15-819 Homotopy Type Theory Lecture Notes

Nathan Fulton

October 9 and 11, 2013

1 Contents

These notes summarize and extend two lectures from Bob Harper's Homotopy Type Theory course. The cumulative hierarchy of type universes, Extensional Type theory, the ∞-groupoid structure of types and iterated identity types are presented.

2 Motivation and Overview

Recall from previous lectures the definitions of functionality and transport. Functionality states that functions preserve identity; that is, domain elements equal in their type map to equal elements in the codomain. Transportation states the same for type families. Traditionally, this means that if $a =_A a'$, then $B[a]$ true iff $B[a']$ true. In proof-relevant mathematics, this logical equivalence is generalized to a statement about identity in the family: if $a =_A a'$, then $B[a] =_B B[a']$.

Transportation can be thought of in terms of functional extensionality. Unfortunately, extensionality fails in ITT. One way to recover extensionality, which comports with traditional mathematics, is to reduce all identity to reflexivity. This approach, called Extensional Type theory (ETT), provides a natural setting for set-level mathematics.

The HoTT perspective on ETT is that the path structure of types need not be limited to that of strict sets. The richer path structure of an ∞-groupoid is induced by the induction principle for identity types. Finding a type-theoretic description of this behavior (that is, introduction, elimination and computation rules which comport with Gentzen's Inversion Principle) is an open problem.

3 The Cumulative Hierarchy of Universes

In previous formulations of ITT, we used the judgement $A\,\mathsf{type}$ when forming types. In this setting, many types are natural to write down but impossible to form. As a running example for the section, consider the following: $\mathsf{if}\, M, \overline{17}, \mathsf{tt} : \mathsf{if}\, M, \mathsf{Nat}, \mathsf{Bool}$. Assuming the well-formedness of the type, elimination rules behave as expected: $\overline{17} : \mathsf{if}\,\mathsf{tt}, \mathsf{Nat}, \mathsf{Bool} \equiv \mathsf{Nat}$ and $\mathsf{tt} : \mathsf{if}\,\mathsf{ff}, \mathsf{Nat}, \mathsf{Bool} \equiv \mathsf{Bool}$.

Forming this type is not possible using the current formation rule for $\overline{\mathsf{if}}$. Type universes address this shortcoming by generalizing type formation rules. A recursively generated cumulative hierarchy of universes ($\mathcal{U}_i$) is introduced. Instead of defining type formation in terms of a judgement $A\,\mathsf{type}$, formation rules state the relative location of relevant types in the hierarchy; that is, judgements that $A\,\mathsf{type}$ are replaced with judgements of the form $A : \mathcal{U}_i$.

The definition of type universes includes three new rules[1]

$$\frac{\Gamma\,\mathsf{ctx}}{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\ \mathcal{U}\text{-intro} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_{i+1}}\ \mathcal{U}\text{-cumul} \qquad \frac{A \equiv B : \mathcal{U}_i}{A \equiv B : \mathcal{U}_{i+1}}\ \mathcal{U}\text{-}\equiv$$

The $\mathcal{U}$-intro rule introduces an unbounded hierarchy of universes, each of which inhabits the next universe. The second rule states that these universes are cumulative, and the third ensures that equality is preserved in higher universes. The $\mathcal{U}$-$\equiv$ rule is a derived rule in the HoTT book presentation.

In addition to these rules, every type formation rule establishes relative positions of relevant types. For example[2]:

$$\frac{A : \mathcal{U}_i \quad M, N : A}{Id_A M, N : \mathcal{U}_i}\ \mathcal{U}\text{Id-F} \qquad \frac{A : \mathcal{U}_i \quad x : A \vdash B : \mathcal{U}_i}{\Pi_{x:A} B : \mathcal{U}_i}\ \mathcal{U}\Pi\text{-F}$$

$$\frac{\Gamma\,\mathsf{ctx}}{1 : \mathcal{U}_i}\ \mathcal{U}1\text{-F} \qquad \frac{\Gamma\,\mathsf{ctx}}{0 : \mathcal{U}}\ \mathcal{U}0\text{-F} \qquad \frac{A : \mathcal{U} \quad B : \mathcal{U}}{A + B : \mathcal{U}}\ \mathcal{U} + \text{-F}$$

The addition of universes to ITT solves the problem identified by the running example. As the example suggests, these hierarchies increase the expressiveness of ITT. This is established by identifying a statement that cannot be proven only in the presence of universes.

[1] See A1.1 and A2.3 of the HOTT book for discussion.

[2] See appendix 2 of [2] for a full formulation

example: Jan Smith established that without universes, it is not provable that $\mathsf{succ}(-) \neq 0$ [5]. However, in Martin-Löf's Type Theory, it is provable that $n : \mathsf{Nat}$, $\mathsf{succ}(n) \not\equiv 0$[3]. In fact, Smith proved that *any* negation of an equivalence cannot be proven without universes.

exercise: Show that the operators fst and snd can be defined from split.

3.1 Typical Ambiguity

In the examples above, subscripts on each universe create significant notational overhead. Therefore, these indices are elided whenever intent is obvious. When implemented with pen and paper, this is called **typical ambiguity**. Its mechanization in Coq is referred to as **universe polymorphism**.

3.2 Alternatives to the Hierarchy

The introduction of an infinite hierarchy of universes complicates the theory. An uninitiated reader might wonder whether an infinite, cumulative hierarchy is really necessary. This section presents three alternatives. The first alternative works, but has some disadvantages. The other two alternatives have significant problems, demonstrating that the complexity induced by type universes is essential to a consistent and sufficiently expressive definition of ITT.

3.2.1 Large Elimination

Intensional Type Theory can be consistently formulated without a hierarchy. The approach, called Large Elimination, rules the correct types into the theory by hand. In the case of the running example, the rule would be:

$$\frac{M : Bool \quad A\ \mathsf{type} \quad B\ \mathsf{type}}{\mathsf{if}\ M, A, B\ \mathsf{type}}\ \text{LE-If}$$

Similar rules must be provided for each type formation rule. The universal approach is preferred because it is less ad hoc —large elimination requires the addition of new rules for each affected type.

[3] See page 86 of Programming in Martin-Löf's Type Theory [1].

3.2.2 A Single Universe

The running example may be addressed without introducing a recursively defined hierarchy of universes. One alternative is to replace the $\mathcal{U}$ rules above with a single universe. In this case, the important choice is whether $\mathcal{U} : \mathcal{U}$.

If the universe is not self-inclusive, The formulation problem discussed above re-emerges. For instance, the type $\mathsf{if}\, M, \mathcal{U}, \mathcal{U} \to \mathcal{U}$ is not formable without a recursively defined hierarchy. The same observation applies at the top of any finite hierarchy.

3.2.3 The Inconsistent Approach

An insightful reader might observe that this problem can be resolved by patching the single universe system with a rule which allows the universe to contain itself:

$$\frac{}{\Gamma \vdash \mathcal{U} : \mathcal{U}}\ \mathcal{U}\text{-inconsistent}$$

This system allows the formation of $\mathsf{if}\, M, \mathcal{U}, \mathcal{U} \to \mathcal{U}$. However, it also destroys the consistency of the theory.

Exercise: Reproduce the Burali-Forte Paradox within a system equipped with U-cumul-inconsistent[4].

4 Proof-relevance and Extensionality

Martin-Löf's Type Theory is significant because it introduces the notion of *proof relevance*. Intuitively, this expresses the idea that proofs can be treated as mathematical objects.

4.1 The Theorem of Choice

It is well-known that the Axiom of Choice is independent of the axioms of set theory. However, choice can be derived in ITT. The derivation provides an excellent example of proof relevance in action.

The theorem of choice states that if xCy is total, then there must exist a function (f) which associates each x with a chosen $y = f(x)$. We can state this formally in ITT.

Theorem of Choice. $\vdash e : \Pi_{x:A}\Sigma_{y:B}C(x,y) \to \Sigma_{f:A\to B}.\Pi_{x:A}C(x,f(x))$.

[4]This formulation of the paradox is due to Girard 1972, and is referred to as Girard's Paradox.

The proof, provided in [4], involves finding a derivation of:

$$F : \Pi_{x:A}\Sigma_{y:B}C(x,y) \vdash \lambda F. < \lambda x.\mathsf{fst}F(x), \lambda x.\mathsf{snd}F(x) >: \Sigma_{f:A\to B}.\Pi_{x:A}C(x,f(x))$$

In the proof, F is both an assumption and a mathematical object (namely, a product). Therefore, the proof may rely upon not only the inhabitation of the type of F, but also F itself.

Note: In future lectures, banana brackets will be used to recover a more traditional reading of F by suppressing the ability to use it as a piece of data in the proof. For now, the significant observation is that proof irrelevance can be recovered within ITT.

4.2 Failure of Extensionality

Marin-Löf demonstrated that the Axiom of Extensionality fails in ITT; in ITT, it is not the case that if $p : Id_A(M,N)$ for closed M, N, A, then $M \equiv N : A$.

Extensional Type theory (ETT) addresses the failure of extensionality in ITT by endowing the theory with the principle of equality of reflection. Concretely, ETT introduces two new rules which reduce identity to equivalence. Therefore, all identity paths on a type are the reflexive path.

$$\frac{\Gamma \vdash p : Id_A(M,N)}{\Gamma \vdash M \equiv N : A}\ \text{Eq-Refl} \qquad \frac{p : Id_A(M,N)}{\Gamma \vdash p \equiv \mathsf{refl}(M) : Id_A(M,M)}\ \text{UIP}$$

The first rule, equality of reflection, states that proof of an identification is sufficient to show judgemental equality in the type. The second rule, Uniqueness of Identity Proofs, states that any path is the reflexive path.

Although extensionality does not hold generally for ITT, uniqueness of identity proofs may be recovered for a large class of types.

Hedberg's Theorem. *Any set with decidable identities has collapsed identity sets [3][5].*

4.2.1 ETT vs ITT

The essential difference between ETT and ITT is the algebraic structure of types. ETT reduces all identity paths to reflexivity. As a result, the path structure of types

[5]There is a Coq proof by Nicolai Kraus online: http://www.cs.nott.ac.uk/~ngk/hedberg_direct.v

in ETT is homotopically discrete. ITT admits a much richer path structure on types: two paths $p : Id_A(A, B)$ and $q : Id_A(B, C)$ may by equal but not trivially equal.

The two other major differences between ETT and ITT are decidability of type checking and fitness for set-level mathematics.

The UIP rules introduces proof search as a valid mode of operation for the type checker. Therefore, type checking is not decidable in ETT[6]. Decidability is not an important criterion for two reasons. First, the standard mode of operation in a mechanized ETT (e.g. NuPRL) does not result in proof search. Second, type checking in ITT quickly becomes intractable.

A more important secondary distinction between ETT and ITT is fitness for set- level mathematics. Types in ETT have the structure of an h-set; therefore, set-level mathematics is much nicer in NuPRL than in Coq. Whereas extensionality and transport come for free in NuPRL, Coq users must induce this structure by programming in terms of a setoid. However, the convienance of ETT comes at a cost: the path structure of its types in necessarily limited due to Hedberg's Theroem.

Just as proof-relevant mathematics subsumes proof-irrelevant mathematics as a special case, the ∞-groupoid structure of types in ITT may be forgotten so that ETT is recovered as a special case. In fact, this is essentially what happens with Setoid in Coq.

5 Algebraic Structure of Identity Types

Recall that the induction principle for identity types states that for $x, y : A$, there exists an identity type $x =_A y$. Furthermore, proving a property for these elements and a path $p : x =_A y$ consists of proving the property in the reflexive case (that is, for x, x, refl_x).

The full implications of this principle were not understood when it was first introduced. A realization central to Homotopy Type Theory is that the induction principle for identity types gives rise to an entire hierarchy of iterated identity types. That is, due to J, we can form the type $p =_{\mathsf{Id}_A(x,y)} q$ and so on. Homotopy Type Theory is so-called, in part, because these types form the same structure as iterated homotopies: that of an $\infty-$groupoid.

Whereas the universes provide a mechanism for reasoning about size in an iterative fashion, the iterated identities provide an account of dimension. In the example above, x and y are start and end points. The first identity corresponds to a path between

[6]Type checking for ITT is decidable

the elements. Paths between $p, q : \mathsf{Id}_{\mathsf{Id}_A(x,y)}$ are homotopies, or 2-dimensional paths. Each iteration corresponds to an increase in dimension[7].

Before proceeding with a presentation of the groupoid axioms in terms of ITT, it is useful to recall the derivable equivalence relation:

(1) $id_A(M) := \mathsf{refl}_A(M) : Id_A(M, M)$
(2) $p : Id_A(M, N) \vdash p^{-1} : Id_A(N, M)$
(3) $p : Id_A(M, N), q : id_A(N, P) \vdash p \centerdot q : Id_A(M, P)$

The second and third are theorems provable by path induction, since composition and inversion are both defined in terms of J. Therefore, we may read identity types propositionally as witnesses of an equivalence, and computationally as abstract data types upon which we may operate. The two notations for identity types, $x =_A y$ and $\mathsf{Id}_{A(x,y)}$, typically elucidate the intended reading. While the former reading comports with more traditional interpretations of equality, the latter gives rise to the iterated identity types.

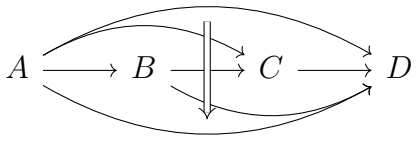
Remark. *Despite the correspondence with classical (analytic) homotopy theory, maps should be thought of synthetically. As a result, path concatenation is not defined in terms of function composition; hence, the* $\centerdot$ *notation.*

5.1 The Groupoid Laws

The groupoid laws may be forumated as coherence theorems, each proven by path induction using J:

inv-right	$p \centerdot p^{-1} =_{Id_A(M,M)} \mathsf{id}(M)$
inv-left	$p^{-1} \centerdot p =_{Id_A(N,N)} \mathsf{id}(N)$
unit-right	$p \centerdot id(N) =_{Id_A(M,N)} p$
unit-left	$id(M) \centerdot p =_{Id_A(M,N)} p$
assoc	$(p \centerdot q) \centerdot r =_{Id_A(M,P)} p \centerdot (q \centerdot r)$

Before proceeding an explanation of how these are proven, some motivation may be helpful. Consider the following diagram for the associativity theorem:



This diagram illuminates the weak nature of the path structure: associativity holds only because it holds at yet higher type. Iterated identity types are given structure by this higher coherence.

[7]Dimension is also referred to as homotopy level.

Theorem 1. *The groupoid laws hold.*

Full proofs are available in chapter 2 of [2]. We outline portions of the argument here for the sake of later discussion. For the first inverse theorem, perform path induction on p. It suffices to consider that $p = \mathsf{refl}_x$. We have by the definition of refl_x^{-1} that $\mathsf{refl}_x \centerdot \mathsf{refl}_x^{-1} = \mathsf{refl}_x$. The other inverse argument follows similarly. The cases for the unit theorems and associativity are similar. Each follows by path induction on p, considering the case where $p = \mathsf{refl}_x$.

5.2 Maps preserve structure

Given this structure, it is natural to ask whether mappings preserve the groupoid structure. Recall that ap preserves identities.

Theorem 2. *If* $f : A \to B$ *and* $p : M =_A M'$ *then* $\mathsf{ap}_f(p) : fM =_B fM'$.

Mappings preserve not just identity, but the entire groupoid structure. Proving this requires showing that *ap* preserves identity, inversion and composition. That is,

Theorem 3. *For a function* $f : A \to B$ *and paths* $p : x =_A y$, $q : y =_A z$

1) $\mathsf{ap}_f(\mathsf{refl}(x)) \equiv \mathsf{refl}(f(x))$

2) $\mathsf{ap}_f(p^{-1}) = ap_f(p)^{-1}$

3) $\mathsf{ap}_f(p \centerdot q) =_{\mathsf{Id}_a(x,z)} ap_f(p) \centerdot ap_f(q)$

Exercse: Prove that maps preserve functoriality. A formal proof will be included in a coming revision of these notes.

5.3 Does Homotopy Type Theory have a Computational Interpretation?

In the sketch of the groupoid proofs, the general case of p is reduced to the case of refl_x. Currently, this is justified by a categorical model. This is not natural or desirable because the distinguishing characteristic of type theory is its computational content characterized by Gentzen's Inversion Principle. The model-based justification is insufficient, in part, because it does not provide a way of *running* HoTT programs. The constructivity of Homotopy Type Theory is important because Hedberg's Theorem collapses the dimensional tower developed in this section. However, determining whether there is a computational interpretation of Homotopy Type theory is a principle open problem.

References

[1] Jan M. Smith Bengt Nodström, Kent Petersson. Programming in martin-löf's type theory. `http://www.cse.chalmers.se/research/group/logic/book/book.pdf`, 1990.

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

[3] Michael Hedberg. A coherence theorem for martin-löf's type theory. *J. Funct. Program.*, 8(4):413–436, July 1998.

[4] Per Martin-Löf. Intuitionistic type theory. `http://intuitionistic.files.wordpress.com/2010/07/martin-lof-tt.pdf`, 1980.

[5] Jan M. Smith. An interpretation of martin-lof's type theory in a type-free theory of propositions. *J. Symb. Log.*, 49(3):730–753, 1984.