arXiv:2601.04634v1 [cs.LO] 8 Jan 2026

有限数学：使数学语义与有限计算相协调

LIAN WEN

信息与通信技术学院，格里菲斯大学，澳大利亚
e-mail address：l.wen@griffith.edu.au

Abstract. 编程语言和计算语义中使用的经典数学模型依赖于诸如无限精度实数、无界集合和不受限制的计算等理想化抽象。相比之下，具体计算本质上是有限的，需要在有界精度、有界内存和明确的资源约束下运行。这种差异使得对有限执行条件下的数值行为、代数性质和终止性进行语义推理变得复杂。

本文介绍有限数学（LM），这是一种使数学推理与有限计算相协调的有界语义框架。有限数学将对数值幅度、数值精度和结构复杂度的约束明确化，并将其作为基础要素。由单一界限 M 参数化的有限数值域配备了确定性的值映射算子，用于强制实施量化和明确的边界行为。函数和算子保留其经典数学解释，仅在语义边界处映射到有界域中，从而将含义与有界求值分离开来。

在可表示边界内，有限数学与经典算术一致；超出界限时，偏差是明确、确定且可分析的。此外，通过限制集合基数，有限数学可防止隐含的无限行为经由结构性构造重新引入。因此，在有限数学下实现的计算会产生有限状态语义模型，为在有限计算场景中推理算术、结构与执行提供了有原则的基础。

1. Introduction

经典数学模型构成了程序语言语义以及关于计算的形式化推理的基础 [13]。在这些模型中，程序和表达式通常通过无限精度实数、无界集合和不受限制的计算等理想化抽象来解释。虽然这些抽象有助于建立优雅的理论，但当它们面对具体计算固有的有限性时，便会造成持续存在的语义鸿沟；具体计算是在有界精度、有界内存和明确资源约束下运行的。

这种差距以一些广为人知且具有重要结构意义的方式显现出来。有限精度算术会出现舍入异常、相等性失败和非结合性等行为，使得对数值表达式进行语义推理变得复杂 [4, 5]。更普遍地说，经典语义模型通常假设计算资源不受限制，尽管具体执行必然局限于有限表示。甚至一些基本问题，例如

© L.Wen 知识共享

终止性，在经典计算模型中也是不可判定的 [15, 14], ，尽管有界机器会导出有限状态空间。

这些差异并非只是实现造成的伪影，而是反映了理想化数学语义与有限计算之间更深层的不匹配。将有限性视为隐含或次要问题，会迫使语义推理依赖有关精度、幅值和结构的外部假设，从而掩盖经典性质何时以及为何成立，又何时失效。

本文提出有限数学（*LM*），这是一种旨在使数学推理与有限计算相契合的语义框架。有限数学不再假设精度和结构不受限制，而是将对数值幅度、数值精度和结构复杂度的约束明确设定为基础。单一参数 M 决定最大可表示幅度、最小正精度单位，以及可表示数值的总数。算术定义在一个有限且均匀离散的域上，并通过确定性值映射算子来实现，该算子负责执行量化并明确处理边界行为。

除数值界限外，有限数学还限制了该框架所容纳的集合大小。通过将集合基数限制为随数值域的表示容量相应变化，LM 避免组合构造将隐式的无限性重新引入模型。因此，所有集合都是有限、可枚举且可在内部表示的，从而确保量化和结构推理始终与有界计算相容。这一视角在哲学上与有限模型论相契合，同时侧重于数值语义，而非逻辑表达能力 [10]。

有限数学的一项核心设计原则，是将语义含义与有界求值分离。函数和算子保留其经典数学解释，而单一的语义边界则将结果映射到有限数学域。这种方法可避免过早量化，限制复合表达式中的误差累积，并在中间结果仍处于可表示边界内时保留经典代数行为。超出界限时，产生的偏差是明确、确定且可从语义角度分析的。

在有界内存条件下实现有限数学时，计算在有限状态空间内进行。因此，每次执行要么停止，要么最终重新访问先前出现过的状态，并进入循环演化。这一观察并未改变经典不可判定性结果，因为这些结果依赖无界计算模型；但它明确揭示了有界执行所带来的有限状态语义。

有限数学并非旨在取代经典数学或实分析。相反，它为有限精度算术、有界结构和有限计算提供了基础性的语义框架。通过将资源界限明确呈现，而非隐而不宣，有限数学阐明了有限性在语义上的后果，并为在有限数学环境中推理计算提供了原则性基础。

贡献。 本文的主要贡献如下：

- 一个具有明确幅值和精度约束的有界数值域；
- 一种确定性值映射算子，使算术语义与有限计算保持一致；
- 一种有原则的集合基数限制，确保可枚举性和内部可表示性；

- 一种语义算子映射框架，将数学含义与有界求值区分开来；

- 在可表示边界内外对代数性质的分析；
- 由有界内存计算诱导的有限状态语义解释。

2. Design Principles of Limited Math

有限数学遵循一组简明明确的设计原则，旨在协调数学严谨性与实际可计算性。有限数学并不将有限性视为实现层面的产物，而是将资源界限直接纳入其形式结构。本节概述该框架的核心原则，并说明这些原则在构建理论过程中所发挥的作用。

2.1. **有界数值幅度**。 有限数学的首要原则是，所有数值都具有有界幅值。参数 $M > 0$ 决定了可表示绝对值的最大值。任何实数幅值超过该界限的量，都会通过饱和规则确定性地映射到边界值。

幅值设限体现了有限计算的现实：溢出不可避免，必须明确处理。在经典数学中，幅值没有界限，数值可以不受限制地增长。相比之下，在有限数学中，幅值界限是一等概念，并清晰体现在语义中。因此，溢出行为是明确且可分析的，而非隐含或未定义的。

2.2. **有界数值精度**。 第二项原则是有界精度。有限数学在均匀的离散网格上表示数值，其最小正单位为 $1/M$。所有数值都是该单位的整数倍，因此每个可表示的数值都有有限的描述，并且可以在没有舍入歧义的情况下进行运算。

这一选择使有限数学采用定点算术，而非浮点运算。与浮点系统不同，有限数学避免了非均匀间距、隐式舍入模式和异常值。精度是显式、均匀且由 M 参数化的，用户可以据此为特定问题领域选择合适的分辨率。

精度设限并不被视为一种限制，而是一种有意的建模选择。它确保算术运算具有确定性，且数值误差有界且可预测。

2.3. **有界结构复杂度**。第三项原则涉及结构复杂度。除了限制单个数值的大小外，有限数学还限制理论所允许的集合大小。具体而言，所有集合的基数都必须受 M 的某个函数所界定，该函数的选取应与数值域的表示容量相匹配。

这一限制的动机有两方面。首先，数值域本身只包含有限个不同的值。若允许基数超出这一容量的集合存在，就会引入无法在内部索引、枚举或区分的对象。其次，经典数学中的许多无穷现象并非源自数本身，而是源自不受限制的集合构造。通过限制集合大小，有限数学可防止隐含无穷通过组合方式重新进入理论。

这一限制确保所有集合都是有限的、可枚举的，并且能够在内部表示。因此，在有限数学中，对集合进行量化具有实际操作意义，且与有限计算相兼容。

2.4. **语义与执行的分离**。有限数学的一项关键设计原则是将语义定义与执行分离。数学函数和算子在经典实值数学层面进行解释，其含义已有明确界定。随后，通过确定性值映射算子将其结果映射到有界的 LM 域中。

这种分离避免了过早量化，并最大限度地减少了复合操作中的误差累积。它还明确了导数和极限等分析学概念的地位：这些概念并不内化于 LM 之中，而是可以作为语义构造加以处理，其结果随后在有界域中表示。

通过区分含义与表示，LM 在维持严格执行约束的同时，保留了概念上的清晰性。

2.5. **有界计算与内存（实现原则）**。当有限数学在计算机上实现时，还需遵守一项额外的实际约束：内存是有限的。在固定的内存上限下，任何 LM 程序的操作语义都会导出一个有限状态空间。因此，每次执行要么终止，要么最终重复先前访问过的状态。

这一观察并不能解决经典不可判定性结果，因为这些结果依赖于无界的计算模型。相反，它反映了实际机器的运行现实，并为在有界环境中精确推理程序行为奠定了基础。在这种情况下，终止性原则上是可判定的，因为可以在有限状态空间内穷举探索执行过程。

这一有界内存原则仅适用于实现，并不会对抽象数学框架施加超出实际适用性所必需的限制。

2.6. **总结**。 这些原则共同将有限数学定义为一种有界、明确且内部一致的框架。通过限制幅值、精度、结构复杂度以及（在相关情况下）计算，有限数学消除了隐藏的无穷性，使数学推理与有限执行保持一致。这些设计选择并非旨在取代经典数学，而是为了提供一个受控环境，使数学结构和算法能够在明确的资源界限下得到研究。

3. The Limited Math Framework

本节给出有限数学的形式化定义。我们首先定义有界数值域，接着介绍用于强制执行幅值和精度约束的值映射算子。然后，我们引入对集合基数的限制，以界定结构复杂度。

3.1. **数值域**。 有限数学由正整数 $b \geq 1$ 参数化，表示分配给每个数值分量的位数。我们定义

$$M := 2^b - 1.$$

参数 M 同时决定可表示幅度的最大值和最小正精度单位。

定义 3.1 （有限数学数值域）。有限数学数值域定义为

$$\mathcal{N}_M = \left\{ \frac{k}{M} \;\middle|\; k \in \mathbb{Z},\ -M^2 \leq k \leq M^2 \right\}.$$

$\mathcal{N}_M$中的每个值都是 $1/M$的精确整数倍，且绝对值最大为 M。可表示值的不同取值总数有限，数量级为 M^2。最大绝对值为 M，对应于 $k = \pm M^2$。

该数值域自然对应于一种具有有界幅值和均匀精度的对称定点表示。对于特殊情形 $b = 1$，可得 $M = 1$ 和$\mathcal{N}_1 = \{-1, 0, 1\}$，这构成了最小的非平凡有限数学系统。

3.2. **值映射算子**。为了将实数值量与有界域$\mathcal{N}_M$联系起来，有限数学采用确定性值映射算子，同时实现饱和与量化。

定义 3.2 （值映射）。 值映射算子 $\Phi_M : \mathbb{R} \to \mathcal{N}_M$ 定义如下

$$\Phi_M(x) = \begin{cases} M, & x \geq M, \\ -M, & x \leq -M, \\ \frac{\lfloor Mx \rfloor}{M}, & -M < x < M. \end{cases}$$

算子 Φ_M 将每个实数映射为一个可表示的有限数学值，在可表示范围内进行量化，并在边界处进行饱和处理。此定义确保有限数学中算术的语义解释是全定义且确定性的。

语义行为与执行行为。值映射算子 Φ_M 是一种语义构造：它定义了如何在有限的有限数学域中解释无界数学值。当有限数学嵌入具体编程语言或执行模型时，边界行为在运行时不一定要通过饱和来实现。相反，饱和是对边界结果的一种语义刻画，而实现可以选择显式报告边界违规（例如，通过触发陷阱），以强制遵守假定的数值界限。

3.3. **算术运算**。 有限数学中的算术运算通过将经典实值运算与值映射算子组合来定义。

定义 3.3 (LM 算术)。 对于 $x,\ y \in \mathcal{N}_M$，定义：

$$x \oplus y := \Phi_M(x + y), \qquad x \otimes y := \Phi_M(x \cdot y).$$

这些运算确保 $\mathcal{N}_M$ 在加法和乘法下封闭。当中间的实数值结果保持在有界范围 $[-M, M]$ 内，且与可表示网格对齐时，LM 算术与经典算术完全一致。当达到边界时，其语义结果由 Φ_M刻画；具体实现也可以检测此类越界并明确发出信号。

3.4. **有界集合基数**。 除了限制数值的范围外，有限数学还限制理论所允许集合的大小。

定义 3.4（有界集合）。 有限数学中的所有集合，其基数都不得超过 M^2。

这一限制反映了数值域 $\mathcal{N}_M$ 有限的表示容量，而该数值域本身包含 $O(M^2)$ 个不同的值。若允许集合超过此上限，就会引入无法通过 LM 算术在内部进行索引或区分的对象。

限制集合基数可确保所有集合都是有限的、可枚举的，并且与有界计算兼容。这也能防止通过不受限制的集合构造，将隐式的无穷构造引入该框架。

3.5. **讨论**。 数值域 $\mathcal{N}_M$、值映射算子 Φ_M 和有界集合基数共同构成了有限数学的核心结构。这些组成部分建立了一个有限但可扩展的数学宇宙，其中幅值、精度和结构复杂度都是明确的参数。

在后续章节中，我们将以此框架为基础定义函数和算子的映射，分析有界求值下的代数性质，并研究有限数学在实际场景中的计算影响。

4. Operator and Function Mapping

有限数学的核心目标之一，是在经典数学语义与有界执行之间建立严谨的对应关系。为此，有限数学区分函数和算子的语义定义（仍遵循经典语义）与它们在有界数值域中的执行。本节将这种区分形式化，并定义如何将函数和算子映射到有限数学中。

4.1. **函数映射**。 设 $f : \mathbb{R} \to \mathbb{R}$ 是一个具有经典数学定义的实值函数。有限数学为 f 关联一个在有限数学数值域上运行的有界对应算子。

定义 4.1（映射函数）。 给定一个实值函数 $f : \mathbb{R} \to \mathbb{R}$，其有限数学映射为函数

$$f^{(M)} : \mathcal{N}_M \to \mathcal{N}_M$$

定义为

$$f^{(M)}(x) := \Phi_M(f(x)), \quad x \in \mathcal{N}_M.$$

这个定义确保每次函数调用都会产生有界域内的值。无论 f超出可表示范围后的行为如何，该映射都是确定且全定义的。

这种语义定义与执行之间的区分，符合程序语言语义的标准方法：意义的定义独立于表示形式或求值策略 [13]。

4.2. **复合语义**。 在有界环境中，函数复合需要特别谨慎，因为朴素的中间量化可能会造成不必要的误差累积，并破坏代数结构。

设 $f, g\colon \mathbb{R} \to \mathbb{R}$ 为实值函数。经典复合定义为 $(f \circ g)(x) = f(g(x))$。有限数学将映射复合定义如下。

定义 4.2（映射复合）。 有限数学中 f 与 g 的复合定义为

$$(f \circ g)^{(M)}(x) := \Phi_M(f(g(x))), \quad x \in \mathcal{N}_M.$$

重要的是，只有在完整的实值函数复合求值之后才进行一次量化。中间结果不会被映射到 LM 网格上。

这种末端取整 语义可最大限度地减少量化误差，并保留语义上的结合性：

$$\big((f \circ g) \circ h\big)^{(M)} = \big(f \circ (g \circ h)\big)^{(M)} = \Phi_M(f(g(h(x)))).$$

4.3. **作为映射函数的算术运算符**。 有限数学中的算术运算符是一般映射原理的特例。

令 $+$ 和 $\cdot$ 分别表示经典的实数加法和乘法。它们在 LM 中的对应运算定义如下：

$$x \oplus y := \Phi_M(x + y), \qquad x \otimes y := \Phi_M(x \cdot y), \quad x, y \in \mathcal{N}_M.$$

这些定义与第 3 节中引入的定义一致，也符合函数映射框架。因此，算术运算可解释为先执行经典运算，再通过一次映射将结果映入有界域。

4.4. **范围内等价与边界效应**。 当中间实数值结果保持在可表示范围内，并与 LM 网格对齐时，映射算子就如同恒等映射。

命题 4.1 (范围内等价)。 设 $x,\ y \in \mathcal{N}_M$。若 $x + y \in \mathcal{N}_M$，则

$$x \oplus y = x + y.$$

同样，如果 $x \cdot y \in \mathcal{N}_M$，那么

$$x \otimes y = x \cdot y.$$

在此范围内区间之外，饱和与量化会显式引入相对于经典算术的偏差。这些边界效应是确定性的且可分析，并取代了实际计算中常见的隐式溢出或舍入行为。

4.5. **高阶算子**。 映射原理自然地扩展到高阶算子。令

$$\mathcal{O} : (\mathbb{R} \to \mathbb{R})^k \to (\mathbb{R} \to \mathbb{R})$$

可以是经典算子，例如微分或积分。有限数学通过以下方式定义映射后的算子：

$$\mathcal{O}^{(M)}(f_1, \ldots, f_k)(x) := \Phi_M(\mathcal{O}(f_1, \ldots, f_k)(x)), \quad x \in \mathcal{N}_M.$$

这种方法在强制有界执行的同时，保留了算子的经典语义含义。依赖无穷构造的算子不会被纳入有限数学内部；相反，需要时会将其结果映射到有界域中。

4.6. **讨论**。 算子与函数映射框架为经典数学与有界计算搭建了一座原则性的桥梁。通过将语义定义与执行分离，有限数学避免了过早量化，在可能的情况下保留代数结构，并在达到界限时明确指出偏差。

这种设计尤其适用于计算推理：函数以抽象方式定义，却在有限精度和资源约束下执行。

这种语义分离体现了编程语言设计中通常采用的指称意义与操作实现之间的区分。

5. Analysis of Algebraic Properties

本节分析有限数学运算的代数行为。我们区分范围内区间和边界区间：在前者，中间结果始终处于可表示边界内；在后者，则会发生饱和或量化。这一区分阐明了在有界环境下，哪些经典代数定律得以保持，哪些则必然被破坏。

5.1. **范围内区间**。 如果一个表达式的所有中间实数值结果都处于可表示区间 $[-M, M]$ 内，并与 LM 数值网格对齐，则称该表达式是在范围内 求值的。

定义 5.1（范围内求值）。 如果一个 LM 表达式在采用末端取整语义求值时，所有中间实数值结果都属于 $\mathcal{N}_M$，则称该表达式是在范围内求值的。

在此区间内，值映射算子充当恒等映射，LM 算术与经典算术一致。

定理 5.1（范围内加法律的保持）。 设 x、y、$z \in \mathcal{N}_M$。若 $x + y$、$y + z$和 $x + y + z$ 都属于 $\mathcal{N}_M$，则：

- $x \oplus y = y \oplus x$ *(交换律)*,
- $(x \oplus y) \oplus z = x \oplus (y \oplus z)$ *(结合律)*。

证明。 在所述条件下，不会发生饱和或量化。根据定义，

$$x \oplus y = \Phi_M(x + y) = x + y,$$

所有中间和也同样如此。因此，经典的结合律和交换律可以直接适用。

□

当乘积保持在范围内且可表示时，乘法也有类似的结果。

5.2. **边界区间与规律失效**。 当中间结果超出可表示界限或落在网格点之间时，值映射算子会引入饱和或量化。在此区间内，某些经典代数定律不再成立。

命题 5.1 （饱和条件下的结合律失效）。 存在 x， y， $z \in \mathcal{N}_M$ 使得

$$(x \oplus y) \oplus z \neq x \oplus (y \oplus z).$$

证明。 设 $x = M$、 $y = M$ 和 $z = -M$。则：

$$(x \oplus y) \oplus z = \Phi_M(\Phi_M(2M) - M) = \Phi_M(M - M) = 0,$$

而

$$x \oplus (y \oplus z) = \Phi_M(M + \Phi_M(0)) = M.$$

□

此例表明，一旦涉及饱和，结合律便会失效。重要的是，这种失效是明确且确定性的。

5.3. **分配律**。 加法与乘法之间的分配律也只在范围内区间成立。

命题 5.2 (条件分配律**)**。 设 $x, y, z \in \mathcal{N}_M$。如果 $x(y + z)$、 xy 和 xz 都属于 $\mathcal{N}_M$，那么

$$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z).$$

在此区间之外，量化或饱和可能导致分配律失效。这种行为反映了数值域的有界性，并以显式语义取代隐式的溢出行为。

5.4. **消去律与顺序敏感性**。 经典消去律通常不适用于有限数学。

命题 5.3 （消去律失效）。 存在 x、 y、 $z \in \mathcal{N}_M$ 使得

$$x \oplus y = x \oplus z \quad but \quad y \neq z.$$

此类失效源于饱和或量化，并意味着一旦出现边界效应，表达式求值顺序就可能影响结果。

5.5. **解释**。 边界区间中某些代数定律的失效，并非有限数学的缺陷，而是明确给出界限所带来的结果。在经典数学中，这种边界行为会通过假设予以排除；在实际计算中，它往往被隐藏起来，或未作定义。

有限数学直接呈现这些效应，并将其限制在定义明确的计算区域内。在范围内区间，经典推理依然适用，不受影响。在其范围之外，偏差则是明确、确定且可分析的。

5.6. **总结**。 有限数学支持双层代数解释：

- 范围内，LM 算术与经典代数一致；
- 在边界处，饱和与量化会引入受控的偏差。

这种结构使我们能够精确推理：何时可以安全地应用经典代数定律，何时必须考虑有界效应。

6. Derivatives and Analytic Operators

极限、导数和积分等解析概念在经典数学中发挥着核心作用。然而，这些概念从根本上依赖无穷构造和任意小的扰动。由于有限数学在有限离散域中运作，且精度和幅值都有明确界限，因此若不违背该框架的设计原则，就无法将这类构造纳入其中。

本节阐明有限数学中解析算子的地位，并提出一种有原则的方法，在不损害严谨性或稳定性的前提下将其纳入其中。

6.1. **为什么导数并非有限数学的内在要素**。在经典分析中，函数 f 在点 x处的导数定义为一个极限：

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

该定义预设了 h 任意小的非零值存在，并且存在一个无限细化的过程。相比之下，在有限数学中：

- 数值域是有限且离散的；
- 最小正单位固定为 $1/M$；
- 无法表达对无穷序列取极限。

因此，经典导数无法在 LM 数值域内以内在方式定义。尝试在离散网格上使用有限差分近似导数，会导致不稳定性和伪影，尤其是在函数经过量化时。例如，斜率较小的线性函数在基于网格的差分中可能会表现为局部常数。

这一局限并非偶然，而是结构性的：任何满足线性、局部性和尺度不变性的内在导数算子，都会重新引入与有限数学有界特性不相容的无穷假设。

6.2. **解析算子的语义解释**。 有限数学并不将解析算子内化，而是将其视为在经典数学层面定义的语义构造。随后，使用值映射算子将其结果映射到有界域中。

定义 6.1（映射导数）。 设 $f : \mathbb{R} \to \mathbb{R}$ 是一个经典意义上的可微函数。有限数学中 f 的导数定义为

$$(Df)^{(M)}(x) := \Phi_M\big(f'(x)\big), \quad x \in \mathcal{N}_M.$$

该定义保留了经典微分的含义，同时确保导数值能够在有限数学域内表示。量化仅在最终阶段进行一次，从而避免因反复贴合而导致的不稳定性。

6.3. 稳定性考量。 语义映射方法避免了与离散微分相关的常见病态现象。例如，考虑线性函数$f(x) = 0.3x$。在直接应用于有限数学映射函数的朴素有限差分格式中，微小增量可能因量化而丢失，从而在较大区域内得到零导数。然而，根据映射导数定义，经典导数$f'(x) = 0.3$得以保留，并且在 $0.3 \in \mathcal{N}_M$时能够精确表示。

这说明了一条普遍原则：在有界域内，将微分视为语义操作而非运算过程，能够保留其分析意义。

6.4. 扩展到其他解析算子。 同样的方法也适用于其他解析算子，例如积分或高阶微分。设

$$\mathcal{O} : (\mathbb{R} \to \mathbb{R})^k \to (\mathbb{R} \to \mathbb{R})$$

be 一个经典解析算子。有限数学定义了它的有界对应算子 by

$$\mathcal{O}^{(M)}(f_1, \ldots, f_k)(x) = \Phi_M(\mathcal{O}(f_1, \ldots, f_k)(x)), \quad x \in \mathcal{N}_M.$$

这种统一处理确保了解析构造之间的一致性，同时保持有限数学的有界性和确定性，且不会将无穷假设重新引入数值域。

7. Bounded Computation and Memory

前文各节将有限数学定义为一种在数值、函数和结构层面具有界限的数学框架。本节将考察在计算机上实现有限数学时，这些界限所带来的影响。具体而言，我们将分析对内存施加明确界限的影响，并说明这将如何导出一种具有强大理论保证的有限状态执行模型。

本节将区分执行语义与执行模型：前者描述程序行为的含义，后者则提供实现该行为的具体数学结构。

7.1. 有界内存执行模型。 实际的计算系统使用有限内存运行。经典计算模型（例如图灵机）虽然假设纸带无界，但这类假设属于抽象，在实践中并不成立。当有限数学在计算机上实现时，所有数据结构、寄存器和控制状态都必然受到界限约束。

因此，我们考虑一种满足以下条件的执行模型：

- 数值的范围为有限域 $\mathcal{N}_M$；
- 所有集合的基数均有界；
- 总内存使用量受固定的有限上限约束。

在这些假设下，LM 程序的全局执行状态由有限个组成部分构成，包括程序计数器、有界内存内容，以及任何辅助寄存器或栈。这类状态的总数是有限的。

7.2. 有限状态执行语义。 令 Σ 表示有界内存 LM 程序所有可能的全局执行状态的集合。由于状态的每个组成部分都取自有限集合，Σ 本身也是有限的。

程序执行会导出一个确定性转移函数

$$\delta : \Sigma \to \Sigma \cup \{\mathsf{HALT}\},$$

其中 HALT 表示终止停机状态。

此类有限状态转移系统是验证技术（如模型检测）的核心，相关技术会在有界状态空间 [2] 上分析可达性和终止性。

7.3. 终止或循环性质。 状态空间的有限性直接推出以下结果。

定理 7.1 (终止或循环)。 对于任何在固定内存上限下执行的确定性有限数学程序，以及任何固定输入，其执行要么在有限步后停机，要么最终重新访问先前遇到的某个全局状态，并从此进入循环。

证明。 由于全局状态集合 Σ 是有限的，任何无限执行都必定会多次访问某个状态。转移函数具有确定性，这意味着一旦某个状态再次出现，后续执行就会重复相同的状态序列，从而形成一个循环。如果没有发生这种重复，执行就必定会到达停机状态。 □

7.4. 有界环境中的终止性的可判定性。 终止或循环性质的一个直接推论是，在有界执行模型中，终止性是可判定的。

给定一个固定的 LM 程序及其输入，可以通过记录访问过的状态来模拟执行。如果执行到达停机状态，该程序就会终止。如果遇到先前访问过的状态，执行就会无限循环，永远不会停机。

这一结果并不与经典不可判定性定理相矛盾，例如停机问题；这类定理适用于具有无界内存的计算模型。相反，这反映了这样一个事实：有界内存系统对应于有限状态机，而对于有限状态机，可达性和终止性原则上都是可判定的。

7.5. 实际解释。 尽管有界内存 LM 程序的状态空间在实践中可能极其庞大，但终止或循环行为这一理论保证具有重要的概念意义。它确保：

- 非终止不会由无界状态增长导致；
- 无限执行对应于显式的循环行为；
- 程序行为完全由有限状态转移系统决定。

这些性质对于验证、安全关键系统和长时间运行的计算尤其重要，因为理解并控制非终止是必不可少的。

7.6. **总结**。 通过在数值和结构约束之外，对内存施加明确的界限，有限数学为已实现的程序提供了有限状态执行语义。在这种语义下，每次计算要么终止，要么最终进入循环，因此终止性原则上变得可判定。这一结果进一步凸显了有限数学作为一种框架的作用：它使数学推理与有限计算的现实相一致，而无需诉诸理想化的无界模型。

8. Applications and Case Studies

本节展示有限数学如何应用于各种计算场景。我们不追求面面俱到，而是聚焦于具有代表性的场景，说明对幅值、精度、结构和计算设定明确界限，如何带来更清晰的语义和更可预测的行为。

8.1. **最小系统：运用** $M = 1$ **进行定性推理**。 当 $b = 1$ 时，便得到最小的非平凡有限数学系统，即 $M = 1$ 和

$$\mathcal{N}_1 = \{-1, 0, 1\}.$$

尽管这个系统很简单，但它适用于定性推理和决策。这三个值可以分别解释为负向、中性和正向状态，也可以解释为拒绝、不确定和接受。算术运算可简化为饱和加法和符号乘法，从而产生确定且易于解释的行为。

这个最小系统非常适合用于：

- 三值逻辑与规则评估；
- 粗粒度决策系统；
- 基于方向性反馈的控制逻辑；
- 受限环境中的可解释推理。

该 $M = 1$ 案例表明，有限数学不仅仅是一种数值近似框架，更是一类系统，其中纯定性计算是其极限情形。

8.2. **选择** M **：根据问题规模**。在实际应用中，参数 M 根据问题领域的规模和精度要求进行选择。较小的 M 值可实现粗略但稳健的计算，而较大的值则能提供更精细的分辨率，但表示成本也会随之增加。

例如：

- 小型离散系统（例如计数器、团队规模、政策阈值）可能只需要适中的精度，因此 $M = 15$ 或 $M = 31$ 这样的值就足够了。
- 通用数值计算可以使用 $M = 255$ 这样的适中值来支持，对应于分辨率为 $1/255$ 的均匀定点精度。
- 对于科学与工程应用，测量不确定性往往远大于 $M = 1023$ 或 $M = 65535$ 的最小 LM 单位，因此这类值足以满足许多任务的需求。

这种显式参数化与浮点运算形成对比：浮点运算的精度和范围由硬件固定，且往往对用户而言并不透明。

8.3. 示例性实例化：基于 C 的算术模型。有限数学旨在作为一种语义框架，而非具体的编程语言设计。不过，在熟悉的执行模型中实例化该框架，有助于阐明其语义原则如何与有限表示和有界算术相互作用。为此，我们考虑一个基于 C 编程语言算术模型的示例性实例化；该模型对定宽整数的处理与常见硬件行为密切相符。值得注意的是，C 标准规定有符号整数溢出的行为未定义，同时明确规定了算术行为的其他方面 [7]。

定点实例化。我们使用带符号的定点表示来实例化有限数学：整数幅值占 8 位，小数精度占 8 位（Q8.8）。数值存储为 16 位二进制补码整数，并解释为 $\mathrm{val}(x) = x/2^8$. 由此得到与 LM 框架一致的有限数值域，具有统一的精度以及明确的幅值界限。

加法语义。 对于两个 Q8.8 数值 $x,\ y$，加法通过将两个操作数扩展为更大的整数类型、计算和，并检查结果是否处于可表示范围内来执行。如果结果在范围内，则原样返回；否则，执行陷阱。从语义上看，这相当于先执行经典加法，再进行一次边界检查，体现了配备溢出检测的硬件加法器的行为。

乘法语义。 乘法会扩展操作数以生成 32 位中间乘积，然后通过右移 8 位重新缩放，以恢复 Q8.8 精度。右移之前可以选择应用对称舍入。如果重新缩放后的结果超出可表示范围，执行陷阱；否则，结果将作为 16 位数值返回。这体现了定点算术中采用的标准扩展–乘法–重新缩放数据通路。

基于陷阱的边界策略。与通常采用回绕或饱和处理的定点算术不同，有限数学采用触及边界即触发陷阱的策略。根据这种解释，假定界限 M 足以涵盖所有预期的语义求值。因此，达到数值边界表明违反了假定的语义域，而不是产生了另一种算术结果。

语义解释。在这种实例化方式下，LM 算术运算符对应于经典实值运算，随后只进行一次语义边界检查。当所有中间结果都处于范围内时，LM 算术与经典算术完全一致。当超出边界时，偏差会明确且确定地显现。这说明，LM 如何在保留可表示极限内经典行为的同时，将数学含义与有界求值区分开来。

这个示例表明，有限数学可以在熟悉的有限算术模型中实例化，而无需引入新的表示形式或执行机制。其目的在于说明：如何在不依赖任何特定编程语言设计或实现策略的前提下，具体实现有界语义解释。

8.4. 有界计算与验证。 结合有界内存后，有限数学可形成有限状态执行模型。这对程序验证与分析具有直接意义。

在安全关键型和长期运行的系统中，非终止和不受控制的状态增长是主要问题。在有限数学执行模型下，每次计算要么终止，要么进入循环。这使得：

- 能够在有界环境中进行可判定的终止性分析；
- 能够明确检测循环行为；
- 原则上能够穷举探索执行状态。

虽然由此产生的状态空间可能很大，但这些理论保证支持在受限环境中对程序行为进行精确推理。

8.5. 与物理界限的概念一致性。 尽管有限数学的动机来自计算而非物理学，但在概念上，它与这样一种观点相容：物理系统同时存在有意义尺度的最小值和最大值。例如，物理理论表明，空间分辨率存在下限，可观测范围存在上限。

谨慎解读时，这一观点与有限数学的有界特性相契合：二者都摒弃未经审视的无穷，转而采用明确的界限。这里并未声称有限数学能够完整、细致地模拟物理现实；相反，这一类比凸显了有界数学推理更广泛的意义。

8.6. 总结。 这些案例研究表明，有限数学适用于各种计算场景，从定性推理到数值编程，再到有界验证。通过将界限明确设定为理论的核心，有限数学提供了更清晰的语义、更可预测的行为，以及一个用于推理有限计算的统一框架。

9. Discussion、 Related Work、 and Scope

有限数学与若干成熟的研究领域存在交集，包括有限精度算术、有界计算和计算形式模型。本节将有限数学置于这一研究格局中，阐明其与现有方法的关系，并讨论其预期范围与局限。

9.1. 与浮点与定点算术的关系。 有限精度算术在理论和实践中都得到了广泛研究，涉及数值分析、编程语言设计和硬件架构。IEEE 754 对浮点运算进行了标准化，它是现代编程语言和硬件平台中的主流表示方式 [6, 4], 。这种设计强调将算术行为明确纳入编程模型，其历史可以追溯到浮点标准发展的早期 [8]。

尽管浮点运算无处不在，但众所周知，它会因舍入、非均匀间距、非结合性和异常值而表现出违反直觉的行为。这些行为使关于数值正确性的形式化推理变得复杂，并推动了围绕数值稳定性、误差分析以及

稳健算法设计的大量研究 [5]。当使用实数抽象分析浮点计算时，这些行为也给程序验证带来了重大挑战[11]。

长期以来，人们主要通过语义抽象技术，尤其是抽象解释 [3]，来推理有界表示下的数值行为。从编程语言的角度来看，理想化实数模型与有限精度执行之间的语义鸿沟，早已被认为是微妙错误和验证挑战的来源。语言规范通常使用实数抽象描述算术，同时依赖底层浮点实现，却未明确说明舍入行为和边界效应，或对此规定不充分。这种不匹配使得在存在有限精度的情况下，难以推理相等性、结合律和程序等价性等性质。

定点算术通过采用统一精度和明确限定的幅值，提供了一种更简单且更可预测的替代方案。它广泛应用于嵌入式系统和数字信号处理领域，在这些领域中，资源约束和确定性行为至关重要。然而，定点算术通常被视为一种实现或表示技术，而非具有明确语义解释的数学框架 [9]。其正确性通常从位宽选择和缩放规范的层面进行论证，而不是通过形式化语义模型来论证。

有限数学与浮点和定点方法的不同之处在于，它将有界幅值和有界精度视为数学域本身的公理属性。有限数学既不提出新的数值格式，也不规定特定的硬件或语言级表示。相反，它提供了一个语义框架，在其中，量化、边界行为以及偏离经典算术的情况都是明确、确定且可形式化分析的。从这个意义上说，有限数学通过为推理现有数值表示所带来的语义后果提供原则性依据，补充了这些表示，而不是充当一种替代性的数值编码。

9.2. 为什么有限数学不只是定点算术。一个自然的问题是：有限数学是否只是以不同的表述方式归结为定点算术。尽管有限数学与定点表示在表面上有某些相似之处——例如精度均匀且幅值有界——但两者在意图、抽象层次和范围上存在根本差异。

关于定点和半数值计算的经典论述，主要关注表示选择、缩放策略和实现技术。这些论述探讨如何在有限硬件上高效地编码和操作数字，基础性的半数值算法研究 [9]便是其中的例子。在这种情境下，正确性通常从操作层面论证，例如通过避免溢出或选择足够的位宽，而非从语义层面论证。

实践中通常使用的定点算术规定了数值的存储和计算方式，但其本身并未说明经典数学含义如何与有界执行相关联。具体而言，它没有解释代数定律在数值边界处如何变化、误差如何在复合表达式中累积，或经典推理在何种条件下仍然有效。

相比之下，有限数学是在语义层面而非表征层面定义的。经典实值函数和算子保留其数学意义，而单一的值映射算子则在无界语义与有界

执行之间发挥中介作用。这种将意义与执行分离的做法，类似于程序语言语义中的标准方法：指称意义独立于具体的求值策略而定义 [13]。

有限数学将有界幅值、有界精度和边界行为视为公理，而非偶然现象，从而提供了一个框架，用于推理偏离经典算术的时机 与原因 。从这个意义上说，有限数学并非一种替代性数值格式，而是用于理解有界计算及其后果的语义框架。

9.3. 与有界算术和有限模型论的关系。 有界算术和有限模型论研究具有明确有限性约束的数学与逻辑系统，将规模界限视为根本属性，而非对无限结构的近似。这些领域通过限制计算和推理所涉及的论域，在表达能力、可判定性和复杂性方面取得了深刻成果。

尤其是有限模型论，它研究逻辑公式在有限结构上的行为，并揭示有限性会从根本上改变模型的性质、表达能力和可定义性 [10]。通过摒弃隐含的无穷假设，它强调，在有限域上进行推理需要独特的语义与方法论工具。

有限数学在哲学上与这一观点相契合，因为它拒绝未经审视的无穷，并将有界性视为一项基本属性。有限数学对集合基数的限制避免了隐含的组合爆炸，并确保所有集合都是有限的、可枚举的，且可在内部表示。在这一点上，有限数学秉持有限模型论的直觉：不能不加后果地假定无界构造。

然而，有限数学在关注重点和目的上有着根本不同。有界算术和有限模型论主要关注逻辑表达能力、证明系统和复杂性理论分类。相比之下，有限数学不是一种逻辑理论，也不旨在刻画在给定复杂性类别中能够证明或计算什么。它关注的是数值语义：有界幅值、有界精度和有界结构如何与算术运算及程序执行相互作用。

因此，有限数学应被视为对这些理论框架的补充，而非与之竞争。它采用这些框架所倡导的有限视角，但将其用于编程语言中数值计算的语义解释，而非逻辑可定义性或证明论强度。

9.4. 与经典计算模型的关系。 经典计算模型，包括图灵机和λ演算，都假设内存无界且论域无限。这些抽象为停机问题不可判定性等基础性结果奠定了基础 [15] ，并且对于理论计算机科学仍不可或缺 [14]。然而，当计算受到明确的资源界限约束时，执行必然局限于有限状态空间；这一视角已在形式化验证领域通过有限状态自动机模型和资源有界自动机模型得到广泛研究 [1]。

有限数学并不质疑这些模型及其相关结果。相反，它遵循形式化处理程序语言语义时所强调的、抽象意义与具体执行模型之间的标准语义分离 [16]。随后，它采用一种互补的视角，反映计算在物理机器上的实际运行情况

。当数值、数据结构和内存都受到明确限制时，程序执行必然局限于有限状态空间。

在此类有界条件下，执行可以建模为有限个全局状态上的确定性转换系统。在这种设定中，每次计算要么到达停机状态，要么最终重新访问先前遇到的状态并进入循环。这一观察并非新的理论结果，也不会削弱经典不可判定性定理，因为这些定理本就以无界模型为前提。

有限数学的贡献在于将这种有限状态解释明确地纳入语义。通过使数学推理与有界执行相对齐，LM 提供了一个精确框架，用以理解经典的非终止、发散和不可判定性概念如何在真实计算系统中体现，或无法体现。

9.5. **范围与局限**。 有限数学有意限制表达能力，以使资源界限明确且具有语义意义。特别是，极限、导数和积分等分析学概念并非 LM 数值域的内在组成部分；若不重新引入无穷假设，就无法对其进行操作性定义。区间算术等替代方法通过将实数值结果限制在数值界限内来提供正确性保证，但其目标和权衡与有限数学所强调的语义对齐不同 [12]。如前所述，这类构造是在语义层面加以处理，而非纳入有界框架之中。

同样，经典代数定律在有限数学中只有在明确满足范围内条件时才成立。一旦达到数值界限或精度限制，饱和或量化效应就可能导致结合律、分配律或消去律失效。这些偏差并非该框架的伪影，而是明确引入有限精度和有界幅值所带来的直接后果。

因此，有限数学并不旨在取代经典数学、实分析或无界计算模型。它的目标更为有限，也更为明确：提供一个语义框架，使有限性带来的后果清晰可见、具有确定性且可供分析。有限数学最适合用于推理数值行为、程序正确性和执行性质，尤其是在有界精度、有界结构和有界资源既不可或缺又无法避免的环境中。

9.6. **总结**。 有限数学借鉴了数值计算、有限结构和形式语义中的思想，同时提出了独特的视角。通过将有界性视为基本原则，而非实现细节，有限数学为在真实编程语言中推理数值行为和计算提供了连贯的框架。它的贡献在于阐明有限性的语义后果，并使这些后果明确、可分析且具有严格的数学依据。

10. Conclusion and Future Work

本文提出了有限数学（*LM*），这是一种有界数学框架，旨在使形式化推理与有限计算的现实相契合。有限数学明确规定数值幅度、数值精度、结构复杂度以及（在实现时）内存的界限，从而消除传统计算数学抽象中常见的隐含无穷。

LM 的核心是一个有限数值域，并配备了一个确定性值映射算子，用以实施饱和与量化。函数和算子先在经典层面进行语义解释，再映射到有界域中，从而将含义与执行分离，同时尽可能减少不必要的精度损失。在范围内条件下，LM 与经典算术一致；在边界处，偏差则明确、确定且可分析。通过进一步限定集合基数，LM 确保所有集合均可枚举且可在内部表示，防止无穷行为通过组合构造再次引入。

在计算机实现中与有界内存相结合时，有限数学会产生一种有限状态执行模型，在该模型中，每次计算要么停止，要么最终进入循环。这一结果并不与经典不可判定性定理相矛盾，而是反映了有限机器的实际运行情况，并使我们能够在有界环境中精确推理程序行为。

有限数学并非旨在取代经典数学或实分析，而是作为一种互补框架，用于研究明确受资源约束的数学与计算。它的价值在于为有限精度算术、有界结构和有限计算提供更清晰的语义、可预测的行为，以及统一的推理基础。

未来工作。 这项工作自然引出了若干未来研究方向，包括深入分析 LM 算术下的稳定性与误差累积，将 LM 语义纳入编程语言规范和验证工具，以及探索基于 LM 的优化与控制模型。对有限数学的概率或随机变体进行扩展研究，以及在真实程序中将其与浮点语义进行实证比较，也都是值得进一步探索的方向。

有限数学将有界性视为一项基础原则，而非实现层面的产物，由此为更明确、更严谨地理解实际执行的计算开辟了一条道路。

References

[1] Rajeev Alur 和 David L. Dill。《定时自动机理论》。《 *Theoretical Computer Science*》，126(2):183– 235，1994年。[2] Edmund M. Clarke、Orna Grumberg 和 Doron A. Peled。《 *Model Checking*》。麻省理工学院出版社，1999年。[3] Patrick Cousot 和 Radhia Cousot。《抽象解释：程序静态分析的统一格模型》。《 *Proceedings of the ACM Symposium on Principles of Programming Languages (POPL)*》，第238–252页，1977年。[4] David Goldberg。《每位计算机科学家都应该了解的浮点运算知识》。《 *ACM Computing Surveys* 23 1 5 48, 1991 》，( ): – 。[5] Nicholas J. Higham。《 *Accuracy and Stability of Numerical Algorithms*》。SIAM，2002年。[6] 电气电子工程师学会。《IEEE浮点算术标准》。技术报告 IEEE Std 754-2019，电气电子工程师学会，2019年。[7] 国际标准化组织。《编程语言——C语言》。技术报告 ISO/IEC 9899:2018，国际标准化组织，2018年。[8] William Kahan。《IEEE二进制浮点算术标准754》。讲义，加利福尼亚大学伯克利分校，1996年。[9] Donald E. Knuth。《 *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*》。Addison-Wesley，1997年。[10] Leonid Libkin。《 *Elements of Finite Model Theory*》。施普林格，2004年。

[11] David Monniaux。《验证浮点计算的陷阱》。《 *ACM Transactions on Programming Languages and Systems* 30 3 2008 》，( )，。[12] Ramon E. Moore。《 *Introduction to Interval Analysis*》。SIAM，2009年。[13] Benjamin C. Pierce。《 *Types and Programming Languages*》。麻省理工学院出版社，2002年。[14] Michael Sipser。《 *Introduction to the Theory of Computation*》。圣智学习，2012年。[15] Alan M. Turing。《关于可计算数及其在判定问题中的应用》。《*of the London Mathematical Society* 1936 *Proceedings* 》，。[16] Glynn Winskel。《 *The Formal Semantics of Programming Languages*》。麻省理工学院出版社，1993年。

本作品采用知识共享署名许可协议授权。如需查看该许可协议，请访问 https://creativecommons.org/licenses/by/4.0/，或致函知识共享，第二街171号300室，旧金山，加利福尼亚州 94105，美国；或艾森纳赫街2号，10777 柏林，德国