The Concept of Set in Constructive Analysis

Christian Neukirchen

München 2011

The Concept of Set in Constructive Analysis

Christian Neukirchen

Bachelorarbeit
am Mathematischen Institut
der Ludwig-Maximilians-Universität München

vorgelegt von
Christian Neukirchen
geboren in Biberach am 16.08.1987

München, den 9. Oktober 2011

Contents

Preface

In order to transfer results from different branches of mathematics, a mathematician has work to do: definitions need to be compared, concepts mapped, mismatches found, invariants kept. In this work, such a transfer happens on two levels: first, we want to state, analyze and prove classical results of analysis in a constructive setting; secondly, we need to formulate these results in the typed calculus of Minlog.

The first transfer has largely been made by Errett Bishop and Douglas Bridges in their seminal book "Constructive Analysis" [BB1985]. It is central to this thesis, in which we analyze the objects and foundations used for their formulation of large parts of classic analysis in a constructive way.

The second transfer, taking the proofs of "Constructive Analysis" and making them work together with types, we need to do ourselves. Luckily, the types are mostly unproblematic.

This thesis consists of five parts: first, we look at set theory and the lack thereof in Minlog's typed calculus; second, we research the different kinds of sets and their uses in [BB1985]; third, we formally prove some theorems, focussing about finite sets; fourth, we consider how measure theory can be formulated in a constructive setting; fifth, we conclude our work.

Acknowledgements. First of all, my thanks go to my advisor Prof. Helmut Schwichtenberg who, despite being a professor emeritus, gave me a Bachelor thesis. For his insight and continuous support, I'm very grateful.

Thanks also go to the participants of the seminar "Rechnerischer Gehalt von Beweisen", at which I presented an early version of the ideas that now make the second chapter; they gave constructive critique. I also want to thank Rod Carvalho for publishing his bookmark stream, where I have stumbled upon several interesting papers useful for this thesis.

Finally, special thanks goes to Douglas Bridges, who, during his visit in Munich, answered many questions in the early stages of this thesis.

— Christian Neukirchen
October 9, 2011

CHAPTER 1

Sets and Types

When formalizing classical mathematics, the most popular approach in the last century has been to use set theory as a foundation. However, non-axiomatic formulations of set theory such as the informal description by Cantor himself and Frege's *naive set theory* have quickly been shown to be inconsistent and liable to paradoxes, e.g. Russell's antinomy [Fre1893, Rus1902]. The axiomatisation of set theory, for example by Zermelo and Fraenkel, is one approach to avoid these issues.

When we deal with constructive mathematics, and especially with computer assisted proofs, a different foundation such as type theory is more viable. Type theory was invented by Russell and popularized in his attempt—together with Whitehead—to formulize all mathematical truth in *Principia Mathematica*. Later, it was developed further by Church into the simply typed λ-calculus. The Minlog system [HS2011] uses a such a simply typed λ-calculus, augmented with inductively defined types.

In order to constructively prove theorems from set-heavy fields in classical mathematics, such as measure theory, we thus need to shift paradigms: we need to encode use of sets into inductive definitions and typed functions if we wish to keep proofs similar. This is not possible in general, since "full" set theory is far too powerful and non-constructive for such a mapping to exist. However, we will see that one can heavily strip down set theory and still provide substitutes of most constructs.

When Bishop wrote *Foundations of Constructive Analysis* [Bis1967], he was deliberately frugal with details on foundations. While there have been foundations for Constructive Analysis which use untyped λ-calculi (e.g. [Fef1979]), it is well possible to adapt the book to an λ-calculus with inductively defined types.

Most readers versed with set theory as a foundation will know the representation of ordered pairs as sets by Kuratowski:

$$(a, b) = \{a, \{a, b\}\}$$

Such a set would not be allowed in our typed setting at all, because this construction has no type: one element of the set is of *type of a* while the second is of type *set of type of a and b*, whereas we require all elements of a sets to have the same type.

For the same reason, a set like this is not allowed:

$$\{1, \frac{2}{3}, \pi, \mathsf{true}\}$$

While we can regard the first three elements as real numbers, "true" is not a real number; thus the elements don't all have the same type, and the set is not well-typed. All possible set elements we consider are therefore contained in a *universe*. Classic constructions such as the compactification of the reals

$$\mathbb{R} \cup \{-\infty, +\infty\}$$

are not allowed per se. Rather, we need to define a new type $\overline{\mathbb{R}}$ which is a *sum type* of a real number and the infinities, and then we consider "subsets of $\overline{\mathbb{R}}$".

We even have to restrict sets further: only certain kinds of sets can be represented with a type, for example finite sets. For such sets, we also can define sets-of-sets, e.g. a finite set of finite sets of natural numbers. However, constructions like the powerset are not allowed in general.

Mathematicians familiar with the classic use of set theory are often shocked when they see how limited the constructive concepts of sets are. Yet, they are powerful enough to go a long way.

CHAPTER 2

Formalizing set concepts in Constructive Analysis

1. Preliminaries

The inductive set definitions we work with are meant to be used in the context of the Theory of Computable Functionals (TCF), a simply-typed lambda calculus with parameterized types. Predicates and objects are defined inductively.

In its most general presentation, TCF looks like this [HS2010, pp. 114–115] ($A, B, C \in \mathrm{F}(\vec{Y})$ are formula forms ranging over possibly zero predicate variables $\vec{Y} = Y_0, \ldots, Y_n$. $I, P \in \mathrm{Preds}(\vec{Y})$ are predicate forms and $K_i \in \mathrm{Cl}_X(\vec{Y})$ clause forms):

$$Y_l\vec{r} \in \mathrm{F}(\vec{Y}), \qquad \frac{A \in \mathrm{F} \quad B \in \mathrm{F}(\vec{Y})}{A \to B \in \mathrm{F}(\vec{Y})}, \qquad \frac{A \in \mathrm{F}(\vec{Y})}{\forall_x A \in \mathrm{F}(\vec{Y})}$$

$$\frac{C \in \mathrm{F}(\vec{Y})}{\{\vec{x} \mid C\} \in \mathrm{Preds}(\vec{Y})}, \qquad \frac{P \in \mathrm{Preds}(\vec{Y})}{P\vec{r} \in \mathrm{F}(\vec{Y})},$$

$$\frac{K_0, \ldots, K_{k-1} \in \mathrm{Cl}_X(\vec{Y})}{\mu_X(K_0, \ldots, K_{k-1}) \in \mathrm{Preds}(\vec{Y})} \quad (k \geq 1),$$

$$\frac{\vec{A} \in \mathrm{F}(\vec{Y}) \quad \vec{B}_0, \ldots, \vec{B}_{n-1} \in \mathrm{F}}{\forall_{\vec{x}}(\vec{A} \to (\forall_{\vec{y}_\nu}(\vec{B}_\nu \to X\vec{s}_\nu))_{\nu<n} \to X\vec{t}\,) \in \mathrm{Cl}_X(\vec{Y})} \quad (n \geq 0).$$

Every inductively defined predicate $I := \mu_X(K_0, \ldots, K_{k-1})$ gives rise to these axioms:

(Intro) $$\forall_{\vec{x}}(\vec{A} \to (\forall_{\vec{y}_\nu}(\vec{B}_\nu \to I\vec{s}_\nu))_{\nu<n} \to^{\mathrm{c}} I\vec{t}\,)$$

(Elim) $$\forall^{\mathrm{nc}}_{\vec{x}}(I\vec{x} \to^{\mathrm{c}} (K_i(I, P))_{i<k} \to^{\mathrm{c}} P\vec{x})$$

where

$$K_i(I, P) := \forall_{\vec{x}}(\vec{A} \to (\forall_{\vec{y}_\nu}(\vec{B}_\nu \to I\vec{s}_\nu))_{\nu<n} \to^{\mathrm{c}} (\forall_{\vec{y}_\nu}(\vec{B}_\nu \to P\vec{s}_\nu))_{\nu<n} \to^{\mathrm{c}} P\vec{t}\,)$$

In the following, we assume a reasonable definition of the natural numbers, a boolean data type and tuples, as well as a construction of whole and rational numbers based upon them.

A *function* $f : A \to B$ from a set A (its domain *dmn* A) to a set B (the codomain *cod* B) is a finite routine for constructing $f(a)$ from an object $a \in A$ which satisfies $f(a) = f(a')$ whenever $a, a' \in A$ and $a = a'$.

If $f : A \to B$ and $g : B \to A$ are functions such that $g(f(a)) = a$ and $f(g(b)) = b$ for all $a \in A, b \in B$, we say that f is *bijective* and call g its inverse (and vice versa). We also write $g = f^{-1}$. We say that A *has a bijection* to B.

2. Real numbers

Since real numbers are at the core of all analysis and there are several issues arising in their constructive use, we will give them a closer look here.

Bishop/Bridges give a simple construction of the real numbers in [BB1985, Section 2.2]: a *real number* is defined as a sequence (x_n) of rational numbers, such that

$$|x_m - x_n| \leq m^{-1} + n^{-1} \qquad (m, n \in \mathbb{N})$$

Then, two real numbers $x := (x_n)$, $y := (y_n)$ are equal if

$$|x_n - y_n| \leq 2n^{-1} \qquad (n \in \mathbb{N})$$

Minlog however, uses a slightly different exposition with a varying *module*, explained in detail in [HS2007]: a *real number* is defined as a pair of a sequence (x_n) of rational numbers together with a function $M : \mathbb{N} \to \mathbb{N}$, such that:

$$|x_m - x_n| \leq 2^{-k} \qquad (m, n \geq M(k))$$
$$M(k) \leq M(l) \qquad (k \leq l).$$

With this definition, two real numbers $x := (x_n, M)$, $y := (y_n, N)$ are equal if

$$|a_{M(k+1)} - b_{N(k+1)}| \leq 2^{-k} \qquad (k \in \mathbb{N})$$

This definition is easier for proofs, especially when working with power series operating on real numbers. The moduli will be omitted when they are clear or irrelevant.

It is important to realize that (in)equality **cannot** always be shown for real numbers: Let n_k be 0 if $k - 4$ can be expressed as the sum of two primes, and 1 else. Then define $G_k = n_k \cdot 2^{-k}$. It can be shown that $G \geq 0$, but showing *whether* $G > 0$ or $G = 0$ requires proving the Goldbach conjecture.

A set with a finite routine to check whether any two elements is equal is called a *discrete set*.

3. Sets and setoids

To quote [BB1985]:

> *. . . a* set *is defined by describing what must be done to construct an* element *of the set, and what must be done to show that two elements of the set are* equal.

Commonly, a set-with-equivalence is called a *setoid*, but we will usually regard setoids with an implied equivalence and just call them *sets*. Note that two sets with the same objects but different definitions of equality are considered fundamentally different. (Take the real numbers, and then define all elements to be equal. The resulting set will be finite.)

The well-known set operations such as intersection, union and set difference are defined in intuitionistic ways: For two sets A and B, to construct an element of the union $A \cup B$, we need to *construct* an element $a \in A$ or an element $b \in B$. To construct an element of the intersection $A \cap B$, we need to construct elements $a \in A$ and $b \in B$ that are considered equal.

Subsets are defined in [BB1985] by *inclusion maps*, which are restrictions of the identity on the superset: A *subset* $A \subset B$ is a pair (A, i) consisting of a set A and a function $i : A \to B$, called the *inclusion map*, such that for all a, a' in A, $a = a'$ if and only if $i(a) = i(a')$.

Relations like $A \subset B$ and $A = B$ ($:= A \subset B \wedge B \subset A$) are shown by construction of such inclusion maps.

4. Sets given by a formula

The perhaps most natural and general representation is the one given by a property of its elements, as a formula F_S for elements of the universe:

$$S := \{x \in U \mid F_S(x)\}$$

Set operations are easy to define by logical operators:

$$\begin{aligned} S \cup T &:= \{x \mid F_S(x) \vee F_T(x)\} \\ S \cap T &:= \{x \mid F_S(x) \wedge F_T(x)\} \\ S - T &:= \{x \mid F_S(x) \wedge \neg F_T(x)\} \end{aligned}$$

Set difference (and complementation $\bar{S} = U - S$) have the usual problems related to negation in constructive mathematics. Later, we will see set constructions where we do not need negation for complementation.

Sets given by a formula are difficult to deal with in general. Thus, we will take a closer look at set constructions and specialized set concepts that are more appropriate for constructive mathematics.

5. Finite and subfinite sets

A set which has a bijection to a subset $N_n = \{0, \ldots, n\} \subset \mathbb{N}$ up to some $n \in \mathbb{N}$ is called *finite*. That is, we need to construct two functions $f, g : \mathbb{N} \to \mathbb{N}$ such that $\forall_{i \leq n}(f(g(i)) = i \wedge g(f(i)) = i)$.

For some $n \in \mathbb{N}$, if there is a mapping from $N_n = \{0, \ldots, n\} \subset \mathbb{N}$ up to n *onto* a set S, the set is called *subfinite*. That is, we need to construct a function $f : \mathbb{N} \to S$ such that $\forall_{s \in S} \exists_{i \leq n} f(i) = s$.

Note that these concepts inherently depend on the *equality* used. Given the example from Section 2: how many elements does $M := \{0, G\}$ have? We know it is 2-subfinite because there cannot be more than two elements by construction, but we cannot say it is finite, because we do now know the size of M.

Finite and subfinite sets are used for most "housekeeping" tasks in classical proofs, e.g. for mesh points in the Riemann integral, or for the roots of polynomials. In constructive proofs, such uses can be replaced with finite lists if subfiniteness is enough (the length of the list then is the boundary), or if equality is decidable.

With these kinds of sets, we can decide membership and the subset relation (if equality is decidable), and define set operations like union and intersection in finitely many steps.

6. Countably infinite sets

A countably infinite set is a set for which there is a bijection to $\mathbb{N}$.

The cardinality of these sets makes proofs more difficult. Membership and the subset relation need to be proper predicates now and cannot be considered boolean functions anymore. Likewise, set operations need careful definition to be useful in proofs.

Within proofs, it is often enough to consider *at most* infinite sets, which can be represented as series (which usually are represented as functions $f : \mathbb{N} \to A$).

7. Uncountable sets, Intervals

One step further, analysis is full of uncountable sets, which are often intervals and thus rather manageable. We consider intervals as pairs of numbers for the different endpoints and encoding which endpoints are open or closed. Membership is then easy to define, and just requires comparability of the objects. Finite set operations can be calculated.

Note that a conjectural simplification of intervals by reducing them to open intervals and singleton objects $\{x\}$ does not work: To show an element x is contained in $[0, \infty)$ requires a proof of $x \leq 0$, but to show $x \in \{0\} \cup (0, \infty)$ requires a proof of $x = 0$ *or* $x < 0$.

8. Detachable sets

The more interesting representations for sets we will consider next use the concept of *characteristic functions*.

A set $A : \sigma$ has a characteristic function $\chi_A : \sigma \to \{0, 1\}$, such that:

$$\chi_A(a) = \begin{cases} 1 & \text{if } a \in A \\ 0 & \text{if } a \notin A \end{cases}$$

Functions like these may be partial. In case they are total and finite, membership is finitely decidable, and we call the sets described *detachable sets*.

Set operations on detachable sets are always well-defined:

$$\chi_{A\cup B} = \chi_A \cdot \chi_B$$
$$\chi_{A\cap B} = \chi_A + \chi_B - \chi_A \cdot \chi_B$$

Also, we can define *complements* by subtraction:

$$\chi_{\overline{A}} = 1 - \chi_A$$

9. Complemented sets

Unfortunately, detachable sets are rare in most proofs. However, it is often possible to combine sets with an *inequality* between elements, and we can then define *complemented sets*:

A *complemented set* $A \subset X$ is a pair (A^1, A^0) of subsets of X such that $x \neq y$ whenever $x \in A^1$ and $y \in A^0$. Then there is a characteristic function $\chi_A : A^1 \cup A^0 \to \{0, 1\}$

$$\chi_A(a) = \begin{cases} 1 & \text{if } x \in A^1 \\ 0 & \text{if } x \in A^0 \end{cases}$$

Now, we also can express *complements* and *set difference* of two complemented sets, A and B:

$$\overline{A} = (A^0, A^1)$$
$$\chi_{A-B} = \chi_A \cdot (1 - \chi_B)$$

Countable unions and intersections of a sequence $(A_n)_{n=0}^{\infty}$ of complemented sets can be defined by suprema and infima of sequences of their characteristic functions.

$$\bigcup_{n=0}^{\infty} A_n = \bigvee_{n=0}^{\infty} \chi_{A_n}$$
$$\text{where } \left(\bigvee_{n=0}^{\infty} f_n\right)(s) = \sup\{f_n(s) : n \in \mathbb{N}\} \qquad \text{(partial)}$$

It is important to realize that complemented sets make use of an *affirmative* notion of inequality and not a negation of equality:

Consider the positive real numbers. By the definition above, $(\mathbb{R}_0^+, \mathbb{R}^-)$ is a complemented set. However, we could define $x \in \overline{\mathbb{R}_0^+}$ as: $x \in \mathbb{R}_0^+$ leads to a contradiction. Then there is no way of showing $x \in \mathbb{R}^-$, if $x \in \mathbb{R}_0^+$ is contradictory. But we can show that when $x \in \mathbb{R}^-$ leads to a contradiction, $x \in \mathbb{R}_0^+$ follows. (In general, we can show that $(x > y \to \mathbf{F}) \to x \leq y$.)

Thus, $\overline{\mathbb{R}^-} = \mathbb{R}_0^+$, but $\overline{\mathbb{R}_0^+} \neq \mathbb{R}^-$. But we want complementation to be stable.

We will return to complemented sets in the chapter about measure theory, where they are very important.

CHAPTER 3

Proofs about sets

1. Coquand's finite sets

Coquand proposed this inductive definition of finite sets [TC]: (The notation f^x_y here means: consider the function f changed such, that it returns y when given x as an argument.)

(`InitFin`) $$\mathrm{Fin}(\lambda_x \mathbf{false})$$

(`GenFin`) $$\forall_f \forall_n (\mathrm{Fin}\ f^n_{\mathbf{false}} \to \mathrm{Fin}\ f)$$

A formalization of this, specialized to sets of natural numbers since we require decidable equality, can be seen in listing 1.

In order to evaluate this definition, consider listing 2 as an example. It shows the supposedly simple task of proving that a set with one element is equal. The set is here represented by $\lambda_x x = 5$, that is, $\{5\}$.

THEOREM 1. *The set* $\{5\}$ *is finite.*

PROOF. To prove finiteness of $\{5\}$, we apply the induction step (`GenFin`) once, assuming 5 is in the set. Then we prove the resulting function, which now is false for 5, always returns false and apply (`InitFin`). □

Even though, or rather because, this is a very simple example, we can see the problems with this kind of definition quickly: first, we cannot apply `GenFin` without already knowing an element in the set. If we want to talk about generic sets, we are stuck here, and even with concrete representations, this requires much insight into the function. Second, we have no way of constructing elements. Is this even a valid set representation according to the requirements of Bishop/Bridges as explained on page 10? And, last, we only know we are done when we have proven the "final" function to be constant false (which may not be possible), thus we cannot estimate set size from above. Therefore, this definition doesn't help in defining subfinite sets.

```
(add-ids
 (list (list "Fin" (make-arity (py "nat=>boole"))))
 '("Fin([n] False)" "InitFin")
 '("allnc f(all n(Fin([m][if (m=n) False (f m)])
                -> Fin(f)))" "GenFin"))

(set-goal (pf "Fin ([n]n=5)"))
```

Listing 1. An inductive definition of finite sets

```
(add-var-name "f" "g" (py "nat=>boole"))
(add-global-assumption "Ext"
  (pf "all f,g(all n f n=g n -> f eqd g)"))

(set-goal (pf "Fin ([n]n=5)"))
(use "GenFin" (pt "5"))
(ng)                        ; Fin([n0][if (n0=5) False (n0=5)])
(simp (pf "[n0][if (n0=5) False (n0=5)] eqd [n0]False"))
(use "InitFin")
(use "Ext")
(ng)
(assume "n")
(cases 'auto)                   ; [if (n=5) False (n=5)]=False
  (assume "Useless")
  (use "Truth-Axiom")
  (assume "Useless")
  (use "Truth-Axiom")
(save "SingletonFiveIsFinite")
```

Listing 2. Proof that the set $\{5\}$ is finite

2. Proofs about bijections

The following theorem has been taken from an exercise in [RMFRWR1988].

THEOREM 2. *Show that a function is a bijection if and only if it is one-to-one and onto.*

The terms are defined as:

one-to-one: $a_1 = a_2$ *if* $f(a_1) = f(a_2)$.
onto: *For each* $b \in B$ *there exists* $a \in A$ *such that* $f(a) = b$
bijection: *A function* g *exists such that* $g(f(x)) = x$ *and* $f(g(y)) = y$ *for all* x, y.

PROOF. We prove both directions seperately. Listing 3 shows the first part of the proof, going from a bijection to a one-to-one and onto function. One-to-one results from injectivity of $g \circ f = id$. Onto is trivial by evaluating the inverse function.

To prove the existence of the inverse function, we need to assume an "axiom of choice" $\forall_x \exists_y (f(y) = x) \rightarrow \exists_g \forall_x (x = f(g(x)))$. Then, for any element in the domain of f we know existence of an inverse element and can construct the inverse function. □

```
(add-var-name "f" "g" (py "nat=>nat"))
(add-var-name "x" "y" (py "nat"))

;; bijection => one-to-one and onto
(set-goal (pf "all f,g((all x g(f x) = x) -> (all y f(g y) = y)
                     -> all x1, x2 (f x1 = f x2 -> x1 = x2)
                      & all y (ex x (f x = y)))"))

(assume "f" "g" "f-bij" "g-bij")
(split)

(assume "x1" "x2" "f x1=f x2")
(simp (pf "g(f x1)=g(f x2) -> x1=x2"))
(use "Truth-Axiom")
(simp (pf "f x1 = f x2"))
(use "Truth-Axiom")
(use "f x1=f x2")
(simp (pf "g(f x1) = x1"))
(simp (pf "g(f x2) = x2"))
(assume "x1=x2")
(use "x1=x2")
(use "f-bij")
(use "f-bij")

(assume "y")
(ex-intro (pt "g(y)"))
(use "g-bij")
(save "6a")
```

Listing 3. First part of theorem 2

```
(add-global-assumption "AC"
  (pf "all f(all x ex y f(y) = x -> ex g all x x = f(g x))"))

;; one-to-one and onto => bijection
(set-goal (pf "all f(all x1, x2 (f x1 = f x2 -> x1 = x2) ->
                   all y ex x f x = y ->
                   ex g (all x g(f x) = x
                       & all y f(g y) = y))"))

(assume "f" "f-one-to-one" "f-onto")

(assert (pf "ex g all x x = f(g x)"))
(use "AC")
(use "f-onto")

(assume "g-exist")
(by-assume-with "g-exist" "g" "g-inv")
(ex-intro (pt "g"))
(split)

(assume "x")
(use "Nat=Symm")
(use "f-one-to-one")
(use "g-inv")

(assume "x")
(use "Nat=Symm")
(use "g-inv")

(save "6b")
```

Listing 4. Second part of theorem 2

3. A characterization of finite sets

Significantly more work requires the following theorem:

THEOREM 3. *On a finite set every injective function is surjective.*

PROOF. We represent finite sets as functions $f : \mathbb{N} \to \mathbb{N}$ with the property $\forall_x(x < n \to f(x) < n)$ for a set with n elements.

This is how the proof works (proof idea and outline by Prof. Schwichtenberg):

Induction on the number n of elements.
Base. Trivial.
Step. (1) Assume $f(n) < n$.
Let g map $\{0, \dots, n\}$ without $f(n)$ onto $\{0, \dots, n-1\}$, mapping $i < f(n)$ to itself and $i > f(n)$ to $i - 1$. Let h be its inverse.
Let $k < n + 1$. We show that k is an f-value.
If $k = f(n)$ we are done.
Let $k \neq fn$.
Case $k < f(n)$. By IH for $g \circ f$ and k we have i such that $g(f(i)) = k$, hence $f(i) = k$ (since g is the identity below $f(n)$).
Case $f(n) \leq k$. Then $f(n) < k$ since $k \neq f(n)$. Hence $k - 1 < n$ and $k - 1 = g(k)$. By IH for $g \circ f$ and $k - 1$ we have an i such that $g(f(i)) = k - 1$, hence $g(f(i)) = g(k)$, hence $h(g(f(i))) = h(g(k))$. But $h \circ g$ is the identity, therefore $f(i) = k$.
(2) Assume $f(n) = n$. Use the IH.

You can find the full proof, formalized by me in Minlog, in Appendix A. □

CHAPTER 4

Constructive integration theory

1. The Riemann integral

In [BB1985, Chapter 2.6] define the notion of *integral* for continuous functions on compact integrals [BB1985, p.51] by a construction equivalent to the classic formulation of the Riemann integral:

$$\int_a^b f(x)\,dx = \frac{b-a}{n}\sum_{i=0}^{n-1} f(a+i\frac{b-a}{b})$$

(This is the original formulation by Riemann and not the often taught—and equivalent—Darboux integral with upper and lower sums.)

The formulation makes use of *proper intervals* $[a, b]$ and *finite partitions* thereof, which are regarded as *finite sequences* of real numbers. There's also the concept of *mesh width*, using the maximum of a finite set:

$$\text{mesh } P := \max\{a_{i+1} - a_i : 0 \leq i \leq n-1\}$$

This of course can be defined recursively by induction on n.

There is one more operation being used in the proof of [BB1985, Theorem 2.6.3], namely "Let $z_k \in [c_k, c_{k+1}]$ $(0 \leq k \leq r-1)$" with no further requirements on z_k, thus we can set, e.g. $z_k = \frac{1}{2}(c_k + c_{k+1})$.

Therefore, the simple concept of proper intervals is enough to support this elementary definition of integral.

In most proofs about the Riemann Integral, one will choose a common increment for all meshes involved and doesn't need to use sets at all, but simply multiples of the common increment.

2. The Daniell integral

Most of Chapter 6 of [BB1985] deals with an constructive definition of integral as by Daniell, which is equivalent to the Lebesgue integral. The advantage of this approach—especially in the light of thorough revision of these chapters compared to [Bis1967]—is that the concept of *measurable set* and *integrable set* can be defined in terms of their characteristic functions.

In contrast, [Bis1967] defined *Borel sets* which, all taken together, form the well-known concept of a σ-algebra. This is problematic because they require ordinals:

the new approach in [BB1985] due to [BC1972] can be based solely on Heyting Arithmetic, as by [Fef1997, p.14].

To the mathematician familiar with classic formulations of measure theory, e.g. [HB2001], the Bishop/Bridges formulation seems a bit "backward", since measurable sets are defined by measurable characteristic functions, which are defined by integrable sets, which are defined by integrable functions. However, the elegant definition of the Daniell integral compensates for this.

3. Integration spaces

An *integration space* is a triple (X, L, I) of a nonvoid-set X with an inequality (giving rise to *complemented sets*), a subset L of the strongly extensional partial functions from X to $\mathbb{R}$, and a function I of L into R subject to conditions (classically) equivalent to the Daniell axioms.

The requirements for L are:

- Functions in L must be closed under linear combinations, taking the absolute value, and being clipped below 1.
- Existence of an explicit function p in L with $I(p) = 1$.
- Given convergence of a sequence of non-negative functions $(f_n)_{n\in\mathbb{N}}$, together with $\sum_{n\in\mathbb{N}} I(f_n) < f(n)$, there is an $x \in X$ with $\sum_{n\in\mathbb{N}} f_n(x)$ convergent and $\sum_{n\in\mathbb{N}} f_n(x) < f(x)$
- Given $f \in L$, $\lim_{n\to\infty} I(f \wedge n) = I(n)$ as well as $\lim_{n\to\infty} I(|f| \wedge n^{-1}) = 0$. ($\wedge$ is the minimum of two functions.)

The following proofs use countable unions of domains of functions f_n in L in order to prove that $\bigcup_{n\in\mathbb{N}} \operatorname{dmn} f_n$ is nonvoid [BB1985, Proposition 6.1.6]; this is grounded on the fact that if it were, $\sum_{n\in\mathbb{N}} f_n(x)$ would not converge because it would not be defined.

The text goes on with an example of an integration space, namely positive measures on a locally compact space [BB1985, Theorem 6.1.10]. (A locally compact space is a set with a metric, where every bounded subset is contained in a compact subset; we can regard it as countable union of compact spheres (= closed intervals on $\mathbb{R}$) here).

There are two common examples of measures, namely the counting measure on $\mathbb{Z}$, and the Lebesgue measure on $\mathbb{R}$. The counting measure $\int f d\mu_\alpha$ is unproblematic, since it easily can be calculated on the finite domain of f. The Lebesgue measure is defined here by the Riemann integral of the test function.

4. Complete Extension of an Integral

[BB1985, Section 6.2] defines a few important terms for the rest of the discussion; namely, what an *integrable function* is (a series of function in L that converges, and the series of integrals of their absolute values converge, too)—and I is extended to be defined on them. Also, *full sets* are defined as subsets of X which are a

superset of the domain of an integrable function, and they are shown to be closed on countable intersection. [BB1985, Proposition 6.2.9] shows then, that if *any* f in $\mathcal{F}(X)$ and integrable function g are equal on a full set, f is integrable and the integrals agree, too. Thus the reasoning "what happens outside a full set may be ignored" [BB1985, p.225]. This definition of equality is also used to make the class L_1 of integrable functions a *set*.

The rest of the chapter defines a metric on L_1 and shows that (X, L_1, I) is an integration space, the *completely extended integration space*.

5. Integrable sets

In [BB1985, Section 6.3], *integrable sets* are defined via their characteristic function $\chi_A : A^1 \cup A^2 \to \{0, 1\}$ on a complemented set A; it is called *integrable* when χ_A is an integrable function. Then measure $\mu(A) := I(\chi_A)$ is defined on these sets.

A few properties of sets of measure 0 follow, and it is shown that integrable sets are closed under $\wedge$ (intersection) and $\vee$ (union) as well as $-$ (set difference). These operations are then extended to sequences of integrable sets, given that these sequences already converge in measure, that is:

$$\lim_{k\to\infty} \mu(\bigvee_{n=1}^{k} A_k) = \lambda \in \mathbb{R} \quad \Rightarrow \quad \mu(\bigvee_{n} A_k) = \lambda$$

Two more useful propositions follow, showing:

$$A_1^1 \supset A_2^1 \supset \cdots \text{ and } \lim_{n\to\infty} \mu(A_n) = \lambda \in \mathbb{R} \quad \Rightarrow \quad \mu(A_n) = \lambda$$

$$\sum_n \mu(A_n) \text{ converges} \quad \Rightarrow \quad \mu(\bigvee_n A_n) \leq \sum_n \mu(A_n)$$

These propositions are useful tools for dealing with countably infinite operations on integrable sets.

6. Measurable sets

The exposition of measure theory in [BB1985] continues with three sections of delicate content that we will skip since they are too technical and would go beyond the scope of this thesis. Essentially, they define a theory of "profiles", which are needed to construct sufficiently many integrable sets. Using profiles, we can show for example that for all except countably many t, the complemented sets $(f \geq t) := (\{x \in X \mid f(x) \geq t\})$ and $(f > t)$ are integrable and have the same measure. The interested reader is highly encouraged to read these sections in parallel with [BC1972], where the ideas come across a bit more clearly.

With a completely extended integral I on a set X, we can now define *measurable functions*, which can be approximated arbitrarily close by integrable functions.

Likewise, a *measurable set* is a complemented set whose characteristic function is measurable.

All taken together, we now have defined and constructed measurable sets starting from the Daniell integral.

7. Set representations in constructive measure theory

In this chapter, many different kinds of sets where mentioned: measureable sets, integrable sets, full sets. They all reside in X, the non-void set with inequality, that is the foundation of the integration space. Essentially, we based whole measure and integration theory on complemented sets, which "merely" are (partial) characteristic functions. The Theory of Computable Functions works well for such functions, but it is hard to gain computational value from them.

CHAPTER 5

Conclusions

We have shown that the ideas and proofs in [BB1985] can be implemented in a typed, inductive setting. Many kinds of sets arising in constructive analysis can be represented as functions or inductive data types while keeping their characteristic properties.

Since most proofs are kept very abstract, varying implementations of sets can be used while keeping the proof ideas the same.

In chapter 3 we have analyzed two different representations of finite sets, and formalized three proofs of trivial to moderate difficulty and computer-checked them.

In chapter 4 we summarized the construction of measure and integration theory as in [BB1985] and saw it is centered around the concept of complemented sets, for which characteristic functions are a natural representation.

.

Due to the limited scope of this Bachelor thesis, many topics where closer investigation should prove fruitful remain: formalization of the chapter on measure theory in [BB1985] is a major task, but can be done in steps. Another task is to come up with set representations that have more computational value, and to investigate which programs can be extracted from various proofs. One possible experiment would be to base the characteristic functions involved on concrete inductively defined data types and try to gain computational content in this way.

All together, there are many applications of these foundations in mathematics: from theoretic benefits such as proving theorems in stochastic theory and financial mathematics to practial applications such as the verification of numerical algorithms, generation of algorithms operating on stream representations of real numbers, and actual calculation of integrals.

APPENDIX A

On a finite set every injective function is surjective

This is the full proof of Theorem 3 on page 20. Due to its length, the output of `(check-and-display-proof)` can be found on the included CD-ROM.

```
(add-var-name "f" "g" "h" (py "nat=>nat"))
(add-var-name "x" "y" "i" (py "nat"))


;; Lemmata
(set-goal "all n1,n2(0<n1 -> n1<n2 -> Pred n1<Pred n2)")
(cases)
 (assume "n2" "Absurd" "nvm")
 (use "Efq")
 (use "Absurd")
(assume "n1")
 (cases)
  (assume "nvm" "Absurd")
  (use "Efq")
  (use "Absurd")
 (assume "n2")
 (ng)
 (assume "Truth" "n1<n2")
 (use "n1<n2")
(save "NatLtMonPred")

(set-goal "all n1,n2(n1<n2 -> n1<=Pred n2)")
(assume "n1")
(cases)
 (assume "Absurd")
 (use "Efq")
 (use "Absurd")
(assume "n2" "n1<Succ n2")
(use "NatLtSuccToLe")
(use "n1<Succ n2")
```

```
(save "NatLtToLePred")

(set-goal "all n1,n2(n1<Succ n2 -> n1<Succ(Succ n2))")
(cases)
 (assume "n2" "Useless")
 (use "Truth-Axiom")
(assume "n1")
(ng #t)
(assume "n2" "n1<n2")
(use "NatLtLtSuccTrans" (pt "n2"))
(use "n1<n2")
(use "NatSuccLeToLt")
(use "Truth-Axiom")
(save "n1<Succ n2 -> n1<Succ(Succ n2)")

(set-goal "all x,y(x<Succ y -> (x=y -> F) -> x<y)")
(ind)
 (cases)
  (assume "Truth" "Absurd")
  (use "Absurd")
  (use "Truth-Axiom")
 (assume "y" "Useless" "Useless2")
 (use "Truth-Axiom")
(assume "x" "IH")
(ng #t)
(cases)
 (assume "Absurd" "Useless")
 (use "Efq")
 (use "Absurd")
(assume "y")
(ng #t)
(use "IH")
(save "x<Succ y -> (x=y -> F) -> x<y")


;;; Main part of proof:
;;; On a finite set every injective function is surjective.
(set-goal (pf "all n,f(
   all x (x<n -> f x < n) ->
   all x,y (x<y -> y<n -> f x=f y -> F) ->
   all x (x<n -> ex y (y<n & f y = x)))"))
```

```
;;; Induction on the number n of elements.

(ind)

;; Base
(assume "f" "finite" "f-inj" "x" "Absurd")
(use "Efq")
(use "Absurd")

;; Step
(assume "n" "IH" "f" "finite" "f-inj")
(assume "x" "x<Succ n")

;;; (1) Assume f n<n.

(use "NatLtSuccCases" (pt "n") (pt "f n"))
(use "finite")
(use "Truth-Axiom")

(assume "f n<n")

;;; Let g map 0...,n without f n onto 0...n-1, mapping i<f n to itself
;;; and i>f n to i-1. Let h be its inverse.

(assert (pf "ex g all i g i = [if (i<f n) i (i--1)]"))
 (ex-intro (pt "[i][if (i<f n) i (i--1)]"))
 (assume "i")
 (use "Truth-Axiom")
 (assume "g-exp-hyp")
(by-assume "g-exp-hyp" "g" "g-prop")

(assert (pf "ex h all i h i = [if (i<f n) i (i+1)]"))
 (ex-intro (pt "[i][if (i<f n) i (i+1)]"))
 (assume "i")
 (use "Truth-Axiom")
 (assume "h-exp-hyp")
(by-assume "h-exp-hyp" "h" "h-prop")

;; gof correct
(assert (pf "all x (x<n -> g(f x) < n)"))
 (cases (pt "n"))
  (assume "n=0" "y" "Absurd")
```

```
  (use "Efq")
  (use "Absurd")
 (assume "n0" "n=Succ n0" "y" "y<Succ n0")
 (cases (pt "f y=0"))
  (assume "f y=0")
  (simp "f y=0")
  (simp "g-prop")
  (ng #t)
  (use "Truth-Axiom")
 (assume "f y=0 -> F")
 (simp "g-prop")
 (cases (pt "f y<f n"))
  (assume "f y<f n")
  (ng #t)
  (use "NatLtTrans" (pt "f n"))
  (use "f y<f n")
  (simp (pf "Succ n0=n"))
  (use "f n<n")
  (use "Nat=Symm")
  (use "n=Succ n0")
 (assume "f y<f n -> F")
 (ng #t)
 (assert (pf "Succ n0=Pred(Succ n)"))
 (ng #t)
 (use "Nat=Symm")
 (use "n=Succ n0")
 (assume "Succ n0=Pred(Succ n)")
 (simp "Succ n0=Pred(Succ n)")
 (use "NatLtMonPred")
 (cases (pt "f y"))
  (assume "f y=0")
  (use "Efq")
  (use "f y=0 -> F")
  (use "f y=0")
 (assume "n2" "Useless")
 (use "Truth-Axiom")
 (use "finite")
 (simp (pf "n=Succ n0"))
 (use "n1<Succ n2 -> n1<Succ(Succ n2)")
 (use "y<Succ n0")
 (use "n=Succ n0")
(assume "gof-correct")
```

```
;; hog id except for i=f n
(assert (pf "all x ((x = f n -> F) -> h(g(x))=x)"))
 (assume "y" "y=f n -> F")
 (simp "g-prop")
 (cases (pt "y<f n"))
  (assume "y<f n")
  (ng #t)
  (simp "h-prop")
  (simp "y<f n")
  (use "Truth-Axiom")
 (assume "y<f n -> F")
 (ng #t)
 (assert (pf "f n<y"))
  (use "NatNotLeToLt")
  (assume "y<=f n")
  (use "NatLeCases" (pt "f n") (pt "y"))
  (use "y<=f n")
  (use "y<f n -> F")
  (use "y=f n -> F")
 (assume "f n<y")
 (simp "h-prop")
 (assert (pf "Pred y<f n -> F"))
  (assume "Pred y<f n")
  (assert (pf "Pred y<Pred y"))
   (use "NatLtLeTrans" (pt "f n"))
   (use "Pred y<f n")
   (use "NatLtToLePred")
   (use "f n<y")
  (assume "Absurd")
  (use "Absurd")
 (assume "Pred y<f n -> F")
 (simp "Pred y<f n -> F")
 (ng #t)
 (cases (pt "y"))
  (assume "y=0")
  (use "Efq")
  (simphyp-with-to "f n<y" "y=0" "Absurd")
  (use "Absurd")
 (assume "y0" "Useless")
 (use "Truth-Axiom")
(assume "hog-id")
```

```
;; gof injective
(assert (pf "all x,y(x<y -> y<n -> g(f x) = g(f y) -> F)"))
 (assume "x2" "y2" "x2<y2" "y2<n")
 (assert (pf "h(g(f y2))=f y2"))
  (use "hog-id")
  (use "f-inj")
  (use "y2<n")
  (use "Truth-Axiom")
 (assume "h(g(f y2))=f y2")
 (assume "g(f x2)=g(f y2)")
 (use "f-inj" (pt "x2") (pt "y2"))
 (use "x2<y2")
 (use "NatLtTrans" (pt "n"))
 (use "y2<n")
 (use "Truth-Axiom")
 (simp "<-" "h(g(f y2))=f y2")
 (assert (pf "h(g(f x2))=f x2"))
  (use "hog-id")
  (use "f-inj")
  (use "NatLtTrans" (pt "y2"))
  (use "x2<y2")
  (use "y2<n")
  (use "Truth-Axiom")
 (assume "h(g(f x2))=f x2")
 (simp "<-" "h(g(f x2))=f x2")
 (assert (pf "all x,y(x=y -> h x=h y)"))
  (assume "x3" "y3" "x3=y3")
  (simp "x3=y3")
  (use "Truth-Axiom")
 (assume "x=y -> h x=h y")
 (simp "x=y -> h x=h y")
 (use "Truth-Axiom")
 (use "g(f x2)=g(f y2)")
(assume "gof-inj")

;;; Let x<n+1. We show that x is an f-value.
;;; If x=f n we are done.

(cases (pt "x=f n"))
 (assume "x=f n")
 (ex-intro (pt "n"))
```

```
 (split)
 (use "Truth-Axiom")
 (use "Nat=Symm")
 (use "x=f n")

;;; Let x != f n.

(assume "x=f n -> F")
(cases (pt "x<f n"))
(assume "x<f n")

;;; Case x<f n.
;;; By IH for gof and x we have i such that g(f i)=x,
;;; hence f i=x (since g is the identity below f n).

(assert (pf "ex y(y<n & ([y]g(f y))y=x)"))
 (use "IH")
 (assume "x2" "x2<n")
 (ng #t)
 (use "gof-correct")
 (use "x2<n")
 (ng #t)
 (use "gof-inj")
 (use "NatLtTrans" (pt "f n"))
 (use "x<f n")
 (use "f n<n")
 (ng #t)
(assume "ex-g(f y)=x")

(by-assume "ex-g(f y)=x" "y" "y-prop")
(inst-with-to "y-prop" 'left "y<n")
(inst-with-to "y-prop" 'right "g(f y)=x")
(drop "y-prop")
(ex-intro (pt "y"))
(split)
(use "NatLtTrans" (pt "n"))
(use "y<n")
(use "Truth-Axiom")

(assert (pf "h(g(f y))=f y"))
 (use "hog-id")
 (use "f-inj")
```

```
 (use "y<n")
 (use "Truth-Axiom")
(assume "h(g(f y))=f y")
(simp "<-" "h(g(f y))=f y")

(assert (pf "all x,y(x=y -> h x=h y)"))
 (assume "x3" "y3" "x3=y3")
 (simp "x3=y3")
 (use "Truth-Axiom")
(assume "x=y -> h x=h y")
(simp (pf "x=h(x)"))

(use "x=y -> h x=h y")
(use "g(f y)=x")

(simp "h-prop")
(simp "x<f n")
(use "Truth-Axiom")

;;; Case f n<=x.
;;; Then f n<x since x != f n. Hence x-1<n and x-1=g x.

(assume "x<f n -> F")
(assert (pf "f n<=x"))
 (use "NatNotLtToLe")
 (use "x<f n -> F")
(assume "f n<=x")
(use "NatLeCases" (pt "x") (pt "f n"))
(use "f n<=x")
(assume "f n<x")
(assert (pf "Pred x<n"))
 (simp (pf "n=Pred(Succ n)"))
 (use "NatLtMonPred")
 (use "NatLeLtTrans" (pt "f n"))
 (use "Truth-Axiom")
 (use "f n<x")
 (use "x<Succ n")
 (use "Truth-Axiom")
(assume "Pred x<n")
(assert (pf "g x=Pred x"))
 (simp "g-prop")
 (simp "x<f n -> F")
```

```
 (use "Truth-Axiom")
(assume "g x=Pred x")

;;; By IH for gof and x-1 we have an i such that g(f i)=x-1,
;;; hence g(f i)=g x, hence h(g(f i))=h(g x).
;;; But hog is the identity, therefore f i=x.

(assert (pf "ex y(y<n & ([y]g(f y))y=Pred x)"))
 (use "IH")
 (assume "x2" "x2<n")
 (ng #t)
 (use "gof-correct")
 (use "x2<n")
 (ng #t)
 (assume "x2" "y2" "x2<y2")
 (use "gof-inj")
 (use "x2<y2")
 (use "Pred x<n")
 (ng #t)
(assume "ex-g(f y)=Pred x")

(by-assume "ex-g(f y)=Pred x" "y" "y-prop")
(inst-with-to "y-prop" 'left "y<n")
(inst-with-to "y-prop" 'right "g(f y)=Pred x")
(drop "y-prop")
(ex-intro (pt "y"))
(split)
(use "NatLtTrans" (pt "n"))
(use "y<n")
(use "Truth-Axiom")
(assert (pf "g(f y)=g x"))
 (simp "g(f y)=Pred x")
 (simp "g-prop")
 (simp "x<f n -> F")
 (use "Truth-Axiom")
(assume "g(f y)=g x")
(assert (pf "h(g(f y))=h(g x)"))
 (simp "g(f y)=g x")
 (use "Truth-Axiom")
 (simp "hog-id" (pt "f y") (pt "x"))
(assume "f y=h(g x)")
(use "Nat=Trans" (pt "h(g x)"))
```

```
(use "f y=h(g x)")
(use "hog-id")
(assume "x=f n")
(use "x=f n -> F")
(use "x=f n")
(use "f-inj")
(use "y<n")
(use "Truth-Axiom")

(assume "f n=x")
(ex-intro (pt "n"))
(split)
(use "Truth-Axiom")
(use "f n=x")

;;; (2) Assume f n=n. Use the IH.

(assume "f n=n")
(use "NatLtSuccCases" (pt "n") (pt "x"))
(use "x<Succ n")
(assume "x<n")
(assert (pf "ex y(y<n & f y=x)"))
 (use "IH")
 (assume "x2" "x2<n")
 (assert (pf "f x2<Succ n"))
  (use "finite")
  (use "NatLtTrans" (pt "n"))
  (use "x2<n")
  (use "Truth-Axiom")
 (assume "f x2<Succ n")
 (assert (pf "f x2=n -> F"))
  (simp "<-" "f n=n")
  (use "f-inj")
  (use "x2<n")
  (use "Truth-Axiom")
 (assume "f x2=n -> F")
 (use "x<Succ y -> (x=y -> F) -> x<y")
 (use "f x2<Succ n")
 (use "f x2=n -> F")
 (assume "x2" "y2" "x2<y2" "y2<n")
 (use "f-inj")
 (use "x2<y2")
```

```
 (use "NatLtTrans" (pt "n"))
 (use "y2<n")
 (use "Truth-Axiom")
 (use "x<n")
(assume "ex-f y=x")

(by-assume "ex-f y=x" "y" "y-prop")
(ex-intro (pt "y"))
(inst-with-to "y-prop" 'left "y<n")
(inst-with-to "y-prop" 'right "f y=x")
(drop "y-prop")
(split)
(use "NatLtTrans" (pt "n"))
(use "y<n")
(use "Truth-Axiom")
(use "f y=x")

(assume "x=n")
(ex-intro (pt "x"))
(split)
(use "x<Succ n")
(simp "x=n")
(use "f n=n")

(save "fin-inj-surj")
```

Bibliography

[BB1985] Errett Bishop and Douglas Bridges, *Constructive Analysis*, Springer, Berlin, 1985.

[BC1972] Errett Bishop and Henrey Cheng, *Constructive measure theory*, Memoirs of the American Mathematical Society, American Mathematical Society, 1972.

[Bis1967] Errett Bishop, *Foundations of Constructive Analysis*, McGraw-Hill, New York, 1967.

[Fre1893] Gottlob Frege, *Grundgesetze der Arithmetik, begriffsschriftlich abgeleitet*, Vol. 1, Jena, 1893.

[HB2001] Heinz Bauer, *Measure and Integration Theory*, de Gruyter Studies in Mathematics, Walter de Gruyter, Berlin, 2001.

[HS2007] Helmut Schwichtenberg, *Elemente der Zahlentheorie, Aufbau des Zahlensystems.* Kapitel 6. Reele Zahlen, Lecture Notes, 2007/2008.

[HS2010] ———, *A Theory of Computable Functionals*, 2010. `http://www.minlog-system.de`.

[HS2011] ———, *Minlog reference manual*, 2011. `http://www.minlog-system.de`.

[Fef1979] Solomon Feferman, *Constructive Theories of Functions and Classes*, Logic Colloquium '78 (1979), 159–224.

[Fef1997] ———, *Relationships between constructive, predicative, and classical systems of analysis.* Proof Theory: History and Philosophical Significance, Lecture Notes, 1997.

[RMFRWR1988] Ray Mines, Fred Richman, and Wim Ruitenburg, *A course in constructive algebra*, Universitext, Springer, Berlin, 1988.

[Rus1902] Bertrand Russell, *Letter to Frege* (1902); reprinted in Jean van Heijenoort, *From Frege to Gödel*, Harvard University Press, Cambridge, Mass. 1967.

[TC] Thierry Coquand. *Informal discussion with Helmut Schwichtenberg.*

Hiermit versichere ich, dass die zum heutigen Tag an der Fakultät für Mathematik, Informatik und Statistik eingereichte Bachelorarbeit zum Thema "The Concept of Set in Constructive Analysis" selbstständig verfasst wurde und ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Datum, Unterschrift: _____