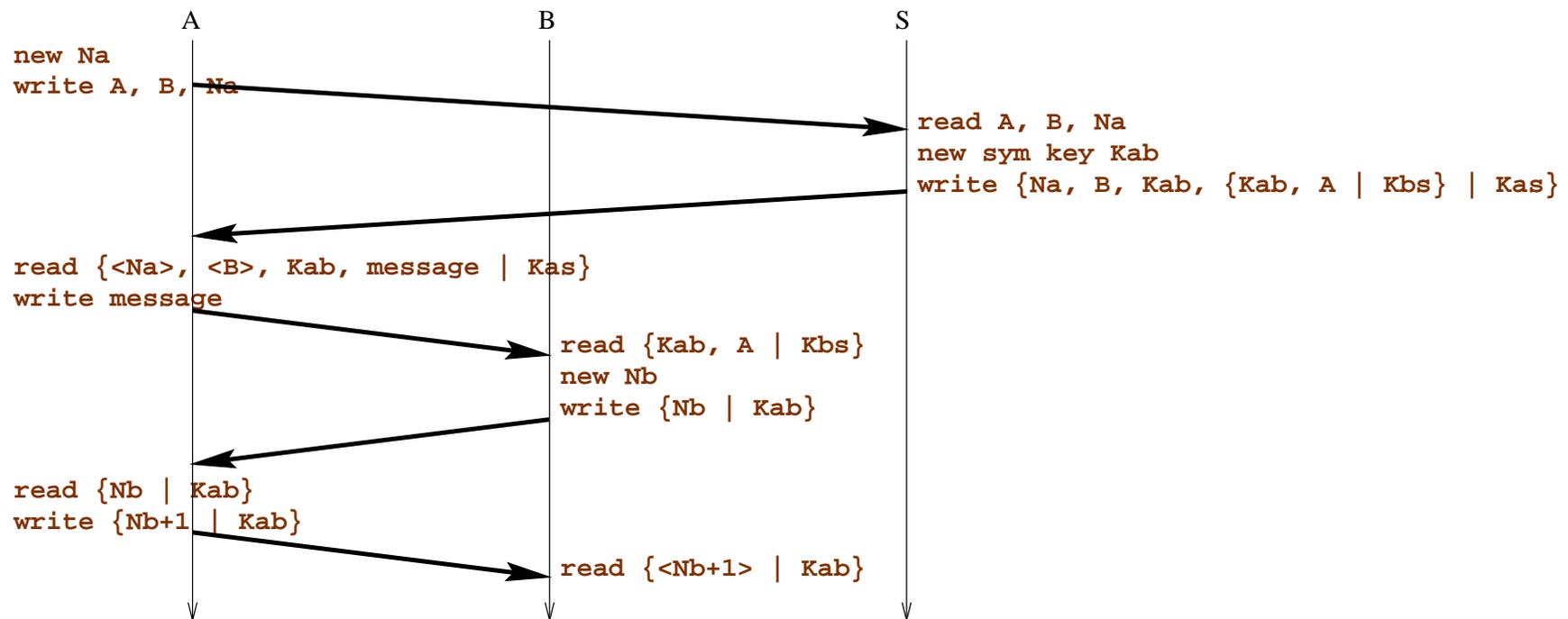


# **Relations logiques pour types monadiques ... et homologie?**

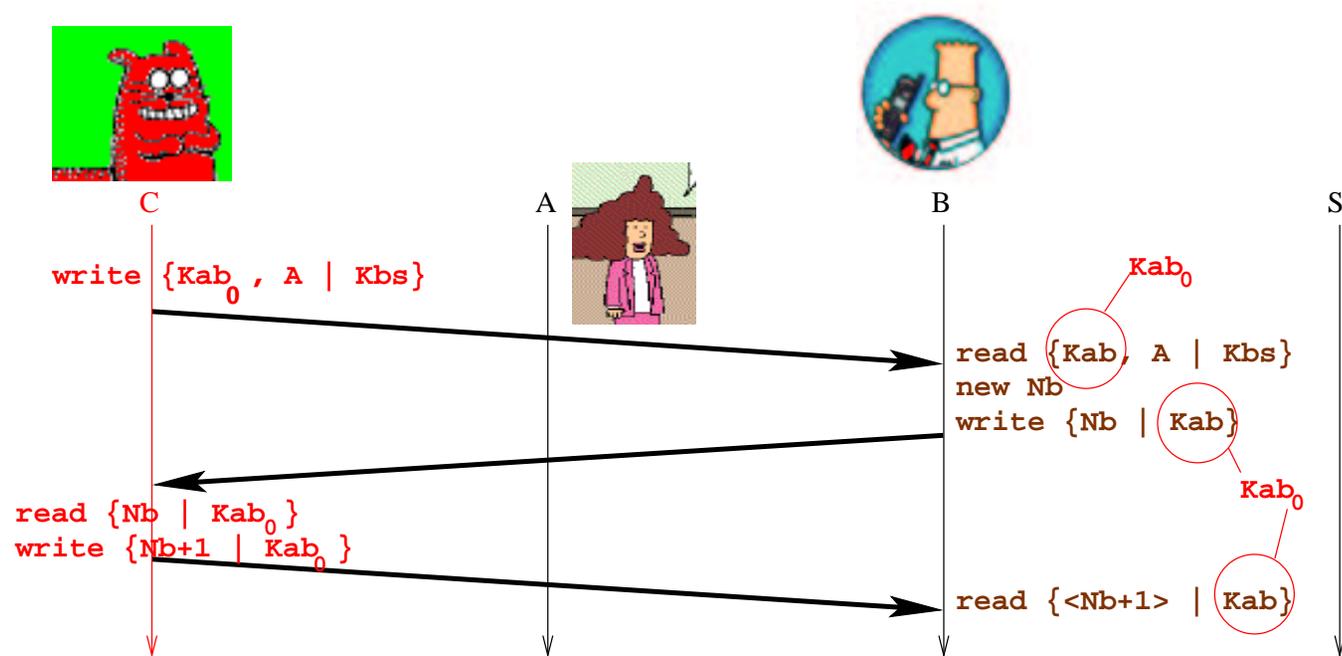
Jean Goubault-Larrecq,

LSV/CNRS UMR 8643 & INRIA Futurs projet SECSI & ENS Cachan

# Motivations (1): le protocole de Needham-Schroeder à clés privées



## Motivations (2): ... une attaque



## Le $\lambda$ -calcul cryptographique [Sumii et Pierce, CSFW'2001]

Un langage de modélisation des protocoles cryptographiques:

|   |  |
|---|--|
| $e ::= x$   | variables                                |
| $e_1 e_2$   | application de la fonction $e_1$ à $e_2$ |
| $\lambda x \cdot e$   | abstraction: fonction $x \mapsto e$      |
| $\pi_i e$   | projection ( $i = 1, 2$ )                |
| $\langle e_1, e_2 \rangle$  | paire                                    |
| $()$  | rien (!)                                 |
| $(\nu x) e$   | création de noms frais $x$               |
| $\{e_1\}_{e_2}$   | chiffrement                              |
| $\text{let } \{x\}_{e_2} = e_1 \text{ in } e_3 \text{ else } e_4$ | déchiffrement                            |
| ...   |  |

## ... sert à programmer des protocoles cryptographiques

---

(avec quelques mensonges...)

```
A := (νNa)send (A, B, Na);  
      read (λx · let {n, b, kab, m}Kas = x in  
            if n = Na ∧ b = B then  
              send (m); ...)
```

... sert surtout à **vérifier** des protocoles cryptographiques

---

Équivalence **observationnelle**:

$$e \cong e' \text{ ssi } \forall \mathcal{C}[] \cdot \mathcal{C}[e] \rightarrow^* \text{true} \Leftrightarrow \mathcal{C}[e'] \rightarrow^* \text{true}$$

Exemples [Pitts & Stark, MFCS'1993]:

$$(\nu n) \lambda x \cdot (x = n) \cong \lambda x \cdot \text{false} \quad (\nu n)(\nu n') \lambda f \cdot (fn = fn') \cong \lambda f \cdot \text{true}$$

## Typage

---

En fait, les termes sont **typés**:

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \text{ (Var)}$$

$$\frac{\Gamma, n : \text{name} \vdash e : \tau}{\Gamma \vdash (\nu n)e : \tau} \text{ (New)}$$

$$\frac{\Gamma, x : \tau \vdash e : \tau'}{\Gamma \vdash \lambda x . e : \tau \rightarrow \tau'} \text{ (Abs)}$$

$$\frac{\Gamma \vdash e_1 : \tau \rightarrow \tau' \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 e_2 : \tau'} \text{ (App)}$$

et l'on demande que

$$\Gamma \vdash e \cong_{\tau} e'$$

## Plan

---

Mon but (à terme): trouver des critères **calculables** d'équivalence observationnelle.

Idée: “il y a des monades alors il y a de l'homologie” (enfin, il devrait...)

- Relations logiques dans le  $\lambda$ -calcul (pas crypto);
- Relations logiques dans le  $\lambda$ -calcul monadique; (avec David Nowak et Slawek Lasota)
- Passage aux objets co-simpliciaux;
- Passage à la cohomologie.

## Plan

---

- Relations logiques dans le  $\lambda$ -calcul (pas crypto);
- Relations logiques dans le  $\lambda$ -calcul monadique; (avec David Nowak et Slawek Lasota)
- Passage aux objets co-simpliciaux;
- Passage à la cohomologie.

## Relations logiques pour le $\lambda$ -calcul (pas crypto)

Une **relation logique**  $R$  est une famille de relations  $R_\tau$ , une pour chaque type  $\tau$ , telle que:

$$f R_{\tau \rightarrow \tau'} f' \Leftrightarrow \forall a R_\tau a' \cdot f a R_{\tau'} f' a'$$

$$a R_{\tau \times \tau'} a' \Leftrightarrow \pi_1 a R_\tau \pi_1 a' \wedge \pi_2 a R_{\tau'} \pi_2 a'$$

$$a R_1 a'$$

**Prop** (“Basic Lemma”) si  $x_1 : \tau_1, \dots, x_n : \tau_n \vdash e : \tau$ ,

et  $a_1 R_{\tau_1} a'_1, \dots, a_n R_{\tau_n} a'_n$ ,

alors  $e[x_1 := a_1, \dots, x_n := a_n] R_\tau e[x_1 := a'_1, \dots, x_n := a'_n]$ .

**Corollaire:** si  $a R_\tau a'$  alors  $\vdash a \cong_\tau a'$ .

Comment étendre ceci au cas des **noms**?

## Avant de continuer, généralisons: la **sémantique**

On interprète les  $\lambda$ -termes typés dans une **catégorie cartésienne close**  $\mathcal{C}$ .

$\mathcal{C} \llbracket \tau \rrbracket, \mathcal{C} \llbracket \Gamma \rrbracket$  : objet       $\mathcal{C} \llbracket \Gamma \vdash e : \tau \rrbracket$  : morphisme  $\mathcal{C} \llbracket \Gamma \rrbracket \rightarrow \mathcal{C} \llbracket \tau \rrbracket$

$$\mathcal{C} \llbracket \dots, x_i : \tau_i, \dots \vdash x_i : \tau_i \rrbracket = \dots \times \mathcal{C} \llbracket \tau_i \rrbracket \times \dots \xrightarrow{\pi_i} \mathcal{C} \llbracket \tau_i \rrbracket$$

$$\mathcal{C} \llbracket \Gamma \vdash e_1 e_2 : \tau' \rrbracket = \mathcal{C} \llbracket \Gamma \rrbracket \xrightarrow{\mathcal{C} \llbracket e_1 \rrbracket} \overbrace{\mathcal{C} \llbracket \tau \rightarrow \tau' \rrbracket}^{\text{hom}(\mathcal{C} \llbracket \tau \rrbracket, \mathcal{C} \llbracket \tau' \rrbracket)} \xrightarrow{\text{App}} \mathcal{C} \llbracket \tau' \rrbracket$$

$$\xrightarrow{\mathcal{C} \llbracket e_2 \rrbracket} \times \mathcal{C} \llbracket \tau \rrbracket$$

$$\mathcal{C} \llbracket \Gamma \vdash \lambda x \cdot e : \tau \rightarrow \tau' \rrbracket = \mathcal{C} \llbracket \Gamma \rrbracket \xrightarrow{\Lambda(\mathcal{C} \llbracket \Gamma, x : \tau \vdash e : \tau' \rrbracket)} \text{hom}(\mathcal{C} \llbracket \tau \rrbracket, \mathcal{C} \llbracket \tau' \rrbracket)$$

$$\left( \begin{array}{c} \mathcal{C} \llbracket \Gamma, x : \tau \rrbracket \\ = \mathcal{C} \llbracket \Gamma \rrbracket \times \mathcal{C} \llbracket \tau \rrbracket \end{array} \xrightarrow{\mathcal{C} \llbracket e \rrbracket} \mathcal{C} \llbracket \tau' \rrbracket \right)$$

## L'usage des CCC est bien une généralisation

---

**Lemme** Pour tout ensemble  $\Sigma$  (vu comme une catégorie), il existe une CCC **libre**  $\lambda(\Sigma)$  au-dessus de  $B$ :

Objets = types  $\tau ::= \Sigma | \tau \rightarrow \tau | \tau \times \tau$

Morphismes =  $\lambda$ -termes modulo  $\beta\eta$ -équivalence  $=_{\beta\eta}$

$\tau_1 \times \dots \times \tau_n \xrightarrow{e} \tau$

où  $x_1 : \tau_1, \dots, x_n : \tau_n \vdash e : \tau$ ,

mod  $(\lambda x \cdot e)e' =_{\beta\eta} e[x := e']$ ,

$\lambda x \cdot ex =_{\beta\eta} e$  ( $x$  non libre dans  $e$ )

## La sémantique, en diagrammes

---

Pour toute CCC  $\mathcal{C}$ , la sémantique est entièrement décrite par le diagramme suivant dans *Cat*:

$$\begin{array}{ccc} \Sigma & \xrightarrow{\underline{\mathcal{C}}} & \lambda(\Sigma) \\ \downarrow & \swarrow \mathcal{C}[-] & \\ \mathcal{C} & & \end{array}$$

où  $\mathcal{C}[-]$  est donc une **représentation de CCC**.

## Relations logiques, catégoriquement [Mitchell&Scedrov, CSL'1992]

Soit  $\mathbb{C}$  une catégorie avec produits finis et pullbacks, (penser **Set**)  
 $|-| : \mathbf{C} \rightarrow \mathbb{C}$  un foncteur préservant les produits. (penser  $\mathbf{C}(1, -)$ )

Le **sous-scone**  $\mathbf{C} \Downarrow \mathbb{C}$  est la catégorie: ( $\sim$  comma-catégorie)

Objets = triplets  $\langle S \in \mathbb{C}, m, A \in \mathbf{C} \rangle$ ,  
 où  $S \hookrightarrow^m |A|$  ( $m$  **mono**) (penser sous-ensemble)

Morphismes = couples  $(u, v)$  de morphismes dans  $\mathbf{C}$   
 $\langle S, m, A \rangle \xrightarrow{(u,v)} \langle S', m', A' \rangle$  tels que

$$\begin{array}{ccc} S \hookrightarrow^m |A| & & \\ u \downarrow & & \downarrow |v| \\ S' \hookrightarrow^{m'} |A'| & & \end{array}$$

Trivial mais important:  $U : \mathbf{C} \Downarrow \mathbb{C} \rightarrow \mathbf{C}$  (oubli).

$$U \langle S, m, A \rangle = A \quad U(u, v) = v$$

## Le “Basic Lemma”, catégoriquement

---

Si on arrive à montrer que  $\mathcal{C} \Downarrow \mathbb{C}$  est une CCC, alors:

$$\begin{array}{ccc} \Sigma & \xrightarrow{\subseteq} & \lambda(\Sigma) \\ \downarrow & \searrow & \downarrow \\ \mathcal{C} \Downarrow \mathbb{C} & \xrightarrow{U} & \mathbb{C} \end{array}$$

$\mathcal{C} \Downarrow \mathbb{C} \xrightarrow{\mathcal{C} \Downarrow \mathbb{C}[-]} \mathbb{C}$  and  $\lambda(\Sigma) \xrightarrow{\mathbb{C}[-]} \mathbb{C}$

**Exercice** Vérifier que ça donne les relations logiques du début de cette présentation...

$$(\mathcal{C} = \mathbb{C} = \lambda(\Sigma), \lfloor - \rfloor = \text{id})$$

## La construction d'une structure de CCC sur $\mathcal{C} \downarrow \mathcal{C}$

---

On commence par le trivial:

- Objet terminal

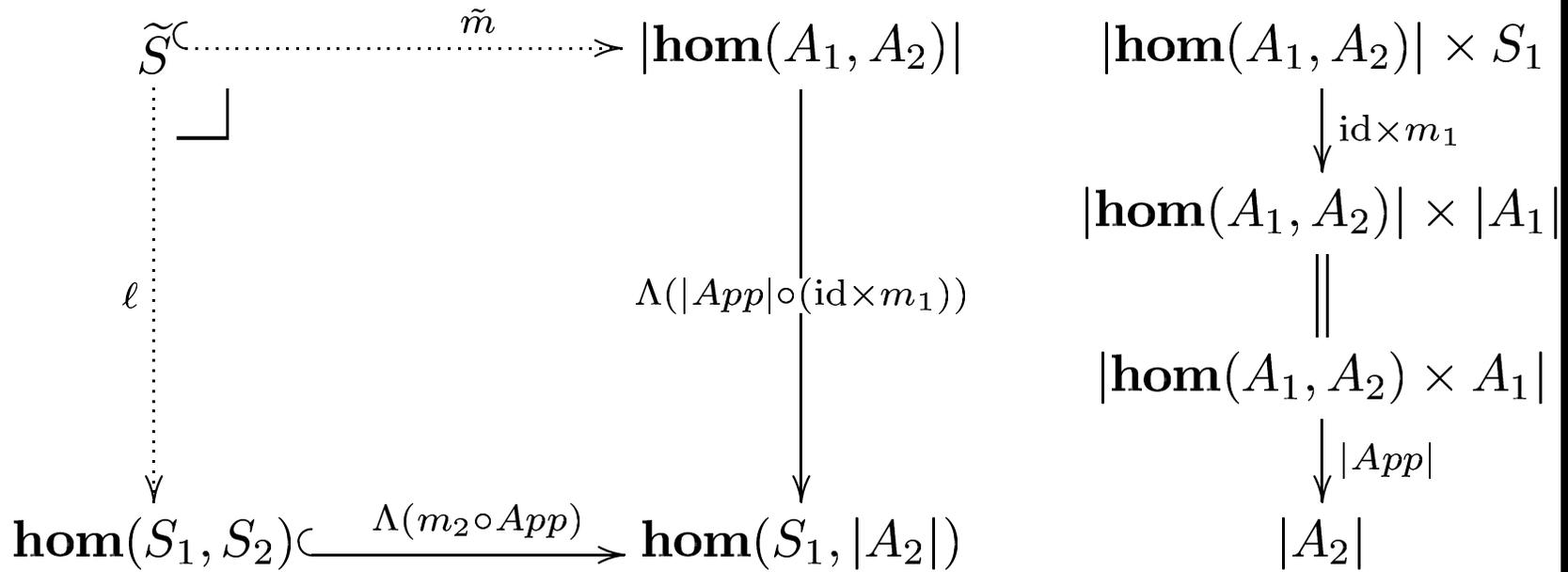
$$\langle 1_{\mathcal{C}}, \text{id}, 1_{\mathcal{C}} \rangle$$
$$1_{\mathcal{C}} \hookrightarrow \text{id} \rightarrow |1_{\mathcal{C}}| = 1_{\mathcal{C}}$$

- Produit  $\langle S_1, m_1, A_1 \rangle \times \langle S_2, m_2, A_2 \rangle = \langle S_1 \times S_2, m_1 \times m_2, A_1 \times A_2 \rangle$

$$S_1 \times S_2 \hookrightarrow m_1 \times m_2 \rightarrow |A_1| \times |A_2| = |A_1 \times A_2|$$

- L'exponentielle se construit grâce aux pullbacks de  $\mathcal{C}$ ...

$$\tilde{S} \xrightarrow{\tilde{m}} |\mathbf{hom}(A_1, A_2)| = \mathbf{hom}(S_1 \xrightarrow{m_1} |A_1|, S_2 \xrightarrow{m_2} |A_2|)$$



$$\mathbf{hom}(S_1, S_2) \times S_1 \xrightarrow{App} S_2 \xrightarrow{m_2} |A_2|$$

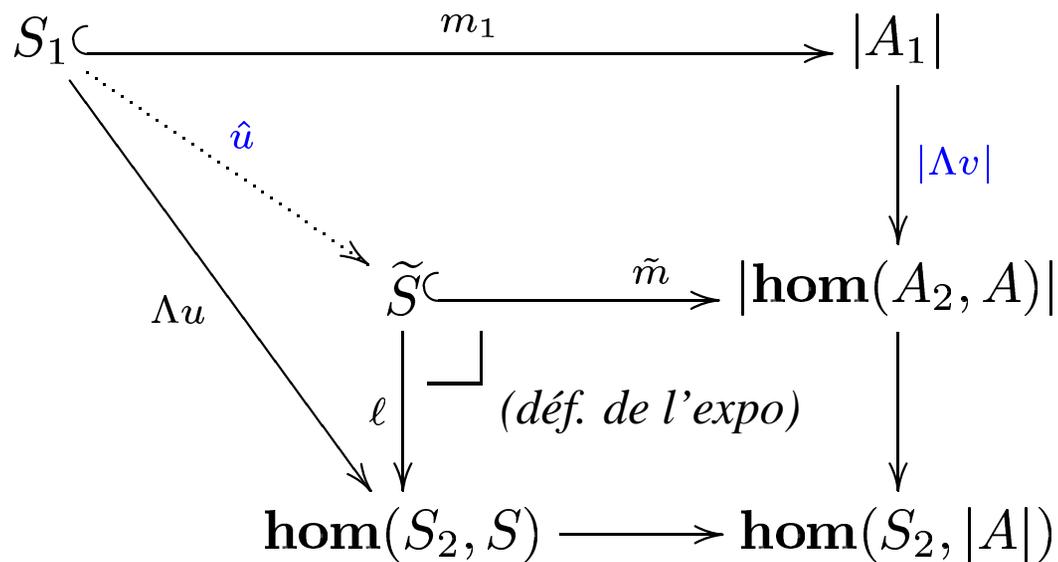
## Application

$$\mathbf{hom}( S_1 \xrightarrow{m_1} |A_1| , S_2 \xrightarrow{m_2} |A_2| ) \times ( S_1 \xrightarrow{m_1} |A_1| )$$

$$\begin{array}{ccc}
 \tilde{S} \times S_1 & \xrightarrow{\tilde{m} \times \text{id}} & |\mathbf{hom}(A_1, A_2)| \times S_1 \xrightarrow{\text{id} \times m_1} |\mathbf{hom}(A_1, A_2)| \times |A_1| \\
 \downarrow \ell \times \text{id} & \text{(d\'ef. de l'expo)} & \downarrow \\
 \mathbf{hom}(S_1, S_2) \times S_1 & \longrightarrow & \mathbf{hom}(S_1, |A_2|) \times S_1 \quad \text{(calcul)} \\
 \downarrow \text{App} & \text{(calcul)} & \downarrow \text{App} \\
 S_2 \xrightarrow{m_2} & |A_2| & \xleftarrow{|App|} |\mathbf{hom}(A_1, A_2)| \times |A_1|
 \end{array}$$

## Abstraction ( $\Lambda(\dots)$ )

Soit  $\langle S_1, m_1, A_1 \rangle \times \langle S_2, m_2, A_2 \rangle \xrightarrow{(u,v)} \langle S, m, A \rangle$ ,  
 on définit  $\Lambda(u, v) = (\hat{u}, \Lambda v)$  tel que:



Voilà, le sous-scone est une CCC!

## Plan

---

- Relations logiques dans le  $\lambda$ -calcul (pas crypto);
- Relations logiques dans le  $\lambda$ -calcul monadique; (avec David Nowak et Slawek Lasota)
- Passage aux objets co-simpliciaux;
- Passage à la cohomologie.

## Le $\lambda$ -calcul monadique, syntaxiquement

---

Un nouveau type:  $\tau ::= \Sigma | \tau \rightarrow \tau | \mathbf{T}\tau$ .

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \text{ (Var)}$$

$$\frac{\Gamma, x : \tau \vdash e : \tau'}{\Gamma \vdash \lambda x \cdot e : \tau \rightarrow \tau'} \text{ (Abs)}$$

$$\frac{\Gamma \vdash e_1 : \tau \rightarrow \tau' \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 e_2 : \tau'} \text{ (App)}$$

$$\frac{\Gamma \vdash u : \tau}{\Gamma \vdash \text{val } u : \mathbf{T}\tau} \text{ (TI)}$$

$$\frac{\Gamma \vdash u : \mathbf{T}\tau \quad \Gamma, x : \tau \vdash v : \mathbf{T}\tau'}{\Gamma \vdash \text{let val } x = u \text{ in } v : \mathbf{T}\tau'} \text{ (TE)}$$

## La création de noms, en $\lambda$ -calcul monadique

---

Idée: étant donné  $A$ ,  $TA \sim \{(\nu n_1, \dots, n_k)v \mid n_1, \dots, n_k : \text{name}, v \in A\}$ .

On se donne une constante  $\text{new} : T\text{name} \sim (\nu n)n$

Alors:

$$(\nu n)e \sim \text{let val } n = \text{new in } e$$

## Sémantique

On se donne maintenant une CCC  $\mathcal{C}$  avec une **monade forte**  $(T, \eta, \mu, t)$  (une **let-catégorie**).

$$\mathcal{C} \llbracket T\tau \rrbracket = T\mathcal{C} \llbracket \tau \rrbracket$$

$$\mathcal{C} \llbracket \Gamma \vdash \text{val } e : T\tau \rrbracket = \mathcal{C} \llbracket \Gamma \rrbracket \xrightarrow{\mathcal{C}[e]} \mathcal{C} \llbracket \tau \rrbracket \xrightarrow{\eta} T\mathcal{C} \llbracket \tau \rrbracket$$

$$\begin{array}{c} \mathcal{C} \llbracket \Gamma \vdash \text{let val } x = e \text{ in } e' : T\tau' \rrbracket = \mathcal{C} \llbracket \Gamma \rrbracket \begin{array}{l} \xrightarrow{\text{id}} \mathcal{C} \llbracket \Gamma \rrbracket \\ \xrightarrow{\mathcal{C}[e]} \times T\mathcal{C} \llbracket \tau \rrbracket \end{array} \\ \downarrow t \\ T(\mathcal{C} \llbracket \Gamma \rrbracket \times \mathcal{C} \llbracket \tau \rrbracket) \\ \downarrow T\mathcal{C}[e'] \\ T^2\mathcal{C} \llbracket \tau' \rrbracket \xleftarrow{\mu} T\mathcal{C} \llbracket \tau' \rrbracket \end{array}$$

## La sémantique, catégoriquement

**Prop** Le  $\lambda$ -calcul monadique typé forme la let-catégorie **libre**  $Comp(\Sigma)$ .

$$\eta_\tau = x : \tau \vdash \text{val } x : T\tau$$

$$\mu_\tau = x : T^2\tau \vdash \text{let val } y = x \text{ in } y : T\tau$$

$$t_{\tau, \tau'} = x : \tau, y : T\tau' \vdash \text{let val } z = y \text{ in } \langle x, z \rangle : T\tau \times \tau'$$

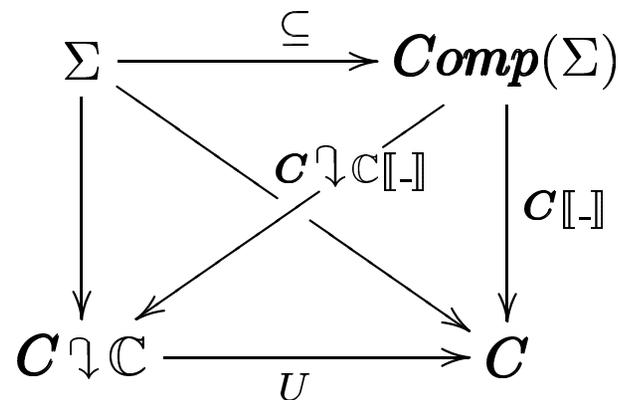
Pour toute let-catégorie  $\mathbf{C}$ , sémantique=:

$$\begin{array}{ccc} \Sigma & \xrightarrow{\subseteq} & \mathbf{Comp}(\Sigma) \\ \downarrow & \swarrow c[-] & \\ \mathbf{C} & & \end{array}$$

où  $\mathbf{C}[-]$  est donc une **représentation de let-catégorie**.

## Comment fabriquer des relations logiques avec des monades

Si on arrive à montrer que  $\mathbf{C} \Downarrow \mathbb{C}$  est une **let-catégorie**, alors:



où  $\mathbf{C} \Downarrow \mathbb{C} [-]$  et  $\mathbf{C} [-]$  sont des représentations de let-catégories...

Il n'y a plus qu'à trouver une structure de monade forte sur  $\mathbf{C} \Downarrow \mathbb{C} [-]$ .

Pour ça, on suppose qu'on a déjà une monade forte  $(T, \eta, \mu, t)$  sur  $\mathbb{C}$

+ 7 (!) autres hypothèses...

## Les ingrédients nécessaires (dans $\mathbb{C}$ ): 1. **factorisation mono**

Un système de **factorisation mono** est la donnée de deux classes de morphismes de  $\mathbb{C}$ ,  
celle des **pseudoepis**  $\twoheadrightarrow$ , celle **monos pertinents**  $\hookrightarrow$ .

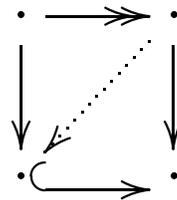
(a) tout mono pertinent est mono;

(on demande aussi que les monos pertinents soient préservés par pullbacks)

(b) tout iso est à la fois  $\twoheadrightarrow$  et  $\hookrightarrow$  ;

(c) les deux classes sont stables par composition avec les isos;

(d) Propriété de **relèvement**:



## Les ingrédients nécessaires (dans $\mathbb{C}$ ): 2. **distributivité**

Il nous faut une transformation naturelle

$$\sigma : T|-| \Rightarrow |T|$$

telle que

$$\begin{array}{ccccc}
 & & & & T^2|A| \\
 & & & & \downarrow T\sigma_A \\
 & & & \mu_{|A|} \swarrow & T|TA| \\
 & & T|A| & & \downarrow \sigma_{TA} \\
 & \nearrow \eta_{|A|} & \downarrow \sigma_A & \downarrow \sigma_A & \downarrow \sigma_{TA} \\
 |A| & \xrightarrow{\eta_A} & |TA| & & |T^2A| \\
 & & & \longleftarrow \mu_A & \\
 & & & & T|A|
 \end{array}$$

... autrement dit, un morphisme de monades.

# Construction de $\tilde{T}$

$$\tilde{T}(S \hookrightarrow^m |A|) = \begin{array}{ccc} TS & \xrightarrow{Tm} & T|A| \\ e \downarrow & & \downarrow \sigma_A \\ \tilde{S} & \xrightarrow{m} & |TA| \end{array}$$

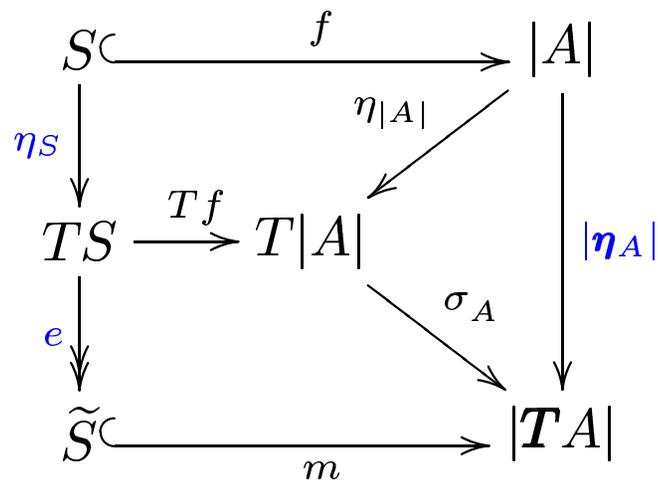
$$\tilde{T} \left( \begin{array}{ccc} S \hookrightarrow^m |A| & & \searrow |v| \\ \swarrow u & & \\ S' \hookrightarrow^{m'} |A'| & & \end{array} \right)$$

$$= \begin{array}{ccccc} TS & \xrightarrow{Tm} & T|A| & & \searrow T|v| \\ e \downarrow & \swarrow Tu & \downarrow & \searrow & \\ TS' & \xrightarrow{Tm'} & T|A'| & & \downarrow \sigma_{A'} \\ \tilde{S} & \xrightarrow{m} & |TA| & & \downarrow \sigma_{A'} \\ \downarrow e' & & \downarrow e' & & \downarrow \sigma_{A'} \\ \tilde{S}' & \xrightarrow{m'} & |TA'| & & \downarrow \sigma_{A'} \\ \swarrow \hat{u} & & \swarrow T|v| & & \end{array}$$

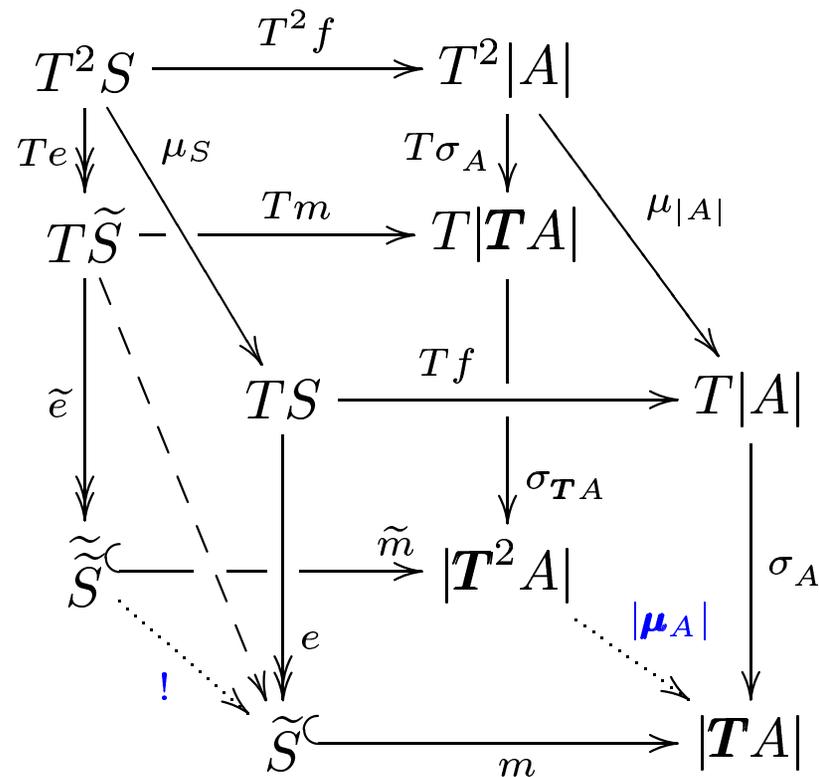
# Construction de l'unité $\tilde{\eta}$

$$S \hookrightarrow \xrightarrow{f} |A|$$


---



# Construction de la multiplication $\tilde{\mu}$

$$S \hookrightarrow_f |A|$$


à condition que  $T$  préserve les pseudoepis (on peut demander un peu moins...)

## Construction de la force

---

... je vous l'épargne!

---

La morale: tout se fait naturellement.

Mais qu'est-ce que ça donne en vrai?



## La relation logique obtenue

---

$$(\nu s_1)a_1 \tilde{S} s (\nu s_2)a_2 \iff \exists s_0 \in \mathcal{I} \cdot \exists i_1 : s_1 \rightarrow s_0 \in \mathcal{I} \cdot \exists i_2 : s_2 \rightarrow s_0 \in \mathcal{I} \cdot \\ a_1[s_1 := i_1(s_1)] S(s + s_0) a_2[s_2 := i_2(s_2)]$$

- ressemble au traitement des noms en bisimulation cadrée [Abadi & Gordon, NJC'1998]
- ressemble aux relations logiques de [Pitts & Stark, MFCS'1993]
- est en fait moins précise [Zhang 2002]  
... il faut passer à  $\mathbf{Set}^{\mathcal{I}, \mathcal{I}}$  [Nowak 2003],  
... mais c'est déjà assez compliqué pour cet exposé, non?

## Autres applications

---

- Monade  $\mathbb{P}$  dans **Set**:  $A \rightarrow TA$  est le type des  **systèmes de transitions** ,  
... on retrouve les bisimulations fortes.
- Monade de l'espace des mesures dans  $\mathcal{C} = \mathit{Mes}$ ,  $\mathbb{C} = \mathit{Set}$ :  $A \rightarrow TA$   
est le type des  **chaînes de Markov** ,  
... on retrouve les bisimulations probabilistes [Larsen & Skou,  
I&C'1991]  
... en fait, ça marche mieux avec l'espace des valuations continues dans  
 $\mathcal{C} = \mathbb{C} = \mathit{Cpo}$ .

## Plan

---

- Relations logiques dans le  $\lambda$ -calcul (pas crypto);
- Relations logiques dans le  $\lambda$ -calcul monadique; (avec David Nowak et Slawek Lasota)
- Passage aux objets co-simpliciaux;
- Passage à la cohomologie.

## Que peut-on faire ici?

---

La question à résoudre est:

Étant donnés deux termes  $e, e'$  de type  $\tau$ , existe-t-il une relation logique  $R$  telle que  $e R_\tau e'$ ?

Je fais le pari que c'est indécidable, ou trop compliqué en pratique.

Je souhaite donc trouver un critère d'existence [ou de non-existence] de telles relations logiques.

Comme on a des **monades**, la **(co)homologie de Barr-Beck** semble une voie toute tracée, non?

## Une fausse piste (pour les noms)

---

Dans  $\mathbf{Set}^{\mathcal{I}}$ , on a déjà une structure **co-semi-simpliciale**.

En vrai,  $\mathbf{Set}^{\mathcal{I}} \cong \mathbf{Set}^{\Delta^-}$ , où  $\Delta^-$  est la catégorie semi-simpliciale (on a juste les faces).

Ça suffit pour définir un complexe de cochaînes (donc une cohomologie):

- $C_n[\tau] = \mathbb{Z}\mathbf{Set} [[\tau]] ([n])$ ,
- $d : C_n[\tau] \rightarrow C_{n+1}[\tau], x \mapsto \sum_{i=0}^{n+1} (-1)^i \partial^i x$ .

Mais tout se dégrade très vite:

- On n'a pas Eilenberg-Zilber, donc pas Künneth non plus.  
(Hum, on a peut-être Künneth... merci à l'assistance de me l'avoir signalé...)
- Une homotopie (co)simpliciale ne fournit pas en général d'homotopie de chaînes dans  $C_\bullet$  (!)

## Prenons la monade au sérieux...

Plongement  $(-)^*$  de  $\mathbb{C}$  dans  $\mathbb{C}^\Delta$ , via la monade [Barr & Beck, STCHT'1969]:

- $A^* = (T^{n+1} A)_{n \geq -1}$ ;
- $\partial^i : A_n^* \rightarrow A_{n+1}^* = T^i \eta_{T^{n+1-i} A}$ ,  $s^i : A_{n+1}^* \rightarrow A_n^* = T^i \mu_{T^{n-i} A}$ .

Si  $\mathbb{C} = \mathbf{Set}$  ce que vous voulez,  $\mathbb{C}^\Delta$  est une CCC avec pullbacks, il y a une factorisation epi-mono, il y a même une monade forte:

- $T^* K = K[-1]$  (co-décalage,  $K[-1]_n = K_{n-1}$ )  
**Note:**  $T^* A^* = (T A)^*$ ;
- $\eta_{K^*} = \partial^0$ ,  $\mu_{K^*} = s^0$ .

On peut (a priori) refaire le sous-scone dans  $\mathbb{C}^\Delta$  au lieu de  $\mathbb{C}$  (c'est même plus précis!), en composant  $|-|^* = (-)^* \circ |-|$ , mais...

## Problème: $(-)^*$ ne préserve pas les produits finis!

---

... parce que  $T$  ne préserve pas en général les produits finis.

Si  $T = \mathbb{P}$ , ça voudrait dire  $\mathbb{P}(A \times B) = \mathbb{P}A \times \mathbb{P}B$ .

Dans le cas de la monade des noms, ça reviendrait à mettre en bijection

les  $(\nu s)(a, b) \equiv (\nu s, s')(a, b)$  ( $s'$  non utilisé **ni** par  $a$  **ni** par  $b$ )

et

les  $(\nu s)a, (\nu s)b \equiv (\nu s, s')a, (\nu s, s'')b$  ( $s'$  non utilisé par  $a$  mais peut-être par  $b$ , et symétriquement)

Catégoriquement,  $(-)^*$  préserverait  $\times$  si  $\mathcal{I}$  était une catégorie filtrante (conjecture), mais ce n'est pas le cas...

## On s'en sort quand même!

---

On refait toute la construction du sous-scone en supposant seulement que  $|-|^*$  est **monoïdal**:

$$\begin{array}{ccc} |A_1|^* \times |A_2|^* & \xrightarrow{\theta_{A_1, A_2}} & |A_1 \times A_2|^* \\ 1 & \xrightarrow{\iota} & |1|^* \end{array}$$

avec  $\theta$  naturel + conditions de cohérence.

Nécessite en plus une **factorisation mono** dans  $\mathbb{C}^\Delta$  .. mais on en a déjà une!

## Parenthèse: $(-)^*$ est monoïdal dans $\mathbb{C}^\Delta$

... car  $T$  est une monade **forte**.

$$\begin{array}{ccc} TA_1 \times TA_2 & \xrightarrow{t} & T(TA_1 \times A_2) \\ & & \downarrow T(\cong) \\ T^2(A_2 \times A_1) & \xleftarrow{Tt} & T(A_2 \times TA_1) \\ & & \downarrow T^2(\cong) \\ T^2(A_1 \times A_2) & \xrightarrow{\mu} & T(A_1 \times A_2) \end{array}$$

On en déduit que  $T$  est monoïdal dans  $\mathbb{C}$ , donc  $(-)^*$  monoïdal dans  $\mathbb{C}^\Delta$ , donc aussi  $|-|^*$ .

## La construction d'une structure de CCC sur $\mathbb{C} \Downarrow \mathbb{C}^\Delta$

On commence par le [un peu moins] trivial:

- Objet terminal

$$\langle 1_{\mathbb{C}}, \iota, 1_{\mathbb{C}} \rangle$$

$$1_{\mathbb{C}} \xrightarrow{\iota} |1_{\mathbb{C}}|^*$$

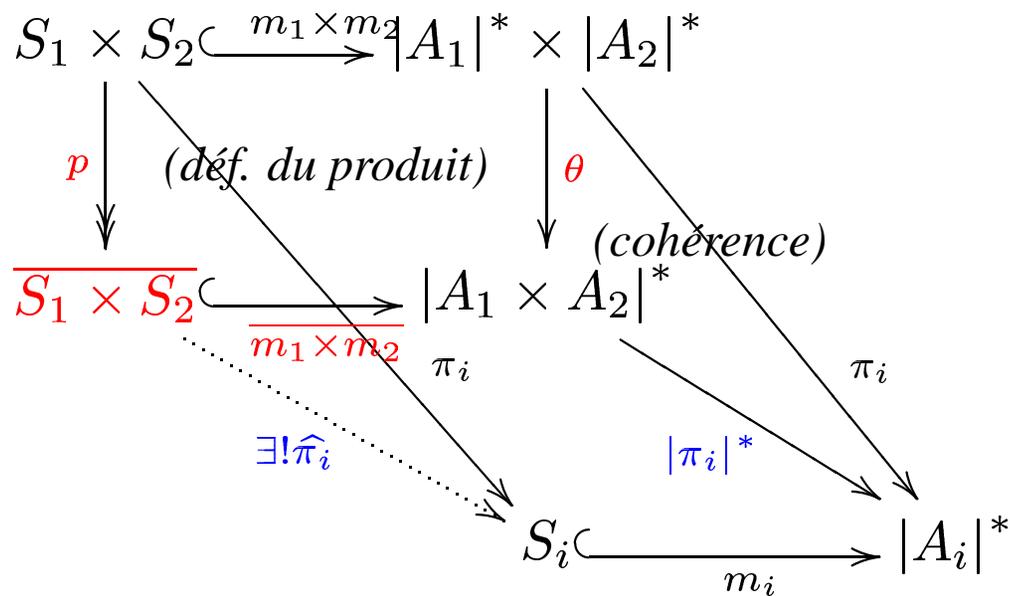
- Produit  $\langle S_1, m_1, A_1 \rangle \times \langle S_2, m_2, A_2 \rangle = \langle \overline{S_1 \times S_2}, \overline{m_1 \times m_2}, A_1 \times A_2 \rangle$

$$S_1 \times S_2 \xrightarrow{m_1 \times m_2} |A_1|^* \times |A_2|^*$$

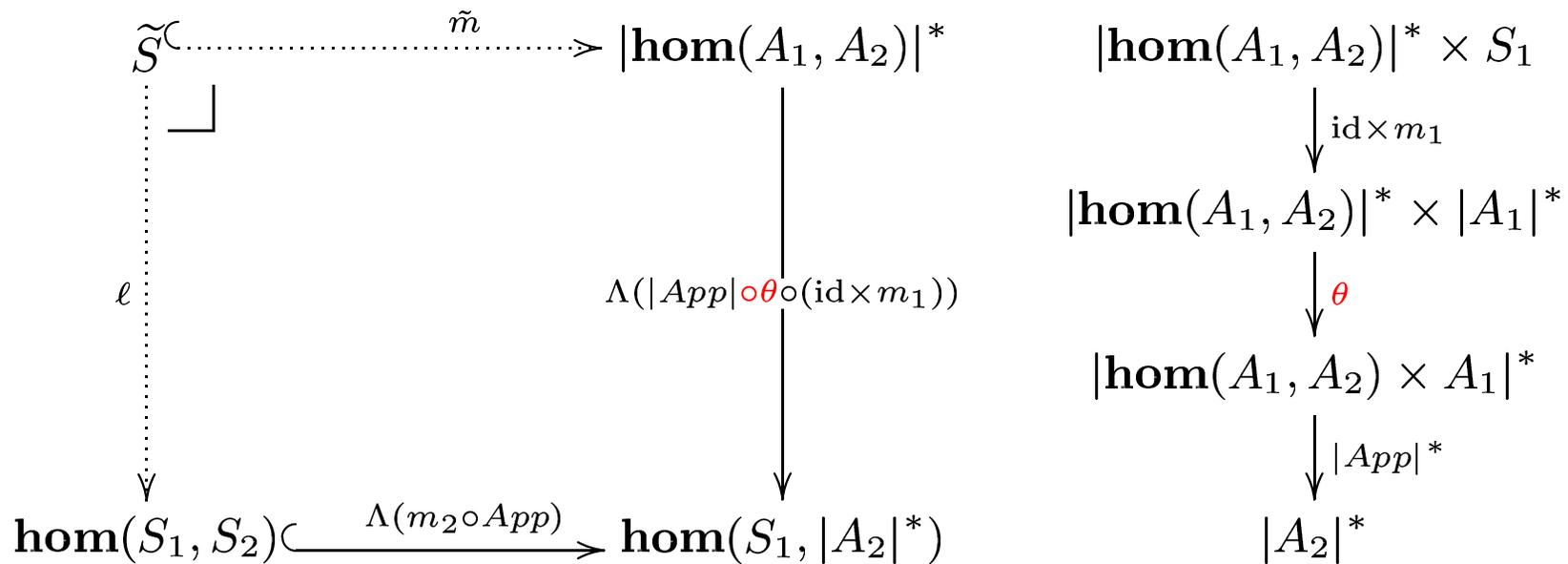
$$\begin{array}{ccc} \vdots & & \downarrow \theta \\ p \downarrow & & \downarrow \theta \\ \overline{S_1 \times S_2} \xrightarrow{m_1 \times m_2} & & |A_1 \times A_2|^* \end{array}$$

- Les projections se construisent par relèvement (transparent suivant);
- L'exponentielle se construit grâce aux pullbacks de  $\mathbb{C}$ ... (à suivre)

# Projections, par relèvement



$$\tilde{S} \hookrightarrow^{\tilde{m}} |\mathbf{hom}(A_1, A_2)|^* = \mathbf{hom}(S_1 \hookrightarrow^{m_1} |A_1|^*, S_2 \hookrightarrow^{m_2} |A_2|^*)$$



$$\mathbf{hom}(S_1, S_2) \times S_1 \xrightarrow{App} S_2 \xrightarrow{m_2} |A_2|^*$$

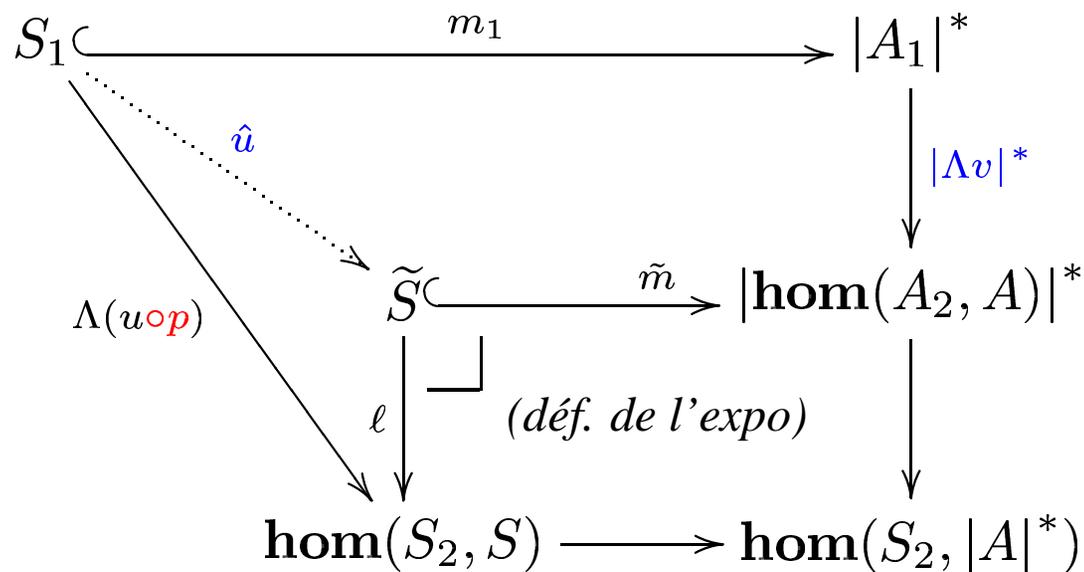
# Application

$$\begin{array}{ccc}
 \tilde{S} \times S_1 & \xrightarrow{\tilde{m} \times \text{id}} & |\mathbf{hom}(A_1, A_2)|^* \times S_1 \xrightarrow{\text{id} \times m_1} |\mathbf{hom}(A_1, A_2)|^* \times |A_1|^* \\
 \downarrow \ell \times \text{id} & \text{(d\'ef. de l'expo)} & \downarrow \\
 \mathbf{hom}(S_1, S_2) \times S_1 & \longrightarrow & \mathbf{hom}(S_1, |A_2|^*) \times S_1 \quad \text{(calcul)} \\
 \downarrow \text{App} & \text{(calcul)} & \downarrow \text{App} \\
 S_2 \hookrightarrow & \xrightarrow{m_2} & |A_2|^* \xleftarrow{|App|^*} |\mathbf{hom}(A_1, A_2) \times A_1|^* \\
 \uparrow \widehat{App} & & \\
 \overline{\tilde{S} \times S_1} & \xrightarrow{\tilde{m} \times m_1} & 
 \end{array}$$

$\mathbf{hom}( S_1 \hookrightarrow^{m_1} |A_1|^* , S_2 \hookrightarrow^{m_2} |A_2|^* ) \times ( S_1 \hookrightarrow^{m_1} |A_1|^* )$

## Abstraction ( $\Lambda(\dots)$ )

Soit  $\langle S_1, m_1, A_1 \rangle \times \langle S_2, m_2, A_2 \rangle \xrightarrow{(u,v)} \langle S, m, A \rangle$ ,  
 on définit  $\Lambda(u, v) = (\hat{u}, \Lambda v)$  tel que:



Voilà, le sous-scone est une CCC, et pourtant  $|_*$  ne préserve pas les produits. (Pour les monades, aucune modification.)

## Résumé

---

On a fabriqué un sous-scone  $\mathbf{C} \downarrow \mathbb{C}^\Delta$ , qui fournit une relation logique **de Kripke** du  $\lambda$ -calcul monadique.

Fonctionne pour toute monade **forte**.

Nécessite juste que  $\mathbb{C}^\Delta$  soit une CCC avec pullbacks:

- pas vrai en général;
- OK si  $\mathbb{C} = \mathbf{Set}^{\mathcal{I}}$  (car  $\mathbb{C}^\Delta \cong \mathbf{Set}^{\mathcal{I} \times \Delta}$  topos élémentaire).

# Plan

---

- Relations logiques dans le  $\lambda$ -calcul (pas crypto);
- Relations logiques dans le  $\lambda$ -calcul monadique; (avec David Nowak et Slawek Lasota)
- Passage aux objets co-simpliciaux;
- Passage à la cohomologie. (beaucoup plus prospectif!)

**Note:** les transparents suivants sont complètement remaniés comparés à ma présentation du 14 février 2003.

## Rappel: complexes de cochaînes (bornés par le bas)

Soit  $\mathcal{A}$  une catégorie abélienne. (par ex., groupes abéliens  $\mathbf{Ab}$ )

Un **complexe de cochaînes** dans  $\mathcal{A}$  est une suite d'objets  $C_n$  de  $\mathcal{A}$

$$\dots \longrightarrow 0 \longrightarrow \dots \longrightarrow 0 \longrightarrow C_{-k} \xrightarrow{d_{-k+1}} C_{-k+1} \xrightarrow{d_{-k}} \dots \xrightarrow{d_n} C_n \xrightarrow{d_{n+1}} \dots$$

tels que  $d_{n+1} \circ d_n = \mathbf{0}$ . (en abrégé,  $d^2 = \mathbf{0}$ )

La catégorie  $\mathbf{CoCh}(\mathcal{A})$  des complexes de cochaînes a comme morphismes les  $(f_n)_{n \in \mathbb{Z}}$  qui font commuter:

$$\begin{array}{ccccccccccc} \dots & \longrightarrow & 0 & \longrightarrow & \dots & \longrightarrow & 0 & \longrightarrow & C_{-k} & \xrightarrow{d_{-k+1}} & C_{-k+1} & \xrightarrow{d_{-k}} & \dots & \xrightarrow{d_n} & C_n & \xrightarrow{d_{n+1}} & \dots \\ & & \downarrow & & & & \downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\ \dots & \longrightarrow & 0 & \longrightarrow & \dots & \longrightarrow & 0 & \longrightarrow & C'_{-k} & \xrightarrow{d_{-k+1}} & C'_{-k+1} & \xrightarrow{d_{-k}} & \dots & \xrightarrow{d_n} & C'_n & \xrightarrow{d_{n+1}} & \dots \end{array}$$

**Note:**  $\mathbf{CoCh}(\mathcal{A})$  est de nouveau une catégorie abélienne.

## Rappel: objets cosimpliciaux et complexes de cochaînes (Barr-Beck)

La catégorie des **objets cosimpliciaux** de  $\mathcal{C}$  est  $\mathcal{C}^\Delta$ .

Tout foncteur  $E : \mathcal{C} \rightarrow \mathcal{A}$  induit un foncteur  $C(\bullet, E) : \mathcal{C}^\Delta \rightarrow \mathbf{CoCh}(\mathcal{A})$ :

- $C(K, E) =$

$$\dots \longrightarrow 0 \longrightarrow \dots \longrightarrow E(K_{-1}) \xrightarrow{d_0} E(K_0) \xrightarrow{d_1} \dots \xrightarrow{d_n} E(K_n) \xrightarrow{d_{n+1}} \dots$$

$$d_n : E(K_{n-1}) \rightarrow E(K_n) = \sum_{i=0}^n (-1)^i E(\partial_n^i)$$

- si  $f : K \rightarrow L$ ,  $C(f, E)$  est:

$$\begin{array}{ccccccccccc} \dots & \longrightarrow & 0 & \longrightarrow & \dots & \longrightarrow & E(K_{-1}) & \xrightarrow{d_0} & E(K_0) & \xrightarrow{d_1} & \dots & \xrightarrow{d_n} & E(K_n) & \xrightarrow{d_{n+1}} & \dots \\ & & \downarrow 0 & & & & \downarrow E(f_{-1}) & & \downarrow E(f_0) & & & & \downarrow E(f_n) & & \\ \dots & \longrightarrow & 0 & \longrightarrow & \dots & \longrightarrow & E(L_{-1}) & \xrightarrow{d_0} & E(L_0) & \xrightarrow{d_1} & \dots & \xrightarrow{d_n} & E(L_n) & \xrightarrow{d_{n+1}} & \dots \end{array}$$

En fait,  $C(\bullet, E)$  est même défini sur les objets co-semi-simpliciaux (cf. p. 37).

## Cohomologie de Barr-Beck

---

Soit  $H^n(K, E) = \ker d_{n+1} / \text{im } d_n$ ,  $n \in \mathbb{N}$ .

Définit une famille de foncteurs  $H^\bullet(-, E) : \mathcal{C}^\Delta \rightarrow \mathcal{A}$ ,

$H^n(-, E)$  est la  $n$ ième **cohomologie à coefficients dans  $E$** .

Exemples:

- $\mathcal{A} = \mathbf{Ab}$ ,  $\mathcal{C} = \mathbf{Set}^{op}$ ,  $E$  foncteur groupe abélien libre  
(co)homologie des ensembles (co)simpliciaux;
- $\mathcal{A} = \mathbf{AMod}$  ( $A$  anneau),  $\mathcal{C} = \mathbf{Grp}^{op}$ ,  $E$  foncteur  $- \otimes_A N$   
homologie des  $A$ -modules  $M$  à coefficients dans  $N$  ( $\text{Tor}_\bullet^A(M, N)$ )

## Cochâmes et structure cartésienne

---

On a:

- $C(K \times L, E) = C(K, E) \otimes C(L, E),$

$$\text{où } (C \otimes C')_n = \coprod_{p+q=n} C_p \otimes C'_q;$$

$$d(x_p \otimes y_q) = dx_p + dy_q$$

- $C(\mathbf{hom}(K, L), E) = \mathbf{hom}(C(K, E), C(L, E)),$

$$\text{où } \mathbf{hom}(C, C')_n = \prod_{p+q=n} \mathbf{hom}(C_p, C'_q),$$

$$d(f \in \mathbf{hom}(C_p, C'_q)) = d_{q+1} \circ f + (-1)^{p+q+1} f \circ d_p.$$

- $- \otimes K \dashv \mathbf{hom}(K, -)$ , on a donc une structure symétrique monoïdale close sur  $\mathbf{CoCh}(\mathcal{A})$ ,

et le foncteur  $C(-, E)$  préserve la structure symétrique monoïdale close de la CCC  $\mathcal{C}$ .

## Homologie et structure cartésienne

---

Si  $\mathcal{A}$  est la catégorie des  $k$ -espaces vectoriels,

$$H^\bullet(K \times L, E) = H^\bullet(K, E) \otimes H^\bullet(L, E), \text{ et}$$

$$H^\bullet(\mathbf{hom}(K, L), E) = \mathbf{hom}(H^\bullet(K, E), H^\bullet(L, E)).$$

En général ( $\mathcal{A}$  catégorie des groupes abéliens ou des  $R$ -modules par ex.), s'ajoutent des Tor (...) aux premiers et des Ext (...) aux seconds. (Théorème des coefficients universels.)

## Coefficients dans le cas de la monade des noms

---

On cherche un foncteur  $E : \mathbb{C} = \mathbf{Set}^{\mathcal{I}} \rightarrow \mathcal{A}$ , où  $\mathcal{A}$  est une catégorie abélienne bien choisie.

Il y a un choix **canonique**:

- $\mathcal{A} = \mathbf{CoCh}(A\mathbf{Mod})$  (complexes de cochaînes de  $A$ -modules)  
(ma préférence personnelle:  $\mathcal{A} = \mathbb{F}_p$  corps fini...)
- $E$  envoie chaque  $(K_n)_{n \geq -1}$  objet de  $\mathbf{Set}^{\mathcal{I}}$  vers le complexe des  $C_n = A[K_n]$  ( $A$ -module libre au-dessus de  $K_n$ ), avec la différentielle évidente (cf. p. 37).

## Piste 1: le sous-scone dans les complexes de cochaînes

---

On avait une sémantique dans  $\mathbb{C}$ , puis une dans  $\mathbb{C}^\Delta$ , donc on pourrait passer à une sémantique dans  $\mathbf{CoCh}(\mathcal{A})$ , via un foncteur coefficients  $E$ ;

Naturellement,  $E$  envoie:

- des produits cartésiens  $\times$  vers des produits tensoriels  $\otimes$ ;
- des exponentielles (cartésiennes) vers des exponentielles (tensorielles).

En fait, dans  $\mathbf{AMod}$ , on a même les projections  $\pi_i : A_1 \otimes A_2 \rightarrow A_i$ .  
(Conjecture: ça passe à  $\mathbf{CoCh}(\mathbf{AMod})$ .)

On peut refaire toute la théorie des sous-scones pratiquement sans modification! (Remplacer  $\times$  par  $\otimes$  partout...)

Mais que serait une catégorie-symétrique-monoïdale-avec-monades-etc. libre? (Nécessaire pour obtenir un diagramme similaire à la p. 25.)

## Piste 2: Comment bien algébriser les sous-scones

---

Remarque fondamentale:  $S \xrightarrow{m} |A|^*$  est essentiellement la donnée d'une **inclusion** de  $S$  dans  $|A|^*$ .

(Attention: tout mono n'est pas en général une inclusion [un mono régulier]...)

Il y a donc un sens à passer à la cohomologie en considérant la **cohomologie relative** de  $|A|^*$  par rapport à  $S$  dans  $\mathbb{C}^\Delta$ .

En somme, au lieu de définir un sous-scone  $\mathcal{C} \downarrow \mathbf{CoCh}(AMod)$  des complexes de chaînes, on définit une homologie relative au-dessus du sous-scone  $\mathcal{C} \downarrow \mathbb{C}^\Delta$ .

## Cohomologie relative sur $S \xrightarrow{m} |A|^*$

On demande que l'image par  $E$  d'un mono pertinent dans  $\mathbb{C}^\Delta$  soit un mono de  $\mathcal{A}$ . (C'est vrai pour le  $E$  canonique de la p. 54.)

On a alors la suite exacte courte:

$$0 \longrightarrow E(S) \xrightarrow{E(m)} E(|A|^*) \xrightarrow{q} \text{coker } E(m) \longrightarrow 0$$

où  $q$  est l'application quotient,  $\text{coker } E(m) = |A|^* / \text{i}_m E(m)$  (“=  $|A|^* / S$ ”)

On en déduit la suite exacte longue classique de cohomologie:

$$\dots \longrightarrow H^n(S) \xrightarrow{E(m)} H^n(|A|^*) \xrightarrow{q} H^n(|A|^* / S) \xrightarrow{\tilde{d}} H^{n+1}(S) \xrightarrow{E(m)} \dots$$

(On abrège  $H^\bullet(-, E)$  en  $H^\bullet(-)$ .)

## Produits, exponentielles, monades, etc.

---

Peut-on calculer:

- $H^\bullet(|A_1 \times A_2|^* / \overline{S_1 \times S_2})$  en fonction des  $H^\bullet(|A_i|^* / S_i)$ ,  $i = 1, 2$ ?  
(Notations: cf. p. 42)

à trouver (Künneth, i.e., coefficients universels?)...

- $H^\bullet(|\mathbf{hom}(A_1, A_2)|^* / \tilde{S})$  en fonction des  $H^\bullet(|A_i|^* / S_i)$ ,  $i = 1, 2$ ?  
(Notations: cf. p. 44)

à trouver (coefficients universels?)...

- $H^\bullet(|TA| / \tilde{S})$ ? (Notations: cf. p. 28)

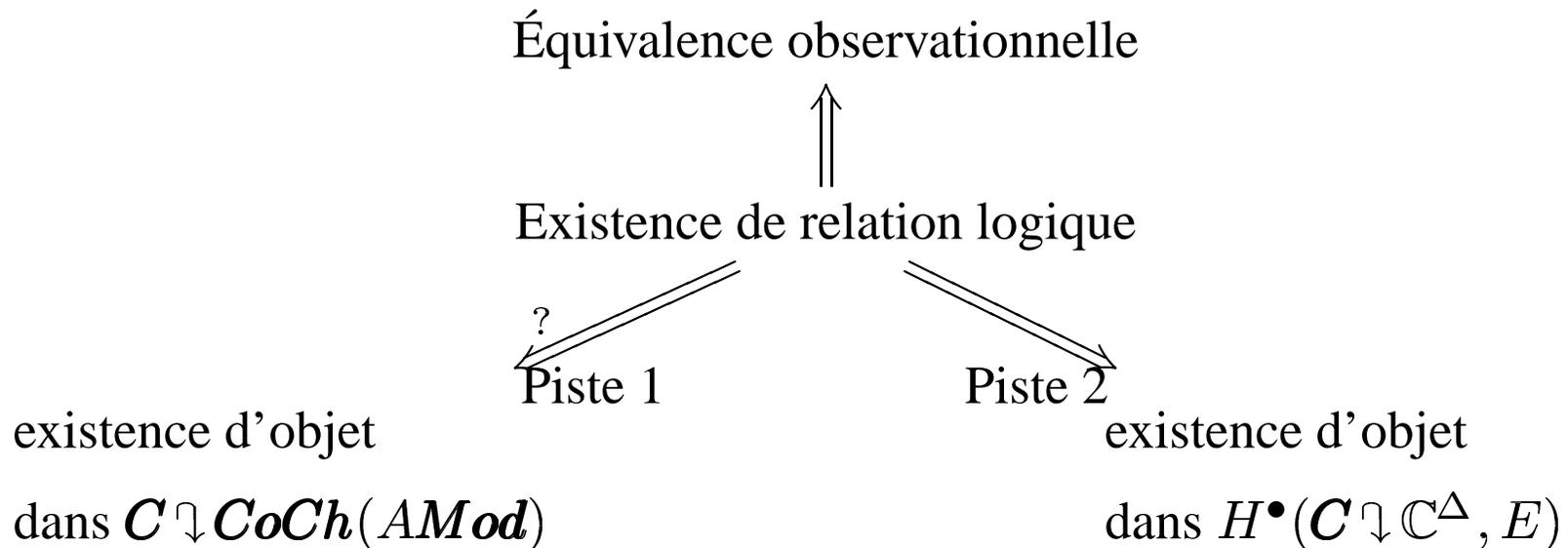
Ça, c'est facile:  $H^n(|TA| / \tilde{S}) = H^{n-1}(A/S)$ .

Et finalement: tout ceci répond-il à la question de l'approximation de l'existence d'une relation logique reliant deux termes donnés?

à trouver...

## Conclusion

Il y a du travail... et les implications sont a priori dans le mauvais sens:



... il manque une implication du bas vers le haut (Hurewicz?)

Peut toujours servir à montrer l'inexistence d'une preuve d'équivalence observationnelle par relation logiques;

donc à l'inéquivalence observationnelle si relations logiques complètes.