arXiv:2303.16359v1 [cs.AI] 28 Mar 2023

通过将新任务合成为随堂测验，实现基于积木编程的自适应支架

Ahana Ghosh[1] Sebastian Tschiatschek[2] Sam Devlin[3] Adish Singla[1]MPI-SWS, {gahana、adishs}@mpi-sws.org[2]维也纳大学,
sebastian.tschiatschek@univie.ac.at[3]Microsoft 研究院,
sam.devlin@microsoft.com

摘要

基于代码块的编程环境越来越多地用于向初学者介绍计算概念。然而，鉴于编程任务在概念性和开放性方面的特点，初学者学生常常难以适应这些环境。为了有效支持难以完成给定任务的学生，提供引导学生走向解答的自适应支架十分重要。我们提出了一种基于随堂测验的支架框架，这些随堂测验以多项选择编程任务的形式呈现。为了自动生成这些随堂测验，我们提出了一种新颖的算法 PQUIZSYN。更正式地说，给定一个包含解答代码的参考任务以及学生当前尝试，PQUIZSYN 会合成用于随堂测验的新任务，并具备以下特征：(a) 自适应性（即针对学生当前尝试进行个性化），(b) 可理解性（即易于理解和解答），以及 (c) 隐蔽性（即不泄露解答代码）。我们的算法利用基于符号推理和基于图的代码表示技术来合成这些任务。我们表明，对于来自 Hour of Code：迷宫挑战 [11] 和 Karel [9]的参考任务，针对不同的学生作答，我们的算法能够生成数百个随堂测验。我们通过使用评估量规进行专家评分来评估这些随堂测验的质量。此外，我们还构建了一个在线平台，用于练习基于积木的编程任务，并通过基于突击测验的反馈为其提供支持，同时报告了一项初步用户研究的结果。

1 引言

基于代码块的可视化编程平台的兴起，使编程对初学者学生而言更具互动性和吸引力。基于积木的编程使用“代码块”，减轻了语法负担，并聚焦于关键的编程概念。在 Scratch 等语言取得成功的推动下， [33], Code.org 推出的 Hour of Code 等计划 [12], 以及 CodeHS.com 提供的 Karel 编程入门等在线课程 [9, 25], , 基于积木的编程已成为计算机科学入门教育不可或缺的一部分。

这些平台上的编程任务具有概念性且开放式，需要通过多步演绎推理才能解决，因此对学生来说颇具挑战。为了有效地帮助遇到困难的学生解决特定任务，提供针对其作答的反馈十分重要。然而，对于拥有数百万学生的平台而言，人工辅导者无法为学生提供反馈。因此，亟需自动反馈生成系统为学生提供个性化支持 [13, 22]。该领域已有研究探索了在任务中生成个性化反馈的各种方法，例如，以学生尝试中可使用的后续代码块的形式提供下一步提示 [26, 31, 35, 36, 44], 提供自适应示例 [28,32, 43], 以及对学生的错误概念进行数据驱动分析 [5, 6, 16, 24, 39–41]。

预印本。已被 AIED’22 会议接收为论文。

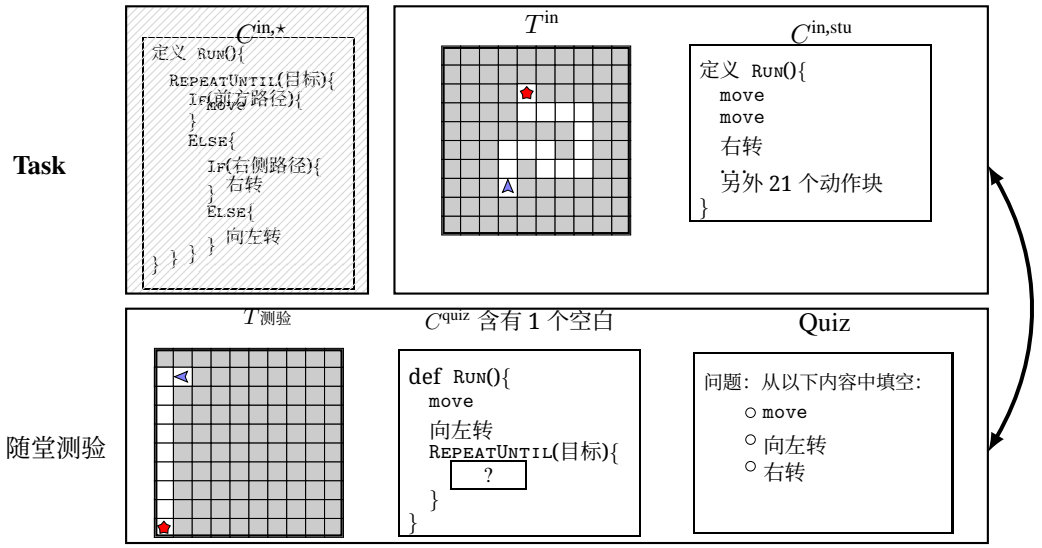


图1：我们基于随堂测验的框架示意图。“任务”面板显示了一个来自 HOC [11], 的输入任务 T^{in} 、学生当前的尝试 C^{instu}以及解答代码 $C^{in}{}_{\star}$（不会向学生透露）。学生目前未能成功解决该任务：当前尝试 C^{instu} 无法在允许的最大代码块数量（7 个代码块）内解决视觉谜题，也没有使用任何必需的构念（RepeatUntil 和 IfElse 构念）。“随堂测验”面板显示了我们的算法以任务-代码对（$T^{quiz}\,C^{quiz}$）形式生成的随堂测验，以及一道介绍 RepeatUntil构念的选择题。学生解决随堂测验后，会继续完成输入任务。当学生需要帮助时，系统将调用该框架；重要的是，呈现给学生的随堂测验会根据学生当前的尝试 C^{instu}进行自适应调整。此外，我们的算法生成的随堂测验易于理解和解决，并且 C^{quiz} 充分隐藏了 $C^{in}{}_{\star}$。

本文探讨了一种个性化反馈生成的替代方法，旨在引导学生找到任务的解决方案，同时融入探究驱动和问题解决的要素 [14]。具体而言，我们提出了一种基于随堂测验的支架框架，其中包含以多项选择题形式呈现的新编程任务。[1] 我们的框架受到先前研究的启发；这些研究表明，多项选择题有助于初学者学生学习编写代码 [17, 19, 30,38, 42]。该框架的使用方式如下：给定一项任务和学生当前未成功的尝试，框架可以通过提供一份旨在纠正其错误概念的随堂测验来帮助学生。为了使支架有效，我们围绕三个特征设计随堂测验中的新编程任务：自适应性、可理解性和隐蔽性；详情见图 1 和第 2.1 节。然而，手工设计这些新测验十分耗时，而且在需要针对大量任务和不同学生作答时，还可能容易出错。因此，我们尝试通过合成新的编程任务来自动生成这些随堂测验。

1.1 主要挑战与我们的贡献

利用上述特征合成新的视觉编程任务面临诸多挑战，包括以下几点：(i) 当前用于合成视觉编程任务的技术无法适应学生作答 [1]；(ii) 如 [1], 所示，从视觉任务空间到其解答代码的映射具有高度不连续性，因此基于任务变异的技术难以奏效 [27, 37]；(iii) 可能的任务及其解答空间具有潜在的无限性，因此依赖穷举的技术难以处理 [2, 4, 37]。

在本研究中，我们开发了一种新颖的算法 PQUIZSYN，用于综合生成具备我们支架框架所需特征的随堂测验。我们的算法通过采用符号执行、搜索算法和基于图的代码表示技术，克服了上述

[1] 我们将这些多项选择题称为“随堂测验”，因为该框架可以在学生需要帮助时随时呈现这些测验 [7]。

挑战。我们的主要贡献如下：（I）我们提出了一种模块化且可扩展的算法，用于生成分三个阶段运行的随堂测验（见第2节和第3节）；[2]（II）我们展示了该方法能够针对来自真实世界编程平台的参考任务中的不同类型学生作答，生成数百个随堂测验（见第4节）；（III）我们使用多维度评估量表，通过专家评分评估了算法的质量（见第 5 节）；（IV）我们基于该框架构建了一个在线平台，并通过初步用户研究展示了基于突击测验的反馈的效用（见第6节）。[3]

1.2 补充相关工作

通过建模编程概念提供反馈。除了上述下一步提示等方法外，还有大量工作致力于通过建模编程概念来生成反馈。在此，研究者提出了多种技术，包括：(a) 通过分析学生作答来检测难点概念 [6, 39, 40]；(b) 使用任务专用量规和神经程序嵌入发现学生误解 [41]；(c) 通过知识组件定义概念 [3, 15, 34]。

反馈方法的评估。开发反馈生成方法时，一个重要的考量因素是其评估标准。大多数下一步反馈生成方法都基于专家标注或自动化程序进行评估 [24, 26, 29]。相比之下，由示例驱动的反馈技术通常使用多维评分量表进行评估 [32, 43]。在我们的工作中，我们通过评分标准进行专家评分，并开展一项初步用户研究，以评估支架框架。

2 问题设定与定义

本节将形式化阐述我们的目标，并介绍重要的技术定义。

2.1 问题设置

任务空间。我们将任务空间定义为 $\mathbb{T}$。一个任务 $T \in \mathbb{T}$ 由一个视觉谜题和一组可用的代码块类型组成（例如，移动、 RepeatUntil），这些类型可用于解答代码。此外，解答代码中的代码块数量必须处于特定的大小阈值以内。我们将学生当前正在解决的任务表示为 $T^{\text{in}} \in \mathbb{T}$；参见图 1 中的 T^{in}。在本文中，我们使用来自 Code.org 的 Hour of Code：迷宫挑战 [11] [10] 以及来自 CodeHS.com 的 Karel 编程入门 [9] [8]中的任务；下文分别将它们称为 HOC 任务和 Karel任务。

代码空间。我们将所有可能代码的空间定义为 $\mathbb{C}$，并使用领域专用语言（DSL） [20]对其进行表示。具体而言，对于与 HOC 和 Karel任务相关的代码，我们使用基于 [1]的 DSL。代码 $C \in \mathbb{C}$ 具有以下属性： C_{blocks} 是 C中所使用的代码块类型集合， C_{size} 是所使用代码块的数量，而 C_{depth} 是 C的抽象语法树深度。我们将该空间中的距离度量表示为 $D_{\mathbb{C}}$。对于给定的 $C \in \mathbb{C}$ 和正整数 l，我们将邻域函数定义为 $\mathcal{N}_{\mathbb{C}}(C, l) = \{C' \mid D_{\mathbb{C}}(C', C) \leq l\}$。任务 T^{in} 的解答代码 C^{in}、$^{\star} \in \mathbb{C}$ 使用允许的代码块类型，并在指定的大小阈值内解决视觉谜题。T中的学生尝试表示为 C^{in}、$^{\text{stu}} \in \mathbb{C}$。

目标。对于输入任务 T^{in} 及其解答代码 C^{in},*，给定学生当前的尝试 C^{in},^{stu}，我们的目标是生成一个随堂测验，其形式为一个新的任务-代码对 $(T^{\text{quiz}}, C^{\text{quiz}})$，并根据以下特征进行设计：(i) 自适应性，即 C^{quiz} 考虑到 C^{in},* 和 C^{in},^{stu}，确保 C^{quiz} 针对学生当前的尝试进行个性化设计；(ii) 可理解性，即 C^{quiz} 正确地解答 T^{quiz}，并且随堂测验易于理解和解答，不会使学生感到困惑；(iii) 隐蔽性，即 $D_{\mathbb{C}}(C^{\text{quiz}}, C^{\text{in}}$,$^{\star})$ 较高，确保 C^{quiz} 充分隐蔽解答代码 C^{in},*，而不会直接透露答案，以鼓励问题解决。

[2]该算法的实现已公开发布于 https://github.com/machine-teaching-group/aied2022_pquizsyn_代码[3]在线平台已公开发布于 https://www.teaching-blocks-hints.cc/

2.2 技术定义

草图空间。我们使用一种称为草图 [2, 37] 的更高层次抽象，来捕捉代码的关键概念元素。代码的草图保留了其中重要的编程构造。与代码 DSL 类似，我们基于 [1]，使用草图 DSL 定义草图空间 $\mathbb{S}$。与代码的抽象语法树表示类似，我们将草图表示为一棵树，其节点为编程构造。从代码空间到草图空间的映射由多对一映射 $\Psi\colon \mathbb{C} \to \mathbb{S}$ 表示；也就是说，代码 C 在 $\mathbb{S}$ 中的表示为 $\Psi(C)$。由于 $\mathbb{S}$ 是 $\mathbb{C}$ 的一种抽象，$\mathbb{C}$ 中的多个元素可以对应于 $\mathbb{S}$ 中的单个元素。与 $D_{\mathbb{C}}$ 和 $\mathcal{N}_{\mathbb{C}}$ 类似，我们将草图空间中的距离度量记为 $D_{\mathbb{S}}$，将邻域函数记为 $\mathcal{N}_{\mathbb{S}}(S, l) = \{S' \mid D_{\mathbb{S}}(S', S) \leq l\}$，其中给定一个 $S \in \mathbb{S}$ 和一个正整数 l。

草图子结构。对于草图 S，我们将子结构定义为一棵包含 S 截至特定深度的节点且共享同一根节点的子树；请注意，草图的子结构也是草图。我们将 S 的所有子结构组成的集合记为 SUBSTRUCTS(S) $\subseteq \mathbb{S}$；集合 SUBSTRUCTS(S) 的大小通常较小。例如，图 2b 所示的草图具有以下 4 个子结构：(i) {Run}；(ii) {Run {RepeatUntil（目标）}}；(iii) {Run {RepeatUntil（目标）{IfElse（B）}}}；以及 (iv) {Run {RepeatUntil（目标）{IfElse （B）{{}； {IfElse （B）}}}}}。

代码约简。对于带有草图 S 的代码 $C \in \mathbb{C} := \Psi(C)$，考虑草图 $S_{\text{sub}} \in$ SUBSTRUCTS(S) 中的一个。我们将相对于草图 S_{sub} 的 C 的代码约简集合定义为：在保留草图 S_{sub} 的同时移除 C 的一个或多个节点后得到的所有代码；请注意，代码的约简也是代码。我们将所有约简组成的集合记为 REDCODES($C \mid S_{\text{sub}}$) $\subseteq \mathbb{C}$。例如，对于图 1 中的 C^{in}、$\star$ 和 $S_{\text{sub}} =$ {Run{RepeatUntil（目标）}}，集合 REDCODES($C^{\text{in}},^{\star} \mid S_{\text{sub}}$) 具有以下 3 个代码：(i) {Run {RepeatUntil （目标） { 移动 }}}；(ii) {Run{RepeatUntil（目标）{右转}}}；以及 (iii) {Run{RepeatUntil（目标）{向左转}}}。

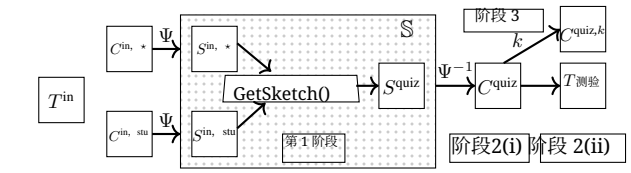
3 我们的算法 PQUIZSYN

本节将介绍我们的算法，该算法通过合成新任务来生成随堂测验。有人可能会想，先生成一个新的视觉谜题，再获取其解答代码，以此来合成任务。正如第 1 节所讨论的，以及 [1], 所示，从视觉任务空间到其解答代码的映射具有高度的不连续性，因此直接在任务空间中推理出理想任务并不可行。然而，由于我们希望生成的任务也能考虑学生当前的尝试，因此 [1] 中的任务合成算法并不适用于我们的工作。为此，我们开发了一种新颖的算法 PQUIZSYN（通过合成生成编程随堂测验），用于生成能够适应学生当前尝试的任务。我们的算法分为三个阶段：（i）第 1 阶段根据任务的解答代码和学生当前的尝试生成草图；（ii）阶段 2 将该草图实例化为一个新的任务-代码对；（iii）阶段 3 根据新的任务-代码对生成随堂测验。图 2a 展示了这些阶段，具体细节如下。

3.1 第 1 阶段：生成快问快答草图 S^{quiz}

我们首先描述算法的第 1 阶段，如图 2a 所示。在此阶段，`GetSketch()` 例程会返回一个合适的草图 S^{quiz}，供后续阶段实例化。该例程的输入是学生草图 $S^{\text{in}},^{\text{stu}} := \Psi(C^{\text{in}},^{\text{stu}})$ 和解答草图 $S^{\text{in}},^{\star} := \Psi(C^{\text{in}},^{\star})$。通过先在草图空间中操作，我们可以在后续阶段生成有意义的自适应代码。为了根据第 2.1 节提到的特征生成随堂测验，随堂测验草图 S^{quiz} 需要具备以下属性：(i) S^{quiz} 应引导学生找到解答草图 $S^{\text{in}},^{\star}$，即 $D_{\mathbb{S}}(S^{\text{quiz}}, S^{\text{in}},^{\star})$ 应较低；(ii) S^{quiz} 应根据学生草图 $S^{\text{in}},^{\text{stu}}$ 进行自适应调整，即当 l 的值较低时，$S^{\text{quiz}} \in \mathcal{N}_{\mathbb{S}}(S^{\text{in}},^{\text{stu}}, l)$ 应较低。虽然这些条件确保 S^{quiz} 能够引导学生找到解答草图并具有自适应性，但它们也可能导致生成的草图不属于解答草图的子结构集合，即 $S^{\text{quiz}} \notin$ SUBSTRUCTS($S^{\text{in}},^{\star}$)——在这种情况下，相对于 S^{quiz}，$C^{\text{in}},^{\star}$ 不存在有效代码约简（见第 2.2 节），这使得将草图实例化为理想代码颇具挑战性 C^{quiz}（见第 5 节中的算法变体 PQS-ONEHOP 和脚注 4）。因此，`GetSketch()` 按如下方式生成 S^{quiz}（见图 3）：

(i) 选择 $\hat{l}$ 作为满足 $\mathcal{N}_{\mathbb{S}}(S^{\text{in}},^{\text{stu}}, l) \cap$ SUBSTRUCTS($S^{\text{in}},^{\star}$) 非空的最小 $l \in \{1, 2, \ldots\}$。(ii) 生成 $S^{\text{quiz}} \in \text{argmin}_{S \in \mathcal{N}_{\mathbb{S}}(S^{\text{in}},^{\text{stu}}, \hat{l}) \cap \text{SubStructs}(S^{\text{in}},^{\star})} D_{\mathbb{S}}(S, S^{\text{in}},^{\star})$。



(a) 我们的算法 PQUIZSYN

```
def Run(){
 RepeatUntil(目标){
    If(B){}
    Else{
      If(B){}
      Else{}
    }
 }
}
```

(b) $S^{\text{in},\star}$

```
定义 Run(){}
```

(c) $S^{\text{in, stu}}$

```
def Run(){
 RepeatUntil(目标){}
}
```

(d) S^{quiz}

```
def Run(){
  move
  向左转
  RepeatUntil(目标){
    move
  }
}
```

(e) C^{quiz}

图2：（a）展示了 PQUIZSYN。具体而言，我们可以使用输入任务 T^{in}、其解答代码 C^{in}、$\star$ 以及图1中的当前学生尝试 C^{in}、^{stu} 来实例化所介绍的算法。C^{in}、$\star$ 的草图如（b）所示，C^{in}、^{stu} 的草图如（c）所示，C^{quiz} 的草图如（d）所示，而随堂测验 C^{quiz} 的代码如（e）所示。

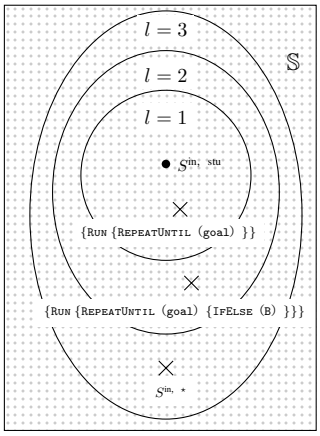


图3：针对图1所示场景的 PQUIZSYN 阶段1。X 展示了 $l \in \{1,2,3\}$ 的 S^{in}、^{stu} 的 l 跳邻域中 S^{in}、$\star$ 的子结构。详情见第3.1节。

3.2 阶段 2：从 S^{quiz} 合成（T^{quiz}，C^{quiz}）

接下来，我们介绍算法的阶段 2。首先，如图 2a 的阶段2(i)所示，我们从 S^{quiz}生成 C^{quiz} 。具体而言，对于第 1 阶段生成的草图 S^{quiz}，我们采用 [1] 中提出的代码变异方法，以获得代码 C^{quiz}。然而，该方法需要有意义的起始代码 C^{seed}。由于按照第 1 阶段的设计，$S^{\text{quiz}} \in$ SUBSTRUCTS(S^{in},$\star$)，我们首先从 REDCODES(C^{in},*| S^{quiz}) 集合中选取 C^{seed} 。 [1] 的方法为我们提供了针对 C^{seed}的多种代码变异。这些代码变异与 C^{seed} 和 C^{in}的差异程度$_{\star}$，由基于布尔变量、条件语句和 C^{seed}的动作块（移动、向左转、右转、拾取标记、放置标记）取值所施加的约束，以及对所得代码大小的约束共同控制。具体而言，这些变异使我们能够控制 $D_{\mathbb{C}}(C^{\text{quiz}}, C^{\text{in}},\star)$ 的变化程度，这是第 2.1 节所述的一项理想特性。

接下来，如图 2a 的阶段 2(ii) 所示，我们根据代码 C^{quiz} 生成一个新任务 T^{quiz}。具体来说，我们生成 T^{quiz}，使其解答代码为 C^{quiz}。我们结合符号执行和最佳优先搜索技术来实现这一点，并以 [1] 中提出的任务合成方法为基础。

3.3 阶段 3：从（T^{quiz}，C^{quiz}）生成多选题

在此阶段，我们生成一道包含一组固定答案选项的随堂测验；见图1和图5。我们选取一个任务-代码对（T^{quiz}，C^{quiz}），并根据暴露参数 k 确定仅展示 C^{quiz} 的一部分，即 C^{quiz} 中包含 k 个空白。这些空白必须由学生从答案选项中选出内容填入，从而解答 T^{quiz}。具体来说，我们生成包含 $k = 1$ 个空白的随堂测验。为了确定测验中的空白，我们对 C^{quiz} 进行中序遍历，并将最后一个叶节点留作空白。

4 PQUIZSYN 在现实世界任务中的应用

在本节中，我们展示 PQUIZSYN 在从真实的基于积木的编程平台中选取的六个参考任务上的表现：HOC [11] 和 Karel [9]。图4列出了这些任务及其来源。这些任务的复杂性各不相同，其衡量依据是解答代码中的编程构造；各自解答草图的多样性也体现了这一点 S^{in},$\star$。对于 S^{in},$\star$ 的完整子结构集合，图4列出了总数

[4]当 $S^{\text{quiz}} \notin$ SUBSTRUCTS($S^{\text{in, }\star}$) 时，我们将 C^{seed} 设置为 S^{quiz} 的随机实例化结果——参见第 5 节中的算法变体 PQS-ONEHOP。

名称，T^{in}的来源	$C^{\text{in},\star}_{T^{\text{in}}}$的尺寸，$S^{\text{in},\star}$	$S^{\text{quiz}} \in$子结构$(S^{\text{in},\star})$	$\#C^{\text{quiz}}$	$\#T$测验
T-1 HOC：迷宫**08** [11]	6 {Run {Repeat; Repeat}}	{Run} {Run {Repeat}} $S^{\text{in},\star}$	22 34 179	220 340 1790
T-2 HOC：迷宫**16** [11]	5 {Run {RUntil {If}}}	{Run} {Run {RUntil}} $S^{\text{in},\star}$	10 6 19	100 60 190
T-3 HOC：迷宫**18** [11]	5 {Run {RUntil {IfElse}}	{Run} {Run {RUntil}} $S^{\text{in},\star}$	10 6 9	100 60 90
T-4 HOC：迷宫**20** [11]	7 {Run {RUntil {IfElse {{}; {IfElse}}}}}	{Run} {Run {RUntil}} {Run {RUntil {IfElse}}} $S^{\text{in},\star}$	10 6 9 10	100 60 90 100
T-5 Karel:Opposite [9]	6 {Run {Repeat {IfElse}}}	{Run} {Run {Repeat}} $S^{\text{in},\star}$	73 118 343	730 1180 3430
T-6 Karel:Diagonal [9]	8 {Run {While}}	{Run} $S^{\text{in},\star}$	447 579	4470 5790

Fig图 4：PQUIZSYN 应用于六个 HOC 和 Karel 参考任务；详情请参见第4节。Fo r bre为简洁起见，草图已缩写，例如，将 RepeatUntil(目标)写作 RUntil。

我们的算法生成的随堂测验数量，这些测验以唯一任务-代码对（$T^{\text{quiz}}, C^{\text{quiz}}$）的形式呈现。从图中可以看出，对于每个子结构，我们的算法会生成 50 到数千个随堂测验。对于这些任务的任何潜在学生尝试，按照设计，PQUIZSYN 的第 1 阶段都会生成一个此类特定任务子结构——因此，对于每次尝试，我们都可以向学生提供若干独特且自适应的随堂测验。请注意，对于每个子结构，我们的算法生成的任务数量高于代码数量。这是因为阶段 2(ii) 中使用的任务合成方法可以在图 2a 的阶段 2(ii) 中为单个代码生成多个任务。具体而言，对于每个新代码，我们会获得 10 个不同的任务。例如，图 1 和图 5 分别展示了 PQUIZSYN 针对任务 T-4 和 T-5 上特定学生尝试生成的随堂测验。

5 通过多维评分量表进行专家研究

在本节中，我们根据目标中指定的期望特征，即自适应性、可理解性和隐蔽性（见第 2.1 节），评估 PQUIZSYN。具体而言，我们旨在将 PQUIZSYN 与第 3 节中因采用不同设计选择而产生的变体进行比较。为此，我们通过多维评分量表开展专家研究。

PQUIZSYN 算法的变体。我们将 PQUIZSYN 的性能与以下变体进行比较：PQS-FULLHOP、PQS-ONEHOP 和 PQS-REDCODE。PQS-FULLHOP 和 PQS-ONEHOP 与 PQUIZSYN 的区别仅在于图 2a 第 1 阶段生成 S^{quiz} 时所用的 `GetSketch()` 例程。具体而言，PQS-FULLHOP 的第 1 阶段始终返回解答代码的草图，即 $S^{\text{quiz}} := S^{\text{in},\star}$；PQS-ONEHOP 的第 1 阶段直接从 $S^{\text{in},\text{stu}}$ 的一跳邻域中返回草图，即 $S^{\text{quiz}} \in \mathcal{N}_{\mathbb{S}}(S^{\text{in},\text{stu}}, 1)$。第三个基线 PQS-REDCODE 与 PQUIZSYN 的区别仅在于图 2a 的阶段2(i)，即从 S^{quiz} 生成 C^{quiz} 时。具体而言，PQS-REDCODE 的阶段2(i) 会根据第 1 阶段获得的草图，对解答代码进行直接约简，生成 C^{quiz}，即 $C^{\text{quiz}} \in \text{REDCODES}(C^{\text{in},\star} \mid S^{\text{quiz}})$。

模拟学生尝试。为了进行这项专家评估，我们模拟了基于代码块的编程领域中出现的学生失败尝试 [26]。具体而言，对于每个参考任务，我们手动创建了四种学生尝试，如下所示：(a) Stu-A：$C^{\text{in},\text{stu}}$ 仅使用动作块，即（移动、向左转、右转、拾取标记、放置标记）；(b) Stu-B：$C^{\text{in},\text{stu}}$ 使用了 $C^{\text{in},\star}$中一部分编程构念；(c) Stu-C：$C^{\text{in},\text{stu}}$ 与 $C^{\text{in},\star}$在结构上相同，即 $S^{\text{in},\text{stu}} = S^{\text{in},\star}$；(d) Stu-D：$C^{\text{in},\text{stu}}$ 的结构比 $C^{\text{in},\star}$更复杂。这四种尝试类型涵盖了算法部署时可能遇到的所有场景（参见第 6 节）。

多维度评估量表。受 [32, 43], 中的评估量规启发，我们采用包含三个属性的多维量表来评估随堂测验，每个属性均使用三点评分量表进行评分（分数越高越好）。具体而言，我们设置了：（i）自适应属性，用于衡量随堂测验针对当前学生尝试的个性化程度（3：高；2：中；

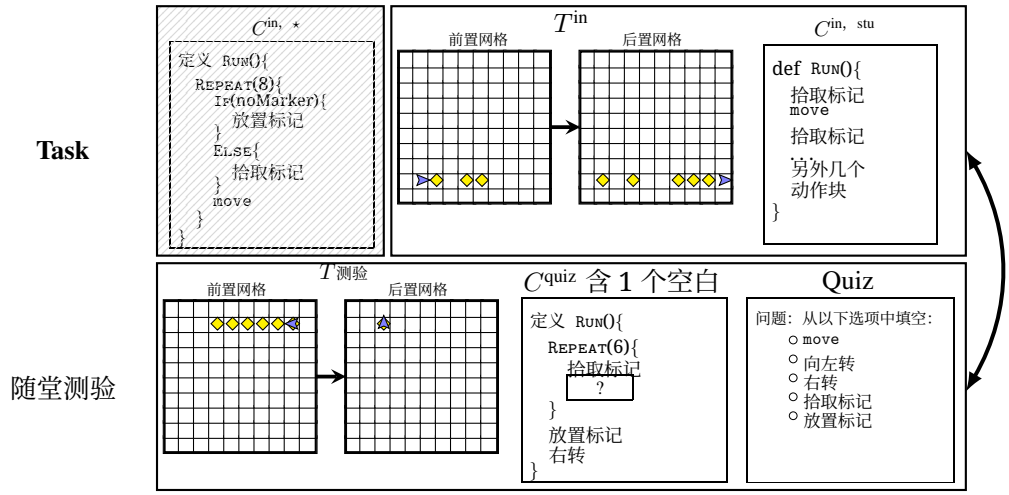


图5：与图1类似，此处我们以一个 Karel 任务 T-5（见图4）展示我们的框架。Karel任务 [25] 由一对视觉网格（前置网格、后置网格）组成，目标是编写代码，使其在执行时将前置网格转换为后置网格。

1：低）；（ii）可理解性属性，用于衡量随堂测验是否易于理解和解答（3：容易；2：有时可能会让学生困惑；1：要么不正确，要么极难解答）；（iii）隐藏性属性，用于衡量随堂测验隐藏解答代码的程度（3：充分隐藏；2：部分揭示解答；1：在很大程度上揭示解答）。总体分数表示一项随堂测验三个属性得分之和。

专家研究设置。我们选取了三个涵盖不同构念类型和复杂性的任务：图4中的 T-1、T-4 和 T-5。因此，我们总共评估了 48 个场景：4 种算法变体 $\times 4$ 学生类型 $\times 3$ 任务（图1和图5展示了示例场景）。两位具有基于积木编程经验的研究人员分别独立评估了这 48 个场景。评估通过网络调查进行：调查随机呈现一个场景，并依据评分标准进行评估。

专家研究结果。首先，我们使用各属性的二次加权科恩卡帕系数评定者间一致性信度值[32] 来验证专家评分：0.62（自适应性）、0.69（易理解性）、0.79（隐蔽性）和 0.7（总体）。这些数值表明评分者之间具有较高的一致性。平均评分见图6，且

算法	自适应可理解隐藏			总体
PQS-FullHop	2	2.8(0.1)	3.0(0.0)	7.8(0.8)
PQS-ONEHOP	2.8（0.1）	2.5(0.6)	3.0(0.0)	8.3(0.7)
PQS-REDCODE	2.7(0.3)	3.0(0.0)	1.5(0.4)	7.2(0.7)
PQUIZSYN	2.7(0.2)	3.0(0.0)	2.9(0.1)	8.6(0.3)

图6：不同算法的属性评分均值（方差）。分数越高越好。PQUIZSYN 在所有三个属性上表现出色，且总体得分最高；详情请参见第 5 节。

PQUIZSYN 的总体得分最高。我们根据克鲁斯卡尔-沃利斯显著性检验 [21]，按属性分析这些评分；接下来讨论的结果在统计上具有显著性，且$p < 0.01$。在自适应性属性方面，PQS-FULLHOP 的表现显著较差，因为它没有考虑学生尝试（参见第3.1节）。在可理解性属性方面，PQS- ONEHOP 的表现显著较差，因为存在一些情况：未找到相对于 S^{quiz} 的 $C^{in,\star}$ 的有效代码约简（参见脚注 4、第3.2节）。最后，在隐藏性属性方面，PQS-REDCODE 的表现显著较差，因为它通过直接约简$_{C^{in,\star}}$获得了 C^{quiz}，且未进行任何变异（参见第3.2节）。

6 通过在线平台进行用户研究

我们使用 Blockly Games 程序库 [18]，基于 PQUIZSYN 框架构建了一个在线平台。该在线平台向公众开放——请参见脚注3、第1.1节。平台为参与者提供了一个界面，可用于练习基于积木的编程任务，并在遇到困难时获得突击测验

式反馈。本节报告一项初步用户研究的结果，旨在评估我们的支架框架与其他反馈方法相比的有效性。

参与会话和反馈方法。我们平台上的单次会话由三个步骤组成。在步骤 A 中，参与者会看到一个任务，并有 10 次执行机会来解决该任务。如果参与者未能在步骤 A 解决任务，则会进入步骤 B，并随机分配一种反馈方法（NOHINT、NEXTSTEP 和 PQUIZSYN，具体如下所述）。完成步骤 B 后，参与者会在步骤 C 中继续尝试该任务，并额外获得 10 次执行机会。请注意，在单次会话中，反馈方法只会调用一次。接下来，我们将介绍步骤 B 中的不同反馈方法。NOHINT 表示一种基线情形，即参与者在没有任何反馈的情况下直接进入步骤 C。NEXTSTEP 指的是作为反馈的下一步提示：系统会更新参与者的代码，使其更接近解答代码 [26, 31, 35, 36, 44]；我们优先采用涉及编程构造的下一步编辑（例如， RepeatUntil），而不是涉及基本动作的编辑（例如，移动）。PQUIZSYN 是我们的基于突击测验的反馈。

用户研究结果。我们对从 A mazon Mechanical Turk 招募的参与者开展了一项初步用户研究；研究开始前已获得机构审查委员会的批准。参与者均为美国成年人，没有基于积木的可视化编程方面的专业知识。鉴于研究成本较高（每位参与者超过 3 美元），我们选取了两个任务进行研究：图4中的 T-3 和 T-5。详细结果见图7。共有 575 名独立参与者；其中 0.774

反馈总数	（步骤 B）		解决比例（步骤 C）		
	两者	T-3 T-5	两者	T-3	T-5
NOHINT	151	63 88	0.046	0.079	0.023
NEXTSTEP	146	63 83	0.082	0.127	0.048
PQUIZSYN	148	86		0.177	0.093

图7：任务 T-3 和 T-5 的结果（“Both”表示汇总结果）。在 Step-A 中，共有 575 名参与者（T-3 为 293 名，T-5 为 282 名）；其中约 0.774（T-3 为 0.642，T-5 为 0.911）的参与者未能在 Step-A 完成任务，随后进入 Step-B / Step-C，并被随机分配到一种反馈方式。

有一部分参与者未能在步骤 A 解决任务，随后进入步骤 B。在步骤 B 中，PQUIZSYN 被分配给 148 名参与者（其中 0.60 的参与者成功解决了所呈现的随堂测验）。随后，这些参与者中有 0.128 的比例在步骤 C 解决了任务。这里，0.128 表示被分配到 PQUIZSYN 的参与者的成功率；相比之下，NEXTSTEP 为 0.082，NOHINT 为 0.046——见图 7。总体而言，PQUIZSYN 的表现优于 NEXTSTEP（根据 χ^2 检验，差异不显著，$p = 0.19$）和 NOHINT（根据 χ^2 检验，差异显著，$p = 0.01$） [23]。这些初步结果证明了提供基于随堂测验的反馈的实用性。

7 结论与展望

我们提出了一种新颖的、基于随堂测验的基于积木的编程支架框架，其中融入了探究驱动和问题解决等方面。我们开发了模块化合成算法 PQUIZSYN，用于生成这些随堂测验。在使用多维评分量表进行专家评估后，我们开发了一个由该支架框架支持的在线平台。尽管使用该平台开展的初步用户研究结果证明了我们的基于随堂测验的框架的实用性，但仍有若干值得继续研究的方向，包括：（i）扩展平台，使其能够在单次参与会话中提供多轮反馈，并测量不同方法的功效；（ii）将我们合成的随堂测验与专家生成的测验进行比较；（iii）对初学者学生开展纵向研究，以测量问题解决能力的长期提升；（iv）将我们的框架扩展到更复杂的基于代码块的编程领域。

致谢。我们感谢审稿人提供的反馈。Ahana Ghosh 对 Microsoft Research 通过其博士奖学金计划提供的支持表示感谢。由欧盟资助/共同资助（欧洲研究委员会、TOPS、101039090）。然而，文中表达的观点和意见仅代表作者个人，并不一定反映欧盟或欧洲研究委员会的观点和意见。欧盟和资助机构均不对这些观点和意见承担责任。

参考文献

[1] Umair Z. Ahmed、Maria Christakis、Aleksandr Efremov、Nigel Fer nandez、Ahana Ghosh、Ab- hik Roychoudhury 和 Adish Singla。为基于积木的编程合成任务。发表于 NeurIPS，2020年。[2] Umair Z Ahmed、Sumit Gulwani 和 Amey Karkare。自动生成自然演绎题目及其解答。发表于 IJCAI，2013年。[3] Bita Akram、Hamoon Azizsoltani、Wookhee Min、Eric N. Wiebe、Bradford W. Mott、Anam Navied、Kristy Elizabeth Boyer 和 James C. Lester。利用高斯过程回归从学生程序中自动评估计算机科学能力。发表于 EDM，2020年。[4] Chris Alvin、Sumit Gulwani、Rupak Majumdar 和 Supratik Mukhopadhyay。几何证明题的合成。发表于 AAAI，2014年。[5] Rudy Bunel、Matthew J. Hausknecht、Jacob Devlin、Rishabh Singh 和 Pushmeet Kohli。利用语法和强化学习进行神经程序合成。发表于 ICLR，2018年。[6] Yuliya Cherenkova、Daniel Zingaro 和 Andrew Petersen。在大型题目数据集中识别具有挑战性的 CS1 概念。发表于 SIGCSE，2014年。[7] Vincent A. Cicirello。随堂测验在 CS1 中的作用与有效性。发表于 SIGCSE，2009 年。[8] CodeHS.com。CodeHS—— 教授编程和计算机科学。https://codehs.com/。[9]CodeHS.com。使用小狗Karel 学习编程入门。https://codehs.com/info/curriculum/introkarel。[10] Code.org。Code.org——学习计算机科学。https://code.org/。[11]Code.org。Hour of Code——经典迷宫挑战。https://studio.code.org/s/hourofcode。[12] Code.org。编程一小时倡议。https://hourofcode.com/。[13] Christa Cody、Mehak Maniktala、Nicholas Lytle、Min Chi 和 Tiffany Barnes。进一步前瞻的影响：在智能数据驱动逻辑辅导系统中比较两种数据驱动的非请求式提示类型对表现的影响。IJAIED，2021年。[14] Lucas Cordova、Jeffrey C. Carver、Noah Gershmel 和 Gursimran Walia。软件测试教育中探究式概念反馈与传统详细反馈机制的比较：一项实证研究。发表于 SIGCSE，2021年。[15] Will Crichton、Georgia Gabriela Sampaio 和 Pat Hanrahan。自动化程序结构分类。发表于 SIGCSE，2021年。[16] Aleksandr Efremov、Ahana Ghosh 和 Adish Singla。通过强化学习和程序合成进行提示策略的零样本学习。发表于 EDM，2020年。[17] Alexandru Ene 和 Cosmin Stirbu。自动生成 Java编程语言测验。发表于 ECAI，2019年。[18] Blockly Games。面向未来程序员的游戏。https://blockly.games/。[19] Shuchi Grover。构建面向 K-12计算机科学课堂的概念学习形成性评估框架。发表于 SIGCSE，2021年。[20] Sumit Gulwani、Oleksandr Polozov 和 Rishabh Singh。程序合成。《编程语言基础与趋势》，2017年。[21] Thomas W MacFarland 和 Jan M Yates。基于秩的 Kruskal–Wallis H检验：单因素方差分析（方差分析）。载于《使用 R 进行生物科学非参数统计入门》。施普林格，2016年。[22] Samiha Marwan、Ge Gao、Susan Fisk、Thomas Price 和 Tiffany Barnes。自适应即时反馈能够提高编程初学者的参与度以及在计算机科学领域持续学习的意愿。发表于 ICER，2020年。[23] Mary L McHugh

. 卡方独立性检验 . Biochemia Medica , 2013.

[24] Benjamin Paaßen、Barbara Hammer、Thomas William Price、Tiffany Barnes 和 Niels Pinkwart。《连续提示工厂——在连续且无限的空间中提供提示》。JEDM，2018年。[25] Richard E Pattis。《机器人卡雷尔：编程艺术入门》。约翰·威立父子出版公司，1981年。[26] Chris Piech、Mehran Sahami、Jonathan Huang 和 Leonidas J. Guibas。《通过推断问题解决策略自主生成提示》。载于 L@S，2015年。[27] Oleksandr Polozov、Eleanor O'Rourke、Adam M. Smith、Luke Zettlemoyer、Sumit Gulwani 和 Zoran Popovic。《个性化数学文字题生成》。载于 IJCAI，2015年。[28] Thomas W Price、Yihuan Dong 和 Dragan Lipovac。《iSnap：迈向新手编程环境中的智能辅导》。载于 SIGCSE，2017年。[29] Thomas W. Price、Yihuan Dong、Rui Zhi、Benjamin Paaßen、Nicholas Lytle、Veronica Catet´e 和 Tiffany Barnes。《数据驱动的编程提示生成算法质量比较》。IJAIED，2019年。[30] Thomas W. Price、Joseph Jay Williams、Jaemarie Solyst 和 Samiha Marwan。《通过在线编程作业中的教师解答吸引学生》。载于 CHI，2020年。[31] Thomas W. Price、Rui Zhi 和 Tiffany Barnes。《开放式编程数据驱动反馈算法的评估》。载于 EDM，2017年。[32] Thomas W. Price、Rui Zhi 和 Tiffany Barnes。《不确定性下的提示生成：提示质量对寻求帮助行为的影响》。载于 AIED，2017年。[33] Mitchel Resnick 等。《Scratch：人人都能编程》。ACM，2009年。[34] Kelly Rivers、Erik Harpstead 和 Kenneth R. Koedinger。《编程学习曲线分析：学生在哪些概念上遇到困难？》。载于 ICER，2016年。[35] Kelly Rivers 和 Kenneth R. Koedinger。《通过解空间路径构建实现提示生成自动化》。载于 ITS，2014年。[36] Kelly Rivers 和 Kenneth R. Koedinger。《广阔解空间中的数据驱动提示生成：一种自我改进的 Python 编程辅导系统》。IJAIED，2017年。[37] Rohit Singh、Sumit Gulwani 和 Sriram Rajamani。《自动生成代数题》。载于 AAAI，2012年。[38] Reza Soltanpoor、Charles Thevathayan 和 Daryl J. D'Souza。《通过个性化处方测验为编程新手提供自适应补救教学》。载于 ITiCSE，2018年。[39] Eliane Stampfer Wiese、Anna N. Rafferty 和 Armando Fox。《新手的代码可读性、结构与理解之间的联系：事情并不简单》。载于 ICSE，2019年。[40] Eliane Stampfer Wiese、Anna N. Rafferty、Daniel M. Kopta 和 Jacqulyn M. Anderson。《复现新手在编码风格方面的困难》。载于 ICPC，2019年。[41] Mike Wu、Milan Mosse、Noah D. Goodman 和 Chris Piech。《代码教育的零样本学习：结合深度学习推理的评分标准抽样》。载于 AAAI，2019年。[42] Lishan Zhang、Baoping Li、Qiujie Zhang 和 I. Hsiao。《针对选择题的分布式练习策略是否有助于新手学习编程？》。iJET，2020年。[43] Rui Zhi、Samiha Marwan、Yihuan Dong、Nicholas Lytle、Thomas W. Price 和 Tiffany Barnes。《面向新手编程的数据驱动示例反馈》。载于 EDM，2019年。[44] Kurtis Zimmerman 和 Chandan Raj Rupakheti。《用于向新手推荐程序元素的自动化框架（N）》。载于 ASE，2015年。