计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(1):273−278 [doi: 10.15888/j.cnki.csa.008258]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向形式化证明的命令生成技术①

莫广帅[1], 熊　焰[2], 黄文超[2]

[1](中国科学技术大学 网络空间安全学院, 合肥 230027)
[2](中国科学技术大学 计算机科学与技术学院, 合肥 230027)
通信作者: 黄文超, E-mail: huangwc@ustc.edu.cn

摘　要: 随着软件规模的不断增大, 软件安全问题日益严重. 作为软件系统安全检测的有效手段, 形式化证明旨在利用数学方法完成对软件属性的严格验证. 常用的形式化证明方法利用模式匹配来进行定理证明, 但存在策略生成不完备等缺陷. 本文提出一种基于注意力机制的命令预测框架, 将 LSTM 与 Coq 结合, 预测定理证明过程中的策略和参数. 实验结果表明本文提出的模型在生成命令的准确度方面高于现有工作 (本工作预测命令准确率为 28.31%).

关键词: 形式化证明; Coq; 命令预测; LSTM; 注意力机制

引用格式: 莫广帅,熊焰,黄文超.面向形式化证明的命令生成技术.计算机系统应用,2022,31(1):273–278. http://www.c-s-a.org.cn/1003-3254/8258.html

Command Generation Technology for Formal Proof

MO Guang-Shuai[1], XIONG Yan[2], HUANG Wen-Chao[2]

[1](Cyberspace Security, University of Science and Technology of China, Hefei 230027, China)
[2](School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

Abstract: With the continuous increase in software scale, software security faces increasingly severe challenges. As an effective means of detecting software system security, formal proof aims to use mathematical methods to complete rigorous verification of software attributes. Commonly used formal proof methods prove theorems with pattern matching, which, however, suffer from defects such as incomplete strategy generation. This study proposes a command prediction framework based on the attention mechanism. It combines long short-term memory (LSTM) with Coq to predict the strategies and parameters during theorem proving. The experimental results show that the model proposed in this study is superior to existing ones in the accuracy of command generation (the accuracy of command prediction is 28.31% in this paper).

Key words: formal proof; Coq; command prediction; LSTM; attention mechanism

近年来, 随着软件规模的快速增大, 软件系统安全问题日益增多. 因此, 开发者需要在软件开发的过程中减少软件漏洞, 保障软件安全. 形式化证明利用数学方法对软件属性进行建模, 验证软件是否满足相应的安全目标. 形式化方法已成为构建高可信软件系统的有效手段.

通常检测和分析大规模软件系统需要使用具有高阶逻辑描述能力的工具, 如 Coq[1]、Isabelle HOL[2,3]. 在证明工具中, 证明人员使用策略与工具交互证明定理. 但是, 这种证明方式存在一定局限性, 需要消耗证明人员大量的时间和精力. 因此, 实现定理证明的自动化是必不可少且有意义的工作.

目前, 研究者提出了许多自动化证明工作. Paulson 等[4] 利用现有的 ATP 系统证明定理. Gransden 等[5] 从

① 基金项目: 国家自然科学基金 (61972369)
收稿时间: 2021-03-23; 修改时间: 2021-04-19, 2021-04-26; 采用时间: 2021-04-28; csa 在线出版时间: 2021-12-17

已经编译成功的证明库中匹配当前的上下文状态. Alemi 等[6] 第一次将深度学习算法应用于大规模的定理证明. 但是这些工作存在向量信息不足、命令生成不完备等问题.

现有的自动定理证明工作面临两个挑战. 1) 策略空间大. 在 Coq 中, 证明人员需要从 221 个策略库中生成合适的策略. 2) 参数结构复杂. 策略参数可以由全局表达式、复杂的代码行和全局中搜索出的中间引理组成.

针对定理证明工具中的命令生成问题, 本文提出了以下工作: 1) 将机器学习与 Coq 结合, 利用 LSTM 等网络预测证明中的策略和参数. 2) 结合注意力机制, 自动关注在训练过程中起决定作用的特征, 捕捉重要的信息, 减少无用信息的关注.

1 相关工作

本节介绍近年来自动化证明方面的研究工作. 现有的自动化证明工作可以分为结合自动定理证明和结合交互式证明两类.

1.1 结合自动定理证明

随着自动化定理证明和机器学习的发展, 在不需要手工设计特征的情况下, Alemi 等[6] 首次证明了神经网络模型有助于进行大规模自动逻辑推理, 并第一次将深度学习应用于大规模的定理证明. Wang 等[7] 提出了一种基于深度学习的前提选择方法. 他们将一个高阶逻辑公式表示为一个图, 完全保留语法和语义信息. 然后为图的每个节点指定一个初始嵌入向量, 将所有节点的嵌入集合起来形成整个图的嵌入. 这种方式提高了前提选择的准确性. 但是, ATP[8] 仍然需要将定理转换为合取范式 (CNF), 复杂的软件系统依旧无法证明.

一些工作将现有的交互式定理证明与自动定理证明结合, 允许用户直接使用现有的 ATP 系统. Paulson 等[4] 将 Isabelle 证明助手中的定理转换为一阶逻辑, 之后使用 ATP 分析定理, 并将证明转换回 Isabelle 中, 从而证明定理. 类似的“理论”也被用于其他证明工具 (Kaliszyk 等[3]; Urban 等[9]; Czajka 等[10]). 这种证明方式基本上绕过了交互式证明工具, 并将工作外包给外部 ATP. 然而, 这种证明方式最大的缺陷在于定理转化过程中可能会导致属性的缺失.

ATP 将定理的前提和目标转换为合取范式 (CNF) 中的一阶子句, 并通过证明一阶子句来证明定理. ATP 的优点在于提高了定理证明过程的自动化程度, 缺陷在于它很难完成复杂软件系统功能正确性的全部验证. 针对软件系统复杂的数据类型和逻辑, ATP 只能处理有限的情况.

1.2 结合交互式定理证明

在交互式定理证明中, 开发人员利用证明工具手动编写证明脚本来完成属性的验证. 交互式定理证明的优势在于工具能够为软件的验证产生证明过程, 此外, 手动编写脚本使得复杂软件的验证成为可能. 但是, 交互式定理证明存在自动化程度低、验证代价比较高等缺陷.

机器学习算法的发展为定理自动证明提供了新的思路. 一些工作注重与机器学习的交互: Komendantskaya 等[11] 侧重于 ITP 和机器学习接口之间的交互式接口方法. 但是这项工作并未尝试生成证明, 就其接口而言, 重点在于是使许多没有机器学习经验的用户可以使用许多简单但有用的选项. 同样的还有 Huang 等[12] 提供了 Coq 证明的结构化 Python 表示, 包括证明中遇到的所有证明状态、采取的步骤和表达式抽象语法树 (ASTs), 还支持与 Coq 的轻量级交互, 以便它可以用于动态构建证明. 但无法构建复杂的策略以便生成证明.

张恒若等[13] 在 2017 年基于 Z3 设计了 Coq 自动证明的实现. 其将 Coq 与 Z3 约束求解器结合在一起, 设计了一个编译框架, 将 Coq 中的定理翻译为 Z3 可判定的明提. 最后在 Coq 中能够调用 Z3 中的命令进行证明演算.

但是, 张恒若等人工作缺陷在于:

1) Coq 与 Z3 的类型系统存在巨大差异, 无法准确转换定理表达式.

2) Coq 中的很多基本类型是开发者直接利用递归定义的, 但在 Z3 中有其提供的基本类型. 例如, Int (自然数) 在 Coq 中需要开发者用递归的方式手动定义, 但在 Z3 中提供对应的 Int 类型. 如果将 Coq 中的类型转换为 Z3 中的类型, 其解析证明的效率将会降低.

另一些工作利用机器学习产生证明选项证明定理: Gauthier 等[14] 通过从数据集中搜索少量候选策略来生成策略集, 这些候选策略根据当前需要证明的定理来生成. 每个候选策略视为一个可能的行为, 并通过学习值函数和蒙特卡罗树搜索进行评估, 最后选出需要的策略. 虽然比 SEPIA 更具适应性, 但生成的策略仍然是

从一个具有预定的固定集合中选择的, 并且难以将其推广到数据集外的其他策略.

Yang 等[15] 通过与证明助手的交互, 为定理证明提供了一个大规模的数据集和学习环境. 其所使用数据集包括来自 Coq 证明助手中的 123 个开源软件项目的 71K 人工证明 (Barras 等). 数据集涵盖了广泛的应用领域, 包括数学, 计算机硬件, 编程语言等. Yang 等以抽象语法树的形式生成策略, 可以用来证明以前的自动证明器所不能达到的新定理.

2 方案设计

本节将介绍系统的一些设计细节. 图 1 是预测命令的框架图. 分为 4 个模块: 数据提取模块、向量构建模块、预测参数模块和命令结合模块.

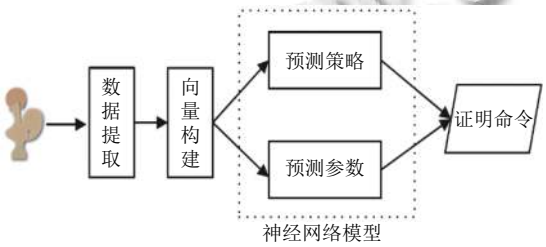

图 1 预测命令框架

本节针对命令预测问题进行抽象. 首先, 对于命令预测, 本工作从 Coq 证明助手中提取证明数据, 包括前提、目标和一些结构化信息. 之后, 这些前提和目标以及其他的特征将被编码成高维向量, 经过压缩降维之后, 作为神经网络模型输入. 机器学习模型将会分别预测策略和参数. 最后, 本工作结合预测和参数形成证明命令.

2.1 问题模型

本工作将$K[\gamma]$定义为γ预测概率, 其中, γ表当前的状态 (当前前提、目标、策略). R 代表状态空间.

$$K[\gamma]=\gamma\to R \tag{1}$$

在证明状态$\gamma\in S$, $\gamma-predictor\ K[\gamma]$被定义为在$\gamma$上预测的概率的函数, 其预测出的策略或者参数:

$$E[\gamma]=S\to K[\gamma] \tag{2}$$

预测器 P 代表针对下一个证明状态的预测. Γ代表策略的预测, Λ代表参数的预测:

$$\mathrm{P}: E[\Gamma\times\Lambda] \tag{3}$$

本工作的机器学习模型将会对当前的状态做两种预测, 一种是策略P_{tac}, 一种是参数P_{arg}. 数学表示为:

$$P_{\mathrm{tac}}: E[\Gamma] \tag{4}$$

$$P_{\mathrm{arg}}: \Gamma\to E[\Lambda] \tag{5}$$

$P(\sigma)$代表模型预测的命令, 将策略P_{tac}和参数P_{arg}结合在一起.

$$P(\sigma)=\lambda(\gamma,\alpha)P_{\mathrm{tac}}(\alpha)(\gamma)P_{\mathrm{arg}}(\gamma)(\sigma)(\alpha) \tag{6}$$

2.2 信息提取

Coq 将证明逻辑信息放在底层, 并不在 IDE 显示. 此外, 由于定理的结构过于复杂, 其证明信息可能无法直接解析使用. 本工作中, 神经网络模型直接提取命令行中的信息构建证明.

2.3 向量构建

Coq 证明助手以类型为基础, 如果直接使用证明工具中的数据作为编码, 可能导致数据冗余、训练效果差等情况. 例如, ComCert 项目中的一个定理:

Theorem identity_fn_applied_twice:
Forall (f : bool -> bool),(forall (x : bool), fx = x) ->
Forall (b : bool), f(f b) = b.

在证明过程中, 存在两个子目标:

f (f true) = true

f (f false) = false

两个子目标的证明脚本为:

rewrite <- H.

rewrite <- H.

reflexivity.

如果直接编码子目标, 由于参数改变, 无法有效学习第一个子目标中的经验, 利用定理中类型替代参数是一个有效的选择. 因此, 本工作所使用的数据特征有定理名称、定理主体和参数类型.

2.4 预测策略

为了预测策略, 本工作收集现有的 Coq 证明库和常见的项目组成数据集. 为反应策略预测的侧重点, 本工作主要使用以下几个数据特征:

1) 当前需要证明的目标; 很多策略往往是根据目标具体选择的. 例如, 如果定理以 forall 开头, 那么下一个策略可能是 intro 或 induction.

2) 上一步的策略; 定理证明中策略的使用具有空间局部性. 一些策略之后往往使用相同的策略. 例如, 如果前一个策略是 intro, 后面经常使用 auto; 而 inversion 经常在 subset 后使用.

3) 当前上下文中的前提. 策略的局部参数存在当

前的上下文参数中. 因此, 当前的前提需要作为预测策略的特征. 考虑到下面的前提:

H : is_path(cons (pair s d) m)

这种情况下, 机器学习模型可能预测 inversion H 或者 rewrite->H, H 代表上下文中的假设.

所有的这些特征被编码成一个高维的向量后, 经过 PCA 降维成一个 128 维的向量, 然后送入一个每层拥有 128 个节点的完全连接的 3 层前馈神经网络 (feed-forward neural network), 计算策略的概率分布. 图 2 是针对策略的预测模型, 目的是找出策略在策略分布图中的权重.

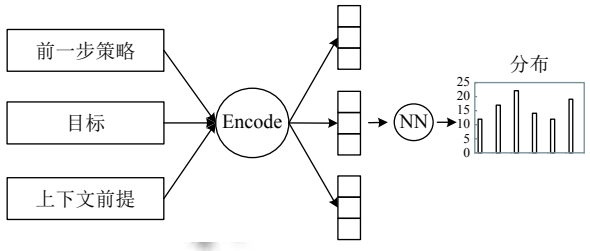

图 2 预测策略框架

用于计算节点的前馈神经网络有 3 层: 输入层, 隐藏层与输出层.

对于给定的参数集合W,b, 输出层的输出 h 为:

$$h(x) = f(W_{11}a_1 + W_{12}a_2 + W_{13}a_3 + b_1) \tag{7}$$

其中, a_1,a_2,a_3与输入向量有关. 前馈神经网络的计算节点为 GRU, 相较于 LSTM, GRU 的模型计算效率更高. 本工作只生成原子策略, 而不包括“tac1; tac2”等复合策略. 因为所有的定理都可以在没有复合策略的情况下完成.

2.5 预测参数

图 3 是预测参数的总体框架. 将相应的特征编码为向量后, 送入到机器学习模型中后, 训练得到命令的分布.

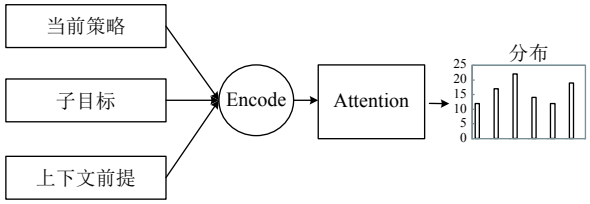

图 3 预测参数框架

深度学习模型预测 3 种参数: 1) 假设中参数; 2) 目标中参数; 3) 中间定理.

预测参数模型使用以下几个特征: 1) 预测的策略; 2) 当前子目标; 3) 当前前提.

与预测策略相比, 预测参数模型并没有使用前一步策略这个特征. 基于观察, 当前使用的参数与上一步使用的命令并不存在关联. 此外, 当前使用的参数与当前的策略存在一定关联性.

例如, 如果是原子性策略, 那么策略就不需要模型预测参数; intro 与 rewrite 使用不同的参数. 因此, 预测参数模型使用了当前策略这一特征.

与预测策略相似, 将当前特征编码为 128 维的向量表示. 经过 PCA 降维之后, 送入到神经网络中训练. 最后对当前状态进行预测, 得到当前参数的分布.

2.6 注意力网络

为解决自动定理证明中的策略参数预测问题, 本工作引入注意力机制, 注意力机制可以自动关注在训练过程中起决定作用的特征, 捕捉重要的信息, 减少无用信息的关注.

注意力机制包含两部分结构: 编码部分 (encoder) 和解码部分 (decoder). 本工作选取 encoder 和 decoder 都为 LSTM. 这是因为, 相较于 RNN, LSTM 采用一个 cell state 来保存长期记忆, 再配合门机制对信息进行过滤, 从而达到对长期记忆的控制.

在编码部分, h 代表各个时刻的隐藏层状态. 将隐藏层状态汇总, 生成编码向量 T. q 代表对应的权重.

$$T = q(h_1, h_2, h_3, \cdots, h_t) \tag{8}$$

在解码部分, $h_1 \sim h_t$ 代表编码器隐藏层状态, S_{i-1}代表解码器隐藏层状态. W_aU_a代表对应神经网络层的权重, e_{ij}代表$h_1 \sim h_t$与S_{i-1}的相关程度.

$$e_{ij} = a(S_{i-1}, h_j) = V_a^t \tanh(W_aS_{i-1} + U_ah_j) \tag{9}$$

其中, α_{ij}表示第 i 个输出中前一个隐藏层状态S_{i-1}与第 j 个输入隐层向量h_j之间的相关性. α_{ij}可以由e_{ij}传入到对应函数归一化权重值得到.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{t}\exp(e_{ik})} \tag{10}$$

对$h_1 \sim h_t$进行加权求和得到此次解码所对应的输出$(x_1, x_2, x_3, \cdots, x_t)$的编码向量$C_i$.

$$C_i = \sum_{j=1}^{t}\alpha_{ik}h_j \tag{11}$$

有了C_i之后, 根据编码向量C_i进行解码, 先计算解

码器 i 时刻的隐藏层状态S_i, 再计算解码器 i 时刻的输出Y_i计算如下:

$$S_i = f(S_{i-1}, Y_{i-1}, C_i) \tag{12}$$

$$Y_i = g(y_{i-1}, S_i, C_i) \tag{13}$$

2.7 结合策略和参数

在预测策略和参数之后, 本工作结合策略和参数形成证明命令.

现有的工作使用贪婪搜索结合策略和命令. 在这种结合方式将选择概率最高的预测策略做证明. 同样, 机器学习模型选择概率最高的参数. 算法可以通过赋予第一个参数较高的权重来实现这种选择. 但是, 这种方法的缺陷在于策略的最终选择没有考虑到参数与策略是否匹配. 例如, 预测模型可能会预测 rewrite, 但却发现没有合适的前提可以用来作为参数.

本工作并没有将策略和参数的概率进行标准化等处理, 而是直接选出评分较高的策略, 依据策略选择最高概率的参数. 这种方式提供了策略和参数的平衡, 将两者都考虑在内. 此外, 可以根据策略的类型, 来判断是否使用相应的参数, 进行一些后期的选择处理. 本部分的实验将会在 ComCert 项目上进行实验. 这部分会在相应的实验章节展示.

3 实验分析

本工作利用机器学习与 Coq 进行信息交互, 实现了在交互式定理证明工具中自动生成证明命令的框架. CompCert 是验证 C 语言编译器安全属性的形式化项目. 本小节预测 CompCert 项目中的命令, 测试本系统的生成命令的准确率.

实验的环境配置是 Intel Broadwell E5-2660V4 2.0 GHz CPU, 128 GB 的内存, 4 根 1080Ti 的 GPU, 操作系统是 Ubuntu 16.04 LTS.

本工作通过训练来自 CompCert 的 162 个文件的证明数据并以其中 13 个文件作为测试集. 测试集包含 501 个证明脚本, 一共 13 867 个证明命令, 如表 1.

表 1 预置文件

训练文件	测试文件	定理数	命令数
162	13	501	13 867

在所有的证明命令中, 本工作能够预测 28.31% (3926/13 867) 的命令, 而在 proof 证明命令中, 本工作能够预测 41.6% (3968/9537) 的命令. 证明命令包含策略和参数. 与其他工作相比, 本工作命令准确率提高 2%. 结合注意力机制, 本工作对证明信息的向量构建更丰富, 机器学习模型更合适.

3.1 策略预测结果

在预测策略时, 本工作测试了原始证明中的证明策略在前 3 个预测和前 5 个预测中出现的频率. 如表 2. 对于测试集证明中的策略, 在前 3 个预测中存在正确的策略占 58. 4%, 在前 5 个预测中存在正确的策略占 65. 6%. 此外, 从表中可以得知, 一些常用的策略如 intros, simpl 等预测的较为准确, 但是一些比较复杂的策略 inversion 预测效果较差. 简要分析, 有两个方面的原因, 一是这些策略使用的目标场景复杂. 二是训练集中这些策略较少.

表 2 预测策略

实际策略	预测1	占比 (%)	预测2	占比 (%)	预测3	占比 (%)
Induction	Intros	99	Simpl	0.7	Eauto	0.2
Simpl	Simpl	**79.1**	Intros	20.3	Intuition	0.1
Intros	Intros	**99.5**	Eauto	0.3	congruen	0.2
Auto	Eauto	50.7	traceEq	43.1	Easy	3.7
Simpl	Intros	50.5	Simpl	49.4	Eauto	0.1
Intros	Intros	**99**	Eauto	0.01	Intuition	0
Inversion	Eauto	95.5	Easy	2.2	Simpl	1.4

表 3 为预测全部策略的次数的百分比以及正确的概率, 表中为真实策略、策略的预测次数占比以及正确的百分比. 例如 simpl 策略在所有的预测数量中占 10%, 正确的概率为 7%.

表 3 策略预测正确率

策略	次数占比 (%)	正确率 (%)
Simpl	10	7
Intros	27	58
Auto	45	6
Congruence	41	7
Eauto	42	5

3.2 参数预测结果

在预测命令时, 本工作预测了测试集中 28.31% 的命令. 当本工作预测策略正确时, 命令中 29.5% 的参数是错误的. 然而, 需要注意的是, 许多常见的策略没有任何参数, 这些策略在证明的过程中占据了很大一部分比例.

在交互式定理证明工具中, 策略参数由中间定理、前提和一些复杂表达式组成. 例如命令 destruct (zie (pos0 + typesize ty0) pos), 策略参数为 zie (pos0 +

typesize ty0) pos. 这种复杂表达式组成的参数和当前上下文环境和目标定理有关, 现有的工作暂时无法生成. 此外, 这种形式的参数在手工定理证明中使用的较少, 机器学习模型预测效果较差. 复杂参数的一个处理方式是将对应的参数加载到当前证明的前提中, 以上下文参数的方式处理中间定理, 这些工作将会在未来的研究中探讨.

4 结论与展望

本文将机器学习与定理证明工具 Coq 结合, 分别预测策略和参数, 提高命令预测的准确度. 本文首先提取 Coq 中证明信息, 这些信息包括前提、目标和一些结构化数据. 然后, 将这些前提和目标以及其他的特征编码成高维向量, 作为神经网络模型的输入. 本文预测了测试集中 28.31% 的命令, 验证了本工作框架的有效性. 在未来的研究中, 本工作将会根据预测的命令形成完整的证明, 提高证明的效率.

参考文献

1 Gonthier G, Asperti A, Avigad J, *et al*. A machine-checked proof of the odd order theorem. Proceedings of the 4th International Conference on Interactive Theorem Proving. Rennes: Springer, 2013. 163–179.

2 Berghofer S, Nipkow T, Urban C, *et al*. Theorem proving in higher order logics. Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics. Munich: Springer, 2009.

3 Kaliszyk C, Urban J. Learning-assisted automated reasoning with flyspeck. Journal of Automated Reasoning, 2014, 53(2): 173–213. [doi: 10.1007/s10817-014-9303-3]

4 Paulson LC, Blanchette JC. Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. Workshop on Practical Aspects of Automated Reasoning (PAAR-2010). Edinburgh: PAAR, 2010. 1–10.

5 Gransden T, Walkinshaw N, Raman R. SEPIA: Search for proofs using inferred automata. Proceedings of the 25th International Conference on Automated Deduction. Berlin: Springer, 2015. 246–255.

6 Alemi AA, Chollet F, Een N , *et al*. DeepMath —Deep sequence models for premise selection. arXiv:1606.04442, 2016.

7 Wang MZ, Tang YH, Wang J, *et al*. Premise selection for theorem proving by deep graph embedding. Proceedings of the 31st International Conference on Neural Information Processing Systems. Red Hook: Curran Associates Inc., 2017. 2783–2793.

8 Bibel W. Automated Theorem Proving. Springer Science & Business Media, 2013

9 Urban J. MPTP —motivation, implementation, first experiments. Journal of Automated Reasoning, 2004, 33(3–4): 319–339. [doi: 10.1007/s10817-004-6245-1]

10 Czajka Ł, Kaliszyk C. Hammer for Coq: Automation for dependent type theory. Journal of Automated Reasoning, 2018, 61(1–4): 423–453. [doi: 10.1007/s10817-018-9458-4]

11 Komendanskaya E, Heras J, Grov G. Machine learning in proof general: Interfacing interfaces. arXiv: 1212.3618, 2012.

12 Huang D, Dhariwal P, Song D, *et al*. Gamepad: A learning environment for theorem proving. arXiv: 1806.00608, 2018.

13 张恒若, 付明. 基于 Z3 的 Coq 自动证明策略的设计和实现. 软件学报, 2017, 28(4): 819–826. [doi: 10.13328/j.cnki.jos.005196]

14 Gauthier T, Kaliszyk C, Urban J. Learning to reason with HOL4 tactics. arXiv:1804.00595, 2018.

15 Yang K, Deng J. Learning to prove theorems via interacting with proof assistants. International Conference on Machine Learning. Long Beach: PMLR, 2019. 6984–6994.