About categorical semantics

Dominique Duval

LJK, University of Grenoble

October 15., 2010
Capp Café, LIG, University of Grenoble

Outline

The issue

Semantics of programming languages

- several paradigms (functional, imperative, object-oriented,...)
- several kinds of semantics (denotational, operational,...)

more precisely

- effects (states, exceptions, ...)

still more precisely in this talk

- states in imperative programming

With Jean-Claude Reynaud, Jean-Guillaume Dumas,
Christian Lair, César Domínguez, Laurent Fousse

(Something else:
graph rewriting, with Rachid Echahed and Frédéric Prost)

The approach

We use category theory (rather than "usual" logic).

- 1940's Eilenberg and Mac Lane: categories, functors, ...
- 1950's Kan: adjunction
- 1960's Ehresmann: sketches
- 1960's Lawvere: adjunction in logic
- 1970's Lambek: the Curry-Howard-Lambek correspondence

Categorical semantics, for functional languages

Curry-Howard-Lambek correspondence:

logic	programming	categories
propositions	*types*	*objects*
proofs	*terms*	*morphisms*
intuitionistic logic	simply typed lambda calculus	cartesian closed categories
$\frac{A \quad A \Rightarrow B}{B}$	$\frac{a{:}A \quad \lambda x.t{:}A \to B}{(\lambda x.t)\ a{:}B}$	$\frac{a{:}U \to A \quad f{:}A \to B}{f \circ a{:}U \to B}$

Categories

A category $\mathbf{C}$ is made of

- objects $X, Y, \ldots$
- morphisms $f : X \to Y, \ldots$

with

- identities $\mathtt{id}_X : X \to X$
- composition $g \circ f : X \to Z$ for every $f : X \to Y, g : Y \to Z$

such that $\circ$ is associative and $\mathtt{id}$'s are units for $\circ$

- A category with at most one $X \to Y$ for each X, Y is a preorder
- A category with one object X is a monoid

Functors

A functor $F : \mathbf{C} \to \mathbf{D}$ is a homomorphism of categories

Examples

Mon $\to$ **Set**: a monoid $(M, \times, e) \mapsto$ the underlying set M

Set $\to$ **Mon**: a set $A \mapsto$ the monoid of words $(A^*, ., \varepsilon)$

A category of logics

The study of computational effects led us to the question:

in categorical terms

- what is "a logic"?
- what is "a homomorphism of logics"?

i.e.: what is "the" category of logics?

We have built "a" category of logics:
the category of diagrammatic logics

Outline

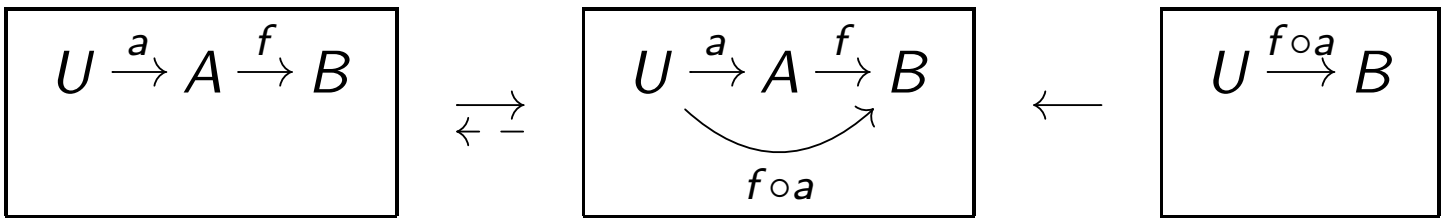
Modus ponens vs. composition

Modus ponens

$$\frac{A \quad A \Rightarrow B}{B}$$

Composition rule

$$\frac{a : U \to A \quad f : A \to B}{f \circ a : U \to B}$$



Deduction rules

$$\frac{H}{C} \quad \text{seen as} \quad \boxed{H} \longrightarrow \boxed{H \cup C} \longleftarrow \boxed{C}$$

where H, C, $H \cup C$ are specifications,
i.e., presentations of theories $L(H)$, $L(C)$, $L(H \cup C)$

Specifications:

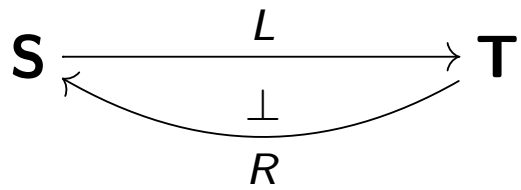
$$\boxed{H} \xrightarrow{(homo)} \boxed{H \cup C} \longleftarrow \boxed{C}$$

Theories:

$$\boxed{L(H)} \xrightarrow[\longleftarrow]{(iso)} \boxed{L(H \cup C)} \longleftarrow \boxed{L(C)}$$

Diagrammatic logics

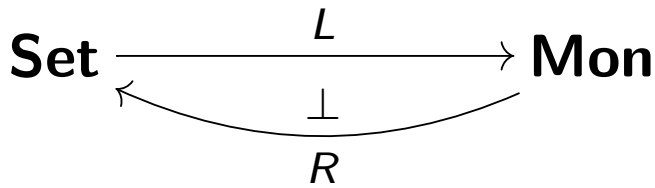
Definition. A logic is an adjunction



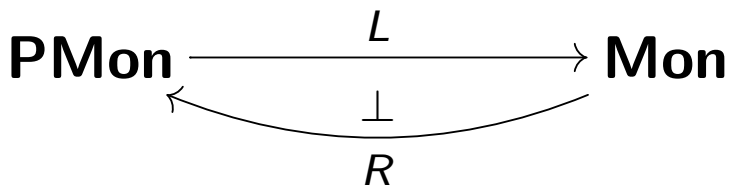
with R full and faithful
i.e., with $L \circ R \cong \mathtt{id}_{\mathbf{T}}$
i.e., with L a localizer [Gabriel-Zisman1967]

(and this comes from a morphism of limit sketches [Ehresmann1968])

Non-example

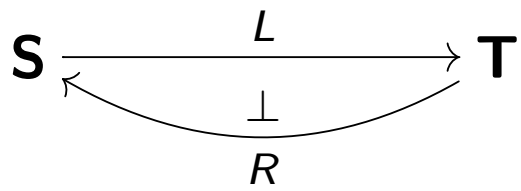


Example



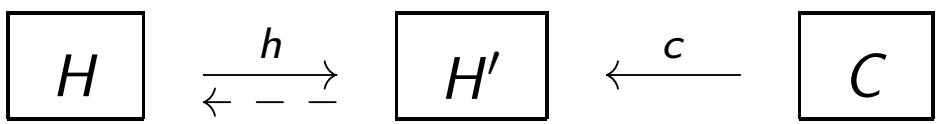
Specifications and theories

With respect to a logic



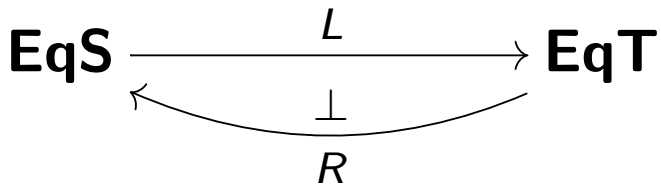
- **S**: category of specifications
- **T**: category of theories

Every morphism in **T** comes from some L-fraction $(\frac{c}{h}...)$



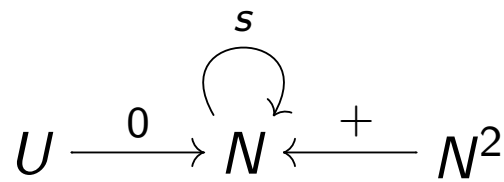
So, a logic corresponds to a family of deduction rules

Equational logic



EqS: cat. of equational specifications
EqT: cat. of equational theories

Example (U stands for `unit` or `void`)

A specification Σ_{nat}: 

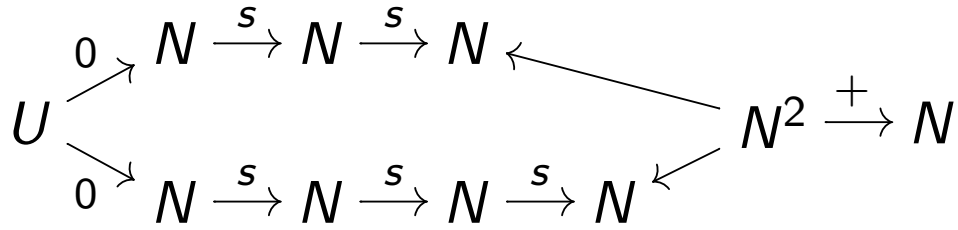
0+y=y
s(x)+y=s(x+y)

Two theories $L(\Sigma_{\mathrm{nat}})$ and Θ_{set}

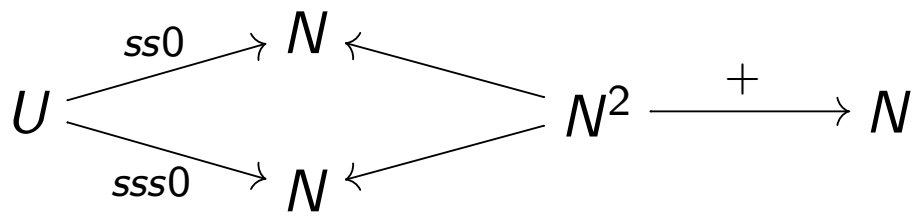
Terms as morphisms

A term for the equational specification Σ_{nat}:

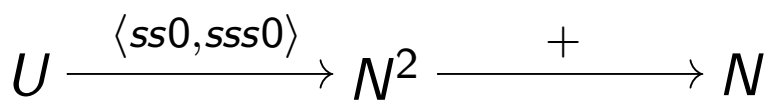
$ss0 + sss0$, closed term of type N



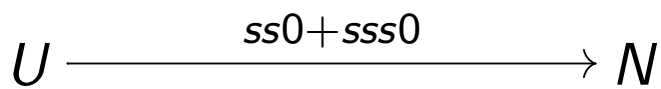
composition rule



pairing rule

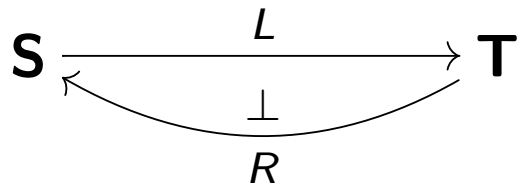


composition rule



Models

With respect to a logic

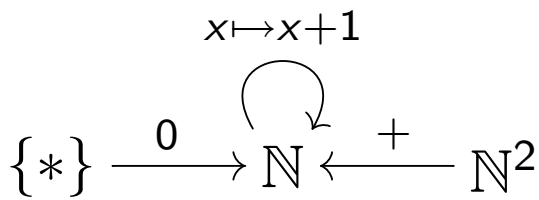


given a specification Σ and a theory Θ,
a model of Σ in Θ is (equivalently, by adjunction)

$$\Sigma \xrightarrow{M} R(\Theta) \quad \text{in } \mathbf{S} \quad \text{or} \quad L(\Sigma) \xrightarrow{M} \Theta \quad \text{in } \mathbf{T}$$

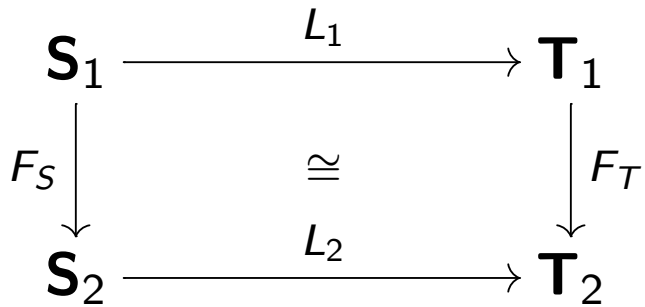
Example (equational logic)

a model of Σ_{nat} in Θ_{set}:



Homomorphisms of logics

Definition. A homomorphism of logics $F : L_1 \longrightarrow L_2$ is a pair of left adjoints (F_S, F_T) such that



(and this comes from a commutative square of morphisms of limit sketches)

So, we get the category of diagrammatic logics

Outline

"Bank account", in C++

```
Class BankAccount {...
    int balance ( ) const ;
    void deposit (int) ;
...}
```

from this C++ syntax to an equational specification?

- apparent specification

  $\texttt{balance} : \texttt{void} \longrightarrow \texttt{int}$

  $\texttt{deposit} : \texttt{int} \longrightarrow \texttt{void}$

  the intended interpretation is not a model

- explicit specification

  $\texttt{balance} : \texttt{state} \longrightarrow \texttt{int}$

  $\texttt{deposit} : \texttt{int} \times \texttt{state} \longrightarrow \texttt{state}$

  the intended interpretation is a model,
  but the object-oriented flavour is lost

"Decorations"

Decorations:

- m for modifiers
- a for accessors (`const` methods)
- p for pure functions

- decorated specification

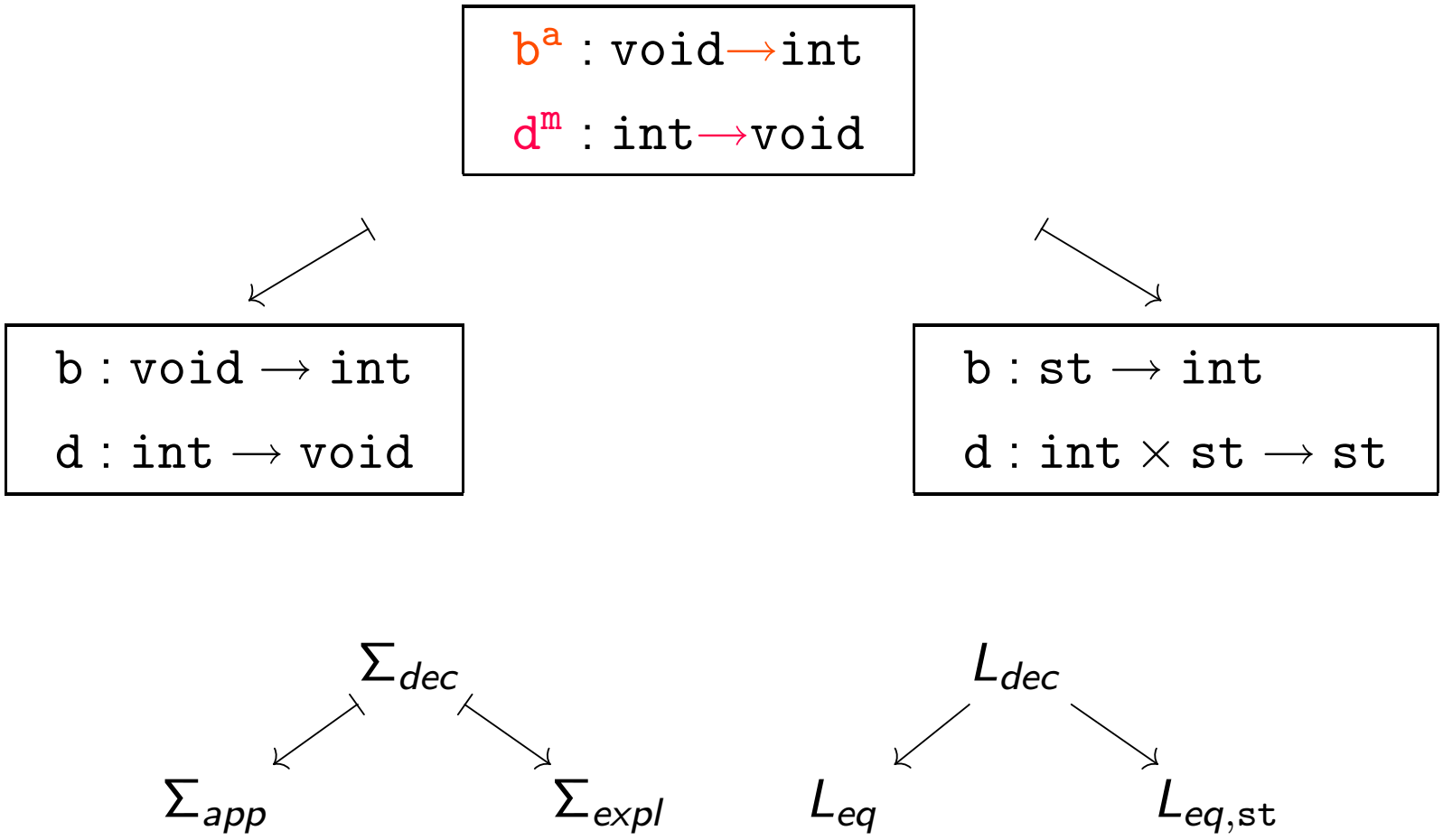
  $\texttt{balance}^{\texttt{a}} : \texttt{void} \rightarrow \texttt{int}$

  $\texttt{deposit}^{\texttt{m}} : \texttt{int} \rightarrow \texttt{void}$

  the intended interpretation is a model
  and the object-oriented flavour is preserved
  but this is not an equational specification!

However, it is a specification for some diagrammatic logic L_{dec} called the decorated equational logic

Homomorphisms of logics



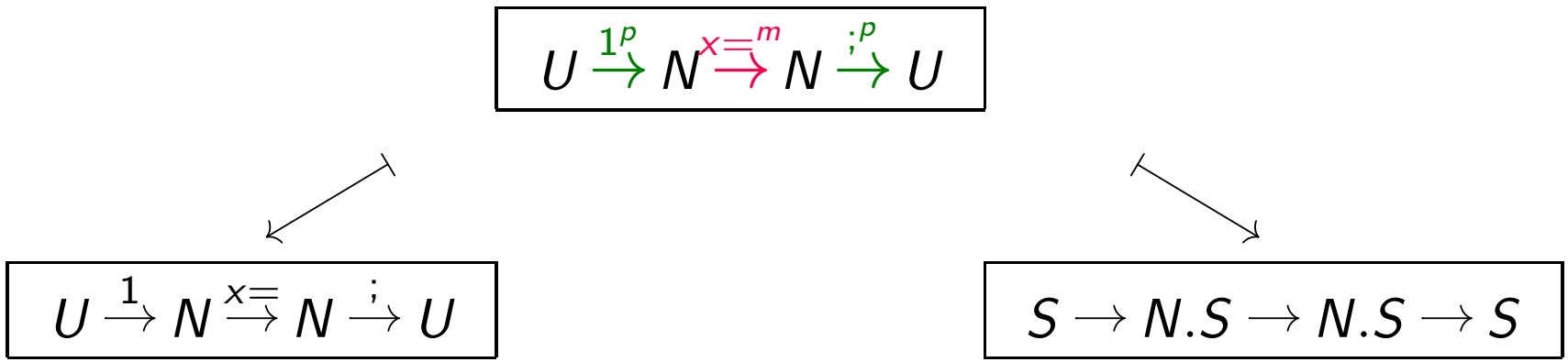
Instructions as decorated morphisms

A program in C

```
int x, y, z;
x = 1;
y = 2;
z = (y = ++x) + (x = ++y);
```

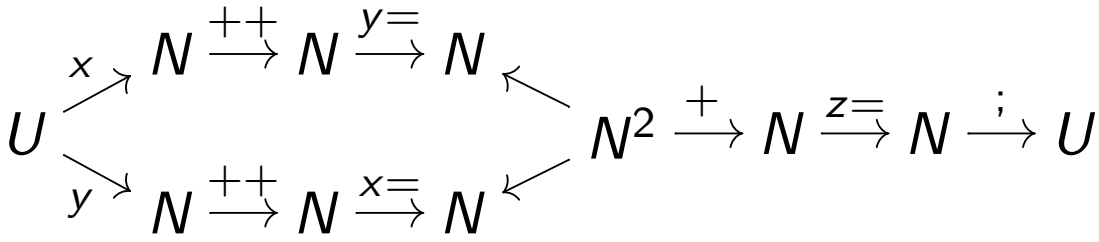
- if `y = ++x` is evaluated before `x = ++y`
  then in the resulting state $x = 3, y = 2, z = 5$
- if `x = ++y` is evaluated before `y = ++x`
  then in the resulting state $x = 3, y = 4, z = 7$

```
x = 1;
```

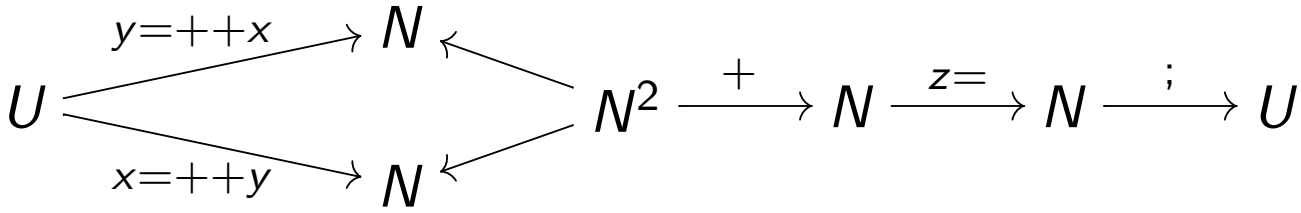

$U \xrightarrow{1^p} N \overset{x=}{\Rightarrow}{}^{m} N \xrightarrow{;^p} U$
$U \xrightarrow{1} N \overset{x=}{\Rightarrow} N \xrightarrow{;} U$
$S \rightarrow N.S \rightarrow N.S \rightarrow S$

```
z = (y = ++x) + (x = ++y);
```

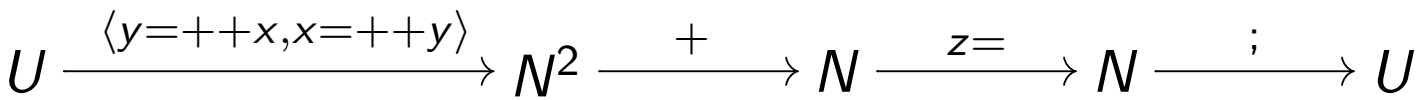
Apparently (cf. $ss0 + sss0$)



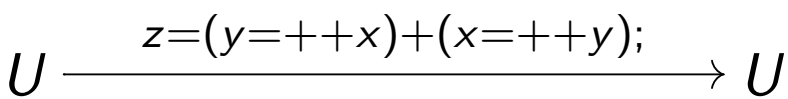
composition rule



pairing rule



composition rule

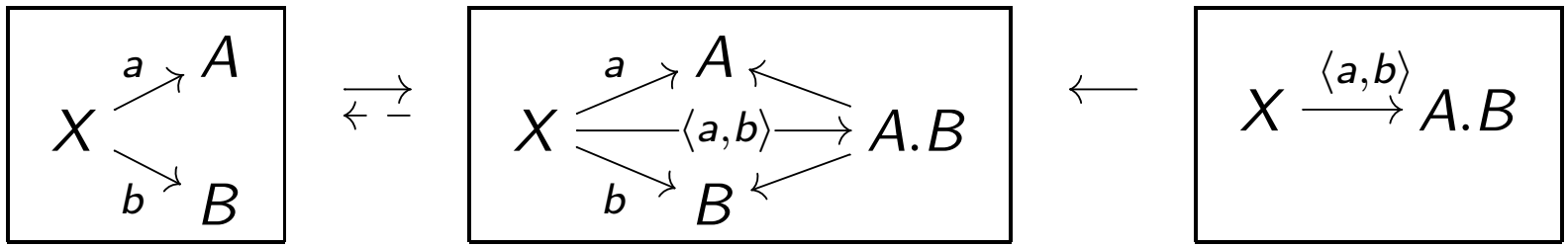


The pairing rule(s)

BUT the pairing rule cannot be decorated!

Pairing rule

$$\frac{a : X \to A \quad b : X \to B}{\langle a, b \rangle : X \to A \times B}$$

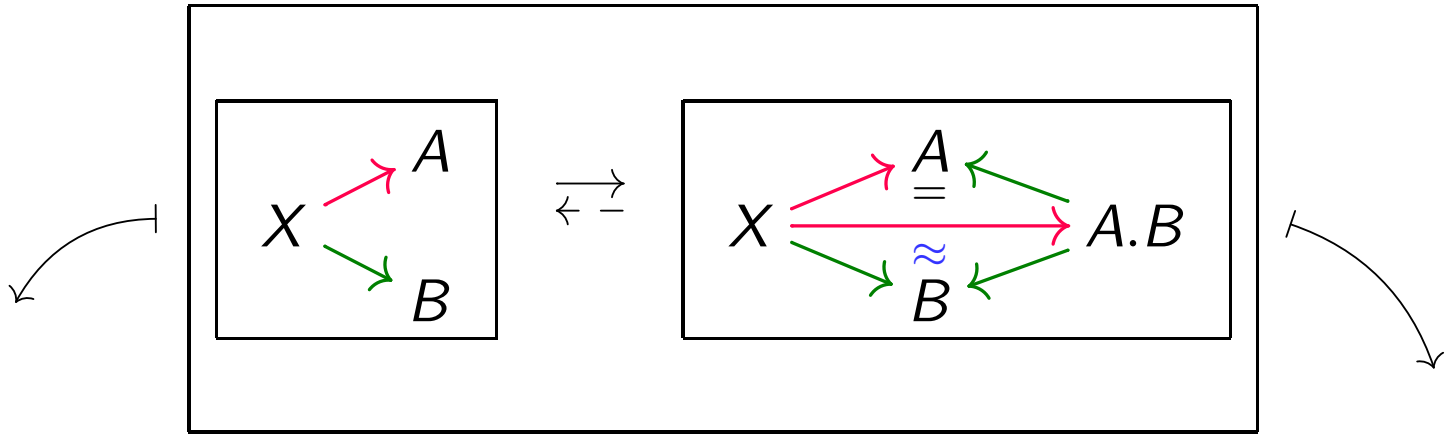


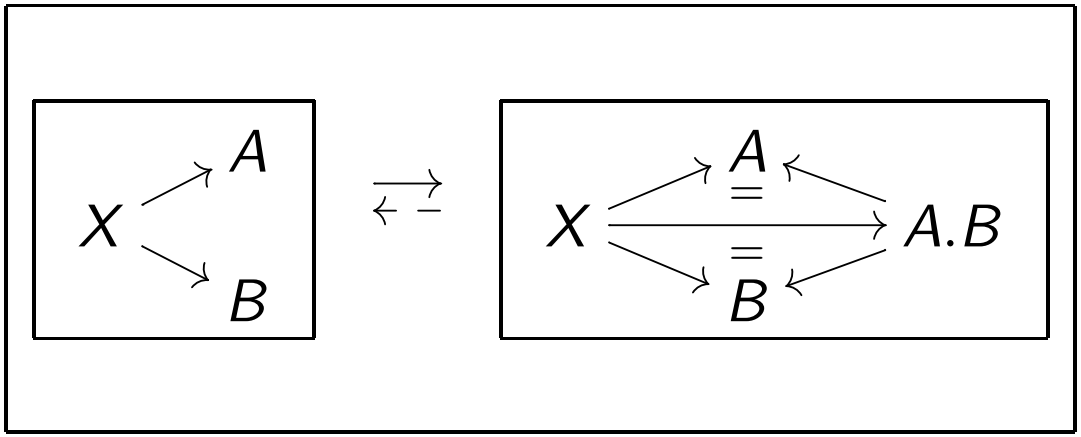
The pairing rule can be decorated when either a or b is pure
When both a and b are modifiers, the pairing rule may be replaced by one of the two sequential pairing rules, which are apparently equivalent and which can be decorated

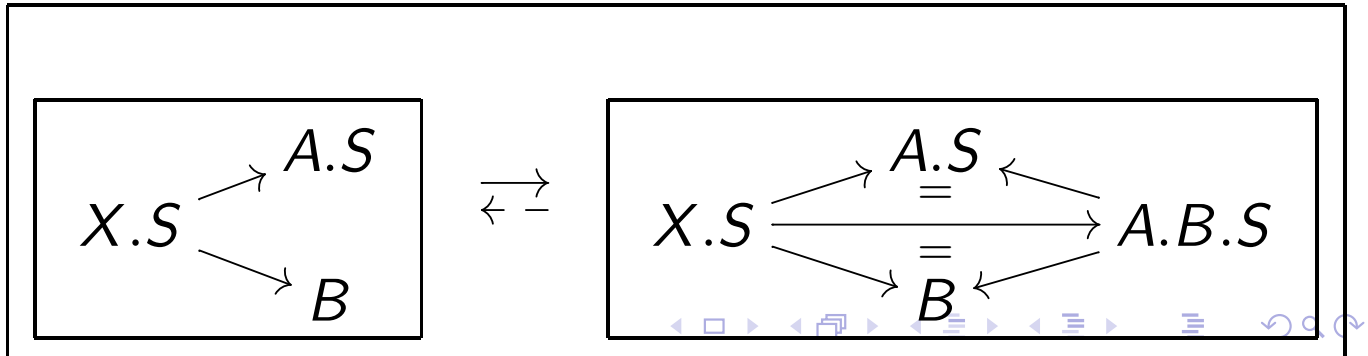
Sequential pairing rules

$$\frac{a : X \to A \quad b : X \to B}{(\mathtt{id}_A \times b) \circ \langle a, \mathtt{id}_X \rangle : X \to A \times B} \qquad \frac{a : X \to A \quad b : X \to B}{(a \times \mathtt{id}_B) \circ \langle \mathtt{id}_X, b \rangle : X \to A \times B}$$

A decorated pairing rule







Outline

Categorical semantics, beyond functional languages

What is an effect?

- Moggi [1989], cf. Haskell:
  an effect "is" a monad
- Plotkin & Power [2001]:
  an effect "is" a Lawvere theory
- DDFR [2010]
  an effect "is" a mismatch between syntax and semantics
  which can be described by a span of diagrammatic logics

In favour of our approach:

(+) a new point of view on states

(+) a new point of view on multivariate operations

(+) a completely new point of view on exceptions with handling

(+) a duality between states and exceptions

Some papers

- J.-G. Dumas, D. Duval, L. Fousse, J.-C. Reynaud.
  States and exceptions are dual effects.
  arXiv:1001.1662 (2010).
- J.-G. Dumas, D. Duval, J.-C. Reynaud.
  Cartesian effect categories are Freyd-categories.
  JSC (2010).
- C. Dominguez, D. Duval.
  Diagrammatic logic applied to a parameterization process.
  MSCS 20(04) p. 639-654 (2010).
- D. Duval.
  Diagrammatic Specifications.
  MSCS (13) 857-890 (2003).