

Natürliches Schließen in Coq

Ein einführendes Tutorial

Burkhardt Renz

Technische Hochschule Mittelhessen,
Fachbereich MNI,
Wiesenstr. 14, D-35390 Gießen
`Burkhardt.Renz@nmi.thm.de`

Rev 2.0 – 22. Januar 2019

Inhaltsverzeichnis

1	Das Beweissystem des natürlichen Schließens	2
1.1	Implikation	3
1.2	Konjunktion	4
1.3	Disjunktion	4
1.4	Negation	5
1.5	Allquantor	5
1.6	Existenzquantor	6
1.7	Gleichheit	6
1.8	Beispiele	6
1.9	Weitere Regeln für die klassische Logik	8
2	Natürliches Schließen in Coq	8
2.1	Implikation	9
2.2	Konjunktion	13
2.3	Disjunktion	17
2.4	Negation	22
2.5	Allquantor	25
2.6	Existenzquantor	27
2.7	Gleichheit	29
2.8	Verallgemeinerung der bewiesenen Aussagen	30
2.9	Beispiele	32
2.10	Charakterisierungen der klassischen Logik	34
	Literaturverzeichnis	38

Dieses Tutorial stellt das Beweissystem des natürlichen Schließens in Coq vor — einführend und an Beispielen erläuternd.

Der erste Teil des Tutorials behandelt das natürliche Schließen als Beweissystem im Allgemeinen. Es geht zurück auf die grundlegende Arbeit „Untersuchungen über das logische Schließen“ von Gerhard Gentzen [Gen35]. Gut verständliche Einführungen in das Thema findet man in „Logic in Computer Science“ von Michael Huth und Mark Ryan [HR04] sowie in „Proof and Disproof in Formal Logic“ von Richard Bornat [Bor05]. Richard Bornat hat mit **Jape** auch ein Programm entwickelt, in dem man interaktiv Beweise im natürlichen Schließen entwickeln (und überprüfen) kann. Ein ähnliches Programm habe ich zusammen mit Studierenden der Technischen Hochschule Mittelhessen in der Programmiersprache Clojure entwickelt. Eine knappe Anleitung findet man auf dem **Wiki zur Logic Workbench**. (Dort wird auch erläutert, wie man die Logic Workbench als Plugin im Editor Atom verwenden kann. Die Beispiele in 1.8 können mit Jape oder in der Logic Workbench nachvollzogen werden.)

Im zweiten Teil verwenden wir den Beweisassistenten **Coq**, um Beweise im natürlichen Schließen zu machen. Coq ist ein mächtiges Instrument und wir kratzen mit dieser Einführung gewissermaßen nur an der Oberfläche. Aber vielleicht ist dies ein guter Einstieg, um sich mit Coq zu befassen. Ein grundlegendes Buch über Coq ist „Interactive Theorem Proving and Program Development“ von Yves Bertot und Pierre Castéran [BC04], insbesondere Kapitel 5 „Everyday Logic“. In diesem Tutorial werden wir die Techniken von Coq recht informell einführen und benutzen. Präzise Definitionen der verwendeten Kommandos und Taktiken findet man im eben genannten Buch, aber auch detaillierter im **Reference Manual** von Coq.

Die Beispiele im zweiten Teil des Tutorials wurden natürlich in Coq selbst entwickelt. Den Quelltext findet man auf meiner **Webseite**. Das Tutorial wurde mit `coq-tex` überprüft, einem Filter, der in \LaTeX eingebetteten Coq-Code evaluiert.

1 Das Beweissystem des natürlichen Schließens

Das Beweissystem des natürlichen Schließens für die Aussagen- und Prädikatenlogik mit Gleichheit ist ein formales Kalkül, das Herleitungen von Formeln durch die Anwendung von vorgegebenen Schlussregeln erlaubt. Das Kalkül des natürlichen Schließens wurde 1934 von Gerhard Gentzen¹ und unabhängig von ihm von Stanisław Jaśkowski² entwickelt.

¹ Gerhard Gentzen (1909–1945), deutscher Mathematiker und Logiker, siehe **Wikipedia über Gerhard Gentzen** und [Gen35].

² Stanisław Jaśkowski (1906–1965), polnischer Logiker, siehe **Wikipedia über Stanisław Jaśkowski**.

Die Bezeichnung „Natürliches Schließen“ (auch „Natürliche Deduktion“) rührt daher, dass die Regeln des Kalküls das „natürliche“ Argumentieren von Mathematikern formalisieren.

„Mein erster Gesichtspunkt war folgender: Die Formalisierung des logischen Schließens, wie sie insbesondere durch Frege, Russell und Hilbert entwickelt worden ist, entfernt sich ziemlich weit von der Art des Schließens, wie sie in Wirklichkeit bei mathematischen Beweisen geübt wird. [...] Ich wollte nun zunächst einmal einen Formalismus aufstellen, der dem wirklichen Schließen möglichst nahekommt. So ergab sich ein ‚Kalkül des natürlichen Schließens‘.“[Gen35, S. 176].

In der formalen Sprache der Aussagenlogik verwendet man üblicherweise die Junktoren \rightarrow für die Implikation, \wedge für die Konjunktion, \vee für die Disjunktion und \neg für die Negation. In der Prädikatenlogik mit Gleichheit hat man außerdem die Symbole $\forall x$ für den Allquantor, $\exists x$ für den Existenzquantor sowie $=$ für die Gleichheit von Termen, die Objekte des Universum bezeichnen.

Die Regeln des natürlichen Schließens geben an, wie man eine Formeln unter bestimmten Gegebenheiten syntaktisch umformen darf. Für das natürliche Schließen in der intuitionistischen Logik kann man dadurch eine Semantik definieren: Alle Formeln, die sich mittels der Regeln beweisen lassen, sind wahr. In der klassischen Logik definiert man die Semantik durch die Gültigkeit von Formeln in allen Modellen der Sprache. Die Regeln sind natürlich so gemacht, dass sie in Bezug auf dieses Definition der Semantik Wahrheit erhalten.

Für die Herleitung von Formeln gibt es im natürlichen Schließen pro logischem Symbol zwei Regeln:

- eine, die das Symbol einführt (*Introduction*, abgekürzt durch *i*) und
- eine zweite, die das Symbol entfernt (*Elimination*, abgekürzt durch *e*).

Im Folgenden werden die Regeln des natürlichen Schließens für die intuitionistische Logik vorgestellt. Die Symbole ϕ , ψ und χ sind Symbole der Metasprache, sie stehen für beliebige Formeln.

Jede Regel gibt an, was *gegeben* sein muss (oberhalb des Strichs), damit die Umformung gemacht werden darf, also was sich aus dem Gegebenen *ergibt* (unterhalb des Strichs).

1.1 Implikation

Die Regeln für die Implikation sind:

	<i>Einführung</i>	<i>Elimination</i>
\rightarrow	$\frac{\boxed{\begin{array}{c} \phi \\ \vdots \\ \psi \end{array}}}{\phi \rightarrow \psi} \rightarrow \text{i}$	$\frac{\phi \quad \phi \rightarrow \psi}{\psi} \rightarrow \text{e, MP}$

Die Implikation leitet man her, indem man die Hypothese als gegeben annimmt und dann daraus die Folgerung herleitet. In der Regel wird in der Box oberhalb des Strichs angegeben, dass ϕ nur *innerhalb* der Box als gegeben angenommen werden darf. Die senkrechten Punkte \vdots markieren die Beweisverpflichtung, nämlich dass sie durch einen Beweis ersetzt werden müssen, der ψ aus ϕ herleitet.

Die Implikation kann man entfernen, wenn man die Hypothese ϕ bewiesen hat und ebenso, dass $\phi \rightarrow \psi$ gilt. Dann hat man ψ bewiesen. Diese Schlussfigur ist schon seit der Antike geläufig und wird als *Modus ponens* bezeichnet, deshalb auch die Abkürzung MP.

1.2 Konjunktion

	<i>Einführung</i>	<i>Elimination</i>
\wedge	$\frac{\phi \quad \psi}{\phi \wedge \psi} \wedge \text{i}$	$\frac{\phi \wedge \psi}{\phi} \wedge \text{e}_1 \quad \frac{\phi \wedge \psi}{\psi} \wedge \text{e}_2$

Die Konjunktion kann man einführen, wenn man Herleitungen für die beiden Formeln der Konjunktion bereits hat.

Für die Elimination der Konjunktion gibt es zwei Subregeln: Eine Herleitung der Gesamtformel der Konjunktion kann man sowohl als Herleitung der linken Teilformel als auch der rechten Teilformel nehmen.

1.3 Disjunktion

	<i>Einführung</i>	<i>Elimination</i>
\vee	$\frac{\phi}{\phi \vee \psi} \vee \text{i}_1 \quad \frac{\psi}{\phi \vee \psi} \vee \text{i}_2$	$\frac{\phi \vee \psi \quad \boxed{\begin{array}{c} \phi \\ \vdots \\ \chi \end{array}} \quad \boxed{\begin{array}{c} \psi \\ \vdots \\ \chi \end{array}}}{\chi} \vee \text{e}$

Wenn man eine Herleitung für ϕ hat, hat man auch eine Herleitung für $\phi \vee \psi$, ebenso darf man die Herleitung von ψ als Beweis für $\phi \vee \psi$ nehmen.

Will man die Konjunktion entfernen und dabei χ herleiten, muss man für jede Teilformel der Konjunktion eine Herleitung von χ finden. Diese Regel entspricht also der Beweistechnik der Fallunterscheidung.

1.4 Negation

	<i>Einführung</i>	<i>Elimination</i>
\neg	$\frac{\boxed{\begin{array}{c} \phi \\ \vdots \\ \perp \end{array}}}{\neg\phi} \quad \neg\text{i}$	$\frac{\phi \quad \neg\phi}{\psi} \quad \neg\text{e, EFQ}$

Will man beweisen, dass $\neg\phi$ gilt — also die Negation einführen —, nimmt man an, dass ϕ bewiesen ist und führt diesen Beweis dann fort, bis man den Widerspruch \perp hergeleitet hat. Daraus ergibt sich, dass $\neg\phi$ bewiesen ist.

Hat man sowohl eine Herleitung für ϕ als auch eine für $\neg\phi$, dann hat man den Widerspruch bewiesen, kann daraus eine beliebige Formel folgern und hat die Negation entfernt. Dass aus dem Widerspruch jede beliebige Aussage folgt, wird auch als *Ex falso quodlibet* oder genauer *Ex falso sequitur quodlibet* bezeichnet. Oft wird die Regel zerlegt in zwei Regeln:

$$\frac{\phi \quad \neg\phi}{\perp} \quad \text{und} \quad \frac{\perp}{\phi}$$

Die bisher diskutierten Regeln gelten für die intuitionistische Aussagenlogik. Für die Prädikatenlogik mit Gleichheit kommen noch die folgenden Regeln hinzu.

1.5 Allquantor

	<i>Einführung</i>	<i>Elimination</i>
\forall	$\frac{\boxed{\begin{array}{c} x_0 \\ \vdots \\ \phi[x_0/x] \end{array}}}{\forall x\phi} \quad \forall\text{x i}$	$\frac{\forall x\phi}{\phi[t/x]} \quad \forall\text{x e}$

Um den Allquantor einzuführen, hat man folgende Beweisverpflichtung: Gegeben sei ein beliebiges Objekt x_0 des Universums. Man muss dann zeigen, dass die Formel ϕ mit x_0 an Stelle der Variablen x gilt (dies schreibt man kurz als $\phi[x_0/x]$). Dabei darf in dieser Herleitung keinerlei spezielle Eigenschaft von x_0 vorkommen, denn x_0 steht ja für ein *beliebiges* Objekt des Universums. Man sagt auch, dass x_0 ein *frisches* beliebiges Objekt ist, sein Name darf somit nicht außerhalb der Box vorkommen.

Die Entfernung des Allquantors ist ein naheliegender Schritt: Wenn ϕ für alle x gilt, dann kann man ein beliebiges konkretes t des Universums an Stelle von x in die Formel ϕ einsetzen.

1.6 Existenzquantor

	<i>Einführung</i>	<i>Elimination</i>
\exists	$\frac{\phi[t/x]}{\exists x \phi} \quad \exists x \text{ i}$	$\frac{\exists x \phi \quad \boxed{\begin{array}{c} x_0 \quad \phi[x_0/x] \\ \vdots \\ \chi \end{array}}}{\chi} \quad \exists x \text{ e}$

Den Existenzquantor kann man einführen, indem man einen *Zeugen* vorweist: Gilt ϕ mit t an Stelle von x , dann gibt es offenbar ein x für das ϕ gilt, nämlich eben t .

Will man den Existenzquantor entfernen, muss man ein beliebiges Objekt x_0 nehmen, das ϕ an Stelle von x erfüllt und hat nun die Beweisverpflichtung zu zeigen, dass daraus χ herleitbar ist. In dieser Herleitung darf man keine spezielle Aussage über x_0 verwenden, außer $\phi[x_0/x]$.

1.7 Gleichheit

	<i>Einführung</i>	<i>Elimination</i>
$=$	$\frac{}{t = t} \quad = \text{ i, ID}$	$\frac{t_1 = t_2 \quad \phi[t_1/x]}{\phi[t_2/x]} \quad = \text{ e, SUB}$

Die Regel ID besagt, dass ein Symbol, das für ein Objekt steht dieses eindeutig bestimmt. Dies ist gewissermaßen die Charakteristik der Gleichheit.

Die Entfernung der Gleichheit besteht darin, dass wenn t_1 und t_2 gleich sind, man in einer Formel ϕ t_1 durch t_2 ersetzen kann. Dies klingt wie selbstverständlich, muss aber mit Vorsicht gehandhabt werden. Es sind nur gültige Substitutionen erlaubt: In allen Substitutionen $\phi[t/x]$ muss t frei für x in der Formel ϕ sein, d.h. keine freie Variable y in t gelangt durch das Einsetzen von x in ϕ in den Bereich eines Quantors $\forall y$ oder $\exists y$.

1.8 Beispiele

Gentzen zeigt in [Gen35, S. 183] an drei Beispielen, wie das natürliche Schließen geht. Für diese Beispiele folgen nun die Herleitungen. Dabei wird die Notation für die Beweise verwendet, wie sie in [HR04] definiert wurde.

Beispiel 1

Bewiesen werden soll die Formel

$$(X \wedge (Y \vee Z)) \rightarrow ((X \vee Y) \wedge (X \vee Z))$$

Im folgenden Beweis geben die Angaben rechts die jeweils verwendete Regel an und die Zeile (oder Zeilen), auf die sie angewandt wurden.

1.	$(X \vee (Y \wedge Z))$	assumption
2.	X	assumption
3.	$(X \vee Y)$	$\vee i_1$ [2]
4.	$(X \vee Z)$	$\vee i_1$ [2]
5.	$((X \vee Y) \wedge (X \vee Z))$	$\wedge i$ [3 4]
6.	$(Y \wedge Z)$	assumption
7.	Y	$\wedge e_1$ [6]
8.	$(X \vee Y)$	$\vee i_2$ [7]
9.	Z	$\wedge e_2$ [6]
10.	$(X \vee Z)$	$\vee i_2$ [9]
11.	$((X \vee Y) \wedge (X \vee Z))$	$\wedge i$ [8 10]
12.	$((X \vee Y) \wedge (X \vee Z))$	$\vee e$ [1 [2 5] [6 11]]
13.	$(X \wedge (Y \vee Z)) \rightarrow ((X \vee Y) \wedge (X \vee Z))$	$\rightarrow i$ [[1 12]]

Beispiel 2

Herleitung für

$$\exists x \forall y F(xy) \rightarrow \forall y \exists x F(xy)$$

1.	$\exists x \forall y F(xy)$	assumption
2.	a	assumption
3.	$\forall y F(ay)$	assumption
4.	b	assumption
5.	$F(ab)$	$\forall e$ [3 4]
6.	$\exists x F(xb)$	$\exists i$ [2 5]
7.	$\forall y \exists x F(xy)$	$\forall i$ [[4 6]]
8.	$\forall y \exists x F(xy)$	$\forall i$ [[2 7]]
9.	$\exists x \forall y F(xy) \rightarrow \forall y \exists x F(xy)$	$\rightarrow i$ [[1 8]]

Beispiel 3

Als drittes Beispiel folgt ein Beweis für

$$\neg \exists x G(x) \rightarrow \forall y \neg G(y)$$

Der Beweis folgt wieder der Argumentation von Gentzen in [Gen35, S. 183]:

1.	$\neg \exists x G(x)$	assumption
2.	a	assumption
3.	$G(a)$	assumption
4.	$\exists x G(x)$	$\exists i [2\ 3]$
5.	\perp	$\neg e [1\ 4]$
6.	$\neg G(a)$	$\neg i [[3\ 5]]$
7.	$\forall y \neg G(y)$	$\forall i [[2\ 6]]$
8.	$\neg \exists x G(x) \rightarrow \forall y \neg G(y)$	$\rightarrow i [[1\ 7]]$

1.9 Weitere Regeln für die klassische Logik

Die bisher diskutierten Regeln gelten für die intuitionistische Logik. Aus ihnen kann man das Gesetz des ausgeschlossenen Dritten (*Tertium non datur*) nicht herleiten.

Fügt man dieses als Regel hinzu

$$\frac{}{\phi \vee \neg \phi} \text{ TND}$$

erhält man die Regeln für das natürliche Schließen in der klassischen Logik.

Das Beweiswerkzeug Coq arbeitet üblicherweise in der intuitionistischen Logik. Wir wollen mit Coq *konstruktive* Beweise führen. Man kann allerdings in Coq auch die Äquivalenz verschiedener Charakterisierungen der klassischen Logik zeigen, was wir später tun werden.

2 Natürliches Schließen in Coq

Im Beweisassistenten Coq kann man Beweise im natürlichen Schließen machen. In Coq formuliert man eine zu beweisende Aussage und verwendet dann die in Coq verfügbaren *Taktiken*, um das Beweisziel herzuleiten.

In diesem Abschnitt werden die Taktiken vorgestellt, die den Regeln des natürlichen Schließens entsprechen – und en passant ein paar mehr, die die Herleitungen erleichtern. Die hier dargestellten Taktiken werden erläutert in [BC04, Kap. 5 „Everyday Logic“].

Folgende Tabelle (siehe [BC04, S. 130]) gibt einen Überblick³:

³Eine Tabelle, die statt `elim` die Taktik `destruct` verwendet, findet man in [PM11, S. 7].

	<i>Einführung</i>	<i>Elimination</i>
\rightarrow	<code>intro</code>	<code>apply</code>
\wedge	<code>split</code>	<code>elim</code>
\vee	<code>left, right</code>	<code>elim</code>
\neg	<code>intro</code>	<code>elim</code>
\forall	<code>intro</code>	<code>apply</code>
\exists	<code>exists v</code>	<code>elim</code>
$=$	<code>reflexivity</code>	<code>rewrite</code>

In Coq werden Beweise im Allgemeinen durch Anwenden der Regeln des natürlichen Schließens *rückwärts* geführt: man verändert das Ziel des Beweises durch die Taktiken.

2.1 Implikation

Um eine Implikation — bestehend aus einer Hypothese und einer Schlussfolgerung — einzuführen, verwendet man die Taktik `intro`, die die Hypothese annimmt und als Ziel die Schlussfolgerung setzt. Man hat also die Aufgabe aus der angenommenen Hypothese die Schlussfolgerung zu beweisen. Ist dies gelungen, dann hat man die Implikation selbst gezeigt.

Als Beispiel nehmen wir die Implikation $P \rightarrow Q$.

Betrachten wir was im Einzelnen passiert, indem wir protokollieren, was der Coq-Prozessor ausgibt. Die Eingaben sind dabei in nichtproportionaler Schriftart und die Ausgaben von Coq sind schräg gestellt.

Wenn man die Aussage, die man beweisen möchte, formuliert, gibt Coq an, dass ein Ziel erreicht werden muss, nämlich $P \rightarrow Q$, und dass der Beweis in der Umgebung erfolgt, die oberhalb der doppelt gestrichelten Linie angegeben wird.

```
Coq
Section Natural_Deduction.
Section impl_i.
Variables P Q: Prop.
P is declared
Q is declared
Example impl_i: P -> Q.
1 subgoal

  P, Q : Prop
  =====
  P -> Q
```

Ein Beweis beginnt mit dem Kommando **Proof**. Die Taktik **intro** führt die mit H bezeichnete Hypothese in die Umgebung ein und das Ziel ist nun Q . Das bedeutet, dass wir unter der Annahme, dass es einen Beweis für P gibt das Ziel Q zeigen müssen.

```
Coq _____
Proof.
  intro H.
  1 subgoal

    P, Q : Prop
    H : P
    =====
    Q
Coq _____
```

Wir sind jetzt also in einer Situation, in der wir Q aus P herleiten müssen. Das geht natürlich nicht, deshalb brechen wir den Beweis ab.

```
Coq _____
Abort.
Coq _____
```

Wenn wir als Ziel $P \rightarrow P$ beweisen möchten, dann geht das leicht:

```
Coq _____
Example impl_i': P -> P.
  1 subgoal

    P, Q : Prop
    =====
    P -> P
Proof.
  intro H.
  1 subgoal

    P, Q : Prop
    H : P
    =====
    P
Coq _____
```

Das Ziel P ist nun denkbar einfach zu erreichen, denn P ist ja gegeben. Die Taktik **assumption** überprüft die lokale Umgebung danach, ob es dort eine

Hypothese gibt, die das Ziel ergibt. In unserem Beispiel ist die offensichtlich der Fall.

```
Coq
  assumption.
No more subgoals.
Qed.
impl_i' is defined
End impl_i.
```

Voilà.

Um die Implikation zu entfernen, verwendet man in Coq die Taktik **apply**.

Die Regel hat zwei Voraussetzungen, die man in Coq vorgeben kann:

```
Coq
Section impl_e.
Variables P Q: Prop.
Hypothesis (H1: P) (H2: P -> Q).
Example impl_e: Q.
Proof.
  apply H2.
  exact H1.
Qed.
```

Betrachten wir wieder im Einzelnen, wie der Beweis geht:

```
Coq
Reset impl_e.
Example impl_e: Q.
1 subgoal

  P, Q : Prop
  H1 : P
  H2 : P -> Q
  =====
  Q
Proof.
```

Unser Ziel ist es Q zu zeigen. Da die Hypothese $H2$ aussagt, dass $P \rightarrow Q$ impliziert, genügt es offenbar P zu zeigen, dann gilt auch Q . Diesen Schritt im Beweis erreicht die Taktik **apply**:

```

Coq
  apply H2.
1 subgoal

  P, Q : Prop
  H1 : P
  H2 : P -> Q
  =====
  P

```

Man sieht an diesem Beispiel, wie die Taktik **apply** hier die Voraussetzung $P \rightarrow Q$ rückwärts anwendet.

Jetzt ist das Ziel P . Und diese Aussage ist gerade $H1$ in der lokalen Umgebung. Man könnte nun wie oben **assumption** verwenden. Wir lernen eine neue Taktik kennen: **exact** mit expliziter Angabe der zu verwendenden Aussage beendet den Beweis.

```

Coq
  exact H1.
No more subgoals.
Qed.
impl_e is defined
End impl_e.

```

Als interessanteres Beispiel für die Implikation zeigen wir das Lemma über das schwache Gesetz von Peirce.

Das Gesetz von Peirce (nach Charles Sanders Peirce⁴) lautet

$$((P \rightarrow Q) \rightarrow P) \rightarrow P$$

Diese Aussage impliziert das Gesetz vom ausgeschlossenen Dritten und kann in der intuitionistischen Logik nicht bewiesen werden. Wir werden in Abschnitt 2.10 sehen, dass das Gesetz von Peirce eine der Charakterisierungen der klassischen Logik ist.

Ein schwächere Version des Gesetzes kann man aber in der intuitionistischen Logik zeigen:

$$(((P \rightarrow Q) \rightarrow P) \rightarrow P) \rightarrow Q$$

⁴ Charles Sanders Peirce (1839–1914), US-amerikanischer Philosoph, Logiker und Mathematiker, siehe [Wikipedia über Charles Sanders Peirce](#).

```

Coq
Section weak_peirce.
Variables P Q: Prop.
Theorem weak_peirce: (((P -> Q) -> P) -> P) -> Q -> Q.
Proof.
  intro H0. apply H0.
  intro H1. apply H1.
  intro H2. apply H0.
  intro H3.
  exact H2.
Qed.
End weak_peirce.

```

Wir haben den Satz in der [Section weak_peirce](#) bewiesen, die wir dann wieder geschlossen haben. Die Variablen P und Q leben lokal in dieser [Section](#). Aber in unserem Beweis haben wir keinerlei speziellen Eigenschaften von P oder Q verwendet, sie stehen für *beliebige* Aussagen. Wenn die [Section](#) geschlossen wird, verallgemeinert Coq den Beweis: Wenn wir mit dem Kommando [Check](#) den Typ von `weak_peirce` überprüfen, sehen wir, dass wir in der Tat die Aussage für *alle* P und Q bewiesen haben.

```

Coq
Check weak_peirce.
weak_peirce
  : forall P Q : Prop, (((P -> Q) -> P) -> P) -> Q -> Q

```

2.2 Konjunktion

Für die Einführung der Konjunktion hat Coq die Taktik [split](#), die uns auferlegt, die linke und die rechte Seite der Konjunktion herzuleiten.

```

Coq
Section and_i.
Variables P Q: Prop.
Hypothesis (H1: P) (H2: Q).
Example and_i: P /\ Q.
Proof.
  split.
  - exact H1.
  - exact H2.
Qed.

```

In Schritten: Die Taktik **split** teilt das Ziel der Konjunktion in zwei Teile auf.

```

Coq
Reset and_i.
Example and_i: P /\ Q.
1 subgoal

  P, Q : Prop
  H1 : P
  H2 : Q
  =====
  P /\ Q
Proof.
  split.
2 subgoals

  P, Q : Prop
  H1 : P
  H2 : Q
  =====
  P
subgoal 2 is:
  Q
Coq

```

Nun müssen wir zwei Ziele zeigen, nämlich Ziel 1 P und Ziel 2 Q . Die Beweise der beiden Unterziele können wir durch den Bindestrich – gliedern.

Die beiden Aussagen sind wieder sehr einfach zu beweisen, denn wir haben sie ja vorausgesetzt. Sie sind deshalb Aussagen in unserer Umgebung. Wir verwenden erst die erste.

```

Coq
- exact H1.
1 subgoal

  P, Q : Prop
  H1 : P
  H2 : Q
  =====
  P
This subproof is complete, but there are some unfocused goals.
Focus next goal with bullet -.
1 subgoal

```

subgoal 1 is:

Q

Coq

Nun bleibt noch ein Ziel: Q . Der Bindestrich zeigt uns die Umgebung für dieses Teilziel an.

Coq

- exact H2.

1 subgoal

P, Q : Prop

H1 : P

H2 : Q

=====

Q

No more subgoals.

Qed.

and_i is defined

End and_i.

Coq

Für die Elimination der Konjunktion haben wir in Coq die Taktik **elim**. Im ersten Beispiel folgern wir die linke Seite der Konjunktion.

Coq

Section and_e.

Variables P Q: Prop.

Hypothesis (H: P /\ Q).

Example and_e1: P.

elim H.

intros.

assumption.

Qed.

Coq

Das geht mit der Taktik **destruct** einfacher:

Coq

Example and_e2: Q.

Proof.

destruct H as [H1 H2].

exact H2.

Qed.

Coq

Wir sehen uns **destruct** genauer an:

```

Coq
Reset and_e2.
Example and_e2: Q.
1 subgoal

  P, Q : Prop
  H : P /\ Q
  =====
  Q
Proof.
  destruct H as [H1 H2].
1 subgoal

  P, Q : Prop
  H : P /\ Q
  H1 : P
  H2 : Q
  =====
  Q
  exact H2.
No more subgoals.
Qed.
and_e2 is defined
End and_e.

```

Als etwas interessanteres Beispiel zeigen wir, dass die Konjunktion kommutativ ist.

Mit dem Semikolon ; kann man in Coq Taktiken kombinieren und als einen Schritt an den Coq-Prozessor übergeben.

```

Coq
Section and_comm.
Variables P Q: Prop.
Theorem and_comm: P /\ Q -> Q /\ P.
Proof.
  intro H.
  destruct H as [H1 H2].
  split; assumption.
Qed.
End and_comm.

```

2.3 Disjunktion

Die Disjunktion wird in Coq mit der Taktik `left` bzw, `right` eingeführt:

```
Coq
Section or_i.
Variables P Q: Prop.
Hypothesis (H1: P).
Example or_i1: P \/ Q.
Proof.
  left.
  exact H1.
Qed.
```

```
Coq
Hypothesis (H2: Q).
Example or_i2: P \/ Q.
Proof.
  right.
  exact H2.
Qed.
End or_i.
```

Die Elimination der Disjunktion kann man mit der Taktik `elim` machen.

```
Coq
Section or_e.
Variables P Q R: Prop.
Hypothesis (H: P \/ Q) (H1: P -> R) (H2: Q -> R).
Example or_e: R.
Proof.
  elim H.
  - intro Hp; apply H1; assumption.
  - intro Hq; apply H2; assumption.
Qed.
Reset or_e.
```

Im Einzelnen:

```
Coq
Example or_e: R.
```

1 subgoal

$P, Q, R : Prop$

$H : P \vee Q$

$H1 : P \rightarrow R$

$H2 : Q \rightarrow R$

=====

R

Proof.

elim H.

2 subgoals

$P, Q, R : Prop$

$H : P \vee Q$

$H1 : P \rightarrow R$

$H2 : Q \rightarrow R$

=====

$P \rightarrow R$

subgoal 2 is:

$Q \rightarrow R$

- intro HP; apply H1; assumption.

1 subgoal

$P, Q, R : Prop$

$H : P \vee Q$

$H1 : P \rightarrow R$

$H2 : Q \rightarrow R$

=====

$P \rightarrow R$

This subproof is complete, but there are some unfocused goals.

Focus next goal with bullet -.

1 subgoal

subgoal 1 is:

$Q \rightarrow R$

- intro HQ; apply H2; assumption.

1 subgoal

$P, Q, R : Prop$

$H : P \vee Q$

$H1 : P \rightarrow R$

$H2 : Q \rightarrow R$

=====

$Q \rightarrow R$

No more subgoals.

Qed.

or_e is defined

Reset or_e.

Coq

Eine Alternative ist die Verwendung der Taktik **destruct**:

Coq

Example or_e: R.

1 subgoal

P, Q, R : Prop

H : P \vee Q

H1 : P \rightarrow R

H2 : Q \rightarrow R

=====

R

Proof.

destruct H as [HP|HQ].

2 subgoals

P, Q, R : Prop

H : P \vee Q

H1 : P \rightarrow R

H2 : Q \rightarrow R

HP : P

=====

R

subgoal 2 is:

R

apply H1; assumption.

1 subgoal

P, Q, R : Prop

H : P \vee Q

H1 : P \rightarrow R

H2 : Q \rightarrow R

HQ : Q

=====

R

apply H2; assumption.

No more subgoals.

Qed.

or_e is defined

End or_e.

Coq

Zwei weitere, etwas interessantere Beispiele:

```
Coq
Section or_comm.
Variables P Q: Prop.
Theorem or_comm: P ∨ Q -> Q ∨ P.
Proof.
  intro H.
  destruct H as [HP|HQ].
  right; assumption.
  left; assumption.
Qed.
End or_comm.
```

```
Coq
Section or_assoc.
Variables P Q R: Prop.
Theorem or_assoc: (P ∨ Q) ∨ R -> P ∨ (Q ∨ R).
Proof.
  intro H.
  elim H.
  - intro H1; elim H1.
    - intro H11; left; assumption.
    - intro H12; right; left; assumption.
  - intro H2; right; right; assumption.
Qed.
```

Im Beispiel der Assoziativität brauchen wir mehrere **intros**. Dies kann man vereinfachen durch den Mechanismus der zerlegenden Bindung (*Destructuring*) in Coq. Am Beispiel der Assoziativität der Disjunktion sei dies nochmals demonstriert:

```
Coq
Reset or_assoc.
Theorem or_assoc: (P ∨ Q) ∨ R -> P ∨ (Q ∨ R).
Proof.
  intro H.
  destruct H as [[HP | HQ] | HR].
  - left; assumption.
  - right; left; assumption.
  - right; right; assumption.
Qed.
```

Wir können im Detail verfolgen, wie die Hypothesen abzuarbeiten sind:

```

Coq
Reset or_assoc.
Theorem or_assoc: (P ∨ Q) ∨ R -> P ∨ (Q ∨ R).
1 subgoal

  P, Q, R : Prop
  =====
  (P ∨ Q) ∨ R -> P ∨ Q ∨ R
Proof.
  intro H.
1 subgoal

  P, Q, R : Prop
  H : (P ∨ Q) ∨ R
  =====
  P ∨ Q ∨ R
  destruct H as [[HP | HQ] | HR].
3 subgoals

  P, Q, R : Prop
  HP : P
  =====
  P ∨ Q ∨ R
subgoal 2 is:
  P ∨ Q ∨ R
subgoal 3 is:
  P ∨ Q ∨ R

```

Nach der Zerlegung von H haben wir drei Unterziele und für die Herleitung des ersten ist die Voraussetzung HP gegeben.

```

Coq
- left; assumption.
1 subgoal

  P, Q, R : Prop
  HP : P
  =====
  P ∨ Q ∨ R
This subproof is complete, but there are some unfocused goals.
Focus next goal with bullet -.

```

```

2 subgoals
subgoal 1 is:
  P \ / Q \ / R
subgoal 2 is:
  P \ / Q \ / R

```

Coq

Nach der Einführung der Disjunktion ergibt sich das Teilziel aus den gegebenen Voraussetzungen. Bleiben zwei weitere Teilziele, die ganz analog zu beweisen sind:

Coq

```

- right; left; assumption.
1 subgoal

```

```

P, Q, R : Prop
HQ : Q

```

```

=====

```

```

P \ / Q \ / R

```

*This subproof is complete, but there are some unfocused goals.
Focus next goal with bullet -.*

```

1 subgoal
subgoal 1 is:
  P \ / Q \ / R

```

```

- right; right; assumption.

```

```

1 subgoal

```

```

P, Q, R : Prop
HR : R

```

```

=====

```

```

P \ / Q \ / R

```

No more subgoals.

Qed.

or_assoc is defined

End or_assoc.

Coq

2.4 Negation

Die Negation führt man mit der Taktik `intro` ein.

Beispiel:

Coq

```

Section not_i.

```

```

Variable P: Prop.
Hypothesis (HP: P -> False).
Example not_i: ~P.
Proof.

```

Coq

Die Taktik **intro** ersetzt das Ziel $\neg P$ durch die Beweisverpflichtung $P \rightarrow \bot$:

Coq

```

  intro H.
1 subgoal

  P : Prop
  HP : P -> False
  H : P
  =====
  False

```

Coq

Dies ist in unserem Beispiel sehr einfach, weil wir es ja vorausgesetzt haben.

Coq

```

  apply HP; assumption.
Qed.
End not_i.

```

Coq

Für die Elimination von \neg nimmt man die Taktik **elim**:

Coq

```

Section not_e.
Variables P Q: Prop.
Hypothesis (H: P /\ ~P).
Example not_e: Q.
Proof.
  elim H.
  intros H1 H2.
  elim H2.
  assumption.
Qed.

```

Coq

Wir haben oben bereits gesehen, dass dieses Beispiel auch einfacher geht:

```

Coq
Reset not_e.
Example not_e: Q.
Proof.
  destruct H as [H1 H2].
  elim H2; assumption.
Qed.
Coq

```

Oder mit der Taktik **absurd**:

```

Coq
Reset not_e.
Example not_e: Q.
Proof.
  absurd P; destruct H; assumption.
Qed.
End not_e.
Coq

```

Nun zum Abschluss des Abschnitts über die Negation noch zwei interessantere Beispiele:

```

Coq
Section double_neg.
Variable P: Prop.
Theorem double_neg: P -> ~~P.
Proof.
  intros H H1.
  elim H1.
  assumption.
Qed.
End double_neg.
Coq

```

Die umgekehrte Richtung $\neg\neg P \rightarrow P$ gilt in der intuitionistischen Logik nicht. Sie ist vielmehr eine Charakterisierung der klassischen Logik und äquivalent zu $P \vee \neg P$. Dies werden wir mit Coq beweisen im Abschnitt [2.10](#).

```

Coq
Section contraposition.
Variables P Q: Prop.
Theorem contraposition: (P -> Q) -> ~Q -> ~P.
Proof.

```

```

    intros H H1 H2.
    elim H1.
    apply H; assumption.
Qed.
End contraposition.

```

2.5 Allquantor

Die drastische Erweiterung der Ausdrucksmöglichkeiten in der Prädikatenlogik gegenüber der Aussagenlogik kommt dadurch zustande, dass wir neben Aussagen Objekte eines Universums haben, *über* die wir etwas aussagen können, was für diese Objekte wahr oder falsch sein kann — eben *Prädikate*.

In Coq können wir das Universum als Menge U vom Typ `Set` deklarieren und für unsere Beispiele ein Element a des Universums sowie ein einstelliges Prädikat S vorgeben.

```


```

```

Section quantors.
Variable U: Set.      (* Das Universum *)
Variable S: U -> Prop.

```

Für die Einführung des Allquantors hat Coq die Taktik `intro`, die uns ein beliebiges Objekt des Universums gibt, mit dem wir die Aussage im Allquantor herleiten können.

```


```

```

Section forall_i.
Example forall_i: forall x: U, S x -> S x.
Proof.
  intros x H.
  assumption.
Qed.

```

Im Detail:

```


```

```

Reset forall_i.
Example forall_i: forall x: U, S x -> S x.
1 subgoal

  U : Set

```

```

S : U -> Prop
=====
forall x : U, S x -> S x
Proof.
  intros x H.
1 subgoal

U : Set
S : U -> Prop
x : U
H : S x
=====
S x
assumption.
No more subgoals.
Qed.
forall_i is defined
End forall_i.
Coq

```

Das Element des Universums, das mit **intro** eingeführt wird, mit ein *frisches* Element sein. Verlangen wir an Stelle von **intro x intro a** für ein *a*, das wir bereits im Kontext definiert haben, gibt Coq die Fehlermeldung **a is already used** aus!

Die Elimination des Allquantors macht man mit der Taktik **apply**, wie folgendes Beispiel zeigt:

```

Coq
Section forall_e.
Variable a: U.
Example forall_e: (forall x: U, S x) -> S a.
Proof.
  intro H.
  apply H.
Qed.
End forall_e.
Coq

```

Im folgenden Beispiel verwenden wir ein zweistelliges Prädikat *T* und eine einstellige Funktion *f* auf dem Universum *U*.

Wir geben mit dem Kommando **Hypothesis** Eigenschaften von *T* vor: *T* ist reflexiv und überdies ist *T* abgeschlossen bei der Anwendung der Funktion *f* auf die zweite Stelle von *T*. Diese Eigenschaften können wir dann im Beweis des Beispiels verwenden.

```

Coq
Section Ex_forall.
Variable T: U -> U -> Prop.
Variable f: U -> U.
Hypothesis T_reflexiv: forall x: U, T x x.
Hypothesis T_f: forall x y: U, T x y -> T x (f y).

```

Außerdem lernen wir die Taktik **repeat** kennen. Um das Ziel zu zeigen, muss man die gegebene Voraussetzung T_f dreimal anwenden, um den dreimaligen Funktionsaufruf von f herzuleiten. Dies kann man mit **repeat** erreichen, die `apply T_f` solange auf das Ziel anwendet, bis kein Fortschritt mehr erzielt wird.

```

Coq
Example Ex: forall x: U, T x (f (f (f x))).
Proof.
  intro x.
  repeat apply T_f.
  apply T_reflexiv.
Qed.
End Ex_forall.

```

2.6 Existenzquantor

Die Taktik **exists** v dient in Coq der Einführung des Existenzquantors. Dazu müssen wir ein in der Umgebung tatsächlich vorhandenes Objekt des Universum für v verwenden, etwa unsere Variable a .

```

Coq
Section exists_i.
Variable a: U.
Example exists_i: (forall x: U, S x) -> (exists x: U, S x).
Proof.
  intro H.
  exists a.
  apply H.
Qed.

```

Im Detail:

```

Coq

```

```

Reset exists_i.
Example exists_i: (forall x: U, S x) -> (exists x: U, S x).
1 subgoal

  U : Set
  S : U -> Prop
  a : U
  =====
  (forall x : U, S x) -> exists x : U, S x
Proof.
  intro H.
1 subgoal

  U : Set
  S : U -> Prop
  a : U
  H : forall x : U, S x
  =====
  exists x : U, S x
exists a.
1 subgoal

  U : Set
  S : U -> Prop
  a : U
  H : forall x : U, S x
  =====
  S a
  apply H.
No more subgoals.
Qed.
exists_i is defined
End exists_i.

```

Coq

Wir zeigen die Elimination des Existenzquantors mit der Taktik **elim** an einem etwas interessanteren Beispiel, nämlich

$$\exists x S(x) \rightarrow \neg(\forall x \neg S(x))$$

```

Coq
Section exists_e.
Example exists_e: (exists x: U, S x) -> ~(forall x: U, ~S x).
Proof.
  intro H; intro H1.

```

```

    elim H.
    assumption.
Qed.
Coq

```

Oder:

```

Coq
Reset exists_e.
Example exists_e: (exists x: U, S x) -> ~(forall x: U, ~S x).
Proof.
  intro H.
  destruct H as (x, HS).
  intro HN.
  elim HN with (x:=x).
  assumption.
Qed.
Coq

```

Und zum Abschluss der Beispiele zum Existenzquantor noch eine weitere interessante Aussage:

$$\forall x S(x) \rightarrow \neg(\exists y \neg S(y))$$

```

Coq
Theorem forall_not_exists_not: (forall x: U, S x) ->
    ~(exists y: U, ~S y).
Proof.
  intro H. intro H1.
  destruct H1 as (x, HS).
  elim HS.
  apply H.
Qed.
End exists_e.
Coq

```

2.7 Gleichheit

Um die Beispiele für die Gleichheit zeigen zu können, brauchen wir Objekte des Universums.

```

Coq
Section equal_i.
Variable t: U.
Coq

```

Mit der Taktik `reflexivity` führt man die Gleichheit ein:

```
Coq
Example equal_i: t = t.
Proof.
  reflexivity.
Qed.
End equal_i.
```

Die Elimination der Gleichheit geschieht mit der Taktik `rewrite`, genauer gesagt der Variante von `rewrite`, die die Gleichheit $t1 = t2$ in $H1$ von rechts nach links verwendet, also im Ziel $S\ t2$ das Objekt $t2$ durch $t1$ ersetzt.

```
Coq
Section equal_e.
Variables t1 t2: U.
Hypothesis (H1: t1 = t2) (H2: S t1).
Example equal_e:S t2.
Proof.
  rewrite <- H1.
  assumption.
Qed.
End equal_e.
```

Als abschließendes Beispiel zeigen wir, dass die Gleichheit symmetrisch ist.

```
Coq
Theorem equal_sym: forall x y: U, x = y -> y = x.
Proof.
  intros x y H.
  rewrite <- H.
  reflexivity.
Qed.
End quantors.
```

2.8 Verallgemeinerung der bewiesenen Aussagen

Wir beenden nun die `Section Natural_Deduction`. Die in der Section deklarierten lokalen Variablen sind dann nicht mehr verfügbar.

```
Coq
End Natural_Deduction.
```

Für die Aussagen, die wir bewiesen haben, geschieht eine sehr wesentliche Verallgemeinerung, wie wir bereits ob am Beispiel `weak_peirce` gesehen haben.

Wir haben zum Beispiel die Variablen P und Q vom Type `Prop` lokal deklariert und im Beweis der Kommutativität des Junktors \vee verwendet. Dabei haben wir aber keinerlei *spezifischen* Eigenschaften der beiden Aussagen verwendet. Eigentlich haben wir gezeigt, dass die Kommutativität für zwei beliebige Aussagen gilt, nicht nur für P und Q . Und dies ist in Coq auch tatsächlich so, wie der Kommando `Check` zeigt. `Check` zeigt den Typ eines Ausdrucks an:

```
Coq
Check or_comm.
or_comm
      : forall P Q : Prop, P \/ Q -> Q \/ P
```

Und wir können als `or_comm` für beliebige Aussagen anwenden:

```
Coq
Section Use_or_comm.
Variables X Y: Prop.
X is declared
Y is declared
Example or_c: X \/ Y -> Y \/ X.
1 subgoal

  X, Y : Prop
  =====
  X \/ Y -> Y \/ X
Proof.
  apply or_comm.
No more subgoals.
Qed.
or_c is defined
End Use_or_comm.
```

Was geschieht mit Formeln, die Quantoren beinhalten?

```
Coq
Check forall_not_exists_not.
forall_not_exists_not
      : forall (U : Set) (S : U -> Prop),
```

$$(forall\ x : U, S\ x) \rightarrow \sim (exists\ y : U, \sim S\ y)$$

Coq

Es wird verallgemeinert für beliebige Mengen und entsprechende Prädikate.

Im *Calculus of Inductive Constructions*, der Coq zugrundeliegt, gibt es ein viel allgemeineres Konzept als die Menge. Tatsächlich ist in Coq eine Menge vom Type `Type(0)` und darauf aufbauend gibt es eine Hierarchie von Typen, wobei der Typ jedes Term auf die Level i ein Term auf dem Level $i + 1$ ist. Dies wird genau erläutert in [BC04, Abschnitt 2.5.2].

In der Standard-Bibliothek von Coq wird deshalb die Formel für beliebige Typen, nicht für `Set` bewiesen:

Coq

```
Lemma all_not_ex_not:
  forall (U: Type) (P: U -> Prop),
    (forall x: U, P x) -> ~ (exists x:U, ~ P x).
```

Proof.

```
  intros.
  intro H1.
  elim H1.
  intros x H2.
  elim H2.
  apply H.
```

Qed.

Coq

Coq

```
Check all_not_ex_not.
```

```
all_not_ex_not
```

```
  : forall (U : Type) (P : U -> Prop),
    (forall x : U, P x) -> ~ (exists x : U, ~ P x)
```

Coq

2.9 Beispiele

Nachdem wir an Beispielen gesehen haben, wie die Regeln des natürlichen Schließens in Coq durch die jeweiligen Taktiken eingesetzt werden, um das Beweisziel zu erreichen, wollen wir es auf die drei Beispiele aus der Arbeit von Gerhard Gentzen anwenden, analog zum Abschnitt 1.8.

Coq

```
Section Gentzen.
```

Coq

Beispiel 1

$$(X \wedge (Y \vee Z)) \rightarrow ((X \vee Y) \wedge (X \vee Z))$$

Coq

```

Variables X Y Z: Prop.
Example Beispiel1: X /\ (Y \/ Z) -> (X \/ Y) /\ (X \/ Z).
Proof.
  intro H.
  destruct H as [HX [HY | HZ]].
  - split; repeat left; assumption.
  - split; repeat left; assumption.
Qed.

```

Coq

Tatsächlich gehen so einfache Formeln in Coq auch automatisch:

Coq

```

Reset Beispiel1.
Example Beispiel1: X /\ (Y \/ Z) -> (X \/ Y) /\ (X \/ Z).
Proof.
  tauto.
Qed.

```

Coq

Die Taktik **tauto** verwendet einen Entscheidungsalgorithmus, der für *alle* Tautologien der intuitionistischen Aussagenlogik zum Ziel führt.

Beispiel 2

$$\exists x \forall y F(xy) \rightarrow \forall y \exists x F(xy)$$

Coq

```

Variables (U: Set) (F: U -> U -> Prop) (G: U -> Prop).
Example Beispiel2: (exists x: U, forall y: U, F x y) ->
  (forall y: U, exists x: U, F x y).
Proof.
  intros H1 a.
  elim H1.
  intros b H2.
  exists b.
  apply H2.
Qed.

```

Coq

Auch für dieses Beispiel ist Coq viel mächtiger:

```
Coq
Reset Beispiel2.
Example Beispiel2: (exists x: U, forall y: U, F x y) ->
                  (forall y: U, exists x: U, F x y).

Proof.
  firstorder.
Qed.
```

Die Taktik **firstorder** ist eine (experimentelle – wie die Referenz sagt) Erweiterung von **tauto** für die intuitionistische Prädikatenlogik.

Beispiel 3

Schließlich zeigen wir noch

$$\neg \exists x G(x) \rightarrow \forall y \neg G(y)$$

```
Coq
Example Beispiel3: (~ exists x: U, G x) -> (forall y: U, ~ G y).
Proof.
  intros H a ga.
  apply H.
  exists a.
  assumption.
Qed.
```

```
Coq
End Gentzen.
```

2.10 Charakterisierungen der klassischen Logik

In diesem Abschnitt wollen wir die Äquivalenz von fünf Charakterisierungen der klassischen Logik mit Coq zeigen. (Dies ist Aufgabe 5.7 in [BC04, S. 123]).

```
Coq
Section Classical.
Definition peirce
  := forall P Q : Prop, ((P -> Q) -> P) -> P.
Definition notnot_e
```

```

:= forall P : Prop, ~~P -> P.
Definition tnd
:= forall P : Prop, P \/ ~P.
Definition dm_not_and_not
:= forall P Q : Prop, ~(~P /\ ~Q) -> P \/ Q.
Definition implies_to_or
:= forall P Q : Prop, (P -> Q) -> (~P \/ Q).

```

In den folgenden Beispielen kommt die Taktik **unfold** zum Einsatz, die δ -Reduktionen anwendet. Eine δ -Reduktion besteht darin, Bezeichner durch ihre Definitionen zu ersetzen.

```

Coq
Lemma peirce_notnot_e: peirce -> notnot_e.
Proof.
  unfold peirce.
  intros Hpeirce P H.
  apply (Hpeirce P False).
  intro H1.
  elim H.
  assumption.
Qed.

```

Im folgenden Beispiel verwenden wir die Taktik **absurd**. Diese Taktik wendet die Elimination des Widerspruchs an, d.h. sie leitet das Ziel vom Widerspruch her und erzeugt als neue Ziele $\neg P$ und P .

```

Coq
Lemma notnot_e_tnd: notnot_e -> tnd.
Proof.
  unfold tnd.
  intros Hnotnot_e P.
  apply Hnotnot_e.
  intro H.
  absurd P. (* Um False zu zeigen, zeigt man ~ P und P *)
  - intro H1.
    apply H; left; assumption.
  - apply Hnotnot_e.
    intro H2.
    apply H; right; assumption.
Qed.

```

Coq

Lemma tnd_dm_not_and_not : tnd -> dm_not_and_not.

Proof.

```

  intro Htnd.
  unfold dm_not_and_not.
  intros P Q H.
  assert (P \/ ~P).
  apply Htnd.
  assert (Q \/ ~Q).
  apply Htnd.
  elim H0.
  - intro HP; left; exact HP.
  - elim H1.
    - intro HQ; right; exact HQ.
    - intros HnQ HnP.
      elim H.
      split; repeat assumption.

```

Qed.

Coq

In diesem Beispiel wurde die Taktik `assert` verwendet. Diese Taktik erzeugt eine neue Hypothese und zugleich eine Beweisverpflichtung für sie. Im folgenden Beispiel kommt ferner die Taktik `trivial` vor: Diese Taktik wird typischerweise eingesetzt, um offensichtliche Beweisschritte zu machen.

Coq

Lemma dm_implies_to_or : dm_not_and_not -> implies_to_or.

Proof.

```

  intro Hdm.
  unfold implies_to_or.
  intros P Q H.
  apply Hdm.
  intro H1.
  elim H1.
  intros H2 H3.
  assert P.
  assert (hc: P \/ ~P).
  - apply Hdm.
    intro Hx.
    elim Hx.
    intros Hy Hz.
    apply Hz.
    exact Hy.
  - elim hc.
    - trivial.

```

```

      - intro H4.
        elim H2.
        exact H4.
    - apply H3.
      apply H.
      exact H0.

```

Qed.

Coq

Coq

Lemma implies_to_or_peirce : implies_to_or -> peirce.

Proof.

```

  intro Himp.
  unfold peirce.
  intros P Q H.
  assert (H1: ~P \ / P).
  - apply Himp.
    trivial.
  - elim H1.
    - intro H2; apply H.
      intro HP; elim H2.
      exact HP.
    - trivial.

```

Qed.

Coq

Coq

End Classical.

Coq

Literaturverzeichnis

- [BC04] BERTOT, Yves ; CASTÉLAN, Pierre: *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*. Berlin : Springer, 2004
- [Bor05] BORNAT, Richard: *Proof and Disproof in Formal Logic: An introduction for programmers*. Oxford : Oxford University Press, 2005
- [Gen35] GENTZEN, Gerhard: Untersuchungen über das logische Schließen. I. In: *Mathematische Zeitschrift* 39 (1935), S. 176–210
- [HR04] HUTH, Michael ; RYAN, Mark: *Logic in Computer Science: Modelling and Reasoning about Systems*. 2. Auflage. Cambridge : Cambridge University Press, 2004
- [PM11] PAULIN-MOHRING, Christine: *Introduction to the Coq proof-assistant for practical software verification*. Course notes for the 8th LASER Summer School on Software Engineering (LASER 2011), siehe <https://www.lri.fr/~paulin>, 2011