Programmes, preuves et fonctions

le ménage à trois de Curry-Howard

Emmanuel Beffara et Lionel Vaux

Institut de Mathématiques de Luminy
CNRS & Université d'Aix-Marseille

École Jeunes Chercheurs en Informatique Mathématique
Perpignan, 8–12 avril 2013

Plan général

1. La correspondance de Curry-Howard (le matin)
   - logique : vérité et preuves
   - λ-calcul et élimination des coupures
2. Sémantique dénotationnelle et logique linéaire (digestion)
3. Au delà du fonctionnel (sieste)

Première partie

Cours accéléré de logique

Valeurs de vérité

Déduction naturelle

Intuitionnisme

Les questions des fondements

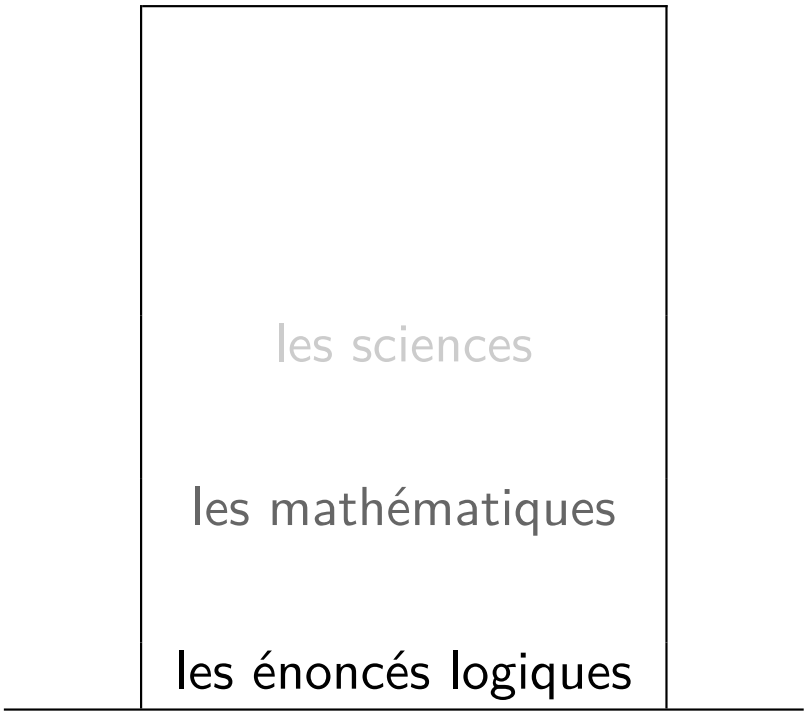
Le langage mathématique s'écrit (virtuellement) avec des symboles :

$$\forall n \Big(n \geq 3 \rightarrow \forall x \forall y \forall z \Big(x^n + y^n = z^n \ \rightarrow \ x = 0 \vee y = 0 \Big) \Big)$$

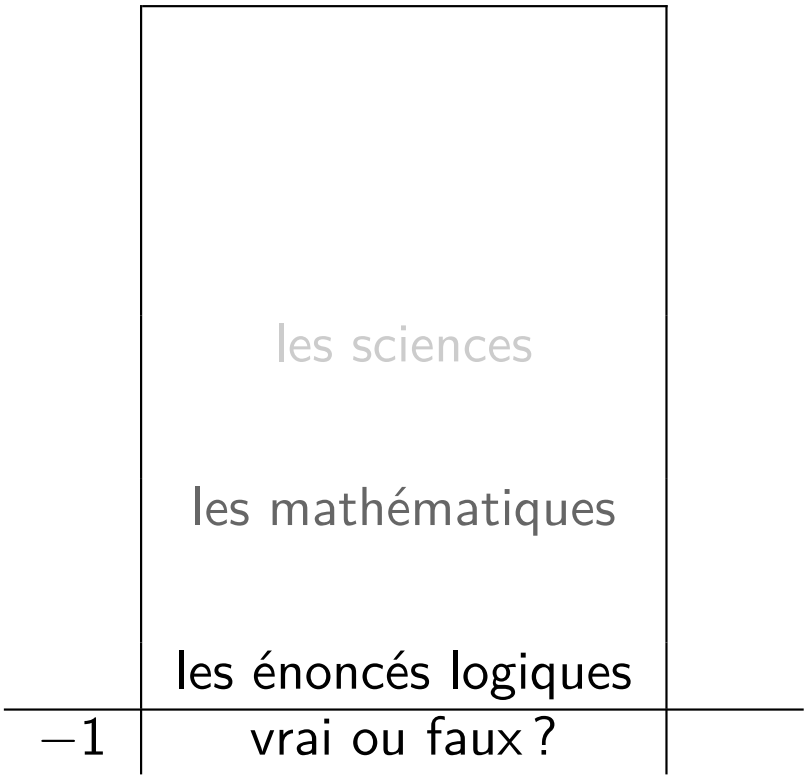
Certaines suites de symboles disent des choses vraies, d'autres pas.

- Qu'est-ce qu'un énoncé bien formé ?
- Qu'est-ce qui est vrai, qu'est-ce qui est faux ?
- Qu'est-ce qui constitue une démonstration correcte ?
- Comment justifier tout ça ?

Les sous-sols de la logique

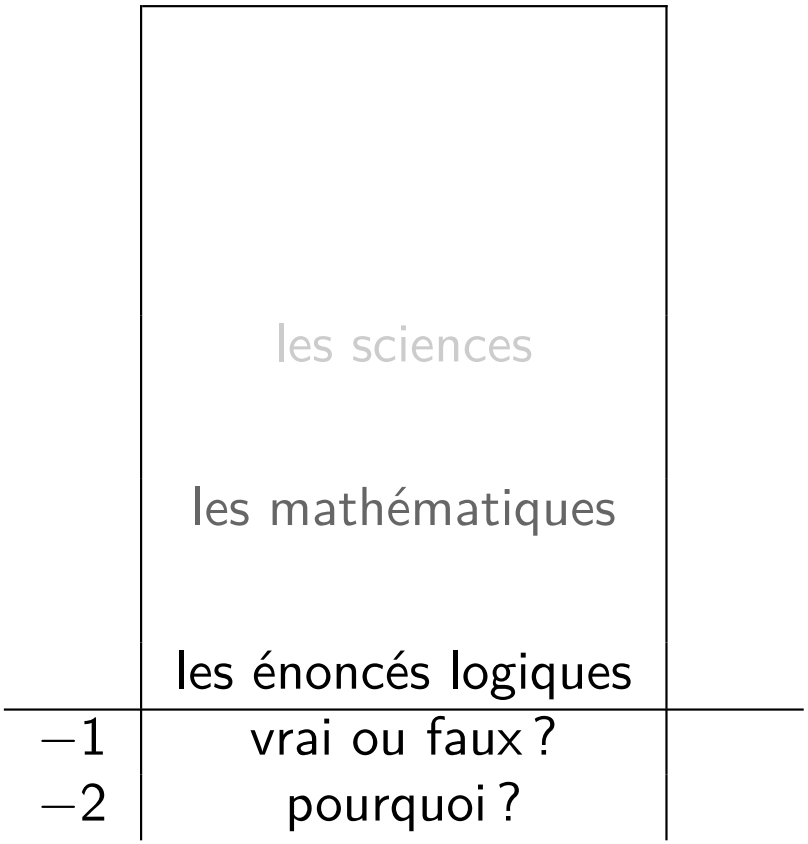


Les sous-sols de la logique



valeurs de vérité, modèles

Les sous-sols de la logique



Formules logiques

On se donne un langage de formules :

$A, B :=$	α	énoncé atomique
	$A \to B$	implication
	$A \wedge B$	conjonction
	$A \vee B$	disjonction
	$\neg A$	négation
	$A \leftrightarrow B$	équivalence
	$\top$	vrai
	$\bot$	faux
	$\forall x.A$	quantification universelle
	$\exists x.A$	quantification existentielle

Formules logiques

On se donne un langage de formules propositionnelles :

$A, B :=$	X	variable propositionnelle
	$A \to B$	implication
	$A \wedge B$	conjonction
	$A \vee B$	disjonction
	$\neg A$	négation
	$A \leftrightarrow B$	équivalence
	$\top$	vrai
	$\bot$	faux

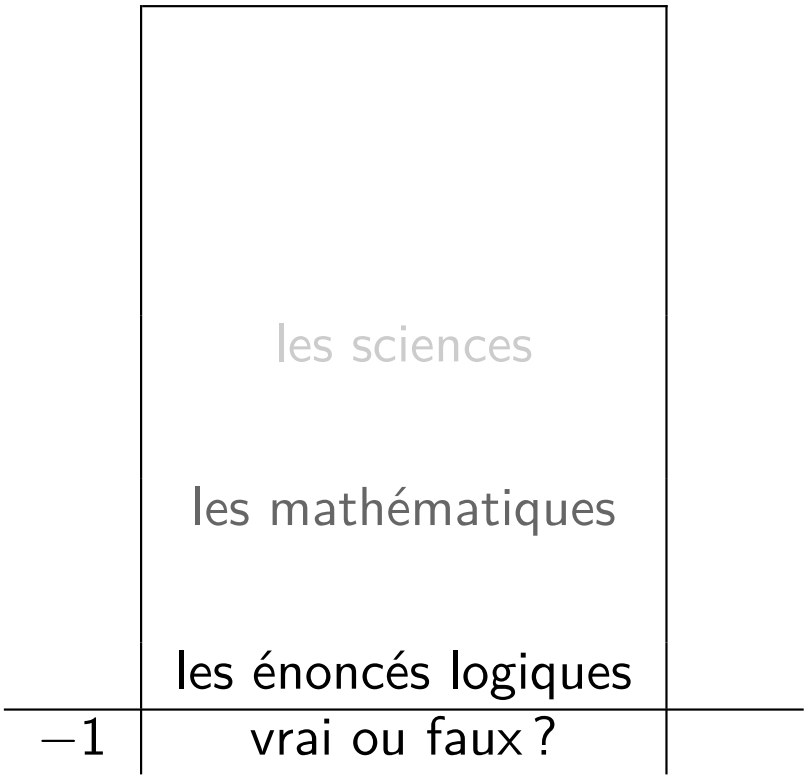
Formules logiques

On se donne un langage de formules propositionnelles :

$$
\begin{array}{rll}
A, B := & X & \text{variable propositionnelle} \\
& A \to B & \text{implication} \\
& A \wedge B & \text{conjonction} \\
& A \vee B & \text{disjonction} \\
& \neg A & \text{négation} \\
& A \leftrightarrow B & \text{équivalence} \\
& \top & \text{vrai} \\
& \bot & \text{faux}
\end{array}
$$

- Au -1, on interprète les formules par des valeurs de vérité : vrai et faux (dans le cas le plus simple).
- Au -2, on formalise les démonstrations. Théorème de complétude, etc.

Les sous-sols de la logique



Les tables de la vérité (vraie)

Connecteurs

Les connecteurs sont des fonctions booléennes :

A	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$	$\neg A$	$\bot$	$\top$
0	0	0	0	1	1	1	0	1
0	1	0	1	1	0	1	0	1
1	0	0	1	0	0	0	0	1
1	1	1	1	1	1	0	0	1

Les tables de la vérité (vraie)

Connecteurs

Les connecteurs sont des fonctions booléennes :

A	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$	$\neg A$	$\bot$	$\top$
0	0	0	0	1	1	1	0	1
0	1	0	1	1	0	1	0	1
1	0	0	1	0	0	0	0	1
1	1	1	1	1	1	0	0	1

Les tables de la vérité (vraie)

Variables

Les variables propositionnelles sont des... variables dans $\{0, 1\}$.

Les tables de la vérité (vraie)

Variables

Les variables propositionnelles sont des. . . variables dans $\{0, 1\}$.

Une *distribution de valeurs de vérité* (dvv)
= un choix de valeurs pour les variables
= un numéro de ligne dans une table de vérité

Les tables de la vérité (vraie)

Formules

On en déduit les valeurs des formules :

A	B	$A \to \neg B$	
0	0	1	
0	1	1	
1	0	1	
1	1	0	

A	B	$A \wedge B$	$A \vee B$	$A \to B$	$A \leftrightarrow B$	$\neg A$	$\bot$	$\top$
0	0	0	0	1	1	1	0	1
0	1	0	1	1	0	1	0	1
1	0	0	1	0	0	0	0	1
1	1	1	1	1	1	0	0	1

Les tables de la vérité (vraie)

Formules

On en déduit les valeurs des formules :

A	B	$A \to \neg B$	$B \wedge (A \to \neg B)$	
0	0	1	0	
0	1	1	1	
1	0	1	0	
1	1	0	0	

A	B	$A \wedge B$	$A \vee B$	$A \to B$	$A \leftrightarrow B$	$\neg A$	$\bot$	$\top$
0	0	0	0	1	1	1	0	1
0	1	0	1	1	0	1	0	1
1	0	0	1	0	0	0	0	1
1	1	1	1	1	1	0	0	1

Les tables de la vérité (vraie)

Formules, équivalences

On en déduit les valeurs des formules :

A	B	$A \to \neg B$	$B \wedge (A \to \neg B)$	$B \wedge \neg A$	
0	0	1	0	0	
0	1	1	1	1	
1	0	1	0	0	
1	1	0	0	0	

- Deux formules sont *équivalentes* si elles ont les mêmes valeurs.

A	B	$A \wedge B$	$A \vee B$	$A \to B$	$A \leftrightarrow B$	$\neg A$	$\bot$	$\top$
0	0	0	0	1	1	1	0	1
0	1	0	1	1	0	1	0	1
1	0	0	1	0	0	0	0	1
1	1	1	1	1	1	0	0	1

Les tables de la vérité (vraie)

Formules, équivalences, tautologies

On en déduit les valeurs des formules :

A	B	$A \to \neg B$	$B \wedge (A \to \neg B)$	$B \wedge \neg A$	$A \vee (A \to \neg B)$	
0	0	1	0	0	1	
0	1	1	1	1	1	
1	0	1	0	0	1	
1	1	0	0	0	1	

- Deux formules sont *équivalentes* si elles ont les mêmes valeurs.
- Une *tautologie* est une formule vraie (pour toute dvv)

A	B	$A \wedge B$	$A \vee B$	$A \to B$	$A \leftrightarrow B$	$\neg A$	$\bot$	$\top$
0	0	0	0	1	1	1	0	1
0	1	0	1	1	0	1	0	1
1	0	0	1	0	0	0	0	1
1	1	1	1	1	1	0	0	1

Les tables de la vérité (vraie)

Formules, équivalences, tautologies

On en déduit les valeurs des formules :

A	B	$A \to \neg B$	$B \wedge (A \to \neg B)$	$B \wedge \neg A$	$A \vee (A \to \neg B)$	$\cdots$
0	0	1	0	0	1	$\cdots$
0	1	1	1	1	1	$\cdots$
1	0	1	0	0	1	$\cdots$
1	1	0	0	0	1	$\cdots$

- Deux formules sont *équivalentes* si elles ont les mêmes valeurs.
- Une *tautologie* est une formule vraie (pour toute dvv)

A	B	$A \wedge B$	$A \vee B$	$A \to B$	$A \leftrightarrow B$	$\neg A$	$\bot$	$\top$
0	0	0	0	1	1	1	0	1
0	1	0	1	1	0	1	0	1
1	0	0	1	0	0	0	0	1
1	1	1	1	1	1	0	0	1

Les tables de la vérité (vraie)

Quelques vérités *classiques* classiques

$$A \to A \qquad A \vee \neg A \qquad A \vee (A \to B) \qquad (A \to B) \vee (B \to A)$$

$$A \wedge B \leftrightarrow B \wedge A \quad A \wedge (B \wedge C) \leftrightarrow (A \wedge B) \wedge C \quad A \wedge (B \vee C) \leftrightarrow (A \wedge B) \vee (A \wedge C)$$

$$(A \to B) \leftrightarrow (B \vee \neg A) \qquad A \leftrightarrow \neg\neg A \qquad \neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B)$$

Codages

On peut coder par exemple :

$$A \leftrightarrow B = (A \wedge B) \vee \neg(A \vee B) \quad A \rightarrow B = \neg A \vee B \quad A \vee B = \neg(\neg A \wedge \neg B)$$

ou encore

$$A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A) \quad A \wedge B = \neg(A \rightarrow \neg B) \quad \neg A = A \rightarrow \bot$$

Jeux complets de connecteurs

Par exemple $\{\wedge, \vee, \neg\}$, $\{\wedge, \neg\}$, $\{\rightarrow, \neg\}$, $\{\rightarrow, \bot\}$, ...

Les tables de la vérité (vraie)

C'est bien, mais pas top

- les valeurs de vérité formalisent l'intuition usuelle
- c'est simple (on se ramène à de l'algèbre sur $\{0, 1\}$)

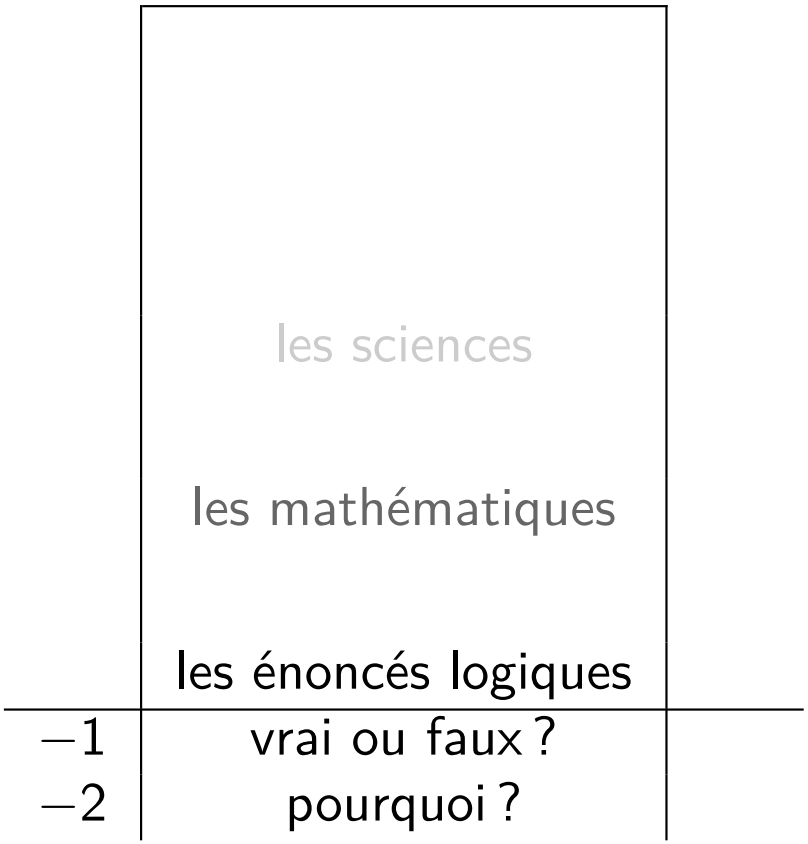
Les tables de la vérité (vraie)

C'est bien, mais pas top

- les valeurs de vérité formalisent l'intuition usuelle
- c'est simple (on se ramène à de l'algèbre sur $\{0, 1\}$)

- on n'apprend rien
- ça n'a rien à voir avec la démonstration mathématique

Les sous-sols de la logique



valeurs de vérité, modèles
démonstrations

Qu'est-ce qu'une démonstration ?

Exemples

Une suite d'affirmations justifiées

1. SAT est NP-complet [théorème]
2. SAT se réduit à 3-COLORATION [lemme]
3. 3-COLORATION est NP-dur [d'après 1 et 2]
4. 3-COLORATION est NP [lemme]
5. 3-COLORATION est NP-complet [d'après 3 et 4]

Une suite d'affirmations justifiées, avec un contexte

1. supposons $\mathbf{P} \neq \mathbf{NP}$	[hypothèse PNP]
1.1 supposons que p-COLORATION est **XP**	[hypothèse XP]
1.1.1 3-COLORATION a une complexité $O(n^{O(3)})$	[d'après XP]
1.1.2 3-COLORATION est **P**	[d'après 1.1.1]
1.1.3 3-COLORATION est **NP**-complet	[théorème]
1.1.4 $\mathbf{P} = \mathbf{NP}$	[d'après 1.1.2 et 1.1.3]
1.1.5 contradiction	[d'après PNP et 1.1.4]
1.2 p-COLORATION n'est pas **XP**	[d'après 1.1]
2. si $\mathbf{P} \neq \mathbf{NP}$ alors p-COLORATION n'est pas **XP**	[d'après 1]

Qu'est-ce qu'une démonstration ?

Arbres de déduction : idées

- l'important c'est la justification (déduction)
- un « état » de démonstration =
  les hypothèses sous lesquelles on travaille + la formule démontrée

On considère donc des arbres dont les nœuds sont les états de démonstration, justifiés par les sous-arbres correspondants.

Qu'est-ce qu'une démonstration ?

Arbres de déduction : idées

- l'important c'est la justification (déduction)
- un « état » de démonstration =
  les hypothèses sous lesquelles on travaille + la formule démontrée

On considère donc des arbres dont les nœuds sont les états de démonstration, justifiés par les sous-arbres correspondants.

Définition (Séquent)

$A_1, \ldots, A_n \vdash A$: on démontre A sous les hypothèses $A_1, \ldots, A_n$.

Qu'est-ce qu'une démonstration ?

Arbres de déduction : idées

- l'important c'est la justification (déduction)
- un « état » de démonstration =
  les hypothèses sous lesquelles on travaille + la formule démontrée

On considère donc des arbres dont les nœuds sont les états de démonstration, justifiés par les sous-arbres correspondants.

Définition (Séquent)

$A_1, \ldots, A_n \vdash A$: on démontre A sous les hypothèses $A_1, \ldots, A_n$.

Définition (Déduction)

$$\frac{\Gamma_1 \vdash A_1 \quad \cdots \quad \Gamma_n \vdash A_n}{\Delta \vdash B}$$

- l'important c'est la justification (déduction)
- un « état » de démonstration =
  les hypothèses sous lesquelles on travaille + la formule démontrée

On considère donc des arbres dont les nœuds sont les états de démonstration, justifiés par les sous-arbres correspondants.

Définition (Séquent)

$A_1, \ldots, A_n \vdash A$: on démontre A sous les hypothèses $A_1, \ldots, A_n$.

Définition (Déduction)

$$\frac{\Gamma_1 \vdash A_1 \quad \cdots \quad \Gamma_n \vdash A_n}{\Delta \vdash B}$$

($\Downarrow$) si on montre $\Gamma_1 \vdash A_1, \ldots, \Gamma_n \vdash A_n$ alors on peut déduire $\Delta \vdash B$

- l'important c'est la justification (déduction)
- un « état » de démonstration =
  les hypothèses sous lesquelles on travaille + la formule démontrée

On considère donc des arbres dont les nœuds sont les états de démonstration, justifiés par les sous-arbres correspondants.

Définition (Séquent)

$A_1, \ldots, A_n \vdash A$: on démontre A sous les hypothèses $A_1, \ldots, A_n$.

Définition (Déduction)

$$\frac{\Gamma_1 \vdash A_1 \quad \cdots \quad \Gamma_n \vdash A_n}{\Delta \vdash B}$$

($\Uparrow$) pour montrer $\Delta \vdash B$ il suffit de montrer (séparément) $\Gamma_1 \vdash A_1, \ldots, \Gamma_n \vdash A_n$

Qu'est-ce qu'une démonstration ?

Arbres de déduction : un exemple

$$\frac{}{\vdash B \wedge (A \to \neg B) \to \neg A}$$

Qu'est-ce qu'une démonstration ?

Arbres de déduction : un exemple

$$\dfrac{\dfrac{}{B \wedge (A \to \neg B) \vdash \neg A}}{\vdash B \wedge (A \to \neg B) \to \neg A}$$

Qu'est-ce qu'une démonstration ?

Arbres de déduction : un exemple

$$\dfrac{\dfrac{\dfrac{}{B \wedge (A \to \neg B), A \vdash \bot}}{B \wedge (A \to \neg B) \vdash \neg A}}{\vdash B \wedge (A \to \neg B) \to \neg A}$$

Qu'est-ce qu'une démonstration ?

Arbres de déduction : un exemple

$$\dfrac{\dfrac{\dfrac{}{B, A \to \neg B, A \vdash \bot}}{B \wedge (A \to \neg B), A \vdash \bot}}{B \wedge (A \to \neg B) \vdash \neg A}$$
$$\vdash B \wedge (A \to \neg B) \to \neg A$$

Qu'est-ce qu'une démonstration ?

Arbres de déduction : un exemple

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B, A \to \neg B, A \vdash \neg B} \quad \dfrac{}{B, A \to \neg B, A \vdash B}}{B, A \to \neg B, A \vdash \bot}}{B \wedge (A \to \neg B), A \vdash \bot}}{B \wedge (A \to \neg B) \vdash \neg A}}{\vdash B \wedge (A \to \neg B) \to \neg A}$$

Qu'est-ce qu'une démonstration ?

Arbres de déduction : un exemple

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B, A \to \neg B, A \vdash \neg B} \quad \dfrac{}{B, A \to \neg B, A \vdash B}}{B, A \to \neg B, A \vdash \bot}}{B \wedge (A \to \neg B), A \vdash \bot}}{B \wedge (A \to \neg B) \vdash \neg A}$$
$$\vdash B \wedge (A \to \neg B) \to \neg A$$

- Quelles déductions sont valides ?

Qu'est-ce qu'une démonstration ?

Arbres de déduction : un exemple

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B, A \to \neg B, A \vdash \neg B} \quad \dfrac{}{B, A \to \neg B, A \vdash B}}{B, A \to \neg B, A \vdash \bot}}{B \wedge (A \to \neg B), A \vdash \bot}}{B \wedge (A \to \neg B) \vdash \neg A}}{\vdash B \wedge (A \to \neg B) \to \neg A}$$

- Quelles déductions sont valides ?
- Il faut des règles.

Exemples

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad \frac{\Gamma, A, B \vdash C}{\Gamma A \wedge B \vdash C} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B} \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B} \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \vdash A \quad A \vdash B}{\Gamma \vdash B} \qquad \frac{}{\vdash A \to A}$$

Exemples

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad \frac{\Gamma, A, B \vdash C}{\Gamma A \wedge B \vdash C} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B} \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B} \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \vdash A \quad A \vdash B}{\Gamma \vdash B} \qquad \frac{}{\vdash A \to A}$$

- il faut qu'elles soient correctes (pour la vérité booléenne)
- on a intérêt à en prendre assez peu

(on ne considère que $\to$ et $\bot$: ça suffit du point de vue booléen)

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \langle\to\rangle \qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}\ [\to]$$

$$\frac{}{\Gamma, A \vdash A} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash A}$$

Partis pris

- on se concentre sur les conclusions (déductions « naturelles »)
- on gère les contextes de manière *additive*

Déduction naturelle

Exemple de démonstration 1

$$\frac{}{\vdash (A \to B) \to (B \to C) \to A \to C}\ \langle \to \rangle$$

Déduction naturelle

Exemple de démonstration 1

$$\dfrac{\dfrac{}{A \to B \vdash (B \to C) \to A \to C}\ \langle\to\rangle}{\vdash (A \to B) \to (B \to C) \to A \to C}\ \langle\to\rangle$$

Déduction naturelle

Exemple de démonstration 1

$$\dfrac{\dfrac{\dfrac{}{A \to B, B \to C \vdash A \to C}\ \langle\to\rangle}{A \to B \vdash (B \to C) \to A \to C}\ \langle\to\rangle}{\vdash (A \to B) \to (B \to C) \to A \to C}\ \langle\to\rangle$$

Déduction naturelle

Exemple de démonstration 1

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash C}\,[\to]}{A \to B, B \to C \vdash A \to C}\,\langle\to\rangle}{A \to B \vdash (B \to C) \to A \to C}\,\langle\to\rangle}{\vdash (A \to B) \to (B \to C) \to A \to C}\,\langle\to\rangle$$

avec $\Gamma = A \to B, B \to C, A$.

Déduction naturelle

Exemple de démonstration 1

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash B}\,[\to] \quad \dfrac{}{\Gamma \vdash B \to C}\,(ax)}{\Gamma \vdash C}\,[\to]}{A \to B, B \to C \vdash A \to C}\,\langle\to\rangle}{A \to B \vdash (B \to C) \to A \to C}\,\langle\to\rangle}{\vdash (A \to B) \to (B \to C) \to A \to C}\,\langle\to\rangle$$

avec $\Gamma = A \to B, B \to C, A$.

Déduction naturelle

Exemple de démonstration 1

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash A \to B}\,(ax) \quad \dfrac{}{\Gamma \vdash A}\,(ax)}{\Gamma \vdash B}\,[\to] \quad \dfrac{}{\Gamma \vdash B \to C}\,(ax)}{\Gamma \vdash C}\,[\to]}{A \to B, B \to C \vdash A \to C}\,\langle\to\rangle}{A \to B \vdash (B \to C) \to A \to C}\,\langle\to\rangle}{\vdash (A \to B) \to (B \to C) \to A \to C}\,\langle\to\rangle$$

avec $\Gamma = A \to B, B \to C, A$.

Déduction naturelle

Exemple de démonstration 2

$$\frac{}{\vdash (A \wedge \neg A) \to B}\ \langle \to \rangle$$

Déduction naturelle

Exemple de démonstration 2

$$\frac{}{\vdash \neg(A \to \neg\neg A) \to B}\ \langle\to\rangle$$

Déduction naturelle

Exemple de démonstration 2

$$\dfrac{\dfrac{}{\neg(A \to \neg\neg A) \vdash B}\ [\bot]}{\vdash \neg(A \to \neg\neg A) \to B}\ \langle\to\rangle$$

Déduction naturelle

Exemple de démonstration 2

$$\dfrac{\dfrac{\dfrac{}{\neg(A \to \neg\neg A) \vdash \bot}\,[\to]}{\neg(A \to \neg\neg A) \vdash B}\,[\bot]}{\vdash \neg(A \to \neg\neg A) \to B}\,\langle\to\rangle$$

Déduction naturelle

Exemple de démonstration 2

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(A \to \neg\neg A) \vdash A \to \neg\neg A}\,\langle\to\rangle \qquad \dfrac{}{\neg(A \to \neg\neg A) \vdash \neg(A \to \neg\neg A)}\,(ax)}{\neg(A \to \neg\neg A) \vdash \bot}\,[\to]}{\neg(A \to \neg\neg A) \vdash B}\,[\bot]}{\vdash \neg(A \to \neg\neg A) \to B}\,\langle\to\rangle$$

Déduction naturelle

Exemple de démonstration 2

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(A \to \neg\neg A), A \vdash \neg\neg A}\,\langle\to\rangle}{\neg(A \to \neg\neg A) \vdash A \to \neg\neg A}\,\langle\to\rangle \qquad \dfrac{}{\neg(A \to \neg\neg A) \vdash \neg(A \to \neg\neg A)}\,(ax)}{\dfrac{\neg(A \to \neg\neg A) \vdash \bot}{\neg(A \to \neg\neg A) \vdash B}\,[\bot]}\,[\to]}{\vdash \neg(A \to \neg\neg A) \to B}\,\langle\to\rangle$$

Déduction naturelle

Exemple de démonstration 2

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \bot}\,[\to]}{\neg(A \to \neg\neg A), A \vdash \neg\neg A}\,\langle\to\rangle}{\neg(A \to \neg\neg A) \vdash A \to \neg\neg A}\,\langle\to\rangle \qquad \dfrac{}{\neg(A \to \neg\neg A) \vdash \neg(A \to \neg\neg A)}\,(ax)}{\neg(A \to \neg\neg A) \vdash \bot}\,[\to]}{\neg(A \to \neg\neg A) \vdash B}\,[\bot]}{\vdash \neg(A \to \neg\neg A) \to B}\,\langle\to\rangle$$

avec $\Gamma = A \to B, B \to C, A$.

Déduction naturelle

Exemple de démonstration 2

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \neg A}\,(ax) \quad \dfrac{}{\Gamma \vdash A}\,(ax)}{\Gamma \vdash \bot}\,[\to]}{\neg(A \to \neg\neg A), A \vdash \neg\neg A}\,\langle\to\rangle}{\neg(A \to \neg\neg A) \vdash A \to \neg\neg A}\,\langle\to\rangle \quad \dfrac{}{\neg(A \to \neg\neg A) \vdash \neg(A \to \neg\neg A)}\,(ax)}{\neg(A \to \neg\neg A) \vdash \bot}\,[\to]}{\neg(A \to \neg\neg A) \vdash B}\,[\bot]}{\vdash \neg(A \to \neg\neg A) \to B}\,\langle\to\rangle$$

avec $\Gamma = A \to B, B \to C, A$.

Déduction naturelle

Exemple de démonstration 2

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \neg A}\,(ax) \quad \dfrac{}{\Gamma \vdash A}\,(ax)}{\Gamma \vdash \bot}\,[\to]}{\neg(A \to \neg\neg A), A \vdash \neg\neg A}\,\langle\to\rangle}{\neg(A \to \neg\neg A) \vdash A \to \neg\neg A}\,\langle\to\rangle \quad \dfrac{}{\neg(A \to \neg\neg A) \vdash \neg(A \to \neg\neg A)}\,(ax)}{\neg(A \to \neg\neg A) \vdash \bot}\,[\to]}{\neg(A \to \neg\neg A) \vdash B}\,[\bot]}{\vdash \neg(A \to \neg\neg A) \to B}\,\langle\to\rangle$$

avec $\Gamma = A \to B, B \to C, A$.

- Hrumpfff !

Déduction naturelle

Exemple de démonstration 2

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \neg A}\,(ax) \quad \dfrac{}{\Gamma \vdash A}\,(ax)}{\Gamma \vdash \bot}\,[\to]}{\neg(A \to \neg\neg A), A \vdash \neg\neg A}\,\langle\to\rangle}{\neg(A \to \neg\neg A) \vdash A \to \neg\neg A}\,\langle\to\rangle \quad \dfrac{}{\neg(A \to \neg\neg A) \vdash \neg(A \to \neg\neg A)}\,(ax)}{\neg(A \to \neg\neg A) \vdash \bot}\,[\to]}{\neg(A \to \neg\neg A) \vdash B}\,[\bot]}{\vdash \neg(A \to \neg\neg A) \to B}\,\langle\to\rangle$$

avec $\Gamma = A \to B, B \to C, A$.

- Hrumpfff !
- Et $\neg\neg A \to A$?

Déduction naturelle

Raisonner sur la vérité

$$\dfrac{\dfrac{?}{\neg\neg A \vdash A}}{\vdash \neg\neg A \to A}\ \langle\to\rangle$$

Déduction naturelle

Raisonner sur la vérité

$$\dfrac{\dfrac{?}{\neg\neg A \vdash \bot}}{\neg\neg A \vdash A}\,[\bot]}{\vdash \neg\neg A \to A}\,\langle\to\rangle$$

Déduction naturelle

Raisonner sur la vérité

$$\dfrac{\dfrac{\neg\neg A \vdash ? \quad \neg\neg A \vdash ? \to A}{\neg\neg A \vdash A}\,[\to]}{\vdash \neg\neg A \to A}\,\langle\to\rangle$$

$$\dfrac{\begin{array}{c}?\\ \neg\neg A \vdash A\end{array}}{\vdash \neg\neg A \to A}\ \langle\to\rangle$$

- On ne voit pas trop (sauf si par exemple $A = \neg B$)

$$\dfrac{\begin{array}{c}?\\ \neg\neg A \vdash A\end{array}}{\vdash \neg\neg A \to A}\ \langle\to\rangle$$

- On ne voit pas trop (sauf si par exemple $A = \neg B$)
- On a oublié quelque chose pour raisonner sur la vérité

$$\dfrac{\begin{array}{c}?\\ \neg\neg A \vdash A\end{array}}{\vdash \neg\neg A \to A}\ \langle\to\rangle$$

- On ne voit pas trop (sauf si par exemple $A = \neg B$)
- On a oublié quelque chose pour raisonner sur la vérité
- Par exemple, le raisonnement par l'absurde :

$$\dfrac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\ (\bot)$$

$$\frac{\begin{array}{c}?\\ \neg\neg A \vdash A\end{array}}{\vdash \neg\neg A \to A}\ \langle\to\rangle$$

- On ne voit pas trop (sauf si par exemple $A = \neg B$)
- On a oublié quelque chose pour raisonner sur la vérité
- Par exemple, le raisonnement par l'absurde :

$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\ (\bot)$$

Attention, c'est bien différent de $\dfrac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \langle\to\rangle$

Déduction naturelle

Raisonner sur la vérité

$$\dfrac{\dfrac{\dfrac{}{\neg\neg A, \neg A \vdash \neg A}\,(ax) \quad \dfrac{}{\neg\neg A, \neg A \vdash \neg\neg A}\,(ax)}{\neg\neg A, \neg A \vdash \bot}\,[\to]}{\dfrac{\neg\neg A \vdash A}{\vdash \neg\neg A \to A}\,\langle\to\rangle}\,(\bot)$$

- On ne voit pas trop (sauf si par exemple $A = \neg B$)
- On a oublié quelque chose pour raisonner sur la vérité
- Par exemple, le raisonnement par l'absurde :

$$\dfrac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\,(\bot)$$

Attention, c'est bien différent de $\dfrac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\,\langle\to\rangle$

$$\dfrac{\dfrac{\dfrac{}{\neg\neg A, \neg A \vdash \neg A}\,(ax) \quad \dfrac{}{\neg\neg A, \neg A \vdash \neg\neg A}\,(ax)}{\neg\neg A, \neg A \vdash \bot}\,[\to]}{\neg\neg A \vdash A}\,(\bot)}{\vdash \neg\neg A \to A}\,\langle\to\rangle$$

- On ne voit pas trop (sauf si par exemple $A = \neg B$)
- On a oublié quelque chose pour raisonner sur la vérité
- Par exemple, le raisonnement par l'absurde :

$$\dfrac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\,(\bot)$$

  Attention, c'est bien différent de $\dfrac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\,\langle\to\rangle$

- Également nécessaire pour des formules sans $\bot$: $((A \to B) \to A) \to A$.

Exercice page 104 : prouver cette formule en présence de $\bot$

Déduction naturelle

Et les gagnants sont. . .

$$\frac{}{\Gamma, A \vdash A}\ (ax) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \langle\to\rangle \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ [\to] \qquad \frac{\Gamma, A \to \bot \vdash \bot}{\Gamma \vdash A}\ (\bot)$$

Déduction naturelle

Et les gagnants sont. . .

$$\frac{}{\Gamma, A \vdash A}\ (ax) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \langle\to\rangle \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ [\to] \qquad \frac{\Gamma, A \to \bot \vdash \bot}{\Gamma \vdash A}\ (\bot)$$

A-t-on encore oublié quelque chose ?

Déduction naturelle

Et les gagnants sont. . .

$$\frac{}{\Gamma, A \vdash A}\ (ax) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \langle\to\rangle \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ [\to] \qquad \frac{\Gamma, A \to \bot \vdash \bot}{\Gamma \vdash A}\ (\bot)$$

A-t-on encore oublié quelque chose ? Non :

Théorème (Complétude)

Si A est une tautologie, alors il existe une preuve de $\vdash A$ en déduction naturelle classique.

Déduction naturelle

Et les gagnants sont. . .

$$\frac{}{\Gamma, A \vdash A}\ (ax) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \langle\to\rangle \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ [\to] \qquad \frac{\Gamma, A \to \bot \vdash \bot}{\Gamma \vdash A}\ (\bot)$$

A-t-on encore oublié quelque chose ? Non :

Théorème (Complétude)

Si A est une tautologie, alors il existe une preuve de $\vdash A$ en déduction naturelle classique.

- Facile. . .

Voir Théorème 3.1.3, page 104

.

Déduction naturelle

Et les gagnants sont...

$$\frac{}{\Gamma, A \vdash A}\ (ax) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \langle\to\rangle \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ [\to] \qquad \frac{\Gamma, A \to \bot \vdash \bot}{\Gamma \vdash A}\ (\bot)$$

A-t-on encore oublié quelque chose ? Non :

Théorème (Complétude)

Si A est une tautologie, alors il existe une preuve de $\vdash A$ en déduction naturelle classique.

- Facile...

Voir Théorème 3.1.3, page 104

.

- On aurait pu faire pareil avec d'autres jeux complets de connecteurs.

Intuitionnisme

Sans raisonnement par l'absurde

$$\frac{}{\Gamma, A \vdash A}\ (ax) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \langle\to\rangle \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ [\to]$$

Intuitionnisme

Sans raisonnement par l'absurde

$$\frac{}{\Gamma, A \vdash A}\ (ax) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \langle\to\rangle \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ [\to]$$

- ça reste correct
- mais c'est incomplet

Intuitionnisme

Sans raisonnement par l'absurde

$$\frac{}{\Gamma, A \vdash A}\ (ax) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \langle\to\rangle \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ [\to]$$

- ça reste correct
- mais c'est incomplet

Pourquoi s'embêter avec ça ?

- en 1900 : parce que
- en 2000 : parce qu'on peut dire quelque chose d'intéressant dans ce cas

Une explication « procédurale » des preuves

- une preuve de $A \wedge B$ est donnée par une preuve de A et une preuve de B
- une preuve de $A \vee B$ est donnée par une preuve de A (et le bit « gauche ») ou une preuve de B (et le bit « droit »)
- une preuve de $A \to B$ est une *fonction* qui transforme une preuve de A en une preuve de B
- il n'y a pas de preuve de $\bot$

Une explication « procédurale » des preuves

- une preuve de $A \wedge B$ est donnée par une preuve de A et une preuve de B
- une preuve de $A \vee B$ est donnée par une preuve de A (et le bit « gauche ») ou une preuve de B (et le bit « droit »)
- une preuve de $A \to B$ est une *fonction* qui transforme une preuve de A en une preuve de B
- il n'y a pas de preuve de $\bot$

Si on la prend au sens littéral (un connecteur = une opération ensembliste) cette idée revient au modèle booléen dès qu'il y a une négation :

- si A a une preuve $\neg A$ n'en a pas ;
- si A n'a pas de preuve $\neg A$ en a exactement une.

Quand même. . .

- une preuve de $A \vdash B$ est une « fonction » qui transforme une preuve de A en une preuve de B

$$\dfrac{A \overset{f}{\vdash} B \qquad \dfrac{B \overset{g}{\vdash} C}{\vdash B \to C}\ \langle\to\rangle}{A \vdash C}\ [\to]$$

Quand même. . .

- une preuve de $A \vdash B$ est une « fonction » qui transforme une preuve de A en une preuve de B

$$\dfrac{x : A \overset{f}{\vdash} f(x) : B \qquad \dfrac{y : B \overset{g}{\vdash} g(y) : C}{\vdash (y \mapsto g(y)) : B \to C}\ \langle\to\rangle}{x : A \vdash g(f(x)) : C}\ [\to]$$

Quand même. . .

- une preuve de $A \vdash B$ est une « fonction » qui transforme une preuve de A en une preuve de B

$$\dfrac{x : A \overset{f}{\vdash} f(x) : B \qquad \dfrac{y : B \overset{g}{\vdash} g(y) : C}{\vdash (y \mapsto g(y)) : B \to C}\ \langle\to\rangle}{x : A \vdash g(f(x)) : C}\ [\to]$$

- ça *calcule* la composition des fonctions !

Quand même. . .

- une preuve de $A \vdash B$ est une « fonction » qui transforme une preuve de A en une preuve de B

$$\dfrac{x : A \overset{f}{\vdash} f(x) : B \qquad \dfrac{y : B \overset{g}{\vdash} g(y) : C}{\vdash (y \mapsto g(y)) : B \to C}\ \langle\to\rangle}{x : A \vdash g(f(x)) : C}\ [\to]$$

- ça *calcule* la composition des fonctions !
- mais de quoi parle-t-on ?

Quand même. . .

- une preuve de $A \vdash B$ est une « fonction » qui transforme une preuve de A en une preuve de B

$$\dfrac{x : A \overset{f}{\vdash} f(x) : B \qquad \dfrac{y : B \overset{g}{\vdash} g(y) : C}{\vdash (y \mapsto g(y)) : B \to C}\ \langle\to\rangle}{x : A \vdash g(f(x)) : C}\ [\to]$$

- ça *calcule* la composition des fonctions !
- mais de quoi parle-t-on ?
- le suspense est insoutenable. . .

Deuxième partie

Le λ-calcul et la correspondance de Curry-Howard

La démarche axiomatique

Qu'est-ce qu'une fonction ?

La démarche axiomatique

Qu'est-ce qu'une fonction ?

La réponse à cette question a varié au cours du temps.

avant le XVIIème siècle une quoi ?

Leibniz (XVIIème) une courbe

Euler (XVIIIème) une formule, un calcul

Dirichlet (XIXème) une relation

Mêmes questions pour les réels, les entiers, les ensembles...
les démonstrations, les calculs...

En fin de compte, pour fonder le sens des démonstrations sur la notion de fonction, on cherche une axiomatique pure des fonctions.

λ-calcul : le langage

Le langage :

$M, N :=$	$\lambda x.M$	fonction qui à x associe M
	$(M)N$	application de la fonction M à l'argument N
	x	utilisation d'une variable

Par exemple :

- la fonction identité : $\lambda x.x$ $\qquad x \mapsto x$
- composition de fonctions : $\lambda x.(f)(g)x$ $\qquad x \mapsto f(g(x))$
- composition avec f : $\lambda g.\lambda x.(g)(f)x$ $\qquad g \mapsto (x \mapsto g(f(x)))$

Variables, paramètres et indéterminées

Il est important de bien distinguer la *variable* d'une fonction, ses éventuels *paramètres*, et d'autre part la notion d'*indéterminée*.

Une analogie :

- Dans $\mathbb{R}[X, Y]$, les polynômes $X^2 + 5$ et $Y^2 + 5$ sont distincts.
- Les fonctions $x \mapsto x^2 + 5$ et $y \mapsto y^2 + 5$ sont identiques.

Habituellement, en mathématiques, il y a beaucoup d'implicite dans les notations pour les fonctions :

La dérivée de la fonction $2x^2 + 3x + 1$ *est* $4x + 3$.

En λ-calcul, il n'y a **jamais** d'implicite de ce genre.

Substitution et évaluation

Considérons deux termes :

$$M := \lambda g.\lambda x.(g)(f)x$$
$$N := \lambda y.((h)y)y$$

$(M)N$ est l'application de la fonction M à l'argument N donc :

$$\begin{aligned}(M)N &= \big(\lambda g.\lambda x.(g)(f)x\big)\,N \\ &= \lambda x.(N)(f)x\end{aligned}$$

Substitution et évaluation

Considérons deux termes :

$$M := \lambda g.\lambda x.(g)(f)x$$
$$N := \lambda y.((h)y)y$$

$(M)N$ est l'application de la fonction M à l'argument N donc :

$$\begin{aligned}(M)N &= (\lambda g.\lambda x.(g)(f)x)\,N \\ &= \lambda x.(N)(f)x \\ \ldots \text{ et on peut continuer } &\ldots \\ &= \lambda x.(\lambda y.((h)y)y)(f)x \\ &= \lambda x.((h)(f)x)(f)x\end{aligned}$$

Deux axiomes

La notion centrale est la substitution *sans capture* :

- dans un terme $\lambda x.M$, la variable x est muette

$$\lambda g.\lambda x.(g)(f)x = \lambda h.\lambda z.(h)(f)z$$

- on note $M[N/x]$ le remplacement de chaque occurence de x dans M par N, en renommant au besoin les variables muettes :

$$\big(\lambda x.(g)(f)x\big)\big[(f)x \;/\; g\big] = \lambda y.((f)x)(f)y$$

D'où les deux axiomes fondamentaux du λ-calcul :

- α-équivalence : les variables sont muettes (implicite en général)

$$\lambda x.M = \lambda y.M[y/x] \qquad \text{si } y \text{ n'apparaît pas dans } M$$

- β-équivalence : l'évaluation des fonctions

$$(\lambda x.M)N =_\beta M[N/x]$$

Programmer en λ-calcul

- Un booléen est une façon de choisir entre deux possibilités :

$$\text{true} := \lambda x.\lambda y.x$$
$$\text{false} := \lambda x.\lambda y.y$$
$$\text{if } A \text{ then } B \text{ else } C := ((A)B)C$$

Programmer en λ-calcul

- Un booléen est une façon de choisir entre deux possibilités :

$$\begin{aligned} \text{true} &:= \lambda x.\lambda y.x \\ \text{false} &:= \lambda x.\lambda y.y \\ \text{if } A \text{ then } B \text{ else } C &:= ((A)B)C \end{aligned}$$

- Un entier n est un opérateur qui compose n fois une fonction :

$$\underline{n} := \lambda f.\lambda x.\underbrace{(f)(f)\cdots(f)}_{n \text{ fois}}x$$

$$\begin{aligned} \text{succ} &:= \lambda n.\lambda f.\lambda x.(f)(n)fx \\ \text{add} &:= \lambda m.\lambda n.\lambda f.\lambda x.(m)f(n)fx \quad \text{ou } \lambda m.\lambda n.((m)\text{succ})n \\ \text{mul} &:= \lambda m.\lambda n.\lambda f.\lambda x.((m)(n)f)x \quad \text{ou } \lambda m.\lambda n.((m)(\text{add})n)\underline{0} \\ \text{isZero} &:= \lambda n.n(\lambda b.\text{false})\text{true} \end{aligned}$$

Exercice page 108 : vérifier que ces définitions fonctionnent

Représentation des données en λ-calcul

On peut généraliser cette représentation à n'importe quel type de données inductif :

```
type 'a truc =
  | Cas1 of 'a * 'a truc
  | Cas2 of int * 'a
  | Cas3
```

Une donnée de ce type est un opérateur qui attend une fonction pour chaque cas et « construit » la donnée avec ces fonctions :

`Cas1 a t`	$\lambda c_1.\lambda c_2.\lambda c_3.((c_1)a)(((t)c_1)c_2)c_3$
`Cas2 n a`	$\lambda c_1.\lambda c_2.\lambda c_3.((c_2)n)a$
`Cas3`	$\lambda c_1.\lambda c_2.\lambda c_3.c_3$

Remarque : les fonctions de plusieurs variables sont *curryfiées*.

Représentation des données en λ-calcul

Exemple : calcul du prédécesseur

Un entier est représenté par un itérateur.
À partir du terme pour n, comment itérer $n-1$ fois ?

- Idée : si on itère n fois $\langle x, y\rangle \mapsto \langle y, y+1\rangle$ en partant de $\langle 0, 0\rangle$, on obtient $\langle n-1, n\rangle$.

Représentation des données en λ-calcul

Exemple : calcul du prédécesseur

Un entier est représenté par un itérateur.
À partir du terme pour n, comment itérer $n-1$ fois ?

- Idée : si on itère n fois $\langle x, y\rangle \mapsto \langle y, y+1\rangle$ en partant de $\langle 0, 0\rangle$, on obtient $\langle n-1, n\rangle$.
- Représentation des couples :

$$\langle A, B\rangle := \lambda f.((f)A)B$$

$$\pi_1 := \lambda p.(p)\lambda x.\lambda y.x \qquad \langle A, B\rangle \mapsto A$$

$$\pi_2 := \lambda p.(p)\lambda x.\lambda y.y \qquad \langle A, B\rangle \mapsto B$$

Représentation des données en λ-calcul

Exemple : calcul du prédécesseur

Un entier est représenté par un itérateur.
À partir du terme pour n, comment itérer $n-1$ fois ?

- Idée : si on itère n fois $\langle x, y\rangle \mapsto \langle y, y+1\rangle$ en partant de $\langle 0, 0\rangle$, on obtient $\langle n-1, n\rangle$.
- Représentation des couples :

$$\begin{aligned} \langle A, B\rangle &:= \lambda f.((f)A)B \\ \pi_1 &:= \lambda p.(p)\lambda x.\lambda y.x \qquad & \langle A, B\rangle \mapsto A \\ \pi_2 &:= \lambda p.(p)\lambda x.\lambda y.y \qquad & \langle A, B\rangle \mapsto B \end{aligned}$$

- On obtient le terme :

$$\text{pred} := \lambda n.(\pi_1)((n)\lambda p.\langle(\pi_2)p, (S)(\pi_2)n\rangle)\langle \underline{0}, \underline{0}\rangle$$

Quid de la cohérence ?

On oriente la β-équivalence :

$$(\lambda x.M)N \rightsquigarrow M[N/x] \qquad \beta\text{-réduction}$$

Théorème (Church-Rosser)

Pour tous λ-termes M et N tels que $M =_\beta N$, il existe un λ-terme R tel que $M \rightsquigarrow \cdots \rightsquigarrow R$ et $N \rightsquigarrow \cdots \rightsquigarrow R$.

C'est-à-dire qu'il suffit de réduire pour établir une équivalence.

Corollaire

Si M et N sont deux λ-termes irréductibles, alors $M =_\beta N$ si et seulement si $M = N$.

Les termes irréductibles sont donc appelés formes normales.

Normalisation

Les formes normales sont donc des représentants canoniques de classes de β-équivalence, il suffit de réduire autant qu'on peut pour **calculer** la forme normale.

Normalisation

Les formes normales sont donc des représentants canoniques de classes de β-équivalence, il suffit de réduire autant qu'on peut pour **calculer** la forme normale.

Posons $\delta := \lambda x.(x)x$ et $\Omega := (\delta)\delta$, alors $\Omega \rightsquigarrow \Omega$.
On n'atteint pas de forme normale !

Normalisation

Les formes normales sont donc des représentants canoniques de classes de β-équivalence, il suffit de réduire autant qu'on peut pour **calculer** la forme normale.

Posons $\delta := \lambda x.(x)x$ et $\Omega := (\delta)\delta$, alors $\Omega \rightsquigarrow \Omega$.
On n'atteint pas de forme normale !

Théorème

Le problème de l'existence de forme normale pour un terme donné est indécidable.

Expressivité

Théorème

Une fonction partielle $f : \mathbb{N} \to \mathbb{N}$ est récursive si et seulement si il existe un λ-terme F tel que

- *si $f(n)$ est défini alors $(F)\underline{n} =_\beta \underline{f(n)}$,*
- *si $f(n)$ n'est pas défini alors $(F)\underline{n}$ n'a pas de forme normale.*

Le typage

Pour des termes qui ont un sens, on utilise des types :

$$\begin{array}{rll} A, B := & \alpha & \text{type de base (donné)} \\ & A \to B & \text{fonction de } A \text{ dans } B \end{array}$$

Moralement, un type est un ensemble de termes, on l'interprète comme un ensemble de « valeurs » possibles.

Un jugement de typage est de la forme

$$x_1 : A_1, \ldots, x_n : A_n \vdash M : B$$

Si chaque x_i prend une valeur dans le type A_i correspondant, alors M prendra une valeur dans le type B.

Règles de typage

Le λ-calcul simplement typé

- construction d'une fonction

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}$$

- application d'une fonction

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M)N : B}$$

- utilisation d'une variable

$$\frac{}{\Gamma, x : A \vdash x : A}$$

Règles de typage

Le λ-calcul simplement typé

- construction d'une fonction

$$\frac{\Gamma, \quad A \vdash \quad B}{\Gamma \vdash \quad A \to B}$$

- application d'une fonction

$$\frac{\Gamma \vdash \quad A \to B \quad \Gamma \vdash \quad A}{\Gamma \vdash \quad B}$$

- utilisation d'une variable

$$\frac{}{\Gamma, \quad A \vdash \quad A}$$

Typage et réduction

Théorème (*subject reduction*)

Soient deux λ-termes M et N. Si M est typable, par un jugement $\Gamma \vdash M : A$ et si $M \rightsquigarrow N$ alors N est typable par $\Gamma \vdash N : A$.

Typage et réduction

Théorème (*subject reduction*)

Soient deux λ-termes M et N. Si M est typable, par un jugement $\Gamma \vdash M : A$ et si $M \rightsquigarrow N$ alors N est typable par $\Gamma \vdash N : A$.

Observons ce qui se passe là où on réduit :

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x : A \vdash x : A}}{\vdots}\ \dfrac{}{\Gamma, x : A \vdash M : B}}{\Gamma \vdash \lambda x.M : A \to B} \quad \Gamma \vdash N : A}{\Gamma \vdash (\lambda x.M)N : B} \quad \rightsquigarrow \quad \dfrac{\Gamma \vdash N : A}{\vdots}\ \Gamma \vdash M[N/x] : B$$

Il y a un lemme de substitution qui justifie que la démonstration de droite est correcte.

Typage et normalisation

Théorème (normalisation forte)

Un terme typable n'a pas de suite infinie de β-réductions.

Typage et normalisation

Théorème (normalisation forte)

Un terme typable n'a pas de suite infinie de β-réductions.

Idée de démonstration :

- interpréter sémantiquement les types :

$$|A \to B| := \{M \mid \forall N \in |A|, (M)N \in |B|\},$$

- démontrer que l'adéquation par induction sur les termes :

$$\text{si } (x_i : A_i)_i \vdash M : B \quad \text{alors} \quad \forall N_i \in |A_i|, M[N_i/x_i] \in |B|,$$

- si l'interprétation des types de base ne contient que des termes sans réduction infinie, la propriété se propage à tous les types.

Voir page 111 : une autre démonstration (Théorème 3.1.13)

Remarque en passant

Le typage traduit formellement l'intuition selon laquelle les λ-termes sont des fonctions, cette intuition garantit la cohérence du système. C'est la base de la *sémantique dénotationnelle* :

- on choisit un ensemble $[\![\alpha]\!]$ pour chaque type de base α,
- on définit l'ensemble qui interprète les types composés : $[\![A \to B]\!] := [\![B]\!]^{[\![A]\!]}$,
- on interprète un terme de type A par un point de $[\![A]\!]$, de la façon intuitive.

Alors les termes sont interprétés par de vraies fonctions et l'interprétation est invariante par β-réduction.

Typage et réduction, épisode 2

On a déjà vu que les règles de typage sont les mêmes que les règles de la déduction naturelle, décorées avec des λ-termes.

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x : A \vdash x : A}}{\vdots}}{\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \quad \Gamma \vdash N : A}{\Gamma \vdash (\lambda x.M)N : B} \quad \rightsquigarrow \quad \dfrac{\Gamma \vdash N : A}{\vdots}{\Gamma \vdash M[N/x] : B}$$

Que signifie cette réduction du point de vue de la déduction naturelle ?

Typage et réduction, épisode 2

On a déjà vu que les règles de typage sont les mêmes que les règles de la déduction naturelle, décorées avec des λ-termes.

$$\dfrac{\dfrac{\overline{\Gamma, A \vdash A}}{\vdots}\ \Gamma, A \vdash B}{\Gamma \vdash A \to B} \quad \Gamma \vdash A \qquad \rightsquigarrow \qquad \begin{array}{c} \Gamma \vdash A \\ \vdots \\ \Gamma \vdash B \end{array}$$
$$\Gamma \vdash B$$

C'est une réécriture des démonstrations : l'élimination des coupures.

Élimination des coupures

Définition

En déduction naturelle, un *coupure* est l'enchaînement d'une règle d'introduction et d'une règle d'élimination de la même formule.

Théorème

Un séquent $\Gamma \vdash A$ est démontrable en déduction naturelle si et seulement si il est démontrable avec une démonstration sans coupure.

Interpréter la démonstration comme un λ-terme typé, le normaliser, le typage de la forme normale est une démonstration sans coupure.

Élimination des coupures

Définition

En déduction naturelle, un *coupure* est l'enchaînement d'une règle d'introduction et d'une règle d'élimination de la même formule.

Théorème

Un séquent $\Gamma \vdash A$ est démontrable en déduction naturelle si et seulement si il est démontrable avec une démonstration sans coupure.

Interpréter la démonstration comme un λ-terme typé, le normaliser, le typage de la forme normale est une démonstration sans coupure.

Pour traiter le type $\bot$, on ajoute :

$$\frac{\Gamma \vdash M : \bot}{\Gamma \vdash \mathcal{A}[M] : A} \qquad (\mathcal{A}[M])N \rightsquigarrow \mathcal{A}[M] \qquad \text{une fonction } \textit{abort}$$

Élimination des coupures et cohérence logique

Théorème

Le séquent $\vdash \neg\neg\alpha \to \alpha$ *n'est pas démontrable en logique minimale.*

Élimination des coupures et cohérence logique

Théorème

Le séquent $\vdash \neg\neg\alpha \to \alpha$ *n'est pas démontrable en logique minimale.*

Cherchons une démonstration *sans coupure* :

$$\frac{?}{\vdash M : \neg\neg\alpha \to \alpha}$$

Élimination des coupures et cohérence logique

Théorème

Le séquent $\vdash \neg\neg\alpha \to \alpha$ *n'est pas démontrable en logique minimale.*

Cherchons une démonstration *sans coupure* : avec $\Gamma = x : \neg\neg\alpha$

$$\dfrac{\dfrac{?}{\Gamma \vdash M : \alpha}}{\vdash \lambda x.M : \neg\neg\alpha \to \alpha}$$

Élimination des coupures et cohérence logique

Théorème

Le séquent $\vdash \neg\neg\alpha \to \alpha$ *n'est pas démontrable en logique minimale.*

Cherchons une démonstration *sans coupure* : avec $\Gamma = x : \neg\neg\alpha$

$$\dfrac{\dfrac{\dfrac{?}{\Gamma \vdash M : A_1 \to \alpha} \quad \dfrac{\vdots}{\Gamma \vdash N_1 : A_1}}{\Gamma \vdash (M)N_1 : \alpha}}{\vdash \lambda x.(M)N_1 : \neg\neg\alpha \to \alpha}$$

Élimination des coupures et cohérence logique

Théorème

Le séquent $\vdash \neg\neg\alpha \to \alpha$ *n'est pas démontrable en logique minimale.*

Cherchons une démonstration *sans coupure* : avec $\Gamma = x : \neg\neg\alpha$

$$\dfrac{\dfrac{\dfrac{?}{\Gamma \vdash M : A_2 \to A_1 \to \alpha} \quad \dfrac{\vdots}{\Gamma \vdash N_2 : A_2}}{\Gamma \vdash (M)N_2 : A_1 \to \alpha} \quad \dfrac{\vdots}{\Gamma \vdash N_1 : A_1}}{\Gamma \vdash ((M)N_2)N_1 : \alpha}$$
$$\vdash \lambda x.((M)N_2)N_1 : \neg\neg\alpha \to \alpha$$

Élimination des coupures et cohérence logique

Théorème

Le séquent $\vdash \neg\neg\alpha \to \alpha$ *n'est pas démontrable en logique minimale.*

Cherchons une démonstration *sans coupure* : avec $\Gamma = x : \neg\neg\alpha$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{?}{\Gamma \vdash M : A_k \to \cdots A_2 \to A_1 \to \alpha}}{\vdots}}{\Gamma \vdash \cdots(M)N_k\cdots : A_2 \to A_1 \to \alpha}\quad\dfrac{\vdots}{\Gamma \vdash N_2 : A_2}}{\Gamma \vdash (\cdots(M)N_k\cdots)N_2 : A_1 \to \alpha}\quad\dfrac{\vdots}{\Gamma \vdash N_1 : A_1}}{\Gamma \vdash ((\cdots(M)N_k\cdots)N_2)N_1 : \alpha}}{\vdash \lambda x.((\cdots(M)N_k\cdots)N_2)N_1 : \neg\neg\alpha \to \alpha}$$

Élimination des coupures et cohérence logique

Théorème

Le séquent $\vdash \neg\neg\alpha \to \alpha$ *n'est pas démontrable en logique minimale.*

Cherchons une démonstration *sans coupure* : avec $\Gamma = x : \neg\neg\alpha$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{?}{\Gamma \vdash M : A_k \to \cdots A_2 \to A_1 \to \alpha}}{\vdots}}{\Gamma \vdash \cdots(M)N_k\cdots : A_2 \to A_1 \to \alpha} \quad \dfrac{\vdots}{\Gamma \vdash N_2 : A_2}}{\Gamma \vdash (\cdots(M)N_k\cdots)N_2 : A_1 \to \alpha} \quad \dfrac{\vdots}{\Gamma \vdash N_1 : A_1}}{\Gamma \vdash ((\cdots(M)N_k\cdots)N_2)N_1 : \alpha}}{\vdash \lambda x.((\cdots(M)N_k\cdots)N_2)N_1 : \neg\neg\alpha \to \alpha}$$

Impossible !

Exercice page 111 : faire de même avec la loi de Peirce

La correspondance

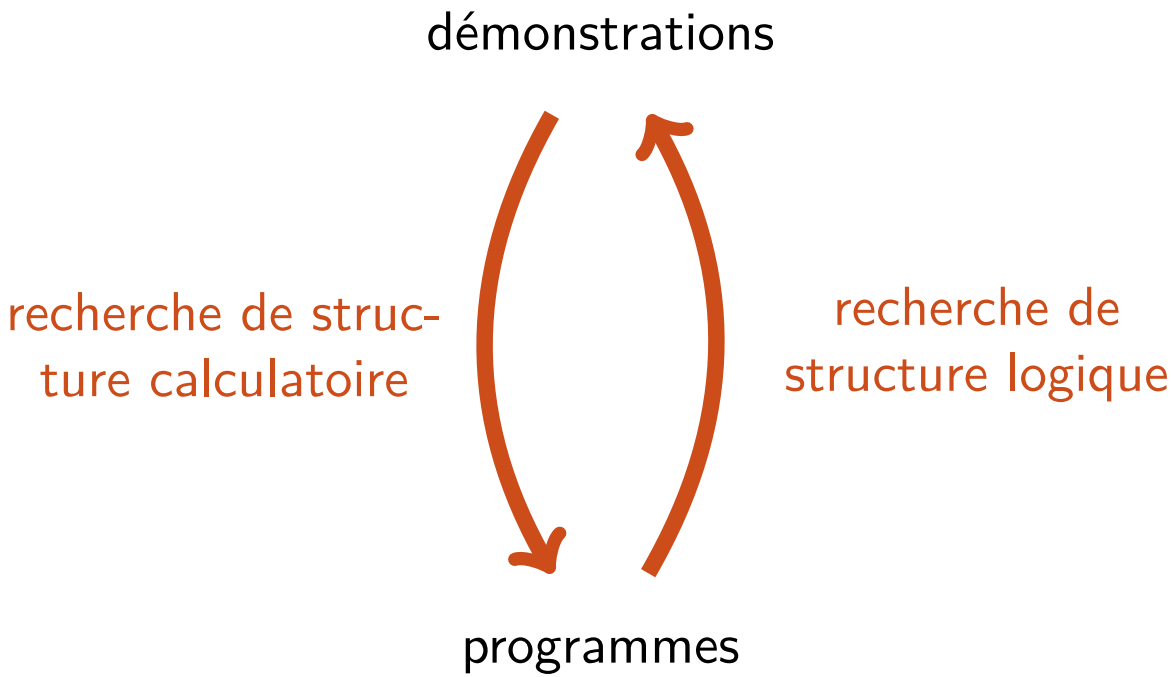
logique	**informatique**
formule	type
démonstration	programme
élimination des coupures	évaluation
règle de déduction	instruction élémentaire
lemme, proposition	sous-programme
théorie	interface de programmation
modèle	bibliothèque, environnement
traductions	compilations

...et les formules changèrent de sens

Quelques tautologies :

- $\alpha \to \alpha$
  l'identité
- $\alpha \to \alpha \to \alpha$
  les booléens
- $(\alpha \to \alpha) \to \alpha \to \alpha$
  les entiers
- $(X \to \alpha \to \alpha) \to \alpha \to \alpha$
  les listes à valeurs dans X
- $A \wedge B = (A \to B \to \alpha) \to \alpha$
  produit cartésien de A et B
- $A \vee B = (A \to \alpha) \to (B \to \alpha) \to \alpha$
  somme directe de A et B

Win-win partnership



Troisième partie

Sémantique dénotationnelle

(et logique linéaire)

Les preuves comme fonctions

Modèle relationnel

Pour aller plus loin

Retour à l'interprétation fonctionnelle

Curry–Howard

une preuve $=$ un λ-terme typé $=$ un programme fonctionnel

Retour à l'interprétation fonctionnelle

Curry–Howard

une preuve $=$ un λ-terme typé $=$ un programme fonctionnel

Et les « vraies » fonctions ?

Rappel : Remarque en passant

Le typage traduit formellement l'intuition selon laquelle les λ-termes sont des fonctions, cette intuition garantit la cohérence du système. C'est la base de la *sémantique dénotationnelle* :

- on choisit un ensemble $[\![\alpha]\!]$ pour chaque type de base α,
- on définit l'ensemble qui interprète les types composés : $[\![A \to B]\!] := [\![B]\!]^{[\![A]\!]}$,
- on interprète un terme de type A par un point de $[\![A]\!]$, de la façon intuitive.

Alors les termes sont interprétés par de vraies fonctions et l'interprétation est invariante par β-réduction.

Rappel : Remarque en passant

Le typage traduit formellement l'intuition selon laquelle les λ-termes sont des fonctions, cette intuition garantit la cohérence du système. C'est la base de la *sémantique dénotationnelle* :

- on choisit un ensemble $[\![\alpha]\!]$ pour chaque type de base α,
- on définit l'ensemble qui interprète les types composés : $[\![A \to B]\!] := [\![B]\!]^{[\![A]\!]}$,
- on interprète un terme de type A par un point de $[\![A]\!]$, de la façon intuitive.

Alors les termes sont interprétés par de vraies fonctions et l'interprétation est invariante par β-réduction.

- donne des notations pratiques (λ)
- curryfication : $A \times B \to C \simeq A \to B \to C$

Rappel : Remarque en passant

Le typage traduit formellement l'intuition selon laquelle les λ-termes sont des fonctions, cette intuition garantit la cohérence du système. C'est la base de la *sémantique dénotationnelle* :

- on choisit un ensemble $[\![\alpha]\!]$ pour chaque type de base α,
- on définit l'ensemble qui interprète les types composés : $[\![A \to B]\!] := [\![B]\!]^{[\![A]\!]}$,
- on interprète un terme de type A par un point de $[\![A]\!]$, de la façon intuitive.

Alors les termes sont interprétés par de vraies fonctions et l'interprétation est invariante par β-réduction.

- donne des notations pratiques (λ)
- curryfication : $A \times B \to C \simeq A \to B \to C$
- il y a trop de morphismes (non dénombrable) et pas assez de structure (du coup on ne peut pas en déduire un modèle du λ-calcul pur)

Rappel : Remarque en passant

Le typage traduit formellement l'intuition selon laquelle les λ-termes sont des fonctions, cette intuition garantit la cohérence du système. C'est la base de la *sémantique dénotationnelle* :

- on choisit un ensemble $[\![\alpha]\!]$ pour chaque type de base α,
- on définit l'ensemble qui interprète les types composés : $[\![A \to B]\!] := [\![B]\!]^{[\![A]\!]}$,
- on interprète un terme de type A par un point de $[\![A]\!]$, de la façon intuitive.

Alors les termes sont interprétés par de vraies fonctions et l'interprétation est invariante par β-réduction.

- donne des notations pratiques (λ)
- curryfication : $A \times B \to C \simeq A \to B \to C$
- il y a trop de morphismes (non dénombrable) et pas assez de structure (du coup on ne peut pas en déduire un modèle du λ-calcul pur)
- on n'apprend rien sur la logique ou le calcul

Sémantique dénotationnelle

Principe

- $[\![\text{un type/une formule}]\!] = \text{un } \textit{espace}$;
- $[\![\text{un terme/une preuve}]\!] = \text{un } \textit{morphisme}$, *i.e.*

$$[\![\Gamma \vdash M : A]\!] : [\![\Gamma]\!] \to [\![A]\!];$$

- $M \rightsquigarrow N$ implique $[\![M]\!] = [\![N]\!]$.

Sémantique dénotationnelle

Principe

- $[\![\text{un type/une formule}]\!] = \text{un } \textit{espace}$;
- $[\![\text{un terme/une preuve}]\!] = \text{un } \textit{morphisme}$, *i.e.*

$$[\![\Gamma \vdash M : A]\!] : [\![\Gamma]\!] \to [\![A]\!];$$

- $M \rightsquigarrow N$ implique $[\![M]\!] = [\![N]\!]$.

- On déplace l'*infini* : de la dynamique vers les objets.

Sémantique dénotationnelle

Principe

- $[\![\text{un type/une formule}]\!] = \text{un } \textit{espace}$;
- $[\![\text{un terme/une preuve}]\!] = \text{un } \textit{morphisme}$, *i.e.*

$$[\![\Gamma \vdash M : A]\!] : [\![\Gamma]\!] \to [\![A]\!];$$

- $M \rightsquigarrow N$ implique $[\![M]\!] = [\![N]\!]$.

- On déplace l'*infini* : de la dynamique vers les objets.
- Avec un peu de chance, on apprend quelque chose.

Sémantique dénotationnelle

Principe

- $[\![\text{un type/une formule}]\!] = \text{un } \textit{espace}$;
- $[\![\text{un terme/une preuve}]\!] = \text{un } \textit{morphisme}$, *i.e.*

$$[\![\Gamma \vdash M : A]\!] : [\![\Gamma]\!] \to [\![A]\!];$$

- $M \rightsquigarrow N$ implique $[\![M]\!] = [\![N]\!]$.

- On déplace l'*infini* : de la dynamique vers les objets.
- Avec un peu de chance, on apprend quelque chose.

modèle dénotationnel	$\neq$	modèle de vérité
interprétation des preuves (niveau -2)	$\neq$	interprétation des formules (niveau -1)

Continuité (Scott)

Idée

Pour produire un bit de résultat, un programme consulte une quantité finie d'information.

Continuité (Scott)

Idée

Pour produire un bit de résultat, un programme consulte une quantité finie d'information.

Le comportement d'un programme est entièrement spécifié par ce qu'il fait sur des approximations finies de l'entrée (en temps fini, on ne lit qu'une partie finie de l'entrée).

Continuité (Scott)

Idée

Pour produire un bit de résultat, un programme consulte une quantité finie d'information.

Le comportement d'un programme est entièrement spécifié par ce qu'il fait sur des approximations finies de l'entrée (en temps fini, on ne lit qu'une partie finie de l'entrée).

Exemple (de truc pas continu)

Le test à zéro sur les flux binaires.

Continuité (Scott)

Idée

Pour produire un bit de résultat, un programme consulte une quantité finie d'information.

Le comportement d'un programme est entièrement spécifié par ce qu'il fait sur des approximations finies de l'entrée (en temps fini, on ne lit qu'une partie finie de l'entrée).

Exemple (de truc pas continu)

Le test à zéro sur les flux binaires.

Originellement formalisé à partir d'une notion d'ordre :
les domaines (des CPOs particuliers).

Stabilité (Berry)

Idée

Lorsqu'un programme produit un bit de résultat, on peut identifier une approximation minimale l'entrée qui correspond à ce bit.

Stabilité (Berry)

Idée

Lorsqu'un programme produit un bit de résultat, on peut identifier une approximation minimale l'entrée qui correspond à ce bit.

C'est ce que le programme a regardé de l'entrée avant de répondre (séquentialité).

Stabilité (Berry)

Idée

Lorsqu'un programme produit un bit de résultat, on peut identifier une approximation minimale l'entrée qui correspond à ce bit.

C'est ce que le programme a regardé de l'entrée avant de répondre (séquentialité).

Exemple (de truc pas stable)

Le OU parallèle.

Cohérence (Girard)

Grandes lignes

Reprend la condition de stabilité pour des domaines très particuliers : espaces = graphes, points = cliques ordonnées par inclusion.

Cohérence (Girard)

Grandes lignes

Reprend la condition de stabilité pour des domaines très particuliers :
espaces = graphes, points = cliques ordonnées par inclusion.

Les fonctions stables entre espaces de cohérence sont caractérisées par leur. . .

. . . trace

$$\mathrm{tr}\, f = \{(a, \beta) \mid a \subset_f |A| \text{ minimal tel que } \beta \in f(a)\}.$$

Cohérence (Girard)

Grandes lignes

Reprend la condition de stabilité pour des domaines très particuliers :
espaces = graphes, points = cliques ordonnées par inclusion.

Les fonctions stables entre espaces de cohérence sont caractérisées par leur. . .

. . . trace

$$\mathrm{tr}\, f = \{(a, \beta) \mid a \subset_f |A| \text{ minimal tel que } \beta \in f(a)\}.$$

Exemple : $\lambda a.a : \mathcal{C}(A) \to \mathcal{C}(A)$

$$\mathrm{tr}(\lambda a.a) = \{(\{\alpha\}, \alpha) \mid \alpha \in a \subset |A|\}$$

Cohérence (Girard)

Grandes lignes

Reprend la condition de stabilité pour des domaines très particuliers :
espaces = graphes, points = cliques ordonnées par inclusion.

Les fonctions stables entre espaces de cohérence sont caractérisées par leur. . .

. . . trace

$$\mathrm{tr}\, f = \{(a, \beta) \mid a \subset_f |A| \text{ minimal tel que } \beta \in f(a)\}.$$

Exemple : $\lambda a.a : \mathcal{C}(A) \to \mathcal{C}(A)$

$$\mathrm{tr}(\lambda a.a) = \{(\{\alpha\}, \alpha) \mid \alpha \in a \subset |A|\}$$

Et les traces forment un espace de cohérence !

Cohérence (Girard)

Grandes lignes

Reprend la condition de stabilité pour des domaines très particuliers :
espaces = graphes, points = cliques ordonnées par inclusion.

Les fonctions stables entre espaces de cohérence sont caractérisées par leur. . .

. . . trace

$$\mathrm{tr}\, f = \{(a, \beta) \mid a \subset_f |A| \text{ minimal tel que } \beta \in f(a)\}.$$

Exemple : $\lambda a.a : \mathcal{C}(A) \to \mathcal{C}(A)$

$$\mathrm{tr}(\lambda a.a) = \{(\{\alpha\}, \alpha) \mid \alpha \in a \subset |A|\}$$

Et les traces forment un espace de cohérence !

- À l'origine de la logique linéaire.

Cohérence (Girard)

Grandes lignes

Reprend la condition de stabilité pour des domaines très particuliers :
espaces = graphes, points = cliques ordonnées par inclusion.

Les fonctions stables entre espaces de cohérence sont caractérisées par leur. . .

. . . trace

$$\mathrm{tr}\, f = \{(a, \beta) \mid a \subset_f |A| \text{ minimal tel que } \beta \in f(a)\}.$$

Exemple : $\lambda a.a : \mathcal{C}(A) \to \mathcal{C}(A)$

$$\mathrm{tr}(\lambda a.a) = \{(\{\alpha\}, \alpha) \mid \alpha \in a \subset |A|\}$$

Et les traces forment un espace de cohérence !

- À l'origine de la logique linéaire.
- On va faire plus simple.

Modèle relationnel

En bref

Principe

- un type $A \rightsquigarrow$ un ensemble $[\![A]\!]$
- on s'en tient aux traces : $s : A \rightsquigarrow [\![s]\!] \subset [\![A]\!]$
- on « compte » les appels à l'argument : $[\![A \to B]\!] = \mathcal{M}_f([\![A]\!]) \times B$

Modèle relationnel

En bref

Principe

- un type $A \rightsquigarrow$ un ensemble $[\![A]\!]$
- on s'en tient aux traces : $s : A \rightsquigarrow [\![s]\!] \subset [\![A]\!]$
- on « compte » les appels à l'argument : $[\![A \to B]\!] = \mathcal{M}_f([\![A]\!]) \times B$

$\mathcal{M}_f(A)$: multiensembles finis (listes modulo permutations) sur A

Modèle relationnel

En bref

Principe

- un type $A \rightsquigarrow$ un ensemble $[\![A]\!]$
- on s'en tient aux traces : $s : A \rightsquigarrow [\![s]\!] \subset [\![A]\!]$
- on « compte » les appels à l'argument : $[\![A \to B]\!] = \mathcal{M}_f([\![A]\!]) \times B$

$\mathcal{M}_f(A)$: multiensembles finis (listes modulo permutations) sur A

Intuition

Pour $s : A \to B$,

$$([\alpha_1, \ldots, \alpha_n], \beta) \in [\![s]\!]$$

se lit :

« s produit β en consommant $\alpha_1, \ldots, \alpha_n$ »

(on compte les multiplicités : notion de ressource).

Types

- on fixe un ensemble $[\![X]\!]$ pour chaque variable propositionnelle X
- $[\![A \to B]\!] = \mathcal{M}_f([\![A]\!]) \times B$

Types

- on fixe un ensemble $[\![X]\!]$ pour chaque variable propositionnelle X
- $[\![A \to B]\!] = \mathcal{M}_f([\![A]\!]) \times B$

Séquents

$$[\![A_1, \ldots, A_n \vdash B]\!] = \mathcal{M}_f([\![A_1]\!]) \times \cdots \times \mathcal{M}_f([\![A_n]\!]) \times [\![B]\!]$$

Types

- on fixe un ensemble $[\![X]\!]$ pour chaque variable propositionnelle X
- $[\![A \to B]\!] = \mathcal{M}_f([\![A]\!]) \times B$

Séquents

$$[\![A_1, \ldots, A_n \vdash B]\!] = \mathcal{M}_f([\![A_1]\!]) \times \cdots \times \mathcal{M}_f([\![A_n]\!]) \times [\![B]\!]$$

Termes

Si $\Gamma \vdash s : A$, on veut $[\![s]\!] \subset [\![\Gamma \vdash A]\!]$.
On définit un système d'inférence :

$$(\overline{\alpha}_1, \ldots, \overline{\alpha}_n, \beta) \in [\![s]\!] \quad \text{ssi} \quad x_1^{\overline{\alpha}_1} : A_1, \ldots, x_n^{\overline{\alpha}_n} : A_n \vdash s^{\beta} : B$$

$$\frac{}{\Gamma^{[]}, x^{[\alpha]} : A \vdash x^{\alpha} : A}$$

$$\frac{\Gamma^{\overline{\gamma}}, x^{\overline{\alpha}} : A \vdash s^{\beta} : B}{\Gamma^{\overline{\gamma}} \vdash \lambda x.s^{(\overline{\alpha},\beta)} : A \to B}$$

$$\frac{\Gamma^{\overline{\gamma}_0} \vdash s^{([\alpha_1,\ldots,\alpha_k],\beta)} : A \to B \quad \Gamma^{\overline{\gamma}_1} \vdash t^{\alpha_1} : A \quad \cdots \quad \Gamma^{\overline{\gamma}_k} \vdash t^{\alpha_k} : A}{\Gamma^{\sum_{j=0}^{k} \overline{\gamma}_j} \vdash (s)t^{\beta} : B}$$

Calculer les sémantiques de :

- $\lambda x.x : A \to A$
- $\lambda x.\lambda y.(x)y : (A \to B) \to A \to B$
- $\lambda x.\lambda y.(x)(x)y : (A \to A) \to A \to A$

Calculer les sémantiques de :

- $\lambda x.x : A \to A$
- $\lambda x.\lambda y.(x)y : (A \to B) \to A \to B$
- $\lambda x.\lambda y.(x)(x)y : (A \to A) \to A \to A$

Voir page 111

Modèle relationnel

C'est une sémantique dénotationnelle

Théorème

Si $\Gamma \vdash s : B$ *et* $s =_\beta t$ *alors* $[\![s]\!] = [\![t]\!]$.

Cas principal de la démonstration.

Si $\Gamma, x : A \vdash s : B$ et $\Gamma \vdash t : A$, alors $[\![(\lambda x.s)t]\!] = [\![s[t/x]]\!]$. □

Modèle relationnel pour le λ-calcul pur

Idée

Si on résout l'équation $D = D \to D = \mathcal{M}_f(D) \times D$, on peut typer tous les λ-termes avec D.

Modèle relationnel pour le λ-calcul pur

Idée

Si on résout l'équation $D = D \to D = \mathcal{M}_f(D) \times D$, on peut typer tous les λ-termes avec D.

Il y a des solutions.

Voir page 116 : Définition 3.2.6

Modèle relationnel pour le λ-calcul pur

Idée

Si on résout l'équation $D = D \to D = \mathcal{M}_f(D) \times D$, on peut typer tous les λ-termes avec D.

Il y a des solutions.

Voir page 116 : Définition 3.2.6

Théorème (Normalisation)

- $[\![s]\!] \neq \emptyset$ *ssi* s *a une forme normale de tête*
- s *a une forme normale ssi il y a un élément de* $[\![s]\!]$ *qui « ne triche pas »*

Modèle relationnel pour le λ-calcul pur

Idée

Si on résout l'équation $D = D \to D = \mathcal{M}_f(D) \times D$, on peut typer tous les λ-termes avec D.

Il y a des solutions.

Voir page 116 : Définition 3.2.6

Théorème (Normalisation)

- $[\![s]\!] \neq \emptyset$ *ssi* s *a une forme normale de tête*
- s *a une forme normale ssi il y a un élément de* $[\![s]\!]$ *qui « ne triche pas »*

Théorème (*Full abstraction*)

On capture précisément la dynamique.

Modèle relationnel pour le λ-calcul pur

Idée

Si on résout l'équation $D = D \to D = \mathcal{M}_f(D) \times D$, on peut typer tous les λ-termes avec D.

Il y a des solutions.

Voir page 116 : Définition 3.2.6

Théorème (Normalisation)

- $[\![s]\!] \neq \emptyset$ *ssi* s *a une forme normale de tête*
- s *a une forme normale ssi il y a un élément de* $[\![s]\!]$ *qui « ne triche pas »*

Théorème (*Full abstraction*)

On capture précisément la dynamique.

On peut même parler de complexité.

On peut décomposer

$$\frac{\Gamma^{\overline{\gamma}_0} \vdash s^{([\alpha_1,\ldots,\alpha_k],\beta)} : A \to B \qquad \Gamma^{\overline{\gamma}_1} \vdash t^{\alpha_1} : A \qquad \cdots \qquad \Gamma^{\overline{\gamma}_k} \vdash t^{\alpha_k} : A}{\Gamma^{\sum_{j=0}^{k} \overline{\gamma}_j} \vdash (s)t^{\beta} : B}$$

en

$$\frac{\Gamma^{\overline{\gamma}_1} \vdash t^{\alpha_1} : A \qquad \cdots \qquad \Gamma^{\overline{\gamma}_k} \vdash t^{\alpha_k} : A}{\Gamma^{\sum_{j=1}^{k} \overline{\gamma}_j} \vdash t^{[\alpha_1,\ldots,\alpha_k]} : !A}$$

et

$$\frac{\Gamma^{\overline{\gamma}} \vdash s^{(\overline{\alpha},\beta)} : A \to B \qquad \Gamma^{\overline{\gamma}'} \vdash t^{\overline{\alpha}} : !A}{\Gamma^{\overline{\gamma}+\overline{\gamma}'} \vdash (s)t^{\beta} : B}$$

On peut décomposer

$$\frac{\Gamma^{\overline{\gamma}_0} \vdash s^{([\alpha_1,\ldots,\alpha_k],\beta)} : A \to B \qquad \Gamma^{\overline{\gamma}_1} \vdash t^{\alpha_1} : A \qquad \cdots \qquad \Gamma^{\overline{\gamma}_k} \vdash t^{\alpha_k} : A}{\Gamma^{\sum_{j=0}^{k} \overline{\gamma}_j} \vdash (s)t^{\beta} : B}$$

en

$$\frac{\Gamma^{\overline{\gamma}_1} \vdash t^{\alpha_1} : A \qquad \cdots \qquad \Gamma^{\overline{\gamma}_k} \vdash t^{\alpha_k} : A}{\Gamma^{\sum_{j=1}^{k} \overline{\gamma}_j} \vdash t^{[\alpha_1,\ldots,\alpha_k]} : !A}$$

et

$$\frac{\Gamma^{\overline{\gamma}} \vdash s^{(\overline{\alpha},\beta)} : A \to B \qquad \Gamma^{\overline{\gamma}'} \vdash t^{\overline{\alpha}} : !A}{\Gamma^{\overline{\gamma}+\overline{\gamma}'} \vdash (s)t^{\beta} : B}$$

On sépare la génération des ressources (modalité !) de la partie purement implicative :

$$A \to B = !A \multimap B$$

On peut décomposer

$$\frac{\Gamma^{\overline{\gamma}_0} \vdash s^{([\alpha_1,\ldots,\alpha_k],\beta)} : A \to B \qquad \Gamma^{\overline{\gamma}_1} \vdash t^{\alpha_1} : A \qquad \cdots \qquad \Gamma^{\overline{\gamma}_k} \vdash t^{\alpha_k} : A}{\Gamma^{\sum_{j=0}^{k} \overline{\gamma}_j} \vdash (s)t^{\beta} : B}$$

en

$$\frac{\Gamma^{\overline{\gamma}_1} \vdash t^{\alpha_1} : A \qquad \cdots \qquad \Gamma^{\overline{\gamma}_k} \vdash t^{\alpha_k} : A}{\Gamma^{\sum_{j=1}^{k} \overline{\gamma}_j} \vdash t^{[\alpha_1,\ldots,\alpha_k]} : !A}$$

et

$$\frac{\Gamma^{\overline{\gamma}} \vdash s^{(\overline{\alpha},\beta)} : A \to B \qquad \Gamma^{\overline{\gamma}'} \vdash t^{\overline{\alpha}} : !A}{\Gamma^{\overline{\gamma}+\overline{\gamma}'} \vdash (s)t^{\beta} : B}$$

On sépare la génération des ressources (modalité !) de la partie purement implicative :

$$A \to B = !A \multimap B$$

Nous voilà dans les années (19)80...

Sémantique quantitative

Un autre regard sur la « continuité » : fonctions analytiques.

On interprète les types/formules comme des espaces vectoriels et les termes/preuves par des séries entières : si $t : A \to B$ et $u : A$,

$$((t)u)_\beta = \sum_{\overline{a}} t_{(\overline{a},\beta)} \cdot u^{\overline{a}}$$

où $u^{[\alpha_1,\ldots,\alpha_n]} = \prod_i u_{\alpha_i}$.

Version quantitative de $\beta \in tu$ ssi $\exists \overline{a} \in \mathcal{M}_f(u),\ (\overline{a},\beta) \in t$.

Il faut un truc en plus pour que la somme converge :

- de gros coefficients (façon Girard, $\sim$1985) ;
- de la topologie (façon Ehrhard, $\sim$2000) ;
- des sommes finies : espaces de finitude.

Quatrième partie

Au delà du fonctionnel

Données et quantification

- Second ordre : le polymorphisme
- Premier ordre : la spécification

Contrôle et logique classique

- Contrôle en λ-calcul
- Interprétation du raisonnement classique

Des théorèmes standard comme spécifications

Au delà du calcul fonctionnel

Dans sa plus simple expression, Curry-Howard établit une correspondance entre programmation fonctionnelle pure et calcul propositionnel intuitionniste. Mais...

En programmation il y a

- des effets de bord (variables globales, références...)
- des structures de contrôle (exceptions, continuations...)
- des entrées-sorties

Dans le raisonnement mathématique, il y a

- des objets, des quantificateurs, des axiomes
- du raisonnement par l'absurde

Si la correspondance n'est pas fortuite, on doit pouvoir l'étendre en faisant le lien entre ces choses-là.

Typage des entiers de Church

Typons un entier de Church :

$$\dfrac{\dfrac{}{f : A \to A \vdash f : A \to A} \quad \dfrac{\dfrac{}{f : A \to A \vdash f : A \to A} \quad \dfrac{}{x : A \vdash x : A}}{f : A \to A, x : A \vdash (f)x : A}}{\dfrac{\dfrac{f : A \to A, x : A \vdash (f)(f)x : A}{f : A \to A \vdash \lambda x.(f)(f)x : A \to A}}{\vdash \lambda f.\lambda x.(f)(f)x : (A \to A) \to A \to A}}$$

De façon générale :

- notons $\mathbb{N}_A$ le type $(A \to A) \to A \to A$
- pour tout type A et tout entier n, on a $\vdash \lambda f.\lambda x.(f)^n x : \mathbb{N}_A$.

Typage des entiers de Church

Là où les problèmes commencent

Peut-on typer les opérations qui calculent avec ces entiers ?

- L'addition :
  $\lambda m.\lambda n.\lambda f.\lambda x.((m)f)((n)f)x : \mathbb{N}_A \to \mathbb{N}_A \to \mathbb{N}_A$

Typage des entiers de Church

Là où les problèmes commencent

Peut-on typer les opérations qui calculent avec ces entiers ?

- L'addition :
  $\lambda m.\lambda n.\lambda f.\lambda x.((m)f)((n)f)x : \mathbb{N}_A \to \mathbb{N}_A \to \mathbb{N}_A$
- La multiplication :
  $\lambda m.\lambda n.\lambda f.\lambda x.((m)(n)f)x : \mathbb{N}_A \to \mathbb{N}_A \to \mathbb{N}_A$

Typage des entiers de Church

Là où les problèmes commencent

Peut-on typer les opérations qui calculent avec ces entiers ?

- L'addition :
  $\lambda m.\lambda n.\lambda f.\lambda x.((m)f)((n)f)x : \mathbb{N}_A \to \mathbb{N}_A \to \mathbb{N}_A$
- La multiplication :
  $\lambda m.\lambda n.\lambda f.\lambda x.((m)(n)f)x : \mathbb{N}_A \to \mathbb{N}_A \to \mathbb{N}_A$
- L'exponentiation :
  $\lambda m.\lambda n.\lambda f.\lambda x.(((m)n)f)x : \mathbb{N}_{A\to A} \to \mathbb{N}_A \to \mathbb{N}_A$

Typage des entiers de Church

Là où les problèmes commencent

Peut-on typer les opérations qui calculent avec ces entiers ?

- L'addition :
  $\lambda m.\lambda n.\lambda f.\lambda x.((m)f)((n)f)x : \mathbb{N}_A \to \mathbb{N}_A \to \mathbb{N}_A$
- La multiplication :
  $\lambda m.\lambda n.\lambda f.\lambda x.((m)(n)f)x : \mathbb{N}_A \to \mathbb{N}_A \to \mathbb{N}_A$
- L'exponentiation :
  $\lambda m.\lambda n.\lambda f.\lambda x.(((m)n)f)x : \mathbb{N}_{A\to A} \to \mathbb{N}_A \to \mathbb{N}_A$
- Comment typer la fonction $a \mapsto b \mapsto a^b + a$?

Typage des entiers de Church

Là où les problèmes commencent

Peut-on typer les opérations qui calculent avec ces entiers ?

- L'addition :
  $\lambda m.\lambda n.\lambda f.\lambda x.((m)f)((n)f)x : \mathbb{N}_A \to \mathbb{N}_A \to \mathbb{N}_A$
- La multiplication :
  $\lambda m.\lambda n.\lambda f.\lambda x.((m)(n)f)x : \mathbb{N}_A \to \mathbb{N}_A \to \mathbb{N}_A$
- L'exponentiation :
  $\lambda m.\lambda n.\lambda f.\lambda x.(((m)n)f)x : \mathbb{N}_{A\to A} \to \mathbb{N}_A \to \mathbb{N}_A$
- Comment typer la fonction $a \mapsto b \mapsto a^b + a$?

Théorème (Schwichtenberg, 1976)

Les λ-termes de type $\mathbb{N}_\alpha \to \mathbb{N}_\alpha$ définissent exactement les fonctions polynomiales (généralisées avec des conditionnelles).

Quantification sur les types

On étend le langage des types :

$$
\begin{array}{rll}
A, B := & \alpha & \text{type de base} \\
& A \to B & \text{implication} \\
& \bot & \text{faux} \\
& X & \text{variable de type} \\
& \forall X\, A & \text{type polymorphe}
\end{array}
$$

On étend les règles de typage en conséquence :

$$
\frac{\Gamma \vdash M : A \qquad X \notin \Gamma}{\Gamma \vdash M : \forall X\, A}
\qquad\qquad
\frac{\Gamma \vdash M : \forall X\, A}{\Gamma \vdash M : A[B/X]}
$$

C'est du *polymorphisme* : $M : \forall X\, A(X)$ signifie que M a le type $A(B)$ pour tous les choix possibles de B, et c'est écrit dans le type.

Polymorphisme et données

Posons maintenant $\mathbb{N} = \forall X\, \mathbb{N}_X = \forall X\, ((X \to X) \to X \to X)$.

- Les entiers de Church ont tous le type $\mathbb{N}$.

Polymorphisme et données

Posons maintenant $\mathbb{N} = \forall X\, \mathbb{N}_X = \forall X\, ((X \to X) \to X \to X)$.

- Les entiers de Church ont tous le type $\mathbb{N}$.
- Toutes les fonctions récursives primitives (et pas seulement elles) sont représentables par des termes de type $\mathbb{N} \to \mathbb{N}$.

Exercice page 127 : vérifier que l'addition a le type $\mathbb{N} \to \mathbb{N} \to \mathbb{N}$

Polymorphisme et données

Posons maintenant $\mathbb{N} = \forall X\, \mathbb{N}_X = \forall X\, ((X \to X) \to X \to X)$.

- Les entiers de Church ont tous le type $\mathbb{N}$.
- Toutes les fonctions récursives primitives (et pas seulement elles) sont représentables par des termes de type $\mathbb{N} \to \mathbb{N}$.

  Exercice page 127 : vérifier que l'addition a le type $\mathbb{N} \to \mathbb{N} \to \mathbb{N}$
- Mieux : les entiers de Church sont les seuls termes de type $\mathbb{N}$ modulo $\beta\eta$-équivalence
- Les termes de type $\mathbb{N} \to \mathbb{N}$ définissent tous des fonctions totales sur les entiers.

Polymorphisme et données

Posons maintenant $\mathbb{N} = \forall X\, \mathbb{N}_X = \forall X\, ((X \to X) \to X \to X)$.

- Les entiers de Church ont tous le type $\mathbb{N}$.
- Toutes les fonctions récursives primitives (et pas seulement elles) sont représentables par des termes de type $\mathbb{N} \to \mathbb{N}$.

  Exercice page 127 : vérifier que l'addition a le type $\mathbb{N} \to \mathbb{N} \to \mathbb{N}$
- Mieux : les entiers de Church sont les seuls termes de type $\mathbb{N}$ modulo $\beta\eta$-équivalence
- Les termes de type $\mathbb{N} \to \mathbb{N}$ définissent tous des fonctions totales sur les entiers.

Sur le même principe, on peut définir la plupart des types de données.

Expressivité du polymorphisme

Les théorèmes fondamentaux fonctionnent toujours :

- le typage est préservé par β-réduction,
- les termes typés n'ont pas de suite infinie de réductions.

Expressivité du polymorphisme

Les théorèmes fondamentaux fonctionnent toujours :

- le typage est préservé par β-réduction,
- les termes typés n'ont pas de suite infinie de réductions.

On peut représenter les autres connecteurs logiques :

- $\bot := \forall X\, X$ vide
- $A \wedge B := \forall X((A \to B \to X) \to X)$ produit cartésien
- $A \vee B := \forall X((A \to X) \to (B \to X) \to X)$ union disjointe

Expressivité du polymorphisme

Les théorèmes fondamentaux fonctionnent toujours :

- le typage est préservé par β-réduction,
- les termes typés n'ont pas de suite infinie de réductions.

On peut représenter les autres connecteurs logiques :

- $\bot := \forall X\, X$ vide
- $A \wedge B := \forall X((A \to B \to X) \to X)$ produit cartésien
- $A \vee B := \forall X((A \to X) \to (B \to X) \to X)$ union disjointe

Un indice sur la puissance du système de types :

- la typabilité dans les types simples est décidable
  (en temps élémentaire — c'est l'inférence de type en Caml)
- la typabilité avec polymorphisme est indécidable.

Le calcul des prédicats

Considérons un énoncé de l'arithmétique :

> Si n est un produit de deux entiers alors l'un de ces entiers est 1.

Le calcul des prédicats

Considérons un énoncé de l'arithmétique :

$$\forall a \in \mathbb{N}\, \forall b \in \mathbb{N}\, n = a \times b \rightarrow (a = 1 \vee b = 1)$$

Le calcul des prédicats

Considérons un énoncé de l'arithmétique :

$$\forall a \in \mathbb{N}\, \forall b \in \mathbb{N}\; n = a \times b \rightarrow (a = 1 \vee b = 1)$$

Peut-on interpréter une telle formule comme un type ?

Le calcul des prédicats

Considérons un énoncé de l'arithmétique :

$$\forall a \in \mathbb{N}\, \forall b \in \mathbb{N}\; n = a \times b \to (a = 1 \vee b = 1)$$

Peut-on interpréter une telle formule comme un type ?

Intuitivement, un terme de ce type :

- prend deux entiers a et b en arguments,
- attend une justification que $a \times b = n$,
- renvoie une justification que $a = 1$ ou $b = 1$.

Typer avec de l'arithmétique

On décore les variables de types avec des termes et on autorise la quantification :

$$
\begin{array}{rll}
A, B := & X(t_1, \ldots, t_k) & \text{variable de type} \\
& A \to B & \text{implication} \\
& \forall^2 X\, A & \text{polymorphisme} \\
& \forall^1 x\, A & \text{quantification du premier ordre}
\end{array}
$$

Les t_i sont des termes d'un langage choisi une fois pour toutes.
Par exemple, pour l'arithmétique, on a des expressions formées avec des variables et $0, S, +, \times$.

Typer avec de l'arithmétique

On décore les variables de types avec des termes et on autorise la quantification :

$$
\begin{array}{rll}
A, B := & X(t_1, \ldots, t_k) & \text{variable de type} \\
& A \to B & \text{implication} \\
& \forall^2 X\, A & \text{polymorphisme} \\
& \forall^1 x\, A & \text{quantification du premier ordre}
\end{array}
$$

Les t_i sont des termes d'un langage choisi une fois pour toutes.
Par exemple, pour l'arithmétique, on a des expressions formées avec des variables et $0, S, +, \times$.

Mêmes règles de typage que pour le polymorphisme :

$$\frac{\Gamma \vdash M : A \qquad x \notin \Gamma}{\Gamma \vdash M : \forall^1 x\, A} \qquad \frac{\Gamma \vdash M : \forall^1 x\, A}{\Gamma \vdash M : A[t/x]}$$

Décoration des entiers

Raffinement du type des entiers de Church :

$$\mathbb{N}(n) := \forall T\ (\forall x\ T(x) \to T(S(x))) \to T(0) \to T(n)$$

Décoration des entiers

Raffinement du type des entiers de Church :

$$\mathbb{N}(n) := \forall T\ (\forall x\ T(x) \to T(S(x))) \to T(0) \to T(n)$$

C'est le type d'une itération, sauf que

- on a une famille de types T_i,
- pour chaque i, la fonction f va de T_i dans T_{i+1},
- la valeur initiale x est dans T_0,
- alors $f^n(x)$ est dans T_n.

Décoration des entiers

Raffinement du type des entiers de Church :

$$\mathbb{N}(n) := \forall T\ (\forall x\ T(x) \to T(S(x))) \to T(0) \to T(n)$$

C'est le type d'une itération, sauf que

- on a une famille de types T_i,
- pour chaque i, la fonction f va de T_i dans T_{i+1},
- la valeur initiale x est dans T_0,
- alors $f^n(x)$ est dans T_n.

La caractérisation des entiers est raffinée :

- pour tout n, le terme $\underline{n}$ est de type $\mathbb{N}(S^n(0))$,
- tout terme de type $\mathbb{N}(S^n(0))$ est équivalent à $\underline{n}$.

Les types comme spécifications

Le type d'une fonction décrit ce qu'elle calcule

$$\text{add} : \forall m \, \forall n \, \mathbb{N}(m) \to \mathbb{N}(n) \to \mathbb{N}(m+n)$$

Les types comme spécifications

Le type d'une fonction décrit ce qu'elle calcule

$$\text{add} : \forall m \forall n \, \mathbb{N}(m) \to \mathbb{N}(n) \to \mathbb{N}(m+n)$$

Réciproquement, un type spécifie un comportement :

$$M : \forall m \forall n \, \mathbb{N}(m) \to \mathbb{N}(n) \to \mathbb{N}(m \times n + n + S^3(0))$$

un tel terme représente forcément la fonction $\langle x, y \rangle \mapsto xy + y + 3$.

Les types comme spécifications

Le type d'une fonction décrit ce qu'elle calcule

$$\text{add} : \forall m \forall n \, \mathbb{N}(m) \to \mathbb{N}(n) \to \mathbb{N}(m+n)$$

Réciproquement, un type spécifie un comportement :

$$M : \forall m \forall n \, \mathbb{N}(m) \to \mathbb{N}(n) \to \mathbb{N}(m \times n + n + S^3(0))$$

un tel terme représente forcément la fonction $\langle x, y \rangle \mapsto xy + y + 3$.

Les énoncés mathématiques font des choses :

$$\forall p \, \forall q \, \mathbb{N}(p) \to \mathbb{N}(q) \to (p \times q = n) \to ((p = 1) \vee (q = 1))$$

Apparté si vous le souhaitez :
comment définir l'égalité dans notre système.

Digression : les assistants de preuve

Coq, Agda, Isabelle . . . pour quoi faire ?

- Écrire à la main une démonstration d'un résultat mathématique en déduction naturelle est un bon moyen de devenir fou.

  les outils facilitent l'écriture en automatisant partiellement

- Une preuve induit un λ-terme qui peut être vu comme un programme . . .

  l'extraction de programme permet de transformer ce terme en un vrai programme

En plus, un programme extrait d'une preuve est *certifié* :
par construction, il respecte sa spécification.

Et la logique *classique* ?

Jusque là, on est resté en logique *intuitionniste*, en particulier $\neg\neg A \to A$ n'est pas démontrable en général.

Que signifie le type $\bot$?

- $\bot = \forall X\, X$, un terme de type $\bot$ doit se comporter « bien » quel que soit le type attendu.
- Est-ce possible ?

Et la logique *classique* ?

Jusque là, on est resté en logique *intuitionniste*, en particulier $\neg\neg A \to A$ n'est pas démontrable en général.

Que signifie le type $\bot$?

- $\bot = \forall X\, X$, un terme de type $\bot$ doit se comporter « bien » quel que soit le type attendu.
- Est-ce possible ? En vraie programmation, oui !

Et la logique *classique* ?

Jusque là, on est resté en logique *intuitionniste*, en particulier $\neg\neg A \to A$ n'est pas démontrable en général.

Que signifie le type $\bot$?

- $\bot = \forall X\, X$, un terme de type $\bot$ doit se comporter « bien » quel que soit le type attendu.
- Est-ce possible ? En vraie programmation, oui !
- C'est une levée d'exception,
  un programme qui ne rend pas la main à son contexte mais saute à un autre point d'exécution directement.

Il s'agit de *contrôle* du flot d'exécution.

Contrôle en λ-calcul

Considérons une preuve qui utilise la loi de Peirce :

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma, k : \neg A \vdash M : A\end{array}}{\begin{array}{c}\Gamma \vdash \mathcal{C}_k[M] : A\\ \vdots\end{array}}$$

Contrôle en λ-calcul

Considérons une preuve qui utilise la loi de Peirce :

$$\dfrac{\dfrac{\vdots}{\Gamma, k : \neg A \vdash N : A} \qquad \dfrac{}{\Gamma, k : \neg A \vdash k : A \to \bot}}{\Gamma, k : \neg A \vdash (k)N : \bot}$$

$$\vdots$$

$$\dfrac{\Gamma, k : \neg A \vdash M : A}{\Gamma \vdash \mathcal{C}_k[M] : A}$$

$$\vdots$$

Comment éliminer les coupures ?

Contrôle en λ-calcul

Considérons une preuve qui utilise la loi de Peirce :

$$\dfrac{\dfrac{\vdots}{\Gamma, k : \neg A \vdash N : A} \qquad \dfrac{}{\Gamma, k : \neg A \vdash k : A \to \bot}}{\Gamma, k : \neg A \vdash (k)N : \bot}$$

$$\vdots$$

$$\dfrac{\Gamma, k : \neg A \vdash M : A}{\Gamma \vdash \mathcal{C}_k[M] : A}$$

$$\vdots$$

Comment éliminer les coupures ?

Contrôle en λ-calcul

Considérons une preuve qui utilise la loi de Peirce :

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma, k : \neg A \vdash N : A\end{array}}{\begin{array}{c}\Gamma \vdash \mathcal{C}_k[N] : A\\ \vdots\end{array}}$$

Comment éliminer les coupures ?

Contrôle en λ-calcul

Considérons une preuve qui utilise la loi de Peirce :

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma, k : \neg A \vdash N : A\end{array}}{\begin{array}{c}\Gamma \vdash \mathcal{C}_k[N] : A\\ \vdots\end{array}}$$

C'est un peu approximatif ...

La machine de Krivine (simplifiée)

Pour avoir une notion de contrôle en λ-calcul, il faut traiter explicitement un programme et son contexte.

- Un état de la machine (M et les N_i sont des λ-termes clos) :

$$\underbrace{M}_{\text{programme}} \star \underbrace{N_1 \cdots N_k}_{\text{pile}}$$

- Instructions de base :

$$\begin{array}{lcll} (M)N \star \pi & \rightsquigarrow & M \star N \cdot \pi & \text{push } N; M \\ \lambda x.M \star N \cdot \pi & \rightsquigarrow & M[N/x] \star \pi & \text{pop } x; M \end{array}$$

  On ajoutera plus tard d'autres instructions.

- Cette machine calcule des « formes normales de tête faibles »

Définition interactive des types

Ça n'a plus de sens de formuler la cohérence comme préservation du typage par réduction, parce qu'il n'y a plus de réduction des termes !

Définition interactive des types

Ça n'a plus de sens de formuler la cohérence comme préservation du typage par réduction, parce qu'il n'y a plus de réduction des termes !

- On se donne une observation $\perp\!\!\!\perp$ = un ensemble d'états (ceux qui terminent, ceux qui finissent sur l'entier 42...)
- On déduit une dualité et une équivalence observationnelle :

$$M \perp\!\!\!\perp N_1 \cdots N_2 \iff M \star N_1 \cdots N_2 \in \perp\!\!\!\perp$$
$$M \simeq N \iff \forall \pi,\ M \perp\!\!\!\perp \pi \text{ ssi } N \perp\!\!\!\perp \pi$$

- Puis une interprétation des types :

$$M \in [\![A \to B]\!] \iff \forall N \in [\![A]\!], \forall \pi \perp\!\!\!\perp [\![B]\!],\ M \perp\!\!\!\perp N \cdot \pi$$

  Notons qu'on a $[\![A]\!] = [\![A]\!]^{\perp\!\!\!\perp\perp\!\!\!\perp}$.
- Les termes typés réalisent leur type, pas l'inverse en général.

Le sens de l'absurde

Interprétation de la négation :

- la formule $\bot$ est le type de ce qui ne rend pas la main
- $\neg A = A \to \bot$ reçoit du A et ne rend pas la main
  c'est un continuation qui attend du A

Le sens de l'absurde

Interprétation de la négation :

- la formule $\bot$ est le type de ce qui ne rend pas la main
- $\neg A = A \to \bot$ reçoit du A et ne rend pas la main
  c'est un continuation qui attend du A

On étend la machine de Krivine avec des instructions de contrôle :

$$\begin{array}{lll} \mathsf{cc} \star M \cdot \pi \rightsquigarrow M \star k_\pi \cdot \pi & & \text{call/cc} \\ k_\pi \star M \cdot \theta \rightsquigarrow M \star \pi & & \text{restore } \pi \end{array}$$

alors cc *réalise* le type $\forall X \forall Y ((X \to Y) \to X) \to X$ loi de Peirce

Le sens de l'absurde

Interprétation de la négation :

- la formule $\bot$ est le type de ce qui ne rend pas la main
- $\neg A = A \to \bot$ reçoit du A et ne rend pas la main
  c'est un continuation qui attend du A

On étend la machine de Krivine avec des instructions de contrôle :

$$\begin{array}{lll} \mathsf{cc} \star M \cdot \pi \rightsquigarrow M \star k_\pi \cdot \pi & & \text{call/cc} \\ k_\pi \star M \cdot \theta \rightsquigarrow M \star \pi & & \text{restore } \pi \end{array}$$

alors cc *réalise* le type $\forall X \forall Y ((X \to Y) \to X) \to X$ loi de Peirce

La logique intuitionniste étendue avec la loi de Peirce comme axiome démontre les mêmes choses que la logique classique : on peut maintenant interpréter comme un programme n'importe quelle démonstration en logique du premier ordre.

Le tiers exclu

Le terme

$$T := \lambda f.\lambda g.\mathrm{cc}(\lambda k.f(\lambda a.k(ga)))$$

```
pop f; pop g; call/cc k;
push { pop a; push { push a ; jump g }; jump k }; jump f
```

est de type

$$\forall X\ (\neg A \to X) \to (A \to X) \to X \quad = \quad \neg\alpha \vee \alpha$$

et il implémente un gestionnaire d'exceptions :

$$\begin{array}{rll} T \star f \cdot g \cdot \pi & \rightsquigarrow f \star e_{g,\pi} \cdot \pi & \text{try } f \text{ with } e \to g \\ e_{g,\pi} \star a \cdot \theta & \rightsquigarrow g \star a \cdot \pi & \text{raise } e(a) \end{array}$$

se généralise à des structure de contrôle plus complexes.

Théorie des ensembles

La même technique appliquée à la théorie des ensembles est un programme de recherche...

l'axiome de fondation

$$\forall a.(\forall x.(x \in a \to P(x)) \to P(a)) \to \forall a.P(a)$$

est le type du point fixe : Y tel que $Y\phi = \phi(Y\phi)$.
Mais prouver que ϕ est bien typé revient à prouver que la récursion termine !

l'axiome du choix spécifie une horloge...

l'axiome de l'ultrafiltre spécifie un gestionnaire de mémoire en ajoutant un état global à la machine...