Technische Universität München
Department of Informatics
Diploma Thesis

Formalizing Integration Theory, with an Application to Probabilistic Algorithms

Stefan Richter

Examiner: Tobias Nipkow
Advisor: Tobias Nipkow
Thesis handed in on: 15th May 2003

Declaration

I hereby affirm that I have composed this diploma thesis single-handed, and that I only used the quoted resources and auxiliary means.

Stefan Richter, 15th May 2003

Abstract

Inter alia, Lebesgue-style integration plays a major role in advanced probability. We formalize a significant part of its theory in Higher Order Logic using the generic interactive theorem prover Isabelle/Isar. This involves concepts of elementary measure theory, real-valued random variables as Borel-measurable functions, and a stepwise inductive definition of the integral itself. Building on previous work about formal verification of probabilistic algorithms, we exhibit an example application in this domain; another primitive for randomized functional programming is developed to this end. All proofs are carried out in human readable style using the Isar language.

Contents

1 Prologue **4**

1.1 Formalized mathematics 4

1.2 Lebesgue integration 7

1.3 The Isabelle/Isar-HOL-Real environment 9

1.4 About the presentation 11

2 Measurable Functions **12**

2.1 Preliminaries 12

2.1.1 Sigma Algebras 12

2.1.2 Monotone convergence 17

2.1.3 Measure spaces 21

2.2 Real-valued random variables 26

3 Integration **40**

3.1 Two approaches that failed 40

3.1.1 A closed expression 40

3.1.2 A one-step inductive definition 41

3.2 The three-step approach 42

3.2.1 Simple functions 43

3.2.2 Nonnegative Functions 53

3.2.3 Integrable functions 64

4 Probabilistic Algorithms **72**

4.1 The probability space 72

4.2 A new primitive 77

4.3 The first moment method 84

5 Epilogue **90**

A Auxiliary Theories **92**
A.1 Rational numbers 92
A.2 Finite sum properties 99

Acknowledgements

I thank Tobias Nipkow, who helped me find and gave me the opportunity to pursue this topic. His door was always open, and it has been a great privilege to have him as an advisor.

Moreover, the people in his theorem proving group have been very friendly and helpful, even prior to this work. I want to especially acknowledge Markus Wenzel and Gerwin Klein for generous Isabelle support. Martin Strecker and Clemens Ballarin contributed valuable literature.

I am grateful to Joe Hurd of the University of Cambridge, who not only inspired this thesis, but also gave me a lot of encouraging feedback.

My family have all been very supportive over the years. My parents, most of all, made everything possible and bolstered me in times of crisis.

Sincere thanks are due to my fellow artists, especially to Alex, Bertolt, Chris, Dirk and Verena for persevering proofreading and encouragement.

Chris has been with me through it all. Her empathy and patience kept me going on a thousand occasions. I cannot begin to thank her enough for everything she did.

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" ([11] 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. We do not undertake such analysis here, as I deem it too complex an example for the application of the theory that is the main part of this thesis. Still, an application to probabilistic algorithms is presented in chapter 4, even if in the end we prove a combinatorial statement. Also, the formalization takes place in a Higher Order Logic environment, even though we do not use the HOL theorem prover [8] but the Isabelle/Isar HOL instantiation [16, 25].

1.1 Formalized mathematics

John Harrison has written a treatise [9] entitled precisely like this section, and here we merely subsume a few arguments that support the idea. For a more detailed discussion, including a history of formal mathematics, the interested reader is referred to the original paper.

One of the most immediate applications crossing the mind is formal verification of technical systems. This means that the requirements for a technical design, like an algorithm or hardware architecture, are written down in a formal language — a process known as specification — and the final product is then proven to fulfill this specification. Though in practice this is often a tedious (and thus expensive) work, there are areas, such as security critical software, in air traffic or medicine applications for example, where it may be applied successfully. We may well see a rise of such methods with the increasing impact of computing technology in everyday life.

The deeper benefit sought after in this viewpoint is a mechanized check for errors. Correctness is also an important aspect in traditional mathematics; and as Harrison ([9] p. 14) puts it, “In formalizing mathematics, we must rephrase informal constructs in terms of formal ones, which is merely an extreme case of defining non-rigorous concepts rigorously.”. In this respect, theorem proving may be gainful in classical mathematical research, too.

For this argument to hold, correctness of the checking machinery must of course be ensured. In our case of Isabelle, as well as in the HOL system, this property is ensured by construction following the so-called LCF paradigm. Robin Milner [13] developed the underlying design philosophy, which employs a small trusted logical kernel to flexibly implement any logic that is built on top by reducing any proof to the simple deduction steps the LCF kernel allows. For obvious reasons, we cannot detail the construction here.

There are many more reasons to make mathematical reasoning precise enough to be checked upon by a software system, and nine of them are enumerated in a manifesto by the QED project [1], which aims for formalization of eventually all of mathematics:

> first to help mathematicians cope with the explosion in mathematical knowledge
>
> second to help development of highly complex IT systems by facilitating the use of formal techniques
>
> third to help in mathematical education
>
> fourth to provide a cultural monument to “the fundamental reality of truth”
>
> fifth to help preserve mathematics from corruption
>
> sixth to help reduce the ‘noise level’ of published mathematics
>
> seventh to help make mathematics more coherent
>
> eighth to add to the body of explicitly formulated mathematics
>
> ninth to help improve the low level of self-consciousness in mathematics

These points are elaborated on in the original text, which also contains a rebuttal to a lot of possible objections to the idea. Therefore, a short explanation shall suffice here.

The opening point aims at the creation of a universal database of formal text, but this also requires the act of formalization itself in the first place. I have already commented on the use in technical systems.

The next three arguments are of a more cultural nature. Interactive proof systems and formal theories can serve students to achieve training in mathematical subjects by hands-on experience, for "The development of mathematical ability is notoriously dependent upon 'doing' rather than upon 'being told' or 'remembering' " (ibid.).

Furthermore, the achievement of (a large body of) formal mathematics laid out in documents can be seen as an aesthetic cultural monument. The authors of the QED manifesto cite Aristotle to support this idea: "That which is proper to each thing is by nature best and most pleasant for each thing; for man, therefore, the life according to reason is best and pleasantest, since reason more than anything else is man." (ibid.).

A motive that might seem more arcane at first glance is the protection of mathematics from corruption. It is not so absurd if you consider the influence that, for example, politics, emotion, or a certain kind of fashion exert on any science. For the most abhorrent imaginable instance, in Nazi Germany there was a strong movement towards the establishment of a so-called "Deutsche Mathematik", which concentrated on applications and marginalized a "Jewish style" of more theoretic research. A machine checkable foundation could supply proof in a deeper sense if doubt arises as to the validity of certain theories.

The final four points refer to aspects of mathematical research practice. They stress that formalization could reduce the amount of erroneous or redundant results published, help unite the notation of different branches of mathematics via a natural process of generalization, make explicit a large amount of "mathematical folklore" — lemmata and techniques that are used without conscious consideration — and might ease a meta analysis of the structure of mathematics.

While one certainly does not have to subscribe to all of these arguments in equal proportion, I feel they provide a secure ground for the justification of efforts like the present thesis.

1.2 Lebesgue integration

Now that some of the reasoning behind formal reasoning in general has been illustrated, it remains to explain the choice of the mathematical theory that is to be presented in this thesis[1]. But allow me to comment on the title first. In the literature on analysis and related fields, Lebesgue integration is often understood as integration with respect to the Lebesgue measure[2], the standard measure for sets of real numbers. We do not cover it at all. Instead, this thesis concentrates on a field that is often simply called integration theory. The way it develops, however, was generalized from the approach that Henri Lebesgue took to integration of real functions. We could therefore speak of "Lebesgue-style" integration. From now on, this will be shortened to Lebesgue integration, but should always be understood as a generic method independent of any specific measure.

As indicated in the very beginning of this chapter, 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 valuable with respect to a (probability) measure. This concept is not conveyed to most people when they first learn about integration. Rather, an integral is often introduced as the area between the graph of a function and the x-axis in a coordinate system, or as the anti-derivative of a function. These are strongly concrete ideas. The first leads into the definition the Riemann integral, and the second is sometimes described as the Newton integral but usually not used as a definition. Indeed, it is the gist of the Fundamental Theorem of Calculus that these two notions coincide.

The idea of integration as measurement of an area can be traced back to ancient Greece. Newton and Leibnitz found the interpretation as inverse operation to differentiation at the end of the 17th century. A.L. Cauchy developed requirements for integrability in the 19th century with a theory of limits, establishing the condition of continuity for a function to have an integral. In 1854, B. Riemann made the Cauchy approach more precise, also permitting non-continuous functions. The Riemann integral is achieved by partitioning the *domain* of a function in intervals, summing up the products of the infimum/supremum of the function in each interval with its respective length, and taking the limit to infinitely many intervals.

In 1902, Henri Lebesgue presented his new definition of an integral that takes an opposite approach: Now the *range* of a function is discretized. The definition can be made stepwise again. One begins taking into account only so-called simple or elementary functions having only finitely many different

[1]The following remarks about the history of integration and different concepts of integrals are based on the introduction of [2] and the section about integration in [10], as well as [23]. Detailed references to historical literature can be found in these works.

[2]The notion of measure will be covered in detail later on.

positive values. The products of these values with the *measures* of their preimages are then summed up to yield the integral. If a nonnegative function can now be represented as the limit of simple functions, its integral is simply defined as the limit of the integrals of these functions. Finally, general functions can be represented as the difference between two positive functions, and the integral is precisely the difference between their integrals. This construction lies at the heart of the present thesis, so we will develop it in far greater detail later on.

The new way of integration accounts for significant advantages: To begin with, it is strictly more general. Any Riemann integral is also a Lebesgue integral, and the two coincide. The inclusion is not valid in the other direction, the typical example being the characteristic function of the set of rational numbers $\chi\mathbb{Q}$, the Lebesgue integral of which is 0, with the Riemann integral undefined. This does not hold if you consider improper Riemann integrals. They are a somewhat ad hoc extension of Riemann integrals that is neither a subset nor a superset of the Lebesgue type. This complication stems entirely from the nonnegative nature of Lebesgue integration: A function f is Lebesgue integrable if and only if $|f|$ is integrable. In a way, this condition is not necessary, but it simplifies matters greatly.

Later in the 20th century, an even more general notion was conceived of that encompassed all other definitions. It was first discovered in the 1910s by Arnaud Denjoy and Oskar Perron, whose definitions were equivalent but highly complex. Only in the 1950s, Ralph Henstock and Jaroslav Kurzweil found a considerably simpler formulation of what is now often called the gauge integral. Indeed, theirs is only a slight generalization — at least over compact intervals — of the Riemann integral, nevertheless resulting in a much more powerful theory. That most powerful integral has even been formalized in the HOL theorem prover by Harrison [10]. So why do we need a Lebesgue integral at all?

As described above, other than in the alternative approaches, the concept of a *measure* lies at the heart of Lebesgue integration. A measure is simply a function fulfilling 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. Note that the construction described above, unlike the Riemann one, does not depend on the structure of the 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[11]; our integral is then just a tool that extends this work in the sense depicted at the very beginning of this chapter. Now the gauge integral can be used in such a general setting, too. However, the simplicity that makes it so elegant in real analysis (especially over compact intervals) seems to get lost in those cases, as the intuition behind it is very similar to the Riemann construction, which depends heavily on the structure of the real numbers as domain.

Moreover, a third advantage of the Lebesgue approach is the clear and beautiful theory it yields. The integrable functions form mathematically regular spaces and the behavior of the integral regarding limits is as simple as can be. In comparison, the gauge integral has somewhat more intricate properties as the functions that it alone allows us to integrate often seem like rather pathological cases. We cannot go into detail here and in this work the effects might not yet be felt, but the Lebesgue integral therefore seems well suited for formalization.

A last — and in my opinion quite compelling — reason for our choice is that the theory that is to be formalized here is a very mature and oft-employed one. It is the theory taught in most graduate calculus courses and there is a lot of experience in finding good ways to present it, in textbooks for example. The Henstock-Kurzweil theory still has to catch up here, even if it is in terms of popularity only.

1.3 The Isabelle/Isar-HOL-Real environment

As stated before, the formalization is performed in the theorem prover Isabelle [20, 19], using the Isar language [25], the HOL logic [17], and the Real theory [7].

Isabelle is a generic theorem prover. That is, different object logics can be defined in a meta logic. It is based on ML, a non-lazy functional language [21]. Until recently, proofs for Isabelle consisted of tactic scripts in ML. In a backward proof style, so-called tactics were applied to simplify goals into subgoals or solve them completely. It was even possible to create your own tactics with relatively little effort. One major drawback to this style of proof is that it cannot be understood by humans without looking at the emerging subgoals in a replayed proof process. Still, in most theorem provers, a similar style is common.

On the other hand, there have long been efforts to create formal proof documents that people can actually read, the most prominent arguably being MIZAR [24]. Lately, the Isar language, which improves on many shortcomings of MIZAR and similar systems [25], has been devised to support "intelligible Semi-Automatic Reasoning" in Isabelle and possibly other theorem provers. The style of proof document supported by such an environment is known as declarative, meaning that intermediate goals are declared explicitly and then solved either by simple automatic steps or by a recursive new proof. Isar proof scripts can look almost as textbook proofs, but the final document depends on the author's style and experience, of course. All proofs in this thesis are conducted in a declarative manner.

The logic that is used in what follows is HOL, a Higher Order Logic, which basically means that functions are allowed as arguments to functions. It is based on "a simple theory of types" by Alonzo Church [4], adding polymorphism above all. "HOL can best be understood as a simply-typed version of classical set theory" [3]. It includes the axiom of choice via a special ε operator, of which we will make use. HOL is by far the most popular of Isabelle's object logics, a fact that is witnessed by a large library of existing theories. Isabelle/HOL is but one of many incarnations of Higher Order Logic, with the Cambridge HOL system [8] arguably the best known.

Among the many preexisting theories we will use in what follows, HOL-Real is probably most prominent, since integration is usually done into the real numbers. The libraries defining the reals and their properties have been created as a sort of by-product in an effort to formalize non-standard analysis [7], so the hyper-reals are available, too. Nevertheless, we restrict ourself to the use of the common real numbers. First, to keep the terminology as consistent as possible with theories that use the libraries of the HOL theorem prover, chiefly with the work of Hurd [11] that this thesis is based on. Second, to minimize learning and translation efforts for the author as well as for the reader, since the rivaling theory is supposedly called "non-standard" for a reason.

I refrain from giving an introduction to the Isar language here. Markus Wenzel's PhD thesis [25] is a highly detailed reference. Instead, I rely on the similarity to textbook proofs that is hoped to be achieved and point out peculiarities where they occur.

[3] From the abstract of [17].

1.4 About the presentation

The dominant part of this thesis consists of commented Isar text, rendered by the document preparation facilities included in Isabelle/Isar. Any theorem found and used in the process is displayed, in contrast to only select proofs. This practice necessitates a slightly formal style to a degree. I hope the following will still be "intelligible" as the Isar acronym postulates.

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. Two approaches that failed to establish the necessary facts are shortly commented on. Eventually, we build the integral for increasingly complex functions and prove essential properties, discovering the connection with measurability in the end.

Before ending the work with a short summary and suggestions for future work, we test our achievements in an application. The first moment method is applied to the problem of satisfiability for propositional formulas in conjunctive normal form with k literals per clause. Though the setup is simple enough in terms of integration, a new primitive is needed to represent the probabilistic programs involved.

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-Algebra2* = *Main*:

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. *Sigma-Algebra2* is built upon *Sigma-Algebra*, a tiny example demonstrating the use of inductive definitions by Markus Wenzel. This theory as well as *Measure* in 2.1.3 is heavily influenced by Joe Hurd's thesis [11] 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.

constdefs
  sigma-algebra:: $'a\ set\ set \Rightarrow bool$
  sigma-algebra $A \equiv$

$\{\} \in A \wedge (\forall\, a.\ a \in A \longrightarrow -a \in A) \wedge$
$(\forall\, a.\ (\forall\ \ i{::}nat.\ a\ i \in A) \longrightarrow (\bigcup i.\ a\ i) \in A)$

The **constdefs** command declares and defines at the same time 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.

consts
$sigma :: {}'a\ set\ set \Rightarrow {}'a\ set\ set$

inductive *sigma A*
intros
basic: $a \in A \Longrightarrow a \in sigma\ A$
empty: $\{\} \in sigma\ A$
complement: $a \in sigma\ A \Longrightarrow -a \in sigma\ A$
Union: $(\bigwedge i{::}nat.\ a\ i \in sigma\ A) \Longrightarrow (\bigcup i.\ a\ i) \in sigma\ A$

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

theorem *sigma-UNIV*: $UNIV \in sigma\ A$

theorem *sigma-Inter*:
$(\bigwedge i{::}nat.\ a\ i \in sigma\ A) \Longrightarrow (\bigcap i.\ a\ i) \in sigma\ A$

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 $A \subseteq sigma\ A$
— 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* $\subseteq$ *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.

consts
trivial-series:: $'a\ set \Rightarrow 'a\ set \Rightarrow (nat \Rightarrow 'a\ set)$

The new constant is only declared but not yet defined.

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

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

  proof (*rule set-ext*)

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

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

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

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

qed

also have $(\bigcap i::nat.\ trivial\text{-}series\ a\ b\ i) \in A$
proof $-$
{ fix i
from $a\ b$ **have** $trivial\text{-}series\ a\ b\ i \in A$
by $(cases\ i)\ auto$
}
hence $\bigwedge i.\ trivial\text{-}series\ a\ b\ i \in sigma\ A$
by $(simp\ only{:}\ sigma.basic)$
hence $(\bigcap i::nat.\ trivial\text{-}series\ a\ b\ i) \in sigma\ A$
by $(simp\ only{:}\ sigma\text{-}Inter)$
with s **show** $?thesis$
by $(simp\ only{:}\ sigma\text{-}sigma\text{-}algebra)$
qed

finally show $?thesis$ **.**
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 $a{:}(\bigwedge i::nat.\ i \in S \Longrightarrow a\ i \in sigma\ A)$
shows $(\bigcap i{\in}S.\ a\ i) \in sigma\ A$
proof $-$
from a **have** $\bigwedge i.\ (if\ i{\in}S\ then\ \{\}\ else\ UNIV) \cup a\ i \in sigma\ A$
by $(simp\ add{:}\ sigma.intros\ sigma\text{-}UNIV)$
hence $(\bigcap i.\ (if\ i{\in}S\ then\ \{\}\ else\ UNIV) \cup a\ i) \in sigma\ A$
by $(rule\ sigma\text{-}Inter)$
also have $(\bigcap i.\ (if\ i{\in}S\ then\ \{\}\ else\ UNIV) \cup a\ i) = (\bigcap i{\in}S.\ a\ i)$
by *force*
finally show $?thesis$ **.**
qed

lemma assumes s: *sigma-algebra a* **shows** *sigma-algebra-UNIV*: $UNIV{\in}a$
proof $-$
from s **have** $\{\}{\in}a$ **by** $(unfold\ sigma\text{-}algebra\text{-}def)\ blast$
with s **show** $?thesis$ **by** $(unfold\ sigma\text{-}algebra\text{-}def)\ auto$
qed

end

2.1.2 Monotone convergence

theory *MonConv* = *Lim*:

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.

$mon\text{-}conv\ u\ f \equiv (\forall\, n.\ u\ n \leq u\ (Suc\ n)) \wedge isLub\ UNIV\ (range\ u)\ f$

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.

It still seems worthwhile to add the type of real- (or rather ordered-) valued functions to the ordered types by defining the less-or-equal relation pointwise.

instance *fun* :: (*type*,*ord*)*ord* **..**

defs
le-fun-def: $f \leq g \ \equiv \forall\, x.\ f\ x \leq g\ x$

The following theorem is often used in this context and therefore even added to the calculational transitivity rules. For reasons of brevity, the proof is omitted here, as will be the case with several of the subsequent facts.

theorem assumes $f \leq (k{::}'a \Rightarrow real)$ **and** $k \leq g$
shows *realfun-le-trans*[*trans*]: $f \leq g$

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
$mon\text{-}conv{::}\ (nat \Rightarrow 'a) \Rightarrow 'a{::}ord \Rightarrow bool\ (\text{-}{\uparrow}\text{-}\ [60,61]\ 60)$

[1] For the concept of axiomatic type classes, see [15, 26]

defs (**overloaded**)
real-mon-conv: $x{\uparrow}(y{::}real) \equiv (\forall\, n.\ x\ n \leq x\ (Suc\ n)) \wedge x\ {-}{-}{-}{-}{>}\ y$
realfun-mon-conv:
$u{\uparrow}(f{::}'a \Rightarrow real) \equiv (\forall\, n.\ u\ n \leq u\ (Suc\ n)) \wedge\ (\forall\, w.\ (\lambda n.\ u\ n\ w)\ {-}{-}{-}{-}{>}\ f\ w)$
set-mon-conv: $A{\uparrow}(B{::}'a\ set) \equiv (\forall\, n.\ A\ n \leq A\ (Suc\ n)) \wedge B = (\bigcup n.\ A\ n)$

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

The long arrow signifies convergence of real sequences as defined in the theory *SEQ* [7]. 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 assumes *ls*: $x\ {-}{-}{-}{-}{>}\ y$
shows *limseq-shift*: $(\lambda m.\ x\ (m{+}i))\ {-}{-}{-}{-}{>}\ y$ **using** *ls*

theorem assumes *mon-conv*: $x{\uparrow}(y{::}real)$
shows *real-mon-conv-le*: $x\ i \leq y$
proof −
from *mon-conv* **have** $(\lambda m.\ x\ (m{+}i))\ {-}{-}{-}{-}{>}\ y$
by (*simp add*: *real-mon-conv limseq-shift*)
also from *mon-conv* **have** $\forall\, m.\ x\ i \leq x\ (m{+}i)$ **by** (*simp add*: *mon-conv-mon*)
ultimately show *?thesis* **by** (*rule LIMSEQ-le-const*)
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 $x\ i\ w \leq y\ w$
by (*rule real-mon-conv-le*)
}
thus *?thesis* **by** (*simp add*: *le-fun-def*)
qed

lemma assumes *mon-conv*: $x{\uparrow}(y{::}real)$
and *less*: $z < y$
shows *real-mon-conv-outgrow*: $\exists\, n.\ \forall\, m.\ n \leq m \longrightarrow z < x\ m$

proof −
  from *less* **have** $0 < y{-}z$
    by *simp*
  with *mon-conv* **have** $\exists\, n. \forall\, m.\ n \leq m \longrightarrow |x\ m + -\ y| < y{-}z$
    by (*simp add*: *real-mon-conv LIMSEQ-def*)
  also
  { fix *m*
    from *mon-conv* **have** $x\ m \leq y$
      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 *fast*
qed

theorem *real-mon-conv-times*:
  assumes *xy*: $x\uparrow(y::real)$ **and** *nn*: $0{\leq}z$
  shows $(\lambda m.\ z{*}x\ m)\uparrow(z{*}y)$

theorem *realfun-mon-conv-times*:
  assumes *xy*: $x\uparrow(y::'a{\Rightarrow}real)$ **and** *nn*: $0{\leq}z$
  shows $(\lambda m\ w.\ z{*}x\ m\ w)\uparrow(\lambda w.\ z{*}y\ w)$

theorem *real-mon-conv-add*:
  assumes *xy*: $x\uparrow(y::real)$ **and** *ab*: $a\uparrow(b::real)$
  shows $(\lambda m.\ x\ m + a\ m)\uparrow(y + b)$

theorem *realfun-mon-conv-add*:
  assumes *xy*: $x\uparrow(y::'a{\Rightarrow}real)$ **and** *ab*: $a\uparrow(b::'a \Rightarrow real)$
  shows $(\lambda m\ w.\ x\ m\ w + a\ m\ w)\uparrow(\lambda w.\ y\ w + b\ w)$

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 *mon* **have** *m2*: $\forall\, n.\ c\ n \leq c\ (Suc\ n)$
    by *simp*
  also

def $g \equiv (\lambda n{::}nat.\ x)$
hence $\forall\, n.\ g(Suc\ n) \leq g(n)$
by *simp*
moreover
— This is like **also** but lacks the transitivity step.

from *bound* **have** $\forall\, n.\ c\ n \leq g(n)$
by (*simp add*: *g-def*)

ultimately have
$\exists\, l\ m.\ l \leq m \wedge ((\forall\, n.\ c\ n \leq l) \wedge c{-}{-}{-}{-}{>}\ l) \wedge$
$(\forall\, n.\ m \leq g\ n) \wedge (g\ {-}{-}{-}{-}{>}\ m)$
by (*rule lemma-nest*)
then obtain *l m* **where** *lm*: $l \leq m$ **and** *conv*: $c\ {-}{-}{-}{-}{>}\ l$
and *gm*: $g\ {-}{-}{-}{-}{>}\ m$
by *fast*
from *gm g-def* **have** $m{=}x$
by (*simp add*: *LIMSEQ-const LIMSEQ-unique*)
with *lm conv m2* **show** *?thesis*
by (*auto simp add*: *real-mon-conv*)
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=Sigma-Algebra2+MonConv+NthRoot*:

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 [11].

constdefs

$measurable :: 'a\ set\ set \Rightarrow 'b\ set\ set \Rightarrow ('a \Rightarrow 'b)\ set$
$measurable\ F\ G \equiv \{f.\ \forall g{\in}G.\ f\ -`\ g \in F\}$

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].

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

$measure :: ('a\ set\ set * ('a\ set \Rightarrow real)) \Rightarrow ('a\ set \Rightarrow real)$
$measure \equiv snd$

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.

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

$countably\text{-}additive :: ('a\ set\ set * ('a\ set \Rightarrow real)) \Rightarrow bool$
$countably\text{-}additive\ M \equiv (\forall f{::}(nat \Rightarrow 'a\ set).\ range\ f \subseteq measurable\text{-}sets\ M$
$\wedge\ (\forall m\ n.\ m \ne n \longrightarrow f\ m \cap f\ n = \{\})\ \wedge\ (\bigcup i.\ f\ i) \in measurable\text{-}sets\ M$
$\longrightarrow (\lambda n.\ measure\ M\ (f\ n))\ sums\ \ measure\ M\ (\bigcup i.\ f\ i))$

This last property deserves some comments. The conclusion is usually — also in the aforementioned source — phrased as
$measure\ M\ (\bigcup i.\ f\ i) = (\sum n.\ measure\ M\ (f\ n))$.
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.

[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.

[3] Which is merely syntactic sugar for the *suminf* functional from the *Series* theory [7].

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].

measure-space:: (*'a set set* $*$ (*'a set* $\Rightarrow$ *real*)) $\Rightarrow$ *bool*
measure-space M $\equiv$ *sigma-algebra* (*measurable-sets M*) $\wedge$
positive M $\wedge$ *countably-additive M*

Note that our definition is restricted to finite measure spaces — that is, *measure M UNIV* $< \infty$ — 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 the proofs 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 [2] is relatively straightforward, the only difficulty that arises being rule induction.

theorem assumes *sig*: *sigma-algebra a* **and** *meas*: $f \in$ *measurable a b* **shows**
measurable-lift: $f \in$ *measurable a* (*sigma b*)
proof –
def $Q \equiv \{q.\ f -` q \in a\}$
with *meas* **have** *1*: $b \subseteq Q$ **by** (*auto simp add*: *measurable-def*)

{ fix x **assume** $x \in$*sigma b*
hence $x \in Q$
proof (*induct rule*: *sigma.induct*)
case *basic*
from *1* **show** $\bigwedge a.\ a \in b \Longrightarrow a \in Q$ **..**
next
case *empty*
from *sig* **have** $\{\} \in a$
by (*simp only*: *sigma-algebra-def*)
thus $\{\} \in Q$
by (*simp add*: *Q-def*)
next
case *complement*
fix r **assume** $r \in Q$
then obtain *r1* **where** *im*: $r1 = f -` r$ **and** *a*: $r1 \in a$
by (*simp add*: *Q-def*)
with *sig* **have** $-r1 \in a$
by (*simp only*: *sigma-algebra-def*)

[4] Joe Hurd inherited this practice from a very influential probability textbook [27]

with *im Q-def* **show** $-r \in Q$
  by (*simp add*: *vimage-Compl*)
next
  case *Union*
  fix *r* **assume** $\bigwedge i::nat.\ r\ i \in Q$
  then obtain *r1* **where** *im*: $\bigwedge i.\ r1\ i = f -` r\ i$ **and** *a*: $\bigwedge i.\ r1\ i \in a$
    by (*simp add*: *Q-def*)
  from *a sig* **have** *UNION UNIV r1* $\in a$
    by (*auto simp only*: *sigma-algebra-def*)
  with *im Q-def* **show** *UNION UNIV r* $\in Q$
    by (*auto simp add*: *vimage-UN*)
qed }

hence (*sigma b*) $\subseteq Q$ **..**
thus $f \in$ *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.

consts
  mkdisjoint:: $(nat \Rightarrow 'a\ set) \Rightarrow (nat \Rightarrow 'a\ set)$

primrec
  mkdisjoint $A\ 0 = A\ 0$
  mkdisjoint $A\ (Suc\ n) = A\ (Suc\ n) - A\ n$

lemma *mkdisjoint-un*:
  assumes *up*: $\bigwedge n.\ A\ n \subseteq A\ (Suc\ n)$
  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: ⋀n. A n ∈ measurable-sets M and AB: A↑B
  shows (λn. measure M (A n)) ----> measure M B
proof -

  from AB have up: ⋀n. A n ⊆ A (Suc n)
    by (simp only: set-mon-conv)

  { fix i
    have mkdisjoint A i ∈ measurable-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) ∩ - 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: ⋀i. mkdisjoint A i ∈ measurable-sets M .

  with ms have un: (⋃ i. mkdisjoint A i) ∈ measurable-sets M
    by (simp add: measure-space-def sigma-algebra-def)
  moreover
  from i have range: range (mkdisjoint A) ⊆ measurable-sets M
    by fast
  moreover
  from up have ∀ i j. i ≠ j ⟶ mkdisjoint A i ∩ 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 $(\lambda n.\ sumr\ 0\ n\ (\lambda i.\ measure\ M\ (mkdisjoint\ A\ i)))$
$---->(\sum i.\ measure\ M\ (mkdisjoint\ A\ i))$
by $(rule\ summable\text{-}sumr\text{-}LIMSEQ\text{-}suminf)$

hence $(\lambda n.\ sumr\ 0\ (Suc\ n)\ (\lambda i.\ measure\ M\ (mkdisjoint\ A\ i)))$
$---->(\sum i.\ measure\ M\ (mkdisjoint\ A\ i))$
by $(rule\ LIMSEQ\text{-}Suc)$

ultimately have $(\lambda n.\ sumr\ 0\ (Suc\ n)\ (\lambda i.\ measure\ M\ (mkdisjoint\ A\ i)))$
$---->(measure\ M\ (\bigcup i.\ mkdisjoint\ A\ i))$ **by** *simp*

also
{ fix n
from *up* **have** $A\ n = (\bigcup i{\in}\{..n\}.\ mkdisjoint\ A\ i)$
by $(rule\ mkdisjoint\text{-}un)$
hence $measure\ M\ (A\ n) = measure\ M\ (\bigcup i{\in}\{..n\}.\ mkdisjoint\ A\ i)$
by *simp*

also have
$(\bigcup i{\in}\{..n\}.\ mkdisjoint\ A\ i) = (\bigcup i.\ if\ i{\le}n\ then\ mkdisjoint\ A\ i\ else\ \{\})$
proof $-$
have $UNIV = \{..n\} \cup \{)n..\}$ **by** *auto*
hence $(\bigcup i.\ if\ i{\le}n\ then\ mkdisjoint\ A\ i\ else\ \{\}) =$
$(\bigcup i{\in}\{..n\}.\ if\ i{\le}n\ then\ mkdisjoint\ A\ i\ else\ \{\})$
$\cup\ (\bigcup i{\in}\{)n..\}.\ if\ i{\le}n\ then\ mkdisjoint\ A\ i\ else\ \{\})$
by $(auto\ simp\ add{:}\ UN\text{-}Un)$
also
{ have $(\bigcup i{\in}\{)n..\}.\ if\ i{\le}n\ then\ mkdisjoint\ A\ i\ else\ \{\}) = \{\}$
by *force* **}**
hence $\ldots = (\bigcup i{\in}\{..n\}.\ mkdisjoint\ A\ i)$
by *auto*
finally show
$(\bigcup i{\in}\{..n\}.\ mkdisjoint\ A\ i) = (\bigcup i.\ if\ i{\le}n\ then\ mkdisjoint\ A\ i\ else\ \{\})$ **..**
qed

ultimately have
$measure\ M\ (A\ n) = measure\ M\ (\bigcup i.\ if\ i{\le}n\ then\ mkdisjoint\ A\ i\ else\ \{\})$
by *simp*

also
from *i ms* **have**
$un{:}\ (\bigcup i.\ if\ i{\le}n\ then\ mkdisjoint\ A\ i\ else\ \{\}) \in measurable\text{-}sets\ M$
by $(simp\ add{:}\ measure\text{-}space\text{-}def\ sigma\text{-}algebra\text{-}def)$
moreover
from *i ms* **have**
$range\ (\lambda i.\ if\ i{\le}n\ then\ mkdisjoint\ A\ i\ else\ \{\}) \subseteq measurable\text{-}sets\ M$
by $(auto\ simp\ add{:}\ measure\text{-}space\text{-}def\ sigma\text{-}algebra\text{-}def)$

```
    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
      sumr 0 (Suc n) (λi. measure M (if i ≤ n then mkdisjoint A i else {}))
      by (rule series-zero)
    hence (∑ i. measure M (if i ≤ n then mkdisjoint A i else {})) =
      sumr 0 (Suc n) (λi. measure M (if i ≤ n then mkdisjoint A i else {}))
      by (rule sums-unique[THEN sym])
    also
    have ... = sumr 0 (Suc n) (λi. measure M (mkdisjoint A i))
      by (simp add: sumr-fun-eq)
    finally have
      measure M (A n) = sumr 0 (Suc n) (λi. 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

end
```

2.2 Real-valued random variables

theory *RealRandVar* = *Measure* + *Rats*:

While most of the above material was modeled after Hurd's work (but still proved independently), the original content of this thesis basically starts here. 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 [5, 6]. 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." [11]. 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.

constdefs
Borelsets:: *real set set* ($\mathbb{B}$)
$\mathbb{B}$ == *sigma* $\{S.\ \exists\, u.\ S{=}\{..u\}\}$

rv:: $({}'a\ set\ set * ({}'a\ set \Rightarrow real)) \Rightarrow ({}'a \Rightarrow real)\ set$
rv M == $\{f.\ measure\text{-}space\ M \wedge f \in measurable\ (measurable\text{-}sets\ M)\ \mathbb{B}\}$

As explained in the first paragraph, the preceding definitions 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 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 [3]. It requires some concepts of topologic spaces, which made the following elementary course, based on Bauer's excellent book [2], 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.

constdefs
distribution::
$({}'a\ set\ set * ({}'a\ set \Rightarrow real)) \Rightarrow ({}'a \Rightarrow real) \Rightarrow (real\ set \Rightarrow real)\ (law)$

$f \in rv\ M \Longrightarrow law\ M\ f == (measure\ M) \circ (vimage\ f)$

characteristic-function:: $'a\ set \Rightarrow ('a \Rightarrow real)$ (χ- [*1000*])
$\chi A\ x == if\ x \in A\ then\ 1\ else\ 0$

lemma *char-empty*: $\chi\{\} = (\lambda t.\ 0)$
proof (*rule ext*)
  fix *t*
  show $\chi\{\}\ t = 0$ **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*: $(\lambda x.\ (c::real)) \in measurable\ S\ X$
proof (*unfold measurable-def*, *rule*, *rule*)
  fix *g*
  show $(\lambda x.\ c) -`\ g \in S$
  proof (*cases* $c \in g$)
    case *True*
    hence $(\lambda x.\ c) -`\ g = UNIV$
      by *blast*
    also from *sigma* **have** $UNIV \in S$
      by (*rule sigma-algebra-UNIV*)
    finally show *?thesis* **.**
  next
    case *False*
    hence $(\lambda x.\ c) -`\ g = \{\}$
      by *blast*
    also from *sigma* **have** $\{\} \in S$
      by (*simp only*: *sigma-algebra-def*)
    finally show *?thesis* **.**

  qed
qed

theorem assumes *ms*: *measure-space M*
  shows *const-rv*: $(\lambda x.\ c) \in rv\ M$ **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*: $a \in S$ **and** *sigma*: *sigma-algebra S* **shows**
char-measurable : $\chi a \in measurable\ S\ x$

theorem assumes *ms*: *measure-space M* **and** *A*: $A \in$ *measurable-sets M*
shows *char-rv*: $\chi A \in rv\ M$ **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: $(f \in rv\ M) = (\forall a.\ \{w.\ f\ w \leq a\} \in$ *measurable-sets M*$)$
proof −

have $f \in rv\ M \Longrightarrow \forall a.\ \{w.\ f\ w \leq a\} \in$ *measurable-sets M*
proof
{ fix *a*
assume $f \in$ *measurable* (*measurable-sets M*) $\mathbb{B}$
hence $\forall b \in \mathbb{B}.\ f -\text{`}\ b \in$ *measurable-sets M*
by (*unfold measurable-def*) *blast*
also have $\{..a\} \in \mathbb{B}$
by (*simp only*: *Borelsets-def*) (*rule sigma.basic*, *blast*)
ultimately have $\{w.\ f\ w \leq a\} \in$ *measurable-sets M*
by (*auto simp add*: *vimage-def*)
}
thus $\bigwedge a.\ f \in rv\ M \Longrightarrow \{w.\ f\ w \leq a\} \in$ *measurable-sets M*
by (*simp add*: *rv-def*)
qed

also have $\forall a.\ \{w.\ f\ w \leq a\} \in$ *measurable-sets M* $\Longrightarrow f \in rv\ M$
proof −
assume $\forall a.\ \{w.\ f\ w \leq a\} \in$ *measurable-sets M*
hence $f \in$ *measurable* (*measurable-sets M*)$\{S.\ \exists u.\ S=\{..u\}\}$
by (*auto simp add*: *measurable-def vimage-def*)
with *ms* **have** $f \in$ *measurable* (*measurable-sets M*) $\mathbb{B}$
by (*simp only*: *Borelsets-def measure-space-def measurable-lift*)
thus *?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*: $\forall a.\ \{w.\ f\ w \leq a\} \in A$
shows *le-less*: $\forall a.\ \{w.\ f\ w < (a::real)\} \in A$
proof
fix *a::real*
from *le sigma* **have** $(\bigcup n::nat.\ \{w.\ f\ w \leq a - \mathit{inverse}\ (\mathit{real}\ (\mathit{Suc}\ n))\}) \in A$
by (*simp add*: *sigma-algebra-def*)

also
have $(\bigcup n::nat.\ \{w.\ f\ w \leq a - inverse\ (real\ (Suc\ n))\}) = \{w.\ f\ w < a\}$
proof $-$
{
fix $w\ n$
have $0 < inverse\ (real\ (Suc\ (n::nat)))$
by *simp*
hence $f\ w \leq a - inverse\ (real\ (Suc\ n)) \Longrightarrow f\ w < a$
by *arith*
}
also
{ **fix** w
have $(\lambda n.\ inverse\ (real\ (Suc\ n)))\ {-}{-}{-}{-}{>}\ 0$
by (*rule LIMSEQ-inverse-real-of-nat*)

also assume $f\ w < a$
hence $0 < a - f\ w$ **..**

ultimately have
$\exists\, n0.\ \forall\, n.\ n0 \leq n \longrightarrow abs\ (inverse\ (real\ (Suc\ n))) < a - f\ w$
by (*auto simp add*: *LIMSEQ-def*)
then obtain n **where** $abs\ (inverse\ (real\ (Suc\ n))) < a - f\ w$
by *blast*
hence $f\ w \leq a - inverse\ (real\ (Suc\ n))$
by *arith*
hence $\exists\, n.\ f\ w \leq a - inverse\ (real\ (Suc\ n))$ **..**
}
ultimately show *?thesis* **by** *auto*
qed
finally show $\{w.\ f\ w < a\} \in A$ **.**
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: $(f \in rv\ M) = (\forall a.\ \{w.\ a \leq f\ w\} \in \mathit{measurable\text{-}sets}\ M)$
proof −
from *ms* **have** $(f \in rv\ M) = (\forall a.\ \{w.\ f\ w \leq a\} \in \mathit{measurable\text{-}sets}\ M)$
by (*rule rv-le-iff*)
also have $\ldots = (\forall a.\ \{w.\ a \leq f\ w\} \in \mathit{measurable\text{-}sets}\ M)$ (**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: $(f \in rv\ M) = (\forall a.\ \{w.\ a < f\ w\} \in \mathit{measurable\text{-}sets}\ M)$

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

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 $(\lambda x.\ a + (g\ x) * b) = (\lambda x.\ a)$
by *simp*

also
from g **have** $(\lambda x.\ a) \in rv\ M$
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*: $\bigwedge x\ c.\ (a + g\ x * b \leq c) = (g\ x * b\ \leq c - a)$
by *arith*
have $\forall c.\ \{w.\ \ a + g\ w * b \leq c\} \in$ *measurable-sets M*
proof (*cases* $b<0$)
case *False*
from *prems* **have** $0<b$ **by** *arith*
hence $\bigwedge x\ c.\ \ (g\ x * b\ \leq c - a) = (g\ x \leq (c - a)\ /\ b)$
by (*rule pos-real-le-divide-eq* [*THEN sym*])
with *calc* **have** $\bigwedge c.\ \{w.\ \ a + g\ w * b \leq c\} = \{w.\ g\ w \leq (c - a)\ /\ b\}$
by *simp*

also from *ms g* **have** $\forall a.\ \{w.\ g\ w \leq a\} \in$ *measurable-sets M*
by (*simp add*: *rv-le-iff*)

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

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

also from *ms g* **have** $\forall a.\ \{w.\ a \leq g\ w\ \} \in$ *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 $\{w.\ f\ w \le g\ w\}$. 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[5], so I built a theory with the necessary properties on my own. It can be found in the appendix A.1.

lemma assumes f: $f \in rv\ M$ **and** g: $g \in rv\ M$
  shows *rv-less-rv-measurable*: $\{w.\ f\ w < g\ w\} \in$ *measurable-sets* M
proof –
  from g **have** *ms*: *measure-space* M
    by (*simp add*: *rv-def*)
  have $\{w.\ f\ w < g\ w\} = (\bigcup i.\ let\ s = n\text{-}to\text{-}rat\ i\ in\ \{w.\ f\ w < s\} \cap \{w.\ s < g\ w\})$
  proof
    { fix w **assume** $w \in \{w.\ f\ w < g\ w\}$
      hence $f\ w < g\ w$ **..**
      hence $\exists\, s{\in}\mathbb{Q}.\ f\ w < s \wedge s < g\ w$
        by (*rule rats-dense-in-real*)
      hence $\exists\, s{\in}\mathbb{Q}.\ w \in \{w.\ f\ w < s\} \cap \{w.\ s < g\ w\}$
        by *simp*
      hence $\exists\, i.\ w \in (let\ s = n\text{-}to\text{-}rat\ i\ in\ \{w.\ f\ w < s\} \cap \{w.\ s < g\ w\})$
        by (*simp add*: *Rats-def Let-def*)
      hence $w \in (\bigcup i.\ let\ s = n\text{-}to\text{-}rat\ i\ in\ \{w.\ f\ w < s\} \cap \{w.\ s < g\ w\})$
        by *simp*
    }
    thus
      $\{w.\ f\ w < g\ w\} \subseteq (\bigcup i.\ let\ s = n\text{-}to\text{-}rat\ i\ in\ \{w.\ f\ w < s\} \cap \{w.\ s < g\ w\})$ **..**

    show
      $(\bigcup i.\ let\ s = n\text{-}to\text{-}rat\ i\ in\ \{w.\ f\ w < s\} \cap \{w.\ s < g\ w\}) \subseteq \{w.\ f\ w < g\ w\}$
      by (*force simp add*: *Let-def*)

  qed

  also have
    $(\bigcup i.\ let\ s = n\text{-}to\text{-}rat\ i\ in\ \{w.\ f\ w < s\} \cap \{w.\ s < g\ w\}) \in$ *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** $\{w.\ f\ w < s\} \in$ *measurable-sets* M (**is** $?a{\in}?M$)
        by (*simp add*: *rv-less-iff*)
      moreover from *ms g* **have** $\{w.\ s < g\ w\} \in\ ?M$ (**is** $?b \in\ ?M$)
        by (*simp add*: *rv-gr-iff*)

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

ultimately have *?a* $\cap$ *?b* $\in$ *?M*
by (*rule sigma-algebra-inter*)
}
hence
$\forall$ *i*. *let s* = *n-to-rat i in* {*w*. *f w* < *s*} $\cap$ {*w*. *s* < *g w*} $\in$ *measurable-sets M*
by (*simp add*: *Let-def*)
with *sig* **show** *?thesis*
by (*auto simp only*: *sigma-algebra-def Let-def*)
qed

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

lemma assumes *f*: *f* $\in$ *rv M* **and** *g*: *g* $\in$ *rv M*
shows *rv-le-rv-measurable*: {*w*. *f w* $\leq$ *g w*} $\in$ *measurable-sets M* (**is** *?a* $\in$ *?M*)
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-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-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*: (λ*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*
also from *g* **have** (λ*w*. *a* + (*g w*)∗(−*1*)) $\in$ *rv M*

```
    by (rule affine-rv)
  with f have {w. a + (g w)∗(−1) ≤ f w} ∈ measurable-sets M
    by (simp add: rv-le-rv-measurable)
  finally have {w. a ≤ f w + g w} ∈ measurable-sets M .
}
with ms show ?thesis
  by (simp add: 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: (a² ≤ b²) = (|a| ≤ |b::real|)

lemma assumes f: f ∈ rv M
  shows rv-square: (λw. (f w)²) ∈ rv M

lemma realpow-two-binomial-iff: (f+g::real)² = f² + 2∗(f∗g) + g²

lemma times-iff-sum-squares: f∗g = (f+g)²/4 − (f−g)²/(4::real)

theorem assumes f: f ∈ rv M and g: g ∈ rv M
  shows rv-times-rv: (λw. f w ∗ g w) ∈ rv M
proof −
  have (λw. f w ∗ g w) = (λw. (f w + g w)²/4 − (f w − g w)²/4)
    by (simp only: times-iff-sum-squares)
  also have . . . = (λw. (f w + g w)²∗inverse 4 − (f w + − g w)²∗inverse 4)
    by (simp add: real-diff-def)
  also from f g have . . . ∈ rv M
  proof −
    from f g have (λw. (f w + g w)²) ∈ rv M
      by (simp add: rv-plus-rv rv-square)
    hence (λw. 0+(f w + g w)²∗inverse 4) ∈ rv M
      by (rule affine-rv)
    also from g have (λw. 0 + (g w)∗−1 ) ∈ rv M
      by (rule affine-rv)
    with f have (λw. (f w + − g w)²) ∈ rv M
      by (simp add: rv-plus-rv rv-square)
    hence (λw. 0+(f w + − g w)²∗−inverse 4) ∈ rv M
      by (rule affine-rv)
    ultimately show ?thesis
      by (simp add: rv-plus-rv real-diff-def)
  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** $u\ i\ w \le f\ w$
          by (*rule real-mon-conv-le*)
        also assume $f\ w \le a$
        finally have $u\ i\ w \le a$ **.**
      }

      also
      { assume $\bigwedge i.\ u\ i\ w \le a$
        also from *up* **have** $(\lambda n.\ u\ n\ w)$ ----> $f\ w$
          by (*simp only*: *real-mon-conv*)
        ultimately have $f\ w \le a$
          by (*simp add*: *LIMSEQ-le-const2*)
      }
      ultimately have $(f\ w \le a) = (\forall i.\ u\ i\ w \le a)$ **by** *fast*
    }
    hence $\{w.\ f\ w \le a\} = (\bigcap i.\ \{w.\ u\ i\ w \le a\})$ **by** *fast*
    also
    from *ms u* **have** $\bigwedge i.\ \{w.\ u\ i\ w \le a\} \in sigma(measurable\text{-}sets\ M)$
      by (*simp add*: *rv-le-iff sigma.intros*)
    hence $(\bigcap i.\ \{w.\ u\ i\ w \le a\}) \in sigma(measurable\text{-}sets\ M)$
      by (*rule sigma-Inter*)

with *ms* **have** $(\bigcap i.\ \{w.\ u\ i\ w \leq a\}) \in$ *measurable-sets M*
by (*simp only*: *measure-space-def sigma-sigma-algebra*)
finally have $\{w.\ f\ w \leq a\} \in$ *measurable-sets M* **.**
}
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.

constdefs
nonnegative:: $('a \Rightarrow ('b::\{ord,zero\})) \Rightarrow bool$
nonnegative f $\equiv \forall x.\ 0 \leq f\ x$

positive-part:: $('a \Rightarrow ('b::\{ord,zero\})) \Rightarrow ('a \Rightarrow 'b)$ (*pp*)
pp f x $==$ *if* $0 \leq f(x)$ *then f x else 0*

negative-part:: $('a \Rightarrow ('b::\{ord,zero,minus\})) \Rightarrow ('a \Rightarrow 'b)$ (*np*)
np f x $==$ *if* $0 \leq f(x)$ *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 abs-minus-eqI2 abs-eqI1*)

lemma *nn-pp-np*: **assumes** *nonnegative f*
shows $pp\ f = f$ **and** $np\ f = (\lambda t.\ 0)$ **using** *prems*
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** *prems*
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*)
also note *fm* **also**
have $\{w.\ np\ f\ w \leq a\} = \{w.\ -a \leq f\ w\}$
by (*auto simp add*: *negative-part-def*)
moreover from *ms f* **have** $\ldots \in$ *measurable-sets M*
by (*simp add*: *rv-ge-iff*)
ultimately show *?thesis* **by** *simp*
next
case *False*
hence $\{w.\ pp\ f\ w \leq a\} = \{\}$
by (*auto simp add*: *positive-part-def*)
also from *False* **have** $\{w.\ np\ f\ w \leq a\} = \{\}$
by (*auto simp add*: *negative-part-def*)
moreover from *ms* **have** $\{\} \in$ *measurable-sets M*
by (*simp add*: *measure-space-def sigma-algebra-def*)
ultimately show *?thesis* **by** *simp*
qed

} **with** *ms* **show** $pp\ f \in rv\ M$ **and** $np\ f \in rv\ M$
by (*auto simp add*: *rv-le-iff*)
qed

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

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 thesis. 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 [2]. 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 ([3] p. 172) is strikingly short. For nonnegative measurable functions f:

> $\int f d\mu = sup \sum_i [inf_{\omega \in A_i} f(w)] \mu(A_i)$.
>
> 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" ([3] 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.

consts
integral-set:: $('a\ set\ set * ('a\ set \Rightarrow real)) \Rightarrow (('a \Rightarrow real) * real)\ set$

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

The technique is also encountered in the *Finite-Set* theory from the Isabelle library. It is used there to define the *setsum* 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 I am still not sure if there is a proof or a counter-example or neither.

3.2 The three-step approach

theory *Integral = RealRandVar+SetsumThms*:

Having learnt from my failures, we take the safe and clean way of Heinz Bauer [2]. 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. This theory uses a collection of lemmata on the previously mentioned *setsum* operator, put together in the appendix A.2. Occasionally, a theorem will be trailed by the **sorry** keyword. This interactive Isabelle command indicates that the respective fact has not yet been proven formally due to lack of time.

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, the integral operator needs to take a tuple, consisting of measurable sets and measure, as his 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. At least one element could be preserved from the last-mentioned trial, namely the use of inductive set definitions. This one constructs simple function integral sets.

consts
$\mathit{sfis}{::}\ ('a \Rightarrow \mathit{real}) \Rightarrow ('a\ \mathit{set}\ \mathit{set} * ('a\ \mathit{set} \Rightarrow \mathit{real})) \Rightarrow \mathit{real}\ \mathit{set}$
inductive *sfis f M*
intros
$\mathit{base}{:}\ \llbracket f = (\lambda t.\ \sum i{\in}(S{::}\mathit{nat}\ \mathit{set}).\ x\ i * \chi(A\ i)\ t);$
$\forall\, i \in S.\ A\ i \in \mathit{measurable\text{-}sets}\ M;\ \mathit{nonnegative}\ x;\ \mathit{finite}\ S;$
$\forall\, i{\in}S.\ \forall\, j{\in}S.\ i \neq j \longrightarrow A\ i \cap A\ j = \{\};\ (\bigcup i{\in}S.\ A\ i) = \mathit{UNIV}\rrbracket$
$\Longrightarrow (\sum i{\in}S.\ x\ i * \mathit{measure}\ M\ (A\ i)) \in \mathit{sfis}\ f\ M$

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 and relegated to the appendix A.1. 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 *ms*: *measure-space M* **and** *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 *measure-split*: $measure\ M\ A = (\sum j{\in}R.\ measure\ M\ (A \cap B\ j))$
sorry

theorem *sfis-present*:
assumes *ms*: *measure-space M* **and** *a*: $a \in sfis\ f\ M$ **and** *b*: $b \in sfis\ g\ M$
shows $\exists\ z1\ z2\ C\ K.$
$f = (\lambda t.\ \sum i{\in}(K{::}nat\ set).\ z1\ i * \chi(C\ i)\ t) \wedge g = (\lambda t.\ \sum i{\in}K.\ z2\ i * \chi(C\ i)\ t)$
$\wedge\ a = (\sum i{\in}K.\ z1\ i * measure\ M\ (C\ i))$
$\wedge\ b = (\sum i{\in}K.\ z2\ i * measure\ M\ (C\ i))$
$\wedge\ finite\ K \wedge (\forall i{\in}K.\ \forall j{\in}K.\ i \neq j \longrightarrow C\ i \cap C\ j = \{\})$
$\wedge\ (\forall i \in K.\ C\ i \in measurable\text{-}sets\ M) \wedge (\bigcup i{\in}K.\ C\ i) = UNIV$
$\wedge\ nonnegative\ z1 \wedge nonnegative\ z2$
using *a*
proof *cases*
case (*base A R x*)
show *?thesis* **using** *b*
proof *cases*
case (*base B S y*)

from *prems* **have** *ms*: *measure-space M*
and *f*: $f = (\lambda t.\ \sum i{\in}(R{::}nat\ set).\ x\ i * \chi(A\ i)\ t)$
and *a*: $a = (\sum i{\in}R.\ x\ i * measure\ M\ (A\ i))$
and *Ams*: $\forall i \in R.\ A\ i \in measurable\text{-}sets\ M$
and *R*: *finite R* **and** *Adis*: $\forall i{\in}R.\ \forall j{\in}R.\ i \neq j \longrightarrow A\ i \cap A\ j = \{\}$
and *Aun*: $(\bigcup i{\in}R.\ A\ i) = UNIV$
and *g*: $g = (\lambda t.\ \sum i{\in}(S{::}nat\ set).\ y\ i * \chi(B\ i)\ t)$
and *b*: $b = (\sum j{\in}S.\ y\ j * measure\ M\ (B\ j))$
and *Bms*: $\forall i \in S.\ B\ i \in measurable\text{-}sets\ M$
and *S*: *finite S*
and *Bdis*: $\forall i{\in}S.\ \forall j{\in}S.\ i \neq j \longrightarrow B\ i \cap B\ j = \{\}$
and *Bun*: $(\bigcup i{\in}S.\ B\ i) = UNIV$
and *x*: *nonnegative x* **and** *y*: *nonnegative y*
by *simp*

def $C \equiv (\lambda(i,j).\ A\ i \cap B\ j) \circ n\text{-}to\text{-}n2$
def $z1 \equiv (\lambda k.\ x\ (fst\ (n\text{-}to\text{-}n2\ k)))$
def $z2 \equiv (\lambda k.\ y\ (snd\ (n\text{-}to\text{-}n2\ k)))$
def $K \equiv \{k.\ \exists i{\in}R.\ \exists j{\in}S.\ k = n2\text{-}to\text{-}n\ (i,j)\}$
def $G \equiv (\lambda i.\ (\lambda j.\ n2\text{-}to\text{-}n\ (i,j))\ `\ S)$
def $H \equiv (\lambda j.\ (\lambda i.\ n2\text{-}to\text{-}n\ (i,j))\ `\ R)$

{ fix t
{ fix i
from *Bun S Bdis* **have** $\chi(A\ i)\ t = (\sum j{\in}S.\ \chi(A\ i \cap B\ j)\ t)$
by (*rule char-split*)
hence $x\ i * \chi(A\ i)\ t = (\sum j{\in}S.\ x\ i * \chi(A\ i \cap B\ j)\ t)$
by (*simp add*: *setsum-times-real*)
also
{ fix j
have $S{=}S$ **and**
$x\ i * \chi(A\ i \cap B\ j)\ t = (let\ k{=}n2\text{-}to\text{-}n(i{,}j)\ in\ z1\ k * \chi(C\ k)\ t)$
by (*auto simp add*: *C-def z1-def Let-def n2-to-n-inj n-to-n2-def*)
}
hence $\ldots = (\sum j{\in}S.\ let\ k{=}n2\text{-}to\text{-}n\ (i{,}j)\ in\ z1\ k * \chi(C\ k)\ t)$
by (*rule setsum-cong*)
also from S **have** $\ldots = (\sum k{\in}(G\ i).\ z1\ k * \chi(C\ k)\ t)$
by (*simp add*: *G-def Let-def setsum-image o-def*)
finally have *eq*: $x\ i * \chi(A\ i)\ t = (\sum k{\in}\ G\ i.\ z1\ k * \chi(C\ k)\ t)$ **.**

from *ms Bun S Bdis* **have**
$measure\ M\ (A\ i) = (\sum j{\in}S.\ measure\ M\ (A\ i \cap B\ j))$
by (*rule measure-split*)
hence $x\ i * measure\ M\ (A\ i) = (\sum j{\in}S.\ x\ i * measure\ M\ (A\ i \cap B\ j))$
by (*simp add*: *setsum-times-real*)

also
{ fix j
have $S{=}S$ **and** $x\ i * measure\ M\ (A\ i \cap B\ j) =$
$(let\ k{=}n2\text{-}to\text{-}n(i{,}j)\ in\ z1\ k * measure\ M\ (C\ k))$
by (*auto simp add*: *C-def z1-def Let-def n2-to-n-inj n-to-n2-def*)
}

hence $\ldots = (\sum j{\in}S.\ let\ k{=}n2\text{-}to\text{-}n\ (i{,}j)\ in\ z1\ k * measure\ M\ (C\ k))$
by (*rule setsum-cong*)

also from S **have** $\ldots = (\sum k{\in}(G\ i).\ z1\ k * measure\ M\ (C\ k))$
by (*simp add*: *G-def Let-def setsum-image o-def*)

finally have
eq2: $x\ i * measure\ M\ (A\ i) = (\sum k{\in}(G\ i).\ z1\ k * measure\ M\ (C\ k))$ **.**

from S **have** G: *finite* $(G\ i)$
by (*simp add*: *finite-imageI G-def*)

{ fix k **assume** $k \in G\ i$
then obtain j **where** *kij*: $k{=}n2\text{-}to\text{-}n\ (i{,}j)$
by (*auto simp only*: *G-def*)
{
fix *i2* **assume** *i2*: $i2 \neq i$

```
      { fix k2 assume k2 ∈ G i2
        then obtain j2 where kij2: k2=n2-to-n (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 n2-to-n-inj have  n2-to-n (i2,j2) ≠ n2-to-n (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 eq2 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 eq2: ⋀i. x i * measure M (A i) =
  (∑k∈(G i). z1 k * measure M (C k))
  and Gdis: ⋀i j. i ≠ j ⟹ G i ∩ G j = {} .
from eq eq2 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: setsum-UN-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: setsum-times-real Int-commute)
  also
  { fix i
    have R=R and
      y j * χ(A i ∩ B j) t = (let k=n2-to-n(i,j) in z2 k * χ(C k) t)
      by (auto simp add: C-def z2-def Let-def n2-to-n-inj n-to-n2-def)
  }
  hence ... = (∑i∈R. let k=n2-to-n (i,j) in z2 k * χ(C k) t)
    by (rule setsum-cong)
  also from R have ... = (∑k∈(H j). z2 k * χ(C k) t)
    by (simp add: H-def Let-def setsum-image o-def)
  finally have eq: y j * χ(B j) t = (∑k∈ H j. z2 k * χ(C k) t) .
```

```
    from ms Aun R Adis have measure M (B j) =
      (∑ i∈R. measure M (B j ∩ A i))
      by (rule measure-split)
    hence y j ∗ measure M (B j) = (∑ i∈R. y j ∗ measure M (A i ∩ B j))
      by (simp add: setsum-times-real Int-commute)
    also
    { fix i
      have R=R and y j ∗ measure M (A i ∩ B j) =
        (let k=n2-to-n(i,j) in z2 k ∗ measure M (C k))
        by (auto simp add: C-def z2-def Let-def n2-to-n-inj n-to-n2-def)
    }
    hence . . . = (∑ i∈R. let k=n2-to-n(i,j) in z2 k ∗ measure M (C k))
      by (rule setsum-cong)
    also from R have . . . = (∑ k∈(H j). z2 k ∗ measure M (C k))
      by (simp add: H-def Let-def setsum-image o-def)
    finally have eq2:
      y j ∗ measure M (B j) = (∑ k∈(H j). z2 k ∗ measure M (C k)) .

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

    { fix k assume k ∈ H j
      then obtain i where kij: k=n2-to-n (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=n2-to-n (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 n2-to-n-inj have  n2-to-n (i2,j2) ≠ n2-to-n (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 eq2 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 eq2: ⋀j. y j ∗ measure M (B j) =
    (∑ k∈(H j). z2 k ∗ measure M (C k))
    and Hdis: ⋀i j. i ≠ j ⟹ H i ∩ H j = {} .
```

from *eq g* **have** $g\ t = (\sum j{\in}S.\ (\sum k{\in}H\ j.\ z2\ k * \chi(C\ k)\ t))$
by *simp*
also from *eq2 b* **have** $b = (\sum j{\in}S.\ (\sum k{\in}(H\ j).\ z2\ k * measure\ M\ (C\ k)))$
by *simp*
moreover have $K = (\bigcup j{\in}S.\ H\ j)$
by (*auto simp add*: *K-def H-def*)
moreover note *H Hdis S*
ultimately have *g*: $g\ t = (\sum k{\in}K.\ z2\ k * \chi(C\ k)\ t)$ **and** *K*: *finite K*
and *b*: $b = (\sum k{\in}K.\ z2\ k * measure\ M\ (C\ k))$
by (*auto simp add*: *setsum-UN-disjoint*)

{ **fix** *i*
from *Bun* **have** $(\bigcup k{\in}G\ i.\ C\ k) = A\ i$
by (*simp add*: *G-def C-def n-to-n2-def n2-to-n-inj*)
}
with *Aun* **have** $(\bigcup i{\in}R.\ (\bigcup k{\in}G\ i.\ C\ k)) = UNIV$
by *simp*
hence $(\bigcup k{\in}(\bigcup i{\in}R.\ G\ i).\ C\ k) = UNIV$
by *simp*
with *KG* **have** *Kun*: $(\bigcup k{\in}K.\ C\ k) = UNIV$
by *simp*

note *f g a b Kun K*
}

hence *f*: $f = (\lambda t.\ (\sum k{\in}K.\ z1\ k * \chi(C\ k)\ t))$
and *g*: $g = (\lambda t.\ (\sum k{\in}K.\ z2\ k * \chi(C\ k)\ t))$
and *a*: $a = (\sum k{\in}K.\ z1\ k * measure\ M\ (C\ k))$
and *b*: $b = (\sum k{\in}K.\ z2\ k * measure\ M\ (C\ k))$
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** $k1{\in}K$ **and** $k2{\in}K$ **and** *diff*: $k1 \neq k2$
with *K-def* **obtain** *i1 j1 i2 j2* **where**
RS: $i1 \in R \wedge i2 \in R \wedge j1 \in S \wedge j2 \in S$
and *k1*: $k1 = n2\text{-}to\text{-}n\ (i1,j1)$ **and** *k2*: $k2 = n2\text{-}to\text{-}n\ (i2,j2)$
by *auto*

with *diff* **have** $(i1,j1) \neq (i2,j2)$
by *auto*

with *RS Adis Bdis k1 k2* **have** $C\ k1 \cap C\ k2 = \{\}$
by (*simp add*: *C-def n-to-n2-def n2-to-n-inj*) *fast*
}
moreover

{ fix *k* **assume** $k \in K$
with *K-def* **obtain** *i j* **where** *R*: $i \in R$ **and** *S*: $j \in S$
and *k*: $k = \textit{n2-to-n}\ (i,j)$
by *auto*
with *Ams Bms* **have** $A\ i \in \textit{measurable-sets}\ M$ **and** $B\ j \in \textit{measurable-sets}\ M$
by *auto*
with *ms* **have** $A\ i \cap B\ j \in \textit{measurable-sets}\ M$
by (*simp add*: *measure-space-def sigma-algebra-inter*)
with *k* **have** $C\ k \in \textit{measurable-sets}\ M$
by (*simp add*: *C-def n-to-n2-def n2-to-n-inj*)
}
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*: $a \in \textit{sfis}\ f\ M$ **and** *b*: $b \in \textit{sfis}\ g\ M$
shows *sfis-add*: $a+b \in \textit{sfis}\ (\lambda w.\ f\ w + g\ w)\ M$
proof −
from *prems* **have**
$\exists\ z1\ z2\ C\ K.\ f = (\lambda t.\ \sum i{\in}(K{::}\textit{nat set}).\ z1\ i * \chi(C\ i)\ t) \wedge$
$g = (\lambda t.\ \sum i{\in}K.\ z2\ i * \chi(C\ i)\ t) \wedge a = (\sum i{\in}K.\ z1\ i * \textit{measure}\ M\ (C\ i))$
$\wedge\ b = (\sum i{\in}K.\ z2\ i * \textit{measure}\ M\ (C\ i))$
$\wedge\ \textit{finite}\ K \wedge (\forall i{\in}K.\ \forall j{\in}K.\ i \neq j \longrightarrow C\ i \cap C\ j = \{\})$
$\wedge\ (\forall i \in K.\ C\ i \in \textit{measurable-sets}\ M) \wedge (\bigcup i{\in}K.\ C\ i) = \textit{UNIV}$
$\wedge\ \textit{nonnegative}\ z1 \wedge \textit{nonnegative}\ z2$
by (*rule sfis-present*)

then obtain *z1 z2 C K* **where** *f*: $f = (\lambda t.\ \sum i{\in}(K{::}\textit{nat set}).\ z1\ i * \chi(C\ i)\ t)$
and *g*: $g = (\lambda t.\ \sum i{\in}K.\ z2\ i * \chi(C\ i)\ t)$
and *a2*: $a = (\sum i{\in}K.\ z1\ i * \textit{measure}\ M\ (C\ i))$
and *b2*: $b = (\sum i{\in}K.\ z2\ i * \textit{measure}\ M\ (C\ i))$
and *CK*: $\textit{finite}\ K \wedge (\forall i{\in}K.\ \forall j{\in}K.\ i \neq j \longrightarrow C\ i \cap C\ j = \{\}) \wedge$
$(\forall i{\in}K.\ C\ i \in \textit{measurable-sets}\ M) \wedge \textit{UNION}\ K\ C = \textit{UNIV}$
and *z1*: *nonnegative z1* **and** *z2*: *nonnegative z2*
by *auto*

{ **fix** t
  from $f\ g$ **have**
    $f\ t + g\ t = (\sum i{\in}K.\ z1\ i * \chi(C\ i)\ t) + (\sum i{\in}K.\ z2\ i * \chi(C\ i)\ t)$
    by *simp*
  also have $\ldots = (\sum i{\in}K.\ z1\ i * \chi(C\ i)\ t + z2\ i * \chi(C\ i)\ t)$
    by (*rule setsum-addf*[*THEN sym*])
  also have $\ldots = (\sum i{\in}K.\ (z1\ i + z2\ i) * \chi(C\ i)\ t)$
    by (*simp add*: *real-add-mult-distrib*)
  finally have $f\ t + g\ t = (\sum i{\in}K.\ (z1\ i + z2\ i) * \chi(C\ i)\ t)$ **.**
}

also
{ **fix** t
  from *z1* **have** $0 \le z1\ t$
    by (*simp add*: *nonnegative-def*)
  also from *z2* **have** $0 \le z2\ t$
    by (*simp add*: *nonnegative-def*)
  ultimately have $0 \le z1\ t + z2\ t$
    by (*rule real-le-add-order*)
}

hence *nonnegative* $(\lambda w.\ z1\ w + z2\ w)$
  by (*simp add*: *nonnegative-def ext*)
moreover note *CK*
ultimately have
  $(\sum i{\in}K.\ (z1\ i + z2\ i) * measure\ M\ (C\ i)) \in sfis\ (\lambda w.\ f\ w + g\ w)\ M$
  by (*auto simp add*: *sfis.base*)
also
from *a2 b2* **have** $a{+}b = (\sum i{\in}K.\ (z1\ i + z2\ i) * measure\ M\ (C\ i))$
  by (*simp add*: *setsum-addf*[*THEN sym*] *real-add-mult-distrib*)
ultimately show *?thesis* **by** *simp*
qed

lemma assumes *ms*: *measure-space* M **and** *a*: $a \in sfis\ f\ M$
  and *b*: $b \in sfis\ g\ M$ **and** *fg*: $f{\le}g$
  shows *sfis-mono*: $a \le b$
proof −

from *ms a b* **have**
  $\exists\ z1\ z2\ C\ K.\ f = (\lambda t.\ \sum i{\in}(K{::}nat\ set).\ z1\ i * \chi(C\ i)\ t) \wedge$
  $g = (\lambda t.\ \sum i{\in}K.\ z2\ i * \chi(C\ i)\ t) \wedge a = (\sum i{\in}K.\ z1\ i * measure\ M\ (C\ i))$
  $\wedge\ b = (\sum i{\in}K.\ z2\ i * measure\ M\ (C\ i))$
  $\wedge\ finite\ K \wedge (\forall i{\in}K.\ \forall j{\in}K.\ i \ne j \longrightarrow C\ i \cap C\ j = \{\})$
  $\wedge\ (\forall i \in K.\ C\ i \in measurable\text{-}sets\ M) \wedge (\bigcup i{\in}K.\ C\ i) = UNIV$
  $\wedge\ nonnegative\ z1 \wedge nonnegative\ z2$
  by (*rule sfis-present*)

then obtain *z1 z2 C K* **where** *f*: $f = (\lambda t.\ \sum i{\in}(K{::}nat\ set).\ z1\ i * \chi(C\ i)\ t)$
and *g*: $g = (\lambda t.\ \sum i{\in}K.\ z2\ i * \chi(C\ i)\ t)$
and *a2*: $a = (\sum i{\in}K.\ z1\ i * measure\ M\ (C\ i))$
and *b2*: $b = (\sum i{\in}K.\ z2\ i * measure\ M\ (C\ i))$
and *K*: *finite K* **and** *dis*: $(\forall i{\in}K.\ \forall j{\in}K.\ i \neq j \longrightarrow C\ i \cap C\ j = \{\})$
and *Cms*: $(\forall i{\in}K.\ C\ i \in measurable\text{-}sets\ M)$ **and** *Cun*: *UNION K C = UNIV*
by *auto*

{ fix *i* **assume** *iK*: $i \in K$
{ assume $C\ i \neq \{\}$
then obtain *t* **where** *ti*: $t \in C\ i$
by *auto*
hence $z1\ i = z1\ i * \chi(C\ i)\ t$
by (*simp add*: *characteristic-function-def*)
also
from *dis iK ti* **have** $K{-}\{i\} = K{-}\{i\}$
and $\bigwedge x.\ x \in K{-}\{i\} \Longrightarrow z1\ x * \chi(C\ x)\ t = 0$
by (*auto simp add*: *characteristic-function-def*)
hence $0 = (\sum k{\in}K{-}\{i\}.\ z1\ k * \chi(C\ k)\ t)$
by (*simp only*: *setsum-0 setsum-cong*)
with *K iK* **have** $z1\ i * \chi(C\ i)\ t = (\sum k{\in}K.\ z1\ k * \chi(C\ k)\ t)$
by (*simp add*: *setsum-diff-real*)
also
from *fg f g* **have** $(\sum i{\in}K.\ z1\ i * \chi(C\ i)\ t) \le (\sum i{\in}K.\ z2\ i * \chi(C\ i)\ t)$
by (*simp add*: *le-fun-def*)
also
from *dis iK ti* **have** $K{-}\{i\} = K{-}\{i\}$
and $\bigwedge x.\ x \in K{-}\{i\} \Longrightarrow z2\ x * \chi(C\ x)\ t = 0$
by (*auto simp add*: *characteristic-function-def*)
hence $0 = (\sum k{\in}K{-}\{i\}.\ z2\ k * \chi(C\ k)\ t)$
by (*simp only*: *setsum-0 setsum-cong*)
with *K iK* **have** $(\sum k{\in}K.\ z2\ k * \chi(C\ k)\ t) = z2\ i * \chi(C\ i)\ t$
by (*simp add*: *setsum-diff-real*)
also
from *ti* **have** $\ldots = z2\ i$
by (*simp add*: *characteristic-function-def*)
finally
have $z1\ i \le z2\ i$ **.**
}
hence *h*: $C\ i \neq \{\} \Longrightarrow z1\ i \le z2\ i$ **.**

have $z1\ i * measure\ M\ (C\ i) \le z2\ i * measure\ M\ (C\ i)$
proof (*cases* $C\ i \neq \{\}$)
case *False*

with *ms* **show** *?thesis*
by (*auto simp add*: *measure-space-def positive-def*)

next
case *True*
with *h* **have** $z1\ i \le z2\ i$
by *fast*
also from *iK ms Cms* **have** $0 \le measure\ M\ (C\ i)$
by (*auto simp add*: *measure-space-def positive-def*)
ultimately show *?thesis*
by (*simp add*: *real-mult-le-mono1*)
qed
}
with *a2 b2* **show** *?thesis*
by (*simp add*: *setsum-mono-real*)
qed

lemma *sfis-unique*:
assumes *ms*: *measure-space M* **and** *a*: $a \in$ *sfis f M* **and** *b*: $b \in$ *sfis f M*
shows $a=b$ **using** *prems*
proof −
have $f \le f$ **by** (*simp add*: *le-fun-def*)
with *prems* **have** $a \le b$ **and** $b \le a$
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 setsums, 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*: $A \in$ *measurable-sets M*
shows *measure M A* $\in$ *sfis* $\chi A\ M$

lemma *sfis-times*:
assumes *a*: $a \in$ *sfis f M* **and** *z*: $0 \le z$
shows $z*a \in$ *sfis* $(\lambda w.\ z*f\ w)\ M$

lemma assumes *ms*: *measure-space M*
and *a*: $\forall i \in S.\ a\ i \in$ *sfis* $(f\ i)\ M$ **and** *S*: *finite S*
shows *sfis-setsum*: $(\sum i \in S.\ a\ i) \in$ *sfis* $(\lambda t.\ \sum i \in S.\ f\ i\ t)\ M$

lemma *sfis-intro*:
assumes *ms*: *measure-space M* **and** *Ams*: $\forall\, i \in S.\ A\ i \in$ *measurable-sets M*
and *nn*: *nonnegative x* **and** *S*: *finite S*
shows $(\sum i{\in}S.\ x\ i * measure\ M\ (A\ i)) \in$
sfis $(\lambda t.\ \sum i{\in}(S{::}nat\ set).\ x\ i * \chi(A\ i)\ t)\ M$
proof −
{ **fix** *i* **assume** *iS*: $i \in S$
with *ms Ams* **have** $measure\ M\ (A\ i) \in sfis\ \chi(A\ i)\ M$
by (*simp add*: *sfis-char*)
with *nn* **have** $x\ i * measure\ M\ (A\ i) \in sfis\ (\lambda t.\ x\ i * \chi(A\ i)\ t)\ M$
by (*simp add*: *nonnegative-def sfis-times*)
}
with *ms S* **show** *?thesis*
by (*simp add*: *sfis-setsum*)
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 *sfis-rv*:
assumes *ms*: *measure-space M* **and** *f*: $a \in$ *sfis f M*
shows $f \in rv\ M$
sorry

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*: $\bigwedge z.\ [\![0{<}z;\ z{<}1]\!] \Longrightarrow z * r \le y$
shows $r \le y$
sorry

```
lemma sfis-mon-conv-mono:
  assumes uf: u↑f and xu: ⋀n. x n ∈ sfis (u n) M and xy: x↑y
  and sr: r ∈ sfis s M and sf: s ≤ f and ms: measure-space M
  shows r ≤ y using sr
proof cases
  case (base A S a)

  { 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 prems 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 R c)
        { 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 setsum-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: setsum-times-real[THEN sym])
          also
          from prems have nonnegative s
            by (simp add: sfis-nn)
```

```
      with nnu B-def prems
      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 prems 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 set-mon-conv, rule)
  { fix n
    from uf have um: u n ≤ u (Suc n)
      by (simp add: realfun-mon-conv)
    {
      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 $0 \leq u\ n\ t$
by (*simp add*: *nonnegative-def*)
finally show *?thesis*
by *rule*

next
case *False*
from *sr* **have** *nonnegative s*
by(*rule sfis-nn*)
hence $0 \leq s\ t$
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*: *real-mult-less-mono1*)
also from *sf* **have** $\ldots \leq f\ t$
by (*simp add*: *le-fun-def*)
finally have $z * s\ t < f\ t$ **.**

also from *uf* **have** $(\lambda m.\ u\ m\ t)\uparrow f\ t$
by (*simp add*: *realfun-mon-conv-iff*)
ultimately have $\exists\, n.\forall\, m.\ n \leq m \longrightarrow z*s\ t < u\ m\ t$
by (*simp add*: *real-mon-conv-outgrow*)
hence $\exists\, n.\ z*s\ t < u\ n\ t$
by *fast*
thus *?thesis*
by (*auto simp add*: *order-less-le*)
qed

hence $\exists\, n.\ t \in B\ n$
by (*simp add*: *B-def*)
hence $t \in (\bigcup n.\ B\ n)$
by *fast*
}
thus $UNIV=(\bigcup n.\ B\ n)$
by *fast*
qed

{ fix *j* **assume** *jS*: $j \in S$
note *ms*
also

from *jS ABms* **have** $\bigwedge n.\ A\ j \cap B\ n \in$ *measurable-sets M*
by *auto*
moreover
from *Bun* **have** $(\lambda n.\ A\ j \cap B\ n)\uparrow(A\ j)$
by (*auto simp add*: *set-mon-conv*)
ultimately have $(\lambda n.$ *measure M* $(A\ j \cap B\ n))$ ––––> *measure M* $(A\ j)$
by (*rule measure-mon-conv*)

hence $(\lambda n.\ a\ j *$ *measure M* $(A\ j \cap B\ n))$ ––––> $a\ j *$ *measure M* $(A\ j)$
by (*simp add*: *LIMSEQ-const LIMSEQ-mult*)
}
hence $(\lambda n.\ \sum j{\in}S.\ a\ j *$ *measure M* $(A\ j \cap B\ n))$
––––> $(\sum j{\in}S.\ a\ j *$ *measure M* $(A\ j))$
by (*rule limseq-setsum*)
hence $(\lambda n.\ z* (\sum j{\in}S.\ a\ j *$ *measure M* $(A\ j \cap B\ n)))$
––––> $z*(\sum j{\in}S.\ a\ j *$ *measure M* $(A\ j))$
by (*simp add*: *LIMSEQ-const LIMSEQ-mult*)

with *1 prems* **have** $z{*}r \le y$
by (*auto simp add*: *LIMSEQ-le real-mon-conv*)
}
hence *zr*: $\bigwedge z.\ [\![0{<}z;\ z{<}1]\!] \Longrightarrow z * r \le y$ **.**
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.

consts
nnfis:: $('a \Rightarrow real) \Rightarrow ('a\ set\ set * ('a\ set \Rightarrow real)) \Rightarrow real\ set$
inductive *nnfis f M*
intros
base: $[\![u{\uparrow}f;\ \bigwedge n.\ x\ n \in$ *sfis* $(u\ n)\ M;\ x{\uparrow}y]\!] \Longrightarrow y \in$ *nnfis f M*

lemma *sfis-nnfis*:
assumes *s*: $a \in$ *sfis f M*
shows $a \in$ *nnfis f M*

lemma *nnfis-times*:
assumes *ms*: *measure-space M* **and** *a*: $a \in$ *nnfis f M* **and** *nn*: $0{\le}z$
shows $z{*}a \in$ *nnfis* $(\lambda w.\ z{*}f\ w)\ M$

lemma *nnfis-add*:
  assumes *ms*: *measure-space M* **and** *a*: $a \in$ *nnfis f M* **and** *b*: $b \in$ *nnfis g M*
  shows $a+b \in$ *nnfis* $(\lambda w.\ f\ w + g\ w)$ *M*

lemma assumes *ms*: *measure-space M* **and** *a*: $a \in$ *nnfis f M*
  and *b*: $b \in$ *nnfis g M* **and** *fg*: $f \le g$
  shows *nnfis-mono*: $a \le b$ **using** *a*
proof (*cases*)
  case (*base u x y*)
  from *b* **show** *?thesis*
  proof (*cases*)
    case (*base v r s*)
    { fix *m*
      from *prems* **have** $u\ m \le f$
        by (*simp add*: *realfun-mon-conv-le*)
      also note *fg* **finally have** $u\ m \le g$ **.**
      with *prems* **have** $v \uparrow g$ **and** $\bigwedge n.\ r\ n \in$ *sfis* $(v\ n)$ *M* **and** $r \uparrow s$
        and $x\ m \in$ *sfis* $(u\ m)$ *M* **and** $u\ m \le g$ **and** *measure-space M*
        by *simp-all*
      hence $x\ m \le s$
        by (*rule sfis-mon-conv-mono*)
    }
    with *prems* **have** $y \le s$
      by (*auto simp add*: *real-mon-conv LIMSEQ-le-const2*)
    thus *?thesis* **using** *prems*
      by *simp*
  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. 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.

constdefs
  upclose:: $('a \Rightarrow real) \Rightarrow ('a \Rightarrow real) \Rightarrow ('a \Rightarrow real)$
  upclose f g $\equiv (\lambda t.\ max\ (f\ t)\ (g\ t))$

consts
  mon-upclose-help :: $nat \Rightarrow (nat \Rightarrow nat \Rightarrow 'a \Rightarrow real) \Rightarrow nat \Rightarrow ('a \Rightarrow real)$ (*muh*)

primrec
  $muh\ 0\ u\ m = u\ m\ 0$
  $muh\ (Suc\ n)\ u\ m = upclose\ (u\ m\ (Suc\ n))\ (muh\ n\ u\ m)$

constdefs
  mon-upclose :: $(nat \Rightarrow nat \Rightarrow 'a \Rightarrow real) \Rightarrow nat \Rightarrow ('a \Rightarrow real)$ (*mu*)
  $mu\ u\ m \equiv muh\ m\ u\ m$

lemma *upclose-sfis*:
  assumes *f*: $a \in sfis\ f\ M$ **and** *g*: $b \in sfis\ g\ M$ **and** *ms*: *measure-space M*
  shows $\exists c.\ c \in sfis\ (upclose\ f\ g)\ M$
  sorry

lemma *mu-sfis*:
  assumes *u*: $\bigwedge m\ n.\ \exists a.\ a \in sfis\ (u\ m\ n)\ M$ **and** *ms*: *measure-space M*
  shows $\exists c.\ \forall m.\ c\ m \in sfis\ (mu\ u\ m)\ M$
sorry

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 $\bigwedge m.\ muh\ n\ u\ m \le f\ n$
    proof (*induct n*)
      case (*0 m*)
      from *uf* **have** $u\ m\ 0 \le f\ 0$
        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 real-mon-conv)
      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) arith
  }
  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 real-mon-conv)

  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 ∀n. muh n u m t ≤ muh (Suc n) u m t
          by simp
        hence muh n u m t ≤ muh (n+(m−n)) u m t
          by (rule lemma-f-mono-add)
        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 fast
    also from uf have (λm. u m n t) −−−−> f n t
      by (simp add: realfun-mon-conv-iff real-mon-conv)
    moreover
```

from *conv* **have** $(\lambda n.\ mu\ u\ n\ t)$ ----> l
by (*simp add*: *real-mon-conv*)
ultimately have $f\ n\ t \leq l$
by (*simp add*: *LIMSEQ-le*)
}
also from *fh* **have** $(\lambda n.\ f\ n\ t)$ ----> $h\ t$
by (*simp add*: *realfun-mon-conv-iff real-mon-conv*)
ultimately have $h\ t \leq l$
by (*simp add*: *LIMSEQ-le-const2*)

with *lh* **have** $l = h\ t$
by *simp*
with *conv* **have** $(\lambda n.\ mu\ u\ n\ t)\uparrow(h\ t)$
by *simp*
}
with *mon-upclose-def* **show** $mu\ u\uparrow h$
by (*simp add*: *realfun-mon-conv-iff*)
qed

theorem *nnfis-mon-conv*:
assumes *fh*: $f\uparrow h$ **and** *xf*: $\bigwedge n.\ x\ n \in nnfis\ (f\ n)\ M$ **and** *xy*: $x\uparrow y$
and *ms*: *measure-space M*
shows $y \in nnfis\ h\ M$
proof –
def $u \equiv (\lambda n.\ SOME\ u.\ u\uparrow(f\ n) \wedge (\forall m.\ \exists a.\ a \in sfis\ (u\ m)\ M))$
{ **fix** n
from *xf*[*of n*] **have** $\exists u.\ u\uparrow(f\ n) \wedge (\forall m.\ \exists a.\ a \in sfis\ (u\ m)\ M)$ (**is** $\exists x.\ ?P\ x$)
proof (*cases*)
case (*base r a b*)
hence $r\uparrow(f\ n)$ **and** $\bigwedge m.\ \exists a.\ a \in sfis\ (r\ m)\ M$ **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 $urev \equiv (\lambda m\ n.\ u\ n\ m)$
ultimately have *uf*: $\bigwedge n.\ (\lambda m.\ urev\ m\ n)\uparrow(f\ n)$
and *sf*: $\bigwedge n\ m.\ \exists a.\ a \in sfis\ (urev\ m\ n)\ M$
by *auto*

from *uf fh* **have** *up*: $mu\ urev\uparrow h$
by (*rule mu-help*)

from *uf fh* **have** *le*: $\bigwedge n.\ mu\ urev\ n \le f\ n$
by (*rule mu-help*)
from *sf ms* **obtain** *c* **where** *sf2*: $\bigwedge m.\ c\ m \in sfis\ (mu\ urev\ m)\ M$
by (*force simp add*: *mu-sfis*)

{ fix *m*
from *sf2* **have** $c\ m \in nnfis\ (mu\ urev\ m)\ M$
by (*rule sfis-nnfis*)
with *ms le*[*of m*] *xf*[*of m*] **have** $c\ m \le x\ m$
by (*simp add*: *nnfis-mono*)
} hence $c \le x$ **by** (*simp add*: *le-fun-def*)
also
{ fix *m* **note** *ms* **also**
from *up* **have** $mu\ urev\ m \le mu\ urev\ (Suc\ m)$
by (*simp add*: *realfun-mon-conv*)
moreover from *sf2* **have** $c\ m \in sfis\ (mu\ urev\ m)\ M$
and $c\ (Suc\ m) \in sfis\ (mu\ urev\ (Suc\ m))\ M$
by *fast*
ultimately have $c\ m \le c\ (Suc\ m)$
by (*simp add*: *sfis-mono*)
}
moreover note *xy*
ultimately have $\exists\, l.\ c\uparrow l \wedge l \le y$
by (*simp add*: *real-mon-conv-dom*)
then obtain *l* **where** *cl*: $c\uparrow l$ **and** *ly*: $l \le y$
by *fast*

from *up sf2 cl* **have** *int*: $l \in nnfis\ h\ M$
by (*rule nnfis.base*)

{ fix *n*
from *fh* **have** $f\ n \le h$
by (*rule realfun-mon-conv-le*)
with *ms xf*[*of n*] *int* **have** $x\ n \le l$
by (*rule nnfis-mono*)
} with *xy* **have** $y \le l$
by (*auto simp add*: *real-mon-conv 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** $a \in$ *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 *nnfis-rv*:
assumes *ms*: *measure-space M* **and** *f*: $a \in$ *nnfis f M*
shows $f \in$ *rv M*
sorry

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

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*: $f \le g$ **and** *nn*: *nonnegative f*
shows *nnfis-dom-conv*: $\exists\, a.\ a \in nnfis\ f\ M \wedge a \le b$ **using** *b*
proof (*cases*)
case (*base v r z*)
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** $u\ n \le f$
by ($rule\ realfun\text{-}mon\text{-}conv\text{-}le$)
also note fg
also
from xu **have** $x\ n \in nnfis\ (u\ n)\ M$
by ($rule\ sfis\text{-}nnfis$)
moreover note $b\ ms$
ultimately have le: $x\ n \le b$
by ($simp\ add$: $nnfis\text{-}mono$)

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

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

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

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

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*≤(*0*::*real*)
    by (*rule real-le-refl*)
  with *ms nn* **have** *0*∗*a* ∈ *nnfis* ($\lambda t.$ *0*∗*f t*) *M*
    by (*rule nnfis-times*)
  with *0* **have** *02*: *0* ∈ *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* ∈ *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 $\int f \,\partial\, M = a$
    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* ($\lambda t.$ *f t* − *g t*) *M* **and** $\int (\lambda t.\ f\ t - g\ t) \,\partial\, M = a - b$
proof −
  from *ms a b* **have** ($\lambda t.$ *f t* − *g t*) ∈ *rv M*
    by (*auto simp only*: *nnfis-rv rv-minus-rv*)
  hence *prv*: *pp* ($\lambda t.$ *f t* − *g t*) ∈ *rv M* **and** *nrv*: *np* ($\lambda t.$ *f t* − *g t*) ∈ *rv M*
    by (*auto simp only*: *pp-np-rv*)

have *nnp*: *nonnegative* (*pp* ($\lambda t.$ *f t* $-$ *g t*))
and *nnn*: *nonnegative* (*np* ($\lambda t.$ *f t* $-$ *g t*))
by (*simp-all add*: *nonnegative-def positive-part-def negative-part-def*)

from *ms a b* **have** *fg*: $a+b \in$ *nnfis* ($\lambda t.$ *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** $0 \le f\ t$ **and** $0 \le g\ t$
by (*simp-all add*: *nonnegative-def*)
hence (*pp* ($\lambda t.$ *f t* $-$ *g t*)) $t \le f\ t + g\ t$ **and** (*np* ($\lambda t.$ *f t* $-$ *g t*)) $t \le f\ t + g\ t$
by (*simp-all add*: *positive-part-def negative-part-def*)
}
hence (*pp* ($\lambda t.$ *f t* $-$ *g t*)) $\le$ ($\lambda t.$ *f t* $+$ *g t*)
and (*np* ($\lambda t.$ *f t* $-$ *g t*)) $\le$ ($\lambda t.$ *f t* $+$ *g t*)
by (*simp-all add*: *le-fun-def*)
with *fg nnf nng prv nrv nnp nnn ms*
have $\exists\, l.\ l \in$ *nnfis* (*pp* ($\lambda t.$ *f t* $-$ *g t*)) $M \wedge l \le a+b$
and $\exists\, k.\ k \in$ *nnfis* (*np* ($\lambda t.$ *f t* $-$ *g t*)) $M \wedge k \le a+b$
by (*auto simp only*: *nnfis-dom-conv*)
then obtain *l k* **where** *l*: $l \in$ *nnfis* (*pp* ($\lambda t.$ *f t* $-$ *g t*)) *M*
and *k*: $k \in$ *nnfis* (*np* ($\lambda t.$ *f t* $-$ *g t*)) *M*
by *auto*
with *ms* **show** *i*: *integrable* ($\lambda t.$ *f t* $-$ *g t*) *M*
by (*auto simp add*: *integrable-def*)

{ fix *t*
have $f\ t - g\ t =$ (*pp* ($\lambda t.$ *f t* $-$ *g t*)) $t -$ (*np* ($\lambda t.$ *f t* $-$ *g t*)) t
by (*rule f-plus-minus*)

hence $f\ t +$ (*np* ($\lambda t.$ *f t* $-$ *g t*)) $t = g\ t +$ (*pp* ($\lambda t.$ *f t* $-$ *g t*)) t
by *arith*
}
hence ($\lambda t.$ *f t* $+$ (*np* ($\lambda t.$ *f t* $-$ *g t*)) *t*) $=$
($\lambda t.$ *g t* $+$ (*pp* ($\lambda t.$ *f t* $-$ *g t*)) *t*)
by (*rule ext*)
also
from *ms a k b l* **have** $a+k \in$ *nnfis* ($\lambda t.$ *f t* $+$ (*np* ($\lambda t.$ *f t* $-$ *g t*)) *t*) *M*
and $b+l \in$ *nnfis* ($\lambda t.$ *g t* $+$ (*pp* ($\lambda t.$ *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** ($THE\ i.\ i \in nnfis\ (pp\ (\lambda t.\ f\ t - g\ t))\ M) = l$
and ($THE\ i.\ i \in nnfis\ (np\ (\lambda t.\ f\ t - g\ t))\ M) = k$
by (*auto simp add*: *nnfis-unique*)
moreover note *i*
ultimately show $\int (\lambda t.\ f\ t - g\ t)\ \partial\ M = a - b$
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 *integrable-rv*:
assumes *int*: *integrable f M*
shows $f \in rv\ M$
sorry

theorem *integral-char*:
assumes *ms*: *measure-space M* **and** *mA*: $A \in measurable\text{-}sets\ M$
shows $\int \chi A\ \partial\ M = measure\ M\ A$ **and** *integrable* $\chi A\ M$

theorem *integral-add*:
assumes *f*: *integrable f M* **and** *g*: *integrable g M*
shows *integrable* $(\lambda t.\ f\ t + g\ t)\ M$
and $\int (\lambda t.\ f\ t + g\ t)\ \partial M = \int f\ \partial M + \int g\ \partial M$
proof −
def $u \equiv (\lambda t.\ pp\ f\ t + pp\ g\ t)$
def $v \equiv (\lambda t.\ np\ f\ t + np\ g\ t)$

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

from *ms pf pg u-def* **have**
u: $pf+pg \in nnfis\ u\ M$
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*: $pg \in nnfis\ (pp\ g)\ M$
  and *ng*: $ng \in nnfis\ (np\ g)\ M$
  by (*auto simp add*: *integrable-def*)

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

also
from *ms pf pg ng nf* **have** $(THE\ i.\ i \in nnfis\ (pp\ f)\ M) = pf$
  and $(THE\ i.\ i \in nnfis\ (np\ f)\ M) = nf$
  and $(THE\ i.\ i \in nnfis\ (pp\ g)\ M) = pg$
  and $(THE\ i.\ i \in nnfis\ (np\ g)\ M) = ng$
  by (*auto simp add*: *nnfis-unique*)
with *f g* **have** $\int\ f\ \partial M = pf - nf$
  and $\int\ g\ \partial M = pg - ng$
  by (*auto simp add*: *integral-def*)

ultimately show *?thesis*
  by *simp*
qed

theorem *integral-times*:
  assumes *int*: *integrable f M*
  shows *integrable* $(\lambda t.\ a{*}f\ t)\ M$ **and** $\int\ (\lambda t.\ a{*}f\ t)\ \partial M = a{*}\int\ f\ \partial M$

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 and may itself serve as a neat example for applying them.

theorem assumes *int*: *integrable f M* **and** *a*: $0<a$
  and *intp*: *integrable* $(\lambda x.\ |f\ x|\ \hat{}\ p)\ M$
  shows *markov-ineq*: $law\ M\ f\ \{a..\} \leq\ \int\ (\lambda x.\ |f\ x|\ \hat{}\ p)\ \partial M\ /\ (a\hat{}p)$
proof −
  from *int* **have** *rv*: $f \in rv\ M$
    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^p)∗law M f {a..} = (a^p)∗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^p)∗∫ χ{w. a ≤ f w} ∂ M
    by (auto simp add: integral-char)
  note eq2 also
  from int2 have int3: integrable (λt. (a^p)∗χ{w. a ≤ f w} t) M
    and eq3: ... = ∫ (λt. (a^p)∗χ{w. a ≤ f w} t) ∂ M
    by (auto simp add: integral-times)
  note eq3 also
  { fix t
    from a have (a^p)∗χ{w. a ≤ f w} t ≤ |f t| ^ p
    proof (cases a ≤ f t)
      case False
      thus ?thesis
        by (simp add: characteristic-function-def)
    next
      case True
      with a have a ^ p ≤ (f t) ^ p
        by (simp add: realpow-le)
      also
      have (f t) ^ p ≤ |(f t) ^ p|
        by arith
      hence (f t) ^ p ≤ |f t| ^ p
        by (simp add: realpow-abs)
      finally
      show ?thesis
        by (simp add: characteristic-function-def)
    qed
  }
  with int3 intp have ... ≤ ∫ (λx. |f x| ^ p) ∂M
    by (simp add: le-fun-def integral-mono)

  also
  from a have 0 < a^p
    by (rule realpow-gt-zero)
  ultimately show ?thesis
    by (simp add: pos-real-le-divide-eq real-mult-commute)
qed

end
```

Chapter 4

Probabilistic Algorithms

To take up a point from the prologue, one major motive for formalizing integration is to formalize expectation. Indeed, the expectation of a random variable is nothing but its integral. This simple fact makes it possible to use all the theorems about integration to manipulate expected values. In the application I chose, only two properties are needed, namely additivity and the Markov inequation. The latter gives rise to the so-called first moment method. Before going into the details of the use case, a concrete probability space is required.

4.1 The probability space

theory *Imported* = *Measure*+*SupInf*:

In this section, no single theorem is truly proven. Instead, as the theory[1] title suggests, it imports some notions that were developed formally by Joe Hurd [11] in the HOL theorem prover. They lay the foundations necessary to reason about functional probabilistic programs. The way Hurd arranges for this is by a monadic notation. Random is modeled by an infinite sequence of boolean values ("coin flips") that is passed around by the algorithms. We introduce the type of sequences to this end, represented as functions from the natural numbers to some basis type, which will always be the booleans in what follows.

types $'a\ seq = nat \Rightarrow\ 'a$

There are some canonical operations on this type, reminiscent of the typical list operations.

[1] The *SupInf* theory merely contains standard definitions of suprema and infima and is therefore omitted.

constdefs
$shd :: {'a}\ seq \Rightarrow {'a}$
$shd\ s \equiv s\ 0$
$stl :: {'a}\ seq \Rightarrow {'a}\ seq$
$stl\ s \equiv s \circ Suc$
$sdest :: {'a}\ seq \Rightarrow ({'a} * {'a}\ seq)$
$sdest\ s \equiv (shd\ s,\ stl\ s)$

consts
$stake :: nat \Rightarrow {'a}\ seq \Rightarrow {'a}\ list$
$sdrop :: nat \Rightarrow {'a}\ seq \Rightarrow {'a}\ seq$
$scons :: {'a} \Rightarrow {'a}\ seq \Rightarrow nat \Rightarrow {'a}$

primrec
$stake\ 0\ s = []$
$stake\ (Suc\ n)\ s = shd\ s\ \#\ stake\ n\ (stl\ s)$

primrec
$sdrop\ 0\ s = s$
$sdrop\ (Suc\ n)\ s = sdrop\ n\ (stl\ s)$

primrec
$scons\ a\ s\ 0 = a$
$scons\ a\ s\ (Suc\ n) = s\ n$

With *sdest*, a first probabilistic program has already been exhibited. In general, randomized algorithms take the stream of random bits as an extra parameter and pass on the unused bits — still an infinite sequence — as the second component of a result pair. This process is characteristic of the so-called state-transformer monad, which is a standard method in pure, i.e. stateless, functional languages. It can be hidden from the user by function abstraction, employing the following two monadic operators.

constdefs
$unit :: {'a} \Rightarrow {'b} \Rightarrow ({'a} * {'b})$
$unit \equiv Pair$

$bind :: ({'a} \Rightarrow ({'b} * {'a})) \Rightarrow ({'b} \Rightarrow {'a} \Rightarrow {'c}) \Rightarrow ({'a} \Rightarrow {'c})$
$bind\ f\ g \equiv (split\ g) \circ f$

While the latter is the monadic equivalent of function composition, the former is used to lift values to the tuple form, just handing on the sequence it is given. In conjunction with *sdest*, any useful probabilistic program may be expressed via these primitives.

The name *unit* is the common term in monad theory and Hurd adopts it for this reason. It has already been given another meaning in Isabelle/HOL though, whereas the corresponding function is known as *Pair*. This being a rather descriptive name, we will stick to it instead.

Prior to generating the probability space of boolean sequences, the notion of such a space has to be defined first. With it comes a change in terminology. In this context, measurable sets are called *events* and measures *probabilities*.

constdefs
prob-space:: ('*a set set* ∗ ('*a set* ⇒ *real*)) ⇒ *bool*
prob-space M ≡ (*measure-space M* ∧ *measure M UNIV* = *1*)

consts
events:: ('*a set set* ∗ ('*a set* ⇒ *real*)) ⇒ '*a set set*
prob:: ('*a set set* ∗ ('*a set* ⇒ *real*)) ⇒ ('*a set* ⇒ *real*)

translations
events == *measurable-sets*
prob == *measure*

The construction of a sensible probability measure and its sigma algebra of measurable sets for boolean sequences — this is known as the Bernoulli space — is a nontrivial feat. One begins by defining more or less obvious probabilities on a set of sets called an algebra. An algebra is almost a sigma algebra, but we require the union condition to hold for two[2] sets only.

This algebra with its premeasure, as it is named in mathematical literature, is then lifted to a sigma algebra including a measure that maintains the original values on the primal sets. The process is performed nonconstructively, involving a weaker version of Carathéodory's extension theorem, which states the existence of such a measure[3]. The technical details are explained in the foregoing source.

consts
prefix-seq:: '*a list* ⇒ '*a seq*
is-prefix:: '*a list* ⇒ '*a list* ⇒ *bool*

primrec
prefix-seq (*h*#*t*) = *scons h* (*prefix-seq t*)

primrec
is-prefix [] *l* = *True*
is-prefix (*x*#*xs*) *l* = (*if l*=[] *then False*
else (*x*=*hd l*) ∧ *is-prefix xs* (*tl l*))

[2] and thus for finitely many

[3] There are stronger forms that permit lifting from semirings, for instance. The most famous application is the construction of the Lebesgue measure.

constdefs
prefix-set:: $'a$ *list* $\Rightarrow$ $'a$ *seq set*
prefix-set $l \equiv \{s.\ stake\ (length\ l)\ s = l\}$

embed:: $'a$ *list list* $\Rightarrow$ $'a$ *seq set*
embed $ll \equiv \bigcup l \in set\ ll.\ prefix\text{-}set\ l$

algebra:: $'a$ *set set* $\Rightarrow$ *bool*
algebra $A ==$
$\{\} \in A \wedge (\forall a \in A.\ -a \in A \wedge (\forall b \in A.\ a \cup b \in A))$

constdefs
bernoulli-algebra:: *bool seq set set*
bernoulli-algebra $\equiv \{A.\ \exists l.\ A = embed\ l\}$

mu0:: *bool list list* $\Rightarrow$ *real* $(\mu 0)$
$\mu 0\ ll \equiv \sum\ l \in set\ ll.\ 2\hat{}(-length\ l)$

mu:: *bool seq set* $\Rightarrow$ *real* (μ)
$\mu\ A \equiv infimum\ \{l.\ embed\ l = A\}\ \mu 0$

lemma *mu-positive*: *positive* (*bernoulli-algebra*,μ)
sorry

lemma *mu-countably-additive*: *countably-additive* (*bernoulli-algebra*,μ)
sorry

lemma *ba-algebra*: *algebra bernoulli-algebra*
sorry

lemma *caratheodory-light*:
$[\![$*algebra* (*measurable-sets M0*); *positive M0*; *countably-additive M0*$]\!]$
$\Longrightarrow \exists M.\ (\forall A.\ A \in$ *measurable-sets M0* $\longrightarrow$ *measure M A* $=$ *measure M0 A*)
$\wedge$ *measurable-sets M* $=$ *sigma* (*measurable-sets M0*) $\wedge$ *measure-space M*
sorry

constdefs
bern:: (*bool seq set set*) $*$ (*bool seq set* $\Rightarrow$ *real*)
bern $\equiv$ *let M0* $=$ (*bernoulli-algebra*,μ) *in*
$\varepsilon\ M.\ (\forall A.\ A \in$ *events M0* $\longrightarrow$ *prob M A* $=$ *prob M0 A*)
$\wedge$ *events M* $=$ *sigma* (*events M0*) $\wedge$ *measure-space M*

P:: *bool seq set* $\Rightarrow$ *real*
P $\equiv$ *prob bern*

Epsilon:: *bool seq set set*
Epsilon ≡ *events bern*

lemma *ms-bern*: *measure-space* (*Epsilon*,*P*)
sorry

lemma *ps-bern*:
prob-space (*Epsilon*,*P*)
sorry

lemma *UNIV-prob*:
P UNIV = *1*
sorry

Joe Hurd goes on developing a handy characterization of probabilistic programs called strong function independence. It amounts to a compositional version of measurability and independence, the latter meaning independence between the first and second component of the result pair. That is, the program may not use the bit-stream it passes back in the calculation of the proper result.

Luckily, the property is compositional in the sense that it holds for any program that is constructed using only the three primitives mentioned previously. Again, it is not necessary to understand the exact definition for our purpose.

constdefs
prefix-cover:: *bool list set* ⇒ *bool*
prefix-cover C ≡ (∀ *l1*∈*C*. ∀ *l2*∈*C*. *l1* ≠ *l2* ⟶ ¬ *is-prefix l1 l2*) ∧
P (⋃*l*∈*C*. *prefix-set l*) = *1*

indep-fn:: (*bool seq* ⇒ (*'a* ∗ (*bool seq*))) *set*
indep-fn ≡ {*f*. (∃ *F*::(*nat*⇒*'a*). *range* (*fst* ∘ *f*) ⊆ *range F*) ∧
(*fst* ∘ *f*) ∈ *measurable Epsilon UNIV* ∧ (*snd* ∘ *f*) ∈ *measurable Epsilon Epsilon*
∧ (∃ *C*. *prefix-cover C* ∧
(∀ *l s*. (*l* ∈ *C* ∧ *s* ∈ *prefix-set l*
⟶ *f s* = (*fst* (*f* (*prefix-seq l*)), *sdrop* (*length l*) *s*))))}

lemma *Pair-indep*: ⋀*a*. (*Pair a*) ∈ *indep-fn*
sorry

lemma *bind-indep*:
⟦*f* ∈ *indep-fn*; ⋀*a*. *g a* ∈ *indep-fn*⟧ ⟹ *bind f g* ∈ *indep-fn*
sorry

lemma *sdest-indep*: $sdest \in indep\text{-}fn$
  sorry

For us, the crucial point of strong functional independence, besides proving measurability for our programs, is the following lemma, which formalizes the idea of independence just explained. It could not be found in Hurd's thesis proper, but in the formal text itself. Moreover, the formulation is slightly different, since predicates are employed instead of sets in the HOL theories. Basically this means exchanging function composition for the inverse image operator.

lemma *indep-fn-prob*:
$$[\![f \in indep\text{-}fn;\ q \in Epsilon]\!] \Longrightarrow$$
$$P\ ((fst \circ f) -` p \cap (snd \circ f) -` q)$$
$$= P\ ((fst \circ f) -` p) * P\ q$$
sorry

To start reasoning about probabilities, we need one of them to begin with. The last lemma in this section provides this humble starting point.

lemma *shd-sdrop-prob*: $P\ \{s.\ shd\ (sdrop\ n\ s){=}b\} = 1/2$
  sorry

It should be mentioned that there is research in progress that aims to truly import proofs from Cambridge HOL into Isabelle at Tobias Nipkow's theorem proving group in Munich. For the time being, converting the definitions and assuming the facts by hand should suffice to base a mere example on.

end

4.2 A new primitive

theory *Lsdest* = *Imported*:

It is time to introduce the example application that is being formalized in section 4.3. We will be looking at the most simplistic possible program for finding a satisfying assignment for a propositional formula in conjunctive normal form where any clause consists of exactly k literals. This problem is known as k-SAT. Our algorithm simply selects a random assignment for all of the n variables. We are interested in the probability that the assignment fails to satisfy a given clause. The reasons behind this will become clear in a while.

In the previous section it was stated that one should be able to construct any randomized functional program from the three primitive building blocks defined there. Of course, this also holds for the program we have in mind. Nevertheless, when you try following the style these constructs suggest, taking one random bit at a time and evaluating somewhere in between, you run into problems. That is to say, the clauses are not independent in general. A variable may appear in several clauses, and it would be wrong to fetch a new bit from the stream every time it is evaluated. Ergo, the simplest way to perform the evaluation of a clause independently from the rest is to get a list of all n random bits beforehand.

A function accomplishing this is not hard to devise. It is elementary enough to possibly support a lot of programs.

consts
  lsdest:: *nat* ⇒ *'a seq* ⇒ (*'a list* ∗ *'a seq*)

primrec
  lsdest 0 = *Pair* []
  lsdest (*Suc n*) = *bind sdest* (λ*x*. *bind* (*lsdest n*) (λ*l*. *Pair* (*x*#*l*)))

lemma *lsdest-length*: ⋀*s*. *length* (*fst* (*lsdest n s*)) = *n*
 by (*induct n*) (*auto simp add*: *bind-def split-def*)

lemma *lsdest-indep*: *lsdest n* ∈ *indep-fn*
  by (*induct n*) (*auto simp add*: *Pair-indep sdest-indep bind-indep*)

lemma *lsdesttakedrop*: ⋀*s*. *lsdest n s* = (*stake n s*, *sdrop n s*)
  by (*induct n*) (*simp-all add*: *bind-def sdest-def*)

We need to know more than these facts to make use of the new primitive. The real challenge here lies in getting to the probability distribution. Mind that it is not sufficient to know the probability of a whole list or a single bit. To draw conclusions about clauses, we need assertions about any k positions in a list of length n. On the other hand, this furnishes the whole common probability distribution. A few lemmata pave the way.

lemma *sdrop-stl*: ⋀*s*. *sdrop n* (*stl s*) = *stl* (*sdrop n s*)

lemma *stake-shd*:
  ⋀*s*. *stake u s* @ [*shd* (*sdrop u s*)] = *shd s* # *stake u* (*stl s*)
  by (*induct u*) *simp-all*

lemma *stake-prob*: ⋀*r m*. *Suc r* ≤ *n* ⟹ *P* {*s*. *stake n* (*sdrop m s*)!*r* = *b*} = *1/2*
proof (*induct n*)
  case *0* **thus** *?case* **by** *simp*
next
  case (*Suc n r m*)
  thus *?case*
  proof (*cases r=0*)
    case *True*

```
    hence ⋀s. stake (Suc n) (sdrop m s)!r = shd (sdrop m s)
      by simp
    hence {s. stake (Suc n) (sdrop m s)!r = b} = {s. shd (sdrop m s) = b}
      by fast
    thus ?thesis
      by (simp only: shd-sdrop-prob)
  next
    case False
    then obtain u where u:r = Suc u
      by (auto simp add: gr0-conv-Suc)
    hence ⋀s. stake (Suc n) (sdrop m s)!r = stake n (stl (sdrop m s))!u
      by simp
    also
    { fix s
      have stake n (stl (sdrop m s))!u = stake n (sdrop (Suc m) s)!u
        by (simp only: sdrop-stl[THEN sym] sdrop.simps[THEN sym])
    }
    finally
    have {s. stake (Suc n) (sdrop m s) ! r = b} =
      {s. stake n (sdrop (Suc m) s) ! u = b}
      by fast
    also
    from u Suc have Suc u ≤ n by simp
    with Suc have P {s. stake n (sdrop (Suc m) s) ! u = b} = 1 / 2
      by fast
    finally show ?thesis .
  qed
qed
```

This generalizes the simple probability fact from the last section and almost immediately results in the probability for a single bit in the *lsdest* list.

```
theorem lsdest-prob:
  assumes le: Suc r ≤ n
  shows P {s. (fst (lsdest n s))!r = b} = 1/2
```

Another generalization, this time of the recursive *lsdest* definition, enables the decisive theorem.

```
lemma lsdest-split:
  assumes le: r ≤ n
  shows lsdest n = bind (lsdest r) (λ(l::'a list).
  bind (lsdest (n−r)) (λk. Pair (l@k)))
```

The upcoming powerful result closes the theory. Notice that the proposition had to be strengthened to arrive at a viable induction hypothesis. That is why a simplified version is attached.

```
lemma lsdest-prob2:
  shows ⋀S n m. ⟦finite S; card S = k; ∀ r∈S. Suc (r−m) ≤ n ∧ m ≤ r⟧
  ⟹ P (⋂ r∈S. {s. (fst (lsdest n s))!(r−m) = b r}) = (1/2) ^k
proof (induct k)
  case (0 S)
  hence S={}
    by simp
  thus ?case
    by (simp add: UNIV-prob)

next
  case (Suc k S n m)
  then obtain r' where r' ∈ S
    by force
  hence (LEAST r. r ∈ S) ∈ S
    by (rule LeastI)
  also
  have ⋀y. y∈S ⟹ (LEAST r. r ∈ S) ≤ y
    by (rule Least-le)
  moreover
  def r ≡ (LEAST r. r ∈ S)
  ultimately have rS:r ∈ S and ∀ y∈S. r ≤ y
    by simp-all
  hence rl:∀ y∈S−{r}. r < y
    by auto

  from rS have S = insert r (S−{r})
    by auto
  with rl have Sr2:S = insert r {r2. r2 ∈ S ∧ r < r2}
    by auto
  hence (⋂ r∈S. {s. (fst (lsdest n s))!(r−m) = b r}) =
    (⋂ r∈ insert r {r2. r2 ∈ S ∧ r < r2}.
    {s. (fst (lsdest n s))!(r−m) = b r})
    by simp
  also have ... = {s. (fst (lsdest n s))!(r−m) = b r} ∩
    (⋂ r2 ∈ {r2. r2 ∈ S ∧ r < r2}. {s. (fst (lsdest n s))!(r2−m) = b r2})
    by simp
  finally
  have P (⋂ r∈S. {s. fst (lsdest n s) ! (r−m) = b r}) =
    P ({s. fst (lsdest n s) ! (r−m) = b r} ∩
    (⋂ r2∈{r2. r2 ∈ S ∧ r < r2}. {s. fst (lsdest n s) ! (r2−m) = b r2}))
    by simp

  also
  from prems rS have le: Suc (r−m) ≤ n
    by simp
  { fix s
```

```
  from le have fst (lsdest n s)!(r−m) =
    fst (bind (lsdest (Suc (r−m)))
    (λl. bind (lsdest (n−(Suc (r−m)))) (λk. Pair (l@k))) s)!(r−m)
    by (simp add: lsdest-split)
  hence fst (lsdest n s)!(r−m) = fst (lsdest (Suc (r−m)) s)!(r−m)
    by (simp add: nth-append bind-def split-def lsdest-length del: lsdest.simps)
}
hence {s. fst (lsdest n s)!(r−m) = b r} =
  {s. fst (lsdest (Suc (r−m)) s)!(r−m) = b r}
  by fast

moreover
from prems rS have mr:m ≤ r
  by simp

{ fix s r2 assume r < r2
  with mr have l2: ¬ (r2−m) < Suc (r−m) by arith
  from le have fst (lsdest n s) ! (r2−m) =
    fst (bind (lsdest (Suc (r−m)))
    (λl. bind (lsdest (n−(Suc (r−m)))) (λk. Pair (l@k))) s)!(r2−m)
    by (simp add: lsdest-split)
  with l2 have fst (lsdest n s) ! (r2−m) =
    fst (lsdest (n − Suc (r−m))
    (snd (lsdest (Suc (r−m)) s))) ! ((r2 − m) − Suc (r−m))
    by (simp add: nth-append bind-def split-def lsdest-length del: lsdest.simps)
}
hence (⋂ r2∈{r2. r2 ∈ S ∧ r < r2}. {s. fst (lsdest n s) ! (r2−m) = b r2}) =
  (⋂ r2∈{r2. r2 ∈ S ∧ r < r2}. {s. fst (lsdest (n − Suc (r−m))
  (snd (lsdest (Suc (r−m)) s))) ! ((r2−m) − Suc (r−m)) = b r2})
  by fast

ultimately have P (⋂ r∈S. {s. fst (lsdest n s) ! (r−m) = b r}) =
  P ({s. fst (lsdest (Suc (r−m)) s)!(r−m) = b r} ∩
  (⋂ r2∈{r2. r2 ∈ S ∧ r < r2}. {s. fst (lsdest (n − Suc (r−m))
  (snd (lsdest (Suc (r−m)) s))) ! ((r2−m) − Suc (r−m)) = b r2}))
  by simp

also
have ... = P ((fst ∘ lsdest (Suc (r−m))) −‘ {l. l!(r−m) = b r}
  ∩ (snd ∘ lsdest (Suc (r−m))) −‘ (⋂ r2∈{r2. r2 ∈ S ∧ r < r2}.
  {s. fst (lsdest (n − Suc (r−m)) s) ! ((r2−m) − Suc (r−m)) = b r2}))
  by (simp add: vimage-INT)
also
{
  { fix r2
    from lsdest-indep[of (n − Suc (r−m))]
    have (fst ∘ lsdest (n − Suc (r−m))) −‘
      {l. l! ((r2−m) − Suc (r−m)) = b r2} ∈ Epsilon
      by (auto simp only: indep-fn-def measurable-def)
```

hence $\{s.\ fst\ (lsdest\ (n - Suc\ (r{-}m))\ s)\ !$
$((r2{-}m) - Suc\ (r{-}m)) = b\ r2\} \in sigma\ Epsilon$
by (*simp add*: *sigma.intros*)
}
hence $(\bigcap r2 \in \{r2.\ r2 \in S \wedge r < r2\}.$
$\{s.\ fst\ (lsdest\ (n - Suc\ (r{-}m))\ s)\ !$
$((r2{-}m) - Suc\ (r{-}m)) = b\ r2\}) \in sigma\ Epsilon$
by (*simp add*: *sigma-INTER*)
with *ms-bern* **have** $(\bigcap r2 \in \{r2.\ r2 \in S \wedge r < r2\}.$
$\{s.\ fst\ (lsdest\ (n - Suc\ (r{-}m))\ s)\ !\ ((r2{-}m) - Suc\ (r{-}m)) = b\ r2\})$
$\in Epsilon$
by (*simp add*: *measurable-sets-def measure-space-def sigma-sigma-algebra*)
}
with *lsdest-indep* [*of Suc* $(r{-}m)$]
have $\ldots = P\ ((fst \circ lsdest\ (Suc\ (r{-}m)))\ -`\ \{l.\ l!(r{-}m) = b\ r\}) *$
$P\ (\bigcap r2{\in}\{r2.\ r2 \in S \wedge r < r2\}.$
$\{s.\ fst\ (lsdest\ (n - Suc\ (r{-}m))\ s)\ !\ ((r2{-}m) - Suc\ (r{-}m)) = b\ r2\})$
by (*rule indep-fn-prob*)

also
{ from *le-refl*[*of Suc* $(r{-}m)$] **have**
$P\ (\{s.\ fst\ (lsdest\ (Suc\ (r{-}m))\ s)!(r{-}m) = b\ r\}) = 1/2$
by (*rule lsdest-prob*) **}**
hence $\ldots = 1/2\ *$
$P\ (\bigcap r2{\in}\{r2.\ r2 \in S \wedge r < r2\}.$
$\{s.\ fst\ (lsdest\ (n - Suc\ (r{-}m))\ s)\ !\ ((r2{-}m) - Suc\ (r{-}m)) = b\ r2\})$
by *simp*

also
{ from *mr* **have** $\bigwedge r2.\ r2{\in}\{r2.\ r2 \in S \wedge r < r2\} \Longrightarrow$
$(r2{-}m) - Suc\ (r{-}m) = r2 - Suc\ r$
by *arith*
hence $(\bigcap r2{\in}\{r2.\ r2 \in S \wedge r < r2\}.$
$\{s.\ fst\ (lsdest\ (n - Suc\ (r{-}m))\ s)\ !\ ((r2{-}m) - Suc\ (r{-}m)) = b\ r2\}) =$
$(\bigcap r2{\in}\{r2.\ r2 \in S \wedge r < r2\}.$
$\{s.\ fst\ (lsdest\ (n - Suc\ (r{-}m))\ s)\ !\ ((r2 - Suc\ r)) = b\ r2\})$
by *auto*
}
hence $\ldots = 1/2 * P\ (\bigcap r2{\in}\{r2.\ r2 \in S \wedge r < r2\}.$
$\{s.\ fst\ (lsdest\ (n - Suc\ (r{-}m))\ s)\ !\ ((r2 - Suc\ r)) = b\ r2\})$
by *simp*

also
from *prems Sr2* **have** *finite* $(insert\ r\ \{r2.\ r2 \in S \wedge r < r2\})$
by *simp*
hence *fin2*:*finite* $\{r2.\ r2 \in S \wedge r < r2\}$
by (*simp only*: *finite-insert*)
with *prems Sr2* **have** *?this* **and** *card* $(insert\ r\ \{r2.\ r2 \in S \wedge r < r2\}) = Suc\ k$
by *simp-all*

```
  hence c2:card {r2. r2 ∈ S ∧ r < r2} = k
    by (simp add: card-insert)

  { fix d assume d ∈ {r2. r2 ∈ S ∧ r < r2}
    hence d ∈ S and rd:r < d
      by simp-all
    with prems have i:Suc (d − m) ≤ n and ii:m ≤ d
      by simp-all
    from i rd mr ii have Suc (d− Suc r) ≤ (n−Suc (r−m)) ∧ Suc r ≤ d
      by arith
  }
  hence ∀ d∈{r2. r2 ∈ S ∧ r < r2}.
    Suc (d−(Suc r)) ≤ (n−Suc (r−m)) ∧ Suc r ≤ d
    by simp

  with fin2 c2 prems have nn:0<k ⟹ P (⋂ r2∈{r2. r2 ∈ S ∧ r < r2}.
    {s. fst (lsdest (n − Suc (r−m)) s) ! (r2 − Suc r) = b r2}) = (1/2) ^k
    by simp

  have P (⋂ r2∈{r2. r2 ∈ S ∧ r < r2}.
    {s. fst (lsdest (n − Suc (r−m)) s) ! (r2 − Suc r) = b r2}) = (1/2) ^k
  proof (cases k)
    case 0
    with fin2 c2 have {r2. r2 ∈ S ∧ r < r2} = {}
      by simp
    hence (⋂ r2∈{r2. r2 ∈ S ∧ r < r2}.
      {s. fst (lsdest (n − Suc (r−m)) s) ! (r2 − Suc r) = b r2}) = UNIV
      by auto
    hence P (⋂ r2∈{r2. r2 ∈ S ∧ r < r2}.
      {s. fst (lsdest (n − Suc (r−m)) s) ! (r2 − Suc r) = b r2}) = 1
      by (simp add: UNIV-prob)
    with 0 show ?thesis by simp
  next
    case (Suc g)
    hence 0<k by simp
    with nn show ?thesis by simp
  qed

  finally show ?case by simp
qed

lemma lsdest-probs:
  shows ⟦finite S; card S = k; ∀ r∈S. Suc r ≤ n⟧
  ⟹ P (⋂ r∈S. {s. (fst (lsdest n s))!r = b r}) = (1/2) ^k

end
```

4.3 The first moment method

theory *kSAT* = *Integral*+*Lsdest*:

As you can see from the *theory* line, we finally get to use the integral along with the new primitive just developed. Surprisingly, the end result will be a combinatorial statement about the existence of an assignment for a k-SAT instance. Notwithstanding, probability and expectation properties of some randomized programs are going to lead us there. The technique employed — known as the first moment method — is a standard one in the field of randomized algorithms; it may be found in the authoritative textbook on the subject by Motwani and Raghavan [14].

Let the problem be defined first.

consts
$absdistinct$:: $int\ list \Rightarrow bool$
$clauseeval$:: $int\ list \Rightarrow bool\ list \Rightarrow bool$
$CNFeval$:: $(int\ list)\ list \Rightarrow bool\ list \Rightarrow bool$

primrec
$absdistinct\ [] = True$
$absdistinct\ (x\#xs) = (x \notin set\ xs \wedge -x \notin set\ xs \wedge absdistinct\ xs)$

constdefs
$nvarkclauses$:: $nat \Rightarrow nat \Rightarrow int\ list\ set$ (- *var* - *clauses*)
$n\ var\ k\ clauses \equiv \{ls.\ length\ ls = k\ \wedge$
$(\forall x \in set\ ls.\ x{\neq}0 \wedge |x| \leq int\ n) \wedge absdistinct\ ls\}$

$nvarksat$:: $nat \Rightarrow nat \Rightarrow ((int\ list)\ list)\ set$ (- *var* - *SAT*)
$n\ var\ k\ SAT \equiv \{A.\ \forall ls \in set\ A.\ ls \in n\ var\ k\ clauses\}$

lemma assumes $a \in set\ C$ **and** $b \in set\ C$
and $absdistinct\ C$ **and** $-a{\neq}a$
shows *clause-dist*: $-a \neq b$ **using** *prems*
by (*induct C*) *auto*

lemma *clauses-card*:
assumes *C*: *absdistinct C*
shows $card\ (set\ C) = length\ C$ **using** *C*
by (*induct C*) *auto*

Thus formulas are modeled as lists of clauses, which in turn are represented by lists of integers. The absolute value of a number stands for the variable name, the sign signifying negation of the literal. For an illustrative instance,-4 means the 4th variable inverted, and 0 is not allowed. A variable may not appear twice in any clause, as ensured by the *absdistinct* predicate. Looking at the following example might clarify the notation.

theorem $[[1,2,3],[-4,-2,1]] \in 4\ var\ 3\ SAT$
by (*simp add*: *nvarksat-def nvarkclauses-def*)

Formulas can be evaluated at an assignment, that is a list of booleans, by primitive recursive functions.

primrec
clauseeval [] *l* = *False*
clauseeval (*x*#*xs*) *l* = (*if* (*0*<*x*) *then* (*l*!*nat* (*x*+−*1*) ∨ *clauseeval xs l*)
else if (*x*<*0*) *then* (¬(*l*!*nat* (−*1*+−*x*)) ∨ *clauseeval xs l*)
else True)

primrec
CNFeval [] *l* = *True*
CNFeval (*x*#*xs*) *l* = (*clauseeval x l* ∧ *CNFeval xs l*)

lemma *CNF-clause*: *CNFeval F l* = (∀ *C*∈(*set F*). *clauseeval C l*)
by (*induct F*) *auto*

Now we may randomize these functions, obtaining just the simple programs described in 4.2. In addition, an indicator variable is defined that takes the value 1 for exactly those elementary events — or rather bit sequences — where the argument clause is not satisfied.

constdefs
randCNFeval:: (*int list*) *list* ⇒ *nat* ⇒ *bool seq* ⇒ (*bool* ∗ (*bool seq*))
randCNFeval F n s ≡ (*CNFeval F* (*stake n s*), *sdrop n s*)

randclauseeval:: *int list* ⇒ *nat* ⇒ *bool seq* ⇒ (*bool* ∗ (*bool seq*))
randclauseeval C n ≡ *bind* (*lsdest n*) (λ*l*. *Pair* (*clauseeval C l*))

indicator:: *int list* ⇒ *nat* ⇒ *bool seq* ⇒ *real*
indicator C n ≡ χ{*s*. ¬ *fst* (*randclauseeval C n s*)}

lemma *randCNFeval-bind-Pair*: *randCNFeval F n s* =
bind (*lsdest n*) (λ*l*. *Pair* (*CNFeval F l*)) *s*
by (*simp add*: *randCNFeval-def bind-def lsdesttakedrop*)

lemma *rand-CNF-clause*: *fst* (*randCNFeval F n s*) =
(∀ *C*∈*set F*. *fst* (*randclauseeval C n s*))
by (*simp add*:
CNF-clause randCNFeval-bind-Pair randclauseeval-def bind-def split-def)

We just saw that both *randclauseeval* and *randCNFeval* can be built from *Pair*, *bind* and *sdest* alone. Hence they are strongly independent functions.

In particular, indicator is a characteristic function for an event.

lemma *rce-indep*: *randclauseeval C n* $\in$ *indep-fn*
by (*simp add*: *lsdest-indep bind-indep Pair-indep randclauseeval-def*)

lemma *rce-events*: $\{s.\ \neg$ *fst* (*randclauseeval C n s*)$\}$ $\in$ *Epsilon*
proof $-$
from *rce-indep* **have** (*fst* $\circ$ *randclauseeval C n*) $-$‘ $\{$*False*$\}$ $\in$ *Epsilon*
by (*simp add*: *rce-indep indep-fn-def measurable-def*)
thus *?thesis*
by (*simp add*: *vimage-def*)
qed

The next step is to compute the measure of this event, the probability that a given clause is not satisfied. In spite of the preparatory work on *lsdest*, the greatest difficulty lies in here. Though a rough idea should have emerged until now, it is technically demanding to arrive at a setup where *lsdest-probs* may be applied instantly. No general insight is gained from the proofs, so they are left out.

lemma *clauseeval-inj*:
assumes *cont*: $(\lambda x.\ \mathit{if}\ 0{<}x\ \mathit{then\ nat}\ (x{+}{-}1)\ \mathit{else\ nat}\ ({-}1{+}{-}x))\ p$
$= (\lambda x.\ \mathit{if}\ 0{<}x\ \mathit{then\ nat}\ (x{+}{-}1)\ \mathit{else\ nat}\ ({-}1{+}{-}x))\ r$
and *r0*: $r \neq 0$ **and** *p0*: $p \neq 0$
shows $p = r \vee -p = r$

lemma *not-clauseeval*:
$\bigwedge C.\ C \in n\ \mathit{var}\ k\ \mathit{clauses} \Longrightarrow$
$\exists\, b.\ \forall\, l.\ (\neg\ \mathit{clauseeval}\ C\ l) =$
$(l \in (\bigcap r{\in}((\lambda x.\ \mathit{if}\ 0{<}x\ \mathit{then\ nat}\ (x{+}{-}1)\ \mathit{else\ nat}\ ({-}1{+}{-}x))\text{‘}\mathit{set}\ C).$
$\{l.\ l!r = b\ r\}))$

theorem assumes *C*: $C \in n$ *var k clauses*
shows *rce-prob*: $P\ \{s.\ \neg$ *fst* (*randclauseeval C n s*)$\} = (1/2)\hat{\ }k$

We should take a moment to appreciate this first result. It embodies the gist of the probabilistic analysis for the *randclauseeval* randomized algorithm. What is more, it enables the primal application of integration in the following theorem.

theorem assumes $C \in n$ *var k clauses*
shows *ind-int*: $\int$ (*indicator C n*) ∂ (*Epsilon*, *P*) $= (1/2)\hat{\ }k$
and *integrable* (*indicator C n*) (*Epsilon*, *P*)
using *prems*
by (*auto simp add*: *ms-bern measurable-sets-def rce-events*
rce-prob integral-char indicator-def measure-def)

Here we encounter an expectation in the true sense for the first time in this thesis. Like any expectation it sums up easily.

lemma *nat-lessThan-card*:
  card {..(*k*::*nat*)(} = *k*
  by (*induct k*) (*auto simp add*: *lessThan-Suc*)

theorem *sum-ind-int*:
  assumes *sat*: $F \in n\ var\ k\ SAT$
  shows
  $\int\ (\lambda s.\ \sum m{\in}\{..(length\ F)(\}.\ indicator\ (F!m)\ n\ s)\ \partial(Epsilon,\ P)$
  $= real\ (length\ F)/2\hat{}k$
  and *integrable* $(\lambda s.\ \sum m{\in}\{..(length\ F)(\}.\ indicator\ (F!m)\ n\ s)(Epsilon,\ P)$

The result just obtained contributes all the information about probabilistic programs we will need: The expected number of unsatisfied clauses with our simplistic algorithm is the total number of clauses divided by 2^k. It is only now that the first moment method comes into play. The point put forward by this proposition is that if the expected value of nonnegative random variable is less than 1, then there must be an event witnessing this. The proof turns out to be rather elementary from the Markov inequation.

theorem *first-moment-method*:
  assumes *int1*: *integrable f M* **and** *nn*: *nonnegative f* **and** *int2*: $\int\ f\ \partial\ M < 1$
  shows *law M f* {*1*..} < *1*
proof −
  from *nn* **have** *eq*: $f = (\lambda t.\ |f\ t|\ \hat{}1)$
    by (*simp-all add*: *nonnegative-def abs-eqI1*)
  with *int1* **have** $0 < (1::real)$ **and** *integrable* $(\lambda t.\ |f\ t|\ \hat{}1)\ M$
    by *simp-all*
  with *int1* **have** *law M f* $\{1..\} \le \int\ (\lambda t.\ |f\ t|\ \hat{}1)\ \partial\ M\ /\ (1\hat{}1)$
    by (*rule markov-ineq*)
  also from *eq int2* **have** $\ldots < 1$
    by *simp*
  finally show *?thesis* **.**
qed

corollary *fmm*:
  assumes *int1*: *integrable f M* **and** *int2*: $\int\ f\ \partial\ M < 1$
  and *ps*: *prob-space M*
  shows $\exists\, s.\ f\ s < 1$
proof (*cases nonnegative f*)
  case *False*

  then obtain *s* **where** $\neg\ 0 \le f\ s$
    by (*auto simp add*: *nonnegative-def*)
  hence $f\ s < 1$ **by** *arith*

```
  thus ?thesis by fast
next
  case True
  from int1 True int2 have law M f {(1::real)..} < 1
    by (rule first-moment-method)
  with int1 ps have (f −‘ {(1::real)..}) ≠ UNIV
    by (auto simp add: integrable-rv prob-space-def distribution-def)
  then obtain s where ¬ 1 ≤ f s
    by auto
  hence f s < 1 by arith
  thus ?thesis
    by fast
qed
```

In the application we have in mind, a random bit-stream that makes the indicator variables sum to a value less than 1 corresponds to a satisfying assignment.

```
lemma assumes nk: F ∈ n var k SAT and
  sum: (∑ m∈{..(length F)(}. indicator (F!m) n env) < 1
  shows satisfy: CNFeval F (stake n env)
proof −
  have fin: finite {..(length F)(}
    by simp
  {
    fix i assume i ∈ {..(length F)(}
      and indicator (F!i) n env = 1
    with sum fin have (∑ m∈{..(length F)(}−{i}. indicator (F!m) n env) < 0
      by (simp add: setsum-diff-real)
    also have ∀ j∈{..(length F)(}−{i}. 0 ≤ indicator (F!j) n env
      by (simp add: indicator-def characteristic-function-def)
    hence 0 ≤ (∑ m∈{..(length F)(}−{i}. indicator (F!m) n env)
      by (rule setsum-ge0-real)

    finally have False by simp
  }
  hence ∀ i ∈ {..(length F)(}. fst (randclauseeval (F ! i) n env)
    by (auto simp add: indicator-def characteristic-function-def if-def)
  hence ∀ C ∈ set F. fst (randclauseeval C n env)
    by (auto simp add: set-conv-nth)
  hence fst (randCNFeval F n env)
    by (simp add: rand-CNF-clause)
  thus ?thesis
    by (simp add: randCNFeval-def)
qed
```

In the end we have shown that a satisfying assignment always exists if there are less than 2^k clauses in a k-CNF formula.

theorem assumes *nk*:$F \in n\ var\ k\ SAT$ **and** *l*:$real(length\ F) < 2\hat{}k$
  shows *existence*: $\exists\, l.\ CNFeval\ F\ l$
proof –
  from *nk* **have** *int*:
    $integrable\ (\lambda s.\ \sum m\in\{..(length\ F)(\}.\ indicator\ (F!m)\ n\ s)\ (Epsilon,\ P)$
    by (*rule sum-ind-int*)
  from *nk* **have**
    $\int\ (\lambda s.\ \sum m\in\{..(length\ F)(\}.\ indicator\ (F!m)\ n\ s)\ \partial(Epsilon,\ P)$
    $= real\ (length\ F)/2\hat{}k$
    by (*rule sum-ind-int*)
  also
  have $(0::real) < 2\hat{}k$
    by (*simp add*: *realpow-gt-zero*)
  with *l* **have** $(real\ (length\ F)/2\hat{}k < 1)$
    by (*simp add*: *pos-real-divide-less-eq*)
  finally have
    $\int\ (\lambda s.\ \sum m\in\{..(length\ F)(\}.\ indicator\ (F!m)\ n\ s)\ \partial(Epsilon,\ P) < 1$ **.**

  with *int ps-bern* **obtain** *env* **where**
    $(\sum m\in\{..(length\ F)(\}.\ indicator\ (F!m)\ n\ env) < 1$
    by (*force simp add*: *fmm*)
  with *nk* **have** $CNFeval\ F\ (stake\ n\ env)$
    by (*rule satisfy*)
  thus *?thesis* **..**
qed

The final result is a purely combinatorial fact, serving as an example for the application of probability and integration in an apparently unrelated field. There are stronger versions, requiring more intricate algorithms and tools[4]. Anyhow, this one should suffice to display the power of the basic integral properties.

end

[4]The Lovasz Local Lemma is a prominent example.

Chapter 5

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 in this thesis? 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, for the reasons given in section 1.1, 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.

In this respect, research is always incomplete. In spite of everything, it is a pity that some theorems in this chapter could not be finished in time, even if it was for days only. This will naturally be made up for very soon. The delay was caused in part by the abortive paths to integration described in section 3.1. The issue of finding the deeper reason for the latter failure is still unresolved.

Another 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.

To my mind, the example application conveys its point in a satisfactory manner. As a side effect, another building block for functional probabilistic programming, or what is more, its essential properties, could be obtained. Without a doubt, there is an infinite amount of further examples, including more involved varieties of the first moment method or the run-time analysis of probabilistic quicksort. They lend themselves to continued work demonstrating the power of interactive proof systems. Hopefully, some of this future work may benefit from the foundations laid in the present thesis.

Appendix A

Auxiliary Theories

A.1 Rational numbers

theory *Rats* = *Real*:

A dense and countable subset of the *real* type was needed for some measurability proofs. That is why I developed this theory.

To begin with, an injective function from $\mathbb{N}^2$ to $\mathbb{N}$ is defined[1]. Its inverse is then a surjective mapping into $\mathbb{N}^2$. Another iteration yields three natural numbers, one for enumerator, denominator, and sign respectively. The rationals are now exactly the range of the resulting function on $\mathbb{N}$, which already proves them countable, without even defining this concept.

Much to my delight, these functions could be reused for the simple function integral properties.

constdefs

$n2\text{-}to\text{-}n :: (nat * nat) \Rightarrow nat$
$n2\text{-}to\text{-}n\ pair \equiv let\ (n,m) = pair\ in\ (n+m) * Suc\ (n+m)\ div\ 2 + n$

$n\text{-}to\text{-}n2 ::\ nat \Rightarrow (nat * nat)$
$n\text{-}to\text{-}n2 \equiv inv\ n2\text{-}to\text{-}n$

$n3\text{-}to\text{-}rat :: nat \Rightarrow nat \Rightarrow nat \Rightarrow real$
$n3\text{-}to\text{-}rat\ a\ b\ c \equiv if\ 2\ dvd\ a\ then\ real\ b\ /\ real\ c\ else\ -\ real\ b\ /real\ c$

$n\text{-}to\text{-}rat :: nat \Rightarrow real$
$n\text{-}to\text{-}rat\ n \equiv let\ (a,x) = n\text{-}to\text{-}n2\ n\ ;\ (b,c) = n\text{-}to\text{-}n2\ x\ in$
$n3\text{-}to\text{-}rat\ a\ b\ c$

[1] The function as well as the proofs are derived from [18] p. 85.

  Rats:: *real set* ($\mathbb{Q}$)
  $\mathbb{Q}$ ≡ *range n-to-rat*

lemma *dvd2-a-x-suc-a*: *2 dvd a ∗* (*Suc a*)

lemma assumes *eq*: *n2-to-n* (*u*,*v*) = *n2-to-n* (*x*,*y*)
  shows *n2-to-n-help*: *u*+*v* ≤ *x*+*y*
proof (*rule classical*)
  assume ¬ *?thesis*
  hence *contrapos*: *x*+*y* < *u*+*v*
    by *simp*
  have *n2-to-n* (*x*,*y*) < (*x*+*y*) ∗ *Suc* (*x*+*y*) *div 2* + *Suc* (*x* + *y*)
    by (*unfold n2-to-n-def*) (*simp add*: *Let-def*)
  also have . . . = (*x*+*y*)∗*Suc*(*x*+*y*) *div 2* + *2* ∗ *Suc*(*x*+*y*) *div 2*
    by (*simp only*: *div-mult-self1-is-m*)
  also have . . . = (*x*+*y*)∗*Suc*(*x*+*y*) *div 2* + *2* ∗ *Suc*(*x*+*y*) *div 2*
    + ((*x*+*y*)∗*Suc*(*x*+*y*) *mod 2* + *2* ∗ *Suc*(*x*+*y*) *mod 2*) *div 2*
  proof −
    have *2 dvd* (*x*+*y*)∗*Suc*(*x*+*y*)
      by (*rule dvd2-a-x-suc-a*)
    hence (*x*+*y*)∗*Suc*(*x*+*y*) *mod 2* = *0*
      by (*simp only*: *dvd-eq-mod-eq-0*)
    also
    have *2* ∗ *Suc*(*x*+*y*) *mod 2* = *0*
      by (*rule mod-mult-self1-is-0*)
    ultimately have
      ((*x*+*y*)∗*Suc*(*x*+*y*) *mod 2* + *2* ∗ *Suc*(*x*+*y*) *mod 2*) *div 2* = *0*
      by *simp*
    thus *?thesis*
      by *simp*
  qed
  also have . . . = ((*x*+*y*)∗*Suc*(*x*+*y*) + *2*∗*Suc*(*x*+*y*)) *div 2*
    by (*rule div-add1-eq*[*THEN sym*])
  also have . . . = ((*x*+*y*+*2*)∗*Suc*(*x*+*y*)) *div 2*
    by (*simp only*: *add-mult-distrib*[*THEN sym*])
  also from *contrapos* **have** . . . ≤ ((*Suc*(*u*+*v*))∗(*u*+*v*)) *div 2*
    by (*simp only*: *mult-le-mono div-le-mono*)
  also have . . . ≤ *n2-to-n* (*u*,*v*)
    by (*unfold n2-to-n-def*) (*simp add*: *Let-def*)
  finally show *?thesis*
    by (*simp only*: *eq*)
qed

lemma *n2-to-n-inj*: *inj n2-to-n*
proof −
  {**fix** *u v x y* **assume** *n2-to-n* (*u*,*v*) = *n2-to-n* (*x*,*y*)
    hence *u*+*v* ≤ *x*+*y* **by** (*rule n2-to-n-help*)
    also from *prems*[*THEN sym*] **have** *x*+*y* ≤ *u*+*v*
      by (*rule n2-to-n-help*)
    finally have *eq*: *u*+*v* = *x*+*y* **.**
    with *prems* **have** *ux*: *u*=*x*
      by (*simp add*: *n2-to-n-def Let-def*)
    with *eq* **have** *vy*: *v*=*y*
      by *simp*
    with *ux* **have** (*u*,*v*) = (*x*,*y*)
      by *simp*
  }
  hence $\bigwedge$*x y*. *n2-to-n x* = *n2-to-n y* $\Longrightarrow$ *x*=*y*
    by *auto*
  thus *?thesis*
    by (*unfold inj-on-def*) *simp*
qed

lemma *n-to-n2-surj*: *surj n-to-n2*
  by (*simp only*: *n-to-n2-def n2-to-n-inj inj-imp-surj-inv*)

theorem *nat-nat-rats*: *real* (*a*::*nat*)/*real* (*b*::*nat*) $\in \mathbb{Q}$
proof −
  from *n-to-n2-surj* **obtain** *x* **where** (*a*,*b*) = *n-to-n2 x*
    by (*auto simp only*: *surj-def*)
  also from *n-to-n2-surj* **obtain** *n* **where** (*0*,*x*) = *n-to-n2 n*
    by (*auto simp only*: *surj-def*)
  moreover have *n3-to-rat 0 a b* = *real a*/*real b*
    by (*simp add*: *n3-to-rat-def*)
  ultimately have *real a*/*real b* = *n-to-rat n*
    by (*auto simp add*: *n-to-rat-def Let-def split*: *split-split*)
  hence *real a*/*real b* ∈ *range n-to-rat*
    by (*auto simp add*: *image-def*)
  thus *?thesis*
    by (*simp add*: *Rats-def*)
qed

theorem *minus-nat-nat-rats*: $-$ *real* (*a*::*nat*)/*real* (*b*::*nat*) $\in \mathbb{Q}$
proof $-$
  from *n-to-n2-surj* **obtain** *x* **where** (*a*,*b*) = *n-to-n2 x*
    by (*auto simp only*: *surj-def*)
  also from *n-to-n2-surj* **obtain** *n* **where** (*1*,*x*) = *n-to-n2 n*
    by (*auto simp only*: *surj-def*)
  moreover have *n3-to-rat 1 a b* = $-$ *real a*/*real b*
    by (*simp add*: *n3-to-rat-def*)
  ultimately have $-$ *real a*/*real b* = *n-to-rat n*
    by (*auto simp add*: *n-to-rat-def Let-def split*: *split-split*)
  hence $-$ *real a*/*real b* $\in$ *range n-to-rat*
    by (*auto simp add*: *image-def*)
  thus *?thesis*
    by (*simp add*: *Rats-def*)
qed

The following lemmata do not seem to exist in the *RealAbs* theory, but I think they should. The proof is of unexpected complexity, since there are a number of theorems on *abs*, conversion from *int* to *real*, etc. missing.

lemma *real-of-int-abs*: |*real* (*x*::*int*)| = *real* |*x*|

lemma *real-abs-div*: |(*a*::*real*)/*b*| = |*a*|/|*b*|

lemma *not-neg-abs*: $\neg$ *neg* |*a*|

theorem *int-int-rats*: *real* (*a*::*int*)/*real* (*b*::*int*) $\in \mathbb{Q}$
proof (*cases real a*/*real b* < *0*)
  case *False*
  hence (*real a*/*real b*) = |*real a*/*real b*|
    by *arith*
  also have ... = *real* |*a*|/*real* |*b*|
    by (*simp only*: *real-abs-div real-of-int-abs*)
  also have ... = *real* (*nat* |*a*|)/*real* (*nat* |*b*|)
    by (*simp add*: *not-neg-abs real-of-nat-real-of-int*)
  finally show *?thesis*
    by (*simp only*: *nat-nat-rats*)
next
  case *True*
  hence (*real a*/*real b*) = $-$|*real a*/*real b*|
    by *arith*
  also have ... = $-$ *real* (*nat* |*a*|)/*real* (*nat* |*b*|)
    by (*simp add*:
      real-abs-div real-of-int-abs not-neg-abs real-of-nat-real-of-int)
  finally show *?thesis*

    by (*simp only*: *minus-nat-nat-rats*)
qed

theorem assumes *a*: $z \in \mathbb{Q}$
shows *rats-int-int*: $\exists\ x\ y.\ z = real\ (x::int)/real\ (y::int)$
theorem assumes *a*: $z \in \mathbb{Q}$
shows *rats-int-intnot0*: $\exists\ x\ y.\ z = real\ (x::int)/real\ (y::int) \wedge y{\neq}0$

theorem assumes *a*: $a \in \mathbb{Q}$ **and** *b*: $b \in \mathbb{Q}$
  shows *rats-plus-rats*: $a{+}b \in \mathbb{Q}$
proof −
  from *a* **obtain** *x y* **where** $a = real\ (x::int)/real\ (y::int) \wedge y{\neq}0$
    by (*force simp add*: *rats-int-intnot0*)
  also from *b* **obtain** *xb yb* **where** $b = real\ (xb::int)/real\ (yb::int) \wedge yb{\neq}0$
    by (*force simp add*: *rats-int-intnot0*)

  ultimately have *yn0*: $y{\neq}0$ **and** *ybn0*: $yb{\neq}0$
    and *eq*: $a{+}b = real\ x/real\ y + real\ xb/real\ yb$
    by *auto*

  note *eq* **also from** *yn0 ybn0*
  have $\ldots = real\ yb * real\ x\ /\ (real\ yb * real\ y) +$
    $real\ y * real\ xb\ /\ (real\ yb * real\ y)$ (**is** - $= ?X/?Z + ?Y/?Z$)
    by (*simp add*: *real-mult-div-cancel1*[*THEN sym*] *real-mult-commute*)
  also have $\ldots = (?X + ?Y)/?Z$
    by (*rule real-add-divide-distrib*[*THEN sym*])
  also have $\ldots = real\ (yb{*}x + y{*}xb)\ /\ real\ (yb{*}y)$
    by (*simp only*: *real-of-int-mult real-of-int-add*)

  finally show *?thesis* **by** (*simp only*: *int-int-rats*)
qed

The density proof was first to be adapted from a Mizar document [12]. Alas, it depends on a Gauss bracket (or floor function) that could not be found anywhere in Isabelle/HOL; and it turned out many lemmata are missing about the relation between integers and reals. Fortunately, a much more elementary proof was discovered in “Real Analysis” by H.L. Royden ([22] p. 32 ff). It directly employs the axiom of Archimedes, which is already in the *RComplete* theory.

lemma assumes *nn*: $0 \leq x$ **and** *ord*: $x<y$
  shows *rats-dense-in-nn-real*: $\exists\, r \in \mathbb{Q}.\ \ x<r \wedge r<y$
proof $-$
  from *ord* **have** $0 < y-x$ **..**
  with *reals-Archimedean* **obtain** $q{::}nat$
    where *q*: $inverse\ (real\ q) < y-x$ **and** *qpos*: $0 < real\ q$
    by *auto*

  def $p \equiv LEAST\ n.\ \ y \leq real\ (Suc\ n)/real\ q$

  from *reals-Archimedean2* **obtain** $n{::}nat$ **where** $y * real\ q < real\ n$
    by *auto*
  with *qpos* **have** *ex*: $y \leq real\ n/real\ q$ (**is** $?P\ n$)
    by (*simp add*: *pos-real-less-divide-eq*[*THEN sym*])
  also from *nn ord* **have** $\neg\ y \leq real\ (0{::}nat)\ /\ real\ q$
    by *simp*
  ultimately have *main*: $(LEAST\ n.\ y \leq real\ n/real\ q) = Suc\ p$
    by (*unfold p-def*) (*rule Least-Suc*)
  also from *ex* **have** $?P\ (LEAST\ x.\ ?P\ x)$
    by (*rule LeastI*)
  ultimately have *suc*: $y \leq real\ (Suc\ p)\ /\ real\ q$
    by *simp*

  def $r \equiv real\ p/real\ q$

  have $x=y-(y-x)$
    by *simp*
  also from *suc q* **have** $\ldots < real\ (Suc\ p)/real\ q - inverse\ (real\ q)$
    by *arith*
  also have $\ldots = real\ p\ /\ real\ q$
    by (*simp only*: *real-inverse-eq-divide real-diff-def real-of-nat-Suc*
      real-minus-divide-eq[*THEN sym*] *real-add-divide-distrib*[*THEN sym*]) *simp*
  finally have *1*: $x<r$
    by (*unfold r-def*)

  have $p<Suc\ p$ **.. also note** *main*[*THEN sym*]
  finally have $\neg\ ?P\ p$
    by (*rule not-less-Least*)
  hence *2*: $r<y$
    by (*simp add*: *r-def*)

  from *r-def* **have** $r \in \mathbb{Q}$
    by (*simp only*: *nat-nat-rats*)

  with *1 2* **show** *?thesis* **by** *fast*
qed

theorem assumes *ord*: $x<y$
shows *rats-dense-in-real*: $\exists\, r \in \mathbb{Q}.\ \ x<r \wedge r<y$
proof $-$
from *reals-Archimedean2* **obtain** n::*nat* **where** $-x <$ *real* n
by *auto*
hence $0 \leq x +$ *real* n
by *arith*
also from *ord* **have** $x +$ *real* $n < y +$ *real* n
by *arith*
ultimately have $\exists\, r \in \mathbb{Q}.\ x +$ *real* $n < r \wedge r < y +$ *real* n
by (*rule rats-dense-in-nn-real*)
then obtain r **where** *r1*: $r \in \mathbb{Q}$ **and** *r2*: $x +$ *real* $n < r$
and *r3*: $r < y +$ *real* n
by *blast*
have $r -$ *real* $n = r +$ *real* (*int* n)/*real* (-1::*int*)
by (*simp add*: *real-of-int-real-of-nat*)
also from *r1* **have** $r +$ *real* (*int* n)/*real* (-1::*int*) $\in \mathbb{Q}$
by (*simp only*: *int-int-rats rats-plus-rats*)
also from *r2* **have** $x < r -$ *real* n
by *arith*
moreover from *r3* **have** $r -$ *real* $n < y$
by *arith*

ultimately show *?thesis*
by *fast*
qed

end

A.2 Finite sum properties

theory *SetsumThms* = *SEQ*:

This theory is but a collection of simple properties of the *setsum* operator. Most of them are unproven for lack of time, but all of them should be rather obvious.

lemma *setsum-diff-real*: $\mathit{finite}\ A \Longrightarrow (\mathit{setsum}\ f\ (A - \{a\}) :: \mathit{real}) =$
$(\mathit{if}\ a{:}A\ \mathit{then}\ \mathit{setsum}\ f\ A - f\ a\ \mathit{else}\ \mathit{setsum}\ f\ A)$
by (*erule finite-induct*) (*auto simp add*: *insert-Diff-if*)

lemma *setsum-times-real*: $(a{::}\mathit{real}) * (\sum i{\in}R.\ f\ i) = (\sum i{\in}R.\ a{*}f\ i)$
sorry

lemma assumes *fin*: *finite X*
shows *setsum-image*: $\mathit{setsum}\ f\ (s\ \text{‘}\ X) = \mathit{setsum}\ (f \circ s)\ X$
sorry

lemma *setsum-mono-real*:
assumes *le*: $\bigwedge i.\ i{\in}K \Longrightarrow f\ i \le (g\ i{::}\mathit{real})$
shows $(\sum i{\in}K.\ f\ i) \le (\sum i{\in}K.\ g\ i)$
sorry

lemma *limseq-setsum*:
assumes *n*: $\bigwedge n.\ n \in S \Longrightarrow X\ n\ {-}{-}{-}{-}{>}\ L\ n$
shows $(\lambda m.\ \sum n{\in}S.\ X\ n\ m)\ {-}{-}{-}{-}{>}\ (\sum n{\in}S.\ L\ n)$
sorry

lemma *setsum-const-real*:
$\mathit{finite}\ A \Longrightarrow (\sum k{\in}A.\ a) = \mathit{real}\ (\mathit{card}\ A){*}a$
sorry

lemma *setsum-ge0-real*:
assumes *f*: $\forall\, i \in S.\ 0 \le f\ i$
shows $(0{::}\mathit{real}) \le (\sum i{\in}S.\ f\ i)$
proof –
from *f* **have** $(\sum i{\in}S.\ 0) \le (\sum i{\in}S.\ f\ i)$
by (*simp add*: *setsum-mono-real*)
thus *?thesis*
by (*simp add*: *setsum-0*)
qed

end

Bibliography

[1] Anonymous. The QED manifesto. In Alan Bundy, editor, *12th International Conference on Automated Deduction (CADE-12)*, volume 814 of *Lecture Notes in Artificial Intelligence*, 1994. Available on the web as http://www.rbjones.com/rbjpub/logic/qedres00.htm.

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

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

[4] Alonzo Church. A simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[5] 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.

[6] 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.

[7] 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/.

[8] Michael J. C. Gordon and Thomas F. Melham. *Introduction to HOL: A theorem proving environment for higher order logic*. Cambridge University Press, 1993.

[9] John Harrison. Formalized mathematics. Technical Report 36, Turku Centre for Computer Science (TUCS), 1996. Available on the web as http://www.cl.cam.ac.uk/users/jrh/papers/form-math3.html.

[10] John Harrison. *Theorem Proving with the Real Numbers*. Springer, 1996. Available on the web as http://citeseer.nj.nec.com/harrison96theorem.html.

[11] 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.

[12] Andrzej Kondracki. Basic properties of rational numbers. *Journal of Formalized Mathematics*, 2, 1990. Available on the web as http://mizar.uwb.edu.pl/JFM/Vol2/rat_1.html.

[13] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1977.

[14] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1997.

[15] 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.

[16] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. Available on the web as http://isabelle.in.tum.de/dist/Isabelle2003/doc/tutorial.pdf.

[17] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. Isabelle's logics: HOL, 2002. Unpublished. Available on the web as http://isabelle.in.tum.de/doc/logics-HOL.pdf.

[18] Arnold Oberschelp. *Rekursionstheorie*. BI-Wissenschafts-Verlag, 1993.

[19] Lawrence C. Paulson. Isabelle: The next 700 theorem provers. In Piergiorgio Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990. Available on the web as http://www.cl.cam.ac.uk/Research/Reports/TR143-lcp-experience.pdf.

[20] Lawrence C. Paulson. Isabelle: A generic theorem prover. *Lecture Notes in Computer Science*, 828:xvii + 321, 1994.

[21] Lawrence C. Paulson. *ML for the Working Programmer*. Cambridge University Press, second edition, 1996.

[22] Halsey Lawrence Royden. *Real Analysis*. Macmillan, 1968.

[23] Eric Schechter. An introduction to the gauge integral, 2001. Unpublished. Available on the web as http://www.math.vanderbilt.edu/~schectex/ccc/gauge/.

[24] Andrzej Trybulec and Howard Blair. Computer assisted reasoning with MIZAR. In Joshi Aravind, editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 26–28, 1985.

[25] Markus Wenzel. *Isabelle/Isar — a versatile environment for human-readable formal proof documents.* PhD thesis, Technische Universität München, 2002. Available on the web as http://isabelle.in.tum.de/Isar/isar-thesis-Isabelle2002.pdf.

[26] 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.

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