

WANG Heran, ZHAO Ming, DONG Xiaogang, GU Bin, LI Xiaofeng, ZHONG Ruiming. Evaluation Methods for Intelligent Synthesis Technology of Aerospace Control Software (in Chinese). *Chinese Journal of Space Science*, 2025, 45(3): 847–865. CSTR:32142.14.cjss.2024-0041. DOI:10.11728/cjss.2025.03.2024-0041

面向航天控制软件智能合成技术评价方法*

王鹤然 赵明 董晓刚

顾斌 李晓锋 钟睿明

(北京控制工程研究所 北京 100190)

摘要 程序合成是自动生成满足用户意图程序代码的软件开发活动,随着人工智能在程序合成领域的成功应用,智能程序合成技术逐渐成为软件开发的新范式。虽然现有一些智能程序合成技术的评价方法,但是仍面临许多问题需要进一步完善和改进。本文通过调研智能程序合成技术使用的评价标准以及分析当前主流智能程序合成技术的评价方法,分析并完善了智能程序合成技术的评价指标,结合航天嵌入式软件的特点,构建了航天嵌入式软件智能合成的层级式评价指标体系,设计了以动态和静态相结合为主的面向航天控制软件智能合成技术的综合评价方法。通过实验验证其中动静结合评价方法的有效性,其能够获得与人类评分相比更高的皮尔逊相关系数。

关键词 程序合成, 人工智能, 评价指标, 评价指标体系, 评价方法

中图分类号 TP311.5

Evaluation Methods for Intelligent Synthesis Technology of Aerospace Control Software

WANG Heran ZHAO Ming DONG Xiaogang

GU Bin LI Xiaofeng ZHONG Ruiming

(Beijing Institute of Control Engineering, Beijing 100190)

Abstract Program synthesis is a technique for automatically generating programs, which derives corresponding program code from given specifications or requirements. With the successful application of artificial intelligence in the field of program synthesis, intelligent program synthesis technology has become a new paradigm for software development. Although there are some evaluation methods for intelligent program synthesis technology, there are still many challenges that need further improvement and refinement. This paper summarizes and refines evaluation indicators for intelligent program synthesis technology by investigating the evaluation criteria used in intelligent program synthesis technology and analyzing the mainstream evaluation methods of intelligent program synthesis technology. Combined with the characteristics of aerospace embedded software, a hierarchical evaluation indicator system for

* 国家自然科学基金项目资助 (62192730, 62192735, U21B2015)

2024-03-14 收到原稿, 2024-04-25 收到修定稿

intelligent synthesis of aerospace embedded software is constructed, and a comprehensive evaluation method for intelligent synthesis technology of aerospace control software mainly based on dynamic and static combination is designed. By calculating the Pearson correlation coefficient with ChatGPT3.5 simulating human scores, it is found that the proposed combined dynamic and static evaluation method can obtain a higher correlation coefficient than either dynamic or static evaluation methods alone, and can reflect the improvement of performance after model iteration.

Key words Program synthesis, Artificial intelligence, Evaluation indicators, Evaluation indicator system, Evaluation methods

0 引言

随着中国航天事业的不断发展和进步,航天器的任务越来越复杂、航天器型号的更迭越来越快,应用于航天领域的嵌入式软件的系统规模和复杂性急剧上升。Yang 等^[1]认为嵌入式软件已经成为航天型号的重要组成部分和确保型号任务成功的重要因素。随着中国载人航天后续工程、探月后续工程等任务的相继立项和实施,航天嵌入式软件呈现出功能日益复杂、数量与规模激增、开发周期缩短、需求变化加快、质量要求更高的发展趋势。显然,传统的软件开发模式已经无法满足当下及未来嵌入式软件的开发需求。程序合成是自动生成满足用户意图程序代码的软件开发活动,具有极大的潜力来解决航天嵌入式软件开发过程中所面临的问题。并且随着人工智能在程序合成领域的成功应用,智能程序合成技术逐渐成为软件开发的新范式。将智能程序合成技术应用到航天领域中,能够更好地提高航天嵌入式软件开发的效率,减轻开发人员的负担,加速中国航天事业智能化的发

展。然而,现有智能程序合成技术的评价方法在评价维度、泛化能力以及实际适用性等方面仍存在亟待解决的问题。在航天领域中,智能程序合成技术的评价尤为重要,因为航天嵌入式软件的可靠性和稳定性对于任务成功具有至关重要的影响。因此,设计适用于航天领域的智能程序合成的评价方法势在必行,这将为航天嵌入式软件开发提供重要指导和支持,推动航天领域软件开发的智能化进程,如图1所示。

研究智能程序合成的评价方法,首先需要清楚评价指标、评价指标体系、评价方法三者的关系。评价指标是指用于评估系统、模型或过程性能、质量及有效性等方面的特定可量化标准或衡量维度;评价指标体系是指由多个相关评价指标组成的结构化框架或体系,用于全面评估系统、模型或过程的各个方面;评价方法是指用于收集数据、进行分析和判断以评估系统、模型或过程的方法或程序。评价指标通常是定量或定性的度量标准,而评价指标体系将不同的评价指标组织起来,以便系统地分析和评估被评估对象的综合表现。评价指标体系中的每个指标都对应对着评价

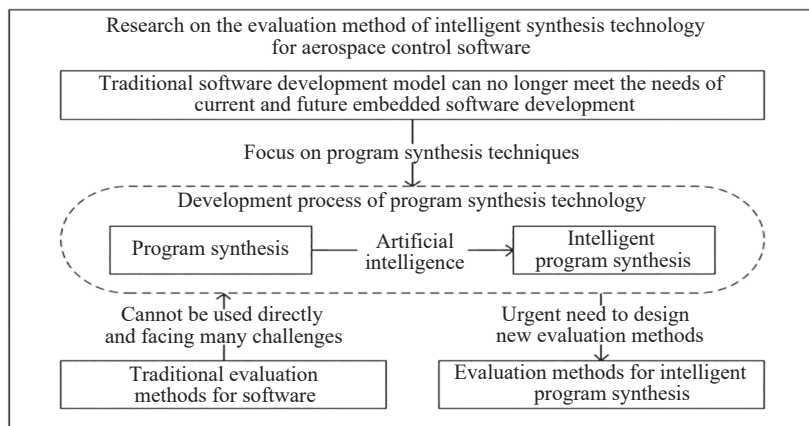


图1 研究背景

Fig. 1 Research background

指标体系中的一个特定方面, 通过综合考量这些评价指标的表现, 可以得到被评估对象整体情况的表现. 评价方法则提供了实际进行评价所需的操作步骤和技术手段, 其根据评价指标体系提供的评价指标来收集、分析和解释相关数据, 从而得到对被评估对象的评价结果. 评价指标、评价指标体系和评价方法三者相互依存, 共同构建了一个完整的评价体系. 为此, 提出了金字塔形的理论框架 (见图 2).

根据上述对评价指标、评价指标体系和评价方法的理解, 设计一个评价方法应该先选择合适的评价指标, 利用这些评价指标构建评价指标体系, 选择合适的手段和技术完成对评价方法的设计. 因此, 本文对智能程序合成评价方法从正向出发, 分析为验证智能程序合成有效性而使用的评价标准, 从这些标准中归纳出智能程序合成技术的评价指标, 为后续评价指标体系的构建提供支持. 采用逆向思维, 调研智能程序合成技术的主流评价方法, 学习这些评价方法中使用的评价指标或者评价指标体系以及如何使用评价指标对合成技术进行评价. 通过分析这些评价方法使用的技术手段及其各自的优缺点, 完善智能程序合成技

术的评价指标, 并为后续设计评价方法提供指导. 结合航天嵌入式软件的特点, 从完善后的智能程序合成技术的评价指标中选择合适的评价指标, 或者设计一些航天领域特有的评价指标来构建航天嵌入式软件智能合成的评价指标体系. 借鉴智能合成评价方法使用的技术手段, 设计面向航天控制软件智能程序合成技术的评价方法, 并通过实验验证设计的评价方法的有效性, 具体如图 3 所示.

1 智能程序合成技术的评价指标

调研国内外与智能程序合成技术相关的高质量文献共计 55 篇, 在检索文献时, 主要通过谷歌学术搜索, ACM Digital Library, IEEE Xplore, Springer Link Online Library 等, 使用关键词 Program Synthesis, Artificial Intelligence, Evaluation Metric 等进行检索并筛选出 2010 年至今的使用人工智能技术进行程序合成文献. 检索的文献主要是发表在软件工程/系统软件/程序设计语言 A 类国际学术会议 ES-EC/FSE, ASE, ICSE; 人工智能 A 类国际学术会议

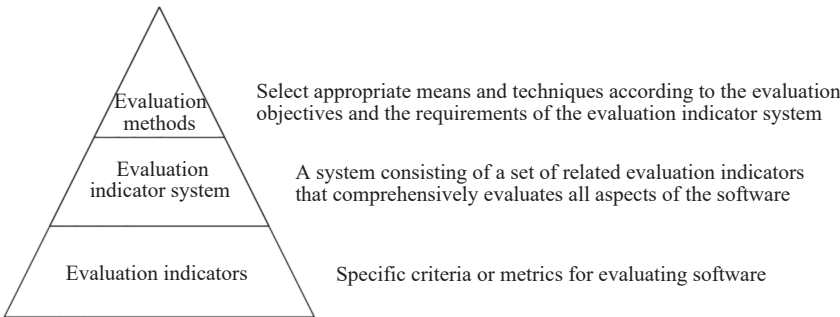


图 2 金字塔形理论框架
Fig. 2 Pyramid-based theoretical framework

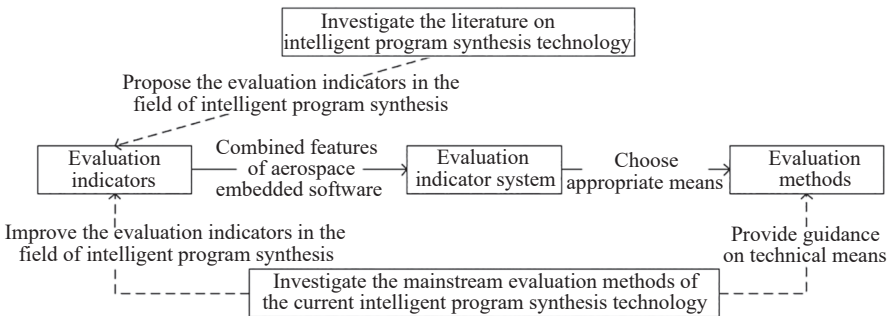


图 3 智能程序合成评价方法的主要研究内容
Fig. 3 Key research aspects in the evaluation methodologies of intelligent program synthesis

AAAI, NeurIPS, ACL, ICML; 并且额外选择了计算机科学理论 A 类国际学术会议 CAV 和 ICLR 会议.

通过分析智能程序合成技术的相关文献发现, 主要从合成程序的质量、合成程序的效率和训练使用的数据量三个维度出发, 验证合成技术的有效性, 即从合成的结果、合成的过程以及训练合成器的角度对合成技术进行评价. 从这 55 篇智能程序合成技术的相关文献中提取出验证合成有效性使用的评价标准, 通过对这些评价标准进行归纳, 得出程序正确性、程序规模、程序相似度、合成时间、候选程序数量和训练数据量共计 6 个评价指标. 使用这些评价指标的相关文献以及每个评价指标的使用频率如表 1 所示.

1.1 合成结果层面的评价指标

在智能程序合成技术的合成结果层面, 即根据合成程序的质量对合成进行评价, 得出智能程序合成技术的三个评价指标: 程序正确性、程序规模和程序相似度.

程序正确性是使用频率最高的评价指标, 其中一种计算方式是统计合成程序通过测试用例的比例, 即使用训练数据集训练合成器, 使用合成器合成的程序通过测试数据集的比例来判断合成的有效性. 例如, Shin 等^[2]使用 Top-K 泛化 (Top-K Generalization) 来判断合成的有效性, 即前 K 个合成程序中有多少比例的程序在用于指定程序的 5 个输入/输出示例以及 1 个保留输入/输出示例上具有正确的行为. Huang 等^[3]则认为满足一小组输入/输出示例的程序不一定能够满足数量更多的输入/输出示例, 最好的选择是将功能等价性作为评价标准, 但是判断功能等价性很困难. 为此, 使用随机生成的 95 个保留输入/输出示例来代替功能等价性, 可以更好地表示合成

程序的正确性. Ramani 和 Karande^[4]使用 $\text{pass}@k$ 作为评价标准, 即前 K 个合成的程序采样中, 至少有一个程序采样完全正确的比例, 这种方法需要为合成的程序采样设计完备的测试数据集; 另一种计算程序正确性的方式是判断合成程序解决问题的能力, 即在一定限制条件下解决任务/问题/基准的数量. 例如, Jain 等^[5]使用用户与系统交互时正确解决任务的百分比来判断合成的有效性. Christakopoulou 和 Kalai^[6]使用成功解决测试问题的比例来计算合成程序的正确性. Odena 等^[7]则考虑在规定候选表达式数量或者墙钟时间内解决基准的数量.

程序规模也是从合成结果的角度, 对合成器合成能力进行评价的评价指标, 主要是通过统计合成程序中标记或程序结构的数量、合成程序的长度等来计算的. 例如, Dumancic 等^[8]统计合成程序中字面常量和谓词的数量来判断合成器的合成能力. Rosin^[9]则关注的是合成程序中指令和循环的数量. Zohar 和 Wolf^[10]使用不同长度的程序对合成器进行训练, 通过比较合成程序的长度来判断程序合成的能力.

此外, 还有很多研究通过比较合成程序与参考程序的相似程度, 来判断合成程序的质量, 将这类评价标准归纳为程序相似度, 可以通过比较合成程序与参考程序中代码结构或函数调用的相似程度来计算. 例如, Murali 等^[56]通过分析合成的程序中是否具有目标抽象语法树来比较代码结构的相似程度、使用合成程序与参考程序中应用程序接口 (Application Programming Interface, API) 方法调用序列集合的平均最小杰卡德距离 (Jaccard Distance) 来比较函数调用的相似程度. 平均最小杰卡德距离主要用于衡量两个集合之间的相似度. 除此之外, 还可以将合成程序和

表 1 智能程序合成技术的评价指标及其使用的频率

Table 1 Evaluation indicators of intelligent program synthesis technology and their frequency of use

评价层面	评价指标	相关文献	使用频率/(%)
合成结果	程序正确性	[2-55]	46.49
	程序规模	[3,8-12,15,16,22,23,31,33,34,39,42,46]	9.73
	程序相似度	[10,11,19,47,48,56]	5.41
合成过程	合成时间	[6,8,10-13,15,19-21,24,27,34,36,39,41-45,47,49,57]	13.51
	候选程序数量	[7,12-14,16,18,24,26,27,31,39-41,57]	9.19
训练合成器	训练数据量	[9,14-17,33,35,36,38,44]	6.49
其他	—	[11,15,22,25,44-46,48,50,51,55,57,58]	9.18

参考程序看作 tokens 序列, 通过计算两个序列的 token 或 tokens 组合的重叠精度来计算程序相似度. 例如, Papineni 等^[59]使用 BLEU 作为评价标准来判断合成器合成程序的好坏.

1.2 合成过程层面的评价指标

在智能程序合成技术的合成过程层面, 即根据合成程序的效率, 也就是合成目标程序的快慢对合成进行评价, 总结出了智能程序合成技术的两个评价指标: 合成时间和候选程序数量.

合成时间也是使用频率比较高的评价指标, 其是指从开始执行程序合成, 到最终生成可执行程序或目标代码所花费的时间. 例如, Shi 等^[12]计算从程序开始执行到合成期望程序所经过的实际时间来评估程序合成的效率. Kalyan 等^[13]使用找到正确程序所花费的墙钟时间作为评价标准对合成进行评价. Raghothaman 等^[57]则以合成器生成前 10 个解决方案需要的时间为标准评价合成器的效率.

候选程序数量也是影响合成时间的一个重要因素, 因此将其作为评价程序合成效率的一个评价指标. 该数量具体指在自动或半自动地合成满足规约的程序时, 合成器生成的程序变体的数量. 如果考虑的候选程序数量过多, 则会延长合成时间, 降低合成效率; 反之, 则会限制程序的搜索范围, 影响合成效果. 例如, Odena 等^[7]将考虑候选程序的数量作为评价标准, 通过修改候选程序的数量来提高程序合成的效率. Valkov 等在符号搜索中考虑了候选程序的数量, 可以更全面地探索搜索空间, 提高了找到最优解的可能性^[14].

1.3 训练合成器层面的评价指标

在智能程序合成技术的训练合成器层面, 即根据训练合成器使用的训练数据集的量级对合成进行评价, 总结出了训练数据量一个评价指标.

训练数据量通常是指训练程序合成器使用的训练数据集的大小, 通常希望在合成性能不变的前提下, 使用的训练数据集越小越好. 或者通过提高训练数据集的量级, 让合成器达到更好的性能. 例如, Handa 和 Rinard^[15]统计合成正确程序需要的输入/输出示例的最小数量, 并探讨了输入/输出示例的数量对程序合成性能的影响. Nye 等^[16]将训练程序的数量作为评价智能合成系统的一个标准, 通过增加训练程序的数量来提高合成器的学习能力和泛化能力. Chen 等^[17]则考虑训练数据的大小对系统性能的影响, 并且探讨了如何使用有限的训练数据来提高合成器的性能.

1.4 分析

从智能程序合成技术的合成程序的质量、合成程序的效率和训练使用的数据量三个维度出发, 即从智能程序合成的结果、过程以及训练合成器的角度出发, 从 55 篇智能程序合成技术的相关文献中提取出了验证合成有效性的评价标准, 通过对这些评价标准进行归纳总结, 总结出了 6 个智能程序合成技术的评价指标: 程序正确性、程序规模、程序相似度、合成时间、候选程序数量和训练数据量, 如图 4 所示.

2 智能程序合成技术的评价方法

通过谷歌学术搜索, ACM Digital Library, IEEE Xplore, Springer Link Online Library 等, 使用关键词 Intelligent Synthesis, Evaluation Methods 等对目前主流的智能程序合成的评价方法进行了检索, 并收集了相关文献. 通过调研发现, 当前主流的智能程序合成的评价方法主要是从程序合成的质量, 即从合成结果的角度出发, 对程序合成技术进行评价. 通过分析这些评价方法中使用的评价指标, 以及

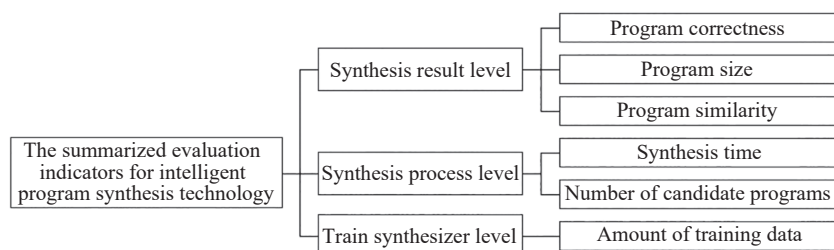


图 4 总结出的智能程序合成技术的评价指标

Fig. 4 Summarized evaluation indicators for intelligent program synthesis technology

根据第 1 节中提出的智能程序合成技术的评价指标, 可以将其分为程序正确性的评价方法、程序相似度的评价方法以及其他的评价方法.

2.1 程序正确性的评价方法

程序正确性的评价方法以程序正确性为评价指标对智能程序合成进行评价, 通过运行和测试合成程序来判断其是否满足预期的功能, 即根据合成程序通过单元测试的情况来判断程序的功能正确性. 例如, Kulal 等^[60]发现精确序列匹配无法解释合成程序的功能正确性, 精确序列匹配是将合成程序与参考程序看为 tokens 序列, 根据参考程序的序列寻找与其完全匹配的合成程序序列的过程. 为此, 提出了判断合成程序功能正确性的评价方法 $\text{pass}@k$. 其中 k 表示为每个问题生成程序采样的数量, 只要其中任何一个程序通过了单元测试, 则认为该问题被解决, 并报告解决问题所占的比重. Chen 等^[61]也认为应该将合成程序的正确性作为一个评价指标用于程序合成任务. 但是, 他们认为 Kulal 等^[60]用来计算功能正确性的方法具有较大的方差, 因为 k 个程序采样通过单元测试的数量的多少也会反映出合成程序的质量. 因此, 为每个问题生成 n ($n \geq k$) 个程序采样, 统计通过单元测试正确程序采样 c ($c \leq n$) 的数量, 并计算期望 E_{pro} 以更准确地计算功能正确性, 评价方法得分 $S_{\text{pass}@k}$ 的计算公式如下:

$$S_{\text{pass}@k} = E_{\text{pro}} \left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]. \quad (1)$$

Kulal 等^[60]和 Chen 等^[61]通过运行和测试合成程序来计算合成程序的正确性, 进而判断合成的程序是否满足预期的功能. 这种针对功能正确性的评价方法显然更加合理, 因为即使合成程序和参考程序的语义相同, 却也不一定可以实现相同的功能, 可能在实现细节、边界条件和错误处理等方面存在差异. 但是, 这种评价方法需要为每个问题设置完备的单元测试来判断合成程序的功能正确性, 而且需要通过足够多的单元测试才能区分不同的合成器. 此外, 即使合成器生成的程序没有实现预期的功能, 也不意味着合成程序的质量较差, 可能通过适当的修改也能实现与参考程序相同的功能. 因此, 这种评价功能正确性的评价方法在面对通过单元测试数量较少或者相近的情

况时, 具有一定的局限性.

2.2 程序相似度的评价方法

程序相似度的评价方法是将程序相似度作为评价指标, 对智能程序合成进行评价, 这种评价方法最简单的思路是将合成程序和参考程序视为 tokens 序列, 即字符串序列, 通过比较两个字符串序列的相似程度来判断合成程序的质量. 例如有很多研究使用机器翻译领域的评价指标 BLEU (Bilingual Evaluation Understudy)^[12]来判断合成程序的好坏. 其将合成程序和参考程序看作连续的 tokens 序列, 通过计算单个 token 或者连续 tokens 组合的重叠精度, 来判断合成程序的质量, 评价方法得分 S_{BLEU} 的计算公式如下:

$$S_{\text{BLEU}} = B_p \exp \left[\sum_{n=1}^N w_n \lg(P_n) \right]. \quad (2)$$

其中, B_p 为短惩罚因子, 用来惩罚长度比参考程序序列短的合成程序, P_n 和 w_n 分别是每个 token 组合 (n -grams) 的重叠精度和其相应的权重, n 的取值通常为 1~4. 也有使用 METEOR (Metric for Evaluation of Translation with Explicit Ordering)^[62]进行评价的方法. METEOR 同样是机器翻译领域的评价指标, 其目标是解决 BLEU 中的一些固有缺陷, 其在 BLEU 的基础上扩展了词形变体和同义词的匹配, 同时将词序纳入评估范畴, 并设置基于词序变化的罚分机制, 评价方法得分 S_{METEOR} 的计算公式如下:

$$S_{\text{METEOR}} = F_{\text{mean}} \cdot (1 - P). \quad (3)$$

其中, F_{mean} 为召回率和精度的调和平均值, P 为基于词序的惩罚因子. 使用 BLEU 和 METEOR 对合成程序的质量进行评价都是基于 token 级别的重叠精度计算, 还有通过计算字符级别的重叠精度来评价合成的质量. 例如, CHRF (Character N-gram F-score)^[63]就是使用字符级别的 n -grams 重叠精度来度量合成程序与参考程序的相似程度, 其也是机器翻译领域的评价指标, 能够更好地处理一些语法结构不同但是意思相近的翻译. 由于字符级的评价结果可能过于乐观, 因此, 在 CHRF 的基础上, 又提出了 CHRF++^[64], 即计算字符级和单词级重叠精度的组方法.

BLEU, METEOR, CHRF/CHRF++都是机器翻译领域的评价指标, 用来评价机器翻译的质量, 但是有很多研究直接将这些评价指标应用于程序合成领域, 他们将合成程序和参考程序看作 tokens 序列,

通过计算两者的相似程度, 来判断合成程序的质量. 这种评价方法足够简单, 并被广泛使用, 而且当合成的程序无法运行来计算合成程序的正确性时, 这种评价方法也可以在一定程度上对合成的质量进行判断. 然而, 这种评价方法只是将程序简单地看作字符串序列, 并没有考虑程序固有的语法和语义信息, 因此也具有一定的局限性.

为了解决上述评价方法的局限性, 有研究在评价合成程序与参考程序的相似程度过程中融入了代码固有的语法和语义信息. 例如, Tran 等^[65]提出了一种评价方法 RUBY, 其使用了三种抽象层次由低到高的程序相似程度的评价方法, 分别为字符串相似程度 (String Similarity, STS) 评价、树相似程度 (Tree Similarity, TRS) 评价和图相似程度 (Graph Similarity, GRS) 评价方法. 评价方法得分 S_{RUBY} 的计算公式如下:

$$S_{\text{RUBY}} = \begin{cases} S_{\text{GRS(R,T)}}, & \text{if PDGs are applicable;} \\ S_{\text{TRS(R,T)}}, & \text{if ASTs are applicable;} \\ S_{\text{STS(R,T)}}, & \text{otherwise.} \end{cases} \quad (4)$$

其中, R和T分别表示参考程序和合成程序, $S_{\text{STS(R,T)}}$, $S_{\text{TRS(R,T)}}$ 和 $S_{\text{GRS(R,T)}}$ 分别为字符串相似度得分, 树相似度得分和图相似度得分. 使用评价方法的抽象层次越高, 越能实现与语义得分更高的相关系数. Tran 等^[65]通过比较合成程序和参考程序的抽象语法树 (Abstract Syntax Tree, AST) 和程序依赖图 (Program Dependency Graph, PDG) 的最小编辑距离来计算两者的相似程度得分, 从而引入了程序的语法和语义信息. 抽象语法树是编程语言中用于表示代码结构的一种树形数据结构, 程序依赖图是软件工程中用于表示程序中各种依赖关系的图形表示. Ren 等^[66]同样提出了一种评价方法 CodeBLEU, 其在 BLEU 的基础上通过 AST 和数据流图 (Data Flow Graph, DFG) 融入了代码的语法和语义信息. 数据流图是用于描述计算机程序中各个计算操作之间的数据依赖关系的图形表示方法. 评价方法得分 S_{CodeBLEU} 的计算公式如下:

$$S_{\text{CodeBLEU}} = \alpha \cdot S_{\text{BLEU}} + \beta \cdot S_{\text{BLEU}_w} + \gamma \cdot S_{\text{ast}} + \delta \cdot S_{\text{df}}. \quad (5)$$

其中, α , β , γ , δ 为权重, S_{BLEU_w} 是加权的 BLEU 得分, 能够为程序中关键字的匹配赋予更高的权重,

S_{ast} 和 S_{df} 分别是合成程序和参考程序的抽象语法树子树的匹配和数据流的匹配得分.

RUBY 和 CodeBLEU 在上述使用机器翻译领域评价指标进行评价的方法的基础上, 进一步融入了对程序语法和语义的评价, 能够更准确地计算合成程序与参考程序的相似程度. 但是, 这些评价方法都是静态的方法, 即不需要运行合成的程序就能够对其进行评价. 因此, 这些评价方法无法验证合成的程序是否满足预期的功能, 即程序正确性, 所以具有一定的局限性. 此外, 如果合成的程序不够正确而导致无法构建抽象语法树或者程序依赖图时, RUBY 只能通过字符串的最小编辑距离来判断合成程序的质量, 因此具有一定的不稳定性; CodeBLEU 是通过计算四个组成部分的加权和来对合成的质量进行评价, 但如何合理分配每一个部分的权重仍然是一个重要问题.

2.3 其他评价方法

除了使用程序正确性和程序相似度的评价方法外, 还调研到了一些其他的智能程序合成的评价方法, 通过分析这些评价方法中使用的评价指标, 总结出了一些新的智能程序合成的评价指标: 程序执行效率、程序安全性、程序有效性、程序复杂性、程序可靠性和程序可读性, 将这些评价指标加入第 1 节中总结的评价指标, 以完善智能程序合成技术的评价指标.

程序执行效率是指程序运行时所消耗的时间和资源, 通常用来衡量程序的性能和效率. 例如, Pan 和 Lyu^[67]使用程序执行效率对合成进行评价, 发现了大语言模型经常生成执行效率较低的程序而难以满足实时性要求的问题, 因此他们引入了高效程序生成基准, 用来评估选择高效程序解决问题的能力. 为此, 他们使用相对效率 (Relative Efficiency, RE) 来衡量合成程序的执行效率. 相对效率是通过比较合成程序的运行时间与参考程序的最小和最大的运行时间来计算的. 由于计算相对效率需要执行程序, 因此当合成程序中有标记错误时会导致不适用. 对此, 专门训练了一个用于预测程序运行时间的模型. 然而, 构建这样的模型本身就面临很多困难, 而且程序的运行时间会受到许多因素的影响, 如何准确地获得程序的运行时间也是一个需要解决的问题.

程序安全性是指程序在设计、开发、部署和运行过程中, 能够有效地防止各种恶意攻击、数据泄露、服务拒绝等安全威胁, 保护系统和用户的数据、资产

和隐私不受损害的能力. 例如, Improtota 使用程序安全性对程序合成进行评价, 他提出了一个新的数据中毒攻击策略, 通过在训练数据中注入安全漏洞来影响人工智能程序合成器, 分析合成的程序中是否包含预期的漏洞或安全缺陷来判断攻击是否成功^[68]. 为了量化攻击的效果, Improtota 使用攻击成功率进行衡量, 即测试数据中合成有漏洞的程序占总程序的比例. 但是, 这种评价方法仅限于已知的安全漏洞, 而无法捕获未知的或者新型的漏洞. Siddiq 和 Santos 同样使用程序安全性对合成器的性能进行评价, 他们从安全编码的角度创建了一个评价数据集来衡量机器学习模型合成程序的质量^[69]. 通过将数据集中的提示提供给合成器, 然后检查合成程序来评价程序合成技术的安全性. 评价过程可以手动进行, 例如比较合成程序与数据集中不安全程序代码样本的相似程度; 也可以自动进行, 例如通过使用现有的静态分析器来自动查找合成程序中的漏洞. 同样地, 该评价数据集也仅限于已知的漏洞.

此外, 还有一些综合的智能程序合成的评价方法. 例如, Su 等^[70]提出了一个包括程序有效性、程序正确性、程序复杂性、程序可靠性、程序安全性和程序可读性在内的 6 个评价指标的评价指标体系, 并通过计算其权重和来评价合成的好坏. 其中程序有效性是指合成的程序能否正确运行; 程序正确性衡量程序满足需求的程度; 程序复杂性主要考虑程序是否过于复杂; 程序可靠性和安全性评估程序中的错误和漏洞的数量以及解决的难度; 程序可读性衡量用户理解程

序的难度. 但是, 程序的复杂性和可读性涉及一定的主观判断, 这可能会导致评分的差异从而影响评价结果. Kovalchuk 等^[71]在实际问题回答场景中比较了基于测试和基于度量的评价方法. 其中基于测试的评价方法自动生成测试用例, 并且应用于合成器的生成程序, 计算通过测试的数量来评价合成程序的质量; 基于度量的评价方法使用常见的自然语言处理指标, 例如采用 BLEU 等来评价合成程序与参考程序的相似程度. 他们通过研究发现, 仅依赖于基于测试或基于度量的评价方法对合成程序质量的评价相当有限, 具有较高评分的合成程序可能无法通过测试用例, 而有些评价较低合成程序却能通过测试用例. 因此, 他们认为有效地结合这两种评价方法能够提高评价的准确性.

2.4 分析

分析综上智能程序合成的主流评价方法, 根据这些评价方法中使用的评价指标以及如何使用这些评价指标对合成技术进行评价, 提出了一些新的评价指标并完善了智能程序合成技术的评价指标: 合成结果层面, 在程序正确性、程序规模和程序相似度的基础上, 按照上述调研的顺序, 增加了程序执行效率、程序安全性、程序有效性、程序复杂性、程序可靠性和程序可读性 6 个评价指标; 在合成过程层面, 有合成时间和候选程序数量两个评价指标; 在训练合成器层面, 只有训练数据量一个评价指标. 这 12 个评价指标将为后续构建航天领域中智能合成技术的评价指标体系提供支持, 如图 5 所示.

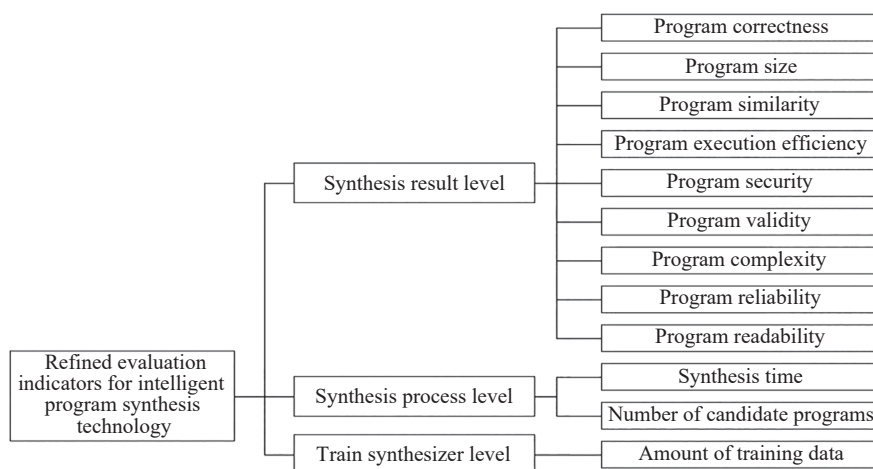


图 5 完善后的智能程序合成技术的评价指标

Fig. 5 Refined evaluation indicators for intelligent program synthesis technology

在调研主流智能程序合成技术的评价方法时发现, 绝大多数评价方法都是从合成结果的角度出发, 对程序合成技术进行评价. 此外, 合成结果是智能程序合成的最终产物, 其直接关系到合成器的性能表现. 因此, 合成结果层面应该占据较大权重. 合成过程层面考虑的是合成时间和候选程序数量, 良好的合成过程能够提高合成效率和合成结果的质量, 即在用户可接受的时间内返回最优目标结果. 因此, 合成过程层面应该占据次高的权重. 训练合成器层面主要关注训练数据量, 其虽然影响合成过程和合成结果, 但相对来说直接影响较小, 因此分配较小的权重.

另外, 通过调研发现, 在合成结果层面中使用程序正确性和程序相似度这两个评价指标的评价方法较多. 而且, 如果仅使用程序正确性作为评价指标, 在通过单元测试数量较低或者相近的情况下, 可能无法准确地判断合成程序的质量; 如果仅使用程序相似度作为评价指标, 即使在考虑程序语法和语义的情况下, 合成程序可能也无法满足预期的要求. 而衡量程序正确性需要运行程序以检查功能正确性, 即动态地评价智能程序合成技术; 衡量程序相似度只需要静态分析合成程序与参考程序的相似程度, 即静态地评价智能程序合成技术. 因此, 一种更有效的智能程序合成技术评价方法的设计思路是在合成结果层面有效地结合程序正确性和程序相似度这两个评价指标, 以动态和静态相结合的方式实现对智能程序合成技术的评价. 当设计面向航天控制软件智能合成技术的评价方法时, 对于合成过程层面和训练合成器层面, 可以维持原有的评价指标不变. 对于合成结果层面, 除了程序正确性和程序相似度两个评价指标外, 可以根据航天嵌入式软件的特点, 选择或设计相应的评价指

标, 如图 6 所示.

3 面向航天控制软件智能合成技术的评价方法

为了更有效地评价智能程序合成技术, 从完善后的智能程序合成技术的评价指标中选择合适的指标. 在此基础上, 分析并结合航天嵌入式软件的特点, 选择或设计相应的面向航天控制软件智能合成技术的评价指标. 从合成结果、合成过程和训练合成器的层面, 即合成程序的质量、合成程序的效率和训练使用的数据量三个层面出发, 根据以上评价指标, 构建面向航天控制软件智能合成技术的评价指标体系. 参考当前主流智能程序合成技术的评价方法, 设计面向航天控制软件智能合成技术的评价方法.

3.1 选择与设计面向航天控制软件智能合成技术的评价指标

对于合成过程和训练合成器层面, 维持原有的评价指标不变, 即合成过程层面选择合成时间和候选程序数量作为评价指标, 训练合成器层面选择训练数据量作为评价指标. 对于合成结果层面, 根据第 2 节中提出的更有效的智能程序合成技术的评价方法, 选择程序正确性和程序相似度两个评价指标, 通过结合动态和静态的评价方法, 实现对智能程序合成技术更有效的评价.

在此基础上, 分析并结合航天嵌入式软件的特点, 选择或设计面向航天控制软件智能合成技术的评价指标. 航天嵌入式软件是应用在航天器上的软件系统, 具有高可靠性和安全性、严格的实时性、硬件平台受限以及多样化的任务需求. 对于高可靠性和安全

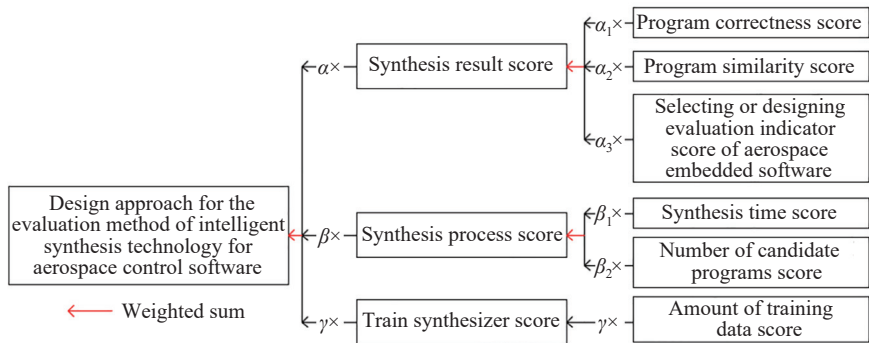


图 6 面向航天控制软件智能合成技术评价方法的设计思路

Fig. 6 Design approach for the evaluation method of intelligent synthesis technology for aerospace control software

性, 航天嵌入式软件通常用于控制和操作航天器, 因此对其可靠性和安全性要求极高. 即使在极端环境下或面临异常情况时, 软件也必须能够稳定运行, 确保航天器的安全和任务的完成; 对于严格的实时性, 航天嵌入式软件往往需要在严格的实时约束下运行, 确保航天器能够及时响应各种指令和环境变化, 保证任务的顺利执行; 对于硬件平台受限, 航天嵌入式软件运行在资源受限的硬件平台上, 通常具有较小的存储容量和有限的计算能力, 因此需要精心设计和优化, 尽量减少资源消耗和系统开销; 对于多样化的任务需求, 航天嵌入式软件需要满足多样化的任务需求, 包括导航控制、数据采集、传感器控制、通信管理等功能, 因此软件架构和设计必须灵活且可扩展, 以适应不同任务和应用场景. 综上所述, 开发和评价航天嵌入式软件需要充分考虑这些特点, 并采取相应的设计和评估策略, 以确保软件系统能够稳定可靠地运行在航天器上.

如图 7 所示, 对于航天嵌入式软件的高可靠性和安全性, 选择程序可靠性和程序安全性两个评价指标, 分别对合成程序的可靠性和安全性进行评价; 对于航天嵌入式软件的严格的实时性, 设计响应时间作为评价指标, 衡量合成程序能否在规定的时间内响应外部事件. 选择任务完成度作为评价指标, 衡量合成程序能否在规定的时间内产生预期的输出或者正确完成相关的操作; 对于航天嵌入式软件的硬件平台受限, 选择程序执行效率和程序规模作为评价指标, 衡量合成程序在有限资源下能否高效地运行以及占用存储空间的情况; 对于航天嵌入式软件的多样化的任务需求, 设计可维护性和可扩展性作为评价指标, 衡量合成程序是否易于理解和修改以便灵活地添加新

的功能需求.

3.2 构建面向航天控制软件智能合成技术的评价指标体系

根据选择和设计的智能程序合成技术的评价指标, 从合成结果、合成过程和训练合成器三个层面出发, 构建了面向航天控制软件智能合成技术的评价指标体系. 在合成结果层面, 除程序正确性和程序相似度之外, 这里结合航天嵌入式软件的特点, 选择和设计了程序可靠性、程序安全性、响应时间、任务完成度、程序执行效率、程序规模、程序可维护性和程序可扩展性, 共计十个评价指标; 在合成过程层面, 包括合成时间和候选程序数量两个评价指标; 在训练合成器层面, 使用训练数据量作为评价指标. 面向航天控制软件智能合成技术的分层式评价指标体系如图 8 所示.

3.3 设计面向航天控制软件智能合成技术的评价方法

根据第 2 节中评价方法的设计思路, 首先选择合成结果层面的程序正确性和程序相似度两个评价指标, 设计了一种动态和静态相结合的智能程序合成技术的评价方法, 来更有效地对合成技术进行评价. 当设计面向航天控制软件智能合成技术的评价方法时, 维持合成过程层面和训练合成器层面原有的评价指标不变, 即合成过程层面使用合成时间和候选程序数量作为评价指标, 训练合成器层面使用训练数据量作为评价指标.

在此基础上, 结合航天嵌入式软件的特点, 根据构建的航天控制软件智能合成技术的评价指标体系, 设计了一种面向航天控制软件智能合成技术的综合评价方法.

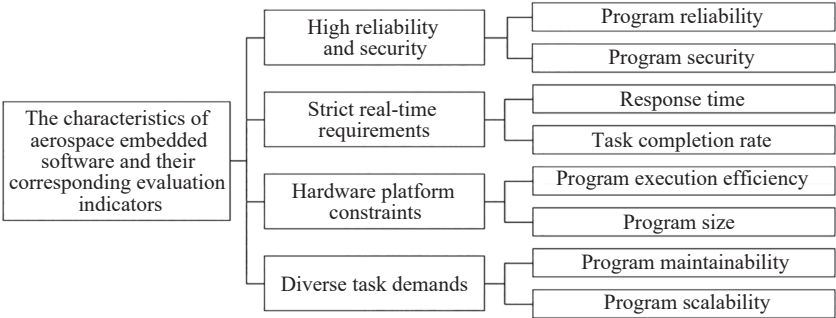


图 7 航天嵌入式软件的特点及其相应的评价指标

Fig. 7 Characteristics of aerospace embedded software and their corresponding evaluation indicators

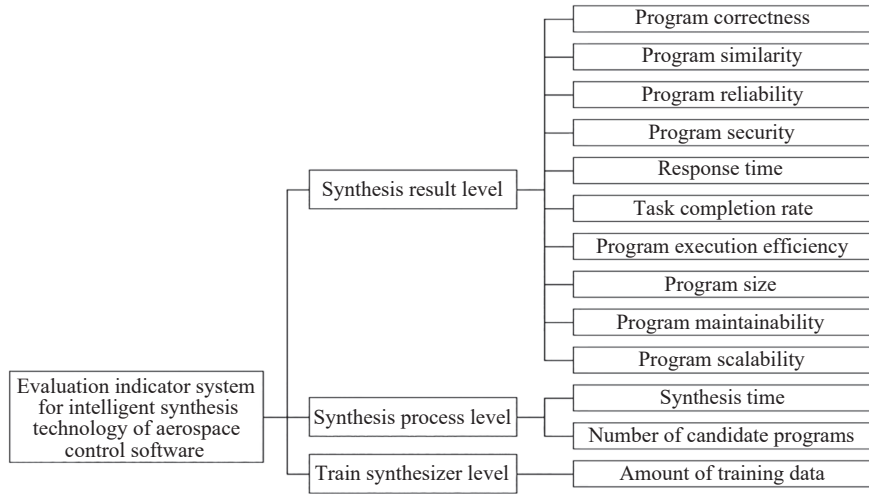


Fig. 8 Evaluation indicator system for intelligent synthesis technology of aerospace control software

3.3.1 动态与静态相结合的智能程序合成技术的评价方法

通过调研当前主流的智能程序合成技术的评价方法发现,使用程序正确性和程序相似度的评价方法很多,但是都没有将两者进行有效的结合.程序正确性主要是衡量合成程序的功能正确性,其需要动态地运行程序,通过观察合成程序能否通过单元测试来判断其是否满足用户的预期要求.虽然考虑程序正确性更加合理,但是当合成程序无法通过足够多的单元测试或者合成程序通过单元测试数量相近时,则具有一定的局限性.程序相似度主要是将合成程序与参考程序看为 tokens 序列,静态分析两个序列之间的匹配程度.当程序正确性无法区分合成器的性能时,程序相似度可以比较合成程序与参考程序的抽象语法树匹配和数据流图匹配程度,即语法和语义的匹配程度,来进一步区分合成器的性能.即使当合成程序质量较差而导致无法构建抽象语法树和数据流图时,程序相似度还可以通过简单地比较 tokens 的匹配程度来判断合成器的性能.因此,本文设计了一种动静结合的智能程序合成技术的评价方法,利用程序正确性和程序相似度两个评价指标,以更有效地评价智能程序合成技术.以下将对评价方法使用的技术手段以及权重的分配进行详细介绍.

程序正确性评估合成程序是否符合预期的功能,程序相似度评估合成程序与参考程序的相似程度.对于程序正确性,利用文献 [60,61] 中的方法,通过计算合成的前 1, 10 或 100 个程序通过单元测试的概率来

对合成进行评价,即 pass@1, pass@10, pass@100; 对于程序相似性,利用文献 [64,66] 中的方法,将合成程序与参考程序看作 tokens 序列,通过计算两者字符级和单词级、语法树子树以及数据流的重叠精度来对合成进行评价,即使用 CHRF++ 和 CodeBLEU 中的抽象语法树和数据流图匹配.

如果仅从合成结果的角度且不考虑航天嵌入式软件特点的情况下对智能程序合成技术进行评价,可以给程序正确性和程序相似度各分配 0.5 的权重系数,因为在合成程序通过单元测试数量较低或者相近的情况下,通过比较程序相似度可以有效地解决这种局限性.在计算程序正确性时,由于希望智能程序合成技术更加有效,所以主要考虑合成前 1 和前 10 个程序的正确性,并分别赋予 0.9 和 0.1 的权重系数;在计算程序相似度时,通过融入程序的语法和语义信息来更有效地评价合成器性能,所以为抽象语法树和数据流图匹配赋予各 0.2 的权重系数.当合成程序质量较差而导致无法构建抽象语法树时,可以使用字符级和单词级相结合的计算程序相似度的方法,为其赋予 0.1 的权重系数.这种动态与静态相结合的智能程序合成技术评价方法得分计算公式如下:

$$S_{\text{Method}} = \alpha_1 S_{\text{corr}} + \alpha_2 S_{\text{simi}}. \quad (6)$$

其中, α_1 和 α_2 为权重, S_{corr} 为程序正确性得分, S_{simi} 为程序相似度得分.

利用不同的技术手段和相应的权重分配对式 (6) 展开后得到

$$S_{\text{Method}} = 0.5 \times 0.9 \times S_{\text{pass@1}} + 0.5 \times 0.1 \times S_{\text{pass@10}} + 0.2 \times S_{\text{ast}} + 0.2 \times S_{\text{df}} + 0.1 \times S_{\text{chrf}}. \quad (7)$$

其中, α_1 和 α_2 为权重, S_{corr} 为程序正确性得分, 通过合成前 1 个程序正确性得分 $S_{\text{pass@1}}$ 和合成前 10 个程序正确性得分 $S_{\text{pass@10}}$ 进行展开, 分别赋予权重为 0.45 和 0.05, S_{simi} 通过抽象语法树相似度得分 S_{ast} 、数据流图相似度得分 S_{df} 和字符串相似度得分 S_{chrf} 进行展开, 分别赋予权重为 0.2, 0.2 和 0.1.

3.3.2 面向航天控制软件智能合成技术的综合评价方法

当设计面向航天控制软件智能合成技术的评价方法时, 合成过程层面依然选择合成时间和候选程序数量作为评价指标, 训练合成器层面选择训练数据量作为评价指标. 对于合成结果层面, 在动静结合的智能程序合成技术评价方法的基础上, 结合航天嵌入式软件高可靠性和安全性、严格的实时性、硬件平台受限以及多样化的任务需求的特点, 额外选择和设计了程序可靠性、程序安全性、响应时间、任务完成度、程序执行效率、程序规模、程序可维护性和程序可扩展性作为评价指标. 以下将对设计的面向航天控制软件智能合成技术的综合评价方法以及合成结果、合成过程与训练合成器三个层面的评价指标的计算方式进行详细介绍.

从整体的层面出发, 设计的面向航天控制软件智能合成技术的综合评价方法通过计算合成结果、合成过程和训练合成器三个层面的加权和来得到智能程序合成技术的最终得分 S_{Method} , 有

$$S_{\text{Method}} = \alpha S_{\text{res}} + \beta S_{\text{pro}} + \gamma S_{\text{tra}}. \quad (8)$$

其中, α , β , γ 为权重系数, S_{res} 为合成结果得分, S_{pro} 为合成过程得分, S_{tra} 为训练合成器得分. 可以根据第 2 节小结中的分析, 分别给予合成结果层面、合成过程层面以及训练合成器层面较高、次高、较低的权重.

从合成结果、合成过程和训练合成器三个层面对式 (8) 进行展开. 其中合成结果层面除了程序正确性和程序相似度外, 还包括程序可靠性、程序安全性、响应时间、任务完成度、程序执行效率、程序规模、程序可维护性和程序可扩展性 8 个评价指标. 通过计算 10 个评价指标的加权和可以得到合成结果的得分 S_{res} , 即

$$S_{\text{res}} = \alpha_1 S_{\text{corr}} + \alpha_2 S_{\text{simi}} + \alpha_3 S_{\text{reli}} + \alpha_4 S_{\text{secu}} + \alpha_5 S_{\text{resp}_t} + \alpha_6 S_{\text{comp}} + \alpha_7 S_{\text{eff}} + \alpha_8 S_{\text{size}} + \alpha_9 S_{\text{main}} + \alpha_{10} S_{\text{scal}}. \quad (9)$$

其中, $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8, \alpha_9, \alpha_{10}$ 为式 (8) 中 α 的子权重系数, S_{corr} 为程序正确性得分, S_{simi} 为程序相似度得分, S_{reli} 为程序可靠性得分, S_{secu} 为程序安全性得分, S_{resp_t} 为响应时间得分, S_{comp} 为任务完成度得分, S_{eff} 为程序执行效率得分, S_{size} 为程序规模得分, S_{main} 为程序可维护性得分, S_{scal} 为程序可扩展性得分. 以下将对除了程序正确性和程序相似度外的评价指标的计算方式进行详细介绍. 对于程序可靠性, 可以利用文献 [60] 中的方法, 评估合成程序中缺陷的数量以及改正这些缺陷需要的努力, 可以为缺陷更少的合成技术给予更高的人类评分; 对于程序安全性, 可以利用文献 [68,69] 中的方法, 修改部分训练数据以注入安全漏洞, 通过分析合成程序中是否包含预期的漏洞来进行评价, 可以为包含漏洞数量较少的合成技术给予更高的人类评分; 对于响应时间, 可以测量实际运行中合成程序在收到外部事件或指令后的响应时间, 可以为响应时间更快的合成技术给予更高的人类评分; 对于任务完成度, 衡量合成程序能否在规定的时间内产生预期的输出或执行正确的操作, 可为在规定时间内任务完成度较高的合成技术给予较高的人类评分; 对于程序执行效率, 可利用文献 [67] 中的方法, 通过统计合成程序的运行时间或者使用模型预测合成程序的运行时间来计算相对效率, 过滤掉效率较低的候选程序, 对执行效率更高的合成技术赋予更高的人类评分; 对于程序规模, 评估合成程序在实现相同功能的前提下, 能否通过优化使得合成程序的规模更小, 从而满足硬件平台受限的要求, 并赋予其更高的人类评分; 对于程序可维护性, 可利用文献 [70] 中的方法, 关注合成程序代码清晰度、注释质量、命名规范性等, 对代码结构清晰、注释准确、命名规范良好的合成程序给予更高的人类评分; 对于程序可扩展性, 可以评估合成程序是否有模块化设计或者标准化接口来实现可扩展性, 对达到这些要求的合成程序给予更高的人类评分.

合成过程层面包括合成时间和候选程序数量两个评价指标, 通过计算两个评价指标的加权和来得到合成过程得分 S_{pro} , 即

$$S_{\text{pro}} = \beta_1 S_{\text{synt}_t} + \beta_2 S_{\text{cand}} \quad (10)$$

其中, β_1 和 β_2 为式 (8) 中 β 的子权重系数, S_{synt_t} 为合成时间得分, S_{cand} 为候选程序数量得分. 合成时间主要评估合成器能否在用户可接受的时间内合成目标程序. 通常情况下, 对于能够在较短时间内找到目标解决方案的智能程序合成技术, 给予较高的人类评分; 候选程序数量主要评估在考虑候选程序数量相同的情况下, 合成器能否找到更优的解决方案. 或者反之, 合成器找到目标程序需要考虑的候选程序数量的多少. 通常情况下, 为找到目标程序且考虑较少候选程序数量的合成技术给予较高的人类评分.

训练合成器层面仅包含一个评价指标, 即使用训练数据量对智能程序合成技术进行评价. 因此, 训练合成器得分是根据训练数据量计算得到的分数 S_{tra} , 即

$$S_{\text{tra}} = S_{\text{data}} \quad (11)$$

其中, S_{data} 为训练数据量得分, 训练数据量主要评估在实现相近合成器性能的前提下, 使用的训练数据集的大小. 通常情况下, 对于实现相近性能却使用较小训练数据量的智能程序合成技术, 给予较高的人类评分.

3.4 分析

为了更有效地评价智能程序合成技术, 并且根据航天嵌入式软件的特点, 选择和设计了面向航天控制软件智能合成技术的评价指标. 在确定评价指标后, 从合成结果、合成过程和训练合成器三个角度构建了面向航天控制软件智能合成技术的层级式评价指标体系. 在设计评价方法时, 先选择了合成结果层面的程序正确性和程序相似度两个评价指标, 设计了一种动态和静态相结合的更有效的智能程序合成技术的评价方法, 如图 9 所示. 在此基础上, 维持合成过程层面和训练合成器层面原有的评价指标不变, 在合成结果层面融入了对航天嵌入式软件特点的评价, 设计了一种面向航天控制软件智能合成技术的综合评价方法, 如图 10 所示.

4 智能程序合成技术评价方法的有效性验证

由于缺少公开的航天代码数据集以及开源的航天控制软件智能合成技术工具, 因此, 在不考虑人类打分主观因素的影响下, 选择公开的代码数据集和开源的智能程序合成工具, 对第 3 节中提出的面向航天控制软件智能合成技术的综合评价方法进行可行性验证, 即对第 3 节中提出的智能程序合成技术的动静结合的评价方法进行有效性验证. 选择公开代码数据集 HumanEval, 并在 AutoDL 平台上部署开源的智能程序合成模型, 即大语言模型 ChatGLM-6B, ChatGLM2-6B, ChatGLM3-6B, 通过 prompt 的方式, 运行数据集得到智能合成程序实验数据. 使用公认能力较强的 ChatGPT3.5, 通过调用 ChatGPT3.5 的 API 接口, 来模拟人类对合成程序进行评分. 计算评价方法的最终得分与模拟人类评分的皮尔逊相关系数, 与仅动态和仅静态的评价方法的皮尔逊相关系数进行比较, 验证提出的智能程序合成技术评价方法的有效性.

4.1 实验前期准备及实验过程

在 AutoDL 平台上租赁了三台配备 RTX 4090 GPU (24 GB 显存) 的服务器作为计算资源, 分别部署了大语言模型 ChatGLM-6B, ChatGLM2-6B, ChatGLM3-6B. 此外, 还将开源的 HumanEval 数据集部署到了本地环境中. HumanEval 数据集一共包含 164 个任务, 针对每个任务, 让模型生成 200 个程序样本. 将其作为运行的结果保存在 JSONL 文件中, 每个文件包含 164×200 条数据. 每个程序样本有 task_id 和 completion 两个字段, 分别保存任务序列号和模型生成的程序.

通过上述部署和调用大语言模型来运行 HumanEval 数据集, 最终得到了三个 JSONL 文件, 以供后续计算智能程序合成技术评价方法的得分. 然

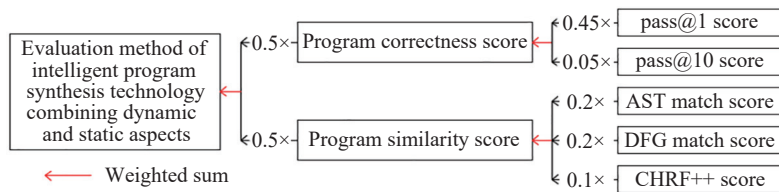


图 9 动态和静态相结合的智能程序合成技术的评价方法

Fig. 9 Evaluation method of intelligent program synthesis technology combining dynamic and static aspects

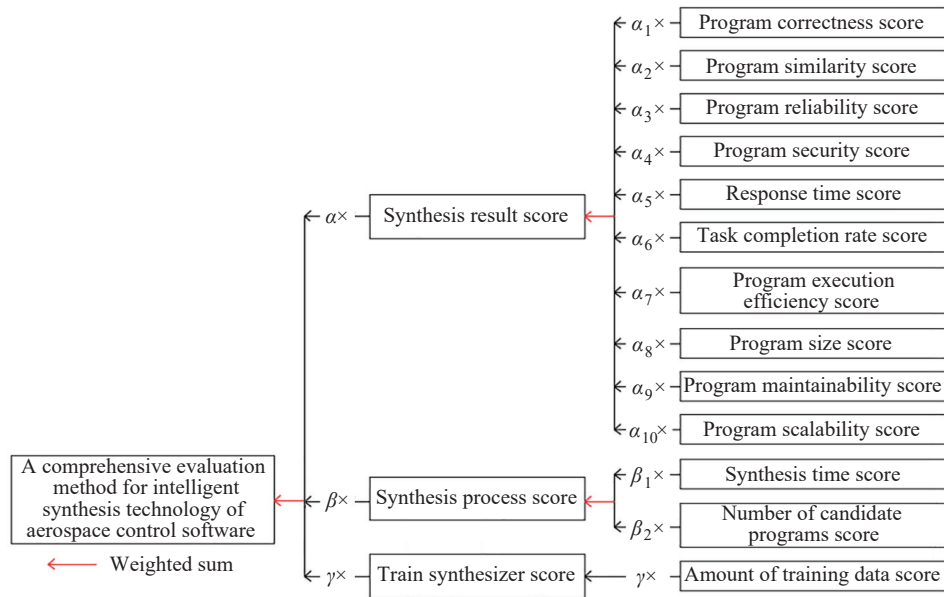


图 10 面向航天控制软件智能合成技术的综合评价方法

Fig. 10 A comprehensive evaluation method for intelligent synthesis technology of aerospace control software

而，在处理 ChatGLM-6B 和 ChatGLM2-6B 的 JSONL 文件时发现了两个问题：(1) completion 字段并非标准的 Python 程序，而是由代码解释、代码块组成的字符串序列，这种格式不便于后续执行程序来进行评价；(2) 由于 ChatGLM-6B 和 ChatGLM2-6B 的模型能力有限，部分生成的 completion 字段无限长，显然不满足要求。

针对问题 (1)，为了便于程序执行以评价程序的正确性，编写自动化脚本提取代码块中的程序，并删除代码解释等冗余信息；针对问题 (2)，无限长的合成程序明显不符合要求，所以编写脚本自动化地识别合成程序的长度，如果合成程序长度过长，则置 completion 字段为空，即不计算该程序采样的得分。此外，通过对 HumanEval 数据集进行分析发现，其提供的标准参考程序仅包括函数体，而缺少函数定义和所使用的 Python 包。为了便于后续对合成程序的评价，手动补全 164 个任务的标准参考程序。以上对合成结果和数据集的数据处理不仅便于计算评价方法的得分，而且不影响对合成器性能的评价。

4.2 实验结果和评价方法的有效性验证

根据第 3 节中的内容，依次计算 CHRF++、抽象语法树匹配、数据流图匹配、pass@1、pass@10、pass@100、CodeBLEU 以及设计的动态和静态相结合的智能程序合成技术评价方法的得分，最终的实验

结果如表 2 所示。

根据上述实验结果可以发现，对于 ChatGLM-6B, ChatGLM2-6B, ChatGLM3-6B 来说，包括提出的动静结合评价方法在内的所有方法得分都表现出了一致的上升的趋势。尤其是对于动态的评价方法，ChatGLM3-6B 的得分显著上升，说明其能够合成更符合预期功能的程序；而静态评价方法得分的上升趋势较为缓慢，ChatGLM3-6B 的得分有小幅度的提高。为了验证提出的智能程序合成技术评价方法比仅静态和仅动态的方法更加有效，调用公认能力较强的 ChatGPT3.5 的 API 接口，让其作为编程专家，对照参考程序，对最终合成程序结果进行打分，打分范围为 1 (非常不好) 至 5 (非常好)，模拟人类评分的结果如表 3 所示。

通过计算每种评价方法与模拟人类评分的皮尔逊相关系数，计算结果如表 4 所示。实验结果显示，对于 ChatGLM-6B, ChatGLM2-6B, ChatGLM3-6B 来说，设计的动静结合的智能程序合成技术的评价方法都能够一定程度上比仅动态和仅静态的评价方法获得更高的相关系数，即更贴近于模拟人类的评分。此外，发现随着模型性能提升，计算的皮尔逊相关系数反而有一定程度的下降，并且动态比静态评价方法有更快的下降趋势。这可能是由于评价方法考虑的方面与模拟人类评分偏好不一致导致的，ChatGPT3.5

表 2 实验结果
Table 2 Experimental results

评价方法	ChatGLM-6B	ChatGLM2-6B	ChatGLM3-6B
CHRF++	0.406261948	0.435981525	0.50124219
AST_MATCH	0.312514349	0.355593869	0.44914179
DFG_MATCH	0.517394044	0.548241746	0.60469280
pass@k (k=1)	0.040579268	0.093689024	0.62344512
pass@k (k=10)	0.135562582	0.192008573	0.79927887
pass@k (k=100)	0.257184867	0.307372033	0.83987480
CodeBLEU	0.259490443	0.289887388	0.35368544
本文评价方法	0.231646673	0.276125765	0.58140539

表 3 ChatGPT3.5 模拟人类评分结果
Table 3 Human rating results simulating by ChatGPT3.5

模型	ChatGLM-6B	ChatGLM2-6B	ChatGLM3-6B
模拟人类评分	2.361817523	2.758730102	4.199085366

表 4 皮尔逊相关系数的计算结果
Table 4 Calculations of the Pearson correlation coefficient

评价方法	ChatGLM-6B	ChatGLM2-6B	ChatGLM3-6B
CHRF++	0.340366625	0.460577583	0.382847599
AST_MATCH	0.370650915	0.379543662	0.342105016
DFG_MATCH	0.379022475	0.339336446	0.155618439
pass@k (k=1)	0.602161597	0.576083246	0.380828102
pass@k (k=10)	0.578192141	0.505663582	0.228842017
pass@k (k=100)	0.457669783	0.442143948	0.151757714
CodeBLEU	0.418648037	0.453922700	0.358580074
本文评价方法	0.614578225	0.594448137	0.425688413

模拟人类对合成程序进行评分相较于程序的功能正确性,可能更关注于合成程序与参考程序的相似程度。

5 结论与发展趋势

5.1 结论

传统的软件开发模式已经无法满足当下及未来嵌入式软件的开发需求。程序合成技术能够自动生成满足用户需求的程序代码,并且结合了人工智能技术的智能程序合成逐渐成为软件开发的新范式,能够有效地解决嵌入式软件开发中所面临的问题。虽然现在有一些智能程序合成技术的评价方法,但是面临着许

多问题,需要进一步完善和改进。本文针对面向航天控制软件智能合成技术的评价方法展开研究,主要解决了以下问题。

(1) 提出并完善了智能程序合成技术的评价指标。通过调研智能程序合成技术的相关文献,分析为了验证合成有效性使用的评价标准,从这些评价标准中总结出了智能程序合成技术的评价指标。然后,通过调研当前智能程序合成技术的主流评价方法,分析评价方法中使用的评价指标体系或评价指标,以此完善了智能程序合成技术的评价指标。完善后的智能程序合成技术的评价指标包括:程序正确性、程序规模、程序相似度、程序执行效率、程序安全性、程序有效性、

程序复杂性、程序可靠性、程序可读性、合成时间、候选程序数量、训练数据量共计十二个评价指标。

(2) 构建了面向航天控制软件智能合成技术的层级式评价指标体系。在完善后的智能程序合成技术的评价指标的基础上, 结合了航天嵌入式软件特点, 选择合适的评价指标或设计航天领域特有的评价指标来构建评价指标体系。从合成结果、合成过程和训练合成器三个层面构建了面向航天控制软件智能合成技术的分层评价指标体系。其中, 合成结果层面包括程序正确性、程序相似度、程序可靠性、程序安全性、响应时间、任务完成度、程序执行效率、程序规模、程序可维护性和程序可扩展性十个评价指标, 合成过程层面包括合成时间和候选程序数量两个评价指标, 训练合成器层面包括训练数据量一个评价指标。

(3) 设计了面向航天控制软件智能合成技术的综合评价方法。通过调研当前智能程序合成技术的主流评价方法, 学习评价方法中使用的技术手段, 利用程序正确性和程序相似度两个评价指标, 设计了一种动态和静态评价相结合的智能程序合成技术的评价方法, 可以更有效地评价合成技术。在此基础上, 结合了航天嵌入式软件的特点, 在合成结果层面融入了对航天控制软件的评价, 设计了一种面向航天控制软件智能合成技术的综合评价方法。通过迭代模型运行公开的代码数据集, 并模拟人类对合成程序进行打分发现, 提出的动静结合的评价方法与其他评价方法相比, 与模拟人类评分的皮尔逊相关系数更高, 进而证明了智能程序合成技术评价方法的有效性以及面向航天控制软件智能合成技术综合评价方法的可行性。

5.2 需要进一步开展的工作

本文只是验证了提出的智能程序合成技术评价方法的有效性, 而没有对面向航天控制软件智能合成技术的综合评价方法进行有效性验证。虽然智能程序合成技术的评价方法是综合评价方法的核心部分, 但是在真实场景下的评价效果还是未知的。后续可以创建航天领域中智能程序合成技术的数据集, 利用提出的面向航天控制软件智能合成技术的综合评价方法, 对航天嵌入式软件智能合成工具或算法进行评价, 以验证评价方法的有效性。

对于提出的智能程序合成技术的评价方法, 只是对合理分配的权重系数进行了有效性验证, 但是没有尝试其他的权重系数分配。在后续的工作中, 可以利

用其他技术手段, 尝试寻找最优的权重系数分配, 以更有效地对智能程序合成技术进行评价。

5.3 局限性与发展趋势

现有的软件智能合成的评价方法仍未趋于成熟, 还在不断地完善和发展, 同样面临着许多不确定性, 以下重点阐述评价指标的阈值、评价指标体系的全面性、评价方法的有效性以及未来的发展方向。

评价指标的阈值是指在评估模型性能或进行决策时, 设定的一个临界点或者分割点。这个阈值对于判断结果的分类或者确定某个行为是否执行起着关键作用。如何根据不同的任务场景设置每个评价指标的阈值是需要解决的问题, 在未来设计评价指标阈值方面, 可能需要考虑以下因素: 业务需求, 根据实际应用的需求和成本效益分析来确定合适的阈值; 数据分布, 观察样本数据的分布特征, 选择能有效分离两类数据的阈值; 模型性能, 通过交叉验证或其他验证方法, 评估不同阈值下的模型性能, 选择模型性能最优的阈值。

评价指标体系的全面性是指在设计和实施评价指标体系时, 应该确保覆盖所有相关的、重要的和有价值的方面, 以全面地反映被评价对象的特性、性能或效果。现有的软件智能合成的评价方法都没有论证评价指标体系的全面性, 而设计一个全面的指标体系也是一个巨大的工程。未来在设计指标体系的全面性方面可以参考以下几个关键点: 多维度考量, 全面的评价方法应当从多个角度和层面来衡量目标; 全过程跟踪, 如果评价的是一个过程或者系统, 要求评价方法能够贯穿其整个生命周期; 定量与定性结合, 不仅包括可量化的数据指标, 也应包含定性描述和评估。

评价方法的有效性是指评价方法在实际应用中能否准确、可靠地衡量和反映被评价对象的特性、性能或效果。现有的智能程序合成的评价方法大多使用皮尔逊相关系数(线性关系、等方差、正态分布、无异常值)或者斯皮尔曼相关系数(非正态分布或者含有异常值), 通过生成代码在评价指标上的得分与人类打分的相关程度来判断评价方法或者评价指标的有效性。未来在设计评价方法的有效性方面可以参考以下几个关键点: 信度, 即有效的评价指标应该在不同时间、不同评价者或使用相同方法进行多次测量时具有相似的结果; 效度, 即评价方法是否真正测量了目标测量的内容; 操作性和实用性, 即评价方法在实际操作中的可行性。

参考文献

- [1] YANG Mengfei, GU Bin, DUAN Zhenhua, *et al.* Intelligent program synthesis framework and key scientific problems for embedded software[J]. *Chinese Space Science and Technology*, 2022, 42(4): 1-7 (杨孟飞, 顾斌, 段振华, 等. 嵌入式软件智能合成框架及关键科学问题 [J]. 中国空间科学技术, 2022, 42(4): 1-7)
- [2] SHIN R, POLOSUKHIN I, SONG D. Improving neural program synthesis with inferred execution traces[C]//Proceedings of the 32nd International Conference on Neural Information Processing Systems. Montreal: Curran Associates Inc., 2018: 8931-8940
- [3] HUANG D, ZHANG R, HU X, *et al.* Neural program synthesis with query[C]//The 10th International Conference on Learning Representations. Virtual Event: OpenReview. net, 2022
- [4] RAMANI G, KARANDE S. Synthesis of mathematical programs from natural language specifications[OL]. arXiv preprint: 2304. 03287, 2023
- [5] JAIN N, VAIDYANATH S, IYER A, *et al.* Jigsaw: large language models meet program synthesis[C]//Proceedings of the 44th International Conference on Software Engineering. Pittsburgh: ACM, 2022: 1219-1231
- [6] CHRISTAKOPOULOU K, KALAI A T. Glass-box program synthesis: a machine learning approach[C]//Proceedings of the 32nd AAAI Conference on Artificial Intelligence. New Orleans: AAAI Press, 2018: 646-653
- [7] ODENA A, SHI K, BIEBER D, *et al.* BUSTLE: bottom-up program synthesis through learning-guided exploration [C]//The 9th International Conference on Learning Representations. Austria: OpenReview. net, 2021
- [8] DUMANCIC S, GUNS T, CROPPER A. Knowledge re-factoring for inductive program synthesis[C]//Proceedings of the 35th AAAI Conference on Artificial Intelligence. Virtual Event: AAAI Press, 2021: 7271-7278
- [9] ROSIN C D. Stepping stones to inductive synthesis of low-level looping programs[C]//Proceedings of the 33rd AAAI Conference on Artificial Intelligence. Honolulu: AAAI Press, 2019: 2362-2370
- [10] ZOHAR A, WOLF L. Automatic program synthesis of long programs with a learned garbage collector[C]//Proceedings of the 32nd International Conference on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018: 2098-2107
- [11] HONG J, DOHAN D, SINGH R, *et al.* Latent programmer: discrete latent codes for program synthesis[C]//Proceedings of the 38th International Conference on Machine Learning. Virtual Event: PMLR, 2021: 4308-4318
- [12] SHI K, DAI H J, ELLIS K, *et al.* CROSSBEAM: learning to search in bottom-up program synthesis[C]//The 10th International Conference on Learning Representations. Virtual Event: OpenReview. net, 2022
- [13] KALYAN A, MOHTA A, POLOZOV O, *et al.* Neural-guided deductive search for real-time program synthesis from examples[C]//The 6th International Conference on Learning Representations. Vancouver: OpenReview. net, 2018
- [14] VALKOV L, CHAUDHARI D, SRIVASTAVA A, *et al.* HOUDINI: lifelong learning as program synthesis[C]//The 32nd International Conference on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018: 8701-8712
- [15] HANDA S, RINARD M C. Inductive program synthesis over noisy data[C]//The 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. USA: ACM, 2020: 87-98
- [16] NYE M I, HEWITT L B, TENENBAUM J B, *et al.* Learning to infer program sketches[C]//Proceedings of the 36th International Conference on Machine Learning. Long Beach: PMLR, 2019: 4861-4870
- [17] CHEN X Y, SONG D, TIAN Y D. Latent execution for neural program synthesis[C]//Proceedings of the 35th International Conference on Neural Information Processing Systems. Virtual Event: Curran Associates Inc., 2021: 22196-22208
- [18] FIJALKOW N, LAGARDE G, MATRICON T, *et al.* Scaling neural program synthesis with distribution-based search[C]//Proceedings of the 36th AAAI Conference on Artificial Intelligence. Virtual Event: AAAI Press, 2022: 6623-6630
- [19] THAKOOR S, SHAH S, RAMAKRISHNAN G, *et al.* Synthesis of programs from multimodal datasets[C]//Proceedings of the 32nd AAAI Conference on Artificial Intelligence. New Orleans: AAAI Press, 2018: 184-191
- [20] RAZA M, GULWANI S. Automated data extraction using predictive program synthesis[C]//Proceedings of the 31st AAAI Conference on Artificial Intelligence. San Francisco: AAAI Press, 2017: 882-890
- [21] QUIRK C, MOONEY R, GALLEY M. Language to code: learning semantic parsers for if-this-then-that recipes[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. Beijing: ACL, 2015: 878-888
- [22] ZHANG Y T. Scalability and precision improvement of neural program synthesis[C]//The 35th IEEE/ACM International Conference on Automated Software Engineering. Melbourne: IEEE, 2020: 1391-1393
- [23] CHASINS S, PHOTHILIMTHANA P M. Data-driven synthesis of full probabilistic programs[C]//The 29th International Conference on Computer Aided Verification. Heidelberg: Springer, 2017: 279-304
- [24] SI X J, LEE W, ZHANG R, *et al.* Syntax-guided synthesis of datalog programs[C]//The 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Lake Buena Vista: ACM, 2018: 515-527
- [25] LAICH L, BIELIK P, VECHEV M T. Guiding program synthesis by learning to generate examples[C]//The 8th International Conference on Learning Representations. Addis Ababa: OpenReview. net, 2020
- [26] ODENA A, SUTTON C. Learning to represent programs

- with property signatures[C]//The 8th International Conference on Learning Representations. Addis Ababa: OpenReview. net, 2020
- [27] SI X J, YANG Y, DAI H J, *et al.* Learning a meta-solver for syntax-guided program synthesis[C]//The 7th International Conference on Learning Representations. New Orleans: OpenReview. net, 2019
- [28] SHIN R, KANT N, GUPTA K, *et al.* Synthetic datasets for neural program synthesis[C]//The 7th International Conference on Learning Representations. New Orleans: OpenReview. net, 2019
- [29] CHEN X Y, LIU C, SONG D. Execution-guided neural program synthesis[C]//The 7th International Conference on Learning Representations. New Orleans: OpenReview. net, 2019
- [30] BUNEL R, HAUSKNECHT M J, DEVLIN J, *et al.* Leveraging grammar and reinforcement learning for neural program synthesis[C]//The 6th International Conference on Learning Representations. Vancouver: OpenReview. net, 2018
- [31] POLOSUKHIN I, SKIDANOV A. Neural program search: solving programming tasks from description and examples[C]//The 6th International Conference on Learning Representations. Vancouver: OpenReview. net, 2018
- [32] SHIN R, POLOSUKHIN I, SONG D. Towards specification-directed program repair[C]//The 6th International Conference on Learning Representations. Vancouver: OpenReview. net, 2018
- [33] PARISOTTO E, MOHAMED A R, SINGH R, *et al.* Neuro-symbolic program synthesis[C]//The 5th International Conference on Learning Representations. Toulon: OpenReview. net, 2017
- [34] BALOG M, GAUNT A L, BROCKSCHMIDT M, *et al.* DeepCoder: learning to write programs[C]//The 5th International Conference on Learning Representations. Toulon: OpenReview. net, 2017
- [35] ALET F, LOPEZ-CONTRERAS J, KOPPEL J, *et al.* A large-scale benchmark for few-shot program induction and synthesis[C]//Proceedings of the 38th International Conference on Machine Learning. Virtual Event: PMLR, 2021: 175-186
- [36] PU Y W, MIRANDA Z, SOLAR-LEZAMA A, *et al.* Selecting representative examples for program synthesis[C]//Proceedings of the 35th International Conference on Machine Learning. Stockholm: PMLR, 2018: 4158-4167
- [37] SUN S H, NOH H, SOMASUNDARAM S, *et al.* Neural program synthesis from diverse demonstration videos[C]//Proceedings of the 35th International Conference on Machine Learning. Stockholm: PMLR, 2018: 4797-4806
- [38] DEVLIN J, UESATO J, BHUPATIRAJU S, *et al.* RobustFill: neural program learning under noisy I/O[C]//Proceedings of the 34th International Conference on Machine Learning. Sydney: PMLR, 2017: 990-998
- [39] MENON A K, TAMUZ O, GULWANI S, *et al.* A machine learning framework for programming by example[C]//Proceedings of the 30th International Conference on Machine Learning. Atlanta: PMLR, 2013: 187-195
- [40] GU X D, ZHANG H Y, KIM S. Deep code search[C]//Proceedings of the 40th International Conference on Software Engineering. Gothenburg: ACM, 2018: 933-944
- [41] DESAI A, GULWANI S, HINGORANI V, *et al.* Program synthesis using natural language[C]//Proceedings of the 38th International Conference on Software Engineering. Austin: ACM, 2016: 345-356
- [42] SHRIVASTAVA D, LAROCHELLE H, TARLOW D. Learning to combine per-example solutions for neural program synthesis[C]//Proceedings of the 35th International Conference on Neural Information Processing Systems. Virtual Event: Curran Associates Inc., 2021: 6102-6114
- [43] CUI G F, ZHU H. Differentiable synthesis of program architectures[C]//Proceedings of the 35th International Conference on Neural Information Processing Systems. Virtual Event: Curran Associates Inc., 2021: 11123-11135
- [44] YANG Y D, INALA J P, BASTANI O, *et al.* Program synthesis guided reinforcement learning for partially observed environments[C]//Proceedings of the 34th International Conference on Neural Information Processing Systems. Virtual Event: Curran Associates Inc., 2021: 29669-29683
- [45] SHAH A, ZHAN E, SUN J J, *et al.* Learning differentiable programs with admissible neural heuristics[C]//Proceedings of the 34th International Conference on Neural Information Processing Systems. Virtual Event: Curran Associates Inc., 2020: 4940-4952
- [46] GUPTA K, CHRISTENSEN P E, CHEN X Y, *et al.* Synthesize, execute and debug: learning to repair for neural program synthesis[C]//Proceedings of the 34th International Conference on Neural Information Processing Systems. Virtual Event: Curran Associates Inc., 2020: 17685-17695
- [47] ELLIS K, NYE M, PU Y W, *et al.* Write, execute, assess: program synthesis with a REPL[C]//Proceedings of the 33rd International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019: 9165-9174
- [48] SHIN R, ALLAMANIS M, BROCKSCHMIDT M, *et al.* Program synthesis and semantic parsing with learned code idioms[C]//Proceedings of the 33rd International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019: 10824-10834
- [49] ELLIS K, MORALES L, SABLÉ-MEYER M, *et al.* Learning libraries of subroutines for neurally-guided Bayesian program induction[C]//Proceedings of the 32nd International Conference on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018: 7816-7826
- [50] ZHANG L, ROSENBLATT G, FETAYA E, *et al.* Neural guided constraint logic programming for program synthesis[C]//Proceedings of the 32nd International Conference on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018: 1744-1753
- [51] LIANG C, NOROUZI M, BERANT J, *et al.* Memory augmented policy optimization for program synthesis and semantic parsing[C]//Proceedings of the 32nd International Conference on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018: 10015-10027

- [52] CHEN X Y, LIU C, SHIN R, *et al.* Latent attention for if-then program synthesis[C]//Proceedings of the 30th International Conference on Neural Information Processing Systems. Barcelona: Curran Associates Inc., 2016: 4581-4589
- [53] ELLIS K, SOLAR-LEZAMA A, TENENBAUM J B. Unsupervised learning by program synthesis[C]//Proceedings of the 28th International Conference on Neural Information Processing Systems. Montréal: MIT Press, 2015: 973-981
- [54] ELLIS K, WONG C, NYE M, *et al.* DreamCoder: bootstrapping inductive program synthesis with wake-sleep library learning[C]//The 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation. Canada: ACM, 2021: 835-850
- [55] BEDNAREK J, PIASKOWSKI K, KRAWIEC K. Ain't nobody got time for coding: structure-aware program synthesis from natural language[OL]. arXiv preprint: 1810.09717, 2019
- [56] MURALI V, QI L, CHAUDHURI S, *et al.* Neural sketch learning for conditional program generation[C]//The 6th International Conference on Learning Representations. Vancouver: OpenReview. net, 2018
- [57] RAGHOTHAMAN M, WEI Y, HAMADI Y. SWIM: synthesizing what I mean-code search and idiomatic snippet synthesis[C]//2016 IEEE/ACM 38th International Conference on Software Engineering. Austin: ACM, 2016: 357-367
- [58] BHUPATIRAJU S, AGRAWAL K K, SINGH R. Towards mixed optimization for reinforcement learning with program synthesis[OL]. arXiv preprint: 1807.00403, 2018
- [59] PAPINENI K, ROUKOS S, WARD T, *et al.* BLEU: a method for automatic evaluation of machine translation[C]//Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. Philadelphia: ACL, 2002: 311-318
- [60] KULAL S, PASUPAT P, CHANDRA K, *et al.* SPoC: search-based pseudocode to code[C]//Proceedings of the 33rd International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019: 11883-11894
- [61] CHEN M, TWOREK J, JUN H, *et al.* Evaluating large language models trained on code[OL]. arXiv preprint: 2107.03374, 2021
- [62] BANERJEE S, LAVIE A. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments[C]//Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. Ann Arbor: ACL, 2005: 65-72
- [63] POPOVIĆ M. chrF: character n-gram F-score for automatic MT evaluation[C]//Proceedings of the 10th Workshop on Statistical Machine Translation. Lisbon: ACL, 2015: 392-395
- [64] POPOVIĆ M. chrF++: words helping character n-gram[C]//Proceedings of the 2nd Conference on Machine Translation. Copenhagen: ACL, 2017: 612-618
- [65] TRAN N, TRAN H, NGUYEN S, *et al.* Does BLEU score work for code migration?[C]//IEEE/ACM 27th International Conference on Program Comprehension. Montréal: IEEE, 2019: 165-176
- [66] REN S, GUO D Y, LU S, *et al.* CodeBLEU: a method for automatic evaluation of code synthesis[OL]. arXiv preprint: 2009.10297, 2020
- [67] PAN Y, LYU C. Measuring efficient code generation with GEC[C]//The 14th Asia-Pacific Symposium on Internetware. Hangzhou: ACM, 2023: 249-258
- [68] IMPROTA C. Poisoning programs by un-repairing code: security concerns of AI-generated code[C]//IEEE 34th International Symposium on Software Reliability Engineering Workshops. Florence: IEEE, 2023: 128-131
- [69] SIDDIQ M L, SANTOS J C S. SecurityEval dataset: mining vulnerability examples to evaluate machine learning-based code generation techniques[C]//The 1st International Workshop on Mining Software Repositories Applications for Privacy and Security. Singapore: ACM, 2022: 29-33
- [70] SU H R, AI J, YU D, *et al.* An evaluation method for large language models' code generation capability[C]//The 10th International Conference on Dependable Systems and Their Applications. Tokyo: IEEE, 2023: 831-838
- [71] KOVALCHUK S, FEDRUSHKOV D, LOMSHAKOV V, *et al.* Test-based and metric-based evaluation of code generation models for practical question answering[C]//The International Conference on Code Quality. St. Petersburg: IEEE, 2023: 73-86

作者简介



王鹤然 男, 1998年3月出生, 现为北京控制工程研究所助理工程师, 主要研究方向为程序合成、航天嵌入式软件智能合成技术评价方法。

E-mail: wangheranjisuanji@163.com



赵明 (通信作者) 男, 1995年2月出生, 现为北京控制工程研究所工程师, 主要研究方向为软件智能合成、嵌入式软件架构设计方法。

E-mail: zhaoming0205@outlook.com