Master Parisien de Recherche en Informatique

Modèle des langages de programmation

Domaines, Catégories, Jeux

(Cours 2.2)

Paul-André Melliès

www.irif.fr/~mellies/mpri.html

Modèles des langages de programmation

Domaines, catégories, jeux

Introduction au cours

Calendar 2020

September 15	Paul-André Melliès	Categories, Linear Logic
September 20	Paul-André Melliès	& Scott domains, Coherence Spaces
September 27	Paul-André Melliès	
October 6	Paul-André Melliès	
October 13	Paul-André Melliès	
October 20	Paul-André Melliès	
October 27	Paul-André Melliès	
November 3	Paul-André Melliès	
November 10	Thomas Ehrhard	Quantitative Semantics
November 17	Thomas Ehrhard	& Probabilistic Lambda-Calculus
November 24	Examination period	
December 1	Examination period	
December 8	Thomas Ehrhard	
December 15	Thomas Ehrhard	
December 22	Thomas Ehrhard	

Calendar 2021

January 5	Thomas Ehrhard	
January 12	Thomas Ehrhard	
January 19	Michele Pagani	Quantitative Semantics
January 26	Michele Pagani	& Abstract Machines
February 2	Michele Pagani	
February 9	Michele Pagani	
February 16	Michele Pagani	
February 23	Michele Pagani	
March 2	Examination period	
March 9	Examination period	

Semantics

A **mathematical** investigation of programming languages and of their compilation schemes.

Functional or **imperative** languages based on a kernel of λ-calculus:

PCF	Algol	ML	OCAML	JAVA
λ-calculus	states	exceptions	modules	concurrence
higher order		references	objets	synchronisation
typing				threads
recursion				

Semantics

Aim — a **mathematical theory** of programming languages
from **design** to **compilation** in machine code.

$$\boxed{\begin{array}{c} \text{High level program} \\ \Big\Downarrow \ \text{compilation} \\ \text{Low level code} \end{array}}$$

Required to certify the **preservation of meaning** during compilation

Syntax or semantics ?

One should grasp a programming language from its two sides:

- the syntactic manipulations
- the **meaning** of these syntactic manipulations

In the same way, in algebraic topology, one constructs spaces

- as higher-dimensional triangulations (syntax)
- then computes their homology groups (semantics)

$$\begin{tikzcd} \cdot \arrow[r, "b"] & \cdot \\ \cdot \arrow[u, "a"] \arrow[r, "b"'] & \cdot \arrow[u, "a"'] \end{tikzcd} \qquad \rightsquigarrow \qquad \text{a torus}$$

... this leading to a geometry of **language** and **reasoning** !

Connections with contemporary algebra

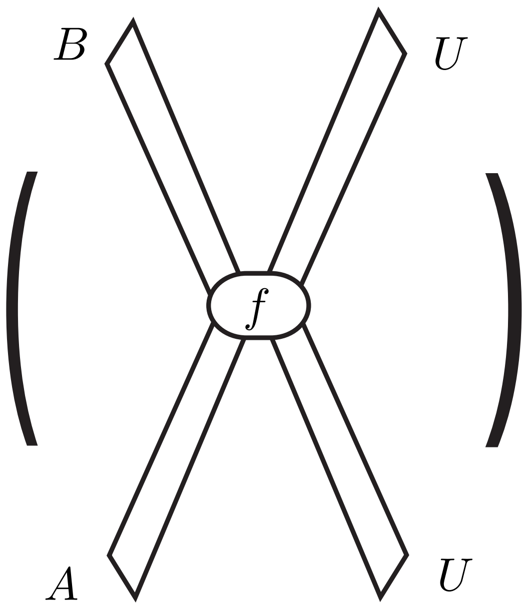
A **trace** in a « monoidal » category $\mathcal{C}$ is an operator

$$\mathrm{Tr}^{U}_{A,B} \qquad \frac{A \otimes U \longrightarrow B \otimes U}{A \longrightarrow B}$$

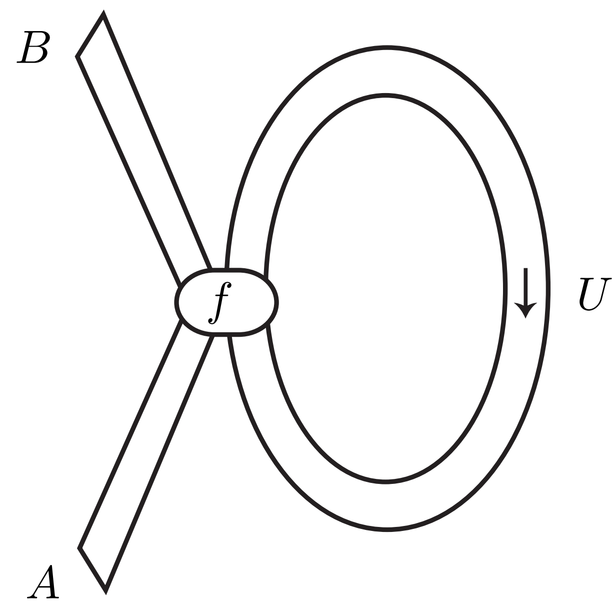
depicted as a « **feedback** » in string diagrams:

Trace operators

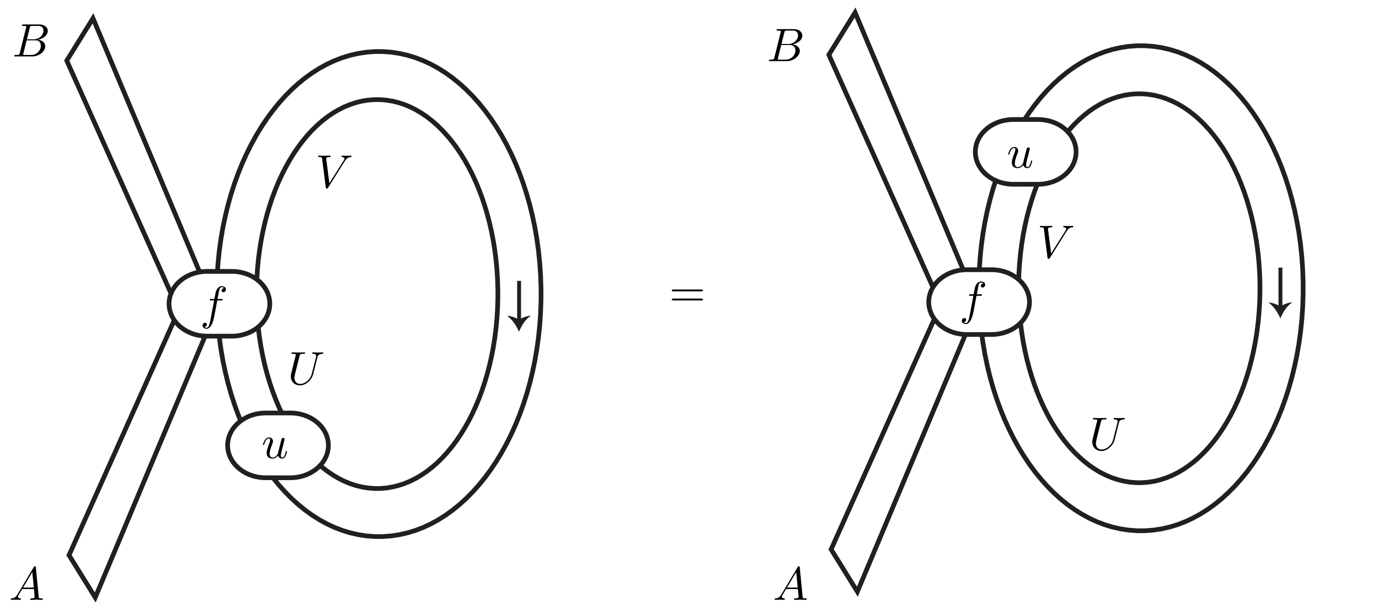
$\mathrm{Tr}^{U}_{A,B}$



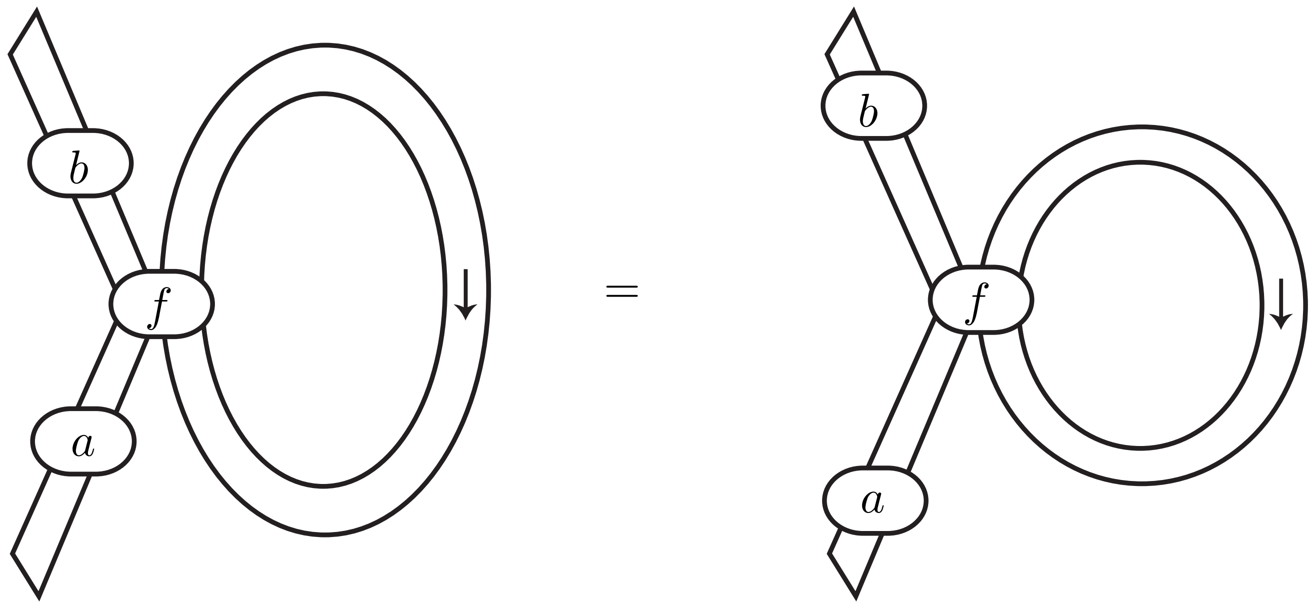
$=$



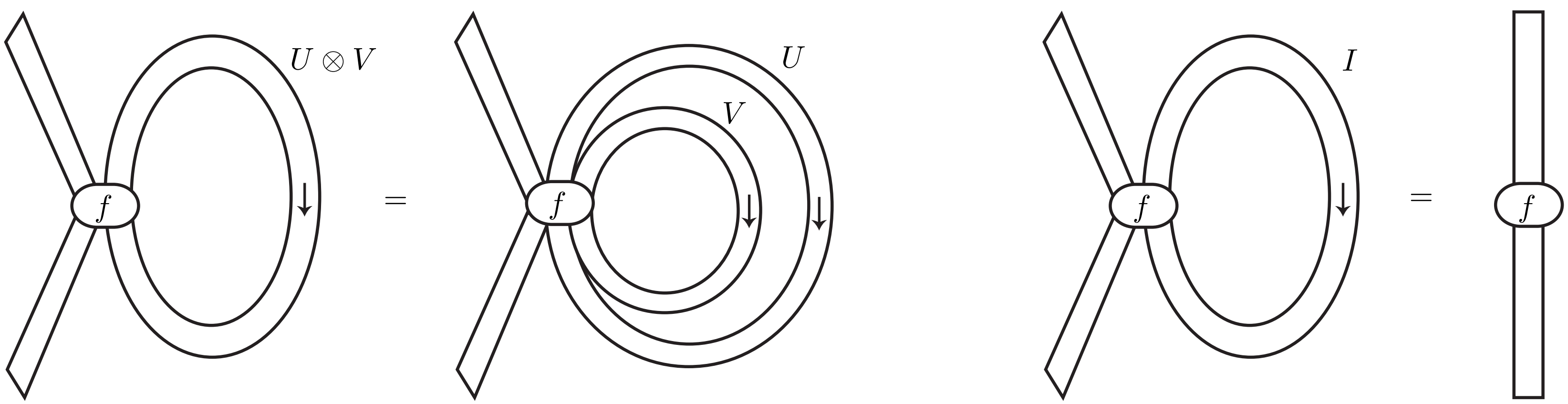
Sliding



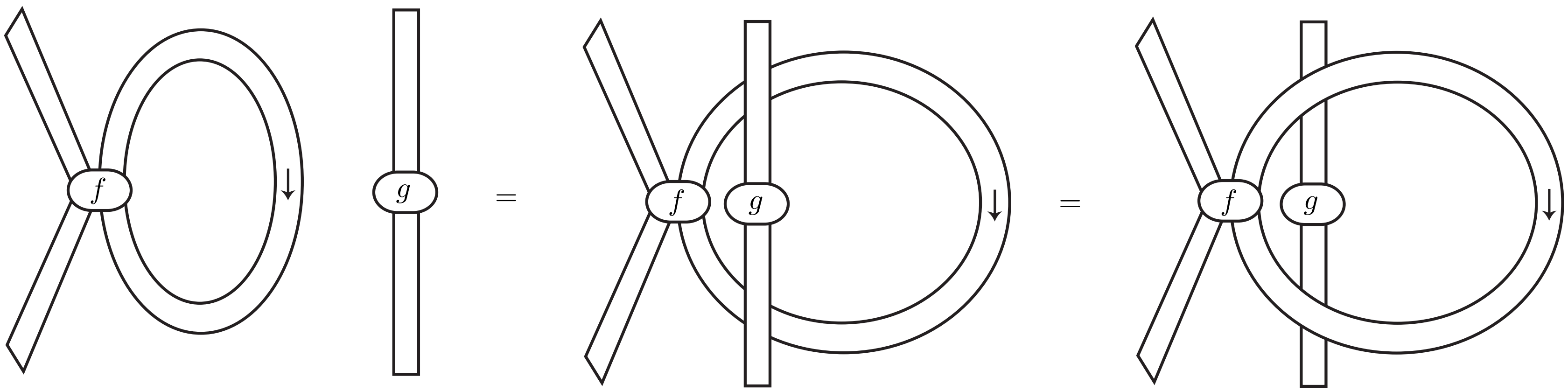
Tightening



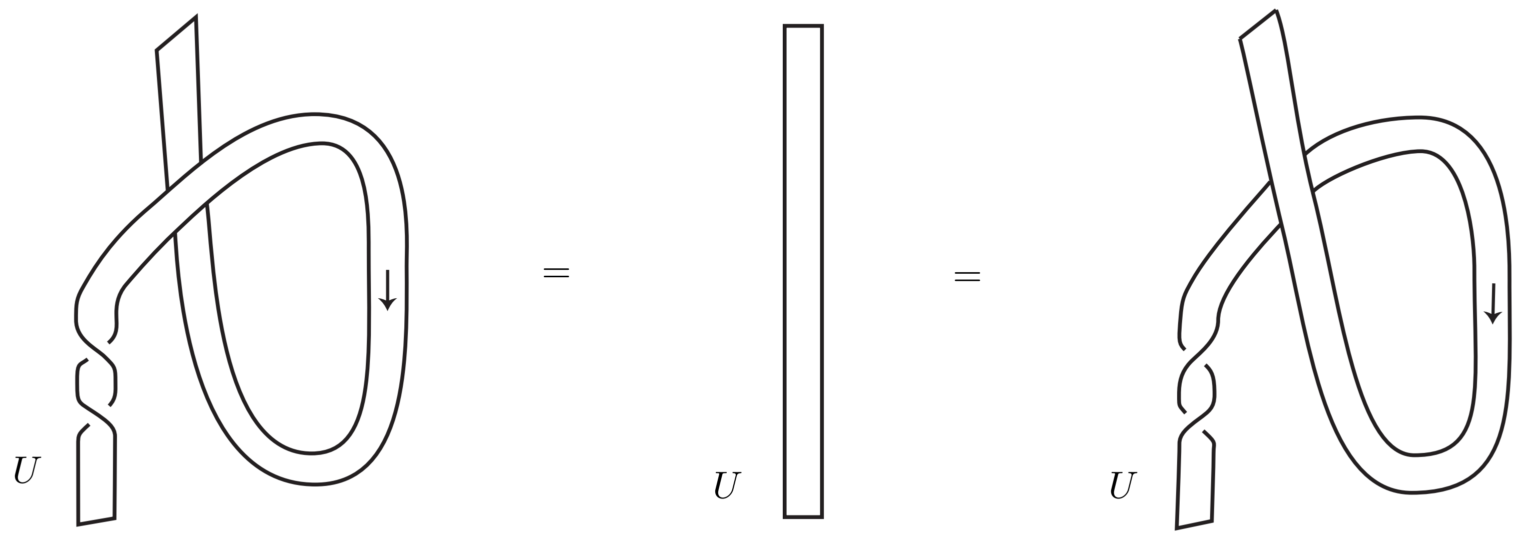
Vanishing



Superposition



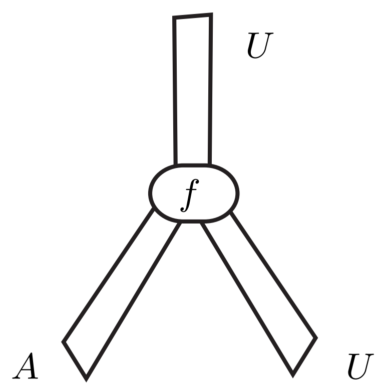
Yanking



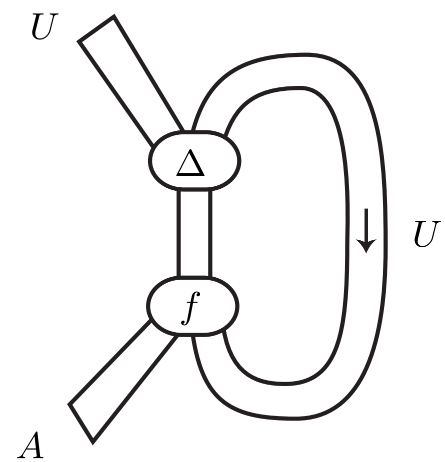
Feedback = recursion

In a « cartesian » category:

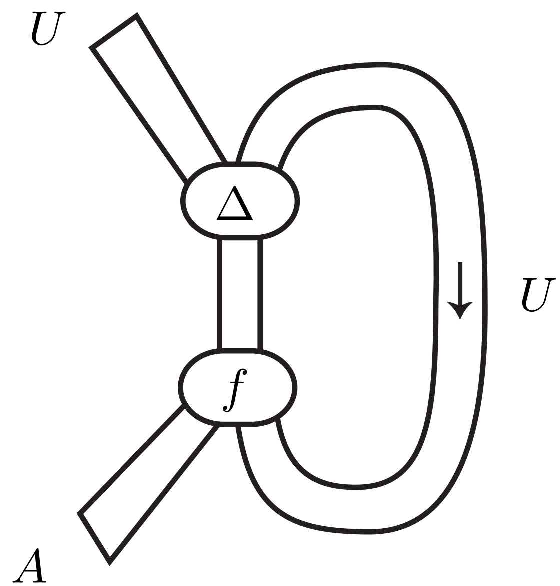
Fix :



$\mapsto$

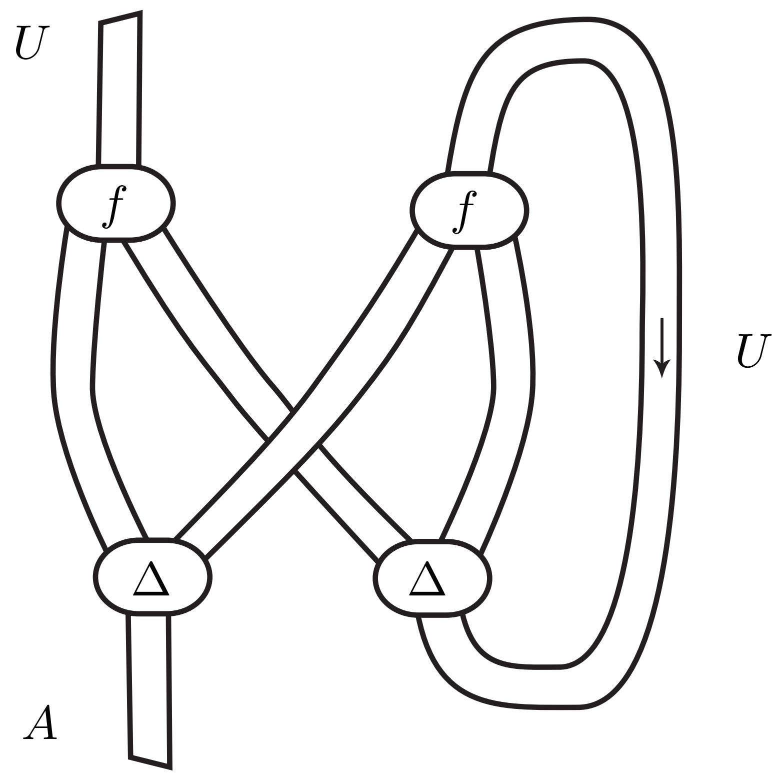


Recursion derived from feedback

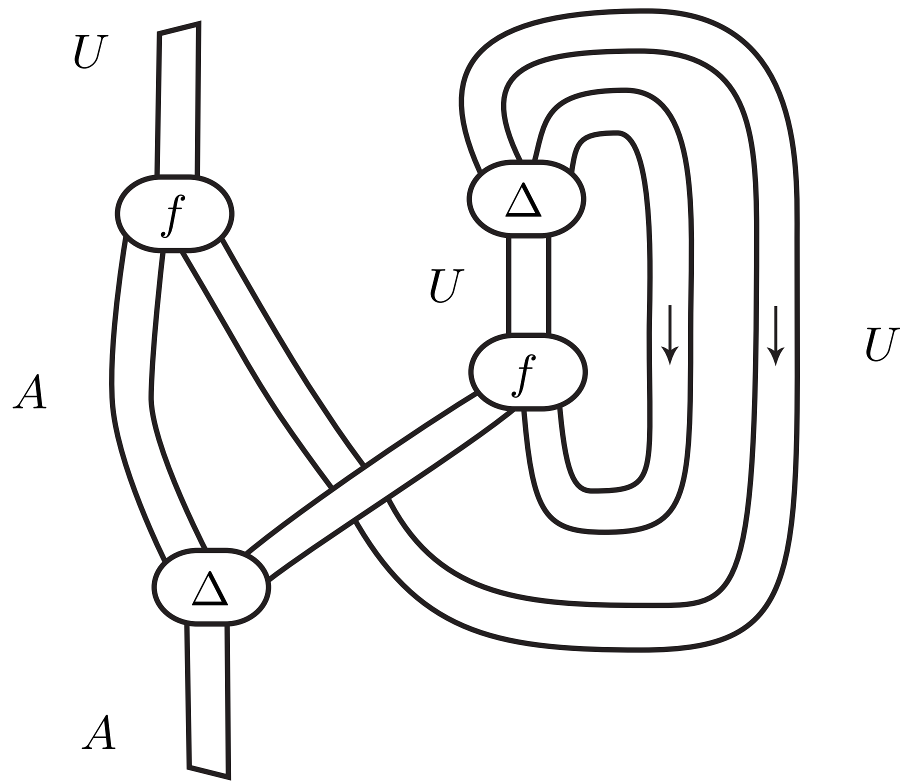


$Fix(f)(a)$

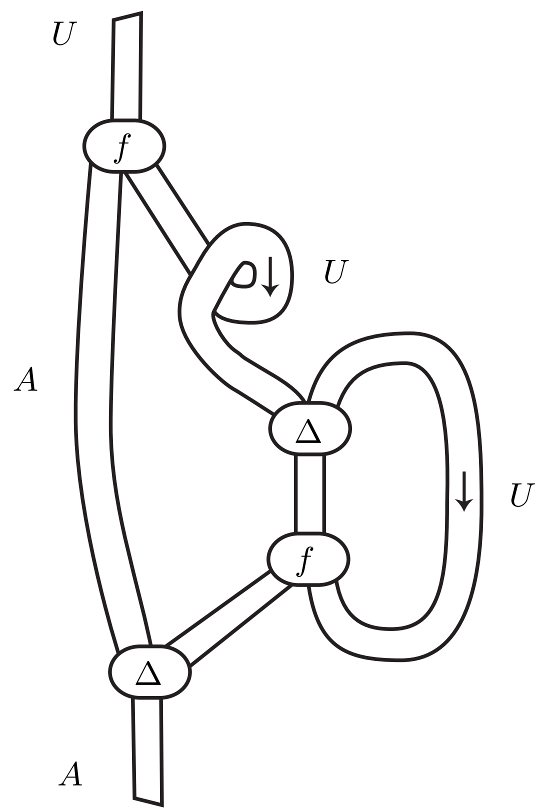
Recursion derived from feedback



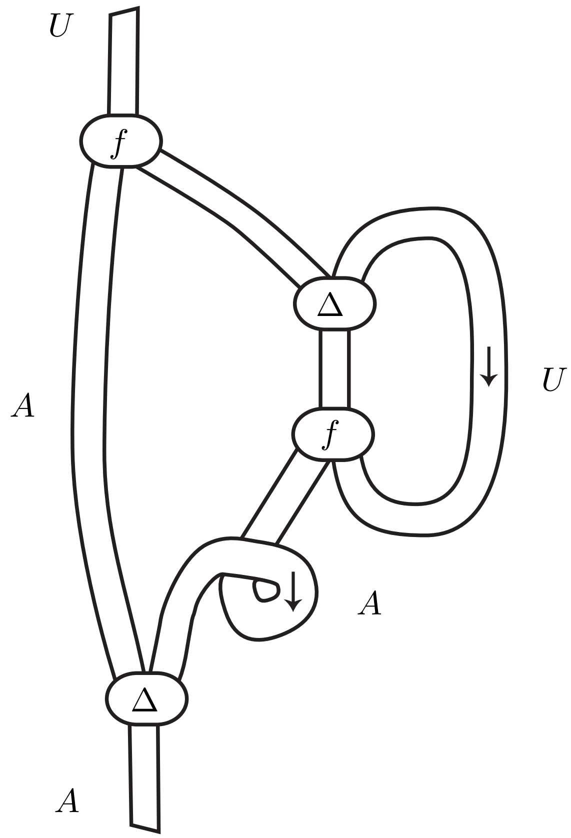
Recursion derived from feedback



Recursion derived from feedback

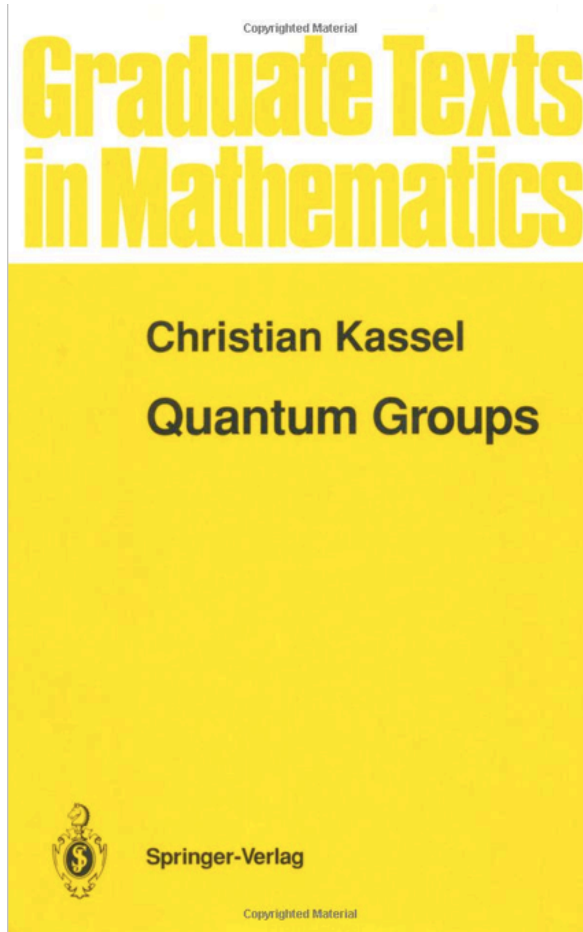


Recursion derived from feedback



$$Fix(f)(a) \quad = \quad f(a, Fix(f)(a))$$

A bridge between programming & knot theory



Syntax

Church 1935 — syntactic invention of the λ-calculus

The λ**-calculus** is a **formal** calculus of functions.

The expressions of the λ-calculus are called λ**-terms.**

In the λ-calculus, every λ-term P is at the same time:

▷ a **function** which takes every λ-term as argument,

▷ an **argument** of any other λ-term, including itself.

For a very long time, people believed that the λ-calculus was **a purely formal system** without mathematical meaning — until Dana Scott's discovery of a denotational semantics in 1969.

Semantics

Domain semantics

Key idea: The semantics of a program :

$$P \quad : \quad A \quad \longrightarrow \quad B$$

is a **function** :

$$[P] \quad : \quad [A] \quad \longrightarrow \quad [B]$$

from the **domain** of inputs A to the **domain** of outputs B.

Definition: a domain is an ordered set with filtered limits.

This defines a **category** where the interpretation is **compositional**:

$$[A] \xrightarrow{[P]} [B] \xrightarrow{[Q]} [C] \quad = \quad [A] \xrightarrow{[P;Q]} [C]$$

Game semantics

Key idea: A program

$$P \quad : \quad A \quad \longrightarrow \quad B$$

is interpreted as an interactive **strategy** :

$$[P] \quad : \quad [A] \quad \multimap \quad [B]$$

which plays on the **input game** A and the **output game** B.

New fact: the meaning of a program is an automaton !

Game semantics = **idealized** and **compositional** compilation

Game semantics

The **evaluation** of a program P against its environment E

$$\text{Program} \qquad \longleftrightarrow \qquad \text{Environment}$$

may be understood as the interactive exploration/ of the program P by its environment E, and conversely, of E by P.

$\boxed{\text{Evaluation is an interactive form of } \textbf{pattern matching}.}$

Categories

Categories

A category $\mathcal{C}$ is given by

[0] a class of **objects**

[1] a set $\mathbf{Hom}(A, B)$ of **morphisms**

$$f \quad : \quad A \quad \longrightarrow \quad B$$

for every pair of objects (A, B)

[2] a **composition law**

$$\circ \quad : \quad \mathbf{Hom}(B, C) \times \mathbf{Hom}(A, B) \quad \longrightarrow \quad \mathbf{Hom}(A, C)$$

[2] an **identity** morphism

$$id_A \quad : \quad A \quad \longrightarrow \quad A$$

for every object A,

Categories

satisfying the following properties:

[3] the composition law $\circ$ is associative:

$$\begin{array}{l} \forall f \in \mathbf{Hom}(A,B) \\ \forall g \in \mathbf{Hom}(B,C) \\ \forall h \in \mathbf{Hom}(C,D) \end{array} \qquad f \circ (g \circ h) = (f \circ g) \circ h$$

[3] the morphisms id are neutral elements

$$\forall f \in \mathbf{Hom}(A,B) \qquad f \circ id_A \;=\; f \;=\; id_B \circ f$$

Examples

1. the category **Ens** of sets and functions
2. the category **Pord** of partial orders and monotone functions
3. the category **Dom** of domains and continuous functions
4. the category **Coh** of coherence spaces and linear maps
5. every partial order
6. every monoid

Linear logic

Intuitionistic logic in 1985

A converging constellation of ideas:

Cartesian closed
categories
(algebra)

λ-calculus and
natural deduction
(geometry)

Scott
models
(static)

$\boxed{\text{Cut elimination}}$

Sequent
calculus
(syntax)

Concrete
data structures
(dynamic)

Krivine machine
and CAM
(compilation)

La secrète noirceur du lait: linear decomposition of the λ-calculus

Church numerals

$$\mathbf{0} = (\lambda f.\lambda x.x) \qquad \mathbf{1} = (\lambda f.\lambda x.fx) \qquad \mathbf{2} = (\lambda f.\lambda x.f(f(x)))$$

In the case of the numeral $\mathbf{1}$, a series of **atomic** computations which simply reorganize the term:

$$\mathbf{1}PQ \quad \longrightarrow \quad (\lambda x.Px)Q \quad \longrightarrow \quad PQ$$

In the case of $\mathbf{0}$ and $\mathbf{2}$, a series of **molecular** computations

$$\mathbf{0}PQ \xrightarrow{*} (\lambda x.x)Q \longrightarrow Q \qquad \mathbf{2}PQ \xrightarrow{*} (\lambda x.P(Px))Q \longrightarrow P(PQ)$$

where $\xrightarrow{*}$ **erases** or **duplicates** the argument.

However, the rewrite $\xrightarrow{*}$ is only one step of β-reduction.

Linear decomposition of the λ-calculus

Main idea – decompose the duplication and erasing mechanisms of the λ-calculus.

What for?

Extract the syntactic **atoms** of the existing **molecules**, and build something like a **Mendeleiev table**.

Ambitious, but it works wonderfully well !

The decomposition of LJ and LK in LL:

$$A \Rightarrow B \quad = \quad !A \multimap B$$

Linear logic

Linear logic offers a **global** reunderstanding of LJ and LK.

Monoidal
categories
(algebra)

Proof
nets
(geometry)

Denotational
semantics
(static)

$\boxed{\text{Cut elimination}}$

Sequent
calculus
(syntax)

Game
semantics
(dynamic)

Abstract
machines
(compilation)

Benefits of linear logic

1. simplified the models of the λ-calculus, both

   - **static** $\longrightarrow$ domains seen as coherence spaces
   - **dynamic** $\longrightarrow$ CDSs seen as dialogue games

2. shed light on **optimal implementations** of the λ-calculus (Lévy) and opened the study of **polynomial time typing,**

3. revealed the **broken symmetry** in intuitionistic logic between Player (Program) and Opponent (Environment).

Semantics today

- ▷ semantics of **low level languages**
- ▷ a mathematical theory of **concurrency**
- ▷ **resource allocation**, **side effects** and **complexity**
- ▷ a **new generation** of languages and proof assistants
- ▷ realizability models of **Zermelo-Frankel set theory**
- ▷ **homotopy type theory**
- ▷ emerging connections to **knot theory** and **physics**