Einführung in die konstruktive Logik

Andreas Abel

Oktober 2007

Kapitel 1

Aussagenlogik

1.1 Motivation

Disjunktionseigenschaft. Wenn Aussage $A \vee B$ gilt, wüsste man gerne, ob nun A oder B wahr ist. Im Allgemeinen kann einem in der klassischen Logik dieser Wunsch jedoch nicht erfüllt werden. Das Gesetz vom ausgeschlossenen Dritten (lat. *tertium non datur* (TND), engl. *law of the excluded middle*) $A \vee \neg A$ gilt jedoch für beliebige Aussagen A, ohne dass man einen Beweis für A oder sein Gegenteil hätte. Spekulation: Das entspricht philosophisch der Postulation eines logisch allwissenden Gottes.

Man kann sich jedoch auch auf den Standpunkt des *Konstruktivisten* stellen: Wahr ist nur, was bewiesen ist. Dann erfordert die Wahrheit von $A \vee \neg A$ entweder einen Beweis von A oder von $\neg A$, also kann TND nicht allgemein gelten. Während die klassische Logik bereits im Altertum formalisiert wurde (Aristoteles), ist die konstruktive (oder *intuitionistische*) Logik eine Entwicklung des 20. Jahrhunderts, angestoßen durch die große Grundlagenkrise der Mathematik um 1900.

1.2 Brouwer-Heyting-Kolmogorov-Interpretation

Stellen wir uns nun auf den intuitionistischen Standpunkt. Was gilt nun als Beweis einer Aussage A? Kolmogorov [Kol32] sieht eine Aussage als Aufgabenstellung an und einen Beweis als eine Vorschrift, wie die Lösung zu erlangen ist.

- Ein Beweis von $A \wedge B$ ist ein Paar (p, q), wobei p ein Beweis von A und q ein Beweis von B ist.
- Ein Beweis von $A \vee B$ ist ein Paar (b, p), wobei b ein Bit ist und p ein Beweis von A, falls $b = 0$, und ein Beweis von B, falls $b = 1$.

- Ein Beweis von $A \Rightarrow B$ ist ein Programm f, dass einen beliebigen Beweis von A in einen Beweis von B transformiert.

- Ein Beweis von $\neg A$ ist ein Programm f, dass einen beliebigen Beweis von A in einen Beweis von $\bot$ transformiert.

- Der Beweis von $\top$ ist trivial.

Diese Definition kann auch als Vorschrift zur Vereinfachung von Aufgabenstellungen gelesen werden, z. B., "um die Aufgabe $A \wedge B$ zu lösen, löse sowohl A als auch B", "um die Aufgabe $A \vee B$ zu lösen, entscheide dich, entweder A oder B zu lösen". Die Aufgabe $\top$ kann als schon gelöst betrachtet werden. Die Aufgabe $\bot$ kann nicht weiter vereinfacht werden, d.h., es gibt keinen kanonischen Beweis von $\bot$.

Mehr zur BHK-Interpretation unter "Beweisterme".

Referenz: Andrei Kolmogorov, *Zur Deutung der intuitionistischen Logik* (1932) [Kol32].

1.3 Natürliches Schließen

Engl. natural deduction (ND).

Aussagen:

$$A, B, C ::= P \mid \top \mid \bot \mid A \wedge B \mid A \vee B \mid A \Rightarrow B.$$

Die Negation $\neg A$ ist definiert als $A \Rightarrow \bot$.

Der Kalkül des Natürlichen Schließens

$$\frac{\begin{array}{c}\overline{A}\\ \vdots\\ B\end{array}}{A \Rightarrow B}\Rightarrow\mathsf{I} \qquad \frac{A \Rightarrow B \qquad A}{B}\Rightarrow\mathsf{E}$$

$$\frac{A \qquad B}{A \wedge B}\wedge\mathsf{I} \qquad \frac{A \wedge B}{A}\wedge\mathsf{E}_1 \qquad \frac{A \wedge B}{B}\wedge\mathsf{E}_2$$

$$\frac{A}{A \vee B}\vee\mathsf{I}_1 \qquad \frac{B}{A \vee B}\vee\mathsf{I}_2$$

$$\frac{A \vee B \qquad \begin{array}{c}\overline{A}\\ \vdots\\ C\end{array} \qquad \begin{array}{c}\overline{B}\\ \vdots\\ C\end{array}}{C}\vee\mathsf{E}$$

$$\frac{}{\top}\top\mathsf{I} \qquad \frac{\bot}{C}\bot\mathsf{E}$$

Dabei entsprechen die Einführungsregeln (*introduction rules*) mit Namenssuffix I den BHK-Konstruktionsprinzipien für Beweise. Die Eliminationsregeln (*elimination rules*) mit Suffix E sind nötig, um hypothetische Beweise zu zerlegen, um z.B. aus einem angenommenen Beweis von $A \wedge B$ einen Beweis von A zu machen. Einen Beweis mit einer I-Regel zu konstruieren und unmittelbar danach mit einer E-Regel wieder zu zerlegen, stellt einen Umweg dar, und kann abgekürzt werden (siehe Beweisreduktionen).

1.4 Natürliches Schließen in Tutch

Tutch (*tut*orial proof *ch*ecker) ist ein Beweisprüfer für natürliches Schließen zur Übungszwecken. Download und Dokumentation siehe:

`http://www.tcs.ifi.lmu.de/~abel/tutch`

Tutch ist in SML implementiert und benötigt einen aktuellen Standard ML of New Jersey Compiler zur Installation. Entwicklung eines Beweises in Tutch: Tutch kann auch unvollständige Beweise prüfen, meldet dann aber einen Fehler der auf die Lücke zeigt. Dieses Feature machen wir uns zunutze und entwickeln einen Beweis inkrementell:

```
proof andI : A => B => A & B =
begin
```

Formel	in Tutch
$\top$	`T`
$\bot$	`F`
$A \wedge B$	`A & B`
$A \vee B$	`A | B`
$A \Rightarrow B$	`A => B`
$\neg A$	`~ A`
$A \Leftrightarrow B$	`A <=> B`

Tabelle 1.1: Propositionale Formeln in Tutch.

```
  A => B => A & B
end;
```

Dies ist die minimale Eingabe, die Tutch verarbeiten kann. `proof`, `begin`, `end` sind Schlüsselwörter. Wir entwickeln einen Beweis der Aussage `A => B => A & B` und geben ihm den Namen `andI`. Zwischen `begin` und `end` steht der Beweis als Liste von Aussagen, die letzte Aussage muss mit der zu beweisenden übereinstimmen.

Nach Aufruf von `tutch andI.tut` erhalten wir folgendes Ergebnis:

```
TUTCH 0.52 beta, $Date: 2002/10/24 19:25:49 $
[Opening file andI.tut]
Proving andI: A => B => A & B ...
andI.tut:4.3-4.18 Error:
Unjustified line .  |-  A => B => A & B
Assuming this line, checking remainder...
Proof incomplete
[Closing file andI.tut]
```

Die Datei ist also syntaktisch korrekt, aber der Beweis ist unvollständig. Um die Implikation `A => (B => A & B)` zu beweisen, nehmen wir `A` an und beweisen unter dieser Annahme `B => A & B`. In Tutch gibt es dafür die eckigen Klammern: Die erste Aussage nach der öffnenden Klammer ist die Annahme, die letzte Aussage vor der schliessenden Klammer die Konklusion.

```
proof andI: A => B => A & B =
begin
[ A;

  B => A & B ];
A => B => A & B
end;
```

Nun meldet Tutch:

```
Unjustified line A  |-  B => A & B
```

Weiter nehmen wir `B` an und beweisen `A & B`:

```
proof andI: A => B => A & B =
begin
[ A;
  [ B;
    A & B ];
  B => A & B ];
A => B => A & B
end;
```

Der Beweis ist fertig, da `A & B` aus den sichtbaren, darüberliegenden Aussagen durch Anwendung einer einzigen Regel, Und-Einführung, folgt. Mit `tutch -v andI.tut` erhalten wir ein Protokoll der Beweisprüfung:

```
Proving andI: A => B => A & B ...
  1  [ A;
  2    [ B;
  3      A & B ];              by AndI 1 2
  4    B => A & B ];           by ImpI 3
  5  A => B => A & B           by ImpI 4
QED
```

Tutch hat jede Aussage (außer den Annahmen) mit der Regel annotiert, die die Aussage beweist, mit den Nummern der Aussagen, die an diesem Schritt direkt beteiligt sind.

1.5 Natürliches Schließen mit expliziten Annahmen

Besonders wenn man etwas über den Kalkül des natürlichen Schließens zeigen möchte, ist es gut, die aktuell sichtbaren Annahmen zu jeder Zeit explizit zu nennen. Damit ergeben sich folgende Regeln:

Der Kalkül des Natürlichen Schließens mit expliziten Annahmen $\Gamma \vdash A$. Γ sei eine Menge von Aussagen, die aktuellen Annahmen (Hypothesen).

$$\frac{A \in \Gamma}{\Gamma \vdash A}\,\mathsf{hyp} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\Rightarrow\!\mathsf{I} \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}\Rightarrow\!\mathsf{E}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\wedge\mathsf{I} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\wedge\mathsf{E}_1 \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\wedge\mathsf{E}_2$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\vee\mathsf{I}_1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\vee\mathsf{I}_2$$

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}\vee\mathsf{E}$$

$$\frac{}{\Gamma \vdash \top}\top\mathsf{I} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash C}\bot\mathsf{E}$$

Es ist eine Regel hyp hinzu gekommen, die besagt: Jede Annahme gilt als bewiesen. Die Regeln $\Rightarrow\!\mathsf{I}$ und $\vee\mathsf{E}$ fügen neue Annahmen hinzu (liest man die Regeln von unten nach oben), alle anderen Regeln lassen Γ unberührt.

1.6 Beweisterme

Nach der BHK-Interpretation ist ein Beweis ein Programm zur Lösung einer Aufgabe. Im Folgenden formalisieren wir diese Programme.

Beweisterme (Church-Stil):

$r, s, t ::= x \mid \lambda x^A.t \mid r\,s$	einfach getypter λ-Kalkül (für $\Rightarrow$)
$\mid (r, s) \mid \mathsf{fst}\,r \mid \mathsf{snd}\,r$	Paare und Projektionen (für $\wedge$)
$\mid \mathsf{inl}_B\,t \mid \mathsf{inl}_A\,t$	Injektionen und...
$\mid \mathsf{case}\,r\,\mathsf{of}\,\mathsf{inl}\,x^A \Rightarrow s \mid \mathsf{inr}\,y^B \Rightarrow t$	...Fallunterscheidung (für $\vee$)
$\mid ()$	leeres Tupel (für $\top$)
$\mid \mathsf{abort}_C\,r$	Ausnahme (*exception*) (für $\bot$)

Typisierung: Γ sei eine Menge von Variable-Aussage-Paaren, geschrieben $x : A$. Wir definieren induktiv ein Urteil $\Gamma \vdash t : C$ das besagt: *Unter den Annahmen*

Γ *ist der Term* t *ein gültiger Beweis der Aussage* C*.*

$$\frac{(x{:}A) \in \Gamma}{\Gamma \vdash x : A}\,\mathsf{hyp}$$

$$\frac{\Gamma, x{:}A \vdash t : B}{\Gamma \vdash \lambda x^A.\, t : A \Rightarrow B}\Rightarrow\!\mathsf{I} \qquad \frac{\Gamma \vdash r : A \Rightarrow B \qquad \Gamma \vdash s : A}{\Gamma \vdash r\, s : B}\Rightarrow\!\mathsf{E}$$

$$\frac{\Gamma \vdash r : A \qquad \Gamma \vdash s : B}{\Gamma \vdash (r,\, s) : A \wedge B}\wedge\mathsf{I} \qquad \frac{\Gamma \vdash r : A \wedge B}{\Gamma \vdash \mathsf{fst}\, r : A}\wedge\mathsf{E}_1 \qquad \frac{\Gamma \vdash r : A \wedge B}{\Gamma \vdash \mathsf{snd}\, r : B}\wedge\mathsf{E}_2$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathsf{inl}_B\, t : A \vee B}\vee\mathsf{I}_1 \qquad \frac{\Gamma \vdash t : B}{\Gamma \vdash \mathsf{inr}_A\, t : A \vee B}\vee\mathsf{I}_2$$

$$\frac{\Gamma \vdash r : A \vee B \qquad \Gamma, x{:}A \vdash s : C \qquad \Gamma, y{:}B \vdash t : C}{\Gamma \vdash \mathsf{case}\, r\ \mathsf{of}\ \mathsf{inl}\, x^A \Rightarrow s \mid \mathsf{inr}\, y^B \Rightarrow t : C}\vee\mathsf{E}$$

$$\frac{}{\Gamma \vdash () : \top}\top\mathsf{I} \qquad \frac{\Gamma \vdash r : \bot}{\Gamma \vdash \mathsf{abort}_C\, r : C}\bot\mathsf{E}$$

Die Propositions-Annotationen in den Beweistermen zu $\Rightarrow\!\mathsf{I}$ und $\bot\mathsf{E}$ sind so gewählt, dass man die bewiesene Proposition C zu einem Beweisterm t durch strukturelle Rekursion über t mühelos reproduzieren kann.

Übung 1 Gegeben ein Typisierungskontext Γ und ein Beweisterm t im Church-Stil berechnet die Funktion $\mathsf{typeOf}(\Gamma \vdash t)$ die von t bewiesene Aussage A durch Rekursion über t. Falls t nicht wohlgetypt ist in Γ, wird $\emptyset$ zurückgegeben. Definieren Sie typeOf, in Anlehnung an obige Regeln.

In den Einführungsregeln (I, engl. *introduction*) erkennen wir die Definition eines kanonischen Beweises nach Brouwer-Heyting-Kolmogorov wieder. Es gibt keinen kanonischen Beweis der Absurdität, also keine Regel $\bot\mathsf{I}$. Die Beseitigungsregeln (E, engl. *elimination*) erklären, wie man vorhandene Beweise verwenden kann, um die Beweisaufgabe zu lösen. Auf den trivialen Beweis von $\top$ hat man keine Arbeit verwendet, kann ihn also auch nicht sinnvoll benutzen, deshalb keine Regel $\top\mathsf{E}$.

1.7 Beweisterme in Tutch

In Tutch werden Beweisterme sehr ähnlich geschrieben, es gibt die folgenden Unterschiede:

1. Keine Propositions-Annotationen.
2. $\lambda x.t$ wird geschrieben als `fn x => t` (wie in SML).

3. Fallunterscheidung wird mit `end` abgeschlossen, also

```
case r of inl x => s | inr y => t end
```

Ein Tutch-Beweis kann mit Beweistermen annotiert werden, die von Tutch dann auf Korrektheit geprüft werden. Z.B.

```
annotated proof andI : A => B => A & B =
begin
[ x : A;
  [ y : B;
    (x,y) : A & B ];
  fn y => (x,y) : B => A & B ];
fn x => fn y => (x,y) : A => B => A & B
end;
```

Das Schlüsselwort `annotated` zeigt an, dass nun jede Aussage einen Beweisterm erfordert. Annahmen werden mit Variablen annotiert, hergeleitete Aussagen mit einem zusammengesetzten Term. Dabei ist jeder Term ein voller Beweis für die dahinterstehende Aussage, dadurch ergibt sich eine gewaltige Redundanz: Jeder Teilbeweis muss vollständig wiederholt werden.

1.8 Coq

Das Beweissystem Coq basiert auf einem Kalkül ähnlich dem natürlichen Schließen.

```
Section Example1.

Variables (A B C: Prop).

Theorem currying : and A B -> C <-> A -> B -> C.
split.
 intros f a b.
 apply f.
 split. assumption. assumption.
intros f ab.
inversion ab.
apply f. assumption. assumption.
Qed.

Theorem orElim : or A B -> C <-> and (A -> C) (B -> C).
split.
 intros f.
 split.
  intros a. apply f. left. exact a.
 intros b. apply f. right. exact b.
intros gh d.
inversion_clear gh as [ g h ].
inversion_clear d as [ a | b ].
 apply g. assumption.
apply h. assumption.
Qed.

End Example1.
```

Eine Coq-Datei kann mehrere `Section`s enthalten, die jeweils lokale Annahmen (z.B: `Variables`) deklarieren dürfen. Das ist dem `theory`-Mechanismus von PVS ähnlich. Die Bibliothek `Coq.Init.Logic`, die automatisch geladen wird, enthält die Definition der logischen Konnektive. `Prop` ist der Typ der Aussagen (war `boolean` in PVS).

Tabelle 1.3 zeigt die den Beweisregeln des natürlichen Schließens entsprechende Coq-Taktiken. Die Entsprechung ist nicht eins-zu-eins, die `inversion`-Taktik entspricht eher einer Links-Regel des Sequenzenkalküls denn einer Eliminationsregel des natürlichen Schließens. Varianten:

- `intros x y z` führt gleich 3 Hypothesen ein.
- `apply f` führt auch mehrere $\Rightarrow$E-Schritte auf einmal aus, z.B., falls $f : A \Rightarrow B \Rightarrow C$ und das aktuelle Ziel C ist, dann werden zwei neue Ziele A

Formel	in Coq	Alternative
$\top$	`True`	
$\bot$	`False`	
$A \wedge B$	`A /\ B`	`and A B`
$A \vee B$	`A \/ B`	`or A B`
$A \Rightarrow B$	`A -> B`	
$\neg A$	`~ A`	`not A`
$A \Leftrightarrow B$	`A <-> B`	`iff A B`

Tabelle 1.2: Propositionale Formeln in Coq.

Regel	Coq-Taktik	Alternative
hyp	`exact x`	`assumption`
$\Rightarrow$I	`intros x`	
$\Rightarrow$E	`apply r`	
$\top$I	`split`	
$\wedge$I	`split`	
$\wedge$E	`inversion x`	`elim x`
$\vee$I$_1$	`left`	
$\vee$I$_2$	`right`	
$\vee$E	`inversion x`	`elim x`
$\bot$E	`inversion x`	`elim x`
weak	`clear x`	

Tabelle 1.3: Natürliches Schließen in Coq.

und B erzeugt.

- `split` kann auch angewendet werden, falls das Ziel von der Form $A_1 \to \cdots \to A_n \to C$ ist und C eine Konjuktion. Dann werden vor der Aufsplittung die Aussagen A_i als Hypotheses eingeführt. Dasselbe gilt analog für `left` und `right`.

- `inversion_clear h` zerlegt wie `inversion h` die Hypothese h in ihre Bestandteile, zusätzlich wird noch h entfernt. Beide Taktiken kann noch eine Liste von Bezeichnern `[ x1 ... xn ]` für die neuen Hypothesen mitgeliefert werden. Entstehen durch die Zerlegung mehrere Unterziele (wie z.B. bei $\vee$E), so müssen Bezeichner für jedes Unterziel mitgeliefert werden. Die Bezeichnerliste wird dann durch `|` entsprechend partitioniert.

- `elim x` kann auch auf Hypothesen der Form $A_1 \to \cdots \to A_n \to C$ und C Konjunktion, Disjunktion oder Falsum. Dann werden die A_i als neue Ziele hinzugefügt.

- `cut (A)` fügt die Annahme A hinzu, die dann separat noch bewiesen werden muss.

Beweisterme. Wurde `Theorem bla : A` in Coq bewiesen, so kann der Beweisterm mit `Print bla` (oder auch `Check bla`) eingesehen werden. In Coq können Beweisterme auch direkt eingegeben werden. Hier haben wir einige Aussagen mit Termen bewiesen, u.a. die des letzten Beispiels:

```
Section Example2.

Variables (A B C: Prop).

Definition curry  : (and A B -> C) -> A -> B -> C
  := fun f a b => f (conj a b).

Definition curry'
  := fun (f : and A B -> C) (a : A) (b : B) => f (conj a b).

Definition uncurry : (A -> B -> C) -> and A B -> C
  := fun f p => f (proj1 p) (proj2 p).

Definition join :  and (A -> C) (B -> C) -> or A B -> C
  := fun gh d =>
     match d with
     | or_introl a => proj1 gh a
     | or_intror b => proj2 gh b
     end.

Definition fsplit : (or A B -> C) -> and (A -> C) (B -> C)
```

```
  := fun f => conj (fun a => f (or_introl B a))
                   (fun b => f (or_intror A b)).

Definition abort : False -> C
  := fun h => match h with end.

Definition delay : C -> (True -> C)
  := fun c i => c.
Definition force : (True -> C) -> C
  := fun f => f I.

End Example2.
```

Die `Definition` eines Beweistermes kann entweder die zu beweisende Proposition mit angeben (Bsp. `curry`) oder vom System inferieren lassen (Bsp. `uncurry`). Dann müssen aber die Aussagen zu den (wenigstens einigen) Hypothesenvariablen angegeben werden. (Die Annotation der Variable `a` und `b` ist hier überflüssig.) Besondere Vorsicht ist bei `or_introl` und `or_intror` geboten: Sie

Beweisterm	in Coq
$\lambda x^A.\, t$	`fun (x : A) => t`
$r\, s$	`r s`
$()$	`I`
(r, s)	`conj r s`
$\mathsf{fst}\, r$	`proj1 r`
$\mathsf{snd}\, r$	`proj2 r`
$\mathsf{inl}\, r : A \vee B$	`or_introl B r`
$\mathsf{inr}\, r : A \vee B$	`or_intror A r`
$\mathsf{case}\, r\, \mathsf{of}\, \mathsf{inl}\, x^A \Rightarrow s \mid \mathsf{inr}\, y^B \Rightarrow t$	`match r with | or_introl x => s | or_intror y => t end`
$\mathsf{abort}_C\, r$	`match r return C with end`

Tabelle 1.4: Beweisterme in Coq.

benötigen als erstes Argument die Aussage der Disjunktion, die *nicht* vom Beweisterm (zweites Argument) beweisen wird. In einem `match`-Ausdruck muss dieses zusätzliche Argument aber wegelassen werden. Die $\bot$E-Term `abort` wird als Fallunterscheidung über 0 Alternativen repräsentiert. Die durch ein `match` zu beweisende Aussage `C` kann immer mit `return C` angegeben werden, muss aber nicht.

1.9 Herleitbare und zulässige Regeln

Wir verwenden J als Abkürzung für ein Urteil (engl. *judgement*), z.B. $\Gamma \vdash A$ oder $\Gamma \vdash t : A$.

Definition 2 (Herleitbare Regel) Eine Beweisregel

$$\frac{J_1 \dots J_n}{J}$$

heisst *herleitbar* (engl. *derivable*), falls es einen Beweisbaum von J mit Blättern $J_1, \dots, J_n$ gibt.

Beispiel 3 (Schnittregel) Die Regel

$$\frac{\Gamma \vdash A \qquad \Gamma, A \vdash B}{\Gamma \vdash B}\,\mathsf{cut}$$

ist im Kalkül des Natürlichen Schließens herleitbar. Wir geben eine Herleitung mit Beweistermen an:

$$\frac{\Gamma \vdash s : A \qquad \dfrac{\Gamma,\, x{:}A \vdash t : B}{\Gamma \vdash \lambda x^A.\, t : A \Rightarrow B}\,\Rightarrow\!\mathsf{I}}{\Gamma \vdash (\lambda x^A.\, t)\, s : B}\,\Rightarrow\!\mathsf{E}$$

Definition 4 (Zulässige Regel) Eine Beweisregel

$$\frac{J_1 \dots J_n}{J}$$

heisst *zulässig* (engl. *admissible*), falls aus den Herleitungen von $J_1, \dots, J_n$ eine Herleitung von J berechnet werden kann.

Beispiel 5 Die beiden Regeln

$$\frac{\Gamma \vdash t : A}{\Gamma, \Gamma' \vdash t : A}\,\mathsf{weak} \qquad \frac{\Gamma, x{:}A \vdash t : B \qquad \Gamma \vdash s : A}{\Gamma \vdash t[s/x] : B}\,\mathsf{subst}$$

sind zulässig. Beweis siehe folgende Lemmata.

Lemma 6 (Abschwächung) *Wenn* $\Gamma \vdash t : A$*, dann* $\Gamma, \Gamma' \vdash t : A$*.*

Beweis. Durch Induktion über die Herleitung von $\Gamma \vdash t : A$. □

Lemma 7 (Substitution) *Wenn* $\Gamma \vdash s : A$ *und* $\Gamma, x : A \vdash t : B$*, dann* $\Gamma \vdash t[s/x] : B$*.*

Beweis. Durch Induktion über die zweite Herleitung. □

1.10 Fragment $\Rightarrow\wedge\top$: Beweisreduktionen und normale Beweise

1.10.1 Beweisreduktionen

β-Reduktionen. Die Einführung eines logischen Konnektives, unmittelbar gefolgt von dessen Beseitigung, stellt einen Umweg im Beweis dar, der schematisch entfernt werden kann.

Konjunktion.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash r : A\end{array} \qquad \begin{array}{c}\vdots\\ \Gamma \vdash s : B\end{array}}{\Gamma \vdash (r,s) : A \wedge B}\,\wedge\mathsf{I}}{\Gamma \vdash \mathsf{fst}\,(r,s) : A}\,\wedge\mathsf{E}_1 \quad \longrightarrow \quad \begin{array}{c}\vdots\\ \Gamma \vdash r : A\end{array}$$

Analog für $\wedge\mathsf{I}$ gefolgt von $\wedge\mathsf{E}2$

Implikation.

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x{:}A, \Gamma' \vdash x : A}\,\mathsf{hyp}}{\begin{array}{c}\vdots\\ \Gamma, x{:}A \vdash t : B\end{array}}}{\Gamma \vdash \lambda x t : A \to B}\,\Rightarrow\mathsf{I} \qquad \begin{array}{c}\vdots\\ \Gamma \vdash s : A\end{array}}{\Gamma \vdash (\lambda x t)\, s : B}\,\Rightarrow\mathsf{E} \quad \longrightarrow \quad \dfrac{\begin{array}{c}\vdots\\ \Gamma, \Gamma' \vdash s : A\end{array}}{\begin{array}{c}\vdots\\ \Gamma \vdash t[s/x] : B\end{array}}$$

Tautologie. Für $\top$ gibt es keine Beseitigungsregel, daher keine β-Reduktion.

Zusammenfassung β-Reduktion.

$$\begin{array}{lll} (\beta_{\wedge 1}) & \mathsf{fst}\,(r,s) & \longrightarrow \quad r \\ (\beta_{\wedge 2}) & \mathsf{snd}\,(r,s) & \longrightarrow \quad s \\ \\ (\beta_{\Rightarrow}) & (\lambda x^A t)\, s & \longrightarrow \quad t[s/x] \end{array}$$

Der jeweils linke Term heißt *Redex*, der rechte *Redukt*.

An jeder Stelle des Beweisbaumes darf eine Reduktion durchgeführt werden. Zu den fünf Reduktionsaxiomen kommt noch folgende Kongruenz-Regel hinzu:

$$\dfrac{t \longrightarrow t'}{C[t] \longrightarrow C[t']}$$

Dabei ist $C[\bullet]$ ein *Kontext*, also ein Term mit genau einem Loch $\bullet$, und $C[t]$ bezeichnet die Einsetzung des Terms t für das Loch. Es genügt, die Regel für Kontexte der Tiefe 1 anzunehmen.

$$C \quad ::= \quad \lambda x.\bullet \mid \bullet\, s \mid r\, \bullet \mid (\bullet, r) \mid (s, \bullet) \mid \mathsf{fst}\, \bullet \mid \mathsf{snd}\, \bullet$$

Theorem 8 (Subject reduction) *Wenn* $\Gamma \vdash t : A$ *und* $t \longrightarrow t'$*, dann auch* $\Gamma \vdash t' : A$*.*

Beweis. Durch Induktion über die Herleitung von $t \longrightarrow t'$. □

1.10.2 Normale Beweise

Ein *normaler* Beweis ist ein Beweis ohne Umwege (β-Redexe). Ein *neutraler* Beweis kann in einen Normalen anstelle einer Hypothese eingesetzt werden, ohne dass ein Redex entsteht.

Definition 9 (Normal und neutral)

1. Ein Beweisterm t heißt normal, falls es keinen Term t' gibt, so dass $t \longrightarrow t'$.
2. Ein Beweisterm s heißt neutral, falls er zusätzlich nicht in einer Einführung endet. (Also keine λ-Abstraktion und kein Tupel ist.)

Charakterisierung der normalen und neutralen Beweise.

$$\begin{array}{ll} \Gamma \vdash t \downarrow A & t \text{ ist ein neutraler Beweis von } A \\ \Gamma \vdash t \uparrow A & t \text{ ist ein normaler Beweis von } A \end{array}$$

Die erste Regel besagt: Jeder neutrale Term ist auch normal.

$$\frac{\Gamma \vdash r \downarrow A}{\Gamma \vdash r \uparrow A}\,{\downarrow\uparrow} \qquad \frac{(x\!:\!A) \in \Gamma}{\Gamma \vdash x \downarrow A}\,\mathsf{hyp}$$

$$\frac{\Gamma, x\!:\!A \vdash t \uparrow B}{\Gamma \vdash \lambda x^A.\, t \uparrow A \Rightarrow B}\,{\Rightarrow}\mathsf{I} \qquad \frac{\Gamma \vdash r \downarrow A \Rightarrow B \qquad \Gamma \vdash s \uparrow A}{\Gamma \vdash r\, s \downarrow B}\,{\Rightarrow}\mathsf{E}$$

$$\frac{\Gamma \vdash r \uparrow A \qquad \Gamma \vdash s \uparrow B}{\Gamma \vdash (r,\, s) \uparrow A \wedge B}\,{\wedge}\mathsf{I} \qquad \frac{\Gamma \vdash r \downarrow A \wedge B}{\Gamma \vdash \mathsf{fst}\, r \downarrow A}\,{\wedge}\mathsf{E}_1 \qquad \frac{\Gamma \vdash r \downarrow A \wedge B}{\Gamma \vdash \mathsf{snd}\, r \downarrow B}\,{\wedge}\mathsf{E}_2$$

$$\frac{}{\Gamma \vdash () \uparrow \top}\,\top\mathsf{I}$$

Theorem 10 (Korrektheit)

1. $\Gamma \vdash t \downarrow C$ *gdw.* $\Gamma \vdash t : C$ *und* t *neutral.*
2. $\Gamma \vdash t \uparrow C$ *gdw.* $\Gamma \vdash t : C$ *und* t *normal.*

Beweis. Simultan durch Induktion nach t. □

Sei $\longrightarrow^*$ die reflexiv-transitive Hülle von $\longrightarrow$.

Theorem 11 (Normalisierung) *Wenn* $\Gamma \vdash t : A$*, dann gibt es ein* t' *mit* $t \longrightarrow^* t'$ *und* $\Gamma \vdash t \uparrow A$*.*

Beweis. Wir definieren ein Maß auf Beweisen, das mit jedem Reduktionsschritt abnimmt. Dies kann nicht die Termgröße sein, den durch die Entfernung des Redexes $(\lambda x t)\, s$ kann der Term größer werden (Verdopplung von s. Z.B.: Der Redex von $(\lambda f \lambda x.\, f\,(f\,(f\,x)))\,(\lambda y.y)$ ist $\lambda x.\,(\lambda y.y)\,((\lambda y.y)\,((\lambda y.y)\,x))$. Es sind neue Redexe entstanden, die jedoch von geringerem *Rang* sind, wobei der Rang der Typ der substitutierten Variable ist. Sei $f : A \Rightarrow A$, dann ist der Rang des ursprünglichen Redexes $A \Rightarrow A$, die neuen Redexe $(\lambda y.y)\,\_$ haben nur Rang A, den Typen von y.

Wir definieren die Ordnung einer Aussage A rekursiv durch $|\top| = |P| = 0$, $|A \Rightarrow B| = \max(1 + |A|, |B|)$ und $|A \wedge B| = \max(|A|, |B|)$. Der Rang eines Redexes der Form $(\lambda x^A t)\, s$ sei $|A|$, ein Redex der Form $\mathsf{fst}\,(r, s) : A$ habe den Rang $|A|$. Gegeben ein Term t, sei $\#(t, 0)$ die Größe von t (Anzahl der Knoten in der Baumdarstellung von t) und $\#(t, n+1)$ die Anzahl der Redexe vom Rang n in t. Wir setzen $t > t'$ falls es ein $n \geq 0$ gibt mit $\#(t, n) > \#(t', n)$ und $\#(t, m) \geq \#(t', m)$ für alle $m > n$. Die so definierte (lexikographische) Ordnung ist wohlfundiert. Wir zeigen nun: Wenn $\Gamma \vdash t : A$ und $t \longrightarrow t'$ entfernt einen Redex r, dessen Rang den Rang jedes Redexes r' in einem echten Subterm von r übersteigt, dann $t > t'$. Damit erreicht man die Normalform in endlich vielen Schritten.

Einen solchen Redex findet man immer, jeder innerste Redex (also dessen Subterme keinen Redex enthalten), ist geeignet.

Reduzieren wir einen $\wedge$-Redex, z.B. $(t f s t\,(r, s))\, t \longrightarrow r\, t$, so wird der Term kleiner. Es kann zwar ein neuer Redex, z.B. $r\, t$, entstanden sein, dessen Rang ist jedoch kleiner.

Der kritische Fall ist der $\Rightarrow$-Redex $(\lambda x^A t)\, s \longrightarrow t[s/x]$. Falls s keine Einführung ist, entstehen keine neuen Redexe, wir haben zwar den Term vergrößert und Redexe von in s vervielfacht, jedoch haben sie einen Rang $< |A|$. Ist s eine Einführung, können zusätzlich noch neue Redexe entstehen, diese haben jedoch kleineren Rang. Z.B. sei $s = \lambda y^B.s'$, dann ist zwingenderweise $A = B \Rightarrow C$, und enthalte t eine Subterm $x\, r$. Dann ensteht der neue Redex $(\lambda y^B.s')\, r$, jedoch nur vom Rang $|B|$. □

Alternativ können wir das Theorem auch direkt durch Induktion über t zeigen [JM03]. Für den Applikationsfall brauchen wir $\Gamma \vdash r \uparrow A \Rightarrow B$ und $\Gamma \vdash s \uparrow A$ impliziert $r\, s \longrightarrow^* t'$ und $\Gamma \vdash t' \uparrow B$. Dies folgt aus dem Substitutionslemma:

Lemma 12 (Substitution) *Wenn* $\Gamma, x : A, \Gamma' \vdash t \uparrow C$ *und* $\Gamma \vdash s \uparrow A$*, dann* $t[s/x] \longrightarrow^* t'$ *und* $\Gamma, \Gamma' \vdash t' \uparrow C$*.*

Beweis. Wir beweisen dies durch lexikographische Induktion über $(|A|, t)$, simultan mit: Wenn $\Gamma, x:A, \Gamma' \vdash t \downarrow C$ und $\Gamma \vdash s \uparrow A$, dann $t[s/x] \longrightarrow^* t'$ und entweder $\Gamma, \Gamma' \vdash t' \downarrow C$, oder $\Gamma, \Gamma' \vdash t' \uparrow C$ und $|C| \leq |A|$.

Im Fall $t = x$ ist $t[s/x] = s$ schon normal und $C = A$.

Im Fall $t = y$ ist $t[s/x] = y$ neutral.

Im Fall $t = t_1\, t_2$ gilt nach Induktionshypothese $t_1[s/x] \longrightarrow^* t_1'$ und $t_2[s/x] \longrightarrow^* t_2'$ mit $\Gamma, \Gamma' \vdash t_2' \uparrow A$. Falls $\Gamma, \Gamma' \vdash t_1' \downarrow B \Rightarrow C$, dann gilt sofort $\Gamma, \Gamma' \vdash t_1'\, t_2' \downarrow C$. Andernfalls ist $t_1' = \lambda y r$ für ein r und $\Gamma, \Gamma', y:B \vdash r \uparrow C$. Da $|B \Rightarrow C| < |A|$ nach Ind.hyp., also $|B| < |A|$, können wir die Ind.hyp. erneut anwenden und erhalten $r[t_2'/y] \longrightarrow^* r'$ und $\Gamma, \Gamma' \vdash r' \uparrow C$. Insgesamt $t_1\, t_2[s/x] \longrightarrow^* t_1' t_2' = (\lambda y.r)\, t_2' \longrightarrow r[t_2'/y] \longrightarrow^* r'$.

Die übrigen Fälle sind unproblematisch. □

Anwendung des Normalisierungs-Satzes: Wenn es keinen normalen Beweis einer Aussage gibt, so ist die Aussage im Kalkül des natürlichen Schließens überhaupt nicht beweisbar, also konstruktiv nicht gültig.

Beispiel 13 (Peirce-Formel) Die Formel $P \equiv ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$ ist eine klassische Tautologie, aber konstruktiv nicht gültig. Man sieht leicht, dass ein normaler Beweis von P scheitern muss.

1.11 Volle Aussagenlogik: Beweisreduktionen und normale Beweise

Bei der Behandlung des natürlichen Schließens haben wir gesehen, dass Einführungsregeln dazu dienen, Beweise für größere Aussagen aus Beweisen kleinerer Aussagen zu konstruieren, und mit Eliminationsregeln hypothetische Beweise zerlegt werden können um sie für die Konstruktion von neuen Beweisen nutzbar zu machen. Der Kalkül allerdings erlaubt auch Umwege, d.h. die Konstruktion eines Beweises gefolgt von seiner sofortigen Zerlegung. Z.B.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash r : A\end{array} \qquad \begin{array}{c}\vdots\\ \Gamma \vdash s : B\end{array}}{\Gamma \vdash (r, s) : A \wedge B}\,\wedge\mathsf{I}}{\Gamma \vdash \mathsf{fst}\,(r, s) : A}\,\wedge\mathsf{E}_1$$

Im Folgenden zeigen wir, wie wir solche Umwege durch Beschränkung der Benutzung der I-Regeln auf die Konstruktionsphase und der E-Regeln auf die Zerlegungsphase verhindern können. Umwegsfreie Beweise bezeichnen wir als *normal*. Danach zeigen wir, wie wir Umwege in Beweisen durch Beweisreduktionen systematisch entfernen können.

1.11.1 Normale Beweise

Es wird sich zeigen, dass normale Beweisterme keiner Annotationen bedürfen, um aus Korrektheit geprüft zu werden. Wir können daher auf reine λ-Terme zurückfallen.

Beweisterme (Curry-Stil):

$$\begin{array}{rcl} r, s, t & ::= & x \mid \lambda x t \mid r\, s \\ & & \mid (r, s) \mid \mathsf{fst}\, r \mid \mathsf{snd}\, r \\ & & \mid \mathsf{inl}\, t \mid \mathsf{inl}\, t \mid \mathsf{case}\, r\ \mathsf{of}\ \mathsf{inl}\, x \Rightarrow s \mid \mathsf{inr}\, y \Rightarrow t \\ & & \mid () \\ & & \mid \mathsf{abort}\, r \end{array}$$

Wir versehen nun den Kalkül des natürlichen Schließens mit zwei *Modi*:

- Konstruktions-/Synthese-Modus: Wir beweisen eine zusammengesetzte Aussage durch Beweise (eines) ihrer Teile. Darunter fallen die Regeln $\Rightarrow\mathsf{I}$, $\wedge\mathsf{I}$, $\vee\mathsf{I}_1$, $\vee\mathsf{I}_2$, $\top\mathsf{I}$. Diesem Modus ist der Pfeil $\uparrow$ zugeordnet, der die Leserichtung der Regeln bei der Beweissuche andeuted: Von der Folgerung (*conclusion*) zu den Voraussetzungen (*premises*). In dieser Leserichtung wird die zu beweisende Aussage kleiner.
- Zerlegungs/Analyse-Modus: Wir zerlegen eine Annahme in ihre Teile. Vorzeigeinstanzen dieser Regel sind nur $\wedge\mathsf{E}_1$ und $\wedge\mathsf{E}_2$. Die Regel hyp macht

eine Annahme für die Analyse verfügbar. Durch die Regel $\downarrow\uparrow$ können wir den Modus wechseln. So kann eine (zerlegte) Annahme in Konstruktionen verwendet werden. Alternative Sichtweise: Komme ich in der Konstruktion nicht mehr weiter, z.B. weil die zu beweisende Formel atomar ist, so muss ich die Formel durch Analyse der Annahmen beweisen.

$$\frac{\Gamma \vdash r \downarrow A}{\Gamma \vdash r \uparrow A}\downarrow\uparrow \qquad \frac{(x{:}A) \in \Gamma}{\Gamma \vdash x \downarrow A}\,\mathsf{hyp}$$

$$\frac{\Gamma, x{:}A \vdash t \uparrow B}{\Gamma \vdash \lambda x t \uparrow A \Rightarrow B}\Rightarrow\!\mathsf{I} \qquad \frac{\Gamma \vdash r \downarrow A \Rightarrow B \qquad \Gamma \vdash s \uparrow A}{\Gamma \vdash r\, s \downarrow B}\Rightarrow\!\mathsf{E}$$

$$\frac{\Gamma \vdash r \uparrow A \qquad \Gamma \vdash s \uparrow B}{\Gamma \vdash (r,\, s) \uparrow A \wedge B}\wedge\mathsf{I} \qquad \frac{\Gamma \vdash r \downarrow A \wedge B}{\Gamma \vdash \mathsf{fst}\, r \downarrow A}\wedge\mathsf{E}_1 \qquad \frac{\Gamma \vdash r \downarrow A \wedge B}{\Gamma \vdash \mathsf{snd}\, r \downarrow B}\wedge\mathsf{E}_2$$

$$\frac{\Gamma \vdash t \uparrow A}{\Gamma \vdash \mathsf{inl}\, t \uparrow A \vee B}\vee\mathsf{I}_1 \qquad \frac{\Gamma \vdash t \uparrow B}{\Gamma \vdash \mathsf{inr}\, t \uparrow A \vee B}\vee\mathsf{I}_2$$

$$\frac{\Gamma \vdash r \downarrow A \vee B \qquad \Gamma, x{:}A \vdash s \uparrow C \qquad \Gamma, y{:}B \vdash t \uparrow C}{\Gamma \vdash \mathsf{case}\, r\, \mathsf{of}\, \mathsf{inl}\, x \Rightarrow s \mid \mathsf{inr}\, y \Rightarrow t \uparrow C}\vee\mathsf{E}$$

$$\frac{}{\Gamma \vdash () \uparrow \top}\top\mathsf{I} \qquad \frac{\Gamma \vdash r \downarrow \bot}{\Gamma \vdash \mathsf{abort}\, r \uparrow C}\bot\mathsf{E}$$

Die Mehrzahl der Eliminationsregeln ($\Rightarrow\mathsf{E}$, $\vee\mathsf{E}$, $\bot\mathsf{E}$) erwähnt beide Modi. Diese Regeln werden nun noch einzeln erklärt:

$\Rightarrow\mathsf{E}$ Um sich eine Hypothese der Form $A \Rightarrow B$ zunutze machen, d.h., zu zerlegen, muss man zuerst A beweisen. Dies muss unter Umständen durch Konstruktionen geschehen und erzeugt keine Umwege. Zum Beispiel wollen wir C aus A und $(A \vee B) \Rightarrow C$ beweisen. Aus der Annahme A konstruieren wir $A \vee B$ mit $\vee\mathsf{I}_1$, danach können wir die Annahme $(A \vee B) \Rightarrow C$ verwenden und erhalten C.

$$\dfrac{\dfrac{y \downarrow (A \vee B) \Rightarrow C \qquad \dfrac{\dfrac{x \downarrow A}{x \uparrow A}\downarrow\uparrow}{\mathsf{inl}\, x \uparrow A \vee B}\vee\mathsf{I}_1}{y\,(\mathsf{inl}\, x) \downarrow C}\Rightarrow\!\mathsf{E}}{y\,(\mathsf{inl}\, x) \uparrow C}\downarrow\uparrow$$

$\vee\mathsf{E}$ Hat man eine *Annahme* der Form $A \vee B$, kann man daraus nicht A oder B folgern, da man nicht weiss, welches der beiden wahr ist. (Ist $A \vee B$ jedoch *konstruiert*, kann man durch Inspektion des Beweises sehen ob A oder B wahr ist.) Will man nun die Annahme $A \vee B$ für den Beweis

einer dritten Aussage C verwenden, so muss man auf beide Möglichkeiten gefasst sein, d.h. man muss einen Beweis von C unter der Annahme von A bereitstellen, und einen Beweis von C unter der Annahme von B. Diese Beweise können im Allgemeinen konstruiert sein, daher der Modus $\uparrow$. Im Folgenden beweisen wir B aus den Annahmen $v : A$ und $x : B \vee (A \Rightarrow B)$.

$$\dfrac{x \downarrow B \vee (A \Rightarrow B) \qquad \dfrac{\dfrac{}{y \downarrow B}\,\mathsf{hyp}}{y \uparrow B}\downarrow\uparrow \qquad \dfrac{\dfrac{\dfrac{}{z \downarrow A \Rightarrow B}\,\mathsf{hyp} \qquad \dfrac{v \downarrow A}{v \uparrow A}\downarrow\uparrow}{z\,v \downarrow B}\Rightarrow\mathsf{E}}{z\,v \uparrow B}\downarrow\uparrow}{B}\vee\mathsf{E}$$

Die lokalen Annahmen y und z werden durch $\vee\mathsf{E}$ eingeführt.

$\bot\mathsf{E}$ Die Form dieser Regel ist analog zu $\vee\mathsf{E}$, wenn man das Falsum $\bot$ als *Disjunktion mit null Alternativen* betrachtet. Die Annahme $\bot$ ist nicht weiter zerlegbar; jedoch kann jede Aussage C durch aus $\bot$ trivial bewiesen werden: *ex falsum quot libet*.

Führen wir folgende Bezeichnungen ein: Falls $\Gamma \vdash t \uparrow C$ eine gültige Herleitung hat, so heisse der Beweisterm t *normal*. Falls $\Gamma \vdash t \downarrow C$ eine gültige Herleitung hat, so heisse t *neutral*.

Warum können wir hier mit Termen im Curry-Stil arbeiten, brauchen also die Annotationen, die Terme im Church-Stil bei sich tragen, nicht mehr? Nun, im Falle von $\lambda x^A.t$, $\mathsf{inl}_B\, t$, $\mathsf{inr}_A\, t$ und $\mathsf{abort}_C\, r$ befinden wir uns im Konstruktionsmodus, wissen also, welche Aussage wir gerade beweisen. Im Falle von $\mathsf{case}\, r\, \mathsf{of}\, \mathsf{inl}\, x^A \Rightarrow s \mid \mathsf{inr}\, y^B \Rightarrow t$ ist r von den Annahmen hergeleitet worden, es muss von der Form $A \vee B$ sein, also können wir A und B daraus ablesen.

Übung 14 (Bidirectional type-checking) Definieren Sie durch wechselseitige Rekursion über r Funktionen $\mathsf{check}(\Gamma, r, A)$ und $\mathsf{infer}(\Gamma, r)$. Dabei soll $\mathsf{check}(\Gamma, r, A)$ *wahr* zurückliefern gdw. $\Gamma \vdash r \uparrow A$, und $\mathsf{infer}(\Gamma, r)$ soll A zurückliefern gdw. $\Gamma \vdash r \downarrow A$.

1.11.2 Beweisreduktionen

Was ist nun von allgemeinen, nicht-normalen Beweisen $\Gamma \vdash t : C$ zu halten? Sie können durch Beweisreduktionen in normale überführt werden. Dies zeigen wir im Folgenden.

β-Reduktionen. Die Einführung eines logischen Konnektives, unmittelbar gefolgt von dessen Beseitigung, stellt einen Umweg im Beweis dar, der schematisch entfernt werden kann.

Konjunktion.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash r : A\end{array} \qquad \begin{array}{c}\vdots\\ \Gamma \vdash s : B\end{array}}{\Gamma \vdash (r,s) : A \wedge B}\,\wedge\mathsf{I}}{\Gamma \vdash \mathsf{fst}\,(r,s) : A}\,\wedge\mathsf{E}_1 \qquad \longrightarrow \qquad \begin{array}{c}\vdots\\ \Gamma \vdash r : A\end{array}$$

Analog für $\wedge\mathsf{I}$ gefolgt von $\wedge\mathsf{E}2$

Implikation.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, x{:}A, \Gamma' \vdash x : A}\,\mathsf{hyp}}{\begin{array}{c}\vdots\\ \Gamma, x{:}A \vdash t : B\end{array}}}{\Gamma \vdash \lambda x t : A \to B}\,\Rightarrow\mathsf{I} \qquad \begin{array}{c}\vdots\\ \Gamma \vdash s : A\end{array}}{\Gamma \vdash (\lambda x t)\, s : B}\,\Rightarrow\mathsf{E} \qquad \longrightarrow \qquad \begin{array}{c}\vdots\\ \Gamma, \Gamma' \vdash s : A\\ \vdots\\ \Gamma \vdash t[s/x] : B\end{array}$$

Disjunktion.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash r : A\end{array}}{\Gamma \vdash \mathsf{inl}\, r : A \vee B}\,\vee\mathsf{I}_1 \qquad \begin{array}{c}\vdots\\ \Gamma, x{:}A, \Gamma' \vdash x : A\\ \vdots\\ \Gamma \vdash x{:}A \vdash s : C\end{array}\,\mathsf{hyp} \qquad \begin{array}{c}\vdots\\ \Gamma, y{:}B, \Gamma' \vdash y : B\\ \vdots\\ \Gamma \vdash y{:}B \vdash t : C\end{array}\,\mathsf{hyp}}{\Gamma \vdash \mathsf{case}\,\mathsf{inl}\, r\ \mathsf{of}\ \mathsf{inl}\, x \Rightarrow s \mid \mathsf{inr}\, y \Rightarrow t : C}\,\vee\mathsf{E}$$

$$\longrightarrow \qquad \begin{array}{c}\vdots\\ \Gamma, \Gamma' \vdash r : A\\ \vdots\\ \Gamma \vdash s[r/x] : C\end{array}$$

Analog für $\vee\mathsf{I}_2$ gefolgt von $\vee\mathsf{E}$.

Tautologie, Absurdität. Für $\top$ und $\bot$ gibt es jeweils nur entweder Einführungs- oder Beseitigungsregel, daher keine β-Reduktion.

Zusammenfassung β-Reduktion.

$$\begin{array}{lll}
(\beta_{\wedge 1}) & \mathsf{fst}\,(r,s) & \longrightarrow\ r\\
(\beta_{\wedge 2}) & \mathsf{snd}\,(r,s) & \longrightarrow\ s\\[1ex]
(\beta_{\Rightarrow}) & (\lambda x t)\, s & \longrightarrow\ t[s/x]\\[1ex]
(\beta_{\vee 1}) & \mathsf{case}\,\mathsf{inl}\, r\ \mathsf{of}\ \mathsf{inl}\, x \Rightarrow s \mid \mathsf{inr}\, y \Rightarrow t & \longrightarrow\ s[r/x]\\
(\beta_{\vee 2}) & \mathsf{case}\,\mathsf{inr}\, r\ \mathsf{of}\ \mathsf{inl}\, x \Rightarrow s \mid \mathsf{inr}\, y \Rightarrow t & \longrightarrow\ t[r/y]
\end{array}$$

Auswertungskontexte.

$$e ::= \bullet\, s \mid \mathsf{fst}\,\bullet \mid \mathsf{snd}\,\bullet \mid \mathsf{case}\,\bullet\,\mathsf{of}\,\mathsf{inl}\,x \Rightarrow s \mid \mathsf{inr}\,y \Rightarrow t \mid \mathsf{abort}_C\,\bullet$$

Permutations-Reduktionen.

$$\begin{array}{lll} (\pi_\vee) & e[\mathsf{case}\,r\,\mathsf{of}\,\mathsf{inl}\,x \Rightarrow s \mid \mathsf{inr}\,y \Rightarrow t] & \longrightarrow \quad \mathsf{case}\,r\,\mathsf{of}\,\mathsf{inl}\,x \Rightarrow e[s] \mid \mathsf{inr}\,y \Rightarrow e[t] \\ (\pi_\bot) & e[\mathsf{abort}\,r] & \longrightarrow \quad \mathsf{abort}\,r \end{array}$$

Bemerkung 15 Für Church-Terme wäre die Regel $e[\mathsf{abort}_C\,r] \longrightarrow \mathsf{abort}_X\,r$ so problematisch, es gibt allerdings einen sehr einfachen Algorithmus, der aus e und dem Typ A des Loches in e den Typ $X = e[A]$ des Kontextes e berechnet. Dieser funktioniert allerdings nur für wohlgetypte Terme.

$$\begin{array}{lcl} e[(A \Rightarrow B)\,s] & = & e[B] \\ e[\mathsf{fst}\,(A \wedge B)] & = & e[A] \\ e[\mathsf{snd}\,(A \wedge B)] & = & e[B] \\ e[\mathsf{abort}_A\,\bot] & = & e[A] \\ e[\mathsf{case}\,A \vee B\,\mathsf{of}\,\mathsf{inl}\,x^A \Rightarrow s \mid \mathsf{inr}\,y^B \Rightarrow t] & = & ??? \end{array}$$

Hier reichen sogar die Annotationen an case nicht!

Kongruenz. An jeder Stelle des Beweisbaumes darf eine Reduktion durchgeführt werden. Zu den fünf Reduktionsaxiomen kommt noch folgende Kongruenz-Regel hinzu:

$$\frac{t \longrightarrow t'}{C[t] \longrightarrow C[t']}$$

Dabei ist $C[\bullet]$ ein *Kontext*, also ein Term mit genau einem Loch $\bullet$, und $C[t]$ bezeichnet die Einsetzung des Terms t für das Loch.

Nun können wir die Begriffe *normal* und *neutral* noch einmal neu definieren, diesmal abstrakter: Ein *normaler* Beweis ist ein Beweis ohne Umwege. Ein *neutraler* Beweis kann in einen normalen Anstelle einer Hypothese eingesetzt werden, ohne dass ein Umweg entsteht.

Definition 16 (Normal und neutral)

1. Ein Beweisterm t heißt normal, falls es keinen Term t' gibt, so dass $t \longrightarrow t'$.
2. Ein Beweisterm s heißt neutral, falls dessen Einsetzung $t[s/x]$ in einen beliebigen normalen Term t wieder normal ist.

Die bisherigen Definitionen von normal und neutral waren korrekt, jedoch vollständig nur für wohlgetypte Terme.

Theorem 17 (Korrektheit)

1. *Wenn* $\Gamma \vdash t \downarrow C$ *oder* $\Gamma \vdash t \uparrow C$*, dann ist* t *normal.*
2. *Wenn* $\Gamma \vdash s \downarrow A$ *und* $\Gamma, x\!:\!A, \Gamma' \vdash t \uparrow C$*, dann ist* $\Gamma, \Gamma' \vdash t[s/x] \uparrow C$*.*

Lemma 18 (Subject reduction) *Wenn* $\Gamma \vdash t : A$ *und* $t \longrightarrow t'$*, dann auch* $\Gamma \vdash t' : A$*.*

Beweis. Durch Induktion über die Herleitung von $\Gamma \vdash t : A$. □

Alternativ kann *subject reduction* auch über die Herleitung von $t \longrightarrow t'$ beweisen werden, dann sollte aber die Kongruenzregel verfeinert werden, dass nur Kontexte der Tiefe 1 zugelassen werden.

$$\begin{array}{rcl} c & ::= & \lambda x.\bullet \mid \bullet\, s \mid r\, \bullet \mid (\bullet, r) \mid (s, \bullet) \mid \mathsf{fst}\, \bullet \mid \mathsf{snd}\, \bullet \\ & \mid & \mathsf{case}\, \bullet\, \mathsf{of}\, \mathsf{inl}\, x \Rightarrow s \mid \mathsf{inr}\, y \Rightarrow r \mid \mathsf{case}\, r\, \mathsf{of}\, \mathsf{inl}\, x \Rightarrow \bullet \mid \mathsf{inr}\, s \Rightarrow t \\ & \mid & \mathsf{case}\, r\, \mathsf{of}\, \mathsf{inl}\, x \Rightarrow s \mid \mathsf{inr}\, y \Rightarrow \bullet \mid \mathsf{abort}\, \bullet \end{array}$$

1.12 Sequenzenkalkül

Teilformeleigenschaft.

Klassischer Sequenzenkalkül LK. Hat zwei $\vee$R Regeln, mit Kontraktion ist das äquivalent zu dem in PVS implementierten Sequenzenkalkül mit nur einer $\vee$R-Regel. Den intuitionistischen Sequenzenkalkül LJ erhält man aus LK durch Beschränkung der rechten Seite auf höchstens eine Formel, wobei die leere rechte Seite mit $\bot$ gleichgesetzt wird. (Strukturregeln rechts fallen alle weg.)

Beweissuche mittels Sequenzenkalkül. Ein anderer Zugang zu schnittfreiem Sequenzenkalkül geht über normale ND-Beweise.

1.12.1 Sequenzenkalkül für Beweissuche

Intuitionistischer Sequenzenkalkül.

$$\frac{}{\Gamma, A \Longrightarrow A}\,\mathsf{hyp}$$

$$\frac{\Gamma, A \Rightarrow B, B \Longrightarrow C \qquad \Gamma, A \Rightarrow B \Longrightarrow A}{\Gamma, A \Rightarrow B \Longrightarrow C}\Rightarrow\mathsf{L} \qquad \frac{\Gamma, A \Longrightarrow B}{\Gamma \Longrightarrow A \Rightarrow B}\Rightarrow\mathsf{R}$$

$$\frac{\Gamma, A, B \Longrightarrow C}{\Gamma, A \wedge B \Longrightarrow C}\wedge\mathsf{L} \qquad \frac{\Gamma \Longrightarrow A \qquad \Gamma \Longrightarrow B}{\Gamma \Longrightarrow A \wedge B}\wedge\mathsf{R}$$

$$\frac{\Gamma, A \Longrightarrow C \qquad \Gamma, B \Longrightarrow C}{\Gamma, A \vee B \Longrightarrow C}\vee\mathsf{L} \qquad \frac{\Gamma \Longrightarrow A}{\Gamma \Longrightarrow A \vee B}\vee\mathsf{R}_1 \qquad \frac{\Gamma \Longrightarrow B}{\Gamma \Longrightarrow A \vee B}\vee\mathsf{R}_2$$

$$\frac{}{\Gamma, \bot \Longrightarrow C}\bot\mathsf{L} \qquad \frac{}{\Gamma \Longrightarrow \top}\top\mathsf{R}$$

Sequenzenkalkül als Produzent normaler Terme.

$$\frac{}{\Gamma,\ r \downarrow A \Longrightarrow r \uparrow A}\ \mathsf{hyp}$$

$$\frac{\Gamma,\ r \downarrow A \Rightarrow B,\ r\,s \downarrow B \Longrightarrow t \uparrow C \qquad \Gamma,\ r \downarrow A \Rightarrow B \Longrightarrow s \uparrow A}{\Gamma,\ r \downarrow A \Rightarrow B \Longrightarrow t \uparrow C}\ \Rightarrow\mathsf{L}$$

$$\frac{\Gamma,\ x \downarrow A \Longrightarrow t \uparrow B}{\Gamma \Longrightarrow \lambda x t \uparrow A \Rightarrow B}\ \Rightarrow\mathsf{R}$$

$$\frac{\Gamma,\ \mathsf{fst}\,r \downarrow A,\ \mathsf{snd}\,r \downarrow B \Longrightarrow t \uparrow C}{\Gamma,\ r \downarrow A \wedge B \Longrightarrow t \uparrow C}\ \wedge\mathsf{L} \qquad \frac{\Gamma \Longrightarrow t_1 \uparrow A \qquad \Gamma \Longrightarrow t_2 \uparrow B}{\Gamma \Longrightarrow (t_1, t_2) \uparrow A \wedge B}\ \wedge\mathsf{R}$$

$$\frac{\Gamma,\ x \downarrow A \Longrightarrow s \uparrow C \qquad \Gamma,\ x \downarrow B \Longrightarrow t \uparrow C}{\Gamma,\ r \downarrow A \vee B \Longrightarrow \mathsf{case}\,r\ \mathsf{of}\ \mathsf{inl}\,x \Rightarrow s\ \mid\ \mathsf{inr}\,y \Rightarrow t \uparrow C}\ \vee\mathsf{L}$$

$$\frac{\Gamma \Longrightarrow t \uparrow A}{\Gamma \Longrightarrow \mathsf{inl}\,t \uparrow A \vee B}\ \vee\mathsf{R}_1 \qquad \frac{\Gamma \Longrightarrow t \uparrow B}{\Gamma \Longrightarrow \mathsf{inr}\,t \uparrow A \vee B}\ \vee\mathsf{R}_2$$

$$\frac{}{\Gamma,\ r \downarrow \bot \Longrightarrow \mathsf{abort}\,r \uparrow C}\ \bot\mathsf{L} \qquad \frac{}{\Gamma \Longrightarrow () \uparrow \top}\ \top\mathsf{R}$$

1.13 Kripke-Modelle

Was soll als wahr gelten in der konstruktiven Logik? In der klassischen Logik postuliert man einen logisch allwissenden Gott, der zu jeder Aussage die Antwort "wahr" oder "falsch" kennt. In der konstruktiven Logik setzt man dagegen einen Erkenntnisbegriff ein: Das Wissen um die Wahrheit/Falschheit von Aussagen nimmt stetig zu, und idealerweise monoton, d.h. (im Gegensatz zur Geschichte) geht Wissen nie mehr verloren. Ist also eine Aussage zu einem Zeitpunkt als wahr erkannt, so auch in allen zukünftigen Zeitpunkten. Die Aussage $\neg A$ ist wahr, wenn A nie wahr werden kann, d.h. in allen Zukunften die Annahme von A zu Widersprüchen führt. Eine Implikation $A \Rightarrow C$ ist wahr, wenn die Annahme von A in allen Zukunften C erzwingt.

Die Aussage "wenn die Sonne scheint, regnet es nicht" wird also nicht nur dadurch wahr, dass die Sonne gerade nicht scheint oder es gerade nicht regnet (klass. Implikation), sondern es darf eben nie mehr regnen, wenn die Sonne scheint.

Diese Überlegungen führen zu folgender Interpretation von konstruktiver Aussagenlogik.

Ein Aussagenlogisches Kripke-Modell ist ein Tripel $(K, \leq, \Vdash)$ so dass

- K ist eine nicht-leere Menge von Welten,
- $\leq$ ist partielle Präordnung auf K (reflexiv, transitiv),
- $\Vdash$ ist eine Relation zwischen Welten und Aussagenvariablen, $k \Vdash P$ bedeuted "k erfüllt P",
- die Erfüllungsrelation $\Vdash$ ist monoton, d.h.

$$k \Vdash P \quad \text{und} \quad k' \leq k \quad \text{implizieren} \quad k' \Vdash P.$$

Die Erfüllungsrelation wird wie folgt auf Aussagen erweitert:

- $k \Vdash \top$
- $k \Vdash A \wedge B$ gdw. $k \Vdash A$ und $k \Vdash B$
- $k \Vdash A \vee B$ gdw. $k \Vdash A$ oder $k \Vdash B$
- $k \Vdash A \Rightarrow B$ gdw. für alle $k' \leq k$, wenn $k' \Vdash A$, dann $k' \Vdash B$
- $k \nVdash \bot$

Übung 19 (Monotonie für beliebige Formeln) Zeigen Sie: Wenn $k' \leq k \Vdash A$, dann $k' \Vdash A$.

Das natürliche Schließen ist vollständig für Kripke-Modelle, d.h., gilt eine Aussage A in allen Modellen, so ist sie beweisbar. Unbeweisbarkeit kann man also durch Angabe eines Gegenmodells zeigen.

Beispiel 20 Gegenmodell zu $P \vee \neg P$: Nehme die Welten $K = \{\emptyset, \{P\}\}$ mit $k \Vdash P$ gdw. $P \in k$ und $k' \leq k$ gdw. $k' \supseteq k$. Dann $\emptyset \nVdash P \vee \neg P$, da $\emptyset \nVdash P$ und $\emptyset \nVdash P \Rightarrow \bot$. Letzteres gilt, weil es eine Zukunft $\{P\}$ von $\emptyset$ gibt so dass $\{P\} \Vdash P$ aber $\{P\} \nVdash \bot$.

Dasselbe Gegenmodell widerlegt die Pierce-Formel $((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$.

$k \Vdash \Gamma$ gelte gdw. $k \Vdash A$ für alle $A \in \Gamma$.

Theorem 21 (Korrektheit des natürlichen Schließens) *Wenn* $\Gamma \vdash A$ *und* $k \Vdash \Gamma$*, dann* $k \Vdash A$*.*

Beweis. Durch Induktion über $\Gamma \vdash A$.

Fall

$$\frac{A \in \Gamma}{\Gamma \vdash A}$$

$k \Vdash A$ nach Def. von $k \Vdash \Gamma$.

Fall

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$$

Um $k \Vdash A \Rightarrow B$ zu zeigen, wähle beliebiges $k' \leq k$ mit $k' \Vdash A$ und zeige $k' \Vdash B$. Mit Monotonie gilt $k' \Vdash \Gamma$ also $k' \Vdash \Gamma, A$. Nach I.V. folgt sofort $k' \Vdash B$.

Fall

$$\frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

Nach I.V. gilt $k \Vdash A$ und $k \Vdash A \Rightarrow B$, und da $k \leq k$ also auch $k \Vdash B$.

Fall

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}$$

Nach I.V. gilt $k \Vdash A \vee B$. Falls $k \Vdash A$, dann $k \Vdash \Gamma, A$ und nach I.V. $k \Vdash C$. Falls $k \Vdash B$, analog.

Fall

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash C}$$

Nach I.V. gilt $k \Vdash \bot$, ein Widerspruch. *Ex falsum sequitur quodlibet*, also $k \Vdash C$.

□

1.14 Vollständigkeit des natürlichen Schließens

Wir zeigen die Vollständigkeit durch Konstruktion eines universellen Modells. Wir beschränken uns auf das negative Fragment $P, \top, \wedge, \Rightarrow$, für die Behandlung von Disjunktionen braucht man entweder eine klassische Metalogik, oder Beth-Modelle, eine Erweiterung von Kripke-Modellen.

Als Welten nehmen wir Kontexte (Mengen von Formeln) und definieren unser Modell durch:

- $\Gamma \Vdash P$ gdw. $\Gamma \vdash P$.
- $\Gamma' \leq \Gamma$ gdw. $\Gamma' \supseteq \Gamma$.

Lemma 22 $\Gamma \vdash A$ *gdw.* $\Gamma \Vdash A$.

Beweis. Durch Induktion über A.

Fall $A = P$. Siehe Def. von $\Gamma \Vdash P$.

Fall $A = B \Rightarrow C$. "Dann:" Wir zeigen $\Gamma \Vdash B \Rightarrow C$, dazu sei $\Gamma' \leq \Gamma$ beliebig mit $\Gamma' \Vdash B$. Nach I.V. gilt $\Gamma \vdash B$, also mit $\Rightarrow\mathsf{E}$, $\Gamma \vdash C$. Wiederum nach I.V. gilt $\Gamma \Vdash C$. "Wenn:" Wir zeigen $\Gamma \vdash B \Rightarrow C$ durch $\Rightarrow\mathsf{I}$. Nach Vorraussetzung $\Gamma \Vdash B \Rightarrow C$ gilt $\Gamma, B \Vdash C$, da $\Gamma, B \leq \Gamma$ und $\Gamma, B \Vdash B$ (dies gilt nach I.V., da $\Gamma, B \vdash B$). Also gilt nach I.V. $\Gamma, B \vdash C$.

Fall $A = B \wedge C$. Einfach nach I.V., da $\Gamma \vdash B \wedge C$ gdw. $\Gamma \vdash B$ und $\Gamma \vdash C$, und ebenso für $\Gamma \Vdash B \wedge C$.

□

Korollar 23 $\Gamma \Vdash \Gamma$.

Theorem 24 (Vollständigkeit des natürlichen Schließens) *Es gelte für alle Modelle und alle Welten k: $k \Vdash \Gamma$ impliziert $k \Vdash A$. Dann $\Gamma \vdash A$.*

Beweis. Speziell gilt für das universelle Modell und $k = \Gamma$, dass $\Gamma \Vdash A$. Also $\Gamma \vdash A$ nach Lemma 22. □

Erweiterung auf volle Aussagenlogik. Der Beweis von Lemma 22 scheitert für Disjunktionen $A = B \vee C$. Wir haben $\Gamma \vdash B \vee C$ und müssen zeigen, dass $\Gamma \Vdash B$ oder $\Gamma \Vdash C$. Es kann aber sein, dass weder $\Gamma \vdash B$ noch $\Gamma \vdash C$ (z.B. $\Gamma = B \vee C$), also ist die I.V. nicht anwendbar. Die Idee ist, nun solche Kontexte Γ zuzulassen (wir nennen sie saturiert), die, falls $\Gamma \vdash B \vee C$, entweder $\Gamma \vdash B$ oder $\Gamma \vdash C$ erzwingen. Ähnliche Probleme ergeben sich für $A = \bot$. Hier kommen wir von $\Gamma \vdash \bot$ nicht weiter. Idee hier ist, nur konsistente Kontexte zuzulassen mit $\Gamma \nvdash \bot$.

Definition 25 Ein Kontext Γ heißt saturiert für D, falls

1. $\Gamma \nvdash \bot$ (Γ ist konsistent), und
2. für alle Teilformeln $A \vee B$ von Γ, D mit $\Gamma \vdash A \vee B$ gilt $\Gamma \vdash A$ oder $\Gamma \vdash B$.

Lemma 26 (Saturierung) *(Sei C Teilformel von D.) Falls $\Gamma \nvdash C$ so gibt es einen für D saturierten Kontext $\hat{\Gamma} \leq \Gamma$ mit $\hat{\Gamma} \nvdash C$.*

Beweis. Da $\Gamma \nvdash C$, ist Γ konsistent. Sei $(A_i \vee B_i)_{0 \leq i < n}$ eine Aufzählung aller Subformeln von Γ, D mit $\Gamma \vdash A_i \vee B_i$. Wir konstruieren eine Folge von konsistenten Kontexten Γ_i durch $\Gamma_0 = \Gamma$ und

$$\Gamma_{i+1} = \begin{cases} \Gamma_i & \text{falls } \Gamma_i \vdash A_i \text{ oder } \Gamma_i \vdash B_i \\ \Gamma_i, A_i & \text{falls } \Gamma_i, A_i \nvdash C \\ \Gamma_i, B_i & \text{falls } \Gamma_i, B_i \nvdash C \end{cases}$$

Die Fallunterscheidung ist vollständig, da aus $\Gamma_i, A_i \vdash C$ und $\Gamma_i, B_i \vdash C$ mit $\vee\mathsf{E}$ $\Gamma \vdash C$ folgt, im Widerspruch zur Annahme.

Nun ist $\hat{\Gamma} = \Gamma_n$ der gesuchte Kontext. □

Wir wollen nun die Vollständigkeit des natürlichen Schließens zeigen, also $\Gamma \vdash D$ falls für alle Modelle und Welten k gilt: $k \Vdash \Gamma$ implies $k \Vdash D$. Dazu betrachten wir ein universelles Modell, in dem eine Welt ein für D saturierter Kontext Γ ist. Erfüllung und Zukunft seien wie zuvor definiert.

Lemma 27 *Sei Γ saturiert für D und C eine Teilformel von Γ, D. Dann $\Gamma \vdash C$ gdw. $\Gamma \Vdash C$.*

Beweis. Induktion über C.

Fall $C = A \Rightarrow B$. "Dann:" Wir zeigen $\Gamma \Vdash B \Rightarrow C$, dazu sei $\Gamma' \leq \Gamma$ saturiert für D beliebig mit $\Gamma' \Vdash B$. Nach I.V. gilt $\Gamma \vdash B$, also mit $\Rightarrow\mathsf{E}$, $\Gamma \vdash C$. Wiederum nach I.V. gilt $\Gamma \Vdash C$. "Wenn:" Beweis durch Widerspruch. Sei $\Gamma \nvdash A \Rightarrow B$. Dann auch $\Gamma, A \nvdash B$. Nach Lemma 26 gibt es einen Kontext $\hat{\Gamma} \leq \Gamma, A$ saturiert für D mit $\hat{\Gamma} \nvdash B$. Nach I.V. also $\hat{\Gamma} \nVdash B$. Wegen $\hat{\Gamma} \leq \Gamma, A \vdash A$ gilt aber $\hat{\Gamma} \Vdash A$ und damit $\hat{\Gamma} \Vdash B$. Widerspruch!

Fall $C = A \vee B$. "Dann:" Es gilt $\Gamma \vdash A \vee B$. Da $A \vee B$ Teilformel von Γ, D und Γ saturiert für D, gilt entweder $\Gamma \vdash A$ oder $\Gamma \vdash B$, nach I.V. also $\Gamma \Vdash A$ oder $\Gamma \Vdash B$. "Wenn:" Im Falle $\Gamma \Vdash A$ gilt nach I.V. $\Gamma \vdash A$, also $\Gamma \vdash A \vee B$. Rest analog.

Fall $C = \bot$. $\Gamma \Vdash \bot$ und $\Gamma \vdash \bot$ sind beide falsch, da Γ konsistent.

□

Theorem 28 (Vollständigkeit des natürlichen Schließens) *Es gelte für alle Modelle und alle Welten k: $k \Vdash \Gamma$ impliziert $k \Vdash D$. Dann $\Gamma \vdash D$.*

Beweis. Beweis durch Widerspruch. Nehmen wir an, dass $\Gamma \not\vdash D$. Dann gibt es nach Lemma 26 ein $\hat{\Gamma} \leq \Gamma$ saturiert für D mit $\hat{\Gamma} \not\vdash D$. Nach Lemma 27 also $\hat{\Gamma} \not\Vdash D$. Allerdings gilt $\hat{\Gamma} \vdash \Gamma$ und damit wiederum nach Lemma 27 $\hat{\Gamma} \Vdash \Gamma$. Für das universelle Modell der für D saturierten Kontexte gilt also nach Annahme $\hat{\Gamma} \Vdash D$, ein Widerspruch! □

1.15 Curry-Howard-Isomorphismus und einfach getypter Lambda-Kalkül

Nach der Brower-Heyting-Kolmogorov-Interpretation sind Beweise Objekte (z.B. Paare) oder Rechenverfahren (z.B. Beweis der Implikation). 1958 entdeckte Haskell Curry die Übereinstimmung von Beweisen im Hilbert-Kalkül mit Termen partiell kombinatorischer Algebren (ein Berechnungsmodell). 1969 stellte William Howard fest, dass Beweise im Kalkül des natürlichen Schliessens mit λ-Termen übereinstimmen (ein weiteres Berechnungsmodell). Der Typ eines Terms entspricht der Aussage, die dieser Term beweist.

Aussage	Typ
Konjunktion $A \wedge B$	kartesisches Produkt $\sigma \times \tau$
Disjunktion $A \vee B$	disjunkte Summe $\sigma + \tau$
Implikation $A \Rightarrow B$	Funktionsraum $\sigma \to \tau$
Trivialität $\top$	einelementiger Typ 1
Absurdität $\bot$	leerer Typ 0

Die Regeln für natürliches Schließen mit Beweistermen kann man also eins-zu-eins in Typisierungsregeln für λ-Terme übertragen. Im folgenden trennen wir strikt zwischen *freien Variablen* $a, b, c \ldots$, auch *Parameter* oder *neue Konstantensymbole* genannt, und *gebundenen Variablen* $x, y, z \ldots$. Ein Typisierungskontext Γ bindet Parameter an Typen.

Typisierungsregeln für Parameter und Funktionen:

$$\frac{(a{:}\tau) \in \Gamma}{\Gamma \vdash a : \tau} \qquad \frac{\Gamma, a{:}\sigma \vdash t[a/x] : \tau}{\Gamma \vdash \lambda x t : \sigma \to \tau}\ a \text{ neu} \qquad \frac{\Gamma \vdash r : \sigma \to \tau \qquad \Gamma \vdash s : \sigma}{\Gamma \vdash r\, s : \tau}$$

In der Typisierungsregel für $\lambda x t$ muss a ein neuer Parameter sein, der noch nicht in Γ deklariert ist.

Der Wert (d.h., das Ergebnis) eines Programmes wird berechnet, in dem man alle *Reduktionen* durchführt, die Möglich sind. Reduktion von Termen haben wir schon unter dem Begriff *Beweisreduktion* kennengelernt. Bsp.:

$$(\lambda x t)\, s \quad \longrightarrow \quad t[s/x]$$

Übung 29 (Programmieren im λ-Kalkül) Schreiben Sie Programme, deren Typen wie folgt vorgegeben sind.

1. $\mathsf{id} : \tau \to \tau$.
2. $\mathsf{curry} : (\sigma \times \sigma' \to \tau) \to \sigma \to \sigma' \to \tau$.
3. $\mathsf{uncurry} : (\sigma \to \sigma' \to \tau) \to \sigma \times \sigma' \to \tau$.
4. $\mathsf{comp} : (\tau_2 \to \tau_3) \to (\tau_1 \to \tau_2) \to \tau_1 \to \tau_3$.

Zeigen Sie, dass $\mathsf{compcurryuncurry} \longrightarrow^+ \lambda f \lambda x \lambda y.\, f\, x\, y$.

Kapitel 2

Prädikatenlogik und Typentheorie

2.1 Konstruktive Prädikatenlogik

Als Sprache verwenden wir den einfach getypten Lambda-Kalkül mit dem Typisierungsurteil $\Gamma \vdash t : \tau$.

Prädikatenlogik erweitert die Aussagenlogik um zwei neue logische Konnektive, $\forall$ und $\exists$.

BHK-Interpretation:

- Ein Beweis von $\forall x{:}\tau.\, A$ ist ein Verfahren, dass für ein beliebiges Element $a : \tau$ einen Beweis von $A[a/x]$ liefert.
- Ein Beweis von $\exists x{:}\tau.\, A$ ist ein Paar (w, p), bestehend aus einem Zeugen $w : \tau$ und einem Beweis p von $A[w/x]$.

Beispiel 30 $((\exists x{:}\tau.P) \Rightarrow Q) \Rightarrow \forall x{:}\tau.P \Rightarrow Q$ entspricht dem *currying*.

Die Aussage $(\forall x : \tau.\, P \vee Q(x)) \Rightarrow P \vee \forall x : \tau.\, Q(x)$ (wobei x nicht frei in P vorkomme) ist konstruktiv nicht gültig. Um zu entscheiden, ob P gilt oder $\forall x : \tau.\, Q(x)$, müßte man erst für alle x in τ testen, ob P oder $Q(x)$ gilt. Im allgemeinen kann man aber das nicht in endlicher Zeit tun.

Beispiel 31 (Induktion) Das Induktionsaxiom für das Prädikat P kann formuliert werden als $P\,\mathsf{zero} \Rightarrow (\forall n{:}\mathsf{nat}.P\,n \Rightarrow P\,(\mathsf{succ}\,n)) \Rightarrow \forall n{:}\mathsf{nat}.\, P\,n$.

Regeln des natürlichen Schliessens:

$$\frac{\Gamma, a{:}\tau \vdash A[a/x]}{\Gamma \vdash \forall x{:}\tau.\, A}\,\forall\mathsf{I}\ a \text{ neu} \qquad \frac{\Gamma \vdash \forall x{:}\tau.\, A \qquad \Gamma \vdash t : \tau}{\Gamma \vdash A[t/x]}\,\forall\mathsf{E}$$

$$\frac{\Gamma \vdash A[t/x] \qquad \Gamma \vdash t : \tau}{\Gamma \vdash \exists x{:}\tau.\, A}\,\exists\mathsf{I} \qquad \frac{\Gamma \vdash \exists x{:}\tau.\, A \qquad \Gamma, a{:}\tau, A[a/x] \vdash C}{\Gamma \vdash C}\,\exists\mathsf{E}$$

Beweisterme und Typisierung: Für $\forall\mathsf{I}$ verwenden wir die Abstraktion (wie für $\Rightarrow\mathsf{I}$), und für $\forall\mathsf{E}$ die Applikation (wie für $\Rightarrow\mathsf{E}$).

$$\frac{\Gamma, a{:}\tau \vdash p[a/y] : A[a/x]}{\Gamma \vdash \lambda y.p : \forall x{:}\tau.\, A}\,\forall\mathsf{I}\ a \text{ neu} \qquad \frac{\Gamma \vdash p : \forall x{:}\tau.\, A \qquad \Gamma \vdash t : \tau}{\Gamma \vdash p\,t : A[t/x]}\,\forall\mathsf{E}$$

In Coq wird $\lambda y.p$ als `fun y => p` geschrieben.

Ein Beweis einer existentiellen Aussage $\exists x{:}\tau.A$ ist ein Paar (t, p) bestehend aus einem Zeugen $t : \tau$ und einem Beweis p von $A[t/x]$.

$$\frac{\Gamma \vdash p : A[t/x] \qquad \Gamma \vdash t : \tau}{\Gamma \vdash (t, p) : \exists x{:}\tau.\, A}\,\exists\mathsf{I} \qquad \frac{\Gamma \vdash p : \exists x{:}\tau.\, A \qquad \Gamma, a{:}\tau, h : A[a/x] \vdash q : C}{\Gamma \vdash \mathsf{let}\ (a, h){=}p\ \mathsf{in}\ q : C}\,\exists\mathsf{E}$$

In Coq wird das Paar mit `ex_intro _ t p` gebildet und mit

```
match p with ex_intro a h => q
```

eliminiert.

2.2 Kripke-Modelle für Prädikatenlogik

Ein Kripke-Modell für die Prädikatenlogik ist ein Quadrupel $(K, \leq, \Vdash, \{[\![-]\!]^k\}_{k\in K})$, wobei $(K, \leq, \Vdash)$ ein aussagenlogisches Kripke-Modell ist. Für jede Welt $k \in K$ ist $[\![-]\!]^k$ eine Interpretation der Typen und Funktions- und Prädikatensymbole. Die Interpretationen erfüllen folgende Monotoniebedingungen:

Falls $k' \leq k$, so gilt:

- Für alle Typen τ: $[\![\tau]\!]^k \subseteq [\![\tau]\!]^{k'}$ (neue Elemente dürfen hinzukommen, alte Elemente dürfen nicht verschwinden).
- Für alle Funktionssymbole $f : \tau_1 \times \cdots \times \tau_n \to \tau$, $[\![f]\!]^k(x_1, \ldots, x_n) = [\![f]\!]^{k'}(x_1, \ldots, x_n)$ für alle $x_i \in [\![\tau_i]\!]^k$. (Interpretation von Funktionen kann sich nicht ändern, nur auf neue Elemente erweitern).
- Für alle Prädikatensymbole $P : \tau_1 \times \cdots \times \tau_n \to \mathsf{Prop}$, $[\![P]\!]^k(x_1, \ldots, x_n)$ impliziert $[\![P]\!]^{k'}(x_1, \ldots, x_n)$ für alle $x_i \in [\![\tau_i]\!]^k$ (Prädikate dürfen nur "wahrer" werden).

Eine Umgebung η ist eine partielle Abbildung von Variablen und Parametern auf Elemente der Typinterpretationen einer Welt.

Die Interpretation $[\![t]\!]^k_\eta$ eines Terms t in der Umgebung ist definiert durch:

$$\begin{array}{lcl} [\![x]\!]^k_\eta & = & \eta(x) \\ [\![f(t_1, \ldots, t_n)]\!]^k_\eta & = & [\![f]\!]^k([\![t_1]\!]^k_\eta, \ldots [\![t_n]\!]^k_\eta) \end{array}$$

Lemma 32 (Korrektheit der Term-Interpretation) *Wenn $\Gamma \vdash t : \tau$ und $\eta(a) \in [\![\Gamma(a)]\!]^k$ für alle $a \in \mathsf{FV}(t)$, dann $[\![t]\!]^k_\eta \in [\![\tau]\!]^k$.*

Die Erfüllbarkeit $k \Vdash A\ \eta$ ist eine Relation zwischen einer Welt k, einer Formel A und einer Umgebung η. Folgende Bedingungen müssen erfüllt sein:

- $k \Vdash P(\vec{t})\ \eta$ gdw. $[\![P]\!]^k([\![\vec{t}]\!]^k_\eta)$.
- $k \Vdash (\forall x{:}\tau.A)\ \eta$ gdw. $k' \Vdash A\ \eta[x \mapsto o]$ für alle $k' \leq k$ und alle $o \in [\![\tau]\!]^{k'}$.
- $k \Vdash (\exists x{:}\tau.A)\ \eta$ gdw. es ein $o \in [\![\tau]\!]^k$ gibt, so dass $k \Vdash A\ \eta[x \mapsto o]$.
- $k \Vdash (A \Rightarrow B)\ \eta$ gdw. $k' \Vdash A\ \eta$ impliziert $k' \Vdash B\ \eta$ für alle $k' \leq k$.
- Klauseln für die übrigen aussagenlogischen Konnektive, analog zu Kripke-Modellen für Aussagenlogik.

Beispiel 33 (Markov-Prinzip) Wir konstruieren ein Gegenmodell zu der klassisch gültigen Formel $\neg(\forall x : \tau.\, \neg P(x)) \Rightarrow \exists x : \tau.\, P(x)$ [Mos07]. Das Modell besteht aus der Gegenwart und einer Zukunft. Der Typ τ besteht jeweils aus genau einem Element o. $P(o)$ gelte in der Zukunft, aber nicht in der Gegenwart. Damit ist die Folgerung $\exists x : \tau.\, P(x)$ falsch. Wir müssen zeigen, dass die Prämisse $\neg(\forall x : \tau.\, \neg P(x))$ wahr ist. Dazu muss in allen Welten $\forall x : \tau.\, \neg P(x)$ falsch sein. Es genügt, dass $\neg P(x)$ in der Zukunft falsch ist, was auch der Fall ist.

Beispiel 34 (Markov-Prinzip Variante) Ein Gegenmodell zu $(\neg\forall x{:}\tau.\, P(x)) \Rightarrow \exists x : \tau.\, \neg P(x)$: Die Interpretation der Bedingung $(\neg\forall x : \tau.\, P(x))$ ist: Jede Welt hat eine Zukunft, in der es ein x gibt, so dass nicht $P(x)$. Salopp: immer wieder gibt es ein x, dass P nicht erfüllt. Die Folgerung $\exists x{:}\tau.\, \neg P(x)$ liesst sich als: In der Gegenwart gibt es ein x, dass in aller Zukunft nie P erfüllt.

Ein Gegenmodell sieht zum Beispiel so aus: τ wird in allen Welten als $\mathbb{N}$ interpretiert, und $P^k(x)$ gdw. $k > x$. Damit gibt es in jeder Welt k ein Element $x = k$, dass P nicht erfüllt, aber jedes Element erfüllt irgendwann einmal P.

Wachsende Typinterpretationen braucht man, um folgende Formel zu widerlegen:

Beispiel 35 Gegenmodell zu $(\forall x : \tau.\, P \vee Q(x)) \Rightarrow P \vee (\forall x : \tau.\, Q(x))$: Welten $K = 0, 1$. Gegenwart: $[\![\tau]\!]^0 = \{0\}, [\![P]\!]^0 = \emptyset, [\![Q]\!]^0 = \{0\}$. Zukunft: $[\![\tau]\!]^1 = \{0, 1\}, [\![P]\!]^1 = \{()\}, [\![Q]\!]^1 = \{0\}$. Dann ist in jeder Welt P wahr oder $Q(x)$ für alle $x : \tau$, aber weder ist P wahr in der Gegenwart noch in allen Welten $Q(x)$ für alle $x : \tau$.

Theorem 36 In jedem Kripke-Modell, in dem die Typinterpretationen in der Zeit fest bleiben, also $[\![\tau]\!]^k = [\![\tau]\!]^{k'}$ für alle k, k', gilt $(\forall x{:}\tau.\, P \vee Q(x)) \Rightarrow P \vee (\forall x{:}\tau.\, Q(x))$.

Beweis. Entweder gilt in allen Welten $Q(x)$ für alle $x : \tau$. Oder es gibt eine Welt k und ein $x : \tau$, so dass $Q(x)$ in k nicht gilt. Dann gilt aber $Q(x)$ auch in der Gegenwart nicht. Also muss in der Gegenwart P gelten.

Dieser Beweis funktioniert nicht für wachsende Interpretationen, da das $x : \tau$ der Welt k in der Gegenwart nicht notwendigerweise existiert. □

Kapitel 3

Induktion

3.1 Induktive Datentypen

Ein *freier Datentyp* ist gegeben durch eine Menge von *Konstruktoren*, mit denen Elemente diese Typs erzeugt werden können.

Beispiel 37 (Binärbäume) Binärbäume, die an den Knoten mit natürlichen Zahlen beschriftet sind, werden von den folgenden Konstruktoren erzeugt.

$$\begin{array}{lcl} \mathsf{leaf} & : & \mathsf{nattree} \\ \mathsf{node} & : & \mathsf{nattree} \to \mathsf{nat} \to \mathsf{nattree} \to \mathsf{nattree} \end{array}$$

Bsp: $\mathsf{node}(\mathsf{node}(\mathsf{leaf}, 1, \mathsf{leaf}), 0, \mathsf{leaf})$.

Moderne funktionale Sprachen, wie Haskell und ML, erlauben die Konstruktion von freien Datentypen mit Datentypdeklarationen, z.B. Haskell:

```
data NatTree where
  Leaf :: NatTree
  Node :: NatTree -> Nat -> NatTree -> NatTree
```

Von *induktiven (Daten)typen* spricht man, wenn man nur die *endlich erzeugten* bzw. *fundierten* Elemente des freien Typs betrachtet. Folgender Baum wird also ausgeschlossen:

Beispiel 38 (Unendlicher Baum) Sei $f(x) = \mathsf{node}(\mathsf{leaf}, 0, x)$. Dann ist $f(f(f(f(\ldots))))$ ein unendlicher linearer Baum, der die Gleichung $x = f(x)$ löst.

In Coq kann man eigene induktive Datentypen mit dem `Inductive`-Kommando definieren.

```
Inductive nattree : Set :=
| leaf : nattree
| node : nattree -> nat -> nattree -> nattree.
```

Nichtrekursive und rekursive Funktionen lassen sich wie folgt definieren (Vergleichsoperation `le_gt_dec` auf natürlichen Zahlen wird durch `Require Import Compare_dec` verfügbar):

Beispiel 39 (Einfügen in Suchbaum)

```
Definition singleton (n : nat) : nattree := node leaf n leaf.

(* Insert a number n into the binary search tree t. *)
Fixpoint insert (n : nat) (t : nattree) : nattree :=
match t with
| leaf => singleton n
| node l m r => match le_gt_dec n m with
    | left _ => node (insert n l) m r     (* n <= m *)
    | right _ => node l m (insert n r)    (* n >  m *)
    end
end.
```

3.1.1 Die natürlichen Zahlen als induktiver Typ

Eine natürliche Zahl n kann ich durch n-malige Anwendung der Nachfolgerfunktion auf die Null erzeugen. Damit kann man die natürlichen Zahlen induktiv generieren:

```
Inductive nat : Set :=
| O : nat
| S : nat -> nat.
```

(So ist `nat` in der Coq-Standardbibliothek definiert.) Dabei ist die Nachfolgerfunktion `S` injektiv und die Konstruktoren erzeugen stets verschiedene Elemente. Beide Tatsachen kann man in Coq beweisen, mit der `inversion`-Taktik.

```
Theorem peano7 : forall n, O <> S n.
intros n H.
inversion H.
Qed.

Theorem peano8 : forall n m, S n = S m -> n = m.
intros n m H.
inversion H.
reflexivity.
Qed.
```

Eine konstruktiv schwächere Formulierung der Injektivität lässt sich daraus ableiten: Verschiedene Zahlen haben auch verschiedene Nachfolger.

```
Theorem notEqS : forall n m, n <> m -> S n <> S m.
intros n m N E.
```

```
apply N.
apply peano8.
assumption.
Qed.
```

Die zu `peano8` umgekehrte Richtung ist trivialerweise wahr, da man stets Gleiches mit Gleichem ersetzen kann (Taktik `rewrite`).

```
Theorem eqS : forall n m, n = m -> S n = S m.
intros n m H.
rewrite H.
reflexivity.
Qed.
```

Coq erzeugt zu jeder induktiven Definition automatisch ein Induktionsprinzip. Im Fall von `nat` können wir es mit `Print nat_ind` einsehen.

```
nat_ind :  forall P : nat -> Prop,
           P 0 ->
           (forall n : nat, P n -> P (S n)) ->
           forall n : nat, P n
```

Damit können wir nun beweisen, dass eine Zahl von ihrem Nachfolger verschieden ist. Das Induktionsprinzip kann mit der `induction`-Taktik komfortabel benutzt werden.

```
Theorem nNotSn : forall n, n <> S n.
induction n.
 apply peano7.
apply notEqS.
assumption.
Qed.
```

Der erzeugte Beweisterm ist im Wesentlichen ein Aufruf von `nat_ind`. Wir können also auch einen Beweisterm direkt angeben:

```
Definition nNotSn' : forall n, n <> S n :=
  nat_ind (fun n => n <> S n)
          (peano7 0)
          (fun n (IHn : n <> S n) => notEqS n (S n) IHn).
```

3.1.2 Primitive Rekursion

Primitive rekursiv definierte Funktionen werden mit dem `Fixpoint`-Kommando definiert.

Beispiel 40 (Addition)

```
Fixpoint add (n m : nat) : nat :=
match m with
| O => n
| S m' => S (add n m')
end.
```

Addition wird hier durch Rekursion über das zweite Argument berechnet. Die Tatsache `add n 0 = n` ist daher trivial, da die rechte Seite der Gleichung zur linken vereinfacht werden kann.

```
Lemma add_n_0 : forall n, add n 0 = n.
reflexivity.
Qed.
```

Der Ausdruck `add 0 m` kann jedoch nicht vereinfacht werden, da `m` unbekannt ist. Ein entsprechendes Theorem bedarf induktiven Beweises.

```
Lemma add_0_n : forall n, add 0 n = n.
induction n.
 reflexivity.
simpl.
apply eqS.
assumption.
Qed.
```

3.1.3 Parametrisierte induktive Typen

Binärbäume mit Markierungen aus einer Menge `A`.

```
Section BinTree.

Hypothesis (A : Set).

Inductive bintree : Set :=
| leaf : bintree
| node : bintree -> A -> bintree -> bintree.

End BinTree.

Print bintree.
```

Dasselbe erreicht man ohne `Section` durch Angabe des Parameters `A` in der `Inductive`-Deklaration.

```
Inductive bintree (A : Set) : Set :=
| node : bintree A
| leaf : bintree A -> A -> bintree A -> bintree A.
```

Beim Konstruieren eines Elements des induktiven Types benötigen die Konstruktoren als erstes Argument die Instanz des Parameters `A`.

```
Definition atree := node _ (node _ (leaf nat) 5 (leaf _)) 3 (leaf _).
```

Meist kann Coq sie jedoch inferieren, und es reicht ein `_`.

Beim Zerlegen einer Datenstruktur mit `match` ist die Angabe des Parameters jedoch weder nötig noch erlaubt:

```
Require Import Max.

Fixpoint height (A : Set) (t : bintree A) : nat :=
match t with
| leaf => 0
| node l _ r => S (max (height _ l) (height _ r))
end.
```

Binärbäume und Listen sind *homogene* Datentypen. Bei *heterogenen* Datentypen darf sich der Typparameter im Typ der Unterbäume ändern. Z.B. perfekte Bäume:

```
Inductive ptree (A : Set) : Set :=
| pleaf : A -> ptree A
| pnode : ptree (A * A) -> ptree A.
```

Perfekte Bäume entsprechen vollständigen Blatt-markierten Binärbäumen. Ein vollständiger Binärbaum hat immer genau 2^n Blätter.

```
Implicit Arguments pleaf [A].
Implicit Arguments pnode [A].

Definition pt3 := pnode (pnode (pnode (pleaf
                  (((0,1),(2,3)),((4,5),(6,7)))))).
```

Der Baum `pt3` hat die Höhe 3 und $2^3 = 8$ Blätter von Typ `nat`. Diese sind repräsentiert als ein Blatt vom Typ:

```
((nat * nat) * (nat * nat)) * ((nat * nat) * (nat * nat))
```

Weiterführende Literatur: Bird et al. [BP99], Abel et al. [AMU05].

Folgende Verwendung von Parametern ist ungültig:

```
Inductive exp (A : Set) : Type :=
| num  : nat -> exp nat
| pair : forall B, exp A -> exp B -> exp (A * B).
```

Im Zieltyp eines Konstruktors muss der Parameter die Variable `A` sein.

3.1.4 Positivität

Negative Datentypen bedingen Nichttermination ohne explizite Rekursion.

3.2 Induktive Prädikate und Relationen

Transitive Hülle.

$\leq$ für natürliche Zahlen.

Propositionale Gleichheit.

Kapitel 4

Abhängige Typen

4.1 Induktive Familien

Bsp. Typ der Pfade. Typ der Vektoren (in der Mathematik).

4.2 Abhängiger Funktionsraum und abhängiger Paartyp

Pi und Sigma typen: Curry-Howard-Iso und Gleichheitsregeln.

4.3 Typ- und Gleichheitsregeln für induktive Familien

Regeln für Konstruktoren.
Eliminationsprinzipien.
Definitionale Gleichheit.
Universen.
Große Eliminationen.
Beweis der Distinguiertheit der Konstruktoren. [GMM06]

Literaturverzeichnis

[AMU05] Andreas Abel, Ralph Matthes, and Tarmo Uustalu. Iteration schemes for higher-order and nested datatypes. *Theoretical Computer Science*, 333(1–2):3–66, 2005.

[BP99] Richard S. Bird and Ross Paterson. De Bruijn notation as a nested datatype. *Journal of Functional Programming*, 9(1):77–91, 1999.

[GMM06] Healfdene Goguen, Conor McBride, and James McKinna. Eliminating dependent pattern matching. In Kokichi Futatsugi, Jean-Pierre Jouannaud, and José Meseguer, editors, *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, volume 4060 of *Lecture Notes in Computer Science*, pages 521–540. Springer-Verlag, 2006.

[JM03] Felix Joachimski and Ralph Matthes. Short proofs of normalization. *Archive of Mathematical Logic*, 42(1):59–87, 2003.

[Kol32] Andrei Kolmogorov. Zur Deutung der intuitionistischen Logik. *Mathematische Zeitschrift*, 35:58–65, 1932.

[Mos07] Joan Moschovakis. Intuitionistic logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2007.