Four Lectures on Proof-theoretic Semantics

Midlands Graduate School in the Foundations of Computing Science Sheffield, April 2025

David Pym
UCL Computer Science and UCL Philosophy
Institute of Philosophy, School of Advanced Study
University of London

Schedule

1. What is Proof-theoretic Semantics (P-tS)?
   - Inferentialism.
   - Consequence.
   - Proof-theoretic Validity (P-tV).
2. Base-extension Semantics (B-eS):
   - B-eS for Intuitionistic Propositional Logic.
   - Naturality, categorically speaking.
   - B-eS and P-tV.

Schedule

3. Reductive Logic, Tactical Proof, and Logic Programming:
   - Reductive Logic and P-tV.
   - Tactical Proof.
   - Remarks on Logic Programming, B-eS, and Coalgebra.
4. Modal and Substructural Logics, Resource Semantics, and Modelling:
   - B-eS for Modal Logics.
   - B-eS for Substructural Logics.
   - Resource Semantics and Modelling with B-eS.

Schedule

- Most of what we will introduce will be quite new to most people, with a fairly significant philosophical basis, and with quite a lot of ground to be covered.
- Our approach will mainly be conceptual, with little detailed, formal proof.
- Nevertheless, the formal details of everything we cover are available in books and papers that will be referenced.

Lecture 3: Reductive Logic, Tactical Proof, and Logic Programming

I want to say some quite simple — but also rather speculative — things, trying to make some connections between

- reductive logic,
- tactical proof construction,
- proof-theoretic semantics, and
- coalgebraic semantics, as a possible unifying theory.

Reductive Logic: Proof Theory and Proof-search

Here, we have the most basic idea: read inference rules as *reduction operators*, from conclusion to premisses. Instead of the *deduction*

$$\frac{\text{Premiss}_1 \quad \ldots \quad \text{Premiss}_k}{\text{Conclusion}} \Downarrow,$$

the *reduction*

$$\frac{\text{Sufficient Premiss}_1 \quad \ldots \quad \text{Sufficient Premiss}_k}{\text{Putative Conclusion}} \Uparrow$$

Here failure to construct a proof derives from failure to reduce to axiom sequents, say. For example,

$$\frac{\Gamma \vdash \mathrm{p}}{\ldots} \Uparrow$$

and there is no unpacking of Γ that exposes an occurrence of p.

Reductive Logic: Kripke Semantics and 'Model-checking'

Model-theoretic satisfaction relations also work like this, of course:

$$w \models_{\mathcal{M}} \phi \wedge \psi \quad \text{iff} \quad w \models_{\mathcal{M}} \phi \text{ and } w \models_{\mathcal{M}} \psi$$

$$w \models_{\mathcal{M}} \phi * \psi \quad \text{iff} \quad \text{there are worlds } u \text{ and } v \text{ s.t. } R(u, v, w) \text{ and } u \models_{\mathcal{M}} \phi \text{ and } v \models_{\mathcal{M}} \psi$$

and so on.

Here failure to construct a realizer —that is, a transmitter of truth — derives from failure to reduce to satisfiable atoms:

$$w \models_{\mathcal{M}} \mathrm{p} \quad \text{iff} \quad w \in \mathcal{V}(\mathrm{p})$$

That is, we reduce to atoms that do not satisfy this condition.

Reductive Logic

$$\begin{array}{lrcl} \text{hereditary Harrop} & D & := & A \mid D \wedge D \mid G \supset A \\ \text{formulae (hHfs)} & G & := & A \mid G \wedge G \mid D \supset G \mid G \vee G \end{array}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\mathcal{P} \vdash G'} \quad \dfrac{}{A' \vdash A'}\,Ax}{\mathcal{P} \vdash A'}\; G' \supset A' \in \mathcal{P}}{\vdots\ \text{right rules}}}{\mathcal{P} \vdash G} \quad \dfrac{}{A \vdash A}\,Ax}{\mathcal{P} \vdash A}\; G \supset A \in \mathcal{P}$$

Resolution, that is uniform proofs with just the resolution rule (essentially, $\supset L$) the only left rule, can be seen as being complete for hHfs.

Reductive Logic

- Logic programming standardly comes with a least fixed point semantics.
- Interpret programs as sets of atoms (i.e., as subsets of the Herbrand universe).
- Form a complete lattice of such interpretations.
- Define an operator on such interpretations that corresponds to the resolution step above.
- The meaning of a program is given by the interpretation that is the least fixed point of operator.

This construction is intimately connected to Sandqvist's completeness theorem. See Alexander Gheorghiu and David Pym. Definite formulae, Negation-as-Failure, and the Base-extension Semantics of Intuitionistic Propositional Logic. *Bulletin of the Section of Logic*, 2023.

Reductive Logic

Towards a semantic perspective:

- Let's focus on proofs for now, mainly think of IPL and CPL for now.
- In reductive logic, we encounter a space — of 'reductions' — that is larger than the space of proofs. And non-proofs are interesting, useful, and mathematically well structured.
- Semantic structures must account for this. And also for control in proof-search — many choices are encountered in performing reductions.
- Let's unpack this a bit, following Pym & Ritter, Oxford Logic Guide 2004, has a detailed set-up for classical and intuitionistic logic.

- When reading the rules of calculus as reduction operators — let's think for now of LJ, the sequent calculus for intuitionistic logic — the key point is that when constructing a reduction, we may encounter leaves of the form

$$\frac{}{p_1 \ldots p_k \ ?\!\!-\ \ q} \quad q \text{ not any } p_i$$

- No further reductions possible.
- In terms of proof-objects, we don't have

$$x_1 : p_1, \ldots, x_k : p_k \vdash x_i : p_i$$

- Instead, we must introduce an indeterminate

$$x_1 : p_1, \ldots, x_k : p_k \vdash \alpha : q$$

Why is This Useful?

- Why is considering reductions that are proofs useful?
- Because in practical reasoning
  - such things are encounter a great deal, and
  - they can be used to determine proofs.
- Proof-search examples:
  - In first-order predicate logic, relax the side-conditions on the quantifier rules.
  - In substructural logic, calculate the distribution of formulae in multiplicative rules *not locally*, but *globally*.
- All this can be expressed in terms of P-tV, as we shall later see.

Theoretical Backstory (not for the faint-hearted)

- A *reduction model* is a fibred structure $\mathcal{R}$ — in the sense of the use of fibred / indexed categories in categorical logic — interpreting propositions / proofs relative to indeterminates, which stand for *terms and propositions* that remain to be calculated — these will correspond to *open proofs* in the Dummett-Prawitz-Schroeder-Heister view recalled below.
- From Pym and Ritter (2004):

$$\mathcal{R} = \langle (\mathcal{E}, F), [\![-]\!], \models \rangle$$

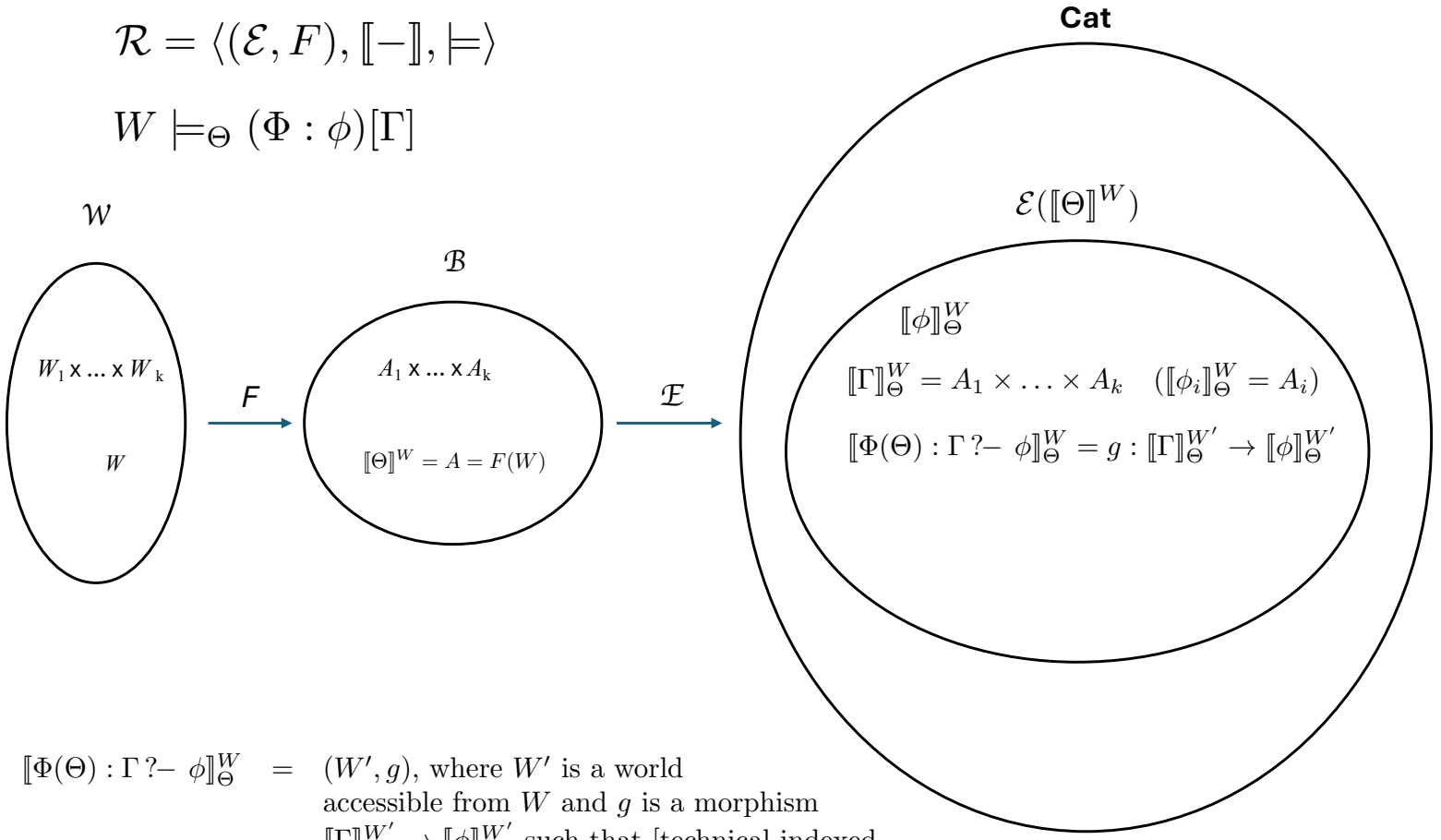
where

  - $\mathcal{E} : \mathcal{B}^{op} \to \mathbf{Cat}$, where $\mathcal{B}$ — the category of indeterminates — has finite products, each $\mathcal{E}(B)$ is a biCCC, and each $\mathcal{E}(f)$ preserves biCC structure on the nose
  - $F : \mathcal{W} \to \mathcal{B}$ preserves finite products (for technical reasons)
  - $[\![-]\!]$ interprets reduction operators and reductions
  - $\models$ is a satisfaction relation, with familiar properties.

Theoretical Backstory

$$\mathcal{R} = \langle (\mathcal{E}, F), [\![-]\!], \models \rangle$$

$$W \models_\Theta (\Phi : \phi)[\Gamma]$$



$[\![\Phi(\Theta) : \Gamma \,?\!-\, \phi]\!]^W_\Theta$ = (W', g), where W' is a world accessible from W and g is a morphism $[\![\Gamma]\!]^{W'}_\Theta \to [\![\phi]\!]^{W'}_\Theta$ such that [technical indexed category condition]

Theoretical Backstory

- Along with this, we obtain a semantic judgement, defined relative to the model:

$$W \models_{\Theta} (\Phi : \phi)\Gamma$$

  between worlds W, indeterminates in Θ, sequents $\Gamma \mathrel{?\text{-}} \phi$, and reductions, Φ.
- This yields a consequence relation in the evident way.
- Picks up many judgements of interest.
- This judgement asserts that at stage W in a model that interprets indeterminates Θ, Φ is a reduction for the putative conclusion $\Gamma \mathrel{?\text{-}} \phi$, which may be written as $\Gamma \mathrel{?\text{-}} \Phi : \phi$.

Theoretical Backstory

- In this 'truth-functional' sense, *soundness* means that all Γ ?- ϕ for which a reduction can be calculated are true in the model and *completeness* means that there is a (term) reduction model for which all true Γ ?- ϕ have reductions.
- Again, Pym & Ritter, Oxford Logic Guide 2004, has the detail for IPL and CPL.

Theoretical Backstory

- A reduction Φ is interpreted as a map

$$[\![\Gamma]\!]^W_\Theta \xrightarrow{[\![\Phi]\!]^W_\Theta} [\![\Delta]\!]^W_\Theta$$

- *Soundness* means that every reduction that can be calculated can be so interpreted in the model.
- *Completeness* means that there is a (term) model consisting of exactly the reductions that can be calculated.
- Both are stated relative to the judgement

Theoretical Backstory (Aside)

In this denotational setting, we can also seek to interpret not only the realizer of a consequence but the control process.

- For example, Prolog's strategy of left-to-right clause selection with depth-first traversal and cut, and the input-output model.
- A control process is associated with the realizer Φ, constructed using the process E:

$$[\![\Gamma]\!]^W_\Theta \overset{[\![E:\Phi]\!]^W_\Theta}{\rightarrow} [\![\Delta]\!]^W_\Theta$$

- There is a well-developed theory of (bi)simulation (equality) of processes.
- We can explore examples of game-theoretic models that are able to account for both the structural and operational aspects of reductive logic.
- Again, Pym & Ritter, Oxford Logic Guide, 2004. Many open questions remain in this area.

Milner and LCF: A Concrete Theory of Proof-search in Reductive Logic

In the late 70s and early 80s, Robin Milner and colleagues worked on machine-assisted proof, producing the 'Stanford LCF' and 'Edinburgh LCF' systems.

This work was about 'goal-oriented reasoning' — that is, proof-search.

- R. Milner. The use of machines to assist in rigorous proof. *Phil. Trans. R. Soc. Lond.* A 312, 411–422 (1984).
- M. Gordon. Tactics for mechanized reasoning: a commentary on Milner (1984) 'The use of machines to assist in rigorous proof'. *Phil. Trans. R. Soc. Lond.* A 373:20140234 (2015).

Goals, Theorems, and Procedures

Following Milner:

- A *theorem* is a (proved) sequent $\Gamma \vdash \phi$
- A *goal* G is a sequent Γ ?– ϕ

Then:

- An *event* E is [the proving of] a theorem
- An event $\Delta \vdash \psi$ *achieves* a goal Γ ?– ϕ if, for some $\Theta \subseteq \Gamma$, Θ ?– $\phi \quad \simeq \quad \Delta$?– ψ, for some equivalence (generalizing Milner a bit here).
- A *procedure* is a partial function

$$\rho : (\text{list of theorems}) \rightharpoonup \text{theorem}$$

Tactics

- A *tactic* is a partial function that takes a goal and returns a list of goals and a procedure:

$$\text{tactic} : \text{goal} \rightharpoonup \text{goal list} \times \text{procedure}$$

- Elementary tactics are given by the the reduction operators that correspond to the 'inverses' of inference rules

$$\frac{\text{Premiss}_1 \quad \ldots \quad \text{Premiss}_k}{\text{Conclusion}} \Downarrow$$

$$\frac{\text{Subgoal}_1 \quad \ldots \quad \text{Subgoal}_k}{\text{Goal}} \Uparrow$$

- A tactic T is *valid* if, whenever

$$T(G) = ([G_1, \ldots G_n)], \rho)$$

is defined and whenever $[E_1, \ldots, E_n]$ respectively achieve the goals $[G_1, \ldots, G_n]$, then the event $\rho([E_1, \ldots, E_n])$ achieves G.

Tacticals

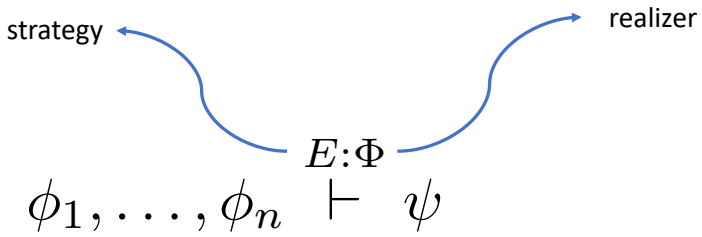
- Complex goals require, in practice, complex strategies.
- Need combinators, called *tacticals*, for composing tactics.
- Tactical combinations of tactics are themselves tactics.
- Examples would include:
  - Basic sequencing
  - The definition of *uniform proof* in the sense of Miller
  - In some sense, this is the origin of the ML ('Meta-Language') family of programming languages.

Relate to Theoretical Backstory

- Recall

$$[\![\phi_1, \ldots, \phi_n]\!]^W_\Theta \xrightarrow{[\![E:\Phi]\!]^W_\Theta} [\![\psi]\!]^W_\Theta$$

- Here, abusing notation a bit



- A strategy is a tactical combination of tactics
- A procedure converts/reduces realizers to proofs, working up to $\simeq$.

Relate to Proof-theoretic Semantics

We can see, informally for now, a correspondence between Milner's analysis and current ideas in proof-theoretic semantics.

Let's work with something like the (Dummett-)Prawitz, Schroeder-Heister, Piecha , ... , recalling the approach that we introduced in Lecture 1.

- *Proof structures* $\mathcal{S}$ that are tree-like arrangements of *sequents*.
- A *justification*

$$\mathbf{J} : \mathcal{S} \to \mathcal{S}$$

  that maps structures to structures.
- An 'open' / 'closed' proof is a proof depending / not depending on assumptions, respectively.
- Don't need to be concerned with 'canonical' proofs for now.

Relate to Proof-theoretic Semantics

- Let S be a base.
- Every closed proof in S is $\langle \mathbf{J}, S \rangle$-valid.
- An open proof is $\langle \mathbf{J}, S \rangle$-valid if every substitution instance by closed proofs is $\langle \mathbf{J}, S \rangle$-valid.
- Recall that we get a notion of consequence

$$\phi_1, \ldots, \phi_n \models_{\langle \mathbf{J}, S \rangle} \psi$$

  if there is an open proof $\mathcal{D}$ of ψ from $\phi_1, \ldots, \phi_n$ such that all closed substitution instances of $\mathcal{D}$ are $\langle \mathbf{J}, S \rangle$-valid.
- Then we get *logical consequence* as follows:

$$\phi_1, \ldots, \phi_n \models \psi$$

  iff $\phi_1, \ldots, \phi_n \models_{\langle \mathbf{J}, S \rangle} \psi$ holds for every S.

Relate to Proof-theoretic Semantics

- Validity of proofs with respect to **J** and atomic system S: denote by

$$\langle \mathbf{J}, S \rangle \models \phi$$

  that **J** generates a closed proof of ϕ w.r.t S

- Then

$$\phi_1, \ldots, \phi_n \models \psi \quad \text{iff} \quad \begin{array}{l} \text{there is a } \mathbf{J} \text{ s.t. for every } S \\ \text{and all } \mathbf{J}_1, \ldots, \mathbf{J}_n, \text{ if} \\ (\mathbf{J}_1, S) \models \phi_1, \ldots, (\mathbf{J}_n, S) \models \phi_n, \\ \text{then } (\mathbf{J}, S) \models \psi \end{array}$$

- $\mathbf{J}(\mathbf{J}_1, \ldots, \mathbf{J}_n)$ amounts to the procedure, relative to possibly generalized S ('ground' inferences).

Interpretation in Proof-theoretic Semantics

Very roughly:

initial goal	$\phi_1, \ldots, \phi_n \models_{\langle \mathbf{J}, \mathcal{S} \rangle} \phi$
a sequence of tactics	a proof-like structure
a correct argument	canonical proof
procedure	justification

This has been set up properly in:

A. Gheorghiu and D. Pym. Proof-theoretic semantics and tactical proof. Manuscript at `https://arxiv.org/abs/2301.02302`.

- A. Gheorghiu and D. Pym. Semantic Foundations of Reductive Reasoning. To appear, Topoi, 2025.

Back to Proof-search

- The reductive view of logic is just as declarative as the deductive view.
- Proof-search, however, is operational and inherently stateful.
- Tactical proof is a declarative presentation of a stateful idea — a minor tweak makes this explicit.
- Historically, mathematics has not been all that good at handling stateful things in elegant ways.
- Relatively recently — in the great scheme of history — the concept of *coalgebra* has emerged to fill this gap.

Coalgebra

Coalgebra can be seen as the algebraic generalization of coinduction.

Essentially, final coalgebras stand in the same relation to coinduction as initial algebras to induction.

- A *coalgebra* for an endofunctor $F : \mathcal{C} \to \mathcal{C}$ — an 'F-coalgebra' — is a morphism $\alpha : X \to FX$ in $\mathcal{C}$, usually written (X, α).
- Intuitively, F assigns structure to a state space X, while α describes the dynamics/structural decomposition for a system that traverses this structured space.
- This concept subsumes and generalizes phenomena as wide-ranging as automata, context-free grammars, datatypes, games, program semantics, and transition systems.

Coalgebra

- In fact, Kripke semantics can be seen coalgebraically.
- Example: BI, the logic of bunched implications (O'Hearn & Pym, BSL 1999):
  - Essentially, freely combines IL and MILL in a bunched proof-theoretic framework.
  - Name comes from sequent calculus: bunched contexts, separating intuitionistic and linear parts.
  - Very different logic from LL: for example, $\phi \to \psi = !\phi \multimap \psi$ does not hold.
- It doesn't matter that it's BI — though we will revisit BI in B-eS in Lecture 4.
- For now, just a useful example of using coalgebra before we move to proof-search/construction.

Coalgebra: How to Represent a Logic

- BI's fully general semantics is based on ternary relations or on Grothendieck topologies, allowing the full metatheory. That's a long story.
- For now, let's take an ordered monoid $(R, \sqsubseteq, \circ, e)$ of worlds, r, s, t, ...

$$
\begin{array}{rcl}
r \models \mathrm{p} & \text{iff} & r \in \mathcal{V}(\mathrm{p}) \\
r \models \bot & \text{never} & \\
r \models \top & \text{always} & \\
r \models \phi \vee \psi & \text{iff} & r \models \phi \text{ or } r \models \psi \\
r \models \phi \wedge \psi & \text{iff} & r \models \phi \text{ and } r \models \psi \\
r \models \phi \to \psi & \text{iff} & \text{for all } s \sqsubseteq r,\ s \models \phi \text{ implies } s \models \psi \\
 & & \\
r \models I & \text{iff} & r \sqsubseteq e \\
r \models \phi * \psi & \text{iff} & \text{there are worlds } s \text{ and } t \text{ such that} \\
 & & r \sqsubseteq (s \circ t) \text{ and } s \models \phi \text{ and } t \models \psi \\
r \models \phi \mathrel{-\!\!*} \psi & \text{iff} & \text{for all } s \text{ such that } (r \circ s) \text{ and } s \models \phi, \\
 & & r \circ s \models \psi
\end{array}
$$

Truth-functional Semantics, Coalgebraically

- BI can be given by coalgebras for the functor $T : \mathcal{C} \to \mathcal{C}$,

$$TX = \mathbb{2} \times \mathsf{P}_c(X \times X) \times \mathsf{P}_c(X^{op} \times X),$$

  where $\mathcal{C}$ is the category of posets, $\mathbb{2}$ the two element poset and P_c the convex powerset functor (Egli–Milner order).
- The first component interprets of the unit constant I, the second $*$, and the third $-\!\!*$.
- Given a monoid $(R, \circ, e)$, a poset is given by setting $r \sqsubseteq s$ iff there exists r' such that $r \circ r' = s$.
- Then the coalgebra $\alpha : R \to \mathbb{2} \times \mathsf{P}_c(R \times R) \times \mathsf{P}_c(R^{op} \times R)$ is:
  - $\pi_0(\alpha(r))$ is 1 if $r \sqsubseteq e$ and 0 otherwise — for I
  - $\pi_1(\alpha(r)) = \{(s,t) \mid s \circ t \sqsubseteq r\}$ — for the conjunction $*$
  - $\pi_2(\alpha(r)) = \{(s,t) \mid r \circ s = t\}$ — for the implication $-\!\!*$.

- The coalgebraic interpretation of the logic is given, essentially, by a natural transformation δ from a functor that forms the formulae of the BI to the functor T.
- In the specific case of $*$, given interpretations for ϕ and ψ, we obtain the interpretation

$$\delta_X(\phi * \psi) = \{u \in TX \mid \exists (x, y) \in \pi_1(u), x \in \delta_X(\phi), y \in \delta_X(\psi)\}.$$

- In the coalgebra associated to a monoid, this corresponds precisely to the given truth-functional clause for $*$, but the class of coalgebraic models strictly extends the class of truth-functional models.

Proof-search With Substructural Connectives

- Having fixed the basic idea, let's look at proof-search.
- From the computational perspective, the reduction operators — the tactics, and so the tacticals — that correspond to the inference rules for multiplicative connectives, such as $\otimes$ and $\multimap$, and $*$ and $\mathrel{-\!\!*}$, are problematic.
- For example,

$$\frac{\Delta_1 \vdash \psi \quad \Delta_2 \vdash \chi}{\Gamma \vdash \psi * \chi} \quad \Gamma = \Delta_1, \Delta_2$$

- How to calculate the Δs?
- Iterates through the search: suppose $\psi = \psi_1 * \psi_2$, then need $\Delta = \Delta_1, \Delta_2$ ... .
- Computationally expensive (potentially both time and space).
- Not just right rules,

$$\frac{\Gamma_1 \vdash \phi \quad \Gamma_2, \psi \vdash \chi}{\Gamma, \phi \mathrel{-\!\!*} \psi \vdash \chi} \quad \Gamma = \Gamma_1, \Gamma_2$$

The input–output model

$$\dfrac{\begin{array}{l}\Gamma \backslash \Delta \vdash \phi_1 \\ \vdots \\ \Gamma \vdash \phi_1 \end{array} \qquad \Delta \vdash \phi_2}{\Gamma \vdash \phi_1 * \phi_2}$$

The input–output model (Miller)

- All ‘resources’ are sent up the first branch.
- Those required to close the branch (if possible) are retained on the branch, with what remains being sent to the next branch.
- Consider how this gives rise to an interesting notion of a **J** (don’t worry for now about bases for this).
- Generalize this strategy: more **J**s (see Harland and Pym, *Resource-distribution via Boolean Constraints*, ACM ToCL, 2003).

Back to coalgebra: the input-output model

Coalgebraically, we can see this as a further structuring of the search space TX by updating to $Bool \times In \times TX$.

Then the coalgebra $\alpha : X \to Bool \times In \times TX$ works as follows:

- At a reduction with a multiplicative conjunction leaf $\Gamma \vdash \phi_1 * \phi_2$, α is designed to choose to reduce the left-hand premiss.
- *In* outputs a list of the formulae required for the current proof of ϕ_1, and
- *Bool* is a test for termination of that branch.
- If a proof is found, the next step of computation defined by α is to begin reducing the right-hand premiss with respect to the context given by Γ minus the current value of *In*.
- *In* is then reset to the empty list and *Bool* to false.
- Could, at least in principle, capture proof-search by taking coalgebras over reduction models.
- But, in the light of recent work, we'd like to simplify things somewhat.

Back to coalgebra: tactical proof

How can we see tactical proof coalgebraically?

- A *tactic* is a partial function that takes a goal and returns a list of goals and a procedure:

$$\text{tactic} : \text{goal} \rightharpoonup \text{goal list} \times \text{procedure}$$

- How to see this statefully, and so coalgebraically?

$$\text{tactic} : \text{goal} \rightharpoonup \text{goal list} \times \textbf{next goal} \times \text{procedure}$$

- That is,

$$\frac{\mathrm{Subgoal}_1 \quad \ldots \quad \textbf{Next Subgoal} \quad \ldots \quad \mathrm{Subgoal}_k}{\mathrm{Goal}} \Uparrow$$

- Cf. the input-output model or hereditary Harrop resolution

Hereditary Harrop resolution tactic

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\mathcal{P} \vdash \boldsymbol{G'}} \quad \dfrac{}{A' \vdash A'}\,Ax}{\mathcal{P} \vdash A'}\; G' \supset A' \in \mathcal{P}}{\vdots\ \text{right rules}}}{\mathcal{P} \vdash \boldsymbol{G}} \qquad \dfrac{\dfrac{}{A \vdash A}\,Ax}{\mathcal{P} \vdash A}\; G \supset A \in \mathcal{P}$$
$$\text{reduce } G$$

Here the stateful, coalgebraic specification of the '**Next Subgoal**' selects right-hand premisses until atomic and selects the left-hand premiss and specifies, in the case of Prolog, the leftmost matching clause.

Why coalgebra?

- The motivation for adopting a coalgebraic approach is strong; it handles both
  - Kripke semantics, as a framework for defining logics, and
  - Proof-search and model-checking procedures.
- The latter point perhaps deserves some expansion.
- Search procedures are *not* naturally functional, but are naturally stateful. ML, the programming language initially developed as language for specifying tactics in the 'Logic for Computable Functions' (LCF) , is not a purely functional language. Rather, it makes explicit use of imperative *exceptions*.
- Exceptions are used to handle failure and continuation/ resumption — essential features of search procedures.
- Thus while deduction naturally has functional accounts, reduction does not.

Directions?

- Can we reconstruct P-tS as a reductive theory? In terms of base-extension semantics (B-eS),
  - bases can be directly read as reduction operators
  - but application combinators, which apply base rules, must be reinterpreted — choice of operator for reduction?
  - then, how to incorporate stateful proof-search procedures?
- But this B-eS view doesn't address the failure of a search to find a proof.
- This takes us back to the picture of Dummett / Prawitz / Schroeder-Heister / Milner, which handles proof-like structures that are not necessarily proofs
- So, perhaps a reductive P-tS requires a unification of these two views?

Additional References

- D. Miller. A Logical Analysis of Modeules in Logic Programming. *The Journal of Logic Programming* 6(1–2), 1989, 79–108.
- R. Milner. The Use of Machines to Assist in Rigorous Proof. *Phil. Proc. Royal Soc. A*, 1984. `https://doi.org/10.1098/rsta.1984.0067`.
- D. Pym and E. Ritter. *Reductive Logic and Proof-search: Proof Theory, Semantics, and Control*. Oxford Logic Guides 45. Oxford University Press, 2004.
- D. Pym. Reductive logic & proof-theoretic semantics: a coalgebraic perspective. In: *Proc. Proof-theoretic Semantics: Assessment and Future Perspectives*, P. Schroeder-Heister and T. Piecha (editors), Third Tübingen Conference on Proof-theoretic Semantics, 27–30 March 2019. `https://publikationen.uni-tuebingen.de/xmlui/handle/10900/93935`.

- Alexander Gheorghiu, Simon Docherty, and David Pym. Reductive Logic, Proof-search, and Coalgebra: A Perspective from Resource Semantics. Revised version to appear, Samson Abramsky on Logic and Structure in Computer Science and Beyond, Alessandra Palmigiano and Mehrnoosh Sadrzadeh (editors), Outstanding Contributions to Logic, Springer, 2023. `http://www.cs.ucl.ac.uk/staff/D.Pym/rlp-sc.pdf`.
- A. Gheorghiu and D. Pym. Proof-theoretic Semantics and Tactical Proof. Submitted, 2023. `https://arxiv.org/pdf/2301.02302`
- A. Gheorghiu and D. Pym. Semantic Foundations of Reductive Reasoning. To appear, *Topoi*, 2025. `http://www.cs.ucl.ac.uk/staff/D.Pym/Semantic_Foundations_of_Reduction_Reasoning.pdf`.
- J. Harland and D. Pym. Resource-distribution via Boolean constraints. *ACM Trans. on Comp. Logic* 4(1), 56–90.
- Michael J. Gordon, Arthur J. Milner, Christopher P. Wadsworth. Edinburgh LCF: A Mechanized Logic of Computation. LNCS 78. Springer, 1979.

Schedule

1. What is Proof-theoretic Semantics (P-tS)?
   - Inferentialism.
   - Consequence.
   - Proof-theoretic Validity (P-tV).
2. Base-extension Semantics (B-eS):
   - B-eS for Intuitionistic Propositional Logic.
   - Naturality, categorically speaking.
   - B-eS and P-tV.

Schedule

3. Reductive Logic, Tactical Proof, and Logic Programming:
   - Reductive Logic and P-tV.
   - Tactical Proof.
   - Remarks on Logic Programming, B-eS, and Coalgebra.
4. Modal and Substructural Logics, Resource Semantics, and Modelling:
   - B-eS for Modal Logics.
   - B-eS for Substructural Logics.
   - Resource Semantics and Modelling with B-eS.

Schedule

- Most of what we will introduce will be quite new to most people, with a fairly significant philosophical basis, and with quite a lot of ground to be covered.
- Our approach will mainly be conceptual, with little detailed, formal proof.
- Nevertheless, the formal details of everything we cover are available in books and papers that will be referenced.

Additional Material

Generalizing Input-Output

At the level of generality of the coalgebraic picture — remaining agnostic about the exact nature of the termination test — it is easy to see how this coalgebraic description could incorporate even more general examples like the resource-distribution model of Harland & Pym (Resource-distribution via Boolean constraints, ACM ToCL, 2000), where the test is solutions to Boolean constraints.

More generally still, this can be seen as the use of the classical (sequent) calculus, as a meta-calculus for the reductive (proof-search) view of non-classical logics, L.

$$\text{L-search} = \text{LK-search} + \text{Conditions}$$

- Dummett's restriction of multiple-conclusion sequent calculus for IL
- Essentially modal conditions
- Resource-distribution in substructural logics ...

Actually, it's the and–or combinatorics that matter, with negation a sometimes-convenient tool.

The calculus with constraints

The basic idea is handle multiplicative rules as follows:

- Use additive versions instead.
- But, label each formula with a Boolean variable to capture whether or not its occurrence is 'real'.
- Set up a system of Boolean equations that characterize the existence of multiplicatively correct proofs.
- Solve the equations.

Now we can sketch the set-up of formal calculus that captures this perspective. We will need a some notation and definitions.

- An *annotated formula* is a formula ϕ together with a Boolean expression e, written $\phi[e]$.
- The grammar of Boolean expressions is $e ::= x \mid \overline{x} \mid x.e \mid \overline{x}.e$.
- $exp(\phi)$ denotes the Boolean expression associated with ϕ.
- A sequent consisting wholly of annotated formulae is known as a *resource sequent*.

- Given a multiset of annotated formulae, $\Delta = \{ \phi_1[e_1], \ldots, \phi_n[e_n] \}$ and a total assignment I of Boolean variables in Δ, define $\Delta[I] = \{ \phi_1[v_1], \ldots, \phi_n[v_n] \}$, where each e_i has value v_i under I.
- Let $\Delta[I]^1$ denote the multiset of annotated formulae in $\Delta[I]$ s.t. e evaluates to 1 under I.
- If $V = \{x_1, \ldots, x_n\}$ is a set of Boolean vars, then let $\overline{V}$ denote $\{\overline{x_1}, \ldots, \overline{x_n}\}$. Let $\{e\}^n$ denote the multiset which contains n copies of the Boolean expression e.
- Let $\Gamma = \{ \phi_1[e_1], \ldots, \phi_n[e_n] \}$ be a multiset of annotated formulae and let $\{x_1, \ldots, x_n\}$ be a set of Boolean vars not occurring in Γ. Then define $\Gamma \,.\, \{x_1, \ldots, x_n\} = \{ \phi_1[e_1.x_1], \ldots, \phi_n[e_n.x_n] \}$.

The calculus with constraints

Now we can give the calculus of constraints for Linear Logic (the calculus for BI is similar; See Harland and Pym, Resource-distribution via Boolean constraints, ACM ToCL, 2000.

Rather than give the whole calculus, we give examples of a few key types of rules.

- Multiplicatives
- Additives
- Structural rules (exponentials).
- Axiom

The calculus with constraints

$$\frac{\Gamma, \phi_1[e], \phi_2[e] \vdash \Delta}{\Gamma, (\phi_1 \otimes \phi_2)[e] \vdash \Delta} \otimes\mathsf{L} \qquad \frac{\Gamma.V \vdash \phi_1[e], \Delta.W \quad \Gamma.\overline{V} \vdash \phi_2[e], \Delta.\overline{W}}{\Gamma \vdash (\phi_1 \otimes \phi_2)[e], \Delta} \otimes\mathsf{R}$$

$$\frac{\Gamma, \phi_1[x.e], \phi_2[\overline{x}.e] \vdash \Delta}{\Gamma, (\phi_1 \& \phi_2)[e] \vdash \Delta} \&\mathsf{L} \qquad \frac{\Gamma \vdash \phi_1[e], \Delta \quad \Gamma \vdash \phi_2[e], \Delta}{\Gamma \vdash (\phi_1 \& \phi_2)[e], \Delta} \&\mathsf{R}$$

$$\frac{\Gamma, \phi[e] \vdash \Delta}{\Gamma, (!\phi)[e] \vdash \Delta} !\mathsf{L} \qquad \frac{!\Gamma \vdash \phi[e], ?\Delta}{!\Gamma \vdash !(\phi)[e], ?\Delta} !\mathsf{R}$$

$$\frac{e_1 = e_2 = 1 \quad \forall e_3 \in exp(\Gamma \cup \Delta)(e_3 = 0)}{\Gamma, \mathsf{p}[e_1] \vdash \mathsf{p}[e_2], \Delta} \text{Axiom}$$

The calculus with constraints

We define *resource derivations*:

- a resource derivation is a tree regulated by the rules of the resource calculus
- each formula in the endsequent is assigned a distinct Boolean variable, together with a partial assignment of the Boolean variables appearing in the derivation
- a resource deriviation is *total* if its assignment of the Boolean variables is total, other wise it is partial
- it is *closed* if all of its leaves are axioms.

A *resource proof* is a total, closed resource derivation in which all the Boolean vars in the endsequent and all principal formulae are assigned the value 1.

Logical theory

- Soundness: Let $\Gamma \vdash \Delta$ be a resource proof. If $\Gamma \vdash \Delta$ has a resource proof R with Boolean assignment I, then the linear proof corresponding to R is a linear proof of $\Gamma[I]^1 \vdash \Delta[I]^1$.

  The proof proceeds by induction on the structure of resource proofs. As an example, consider the case for $\otimes$R.

  If the last rule of the resource proof is $\otimes$R, then $\Delta = \phi_1[1], \Delta'$, and $\Gamma.V \vdash \phi_1[1], \Delta'.W$ and $\Gamma.\overline{V} \vdash \phi_2[1], \Delta'.\overline{W}$ have resource proofs, where V and W are disjoint sets of Boolean vars. By IH, there are linear proofs of $(\Gamma.V)[I]^1 \vdash (\phi_1)[I]^1, (\Delta'.W)[I]^1$ and other one, so there is a linear proof of $(\Gamma.V)[I]^1, (\Gamma.\overline{V})[I]^1 \vdash (\phi_1 \otimes \phi_2)[I]^1, \Delta'[I]^1$, as required.

Logical theory

- Completeness: If $\Gamma \vdash \Delta$ has a proof Φ in the linear sequent calculus, then there are disjoint sets of Boolean vars V and W such that $\Gamma.V \vdash \Delta.W$ has a resource proof R and the linear proof tree corresponding to R is Φ.

  The proof proceeds by induction on the structure of proofs in the linear sequent calculus. It is mainly bureaucratic. We sketch the case for $\otimes$R.

  The following (immediately provable) lemma is useful: If $\Gamma \vdash \Delta$ has a closed resource derivation, then $\Gamma\,, \phi[0] \vdash \Delta$ and $\Gamma \vdash \phi[0]\,, \Delta$ also have closed resource derivations.

Logical theory

If the last rule in the linear proof Φ is $\otimes$R, then we have that $\Gamma = \Gamma_1, \Gamma_2$ and $\Delta = \phi_1 \otimes \phi_2$ s.t. $\Gamma_1 \vdash \phi_1, \Delta_1$ $\Gamma_2 \vdash \phi_2, \Delta_2$ are provable in linear sequent calculus.

By IH, there are disjoint V_1, V_2, W_1, W_2 s.t. $\Gamma_1.V_1 \vdash \phi_1, \Delta_1.W_1$ $\Gamma_2.V_2 \vdash \phi_2, \Delta_2.W_2$ have resource proofs.

By the lemma above, there are closed resource derivations of $\Gamma_1.V_1, \Gamma_2.\{0\}^n \vdash \phi_1, \Delta_1.W_1, \Delta_2.\{0\}^n$ and the other one. It follows that there are V and W and a total assignment I of $V \cup W$ s.t. $(\Gamma_1.V_1, \Gamma_2.V_2).V \vdash \phi_1, (\Delta_1.W_1, \Delta_2.W_2).W$ and $(\Gamma_1.V_1, \Gamma_2.V_2).\overline{V} \vdash \phi_1, (\Delta_1.W_1, \Delta_2.W_2).\overline{W}$ have resource proofs, and the result follows by application of the $\otimes$R rule for resource derivations.

Strategies revisited

- The formal set-up we've establish allows us to give a precise account of the computational strategies — lazy, eager, and intermediate — sketched above.
- The construction of a resource proof — which, combinatorially, is additive (cf. classical sequent calculus) — generates sets of Boolean constraints.
  - *Lazy distribrution* solves one multiplicative branch's worth of Boolean constraints at a time.
  - *Eager distribution* waits until *all* leaves are closed before attempt to solve the set of constraints.
  - *Intermediate distribution* a specified finite number of multiplicative branches is solved, before proceeding to the next batch.

Example

- Recall $p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)$.
- A resource proof looks like this (abusing notation here and there):

$$\dfrac{\dfrac{P_1 \qquad P_2}{p[x_1], p[x_2], q[x_3], q[x_4] \vdash p \otimes q} \quad \dfrac{P_3 \qquad P_4}{p[\overline{x_1}], p[\overline{x_2}], q[\overline{x_3}], q[\overline{x_4}] \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)}$$

where the leaves are as follows:

$$P_1 : p[x_1.y_1], p[x_2.y_2], q[x_3.y_3], q[x_4.y_4] \vdash p$$

$$P_2 : p[x_1.\overline{y_1}], p[x_2.\overline{y_2}], q[x_3.\overline{y_3}], q[x_4.\overline{y_4}] \vdash q$$

$$P_3 : p[\overline{x_1}.z_1], p[\overline{x_2}.z_2], q[\overline{x_3}.z_3], q[\overline{x_4}.z_4] \vdash p$$

$$P_4 : p[\overline{x_1}.\overline{z_1}], p[\overline{x_2}.\overline{z_2}], q[\overline{x_3}.\overline{z_3}], q[\overline{x_4}.\overline{z_4}] \vdash q$$

The lazy strategy yields the following sequence of constraints and solutions:

Leaf	**Constraints added**	**Solutions**
P_1	$x_1.y_1 = 1\,,\ x_2.y_2 = 0$ $x_3.y_3 = 0\,,\ x_4.y_4 = 0$	$x_1 = 1\,,\ y_1 = 1$
P_2	$x_2.\overline{y_2} = 0\,,\ x_3.\overline{y_3} = 1$ $x_4.\overline{y_4} = 0$	$x_3 = 1\,,\ y_3 = 0$ $x_3.y_3 = 0$
P_3	$z_2 = 1\,,\ z_4 = 0$	$z_2 = 1\,,\ z_4 = 0$
P_4	$\overline{x_4}.\overline{z_4} = 1$	

The eager strategy would collect the entire set of equations below and solve them simultaneously:

$$x_1.y_1 = 1\,.\,x_2.y_2 = 0\,,\; x_3.y_3 = 0\,,\; x_4.y_4 = 0$$

$$x_1.\overline{y_1} = 0\,.\,x_2.\overline{y_2} = 0\,,\; x_3.\overline{y_3} = 0\,,\; x_4.\overline{y_4} = 0$$

$$\overline{x_1}.z_1 = 0\,.\,\overline{x_2}.z_2 = 0\,,\; \overline{x_3}.z_3 = 0\,,\; \overline{x_4}.z_4 = 0$$

$$\overline{x_1}.\overline{z_1} = 0\,.\,\overline{x_2}.\overline{z_2} = 0\,,\; \overline{x_3}.\overline{z_3} = 0\,,\; \overline{x_4}.\overline{z_4} = 1$$

Intermediate strategies introduce more structure solutions between these two extremes.

Other bits and pieces

- Stephen Read, in his book *Relevant Logic*, gives a systematic account of a (large) family of relevant systems through a few simple rules.
- The Boolean constaints approach can be applied to this family.
- BI is not in that family, but, as remarked, the approach applies. (BI is closely related to DW.)
- As usual, see Harland and Pym, ACM ToCL 2000, for details.
- A key, closely related, idea is that of *abduction*, usually attributed, in modern logic at least, to Charles Sanders Pearce (1839–1914).
- Abduction is the basis of the INFER static analyser based on Separation Logic.
- Conjecture that these methods can be of help in that work.