Lambda calculs et catégories

Paul-André Melliès

Master Parisien de Recherche en Informatique

Paris, Octobre 2010

Plan de la séance

1 – Lambda-calcul typé du second ordre

2 – Une sémantique opérationnelle

3 – Une topologie

4 – Les variétés comme type sémantique

5 – Théorème fondamental

6 – Application: théorème de normalisation

Part I

Second order lambda calculus

Polymorphism

Curry 1958: the simply typed λ-calculus

It is possible to type the λ-terms by simple types A, B constructed by the grammar:

$$A, B ::= \alpha \mid A \Rightarrow B.$$

A typing context Γ is a finite list $\Gamma = (x_1 : A_1, ..., x_n : A_n)$ where x_i is a variable and A_i is a simple type, for all $1 \leq i \leq n$.

A sequent is a triple:

$$x_1 : A_1, ..., x_n : A_n \vdash P : B$$

where $x_1 : A_1, ..., x_n : A_n$ is a typing context, P is a λ-term and B is a simple type.

Curry 1958: the simply-typed λ-calculus

Variable

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

Abstraction

$$\frac{\Gamma, x : A \vdash P : B}{\Gamma \vdash \lambda x.P : A \Rightarrow B}$$

Application

$$\frac{\Gamma \vdash P : A \Rightarrow B \qquad \Delta \vdash Q : A}{\Gamma, \Delta \vdash PQ : B}$$

Weakening

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

Contraction

$$\frac{\Gamma, x : A, y : A \vdash P : B}{\Gamma, z : A \vdash P[x, y \leftarrow z] : B}$$

Permutation

$$\frac{\Gamma, x : A, y : B, \Delta \vdash P : C}{\Gamma, y : B, x : A, \Delta \vdash P : C}$$

Girard 1972: second-order λ-calculus

The idea is to extend the usual simply typed lambda-calculus with second-order quantification on type variables.

Types are simple types extended with second-order variables:

$$A, B ::= \alpha \mid A \Rightarrow B \mid \forall\alpha.A$$

A typing context Γ is a finite list constructed by the grammar

$$\Gamma = \texttt{nil} \quad | \quad \Gamma, x : A \quad | \quad \Gamma, \alpha$$

where:

- `nil` is the empty list
- x is a term variable and A is a type,
- α is a type variable.

Girard 1972: second-order λ-calculus

Second order abstraction

$$\frac{\Gamma, \alpha \vdash P : A}{\Gamma \vdash P : \forall \alpha . A}$$

Second order application

$$\frac{\Gamma \vdash P : \forall \alpha . A}{\Gamma \vdash P : A[\alpha := B]}$$

Properties of second-order polymorphism

A λ-term P is typed when there exists a typing context Γ and a second-order type A such that:

$$\Gamma \vdash P : A$$

One establishes that the set of typed λ-terms is closed under β-réduction:

Subject Reduction: If $\Gamma \vdash P : A$ and $P \longrightarrow_\beta Q$, then $\Gamma \vdash Q : A$.

A λ-term P is strongly normalizing when all the rewriting paths based on β-reduction:

$$P \longrightarrow_\beta P_1 \longrightarrow_\beta P_2 \longrightarrow_\beta \cdots \longrightarrow_\beta P_n \longrightarrow_\beta \cdots$$

terminate.

Strong normalisation: Every typed λ-term P is strongly normalizing.

Curry-Howard (1)

Minimal intuitionistic logic

Variable

$$\frac{}{A \vdash A}$$

Abstraction

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$$

Application

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

Weakening

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

Contraction

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

Permutation

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$$

Curry-Howard (1)

simply-typed λ-calculus

Variable

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

Abstraction

$$\frac{\Gamma, x : A \vdash P : B}{\Gamma \vdash \lambda x.P : A \Rightarrow B}$$

Application

$$\frac{\Gamma \vdash P : A \Rightarrow B \qquad \Delta \vdash Q : A}{\Gamma, \Delta \vdash PQ : B}$$

Weakening

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

Contraction

$$\frac{\Gamma, x : A, y : A \vdash P : B}{\Gamma, z : A \vdash P[x, y \leftarrow z] : B}$$

Permutation

$$\frac{\Gamma, x : A, y : B, \Delta \vdash P : C}{\Gamma, y : B, x : A, \Delta \vdash P : C}$$

Curry-Howard for second-order logic

Second order abstraction $$\frac{\Gamma, \alpha \ \vdash \quad A}{\Gamma \vdash \quad \forall \alpha . A}$$

Second order application $$\frac{\Gamma \ \vdash \quad \forall \alpha . A}{\Gamma \ \vdash \quad A[\alpha := B]}$$

Encoding the natural numbers

The type Nat of natural numbers is defined as:

$$Nat \quad = \quad \forall\alpha \quad . \quad (\alpha \Rightarrow \alpha) \quad \Rightarrow \quad (\alpha \Rightarrow \alpha)$$

Exercise: check that every Church numeral n is indeed of type Nat:

$$\vdash \quad \lambda f.\lambda a.\ \underbrace{f \cdots f}_{n \text{ times}}\ (a) \quad : \quad \forall\alpha.(\alpha \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha)$$

Encoding finite lists

The type $List$ of finite lists of elements of A is defined as:

$$List(A) \quad = \quad \forall\alpha \quad . \quad (\, A \Rightarrow \alpha \Rightarrow \alpha \,) \quad \Rightarrow \quad (\, \alpha \Rightarrow \alpha \,)$$

The list $[a_1, a_2, \cdots, a_n]$ is encoded as

$$[a_1, \cdots, a_n] \quad = \quad \lambda f.\lambda x.fa_1(fa_2(\cdots fa_nx)\cdots)$$

and the empty list is encoded as

$$nil \quad = \quad \lambda f.\lambda x.x$$

So, for instance:

$$[a_1, a_2] \quad = \quad \lambda f.\lambda x.fa_1(fa_2x)$$

Exercise: establish that the encoding of finite lists is of the expected type:

$$\vdash \quad [a_1, \cdots, a_n] \quad : \quad List(A)$$

Appending lists

The following λ-term appends two finite lists:

$$Append \quad = \quad \lambda list_1.\ \lambda list_2.\ \lambda f.\ \lambda x.\ list_1\, f\, (list_2\, f\, x)$$

Exercise: check that the λ-term Append has the expected type:

$$\vdash \quad Append \quad : \quad \forall\gamma \quad . \quad List(\gamma) \quad \Rightarrow \quad List(\gamma) \quad \Rightarrow \quad List(\gamma)$$

and the expected behaviour:

$$Append[a_1, \cdots, a_k][a_{k+1}, \cdots a_n] \quad \longrightarrow_\beta \quad \cdots \quad \longrightarrow_\beta \quad [a_1, \cdots, a_n]$$

Mapping lists

Given a λ-term h of type

$$h \quad : \quad A \;\Rightarrow\; B$$

the λ-term

$$Map \quad = \quad \lambda h.\ \lambda list.\ \lambda f.\ \lambda x.\ list\,(\lambda a.f(ha))\,x$$

transforms every list

$$[a_1, \cdots, a_n]$$

of elements of A into the list

$$[ha_1, \cdots, ha_n]$$

of elements of B.

Exercise: check that the λ-term Map has the expected type:

$$\vdash \quad Map \quad : \quad \forall\alpha\ .\ \forall\beta\ .\ \ (\alpha \;\Rightarrow\; \beta) \quad\Rightarrow\quad (List(\alpha) \;\Rightarrow\; List(\beta))$$

and the expected behaviour:

$$Map\ \ h\ \ [a_1, \cdots, a_n] \quad \longrightarrow_\beta \quad \cdots \quad \longrightarrow_\beta \quad [ha_1, \cdots, ha_n]$$

Encoding binary trees

The type $BinTree$ of binary trees with leaves of type A is defined as:

$$BinTree(A) \quad = \quad \forall\alpha \quad . \quad (\,\alpha \times \alpha \Rightarrow \alpha\,) \quad \Rightarrow \quad (\,A \Rightarrow \alpha\,)$$

Exercise: define the λ-term associated to a given binary tree, and construct a λ-term Flatten of type

$$\vdash \quad Flatten \quad : \quad \forall\alpha.\ BinTree(\alpha) \quad \Rightarrow \quad List(\alpha)$$

which flattens every binary tree into the list of its leaves, ordered from left to right.

Part II

An operational semantics

Terms and evaluation contexts

The terms

The usual **untyped** λ-calculus extended with **pairs** and **conditionals**.

e	$::=$	x	variable
	$\mid$	$\lambda x.e$	abstraction
	$\mid$	$e\, e$	application
	$\mid$	(e, e)	pair
	$\mid$	$\texttt{fst}(e)$	first projection
	$\mid$	$\texttt{snd}(e)$	second projection
	$\mid$	$\texttt{if}\ e\ \texttt{then}\ e\ \texttt{else}\ e$	conditional
	$\mid$	$\texttt{true}$	constant true
	$\mid$	$\texttt{false}$	constant false

The evaluation contexts

Evaluation contexts are **stacks** or **finite lists** of operations:

E	$::=$	$\mathtt{nil}$	empty context
	$\mid$	$e \cdot \mathrm{E}$	application
	$\mid$	$\mathtt{fst} \cdot \mathrm{E}$	first projection
	$\mid$	$\mathtt{snd} \cdot \mathrm{E}$	second projection
	$\mid$	$(\mathtt{if}\ e, e) \cdot \mathrm{E}$	conditional

The evaluation bracket

Every term e and context E combine as a term

$$\langle e \mid \mathrm{E} \rangle$$

defined just in the usual way:

$$
\begin{array}{lcl}
\langle e \mid \mathtt{nil} \rangle & = & e \\
\langle e \mid e' \cdot \mathrm{E} \rangle & = & \langle e\, e' \mid \mathrm{E} \rangle \\
\langle e \mid \mathtt{fst} \cdot \mathrm{E} \rangle & = & \langle \mathtt{fst}(e) \mid \mathrm{E} \rangle \\
\langle e \mid \mathtt{snd} \cdot \mathrm{E} \rangle & = & \langle \mathtt{snd}(e) \mid \mathrm{E} \rangle \\
\langle e \mid (\mathtt{if}\ e_1, e_2) \cdot \mathrm{E} \rangle & = & \langle \mathtt{if}\ e\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 \mid \mathrm{E} \rangle
\end{array}
$$

The dynamics

Five rewriting rules:

∘ the β-rule:

$$\langle \lambda x.e \mid e' \cdot \mathrm{E} \rangle \rightarrow \langle e[x := e'] \mid \mathrm{E} \rangle$$

∘ the two projection rules for the pair:

$$\begin{array}{rcl} \langle (e_1, e_2) \mid \mathtt{fst} \cdot \mathrm{E} \rangle & \rightarrow & \langle e_1 \mid \mathrm{E} \rangle \\ \langle (e_1, e_2) \mid \mathtt{snd} \cdot \mathrm{E} \rangle & \rightarrow & \langle e_2 \mid \mathrm{E} \rangle \end{array}$$

∘ the two rules for the conditional:

$$\begin{array}{rcl} \langle \mathtt{true} \mid (\mathtt{if}\ e_1, e_2) \cdot \mathrm{E} \rangle & \rightarrow & \langle e_1 \mid \mathrm{E} \rangle \\ \langle \mathtt{false} \mid (\mathtt{if}\ e_1, e_2) \cdot \mathrm{E} \rangle & \rightarrow & \langle e_2 \mid \mathrm{E} \rangle \end{array}$$

Sums

Possible to extend the language of **terms** with three operators

$$\texttt{inl}(e) \qquad \texttt{inr}(e) \qquad \texttt{caseof}(e, e_1, e_2)$$

the language of **contexts** with one operator

$$(\texttt{case}\ e_1, e_2) \cdot \mathrm{E}$$

Then, add the equation:

$$\langle e \mid (\texttt{case}\ e_1, e_2) \cdot \mathrm{E} \rangle \ = \ \langle \texttt{caseof}(e, e_1, e_2) \mid \mathrm{E} \rangle$$

and the two rewriting rules:

$$\langle \texttt{inl}(e) \mid (\texttt{case}\ e_1, e_2) \cdot \mathrm{E} \rangle \ \longrightarrow \ \langle e_1\, e \mid \mathrm{E} \rangle$$
$$\langle \texttt{inr}(e) \mid (\texttt{case}\ e_1, e_2) \cdot \mathrm{E} \rangle \ \longrightarrow \ \langle e_2\, e \mid \mathrm{E} \rangle$$

Part III

A topology

Well-typed terms cannot go wrong

Safe terms

A term e is **safe** when it loops, or when it reduces to the boolean constant `true` or to the boolean constant `false`.

$(\lambda x.xx)(\lambda x.xx)$ `true` $(\lambda x.x)$`true`

A term is **unsafe** when it is not safe.

`fst(true)` (syntax error!)

Safe terms

More generally, we may fix a set of safe terms

$$\perp\!\!\!\perp$$

with the single requirement that $\perp\!\!\!\perp$ is closed under **reverse reduction**:

for all e_1, e_2 $\quad e_1 \rightarrow e_2$ and $e_2 \in \perp\!\!\!\perp$

implies

$$e_1 \in \perp\!\!\!\perp$$

Orthogonality

$$e \perp E$$

means that

the term $\langle e \mid E \rangle$ is **safe**

Orthogonality

Given a set S of evaluation contexts,

$$e \perp S$$

means that

the term $\langle e \mid \mathrm{E} \rangle$ is safe for every context $\mathrm{E} \in S$

Variety

The terms in the set

$$S^{\perp} \quad = \quad \left\{ e \mid e \perp S \right\}$$

are the terms which **combine safely** with any element of S.

We define:

A **variety** is a set of terms of the form $X = S^{\perp}$

Varieties = closed sets of terms

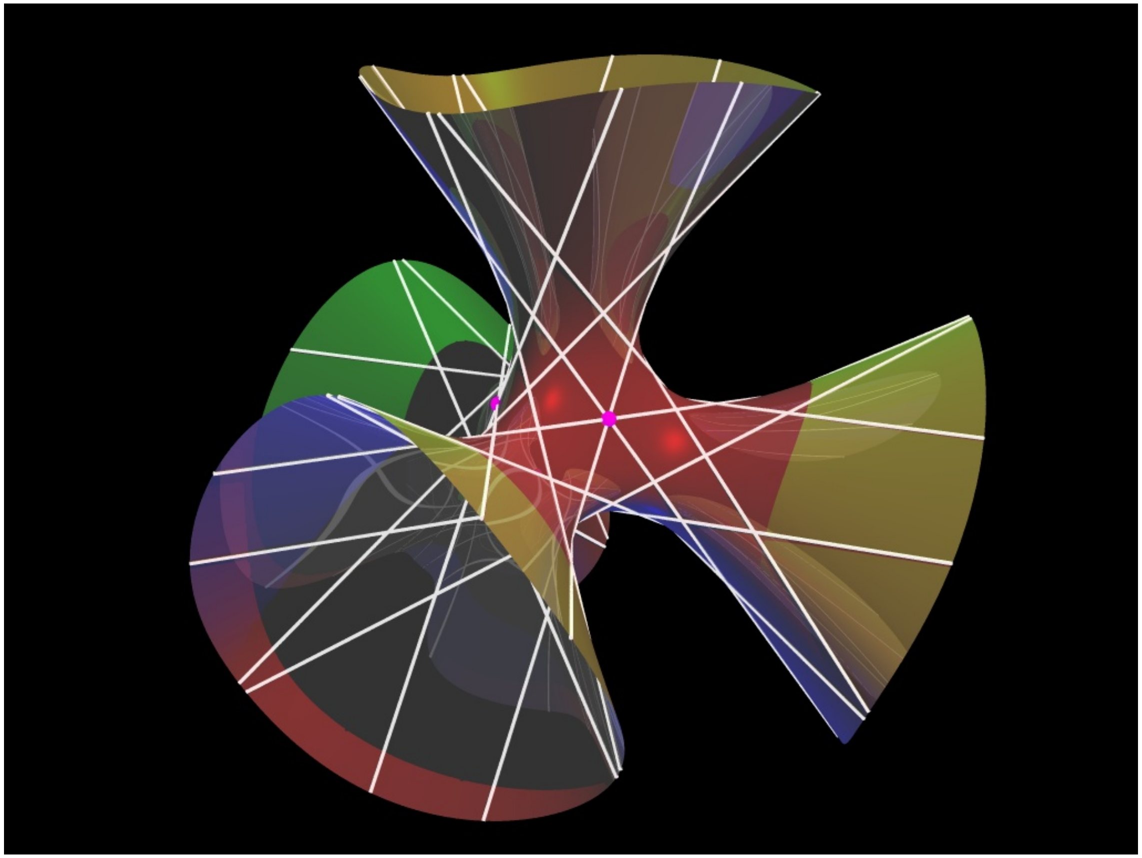
The set of varieties is closed under **arbitrary intersection**, and thus defines a **closure operator** — computed by **biorthogonality**:

$$X \mapsto X^{\perp\perp}$$

A **variety** is a set of terms of the form $X = X^{\perp\perp}$

Algebraic geometry as guideline

λ-terms	$\sim$	points
evaluation contexts	$\sim$	polynomials
$\langle x \mid P \rangle$	$\sim$	$P(x)$
safe computation	$\sim$	equality to zero
$S^{\perp}$	$\sim$	$V(S)$



Part IV

Varieties as semantic types

A modern view on “candidats de réducibilité”

Arrow type

Given two sets X and Y of terms, define

$$X \Rightarrow Y$$

as the set of terms f satisfying:

$$\forall e \in X, \qquad f e \in Y.$$

Fact: $X \Rightarrow Y$ is a variety when Y is a variety.

Arrow type

$X \Rightarrow Y$ is a variety when Y is a variety $Y = S^{\perp}$

Proof. $$f \in X \Rightarrow Y$$

$\iff$ $\forall e \in X,\ \forall \mathrm{E} \in S,$ the term $\langle fe \mid \mathrm{E} \rangle$ is safe.

$\iff$ $\forall e \in X,\ \forall \mathrm{E} \in S,$ the term $\langle f \mid e \cdot \mathrm{E} \rangle$ is safe.

$\iff$ $$f \ \perp \ X \cdot S$$

Product types

Given two sets X and Y of terms, define

$$X \times Y$$

as the set of terms f which **loop** or **reduce** to a pair

$$(e_1, e_2)$$

in which $e_1 \in X$ and $e_2 \in Y$.

Fact: $X \times Y$ is a variety when X and Y are varieties.

Subtyping

$$X \subseteq Y$$

Intersection and union types

$$X \wedge Y \quad = \quad X \cap Y$$

$$X \vee Y \quad = \quad (X \cup Y)^{\perp\perp}$$

Note that we need to close the union here!

Universal and existential polymorphism

Given a family $(X_\alpha)_{\alpha \in V}$ of varieties indexed by α running on a set W,

$$\forall \alpha . X_\alpha \quad = \quad \bigcap_{\alpha \in W} X_\alpha$$

$$\exists \alpha . X_\alpha \quad = \quad (\bigcup_{\alpha \in W} X_\alpha \;)^{\perp\perp}$$

Note that we need to close the union here!

Algebraic geometry as guideline

The **Zariski topology** is defined as the set of varieties $V(S)$.

The union $X \cup Y$ of two Zariski varieties X and Y is a variety because the **product** of polynomials behaves like the **parallel or**:

$$\langle x \mid PQ \rangle = 0 \quad \Longleftrightarrow \quad \langle x \mid P \rangle = 0 \quad \text{or} \quad \langle x \mid Q \rangle = 0.$$

Algebraic geometry as guideline

$X \cup Y$ is closed if and only if, for every term e,

$$e \in (X \cup Y)^{\perp\perp} \quad \Rightarrow \quad e \in X \cup Y,$$

or equivalently,

$$e \perp X^{\perp} \cap Y^{\perp} \quad \Rightarrow \quad e \in X \cup Y,$$

This is true when $X^{\perp} \cap Y^{\perp}$ contains all the evaluation contexts $\mathrm{E} \circledvee \mathrm{E}'$.

$$\langle e \mid \mathrm{E} \circledvee \mathrm{E}' \rangle \rightarrow \langle e \mid \mathrm{E} \rangle \circledvee \langle e \mid \mathrm{E}' \rangle$$

An angelic interpretation of choice

Part V

The main theorem

Towards a semantic proof of normalization

VAR-ACCESS

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}$$

APP

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\, e_2 : \tau_1}$$

ABS

$$\frac{\Gamma, x : \tau_2 \vdash e : \tau_1}{\Gamma \vdash \lambda x.e : \tau_2 \to \tau_1}$$

PAIR

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2}$$

FST

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \mathtt{fst}(e) : \tau_1}$$

SND

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \mathtt{snd}(e) : \tau_2}$$

CONSTANT TRUE

$$\Gamma \vdash \mathtt{true} : \mathit{Bool}$$

CONSTANT FALSE

$$\Gamma \vdash \mathtt{false} : \mathit{Bool}$$

CONDITIONAL

$$\frac{\Gamma \vdash e_1 : \mathit{Bool} \quad \Gamma \vdash e_2 : \tau \qquad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \mathtt{if}\ e_1\ \mathtt{then}\ e_2\ \mathtt{else}\ e_3 : \tau}$$

FIXPOINT

$$\frac{\Gamma \vdash e : \tau \to \tau}{\Gamma \vdash Y e : \tau}$$

ALL-INTRO

$$\frac{\Gamma, \alpha \vdash e : \tau}{\Gamma \vdash e : \forall \alpha.\tau}$$

ALL-ELIM

$$\frac{\Gamma \vdash e : \forall \alpha.\tau}{\Gamma \vdash e : \tau[\tau'/\alpha]}$$

EXISTS-INTRO

$$\frac{\Gamma \vdash e : \tau[\tau'/\alpha]}{\Gamma \vdash e : \exists \alpha.\tau}$$

EXISTS-ELIM

$$\frac{\Gamma \vdash e : \exists \alpha.\tau' \quad \Gamma, \alpha, x : \tau' \vdash \langle x \mid \mathrm{E} \rangle : \tau \quad \alpha \notin FV(\tau) \qquad x \notin FV(\mathrm{E})}{\Gamma \vdash \langle e \mid \mathrm{E} \rangle : \tau}$$

SUB

$$\frac{\Gamma \vdash e : \tau' \qquad \tau' <: \tau}{\Gamma \vdash e : \tau}$$

Fundamental theorem

Suppose that the term e is typed by the sequent:

$$\vdash \quad e \quad : \quad A$$

Then,

$$e \quad \in \quad [\![A]\!]$$

where the variety $[\![A]\!]$ is the interpretation of the type A.

Application: a weak normalization theorem

This works for any choice of notion of safety.

In particular, suppose that

$$\perp\!\!\!\perp$$

denotes the set of weakly normalizing terms.

In that case, the variety $[\![A]\!]$ contains only weakly normalizing terms.

Consequence: every typed λ-term is weakly normalizing.