GLoIn-Weihnachtsblatt

Zur Vorlesung *Grundlagen der Logik in der Informatik* (WS 2023/24)
Update: A2, A3 jetzt bepunktet. Abgabe: 19. März 2024

Aktualisierte Hinweise:

- Zweck dieses Bonusblatts ist es, Coq und Induktion besser zu verstehen.
- Aufgabe A2 und A3 sind jetzt bepunktet. (A1 bleibt unbepunktet).
- Für die volle Punktzahl muss der abgegebene Code lesbar sein! Nutzen Sie zu diesem Zweck Listen (mittels `-`, `*`, `+`) um Induktions- und Äquivalenzbeweise in ihre Fälle zu untergliedern (so wie es in der Lösung von `semantik_entscheidbar` in `structinduct.v` und in der Lösung der P1 auf Blatt 6 gemacht wird).
- Abgabe: Einzeln unter Angabe der RRZE-Kennung (`ab12defg`) per E-Mail an:

  `cs8-gloin-tutoren@lists.fau.de`

Aufgabe A1 Curry-Howard-Korrespondenz (0 Punkte)

Coq und viele andere Theorembeweiser arbeiten intern mit der *Curry-Howard-Korrespondenz*, die besagt, dass Beweisen das gleiche wie (funktionales) Programmieren ist. Wenn Sie in Coq einen Beweis mittels Taktiken aufschreiben, wird intern ein Programm konstruiert, das man genauso gut auch manuell direkt als Beweis aufschreiben kann:

Grundidee ist, dass `A -> B` nicht nur als *Implikation* verstanden werden kann, sondern in Coq tatsächlich *Funktionen/Programme die einen Parameter vom Typ `A` erwarten und Rückgabewert `B` haben*, ähnlich wie die Menge $A \to B$ aller Funktionen von A nach B. Um eine Funktion von `A` nach `B` in Coq zu definieren, schreibt man

```
(fun (a:A) ⇒ ... (* hier etwas vom Typ B *) ... )
```

Beispielsweise können wir wie folgt die Identitätsfunktion definieren:

```
Parameters A B C D : Prop.
Definition def_id : A → A := (fun a ⇒ a). (* aus dem Kontext ist 'a : A' klar. *)
```

Da der Funktionstyp selber nur ein Typ ist, können wir Funktionen auch schachteln:

```
Definition const : A → (B → A) := (fun a ⇒ (fun b ⇒ a)).
```

Diese Funktion beweist die Implikation $A \to (B \to A)$, oder in Coq:

```
Theorem thm_const : A → (B → A).
Proof. exact const. Qed.
```

Wenn andersherum eine Funktion `f: A→ B` und ein `a` vom Typ `A` gegeben ist, dann ist die *Funktionsanwendung* `f a` vom Typ `B`. Konjunktion, Disjunktion und $\bot$ sind *Datentypen*, die die entsprechenden Intro- und Eliminationsregeln als Funktionen mitbringen. Detailierte Erklärungen finden Sie in der in der Coq-Vorlage `curryhoward.v`.

Aufgabe: Beweisen Sie die folgenden Formeln als Funktion (ohne Verwendung von `apply NNPP` oder ähnlichem) und ergänzen Sie sie in `curryhoward.v`:

(a) $(A \vee B \to C) \to (A \to C) \wedge (B \to C)$

(b) $(A \to (B \to C)) \to (A \wedge B \to C)$

(c) $\neg\neg(A \vee \neg A)$

(d) $\neg\neg A \wedge \neg\neg B \to \neg\neg(A \wedge B)$

Aufgabe A2 Vollständige Induktion in Coq (5 Punkte)

Beweisen Sie Aufgabe A4 vom Übungsblatt 3 in Coq, indem Sie in `induction.v` alle fehlenden Beweise einfügen. In der Aufgabe damals hatten wir für $A \in \mathscr{A}$ die Funktion $\phi\colon \mathbb{N} \to \mathscr{F}$ rekursiv definiert:

$$\phi(0) \;=\; A \to B \qquad \phi(k+1) \;=\; \phi(k) \to A$$

Zeigen Sie in Coq, dass die Formel für gerade $k > 0$ gültig ist und dass sie für ungerade k äquivalent zu A ist. Für die Induktion benötigen Sie lediglich die Taktiken `induction` und `simpl`. Weitere Erklärungen dazu finden Sie in den Kommentaren im verlinkten Coq-Quellcode.

Aufgabe A3 Strukturelle Induktion in Coq (5 Punkte)

In `structinduct.v` finden Sie Erklärungen zur strukturellen Induktion in Coq. Um Induktion über Formeln führen zu können, müssen wir erst die Menge der Formeln ($\mathscr{F}$) und deren Semantik ($\models$) innerhalb von Coq nachbauen. Fügen Sie in `structinduct.v` alle fehlenden Induktionsbeweise ein:

- Wenn eine Formel nur aus Atomen und $\wedge$ aufgebaut ist, dann wird sie von der konstant wahren Wahrheitsbelegung erfüllt.
- Die Semantik ist monoton (Aufgabe A5 von Übungsblatt 7).