Game Over oder Q.E.D?

Aufgabenauswahl
SoSe 2023, HHU

January 23, 2024

1 Installation

https://leanprover-community.github.io/get_started.html

Eventuell könnte es auch funktionieren, einfach das "lean"-Plugin im VSCode zu installieren und dann wenn Pop-ups erscheinen, die benötigte elan toolchain zu installieren.

1.1 Beispiel Projekt

Ein Übungsprojekt, das im VSCode läuft, könnt ihr mit

```
leanproject get mathematics_in_lean

cd mathematics_in_lean
code .
```

(in der Command Line eingegeben) herunterladen. Siehe auch:

https://leanprover-community.github.io/mathematics_in_lean/01_Introduction.html#getting-started

Daraus könntet ihr Aufgaben aus `02_basic` oder `03_logic` zum Einstieg lösen. Der Infoview rechts im VSCode sollte den Beweisstand am Cursor anzeigen, ähnlich wie das im Spiel der Fall war.

1.2 Neues Projekt

Um ein neues Projekt (inklusive mathlib) zu starten, tippt man

```
leanproject new my_project_name

cd my_project_name
code .
```

starten. Dann unter `src` eine neue Datei, z.B. `main.lean` erstellen und darin `#eval "Hello World"`! eingeben. Das Lean plugin im VSCode sollte dann einen Infoview anzeigen.

1.3 Known Windows stuff

- Man muss `git` installiert haben. Falls dies noch nicht der Fall ist (und ein Fehler erscheint), kann man dieses hier runterladen: https://gitforwindows.org/
- Bei weiteren Problemen finde ich vielleicht auf Zulip antworten :)

2 Lean3 vs. Lean4

Das Spiel war in Lean4 geschrieben, die Projekte werden aber in Lean3 sein, da die mathematische Library noch nicht vollstänging geportet ist und in Lean4 noch mit allerlei Problemen zu kämpfen hat. Für den Anwender sind sich die beiden Versionen aber sehr ähnlich.

	Lean 3	Lean 4

Statements:	`theorem my_theorem (n : ℕ) : 0 ≤ n :=`
	`begin`
	`  rw [my_lemma],`
	`  assumption,`
	`end`
		`theorem my_theorem (n : ℕ) : 0 ≤ n := by`
		`  rw [my_lemma]`
		`  assumption`

Implizite Funktionen:	`λ (x : ℕ), x ^ 2`	`fun x ↦ x ^ 2`
		`-- or`
		`fun (x : ℕ) => x ^ 2`

Indentation	optional (dafür Taktiken **mit Komma trennen!)**	ist wichtig (wie in Python). Ein Block von Taktiken muss einheitlich eingerückt sein.
Namensgebung:	`-- alles snake case` `even_square, even, add_comm_ring`	`-- Grossschreibung haengt davon ab, ob es ein Theorem/Definition/Struktur ist:` `even_square, Even, AddCommRing`
`rw`	Klammern optional: `rw add_comm`, oder `rw [add_comm],`	Klammern Notwendig: `rw [add_comm]`

Expliziter Funktionsaufruf:	`-- Bsp thm.`
	`theorem my_thm {a b c : ℕ} (h : Prop) : h`
	
	`have f := @my_theorem _ d _ g,`
		`-- Bsp thm.`
		`theorem my_thm {a b c : ℕ} (h : Prop) : h`
		
		`have f := @my_theorem _  d _ g`
		`-- or`
		`have f := my_theorem (b := d) g`

`constructor` Taktik	`constructor` oder `split` machen beide ziemlich das Gleiche.	nur `constructor`
`lemma` keyword	`lemma` und `theorem` sind Synonyme.	ich glaube, `lemma` existiert nicht mehr und man verwendet immer `theorem`.
∀-Notation	Π ist ein anderes Symbol, das manchmal anstatt ∀ verwendet wird.	Nur noch ∀.

3 Projekte

- Ziel des Projekt ist es, ein Thema anhand von ein paar Übungsaufgaben einzuführen, und damit eine Grundlage zur Erweiterung des Lernspiels zu entwickeln.
- Das Projekt kann aus einer (oder mehreren) Lean Dateien bestehen, die einzelnen Übungsaufgaben (je nach Thema 5-20, aufeinander aufbauend) sind jeweils ein `theorem` Statement, mit einem Taktik-Beweis.
- Zwischen den `theorem` Statements dürft ihr gerne mit Kommentaren weiter erklären, was man für eine Aufgabe braucht, z.B. wenn ihr Resultate aus der Mathlib annehmt und braucht, ohne diese zu beweisen. (/– comment –/ oder –– *comment*)
- Bei den Beweisen sollte der Fokus auf mathematische Verständlichkeit gelegt werden (und nicht etwas wie in Mathlib, kurze, optimierte Kompilierzeit). Insbesondere kann es auch sinnvoll sein, ein Resultat weniger generalisiert zu zeigen, als es in den Mathlib auftaucht.
- Häufig werden die Resultate schon in ähnlicher Form in der Mathlib sein. Dann muss unter anderem eine Entscheidung getroffen werden: Will das Projekt die Resultate von Grund auf nochmals beweisen, mit Fokus auf die Beweistechnik und -lesbarkeit, oder möchte der Fokus darauf bestehen, die API um ein Thema gut kennenzulernen und Übungsaufgaben dazu bearbeitet werden.
- Insbesondere kann es vorkommen, dass manche Projekte zu einfach oder zu schwierig sind. Dann sollte die Aufgabenstellung modifiziert werden (i.e. die gestellten Aufgaben sind nicht unbedingt als Mindestanforderung zu verstehen, sondern eher als Guideance.)

behandelt wurde.

Aufgaben

Ich habe die Projekte ungefähr nach Schwierigkeit eingestuft:

(*) : Sollte relative kurz und geradlinig ablaufen. Für diejenigen, die gerne langsam anfangen und dann das Projekt eigenständig erweitern wollen.
(**) : Dürfte etwas mehr zu schreiben geben, aber keine grossen Überaschungen. Für diejenigen, die gerne ein Ziel vor Augen haben und darauf hin arbeiten.
(***) : Schwer einzuschätzen. Könnte in knifflige Probleme laufen, kann aber auch sein, dass es relativ einfach geht. Für diejenigen, die gerne etwas herumprobieren.

Aber bei allen gilt, dass wenn Schwierigkeiten auftreten, dann können wir diese gemeinsam angehen oder umgehen, also probiert gerne auch etwas, was euch interessiert.

3.1 Grundlagen

1 Endliche Mengen (*)

Jede endliche Vereinigung endlicher Mengen ist wieder endlich.

Die Potenzmenge einer endlichen Menge ist wieder endlich.

Ist M endlich und nicht leer, so sind die Menge $\mathfrak{P}^0(M) \subset \mathfrak{P}(M)$ der Teilmengen, die aus einer geraden Anzahl von Elementen bestehen, und die Menge $\mathfrak{P}^1(M)$ der Teilmengen, die aus einer ungeraden Anzahl von Elementen bestehen, gleich mächtig.

Hier geht es ersten darum, den Unterschied zwischen 'set U' und 'finset U' einzuführen. Für 'finset U' sind diese Aufgaben dann alle praktisch trivial. Für '(S : set U)', muss man die Annahme '[finite S]' hinzufügen, dann werden diese Aufgaben interessanter.

3.2 Lineare Algebra

2 Matrizen I (*)

Jede invertierbare Matrix ist ein Produkt von Elementarmatrizen.

Jede strikte obere Dreiecksmatrix ist nilpotent.

Für jede nilpotente Matrix A ist $(\mathbb{1} - A)$ invertierbar.

Hier geht es vorallem darum, wie man mit konkreten Matrizen arbeitet. Es wäre also durchaus gut, Matrizen von Grund auf zu erklären. Eine Challenge könnte die Interaktion von abstrakten Matrizen ("sei m eine Matrix") und konkreten Matrizen ('!![2, 4; 5, 6]') sein.

3 Matrizen II (*)

Berechne das Zentrum von $\mathrm{GL_n}(\mathbb{Q})$.

Sei K ein Körper. Zeige, dass die Spur die einzige Abbildung $t\colon \mathrm{Mat}(n \times n) \to K$ ist, für die gilt: t ist linear, t wirft die Einheitsmatrix auf 1, und $t(AB) = t(BA)$.

Dieses Projekt beschäftigt sich mehr damit, was man über abstrakte Matrizen zeigen kann. Die Aufgaben können nach belieben erweitert werden.

4 Orthonomalisierung (**)

Führe das Gram-Schmidt-Orthonormlisierungsverfahren erneut ein.

Benutze es, um klassische Aufgaben zu lösen

Dieses Projekt ist etwas wager formuliert und verlangt entsprechend etwas Eigeninitiative. Das GS-Orthonormalisierungsverfahren existiert prinzipiell in Mathlib, ist aber für generelle normierte Räume gezeigt. Entsprechend, muss man sich eventuell mit etwas API zurechtfinden und das Resultat spezifizieren (z.B. konkret für Vektorräume, über $\mathbb{R}$ oder so)

5 Vektorräume: Basis & Dimension (**)

Konstruiere auf der Menge alle Abbildungen $\mathrm{Abb}(\mathbb{R},\mathbb{R}) := \{f\colon \mathbb{R} \to \mathbb{R}\}$ eine $\mathbb{R}$-Vektorraumstruktur. Konstruiere ferner in Lean für $n \in \mathbb{Z}$ die folgenden Abbildungen $f_n\colon \mathbb{R} \to \mathbb{R}$:

$$f_n(x) := \begin{cases} 0 & \text{falls } x < n \\ 1 & \text{falls } x \geq n \end{cases}$$

Zeige schließlich, dass die Menge $\{f_n \mid n \in \mathbb{Z}\}$ aller dieser Abbildungen linear unabhängig ist in $\mathrm{Abb}(\mathbb{R},\mathbb{R})$. Zeige ferner, dass es sich nicht um ein Erzeugendensystem handelt.

Eine *Fahne der Länge d* in einem Vektorraum V ist eine Kette von Untervektorräumen von V der Form $U_0 \subsetneq U_1 \subsetneq U_2 \subsetneq \cdots \subsetneq U_d$. Ist V endlich-dimensional, so ist die maximale Länge einer solchen Fahne gleich der Dimension von V.

Dieses Projekt führt Basen ein und die Dimension eines Vektorraums. Einige Notationen könnten etwas anspruchsvoll sein, zum Beispiel gibt es verschiedene Arten in Lean über "Dimension" zu reden, und man muss manchmal herausfinden, wie diese interagieren. Prinzipiell sollte das Projekt aber straight forward sein.

6 Vektorräume: Kardinalität (*)

Sei V ein endlich dimensionaler $\mathbb{F}_p$-Vektorraum mit Dimension n. Zeige $\#V = p^n$.

(zwischenresultat) Sei K ein Körper mit Kardinalität $\aleph_0$ und sei V ein endlich dimensionaler K-Vektorraum. Zeige, dass für die Kardinalität von V gilt: $\#V \leq \aleph_0$. ($\aleph_0$ is die Kardinalität von $\mathbb{N}$)

Zeige, dass $\mathbb{R}$ kein endlich dimensionaler $\mathbb{Q}$-Vektorraum ist.

Dieses Projekt führt Kardinalität von Mengen ein und arbeitet damit. Bei diesem Projekt werden keine grösseren Hürden auftauchen, aber es wird schon etwas zu tun geben.

Benutze das Resultat "$\mathbb{R}$ ist überabzählbar" aus der Mathlib ohne es zu beweisen! (Der Beweis davon ist relativ komplex)

7 Eigenwerte (**)

Wenn μ ein Eigenwert von f ist, ist μ^2 ein Eigenwert von $f \circ f$.

Sei $f\colon V \to V$ ein Endomorphismus eines endlich-dimensionalen Vektorraums derart, dass $f \circ f = 0$. Zeige, dass f genau einen Eigenwert hat, nämlich 0.

Sei V ein endlich-dimensionaler Vektorraum über K, und seien $f, g \in_K (V)$ zwei Endomorphismen. Zeige, dass die Eigenwerte von $f \circ g$ und $g \circ f$ übereinstimmen.
(Für Eigenwerte ungleich Null ist die Annahme, dass V endlich-dimensional sei, überflüssig. Es hilft auch im endlich-dimensionalen Fall, Eigenwerte ungleich Null und Null separat zu betrachten.)

Dieses Projekt führt Eigenwerte ein und arbeitet mit diesen. Auch hier kann man die Aufgaben, die man tatsächlich über Eigenwerte löst, variieren.

Analysis

8 Folgen und Limits (*)

Sei $I = [a, b]$ und $f : I \to \mathbb{R}$ stetig. mit $f(I) \subset I$. Zeige, dass es ein $x \in I$ gibt, so dass $f(x) = x$.

Berechne verschiedene Limits mit Hilfe der Theoreme aus Analysis I.

Dieses Projekt sollte sehr leicht sein, aber trotzdem beliebig ausbaubar. Etwas Eigeninitiative bei der Auswahl von beliebten Aufgaben ist erforderlich, und dabei kann man sich das Projekt selbständig anspruchsvoller gestalten.

9 Reihen und Konvergenz (**)

Sei $n, N \in \mathbb{N}$, $z \in \mathbb{C}$. Zeige $\sum_{k=0}^{n} \binom{N+k}{k} = \binom{N+1+n}{n}$.
Zeige $\frac{1}{(1-z)^{N+1}} = \sum_{n=0}^{\infty} \binom{N+n}{n} z^n$.

Zeige dass $\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$ konvergiert für $s > 1$ und divergiert für $s \geq 1$.

Reihen sollten so in Lean bestehen und eine gute API haben. Die Aufgabe dürfte dadurch relativ einfach werden, allerdings kenn ich mich mit dem Teil der Library nicht aus, weshalb es auch sein kann, dass das Projekt etwas schwieriger wird. Unabhängig davon kann man dieses Projekt über Reihen und Konvergenz sehr gut an den gewünschten AUfwand anpassen.

10 Integration (**)

Lerne wie man mit Lean Übungen zur Integration löst und berechne ein paar klassische Integrale deiner Wahl, z.B. $\int_0^{\infty} \frac{1}{1+e^x} dx$ oder $\int_{\frac{1}{a}}^{a} \frac{x^2}{1+x^4}$.

Auch hier kenne ich das Interface von Lean gar nicht und die Schwierigkeit ist dadurch schwer abzuschätzen. Aber das Projekt kann gut dem Fortschritt angepasst werden.

11 Banach'scher Fixpunktsatz (***)

Beweise den Fixpunktsatz von Banach: Sei A eine nicht-leere abgeschlossene Teilmenge eines vollständigen metrischen Raumes und $T : A \to A$ eine kontrahierende Abbildung. Dann hat T genau einen Fixpunkt

Der Satz sollte bereits in Mathlib sein, und zu einem grossen Teil sollte man den Beweis daraus adaptieren können. Jedoch hat diese Aufgabe doch mit einigen mathematischen Definitionen zu tun, und es könnten unerwartete Hürden auftauchen.

Gruppentheorie

12 Normalteiler (**)

Normalteiler: Jede Untergruppe vom Index 2 ist ein Normalteiler.

Die alternierende Gruppe ist eine normale Untergruppe der symmetrischen.

Dieses Projekt ist vorallem darauf ausgelegt, Gruppentheorie kennenzulernen. Normalteiler sind eine Option, man kann aber auch nach Wunsch in andere Richtungen gehen, wie zum Beispiel simple Gruppen, etc.

13 Endliche Gruppen (*)

Zeige, dass jede Gruppe der Ordnung ≤ 3 abelsch ist.

Berechne alle Gruppen der Ordung $4, 5$.

Die symmetrische Gruppe S_n ist für kein $n > 2$ kommunativ.

Dieses Projekt sollte mathematisch nicht sonderlich anspruchsvoll sein, aber es gilt genügend API zu endlichen Gruppen kennenzulernen und herauszufinden, wie diese sinnvoll gebraucht wird.

14 Satz des Euklid (*)

(*) Es gibt unendlich viele Primzahlen

Das sollte relative direkt machbar sein, aber gewisse "triviale" Schritte sind in Lean vielleicht ein bisschen anspruchsvoller als auf Papier. In Mathlib gibt es zwei Notationen von "prim": 'prime' und 'nat.prime', ich würde vermutlich mit 'nat.prime' anfangen.

Könnte man gegebenenfalls Richtung Primfaktorzerlegungen erweitern, oder noch in folgende Richtung schauen:

Berechne Einheitengruppe der Gaußschen Zahlen $\mathbb{Z}[\mathbf{i}]$.

15 Fermat's Little Theorem (*)

Beweise Fermat's Little Theorem per Induktion. ($a^p \equiv p \pmod{p}$.)

Zeige $(a+1)^p \equiv a^p + 1 \pmod{p}$.

Dieses Projekt sollte ebenfalls eher kurz sein, kann dann aber erweitert werden, z.B. Anwendungen des Theorems zu Ordnungen von Gruppen.

16 Euklidischer Algorithmus (*)

Zeige, dass wenn $\gcd(a, n) \mid c$ then $ax \equiv c \pmod{n}$

Benutze den Euklidischen Algorithmus um $\gcd(73, 25)$ zu berechnen.

Sei $a^2 + b^2 = c^2$. Zeige, dass a oder b durch 3 teilbar ist.

Die letzte Aufgabe hat nichts mit dem Euklidischen Algorithmus zu tun, aber würde noch gut ins Thema mit Restklassenrechnung (modulo) passen.

17 Diverses (*)

$\sqrt{2}$ ist irrational.

Zeige oder widerlege: $17^4 < 31^{11}$. $2^{3333} < 3^{2222}$.

Arbeite mit Induktion über $\mathbb{Z}$

Dieses Projekt könnte generelle grundlegende Aufgaben zum rechnen mit Zahlen (natürlich, ganz, rational, ...) angehen.

Das erste sollte ein ganz kurzer Beweis sein. Das zweite stimmt per `rfl`, aber wenn man Aufgaben mit grossen Zahlen angibt, dann kriegt man timeouts. Entsprechend braucht man andere Rechenmethoden.

Projekte aus höheren Semestern

Diese Projekte gehen potentiell über den Inhalt von den ersten 2-3 Semestern eines Mathestudiums hinaus, und könnten entsprechend etwas anspruchsvoller sein.

18 Noether'scher Polynomring (**)

Sei R ein noether'scher Ring. Zeige, dass $R[X]$ ebenfalls noether'sch ist.

Dies ist mathematisch eine sehr schöne und klassische Übungsaufgabe. Challenges könnten eventuell daraus entstehen Polynomringe in Lean zu verstehen und damit zu arbeiten.

19 Isomorphiesätze (**)

Zeige den Homomorphiesatz $G/\ker(f) \cong \text{img}(f)$.

Zeige die verschiedenen Isomorphiesätze als Korollare

Finde Übungen zu den Sätzen

Die Resultate sind alle in Mathlib, es geht auch hier mehrheitlich darum, diese zu finden und sinnvoll anzuwenden. Es könnte interessant sein, diese neu zu Beweisen, da die Beweise in Mathlib aus Optimierungsgründe nicht schön mathematisch präsentiert sind.

20 Sylow Sätze (***)

Beweise die Sylow-Sätze

Leite daraus den Satz von Cauchy her: Sei G eine endliche Gruppe und p eine Primzahl, die die Gruppenordnung $\#G$ teilt. Dann existiert ein Element $g \in G$ mit $\text{ord}(g) = p$.

Zeige, dass jede Gruppe der Ordnung 15 zyklisch ist.

Sylow Gruppen haben eine extensive API in Mathlib. Diese zu verwenden, und jenachdem neu zu Beweisen kann doch etwas aufwand benötigen oder nicht ganz trivial sein.

21 Nakayamas Lemma (***)

Leite Nakayamas Lemma her: Sei M ein endlich erzeugtes R-Modul und I ein ideal im Jacobson-Radikal $J(R)$. Dann ist $IM \neq M$

Sei R ein kommutativer Ring und M ein endlich erzeugtes R-Modul. Zeige, dass jeder surjektive Endomorphismus $f : M \to M$ injektiv ist.

Für dieses Projekt könnte es eventuell gewisse Lücken in der Mathlib geben, und man muss eventuell gewisse Terme selber definieren.

22 Legendre Symbol (*)

Zeige die bekannten Rechenregeln zu Legendre Symbolen.

Zeige dass $\sum_{a=0}^{p-1} \left(\frac{a}{p}\right) = 0$, wobei $\left(\frac{a}{p}\right)$ das Legendre-Symbol darstellt.

Dieses Projekt sollte nicht sonderlich anspruchsvoll sein und kann vermutlich einfach erweitert werden auf verwandte Elemente der Zahlentheorie. Das Legendre Symbol und die Rechenregeln sind

mehrheitlich in Mathlib vorhanden und ein Grossteil des Projekts wäre, diese API kennenzulernen, damit zu arbeiten und eine gute Einführung zusammenzustellen.

23 Chinese Remainder Theorem (**)

Arbeite den Beweis des CRT auf zu einer Serie von Aufgaben

Benuzte Chinese Remainder Theorem um verschiedene Kongruenzen zu lösen.

Die Aufarbeitung des Beweises könnte relative aufwendig sein, aber nicht übermässig schwierig, da dieser in der Mathlib existiert. Die Anwendung sollte aber sehr einfach sein und beliebig erweiterbar, da man sich zum Beispiel kompliziertere Kongruenzen anschauen kann. Das Projekt kann sich nach belieben dem beiden Teilen mehr oder weniger widmen.

24 Topologie (**)

Zeige dass $\mathbb{S}^2$ einfach-zusammenhängend ist. Jeder zusammenziehbare Raum ist einfach-zusammenhängend.

Einführung in grundlegende topologische Begriffe, wie z.B. diese Aufgabe. Dieses Projekt ist sehr vage formuliert und hat viel Entfaltungsmöglichkeit.

25 Topologie (***)

Sei $\exp\colon \mathbb{C} \to \mathbb{C} - \{0\}, z \mapsto \sum \frac{z^n}{n!}$. Zeige, dass $(\mathbb{C}, \exp)$ eine Überlagerung von $\mathbb{C} - \{0\}$ ist. Benutze dies, um die Fundamentalgruppe von $\mathbb{C} - \{0\}$ zu berechnen.

Die Aufgabe dürfte schon mathematisch einigermassen ansprucshvoll sein, sowie dies in Lean zu übersetzen. Der Topologieteil von Mathlib ist allerdings gut ausgebaut (wenn auch etwas zu generalisiert), womit dies machbar sein sollte.

4 Hilfsmittel

4.1 Documentation

(1) Zum einen gibt es die Mathlib documentation

https://leanprover-community.github.io/mathlib_docs/index.html

Da könnt ihr nach Lemmas suchen. Die Suche sucht nach substrings, also wenn ihr etwas wie `even n` ↔ ¬ `odd n` sucht, gebt ihr am besten einmal `evenifnotodd` in die Suche ein.

(2) Alternativ gibt es auch noch dieses Suchtool:

http://mathlib-search.edayers.com/?query=Main+Theorem+of+Algebra

Dieses versucht natürliche Sprache zu verarbeiten, also eine Eingabe in Englisch. Die Funktion ist allerdings experimental.

(3) Fragt einfach auf Zulip, entweder im privaten stream "hhu" oder alternative in "new members" oder "is there code for X?".

4.2 Chat

Zulip ist das "Stackoverflow" für Lean. Für Fragen zu den Projekten und Lean, meldet ihr euch am besten auf Zulip an:

https://leanprover.zulipchat.com/#narrow/stream/385218-hhu

(am besten mit echtem Namen)

Da gibt es einen privaten Stream "hhu" (zu dem ich euch hinzufügen muss), wo wir Fragen um die Projekte besprechen können.

Zudem ist der Stream "new members" perfekt um grundlegende Fragen zu Lean an die Community zu Fragen.