迈向数学与代码正确性的自动形式化：基础证明实验

Garett Cunningham电子与计算机科学学院 俄亥俄大学 俄亥俄州雅典 45701
gc974517@ohio.edu

Razvan C.Bunescu计算机科学系 北卡罗来纳大学夏洛特分校 北卡罗来纳州夏洛特 28223
razvan.bunescu@uncc.edu

David Juedes
电子与计算机科学学院 俄亥俄大学 俄亥俄州雅典 45701
juedes@ohio.edu

arXiv:2301.02195v1 [cs.CL] 5 Jan 2023

摘要

数学证明日益复杂，使数学家进行人工验证变得极其耗费认知资源。自动形式化旨在解决这一问题：将以自然语言写成的证明转换为形式化表示，并通过交互式定理证明器进行计算机验证。本文介绍了一种基于通用变换器架构的语义解析方法，可将初等数学证明转换为等价的形式化表达，并用 Coq 交互式定理证明器的语言表示。我们还使用同一架构进行训练，使其能够将附有霍尔三元组的简单命令式代码转换为可在 Coq 中形式验证的正确性证明。在人工构造和人工撰写的证明这一有限领域上的实验表明，这些模型能够很好地泛化到训练期间未见过的中等长度证明，以及自然语言表达的不同变体。

1 引言

对于未经深入了解的人来说，数学证明不过是人们写下的一种论证，用以使他人相信数学真理。然而，从实质上说，数学证明必须具有超越书面论证本身的形式基础。缺乏这种基础的论证可能存在致命错误，甚至逻辑矛盾（例如，罗素悖论（Irvine 和 Deutsch，2021））。尽管如此，以自然语言写成的数学论证仍是常态，并且具有重要价值。

在Tymoczko（1979）广为人知的论文中，他讨论了一项当时颇具争议的四色定理证明（Appel 和 Haken，1977；Appel 等人，1977），并探讨了“什么是数学证明？”他认为，所有数学证明都必须满足以下条件：（i）具有说服力，（ii）可供审查，（iii）可以形式化。前两点是针对读者而言——证明必须能够说服数学家，并且让他们理解。关于第三点，他指出：“大多数数学家和哲学家都认为，任何可接受的证明都可以形式化。我们总能找到一种适当的形式语言和理论，将非形式证明纳入其中，并‘补充完善’为严谨的形式证明。”对大多数数学家来说，第三点对于确保数学证明中不存在细微但致命的逻辑错误至关重要。

自 20 世纪 70 年代以来，数学领域的重要成果在完全形式化方面取得了长足进展。例如，费特-汤普森定理（Gonthier 等人，2013；Gonthier，2013）和四色定理（Gonthier，2008）已使用证明助手 Coq 进行了形式化验证（Bertot 和 Castéran，2013）；开普勒猜想（Hales，2005；Hales 等人，2017）则已使用证明助手 Isabelle 和 HOL Light 进行了形式化验证（Nipkow 等人，2002）。此外，证明助手在软件验证中展现出巨大价值，例如对 C 编译器进行全面认证（Leroy，2009）。证明代码正确行为的证明，与纯数学中的证明具有相似的结构；在这类证明中，霍尔逻辑等系统取代了标准的一阶逻辑。因此，Tymoczko 关于数学证明的标准也可以扩展到程序验证。对许多专家而言，LaTeX 是满足前两项标准的出色工具。此外，精心编写的 LaTeX（Higham，2020）还提供了丰富的结构，有助于满足第三项标准。

绝大多数现代数学都以自然语言（NL）表述，其中绝大多数采用 LaTeX 排版。使用证明助手将数学完全形式化仍是一项艰巨且耗时的任务。本文通过探索如何运用现代机器学习技术，将经过精心编写的 LaTeX 转换为等价且经过形式化验证的内容，为弥合这一差距迈出了初步步伐

Coq 中的数学，这一过程在文献中称为自动形式化（Szegedy，2020）。

Wang 等人 (2018年、2020) 探索了将数学陈述从 LaTeX 翻译为 Mizar 的类似任务，使用基于长短期记忆网络并采用注意力机制的模型。为了生成对齐的 LaTeX-Mizar 语对，他们使用了一种工具（Bancerek，2006），该工具可将顶层 Mizar 陈述转换为人工生成的 LaTeX 句子。Mizar 可读性强，且其长度与对应的 LaTeX 版本相近，因此这项任务较为容易。 Carman (2021) 评估了长短期记忆网络将一组受限的、人工生成的简单算术表达式定理形式化的能力，并报告称，对于训练期间见过的表达式长度，模型取得了不错的效果。最近，Wu 等人(2022) 评估了 Codex 和 PaLM 在规模小得多、由人类撰写的代数与数论定理集上的表现。

与以往工作不同，我们研究了对定理及其证明进行自动形式化，并将研究范围扩展至代码正确性的证明。我们使用若干手工编写的数学陈述来抽象出一种复杂语法，并以此生成一个数据集，其中包含篇幅显著更长、类型更多样的数学定理和证明。我们基于通用变换器（Dehghani 等人，2018年）开发了一种架构，并改进了复制机制（Gu 等人，2016），以便在测试时处理任意数量和变量名。我们对模型进行了广泛评估，考察其能否系统地泛化到训练期间未见过的陈述长度；我们报告了序列级准确率，以及语义级准确率。后者通过综合定理的序列级准确率，并运行 Coq 判断生成的证明是否正确来计算。代码和数据已发布于 https://github.com/gc974517/autoformalization。

2 定理和证明数据集

我们创建了两个相互独立的数学陈述数据集，总体上对应四类定理和证明：第一个数据集包含三类算术陈述（偶数-奇数、合数和幂），详见第 2.1 节；第二个数据集包含通过霍尔逻辑陈述代码正确性的内容（POLY），详见第 2.2 节。在每个示例中，输入的定理—证明对以 LaTeX 给出，而形式化后的输出则以 Coq 表示。这项工作聚焦于证明助手 Coq（Bertot 和 Castéran，2013），因为：（a）已经为其开发了丰富的数学库；（b）它已成功用于推理重要的计算产物，例如 CompCert C 编译器（Leroy，2009）；（c）它拥有丰富的证明助手训练材料，内容涉及软件验证（Pierce 等人，2010）。

每一类示例都展示了成功实现数学定理和证明自动形式化所必需的特征。例如，幂和合数示例可能会定义有用的术语，使定理更加简短，例如证明 4 是一个平方数；反之，也可能直接陈述定理，而不作任何前置定义，例如证明 $\exists n.\, n^2 = 4$。如图3 和4所示，在 Coq 中，这对应于使用`Definition` 关键字为命题创建别名。此外，数据集中的示例还对第3.1节所述的复制机制进行了压力测试，检验其能否学会在数学表达式中包含正确数量的术语并将其按正确顺序排列，以及如何在定理和证明中放置这些术语，从而能够推广到数学语言中的任意词元。

对于四类定理和证明，我们分别手动创建了几个示例，以指导构建一种复杂语法，再用它生成规模大得多、种类丰富得多的数学定理和证明数据集。每个数据集都以相同的方式，使用相应的语法生成。首先，抽取一个随机种子，用于控制定理、证明以及定义（如有）的整体结构。随后，使用从另一套上下文无关文法中抽取的短语，补全证明的骨架结构。对骨架结构进行粗粒度控制，可以构造出具有有趣特征的示例，例如包含子引理、采用正向或反向的证明方向、涉及共指，或为定理添加额外条件等。

将自然语言中的数学陈述形式化为 Coq 语句时，许多困难源于数学证明详细程度的差异很大，以及自然语言证明中被视为可接受的推理步骤与 Coq 中的推理步骤经常不匹配。此外，对于任何给定的定理，都可能存在多个 Coq 证明，其粒度各不相同。我们

LaTeX 输入序列

定理。 $28M + 308$ 是偶数。

证明。 我们知道， $\mathbb{N}$ 中偶数的求和会是一个偶数。注意，308 是偶数。此外， $M \times 28$ 这一对显然是偶数。这是因为系数 28 是偶数。

□

Coq 输出序列

```
Require Import Arith.
定理 M28_308:
    forall M : nat, 自然数偶数判定(28 * M  + 308)
 = true.证明.intros.repeat rewrite Nat.even_
add. assert (H1: 自然数偶数判定 308 =true). {
auto. } assert (H2: 自然数偶数判定  28 =true). {
auto. } assert (H3: 自然数偶数判定 (28 * M) =
true).{ rewrite Nat.even_mul.rewrite H2.auto. }
rewrite H1.rewrite H3.auto.Qed.
```

LaTeX 输入序列

定理。 $450 + a \cdot 192 + j \cdot 462$ 对于任意自然数项 j和 a都保证为偶数。

证明。 可以证明， $192 \cdot a + j \cdot 462$ 显然是偶数。注意，在 $\mathbb{N}$ 中， $192a$ 是偶数，因为偶数整数与 $\mathbb{N}$ 中的任意数相乘，结果必定是偶数。同样， $462j$ 显然是 $\mathbb{N}$中的偶数。其结论可由以下事实得出：偶数与偶数相加，结果仍为偶数。因此，定理得证。

□

Coq 输出序列

```
导入算术库。定理a450_192j_450_偶数：对于所
有 j a : 自然数集，Nat.even (192 * a  + 462*j
 + 450)  = true。证明。引入变量。重写
Nat.even_加法。断言 (H1: Nat.even (192 * a)
 = true)。{ 重写 Nat.even_乘法。自动。} 断言
(H2: Nat.even (462 * j)  = true)。{ 重写
Nat.even_乘法。自动。} 断言 (H3: Nat.even
(192 * a  + 462* j)  = true)。{ 重复重写
Nat.even_加法。重写 H1。重写 H2。自动。} 重
写 H3。自动。证毕。
```

图 1：奇偶集合生成的示例。

通过要求 Coq 证明的结构与自然语言证明的整体结构相匹配，来消除这种歧义。具体做法是准同步生成数学陈述的 LaTeX 和 Coq 版本，同时允许进行一些简单的重新排序，以提升泛化性能，例如交换交换运算的参数。

图 2：EVEN-ODD 数据集中子引理使用的实例。在证明定理之前，先给出了非恒定项之和为偶数的证明（断言 H3）。

总的来说，在从上下文无关文法中采样并考虑措辞变化之前，基于语法的方法在理论上可以生成超过 2.83 亿个独特的算术示例，以及超过 491,000 个独特的代码示例。

2.1 算术陈述

我们生成了三类数学陈述，即定理-证明对：

- 奇偶：一个表达式是偶数或奇数。
- 合数：一个数是合数。
- 幂：一个数是 n的整数次幂。

奇偶示例包含由具有偶数系数的 n 变量构成的算术表达式，这些表达式与一个常数项相加，这意味着该常数的奇偶性决定了完整表达式。证明会根据我们手动设计的语法，以不同严谨程度利用这一事实，其示例如图1所示。Coq 程序与配套的 LaTeX 示例同时生成。图 2 中的示例展示了如何生成并使用先前的事实来证明一个隐式子引理，这同时体现在自然语言和对应的 Coq 版本中。

关于 幂和 合数 的定理和证明示例在 LaTeX 和 Coq 形式中具有相似的结构，如图3和 4所示。定理断言存在一个自然数，使得某个定义性质成立；其证明是构造性的，不同之处在于，合数示例证明的是分解为 n 因子。

对于训练和测试，我们分别生成 5,000 个奇偶、5,000 个合数，以及 2,000 个幂.....

LaTeX 输入序列

定义。 我们定义：若取某些 R、 $Q \in \mathbb{N}$，满足 Q、$R \geq 2$ 和 $Q \times R = w$，则 $w \in \mathbb{N}$ 是一个合数 自然数。

定理。 35 是一个合数整数。

证明。 请记住，合成自然数是 Q 与 R 的乘积，其中 Q 且 $R \geq 2$。令 $R = 7, Q = 5$。我们根据 $35 = Q \cdot R$论证该结果成立。 □

Coq 输出序列

```
导入 Lia。定义 合数 (w :
自然数) :=存在 RQ :自然数,
(Q  >= 2) /\ (R  >= 2)/
\ (Q * R  =w).定理 w_合
数: 合数 35.证明. 展开合数。
存在 7。存在 5。lia。证毕。
```

图 3：生成的合数示例。

LaTeX 输入序列

定义。 我们称 o 是一个平方数，当且仅当存在某个整数 Z，使得 $Z \geq 2$ 且 $o = Z^2$。

定理。 $o = 64$ 是一个平方数。

证明。 设 $Z = 8$。注意到 $64 = 8^2$。还要注意，$Z = 8$ 大于或等于 2。因此，64 是一个平方整数。 □

Coq 输出序列

```
RequireImport Lia.定义
square(o :nat):=exists
Z: nat,(Z  >= 2)/\(o =Z^2).
定理 square_64:square 64.
证明.unfold
square.exists 8.assert
(H1: 8 >= 2).{lia. }
repeat split.apply
H1.Qed.
```

图 4：来自幂集合的生成示例。

样例。我们使用 $n \in \{2, 3, 5, 7, 9\}$ 的取值进行训练，并使用 $n \in \{2, 3, \ldots, 12\}$ 的取值进行测试，其中 n 分别表示算术表达式中的变量个数、因子个数或幂次，依次对应上述各项。这样做是为了评估模型对未见过的算术表达式长度和因子数量的泛化能力。

2.1.1 手写示例

我们还创建了一个小型集合，其中包含 45 对由人工编写的 LaTeX 定理-证明对，用于评估模型在我们手动生成的语法之外的示例上的表现。这些示例不同于最初用于指导生成语法开发的人工编写示例。算术集合中的每种证明类型各有 15 个示例，使用相同的词汇，但包含若干未见过的语法结构。

2.2 代码正确性陈述

我们创建了一个数据集，其中包含关于用命令式编程语言编写的短程序的正确性证明*Imp* (Pierce 等人，2018年)，我们称之为POLY。这些程序代表了用于计算多项式的各种算法，其正确性证明验证了这些程序是否正确地将多项式建模为数学函数。证明采用完全装饰的程序形式，或采用霍尔三元组序列的形式，并在中间步骤之间用自然语言说明理由。示例如图5所示。

对于训练和测试数据，我们均生成 5,000 个示例。我们使用包含 2、3、5、7、9 和 11 行的程序进行训练，然后在包含 2 至 14 行的程序上进行测试，以评估模型对新程序长度的泛化能力。

3 语义解析架构

为了将 LaTeX 语句形式化为 Coq，我们开发了一种基于通用变换器（Dehghani 等人，2018年）的编码器-解码器架构。与 Csordás 等人（2021）类似，我们通过在基础 Transformer（Vaswani 等人，2017）的编码器和解码器中加入递归传递来实现这一点，从而使模型类似于不具备自适应计算时间（ACT）的通用变换器。此外，我们引入了复制机制，并支持词汇表外数学术语。

3.1 复制机制

数学语言包含自然语言中罕见或不存在的特征，例如数字、变量和经过严格定义的术语。为了解决通用数学术语的使用问题，这些词元在

LaTeX 输入序列

定理。 考虑以下命令序列，使得

```
S := 3; S := 3 + S *
Z; S := 1 + S * Z
```

允许 $Z = y$，对于任意自然数 y，在运行此代码之前，然后在指令集执行完毕后 $S = 3 \times y^2 + 3 \times y + 1$。

证明。 根据通常的霍尔逻辑应用：

$$\{Z = y\}$$
```
S := 3;
```
$$\{Z = y \wedge S = 3\}$$
```
S := 3 + S * Z;
```
$$\{Z = y \wedge S = 3 \times y + 3\}$$
```
S := 1 + S * Z
```
$$\{Z = y \wedge S = 3 \times y^2 + 3 \times y + 1\}$$

因此，可以证明该程序是正确的。 □

Coq 输出序列

```
Require Import String.From
PLFRequire Import Imp.From
PLFRequire Import Hoare.定理 poly_
代码_正确：对于任意 y : nat, {{ Z =
y }} S: = 3; S: = 3 +S * Z; S: = 1 +
S * Z{{S = 3* y ^ 2 + 3*y + 1}}。

证明。
    intros. apply hoare_seqwith (Q := (
(Z = y /\ S = 3) )%断言). apply hoare_
seqwith (Q := ( (Z = y /\ S = 3* y + 3) )%
断言). apply hoare_seqwith (Q := ( (Z = y /
\ S = 3* y^2 + 3 * y + 1) )%断言). all:
eapply hoare_consequence_pre; try
(apply hoare_asgn || assn_auto''). Qed.
```

图 e 5：生成的 POLY 示例： [左侧] 霍尔逻辑证明； [右侧]Co 中的代码正确性证明 q.

LaTeX 输入中包含表示其用途的通用形式，例如 <var1> 直到 <varN> 表示变量，这有效地确保了对变量重命名的泛化（Ferreira 等人，2022）； <nat1> 直到<natN> 表示数字； <def> 表示定义，并结合了从 Gu 等人（2016）改编的复制机制。请注意，对于定理及其证明中的每个唯一数值常量或变量字面量，都会引入一个不同的通用词元，并在 Coq 版本中使用相应的通用词元。例如，考虑图 3 中的⟨LaTeX, Coq⟩ 配对， <nat1>、 <nat2>、<nat3>和 <nat4> 将分别用于替换常量 2、35、7 和 5，并且在 LaTeX 和 Coq 语句中均如此。同样， <var1>、<var2>和 <var3> 用于替换变量字面量 w、 R和 Q。这与在所有地方仅使用两个通用词元 <nat> 和 <var> 形成对比，后者会使所有数字都形成共指，所有变量也都形成共指。初步实验验证了在 LaTeX 和 Coq 语句中保留正确共指关系的同时，对这些区别进行编码的效用。

总体而言，通过对数字、变量和定义使用通用词元，只需训练有限数量的嵌入——而且模型必须利用上下文信息，才能恰当地将词元复制到 Coq 输出中。这样一来，模型就能够泛化到未见过的数字以及变量名和定义名。

原始的 CopyNet（Gu 等人，2016）采用了带有复制机制的编码器-解码器架构，用于计算生成词表内词元与将输入序列中的词元复制到输出中的概率。我们的自动形式化任务保证生成（g）词元与复制（c）词元互斥，因此可以使用简化公式计算生成词元的概率 y_t 在时间步 t。令 $\mathcal{V}_c$ 表示 Coq 词表， X 表示 LaTeX 词元输入序列， $\mathcal{X}$ 表示 X中的唯一词元集合，则生成 y_t 的概率计算如下：

$$p(y_t) = \begin{cases} p(y_t, \mathrm{g}) = \dfrac{1}{Z_t} e^{\psi_g(y_t)}, & y_t \in \mathcal{V}_c \\ p(y_t, \mathrm{c}) = \dfrac{1}{Z_t} \displaystyle\sum_{x_j \in X : x_j = y_t} e^{\psi_c(x_j)}, & y_t \in \mathcal{X} \end{cases}$$

where $Z_t = \displaystyle\sum_{y_t \in \mathcal{V}_c} e^{\psi_g(y_t)} + \sum_{x_j \in X} e^{\psi_c(x_j)}$. The scor-

生成函数定义为 $\psi_g(y_t) = \mathbf{v}_{y_t}^\top \mathbf{W}_o \mathbf{s}_t$ 和 $\psi_c(x_j) = \tanh\left(\mathbf{h}_j^\top \mathbf{W}_c\right)\mathbf{s}_t$，其中 $\mathbf{v}_{y_t}$ 是 y_t 的独热编码，$\mathbf{h}_j$ 是输入词元 x_j 对应的编码器隐藏状态，$\mathbf{s}_t$ 是步骤 t 的解码器状态，且 $\mathbf{W}_o$ 和 $\mathbf{W}_c$ 是可学习参数。

3.2 编码器-解码器架构

我们对标准 Transformer 架构做了几项关键改动：

- 概率通过上文的 $p(y_t)$ 计算。
- 移除了绝对位置编码。
- 自注意力使用相对位置表示，如Shaw 等人 (2018年所述。
- N 编码器/解码器块堆栈包含 T 循环传递。

模型的其他所有方面均与原始 Transformer 保持不变。在我们的模型架构中，我们强调相对位置信息，而非绝对位置信息。对奇偶-数据集的初步评估表明，在训练时未见过的表达式长度上，使用绝对位置编码的 Transformer 模型序列级准确率为 0%。消除对绝对位置的依赖，便可解决这类系统泛化问题。因此，在基于 Transformer 的模型中使用相对位置编码，对于实现更强的系统泛化至关重要；这也与Csordás 等人(2021)在其他自然语言处理任务中的研究发现一致。

4 实验评估

为了评估训练后模型的性能，我们进行了两项主要实验：首先针对算术示例集合，然后针对代码正确性示例集合。所有模型均以序列级准确率进行评估；只有当生成的定理及其证明对应的 Coq 序列与真实序列逐个词元完全匹配时，才认为示例处理正确。我们还报告语义级 准确率：生成的 Coq 定理必须达到完美的序列级准确率，并且 Coq 引擎必须验证生成的 Coq 证明确实证明了生成的 Coq 定理，而不论其是否与真实证明版本匹配。这表明模型能够依据自然语言版本，正确翻译定理并成功给出证明，从而理解自然语言证明的大意

	奇偶		合数		POLY
n	Seq	Sem	Seq	Sem	Both
2	99.6	99.8	76.7	97.6	100.0
3	99.4	99.6	64.6	94.2	100.0
4	99.4	99.4	56.1	93.9	82.1
5	99.2	99.6	54.9	94.4	99.2
6	98.8	98.8	57.1	94.3	45.1
7	99.1	99.5	58.5	93.4	96.5
8	93.8	94.0	53.5	88.3	15.7
9	98.6	98.6	53.7	93.7	98.2
10	7.0	7.0	1.2	1.6	35.6
11	0.0	0.0	0.0	0.0	93.5
12+	0.0	0.0	0.0	0.0	0.0

幂 序列 = 100.0 语义 = 100

表1：测试样例的序列级（Seq）和语义级（Sem）准确率（%），按表达式长度划分，POWERS 除外。

并以自然语言版本为指导。

所有实验均在一块配备 48GB 内存的英伟达 RTX-A6000 GPU 上完成。

4.1 算术陈述

我们在完整数据集上评估 Transformer 模型，该数据集结合了 EVEN-ODD + COMPOSITES + POWERS，并在每个序列中同时使用定理及其证明。我们调优的模型嵌入和状态大小均为 32，前馈层宽度为 256，包含 4 个编码器和解码器模块、4 次循环传递、4 个注意力头，自注意力的裁剪值为 2。我们使用大小为 20 的小批次训练该模型，采用 Adam 优化，并使用 $\beta_1 = 0.9$、$\beta_2 = 0.98$、$\varepsilon = 1e-9$；初始学习率为 0.001，根据训练损失平台期按 $1/\sqrt{10}$ 的因子进行退火，耐心值为 5 个训练轮次。

表1 中的结果表明，模型能够很好地泛化到 $\{4, 6, 8\}$ 的中间长度，少量正确翻译的样例甚至超过了训练中使用的最大长度 9。除此之外，模型无法泛化到更长的未见长度，这并不令人意外，因为已知 Transformer模型 在多种自然语言处理任务中面对较长输入时，系统泛化能力会显著失效（Csordás 等人，2021）；在较长的符号积分问题中，准确率也会大幅下降（Welleck 等人，2022）。改用语义层面评估后，合数的准确率显著提高，而奇偶的提升幅度则较为有限。

4.2 代码正确性陈述

我们将研究范围扩展至使用霍尔逻辑语言表示程序正确性证明的数据。我们训练了一个单独的模型，其嵌入维度、状态规模、前馈层宽度和学习率均与第 4.1 节相同。模型深度增加至 8 个编码器和解码器模块，进行 8 次循环传递，使用 8 个注意力头，裁剪值为 8。模型使用 Adam，以大小为 1 的小批次进行训练，耐心值设为 3 个训练轮次。

表1 所示的 POLY 结果表明，模型能够泛化到训练期间未见过的程序行数 $\{4, 6, 8, 10\}$；随着程序长度增加，收益逐渐递减，最终对于超过训练期间最大长度的程序，模型将无法泛化。我们观察到，增加模型深度显著提升了泛化能力。与算术实验具有相同超参数的模型，在中等程序长度上的序列级准确率不到其一半。因此，进一步增加模型深度或许能使模型在中等长度上的表现更接近最优泛化水平，但代价是需要显著更多的计算资源。此外，POLY 示例也远不易出现不会导致致命后果的词元交换错误。我们观察到，语义级准确率与序列级准确率完全相同，因为所有复制错误都会破坏证明的有效性。因此，准确率合并为一列展示（两者）。

4.3 手写示例

我们还评估了训练后模型在由45个人编写的 LaTeX 定理-证明对组成的数据集上的语义级准确率。具体做法是人工核验生成的 Coq 定理是否与 LaTeX 版本对应，以及后续证明是否符合 Coq 解释器的要求。经过完整训练的模型在奇偶和合数任务上的准确率均为 53.3%，在幂任务上的准确率为 73.3%。

几乎所有情况下，错误都局限于对词汇表外词元的处理不当，例如在定义中抄错变量，或遗漏证明中与某个项相关的断言。除此之外，模型生成的 Coq 代码在语法上都是正确的。错误与那些明显偏离人工数据语法结构的示例有很强的相关性。因此，正如Wu 等人（2022）评估的预训练语言模型，或是在数学语料上预训练的新模型（例如 MATH（Hendrycks 等人，2021））或许能够缓解自然语言与形式数学数据稀缺且缺乏对齐所造成的问题。

5 结论

正如我们所见，在范围非常有限的数学和代码正确性证明领域中，训练机器学习模型执行自动形式化是可行的。这些模型展现出系统地泛化至新的表达式长度和程序规模的能力。此外，这些模型能够翻译此前未见过的手写自然语言示例，尽管准确率较低。我们希望这种方法能够应用于更大范围的数学自动形式化和代码验证。

正如Szegedy(2020) 所言：“自动形式化不仅是一项挑战：成功实现自动形式化将代表着通用人工智能的一项突破，并对各个领域产生重大影响。”我们认为，自动形式化对教育的影响尤为重大：将其整合到初等数学和软件验证课程所使用的证明助手中，有助于发现学生证明中遗漏的步骤或存在的错误理解。

参考文献

K. Appel 和 W. Haken。1977。每个平面图都可以四色染色，第一部分：放电法。《伊利诺伊数学杂志》，21(3):429 – 490。K. Appel、W. Haken 和 J. Koch。1977。每个平面图都可以四色染色。第二部分：可约性。《伊利诺伊数学杂志》，21:491–567。Grzegorz Bancerek。2006。形式化数学中的自动翻译。《机械化数学及其应用》，5(2):19–31。Yves Bertot 和 Pierre Castéran。2013。交互式定理证明与程序开发：*Coq'Art*：归纳构造演算。施普林格科学与商业媒体。Benjamin Andrew Carman。2021。将 *LaTeX* 翻译为 *Coq*：一种用于形式化自然语言证明的循环神经网络方法。博士论文，俄亥俄大学。

Róbert Csordás、Kazuki Irie 和 Juergen Schmidhu- ber。2021。细节决定成败：简单技巧提升 Transformer模型的系统泛化能力。载于*2021* 年自然语言处理实证方法会议论文集，第619–634页。

Mostafa Dehghani、Stephan Gouws、Oriol Vinyals、Jakob Uszkoreit 和 Łukasz Kaiser。2018年。通用 Transformer。*arXiv*预印本 *arXiv:1807.03819*。

Deborah Ferreira、Mokanarangan Thayaparan、Marco Valentino、Julia Rozanova 和 Andre Freitas。2022。要成为整数，还是不要成为整数？数学文本中的变量编码。载于计算语言学协会发现：*ACL 2022*，第 938–948 页，都柏林，爱尔兰。计算语言学协会。

G. Gonthier。2013。工程数学：奇数阶定理的证明。载于*ACM* 编程语言原理年度研讨会会议录，第 1–2 页。被引用 12 次。

Georges Gonthier。2008。形式证明——四色定理。《美国数学学会通报》，55(11):1382–1393。

Georges Gonthier、Andrea Asperti、Jeremy Avigad、Yves Bertot、Cyril Cohen、François Garillot、Stéphane Le Roux、Assia Mahboubi、Russell O'Connor、Sidi Ould Biha、Ioana Pasca、Laurence Rideau、Alexey Solovyev、Enrico Tassi 和 Laurent Théry。2013。奇数阶定理的机器验证证明。载于交互式定理证明，第 163–179 页，柏林，海德堡。Springer Berlin Heidelberg。

Jiatao Gu、Zhengdong Lu、Hang Li 和 Victor OK Li。2016 年。在序列到序列学习中引入复制机制。载于第 *54* 届计算语言学协会年会论文集（第 *1* 卷：长篇论文），第 1631–1640 页。

Thomas Hales、Mark Adams、Gertrud Bauer、Tat Dat Dang、John Harrison、Le Truong Hoang、Cezary Kaliszyk、Victor Magron、Sean McLaughlin、Tat Thang Nguyen 等。2017 年。开普勒猜想的形式证明。《数学论坛：*Pi*》，5:e2。

Thomas C. Hales。2005 年。开普勒猜想的证明。*Ann.Math. (2)*，162(3):1065–1185。

Dan Hendrycks、Collin Burns、Saurav Kadavath、Akul Arora、Steven Basart、Eric Tang、Dawn Song 和 Jacob Steinhardt。2021 年。使用数学数据集衡量数学问题求解能力。*NeurIPS*。

Nicholas J. Higham。2020年。《数学科学写作手册》，第三版。工业与应用数学学会，宾夕法尼亚州费城，美国。

Andrew David Irvine 和 Harry Deutsch。2021。《罗素悖论》。载于 Edward N. Zalta 主编的《斯坦福哲学百科全书》，2021年春季版。斯坦福大学形而上学研究实验室。

Xavier Leroy。2009。《现实编译器的形式化验证》。《ACM通讯》，52(7):107– 115。

Tobias Nipkow、Lawrence C Paulson 和 Markus Wenzel。2002 年。*Isabelle/HOL*：用于高阶逻辑的证明助手，第 2283 卷。施普林格科学与商业媒体。Benjamin C Pierce、Chris Casinghino、Marco Gaboardi、Michael Greenberg、C˘在˘alin Hri.tcu、Vilhelm Sjöberg 和 Brent Yorgey。2010 年。软件基础。网页：*http://www. cis. upenn. edu/bcpierce/sf/current/index. html*。

Benjamin C. Pierce、Arthur Azevedo de Amorim、Chris Casinghino、Marco Gaboardi、Michael Greenberg、C˘at˘alin Hri.tcu、Vilhelm Sjöberg、Andrew Tolmach 和 Brent Yorgey。2018年。编程语言基础。软件基础丛书，第2卷。电子教材。版本5.5。http://www.cis.upenn.edu/~bcpierce/sf。

Peter Shaw、Jakob Uszkoreit 和 Ashish Vaswani。2018年。带相对位置表示的自注意力。载于*2018*年北美计算语言学协会会议论文集：人类语言技术，第*2*卷（短篇论文），第464–468页，新奥尔良，路易斯安那州。计算语言学协会。

Christian Szegedy。2020。迈向自动形式化与通用人工智能的一条有前景的道路。载于《智能计算机数学》。*CICM 2020*。《计算机科学讲义》，第12236卷，第3–20页。Springer。

Thomas Tymoczko。1979。四色问题及其哲学意义。《哲学杂志》，76(2)：57–83。

Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017。注意力机制就是你所需要的一切。《神经信息处理系统进展》，30：5998–6008。

Qingxiang Wang、Chad Brown、Cezary Kaliszyk 和 Josef Urban。2020。在 Mizar 中探索神经机器翻译在数学自动形式化中的应用。载于第九届 *ACM SIGPLAN*认证程序与证明国际会议论文集，CPP2020，第85–98页，美国纽约州纽约市。计算机协会。

Qingxiang Wang、Cezary Kaliszyk 和 Josef Urban。2018年。将非形式数学神经翻译为形式数学的首次实验。载于智能计算数学，第255–270页，Cham：Springer International Publishing。

Sean Welleck、Peter West、Jize Cao 和 Yejin Choi。2022。序列模型中的符号脆弱性：符号数学中的系统泛化。人工智能 *AAAI* 会议论文集，36(8):8629–8637。期号：8。

Yuhuai Wu、Albert Q Jiang、Wenda Li、Markus N Rabe、Charles Staats、Mateja Jamnik 和 Christian Szegedy。2022。借助大型语言模型实现自动形式化。*arXiv* 预印本 *arXiv:2205.12615*。