Computer-generated proofs for the monoidal structure of the smash product

Guillaume Brunerie

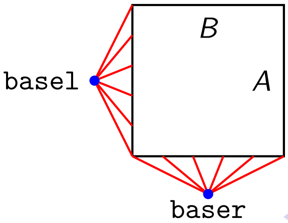
November 8, 2018
HoTTEST

The smash product as a higher inductive type

Definition

Given two pointed types $(A, \star_A)$ and $(B, \star_B)$, their smash product $A \wedge B$ is defined as the higher inductive type with constructors:

$$
\begin{aligned}
\mathtt{proj} &: A \times B \to A \wedge B, \\
\mathtt{basel} &: A \wedge B, \\
\mathtt{baser} &: A \wedge B, \\
\mathtt{pushl} &: (a : A) \to \mathtt{proj}(a, \star_B) = \mathtt{basel}, \\
\mathtt{pushr} &: (b : B) \to \mathtt{proj}(\star_A, b) = \mathtt{baser}.
\end{aligned}
$$



1-coherent monoidality

Goal

We want to prove (in book HoTT) that the smash product is a 1-coherent symmetric monoidal product on pointed types.[1]

This means that:

- The smash product is functorial (on pointed maps).
- There is a natural involution $\sigma_{A,B} : A \wedge B \to B \wedge A$.
- There is a natural equivalence $\alpha_{A,B,C} : (A \wedge B) \wedge C \to A \wedge (B \wedge C)$.
- It satisfies the hexagon and pentagon coherences.
- It has a unit with a triangular coherence.

This is used in particular to prove that the cup product on cohomology is associative.

[1] see pages 88 and 89 of my PhD thesis

Basic idea

All we have to do is to define various functions:

$$
\begin{aligned}
(x : A \wedge B) \to P(x) &\quad \text{(6 of them)} \\
(x : (A \wedge B) \wedge C) \to P(x) &\quad \text{(4 of them)} \\
(x : A \wedge (B \wedge C)) \to P(x) &\quad \text{(2 of them)} \\
(x : ((A \wedge B) \wedge C) \wedge D) \to P(x) &\quad \text{(1 of them)}
\end{aligned}
$$

where $P(x)$ is either constant or an equality $f(x) = g(x)$.

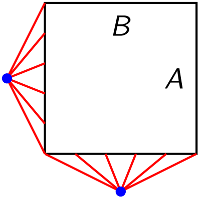
We define them by (iterated) induction on the smash product.

- In the (iterated) `proj` case, we know what to do.
- In the other cases, we "just" need to do some complicated path algebra.

Recursion rule

Given a type C, in order to define a map $f : A \wedge B \to C$, we need to define five terms/functions $f_{\mathtt{proj}}$, $f_{\mathtt{basel}}$, $f_{\mathtt{baser}}$, $f_{\mathtt{pushl}}$ and $f_{\mathtt{pushr}}$ such that:

$$
\begin{aligned}
f &: A \wedge B \to C \\
f(\mathtt{proj}(a,b)) &:\equiv f_{\mathtt{proj}}(a,b) && (f_{\mathtt{proj}} : A \times B \to C) \\
f(\mathtt{basel}) &:\equiv f_{\mathtt{basel}} && (f_{\mathtt{basel}} : C) \\
f(\mathtt{baser}) &:\equiv f_{\mathtt{baser}} && (f_{\mathtt{baser}} : C) \\
\mathsf{ap}_f(\mathtt{pushl}(a)) &:= f_{\mathtt{pushl}}(a) && (f_{\mathtt{pushl}} : (a : A) \to f_{\mathtt{proj}}(a, \star_B) =_C f_{\mathtt{basel}}) \\
\mathsf{ap}_f(\mathtt{pushr}(b)) &:= f_{\mathtt{pushr}}(b) && (f_{\mathtt{pushr}} : (b : B) \to f_{\mathtt{proj}}(\star_A, b) =_C f_{\mathtt{baser}})
\end{aligned}
$$



Example 1: commutativity

$$
\begin{aligned}
\sigma_{A,B} &: A \wedge B \to B \wedge A \\
\sigma_{A,B}(\mathtt{proj}(a, b)) &:= \mathtt{proj}(b, a) \\
\sigma_{A,B}(\mathtt{basel}) &:= \blacksquare^0 \\
\sigma_{A,B}(\mathtt{baser}) &:= \blacksquare^0 \\
\mathsf{ap}_{\sigma_{A,B}}(\mathtt{pushl}(a)) &:= \blacksquare^1 \\
\mathsf{ap}_{\sigma_{A,B}}(\mathtt{pushr}(b)) &:= \blacksquare^1
\end{aligned}
$$

Example 1: commutativity

$$
\begin{aligned}
\sigma_{A,B} &: A \wedge B \to B \wedge A \\
\sigma_{A,B}(\mathtt{proj}(a, b)) &:= \mathtt{proj}(b, a) \\
\sigma_{A,B}(\mathtt{basel}) &:= \mathtt{baser} \\
\sigma_{A,B}(\mathtt{baser}) &:= \mathtt{basel} \\
\mathsf{ap}_{\sigma_{A,B}}(\mathtt{pushl}(a)) &:= \mathtt{pushr}(a) \\
\mathsf{ap}_{\sigma_{A,B}}(\mathtt{pushr}(b)) &:= \mathtt{pushl}(b)
\end{aligned}
$$

Pointed maps

Definition

Given two pointed types $(A, \star_A)$ and $(A', \star_{A'})$, a pointed map from A to A' is a pair $(f, \star_f)$ where

$$f : A \to A'$$
$$\star_f : f(\star_A) = \star_{A'}$$

Example 2: functoriality

We have two pointed maps $f : A \to A'$ and $g : B \to B'$.

$$
\begin{aligned}
(f \wedge g) &: A \wedge B \to A' \wedge B' \\
(f \wedge g)(\mathtt{proj}(a, b)) &:= \mathtt{proj}(f(a), g(b)) \\
(f \wedge g)(\mathtt{basel}) &:= \mathtt{basel} \\
(f \wedge g)(\mathtt{baser}) &:= \mathtt{baser} \\
\mathsf{ap}_{f \wedge g}(\mathtt{pushl}(a)) &:= \blacksquare^1 \\
\mathsf{ap}_{f \wedge g}(\mathtt{pushr}(b)) &:= \blacksquare^1
\end{aligned}
$$

Example 2: functoriality

We have two pointed maps $f : A \to A'$ and $g : B \to B'$.

$$
\begin{aligned}
(f \wedge g) &: A \wedge B \to A' \wedge B' \\
(f \wedge g)(\mathtt{proj}(a, b)) &:= \mathtt{proj}(f(a), g(b)) \\
(f \wedge g)(\mathtt{basel}) &:= \mathtt{basel} \\
(f \wedge g)(\mathtt{baser}) &:= \mathtt{baser} \\
\mathsf{ap}_{f \wedge g}(\mathtt{pushl}(a)) &:= \blacksquare^1 \\
\mathsf{ap}_{f \wedge g}(\mathtt{pushr}(b)) &:= \blacksquare^1
\end{aligned}
$$

The two holes have type

$$\mathtt{proj}(f(a), g(\star_B)) = \mathtt{basel} \qquad \mathtt{proj}(f(\star_A), g(b)) = \mathtt{baser}$$

... with rewriting

We have

$$\begin{aligned} &\mathtt{proj}(f(a), g(\star_B)) \\ &\rightsquigarrow \mathtt{proj}(f(a), \star_{B'}) \quad \text{via} \quad \star_g \text{ in the second argument of } \mathtt{proj} \\ &\rightsquigarrow \mathtt{basel} \quad \text{via} \quad \mathtt{pushl}(f(a)) \end{aligned}$$

Therefore we can fill the first hole with

$$\mathsf{ap}_{\mathtt{proj}(f(a),-)}(\star_g) \bullet \mathtt{pushl}(f(a))$$

. . . with rewriting

We have

$$\begin{aligned}
&\mathtt{proj}(f(a), g(\star_B)) \\
&\rightsquigarrow \mathtt{proj}(f(a), \star_{B'}) \quad \text{via} \quad \star_g \ \text{ in the second argument of } \mathtt{proj} \\
&\rightsquigarrow \mathtt{basel} \quad \text{via} \quad \mathtt{pushl}(f(a))
\end{aligned}$$

Therefore we can fill the first hole with

$$\mathsf{ap}_{\mathtt{proj}(f(a),-)}(\star_g) \bullet \mathtt{pushl}(f(a))$$

Similarly for the second hole:

$$\begin{aligned}
&\mathtt{proj}(f(\star_A), g(b)) \\
&\rightsquigarrow \mathtt{proj}(\star_{A'}, g(b)) \quad \text{via} \quad \star_f \ \text{ in the first argument of } \mathtt{proj} \\
&\rightsquigarrow \mathtt{baser} \quad \text{via} \quad \mathtt{pushr}(g(b))
\end{aligned}$$

Proof-relevant rewriting

$$
\begin{array}{rcll}
f(\star_A) & \rightsquigarrow & \star_{A'} & \text{via} \quad \star_f \\
\mathtt{proj}(a, \star_B) & \rightsquigarrow & \mathtt{basel} & \text{via} \quad \mathtt{pushl}(a) \\
\mathtt{proj}(\star_A, b) & \rightsquigarrow & \mathtt{baser} & \text{via} \quad \mathtt{pushr}(b) \\
\mathtt{basel} & \rightsquigarrow & \mathtt{proj}(\star_A, \star_B) & \text{via} \quad \mathtt{pushl}(\star_A) \\
\mathtt{baser} & \rightsquigarrow & \mathtt{proj}(\star_A, \star_B) & \text{via} \quad \mathtt{pushr}(\star_B) \\
\mathtt{proj}(\star_A, \star_B) & \not\rightsquigarrow & &
\end{array}
$$

$$
\begin{array}{rcll}
f(u) & \rightsquigarrow & f(u') & \text{via} \quad \mathsf{ap}_f(p) \\
 & & & (\text{if } u \rightsquigarrow u' \text{ via } p) \\
u & \rightsquigarrow & u'' & \text{via} \quad p \bullet p' \\
 & & & (\text{if } u \rightsquigarrow u' \text{ via } p \text{ and } u' \rightsquigarrow u'' \text{ via } p')
\end{array}
$$

Squares

We use squares and cubes in the sense of [LB15][2] .

Definition

The type

$$\texttt{Square} : \{A : \texttt{Type}\}\{a, b, c, d : A\}$$
$$(p : a = b)(q : c = d)(r : a = c)(s : b = d) \to \texttt{Type}$$

is defined as the inductive family with one constructor

$$\texttt{ids} : \texttt{Square}(\texttt{idp}, \texttt{idp}, \texttt{idp}, \texttt{idp})$$

[2] D. Licata, G. Brunerie, *A Cubical Approach to Synthetic Homotopy Theory*, LICS 2015

Application of a homotopy to a path

Given a dependent function (where $g, h : A \to B$)

$$f : (x : A) \to g(x) =_B h(x)$$

and a path

$$p : a =_A a'$$

we have

$$\mathsf{ap}_f^+(p) : \mathtt{Square}(\mathsf{ap}_g(p), \mathsf{ap}_h(p), f(a), f(a'))$$

$$\begin{tikzcd}
g(a) \arrow[r, dash, "f(a)"] \arrow[d, dash, "\mathsf{ap}_g(p)"'] & h(a) \arrow[d, dash, "\mathsf{ap}_h(p)"] \\
g(a') \arrow[r, dash, "f(a')"'] & h(a')
\end{tikzcd}$$

Induction rule (into an identity type)

Given a type C and two functions $g, h : A \wedge B \to C$, in order to define a map

$$f : (x : A \wedge B) \to g(x) =_C h(x),$$

we need

$$\begin{aligned}
f(\mathtt{proj}(a,b)) &: g(\mathtt{proj}(a,b)) =_C h(\mathtt{proj}(a,b)) \\
f(\mathtt{basel}) &: g(\mathtt{basel}) =_C h(\mathtt{basel}) \\
f(\mathtt{baser}) &: g(\mathtt{baser}) =_C h(\mathtt{baser}) \\
\mathsf{ap}_f^+(\mathtt{pushl}(a)) &: \mathsf{Square}(\mathsf{ap}_g(\mathtt{pushl}(a)), \mathsf{ap}_h(\mathtt{pushl}(a)), \\
&\qquad\qquad f(\mathtt{proj}(a, \star_B)), f(\mathtt{basel})) \\
\mathsf{ap}_f^+(\mathtt{pushr}(b)) &: \mathsf{Square}(\mathsf{ap}_g(\mathtt{pushr}(b)), \mathsf{ap}_h(\mathtt{pushr}(b)), \\
&\qquad\qquad f(\mathtt{proj}(\star_A, b)), f(\mathtt{baser}))
\end{aligned}$$

Example 2: naturality of commutativity

We have two pointed maps $f : A \to A'$ and $g : B \to B'$.

$$\sigma\text{-}\mathtt{nat}_{f,g} : (x : A \wedge B) \to \sigma_{A',B'}((f \wedge g)(x)) = (g \wedge f)(\sigma_{A,B}(x))$$

$$\begin{aligned}
\sigma\text{-}\mathtt{nat}_{f,g}(\mathtt{proj}(a,b)) &:= \mathtt{idp}_{\mathtt{proj}(g(b),f(a))} \\
\sigma\text{-}\mathtt{nat}_{f,g}(\mathtt{basel}) &:= \mathtt{idp}_{\mathtt{baser}} \\
\sigma\text{-}\mathtt{nat}_{f,g}(\mathtt{baser}) &:= \mathtt{idp}_{\mathtt{basel}} \\
\mathsf{ap}^{+}_{\sigma\text{-}\mathtt{nat}_{f,g}}(\mathtt{pushl}(a)) &:= \blacksquare^2 : \mathsf{Square}(\mathsf{ap}_{\sigma_{A',B'} \circ (f \wedge g)}(\mathtt{pushl}(a)), \\
&\qquad\qquad \mathsf{ap}_{(g \wedge f) \circ \sigma_{A,B}}(\mathtt{pushl}(a)), \\
&\qquad\qquad \mathtt{idp}_{\mathtt{proj}(g(\star_B),f(a))}, \\
&\qquad\qquad \mathtt{idp}_{\mathtt{baser}}) \\
\mathsf{ap}^{+}_{\sigma\text{-}\mathtt{nat}_{f,g}}(\mathtt{pushr}(b)) &:= \blacksquare^2 : [\ldots]
\end{aligned}$$

More rewriting!

$\mathsf{ap}_{\sigma_{A',B'}\circ(f\wedge g)}(\mathtt{pushl}(a))$

More rewriting!

$$\mathsf{ap}_{\sigma_{A',B'}\circ(f\wedge g)}(\mathtt{pushl}(a))$$
$$\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{f\wedge g}(\mathtt{pushl}(a)))$$

More rewriting!

$$\begin{aligned}
&\mathsf{ap}_{\sigma_{A',B'}\circ(f\wedge g)}(\texttt{pushl}(a))\\
&\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{f\wedge g}(\texttt{pushl}(a)))\\
&\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{\texttt{proj}(f(a),-)}(\star_g)\bullet\texttt{pushl}(f(a)))
\end{aligned}$$

More rewriting!

$$
\begin{aligned}
&\mathsf{ap}_{\sigma_{A',B'}\circ(f\wedge g)}(\mathtt{pushl}(a))\\
&\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{f\wedge g}(\mathtt{pushl}(a)))\\
&\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{\mathtt{proj}(f(a),-)}(\star_g)\bullet\mathtt{pushl}(f(a)))\\
&\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{\mathtt{proj}(f(a),-)}(\star_g))\bullet\mathsf{ap}_{\sigma_{A',B'}}(\mathtt{pushl}(f(a)))
\end{aligned}
$$

More rewriting!

$$
\begin{aligned}
&\mathsf{ap}_{\sigma_{A',B'}\circ(f\wedge g)}(\mathtt{pushl}(a))\\
&\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{f\wedge g}(\mathtt{pushl}(a)))\\
&\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{\mathtt{proj}(f(a),-)}(\star_g)\bullet \mathtt{pushl}(f(a)))\\
&\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{\mathtt{proj}(f(a),-)}(\star_g))\bullet \mathsf{ap}_{\sigma_{A',B'}}(\mathtt{pushl}(f(a)))\\
&\rightsquigarrow \mathsf{ap}_{\mathtt{proj}(-,f(a))}(\star_g)\bullet \mathtt{pushr}(f(a))
\end{aligned}
$$

More rewriting!

$$
\begin{aligned}
&\mathsf{ap}_{\sigma_{A',B'}\circ(f\wedge g)}(\mathtt{pushl}(a))\\
&\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{f\wedge g}(\mathtt{pushl}(a)))\\
&\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{\mathtt{proj}(f(a),-)}(\star_g)\bullet\mathtt{pushl}(f(a)))\\
&\rightsquigarrow \mathsf{ap}_{\sigma_{A',B'}}(\mathsf{ap}_{\mathtt{proj}(f(a),-)}(\star_g))\bullet\mathsf{ap}_{\sigma_{A',B'}}(\mathtt{pushl}(f(a)))\\
&\rightsquigarrow \mathsf{ap}_{\mathtt{proj}(-,f(a))}(\star_g)\bullet\mathtt{pushr}(f(a))
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{ap}_{(g\wedge f)\circ\sigma_{A,B}}(\mathtt{pushl}(a))\\
&\rightsquigarrow \mathsf{ap}_{g\wedge f}(\mathsf{ap}_{\sigma_{A,B}}(\mathtt{pushl}(a)))\\
&\rightsquigarrow \mathsf{ap}_{g\wedge f}(\mathtt{pushr}(a))\\
&\rightsquigarrow \mathsf{ap}_{\mathtt{proj}(-,f(a))}(\star_g)\bullet\mathtt{pushr}(f(a))
\end{aligned}
$$

More rewriting rules

$$
\begin{aligned}
\mathsf{ap}_{\sigma_{A,B}}(\mathtt{pushl}(a)) &\rightsquigarrow \mathtt{pushr}(a) && \text{and other } \beta\text{-reduction rules for HITs} \\
\mathsf{ap}_{\lambda x.x}(p) &\rightsquigarrow p \\
\mathsf{ap}_g(\mathsf{ap}_f(p)) &\rightsquigarrow \mathsf{ap}_{g\circ f}(p) \\
\mathsf{ap}_{g\circ f}(p) &\rightsquigarrow \mathsf{ap}_g(p') && (\text{if } \mathsf{ap}_f(p) \rightsquigarrow p') \\
\mathsf{ap}_f(u \bullet v) &\rightsquigarrow \mathsf{ap}_f(u) \bullet \mathsf{ap}_f(v) \\
u \bullet v &\rightsquigarrow u' \bullet v' && (\text{if } u \rightsquigarrow u' \text{ and } v \rightsquigarrow v', \\
& && \quad \text{via horizontal composition})
\end{aligned}
$$

Example 4: associativity

$$\alpha_{A,B,C} : (A \wedge B) \wedge C \to A \wedge (B \wedge C),$$

$$\alpha_{A,B,C}(\mathtt{proj}(x, c)) :\equiv \alpha^{\mathtt{proj}}_{A,B,C}(x, c),$$

$$\alpha_{A,B,C}(\mathtt{basel}) :\equiv \blacksquare^{0},$$

$$\alpha_{A,B,C}(\mathtt{baser}) :\equiv \blacksquare^{0},$$

$$\mathsf{ap}_{\alpha_{A,B,C}}(\mathtt{pushl}(x)) :\equiv \alpha^{\mathtt{pushl}}_{A,B,C}(x),$$

$$\mathsf{ap}_{\alpha_{A,B,C}}(\mathtt{pushr}(c)) :\equiv \blacksquare^{1}.$$

Example 4: associativity

$$
\begin{aligned}
\alpha_{A,B,C} &: (A \wedge B) \wedge C \to A \wedge (B \wedge C),\\
\alpha_{A,B,C}(\mathtt{proj}(x, c)) &:= \alpha^{\mathtt{proj}}_{A,B,C}(x, c),\\
\alpha_{A,B,C}(\mathtt{basel}) &:= \blacksquare^{0},\\
\alpha_{A,B,C}(\mathtt{baser}) &:= \blacksquare^{0},\\
\mathsf{ap}_{\alpha_{A,B,C}}(\mathtt{pushl}(x)) &:= \alpha^{\mathtt{pushl}}_{A,B,C}(x),\\
\mathsf{ap}_{\alpha_{A,B,C}}(\mathtt{pushr}(c)) &:= \blacksquare^{1}.
\end{aligned}
$$

$$
\begin{aligned}
\alpha^{\mathtt{proj}}_{A,B,C} &: A \wedge B \to C \to A \wedge (B \wedge C),\\
\alpha^{\mathtt{proj}}_{A,B,C}(\mathtt{proj}(a, b), c) &:= \mathtt{proj}(a, \mathtt{proj}(b, c)),\\
&\quad [\blacksquare^{0} \dots \blacksquare^{0} \dots \blacksquare^{1} \dots \blacksquare^{1}]
\end{aligned}
$$

Example 4: associativity

$$\begin{aligned}
\alpha_{A,B,C} &: (A \wedge B) \wedge C \to A \wedge (B \wedge C), \\
\alpha_{A,B,C}(\mathtt{proj}(x, c)) &:= \alpha^{\mathtt{proj}}_{A,B,C}(x, c), \\
\alpha_{A,B,C}(\mathtt{basel}) &:= \blacksquare^0, \\
\alpha_{A,B,C}(\mathtt{baser}) &:= \blacksquare^0, \\
\mathsf{ap}_{\alpha_{A,B,C}}(\mathtt{pushl}(x)) &:= \alpha^{\mathtt{pushl}}_{A,B,C}(x), \\
\mathsf{ap}_{\alpha_{A,B,C}}(\mathtt{pushr}(c)) &:= \blacksquare^1.
\end{aligned}$$

$$\begin{aligned}
\alpha^{\mathtt{proj}}_{A,B,C} &: A \wedge B \to C \to A \wedge (B \wedge C), \\
\alpha^{\mathtt{proj}}_{A,B,C}(\mathtt{proj}(a, b), c) &:= \mathtt{proj}(a, \mathtt{proj}(b, c)), \\
& [\blacksquare^0 \ldots \blacksquare^0 \ldots \blacksquare^1 \ldots \blacksquare^1]
\end{aligned}$$

$$\begin{aligned}
\alpha^{\mathtt{pushl}}_{A,B,C} &: (x : A \wedge B) \to \alpha^{\mathtt{proj}}_{A,B,C}(x, \star_C) = \alpha_{A,B,C}(\mathtt{basel}), \\
& [\blacksquare^1 \ldots \blacksquare^1 \ldots \blacksquare^1 \ldots \blacksquare^2 \ldots \blacksquare^2]
\end{aligned}$$

Hexagon

$$
\begin{tikzcd}
 & (A \wedge B) \wedge C \arrow[r, "\alpha_{A,B,C}"] \arrow[dl, "\sigma_{A,B} \wedge \mathrm{id}_C"'] & A \wedge (B \wedge C) \arrow[dr, "\sigma_{A,B\wedge C}"] & \\
(B \wedge A) \wedge C \arrow[dr, "\alpha_{B,A,C}"'] & & & (B \wedge C) \wedge A \arrow[dl, "\alpha_{B,C,A}"] \\
 & B \wedge (A \wedge C) \arrow[r, "\mathrm{id}_B \wedge \sigma_{A,C}"'] & B \wedge (C \wedge A) &
\end{tikzcd}
$$

Example 5: hexagon

$$\texttt{hexagon}_{A,B,C} : (x : (A \wedge B) \wedge C) \to \texttt{Id}(\dots,\dots)$$
$$[\texttt{hexagon}^{\texttt{proj}}_{A,B,C} \dots \blacksquare^{1} \dots \blacksquare^{1} \dots \texttt{hexagon}^{\texttt{pushl}}_{A,B,C} \dots \blacksquare^{2}]$$

Example 5: hexagon

$$\texttt{hexagon}_{A,B,C} : (x : (A \wedge B) \wedge C) \to \texttt{Id}(\ldots, \ldots)$$
$$[\texttt{hexagon}^{\texttt{proj}}_{A,B,C} \ldots {\color{blue}\blacksquare}^{\color{blue}1} \ldots {\color{blue}\blacksquare}^{\color{blue}1} \ldots \texttt{hexagon}^{\texttt{pushl}}_{A,B,C} \ldots {\color{red}\blacksquare}^{\color{red}2}]$$

$$\texttt{hexagon}^{\texttt{proj}}_{A,B,C} : (x : A \wedge B) \to C \to \texttt{Id}(\ldots, \ldots)$$
$$[\texttt{idp} \ldots {\color{blue}\blacksquare}^{\color{blue}1} \ldots {\color{blue}\blacksquare}^{\color{blue}1} \ldots {\color{red}\blacksquare}^{\color{red}2} \ldots {\color{red}\blacksquare}^{\color{red}2}]$$

Example 5: hexagon

$$\texttt{hexagon}_{A,B,C} : (x : (A \wedge B) \wedge C) \to \texttt{Id}(\ldots, \ldots)$$
$$[\texttt{hexagon}^{\texttt{proj}}_{A,B,C} \ldots {\color{blue}\blacksquare^{1}} \ldots {\color{blue}\blacksquare^{1}} \ldots \texttt{hexagon}^{\texttt{pushl}}_{A,B,C} \ldots {\color{red}\blacksquare^{2}}]$$

$$\texttt{hexagon}^{\texttt{proj}}_{A,B,C} : (x : A \wedge B) \to C \to \texttt{Id}(\ldots, \ldots)$$
$$[\texttt{idp} \ldots {\color{blue}\blacksquare^{1}} \ldots {\color{blue}\blacksquare^{1}} \ldots {\color{red}\blacksquare^{2}} \ldots {\color{red}\blacksquare^{2}}]$$

$$\texttt{hexagon}^{\texttt{pushl}}_{A,B,C} : (x : A \wedge B) \to C \to \texttt{Square}(\ldots, \ldots, \ldots, \ldots)$$
$$[{\color{red}\blacksquare^{2}} \ldots {\color{red}\blacksquare^{2}} \ldots {\color{red}\blacksquare^{2}} \ldots \blacksquare^{3} \ldots \blacksquare^{3}]$$

Example 6: pentagon

$$\texttt{pent} : (x : ((A \wedge B) \wedge C) \wedge D) \to \texttt{Id}(\ldots, \ldots)$$

Example 6: pentagon

$$
\begin{aligned}
\texttt{pent} &: (x : ((A \wedge B) \wedge C) \wedge D) \to \texttt{Id}(\dots, \dots) \\
\texttt{pent} &: [\texttt{pent}^{\texttt{proj}} \dots \blacksquare^{1} \dots \blacksquare^{1} \dots \texttt{pent}^{\texttt{pushl}} \dots \blacksquare^{2}] \\
\texttt{pent}^{\texttt{proj}} &: [\texttt{pent}^{\texttt{proj},\texttt{proj}} \dots \blacksquare^{1} \dots \blacksquare^{1} \dots \texttt{pent}^{\texttt{proj},\texttt{pushl}} \dots \blacksquare^{2}] \\
\texttt{pent}^{\texttt{proj},\texttt{proj}} &: [\texttt{idp} \dots \blacksquare^{1} \dots \blacksquare^{1} \dots \blacksquare^{2} \dots \blacksquare^{2}] \\
\texttt{pent}^{\texttt{proj},\texttt{pushl}} &: [\blacksquare^{2} \dots \blacksquare^{2} \dots \blacksquare^{2} \dots \blacksquare^{3} \dots \blacksquare^{3}] \\
\texttt{pent}^{\texttt{pushl}} &: [\texttt{pent}^{\texttt{pushl},\texttt{proj}} \dots \blacksquare^{2} \dots \blacksquare^{2} \dots \texttt{pent}^{\texttt{pushl},\texttt{pushl}} \dots \blacksquare^{3}] \\
\texttt{pent}^{\texttt{pushl},\texttt{proj}} &: [\blacksquare^{2} \dots \blacksquare^{2} \dots \blacksquare^{2} \dots \blacksquare^{3} \dots \blacksquare^{3}] \\
\texttt{pent}^{\texttt{pushl},\texttt{pushl}} &: [\blacksquare^{3} \dots \blacksquare^{3} \dots \blacksquare^{3} \dots \blacksquare^{4} \dots \blacksquare^{4}]
\end{aligned}
$$

Cubical proof-relevant rewriting

Definition

Given $f : A \to B$ and $sq : \mathtt{Square}_A(p, q, r, s)$, we have

$$\mathsf{ap}_f^2(sq) : \mathtt{Square}_B(\mathsf{ap}_f(p), \mathsf{ap}_f(q), \mathsf{ap}_f(r), \mathsf{ap}_f(s)).$$

Cubical proof-relevant rewriting

Definition

Given $f : A \to B$ and $sq : \mathtt{Square}_A(p, q, r, s)$, we have

$$\mathsf{ap}^2_f(sq) : \mathtt{Square}_B(\mathsf{ap}_f(p), \mathsf{ap}_f(q), \mathsf{ap}_f(r), \mathsf{ap}_f(s)).$$

We want a reduction rule

$$\mathsf{ap}^2_{\lambda x.x}(sq) \rightsquigarrow sq$$

Cubical proof-relevant rewriting

Definition

Given $f : A \to B$ and $sq : \mathtt{Square}_A(p, q, r, s)$, we have

$$\mathsf{ap}^2_f(sq) : \mathtt{Square}_B(\mathsf{ap}_f(p), \mathsf{ap}_f(q), \mathsf{ap}_f(r), \mathsf{ap}_f(s)).$$

We want a reduction rule

$$\mathsf{ap}^2_{\lambda x.x}(sq) \rightsquigarrow sq$$

... but the two sides don't have the same type (!).

Cubical proof-relevant rewriting

Definition

Given $f : A \to B$ and $sq : \texttt{Square}_A(p, q, r, s)$, we have

$$\mathsf{ap}^2_f(sq) : \texttt{Square}_B(\mathsf{ap}_f(p), \mathsf{ap}_f(q), \mathsf{ap}_f(r), \mathsf{ap}_f(s)).$$

We want a reduction rule

$$\mathsf{ap}^2_{\lambda x.x}(sq) \rightsquigarrow sq$$

... but the two sides don't have the same type (!).

Cubical proof-relevant rewriting:
If s and s' are two squares, we say

$$s \rightsquigarrow s' \quad \text{via} \quad c$$

if c is a cube with s and s' as two of its opposite faces.

More variants of ap

	$p : \mathtt{Id}_A$	$p : \mathtt{Square}_A$	$p : \mathtt{Cube}_A$
$f : A \to B$	$\mathsf{ap}_f(p)$	$\mathsf{ap}^2_f(p)$	
$f : A \to \mathtt{Id}_B$	$\mathsf{ap}^+_f(p)$		
$f : A \to \mathtt{Square}_B$			
$f : A \to \mathtt{Cube}_B$			

More variants of ap

	$p : \mathtt{Id}_A$	$p : \mathtt{Square}_A$	$p : \mathtt{Cube}_A$
$f : A \to B$	$\mathsf{ap}_f(p)$	$\mathsf{ap}^2_f(p)$	$\mathsf{ap}^3_f(p)$
$f : A \to \mathtt{Id}_B$	$\mathsf{ap}^{+}_f(p)$	$\mathsf{ap}^{2,+}_f(p)$	$\mathsf{ap}^{3,+}_f(p)$
$f : A \to \mathtt{Square}_B$	$\mathsf{ap}^{++}_f(p)$	$\mathsf{ap}^{2,++}_f(p)$	
$f : A \to \mathtt{Cube}_B$	$\mathsf{ap}^{+++}_f(p)$		

More variants of ap

	$p : \mathtt{Id}_A$	$p : \mathtt{Square}_A$	$p : \mathtt{Cube}_A$
$f : A \to B$	$\mathsf{ap}_f(p)$	$\mathsf{ap}^2_f(p)$	$\mathsf{ap}^3_f(p)$
$f : A \to \mathtt{Id}_B$	$\mathsf{ap}^+_f(p)$	$\mathsf{ap}^{2,+}_f(p)$	$\mathsf{ap}^{3,+}_f(p)$
$f : A \to \mathtt{Square}_B$	$\mathsf{ap}^{++}_f(p)$	$\mathsf{ap}^{2,++}_f(p)$	
$f : A \to \mathtt{Cube}_B$	$\mathsf{ap}^{+++}_f(p)$		

They interact in various ways, for instance

$$
\begin{aligned}
\mathsf{ap}^2_g(\mathsf{ap}^+_f(p)) &\rightsquigarrow \mathsf{ap}^+_{\mathsf{ap}_g \circ f}(p) \\
\mathsf{ap}^+_g(\mathsf{ap}_f(p)) &\rightsquigarrow \mathsf{ap}^+_{g \circ f}(p) \\
\mathsf{ap}^+_f(p \bullet q) &\rightsquigarrow \mathsf{ap}^+_f(p) \diamond \mathsf{ap}^+_f(q) \\
\mathsf{ap}^+_{\lambda x.p(x) \bullet q(x)}(r) &\rightsquigarrow \mathsf{ap}^+_{\lambda x.p(x)}(r) \blacklozenge \mathsf{ap}^+_{\lambda x.q(x)}(r)
\end{aligned}
$$

Globular coherences

We can construct any map of the form:

$$\begin{aligned}\mathtt{coh} : (X : \mathtt{Type})&(a : X)\\ &[\ldots]\\ &(x_n : T_n)(p_n : x_n = u_n)\\ &[\ldots]\\ &\to T\end{aligned}$$

where T_n, u_n and T are built only from previous variables and other coherences, and T is an identity type.

Idea: path induction on all of the p_n, then give `idp`.

Use: p_n represents a rewriting rule, and x_n the term being rewritten.

Cubical coherences

We also need to allow pairs of arguments of the form

$$(x_n : T_n)(p_n : \mathtt{Square}(x_n, u_n, v_n, w_n))$$

$$(x_n : T_n)(p_n : \mathtt{Cube}(x_n, u_n, v_n, w_n, r_n, s_n))$$

We can still construct all such coherences, using a generalized version of J where three sides of a square are fixed and one side is free.

Algorithm for building the proof

In order to fill a hole ($\blacksquare^1$, $\blacksquare^2$, $\blacksquare^3$ or $\blacksquare^4$) we proceed as follows.
The variables are ℓ_1 a list of terms and ℓ_2 a list of pairs of terms.

- We start with ℓ_1 consisting of all the faces (in every dimension) of the hole, and ℓ_2 is empty.
- Take the first element t of ℓ_1.
- If it is the base point, or is already present in ℓ_2, discard it.
- Otherwise, reduce it (it gives an n-cube s which has t as one of its faces), add (t, s) to ℓ_2 and all the other faces of s to ℓ_1.
- Repeat until ℓ_1 is empty.
- Build a cubical coherence out of ℓ_2.
- Use that coherence to fill the hole.

Example

We want to prove $\mathtt{proj}(f(a), g(\star_B)) = \mathtt{basel}$.

$\ell_1 = [\mathtt{proj}(f(a), g(\star_B)), \mathtt{basel}]$
$\ell_2 = []$

$$\mathtt{proj}(f(a), g(\star_B)) \rightsquigarrow \mathtt{proj}(f(a), \star_{B'}) \text{ via } \mathsf{ap}_{\mathtt{proj}(f(a),-)}(\star_g)$$

$\ell_1 = [\mathtt{proj}(f(a), \star_{B'}), \mathtt{basel}]$
$\ell_2 = [(\mathtt{proj}(f(a), g(\star_B)), \mathsf{ap}_{\mathtt{proj}(f(a),-)}(\star_g))]$

Example

$$\ell_1 = [\texttt{proj}(f(a), \star_{B'}), \texttt{basel}]$$
$$\ell_2 = [(\texttt{proj}(f(a), g(\star_B)), \mathsf{ap}_{\texttt{proj}(f(a),-)}(\star_g))]$$

$$\texttt{proj}(f(a), \star_{B'}) \rightsquigarrow \texttt{basel} \text{ via } \texttt{pushl}(f(a))$$

$$\ell_1 = [\texttt{basel}, \texttt{basel}]$$
$$\ell_2 = [(\texttt{proj}(f(a), g(\star_B)), \mathsf{ap}_{\texttt{proj}(f(a),-)}(\star_g)), (\texttt{proj}(f(a), \star_{B'}), \texttt{pushl}(f(a)))]$$

Example

$$\ell_1 = [\texttt{basel}, \texttt{basel}]$$
$$\ell_2 = [(\texttt{proj}(f(a), g(\star_B)), \mathsf{ap}_{\texttt{proj}(f(a),-)}(\star_g)),$$
$$(\texttt{proj}(f(a), \star_{B'}), \texttt{pushl}(f(a)))]$$

$$\texttt{basel} \rightsquigarrow \texttt{proj}(\star_{A'}, \star_{B'}) \text{ via } \texttt{pushl}(\star_{A'})$$

$$\ell_1 = [\texttt{proj}(\star_{A'}, \star_{B'}), \texttt{basel}]$$
$$\ell_2 = [(\texttt{proj}(f(a), g(\star_B)), \mathsf{ap}_{\texttt{proj}(f(a),-)}(\star_g)),$$
$$(\texttt{proj}(f(a), \star_{B'}), \texttt{pushl}(f(a)))]$$
$$(\texttt{basel}, \texttt{pushl}(\star_{A'}))]$$

We're done, as everything in ℓ_1 is either in ℓ_2 or the base point.

Example

$$\ell_2 = [(\texttt{proj}(f(a), g(\star_B)), \mathsf{ap}_{\texttt{proj}(f(a),-)}(\star_g)),$$
$$(\texttt{proj}(f(a), \star_{B'}), \texttt{pushl}(f(a)))]$$
$$(\texttt{basel}, \texttt{pushl}(\star_{A'}))]$$

$$\begin{array}{llll}
\texttt{coh} : & (X : \texttt{Type}) & & (A' \wedge B') \\
& (a : X) & & (\texttt{proj}(\star_{A'}, \star_{B'})) \\
& (x_0 : X) & & (\texttt{basel}) \\
& (p_0 : a = x_0) & & (\texttt{pushl}(\star_{A'})) \\
& (x_1 : X) & & (\texttt{proj}(f(a), \star_{B'})) \\
& (p_1 : x_1 = x_0) & & (\texttt{pushl}(f(a))) \\
& (x_2 : X) & & (\texttt{proj}(f(a), g(\star_B))) \\
& (p_2 : x_2 = x_1) & & (\mathsf{ap}_{\texttt{proj}(f(a),-)}(\star_g)) \\
& \quad \to x_2 = x_0 & &
\end{array}$$

The result is the desired term of type $\texttt{proj}(f(a), g(\star_B)) = \texttt{basel}$.

Metaprogramming

It seems possible to do it in theory, but it is so technical that we do not want to do it by hand.

Metaprogramming

It seems possible to do it in theory, but it is so technical that we do not want to do it by hand.

Solution

Write a program which generates a formal proof for us!

Metaprogramming

It seems possible to do it in theory, but it is so technical that we do not want to do it by hand.

Solution

Write a program which generates a formal proof for us!

The generated proof is written in the proof assistant Agda, and the generating program is also written in Agda, used as a programming language.

Metaprogramming

It seems possible to do it in theory, but it is so technical that we do not want to do it by hand.

Solution

Write a program which generates a formal proof for us!

The generated proof is written in the proof assistant Agda, and the generating program is also written in Agda, used as a programming language.

Workflow

```
$ agda --compile SmashGenerate.agda
                            # generate the executable
$ ./SmashGenerate > Result.agda  # generate the proof
$ agda Result.agda               # check the proof
```

Results

The current version can prove almost everything except for the pentagon. In particular it can construct/prove

- $f \wedge g$, compatibility with identities
- σ, involutivity, naturality
- α, α^{-1}, inverses to each other, naturality (takes 10 minutes and 25 GB of memory)
- the hexagon (takes 7 minutes and 8 GB of memory)

Future directions

- Finish the pentagon and the few other things missing.
- Get a full meta-theoretic proof that it does work.
- Prove that the smash product is ∞-coherent (externally).
- Can this idea of higher dimensional rewriting be applied in other situations?

In topology

In topology, $A \wedge B$ is defined as a quotient.

- We identify points with each other, instead of adding paths between them.
- It is easy to define, e.g., $\alpha_{A,B,C} : (A \wedge B) \wedge C \to A \wedge (B \wedge C)$.
- The pentagon is trivial.
- It is *not* easy to prove that $\alpha_{A,B,C}$ is continuous!

The big picture

- There are some propositional equalities that we would like to pretend are reduction rules.

The big picture

- There are some propositional equalities that we would like to pretend are reduction rules.
- Cubical type theory does turn many of them into reduction rules, but we can't really hope for, e.g., $f(\star_A) \rightsquigarrow \star_{A'}$ or $\mathtt{proj}(a, \star_B) \rightsquigarrow \mathtt{basel}$ to ever be an actual reduction rule.

The big picture

- There are some propositional equalities that we would like to pretend are reduction rules.
- Cubical type theory does turn many of them into reduction rules, but we can't really hope for, e.g., $f(\star_A) \rightsquigarrow \star_{A'}$ or $\mathtt{proj}(a, \star_B) \rightsquigarrow \mathtt{basel}$ to ever be an actual reduction rule.
- Can we find an automated way to handle such propositional reduction rules?

The big picture

- There are some propositional equalities that we would like to pretend are reduction rules.
- Cubical type theory does turn many of them into reduction rules, but we can't really hope for, e.g., $f(\star_A) \rightsquigarrow \star_{A'}$ or $\mathtt{proj}(a, \star_B) \rightsquigarrow \mathtt{basel}$ to ever be an actual reduction rule.
- Can we find an automated way to handle such propositional reduction rules?
- To a user of the proof assistant, it would look like things reduce, in reality the proof assistant is doing all the work behind the scenes.