



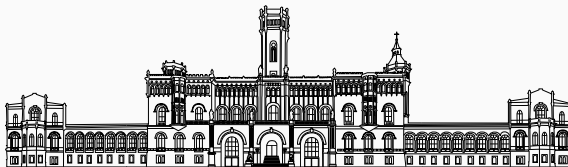
# Homotopietheorie

Oberseminar Theoretische Informatik

---

Florian Chudigiewitsch

Institut für Theoretische Informatik

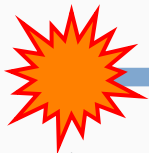


- Geschichtliches
- Grundlagen der Typentheorie
- Homotopietypentheorie
- Anwendungen & Aktuelle Forschungsfragen

The image features a white square background with four blue triangles positioned at the corners. The top-left and bottom-left triangles are a light blue color, while the top-right and bottom-right triangles are a darker blue color. The word "Geschichtliches" is centered in the white space.

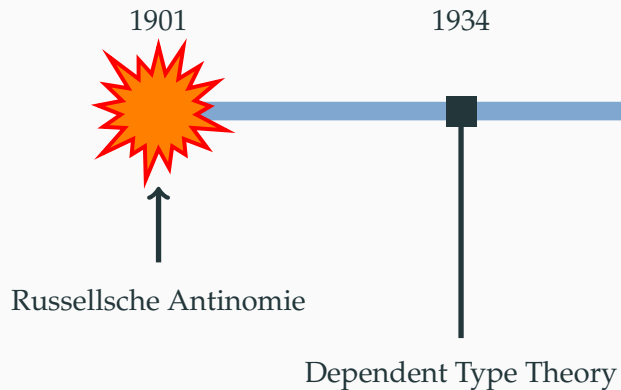
Geschichtliches

1901

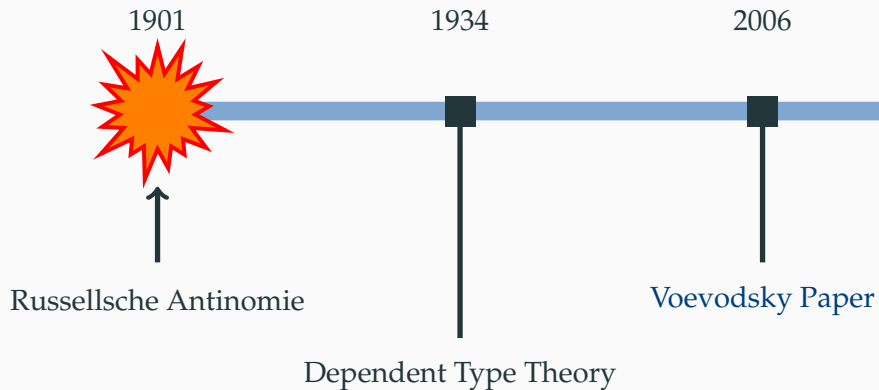


Russellsche Antinomie

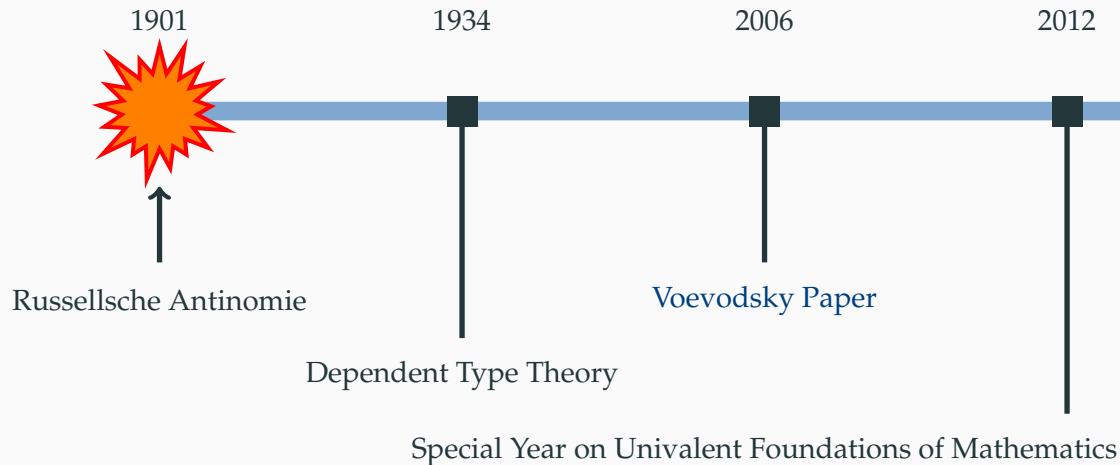
# Geschichtliches



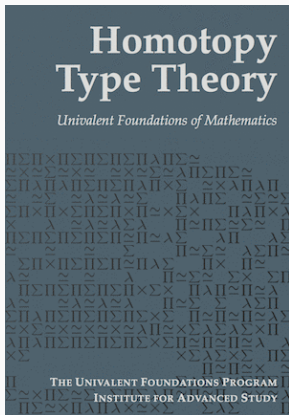
# Geschichtliches



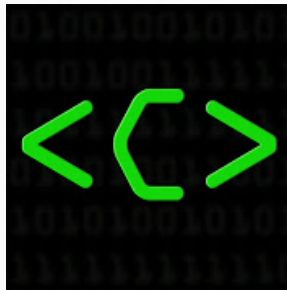
# Geschichtliches



# Quellen und Referenzen

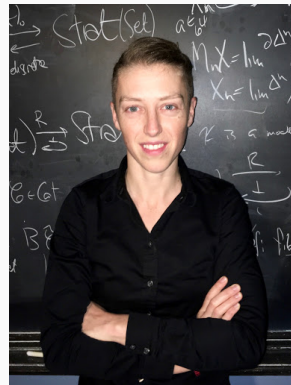


<https://homotopytypetheory.org/book/> [Uni13]



## Computerphile (Youtube)

- Type Theory
- Propositions as Types
- Voevodsky
- Homotopy Type Theory



## Emily Riehl

- Video
- Slides



The slide features four blue triangles in the corners. The top-left and bottom-left triangles are a light blue color, while the top-right and bottom-right triangles are a darker blue color. They are all right-angled triangles pointing towards the center of the slide.

# Grundlagen der Typentheorie

- Law of excluded middle (LEM) wichtiges Axiom in der klassischen Mathematik
- $\vdash \varphi \vee \neg\varphi$
- Ermöglicht Widerspruchsbeweise
- Konstruktive Logik verzichtet auf LEM
- Dadurch wird durch einen Beweis immer ein „Witness“ erzeugt
- Keine wirkliche Einschränkung, da man LEM jederzeit als Annahme hinzunehmen kann

# Dependent Type Theory: Die vier „Grundaussagen“

Die vier Grundformen („judgements“) der „wohlgeformten Formeln“ der Dependent Type Theory sind:

Formel	Interpretation	Beispiel
$\Gamma \vdash A \text{ type}$	„ $A$ ist ein Typ“	$\mathbb{N} \text{ type}$
$\Gamma \vdash a : A$	„ $a$ ist ein Term vom Typ $A$ “	$1 : \mathbb{N}$
$\Gamma, x : A \vdash B(x) \text{ type}$	„ $B(x)$ ist eine Typfamilie über $A$ “	$n : \mathbb{N} \vdash \mathbb{R}^n \text{ type}$
$\Gamma, x : A \vdash b(x) : B(x)$	„ $b(x)$ ist eine Termfamilie“	$n : \mathbb{N} \vdash \vec{0} : \mathbb{R}^n$

$\Gamma$  ist der *Kontext*, der die Typen aller vorkommenen Variablen deklariert.

*Universum*: Typ, dessen Elemente Typen sind. Bilden Hierarchie

$$\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \dots$$

# Dependent Type Theory: Die vier „Grundregelarten“ I

- Name:  $\times$ -formation rules
- Beschreibung: Haben wir Typen  $A$  und  $B$  gegeben, gibt es einen Produkttypen  $A \times B$ .
- Formal:

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash B \text{ type}}{\Gamma \vdash A \times B \text{ type}}$$

# Dependent Type Theory: Die vier „Grundregelarten“ II

- Name:  $\times$ -introduction rules
- Beschreibung: Haben wir Terme  $a : A$  und  $b : B$  gegeben, gibt es einen Term  $(a, b) : A \times B$ .
- Formal:

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : A \times B}$$

# Dependent Type Theory: Die vier „Grundregelarten“ III

- Name:  $\times$ -elimination rules
- Beschreibung: Haben wir einen Term  $p : A \times B$  gegeben, gibt es Terme  $\text{pr}_1(p) : A$  und  $\text{pr}_2(p) : B$ .
- Formal:

$$\frac{\Gamma \vdash p : A \times B}{\Gamma \vdash \text{pr}_1(p) : A}$$

$$\frac{\Gamma \vdash p : A \times B}{\Gamma \vdash \text{pr}_2(p) : B}$$

**Weiterhin:** *Judgemental equality* ( $\alpha$ -conversion), *computation rules* ( $\beta$ -reduction):  
verbinden introduction und elimination rules, optionales *uniqueness principle*  
( $\eta$ -expansion).

# Funktionstypen

- $\rightarrow$ -formation: Haben wir Typen  $A$  und  $B$  gegeben, gibt es einen Typen  $A \rightarrow B$ .
- $\rightarrow$ -introduction: Haben wir im Kontext eines Terms  $x : A$  einen Term  $b(x) : B$  gegeben, gibt es einen Term  $\lambda x.b(x) : A \rightarrow B$ . Formal:

$$\frac{\Gamma, x : A \vdash b(x) : B}{\Gamma \vdash \lambda x.b(x) : A \rightarrow B}$$

- $\rightarrow$ -elimination: Haben wir Terme  $f : A \rightarrow B$  und  $a : A$  gegeben, gibt es einen Term  $f(a) : B$ .
- Zwei computation rules.

# Propositions as Types

- Aussagen werden durch Typen repräsentiert
- Man beweist sie indem man einen Term vom entsprechenden Typ erzeugt

- 

$$\frac{\text{Beweise}}{\text{Aussagen}} = \frac{\text{Programme}}{\text{Typen}}$$

- Klassisch:  $\text{Prop} \leftrightarrow \text{Bool}$ , „Wahrheit“
- Andere Möglichkeit:  $\text{Prop} \leftrightarrow \text{Types}$ , „Zeugnis“



# Propositions as Types: Beispiel

## Beispiel 1

**Aussage:** Für beliebige Typen  $P$  und  $Q$  gibt es den Term

$$\text{modus-ponens} : P \times (P \rightarrow Q) \rightarrow Q.$$

**Beweis:** Mit  $\rightarrow$ -introduction aus Term  $x : P \times (P \rightarrow Q)$  einen Term aus  $Q$  generieren.  $x$ -elimination liefert uns  $\text{pr}_1(x) : P$  und  $\text{pr}_2(x) : P \rightarrow Q$ .  
 $\rightarrow$ -elimination liefert  $(\text{pr}_2(x))(\text{pr}_1(x)) : Q$ .

Somit:  $\text{modus-ponens} \equiv \lambda x. (\text{pr}_2(x))(\text{pr}_1(x))$ . ■

*Curry-Howard-Isomorphismus:* Ein Beweis korrespondiert zu einem Computerprogramm, welches einen Term vom Typ der Aussage zurückgibt.

# Gleichheit als Identitätstyp

Mathematische Gleichheit wird über Identitätstypen ausgedrückt.

- =-formation: Haben wir einen Typ  $A$  und zwei Terme  $x, y : A$  gegeben, gibt es einen Typ  $x =_A y$ .
- =-introduction: Haben wir einen Term  $x : A$  gegeben, so gibt es einen Term  $\text{refl}_x : x =_A x$ .

Elimination rule via path induction:

Haben wir eine Typfamilie  $\Gamma, x, y : A, p : x =_A y \vdash B(x, y, p)$  gegeben und wollen einen Term von  $B(x, y, p)$  konstruieren, reicht es anzunehmen, dass  $y$   $x$  und  $p$   $\text{refl}_x$  ist.

# Gleichheit ist eine Äquivalenzrelation

**Lemma 2** Für beliebige  $x, y : A$  gilt  $(x =_A y) \rightarrow (y =_A x)$ .

**Beweis.**

Hausaufgabe. ☺



**Lemma 3** Für beliebige  $x, y, z : A$  gilt  $(x =_A y) \rightarrow ((y =_A z) \rightarrow (x =_A z))$ .

**Beweis.**

Hausaufgabe. ☺

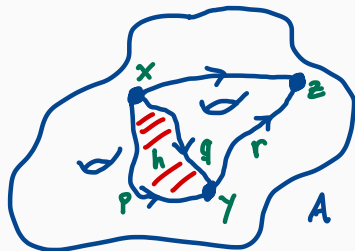


The slide features four blue triangles in the corners, each with its hypotenuse facing the center. The top-left and bottom-left triangles are a light blue, while the top-right and bottom-right triangles are a darker blue. The text is centered in the middle of the slide.

# Homotopietheorie

# Path induction homotopisch interpretiert

- Typ  $A \leftrightarrow$  Raum  $A$
- Term  $a : A \leftrightarrow$  Punkt  $a$  in  $A$
- Term  $p : x =_A y \leftrightarrow$  Pfad  $p$  von  $x$  nach  $y$  in  $A$
- Term  $p : p =_{x=Ay} a \leftrightarrow$  Homotopie  $h$  von  $p$  nach  $q$  in  $A$

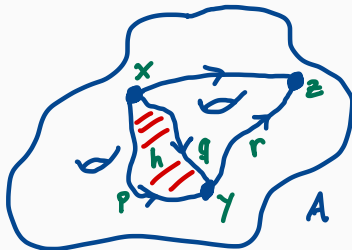


- Symmetrie und Transitivität wird als Umkehrung und Komposition von Pfaden, Homotopien, höheren Homotopien... interpretiert.
- **van den Berg/Garner** und **Lumsdaine**: Typen haben die Struktur eines schwachen  $\infty$ -Gruppoiden
- Unterschied: Homotopietheorie analytisch, Homotopietypentheorie synthetisch

# „Indiscernibility of identicals“

## Beantwortete offene Frage der Beweistheorie:

- „Indiscernibility of identicals“: Wenn zwei Beweise  $p$  und  $q$  beide  $A$  zeigen, kann man dann immer  $p = q$  zeigen?
- Nein!
- Homotopie-Äquivalenzklassen von Schleifen an einem Punkt  $x_0$  bilden die *fundamentale Gruppe*.

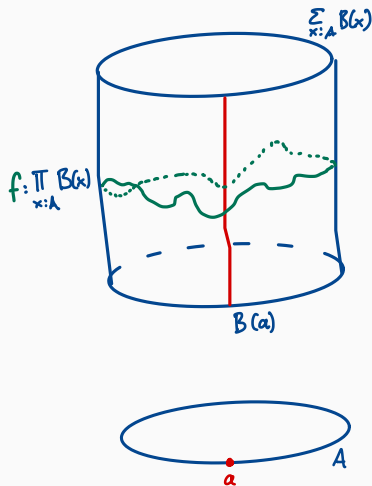


# Weitere Typen – mit homotopischer Interpretation

- Typfamilie  $x : A \vdash B(x)$  type  $\Leftrightarrow$  Faserung über  $A$
- Dependent sum Typ  $\sum_{x:A} B(x) \Leftrightarrow$  Totalraum einer Faserung

Beispiel:

$$\text{Gruppoid} \equiv \sum_{A:\mathcal{U}} (A \rightarrow A \rightarrow A)$$

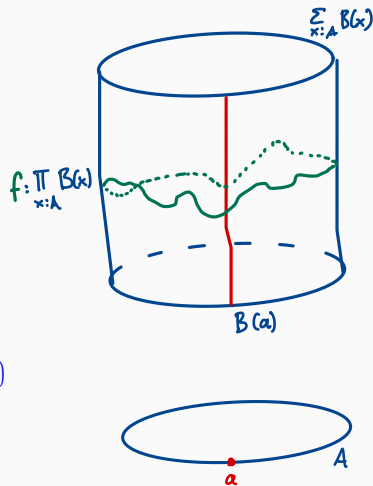


# Weitere Typen – mit homotopischer Interpretation

- Typfamilie  $x : A \vdash B(x)$  type  $\Leftrightarrow$  Faserung über  $A$
- Dependent function Typ  $\prod_{x:A} B(x) \Leftrightarrow$  Raum der Sektionen

Beispiel:

$$\text{swap} : \prod_{A:\mathcal{U}} \prod_{B:\mathcal{U}} \prod_{C:\mathcal{U}} (A \rightarrow B \rightarrow C) \rightarrow (B \rightarrow A \rightarrow C)$$





## Definition 4: (Zusammenziehbare Typen)

Es gibt einen eindeutigen Term vom Typ  $A$  gdw.

$$\sum_{a:A} \prod_{x:A} a =_A x$$

bewohnt ist, bzw. wenn der Raum  $A$  *zusammenziehbar* ist.

## Definition 5: (Typenäquivalenz)

Zwei Typen  $A$  und  $B$  sind *äquivalent*, wenn der Typ

$$A \simeq B \equiv \sum_{f:A \rightarrow B} \left( \sum_{g:B \rightarrow A} \prod_{a:A} g(f(a)) =_A a \right) \times \left( \sum_{h:B \rightarrow A} \prod_{b:B} f(h(b)) =_B b \right)$$

bewohnt ist.

Man kann leicht beweisen, dass

$$(A = B) \rightarrow (A \simeq B).$$

**Definition 6: (Univalenzaxiom (Voevodsky))**

$$(A = B) \simeq (A \simeq B)$$

# Der „Stein von Rosette“ für HoTT

Typen	Logik	Mengen	Homotopie
$A$	Aussage	Menge	Raum
$a : A$	Beweis	Element	Punkt
$B(x)$	Prädikat	Mengenfamilie	Faserung
$b(x) : B(x)$	Bedingter Beweis	Elementfamilie	Sektion
$0, 1$	$\perp, \top$	$\emptyset, \{\emptyset\}$	$\emptyset, \star$
$A + B$	$A \vee B$	Disjunkte Vereinigung	Coprodukt
$A \times B$	$A \wedge B$	Menge von Paaren	Produktraum
$A \rightarrow B$	$A \Rightarrow B$	Menge von Funktionen	Funktionsraum
$\sum_{(x:A)} B(x)$	$\exists_{x:A} B(x)$	Disjunkte Summe	Totalraum
$\prod_{(x:A)} B(x)$	$\forall_{x:A} B(x)$	Produkt	Raum der Sektionen
$\text{Id}_A$	Gleichheit ( $=$ )	$\{(x, x) \mid x \in A\}$	Pfadraum $A^I$

# Was gibt es noch?

- HoTT und der  $\lambda$ -Kalkül
- Higher inductive types
- Beweistheorie in HoTT

The slide features four blue triangles in the corners, each pointing towards the center. The top-left and bottom-left triangles are a light blue, while the top-right and bottom-right triangles are a darker blue. The text is centered in a dark blue, serif font.

# Anwendungen & Aktuelle Forschungsfragen

- Homotopietheorie
- Kategorientheorie
- Theorembeweiser
- Programmverifikation
- Funktionale Programmierung

- Informelle Typentheorie
- Formalisierung der klassischen Mathematik in HoTT
- Konstruktivität des Univalenzaxioms (Cubical Type Theory)
- HoTT auf diskreten Räumen (wie  $\mathbb{N}$ ) führt zu vielen „unnötigen“ Identitätstermen
  - Möglichkeit, diese zu kollabieren
- HoTT und Topoi
  - Intuitionistische Higher Order Logic ist die interne Sprache von 1-Topoi, HoTT könnte die von  $(\infty, 1)$ -Topoi sein





# Aktuelle Forschungsfragen: Komplexitätstheorie?

- Wenn HoTT Grundlage der Mathematik sein kann, muss man mit ihr auch Komplexitätstheorie betreiben können
- Klassische Komplexitätstheorie sehr „mengenzentriert“
- HoTT eng verbunden mit funktionalen Programmiersprachen
- Mathematische Strukturen sind „First Class Citizens“ in HoTT
  - Codierung egal
- Eher rekursionstheoretische Ansätze erforderlich [[Con95](#)]
  - Maschinenunabhängige Komplexitätstheorie
  - Implizite Komplexitätstheorie [[Lag12](#)]

# Aktuelle Forschungsfragen: Komplexitätstheorie?



Begriff der  
„Linearität“ in HoTT\*

\* Danke an Prof. Thorsten Altenkirch [McB; Atk18]

# Aktuelle Forschungsfragen: Komplexitätstheorie?



Begriff der  
„Linearität“ in HoTT\*

Definition von Komplexitätsmaßen

\* Danke an Prof. Thorsten Altenkirch [McB; Atk18]

# Aktuelle Forschungsfragen: Komplexitätstheorie?



Begriff der  
„Linearität“ in HoTT\*

Definition von Komplexitätsmaßen

Charakterisierung von sinnvollen  
Komplexitätsklassen und Hierarchien

\* Danke an Prof. Thorsten Altenkirch [McB; Atk18]

# Aktuelle Forschungsfragen: Komplexitätstheorie?



Begriff der  
„Linearität“ in HoTT\*

Definition von Komplexitätsmaßen

Charakterisierung von sinnvollen  
Komplexitätsklassen und Hierarchien

Charakterisierung von effizienter Berechenbarkeit [Ste]

\* Danke an Prof. Thorsten Altenkirch [McB; Atk18]

Danke!

Fragen?



Robert Atkey. *Syntax and Semantics of Quantitative Type Theory*. 2018. DOI: <https://doi.org/10.1145/3209108.3209189>.



R.L. Constable. “Expressing computational complexity in constructive type theory”. In: *Leivant D. (eds) Logic and Computational Complexity* (1995). DOI: [https://doi.org/10.1007/3-540-60178-3\\_82](https://doi.org/10.1007/3-540-60178-3_82).




U. Dal Lago. “A Short Introduction to Implicit Computational Complexity”. In: *Bezhanishvili N., Goranko V. (eds) Lectures on Logic and Computation* (2012). DOI: [https://doi.org/10.1007/978-3-642-31485-8\\_3](https://doi.org/10.1007/978-3-642-31485-8_3).



 Conor McBride. *I Got Plenty o' Nuttin'*. URL: <https://personal.cis.strath.ac.uk/conor.mcbride/PlentyO-CR.pdf>.

 Stephen Cook Stephen Bellatoni. *A New Recursion-Theoretic Characterization of the Polytime Functions*. URL: <https://www.cs.toronto.edu/~sacook/homepage/ptime.pdf>.

 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: <https://homotopytypetheory.org/book>, 2013.