Natürliches Schließen in Coq

Ein einführendes Tutorial

Burkhardt Renz

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

Rev 1.0 – 6. Januar 2019

Inhaltsverzeichnis

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 Kapital 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 natürliches 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 Schließen wurde 1934 von Gerhard Gentzen[1] und unabhängig von ihm von Stanisław Jaśkowski[2] 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:

	Einführung	*Elimination*
$\to$	$\dfrac{\boxed{\begin{array}{c}\phi\\ \vdots\\ \psi\end{array}}}{\phi \to \psi}\ \mathsf{\to i}$	$\dfrac{\phi \quad \phi \to \psi}{\psi}\ \mathsf{\to e, MP}$

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 $\phi \to \psi$ 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

	Einführung	*Elimination*	
$\wedge$	$\dfrac{\phi \quad \psi}{\phi \wedge \psi}\ \mathsf{\wedge i}$	$\dfrac{\phi \wedge \psi}{\phi}\ \mathsf{\wedge e_1}$	$\dfrac{\phi \wedge \psi}{\psi}\ \mathsf{\wedge e_2}$

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

	Einführung		*Elimination*
$\vee$	$\dfrac{\phi}{\phi \vee \psi}\ \mathsf{\vee i_1}$	$\dfrac{\psi}{\phi \vee \psi}\ \mathsf{\vee i_2}$	$\dfrac{\phi \vee \psi \quad \boxed{\begin{array}{c}\phi\\ \vdots\\ \chi\end{array}} \quad \boxed{\begin{array}{c}\psi\\ \vdots\\ \chi\end{array}}}{\chi}\ \mathsf{\vee e}$

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

	Einführung	*Elimination*
$\neg$	$\dfrac{\boxed{\begin{array}{c}\phi\\ \vdots\\ \bot\end{array}}}{\neg\phi}$ ¬i	$\dfrac{\phi \quad \neg\phi}{\psi}$ ¬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 $\bot$ 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:

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

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

1.5 Allquantor

	Einführung	*Elimination*
$\forall$	$\dfrac{\boxed{\begin{array}{l}x_0\\ \qquad\vdots\\ \qquad\phi[x_0/x]\end{array}}}{\forall x\phi}$ ∀x i	$\dfrac{\forall x\phi}{\phi[t/x]}$ ∀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 $\phi[x_o/x]$). 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

	Einführung	*Elimination*
$\exists$	$\dfrac{\phi[t/x]}{\exists x\phi}\ \exists\mathsf{x\ i}$	$\dfrac{\exists x\phi \quad \boxed{\begin{array}{ll} x_0 & \phi[x_0/x] \\ & \vdots \\ & \chi \end{array}}}{\chi}\ \exists\mathsf{x\ e}$

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 $\phi[x_o/x]$.

1.7 Gleichheit

	Einführung	*Elimination*
$=$	$\dfrac{}{t = t}\ =\mathsf{i},\mathsf{ID}$	$\dfrac{t_1 = t_2 \quad \phi[t_1/x]}{\phi[t_2/x]}\ =\mathsf{e},\mathsf{SUB}$

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

$$(X \land (Y \lor Z)) \to ((X \lor Y) \land (X \lor Z))$$

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.	$(X \wedge (Y \vee Z)) \to ((X \vee Y) \wedge (X \vee Z))$	$\to$ i [[1 12]]

Beispiel 2

Herleitung für

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

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.	$\exists x \forall y F(xy) \to \forall y \exists x F(xy)$	$\to$ i [[1 8]]

Beispiel 3

Als drittes Beispiel folgt ein Beweis für

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

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.	$\bot$	$\neg$e [1 4]
6.	$\neg G(a)$	$\neg$i [[3 5]]
7.	$\forall y \neg G(y)$	$\forall$i [[2 6]]
8.	$\neg\exists x G(x) \to \forall y \neg G(y)$	$\to$ 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 Überblick:

	Einführung	*Elimination*
$\rightarrow$	`intro`	`apply`
$\wedge$	`split`	`elim`
$\vee$	`left`, `right`	`elim`
$\neg$	`intro`	`elim`
$\forall$	`intro`	`apply`
$\exists$	`exists` v	`elim`
$=$	`reflexivity`	`rewrite`

Wir beginnen die Vorstellung der Taktiken für das natürliche Schließen, indem wir eine `Section` öffnen und drei Variablen vom Type `Prop` definieren.

In einer `Section` kann man Variablen lokal deklarieren. Wie wir später sehen werden, werden in der `Section` definierte globale Objekte beim Beenden der Section verallgemeinert: Sie gelten nicht nur für die lokal verwendeten Aussagen, sondern für alle Aussagen $P, Q, R, \ldots$.

In Coq hat jedes Symbol einen Typ. Da wir zunächst die Junktoren der Aussagenlogik diskutieren wollen, werden drei Aussagen vom Typ `Prop` definiert.

Coq
```
Section Natural_Deduction.
Variables P Q R: Prop.
```
Coq

Ab jetzt können wir diese Aussagen in unseren Beispielen verwenden.

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 eine triviale Implikation, nämlich $P \rightarrow P$. Der Beweis ist dann denkbar einfach:

Coq
```
Example impl_i: P -> P.
Proof.
 intro H.
 assumption.
Qed.
```
Coq

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.

Das Kommando `Reset` setzt den eben geführten Beweis zurück, damit man neu mit der Herleitung beginnen kann.

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

Coq

```
Reset impl_i.
Example impl_i: P -> P.
1 subgoal

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

Coq

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 P. Das bedeutet, dass wir unter der Annahme, dass es einen Beweis für P gibt das Ziel P zeigen müssen.

Coq

```
Proof.
 intro H.
1 subgoal

  P, Q, R : 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
```

Voilà.

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

Als Beispiel nehmen wir den *Modus ponens*:

```
Example impl_e: P -> (P -> Q) -> Q.
Proof.
 intros H1 H2.
 apply H2.
 exact H1.
Qed.
```

Betrachten wir wieder im Einzelnen, wie der Beweis geht:

```
Reset impl_e.
Example impl_e: P -> (P -> Q) -> Q.
1 subgoal

  P, Q, R : Prop
  ============================
  P -> (P -> Q) -> Q
```

Unser Ziel ist die Aussage $P \to (P \to Q) \to Q$.

Als ersten Schritt verwenden wir die Taktik `intros`, um die beiden Hypothesen P und $P \to Q$ in die lokale Umgebung zu bringen. Dies ist zweimal die Einführung der Implikation, die wir oben gesehen haben.

```
Proof.
 intros H1 H2.
1 subgoal

  P, Q, R : Prop
  H1 : P
```

```
  H2 : P -> Q
  ============================
  Q
```

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

```
 apply H2.
1 subgoal

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

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.

```
 exact H1.
No more subgoals.
Qed.
impl_e is defined
```

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

Das Gesetz von Peirce (nach Charles Sanders Peirce[3]) lautet

$$((P \to Q) \to P) \to P$$

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.

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

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

$$((((P \to Q) \to P) \to P) \to Q) \to Q$$

Coq

```
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.
```

Coq

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.

Wir verwenden als Beispiel die Aussage $P \to Q \to Q \wedge P$.

Coq

```
Example and_i: P -> Q -> P /\ Q.
Proof.
  intros H1 H2.
  split.
  - exact H1.
  - exact H2.
Qed.
```

Coq

In Schritten:

Zunächst führen wir die beiden Hypothesen ein und wenden dann die Taktik **`split`** an.

Coq

```
Reset and_i.
Example and_i: P -> Q -> P /\ Q.
1 subgoal

  P, Q, R : Prop
  ============================
```

```
  P -> Q -> P /\ Q
Proof.
  intros H1 H2.
1 subgoal

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

  P, Q, R : 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, R : 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, R : Prop
  H1 : P
  H2 : Q
  ============================
  Q
No more subgoals.
Qed.
and_i is defined
```

Coq

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

Coq

```
Example and_e1: P /\ Q -> P.
  intro H.
  elim H.
  intros.
  assumption.
Qed.
```

Coq

Das geht in diesem Beispiel auch einfacher:

Coq

```
Example and_e2: P /\ Q -> Q.
Proof.
  intro H.
  apply H.
Qed.
```

Coq

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
```
Theorem and_comm: P /\ Q -> Q /\ P.
Proof.
  intro H.
  elim H.
  intros H1 H2.
  split; assumption.
Qed.
```
Coq

2.3 Disjunktion

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

Coq
```
Example or_i1: P -> P \/ Q.
Proof.
  intro H.
  left.
  exact H.
Qed.
```
Coq

Coq
```
Example or_i2: Q -> P \/ Q.
Proof.
  intro H.
  right.
  exact H.
Qed.
```
Coq

Zwei weitere, etwas interessantere Beispiele:

Coq
```
Theorem or_comm: P \/ Q -> Q \/ P.
Proof.
  intro H.
  elim H.
  - intro H1; right; assumption.
  - intro H2; left; assumption.
Qed.
```
Coq

Coq
```
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.
```
Coq

Im Beispiel der Assoziativität brauchen wir mehrere `intro`s. Dies kann man vereinfachen durch den Mechanismus der zerlegenden Bindung (*Destructuring*) in Coq. Am Beispiel der Assoziativität der Disjunktion sei dies 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.
```
Coq

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
```
Coq

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
```
Coq

2.4 Negation

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

Beispiel:

Coq
```
Example not_i: ~ (P /\ ~P).
Proof.
  intro H.
  destruct H as [H1 H2].
  apply H2; assumption.
Qed.
```
Coq

Da die Negation $\neg\phi$ gerade $\phi \to \bot$ bedeutet, ergibt die Taktik **intro** als Ziel den Widerspruch, wie man an unserem Beispiel sehen kann:

Coq
```
Reset not_i.
Example not_i: ~ (P /\ ~P).
1 subgoal

  P, Q, R : Prop
```

```
  ============================
  ~ (P /\ ~ P)
Proof.
  intro H.
1 subgoal

  P, Q, R : Prop
  H : P /\ ~ P
  ============================
  False
```

Coq

Die Hypothese H ist nun $P \wedge \neg P$. Wir können sie zerlegen in die beiden Teile und dann den Widerspruch herbeiführen.

Coq

```
  destruct H as [H1 H2].
1 subgoal

  P, Q, R : Prop
  H1 : P
  H2 : ~ P
  ============================
  False
  apply H2; assumption.
No more subgoals.
Qed.
not_i is defined
```

Coq

Für die Elimination von ¬ nimmt man die Taktik **elim**:

Coq

```
Example not_e: P /\ ~P -> Q.
Proof.
  intro H.
  elim H.
  intro H1.
  intro H2.
  elim H2.
  assumption.
Qed.
```

Coq

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

Coq

```
Reset not_e.
Example not_e: P /\ ~P -> Q.
Proof.
  intro H.
  destruct H as [H1 H2].
  elim H2; assumption.
Qed.
```

Coq

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

Coq

```
Theorem double_neg: P -> ~~P.
Proof.
  intros H H1.
  elim H1.
  assumption.
Qed.
```

Coq

Die umgekehrte Richtung $\neg\neg P \to P$ 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

```
Theorem contraposition: (P -> Q) -> ~Q -> ~P.
Proof.
  intros H H1 H2.
  elim H1.
  apply H; assumption.
Qed.
```

Coq

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.

Coq

```
Variable U: Set.    (* Das Universum *)
Variable a: U.
Variable S: U -> Prop.
```

Coq

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.

Coq

```
Example forall_i: forall x: U, S x -> S x.
Proof.
  intro x.
  intro H.
  assumption.
Qed.
```

Coq

Im Detail:

Coq

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

  P, Q, R : Prop
  U : Set
  a : U
  S : U -> Prop
  ============================
  forall x : U, S x -> S x
Proof.
  intro x.
1 subgoal

  P, Q, R : Prop
  U : Set
  a : U
  S : U -> Prop
  x : U
  ============================
  S x -> S x
  intro H.
```

```
1 subgoal

  P, Q, R : Prop
  U : Set
  a : U
  S : U -> Prop
  x : U
  H : S x
  ============================
  S x
  assumption.
No more subgoals.
Qed.
forall_i is defined
```

Coq

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

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

Coq

```
Example forall_e: (forall x: U, S x) -> S a.
Proof.
  intro H.
  apply H.
Qed.
```

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).
```

Coq

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.
```

Coq

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

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

Coq

Im Detail:

Coq

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

  P, Q, R : Prop
  U : Set
  a : U
  S : U -> Prop
  ============================
  (forall x : U, S x) -> exists x : U, S x
Proof.
```

```
  intro H.
1 subgoal

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

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

Coq

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

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

Coq

```
Example exists_e: (exists x: U, S x) -> ~(forall x: U, ~S x).
Proof.
  intro H.
 intro H1.
  elim H.
  assumption.
Qed.
```

Coq

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

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

Coq

```
Theorem forall_not_exists_not: (forall x: U, S x) ->
                                 ~(exists y: U, ~S y).
Proof.
  intro H.
  intro H1.
  elim H1.
  intros x H2.
  elim H2.
  apply H.
Qed.
```

Coq

2.7 Gleichheit

Um die Beispiele für die Gleichheit zeigen zu können, brauchen wir ein paar Objekte des Universums:

Coq

```
Variables t t1 t2: U.
```

Coq

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

Coq

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

Coq

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 t2` durch `t1` ersetzt.

Coq

```
Example equal_e: t1 = t2 /\ S t1 -> S t2.
Proof.
  intro H.
  destruct H as [H1 H2].
  rewrite <- H1.
  assumption.
Qed.
```

Coq

In diesem Beispiel haben wir wieder die zerlegende Bindung (*Destructuring*) eingesetzt.

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.
```
Coq

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.
```
Coq

Aber für die Aussagen, die wir bewiesen haben, geschieht eine sehr wesentliche Verallgemeinerung. Wir haben zum Beispiel die Variablen P und Q vom Type `Prop` lokal deklariert und im Beweis der Kommutativität des Junktors $\lor$ 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
```
Coq

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.
```
Coq

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) -> ~ (exists y : U, ~ 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 intutionistischen 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.
```

Coq

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

Coq

```
End Gentzen.
```

Coq

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).
```

Coq

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.
```

Coq

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

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 offensichtliches 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 ; CASTÉRAN, 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