Reductive logic & proof-theoretic semantics: a coalgebraic perspective

David Pym

Thanks to Simon Docherty and Alexandra Silva

UCL & The Alan Turing Institute

Proof-theoretic Semantics
Assessment and Future Perspectives
Third Tübingen Conference on Proof-theoretic Semantics,
27–30 March 2019

I want to say some quite simple things, trying to make some connections between

- reductive logic & tactical proof construction,
- proof-theoretic semantics, and
- coalgebraic semantics, as a candidate unifying approach.

Reductive logic: proof theory and proof-search

Here, we have the most basic idea: we 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 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

Towards a semantic perspective.

- Reductive logic:
  - Proof-search
    - Syntactic Reductions (e.g., for logic programming, theorem proving)
    - Bigger space than proofs
  - Truth-functional semantics
    - Semantic Reductions (e.g., for model-checking)
    - Bigger space than realizers
  - Distinction at axioms and atomics, respectively.
- But these larger spaces are not sufficient alone to characterize reductive logics: typically, reductions are one-to-many, whereas, typically, deductions are many-to-one.
- Let's unpack this a bit, following Pym & Ritter, *Reductive Logic and Proof-search: Proof Theory, Semantics, and Control*, Oxford Logic Guides, 2004, has a detailed set-up for classical and intuitionistic logic.

Theoretical backstory

- A *reduction model* is a fibred structure $\mathcal{R}$ — in the sense of the use of fibred and indexed categories, and doctrines, in categorical logic — interpreting propositions and proofs — relative to indeterminates, interpreted using polynomial constructions, which stand for terms and propositions that remain to be calculated.
- Along with this, we need a semantic judgement, defined relative to the model,

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

  between worlds W, indeterminates in Θ, sequents $\Gamma \vdash \phi$, and reductions, Φ.
- At world W, relative to Θ, Φ is a reduction of $\Gamma \vdash \phi$:

$$\Gamma \vdash \Phi : \phi$$

Theoretical backstory

- In this truth-functional sense, *soundness* means that all $\Gamma \mathrel{?\text{-}} \phi$ 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 $\Gamma \mathrel{?\text{-}} \phi$ have reductions.
- Again, Pym & Ritter, Oxford Logic Guides, 2004, has the details.

Theoretical backstory

- A reduction Φ is interpreted as a map

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

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

Theoretical backstory

- 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:

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

- 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 Guides, 2004.

Milner and LCF: a concrete theory of proof-search

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

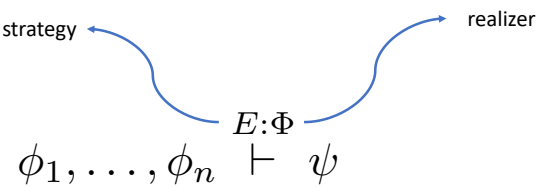
- Complex goals require, in practice, complex strategies.
- Need combinators, called *tacticals*, for composing tactics.
- Tactical combinations of tactics are themselves tactics.
- Examples would include:
  - Basic sequencing
  - The definition of *uniform proof* in the sense of Miller
  - In some sense, 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 $\simeq$.

Proof-theoretic semantics

Proof-theoretic used to be known as the *theory of meaning* — that is, of logical of constructs — in the sense of Prawitz, Martin-Löf, Sundholm, and others, mainly in the Scandinavian logical school.

- Basic idea: provide theory of logical validity that is based on proof-theoretic structures instead model-theoretic structures.
- We can think of this as working with mathematical structures that are built out of proof systems (inference rules, meaningfully organized) instead of satisfaction relations (truth in models).

Proof-theoretic semantics

Some questions that proof-theoretic semantics asks.

- In order to understand how it is that proof characterizes meaning, of structures are proofs delineating examples?
- One answer, inspired by truth semantics!, is that inferences rules are special cases of relations on sequents (or other basic units of a proof system).
- Then, within this bigger space of constructions, how can proofs — or, more generally, things equivalent to proofs — be identified? There's a kind of subtext of constructivism here.

I'll try to look at all this in terms of Milner's theory of tactical proof.

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 Prawitz, Schröder-Heister, ... approach (e.g., Proof-theoretic vs model-theoretic consequence, 2008).
- *Proof structures* $\mathcal{D}$ that are tree-like arrangements of *sequents*.
- A *justification system* $\mathcal{J}$ that maps structures to structures.
- Idea is that justifications pick out those structures that correspond to proofs in a 'ground' system of proof-theoretic rules.

Relate to proof-theoretic semantics

- Validity with respect to $\mathcal{J}$ and atomic system S (inference restricted to propositional atoms):

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

- $\mathcal{J}(\mathcal{J}_1, \ldots, \mathcal{J}_n)$ amounts to the procedure, relative to possibly generalized S ('ground' inferences).
- Strategy not included in this picture (as I understand it).

Towards a coalgebraic approach

- A *coalgebra* for an endofunctor $F : \mathcal{C} \to \mathcal{C}$ 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 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.

This approach provides an algebraic framework within which to generalize the theoretical backstory.

Towards a coalgebraic approach — first, a logic

- Kripke semantics can be seen coalgebraically.
- Example: BI, the logic of bunched implications (O'Hearn & Pym, BSL 1999) the basis of Separation Logic.
  - 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.

Towards a coalgebraic approach — first, a logic

- Ordered partial monoid $(R, \sqsubseteq, \circ, e)$ of worlds, r, s, t, ...

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

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 $r = e$ and 0 if $r \neq e$ — for I
  - $\pi_1(\alpha(r)) = \{(s, t) \mid s \circ t \leq r\}$ — for $*$
  - $\pi_2(\alpha(r)) = \{(s, t) \mid r \circ s = t\}$ — for $-\!\!*$

- 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

$$\delta_X(\phi * \psi) = \{t \in TX \mid \exists (x, y) \in \pi_2(t), x \in \delta_X(\phi), y \in \delta_X(\psi)\}$$

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

What's the use of all this?

Proof-search with substructural connectives

- From the computational perspective, the reduction operators that correspond to the inference rules for multiplicative connectives, such as $\otimes$ and $\multimap$, and $*$ and $\mathrel{-\!\!*}$, are problematic.
- For example,

$$\frac{\Gamma_1 \vdash \phi_1 \quad \Gamma_2 \vdash \phi_2}{\Gamma \vdash \phi_1 * \phi_2} \quad \Gamma = \Gamma_1, \Gamma_2$$

- How to calculate the Γs?
- Iterates through the search: suppose $\phi_1 = \psi_1 * \psi_2$, then need $\Gamma_1 = \Delta_1, \Delta_2 \ \ldots$.
- 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

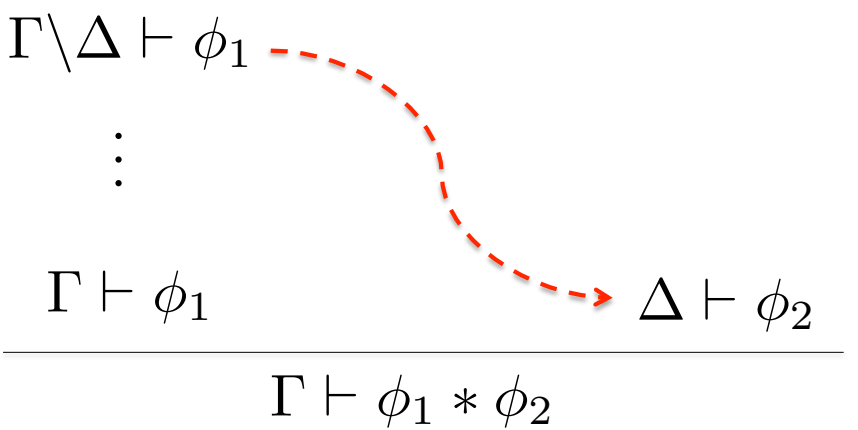


Figure 1: The input–output model (Miller)

- 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.

Back to coalgebra, for 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;
- *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.

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 and tacticals in 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.

Generalizing

At this level of generality — remaining agnostic about the exact nature of the termination test — it is easy to see how this coalgebraic description could incorporate 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.

A general approach to resource distribution

- We consider a sequent calculus for, for example, Linear Logic in which the non-deterministic splitting of contexts at multiplicative reductions is explicit.
- This allows us to set up a calculus which is independent of the choice of strategy that is used to distribute formulae, but which makes the necessary constraints explicit.
- To motivate/frame the approach, let's start with examples of key strategies for calculating multiplicative splitting.
- Method is very general, actually ...
- Let's begin with a simple example: the provable Linear Logic sequent

$$p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)$$

Lazy distribution

First pass all of context to a chosen (lefmost, say) branch, calculate which formulae are required to close the branch, and pass the remaining formulae to the next branch.

- So, first

$$\dfrac{\dfrac{\dfrac{}{\not{p}, p, \not{q}, \not{q} \vdash p} \quad \dfrac{}{X_2 \vdash q}}{p, p, q, q \vdash p \otimes q} \quad \dfrac{}{X_1 \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)}$$

- Then X_2 gets p, q, q and uses a q and so X_1 gets p and q.
- Repeat for the leftmost remaining branch, and the proof

$$\dfrac{\dfrac{\dfrac{}{\not{p}, p, \not{q}, \not{q} \vdash p} \quad \dfrac{}{\not{p}, q, \not{q} \vdash q}}{p, p, q, q \vdash p \otimes q} \quad \dfrac{\dfrac{}{p, \not{q} \vdash p} \quad \dfrac{}{\not{p}, q \vdash q}}{p, q \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)}$$

Lazy distribution

- Let's see how a lazy search can fail.
- So, first

$$\dfrac{\dfrac{\dfrac{}{\not{p}, p, \not{q}, \not{q} \vdash p} \quad \dfrac{}{X_2 \vdash q}}{p, p, q, q \vdash p \otimes q} \qquad \dfrac{}{p, p, q, q \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \,\&\, (p \otimes q)}$$

- Again, X_2 gets p, q, q and uses a q.
- Now we fail: all of the leaves on the left-hand branch have been closed. unused formulae remain, and there is nowhere to send them (the branching point below is additive).

Eager distribution

- Pass all formulae to all possible branches

$$\dfrac{\dfrac{\dfrac{}{p, \cancel{p}, \cancel{q}, \cancel{q} \vdash p} \quad \dfrac{}{\cancel{p}, \cancel{p}, q, \cancel{q} \vdash q}}{X_1 \vdash p \otimes q} \quad \dfrac{\dfrac{}{\cancel{p}, p, \cancel{q}, \cancel{q} \vdash p} \quad \dfrac{}{\cancel{p}, \cancel{p}, \cancel{q}, q \vdash q}}{X_2 \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)}$$

- p , p , q , q has to all leaves.
- Requirements to close each leaf can be solved simultaneously.
- So a proof is determined.

Intermediate distribution

- Lazy and eager are extreme points in the space of possible strategies here.
- In between, we have ‘intermediate’ strategies.
- That is, where many multiplicative branch is explored simultaneously *up to some specified maximum number*.
- Lazy is depth-first, eager is breadth-first, intermediate is bounded depth-first (cf. ‘iterative deepening’).

The generality of the approach

See Harland and Pym, Resource-distribution via Boolean constraints, ACM ToCL 2000 for the following:

- Formulation of general sequent calculus with Boolean constraints;
- Soundness and completeness results;
- The method appllies to full Linear Logic;
- The method applies to BI's sequent calculus;
- The method applies to the full family of relevant logics as described in Read's *Relevant Logic*, Blackwell, 1988.

Additionally, we conjecture that the method applies to the families of layered graph logics and the families of bunched modal and epistemic logics sketched in these lectures.

We also conjecture fits into the coalgebraic framework.

Summary to take away

- General, theoretically supported view of reductive logic — proof-theoretically and model-theoretically.
- Uniform coalgebraic treatment of structure and control.
- Sits in the framework of proof-theoretic semantics.
- Provides a setting for systematic understanding of

$$\text{Reductive Logic} = \text{Structure} + \text{Control}$$