Démonstrations générées par ordinateur pour le produit smash

Guillaume Brunerie

11 avril 2018
Journées πr^2, Fontainebleau

On se place dans la théorie homotopique des types, c'est à dire la théorie des types dépendants avec

- produits et sommes dépendantes,
- type identité $u = v$, avec constructeur $\mathtt{idp}_u : u = u$, règle d'élimination J, et règle de réduction définitionnelle,
- pas d'imprédicativité (pas de Prop),
- pas d'unicité des preuves d'égalité,
- types inductifs supérieurs.

Étant donnés $f : A \to B$ et $p : a = a'$, on a

$$\mathrm{ap}_f(p) : f(a) = f(a')$$

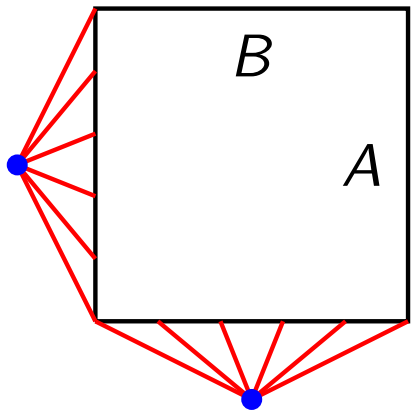
défini par induction sur p.

Le produit smash

Définition

Étant donnés deux types pointés $(A, \star_A)$ et $(B, \star_B)$, leur produit smash $A \wedge B$ est le type inductif supérieur ayant les constructeurs

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



Règle d'élimination

Étant donnés $P : A \wedge B \to \mathtt{Type}$ et

$$\begin{aligned}
\mathtt{proj}^* &: (a : A)(b : B) \to P(\mathtt{proj}(a, b)),\\
\mathtt{basel}^* &: P(\mathtt{basel}),\\
\mathtt{gluel}^* &: (a : A) \to \mathtt{proj}^*(a, \star_B) =^P_{\mathtt{gluel}(a)} \mathtt{basel}^*,\\
\mathtt{baser}^* &: P(\mathtt{baser})\\
\mathtt{gluer}^* &: (b : B) \to \mathtt{proj}^*(\star_A, b) =^P_{\mathtt{gluer}(b)} \mathtt{baser}^*,
\end{aligned}$$

on obtient

$$\wedge\text{-elim} : (x : A \wedge B) \to P(x)$$

avec des règles de réduction définitionnelles pour `proj`, `basel` et `baser`, et propositionnelles pour `gluel` et `gluer`.

Monoïdalité 1-cohérente

Objectif

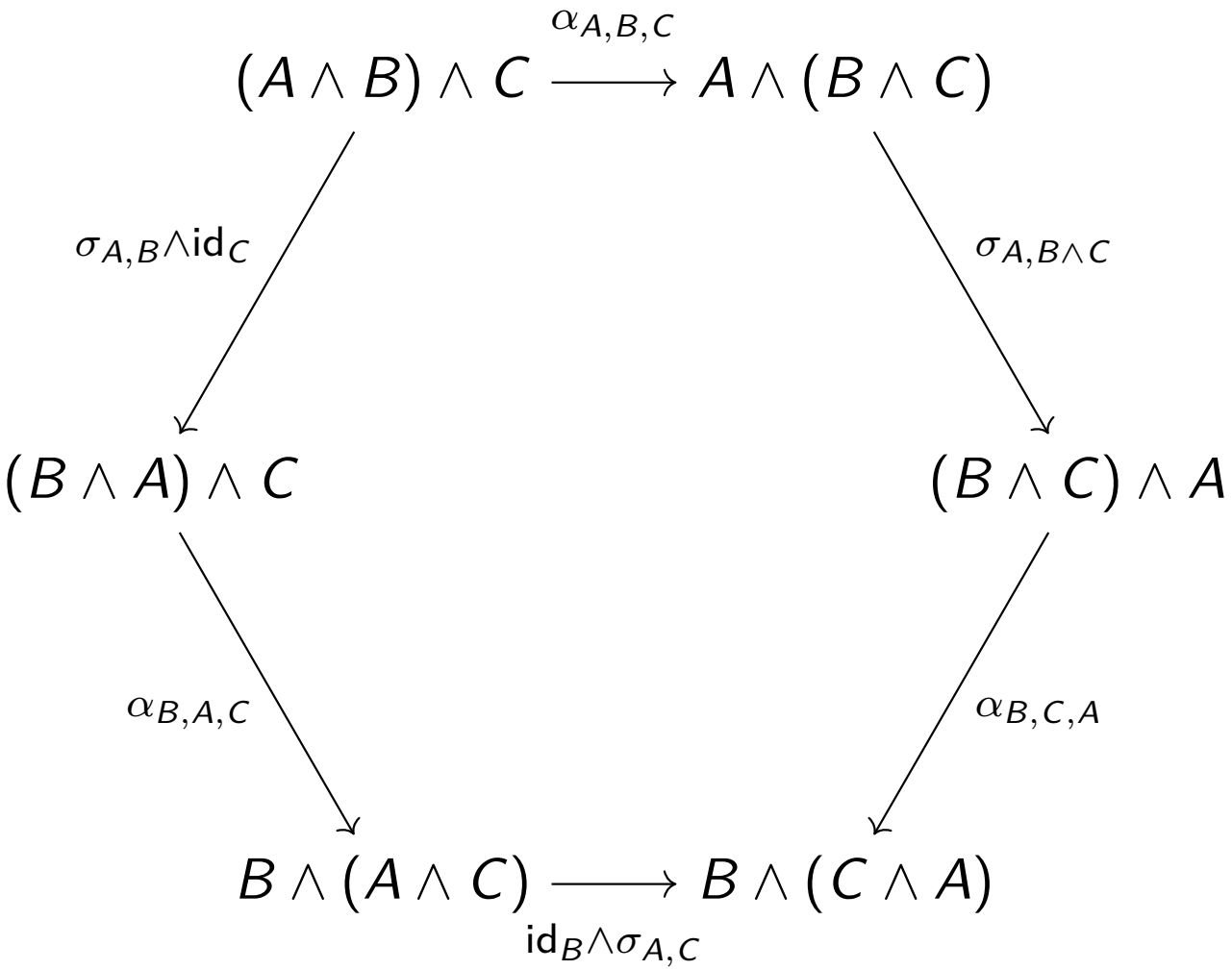
On veut une démonstration (formalisée) du fait que le produit smash est un produit monoïdal symétrique 1-cohérent. [1]

Cela veut dire que :

- Le produit smash est fonctoriel.
- Il y a une fonction $\sigma_{A,B} : A \wedge B \to B \wedge A$, qui est naturelle et satisfait $\sigma_{B,A} \circ \sigma_{A,B} = \mathrm{id}_{A \wedge B}$.
- Il y a une fonction $\alpha_{A,B,C} : (A \wedge B) \wedge C \to A \wedge (B \wedge C)$, qui est naturelle et a un inverse.
- On a le pentagone de MacLane et l'hexagone.
- Il y a une unité avec un certain diagramme triangulaire.

1. voir pages 88 et 89 de ma thèse

Hexagone


$(A \wedge B) \wedge C$
$\alpha_{A,B,C}$
$A \wedge (B \wedge C)$
$\sigma_{A,B} \wedge \mathrm{id}_C$
$\sigma_{A,B \wedge C}$
$(B \wedge A) \wedge C$
$(B \wedge C) \wedge A$
$\alpha_{B,A,C}$
$\alpha_{B,C,A}$
$B \wedge (A \wedge C)$
$B \wedge (C \wedge A)$
$\mathrm{id}_B \wedge \sigma_{A,C}$

Idée

On doit simplement définir différentes fonctions de la forme suivante :

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

où $P(x)$ est soit constant, soit un type identité $f(x) = g(x)$.

On les définit par récurrence (itérée) sur le produit smash.

- Dans le cas `proj` (itéré), on sait ce qu'on doit faire.
- Dans les autres cas, on doit "simplement" résoudre un problème d'algèbre de chemins.

Exemple 1 : commutativité

$$
\begin{aligned}
\sigma_{A,B} &: A \wedge B \to B \wedge A, \\
\sigma_{A,B}(\mathtt{proj}(a,b)) &:= \mathtt{proj}(b,a), \\
\sigma_{A,B}(\mathtt{basel}) &:= \mathtt{proj}(\star_B, \star_A), \\
\mathrm{ap}_{\sigma_{A,B}}(\mathtt{gluel}(a)) &:= \blacksquare_1 : \mathtt{proj}(\star_B, a) = \mathtt{proj}(\star_B, \star_A), \\
\sigma_{A,B}(\mathtt{baser}) &:= \mathtt{proj}(\star_B, \star_A), \\
\mathrm{ap}_{\sigma_{A,B}}(\mathtt{gluer}(b)) &:= \blacksquare_2 : \mathtt{proj}(b, \star_A) = \mathtt{proj}(\star_B, \star_A).
\end{aligned}
$$

On peut résoudre les termes manquants :

$$
\begin{aligned}
\blacksquare_1 &:= \mathtt{gluer}(a) \cdot \mathtt{gluer}(\star_A)^{-1} \\
\blacksquare_2 &:= \mathtt{gluel}(b) \cdot \mathtt{gluel}(\star_B)^{-1}
\end{aligned}
$$

Carrés

On utilise des carrés et des cubes dans le sens de [LB15] [2] .

Définition

Le type

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

est défini comme étant la famille inductive de types avec un seul constructeur

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

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

Application d'une homotopie à un chemin

Étant données deux fonctions $f, g : A \to B$, une homotopie $h : (x : A) \to f(x) =_B g(x)$ et un chemin $p : a =_A a'$, on a

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

$$\begin{array}{ccc} f(a) & \xrightarrow{h(a)} & g(a) \\ {\scriptstyle \mathrm{ap}_f(p)}\downarrow & & \downarrow{\scriptstyle \mathrm{ap}_g(p)} \\ f(a') & \xrightarrow[h(a')]{} & g(a') \end{array}$$

Exemple 2 : involutivité de la commutativité

$$\sigma\text{-inv}_{A,B} : (x : A \wedge B) \to \sigma_{B,A}(\sigma_{A,B}(x)) = x$$

$$\sigma\text{-inv}_{A,B}(\text{proj}(a,b)) := \text{idp}_{\text{proj}(a,b)}$$

$$\sigma\text{-inv}_{A,B}(\text{basel}) := \blacksquare_1 : \text{proj}(\star_A, \star_B) = \text{basel}$$

$$\begin{aligned}\text{ap}^{+}_{\sigma\text{-inv}_{A,B}}(\text{gluel}(a)) := \blacksquare_2 : \text{Square}(&\text{ap}_{\lambda x.\sigma_{B,A}(\sigma_{A,B}(x))}(\text{gluel}(a)),\\ &\text{ap}_{\lambda x.x}(\text{gluel}(a)),\\ &\text{idp}_{\text{proj}(a,\star_B)},\\ &\blacksquare_1)\end{aligned}$$

$$\sigma\text{-inv}_{A,B}(\text{baser}) := \blacksquare_3 : \text{proj}(\star_A, \star_B) = \text{baser}$$

$$\text{ap}^{+}_{\sigma_{A,B}}(\text{gluer}(b)) := \blacksquare_4 : \text{Square}([\ldots])$$

Règles de réduction pour résoudre le second but

Afin de résoudre $\blacksquare_2$, on doit utiliser :

$$\begin{aligned}
\mathrm{ap}_{\lambda x.x}(\mathtt{gluel}(a)) &= \mathtt{gluel}(a) \\
\mathrm{ap}_{\lambda x.\sigma_{B,A}(\sigma_{A,B}(x))}(\mathtt{gluel}(a)) &= \mathrm{ap}_{\sigma_{B,A}}(\mathrm{ap}_{\sigma_{A,B}}(\mathtt{gluel}(a))) \\
\mathrm{ap}_{\sigma_{A,B}}(\mathtt{gluel}(a)) &= \mathtt{gluer}(a) \cdot \mathtt{gluer}(\star_A)^{-1} \\
\mathrm{ap}_{\sigma_{B,A}}(\mathtt{gluer}(a) \cdot \mathtt{gluer}(\star_A)^{-1}) &= \mathrm{ap}_{\sigma_{B,A}}(\mathtt{gluer}(a)) \cdot \mathrm{ap}_{\sigma_{B,A}}(\mathtt{gluer}(\star_A))^{-1} \\
\mathrm{ap}_{\sigma_{B,A}}(\mathtt{gluer}(a)) &= \mathtt{gluel}(a) \cdot \mathtt{gluel}(\star_A)^{-1} \\
\mathrm{ap}_{\sigma_{B,A}}(\mathtt{gluer}(\star_A)) &= \mathtt{gluel}(\star_A) \cdot \mathtt{gluel}(\star_A)^{-1}
\end{aligned}$$

On doit ensuite construire un élément de type

$$\begin{aligned}
&\mathtt{Square}((\mathtt{gluel}(a) \cdot \mathtt{gluel}(\star_A)^{-1}) \cdot (\mathtt{gluel}(\star_A) \cdot \mathtt{gluel}(\star_A)^{-1})^{-1}, \\
&\qquad \mathtt{gluel}(a), \mathrm{idp}_{\mathtt{proj}(a,\star_B)}, \mathtt{gluel}(\star_A))
\end{aligned}$$

Fin de la construction

On généralize sur `basel`, $\texttt{proj}(\star_A, \star_B)$, $\texttt{proj}(a, \star_B)$, $\texttt{gluel}(a)$ et $\texttt{gluel}(\star_A)$. Étant donnés $X : \texttt{Type}$, $x, y, z : X$, $p : z = x$ et $q : y = x$ quelconques, on considère

$$\texttt{Square}((p \cdot q^{-1}) \cdot (q \cdot q^{-1})^{-1}, p, \texttt{idp}_z, q)$$

Finalement, on procède par induction sur p et q, et on renvoie `ids`. □

Exemple 3 : associativité

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

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

$$\alpha_{A,B,C}(\mathtt{basel}) := \blacksquare,$$

$$\alpha_{A,B,C}(\mathtt{gluel}(x)) := \alpha^{\mathtt{gluel}}_{A,B,C}(x),$$

$$\alpha_{A,B,C}(\mathtt{baser}) := \blacksquare,$$

$$\alpha_{A,B,C}(\mathtt{gluer}(c)) := \blacksquare.$$

Exemple 3 : associativité

$$\alpha_{A,B,C} : (A \wedge B) \wedge C \to A \wedge (B \wedge C),$$
$$\alpha_{A,B,C}(\mathtt{proj}(x, c)) := \alpha^{\mathtt{proj}}_{A,B,C}(x, c),$$
$$\alpha_{A,B,C}(\mathtt{basel}) := \blacksquare,$$
$$\alpha_{A,B,C}(\mathtt{gluel}(x)) := \alpha^{\mathtt{gluel}}_{A,B,C}(x),$$
$$\alpha_{A,B,C}(\mathtt{baser}) := \blacksquare,$$
$$\alpha_{A,B,C}(\mathtt{gluer}(c)) := \blacksquare.$$

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

Exemple 3 : associativité

$$\alpha_{A,B,C} : (A \wedge B) \wedge C \to A \wedge (B \wedge C),$$
$$\alpha_{A,B,C}(\mathtt{proj}(x, c)) := \alpha^{\mathtt{proj}}_{A,B,C}(x, c),$$
$$\alpha_{A,B,C}(\mathtt{basel}) := \blacksquare,$$
$$\alpha_{A,B,C}(\mathtt{gluel}(x)) := \alpha^{\mathtt{gluel}}_{A,B,C}(x),$$
$$\alpha_{A,B,C}(\mathtt{baser}) := \blacksquare,$$
$$\alpha_{A,B,C}(\mathtt{gluer}(c)) := \blacksquare.$$

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

$$\alpha^{\mathtt{gluel}}_{A,B,C} : A \wedge B \to C \to [\ldots] = [\ldots],$$
$$[\ldots \blacksquare \ldots \blacksquare \ldots \blacksquare \ldots \blacksquare \ldots \blacksquare \ldots]$$

Éléments irréductibles de $A \wedge (B \wedge C)$

On peut avoir besoin de généraliser sur les éléments suivants dans $A \wedge (B \wedge C)$. Est-ce que "path-induction" va marcher ?

$\mathtt{proj}(a, \mathtt{basel})$

$\mathtt{proj}(\star_A, \mathtt{basel})$

$\mathtt{proj}(a, \mathtt{baser})$

$\mathtt{proj}(\star_A, \mathtt{baser})$

~~$\mathtt{proj}(a, \mathtt{proj}(b, c))$~~

$\mathtt{proj}(a, \mathtt{proj}(b, \star_C))$

$\mathtt{proj}(a, \mathtt{proj}(\star_B, c))$

$\mathtt{proj}(a, \mathtt{proj}(\star_B, \star_C))$

$\mathtt{proj}(\star_A, \mathtt{proj}(b, c))$

$\mathtt{proj}(\star_A, \mathtt{proj}(b, \star_C))$

$\mathtt{proj}(\star_A, \mathtt{proj}(\star_B, c))$

$\mathtt{proj}(\star_A, \mathtt{proj}(\star_B, \star_C))$

$\mathtt{ap}_{\mathtt{proj}(a,-)}(\mathtt{gluel}(b))$

$\mathtt{ap}_{\mathtt{proj}(a,-)}(\mathtt{gluel}(\star_B))$

$\mathtt{ap}_{\mathtt{proj}(\star_A,-)}(\mathtt{gluel}(b))$

$\mathtt{ap}_{\mathtt{proj}(\star_A,-)}(\mathtt{gluel}(\star_B))$

$\mathtt{ap}_{\mathtt{proj}(a,-)}(\mathtt{gluer}(c))$

$\mathtt{ap}_{\mathtt{proj}(a,-)}(\mathtt{gluer}(\star_C))$

$\mathtt{ap}_{\mathtt{proj}(\star_A,-)}(\mathtt{gluer}(c))$

$\mathtt{ap}_{\mathtt{proj}(\star_A,-)}(\mathtt{gluer}(\star_C))$

$\mathtt{basel}$

$\mathtt{baser}$

$\mathtt{gluel}(a)$

$\mathtt{gluel}(\star_A)$

$\mathtt{gluer}(\mathtt{basel})$

$\mathtt{gluer}(\mathtt{baser})$

$\mathtt{gluer}(\mathtt{proj}(b, c))$

$\mathtt{gluer}(\mathtt{proj}(b, \star_C))$

$\mathtt{gluer}(\mathtt{proj}(\star_B, c))$

$\mathtt{gluer}(\mathtt{proj}(\star_B, \star_C))$

$\mathtt{ap}^{+}_{\mathtt{gluer}}(\mathtt{gluel}(b))$

$\mathtt{ap}^{+}_{\mathtt{gluer}}(\mathtt{gluel}(\star_B))$

$\mathtt{ap}^{+}_{\mathtt{gluer}}(\mathtt{gluer}(c))$

$\mathtt{ap}^{+}_{\mathtt{gluer}}(\mathtt{gluer}(\star_C))$

Cohérences globulaires (tout type est un ∞-groupoïde)

On peut construire n'importe quelle fonction de la forme

$$\begin{aligned}
&\texttt{coh} : (X : \texttt{Type})(a : X) \\
&\qquad [\dots] \\
&\qquad (x_n : T_n)(p_n : x_n = u_n) \qquad (\text{or } u_n = x_n) \\
&\qquad [\dots] \\
&\qquad \to T
\end{aligned}$$

où T_n, u_n et T sont construits en utilisant des variables x_k et p_k et d'autres cohérences, et T est un type identité.

Idée : induction sur tous les p_n, puis on renvoie `idp`. Ça marche parce que lorsque tous les arguments p_k sont `idp`, c'est définitionnellement égal à `idp`.

Cohérences cubiques

On a également besoin d'autoriser des paires d'arguments de la forme

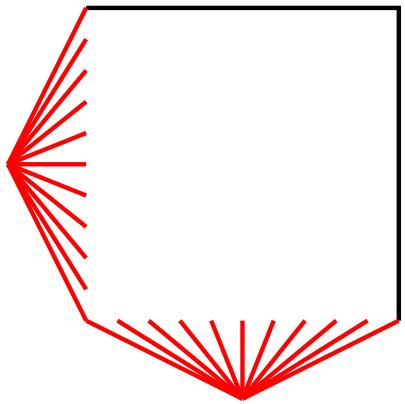
$$(x_n : T_n)(p_n : \texttt{Square}(x_n, u_n, v_n, w_n))$$

avec x_n dans chacune des quatre positions possibles, et de même avec des cubes...

Il est toujours possible de construire de telles cohérences, en utilisant un principe d'induction généralisé, où un côté d'un carré est libre et les trois autres sont fixes.

Tout contexte de la forme $(a : X)$ $[\ldots]$ $(x_n : [\ldots])(p_n : [\ldots])$ est un contexte (cubique) contractile.

Intuition



$A \wedge B$

(et imaginer le schéma correspondant à $A \wedge (B \wedge C)$)

Intuition

Tous les éléments sur lesquels on veut généraliser sont dans la partie rouge, qui est “contractile“.

Formulation plus précise

Ébauche de définition (externe)

Étant donné un type A, un système contractile sur A est une suite de termes $(u_i : \Gamma_i \to T_i)$ où chaque T_i est soit A, soit un type identité de A, soit un type carré de A, etc., telle que pour toute famille finie $(\gamma_k : \Gamma_{i_k})$, il existe une famille finie $(\delta_j : \Gamma_{i'_j})$, telle que

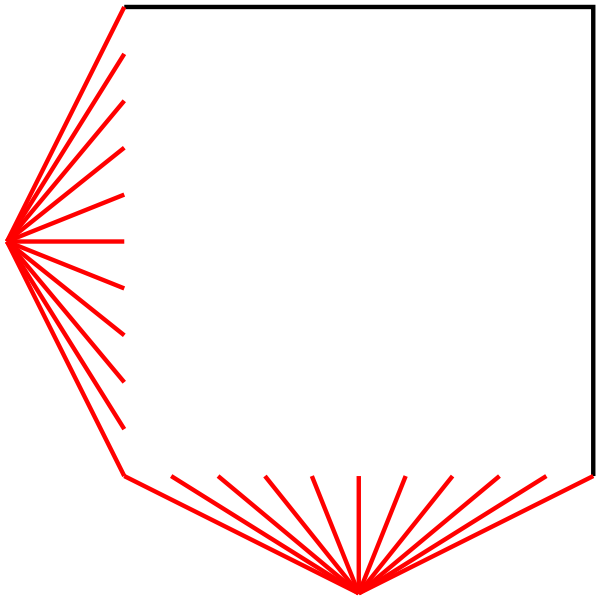
- chaque $u_{i_k}(\gamma_k)$ est un des $u_{i'_j}(\delta_j)$,
- la famille $u_{i'_j}(\delta_j)$ a la forme d'un contexte contractile,
- tous les $u_{i'_j}(\delta_j)$ sont définitionnellement différents

Example

La famille suivante est un système contractile sur $A \wedge B$:

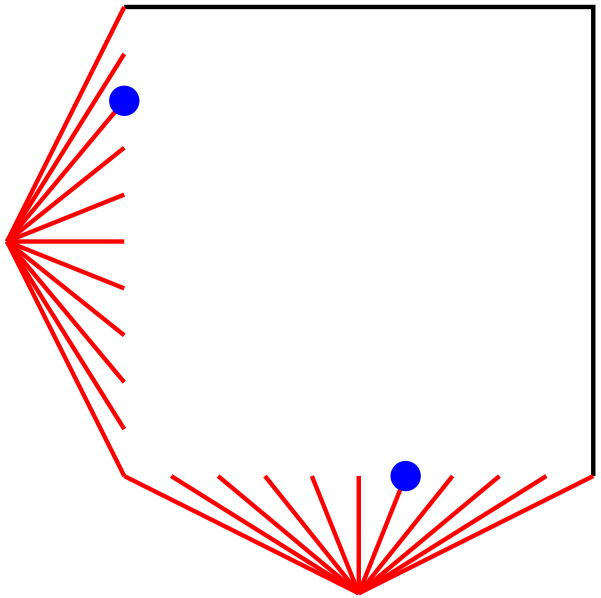
$$(\texttt{basel}, \texttt{baser}, \texttt{gluel}, \texttt{gluer}, \lambda a.\texttt{proj}(a, \star_B), \lambda b.\texttt{proj}(\star_A, b))$$

Résultat principal



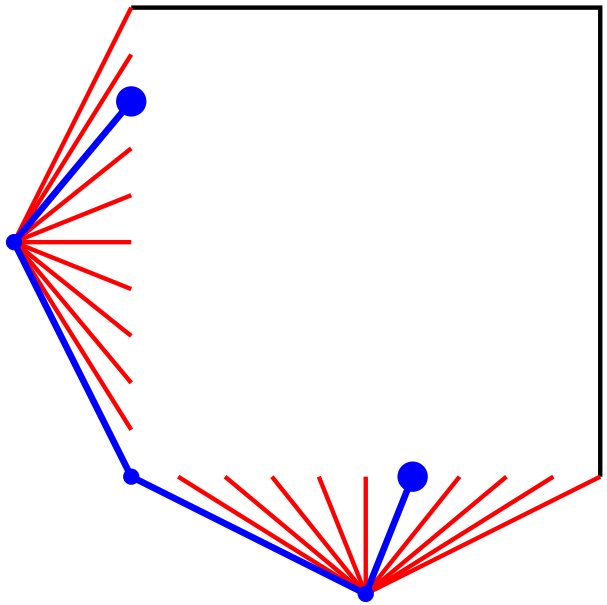
$A \wedge B$

Résultat principal



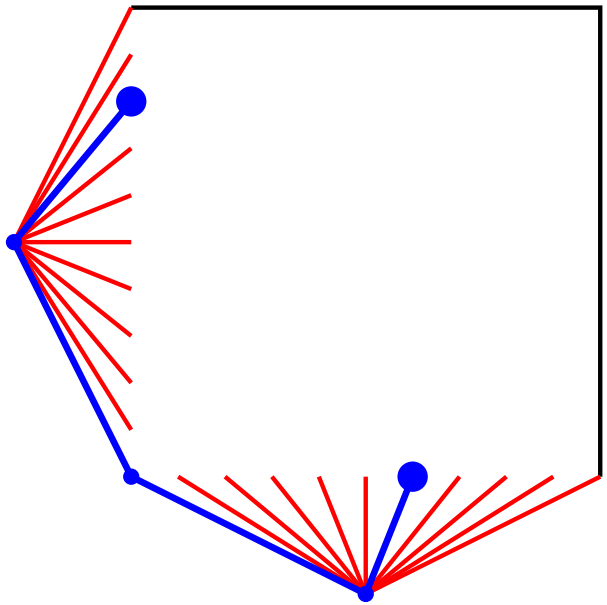
$A \wedge B$

Résultat principal



$A \wedge B$

Résultat principal



$A \wedge B$

Proposition (externe)

Étant donné un système contractile $\mathcal{C}_A$ sur un type A, pour n'importe quelle frontière d'un chemin/carré/cube dans $\mathcal{C}_A$, il existe un terme qui la remplit.

Associativité (fin)

Dans la définition de $\alpha_{A,B,C}$, pour tout ■ on doit trouver un chemin/carré/cube dont la frontière est dans $\mathcal{C}_{A\wedge(B\wedge C)}$. La proposition précédente implique (constructivement) qu'il existe un moyen de le faire. □

D'autres termes qu'on veut construire

$$\alpha^{-1}_{A,B,C} : A \wedge (B \wedge C) \to (A \wedge B) \wedge C,$$
$$\alpha\text{-rinv}_{A,B,C} : (x : (A \wedge B) \wedge C) \to \alpha^{-1}_{A,B,C}(\alpha_{A,B,C}(x)) = x$$
$$\alpha\text{-linv}_{A,B,C} : (x : A \wedge (B \wedge C)) \to \alpha_{A,B,C}(\alpha^{-1}_{A,B,C}(x)) = x$$

$$\text{hexagon} : (x : (A \wedge B) \wedge C) \to (\text{id}_B \wedge \sigma_{A,C})(\alpha_{B,A,C}((\sigma_{A,B} \wedge \text{id}_C)(x))) = \alpha_{B,C,A}(\sigma_{A,B\wedge C}(\alpha_{A,B,C}(x)))$$

$$\text{pentagon} : (x : ((A \wedge B) \wedge C) \wedge D) \to (\text{id}_A \wedge \alpha_{B,C,D})(\alpha_{A,B\wedge C,D}((\alpha_{A,B,C} \wedge \text{id}_D)(x)) = \alpha_{A,B,C\wedge D}(\alpha_{A\wedge B,C,D}(x))$$

Problèmes supplémentaires

Étant donnés $f : A \to B$ et $sq : \mathtt{Square}(p, q, r, s)$, on a

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

Question

Y a-t-il un terme de type

$$\mathrm{ap}^2_{\lambda x.f(g(x))}(sq) = \mathrm{ap}^2_f(\mathrm{ap}^2_g(sq))?$$

Problèmes supplémentaires

Étant donnés $f : A \to B$ et $sq : \mathtt{Square}(p, q, r, s)$, on a

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

Question

Y a-t-il un terme de type

$$\mathrm{ap}^2_{\lambda x.f(g(x))}(sq) = \mathrm{ap}^2_f(\mathrm{ap}^2_g(sq))?$$

Ce n'est même pas bien typé !

Problèmes supplémentaires

Étant donnés $f : A \to B$ et $sq : \mathtt{Square}(p, q, r, s)$, on a

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

Question

Y a-t-il un terme de type

$$\mathrm{ap}^2_{\lambda x.f(g(x))}(sq) = \mathrm{ap}^2_f(\mathrm{ap}^2_g(sq))?$$

Ce n'est même pas bien typé !

Solution

On a un terme de type

$$\mathtt{Cube}(\mathrm{ap}^2_{\lambda x.f(g(x))}(sq), \mathrm{ap}^2_f(\mathrm{ap}^2_g(sq)), \text{ap-}\circ_{f,g}(p), \text{ap-}\circ_{f,g}(q), \text{ap-}\circ_{f,g}(r), \text{ap-}\circ_{f,g}(s))$$

Problèmes supplémentaires (2)

Séparer la construction en deux étapes (faire toutes les “règles de réduction”, et puis résoudre le problème qu'il reste) n'est en fait pas une bonne idée, parce qu'on peut avoir besoin de démontrer certaines cohérences entre les “règles de réduction”, et ce n'est alors pas facile de les incorporer dans la construction.

Par exemple, il y a beaucoup de façons de réduire

$$\mathrm{ap}_{\lambda x.x}(\mathtt{gluel}(\star_A) \cdot \mathrm{ap}_{\lambda x.\sigma_{B,A}(\sigma_{A,B}(x))}(\mathtt{gluel}(\star_A)^{-1} \cdot \mathtt{gluer}(\star_B)))$$

Une solution est de combiner les deux étapes en une, en généralisant également sur les “règles de réduction” et les cohérences entre elles.

Mais la théorie des systèmes contractiles devient alors beaucoup plus compliquée. . .

Métaprogrammation

Ça a l'air possible à faire en théorie, mais afin d'obtenir une preuve formelle, on préfèrerait ne pas le faire à la main...

Métaprogrammation

Ça a l'air possible à faire en théorie, mais afin d'obtenir une preuve formelle, on préfèrerait ne pas le faire à la main...

Solution

Écrire un programme qui génère une preuve formelle automatiquement (en Agda).

Métaprogrammation

Ça a l'air possible à faire en théorie, mais afin d'obtenir une preuve formelle, on préfèrerait ne pas le faire à la main...

Solution

Écrire un programme qui génère une preuve formelle automatiquement (en Agda).

Le premier programme est également écrit en Agda, utilisé comme langage de programmation.

Workflow

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

Démo

J'ai la commutativité, l'associativité (et les preuves que ce sont des équivalences) et l'hexagone. Le pentagone, pas encore. . .

(démo)

Démo

J'ai la commutativité, l'associativité (et les preuves que ce sont des équivalences) et l'hexagone. Le pentagone, pas encore. . .

(démo)

La vérification de l'hexagone prend 45 minutes et 45 GB de mémoire, la majorité étant utilisé pour construire les cohérences.

Théorie cubique des types ?

Beaucoup de problèmes viennent des “règles de réduction” qui ne sont pas définitionnelles.

Théorie cubique des types ?

Beaucoup de problèmes viennent des “règles de réduction” qui ne sont pas définitionnelles.

Idée

Utiliser la théorie cubique des types où elles sont définitionnelles.

Théorie cubique des types ?

Beaucoup de problèmes viennent des “règles de réduction” qui ne sont pas définitionnelles.

Idée

Utiliser la théorie cubique des types où elles sont définitionnelles.

Problème potentiel

On utilise la règle définitionnelle de J afin de définir les cohérences, et l'espace des chemins de la théorie cubique des types a seulement la version propositionnelle.

À explorer. . .

Réflexion ?

Réflexion dans le style de ssreflect

On veut construire $a : A$ par "induction" sur A. On définit :

- un type D (ex. $D = \{x : I \mid \mathrm{ok}(\mathrm{x}) = \mathrm{true}\}$, I un type inductif),
- une fonction d'interprétation $[\![-]\!] : D \to \mathtt{Type}$,
- une fonction $\mathrm{solve} : (d : D) \to [\![d]\!]$,
- un élément $d_A : D$ tel que $[\![d_A]\!] \equiv A$.

On pose $a := \mathrm{solve}(d_A)$.

L'intérêt de cette approche est que vérifier $a : A$ est très rapide.

Réflexion ?

Réflexion dans le style de ssreflect

On veut construire $a : A$ par "induction" sur A. On définit :

- un type D (ex. $D = \{x : I \mid \mathrm{ok}(\mathrm{x}) = \mathrm{true}\}$, I un type inductif),
- une fonction d'interprétation $[\![-]\!] : D \to \mathtt{Type}$,
- une fonction $\mathrm{solve} : (d : D) \to [\![d]\!]$,
- un élément $d_A : D$ tel que $[\![d_A]\!] \equiv A$.

On pose $a := \mathrm{solve}(d_A)$.
L'intérêt de cette approche est que vérifier $a : A$ est très rapide.

Ici :

- D représenterait toutes les cohérences d'∞-groupoïdes,
- $[\![d]\!]$ serait le type de fonctions correspondant,
  $\to$ on ne sait pas faire, tour infinie de cohérences...
- solve serait une preuve (interne !) que tout type est un ∞-groupoïde

Travaux futurs

- Finir le pentagone (soit en utilisant des hypercubes, soit en essayant de le faire globulaire) et les quelques autres parties qui manquent.
- Essayer d'adapter le programme pour `cubicaltt` afin de comparer.
- Développer la théorie des contextes contractile afin d'obtenir une preuve lisible par un humain.
- Démontrer que le produit smash est ∞-cohérent (de façon externe).
- Appliquer la méthode des systèmes contractiles dans d'autres situations.