使用 Isabelle 证明助手进行考试

Frederik Krogsdal Jacobsen JørgenVilladsen丹麦
技术大学，丹麦孔恩斯灵比 fkjac@dtu.dk jovi@dtu.dk

我们介绍一种方法，用于评估学生在使用 Isabelle 证明助手的自动推理课程中的学习成果。该方法既能评估学生对各种逻辑证明系统中形式化证明的一般理解，也能评估其对 Isabelle/HOL 高阶逻辑中证明的理解。使用 Isabelle 几乎可以自动完成考试中大部分内容的评分。我们通过若干示例题目阐述这一方法，并说明为何我们认为所选的各类题目都能恰当地衡量预期的学习成果。最后，我们讨论在自动推理课程考试中采用这一方法的经验，并提出未来可能的研究方向。

1 引言

我们目前在丹麦技术大学开设一门理学硕士级别的自动推理课程，使用 Isabelle 证明助手 [11] 作为主要工具。该课程为一门 5个欧洲学分转换与累积系统学分的选修课，课程主页如下：

https://courses.compute.dtu.dk/02256/

本课程介绍自动定理证明和交互式定理证明，并使用 Isabelle 将课程中介绍的几乎所有概念形式化。我们开发了若干外部工具，以便讲授自然演绎和序列演算中的基础证明，同时逐步引导学生了解 Isabelle 的全部强大功能。课程的学习目标如下：

1. 解释课程中介绍的基本概念 2. 形式化表达数学定理和 IT 系统的属性 3. 掌握自然演绎证明系统 4. 联系一阶逻辑、高阶逻辑和类型理论 5. 以过程式风格和声明式风格构造形式化证明 6. 使用自动和交互式计算机系统进行自动推理 7. 评估证明助手及相关工具的可信度 8. 清晰、准确地阐述题目的解答

我们希望学生在开始这门课程前，已经掌握一些逻辑知识，并且较为熟练地运用函数式编程。此外，我们还希望学生具备一些用于演绎的人工智能算法基础知识。我们的计算机科学与工程本科课程包含多门课程，介绍这些主题；许多学生都修读过这些课程。然而，我们的许多理学硕士学生来自其他大学或其他

J. Marcos、W. Neuper 和 P. Quaresma（编）：《教育软件的定理证明组件》2022（ThEdu'22），EPTCS 375，2023 年，第 63–76 页，
doi:10.4204/EPTCS.375.6

© F. K.Jacobsen & J. Villadsen 本作品采用知识共享署名许可协议授权。

本科课程体系，因此可能只熟悉我们认为属于先修课程要求的部分主题。在我们的机构，先修课程要求并非强制，而只是建议修读（部分原因是为了避免承担繁重的行政工作，例如判定本校本科课程与其他大学课程之间的等同性）。因此，我们不能指望学生熟悉所有先修课程要求，所以会在整个课程中尽力讲解相关内容。

在课程期间，我们需要检验学生的学习成果。我们要求学生在课程期间提交六份作业，同时在课程结束时安排一场两小时的笔试。本文将说明我们如何设计这场考试，使其既能利用 Isabelle 提供的自动化功能，又能检验学生对广义逻辑以及在 Isabelle 中证明定理的具体方法的理解。我们发现，这种安排很适合我们的课程；考试时课程共有 41 名注册学生。

这是一场两小时的笔试，允许使用所有辅助材料（包括互联网，但学生当然不得在考试期间向他人求助）。考试试卷由一个需要填写的 Isabelle 文件和一个“库”文件组成，后者包含考试所用证明系统的定义。考试包含五道题（分值相同），每道题各有两个小问。五道题可以按任意顺序作答。题目涉及以下五个主题：

1. 不使用自动化的 Isabelle 证明 2. 在 Isabelle/HOL 中验证函数式程序 3. 自然演绎证明 4. 序列演算证明 5. 使用 Isar 在 Isabelle/HOL 中进行一般证明

本文将举例说明我们为每个主题设计的各类问题，并解释为什么我们认为这些问题足以检验各主题的学习成果。每道题的分值相同，但每道题中的两个问题分值可能不同。这样，我们就可以在部分题目中设置一道较容易、一道较难的问题。我们发现，这有助于学生在考试中避免“僵住”，因为他们能先取得一些小小的成功。

本文所列的例题及其解答取自一份模拟考试；这份试卷也会发给学生，让他们在正式考试前熟悉题型（且不会影响成绩）。在我们所在的机构，学生通常可以充分查阅往年的考试题，往往也包括参考答案。由于今年我们对考试形式进行了相当大的调整，因此我们引入了模拟考试，让学生有机会询问如何填写发给他们的 Isabelle 文件，并了解正式考试会考查哪些类型的问题。需要指出的是，本文提到的考试题并未检验上述所有学习目标。课程期间提交的作业会检验其余学习目标，也会纳入最终成绩的评定。

使用 Isabelle 举办考试的主要好处之一，当然是评分大多几乎可以自动完成。由于 Isabelle 能检查学生提交的证明是否正确，考试评分所需的工作仅仅是手动检查一些格式问题，并确认学生没有更改任何定义，以至于他们证明的内容与题目要求不同。我们没有遇到过学生恶意更改定义以简化证明的情况，但有些学生会误解作业要求，写出错误的证明，然后（以明显且未加掩饰的方式）更改定义，让证明能够通过。对于规模较大的课程，完全自动化评分或许更可取，甚至是必要的；但这么做会对我们能够提出的问题类型施加严格得多的限制，因此我们选择不这样做。例如，这将导致我们无法给那些证明思路正确、

但提交的文件存在轻微语法错误（例如，忘了在原本正确的证明末尾加上证毕）的学生部分分数，而这种情况相当常见。Pierce [12] 描述了采用全自动方法所面临的一些挑战。对于规模较小的课程，我们认为口试或基于项目的考试可能更容易安排，也更有益；但就我们课程的规模而言，我们实在没有时间采用这些方式。

我们还开设一门理学学士层次的逻辑系统与逻辑编程课程，为许多学生提供必备的逻辑基础。在这门课程中，学生会简要接触 Isabelle，但仅限于范围有限得多的内容，而且期末考试时不会使用 Isabelle [17]。

在下一节中，我们将选取若干相关工作，对比我们的方法。随后几节将讨论并展示一些考试题及其解答，并说明这些题目考查哪些学习成果。接着，我们将介绍在实践中使用这些考试题的经验，并讨论可能的未来工作，最后作出总结。

2 相关工作

此前，证明助手 Isabelle/HOL 曾用于理学硕士层次的程序语言语义及其应用教学 [8]。这门课程有意避免以逻辑本身为重点，而是将逻辑和 Isabelle 都作为理解语义的工具。相比之下，我们的课程旨在教授逻辑，因此也会讲授 Isabelle 内部工作机制的部分内容。此外，我们的课程并不十分关注应用。另一方面，两门课程也有相似之处：我们同样不以教授证明助手本身为目的，而是将其作为理解和运用逻辑的工具。值得注意的是，语义课程的考试完全不使用 Isabelle。

近来，Isabelle/HOL 也被用于教授算法和数据结构 [10]。这门课程同样不以逻辑本身为重点，而是将逻辑和 Isabelle 作为理解算法与数据结构的工具。不过，仍有必要介绍一些 Isabelle，这部分内容占课程的前三分之一。这门课程也使用 Isabelle 进行考试，但并不侧重于逻辑。

证明助手 Coq 已被用于一系列涵盖基础逻辑、编程语言、函数式算法和分离逻辑的教科书 [13]。这些书也相当注重学习 Coq 证明助手；与 Isabelle 的用户界面相比，Coq 的用户界面与纸笔证明的差异更大。书中包含许多习题，均须在 Coq 中完成；此外还提供可自动评阅大多数习题的脚本。这些习题经过专门设计，可以自动评阅。我们认为，与关于逻辑的证明相比，关于编程语言和算法主题的证明更容易实现自动评阅；这是因为据我们的经验，逻辑证明更难拆分为彼此独立的引理，而且需要更多辅助概念。这些书只涉及基础逻辑，主要是为了展示 Coq 的功能。据我们所知，这些书所用于的课程，其考试并不使用 Coq，不过曾布置过使用 Coq 完成的作业练习 [12]。

证明助手 Lean 已被用于教授程序语言语义和数学基础的基本内容 [2]。这门课程也相当注重学习 Lean 证明助手本身。课程考试采用纸笔形式，但会给学生提供使用 Lean 语法编写的题目，并要求他们用 Lean 语法写出若干答案。考试包括关于某些 Lean 项类型的问题、要求学生用 Lean 编写定义的问题，以及要求学生证明 Lean 中某些定义所具备性质的问题。证明不必是真正的 Lean 证明，但必须非常形式化。

Lean 已经有 因此已被用于教授一般数学 [3, 4]。关于证明助手在一般数学教学中的使用情况，另有一篇较长的综述 of

```
section ‹Problem 2 - 20%›

subsection ‹Question 1›

text ‹

Using only the constructors 0 and Suc and no arithmetical operators, define a recursive function
triple :: ‹nat ⇒ nat› and prove ‹triple n = 3 * n›.

›

fun triple :: ‹nat ⇒ nat› where
  ‹triple 0 = 0› |
  ‹triple (Suc n) = Suc (Suc (Suc (triple n)))›

lemma ‹triple n = 3 * n›
  by (induct n) simp_all
```

图1：问题2——给定的文本及下方的解答。

可参见 [1]。

证明助手 Agda 曾用于我们机构的一门课程中讲授逻辑和函数式编程，但仅用于一个示例，并未用于考试 [7]。

3 Isabelle 证明：不使用自动化

考试的第一道题旨在检验学生是否理解高阶逻辑的证明系统，它是 Isabelle/HOL 的基础。为此，我们要求学生证明一些简单逻辑公式的有效性，例如 $p \leftrightarrow \neg\neg p$ 以及

$$(\exists x.\forall y.r(x,y)) \rightarrow (\forall y.\exists x.r(x,y))$$

学生必须在 Isabelle 中对这些公式的有效性进行形式化证明，因此这道题也检验学生是否掌握了 Isabelle 证明所用的自然演绎证明系统，以及学生是否真正知道如何使用 Isabelle。

为了避免学生只是使用 Isabelle 的 Sledgehammer 等自动化工具来证明公式，我们在 Isabelle/Pure 中重新实现了不含任何自动化的高阶逻辑，并要求学生直接使用该系统中的证明规则。我们最近介绍了这一针对直觉主义和经典命题逻辑的重新实现 [15]。为了这门课程，我们加入了常用量词，从而得到一阶逻辑，最后又重新实现了高阶逻辑 [16]。使用这一系统时，如果学生先写出自己想要证明的子目标，Isabelle 通常仍能建议合适的证明规则，但一次只能建议一个步骤，而且并非所有情况都能做到。

4 Isabelle/HOL 中函数式程序的验证

考试的第二道题主要用于考查学生是否能够在 Isabelle/HOL 中编程和证明，以及是否能够用形式化方式表达信息技术系统的性质。为此，我们要求学生编写非常简单的程序，并证明这些程序的一些简单性质。这些题目类似于《Isabelle/HOL 中的编程与证明》教程 [9] 中的一些较简单的习题。图 1 给出了一道示例题，其中也包含了答案。

设计这类题目时，要让我们要求学生编写和证明的程序及性质达到适当的复杂程度，相当困难。即使是非常简单的性质，要进行形式化证明也可能十分困难，而学生在考试情境的巨大压力下，只有几分钟时间解题。因此，我们将程序和证明都设计得非常简单，目标是让有经验的 Isabelle 用户觉得题目轻而易举，但让从未使用过 Isabelle 的人难以在几分钟内解出。请注意，解答需要使用归纳法，这意味着无法直接使用 Sledgehammer 工具证明该性质。我们的意图是，让真正认真上过这门课程并且知道如何使用 Isabelle 的学生觉得这些题目相对容易，而没有在课程中认真听讲的学生则会觉得这些题目非常难。

虽然考试本身的题目相当简单，但学生在课程期间还会拿到一些包含更复杂问题的作业。主题包括使用关系和集合进行编程、定义归纳类型并在其上定义递归函数，以及通过结构归纳法证明性质，其中包括对归纳定义的关系进行规则归纳。

5 自然演绎与序列演算证明

考试接下来的两道题分别要求在自然演绎系统和序列演算系统中进行证明。这两个系统都涉及带函数的经典一阶逻辑，但比 Isabelle 更容易使用，因为 Isabelle 可能相当复杂。我们设计了两个外部工具，学生可以用它们在这些证明系统中编写形式化证明，随后这些证明会自动转换为 Isabelle 证明。自然演绎系统 NaDeA 采用图形界面，序列演算系统 SeCaV 则采用文本界面 [5,14]。这两种界面都通过 Web 应用程序实现，学生在考试期间可以访问这些应用程序，因为他们能够连接互联网：

https://nadea.compute.dtu.dk/

https://secav.compute.dtu.dk/

这些系统的核心理念是，让学生在尝试证明一个公式有效时，更容易了解自己有哪些选择。在 NaDeA 中，图形界面意味着学生无法输入语法错误的公式。此外，NaDeA 会自动筛选系统中的证明规则，只在图形界面中显示当前情况下实际可用的规则。最后，系统会自动处理证明规则的实际应用，包括跟踪证明树中的分支。

SeCaV 工具与 Isabelle 中的 apply 风格证明更为接近，不过它采用的是序列演算，而非自然演绎。该工具以文本形式接收输入，因此学生在输入公式和证明规则时，也可能像在 Isabelle 中一样犯语法错误。与 NaDeA 的另一个主要区别在于，学生必须手动判断应用某条证明规则会产生什么结果，并自行写出应用该规则后的结果。不过，由于其证明系统比 Isabelle/HOL 的证明系统简单得多，SeCaV 系统能够在学生出错时给出相当详细的警告和错误消息。

我们已在别处更详细地介绍了这些系统在教学和逻辑设计方面的考量 [6]。一个关键点是，我们为 NaDeA 和 SeCaV 都形式化了语法、语义和证明系统。我们还为这些证明系统建立了形式化的可靠性和完备性定理。这些形式化内容是课程教学材料的一部分。在考试中，我们要求学生证明命题公式和一阶公式，并涉及经典公式，使得

学生需要理解如何在经典逻辑中进行推理。因此，这些题目考查学生是否掌握了自然演绎，是否理解何时需要使用经典证明规则以及如何使用它们，以及能否借助交互式计算机系统构造形式化证明。NaDeA 和 SeCaV 都在每位学生各自的浏览器中运行，因此任意数量的学生都可以同时使用这些工具，而不会遇到性能问题。

6 使用 Isar 编写 Isabelle/HOL 中的一般证明

考试的最后一道题旨在检验学生对 Isabelle/HOL 的 Isar 语言中结构化证明的理解，也就是学生是否能够以声明式风格构造形式化证明。该题要求学生完成一项较复杂性质的证明，例如图 2a 中的例子。图 2a 中给出的问题包含一个引理，以及一段对该引理的“证明”评注，其中有几处错误。学生必须修正这些错误，从而完成证明。这类题目的思路是避免设计复杂度适当的性质这一难题；第 4 节也提到了这一点。给学生一份部分完成的证明，并要求他们修正其中的错误，我们便可以要求学生证明一些性质，而他们若是在考试情境的有限时间内，原本无法证明这些性质。这样，我们就能检验学生对非平凡证明的理解，而不至于沦为单纯测试学生是否会使用 Isabelle/HOL 的自动化功能。

我们将通过演示如何解答模拟考试中的一道题目，来展示这一概念。图 2b 中，我们取消了对部分完成的证明的注释，这使 Isabelle/jEdit 开发环境能够突出显示证明中的错误。请注意，证明中的若干关键字带有红色下划线，某些变量则有棕色背景。将光标移到带下划线的关键字上，或查看 Isabelle/jEdit 中单独的“输出”窗格，都可以获得关于错误的更详细信息。需要指出的是，虽然 Isabelle 能够突出显示错误，并且通常会提供一些关于错误原因的信息，但 Isabelle 无法说明如何修正错误。从上往下看，我们注意到的第一个错误是这一行 **show** ⟨$p\ x$⟩ 包含了变量 x，但该变量尚未定义。实际上，给定的证明在前一行引入了一个局部常量 a，因此，只需将所有高亮显示的x 改为a，即可修正证明，如图 2c 所示。接下来，我们来看这个子证明，它试图证明刚才修正的目标。在图 2c 中，我们发现关键字 **show** 和**assume** 的位置颠倒了，因此这个错误的证明先尝试证明假设，最后才假设结论。只需交换这两个关键字即可修正，如图 2d 所示。接下来，在子证明中用于证明 $\bot$ 的公式 $\neg\,(\exists x.\ \neg\, p\ x)$，在证明进行到此处时并不是一个可用的假设。要修正这个错误，需要比前面几个错误更深入地了解 Isabelle/HOL，因为问题实际上不在这一行，而是在证明开头的地方：那里作出了一个错误的假设。证明以经典反证法的证明规则开头，但第一个假设假定的是原公式，而不是它的否定。像图 3a 那样修正这一点后，证明后续部分就能使用所需的公式作为假设。证明中的最后一个错误也与反证法有关，因为证明的最后一行试图证明 $\top$，而反证法的正确结束方式是证明 $\bot$。修正后得到图 3b，Isabelle 验证其为正确的证明。

虽然这个引理及其证明都相当简单，但考试情境中的时间限制意味着学生需要非常迅速地找出并修正错误。在课程期间，学生需要在习题中以及作为指定问题，从头证明更复杂的引理；这些问题会纳入最终成绩的评定。对于这些引理，学生有充足得多的时间，也可以向助教求助，由他们帮助学生走上正确的轨道。在考试情境中，我们希望

```
section ‹Problem 5 - 20%›

subsection ‹Question 1›

text ‹

Replace \<proof> with the "proof ... qed" lines
in the following comment and correct the errors
such that the structured proof is a proper proof
in Isabelle/HOL (do not alter the lemma text).

›

lemma Foobar:
  assumes ‹¬ (∀x. p x)›
  shows ‹∃x. ¬ p x›
  \<proof>

(*

proof (rule ccontr)
  assume ‹∃x. ¬ p x›
  have ‹∀x. p x›
  proof
    fix a
    show ‹p x›
    proof (rule ccontr)
      show ‹¬ p x›
      then have ‹∃x. ¬ p x› ..
      with ‹¬ (∃x. ¬ p x)› assume ⊥ ..
    qed
  qed
  with ‹¬ (∀x. p x)› show ⊤ ..
qed
```

(a) 问题 5

```
lemma Foobar:
  assumes ‹¬ (∀x. p x)›
  shows ‹∃x. ¬ p x›
proof (rule ccontr)
  assume ‹∃x. ¬ p x›
  have ‹∀x. p x›
  proof
    fix a
    show ‹p x›
    proof (rule ccontr)
      show ‹¬ p x›
      then have ‹∃x. ¬ p x› ..
      with ‹¬ (∃x. ¬ p x)› assume ⊥ ..
    qed
  qed
  with ‹¬ (∀x. p x)› show ⊤ ..
qed
```

(b) 问题 5 — 步骤 1

```
lemma Foobar:
  assumes ‹¬ (∀x. p x)›
  shows ‹∃x. ¬ p x›
proof (rule ccontr)
  assume ‹∃x. ¬ p x›
  have ‹∀x. p x›
  proof
    fix a
    show ‹p a›
    proof (rule ccontr)
      show ‹¬ p a›
      then have ‹∃x. ¬ p x› ..
      with ‹¬ (∃x. ¬ p x)› assume ⊥ ..
    qed
  qed
  with ‹¬ (∀x. p x)› show ⊤ ..
qed
```

(c) 问题 5 — 步骤 2

```
lemma Foobar:
  assumes ‹¬ (∀x. p x)›
  shows ‹∃x. ¬ p x›
proof (rule ccontr)
  assume ‹∃x. ¬ p x›
  have ‹∀x. p x›
  proof
    fix a
    show ‹p a›
    proof (rule ccontr)
      assume ‹¬ p a›
      then have ‹∃x. ¬ p x› ..
      with ‹¬ (∃x. ¬ p x)› show ⊥ ..
    qed
  qed
  with ‹¬ (∀x. p x)› show ⊤ ..
qed
```

(d) 问题 5 — 步骤 3

图 2：问题 5 — 给定文本和步骤 1–3。

```
lemma Foobar:
  assumes ‹¬ (∀x. p x)›
  shows ‹∃x. ¬ p x›
proof (rule ccontr)
  assume ‹¬ (∃x. ¬ p x)›
  have ‹∀x. p x›
  proof
    fix a
    show ‹p a›
    proof (rule ccontr)
      assume ‹¬ p a›
      then have ‹∃x. ¬ p x› ..
      with ‹¬ (∃x. ¬ p x)› show ⊥ ..
    qed
  qed
  with ‹¬ (∀x. p x)› show ⊤ ..
qed
```

(a) 问题 5 — 第 4 步

```
lemma Foobar:
  assumes ‹¬ (∀x. p x)›
  shows ‹∃x. ¬ p x›
proof (rule ccontr)
  assume ‹¬ (∃x. ¬ p x)›
  have ‹∀x. p x›
  proof
    fix a
    show ‹p a›
    proof (rule ccontr)
      assume ‹¬ p a›
      then have ‹∃x. ¬ p x› ..
      with ‹¬ (∃x. ¬ p x)› show ⊥ ..
    qed
  qed
  with ‹¬ (∀x. p x)› show ⊥ ..
qed
```

(b) 问题 5 — 第 5 步 / 解答

图 3：问题 5 — 第 4–5 步 / 解答。

这样可以避免出现解题需要“想到正确思路”的情况，因为学生没有太多时间尝试不同的方法。我们发现，为学生提供一份以部分证明形式呈现的提纲，有助于避免这一问题。当然，也可以给学生更多考试时间，但这会大幅增加设计考试题和评分的工作量，而且在繁忙的考试期间，也会更难找到适合考试的场地和时段。

7 实践经验

我们在五月最近一次考试中使用了课程的考试题目。报名参加考试的41名学生中，有36人通过了课程。其余五名学生没有参加考试，另有两名学生在截止日期前取消了考试报名。

课程成绩根据考试成绩以及课程期间提交的作业综合评定。作业在课程期间陆续批改，学生每提交一次作业后都会收到反馈。最终成绩是根据作业和考试对整体表现进行评估后给出的。作业成绩与考试成绩之间有较好的相关性；将综合成绩与根据学生作业表现提出的初步成绩进行比较时，没有学生的成绩变动超过一个等级。

这门课程的平均成绩为 9.9 分（满分 12 分）（采用丹麦国家评分体系，其中 12 分相当于欧洲学分转换与累积系统 A 等，10 分相当于欧洲学分转换与累积系统 B 等）。这个平均分看起来可能很高，但需要注意的是，统计中只包括了实际参加考试的学生。丹麦学生如果觉得自己尚未做好应考准备，通常会选择缺考，因为之后可以再参加考试，而不必因准备不足而“冒着”以较低最终成绩通过考试的风险。只要学生在 3 次机会内通过考试，这种缺考方式就不会影响其成绩；但通过考试的学生不得

再次参加考试。如果将缺考学生按最低可能成绩计入，平均分则为 7.8 分（略高于欧洲学分转换与累积系统 C 等），这符合丹麦评分标准所设定的预期平均水平。平均成绩相对较高的另一个可能原因，当然是这门课程属于高阶非必修课程，因此学生会自我筛选，留下对课程主题有动力且感兴趣的人。在我们的机构，学生可以先旁听课程数周，再决定是否正式选修；事实上，最初的 69 名学生中，略多于三分之一在退课截止日期前退选了课程。我们推测，这些学生中有许多人参加考试时的表现会不如留下来的学生。

根据考试结果和相对表现调整成绩，以获得特定的成绩分布（所谓的“曲线评分”）在丹麦是违法的，因此不可采用。相反，教师应在多个学期中调整课程难度，以实现预期的成绩分布。我们会考虑在下次开课时稍微提高课程难度，使平均成绩有所降低。

学生在考试期间理解如何使用 Isabelle 或如何提交解答方面没有遇到任何问题。提交的解答也普遍表明，学生明白每道题应该如何作答，尽管他们当然并非总能给出正确解答。许多学生提交的 Isabelle 文件存在语法错误，但这些错误通常很容易修复，主要是因为未将放弃的证明尝试标记为此类。

我们发现，考试答卷总体上很容易评分。在大多数情况下，只要确认学生采用了合理的方法，并且 Isabelle 没有在其证明中发现错误，几乎就能自动检查出题目的正确解答。部分解答则较难评分，因为很难估计学生距离完成证明究竟还有多远。不过，这种难度并不比用纸笔进行证明时更大；而且，Isabelle 同样能够通过突出显示证明中的错误和缺漏来提供帮助。

由于学生在考试期间可以使用所有辅助资源（包括互联网），因此我们应提及作弊的可能性。考试由多名监考人员监考，以发现作弊行为，且考场内不允许使用手机。如果学生通过相互交流作弊，我们预计会看到相同的解答，但没有发现这方面的任何异常。给学生提供略有不同的题目或许可以减少作弊，但我们认为很难确保这些题目的难度相同。我们开发了辅助工具的版本，可供下载并在浏览器中本地（离线）运行；在今后的考试中，我们考虑不允许学生访问互联网。

本节接下来将介绍学生对考试的评价，然后更详细地讨论学生在尝试解答考试中各类题目时的表现和遇到的问题。

7.1 学生评价

我们机构的所有学生都被要求对自己参加的考试进行匿名评价。该课程考试的评价表面向43名学生开放（包括两名在截止日期前取消考试报名的学生）。共有17名学生填写了评价表。在评价该课程的学生中，15人在评价时尚未收到成绩，另有一名学生已收到成绩。一名学生在评价表中表示自己没有参加该课程。

第一个问题评估中的一个问题询问考试在形式、内容和复杂程度方面是否与教学活动相符 in

课程。7名学生表示完全同意，5名表示同意，3名表示中立，1名表示不同意。因此，学生平均认为考试与课程中的其他活动相符。几名学生还写下了补充意见，表示对需要使用经典证明规则的证明感到困惑。一名学生希望增加一些“中等难度”的题目。

评估中的下一个问题询问考试的形式和内容是否符合课程的学习目标。6 名学生回答说他们完全同意，5 名表示同意，4 名表示中立，1 名表示不同意。因此，学生平均认为考试符合课程的学习目标。

评估的最后一部分征求进一步的意见和建议。几名学生写道，考试内容符合他们的预期，而且模拟考试在这方面很有帮助。一些学生建议取消考试，改为通过基于项目的评估来评价他们的学习情况。一名学生表示，自己不知道如何正确中止证明尝试，以便继续回答下一道题，同时避免在 Isabelle 中出现错误。我们将在关于未来工作的章节中再讨论这些建议。

7.2 不使用自动化的 Isabelle 证明

考试的第一道题似乎是最难的题目之一。如前所述，这道题分为两个小题，其中一个明显比另一个难。较简单的小题旨在让学生先“轻松获胜”，再开始处理其余题目。

大多数学生都完成了较简单的题目，但完成两道题的学生寥寥无几。约有一半学生对较难的题目进行了合理的尝试，尝试证明，但仍缺少重要部分。几名学生对于较难的题目完全没有作答。

这基本符合预期，因为这类题目需要创造力，也需要“想到正确思路”才能解出。较难的题目要求运用经典证明规则，而根据我们的经验，这些规则通常会让学生感到困惑。

7.3 Isabelle/HOL 中函数式程序的验证

大多数学生都完整解答了考试的第二道题。几名学生只完整解答了其中一道题，只有少数学生两道题都没有解答。这或许表明，我们对这道题的难度设定有些过于谨慎。在整个课程中，我们从作业里布置的类似题目中发现，很难预测学生会觉得程序验证题目难还是容易。因此，我们决定宁可把题目出得太简单一些。

7.4 自然演绎证明

大约一半的学生完整地解答了考试的第三道题。这道题又分为较简单和较难的两个小问，所有学生都解答了较简单的小问。有几名学生对较难的小问进行了合理的证明尝试，但仍有重要部分缺失。

这里的情况与第一题类似，难度较高的问题同样需要使用经典证明规则。这再次表明，学生觉得经典证明规则比较难用。

7.5 序列演算证明

所有学生都完整解答了考试的第四题。这或许有些令人意外，因为这道题中的一个问题也需要使用经典证明规则。然而，在 SeCaV 的序列演算系统中，学生无需像在 NaDeA 和 Isabelle/HOL 的自然演绎系统中那样，明确选择在证明的哪个位置应用经典证明规则。因此，在使用 SeCaV 时，涉及经典证明规则的证明不一定比不涉及这些规则的证明更难。基于这一点，我们可以考虑在未来的考试中提高这类题目的难度。

7.6 使用 Isar 在 Isabelle/HOL 中的一般证明

考试的第五道也是最后一道题，大约有一半的学生完整地解答了。大多数学生完整地解答了其中一道题，并部分解答了另一道题。只有少数学生一道题也没解出来，而其中几乎所有人都提交了部分解答。

其中一道题比另一道难，因此这在意料之中。这道题也最容易获得部分解答，因为题目要求学生修正若干个或多或少彼此独立的错误。

8 未来工作

进一步研究有许多潜在方向。如上所述，我们会根据每年的经验，持续调整考试的难度。这必然也包括为每场考试设计新题目。这带来了一个问题：根据我们的经验，要找到难度恰当的题目并不容易。因此，一个有意思的潜在研究方向是制定创建题目的指南和评估标准。我们预计这并非易事，因为不同学生可能觉得不同题目难度各异，而且通常很难预测一道验证问题有多难。要设计出好的指南，就有必要对更大规模的学生群体和问题类别开展严谨研究。一种可能的做法是，跟踪学生在解答特定类型题目时的行为，找出他们遇到的问题，再制定标准，在评估题目难度时考虑最常见的错误和误解。我们预计，需要开发新工具来支持这类行为研究。

另一个相关的潜在研究方向是开发更多辅助工具，以控制题目的潜在难度。如上文所述，我们已经创建了若干可与 Isabelle 交互的工具，但这些工具会限制学生可选的操作，从而更容易估计我们所设计题目的难度。让学生补完已有的部分证明，也是限制学生误解题目的方式之一。我们预计，各个机构可能已经在使用许多其他方法，因此鼓励大家分享自己的想法和经验。此外，我们预计，其他领域在处理类似题目时积累的经验，例如几何或图论，也可能很有价值；反过来，我们的经验也可能有助于开发可用于这些领域的辅助工具。学生似乎特别难以解答需要运用形式化经典推理的题目，因此，对这类题目的难度加以控制尤其值得探讨。

一种截然不同的选择是完全取消笔试。我们认为，不进行笔试而评估学习成果主要有两种方式：一是根据课程期间提交的作业进行口试，

二是在课程结束时提交一份报告，完成一个规模较大的项目。口试适用于规模较小的课程，但随着课程规模扩大，口试就难以实施了。就我们的课程规模而言，我们认为，若不投入过多时间进行考试，就无法实施口试。另一方面，我们认为，完成一个旨在证明若干定理的大型项目，其评分所需时间不会比笔试多太多。不过，这类项目需要精心设计，让学生能够循序渐进地完成，避免不堪重负；我们预计，设计项目作业会需要投入大量工作。此外，与考试题目相比，这类项目的难度可能也更难估算。

另一项潜在的未来工作，是开发能够完全或几乎完全自动评分的题目。如上文所述，借助我们现有的考试题型，Isabelle 已经让评分变得相当容易。不过，对于规模非常大的课程，完全自动评分或许是必要的。当然，我们可以要求学生提交的文件不含语法错误（这样 Isabelle 的批量检查工具才能处理这些文件），但我们认为，这种限制可能过于严苛。根据我们的经验，许多学生很难区分错误与正确中止的证明尝试，因此提交的文件中往往包含大量错误。一个可行的方向是开发一种修复含有错误的 Isabelle 理论的工具，既可以自动修复，也可以针对文件中发现的错误提出修复建议。我们预计，要开发出实用的此类工具并非易事。

无论如何，即使我们确信所有学生提交都能由 Isabelle 批量检查工具处理，要实现完全自动评分，仍需谨慎设计题目，确保其中包含合理的阶段性目标，以便逐项评分。我们或许可以借鉴 Software Foundations 教材的自动评分脚本；这些脚本包含使用 Coq 证明助手 [12,13] 编写的习题。不过，我们预计，要为自然演绎证明和序列演算证明相关题目设计出包含合理阶段性目标的问题，需要重新设计我们的工具，使其支持在证明过程中途停止。目前，对于风格极差的证明，我们也会扣除少量分数；具体而言，明显过长或复杂程度远超必要的证明无法获得满分。我们预计，自动作出此类判断会很困难，但或许可以开发一种工具，自动为合理的证明评分，并将“有问题”的提交标记出来，交由人工审查。

9 结论

在使用 Isabelle 开设自动推理课程时，我们需要检验学习成果，因此选择通过考试来进行。我们介绍了为此采取的方法，并说明 Isabelle 如何帮助我们快速评阅考试答卷。我们还介绍了考试中各种类型的问题，以及这些问题如何通过一系列示例，检验学生对一般逻辑以及尤其是 Isabelle/HOL 形式化证明的不同方面的理解。

学生通常似乎理解如何解决这些题目，并且对这种考试形式持积极态度。采用这种方法设计考试的主要挑战，是恰当地平衡难度：既要避免题目过于简单，又要确保所有学生都能在每道题上至少取得一些进展。我们说明了如何将简单的编程问题与要求补全较长证明的题目结合起来，以便在多个层面考查学生的理解。其他课程可能有不同的设计限制，尤其是学生人数众多的课程，此时全自动评分可能更受青睐，甚至不可或缺。

我们认为，本文所述的经验和技术，也可能适用于其他领域的考试，尤其是那些面临类似限制、希望或需要全部或部分实现自动评分的领域。这些领域可能包括几何学、图论，以及其他任何推理起重要作用、学生容易误解题目或证明的领域。我们预计，要让没有证明助手使用经验的学生能够接触这些领域中的形式化推理，可能需要采用类似 NaDeA 和 SeCaV 的专用工具。

致谢

我们感谢 Asta Halkjær From、Deniz Sarikaya、Frederik Lyhne Andersen、Simon Tobias Lund 以及匿名审稿人对稿件草稿提出的宝贵意见。

参考文献

[1] Jeremy Avigad 和 John Harrison（2014）：形式化验证数学。《style id='4'>ACM 通讯 57(4)，第 66–75 页，doi:10.1145/2591012。[2]Jasmin Christian Blanchette（2021）：逻辑验证。https://lean-forward.github.io/ logical-verification/2021/。课程简介。[3] Kevin Buzzard 和 Mohammad Pedramfar（2021）：自然数游戏，*1.3.3* 版。https://www. ma.imperial.ac.uk/~ buzzard/xena/natural_number_game/。教育游戏。[4] Edmund M. Clarke 和 Jeremy Avigad（2015）：交互式定理证明。http://lean.github. io/cmu-15815-s15/。课程简介。[5] Asta Halkjær From、Frederik Krogsdal Jacobsen 和 Jørgen Villadsen（2022）：*SeCaV*：*Isabelle/HOL* 中的序列演算验证器。载于 Mauricio Ayala-Rincon 和 Eduardo Bonelli（编）：第 16 届逻辑与语义框架及其应用会议论文集，布宜诺斯艾利斯，阿根廷（线上），2021年7月23日至24日，《理论计算机科学电子论文集》357，开放出版协会，第 38–55 页，doi:10.4204/EPTCS.357.4。[6] Asta Halkjær From、Jørgen Villadsen 和 Patrick Blackburn（2020）：将 *Isabelle/HOL* 作为教授逻辑的元语言。载于 Pedro Quaresma、Walther Neuper 和 João Marcos（编）：第九届教育软件定理证明组件国际研讨会论文集，*ThEdu@IJCAR 2020*，法国巴黎，*2020*年*6*月*29*日，《理论计算机科学电子论文集》328，开放出版协会，第 18–34 页，doi:10.4204/EPTCS.328.2。[7] Asta Halkjær From 和 Jørgen Villadsen（2021）：使用 *Agda* 和 *Isabelle/HOL* 教授自动推理与形式化验证的函数式编程。载于：第十届函数式编程教育趋势国际研讨会（*TFPIE 2021*）——仅报告／线上论文，第 1–20 页。参见 https://wiki.tfpie.science.ru.nl/TFPIE2021。[8] Tobias Nipkow（2012）：使用证明助手教授语义学：不再有 *LSD* 迷幻之旅式证明。载于 Viktor Kuncak 和 Andrey Rybalchenko（编）：验证、模型检测与抽象解释——第十三届国际会议 *VMCAI 2012*，费城，美国宾夕法尼亚州，*2012*年*1*月*22*日至*24*日。论文集，《计算机科学讲义》7148，施普林格，第 24–38 页，doi:10.1007/978-3-642-27940-9_3。[9] Tobias Nipkow（2021）：*Isabelle/HOL* 中的编程与证明（教程）。https://isabelle.in. tum.de/doc/prog-prove.pdf。[10] Tobias Nipkow（2021）：使用证明助手教授算法与数据结构（特邀报告）。载于 Catalin Hritcu 和 Andrei Popescu（编）：*CPP '21*：第十届 *ACM SIGPLAN* 认证程序与证明国际会议，线上活动，丹麦，*2021*年*1*月*17*日至*19*日，ACM，第 1–3 页，doi:10.1145/3437992.3439910。

[11] Tobias Nipkow、Lawrence C. Paulson 和 Markus Wenzel（2002）：*Isabelle/HOL*——高阶逻辑证明助手。《计算机科学讲义》2283，施普林格，doi:10.1007/3-540-45949-9。[12] Benjamin C. Pierce（2009）：*Lambda*，终极助教：使用证明助手教授编程语言基础。载于：第十四届 *ACM SIGPLAN* 函数式编程国际会议论文集，ICFP '09，计算机协会，纽约，美国纽约州，第 121–122 页，doi:10.1145/1596550.1596552。[13] Benjamin C. Pierce、Arthur Azevedo de Amorim、Chris Casinghino、Marco Gaboardi、Michael Greenberg、C˘at˘alin Hri¸tcu、Vilhelm Sjöberg 和 Brent Yorgey（2022）：逻辑基础。*SoftwareFoundations* 1，电子教材。6.2 版，http://softwarefoundations.cis.upenn.edu。[14] Jørgen Villadsen、Andreas Halkjær From 和 Anders Schlichtkrull（2018）：自然演绎助手（*NaDeA*）。载于 Pedro Quaresma 和 Walther Neuper（编）：第七届教育软件定理证明组件国际研讨会论文集，*THedu@FLoC 2018*，英国牛津，*2018*年*7*月*18*日，《理论计算机科学电子论文集》290，开放出版协会，第 14–29 页，doi:10.4204/EPTCS.290.2。[15] Jørgen Villadsen、Asta Halkjær From 和 Patrick Blackburn（2022）：使用 *Isabelle* 教授直觉主义与经典命题逻辑。载于 João Marcos、Walther Neuper 和 Pedro Quaresma（编）：第十届教育软件定理证明组件国际研讨会论文集，美国卡内基梅隆大学（远程），美国宾夕法尼亚州匹兹堡，2021年7月11日，《理论计算机科学电子论文集》354，开放出版协会，第 71–85 页，doi:10.4204/EPTCS.354.6。[16] Jørgen Villadsen（2022）：*Isabelle/Pure* 中经典高阶逻辑的一种表述。《逻辑与人工智能杂志》（*LAI*）。已录用。[17] Jørgen Villadsen 和 Frederik Krogsdal Jacobsen（2021）：在两门逻辑与自动推理课程中使用 *Isabelle*。载于 João F. Ferreira、Alexandra Mendes 和 Claudio Menghi（编）：形式化方法教学，施普林格国际出版，查姆，第 117–132 页，doi:10.1007/978-3-030-91550-6_9。[18] Makarius Wenzel（2021）：*Isabelle/Isar* 参考手册。https://isabelle.in.tum.de/doc/ isar-ref.pdf。