It's the end of the world as we know it

and I feel fine

Alberto Momigliano

LFMTP, July 24, 2026

It's the end of the world as we know it

and I feel ~~fine~~ funny

Alberto Momigliano

LFMTP, July 24, 2026

Roadmap

A tale in two parts

- Part 1 How did we get here? Logical frameworks, benchmarks, and the rise of AI.
- Part 2 What remains difficult? The metatheory of substructural systems.

Part 1: A look back and forward

In the beginning

Robert Harper, Furio Honsell, Gordon D. Plotkin: *A Framework for Defining Logics.* LICS 1987

The first two logical frameworks workshops:

- 1991: *Logical Frameworks*, CUP, edited by Huet & Plotkin
- 1993: *Logical Environments*, CUP, same editors

Not to rain on this parade, but check out these high-rollers:

> *Aczel, de Bruijn, Basin, Pfenning, Burstall, Felty, Coquand, Dybjer, Pym, Constable, Schroeder-Heister, Berardi, Bertot, Nipkow, Goguen(s), Parigot*

Lots of groundbreaking theory, case studies in the single digits.

After a gap

The *LFM* workshop series restarted in 1999 and alternated with

- *Merλin* " Mechanized reasoning about languages with variable binding", started by Ambler, Crole and myself in 2001 (name due to Frank Pfenning "because it's kind of black magic")
- They merged in 2006 yielding *LFMTP*
- In the same year *LSFA* was created as an adjacent but broader sibling with a South American focus

Many celeb contributors, from the "founders" to the second generation. Same mix of theory and case studies, but lately the workshop has been struggling to attract contributions.

The arc of LFM workshops

From the foundations of logical frameworks to:

- mechanized reasoning about binding to:
- proof-assistant ecosystems for PL metatheory.

What's next?

The arc of LFM workshops

From the foundations of logical frameworks to:

- mechanized reasoning about binding to:
- proof-assistant ecosystems for PL metatheory.

What's next? Is there a "next"? Let's look to one way our field has tried to progress:

The arc of LFM workshops

From the foundations of logical frameworks to:

- mechanized reasoning about binding to:
- proof-assistant ecosystems for PL metatheory.

What's next? Is there a "next"? Let's look to one way our field has tried to progress:

Benchmarks/Challenges.

Benchmarks: what are they good for?

Absolutely nothing?

Benchmarks are proposed to gauge and advance the state of the art

- POPLMark [2005] /Ademyr et al.)
- List-machine [2006/2012] (Appel, Dockins and Leroy)
- Benchmarks for Reasoning with with Syntax Trees Containing Binders and Contexts of Assumption [2017] (Felty, A.M. and Pientka)
- POPLMark Reloaded [2017/2019] (Abel, Allais, Hameer, AM, Pientka, Schäfer, Stark)
- Concurrent Calculi Formalisation Benchmark [2024] (Carbone et al.)

How useful have benchmarks been in our field?

Absolutely nothing?

Benchmarks are proposed to gauge and advance the state of the art

- POPLMark [2005] /Ademyr et al.)
- List-machine [2006/2012] (Appel, Dockins and Leroy)
- Benchmarks for Reasoning with with Syntax Trees Containing Binders and Contexts of Assumption [2017] (Felty, A.M. and Pientka)
- POPLMark Reloaded [2017/2019] (Abel, Allais, Hameer, AM, Pientka, Schäfer, Stark)
- Concurrent Calculi Formalisation Benchmark [2024] (Carbone et al.)

How useful have benchmarks been in our field?

Define **useful**

If “useful” means citations (source: Scholar)

- POPLMark Challenge: some 460
- A list-machine benchmark for mechanized metatheory: 18 (a rare miss by Appel & Leroy)
- Benchmarks for Reasoning with with Syntax Trees Containing Binders and Contexts of Assumptions: 19
- POPLMark Reloaded: 76
- The Concurrent Calculi Formalisation Benchmark: 9

Don’t take citations too seriously.

POPLMark: the original dream

"How close are we to a world where every paper on programming languages is accompanied by an electronic appendix with machine-checked proofs?"

- **Stress:** binding, complex induction, experimentation with representations, and reuse, using $F_{<:}$.
- **Method:** give PL researchers and proof-assistant developers a shared problem on which to compare approaches.
- **Ambition:** make machine-checked appendices routine—"mechanized metatheory for the masses."

Five benchmarks, five roles

Benchmark	Main stress	Dominant role
POPLMark	Binders and complex induction	A meeting place and measuring instrument: compare approaches and make mechanization mainstream.
List-machine	Executable semantics, extraction, proof–program links	An engineering acceptance test for realistic compiler verification.
ORBI	Structured contexts and context relations	A design specification for future metalanguages and proof environments.
POPLMark Reloaded	Kripke logical relations, open terms, simultaneous substitutions	Stress-test systems and libraries; expose missing abstractions and implementation weaknesses.
Concurrent calculi	Linearity, scope extrusion, coinduction	Community infrastructure: consolidate techniques and extend mechanization to concurrency.

Impact of benchmarks on proof assistants

- Abella & Beluga appeared after POPLMark, but have their foundations and motivations in work from the late 90's
  - The ∇ quantifier? Yeah, but again, it comes the proof theory of generic quantification
  - Similar remarks for Isabelle/Nominal (OK, it's a library) stemming from Gabbay's PhD thesis [1999]
- All the shades of Agda?
  - Lots of movement on the coinduction side, which, ironically enough, was not a challenge till 2024
  - Cubical Agda? Neat, but we know where it comes from and it ain't the benchmarks

Impact, ctd.

- POPLMark was solved in many systems and with various techniques, but
  1. did PA implemented new features to addresses the challenges?
  2. Have new techniques for handling binders been developed for that?

Impact, ctd.

- POPLMark was solved in many systems and with various techniques, but
  1. did PA implemented new features to addresses the challenges?
  2. Have new techniques for handling binders been developed for that?
- Very few indeed
  1. Abella switched from “uniform proofs” to “focusing”, possibly motivated by the third order Twelf solution to POPLMark by Pientka.

Impact, ctd.

- POPLMark was solved in many systems and with various techniques, but
  1. did PA implemented new features to addresses the challenges?
  2. Have new techniques for handling binders been developed for that?
- Very few indeed
  1. Abella switched from “uniform proofs” to “focusing”, possibly motivated by the third order Twelf solution to POPLMark by Pientka. (This is a contentious story – ask me after a drink or two)
  2. According to “Reloaded”, bugs were found in Beluga and fixed

Libraries: the durable infrastructure

- POPLMark correlates with the emergence of libraries for reasoning about binders: *Nominal, Hybrid, Metalib, LNGen, LN, Autosubst, Knot, DBLib*.
- Many underlying ideas predate the benchmarks; partial exceptions include *generic syntax, SOAS*, and *Tealeaves*, with more category-theoretic origins.
- Libraries have been a major enabling technology over the past twenty years. Nevertheless

Libraries: the durable infrastructure

- POPLMark correlates with the emergence of libraries for reasoning about binders: *Nominal, Hybrid, Metalib, LNGen, LN, Autosubst, Knot, DBLib*.
- Many underlying ideas predate the benchmarks; partial exceptions include *generic syntax, SOAS*, and *Tealeaves*, with more category-theoretic origins.
- Libraries have been a major enabling technology over the past twenty years. Nevertheless

  "Our experience of applying the LN approach as advertised was more painful than we had anticipated [. . .] So we cannot, hand-on-heart, recommend the vanilla LN style for anything but small, kernel language development." (*Rossberg, Russo, and Dreyer, F-ing Modules, JFP 2014*)

What did benchmarks actually change?

Indirectly: quite a lot

- Shared problems, vocabulary, and comparison points.
- Tutorials, regression tests, and reusable developments.
- Adoption of mechanization and formation of communities.
- Pressure for binder and context infrastructure.

My reading

Benchmarks have been more successful as *coordination devices* than as requirements documents for proof assistants.

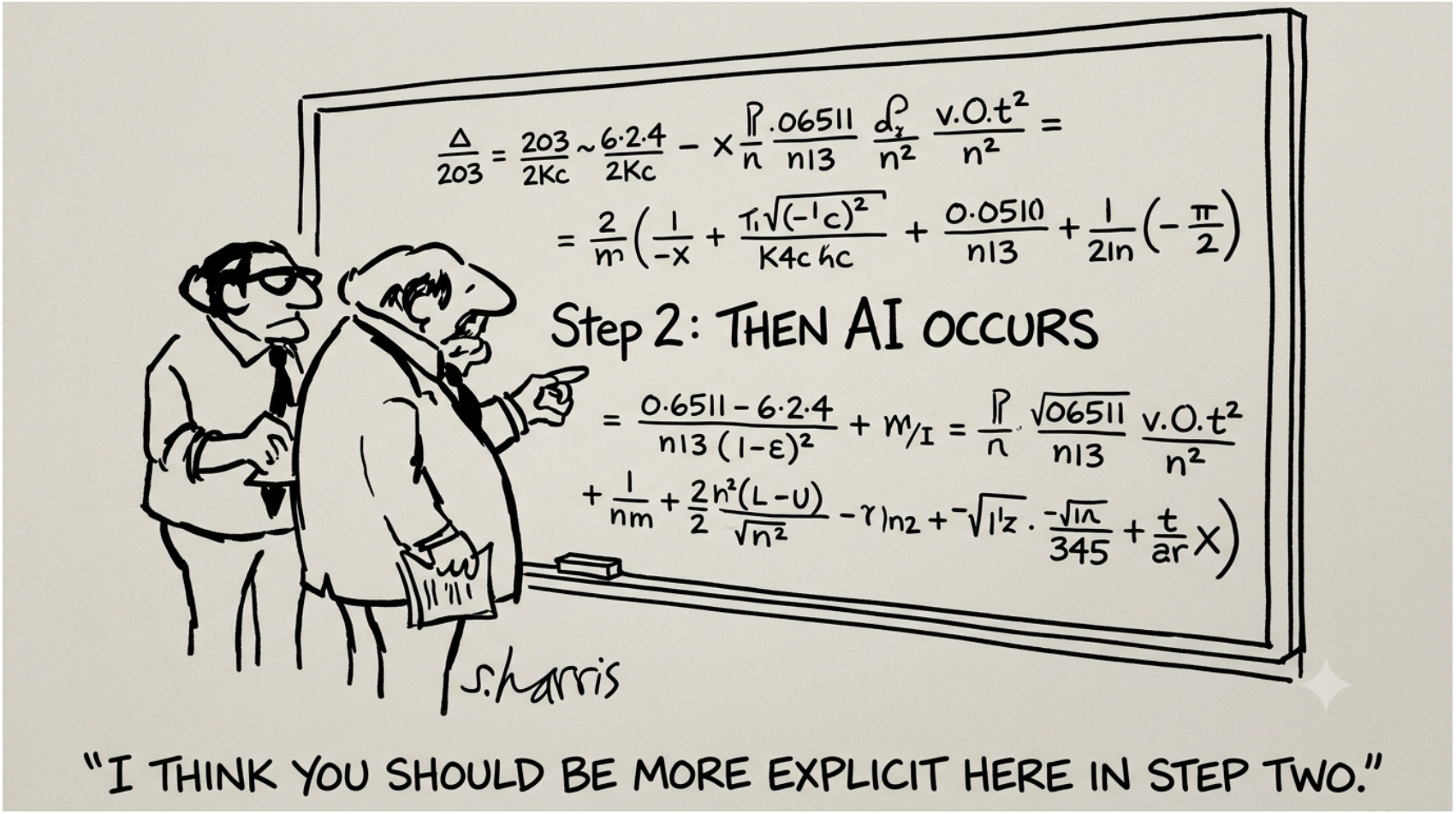
What's next

The accepted wisdom

> *"Despite extensive research both on the theoretical and practical fronts, formalising, reasoning about, and implementing languages with variable binding is still a daunting endeavour — repetitive boilerplate and the overly complicated metatheory of capture-avoiding substitution often get in the way of progressing on to the actually interesting properties of a language."* *Formal Metatheory of Second-Order Abstract Syntax, by Fiore and Szamozvancev, POPL 2022*

Until a few months ago, I would have agreed wholeheartedly.

The Rise of the Machines



Modified from the original Sidney Harris cartoon.

From a bemused observer

- I'm just a vanilla user of AI, but even I can foresee the synergies between AI and proof assistants
- Check out Leo de Moura's blog and the *Signal Shot* project aiming to verify the Signal protocol and its Rust implementation in Lean.
- See some recent experience reports closer to our field:
  - "Machine-Generated, Machine-Checked Proofs for a Verified Compiler" (Zoe Paraskevopoulou, Feb. 2026)
  - "AI-Assisted Completion of CertiGC Proofs" (Shengyi Wang, June 2026)
- There are also failures:
  building an unverified compiler with coding agents:

  > *"Four agents spent 14 days and 93,000 lines of Lean building a verified JS-to-WASM compiler from scratch. The compiler ran; the proofs didn't close."*

Anecdata (in my circle)

- Lanese & Sacerdoti are using `Aristotle.ai` to formalize papers about reversible computations in Lean: apparently, pretty good for meta-theorems, sucks for establishing bisimilarity
- MC had Claude do in Lean the soundness proofs of a submission to a certain PL conference (some 40*k* lines)
- Same for GV for his submission (Gemini in Agda)

While top-level researchers, none of the latter two were *power users* of the chosen PA and would have taken months to do it, if at all.

Are we therefore close to POPLMark's dream?

> *"a future where the papers in conferences such as POPL and ICFP are routinely accompanied by mechanically checkable proofs of the theorems they claim."*

Anecdata (in my circle)

- Lanese & Sacerdoti are using `Aristotle.ai` to formalize papers about reversible computations in Lean: apparently, pretty good for meta-theorems, sucks for establishing bisimilarity
- MC had Claude do in Lean the soundness proofs of a submission to a certain PL conference (some 40*k* lines)
- Same for GV for his submission (Gemini in Agda)

While top-level researchers, none of the latter two were *power users* of the chosen PA and would have taken months to do it, if at all.

Are we therefore close to POPLMark's dream?

> *"a future where the papers in conferences such as POPL and ICFP are routinely accompanied by mechanically checkable proofs of the theorems they claim."*

Or is that some kind of nightmare?

Elegance is not optional.

- I spent some 20 years arguing about the superiority of HOAS w.r.t. more concrete approaches:
  - You do not need to worry about α equivalence
  - You get substitution from β reduction (look ma, no shifting)
  - Weakening and substitution lemmas for free!
- Sure, you have to work in some niche proof assistant with almost zero library, but it's worth it.

Elegance is ~~not~~ optional.

- I spent some 20 years arguing about the superiority of HOAS w.r.t. more concrete approaches:
  - You do not need to worry about α equivalence
  - You get substitution from β reduction (look ma, no shifting)
  - Weakening and substitution lemmas for free!
- Sure, you have to work in some niche proof assistant with almost zero library, but it's worth it.
- Is it?

Elegance is ~~not~~ optional.

- I spent some 20 years arguing about the superiority of HOAS w.r.t. more concrete approaches:
  - You do not need to worry about α equivalence
  - You get substitution from β reduction (look ma, no shifting)
  - Weakening and substitution lemmas for free!
- Sure, you have to work in some niche proof assistant with almost zero library, but it's worth it.
- Is it?
- Why don't I just throw *tokens* at it? What's the pain of proving an ugly substitution lemma, if the agent does it for me?

Elegance is ~~not~~ optional.

- I spent some 20 years arguing about the superiority of HOAS w.r.t. more concrete approaches:
  - You do not need to worry about α equivalence
  - You get substitution from β reduction (look ma, no shifting)
  - Weakening and substitution lemmas for free!
- Sure, you have to work in some niche proof assistant with almost zero library, but it's worth it.
- Is it?
- Why don't I just throw *tokens* at it? What's the pain of proving an ugly substitution lemma, if the agent does it for me?
- Why do I even need a binder library if the agent can encode my system in whatever syntax I fancy?

Elegance is ~~not~~ optional.

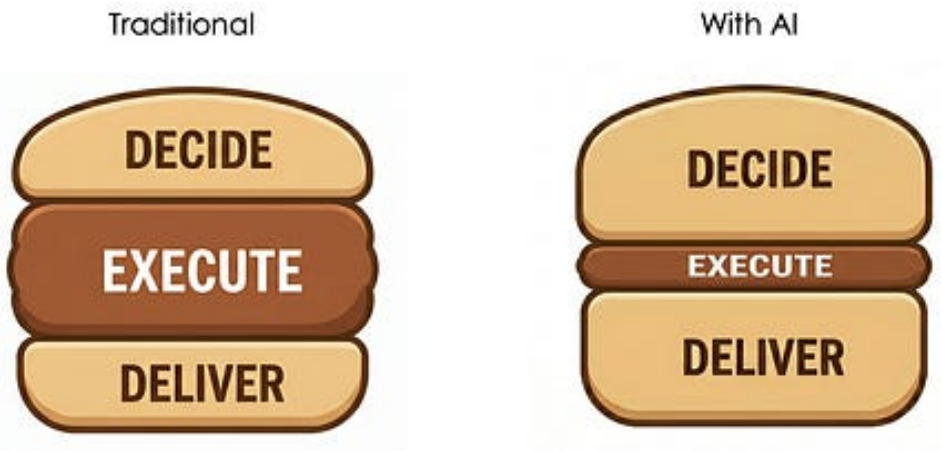
- I spent some 20 years arguing about the superiority of HOAS w.r.t. more concrete approaches:
  - You do not need to worry about α equivalence
  - You get substitution from β reduction (look ma, no shifting)
  - Weakening and substitution lemmas for free!
- Sure, you have to work in some niche proof assistant with almost zero library, but it's worth it.
- Is it?
- Why don't I just throw *tokens* at it? What's the pain of proving an ugly substitution lemma, if the agent does it for me?
- Why do I even need a binder library if the agent can encode my system in whatever syntax I fancy?

$\Rightarrow$ Elegance may become optional; adequacy, auditability, and semantic understanding will not.

The decide-execute-deliver sandwich in SE

The programmer vs coder controversy in software engineering:

- The decide layer: understanding customer requirements, developing the specification, planning, etc.
- The actual coding and debugging
- Understanding code enough to be accountable for its release



Effort shifts from rowing the boat to steering the ship and figuring out where we even want to go [. . .] The floor (AI capability) keeps moving up but so does the ceiling (AI-assisted human capability). (Arvind Narayanan)

The enhanced games

What's next for a workshop such as LFMTP?

- **Enhanced** LFMTP: you can freely submit your (?) agent-generated, machine-checked artifact

The enhanced games

What's next for a workshop such as LFMTP?

- **Enhanced** LFMTP: you can freely submit your (?) agent-generated, machine-checked artifact
  - Who will review it?
  - What is the intended audience?
  - What's the contribution?
- A formalization paper should not be a contribution in itself unless it offers
  - a new specification or adequacy result;
  - a reusable abstraction or library;
  - a mechanization technique;
  - a comparison that changes our understanding of systems;
  - a counterexample or defect in the paper theory ...

This is a general problem in CS and Mathematics, but we are at the forefront.

Benchmarks after agents

Can we re-think the very idea of benchmarks?
A future benchmark should not ask merely whether an agent can close the proof, but test:

- whether the formal statement matches the intended mathematics;
- whether hidden axioms, holes, vacuity, or inconsistent assumptions are detected;
- whether a human can understand it and how much expert intervention is needed in specification and invariant discovery.
- whether the method transfers across representations or assistants

. . .

Mission accomplished?

No technology lasts forever, and that is a sign of progress, not failure. The goal is not to build a system that endures indefinitely [. . .] If something genuinely superior emerges, that means the mission succeeded. (Leo de Moura)

Mission accomplished?

> *No technology lasts forever, and that is a sign of progress, not failure. The goal is not to build a system that endures indefinitely [. . .] If something genuinely superior emerges, that means the mission succeeded. (Leo de Moura)*

If, as the old Gauls feared, the sky is about to fall on mechanized metatheory in the shape of the AI apocalypse, we proceed from the same old Gaulish response:

Mission accomplished?

> *No technology lasts forever, and that is a sign of progress, not failure. The goal is not to build a system that endures indefinitely [. . .] If something genuinely superior emerges, that means the mission succeeded. (Leo de Moura)*

If, as the old Gauls feared, the sky is about to fall on mechanized metatheory in the shape of the AI apocalypse, we proceed from the same old Gaulish response:

tomorrow never comes

What remains difficult?

If agents make proof construction cheap, it is still up to us to choose representations, specifications, invariants, and abstractions that make the *right* theorem formulable, checkable, and understandable.

The design problem

The interesting contribution is often not completing more proofs, but choosing a representation in which much of the proof bureaucracy disappears.

Substructural metatheory is a good test case.

Part 2
formalizing the metatheory of substructural systems

Joint work with: Marco Carbone, Ryan Kavanagh, Brigitte Pientka, Chuta Sano, Daniel Zackon

Why substructural systems

- Substructural (*e.g.* linear) logics restrict weakening, contraction, and exchange, prohibiting variables from being duplicated, discarded or reordered by default, then selectively re-introducing these capabilities where needed
- Resource-sensitive computations are common in quantum computing, type systems, memory management, *etc.*
- Some substructural logics:

Logic	Affected rules	Assumption usage
Linear	W,C omitted	exactly once
Strict	W omitted	at least once
Affine	C omitted	at most once
Graded	W,C controlled	usage governed by a grade
Ordered	W,C,E omitted	in order, exactly once
...	...	...

Why substructural systems

- Substructural (*e.g.* linear) logics restrict weakening, contraction, and exchange, prohibiting variables from being duplicated, discarded or reordered by default, then selectively re-introducing these capabilities where needed
- Resource-sensitive computations are common in quantum computing, type systems, memory management, *etc.*
- Some substructural logics:

Logic	Affected rules	Assumption usage
Linear	W,C omitted	exactly once
Strict	W omitted	at least once
Affine	C omitted	at most once
Graded	W,C controlled	usage governed by a grade
Ordered	W,C,E omitted	in order, exactly once
...	...	...

The elusive linear logical (meta)framework

- There are several formalizations of LL, but no consensus on how to formalize the meta-theory of substructural object logics.
- Since the late 1990s, Cervesato & Pfenning and McDowell & Miller argued for the need of such a framework, with compelling examples: SR for MiniML with references, cut elimination for LL
  ...

The elusive linear logical (meta)framework

- There are several formalizations of LL, but no consensus on how to formalize the meta-theory of substructural object logics.
- Since the late 1990s, Cervesato & Pfenning and McDowell & Miller argued for the need of such a framework, with compelling examples: SR for MiniML with references, cut elimination for LL
  ...
- It never really materialized

The elusive linear logical (meta)framework

- There are several formalizations of LL, but no consensus on how to formalize the meta-theory of substructural object logics.
- Since the late 1990s, Cervesato & Pfenning and McDowell & Miller argued for the need of such a framework, with compelling examples: SR for MiniML with references, cut elimination for LL
  ...
- It never really materialized
- Lots of valuable theory, several logic programming engines (LLF, Lolli, Forum, LolliMon, Celf...), but no cigar
  hey, I'm still waiting for Abella/Forum :-)

The elusive linear logical (meta)framework

- There are several formalizations of LL, but no consensus on how to formalize the meta-theory of substructural object logics.
- Since the late 1990s, Cervesato & Pfenning and McDowell & Miller argued for the need of such a framework, with compelling examples: SR for MiniML with references, cut elimination for LL
  ...
- It never really materialized
- Lots of valuable theory, several logic programming engines (LLF, Lolli, Forum, LolliMon, Celf...), but no cigar
  hey, I'm still waiting for Abella/Forum :-)
- Partial exceptions: the *SELL* family, the 2-level approach in *Hybrid*
- Successful alternative: *Iris* and friends (*LinearActris, Affect*):
  a different, semantic approach, with very successful engineering

Splitting considered harmful

The issue is context management; take a *multiplicative* rule such as the following in a purely linear lambda calculus:

$$\frac{\Delta \vdash M : A \multimap B \qquad \Delta' \vdash N : A}{\Delta ; \Delta' \vdash M\,N : B}$$

where ; is some non-deterministic split operation

- Prevailing approach to modeling substructural contexts: explicitly "rip contexts apart"
- Encodings close to on-paper formulation, but
  - Whatever datatype you use (lists, multisets, finite maps), you need a library of low-level lemmas.
  - Complex index management when using de Bruijn representations of syntax: For simultaneous substitutions must split contexts *and* substitutions

Thou shalt not split

The linearity predicate Keep an ordinary structural context; track each assumption's use with a separate predicate on terms or derivations.

Contexts as resource vectors Keep the context shape fixed; record availability or usage with algebraic annotations on its entries.

The linearity predicate way

- Karl Crary in 2010 (simplifying early attempts by Avron et al. [1989], Pfenning [1994]) found a cunning way to add linearity etc. to LF w/o changing the framework
- The key move: : substructurality is not a property of **contexts** but of **assumptions**
- You can sprinkle a **linearity predicate** over the typing rules stating when a hypothesis is used only once, on an assumption-by-assumption basis
- The typing system stays structural, hence you still get all the goodies from HOAS

The linearity predicate way

- Karl Crary in 2010 (simplifying early attempts by Avron et al. [1989], Pfenning [1994]) found a cunning way to add linearity etc. to LF w/o changing the framework
- The key move: : substructurality is not a property of **contexts** but of **assumptions**
- You can sprinkle a **linearity predicate** over the typing rules stating when a hypothesis is used only once, on an assumption-by-assumption basis
- The typing system stays structural, hence you still get all the goodies from HOAS

Cons:

- you need a proof term on which to define the predicate
- does not directly handle systems where substructurality is genuinely a context property, such as noncommutative systems

Example: the linear lambda calculus

Twelf Specification

```
of : tm -> tp -> type.

of/llam: of (llam ([x] M x)) (lolli A B)
           <- ({x} of x A -> of (M x) B)
%% NEW!     <- linear ([x] M x).
of/lapp: of (lapp M N) B
           <- of M (lolli A B)
           <- of N A.

linear : (tm -> tm) -> type.

lin/llam : linear ([x] llam ([y] M x y))
            <- ({y} linear ([x] M x y)).
lin/lapp1: linear ([x] lapp (M x) N)
           <- linear M.
lin/lapp2: linear ([x] lapp M (N x))
           <- linear N.
lin/var : linear ([x] x).
```

Standard Typing Rules

$$\frac{}{\Gamma; x : A \vdash x : A}$$

$$\frac{\Gamma; (\Delta, x : A) \vdash M : B}{\Gamma; \Delta \vdash \lambda x.M : A \multimap B}$$

$$\frac{\Gamma; \Delta \vdash M : A \multimap B \qquad \Gamma; \Delta' \vdash N : A}{\Gamma; (\Delta, \Delta') \vdash MN : B}$$

The unusual effectiveness of the linearity predicate

- Adequacy of typing is a bit more involved, but for $\Gamma; \Delta \vdash M : B$ you need to show additionally $\mathrm{lin}(x, M)$ for all $x \in \mathrm{dom}(\Delta)$,
- To prove results such as type preservation for a lambda calculus in a HOAS setting, you need to establish:
  - Linearity is preserved under function composition
  - Some strengthening lemmas
  - Linearity is preserved under the operational semantics
- The rest is smooth sailing, like in structural HOAS.

The unusual effectiveness of the linearity predicate

- Adequacy of typing is a bit more involved, but for $\Gamma; \Delta \vdash M : B$ you need to show additionally $\mathrm{lin}(x, M)$ for all $x \in \mathrm{dom}(\Delta)$,
- To prove results such as type preservation for a lambda calculus in a HOAS setting, you need to establish:
  - Linearity is preserved under function composition
  - Some strengthening lemmas
  - Linearity is preserved under the operational semantics
- The rest is smooth sailing, like in structural HOAS.

Beyond HOAS?

- The no-splitting move avoids many hassles of concrete representations, especially w.r.t. simultaneous substitutions.
- Fits well with intrinsically typed reps.

The linearity predicate and session types

$$\begin{array}{lcl} P, Q & ::= & 0 \mid x!.P \mid x?.P \mid x!y.P \mid x?(y).P \mid P \parallel Q \mid (\boldsymbol{\nu} xy)\ P \\ S, T & ::= & \mathbf{end}_! \mid \mathbf{end}_? \mid ?T.S \mid !T.S \end{array}$$

Note: We need to support resource splitting, consumption, and update

T-INACT
$$\frac{}{\cdot \vdash 0}$$

T-PAR
$$\frac{\Delta_1 \vdash P \qquad \Delta_2 \vdash Q}{\Delta_1, \Delta_2 \vdash P \parallel Q}$$

T-RES
$$\frac{\Delta, x : T, y : \overline{T} \vdash P}{\Delta \vdash (\boldsymbol{\nu} xy)\ P}$$

T-WAIT
$$\frac{\Delta \vdash P}{\Delta, x : \mathbf{end}_? \vdash x?.P}$$

T-CLOSE
$$\frac{\Delta \vdash P}{\Delta, x : \mathbf{end}_! \vdash x!.P}$$

T-SEND
$$\frac{\Delta, x : U \vdash P}{\Delta, x : !T.U, y : T \vdash x!y.P}$$

T-RECV
$$\frac{\Delta, x : U, y : T \vdash P}{\Delta, x : ?T.U \vdash x?(y).P}$$

The linearity predicate and session types

$$\begin{array}{rcl} P, Q & ::= & 0 \mid x!.P \mid x?.P \mid x!y.P \mid x?(y).P \mid P \parallel Q \mid (\boldsymbol{\nu} xy)\ P \\ S, T & ::= & \mathbf{end}_! \mid \mathbf{end}_? \mid ?T.S \mid !T.S \end{array}$$

Note: We need to support resource splitting, consumption, and update

T-INACT

$$\frac{}{\cdot \vdash 0}$$

T-PAR

$$\frac{\Delta_1 \vdash P \qquad \Delta_2 \vdash Q}{\Delta_1, \Delta_2 \vdash P \parallel Q}$$

split

T-RES

$$\frac{\Delta, x : T, y : \overline{T} \vdash P}{\Delta \vdash (\boldsymbol{\nu} xy)\ P}$$

T-WAIT

$$\frac{\Delta \vdash P}{\Delta, x : \mathbf{end}_? \vdash x?.P}$$

T-CLOSE

$$\frac{\Delta \vdash P}{\Delta, x : \mathbf{end}_! \vdash x!.P}$$

T-SEND

$$\frac{\Delta, x : U \vdash P}{\Delta, x : !T.U, y : T \vdash x!y.P}$$

T-RECV

$$\frac{\Delta, x : U, y : T \vdash P}{\Delta, x : ?T.U \vdash x?(y).P}$$

The linearity predicate and session types

$$\begin{array}{rcl} P, Q & ::= & 0 \mid x!.P \mid x?.P \mid x!y.P \mid x?(y).P \mid P \parallel Q \mid (\boldsymbol{\nu} xy)\ P \\ S, T & ::= & \mathbf{end}_! \mid \mathbf{end}_? \mid ?T.S \mid !T.S \end{array}$$

Note: We need to support resource splitting, consumption, and update

T-INACT

$$\frac{}{\cdot \vdash 0}$$

T-PAR

$$\frac{\Delta_1 \vdash P \qquad \Delta_2 \vdash Q}{\Delta_1, \Delta_2 \vdash P \parallel Q}$$

split

T-RES

$$\frac{\Delta, x : T, y : \overline{T} \vdash P}{\Delta \vdash (\boldsymbol{\nu} xy)\ P}$$

T-WAIT

$$\frac{\Delta \vdash P}{\Delta, x : \mathbf{end}_? \vdash x?.P}$$

consume

T-CLOSE

$$\frac{\Delta \vdash P}{\Delta, x : \mathbf{end}_! \vdash x!.P}$$

T-SEND

$$\frac{\Delta, x : U \vdash P}{\Delta, x : !T.U, y : T \vdash x!y.P}$$

T-RECV

$$\frac{\Delta, x : U, y : T \vdash P}{\Delta, x : ?T.U \vdash x?(y).P}$$

The linearity predicate and session types

$$\begin{array}{rcl} P, Q & ::= & 0 \mid x!.P \mid x?.P \mid x!y.P \mid x?(y).P \mid P \parallel Q \mid (\boldsymbol{\nu} xy)\ P \\ S, T & ::= & \mathbf{end}_! \mid \mathbf{end}_? \mid ?T.S \mid !T.S \end{array}$$

Note: We need to support resource splitting, consumption, and update

T-INACT

$$\frac{}{\cdot \vdash 0}$$

T-PAR

$$\frac{\Delta_1 \vdash P \qquad \Delta_2 \vdash Q}{\Delta_1, \Delta_2 \vdash P \parallel Q}$$

split

T-RES

$$\frac{\Delta, x : T, y : \overline{T} \vdash P}{\Delta \vdash (\boldsymbol{\nu} xy)\ P}$$

T-WAIT

$$\frac{\Delta \vdash P}{\Delta, x : \mathbf{end}_? \vdash x?.P}$$

consume

T-CLOSE

$$\frac{\Delta \vdash P}{\Delta, x : \mathbf{end}_! \vdash x!.P}$$

T-SEND

$$\frac{\Delta, x : U \vdash P}{\Delta, x : !T.U, y : T \vdash x!y.P}$$

update

T-RECV

$$\frac{\Delta, x : U, y : T \vdash P}{\Delta, x : ?T.U \vdash x?(y).P}$$

The session situation

- Option 1: change the syntax and semantics of the calculus, using *continuations*, (as in SCP from Sano [2024])
  This is a must if working with HOAS *implicit* contexts
- Option 2: encode contexts explicitly.
  - Then, you may as well use a mainstream proof assistant.
  - The linearity predicate lets us keep the domain of the explicit context structurally intact.
  - it works great with well-scoped dB (see “Mechanizing the Meta-Theory of Session Types in Rocq: a Tutorial”, with Marco Carbone).

The linearity predicate on processes, first order rep

In conjunction with session typing, the predicate enforces exclusive endpoint ownership and hence rules out races on linear endpoints.

$$\textsc{L-Wait}\ \frac{\neg\mathsf{free\_in}(x, P)}{\mathrm{lin}(x, x?.P)} \qquad \textsc{L-WaitCgr}\ \frac{\mathrm{lin}(y, P) \qquad x \neq y}{\mathrm{lin}(y, x?.P)} \qquad \textsc{L-Res}\ \frac{\mathrm{lin}(z, P) \qquad z \notin \{x, y\}}{\mathrm{lin}(z, (\boldsymbol{\nu} xy)\ P)}$$

$$\textsc{L-ParPL}\ \frac{\mathrm{lin}(x, P_1) \qquad \neg\mathsf{free\_in}(x, P_2)}{\mathrm{lin}(x, P_1 \parallel P_2)} \qquad \textsc{L-ParPR}\ \frac{\neg\mathsf{free\_in}(x, P_1) \qquad \mathrm{lin}(x, P_2)}{\mathrm{lin}(x, P_1 \parallel P_2)}$$

$$\textsc{L-RecvCgr}\ \frac{\mathrm{lin}(z, P) \qquad z \neq y}{\mathrm{lin}(z, x?(y).P)}$$

. . . other obvious rules omitted

ST with the linearity predicate

$$\frac{}{\Delta \Vdash_l 0} \qquad \frac{\Delta \Vdash_l P \qquad \Delta \Vdash_l Q}{\Delta \Vdash_l P \parallel Q}$$

$$\frac{\Delta, x : T, y : \overline{T} \Vdash_l P \quad \text{free\_in}(x, P) \Rightarrow \text{lin}(x, P) \quad \text{free\_in}(y, P) \Rightarrow \text{lin}(y, P)}{\Delta \Vdash_l (\boldsymbol{\nu} xy)\ P}$$

$$\frac{\Delta(x) = \mathbf{end}_? \quad \Delta \Vdash_l P}{\Delta \Vdash_l x?.P} \qquad \frac{\Delta(x) = \mathbf{end}_! \quad \Delta \Vdash_l P}{\Delta \Vdash_l x!.P}$$

$$\frac{\Delta(x) = !T.U \quad \Delta(y) = T \quad \Delta + x : U \Vdash_l P}{\Delta \Vdash_l x!y.P}$$

$$\frac{\Delta(x) = ?T.U \quad (\Delta + x : U), y : T \Vdash_l P \quad \text{lin}(y, P)}{\Delta \Vdash_l x?(y).P}$$

- The ambient typing judgment remains structural; binder rules discharge the condition for newly bound endpoints locally.
- Contexts are functions; “+” is just functional update.

The recipe and its ingredients

1. Distinguish principal-channel cases from congruence cases.
2. If the construct binds any new linear endpoint, its typing rule must check its linearity by requiring the appropriate predicate in the premise(s) of the rules (restriction and input typing rules).
3. For a construct that prohibits shared endpoints, use the predicate to express that the resource occurs in at most one branch—for example, parallel composition.

In a first-order representation, there is additional boilerplate, such as explicit handling of free names and preservation of linearity under injective substitutions.

Other applications of LP

- Linearizing **Howe**- style proof of congruence of applicative bisimilarity, answering Roy Crole's question: "How linear is Howe"

Other applications of LP

- Linearizing **Howe**- style proof of congruence of applicative bisimilarity, answering Roy Crole's question: "How linear is Howe" Answer: not much (Codex & I in Abella) — just add the predicate in the binder cases of the candidate relation.
  This applies to other logical relation proofs

Other applications of LP

- Linearizing **Howe**- style proof of congruence of applicative bisimilarity, answering Roy Crole's question: "How linear is Howe" Answer: not much (Codex & I in Abella) — just add the predicate in the binder cases of the candidate relation.
  This applies to other logical relation proofs
- "Deconstructed Proto-Quipper: A Rational Reconstruction" (Kavanagh, Sano, Pientka, PPDP'26). This generalizes the linearity predicate to a mode-safety predicate for adjoint systems and uses it to prove normalization.

What are modes? That's another talk.

Contexts as resource vectors or: tagging along

- CARVe [Zackon *et al.*, 2025]: track resource usage while keeping contexts structurally "intact":
  - entries are annotated with multiplicity tags, so that resource availability is tracked by updating tags rather than modifying the structure of the context
- The idea appears in Walker's chapter in "Advanced Topics in Types and Programming Languages,", was formalized in Twelf by Schack-Nielsen & Schürmann, 2010, and is now everywhere in graded modal and quantitative type theories.

Example (Linear sequent calculus)

$$\dfrac{\dfrac{}{A, A \multimap B \vdash A}\,\text{hyp} \qquad \dfrac{}{A, A \multimap B, B \vdash B}\,\text{hyp}}{A, A \multimap B \vdash B}\,\multimap\text{L}$$

The "gray" assumptions are tagged as unavailable

Tagged Contexts

- Assumptions annotated with elements α of some algebraic structure capturing usage

$$\Delta ::= \cdot \mid \Delta, x :^{\alpha} A$$

- Pointwise resource composition, rather than structural splitting, is induced by the multiplication of the chosen algebra:

$$\frac{}{\cdot \bowtie \cdot = \cdot} \qquad \frac{\Delta_1 \bowtie \Delta_2 = \Delta \qquad \alpha_1 \circ \alpha_2 = \alpha}{(\Delta_1, x :^{\alpha_1} A) \bowtie (\Delta_2, x :^{\alpha_2} A) = \Delta, x :^{\alpha} A}$$

- The (partial) commutative monoid for linear logic $\mathscr{L} = (\{0,1\}, \bullet, 0)$

$\bullet$	0	1
0	0	1
1	1	–

- Again, compatible with de Bruijn indices and simultaneous substitution

Two ways not to split

	Linearity predicate	**Resource annotations**
Where accounting lives	A separate predicate on terms or derivations	Tags attached to context entries
Context shape	Ordinary structural context	Fixed-shape annotated context
Branching	Predicate clauses ensure that a resource occurs in the permitted branch(es)	Pointwise algebraic composition distributes availability across premises
Main proof burden	Composition, pruning, and preservation lemmas for the predicate	Algebraic laws and substitution lemmas for annotated contexts

Neither method commits to HOAS or to de Bruijn syntax. The predicate is especially attractive in HOAS encodings; resource vectors have been used in both Beluga and Rocq.

Sumatra (not the island)

- CARVe initially implemented in Beluga
- Then ported to Rocq, to make it parametric w.r.t. the resource algebra
- Still space for a further generalization:

Sumatra (not the island)

- CARVe initially implemented in Beluga
- Then ported to Rocq, to make it parametric w.r.t. the resource algebra
- Still space for a further generalization: Enter *Sumatra*: **Su**bstructural **M**etatheory with **A**djoin**t** **R**esource **A**nnotations—a Rocq library inspired by adjoint logic and developed in Daniel Zackon's forthcoming PhD thesis.
- Why don't we talk about it during the coffee break?

Conclusions

- Benchmarks have given us shared problems and communities better than giving us new foundations.
- Coding agents may realize the POPLMark dream of routine machine-checked appendices by making proof production cheap but often ugly.
- The research contribution moves towards specification, adequacy, representation, reusable abstraction, and auditability.
- Linearity predicates and resource annotations share the same design move: keep context shape stable and relocate resource accounting.

Take-home message.

AI will not immediately make logical framework research obsolete; it changes which part of it is intellectually valuable.

Conclusions