Nick Rossiter, Michael Heather

Typing in Information Systems

Computer Science and Digital Technologies
Northumbria University, Newcastle NE1 8ST, UK

Nick.rossiter1@btinternet.com

http://nickrossiter.org.uk/process/index.html
http://computing.unn.ac.uk/staff/cgnr1/index.htm

GRAPH OPERADS LOGIC

AGMP X GOL Tallinn 25-28 June 2013

Typing

- Most important property in computing science
- Defines for an attribute:
  - Permissible values
  - Permissible operations

- A type is a category of any complexity, including a discrete category

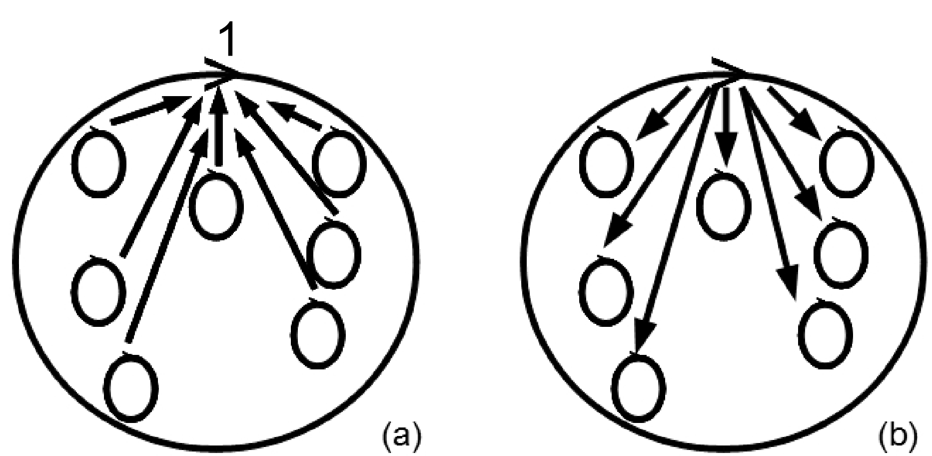
Simplest Typing in Category Theory

(a) A category with a terminal object

(b) The opposite category with typing arrows

B

B^{op} or 1_B



$1_b: b \rightarrow 1_B$

Cartesian
Closed
(potentially)

Identity
Functor
(potentially)

Nature of Typing

- Where does typing come from?

- Typing resides in the system itself.

- Typing is of the nature of the system and forms part of the Universe.

- Typing must therefore reside in nature and arise from relationships in nature.

- The Universe consists of entities related one to the other.

- Thus each entity affects every other.

Existence and Cartesian Closure

- Existence is not just a first order effect but needs an inherent higher order formalism.
- The relationship between any pair of entities depends on every possible path between them.
- In category theory this is the property of cartesian closure found in the highest structure possible -- the identity natural transformation designated as the *topos*.
- However if every entity is related to every other it follows that the relationship is both ways but not just a simple inverse relationship as appears from the laws of physics.

Duality

- A category **C** of objects and arrows between the objects will have a dual $\mathbf{C}^{op}$ with arrows reversed.
- The whole universal structure of both-ways relationships will then be represented by the product $\mathbf{C}^{op} \times \mathbf{C}$.
- This gives rise to the principle of duality throughout the Universe.
- $\mathbf{C}^{op} \times \mathbf{C}$ is cartesian closed, with products, terminal object and exponentials

Ubiquity of Duality

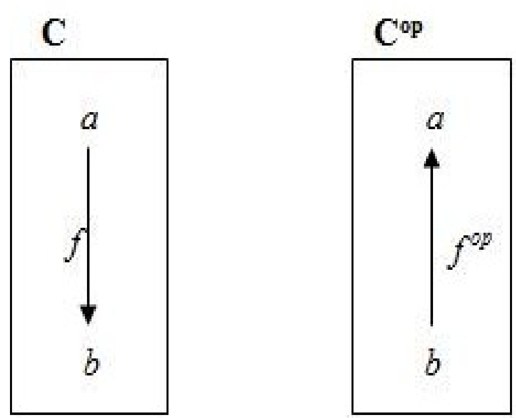
- Duality is a common enough concept in mathematics, philosophy and most of the sciences with some renowned examples like the mind-body duality.
  - It also appears in other versions of contrast as between the dynamic and the static and between global and local.
- To capture the full effect and subtleties of opposing views and relationships a single view of the duality is needed as a process (e.g. a monad).

Duality and Variance

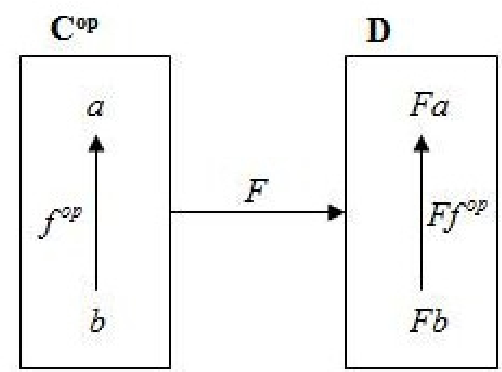
- Duality is not a closed Boolean view.
  - Rather it encapsulates opposite orderings within a single (functorial) concept of variancy.
  - These may be conveniently labelled covariant and contravariant but only relative one to the other and not as absolute descriptions.
- Systems theory is a case in point where these different views need to be integrated.
  - Thus for object-oriented computing systems, covariance is assumed for specialisation (looking forward) and contravariance for generalisation (looking back).
- The natural categories of process as advanced by Whitehead encompass this contravariancy found in reality.

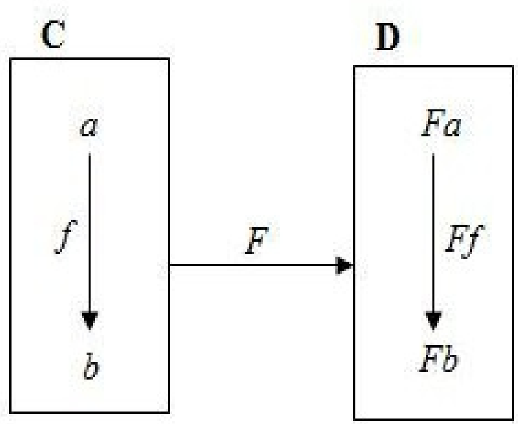
Covariant and Contravariant Functors

Opposite

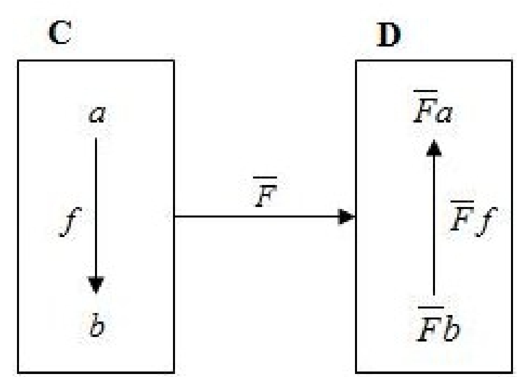


Covariant F





Covariant F



Contravariant F-bar

Contravariancy

- Highlighted by Lawvere in 1969 as basic property in the intension-extension relationship
  - Governing data values in the context of their name and type
  - Basic property of universe
- Lawvere defined the relationship between intension and extension in terms of adjoint functors with contravariant mapping
  - Used concept of hyperdoctrine
  - Some 'translation' needed for applied science

Why Contravariant?

The extension is of the form:

value → *name*   N:1

The intension is of the form:

name → *type*   N:1

If these arrows were reversed, they would not be determinations (functions) so can reject such forms:

name → *value*   1:N

type → *name*   1:N

Turn around one arrow

But the common attribute in extension and intension – *name* – is codomain in extension and domain in intension.

So cannot do simple covariant mapping of one to the other.

Need to turn around the arrow in the intension

name → *type* ➡ *type* → *name*

And map this onto *value* → *name* in extension

So that *value* is related to *type* in the context of a common name

Contravariancy Technicalities

source arrow

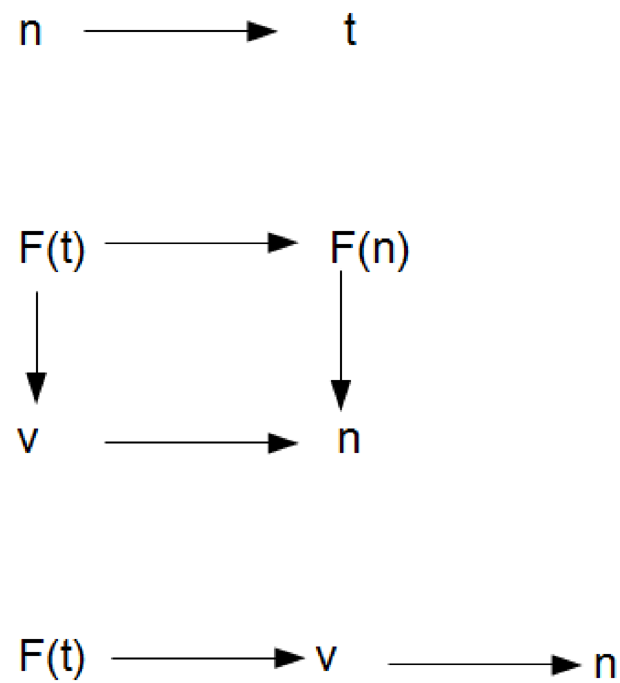
Apply functor F

Turn source round

Map onto target

Square must commute

composition



Ultimate Contravariancy

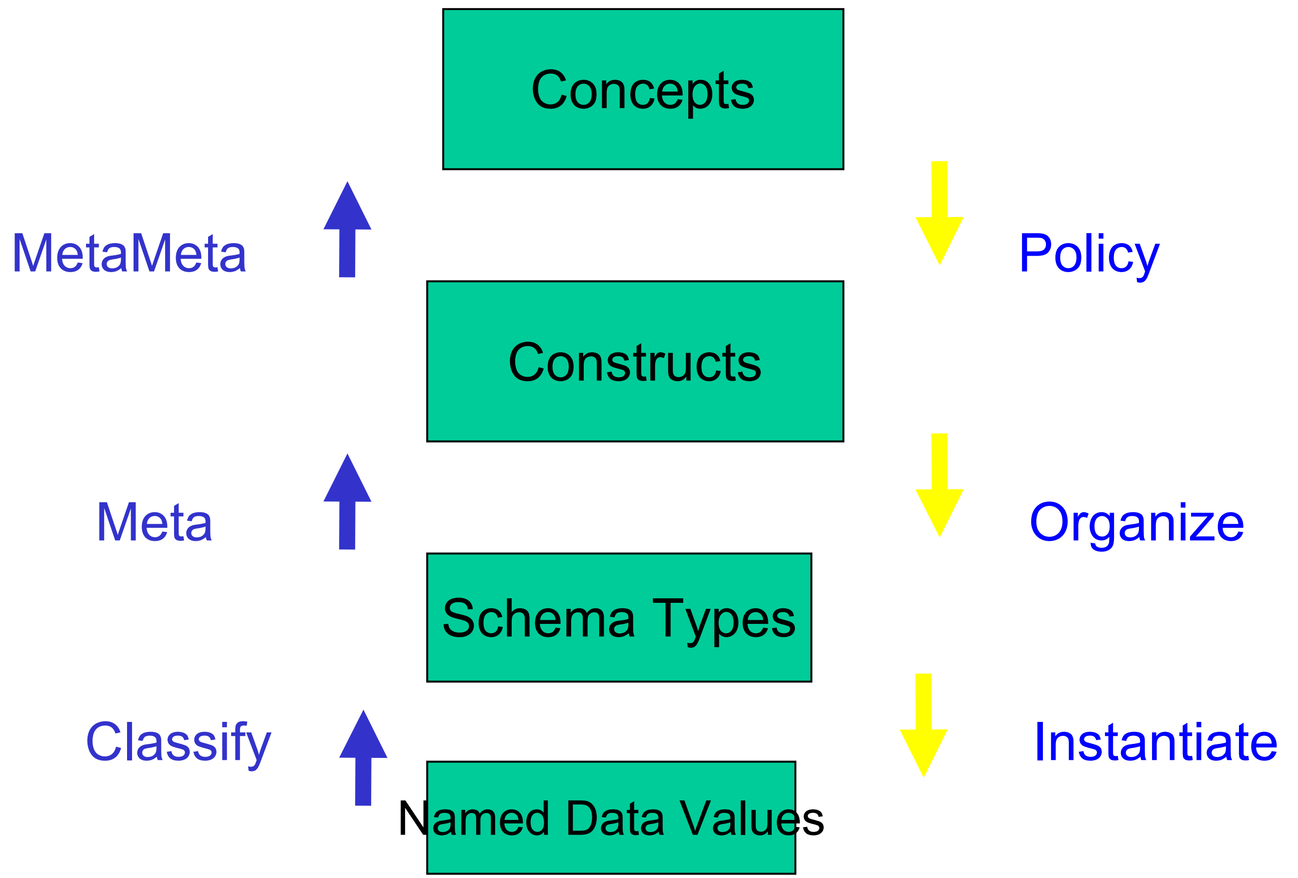
- A three-level structure is sufficient to provide complete closure with internal contravariant logic providing a generalisation of negation.
  - Further levels are redundant
- Contravariancy across levels provides more sophisticated reversals such as reverse engineering.
- The ultimate contravariancy is to be found in the universal adjointness

  - between any pair of functors contravariant one to the other

  - to provide both the quantitative and qualitative semantics of intension-extension logic.

Worked Example of Three-level Architecture

- Choices for realisation in formal terms
- Informal look at structures and relationships
- Outline in informal categories
- Two-way mappings as adjunctions
- Examples of Contravariancy



Downward arrows are intension-extension pairs

Figure 1: Informal requirements for Information System Architecture

Formalising the Architecture

- Requirements:
  - mappings within levels and across levels
  - bidirectional mappings
  - closure at top level
  - open-ended logic
  - relationships (product and coproduct)
- Choice: Category theory as used in mathematics as a workspace for relating different constructions

blue – category, red - functor, green - natural transformation

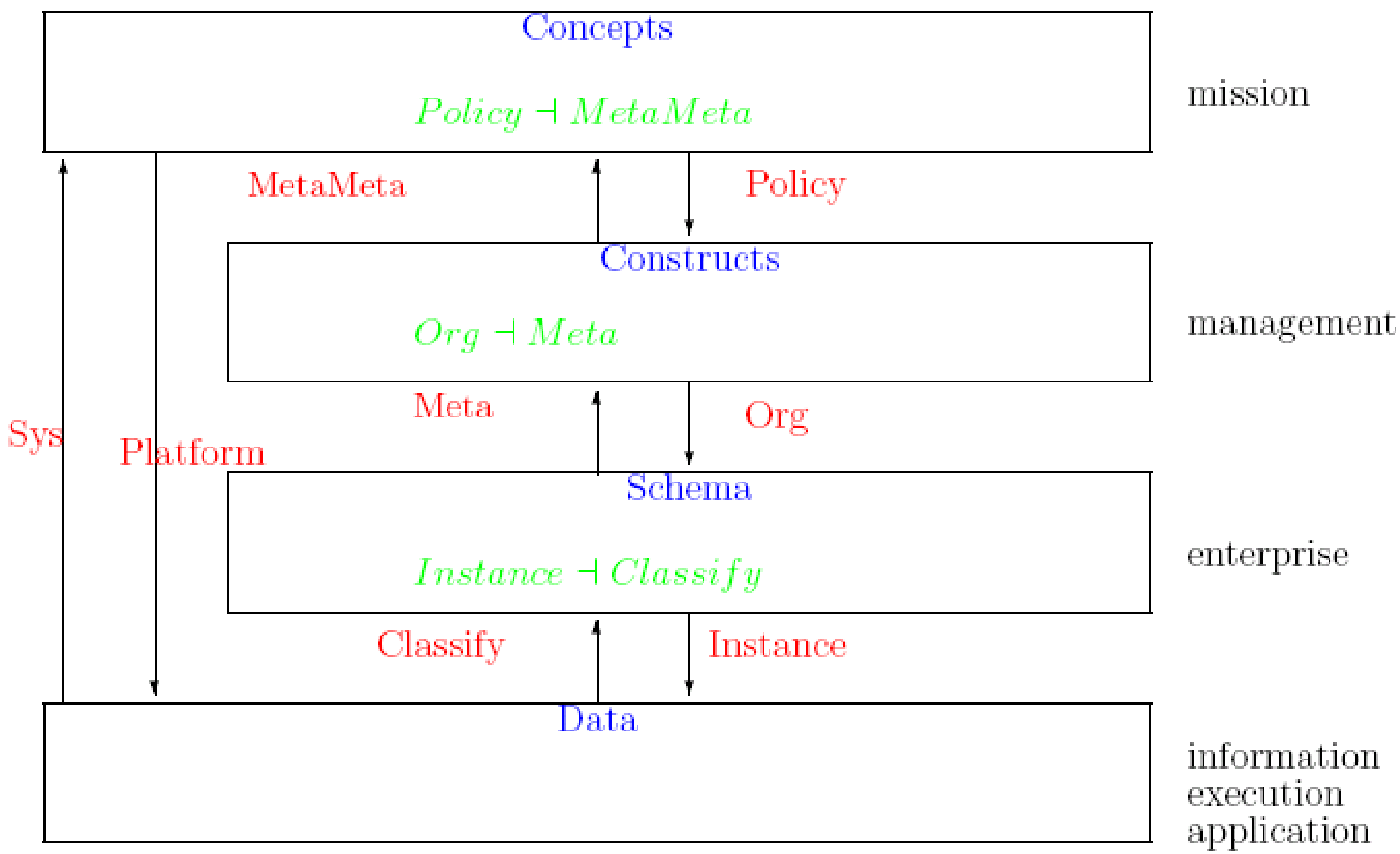


Figure 2: Interpretation of Levels as Natural Schema in General Terms

black - objects

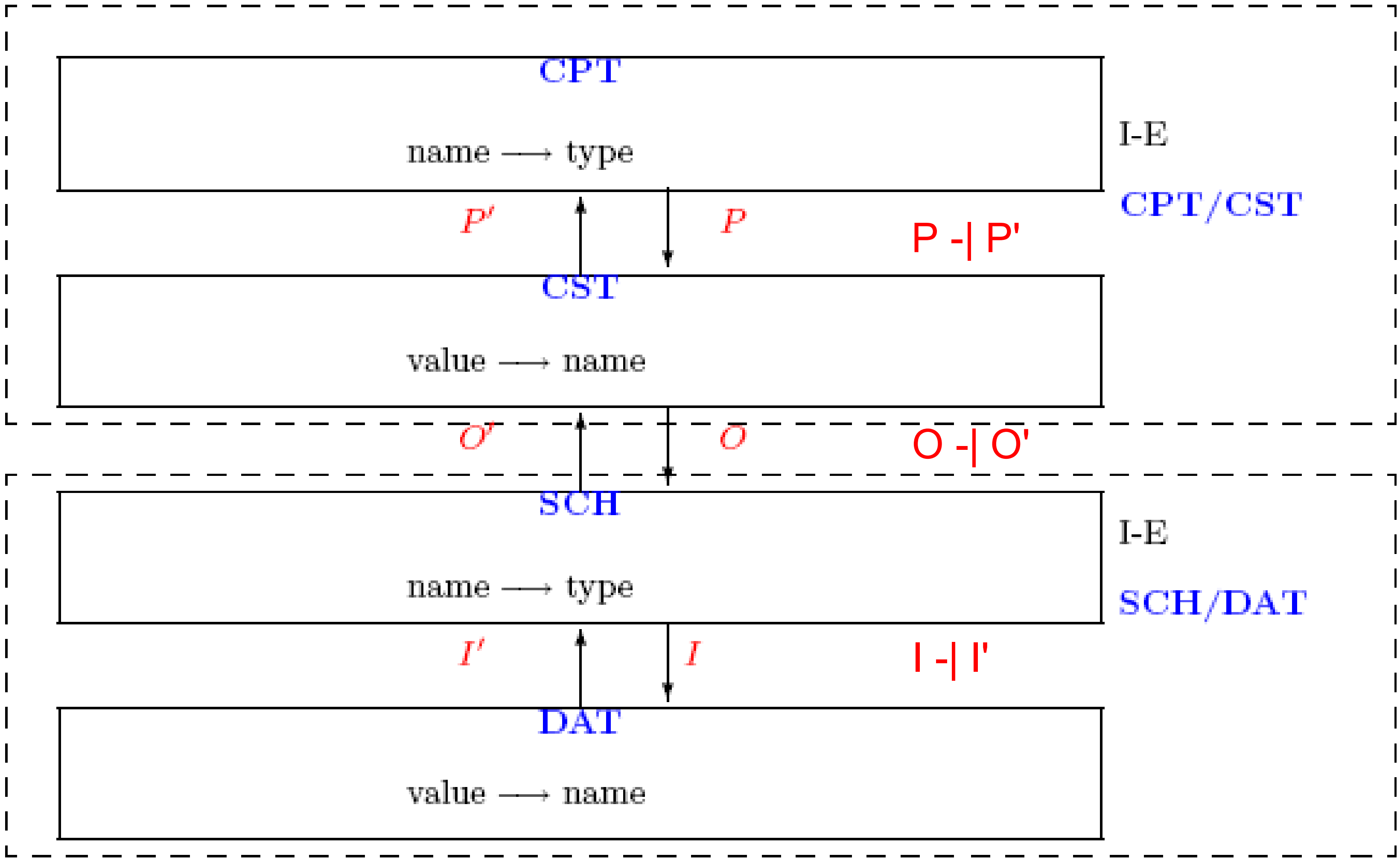


Figure 4: Defining the Three Levels with Contravariant Functors and Intension-Extension (I-E) Pairs

(Organisational interoperability)

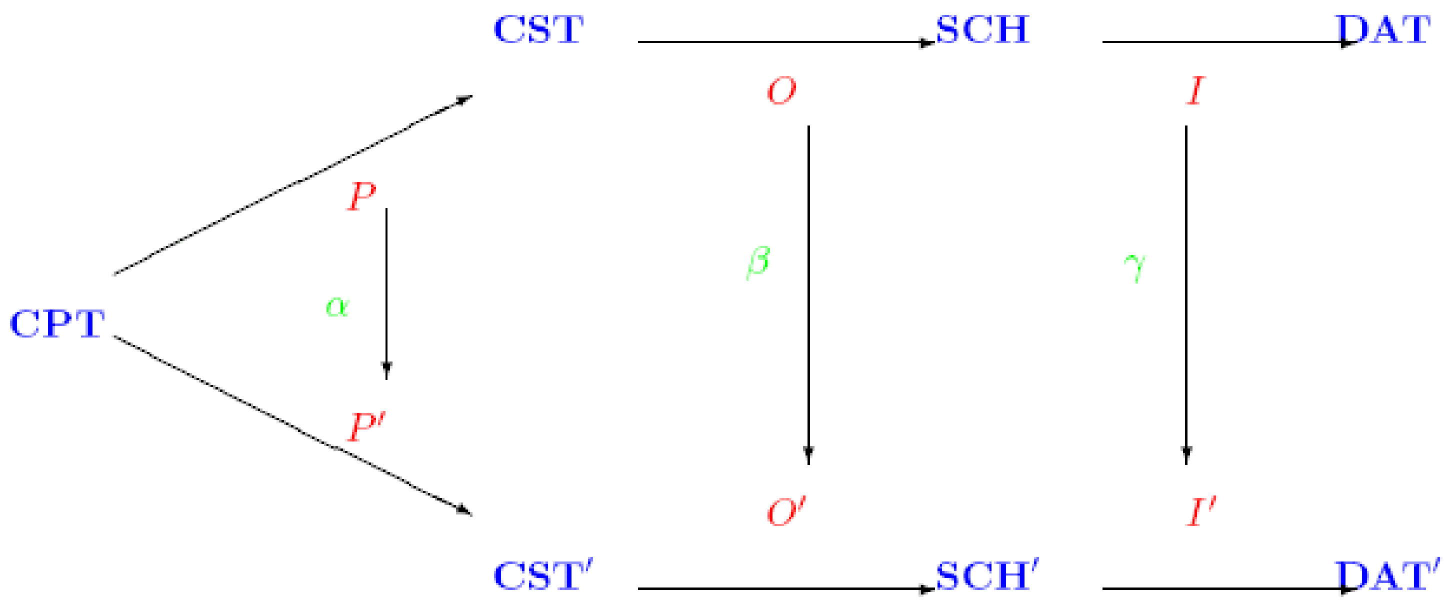


Figure 3: Example for Comparison of Mappings in two Systems
Categories: CPT concepts, CST constructs, SCH schema, DAT data,
Functors: P policy, O org, I instance,
Natural transformations:

Level	Template	Property	Relational Database (aggregation)	Abstract Data Type (encapsulation)
CPT	name ⟶ type	attribute ⟶ property	table ⟶ aggregation	ADT ⟶ encapsulation
P'	↗ ↖	↗ ↖	↗ ↖	↗ ↖
CST	value ⟶ name	registration_no ⟶ attribute	birth_type ⟶ table	BST ⟶ ADT
O'	↗ ↖	↗ ↖	↗ ↖	↗ ↖
SCH	name ⟶ type	car_reg ⟶ registration_no	birth_record ⟶ birth_type	aTree ⟶ BST
I'	↗ ↖	↗ ↖	↗ ↖	↗ ↖
DAT	value ⟶ name	'x123yng' ⟶ car_reg	<'Smith', 25 mar 1980, 'Torquay'> ⟶ birth_record	instance of tree (nodes/links) ⟶ aTree

Figure 5: Examples of Levels in the Three Level Architecture

Cross-over arrows indicate contravariant mapping

If functors are adjoint, there is a unique relationship between them (a natural bijection).

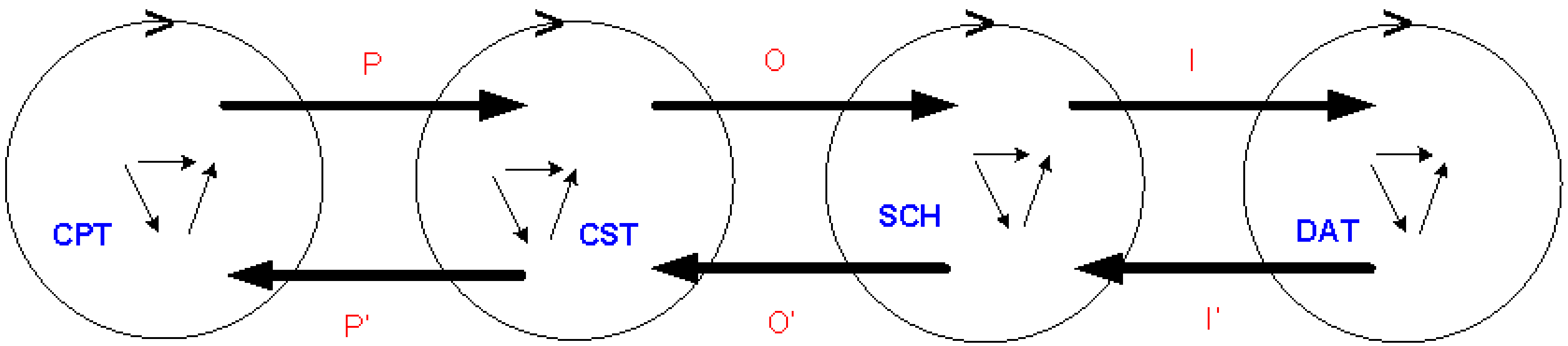


Figure 6: Composition of Adjoints is Natural

Can write for instance IOP -| P'O'I' and OP -| P'O'

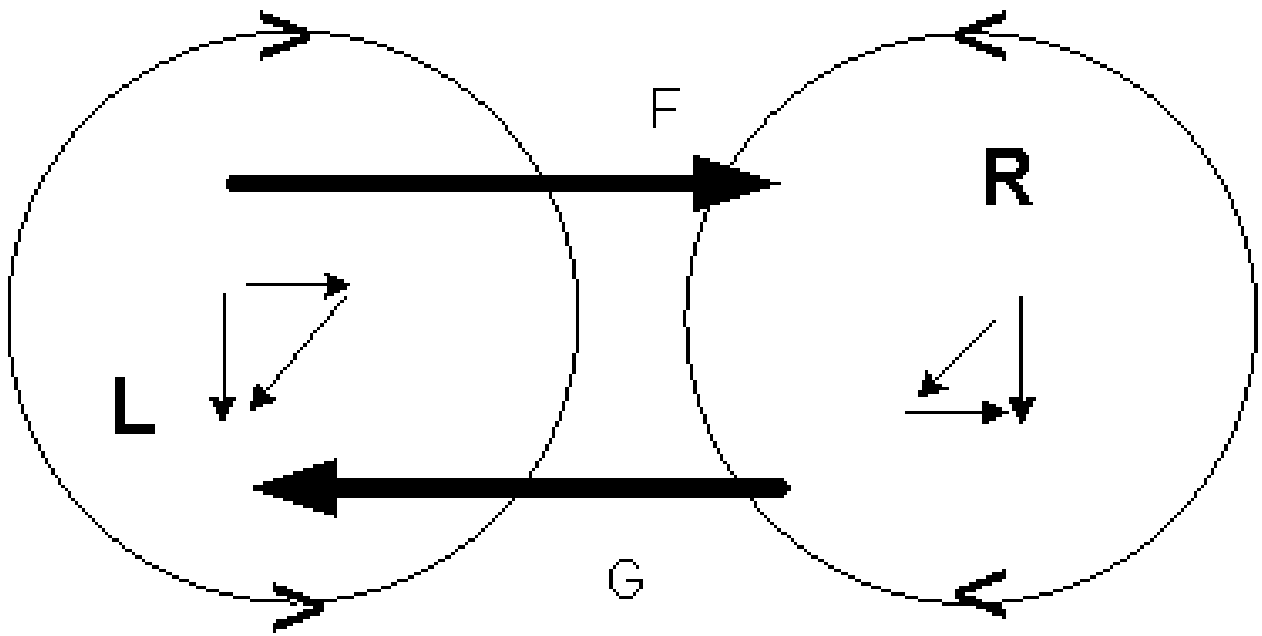
Adjointness

- Probably the most important feature of category theory
- Deals with relative ordering in duality between functors
- For dual functors L and R written (if holds):

  L --| R

  meaning L is left adjoint to R and R is right adjoint to L
- Unique solution to dual relationship between functors
- Full expression is a 4-tuple <L, R, η, ε>

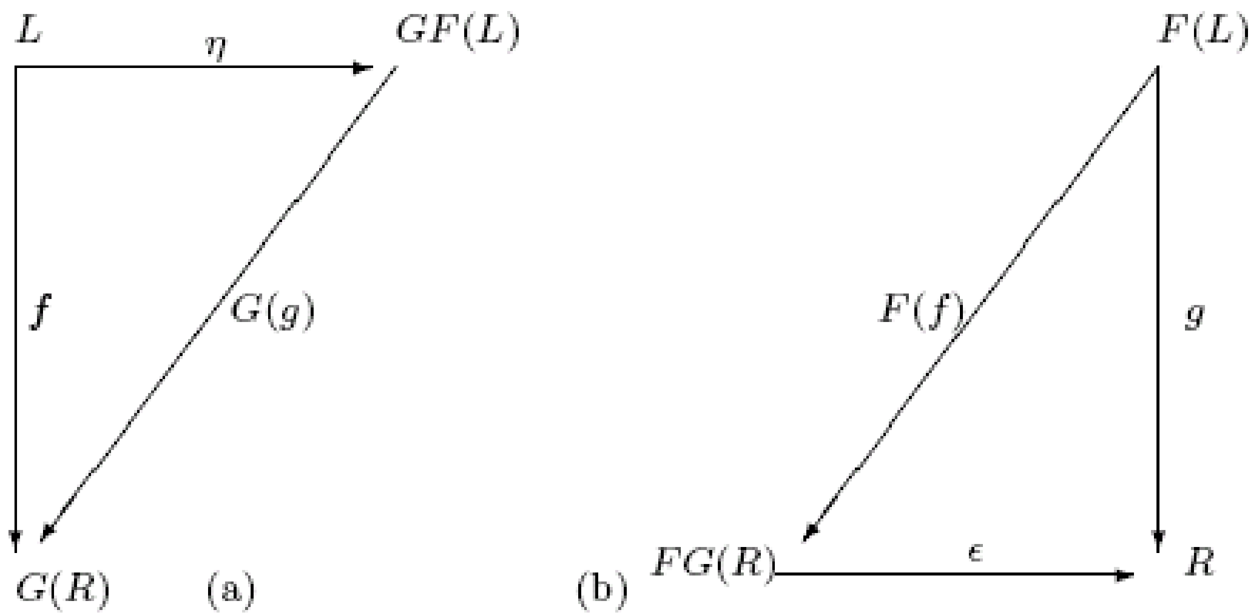
Adjointness between Functors F and G mapping Categories **L** and **R**



F ⊣ G

Composition Triangles in Detail

a) unit of adjunction ❧ b) co-unit of adjunction ☙



If isomorphism, diagram collapses with η = 1 and ε = 0

1_L = GF(L); FG(R) = 1_R