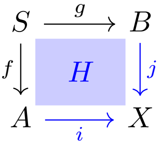
~~Towards a~~ Formalization of Wärn's zigzag construction

Vojtěch Štěpančík

Outline

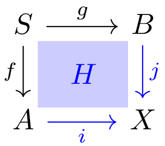
Pushout of $A \xleftarrow{f} S \xrightarrow{g} B$ in agda-unimath

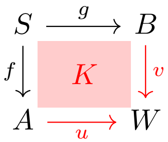
- A cocone (i, j, H) is a pushout



Pushout of $A \xleftarrow{f} S \xrightarrow{g} B$ in agda-unimath

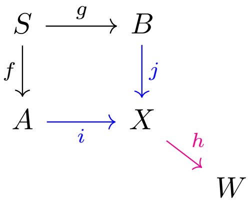
- A cocone (i, j, H) is a pushout if every cocone under the same span

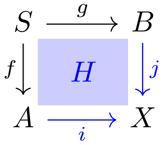


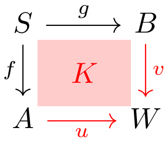


Pushout of $A \xleftarrow{f} S \xrightarrow{g} B$ in agda-unimath

- A cocone (i, j, H) is a pushout if every cocone under the same span induces a unique map $h : X \to W$

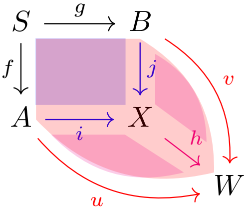


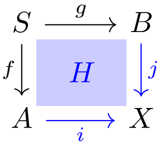


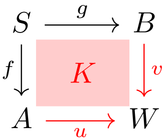


Pushout of $A \xleftarrow{f} S \xrightarrow{g} B$ in agda-unimath

- A cocone (i, j, H) is a pushout if every cocone under the same span induces a unique map $h : X \to W$ which factors W.

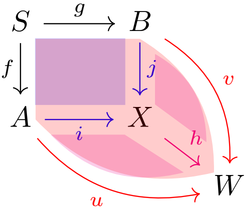




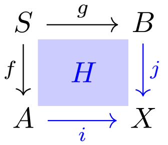


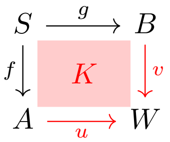
Pushout of $A \xleftarrow{f} S \xrightarrow{g} B$ in agda-unimath

- A cocone (i, j, H) is a pushout if every cocone under the same span induces a unique map $h : X \to W$ which factors W.



- $\leftrightarrow$ "the map from $(X \to W)$ to cocones with vertex W is an equivalence"





Describing identity types of pushouts

- Pushouts are a fundamental method for creating spaces by gluing

[1]Wärn. Path Spaces of Pushouts, 2023, https://dwarn.se/po-paths.pdf

Describing identity types of pushouts

- Pushouts are a fundamental method for creating spaces by gluing
- Understanding path spaces gives connectivity and truncation estimates

[1] Wärn. Path Spaces of Pushouts, 2023, https://dwarn.se/po-paths.pdf

Describing identity types of pushouts

- Pushouts are a fundamental method for creating spaces by gluing
- Understanding path spaces gives connectivity and truncation estimates
- Fixing $x_0 : X$ in a pushout, give $I_{x_0} : X \to \mathcal{U}$ and $I_{x_0}(x) \simeq (x_0 =_X x)$

[1] Wärn. Path Spaces of Pushouts, 2023, `https://dwarn.se/po-paths.pdf`

Describing identity types of pushouts

- Pushouts are a fundamental method for creating spaces by gluing
- Understanding path spaces gives connectivity and truncation estimates
- Fixing $x_0 : X$ in a pushout, give $I_{x_0} : X \to \mathcal{U}$ and $I_{x_0}(x) \simeq (x_0 =_X x)$
- Wärn described the "zigzag construction"[1] in 2023

[1] Wärn. Path Spaces of Pushouts, 2023, `https://dwarn.se/po-paths.pdf`

Describing identity types of pushouts

- Pushouts are a fundamental method for creating spaces by gluing
- Understanding path spaces gives connectivity and truncation estimates
- Fixing $x_0 : X$ in a pushout, give $I_{x_0} : X \to \mathcal{U}$ and $I_{x_0}(x) \simeq (x_0 =_X x)$
- Wärn described the "zigzag construction"[1] in 2023
- No formalization existed for almost two years, until now!

[1] Wärn. Path Spaces of Pushouts, 2023, `https://dwarn.se/po-paths.pdf`

Setting

- Book HoTT
- Built with agda-unimath[2], with plans to upstream

[2] Rijke et al. agda-unimath, `https://github.com/UniMath/agda-unimath`

Setting

- Book HoTT
- Built with agda-unimath[2], with plans to upstream

Consequences of choosing agda-unimath

- Postulated pushouts
- Reusable code
- Performance matters

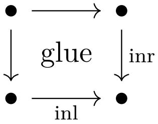
[2] Rijke et al. agda-unimath, `https://github.com/UniMath/agda-unimath`

Conventions

- Fix a span $A \overset{f}{\leftarrow} S \overset{g}{\rightarrow} B$ and its pushout (i, j, H) on X

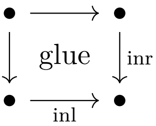
Conventions

- Fix a span $A \xleftarrow{f} S \xrightarrow{g} B$ and its pushout (i, j, H) on X
- For postulated pushouts use $(\mathrm{inl}, \mathrm{inr}, \mathrm{glue})$

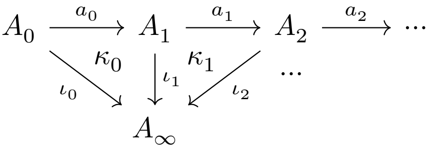


Conventions

- Fix a span $A \xleftarrow{f} S \xrightarrow{g} B$ and its pushout (i, j, H) on X
- For postulated pushouts use $(\mathrm{inl}, \mathrm{inr}, \mathrm{glue})$



- Recall sequential colimits of diagrams $A_\bullet$



or

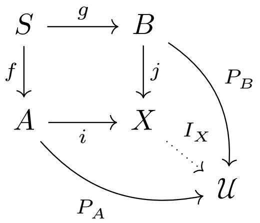
$$A_0 \xrightarrow{a_0} A_1 \xrightarrow{a_1} A_2 \xrightarrow{a_2} \cdots \xrightarrow{\iota} A_\infty$$

Approach

- The goal is to construct a type family $I_{x_0} : X \to \mathcal{U}$

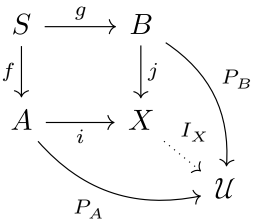
Approach

- The goal is to construct a type family $I_{x_0} : X \to \mathcal{U}$
- Cocone with vertex $\mathcal{U}$



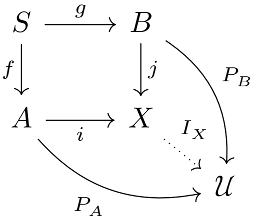
Approach

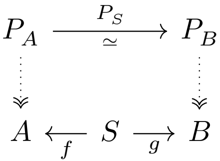
- The goal is to construct a type family $I_{x_0} : X \to \mathcal{U}$
- Cocone with vertex $\mathcal{U}$
- By univalence, $P_A(fs) = P_B(gs)$ is equivalently $P_A(fs) \simeq P_B(gs)$



Approach

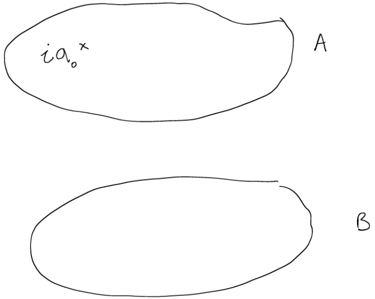
- The goal is to construct a type family $I_{x_0} : X \to \mathcal{U}$
- Cocone with vertex $\mathcal{U}$
- By univalence, $P_A(fs) = P_B(gs)$ is equivalently $P_A(fs) \simeq P_B(gs)$
- **Descent data** (P_A, P_B, P_S): type families and equivalences $P_S(s) : P_A(fs) \simeq P_B(gs)$





Construction of the type families

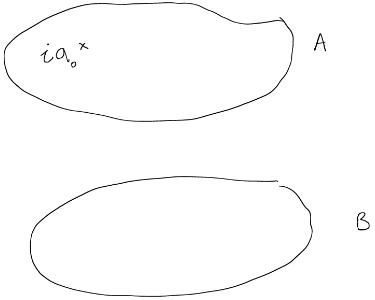
- $P_A(a)$ and $P_B(b)$ defined as sequential colimits



$P_B^0(b)$ $P_A^0(a)$ $P_B^1(b)$ $P_A^1(a)$

Construction of the type families

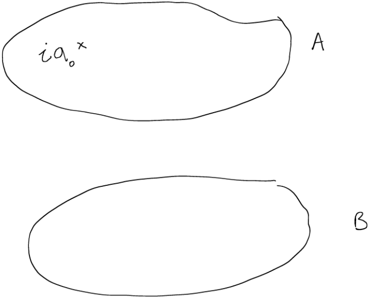
- $P_A(a)$ and $P_B(b)$ defined as sequential colimits
- $P_A^n(a) :=$ "type of paths from $i(a_0)$ to $i(a)$ allowing at most n "crossings" from B to A"



$P_B^0(b)$ $P_A^0(a)$ $P_B^1(b)$ $P_A^1(a)$

Construction of the type families

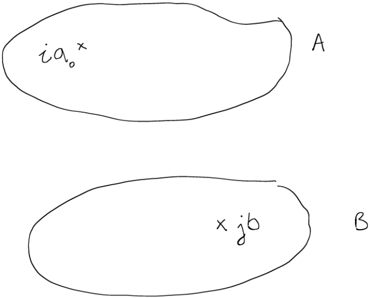
- $P_A(a)$ and $P_B(b)$ defined as sequential colimits
- $P_A^n(a) :=$ "type of paths from $i(a_0)$ to $i(a)$ allowing at most n "crossings" from B to A"
- $P_B^0(b) := \mathbb{0}$, $P_A^0(a) := (a_0 =_A a)$



$P_B^0(b)$ $P_A^0(a)$ $P_B^1(b)$ $P_A^1(a)$

Construction of the type families

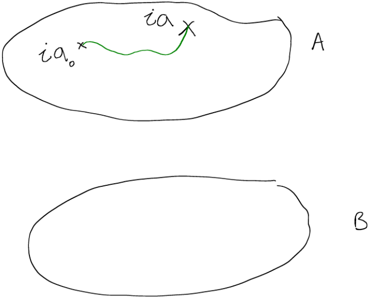
- $P_A(a)$ and $P_B(b)$ defined as sequential colimits
- $P_A^n(a) :=$ "type of paths from $i(a_0)$ to $i(a)$ allowing at most n "crossings" from B to A"
- $P_B^0(b) := \mathbb{0}$, $P_A^0(a) := (a_0 =_A a)$



$P_B^0(b)$ $P_A^0(a)$ $P_B^1(b)$ $P_A^1(a)$

Construction of the type families

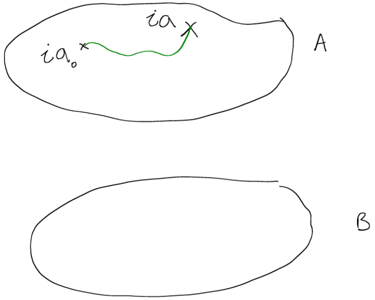
- $P_A(a)$ and $P_B(b)$ defined as sequential colimits
- $P_A^n(a)$:= "type of paths from $i(a_0)$ to $i(a)$ allowing at most n "crossings" from B to A"
- $P_B^0(b) := \mathbb{0}$, $P_A^0(a) := (a_0 =_A a)$



$P_B^0(b)$ $P_A^0(a)$ $P_B^1(b)$ $P_A^1(a)$

Construction of the type families

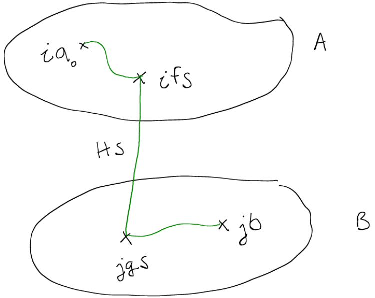
- $P_A(a)$ and $P_B(b)$ defined as sequential colimits
- $P_A^n(a) :=$ "type of paths from $i(a_0)$ to $i(a)$ allowing at most n "crossings" from B to A"
- $P_B^0(b) := \mathbb{0}$, $P_A^0(a) := (a_0 =_A a)$
- $P_A^{n+1}(a)$ is either $P_A^n(a)$, or $P_B^{n+1}(gs)$ and a path $fs =_A a$



$P_B^0(b)$ $P_A^0(a)$ $P_B^1(b)$ $P_A^1(a)$

Construction of the type families

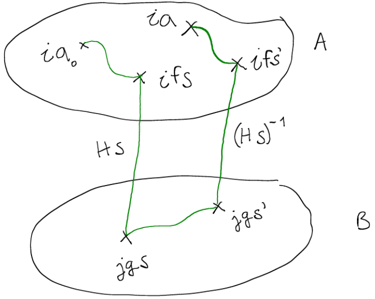
- $P_A(a)$ and $P_B(b)$ defined as sequential colimits
- $P_A^n(a)$:= "type of paths from $i(a_0)$ to $i(a)$ allowing at most n "crossings" from B to A"
- $P_B^0(b) :\equiv \mathbb{0}$, $P_A^0(a) :\equiv (a_0 =_A a)$
- $P_A^{n+1}(a)$ is either $P_A^n(a)$, or $P_B^{n+1}(gs)$ and a path $fs =_A a$



$P_B^0(b)$ $P_A^0(a)$ $P_B^1(b)$ $P_A^1(a)$

Construction of the type families

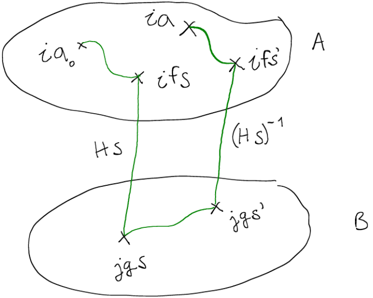
- $P_A(a)$ and $P_B(b)$ defined as sequential colimits
- $P^n_A(a) :=$ "type of paths from $i(a_0)$ to $i(a)$ allowing at most n "crossings" from B to A"
- $P^0_B(b) := \mathbb{0}$, $P^0_A(a) := (a_0 =_A a)$
- $P^{n+1}_A(a)$ is either $P^n_A(a)$, or $P^{n+1}_B(gs)$ and a path $fs =_A a$



$P^0_B(b)$ $P^0_A(a)$ $P^1_B(b)$ $P^1_A(a)$

Construction of the type families

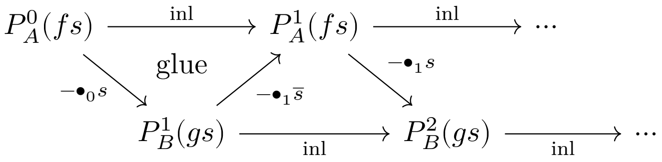
- $P_A(a)$ and $P_B(b)$ defined as sequential colimits
- $P_A^n(a)$:= "type of paths from $i(a_0)$ to $i(a)$ allowing at most n "crossings" from B to A"
- $P_B^0(b) := \mathbb{0}$, $P_A^0(a) := (a_0 =_A a)$
- $P_A^{n+1}(a)$ is either $P_A^n(a)$, or $P_B^{n+1}(gs)$ and a path $fs =_A a$
- "Modulo backtracking": $P_A^{n+1}(a)$ and $P_B^{n+1}(b)$ are pushouts



$P_B^0(b)$ $P_A^0(a)$ $P_B^1(b)$ $P_A^1(a)$

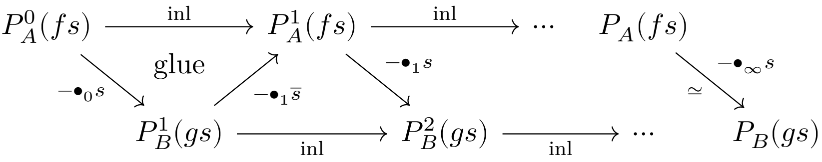
Construction of the equivalences

- To get $P_S(s)$, construct a zigzag between $P_A^\bullet(fs)$ and $P_B^\bullet(gs)$:



Construction of the equivalences

- To get $P_S(s)$, construct a zigzag between $P_A^\bullet(fs)$ and $P_B^\bullet(gs)$:



- The zigzag gives an equivalence $P_S(s) :\equiv - \bullet_\infty s$, completing (P_A, P_B, P_S), defining I_{x_0}

Identity systems

- To show $I_{x_0}(x) \simeq (x_0 = x)$, it suffices to show that I_{x_0} is an *identity system*

[3] Restatement of Kraus, von Raumer. Path Spaces of Higher Inductive Types in Homotopy Type Theory, 2019

Identity systems

- To show $I_{x_0}(x) \simeq (x_0 = x)$, it suffices to show that I_{x_0} is an *identity system*
- We can do that by showing that (P_A, P_B, P_S) is an identity system[3]

[3] Restatement of Kraus, von Raumer. Path Spaces of Higher Inductive Types in Homotopy Type Theory, 2019

Identity systems

- To show $I_{x_0}(x) \simeq (x_0 = x)$, it suffices to show that I_{x_0} is an *identity system*
- We can do that by showing that (P_A, P_B, P_S) is an identity system[3]

Definition (Induction principle of identity types of pushouts)

(P_A, P_B, P_S) with $p_0 : P_A(a_0)$ is an **identity system** if for all dependent descent data $\mathcal{Q} := (Q_A, Q_B, Q_S)$, the evaluation map $\text{ev-refl} : \text{sect}(\mathcal{Q}) \to Q_A(p_0)$ has a section.

[3] Restatement of Kraus, von Raumer. Path Spaces of Higher Inductive Types in Homotopy Type Theory, 2019

Identity systems

- To show $I_{x_0}(x) \simeq (x_0 = x)$, it suffices to show that I_{x_0} is an *identity system*
- We can do that by showing that (P_A, P_B, P_S) is an identity system[3]

Definition (Induction principle of identity types of pushouts)

(P_A, P_B, P_S) with $p_0 : P_A(a_0)$ is an **identity system** if for all dependent descent data $\mathcal{Q} := (Q_A, Q_B, Q_S)$, the evaluation map $\text{ev-refl} : \text{sect}(\mathcal{Q}) \to Q_A(p_0)$ has a section.

- Actually suffices to give a map $Q_A(p_0) \to \text{sect}(\mathcal{Q})$

[3] Restatement of Kraus, von Raumer. Path Spaces of Higher Inductive Types in Homotopy Type Theory, 2019

Zigzag construction is an identity system

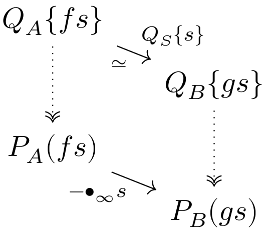
- $P_A(a_0)$ is pointed at $p_0 := \iota_A^0(\mathrm{refl}_{a_0})$

Zigzag construction is an identity system

- $P_A(a_0)$ is pointed at $p_0 :\equiv \iota^0_A(\mathrm{refl}_{a_0})$
- Given

$$Q_A\{a\} : P^\infty_A(a) \to \mathcal{U}$$
$$Q_B\{b\} : P^\infty_B(b) \to \mathcal{U}$$
$$Q_S\{s\} : (p : P^\infty_A(fs)) \to Q_A(p) \simeq Q_B(p \bullet_\infty s)$$

and $q_0 : Q_A(p_0)$



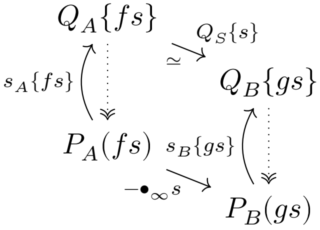
Zigzag construction is an identity system

- $P_A(a_0)$ is pointed at $p_0 := \iota_A^0(\mathrm{refl}_{a_0})$
- Given

$$Q_A\{a\} : P_A^\infty(a) \to \mathcal{U}$$
$$Q_B\{b\} : P_B^\infty(b) \to \mathcal{U}$$
$$Q_S\{s\} : (p : P_A^\infty(fs)) \to Q_A(p) \simeq Q_B(p \bullet_\infty s)$$

and $q_0 : Q_A(p_0)$, we need to produce

$$s_A\{a\} : (p : P_A^\infty(a)) \to Q_A(p)$$
$$s_B\{b\} : (p : P_B^\infty(b)) \to Q_B(p)$$
$$s_S\{s\} : (p : P_A^\infty(fs)) \to Q_S(s_A(p)) = s_B(p \bullet_\infty s).$$



Proof outline

- To get $s_A(a)$ and $s_B(b)$, do induction on the colimits $P_A(a)$ and $P_B(b)$: we need dependent functions

$$s_A^n\{a\} : (p : P_A^n(a)) \to Q_A(\iota_A^n(p))$$
$$s_B^n\{b\} : (p : P_B^n(b)) \to Q_B(\iota_B^n(p))$$

and coherences

Proof outline

- To get $s_A(a)$ and $s_B(b)$, do induction on the colimits $P_A(a)$ and $P_B(b)$: we need dependent functions

$$s_A^n\{a\} : (p : P_A^n(a)) \to Q_A(\iota_A^n(p))$$
$$s_B^n\{b\} : (p : P_B^n(b)) \to Q_B(\iota_B^n(p))$$

  and coherences
- The maps s_A^n and s_B^n are defined together by induction on n

Proof outline

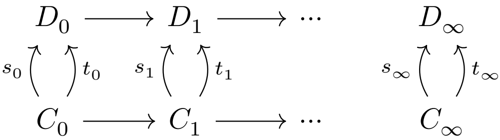
- To get $s_A(a)$ and $s_B(b)$, do induction on the colimits $P_A(a)$ and $P_B(b)$: we need dependent functions

$$s^n_A\{a\} : (p : P^n_A(a)) \to Q_A(\iota^n_A(p))$$
$$s^n_B\{b\} : (p : P^n_B(b)) \to Q_B(\iota^n_B(p))$$

  and coherences
- The maps s^n_A and s^n_B are defined together by induction on n
- Construct $s_S\{s\}$ by proving and using functoriality theorems for sequential colimits

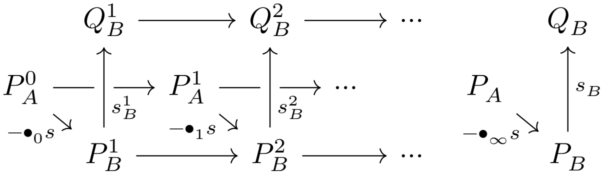
General results about sequential colimits

- "A homotopy of dependent diagram morphisms induces a homotopy of induced functions": $(s_\bullet \sim t_\bullet) \to (s_\infty \sim t_\infty)$



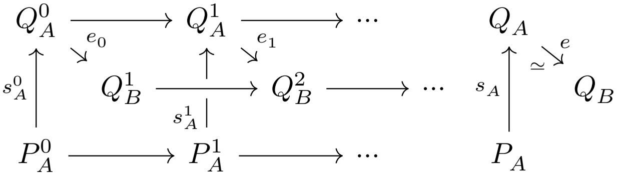
General results about sequential colimits

- "A homotopy of dependent diagram morphisms induces a homotopy of induced functions": $(s_\bullet \sim t_\bullet) \to (s_\infty \sim t_\infty)$
- "Taking the colimit preserves composition of a morphism and a dependent morphism": $s_\infty \circ f_\infty \sim (s_\bullet \circ f_\bullet)_\infty$ (*)



General results about sequential colimits

- "A homotopy of dependent diagram morphisms induces a homotopy of induced functions": $(s_\bullet \sim t_\bullet) \to (s_\infty \sim t_\infty)$
- "Taking the colimit preserves composition of a morphism and a dependent morphism": $s_\infty \circ f_\infty \sim (s_\bullet \circ f_\bullet)_\infty$ (*)
- "Taking the colimit preserves composition of a fiberwise morphism and a dependent morphism": $e_\infty \circ s_\infty \sim (e_\bullet \circ s_\bullet)_\infty$



General results about sequential colimits

- "A homotopy of dependent diagram morphisms induces a homotopy of induced functions": $(s_\bullet \sim t_\bullet) \to (s_\infty \sim t_\infty)$
- "Taking the colimit preserves composition of a morphism and a dependent morphism": $s_\infty \circ f_\infty \sim (s_\bullet \circ f_\bullet)_\infty$ (*)
- "Taking the colimit preserves composition of a fiberwise morphism and a dependent morphism": $e_\infty \circ s_\infty \sim (e_\bullet \circ s_\bullet)_\infty$
- "Dependent cubes induce a dependent square in the colimit":

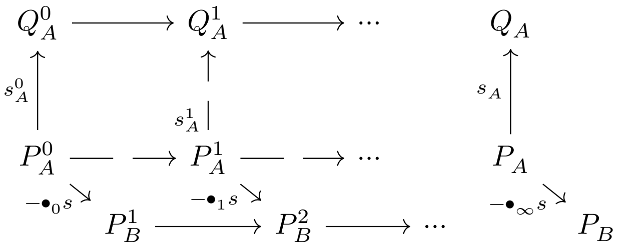
General results about sequential colimits

- "A homotopy of dependent diagram morphisms induces a homotopy of induced functions": $(s_\bullet \sim t_\bullet) \to (s_\infty \sim t_\infty)$
- "Taking the colimit preserves composition of a morphism and a dependent morphism": $s_\infty \circ f_\infty \sim (s_\bullet \circ f_\bullet)_\infty$ (*)
- "Taking the colimit preserves composition of a fiberwise morphism and a dependent morphism": $e_\infty \circ s_\infty \sim (e_\bullet \circ s_\bullet)_\infty$
- "Dependent cubes induce a dependent square in the colimit":

$$
\begin{array}{ccccccccc}
P_A^0 & \longrightarrow & P_A^1 & \longrightarrow & \cdots & & P_A & & \\
& \searrow^{-\bullet_0 s} & & \searrow^{-\bullet_1 s} & & & & \searrow^{-\bullet_\infty s} & \\
& & P_B^1 & \longrightarrow & P_B^2 & \longrightarrow \cdots & & & P_B
\end{array}
$$

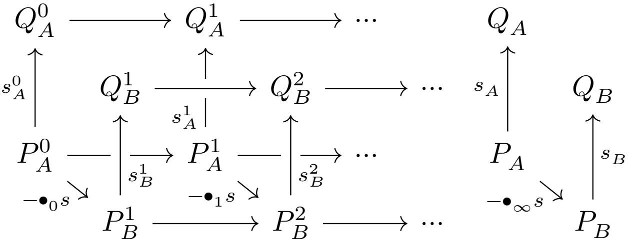
General results about sequential colimits

- "A homotopy of dependent diagram morphisms induces a homotopy of induced functions": $(s_\bullet \sim t_\bullet) \to (s_\infty \sim t_\infty)$
- "Taking the colimit preserves composition of a morphism and a dependent morphism": $s_\infty \circ f_\infty \sim (s_\bullet \circ f_\bullet)_\infty$ (*)
- "Taking the colimit preserves composition of a fiberwise morphism and a dependent morphism": $e_\infty \circ s_\infty \sim (e_\bullet \circ s_\bullet)_\infty$
- "Dependent cubes induce a dependent square in the colimit":



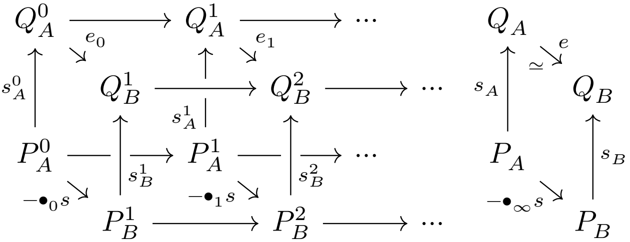
General results about sequential colimits

- "A homotopy of dependent diagram morphisms induces a homotopy of induced functions": $(s_\bullet \sim t_\bullet) \to (s_\infty \sim t_\infty)$
- "Taking the colimit preserves composition of a morphism and a dependent morphism": $s_\infty \circ f_\infty \sim (s_\bullet \circ f_\bullet)_\infty$ (*)
- "Taking the colimit preserves composition of a fiberwise morphism and a dependent morphism": $e_\infty \circ s_\infty \sim (e_\bullet \circ s_\bullet)_\infty$
- "Dependent cubes induce a dependent square in the colimit":



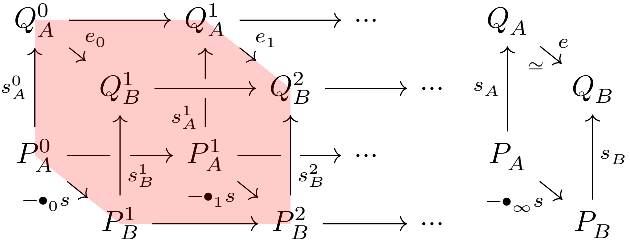
General results about sequential colimits

- "A homotopy of dependent diagram morphisms induces a homotopy of induced functions": $(s_\bullet \sim t_\bullet) \to (s_\infty \sim t_\infty)$
- "Taking the colimit preserves composition of a morphism and a dependent morphism": $s_\infty \circ f_\infty \sim (s_\bullet \circ f_\bullet)_\infty$ (*)
- "Taking the colimit preserves composition of a fiberwise morphism and a dependent morphism": $e_\infty \circ s_\infty \sim (e_\bullet \circ s_\bullet)_\infty$
- "Dependent cubes induce a dependent square in the colimit":



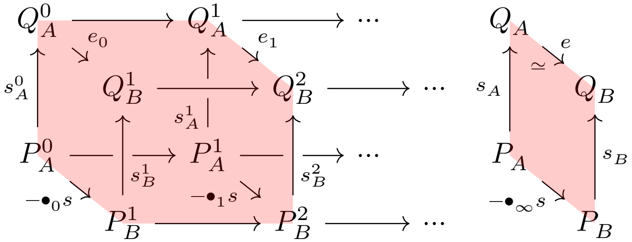
General results about sequential colimits

- "A homotopy of dependent diagram morphisms induces a homotopy of induced functions": $(s_\bullet \sim t_\bullet) \to (s_\infty \sim t_\infty)$
- "Taking the colimit preserves composition of a morphism and a dependent morphism": $s_\infty \circ f_\infty \sim (s_\bullet \circ f_\bullet)_\infty$ (*)
- "Taking the colimit preserves composition of a fiberwise morphism and a dependent morphism": $e_\infty \circ s_\infty \sim (e_\bullet \circ s_\bullet)_\infty$
- "Dependent cubes induce a dependent square in the colimit":



General results about sequential colimits

- "A homotopy of dependent diagram morphisms induces a homotopy of induced functions": $(s_\bullet \sim t_\bullet) \to (s_\infty \sim t_\infty)$
- "Taking the colimit preserves composition of a morphism and a dependent morphism": $s_\infty \circ f_\infty \sim (s_\bullet \circ f_\bullet)_\infty$ (*)
- "Taking the colimit preserves composition of a fiberwise morphism and a dependent morphism": $e_\infty \circ s_\infty \sim (e_\bullet \circ s_\bullet)_\infty$
- "Dependent cubes induce a dependent square in the colimit":



Conclusion

- The construction is formalized and proven correct
- No major issues with formalizing the construction itself
- Defining the sections s_A^n and s_B^n was more difficult, computation-wise
- Drawing the right diagrams helps finding statements of intermediate lemmas, but not so much proving them

Future work

- Formalizing applications
- Generalizing results about sequential colimits
- Optimizing: main file takes ~2.5 minutes[4], 11 GB of RAM, the rest of the library takes ~7.5 minutes; 93% spent in two definitions I didn't talk about

[4] Intel Core Ultra 7 155H

Related work

- Wärn's second article on the subject gives an "unstraightened" version of the construction
- Connors and Thorbjørnsen worked on an independent formalization in Rocq, at the time of writing the commutativity square needs to be formalized
- My master's thesis contains a more extensive description of the development of properties of various colimits leading to the formalization of the zigzag construction

Thank you for your time!

Construction of sections

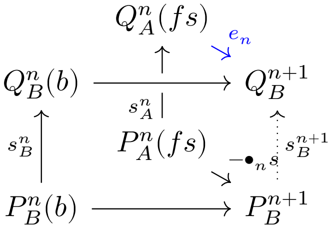
- $s_A^0(\mathrm{refl}) := q_0,\ s_B^0 := \text{ex-falso}$

Construction of sections

- $s^0_A(\text{refl}) := q_0$, $s^0_B :=$ ex-falso
- Next steps by pushout induction

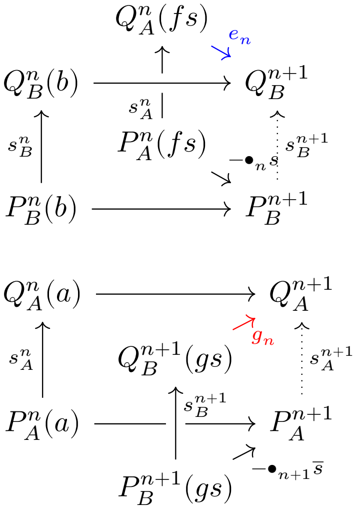
Construction of sections

- $s_A^0(\text{refl}) :\equiv q_0$, $s_B^0 :\equiv \text{ex-falso}$
- Next steps by pushout induction



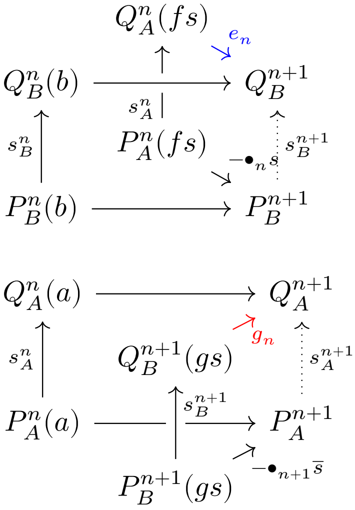
Construction of sections

- $s_A^0(\text{refl}) := q_0$, $s_B^0 :=$ ex-falso
- Next steps by pushout induction



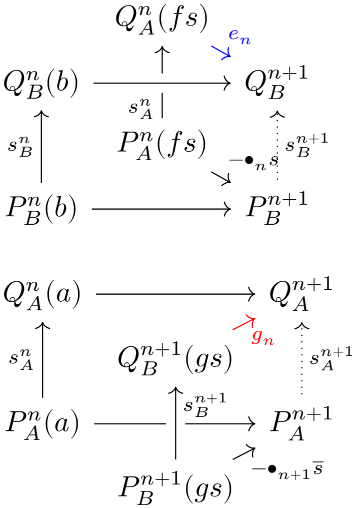
Construction of sections

- $s_A^0(\mathrm{refl}) := q_0,\ s_B^0 := \text{ex-falso}$
- Next steps by pushout induction
- Multiple choices for g_n



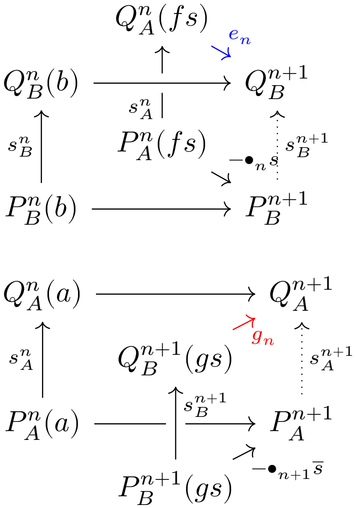
Construction of sections

- $s^0_A(\mathrm{refl}) := q_0$, $s^0_B :=$ ex-falso
- Next steps by pushout induction
- Multiple choices for g_n
- Action of s^{n+1}_B on glue_B using $e_n \circ e_n^{-1} \sim \mathrm{id}$



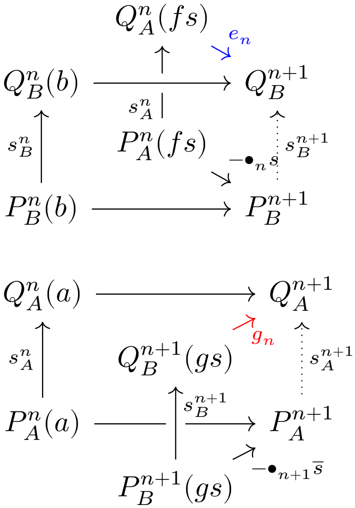
Construction of sections

- $s_A^0(\text{refl}) :\equiv q_0$, $s_B^0 :\equiv \text{ex-falso}$
- Next steps by pushout induction
- Multiple choices for g_n
- Action of s_B^{n+1} on glue_B using $e_n \circ e_n^{-1} \sim \text{id}$
- Action of s_A^{n+1} on glue_A using the square induced by Q_S and path algebra



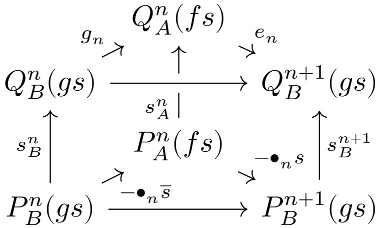
Construction of sections

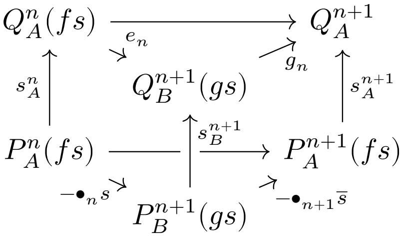
- $s_A^0(\text{refl}) := q_0$, $s_B^0 := \text{ex-falso}$
- Next steps by pushout induction
- Multiple choices for g_n
- Action of s_B^{n+1} on glue_B using $e_n \circ e_n^{-1} \sim \text{id}$
- Action of s_A^{n+1} on glue_A using the square induced by Q_S and path algebra
- K_A^n and K_B^n hold by computation rules for pushouts, completing s_A and s_B



Construction of the square

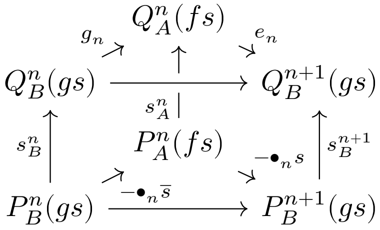
- The prisms can be filled using coherences of computation rules of pushouts

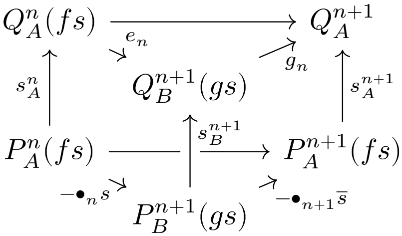




Construction of the square

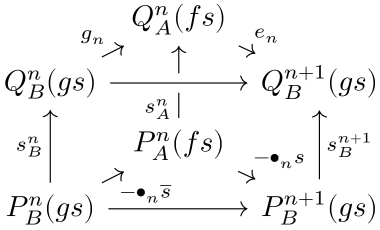
- The prisms can be filled using coherences of computation rules of pushouts
- Pasting the prisms along the diagonal gives almost the correct cubes, the top face needs adjustment

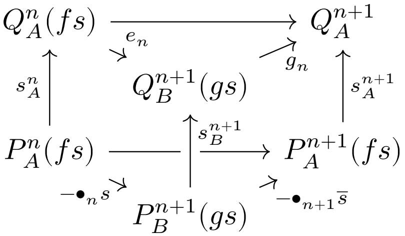




Construction of the square

- The prisms can be filled using coherences of computation rules of pushouts
- Pasting the prisms along the diagonal gives almost the correct cubes, the top face needs adjustment
- Proper abstraction and path induction fixes the top face





Construction of the square

- The prisms can be filled using coherences of computation rules of pushouts
- Pasting the prisms along the diagonal gives almost the correct cubes, the top face needs adjustment
- Proper abstraction and path induction fixes the top face
- The cubes induce the desired square s_S, which finishes the proof