MEETING NOTES

1. Dec 1 2017

1.1. **Chandrika's notes.**

1.1.1. *Rushed Introduction and summary.* Consider the following diagram, where A, B, C are sets, and D is the homotopy pushout of B and C along A.

$$\begin{tikzcd}
A \arrow[r, hook, "f"] \arrow[d, "g"] & B \arrow[d] \\
C \arrow[r] & D
\end{tikzcd}$$

We would like to show that D is a homotopy set as well using the encode-decode method. Therefore, we defined code : $D \to D \to$ Type so that code(x, y) is a proposition.

Amelia spoke of a picture to think of for D. This is my understanding of the picture. D is the following topological space ("the double mapping cylinder"). $(B \cup A \times [0, 1] \cup C)/\sim$, where $\sim$ is defined by:

- $(a, 1) \sim b \iff f(a) = b, \forall a \in A, b \in B$
- $(a, 0) \sim c \iff g(a) = c, \forall a \in A, c \in C$

We have attempted to define code(x,y) to be the homotopy structure of the paths from x to y in this double mapping cylinder.

In Utah, we sucessfully defined encode and began to define decode : code $\to \mathrm{Id}_D$. We defined decode(code(x, y)) for all $x, y \in (B \cup C) \subset D$.

1.1.2. *What's next?* It remains to define $\mathrm{ap}_{\mathrm{decode}}$. I think that these are the types of the various parts of $\mathrm{ap}_{\mathrm{decode}}$.

$$\mathrm{ap}_{\mathrm{decode}}\Big(\mathrm{ap}_{\mathrm{code}}(\mathrm{inl}b, \mathrm{glue}(a))\Big) : \Bigg(\mathrm{decode}\Big(\mathrm{code}\big(\mathrm{inl}b, \mathrm{inl}f(a)\big)\Big) = \mathrm{decode}\Big(\mathrm{code}\big(\mathrm{inl}b, \mathrm{inr}g(a)\big)\Big)\Bigg)$$

$$\mathrm{ap}_{\mathrm{decode}}\Big(\mathrm{ap}_{\mathrm{code}}(\mathrm{glue}(a), \mathrm{inl}b)\Big) : \Bigg(\mathrm{decode}\Big(\mathrm{code}\big(\mathrm{inl}f(a), \mathrm{inl}b\big)\Big) = \mathrm{decode}\Big(\mathrm{code}\big(\mathrm{inl}g(a), \mathrm{inl}b\big)\Big)\Bigg)$$

$$\mathrm{ap}_{\mathrm{decode}}\Big(\mathrm{ap}_{\mathrm{code}}(\mathrm{glue}(a), \mathrm{inr}c)\Big) : \Bigg(\mathrm{decode}\Big(\mathrm{code}\big(\mathrm{inl}f(a), \mathrm{inr}c\big)\Big) = \mathrm{decode}\Big(\mathrm{code}\big(\mathrm{inl}g(a), \mathrm{inr}c\big)\Big)\Bigg)$$

$$\mathrm{ap}_{\mathrm{decode}}\Big(\mathrm{ap}_{\mathrm{code}}(\mathrm{inr}c, \mathrm{glue}(a))\Big) : \Bigg(\mathrm{decode}\Big(\mathrm{code}\big(\mathrm{inr}c, \mathrm{inl}f(a)\big)\Big) = \mathrm{decode}\Big(\mathrm{code}\big(\mathrm{inr}c, \mathrm{inr}g(a)\big)\Big)\Bigg)$$

$$\mathrm{ap}_{\mathrm{decode}}\Big(\mathrm{ap}_{\mathrm{code}}(\mathrm{glue}(a), \mathrm{glue}(a))\Big) : \mathrm{isContr}\Bigg(\mathrm{decode}\Big(\mathrm{code}\big(\mathrm{inl}f(a'), \mathrm{inl}f(a)\big)\Big) = \mathrm{decode}\Big(\mathrm{code}\big(\mathrm{inr}g(a'), \mathrm{inr}g(a)\big)\Big)\Bigg)$$

2. Dec 8 2017

2.1. **Chandrika's notes.**

2.1.1. *Questions to think about and things to do for next time.*

(1) Internalize and share if possible.

(2) Do we have to deal with a push out when defining $\mathrm{ap}_{\mathrm{code}}(b, b)$? Understand why/why not.

- My memory is that while $\mathrm{code}(b, b)$ is a push out, this is not an issue because for $\mathrm{ap}_{\mathrm{code}}$ the relevant situation is when b is in the image of A.
- What happens to $\mathrm{code}(b, b)$ when we restrict to b being in the image of A?

(3) Why isn't defining decode as simple as defining encode? Why can't we use path induction like we did for encode?

- Refer to Michael Schulman's comment: "The point is that the domain of code is a *colimit* of infinity-groupoids, so to define code we have to give a coherently commuting cocone. The part that looks like "defining it on objects" is defining the *morphisms* in that cone, each of which is automatically an infinity-groupoid morphism. The part that looks like "defining ap_code" is showing that that cone commutes (up to coherent homotopy), which amounts to "defining ap_code" *only* on the "new" higher morphisms that are "glued it universally" by the colimit. It's a little hard to distinguish in this case because the infinity-groupoids going into the colimit are all discrete, so they don't have any higher morphisms themselves, but in general there could be some, and we wouldn't have to define ap on those; it would come for free once we defined the morphisms in the cone.

  For encode, we are using the universal property of equality types, which is essentially a colimit-like construction. But there are no homotopies being glued in, and hence no "ap" cases to do.

  For decode, we again use the universal property of the domain of code, so we have ap cases to deal with.

I hope this helps. You may also find it useful to look at chapters 6 and 8 of the HoTT Book."

References