1. Installation von Lean4

- Wir verwenden im Kurs den Visual Studio Code-Editor von Microsoft. Dieser ist für alle gängigen Betriebssysteme frei verfügbar. Laden Sie das Programm herunter, installieren Sie es auf Ihrem Rechner und starten Sie es.
- Danach muss noch die Lean 4-Spracherweiterung installiert werden. Wählen Sie den Menüpunkt File/Preferences/Extensions, suchen Sie nach Lean 4 und klicken Sie auf install.

2. Erzeugung eines Standalone-Projekts

Es gibt zwei verschiedene Typen von Projekten, „Standalone"- und „Mathlib"-Projekte. Erstere enthalten lediglich die wichtigsten Basisfunktionen des Lean-Systems, in Letzteres kann man auf eine umfangreiche Bibliothek von Funktionen zugreifen, die viele Teilgebiete der Mathematik abdecken. Auf der Webseite

`https://leanprover-community.github.io/mathlib-overview.html`

kann man sich einen ersten Eindruck darüber verschaffen. Wir werden in diesem Kurs aber fast ausschließlich mit den Basisfunktionen arbeiten, da es Teil unserer Zielsetzung ist, die Struktur und Arbeitsweise von Lean-Programmen von Grund auf zu verstehen und bei der Formulierung mathematischer Inhalte auf wenig Hilfsmittel angewiesen zu sein, um bei der Verwendung des Systems möglichst flexibel zu werden. Um ein „Standalone"-Projekt zum Laufen zu bringen, führen Sie die folgenden Einzelschritte aus.

- Starten Sie den Visual Studio Code-Editor.
- Wählen Sie den Menüpunkt File/New File..., geben Sie `test.lean` ein, und wählen Sie ein Zielverzeichnis für die Datei (zum Beispiel `/home/miller`). Diese Datei kann wieder gelöscht werden, wenn der Einrichtungsvorgang abgeschlossen ist.
- Im Editor öffnet sich die Textdatei `test.lean`, und in der Kopfzeile erscheint ein ∀-Symbol. Klicken Sie dieses Symbol an, und wählen Sie New Project.../Create Standalone Project. Wählen Sie einen Zielordner für das Projekt, zum Beispiel erneut `/home/miller`. Geben Sie als Projektnamen „`st`" ein und klicken Sie auf den Button Create Project Folder.
- Nach wenigen Augenblicken erscheint ein Fenster mit der Aufschrift Open Project Folder. Klicken Sie auf Cancel, und schließen Sie alle Textfenster.
- Wählen Sie den Menüpunkt File/Open Folder. Gehen Sie in das Verzeichnis `/home/miller/st` (oder in das entsprechende Verzeichnis, wenn Sie einen anderen Zielordner gewählt haben). Klicken Sie auf Open, ohne vorher in dem Verzeichnis eine Datei oder einen Unterordner zu wählen.
- In dem Verzeichnis `/home/miller/st/St` befindet sich eine Datei mit dem Namen `Basic.lean`. Öffnen Sie diese Datei mit File/Open File.... Wir können nun zunächst überprüfen, ob der *Lean*-Interpreter korrekt arbeitet. Löschen Sie alle Textzeilen in der Datei, und fügen Sie statt dessen die Zeile `#eval 2+2` ein. Bewegen Sie den Cursor ans rechte Ende dieser Zeile. Im nebenstehenden Infoview-Fenster sollte eine `4` erscheinen.

- Nun überprüfen wir noch, ob die Einbindung anderer LEAN-Dateien problemlos funktioniert. Legen Sie im Verzeichnis `/home/miller/st/St` ein Unterverzeichnis mit dem Namen `sub` an. Wählen Sie erneut den Menüpunkt FILE/NEW FILE..., geben Sie `imptest.lean` als Dateinamen an, und wählen Sie `/home/miller/st/St/sub` als Zielverzeichnis für diese Datei. Es öffnet sich ein Editorfenster für diese Datei. Schreiben Sie in die Datei die Textzeile `def FortyTwo := 42`. Wechseln Sie in die Datei `Basic.lean`, schreiben Sie in die erste Zeile die Anweisung `import St.sub.imptest` und in die letzte Zeile die Anweisung `#eval FortyTwo`. Klicken Sie im rechststehenden INFOVIEW-Fenster auf den Button RESTART FILE und bewegen Sie den Cursor an das Ende der letzten Zeile.
- Wenn nach dem letzten Schritt im Textfenster keine roten Linien zu sehen sind und im INFOVIEW-Fenster der Wert `42` erscheint, dann hat die Einbindung funktioniert.

Wir können nun LEAN-Programme schreiben, die aus einer beliebigen Anzahl einzelner Dateien zusammengesetzt sind. Sollte die Einbindung einmal nicht mehr funktionieren, dann hilft in der Regel der RESTART FILE-Button, der alle Abhängigkeiten zwischen den Dateien aktualisiert. Sollte dies nicht zum Erfolg führen, wählen Sie den Menüpunkt FILE/CLOSE FOLDER, und öffnen Sie das Projektverzeichnis erneut mit FILE/OPEN FOLDER, wie oben beschrieben. Achten Sie dabei auf die Wahl des richtigen Verzeichnisses.

3. Erzeugung eines Mathlib-Projekts

- Öffnen Sie eine Datei mit der Endung „`.lean`“, oder erstellen sie diese neu, wie oben beschrieben.
- In der Kopfzeile ist wiederum das ∀-Symbol zu sehen. Klicken Sie es an, und wählen Sie diesmal NEW PROJECT.../CREATE PROJECT USING MATHLIB. Wählen Sie einen Zielordner für das Projekt, etwa wieder `/home/miller`. Geben Sie als Projektnamen „`ml`“ ein und klicken Sie auf den Button CREATE PROJECT FOLDER.
- Diesmal dauert es erheblich länger, bis die Erstellung des Projektordners abgeschlossen ist, unter anderem weil eine große Menge an Bibliotheksdateien aus dem Internet heruntergeladen werden. Schließen Sie nach Abschluss des Erstellungsvorgangs wiederum alle Textfenster.
- Wählen Sie den Menüpunkt FILE/OPEN FOLDER. Gehen Sie in das Verzeichnis `/home/miller/ml` und klicken Sie auf OPEN. In dem Verzeichnis `/home/miller/ml/Ml` befindet sich wieder eine Datei mit dem Namen `Basic.lean`. Öffnen Sie diese Datei mit FILE/OPEN FILE.... Diesmal soll in erster Linie überprüft werden, ob LEAN die MATHLIB-Dateien korrekt einbindet. Ergänzen Sie oben in der Datei die Zeile `import Mathlib.Data.Real.Basic` und unten die Zeilen

  ```
  def FortyThree : ℝ := 43
  #check FortyThree
  ```

  wobei das Symbol ℝ durch die Eingabe von `\R` erzeugt wird.
- Gehen Sie mit dem Cursor an das Ende der letzten Zeile. Wenn keine roten Linien zu sehen sind und im INFOVIEW-Fenster die Meldung „`FortyThree : ℝ`“ erscheint, hat die Installation funktioniert.

4. Verwendung von Sonderzeichen

Für einen gut lesbaren Code benötigt man den Zugriff auf mathematische Sonderzeichen. Das LEAN-System verwendet dafür sog. UNICODE-Symbole. Dabei handelt es sich um einen internationalen Zeichensatz, der in vielen gängigen Textverarbeitungssystemen zur Verfügung steht. Die Symbole erzeugt man mit Hilfe der Tastenkombination CTRL-SHIFT-U, gefolgt von einer vierstelligen Hexadezimalzahl. (Im Gegensatz zum gewöhnlichen Dezimalsystem besteht das Hexadezimalzahlsystem aus 16 Ziffern, den Zahlen 0 bis 9 gefolgt von den Buchstaben A bis F. Es gilt also die Zuordnung $A \mapsto 10$, $B \mapsto 11$, ..., $E \mapsto 14$, $F \mapsto 15$, $10 \mapsto 16$, $11 \mapsto 17$ usw.)

Weil es sehr mühsam ist, auf diese Weise auf Sonderzeichen zuzugreifen, stellt LEAN eine Vielzahl von Befehlen zur Verfügung, mit denen man solche Zeichen erzeugen kann. Dabei orientieren sich die Befehle weitgehend an der Syntax des wissenschaftlichen Schreibprogramms LATEX. So erzeugt der Befehl `\alpha` beispielsweise den griechischen Buchstaben α, und mit `\forall` erhält man den Allquantor $\forall$. Eine Überblick über die gesamte vorhandene Notation erhält man, indem man den Menüpunkt $\cdots$/DOCUMENTATION/DOCS:SHOWUNICODEINPUTABBREVATIONS anwählt.

Es ist auch möglich, eigene Abkürzungsbefehle zu definieren. Dazu wählt man den Menüpunkt FILE/PREFERENCES/SETTINGS und gibt in das Textfeld das Suchwort „Custom Translations“ ein. Klicken Sie auf den Punkt „`Edit in settings.json`“. Es öffnet sich ein Textfenster, und in dem enthaltenen Text ist die Zeile

```
"lean4.input.customTranslations": { }
```

zu sehen. Innerhalb der Mengenklammern kann man nun seine eigenen Befehlsdefinitionen eingeben, getrennt durch Kommata. Beispielsweise definiert man durch die Zeile

```
"threequarters": ¾
```

einen Befehl, der das Symbol für den Bruch $\frac{3}{4}$ erzeugt. Bei der Eingabe der Zeile in das Textfenster verwendet man für das Bruchsymbol die Tastenkombination CTRL-SHIFT-U, gefolgt von der Hexadezimalzahl `00BE`. Die vierstelligen Zahlen für beliebige mathematische oder nicht-mathematische Unicode-Zeichen sind an vielen Stellen im Internet zu finden, beispielsweise auf der Seite `www.unicodeplus.com`.