Formalizing Integration Theory, with an Application to Probabilistic Algorithms

Stefan Richter
LuFG Theoretische Informatik
RWTH Aachen
Ahornstraße 55
52056 Aachen
FRG
richter@informatik.rwth-aachen.de

October 11, 2017

Contents

Chapter 1

Prologue

> Verifying more examples of probabilistic algorithms will inevitably necessitate more formalization; in particular we already can see that a theory of expectation will be required to prove the correctness of probabilistic quicksort. If we can continue our policy of formalizing standard theorems of mathematics to aid verifications, then this will provide long-term benefits to many users of the HOL theorem prover.

This quote from the Future Work section of Joe Hurd's PhD thesis "Formal Verification of Probabilistic Algorithms" ([6] p. 131) served as a starting point for the following work. A theory of expectation is nothing but a theory of integration in its probability theoretic underpinnings. And though the proof of correctness for probabilistic quicksort might not need integration, an average runtime analysis certainly will.

As indicated in the very beginning, integration is needed in some way to talk about expectation in probability. The notion that is addressed here is a kind of average value of a random variable with respect to a (probability) measure. The concept of a *measure* lies at the heart of Lebesgue integration. A measure is simply a function satisfying a few sanity properties that maps sets to real numbers. Because the definition does not employ such concrete entities as intervals, it generalizes easily to functions that do not have the real numbers as their domain. In particular, the notion of measure is very natural in the field of probability theory, where a probability measure — nothing but a measure P with $P(\Omega) = 1$ — gives the probability of an event — a measurable subset of Ω.

This Ω might, for example, be the set of all infinite sequences of boolean values, as in Hurd's thesis[6]; our integral is then just a tool that extends this work in the sense depicted at the very beginning of this introduction.

We begin by declaring some preliminary notions, including elementary measure theory and monotone convergence. This leads into measurable real-

valued functions, also known as random variables. A sufficient body of functions is shown to belong to this class. The central chapter is about integration proper. We build the integral for increasingly complex functions and prove essential properties, discovering the connection with measurability in the end.

Chapter 2

Measurable Functions

In this chapter, the focus is on the kind of functions to be integrated. As we will see later on, measurability is a good characterization for these functions. Moreover, the language of measure theory as well as the notion of monotone convergence is used frequently in the definition of the integral. So we begin by formalizing these necessary tools.

2.1 Preliminaries

2.1.1 Sigma algebras

theory *Sigma-Algebra* **imports** *Main* **begin**

The **theory** command commences a formal document and enumerates the theories it depends on. With the *Main* theory, a standard selection of useful HOL theories excluding the real numbers is loaded. This theory includes and builds upon a tiny theory of the same name by Markus Wenzel. This theory as well as *Measure* in 2.1.3 is heavily influenced by Joe Hurd's thesis [6] and has been designed to keep the terminology as consistent as possible with that work.

Sigma algebras are an elementary concept in measure theory. To measure — that is to integrate — functions, we first have to measure sets. Unfortunately, when dealing with a large universe, it is often not possible to consistently assign a measure to every subset. Therefore it is necessary to define the set of measurable subsets of the universe. A sigma algebra is such a set that has three very natural and desirable properties.

definition
sigma-algebra:: $'a\ set\ set \Rightarrow bool$ **where**
$sigma\text{-}algebra\ A \longleftrightarrow$
$\{\} \in A \wedge (\forall a.\ a \in A \longrightarrow -a \in A) \wedge$
$(\forall a.\ (\forall\ i{::}nat.\ a\ i \in A) \longrightarrow (\bigcup i.\ a\ i) \in A)$

The **definition** command defines new constants, which are just named functions in HOL. Mind that the third condition expresses the fact that the union of countably many sets in A is again a set in A without explicitly defining the notion of countability.

Sigma algebras can naturally be created as the closure of any set of sets with regard to the properties just postulated. Markus Wenzel wrote the following inductive definition of the *sigma* operator.

inductive-set
  sigma :: *'a set set* ⇒ *'a set set*
  for *A* :: *'a set set*
  where
    basic: *a* ∈ *A* ⟹ *a* ∈ *sigma A*
  | *empty*: {} ∈ *sigma A*
  | *complement*: *a* ∈ *sigma A* ⟹ − *a* ∈ *sigma A*
  | *Union*: (⋀*i*::*nat*. *a i* ∈ *sigma A*) ⟹ (⋃ *i*. *a i*) ∈ *sigma A*

He also proved the following basic facts. The easy proofs are omitted.

theorem *sigma-UNIV*: *UNIV* ∈ *sigma A*

theorem *sigma-Inter*:
  (⋀*i*::*nat*. *a i* ∈ *sigma A*) ⟹ (⋂ *i*. *a i*) ∈ *sigma A*

It is trivial to show the connection between our first definitions. We use the opportunity to introduce the proof syntax.

theorem assumes *sa*: *sigma-algebra A*
  — Named premises are introduced like this.

  shows *sigma-sigma-algebra*: *sigma A* = *A*
proof

The **proof** command alone invokes a single standard rule to simplify the goal. Here the following two subgoals emerge.

  show *A* ⊆ *sigma A*
    — The **show** command starts the proof of a subgoal.

    by (*auto simp add*: *sigma.basic*)

This is easy enough to be solved by an automatic step, indicated by the keyword **by**. The method **auto** is stated in parentheses, with attributes to it following. In this case, the first introduction rule for the **sigma** operator is given as an extra simplification rule.

  show *sigma A* ⊆ *A*
  proof

Because this goal is not quite as trivial, another proof is invoked, delimiting a block as in a programming language.

    fix *x*

— A new named variable is introduced.

assume $x \in sigma\ A$

An assumption is made that must be justified by the current proof context. In this case the corresponding fact had been generated by a rule automatically invoked by the inner **proof** command.

from *this sa* **show** $x \in A$

Named facts can explicitly be given to the proof methods using **from**. A special name is *this*, which denotes current facts generated by the last command. Usually **from** *this sa* — remember that *sa* is an assumption from above — is abbreviated to **with** *sa*, but in this case the order of facts is relevant for the following method and **with** would have put the current facts last.

by (*induct rule*: *sigma.induct*) (*auto simp add*: *sigma-algebra-def*)

Two methods may be carried out at **by**. The first one applies induction here via the canonical rule generated by the inductive definition above, while the latter solves the resulting subgoals by an automatic step involving simplification.

qed
qed

These two steps finish their respective proofs, checking that all subgoals have been proven.

To end this theory we prove a special case of the *sigma-Inter* theorem above. It seems trivial that the fact holds for two sets as well as for countably many. We get a first taste of the cost of formal reasoning here, however. The idea must be made precise by exhibiting a concrete sequence of sets.

primrec *trivial-series*:: $'a\ set \Rightarrow 'a\ set \Rightarrow (nat \Rightarrow 'a\ set)$
where
trivial-series a b 0 = *a*
| *trivial-series a b* (*Suc n*) = *b*

Using **primrec**, primitive recursive functions over inductively defined data types — the natural numbers in this case — may be constructed.

theorem assumes *s*: *sigma-algebra A* **and** *a*: $a \in A$ **and** *b*: $b \in A$
shows *sigma-algebra-inter*: $a \cap b \in A$
proof −
— This form of **proof** foregoes the application of a rule.

have $a \cap b = (\bigcap i{::}nat.\ trivial\text{-}series\ a\ b\ i)$

Intermediate facts that do not solve any subgoals yet are established this way.

proof (*rule set-eqI*)

The **proof** command may also take one explicit method as an argument like the single rule application in this instance.

fix x

{
fix i
assume $x \in a \cap b$
hence $x \in$ *trivial-series a b i* **by** (*cases i*) *auto*
— This is just an abbreviation for **"from this have"**.
}

Curly braces can be used to explicitly delimit blocks. In conjunction with **fix**, universal quantification over the fixed variable i is achieved for the last statement in the block, which is exported to the enclosing block.

hence $x \in a \cap b \Longrightarrow \forall i.\ x \in$ *trivial-series a b i*
by *fast*
also

The statement **also** introduces calculational reasoning. This basically amounts to collecting facts. With **also**, the current fact is added to a special list of theorems called the calculation and an automatically selected transitivity rule is additionally applied from the second collected fact on.

{ assume $\bigwedge i.\ x \in$ *trivial-series a b i*
hence $x \in$ *trivial-series a b 0* **and** $x \in$ *trivial-series a b 1*
by *this+*
hence $x \in a \cap b$
by *simp*
}
hence $\forall i.\ x \in$ *trivial-series a b i* $\Longrightarrow x \in a \cap b$
by *blast*

ultimately have $x \in a \cap b = (\forall i{::}nat.\ x \in$ *trivial-series a b i*$)$ **..**

The accumulated calculational facts including the current one are exposed to the next statement by **ultimately** and the calculation list is then erased. The two dots after the statement here indicate proof by a single automatically selected rule.

also have $\ldots = (x \in (\bigcap i{::}nat.$ *trivial-series a b i*$))$
by *simp*
finally show $x \in a \cap b = (x \in (\bigcap i{::}nat.$ *trivial-series a b i*$))$ **.**

The **finally** directive behaves like **ultimately** with the addition of a further transitivity rule application. A single dot stands for proof by assumption.

qed

moreover have $(\bigcap i{::}nat.$ *trivial-series a b i*$) \in A$
proof −
{ fix i
from *a b* **have** *trivial-series a b i* $\in A$
by (*cases i*) *auto*
}
hence $\bigwedge i.$ *trivial-series a b i* $\in$ *sigma A*

by (*simp only*: *sigma.basic*)
hence ($\bigcap$ *i*::*nat*. *trivial-series a b i*) ∈ *sigma A*
by (*simp only*: *sigma-Inter*)
with *s* **show** *?thesis*
by (*simp only*: *sigma-sigma-algebra*)
qed

ultimately show *?thesis* **by** *simp*
qed

Of course, a like theorem holds for union instead of intersection. But as we will not need it in what follows, the theory is finished with the following easy properties instead. Note that the former is a kind of generalization of the last result and could be used to shorten its proof. Unfortunately, this one was needed — and therefore found — only late in the development.

theorem *sigma-INTER*:
assumes *a*:($\bigwedge$*i*::*nat*. *i* ∈ *S* ⟹ *a i* ∈ *sigma A*)
shows ($\bigcap$ *i*∈*S*. *a i*) ∈ *sigma A*

lemma assumes *s*: *sigma-algebra a* **shows** *sigma-algebra-UNIV*: *UNIV* ∈ *a*

end

2.1.2 Monotone Convergence

theory *MonConv*
imports *Complex-Main*
begin

A sensible requirement for an integral operator is that it be "well-behaved" with respect to limit functions. To become just a little more precise, it is expected that the limit operator may be interchanged with the integral operator under conditions that are as weak as possible. To this end, the notion of monotone convergence is introduced and later applied in the definition of the integral.

In fact, we distinguish three types of monotone convergence here: There are converging sequences of real numbers, real functions and sets. Monotone convergence could even be defined more generally for any type in the axiomatic type class[1] *ord* of ordered types like this.

mon-conv u f ≡ (∀ *n*. *u n* ≤ *u* (*Suc n*)) ∧ *SUPREMUM UNIV u* = *f*

However, this employs the general concept of a least upper bound. For the special types we have in mind, the more specific limit — respective union — operators are available, combined with many theorems about their properties. For the type of real- (or rather ordered-) valued functions, the less-or-equal relation is defined pointwise.

[1] For the concept of axiomatic type classes, see [7, 9]

$(f \leq g) = (\forall x.\ f\ x \leq g\ x)$

Now the foundations are laid for the definition of monotone convergence. To express the similarity of the different types of convergence, a single overloaded operator is used.

consts
mon-conv:: $(nat \Rightarrow 'a) \Rightarrow 'a$::*ord* $\Rightarrow$ *bool* (-↑- [*60*,*61*] *60*)
overloading
mon-conv-real $\equiv$ *mon-conv* :: - $\Rightarrow$ *real* $\Rightarrow$ *bool*
mon-conv-real-fun $\equiv$ *mon-conv* :: - $\Rightarrow$ ($'a \Rightarrow$ *real*) $\Rightarrow$ *bool*
mon-conv-set $\equiv$ *mon-conv* :: - $\Rightarrow$ $'a$ *set* $\Rightarrow$ *bool*
begin

definition $x\uparrow(y::real) \equiv (\forall n.\ x\ n \leq x\ (Suc\ n)) \wedge x \longrightarrow y$
definition $u\uparrow(f::'a \Rightarrow real) \equiv (\forall n.\ u\ n \leq u\ (Suc\ n)) \wedge\ (\forall w.\ (\lambda n.\ u\ n\ w) \longrightarrow f\ w)$
definition $A\uparrow(B::'a\ set) \equiv (\forall n.\ A\ n \leq A\ (Suc\ n)) \wedge B = (\bigcup n.\ A\ n)$

end

theorem *realfun-mon-conv-iff*: $(u\uparrow f) = (\forall w.\ (\lambda n.\ u\ n\ w)\uparrow((f\ w)::real))$
by (*auto simp add*: *mon-conv-real-def mon-conv-real-fun-def le-fun-def*)

The long arrow signifies convergence of real sequences as defined in the theory *SEQ* [5]. Monotone convergence for real functions is simply pointwise monotone convergence.

Quite a few properties of these definitions will be necessary later, and they are listed now, giving only few select proofs.

lemma assumes *mon-conv*: $x\uparrow(y::real)$
shows *mon-conv-mon*: $(x\ i) \leq (x\ (m+i))$

lemma *limseq-shift-iff*: $(\lambda m.\ x\ (m+i)) \longrightarrow y = x \longrightarrow y$

theorem assumes *mon-conv*: $x\uparrow(y::real)$
shows *real-mon-conv-le*: $x\ i \leq y$
proof −
from *mon-conv* **have** $(\lambda m.\ x\ (m+i)) \longrightarrow y$
by (*simp add*: *mon-conv-real-def limseq-shift-iff*)
also from *mon-conv* **have** $\forall m{\geq}0.\ x\ i \leq x\ (m+i)$ **by** (*simp add*: *mon-conv-mon*)
ultimately show *?thesis* **by** (*rule LIMSEQ-le-const*[*OF - exI*[**where** *x*=*0*]])
qed

theorem assumes *mon-conv*: $x\uparrow(y::('a \Rightarrow real))$
shows *realfun-mon-conv-le*: $x\ i \leq y$
proof −
{fix *w*
from *mon-conv* **have** $(\lambda i.\ x\ i\ w)\uparrow(y\ w)$
by (*simp add*: *realfun-mon-conv-iff*)

```
    hence x i w ≤ y w
      by (rule real-mon-conv-le)
  }
  thus ?thesis by (simp add: le-fun-def)
qed

lemma assumes mon-conv: x↑(y::real)
  and less: z < y
  shows real-mon-conv-outgrow: ∃ n. ∀ m. n ≤ m ⟶ z < x m
proof −
  from less have less': 0 < y−z
    by simp
  have ∃ n.∀ m. n ≤ m ⟶ |x m − y| < y − z
  proof −
    from mon-conv have aux: ⋀r. r > 0 ⟹ ∃ n. ∀ m. n ≤ m ⟶ |x m − y| < r
    unfolding mon-conv-real-def lim-sequentially dist-real-def by auto
    with less' show ∃ n. ∀ m. n ≤ m ⟶ |x m − y| < y − z by auto
  qed
  also
  { fix m
    from mon-conv have x m ≤ y
      by (rule real-mon-conv-le)
    hence |x m − y| = y − x m
      by arith
    also assume |x m − y| < y − z
    ultimately have z < x m
      by arith
  }
  ultimately show ?thesis
    by blast
qed


theorem real-mon-conv-times:
  assumes xy: x↑(y::real) and nn: 0≤z
  shows (λm. z∗x m)↑(z∗y)

theorem realfun-mon-conv-times:
  assumes xy: x↑(y::'a⇒real) and nn: 0≤z
  shows (λm w. z∗x m w)↑(λw. z∗y w)

theorem real-mon-conv-add:
  assumes xy: x↑(y::real) and ab: a↑(b::real)
  shows (λm. x m + a m)↑(y + b)
theorem realfun-mon-conv-add:
  assumes xy: x↑(y::'a⇒real) and ab: a↑(b::'a ⇒ real)
  shows (λm w. x m w + a m w)↑(λw. y w + b w)

theorem real-mon-conv-bound:
```

assumes *mon*: $\bigwedge n.\ c\ n \leq c\ (Suc\ n)$
and *bound*: $\bigwedge n.\ c\ n \leq (x::real)$
shows $\exists\, l.\ c{\uparrow}l \wedge l{\leq}x$
proof −
from *incseq-convergent*[*of c x*] *mon bound*
obtain l **where** $c \longrightarrow l\ \forall\, i.\ c\ i \leq l$
by (*auto simp*: *incseq-Suc-iff*)
moreover — This is like **also** but lacks the transitivity step.
with *bound* **have** $l \leq x$
by (*intro LIMSEQ-le-const2*) *auto*
ultimately show *?thesis*
by (*auto simp*: *mon-conv-real-def mon*)
qed

theorem *real-mon-conv-dom*:
assumes *xy*: $x{\uparrow}(y::real)$ **and** *mon*: $\bigwedge n.\ c\ n \leq c\ (Suc\ n)$
and *dom*: $c \leq x$
shows $\exists\, l.\ c{\uparrow}l \wedge l{\leq}y$
proof −
from *dom* **have** $\bigwedge n.\ c\ n \leq x\ n$ **by** (*simp add*: *le-fun-def*)
also from *xy* **have** $\bigwedge n.\ x\ n \leq y$ **by** (*simp add*: *real-mon-conv-le*)
also note *mon*
ultimately show *?thesis* **by** (*simp add*: *real-mon-conv-bound*)
qed

theorem *realfun-mon-conv-bound*:
assumes *mon*: $\bigwedge n.\ c\ n \leq c\ (Suc\ n)$
and *bound*: $\bigwedge n.\ c\ n \leq (x::'a \Rightarrow real)$
shows $\exists\, l.\ c{\uparrow}l \wedge l{\leq}x$

This brings the theory to an end. Notice how the definition of the limit of a real sequence is visible in the proof to *real-mon-conv-outgrow*, a lemma that will be used for a monotonicity proof of the integral of simple functions later on.

end

2.1.3 Measure spaces

theory *Measure*
imports *Sigma-Algebra MonConv*
begin

Now we are already set for the central concept of measure. The following definitions are translated as faithfully as possible from those in Joe Hurd's thesis [6].

definition
measurable:: $'a\ set\ set \Rightarrow 'b\ set\ set \Rightarrow ('a \Rightarrow 'b)\ set$ **where**
$measurable\ F\ G = \{f.\ \forall\, g{\in}G.\ f\ -`\ g \in F\}$

So a function is called F-G-measurable if and only if the inverse image of any set in G is in F. F and G are usually the sets of measurable sets, the first component of a measure space[2].

definition
measurable-sets:: $('a\ set\ set * ('a\ set \Rightarrow real)) \Rightarrow 'a\ set\ set$ **where**
$measurable\text{-}sets = fst$

definition
measure:: $('a\ set\ set * ('a\ set \Rightarrow real)) \Rightarrow ('a\ set \Rightarrow real)$ **where**
$measure = snd$

The other component is the measure itself. It is a function that assigns a nonnegative real number to every measurable set and has the property of being countably additive for disjoint sets.

definition
positive:: $('a\ set\ set * ('a\ set \Rightarrow real)) \Rightarrow bool$ **where**
$positive\ M \longleftrightarrow measure\ M\ \{\} = 0 \wedge$
$(\forall\, A.\ A{\in}\ measurable\text{-}sets\ M \longrightarrow 0 \leq measure\ M\ A)$

[2] In standard mathematical notation, the universe is first in a measure space triple, but in our definitions, following Joe Hurd, it is always the whole type universe and therefore omitted.

definition
countably-additive:: (*'a set set* ∗ (*'a set* => *real*)) => *bool* **where**
countably-additive M ⟷ (∀ *f*::(*nat* => *'a set*). *range f* ⊆ *measurable-sets M*
∧ (∀ *m n*. *m* ≠ *n* ⟶ *f m* ∩ *f n* = {}) ∧ (⋃ *i*. *f i*) ∈ *measurable-sets M*
⟶ (λ*n*. *measure M* (*f n*)) *sums* *measure M* (⋃ *i*. *f i*))

This last property deserves some comments. The conclusion is usually — also in the aforementioned source — phrased as

measure M (⋃ *i*. *f i*) = (∑ *n*. *measure M* (*f n*)).

In our formal setting this is unsatisfactory, because the sum operator[3], like any HOL function, is total, although a series obviously need not converge. It is defined using the ε operator, and its behavior is unspecified in the diverging case. Hence, the above assertion would give no information about the convergence of the series.

Furthermore, the definition contains redundancy. Assuming that the countable union of sets is measurable is unnecessary when the measurable sets form a sigma algebra, which is postulated in the final definition[4].

definition
measure-space:: (*'a set set* ∗ (*'a set* ⇒ *real*)) ⇒ *bool* **where**
measure-space M ⟷ *sigma-algebra* (*measurable-sets M*) ∧
positive M ∧ *countably-additive M*

Note that our definition is restricted to finite measure spaces — that is, *measure M UNIV* < ∞ — since the measure must be a real number for any measurable set. In probability, this is naturally the case.

Two important theorems close this section. Both appear in Hurd's work as well, but are shown anyway, owing to their central role in measure theory. The first one is a mighty tool for proving measurability. It states that for a function mapping one sigma algebra into another, it is sufficient to be measurable regarding only a generator of the target sigma algebra. Formalizing the interesting proof out of Bauer's textbook [1] is relatively straightforward using rule induction.

theorem assumes *sig*: *sigma-algebra a* **and** *meas*: *f* ∈ *measurable a b* **shows**
measurable-lift: *f* ∈ *measurable a* (*sigma b*)
proof −
def *Q* ≡ {*q*. *f* −‘ *q* ∈ *a*}
with *meas* **have** *1*: *b* ⊆ *Q* **by** (*auto simp add*: *measurable-def*)

{ fix *x* **assume** *x*∈*sigma b*
hence *x*∈*Q*
proof (*induct rule*: *sigma.induct*)

[3] Which is merely syntactic sugar for the *suminf* functional from the *Series* theory [5].
[4] Joe Hurd inherited this practice from a very influential probability textbook [10]

```
    case basic
    from 1 show  ⋀a. a ∈ b ⟹ a ∈ Q ..
  next
    case empty
    from sig have {}∈a
      by (simp only: sigma-algebra-def)
    thus {} ∈ Q
      by (simp add: Q-def)
  next
    case complement
    fix r assume r ∈ Q
    then obtain r1 where im: r1 = f -` r and a: r1 ∈ a
      by (simp add: Q-def)
    with sig have -r1 ∈ a
      by (simp only: sigma-algebra-def)
    with im Q-def show -r ∈ Q
      by (simp add: vimage-Compl)
  next
    case Union
    fix r assume ⋀i::nat. r i ∈ Q
    then obtain r1 where im: ⋀i. r1 i =  f -` r i and a: ⋀i. r1 i ∈        a
      by (simp add: Q-def)
    from a sig have UNION UNIV r1 ∈ a
      by (auto simp only: sigma-algebra-def)
    with im Q-def show UNION UNIV r ∈ Q
      by (auto simp add: vimage-UN)
  qed }

 hence (sigma b) ⊆ Q ..
 thus f ∈ measurable a (sigma b)
   by (auto simp add: measurable-def Q-def)
qed
```

The case is different for the second theorem. It is only five lines in the book (ibid.), but almost 200 in formal text. Precision still pays here, gaining a detailed view of a technique that is often employed in measure theory — making a sequence of sets disjoint. Moreover, the necessity for the above-mentioned change in the definition of countably additive was detected only in the formalization of this proof.

To enable application of the additivity of measures, the following construction yields disjoint sets. We skip the justification of the lemmata for brevity.

```
primrec mkdisjoint:: (nat ⇒ 'a set) ⇒ (nat ⇒ 'a set)
where
  mkdisjoint A 0 = A 0
| mkdisjoint A (Suc n) = A (Suc n) - A n

lemma mkdisjoint-un:
  assumes up: ⋀n. A n ⊆ A (Suc n)
```

shows $A\ n = (\bigcup i{\in}\{..n\}.\ mkdisjoint\ A\ i)$

lemma *mkdisjoint-disj*:
assumes *up*: $\bigwedge n.\ A\ n \subseteq A\ (Suc\ n)$ **and** *ne*: $m \neq n$
shows $mkdisjoint\ A\ m \cap mkdisjoint\ A\ n = \{\}$

lemma *mkdisjoint-mon-conv*:
assumes *mc*: $A{\uparrow}B$
shows $(\bigcup i.\ mkdisjoint\ A\ i) = B$

Joe Hurd calls the following the Monotone Convergence Theorem, though in mathematical literature this name is often reserved for a similar fact about integrals that we will prove in 3.2.2, which depends on this one. The claim made here is that the measures of monotonically convergent sets approach the measure of their limit. A strengthened version would imply monotone convergence of the measures, but is not needed in the development.

theorem *measure-mon-conv*:
assumes *ms*: *measure-space M* **and**
Ams: $\bigwedge n.\ A\ n \in measurable\text{-}sets\ M$ **and** *AB*: $A{\uparrow}B$
shows $(\lambda n.\ measure\ M\ (A\ n)) \longrightarrow measure\ M\ B$
proof −

from *AB* **have** *up*: $\bigwedge n.\ A\ n \subseteq A\ (Suc\ n)$
by (*simp only*: *mon-conv-set-def*)

{ fix *i*
have $mkdisjoint\ A\ i \in measurable\text{-}sets\ M$
proof (*cases i*)
case *0* **with** *Ams* **show** *?thesis* **by** *simp*
next
case (*Suc i*)
have $A\ (Suc\ i) - A\ i = A\ (Suc\ i) \cap - A\ i$ **by** *blast*
with *Suc ms Ams* **show** *?thesis*
by (*auto simp add*: *measure-space-def sigma-algebra-def sigma-algebra-inter*)
qed
}
hence *i*: $\bigwedge i.\ mkdisjoint\ A\ i \in measurable\text{-}sets\ M$ **.**

with *ms* **have** *un*: $(\bigcup i.\ mkdisjoint\ A\ i) \in measurable\text{-}sets\ M$
by (*simp add*: *measure-space-def sigma-algebra-def*)
moreover
from *i* **have** *range*: $range\ (mkdisjoint\ A) \subseteq measurable\text{-}sets\ M$
by *fast*
moreover
from *up* **have** $\forall i\ j.\ i \neq j \longrightarrow mkdisjoint\ A\ i \cap mkdisjoint\ A\ j = \{\}$
by (*simp add*: *mkdisjoint-disj*)
moreover note *ms*
ultimately
have *sums*:

(λi. measure M (mkdisjoint A i)) sums (measure M (⋃ i. mkdisjoint A i))
by (simp add: measure-space-def countably-additive-def)
hence (∑ i. measure M (mkdisjoint A i)) = (measure M (⋃ i. mkdisjoint A i))
by (rule sums-unique[THEN sym])

also
from sums **have** summable (λi. measure M (mkdisjoint A i))
by (rule sums-summable)

hence (λn. ∑ i<n. measure M (mkdisjoint A i))
⟶ (∑ i. measure M (mkdisjoint A i))
by (rule summable-LIMSEQ)

hence (λn. ∑ i<Suc n. measure M (mkdisjoint A i)) ⟶ (∑ i. measure M (mkdisjoint A i))
by (rule LIMSEQ-Suc)

ultimately have (λn. ∑ i<Suc n. measure M (mkdisjoint A i))
⟶ (measure M (⋃ i. mkdisjoint A i)) **by** simp

also
{ fix n
from up **have** A n = (⋃ i∈{..n}. mkdisjoint A i)
by (rule mkdisjoint-un)
hence measure M (A n) = measure M (⋃ i∈{..n}. mkdisjoint A i)
by simp

also have
(⋃ i∈{..n}. mkdisjoint A i) = (⋃ i. if i≤n then mkdisjoint A i else {})
proof −
have UNIV = {..n} ∪ {n<..} **by** auto
hence (⋃ i. if i≤n then mkdisjoint A i else {}) =
(⋃ i∈{..n}. if i≤n then mkdisjoint A i else {})
∪ (⋃ i∈{n<..}. if i≤n then mkdisjoint A i else {})
by (auto split: if-splits)
moreover
{ have (⋃ i∈{n<..}. if i≤n then mkdisjoint A i else {}) = {}
by force **}**
hence ... = (⋃ i∈{..n}. mkdisjoint A i)
by auto
ultimately show
(⋃ i∈{..n}. mkdisjoint A i) = (⋃ i. if i≤n then mkdisjoint A i else {}) **by** simp
qed

ultimately have
measure M (A n) = measure M (⋃ i. if i≤n then mkdisjoint A i else {})
by simp

```
    also
    from i ms have
      un: (⋃ i. if i≤n then mkdisjoint A i else {}) ∈ measurable-sets M
      by (simp add: measure-space-def sigma-algebra-def)
    moreover
    from i ms have
      range (λi. if i≤n then mkdisjoint A i else {}) ⊆ measurable-sets M
      by (auto simp add: measure-space-def sigma-algebra-def)
    moreover
    from up have ∀ i j. i ≠ j ⟶
      (if i≤n then mkdisjoint A i else {}) ∩
      (if j≤n then mkdisjoint A j else {}) = {}
      by (simp add: mkdisjoint-disj)
    moreover note ms
    ultimately have
      measure M (A n) = (∑ i. measure M (if i ≤ n then mkdisjoint A i else {}))
      by (simp add: measure-space-def countably-additive-def sums-unique)

    also
    from ms have
      ∀ i. (Suc n)≤i ⟶ measure M (if i ≤ n then mkdisjoint A i else {}) = 0
      by (simp add: measure-space-def positive-def)
    hence (λi. measure M (if i ≤ n then mkdisjoint A i else {})) sums
      (∑ i<Suc n. measure M (if i ≤ n then mkdisjoint A i else {}))
      by (intro sums-finite) auto
    hence (∑ i. measure M (if i ≤ n then mkdisjoint A i else {})) =
      (∑ i<Suc n. measure M (if i ≤ n then mkdisjoint A i else {}))
      by (rule sums-unique[THEN sym])
    also
    have ... = (∑ i<Suc n. measure M (mkdisjoint A i))
      by simp
    finally have
      measure M (A n) = (∑ i<Suc n. measure M (mkdisjoint A i)) .
  }

  ultimately have
    (λn. measure M (A n)) ⟶⟶ (measure M (⋃ i. mkdisjoint A i))
    by simp

  with AB show ?thesis
    by (simp add: mkdisjoint-mon-conv)
qed
```

2.2 Real-Valued random variables

theory *RealRandVar*
imports *Measure HOL−Library.Countable*
begin

While most of the above material was modeled after Hurd's work (but still proved independently), the original content basically starts here[5]. From now on, we will specialize in functions that map into the real numbers and are measurable with respect to the canonical sigma algebra on the reals, the Borel sigma algebra. These functions will be called real-valued random variables. The terminology is slightly imprecise, as random variables hint at a probability space, which usually requires *measure M UNIV = 1*. Notwithstanding, as we regard only finite measures (cf. 2.1.3), this condition can easily be achieved by normalization. After all, the other standard name, "measurable functions", is even less precise.

A lot of the theory in this and the preceding section has also been formalized within the Mizar project [3, 4]. The abstract of the second source hints that it was also planned as a stepping stone for Lebesgue integration, though further results in this line could not be found. The main difference lies in the use of extended real numbers — the reals together with $\pm\infty$ — in those documents. It is established practice in measure theory to allow infinite values, but "(...) we felt that the complications that this generated (...) more than canceled out the gain in uniformity (...), and that a simpler theory resulted from sticking to the standard real numbers." [6, p. 32f]. Hurd also advocates going directly to the hyper-reals, should the need for infinite measures arise. I agree, nevertheless sticking to his example for the reasons mentioned in the prologue.

definition
Borelsets:: *real set set* (𝔹) **where**
𝔹 = *sigma* {*S*. ∃ *u*. *S*={..*u*}}

definition

rv:: (*'a set set* ∗ (*'a set* ⇒ *real*)) ⇒ (*'a* ⇒ *real*) *set* **where**
rv M = {*f*. *measure-space M* ∧ *f* ∈ *measurable* (*measurable-sets M*) 𝔹}

As explained in the first paragraph, the preceding definitions[6] determine the rest of this section. There are many ways to define the Borel sets. For example, taking into account only rationals for u would also have worked

[5] There are two main reasons why the above has not been imported using Sebastian Skalberg's import tool [8]. Firstly, there are inconveniences caused by different conventions in HOL, meaning predicates instead of sets foremost, that make the consistent use of such basic definitions impractical. What is more, the import tool simply was not available at the time these theories were written.

[6] The notation {..*u*} signifies the interval from negative infinity to u included.

out above, but we can take the reals to simplify things. The smallest sigma algebra containing all the open (or closed) sets is another alternative; the multitude of possibilities testifies to the relevance of the concept.

The latter path leads the way to the fact that any continuous function is measurable. Generalization for $\mathbb{R}^n$ brings another unified way to prove all the measurability theorems in this theory plus, for instance, measurability of the trigonometric and exponential functions. This approach is detailed in another influential textbook by Billingsley [2]. It requires some concepts of topologic spaces, which made the following elementary course, based on Bauer's excellent book [1], seem more feasible.

Two more definitions go next. The image measure, law, or distribution — the last term being specific to probability — of a measure with respect to a measurable function is calculated as the measure of the inverse image of a set. Characteristic functions will be frequently needed in the rest of the development.

definition
distribution::
('*a set set* ∗ ('*a set* ⇒ *real*)) ⇒ ('*a* ⇒ *real*) ⇒ (*real set* ⇒ *real*) (*law*) **where**
f ∈ *rv M* ⟹ *law M f* ≡ (*measure M*) ∘ (*vimage f*)

definition
characteristic-function:: '*a set* ⇒ ('*a* ⇒ *real*) (χ -) **where**
χ *A x* ≡ *if x* ∈ *A then 1 else 0*

lemma *char-empty*: χ {} = (λ*t*. *0*)
proof (*rule ext*)
fix *t*
show χ {} *t* = *0* **by** (*simp add*: *characteristic-function-def*)
qed

Now that random variables are defined, we aim to show that a broad class of functions belongs to them. For a constant function this is easy, as there are only two possible preimages.

lemma assumes *sigma*: *sigma-algebra S*
shows *const-measurable*: (λ*x*. (*c*::*real*)) ∈ *measurable S X*
proof (*unfold measurable-def*, *rule*, *rule*)
fix *g*
show (λ*x*. *c*) −‘ *g* ∈ *S*
proof (*cases c* ∈ *g*)
case *True*
hence (λ*x*::'*a*. *c*) −‘ *g* = *UNIV*
by *blast*
moreover from *sigma* **have** *UNIV* ∈ *S*
by (*rule sigma-algebra-UNIV*)
ultimately show *?thesis* **by** *simp*
next

```
    case False
    hence (λx::'a. c) -` g = {}
      by blast
    moreover from sigma have {} ∈ S
      by (simp only: sigma-algebra-def)
    ultimately show ?thesis by simp

  qed
qed

theorem assumes ms: measure-space M
  shows const-rv: (λx. c) ∈ rv M using ms
  by (auto simp only: measure-space-def const-measurable rv-def)
```

Characteristic functions produce four cases already, so the details are glossed over.

```
lemma assumes a: a ∈ S and sigma: sigma-algebra S shows
char-measurable : χ a ∈ measurable S x

theorem assumes ms: measure-space M and A: A ∈ measurable-sets M
  shows char-rv: χ A ∈ rv M using ms A
  by (auto simp only: measure-space-def char-measurable rv-def)
```

For more intricate functions, the following application of the measurability lifting theorem from 2.1.3 gives a useful characterization.

```
theorem assumes ms: measure-space M shows
  rv-le-iff: (f ∈ rv M) = (∀ a. {w. f w ≤ a} ∈ measurable-sets M)
proof −

  have f ∈ rv M ⟹ ∀ a. {w. f w ≤ a} ∈ measurable-sets M
  proof
    { fix a
      assume f ∈ measurable (measurable-sets M) 𝔹
      hence ∀ b∈𝔹. f -` b ∈ measurable-sets M
        by (unfold measurable-def) blast
      also have {..a} ∈ 𝔹
        by (simp only: Borelsets-def) (rule sigma.basic, blast)
      ultimately have {w. f w ≤ a} ∈ measurable-sets M
        by (auto simp add: vimage-def)
    }
    thus ⋀a. f ∈ rv M ⟹ {w. f w ≤ a} ∈ measurable-sets M
      by (simp add: rv-def)
  qed

  also have ∀ a. {w. f w ≤ a} ∈ measurable-sets M ⟹ f ∈ rv M
  proof −
    assume ∀ a. {w. f w ≤ a} ∈ measurable-sets M
    hence f ∈ measurable (measurable-sets M){S. ∃ u. S={..u}}
      by (auto simp add: measurable-def vimage-def)
```

```
    with ms have f ∈ measurable (measurable-sets M) 𝔹
      by (simp only: Borelsets-def measure-space-def measurable-lift)
    with ms show ?thesis
      by (auto simp add: rv-def)

  qed
  ultimately show ?thesis by rule
qed
```

The next four lemmata allow for a ring deduction that helps establish this fact for all of the signs $<$, $>$ and $\geq$ as well.

```
lemma assumes sigma: sigma-algebra A and le: ∀ a. {w. f w ≤ a} ∈ A
  shows le-less: ∀ a. {w. f w < (a::real)} ∈ A
proof
  fix a::real
  from le sigma have (⋃ n::nat. {w. f w ≤ a − inverse (real (Suc n))}) ∈ A
    by (simp add: sigma-algebra-def)
  also
  have (⋃ n::nat. {w. f w ≤ a − inverse (real (Suc n))}) = {w. f w < a}
  proof −
    {
      fix w n
      have 0 < inverse (real (Suc (n::nat)))
        by simp
      hence f w ≤ a − inverse (real (Suc n)) ⟹ f w < a
        by arith
    }
    also
    { fix w
      have (λn. inverse (real (Suc n))) ⟶ 0
        by (rule LIMSEQ-inverse-real-of-nat)

      also assume f w < a
      hence 0 < a − f w by simp

      ultimately have
        ∃ n0. ∀ n. n0 ≤ n ⟶ abs (inverse (real (Suc n))) < a − f w
        by (auto simp add: lim-sequentially dist-real-def)
      then obtain n where  abs (inverse (real (Suc n))) < a − f w
        by blast
      hence f w ≤ a − inverse (real (Suc n))
        by arith
      hence ∃ n. f w ≤ a − inverse (real (Suc n)) ..
    }
    ultimately show ?thesis by auto
  qed
  finally show {w. f w < a} ∈ A .
qed
```

lemma assumes *sigma*: *sigma-algebra A* **and** *less*: $\forall\, a.\ \{w.\ f\ w < a\} \in A$
shows *less-ge*: $\forall\, a.\ \{w.\ (a{::}real) \leq f\ w\} \in A$
proof
fix *a*::*real*
from *less sigma* **have** $-\{w.\ f\ w < a\} \in A$
by (*simp add*: *sigma-algebra-def*)
also
have $-\{w.\ f\ w < a\} = \{w.\ a \leq f\ w\}$
by *auto*

finally show $\{w.\ a \leq f\ w\} \in A$ **.**
qed

lemma assumes *sigma*: *sigma-algebra A* **and** *ge*: $\forall\, a.\ \{w.\ a \leq f\ w\} \in A$
shows *ge-gr*: $\forall\, a.\ \{w.\ (a{::}real) < f\ w\} \in A$

lemma assumes *sigma*: *sigma-algebra A* **and** *gr*: $\forall\, a.\ \{w.\ a < f\ w\} \in A$
shows *gr-le*: $\forall\, a.\ \{w.\ f\ w \leq (a{::}real)\} \in A$
theorem assumes *ms*: *measure-space M* **shows**
rv-ge-iff: $(f \in rv\ M) = (\forall\, a.\ \{w.\ a \leq f\ w\} \in measurable\text{-}sets\ M)$
proof −
from *ms* **have** $(f \in rv\ M) = (\forall\, a.\ \{w.\ f\ w \leq a\} \in measurable\text{-}sets\ M)$
by (*rule rv-le-iff*)
also have $\ldots = (\forall\, a.\ \{w.\ a \leq f\ w\} \in measurable\text{-}sets\ M)$ (**is** *?lhs* = *?rhs*)
proof −
from *ms* **have** *sigma*: *sigma-algebra* (*measurable-sets M*)
by (*simp only*: *measure-space-def*)
also note *less-ge le-less*
ultimately have *?lhs* $\Longrightarrow$ *?rhs* **by** *blast*
also
from *sigma gr-le ge-gr* **have** *?rhs* $\Longrightarrow$ *?lhs* **by** *blast*
ultimately
show *?thesis* **..**
qed
finally show *?thesis* **.**
qed

theorem assumes *ms*: *measure-space M* **shows**
rv-gr-iff: $(f \in rv\ M) = (\forall\, a.\ \{w.\ a < f\ w\} \in measurable\text{-}sets\ M)$

theorem assumes *ms*: *measure-space M* **shows**
rv-less-iff: $(f \in rv\ M) = (\forall\, a.\ \{w.\ f\ w < a\} \in measurable\text{-}sets\ M)$

As a first application we show that addition and multiplication with constants preserve measurability. This is a precursor to the more general addition and multiplication theorems later on. You can see that quite a few properties of the real numbers are employed.

lemma assumes *g*: $g \in rv\ M$
shows *affine-rv*: $(\lambda x.\ (a{::}real) + (g\ x) * b) \in rv\ M$

```
proof (cases b=0)
  from g have ms: measure-space M
    by (simp add: rv-def)
  case True
  hence (λx. a + (g x) * b) = (λx. a)
    by simp
  also
  from g have (λx. a) ∈ rv M
    by (simp add: const-measurable rv-def measure-space-def)
  ultimately show ?thesis by simp

next
  from g have ms: measure-space M
    by (simp add: rv-def)
  case False
  have calc: ⋀x c. (a + g x * b ≤ c) = (g x * b  ≤ c - a)
    by arith
  have ∀ c. {w.  a + g w * b ≤ c} ∈ measurable-sets M
  proof (cases b<0)
    case False
    with ‹b ≠ 0› have 0<b by arith
    hence ⋀x c.  (g x * b  ≤ c - a) = (g x ≤ (c - a) / b)
      by (rule pos-le-divide-eq [THEN sym])
    with calc have ⋀c. {w.  a + g w * b ≤ c} = {w. g w ≤ (c - a) / b}
      by simp

    also from ms g  have ∀ a. {w. g w ≤ a} ∈ measurable-sets M
      by (simp add: rv-le-iff)

    ultimately show ?thesis by simp

  next
    case True
    hence ⋀x c. (g x * b ≤ c-a) = ((c-a)/b ≤ g x)
      by (rule neg-divide-le-eq [THEN sym])
    with calc have ⋀c. {w.  a + g w * b ≤ c} = {w. (c-a)/b ≤ g w}
      by simp

    also from ms g have ∀ a. {w. a ≤ g w } ∈ measurable-sets M
      by (simp add: rv-ge-iff)

    ultimately show ?thesis by simp
  qed

  with ms show ?thesis
    by (simp only: rv-le-iff [THEN sym])
qed
```

For the general case of addition, we need one more set to be measurable,

namely $\{w.\ f\ w \leq g\ w\}$. This follows from a like statement for $<$. A dense and countable subset of the reals is needed to establish it.

Of course, the rationals come to mind. They were not available in Isabelle/HOL[7], so I built a theory with the necessary properties on my own. [Meanwhile Isabelle has proper rationals and SR's development of the rationals has been moved to and merged with Isabelle's rationals.

```
lemma assumes f: f ∈ rv M and g: g ∈ rv M
  shows rv-less-rv-measurable: {w. f w < g w} ∈ measurable-sets M
proof −
  let ?I i = let s::real = of-rat(nat-to-rat-surj i) in {w. f w < s} ∩ {w. s < g w}
  from g have ms: measure-space M by (simp add: rv-def)
  have {w. f w < g w} = (⋃ i. ?I i)
  proof
    { fix w assume w ∈ {w. f w < g w}
      hence f w < g w ..
      hence ∃ s∈ℚ. f w < s ∧ s < g w by (rule Rats-dense-in-real)
      hence ∃ s∈ℚ. w ∈ {w. f w < s} ∩ {w. s < g w} by simp
      hence ∃ i. w ∈ ?I i
        by(simp add:Let-def)(metis surj-of-rat-nat-to-rat-surj)
      hence w ∈ (⋃ i. ?I i) by simp
    }
    thus {w. f w < g w} ⊆ (⋃ i. ?I i) ..

    show (⋃ i. ?I i) ⊆ {w. f w < g w} by (force simp add: Let-def)
  qed
  moreover have (⋃ i. ?I i) ∈ measurable-sets M
  proof −
    from ms have sig: sigma-algebra (measurable-sets M)
      by (simp only: measure-space-def)
    { fix s
      note sig
      also from ms f have {w. f w < s} ∈ measurable-sets M (is ?a∈?M)
        by (simp add: rv-less-iff)
      moreover from ms g have {w. s < g w} ∈ ?M (is ?b ∈ ?M)
        by (simp add: rv-gr-iff)
      ultimately have ?a ∩ ?b ∈ ?M by (rule sigma-algebra-inter)
    }
    hence ∀ i. ?I i ∈ measurable-sets M by (simp add: Let-def)
    with sig show ?thesis by (auto simp only: sigma-algebra-def Let-def)
  qed
  ultimately show ?thesis by simp
qed


lemma assumes f: f ∈ rv M and g: g ∈ rv M
  shows rv-le-rv-measurable: {w. f w ≤ g w} ∈ measurable-sets M (is ?a ∈ ?M)
```

[7] At least not as a subset of the reals, to the definition of which a type of positive rational numbers contributed [5].

proof −
from *g* **have** *ms*: *measure-space M*
by (*simp add*: *rv-def*)
from *g f* **have** $\{w.\ g\ w < f\ w\} \in ?M$
by (*rule rv-less-rv-measurable*)
also from *ms* **have** *sigma-algebra ?M*
by (*simp only*: *measure-space-def*)

ultimately have $-\{w.\ g\ w < f\ w\} \in ?M$
by (*simp only*: *sigma-algebra-def*)
also have $-\{w.\ g\ w < f\ w\} = ?a$
by *auto*

finally show *?thesis* **.**
qed

lemma assumes *f*: $f \in rv\ M$ **and** *g*: $g \in rv\ M$
shows *f-eq-g-measurable*: $\{w.\ f\ w = g\ w\} \in measurable\text{-}sets\ M$

lemma assumes *f*: $f \in rv\ M$ **and** *g*: $g \in rv\ M$
shows *f-noteq-g-measurable*: $\{w.\ f\ w \neq g\ w\} \in measurable\text{-}sets\ M$

With these tools, a short proof for the addition theorem is possible.

theorem assumes *f*: $f \in rv\ M$ **and** *g*: $g \in rv\ M$
shows *rv-plus-rv*: $(\lambda w.\ f\ w + g\ w) \in rv\ M$
proof −
from *g* **have** *ms*: *measure-space M* **by** (*simp add*: *rv-def*)
{ fix *a*
have $\{w.\ a \leq f\ w + g\ w\} = \{w.\ a + (g\ w)*(-1) \leq f\ w\}$
by *auto*
moreover from *g* **have** $(\lambda w.\ a + (g\ w)*(-1)) \in rv\ M$
by (*rule affine-rv*)
with *f* **have** $\{w.\ a + (g\ w)*(-1) \leq f\ w\} \in measurable\text{-}sets\ M$
by (*simp add*: *rv-le-rv-measurable*)
ultimately have $\{w.\ a \leq f\ w + g\ w\} \in measurable\text{-}sets\ M$ **by** *simp*
}
with *ms* **show** *?thesis*
by (*simp add*: *rv-ge-iff*)
thm *rv-ge-iff*
qed

To show preservation of measurability by multiplication, it is expressed by addition and squaring. This requires a few technical lemmata including the one stating measurability for squares, the proof of which is skipped.

lemma *pow2-le-abs*: $(a^2 \leq b^2) = (|a| \leq |b::real|)$
lemma assumes *f*: $f \in rv\ M$
shows *rv-square*: $(\lambda w.\ (f\ w)^2) \in rv\ M$
lemma *realpow-two-binomial-iff*: $(f+g::real)^2 = f^2 + 2*(f*g) + g^2$

lemma *times-iff-sum-squares*: $f*g = (f+g)^2/4 - (f-g)^2/(4::real)$
by (*simp add*: *power2-eq-square field-simps*)

theorem assumes *f*: $f \in rv\ M$ **and** *g*: $g \in rv\ M$
shows *rv-times-rv*: $(\lambda w.\ f\ w * g\ w) \in rv\ M$
proof −
have $(\lambda w.\ f\ w * g\ w) = (\lambda w.\ (f\ w + g\ w)^2/4 - (f\ w - g\ w)^2/4)$
by (*simp only*: *times-iff-sum-squares*)
also have $\ldots = (\lambda w.\ (f\ w + g\ w)^2 * inverse\ 4 - (f\ w + - g\ w)^2 * inverse\ 4)$
by *simp*
also from *f g* **have** $\ldots \in rv\ M$
proof −
from *f g* **have** $(\lambda w.\ (f\ w + g\ w)^2) \in rv\ M$
by (*simp add*: *rv-plus-rv rv-square*)
hence $(\lambda w.\ 0 + (f\ w + g\ w)^2 * inverse\ 4) \in rv\ M$
by (*rule affine-rv*)
also from *g* **have** $(\lambda w.\ 0 + (g\ w) * -1\ ) \in rv\ M$
by (*rule affine-rv*)
with *f* **have** $(\lambda w.\ (f\ w - g\ w)^2) \in rv\ M$
by (*simp add*: *rv-plus-rv rv-square diff-conv-add-uminus del*: *add-uminus-conv-diff*)
hence $(\lambda w.\ 0 + (f\ w - g\ w)^2 * -inverse\ 4) \in rv\ M$
by (*rule affine-rv*)
ultimately show *?thesis*
by (*simp add*: *rv-plus-rv diff-conv-add-uminus del*: *add-uminus-conv-diff*)
qed

ultimately show *?thesis* **by** *simp*
qed

The case of substraction is an easy consequence of *rv-plus-rv* and *rv-times-rv*.

theorem *rv-minus-rv*:
assumes *f*: $f \in rv\ M$ **and** *g*: $g \in rv\ M$
shows $(\lambda t.\ f\ t - g\ t) \in rv\ M$

Measurability for limit functions of monotone convergent series is also surprisingly straightforward.

theorem assumes *u*: $\bigwedge n.\ u\ n \in rv\ M$ **and** *mon-conv*: $u \uparrow f$
shows *mon-conv-rv*: $f \in rv\ M$
proof −
from *u* **have** *ms*: *measure-space M*
by (*simp add*: *rv-def*)

{
fix *a*
{
fix *w*
from *mon-conv* **have** *up*: $(\lambda n.\ u\ n\ w) \uparrow f\ w$
by (*simp only*: *realfun-mon-conv-iff*)
{

```
      fix i
      from up have u i w ≤ f w
        by (rule real-mon-conv-le)
      also assume f w ≤ a
      finally have  u i w ≤ a .
    }

    also
    { assume ⋀i. u i w ≤ a
      also from up have (λn. u n w) ⟶ f w
        by (simp only: mon-conv-real-def)
      ultimately have f w ≤ a
        by (simp add: LIMSEQ-le-const2)
    }
    ultimately have (f w ≤ a) = (∀ i. u i w ≤ a) by fast
  }
  hence {w. f w ≤ a} = (⋂ i. {w. u i w ≤ a}) by fast
  moreover
  from ms u have ⋀i. {w. u i w ≤ a} ∈ sigma(measurable-sets M)
    by (simp add: rv-le-iff sigma.intros)
  hence (⋂ i. {w. u i w ≤ a}) ∈ sigma(measurable-sets M)
    by (rule sigma-Inter)
  with ms have (⋂ i. {w. u i w ≤ a}) ∈ measurable-sets M
    by (simp only: measure-space-def sigma-sigma-algebra)
  ultimately have {w. f w ≤ a} ∈ measurable-sets M by simp
 }
 with ms show ?thesis
   by (simp add: rv-le-iff)
qed
```

Before we end this chapter to start the formalization of the integral proper, there is one more concept missing: The positive and negative part of a function. Their definition is quite intuitive, and some useful properties are given right away, including the fact that they are random variables, provided that their argument functions are measurable.

```
definition
  nonnegative:: ('a ⇒ ('b::{ord,zero})) ⇒ bool where
  nonnegative f ⟷ (∀ x. 0 ≤ f x)

definition
  positive-part:: ('a ⇒ ('b::{ord,zero})) ⇒ ('a ⇒ 'b) (pp) where
  pp f x = (if 0≤f(x) then f x else 0)

definition
  negative-part:: ('a ⇒ ('b::{ord,zero,uminus,minus})) ⇒ ('a ⇒ 'b) (np) where
  np f x = (if 0≤f(x) then 0 else −f(x))

lemma f-plus-minus: ((f x)::real) = pp f x − np f x
  by (simp add:positive-part-def negative-part-def)
```

lemma *f-plus-minus2*: $(f::'a \Rightarrow real) = (\lambda t.\ pp\ f\ t - np\ f\ t)$
using *f-plus-minus*
by (*rule ext*)

lemma *f-abs-plus-minus*: $(|f\ x|::real) = pp\ f\ x + np\ f\ x$
by (*auto simp add*:*positive-part-def negative-part-def*)

lemma *nn-pp-np*: **assumes** *nonnegative f*
shows *pp f* = *f* **and** *np f* = $(\lambda t.\ 0)$ **using** *assms*
by (*auto simp add*: *positive-part-def negative-part-def nonnegative-def ext*)

lemma *pos-pp-np-help*: $\bigwedge x.\ 0 \leq f\ x \Longrightarrow pp\ f\ x = f\ x \wedge np\ f\ x = 0$
by (*simp add*: *positive-part-def negative-part-def*)

lemma *real-neg-pp-np-help*: $\bigwedge x.\ f\ x \leq (0::real) \Longrightarrow np\ f\ x = -f\ x \wedge pp\ f\ x = 0$
lemma *real-neg-pp-np*: **assumes** $f \leq (\lambda t.\ (0::real))$
shows *np f* = $(\lambda t.\ -f\ t)$ **and** *pp f* = $(\lambda t.\ 0)$ **using** *assms*
by (*auto simp add*: *real-neg-pp-np-help ext le-fun-def*)

lemma assumes *a*: $0 \leq (a::real)$
shows *real-pp-np-pos-times*:
$pp\ (\lambda t.\ a * f\ t) = (\lambda t.\ a * pp\ f\ t) \wedge\ \ np\ (\lambda t.\ a * f\ t) = (\lambda t.\ a * np\ f\ t)$
lemma assumes *a*: $(a::real) \leq 0$
shows *real-pp-np-neg-times*:
$pp\ (\lambda t.\ a * f\ t) = (\lambda t.\ -a * np\ f\ t) \wedge\ \ np\ (\lambda t.\ a * f\ t) = (\lambda t.\ -a * pp\ f\ t)$

lemma *pp-np-rv*:
assumes *f*: $f \in rv\ M$
shows *pp f* $\in$ *rv M* **and** *np f* $\in$ *rv M*
proof −
from *f* **have** *ms*: *measure-space M* **by** (*simp add*: *rv-def*)

{ fix *a*
from *ms f* **have** *fm*: $\{w.\ f\ w \leq a\} \in$ *measurable-sets M*
by (*simp add*: *rv-le-iff*)
have
$\{w.\ pp\ f\ w \leq a\} \in$ *measurable-sets M* $\wedge$
$\{w.\ np\ f\ w \leq a\} \in$ *measurable-sets M*
proof (*cases* $0 \leq a$)
case *True*
hence $\{w.\ pp\ f\ w \leq a\} = \{w.\ f\ w \leq a\}$
by (*auto simp add*: *positive-part-def*)
moreover note *fm* **moreover**
from *True* **have** $\{w.\ np\ f\ w \leq a\} = \{w.\ -a \leq f\ w\}$
by (*auto simp add*: *negative-part-def*)
moreover from *ms f* **have** ... $\in$ *measurable-sets M*
by (*simp add*: *rv-ge-iff*)
ultimately show *?thesis* **by** *simp*

```
    next
      case False
      hence {w. pp f w ≤ a} = {}
        by (auto simp add: positive-part-def)
      also from False have {w. np f w ≤ a} = {}
        by (auto simp add: negative-part-def)
      moreover from ms have {} ∈ measurable-sets M
        by (simp add: measure-space-def sigma-algebra-def)
      ultimately show ?thesis by simp
    qed
  } with ms show pp f ∈ rv M and np f ∈ rv M
    by (auto simp add: rv-le-iff)
qed
```

theorem *pp-np-rv-iff*: $(f::'a \Rightarrow real) \in rv\ M = (pp\ f \in rv\ M \wedge np\ f \in rv\ M)$

This completes the chapter about measurable functions. As we will see in the next one, measurability is the prime condition on Lebesgue integrable functions; and the theorems and lemmata established here suffice — at least in principle — to show it holds for any function that is to be integrated there.

end

Chapter 3

Integration

The chapter at hand assumes a central position in the present paper. The Lebesgue integral is defined and its characteristics are shown in 3.2. To illustrate the problems arising in doing so, we first look at implementation alternatives that did not work out.

3.1 Two approaches that failed

Defining Lebesgue integration can be quite involved, judging by the process in 3.2 that imitates Bauer's way [1]. So it is quite tempting to try cutting a corner. The following two alternative approaches back up my experience that this almost never pays in formalization. The theory that seems most complex at first sight is often the one that is closest to formal reasoning and deliberately avoids "hand-waving".

3.1.1 A closed expression

In contrast, Billingsley's definition [2, p. 172] is strikingly short. For non-negative measurable functions f:

> $\int f d\mu = sup \sum_i \left[inf_{\omega \in A_i} f(w) \right] \mu(A_i)$.
>
> The supremum here extends over all finite decompositions $\{A_i\}$ of Ω into $\mathcal{F}$-sets.[1]

Like the definition, the proofs of the essential properties are also rather short, about three pages in the textbook for almost all the theorems in 3.2; and a proof of uniqueness is obsolete for a closed expression like this. Therefore, I found this approach quite tempting. It turns out, however,

[1] The $\mathcal{F}$-sets are just the measurable sets of a measure space.

that it is unfortunately not well suited for formalization, at least with the background we use.

A complication shared by all possible styles of definition is the lack of infinite values in our theory, combined with the lack of partial functions in HOL. Like the sum operator in 2.1.3, the integral has to be defined indirectly. The classical way to do this employs predicates, invoking ε to choose the value that satisfies the condition:

$\int f\,dM \equiv (\varepsilon\ i.\ \textit{is-integral}\ M\ f\ i)$

To sensibly apply this principle, the predicate has to be ε-free to supply the information if the integral is defined or not. Now the above definition contains up to three additional ε when formalized naively in HOL, namely in the supremum, infimum and sum operators. The sum is over a finite set, so it can be replaced by a total function. For nonnegative functions, the infimum can also be shown to exist everywhere, but, like the supremum, must itself be replaced by a predicate.

Also note that predicates require a proof of uniqueness, thus losing the prime advantage of a closed formula anyway. In this case, uniqueness can be reduced to uniqueness of the supremum/infimum. The problem is that neither suprema nor infima come predefined in Isabelle/Isar as of yet. It is an easy task to make up for this — and I did — but a much harder one to establish all the properties needed for reasoning with the defined entities.

A lot of such reasoning is necessary to deduce from the above definition (or a formal version of it, as just outlined) the basic behavior of integration, which includes additivity, monotonicity and especially the integral of simple functions. It turns out that the brevity of the proofs in the textbook stems from a severely informal style that assumes ample background knowledge. Formalizing all this knowledge started to become overwhelming when the idea of a contrarian approach emerged.

3.1.2 A one-step inductive definition

This idea was sparked by the following note: "(. . .) the integral is uniquely determined by certain simple properties it is natural to require of it" [2, p. 175]. Billingsley goes on discussing exactly those properties that are so hard to derive from his definition. So why not simply define integration using these properties? That is the gist of an inductive set definition, like the one we have seen in 2.1.1. This time a functional operator is to be defined, but it can be represented as a set of pairs, where the first component is the function and the second its integral. To cut a long story short, here is the definition.

inductive-set
integral-set:: $('a\ set\ set * ('a\ set \Rightarrow real)) \Rightarrow (('a \Rightarrow real) * real)\ set$
for $M :: 'a\ set\ set * ('a\ set \Rightarrow real)$

where
char: $[\![f = \chi\ A;\ A \in \textit{measurable-sets}\ M]\!] \Longrightarrow (f,\textit{measure}\ M\ A) \in \textit{integral-set}\ M$
| *add*: $[\![f = (\lambda w.\ g\ w + h\ w);\ (g,x) \in \textit{integral-set}\ M;\ (h,y) \in \textit{integral-set}\ M]\!]$
$\Longrightarrow (f,(x + y)) \in \textit{integral-set}\ M$
| *times*: $[\![f = (\lambda w.\ a*g\ w);\ (g,x) \in \textit{integral-set}\ M]\!] \Longrightarrow (f,a*x) \in \textit{integral-set}\ M$
| *mon-conv*: $[\![u\uparrow f;\ \bigwedge n.\ (u\ n,\ x\ n) \in \textit{integral-set}\ M;\ x\uparrow y]\!]$
$\Longrightarrow (f,y) \in \textit{integral-set}\ M$

The technique is also encountered in the *Finite-Set* theory from the Isabelle library. It is used there to define the *sum* function, which calculates a sum indexed over a finite set and is employed in 3.2. The definition here is much more intricate though.

An obvious advantage of this approach is that almost all important properties are gained without effort. The introduction rule *mon-conv* corresponds to what is known as the Monotone Convergence Theorem in scientific literature; negative functions are also provided for via the *times* rule. To be precise, there is exactly one important theorem missing — uniqueness. That is, every function appears in at most one pair.

From uniqueness together with the introduction rules, all the other statements about integration, monotonicity for example, could be derived. On the other hand, monotonicity implies uniqueness. Much to my regret, none of these two could be proven. The proof would basically amount to a double induction to show that an integral gained via one rule is the same when derived by another. A lot of effort was spent trying to strengthen the induction hypothesis or reduce the goal to a simpler case. All of this was in vain though, and it seems that the hypothesis would have to be strengthened as far as to include the concept of integration in the first place, which in a way defeats the advantages of the approach.

3.2 The three-step approach

theory *Integral*
imports *RealRandVar*
begin

Having learnt from my failures, we take the safe and clean way of Heinz Bauer [1]. It proceeds as outlined in the introduction. In three steps, we fix the integral for elementary ("step-")functions, for limits of these, and finally for differences between such limits.

3.2.1 Simple functions

A simple function is a finite sum of characteristic functions, each multiplied with a nonnegative constant. These functions must be parametrized by

measurable sets. Note that to check this condition, a tuple consisting of a set of measurable sets and a measure is required as the integral operator's second argument, whereas the measure only is given in informal notation. Usually the tuple will be a measure space, though it is not required so by the definition at this point.

It is most natural to declare the value of the integral in this elementary case by simply replacing the characteristic functions with the measures of their respective sets. Uniqueness remains to be shown, for a function may have infinitely many decompositions and these might give rise to more than one integral value. This is why we construct a *simple function integral set* for any function and measurable sets/measure pair by means of an inductive set definition containing but one introduction rule.

inductive-set
sfis:: $('a \Rightarrow real) \Rightarrow ('a\ set\ set * ('a\ set \Rightarrow real)) \Rightarrow real\ set$
for $f :: 'a \Rightarrow real$ **and** $M :: 'a\ set\ set * ('a\ set \Rightarrow real)$
where
base: $[\![f = (\lambda t.\ \sum i{\in}(S{::}nat\ set).\ x\ i * \chi\ (A\ i)\ t);$
$\forall i \in S.\ A\ i \in measurable\text{-}sets\ M;\ nonnegative\ x;\ finite\ S;$
$\forall i{\in}S.\ \forall j{\in}S.\ i \neq j \longrightarrow A\ i \cap A\ j = \{\};\ (\bigcup i{\in}S.\ A\ i) = UNIV]\!]$
$\Longrightarrow (\sum i{\in}S.\ x\ i * measure\ M\ (A\ i)) \in sfis\ f\ M$

As you can see we require two extra conditions, and they amount to the sets being a partition of the universe. We say that a function is in normal form if it is represented this way. Normal forms are only needed to show additivity and monotonicity of simple function integral sets. These theorems can then be used in turn to get rid of the normality condition.

More precisely, normal forms play a central role in the *sfis-present* lemma. For two simple functions with different underlying partitions it states the existence of a common finer-grained partition that can be used to represent the functions uniformly. The proof is remarkably lengthy, another case where informal reasoning is more intricate than it seems. The reason it is included anyway, with the exception of the two following lemmata, is that it gives insight into the arising complication and its formal solution.

The problem is in the use of informal sum notation, which easily permits for a change in index sets, allowing for a pair of indices. This change has to be rectified in formal reasoning. Luckily, the task is eased by an injective function from $\mathbb{N}^2$ into $\mathbb{N}$, which was developed for the rationals mentioned in 2.2. It might have been still easier if index sets were polymorphic in our integral definition, permitting pairs to be formed when necessary, but the logic doesn't allow for this.

lemma assumes *un*: $(\bigcup i{\in}R.\ B\ i) = UNIV$ **and** *fin*: *finite R*
and *dis*: $\forall j1{\in}R.\ \forall j2{\in}R.\ j1 \neq j2 \longrightarrow (B\ j1) \cap (B\ j2) = \{\}$
shows *char-split*: $\chi\ A\ t = (\sum j{\in}R.\ \chi\ (A \cap B\ j)\ t)$**lemma assumes** *measure-space* M **and** $a \in sfis\ f\ M$ **and** $b \in sfis\ g\ M$

```
  shows sfis-present: ∃ z1 z2 C K.
  f = (λt. ∑ i∈(K::nat set). z1 i * χ (C i) t) ∧ g = (λt. ∑ i∈K. z2 i * χ (C i)
t)
  ∧ a = (∑ i∈K. z1 i * measure M (C i)) ∧ b = (∑ i∈K. z2 i * measure M (C
i))
  ∧ finite K ∧ (∀ i∈K. ∀ j∈K. i ≠ j ⟶ C i ∩ C j = {})
  ∧ (∀ i ∈ K. C i ∈ measurable-sets M) ∧ (⋃ i∈K. C i) = UNIV
  ∧ nonnegative z1 ∧ nonnegative z2
  using a
proof cases
  case (base x A R)
  note base-x = this
  show ?thesis using b
  proof cases
    case (base y B S)

    with assms base-x have ms: measure-space M
      and f: f = (λt. ∑ i∈(R::nat set). x i * χ (A i) t)
      and a: a = (∑ i∈R. x i * measure M (A i))
      and Ams: ∀ i ∈ R. A i ∈ measurable-sets M
      and R: finite R and Adis: ∀ i∈R. ∀ j∈R. i ≠ j ⟶ A i ∩ A j = {}
      and Aun: (⋃ i∈R. A i) = UNIV
      and g: g = (λt. ∑ i∈(S::nat set). y i * χ (B i) t)
      and b: b = (∑ j∈S. y j * measure M (B j))
      and Bms: ∀ i ∈ S. B i ∈ measurable-sets M
      and S: finite S
      and Bdis: ∀ i∈S. ∀ j∈S. i ≠ j ⟶ B i ∩ B j = {}
      and Bun: (⋃ i∈S. B i) = UNIV
      and x: nonnegative x and y: nonnegative y
      by simp-all

    def C ≡ (λ(i,j). A i ∩ B j) ∘ prod-decode
    def z1 ≡ (λk. x (fst (prod-decode k)))
    def z2 ≡ (λk. y (snd (prod-decode k)))
    def K ≡ {k. ∃ i∈R. ∃ j∈S. k = prod-encode (i,j)}
    def G ≡ (λi. (λj. prod-encode (i,j)) ‘ S)
    def H ≡ (λj. (λi. prod-encode (i,j)) ‘ R)

    { fix t
      { fix i
        from Bun S Bdis have χ (A i) t = (∑ j∈S. χ (A i ∩ B j) t)
          by (rule char-split)
        hence x i * χ (A i) t = (∑ j∈S. x i * χ (A i ∩ B j) t)
          by (simp add: sum-distrib-left)
        also
        { fix j
          have S=S and
            x i * χ (A i ∩ B j) t = (let k=prod-encode(i,j) in z1 k * χ (C k) t)
            by (auto simp add: C-def z1-def Let-def)
        }
```

```
  hence ... = (∑j∈S. let k=prod-encode (i,j) in z1 k * χ (C k) t)
    by (rule sum.cong)

  also from S have ... = (∑k∈(G i). z1 k * χ (C k) t)
    by (simp add: G-def Let-def o-def
          sum.reindex[OF subset-inj-on[OF prod-encode-snd-inj]])

  finally have eq: x i * χ (A i) t = (∑k∈ G i. z1 k * χ (C k) t) .


  from S have G: finite (G i)
    by (simp add: G-def)

  { fix k assume k ∈ G i
    then obtain j where kij: k=prod-encode (i,j)
      by (auto simp only: G-def)
    {
      fix i2 assume i2: i2 ≠ i

      { fix k2 assume k2 ∈ G i2
        then obtain j2 where kij2: k2=prod-encode (i2,j2)
          by (auto simp only: G-def)

        from i2 have (i2,j2) ≠ (i,j) and (i2,j2) ∈ UNIV
          and (i,j) ∈ UNIV by auto
        with inj-prod-encode have  prod-encode (i2,j2) ≠ prod-encode (i,j)
          by (rule inj-on-contraD)
        with kij kij2 have k2 ≠ k
          by fast
      }
      hence k ∉ G i2
        by fast
    }
  }
  hence ⋀j. i ≠ j ⟹ G i ∩ G j = {}
    by fast
  note eq G this
}
hence eq: ⋀i. x i * χ (A i) t = (∑k∈G i. z1 k * χ (C k) t)
  and G: ⋀i. finite (G i)
  and Gdis: ⋀i j. i ≠ j ⟹ G i ∩ G j = {} .

{ fix i
  assume i∈R
  with ms Bun S Bdis Bms Ams have
    measure M (A i) = (∑j∈S. measure M (A i ∩ B j))
    by (simp add: measure-split)
  hence x i * measure M (A i) = (∑j∈S. x i * measure M (A i ∩ B j))
    by (simp add: sum-distrib-left)
```

```
  also
  { fix j
    have S=S and x i * measure M (A i ∩ B j) =
      (let k=prod-encode(i,j) in z1 k * measure M (C k))
      by (auto simp add: C-def z1-def Let-def)
  }

  hence ... = (∑j∈S. let k=prod-encode (i,j) in z1 k * measure M (C k))
    by (rule sum.cong)

  also from S have ... = (∑k∈(G i). z1 k * measure M (C k))
    by (simp add: G-def Let-def o-def
          sum.reindex[OF subset-inj-on[OF prod-encode-snd-inj]])

  finally have
    x i * measure M (A i) = (∑k∈(G i). z1 k * measure M (C k)) .
}
with refl[of R] have
  (∑i∈R. x i * measure M (A i))
  = (∑i∈R. (∑k∈(G i). z1 k * measure M (C k)))
  by (rule sum.cong)
with eq f a have f t = (∑i∈R. (∑k∈G i. z1 k * χ (C k) t))
  and a = (∑i∈R. (∑k∈(G i). z1 k * measure M (C k)))
  by auto
also have KG: K = (⋃i∈R. G i)
  by (auto simp add: K-def G-def)
moreover note G Gdis R
ultimately have f: f t = (∑k∈K. z1 k * χ (C k) t)
  and a: a = (∑k∈K. z1 k * measure M (C k))
  by (auto simp add: sum.UNION-disjoint)

{ fix j
  from Aun R Adis have χ (B j) t = (∑i∈R. χ (B j ∩ A i) t)
    by (rule char-split)
  hence y j * χ (B j) t = (∑i∈R. y j * χ (A i ∩ B j) t)
    by (simp add: sum-distrib-left Int-commute)
  also
  { fix i
    have R=R and
      y j * χ (A i ∩ B j) t = (let k=prod-encode(i,j) in z2 k * χ (C k) t)
      by (auto simp add: C-def z2-def Let-def)
  }
  hence ... = (∑i∈R. let k=prod-encode (i,j) in z2 k * χ (C k) t)
    by (rule sum.cong)
  also from R have ... = (∑k∈(H j). z2 k * χ (C k) t)
    by (simp add: H-def Let-def o-def
          sum.reindex[OF subset-inj-on[OF prod-encode-fst-inj]])
  finally have eq: y j * χ (B j) t = (∑k∈ H j. z2 k * χ (C k) t) .
```

```
  from R have H: finite (H j)  by (simp add: H-def)

  { fix k assume k ∈ H j
    then obtain i where kij: k=prod-encode (i,j)
      by (auto simp only: H-def)
    { fix j2 assume j2: j2 ≠ j
      { fix k2 assume k2 ∈ H j2
        then obtain i2 where kij2: k2=prod-encode (i2,j2)
          by (auto simp only: H-def)

       from j2 have (i2,j2) ≠ (i,j) and (i2,j2) ∈ UNIV and (i,j) ∈ UNIV
          by auto
        with inj-prod-encode have  prod-encode (i2,j2) ≠ prod-encode (i,j)
          by (rule inj-on-contraD)
        with kij kij2 have k2 ≠ k
          by fast
      }
      hence k ∉ H j2
        by fast
    }
  }
  hence ⋀i. i ≠ j ⟹  H i ∩ H j = {}
    by fast
  note eq H this
}
hence eq: ⋀j.  y j * χ (B j) t = (∑ k∈H j. z2 k * χ (C k) t)
  and H: ⋀i. finite (H i)
  and Hdis: ⋀i j. i ≠ j ⟹ H i ∩ H j = {} .
from eq g have g t = (∑ j∈S. (∑ k∈H j. z2 k * χ (C k) t))
  by simp
also
{ fix j assume jS: j∈S
  from ms Aun R Adis Ams Bms jS have measure M (B j) =
    (∑ i∈R. measure M (B j ∩ A i))
    by (simp add: measure-split)
  hence y j * measure M (B j) = (∑ i∈R. y j * measure M (A i ∩ B j))
    by (simp add: sum-distrib-left Int-commute)
  also
  { fix i
    have R=R and y j * measure M (A i ∩ B j) =
      (let k=prod-encode(i,j) in z2 k * measure M (C k))
      by (auto simp add: C-def z2-def Let-def)
  }
  hence ... = (∑ i∈R. let k=prod-encode(i,j) in z2 k * measure M (C k))
    by (rule sum.cong)
  also from R have ... = (∑ k∈(H j). z2 k * measure M (C k))
    by (simp add: H-def Let-def o-def
          sum.reindex[OF subset-inj-on[OF prod-encode-fst-inj]])
```

```
      finally have eq2:
        y j * measure M (B j) = (∑ k∈(H j). z2 k * measure M (C k)) .
    }
    with refl have (∑ j∈S. y j * measure M (B j)) = (∑ j∈S. (∑ k∈(H j). z2
k * measure M (C k)))
      by (rule sum.cong)
    with b have b = (∑ j∈S. (∑ k∈(H j). z2 k * measure M (C k)))
      by simp
    moreover have K = (⋃ j∈S. H j)
      by (auto simp add: K-def H-def)
    moreover note H Hdis S
    ultimately have g: g t = (∑ k∈K. z2 k * χ (C k) t) and K: finite K
      and b: b = (∑ k∈K. z2 k * measure M (C k))
      by (auto simp add: sum.UNION-disjoint)

    { fix i
      from Bun have (⋃ k∈G i. C k) = A i
        by (simp add: G-def C-def)
    }
    with Aun have (⋃ i∈R. (⋃ k∈G i. C k)) = UNIV
      by simp
    hence (⋃ k∈(⋃ i∈R. G i). C k) = UNIV
      by simp
    with KG have Kun: (⋃ k∈K. C k) = UNIV
      by simp

    note f g a b Kun K
  }

  hence f: f = (λt. (∑ k∈K. z1 k * χ (C k) t))
    and g: g = (λt. (∑ k∈K. z2 k * χ (C k) t))
    and a: a = (∑ k∈K. z1 k * measure M (C k))
    and b: b = (∑ k∈K. z2 k * measure M (C k))
    and Kun: UNION K C = UNIV and K: finite K
    by (auto simp add: ext)

  note f g a b K
  moreover
  { fix k1 k2 assume k1∈K and k2∈K and diff: k1 ≠ k2
    with K-def obtain i1 j1 i2 j2 where
      RS: i1 ∈ R ∧ i2 ∈ R ∧ j1 ∈ S ∧ j2 ∈ S
      and k1: k1 = prod-encode (i1,j1) and k2: k2 = prod-encode (i2,j2)
      by auto

    with diff have (i1,j1) ≠ (i2,j2)
      by auto

    with RS Adis Bdis k1 k2 have C k1 ∩ C k2 = {}
      by (simp add: C-def) fast
  }
```

```
    moreover
    { fix k assume k ∈ K
      with K-def obtain i j where R: i ∈ R and S: j ∈ S
        and k: k = prod-encode (i,j)
        by auto
      with Ams Bms have A i ∈ measurable-sets M and B j ∈ measurable-sets M
        by auto
      with ms have A i ∩ B j ∈ measurable-sets M
        by (simp add: measure-space-def sigma-algebra-inter)
      with k have C k ∈ measurable-sets M
        by (simp add: C-def)
    }
    moreover note Kun

    moreover from x have nonnegative z1
      by (simp add: z1-def nonnegative-def)
    moreover from y have nonnegative z2
      by (simp add: z2-def nonnegative-def)
    ultimately show ?thesis by blast
  qed
qed
```

Additivity and monotonicity are now almost obvious, the latter trivially implying uniqueness.

```
lemma assumes ms: measure-space M and a: a ∈ sfis f M and b: b ∈ sfis g M
  shows sfis-add: a+b ∈ sfis (λw. f w + g w) M
proof −
  from assms have
    ∃ z1 z2 C K. f = (λt. ∑ i∈(K::nat set). z1 i * χ (C i) t) ∧
    g = (λt. ∑ i∈K. z2 i * χ (C i) t) ∧ a = (∑ i∈K. z1 i * measure M (C i))
    ∧ b = (∑ i∈K. z2 i * measure M (C i))
    ∧ finite K ∧ (∀ i∈K. ∀ j∈K. i ≠ j ⟶ C i ∩ C j = {})
    ∧ (∀ i ∈ K. C i ∈ measurable-sets M) ∧ (⋃ i∈K. C i) = UNIV
    ∧ nonnegative z1 ∧ nonnegative z2
    by (rule sfis-present)

  then obtain z1 z2 C K where f: f = (λt. ∑ i∈(K::nat set). z1 i * χ (C i) t)
    and g: g = (λt. ∑ i∈K. z2 i * χ (C i) t)
    and a2: a = (∑ i∈K. z1 i * measure M (C i))
    and b2: b = (∑ i∈K. z2 i * measure M (C i))
    and CK: finite K ∧ (∀ i∈K. ∀ j∈K. i ≠ j ⟶ C i ∩ C j = {}) ∧
    (∀ i∈K. C i ∈ measurable-sets M) ∧ UNION K C = UNIV
    and z1: nonnegative z1 and z2: nonnegative z2
    by auto

  { fix t
    from f g have
      f t + g t = (∑ i∈K. z1 i * χ (C i) t) + (∑ i∈K. z2 i * χ (C i) t)
      by simp
```

```
    also have ... = (∑ i∈K. z1 i * χ (C i) t + z2 i * χ (C i) t)
      by (rule sum.distrib[THEN sym])
    also have ... = (∑ i∈K. (z1 i + z2 i) * χ (C i) t)
      by (simp add: distrib-right)
    finally have f t + g t = (∑ i∈K. (z1 i + z2 i) * χ (C i) t) .
  }

  also
  { fix t
    from z1 have 0 ≤ z1 t
      by (simp add: nonnegative-def)
    also from z2 have 0 ≤ z2 t
      by (simp add: nonnegative-def)
    ultimately have 0 ≤ z1 t + z2 t
      by (rule add-nonneg-nonneg)
  }

  hence nonnegative (λw. z1 w + z2 w)
    by (simp add: nonnegative-def ext)
  moreover note CK
  ultimately have
    (∑ i∈K. (z1 i + z2 i) * measure M (C i)) ∈ sfis (λw. f w + g w) M
    by (auto simp add: sfis.base)
  also
  from a2 b2 have a+b = (∑ i∈K. (z1 i + z2 i) * measure M (C i))
    by (simp add: sum.distrib[THEN sym] distrib-right)
  ultimately show ?thesis by simp
qed

lemma assumes ms: measure-space M and a: a ∈ sfis f M
  and b: b ∈ sfis g M and fg: f≤g
  shows sfis-mono: a ≤ b
proof −

  from ms a b have
    ∃ z1 z2 C K. f = (λt. ∑ i∈(K::nat set). z1 i * χ (C i) t) ∧
    g = (λt. ∑ i∈K. z2 i * χ (C i) t) ∧ a = (∑ i∈K. z1 i * measure M (C i))
    ∧ b = (∑ i∈K. z2 i * measure M (C i))
    ∧ finite K ∧ (∀ i∈K. ∀ j∈K. i ≠ j ⟶ C i ∩ C j = {})
    ∧ (∀ i ∈ K. C i ∈ measurable-sets M) ∧ (⋃ i∈K. C i) = UNIV
    ∧ nonnegative z1 ∧ nonnegative z2
    by (rule sfis-present)

  then obtain z1 z2 C K where f: f = (λt. ∑ i∈(K::nat set). z1 i * χ (C i) t)
    and g: g = (λt. ∑ i∈K. z2 i * χ (C i) t)
    and a2: a = (∑ i∈K. z1 i * measure M (C i))
    and b2: b = (∑ i∈K. z2 i * measure M (C i))
    and K: finite K and dis: (∀ i∈K. ∀ j∈K. i ≠ j ⟶ C i ∩ C j = {})
    and Cms: (∀ i∈K. C i ∈ measurable-sets M) and Cun: UNION K C = UNIV
    by auto
```

```
{ fix i assume iK: i ∈ K
  { assume C i ≠ {}
    then obtain t where ti: t ∈ C i
      by auto
    hence z1 i = z1 i * χ (C i) t
      by (simp add: characteristic-function-def)
    also
    from dis iK ti have K−{i} = K−{i}
      and ⋀x. x ∈ K−{i} ⟹ z1 x * χ (C x) t = 0
      by (auto simp add: characteristic-function-def)
    hence 0 = (∑ k∈K−{i}. z1 k * χ (C k) t)
      by (simp only: sum.neutral-const sum.cong)
    with K iK have z1 i * χ (C i) t = (∑ k∈K. z1 k * χ (C k) t)
      by (simp add: sum-diff1)
    also
    from fg f g have (∑ i∈K. z1 i * χ (C i) t) ≤ (∑ i∈K. z2 i * χ (C i) t)
      by (simp add: le-fun-def)
    also
    from dis iK ti have K−{i} = K−{i}
      and ⋀x. x ∈ K−{i} ⟹ z2 x * χ (C x) t = 0
      by (auto simp add: characteristic-function-def)
    hence 0 = (∑ k∈K−{i}. z2 k * χ (C k) t)
      by (simp only: sum.neutral-const sum.cong)
    with K iK have (∑ k∈K. z2 k * χ (C k) t) = z2 i * χ (C i) t
      by (simp add: sum-diff1)
    also
    from ti have ... = z2 i
      by (simp add: characteristic-function-def)
    finally
    have z1 i ≤ z2 i .
  }
  hence h: C i ≠ {} ⟹ z1 i ≤ z2 i .

  have z1 i * measure M (C i) ≤ z2 i * measure M (C i)
  proof (cases C i ≠ {})
    case False
    with ms show ?thesis
      by (auto simp add: measure-space-def positive-def)

  next
    case True
    with h have z1 i ≤ z2 i
      by fast
    also from iK ms Cms have 0 ≤ measure M (C i)
      by (auto simp add: measure-space-def positive-def )
    ultimately show ?thesis
      by (simp add: mult-right-mono)
  qed
```

```
  }
  with a2 b2 show ?thesis by (simp add: sum-mono)
qed

lemma sfis-unique:
  assumes ms: measure-space M and a: a ∈ sfis f M and b: b ∈ sfis f M
  shows a=b
proof −
  have f≤f by (simp add: le-fun-def)
  with assms have a≤b and b≤a
    by (auto simp add: sfis-mono)
  thus ?thesis by simp
qed
```

The integral of characteristic functions, as well as the effect of multiplication with a constant, follows directly from the definition. Together with a generalization of the addition theorem to sums, a less restrictive introduction rule emerges, making normal forms obsolete. It is only valid in measure spaces though.

```
lemma sfis-char:
  assumes ms: measure-space M and mA: A ∈ measurable-sets M
  shows measure M A ∈ sfis χ A M

lemma sfis-times:
  assumes a: a ∈ sfis f M and z: 0≤z
  shows z∗a ∈ sfis (λw. z∗f w) M

lemma assumes ms: measure-space M
  and a: ∀ i∈S. a i ∈ sfis (f i) M and S: finite S
  shows sfis-sum: (∑ i∈S. a i) ∈ sfis (λt. ∑ i∈S. f i t) M

lemma sfis-intro:
  assumes ms: measure-space M and Ams: ∀ i ∈ S. A i ∈ measurable-sets M
  and nn: nonnegative x and S: finite S
  shows (∑ i∈S. x i ∗ measure M (A i)) ∈
  sfis (λt. ∑ i∈(S::nat set). x i ∗ χ (A i) t) M
proof −
  { fix i assume iS: i ∈ S
    with ms Ams have measure M (A i) ∈ sfis χ (A i) M
      by (simp add: sfis-char)
    with nn have x i ∗ measure M (A i) ∈ sfis (λt. x i ∗ χ (A i) t) M
      by (simp add: nonnegative-def sfis-times)
  }
  with ms S show ?thesis
    by (simp add: sfis-sum)
qed
```

That is nearly all there is to know about simple function integral sets. It will be useful anyway to have the next two facts available.

lemma *sfis-nn*:
assumes *f*: $a \in$ *sfis f M*
shows *nonnegative f*
lemma *sum-rv*:
assumes *rvs*: $\forall\, k{\in}K.\ (f\ k) \in rv\ M$ **and** *ms*: *measure-space M*
shows $(\lambda t.\ \sum k{\in}K.\ f\ k\ t) \in rv\ M$
lemma *sfis-rv*:
assumes *ms*: *measure-space M* **and** *f*: $a \in$ *sfis f M*
shows $f \in rv\ M$ **using** *f*
proof (*cases*)
case (*base x A S*)
hence $f = (\lambda t.\ \sum i{\in}S.\ x\ i * \chi\ (A\ i)\ t)$
by *simp*
also
{ fix *i*
assume $i \in S$
with *base* **have** $A\ i \in$ *measurable-sets M*
by *simp*
with *ms* **have** $(\lambda t.\ x\ i * \chi\ (A\ i)\ t) \in rv\ M$
by (*simp add*: *char-rv const-rv rv-times-rv*)
} with *ms*
have $\ldots \in rv\ M$
by (*simp add*: *sum-rv*)
ultimately show *?thesis*
by *simp*
qed

3.2.2 Nonnegative Functions

There is one more important fact about *sfis*, easily the hardest one to see. It is about the relationship with monotone convergence and paves the way for a sensible definition of *nnfis*, the nonnegative function integral sets, enabling monotonicity and thus uniqueness. A reasonably concise formal proof could fortunately be achieved in spite of the nontrivial ideas involved — compared for instance to the intuitive but hard-to-formalize *sfis-present*. A small lemma is needed to ensure that the inequation, which depends on an arbitrary z strictly between 0 and 1, carries over to $z = 1$, thereby eliminating z in the end.

lemma *real-le-mult-sustain*:
assumes *zr*: $\bigwedge z.\ [\![0{<}z;\ z{<}1]\!] \Longrightarrow z * r \leq y$
shows $r \leq (y{::}real)$
lemma *sfis-mon-conv-mono*:
assumes *uf*: $u{\uparrow}f$ **and** *xu*: $\bigwedge n.\ x\ n \in$ *sfis* $(u\ n)\ M$ **and** *xy*: $x{\uparrow}y$
and *sr*: $r \in$ *sfis s M* **and** *sf*: $s \leq f$ **and** *ms*: *measure-space M*
shows $r \leq y$ **using** *sr*
proof *cases*
case (*base a A S*)
note *base-a* = *this*

```
{ fix z assume znn: 0<(z::real) and z1: z<1
  def B ≡ (λn. {w. z*s w ≤ u n w})

  { fix n
    note ms also
    from xu have xu: x n ∈ sfis (u n) M .
    hence nnu: nonnegative (u n)
      by (rule sfis-nn)
    from ms xu have u n ∈ rv M
      by (rule sfis-rv)
    moreover from ms sr have s ∈ rv M
      by (rule sfis-rv)
    with ms have (λw. z*s w) ∈ rv M
      by (simp add: const-rv rv-times-rv)
    ultimately have B n ∈ measurable-sets M
      by (simp add: B-def rv-le-rv-measurable)
    with ms base have ABms: ∀ i∈S. (A i ∩ B n) ∈ measurable-sets M
      by (auto simp add: measure-space-def sigma-algebra-inter)

    from xu have z*(∑ i∈S. a i * measure M (A i ∩ B n)) ≤ x n
    proof (cases)
      case (base c C R)
      { fix t
        { fix i
          have S=S and a i * χ (A i ∩ B n) t = χ (B n) t * (a i * χ (A i) t)
            by (auto simp add: characteristic-function-def) }
        hence (∑ i∈S. a i * χ (A i ∩ B n) t) =
          (∑ i∈S. χ (B n) t * (a i * χ (A i) t))
          by (rule sum.cong)
        hence z*(∑ i∈S. a i * χ (A i ∩ B n) t) =
          z*(∑ i∈S. χ (B n) t * (a i * χ (A i) t))
          by simp
        also have ... = z * χ (B n) t * (∑ i∈S. a i * χ (A i) t)
          by (simp add: sum-distrib-left[THEN sym])
        also
        from sr have nonnegative s by (simp add: sfis-nn)
        with nnu B-def base-a
        have z * χ (B n) t * (∑ i∈S. a i * χ (A i) t) ≤ u n t
          by (auto simp add: characteristic-function-def nonnegative-def)
        finally have z*(∑ i∈S. a i * χ (A i ∩ B n) t) ≤ u n t .
      }

      also
      from ms base-a znn ABms have
        z*(∑ i∈S. a i * measure M (A i ∩ B n)) ∈
        sfis (λt. z*(∑ i∈S. a i * χ (A i ∩ B n) t)) M
        by (simp add: sfis-intro sfis-times)
      moreover note xu ms
```

```
    ultimately show ?thesis
      by (simp add: sfis-mono le-fun-def)
  qed
  note this ABms
}
hence 1: ⋀n. z * (∑ i∈S. a i * measure M (A i ∩ B n)) ≤ x n
  and ABms: ⋀n. ∀ i∈S. A i ∩ B n ∈ measurable-sets M .

have Bun: (λn. B n)↑UNIV
proof (unfold mon-conv-set-def, rule)
  { fix n
    from uf have um: u n ≤ u (Suc n)
      by (simp add: mon-conv-real-fun-def)
    {
      fix w
      assume z*s w ≤ u n w
      also from um have u n w ≤ u (Suc n) w
        by (simp add: le-fun-def)
      finally have z*s w ≤ u (Suc n) w .
    }
    hence B n ≤ B (Suc n)
      by (auto simp add: B-def)
  }
  thus  ∀ n. B n ⊆ B (Suc n)
    by fast

  { fix t
    have ∃ n. z*s t ≤ u n t
    proof (cases s t = 0)
      case True
      fix n
      from True have z*s t = 0
        by simp
      also from xu have nonnegative (u n)
        by (rule sfis-nn)
      hence 0 ≤ u n t
        by (simp add: nonnegative-def)
      finally show ?thesis
        by rule

    next
      case False
      from sr have nonnegative s
        by(rule sfis-nn)
      hence 0 ≤ s t
        by (simp add: nonnegative-def)
      with False have 0 < s t
        by arith
      with z1 have z*s t < 1*s t
```

```
        by (simp only: mult-strict-right-mono)
      also from sf have ... ≤ f t
        by (simp add: le-fun-def)
      finally have z * s t < f t .

      also from uf have (λm. u m t)↑f t
        by (simp add: realfun-mon-conv-iff)
      ultimately have ∃n.∀m. n≤m ⟶ z*s t < u m t
        by (simp add: real-mon-conv-outgrow)
      hence ∃n. z*s t < u n t
        by fast
      thus ?thesis
        by (auto simp add: order-less-le)
    qed

    hence ∃n. t ∈ B n
      by (simp add: B-def)
    hence t ∈ (⋃n. B n)
      by fast
  }
  thus UNIV=(⋃n. B n)
    by fast
qed

{ fix j assume jS: j ∈ S
  note ms
  also
  from jS ABms have ⋀n. A j ∩ B n ∈ measurable-sets M
    by auto
  moreover
  from Bun have (λn. A j ∩ B n)↑(A j)
    by (auto simp add: mon-conv-set-def)
  ultimately have (λn. measure M (A j ∩ B n)) ⟶⟶ measure M (A j)
    by (rule measure-mon-conv)

  hence (λn. a j * measure M (A j ∩ B n)) ⟶⟶ a j * measure M (A j)
    by (simp add: tendsto-const tendsto-mult)
}
hence (λn. ∑j∈S. a j * measure M (A j ∩ B n))
  ⟶⟶ (∑j∈S. a j * measure M (A j))
  by (rule tendsto-sum)
hence (λn. z* (∑j∈S. a j * measure M (A j ∩ B n)))
  ⟶⟶ z*(∑j∈S. a j * measure M (A j))
  by (simp add: tendsto-const tendsto-mult)

with 1 xy base have z*r ≤ y
  by (auto simp add: LIMSEQ-le mon-conv-real-def)
}
hence zr: ⋀z. 0 < z ⟹ z < 1 ⟹ z * r ≤ y .
```

thus *?thesis* **by** (*rule real-le-mult-sustain*)
qed

Now we are ready for the second step. The integral of a monotone limit of functions is the limit of their integrals. Note that this last limit has to exist in the first place, since we decided not to use infinite values. Backed by the last theorem and the preexisting knowledge about limits, the usual basic properties are straightforward.

inductive-set
nnfis:: ('*a* ⇒ *real*) ⇒ ('*a set set* ∗ ('*a set* ⇒ *real*)) ⇒ *real set*
for *f* :: '*a* ⇒ *real* **and** *M* :: '*a set set* ∗ ('*a set* ⇒ *real*)
where
base: ⟦*u*↑*f*; ⋀*n*. *x n* ∈ *sfis* (*u n*) *M*; *x*↑*y*⟧ ⟹ *y* ∈ *nnfis f M*

lemma *sfis-nnfis*:
assumes *s*: *a* ∈ *sfis f M*
shows *a* ∈ *nnfis f M*

lemma *nnfis-times*:
assumes *ms*: *measure-space M* **and** *a*: *a* ∈ *nnfis f M* **and** *nn*: *0*≤*z*
shows *z*∗*a* ∈ *nnfis* (λ*w*. *z*∗*f w*) *M*

lemma *nnfis-add*:
assumes *ms*: *measure-space M* **and** *a*: *a* ∈ *nnfis f M* **and** *b*: *b* ∈ *nnfis g M*
shows *a*+*b* ∈ *nnfis* (λ*w*. *f w* + *g w*) *M*

lemma assumes *ms*: *measure-space M* **and** *a*: *a* ∈ *nnfis f M*
and *b*: *b* ∈ *nnfis g M* **and** *fg*: *f*≤*g*
shows *nnfis-mono*: *a* ≤ *b* **using** *a*
proof (*cases*)
case (*base u x*)
note *base-u* = *this*
from *b* **show** *?thesis*
proof (*cases*)
case (*base v r*)
{ fix *m*
from *base-u base* **have** *u m* ≤ *f*
by (*simp add*: *realfun-mon-conv-le*)
also note *fg* **finally have** *u m* ≤ *g* **.**
with *ms base-u base* **have** *v*↑*g* **and** ⋀*n*. *r n* ∈ *sfis* (*v n*) *M* **and** *r*↑*b*
and *x m* ∈ *sfis* (*u m*) *M* **and** *u m* ≤ *g* **and** *measure-space M*
by *simp-all*
hence *x m* ≤ *b*
by (*rule sfis-mon-conv-mono*)
}
with *ms base-u base* **show** *a* ≤ *b*
by (*auto simp add*: *mon-conv-real-def LIMSEQ-le-const2*)
qed
qed

corollary *nnfis-unique*:
assumes *ms*: *measure-space M* **and** *a*: $a \in$ *nnfis f M* **and** *b*: $b \in$ *nnfis f M*
shows $a{=}b$

There is much more to prove about nonnegative integration. Next up is a classic theorem by Beppo Levi, the monotone convergence theorem. In essence, it says that the introduction rule for *nnfis* holds not only for sequences of simple functions, but for any sequence of nonnegative integrable functions. It should be mentioned that this theorem cannot be formulated for the Riemann integral. We prove it by exhibiting a sequence of simple functions that converges to the same limit as the original one and then applying the introduction rule.

The construction and properties of the sequence are slightly intricate. By definition, for any f_n in the original sequence, there is a sequence $(u_{mn})_{m\in\mathbb{N}}$ of simple functions converging to it. The nth element of the new sequence is the upper closure of the nth elements of the first n sequences.

definition
upclose:: $('a \Rightarrow real) \Rightarrow ('a \Rightarrow real) \Rightarrow ('a \Rightarrow real)$ **where**
$upclose\ f\ g = (\lambda t.\ max\ (f\ t)\ (g\ t))$

primrec
mon-upclose-help :: $nat \Rightarrow (nat \Rightarrow nat \Rightarrow 'a \Rightarrow real) \Rightarrow nat \Rightarrow ('a \Rightarrow real)\ (muh)$
where
$muh\ 0\ u\ m = u\ m\ 0$
| $muh\ (Suc\ n)\ u\ m = upclose\ (u\ m\ (Suc\ n))\ (muh\ n\ u\ m)$

definition
mon-upclose :: $(nat \Rightarrow nat \Rightarrow 'a \Rightarrow real) \Rightarrow nat \Rightarrow ('a \Rightarrow real)\ (mu)$ **where**
$mu\ u\ m = muh\ m\ u\ m$

lemma *sf-norm-help*:
assumes *fin*: *finite K* **and** *jK*: $j \in K$ **and** *tj*: $t \in C\ j$ **and** *iK*: $\forall\, i{\in}K{-}\{j\}.\ t \notin C\ i$
shows $(\sum i{\in}K.\ (z\ i) * \chi\ (C\ i)\ t) = z\ j$
lemma *upclose-sfis*:
assumes *ms*: *measure-space M* **and** *f*: $a \in$ *sfis f M* **and** *g*: $b \in$ *sfis g M*
shows $\exists\, c.\ c \in sfis\ (upclose\ f\ g)\ M$
lemma *mu-sfis*:
assumes *u*: $\bigwedge m\ n.\ \exists\, a.\ a \in sfis\ (u\ m\ n)\ M$ **and** *ms*: *measure-space M*
shows $\exists\, c.\ \forall\, m.\ c\ m \in sfis\ (mu\ u\ m)\ M$

lemma *mu-help*:
assumes *uf*: $\bigwedge n.\ (\lambda m.\ u\ m\ n)\uparrow(f\ n)$ **and** *fh*: $f\uparrow h$
shows $(mu\ u)\uparrow h$ **and** $\bigwedge n.\ mu\ u\ n \le f\ n$
proof −
show *mu-le*: $\bigwedge n.\ mu\ u\ n \le f\ n$
proof (*unfold mon-upclose-def*)

```
  fix n
  show ⋀m. muh n u m ≤ f n
  proof (induct n)
    case (0 m)
    from uf have u m 0 ≤ f 0
      by (rule realfun-mon-conv-le)
    thus ?case by simp
  next
    case (Suc n m)
    { fix t
      from Suc have muh n u m t ≤ f n t
        by (simp add: le-fun-def)
      also from fh have f n t ≤ f (Suc n) t
        by (simp add: realfun-mon-conv-iff mon-conv-real-def)
      also from uf have (λm. u m (Suc n) t)↑(f (Suc n) t)
        by (simp add: realfun-mon-conv-iff)
      hence u m (Suc n) t ≤ f (Suc n) t
        by (rule real-mon-conv-le)
      ultimately have muh (Suc n) u m t ≤ f (Suc n) t
        by (simp add: upclose-def)
    }
    thus ?case by (simp add: le-fun-def)
  qed
qed

{ fix t
  { fix m n
    have muh n u m t ≤ muh (Suc n) u m t
      by (simp add: upclose-def)
  }
  hence pos1: ⋀m n. muh n u m t ≤ muh (Suc n) u m t .

  from uf have uiso: ⋀m n. u m n t ≤ u (Suc m) n t
    by (simp add: realfun-mon-conv-iff mon-conv-real-def)

  have iso: ⋀n. mu u n t ≤ mu u (Suc n) t
  proof (unfold mon-upclose-def)
    fix n
    have ⋀m. muh n u m t ≤ muh n u (Suc m) t
    proof (induct n)
      case 0 from uiso show ?case
        by (simp add: upclose-def le-max-iff-disj)
    next
      case (Suc n m)

      from Suc have muh n u m t ≤ muh n u (Suc m) t .
      also from uiso have u m (Suc n) t ≤ u (Suc m) (Suc n) t .

      ultimately show ?case
```

```
      by (auto simp add: upclose-def le-max-iff-disj)
  qed
  note this [of n] also note pos1 [of n Suc n]
  finally show muh n u n t ≤ muh (Suc n) u (Suc n) t .
qed

also

{ fix n
  from mu-le [of n]
  have mu u n t ≤ f n t
    by (simp add: le-fun-def)
  also
  from fh have (λn. f n t)↑h t
    by (simp add: realfun-mon-conv-iff)
  hence f n t ≤ h t
    by (rule real-mon-conv-le)
  finally have mu u n t ≤ h t .
}

ultimately have ∃l. (λn. mu u n t)↑l ∧ l ≤ h t
  by (rule real-mon-conv-bound)
then obtain l where
  conv: (λn. mu u n t)↑l and lh: l ≤ h t
  by (force simp add: real-mon-conv-bound)

{ fix n::nat
  { fix m assume le: n ≤ m
    hence u m n t ≤ mu u m t
    proof (unfold mon-upclose-def)
      have u m n t ≤ muh n u m t
        by (induct n) (auto simp add: upclose-def le-max-iff-disj)
      also
      from pos1 have incseq (λn. muh n u m t)
        by (simp add: incseq-Suc-iff)
      hence muh n u m t ≤ muh (n+(m−n)) u m t
        unfolding incseq-def by simp
      with le have muh n u m t ≤ muh m u m t
        by simp
      finally show u m n t ≤ muh m u m t .
    qed
  }
  hence ∃N. ∀m. N ≤ m ⟶ u m n t ≤ mu u m t
    by blast
  also from uf have (λm. u m n t) ⟶⟶ f n t
    by (simp add: realfun-mon-conv-iff mon-conv-real-def)
  moreover
  from conv have (λn. mu u n t) ⟶⟶ l
    by (simp add: mon-conv-real-def)
```

```
      ultimately have f n t ≤ l
        by (simp add: LIMSEQ-le)
    }
    also from fh have (λn. f n t) ⟶ h t
      by (simp add: realfun-mon-conv-iff mon-conv-real-def)
    ultimately have h t ≤ l
      by (simp add: LIMSEQ-le-const2)

    with lh have l = h t
      by simp
    with conv have (λn. mu u n t)↑(h t)
      by simp
  }
  with mon-upclose-def show mu u↑h
    by (simp add: realfun-mon-conv-iff)
qed

theorem nnfis-mon-conv:
  assumes fh: f↑h and xf: ⋀n. x n ∈ nnfis (f n) M and xy: x↑y
  and ms: measure-space M
  shows y ∈ nnfis h M
proof −
  def u ≡ (λn. SOME u. u↑(f n) ∧ (∀ m. ∃ a. a ∈ sfis (u m) M))
  { fix n
    from xf[of n] have ∃ u. u↑(f n) ∧ (∀ m. ∃ a. a ∈ sfis (u m) M) (is ∃ x. ?P x)
    proof (cases)
      case (base r a)
      hence r↑(f n) and ⋀m. ∃ a. a ∈ sfis (r m) M by auto
      thus ?thesis by fast
    qed
    hence ?P (SOME x. ?P x)
      by (rule someI-ex)
    with u-def have ?P (u n)
      by simp
  } also
  def urev ≡ (λm n. u n m)
  ultimately have uf: ⋀n. (λm. urev m n)↑(f n)
    and sf: ⋀n m. ∃ a. a ∈ sfis (urev m n) M
    by auto

  from uf fh have up: mu urev↑h
    by (rule mu-help)
  from uf fh have le: ⋀n. mu urev n ≤ f n
    by (rule mu-help)
  from sf ms obtain c where sf2: ⋀m. c m ∈ sfis (mu urev m) M
    by (force simp add: mu-sfis)

  { fix m
    from sf2 have c m ∈ nnfis (mu urev m) M
```

```
    by (rule sfis-nnfis)
  with ms le[of m] xf[of m] have c m ≤ x m
    by (simp add: nnfis-mono)
 } hence c ≤ x by (simp add: le-fun-def)
 also
 { fix m note ms also
  from up have mu urev m ≤ mu urev (Suc m)
    by (simp add: mon-conv-real-fun-def)
  moreover from sf2 have c m ∈ sfis (mu urev m) M
    and c (Suc m) ∈ sfis (mu urev (Suc m)) M
    by fast+
  ultimately have c m ≤ c (Suc m)
    by (simp add: sfis-mono)
 }
 moreover note xy
 ultimately have ∃ l. c↑l ∧ l ≤ y
  by (simp add: real-mon-conv-dom)
 then obtain l where cl: c↑l and ly: l ≤ y
  by fast

 from up sf2 cl have int: l ∈ nnfis h M
  by (rule nnfis.base)

 { fix n
  from fh have f n ≤ h
    by (rule realfun-mon-conv-le)
  with ms xf[of n] int have x n ≤ l
    by (rule nnfis-mono)
 } with xy have y ≤ l
  by (auto simp add: mon-conv-real-def LIMSEQ-le-const2)

 with ly have l=y
  by simp
 with int show ?thesis
  by simp
qed
```

Establishing that only nonnegative functions may arise this way is a triviality.

```
lemma nnfis-nn: assumes a ∈ nnfis f M
 shows nonnegative f
```

3.2.3 Integrable Functions

Before we take the final step of defining integrability and the integral operator, we should first clarify what kind of functions we are able to integrate up to now. It is easy to see that all nonnegative integrable functions are random variables.

lemma assumes *measure-space M* **and** $a \in$ *nnfis f M*
shows *nnfis-rv*: $f \in$ *rv M*

The converse does not hold of course, since there are measurable functions whose integral is infinite. Regardless, it is possible to approximate any measurable function using simple step-functions. This means that all nonnegative random variables are quasi integrable, as the property is sometimes called, and brings forth the fundamental insight that a nonnegative function is integrable if and only if it is measurable and the integrals of the simple functions that approximate it converge monotonically. Technically, the proof is rather complex, involving many properties of real numbers.

lemma assumes *measure-space M* **and** : $f \in$ *rv M* **and** *nonnegative f*
shows *rv-mon-conv-sfis*: $\exists\, u\ x.\ u{\uparrow}f \wedge (\forall\, n.\ x\ n \in$ *sfis* $(u\ n)\ M)$

The following dominated convergence theorem is an easy corollary. It can be effectively applied to show integrability.

corollary assumes *ms*: *measure-space M* **and** *f*: $f \in$ *rv M*
and *b*: $b \in$ *nnfis g M* **and** *fg*: $f \leq g$ **and** *nn*: *nonnegative f*
shows *nnfis-dom-conv*: $\exists\, a.\ a \in$ *nnfis f M* $\wedge\ a \leq b$ **using** *b*
proof (*cases*)
case (*base v r*)
from *ms f nn* **have** $\exists\, u\ x.\ u{\uparrow}f \wedge (\forall\, n.\ x\ n \in$ *sfis* $(u\ n)\ M)$
by (*rule rv-mon-conv-sfis*)
then obtain *u x* **where** *uf*: $u{\uparrow}f$ **and** *xu*: $\bigwedge n.\ x\ n \in$ *sfis* $(u\ n)\ M$
by *fast*

{ fix *n*
from *uf* **have** $u\ n \leq f$
by (*rule realfun-mon-conv-le*)
also note *fg*
also
from *xu* **have** $x\ n \in$ *nnfis* $(u\ n)\ M$
by (*rule sfis-nnfis*)
moreover note *b ms*
ultimately have *le*: $x\ n \leq b$
by (*simp add*: *nnfis-mono*)

from *uf* **have** $u\ n \leq u\ (Suc\ n)$
by (*simp only*: *mon-conv-real-fun-def*)
with *ms xu*[*of n*] *xu*[*of Suc n*] **have** $x\ n \leq x\ (Suc\ n)$
by (*simp add*: *sfis-mono*)
note *this le*
}
hence $\exists\, a.\ x{\uparrow}a \wedge a \leq b$
by (*rule real-mon-conv-bound*)
then obtain *a* **where** *xa*: $x{\uparrow}a$ **and** *ab*: $a \leq b$
by *auto*

from *uf xu xa* **have** $a \in nnfis\ f\ M$
by (*rule nnfis.base*)
with *ab* **show** *?thesis*
by *fast*
qed

Speaking all the time about integrability, it is time to define it at last.

definition
integrable:: $('a \Rightarrow real) \Rightarrow ('a\ set\ set * ('a\ set \Rightarrow real)) \Rightarrow bool$ **where**

$integrable\ f\ M \longleftrightarrow measure\text{-}space\ M \wedge$
$(\exists x.\ x \in nnfis\ (pp\ f)\ M) \wedge (\exists y.\ y \in nnfis\ (np\ f)\ M)$

definition
integral:: $('a \Rightarrow real) \Rightarrow ('a\ set\ set * ('a\ set \Rightarrow real)) \Rightarrow real$ $(\int\ \text{-}\ \partial\text{-})$ **where**
$integrable\ f\ M \Longrightarrow \int\ f\ \partial M = (THE\ i.\ i \in nnfis\ (pp\ f)\ M) -$
$(THE\ j.\ j \in nnfis\ (np\ f)\ M)$

So the final step is done, the integral defined. The theorems we are already used to prove from the earlier stages are still missing. Only now there are always two properties to be shown: integrability and the value of the integral. Isabelle makes it possible two have both goals in a single theorem, so that the user may derive the statement he desires. Two useful lemmata follow. They help lifting nonnegative function integral sets to integrals proper. Notice how the dominated convergence theorem from above is employed in the latter.

lemma *nnfis-integral*:
assumes *nn*: $a \in nnfis\ f\ M$ **and** *ms*: *measure-space M*
shows *integrable f M* **and** $\int\ f\ \partial\ M = a$
proof −
from *nn* **have** *nonnegative f*
by (*rule nnfis-nn*)
hence $pp\ f = f$ **and** *0*: $np\ f = (\lambda t.\ 0)$
by (*auto simp add*: *nn-pp-np*)
with *nn* **have** *a*: $a \in nnfis\ (pp\ f)\ M$
by *simp*
have $0 \leq (0::real)$
by (*rule order-refl*)
with *ms nn* **have** $0 * a \in nnfis\ (\lambda t.\ 0 * f\ t)\ M$
by (*rule nnfis-times*)
with *0* **have** *02*: $0 \in nnfis\ (np\ f)\ M$
by *simp*
with *ms a* **show** *integrable f M*
by (*auto simp add*: *integrable-def*)
also
from *a ms* **have** $(THE\ i.\ i \in nnfis\ (pp\ f)\ M) = a$
by (*auto simp add*: *nnfis-unique*)
moreover

```
  from 02 ms have (THE i. i ∈ nnfis (np f) M) = 0
    by (auto simp add: nnfis-unique)
  ultimately show ∫ f ∂ M = a
    by (simp add: integral-def)
qed

lemma nnfis-minus-nnfis-integral:
  assumes a: a ∈ nnfis f M and b: b ∈ nnfis g M
  and ms: measure-space M
  shows integrable (λt. f t − g t) M and ∫ (λt. f t − g t) ∂ M = a − b
proof −
  from ms a b have (λt. f t − g t) ∈ rv M
    by (auto simp only: nnfis-rv rv-minus-rv)
  hence prv: pp (λt. f t − g t) ∈ rv M and nrv: np (λt. f t − g t) ∈ rv M
    by (auto simp only: pp-np-rv)

  have nnp: nonnegative (pp (λt. f t − g t))
    and nnn: nonnegative (np (λt. f t − g t))
    by (simp-all add: nonnegative-def positive-part-def negative-part-def)

  from ms a b have fg: a+b ∈ nnfis (λt. f t + g t) M
    by (rule nnfis-add)

  from a b have nnf: nonnegative f and nng: nonnegative g
    by (simp-all only: nnfis-nn)

  { fix t
    from nnf nng have 0 ≤ f t and 0 ≤ g t
      by (simp-all add: nonnegative-def)
    hence (pp (λt. f t − g t)) t ≤ f t + g t and (np (λt. f t − g t)) t ≤ f t + g t
      by (simp-all add: positive-part-def negative-part-def)
  }
  hence (pp (λt. f t − g t)) ≤ (λt. f t + g t)
    and (np (λt. f t − g t)) ≤ (λt. f t + g t)
    by (simp-all add: le-fun-def)
  with fg nnf nng prv nrv nnp nnn ms
  have ∃ l. l ∈ nnfis (pp (λt. f t − g t)) M ∧ l ≤ a+b
    and ∃ k. k ∈ nnfis (np (λt. f t − g t)) M ∧ k ≤ a+b
    by (auto simp only: nnfis-dom-conv)
  then obtain l k where l: l ∈ nnfis (pp (λt. f t − g t)) M
    and k: k ∈ nnfis (np (λt. f t − g t)) M
    by auto
  with ms show i: integrable (λt. f t − g t) M
    by (auto simp add: integrable-def)

  { fix t
    have f t − g t = (pp (λt. f t − g t)) t − (np (λt. f t − g t)) t
      by (rule f-plus-minus)
```

```
    hence f t + (np (λt. f t − g t)) t = g t + (pp (λt. f t − g t)) t
      by arith
  }
  hence (λt. f t + (np (λt. f t − g t)) t) =
    (λt. g t + (pp (λt. f t − g t)) t)
    by (rule ext)
  also
  from ms a k b l have a+k ∈ nnfis (λt. f t + (np (λt. f t − g t)) t) M
    and b+l ∈ nnfis (λt. g t + (pp (λt. f t − g t)) t) M
    by (auto simp add: nnfis-add)
  moreover note ms
  ultimately have a+k = b+l
    by (simp add: nnfis-unique)
  hence l−k=a−b by arith
  also
  from k l ms have (THE i. i ∈ nnfis (pp (λt. f t − g t)) M) = l
    and (THE i. i ∈ nnfis (np (λt. f t − g t)) M) = k
    by (auto simp add: nnfis-unique)
  moreover note i
  ultimately show ∫ (λt. f t − g t) ∂ M = a − b
    by (simp add: integral-def)
qed
```

Armed with these, the standard integral behavior should not be hard to derive. Mind that integrability always implies a measure space, just like random variables did in 2.2.

```
theorem assumes integrable f M
  shows integrable-rv: f ∈ rv M
theorem integral-char:
  assumes ms: measure-space M and mA: A ∈ measurable-sets M
  shows ∫ χ A ∂ M = measure M A and integrable χ A M

theorem integral-add:
  assumes f: integrable f M and g: integrable g M
  shows integrable (λt. f t + g t) M
  and ∫ (λt. f t + g t) ∂M = ∫ f ∂M + ∫ g ∂M
proof −
  def u ≡ (λt. pp f t + pp g t)
  def v ≡ (λt. np f t + np g t)

  from f obtain pf nf where pf: pf ∈ nnfis (pp f) M
    and nf: nf ∈ nnfis (np f) M and ms: measure-space M
    by (auto simp add: integrable-def)
  from g obtain pg ng where pg: pg ∈ nnfis (pp g) M
    and ng: ng ∈ nnfis (np g) M
    by (auto simp add: integrable-def)

  from ms pf pg u-def have
    u: pf+pg ∈ nnfis u M
```

```
    by (simp add: nnfis-add)

  from ms nf ng v-def have
    v: nf+ng ∈ nnfis v M
    by (simp add: nnfis-add)

  { fix t
    from u-def v-def have f t + g t = u t − v t
      by (simp add: positive-part-def negative-part-def)
  }
  hence uvf: (λt. u t − v t) = (λt. f t + g t)
    by (simp add: ext)

  from u v ms have integrable (λt. u t − v t) M
    by (rule nnfis-minus-nnfis-integral)
  with u-def v-def uvf show integrable (λt. f t + g t) M
    by simp

  from pf nf ms have ∫ (λt. pp f t − np f t) ∂M = pf−nf
    by (rule nnfis-minus-nnfis-integral)
  hence ∫ f ∂M = pf−nf
    by (simp add: f-plus-minus[THEN sym])
  also
  from pg ng ms have ∫ (λt. pp g t − np g t) ∂M = pg−ng
    by (rule nnfis-minus-nnfis-integral)
  hence ∫ g ∂M = pg−ng
    by (simp add: f-plus-minus[THEN sym])
  moreover
  from u v ms have ∫ (λt. u t − v t) ∂M = pf + pg − (nf + ng)
    by (rule nnfis-minus-nnfis-integral)
  with uvf have ∫ (λt. f t + g t) ∂M = pf−nf + pg−ng
    by simp
  ultimately
  show ∫ (λt. f t + g t) ∂M = ∫ f ∂M + ∫ g ∂M
    by simp
qed

theorem integral-mono:
  assumes f: integrable f M
  and g: integrable g M and fg: f≤g
  shows ∫ f ∂M ≤ ∫ g ∂M
proof −
  from f obtain pf nf where pf: pf ∈ nnfis (pp f) M
    and nf: nf ∈ nnfis (np f) M and ms: measure-space M
    by (auto simp add: integrable-def)

  from g obtain pg ng where pg: pg ∈ nnfis (pp g) M
    and ng: ng ∈ nnfis (np g) M
    by (auto simp add: integrable-def)
```

```
  { fix t
    from fg have f t ≤ g t
      by (simp add: le-fun-def)
    hence pp f t ≤ pp g t and np g t ≤ np f t
      by (auto simp add: positive-part-def negative-part-def)
  }
  hence pp f ≤ pp g and np g ≤ np f
    by (simp-all add: le-fun-def)
  with ms pf pg ng nf have pf ≤ pg and ng ≤ nf
    by (simp-all add: nnfis-mono)

  also
  from ms pf pg ng nf have (THE i. i ∈ nnfis (pp f) M) = pf
    and (THE i. i ∈ nnfis (np f) M) = nf
    and (THE i. i ∈ nnfis (pp g) M) = pg
    and (THE i. i ∈ nnfis (np g) M) = ng
    by (auto simp add: nnfis-unique)
  with f g have ∫ f ∂M = pf − nf
    and ∫ g ∂M = pg − ng
    by (auto simp add: integral-def)

  ultimately show ?thesis
    by simp
qed

theorem integral-times:
  assumes int: integrable f M
  shows integrable (λt. a∗f t) M and ∫ (λt. a∗f t) ∂M = a∗∫ f ∂M
```

To try out our definitions in an application, only one more theorem is missing. The famous Markov–Chebyshev inequation is not difficult to arrive at using the basic integral properties.

```
theorem assumes int: integrable f M and a: 0<a and intp: integrable (λx. |f x|
^ n) M
  shows markov-ineq: law M f {a..} ≤ ∫ (λx. |f x| ^ n) ∂M / (a^n)
proof −
  from int have rv: f ∈ rv M
    by (rule integrable-rv)
  hence ms: measure-space M
    by (simp add: rv-def)
  from ms rv have ams: {w. a ≤ f w} ∈ measurable-sets M
    by (simp add: rv-ge-iff)

  from rv have (a^n)∗law M f {a..} = (a^n)∗measure M {w. a ≤ f w}
    by (simp add: distribution-def vimage-def)
  also
  from ms ams have int2: integrable χ {w. a ≤ f w} M
    and eq2: ... = (a^n)∗∫ χ {w. a ≤ f w} ∂ M
```

```
    by (auto simp add: integral-char)
  note eq2 also
  from int2 have int3: integrable (λt. (a^n)*χ {w. a ≤ f w} t) M
    and eq3: ... = ∫ (λt. (a^n)*χ {w. a ≤ f w} t) ∂ M
    by (auto simp add: integral-times)
  note eq3 also
  { fix t
    from a have (a^n)*χ {w. a ≤ f w} t ≤ |f t| ^ n
    proof (cases a ≤ f t)
      case False
      thus ?thesis
        by (simp add: characteristic-function-def)
    next
      case True
      with a have a ^ n ≤ (f t) ^ n
        by (simp add: power-mono)
      also
      have (f t) ^ n ≤ |(f t) ^ n|
        by arith
      hence (f t) ^ n ≤ |f t| ^ n
        by (simp add: power-abs)
      finally
      show ?thesis
        by (simp add: characteristic-function-def)
    qed
  }
  with int3 intp have ... ≤ ∫ (λx. |f x| ^ n) ∂M
    by (simp add: le-fun-def integral-mono)

  also
  from a have 0 < a^n
    by (rule zero-less-power)
  ultimately show ?thesis
    by (simp add: pos-le-divide-eq mult.commute)
qed

end
```

Chapter 4

Epilogue

To come to a conclusion, a few words shall subsume the work done and point out opportunities for future research at the same time.

What has been achieved? After opening with some introductory notes, we began translating the language of measure theory into machine checkable text. For the material in section 2.1, this had been done before. Besides laying the foundation for the development, the style of presentation should make it noteworthy.

It is a particularity of the present work that its theories are written in the Isar language, a declarative proof language that aims to be "intelligible". This is not a novelty, nor is it the author's merit. Still, giving full formal proofs in a text intended to be read by people is in a way experimental. Clearly, it is bound to put some strain on the reader. Nevertheless, I hope that we have made a little step towards formalizing mathematical knowledge in a way that is equally suitable for computation and understanding. One aim of the research done has been to demonstrate the viability of this approach. Unquestionably, there is plenty room for improvement regarding the quality of presentation. The language itself has, in my opinion, proven to be fit for a wide range of applications, including the classical mathematics we used it for.

Returning to a more content-centered viewpoint, we discussed the measurability of real-valued functions in section 2.2. As explained there, earlier scholarship has resulted in related theories for the MIZAR environment though the development seems to have stopped. Anyway, the mathematics covered should be new to HOL-based systems.

More functions could obviously be demonstrated to be random variables. We shortly commented on an alternative approach in the section just mentioned. It is applicable to continuous functions, proving these measurable all at once. Efforts on topological spaces would be required, but they constitute an interesting field themselves, so it is probably worth the while.

In the third chapter, integration in the Lebesgue style has been looked at in depth. To my knowledge, no similar theory had been developed in a theorem prover up to this point. We managed to systematically establish the integral of increasingly complex functions. Simple or nonnegative functions ought to be treated in sufficient detail by now. Of course, the repository of potential supplementary facts is vast. Convergence theorems, as well as the interrelationship with differentiation or concurrent integral concepts, are but a few examples. They leave ample space for subsequent work.

A shortcoming of the present development lies in the lack of user assistance. Greater care could be taken to ensure automatic application of appropriate simplification rules — or to design such rules in the first place. Likewise, the principal requirement of integrability might hinder easy usage of the integral. Fixing a default value for undefined integrals could possibly make some case distinctions obsolete. Facets like these have not been addressed in their due extent.

Bibliography

[1] Heinz Bauer. *Maß- und Integrationstheorie*. de Gruyter, 1990.

[2] Patrick Billingsley. *Probability and Measure*. John Wiley, second edition, 1986.

[3] Noboru Endou, Katsumi Wasaki, and Yasunari Shidama. Definitions and basic properties of measurable functions. *Journal of Formalized Mathematics*, 12, 2000. Available on the web as http://mizar.uwb.edu.pl/JFM/Vol12/mesfunc1.html.

[4] Noboru Endou, Katsumi Wasaki, and Yasunari Shidama. The measurability of extended real valued functions. *Journal of Formalized Mathematics*, 12, 2000. Available on the web as http://mizar.uwb.edu.pl/JFM/Vol12/mesfunc2.html.

[5] Jacques D. Fleuriot and Lawrence C. Paulson. Mechanizing nonstandard real analysis. *LMS Journal of Computation and Mathematics*, 3:140–190, 2000. Available on the web as http://www.lms.ac.uk/jcm/3/lms1999-027/.

[6] Joe Hurd. *Formal Verification of Probabilistic Algorithms*. PhD thesis, University of Cambridge, 2002. Available on the web as http://www.cl.cam.ac.uk/~jeh1004/research/papers/thesis.html.

[7] Tobias Nipkow. Order-sorted polymorphism in isabelle. In Gérard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 164–188. Cambridge University Press, 1993. Available on the web as http://www4.informatik.tu-muenchen.de/~nipkow/pubs/lf91.html.

[8] Sebastian Skalberg. Import tool. Available on the web as http://www.mangust.dk/skalberg/isabelle.php.

[9] Markus Wenzel. Using axiomatic type classes in Isabelle, 2002. Unpublished. Available on the web as http://isabelle.in.tum.de/dist/Isabelle2002/doc/axclass.pdf.

[10] David Williams. *Probability with Martingales*. Cambridge University Press, 1991.