Mathematisches Institut
Heinrich-Heine Universität Düsseldorf
Dr. Alexander Bentkamp, Jon Eugster

WiSe 2023/24
21. November 2023

Computer-gestützte Beweisführung

Übungsblatt 5

In allen Aufgaben meinen wir mit „beweisen“ die Erstellung formaler Beweisterme im Sinne von Abschnitt 3.5 des Skripts. Zur besseren Klausurvorbereitung empfehlen wir, diese Übungen ohne Lean zu bearbeiten.

Aufgabe 1. Beweisen Sie:

(a)

$$\textbf{theorem}\ \ \mathsf{true\_iff\_not\_false} : \mathsf{True} \leftrightarrow \neg\mathsf{False} := \ldots$$

Tipp: Nutzen Sie Iff.intro und True.intro, die in der Vorlesung eingeführt wurden. Beachten Sie, dass $\neg s$ definitionell gleich $s \to \mathsf{False}$ ist: Jede Funktion von s nach False ist also ein Element von $\neg s$.

(b)

$$\textbf{theorem}\ \ \mathsf{and\_false}\ (p : \mathsf{Prop}) : (p \wedge \mathsf{False}) \leftrightarrow \mathsf{False} := \ldots$$

Tipp: Hier könnten Iff.intro, And.right und False.elim helfen. Beachten Sie, dass Sie mit einem Beweis $h : \mathsf{False}$ einen Beweis $\mathsf{False.elim}\ h : q$ (genauer $@\mathsf{False.elim}\ q\ h : q$) einer beliebigen Aussage q konstruieren können.

Lösung.

(a)

$$\begin{aligned}&\textbf{theorem}\ \ \mathsf{true\_iff\_not\_false} : \mathsf{True} \leftrightarrow \neg\mathsf{False} :=\\ &\mathsf{Iff.intro}\\ &\quad(\lambda(ht : \mathsf{True})\ (hf : \mathsf{False}).\ hf)\\ &\quad(\lambda(hnf : \neg\mathsf{False}).\ \mathsf{True.intro})\end{aligned}$$

(b)

$$\begin{aligned}&\textbf{theorem}\ \ \mathsf{and\_false}\ (p : \mathsf{Prop}) : (p \wedge \mathsf{False}) \leftrightarrow \mathsf{False} :=\\ &\quad\mathsf{Iff.intro}\ (\lambda h : p \wedge \mathsf{False}.\ \mathsf{And.right}\ h)\ (\lambda h : \mathsf{False}.\ \mathsf{False.elim}\ h)\end{aligned}$$

Aufgabe 2. Beweisen Sie:

Bitte wenden!

(a)

$$\textbf{theorem}\ \ \mathsf{forall\_implies}\ (p\ q : \mathsf{Nat} \to \mathsf{Prop}) :$$
$$(\forall x.\ p\ x \to q\ x) \to (\forall x.\ p\ x) \to (\forall x.\ q\ x) := \ldots$$

Tipp: Am Typ von p und q können wir erkennen, dass alle Variablen x in dieser Aufgabe vom Typ Nat sein sollen. Beachten Sie, dass $\forall$ nur eine alternative Schreibweise für Π ist: Ein Element von $\forall x : \mathsf{Nat}.\ s$ ist also eine Funktion von Nat nach s.

(b)

$$\textbf{theorem}\ \ \mathsf{not\_exists\_not\_of\_forall}\ (p : \mathsf{Nat} \to \mathsf{Prop}) :$$
$$(\forall x.\ p\ x) \to \neg(\exists x.\ \neg\ p\ x) := \ldots$$

Tipp: Nutzen Sie $\mathsf{Exists.elim}$.

Lösung.

(a)

$$\textbf{theorem}\ \ \mathsf{forall\_implies}\ (p\ q : \mathsf{Nat} \to \mathsf{Prop}) :$$
$$(\forall x.\ p\ x \to q\ x) \to (\forall x.\ p\ x) \to (\forall x.\ q\ x) :=$$
$$\lambda(h : \forall x.\ p\ x \to q\ x)\ (hp : \forall x.\ p\ x)\ (x : \mathsf{Nat}).\ h\ x\ (hp\ x)$$

(b)

$$\textbf{theorem}\ \ \mathsf{not\_exists\_not\_of\_forall}\ (p : \mathsf{Nat} \to \mathsf{Prop}) :$$
$$(\forall x.\ p\ x) \to \neg(\exists x.\ \neg\ p\ x) :=$$
$$\lambda(h : \forall x.\ p\ x)\ (hex : \exists x.\ \neg\ p\ x).$$
$$\mathsf{Exists.elim}\ hex\ (\lambda(x : \mathsf{Nat})\ (hx : \neg\ p\ x).\ hx\ (h\ x))$$

Aufgabe 3. Wir definieren ein induktives Prädikat Even von geraden Zahlen. Es sollen also $\mathsf{Even}\ 0$, $\mathsf{Even}\ 2$, $\mathsf{Even}\ 4$, … beweisbar sein, aber nicht $\mathsf{Even}\ 1$, $\mathsf{Even}\ 3$, $\mathsf{Even}\ 5$, ….

(a) Ergänzen Sie den fehlenden Konstruktor „???“:

$$\textbf{inductive}\ \mathsf{Even} : \mathsf{Nat} \to \mathsf{Prop} :=$$
$$\mid\ ???$$
$$\mid\ \mathsf{step} : \forall(n : \mathsf{Nat}).\ \mathsf{Even}\ n \to \mathsf{Even}\ (\mathsf{succ}\ (\mathsf{succ}\ n))$$

(b) Beweisen Sie, dass 4 gerade ist:

$$\textbf{theorem}\ \ \mathsf{even\_four} : \mathsf{Even}\ (\mathsf{succ}\ (\mathsf{succ}\ (\mathsf{succ}\ (\mathsf{succ}\ \mathsf{zero})))) := \ldots$$

Lösung.

(a)

$$\begin{aligned}&\textbf{inductive}\ \mathsf{Even} : \mathsf{Nat} \to \mathsf{Prop} :=\\ &\mid \mathsf{base} : \mathsf{Even}\ 0\\ &\mid \mathsf{step} : \forall (n : \mathsf{Nat}).\ \mathsf{Even}\ n \to \mathsf{Even}\ (\mathsf{succ}\ (\mathsf{succ}\ n))\end{aligned}$$

(b)

$$\begin{aligned}&\textbf{theorem}\ \ \mathsf{even\_four} : \mathsf{Even}\ (\mathsf{succ}\ (\mathsf{succ}\ (\mathsf{succ}\ (\mathsf{succ}\ \mathsf{zero})))) :=\\ &\mathsf{Even.step}\ 2\ (\mathsf{Even.step}\ 0\ \mathsf{Even.base})\end{aligned}$$

Aufgabe 4. Auf einer Insel sind alle Einwohner entweder Ritter, die immer die Wahrheit sagen, oder Knappen, die immer lügen. Ein Besucher trifft die Einwohner Martin und Klaus.

Klaus sagt: „Martin ist ein Knappe.“

Martin sagt: „Klaus und ich sind Ritter.“

Wir wollen formal beweisen, dass Martin ein Knappe ist.

Sei dazu $\mathsf{Einwohner} : \mathsf{Type}$ ein Typ der Einwohner der Insel und seien $\mathsf{klaus} : \mathsf{Einwohner}$ und $\mathsf{martin} : \mathsf{Einwohner}$. Sei $\mathsf{ritter} : \mathsf{Einwohner} \to \mathsf{Prop}$ eine Funktion, die angibt, ob ein Einwohner ein Ritter ist. Da jeder Einwohner entweder Ritter oder Knappe ist, können wir zur Vereinfachung „$\neg\ \mathsf{ritter}\ P$“ für „P ist ein Knappe“ schreiben. Einen Hinweis „P sagt: X“ können wir kodieren als: „$\mathsf{ritter}\ P \leftrightarrow X$“.

Formal können wir diese Aufgabe also wie folgt fassen. Ergänzen Sie den Beweis.

$$\begin{aligned}&\textbf{theorem}\ \ \mathsf{raetsel1}\\ &\quad (h_1 : \mathsf{ritter\ klaus} \leftrightarrow \neg\mathsf{ritter\ martin})\\ &\quad (h_2 : \mathsf{ritter\ martin} \leftrightarrow (\mathsf{ritter\ martin} \wedge \mathsf{ritter\ klaus})) :\\ &\neg\ \mathsf{ritter\ martin} := \ldots\end{aligned}$$

Hierbei kodiert h_1 den Hinweis „Klaus sagt: ‚Martin ist ein Knappe.‘“ und h_2 den Hinweis „Martin sagt: ‚Klaus und ich sind Ritter.‘“

Lösung.

$$\begin{aligned}&\textbf{theorem}\ \ \mathsf{raetsel1}\\ &\quad (h_1 : \mathsf{ritter\ klaus} \leftrightarrow \neg\mathsf{ritter\ martin})\\ &\quad (h_2 : \mathsf{ritter\ martin} \leftrightarrow (\mathsf{ritter\ martin} \wedge \mathsf{ritter\ klaus})) :\\ &\neg\ \mathsf{ritter\ martin} := \lambda hrm.\ \mathsf{Iff.mp}\ h_1\ (\mathsf{And.right}\ (\mathsf{Iff.mp}\ h_2\ hrm))\ hrm\end{aligned}$$

Abgabe: 28. November 2023, bis 16.30 Uhr auf Ilias
(Oder bei technischen Problemen per Email an `jon.eugster@hhu.de`)
Abgabe zu zweit ist erlaubt.