Anleitung für Beweise

Diese Beweise werden mit dem Tool cyp[1] getestet. Ein Beweisdatei besteht aus einem oder mehreren Lemmas. Ein Lemma hat grundsätzlich folgenden Aufbau:

`Lemma` *`<Name>`*`:` *`<Aussage>`*

`<Beweis>`

Ein Beweis kann mit verschiedenen Beweismethoden durchgeführt werden, die im Folgenden erläutert werden.

Zunächst einige allgemeine Syntaxhinweise:

- Kommentare werden mit `--` eingeleitet.
- Als Gleichheitszeichen muss `.=.` verwendet werden
- Groß-/Kleinschreibung ist relevant
- Der Lemmaname kann weggelassen werden, wenn Sie das Lemma nicht in einem späteren Lemma verwenden möchten.

Gleichungskette(n)

So könnte ein Beweis für die Aussage `length ([] ++ ys) .=. length [] + length ys` aussehen:

```
Proof
                        length ([] ++ ys)
    (by def ++)         .=. length ys

                        length [] + length ys
    (by def length)     .=. 0 + length ys
    (by arith)          .=. length ys

QED
```

Syntaxhinweise:

- Jeden Schritt in eigene Zeile
- Gleichheitszeichen + Regel am Anfang
- Jeder Schritt muss mit `by Regel` gerechtfertigt werden
- Die `Regel` kann sein:
  - `def f` für eine Funktion `f`
  - Der Name eines beliebigen Axioms. Diese werden explizit im Übungsplatz genannt.
  - Der Name einer Induktionshypothese oder ähnliches. Mehr dazu später.

[1] `https://github.com/noschinl/cyp`

Beweis per Extensionalität

So könnte ein Beweis für die Aussage `f = g` aussehen:

```
Proof by extensionality with x

    To show: f x .=. g x

    <Beweis>

QED
```

Beweis per Induktion

So könnte ein Beweis für die Aussage `length (xs ++ ys) .=. length xs + length ys` aussehen:

```
Proof by induction on List xs

Case []

    To show: length ([] ++ ys) .=. length [] + length ys

    <Beweis>

Case x:xs

    To show: length ((x:xs) ++ ys) .=. length (x:xs) + length ys
    IH: length (xs ++ ys) .=. length xs + length ys

    <Beweis>

QED
```

Syntaxhinweise:

- Statt `IH` kann ein beliebiger Name angegeben werden.
- Das Lemma `IH` kann dann in dem Beweis verwendet werden.

Beweis per Fallunterscheidung

```
Proof by case analysis on Bool p x

Case True

    Assumption: p x .=. True

    <Beweis>

Case False

    Assumption: p x .=. False

    <Beweis>

QED
```

Syntaxhinweise:

- Statt `Assumption` kann ein beliebiger Name angegeben werden.
- Das Lemma `Assumption` kann dann in dem Beweis verwendet werden.

Vollständiges Beispiel für einen Induktionsbeweis

```
Lemma: length(xs ++ ys) .=. length xs + length ys

Proof by induction on List xs

Case []

    To show: length ([] ++ ys) .=. length [] + length ys

    Proof

                            length ([] ++ ys)
        (by def ++)         .=. length ys

                            length [] + length ys
        (by def length)     .=. 0 + length ys
        (by arith)          .=. length ys

    QED

Case x:xs

    To show: length ((x:xs) ++ ys) .=. length (x:xs) + length ys
    IH: length (xs ++ ys) .=. length xs + length ys

    Proof

                            length ((x : xs) ++ ys)
        (by def ++)         .=. length (x : (xs ++ ys))
        (by def length)     .=. 1 + length (xs ++ ys)
        (by IH)             .=. 1 + (length xs + length ys)
        (by arith)          .=. 1 + length xs + length ys

                            length (x : xs) + length ys
        (by def length)     .=. 1 + length xs + length ys

    QED

QED
```