1 Introduction

In this blog post, we provide an overview of the *Dialectica categories* and explore their connection to **Bel**, the category of *backward error lenses.*

- Developed by De Paiva 1991, the Dialectica categories are categorical models of intuitionistic linear logic. Fixing a base category **C**, we can construct a Dialectica category, abbreviated **DC**, such that logical formulas can be interpreted as objects in **DC** and deductions can be interpreted as morphisms.
- The objects of **DC** are (set-theoretic) *relations* on objects from **C**.
- The morphisms in **DC** are pairs of morphisms in **C** satisfying a certain condition. This category is one of the earliest examples of *lenses*, a data accessor structure we see in functional programming.

Our goal is to use the Dialectica categories to inform our construction of a similar category, **Bel**, from Kellison et al. 2025.

- **Bel** is the category used to interpret a linear type system, BEAN, which synthesizes bounds on roundoff error for numerical programs. We show soundness of these error bounds by interpreting BEAN typing judgments as morphisms in **Bel**.
- The objects of **Bel** are metric spaces, which may be rewritten as families of relations on a set.
- The morphisms in **Bel** are triples of functions reminiscent of lenses.

Sound familiar? We thought so. Now, we're going to introduce linear logic and the categories **DC**, and compare them to BEAN and the category **Bel**.

2 Linear logic

2.1 Deductive logic

A deductive logic is just a set of *axioms* and *inference rules* which tell you how to write down *deductions.* An axiom looks like this:

$$\frac{}{\Gamma, A \vdash A}$$

To the left of $\vdash$ is the *context*, the set of logical formulas that we are given. Here, Γ is a set of formulas, and A is a single formula. To the right of $\vdash$ is the conclusion, A. The $\vdash$ means "proves." So, this axiom says: "Given Γ and A, we can prove A." This seems pretty self-evident, which is why it's an axiom.

An inference rule looks like this:

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \text{ and } B}$$

Here, Γ and Δ are two contexts, and Γ, Δ is a context combining them. The deductions $\Gamma \vdash A$ and $\Delta \vdash B$ above the bar are requirements for deducing $\Gamma, \Delta \vdash A$ and B. So, this rule says, "If given Γ we can prove A, and given Δ we can prove B, then given both Γ and Δ, we can prove A and B. The idea is that any deduction we're able to prove from these rules should be "valid" (consistency), and any valid deduction should be provable from these rules (completeness).

2.2 Intuitionistic linear logic

Linear logic is a *substructural* logic, which means it's missing at least one of the common *structural* rules found in other logics. In this case, linear logic is missing *weakening* and *contraction.*

$$\frac{\Gamma \vdash B}{\Gamma, \Delta \vdash B}\ (\textsc{Weakening}) \qquad\qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ (\textsc{Contraction})$$

Weakening says that if we can prove B from a set of assumptions Γ, then if we are given even more assumptions, we can still prove B. Contraction says that we don't need to use assumptions twice. These rules make sense if we interpret formulas as true statements about the world, e.g. A is "it is raining" and B is "the ground is wet." However, in linear logic, we interpret formulas as *finite resources that must be consumed.* So $\Gamma \vdash A$ means "given exactly all the resources in Γ, we can produce an A." Now, weakening and contraction don't make sense. If $\Gamma \vdash B$, then we cannot conclude $\Gamma, \Delta \vdash B$, because B does not need a Δ to be produced. If $\Gamma, A, A \vdash B$, then B needs two copies of A, and we cannot produce B from just one A. Linear logic is nice because it can model real-world situations, like chemical reactions and concurrent processes.

Intuitionistic logic means that we can't assume the law of the excluded middle (for all A, either A or not A) nor double negation (if not (not A), then A). In a nutshell, this means all our proofs must be constructive. A deeper overview of the history of linear logic and its broader impact on computer science can be seen in Di Cosmo and Miller 2023. For now, consider these four inference rules from intuitionistic linear logic:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \otimes B}\ (\otimes\ \textsc{Intro}) \qquad\qquad \frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \& B}\ (\&\ \textsc{Intro})$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ (\oplus\ \textsc{Intro-R}) \qquad\qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ (\multimap\ \textsc{Intro})$$

The first rule, ($\otimes$ Intro), says that if $\Gamma \vdash A$, and $\Gamma \vdash B$, then $\Gamma \vdash A$ "and" B in the sense that we can produce either A or B, but not at the same time. This is because we are in linear logic, and A and B each use up all the resources in Γ, but we only have one Γ. Alternatively, (& Intro) says that if we have $\Gamma \vdash A$, and $\Delta \vdash B$, then $\Gamma, \Delta \vdash A$ "and" B, where we do have both A and B at the same time. This is because we have all the resources required to produce both. Next, ($\oplus$ Intro-R) says that if $\Gamma \vdash A$, then $\Gamma \vdash A$ "or" B, but we're not necessarily able to produce B. Finally, $\multimap$ is the linear logic version of implication. So ($\multimap$ Intro) says that if we have $\Gamma, A \vdash B$, then if we only have Γ, we just need another A to produce a B. Later, we're going to show how to interpret these rules in our categorical model, **DC**.

3 The Dialectica categories

Now that we understand the logic that we're trying to model, we will give a construction for the category **DC** from a category **C**. This construction is given in Chapter 1 of De Paiva 1991. Given some category **C** with finite limits (importantly, finite products and pullbacks), we can construct a category **DC** as follows.

- The objects in **DC** are triples (U, X, α) where U and X are objects in **C**, and α is a relation on $U \times X$, where U and X are interpreted as sets. What this really means is that α is a *subobject* of the product $U \times X$, but we will stick to the relation terminology.
- The morphisms of **DC** are pairs of morphisms (f, F), where $f : U \to V$ is a "forward" map and $F : U \times Y \to X$ is a "backward" map. A morphism from (U, X, α) to (V, Y, β) must satisfy the following condition:
$$\alpha(u, F(u, y)) \Rightarrow \beta(f(u), y).$$
- The identity morphism on (U, X, α) is (id_U, π_2). (Check that it satisfies the condition!)
- The composition of morphisms $(f, F) : (U, X, \alpha) \to (V, Y, \beta)$ and $(g, G) : (V, Y, \beta) \to (W, Z, \gamma)$ is as follows:
  - The forward map $U \to W$ is the composition $g \circ f$.
  - The backward map $U \times Z \to X$ is composed as follows:
  $$U \times Z \xrightarrow{\Delta \times \mathrm{id}_Z} U \times U \times Z \xrightarrow{\mathrm{id}_U \times f \times \mathrm{id}_Z} U \times V \times Z \xrightarrow{\mathrm{id}_U \times G} U \times Y \xrightarrow{F} X.$$
  Here, $\Delta : U \to U \times U$ is the diagonal morphism. So, the backward map is not as nice but it still works.

Given some extra conditions on **C**, the logical connectives we showed earlier from linear logic have their corresponding operations in **DC**:

- tensor product $\otimes$,
- Cartesian product &,
- internal Hom objects $\beta \multimap \gamma$ satisfying the following natural isomorphism:

$$\mathrm{Hom}(\alpha \otimes \beta, \gamma) \cong Hom(\alpha, \beta \multimap \gamma),$$

- and weak coproducts $\oplus$.

TODO: go through these constructions in more detail. Note that $\otimes$ and & are bifunctors, which define symmetric monoidal structures on **DC**.

4 Interpretation of linear logic into DC

First, fix an interpretation function $[\![\cdot]\!]$, which takes logical formulas to objects in **DC**. Looking back at our logical rules, Γ is a finite collection of logical formulas $A_1, \dots, A_n$. We interpret Γ as:

$$[\![\Gamma]\!] = [\![A_1]\!] \otimes \cdots \otimes [\![A_n]\!].$$

Furthermore, this interpretation function maps the logical connectives to their corresponding operations in **DC**. Now, we can prove the following theorem:

Theorem 1. *If $\Gamma \vdash A$ in intuitionistic linear logic, then there exists a morphism in **DC** $(f, F) : [\![\Gamma]\!] \to [\![A]\!]$.*

Proof. We perform induction on the derivation of $\Gamma \vdash A$ using the inference rules of linear logic. We show the cases for the four inference rules given above. TODO: fill this in □

5 The Bean type system

A type system is just like a logic, except it tells you how to construct a *program* and give it a *type*. TODO

6 The Bel category

Now, we're going to switch gears and talk about the category, **Bel**, giving the semantics of the Bean type system.

- The objects of **Bel** are triples (X, d_X, r) where X is a set, $d_X : X \times X \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ is a function giving a notion of distance, and $r \in \mathbb{R}_{\geq 0} \cup \{\infty\}$ is a constant called the *slack*. This doesn't look too much like a relation right now, but we could alternatively write (X, d_X, r) as $\{R_\varepsilon : \varepsilon \in \mathbb{R}_{\geq 0}\}$ where $R_\varepsilon(x, y) \Leftrightarrow d_X(x, y) \leq \varepsilon$.
- Morphisms between (X, d_X, r_X) and (Y, d_Y, r_Y) are triples of functions $(f, \tilde{f}, b)$ where $f, \tilde{f} : X \to Y$ and $b : X \times Y \rightharpoonup X$. For $x \in X$ and $y \in Y$ such that $d_Y(\tilde{f}(x), y) < \infty$, these functions must satisfy the following conditions:
  1. $d_X(x, b(x, y)) - r_X \leq d_Y(\tilde{f}(x), y) - d_Y$, and
  2. $f(b(x, y)) = y$.

  The idea is that f represents an ideal function, $\tilde{f}$ represents our approximating program, and b maps inputs and approximate outputs to a backward error witnesses.

A well-typed program in Bean corresponds to a morphism in **Bel**. TODO

7 Goals

7.1 Function spaces

In a functional programming language, we ideally want to be able to write, well, functions. TODO

7.2 Probabilistic programs

Can we think of idealized and approximate functions as probabilistic programs in our language? This would entail adding some distribution monad as part of our composition, for which we can use the coend calculus Loregian 2021. The goal is to maintain backwards error bounds even under randomized computation. To do this, we can treat these programs as effectful programs by attaching a monad to the lens-maps.

8 Conclusion

References

De Paiva, Valeria Correa Vaz (1991). *The dialectica categories.* Tech. rep. University of Cambridge, Computer Laboratory.

Di Cosmo, Roberto and Dale Miller (2023). "Linear Logic". In: *The Stanford Encyclopedia of Philosophy.* Ed. by Edward N. Zalta and Uri Nodelman. Fall 2023. Metaphysics Research Lab, Stanford University.

Kellison, Ariel E et al. (2025). "Bean: A Language for Backward Error Analysis". In: *arXiv preprint arXiv:2501.14550.*

Loregian, Fosco (2021). *(Co) end calculus.* Vol. 468. Cambridge University Press.