

# Foundations of *Grundlagen*

Chad E. Brown

`cebrown@ags.uni-sb.de`

Universität des Saarlandes, Saarbrücken, Germany

# Motivation

- I want a Mathematical Assistant System.
- Requirements:
  - Large Library of Definitions, Theorems, Proofs
  - Proof Checking
  - Reasonable Automated Inference
- How can all the Math in the Library Live Together in Harmony?
- Practical, Natural Representation Language
- Practical Foundational System

# Foundations

- What is a practical foundation for Mathematics?
- Criteria:
  - Sufficient to Express “Serious” Math
  - Allow Natural Expression of Ideas
  - Consistent?

# Grundlagen: The History

## Case Study:

- *Grundlagen der Analysis*: Edmund Landau 1920's
- Formalized by L. S. van Benthem Jutting in Automath 1970's
- Recovered by Freek Wiedijk 1990's

# Grundlagen: The Book

Mathematical Constructions in Landau's *Grundlagen der Analysis*:

- (Positive) Natural Numbers:  $1, Suc$
- Fractions (Pairs of Natural Numbers)
- (Positive) rationals (equivalence classes of pairs)
- Cuts (some sets of fractions)
- Real Numbers (Cut,  $-Cut$ , or  $0$ )
- Complex Numbers (pairs of Real's)
- Sums and Products of  $n$ -tuples of Complex Numbers

# What do we need?

- (Positive) Natural Numbers:  $1, Suc$
- Peano Axioms
- Pairs (of Natural Numbers, of Reals)
- Quotients by Equivalence Classes
- Sets (of rationals)
- $n$ -tuples for natural numbers  $n$ .

# Grundlagen: The Automath

Automath – Aut-QE “Quasi-Expression”

Aut-QE provides:  $\lambda$ , application, PROP and TYPE

- Types: If  $\sigma$  and  $\tau$  are TYPE's, then  $\lambda x : \sigma . \tau$  is a TYPE (essentially  $\sigma \rightarrow \tau$ ).
- We have simple types.
- In general,  $\tau(x)$  can be a TYPE depending on  $x : \sigma$  and  $\lambda x : \sigma . \tau$  corresponds to  $\Pi_{x:\sigma} \tau$ .
- We also have dependent types.

# Grundlagen: The Automath

Automath – Aut-QE “Quasi-Expression”

Aut-QE provides:  $\lambda$ , application, PROP and TYPE

- Propositions.

$@ [ a : \text{PROP} ] [ b : \text{PROP} ]$

$\text{imp} := [ x : a ] b : \text{PROP}$

- $\text{imp}(a, b)$  means  $a \supset b$ , but is defined as  $\lambda x : a . b$ .

Idea: If you apply a proof of  $a \supset b$  to a proof of  $a$ , you obtain a proof of  $b$ .

# Jutting's Automath: Logic

- Logic is assumed Classical:

```
@[a : PROP] [w : wel(a)] et := 'prim' : a
```

et (primitive): From a proof of  $\neg\neg a$ , we have a proof of  $a$ .

# Equality and Description

- Equality

$\text{is}(\text{sigma}, x, y)$  : Equality of  $x$  and  $y$  in type  $\sigma$

- Description

- $\text{ind}(\text{sigma}, p, o1)$  : If  $p : \sigma \rightarrow PROP$  is a  $\sigma$ -predicate and  $o1$  is a proof that exactly one  $\sigma$ -element satisfies  $p$ , then  $\text{ind}(\text{sigma}, p, o1)$  is a  $\sigma$ -element.
- $\text{oneax}(\text{sigma}, p, o1)$  :  $\text{ind}(\text{sigma}, p, o1)$  satisfies  $p$ .
- Note:  $\text{ind}(\text{sigma}, p, o1)$  is a term depending on a proof!

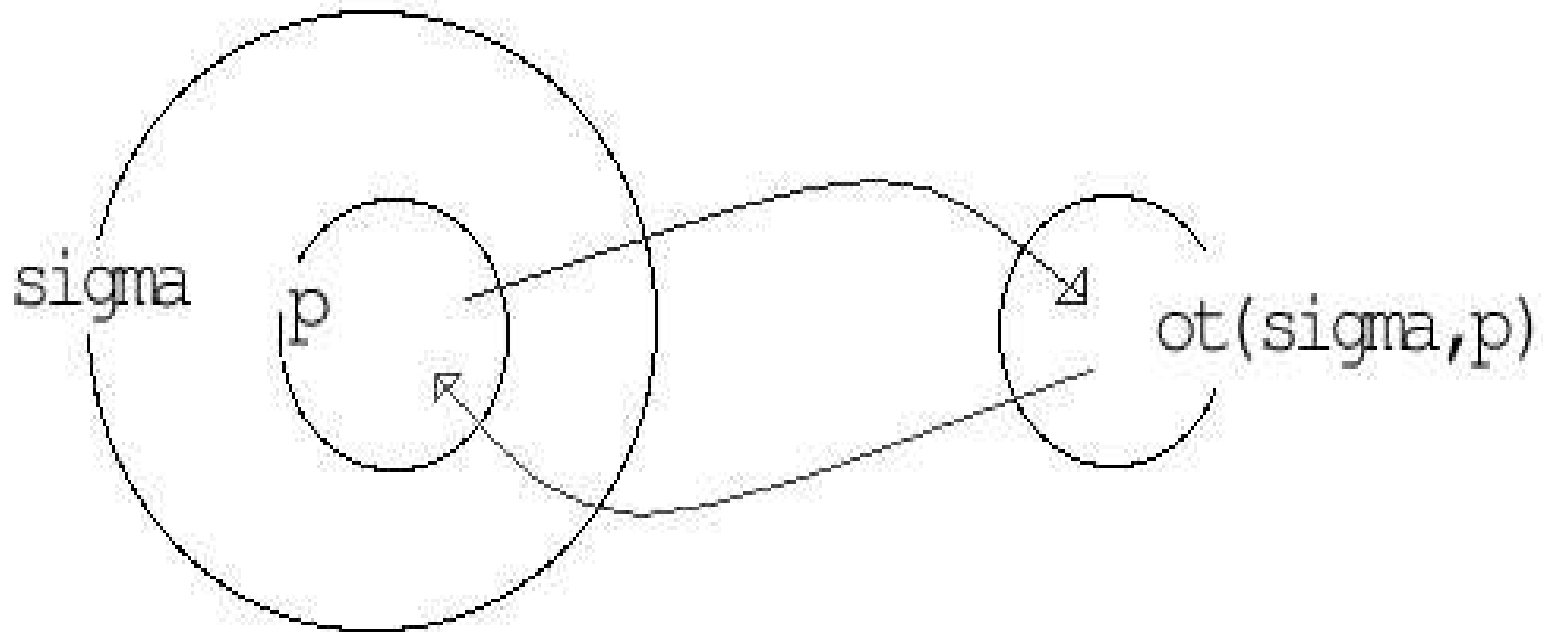
# Functional Extensionality

`f i s i`

For types  $\sigma$  and  $\tau$  and functions  $f, g : \sigma \rightarrow \tau$ , if  $f(x)$  is <sup>$\tau$</sup>   $g(x)$  for all  $x : \sigma$ , then  $f$  is <sup>$\sigma \rightarrow \tau$</sup>   $g$ .

# Predicate Types

- $\text{ot}(\text{sigma}, p)$ : Given any type  $\sigma$  and predicate  $p : \sigma \rightarrow \text{PROP}$ ,  $\text{ot}(\text{sigma}, p)$  is a type isomorphic to the collection of  $\sigma$ -elements satisfying  $p$ .



- Departure from simple types.

# Set Types

- `set (sigma)`: type of sets of  $\sigma$ -elements.

# Natural Numbers

- `nat`: type of natural numbers
- Primitive Constants: `1` and `suc`.
- Axioms: Peano 3, 4, and 5.

# End of Jutting's Primitives

- There are no more primitives in Jutting's translation.
- How is the Grundlagen encoded with these primitives?

# Jutting's Encoding

- Fractions are pairs of natural numbers. What is a pair?
- A pair of two things of type  $\sigma$  is a function from  $\{1, 2\}$  to  $\sigma$ .
- IE: A pair is of type  $1to(2) \rightarrow \sigma$ .
- What is the type  $1to(2)$ ?
- Use type  $nat$  and predicate for “less than or equal to  $n$ ” to define the type  $1to(n)$  for each  $n$ .

# Jutting's Encoding

- Positive Rationals is type of Equivalence Classes of Fractions.
- Equivalence Class Types are defined using Predicate Types.
- ETC...

# Porting From Automath

- Extensive use of Predicate Types
- Can these be eliminated (along with other Aut-QE features) to obtain a simply typed version of the encoding?
- Yes!
- The encoding has been ported (**WITHOUT PROOFS**) to TPS.

# TPS Encoding

- After porting to simply-typed higher-order logic, there are 3 axioms and one axiom schema:
  - Three Peano Axioms at base type  $\iota$ .
  - Description at all types.
- There are 5998 open theorems which should follow from the axioms.

# How do the encodings compare?

- `satz289b`: If  $f$  is an  $x$ -tuple of complex numbers,  $n$  is between 1 and  $x$ , and  $f_n = 0$ , then the product  $f_1 \cdots f_x$  of the  $x$ -tuple is 0.

Jutting proves this using the previously proven:

- `satz289`: Any  $x$ -tuple of complex numbers has product 0 iff there exists some  $n$  between 1 and  $x$  such that  $f_n = 0$ .

# Automath Version

- Theorem:

```
[x:nat][f:[t:1to(x)]cx][n:1to(x)]  
[i:is(<n>f,0c)]  
...is(prod(x,f),0c)
```

- Proof:

```
[x:nat][f:[t:1to(x)]cx][n:1to(x)]  
[i:is(<n>f,0c)]  
satz289b:=th4"1.iff"(prop1".8289",  
prop2".8289",satz289,t41".8289")  
:is(prod(x,f),0c)
```

# Simply Typed Version

Theorem:

$$\begin{aligned} & \forall x_i \forall f_{o(o(o(u))\iota)u}. \forall i_\iota \forall j_\iota [_1\text{TO } o_{uu} x_i j \supset \\ & \quad \_CX_{o(o(o(o(u))\iota)\iota)(o(o(o(u))\iota)\iota)} [f_i]. f_j] \\ & \quad \supset \forall n_\iota. \_1\text{TO } x_n n \supset . \\ & \quad \_C\_IS_{o(o(o(o(u))\iota)\iota)(o(o(o(u))\iota)\iota)} [f_n] \_0C_{o(o(o(u))\iota)\iota} \\ & \supset \_C\_IS [_C\_PROD_{o(o(o(u))\iota)\iota)(o(o(o(u))\iota)u)\iota} x f] \_0C \end{aligned}$$

# Simply Typed Version

## Theorem Omitting Types

$$\begin{aligned} & \forall x \forall f. \forall i \forall j [\_1\text{TO } x \ i \ j \supset \_C\text{X } [f \ i]. \ f \ j] \\ & \supset \forall n. \_1\text{TO } x \ n \ n \supset \_C\text{IS } [f \ n] \_0\text{C} \\ & \supset \_C\text{IS } [\_C\text{PROD } x \ f] \_0\text{C} \end{aligned}$$

# Simply Typed Proof 1

Proof (relative to appropriate hypotheses): ...Interactively  
Constructed Proof...

- (1)  $1 \vdash \_8289\_T41 \wedge \_SATZ289 \wedge \_IFF\_TH4$  Hyp
- (2)  $1 \vdash \_8289\_T41 \wedge \_SATZ289$  Conj: 1
- (3)  $1 \vdash \_SATZ289$  Conj: 2
- (4)  $1 \vdash \forall x \forall f. \forall i \forall j [ \_1TO \ x \ i \ j$   
 $\qquad \qquad \qquad \supset \_CX \ [f \ i]. \ f \ j]$   
 $\qquad \qquad \qquad \supset . \_C\_IS \ [\_C\_PROD \ x \ f]\_0C$   
 $\qquad \qquad \qquad \equiv \exists n. \_1TO \ x \ n \ n$   
 $\qquad \qquad \qquad \wedge \_C\_IS \ [f \ n]\_0C$
- (5)  $5 \vdash \forall i \forall j. \_1TO \ x \ i \ j \supset \_CX \ [f \ i]. \ f \ j$  Defn: 3
- Hyp

# Simply Typed Proof 2

$$(6) \quad 1 \vdash \quad \forall f. \forall i \forall j [ \_1\text{TO } x \ i \ j \\ \supset \_ \text{CX } [f \ i]. \ f \ j] \\ \supset . \ \_ \text{C\_IS } [ \_ \text{C\_PROD } \ x \ f ] \_ \text{OC} \\ \equiv \exists n. \ \_1\text{TO } \ x \ n \ n \\ \wedge \_ \text{C\_IS } [f \ n] \_ \text{OC}$$

UI:  $x$  4

$$(7) \quad 1 \vdash \quad \forall i \forall j [ \_1\text{TO } x \ i \ j \supset \_ \text{CX } [f \ i]. \ f \ j] \\ \supset . \ \_ \text{C\_IS } [ \_ \text{C\_PROD } \ x \ f ] \_ \text{OC} \\ \equiv \exists n. \ \_1\text{TO } \ x \ n \ n \\ \wedge \_ \text{C\_IS } [f \ n] \_ \text{OC}$$

UI:  $f$  6

$$(8) \quad 5 \vdash \quad \forall i \forall j. \ \_1\text{TO } \ x \ i \ j \supset \_ \text{CX } [f \ i]. \ f \ j$$

AB: 5

$$(9) \quad 1, 5 \vdash \quad \_ \text{C\_IS } [ \_ \text{C\_PROD } \ x \ f ] \_ \text{OC} \\ \equiv \exists n. \ \_1\text{TO } \ x \ n \ n \\ \wedge \_ \text{C\_IS } [f \ n] \_ \text{OC}$$

MP: 8 7

# Simply Typed Proof 3

(10) 1,5 ⊢ [  $\_C\_IS$  [  $\_C\_PROD$   $x$   $f$  ]  $\_0C$  ]  
⊃  $\exists n. \_1TO$   $x$   $n$   $n$

$\wedge \_C\_IS$  [  $f$   $n$  ]  $\_0C$  ]  
 $\wedge. \exists n$  [  $\_1TO$   $x$   $n$   $n$   
 $\wedge \_C\_IS$  [  $f$   $n$  ]  $\_0C$  ]

⊃  $\_C\_IS$  [  $\_C\_PROD$   $x$   $f$  ]  $\_0C$

EquivImp: 9

(11) 1,5 ⊢  $\exists n$  [  $\_1TO$   $x$   $n$   $n$   
 $\wedge \_C\_IS$  [  $f$   $n$  ]  $\_0C$  ]  
⊃  $\_C\_IS$  [  $\_C\_PROD$   $x$   $f$  ]  $\_0C$

Conj: 10

(12) 12 ⊢  $\_1TO$   $x$   $n$   $n$

Hyp

(13) 13 ⊢  $\_C\_IS$  [  $f$   $n$  ]  $\_0C$

Hyp

(14) 12,13 ⊢  $\_1TO$   $x$   $n$   $n$   $\wedge$   $\_C\_IS$  [  $f$   $n$  ]  $\_0C$

RuleP: 12 13

(15) 12,13 ⊢  $\exists n. \_1TO$   $x$   $n$   $n$   
 $\wedge \_C\_IS$  [  $f$   $n$  ]  $\_0C$

EGen:  $n$  14

# Simply Typed Proof 4

- (16)  $1, 5, 12, 13 \vdash \_C\_IS \[_C\_PROD \ x \ f]\_0C$  MP: 15 11
- (17)  $1, 5, 12 \vdash \_C\_IS \ [f \ n]\_0C$   
 $\supset \_C\_IS \ [_C\_PROD \ x \ f]\_0C$  Deduct: 16
- (18)  $1, 5 \vdash \_1TO \ x \ n \ n$   
 $\supset . \_C\_IS \ [f \ n]\_0C$   
 $\supset \_C\_IS \ [_C\_PROD \ x \ f]\_0C$  Deduct: 17
- (19)  $1, 5 \vdash \forall n. \_1TO \ x \ n \ n$   
 $\supset . \_C\_IS \ [f \ n]\_0C$   
 $\supset \_C\_IS \ [_C\_PROD \ x \ f]\_0C$  UGen:  $n$  18
- (20)  $1 \vdash \forall i \forall j [\_1TO \ x \ i \ j \supset \_CX \ [f \ i]. \ f \ j]$   
 $\supset \forall n. \_1TO \ x \ n \ n$   
 $\supset . \_C\_IS \ [f \ n]\_0C$   
 $\supset \_C\_IS \ [_C\_PROD \ x \ f]\_0C$  Deduct: 19

# Simply Typed Proof 5

$$\begin{aligned} (21) \quad 1 \vdash \quad & \forall f. \forall i \forall j [ \_1\text{TO } x i j \\ & \quad \supset \_CX [f i]. f j ] \\ & \quad \supset \forall n. \_1\text{TO } x n n \\ & \quad \quad \supset . \_C\_IS [f n]\_0C \\ & \quad \quad \supset \_C\_IS [\_C\_PROD x f]\_0C \\ & \quad \quad \quad \text{UGen: } f \ 20 \end{aligned}$$

$$\begin{aligned} (22) \quad 1 \vdash \quad & \forall x \forall f. \forall i \forall j [ \_1\text{TO } x i j \\ & \quad \supset \_CX [f i]. f j ] \\ & \quad \supset \forall n. \_1\text{TO } x n n \\ & \quad \quad \supset . \_C\_IS [f n]\_0C \\ & \quad \quad \supset \_C\_IS [\_C\_PROD x f]\_0C \\ & \quad \quad \quad \text{UGen: } x \ 21 \end{aligned}$$

$$(23) \quad 1 \vdash \quad \_SATZ289B \quad \text{Defn: } 22$$

# Simply Typed Proof 6

(24)  $\vdash$

$\_8289\_T41 \wedge \_SATZ289 \wedge \_IFF\_TH4 \supset \_SATZ289B$

Deduct: 23

- 24 lines ...
- Not quite as concise as

```
[x:nat][f:[t:1to(x)]cx][n:1to(x)]  
[i:is(<n>f,0c)]  
satz289b:=th4"1.iff"(prop1".8289",  
prop2".8289",satz289,t41".8289")  
:is(prod(x,f),0c)
```

- Neither version is very readable.

# The Grundlagen Test

Every system should have a copy of the Grundlagen.

- Can all the concepts be encoded in the system?
- Can all the proofs be encoded in the system?
- Can all the proofs be *checked* in the system?
- How long does it take to check the proofs?
- What is the deBruijn factor?

*Who's better than Automath?*

# Future Work

- Porting Grundlagen to A Form of Set Theory...
- Encoding Portions of Other Texts:
  - Hardy, Wright *An introduction to the theory of numbers*
  - Dieudonne *Foundations of Modern Analysis*
  - Bartle, Sherbert *Introduction to Real Analysis*
  - Hungerford *Algebra*
  - MacLane *Categories for the Working Mathematician*