HoTT Types

March 27, 2014

Currently this doc contains a (mildly organized) set of notes followed by the intro and chapter 1 from the HoTT Book. Eventually (maybe) the intro and chapter 1 will contain annotations, comments, additional examples, etc., but I have not started that yet, so if you are already familiar with the text you need not read them – I haven't (so far as I recall) changed anything.

The idea is to winnow out some of the strictly mathematical stuff leaving the core "philosophical" stuff, and annotate the text with some comments and quotes from Martin-Löf, Brandom, etc. Or maybe leave the math stuff in, but annotate it with more detailed explanation and examples in programming languages. In any case the purpose is to more fully articulate the link between HoTT's ideas of type and judgment (etc.) to the philosophical debates about language, assertion, proposition from which they emerged. Why? Because I find those bits of the HoTT a little murky, and philosophers like Brandom have a lot to say about the issues. Also, to show more clearly how type theory differs from set theory and classic logic. Another goal is to provide more practical guidance to programmers interested in exploring dependent types.

Contents

Logics

Traditional terms are primitive; propositions are combinations of terms; judgments apply to proposotions

Modern: classic LEM, AC, etc.

Modern: intuitionistic

Expressivism Brandom's version: propositions are primitive; relation to inferential semantics; Price's global expressivism

Proof: Truth and Consequences

3.1 Truth

Why Truth is Not Important in Type Theory (with apologies to R. Brandom[4])

Consequence as prior to truth — [5]

Proof before truth [6]

> **Ed. note 3.1** Propositions are either true or false in classic math and logic; in type theory they are either proven or disproven. In this respect type theory is just like contemporary pragmatism, which (generally speaking) treats truth as otiose; what matters is not truth but function or expressiveness.
>
> This suggests a test for learners: until you've grasped why truth is not imporant in type theory you haven't really grasped type theory.

Traditional (classic) view: a proof is an epistemic device; it displays, exhibits, makes *visible* (if only to the mind's eye) a form of *certain knowledge*.[7]

Alternatives to the spectator theory: pragmatism, know-how over know-that.

> **Ed. note 3.2** TODO: summary of concepts of proof. Emphasize contrast between representationalism and inferentialism. Representationalism is atomistic: you could have only one concept. Inferentialism is holistic: you have to start out with at least two concepts, since every inference involves a premise and a conclusion. Inferentialism is a natural fit for HoTT.
>
> Question: can you have only one type? In other words, is type theory essentially holistic or atomistic?

For HoTT, as for most varieties of constructivism, it is better to abandon traditional notions of proof as something you see in favor of a more pragmatic notion of proof as something you do.

etc.

Critical point: in HoTTwe have two "kinds" of types: propositional types and non-propositional types.[8] If we are to also treat "proof" (or witness or whatever) as a fundamental principle of HoTT, one

[4] Robert B Brandom. Why truth is not important in philosophy. In *Reason in philosophy*, pages 156–176. The Belknap Press of Harvard University Press, Cambridge, Mass., 2009

[5] Peter Schroeder-Heister. Proof-theoretic semantics, self-contradiction, and the format of deductive reasoning. *Topoi*, 31(1):77–85, April 2012b. DOI: 10.1007/s11245-012-9119-x

[6] Peter Schroeder-Heister. Validity concepts in proof-theoretic semantics. *Synthese*, 148(3):525–571, February 2006. DOI: 10.1007/s11229-004-6296-1

[7] The link between knowing and seeing runs very deep in Western culture. Not surprisingly it is closely connected with representationalism and cartesianism generally. It has pretty much dominated Western thinking since Descartes, but has come under strong attack from Pragmatists. Dewey called it "the spectator theory of knowledge."f See [Rorty, 2009] etc.

[8] This is not in general recognized in the HoTT Book , but I think it should be emphasized, if only because it reflects intuition.

that complements the concept of type, then we need to treat both "type" and "proof" as genuses (genii?) of which propositional and non-propositional are species.

Ed. note 3.3 General point (to be made elsewhere, maybe in §6: the concepts of type and proof go together. You cannot have one without the other. That's very different than set theory. You can have sets and elements without proofs.

Long story short: we are in dire need of improved terminology. My suggestion is as follows:

Proof of a proposition In contrast to the classic spectator view, we treat proof not as the exhibition (or: making available for inspection) of the form of a bit of certain knowledge, but as the *demonstrative expression* of the proposition. Alternatively, the expressive demonstration of the proposition. So whereas a classic proof is something that must be "seen" in order to be grasped, a type-theoretic proof is something that must be actively *done*, not merely passively observed. One must be able to follow the construction of the proof.

Proof of a non-propositional type Classically, one only proves propositions, not terms. So the idea of e.g. "proving" the natural numbers doesn't even make sense; it reflects a category mistake. But in HoTT, the concept of "proving" a type is primitive; the problem is that "proving" is the wrong word.

So here's one way to look at it: we construct (make) proofs; but the proofs we construct are expressions of the type (the thing we prove).

3.2 Proof

Ed. note 3.4 Critical point: we want to draw a sharp contrast between the classic and pragmatist conceptions of proof. Classic: the business of proof is to preserve truth; valid inference is defined in terms of truth-preservation; and the knowledge dispensed by proof is *knowledge that* the conclusion is true of the premises are true. Pragmatist: validity of inference is grounded in proprieties (norms) of practice rather than truth; valid inference *expresses* good (material) inference; the knowledge expressed by proof is *know-how*: a practical skill of doing rather than an epistemic state of knowing.

Truth has no substantive role to play; it is just a complement we pay to the premises and conclusions of good inference.

In a nutshell: classically, good inference is what preserves truth; proof is "truth-conditional". Pragmatism turns this upside down and says that truth is what good inference preserves; truth is proof-conditional. The former starts with truth and derives proof; the latter starts with good inference and derives truth.

Relevance to type theory: intuitionistic type theory grew out of what can be called a pragmatist tradition in mathematics and logic, even if the key players did not think of themselves as pragmatists. The "intuitionistic" bit is key; the concept of "type" is neutral with respect to these issues; both classic and pragmatic approaches can use it, and neither can claim exclusive rights to it. ITT provides a pragmatist interpretation of the concept. One of the key themes of Martin-Löf's theory, for example, is an account of assertion (judgment); this is fundamental because it explains the meanings of propositions and proofs. Dummett is a key figure here, cited explicitly by Brandom as a major influence (I don't know if Martin-Löf cites him.) Gentzen, too: natural deduction as a meaning-is-use model of reasoning.

Another way to put it: reality does not tell us which inferences are good, nor which premises are true. That's something we decide, by settling on normative practices.

TODO: centrality of concept of meaning-as-use; role of idea of theory of meaning as essential – Dummett, but also Frege, etc.

Ed. note 3.5 TODO: concise schematic account of classic concept of proof. To prove a proposition is to...? Construct a linguistic expression that corresponds to the truth? True premises plus truth-preserving inferences. Writing a concise account is a bit hairy, since we need to account for both syntax and semantics, axioms and deductions. Basic idea: it's not about construction but about representation. How one goes about making a proof is irrelevant; all that matters is the truth value of the result, meaning that the result must "mirror" objective reality. Even inferences are essentially axiomatic; the truth tables *define* the logical constants and the inferences they are involved in. Proof involves checking the truth values of propositions and inferences in a structure. In other words, truth tells us whether the proof is good or not. Contrast pragmatism: proof tells us whether conclusion is true or not.

Ed. note 3.6 Another critical point: pragmatism is about language, that is discursive practice. It denies that there is anything interesting (read: useful) to say about Truth (capital T), but it has plenty to say about the role that the concept (and terminology) of truth plays in our practical doings and sayings. It's an extremely useful concept, and without linguistic devices like "... is true" we would find discourse vastly more difficult (but not impossible in principle). But that does not compel us think that Truth is a substantial property, of that something called "Truth" exists. Or more specifically, that true propositions are true in virtue of their correspond to "reality".

To prove a proposition is to justify it; in Brandom's pragmatist model, this means to to vindicate *entitlement* to *commitment* to the proposition that is *expressed by asserting* it. There are three ways to do this:

Demonstration One can *say explicitly* what entitles commitment to the proposition. Informally, this means laying out the reasons for it - the chain of inferences in which the proposition plays the role of conclusion or premise.[9] Formally, this means stating a proof in the traditional sense of a schedule of propositions linked by inferences licensed by the deductive system. But validity of such a proof is not to be construed in terms of truth-preservation; rather, it is instituted by normative practices.[10] What counts as good inference for us is determined by what the community has decided to *treat* as good inference, not by what Reality has to say about the matter.

Appeal to Authority Continually articulating explicity proofs would be impossible in practice; but the rational structure of discursive practice means that we can also cite the authority of others who have asserted the proposition. Formally, this means we can simply refer to previously proven propositions instead of repeating their proofs. If the a step in the proof of your proposition depends on the Pythagorean Theorem, for example, you can appeal to the theorem by name to justify that step, rather than proving it explicitly. In Brandom's idiom, what makes this work involves a concept of *intra*personal inheritance of entitlement; it is an aspect of the essentially social nature of discourse and thus reasoning.

Reliable Disposition The trickiest of Brandom's three techniques of justification involves the appeal to the commitments of reliable reporters. By that he means people whose reports are reliable even

[9] Note that this goes both ways, upstream and downstream.

[10] Or, the deductive system is not truth-conditional, but pragmatic (constructive): it encodes know-how rather than knowledge-that. We don't know *that* the truth of the conclusion preserves the truth of the premises; rather we grasp the norms governing correct *use* of the premises, and the normative practices involving *how* to get from the premises to the conclusion—the concept of truth need not ever enter the picture.

if they are unable to provide explicit justification of them. He gives the example of an expert in a certain kind of pottery, whose ("instinctive") judgments as to whether a particular shard is or is not an example of that kind are generally considered reliable.

Ed. note 3.7 I'm not sure where reliable reporting fits into formal reasoning. In general I don't think it plays much of a role in mathematical or logical proofs; mathematicians and logicians do not as a rule simply accept the word of somebody just because his intuitions have proven reliable in the past. They might accept it informally and say that he is probabaly right, but they would nonetheless demand that eventually the intuition be backed by explicit proof. Maybe at a very basic level, such as the intuition that there is a unity and a plurality, is a reliable disposition that we attribute to just about everybody. Brouwer's account of the subjective origin of basic mathematical concepts in terms of intuitions about time, etc. should be mentioned here.

3.3 Logical Consequence

[11]

[12]

[13]

[14]

[15]

3.4 Inference and Deduction

Gentzen

etc

3.5 Of the Ambiguity of Of

"Of" supports two distinct readings. Consider "the conviction of the defendant". If the court did the convicting, then "of" acts as a kind of intermediary between a verbal noun ("conviction" as act or action of convicting) and its direct object (e.g. "The court convicted the defendant"). The conviction affects the defendant from the outside; it does not "belong" to the defendant but to the court. On the other hand, if we take "the conviction of the defendant" to refer to a belief to which the defendant is firmly committed, then the conviction is "internal"; it belongs to and comes from the defendant.

[11] Dag Prawitz. Logical consequence: A constructivist view. In *The Oxford Handbook of Philosophy of Mathematics and Logic*. Oxford, 2005

[12] Dag Prawitz. Meaning approached via proofs. *Synthese*, 148(3):507–524, February 2006. URL http://www.jstor.org/stable/20118707

[13] Dag Prawitz. Inference and knowledge. In *Logica Yearbook 2008*, pages 183–200. College Publications, London, 2009. URL http://www.jstor.org.proxy.uchicago.edu/stable/20116093

[14] Dag Prawitz. The epistemic significance of valid inference. *Synthese*, pages 1–12, March 2011. DOI: 10.1007/s11229-011-9907-7

[15] Dag Prawitz. Truth as an epistemic notion. *Topoi*, pages 1–8. DOI: 10.1007/s11245-011-9107-6

This ambiguity of "of" afflicts phrases like "proof of a proposition" as well. If we can disambiguate it some of the mystery of the relation between types and proofs will vanish.

3.6 Demonstrations and Demonstratives

When we *exhibit* a classic proof of a proposition, the proof comes out as external to the proposition proved, just as a court's conviction of a defendant is external to the defendant. Such a proof is something added or attached to the proposition.

But when we *demonstrate* a proposition,[16] the demonstration (that is, proof) is to be deemed an expression of the proposition in the internal sense: an expression whose source, so to speak, is the proposition itself, rather than the writer of the proof. This may sound odd or even ridiculously anthropomorphic, but if you think about it a bit it makes perfect sense. The mathematical proofs we write down are not really expressions our our thought, but of mathematical structures, entities, relations etc. So they express mathematics.[17]

[16] Note: we demonstrate propositions, not proofs; a demonstration of a proposition *is* a proof.

[17] Actually we should probably think of them as having a dual expressivism. On the one hand they clearly express mathematics; but on the other hand, the particular form a proof takes is an expression of the writer's style or way of thinking.

We can think of a demonstration in this sense as expressing a type's structure, construed as the inferential articulation of the concept of the type.[18]

[18] See §18 for more on the inferential articulation of conceptual content.

The nice thing about this way of thinking is that it resolves the tension between propositional and non-propositional types with respect to proof. In both cases, what HoTT calls proof or witness is to be taken as a demonstrative expression, or expressive demonstration, of the type itself. In the case of propositional types, favor the term "demonstration", with its connotations of progressive unfolding of a logical structure, or better, rational argument. In the case of non-propositional types like $\mathbb{N}$, favor the term "demonstrative", with its adjectival sense of "something having a demonstrative function", rather than a nominal sense of "act or action of demonstrating". So an element[19] of a propositional type we would call a demonstration of the type, and an element of a non-propositional type we would call a demonstrative of the type.

[19] We really must get rid of "element"; it's too suggestive of set theory. Maybe "demonstrative" fits the bill; instead of "element of a type" we would say "demonstrative of a type". Or maybe "demonstrator".
So 2 is a demonstrative of the natural numbers; a proof that "2 is even" is a demonstration that expresses just that "2 is even".

Ed. note 3.8 Demonstration qua demonstration of know-how? Expression as expression of a type's structure - that is, its inferential articulation?

In both cases we have demonstration rather than proof of the type.

Ed. note 3.9 "Demonstrator" as the genus of "demonstration" and "demonstrative". It has the virtue of paralleling "constructor".

Assertion and Judgment

The account of judgment offered in the HoTT Book doesn't really work. Ditto for Martin-Löf's account. For example, it makes sense to say "P is a proposition", but it doesn't make sense to say "P is a judgment". That's because judgment is a act, something one does.

On the other hand, "judgment", like "proposition", can be treated as a verbal noun or as a "plain" noun. Saying "P is a proposition" is usually taken to mean that P refers to what has been proposed. There is no obvious reason not to treat "P is a judgment" in a similar manner: P refers to what has been judged.

However, there is a difference. Judging a proposition (what was proposed) amounts to *evaluating* what was proposed, as good or bad, true or false, or whatever. By contrast, proposing a proposition amounts to merely exhibiting it for consideration. This arguably involves an implicit evaluation - to propose a proposition is to implicitly claim that it is good, or true, etc. But proposing does not involve offering an evaluation that is distinct from what is proposed, whereas judgment does. The two are distinct kinds of speech act, and refering to the content of a speech act is not the same as referring to the speech act itself.

Furthermore, it is not correct to treat the nominal sense of "judgment" as being the content, what has been judged. The nominal sense of "judgment" refers to the act of judgment itself, and not the proposition judged.

Actually, by the same reasoning it is not correct to say that the nominal sense of "proposition" is what-is-proposed; rather, it is the act proposing, nominalized. This makes perfect sense when you consider that "proposing" can also be nominalized; "the proposing" is another way of saying "the proposition".

The same goes for all -tion words: suggestion, opposition, etc. In each case, the word can refer to the doing, or to what is done, and what is done is always the act of doing itself – not the subject or object of the doing.

This suggests we should make a distinction between, for example, the content of a proposition and "proposition". But this term seems to be a special case; it has the usual plain noun sense of what-was-proposed, the usual verbal sense of "proposing", but also the nominalized verbal sense of "act of proposing".

(But then the same considerations apply to "judgment". The difference must go back to semantics.)

Remark 1 The Arabic grammatical tradition captures this distinction beautifully, mainly because the structure of the language makes it simple to do so.

Or put it this way: when we judge a proposition like "2+2=4" to be true, the what-was-judged is not "2+2=4" but the truth of "2+2=4".

Remark 2 But how is this different from ordinary predication, like "The triangle is red" as a proposition? Should we say that what is proposed is not that the triangle is red, but the redness of the triangle? No, since we're treating it as a propostion, and the whole thing is proposed (exhibited). If we judge it to be true, then again the judgment

So saying "P is a judgment" is incoherent if P is taken to refer to nothing more than what is proposed. If P refers to a claim of the form "X is true" (or good, etc.), then "P is a judgment" seems to make more sense; but it doesn't, really. P still refers to an unasserted content; to make sense, we would have to say something like "P is a judgment when asserted". More explicitly, "'X is true' is a judgment" (or better, "'X is true' expresses a judgment") only *exhibits* "X is true", which is a proposition, not a judgment. As a proposition it expresses a judgment; but when embedded (equivalently, quoted) it does not express anything.

Remark 3 Compare: "Snow is white" iff snow is white. The quoted bit is a name of the sentence; it counts as a *mention* of the sentence, which has no force. The unquoted version of same is the sentence itself; it counts as a *use* of the sentence, which has assertional force. Obviously, the occurances of "P" in "P is a proposition" and "P is a judgment are names of a proposition and thus mentions. So they have no force.

The key point is Frege's point: the content of a proposition is distinct from the force of the utterance. That means that P in "P is a proposition" is unasserted, just as it is when embedded, as in "If P then Q". The truth of "P is a proposition" is independent of the truth of P.

So even if we take the act of declaring "P" to be an act of judgment, it does not follow that a reference to P is a reference to the act of judging that P. Hence there is no way to make "P is a judgment" work. If we take P to refer to what was judged, that again is a proposition (or propositional content), so "P is a judgment" is incoherent.

Remark 4 We can assert that P, and we can assert P. We can judge that P, but we cannot judge P. I don't think this is a mere grammatical distintion; I think it reflects a genuine semantic difference.

What's the Big Deal about Equality?

Ed. note 11.1 Equality is arguably the most important concept of HoTT, as far as I can tell, because of the "Univalence Axiom".

"In the intensional version of the theory, with which we are concerned here, one thus has two different notions of equality: propositional equality is the notion represented by the identity types, in that two terms are propositionally equal just if their identity type IdA(a,b) is inhabited by a term. By contrast, definitional equality is a primitive relation on terms and is not represented by a type; it behaves much like equality between terms in the simply-typed lambda-calculus, or any conventional equational theory.

If the terms a and b are definitially equal, then (since they can be freely substituted for each other) they are also propositionally equal; but the converse is generally not true in the intensional version of the theory"[29]

"The constructive character, computational tractability, and proof-theoretic clarity of the type theory are owed in part to this rather subtle treatment of equality between terms, which itself is expressible within the theory using the identity types IdA(a, b)."[30]

[29] Steve Awodey. Type theory and homotopy. URL http://www.andrew.cmu.edu/user/awodey/preprints/TTH.pdf

[30] Steve Awodey. Type theory and homotopy. URL http://www.andrew.cmu.edu/user/awodey/preprints/TTH.pdf

11.1 Substitution

As the quote from Awodey above suggests, the concept of substitutability plays a basic role.

Ed. note 11.2 Compare substitution in lambda calculus, and in Brandom's model. Maybe something about combinatory logic and the elimination of variables?

Expressivity

Instead of “P is a proposition” etc. we should say “P expresses a proposition”.

Determinism

Hypothesis: classical math with LEM and AC is inherently non-deterministic. Constructive math(s) and logic(s) that discard LEM and AC are deterministic.

Modality

Classic proofs (that use LEM) are modal. Consider the way a classic LEM-dependent proof works. You start by stating the hypothesis: P is true. You assume that P is not true; then you derive a contradiction. The conclusion is not merely that P is true, however; it is that P *must* be true.

Constructive proofs, by contrast, are not modal. They do not say what must be the case, they say what *is* the case. Or rather, they *show* what is the case. (I leave aside the question of whether what is, is necessary.)

Habeus Corpus Logics

The principle of “Habeus Corpus”, from the Latin “(You shall) have the body”, was once enshrined as a fundamental principle of Anglo-American law. It was used to force the State to present a detainee in person before the court, back in the days when we occasionally had the temerity to question the wisdom of the State when it tried to disappear people.

Type theory, and constructive logics generally, operate under a writ of habeus corpus that is permanently in effect. Except that this writ requires the production not of a detainee, but of a witness. If you claim to have a proof, you must produce a witness who is competent to testify to that fact. So its actually more like a law of evidence, but I’m too lazy to come up with a clever legalism to express that idea.

Frege

TODO The Frege Point; force v. content, etc.

Martin-Löf

TODO Summarize ML’s remarks on assertion, proposition, etc.

From Truth to Testimony

We have propositions as types, and we have non-propositional types like $\mathbb{N}$. There is an obvious conflict of intuitions here. Propositions, like $1 > 0$, have truth conditions; names like $\mathbb{N}$ do not. How can they be the same kind of thing?

I think the way out of this embarrassment is recognition that the classic concept of truth is not relevant to type theory; or, in a more positive vein, that only a deflationary or minimalist notion of truth should be used in type theory. In type theory one does not say that a proposition *is* true or false; instead one says that a (propositional) type is proven or disproven, or that either it or its negation has a witness[33]. Instead of a concept of truth we have a concept of testimony. Of course, ordinarily witnesses testify as to the truth of some proposition; but the witnesses of type theory do more than that—or rather they do something else, namely they produce or "make" the proposition.

[33] Or, a "maker" or "constructor".

19.1 Proof, Witness, Constructor

Type theory seems to have settled on an idiom; one says, for example, that types have or do not have proofs or witnesses. But there are problems with both of these terms. The former covers too much ground since it includes non-constructive proofs. The latter invokes a misleading metaphor, since a witness testifies to the truth, whereas a type-theoretic witness to a type constructs (makes, produces, etc.) something. In the case of propositional types, constructors "make" the proposition (in the sense that they are inferences that terminate in the proposition); in the case of non-propositional types, constructors make "elements" of the type, which serve as proxies(?) for the type.

Remark 6 Problem: here again propositional and non-propositional types behave differently. Every proof of a proposition has the proposition as its conclusion; they are all "the same" because they all have that element in the same structural position. But the proof of e.g. $\mathbb{N}$ is different. For example, 2 is a witness for $\mathbb{N}$. Or rather, anything that constructs 2 is such a witness. What all such constructions have in common is 2, not $\mathbb{N}$. So they are all clearly proofs of 2, but we want them to be proofs of $\mathbb{N}$. How do we get there?

The Language of HoTT

One way to think about mathematics and logic is in terms of objects, structures, relations, and the like. etc.

But one can also think of it in terms of vocabularies (or idioms, etc.). Then mastering a discipline is not just a matter grasping some content, but also of acquiring practical mastery over a vocabulary.

The vocabulary of set theory has dominated mathematical discourse for most of the last 100 years or so. Starting in the late 1940s, a competing vocabulary based on category theory began to emerge. Today it is not uncommon to see both vocabularies deployed in the same discourse (lecture, paper).

TODO Type theory as a vocabulary - mostly confined to logic, then computer science. Etc. HoTT as the latest distinctive vocab. - covering both math and compsci, also regions of logic. Significantly different that both set theory and classic logic.

"it is possible to directly formalize the world of homotopy types using the class of languages called dependent type systems and in particular Martin-Lof type systems." V. Voevodsky http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/univalent_foundations_project.pdf

Note: "class of languages called dependent type systems" - languages, not theories

"Type systems are syntactic objects which are specified in several steps. First one chooses a formal language L which allows the use of variables and substitution. Then one chooses a collection of relations on the sets of L-expressions with a given set of free variables which is stable under the substitutions. These relations are called the reduction rules and the equivalence relation generated by the reduction rules is called the conversion relation.... A type system based on L is defined as a pair of subsets BB and BBg in the sets of pre-contexts and pre-sequents respectively which satisfy a number of conditions with respect to reduction and substitution. Elements of BB are called the (valid) contexts of a type system and elements of BBg the (valid) sequents of the type system." same, p. 3

HoTT Introduction

Homotopy type theory is a new branch of mathematics that combines aspects of several different fields in a surprising way. It is based on a recently discovered connection between *homotopy theory* and *type theory*. Homotopy theory is an outgrowth of algebraic topology and homological algebra, with relationships to higher category theory; while type theory is a branch of mathematical logic and theoretical computer science. Although the connections between the two are currently the focus of intense investigation, it is increasingly clear that they are just the beginning of a subject that will take more time and more hard work to fully understand. It touches on topics as seemingly distant as the homotopy groups of spheres, the algorithms for type checking, and the definition of weak ∞-groupoids.

Homotopy type theory also brings new ideas into the very foundation of mathematics. On the one hand, there is Voevodsky's subtle and beautiful *univalence axiom*. The univalence axiom implies, in particular, that isomorphic structures can be identified, a principle that mathematicians have been happily using on workdays, despite its incompatibility with the "official" doctrines of conventional foundations. On the other hand, we have *higher inductive types*, which provide direct, logical descriptions of some of the basic spaces and constructions of homotopy theory: spheres, cylinders, truncations, localizations, etc. Both ideas are impossible to capture directly in classical set-theoretic foundations, but when combined in homotopy type theory, they permit an entirely new kind of "logic of homotopy types".

This suggests a new conception of foundations of mathematics, with intrinsic homotopical content, an "invariant" conception of the objects of mathematics — and convenient machine implementations, which can serve as a practical aid to the working mathematician. This is the *Univalent Foundations* program. The present book is intended as a first systematic exposition of the basics of univalent foundations, and a collection of examples of this new style of reasoning — but without requiring the reader to know or learn any formal logic, or to use any computer proof assistant.

We emphasize that homotopy type theory is a young field, and univalent foundations is very much a work in progress. This book should be regarded as a "snapshot" of the state of the field at the time it was written, rather than a polished exposition of an established edifice. As we will discuss briefly later, there are many aspects of homotopy type theory that are not yet fully understood — but as of this writing, its broad outlines seem clear enough. The ultimate theory will probably not look exactly like the one described in this book, but it will surely be *at least* as capable and powerful; we therefore believe that univalent foundations will eventually become a viable alternative to set theory as the “implicit foundation” for the unformalized mathematics done by most mathematicians.

A.1 Type theory

Type theory was originally invented by Bertrand Russell [Russell, 1908], as a device for blocking the paradoxes in the logical foundations of mathematics that were under investigation at the time. It was later developed as a rigorous formal system in its own right (under the name “λ-calculus”) by Alonzo Church [Church, 1933, 1940, 1941]. Although it is not generally regarded as the foundation for classical mathematics, set theory being more customary, type theory still has numerous applications, especially in computer science and the theory of programming languages [Pierce, 2002]. Per Martin-Löf [Martin-Löf, 1998, 1975, 1982, 1984], among others, developed a “predicative” modification of Church’s type system, which is now usually called dependent, constructive, intuitionistic, or simply *Martin-Löf type theory*. This is the basis of the system that we consider here; it was originally intended as a rigorous framework for the formalization of constructive mathematics. In what follows, we will often use “type theory” to refer specifically to this system and similar ones, although type theory as a subject is much broader (see [Sommaruga, 2010, Kamareddine et al., 2004]) for the history of type theory).

“[T]he intuitionistic type theory ([Martin-Löf, 1975]), which I began to develop *solely with the philosophical motive* of clarifying the syntax and semantics of intuitionistic mathematics, may equally well be viewed as a programming language.” [Martin-Löf, 1982] (Emphasis added.)

In type theory, unlike set theory, objects are classified using a primitive notion of *type*, similar to the data-types used in programming languages. These elaborately structured types can be used to express detailed specifications of the objects classified, giving rise to principles of reasoning about these objects. To take a very simple example, the objects of a product type $A \times B$ are known to be of the form (a, b), and so one automatically knows how to construct them and how to decompose them. Similarly, an object of function type $A \to B$ can be acquired from an object of type B parametrized by objects of type A, and can be evaluated at an argument of type A. This rigidly predictable behavior of all objects (as opposed to set theory’s more liberal formation principles, allowing inhomogeneous sets) is one aspect of type theory that has led to its extensive use in verifying the correctness of computer programs. The clear reasoning principles associated with the construction of types also form the basis of modern *computer proof assistants*,which are used for formalizing mathematics and verifying the correctness of formalized proofs. We return to this aspect of type theory below.

One problem in understanding type theory from a mathematical point of view, however, has always been that the basic concept of *type* is unlike that of *set* in ways that have been hard to make precise. We believe that the new idea of regarding types, not as strange sets (perhaps constructed without using classical logic), but as spaces, viewed from the perspective of homotopy theory, is a significant step forward. In particular, it solves the problem of understanding how the notion of equality of elements of a type differs from that of elements of a set.

In homotopy theory one is concerned with spaces and continuous mappings between them, up to homotopy. A *homotopy* between a pair of continuous maps $f : X \to Y$ and $g : X \to Y$ is a continuous map $H : X \times [0,1] \to Y$ satisfying $H(x,0) = f(x)$ and $H(x,1) = g(x)$. The homotopy H may be thought of as a "continuous deformation" of f into g. The spaces X and Y are said to be *homotopy equivalent*, $X \simeq Y$, if there are continuous maps going back and forth, the composites of which are homotopical to the respective identity mappings, i.e., if they are isomorphic "up to homotopy". Homotopy equivalent spaces have the same algebraic invariants (e.g., homology, or the fundamental group), and are said to have the same *homotopy type*.

A.2 Homotopy type theory

Homotopy type theory (HoTT) interprets type theory from a homotopical perspective. In homotopy type theory, we regard the types as "spaces" (as studied in homotopy theory) or higher groupoids, and the logical constructions (such as the product $A \times B$) as homotopy-invariant constructions on these spaces. In this way, we are able to manipulate spaces directly without first having to develop point-set topology (or any combinatorial replacement for it, such as the theory of simplicial sets). To briefly explain this perspective, consider first the basic concept of type theory, namely that the *term* a is of *type* A, which is written:

$$a : A.$$

This expression is traditionally thought of as akin to:

"a is an element of the set A".

However, in homotopy type theory we think of it instead as:

"a is a point of the space A".

Similarly, every function $f : A \to B$ in type theory is regarded as a continuous map from the space A to the space B.

We should stress that these "spaces" are treated purely homotopically, not topologically. For instance, there is no notion of "open subset" of a type or of "convergence" of a sequence of elements of a type. We only have "homotopical" notions, such as paths between points and homotopies between paths, which also make sense in other models of homotopy theory (such as simplicial sets). Thus, it would be more accurate to say that we treat types as ∞-*groupoids*; this is a name for the "invariant objects" of homotopy theory which can be presented by topological spaces, simplicial sets, or any other model for homotopy theory. However, it is convenient to sometimes use topological words such as "space" and "path", as long as we remember that other topological concepts are not applicable.

(It is tempting to also use the phrase *homotopy type* for these objects, suggesting the dual interpretation of "a type (as in type theory) viewed homotopically" and "a space considered from the point of view of homotopy theory". The latter is a bit different from the classical meaning of "homotopy type" as an *equivalence class* of spaces modulo homotopy equivalence, although it does preserve the meaning of phrases such as "these two spaces have the same homotopy type".)

The idea of interpreting types as structured objects, rather than sets, has a long pedigree, and is known to clarify various mysterious aspects of type theory. For instance, interpreting types as sheaves helps explain the intuitionistic nature of type-theoretic logic, while interpreting them as partial equivalence relations or "domains" helps explain its computational aspects. It also implies that we can use type-theoretic reasoning to study the structured objects, leading to the rich field of categorical logic. The homotopical interpretation fits this same pattern: it clarifies the nature of *identity* (or equality) in type theory, and allows us to use type-theoretic reasoning in the study of homotopy theory.

The key new idea of the homotopy interpretation is that the logical notion of identity $a = b$ of two objects $a, b : A$ of the same type A can be understood as the existence of a path $p : a \rightsquigarrow b$ from point a to point b in the space A. This also means that two functions $f, g : A \to B$ can be identified if they are homotopic, since a homotopy is just a (continuous) family of paths $p_x : f(x) \rightsquigarrow g(x)$ in B, one for each $x : A$. In type theory, for every type A there is a (formerly somewhat mysterious) type Id_A of identifications of two objects of A; in homotopy type theory, this is just the *path space* A^I of all continuous maps $I \to A$ from the unit interval. In this way, a term $p : \mathsf{Id}_A(a, b)$ represents a path $p : a \rightsquigarrow b$ in A.

The idea of homotopy type theory arose around 2006 in independent work by Awodey and Warren [Awodey and Warren, 2009] and Voevodsky [Voevodsky, 2006], but it was inspired by Hofmann and Streicher's earlier groupoid interpretation [Hofmann and Streicher, 1998]. Indeed, higher-dimensional category theory (particularly the theory of weak ∞-groupoids) is now known to be intimately connected to homotopy theory, as proposed by Grothendieck and now being studied intensely by mathematicians of both sorts. The original semantic models of Awodey–Warren and Voevodsky use well-known notions and techniques from homotopy theory which are now also in use in higher category theory, such as Quillen model categories and Kan simplicial sets.

Voevodsky recognized that the simplicial interpretation of type theory satisfies a further crucial property, dubbed *univalence*, which had not previously been considered in type theory (although Church's principle of extensionality for propositions turns out to be a very special case of it). Adding univalence to type theory in the form of a new axiom has far-reaching consequences, many of which are natural, simplifying and compelling. The univalence axiom also further strengthens the homotopical view of type theory, since it holds in the simplicial model and other related models, while failing under the view of types as sets.

A.3 Univalent foundations

Very briefly, the basic idea of the univalence axiom can be explained as follows. In type theory, one can have a universe $\mathcal{U}$, the terms of which are themselves types, $A : \mathcal{U}$, etc. Those types that are terms of $\mathcal{U}$ are commonly called *small* types. Like any type, $\mathcal{U}$ has an identity type $\mathsf{Id}_{\mathcal{U}}$, which expresses the identity relation $A = B$ between small types. Thinking of types as spaces, $\mathcal{U}$ is a space, the points of which are spaces; to understand its identity type, we must ask, what is a path $p : A \rightsquigarrow B$ between spaces in $\mathcal{U}$? The univalence axiom says that such paths correspond to homotopy equivalences $A \simeq B$, (roughly) as explained above. A bit more precisely, given any (small) types A and B, in addition to the primitive type $\mathsf{Id}_{\mathcal{U}}(A, B)$ of identifications of A with B, there is the defined type $\mathsf{Equiv}(A, B)$ of equivalences from A to B. Since the identity map on any object is an equivalence, there is a canonical map,

$$\mathsf{Id}_{\mathcal{U}}(A, B) \to \mathsf{Equiv}(A, B).$$

The univalence axiom states that this map is itself an equivalence. At the risk of oversimplifying, we can state this succinctly as follows:

Univalence Axiom: $(A = B) \;\simeq\; (A \simeq B)$.

In other words, identity is equivalent to equivalence. In particular, one may say that "equivalent types are identical". However, this phrase is somewhat misleading, since it may sound like a sort of "skeletality" condition which *collapses* the notion of equivalence to coincide with identity, whereas in fact univalence is about *expanding* the notion of identity so as to coincide with the (unchanged) notion of equivalence.

From the homotopical point of view, univalence implies that spaces of the same homotopy type are connected by a path in the universe $\mathcal{U}$, in accord with the intuition of a classifying space for (small) spaces. From the logical point of view, however, it is a radically new idea: it says that isomorphic things can be identified! Mathematicians are of course used to identifying isomorphic structures in practice, but they generally do so by "abuse of notation", or some other informal device, knowing that the objects involved are not "really" identical. But in this new foundational scheme, such structures can be formally identified, in the logical sense that every property or construction involving one also applies to the other. Indeed, the identification is now made explicit, and properties and constructions can be systematically transported along it. Moreover, the different ways in which such identifications may be made themselves form a structure that one can (and should!) take into account.

Thus in sum, for points A and B of the universe $\mathcal{U}$ (i.e., small types), the univalence axiom identifies the following three notions:

- (logical) an identification $p : A = B$ of A and B
- (topological) a path $p : A \rightsquigarrow B$ from A to B in $\mathcal{U}$
- (homotopical) an equivalence $p : A \simeq B$ between A and B.

A.4 Higher inductive types

One of the classical advantages of type theory is its simple and effective techniques for working with inductively defined structures. The simplest nontrivial inductively defined structure is the natural numbers, which is inductively generated by zero and the successor function. From this statement one can algorithmically extract the principle of mathematical induction, which characterizes the natural numbers. More general inductive definitions encompass lists and well-founded trees of all sorts, each of which is characterized by a corresponding "induction principle". This includes most data structures used in certain programming languages; hence the usefulness of type theory in formal reasoning about the latter. If conceived in a very general sense, inductive definitions also include examples such as a disjoint union $A + B$, which may be regarded as "inductively" generated by the two injections $A \to A + B$ and $B \to A + B$. The "induction principle" in this case is "proof by case analysis", which characterizes the disjoint union.

In homotopy theory, it is natural to consider also "inductively defined spaces" which are generated not merely by a collection of *points*, but also by collections of *paths* and higher paths. Classically, such spaces are called *CW complexes*. For instance, the circle S^1 is generated by a single point and a single path from that point to itself. Similarly, the 2-sphere S^2 is generated by a single point b and a single two-dimensional path from the constant path at b to itself, while the torus T^2 is generated by a single point, two paths p and q from that point to itself, and a two-dimensional path from $p \cdot q$ to $q \cdot p$.

By using the identification of paths with identities in homotopy type theory, these sort of "inductively defined spaces" can be characterized in type theory by "induction principles", entirely analogously to classical examples such as the natural numbers and the disjoint union. The resulting *higher inductive types* give a direct "logical" way to reason about familiar spaces such as spheres, which (in combination with univalence) can be used to perform familiar arguments from homotopy theory, such as calculating homotopy groups of spheres, in a purely formal way. The resulting proofs are a marriage of classical homotopy-theoretic ideas with classical type-theoretic ones, yielding new insight into both disciplines.

Moreover, this is only the tip of the iceberg: many abstract constructions from homotopy theory, such as homotopy colimits, suspensions, Postnikov towers, localization, completion, and spectrification, can also be expressed as higher inductive types. Many of these are classically constructed using Quillen's "small object argument", which can be regarded as a finite way of algorithmically describing an infinite CW complex presentation of a space, just as "zero and successor" is a finite algorithmic description of the infinite set of natural numbers. Spaces produced by the small object argument are infamously complicated and difficult to understand; the type-theoretic approach is potentially much simpler, bypassing the need for any explicit construction by giving direct access to the appropriate "induction principle". Thus, the combination of univalence and higher inductive types suggests the possibility of a revolution, of sorts, in the practice of homotopy theory.

A.5 Sets in univalent foundations

We have claimed that univalent foundations can eventually serve as a foundation for "all" of mathematics, but so far we have discussed only homotopy theory. Of course, there are many specific examples of the use of type theory without the new homotopy type theory features to formalize mathematics, such as the recent formalization of the Feit–Thompson odd-order theorem in Coq [Gonthier et al., 2013].

But the traditional view is that mathematics is founded on set theory, in the sense that all mathematical objects and constructions can be coded into a theory such as Zermelo–Fraenkel set theory (ZF). However, it is well-established by now that for most mathematics outside of set theory proper, the intricate hierarchical membership structure of sets in ZF is really unnecessary: a more "structural" theory, such as Lawvere's Elementary Theory of the Category of Sets [Lawvere, 2005], suffices.

In univalent foundations, the basic objects are "homotopy types" rather than sets, but we can *define* a class of types which behave like sets. Homotopically, these can be thought of as spaces in which every connected component is contractible, i.e. those which are homotopy equivalent to a discrete space. It is a theorem that the category of such "sets" satisfies Lawvere's axioms (or related ones, depending on the details of the theory). Thus, any sort of mathematics that can be represented in an ETCS-like theory (which, experience suggests, is essentially all of mathematics) can equally well be represented in univalent foundations.

This supports the claim that univalent foundations is at least as good as existing foundations of mathematics. A mathematician working in univalent foundations can build structures out of sets in a familiar way, with more general homotopy types waiting in the foundational background until there is need of them. For this reason, most of the applications in this book have been chosen to be areas where univalent foundations has something *new* to contribute that distinguishes it from existing foundational systems.

Unsurprisingly, homotopy theory and category theory are two of these, but perhaps less obvious is that univalent foundations has something new and interesting to offer even in subjects such as set theory and real analysis. For instance, the univalence axiom allows us to identify isomorphic structures, while higher inductive types allow direct descriptions of objects by their universal properties. Thus we can generally avoid resorting to arbitrarily chosen representatives or transfinite iterative constructions. In fact, even the objects of study in ZF set theory can be characterized, inside the sets of univalent foundations, by such an inductive universal property.

A.6 Informal type theory

One difficulty often encountered by the classical mathematician when faced with learning about type theory is that it is usually presented as a fully or partially formalized deductive system. This style, which is very useful for proof-theoretic investigations, is not particularly convenient for use in applied, informal reasoning. Nor is it even familiar to most working mathematicians, even those who might be interested in foundations of mathematics. One objective of the present work is to develop an informal style of doing mathematics in univalent foundations that is at once rigorous and precise, but is also closer to the language and style of presentation of everyday mathematics.

In present-day mathematics, one usually constructs and reasons about mathematical objects in a way that could in principle, one presumes, be formalized in a system of elementary set theory, such as ZFC — at least given enough ingenuity and patience. For the most part, one does not even need to be aware of this possibility, since it largely coincides with the condition that a proof be "fully rigorous" (in the sense that all mathematicians have come to understand intuitively through education and experience). But one does need to learn to be careful about a few aspects of “informal set theory”: the use of collections too large or inchoate to be sets; the axiom of choice and its equivalents; even (for undergraduates) the method of proof by contradiction; and so on. Adopting a new foundational system such as homotopy type theory as the *implicit formal basis* of informal reasoning will require adjusting some of one’s instincts and practices. The present text is intended to serve as an example of this “new kind of mathematics”, which is still informal, but could now in principle be formalized in homotopy type theory, rather than ZFC, again given enough ingenuity and patience.

It is worth emphasizing that, in this new system, such formalization can have real practical benefits. The formal system of type theory is suited to computer systems and has been implemented in existing proof assistants. A proof assistant is a computer program which guides the user in construction of a fully formal proof, only allowing valid steps of reasoning. It also provides some degree of automation, can search libraries for existing theorems, and can even extract numerical algorithms from the resulting (constructive) proofs.

We believe that this aspect of the univalent foundations program distinguishes it from other approaches to foundations, potentially providing a new practical utility for the working mathematician. Indeed, proof assistants based on older type theories have already been used to formalize substantial mathematical proofs, such as the four-color theorem and the Feit–Thompson theorem. Computer implementations of univalent foundations are presently works in progress (like the theory itself). However, even its currently available implementations (which are mostly small modifications to existing proof assistants such as Coq and Agda) have already demonstrated their worth, not only in the formalization of known proofs, but in the discovery of new ones. Indeed, many of the proofs described in this book were actually *first* done in a fully formalized form in a proof assistant, and are only now being “unformalized” for the first time — a reversal of the usual relation between formal and informal mathematics.

One can imagine a not-too-distant future when it will be possible for mathematicians to verify the correctness of their own papers by working within the system of univalent foundations, formalized in a proof assistant, and that doing so will become as natural as typesetting their own papers in TeX. In principle, this could be equally true for any other foundational system, but we believe it to be more practically attainable using univalent foundations, as witnessed by the present work and its formal counterpart.

A.7 Constructivity

One of the most striking differences between classical foundations and type theory is the idea of *proof relevance*, according to which mathematical statements, and even their proofs, become first-class mathematical objects. In type theory, we represent mathematical statements by types, which can be regarded simultaneously as both mathematical constructions and mathematical assertions, a conception also known as *propositions as types*. Accordingly, we can regard a term $a : A$ as both an element of the type A (or in homotopy type theory, a point of the space A), and at the same time, a proof of the proposition A.[42] To take an example, suppose we have sets A and B (discrete spaces), and consider the statement "A is isomorphic to B". In type theory, this can be rendered as:

[42] This suggests (to me, anyway) that "element of type A" and "proof of the proposition A" are different things, and $a : A$ can be used to name both. But I don't think that's right; $a : A$ is one thing. Better to say that they are distinct ways of looking at one thing.

$$\mathsf{Iso}(A,B) :\equiv \sum_{(f:A\to B)} \sum_{(g:B\to A)} \Big(\big(\textstyle\prod_{(x:A)} g(f(x)) = x\big) \times \big(\textstyle\prod_{(y:B)} f(g(y)) = y\big) \Big).$$

Reading the type constructors $\Sigma, \Pi, \times$ here as "there exists", "for all", and "and" respectively yields the usual formulation of "A and B are isomorphic"; on the other hand, reading them as sums and products yields the *type of all isomorphisms* between A and B! To prove that A and B are isomorphic, one constructs a proof $p : \mathsf{Iso}(A,B)$, which is therefore the same as constructing an isomorphism between A and B, i.e., exhibiting a pair of functions f, g together with *proofs* that their composites are the respective identity maps. The latter proofs, in turn, are nothing but homotopies of the appropriate sorts. In this way, *proving a proposition is the same as constructing an element of some particular type*. In particular, to prove a statement of the form "A and B" is just to prove A and to prove B, i.e., to give an element of the type $A \times B$. And to prove that A implies B is just to find an element of $A \to B$, i.e. a function from A to B (determining a mapping of proofs of A to proofs of B).

The logic of propositions-as-types is flexible and supports many variations, such as using only a subclass of types to represent propositions. In homotopy type theory, there are natural such subclasses arising from the fact that the system of all types, just like spaces in classical homotopy theory, is "stratified" according to the dimensions in which their higher homotopy structure exists or collapses. In particular, Voevodsky has found a purely type-theoretic definition of *homotopy n-types*, corresponding to spaces with no nontrivial homotopy information above dimension n. (The 0-types are the "sets" mentioned previously as satisfying Lawvere's axioms.) Moreover, with higher inductive types, we can universally "truncate" a type into an n-type; in classical homotopy theory this would be its n^{th} Postnikov section. Particularly important for logic is the case of homotopy (-1)-types, which we call *mere propositions*. Classically, every (-1)-type is empty or contractible; we interpret these possibilities as the truth values "false" and "true" respectively.

Using all types as propositions yields a "constructive" conception of logic (for more on which, see [Kolmogorov, 1932, Troelstra and van Dalen, 1988a,b]), which gives type theory its good computational character. For instance, every proof that something exists carries with it enough information to actually find such an object; and from a proof that "A or B" holds, one can extract either a proof that A holds or one that B holds. Thus, from every proof we can automatically extract an algorithm; this can be very useful in applications to computer programming.

However, this logic does not faithfully represent certain important classical principles of reasoning, such as the axiom of choice and the law of excluded middle. For these we need to use the "(-1)-truncated" logic, in which only the homotopy (-1)-types represent propositions; and under this interpretation, the system is fully compatible with classical mathematics. Homotopy type theory is thus compatible with both constructive and classical conceptions of logic, and many more besides.

More specifically, consider on one hand the *axiom of choice*: "if for every $x : A$ there exists a $y : B$ such that $R(x, y)$, there is a function $f : A \to B$ such that for all $x : A$ we have $R(x, f(x))$." The pure propositions-as-types notion of "there exists" is strong enough to make this statement simply provable — yet it does not have all the consequences of the usual axiom of choice. However, in (-1)-truncated logic, this statement is not automatically true, but is a strong assumption with the same sorts of consequences as its counterpart in classical set theory.

On the other hand, consider the *law of excluded middle*: "for all A, either A or not A." Interpreting this in the pure propositions-as-types logic yields a statement that is inconsistent with the univalence axiom. For since proving "A" means exhibiting an element of it, this assumption would give a uniform way of selecting an element from every nonempty type — a sort of Hilbertian choice operator. Univalence implies that the element of A selected by such a choice operator must be invariant under all self-equivalences of A, since these are identified with self-identities and every operation must respect identity; but clearly some types have automorphisms with no fixed points, e.g. we can swap the elements of a two-element type. However, the "(-1)-truncated law of excluded middle", though also not automatically true, may consistently be assumed with most of the same consequences as in classical mathematics.

In other words, while the pure propositions-as-types logic is "constructive" in the strong algorithmic sense mentioned above, the default

(-1)-truncated logic is "constructive" in a different sense (namely, that of the logic formalized by Heyting under the name "intuitionistic"); and to the latter we may freely add the axioms of choice and excluded middle to obtain a logic that may be called "classical". Thus, the homotopical perspective reveals that classical and constructive logic can coexist, as endpoints of a spectrum of different systems, with an infinite number of possibilities in between (the homotopy n-types for $-1 < n < \infty$). We may speak of "LEM_n" and "AC_n", with AC_∞ being provable and LEM_∞ inconsistent with univalence, while AC_{-1} and LEM_{-1} are the versions familiar to classical mathematicians (hence in most cases it is appropriate to assume the subscript (-1) when none is given). Indeed, one can even have useful systems in which only *certain* types satisfy such further "classical" principles, while types in general remain "constructive".

It is worth emphasizing that univalent foundations does not *require* the use of constructive or intuitionistic logic. Most of classical mathematics which depends on the law of excluded middle and the axiom of choice can be performed in univalent foundations, simply by assuming that these two principles hold (in their proper, (-1)-truncated, form). However, type theory does encourage avoiding these principles when they are unnecessary, for several reasons.

First of all, every mathematician knows that a theorem is more powerful when proven using fewer assumptions, since it applies to more examples. The situation with AC and LEM is no different: type theory admits many interesting "nonstandard" models, such as in sheaf toposes, where classicality principles such as AC and LEM tend to fail. Homotopy type theory admits similar models in higher toposes, such as are studied in [Toën and Vezzosi, 2002, Rezk, 2005, Lurie, 2009]. Thus, if we avoid using these principles, the theorems we prove will be valid internally to all such models.

Secondly, one of the additional virtues of type theory is its computable character. In addition to being a foundation for mathematics, type theory is a formal theory of computation, and can be treated as a powerful programming language. From this perspective, the rules of the system cannot be chosen arbitrarily the way set-theoretic axioms can: there must be a harmony between them which allows all proofs to be "executed" as programs. We do not yet fully understand the new principles introduced by homotopy type theory, such as univalence and higher inductive types, from this point of view, but the basic outlines are emerging; see, for example, [Licata and Harper, 2012]. It has been known for a long time, however, that principles such as AC and LEM are fundamentally antithetical to computability, since they assert baldly that certain things exist without giving any way to compute them. Thus, avoiding them is necessary to maintain the character of type theory as a theory of computation.

Fortunately, constructive reasoning is not as hard as it may seem. In some cases, simply by rephrasing some definitions, a theorem can be made constructive and its proof more elegant. Moreover, in univalent foundations this seems to happen more often. For instance:

(i) In set-theoretic foundations, at various points in homotopy theory and category theory one needs the axiom of choice to perform transfinite constructions. But with higher inductive types, we can encode these constructions directly and constructively. In particular, none of the "synthetic" homotopy theory in **??** requires LEM or AC.
(ii) In set-theoretic foundations, the statement "every fully faithful and essentially surjective functor is an equivalence of categories" is equivalent to the axiom of choice. But with the univalence axiom, it is just *true*; see **??**.
(iii) In set theory, various circumlocutions are required to obtain notions of "cardinal number" and "ordinal number" which canonically represent isomorphism classes of sets and well-ordered sets, respectively — possibly involving the axiom of choice or the axiom of foundation. But with univalence and higher inductive types, we can obtain such representatives directly by truncating the universe; see **??**.
(iv) In set-theoretic foundations, the definition of the real numbers as equivalence classes of Cauchy sequences requires either the law of excluded middle or the axiom of (countable) choice to be well-behaved. But with higher inductive types, we can give a version of this definition which is well-behaved and avoids any choice principles; see **??**.

Of course, these simplifications could as well be taken as evidence that the new methods will not, ultimately, prove to be really constructive. However, we emphasize again that the reader does not have to care, or worry, about constructivity in order to read this book. The point is that in all of the above examples, the version of the theory we give has independent advantages, whether or not LEM and AC are assumed to be available. Constructivity, if attained, will be an added bonus.

Given this discussion of adding new principles such as univalence, higher inductive types, AC, and LEM, one may wonder whether the resulting system remains consistent. (One of the original virtues of type theory, relative to set theory, was that it can be seen to be consistent by proof-theoretic means). As with any foundational system, consistency is a relative question: "consistent with respect to what?" The short answer is that all of the constructions and axioms considered in this book have a model in the category of Kan complexes, due to Voevodsky [Kapulkin et al., 2012] (see [Lumsdaine and Shulman, 2013] for higher inductive types). Thus, they are known to be consistent relative to ZFC (with as many inaccessible cardinals as we need nested univalent universes). Giving a more traditionally type-theoretic account of this consistency is work in progress (see, e.g., [Licata and Harper, 2012, Barras et al., 2013]).

We summarize the different points of view of the type-theoretic operations in Table 1.

Table 1: Comparing points of view on type-theoretic operations

Types	Logic	Sets	Homotopy
A	proposition	set	space
$a : A$	proof	element	point
$B(x)$	predicate	family of sets	fibration
$b(x) : B(x)$	conditional proof	family of elements	section
$\mathbf{0}$,	$\bot, \top$	$\emptyset, \{\emptyset\}$	$\emptyset, *$
$A + B$	$A \vee B$	disjoint union	coproduct
$A \times B$	$A \wedge B$	set of pairs	product space
$A \to B$	$A \Rightarrow B$	set of functions	function space
$\sum_{(x:A)} B(x)$	$\exists_{x:A} B(x)$	disjoint sum	total space
$\prod_{(x:A)} B(x)$	$\forall_{x:A} B(x)$	product	space of sections
Id_A	equality $=$	$\{ (x, x) \mid x \in A \}$	path space A^I

A.8 Open problems

For those interested in contributing to this new branch of mathematics, it may be encouraging to know that there are many interesting open questions.

Perhaps the most pressing of them is the "constructivity" of the Univalence Axiom, posed by Voevodsky in [Voevodsky, 2012]. The basic system of type theory follows the structure of Gentzen's natural deduction. Logical connectives are defined by their introduction rules, and have elimination rules justified by computation rules. Following this pattern, and using Tait's computability method, originally designed to analyse Gödel's Dialectica interpretation, one can show the property of *normalization* for type theory. This in turn implies important properties such as decidability of type-checking (a crucial property since type-checking corresponds to proof-checking, and one can argue that we should be able to "recognize a proof when we see one"), and the so-called "canonicity property" that any closed term of the type of natural numbers reduces to a numeral. This last property, and the uniform structure of introduction/elimination rules, are lost when one extends type theory with an axiom, such as the axiom of function extensionality, or the univalence axiom. Voevodsky has formulated a precise mathematical conjecture connected to this question of canonicity for type theory extended with the axiom of Univalence: given a closed term of the type of natural numbers, is it always possible to find a numeral and a proof that this term is equal to this numeral, where this proof of equality may itself use the univalence axiom? More generally, an important issue is whether it is possible to provide a constructive justification of the univalence axiom. What about if one adds other homotopically motivated constructions, like higher inductive types? These questions remain open at the present time, although methods are currently being developed to try to find answers.

Another basic issue is the difficulty of working with types, such as the natural numbers, that are essentially sets (i.e., discrete spaces), containing only trivial paths. At present, homotopy type theory can really only characterize spaces up to homotopy equivalence, which means that these "discrete spaces" may only be *homotopy equivalent* to discrete spaces. Type-theoretically, this means there are many paths that are equal to reflexivity, but not *judgmentally* equal to it (see **??** for the meaning of "judgmentally"). While this homotopy-invariance has advantages, these "meaningless" identity terms do introduce needless complications into arguments and constructions, so it would be convenient to have a systematic way of eliminating or collapsing them.

A more specialized, but no less important, problem is the relation between homotopy type theory and the research on *higher toposes* currently happening at the intersection of higher category theory and homotopy theory. There is a growing conviction among those familiar with both subjects that they are intimately connected. For instance, the notion of a univalent universe should coincide with that of an object classifier, while higher inductive types should be an "elementary" reflection of local presentability. More generally, homotopy type theory should be the "internal language" of $(\infty, 1)$-toposes, just as intuitionistic higher-order logic is the internal language of ordinary 1-toposes. Despite this general consensus, however, details remain to be worked out — in particular, questions of coherence and strictness remain to be addressed — and doing so will undoubtedly lead to further insights into both concepts.

But by far the largest field of work to be done is in the ongoing formalization of everyday mathematics in this new system. Recent successes in formalizing some facts from basic homotopy theory and category theory have been encouraging; some of these are described in **????**. Obviously, however, much work remains to be done.

The homotopy type theory community maintains a web site and group blog at http://homotopytypetheory.org, as well as a discussion email list. Newcomers are always welcome!

A.9 How to read this book

This book is divided into two parts. **??**, “Foundations”, develops the fundamental concepts of homotopy type theory. This is the mathematical foundation on which the development of specific subjects is built, and which is required for the understanding of the univalent foundations approach. To a programmer, this is “library code”. Since univalent foundations is a new and different kind of mathematics, its basic notions take some getting used to; thus **??** is fairly extensive.

??, “Mathematics”, consists of four chapters that build on the basic notions of **??** to exhibit some of the new things we can do with univalent foundations in four different areas of mathematics: homotopy theory (**??**), category theory (**??**), set theory (**??**), and real analysis (**??**). The chapters in **??** are more or less independent of each other, although occasionally one will use a lemma proven in another.

A reader who wants to seriously understand univalent foundations, and be able to work in it, will eventually have to read and understand most of **??**. However, a reader who just wants to get a taste of univalent foundations and what it can do may understandably balk at having to work through over 200 pages before getting to the “meat” in **??**. Fortunately, not all of **??** is necessary in order to read the chapters in **??**. Each chapter in **??** begins with a brief overview of its subject, what univalent foundations has to contribute to it, and the necessary background from **??**, so the courageous reader can turn immediately to the appropriate chapter for their favorite subject. For those who want to understand one or more chapters in **??** more deeply than this, but are not ready to read all of **??**, we provide here a brief summary of **??**, with remarks about which parts are necessary for which chapters in **??**.

?? is about the basic notions of type theory, prior to any homotopical interpretation. A reader who is familiar with Martin-Löf type theory can quickly skim it to pick up the particulars of the theory we are using. However, readers without experience in type theory will need to read **??**, as there are many subtle differences between type theory and other foundations such as set theory.

?? introduces the homotopical viewpoint on type theory, along with the basic notions supporting this view, and describes the homotopical behavior of each component of the type theory from **??**. It also introduces the *univalence axiom* (**??**) — the first of the two basic innovations of homotopy type theory. Thus, it is quite basic and we encourage everyone to read it, especially **??**–**??**.

?? describes how we represent logic in homotopy type theory, and its connection to classical logic as well as to constructive and intuitionistic logic. Here we define the law of excluded middle, the axiom of choice, and the axiom of propositional resizing (although, for the most part, we do not need to assume any of these in the rest of the book), as well as the *propositional truncation* which is essential for representing traditional logic. This chapter is essential background for **????**, less important for **??**, and not so necessary for **??**.

???? study two special topics in detail: equivalences (and related notions) and generalized inductive definitions. While these are important subjects in their own rights and provide a deeper understanding of homotopy type theory, for the most part they are not necessary for **??**. Only a few lemmas from **??** are used here and there, while the general discussions in **??????** are helpful for providing the intuition required for **??**. The generalized sorts of inductive definition discussed in **??** are also used in a few places in **????**.

?? introduces the second basic innovation of homotopy type theory — *higher inductive types* — with many examples. Higher inductive types are the primary object of study in **??**, and some particular ones play important roles in **????**. They are not so necessary for **??**, although one example is used in **??**.

Finally, **??** discusses homotopy n-types and related notions such as n-connected types. These notions are important for **??**, but not so important in the rest of **??**, although the case $n = -1$ of some of the lemmas are used in **??**.

This completes **??**. As mentioned above, **??** consists of four largely unrelated chapters, each describing what univalent foundations has to offer to a particular subject.

Of the chapters in **??**, **??** (Homotopy theory) is perhaps the most radical. Univalent foundations has a very different "synthetic" approach to homotopy theory in which homotopy types are the basic objects (namely, the types) rather than being constructed using topological spaces or some other set-theoretic model. This enables new styles of proof for classical theorems in algebraic topology, of which we present a sampling, from $\pi_1(\mathbb{S}^1) = \mathbb{Z}$ to the Freudenthal suspension theorem.

In **??** (Category theory), we develop some basic (1-)category theory, adhering to the principle of the univalence axiom that *equality is isomorphism*. This has the pleasant effect of ensuring that all definitions and constructions are automatically invariant under equivalence of categories: indeed, equivalent categories are equal just as equivalent types are equal. (It also has connections to higher category theory and higher topos theory.)

?? (Set theory) studies sets in univalent foundations. The category of sets has its usual properties, hence provides a foundation for any mathematics that doesn't need homotopical or higher-categorical structures. We also observe that univalence makes cardinal and ordinal numbers a bit more pleasant, and that higher inductive types yield a cumulative hierarchy satisfying the usual axioms of Zermelo–Fraenkel set theory.

In **??** (Real numbers), we summarize the construction of Dedekind real numbers, and then observe that higher inductive types allow a definition of Cauchy real numbers that avoids some associated problems in constructive mathematics. Then we sketch a similar approach to Conway's surreal numbers.

Each chapter in this book ends with a Notes section, which collects historical comments, references to the literature, and attributions of results, to the extent possible. We have also included Exercises at the end of each chapter, to assist the reader in gaining familiarity with doing mathematics in univalent foundations.

Finally, recall that this book was written as a massively collaborative effort by a large number of people. We have done our best to achieve consistency in terminology and notation, and to put the mathematics in a linear sequence that flows logically, but it is very likely that some imperfections remain. We ask the reader's forgiveness for any such infelicities, and welcome suggestions for improvement of the next edition.

References

ISO/IEC 13568:2002. *Information technology – Z formal specification notation – Syntax, type system and semantics*. ISO, Geneva, 2002. URL http://www.iso.org/iso/catalogue_detail.htm?csnumber=21573.

Steve Awodey. Type theory and homotopy. URL http://www.andrew.cmu.edu/user/awodey/preprints/TTH.pdf.

Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 146:45–55, 2009.

Bruno Barras, Thierry Coquand, and Simon Huber. A generalization of Takeuti-Gandy interpretation. http://uf-ias-2012.wikispaces.com/file/view/semi.pdf, 2013.

Michael Beeson. *Foundations of Constructive Mathematics*. Springer, 1985.

Robert Brandom. *Making It Explicit: Reasoning, Representing, and Discursive Commitment*. Harvard University Press, June 1998a.

Robert Brandom. *Making It Explicit: Reasoning, Representing, and Discursive Commitment*. Harvard University Press, June 1998b. ISBN 9780674543300.

Robert Brandom. *Articulating reasons : an introduction to inferentialism*. Harvard University Press, Cambridge, Mass.; London, 2001.

Robert B Brandom. Why truth is not important in philosophy. In *Reason in philosophy*, pages 156–176. The Belknap Press of Harvard University Press, Cambridge, Mass., 2009.

Alonzo Church. A set of postulates for the foundation of logic 2. *Annals of Mathematics*, 34:839–864, 1933.

Alonzo Church. A formulation of of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

Alonzo Church. *The Calculi of Lambda Conversation*. Princeton University Press, 1941.

Robert L. Constable, Stuart F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, Robert W. Harper, Douglas J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, James T. Sasaki, and Scott F. Smith. *Implementing Mathematics with the NuPRL Proof Development System*. Prentice Hall, 1986.

Coq Development Team. *The Coq Proof Assistant Reference Manual*. INRIA-Rocquencourt, 2012.

Thierry Coquand. The paradox of trees in type theory. *BIT Numerical Mathematics*, 32(1):10–14, 1992.

Nicolaas Govert de Bruijn. *AUTOMATH, a language for mathematics*. Les Presses de l'Université de Montréal, Montreal, Quebec, 1973. Séminaire de Mathématiques Supérieures, No. 52 (Été 1971).

Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Thery. A machine-checked proof of the odd order theorem. In *Interactive Theorem Proving*, 2013.

Arend Heyting. *Intuitionism: an introduction*. Studies in logic and the foundations of mathematics. North-Holland Pub. Co., 1966.

Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In Giovanni Sambin and Jan M. Smith, editors, *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 83–111. Oxford University Press, New York, 1998.

William A. Howard. The formulae-as-types notion of construction. In J. Roger Seldin, Jonathan P.; Hindley, editor, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980. original paper manuscript from 1969.

Fairouz Kamareddine, Twan Laan, and Rob Nederpelt. *A Modern Perspective on Type Theory: From its Origins until Today*. Number 29 in Applied Logic. Kluwer, 2004.

Chris Kapulkin, Peter LeFanu Lumsdaine, and Vladimir Voevodsky. The simplicial model of univalent foundations, 2012. arXiv:1211.2851.

Andrey Kolmogorov. Zur Deutung der intuitionistischen Logik. *Mathematische Zeitschrift*, 35:58–65, 1932.

F. William Lawvere. An elementary theory of the category of sets (long version) with commentary. *Reprints in Theory and Applications of Categories*, 11:1–35, 2005. ISSN 1201-561X. Reprinted and expanded from Proc. Nat. Acad. Sci. U.S.A. **52** (1964), With comments by the author and Colin McLarty.

F. William Lawvere. Adjointness in foundations. *Reprints in Theory and Applications of Categories*, 16:1–16, 2006. Reprinted from Dialectica **23** (1969).

Daniel R. Licata and Robert Harper. Canonicity for 2-dimensional type theory. In *Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 337–348, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1083-3. DOI: 10.1145/2103656.2103697. URL http://doi.acm.org/10.1145/2103656.2103697.

Peter LeFanu Lumsdaine and Michael Shulman. Higher inductive types. In preparation, 2013.

Jacob Lurie. *Higher topos theory*. Number 170 in Annals of Mathematics Studies. Princeton University Press, 2009. arXiv:math.CT/0608040.

Per Martin-Löf. An intuitionistic theory of types: predicative part. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium '73, Proceedings of the Logic Colloquium*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 73–118. North-Holland, 1975.

Per Martin-Löf. Constructive mathematics and computer programming. In L. Jonathan Cohen, Jerzy Łoś, Helmut Pfeiffer, and Klaus-Peter Podewski, editors, *Logic, Methodology and Philosophy of Science VI, Proceedings of the Sixth International Congress of Logic, Methodology and Philosophy of Science, Hannover 1979*, volume 104 of *Studies in Logic and the Foundations of Mathematics*, pages 153–175. North-Holland, 1982. DOI: 10.1016/S0049-237X(09)70189-2. URL http://dx.doi.org/10.1016/S0049-237X(09)70189-2.

Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in Proof Theory*. Bibliopolis, 1984. ISBN 88-7088-105-9.

Per Martin-Löf. An intuitionistic theory of types. In Giovanni Sambin and Jan M. Smith, editors, *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 127–172. Oxford University Press, 1998.

Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Chalmers, Göteborg University, 2007.

Christine Paulin-Mohring. Inductive Definitions in the System Coq - Rules and Properties. In Marc Bezem and Jan Friso Groote, editors, *Proceedings of the conference Typed Lambda Calculi and Applications*, number 664 in Lecture Notes in Computer Science, 1993.

Benjamin C. Pierce. *Types and Programming Languages*. MIT Press, 2002.

Dag Prawitz. Truth as an epistemic notion. *Topoi*, pages 1–8. DOI: 10.1007/s11245-011-9107-6.

Dag Prawitz. Logical consequence: A constructivist view. In *The Oxford Handbook of Philosophy of Mathematics and Logic*. Oxford, 2005.

Dag Prawitz. Meaning approached via proofs. *Synthese*, 148(3): 507–524, February 2006. URL http://www.jstor.org/stable/20118707.

Dag Prawitz. Inference and knowledge. In *Logica Yearbook 2008*, pages 183–200. College Publications, London, 2009. URL http://www.jstor.org.proxy.uchicago.edu/stable/20116093.

Dag Prawitz. The epistemic significance of valid inference. *Synthese*, pages 1–12, March 2011. DOI: 10.1007/s11229-011-9907-7.

Huw Price. *Naturalism without mirrors*. Oxford University Press, Oxford; New York, 2010.

Huw Price. *Expressivism, pragmatism and representationalism*. Cambridge University Press, Cambridge, UK, 2013.

Charles Rezk. Toposes and homotopy toposes. http://www.math.uiuc.edu/~rezk/homotopy-topos-sketch.pdf, 2005. URL http://www.math.uiuc.edu/~rezk/homotopy-topos-sketch.pdf.

Richard Rorty. *Philosophy and the Mirror of Nature*. Princeton University Press, Princeton, N.J., 2009.

Bertand Russell. Mathematical logic based on the theory of types. *American Journal of Mathematics*, 30:222–262, 1908.

Peter Schroeder-Heister. Validity concepts in proof-theoretic semantics. *Synthese*, 148(3):525–571, February 2006. DOI: 10.1007/s11229-004-6296-1.

Peter Schroeder-Heister. Proof-theoretic semantics. December 2012a. URL http://plato.stanford.edu/archives/win2012/entries/proof-theoretic-semantics/.

Peter Schroeder-Heister. Proof-theoretic semantics, self-contradiction, and the format of deductive reasoning. *Topoi*, 31(1):77–85, April 2012b. DOI: 10.1007/s11245-012-9119-x.

Dana Scott. Constructive validity. In M. Laudet, D. Lacombe, L. Nolin, and M. Schützenberger, editors, *Symposium on Automatic Demonstration*, volume 125, pages 237–275. Springer-Verlag, 1970.

Giovanni Sommaruga. *History and Philosophy of Constructive Type Theory*. Number 290 in Synthese Library. Kluwer, 2010.

Thomas Streicher. Investigations into intensional type theory, 1993. Habilitationsschrift, Ludwig-Maximilians-Universität München.

William W. Tait. Intensional interpretations of functionals of finite type. I. *The Journal of Symbolic Logic*, 32:198–212, 1967. ISSN 0022-4812.

William W. Tait. Constructive reasoning. In *Logic, Methodology and Philos. Sci. III (Proc. Third Internat. Congr., Amsterdam, 1967)*, pages 185–199. North-Holland, Amsterdam, 1968.

Bertrand Toën and Gabriele Vezzosi. Homotopical algebraic geometry I: Topos theory, 2002. arXiv:math/0207028.

Anne Sjerp Troelstra and Dirk van Dalen. *Constructivism in mathematics. Vol. I*, volume 121 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1988a. ISBN 0-444-70266-0; 0-444-70506-6. An introduction.

Anne Sjerp Troelstra and Dirk van Dalen. *Constructivism in mathematics. Vol. II*, volume 123 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1988b. ISBN 0-444-70358-6. An introduction.

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

Vladimir Voevodsky. A very short note on the homotopy λ-calculus. http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/Hlambda_short_current.pdf, 2006.

Vladimir Voevodsky. A universe polymorphic type system. http://uf-ias-2012.wikispaces.com/file/view/Universe+polymorphic+type+sytem.pdf, 2012.