A formal foundation of systems theory and systems engineering

Dr. Dominique Luzeaux

dominique.luzeaux@polytechnique.org

A formal foundation of SE

- **Our goal**: a formal theory of systems engineering.

- **Our vision**: introduce the as minimal as possible mathematical foundations, in order to engineer complex systems, with sufficient abstraction in order to be implementation independent.

- **Main capabilities addressed**: system modeling, system specification, system verification, system engineering process modeling.

Quick overview of literature

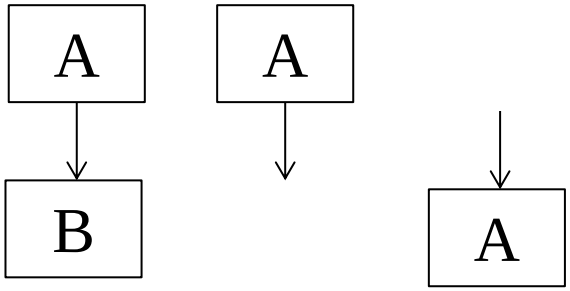
- **System modeling**: structural (e.g. co-algebras, categories, sketches) or behavioral (e.g. abstract machines: FA, PN…) approaches.
- **System specification**: logical approaches (axiomatic presentation, sequent calculi) based on structural or behavioral models.
- **System verification**: logical approaches (completeness results, theorem proving) based on structural or behavioral models.
- **System engineering process**: ??

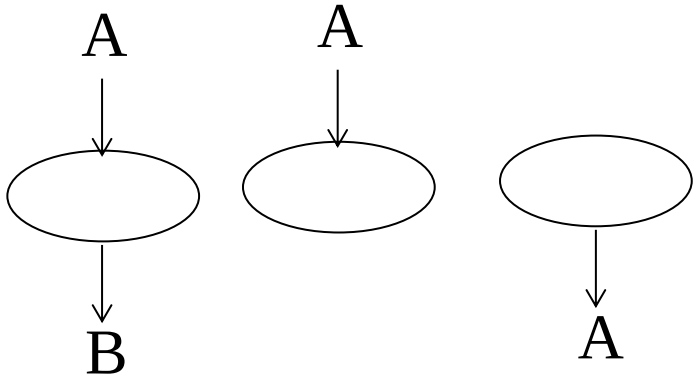
→ (almost) all research focuses on one of the previous issues but not on an encompassing framework.

What is a system.

- *“a combination of interacting elements (…)”*

- **SysML**: blocks (sometimes with explicit I/O ports) with oriented links between them.
- **OPM**: things, i.e. objects and processes where a process transforms (generates or consumes) an object.





- In other words: A ⟶ B A ⟶ ● ● ⟶ A

- *Actually, one should consider different types of arrows.*

A mathematical first model

- We will manipulate: nodes (objects) and edges (arrows).
- A powerful existing mathematical framework: **Category theory**.

  → Foundations of mathematics possible.

  → Algebra/topology/logic can be done within the same framework.

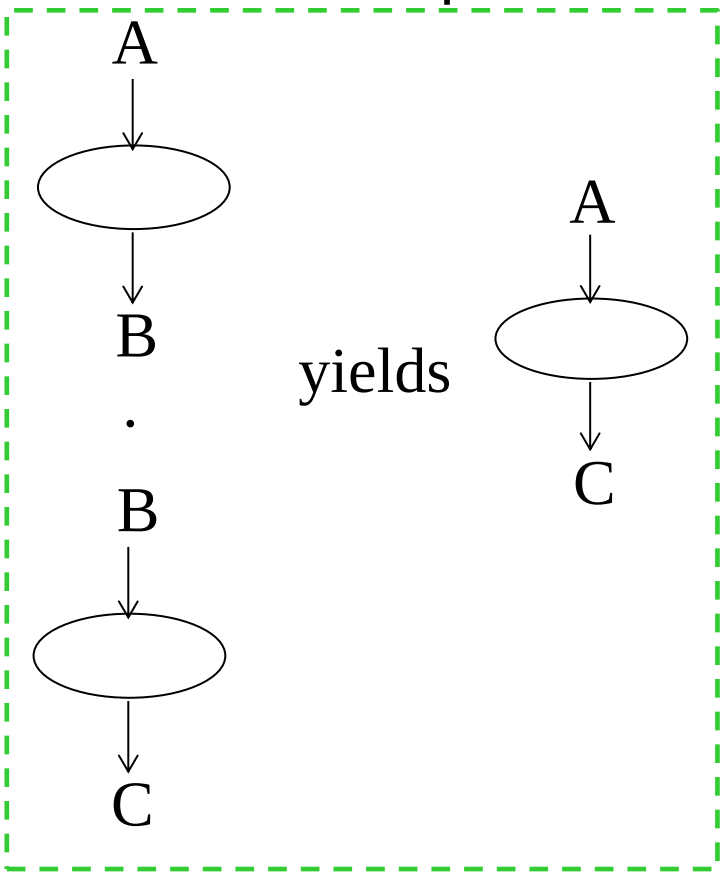
  → Abstract (but graphical) formalization.

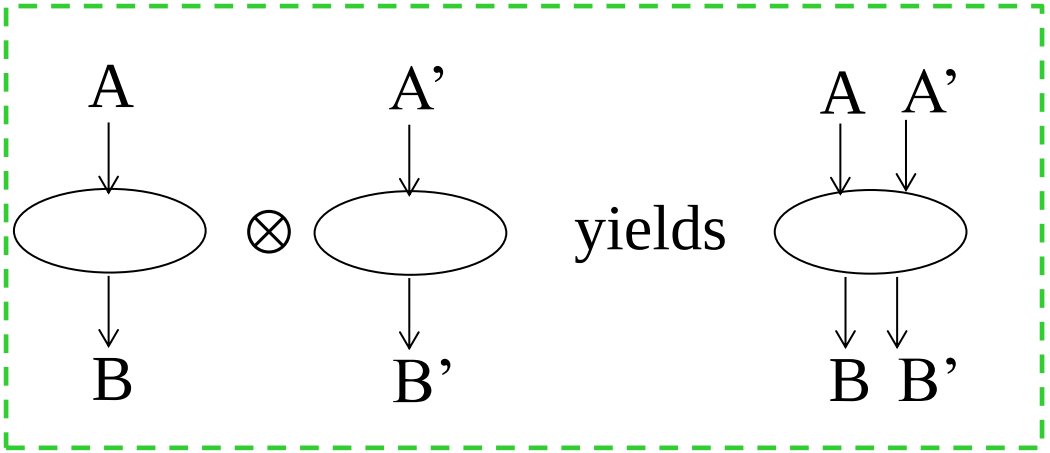
  → Unification of concepts.

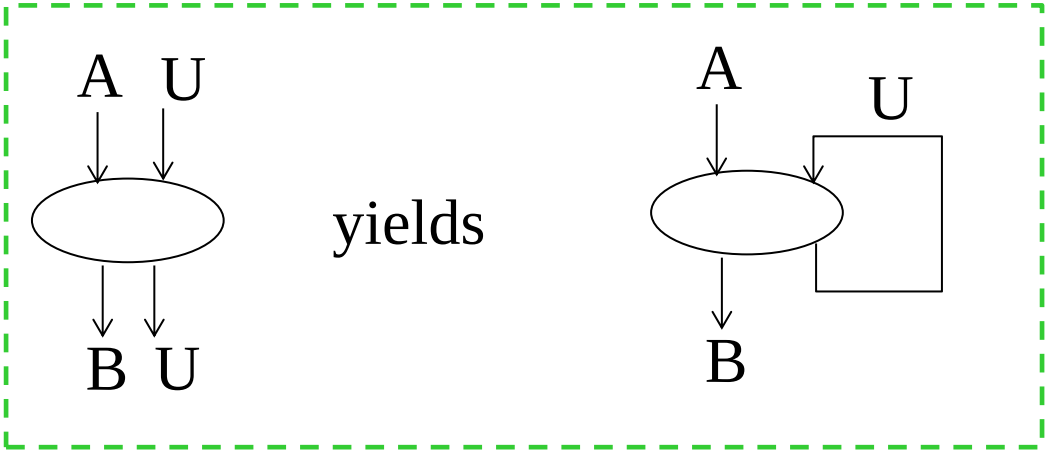
- *For the category theory expert: arrows can be typed (introduce slice categories).*

Working with systems in engineering

- Sequential composition.
- Parallel composition.
- Feedback loops.







Beware the bias of our education!

- All textbooks introduce as first categorical notions: Cartesian products and co-products (≃ unions) → influence of Bourbakian set theory!
- But what does a Cartesian product of systems mean?
  - *Definition: "For two systems A and B, there is a system C such that C can be mapped to A and to B, and for any system D sharing that property, there is a unique map from D to C".*
- And the co-product? (exchange "to" and "from").
- Actually:
  - A *Cartesian product* corresponds to juxtaposition by taking interface constraints into account (cancel some differences through satisfaction of boundary conditions).
  - A *co-product* corresponds to sharing or synchronization (identification of synchronization points given by a quotient of an equivalence relation).

→ Should these concepts be BASIC bricks of systems and SE?

Why this bias (found in most literature on formal models of systems)?

- Because:
  - Products are the building blocks of set theory.
  - Products are extensively used in programming languages (*arrays, lists*).
  - With products, you define exponentials (*functions become objects as in Functional Programming*).
  - Closed Cartesian categories correspond to untyped lambda-calculus, which provide canonical models of standard computation.

- BUT:
  - **Not adequate to define easily feedback** (*need of exhibiting explicitly behaviors and using a replacement-like technique: y=f(x,u) and u=h(x) yields y=f(x,h(x))*).
  - **Not adequate to model consumption and limited generation of resources** (*… one of the drawbacks of classical logic*): inherent to most verification techniques (hence the necessity of adding ad hoc counters or tokens in most usual formal models).

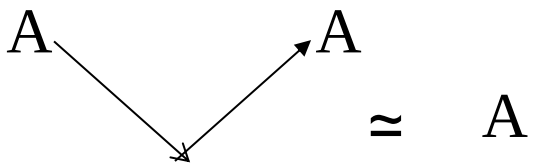
The way forward: basics

- A category (**objects**; **morphisms** for sequential composition; a **tensor product** for parallel composition).

  → *(symmetric) monoidal category*.

- Add for convenience a **dual operator**, in order to "simplify" non-productive loops:

  → *compact closed categories*.



- **That's it!**

  - You get for free a **trace** for symmetric monoidal categories (→ feedback).

- *Remark: these categories have been used recently in quantum physics (because of creation and annihilation operators in QFT). And they correspond to categories of several classes of Petri nets.*

The way forward: advanced (1/2)

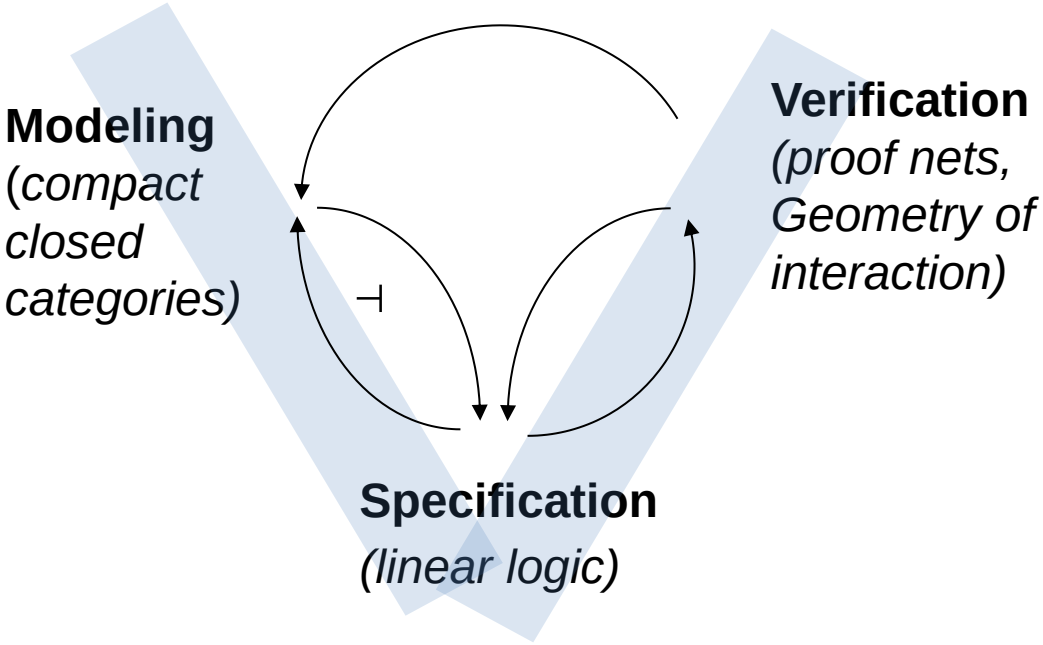
- Compact closed categories correspond to multiplicative intuitionistic linear logic (*the adequate modification of classical logic in order to manage finite resources: consumption and generation can be adequately modeled in logical reasoning → specification and validation*).

- Compact closed categories + products correspond to additive multiplicative intuitionistic linear logic (*external and internal choices can be modeled*).

- Compact closed categories + comonad obtained by adjunction with a cartesian closed category (*model of "traditional" computation*) correspond to full intuitionistic linear logic (*including some resources to be finite and other to be infinite*).

The way forward: advanced (2/2)

- The correspondences are instances of a deeper result from categorical logic:



- … the three main steps of the SE life-cycle process are related together in an encompassing formal framework.

Concluding remarks

- Objects can be enriched: several types (slice categories), time-varying (*sheaves, modeling among others hybrid systems with infinitesimal time*).
- Additional structure can be given on the base category (*products, coproducts, limits…*) to model synchronization, hierarchies between systems (*e.g. recursive models as in SysML or OPM*).
- Monads can be added (*as arising in co-algebraic or some modal logic approaches*) and give birth to Lafont categories.
- **This research lays also the path for less ambiguous semantics of modeling languages.**