Agda lecture 9 Q&A (Solved)

HoTTEST Summer School 2022

The HoTTEST TAs

18th August 2022

Q1 hcomp in 3rd dimension outputs 3 faces or just the 1, top one?

You can specify as many sides as you want, actually!

The output is the top face

But always only 3 input faces suffice?

I think you need 5 faces?

Zero input vertical faces (apart from the base) also suffice, but the top that you produce will have a lot less constraints on it in that case

Will we see examples of that?

https://discord.com/channels/964553017212956692/965195997363961937/1006643419491344568

Q2 why do we pick $y \equiv z$ path as a base for hcomp in the double composition thing? Does it matter which one we pick?

live answered

Q3 New question for clarity: what is hcomp of 0 sides and some specific base?

live answered

https://discord.com/channels/964553017212956692/965195997363961937/1006643419491344568

This link doesn't work for me :(

Also:

– This normalises to 0
foo : $\mathbb{N}$
foo = **hcomp** $\{\varphi = i0\}$ $(\lambda i \to \lambda\{()\})$ 0

– This does not reduce
foo2 : $\mathbb{S}^1$
foo2 = **hcomp** $\{\varphi = i0\}$ $(\lambda i \to \lambda\{()\})$ **base**

Q4 Is comp on zero sides the same as transp?

live answered

Q5 What is comp? What is the difference of that to hcomp?

heterogeneous composition

We're way too sidetracked $> \_ >$

in as little words as possible: it's when the *type* of the box varies as you go from the base to the top

comp is basically a combination of hcomp and transp.

So in hcomp, the entire expression has to live in one type (unless there's an interval variable in the context, in which case it can differ across i-values). However, it cannot differ vertically

it's when the implementation is a nightmare and I'm trying to get rid of it

Who came up with using 'H' to distinguish Homogeneous from Heterogeneous? :P

'homogeneous'is a restrictive adjective as opposed to the general composition

Q6 Is Bool equivalent to Partial (φ and $\sim\varphi$) ? (loosely speaking)

**or

Yes, I think that you can prove that!

For a specific value of φ

Partial $(\varphi \vee \sim \varphi)$ of what? And they can't be equivalent: one is a type, the other is in SSet

Oh, also, for Bool, that SSet would naturally have four distinct terms

So, yeah, the answer is no

Although I think that you can define a hetrogeneous equivalence SSet between A : Type and B : SSet, in which case we would have $\Sigma\ I\ (\lambda i \to \text{Partial}(i \vee \sim i))\ X \simeq Bool \to X$

i think that should be a Π rather than a Σ

Yes; the LHS should be $(i : I) \to \text{Partial}(i \vee \sim i)\ X$

Q7 What is Agda doing to determine if phi is exhaustively covered by these equational patterns? (Or does it check that?)

What is Agda doing to determine if phi is exhaustively covered by these equational patterns? (Or does it check that?)

It's a bit complicated but the disjunction of all the patterns you've written has to be interprovable with φ

Q8 Can we define natural numbers are subtypes of integers using this definition?

This is only for subtypes determined by a proposition phi over the interval, not arbitrary propositions.

φ can only involve interval variables, not variables of other types, so no

Q9 Is this a good time to ask why we're using a different definition of equivalence (contractible fibers) for cubical agda than we did in the earlier part of the course (has section and retraction)?

live answered

Q10 Is Glue useful in another example that is not for univalence? Or after proving univalence, Glue is not more necessary?

live answered

Glue has two purposes: to make sure that Type is univalent, and also to define hcomp in Type.

(Update: not any more)

Q11 If Glue is a primitive only used to define ua, why is it interesting that we can define ua instead of postulating it?

If Glue is a primitive only used to define ua, why is it interesting that we can define ua instead of postulating it?

live answered

The interesting part is that we can add equations involving Glue, in particular, saying what transp in Glue does. This allows ua-beta to be definable as well.

Q12 How does ua compute? Is there a way to predict it without needing to know all the details of how Glue works internally.

live answered

Are there rules of thumb?

ua itself is just a type called "Glue ..." that is a normal form, it doesn't compute directly by itself (much like how Pi types just are what they are). But then there are some very complicated equations about hcomp/transp in it that are hard to summarize.

Where can I read about it?

It's discussed in Section 6.2 of https://drops.dagstuhl.de/opus/volltexte/2018/8475/pdf/LIPIcs-TYPES-2015-5.pdf but I'm not sure if that's the best source.

Thanks!

Q13 Why we need ua_β? I thought ua = univalence

That's the propositional computation law for ua

univalence says that ua is an equivalence

Usually "univalence" is that "A = B is equivalent to Equiv A B", and "ua" is only a map going from right to left. When you tack on ua-beta then together they imply the full statement.

Q14 isoToEquiv in our terms is just the proof that equivalences have contractible fibers?

yes

Yup

To be more precise, it's a proof that isomorphisms have contractible fibers.

Q15 Do we need S to be constrained to Types here? Doesn't this hold for all functions?

Do we need S to be constrained to Types here? Doesn't this hold for all functions?

I think that the rest of the discussion (e.g. this being an equivalence) depends on it being a type

Q16 What is this Agda command that you put a function and it also puts the holes?

C-c C-r i think

Refine: C-c C-r

Q17 How is "structure" a function Type → Type? If I have a structure of a group, there may be more than one way to put a structure of a group on a single type.

The function Type → Type in this case would be "the type of group structures on a type": so it'd be a big Σ of multiplication, unit, inverse, the laws, etc, all parametrised over the underlying type

So the type X in S X would be the "carrier" of the structure

If there were at most one way to put a group structure on a type, then group structure would be a function Type -¿ Prop :)

Q18 Is SIP only true under UA?

i think SIP applied to 'no structure' is just univalence?

ˆ nice

It's a fun exercise to show that SIP -¿ UA

At least in normal Agda, as in cubical UA is provable anyways...

Q19 What about in the other direction? If we defined addition on binary numbers (it's computationally faster I believe) and then transported it to N, would it be really faster?

This question is about computational complexity

live answered

We're discussing this now!

If your inputs are exponentially big (written in unary), then there's no way Agda will be able to magically add them in polynomial time.

I think it's basically converting both inputs unary numbers to binary, then adding in binary and converting back

The overhead of converting back and forth from unary is roughly the same as the cost of the addition, so we can't gain anything, but maybe the question could be relevant for a more complicated operation?

Except that Agda has more efficient implementation of built-in addition, so in fact using built-in Nat addition is the best possible option!

Q20 I know that asypmtotic complexity is hopeless, but if we benchmarked it just for time, it could be faster?

It was supposed to be a comment to the upper question, sorry

Q21 Isn't this a case where we need to know the computational behaviour of SIP in order to write this proof?

You can prove a coherence law about transporting tice, for example

I don’t think I understand

Q22 Anders, What font and theme are you using, looks nice!

live answered