Operationalizing the Formalizability of Mathematics Problems for Intelligent Tutoring Systems: Taxonomy, Measurement Protocol, and Educational Impact*

Oleksandr Yevdokymov[1,†], Andriy Chukhray[1,*,†] and Tetiana Stoliarenko[2,†]

[1] National aerospace university «Kharkiv Aviation Institute», Vadyma Manka St. 17, 61070 Kharkiv, Ukraine
[2] Simon Kuznets Kharkiv National University of Economics, Nauky Avenue, 9A, 61166 Kharkiv, Ukraine

Abstract

Problem formalizability is examined under explicitly defined constraints of time and interaction budgets. A five-level taxonomy (F0–F4) is introduced to characterize different degrees of formalizability, together with a replicable protocol for estimating the distribution of these levels within a given course. To support practical applicability in intelligent tutoring systems (ITS), a systematic mapping is established between learner-facing mathematical interfaces and the programmatic APIs of state-of-the-art proof assistants (Lean, Coq/Rocq, Isabelle) as well as external Automated Theorem Proving (ATP) and Satisfiability Modulo Theories (SMT) back-ends. Worked mathematical examples are provided to demonstrate how this mapping can be realized in practice. The study references current manuals and standards (Isabelle2025 & Sledgehammer, Coq/Rocq 8.19, Mathematics in Lean 2025, SMT-LIB 2.7, Z3, and cvc5 documentation) along with relevant benchmarks (miniF2F, LeanDojo, MATH) [1]–[6], [8]–[16]. To demonstrate feasibility, we prepared an illustrative pilot on a curated set of undergraduate-level problems drawn from typical algebra/calculus exercises. This pilot was not deployed in a live course, and its figures are demonstrative rather than course-level estimates. We supply a small reproducibility package (data, solver configurations, and proof-assistant scripts) to enable future replications and a planned in-situ study

Keywords

intelligent tutoring systems; formal proofs; automated theorem proving; Lean; Coq/Rocq; Isabelle; Sledgehammer; TPTP; SMT-LIB 2.7; Z3; cvc5; ε–δ analysis; ODEs.

1. Introduction

Modern proof assistants, such as Isabelle/HOL, Coq (Rocq), and Lean, in combination with Automated Theorem Proving (ATP) and Satisfiability Modulo Theories (SMT) back-ends, are now capable of verifying extensive areas of mathematics, supported by steadily expanding libraries and advanced automation techniques [1]–[5]. Nevertheless, a systematic planning instrument is still lacking for educators to determine what proportion of a course's problems can be readily formalized and proved with modest instructional support, and which problems remain resistant under comparable constraints. This paper proposes a resource-sensitive definition of *formalizability*, introduces a five-level taxonomy (F0–F4), and establishes a replicable measurement protocol. In addition, a structured mapping is presented from learner-facing mathematical interfaces to the application programming interfaces (APIs) of Isabelle/Sledgehammer, Lean, and Rocq/Coq, together with standardized bridges to external ATP (TPTP) and SMT-LIB back-ends (current version 2.7, released February 5, 2025) [2], [6], [11].

*ProfIT AI'25: 5th International Workshop of IT-professionals on Artificial Intelligence, October 15–17, 2025, Liverpool, UK
* You should use this document as the template for preparing your publication. We recommend using the latest version of the CEURART style.
1* Corresponding author.
† These authors contributed equally.
o.yevdokymov@khai.edu (O. Yevdokymov); achukhray@gmail.com (A. Chukhray); t_stol@ukr.net (T. Stoliarenko)
0009-0008-9687-6344 (O. Yevdokymov); 0000-0002-8075-3664 (A. Chukhray); 0000-0002-7202-316X (T. Stoliarenko)

2. Related Work: Educational Uses of Proof Assistants and ATP/SMT

Lean in mathematics education. A growing body of work analyzes Lean as a vehicle for teaching proof. Thoma & Iannone report an exploratory study with first-year undergraduates, observing positive effects on students' ability to construct proofs—even on paper—after working with Lean [17]. Massot details didactic goals and classroom experience, including controlled natural-language and "verbose" modes for beginners [18]. A pilot by Bottoni in a "Foundations of Mathematics" course evaluates Lean's impact on understanding and on organizing practical sessions [19]. Beyond formal studies, there is a wide practice of university courses and workshops that integrate Lean into undergraduate math curricula (e.g., "Courses using Lean" and the "Learning Mathematics with Lean" event series) [20], [21]. Collectively, these sources underscore Lean's strengths for instant feedback, graded hints, and scaffolded library use, while leaving open a course-level metric of what fraction of typical problems can be quickly formalized.

Coq (*Rocq*) in logic/programming courses. The long-running textbook series Software Foundations has served as a "gentle on-ramp" to Coq for logic, semantics, and algorithm verification; a "15 years on" retrospective summarizes pedagogical principles and experience at scale [22], [23]. Earlier case studies document teaching logic and formal methods with Coq in classroom settings [24]. These works highlight the effectiveness of tactical learning (induction, equality rewriting, lia/ring automation) and stepwise feedback, but they do not provide *resource-oriented* per-course estimates of problem shares.

Isabelle/HOL in formal methods — and even examinations. Several "from the classroom" reports exist. *Villadsen et al.* describe Isabelle in two courses on logic and automated reasoning and share organizational practices [25]. *Jacobsen* shows how exams in automated reasoning can be built in Isabelle so that a large portion of grading is semi-automated [26]. Updated tutorials (e.g., *Nipkow's Programming and Proving in Isabelle/HOL* and the Isabelle tutorial) serve as pedagogical foundations, especially for induction and equivalence transformations typically used near the start of a course [27], [28]. In practice, automation (notably Sledgehammer) reduces manual burden; however, none of these works quantify a course in terms of "% of problems F0/F1…" under fixed time/hint budgets.

Where SMT and ATP fit pedagogically. In deductive verification education, Why3-based courses show how to structure topics (loop invariants, ghost code, specifications) with industrial SMT back-ends [29]. In the ACL2 community, pedagogic IDEs such as DRACULA and Proof Pad ease students' first steps [30], [31]. For auto-checking programming tasks, SMT-based systems (e.g., AutoRubric) reduce student code to solver formulas for equivalence checking against references [32]. There are also student-oriented guides to SMT modeling and solver use [33]. The conclusion across these lines is consistent: SMT/ATP cover QF_LIA/NRA and equality problems well (often F0–F1 in similar to proposed scale), yet their role in a comprehensive course profile is usually described narratively—without systematic statistics.

Links to AI-assisted proving and benchmarks. For hint selection and advanced challenge sets, two modern corpora are particularly relevant: LeanDojo (programmatic access to Lean proof states with retrieval augmentation) and miniF2F (formalized Olympiad-level problems) [34], [35]. These resources focus on proof difficulty and lemma retrieval, but do not offer course-oriented estimates of "what fraction of real curricular problems can be quickly formalized." That precise gap is addressed by this paper's methodology (UI→API mapping in §3 and measurement protocol in §6) [1]–[16].

A complementary perspective is provided by A. Chukhray et al., where proposed that formal verification systems (i.e., interactive proof assistants and automated theorem provers) could be integrated into Intelligent Tutoring Systems for higher mathematics courses in order to enable the checking of complex problems, not just routine exercises [36]. This vision directly motivates our resource-aware taxonomy, since it raises the practical question of which types of problems are formally checkable within reasonable budgets and how such capabilities can be embedded into adaptive ITS workflows.

3. Interface mapping (learner math UI → prover APIs → ATP/SMT)

3.1. Conceptual layers

- (A) Learner UI (math layer): typed expressions, goals, and actions (Simplify, Rewrite, Prove, Search lemma, Solve inequalities).
- (B) Logical AST: typed terms, binders $(\forall, \exists)$, connectives, algebraic structures (semirings, rings, fields), vectors/matrices [7].
- (C) Prover interface: tactics + libraries in Lean (mathlib), Rocq/Coq, Isabelle/HOL [1], [3]–[5].
- (D) External automation: Sledgehammer bridges Isabelle to ATP/SMT; direct SMT-LIB calls target Z3/cvc5 for fragments like QF_LIA/NRA; TPTP is the de-facto ATP format [2], [6], [8] ,[9], [11]–[13].

3.2. Data model and translation

Let the UI grammar of terms be

$$t ::= x \mid c \mid f(t) \mid t \oplus t \mid (t, t) \mid \{t \mid \phi(t)\} \tag{1}$$

and formulas

$$\phi ::= t = t \mid t \leq t \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \forall\, x.\phi \mid \exists\, x.\phi. \tag{2}$$

A compiler Φ maps UI inputs to a typed AST [7] of the target system (HOL with type classes/locales). Constraints ϕ are normalized to solver fragments (e.g., QF_LIA/NRA) when possible; otherwise they remain interactive subgoals. The ITP kernel checks proof terms; external tools only propose steps [1], [6], [8], [9].

3.3. Mapping tables

Table 1
Object-level mapping from math UI to prover APIs and solver logics.

Math UI element	Lean (mathlib)	Coq / Rocq	Isabelle/HOL	ATP/SMT logic
$\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$	`Nat, Int, Rat, Real, Complex`	`nat, Z, Q, R, C`	`nat, int, rat, real, complex`	QF_LIA / (QF_)NRA
Sets, $x \in S, S \subseteq T$	`Set α, x ∈ s, s ⊆ t`	stdlib/ Ensembles	sets-as-predicates/locales	FOL fragments
Algebraic structures	`Semiring, Ring, Field, Module`	ring/field libs	locales/type classes	—
Quantifiers/ connectives	`forall, Exists, ∧, ∨, →`	forall, exists, /\ , \\/, ->	∀, ∃, ∧, ∨, ⟶	FOL fragments
Vectors/Matrices	`Matrix, Fin n`	mathcomp/ stdlib	HOL-Algebra / analysis	linear (partial)

Notes: Lean *mathlib* and *Mathematics in Lean* expose consistent APIs; Rocq/Coq provides `ring/field/lia`; Isabelle organizes algebra via locales and `algebra_simps` [1], [3]–[5].

Table 2
Action-level mapping (UI verb ⟶ tactics / solver calls).

UI action	Lean (mathlib)	Coq / Rocq	Isabelle/HOL	External
Simplify	`simp, simp_all`	`simpl, cbn`	`simp`	—
Rewrite by lemma	`rewrite [lemma]`	`rewrite lemma`	`simp add: lemma`	—
Prove polynomial identity	`ring`	`ring/field`	`algebra_simps`	sometimes SMT
Solve linear constraints	`linarith`	`lia`	`linarith/ arith`	SMT QF_LIA in Z3/cvc5
Search for lemma	`apply?, library_search`	`Search, eauto`	`Sledgehammer`	ATP (E, Vampire) via TPTP
Nonlinear real arith.	`nlinarith` (partial)	`nra, psatz`	`nlinarith` (partial)	SMT QF_NRA (heuristic)

On Isabelle, Sledgehammer bridges to ATP/SMT; SMT-LIB 2.7 is the latest spec (Feb 2025); Z3/cvc5 provide proof/trace documentation [2], [6]–[9], [12], [13].

3.4. Benchmarks

For challenge pools and evaluation in tutors and research: miniF2F (formal Olympiad-level tasks), LeanDojo (programmatic access to Lean proof states), and MATH (text-first, step-by-step competition problems) [14]–[16].

3.5. Correctness by translation (safety)

If the UI⟶AST compiler Φ preserves types and binds symbols to intended structures (e.g., $\mathbb{R}$ as a field), and the target ITP *kernel* checks the proof term π of $\Phi(\phi)$, then the original UI statement ϕ holds in the intended structure. External ATP/SMT calls are advisory; *kernels verify* [1], [6]–[9].

4. Operational definition and the F0–F4 taxonomy

Definition (formalizability under budget B).
A mathematics problem P is *formalizable under* B if there exists:

1. an adequate encoding $[\![P]\!]$ in the target logic/system and
2. a checked derivation (by automation and/or interactive steps) within B, where

$$B = (\textit{time limits}, \textit{interaction limits}, \textit{library scope}) \tag{3}$$

Taxonomy (indicative budgets; tune to course level).

- F0 — Fully automatic. Solved without human hints (≤ 30 s wall time). Typical: rewriting, linear arithmetic, polynomial identities. Lean: `simp`, `linarith`, `ring`; Coq: `lia`, `ring`; Isabelle: `simp`, `linarith`, `algebra_simps`; SMT: QF_LIA/NRA; ATP: equality + rewriting.
- F1 — Minimal hints. 1–3 tactics or a short sketch (≤ 5 min). May reuse one auxiliary lemma; at most 2 external calls (Sledgehammer/SMT).
- F2 — Short structured proof. 5–20 min; several steps and 1–3 lemmas using standard libraries (analysis/algebra).
- F3 — Nontrivial development. 20–60 min+; needs helper lemmas/definitions; substantial tactic/solver orchestration.
- F4 — Impractical under BBB. Lacks abstractions/lemmas or requires costly encodings (e.g., delicate geometry, compactness/existence arguments).

Drivers of complexity. (i) Library coverage (definitions/theorems present?); (ii) Logic/fragment (e.g., first-order vs. higher-order, SMT theory availability); (iii) Proof archetype (equational, inductive, ε–δ, existence); (iv) Lemma search (availability vs. invention).

5. Mathematical foundations and worked examples

This section presents a sequence of representative mathematical results, each formulated as a self-contained statement and accompanied by a proof sketch. For each result, we provide formal code snippets in Lean, Coq (Rocq), and Isabelle/HOL, which serve to illustrate its placement within the proposed F-level taxonomy.

5.1. Algebraic identities (ring normalization)

Theorem 4.1 (Binomial identity). $\forall\ x, y \in \mathbb{R} : (x+y)^2 = x^2 + 2xy + y^2$
Proof sketch. Normalization in the equational theory of commutative semirings.

1. Lean:

```
theorem binomial (x y : ℝ) : (x + y)^2 = x^2 + 2*x*y + y^2 := by ring
```

2. Isabelle/HOL:

```
lemma binomial: "(x + y)^2 = x^2 + 2*x*y + y^2"
by (simp add: algebra_simps)
```

3. Coq / Rocq:

```
From Coq Require Import Reals Psatz.
Open Scope R_scope.
Lemma binomial : forall x y: R, (x + y)^2 = x^2 + 2*x*y + y^2.
Proof. ring. Qed.
```

Typical F-level: F0 (library-backed normalization).

5.2. Linear arithmetic and transitivity

Lemma 4.2. $\text{For all } x, y, z \in \mathbb{Z}, \text{if } x \le y \wedge y \le z \text{ then } x \le z.$
Proof. Linear integer arithmetic.

1. Lean:

```
import Mathlib.Data.Int.Basic
open Int
theorem le_transitive (x y z : ℤ) (h₁ : x ≤ y) (h₂ : y ≤ z) : x ≤ z := by
```

```
  exact le_trans h₁ h₂
```

2. Coq / Rocq:

```
From Coq Require Import ZArith Lia.
Lemma le_transitive (x y z : Z) : x <= y -> y <= z -> x <= z.
Proof. intros; lia. Qed.
```

Typical F-level: F0 (via lia/linarith/SMT QF_LIA).

5.3. Induction and sums

Proposition 4.3 (Arithmetic series). $\forall n \in \mathbb{N}: \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$.

Proof sketch. Induction on n.
Lean 4:

```
open Nat
theorem sum_nat (n : ℕ) :
    (∑ k in Finset.range (n+1), k) = n*(n+1)/2 := by
  -- uses library lemmas about `Finset.range` and sums
  simpa using Nat.sum_range_id
```

Typical F-level: F1–F2 (one-page structured proof with library lemmas).

5.4. ε–δ reasoning for limits

Proposition 4.4. $For\ a, b \in \mathbb{R}, \lim_{x \to a} (b\,x) = b\,a$

Proof sketch.
$Given\ \varepsilon > 0, choose\ \delta = \varepsilon / max(1, |b|).\ If\ 0 < |x-a| < \delta\ then\ |b\,x - b\,a| = |b||x-a| < \varepsilon$

Typical F-level: F2–F3 (uses analysis libraries and ε–δ lemmas).

5.5. Separable ODE

Example 4.5. Solve $y' = ky$ (constant $k \in \mathbb{R}$)
Solution. $y(x) = C\,e^{kx}$, A formal proof uses integration facts or exponential series;
Typical F-level: F2–F3 depending on library coverage (existence/uniqueness, differentiability of e^{kx}).

6. Measurement protocol and statistical estimators

Let $\{P_i\}_{i=1}^{n}$ be atomic problems sampled from a course.
Pipeline:

1. Normalization: harmonize notation; split multi-part items into atomic goals.
2. Automation first: run SMT/ATP with strict timeouts (e.g., 30 s, 120 s).
3. ITP with light hints: Lean/Coq/Isabelle with a cap on hints/tactics (e.g., ≤ 3) and external calls.
4. Labeling: assign F0–F4; record $t_{formalize}$ and t_{prove}, #hints, #external calls, tool versions.
5. Dual annotation: two annotators; reconcile; report Cohen's κ
6. Aggregation: compute per-topic %F0…%F4 and medians.

6.1. Proportions and uncertainty

$$\hat{p}_k = \frac{1}{n}\sum_{i=1}^{n} 1\{label(P_i) = F_k\} \tag{4}$$

Table 3
Common proof archetypes and indicative automation strategies

Archetype	Example topics	Likely F-level	Effective procedures
Equational rewriting	polynomial identities	F0–F1	`simp`, `ring`; SMT arithmetic; ATP equality
Linear inequalities	bounds, systems	F0–F1	`linarith`; `lia`; SMT QF_LIA
Induction	sequences, sums	F1–F2	ITP induction + helper lemma
ε–δ limits	continuity	F2–F3	analysis libraries + guided steps
ODE tasks	separable/linear	F2–F3	rewrite + library facts; limited SMT support
Olympiad assignments	geometry/comb.	F3–F4	bespoke lemmas; weak direct automation

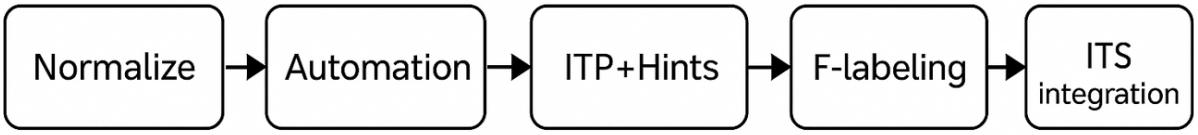

Figure 1: *Measurement pipeline*

7. Integrating F-labels into ITS

Adaptive scheduling.
Mass practice & instant feedback (F0–F1): auto-checkable pools; graded hints tied to mapped tactics (e.g.,"Try linarith"). Strategy instruction (F2): scaffolded sketches; lemma search via apply? / Sledgehammer[2]. Capstones (F3–F4): longer proofs; use miniF2F and LeanDojo tasks to assess retrieval/lemma-search skills; compare with MATH for text-first problem difficulty [14]–[16].

7.1. Feedback mapping with Reverse Projection

Tactic failure ⟶ hint (UI math form).

- System: `linarith` fails.
- Hint: *"Try linarith after isolating linear terms."*

- Math UI mapping: *highlight inequality $x \le y$, $y \le z$ and suggest: "Try combining inequalities to derive $x \le z$."*

Solver counterexample ⟶ minimal falsifying instance.

- System: `linarith` fails.
- Hint: *"Try linarith after isolating linear terms."*
- Math UI mapping: *highlight inequality $x \le y$, $y \le z$ and suggest: "Try combining inequalities to derive $x \le z$."*

Proof state⟶ subgoals listed for the learner with micro-hints.

7.2. Content authoring

Build a bank of *formalized templates* aligned with weekly topics (algebra, analysis, linear algebra, ODEs), each tagged with F-labels and example code for Lean/Coq/Isabelle.

8. Illustrative pilot and reproducibility

Scope and caveat. We conducted an illustrative pilot to exercise the labeling protocol on a curated set of undergraduate problems. Importantly, this pilot was not run in a live course; its purpose is to demonstrate workflow and artifacts, not to estimate course-level shares.
Setup. The set comprises 12 atomic problems (algebraic identities and linear inequalities, induction/sums, ε–δ limits, a separable-ODE pattern). Budgets were fixed as specified earlier: automation timeouts 30 s & 120 s, ITP hint cap ≤ 3, standard libraries for Lean 4 / Coq (Rocq) 8.19 / Isabelle 2025, and Z3 / cvc5 / E / Vampire back-ends. Tool versions and commands are included in the artifact.
Annotation and reliability. Two authors independently labeled the items (F0–F4) under the same budgets; disagreements were reconciled. Cohen's $\kappa = 0.78$ with 83 % raw agreement (10/12), indicating substantial agreement.

Table 4
F-level distribution for the illustrative pilot

F-level	Count	Share (%)	Median time (s) to formalize	Median time (s) to prove
F0	4	33.3	18	10
F1	2	16.7	35	70
F2	4	33.3	55	180
F3	2	16.7	120	420
F4	0	0	-	-

Interpretation. The F0–F1 items are routine equational/linear tasks (e.g., `ring`, `linarith`/`lia`; QF_LIA in SMT). F2 typically needs short structured proofs (induction, ε–δ). F3 requires helper lemmas/definitions (e.g., ODE facts). No F4 appeared under these budgets in this illustrative set. These outcomes should not be read as course-level estimates.

9. Threats to validity

- Version drift. Library growth can shift some items from F2⟶F1 or F1⟶F0; always report tool versions and re-sample each term.
- Annotator expertise. F-labels depend on the skill of annotators in specific ITPs; mitigate with dual annotation and κ.
- Topic idiosyncrasies. "Elementary" geometry/olympiad tricks may be F3–F4 due to modeling overhead.
- Logic mismatch. Results apply primarily to HOL-centric ITPs with ATP/SMT bridges.

10. Conclusion

This paper presents a resource-aware definition of formalizability; introduces the five-level F0–F4 taxonomy; specifies a measurement protocol with practical estimators; expands the set of mathematical exemplars; and provides a UI-to-API mapping that grounds intelligent tutoring system (ITS) adaptation in the concrete capabilities of modern provers. Future work includes: (i) releasing a benchmarked corpus of course problems annotated with F-labels; (ii) conducting multi-institutional replications; and (iii) publishing fully reproducible scripts for Isabelle/HOL, Lean, and Coq (Rocq), together with solver configurations. This paper includes an illustrative pilot (8) that was not deployed in a live course; the reported figures are demonstrative.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-5 Pro for draft structuring and wording. The authors reviewed and edited the content and take full responsibility for the publication.

References

[1] Isabelle Team, "Documentation (Isabelle2025)," 2025. Online. URL: https://isabelle.in.tum.de/documentation.html

[2] J. C. Blanchette, Sledgehammer for Isabelle/HOL (User's Guide), Isabelle2025 Manuals, 2025. Online. URL: https://isabelle.in.tum.de/dist/doc/sledgehammer.pdf

[3] The Rocq/Coq Development Team, The Coq Proof Assistant Reference Manual, v8.19, 2025. Online. URL: https://rocq-prover.org/doc/V8.19.0/refman/index.html

[4] J. Avigad, P. Massot, Mathematics in Lean, 2025. Online. URL: https://avigad.github.io/mathematics_in_lean/mathematics_in_lean.pdf

[5] Lean Community, mathlib4 Documentation, 2025. Online. URL: https://leanprover-community.github.io/mathlib4_docs/index.html

[6] C. Barrett, P. Fontaine, and C. Tinelli, The SMT-LIB Standard, Version 2.7, Feb. 5, 2025. Online. URL: https://smt-lib.org/papers/smt-lib-reference-v2.7-r2025-02-05.pdf

[7] A. Chukhray, D. Dvinskykh, V. Narozhnyy and T. Stoliarenko, "Using an expression tree for adaptive learning," 2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, 2023, pp. 1-5, doi: 10.1109/DESSERT61349.2023.10416497

[8] Microsoft Research, Online Z3 Guide, 2025. URL: https://microsoft.github.io/z3guide/

[9] cvc5 Team, cvc5 Documentation (Proof Rules and Proof Production), 2022–2025. URL: https://cvc5.github.io/docs/cvc5-1.0.0/proofs/proofs.html

[10] H. Barbosa et al., "cvc5: A Versatile and Industrial-Strength SMT Solver," TACAS, 2022, doi: 10.1007/978-3-030-99524-9_24.

[11] G. Sutcliffe, "The TPTP Problem Library," 2025. Online. URL: https://tptp.org/TPTP/

[12] S. Schulz, “The E Theorem Prover,” 2025. URL: https://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html
[13] Vampire Team, “Vampire Theorem Prover,” 2025. URL: https://vprover.github.io/
[14] K. Zheng, J. M. Han, S. Polu, et al., “miniF2F: A Cross-System Benchmark for Formal Olympiad-Level Mathematics,” arXiv:2109.00110, 2021. doi: 10.48550/arXiv.2109.00110
[15] K. Yang, et al., “LeanDojo: Theorem Proving with Retrieval-Augmented Language Models,” NeurIPS Datasets & Benchmarks, 2023. doi: 10.48550/arXiv.2306.15626
[16] D. Hendrycks, et al., “Measuring Mathematical Problem Solving with the MATH Dataset,” NeurIPS Datasets & Benchmarks, 2021. doi: 10.48550/arXiv.2103.03874
[17] A. Thoma and P. Iannone, “Learning about Proof with the Theorem Prover LEAN,” *Int. J. Res. Undergrad. Math. Ed.*, 2022. doi: 10.1007/s40753-021-00140-1.
[18] P. Massot, “Teaching Mathematics Using Lean and Controlled Natural Language,” in *Proc. ITP 2024*, LIPIcs 309:27, 2024. doi: 10.4230/LIPIcs.ITP.2024.27.
[19] M. L. Bottoni, “Teaching ‘Foundations of Mathematics’ with the Lean Theorem Prover,” *arXiv:2501.03352*, 2025.
[20] “Courses using Lean,” Lean community page, 2025.
[21] “Learning Mathematics with Lean—presentations & abstracts,” 2024–2025.
[22] B. C. Pierce, “Software Foundations, 15 years on,” talk slides, Univ. of Pennsylvania, 2023. (Accessed: Sep. 2025).
[23] B. C. Pierce et al., *Software Foundations* book series website, 2025.
[24] M. Henz, “Teaching Experience: Logic and Formal Methods with Coq,” in *LNCS*, 2011, pp. 199–215. Doi: 10.1007/978-3-642-25379-9_16.
[25] J. Villadsen, et al., “Using Isabelle in Two Courses on Logic and Automated Reasoning,” *FMTea’21*, 2021.
[26] F. K. Jacobsen, “On Exams with the Isabelle Proof Assistant,” 2023.
[27] T. Nipkow, *Programming and Proving in Isabelle/HOL*, tutorial/book, 2025.
[28] T. Nipkow, L. C. Paulson, and M. Wenzel, *Isabelle: A Proof Assistant for Higher-Order Logic (Tutorial)*, 2025.
[29] S. Blazy, “Teaching Deductive Verification in Why3 to Undergraduate Students,” in *LNCS*, 2019. doi: 10.1007/978-3-030-32441-4_4.
[30] P. G. Vaillancourt, et al., “ACL2 in DrScheme (DRACULA): A Pedagogic Framework,” 2006.
[31] C. Eggensperger, “Proof Pad: A Pedagogic ACL2 IDE,” in *TFP 2013*, 2013.
[32] J. Cai, “AutoRubric: Autograding Template-Based Exam Programs with SMT,” UC Berkeley Tech. Rep., 2020.
[33] “Z3Guide: A Scalable, Student-Centered, and Extensible Guide to SMT,” *arXiv:2506.08294*, 2025.
[34] K. Yang, et al., “LeanDojo: Theorem Proving with Retrieval-Augmented LMs,” *NeurIPS Datasets & Benchmarks*, 2023. doi: 10.48550/arXiv.2306.15626.
[35] K. Zheng, J. M. Han, S. Polu, et al., “miniF2F: A Cross-System Benchmark for Formal Olympiad-Level Mathematics,” *arXiv:2109.00110*, 2021. doi: 10.48550/arXiv.2109.00110.
[36] A. Chukhray, T. Stoliarenko, O. Yevdokymov, V. Demyanenko, “Possibilities of using Intelligent Tutoring Systems (ITS) in Higher Mathematics Courses,” Open Information and Computer Integrated Technologies, no. 102, pp. 92–119, Feb. 2025, doi: 10.32620/oikit.2024.102.07.