Coq Cheatsheet

1 Syntax

Wahrheitswerte wie `True`, `False` haben den Typ `Prop`.

Ein Prädikat ist eine Funktion, die nach `Prop` abbildet, z.B.

```
Definition relation : X → X → Prop.
```

ist eine Typdefinition binärer Prädikate (Relationen auf X). Eine konkrete Relation ist z.B.:

```
Definition id (x y: X) : Prop := x = y.
```

Zu lesen als: `id x y`, wobei x und y vom Typ `X` sind, ist vom Typ `Prop` und ist wahr (also äquivalent zu `True`) genau dann, wenn `x = y` gilt.

Konjunktion	$A \wedge B$	`A /\ B`
Disjunktion	$A \vee B$	`A \/ B`
Implikation	$A \to B$	`A -> B`
Biimplikation	$A \leftrightarrow B$	`A <-> B`
Negation	$\neg A$	`~ A` oder `A -> False`
Allquantor	$\forall x \in X(p(x))$ $\forall x \in X, y \in X, z \in Z(p(x,y,z))$	`forall x : X, p x` `forall (x y :X) (z : Z), p x y z`
Existenzquantor	$\exists x \in X(p(x))$	`exists x : X, px`

Eine(zu beweisende) Aussage wird mit `Lemma` oder `Theorem` eingeleitet. Das Lemma

```
Lemma involution (R : relation) : forall x y : X, (inv (inv R)) x y ↔ R x y.
Proof.
...
Qed.
```

mit Namen `involution` besagt, dass für beliebige Relationen `R` gilt, dass `inv (inv R) = R`, extensiv über die Paare der Relation `R` definiert: $(x, y) \in$ `inv (inv R)` $\Leftrightarrow (x, y) \in$ `R`. Nach `Proof.` ist der Beweis zu führen (siehe Taktiken) der mit `Qed.` zu beenden ist.

2 Taktiken

Taktik	Ergebnis
`assert (A = B).`	Unterbeweis für $A = B$ öffnen
`unfold id, inv.`	Definition von `id` und `inv` im Ziel einsetzen
`unfold id, inv in H.`	Definition von `id` und `inv` in Hypothese `H` einsetzen

Ziel	Taktik	Ergebnis
`A /\ B`	`split.`	zwei Beweisziele: `A` und `B`
`A <-> B`	`split.`	zwei Beweisziele: `A -> B` und `B -> A`
`A \/ B`	`left.` oder `right.`	Ziel: `A` (`left`) oder `B` (`right`)
`forall x : X, A`	`intro.` oder `intro x.`	Kontext: `x : X`, Ziel: `A`
`forall x y : X, A`	`intros.` oder `intros x0 y0.`	Kontext: `x0, y0 : X`, Ziel: `A`
`A -> B`	`intro.` oder `intro H.`	Kontext: `H: A`, Ziel: `B`
`A -> B -> C`	`intros.` oder `intros H1 H2.`	Kontext: `H1: A` und `H2: B`, Ziel: `C`
`~ A`	`intro.` oder `intro H.`	Kontext: `H: A`, Ziel: `False`

Die Argumente von `intro` und `intros` sind frei wählbar; wenn weggelassen, generiert `Coq` Bezeichnungen automatisch.

Kontext	Taktik	Ergebnis
`H: A /\ B`	`destruct H.` oder `destruct H as [H1 H2].`	Kontext: `H1: A` und `H2: B`
`H: A \/ B`	`destruct H.` oder `destruct H as [H1 | H2].`	Beweisziel 1 mit Kontext `H1: A` Beweisziel 2 mit Kontext `H2: B`
`H: exists x : X, A`	`destruct H.` oder `destruct H as [x' H1].`	Kontext: `x' : X` und `H1: A`
`H: False` oder `H1: A` und `H2: ~ A`	`contradiction.`	Beweisziel gelöst
`H: A -> B` und `H1: A`	`apply H in H1.`	Kontext: `H1: B`

`H: forall x : X,`
`A x -> B x`
und `H1: A y`
	`apply H in H1.`	Kontext: `H1: B y`

Die Argumente von `destruct` in eckigen Klammern nach dem Schlüsselwort `as` sind frei wählbar.

Kontext	Ziel	Taktik	Ergebnis
`H: A`	`A`	`apply H.` oder `assumption.`	Ziel gelöst
`H: A -> B`	`B`	`apply H.`	Ziel: `A`
`H: A <-> B`	`B`	`apply H.`	Ziel: `A`
`H: A <-> B`	`A`	`apply H.`	Ziel: `B`
`H: forall x : X, A x -> B x`	`B y`	`apply H.`	Ziel: `A y`
`H: forall x : X, A x <-> B x`	`B y`	`apply H.`	Ziel: `A y`
`H: forall x : X, A x <-> B x`	`A y`	`apply H.`	Ziel: `B y`
`z : X`	`exists x : X, A x`	`exists z.`	Ziel: `A z`

Kontext	Taktik	Ergebnis
`H: A = B`	`rewrite` $\rightarrow$ `H.`	Ziel: Alle freien `A` werden durch `B` ersetzt
	`rewrite` $\leftarrow$ `H.`	Ziel: Alle freien `B` werden durch `A` ersetzt
	`rewrite` $\rightarrow$ `H in H1.`	Hypothese `H1`: Alle freien `A` werden durch `B` ersetzt
	`rewrite` $\leftarrow$ `H in H1.`	Hypothese `H1`: Alle freien `B` werden durch `A` ersetzt

Ziel	Taktik	Ergebnis
`A = A`	`reflexivity.`	Beweisziel gelöst