arXiv:2303.11730v1 [cs.CV] 21 Mar 2023

抽象视觉推理：求解瑞文渐进矩阵的代数方法

Jingyi Xu[1*] Tushar Vaidya[2○*] Yufei Wu[2○*] Saket Chandra[1] Zhangsheng Lai[3○] KaiFong ErnestChong[1†]新加坡科技设计大学[2]南洋理工大学 [3]新加坡理工学院jingyi xu@mymail.sutd.edu.sg tushar.vaidya@ntu.edu.sg yufei002@e.ntu.edu.sg lai zhangsheng@sp.edu.sg {saket chandra,ernest chong}@sutd.edu.sg

摘要

我们提出了代数机器推理，这是一种适用于抽象推理的新型推理框架。实际上，代数机器推理将复杂的新颖问题求解过程简化为常规的代数计算。我们关注的基本代数对象是某个经过适当初始化的多项式环的理想。我们将说明如何将求解瑞文渐进矩阵（*RPM*）转化为代数中的计算问题，这些问题结合了多种广为人知的代数子程序，包括：计算理想的 *Gr¨obner* 基、检查理想包含等。至关重要的是，理想所满足的额外代数结构使得我们能够对理想进行集合论运算以外的更多操作。

我们的代数机器推理框架不仅能够从给定的答案集中选出正确答案，还能仅凭题目矩阵生成正确答案。在 *I-RAVEN* 数据集上的实验取得了 *93.2%* 的总体准确率，显著优于当前最先进的 *77.0%* 准确率，也超过了人类 *84.4%* 的表现。

1. 引言

谈到机器推理，没有什么比机器最终在智力测试和一般推理任务中超越人类的可能性更能激发我们的想象。即使对人类而言，在智商测试中取得优异成绩，例如通过广为人知的瑞文渐进矩阵（瑞文渐进矩阵） [8], ，也已经是一项不容小觑的成就。一个典型的 RPM 实例由一个题目矩阵和一个答案集组成；参见图1。题目矩阵是一个由 3×3 面板组成的网格

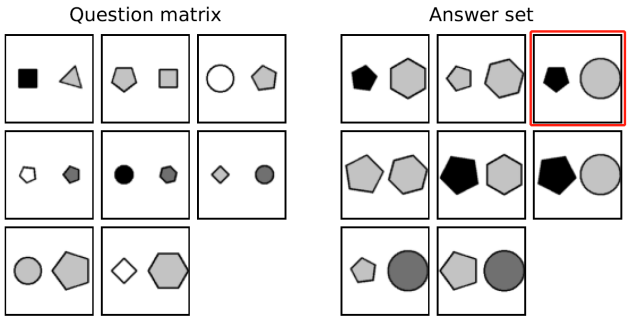


图 1. I-RAVEN 数据集中的一个 RPM 实例示例。正确答案以红框标出。

这些面板遵循某些隐藏规则，其中前 8 个面板填有几何实体，而第 9 个面板则“缺失”。目标是从给定答案集中的 8 个面板里，推断出最后一个面板的正确答案。

解决瑞文渐进矩阵的能力，是认知科学家所说的流体智力的典型体现。“流体”一词指的是发现新关系和进行抽象的思维敏捷性 [56], ，尤其是在解决新颖问题时——这些问题此前从未遇到过。因此，抽象推理新颖问题的能力被广泛视为人类智力的标志 [9]。

尽管近年来机器推理取得了长足进展，[29, 32, 59, 60, 62, 63, 73, 75, 83, 84], 一种常见的批评是，[13, 49, 50] 现有推理框架侧重于需要大量训练的方法，即使是在解决 RPMs 等成熟的推理测试时也是如此。或许最切中要点的是，正如 [13]所主张的，RPM 等推理任务不应需要针对任务

* 贡献相同。 † 通讯作者。
○ 这项工作是在作者此前就职于 SUTD 时完成的。代码：https://github.com/Xu-Jingyi/AlgebraicMR

本研究获得新加坡国家研究基金会通过其 AI Singapore计划（AISG资助编号：AISG-RP-2019-015）和 NRFF计划（NRFFAI1-2019-0005）提供的支持，并获得新加坡教育部通过其第二级研究基金（MOE-T2EP20221-0016）提供的支持。

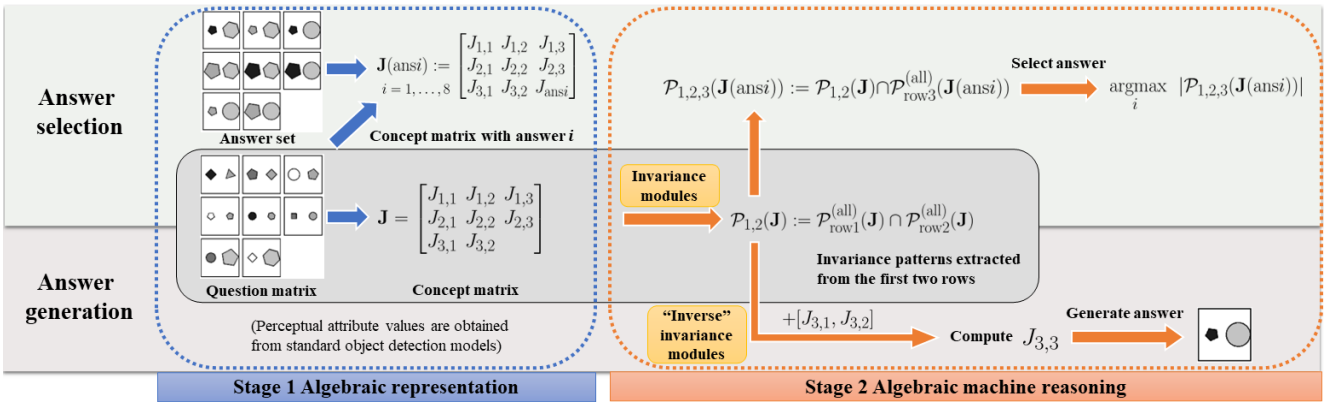


图 2. 我们的代数机器推理框架概览，分为两个阶段。

进行特定的性能优化。毕竟，如果机器通过针对特定任务的数据进行训练来优化性能，那么该任务对机器而言就不可能是新颖的。

为了更好地模拟人类推理，我们提出了一种称为“代数机器推理”的新型推理框架，它非常适用于抽象推理。我们的框架解决瑞文渐进矩阵而无需针对特定任务的数据优化性能，这类似于天赋出众的孩子无需练习瑞文渐进矩阵就能解答这些题目。我们的关键出发点是将概念定义为经过适当初始化的某个多项式环的理想。在代数机器推理中，这些理想被视为“实际研究对象”，无需为其赋予任何数值。我们将阐明如何将 RPM 任务实现为涉及理想的代数计算问题。

我们的推理框架大致分为两个阶段：（1）代数表示；（2）代数机器推理；见图2。在第一阶段，我们根据目标检测模型提取的感知属性值，将 RPM 面板表示为理想。在第二阶段，我们提出4个不变性模块，从瑞文渐进矩阵题目矩阵中提取模式。

总而言之，我们的主要贡献如下：

•我们将“解决 RPM任务”转化为“解决一个代数计算问题”。具体而言，我们展示了如何将抽象模式的发现具体实现为代数计算，即理想的初等分解。

• 在我们的代数机器推理框架中，我们引入了4个不变性模块，用于提取对人类有意义的模式。

• 我们的框架不仅能够从给定的答案集中选出正确答案，还能在无需给定答案集的情况下生成答案。

•在 RAVEN 和 I-RAVEN 数据集上进行的实验表明，我们的推理框架显著优于最先进的方法。

2. 相关工作

RPM 求解器。近年来，使用基于深度学习的方法求解瑞文渐进矩阵受到了广泛关注 [29, 43,62, 84, 85, 88–91]。大多数方法使用神经网络从原始 RPM图像中提取特征，并通过衡量面板之间的相似度来选择答案。另有一些研究则着重于无需答案集即可生成正确答案 [55, 64]。为评估这些方法的推理能力，研究者提出了 PGM [62] 和 RAVEN[83]等类似 RPM 的数据集。随后，研究者又提出了 I-RAVEN [26] 和 RAVEN-FAIR [5] ，以克服 RAVEN 答案集生成中的捷径缺陷。

人工智能中的代数方法。在人工智能中使用代数方法并非新鲜事。多项式方程组常见于计算机视觉 [57] 和机器人学 [12],中，并通过 Gr¨obner 基计算以代数方法求解。在统计学习理论中，代数几何 [78] 和代数统计 [15] 方法被用于研究统计模型中的奇点 [40, 79, 80, 82],、分析层次模型中的泛化误差 [76, 77],、学习概率分布的不变子空间 [34, 38], 以及建模贝叶斯网络 [16, 70]。这些研究的共同主题是研究经过适当定义的代数簇。在深度学习中，代数方法被用于研究神经网络的表达能力 [11, 31, 45, 87]。在自动定理证明中，Gr¨obner 基计算被用于证明检验 [68]。最近，一种一阶逻辑的矩阵表述被应用于 RPM任务 [86], , 其中关系由矩阵近似，推理则被构建为一个双层优化任务，以寻找最佳拟合的矩阵算子。据我们所知，交换代数方法尚未用于机器推理。

3. 拟议的代数框架

在抽象推理中，一个关键的认知步骤是“从观察中发现模式”，具体而言，可表述为——

“在观察中寻找不变性”。本节将介绍如何使用称为理想的代数对象来表示 RPM 实例，如何从这类代数表示中提取模式，以及如何将瑞文渐进矩阵的答案选择和答案生成都作为代数中的计算问题来求解。

3.1. 预备知识

在全文中，令 $R = \mathbb{R}[x_1, \ldots, x_n]$ 为变量 $x_1, \ldots, x_n$ 上的实系数多项式环。特别地， R 在多项式的加法和乘法下封闭，即对于任意的 $a, b \in R$，都有 $a + b, ab \in R$。

3.1.1 代数定义

多项式环中的理想。若存在多项式 $g_1, \ldots, g_k$ 属于 R，使得子集 $I \subseteq R$ .....，则称 $I \subseteq R$ 为一个理想 。

$$I = \{f_1 g_1 + \cdots + f_k g_k | f_1, \ldots, f_k \in R\}$$

包含 $g_1, \ldots, g_k$ 的所有多项式组合。我们称 $\mathcal{G} = \{g_1, \ldots, g_k\}$ 为 I 的生成集，称 $g_1, \ldots, g_k$ 为生成元，并记作 $I = \langle g_1, \ldots, g_k \rangle$ 或 $I = \langle \mathcal{G} \rangle$。注意，理想的生成集并不唯一。如果 I 有一个仅由单项式构成的生成集，那么称 I 为单项式理想。（召回率：单项式 是只有一项的多项式。）给定理想$J_1 = \langle g_1, \ldots, g_k \rangle$ 和 $J_2 = \langle h_1, \ldots, h_\ell \rangle$，有三种基本运算（和、积、交）：

$$\begin{aligned} J_1 + J_2 &:= \langle g_1, \ldots, g_k, h_1, \ldots, h_\ell \rangle; \\ J_1 J_2 &:= \langle \{g_i h_j | 1 \le i \le k, 1 \le j \le \ell\} \rangle; \\ J_1 \cap J_2 &:= \{r \in R : r \in J_1 \text{ and } r \in J_2\}. \end{aligned}$$

涉及理想的大多数代数计算，尤其是“高级”运算（例如初等分解），都需要将计算其 Gr¨obner 基作为关键的初始步骤。更一般地说，Gr¨obner 基的计算构成了代数中大多数算法的骨干；参见附录A.2。

主分解。在交换代数中，理想的主分解是对整数素因数分解概念的广泛推广。其对代数学家的重要性怎么强调都不为过。通俗地说，每个理想 J都可以分解为 $J = J_1 \cap \cdots \cap J_s$ 有限多个主理想的交。这种交称为 J的主分解，而每个 J_j 都称为该分解的主分量。当 J 为单项式理想时，存在一个**唯一的**极小主分解，且其单项式主分量最大 [4]；我们将这组唯一的主分量记作 $\mathrm{pd}(J)$。详情请参见附录A.3。

3.1.2 将概念视为单项式理想

我们将概念 定义为 R 的单项式理想。特别地，零理想 $\langle 0 \rangle \subseteq R$ 即“空”这一概念，可以解释为“不可能”或“什么都没有”；而理想$\langle 1 \rangle = R$ 即“可设想”这一概念，可以解释为“可能”或“一切”。给定一个概念 $J \subseteq R$， J 中的一个单项式称为该概念的实例 。例如，$x_{\text{black}} x_{\text{square}}$ 是 $\langle x_{\text{square}} \rangle$（“正方形”这一概念）的一个实例。对于每个 x_i，我们称 $\langle x_i \rangle \subseteq R$ 是一个原始 概念，而 x_i 是一个原始 实例。

定理 3.1。 尽管 R 中的原始概念有限，但 R 中的概念有无穷多个。此外，若 $J \subseteq R$ 是一个概念，则以下结论成立：

(i) 除非 $J = \langle 0 \rangle$，否则 J 有无穷多个实例。*(ii)* J 有唯一的极小生成集，由有限多个实例组成，我们将其记为$\mathrm{mingen}(J)$。*(iii)* 若 $J \neq \langle 1 \rangle$，则 J 有唯一的一组关联概念 $\{P_1, \ldots, P_k\}$，以及唯一的极小主分解 $J = J_1 \cap \cdots \cap J_k$，使得每个 J_i 都是包含于 P_i 中的概念，并且是在 P_i 中所有作为概念的可能主分量中极大的。

证明见附录 A.4 ，其中证明了定理 3.1 ，并进一步阐述了为何将概念定义为单项式理想能够捕捉人类推理中概念的表达能力。

3.2. 阶段 1：代数表示

我们将使用图 1 所示的 RPM 实例作为贯穿全文的示例，展示完整的代数推理过程：（1）代数表示；以及（2）代数机器推理。本小节将重点介绍第一阶段。回顾一下，每个 RPM 实例都由 16 个填有几何实体的面板组成。在贯穿全文的示例中，每个实体可以用 4 个属性来描述：“颜色”、“大小”、“类型”和“位置”。我们还需要一个额外的属性来表示面板中实体的“数量”。

3.2.1 属性概念

在人类认知中，某些语义相似的概念会自然而然地归为一组，形成一个更一般的概念。例如，“红色”、“绿色”、“蓝色”、“黄色”等概念可以归为一组，形成一个表示“颜色”的新概念。直观地说，我们可以将“颜色”视为一种属性，而将“红色”、“绿色”、“蓝色”、“黄色”视为属性值。

在我们的贯穿示例中，这 5 个属性由 5个概念（单项式理想）表示。一般而言，每个属性的所有可能取值都编码为表示该属性的概念的生成元。不过，为便于说明，我们只考虑图 1 中涉及的属性值来解释示例：

$$\begin{aligned} \mathcal{A}_{\text{num}} &:= \{x_{\text{one}}, x_{\text{two}}\}, \\ \mathcal{A}_{\text{pos}} &:= \{x_{\text{left}}, x_{\text{right}}\}, \\ \mathcal{A}_{\text{type}} &:= \{x_{\text{triangle}}, x_{\text{square}}, x_{\text{pentagon}}, x_{\text{hexagon}}, x_{\text{circle}}\}, \\ \mathcal{A}_{\text{color}} &:= \{x_{\text{white}}, x_{\text{gray}}, x_{\text{dgray}}, x_{\text{black}}\}, \\ \mathcal{A}_{\text{size}} &:= \{x_{\text{small}}, x_{\text{avg}}, x_{\text{large}}\}. \end{aligned}$$

令 $\mathcal{L} :=$ 数量，位置、类型、颜色、大小为属性标签的集合，并令 $\mathcal{A}_{\text{all}} := \bigcup_{\ell \in \mathcal{L}} \mathcal{A}_\ell$。初始化环

$R := \mathbb{R}[\mathcal{A}_{\text{all}}]$ 由 $\mathcal{A}_{\text{all}}$ 中变量构成的所有实系数多项式组成。对于每个 $\ell \in \mathcal{L}$，令 J_ℓ 为概念 $\langle \mathcal{A}_\ell \rangle \subseteq R$。这些概念称为属性概念，具有特定于任务的性质。我们假设人类倾向于根据属性发现并组织复杂模式。因此，在模式提取中，我们将采用这样的归纳偏置：如果表示某种模式的概念属于某个属性概念，那么该概念就被视为有意义。

3.2.2 RPM面板的表示

为了用代数方式编码 RPM图像，我们首先需要训练感知模块，直接从原始图像中提取属性信息。正如我们在实验中采用的一种感知方法是，分别训练 4 个 RetinaNet 模型（每个模型都以 ResNet-50 为骨干网络）来识别除“数量”以外的所有 4 种属性；“数量”则可以通过计算边界框的数量直接推断出来。

提取实体的属性值后，我们就可以将每个面板表示为一个概念。例如，图 1 中 RPM 的左上方面板可以编码为多项式环 R 中的概念$J_{1,1} = \langle x_{\text{two}} x_{\text{left}} x_{\text{square}} x_{\text{black}} x_{\text{avg}}, x_{\text{two}} x_{\text{right}} x_{\text{triangle}} x_{\text{gray}} x_{\text{avg}} \rangle$。其中，$J_{1,1}$ 表示一个包含两个实体的面板：左侧是一个大小为平均值的黑色方块，右侧是一个大小为平均值的灰色三角形。$J_{1,1}$ 中的索引告诉我们，该面板位于第 1 行、第 1 列。类似地，我们可以将题目矩阵中其余 7 个面板编码为概念 $J_{1,2}, J_{1,3}, \ldots, J_{3,2}$，并将 8 个答案选项编码为概念 $J_{\text{ans}1}, \ldots, J_{\text{ans}8}$。一般而言，每个概念中的每个单项式生成元都描述了相应面板中的一个实体。

这8个概念的列表$\mathbf{J} = [J_{1,1}, \ldots, J_{3,2}]$ 称为概念矩阵；它表示缺少第9个面板的 RPM 题目矩阵。令$\mathbf{J}_i := [J_{i,1}, J_{i,2}, J_{i,3}]$ (for$i = 1,2$)表示题目矩阵中的第 i行。

3.3. 阶段 2：代数机器推理

在第3.2节中，我们已经将 RPM 实例中的题目矩阵编码为概念矩阵 $\mathbf{J} = [J_{1,1}, \ldots, J_{3,2}]$。在本小节中，我们将介绍代数框架的推理过程。

我们的目标是提取$\mathbf{J}$单行中的模式，这在数学上可以表述为，在代表该行中各面板的概念之间“寻找不变性”。（同样的过程也可应用于列。）这种看似不够精确的“寻找不变性”概念，可以通过计算初等分解非常具体地实现。理想情况下，我们希望提取对人类有意义的模式。因此，我们设计了4个不变性模块，以模拟人类在模式识别中的认知。

3.3.1 先验知识

为使用代数机器推理，我们采用：

- 属性概念的归纳偏置（参见第 3.2.1节）；
- 数值上的实用二元运算；
- 将概念映射到概念的函数。

存在许多二元运算，例如 $+, -, \times, \div$,最小值,最大值等，可应用于从概念中提取的数值。对于 RPM任务，我们使用 $+, -$。

在代数中，研究代数对象之间的函数，是理解其底层代数结构的一种有效策略。类似地，我们将使用概念上的映射来提取复杂模式。对于 RPM任务，我们需要对 $\mathcal{A}_\ell$ 每个属性 $\ell \in \mathcal{L}$中的值进行循环排序，然后才能提取序列信息。为编码“下一个”的概念，我们引入定义在概念 J上的函数$f_{\text{next}}(J|\Delta)$，其中 Δ 表示步长。生成元 J 中出现的每个变量 $x \in \mathcal{A}_\ell$，都映射到循环序 $\mathcal{A}_\ell$中位于 x之后第 Δ 个的变量。例如，$f_{\text{next}}(\langle x_{\text{square}} x_{\text{gray}} x_{\text{avg}} \rangle | 1) = \langle x_{\text{pentagon}} x_{\text{dgray}} x_{\text{large}} \rangle$，以及 $f_{\text{next}}(\langle x_{\text{square}} \rangle | -2) = \langle x_{\text{circle}} \rangle$。

3.3.2 通过主分解进行推理

给定具有共同“模式”的概念 J_1、……、J_k，我们如何提取这种模式？抽象地说，共同模式可以视为一个包含所有这些概念 K 的概念 J_1、……、J_k。如果存在多个共同模式 K_1、……、K_r，那么每个概念 J_i 都可以分解为 $J_i = K_1 \cap \cdots \cap K_r \cap K_i'$，其中某个理想为 K_i'。因此，我们得到以下代数问题：给定 J_1、……、J_k，计算它们的共同主分量 K_1、……、K_r。

回顾一下，由于概念是单项式理想，概念 J具有唯一的极小主分解。因此，要提取概念 J_1、……、J_k 的共同模式，我们首先必须计算 $\text{pd}(J_1)$……，$\text{pd}(J_k)$，然后提取共同的主分量。这些共同主分量（任意子集）的交将产生一个新概念，可将其解释为概念 J_1、……、J_k的共同模式。作为归纳偏置的一部分，我们只关注包含于属性概念中的主分量。详情请参见附录A.3。

3.3.3 提出不变性模块

我们提出的 4 个不变性模块分别是：(1) 内部不变性模块，(2) 交互不变性模块，(3) 组合不变性模块，以及 (4) 二元运算符不变性模块。直观而言，它们用于检查概念序列中的 4 种一般类型的不变性 J_1，……，J_k(例如 RPM任务中的一行$\mathbf{J}_i = [J_{i,1}, J_{i,2}, J_{i,3}]$)。这些不变性不仅适用于 RPM任务，也可用于其他类似 RPM 的任务，例如基于不同先验知识、不同网格布局等的任务。我们的示例的完整计算细节见附录B.3。

1. 内部不变性模块提取这样的模式：概念 J_i 中的某个属性的取值集合在所有 i中保持不变。首先，我们定义 $J_+ := J_1 + \cdots + J_k$ 以及$J_\cap := J_1 \cap \cdots \cap J_k$；详见第3.1.1节。直观而言，$J_+$ 和

$J_\cap$ 是以两种不同方式捕获整个序列 $J_1, \ldots, J_k$ 信息的概念。接下来，我们计算 J_+ 和 $J_\cap$ 中包含于属性概念的共同主分量。最后，返回与这些共同主分量相关联的属性：$\mathcal{P}_{\text{intra}}([J_1 \ldots J_k]) := \{\text{attr} \in \mathcal{L} \mid \exists I \in \text{pd}(J_+) \cap \text{pd}(J_\cap)\ I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle\}$。

2. 交互不变性模块从 $\text{pd}(J_\cap)$ 与 $\text{pd}(J_+)$ 的集合差中提取模式。随后，我们检查这些提取的模式在多个序列中是否保持不变。提取出的模式集合为：$\mathcal{P}_{\text{inter}}([J_1, \ldots, J_k]) := \Big\{ (\text{attr } \mathcal{I}) \Big| \mathcal{I} \subseteq \text{pd}(\bigcap_{I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle} \forall I \in \mathcal{I} J_\cap - J_+) \Big\}$ $\text{pd}(\,)\ \text{attr} \in \mathcal{L}$，其中 $\mathcal{I}$ 是概念的集合，“–”表示集合差。我们省略 $\text{pd}(J_+)$，以免重复计算前一个模块中已提取的模式。非正式地，对于每一对 $(\text{attr } \mathcal{I})$，$\mathcal{I}$ 中的概念可解释为这样的“主”概念：它们至少对应 $J_1, \ldots, J_k$ 中的一个，但并不对应 $J_1, \ldots, J_k$ 中的所有项，并且包含于 $\langle \mathcal{A}_{\text{attr}} \rangle$ 中。

3. 组合不变性模块从以下新的概念序列中的不变属性值提取模式：

$$[J'_1, \ldots, J'_k] = [f^{k-1}(J_1), f^{k-2}(J_2), \ldots, f(J_{k-1}), J_k],$$

其中 f 是给定函数。直观而言，对于此类模式，存在一些属性，其属性值在 $[f(J_i)\ J_{i+1}]$ 中对所有 $i = 1$、.....、$k-1$均保持不变。通过检查新序列中各概念的主分量的交，得到提取的模式集合：

$$\mathcal{P}_{\text{comp}}([J_1, \ldots, J_k]) := \left\{ (\text{attr}, f) \,\middle|\, \begin{array}{c} \exists I \in \bigcap_{i=1}^{k} \text{pd}(f^{k-i}(J_i)), \\ \text{attr} \in \mathcal{L}, I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle \end{array} \right\}.$$

RPM任务所用的给定函数为$_{\text{next}}$ ·，其中 Δ 表示步数;(参见第3.3.1节。

4. 二元算子模块基于概念上的给定实值函数 g 以及给定的二元运算符集合 Λ，提取数值模式。提取的模式为：

$$\mathcal{P}_{\text{binary}}(\mathbf{J}_i) := \left\{ \overline{\oslash} \,\middle|\, \begin{array}{c} \overline{\oslash} = [\oslash_1, \ldots, \oslash_{k-2}],\ \oslash_i \in \Lambda, \\ g(J_1) \oslash_1 \cdots \oslash_{k-2} g(J_{k-1}) = g(J_k) \end{array} \right\}.$$

3.3.4 提取逐行模式

给定概念矩阵$\mathbf{J} = [J_{1,1}, \ldots, J_{3,2}]$，我们如何从其第 i行提取模式？首先，我们从提取全部 8 个面板中的共同位置值开始：

$$\text{comPos}(\mathbf{J}) := \{p \in \mathcal{A}_{\text{pos}} \mid \exists I \in \textstyle\bigcap_{J \in \mathbf{J}} \text{pd}(J), p \in I\}$$

对于每个共同位置 $p \in \text{comPos}(\mathbf{J})$，我们生成两个新的概念矩阵 $\bar{}\mathbf{J}^{(p)}$ 和 $\hat{}\mathbf{J}^{(p)}$，使得：

- 在 $\bar{}\mathbf{J}_{(p)}$ 中的每个概念 $\bar{J}_{(p)i,j}$，均由 $J_{i,j}$ 中唯一一个可被 p整除的生成元生成；
- 在 $\hat{}\mathbf{J}_{(p)}$ 中的每个概念 $\hat{}_{J(p)i,j}$，均由 $J_{i,j}$ 中所有不能被 p整除的生成元生成。

（召回率：概念的生成元是多项式。）

通俗地说，我们将 RPM 图像中的每个面板拆分为 2 个面板：一个只包含位于共同位置 p的实体，另一个包含所有其余不在位置 p 的实体。这一步使我们能够推理只涉及部分面板的规则。

因此，如果 $\text{comPos}(\mathbf{J}) = \{p_1, \ldots, p_k\}$，那么我们可以将单个概念矩阵扩展为一个概念矩阵列表 $[\mathbf{J}\ \bar{}\mathbf{J}^{(p_1)}, \hat{}\mathbf{J}^{(p_1)} \ldots, \bar{}\mathbf{J}^{(p_k)}, \hat{}\mathbf{J}^{(p_k)}]$。

对于每个概念矩阵J，我们考虑其第 i 行 $\check{}\mathbf{J}_i = [\check{}J_{i,1}, \check{}J_{i,2}, \check{}J_{i,3}]$ (从左到右) $\check{}$，并通过第 3.3.3 节中的 4 个模块从 $\mathbf{J}_i$ 中提取模式。令 $\mathcal{P}(\check{}\mathbf{J}_i)$ 为所有此类模式组成的集合，即，$\mathcal{P}(\check{}\mathbf{J}_i) := \mathcal{P}_{\text{intra}}(\check{}\mathbf{J}_i) \cup \mathcal{P}_{\text{inter}}(\check{}\mathbf{J}_i) \cup \mathcal{P}_{\text{comp}}(\check{}\mathbf{J}_i) \cup \mathcal{P}_{\text{binary}}(\check{}\mathbf{J}_i)$。

最后，对于第 $i = 1,2$ 行，我们定义

$$\mathcal{P}_i^{(\text{all})}(\mathbf{J}) := \bigcup_{\check{\mathbf{J}}} \{(K, \check{\mathbf{J}}) \mid K \in \mathcal{P}(\check{\mathbf{J}}_i)\}, \qquad (1)$$

$\check{}$其中，并集遍历扩展列表中的所有概念矩阵 $\mathbf{J}$，即 $\check{}\mathbf{J} \in [\mathbf{J}\ \bar{}\mathbf{J}^{(p_1)}, \hat{}\mathbf{J}^{(p_1)} \ldots, \bar{}\mathbf{J}^{(p_k)}, \hat{}\mathbf{J}^{(p_k)}]$。注意，$\mathcal{P}_i^{(\text{all})}(\mathbf{J})$ 可视为从原始概念矩阵 $\mathbf{J}$ 的第 i 行中提取的所有模式。如果 $\mathbf{J} = [J_{1,1}, \ldots, J_{3,3}]$ 是一个包含 9 个概念的列表，则可以类似地定义 $\mathcal{P}_3^{(\text{all})}(\mathbf{J})$。

算法1答案选择。

输入： 概念矩阵 $\mathbf{J} = [J_{1,1} \ldots J_{3,2}]$，以及相关的答案集 $[J_{\text{ans}1}, \ldots, J_{\text{ans}8}]$。

```
初始化 comPattern = [0, ..., 0]_{1×8}。
计算 P_{1,2}(J) := P_1^(all)(J) ∩ P_2^(all)(J)。        //见 (1)
对于 i 从 1 到 8执行
    J ← [J_{1,1}, ..., J_{3,2}, J_ansi]
    计算P_3^(all)(J)。
    comPattern[i] ← |P_{1, 2}(J) ∩ P_3^(all)(J)|
返回答案索引 i = argmax_{i'} comPattern[i']。
```

3.4. 求解瑞文渐进矩阵

3.4.1 答案选择

在第3.3.4节中，我们描述了如何使用4个不变性模块提取逐行模式。因此，一种自然的答案选择方法是确定将哪个答案选项填入缺失面板后，能使三个行中共有的模式数量最大化。由此，答案选择可简化为一个简单的优化问题；参见算法1。

3.4.2 答案生成

由于我们的代数机器推理框架能够从原始 RPM 图像中提取出对人类有意义的共同模式，因此无需给定答案集，也能以一种全新的方式生成答案。这类似于一位天赋出众的人类解决 RPM 任务：先识别前两行中的模式，再推断缺失的面板应该是什么。直观地说，我们是在应用4 不变性模块的“逆”运算，来生成代表缺失面板的概念；概览请参见算法2 。

简而言之，给定一个 RPM 概念矩阵 J，我们首先通过 $\mathcal{P}_{1,2}$(J) := $\mathcal{P}_1^{(\text{all})}$ (J) ∩ $\mathcal{P}_2^{(\text{all})}$ (J) 计算前两行中的共同模式；参见 (1)。$\mathcal{P}_{1,2}$(J) 中的每个元素都是一对 (K, ˇJ)，其中 K 是针对单个属性的共同模式ˇ（对应第 1 行和第 2 行），而 J 是相应的概念矩阵。（这代表了天赋出众的人类进行模式发现这一困难步骤。）然后，我们遍历所有共同模式，计算缺失的第九个面板的属性值。（这代表对已发现模式进行常规的一致性检查；完整的算法细节请参见附录 B.2，示例请参见 B.3。）

一般而言，当整合从$\mathcal{P}_{1,2}$(J)中的模式推导出的、适用于$J_{3,3}$ 的所有属性值时，实体可能出现以下情况：(i) 单个属性有多个可能的属性值；或 (ii) 属性值缺失。情况 (i) 出现在为单个属性提取出多个模式时，而情况 (ii) 则出现在该属性没有互不冲突的模式时。对于这两种情况，我们都会从可能的属性值中随机选择一个。

算法 2答案生成。

输入：概念矩阵 J $= [J_{1,1} \dots J_{3,2}]$。 1：对于 K、ˇJ ∈ $\mathcal{P}_1^{(\text{all})}$ (J) ∩ $\mathcal{P}_2^{(\text{all})}$ (J) 中的每一项执行 // 参见 (1) 2：如果 $[\check{}J_{3,}1, \check{}J_{3,2}]$ 与模式 K 不冲突，则 ˇ 3：使用模式 K 计算 $J_{3,}3$ 的属性值。 4：收集上述所有适用于 $J_{3,}$ 3 的属性值。 5：当实体的某个属性存在多个不同值时，执行 6：随机选择一个有效属性值。 7：生成理想 $J_{3,}$ $_3 \subseteq R$。 8：返回$J_{3,}3$ 及对应的图像。

4. 讨论

代数机器推理为机器推理提供了一种全新的范式，超越了数值计算。推理任务中的抽象概念被非常具体地编码为理想，也就是可计算的代数对象。我们将理想视为“实际研究对象”，不要求为其赋予数值。这意味着，我们的框架能够对那些没有自然对应数值的、更具定性或抽象性的概念进行推理。新颖的问题求解，例如从观察结果中发现新的抽象模式，可以具体地通过对理想进行计算来实现（例如，计算理想的初等分解）。尤其是，我们并非在求解多项式方程组，这与代数在人工智能中的现有应用不同（参见第2节）。变量（或原始实例）不会被赋予数值。我们不会在输入值处计算多项式。

从理论角度看，我们提出的方法开创了新的方向。我们在机器推理与交换代数这两个此前完全无关的领域之间建立了新的联系。交换代数中有超过一个世纪积累的深奥成果尚未得到利用。代数方法能否成为解决机器推理领域长期存在的基础性问题的关键？直到2014年，更近一些的时候，ˊ莱昂·博图 [6] 才提出，人类应当“从头开始构建推理能力”，并推测缺失的要素可能是一种代数方法。

为什么用理想来表示概念？为什么不用集合？为什么不用符号表达式，例如多项式？直观而言，我们将概念视为一个“总称”，由该概念的多个（可能无限多个）实例构成。仅将概念视为实例的集合，不足以捕捉人类推理的表达能力。由有限个“原始集合”构成的集合论表示系统，总共只能得到有限个可能的集合。相比之下，我们证明了仅从有限个原始概念就能构造出无限多个概念（定理3.1）。这与我们的直觉一致：人类能够仅凭有限个原始概念表达无限多个概念。主要原因在于，理想“更丰富”的代数结构允许对理想进行远超集合论运算的操作。更多讨论请参见附录A.4。

我们的代数方法为何与基于逻辑的方法（例如基于逻辑编程的方法）有着根本区别？基于逻辑的推理核心思想是，推理可以具体实现为逻辑表达式的归结（或逆归结）。这一思想内在地包含了可满足性的概念；参见 [28]。直观来说，我们有一个逻辑表达式，通常以典范范式表示，并希望为逻辑表达式中的文字赋予真值（真或假），使整个表达式得到满足（即真值为“真”）；更多讨论请参见附录C.1 。事实上，自动定理证明领域许多令人振奋的进展 [2,27,36,39,81,92] 都基于基于逻辑的推理。

相比之下，代数机器推理建立在计算代数和计算机代数系统之上。我们的代数方法的核心思想是，推理可以具体实现为求解代数中的计算问题。关键在于，这里没有可满足性的概念。我们不会为 $R = \mathbb{k}[x_1, \dots, x_n]$中的概念赋予真值（或数值）。具体而言，尽管原始概念 $\langle x_1 \rangle$，……，$\langle x_n \rangle$ 在 R中对应于变量 x_1，……，x_n，我们并不会为原始概念赋值。相反，我们将理想视为“实际研究对象”，并将“求解推理任务”转化为“求解涉及理想的非数值计算问题”。此外，我们的框架能够发现超出 RPM任务既有规则的新模式；参见第5.2节。

在 RPM任务 中，我们有表示“位置”“数量”“类型”“大小”和“颜色”的属性概念；这些概念会根据语义对原始实例进行分类，

	方法	平均准确率	中心	2×2G	3×3G	O-IC	O-IG	L-R	U-D
1	长短期记忆网络 [83]	18.9 / 13.1	26.2 / 13.2	16.7 / 14.1	15.1 / 13.7	21.9 / 12.2	21.1 / 13.0	14.6 / 12.8	16.5 / 12.4
2	WReN [62]	23.8 / 34.0	29.4 / 58.4	26.8 / 38.9	23.5 / 37.7	22.5 / 38.8	21.5 / 22.6	21.9 / 21.6	21.4 / 19.7
3	残差网络 [83]	40.3 / 53.4	44.7 / 52.8	29.3 / 41.9	27.9 / 44.3	46.2 / 63.2	35.8 / 53.1	51.2 / 58.8	47.4 / 60.2
4	残差网络+DRT [83]	40.4 / 59.6	46.5 / 58.1	28.8 / 46.5	27.3 / 50.4	46.0 / 69.1	34.2 / 60.1	50.1 / 65.8	49.8 / 67.1
5	长度 [88]	41.4 / 72.9	56.4 / 80.2	31.7 / 57.5	29.7 / 62.1	52.1 / 84.4	31.7 / 71.5	44.2 / 73.5	44.2 / 81.2
6	CoPINet [84]	46.1 / 91.4	54.4 / 95.1	36.8 / 77.5	31.9 / 78.9	52.2 / 98.5	42.8 / 91.4	51.9 / 99.1	52.5 / 99.7
7	DCNet [91]	49.4 / **93.6**	57.8 / 97.8	34.1 / 81.7	35.5 / 86.7	57.0 / **99.0**	42.9 / **91.5**	58.5 / **99.8**	60.0 / **99.8**
8	NCD [89]	48.2 / 37.0	60.0 / 45.5	31.2 / 35.5	30.0 / 39.5	62.4 / 40.3	39.0 / 30.0	58.9 / 34.9	57.2 / 33.4
9	SRAN [26]	60.8 / -	78.2 / -	50.1 / -	42.4 / -	68.2 / -	46.3 / -	70.1 / -	70.3 / -
10	PrAE [85]	77.0 / 65.0	90.5 / 76.5	85.4 / 78.6	45.6 / 28.6	63.5 / 48.1	60.7 / 42.6	96.3 / 90.1	97.4 / 90.9
11	我们的方法	**93.2** /92.9	**99.5** / **98.8**	**89.6** / **91.9**	**89.7** / **93.1**	**99.6** /98.2	**74.7** /70.1	**99.7** /99.2	**99.5**/ 99.1
	人类 [83]	- / 84.4	- / 95.5	- / 81.8	- / 79.6	- / 86.4	- / 81.8	- / 86.4	- / 81.8

表1. 在 I-RAVEN/RAVEN 上的性能。我们报告平均准确率，以及所有配置的准确率：中心、2×2网格、3×3网格、外-内中心、外内网格、左-右和上下。

归入人类所称的属性。直观而言，属性概念会以一种对任务而言有“意义”的方式，将某些原始概念组合在一起。例如，$\langle x_{\text{white}}, x_{\text{gray}}, x_{\text{black}}\rangle$ 比 $\langle x_{\text{white}}, x_{\text{circle}}, x_{\text{large}}\rangle$ 更“有意义”，因为它是一个更“简单”或更“概括”的概念；我们会将 x_{white}，x_{gray}，x_{black} 视为同一个更宽泛的“颜色”概念的实例。

请注意，原始概念与我们的目标检测模型的预测类别完全对应。这些预测类别已由可用数据隐式确定。因此，我们的方法受限于感知模块能够感知的内容。对于其他任务，例如存在文本数据的任务，可以使用实体抽取方法来识别原始概念。还请注意，我们的方法需要先验知识，因为推理模块没有训练步骤。如果将概念上的用户定义函数替换为通过深度学习优化的可训练函数，则可以缓解这一限制。

一般而言，属性概念的识别因任务而异，最终的推理性能很大程度上取决于识别出的属性概念。实际上，我们选择哪些属性概念，将决定推理框架的归纳偏置：当我们将概念 J 分解为“更简单”的概念（即 $\text{pd}(J)$ 中的主分量）时，只有包含在属性概念中的“更简单”概念才被视为“有意义”。具体而言，设 J，$J' \subsetneq R$ 为概念，且 $\text{pd}(J) = \{J_1, \ldots, J_k\}$ 和 $\text{pd}(J') = \{J'_1, \ldots, J'_\ell\}$，即 J，J' 分别具有极小主分解 $J = J_1 \cap \cdots \cap J_k$ 和 $J' = J'_1 \cap \cdots \cap J'_\ell$。我们可以考察它们的主分量，并从中找出包含在某个共同属性概念中的主分量。例如，若 A 是 R 的一个属性概念，且 $J_1 \subseteq A$ 和 $J'_1 \subseteq A$，则 J 和 J' 共享一个“共同模式”，该模式由属性概念 A 表示。

5. 实验结果

为展示我们框架的有效性，我们在 RAVEN [83]和 I-RAVEN 数据集上进行了实验。在这两个数据集中，瑞文渐进矩阵均按照 7 种配置生成。我们使用 I-RAVEN 数据集中的 4200 张图像训练感知模块 [26]（每种配置 600 张），并用它们预测实体的属性值。感知模块的平均准确率为 96。24%。对于这两个数据集，我们针对每种配置测试了 2000 个实例。总体而言，我们的推理框架运行速度很快（在 16 核第 11 代英特尔 i7 CPU 处理器上，处理 14000 个实例耗时 7 小时）。完整的实验细节请参见附录 B。

5.1 与其他基线方法的比较

表1 比较了我们的方法与10 种其他基线方法的性能。对于方法 1–7，我们采用 [26, 89] 中报告的 I-RAVEN 准确率；对于方法 1–5，则采用 [83, 89] 中报告的 RAVEN 准确率。其余所有准确率均取自原始论文。作为参考，我们还列出了人类在RAVEN 数据集（即非 I-RAVEN）上的表现，该数据报告于 [83]。

5.2 模糊实例与新模式

尽管我们的方法优于所有基线方法，但有些实例中，多个答案选项在我们的框架下获得了相同的最高分。这些情况大多是由于发现了（i）“偶然”出现的非预期规则（例如图3）；或（ii）数据集中实际规则之外的新模式（例如图4）。情况（i）之所以出现，是因为在 I-RAVEN 的设计中，每个属性最多只分配一条规则。

有趣的是，情况（ii）表明，我们的框架能够发现全新的模式，而这些模式最初并未被设计为 I-RAVEN 的规则。在图 4 中，发现的新模式可以说对人类而言非常自然。

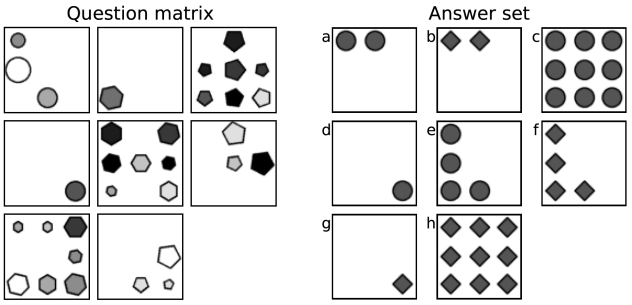


图 3. 一个存在歧义的 RPM 实例示例。给定答案为选项 **g**。对于 I-RAVEN，前两行中的类型序列（“圆形”、“六边形”、“五边形”）遵循类型索引依次递减的递进规则，因此**g** 可能是正确答案。（其余属性值由其他模式确定。）然而，由于类型的一种跨行不变模式（类型集合 {“圆形”、“六边形”、“五边形”} 在各行中保持不变），我们的框架对选项 **d** 和**g**给出了相同的最高分。因此，选项**d** 也可能是正确答案。

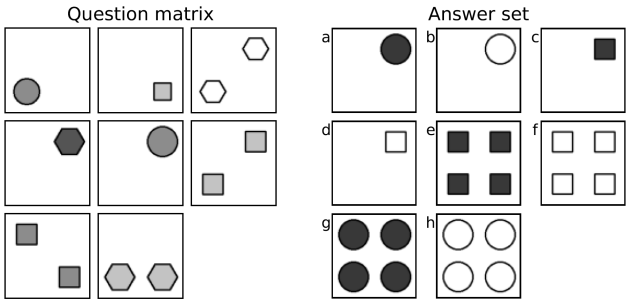


图 4. 一个具有意外新模式的 RPM 实例示例。给定答案为选项 **h**。每一行中，前两个面板的实体数量之和等于第三个面板中的实体数量，因此**h** 可能是正确答案。然而，由于数量的一种新跨行不变模式（非正式地说，每个面板都含有 1 个或 2 个实体），我们的框架对选项 **b** 和**h**给出了相同的最高分。因此，选项**b** 也可能是正确答案。

5.3. 答案生成评估

假设每个 RPM 实例在给定的答案集中都有一个唯一的正确答案。然而，还有许多其他可能的图像也可以接受为正确答案。例如，通过对不涉及任何规则的属性进行随机扰动（例如 I-RAVEN 数据集中的实体角度），从给定正确答案修改而来的图像同样是正确的。基于关于哪些原始感知属性与 RPM任务相关的先验知识，所有这些不同的正确答案（图像）都可以代数编码为同一个概念。因此，为了评估第 3.4.2 节提出的答案生成过程，我们将直接评估生成的概念。

设 $J = \langle e_1, \ldots, e_k \rangle$ 和 $J' = \langle e'_1, \ldots, e'_\ell \rangle$ 为概念分别表示真实答案和我们生成的答案。这里，每个 e_i（或 e'_i）都是形如 $x_i^{(\text{pos})} x_i^{(\text{type})} x_i^{(\text{color})} x_i^{(\text{size})}$ 的单项式，表示由 4 个属性描述的实体。受广为人知的交并比（IoU）概念启发，我们提出一种衡量 J 与 J'之间相似度的新方法。为了定义类似“交”和“并”的概念，我们首先在满足 $x_i^{(\text{pos})} = x_j'^{(\text{pos})}$（即“位置”值相同）的情况下，将 e_i 与e'_j 配对。由于任一面板中实体的“位置”值均由其唯一确定，因此这种配对定义良好。于是，我们可以将 J 和 J' 中的所有实体划分为 3 个集合：

$$S_1 := \{(e_i, e'_j) \mid e_i \in J, e'_j \in J', x_i^{(\text{pos})} = x_j'^{(\text{pos})}\};$$
$$S_2 := \{e_i \in J \mid \nexists e'_j \in J' \text{ such that } (e_i, e'_j) \in S_1\};$$
$$S_3 := \{e'_j \in J' \mid \nexists e_i \in J \text{ such that } (e_i, e'_j) \in S_1\}.$$

我们可以将 S_1 和 $S_1 \cup S_2 \cup S_3$ 分别视为 J 和 J' 的“交”与“并”的类似概念。因此，我们将相似度度量定义如下：

$$\varphi(J, J') := \frac{\sum_{(e_i, e'_j) \in S_1} \phi(e_i, e'_j)}{|S_1| + |S_2| + |S_3|}; \tag{2}$$

$$\phi(e_i, e'_j) := \frac{1}{4} \sum^{a} \mathbb{1}(x_i^{(\text{a})} = x_j'^{(\text{a})}); \tag{3}$$

其中 (3) 中，a 遍历 {位置、类型、颜色、大小}这 4 个属性。这里，$\phi(e_i, e'_j)$ 是 e_i 与 e'_j之间的相似度分数，以公共变量所占比例衡量。

生成答案的总体平均相似度分数为 67.7%。请注意，在一个面板中，某些属性值（例如“大小”、“颜色”和“位置”）对于2×2网格、3×3网格、外内网格 (例如图 3所示）可能完全随机。因此，在这类情况下取得较高的相似度分数，必然需要针对具体任务进行优化，并了解数据的生成方式。而我们并未作出这样的假设。这或许可以解释为何我们的总体相似度分数低于答案选择准确率。

生成图像示例请参见附录B.5。

6. 结论

代数机器推理是一种非常适合抽象推理的推理框架。在当前形式下，我们使用主分解这一关键代数运算，通过专门设计来模仿人类推理的不变性模块，在RPM任务中发现抽象模式。“发现共同模式”可以具体实现为“计算主分解”这一想法适用范围相当广，也有望应用于其他推理任务。

更一般地说，我们的代数方法为利用交换代数和计算代数的浩瀚文献开辟了新的可能性。理想的代数运算（理想商、根基、饱和等）以及代数不变量（深度、高度等）有许多尚未在机器推理（甚至人工智能）中得到探索。我们能否利用它们来应对其他推理任务?

《抽象视觉推理：用代数方法求解瑞文渐进矩阵》的附录

提纲。 作为本文补充材料的一部分，我们提供了更多细节，内容安排在以下三个附录中：

- 附录A 进一步介绍了代数机器推理的相关细节。
- 附录B 进一步介绍了我们求解瑞文渐进矩阵的实验细节。
- 附录C 进一步讨论了代数机器推理，包括其潜在的社会影响，以及它与基于逻辑的推理之间的差异。

A. 更多代数细节

本附录深入阐述了第3节中介绍的代数思想。对于定义和相关术语，我们尽可能使本附录内容自成一体。对于“标准”的代数结果，我们尽可能提供教科书中明确证明的参考文献；否则则给出完整的证明细节。为了便于阅读，读者最好至少对交换代数的基础知识有所了解，例如达到本科抽象代数的水平。若想温和地入门交换代数和计算代数，我们推荐 [12]。若想更深入地学习这一主题，请参阅 [17]。

在整个附录中，设 $R := \Bbbk[x_1, \ldots, x_n]$ 是以 n 个变量 $x_1, \ldots, x_n$为元、系数取自域 $\Bbbk$的多项式环。为简便起见，本文假设$\Bbbk = \mathbb{R}$；事实上，$\Bbbk$ 可以是任意域。特别地，$\Bbbk$ 不必是代数闭域。[1]（注意，$\mathbb{C}$是代数闭域，而 $\mathbb{R}$ 不是。）这意味着我们允许 $\Bbbk = \mathbb{R}$。我们也允许 $\Bbbk$ 是特征非零的域（例如某个素数 $\mathbb{Z}/p\mathbb{Z}$ 对应的有限域 p）。域 $\Bbbk$ 的具体选择与本附录讨论的细节无关，因为推理所需的代数计算都不涉及对多项式各项系数进行数值计算。为明确起见，读者此后可以假设 $\Bbbk = \mathbb{R}$，而不失一般性。

[1]处理涉及 n 个变量的多项式方程组时，若将多项式方程的系数视为 $\Bbbk$中的值，那么该多项式方程组的任何解都是 $\Bbbk^n$中的一个点。人们通常假设 $\Bbbk$ 是代数闭域，以便更轻松地分析解空间。同样，在考察这类多项式方程组所对应的代数簇时，若基域 $\Bbbk$ 是代数闭域，分析起来会更容易。本文不处理或求解多项式方程组，也不研究代数簇。因此，我们不会假设 $\Bbbk$ 是代数闭域。

在代数记法中，我们用“$\langle g_1, \ldots, g_k \rangle$”表示由 $g_1, \ldots, g_k$生成的理想；与之相对，有些作者用“$(g_1, \ldots, g_k)$”来表示理想。对我们而言，圆括号专用于表示元组，例如“$(g_1, \ldots, g_k)$”是一个 k元组。我们使用的其他代数记法均与当今交换代数文献中的记法一致。

Th其余部f Appendix A i的组织ized as follows:

- 附录 A.1 进一步介绍相关概念（即单项式理想）及其相关结果，包括理想的代数运算，并重点讨论单项式理想这一特殊情形。
- 附录 A.2 详细介绍 Gr¨obner 基理论，这是计算代数和交换代数中大多数算法所依赖的关键“主力工具”。
- 附录 A.3 进一步介绍了准素理想和主分解的细节，并讨论了 主分解与代数机器推理的归纳偏置之间的关系。
- 附录 A.4补充完成了第 3.1节 3.1.2中的定理证明，并详细说明了为什么将概念定义为单项式理想能够捕捉人类推理中概念的表达能力。

A.1. 概念、单项式理想与理想运算

令 $\mathbb{N}$ 为非负整数集。对于每个$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$，令$\mathbf{x}^{\boldsymbol{\alpha}}$ 表示 R中的单项式$x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$。根据定义，$R$ 中的单项式 是形如$\mathbf{x}^{\boldsymbol{\alpha}}$的多项式，即单项式的首项系数始终为 1，也就是域 $\Bbbk$中的乘法单位元。$x^{\boldsymbol{\alpha}}$的**次数** 记作 $\deg(\mathbf{x}^{\boldsymbol{\alpha}})$，是 $\alpha_1 + \cdots + \alpha_n$的和。例如，$\deg(x_1^2 x_2^3 x_3^7) = 12$。

A.1.1 概念的代数性质

设 $\mathcal{M}_R$是 R中所有单项式的集合。首先注意到，多项式环 R 是一个以 $\mathcal{M}_R$为基的 $\Bbbk$-向量空间。这意味着，R 中的每个多项式 p 都可写成 $p = \sum_{m \in \mathcal{M}_R} a_m \cdot m$，其中每个 a_m 都是 $\Bbbk$中唯一确定的标量，称为单项式 m 在 p 中的系数。如果单项式 $m \in \mathcal{M}_R$ 的系数 a_m 非零，则称其为 p 的单项式。

命题 A.1。 *[12, 第 2.4 章，引理 3] 设 p 是一个关于 R，且设 J 是一个关于 R的单项式理想。则p 是 J 的一个元素，当且仅当每个单项式 m （属于 p）都是 J的一个元素。*

尽管 $\mathcal{M}_R$ 是无限的（因为它包含所有次数的单项式），但众所周知，每个

理想 I 属于 R 的理想都具有有限的生成集；这就是著名的希尔伯特基定理（参见 [12,第 2.5 章，定理4]或 [17,定理 12] 中的证明）。对于单项式理想，我们有以下更强的结果：

命题 A.2。 *[48,引理 1。2]每个 R 的单项式理想都有一个由有限个单项式生成元组成的唯一的极小生成集。*

为了理解命题A.2，注意，对于理想 $\mathcal{G}$ J R，其生成集若满足不存在严格更小的子集 $\mathcal{G}' \subsetneq \mathcal{G}$ 能够生成 J，则称其为极小生成集。这意味着，若给定单项式理想 J R，以及任意一个 $\mathcal{M} \subseteq \mathcal{M}_R$由单项式组成、能够生成 J的子集，即 $J = \langle \mathcal{M} \rangle$，那么命题A.2 告诉我们，总能找到一个极小子集$\mathcal{M}' \subseteq \mathcal{M}$ （关于集合包含关系而言），它生成相同的理想 J，且这个极小子集 $\mathcal{M}'$ 是唯一确定的，不依赖于我们所选择的 $\mathcal{M}$。这个唯一的极小单项式生成集记作$\mathrm{mingen}(J)$。

给定 R 的任意单项式理想 J，包含在 mingen(J) 中的单项式称为J的极小单项式生成元。由于概念被定义为 R 的单项式理想，我们可以类似地将 J 的极小实例定义为 J 的极小单项式生成元。因此，一个概念由其所有极小实例生成。

回顾一下，理想有三种基本运算：和、积和交。设 $J_1 = \langle g_1, \ldots, g_k \rangle$ 和 $J_2 = \langle h_1, \ldots, h_\ell \rangle$ 是 R 的任意两个理想。那么，和 $J_1 + J_2$、积 $J_1 J_2$ 和交 $J_1 \cap J_2$ 是按如下方式定义的理想：

$$\begin{aligned} J_1 + J_2 &:= \langle g_1, \ldots, g_k, h_1, \ldots, h_\ell \rangle; \\ J_1 J_2 &:= \langle \{ g_i h_j | 1 \le i \le k, 1 \le j \le \ell \} \rangle; \\ J_1 \cap J_2 &:= \{ r \in R : r \in J_1 \text{ and } r \in J_2 \}. \end{aligned}$$

特别地， $J_1 \cap J_2$ 是一个理想，因为对于 $J_1 \cap J_2$ 中任意多项式 $r_1, \ldots, r_s$，多项式组合$r_1 p_1 + \cdots + r_s q_s$（对于 R 中任意多项式 $p_1, \ldots, p_s$）按定义同时包含于 J_1 和 J_2中。

这些理想上的基运算有一个有用的性质：单项式理想在此类运算下仍为单项式理想：

命题 A.3. 如果 J_1 和 J_2 是 R的概念，那么 $J_1 + J_2$、$J_1 \cap J_2$ 和 $J_1 J_2$ 也都是 R的概念。此外，我们总有 $J_1 J_2 \subseteq J_1 \cap J_2$。

证明。 显然， $J_1 + J_2$ 和 $J_1 J_2$ 是单项式理想，因为它们的定义中指定的生成元都是单项式。此外，对于每个 $1 \le i \le k$、 $1 \le j \le \ell$，按照定义，元素 $g_i h_j$ 同时属于 J_1 和 J_2，这意味着 $J_1 J_2$ 的每个极小单项式生成元都是$J_1 \cap J_2$的元素，因此得到 $J_1 J_2 \subseteq J_1 \cap J_2$。

最后，我们将证明 $J_1 \cap J_2$ 是单项式理想。设 $\mathcal{M}' := \mathcal{M}_R \cap J_1 \cap J_2$，并定义单项式理想$J' := \langle \mathcal{M}' \rangle \subseteq R$。根据定义， $\mathcal{M}' \subseteq J_1 \cap J_2$，因此$J' \subseteq J_1 \cap J_2$。为证明逆命题 $J' \supseteq J_1 \cap J_2$成立，任取一个元素 $p \in J_1 \cap J_2$，并写成$p = \sum_{m \in \mathcal{M}_R} a_m \cdot m$，其中每个 $a_m \in \Bbbk$ 都是单项式 m的系数。注意， $p \in J_1$ 且 $p \in J_2$。由于 J_1和 J_2 都是单项式理想，命题 A.1 表明， p 中的每个单项式 m（即其系数为 $a_m \neq 0$）都是 $J_1 \cap J_2$的元素，因此也必属于$\mathcal{M}' = \mathcal{M} \cap J_1 \cap J_2$。于是， $p \in \langle \mathcal{M}' \rangle = J'$。 □

A.1.2 不同类型概念的层次结构

代数机器推理的第一阶段是代数表示。对于 RPM任务，我们从原始 RPM图像中提取属性信息，并将每个 RPM 面板代数地编码为一个概念。我们用于 RPM 面板代数表示的这些概念有一个特别好的性质：它们的极小单项式生成元是平方自由的。回想一下，单项式$\mathbf{x}^{\boldsymbol{\alpha}}$ 在 R（对于某个 $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$）称为**平方自由的**，如果每个指数 α_i 都是 0 或 1，或者等价地，若 $\mathbf{x}^{\boldsymbol{\alpha}} \notin \langle x_1^2, \ldots, x_n^2 \rangle$。（非正式地说，单项式 m是平方自由的，如果其中出现的变量 m 没有“更高次幂”，即所有变量的指数≥ 2均不超过 1。）由平方自由单项式生成的单项式理想称为 **平方自由单项式理想**，交换代数中对这类理想进行了大量研究，因为它们具有丰富的性质 [48, 66]。因此，我们有必要定义以下“基本”概念。

定义 A.4。 如果 $J \neq R$，且其所有极小实例（即极小单项式生成元）都是平方自由的，则称概念 J 在 R 中是**基本的**。

我们将在后面的定理 A.13中证明，基本概念具有特别适合提取模式的结构。

我们还将引入“简单”概念的概念。考虑理想 $J_1 := \langle x_{\text{black}} x_{\text{square}}, x_{\text{white}} x_{\text{circle}} \rangle$，它是一个基本概念，表示“黑色方块或白色圆形”。这样的概念包含于直观上更简单的概念 $J_2 := \langle x_{\text{black}}, x_{\text{white}} \rangle$，后者表示“黑色或白色”。注意， J_2 由两个原始实例 x_{black}和 x_{white}生成。为了体现“黑色或白色”比“黑色方块或白色圆形”更简单这一直觉，我们将 J 中的概念 R 定义为简单的， J 当且仅当它由 R中的非空原始实例集生成，也就是说，它是$\{x_1, \ldots, x_n\}$的非空子集。等价地，若一个概念是原始概念之和，则它是简单的。注意，所有简单概念都是基本概念，因为原始实例是平方自由的。正如我们将在附录A.3.2中看到的，每个基本概念都可以“分解”为多个简单概念。

总而言之，我们得到以下不同类型概念的层次结构：

$$\left\{\text{concepts}\right\} \supseteq \left\{\begin{array}{c}\text{basic}\\ \text{concepts}\end{array}\right\} \supseteq \left\{\begin{array}{c}\text{simple}\\ \text{concepts}\end{array}\right\} \supseteq \left\{\begin{array}{c}\text{primitive}\\ \text{concepts}\end{array}\right\}.$$

A.2. 格罗布纳基理论

本小节回顾格¨罗布纳基理论中的基本术语和若干初等结果。关于理想的格¨罗布纳基是什么，请参见 [69]。

格罗布纳基理论为何重要？本质上，如果我们想要计算涉及理想的计算问题的答案，通常需要先计算输入理想的格罗布纳基，这是关键的第一步。后续计算步骤通常会直接使用算出的格罗布纳基，而非输入理想最初给定的生成集。为避免读者认为格罗布纳基计算只是针对理想的“更高级”运算，我们在此强调，计算理想的交本身就依赖格罗布纳基计算：给定，的理想 $J_1 J_2$，它们分别具有生成集 $\mathcal{G}_1$，$\mathcal{G}_2$，计算交 $J_1 \cap J_2$ 的生成集已经并非易事。

A.2.1 什么是 Gr¨obner 基?

回想一下，R 是一个以 $\mathcal{M}_R$为基的 $\Bbbk$向量空间，其中$\mathcal{M}_R$ 表示 R中的单项式集合。Gr¨obner 基理论的出发点是对集合 $\mathcal{M}_R$进行排序。

定义 A.5. R 上的一个单项式序是$\mathcal{M}_R$上的一个良序，记作 $<$，并满足以下条件：对于 $\mathcal{M}_R$ 中的任意单项式 m_1, m_2, m，只要满足 $m \neq 1$，

$$\text{如果 } m_1 > m_2, \text{ 那么 } mm_1 > mm_2 > m_2。$$

给定一个单项式序 $<$，以及R中的任意多项式 p，总可以写成 $p = c_1 m_1 + \cdots + c_k m_k$，其中 $c_1, \ldots, c_k$ 是 $\Bbbk$中的非零标量，$m_1, \ldots, m_k$是 $\mathcal{M}_R$ 中的单项式，并且按照单项式序<按降序排列$m_1 > \cdots > m_k$。给定 p和 $<$后，这些标量和单项式都是唯一确定的。回想一下，单项式 $m_1, \ldots, m_k$恰好是 p的所有单项式，而每个标量 c_i 都是 p中 m_i 的系数。p 的首项单项式（相对于 $<$），记作 in$_<$(p)，是其中最大的单项式 m_1。默认情况下，我们定义 in$_<$(0) $= 0$。

设 $\mathcal{A} = \{p_1, \ldots, p_k\}$ 是 R 中的一个多项式有限集。写作 $p = \left(\sum_{i=1}^k f_i p_i\right) + q$，其中 f_1、.....、f_k、q 是 R 中的多项式，满足对于所有 $1 \leq i \leq k$，in$_<$(p) $\geq$ in$_<$($f_i p_i$)，并且 q 的任何单项式都不能被包含在 {in$_<$(p_1)、.....、in$_<$(p_k)} 中的任何单项式整除。这样的表示总是存在，称为 p 关于 $\mathcal{A}$ 的标准表示。任意这样的多项式 q 称为 p 关于 $\mathcal{A}$ 的余式。如果 0 是 p 关于 $\mathcal{A}$ 的余式，则称 p 关于 $\mathcal{A}$化简为零。注意，p 的标准表示和余式（关于 $\mathcal{A}$）通常并不唯一。

设 $f, g \in R$ 为不全为零多项式的多项式。f 和 g 关于 $<$的 $\mathcal{S}$-对 是多项式

$$\mathcal{S}(f,g) := \frac{c_g \operatorname{in}_<(g)}{\gcd(\operatorname{in}_<(f), \operatorname{in}_<(g))} f - \frac{c_f \operatorname{in}_<(f)}{\gcd(\operatorname{in}_<(f), \operatorname{in}_<(g))} g,$$

其中，gcd(in$_<$(f), in$_<$(g)) 表示 in$_<$(f) 和 in$_<$(g) 的最大公因式，而 c_f（分别为 c_g）是 f（分别为 g）中 in$_<$(f)（分别为 in$_<$(g)）的系数。对于所有非零多项式$f \in R$，我们约定 gcd(f,0) $= f$。默认定义 $\mathcal{S}$(0,0) $= 0$。容易验证，若 gcd(in$_<$(f), in$_<$(g)) $= 1$，则 $\mathcal{S}(f,g)$ 关于 $\{f, g\}$ 总能化为零。

对于 R的任意理想 I，由 I中所有元素的首项单项式生成的单项式理想 in$_<$(I) :=$\langle${in$_<$(f)$|f \in I$}$\rangle$，称为 I 关于 $<$的初始理想。I 关于 $<$的*Gr¨obner* 基是 I中元素的一个**有限**集 $\mathcal{G}$，满足 in$_<$(I) 由集合 {in$_<$(f)$|f \in \mathcal{G}$} 生成。对于 I，关于任意单项式序的 Gr¨obner 基总是存在的，并且 I 的每个 Gr¨obner 基都必须生成 I。若 $\mathcal{G}$ 是 I的 Gr¨obner 基，则 p关于 $\mathcal{G}$ 的余式唯一。特别地，$p \in I$ 当且仅当 p 关于 $\mathcal{G}$可化为零。

考虑理想 I 的任意有限生成集 $\mathcal{A}$。布赫伯格判据指出，$\mathcal{A}$ 是关于某个给定单项式序 $<$ 的 Gr¨obner 基，当且仅当对于每一对 f、$g \in \mathcal{A}$，$\mathcal{S}(f,g)$ 都能化简为零。该判据导出了布赫伯格算法，这是一种算法：给定理想 I 的有限生成集 $\mathcal{A}$ 作为输入，计算多项式环 R S 中任意理想 I 的 Gr¨obner 基 $\mathcal{G}$。布赫伯格算法的描述如下：以输入生成集 $\mathcal{A}$ 为起点，对于所有生成元 f、$g \in \mathcal{A}$，计算 $\mathcal{S}$ 对 $\mathcal{A}$ 的 $\mathcal{S}(f,g)$ 的余式；如果任何这样的余式 q 非零，则将多项式 q 加入 $\mathcal{A}$，并重复此过程。该过程必定在有限步后终止；终止后得到的最终 $\mathcal{A}$ 是 I 的 Gr¨obner 基，其中包含最初给定的生成集 $\mathcal{A}$。注意，根据定义，最终 Gr¨obner 基 $\mathcal{A}$ 中任意一对元素的所有 $\mathcal{S}$-对，都必须关于 $\mathcal{A}$ 化简为零。

A.2.2 代数中的计算子程序

代数中的计算问题可以借助计算机代数系统求解。知名的计算机代数系统（例如 Maple、Magma、Mathematica、MATLAB；参见[12, 附录C]）常用于求解多项式方程组，而更专业的系统（例如

CoCoA [1], GAP [19], Macaulay2 [24], SageMath [72],Singular [14]）则更常用于代数学家解决这类计算问题：其中理想是“研究的实际对象”，也就是说，不涉及为变量赋予数值。

Gr¨obner 基总是相对于某种给定的单项式序 $<$ 定义的。在许多计算机代数系统中（例如 CoCoA [1], Macaulay2 [24], SageMath [72],Singular [14]），Gr¨obner 基计算的默认单项式序是分次逆字典序，也称为次数逆字典序。

定义 A.6. R 上的分次逆字典序（grevlex 序），记为 $<_{r\ell}$，是 R 上的一种单项式序。对于 R 中任意两个不同的单项式$\mathbf{x}^{\boldsymbol{\alpha}}$ 和$\mathbf{x}^{\boldsymbol{\beta}}$，其中 $\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_n)$ 且 $\boldsymbol{\beta}=(\beta_1,\ldots,\beta_n)$，有$\mathbf{x}^{\boldsymbol{\alpha}} >_{r\ell} \mathbf{x}^{\boldsymbol{\beta}}$，当且仅当以下条件之一成立：$\deg(\mathbf{x}^{\boldsymbol{\alpha}}) > \deg(\mathbf{x}^{\boldsymbol{\beta}})$；或者 $\deg(\mathbf{x}^{\boldsymbol{\alpha}}) = \deg(\mathbf{x}^{\boldsymbol{\beta}})$，且 $\alpha_i < \beta_i$ 对于满足条件的最大值 $1\le i\le n$，有 $\alpha_i\neq\beta_i$。

例如，如果 $n=3$（即 $R=\Bbbk[x_1,x_2,x_3]$），那么 R 中按分次逆字典序排列的 3 次单项式如下：

$$x_1^3 >_{r\ell} x_1^2x_2 >_{r\ell} x_1x_2^2 >_{r\ell} x_2^3 >_{r\ell} x_1^2x_3 >_{r\ell} x_1x_2x_3 >_{r\ell} x_2^2x_3 >_{r\ell} x_1x_3^2 >_{r\ell} x_2x_3^2 >_{r\ell} x_3^3.$$

请注意，在我们对分次逆字典序的定义中，1 次单项式（恰好就是 R 中的 n 变量）按 $x_1 >_{r\ell} x_2 >_{r\ell} > \cdots >_{r\ell} x_n$ 排列。因此，通过对这些变量进行置换，在含有 n 个变量的多项式环上，可以定义 $n!$ 种不同的分次逆字典序。一般来说，我们需要先指定 R 中 n 变量的线性序，然后在这些 1 次单项式的指定线性序基础上定义分次逆字典序。本文中，我们默认且不失一般性地假设，对于 R 上的所有单项式序 $<$，$x_1>x_2>\cdots>x_n$。

理想成员关系。设 $p\in R$ 是一个多项式，设 I 是 R的一个理想，并假设 $\mathcal{G}$ 是 I的一个 Gr¨obner 基。我们如何判断 p 是否包含于 I? 我们可以计算 p 相对于 $\mathcal{G}$的余式。当且仅当 p 包含于 I时，该余式为零；参见 [12, 第 2 章8]。

理想包含。设 J_1，J_2 是 R的理想，并假设 $\mathcal{G}_1$，$\mathcal{G}_2$ 分别是 J_1，J_2的 Gr¨obner 基。我们如何判断 $J_1\subseteq J_2$? 设 $\mathcal{G}_1=\{g_1,\ldots,g_k\}$。当且仅当对所有 $1\le i\le k$，g_i 相对于 $\mathcal{G}_2$ 都能化简为零时，$J_1\subseteq J_2$ 成立。

理想的交。设 J_1,J_2 是 R 的理想，并假设 $\mathcal{G}_1=\{g_1,\ldots,g_k\},\mathcal{G}_2=\{h_1,\ldots,h_\ell\}$ 分别是 J_1,J_2的 Gr¨obner 基。$J_1\cap J_2$的一个可能生成集 $\mathcal{G}$ 是什么? 我们可以按如下方式通过算法计算 $\mathcal{G}$：首先，构造一个“扩展的”多项式环 $R':=\Bbbk[x_1,\ldots,x_n,t]$，其中包含一个额外变量 t，使得 R 是 R' 的子环；然后考虑任意一个单项式序 $<$，使得任何被 t 整除的单项式都大于所有不被 t 整除的单项式。在新环 R' 中构造一个新理想 $K:=\langle tg_1,\ldots,tg_k,(1-t)h_1,\ldots,(1-t)h_\ell\rangle$，针对所述单项式序 $<$，计算 K 的 Gr¨obner 基 $\mathcal{G}'$，然后计算子集 $\mathcal{G}\subseteq\mathcal{G}'$，其中包含所有满足 $\mathrm{in}_<(g')$ 不被 t 整除的元素 $g'\in\mathcal{G}'$。这个最终集合 $\mathcal{G}$ 不仅是 $J_1\cap J_2$ 的生成集；事实上，它还是 $J_1\cap J_2$ 的 Gr¨obner 基。关于该算法为何有效的证明细节，请参见 [12,第 4 章。3]。

有关 Gr¨obner 基理论的更详细论述，包括它与计算代数中的算法以及交换代数中众多计算问题之间的丰富联系，请参见 [17,第 15]章。

A.3. 准素理想与初等分解

初等分解的概念源于对整数素因数分解思想的一种广泛推广，并对数论、群论和代数几何有着深远影响。例如，整数的素因数分解、有限阿贝尔群的分类，以及代数簇分解为不可约分支，都可以解释为三个看似不同的分解定理，而它们都是著名的拉斯克-诺特定理的特殊情形；参见 [17, 定理 3.10]，了解该定理完整的一般表述。

在我们的语境中，拉斯克-诺特定理指出，多项式环 R 中的每个理想都可以分解为有限多个“准素”理想的交。这种分解称为“初等分解”，目前已有多种算法可用于计算多项式环中理想的初等分解；参见 [18,21,65] 了解最先进的算法。在本文中，计算概念（即单项式理想）的初等分解是我们的代数机器推理框架中的关键环节。

A.3.1 究竟什么是主分解?

要严格定义主分解及相关概念，我们首先需要介绍一些更专业的代数术语。

素理想与准素理想。设 J 是 R的一个理想。如果 J 不是整个多项式环 R，即如果 $J\neq\langle 1\rangle$，我们就称 J 是真的。如果 J是满足下列条件的真理想，我们就称 J 是一个素理想：

如果 J_1，J_2 是 R 的理想，且 $J_1J_2\subseteq J$，那么$J_1\subseteq J$ 或 $J_2\subseteq J$ 成立（也可能两者都成立）。

若存在一个元素 $r\in R$ 满足 $r\notin J$，使得$P=\{p\in R|pr\in J\}$，则称 P 的素理想为 R 的关联素理想。如果 J恰有一个关联素理想，我们就称理想 J 为准素的；如果这个唯一的关联

素理想是 P，那么我们称 J是一个 P-准素理想。准素理想还有另一种有用的等价定义；参见 [17,命题 3。9]。若理想 J 是 R 的准素理想，则它是一个真理想，并满足以下条件：

如果 x，y 是 R 的元素，且满足 $xy \in J$，那么要么 $x \in J$，要么对于某个 $k \in \mathbb{Z}^+$，有 $y^k \in J$。

主分解与关联素理想。现在我们陈述拉斯克-诺特定理的一个（特殊情形）：

定理 A.7（[17,定理 3。10]）。R 的每个真理想 J 都可以写成有限多个准素理想 J_1、.....、J_k 的交 $J = J_1 \cap \cdots \cap J_k$，其中这些准素理想属于 R。此外，如果每个 J_i 都是 R 中某个素理想 P_i 的 P_i-准素理想，那么 J 的每个关联素理想都必定是这些素理想 P_1、.....、P_k 之一。

的一个 主分解 是将 J 分解为 R中有限多个准素理想的交。定理 A.7 指出，R 中的真理想总是存在主分解。请注意，一般而言，理想 J可能存在多个主分解。

考虑真理想 J 的一个主分解 $J = J_1 \cap \cdots \cap J_k$。准素理想 J_1、.....、J_k 称为此主分解的主分量。[2] 如果每个 J_i 都是某个素理想 P_i 的 P_i-准素理想，那么我们称 J_i 为（此主分解的） P_i-主分量。如果主分解 $J = J_1 \cap \cdots \cap J_k$ 中主分量的数量 k 在 J 的所有可能主分解中最小，那么我们称 $J = J_1 \cap \cdots \cap J_k$ 为极小主分解。在这种情况下，关联素理想 P_1、.....、P_k 必须两两不同（见 [17, 定理 3.10c]）。请注意，极小主分解通常并不唯一。

理想 $J \subseteq R$ 的根基 定义为

$$\sqrt{J} := \{r \in R | r^k \in J \text{ for some } k \in \mathbb{Z}^+\}.$$

下一个命题阐述了根基与主分解中主分量的关联素理想之间的关系。

命题 A.8. 设 J 是 R 的真理想，且 $J = J_1 \cap \cdots \cap J_k$ 是 J 的一个极小主分解。那么对每个 $1 \leq i \leq k$，根基 $\sqrt{J_i}$ 都是一个素理想，而主分量 J_i 是一个 $\sqrt{J_i}$-准素理想。此外，$\sqrt{J_1}$、.....、$\sqrt{J_k}$ 是 J 的所有互异关联素理想。

证明。设 J_1', J_2' 是 $\sqrt{J_i}$ 的理想，满足 $J_1'J_2' \subseteq \sqrt{J_i}$。任取非零元素 $j_1 \in J_1', j_2 \in J_2'$。根据根基的定义，我们有 $(j_1 j_2)^t \in J$ 对某个 $t \in \mathbb{Z}^+$ 成立。由于 J_i 是主理想，我们推知 $j_1^t \in J_i$ 或者对某个 $k \in \mathbb{Z}^+$ 有 $j_2^{tk} \in J_i$；这意味着 $j_1 \in \sqrt{J_i}$ 或 $j_2 \in \sqrt{J_i}$，也就是说 $\sqrt{J_i}$ 是一个素理想。因此，根据 [17,推论 2.12]，$\sqrt{J_i}$ 是包含 J_i的最小素理想，故 [17,命题 3.9] 表明 $\sqrt{J_i}$ 是 J_i的唯一关联素理想。最后，[17, 定理 3.9c] 完成证明。 □

注 A.9. 命题A.8 的一个重要推论是，尽管理想 J 在 R 中可能有多个不同的极小主分解 $J = J_1 \cap \cdots \cap J_k$，但任何极小主分解所对应的素理想集合始终相同；这个集合称为关联素理想集，它属于 J。

注 A.10。在计算机代数系统中，主分解通常表示为准素理想的列表。尽管已有算法可以计算多项式环中任意真理想的极小主分解[3]（例如，参见 [67] Macaulay2 中的几种实现），但我们提醒读者，有些计算机代数系统可能不支持计算主分解。

唯一极小主分解。在 J 是 R 的真单项式理想这一特殊情形下，J 的极小主分解具有特别好的性质：

定理 A.11（[4]；参见 [17, 例 3。11]）。设 J 是 R 的真单项式理想。那么存在一个极小主分解 $J = J_1 \cap \cdots \cap J_k$，满足以下条件：

- 每个主分量 J_i 都是单项式理想。
- 每个主分量 J_i 都是一个 P_i-准素理想，其中素理想 $P_i = \sqrt{J_i}$ 由 $\{x_1, \ldots, x_n\}$ 的一个子集生成。
- 每个主分量 J_i 都是所有可能的单项式 P_i-主分量中极大的。

此外，任何满足这些性质的极小主主分解 J，除主分量的重新排序外，都是唯一确定的。

简而言之，每个真单项式理想 J R 都有唯一的极小主分解，其主分量为极大的单项式主分量；这些主分量构成的唯一确定的集合 $\{J_1, \ldots, J_k\}$ 记作 (J)。

在我们的代数机器推理表述中（特别参见附录 A.1.2），定理A.11与命题 A.8结合，可得以下推论。

推论 A.12. 设 J 是 R 的一个概念，满足 $J \neq \langle 1 \rangle$，即 J 不是“可设想”概念。那么存在

[2]讨论主分量时，默认每个主分量都是相对于某个给定的主分解而言的。由于主分解并不唯一，若仅提及 J_i 是 J 的一个主分量，却不说明 J_i 所属主分解的上下文，就会产生歧义。

[3]现有用于计算初等分解的算法实现总是返回极小主分解。对于一个固定的理想 J，可能存在多个极小主分解，例如在 Macaulay2 中；参见 [25,67]。

有限多个互不相同的 R 的简单概念 P_1、……、P_k，它们由 J 唯一确定；并且存在概念 J 的一个主分解 $J = J_1 \cap \cdots \cap J_k$，即将其表示为 k 个概念 J_1、……、J_k 的交，其中每个根基 $\sqrt{J_i}$ 都是简单概念 P_i。此外，任何使 J_i 在所有根基为 $\sqrt{J_i} = P_i$ 的可能概念中取最大值的此类分解，都是唯一确定的。

基本概念的主分解。正如我们在附录A.1.2中所述，我们将 RPM 面板以代数方式编码为基本概念，即其极小实例均为平方自由。以下定理说明，基本概念具有特别好的性质：它们的唯一极小主分解中的主分量都是简单概念。

定理 A.13。设 J 是 R的一个基本概念。那么，存在一个主分解 $J = J_1 \cap \cdots \cap J_k$，将概念 J 表示为有限个互不相同的简单概念 J_1、……、J_k的交。此外，除简单概念 J_1、……、J_k的排列顺序不同外，这样的分解是唯一的。

证明。设 $\text{mingen}(J) = \{m_1, \ldots, m_k\}$。由于 $J \neq \langle 1 \rangle$，单项式生成元 m_1、……、m_k 中没有一个能是 1，因此它们的次数都为 ≥ 1。如果 m_1、……、m_k 的次数全都恰好为 1，那么根据定义，J 就是一个简单概念，结论显然成立。因此，下面假设存在某个极小单项式生成元 m_j，使得 $\deg(m_j) \geq 2$。这意味着我们可以将 $m_j = m'm''$写成若干单项式 m'、m'' 的形式，其中这些单项式属于 $\mathcal{M}_R$，且次数均为 ≥ 1。由于 J是一个基本概念，J 的所有极小单项式生成元都必须是平方自由的，因此 m_j 是一个平方自由单项式，这意味着 $\gcd(m', m'') = 1$。于是，通过定义 $\hat{\mathcal{G}} := \text{mingen}(J) \setminus \{m_j\}$，我们便可将 J 分解为一个交

$$J = \langle \hat{\mathcal{G}} \cup \{m'\} \rangle \cap \langle \hat{\mathcal{G}} \cup \{m''\} \rangle, \tag{4}$$

其中，该分解中的两个理想 $\langle \hat{\mathcal{G}} \cup \{m'\} \rangle$ 和 $\langle \hat{\mathcal{G}} \cup \{m''\} \rangle$ 都是平方自由单项式理想。

因此，对于任意平方自由单项式理想 J'在将 J 分解为平方自由单项式理想之交的任何分解中，只要 J' 包含一个次数 J' 为m 的极小单项式生成元 ≥ 2，我们就可以迭代地将 $J = J_1 \cap \cdots \cap J_k$进一步分解为两个严格更大的平方自由单项式理想之交。不断重复这一过程，就会得到分解J_1，其中 J_k，……，都是平方自由单项式理想，且其极小单项式生成元的次数均为1。根据定义，每个这样的 $\sqrt{J_i} = J_i$恰好是一个简单概念。因此，根据推论A.12，我们的断言得证。 □

A.3.2 代数机器推理的归纳偏置

定理 A.13有一个重要推论。它告诉我们，任何 J 属于 R 的基本概念，无论看起来多么“复杂”，都可以分解为简单概念的交。简单概念很容易理解，因为根据定义，它们是原始概念的和。

在所有可能的简单概念中（即其生成集合遍历 $\{x_1, \ldots, x_n\}$ 的所有可能非空子集），有一些对推理至关重要的“特殊”简单概念。这些“特殊”简单概念因任务而异，我们称之为属性概念。简单来说，属性概念是表示某种特定关注属性的概念。

在 RPM任务 中，回想一下，我们有表示“位置”“数量”“类型”“大小”和“颜色”的属性概念；这些简单概念按照其语义对原始实例进行分类，归入人类所称的属性。例如，“类型”属性概念 $J_{\text{type}} = \langle x_{\text{triangle}}, x_{\text{square}}, x_{\text{pentagon}}, x_{\text{hexagon}}, x_{\text{circle}} \rangle$ 是一个由 x_{triangle}，x_{square}，x_{pentagon}，x_{hexagon}，x_{circle}生成的简单概念，它们表示 RPM任务 中“类型”属性的所有可能实例。

属性概念的极小实例可以解释为“属性值”。这意味着，属性概念中包含的简单概念由单个“属性”的“属性值”生成，因此会继承相应属性概念的相同语义。例如，简单概念 $J := \langle x_{\text{triangle}}, x_{\text{pentagon}}, x_{\text{circle}} \rangle$ 包含于 J_{type} 中，并继承“类型”属性的语义。请注意，J_{type} 是一个对人类有意义的概念（表示几何实体可能的类型），因此 J 也会是一个对人类同样有意义的概念（表示几何实体的某些可能类型）。毕竟，J的极小实例都是同一“类型”属性的实例。

请注意，此类属性概念的识别因任务而异，由此产生的推理性能（例如，就提取的模式而言）将很大程度上取决于所识别的这些属性概念。特别是，我们对哪些简单概念属于属性概念的选择，将决定推理框架的归纳偏置：将一个基本概念分解为多个简单概念后，只有包含在属性概念中的简单概念才被视为“有意义”，我们会提取这类简单概念，将其作为代表对人类有意义的模式的概念。粗略地说，使用这种归纳偏置提取常见模式时，我们的推理框架会“偏向”那些包含在属性概念中的“更简单”的概念。

那么原始概念呢？通常，它们是不言自明且易于定义的。这是因为，要精确表述任何推理任务，首先必须识别可能的数据属性值（及其语义），而这自然会告诉我们原始

概念应该是什么。对于 RPM 任务，我们针对“颜色”“大小”“类型”和“位置”这 4 个属性设置了 4 个感知模块，因此，这 4 个属性的所有可能属性值自然构成了原始概念。

一般而言，对于视觉推理任务，感知模型能够预测的可能对象类别，自然会在我们的推理框架中充当原始概念。

A.4. 概念的代数直觉

A.4.1 定理证明3.1

为方便读者，我们重述定理3.1：

定理 3.1. 在 R中存在无穷多个概念，尽管 R中只有有限多个原始概念。此外，如果 $J \subseteq R$ 是一个概念，则以下结论成立：

(i) 除非 $J = \langle 0 \rangle$，否则 J 有无穷多个实例。*(ii)* J 有唯一的极小生成集，由有限多个实例构成，我们将其记为最小生成元(J)。*(iii)* 如果 $J \neq \langle 1 \rangle$，那么 J 有唯一一组相关概念 $\{P_1, \ldots, P_k\}$，以及唯一的极小主分解 $J = J_1 \cap \cdots \cap J_k$，使得每个 J_i 都是包含于 P_i中的一个概念，并且是在P_i 中所有作为概念的可能主分量中极大的一个。

证明。由于 $R = \Bbbk[x_1, \ldots, x_n]$，$R$ 中共有 n 个原始概念，即 $\langle x_1 \rangle$、.....、$\langle x_n \rangle$。由于 R 中所有单项式构成的集合 $\mathcal{M}_R$ 是无限的，且每个 $m \in \mathcal{M}_R$ 都生成一个概念 $\langle m \rangle$，因此 R 中有无限多个概念。为证明 (i)，注意到若 $J \neq \langle 0 \rangle$，则 J 必须至少包含一个单项式 $m \in \mathcal{M}_R$；因此，根据理想的定义，对于所有 $k \in \mathbb{N}$，J 都包含 mx_1^k，所以它有无限多个实例。对于 (ii)，参见命题 A.2。最后，(iii) 直接由推论 A.12 和命题 A.8 得出。 □

A.4.2 与人类推理以及人类对概念的理解相兼容

在认知科学中，概念是理解人类认知最根本的构念 [46, 53]。关于 [7,22,30,42,44,47,58] “概念”是什么（或应该是什么），存在许多相互竞争的理论。即便是看似无伤大雅的问题——这些不同理论所依据的基本原则彼此是否兼容——也曾引发激烈争论，至今仍未解决 [71]。

面对这样的背景，我们并不打算通过提出一种新的概念理论（即作为人类认知构念的概念理论）来参与这场争论。相反，我们希望说明，将概念定义为单项式理想，如何充分体现人类认知中概念各个方面的丰富内涵。对我们而言，

我们的代数方法植根于这样一种直觉：概念是可计算的，它们代表由其与其他概念之间的关系赋予可计算的意义 的抽象。

组合性。概念具有组合性 [30, 35]。一个概念可以由多个“更简单的”概念构成。不同的概念可以共享一个共同概念。在哲学和语言学中，这种概念的组合结构被广泛认为是人类学习新概念这一经验的重要方面 [23]。对于 RPM任务，我们用来编码 RPM面板的代数表示具有组合性。我们可以通过概念的和、积与交来组合概念 R 。 R。我们可以将一个概念分解为多个概念的交（通过主分解）。正如我们在第4节中所讨论，并在附录A.3.2中进一步阐述的那样，抽象模式的发现基于主分解；要使这一代数方法奏效，我们必然需要理想的“组合性”属性；参见附录C.1。

概念的本质。 在一些概念理论中， [22,30, 42], 存在一种模糊的概念，即概念的“本质”。我们可以把这种“本质”视为概念的定义（例如，著名的例子认为，“单身汉”这一概念与“未婚男子”同义）。然而，较新的概念理论则认为，许多对人类有意义的概念（例如“知识”和“真理”）很难用其他概念来精确定义 [20]。某些概念缺乏定义结构，由此产生了一种观点：概念可以根据其组成特征来定义。例如，“羽毛”和“翅膀”可以被视为“鸟”这一概念的特征。

必须仔细区分定义性特征和无关特征。例如，“鸣唱和啁啾”可以是“鸟”这一概念的特征，但并非定义性特征：鸟不必会鸣唱或啁啾，仍可被视为鸟。只有概念的定义性特征才会被视为该概念“本质”的一部分。然而，如何以有原则的方式提取概念的定义性特征，仍然是一个挑战；见 [37]。

对于定义为单项式理想的概念，是否存在类似的“本质”概念？根据定义，我们可以通过指定生成集，在 R 中构造一个概念；这与指定概念的特征非常类似。给定概念 J 在 R中的生成集 $\mathcal{G}$，我们可以计算集合 $\mathrm{mingen}(J)$，即J的极小单项式生成元集合。根据 $\mathrm{mingen}(J)$ 的唯一性（见命题A.2），这意味着我们可以将$\mathrm{mingen}(J)$中的生成元视为与该概念的定义性特征类似。属于 $\mathcal{G}$ 但不属于$\mathrm{mingen}(J)$ 的生成元对于生成 J 而言是冗余的，可以将其视为无关特征。

因此，我们可以将mingen(J) 视为概念 J的“本质”：实际上，mingen(J) 是一个“特殊的”有限实例集，它决定了 J 中所有无限多个可能实例；参见第3.1 节中的定理3.1.2

相比之下，在基于逻辑或集合论的情境中，并没有与 mingen(J) 相类似的概念：对于给定的逻辑表达式，先验地，并不存在一个“显著的”布尔型变量子集（出现在该逻辑表达式中），或一个子句子集，能够表示整个逻辑表达式。对于任意给定的无限集，如果我们对其元素一无所知，就没有自然的方法为其指定一个仍能表示整个无限集的“显著”有限子集。

具有部分定义的概念。在经典概念理论中，概念是根据实例成为该概念成员的必要且充分条件来刻画的 [7]。对这一经典理论的一项关键批评是，人类仍然能够运用具有部分定义的概念进行推理；自20世纪80年代以来，这一直是新古典概念理论的出发点 [46]。

在代数机器推理中，我们对概念的理解足以应对部分定义。以 RPM任务中的属性值“x_{square}”为例。请注意，概念 $\langle x_{\text{square}} \rangle$ 代表“正方形”，其数学定义为平面上的一种几何实体，具有四条等长边和四个直角。然而，解决 RPM任务并不需要完整定义。我们只需根据正方形有四条边这一条件给出一个部分定义；并未明确使用“边长相等”和“角度相等”等其他必要条件。“四条边”这一必要条件编码如下：（i）关系 $f_{\text{next}}(\langle x_{\text{triangle}} \rangle) = \langle x_{\text{square}} \rangle$；（ii）关系$f_{\text{next}}(\langle x_{\text{square}} \rangle) = \langle x_{\text{pentagon}} \rangle$；以及隐含的（iii）一种感知模型，能够将与$x_{\text{square}}$ 相关联的对象类别同 $\mathcal{A}_{\text{type}}$ 中其余变量相关联的其他对象类别区分开来。例如，即使我们不知道正方形具有四个直角这一定义性特征，并允许菱形成为概念 $\langle x_{\text{square}} \rangle$ 的一个实例，只要内角不是解决问题时的相关特征（RAVEN 和 I-RAVEN 数据集均属于这种情况），我们仍然可以解决 RPM任务。

我们的代数方法之所以具有灵活性，尤其是在处理部分定义时，关键在于我们可以自由地为原始概念赋予任意语义。具体而言，R中的原始概念不一定要符合传统意义上的“原始概念”。在认知科学中的经典概念理论里，原始概念主要基于人类感官知觉 [42]。相比之下，R 中原始概念的定义由我们决定，实际上取决于我们希望解决的具体推理任务，以及我们所采用的感知模型；参见附录A.3.2。

此外，与基于逻辑的方法不同，我们不必为所有原始概念“赋予”真值。例如，要确定实例 m 是否包含在概念 $J = \langle x_{\text{white}} x_{\text{square}} \rangle$中，我们不必知道 m 所表示的实体是小还是大。“位置”和“大小”属性值可以是未知的。要确定这一点，我们只需验证必要的条件，即 m 所表示的实体是白色的正方形。即使 J中的某些实例存在未知的属性值，我们仍然可以对所有实例进行推理。因此，我们的推理框架能够处理部分信息，并仍能对某些真值（真或假）未知的实例作出推断；这些真值表示实例是否属于各个原始概念。直观而言，我们不必为所有变量（表示实例是否属于原始概念的陈述）赋予真值，也能运用代数机器推理进行推理。

B. 实验细节补充

本节进一步介绍代数机器推理框架的实现细节，并给出更多实验结果，具体安排如下：

- 附录 B.1进一步介绍代数表示阶段的细节，包括属性概念的设计以及目标检测模块的性能。
- 附录 B.2 介绍如何对 RPM 实例“逆向”应用 4个不变性模块以生成答案的细节。
- 附录B.3 给出 4个不变性模块及答案生成过程的完整计算细节示例。
- 附录 B.4 详细介绍代数机器推理框架的消融研究，以分析答案选择任务中各个不变性模块的有效性。
- 附录 B.5 提供了 I-RAVEN 中每种 RPM 配置的生成答案示例 [26]。

B.1. RPM 任务的代数表示

B.1.1 代数对象的语义

属性概念。一个 RAVEN [83]（或 I- RAVEN [26]）数据集实例由一对文件构成：一个存储原始图像像素值的 NPZ 文件，以及一个 XML 文件，其中存储了给定答案和 16 个面板中各实体属性值的信息。如第3.2.1节所述，我们首先需要定义五个不相交的变量集合（分别表示五种属性），以便用代数方式编码 RPM 实例。为便于说明，本文仅考虑了示例中涉及的属性值。然而，为了

属性值集	变量标签
$\mathcal{A}_{\text{num}} := \{x_0, \ldots, x_8\}$	$x_{\text{one}}, x_{\text{two}}, \ldots, x_{\text{nine}}$
$\mathcal{A}_{\text{pos}} := \{x_9, \ldots, x_{31}\}$	$x_{(0.16,0.16,0.33)}, x_{(0.16,0.5,0.33)}, x_{(0.16,0.83,0.33)}, x_{(0.5,0.16,0.33)}, x_{(0.5,0.5,0.33)}$, $x_{(0.5,0.83,0.33)}, x_{(0.83,0.16,0.33)}, x_{(0.83,0.5,0.33)}, x_{(0.83,0.83,0.33)}$, $x_{(0.25,0.25,0.5)}, x_{(0.25,0.75,0.5)}, x_{(0.75,0.25,0.5)}, x_{(0.75,0.75,0.5)}$, $x_{(0.42,0.42,0.15)}, x_{(0.42,0.58,0.15)}, x_{(0.58,0.42,0.15)}, x_{(0.58,0.58,0.15)}$, $x_{(0.5,0.25,0.5)}, x_{(0.5,0.75,0.5)}, x_{(0.25,0.5,0.5)}, x_{(0.75,0.5,0.5)}, x_{(0.5,0.5,1.0)}, x_{\text{dummy}}$
$\mathcal{A}_{\text{type}} := \{x_{32}, \ldots, x_{36}\}$	$x_{\text{triangle}}, x_{\text{square}}, x_{\text{pentagon}}, x_{\text{hexagon}}, x_{\text{circle}}$
$\mathcal{A}_{\text{color}} := \{x_{37}, \ldots, x_{46}\}$	$x_{\#255}, x_{\#224}, x_{\#196}, x_{\#168}, x_{\#140}, x_{\#112}, x_{\#84}, x_{\#56}, x_{\#28}, x_{\#0}$
$\mathcal{A}_{\text{size}} := \{x_{47}, \ldots, x_{68}\}$	$x_{(0.6,0.15)}, x_{(0.7,0.15)}, x_{(0.8,0.15)}, x_{(0.9,0.15)}$, $x_{(0.4,0.33)}, x_{(0.5,0.33)}, x_{(0.6,0.33)}, x_{(0.7,0.33)}, x_{(0.8,0.33)}, x_{(0.9,0.33)}$, $x_{(0.4,0.5)}, x_{(0.5,0.5)}, x_{(0.6,0.5)}, x_{(0.7,0.5)}, x_{(0.8,0.5)}, x_{(0.9,0.5)}$, $x_{(0.4,1)}, x_{(0.5,1)}, x_{(0.6,1)}, x_{(0.7,1)}, x_{(0.8,1)}, x_{(0.9,1)}$,

表2. 与五个属性的属性值对应的五组变量。

属性	子序列	备注
位置	$\left[x_{(0.16,0.16,0.33)}, x_{(0.16,0.5,0.33)}, x_{(0.16,0.83,0.33)}, x_{(0.5,0.16,0.33)}, x_{(0.5,0.5,0.33)}, x_{(0.5,0.83,0.33)}, x_{(0.83,0.16,0.33)}, x_{(0.83,0.5,0.33)}, x_{(0.83,0.83,0.33)}\right]$, ,,	按.....拆分 宽度（最后一项）
	$\left[x_{(0.25,0.25,0.5)}, x_{(0.25,0.75,0.5)}, x_{(0.75,0.25,0.5)}, x_{(0.75,0.75,0.5)}\right]$	
	$\left[x_{(0.42,0.42,0.15)}, x_{(0.42,0.58,0.15)}, x_{(0.58,0.42,0.15)}, x_{(0.58,0.58,0.15)}\right]$	
	$x_{(0.5,0.25,0.5)}, x_{(0.5,0.75,0.5)}, x_{(0.25,0.5,0.5)}, x_{(0.75,0.5,0.5)}, x_{(0.5,0.5,1.0)}, x_{\text{dummy}}$	$f_{\text{next}}(\langle x \rangle, \Delta) = \langle x_{\text{dummy}} \rangle$
size	$\left[x_{(0.6,0.15)}, x_{(0.7,0.15)}, x_{(0.8,0.15)}, x_{(0.9,0.15)}\right]$	基于.....拆分 宽度（最后一个条目）
	$\left[x_{(0.4,0.33)}, x_{(0.5,0.33)}, x_{(0.6,0.33)}, x_{(0.7,0.33)}, x_{(0.8,0.33)}, x_{(0.9,0.33)}\right]$	
	$\left[x_{(0.4,0.5)}, x_{(0.5,0.5)}, x_{(0.6,0.5)}, x_{(0.7,0.5)}, x_{(0.8,0.5)}, x_{(0.9,0.5)}\right]$	
	$\left[x_{(0.4,1)}, x_{(0.5,1)}, x_{(0.6,1)}, x_{(0.7,1)}, x_{(0.8,1)}, x_{(0.9,1)}\right]$	

表3. 子序列 fo r 属性“位置”和“大小”，其中我们对变量进行循环排序，以捕捉序列 ial 属性值的信息 ues.

在所有 RPM 实例中使用相同的代数表示设置，我们将定义由所有可能属性值组成的变量集合。

在表2中，我们列出了全部 69 个变量（记作 x_0、.....、 x_{68}），并将其分为五个属性集。我们还将为这 69 个变量指定标签；这些标签是字符字符串，旨在便于人类理解各变量的含义。我们将交替使用索引和变量标签，例如 $x_0 = x_{\text{one}}$。请注意，最后一个变量 x_{dummy} 在 $\mathcal{A}_{\text{pos}}$ 中引入，是为了后续使用（与函数 f_{next} 配合），该函数将在组合不变性模块中使用，稍后会对此进行说明。对于其余原始实例，我们将在下文简要说明其含义：

- $\mathcal{A}$中的实例数量 表示给定面板中几何实体的数量。
- 在 $\mathcal{A}$位置中的实例表示与几何实体对应的子面板的位置，形式为 3 元组（x中心 y中心，宽度）。（XML 文件中提供的位置值采用 4 元组形式，即（x中心 y中心，宽度，高度）。由于每个子面板的宽度和高度始终相等，因此省略第 4 个元素。）
- $\mathcal{A}$类型中的实例表示几何实体的类型（或“形状”）。
- $\mathcal{A}$颜色中的实例表示几何实体的灰度像素值，其中 $x_{\#255}$ 表示“白色”，$x_{\#0}$ 表示“黑色”。
- $\mathcal{A}$大小中的实例表示子面板的绝对大小，形式为二元组（相对尺寸，宽度）。

（XML 文件中提供的大小值是实体相对于子面板大小的相对尺寸（即 0。4、.....、0。9）。）

令 $\mathcal{L}:=$ {数量，位置、类型、颜色、大小} 表示属性标签集合。然后，对于每个属性标签 $\ell \in \mathcal{L}$，我们可以定义属性概念 $J_\ell = \langle \mathcal{A}_\ell \rangle$。

组合不变性模中的函数 $f_{\textbf{next}}$。我们在第 3.3.1 节中描述过，对于每个属性标签 $\ell \in \mathcal{L}$，我们按循环序排列 $\mathcal{A}_\ell$ 中的值，并定义函数 $f_{\text{next}}(J|\Delta)$ 来表示“下一个”的含义。这里，Δ 表示步长，用于将生成元概念 J 中出现的任意变量 $x \in \mathcal{A}_\ell$ 映射到循环序下位于 x 之后第 Δ 个的变量（若 Δ 为负，则映射到 x 之前第 $|\Delta|$ 个变量）。每个 $\mathcal{A}_\ell$ 内部的循环序用于确保 $f_{\text{next}}(\langle x \rangle|\Delta)$ 始终与 $\langle x \rangle$ 属于同一属性概念。

对于“数量”、“类型”和“颜色”属性，可以按照表2所示的顺序，自然地排列变量。对于“位置”和“大小”属性，我们将变量序列拆分为多个子序列。在每个子序列中，我们定义一种循环序，使其符合人类直觉所理解的、变量标签语义中固有的顺序信息；参见表3。例如，$x_{(0.75,\,0.75,\,0.5)}$（表示右下方子面板）的“下一个”位置应循环回到 $x_{(0.25,\,0.25,\,0.5)}$（表示左上方子面板）；$x_{(0.9,0.5)}$ 的“下一个”大小应循环回到 $x_{(0.4,0.5)}$，等等。请注意，最后几个“位置”变量 $x_{(0.5,0.25,0.5)}$、.....、$x_{(0.5,0.5,1.0)}$ 不属于任何子序列，因为这些位置变量不存在明显的顺序信息。因此，我们引入一个新变量 x_{dummy}，使得对于任意位置变量 $x \in \{x_{(0.5,0.25,0.5)}, \ldots, x_{(0.5,0.5,1.0)}\}$，

$$f_{\text{next}}(\langle x \rangle|\text{any step } \Delta) = \langle x_{\text{dummy}} \rangle,$$
$$f_{\text{next}}(\langle x_{\text{dummy}} \rangle|\text{any step } \Delta) = \langle x_{\text{dummy}} \rangle.$$

函数 g 在二元算子不变性模块中。在第3.3.3节中，我们介绍了二元算子模块，用于根据给定的概念实值函数 g 以及给定的 Λ 二元算子集合提取数值模式。在 RAVEN 和 I-RAVEN 上的实验中，对于此不变性模块，我们只考虑了“数量”“颜色”和“大小”这三个属性，因为其余两个属性“类型”和“位置”没有明显的数值含义。对于“数量”属性，我们定义函数 g_{num}，将每个概念 J 映射为其唯一的极小生成集中生成元的数量 J：

$$g_{\text{num}}(J) := |\,\text{mingen}(J)|.$$

对于属性“颜色”，我们定义函数 g_{color}，用于提取表2中颜色值序列内的变量索引。请注意，我们仅将 g_{color} 应用于以下概念：J 对于

类别	GTs	Dets	召回率	AP
三角形	120978	120984	1.000	1.000
正方形	95967	95967	1.000	1.000
五边形	121722	121722	1.000	1.000
六边形	95017	95017	1.000	1.000
圆形	118578	118578	1.000	1.000

表4。I-RAVEN 验证集上“类型”属性的目标检测结果。“GTs”表示真实标注对象的数量，“Dets”表示检测到的对象数量，“AP”表示平均精度分数。

其颜色变量在 $\text{mingen}(J)$的所有生成元中均保持不变。类似地，对于属性“大小”，我们定义函数 g_{size}，用于提取表 3中的大小值子序列内的变量索引。

例如，考虑下面这个概念：

$$J = \left\langle \begin{matrix} x_{\text{two}} x_{(0.5,0.25,0.5)} x_{\text{square}} x_{\#255} x_{(0.6,0.5)}, \\ x_{\text{two}} x_{(0.5,0.75,0.5)} x_{\text{circle}} x_{\#255} x_{(0.8,0.5)} \end{matrix} \right\rangle.$$

我们可以计算

- $g_{\text{数量}}(J) = |\text{最小生成元}(J)| = 2$，这表明 J有两个极小单项式生成元；
- $g_{\text{颜色}}(J) = 0$，这表示颜色变量（在 mingen(J) 中的所有生成元之间均保持不变）是颜色值序列中的第一个元素；
- $g_{\text{大小}}(J) =$ None，因为 mingen(J) 中两个生成元的尺寸变量各不相同。

B.1.2 目标检测模块

类别	GTs	Dets	召回率	AP
#255	114701	114743	1.000	1.000
#224	48561	48570	1.000	1.000
#196	49626	49627	1.000	1.000
#168	50013	50016	1.000	1.000
#140	52634	52639	1.000	1.000
#112	51144	51144	1.000	1.000
#84	48390	48396	1.000	1.000
#56	48566	48568	1.000	1.000
#28	44464	44468	1.000	1.000
#0	44163	44167	1.000	1.000

表5. 在 I-RAVEN 验证集上评估的属性“颜色”目标检测结果。

在本小节中，我们将说明如何训练物体检测模块，以从原始 RPM图像中提取属性值。如3.2.2所述，我们分别训练了 4 个标准 RetinaNet 模型（每个模型均以 ResNet-50 为骨干网络），用于提取除“数量”属性之外的所有属性，

类别	GTs	Dets	召回率	AP
(0.16, 0.16, 0.33)	16646	16646	1.000	1.000
(0.16, 0.5, 0.33)	16803	16803	1.000	1.000
(0.16, 0.83, 0.33)	16531	16559	1.000	1.000
(0.5, 0.16, 0.33)	16652	16652	1.000	1.000
(0.5, 0.5, 0.33)	18455	18445	0.999	0.999
(0.5, 0.83, 0.33)	18354	18348	1.000	1.000
(0.83, 0.16, 0.33)	32000	32000	1.000	1.000
(0.83, 0.5, 0.33)	32000	32000	1.000	1.000
(0.83, 0.83, 0.33)	32000	32000	1.000	1.000
(0.25, 0.25, 0.5)	16761	16761	1.000	1.000
(0.25, 0.75, 0.5)	16851	16851	1.000	1.000
(0.75, 0.25, 0.5)	18420	18404	0.999	0.999
(0.75, 0.75, 0.5)	96000	96000	1.000	1.000
(0.42, 0.42, 0.15)	16753	16753	1.000	1.000
(0.42, 0.58, 0.15)	16611	16611	1.000	1.000
(0.58, 0.42, 0.15)	18128	18128	1.000	1.000
(0.58, 0.58, 0.15)	17992	17992	1.000	1.000
(0.5, 0.25, 0.5)	18081	18042	0.998	0.998
(0.5, 0.75, 0.5)	18135	18104	0.998	0.998
(0.25, 0.5, 0.5)	48648	48688	1.000	1.000
(0.75, 0.5, 0.5)	32000	32000	1.000	1.000
(0.5, 0.5, 1.0)	18441	18417	0.999	0.999

表6。在 I-RAVEN 验证集上评估的“位置”属性目标检测结果。

类别	GTs	Dets	召回率	AP
(0.6, 0.15)	23552	19372	0.823	0.823
(0.7, 0.15)	14447	11715	0.811	0.811
(0.8, 0.15)	14811	14805	1.000	1.000
(0.9, 0.15)	20084	20084	1.000	1.000
(0.4, 0.33)	32152	32152	1.000	1.000
(0.5, 0.33)	30635	30635	1.000	1.000
(0.6, 0.33)	30070	31111	1.000	1.000
(0.7, 0.33)	29539	29539	1.000	1.000
(0.8, 0.33)	30528	30528	1.000	1.000
(0.9, 0.33)	29332	29361	1.000	1.000
(0.4, 0.5)	35748	35748	1.000	1.000
(0.5, 0.5)	34319	34319	1.000	1.000
(0.6, 0.5)	32247	32247	1.000	1.000
(0.7, 0.5)	33439	33439	1.000	1.000
(0.8, 0.5)	33002	33002	1.000	1.000
(0.9, 0.5)	32357	32365	1.000	1.000
(0.4, 1.0)	5455	5455	1.000	1.000
(0.5, 1.0)	5829	5829	1.000	1.000
(0.6, 1.0)	5336	5336	1.000	1.000
(0.7, 1.0)	26793	26793	1.000	1.000
(0.8, 1.0)	25638	25636	1.000	1.000
(0.9, 1.0)	26949	26949	1.000	1.000

表7. 在 I-RAVEN 验证集上评估的“大小”属性目标检测结果。

“数量”属性可以通过统计一个面板内预测出的边界框数量直接推断得出。

RAVEN/I-RAVEN 数据集通常包含 70,000 个 RPM 实例，均匀分布在 7 种配置中。在每种配置中，10,000 个 RPM 实例会随机划分为 6,000 个训练实例、2,000 个验证实例和 2,000 个测试实例。在我们的实验中，我们仅使用训练集的 10%，即在 4,200 个 RPM 实例上训练目标检测模块（每种配置 600 个）。每个 RPM 实例都可拆分为 16 张大小为 160×160 的灰度图像，分别对应题目矩阵中的 8 个面板和 8 个答案选项。每张图像包含一个或多个几何实体，我们将对其进行检测。每个几何实体的真实标签和边界框可从给定的 XML 文件中获取。

我们使用`MMDetection` 软件包 [10] 进行标准训练，采用其基于残差网络-50骨干网络的 RetinaNet 模型实现。所有超参数均设为默认值，但以下情况除外：（i）训练期间，所有属性的初始学习率均为 0.003；（ii）在推理阶段，“类型”的置信度阈值设为 0.99，其他 3 个属性的阈值设为 0.95。所有 4 个模块均训练 12 个训练轮次。在包含 14,000 个样本的验证集上得到的评估结果

配置	RAVEN	I-RAVEN
中心	32000	32000
2×2网格	32000	32000
3×3网格	31996	31986
O-IC	23290	23617
O-IG	31988	31987
L-R	31999	31998
U-D	32000	32000
平均比例	96.10%	96。24%

表8。通过目标检测模块提取属性值的感知结果，在 RAVEN 和 I-RAVEN 上进行评估，以每种配置中正确检测出的面板数量（共 $2,000 \times 16$ 个面板）以及正确检测面板的总体比例表示。

个实例（每种配置 2,000 个）见表4-7。

根据上表，针对“类型”属性的目标检测模块在 552,262 个对象中仅出现 6 次错误，表现显著优于针对其他 3 个属性的目标检测模块。因此，我们将“类型”模块用作“锚点”，以整合全部 4 个模块得到的检测结果。具体而言，我们首先

从“类型”检测模块中获取置信度高于 0.99 的检测对象列表。每个对象的预测结果都可以表示为一对（类型标签，边界框）。对于每个检测到的“类型”对象，我们接着从其他 3 个检测模块中获取置信度高于 0.8 的检测结果列表；对于每个列表，我们选择与“类型”模块的锚定边界框相比交并比（IoU）得分最高的（标签，边界框）对。我们假设这 3 个选定的对与“类型”模块生成的对应对所代表的实体相关联。随后，我们可以收集 4 对密切相关的（标签，边界框），每一对都表示一个由某个检测模块返回的检测对象，并将这些对中的标签合并为从单个实体提取的属性值。最后，我们将基于 I-RAVEN 训练的 4 个目标检测模块应用于 RAVEN 和 I-RAVEN 数据集，从原始 RPM 测试图像中提取属性值。评估结果见表8。

B.2.“逆向”不变性模块与答案生成的完整算法细节

在本小节中，我们将详细说明如何根据从前两行以及第 3 行前两个面板中提取的常见模式，为 RPM 实例生成答案。非正式地说，对于每个不冲突的模式对 $(K,\ \check{}\mathbf{J}) \in \mathcal{P}_{1,\ 2}(\mathbf{J}) := \mathcal{P}_1^{(\text{all})}(\mathbf{J}) \cap \mathcal{P}_2^{(\text{all})}(\mathbf{J})$，我们将对第 3 行 $\check{}\mathbf{J}_3 := [\check{}J_{3,\ 1},\ \check{}J_{3,\ 2}]$“逆向”应用相应的不变性模块。回顾如下：

- $\mathcal{P}_{(\text{所有})\ i}(\mathbf{J})$ 表示从概念矩阵$\mathbf{J}$的第 i 行中提取的模式对集合；$\check{}$
- $\mathbf{J}$ 表示扩展列表中的一个概念矩阵$[\mathbf{J}\ \bar{}\mathbf{J}_{(p_1)},\ \hat{}\mathbf{J}_{(p_1)} \ldots,\ \bar{}\mathbf{J}_{(p_k)},\ \hat{}\mathbf{J}_{(p_k)}]$;
- K 表示从$\mathbf{J}$的前两行中提取的、针对某一属性的共同模式。

我们首先介绍每个“逆向”不变性模块的算法细节。

B.2.1 “逆向”内部不变性模块

算法 3“逆向”内部不变性模块。

输入： 一个模式 ℓ 和第 3 行 $\check{}\mathbf{J}_3 := [\check{}J_{3,1},\ \check{}J_{3,2}]$。

$I_{\text{out}} \leftarrow \langle 0 \rangle$。
$\mathcal{I} \leftarrow \text{pd}(\check{}J_{3,1} + \check{}J_{3,\ 2}) \cap \text{pd}(\check{}J_{3,1} \cap \check{}J_{3,\ 2})$。
对 $I \in \mathcal{I}$ **执行**
  如果 $I \subseteq \langle \mathcal{A}_\ell \rangle$ **则**
    $I_{\text{out}} \leftarrow I$.
返回 I_{out}.

给定一个概念序列 $J_1, \ldots, J_k$，我们用 $J_+ :=$ 表示这些概念的和与交。$J_1 + \cdots + J_k$ 和 $J_\cap := J_1 \cap \cdots \cap J_k$。回顾一下，这个内部不变性模块会提取这样的模式：对于某个属性，J_i 中实体对应的属性值集合在所有 i 中保持不变。此类提取的模式集合如下：

$$\mathcal{P}_{\text{intra}}([J_1 \ldots J_k]) := \{\text{属性} \in \mathcal{L} \mid \exists I \in \text{pd}(J_+) \cap \text{pd}(J_\cap)\ I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle\}。$$

因此，对于通过此不变性模块提取的模式对（K，$\check{}\mathbf{J}$），共同模式 K 按定义是一个属性标签 $\ell \in \mathcal{L}$。属性 ℓ 的取值应在$J_{3,\ 1}$、$J_{3,\ 2}$ 和$J_{3,\ 3}$之间保持不变。关于如何计算由$J_{3,\ 3}$对应的属性变量生成的理想，请参见算法3的伪代码。

请注意，算法 3可以自动筛除那些$\check{}$与第 3 行冲突的模式 $\mathbf{J}_3$,，无需任何人工检查。具体而言，如果输入模式 $\ell\check{}$与输入的第 3 行冲突$\mathbf{J}_3$，算法3 的输出将为零理想（$I_{\text{out}} = \langle 0 \rangle$）。对于其余 3 个“逆向”不变性模块，我们也采用相同的机制来自动筛除冲突模式，即存在冲突模式时，输出将为 $I_{\text{out}} = \langle 0 \rangle$。

B.2.2 “逆向”交互不变性模块

算法 4“逆向”交互不变性模块。

输入： 一个模式 $(\ell, \mathcal{I})$ 和第 3 行 $\check{}\mathbf{J}_3 := [\check{}J_{3,1},\ \check{}J_{3,2}]$。

$I_{\text{out}} \leftarrow \langle 0 \rangle$。
计算 $\mathcal{I}'$ 在 (5) 中。
如果 $\mathcal{I} - \mathcal{I}' \neq \varnothing$ 且 $\mathcal{I} \cap \mathcal{I}' \neq \varnothing$ **那么**
  $I_{\text{out}} \leftarrow \sum_{I \in \mathcal{I} - \mathcal{I}'} I$。
返回 I_{out}。

回顾一下，通过交互不变性模块提取的模式集合如下：

$$\mathcal{P}_{\text{inter}}([J_1, \ldots, J_k]) := \left\{ (\text{attr}, \mathcal{I}) \middle| \begin{array}{c} \mathcal{I} \subseteq \text{pd}(J_\cap) - \text{pd}(J_+), \\ \text{attr} \in \mathcal{L}, I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle\ \forall I \in \mathcal{I} \end{array} \right\},$$

对于通过此不变性模块提取的模式对（K，$\check{}\mathbf{J}$），共同模式 $K = (\ell,\ \mathcal{I})$ 是一个有序对，其第一个元素 ℓ 是属性标签，第二个元素 $\mathcal{I}$ 是包含在属性概念 $\langle \mathcal{A}_\ell \rangle$ 中的一组理想。自然地，我们将根据集合差 $\mathcal{I} - \mathcal{I}'$ 推断 $J_{3,\ 3}$ 的属性值，其中 $\mathcal{I}'$ 定义如下：

$$\mathcal{I}' = \left\{ I \middle| I \in \left(\text{pd}(\check{}J_{3,1} \cap \check{}J_{3,2}) - \text{pd}(\check{}J_{3,1} + \check{}J_{3,2}) \right), I \subseteq \langle \mathcal{A}_\ell \rangle \right\}. \tag{5}$$

请注意，这个 $\mathcal{I}'$ 的定义方式与$\check{}$我们对 $\mathcal{P}_{\text{inter}}$ 的定义方式类似（限于$\mathbf{J}$ 的第 3 行和属性 ℓ）。摘要见算法4 。

B.2.3 “逆向”组合不变性模块

算法 5“逆向”组合不变性模块。

输入： 一个模式 $(\ell,\ f_{\text{next}}|_{\Delta=\Delta^*})$ 和第 3 行 $\check{\mathbf{J}}_3 := [\check{J}_{3,1},\ \check{J}_{3,2}]$。

$I_{\text{out}} \leftarrow \langle 0\rangle$。
$\mathcal{I} \leftarrow \text{pd}(f^2_{\text{next}}(\check{J}_{3,\ 1}|\Delta^*)) \cap \text{pd}(f_{\text{next}}(\check{J}_{3,\ 2}|\Delta^*))$。
对于 $I \in \mathcal{I}$ **执行**
  如果 $I \in \langle \mathcal{A}_\ell \rangle$ **则**
    $I_{\text{out}} \leftarrow I$。
返回 I_{out}。

给定一个将概念映射到概念的函数 f，组合不变性模块会从新的概念序列中的不变属性值提取模式：

$$[J'_1, \ldots, J'_k] = [f^{k-1}(J_1), f^{k-2}(J_2), \ldots, f(J_{k-1}), J_k].$$

此类模式的集合定义如下：

$$\mathcal{P}_{\text{comp}}([J_1, \ldots, J_k]) := \left\{ (\text{attr}, f) \;\middle|\; \begin{array}{c} \exists I \in \bigcap_{i=1}^{k} \text{pd}(f^{k-i}(J_i)), \\ \text{attr} \in \mathcal{L}, I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle \end{array} \right\}.$$

每种模式 $K = (\ell,\ f)$ 可作如下解释：属性 ℓ 的属性值在新的概念序列中保持不变。换言之，对于原序列中任意一对相邻概念 $[J_i, J_{i+1}]$，$i = 1$，……，$k-1$，属性 ℓ 的属性值在 $[f(J_i)\ J_{i+1}]$中应保持不变。此性质提供了一种自然的方法：给定序列中前一个概念 J_{i+1}的属性值，计算概念 J_i 的属性值；见算法5。请注意，对于 RPM任务，我们引入 $f_{\text{next}}|_\Delta$ 函数，以便组合不变性模块提取概念矩阵一行中的序列信息。

B.2.4 “逆”二元运算符不变性模块

算法 6“逆”二元运算符不变性模块。

输入： 一个模式 $(\overline{\oslash},\ g_\ell,\ \Lambda)$，以及第 3 行 $\check{\mathbf{J}}_3 := [\check{J}_{3,1},\ \check{J}_{3,2}]$。

$I_{\text{out}} \leftarrow \langle 0\rangle$。
$t \leftarrow g_\ell(\check{J}_{3,\ 1}) \oslash_1 g_\ell(\check{J}_{3,\ 2})$。 // 预期 $g_\ell(J_{3,3})$ 值
$x \leftarrow t$ 个变量，对应属性 se- 属性 ℓ的序列（或子序列）。
$I_{\text{out}} \leftarrow \langle x \rangle$。
返回 I_{out}。

给定一个定义在概念上的实值函数 g 和一组二元运算符 Λ，二元运算符不变性模块可从数量序列中提取模式。

$g(J_1) \ldots, g(J_k)$，描述如下：

$$\mathcal{P}_{\text{binary}}(\mathbf{J}_i) := \left\{ (\overline{\oslash}, g, \Lambda) \;\middle|\; \begin{array}{c} \overline{\oslash} = [\oslash_1, \ldots, \oslash_{k-2}],\ \oslash_i \in \Lambda, \\ g(J_1)\ \oslash_1 \cdots \oslash_{k-2}\ g(J_{k-1}) = g(J_k) \end{array} \right\}.$$

换句话说，我们可以将每种模式表示为一个方程：在序列中的数量之间插入从 Λ中选取的二元运算符即可得到该方程。

对于一个新给定的概念序列 $J'_1, \ldots, J'_{k-1}$ 和模式 $K = ([\oslash_1, \ldots, \oslash_{k-2}], g, \Lambda)$，我们可以直接根据该模式计算出预期的 $g(J'_k)$值：$g(J'_k) = g(J'_1)\ \oslash_1 \cdots \oslash_{k-2}\ g(J'_{k-1})$。然后，我们可以根据函数 g的含义推断出有关概念 J'_k 的信息。

如第B.1节所述，我们引入了函数 $g_{\text{num}}(J)$ 以提取 J 的极小单项式生成元数量（这也表示相应面板中的实体数量），并引入函数 g_{color}（分别是 g_{size}）以提取“颜色”值序列（分别是“大小”值子序列）中的变量索引。因此，我们可以直接根据 $g_\ell(J'_k)$的值，推断出 J'_k 对应的属性变量，适用于$\ell \in$ [“数量”，“颜色”，“大小”]。摘要见算法6 。

算法7答案生成过程详解。**输入：概念矩阵 J** $= [J_{1,1}, \ldots, J_{3,2}]$。1：计算 comPos(**J**) $= [p_1, \ldots, p_k]$。2：$I_{\text{out}} \leftarrow \langle 0\rangle$。3：**对于** $p_i \in [p_1, \ldots, p_k]$ **执行**// 使用固定的“数量”和“位置”进行初始化4：$I^{(i)} \leftarrow \langle x_{k-1} x_{p_i} \rangle$。5：**对于** $K \in \mathcal{P}(\bar{\mathbf{J}}^{(p_i)}{}_1) \cap \mathcal{P}(\bar{\mathbf{J}}^{(p_i)}{}_2)$ **执行**6：$I^* \leftarrow$ 根据给定的输入 K 和 $\bar{\mathbf{J}}^{(p_i)}{}_3$，使用算法3-6之一生成输出理想。7：**如果** $I^* \neq \langle 0\rangle$ **则**8：$I^{(i)} \leftarrow I^{(i)} \cap I^*$9：从 mingen($I^{(i)}$) 中随机选择生成元

10: $I_{\text{out}} \leftarrow I_{\text{out}} + \langle m \rangle$。 11：$\mathcal{G} \leftarrow \varnothing$12: **对于** $m \in$ mingen(I_{out})**执行**13: $m' \leftarrow m$14: **当** $\exists \ell \in \mathcal{L}$满足以下条件：$x$ 不能整除 $m\ \forall x \in \mathcal{A}_\ell$ **执行**15: $x' \leftarrow$ 一个从 $\mathcal{A}_\ell$中随机选取的变量。 16: $m' \leftarrow m'x'$。 17: $\mathcal{G} \leftarrow \mathcal{G} \cup \{m'\}$。 18：$I_{\text{out}} \leftarrow \langle \mathcal{G} \rangle$。 19: **返回** I_{out}。

B.2.5 答案生成的整体流程

在 RAVEN/I-RAVEN 数据集中，面板中某些属性的取值可能是有意随机设置的。

方法	平均准确率	中心	2×2G	3×3G	O-IC	O-IG	L-R	U-D
包含全部4个不变性模块	93.2	99.5	89.6	89.7	99.6	74.7	99.7	99.5
不含内部不变性模块	63.7	63.5	71.6	73.4	60.2	49.8	63.2	64.0
不含交互不变性模块	59.0	64.3	56.8	59.2	59.2	43.5	64.5	65.8
不使用组合不变性模块	69.5	73.0	65.9	64.3	78.1	61.5	72.9	71.2
不使用二元运算符不变性模块	74.6	75.8	69.3	67.1	86.2	70.2	77.3	76.7

表 9. 在 I-RAVEN 数据集上评估我们的代数机器推理框架所得的详细消融研究结果，包括总体答案选择准确率以及全部七种 RPM 配置各自的准确率。

例如，为避免冲突，“数量”和“位置”之间不会同时存在规则，因此，在某些配置（如 2×2网格、3×3网格和外内网格）中，一旦实体数量确定，面板中的实体位置就可能完全随机。其他属性的取值也可能是随机的；例如，见图5 。对于属性值如此随机的概念矩阵，很难提取其中的模式，也几乎不可能生成与给定正确答案相符的答案。因此，对于任意给定的 RPM 实例，我们将通过聚焦于每个单独的位置来简化答案生成过程。

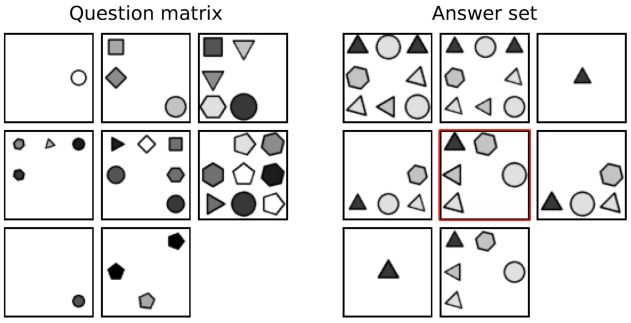


图 5. 一个 RPM 实例示例，其中每个面板内的“位置”“类型”和“颜色”属性取随机值。正确答案以红框标出。

回顾一下，对于表示给定 RPM 实例的概念矩阵 $\mathbf{J}$ ，我们从扩展概念矩阵列表中的每个概念矩阵 $\mathbf{J}$ 中提取共同模式，即 $[\mathbf{J}\ \bar{}\mathbf{J}^{(p_1)}, \hat{}\mathbf{J}^{(p_1)} \ldots, \bar{}\mathbf{J}^{(p_k)}, \hat{}\mathbf{J}^{(p_k)}]$；参见第 3.3.4 节。这里，$p_1, \ldots, p_k$ 是我们从与 $\mathbf{J}$ 对应的题目矩阵的 8 个面板中提取的公共位置，而所有 $\bar{}\mathbf{J}^{(p_t)}, \hat{}\mathbf{J}^{(p_t)}$ 定义如下：

- 每个概念 $\bar{J}_{(p_t)i,\ j}$ 在 $\bar{}\mathbf{J}_{(p_t)}$ 由 $J_{i,\ j}$ 中唯一一个可被 p_t 整除的生成元生成；
- 每个概念 $\hat{}J_{(p_t)i,\ j}$ 在 $\hat{}\mathbf{J}_{(p_t)}$ 由 $J_{i,\ j}$ 中所有不可被 p_t整除的生成元生成。

换言之，对于共同位置 p_t，原题目矩阵中的每个面板都会拆分为两个面板，其中一个面板仅包含位置 p_t 上的实体（由对应的概念表示

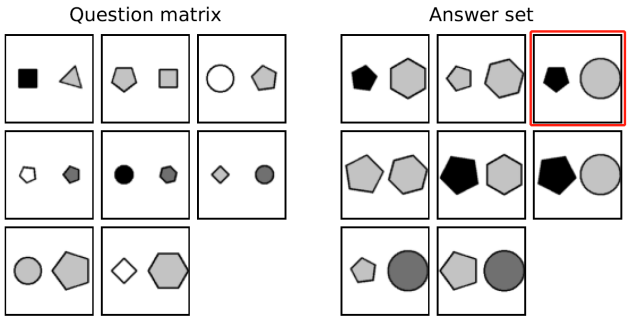


图 6. I-RAVEN 数据集中的一个 RPM 实例示例。正确答案用红框标出。（这与图 1相同；为方便起见，我们将其放在这里。）

，即 $\bar{}\mathbf{J}^{(p_t)}$），另一个面板则包含所有其余实体（由对应的概念，即 $\hat{}\mathbf{J}^{(p_t)}$ 表示）。在答案生成任务中，我们只考虑前文讨论的所有 $\bar{}\mathbf{J}^{(p_i)}$ 。直观而言，我们可以分别提取原题目矩阵各面板中受限于每个单独共同位置的模式。由于属性“数量”和“位置”在每个 $\bar{}\mathbf{J}^{(p_i)}$ 内都是固定的，因此我们只需提取其余 3 个属性的模式。详情请参见算法 7 。请注意，在算法 7 的第 6 行中，生成输出理想 I^*时，要对输入 K 和 $\bar{}\mathbf{J}^{(p_i)}$ 3 应用的具体算法（算法 3-6）由 K的语法唯一确定。

B.3. 不变性模块示例

考虑图 6所示的 RPM 实例。在本小节中，我们将提供该 RPM 实例的具体计算细节，以展示如何使用不变性模块提取模式并生成正确答案。

示例 1（内部不变性模块）。考虑图6中的第一行。通过目标检测模块提取每个面板中实体的属性值，我们可以

得到如下代数表示：

$$\mathbf{J}_1 := [J_{1,1}, J_{1,2}, J_{1,3}],$$
$$J_{1,1} = \langle x_{\text{two}} x_{\text{left}} x_{\text{square}} x_{\text{black}} x_{\text{avg}}, x_{\text{two}} x_{\text{right}} x_{\text{triangle}} x_{\text{gray}} x_{\text{avg}} \rangle,$$
$$J_{1,2} = \langle x_{\text{two}} x_{\text{left}} x_{\text{pentagon}} x_{\text{gray}} x_{\text{avg}}, x_{\text{two}} x_{\text{right}} x_{\text{square}} x_{\text{gray}} x_{\text{avg}} \rangle,$$
$$J_{1,3} = \langle x_{\text{two}} x_{\text{left}} x_{\text{circle}} x_{\text{white}} x_{\text{avg}}, x_{\text{two}} x_{\text{right}} x_{\text{pentagon}} x_{\text{gray}} x_{\text{avg}} \rangle.$$

We first compute $\text{pd}(J_+)$ and $\text{pd}(J_\cap)$.

$$\begin{aligned}
\text{pd}(J_+) = \{&\langle x_{\text{two}} \rangle, \langle x_{\text{avg}} \rangle, \langle x_{\text{left}}, x_{\text{right}} \rangle, \langle x_{\text{white}}, x_{\text{gray}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{triangle}}, x_{\text{square}}, x_{\text{pentagon}}, x_{\text{circle}} \rangle, \langle x_{\text{left}}, x_{\text{gray}} \rangle, \\
&\langle x_{\text{square}}, x_{\text{circle}}, x_{\text{gray}} \rangle, \langle x_{\text{square}}, x_{\text{white}}, x_{\text{gray}} \rangle \\
&\langle x_{\text{circle}}, x_{\text{gray}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{left}}, x_{\text{triangle}}, x_{\text{square}}, x_{\text{pentagon}} \rangle \\
&\langle x_{\text{right}}, x_{\text{square}}, x_{\text{pentagon}}, x_{\text{circle}} \rangle \\
&\langle x_{\text{right}}, x_{\text{square}}, x_{\text{pentagon}}, x_{\text{white}} \rangle \\
&\langle x_{\text{right}}, x_{\text{pentagon}}, x_{\text{circle}}, x_{\text{black}} \rangle \\
&\langle x_{\text{right}}, x_{\text{pentagon}}, x_{\text{white}}, x_{\text{black}} \rangle \\
&\langle x_{\text{triangle}}, x_{\text{square}}, x_{\text{pentagon}}, x_{\text{white}} \rangle \},
\end{aligned}$$

$$\begin{aligned}
\text{pd}(J_\cap) = \{&\langle x_{\text{two}} \rangle, \langle x_{\text{avg}} \rangle, \langle x_{\text{left}}, x_{\text{right}} \rangle, \langle x_{\text{gray}} \rangle, \\
&\langle x_{\text{triangle}}, x_{\text{square}} \rangle, \langle x_{\text{square}}, x_{\text{pentagon}} \rangle, \\
&\langle x_{\text{pentagon}}, x_{\text{circle}} \rangle, \langle x_{\text{left}}, x_{\text{triangle}} \rangle, \langle x_{\text{left}}, x_{\text{square}} \rangle, \\
&\langle x_{\text{left}}, x_{\text{pentagon}} \rangle, \langle x_{\text{right}}, x_{\text{square}} \rangle, \langle x_{\text{right}}, x_{\text{pentagon}} \rangle, \\
&\langle x_{\text{right}}, x_{\text{circle}} \rangle, \langle x_{\text{right}}, x_{\text{white}} \rangle, \langle x_{\text{right}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{triangle}}, x_{\text{black}} \rangle, \langle x_{\text{pentagon}}, x_{\text{white}} \rangle \}.
\end{aligned}$$

对于那些包含在属性概念中的主分量，我们用蓝色将其标出。这些概念被认为对 RPM任务“有意义”。请注意，大多数主分量并不包含在任何属性概念中。

接下来，我们计算集合交集 $\text{pd}(J_+) \cap \text{pd}(J_\cap) = \{\langle x_{\text{two}} \rangle, \langle x_{\text{avg}} \rangle, \langle x_{\text{left}}, x_{\text{right}} \rangle\}$，其含义是所有面板都包含两个大小适中的实体，一个位于左侧，另一个位于右侧。

回想一下，内部不变性模块的输出为：

$$\mathcal{P}_{\text{intra}}([J_1 \dots J_k]) := \{\text{attr} \in \mathcal{L} \mid \exists I \in \text{pd}(J_+) \cap \text{pd}(J_\cap), I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle\}.$$

因此，我们有

$$\mathcal{P}_{\text{intra}}(\mathbf{J}_1) = \{\text{“size”}, \text{“number”}, \text{“position”}\}.$$

示例2（交互不变性模块）。考虑第二行$\mathbf{J}_2$ 在图6。类似地，我们首先得到$\mathbf{J}_2$的代数表示：

$$\mathbf{J}_2 := [J_{2,1}, J_{2,2}, J_{2,3}],$$
$$J_{2,1} = \langle x_{\text{two}} x_{\text{left}} x_{\text{pentagon}} x_{\text{white}} x_{\text{small}}, x_{\text{two}} x_{\text{right}} x_{\text{pentagon}} x_{\text{dgray}} x_{\text{small}} \rangle,$$
$$J_{2,2} = \langle x_{\text{two}} x_{\text{left}} x_{\text{circle}} x_{\text{black}} x_{\text{small}}, x_{\text{two}} x_{\text{right}} x_{\text{hexagon}} x_{\text{dgray}} x_{\text{small}} \rangle,$$
$$J_{2,3} = \langle x_{\text{two}} x_{\text{left}} x_{\text{square}} x_{\text{gray}} x_{\text{small}}, x_{\text{two}} x_{\text{right}} x_{\text{circle}} x_{\text{dgray}} x_{\text{small}} \rangle.$$

接下来，我们按如下方式计算 $\text{pd}(J_+)$,$\text{pd}(J_\cap)$：

$$\begin{aligned}
\text{pd}(J_+) = \{&\langle x_{\text{two}} \rangle, \langle x_{\text{small}} \rangle, \langle x_{\text{left}}, x_{\text{right}} \rangle, \\
&\langle x_{\text{white}}, x_{\text{gray}}, x_{\text{dgray}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{square}}, x_{\text{pentagon}}, x_{\text{hexagon}}, x_{\text{circle}} \rangle, \\
&\langle x_{\text{left}}, x_{\text{dgray}} \rangle, \langle x_{\text{left}}, x_{\text{pentagon}}, x_{\text{hexagon}}, x_{\text{circle}} \rangle, \\
&\langle x_{\text{right}}, x_{\text{square}}, x_{\text{pentagon}}, x_{\text{circle}} \rangle, \\
&\langle x_{\text{right}}, x_{\text{square}}, x_{\text{pentagon}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{right}}, x_{\text{square}}, x_{\text{circle}}, x_{\text{white}} \rangle, \\
&\langle x_{\text{right}}, x_{\text{square}}, x_{\text{white}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{right}}, x_{\text{pentagon}}, x_{\text{circle}}, x_{\text{gray}} \rangle, \\
&\langle x_{\text{right}}, x_{\text{pentagon}}, x_{\text{gray}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{right}}, x_{\text{circle}}, x_{\text{white}}, x_{\text{gray}} \rangle, \\
&\langle x_{\text{right}}, x_{\text{white}}, x_{\text{gray}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{square}}, x_{\text{pentagon}}, x_{\text{circle}}, x_{\text{dgray}} \rangle, \\
&\langle x_{\text{square}}, x_{\text{pentagon}}, x_{\text{dgray}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{square}}, x_{\text{circle}}, x_{\text{white}}, x_{\text{dgray}} \rangle, \\
&\langle x_{\text{square}}, x_{\text{white}}, x_{\text{dgray}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{pentagon}}, x_{\text{hexagon}}, x_{\text{circle}}, x_{\text{gray}} \rangle, \\
&\langle x_{\text{pentagon}}, x_{\text{circle}}, x_{\text{gray}}, x_{\text{dgray}} \rangle, \\
&\langle x_{\text{pentagon}}, x_{\text{gray}}, x_{\text{dgray}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{circle}}, x_{\text{white}}, x_{\text{gray}}, x_{\text{dgray}} \rangle, \},
\end{aligned}$$

$$\begin{aligned}
\text{pd}(J_\cap) = \{&\langle x_{\text{two}} \rangle, \langle x_{\text{small}} \rangle, \langle x_{\text{left}}, x_{\text{right}} \rangle, \langle x_{\text{pentagon}} \rangle, \\
&\langle x_{\text{square}}, x_{\text{circle}} \rangle, \langle x_{\text{hexagon}}, x_{\text{circle}} \rangle, \\
&\langle x_{\text{white}}, x_{\text{dgray}} \rangle, \langle x_{\text{gray}}, x_{\text{dgray}} \rangle, \langle x_{\text{dgray}}, x_{\text{black}} \rangle, \\
&\langle x_{\text{left}}, x_{\text{hexagon}} \rangle, \langle x_{\text{left}}, x_{\text{circle}} \rangle, \langle x_{\text{left}}, x_{\text{dgray}} \rangle, \\
&\langle x_{\text{right}}, x_{\text{square}} \rangle, \langle x_{\text{right}}, x_{\text{circle}} \rangle, \langle x_{\text{right}}, x_{\text{white}} \rangle, \\
&\langle x_{\text{right}}, x_{\text{gray}} \rangle, \langle x_{\text{right}}, x_{\text{black}} \rangle, \langle x_{\text{square}}, x_{\text{dgray}} \rangle, \\
&\langle x_{\text{hexagon}}, x_{\text{black}} \rangle, \langle x_{\text{circle}}, x_{\text{gray}} \rangle, \langle x_{\text{circle}}, x_{\text{dgray}} \rangle \}.
\end{aligned}$$

类似地，我们用蓝色标出了那些包含在属性概念中的主要成分。

请回想一下，交互不变性模块的输出为：

$$\mathcal{P}_{\text{inter}}([J_1, \dots, J_k]) := \left\{ (\text{attr}, \mathcal{I}) \,\middle|\, \begin{array}{c} \mathcal{I} \subseteq \text{pd}(J_\cap) - \text{pd}(J_+), \\ \text{attr} \in \mathcal{L}, I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle \ \forall I \in \mathcal{I} \end{array} \right\},$$

因此， $\mathcal{P}_{\text{inter}}(\mathbf{J}_2)$ 是由以下两对组成的集合：

$$\left\{ \begin{array}{l} (\text{“color”}, \{\langle x_{\text{white}}, x_{\text{dgray}} \rangle, \langle x_{\text{gray}}, x_{\text{dgray}} \rangle, \langle x_{\text{black}}, x_{\text{dgray}} \rangle\}), \\ (\text{“type”}, \{\langle x_{\text{pentagon}} \rangle, \langle x_{\text{square}}, x_{\text{circle}} \rangle, \langle x_{\text{hexagon}}, x_{\text{circle}} \rangle\}) \end{array} \right\}.$$

第一对编码了这样一条信息：提取出的三个“颜色”概念分别对应“白色或深灰色”、“灰色或深灰色”和“黑色或深灰色”。第二对编码了这样一条信息：提取出的三个“类型”概念分别对应“五边形”、“正方形或圆形”和“六边形或圆形”。

示例 3（组合不变性模块）。考虑第一行的右侧子面板 $\bar{\mathbf{J}}_1^{(\text{right})}$，见图 6，以及 $\Delta = 1$。我们可以生成新行：

$$\bar{J}'_{1,1} = f^2_{\text{next}}(\bar{J}^{(\text{right})}_{1,1}|1) = \langle x_{\text{right}} x_{\text{one}} x_{\text{pentagon}} x_{\text{black}} x_{\text{small}} \rangle,$$
$$\bar{J}'_{1,2} = f_{\text{next}}(\bar{J}^{(\text{right})}_{1,2}|1) = \langle x_{\text{left}} x_{\text{two}} x_{\text{pentagon}} x_{\text{dgray}} x_{\text{large}} \rangle,$$
$$\bar{J}'_{1,3} = \bar{J}^{(\text{right})}_{1,3} = \langle x_{\text{right}} x_{\text{one}} x_{\text{pentagon}} x_{\text{gray}} x_{\text{avg}} \rangle.$$

回想一下，组合不变性模块的输出为：

$$\mathcal{P}_{\text{comp}}([J_1, \ldots, J_k]) := \left\{ (\text{attr}, f) \,\middle|\, \begin{array}{c} \exists I \in \bigcap_{i=1}^k \text{pd}(f^{k-i}(J_i)), \\ \text{attr} \in \mathcal{L}, I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle \end{array} \right\}.$$

因此，我们有 $\mathcal{P}_{\text{comp}}(\bar{\mathbf{J}}_1^{(\text{right})}) = \{(\text{“类型”}\ f_{\text{next}}|_{\Delta=1})\}$，这意味着由 f_{next} 以步长 $\Delta = 1$生成的新行中，“类型”值（即 x_{pentagon}）保持不变。

示例 4（二元算子模块）。回想一下，二元算子模块的输出为：

$$\mathcal{P}_{\text{binary}}(\mathbf{J}_i) := \left\{ \overline{\oslash} \,\middle|\, \begin{array}{c} \overline{\oslash} = [\oslash_1, \ldots, \oslash_{k-2}],\ \oslash_i \in \Lambda, \\ g(J_1)\ \oslash_1 \cdots \oslash_{k-2}\ g(J_{k-1}) = g(J_k) \end{array} \right\}.$$

令 g 为概念上的实值函数，定义为 $J \mapsto |\text{mingen}(J)|$。注意，对于图 6 中的两行 $\mathbf{J}_1$ 和 $\mathbf{J}_2$，每个面板 $J_{i,j}$ 都有两个实体，可计算为 $g(J_{i,j}) = 2$。因此，如果 $\Lambda = \{+, -\}$，那么 $\mathcal{P}_{\text{binary}}(\mathbf{J}_1)$ 和 $\mathcal{P}_{\text{binary}}(\mathbf{J}_2)$ 均为空。

示例 5（答案生成）。对于图 6 所示的示例，我们有 $\text{comPos}(\mathbf{J}) = \{x_{\text{left}}, x_{\text{right}}\}$。注意，$\bar{\mathbf{J}}^{(x_{\text{left}})} = \hat{\mathbf{J}}(x_{\text{right}})$。（只有左侧实体的面板与没有右侧实体的面板相同。）因此，我们只考虑概念矩阵列表 $[\mathbf{J}\ \bar{\mathbf{J}}^{(x_{\text{left}})} - \mathbf{J}^{(x_{\text{right}})}]$。

通过迭代应用不变性模块，我们便可计算出 $\mathcal{P}_{1,2}(\mathbf{J}) := \mathcal{P}_1^{(\text{all})}(\mathbf{J}) \cap \mathcal{P}_2^{(\text{all})}(\mathbf{J})$，其 13 个元素标记为 $\mathfrak{P}_1, \ldots, \mathfrak{P}_{13}$：

$\mathfrak{P}_1 = (\quad,\ )$ $\mathfrak{P}_2 = (\quad{}^{-(x_{\text{left}})})$ $\mathfrak{P}_3 = (\quad)$ “数量” J “数量” $\bar{\text{J}}^{(x_{\text{left}})}$ “数量” $\bar{\text{J}}$
$\mathfrak{P}_4 = (\quad,\ )$ $\mathfrak{P}_5 = (\quad{}^{-(x_{\text{left}})})$ $\mathfrak{P}_6 = (\quad{}^{-(x_{\text{right}})})$
$^{(x_{\text{right}})}$ “位置” J “位置” J) “位置” J) $\mathfrak{P}_7 = (\quad \{\langle x_{\text{square}} \rangle, \langle x_{\text{pentagon}} \rangle, \langle x_{\text{circle}} \rangle\}\quad {}^{-(x_{\text{left}})})$
(“类型”) J) $\mathfrak{P}_8 = (\quad f_{\text{next}}|_{\Delta=1}\quad {}^{-(x_{\text{right}})})$ $\mathfrak{P}_9 = (\quad {}^{-}$(“类型”) J) “颜色” $\text{J}^{(x_{\text{right}})})$
$\mathfrak{P}_{10} = (\quad \{\langle x_{\text{white}} \rangle, \langle x_{\text{gray}} \rangle, \langle x_{\text{black}} \rangle\}\quad {}^{-(x_{\text{left}})}$ (“颜色”) J))
$\mathfrak{P}_{11} = (\quad,\ )$ $\mathfrak{P}_{12} = (\quad {}^{-(x_{\text{left}})})$ $\mathfrak{P}_{13} = (\quad$ “大小” J “大小” J) “大小” $\bar{\text{J}}^{(x_{\text{right}})})$

我们还可以生成第 3 行中前两个面板的代数表示 $\mathbf{J}_3 := [J_{3,1}, J_{3,2}]$：

$$J_{3,1} = \langle x_{\text{two}} x_{\text{left}} x_{\text{circle}} x_{\text{avg}} x_{\text{gray}}, x_{\text{two}} x_{\text{right}} x_{\text{pentagon}} x_{\text{gray}} x_{\text{large}} \rangle;$$
$$J_{3,2} = \langle x_{\text{two}} x_{\text{left}} x_{\text{square}} x_{\text{avg}} x_{\text{white}}, x_{\text{two}} x_{\text{right}} x_{\text{hexagon}} x_{\text{gray}} x_{\text{large}} \rangle.$$

在这13种模式中，只有 $\mathfrak{P}_{11}$（实体的“大小”值在$\mathbf{J}$的每一行中都不变）与 $\mathbf{J}_3$，，因为J_3中两个实体的“大小”值、$_1$（或$J_{3,\ 2}$）不同。其余每一种不冲突的模式都提供了有关$J_{3,\ 3}$对应属性值的信息。 $\mathfrak{P}_1$ 至 $\mathfrak{P}_6$：J_3中有两个实体，一个在左侧，另一个在右侧；$\mathfrak{P}_7 \Rightarrow x_{\text{pentagon}}$ 适用于左侧实体；$\mathfrak{P}_8 \Rightarrow x_{\text{circle}}$ 适用于右侧实体；$\mathfrak{P}_9 \Rightarrow x_{\text{gray}}$ 适用于右侧实体；$\mathfrak{P}_{10} \Rightarrow x_{\text{black}}$ 适用于左侧实体；$\mathfrak{P}_{12} \Rightarrow x_{\text{avg}}$ 适用于左侧实体；$\mathfrak{P}_{13} \Rightarrow x_{\text{large}}$适用于右侧实体。因此，我们可以收集所有属性值，构成生成的答案概念

$$J'_{3,3} = \langle x_{\text{two}} x_{\text{left}} x_{\text{pentagon}} x_{\text{avg}} x_{\text{black}}, x_{\text{two}} x_{\text{right}} x_{\text{circle}} x_{\text{large}} x_{\text{gray}} \rangle.$$

B.4. 消融研究

为了研究每个不变性模块的作用，我们进行了消融研究，并在答案选择任务上评估了我们的代数机器推理框架的性能；见表9。

对于任意给定的 RPM 实例，`comPattern` 列表（定义见算法 1）中可能出现并列情况，即 `comPattern` 中有多个条目得分相同且均为最高分。请注意，i 个 `comPattern` 条目表示在题目矩阵中插入第 i 个答案选项后，跨 3 行提取出的共同模式数量。因此，我们报告框架的加权准确率，其定义如下：

$$\frac{1}{N} \sum_{i=1}^{N} \frac{1}{n_i},$$

其中，N 表示 RPM 测试实例的总数，n_i 表示第 i 个测试实例中得分相同且均为最高分的答案选项数量。

如表 9所示，跨不变性模块对最终平均准确率的贡献最大。有趣的是，正是这个模块使我们发现了一种如图 4所示的意外新模式。这是一种相当自然的模式，人类大概也能想到，但它并非 I-RAVEN 的设计规则之一。

B.5. 生成答案示例

为了进一步评估我们的代数机器推理框架在答案生成任务上的表现，我们在图 7 中展示了 I-RAVEN 数据集中给定的正确答案图像与我们生成的答案图像之间的对比。对于每种配置，我们选取前 10 个测试文件，以对比给定的答案图像和生成图像。（在 I-RAVEN 数据集中，每 10 个实例划分为 6 个训练实例、2 个验证实例和 2 个测试实例。因此，每个测试实例的文件名都以“8 test”或“9 test”结尾。）

从图 7可以看出，对于以下配置：中心、外-内中心、左-右和

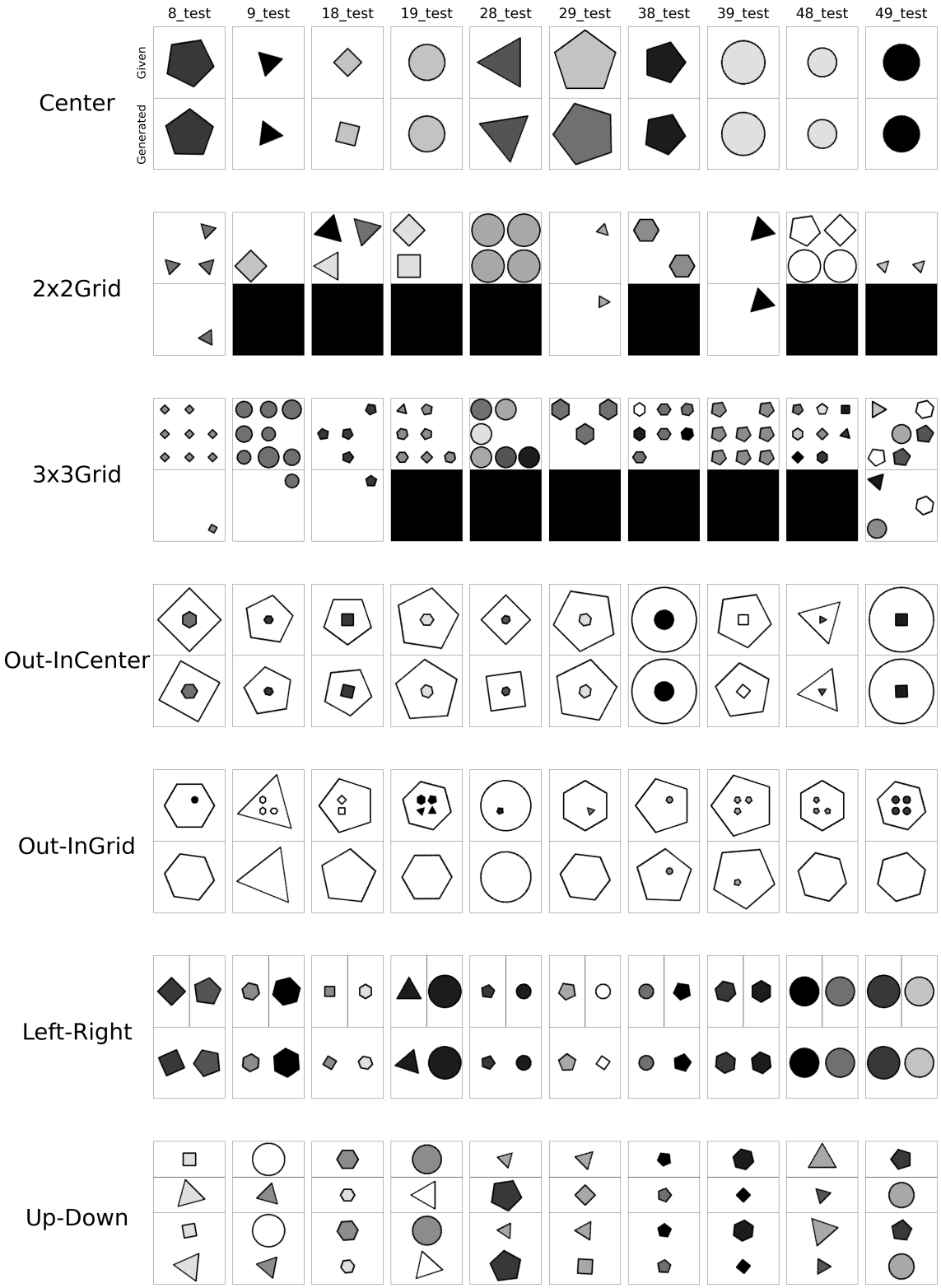


图7. 各配置下前10个 I-RAVEN 测试实例的给定答案图像（上）及其对应的生成图像（下）。对于推理框架未能提取出无冲突模式的测试实例，算法 7 的输出将为 $I_{\text{out}} = \langle 0 \rangle$；我们已将此类生成结果标示为全黑面板。

在上下配置中，我们生成的答案捕捉到了几乎所有显著的图像属性特征，唯独未涉及实体的角度（即相对于面板的朝向），因为 RAVEN/I-RAVEN 的规则中并不涉及这一特征。然而，对于其余配置，我们的推理框架只能生成部分答案。这是因为我们简化了答案生成过程：在题目矩阵的各个面板中，分别提取仅限于每个共同位置的模式。因此，对于 RPM 面板始终包含一组固定位置的配置，我们的推理框架能够有效地提取模式并生成正确答案。另一方面，对于 RPM 面板可能包含任意位置集合的配置，我们的推理框架表现不佳。

C. 进一步讨论

C.1. 为什么代数机器推理不同于基于逻辑的推理?

逻辑是推理的基础。逻辑编程 [3, 28, 41] 是一种基于逻辑的编程范式，是基于逻辑的推理方法所依托的基础计算框架。这些基于逻辑的推理方法的核心思想是，推理可以非常具体地实现为逻辑表达式的归结（或逆归结）。这一思想内含着可满足性的概念；参见 [28]。直观而言，我们有一个逻辑表达式，通常采用某种规范范式表示，并希望为该逻辑表达式中的文字赋予真值（真或假），使整个逻辑表达式得到满足（即真值为“真”）。事实上，当今自动定理证明领域令人振奋的许多进展 [2, 27, 36, 39, 54, 74, 81, 92] 都基于基于逻辑的推理。

相比之下，代数机器推理建立在计算代数和计算机代数系统之上；参见附录A.2.2。如4节所述，我们的代数方法的核心思想是，推理可以非常具体地实现为求解代数中的计算问题。关键在于，这里没有可满足性的概念。我们不为 $R = \Bbbk[x_1, \ldots, x_n]$中的概念赋予真值（或数值）。特别是，尽管 $\langle x_1 \rangle$，……，$\langle x_n \rangle$ 中的原始概念对应于变量 R，……，x_1，我们并不为原始概念赋值。相反，正如我们在3节所强调的，理想被视为“实际研究对象”，而我们将“求解推理任务”归结为“求解涉及理想的非数值计算问题”。

将代数机器推理与归纳逻辑编程进行比较时，这种对比或许更加明显 [51, 52]。两者都旨在解决涉及发现新模式的推理任务，但它们在根本上思维方式各不相同。

在归纳逻辑编程中，我们会得到（作为起点）背景知识 B 和示例 E，两者都以逻辑公式的形式给出。例如，B 可以是蕴含式的合取（$p_1 \wedge \cdots \wedge p_k \rightarrow q$），而 E可以是合取范式公式。目标是发现一个假设 H，即一个必须满足 $B \wedge H \models E$的逻辑公式（也就是说，$B \wedge H$蕴含 E）。通俗地说，我们要发现一个“良好”的假设 H，使其与背景知识和示例都一致。关键的技术挑战在于，如何从众多（可能呈指数级增长的）可能假设中，找到一个在精确定义且可计算的意义上“良好”的假设 H。本质上，归纳逻辑编程将这一技术挑战视为搜索与选择问题 [33, 52, 61]：在所有候选假设构成的空间中，搜索有效假设的一个子集；然后在该子集中，选择一个使预先定义的目标函数达到最大值或最小值的有效假设（例如，选择 H 使柯尔莫哥洛夫复杂度最小）。

相比之下，代数机器推理本质上并非基于搜索。在求解 RPM任务时，通过代数计算来发现新模式。对于 RPM答案生成任务，我们并非在所有候选答案构成的（巨大）空间中搜索有效答案，而是直接计算答案。模式发现由此转变为计算：我们计算各种主分解，并计算哪些主分量包含于属性概念中。对于 RPM答案选择任务，我们将其作为“计算并选择”问题来求解。对于 RPM答案生成任务，我们尽可能根据提取的模式计算属性值，并为那些未涉及提取模式的实体属性随机选择属性值。

在求解瑞文渐进矩阵时，如果尝试用类似于归纳逻辑编程的语言来描述代数方法的推理过程，那么可以表述如下：给定一个以概念矩阵$\mathbf{J}$表示的 RPM 实例，我们从$\mathbf{J}$的前两行中提取模式 $\mathcal{P}_{1,2}(\mathbf{J})$；其中 $\mathcal{P}_{1,2}(\mathbf{J})$ 扮演“背景知识”的角色。随后，我们系统地检查提取出的模式是否与概念 $J_{3,1}$,$J_{3,2}$ （代表题目矩阵第 3 行的两个面板）相冲突；可从 $[J_{3,1}, J_{3,2}]$ 提取的模式则扮演“示例”的角色。最后，我们希望为缺失的第 9 个面板发现一个合适的概念 $J_{3,3}$；其中 $J_{3,3}$ 扮演“假设”的角色。

然而，仔细审视便会发现，我们试图建立的类比存在若干差异。就归纳逻辑编程的设定而言，我们似乎有与“背景知识”“示例”和“假设”相对应的概念，但我们所谓的“背景知识”（即提取的模式 $\mathcal{P}_{1,\ 2}(\mathbf{J})$）并非给定的先验知识，而是

必须经过计算。同样，“示例”也不是给定的先验知识，而是从 $[J_{3,1}, J_{3,2}]$ 中提取、必须经过计算的模式。为了发现“假设”，我们并非在寻找它，而是在计算它。实际上，要使这种“类比”成立，普遍缺少的要素是代数计算，而这正是代数机器推理的本质。

C.2. 代数机器推理的潜在社会影响

代数机器推理有望帮助自动化目前由人类执行的“较简单”的推理任务。在教育领域，代数机器推理有助于设计更好的智力测试。在金融领域，我们的框架可以根据此类案例的不变特征，协助处理个人银行贷款申请或检测欺诈行为。不过，也可能产生负面影响。我们的推理过程是从少量示例（如 RPM任务中的示例）中提取模式，再进行泛化。如果银行贷款申请或欺诈案例已经与某些社会经济群体不公平地关联起来，那么即使采用我们的推理框架，这种不公平也会延续下去。

我们的框架另一个潜在的下游应用领域是医疗诊断。基于患者报告的症状和既往病史进行的诊断决策，可以建模为推理任务。具体症状和医疗状况可以编码为概念，并有望据此得出最终的医学诊断。如果实施得当，用于医学诊断的代数机器推理框架将显著加快诊断决策过程，并降低医疗诊断的成本。不幸的是，这种应用虽然能带来显而易见的益处，也伴随着伦理问题。如果医生在借助代数机器推理进行诊断决策时作出了错误的医学诊断，谁该承担责任？推理框架的创建者会因此面临法律诉讼吗?

偏见也可能内在于机器推理框架的人类 编码过程中。如果编码的概念反映了现存的人类偏见，那么推理输出就可能有误，却看似“合理”。未来若将代数机器推理应用于法律或刑事案件，必须谨慎评估和处理与各交易对方相关的信息（概念）。尽管公平性不在本框架的讨论范围之内，我们的方法仍可揭示潜在的滥用行为。

最后，由于代数机器推理在 RPM任务上的表现能够超越人类表现（RPM 是一种智力测试，最初旨在评估人类的一般智能和抽象推理），我们的工作实际上正在揭开人类认知最重要的堡垒之一。这可能会无意中加剧公众对人工智能将超越人类的担忧。缓解这一问题的一种方式，是通过教育和公众宣传，消除人们对代数机器推理的神秘感。

参考文献

[1] J. Abbott、A. M. Bigatti 和 L. Robbiano。CoCoA：用于交换代数计算的系统。网址：http://cocoa.dima.unige.it。12[2] Ibrahim Abdelaziz、Maxwell Crouse、Bassem Makni、Ver- non Austel、Cristina Cornelio、Shajith Ikbal、Pavan Kapani- pathi、Ndivhuwo Makondo、Kavitha Srinivas、Michael Wit- brock 等。学习引导基于饱和的定理证明器。《IEEE模式分析与机器智能汇刊》，2022。6, 26[3] Krzysztof R Apt。逻辑编程。《理论计算机科学手册》B 卷：形式模型与语义（B），1990：493–574，1990。26[4] Dave Bayer、Andr´e Galligo 和 Mike Stillman。Gr¨obner 基与标量扩张。载于《计算代数几何与交换代数》（科尔托纳，1991），《数学研讨会论文集》，第 XXXIV 卷，第 198–215 页。剑桥大学出版社，剑桥，1993。3, 13[5] Yaniv Benny、Niv Pekar 和 Lior Wolf。尺度局部化抽象推理。载于 IEEE/CVF 计算机视觉与模式识别会议论文集，第 12557–12565 页，2021。2[6] L´eon Bottou。从机器学习到机器推理。《机器学习》，94(2):133–149，2014年。6[7] Susan Carey。概念的起源。《认知与发展杂志》，1(1):37–41，2000。15, 16[8] Patricia A Carpenter、Marcel A Just 和 Peter Shell。某项智力测试测量什么：对瑞文渐进矩阵测验中加工过程的理论解释。《心理学评论》，97(3):404，1990。1[9] John B Carroll 等。人类认知能力：因子分析研究综述。第 1 号。剑桥大学出版社，1993。1[10] Kai Chen、Jiaqi Wang、Jiangmiao Pang、Yuhang Cao、Yu Xiong、Xiaoxiao Li、Shuyang Sun、Wansen Feng、Ziwei Liu、Jiarui Xu 等。MMDetection：OpenMMLab 检测工具箱与基准测试。arXiv 预印本 arXiv:1906.07155，2019。19[11] Kai Fong Ernest Chong。深入考察神经网络的逼近能力。载于国际学习表征会议，2020。2[12] David A Cox、John Little 和 Donal O’Shea。理想、簇与算法：计算代数几何与交换代数导论。施普林格，2015。2, 9, 10, 11, 12[13] Ian Davidson 和 Peter B. Walker。迈向流体机器智能：我们能否打造天才人工智能？AAAI 人工智能大会论文集，33(01):9760–9764，2019。1[14] Wolfram Decker、Gert-Martin Greuel、Gerhard Pfister 和 Hans Sch¨onemann。SINGULAR 4-3-0——一个用于多项式计算的计算机代数系统

。http://www. singular.uni-kl.de，2022。12[15] Mathias Drton、Bernd Sturmfels 和 Seth Sullivant。《代数统计学讲义》，第 39 卷。Springer Science & Business Media，2008。2[16] Eliana Duarte、Orlando Marigliano 和 Bernd Sturmfels。具有有理极大似然估计量的离散统计模型。《Bernoulli》，27(1):135–154，2021。2[17] David Eisenbud。《交换代数》，Graduate Texts in Mathematics 第 150 卷。Springer-Verlag，纽约，1995。兼论代数几何。9, 10, 12, 13[18] David Eisenbud、Craig Huneke 和 Wolmer Vasconcelos。主分解的直接方法。《Invent. Math.》，110(2):207–235，1992。12[19] GAP 团队。GAP——群、算法与编程，4.12.1 版，2022。https://www.gap-system.org。12[20] Edmund Gettier。有根据的真信念就是知识吗？载于《论辩知识》，第 14–15 页。Routledge，2020。15[21] Patrizia Gianni、Barry Trager 和 Gail Zacharias。Gr¨obner 基与多项式理想的主分解。《J. Symbolic Comput.》，6(2-3):149–167，1988。交换代数的计算方面。12[22] Alison Gopnik 和 Henry M Wellman。理论理论。本文的早期版本曾在 1991 年儿童发展研究学会会议上宣读。剑桥大学出版社，1994。15[23] Richard E. Grandy。理解与组合性原则。《Philosophical Perspectives》，4:557–572，1990。15[24] Daniel R. Grayson 和 Michael E. Stillman。Macaulay2：用于代数几何研究的软件系统。网址：http://www.math.uiuc.edu/Macaulay2/。12[25] Serkan Hos¸ten 和 Gregory G Smith。单项式理想。载于《使用 Macaulay 2 进行代数几何计算》，第 73–100 页。施普林格，2002。13[26] Sheng Hu、Yuqing Ma、Xianglong Liu、Yanlu Wei 和 Shi- hao Bai。用于抽象视觉推理的分层规则感知网络。载于 AAAI 人工智能大会（AAAI）论文集，2021。2, 7, 16[27] Geoffrey Irving、Christian Szegedy、Alexander A Alemi、Niklas E´en、Franc¸ois Chollet 和 Josef Urban。Deepmath——用于前提选择的深度序列模型。载于《神经信息处理系统进展》，第 2235 2243 页，2016 6, 26 – 。[28] Joxan Jaffar 和 J-L Lassez。约束逻辑编程。载于第 14 届 ACM SIGACT-SIGPLAN 编程语言原理研讨会论文集，第 111–119 页，1987。6, 26[29] Marius Jahrens 和 Thomas Martinetz。使用多层关系网络求解瑞文渐进矩阵。arXiv 预印本 arXiv:2003.11608，2020。1, 2[30] Hans Kamp 和 Barbara Partee。原型理论与组合性。《Cognition》，57(2):129–191，1995。15[31] Joa Kileel、Matthew Trager 和 Joan Bruna。深度多项式神经网络的表达能力。《神经信息处理系统进展》，2019。2[32] Youngsung Kim、Jinwoo Shin、Eunho Yang 和 Sung Ju Hwang。通过元类比对比学习进行少样本视觉推理。NeurIPS 2020，2020。1[33] Tim Kimber、Krysia Broda 和 Alessandra Russo。失败驱动归纳：学习连通霍恩理论。载于《逻辑编程与非单调推理国际会议论文集》，第 169–181 页。施普林格，2009。26[34] Franz J Kir´aly、Paul Von B¨unau、Frank C Meinecke、Dun- can AJ Blythe、Klaus-Robert M¨uller 和 Kenji Fukumizu。概率分布的代数几何比较。《机器学习研究杂志》，13(3)，2012。2[35] Brenden M Lake、Tomer D Ullman、Joshua B Tenenbaum 和 Samuel J Gershman。智能的要素：从经典论争到工程路线图。《行为与脑科学》，40，2017。15[36] Guillaume Lample 和 Franc¸ois Charton。符号数学的深度学习。载于《国际学习表征会议论文集》，2020。6，26[37] Barbara Landau。真正的祖母请站出来好吗？双重意义表征的心理现实性。《心理语言学研究杂志》，11(1):47–62，1982。15[38] Paul Larsen 和 Franz Kir´aly。一般交的法诺概形与机器学习。《代数与计算国际期刊》，24(07):923–933，2014年。2[39] Wenda Li、Lei Yu、Yuhuai Wu 和 Lawrence C Paulson。Isarstep：高层次数学推理的基准测试。载于《国际学习表征会议论文集》，2021。6，26[40] Shaowei Lin。贝叶斯统计中积分求值的代数方法。博士论文，加州大学伯克利分校，2011。2[41] John W Lloyd。《逻辑编程基础》。Springer Science & Business Media，2012。26[42] John Locke。《人类理解论》，1690。1948。15，16[43] Muyang Lyu、Ruixuan Liu 和 Junyi Wang。使用循环神经网络推理网络求解瑞文渐进矩阵。载于《2022年第七届国际计算智能与应用会议（ICCIA）论文集》，第 32–37 页。电气电子工程师学会，2022。2[44] Barbara C Malt 和 Edward E Smith。自然类别中的相关属性。《言语学习与言语行为杂志》，23(2):250–269，1984。15[45] Petros Maragos、Vasileios Charisopoulos 和 Emmanouil Theodosis。热带几何与机器学习。《电气电子工程师学会会刊》，2021。2[46] Eric Ed Margolis 和 Stephen Ed Laurence。《概念：核心读物》。麻省理工学院出版社，1999。15，16[47] Douglas L Medin、Edward E Smith 等。概念与概念形成。《心理学年度评论》，35(1):113–138，1984。15[48] Ezra Miller 和 Bernd Sturmfels。《组合交换代数》，《数学研究生教材》第 227 卷。Springer-Verlag，纽约，2005。10[49] Melanie Mitchell。人工智能中的抽象与类比生成。arXiv 预印本 arXiv:2102.10717，2021。即将刊于《纽约科学院年刊》。1[50] Melanie Mitchell。为什么人工智能比我们想象的更难。arXiv 预印本 arXiv:2104.12871，2021。1

[51] Stephen Muggleton。归纳逻辑编程。《New Generation Computing》，8(4):295–318，1991。26[52] Stephen Muggleton。逆蕴涵与 Progol。《New Generation Computing》，13(3):245–286，1995。26[53] Gregory Murphy。《概念大百科》。麻省理工学院出版社，2004。15[54] Aditya Paliwal、Sarah Loos、Markus Rabe、Kshitij Bansal 和 Christian Szegedy。高阶逻辑与定理证明的图表示。《AAAI 人工智能大会论文集》，34(03):2967–2974，2020年4月。26[55] Niv Pekar、Yaniv Benny 和 Lior Wolf。为渐进矩阵智力测验生成正确答案。《神经信息处理系统进展》，33 7400，2020 7390 2 : – 。[56] Patrick Perret。儿童归纳推理：发展与教育视角。《认知教育与心理学杂志》，14(3):389–408，2015。1[57] Sylvain Petitjean。计算机视觉中的代数几何与对象表示。载于 Martial Hebert、Jean Ponce、Terry Boult 和 Ari Gross 编，《计算机视觉中的对象表示》，第 155–165 页，柏林、海德堡，1995。施普林格·柏林·海德堡。2[58] Steven Pinker 和 Ray Jackendoff。语言机能：它有何特殊之处？《Cognition》，95(2):201–236，2005。15[59] Meng Qu、Junkun Chen、Louis-Pascal Xhonneux、Yoshua Bengio 和 Jian Tang。RNNLogic：学习用于知识图谱推理的逻辑规则。载于《国际学习表征会议论文集》，2021。1[60] Markus Norman Rabe、Dennis Lee、Kshitij Bansal 和 Christian Szegedy。通过自监督跳跃树训练进行数学推理。载于《国际学习表征会议论文集》，2021。1[61] Oliver Ray、Krysia Broda 和 Alessandra Russo。混合溯因归纳学习：Progol 的一种推广。载于《国际归纳逻辑编程会议论文集》，第 311–328 页。施普林格，2003。26[62] Adam Santoro、Felix Hill、David Barrett、Ari Morcos 和 Timothy Lillicrap。测量神经网络中的抽象推理。载于《国际机器学习会议论文集》，第 4477–4486 页，2018。1，2，7[63] Snejana Shegheva 和 Ashok Goel。用于求解瑞文渐进矩阵智力测验的结构亲和方法。载于《AAAI 人工智能大会论文集》，2018。1[64] Fan Shi、Bin Li 和 Xiangyang Xue。通过潜在高斯过程先验完成瑞文渐进矩阵。载于《AAAI 人工智能大会论文集》，第 35 卷，第 9612–9620 页，2021。2[65] Takeshi Shimoyama 和 Kazuhiro Yokoyama。多项式理想的局部化与主分解。《符号计算杂志》，22(3):247–277，1996。12[66] Richard P. Stanley。《组合学与交换代数》，《数学进展》第 41 卷。Birkh¨auser Boston, Inc.，马萨诸塞州波士顿，第二版，1996。10[67] Mike Stillman、Carolyn Yackel、Justin Chen 和 Mahrud Sayrafi。PrimaryDecomposition：主分解与关联素理想例程。2.0 版。Macaulay2 软件包可在 https://github.com/Macaulay2/M2/tree/master/M2/Macaulay2/packages 获取。13[68] J¨urgen Stuber。交换环的叠加定理证明。载于《自动演绎——应用的基础》，第31–55页。施普林格，1998。2[69] Bernd Sturmfels。什么是.....Gr¨obner 基？《美国数学会通告》，52(10):1199–1200，2005。11[70] Seth Sullivant。高斯贝叶斯网络的代数几何。《应用数学进展》，40 513，2008 4 482 2 ( ): – 。[71] Henry Taylor 和 Peter Vickers。概念碎片化与取消主义的兴起。《欧洲科学哲学杂志》，7(1):17–40，2017。15[72] Sage 开发团队。SageMath，Sage 数学软件系统（9.6版），2022。https://www.sagemath.org。12[73] Sjoerd Van Steenkiste、Francesco Locatello、J¨urgen Schmidhuber 和 Olivier Bachem。解缠表示是否有助于抽象视觉推理？载于《神经信息处理系统进展》，第14245–14258页，2019。1[74] Mingzhe Wang 和 Jia Deng。通过学习生成定理来学习证明定理。载于《神经信息处理系统进展》，第33卷，第18146–18157页，2020。26[75] Po-Wei Wang、Priya Donti、Bryan Wilder 和 Zico Kolter。Satnet：使用可微满足性求解器，连接深度学习与逻辑推理。载于《国际机器学习会议》，第6545–6554页。PMLR，2019。1[76] Sumio Watanabe。不可识别学习机的代数分析。《神经计算》，13(4):899–933，2001年4月。2[77] Sumio Watanabe。层次学习机的代数几何方法。《神经网络》，14(8):1049–1060，2001年10月。2[78] Sumio Watanabe。代数几何与统计学习理论。第25号。剑桥大学出版社，2009。2[79] Sumio Watanabe。一种广泛适用的贝叶斯信息准则。《机器学习研究杂志》，14(3):867–897，2013。2[80] Sumio Watanabe 和 Manfred Opper。奇异学习理论中贝叶斯交叉验证与广泛适用的信息准则的渐近等价性。《机器学习研究杂志》，11(12)，2010。2[81] Yuhuai Wu、Albert Jiang、Jimmy Ba 和 Roger Baker Grosse。INT：用于评估定理证明泛化能力的不等式基准测试。载于《国际学习表征会议》，2021。6, 26[82] Keisuke Yamazaki 和 Sumio Watanabe。混合模型中的奇点与随机复杂度上界。《神经网络》，16(7):1029–1038，2003。2[83] Chi Zhang、Feng Gao、Baoxiong Jia、Yixin Zhu 和 Song-Chun Zhu。RAVEN：一个用于关系与类比视觉推理的数据集。载于《IEEE计算机视觉与模式识别会议论文集》

，第5317–5327页，2019。1, 2, 7, 16[84] Chi Zhang、Baoxiong Jia、Feng Gao、Yixin Zhu、Hongjing Lu 和 Song-Chun Zhu。通过对比学习感知推理。载于《神经信息处理系统进展》，第1075–1087页，2019。1, 2, 7[85] Chi Zhang、Baoxiong Jia、Song-Chun Zhu 和 Yixin Zhu。通过概率溯因与执行进行抽象时空推理。载于《IEEE计算机视觉与模式识别会议论文集》（CVPR），2021。2, 7[86] Chi Zhang、Sirui Xie、Baoxiong Jia、Ying Nian Wu、Song-Chun Zhu 和 Yixin Zhu。学习代数表示，以实现抽象推理中的系统泛化。载于《计算机视觉——ECCV 2022：第17届欧洲会议，以色列特拉维夫，2022年10月23日至27日，论文集》，第XXXIX部分，第692–709页。施普林格，2022。2[87] Liwen Zhang、Gregory Naitzat 和 Lek-Heng Lim。深度神经网络的热带几何。载于《国际机器学习会议》，第5824–5832页。PMLR，2018。2[88] Kecheng Zheng、Zheng-Jun Zha 和 Wei Wei。利用干扰特征进行抽象推理。载于《神经信息处理系统进展》，第32卷，2019。2, 7[89] Tao Zhuo、Qiang Huang 和 Mohan Kankanhalli。瑞文问题矩阵的无监督抽象推理。《IEEE图像处理汇刊》，30:8332–8341，2021。2, 7[90] Tao Zhuo 和 Mohan Kankanhalli。利用神经网络求解瑞文渐进矩阵。arXiv 预印本 arXiv:2002.01646，2020。2[91] Tao Zhuo 和 Mohan Kankanhalli。利用双对比网络实现有效的抽象推理。载于《国际学习表征会议》，2021。2, 7[92] Zsolt Zombori、Josef Urban 和 Chad E Brown。Prolog 技术强化学习证明器。载于《国际自动推理联合会议》，第489–507页。施普林格，2020。6, 26