Oregon Lectures

Logics & Lambda Calculi 3

Valeria de Paiva
Topos Institute

June 2025


Fork

Linear Functional Programming

once upon a time...

- Ritter's PhD on categorical combinators for the Calculus of Constructions (1992) + de Paiva's PhD on models of Linear Logic & linear lambda-calculus (1990)
- Together for Categorical Abstract Machines for Linear Functional Programming
- Project xSLAM = Explicit Substitutions for Linear Abstract Machines (1997-2000)
- "categorical deep structured syntax" borrowing Hyland's phrase

eXplicit Substitutions Linear Abstract Machine:xSLAM

Twenty-five years later:

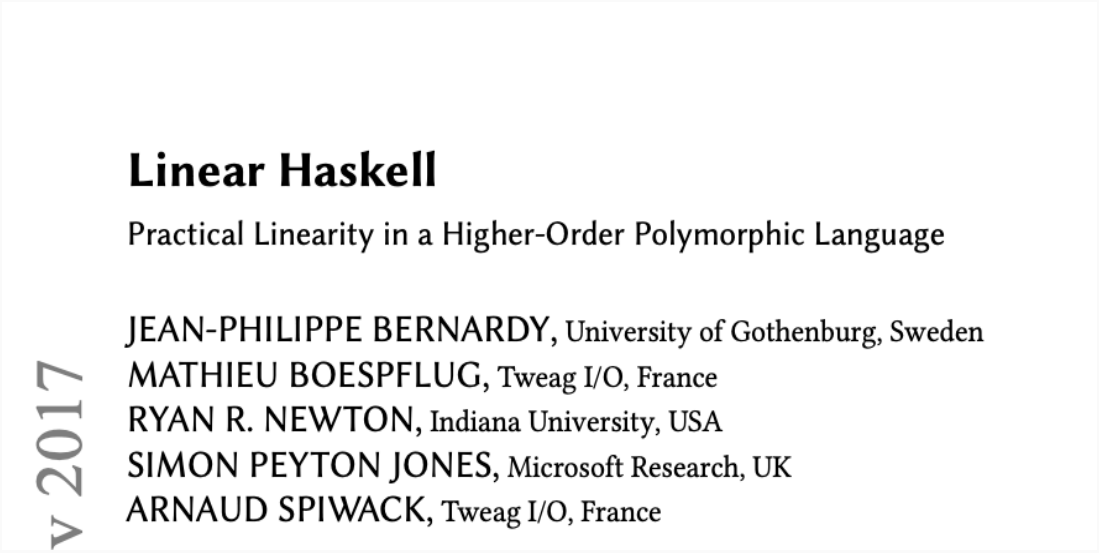
- More than 10 conference papers
- Two journal papers
- Master's thesis: 'An Abstract Machine based on Linear Logic and Explicit Substitutions', F. Alberti, 1997
- International workshops: 'Logical Abstract Machines' (Saarbruecken), 'Logical Abstract Machines' (Birmingham)
- Dagstuhl meeting: 'Linear Logic and Applications' (1999)

Many years later…

(merged June 2020)

time to try again?

Years later…



Linear Haskell

Practical Linearity in a Higher-Order Polymorphic Language

JEAN-PHILIPPE BERNARDY, University of Gothenburg, Sweden
MATHIEU BOESPFLUG, Tweag I/O, France
RYAN R. NEWTON, Indiana University, USA
SIMON PEYTON JONES, Microsoft Research, UK
ARNAUD SPIWACK, Tweag I/O, France

v 2017

Despite their obvious promise and a huge research literature, linear type systems have not made it into mainstream programming languages.
Even though linearity has inspired uniqueness typing in Clean, and ownership typing in Rust.

Explicit substitutions?

> *In computer science, lambda calculi are said to have **explicit substitutions** if they pay special attention to the formalization of the process of substitution. This is in contrast to the standard lambda calculus where substitutions are performed by beta reductions in an implicit manner which is not expressed within the calculus.*

Syntactic calculi of explicit substitutions, plenty of them. No models?

Explicit Substitutions

Substitution is the eminence grise of λ-calculus. The classical β rule

$$(\lambda x.a)b \to_\beta a\{b/x\}$$

uses substitution crucially though informally. Here a and b denote two terms, and $a\{b/x\}$ represents the term a where all free occurrences of x are replaced with b. [...] The correspondence between the theory and its implementations becomes highly nontrivial, and the correctness of the implementations can be compromised. (ACCL1989)

Curry-Howard Correspondence

1963

Lambda-calculus

1965

Cartesian Closed Categories

Intuitionistic Propositional Logic

- Syntactic calculus should have a categorical semantics
- Mathematics as a source of intuitions
- System implementation proved correct by construction
- Showing this for IPL and ILL (works for $CS4_{\Box,\Diamond}$ too)

Why functional languages?

- Programs as mathematical functions transforming inputs into outputs
- Work on inductively defined data structures like lists, trees
- no side effects $\Rightarrow$ can employ mathematical reasoning and substitute equals for equals
- function definition and application central part of languages like Haskell, OCaml, cakeML, Rust, Scala
- (usually) Have strong typechecking
- can detect many errors already at compile-time

NOT in this talk

- Intersection types (idempotent or not)
- Evaluation strategies
- Dynamic types
- Patterns
- Proof-nets
- Nominal syntax
- Geometry of Interaction
- explicit substitutions Reductions, etc

λ-calculus

Terms:

$$M ::= x \mid \lambda x : A.M \mid MM$$

Types:

$$A ::= G \mid A \to A$$

Meaning of terms:
x: Variable (placeholder)
$\lambda x : A.M$: Function expecting input of type A
MM: Function application

Single out well-formed terms (check whether correct kind of argument supplied)

Lambda calculus

$$\frac{\Gamma, x\colon A \vdash x\colon A \qquad \Gamma, x\colon A \vdash M\colon B}{\Gamma \vdash \lambda x\colon A.M\colon A \to B}$$

$$\frac{\Gamma \vdash M\colon A \to B \quad \Gamma \vdash N\colon A}{\Gamma \vdash MN\colon B}$$

Add recursively defined functions via `letrec`
add means for defining inductively defined data structures (lists, trees, etc)
Computation done via β-reduction

$$(\lambda x\colon A.M)N \rightsquigarrow M[N/x]$$

where $M[N/x]$ is M with x replaced by N

Environment machines

- Implementing β reduction efficiently is hard
- Traditional solution: Reduce λ-terms in an environment, storing bindings of terms for variables separately
- have two kinds of expressions:

  Environments $\langle M_i/x_i \rangle$: lists of bindings
  Terms: as before plus new closures
  $f * M$

Environment machines

- Modified β-reduction stores substitution in the environment $(\lambda x.t)u \Rightarrow \langle u/x \rangle * t$
- New **rewrite rules** to eliminate substitutions $\langle u/x \rangle * t \quad \Rightarrow^* \quad t[u/x]$

Transforming this '**trick**' of implementation into a logical calculus
$\Rightarrow$ the reason for 'explicit substitutions'

$\lambda\sigma$-Calculus

- **Term constructs:**
  Terms of the λ-calculus +
  - Application of a substitution to a term: $f * t$
  - Start substitution: $\langle\rangle$
  - Parallel and sequential composition: $\langle f, t/x\rangle, f; g$

$\lambda\sigma$-Calculus

Typing Rules:
Substitutions are judgements $\Gamma \vdash f : \Delta$

Key Idea 1: Every $\lambda\sigma$-term is equivalent to a λ-term.

Key Idea 2: Contexts $z : A \times B$ and $x : A, y : B$ are not equal

Categorical Semantics

Which explicit substitutions?

- which Term constructs to require?
- which Equations to require?
- how to discover terms/equations for different logics?

Methodology: Extend Curry Howard

- Cat theory as syntax debugging
- Internal language gives term constructs & equational theory

Key Ideas:

- ES includes the underlying λ-calculus
- Contexts $z : A \times B$ and $x : A, y : B$ isomorphic

Indexed Categories

- **Contexts:** a cartesian category
  - Objects are contexts, morphisms substitutions
  - structure is parallel composition
- **Terms:** For every context Γ, define the category $D(\Gamma)$

  Objects are variable-type pairs $x : A$

  Morphisms $(x : A) \longrightarrow (t : B)$ are judgements $\Gamma, x : A \vdash t : B$
- **Re-indexing:** $\Gamma \vdash f : \Delta$ induces a functor $D(f) : D(\Delta) \to D(\Gamma)$
- **Summary:** An indexed category is a functor $D : C^{op} \to Cat$

Indexed Categories

- **Plus Points:** Indexed categories induce $\lambda\sigma$-equations

$$
\begin{array}{lcll}
\langle\rangle ; h & = & h & \text{Identity of } C \\
h ; \langle\rangle & = & h & \text{Identity of } C \\
\langle\rangle * t & = & t & \text{Functoriality of } D \\
 & & & \\
(f ; g) ; h & = & f ; (g ; h) & \text{Assoc of } C\text{-comp} \\
(f ; g) * t & = & f * (g * t) & \text{Functoriality of } D
\end{array}
$$

- **Lemma:** Soundness and completeness

Indexed Categories

- **Problem 1:** Indexed Categories are inherently non linear
  - Identities in the fibres require judgements $\Gamma, x : A \vdash x : A$
- **Problem 2:** No syntax for forming substitutions from terms

Context-Handling Categories

- **Solution 1:** Remove identities by changing codomain to **Set**
- **Solution 2:** Add two new natural transformations
- **Defn:** Let $\mathcal{B}$ be a SMC (CCC) with $\mathcal{T} \subseteq |\mathcal{B}|$. A *linear (cartesian) context handling category* is a functor $L : \mathcal{B}^{op} \to Sets^{\mathcal{T}}$ and for each type, natural isomorphisms

  $\mathtt{Sub}_A : L(-)_A \Rightarrow \mathcal{B}(-, A)$ $\mathtt{Term}_A : \mathcal{B}(-, A) \Rightarrow L(-)_A$

- **Yoneda lemma:** There exists $\mathtt{Var}_A \in L(A)_A$ with $\mathtt{Term}_A(f) = f * \mathtt{Var}_A$

Context-Handling Categories

- **Equations:** We get the following equations

$$\begin{array}{lcll} \langle (f * t)/x \rangle & = & f; \langle t/x \rangle & \text{Naturality of Sub} \\ \langle t/x \rangle * x & = & t & \text{One half of the iso} \\ \langle (f * x)/x \rangle & = & f & \text{Other half of the iso} \end{array}$$

E-Categories

- **Type constructors:** require extra structure on the fibres,
- Given $A, B \in \mathcal{T}$, there are $A \to B, A \times B \in \mathcal{T}$. There are isomorphisms, natural in Γ

$$\frac{E((\Gamma, A))_B}{E(\Gamma)_{A \to B}} \qquad \frac{E(\Gamma)_A \times E(\Gamma)_B}{E(\Gamma_{A \times B})}$$

- **Naturality:** distributes explicit subs over terms

$$f * \lambda x.t = \lambda x.f * t \qquad f * (tu) = (f * t)(f * u)$$

- **Model Collapse:** Contexts $z : A \times B$ and $x : A, y : B$ are isomorphic

L-Categories

- Given $A, B \in \mathcal{T}$, there is a type $A \multimap B \in \mathcal{T}$ and isos, natural in Γ $\dfrac{E((\Gamma, A))_B}{E(\Gamma)_{A \multimap B}}$
- **Tensor** doesn't give an iso $z : A \otimes B \cong x : A, y : B$ $\dfrac{E((\Gamma, A, B))_C}{E(\Gamma, A \otimes B)_C}$
- **Ask:** $x : A, y : B \vdash \langle (x \otimes y)/z \rangle : A \otimes B$ has inverse.

$$\frac{\Gamma, x : A, y : B \vdash f : \Delta}{\Gamma, z : A \otimes B \vdash \mathtt{let}\ z\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ f : \Delta}$$

- Category theory guiding the design of the calculus

!-type Constructor

- **Benton:** ! as the comonad of a *monoidal adjunction* between CCCs and SMCCs
- Models of explicit substitutions
  - E cats model $\rightarrow, \times, 1$ types
  - L cats model $\multimap, \otimes, I$ types
- **Defn:** A $!L$-category is a monoidal adjunction where C is an E-cat, B is an L-cat and F, G preserve types.
- **isomorphism** $x : A|\_ \cong \_|z :!A$

$$\frac{\Gamma, x : A|\Delta \vdash f : \Delta}{\Gamma|\Delta, z :!A \vdash \texttt{let } z \text{ be } !x \texttt{ in } f : \Delta}$$

Categorical Theorems

3 theorems for E-cats, L-cats, two for E- and L-cat together into a !L-category

- (right defn) Every E-cat contains an underlying CCC and every CCC extends to an E-cat, in an iso way;
- (soundness) We model the $\lambda\sigma$ calculus in an E-cat;
- (completeness) If for every E-cat, the interpretation of two morphisms f, f' is the same, then the equality of the morphisms is proved by the $\lambda\sigma$ theory.

Similarly for L-categories.

For !L-categories, use Barber's DILL calculus with explicit substitutions xDILL.

First Conclusions

- Indexed categories are inherently non-linear, but we can use presheaves
- Explicit substitutions are tricky to model
- Our models reflect this by “taking isomorphisms seriously”
- We get an extension of Curry-Howard correspondence justifying our calculus
- Have a working indexed-cat model for linear, cartesian, both as well as the original for the Calculus of Constructions

Explicit substitutions are not simply a hack: find maths of clever implementation

ILT: no modality, better behaviour

- Better type theory for better implementations
- Which type theory? why better?
- Categorical Models for Linear and Intuitionistic Type Theory
- Maietti, Ritter, de Paiva, FOSSACS, LNCS Springer, vol 1784, 2000

Linear Type Theories

with categorical models

- Linear λ-calculus
  (Benton, Bierman, de Paiva, Hyland, 1992)
  https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-262.html
- DILL system (Plotkin and Barber, 1996)
  http://www.lfcs.inf.ed.ac.uk/reports/96/ECS-LFCS-96-347/
- LNL system (Benton, 1995)

Categorical Models

- Linear λ-calculus (ILL) A **linear category** is a smcc equipped with a (linear monoidal) comonad !, the co-Kleisli category of ! is a CCC. (loads of conditions)
- DILL system A **DILL-category** is a pair, a smcs and a cartesian category, related by a (symmetric) monoidal adjunction.
- LNL system An **LNL-model** is a (symmetric) monoidal adjunction between a smcc and a ccc.

Linear Curry-Howard Isos

"Relating Categorical Semantics for Intuitionistic Linear Logic" (with P. Maneggia, M. Maietti and E. Ritter), Applied Categorical Structures, vol 13(1):1–36, 2005.

Take home: Categorical models need to be more than sound and complete. Need to provide *internal languages* for the theories they model.

Why a new calculus?

(ILT, Fossacs 2000)

Choosing between the three type theories:

- DILL is best, less verbose than ILL, but closer to what we want to do than LNL.
- Easy formulation of the promotion rule
- Contains the usual lambda-calculus as a subsystem
- however, most FPers would prefer to use a variant of DILL where instead of !, one has two function spaces, $\to$ and $\multimap$
- But then what's the categorical model?

Intuitionistic and Linear Type Theory
(ILT, Calculus)

- If we have two function spaces, but no modality !, how can we model it?
- All the models discussed before have a notion of !, a comonad created by the adjunction.
- Well, we need to use deeper mathematics, i.e. **fibrations** or indexed categories.

Calculus ILT

$$\Gamma \mid a : A \vdash a : A \qquad \Gamma, x : A \mid \_ \vdash x : A$$

$$\frac{\Gamma \mid \Delta, a : A \vdash M : B}{\Gamma \mid \Delta \vdash \lambda a^{A}.M : A \multimap B} \qquad \frac{\Gamma \mid \Delta_1 \vdash M : A \multimap B \quad \Gamma \mid \Delta_2 \vdash N : A}{\Gamma \mid \Delta \vdash M \,{}_{l}N : B}$$

$$\frac{\Gamma, x : A \mid \Delta \vdash M : B}{\Gamma \mid \Delta \vdash \lambda x^{A}.M : A \to B} \qquad \frac{\Gamma \mid \Delta \vdash M : A \to B \quad \Gamma \mid \_ \vdash N : A}{\Gamma \mid \Delta \vdash M \,{}_{i}N : B}$$

$$\frac{\Gamma \mid \Delta_1 \vdash M : A \quad \Gamma \mid \Delta_2 \vdash N : B}{\Gamma \mid \Delta \vdash M \otimes N : A \otimes B}$$

$$\frac{\Gamma \mid \Delta_1 \vdash M : A \otimes B \quad \Gamma \mid a : A, b : B, \Delta_2 \vdash N : C}{\Gamma \mid \Delta \vdash \mathtt{let}\ M\ \mathtt{be}\ a \otimes b\ \mathtt{in}\ N : B}$$

$$\frac{\Gamma \mid \Delta \vdash M : A \quad \Gamma \mid \Delta \vdash N : B}{\Gamma \mid \Delta \vdash (M, N) : A \& B} \qquad \frac{\Gamma \mid \Delta \vdash M : A \& B}{\Gamma \mid \Delta \vdash \mathsf{Fst}(M) : A} \qquad \frac{\Gamma \mid \Delta \vdash M : A \& B}{\Gamma \mid \Delta \vdash \mathsf{Snd}(M) : B}$$

$$\frac{}{\Gamma \mid \emptyset \vdash \bullet : I} \qquad \frac{\Gamma \mid \Delta_1 \vdash M : I \quad \Gamma \mid \Delta_2 \vdash N : C}{\Gamma \mid \Delta \vdash \mathtt{let}\ M\ \mathtt{be}\ \bullet\ \mathtt{in}\ N : C}$$

$$\frac{}{\Gamma \mid \Delta \vdash \circ : 1}$$

Where applicable Δ_1, Δ_2 are disjoint and Δ is a permutation of Δ_1, Δ_2.

Table 1. The typing rules of ILT

ILT Models

- Lawvere's hyperdoctrines satisfying comprehension but
  needs to capture the difference between intuitionistic and linear variables
- A base category B models intuitionistic contexts of ILT
- Each fibre over an object in B modelling a context models terms $\Gamma|\Delta \vdash M : A$ for any context Δ
- The fibres are smccs with tensor products and model the linear constructions of ILT

Internal language theorems

Just as in the previous ‘example’ can prove soundness, completeness and internal language theorems.
Missing an understanding of what is essential, what can be changed, and limitations of the methods.

Conclusions

- Explicit substitutions issues:
  - equations-in-context vs. raw reductions
  - de Bruijn indices vs. variable names
  - untyped vs. typed calculi
  - calculus for abstract machines or for proof assistants, etc..
- our calculus has been driven by the correspondence with the cat semantics
- Ritter showed that every reduction used in abstract machines terminates, thus SN is not necessary for their design.
- still: which other logics can we do?

Thanks!

Intuitionistic Type Theory

Curry-Howard Correspondence Triangles

Lambda-calculus

1963

Cartesian Closed Categories

Intuitionistic Propositional Logic

1965

Linear Type Theory

Curry-Howard Correspondence

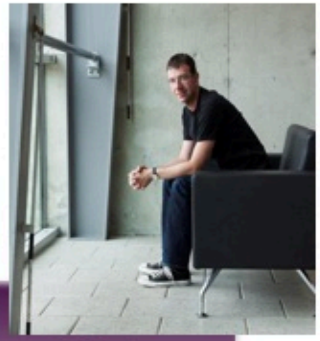


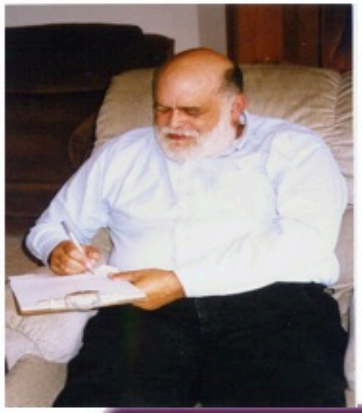
Linear Lambda-Calculus


Linear Categories

Linear Logic

Modal (S4) Curry-Howard Correspondence





Modal S4 Lambda-Calculus

CCC+monoidal comonad

Constructive S4 Modal Logic

Lambek calculus Curry-Howard Correspondence


2-sided Linear Lambda-Calculus

Monoidal bi-closed Categories

Lambek Calculus

Special model: Gödel Dialectica

- **Goal** Prove HA consistent. How?
- **Idea:** Translate every formula A of HA to

$$A^D = \exists u \forall x A_D$$

  where A_D quantifier-free.
- Translation defined by clauses on the connectives.
- 'Easy' to prove the theorem desired, but hard to see **why** it works.

Gödel Dialectica

Theorem (Gödel 1958): if HA proves A, then System T proves quantifier-free $A_D(t, x)$, where x are functionals of finite type, and t a suitable sequence of terms (not containing x).

Proof by induction on length of derivations, Troelstra 1973.

How intuitive are the functionals of finite type?

An internal **categorical model of Gödel's Dialectica interpretation in my phd thesis.**

Categorical Dialectica

Given C with finite limits, build a new category $\mathfrak{Dial}(\mathrm{C})$, with objects $A = (U, X, \alpha)$ where α is a subobject of $U \times X$ in C; this object represents the formula

$$\exists u \forall x \alpha(u, x).$$

A map from $\exists u \forall x \alpha(u, x)$ to $\exists v \forall y \beta(v, y)$ can be thought of as a pair $(f : U \to V, F : U \times Y \to X)$ of terms/maps, subject to the entailment condition

$$\alpha(u, F(u, y)) \vdash \beta(f(u), y).$$

Surprise! A model of Linear Logic, instead of Constructive Logic

Dialectica categories

- Justifies Linear Logic in terms of Gödel's proof-theoretic tool. and conversely.
- Keep the differences that Girard wanted to make.
- Justifies Harper's Trinitarism, connections to programming and using CT as syntax guidance.
- Loads of applications, lenses, games, automata, etc.

Dialectica categories timeline

- 1940 Gödel lecture at Yale
- 1958 published in Dialectica
- 1988 first categorical interpretation
- 2008 fibrational generalization (Biering)
- 2011 modern version (Hofstra)
- 2018 dependent type theory (von Glehm, Moss)

Recent work with Trotta and Spadetto where the assumptions in Gödel's argument (hacks?) are used (2022, 2023)

Applications of Dialectica

- Concurrency theory, Petri nets and others (1990's)
- linear functional programming (2000's)
- partial compilers (Budiu and Plotkin, 2013)
- Lenses, BX-transformations, *Lenses for Philosophers*, Hedges 2017
- Automated Differentiation, Pedrot and Kerjean 2022, LICS2024

Applied Category Theory?



- ”Applied Category Theory in Chemistry, Computing, and Social Networks” MRC 2022
- David Corfield ”Philosophy and Innovation”, nLab 2024
- Bradley, Tai-Danae (2018) ”What is Applied Category Theory?”
- and many more

Category theory and its uses

Corfield, 2024

Category theory and its uses Applications (with approximate dates)

- Mathematics (from 1940s)
- Logic (from 1960s)
- Computing (from 1970s)
- Physics (from 1980s)
- *Applied Category Theory* (from the 2010s)

ACT Philosophy perspective

Corfield, 2024

- Initially a convenient language: ‘standard’ mathematician does not approve of CT
- Now not only a language?

  We would like to say that our proofs are proofs by “formal nonsense” and in particular analysis-free. (Clausen and Scholze)

ACT Philosophy perspective

Corfield, 2024

In Physics:

- Topological quantum field theory (1980s)
- String diagrams as morphisms in monoidal categories for quantum mechanics (2000s)
- Higher gauge theory, 2000s
- Modal and linear homotopy type theory
- The Quantum Monadology, Schreiber

`https://ncatlab.org/davidcorfield/files/NUL.pdf`

ACT Baez et al: How?

Society is increasingly complex and connected through the internet and social media, planetary climate and ecological challenges, transnational organization and global supply chains. To navigate and thrive in this networked world, we rely on scientific advances to help us manage this complexity by enabling robust communication, cooperation, and collaboration.

"Applied Category Theory in Chemistry, Computing, and Social Networks" Notices of AMS (Baez, Cicala, Cho, Otter, de Paiva) 2022

ACT: themes

- compositionality
- functorial semantics,
- implementing these structures into user-friendly software.

a bird’s eye view of themes
logic/programming languages only one of these

ACT: more themes

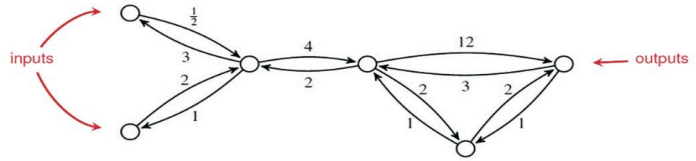


Figure 1. Open Markov process.

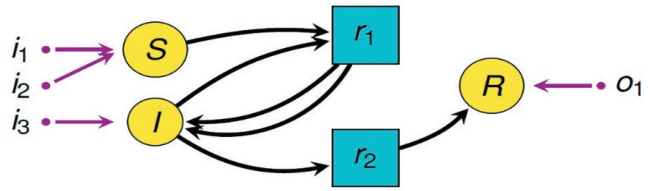


Figure 2. Open SIR model as a Petri net.

- Markov processes
- chemical reaction networks
- Epidemiology
- Petri nets
- Stock and flow diagrams

MRC: outcomes

- Baez's group: X. Li, S. Libkind, N. D. Osgood, E. Patterson, "Compositional modeling with stock and flow diagrams", 5th International Conference ACT, EPTCS 380 (2022), 77–96.
- J. C. Baez, X. Li, S. Libkind, N. D. Osgood and E. Redekopp, "A categorical framework for modeling with stock and flow diagrams", (to appear)
- "Agent-Based Models (Parts 1,2)" blog post, J.C. Baez

MRC: outcomes

- de Paiva's group: J. Dorta, S. Jarvis, and Nelson Niu. "Monoidal structures on generalized polynomial categories". May 2023. ACT 2023, arXiv:2305.05655
- "On a fibrational construction for optics, lenses, and Dialectica categories", M. Capucci, B. Gavranović, A. Malik, F. Rios, J. Weinberger, to appear in ACT2024.
- "The dependent Goedel fibration", D. Trotta, J. Weinberger, V de Paiva, abstract CT2023 Jonathan Weinberger

Much more ACT

- People applying category theory for:
- Causality, probabilistic reasoning, statistics, learning theory, deep neural networks, dynamical systems, information theory, database theory, natural language processing, cognition, consciousness, active inference, systems biology, genomics, epidemiology, chemical reaction networks, neuroscience, complex networks, game theory, robotics, quantum computing,...

Much more ACT

- higher category theory
- Categorical probability theory
- Categorical differentiation
- category theory for AI? geometric deep learning, categorical deep learning
- Deep Learning for Theorem Proving
  `https://arxiv.org/pdf/2404.09939`

Conclusions

- Applied Category Theory now!
- (still) underappreciated (categorical) Curry-Howard correspondence
- Important for interdisciplinary work: Math, Logic and Programming
- favorite example: Dialectica categories, Gödel fibrations and doctrines, rediscovered over and over
- Plenty of other examples/applications to develop

Thanks again!

Some References

- K. Gödel, *Über eine bisher noch nicht benützte erweiterung des finiten standpunktes*, In Dialectica, 12(3-4):280–287. (Translation in Gödel's Collected Works)
- J.-Y. Girard, *Linear Logic*, TCS (1988).
- P. Wadler, *Types as Propositions*, Communications of the ACM, pp 75–84, (2015).
- V. de Paiva, *The Dialectica Categories*, In Proc of Categories in Computer Science and Logic, Boulder, CO, 1987.
- D. Trotta, M. Spadetto, V. de Paiva, *The Gödel Fibration*, MFCS, 2021.
- D. Trotta, M. Spadetto, V. de Paiva, *Dialectica Logical Principles: not only rules*, JLC 2022
- D. Trotta, M. Spadetto, V. de Paiva, *Dialectica Principles via Gödel Doctrines*, TCS, 2023.