HOTT TYPES

Ed. note 0.0.1 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

1
The Pragmatist Enlightenment

Liberation

Pluralism

Ed. note 1.2.1 Not just propositions-as-types, but types-as-propositions. Example: the type $\mathbb{N}$ can be viewed as a proposition "there exists a natural number". This means that there is no authoritative definition of what a type is, which means that pluralism is an essential aspect of type theory. Is this a sharp contrast with traditional mathematics? For pre-modern mathematics, number was unequivocally quantity or magnitude - no pluralism there. Modern mathematics discarded quantitative interpretations of number in favor of structural notions. The issue of pluralism is not so clearly decided there. Once you have isomorphisms, you can't really say that one structure is *the* structure for a given class. Groups, for example. So isn't modern math essentially pluralistic? Well let's look at foundations - set theory doesn't seem to be very pluralistic; a set is a set is a set, and not something else. You can come up with distinct set *theories*, but they all depend on the primitive notion of set, or maybe set membership. Type theory, by contrast, seems to be different. It doesn't have this kind of unity. In fact there are many distinct type theories, so we should probably always use the plural. The primitive seems to be "type"; but the concept of type is not primitive in all type theories—HoTT being a case in point. "In fact, no type former is 'primitive' to the game of type theory in this sense: you can very well have a type theory with no type formers! But it won't be very interesting..." (M. Shulman,)

"Type theory, formal or informal, is a collection of rules for manipulating types and their elements. But when writing mathematics informally in natural language, we generally use familiar words, particularly logical connectives such as "and" and "or", and logical quantifiers such as "for all" and "there exists". In contrast to set theory, type theory offers us more than one way to regard these English phrases as operations on types. This potential ambiguity needs to be resolved, by setting out local or global conventions, by introducing new annotations to informal mathematics, or both."the HoTT Book , p. 101

Normative Pragmatics

Chapter 1 of [1]

Inferential Semantics

Chapter 2 of [2]

Expressivism

See [3]

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

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

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

2
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

Ed. note 2.0.1 From schema to type. E.g. $A, B \vdash A \wedge B$ — traditionally viewed as a schema (involving either substitution or denotation), no construction involved. Move from this to viewing it as a rule of construction or recipe for making something.

3
Proposition

Ed. note 3.0.2 BHK interpretation. How Martin-Löf got it wrong wrt classic interpretation.

4

Proof: Truth and Consequences

Ed. note 4.0.3 Why Truth is not Important in Computation. Proof is a mode of expression. The statement of a proposition–e.g. $1 > 0$–expresses the propostion but leaves its inferential articulation implicit. A proof of the proposition explicitly articulates its upstream inferential structure. To say that a proof demonstrates the truth of the proposition it proves is just to say that it makes (part of) its inferential articulation explicit—in whatever proof formalism you decide to use (more than one possibility). Truth plays no substantial role in this view; it's just a convenient way to say "explicit inferential articulation".

Truth

Why Truth is Not Important in Type Theory (with apologies to R. Brandom[1])

Consequence as prior to truth — [2]

Proof before truth [3]

Ed. note 4.1.1 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.

[1] 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

[2] 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

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

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*.[4]

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

Ed. note 4.1.2 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.

Ed. note 4.1.3 But proof as explicit articulation of inferential structure also comes out as a thing. It must, if we are to have proof objects as mathematical objects; it is not enough just to do a proof, we must produce a proof object—which is just the explicit statement of the inferential structure implicit in the mere statement of a proposition. So if it is something we do it is also something we produce.

etc.

Critical point: in HoTTwe have two "kinds" of types: propositional types and non-propositional types.[5] If we are to also treat "proof" (or witness or whatever) as a fundamental principle of HoTT, one 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 4.1.4 General point (to be made elsewhere, maybe in Chapter 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.

[4] 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.

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

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*[6] 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.[7]

[6] Close, but no cookie. The notion of demonstrative expression must be made concrete, in terms of explicit articulation of inferential structure of proposition.

[7] Hmm. Proof of the type $\mathbb{N}$ by producing a natural number — how does this count as explicit articulation of inferential structure of $\mathbb{N}$? What is that structure, anyway? The inductive definition? In constructing a natural number, e.g. 3, we *use* of the principle of induction, of Succ. Does this count as making the inferential articulation of $\mathbb{N}$ explicit? That would imply that the explicit articulation of a proof must include the entire apparatus, including inference rules, type formation rules, etc. But this would mean that definitions (type formation rules) form part of the proof, which seems circular - you cannot "prove" $\mathbb{N}$ by citing the very definition of $\mathbb{N}$. And while we're at it: what are the type formation rules for propositions like $1 > 0$?

Ed. note 4.1.5 Can we think of each natural number as a dependent type? I.e. something like Succ parameterized by – what?

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).

Proof

Ed. note 4.2.1 Concept of "proof obligation"

Ed. note 4.2.2 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 4.2.3 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 4.2.4 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.[8] 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.[9] 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

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

[9] 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 4.2.5 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.

Logical Consequence

10

11

12

13

14

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

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

[12] 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

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

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

Inference and Deduction

Gentzen

etc

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.

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.

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,[15] 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.[16]

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.[17]

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[18] 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.

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

[16] 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.

[17] See §9.7 for more on the inferential articulation of conceptual content.

[18] 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 4.6.1 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 4.6.2 "Demonstrator" as the genus of "demonstration" and "demonstrative". It has the virtue of paralleling "constructor".

Semantics

4.7.1 Meaning

4.7.2 Model-theoretic Semantics

4.7.3 Proof-theoretic Semantics

"Proof-theoretic semantics is an alternative to truth-condition semantics. It is based on the fundamental assumption that the central notion in terms of which meanings are assigned to certain expressions of our language, in particular to logical constants, is that of proof rather than truth. In this sense proof-theoretic semantics is semantics in terms of proof . Proof-theoretic semantics also means the semantics of proofs, i.e., the semantics of entities which describe how we arrive at certain assertions given certain assumptions. Both aspects of proof-theoretic semantics can be intertwined, i.e. the semantics of proofs is itself often given in terms of proofs."[19]

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

4.7.4 Inferential Semantics

5
Mathematics

Traditional

Modern: classic

Modern: Intuitionism

Ed. note 5.3.1 Why Brouwer should be deemed a pragmatist.

Mathematical Pragmatism

Ed. note 5.4.1 HoTTis (largely) founded on Martin-Löf's account of "judgment" (assertion). I don't know if that's entirely accurate, but it's my story and I'm sticking with it for now. (Martin-Löf was quite specific that his project was motivated by "purely philosophical" considerations. See his 1972 paper.) Brandom's account of assertion is part of a larger, very ambitious project that aims to explain the structure of rationality. It's a thoroughly pragmatic account; everything comes down in the end to "proprieties of practice": conceptual activity (thinking and talking) is explained in terms not of what we know but of what we do (or what we know *how* to do).

Brandom's account of assertion is much more refined and sophisticated than Martin-Löf's. If we replace Martin-Löf's account with Brandom's, then HoTTcomes out as a piece of "mathematical pragmatism" (or pragmatic mathematics): an account mathematics grounded in practice.

Ed. note 5.4.2 TODO: Brandom's philosophy, like most of contemporary pragmatism, subverts the dominant representationalist mode of thinking. It turns things upside-down, or inside-out. So it is with type theory. (In one of his papers Martin-Löf suggests something similar, pointing out that his take on judgment etc. reverts (in some sense) back to practices that preceded the ways of thinking that have dominated modern "classic" mathematics and logic.) The to-do item here is to show how the relation of type theoretic to classic thinking in mathematics and logic parallels the relation between pragmatist (anti-representational, expressivist) thinking and representational (cartesian, platonistic) thinking in philosophy, about rationalism, conceptual content, etc. Show how type-theoretic thinking turns traditional classic thinking inside-out.

6
Type Theory: Foundations

HoTT purports to offer a new foundational concept for mathematics. If we take assertion to be the foundational concept of type theory (I'm not sure this works, but it seems plausible), then Brandom's account of assertion can link type theory to a foundational account of discursive practice (rationality).

Today set theory is the reigning foundational theory of mathematics. It's fairly easy to present it as such: first you list the axioms, then you show how to "construct" the natural numbers from sets, using a successor function. Or you might follow the lead of the Z specification notation[1], and proceed from sets to relations and then to functions. However you do it, it's all pretty intuitive and relatively easy to explain, even to mathophobes.

What would such a foundational presentation look like for HoTT? If HoTT turns out to be a genuinely foundational theory, then it must be grounded in intuition; specifically, we should expect that its basic notions correspond in some way to some collection of pre-theoretic mathematical intuitions, just as the axioms of set theory do, or as the axioms of geometry match our ordinary intuitions about the organization of space as we experience it pre-theoretically.

Presentations of set theory usually begin by discussing the axioms; but even though axioms serve as "unexplained explainers", such a presentation inevitably depends on a yet more primitive layer of concepts. Specifically, not only the (pre-theoretical) concepts of set, subset, and membership, but also axiom and perhaps proof. All of these "preliminary" concepts—let's call them "principles"—correspond more or less directly to intuitions available to any concept-user.

In general, an explicit account of the fundamental *principles* of set theory is either omitted or informally glossed, before the presentation moves on to the axioms. But type theory, in the end, is radically different from set theory at a very fundamental level, as far as I can see. "Set" and "type" are so easily grasped that it is easy think of them as more-or-less the same sort of thing; but if you look hard at them,

[1] 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

they are very different, even fundamentally different. So I think a presentation of HoTT would be well served by beginning with an explicit account of principles, even before moving on to consider primitives of the theory.

Ed. note 6.0.3 Todo: add somewhere in here an explanation of what we mean by "primitive". The basic idea is conceptual independence. A primitive is like an axiom in the traditional sense: an unexplained explainer. If it depends (conceptually) on some other concept, then it is not primitive.

For example: on Brandom's account, representation is not primitive, since it depends on other things. Nor is conceptual content, for that matter, since it is instituted by proprieties of practice, which are primitive.

Another way to look at it: if you can get along without it—discard it and still count as playing the same game—then it is not primitive. So for example dependent types are not primitive: we can play the type-theory game without them.

On the other hand, I'm trying to distinguish between conceptual primitives—what you have to be able to think in order to think type-theoretically—and the specific bits of a particular theory which are to count as primitive for that theory. In other words, we have the primitives of rationality, and we have the primitives of specific theories. HoTT counts as a specific theory; type theory in general, on the other hand, is closely tied (or at least tie-able) to at least one foundational account of rationality (Brandom's).

This might not be worth the trouble, but then again type-theory strikes me as sufficiently radical to merit this kind of close philosophical scrutiny. I think it's hard to overstate the extent to which a type-theoretic mode of thinking differs from classic (set theoretic, classic logic) modes of thinking, and the differences can be quite subtle. After all, its a pretty radical step to say (with Huw Price[Price, 2010]) that we should discard the concept of representation *in toto* since it has no useful theoretical work to do. And I think that type theory and HoTT amount to "natural" idioms for such pragmatist modes of thinking.

Finally: the pragmatic angle, where the ultimate primitive is just practice. From that perspective, we ought to be looking not so much at primitive concepts as primitive practices. Two obvious candidates are categorization (=conceptualization) and counting. Pre-linguistic creatures can do both (language is not a necessary condition for counting), and linguistic communities invent terminology that allows there members to say what they otherwise

would only be able to do—say *that* there are three lions down by the watering hole rather than pantomiming or pointing them out directly. This suggests that one way to present type theory is treat the type of natural numbers as, if not primitive, only one step up from primitive (type).

What are the pre-theoretical principles and primitives of HoTT? The obvious place to start is "type". The concept of "type" obviously emerges from ordinary experience; indeed, it is arguably more primitive than the concept of "set". Just look at the vast literature on the emergence of categorization in developmental and cognitive psychology; the ability to categorize is undoubtedly one of the most primitive human intellectual skills, if not the most primitive. It may even be a primitive animal capability–bees categorize flowers, and every member of sexually reproducing species categorizes possible mates.

What about "axiom"? At first glance it would seem that any foundational account of mathematics (or anything else for that matter) must rest on an axiomatic foundation. Which is just another way of saying that any explanation of anything must eventually bottom out on a bedrock of unexplained explainers. You can't explain everything without entering an infinite regression.

On the other hand, we can view axiomatic explanation as just one "style" of explanation, one of many. When you begin with axioms, you present them as unequivocally (and unquestionnably) true. But this is really a bit of salesmanship; sometimes axioms turn out not to be quite as axiomatic as they seem. Reconceptualizations happen, which may lead us to view axioms in a new light in which they do not look quite as certain. Then axiomatic explanations are still intelligible, but are no longer unquestionnable. The classic example of this sort of evolution is to be found in the history of geometry. Before the development of non-Euclidean geometries in the 19th century, the axioms of Euclidean geometry were not only unquestioned but unquestionable:[2] the idea that parallel lines could meet was not just wrong, but crazy. Today, using axioms to define a geometry is just a way of making clear the assumptions necessary to make the theory work. They no longer represent essential connections to externally available bits of certain knowledge.

[2] I suspect I'm overstating the case here. Mathematicians: is this true?

In other words, axioms are not a necessary condition of adequate explanation. So the question is whether or not the axiomatic style is most appropriate for a presentation of HoTT? On the one hand, it seems to me that it is not necessary; an adequate explanation of HoTT without axioms should be possible. For example, we can treat the concept

of "type" as primitive, even if we cannot find a good way to express it axiomatically.[3]

[3] This is a little fuzzy; maybe it doesn't even make sense. But as long as we're rethinking the foundations of mathematics, we might as well rethink everything.

Ed. note 6.0.4 the HoTT Book addresses this explicitly, p. 20-21. Roughly, it replaces the axioms of set theory with rules. But that's not entirely accurate: the rules of HoTT, after all, are themselves axiomatic. You really cannot get by without some sort of axiom; every journey must start from somewhere. So what is the difference between the axioms in set theory and those in type theory? I think it boils down to the difference between conditional and unconditional statements, but I'm not yet sure what the significance of this is.

Ed. note 6.0.5 Leaving full presentation of principles for later. I think it includes at least proposition and judgment, maybe inference and proof.

In any case, we'll have to begin somewhere, by stating some fundamental principles; then we'll need an account of the primitives, whether they take the form of axioms or not. What are the principles upon which HoTT depends? And once we have some principles (which are external to the theory proper), what are the primitives (which are "inside" the theory)?[4]

[4] Ok, "primitive" sounds a lot like "axiom". But I think there's a difference, even if I can't quite articulate it. Let's provisionally say that a primitive is an axiom without the concommitant commitment to unquestionned certainty.

Ed. note 6.0.6 "Type theory, formal or informal, is a collection of rules for manipulating types and their elements."the HoTT Book p. 101

Here are some possibilities, based on my understanding of the material in HoTTB. Please keep in mind this is coming from somebody who thinks he has a fairly good grasp of what type theory is all about, but is still grappling with HoTT.

Principles of Type Theory

Ed. note 6.1.1 The idea here is to discuss the "primitives" of type theory; that is, the concepts that are essential to any type theory.

Type Obviously a fundamental concept. What to say about it, though, is not so obvious.

Proposition

Judgment

Proof [5] Two kinds, corresponding to the two kinds of provables:[6]

Demonstration - *rational argument* that compels assent to a proposition[7]

Witness - evidence that bears witness to the existence of a kind or category

Inference

Induction Seems pretty basic to me, and as far as I know, nobody has ever been able to explain it in terms of more primitive notions. Without a Principle of Induction (of some kind), we would not be able to, for example, form the Natural Numbers in type theory.

etc.

[5] from Latin *probare* "to make good; esteem, represent as good; make credible, show, demonstrate; test, inspect; judge by trial" (source also of Spanish *probar*, Italian *probare*), from *probus* "worthy, good, upright, virtuous,"

[6] Remember, we're talking about pre-theoretical principles (concepts) here, not about HoTT per se.

[7] "Demonstration" is intuitively satisfying, but conceptually misleading, insofar as it suggests a visual metaphor. That would be classical; but for type theory we want metaphors of construction, not inspection.

The concept of proof in type theory deserves special attention. **??** examines it in detail; here, suffice it to say that it extends beyond the traditional and intuitive notion of proof as something one does to or with propositions. In type theory, proposition types represent propositions, so a type-theoretic proof of a proposition type—call it a "tt-proof"— corresponds to an ordinary proof of a proposition; it essentially involves inference, for example. But type theory also has lots of non-propositional types, like $\mathbb{N}$. These do *not* represent propositions: propositions have truth-values, natural numbers do not. In set theory, there is no connection between sets, elements, and proofs. An element either is, or is not, a member of a given set. Period, full stop. The notion of proof never enters the set-membership picture.[8] In particular, the existence of a set is not dependent on particular members, and the fact that some element is a member of some set has no significance with respect to the existence of the set. By contrast, in type theory construction of an element of a type counts as proof of the type. Etc.[9] But this kind of "proof" is not like proof of a proposition; it does not involve a proposition that may be true or false, and it does not involve inference. Instead it serves as a kind of evidence that shows the type.

[8] That need not mean that proving membership is never an issue. But you don't prove membership; rather, you prove that the element satisfies some predicate, which is a different concept.

[9] FIXME: fix this language.

Ed. note 6.1.2 Is there a significant distinction to be made between proof and witness? I suspect there is, based on the difference between propositions and names. Both count as evidence, but there is a difference between an inferential proof of a proposition and a "testimonial" witness to a kind. Propositions-as-types unifies the two ideas, but does not erase the distinction.

Type Formers

Ed. note 6.2.1 There's a lot to be said here. "Rules" is not quite right; construction rules, for example, are not just laws governing correctness, but "how-to" rules, productive procedures.

If such "rules" are primitives, then we have a very sharp contrast with classic mathematics and logic. The HoTTB doesn't dwell on the rules; it seems to treat them instrumentally, as devices we need in order to come up with new types and elements. But they are substantive; they define what it is to be a type or element.

Conceptually: can we have types without these rules? Intuitively, we obviously can: we don't need (explicit) rules in order to think of, say, 2 as a natural number. Categorization, apparently, does not rely on any notion of construction. This suggests that these rules are not primitive—we could discard them and still have types. But I think maybe that's the wrong way to think about things. It's too representationalist. On the inferentialist model, which is holistic, you cannot think "2" without also grasping its inferential structure. Can we treat this structure as constructive? I think so, but it will take some work to articulate just why and how. This will require considerable refinement of what is meant by "construction".

Type formation rules

Introduction rules i.e. construction rules

Elimination rules

Computation rules

Uniqueness principle

7
Types

the HoTT Book page 27 describes a "general pattern for introduction of a new kind of type". Martin-Löf does this too, somewhere. In the HoTT Book , the list is

Formation Rules

Introduction Rules or constructors

Elimination Rules or eliminators

Computation Rules "which express how an eliminator acts on a constructor"

Uniqueness Principle which "expresses uniqueness of maps into or out of that type. Optional.

The question is where to place this stuff in the description of HoTT. Are these things primitives? Do they form essential aspects of a type? Or in other words, can we have (think of) types without these rules?

the HoTT Book introduces them almost as an afterthought, as a Remark in the third major construction defined in Chapter 1. But I suspect this is a mistake or oversight; it looks to me like these rules are indeed fundamental, essential to the concept of type. In that case, they should be presented along with the introduction of the type concept, rather than in the middle of a description of a particular type.

Terms

Ed. note 7.1.1 "Terms" is Awodey's terminology. More common terminology include: witness; inhabitant. Also proof.

"Under the Curry-Howard cor- respondence, one identifies types with propositions, and terms with proofs..."[1]

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

Witness

Ed. note 7.2.1 In what sense is a proof a witness to a type, or an "inhabitant" of a type? Intuitively this language does not work very well; we don't intuitively think of a proposition as a type "inhabited" by proofs. The notion of proof as "witness" to a type is a substantive epistemological notion; it not only says that the proof is related to the type, but also it says something about the nature of that relationship.

The trick is to see it from the perspective of the machine. A proposition like $1 + 1 = 2$ is just a form to the machine. We can see that it is true just by looking, due to some mysterious epistemic capability. But machines do not have epistemic abilities; a form is a form is a form to a machine. Hammer, nail. So in order for the machine to treat $1 + 1 = 2$ as a *true* proposition, we have to give it something more: a proof. But "proof", again, is an substantive epistemic notion; the machine analog must be purely formal. From the machine perspective, a proof is just another form, or rather, collection of forms (including inference rules as complex forms): to give the machine a proof of P we must provide it with a form or forms that "lead to" (produce, result in) P. To prove a proposition to a machine, we give it forms and reduction rules such that the formal use of those forms and rules results in the form of the proposition to be proved. (FIXME: a more accurately way of putting this would involve reduction of formulae to normal form, confluence, etc.)

So we can think of a proof as a kind of device—just another machine (or machine description), but one whose sole output is the proposition to be proved. Since for any given proposition there may be many ways of building such a proving device, we can treat these devices as forming a kind of equivalence class, which we can identify by taking (the form of) the proposition as a symbol referring to the class. Now the connection to types and witness becomes clear: the equivalence class of such proving devices forms a type, the type of the devices (proofs), and each device (proof) "inhabits" (or as we would prefer, expresses) the type.

8
Curry-Howard

Ed. note 8.0.2 Usually presented as "propositions-as-types", but this suggests an asymmetrical relationship; in fact the principle is that propositions *are* types, and vice-versa. This is a major move in type theory, introduced by Martin-Löf(?) based on work by Curry and Howard. TODO: what exactly are the implications of this principle?

Ed. note 8.0.3 The critical point is that we go minimalist: start with the minimal logical language, which means combinatory logic. It is the isomorphism between the logical constants and the combinators (Curry) that motivates Curry-Howard. Once you see the connection at this minimalist level, it is easy to see it at any level, since the logical constants are the basic building blocks from which all propositions are constructed.

Ed. note 8.0.4 Start with Schoenfinkel and Curry, and the goal of finding the absolute minimum, which means eliminating variables. Then the basic combinatorys, then the isomorphism to the logical constants.

Equivalence of combinatory logics (no vars) and lambda calculus (vars)

Analogies. Proof/proposition, term/type: "There is also a one-to-one correspondence between proofs of a certain proposition in constructive

predicate logic and terms of the corresponding type." (Dependent Types at Work)

Assertion and Judgment

8.1.1 notes

This section needs some serious revision. Here's the straight dope, in a nutshell. In his paper "Truth of a proposition, evidence of a judgement, validity of a proof"[**?**], which is specifically about the philosophical basis of Intuitionisti Type Theory, Martin-Löf attempts to explain the concepts proposition, truth, evidence, proof, and validity. The first part of the paper, which gives some historical and conceptual background, is just right for the most part. He points out, for example, that for the intuitionist proof comes before truth. But he makes a major blunder when he claims that the classic truth-conditional account of the logical connectives, an account that is based on truth table semantics, and the BHK accounts, which treat a proposition as an expectation or task etc., are just different ways of saying the same thing. "Façons de parler", as the saying goes. But I think this is flat out wrong. Classic and intutionistic logic may use the same formulas, but they could not be more different conceptually. Classic truth-conditional logic presupposes something like what Wittgenstein called (in his Tractatus days, at least) a picture theory of meaning. (I may not be getting the exact wording right here, but the idea should be clear enough.) Proof in that kind of logic has nothing to do with construction; it's all about correspondence, a representational relation between language and objective reality. Obviously there's much more to be said about this, but suffice it to say that Martin-Löf's claim that classic logic and intuitionistic BHK logic are in the same line of business strikes me as not only wrong but a little bit shocking. So wrong that I have to wonder why he made that sort of claim. Maybe he was unfamiliar with the pragmatist literature. And we'll just proceed on the assumption that I am not wrong, if you don't mind. I'll provide a more detailed justification of this claim later.

Another thing that looks wrong to me is his account of the BHK interpretation of propositions; but in this case, he has an excuse: R. Brandom's more refined account of assertion and proposition was not yet available. Brandom's account makes the problems with Martin-Löf's account quite clear. The latter follows BHK in treating a proposition as a problem to be solved, a task to be accomplished, or an expectation of a proof, etc. The problem is that propositions are clearly exactly *not* these things. Obviously we may *treat* a proposition as e.g. a task to be accomplished; but that does not determine what a proposition *is*. Or to put it differently Martin-Löf seems to ignore the significance of *force*, which is distinct from conceptual content.

For now I don't have time to explicate the point in detail, so here's the short version: Brandom divides assertion into commitment and

entitlement. And what makes the proposition primitive is that it is the minimal unit of *responsibiity* - Brandom traces this notion to Kant. To assert a proposition is to undertake a commitment to it, and also to license others to challenge ones entitlement to that commitment. Thus it inescapably involves a kind of responsibiity: the responsibility to justify ("vindicate", as Brandom says) one's commitment. One way to do this is to demonstrate the entitlement by giving *reasons* for it.

If you're familiar with the Martin-Löf paper mentioned above, the connection should be fairly obvious. Commitment and entitlement are deontic attitudes, which institute deontic statuses (e.g. being correct or incorrect). They are emphatically *not* properties of propositions. So it is just a mistake to think that propositions are or express tasks, problems, or expectations. On the other hand, *assertion* of a proposition does give rise to a responsibility to vindicate commitment. Talk of "expectation of a proof" is entirely intelligible as a way of saying that assertion licenses others to challenge one's commitment—to expect that one can or will prove it. Talk of task or problem is really a way of talking of the justification or vindication that one is responsible for.[1]

[1] And note that this is not a mere matter of voluntary acceptance of responsibility; it arises because assertion licenses others to *hold* one respondible, treat one *as* responsible, and therefore sanction speakers who fail to vindicate their commitments.

So in the end, Martin-Löf is speaking more or less the right vocabulary, but his explanation is off, and his characterization of propositions as involving something in addition to propositional content (e.g. expectation, task, etc.) is not defensible, at least from a Brandomian perspective. HoTT, unfortunately, duplicates his error in its account of judgment.

The remedy is close at hand, though. All we need do is recognize that what the HoTT Book calls "judgments", like "a : A" and "a := b", are really *stipulations*, and what it calls propositions, *assertions*. A stipulation, unlike an assertion, does not require justification. A stipulator does not license listeners to demand reasons for the stipulation. Of course they can make such demands, but almost by definition we can stipulate *ad libitum*. Listeners who don't like our stipulations can go elsewhere; but no *rational* challenge can be mounted against a stipulation. The reason it makes no sense to ask for a proof of "a : A" is not because it is a "judgment" but because it is a stipulation and therefore needs no justification. By contrast, "propositional equality" is just a fancy (and rather unfortunate) term for "asserted equality".

Another way to look at it: HoTT, following Martin-Löf, attributes special properties to propositions. But that's hard to defend, philosophically, and doesn't amount to genuine explanation; a better way is to explain assertion in terms of deontic attitudes and responsibilities.

8.1.2 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 8.2.1 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"[2]

"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)."[3]

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

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

8.2.1 Substitution

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

Ed. note 8.2.2 Compare substitution in lambda calculus, and in Brandom's model. Maybe something about combinatory logic and the elimination of variables?

9
Misc

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.

Brandom on Assertion

Remark 5 Relevance to type theory: type theory begins with an account of judgment, proposition, etc. Robert Brandom offers a very sophisticated account of these concepts which IMO could be put to very good use in explaining the conceptual foundations of type theory.

Brandom's "deontic scorekeeping model of discursive practice" is a very sophisticated and ambitious philosophical project. But the main point of interest for us, his treatment of assertion, is relatively easy to grasp.

First off, for Brandom logic is fundamentally *expressive*, rather than epistemological. "Logic is for establishing the truth of cerain kinds of claims, by *proving* them. But logic can also be thought of in expressive terms, as a distinctive set of tools for *saying* something that cannot otherwise be made explicit".[Brandom, 2001, 19] One of his favorite examples is the inference from "Pittsburgh is west of Princeton" to "Princeton is east of Pittburgh". We can endorse that inference as a good material inference - material because it follows from the meanings of the terms the sentences contain - even if our language does not contain a conditional connstruction like "if...then". But once we extend our language by adding such a device, we can make that endorsement explicit by saying "If Pittsburgh is west of Princeton, then Princeton is east of Pittsburgh". So "if...then" is an expressive device, rather than an epistemological one.

Brandom's model of assertion involves both a social aspect and a structure of commitment and entitlement. The basic metaphor is that in the game of giving and asking for reasons, interlocutors maintain a deontic scorecard for each other and for themselves, tracking commitments and entitlements.

"According to the model, to treat a performance as an assertion is to treat it as the undertaking or acknowledgment of a certain kind of *commitment*—what will be called a 'doxastic' or 'assertional' commitment. To be doxastically commited is to have a certain social status. Doxastic commitments are normative, more specifically *deontic* statuses. Such statuses are creatures of the practical attitudes of the members of a linguistic community-they are instituted by practices governing the taking and treating of individuals *as* committed. Doxasitc commitments are essentially a kind of deontic status for which the question of *entitlement* can arise. Their inferential articulation, in virtue of which they deserve to be understood as propsitionally contentful, consists in consequential relations among the particular doxastic commitments and entitlements—the ways in which one claim can commit or entitle one to others (for which it accordingaly can serve as a reason)." [Brandom, 1998a, 142]

"Uttering a sentence with assertional force or significance is putting it forward *as* a potential reason. Asserting is givein reasons....The function of assertion is making sentences available for use as premises in inferences." [Brandom, 1998a, 168]

"The basic model of inferential practices that institute assertional significance...is defined by a structure that must be understood in terms of the interaction of three different dimensions. First, there are two different sorts of deontic status involved: *commitments*, and *entitlements* to commitments...The second dimension ... turns on the distinction between the *concomitant* and the *communicative* inheritance of deontic statuses. This is the *social* difference between *intra*personal and *inter*personal uses of a claim as a premise...The third dimension of broadly inferential articulation that is crucial to understanding assertional practice is that in which discursive *authority* is linked to and dependent upon a corresponding *responsibility*.... In asserting a claim, one not only authorizes further assertions (for oneself and for others), but undertakes a responsibility, for one commits oneself to being able to vindicate the original claim by showing that one is entitled to make it."[Brandom, 1998a, 168-171]

"At the core of assertional practice lie three fundamental ways in which one can demonstrate one's entitlement to a clam and thereby fulfill the responsibility associated with making that claim... First... one can demonstrate one's entitlement to a claim by *justifying* it, that is, by giving reasons for it. Giving reasons for a claim always consists in making more claims: asserting premises from which the original claim follows as a conclusion... The second way of vindicating a commitment by demonstrating entitlement to it is to appeal to the authority of another asserter. The *communcative* function of assertions is to license others who hear the claim to reassert it. The significance of this license is that it makes available to those who rely on it and rassert the original claim a special way of ischargin thheir responsibiity to demonstrate their entitlement to it."[Brandom, 1998a, 174; the third way involves invoking one's own authority as a reliable noninferential reporter, which is discussed later in MIE.]

9.7.1 Propositional Content

Note that Brandom's notion of what it is to understand a proposition or proof looks very different from Martin-Löf's. But they converge on the essential point, which involves grasping the inferential relations among concepts and reasons. For ML, understanding a proposition means grasping what counts as a proof (or something like that); for Brandom, it involves grasping the "inferential articulation" of the concept - the network of propositions and inferences relating them that

consitutes the concept itself. This is more or less just like ML's idea: to understand a proposition is to grasp what counts as a reason for the proposition, or—what is the same thing—entitlement to commitment to the proposition.

9.7.2 Applying Brandom's Model

Let's look at what mathematical assertions and judgments look like from a Brandomian perspective.

To start: we have a propositional content, which we can write as "unasserted P". We have commitment, entitlement, and justification (proof).

Uttering—or, usually, writing down—a proposition P makes explicit one's commitment to the content of P, and makes one liable to demonstrate entitlement to that commitment. Hearing—reading—a proposition P entitles one (by "deontic inheritance") to undertake a commitment to P if one is willing to ascribe entitlement to the utterer/author. Otherwise, it authorizes one to demand a reason. One can also record (on one's "deontic scorecard") the speaker's commitment to P while declining to undertake the commitment oneself.

TODO MLTT analyzes the structure of (mathematical) assertion interms of proposition, judgment, truth, etc. Map this structure to Brandom's structure. Brandom's account should turn out better since it is more finely articulated, and distinguishes explicitly between commitment and entitlement.

9.7.3 Understanding Propositions as Types

There is a natural bridge from Brandom's account of assertion (of propositions) to the type-theoretic notion of what it means to understand a type.

In HoTT, to understand a type is to understand what counts as a witness to or proof of that type. Here "witness" is surely the better word, since witnesses are always required to appear in court and offer testimony in person. "Proof" is too broad a term, since (ordinarily) it encompasses classical non-constructive proof as well as constructive proof. A non-constructive proof does not produce a witness; it can only offer hearsay testimony, so to speak. Whatever witnesses a non-constructive proof claim to have cannot be sworn in and so cannot be interrogated, whereas a witness in the dock can be directly challenged. We can *decide* whether or not to accept the testimony of the witness—decide that the offered proof has a flaw.

[On the other hand, the judicial metaphor breaks down at the crucial point: type theoretic witnesses do not really testify on behalf of the proposition; they *produce* the proposition. So maybe it

would be better to drop "witness" in favor of some other term, like "maker". (Actually we already have "constructor".) Ironically, the notion of "truth-maker" plays the central role in classic truth-conditional logic: the truth-conditions of a proposition are just those conditions that "make" it true. Here, instead of truth-makers, we have type- or proposition-constructors; instead of truth-conditional logic, we have proof-conditional logic.]

For Brandom, understanding a proposition is to be construed in inferential terms already. The propositional content of an assertion just *is* its inferential articulation: the network of inferences in which it plays the role of premise or conclusion. So to understand that propositional content is to grasp those inferences, both upstream—inferences in which the proposition serves as conclusion—and downstream—inferences in which it plays the role of premise.

A caveat is in order here. When Brandom talks about propositions and inferences, he is not referring to the machinery of formal logic. His topic is ordinary discourse; for him, formal logic emerges as a set of expressive devices whose functioning is parasitic on the normative structure of discourse practices. Nonetheless, the structure of his philosophical account of discourse practices serves the needs of HoTT just about perfectly.

But an inference of which P is the conclusion is a *reason* for P; it serves as justification that entitles one to a commitment to P. It follows that understanding a proposition means grasping what counts as its justification. And this is just what we need for HoTT. The only difference is that HoTT imposes an additional constraint: the only acceptable justifications are constructive, ones that yield the proposition not only as the conclusion of an inference but as the product of a procedure. In other words, constructive inferences.

Now: how do we get from this notion of what it takes to understand a proposition to the notion of propositions-as-types? First, notice that Brandom's model imposes no restrictions on the *kind* of inferences involved in articulating propositional content. That's because his is the philosophical project of understanding language use in general.[1] HoTT narrows this down by excluding from consideration all but constructive inferences; in effect, it can be viewed as an application of Brandom's model to a restricted subset of the universe of discursive practices. But in both cases, there is no antecedently determinable restriction on the *number* of distinct inferences involved in the articulation of the conceptual content of a proposition.[2] In the case of ordinary language, different people may have wildly different reasons for undertaking a commitment to a particular proposition. In mathematics, as is well know, there many be many distinct proofs for a given theorem. This fundamental *plurality* is what underwrites the notion of propositions-as-types. Since a proposition P may play the role of conclusion in any number of distinct valid inferences, we can generalize and say that P "represents" all those inferences; or we can say, equivalently, that all those inferences are the same *kind* of inference, the kind whose conclusion is P. Calling P the *type* of those inferences, and in turn calling the inferences *witnesses* to (or proofs of) P is then just a linguistic move—there is nothing special about the word "type", and any similar word

[1] On his account, assertion is fundamental; no practice can count as discursive practice unless it involves assertion. His approach stands in contrast to Wittgensteinian approaches that treat language as a motley, with no "downtown", no one fundamental kind of speech act. On the other hand, for mathematicians, logicians, and others working in similarly regimented idioms, assertion is usually not only the fundamental speech act, it is the *only* speech act (within the idiom, that is).

[2] It follows that the meaning of a mathematical statement like $1 + 1 = 2$ is nothing more nor less than its proofs, which although distinct can be treated as equivalent—in the idiom of mathematics, equivalent "up to" their common conclusion(up to iso-conclusionism?). This gives us a single, univocal meaning for the proposition in spite of proof plurality. Which is desireable, since we do not want to say that $1 + 1 = 2$ has a variety of meanings. Unity in diversity. TODO: treat this separately.

would have done as well. The critical point is the relation between the proposition and its proofs.

To recapitulate: to say that a proposition is the type of its proofs is just to say that those proofs form the category (or equivalence class) of all proofs whose conclusion is the proposition; to use the proposition as the designated symbol representing that class or category; and to call the whole mess a "type". By extension, we can use the text of the proposition as the name of the type. So for example the proposition $1 > 0$ is the (symbol of the) type of all proofs whose conclusion is the proposition $1 > 0$; or, the name "$1 > 0$" is the name of that type.

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[3]. 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.

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

9.8.1 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?

10
The Language of HoTT

Ed. note 10.0.1 TODO: disentangle commonly used terminology: parameterized types, generic types, algebraic types, generalized algebraic types, types indexed/parameterized, polymorphism, parametric polymorphism, polytypism etc. etc.

Ed. note 10.0.2 Type formers as entries in a HoTT lexicon, serving not as definitions but as (normative) rules of usage. They don't say what the terms mean, they set out how to use them. That means, specifically, the rules governing the notation, not rules governing denoted entities.

NB: rules of use *of vocabulary* $\neq$ rules of construction *of objects*. But the idea is for one set of rules to work both ways. That's pretty much how model-theoretic semantics connects vocab to semantic domain (completeness and consistency). The difference is that the TT semantic domain here (i.e. objects and their rules of construction) is not a passive, platonistic realm of "real" objects, but a pragmatic "field" of action. So rules of vocab use and rules of construction converge while remaining conceptually distinct. IOW the difference is not metaphysical.

On this view we treat HoTT as truly a vocabulary rather than a theory about something. Or more precisely as a regimented idiom or dialect. The user is free to treat e.g. $\mathbb{N}$ as "defining" a true model of the natural numbers, but HoTT makes no such claim.

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

11
HoTT Types

HoTT primitives are ....[1]

Ed. note 11.0.3 The concept of primitives probably isn't going to work for type theories proper. Type theories seem to be inherently pluralistic, so there is no way to pick out some things as intrinsically primitive in all type theories. Each theory might *define* a set of primitives, but that would define conventions of the theory, not the notion of primitive that we're after here. So if we want to talk about primitives it will have to be as above, involving principles antecedent to any type-theoretic talk. (?)

[1] A proper exposition would list 1) the name of the primitive, e.g. "Dependent Product-type"; 2) the "constructor" symbol, e.g. Π; 3) the analogous concept from set theory; 4) the "rules" for defining a type (formation, construction, elimination, computation, uniqueness).

Simple Types

Ed. note 11.1.1 E.g. $\mathbb{N}$

"Proposition type" Conceptually, at least, this seems primitive. Especially if the concept of "proposition" counts as a pre-theoretic principle. Which implies that proof is also a pre-theoretic principle. Proposition types are fundamentally different than the other kinds of type, since they have truth-conditions.[2] Etc. It follows that proofs are fundamentally different from other kinds of witness.

Ed. note 11.1.2 But Curry-Howard means there is no distinction between types and propositions, so it makes no sense to try to demarcate a "proposition type". E.g. the type $\mathbb{N}$ can be viewed as the

[2] Actually this isn't quite right. Propositions have truth-conditions in classic logic, but not not in type theory. In type theory they have proofs; a propositional type is not true or false, but proven or disproven. But the larger point stands: the *concept* of proposition is different than the concept of, say, natural number.

type of "there exists a natural number". This feature demarcates type theory; in classic logic and math, and esp. traditional logic, there is a fundamental difference between propositions and the terms from which they are constructed. Not so in type theory.

Compound Types

Function "Unlike in set theory, functions are not defined as functional relations; rather they are a primitive concept in type theory."[3] [Univalent Foundations Program, 2013, p. 21]

Product Product types correspond to cartesian products in set theory. The constructor symbol is the same as in set theory: $\times$.[4] "[U]nlike in set theory, where we define ordered pairs to be particular sets and then collect them all together into the cartesian product, in type theory, ordered pairs are a primitive concept, as are functions."[Univalent Foundations Program, 2013, p. 26]

Coproduct type Coproduct types correspond to disjoint unions in set theory. "In type theory, as was the case with functions and products, the coproduct must be a fundamental construction, since there is no previously defined notion of "union of types".[Univalent Foundations Program, 2013, p. 33]

[3] Or: set theory *defines* a function as a set of ordered pairs whose domain has no duplicates; in other words, it treats a function and its "graph" as the same thing. Question: what happens to the graph of a function in type theory?

[4] Why isn't this called the "$\times$-type"?

Ed. note 11.2.1 (Updated) [Update: M. Shulman pulled the scales from my eyes: "From the point of view taken in the book, the difference is only one of perspective, and any type can represent a proposition by simply shifting our perspective on it. For instance, the type Nat represents the proposition "there exists a natural number"."] What I've called "proposition type" is not *formally* distinct; the distinction I'm after is conceptual. But this suggests that we need to add at least one more item to our list of primitives: "ordinary" or simple types. Maybe it would be best to start with the natural numbers as an example of a simple type, rather than function types. Then an example of a proposition type, before proceeding to function type. The items listed (following the the HoTT Book) are really constructions, or let's say complex types, built out of two other types. So maybe the distinction we want is between simple and complex or compound types. Then the simple types would come out as primitive, and the complex types as derived (just like dependent types.) Compare the idea of constructions in

category theory. There categories are primitive and e.g. the product category is an example of a category constructed from other categories.

Ed. note 11.2.2 For consistency, we might want to use symbols to designate all of the primitives, just as we do for Π and Σ. This would give us: $\rightarrow$-types, $\times$-types, and $+$-types.

Dependent Types

Ed. note 11.3.1 Add sth re: types indexed over n v. parameterized over α

Ed. note 11.3.2 Major TODO: "basic" types (as in Z) plus constraints (predicates) v. complex dependent types.

Example: the paradigmatic example for dependent types is a list. This combines a type parameter α and an index n: List $\alpha, n : \mathbb{N}$ means a list of length n of values of type α.

But that's only one possibility. Let's add a real number: List $\alpha, n : \mathbb{N}, x : \mathbb{R}$. In this case x could mean anything: sum of the elements of the list, product, mean, standard deviation, etc. In short x can be any *statistic* computed over the list.

This works intuitively; does it work in type theory? Specifically, Martin-Löf-style theories like HoTT? Would these be genuine types or types plus additional constraints? In other words, is the interpretation of x built-in to the type, or is it an "extra" constraint applied after the fact, as it were.

"Fundamental"[5](but non-primitive) concepts and types. These types seem to be on a par with the primitive types as far as importance goes, but they presuppose the primitives, so cannot themselves be considered primitive.

[5]

Ed. note 11.3.3 Obviously we need a better bit of terminology. "Quasi-primitives"? "Neo-primitives"? These types are not primitive, strictly speaking, but on the other hand they are basic. I think there is another fundamental principle at work here. In set theory, for example, the concept of function is not only not primitive, it isn't necessary. You could discard it and still have set theory. But my intuition tells me that e.g. the concept Π-type is in a sense necessary or essential in type theory,

Ed. note 11.3.4 We can make a distinction between the concept of dependent type, and the two specific dependent types introduced here. Neither is primitive; you can have a type theory without the concept of dependent types. Most programming languages fit this description, whether they have an explicit type discipline or not.

Ed. note 11.3.5 Regarding the notion "quasi-primitive": not a very satisfactory term, but I can't come up with a better one at the moment. What I'd like to show is that the concept of dependent type (maybe also type universe) follows "naturally" or necessarily or essentially from the more primitive concepts. Maybe the right concept here is "induction": the primitive concepts (types) "induce" the concept of dependent type. That would be nice esp. if induction is a primary principle.

Universe Is this a primitive? Probably not, since it builds on the type concept.

Π-*type* Informally, "dependent function" types.[6] The concept of Π-type is a generalization of the concept of function type, so it isn't primitive.

Σ-*type* Informally, "dependent pair" type.[7] The concept of Σ-types is a generalization of the concept of product type.

[6] As a practical matter, I think it would be useful to have an informal term for these types that falls between "dependent function type" and Π-type. Something like "p-function type". Π-type is admirably concise, but I think it should mention "function", since it names a kind of function.

[7] Shouldn't this be called "dependent *product*" type? The type is product, not pair; pairs are "elements" of the type. Informally, maybe "sig-prod type?

Standard Type Library

Ed. note 11.4.1 By analogy to the usual "standard library" of programming languages. The idea is to list commonly used types that are neither primitive nor quasi-primitive; "application" types, in a sense.

Boolean [Univalent Foundations Program, 2013, p. 34]

$\mathbb{N}$ [Univalent Foundations Program, 2013, p. 36] But there's a problem here; actually several. First of all, the section on the naturals in

chapter one does not actually show how to construct them; its really a section about induction, not the natural numbers. Second, what it does say about the naturals is that they start from zero. That obviously won't do; zero is not a natural number, and there is no intuitive notion of zero as a number. You can't even think of it as a number until you've severed the link between the concept of number and the concepts of quantity and/or magnitude. So the natural numbers really must start with 1, not 0.

Proposition Moved to Primitives section.

Identity

Misc. Niceties

Ed. note 11.5.1 tait: "objects are given or constructed as object of a given type". The expression "a : A" expresses the idea that we are given a of type A. It does this by stipulation rather than assertion. Assertions are challengable and must be justified on demand; stipulations are not and need not.

ML Type Theory is centered (more or less) on one of the major logico-philosophical topics of the 20th century, namely the nature of assertion and its relation to propositions and inferences.

You don't have to understand the arcana of this debate in order to understand type theory (or HoTT), but some familiarity with the main outline is very helpful. Actually, I think it's essential, if you want to understand the HoTT Book's account of *judgement*, presented in HoTT Chapter 1 (reproduced below). Fortunately the presentation is relatively straightforward.

Remark 7 Stress: this is largely a philosophical issue, or perhaps an issue in Philosophy of Language. It's really about how our utterances come to have the significances they do.

Outline:

- Frege's elevation of *force* as essential
- Dealing with embedded (and therefore forceless) propositions
- Wittgenstein
- Dummett
- etc.
- Brandom's recent innovation: decompose "assertion" into "commitment" and "entitlement"

What the HoTT Book refers to as judgment (following ML) could also be called assertion. Brandom's account of the "fine structure" of assertion is very helpful here. Among other things, it provides a very simple explanation of how embedded propositions work. Embedded propositions are unasserted; the problem is how to reconcile this with the fact that they are function as assertions if unembedded. On Brandom's account, [todo...]

In other words, we can have commitment with or without entitlement, and vice-versa.

A set membership statement can be explained in terms of commitments and entitlements. A free occurance of e.g. $a \in A$ is ordinarily taken as an assertion (judgment). We can follow Frege and make this *force* explicit: $\vdash a \in A$. The problem with this, however, is that, in contemporary usage, this would make $a \in A$ *logically* true, which is not what we want. Instead we want a representation of committment to the proposition, as at least ordinarily true, without regard to its logical truth.

TODO: logical v. ordinary truth is pretty hairy for non-logicians so the distinction should be explicated.

In sum: the implicity sense of $a \in A$ is something like:

$$\exists\, \Gamma, a, A \mid \Gamma \vdash a \in A$$

Informally: there exists a set of propositions Γ, a value (or object) a, and a set A such that the propostion $a \in A$ is deducible from Γ.

So the meaning of $a \in A$ essentially involves existential quantification. It is a statement about the world, that it contains the relavant entities, not about the entities themselves.

Remark 8 Not quite; $a \in A$ is surely a statement about a, maybe also about A, no? But still there must be an implicity existential quantification over the propositions that entail the statement.

There is a logical subtlety here. $a \in A$ seems to be about a determinate a and a determinate A, but it isn't, not if we take it to be an existentially quantified statement. That's because $\exists\, a, A \mid a \in A$ does not pick out determinate individuals; it just says that *some* such individuals exist in the domain of interpretation. True, a and A are said to be bound by $\exists$, but that's not entirely accurate; quantified variables are not bound in the way that constant symbols like π or 0 are bound. Whatever we go on to say about a and A – e.g. $a \in A$ – remains within the scope of the quantifier, so it does not count as a statement about determinate individuals. It's a statement about the world, that it contains entities that satisfy the predicate.

On the other hand, the same seems to be true of $a : A$: though these symbols be bound, we don't know what they are bound to. They are not bound by an implicit existential quantifier; $a : A$ does *not* mean $\exists\, a, A \mid a : A$.

Remark 9 Plus, quantifiers have to be used with a predicate; strictly speaking, $\exists\, a, A$ is not a complete statement.

By contrast, the Type Theoretic analogue $a : A$ is a statement about a specific value and a specific type, without any quantification. It is not directly a statement about the world, but about part of the world. Or: it expresses both commitment and entitlement. That's why it cannot be embedded in e.g. "if $a : A$ then it is not the case that $b : B$". Embedded propositions cannot carry force, but $a : A$ always carries force, intrinsically, as it were.

11.5.1 $a : A$

Forms go from symbols to terms to sentences; from a to $a + b$ to $a + b = c$.

The "judgment" $a : A$ is clearly a compound term, so it cannot merely name something. But is it a sentence? Does it denote a proposition? Or is it analogous to terms like $a + b$ which are names of a sort but involve some additional meaning beyond mere reference.

It seems it must involve a proposition, or let's say propositional content. We take $a : A$ as a statement of fact, rather than a mere reference to some part of the world. Then how is it distinct from $a \in A$?

The HoTT Book says it is "analogous" to the set-theoretic statement $a \in A$, but essentially different, since $a \in A$ is a proposition but $a : A$ is a judgment. It says that, *when working internally in type theory*, $a : A$ cannot be embedded, as in " if $a : A$ then it is not the case that $b : B$", nor can the judgment $a : A$ be disproved.

So let's look closely at what this means. Earlier, HoTT says that (some) judgments involving A "exist at a different level from the *proposition* A itself, which is an internal statement of the theory." (p. 18) There's a bit of circularity there; what is an "internal statement"?

TODO The nature of "proposition" has been a topic of considerable debate. Review some of the alternative accounts on offer.

The basic idea seems to be based on the well-known concept that propositions by themselves are devoid of force, and must be asserted. HoTT seems to imply that judgments are asserted propositions – or more correctly, assertings of propositions.

This seems a little bit off. Assertion is something only people do. An inked form on a page cannot really be construed as an assertion. So we need to work out the mechanics of how a written form like $a : A$ can be viewed as a "judgment" in this sense. I think Brandom's model of assertion works. It would say, I think, that $a : A$ counts as a judgment (assertion) because by convention we agree to treat it that way, whereas we treat $a \in A$ slightly differently, because of the conventions elaborated by 20th century logic.

When HoTTB refers to "working internally in type theory", it seems, the idea is to consider propositions in isolation from their assertion. Assertion, on this view, is something that comes from outside of the world of propositions. This is perfectly in tune with the idea that asserting is something people do, but that what gets assert*ed* – the *content* of an assertion – is distinct from the assert*ing*.

Remark 10 Sellars called this the notorious -ing/-ed distinction.

This would seem to make $a : A$ an assert*ing*.

We can think of $(a : A)$ as a *given* proposition: one that, while

unasserted, has the same force as a propositional assertion. Or another way to put it would be to say that use of $(a : A)$ is inalienably performative.

In fact $(a : A)$ corresponds nicely to a common linguistic practice, namely combining a proper name and a description, as in "Joan of Arc", "King George", or "Slick Willy". Or, more colloquially, "poor Tom", "angry Joe", or "Gimpel the fool". And the primitive nature of types can be clearly illustrated by analogy with the military. In type theory, every object has a type, just as everybody in the military has a rank. You cannot be in the military unless you have a rank. Within the military, the proper way refer to someone in the miliary is to combine rank and name: General Custer, Sergeant York, Private Bilko. So the difference between $(a : A)$ and $a \in A$ is like the difference between "This is General Custer" and "This is Custer; he is a General".

On the other hand, "This is General Custer" doesn't look much like a *judgment*, although it does look like a *claim*. But not that it is not a claim about the meaning of "General Custer"; rather it is a claim about the relation between "This" and "General Custer". You could be wrong about the name or the rank of whomever you mean by "This", but you cannot be wrong about "General Custer"; that's just a qualified name. Being wrong in this sense about "This is General Custer" is an empirical matter; in type theory, the question of whether $(a : A)$("this is a-of-A") is correct or not never even arises. It doesn't make an empirical assertion, it states a *given*. Or we might say it gives a fact. By contrast, $a \in A$, as a proposition, may be either true or false; when we say "let $a \in A$", we implicitly stipulate that $a \in A$ is to be *assumed* to be true, but it is not *given* as true. In other words, we can gloss it as "$a \in A$ has a truth-value like any proposition, so it could be false, but please assume that it is true."

Another critical distinction: in standard set theory and logic, judgments come from the outside, as it were. But in HoTT, judgments of the form $(a : A)$are internal. They may be derivable inside the system (by production of a proof or witness.) In other words, inference in set theory comes from outside of the world of sets, but inference in HoTT is built in to the structure of types. Inference (construction) is part of the intrinsic meaning of types.

11.5.2 Justification

The HoTT Book's account of judgments in Chapter 1 section 1 seems to conflate the distinction[8] Brandom makes between commitment and entitlement. "Informally, a deductive system is a collection of rules for deriving things called judgments."

But derivation (proof) starts and ends in propositions; commitment

[8] This is only to be expected, since Brandom is the first (so far as I know) to see that assertion (judgment) has an internal structure involving commitment and entitlement (and some other stuff like a social dimension.)

is something else. The derivation or proof provide warrant for entitlement to the commitment - justification of the conclusion. So how would Brandom parse "judgment" as HoTT uses the term?

A

HoTT Book Excerpts

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.1.1 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.1.2 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.1.3 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.1.4 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 clas-

sical 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.1.5 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.1.6 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.1.7 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.[1] 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:

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

[1] 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.

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 A.1.

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

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

A.1.8 *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.1.9 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 $\pi_1(\mathrm{S}^1) = \mathbb{Z}$ 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.

C
Bibliography

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. *Expressiwism, 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.