GARNELO, Irene & LIEBENDÖRFER, Michael
Paderborn

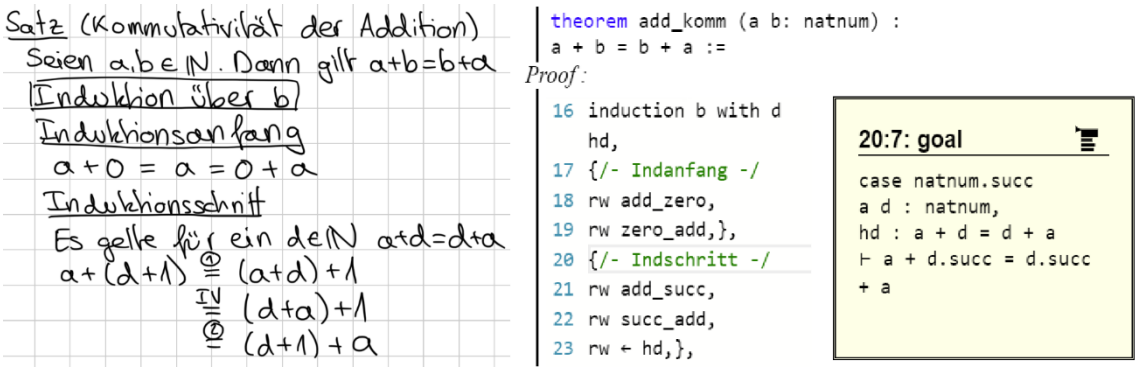
Herausforderungen beim Einsatz interaktiver Theorembeweiser in der Hochschullehre

1. Einleitung

Digitale Werkzeuge sind ein wichtiger Bestandteil der mathematischen Hochschullehre. Lange Zeit gab es allerdings kaum Werkzeuge zum Lernen mathematischen Beweisens (Sangwin & O'Toole, 2017). Aktuell etablieren sich Theorembeweiser als eine relevante digitale Innovation sowohl in der mathematischen Forschung als auch in der Lehre (Massot, 2021). Theorembeweiser sind interaktive Software-Tools, in denen Beweise formuliert werden können und die bei der Bearbeitung eines Beweises Feedback bieten. Wir präsentieren Einblicke in ein laufendes Entwicklungsprojekt zum Einsatz von Lean vor, konkret zu den Herausforderungen die in diesem Rahmen entstehen.

2. Theorie und Hintergründe

In Theorembeweisern werden Beweise in einer computer-formalisierten Sprache geschrieben. Wir zeigen als Beispiel den Beweis der Kommutativität der Addition in den natürlichen Zahlen links in normaler Sprache und rechts im Theorembeweiser Lean:



Nach der Formulierung des Satzes beginnt unter dem Stichwort "Proof" der Beweis, der von einer Feedbackbox begleitet wird. Während der Benutzer den Beweis in der Lean-Sprache führt, gibt das Tool in der gelben Box Feedback zum Beweiszustand. Dazu gehören was gegeben ist, was bereits gezeigt wurde und was noch zu zeigen ist, sowie Fehlermeldungen, falls ein Fehler auftritt. An diesem Beispiel kann man sehen, dass die Struktur der beiden Beweise ähnlich ist, die Sprache aber sehr unterschiedlich ist und Lean-spezifisches Wissen erfordert. Die Lean-Sprache muss also erst gelernt werden, bevor das Tool erfolgreich eingesetzt werden kann.

In: P. Ebers, F. Rösken, B. Barzel, A. Büchter, F. Schacht & P. Scherer (Hrsg.),
Beiträge zum Mathematikunterricht 2024.
57. Jahrestagung der Gesellschaft für Didaktik der Mathematik. WTM.
https://doi.org/10.37626/GA9783959872782.0

Das Erlernen der Lean-Sprache kann mit dem Erlernen einer Programmiersprache verglichen werden. Die Informatikdidaktik teilt das Wissen, welches man zum Erlangen einer Programmiersprache braucht, in drei aufeinanderbauende Ebenen (Bayman, & Mayer, 1983) ein: die syntaktische Ebene (z. B. Klammersetzung), die konzeptuelle Ebene (z. B. Variablenverständnis) und strategisches Wissen (z. B. Debuggen eines Programms). Diese Kategorisierung ist wichtig, weil Kurse oft nur die syntaktische Ebene adressieren, was Studierende daran hindert, die zugrundeliegenden Programmierprinzipien zu lernen und sich strategisches Wissen anzueignen (McGill & Volet, 1997).

Trotz dieser Einstiegshürde hat Lean in der Lehre ein großes Potential, da es Studierenden bei der Arbeit mit Beweisen direktes ausführliches Feedback gibt. Zusätzlich benennt Massot (2021) als Potentiale von Lean das Erlernen der Präzision, die die mathematische Sprache erfordert, sowie die bewusstere Wahrnehmung der Beweisphasen. Thoma und Ianonne (2022) haben empirisch bestätigt, dass Studierende die mathematische Notation sowie Beweisstrukturen besser beherrschen, nachdem das Tool eingesetzt wurde.

Auch beim Beweisen sind syntaktische, konzeptuelle und strategische Hürden bekannt (Moore, 1994). Offen ist, inwieweit Probleme mit Lean auch Lerngelegenheiten für allgemein bedeutendes Wissen zum Beweisen sind. Um dieses Potential besser zu verstehen, betrachten wir die Verbindung des Lernprozesses von Lean und dem des Beweisens. Dafür haben wir als ersten Schritt Probleme mit Lean hinsichtlich der drei Wissens-Ebenen analysiert.

Wir haben eine Lernumgebung entwickelt, die aus einer online-Lernumgebung und zusätzlich Arbeitsblättern besteht. In der online-Umgebung gibt es Aufgaben, die in Lean gelöst werden sollen. Zu jeder Aufgabe gibt es noch eine Einführung in das mathematische Thema und in die neuen Lean-Techniken, die gebraucht werden. Zusätzlich gibt es noch ausklappbare Hinweise. Auf den Aufgabenblättern gibt es begleitende Aufgaben. Diese sollen beim Einstieg in Lean helfen, und sind konkret auf das Wissen auf den drei Ebenen zugeschnitten. Thematisch geht es in der Lernumgebung darum, von den Peano-Axiomen ausgehend grundlegende Eigenschaften der natürlichen Zahlen zu beweisen was dazu führt, dass ein Großteil der Aufgaben Induktionsbeweise sind.

3. Forschungsfrage

Unsere Forschungsfrage für diesen Beitrag lautet: Wie gliedern sich Probleme bei der Arbeit mit der Lean-Lernumgebung hinsichtlich der Programmierkenntnis-Ebenen nach Bayman und Mayer (1983)?

Die gefundenen Probleme sollen dabei mit bekannten Problemen beim Beweisen (Moore, 1994) in Beziehung gesetzt werden.

4. Methode

Die Lernumgebung wurde in das Modul "Mathematik am Computer" eingebunden, welches für Studierende im Gymnasiallehramt im dritten Semester angeboten wird. In zwei 90-minütigen Vorlesungen wurden Studierende in die Themen computergestütztes Beweisen, Theorembeweiser und Lean eingeführt. Der Großteil der Zeit war der Arbeit mit der Lernumgebung gewidmet, bei der fünf Studierende (zwei Paare und ein einzelner Student) videographiert wurden. Die Daten aus den Videos wurden zu je einem Transkript zusammengefasst, in dem die Dialoge, der Code und die Handlungen der Studierenden erfasst sind. Die Transkripte wurden mit einer deduktiv-induktiven Inhaltsanalyse ausgehend von den Kategorien nach Bayman und Mayer (1983) kodiert. Dabei sind Muster entstanden, anhand der wir für die jeweiligen Ebenen typische Hürden identifizieren konnten.

5. Ergebnisse

Studierende sind zunächst auf Probleme syntaktischer Natur gestoßen. Diese sind kaum vergleichbar zu allgemeinen Problemen beim Beweisen und werden hier nicht näher diskutiert.

Auf konzeptueller Ebene wurden Parallelen zu Problemen beim traditionellen Beweisen ohne digitale Werkzeuge sichtbar. Während Studierende generell oft Schwierigkeiten haben, Definitionen zu formulieren, hatten Studierende in unserer Studie Probleme mit der Bedeutung von Schlüsselbegriffen, insbesondere der Beweistechniken in Lean. Manchmal wussten sie auch nicht, welche Auswirkung die Anwendung eines Satzes auf eine gegebenen Aussage haben würde, oder sie interpretierten die Bedeutung von Fehlermeldungen falsch. Studierende hatten zudem Schwierigkeiten mit den Konzepten der Sprache, beispielsweise mit der Bedeutung der Umkehrung eines Satzes, und damit, wie Scopes und Variablen funktionieren. Schließlich war die Anwendung von Schritten, die in einem bestimmten Beweiszustand ungültig sind, ein Indikator für unzureichende Vorstellungen bezüglich des Beweiszustandes und konkreter Beweisschritte.

Strategische Schwierigkeiten hatten Studierende bei allen Schritten eines Beweises. Etwa hatten manche Studierende einen Beweisschritt verwendeten, der für den gegebenen Beweiszustand gültig war, den Beweis jedoch nicht vorangebracht hat. In einigen Fällen wählten Studierende die falsche Variable für eine Induktion aus, oder sie wählten eine kompliziertere Beweisstrategie als notwendig (zum Beispiel Induktion, wenn eine algebraische Umformung ausgereicht hätte).

6. Diskussion

Probleme mit konzeptuellem Wissen sind vergleichbar zu mangelndem Verständnis von Konzepten in der Mathematik. Die hier auftretenden Probleme hingen oft an speziellen Konzepten von Lean, sodass wir auch hier nur wenig Lernpotential für das allgemeine Beweisen sehen.

Strategische Probleme mit Lean waren Problemen beim allgemeinen Beweisen sehr ähnlich. Das ist plausibel, sind sie doch unabhängig von der konkreten Sprache. Allerdings berichtet Moore (1994), dass eine besondere strategische Hürde darin besteht, einen Beweis zu beginnen. Bei uns dagegen war der Start nicht schwieriger als der Rest. Dies könnte daran liegen, dass man in Lean nach jedem Schritt mit einem neuen Beweiszustand konfrontiert ist, der mit dem Start eines neuen Beweises verglichen werden kann. Eine wichtige Frage ist, ob Studierende dadurch zu sehr auf einzelne Schritte und nicht auf die gesamte Beweisstruktur fokussiert sind.

Die Probleme mit Lean überlappen sich nur teilweise mit denen beim Beweisen. Die Aneignung von Lean braucht also eine initiale Leistung. Auf der konzeptuellen und vor allem auf der strategischen Ebene gibt es aber große Überschneidungen. Wenn Lean auf diesen Ebenen das Beweisen unterstützen kann, könnte dies die initiale Leistung rechtfertigen.

7. Literatur

Bayman, P., & Mayer, R. E. (1983). A diagnosis of beginning programmers' misconceptions of basic programming statements. *Communications of the ACM, 26*(9), 677-679.

Massot, P. (2021). *Why formalize mathematics?* https://www.imo.universite-paris-saclay.fr/~patrick.massot/files/exposition/why_formalize.pdf

McGill, T. J. & Volet, S. E. (1997). A Conceptual Framework for Analyzing Students' Knowledge of Programming. *Journal of Research on Computing in Education, 29*(3), 276-297.

Moore, R. C. (1994). Making the transition to formal proof. *Educational Studies in Mathematics, 27*(3), 249-266.

Sangwin, C.J., & O'Toole, C. (2017). Computer programming in the UK undergraduate mathematics curriculum. *International Journal of Mathematical Education in Science and Technology, 48*(1).

Thoma, A., & Iannone, P. (2022). Learning about proof with the theorem prover lean: the abundant number task. *International Journal of Research in Undergraduate Mathematics Education, 8*(1), 64-93.