15-819 Homotopy Type Theory
Lecture Notes

Matthew Maurer and Stefan Muller

March 17, 2018

1 Contents

2 Recap

Recall from last week the construction of the suspension of a type. Note that we are using different notation from the book, namely $\mathsf{Susp}(A)$ as opposed to $\sum A$. The suspension $\mathsf{Susp}(A)$ of A contains two 0-cells:

$$N : \mathsf{Susp}(A)$$

$$S : \mathsf{Susp}(A)$$

and paths $\mathsf{merid}(a)$ between N and S, where $a : A$.

$$\mathsf{merid} : \Pi x : A.N =_{\mathsf{Susp}(A)} S$$

$$x : \mathsf{Susp}(A) \vdash B(X) : \mathcal{U}$$

We also defined recursion and induction on $\mathsf{Susp}(A)$:

$$\frac{n : B \quad s : B \quad x : A \vdash m(x) : n =_B s}{z : \mathsf{Susp}(A) \vdash \mathsf{rec}[B](n; s; x.m) : B}$$

$$\frac{z : \mathsf{Susp}(A) \vdash B(z) : \mathcal{U} \quad n : B(N) \quad s : B(S) \quad x : A \vdash m(x) : n =^{z.B}_{\mathsf{merid}(x)} s}{z : \mathsf{Susp}(A) \vdash \mathsf{ind}[B](n; s; x.m) : B(z)}$$

3 Equivalence of $\mathsf{Susp}(2)$ and $\mathbb{S}^1$

Proposition 1.

$$\mathsf{Susp}(2) \simeq \mathbb{S}^1$$

Proof.

$$f : \mathsf{Susp}(2) \to \mathbb{S}^1$$

defined by

$$N \mapsto \mathsf{base}$$

$$S \mapsto \mathsf{base}$$

On the higher order part of the suspension, we define what the behavior of the one-cells should be, e.g. how ap would act.

$$\mathsf{merid}(tt) \mapsto \mathsf{loop}$$

$$\mathsf{merid}(ff) \mapsto \mathsf{refl}(\mathsf{base})$$

To show equivalence, we now define a quasinverse of f

$$g : \mathsf{Susp}(2) \to \mathbb{S}^1$$

where we send base is arbitrary but must be consistent

$$\mathsf{base} \mapsto N$$

$$\mathsf{loop} \mapsto \mathsf{merid}(tt) \cdot \mathsf{merid}(ff)^{-1}$$

Now need to prove these are inverses

$$\alpha : \Pi x : \mathsf{Susp}(2).g(f(x)) = x$$

By induction

$$\begin{array}{rl} (x = N) & \mathsf{refl}(N) : N = N \\ (x = S) & \mathsf{merid}(ff) : N = S \\ \mathsf{merid}(y) & ? : \mathsf{refl}(N) =^{z.f(g(z)=z}_{\mathsf{merid}(y)} \mathsf{merid}(ff) \end{array}$$

We can case out on this,

$$\mathsf{ap}_g(\mathsf{ap}_f(\mathsf{merid}(tt))^{-1} \cdot \mathsf{refl}(N) \cdot \mathsf{merid}(tt)) = \mathsf{merid}(ff)$$

$$\mathsf{ap}_g(\mathsf{ap}_f(\mathsf{merid}(ff))^{-1} \cdot \mathsf{refl}(N) \cdot \mathsf{merid}(ff)) = \mathsf{merid}(ff)$$

stepping evaluation,

$$\mathsf{ap}_g(\mathsf{loop})^{-1} \cdot \mathsf{merid}(tt) = \mathsf{merid}(ff)$$

$$(\mathsf{merid}(tt) \cdot \mathsf{merid}(ff)^{-1})^{-1} \cdot \mathsf{merid}(ff) = \mathsf{merid}(ff)^{-1-1} \cdot \mathsf{merid}(tt)^{-1} \cdot \mathsf{merid}(tt) = \mathsf{merid}(ff)$$

Our other proof of inversion

$$\beta : \Pi x : \mathbb{S}^1 . f(g(x)) = x$$

proceeds by induction on $\mathbb{S}^1$

$$\begin{array}{ll} (x = \mathsf{base}) & \mathsf{refl}(\mathsf{base}) : f(g(\mathsf{base})) = \mathsf{base} \\ (x = \mathsf{loop}) & \mathsf{ap}_f(\mathsf{ap}_g(\mathsf{loop}))^{-1} \cdot \mathsf{refl}(\mathsf{base}) = \mathsf{refl}(\mathsf{base}) \end{array}$$

□

4 Pointed Type

A pointed type is one with an example inhabitant. If A is a pointed type, then, in our notation, we have $a_0 : A$.

For example, $\Omega(A, a_0) = (a_0 =_A a_0)$, "the loop space", is pointed by $\mathsf{refl}(a_0)$. $\mathsf{Susp}(A)$ pointed by N (or S) is another example of a pointed type.

Pointed maps, also known as strict maps, are maps between two pointed types which map the well-known point of one to the well known point of the other, as in

$$(X, x_0) \multimap (Y, y_0) := \Sigma f : X \to Y . f(x_0) = y_0$$

We set out to prove

Proposition 2.

$$\mathsf{Susp}(A) \multimap B \simeq (A \multimap \Omega B)$$

Proof. Given $f : \mathsf{Susp}(A) \multimap B$, define $g : A \multimap \Omega B$ as

$$g(a) = p_0^{-1} \cdot \mathsf{ap}_f(\mathsf{merid}(a) \cdot \mathsf{merid}(a_0)^{-1}) \cdot p_0$$

where f_0 is the raw map, and p_0 is a proof of distinguished point preservation, e.g. $p_0 : f_0(N) = b_0$, and $f = \langle f_0, p_0 \rangle$

$$q = \mathsf{refl}(b_0)$$

On the other side, given $g : A \multimap \Omega B$, define $f : \mathsf{Susp}(A) \multimap B$

$$f_0(N) = b_0$$

$$f_0(S) = b_0$$

We again define the behavior of the one-cell in the HIT

$$\mathsf{ap}_f(\mathsf{merid}(a)) = g_0(a)$$

where g_0 is the raw map, and q_0 is a proof of distinguished point preservation, e.g. $q_0 : g_0(N) = b_0$ and $f = \langle g_0, q_0 \rangle$ □

5 Pushouts

We temporarily return to conventional set math to define a pushout, which is dual to a pullback, which is an equationally constrained subset of a product. On the other hand, a pushout is essentially a disjoint union of two sets with some of the elements "glued" together. Specifically, assume we have sets A, B and C with maps f and g from C to A and B respectively. The pushout $A \sqcup^C B$ is the disjoint union of A and B with the images $f(C)$ and $g(C)$ merged by merging $f(c)$ and $g(c)$ together for all $c \in C$. In the notation of type theory, we denote the disjoint union of A and B as $A + B$ and use the maps $\mathsf{inl} : A \to A + B$ and $\mathsf{inr} : B \to A + B$. We can then define the pushout as

$$A \sqcup^C B = (A + B)/R$$

where R is the least equivalence relation containing $\forall c \in C.R(\mathsf{inl}(f(c)), \mathsf{inr}(g(c)))$

$A \sqcup^C B$ is the "least such" object in the sense that it has a unique map to any other object D with the same properties:

$$\begin{tikzcd}
C \arrow[r, "g"] \arrow[d, "f"] & B \arrow[d, dashed, "f'"] \arrow[ddr, bend left, "f''"] & \\
A \arrow[r, dashed, "g'"] \arrow[drr, bend right, "g''"] & A \sqcup^C B \arrow[dr, dashed, "!"] & \\
 & & D
\end{tikzcd}$$

where f' and g' are identified by f and g.

5.1 Pushouts of $A \xleftarrow{f} C \xrightarrow{g} B$

Moving back to HoTT, we can define pushouts as a higher inductive type, whose elements are those of the disjoint sum $A + B$, generated by mapping A and B to the pushout $A \sqcup^C B$ by inl and inr, respectively

$$\mathsf{inl} : A \to A \sqcup^C B$$
$$\mathsf{inr} : B \to A \sqcup^C B$$

and whose 1-cells connect $\mathsf{inl}(f(c))$ and $\mathsf{inr}(g(c))$ for every element c of C.

$$\mathsf{glue} : \Pi c : C.\mathsf{Id}_{A \sqcup^C B}(\mathsf{inl}(f(c)), \mathsf{inr}(g(c)))$$

As usual, we can define a recursor $\mathsf{rec}[D](\cdots) : A \sqcup^C B \to D$ (the map implied by the diagram above.) Note that for every element u of C, the recursor requires that there exist a path witnessing the equality of $[f(u)/x]l$ and $[g(u)/y]r$, representing the action of the recursor on glue and ensuring that the images of elements "glued together" in the pushout are also glued together in D.

$$\frac{x : A \vdash l : D \quad y : B \vdash r : D \quad u : C \vdash q : [f(u)/x]l =_D [g(u)/y]r}{z : A \sqcup^C B \vdash \mathsf{rec}[D](x.l; y.r; u.q)(z) : D}$$

We have the usual β rules.

$$rec(x.l; y.r; u.q)(\mathsf{inl}(a)) \equiv [a/x]l$$
$$rec(x.l; y.r; u.q)(\mathsf{inr}(b)) \equiv [b/y]r$$
$$\mathsf{ap}_{rec(x.l;y.r;u.q)}(glue(c)) = [c/u]q$$

The idea of gluing together elements of a higher inductive type with a path for each element of a given type seems very reminiscent of suspensions. In fact, suspensions can be defined as pushouts $A \sqcup^C B$ with the types A and B and the maps f and g trivial:

$$\mathsf{Susp}(A) = 1 \sqcup^A 1$$

$$\begin{tikzcd}
 & A \arrow[dl, "!"'] \arrow[dr, "!"] & \\
1 \arrow[dr, "\mathsf{inl}"] & & 1 \arrow[dl, "\mathsf{inr}"] \\
 & \mathsf{Susp}(A) &
\end{tikzcd}$$

This immediately implies the following:

Corollary 1. *The pushout of two sets needn't be a set.*

This is true since $\mathbb{S}^1 = \mathsf{Susp}(2)$, but $\mathbb{S}^1$ is known not to be a set.

6 Quotients as HITs

The set definition of pushouts is in terms of a quotient, so we might consider defining quotients of sets by props uing HITs.

Consider the type-theoretic representation of A/R, a type A quotiented by the relation R. We define this type to have the normal properties of the quotient. We must be able to project an element of A into A/R, e.g. by computing its representative

$$q : A \to A/R$$

If we have two elements a and b of A that are related by R, that is, $R(a, b)$, then we must have a proof of equality

$$\mathsf{wd} : \Pi a, b : A.R(a, b) \to q(a) = q(b)$$

The above are the expected properties. However, we also wish for A/R to be a set. This requires that all proofs of equality in A/R must themselves be equal. We truncate A/R to a set by adding the required proofs of equality for all paths p, q between elements x and y.

$$\mathsf{trunc} : \Pi x, y : A/R.\Pi p, q : x =_{A/R} y.p = q$$

Saying there is a function from A/R to some type B is the same as saying there is a function from A to B that respects R by mapping related elements to elements that are (propositionally) equal in B.

Proposition 3.

$$(A/R) \to B \simeq \sum_{f:A\to B} \Pi_{a,b:A}.R(a,b) \to f(a) =_B f(b)$$

A proof appears in the textbook.

6.1 Example: $\mathbb{Z}$

We can represent integers as pairs of natural numbers (a, b) (representing the integer $a - b$). Since (infinitely) many pairs correspond to each integer, we quotient out by an appropriate relation.

$$\mathbb{Z} = (\mathbb{N} \times \mathbb{N})/R$$

where $R((a_1, b_1), (a_2, b_2))$ iff $a_1 + b_2 = a_2 + b_1$. This representation will be important in the proof that $\pi_1(\mathbb{S}^1) \simeq \mathbb{Z}$ in Section 8.

7 Truncations as HITs

We have previously seen the propositional truncation $||A||$, which truncates the type A to a proposition by forcing proofs of equality for all values. We now redefine the propositional truncation as a higher inductive type. Since we will generalize this to truncations for all h-levels n, we now write propositional truncation as $||A||_{-1}$.

The higher inductive type $||A||_{-1}$ has a simple constructor

$$|-|_{-1} : A \to ||A||_{-1}$$

To add the 1-cells, squash produces a proof of equality for any two values of A.

$$\mathsf{squash} : \Pi_{a,b:||A||_{-1}}.a =_{||A||_{-1}} b$$

The induction principle is as follows:

$$\frac{z : ||A||_{-1} \vdash P(z) : \mathcal{U} \qquad x : A \vdash p : P(|x|_{-1}) \qquad x, y : ||A||_{-1}, u : P(x), v : P(y) \vdash q : u =^{z.P}_{\mathsf{squash}(x,y)} v}{z : ||A||_{-1} \vdash \mathsf{ind}[z.P](x.p; x, y, u, v.q)(z) : P(z)}$$

We can now use the same methodology to define set truncation as a HIT:

$$|-|_0 : A \to ||A||_0$$

As in the definition of quotients, the set truncation provides proofs of equality for all paths.

$$x, y : ||A||_0, p, q : x =_{||A||_0} y \vdash \mathsf{squash}(x, y, p, q) : p =_{x=y} q$$

And we state an induction principle:

$$\frac{z : ||A||_0 \vdash P : U \qquad x : A \vdash g : P(|A|_0) \qquad x, y : ||A||_0, z : P(x), w : P(y), p, q : x = y, r : z =^{z.P}_{p} w, s : z =^{z.P}_{q} w \vdash ip : p =^{z.P}_{\mathsf{squash}(x,y,p,q)} q}{z : ||A||_0 \vdash \mathsf{ind}[z.P](x.g; x, y, z, w, p, q, r, s.ip)(z) : P(z)}$$

Proposition 4. *If A is a set, it is equivalent to its own set truncation, i.e.*

$$\mathsf{isSet}(A) \to ||A||_0 \simeq A$$

8 Fundamental group of $\mathbb{S}^1$

Recall $\mathbb{S}^1$ is a HIT defined by

$$\mathsf{base} : \mathbb{S}^1$$

$$\mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}$$

Recall $\Omega(A, a_0) := (a_0 =_A a_0)$

The fundamental group of A at a_0, $\pi_1(A, a_0)$, is defined to be $||\Omega(A, a_0)||_0$.

Theorem 1. *We want to show that*

$$\pi_1(\mathbb{S}^1) \simeq Z$$

Proof. Show $\Omega(\mathbb{S}^1, \mathsf{base}) \simeq \mathbb{Z}$ (this suffices by Proposition 4.)

We define a map

$$\mathsf{loop}^{(-)} : \mathbb{Z} \to \Omega(\mathbb{S}^1)$$

as follows:

$$\mathsf{loop}^{(0)} = \mathsf{refl}(\mathsf{base})$$

$$\mathsf{loop}^{(-n)} = \mathsf{loop}^{(-n+1)} \cdot \mathsf{loop}^{-1}$$

$$\mathsf{loop}^{(+n)} = \mathsf{loop}^{(n-1)} \cdot \mathsf{loop}$$

This could be, for example, defined by the $\mathbb{Z}$ recursor

$$winding : \Omega(\mathbb{S}^1) \to \mathbb{Z}$$

We take that $succ : \mathbb{Z} \simeq \mathbb{Z}$, and $pred = succ^{-1}$, defined by shifting all the numbers up/down by one

So, we have

$$ua(succ) : \mathbb{Z} =_{\mathcal{U}} \mathbb{Z}$$

By the recursion principle (not induction)

$$code : \mathbb{S}^1 \to \mathcal{U}$$

$$code(\mathsf{base}) = \mathbb{Z}$$

$$code(\mathsf{loop}) = ua(succ)$$

$$\mathsf{ap}_{code} : \Omega(\mathbb{S}^1) \to (\mathbb{Z} = \mathbb{Z})$$

$$winding(p) = tr[x.x](\mathsf{ap}_{code}(p))(0)$$

Proposition 5.

$$\Pi_{n:Z}.winding(\mathsf{loop}^{(n)}) =_Z n$$

Proof. Straightforward by induction □

Proposition 6.

$$\Pi_{l:\Omega(\mathbb{S}^1)}\mathsf{loop}^{winding(l)} =_{\Omega(\mathbb{S}^1)} l$$

We can't proceed by path induction on l, as the path is not free

$$encode : \Pi_{x:\mathbb{S}^1}(\mathsf{base} = x) \to code(x)$$

$$decode : \Pi_{x:\mathbb{S}^1} code(x) \to \mathsf{base} = x$$

$$encode(x, p) :\equiv tr[code](p)(o)$$

$$decode(x) :\equiv \mathsf{rec}_{\mathbb{S}^1}[z.code(z) \to \mathsf{base} = z](\lambda z.\mathsf{refl}(\mathsf{base}), \lambda n.\mathsf{loop}^{(n)})(x)$$

To complete the proof, you will need a path induction and a circle induction. □