识别为等式约束——这纯粹是一种语义上的识别——并用于简化 IQER。

相比之下，将整个 VTS 树合并为一个供柱形代数分解遍历的量词消去问题，会得到一个存在量化析取式。方程与其他非方程关系一起嵌套在合取式中，因此柱形代数分解无法推导出任何等式约束。所得柱形代数分解在投影中总共包含 13 个多项式，其中 7 个对应 r，6 个对应 a。最终有 283 个叶单元，总共生成了 314 个单元（包括父单元）。

我们在表 I 中比较了多算法量词消去所使用的柱面代数分解统计数据。

有关其 IQER 处理中的多算法方法的更多案例研究，可见 [Tonks(2021)]。

B. 基准测试

[Tonks(2021)],中有许多实验结果，以生存图的形式呈现 [Brain et al.(2017)]，涵盖相当大型的[1] 基准测试集，并存档于 [Tonks(2020a)]。请注意，这种图表样式以“200 个示例”为例，展示每种实现处理其表现最佳的 200 个示例所花费的时间（对该实现而言）；这些示例可能与另一种实现用于绘图的 200 个示例并不相同。

所有基准测试均在一台运行 Maple 2020.1 和 Ubuntu 18.04.3 的计算机上进行，该计算机配备 16GB 随机存取存储器，并搭载运行频率为 2.00 GHz 的 Intel i5-4590T 处理器。

C. 柱面代数分解实现

当请求执行一个对变量顺序没有约束的柱面代数分解时，有许多可用于选择排序的启发式方法（见 §III-A）。基准测试（见图 1）表明，在此处使用贪心策略似乎表现最佳。与“完整 sotd”相比，使用贪心策略的代价更低，同时仍会针对本例中每个 x_1 的选择生成 $\mathcal{O}(n)$ 个投影集。鉴于贪心策略会关注等式约束，而 QuantifierElimination 对这类约束也很敏感，并且该实现本来就打算高度重视等式约束，因此这一结果可能并不令人意外。不过，我们注意到，QuantifierElimination 软件包认为，使用 Gr¨obner 基预处理（该预处理已经选择了变量排序）与使用贪心柱面代数分解变量策略不兼容。因此，对于涉及等式约束的情况，运行时间统计从不包括生成 Gr¨obner 基的时间，这可能会使贪心策略的投影运行时间看起来更有利。

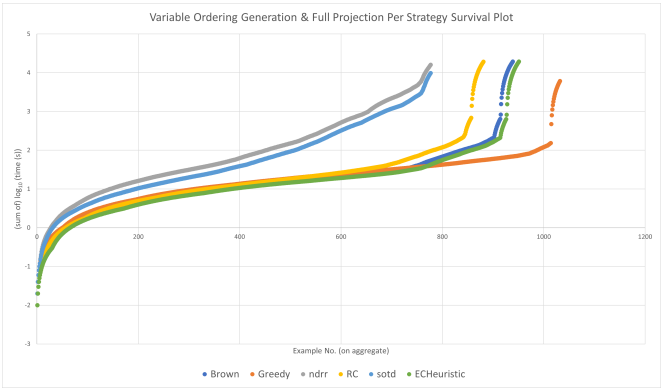


图 1： [Tonks(2021)，图 7。1]

对于大多数示例，ECHeuristic 的效果与使用 Brown 启发式基本相当。总体而言，从曲线末端可以看出，ECHeuristic 略有提升，这很可能是因为它对等式约束给予了额外关注。RegularChains 的 [Chen 和 Moreno Maza（2016）] 启发式所对应曲线的形状与 Brown 或 ECHeuristic 的相似。考察这些策略后可以得出结论：ndrr 和 sotd 不负 $\mathcal{O}(n!)$ 所望。

[1] 452 例如图 2。

D. CAD 基准测试

QuantifierElimination 包含独立实现的柱形代数分解，既有完全柱形代数分解，也有部分 CAD。尽管该软件包中的完全柱形代数分解函数 CylindricalAlgebraicDecompose 接受（带量词的）实数公式，但由于其目标是生成每一个最大叶单元，因此不会计算其生成的叶单元上的真值。它本质上是将公式分解为规范的不等式集和等式约束集，并执行完全柱形代数分解。它还可以接受不含关系运算符的纯多项式集，在这种情况下，真值的计算没有明确定义。部分柱形代数分解函数 PartialCylindricalAlgebraicDecompose 接受无量词公式，在这种情况下会生成一个 CAD 数据对象，可对其进行检查，以统计单元的相关信息，例如真值为真的单元数量。

CAD 基准测试示例的类型相当不统一，有些是多项式列表或关系，有些则只是无量词公式。CAD 计算（见[Tonks(2021)，图 7。4]，）表明，这种 Lazard 风格的实现明显优于之前的投影–提升实现（基本上是 McCallum 方法），但不及正则链实现。 [Tonks(2021)，图 7。5] 表明，正则链通常生成的真单元比 Lazard 风格少（通常少 50 倍），这或许解释了大部分差异。使用 Gröbner 基会减少单元数量，但在当前情况下，这种减少在许多情形中可能并不显著，见 [Tonks(2021)，图 7。6]。

E. 量词消去基准测试

量词消去的一个显而易见的竞争者是 SyNRAC [Yanami 和 Anai（2007）]，，它同样在 Maple 中实现，同时支持 VTS 和 CAD。当 SyNRAC 通过 VTS 进行量词消去得到的公式次数完全过高时，SyNRAC 会以一种非多项式算法的方式切换到 CAD。

在不同等式约束使用方式（单个与多个）下，部分柱面代数分解函数的性能相近。对于不含等式约束的示例，两种方式的表现完全相同，因此曲线大体相似也在意料之中；但事实上，这两种方法完成的示例数量完全相同。需要考虑的一点是，在使用多个等式约束时，如果确实识别出多个等式约束，CAD 提升就必须更频繁地检查幕，尤其是针对更大的（较低层级的）投影多项式，这会增加开销。此外，部分 CAD 通常也必须设法避开更多低层级的幕。

与此同时，通过 QuantifierEliminate 函数实现的多项式算法优于纯部分 CAD 方法。测试的示例集中，有相当一部分是来自经济学数据库的经济学量词消去问题。这些示例中的每个变量大多都是线性的，而且变量数量通常很多。显然，在这类问题上，VTS 的表现优于柱形代数分解（CAD），而 CAD 尤其容易受到变量数量的影响。考虑到纯柱形代数分解方法，与 RegularChains 相比，部分柱面代数分解函数在处理线性示例时的表现出人意料地具有竞争力；更广泛地说 QuantifierElimination 函数在主要为线性或二次的示例上似乎颇具竞争力，这可能是因为使用了 VTS，也可能是因为柱形代数分解（CAD）在变量众多（但可能存在许多等式约束）的情况下，仍能解决这类示例，令人意外。近期使用 Maple 2022 时发现，在 [Uncu 等人（2023）] 中所考虑的 SMT 场景下，QuantifierElimination 在 14% 的示例中比 RegularChains 更快。我们推测，这得益于 VTS 的使用，以及第二作者在此期间为 Maple 原生实现的更出色的实根隔离函数。

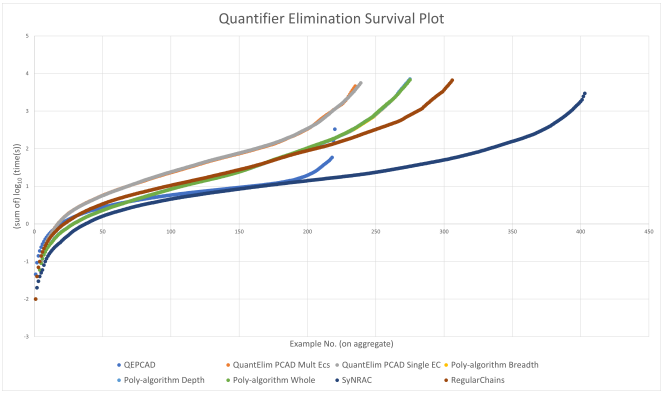


图 2： [Tonks（2021），图 7。8]

图 2 显示，SyNRAC [Yanami 和 Anai（2007）] 显然是一种极具竞争力的实现。Maple 原生的 RegularChains [Chen 和 Moreno Maza（2016）]表现也相差不远；量词消去是该软件包较新的功能，基于他们的柱形代数分解方法。QEPCAD B [Brown（2003）] 是唯一在 Maple 之外测试的软件，其表现令人赞叹。需要注意的是，运行时间可能无法与专为 Maple 生成的运行时间直接比较，但两者采用的超时时间始终相同，因此 QEPCAD B 的“存活率”具有直接可比性。当然，QEPCAD B 使用的底层操作与 Maple 软件包所用的操作完全不同。QEPCAD B 的曲线整体上紧跟 SyNRAC 的曲线，在曲线后段突然急剧上升，这表明它在较小示例上的表现出色，但处理大得多的示例时则有些吃力。

VI. 结论

A. 贡献

1) 与柱形代数分解相比，VTS 在经济学等线性问题上总体表现更佳；不过，Maple 基于柱形代数分解（RegularChains）的标准量词消去在这类问题上可能出奇地有效，这很可能得益于等式约束。2) 柱面代数分解实现新增了“提升约束”功能，适用于约束能够推出空间中的超矩形的问题。这种情况可能出现在生化应用中 [Bradford 等人（2019）]，，其中浓度需要为正。

3) 幂很少见：452 个示例只产生了两个幂问题 [Tonks(2021)，第 280]。4) 多项式算法可以生成见证点：参见 [Tonks(2021)，§2.5]。

B. 未来工作

1) 关于多个等式约束的研究（[Nair(2021)] ）仍需进一步发展。Davenport 等人（2023）（ [Davenport et al.(2023)] ）已将这一理论扩展至[Brown and McCallum(2020)] 的框架，但尚未实现。2) 需要探索 Gr¨obner 基（§III-C）与变量排序（§III-A）之间的相互作用。3) [Huang 等人(2016)], 探讨了何时在 CAD 中使用 Gr¨obner 基，但研究背景是正则链柱面代数分解，而非投影–提升版本。因此，这个问题应重新审视。

参考文献

[Bradford 等人（2019）] R.J. Bradford、J.H. Davenport、M. England、H. Errami、V. Gerdt、D. Grigoriev、C. Hoyt、M. Ko˘sta、O. Radulescu、T. Sturm 和 A. Weber。2019。识别某些生物网络中多稳态的参数化出现。《符号计算杂志》98（2019），84–119。[Bradford 等人] R.J. Bradford、J.H. Davenport、M. England、S. McCallum 和 D.J. Wilson。2013b。布尔组合的柱状代数分解。载于 ISSAC 2013 论文集。125–132。[Bradford 等人] R.J. Bradford、J.H. Davenport、M. England 和 D.J. Wilson。2013a。优化柱状代数分解的问题表述。载于 CICM 2013 论文集，J. Carette 等人（编）。19–34。[Brain 等人（2017）] M.N. Brain、J.H. Davenport 和 A. Griggio。2017。基准测试求解器：SAT 风格。SC^2 *2017* 可满足性检查与符号计算 CEUR 研讨会 1974，RP3（2017），1–15。http://ceur-ws.org/Vol-1974/RP3.pdf [Brown（2003）] C.W. Brown。2003。QEPCAD B：一种使用 CAD 计算半代数集的程序。ACM SIGSAM Bulletin 37，4（2003），97–108。[Brown 和 McCallum(2020)] C.W. Brown 和 S. McCallum。2020。柱状代数分解中对 Lazard 方法的改进。载于《科学计算中的计算机代数 CASC 2020》（Springer LNCS，第 12291 卷），F. Boulier、M. England、T.M. Sadykov 和 E.V. Vorozhtsov（编）。129–149。https://doi.org/10.1007/978-3-030-60026-6 8[Bus´e 和 Mourrain（2009）] L. Bus´e 和 B. Mourrain。2009。某些迭代结式与判别式的显式因子。《数学计算》78（2009），345–386。https://doi.org/doi:10.1090/S0025-5718-08-02111-X [Chen 和 Moreno Maza（2016）] C. Chen 和 M. Moreno Maza。2016。基于正则链的柱状代数分解进行量词消去。《符号计算杂志》75（2016），74–93。[Collins（1975）] G.E. Collins。1975。通过柱状代数分解消去实闭域中的量词。载于《自动机理论与形式语言第二届 GI 会议论文集》（Springer LNCS，第 *33* 134 183 10 1007 3 540 07407 4 17 卷）。– 。[1998] 1998 https://doi.org/ 。 / - - - Collins（ ） G.E. Collins。 。通过柱状代数分解进行量词消去——二十年的进展。载于《量词消去与柱面代数分解》，B.F. Caviness 和 J.R. Johnson（编）。Springer Verlag，维也纳，8–23。https://doi.org/ 10.1007/978-3-7091-9459-1 2[Collins 和 Hong(1991)] G.E. Collins 和 H. Hong。1991。用于量词消去的部分柱状代数分解。《符号计算杂志》12（1991），299–328。[Davenport 等人（2023）] J.H. Davenport、A.S. Nair、G.K. Sankaran 和 A.K. Uncu。2023。Lazard 风格的 CAD 与等式约束。载于 ISSAC 2023 论文集，G. Jeronimo（编）。218–226。[Davenport 等人（2022）] James H. Davenport、Zak Tonks 和 Ali Uncu。2022。量词消去方法的实用评估。载于 SC-Square 2021：可满足性检查与符号计算（CEUR 研讨会论文集，第 3273 卷），Curtis Bright 和 James H. Davenport（编）。CEUR，41–49。https://ceur-ws.org/Vol-3273/paper4.pdf [Dolzmann 等人（2004）] A. Dolzmann、A. Seidl 和 Th. Sturm。2004。CAD 的高效投影序。载于 ISSAC 2004 论文集，J. Gutierrez（编）。111–118。[Huang 等人（2016）] Z. Huang、M. England、J.H. Davenport 和 L.C. Paulson。2016。利用机器学习决定何时使用格罗布纳基对柱面代数分解进行预处理。载于 SYNASC 2016 论文集。45–52。[Huang 等人（2019）] Z. Huang、M. England、D. Wilson、J.H. Davenport 和 L.C. Paulson。2019。利用机器学习改进柱面代数分解。《计算机科学中的数学》11（2019），461–488。[Ko˘sta（2016）] M. Ko˘sta。2016。通过虚拟代换进行实数域量词消去的新概念。博士学位论文。Universit¨at des Saarlandes。[Lazard（1994）] D. Lazard。1994。用于柱面代数分解的改进投影算子。载于《代数几何及其应用：Shreeram S. Abhyankar 六十岁生日会议论文集》，C.L. Bajaj（编）。467–476。[McCallum（1985）] S. McCallum。1985。用于柱面代数分解的改进投影运算。技术报告 578。威斯康星大学麦迪逊分校计算机科学系。https://minds.wisconsin.edu/bitstream/handle/ 1793 58594 578 =1 / /TR . [2001] 2001 pdf?sequence McCallum（ ） S. McCallum。 。在基于 CAD 的量词消去中传播等式约束。载于 ISSAC 2001 论文集，B. Mourrain（编）。223–230。https://doi.org/10.1145/384101. 384132[McCallum 等人(2019)] S. McCallum、A. Parusi´nski 和 L. Paunescu。2019。Lazard 方法用于 CAD 构造的有效性证明。《符号计算杂志》92（2019），52–69。[Nair(2021)] A.S. Nair。2021。柱面代数分解中的幂。博士学位论文。巴斯大学。
h
t
t
p
s://
r
e
s
e
a
r
c
h
p
ortal.bath.ac.uk/en/studentTheses/curtains-in-cylindrical-algebraic-decomposition [Nair 等人(2020)]A.S. Nair、J.H. Davenport 和 G.K. Sankaran。2020。CAD 中的幂：为何它们会造成问题，我们又该如何解决？载于《数学软件——ICMS 2020》，A.M. Bigatti、J. Carette、J.H. Davenport、M. Joswig 和 T. de Wolff（编）。Springer LNCS，第 12097 卷。Springer，17–26。https://doi.org/10.1007/978-3-030-52200-1 2[Strzebo´nski（2022）] A. Strzebo´nski。2022。实多项式系统、Wolfram 语言与系统。https://reference.wolfram.com/language/tutorial/RealPolynomialSystems.html。[Tonks（2019）] Z. Tonks。2019。量词消去系统中的进化式虚拟项代换。SCSC 2019 论文集，CEUR 2460（2019）。[Tonks(2020a)] Z. Tonks。2020a。支持学位论文《实数域量词消去中的多算法技术》的数据存储库。https: //doi.org/ 10 5281 4382083 . /zenodo。Tonks(b) Z. Tonks。b。使用 Maple 进行量词消去的 VTS 与 Lazard 投影 CAD：国际数学软件大会（ICMS）2020 演讲。https://av.tib.eu/media/48017。[Tonks（2021）] Z. Tonks。2021。《实数域量词消去中的多算法技术》。博士学位论文。巴斯大学。https://researchportal.bath.ac.uk/en/studentTheses/poly-algorithmic-techniques-in-real-quantifier-elimination [Uncu 等人（2023）] Ali Kemal Uncu、James H. Davenport 和 Matthew England。2023。不等式归纳证明的 SMT 求解。载于第七届 SC-Square 研讨会论文集，Ali Kemal Uncu 和 Haniel Barbosa（编），第 3458 卷。10–24。[Weispfenning(1988)] V. Weispfenning。1988。域上的线性问题的复杂性。《符号计算杂志》5（1988），3–27。[Weispfenning(1997)] V. Weispfenning。1997。实代数量词消去——二次情形及更进一步。《AAECC》8（1997），85–101。[Wilson 等人（2012）] D.J. Wilson、R.J. Bradford 和 J.H. Davenport。2012。通过格罗布纳基加速柱面代数分解。载于 CICM 2012 论文集，“J. Jeuring 等人”（编）。280–294。[Yanami 和 Anai（2007）] H. Yanami 和 H. Anai。2007。SyNRAC：用于求解实代数约束的 Maple 工具箱。ACM《计算机代数通讯》第 41 卷第 3 期（2007），112–113。[Yang 和 Zeng（2006）] L. Yang 和 Z. Zeng。2006。具有四个参数的钢琴搬运工问题的符号解。载于《几何中的自动演绎》（Springer LNCS，第 3763 卷），H. Hong 和 D. Wang（编）。59–69。