Catégories 3: transformations naturelles

Du λ-calcul simplement typé aux catégories monoïdales

Feuille de rut

- Prochain truc que je veux vous montrer : effets de bord.
- Fondé sur les *catégories pré-monoïdales*.
- Incompréhensibles sans avoir vu les *catégories monoïdales*.
- Utilisent la notion de *transformation naturelle*.

$\rightarrow$ Une bien bonne raison pour avancer un peu la technique.

The Walking Natural Transformation (version pour informaticiens) (terminologie © Baez et Dolan)

- On se place dans $C = \mathsf{Cl}(\Lambda_{\mathrm{ST}})$.
- Rappel :
  - Objets : contextes de typage $\Gamma, \Delta, \Theta, \ldots$,
  - Morphismes : tuples typés $\Gamma' \vdash \gamma : \Gamma$.
  - Notation : $\Gamma \vdash \Delta$
- On va voir un genre de transformation naturelle tellement typique que normalement, après, le concept vous paraîtra, comment dire, . . .

Les morphismes comme transformations naturelles

- Pré-composition par $\Gamma' \vdash \gamma'' : \Gamma$ = substitution γ''^{*}_{Δ} :

$$\begin{array}{rcl} \Gamma \vdash \Delta & \longrightarrow & \Gamma' \vdash \Delta \\ (\Gamma \vdash \gamma' : \Delta) & \longmapsto & (\Gamma' \vdash \gamma' \circ \gamma'' : \Delta) \end{array}$$

  indexée par Δ.

- Post-comp. par $\Delta \vdash \gamma : \Delta'$ = *Yoneda embedding* du morphisme γ, noté $y\gamma_{\Gamma}$:

$$\begin{array}{rcl} \Gamma \vdash \Delta & \longrightarrow & \Gamma \vdash \Delta' \\ (\Gamma \vdash \gamma' : \Delta) & \longmapsto & (\Gamma \vdash \gamma \circ \gamma' : \Delta') \end{array}$$

  indexé par Γ.

- On va voir que $y\gamma$ est une transformation naturelle.

The Walking Natural Transformation

Logique linéaire et λ-calcul linéaire

Catégories monoïdales

Naturalité pour Yoneda

La naturalité, c'est la commutation de ces deux opérations :

$$\begin{tikzcd}
\gamma' \arrow[rrr, mapsto, dashed] \arrow[ddd, mapsto, dashed] & & & \gamma \circ \gamma' \arrow[ddd, mapsto, dashed] \\
& \Gamma \vdash \Delta \arrow[r, "y\gamma_\Gamma"] \arrow[d, "\gamma''^*_\Delta"'] & \Gamma \vdash \Delta' \arrow[d, "\gamma''^*_{\Delta'}"] & \\
& \Gamma' \vdash \Delta \arrow[r, "y\gamma_{\Gamma'}"] & \Gamma' \vdash \Delta' & \\
\gamma' \circ \gamma'' \arrow[rrr, mapsto, dashed] & & & \gamma \circ \gamma' \circ \gamma''
\end{tikzcd}$$

autrement dit, c'est juste l'associativité de la composition.

DESSIN QUI SAUVE

Yoneda sur un exemple : l'application

- Soient $\begin{pmatrix} \Delta & = & (x : \tau^{\tau'}, y : \tau') \\ \Delta' & = & (z : \tau) \end{pmatrix}$.
- La post-comp. voit l'application du λ-calcul

$$\gamma = (\Delta \vdash x\ y : \Delta')$$

comme la fonction $app_\Gamma = y\gamma_\Gamma$:

$$\begin{array}{lcl} \Gamma \vdash \Delta & \longrightarrow & \Gamma \vdash \Delta' \\ (\Gamma \vdash (e, e') : \Delta) & \longmapsto & (\Gamma \vdash (e\ e') : \Delta') \end{array}$$

indexée par Γ.

Yoneda sur un exemple : l'application (suite)

- On a toujours $\begin{pmatrix} \Delta & = & (x : \tau^{\tau'}, y : \tau') \\ \Delta' & = & (z : \tau) \end{pmatrix}$.
- La naturalité dit

$$\begin{tikzcd}
(e, e') \arrow[rrr, dashed, mapsto] \arrow[ddd, dashed, mapsto] & & & (e\ e') \arrow[ddd, dashed, mapsto] \\
& \Gamma \vdash \Delta \arrow[r, "app_\Gamma"] \arrow[d, "\gamma''^*_\Delta"'] & \Gamma \vdash \Delta' \arrow[d, "\gamma''^*_{\Delta'}"] & \\
& \Gamma' \vdash \Delta \arrow[r, "app_{\Gamma'}"] & \Gamma' \vdash \Delta' & \\
(\gamma''^* e, \gamma''^* e') \arrow[rrr, dashed, mapsto] & & & \gamma''^*(e\ e') = ((\gamma''^* e)\ (\gamma''^* e'))
\end{tikzcd}$$

Plus compliqué : l'abstraction

- Soit $abs_\Gamma^{\tau',\tau} : \begin{array}{ccc} \Gamma, x : \tau' \vdash \tau & \longrightarrow & \Gamma \vdash \tau^{\tau'} \\ \Gamma, x : \tau' \vdash e : \tau & \longmapsto & \Gamma \vdash \lambda x.e : \tau^{\tau'}. \end{array}$

$$\begin{tikzcd}
\Gamma & e \arrow[rrr, dashed, mapsto] \arrow[ddd, dashed, mapsto] & & & \lambda x.e \arrow[ddd, dashed, mapsto] \\
& & \Gamma, x : \tau' \vdash \tau \arrow[r, "abs_\Gamma"] \arrow[d, "{(\gamma'', x)^*}"'] & \Gamma \vdash \tau^{\tau'} \arrow[d, "\gamma''^*"] & \\
& & \Gamma', x : \tau' \vdash \tau \arrow[r, "abs_{\Gamma'}"] & \Gamma' \vdash \tau^{\tau'} & \\
\Gamma' \arrow[uuu, "\gamma''"] & \gamma''^* e \arrow[rrr, dashed, mapsto] & & & \gamma''^*(\lambda x.e) = \lambda x.\gamma''^* e
\end{tikzcd}$$

- Rappel pour la capture.
  - On a écrit $\Gamma, x : \tau$ et $\Gamma', x : \tau$.
  - $\rightarrow$ Implicitement, x n'apparaît pas dans Γ ou Γ'.

Plus compliqué : l'abstraction

- Soit $$\begin{array}{rrcl} abs_\Gamma : & \Gamma, x : \tau' \vdash \tau & \longrightarrow & \Gamma \vdash \tau^{\tau'} \\ & \Gamma, x : \tau' \vdash e : \tau & \longmapsto & \Gamma \vdash \lambda x.e : \tau^{\tau'}. \end{array}$$

$$\begin{tikzcd}
\Gamma & e \arrow[rrr, dashed, mapsto] \arrow[ddd, dashed, mapsto] & & & \lambda x.e \arrow[ddd, dashed, mapsto] \\
& & \Gamma, x : \tau' \vdash \tau \arrow[r, "abs_\Gamma"] \arrow[d, "(\gamma'', x)^*"'] & \Gamma \vdash \tau^{\tau'} \arrow[d, "\gamma''^*"] & \\
& & \Gamma', x : \tau' \vdash \tau \arrow[r, "abs_{\Gamma'}"] & \Gamma' \vdash \tau^{\tau'} & \\
\Gamma' \arrow[uuu, "\gamma''"] & \gamma''^* e \arrow[rrr, dashed, mapsto] & & & \gamma''^*(\lambda x.e) = \lambda x.\gamma''^* e
\end{tikzcd}$$

- Rappel pour la capture.
  - On a écrit $\Gamma, x : \tau$ et $\Gamma', x : \tau$.
  - $\rightarrow$ Implicitement, x n'apparaît pas dans Γ ou Γ'.

Bilan :

- Pas encore de définition générale.
- Trois exemples de transformation naturelles à base de Cl(Λ_{ST}) :
  - $y\gamma$: composition avec un morphisme (on voit de haut).
  - *app* : cas particulier, composition avec l'application.
  - *abs* : pas un cas particulier. $e \mapsto \lambda x.e$.
- La naturalité dit des choses sur l'interaction syntaxe / substitution.

Bilan :

- Catégoriquement, post-composition = associativité de la composition.
- C'est bon, on le voit déjà dans $\mathcal{C}$.
- Pour le troisième, il se passe un truc bizarre :
  - On voit pas *abs* dans $\mathcal{C}$.
  - $\rightarrow$ On voit plus de trucs comme transformations naturelles.
- C'est vrai de manière générale : lemme de Yoneda.
- Ça dit, très vaguement,

> Une catégorie correspond exactement aux transformations naturelles entre foncteurs *Hom*.

On verra le détail plus tard.

Transformations naturelles

Définition

Soient F et G deux foncteurs $\mathcal{C} \to \mathcal{D}$.
Une transformation naturelle $\alpha : F \to G$ est un ensemble de morphismes indexé par $Obj(\mathcal{C})$

$$\{\alpha_A : F(A) \to G(A)\}_{A \in Obj(\mathcal{C})}$$

tel que pour tout $f : A \to B$

$$\begin{tikzcd} A \arrow[d, "f"'] & F(A) \arrow[r, "\alpha_A"] \arrow[d, "F(f)"'] & G(A) \arrow[d, "G(f)"] \\ B & F(B) \arrow[r, "\alpha_B"] & G(B) \end{tikzcd}$$

commute.

Là normalement, vous protestez

Là normalement, vous protestez

Pour comprendre pourquoi nos exemples rentrent dans le cadre général, quelques outils.

- L'*opposée* d'une catégorie $\mathcal{C}$: $\mathit{dom} \leftrightarrows \mathit{cod}$. (inverser le sens des flèches).
- Un foncteur de $\mathcal{C}^{op} \to \mathcal{D}$ est juste un foncteur qui inverse le sens des flèches. On dit foncteur *contravariant* de $\mathcal{C}$ vers $\mathcal{D}$.

Raccrochage au cas général

- Pour *app* :
$$\begin{array}{rcl} \Delta & = & x : \tau^{\tau'}, y : \tau' \\ \mathcal{C} & = & Cl(\Lambda_{\mathrm{ST}})^{op} \\ \mathcal{D} & = & \mathbf{Set} \\ F & = & Hom(\cdot\ ;\ \Delta) = \left| \begin{array}{ccc} \Gamma & \mapsto & \Gamma \vdash \Delta \\ \gamma & \mapsto & \gamma^*. \end{array} \right. \end{array}$$
- On a bien,
$$\begin{array}{rcccl} \text{pour } \gamma & : & \Gamma' & \to & \Gamma, \\ F(\gamma) & : & F(\Gamma) & \to & F(\Gamma'), \text{ i.e.,} \\ \gamma^* & : & \Gamma \vdash \Delta & \to & \Gamma' \vdash \Delta. \end{array}$$
- On a aussi pour $\Gamma'' \xrightarrow{\gamma'} \Gamma' \xrightarrow{\gamma} \Gamma,$
$$\begin{array}{rcl} F(\gamma \circ \gamma') & = & (\gamma \circ \gamma')^* = (\gamma'^*(\gamma))^* = \gamma'^* \circ \gamma^* \\ & = & F(\gamma') \circ F(\gamma) \end{array}$$

  et $F(id) = id$.
- Donc F est un foncteur.

Raccrochage au cas général

- De même :

$$\begin{array}{lcl} \Delta' & = & z : \tau \\ \mathcal{C} & = & \mathrm{Cl}(\Lambda_{\mathrm{ST}})^{op} \\ \mathcal{D} & = & \mathbf{Set} \\ G & = & \mathit{Hom}(\cdot\ ;\ \Delta') = \left| \begin{array}{lcl} \Gamma & \mapsto & \Gamma \vdash \Delta' \\ \gamma & \mapsto & \gamma^* \end{array} \right. \end{array}$$

  donne un foncteur.

- En passant : evidemment ça marche pour n'importe quel objet Θ.

$$y(\Theta) \quad : \quad \begin{array}{lcl} \mathcal{C}^{op} & \to & \mathbf{Set} \\ \Gamma & \mapsto & \Gamma \vdash \Delta' \\ \gamma & \mapsto & \gamma^* \end{array}$$

  (chaque objet de $\mathcal{C}$ correspond à un foncteur $\mathcal{C}^{op} \to$ **Set**).

Raccrochage au cas général

$$\begin{tikzcd}
A \arrow[d, "f"'] & F(A) \arrow[r, "\alpha_A"] \arrow[d, "F(f)"'] & G(A) \arrow[d, "G(f)"] \\
B & F(B) \arrow[r, "\alpha_B"] & G(B)
\end{tikzcd}$$

Diagramme appliqué (seule la contravariance change) :

$$\begin{tikzcd}
 & (e, e') \arrow[rrr, dashed, mapsto] \arrow[ddd, dashed, mapsto] & & & (e\ e') \arrow[ddd, dashed, mapsto] \\
\Gamma & & \Gamma \vdash \Delta \arrow[r, "app_\Gamma"] \arrow[d, "\gamma''^*_\Delta"'] & \Gamma \vdash \Delta' \arrow[d, "\gamma''^*_{\Delta'}"] & \\
\Gamma' \arrow[u, "\gamma''"] & & \Gamma' \vdash \Delta \arrow[r, "app_{\Gamma'}"] & \Gamma' \vdash \Delta' & \\
 & (\gamma''^* e, \gamma''^* e') \arrow[rrr, dashed, mapsto] & & & \gamma''^*(e\ e') = ((\gamma''^* e)\ (\gamma''^* e'))
\end{tikzcd}$$

Raccrochage au cas général (abstraction)

De (presque) la même façon, *abs* est une transformation naturelle de

$$Hom(\cdot \times \tau' \ ; \ \tau) \to Hom(\cdot \ ; \ \tau^{\tau'}).$$

Bilan

- Transformation naturelle : très général, pour deux foncteurs $\mathcal{C} \to \mathcal{D}$.
- Cas particulier : $\mathcal{C} = \mathsf{Cl}(\Lambda_{\mathrm{ST}})^{op}$ et $\mathcal{D} = \mathbf{Set}$, Yoneda, post-composition.
- Cas particulier plus dur : abstraction (toujours $\mathcal{C} = \mathsf{Cl}(\Lambda_{\mathrm{ST}})^{op}$ et $\mathcal{D} = \mathbf{Set}$).

Bilan moral

- La naturalité dit que tout se passe bien avec la pré-composition (chez nous la substitution).
- On peut caractériser l'existence de constructions syntaxiques par l'existence de transformations naturelles.
- Demi-exemple : abstraction. On peut caractériser $\mathrm{Cl}(\Lambda_{\mathrm{ST}})$ de façon encore plus satisfaisante que les CCC par l'existence de transformations naturelles dotées de certaines propriétés.
- On va utiliser les transformations naturelles pour définir les catégories monoïdales.
- N'ayez pas peur de la naturalité.

Vers les catégories monoïdales

- Comme pour les CCC et le λ-calcul, il y a l'exemple canonique qui va avec.
- Ici c'est la *logique linéaire intuitionniste multiplicative* propositionnelle (IMLL).

Calcul des séquents IMLL (canal historique)

- Formules atomiques : atomes a.
- Formules : $F ::= a \mid F \multimap F' \mid F \otimes F' \mid I$.
- Séquents : listes finies de formules Γ.
- Juste après, les règles de déduction, mais avant, bande annonce :
  - pas de duplication ni d'effacement d'hypothèse (cf. information quantique),
  - intuitionniste (ici : exactement une formule a droite),
  - mêmes règles que d'hab, mais sans contraction et affaiblissement,
  - subtilité sur le choix de présentation (additif vs. mult.),
  - séquents = intro droite / intro gauche
déduction naturelle = intro droite / élim droite.

IMLL : règles structurelles

$$\frac{}{a \vdash a}\,\textsc{Ax} \qquad \frac{\Gamma, F, F', \Gamma' \vdash G}{\Gamma, F', F, \Gamma' \vdash G}\,\textsc{Exch} \qquad \frac{\Gamma \vdash F' \qquad F', \Gamma' \vdash F}{\Gamma, \Gamma' \vdash F}\,\textsc{Cut}$$

- L'axiome ne permet pas de traîner des hypothèses en plus.
- Echange : facile (retenir qu'il y a un ordre).
- Coupure : explication en termes de ressources.

IMLL : conjonction

R$\otimes$
$$\frac{\Gamma \vdash F \qquad \Gamma' \vdash F'}{\Gamma, \Gamma' \vdash F \otimes F'}$$

L$\otimes$
$$\frac{\Gamma, F, F' \vdash G}{\Gamma, F \otimes F' \vdash G}$$

- Règles normales pour $\wedge$, présentation multiplicative.
- Le tenseur c'est le groupement de deux ressources.
- On peut pas projeter sur une composante ou l'autre.
- On peut extraire les deux composantes en même temps (cf. pattern matching).

IMLL : implication

$$\mathrm{R}\multimap\ \frac{\Gamma, F \vdash F'}{\Gamma \vdash F \multimap F'} \qquad \mathrm{L}\multimap\ \frac{\Gamma \vdash F \qquad F', \Gamma' \vdash G}{\Gamma, F \multimap F', \Gamma' \vdash G}$$

- Règles normales pour $\Rightarrow$.
- L'implication linéaire reçoit la ressource F qu'elle utilise exactement une fois pour produire la ressource F'.
- Explication en termes de ressources :
  - on utilise Γ pour produire F,
  - on utilise ce F et $F \multimap F'$ pour produire un F',
  - on utilise ce F' et Γ' pour produire G.

IMLL : neutre

$$\text{R}I \quad \epsilon \vdash I \qquad\qquad \text{L}I \quad \frac{\Gamma \vdash F}{\Gamma, I \vdash F}$$

- I est le truc gratos.
- On sait le produire et le consommer sans que ça compte.
- Par contre[1], on sait rien en faire.

[1]Et j'insiste sur le “par contre”.

IMLL : résumé

Ax
$$a \vdash a$$

R$\otimes$
$$\frac{\Gamma \vdash F \qquad \Gamma' \vdash F'}{\Gamma, \Gamma' \vdash F \otimes F'}$$

L$\otimes$
$$\frac{\Gamma, F, F' \vdash G}{\Gamma, F \otimes F' \vdash G}$$

R$\multimap$
$$\frac{\Gamma, F \vdash F'}{\Gamma \vdash F \multimap F'}$$

L$\multimap$
$$\frac{\Gamma \vdash F \qquad F', \Gamma' \vdash G}{\Gamma, F \multimap F', \Gamma' \vdash G}$$

RI
$$\epsilon \vdash I$$

LI
$$\frac{\Gamma \vdash F}{\Gamma, I \vdash F}$$

Exch
$$\frac{\Gamma, F, F', \Gamma' \vdash G}{\Gamma, F', F, \Gamma' \vdash G}$$

Cut
$$\frac{\Gamma \vdash F' \qquad F', \Gamma' \vdash F}{\Gamma, \Gamma' \vdash F}$$

Quelques propriétés

(N'essayez pas de les démontrer.)

- $\Gamma, F \vdash G$ ssi $\Gamma \vdash F \multimap G$.
- $\Gamma, F_1, F_2 \vdash G$ ssi $\Gamma, (F_1 \otimes F_2) \vdash G$.
- En particulier, ($F \vdash G$ et $G \vdash F$) ssi $\vdash F \multimapboth G$, avec $F \multimapboth G = ((F \multimap G) \otimes (G \multimap F))$.
- $\vdash F_1 \multimap (F_2 \multimap F_3)$ ssi $\vdash (F_1 \otimes F_2) \multimap F_3$.

Manque de bol

- Les catégories marchent mieux en déduction naturelle.
- Pas encore trop compris pourquoi.
- En tout cas, Curry-Howard est plus proche de la prog en déduction naturelle (c'est du λ-calcul).
- Donc on va refaire IMLL en déduction naturelle,
- directement avec les termes de preuve.

IMLL : règles structurelles

$$\textsc{Ax}\quad \overline{x : a \vdash x : a}$$

$$\textsc{Exch}\quad \frac{\Gamma, x : F, y : F', \Gamma' \vdash e : G}{\Gamma, y : F', x : F, \Gamma' \vdash e : G}$$

- En ND, la notion coupure correspond à celle de redex.
- Donc ça disparait.

IMLL : conjonction

$$\mathrm{I}\otimes \; \frac{\Gamma \vdash e : F \qquad \Gamma' \vdash e' : F'}{\Gamma, \Gamma' \vdash e \otimes e' : F \otimes F'}$$

$$\mathrm{E}\otimes \; \frac{\Gamma \vdash e : F \otimes F' \qquad \Gamma', x : F, y : F' \vdash e' : G}{\Gamma, \Gamma' \vdash \texttt{let}\ (x, y) = e\ \texttt{in}\ e' : G}$$

- La règle multiplicative d'élimination du “et” est inhabituelle.

The Walking Natural Transformation

Logique linéaire et λ-calcul linéaire

Catégories monoïdales

IMLL : implication

$$\frac{\Gamma, x : F \vdash e : F'}{\Gamma \vdash \lambda x.e : F \multimap F'}\ \mathrm{I}\multimap \qquad \frac{\Gamma \vdash e : F' \multimap F \qquad \Gamma' \vdash e' : F'}{\Gamma, \Gamma' \vdash e\, e' : F \multimap F'}\ \mathrm{E}\multimap$$

- Règles normales pour $\Rightarrow$.

IMLL : neutre

$$\frac{\mathrm{I}I}{\epsilon \vdash \star : I}$$

$$\mathrm{E}I \quad \frac{\Gamma \vdash e : I \qquad \Gamma' \vdash e' : F'}{\Gamma, \Gamma' \vdash \mathtt{let}\ \star = e\ \mathtt{in}\ e' : F'}$$

Résumé de la syntaxe

$$
\begin{array}{rcccc}
e & ::= & x & & \\
 & \mid & e \otimes e' & \mid & \texttt{let}\ (x, y) = e\ \texttt{in}\ e' \\
 & \mid & \lambda x.e & \mid & e\ e' \\
 & \mid & \star & \mid & \texttt{let}\ \star = e\ \texttt{in}\ e'.
\end{array}
$$

The Walking Natural Transformation

Logique linéaire et λ-calcul linéaire

Catégories monoïdales

Théorie équationnelle (implication)

$$\textsc{Beta}\quad \frac{\Gamma, x : F \vdash e : F' \qquad \Gamma' \vdash e' : F'}{\Gamma, \Gamma' \vdash (\lambda x.e)\ e' = [x \mapsto e'](e) : F}$$

$$\textsc{Eta}\quad \frac{\Gamma \vdash e : F' \multimap F \qquad x \notin FV(e)}{\Gamma \vdash \lambda x.(e\ x) = e : F' \multimap F}$$

Théorie équationnelle (conjonction)

The Walking Natural Transformation

Logique linéaire et λ-calcul linéaire

Catégories monoïdales

$$\textsc{Match}\quad \frac{\Gamma \vdash e : F \qquad \Gamma' \vdash e' : F' \qquad \Gamma'', x : F, y : F' \vdash e'' : F''}{\Gamma, \Gamma', \Gamma'' \vdash \left(\begin{array}{l} \texttt{let}\ (x, y) = e \otimes e'\ \texttt{in}\ e'' \\ = [x \mapsto e, y \mapsto e'](e'') \end{array} \right) : F''}$$

$$\textsc{Eta-Match}\quad \frac{\Gamma \vdash e : F \otimes F' \qquad x \notin FV(e)}{\Gamma \vdash e = (\texttt{let}\ (x, y) = e\ \texttt{in}\ x \otimes y) : F \otimes F'}$$

Théorie équationnelle (neutre)

$$\textsc{Neutre}\ \frac{\Gamma \vdash e : F}{\Gamma \vdash (\mathtt{let}\ \star = \star\ \mathtt{in}\ e) = e : F}$$

$$\textsc{Eta-Neutre}\ \frac{\Gamma \vdash e : I}{\Gamma \vdash (\mathtt{let}\ \star = e\ \mathtt{in}\ \star) = e : I}$$

The Walking Natural Transformation
Logique linéaire et λ-calcul linéaire
Catégories monoïdales

Théorie équationnelle (le reste inavouable)

Plein de règles très syntaxiques, pour dire que le `let` ... `in` ... commute à tout le monde, genre :

$$\frac{}{\vdash \left(\begin{array}{l} (\mathtt{let}\ (x, y) = (\mathtt{let} \star = e\ \mathtt{in}\ e')\ \mathtt{in}\ e'') \\ = (\mathtt{let} \star = e\ \mathtt{in}\ \mathtt{let}\ (x, y) = e'\ \mathtt{in}\ e'') \end{array} \right) :}$$

- Benton (1995) en annonce 12. Pas vérifié.
- Terminologie standard pour ce genre de règles : *commuting conversions*.

Bilan

- Remarque technique : les règles utilisant la substitution sont bien définies parce que l'équivalent linéaire du lemme de substitution est vrai.

$$\frac{\Gamma, x : F', \Gamma' \vdash e : F \qquad \Gamma \vdash e' : F'}{\Gamma, \Gamma' \vdash [x \mapsto e'](e) : F}$$

  est admissible.

Bilan

- Remarque technique : les règles utilisant la substitution sont bien définies parce que l'équivalent linéaire du lemme de substitution est vrai.

$$\frac{\Gamma, x : F', \Gamma' \vdash e : F \qquad \Gamma'' \vdash e' : F'}{\Gamma, \Gamma'', \Gamma' \vdash [x \mapsto e'](e) : F}$$

est admissible.

Bilan

- Remarque technique : les règles utilisant la substitution sont bien définies parce que l'équivalent linéaire du lemme de substitution est vrai.

$$\frac{\Gamma, x : F', \Gamma' \vdash e : F \qquad \Gamma'' \vdash e' : F'}{\Gamma, \Gamma'', \Gamma' \vdash [x \mapsto e'](e) : F}$$

  est admissible.
- On comprend à peu près ce qu'on fait du point de vue prog.
- Mais ça devient très lourd techniquement (notamment bonjour les preuves en Coq).

Catégories monoïdales

Définition

Une catégorie monoïdale est une catégorie $\mathcal{C}$ équipée d'un foncteur $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, d'un objet I de $\mathcal{C}$, et d'isomorphismes naturels :

- $\alpha_{A,B,C} : A \otimes (B \otimes C) \to (A \otimes B) \otimes C$
  $\lambda_A : I \otimes A \to A$
  $\rho_A : A \otimes I \to A$

tels que les 3 diagrammes de cohérence suivants commutent.

Catégories monoïdales strictes

Définition

Une catégorie monoïdale $\mathcal{C}$ est stricte ssi

- $A \otimes (B \otimes C) = (A \otimes B) \otimes C$,
- $I \otimes A = A = A \otimes I$ *et*
- α, ρ, λ *sont des identités.*

Catégories monoïdales commutatives

Définition

Une catégorie monoïdale $\mathcal{C}$ est commutative ssi elle vient avec un isomorphisme $\gamma_{A,B} : A \otimes B \to B \otimes A$ (désolé pour la collision de notation) tel que les trois diagrammes de cohérence suivants commutent.

Premières propriétés

- Interchange law et whiskering.
- Représentation graphique avec et sans boîtes.
- Naturalité.