机器学习模型的查询语言

Martin Grohe 亚琛工业大学,
德国

摘要

本文讨论带权有限结构的两种逻辑：带求和的一阶逻辑（FO(SUM)）及其递归扩展 IFP(SUM)。这些逻辑源于 Grädel、Gurevich 和 Meer 在20世纪90年代的奠基性工作。在近期与 Standke、Steegmans 和 Van den Bussche 的合作研究中，我们将这些逻辑作为机器学习模型的查询语言进行了研究，具体而言，研究了神经网络；神经网络通常表示为加权图。我将介绍一些可用这些逻辑表达的神经网络查询示例，并讨论其表达能力和计算复杂度方面的基本结果。

2012年 ACM 主题分类 计算理论 → 数据库查询语言（原理）；计算理论 → 有限模型论；计算理论 → 机器学习理论

关键词和短语 查询语言的表达能力、不动点逻辑、加权结构、神经网络、可解释人工智能

资助 本项目由欧盟资助（ERC、SymSim、101054974）。但文中所表达的观点和意见仅代表作者本人，并不一定反映欧盟或欧洲研究理事会的观点和意见。欧盟和资助机构均不对这些观点和意见承担责任。

致谢 感谢 Nicole Schweikardt、Erich Grädel 和 Jan Van den Bussche 对本文早期版本提出宝贵意见。

arXiv:2601.09381v1 [cs.LO] 14 Jan 2026

1 引言

背景

在20世纪80年代，有限模型论 [13, 29] 应运而生，成为数理逻辑的一个新分支，其动机是将数理逻辑中的工具应用于算法以及计算问题复杂性的分析。由于许多情况下，问题实例是有限结构，例如图、布尔公式或电路，或者更一般地说，是有限字母表上的有限字符串，因此需要有限模型论。原则上，任何计算问题的输入都可以描述为二进制字母表上的有限字符串。然而，自然用于描述计算问题的数学模型通常包含超出离散有限结构的数值分量。

有限模型论的一个重要应用领域是关系数据库 [1]。粗略来说，关系数据库是有限关系结构，而 SQL 查询语言的核心是关系演算，也就是一阶逻辑。从这一视角出发，可以获得有价值的见解，例如了解查询语言的表达能力。然而，现实中的数据库具有无限域的数据类型，例如整数；SQL 等查询语言能够表达对这些无限域进行算术运算或其他操作。约束数据库 [27] 甚至会考虑无限模型中的有限定义关系，例如实数有序域中的关系，并通过结合有限模型论与无限模型论的方法进行分析。混合动力系统 [12] 将连续动力学与离散控制结合起来。它们通常由定时自动机或混合自动机等模型描述，并使用包含离散分量和连续分量的逻辑进行分析。加权图是许多优化问题的输入，也是另一种典型的混合模型，包含离散分量和数值分量。或许，

当前计算机科学研究中最重要的混合模型是机器学习模型，尤其是神经网络，它们也可以视为加权图。

尽管所有这些混合模型都包含无限的、通常是连续的部分，但有限模型论的方法似乎也很适合分析混合模型的计算层面。例如，用于描述计算并刻画复杂性类的逻辑，以及用于分析表达能力的 Ehrenfeucht-Fraïssé 游戏和局部性技术。认识到这一点后，Grädel 和 Gurevich [17] 系统地将有限模型论的框架扩展到他们所称的“元有限模型”。他们将有限模型论中的重要逻辑扩展到了元有限领域，并发展出一种基础的描述复杂性理论。Grädel 和 Meer [18] 进一步发展了针对实数上的混合模型的描述复杂性理论，而这些模型正是我们在此最感兴趣的结构。多年来，人们还提出了各种适用于混合模型的其他逻辑，其一般性程度各异，应用目标也各不相同，例如， [14, 15, 28, 32, 33].[1]我还想提一下关于半环语义逻辑的一项非常有趣的研究，例如，[8, 11, 16, 19], 这类逻辑与本文讨论的逻辑不同之处在于，它们不在语言中显式处理权重和数值，而只将其作为公式的语义值。

查询机器学习模型

从刚才讨论的意义上说，机器学习模型是典型的混合模型。由于它们无处不在，人们正从许多不同的视角对其进行研究。我想提出，一阶逻辑（或元有限）模型论和数据库理论的视角，尤其是查询机器学习模型这一理念，有望带来新的见解和方法。

现代数据科学项目在其生命周期中会生成大量模型制品，包括训练运行、超参数配置、架构变体和性能指标。尽管 MLflow 等现代平台提供了有价值的基于元数据的筛选和搜索功能，便于整理实验，但它们几乎无法支持对模型本身的内部行为或语义进行查询。此处的“查询”模型，远不止是在特定输入上评估模型；它还允许我们与模型的内在运作进行交互。模型本身是复杂的结构。特定查询或许能帮助我们了解模型中各个组件的功能。或者从语义层面来看，我们可以将目标设为模型的全局行为，也就是模型对所有输入而非仅对某个特定输入的行为，例如，了解模型的鲁棒性或验证其正确性。

至关重要的是，我们追求的并非一组固定、预先规定的查询，而是一种灵活的语言，使我们能够针对模型的特定属性（可能处于特定应用场景中）制定查询，并通过自适应地细化查询与模型交互。与基于固定、通用解释模式的现有方法（例如针对特定结果给出反事实解释）相比，这或许能让我们更深入地理解或解释机器学习模型。查询模型也是验证的基础（只是换了一种语言来表述：查询相当于数据库领域中的模型检测）。

在 [2, 3],中，作者研究了适用于查询决策树等离散机器学习模型的布尔语言，并分析了这些语言的表达能力和复杂性。然而，大多数机器学习模型都是混合模型（如上文所述）

[1] .....仅举几例。我无意追求面面俱到，本文也并非一篇综述。不妨将其视为一种“个人视角”。尽管如此，这一领域仍值得更全面的探讨。

由离散和连续两部分组成。要全面访问这些模型，查询语言应当体现这一点。在第 3 节和第 4 节中，我们将介绍具备这一能力的两种语言：FO(SUM) 和 IFP(SUM)。这些逻辑源自 Grädel 和 Meer [18] 以及 Grädel 和 Gurevich [17]的研究；我们在此讨论的确切版本最近才在[22, 23]中提出。这两种语言都承袭了数据库查询语言 [1, 9]和有限模型理论的传统，也可视为 Grädel 和 Gurevich 元有限模型理论框架 [17]的具体实例。

需要指出的是，本文将逻辑视为机器学习模型的查询语言，这一视角与研究机器学习模型（如图神经网络和 Transformer）的表达能力的逻辑刻画工作有着根本不同 [4, 20, 21, 31]。

2 预备知识

我们用 $\mathbb{R}, \mathbb{Q}, \mathbb{N}$ 表示实数、有理数和自然数（包括 0）的集合，并令$\mathbb{R}_\perp := \mathbb{R} \cup \{\perp\}$ 和 $\mathbb{Q}_\perp := \mathbb{Q} \cup \{\perp\}$，其中 $\perp$ 是表示未定义值的附加元素。我们将通常的序关系 $\leq$ 从 $\mathbb{R}$ 和 $\mathbb{Q}$ 扩展到 $\mathbb{R}_\perp$ 和 $\mathbb{Q}_\perp$，规定对所有$x \in \mathbb{R}$，$\perp < x$。我们通过规定 $x + y := \perp$, $x - y := \perp$，以及当 $x = \perp$ 或 $y = \perp$时，$x \cdot y := \perp$，来扩展加法、减法和乘法。类似地，当 $x = \perp$或 $y = \perp$ 或 $y = 0$时，规定 $x/y := \perp$，以此扩展除法。

我们用粗体字母表示元组，例如 $\boldsymbol{x} = (x_1, \ldots, x_k)$，并用 $|\boldsymbol{x}|$ 表示该元组的长度 k。

2.1 加权结构

我们在此讨论的混合模型采用加权（有限、关系型）结构的形式。加权词汇表 是由有限个关系符号 和权重函数符号组成的集合。每个符号 S 都有一个元数 $\mathrm{ar}(S) \in \mathbb{N}$。注意，我们允许元数为 0 的关系和权重函数。虽然 0 元关系并没有特别有趣，但 0 元权重函数可能很有用，因为我们可以将它们视为权重常量。一个 Υ 结构 A 由一个有限非空集合 A（即 A 的论域 ）构成；对于每个关系符号 $R \in \Upsilon$，都有一个关系 $R^{\mathcal{A}} \subseteq A^{\mathrm{ar}(R)}$；对于每个权重函数符号 $F \in \Upsilon$，都有一个函数 $F_{\mathcal{A}}\colon A^{\mathrm{ar}(F)} \to \mathbb{R}_\perp$。我们始终用花体字母 A、B . . . 表示结构，并用相应的罗马体字母 A、B、. . .表示它们的论域。

▶ **例 2.1.** 加权图可以描述为二元权重函数符号 wt 的 {wt}-结构 G。G 的论域是该图的顶点集；对于所有顶点 v, w，如果存在一条从 v 到 w 的边，那么 $\mathsf{wt}^{\mathcal{G}}(v, w) \in \mathbb{R}$ 就是这条边的权重；否则，$\mathsf{wt}^{\mathcal{G}}(v, w) = \perp$。对于加权图 G，我们通常用$V\mathcal{G}$ 而不是 G来表示顶点集（论域），并令 $E^{\mathcal{G}} := \{(v, w) \in (V\mathcal{G})^2 \mid \mathsf{wt}^{\mathcal{G}}(v, w) \neq \perp\}$。

如果 $\mathsf{wt}^{\mathcal{G}}$ 是对称的，则加权图 G 是无向的；如果 $\mathsf{wt}^{\mathcal{G}}(v, v) = \perp$对所有顶点 v均成立，则它是无环的。 ⌟

对于词汇表 $\Upsilon \subseteq \Upsilon'$，$\Upsilon$-结构 A 的一个 Υ'-扩展 是一个 Υ'-结构 A′，满足$A' = A$ 且对所有 $S \in \Upsilon$均有 $S^{\mathcal{A}'} = S^{\mathcal{A}}$。如果 A′ 是 A 的扩展，那么 A 是 A′的约化结构 。

请注意，标准有限关系结构是其词汇表不包含任何权重函数符号的加权结构。我们可以将加权结构视为双类型结构：第一类型是有限关系结构，第二类型是扩展实数。我们通常假设实数上还具有额外的结构，例如线性序和算术运算。权重函数将元素从第一类型映射到第二类型。

如果加权结构 A 的所有权重函数的值域都包含于 $\mathbb{Q}_\perp$，则称其为有理数加权结构。在讨论加权结构的算法方面时，我们通常将注意力限制在有理加权结构上。我们固定一种合理的、以二进制字符串表示有理数加权结构的编码，并用 $\|\mathfrak{A}\|$ 表示 A 的编码长度。此外，A 的阶 $|\mathfrak{A}|$ 是其论域中的元素数量，即 $|\mathfrak{A}| := |A|$。注意，对于固定的词汇表Υ，编码长度 $\|\mathfrak{A}\|$ 关于阶 $|\mathfrak{A}|$ 以及权重的最大位长是多项式的。

2.2 前馈神经网络

我们假设读者至少对神经网络有一些初步了解。我们只考虑最简单的神经网络类型——前馈神经网络，也称为多层感知机。我们将神经网络视为由加权边和节点构成的有向无环图。计算发生在节点（即神经元）上。它们对其入邻居的值进行仿射线性组合，然后应用一个（通常是非线性的）激活函数。所得的值会传递给该节点的出邻居。入度为 0 的节点是输入节点；出度为 0 的节点是输出节点。节点进行的仿射线性组合由入边的权重以及节点自身的权重（称为该节点的偏置 ）决定。这里我们考虑的激活函数只有由 $\mathrm{ReLU}(x) := \max\{0, x\}$ 定义的修正线性单元 ，以及恒等函数 $\mathrm{id}(x) = x$ （仅用于输出节点）。

读者应该知道，在实践中，前馈神经网络（FNN）的权重和偏置是通过数据学习得到的，这些数据由神经网络要表示的函数的自变量—函数值对组成。这里我们不关注学习过程，只分析学得的模型。

形式上，前馈神经网络（*FNN*）是一个词汇表为$\Upsilon_{\mathrm{FNN}} := \{\mathrm{wt}, \mathrm{bias}, \leq_{\mathsf{in}}, \leq_{\mathsf{out}}\}$的加权结构 N，由一个二元权函数符号 wt、一个一元权重函数符号 bias 和两个二元关系符号 $\leq_{\mathsf{in}}$，$\leq_{\mathsf{out}}$ 组成，并满足以下条件：

- $E^{\mathcal{N}} := \big\{(v, w) \in N^2 \mid wt^{\mathcal{N}}(v, w) \neq \perp\big\}$是有向无环图 $V\mathcal{N} := N$的边集；$\mathrm{bias}^{\mathcal{N}}(v) = \perp$ 当且仅当 v 是一个输入节点时成立，也就是说，它是在有向无环图 $(V\mathcal{N}E^{\mathcal{N}})$ 中入度为 0 的节点；$\leq_{\mathsf{in}}^{\mathcal{N}}$ 是所有输入节点集合上的线性序，在其他所有节点上未定义；$\leq_{\mathsf{out}}^{\mathcal{N}}$ 是所有输出节点集合上的线性序，也就是说，它是在出度为 0 的节点上的线性序，在其他所有节点上未定义。

FNN 的输入维度 是输入节点的数量，而输出维度 是输出节点的数量。

输入维度为 m、输出维度为 n 的前馈神经网络 N 定义了一个函数 $f^{\mathcal{N}}\colon \mathbb{R}^m \to \mathbb{R}^n$。为定义该函数，我们对每个节点 $v \in V^{\mathcal{N}}$归纳定义一个函数 $f_v^{\mathcal{N}}\colon \mathbb{R}^m \to \mathbb{R}$ 。如果 v 是相对于线性序 $\leq_{\mathsf{in}}$的第 i个输入节点，则令

$f_v^{\mathcal{N}}(x_1,\dots,x_m) := x_i$。如果 v 的入邻居为 $u_1,\dots,u_k$，则令[2]

$$f_v^{\mathcal{N}}(x_1,\dots,x_m) := \mathsf{bias}^{\mathcal{N}}(v) + \sum_{i=1}^{k} \mathsf{wt}(u_i,v)\cdot \mathrm{ReLU}\left(f_{u_i}^{\mathcal{N}}(x_1,\dots,x_m)\right).$$

最后，如果 $v_1,\dots,v_n$ 是按 $\leq_{\mathsf{out}}^{\mathcal{N}}$ 所列顺序排列的 N 的输出节点，那么我们令

$$f^{\mathcal{N}}(x_1,\dots,x_m) := \left(f_{v_1}^{\mathcal{N}}(x_1,\dots,x_m),\dots,f_{v_n}^{\mathcal{N}}(x_1,\dots,x_m)\right).$$

最后需要指出的是，我们可以用类似的方式将其他更复杂的神经网络表示为加权结构。但在本文中，我们将注意力限制在前馈神经网络上。

3 带权重聚合的一阶逻辑

我们的基础逻辑是对加权结构中一阶逻辑的直接扩展。变量和量化的论域是结构的（有限）论域，而数值类型仅为项提供取值。一阶逻辑 FO 在加权结构上 包含两类句法对象，即 公式 φ 和 权重项 θ （下文简称为 项），它们由以下文法定义：

$$\varphi ::= x = y \mid R(x_1,\dots,x_{\mathrm{ar}(R)}) \mid \theta \leq \theta \mid \neg\varphi \mid (\varphi * \varphi) \mid Qx\varphi \tag{3.A}$$
$$\theta ::= 0 \mid 1 \mid F(x_1,\dots,x_{\mathrm{ar}(F)}) \mid (\theta \circ \theta). \tag{3.B}$$

这里，x、y、x_i 是变量，R 是关系符号，$*\in\{\vee、\ \wedge、\ \to\}$ 是布尔联结词，$Q\in\{\exists、\ \forall\}$ 是量词，F 是权重函数符号，而 $\circ\ \in\{+、\ -、\ \cdot\ /\}$ 是算术运算符。一阶逻辑表达式要么是公式，要么是项。表达式 ξ的词汇表是其中出现的所有关系符号和权重函数符号的集合 ξ。（注意，关系$\leq$、常量 0、1 以及算术运算 $+$、$-$、$\cdot$ / 不属于词汇表；我们将它们视为“内置”的关系、函数和常量。）我们以显然的方式定义表达式的自由变量。闭表达式是不含自由变量的表达式。闭公式也称为语句。我们用 $\xi(x_1,\dots,x_k)$ 表示表达式 ξ的自由变量包含于 $x_1、\dots、x_k$ 之中。（这些变量不必全部实际出现在表达式中。）

加权结构上的一阶逻辑语义以显而易见的方式定义。对于每个一阶逻辑表达式 $\xi(x_1、\dots、x_k)$、每个加权结构 A 以及所有元素 $a_1、\dots、a_k\in A$，我们定义一个值 $[\![\xi]\!]^{\mathcal{A}}(a_1、\dots、a_k)$。如果 ξ 是公式，那么 $[\![\xi]\!]^{\mathcal{A}}(a_1、\dots、a_k)$ 是一个布尔值（0 或 1）；如果 ξ 是项，那么 $[\![\xi]\!]^{\mathcal{A}}(a_1、\dots、a_k)\in\mathbb{R}_\perp$。如果 ξ 的词汇表不包含在 A 的词汇表中，那么我们令 $[\![\xi]\!]^{\mathcal{A}}(a_1、\dots、a_k) := 0$（若 ξ 是公式），并令 $[\![\xi]\!]^{\mathcal{A}}(a_1、\dots、a_k) := \perp$（若 ξ 是项）。否则，值 $[\![\xi]\!]^{\mathcal{A}}(a_1、\dots、a_k)$ 通过直接归纳定义，使用布尔联结词和量词的通常语义，以及第 2 节开头所说明的扩展实数运算法则 $\mathbb{R}_\perp$。

对于闭表达式 ξ，我们写作 $[\![\xi]\!]^{\mathcal{A}}$，而不写 $[\![\xi]\!]^{\mathcal{A}}()$；对于公式 $\varphi(x_1,\dots,x_k)$，我们写作 $A \models \varphi(a_1,\dots,a_k)$，而不写 $[\![\xi]\!]^{\mathcal{A}}(a_1,\dots,a_k) = 1$；如果 φ 是一个句子，也可以直接写作 $A \models \varphi$。

[2] 我们对节点所计算函数的定义方式并不常见：我们暂不应用激活函数，而只在将值输入下一个节点时应用 ReLU 激活。这样一来，我们就能确保不对输出节点应用任何激活函数，或者等价地说，应用恒等激活。我们这样做是因为，如果对所有输出值都应用 ReLU，我们的神经网络就只能计算非负函数。

注意，对于每个有理数 $q \in \mathbb{Q}_\perp$，都存在一个词汇表为空的闭项 θ_q，使得对于所有结构 A，$[\![\theta_q]\!]^{\mathscr{A}} = q$ 均成立（我们令 $\theta_\perp := 1/0$）。这意味着我们实际上可以将 $\mathbb{Q}_\perp$ 中的所有元素用作一阶逻辑表达式中的常量。容易看出，一阶逻辑项恰好表示结构的权重中系数为有理数的所有有理函数（多项式的商）。

▶ **例 3.1.** 我们可以用公式 $\mathsf{edge}(x,y) := \mathsf{wt}(x,y) \neq \perp$ 来定义加权图 G（见例 2.1）的边集 $E^{\mathscr{G}}$。这是 $\neg(\mathsf{wt}(x,y) \leq \perp \wedge \perp \leq \mathsf{wt}(x,y))$ 的缩写。[3] 那么，公式 $\mathsf{triangle}(x,y,z) := \mathsf{edge}(x,y) \wedge \mathsf{edge}(y,z) \wedge \mathsf{edge}(z,x)$ 定义了 G 中所有三角形组成的集合。

下面的公式定义了所有权重最小的三角形组成的集合。

$$\mathsf{min\text{-}wt\text{-}triangle}(x,y,z) := \mathsf{triangle}(x,y,z) \wedge \forall x' \forall y' \forall z' \big(\mathsf{triangle}(x',y',z') \\ \rightarrow \mathsf{wt}(x,y) + \mathsf{wt}(y,z) + \mathsf{wt}(z,x) \leq \mathsf{wt}(x',y') + \mathsf{wt}(y',z') + \mathsf{wt}(z',x')\big).$$

⌟

一阶逻辑在加权结构上的表达能力非常有限。例如，很容易证明，不存在一阶逻辑公式能够表达这样一个性质：某个节点的入边权重之和等于出边权重之和。为了证明这一结果，可以考虑所有边权重均为 1 的加权图。此时，这些加权图的顶点元组之间的局部同构，恰好就是其底层无权图中元组之间的局部同构，因此我们可以使用标准的 Ehrenfeucht-Fraïssé 博弈论证。同样，也可以证明不存在能够表达用于计算神经网络所需仿射线性函数的项（见第 2.2 节）。

3.1 FO(SUM)

为了得到表达能力更强的逻辑，我们加入一个聚合算子，使我们能够对可定义集合求和。我们还加入一种条件运算，它很方便，但也可以视为“语法糖”（参见例 3.6）。我们根据规则 (3.A) 和 (3.B)，并加入两条新的项构造规则，来定义 FO(SUM) 逻辑的公式和项：

$$\theta ::= \mathsf{if}\ \varphi\ \mathsf{then}\ \theta\ \mathsf{else}\ \theta \mid \sum_{(x_1,\ldots,x_k):\varphi} \theta. \tag{3.C}$$

求和算子 $\sum_{(x_1,\ \ldots x_k)\ :\ \varphi}$ 绑定变量 x_1，……，x_k。因此，$\sum_{(x_1,\ \ldots x_k)\ :\ \varphi} \theta$ 的自由变量是 φ 和 θ 中的自由变量，但不包括 x_1，……，x_k。if φ then θ_1 else θ_2 的自由变量是 φ，θ_1，θ_2 中的自由变量。

条件项 if φthen θ_1 else θ_2 的语义显而易见：如果 φ 成立，它取 θ_1 的值；否则取 θ_2 的值。为了定义求和算子的语义，假设 $\varphi = \varphi(x_1,\ldots,x_k,y_1,\ldots,y_\ell)$ 和 $\theta = \theta(x_1,\ldots,x_k,y_1,\ldots,y_\ell)$。那么对于项 $\Big(\sum_{(x_1,\ldots,x_k):\varphi} \theta\Big)(y_1,\ldots,y_\ell)$、结构 A 以及元素 $b_1,\ldots,b_\ell \in A$，我们令

$$\left[\!\!\left[\sum_{(x_1,\ldots,x_k):\varphi} \theta \right]\!\!\right]^{\mathscr{A}} (b_1,\ldots,b_\ell) := \sum_{\substack{(a_1,\ldots,a_k)\in A^k \\ \mathscr{A}\models\varphi(a_1,\ldots,a_k,b_1,\ldots,b_\ell)}} [\![\theta]\!]^{\mathscr{A}} (a_1,\ldots,a_k,b_1,\ldots,b_\ell).$$

[3] 从现在起，我们将使用类似的简写，不再对此加以说明。我们还会省略不必要的括号，以提高可读性。

我们将空集上的和定义为 0。此外，如果其中一个被加数未定义（⊥），那么整个和也未定义。

▶ **例 3.2.** 我们先再次考虑加权图。当然，例 3.1 中定义的 FO 公式 edge(x, y) 和 triangle(x, y, z) 也是 FO(SUM) 公式。于是，以下各项

$$\#\mathsf{edges} := \sum_{(x,y):\mathsf{edge}(x,y)} 1 \qquad \text{and} \qquad \#\mathsf{triangles} := \sum_{(x,y,z):\mathsf{triangle}(x,y,z)} 1$$

分别定义了边数和三角形数。 ⌟

▶ **例 3.3.** 接下来，我们转向前馈神经网络。与例 3.1 中一样，我们令 edge(x, y) := wt(x, y) $\neq \bot$为定义边关系的公式。于是，以下项

$$\#\mathsf{weights} := \sum_{(x,y):\mathsf{edge}(x,y)} 1 + \sum_{x:\mathsf{bias}(x)\neq\bot} 1$$

定义了前馈神经网络中权重的总数。 ⌟

▶ **例 3.4.** 对于每个 $d \in \mathbb{N}$，我们将构造一个 FO(SUM) 项 $\mathsf{eval}_d(x)$，用于计算前馈神经网络中深度不超过 d 的节点 x 所计算的函数。这里，前馈神经网络 N 中节点v 的深度 是从输入节点到 v 的有向无环图 ($V_{\mathcal{N}}E^{\mathcal{N}}$) 中最长路径的长度。本例中使用的记号在第 2.2 节中作了说明。

若要对神经网络求值，我们必须指定一个输入。为此，我们在词汇表中添加一个一元权重函数 inp，令 $\Upsilon_{\mathrm{FNN}^I} := \Upsilon_{\mathrm{FNN}} \cup \{\mathsf{inp}\}$。于是，带输入的<style id='7'>FNN with input 是一个 $\Upsilon_{\mathrm{FNN}^I}$-结构 $\mathbf{N}^I$，满足其 Υ_{FNN}-约化结构 N 是一个 FNN，并且 $\mathsf{inp}^{\mathcal{N}^I}(v) \neq \bot$ 当且仅当 v是 N 的输入节点。对于输入维度为m的 FNN N，假设其输入节点为 $u_1, \dots, u_m$，按 $\leq_{\mathsf{in}}$的顺序排列；给定输入 $(r_1, \dots, r_m) \in \mathbb{R}^m$，我们令 $\mathbf{N}(r_1, \dots, r_m)$ 表示通过 $\mathsf{inp}_{\mathcal{N}(r_1,\dots,r_m)}(u_i) = r_i$扩展 N 得到的带输入 FNN。

通过对 $d \geq 0$ 归纳，我们将定义一个 FO(SUM) 项 $\mathsf{eval}_d(x)$，使得对于每个$m \in \mathbb{N}$、每个输入维度为 m的 FNN N、每个节点 $v \in V_{\mathcal{N}}$以及每个 $(r_1, \dots, r_m) \in \mathbb{R}^m$，都有

$$[\![\mathsf{eval}_d]\!]^{\mathcal{N}(r_1,\dots,r_m)}(v) = \begin{cases} f_v^{\mathcal{N}}(r_1, \dots, r_m) & \text{if the depth of } v \text{ is at most } d, \\ \bot & \text{otherwise.} \end{cases}$$

我们令 $\mathsf{eval}_0(x) := \mathsf{inp}(x)$ 和

$$\mathsf{eval}_{d+1}(x) := \mathsf{if}\ \ \mathsf{inp}(x) \neq \bot\ \ \mathsf{then}\ \ \mathsf{inp}(x) \\ \mathsf{else}\ \ \mathsf{bias}(x) + \sum_{y:\mathsf{edge}(y,x)} \mathsf{wt}(y,x) \cdot \mathsf{if}\ \mathsf{eval}_d(y) \geq 0\ \mathsf{then}\ \mathsf{eval}_d(y)\ \mathsf{else}\ 0 \cdot \mathsf{eval}_d(y).$$

要理解这个项，请注意，子项 $\mathsf{if}\ \mathsf{eval}_d(y) \geq 0\ \mathsf{then}\ \mathsf{eval}_d(y)\ \mathsf{else}\ 0$ 计算的是 $\mathsf{eval}_d(y)$ 的 ReLU。在 else 子句中，我们写作 $0 \cdot \mathsf{eval}_d(y)$，而不只是 0，以确保当 $\mathsf{eval}_d(y) = \bot$ 时该项未定义。

利用项 $\mathsf{eval}_d(x)$，对于每个 $i \in \mathbb{N}$，我们都可以定义一个闭项 $\mathsf{eval}_{d,i}$，使得对于所有 $m, n \in \mathbb{N}$、所有输入维度为 m 且输出维度为 n 的前馈神经网络 N，以及所有输入 $(r_1, \dots, r_m) \in \mathbb{R}^m$，如果 N 的深度至多为 d 且 $i \leq n$，那么 $[\![\mathsf{eval}_d]\!]^{\mathcal{N}(r_1,\dots,r_m)}$就是 $f^{\mathcal{N}}(r_1, \dots, r_m) \in \mathbb{R}^n$ 的第 i 个坐标；否则 $[\![\mathsf{eval}_d]\!]^{\mathcal{N}(r_1,\dots,r_m)} = \bot$。 ⌟

前面的例子表明，我们可以在 FO(SUM) 中计算有界深度前馈神经网络。证明有界深度这一假设不可省略并不难。这可由下面的定理 3.14 得出。在讨论该定理之前，我们先来看几个例子。

▶ **例 3.5.** 求和只是聚合算子的一种。其他常见的聚合方式包括算术平均值、计数、最小值和最大值。所有这些都可以仅用求和在 FO(SUM) 中定义。

1. 对于变量元组 $\boldsymbol{x}=(x_1,\ldots,x_k)$ $\boldsymbol{y}=(y_1,\ldots,y_\ell)$ 和公式 $\varphi(\boldsymbol{x},\boldsymbol{y})$，我们定义一个项 $\mathsf{count}_{\boldsymbol{x}:\varphi}(\boldsymbol{y})$，定义如下：$\mathsf{count}^{\boldsymbol{x}:\varphi} := \sum_{\boldsymbol{x}:\varphi} 1$. 于是，对于所有结构 A 和 $\boldsymbol{b}\in A^\ell$，我们有

$$[\![\mathsf{count}_{\boldsymbol{x}:\varphi}]\!]^{\mathscr{A}}(\boldsymbol{b}) = \big|\{\boldsymbol{a}\in A^k \mid \mathscr{A}\models\varphi(\boldsymbol{a},\boldsymbol{b})\}\big|.$$

2. 对于变量元组 $\boldsymbol{x}=(x_1,\ldots,x_k)$ $\boldsymbol{y}=(y_1,\ldots,y_\ell)$、公式 $\varphi(\boldsymbol{x},\boldsymbol{y})$ 和项 $\theta(\boldsymbol{x},\boldsymbol{y})$，我们定义一个项 $\Big(\mathsf{avg}_{\boldsymbol{x}:\varphi}\theta\Big)(\boldsymbol{y})$，定义如下：$\mathsf{avg}_{\boldsymbol{x}:\varphi}\theta := \Big(\sum_{\boldsymbol{x}:\varphi}\theta\Big)/\mathsf{count}^{\boldsymbol{x}:\varphi}$.
于是，对于所有结构 A 和元素 $\boldsymbol{b}\in A^\ell$，值 $[\![\mathsf{avg}_{\boldsymbol{x}:\varphi}\theta]\!]^{\mathscr{A}}(\boldsymbol{b})$ 是所有满足 $\mathrm{A}\models\varphi(\boldsymbol{a},\boldsymbol{b})$ 的 $\boldsymbol{a}\in A^k$ 所对应的值 $[\![\theta]\!]^{\mathscr{A}}(\boldsymbol{a},\boldsymbol{b})$ 的算术平均值。空集上的算术平均值未定义；事实上，如果不存在满足 $\mathrm{A}\models\varphi(\boldsymbol{a},\boldsymbol{b})$ 的 $\boldsymbol{a}\in A^k$，那么 $[\![\mathsf{avg}_{\boldsymbol{x}:\varphi}\theta]\!]^{\mathscr{A}}(\boldsymbol{b}) = \bot$。

3. 对于变量的元组 $\boldsymbol{x}=(x_1,\ldots,x_k)$ $\boldsymbol{y}=(y_1,\ldots,y_\ell)$、公式 $\varphi(\boldsymbol{x},\boldsymbol{y})$ 和项 $\theta(\boldsymbol{x},\boldsymbol{y})$，我们令 $\varphi'(\boldsymbol{x},\boldsymbol{y}) := \varphi(\boldsymbol{x},\boldsymbol{y})\wedge\forall\boldsymbol{x}'\big(\varphi(\boldsymbol{x}',\boldsymbol{y})\to\theta(\boldsymbol{x}',\boldsymbol{y})\le\theta(\boldsymbol{x},\boldsymbol{y})\big)$.
我们通过 $\mathsf{max}_{\boldsymbol{x}:\varphi}\theta := \mathsf{avg}_{\boldsymbol{x}:\varphi'}\theta$. 定义项 $\Big(\mathsf{max}_{\boldsymbol{x}:\varphi}\theta\Big)(\boldsymbol{y})$。

那么对于所有结构 A 和 $\boldsymbol{b}\in A^\ell$，我们有

$$[\![\mathsf{max}_{\boldsymbol{x}:\varphi}\theta]\!]^{\mathscr{A}}(\boldsymbol{b}) = \max\Big\{[\![\theta]\!]^{\mathscr{A}}(\boldsymbol{a},\boldsymbol{b}) \;\Big|\; \boldsymbol{a}\in A^k \text{ such that } \mathscr{A}\models\varphi(\boldsymbol{a},\boldsymbol{b})\Big\}.$$

类似地，我们可以定义项 $\mathsf{min}_{\boldsymbol{x}:\varphi}\theta$。 ⌟

▶ **例 3.6**。 在本例中，我们将展示如何仅使用一阶逻辑和求和运算来表示条件表达式 if φ then θ_1 else θ_2。为简单起见，我们假设 φ，θ_1，θ_2 是闭项；将其推广到任意表达式是直接的。

注意，在例 3.5 中我们没有使用条件表达式，因此可以使用其中定义的聚合算子。注意，我们已有的所有结构 A

$$[\![\mathsf{count}_{x:\varphi}]\!]^{\mathscr{A}} = \begin{cases} |\mathscr{A}| & \text{if } \mathscr{A}\models\varphi,\\ 0 & \text{otherwise.}\end{cases}$$

令 $\eta := (\text{计数}_{x:\varphi})/(\text{计数}_{x:x=x})$。那么 $[\![\eta]\!]^{\mathscr{A}}=1$ 如果 $\mathrm{A}\models\varphi$ 且 $[\![\eta]\!]^{\mathscr{A}}=0$ 否则。因此，项 $\eta\cdot\theta_1+(1-\eta)\cdot\theta_2$ 等价于 if φ then θ_1 else θ_2。 ⌟

▶ **例 3.7**。 为了理解神经网络的内部工作原理，我们可以寻找那些对给定输入的计算结果没有影响的边。我们称这样的边为无用的。

令 $\mathsf{eval}_d(x)$ 为 FO(SUM) 项（来自例 3.4），它定义了深度至多为 d 的节点 x 处带输入的前馈神经网络的值。令 $\mathsf{edge}'(x,y,x_0,y_0) := \mathsf{edge}(x,y)\wedge\neg(x=x_0\wedge y=y_0)$。此公式定义了移除边 (x_0,y_0) 后所得前馈神经网络的边关系。令 $\mathsf{eval}'_d(x,x_0,y_0)$ 为从 $\mathsf{eval}_d(x)$ 得到的项，其中将每个子公式 $\mathsf{edge}(x,y)$ 替换为 $\mathsf{edge}'(x,y,x_0,y_0)$。那么对于每个输入维度为 m 的前馈神经网络 N、每个输入 $\boldsymbol{r}=(r_1,\ldots,r_m)\in\mathbb{R}^m$，以及所有满足 v 的深度至多为 d 的 $v,u_0,v_0\in V\mathrm{N}$，我们有

$$[\![\mathsf{eval}'_d]\!]^{\mathcal{N}(r_1,\ldots,r_m)}(v,u_0,v_0) = [\![\mathsf{eval}_d]\!]^{\left(\mathcal{N}-(u_0,v_0)\right)(r_1,\ldots,r_m)}(v) = f_v^{\mathcal{N}-(u_0,v_0)}(r_1,\ldots,r_m),$$

其中，$\mathrm{N}-(u_0,\ v_0)$ 是通过删除边 $(u_0,\ v_0)$，即设置 $\mathrm{wt}^{\mathcal{N}-(u_0,v_0)}(u_0,\ v_0) := \bot$，从 N 得到的前馈神经网络。

那么，公式

$$\mathsf{useless}(x_0, y_0) \coloneqq \mathsf{edge}(x_0, y_0) \wedge \forall x \big(x \leq_{\mathsf{out}} x \to \mathsf{eval}_d(x) = \mathsf{eval}_d(x, x_0, y_0)\big).$$

定义了所有无用边的集合。 ⌟

最后一个例子表明，我们也可以定义前馈神经网络的自然全局性质，而这些性质不依赖于特定输入。这个例子复杂得多（实际上，这是在 [23] 中证明的一个深刻定理），因此我们不提供任何细节。

▶ **例3.8** ([23])。 在这个例子中，我们要说明，在 FO(SUM) 中可以定义对前馈神经网络所计算的函数进行积分。我们需要固定前馈神经网络的深度 d 和输入维度 m。

我们用权重常量 $\mathsf{a}_1, \dots, \mathsf{a}_m, \mathsf{b}_1, \dots, \mathsf{b}_m$ 来指定要积分区域的边界（回想一下，权重常量是 0 元权重函数）。在这个例子中，对于一个前馈神经网络 N 以及 $\boldsymbol{a} = (a_1, \dots, a_m)$ $\boldsymbol{b} = (b_1, \dots, b_m) \in \mathbb{R}^m$，其中对所有 $i \in [m]$ 都有 $a_i \leq b_i$，我们用 $\mathsf{N}(\boldsymbol{a}, \boldsymbol{b})$ 表示 N 在词汇表 $\Upsilon_{\mathrm{FNN}} \cup \{\mathsf{a}_1, \dots, \mathsf{a}_m, \mathsf{b}_1, \dots, \mathsf{b}_m\}$ 上的扩展，其中 $\mathsf{a}_i^{\mathcal{N}(\boldsymbol{a},\boldsymbol{b})} = a_i$ 和 $\mathsf{b}_i^{\mathcal{N}(\boldsymbol{a},\boldsymbol{b})} = b_i$。

于是存在一个闭合 FO(SUM) 项 $\mathsf{Integrate}_{d,\,m}$ 使得对于所有输入维度为 m、输出维度为 1 且深度至多为 d 的 FNN，以及所有 $\boldsymbol{a} = (a_1, \dots, a_m)$ $\boldsymbol{b} = (b_1, \dots, b_m) \in \mathbb{R}^m$，满足 $a_i \leq b_i$，我们都有

$$[\![\mathsf{integrate}_{d,m}]\!]^{\mathcal{N}(\boldsymbol{a},\boldsymbol{b})} = \int_{a_1}^{b_1} \cdots \int_{a_m}^{b_m} f^{\mathcal{N}}(x_1, \dots, x_m) dx_1 \dots dx_m.$$

证明很困难。它基于 FNN 所计算的（分段线性）函数的柱状单元分解的 FO(SUM) 可定义性。为了至少了解如何定义积分，我们考虑特殊情形 $d = 2$，$m = 1$。也就是说，我们考察具有一个输入节点、一个隐藏层和一个输出节点的 FNN N。这样的 FNN 计算一个分段线性函数 $f^{\mathcal{N}} \colon \mathbb{R} \to \mathbb{R}$。该函数的断点由 N 隐藏层中的节点决定，利用这些节点，我们可以定义所有断点的坐标。此外，我们还可以定义连接各断点的线性片段的斜率。由此，积分便很容易计算。」

3.2 复杂性

在本节中，我们假设所有加权结构都是有理数（见第 2.1 节）。众所周知，在（无权）有限关系结构上对一阶逻辑公式求值属于一致 AC^0，复杂性类，即由 dlogtime 一致的布尔电路族所判定的问题类，其中电路具有 $\neg$、$\wedge$、$\vee$ 无界扇入、有界深度和多项式规模（见 [26]）。

要对有理数加权结构上的 FO 公式和项求值，我们需要对有理数上的项求值。众所周知，这可以在一致 TC^0，复杂性类中完成；该类的定义类似于 AC^0，但除了标准逻辑门 [5, 25]之外，还允许电路中使用阈值门。由于迭代加法（对一族位长为 n的 n 数进行加法）属于一致 TC^0，，因此我们实际上可以完成对 FO(SUM) 公式和项求值所需的所有算术运算，从而得到以下定理。[4]

[4] 很难追溯关于算术在一致 TC^0中的结果。20世纪80年代（可追溯至 [10]）已证明，算术属于非一致 TC^0，，但对于加法、减法、迭代加法以及

▶**定理 3.9**。 FO(SUM) 的数据复杂性属于一致 TC^0。

也就是说，对于每个封闭的 FO(SUM) 表达式 ξ，在给定结构$\mathcal{A}$ 中的值 $[\![\xi]\!]^{\mathcal{A}}$ *of* ξ，都可以通过一个对数时间一致、深度有界且规模为多项式的阈值电路族计算得出。

3.3 模型无关查询

通常，我们只关心神经网络计算出的函数，而不关心网络的结构。请注意，对于前馈神经网络可计算的每个函数，都存在无穷多个截然不同的前馈神经网络计算该函数。我们将结果仅取决于函数而不取决于具体前馈神经网络的查询称为模型无关查询。

如果对于所有 $\mathbf{N}, \mathbf{N}' \in \mathbf{N}$，只要 $f^{\mathcal{N}} = f^{\mathcal{N}'}$，就有 $[\![\xi]\!]^{\mathcal{N}} = [\![\xi]\!]^{\mathcal{N}'}$，那么封闭的 FO(SUM) 表达式 ξ 就称为在一类前馈神经网络$\mathbf{N}$上模型无关。我们可以将这个定义扩展到加权结构上的其他逻辑，特别是将在第 4 节讨论的逻辑 IFP(SUM)。我们甚至可以将定义推广到$\mathbf{N}$上的抽象布尔查询，也就是说，推广到同构不变函数 $\Phi\colon \mathbf{N} \to \{0,1\}$，以及同构不变函数 $\Theta\colon \mathbf{N}\{\text{v30}>$。从另一个角度看，我们还可以将其推广到前馈神经网络扩展类，例如带输入的前馈神经网络（见例 3.4）以及带额外权重常数的前馈神经网络（见例3.8）。但请注意，不变性条件 $f_{\mathcal{N}} = f^{\mathcal{N}'}$始终指的是普通神经网络。例如，对于带输入的神经网络 $\mathbf{N}(\boldsymbol{r}), \mathbf{N}'(\boldsymbol{r}')$，我们仍然要求 $f^{\mathcal{N}} = f^{\mathcal{N}'}$，而不是$f_{\mathcal{N}}(\boldsymbol{r}) = f^{\mathcal{N}'}(\boldsymbol{r}')$。

我们将输入维度为 m、输出维度为 n 的所有前馈神经网络组成的类记为 $\mathbf{N}(m,n)$。此外，我们令 $\mathbf{N}(m,*) := \bigcup_{n\geq 1} \mathbf{N}(m,n)$，并类似地定义 $\mathbf{N}(*,n)$ 和 $\mathbf{N}(*,*)$。注意，$\mathbf{N}(*,*)$ 是所有前馈神经网络组成的类。我们将深度至多为 d 的所有前馈神经网络组成的类记为 $\mathbf{N}_d(*,*)$；对于 $m,n \in \mathbb{N}_{>0} \cup \{*\}$，令 $\mathbf{N}_d(m,n) := \mathbf{N}(m,n) \cap \mathbf{N}_d(*,*)$。此外，我们将带输入的相应前馈神经网络类记为 $\mathbf{N}^I(m,n)$ 和 $\mathbf{N}^I_d(m,n)$。

如果一个表达式、查询或函数完全模型无关，即它在所有前馈神经网络或适当形式的前馈神经网络扩展所构成的类上模型无关，我们就称其为完全模型无关。我们有意不明确说明“适当形式”，也不具体界定结构类，以便以最自然的形式表述结果。对于带输入的前馈神经网络，我们通常考虑类$\mathbf{N}(*,*)$ 或 $\mathbf{N}(*,1)$，或$\mathbf{N}^I(*,*)$ 或$\mathbf{N}^I(*,1)$。

▶ **例 3.10**。 例 3.4 中的 FO(SUM) 项 $\mathrm{eval}_{d,\,i}$ 在 $\mathbf{N}^I_d(*,\ *)$ 上是模型无关的，但在 $\mathbf{N}^I_{d+1}(1,1)$ 上不是模型无关的，因此并非完全模型无关。

不过，对于每个 $i \geq 1$，求值函数 $\mathrm{Eval}_i\colon \mathbf{N}^I(*,\ *) \to \mathbb{R}$，定义为令 $\mathrm{Eval}_i(\mathbf{N}(\boldsymbol{r}))$ 等于向量 $f^{\mathcal{N}}(\boldsymbol{r})$ 的第 i 个元素，是完全模型无关的。下文中，我们通常只关注具有单一输出的前馈神经网络，并将其求值函数记为 Eval，而不是 Eval_1。

▶ **例 3.11.** 例 3.3 中的权重数并非与模型无关，因为计算同一函数的前馈神经网络可能具有不同的权重。同样，句子“权重数 $\leq$10.000”也不是模型无关的。

乘法方面，人们已知这些构造实际上可以做到一致。当然，在 Barrington、Immerman 和 Straubing 研究逻辑和一致 TC^0 [5] 时，这一点就已为人所知。不过，我不知道有哪篇文献证明了这些结果。迭代加法至关重要；因此，我在最近的论文 [21] 中补充了迭代加法属于一致 TC^0 的证明。最困难的情况是除法；Hesse 直到 2000 [24, 25] 才证明除法属于一致 TC^0。

然而，对于每个函数 $f\colon \mathbb{R} \to \mathbb{R}$ 查询“是否存在一个权重数至多为 10.000、能够计算同一函数的前馈神经网络 $\mathbf{N}'$ ？”，也就是说，$\Phi\colon \mathbf{N}(*,*) \to \{0,1\}$ 定义为

$$\Phi(\mathcal{N}) \coloneqq \begin{cases} 1 & \text{there exists an } \mathcal{N}' \in \mathbf{N}(*,*) \text{ such that } [\![\#\mathsf{weights}]\!]^{\mathcal{N}'} \leq 10.000 \\ & \qquad \text{and } f^{\mathcal{N}'} = f^{\mathcal{N}}, \\ 0 & \text{otherwise} \end{cases}$$

是完全与模型无关的。 ⌟

▶ **例3.12**。例3.8中的 FO(SUM) 项 $\mathsf{integrate}_{d,\ m}$ 在由权重常量扩展的适当 $\mathbf{NN}_d(m,\ 1)$ 类上与模型无关，但并非完全与模型无关。

同样，我们可以在由适当权重常量扩展的所有输出维度为 1 的前馈神经网络类上，定义一个完全与模型无关的积分函数 Integrate。

我们还可以考虑 Integrate 查询的特殊情形 $\mathsf{Integrate}^{[0,1]}$，其中将所有积分边界 a_i 设为 0，并将所有 b_i 设为 1。这样便无需为积分边界添加显式的权重常量，我们可以将 $\mathsf{Integrate}^{[0,1]}$ 视为 $\mathbf{N}(*,\ 1)$ 上一个完全与模型无关的函数。⌟

▶ **例3.13**。由下式定义的 $\mathbf{N}(*,\ *)$ 上的布尔查询 Zero

$$\mathsf{Zero}(\mathcal{N}) \coloneqq \begin{cases} 1 & \text{if } f^{\mathcal{N}}(\boldsymbol{r}) = 0 \text{ for all } \boldsymbol{r} \in \mathbb{R}^m, \text{ where } m \text{ is the input dimension of } \mathcal{N}, \\ 0 & \text{otherwise} \end{cases}$$

完全与模型无关。

[22] 已证明，对于有理数前馈神经网络 $\mathbf{N} \in \mathbf{N}(1,1)$，判定 $\mathsf{Zero}(\mathbf{N}) = 0$ 是否成立是 NP完全的。（Wurm [34] 更早证明了，对$\mathbf{N}(*,1)$进行相同的查询也是 NP 完全的。）因此，NonZero 查询是 NP 完全的，这并不令人意外，因为它可以被视为前馈神经网络的某种可满足性查询。需要指出的是，[22] 的证明表明，对于$\mathbf{N}(1,1)$上的 $\mathsf{NonZero}^{[0,1]}$ 查询，询问是否存在一个输入 $r \in [0,1]$，使得对于某个 $\mathbf{N} \in \mathbf{N}(1,1)$，$f^{\mathcal{N}}(r) \neq 0$，该问题已经是 NP 困难的。这也意味着，判定函数 $\mathsf{Integrate}^{[0,1]}$（见例 3.12）是否非零是 NP 困难的。⌟

我们希望了解哪些完全与模型无关的查询可以用 FO(SUM) 表达。答案令人失望：只有平凡的常量查询可以表达。

▶ **定理 3.14** ([23])。设 m、$n \in \mathbb{N}_{>0}$，且 φ 是一个在 $\mathbf{N}(m,n)$ 上与模型无关的 FO(SUM) 句子。那么，对于所有 $\mathcal{N} \in \mathbf{N}(m,n)$，$\mathcal{N} \models \varphi$；或者，对于所有$\mathcal{N} \in \mathbf{N}(m,n)$，$\mathcal{N} \not\models \varphi$。

对于带有输入的前馈神经网络类$\mathbf{N}^I(m,n)$，也有类似结果。因此，求值函数 Eval（见例 3.10）无法用 FO(SUM) 表达。

定理 3.14 的证明很简单。直观地说，它基于这样一个事实：FO(SUM) 只能表达局部性质，但我们可以用由权重为 1 的边构成的长路径扩展前馈神经网络中的边，而不改变所计算的函数。

然而，FO(SUM) 能够在有界深度前馈神经网络类上表达有趣的模型无关查询和函数（参见例 3.10 和例 3.12）。作为衡量模型无关查询表达能力的基准，我们采用一种只能“黑盒”访问前馈神经网络所计算函数、却无法访问前馈神经网络本身的逻辑。我们考虑的逻辑是实数有序域上的一阶逻辑，并扩展了前馈神经网络所计算的函数。对于一个 m 元函数符号 f，我们令 FO(R f) 表示语言$\{+,\cdot 0,1,\leq,f\}$中的一阶逻辑，其中 $+,\cdot$ 是二元函数符号，$0,1$ 是常元符号，而 $\leq$

$\widehat{\ }$是一个二元关系符号。我们通过一个 m 元函数 $f\colon \mathbb{R}^m \to \mathbb{R}$，在实数有序域 $\mathbf{R} = (\mathbb{R}, +, \cdot 0, 1, \leq)$ 的扩展 $(\mathbf{R}, f)$ 上解释 $\mathrm{FO}(\mathbf{R}, f)$ 公式。每个 $\mathrm{FO}(\mathbf{R}, f)$ 句子 φ 都定义了一个关于 $\mathbf{N}(m, 1)$ 的查询，其定义如下：

$$\varphi(\mathcal{N}) = 1 \iff (\mathscr{R}, f^{\mathcal{N}}) \models \varphi.$$

我们对使用同一个符号 φ 表示查询及定义该查询的句子这一点，不应产生任何困惑。显然，该查询在$\mathbf{N}(m,1)$上是不变的。请注意，由于函数符号具有固定元数，要在这种逻辑中进行处理，我们需要固定输入维度和输出维度。输出维度为 1 并非必要。如果我们想查询输出维度为（固定值） n的前馈神经网络，只需添加 n 个函数符号，而不是仅添加一个。

我们也可以定义从前馈神经网络到实数的函数。每个 $\mathrm{FO}(\mathbf{R}, f)$ 公式 $\varphi(y)$ 都定义了一个从$\mathbf{N}(m,1)$到 $\mathbb{R}_\perp$ 的函数，其定义如下：

$$\varphi(\mathcal{N}) = \begin{cases} s & \text{if } (\mathscr{R}, f^{\mathcal{N}}) \models \varphi(s) \text{ and there is no } s' \neq s \text{ such that } (\mathscr{R}, f^{\mathcal{N}}) \models \varphi(s'), \\ \perp & \text{otherwise.} \end{cases}$$

为了定义带有输入的前馈神经网络上的查询，我们使用含有 m 个自由变量的公式 $\varphi(x_1, \ldots, x_m)$。每个这样的公式都定义了$\mathbf{N}^I(m, 1)$上的一个查询，其定义如下：

$$\varphi(\mathcal{N}, r_1, \ldots, r_m) = 1 \quad :\Longleftrightarrow \quad (\mathscr{R}, f^{\mathcal{N}}) \models \varphi(r_1, \ldots, r_m).$$

类似地，我们可以使用含有 ℓ 个自由变量的 $\mathrm{FO}(\mathbf{R}, f)$ 公式，定义由任意固定数量ℓ 个权重常数扩展的$\mathbf{N}(m,1)$中的前馈神经网络上的查询；当然，我们也可以使用一个额外的自由变量来定义函数。

我们还考虑 $\mathrm{FO}(\mathbf{R}_{\mathrm{lin}}, f)$ 的线性片段，它由语言 $\{+, (q)_{q\in\mathbb{Q}}, \leq\}$中的所有公式组成，其中为每个有理数都添加了一个常数符号。我们将这些公式解释为 $\mathbf{R}$ 上扩展结构 $(\mathbf{R}_{\mathrm{lin}}, f)$ 的 $\mathrm{FO}(\mathbf{R}, f)$ 公式。φ 在这种语言中包含有理数常量很有用。由于每个有理数都可以在 $\mathbf{R}$ 中定义，因此逻辑 $\mathrm{FO}(\mathbf{R}, f)$ 中不需要这些额外的常量。$\mathrm{FO}(\mathbf{R}_{\mathrm{lin}}, f)$ 的公式以与 $\mathrm{FO}(\mathbf{R}, f)$ 的公式相同的方式定义关于前馈神经网络的查询。

▶ **例 3.15.** 前馈神经网络 $\mathbf{N}^I(m, 1)$ 上的求值查询 Eval（见例 3.10）可由 $\mathrm{FO}(\mathbf{R}_{\mathrm{lin}}, f)$ 中的公式 $\varphi(x_1, \ldots, x_m, y) := f(x_1, \ldots, x_m) = y$ 表达。

▶ **例3.16**。 在 $\mathbf{N}(m, 1)$ 上的查询 Zero（参见例3.13）可由一阶逻辑$(\mathbf{R}_{\mathrm{lin}}, f)$中的句子 $\varphi := \forall x_1 \ldots \forall x_m f(x_1, \ldots, x_m) = 0$表示。

▶ **例3.17**。 我们可能想了解前馈神经网络计算的函数在多大程度上依赖于某个特定输入特征。对于一个 $\varepsilon > 0$，如果坐标 $i \in [m]$ ε-无关于函数 $f\colon \mathbb{R}^m \to \mathbb{R}$，即对所有 $r_1, \ldots, r_m, r_i' \in \mathbb{R}$，都满足

$$|f(r_1, \ldots, r_m) - f(r_1, \ldots, r_{i-1}, r_i', r_{i+1}, \ldots, r_m)| \leq \varepsilon.$$

对于每个（有理数） $\varepsilon > 0$，我们都可以轻松地定义一个一阶逻辑$(\mathbf{R}_{\mathrm{lin}}, f)$公式，以表达 i 对 f 而言是 ε-无关的。

因此，在 $\mathbf{N}(m, 1)$ 上由 $\mathrm{Irr}_{\varepsilon,i}(\mathbf{N}) = 1$ 当且仅当 i 对 $f^{\mathcal{N}}$ 而言是 ε-无关定义的查询 $\mathrm{Irr}_{\varepsilon,i}$，可由一阶逻辑$(\mathbf{R}_{\mathrm{lin}}, f)$表示。 ⌟

根据定理 3.14，查询 Zero 和 $\mathrm{Irr}_{\varepsilon,\,i}$ on $\mathbf{N}(1,\ 1)$ 无法用 FO(SUM) 表达。甚至不容易看出如何在有界深度前馈神经网络上表达这些查询。不过，以下定理表明，对于所有 d，m 它们都能在 $\mathbf{N}_d(m,\ 1)$ 上用 FO(SUM)可表达。

▶ **定理 3.18** ([23])。 设 $d \in \mathbb{N}_{>0}$。那么，在 $\mathbf{N}_d(m,1)$ 上可由一阶逻辑$(\mathscr{R}_{\text{lin}}, f)$ 表达的每个查询，都可由 FO(SUM) 表达。

事实上，该定理也适用于带输入的前馈神经网络，以及任意通过权重函数进行的扩展。证明较为困难。问题在于，一阶逻辑$(\mathbf{R}_{\text{lin}}, f)$可以对任意实数进行量化，而 FO(SUM) 公式只能访问前馈神经网络中的有限权重集。为了解决这个问题，我们在 FO(SUM) 中定义前馈神经网络所计算的分段线性函数的柱状单元分解。由于函数在每个单元内都是线性的，因此可以统一处理每个单元。随后，我们便能通过对这些单元进行量化来模拟对实数的量化。

▶ **例 3.19**。 让我们再次考察例 3.13、3.16 中的零查询。由定理 3.18 可知，对于每个固定的 m 和 d，Zero 和 NonZero 在$\mathbf{N}_d(m,1)$ 上的限制都可以用 FO(SUM) 表示。因此，根据定理 3.9，这些查询可以在统一的 TC^0 中求值。这一点令人瞩目，因为正如前文所述，NonZero 是一种可满足性问题，而布尔电路的可满足性对于深度为 3 的电路已经是困难问题。」

我们以一个例子结束本节，该例表明，对于有界深度前馈神经网络，存在可用 FO(SUM) 表达、但无法用 FO(R f) 表达的与模型无关查询。

▶ **例 3.20** ([23])。 查询 $\int_0^1 f^{\mathcal{N}}(x)dx = 0$ 在 $\mathbf{N}_2(1,\ 1)$ 上可用 FO(SUM) 表达（见例3.8），但无法用 FO(R f) 表达。这是因为每个连续分段线性函数 $f\colon \mathbb{R} \to \mathbb{R}$ 都可以由深度为 2 的前馈神经网络计算，并且 [23,定理 6.1] 指出，对连续分段线性函数 $f\colon \mathbb{R} \to \mathbb{R}$ 进行积分无法用 FO(R f) 表达。该证明基于一个（深刻的）事实：如果一个集合 $X \subseteq \mathbb{R}$ 具有偶数基数，则无法用 FO(R X) 表达（见 [27, 29]）；此外，还基于从偶数基数到积分的一个（直接的）约化。」

4 带权重聚合的定点逻辑

虽然我们可以认为，在实践中，我们通常使用固定的神经网络架构，只调整权重，但即使无法在 FO(SUM) 中表达有界深度的前馈神经网络的求值函数，也会严重限制其作为前馈神经网络查询语言的适用性。

4.1 IFP(SUM)

要对深度无界的前馈神经网络进行求值，我们需要某种形式的递归。不动点逻辑是一阶逻辑框架中形式化归纳可定义性的经典机制（见 [13, 30]）。我们按照 [22] 定义逻辑 IFP(SUM)，即带权重聚合的膨胀定点逻辑。该逻辑与 [18] 中针对 $\mathbb{R}$-代数的函数定点逻辑密切相关。IFP(SUM) 的公式和权重项遵循与 FO(SUM) 相同的规则，即 (3.A)、(3.B) 和 (3.C)，并增加一条项构造规则：

$$\theta ::= \mathsf{ifp}\big(F(x_1,\dots,x_{\mathrm{ar}(F)}) \leftarrow \theta\big)(x'_1,\dots,x'_{\mathrm{ar}(F)}), \tag{4.A}$$

其中 x_i、x'_i 是变量，F 是权函数符号。请注意，元组（x_1、……、$x_{\mathrm{ar}(F)}$）和（x'_1、……、$x'_{\mathrm{ar}(F)}$）不一定互不相交；事实上，我们通常会令 $x_i = x'_i$ 对所有 i 都成立。

(4.A) 中的 ifp 算子绑定变量 x_1、……、$x_{\mathrm{ar}(F)}$。因此，不动点项 $\mathsf{ifp}\big(F(\boldsymbol{x}) \leftarrow \theta\big)(\boldsymbol{x}')$ 的自由变量是 θ 中除去 $\boldsymbol{x}$ 中的变量，再加上

$\boldsymbol{x}'$中的变量。ifp 运算符还绑定权函数符号 F；我们称 F 是该不动点项的 内涵符号 。θ 中出现的所有其他关系符号和权函数符号都是该不动点项的 外延符号 。

IFP(SUM) 忠实地扩展了 FO(SUM)，因此，要定义 IFP(SUM) 的语义，我们只需定义 ifp 算子的语义。考虑一个项

$$\eta(\boldsymbol{x}', \boldsymbol{y}) \coloneqq \mathsf{ifp}\big(F(\boldsymbol{x}) \leftarrow \theta\big)(\boldsymbol{x}'), \tag{4.B}$$

其中 $\theta = \theta(\boldsymbol{x}, \boldsymbol{y})$ 是词汇表 $\Upsilon \cup \{F\}$ 中的一个项。因此，η 的所有外延符号都属于 Υ。设 $k \coloneqq |\boldsymbol{x}| = |\boldsymbol{x}'| = \mathsf{ar}(F)$，且 $\ell \coloneqq |\boldsymbol{y}|$。设 $\mathfrak{A}$ 是一个 Υ-结构，且 $\boldsymbol{b} \in A^\ell$。对于每个函数 $\widehat{F}\colon A^k \to \mathbb{R}_\perp$，令 $(\mathfrak{A}, \widehat{F})$ 为 $\mathfrak{A}$ 的 $\Upsilon \cup \{F\}$-扩展，其中 $F^{(\mathfrak{A}, \widehat{F})} = \widehat{F}$。我们定义一个由函数 $F^{(i)}\colon A^k \to \mathbb{R}_\perp$ 构成的序列 $(F^{(i)})_{i \geq 0}$，如下所示：

- $F^{(0)}$ 对于所有 $\boldsymbol{a} \in A^k$ 满足 $F^{(0)}(\boldsymbol{a}) \coloneqq \perp$；
- $F^{(i+1)}(\boldsymbol{a}) \coloneqq \begin{cases} [\![\theta]\!]^{(\mathfrak{A}, F^{(i)})}(\boldsymbol{a}, \boldsymbol{b}) & \text{if } F^{(i)}(\boldsymbol{a}) = \perp, \\ F^{(i)}(\boldsymbol{a}) & \text{if } F^{(i)}(\boldsymbol{a}) \neq \perp. \end{cases}$

由于一旦定义了 $F^{(i)}(\boldsymbol{a}) (\neq \perp)$，我们就不会再改变其值，因此存在一个 $i < |A|^k$，使得 $F^{(i)} = F^{(i+1)}$。令 i_∞ 为满足条件的最小 i。那么对于所有 $i \geq i_\infty$，$F^{(i)} = F^{(i_\infty)}$。

最后，(4.B) 中项 $\eta(\boldsymbol{x}', \boldsymbol{y})$ 的值定义为

$$[\![\eta]\!]^{\mathfrak{A}}(\boldsymbol{a}', \boldsymbol{b}) \coloneqq F^{(i_\infty)}(\boldsymbol{a}').$$

▶**例 4.1.** 求值函数 $\mathrm{Eval}\colon \mathbf{N}^I(*,*) \to \mathbb{R}$（参见例 (3.10)）可以用 IFP(SUM) 表达。我们首先定义一个项，用于求出 FNN 每个节点所计算的函数：

$$\begin{aligned} \mathsf{eval\text{-}node}(x) \coloneqq \mathsf{ifp}\Big(F(x) \leftarrow \mathsf{if}\ \ \mathsf{inp}(x) \neq \perp\ \ \mathsf{then}\ \ \mathsf{inp}(x) \\ \mathsf{else}\ \ \mathsf{bias}(x) + \sum_{y : \mathsf{edge}(y,x)} \mathsf{wt}(y,x) \cdot \mathsf{if}\ F(y) \geq 0\, \mathsf{then}\, F(y)\, \mathsf{else}\, 0 \cdot F(y)\Big)(x) \end{aligned}$$

注意，这个项如何在逻辑内部形式化了例 3.4 中项 $\mathrm{eval}_a(x)$ 的（元层次）归纳定义。容易看出，对于每个 $m \in \mathbb{N}_{>0}$、每个 FNN $\mathrm{N} \in \mathbf{N}(m,*)$、每个输入 $\boldsymbol{r} \in \mathbb{R}^m$ 以及每个节点 v，都有 $[\![\mathsf{eval\text{-}node}]\!]^{\mathcal{N}(\boldsymbol{r})}(v) = f_v^{\mathcal{N}}(\boldsymbol{r})$。

现在令 $\mathrm{eval} \coloneqq \mathrm{avg}_{x : x \leq_{\mathsf{out}} x} \mathsf{eval\text{-}node}(x)$。那么，对于每个 FNN $\mathrm{N} \in \mathbf{N}(m,1)$ 和每个输入 $\boldsymbol{r} \in \mathbb{R}^m$，都有 $[\![\mathrm{eval}]\!]^{\mathcal{N}(\boldsymbol{r})} = f^{\mathcal{N}}(\boldsymbol{r})$。⌟

IFP(SUM) 的定义允许嵌套不动点运算符（(4.A) 中的项 θ 可能包含更多不动点运算符），但下述定理表明，这些运算符并非必需。选择条件 是变量之间等式和不等式的合取。

▶定理 *4.2*（[22]）。*1.* 每个 *IFP(SUM)* 公式 $\varphi(\boldsymbol{x})$ 都等价于如下形式的公式：$\exists \boldsymbol{y}\big(\chi(\boldsymbol{y}) \wedge \mathit{ifp}\big(F(\boldsymbol{x}, \boldsymbol{y}) \leftarrow \theta\big)(\boldsymbol{x}, \boldsymbol{y})\big)$，其中 χ 是一个选择条件，而 $\theta(\boldsymbol{x}, \boldsymbol{y})$ 是一个 *FO(SUM)* 项。*2.* 每个 *IFP(SUM)* 项 $\eta(\boldsymbol{x})$ 都等价于如下形式的项：$\mathit{avg}_{\boldsymbol{y} : \chi(\boldsymbol{y})} \mathit{ifp}\big(F(\boldsymbol{x}, \boldsymbol{y}) \leftarrow \theta\big)(\boldsymbol{x}, \boldsymbol{y})$，其中 χ 同样是一个选择条件，而 $\theta(\boldsymbol{x}, \boldsymbol{y})$ 是一个 *FO(SUM)* 项。此外，$\mathit{avg}_{\boldsymbol{y} : \chi(\boldsymbol{y})}$ 是例 *3.5* 中定义的算子。

该定理使用有限模型论中的标准技术证明（参见 [13,第 8]章）。

也可以用单个不动点算子模拟同步归纳。在 [22], 中，这被用于为 IFP(SUM) 提供一种 Datalog 风格语法。

4.2 复杂性

（最小或扩张）不动点逻辑在有限结构上的一个重要性质是，查询可以在多项式时间内求值；并且在有序有限结构上，所有多项式时间查询都可以用该逻辑表达。我们称不动点逻辑在有序有限结构上<殖 id='1'>刻画多项式时间。同样，可以证明 [18] 在有序加权结构上，IFP(SUM) 在实数上的 Blum-Shub-Smale 计算模型中刻画多项式时间 [7, 6]。然而，我们在此关注的是传统位复杂性，而下面的例子表明，在这一模型中，即使在有理加权结构上，IFP(SUM) 项也无法在多项式时间内求值，因为它们可能变得过大。

▶**例 4.3** ([17])。设

$$\sigma(x) \coloneqq \mathsf{ifp}\Big(F(x) \leftarrow \mathsf{if}\ \exists y\, \mathsf{edge}(y,x)\ \mathsf{then}\ \Big(\sum_{y:\mathsf{edge}(y,x)} F(y)\Big)\cdot\Big(\sum_{y:\mathsf{edge}(y,x)} F(y)\Big)\ \mathsf{else}\ 2\Big)(x).$$

如果我们在一个 FNN N $\in$ N(1,1) 上计算该项，而这个网络恰好是一条长度为 d 的路径，那么无论权重如何，对于 N 的输出节点 v，都有 $[\![\sigma]\!]^{\mathcal{N}}(v) = 2^{2^d}$。」

为了解决这个例子所揭示的重复平方问题，我们在 [22] 中引入了 IFP(SUM) 的标量片段 sIFP(SUM)，通过限制内涵权重函数符号在乘法和除法中的使用：在一个 sIFP(SUM) 表达式 ξ 中，对于每个形如 $\theta_1 \cdot \theta_2$ 的子项，θ_1 或 θ_2 不包含内涵符号；对于每个形如 θ_1/θ_2 的子项，项 θ_2 不包含内涵符号。（精确定义稍显棘手，因为我们需要说明在特定上下文中哪些符号算作内涵符号。详情请参阅 [22] 。）

▶ **例 4.4.** 例 4.1 中的项 eval 属于 sIFP(SUM)。例 4.3 中的项 $\sigma(x)$ 则不属于。」

下面的定理表明，在有理数加权结构上，sIFP(SUM) 表达式可以在多项式时间内求值。

▶**定理 4.5** ([22])。 sIFP(SUM) 的数据复杂性属于P。

需要指出的是，这个定理的证明并不像看起来那么简单。难点在于，如何界定表达式求值过程中出现的分数的分母大小。

与 FO(SUM) 一样，现在我们可以询问，在前馈神经网络上，哪些与模型无关的查询可以用 IFP(SUM) 或 sIFP(SUM) 表达。

▶ **例 4.6.** 如果 P $\neq$NP，那么零查询和积分函数 Integrate$^{[0,1]}$就不能用 sIFP(SUM) 表达。原因是 NonZero 和 Integrate$_{[0,1]} \neq 0$ 都是 NP 困难的（见例 3.13）。」

虽然前一个例子中不可表达的原因是计算复杂度很高，但事实证明，我们甚至无法表达所有可在多项式时间内计算的查询，哪怕是在 IFP(SUM) 中也是如此。

▶ **定理 4.7** ([22])。 存在一个关于$\mathbf{N}(1,1)$ 的与模型无关的查询，可以在多项式时间内判定，但即使在所有权重均为1 或0的前馈神经网络上，也无法用IFP(SUM) 表达。

为了证明这个定理，我们利用了这样一个事实：IFP(SUM) 在加权结构的数值部分执行计算的能力有限，因为它没有取值范围为数字的变量，因此也无法对数字进行量化。

因此，IFP(SUM) 在执行涉及位大小较大的有理数的计算时能力有限。事实证明，这是多项式时间可计算查询面临的主要限制。在 [22], 中，我们为前馈神经网络定义了一种范式，即约简前馈神经网络，它们通过一种类似于双模拟的等价关系因式分解得到。每个前馈神经网络 N 都等价于$\widetilde{\ }$一个唯一的约简前馈神经网络 N。对于每个 $b \in \mathbb{N}$，我们令$\mathbf{R}_b(*, *)$ 表示所有满足以下条件的前馈神经网络 N 的类：其约简前馈神经网络 N 的所有权重的位大小都至多为 b。

▶ **定理 4.8** ([22])。 对于每个 $b \in \mathbb{N}$，每个多项式时间可判定的模型无关 c 在$\mathbf{R}_b(*, *)$ 上的查询都可以用sIFP(SUM)表示。

5 总结

我们研究了加权结构上的两种逻辑 FO(SUM) 和 IFP(SUM)，主要考察它们表达关于前馈神经网络的查询的能力。尽管关于这些逻辑的许多重要结果已在 [22, 23], 中得到证明，但仍有许多有趣的问题悬而未决。

一个核心结果是定理 3.18，它指出 FO(SUM) 可以在有界深度的前馈神经网络上模拟 FO($\mathrm{R}_{\mathrm{lin}}$， f)。我们能否将这个定理从 $\mathrm{R}_{\mathrm{lin}}$ 推广到 R？有没有办法将这些结果推广到输入维度任意的前馈神经网络？为此，我们必须调整 FO($\mathrm{R}_{\mathrm{lin}}$， f) 的定义，因为按照目前的定义，我们能够考虑的前馈神经网络的输入维度就是固定函数符号 f 的元数。

我们最初希望，如果用 IFP(SUM) 替代 FO(SUM)，或许就能去掉定理 3.18 中的有界深度假设。鉴于 NonZero 这类简单的、可用 FO($\mathrm{R}_{\mathrm{lin}}$， f) 表达的查询具有 NP 完全性，这似乎不太可能。但我们能否在不作复杂性理论假设的情况下，证明 NonZero 无法用 sIFP(SUM) 或 IFP(SUM) 表达？

我们已经证明，其他聚合算子（计数、算术平均值、最小值和最大值）都可以仅用求和来表达。一般而言，聚合函数可以定义为从实数多重集到实数的函数（见 [17]）。探究不同聚合算子之间的相互定义能力将会很有意思。例如，似乎不太可能用算术平均值来定义求和。

另一方面，是否存在一种查询语言，能够精确表达那些可在多项式时间内计算的、与模型无关的前馈神经网络查询？

参考文献

1 S. Abiteboul、R. Hull 和 V. Vianu。数据库基础。Addison-Wesley，1995。2 Marcelo Arenas、Daniel Báez、Pablo Barceló、Jorge Pérez 和 Bernardo Subercaseaux。用于模型可解释性的符号语言基础。载于 Marc'Aurelio Ranzato、Alina Beygelzimer、Yann N. Dauphin、Percy Liang 和 Jennifer Wortman Vaughan 编，神经信息处理系统进展 *34*：*2021*年神经信息处理系统年会，*NeurIPS 2021*，*2021*年*12*月*6*日至*14*日，线上举行，第 11690–11701 页，2021年。网址： https://proceedings.neurips.cc/paper/2021/hash/60cb558c40e4f18479664069d9642d5a-Abstract.html。3 Marcelo Arenas、Pablo Barceló、Diego Bustamante、Jose Caraball 和 Bernardo Subercaseaux。一种用于解释决策树的统一语言。载于 Pierre Marquis、Magdalena Ortiz 和

Maurice Pagnucco 编，《第21届知识表征与推理原理国际会议论文集》，KR 2024，越南河内。2024年11月2日至8日，2024年。 doi:10.24963/KR.2024/6。4 Pablo Barceló、Egor V. Kostylev、Mikaël Monet、Jorge Pérez、Juan L. Reutter 和 Juan Pablo Silva。图神经网络的逻辑表达能力。载于第8届国际学习表征会议（ICLR 2020）。OpenReview.net，2020年。网址：https://openreview.net/forum?id=r1lZ7AEKvB。5 David A. Mix Barrington、Neil Immerman 和 Howard Straubing。关于 NC[1] 中的一致性。《计算机与系统科学杂志》，41(3)：274–306，1990年。 doi:10.1016/0022-0000(90)90022-D。6 L. Blum、F. Cucker、M. Shub 和 S. Smale。《复杂性与实数计算》。Springer，1998年。7 L. Blum、M. Shub 和 S. Smale。关于实数上的计算理论与复杂性：NP 完全性、递归函数和通用机。《美国数学学会通报》，21：1–46，1989年。8 Sophie Brinke、Erich Grädel、Lovro Mrkonjic 和 Matthias Naaf。无限情形下的半环溯源。载于 Antoine Amarilli 和 Alin Deutsch 编，《计算优雅的溯源——献给 Val Tannen 的论文》，Tannen纪念文集，美国宾夕法尼亚大学，宾夕法尼亚州费城，2024年5月24日至25日，OASIcs 第119卷，第3:1–3:26页。Schloss Dagstuhl - Leibniz信息学中心，2024年。 doi:10.4230/OASICS.TANNEN.3。9 Ashok K. Chandra 和 David Harel。关系查询的结构与复杂性。《计算机与系统科学杂志》，25(1)：99–128，1982年。 doi:10.1016/0022-0000(82)90012-5。10 Ashok K. Chandra、Larry J. Stockmeyer 和 Uzi Vishkin。常数深度可约化性。《SIAM计算杂志》，13(2)：423–439，1984年。 doi:10.1137/0213028。11 Katrin M. Dannert、Erich Grädel、Matthias Naaf 和 Val Tannen。不动点逻辑的半环溯源。载于 Christel Baier 和 Jean Goubault-Larrecq 编，《第29届 EACSL 计算机科学逻辑年会论文集》，LIPIcs 第183卷，第17:1–17:22页。Schloss Dagstuhl - Leibniz信息学中心，2021年。 doi:10.4230/LIPICS.CSL.2021.17。12 Laurent Doyen、Goran Frehse、George J. Pappas 和 André Platzer。混合系统验证。载于 Edmund M. Clarke、Thomas A. Henzinger、Helmut Veith 和 Roderick Bloem 编，《模型检验手册》，第1047–1110页。Springer，2018年。doi:10.1007/978-3-319-10575-8_30。13 Heinz-Dieter Ebbinghaus 和 Jörg Flum。《有限模型论》。Springer，第2版，1999年。14 F. Geerts。从查询语言视角看图学习。载于 Floris Geerts、Hung Q. Ngo 和 Stavros Sintos 编，《第42届 ACM数据库系统原理研讨会论文集》，第373–379页，2023年。 doi:10.1145/3584372.3589936。15 Floris Geerts、Thomas Muñoz、Cristian Riveros、Jan Van den Bussche 和 Domagoj Vrgoc。矩阵查询语言。《SIGMOD Record》，50(3)：6–19，2021年。 doi:10.1145/3503780.3503782。16 E. Grädel 和 V. Tannen。一阶逻辑的溯源分析与半环语义。载于 K. Meer、A. Rabinovich、E. Ravve 和 A. Villaveces 编，《模型论、计算机科学与图多项式：献给 Johann A. Makowsky 的纪念文集》。Birkhäuser，2025年。doi:10.1007/978-3-031-86319-6_21。17 Erich Grädel 和 Yuri Gurevich。亚有限模型论。《信息与计算》，140(1)：26–81，1998年。 doi:10.1006/INCO.1997.2675。18 Erich Grädel 和 Klaus Meer。实数上的描述复杂性理论。载于 Frank Thomson Leighton 和 Allan Borodin 编，《第二十七届 ACM 计算理论年会论文集》，第315–324页，1995年。 doi:10.1145/225058.225151。19 T.J. Green、G. Karvounarakis 和 V. Tannen。溯源半环。载于 Leonid Libkin 编，《第26届 ACM数据库系统原理研讨会论文集》，第31–40页，2007年。 doi:10.1145/1265530.1265535。

18 机器学习查询语言

20 Martin Grohe. 图神经网络的逻辑。载于《第36届 ACM-IEEE 计算机科学逻辑研讨会论文集》，第1–17页，2021年。 doi:10.1109/LICS52264.2021.9470677。21 Martin Grohe. 图神经网络的描述复杂性。TheoretiCS，3(25):1–93，2024年。 doi:10.46298/theoretics.24.25。22 Martin Grohe、Christoph Standke、Juno Steegmans 和 Jan Van den Bussche. 基于加权结构递归查询神经网络。arXiv，2601.03201，2026年。 doi:10.48550/arXiv.2601.03201。23 Martin Grohe、Christoph Standke、Juno Steegmans 和 Jan Van den Bussche. 神经网络查询语言。载于 Sudeepa Roy 和 Ahmet Kara 编，《第28届国际数据库理论会议（ICDT 2025）论文集》，莱布尼茨国际信息学论文集（LIPIcs）第328卷，第9:1–9:18页，2025年。 doi:10.4230/LIPIcs.ICDT.2025.9。24 William Hesse. 除法属于一致 TC^0。载于 Fernando Orejas、Paul G. Spirakis 和 Jan van Leeuwen 编，《第28届国际自动机、语言与编程研讨会论文集》，计算机科学讲义第2076卷，第104–114页。Springer，2001年。 doi:10.1007/3-540-48224-5_9。25 William Hesse、Eric Allender 和 David A. Mix Barrington. 用于除法和迭代乘法的一致常数深度阈值电路。《计算机与系统科学杂志》，65(4):695–716，2002年。 doi:10.1016/S0022-0000(02)00025-9。26 Neil Immerman. 描述复杂性。《计算机科学研究生教材》。Springer，1999年。doi:10.1007/978-1-4612-0539-5。27 Gabriel Kuper、Leonid Libkin 和 Jan Paredaens 编。《约束数据库》。Springer，2000年。28 Dietrich Kuske 和 Nicole Schweikardt. 带计数的一阶逻辑。载于《第32届 ACM/IEEE 计算机科学逻辑年会论文集》，第1–12页，2017年。doi:10.1109/LICS.2017.8005133。29 Leonid Libkin. 《有限模型论基础》。《理论计算机科学丛书：EATCS系列》。Springer，2004年。30 Yiannis N. Moschovakis.《抽象结构上的初等归纳》。北荷兰出版社，1974年。31 Lena Strobl、William Merrill、Gail Weiss、David Chiang 和 Dana Angluin. Transformer 能表达哪些形式语言？一篇综述。《计算语言学协会会刊》，12:543–561，2024年。 doi:10.1162/TACL_A_00663。32 S. Torunczyk. 稀疏数据库上的聚合查询。载于 Dan Suciu、Yufei Tao 和 Zhewei Wei 编，《第39届 ACM数据库系统原理研讨会论文集》，第427–443页，2020年。 doi:10.1145/3375395.3387660。33 Steffen van Bergerem 和 Nicole Schweikardt. 学习由权重聚合逻辑描述的概念。载于 Christel Baier 和 Jean Goubault-Larrecq 编，《第29届 EACSL 计算机科学逻辑年会论文集》，LIPIcs第183卷，第10:1–10:18页。Schloss Dagstuhl - Leibniz信息学中心，2021年。 doi:10.4230/LIPIcs.CSL.2021.10。34 Adrian Wurm. 神经网络中的鲁棒性验证。载于 Bistra Dilkina 编，《第21届约束编程、人工智能与运筹学国际会议论文集》第二部分，计算机科学讲义第14743卷，第263–278页。Springer，2024年。 doi:10.1007/978-3-031-60599-4_18。