HoTT lecture 9 Q&A (Solved)

HoTTEST Summer School 2022

The HoTTEST TAs

5 August 2022

Q1 Can someone give an example of 2 things that are logically equivalent that aren't genuinely equivalent?

Any two inhabited types are logically equivalent, e.g. Nat and Bool

For a general function f: 'f has a two-sided inverse' is logically equivalent, but not equivalent, to 'f is an equivalence'. The latter is a proposition and the former is not

Oh I see, I'm dumb. thank you.

Q2 To get around this predicament, couldn't we require for any $u : A \times B$, u is definitionally equal to $(\mathsf{pr}_1 u, \mathsf{pr}_2 u)$?

live answered

Q3 Why can't we turn the homotopy into an identification?

Why can't we turn the homotopy into an identification?

We'll get to it in just a second!

Because there are models where $f \sim g$ and $\neg(f = g)$ can hold simultaneously! But in HoTT, we can turn the homotopy into an identification (this is the topic of today's lecture)

Oh sorry I forgot to say thx. Thank you

Q4 I suppose this example is related to the Tensor Hom adjunction. In this case we can go from the left adjoint to the right adjoint but apparently we have trouble going the other way around (to establish a genuine equivalence). In general, is the problem going from the right adjoint to the left adjoint?

That's the idea!

Pitch for my SLTC project where I formalise the categorical semantics of function types in Agda: https://github.com/FrozenWinters/stlc

Thank you, Astra, that's good to know.

Q5 Thank you, Astra! So it is the issue of external relationships vs internal structure, I imagine.

That's exactly right! You can think of funext as saying that external behavior determines internal equality. This is actually related to the ext in extensionality

Thank you, Chris!

Q6 Since is-contr is also h-level -2, is there a similarly interesting weak function extensionality for other h-levels?

Yes, this result generalizes to all truncation levels.

That generalisation says that the truncation level of a Pi-type is the truncation level of the codomain

For higher values, it is probably not *equivalent* to funext, I would guess

well wfe for props is enough for contractibles too, I suppose. but for sets and up i'm inclined to agree

Nice; we got to it next!

Good foreshadowing by Emily :)

Q7 so the relationship between weak funext and funext is like the relationship between induction and strong induction?

That's right! See here, for instance: https://homotopytypetheory.org/2011/12/19/strong-funext-from-weak/

I don't really see an analogy here. Strong induction is that you can use arbitrary smaller elements in your inductive call. Other than the weak/strong naming there is not much in common.

I think the analogy is just that the "weak" version implies the "strong" version

Ah ok:)

Q8 What is known about the consistency of the funext axiom with other axioms?

Univalence implies funext.

Thank you, Perry! And other axioms? Is there ever any trouble with them?

Actually, yes! It's inconsistent with many axioms of a more 'computational' nature. For example, 'formal Church's thesis' says that for any function $\mathbb{N} \to \mathbb{N}$, there is a 'program' (we call it a realizer) that realizes it. You can kinda see what goes wrong: this would be able to tell e.g. $\lambda x.x$ and $\lambda x.x + 0$ apart. You could imagine an assignment of realizers that sidesteps this, though, so to see that it's actually inconsistent takes slightly more work.

Thank you, Amelia! That's very helpful to know.

Q9 Is the collection of k-types in a universe, itself a universe? (Maybe for k at least zero?)

Could you say what you mean by "collection" here? Usually you'd form this with a Sigma type.

I mean pretty informally. Like, is this mathematical collection closed under all the ways we ask universes to be closed (products, coproducts, function types, etc.)

In this sense, only for certain type formers. For example, it's not true of coproducts.

Bool is equivalent to the sum of 1+1, later being -2 type, former being a set

Can you be missing coproducts even for k at least zero?

It does hold for sets and above.

Sets and above are closed under Π, Σ, Id, and W

Q10 Is the lack of eta for product types an Agda thing or a type-theoretic thing?

It's a thing that depends on whether you define sigmas as inductive types or records!

In Agda, it depends on how you defined Σ; For type theorists doing type theorists on paper, it depends on a bunch of things like relative humidity, phase of the moon, hours since their last meal, etc

That's an Agda thing, right?

I meant 'doing type theory' :sob:

For records, the projections pr_i are primitive and you do have an eta-law, but for inductive types, you would generally not expect this because the projections are not primitive, but are rather definitions that you had to make by induction

I would think that the usual inductive definition of product types would yield an eta-principle, as the only terms are those created by the intro rule. Right?

Sometimes people don't like having eta for records because it can be inefficient: with eta, if you have a variable of a record type with 20 fields, then its normal form will be the 'eta-expanded' version with 20 fields, each projecting from the variable.

It's interesting and surprising that you can do sigma types in these two different ways. For most other notions, they clearly fall under having a mapping in or mapping out universal property, but not the other

'@Peter, this would only happen with the definition of sum types that Paige gave right at the start - with two eliminators as opposed to an induction principle

* I meant AND types

Q11 A natural transformation in this presentation is 'just' a function? Aren't NTs usually made of a bunch of things (action on objects, action on arrows, commutative law)—do we get all that for free from functions somehow?

Natural transformations are a family of arrows (no action on objects, those are given by the functor) together with naturality. But when we interpret types as groupoids, every function is *automatically* a functor, and every homotopy is *automatically* a natural transformation!

Q12 Is the Yoneda Lemma (for types) given here just the bijection or is it also the natural isomorphism? Does the equivalence simply correspond to a bijection?

The equivalence is automatically natural, as per everything in type theory being natural (approximately)

I say this a person that has spent several months working on coherence conditions...

Thank you, Astra. So this would be functorial in both a and B, yes?

Q13 So representable functors correspond to identity types? Are there further reaching consequences of that, maybe a way to visualize representable functors?

This isn't a full answer, but each value of a rep functor is an infinity-groupoid.