23.5.2024 **Properties of type families over pushouts in HoTT**

Def A span on types A, B is a type S equipped with two maps $A \xleftarrow{f} S \xrightarrow{g} B$

Def A span is a pair of types A, B and a span on them

Def A cocone on a span $\mathcal{S}$ with vertex X is a pair of maps $A \xrightarrow{i} X \xleftarrow{j} B$ and a homotopy witnessing that the square

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] \arrow[dr, phantom, "\nearrow"] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] & X \end{tikzcd}$$

commutes, i.e. $H : (s : S) \to i(f(s)) = j(g(s))$

Def For a cocone (i, j, H), we define the cocone map

$$(X \to Y) \to \text{cocone}\, Y$$

$$h \longmapsto \begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] \arrow[drr, phantom, "h \cdot H \nearrow"] & B \arrow[dr, "j"] & \\ A \arrow[dr, "i"'] & & X \arrow[d, "h"] \\ & X \arrow[r, "h"'] & Y \end{tikzcd}$$

Def We say that a cocone (i, j, H) satisfies the universal property of pushouts if the cocone map is an equivalence for all Y's

Def A dependent cocone over a cone (i, j, H) with vertex $P : X \to \mathcal{U}$ is a triple (i', j', H') consisting of dependent maps

$i' : (a:A) \to P(ia)$; $j' : (b:B) \to P(jb)$;

and a "dependent homotopy" $H' : i' \circ f \sim_H j' \circ g$, i.e.

$(s:S) \to tr_P\,(Hs)\,(i'fs) =_{P_{jgs}} j'gs$

Def For a cocone (i, j, k), we define the dependent cocone map

$$((x:X) \to Px) \longrightarrow \text{dep-cocone } P$$
$$h \longmapsto (h \circ i,\ h \circ j,\ \lambda s \to apd_h\,(Hs))$$
$$: tr_P\,(Hs)\,(hifs) = hjgs$$

Def We say that a cocone satisfies the dependent universal property of pushouts if for all P, the dependent cocone map is an equivalence

Fact For any cocone c, $\Downarrow$ *

- c satisfies the universal property
- c satisfies the dependent universal property

* apparently not everyone is familiar with this symbol — it means "the following are equivalent", and its shape suggests that each property implies the next, looping from the bottom to the top

Note In Homotopy Type Theory, there are often multiple different ways of describing "universality".
In a very handwavy way, they usually require a base sort of objects, maps, dependent/fibered objects and dependent/fibered maps, that we can equip with a functorial structure.
In "Homotopy Initial Algebras in Type Theory", Awodey, Gambino and Sojakova describe this phenomenon where the base sort are (dependent) types and (dependent) functions, and the structure is an algebra of a polynomial functor.
It was then extended by Sojakova in "Higher Inductive Types as Homotopy Initial Algebras" to cases where the structure is an algebra for a W-suspension, which is a generalization of W-types that allows path constructors (which is what we need for e.g. cocones)

To set up, we can rearrange the arguments of the functorial action $(X \to Y) \to \text{structure } X \to \text{structure } Y$, so that for every structured object (X, s), we get a map $\text{ev-map}_Y : (X \to Y) \to \text{structure } Y$.
There's also a dependent version for (X, s)
$\text{dep-ev-map} : ((x : X) \to Px) \to \text{dep-structure}_{(X,s)} P$

The general principle tells us that the following are equivalent:

"initiality"
- ev-map_Y is an equivalence // universal property
- dep-ev-map_P is an equivalence // dependent universal property
- ev-map_Y has a unique section // recursion + uniqueness
- dep-ev-map_P has a section // induction

"induction"

We label the universal properties "initiality", since asking for e.g. ev-map$_Y$ to be an equivalence for all Y means asking for its fibers to be contractible, which amounts to showing that for all structured objects (Y, s'), the type $\Sigma(h : X \to Y).\ \text{ev-map}\ h = s'$ is contractible, and the mental image of (ev-map h) should be evaluating h at the structure s on X, so we are asking for the type of structure preserving maps from (X, s) to any (Y, s') to be contractible. That's a HoTT way of saying that (X, s) is initial.

For a concrete example, we may consider the case of pointed types. The structure on a type A is an element a_0 of A, and the structure on a family $P : A \to \mathcal{U}$ over (A, a_0) is an element p_0 of $P(a_0)$. The evaluation maps turn out to be

$$\begin{aligned} \text{ev-map}_Y : (A \to Y) &\to Y \\ h &\mapsto h\,a_0 \end{aligned} \qquad \begin{aligned} \text{dep-ev-map}_P : ((a : A) \to P a) &\to P a_0 \\ h &\mapsto h\,a_0\ , \end{aligned}$$

so a section of ev-map is a mapping from (Y, y_0) to the type $\Sigma(h : A \to Y).\ h\,a_0 = y_0$ of point-preserving maps, and so on.

At the end of the talk, we will see the induction principle of descent data for identity systems, where the sort of objects is descent data, and the structure is a point. We could instead work with an analogous universal property or recursion & uniqueness (as done by Kraus and von Raumer in "Path Spaces of Higher Inductive Types"). The choice of induction principle was made purely speculatively, guessing which property would be easier to formalize.

Descent

Motivation We want a nice description of what type families over pushouts "look like". We can already see that the type $(X \to \mathcal{U})$ is equivalent to the type of cocones with vertex $\mathcal{U}$. That type includes an identification in $\mathcal{U}$, so we can "clean" it up using univalence

Convention In the rest of the talk, let us assume a span $\mathcal{S}$

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \\ A & \end{tikzcd}$$

, a cocone c

$$\begin{tikzcd} S \arrow[r, "g"] \arrow[d, "f"'] & B \arrow[d, "j"] \\ A \arrow[r, "i"'] \arrow[ur, phantom, "H \Nearrow"] & X \end{tikzcd}$$

and σ witness UP of c satisfying the universal property of pushouts

Def Descent data (for pushouts) consists of type families $P_A : A \to \mathcal{U}$, $P_B : B \to \mathcal{U}$, and a family of equivalences $P_S : (s : S) \to P_A(fs) \simeq P_B(gs)$

Construction A type family $P : X \to \mathcal{U}$ induces the descent data

$$\text{descent-data-family } P := (P \circ i,\ P \circ j,\ \lambda s \to tr_P(Hs))$$

Prop There is a commuting triangle

$$\begin{tikzcd} (X \to \mathcal{U}) \arrow[rr, "\text{cocone-map}", "\simeq"'] \arrow[dr, "\text{dd-fam}"'] & & \text{cocone } \mathcal{U} \arrow[dl, "\simeq", "\text{tot (tot (equiv-eq))}"'] \\ & \text{descent-data} & \end{tikzcd}$$

Proof We need to check that $(P\circ i,\ P\circ j,\ \lambda s \to tr_P(Hs)) = (P\circ i,\ P\circ j,\ \lambda s \to \text{equiv-eq}(ap_P(Hs)))$

Since the first two components are identical, it suffices to show $(s:S) \to tr_P(Hs) = \text{equiv-eq}(ap_P(Hs))$, which is a general principle:

Lemma For all $Y:\mathcal{U}$, $P: Y \to \mathcal{U}$, $x,y:Y$, $p: x=y$, we have
$$tr_P\, p = \text{equiv-eq}(ap_P\, p)$$

Proof By path induction, it suffices to consider the case $y \doteq x$, $p \doteq \text{refl}$. Then the statement reduces to $id = id$, which holds by reflexivity

Corollary The map $(X \to \mathcal{U}) \to \text{descent-data}$ is an equivalence, by the 3-for-2 property of equivalences

Def Given descent data (P_A, P_B, P_S), (Q_A, Q_B, Q_S), the type of morphisms between them is given by the type

$$(P_A, P_B, P_S) \to (Q_A, Q_B, Q_S) := \sum (h_A : (a:A) \to P_A a \to Q_A a),$$
$$\sum (h_B : (b:B) \to P_B b \to Q_B b),$$
$$h_S : (s:S) \to$$

$$\begin{tikzcd}
P_A(fs) \arrow[r, "h_A(fs)"] \arrow[d, "P_S s"'] \arrow[dr, phantom, "\nearrow"] & Q_A(fs) \arrow[d, "Q_S s"] \\
P_B(gs) \arrow[r, "h_B(gs)"'] & Q_B(gs)
\end{tikzcd}$$

Def Similarly, equivalences of descent data are like morphisms, except h_A, h_B are fiberwise equivalences

Prop Equivalences of descent data characterize their identity type, i.e. the canonical map

$$\begin{aligned} \text{equiv-eq-dd} : ((P_A, P_B, P_S) = (Q_A, Q_B, Q_S)) &\to (P_A, P_B, P_S) \simeq (Q_A, Q_B, Q_S) \\ \text{refl} &\longmapsto (\text{id}, \text{id}, \text{refl-htpy}) \end{aligned}$$

is an equivalence

Proof Mechanical by the structure identity principle

Note It will be useful to talk about families equipped with descent data, as a way to parameterize constructions by user-provided equivalences

Def A family equipped with descent data, denoted $P \approx (P_A, P_B, P_S)$, is a triple consisting of a type family $P : X \to \mathcal{U}$, descent data (P_A, P_B, P_S), and an equivalence of descent data descent-data-family $P \simeq (P_A, P_B, P_S)$

Note Unfolding, we have two fiberwise equivalences

$$e_A : (a : A) \to P(ia) \simeq P_A\, a$$
$$e_B : (b : B) \to P(jb) \simeq P_B\, b$$

and a square

$$e_S : (s : S) \to$$

$$\begin{tikzcd} P(ifs) \arrow[r, "e_A(fs)"] \arrow[d, "tr_P(Hs)"'] \arrow[dr, phantom, "\Nearrow"] & P_A(fs) \arrow[d, "P_S s"] \\ P(jgs) \arrow[r, "e_B(gs)"'] & P_B(gs) \end{tikzcd}$$

Corollary For every descent data (P_A, P_B, P_S), there is a unique family $P : X \to \mathcal{U}$ s.t. $P \approx (P_A, P_B, P_S)$

Proof The type $\Sigma(P : X \to \mathcal{U}).\ P \approx (P_A, P_B, P_S)$ is equivalent to $\Sigma(P : X \to \mathcal{U}).\ \text{dd-fam}\ P = (P_A, P_B, P_S)$, which is the type $\text{fib}_{\text{dd-fam}}(P_A, P_B, P_S)$. Since dd-fam is an equivalence, its fibers are contractible

Prop Given two families with descent data $P \approx (P_A, P_B, P_S)$, $Q \approx (Q_A, Q_B, Q_S)$, the descent data corresponding to the family $\lambda x \to Px \to Qx$ is

$$
\begin{aligned}
(&\lambda a \to P_A a \to Q_A a, \\
&\lambda b \to P_B b \to Q_B b, \\
&\lambda s \to \lambda h \to e^Q_S s \circ h \circ {e^P_S s}^{-1}) \\
&\qquad : (P_A(fs) \to Q_A(fs)) \simeq (P_B(gs) \to Q_B(gs))
\end{aligned}
$$

// note that pre- and post-composition by equivalences are equivalences

Proof Uses the characterization of transport in families of function types $\quad tr_{(\lambda x \to Px \to Qx)}\, p\ f = tr_Q\, p \circ f \circ tr_P(p^{-1})$

Construction Given families with descent data $P \approx (P_A, P_B, P_S)$ and $Q \approx (Q_A, Q_B, Q_S)$, there is a map

$$
\begin{aligned}
((x : X) \to Px \to Qx) &\to (P_A, P_B, P_S) \to (Q_A, Q_B, Q_S) \\
h &\mapsto (\lambda a \to e^Q_A a \circ h(ia) \circ {e^P_A a}^{-1}, \\
&\qquad \lambda b \to e^Q_B b \circ h(jb) \circ {e^P_B b}^{-1}, \\
&\qquad \lambda s \to K\,(\mathrm{apd}_h\,(Hs)))
\end{aligned}
$$

where K is the composed equivalence :

$$\left(\mathrm{tr}_{(\lambda x \to Px \to Qx)}(Hs)\,(h(ifs)) = h(jgs)\right) \simeq \left(\mathrm{tr}_Q(Hs)\circ h(ifs)\circ \mathrm{tr}_P(Hs)^{-1} = h(jgs)\right)$$

$$\simeq \left(\mathrm{tr}_Q(Hs)\circ h(ifs) \sim h(jgs)\circ \mathrm{tr}_P(Hs)\right) \simeq \Big(e^Q_B(gs)\circ h(jgs)\circ e^P_B(gs)^{-1}\circ P_s s$$
$$\sim Q_s s \circ e^Q_A(fs)\circ h(ifs)\circ e^P_A(fs)^{-1}\Big)$$

where the last equivalence is given by pasting squares

$$\begin{tikzcd}
P\,ifs \arrow[r, "h(ifs)"] \arrow[d, "\mathrm{tr}_P(Hs)"'] & Q\,ifs \arrow[d, "\mathrm{tr}_Q(Hs)"] \\
P\,jgs \arrow[r, "h(jgs)"'] & Q\,jgs
\end{tikzcd}
\quad \simeq \quad
\begin{tikzcd}
P_A fs \arrow[r, "e^P_A(fs)^{-1}"] \arrow[d, "P_s s"'] & P\,ifs \arrow[r, "h(ifs)"] \arrow[d, "\mathrm{tr}_P(Hs)"] & Q\,ifs \arrow[r, "e^Q_A(fs)"] \arrow[d] & Q_A fs \arrow[d, "Q_s s"] \\
P_B gs \arrow[r, "e^P_B(gs)^{-1}"'] & P\,jgs \arrow[r, "h(jgs)"'] & Q\,jgs \arrow[r, "e^Q_B(gs)"'] & Q_B gs
\end{tikzcd}$$

Prop The above map is an equivalence

Corollary Given a morphism of descent data $(h_A, h_B, h_s) : (P_A, P_B, P_s) \to (Q_A, Q_B, Q_s)$, the type of maps $h : (x:X) \to Px \to Qx$ such that the following diagrams commute, is contractible

$$\begin{tikzcd}
P(ia) \arrow[r, "h(ia)"] \arrow[d, "e^P_A a"'] & Q(ia) \arrow[d, "e^Q_A a"] \\
P_A a \arrow[r, "h_A a"'] & Q_A a
\end{tikzcd}
\qquad
\begin{tikzcd}
P(jb) \arrow[r, "h(jb)"] \arrow[d, "e^P_B b"'] & Q(jb) \arrow[d, "e^Q_B b"] \\
P_B b \arrow[r, "h_B b"'] & Q_B b
\end{tikzcd}$$

$$\begin{tikzcd}
 & P(ifs) \arrow[dl, "\mathrm{tr}_P(Hs)"'] \arrow[d, "e^P_A(fs)"] \arrow[dr, "h(ifs)"] & \\
P(jgs) \arrow[d, "e^P_B(gs)"'] \arrow[dr, "h(jgs)"] & P_A(fs) \arrow[dl, "P_s s"'] \arrow[dr, "h_A(fs)"] & Q(ifs) \arrow[dl, "\mathrm{tr}_Q(Hs)"'] \arrow[d, "e^Q_A(fs)"] \\
P_B(gs) \arrow[dr, "h_B(gs)"'] & Q(jgs) \arrow[d, "e^Q_B(gs)"] & Q_A(fs) \arrow[dl, "Q_s s"] \\
 & Q_B(gs) &
\end{tikzcd}$$

$\mathrm{apd}_h(Hs) : \mathrm{tr}_Q(Hs)\circ h(ifs)\circ \mathrm{tr}_P(Hs^{-1}) = h(jgs)$

$\rightsquigarrow \mathrm{tr}_Q(Hs)\circ h(ifs) \sim h(jgs)\circ \mathrm{tr}_P(Hs)$ is the top face

Prop Similarly a family of equivalences $(x:X) \to Px \simeq Qx$ is given by an equivalence $(P_A, P_B, P_S) \simeq (Q_A, Q_B, Q_S)$

Note These correspondences can be abstracted further by defining sections of descent data, and observing that sections of P correspond to sections of (P_A, P_B, P_S)

Def A <u>section</u> of descent data (P_A, P_B, P_S) is a triple

$$\begin{aligned}
(t_A &: (a:A) \to P_A a, \\
t_B &: (b:B) \to P_B b, \\
t_S &: (s:S) \to P_S s\ (t_A(fs)) = t_B(gs)\ )
\end{aligned}$$

Prop Given $P \simeq (P_A, P_B, P_S)$, sections $t : (x:X) \to Px$ of P are given by sections of (P_A, P_B, P_S), and we have the computation rules

$$\begin{tikzcd}
(a:A) \arrow[r, "t_A"] \arrow[d, "i"'] & P_A a \\
(ia:X) \arrow[r, "t"'] & P(ia) \arrow[u, "e_A a"']
\end{tikzcd}
\qquad
\begin{tikzcd}
(b:B) \arrow[r, "t_B"] \arrow[d, "j"'] & P_B b \\
(jb:X) \arrow[r, "t"'] & P(jb) \arrow[u, "e_B b"']
\end{tikzcd}$$

I'm sorry I don't know how to draw this better

$$\begin{tikzcd}
(s:S) \arrow[r, "t_S"] \arrow[d, "H"'] & P_S s(t_A fs) = t_B(gs) \\
(Hs : ifs = jgs) \arrow[d, "\mathrm{apd}_t"'] & P_S s(e_A(fs)(t\,ifs)) = e_B(gs)(t\,jgs) \arrow[u, "\text{above computations}"'] \\
\mathrm{tr}_P(Hs)(t\,ifs) =_{P\,jgs} t\,jgs \arrow[r, "\mathrm{ap}_{e_B(gs)}"'] & e_B(gs)(\mathrm{tr}_P(Hs)(t\,ifs)) =_{P_B(gs)} e_B(gs)(t\,jgs) \arrow[u, "e_S s^{-1} \cdot -"']
\end{tikzcd}$$

Proof (sketch) The map factorizes as

$$\begin{tikzcd}
((x:X) \to Px) \arrow[r, "\simeq"] \arrow[dr] & \text{dep-cocone } c\ P \arrow[d, "\simeq"] \\
& \text{section } (P_A, P_B, P_S)
\end{tikzcd}$$

Where the top map is an equivalence by UP, and the right map is an equivalence constructed using functoriality of Σ

Prop Given a point $a_0 : A$, the family $\mathrm{Id}_X(ia_0) : X \to \mathcal{U}$
$x \mapsto ia_0 = x$

is characterized by the descent data

$$\begin{aligned}
(I_A &: A \to \mathcal{U} \\
&\quad a \mapsto ia_0 =_X ia \\
I_B &: B \to \mathcal{U} \\
&\quad b \mapsto ia_0 =_X jb \\
I_S &: (s:S) \to I_A(fs) \simeq I_B(gs) \\
&\quad s \mapsto - \cdot Hs \\
&\qquad : (ia_0 = ifs) \simeq (ia_0 = jgs)\ )
\end{aligned}$$

Proof $e_A : (a:A) \to (ia_0 = ia) \simeq (ia_0 = ia)$ ✓ id

$e_B : (b:B) \to (ia_0 = jb) \simeq (ia_0 = jb)$ ✓ id

$e_S : (s:S)$

$$\begin{tikzcd}
(ia_0 = ifs) \arrow[r, "id"] \arrow[d, "\mathrm{tr}_{\mathrm{Id}_X ia_0}(Hs)"'] \arrow[dr, phantom, "\Swarrow"] & (ia_0 = ifs) \arrow[d, "-\cdot Hs"] \\
ia_0 = jgs \arrow[r, "id"'] & (ia_0 = jgs)
\end{tikzcd}$$

✓ general lemma characterizing transport in based identity types

Recap We built a dictionary for translating between families over pushouts and descent data

	Type families	Descent data
obj	$P : X \to \mathcal{U}$	(P_A, P_B, P_S)
hom	$(x:X) \to P x \to Q x$	$(P_A, P_B, P_S) \to (Q_A, Q_B, Q_S)$
equiv	$(x:X) \to P x \simeq Q x$	$(P_A, P_B, P_S) \to (Q_A, Q_B, Q_S)$
section	$(x:X) \to P x$	section (P_A, P_B, P_S)
based identity system	$\lambda x \to x_0 =_X x$	(I_A, I_B, I_S)

todo induction principle of based identity systems, allowing for alternative constructions

"$P : X \to \mathcal{U}$, $x_0 : X$, $p_0 : P x_0$
is an identity system if
for all $Q : (x:X) \to P x \to \mathcal{U}$
the evaluation map
$\text{ev-refl}_Q : ((x:X) \to (p : P x) \to Q\, x\, p)$
$\to Q\, x_0\, p_0$
$h \mapsto h\, x_0\, p_0$
has a section"
then there is a unique equivalence
$(x:X) \to (x_0 = x) \simeq P x$ taking refl_{x_0}
to p_0

We can start:
"(P_A, P_B, P_S), $a_0 : A_0$, $p_0 : P_A\, a_0$
is an identity system
if for all ???"
→ we don't have a
description of families
of shape $(x:X) \to P x \to \mathcal{U}$

then there is a unique equivalence
$(I_A, I_B, I_S) \simeq (P_A, P_B, P_S)$ taking
$\text{refl}_{i a_0}$ to p_0

BUT! We can uncurry

$P : X \to \mathcal{U}$, $x_0 : X$, $p_0 : P\,x_0$
is an identity system if
for all $Q : \Sigma X P \to \mathcal{U}$,
the evaluation map
$\text{ev-refl}_Q : ((u : \Sigma X P) \to Q\,u)$
$\to Q(x_0, p_0)$
$h \mapsto h(x_0, p_0)$
has a section

(P_A, P_B, P_S), $a_0 : A$, $p_0 : P_A\,a_0$
is an identity system if
for all $(Q_{\Sigma A}, Q_{\Sigma B}, Q_{\Sigma S})$
the evaluation map
$\text{ev-refl}' : \text{section}\,(Q_{\Sigma A}, Q_{\Sigma B}, Q_{\Sigma S})$
$\to Q_{\Sigma A}(a_0, p_0)$
$(t_A, t_B, t_S) \mapsto t_A(a_0, p_0)$
has a section

IOW, every $q_0 : Q(x_0, p_0)$
induces a section $h : (u : \Sigma X P) \to Q u$
such that $h(x_0, p_0) = q_0$

IOW, every $q_0 : Q_{\Sigma A}(a_0, p_0)$
induces a section $t : \text{section}(Q_{\Sigma A}, Q_{\Sigma B}, Q_{\Sigma S})$
such that $t_A(a_0, p_0) = q_0$

HOWEVER! Now we are talking about the cocone

$$\begin{tikzcd}
\Sigma S\,(P_A \circ f) \arrow[r, "\text{tot}_g\,P_S"] \arrow[d, "\text{tot}_f\,\text{id}"'] \arrow[dr, phantom, "\text{tot}_H\,(\text{flip}\,e_S)^{-1}"] & \Sigma B\,P_B \arrow[d, "\text{tot}_j\,e_B^{-1}"] \\
\Sigma A\,P_A \arrow[r, "\text{tot}_i\,e_A^{-1}"'] & \Sigma X\,P
\end{tikzcd}$$

where $\text{flip}\,(e_S) : (s : S) \to$

$$\begin{tikzcd}
P_A(fs) \arrow[r, "e_A(fs)^{-1}"] \arrow[d, "P_S s"'] & P(ifs) \arrow[d, "\text{tr}_P(Hs)"] \\
P_B(gs) \arrow[r, "e_B(gs)^{-1}"'] & P(jgs)
\end{tikzcd}$$

and we don't know anything about families over it, since we only know that c is a pushout, not that this Σc is a pushout. Is it?

Flattening

Lemma (flattening) It is!

Note When (properly) stating and proving the flattening lemma, it is convenient to consider families with descent data where the equivalence goes the other way, i.e. $(P_A, P_B, P_S) \approx P$ instead of $P \approx (P_A, P_B, P_S)$.

Lemma (flattening) Given $(P_A, P_B, P_S) \approx P$, the cocone

$$\begin{tikzcd}
\Sigma\, S\, (P_A \circ f) \arrow[r, "\mathrm{tot}_g\, P_S"] \arrow[d, "\mathrm{tot}_f\, \mathrm{id}"'] \arrow[dr, phantom, "\mathrm{tot}_H\, e_S^{-1} \Nearrow"] & \Sigma\, B\, P_B \arrow[d, "\mathrm{tot}_j\, e_B"] \\
\Sigma\, A\, P_A \arrow[r, "\mathrm{tot}_i\, e_A"'] & \Sigma\, X\, P
\end{tikzcd}$$

satisfies the universal property of pushouts

Proof The proof consists of two steps:

1) We show that

$$\begin{tikzcd}
\Sigma\, S\, (P \circ i \circ f) \arrow[r, "\mathrm{tot}_g\, (\mathrm{tr}_P(H s))"] \arrow[d, "\mathrm{tot}_f\, \mathrm{id}"'] \arrow[dr, phantom, "\mathrm{tot}_H\, \mathrm{refl} \Nearrow"] & \Sigma\, B\, (P \circ j) \arrow[d, "\mathrm{tot}_j\, \mathrm{id}"] \\
\Sigma\, A\, (P \circ i) \arrow[r, "\mathrm{tot}_i\, \mathrm{id}"'] & \Sigma\, X\, P
\end{tikzcd}$$

is a pushout

2) We observe that the two squares of total spaces are "equivalent" in a way that preserves the universal property

1) We need to show that

$$\text{cocone-map}_Y : (\Sigma X P \to Y) \to \text{cocone } Y$$
$$h \longmapsto (h \circ \text{tot}_i\, \text{id},\ h \circ \text{tot}_j\, \text{id},\ h \cdot \text{tot}_H\, \text{refl})$$

is an equivalence.

We can form a commuting pentagon

$$\begin{tikzcd}
\Sigma X P \to Y \arrow[rr, "\text{cocone-map}"] & & \begin{array}{l} \Sigma (h_A : \Sigma A (P \circ i) \to Y) \\ \Sigma (h_B : \Sigma B (P \circ j) \to Y) \\ ((s,p) : \Sigma S (P \circ i \circ f)) \to \\ \quad h_A (fs, p) = h_B (gs, \text{tr}_P (Hs)\, p) \end{array} \arrow[d, "\text{curry}^3", "\simeq"'] \\
(x : X) \to Px \to Y \arrow[u, "\text{uncurry}"', "\simeq"] \arrow[dr, "\simeq", "\text{dep-cocone-map}_{(\lambda x \to Px \to Y)}\ \text{of } c!"'] & & \begin{array}{l} \Sigma (h_A : (a : A) \to P(ia) \to Y) \\ \Sigma (h_B : (b : B) \to P(jb) \to Y) \\ (s : S)\, (p : P i f s) \to \\ \quad h_A (fs)\, p =_Y h_B (gs)\, (\text{tr}_P (Hs)\, p) \end{array} \arrow[dl, "M"', "\simeq"] \\
& \begin{array}{l} \Sigma (h_A : (a : A) \to P(ia) \to Y) \\ \Sigma (h_B : (b : B) \to P(jb) \to Y) \\ (s : S) \to \text{tr}_{(\lambda x \to Px \to Y)} (Hs)\, (h_A (fs)) =_{P jgs \to Y} h_B (gs) \end{array} &
\end{tikzcd}$$

where the equivalence $M : ((p : P i f s) \to h_A (fs)\, p = h_B (gs)\, (\text{tr}_P (Hs)\, p))$
$\simeq (\text{tr}_{(\lambda x \to Px \to Y)} (Hs)\, (h_A (fs)) = h_B (gs))$

is constructed in full generality as

For all $i, j : S \to X$, $H : i \sim j$, $P : X \to \mathcal{U}$, $Y : \mathcal{U}$,
$k : (s : S) \to P(is) \to Y$, $\ell : (s : S) \to P(js) \to Y$, $s : S$,

$$((p : P(is)) \to k\, s\, p =_Y \ell\, s\, (\text{tr}_P (Hs)\, p))$$
$$\simeq \text{tr}_{(\lambda x \to Px \to Y)} (H s)\, (k s) =_{P(js) \to Y} \ell s$$

using homotopy induction, we may assume that $j \doteq i$, $H \doteq \text{refl-htpy}$. Then it suffices to show that

$$((p : P(is)) \to k\,s\,p = \ell\,s\,p) \simeq (k\,s = \ell\,s),$$

which holds by function extensionality.

Also by homotopy induction, we show the computation rule: For all $h : (x:X) \to Px \to Y$, we have

$$M\ (h \circ i)\ (h \circ j)\ (\lambda p \to \mathrm{ap}_{\mathrm{uncurry}\ h}\ (\text{eq-}\Sigma(Hs, \mathrm{refl}))) = \mathrm{apd}_h\ (Hs)$$

// sanity check: $\mathrm{uncurry}\ h : \Sigma X P \to Y$, $\text{eq-}\Sigma(Hs, \mathrm{refl}) : (is, p) =_{\Sigma X P} (js, \mathrm{tr}_P (Hs)\ p)$

$$\begin{aligned}
\to \mathrm{ap}\cdots &: h\ (is)\ p = h\ (js)\ (\mathrm{tr}_P (Hs)\ p) \\
\to M\cdots &: \mathrm{tr}\ (Hs)\ (h\ (is)) = h\ (js) \\
\mathrm{apd}_h\ (Hs) &: \mathrm{tr}\ (Hs)\ (h\ (is)) = h\ (js)
\end{aligned}$$

✓

the computation itself is omitted

Chasing an element $h : (x:X) \to Px \to Y$ along the diagram, we get

$$\begin{tikzcd}
 & \mathrm{ucurry}\ h \arrow[r, mapsto] & (\lambda (a,p) \to h\ (ia)\ p,\ \lambda(b,p) \to h\ (jb)\ p,\ \lambda(s,p) \to \mathrm{ap}_{\mathrm{uncurry}\ h}\ (\text{eq-}\Sigma\ (Hs, \mathrm{refl}))) \arrow[d, mapsto] \\
h \arrow[ur, mapsto] \arrow[ddd, mapsto] & & (h \circ i,\ h \circ j,\ \lambda s\ p \to \mathrm{ap}_{\mathrm{uncurry}\ h}\ (\text{eq-}\Sigma(Hs, \mathrm{refl}))) \arrow[d, mapsto] \\
 & & (h \circ i,\ h \circ j,\ \lambda s \to M\ (\lambda p \to \mathrm{ap}_{\mathrm{uncurry}\ h}\ (\text{eq-}\Sigma(Hs, \mathrm{refl})))) \\
 & & \\
 & (h \circ i,\ h \circ j,\ \lambda s \to \mathrm{apd}_h\ (Hs)) &
\end{tikzcd}$$

and the endpoints are equal by the above computation

2) There is a commuting cube (commutativity omitted, but not terribly difficult)

$$
\begin{tikzcd}
 & & \Sigma S\,(P_A \circ f) \arrow[dll, "\mathrm{tot}_f\, \mathrm{id}"'] \arrow[d, "\mathrm{tot}\,(e_A \circ f)"] \arrow[drr, "\mathrm{tot}_g\, P_S"] & & \\
\Sigma A\, P_A \arrow[d, "\mathrm{tot}\, e_A"'] \arrow[drr, "\mathrm{tot}_i\, e_A"] & & \Sigma S\,(P \circ i \circ f) \arrow[dll, "\mathrm{tot}_f\, \mathrm{id}"'] \arrow[drr, "\mathrm{tot}_g (\mathrm{tr}_{P(H_S)})"] & & \Sigma B\, P_B \arrow[dll, "\mathrm{tot}_j\, e_B"'] \arrow[d, "\mathrm{tot}\, e_B"] \\
\Sigma A\,(P \circ i) \arrow[drr, "\mathrm{tot}_i\, \mathrm{id}"'] & & \Sigma X\, P \arrow[d, "\mathrm{id}"] & & \Sigma B\,(P \circ j) \arrow[dll, "\mathrm{tot}_j\, \mathrm{id}"] \\
 & & \Sigma X\, P & &
\end{tikzcd}
$$

where the vertical maps are equivalences. This data is an equivalence of commuting squares, and since the bottom is a pushout, we get that the top is a pushout

Prop The descent data (I_A, I_B, I_S) with $\mathrm{refl}_{ia_0} : I_A\, a_0$ satisfies the induction principle of identity systems

Proof Assume descent data $(Q_{\Sigma A}, Q_{\Sigma B}, Q_{\Sigma S})$. Then there exists a unique family $Q : (\Sigma(x:X).\, ia_0 = x) \to \mathcal{U}$ such that $Q \approx (Q_{\Sigma A}, Q_{\Sigma B}, Q_{\Sigma S})$. There is a commuting square

$$\begin{tikzcd}
(u : \Sigma(x:X).\, ia_0 = x) \to Q\,u \arrow[r, "\simeq"] \arrow[d, "\text{ev-refl}"'] & \text{section } (Q_{\Sigma A}, Q_{\Sigma B}, Q_{\Sigma S}) \arrow[d, "\text{ev-refl}'"] \\
Q(ia_0, \mathrm{refl}_{ia_0}) \arrow[r, "e^Q_A(a_0, \mathrm{refl}_{ia_0})"'] & Q_{\Sigma A}(a_0, \mathrm{refl}_{ia_0})
\end{tikzcd}$$

$$\begin{tikzcd}
t \arrow[r, mapsto] \arrow[dd, mapsto] & (t_A, t_B, t_S) \arrow[d, mapsto] \\
 & t_A(a_0, \mathrm{refl}_{ia_0}) \arrow[d, phantom, "="] \\
t(ia_0, \mathrm{refl}_{ia_0}) \arrow[r, mapsto] & e^Q_A(t(ia_0, \mathrm{refl}_{ia_0}))
\end{tikzcd}$$

commutes by computation of t_A

Since ev-refl, $e^Q_A(a_0, \mathrm{refl}_{ia_0})$, and the top map are equivalences, then ev-refl' is also an equivalence, in particular it has a section

Prop For any descent data (P_A, P_B, P_S) and $p_0 : P_A\, a_0$ satisfying the induction principle of identity systems, there is an equivalence $(I_A, I_B, I_S) \simeq (P_A, P_B, P_S)$ sending refl_{ia_0} to p_0

Proof There is a unique family $P : X \to \mathcal{U}$ such that $P \approx (P_A, P_B, P_S)$. To give an equivalence $(I_A, I_B, I_S) \simeq (P_A, P_B, P_S)$ sending refl_{ia_0} to p_0 is to give an equivalence $(x : X) \to (ia_0 = x) \simeq Px$ sending refl_{ia_0} to $e_A a_0^{-1}(p_0) : P(ia_0)$.

We can define such a map by path induction.

To show that it is an equivalence, by the fundamental theorem of identity types it suffices to show that $\Sigma X P$ is contractible.

It is contractible exactly when it satisfies singleton induction: For all $Q : \Sigma X P \to \mathcal{U}$, the evaluation map

$$\begin{aligned} \text{ev-refl} : ((u : \Sigma X P) \to Q u) &\to Q(ia_0, e_A a_0^{-1}(p_0)) \\ h &\mapsto h(ia_0, e_A a_0^{-1}(p_0)) \end{aligned}$$

has a section. It fits into a commuting square

$$\begin{tikzcd}
\mathrm{section}(Q_{\Sigma A}, Q_{\Sigma B}, Q_{\Sigma S}) \arrow[r, "\simeq"] \arrow[d, "\text{ev-refl}'"'] & (u : \Sigma X P) \to Q u \arrow[d, "\text{ev-refl}"] \\
Q_{\Sigma A}(a_0, p_0) \arrow[r, "e^Q_A(a_0, p_0)^{-1}"'] & Q(ia_0, e_A a_0^{-1}(p_0))
\end{tikzcd}$$

where $(Q_{\Sigma A}, Q_{\Sigma B}, Q_{\Sigma S})$ is the unique descent data such that $Q \approx (Q_{\Sigma A}, Q_{\Sigma B}, Q_{\Sigma S})$. The square commutes by mirroring the computation square of t_A.

Both $e^Q_A(a_0, p_0)^{-1}$ and ev-refl' have a section, so ev-refl has one