UNIVERSITY OF
CAMBRIDGE
Department of Computer
Science and Technology

Univalent Semantics for Varying Evaluation Strategies

Beatrice Szilvasy

St Catharine's College

May 5, 2026

Submitted in partial fulfillment of the requirements for the
Computer Science Tripos, Part III

Total page count: 49

Main chapters (excluding front-matter, references and appendix): 39 pages (pp 1–39)

Main chapters word count: 7674

Methodology used to generate that word count: `texcount`

Declaration

I, Beatrice Szilvasy of St Catharine's College, being a candidate for Part III of the Computer Science Tripos, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. In preparation of this report, I adhered to the Department of Computer Science and Technology AI Policy. [The project required the approved use of {insert name of technology} and such use is acknowledged in the text at the relevant sections.] [I am content for my report to be made available to the students and staff of the University.]

Signed:

Date:

Abstract

TODO: this is the original as in the proposal. I will revise this as the work gets wrapped up.

Call-by-value and call-by-name are dual evaluation strategies in programming languages; when these are mixed in an effectful language, function composition ceases to be associative, rendering ordinary categories unsuitable for modelling their denotational semantics. Suitable constructs such as duploids have been studied in previous work, but in a univalent setting it is desirable to work with "univalent" versions, where equivalences correspond to identity types. In this work we will study and formalise the suitable notions of univalence for these constructs using the Rocq UniMath library.

Acknowledgements

This project would not have been possible without the wonderful support of ….

Contents

1 HALF-POLISHED Introduction

This dissertation is concerned with the semantics of programming languages in which composition is non-associative. That is, languages in which valid programs

$$\texttt{let}\ y := (\texttt{let}\ x := e_1\ \texttt{in}\ e_2)\ \texttt{in}\ e_3 \tag{1.1}$$

and

$$\texttt{let}\ x := e_1\ \texttt{in}\ (\texttt{let}\ y := e_2\ \texttt{in}\ e_3) \tag{1.2}$$

(e_3 does not reference x) may compute differently.[1] One way in which such programs may arise is in the presence of side effects with mixed evaluation order.

To illustrate the effect of evaluation order, consider the program:

$$\texttt{let}\ x := (\texttt{print}(\texttt{"hello"});\ 1)\ \texttt{in}\ x + x \tag{1.3}$$

where evaluating $\texttt{print}(v)$ has the side-effect of outputting v to a console. There are two canonical evaluations of program 1.3 depending on the choice of evaluation order for the "`let`" construct.

- Using *call-by-name* (CBN), evaluate the expression $\texttt{let}\ x := e_1\ \texttt{in}\ e_2$ by first substituting e_1 for x in e_2. Then evaluate the resulting expression $e_1[x := e_2]$. Using call-by-name, program 1.3 outputs `hellohello` and returns 2.
- Using *call-by-value* (CBV), evaluate $\texttt{let}\ x := e_1\ \texttt{in}\ e_2$ by first evaluating e_1 to its value v. Then substitute v for x in e_2. Finally, evaluate the resulting expression $e_2[x := v]$. Using call-by-value, program 1.3 outputs just `hello` but still returns 2.

So long as all `let` expressions in a language use the same choice of evaluation order out of call-by-value or call-by-name, composition will remain associative. Combining

[1] TODO: Perhaps Levy's "let e_1 be x in e_2" syntax would be better?

call-by-value and call-by-name in a single language exposes non-associative composition. Let us write

$$\texttt{let}\ x \overset{\text{V}}{:=} e_1\ \texttt{in}\ e_2 \qquad \text{for call-by-value, and}$$
$$\texttt{let}\ x \overset{\text{N}}{:=} e_1\ \texttt{in}\ e_2 \qquad \text{for call-by-name.}$$

Then consider, for an example of non-associativity, the program

$$\begin{aligned}\texttt{let}\ y \overset{\text{N}}{:=} (\texttt{let}\ x \overset{\text{V}}{:=} (\texttt{print}(\texttt{"hello"});\ 1)\\ \texttt{in}\ (\texttt{print}(\texttt{"world"});\ x))\ \texttt{in}\ y + y\end{aligned} \tag{1.4}$$

of the left-associated form (program 1.1), and

$$\begin{aligned}&\texttt{let}\ x \overset{\text{V}}{:=} (\texttt{print}(\texttt{"hello"});\ 1)\ \texttt{in}\\ &(\texttt{let}\ y \overset{\text{N}}{:=} (\texttt{print}(\texttt{"world"});\ x)\ \texttt{in}\ y + y)\end{aligned} \tag{1.5}$$

its reassociation to the right (program 1.2). Recall that call-by-name `let` substitutes the entire assigned expression into the body, while call-by-value `let` evaluates it first and then substitutes. Then we can see that evaluating program 1.4 will output `helloworldhelloworld`, but program 1.5 will output `helloworldworld`, and both will return 2. Thus we have two programs that differ only up to associativity, and yet compute differently.

It is typical to interpret the semantics of programming languages in *categories* (definition 2.12), but this breaks down for modelling non-associative computation. Let us consider $\texttt{let}\ y := (\texttt{let}\ x := e_1\ \texttt{in}\ e_2)\ \texttt{in}\ e_3$ (program 1.1) and $\texttt{let}\ x := e_1\ \texttt{in}\ (\texttt{let}\ y := e_2\ \texttt{in}\ e_3)$ (program 1.2) to be interpreted in some category $\mathcal{C}$, and assign to each expression $x : A \vdash e : B$ some morphism $[\![e]\!] : [\![A]\!] \to [\![B]\!]$ in $\mathcal{C}$. The question of whether program 1.1 and program 1.2 compute differently then amounts, informally,[2] to the following equation:

$$\Big([\![e_1]\!]\,;[\![e_2]\!]\Big);[\![e_3]\!] \overset{?}{=} [\![e_1]\!]\,;\Big([\![e_2]\!]\,;[\![e_3]\!]\Big). \tag{1.6}$$

However, this is trivialized by the associativity of composition (definition 2.12.4.b) in any category $\mathcal{C}$. If we wish to take seriously the idea that equations of the form in equation 1.6 do not necessarily hold, then we must generalize the notion of a category.

[2] This would normally require extra bookkeeping to keep multiple variables in scope.

In this dissertation, it will be Munch-Maccagnoni's *duploids* [21] that generalize and take the role of categories for modelling non-associative computation.

For what purpose should we mix evaluation orders? One motivation is that it allows subsuming the theories of both call-by-name and call-by-value into a single paradigm such as *call-by-push-value* [15] (CBPV). The denotational semantics of call-by-push-value can be modelled "indirectly" by adjunctions [14], generalizing the widespread use of monads for the semantics of call-by-value [19]. [TODO: sing more praises for Levy and CBPV here because everything good about CBPV also applies for the related semantics in duploids.] Duploids fit into the picture by providing "direct" denotational semantics for mixed evaluation order, and have a correspondence with the aforementioned adjunction models [21]. [TODO: more motivation can't hurt?]

The purpose of this dissertation is to develop the theory of duploids formally in (a variant of) Martin Löf type theory and from a *univalent point of view* [24] (see section 2.2). [TODO: explain UF, briefly? I don't want to dwell on this too long in the introduction because it needs too much background, which is what the next chapter is for.] We[3] formalize some existing results about duploids within the Rocq library UniMath [9]. Some of this work involves transferring existing theorems over to a more recent definition of duploids (see remark 4.4). [NOTE: would be really cool to prove the full "structure theorem" for single-sorted duploids, since that does not yet exist in the literature.] We define what it means for a duploid $\mathcal{D}$ to be *univalent*: that isomorphisms $a \cong b$ in $\mathcal{D}$ (of a suitable kind) render the related objects $a, b : \mathcal{D}$ indistinguishable to the type theory. We show that the original construction of the duploid arising from an adjunction fails to be univalent, and then provide a novel construction of the same duploid which **is** univalent.

1.1 NOTES Outline

Rendered: May 5, 2026

These are my running notes about the structure of this text. Chapters are marked (in the TOC and their heading) with their completion status:

[3] Me, my notebooks, and my proverbial rubber duck.

- NOTES means (rendered) comments from me to the reader of the draft (also typically me). These are also scattered through the draft as TODO, FIXME and NOTE tags, so that the reader knows which parts are incomplete and how.
- TODO means that the section has not been written, or at least is not yet structured in a way that is actually coherent. Paragraphs may need to be rewritten or moved to different sections.
- VERY-ROUGH and ROUGH mean that some content exists and is intended to be legible, but content may be lacking and some things might not be coherent.
- HALF-POLISHED means the section is intended to be comprehensible as-is to the target audience, although is intended to undergo further refinement.
- "DONE" may happen in the future.

This is the running intended structure of the text:

- **Introduction** [3/5]
  - ☒ *Non-Associative Computation*
    - ☒ Explain how non-associative computation arises, especially by mixing evaluation order.
    - ☒ Point out how non-associativity makes categories unsuitable, and the existing solutions of adjunction models and duploids.
  - ☒ Tie back to the project.
  - ☐ *Univalence*
    - ☐ Explain the motivations for equivalence-invariance, especially for programming language semantics (we are masquerading as a CS paper, after all).
  - ☒ Summary of main results
  - ☐ *Overview* (you are here)
    - ∗ I will replace (and considerably shorten) this with prose nearer the end of writing.
- **Background** [2/3]

- ☐ *Mixed Evaluation Order*
- ☒ *Preliminary Type Theory*
  - ☒ Notations and conventions I will use for type-theoretic constructs.
- ☒ *Univalent Categories*
  - ☒ Definition of category in UF.
  - ☒ Definition of univalence.
    - Reflexive graphs?
    - Univalence of categories with extra structure?
  - ☒ Motivation for why assume hom-sets? (i.e. why **Cat** is not a category)

- **Univalent Duploids** (title TBD) [1/3]
  - ☒ *Unital Magmoids* [4/4]
    - ☒ Definition of unital magmoids
    - ☒ Definition of linearity/thunkability/(intermediateness)/polarities.
    - ☒ Polarized subcategories (`linear_category`).
      - Conditions for uniqueness of inverses `inverse_unique_linear`?
      - Isos of unital magmoids?
    - ☒ Linear-and-thunkable isomorphisms `lt_iso`, intermediate isomorphisms? (`intermediate_z_iso_interpose`); preserving polarities (`is_positive_of_lt_iso`).
      - Natural transformations? (If we get on to the 2-category of Duploids properly)
  - ⊟ *Duploids* [3/11]
    - ☒ Definition of preduploids
    - ☒ Polarization property `has_polarities` (makes all linear-and-thunkable isos intermediate `preduploid_lt_iso_is_intermediate`) – maybe defer to univalence section.
    - ☒ Definition of duploids

- ☐ Polarity shifts `has_polarity_shifts` and its universal property `negative_lift`.
- ☐ Unique up to linear-and-thunkable isomorphism.
- ∗ It would be nice to have a good Yoneda's lemma for unital magmoids, which would subsume this.
- ☐ Characterization of thunkability and linearity in terms of polarity shifts `is_linear_iff_force_unwrap`.
- ☐ Characterization of positive and negative objects `is_linear_wrap_of_positive`.
- ☐ The adjunctions within a duploid.
- ☐ Single vs. two-sorted duploids.
  - ☐ Syntactic subcategories `chosen_positive_category`
  - ☐ Equivalence of syntactic subcategories to the semantic ones `adj_equivalence_chosen_positive_to_positive`.
- ☐ The duploid arising from an adjunction (to which I gave the name "oblique duploid")
  - ☐ First part of the structure theorem (every two-sorted duploid is isomorphic to the oblique duploid on its shifts).
  - ☐ Relate to associative computation? E.g. (later) the oblique duploid subsumes kleisli categories and offer an explanation *and resolution* for their non-univalence.
- ☐ The equalizing requirement.

☐ *Univalence of Duploids* [0/5]

- ☐ Statement `is_duploid_univalent`.
  - ☐ Explanation and justification.
  - ☐ Split duploid univalence? [2]
  - ☐ Potentially relate to and mention points from Levy's contextual isomorphisms paper, or defer this to *related work*.

- ☐ Characterization `is_univalent_linear_and_thunkable_from_positive_thunkable_and_negative_linear_categories`.
- ☐ Consequence: shifts become property: `isaprop_has_polarity_shifts`.
- ☐ Consequence: equivalences of duploids are "isos" `is_duploid_equivalence`, `weq_duploid_equivalence_catiso`.
- ☐ Characterization of univalence `duploid_to_linear_and_thunkable_category_rxgraph_iso`.

- **The Univalent Duploid arising from an Adjunction** [0/2]
  - ☐ Failure of univalence for the oblique duploid.
    - e.g. the identity adjunction on a nonempty category.
  - ☐ Envelope duploid
    - ☐ Definition of the envelope duploid `envelope_duploid`.
    - ☐ Equalizing requirement and its characterization.
    - ☐ Univalence of the envelope duploid `is_univalent_envelope_duploid`.
    - ☐ All duploids arise from an adjunction up to equivalence of duploids.
- **Related Work** [0/1]
  - ☐ I made the grave mistake of not taking notes on all the papers as I read them. You might think I would have internalized this during my reading courses from Michaelmas, and you would be wrong. I will endeavour to fill this chapter with notes (probably not rendered) and then reformat to prose over time.
- **(Summary and) Conclusions**
  - Will write once done.
- Leave longer proofs to the appendix?

2 ROUGH Background

2.1 ROUGH Mixed Evaluation Order

It is a well-established tradition in theoretical computer science to consider programs as terms in variants of the lambda calculus. Although Church originally studied a *pure*, side-effect free, lambda calculus for the purposes of logic [3], many languages of practical use include *computational effects*, or simply *effects*, such as input/output (IO), mutable state, or non-termination.

A simply typed lambda calculus term $\Gamma \vdash e : A$ may be interpreted in any cartesian closed category as a morphism $[\![e]\!] : [\![\Gamma]\!] \to [\![A]\!]$ [13]. This interpretation is sound (and complete), in that if two terms $\Gamma \vdash e_1, e_2 : A$ are $\beta\eta$-equivalent, then their interpretations $[\![e_1]\!], [\![e_2]\!] : [\![\Gamma]\!] \to [\![A]\!]$ are equal in all cartesian closed categories (and vice-versa).

Pointing out that $\beta\eta$-equivalence is the wrong notion of program equivalence in the presence of computational effects, Moggi [19, 20] suggested the use of monads to model the semantics of effectful call-by-value lambda calculi instead. This way, given a suitable monad

$$(\mathcal{C} \xrightarrow{T} \mathcal{C}, \quad \mathsf{Id}_{\mathcal{C}} \overset{\eta}{\Rightarrow} T, \quad T^2 \overset{\mu}{\Rightarrow} T)$$

on a category $\mathcal{C}$, a compuational lambda term $\Gamma \vdash e : A$ can be interpreted as a morphism $[\![e]\!] : [\![\Gamma]\!] \to T\,[\![A]\!]$ in $\mathcal{C}$. The objects of $a : \mathcal{C}$ are the types of **values**, while objects $Ta : \mathcal{C}$ are the types of **computations** producing values of the type $a : \mathcal{C}$.

Moggi's interpretation of the computational lambda calculus involves a sort of "compilation" of lambda terms into a monadic style, inserting extra "bookkeeping code," such as units and multiplications, that were not present in the source. In the language of Führmann [7], monads constitute an *indirect semantics* for the call-by-value computational lambda calculus, in contrast to the *direct semantics* provided by, say,

cartesian closed categories for the pure lambda calculus. The difference being that the calculus's syntax is much more closely related to the constructs readily available in a direct semantics, whereas an indirect semantics requires an extra transformation step. Führmann [7] provides a direct semantics for the computational lambda calculus which he calls *abstract Kleisli categories* but which we shall call *thunk-force categories* [21].

Dually, it is possible to use comonads to model the semantics of call-by-name lambda calculi [14, 21]. This way, given a suitable comonad $U : \mathcal{D} \to \mathcal{D}$ on a category $\mathcal{D}$, we can interpret effectful call-by-name lambda terms $\Gamma \vdash e : A$ as morphisms $[\![e]\!] : U\,[\![\Gamma]\!] \to [\![A]\!]$. The dualized analogue of a thunk-force category is a *runnable monad* [21]. TODO: the types $a : \mathcal{D}$ are the stacks, but what are the objects $Ua : \mathcal{D}$?

The use of categories for the semantics of programming languages forces the choice of a fixed evaluation order. Otherwise, as seen in the introduction, composition ceases to be associative, and the use of categories becomes impossible.

This is undesirable for a number of reasons. Firstly, as argued by Levy [15], the combination of call-by-value and call-by-name into a single subsuming paradigm, call-by-push-value, obviates the need to reason about call-by-value and call-by-name separately. This reduces the burden of reasoning about models for both paradigms to reasoning about only one. Call-by-push-value has an indirect semantics in certain adjunctions [14]. Both call-by-value and call-by-name have reasonable translations into call-by-push-value in a way that preserves their equational theory and semantics.

Another motivation is advanced in Munch-Maccagnoni's thesis [22]. The relaxation of associativity captures meaningful properties about the types and programs, namely polarization. Guillaume proposes a direct semantics for mixed evaluation order called *duploids* [21]. Duploids subsume the direct semantics of thunk-force categories and runnable monads, and they also subsume the use of monads and comonads for call-by-value and call-by-name. Specifically, each adjunction $L \dashv R : \mathcal{P} \to \mathcal{N}$ gives rise to a duploid $\mathcal{E}[L \dashv R]$ which contains as subcategories the Kleisli category of $RL : \mathcal{P} \to \mathcal{P}$, and the co-Kleisli category of $LR : \mathcal{N} \to \mathcal{N}$.

The original study of duploids was done on what I call *two-sorted duploids*. These have two separate sorts of object: one sort of **positive** objects, or **values**; and one sort of **negative** objects, or **stacks**. In this work we shall study *single-sorted duploids* instead. In this case, positive and negative objects are distinguished not syntactically,

but *semantically* by an associativity condition. [TODO: I could do well being more specific about these things.]

Although the background and context of this work lies in the semantics of programming languages, programming languages themselves will play little role beyond this section. Instead, we focus our attention entirely on the mathematical structures themselves.

2.2 ROUGH Preliminary Type Theory

This dissertation consists of an English language discussion about a body of mathematics that takes place in the formal language of homotopy type theory. The theorems, definitions, and proofs are presented in ordinary mathematical English, but they are also assigned meaning as types and terms in the language of homotopy type theory. In this section I explain some notation and terminology I will use throughout.

Some knowledge of homotopy type theory and univalent foundations is recommended for readers, as it is necessary to understand the motivations and main results. A sufficient introduction to univalent mathematics as a whole has been written by Grayson [8]. For textbook accounts, I suggest Rijke's *Introduction to Homotopy Type Theory* [23], or the well-known *HoTT Book* [24]. Though insufficient for understanding this dissertation, Escardó has a concise self-contained introduction to the univalence axiom itself [6]. Knowledge of basic dependent type theory is assumed from here on out.

As documented in the UniMath [9] repository, we work in Martin-Löf type theory [18] with intensional identity types and no general W-types. Write $\prod_{x:B} E[x]$ for dependent products, and $\sum_{x:B} E[x]$ and for dependent sums; $A \times B$ for binary products, and $A \amalg B$ for binary sums; $A \to B$ for non-dependent function types.

Identification Types

Perhaps the most important aspect of univalent foundations is its treatment of equality between two two terms $a, b : A$ or types A, B type. There are two distinct notions of equality in Martin-Löf type theory.

The first notion is *judgemental equality*, written $A \equiv B$ type for types, $a \equiv b : A$ for terms, or simply $a \equiv b$ for either. Judgemental equality is symmetric, reflexive,

transitive. It is also respected by all judgements: assuming judgemental equality $a \equiv b$, the type or term a can be substituted throughout for b during type checking. Judgemental equality includes computation ("beta") rules: for example $(\lambda x : B, a)\ b \equiv a[x := b] : E[b]$ for $b : B$ and $x : B \vdash a : E[x]$; and uniqueness ("eta") rules: for example $f \equiv (\lambda x : B, f\ x) : \prod_{x:B} E[x]$.

Judgemental equality also includes definitions: $\mathsf{xyz} \stackrel{\text{def}}{\equiv} a : A$ implies $\mathsf{xyz} \equiv a : A$. Using the computation and uniqueness rules, we may write definitions in an implicit form, so that a need not be given explicitly. For example, the function

$$\mathsf{curry} : \prod_{f:\mathbb{N}\times\mathbb{N}\to\mathbb{N}} \mathbb{N} \to \mathbb{N} \to \mathbb{N}$$

may be defined equivalently either in "closed form" as

$$\mathsf{curry} \stackrel{\text{def}}{\equiv} \lambda f\ x\ y, f\ \langle x, y\rangle\,, \tag{2.1}$$

or "implicitly" as

$$\mathsf{curry}\ f\ x\ y \stackrel{\text{def}}{\equiv} f\ \langle x, y\rangle\,. \tag{2.2}$$

The judgemental equation 2.2 follows from equation 2.1 by computation rules, and the uniqueness of equation 2.1 as defining equation 2.2 follows by the uniqueness rules. It is left for the reader to infer the closed form of any implicit definition.

Judgemental equality is a **judgement**, not a type. Due to this, theorems in expressed in univalent foundations may not talk about judgemental equality. Instead, Martin-Löf type theory includes a second notion of equality: the *identification type*. For any type A and terms $a, b : A$, the identification type is written $a =_A b$ or simply $a = b$. A term $p : a =_A b$ of identification type is sufficient to render a and b indistinguishable within the type theory using *identification induction* as discussed below.

The identification type comes equipped with a constructor $\mathsf{refl}_a : a =_A a$. Judgemental equality implies identification, in that $a \equiv b : A$ gives $\mathsf{refl}_a : a =_A b$. The converse, $p : a =_A b$ implying $a \equiv b : A$, is called *equality reflection*, and we do not allow it. Its absence is the marking characteristic of "intensional" dependent type theory.

Instead of equality reflection, the identification type permits *identification induction*. To define a function $f : \prod_{x,y:A} \prod_{z:x=_A y} E[x, y, z]$, we need only define the case $f\ x\ x\ \mathsf{refl}_x$:

$E[x, x, \mathsf{refl}_x]$. This is justified by the eliminator $\mathsf{J}_{x,y,z.E[x,y,z]}$ with term formation rule

$$\frac{p : a =_A b \qquad r : \prod_{x:A} E[x, x, \mathsf{refl}_x]}{\mathsf{J}_{x,y,z.E[x,y,z]}(p, r) : E[a, b, p]} \tag{2.3}$$

and computation rule

$$\frac{r : \prod_{x:A} E[x, x, \mathsf{refl}_x]}{\mathsf{J}_{x,y,z.E[x,y,z]}(\mathsf{refl}_a, r) \equiv r\ a : E[a, a, \mathsf{refl}_a]}. \tag{2.4}$$

When identification induction is used, it is left for the reader to reconstruct the closed-form term using J. More typically, given $p : a =_A b$ we will informally use substitution to replace occurrences of a with b or vice-versa, particularly when the specific p is not relevant.

Truncation Levels

In the logic of some foundations such as ZFC, the concepts of "propositions" and "sets" are distinct. The latter constitutes "substance" whereas the former describes it. By contrast, in univalent foundations it is the type that takes the role of both substance and its description. TODO: finish this paragraph?

The fundamental behaviour of a proposition over "substance" is that it is computationally irrelevant. This is the concept known as *proof irrelevance*: though two proofs of a proposition may be textually different to the human reader, the logic itself is unable to distinguish one from the other. In univalent foundations, we **define** propositions to be those types which satisfy proof irrelevance.

Definition 2.1. A type A is a *proposition* (or *mere proposition* for emphasis) when for all elements $a, b : A$ there is an identification $a =_A b$.

Being a proposition is the second class in a hierarchical classification of types called *truncation levels*. Truncation levels describe how trivial the type of identifications for a given type is. The first truncation level is the *contractible types*, the second truncation level is propositions, the third is *sets*, and above that we have *n-groupoids* for $n \geq 1$.

Definition 2.2. A type A is *contractible* when there is a chosen element $\mathsf{pt} : A$ and for all elements $a : A$ there is an identification $\mathsf{pt} =_A a$.

Definition 2.3. A type A is a *set*[1] when $a =_A b$ is a proposition for all $a, b : A$; a *1-groupoid* when $a =_A b$ is a set, and generally a $(k+1)$-*groupoid* when $a =_A b$ is a k-groupoid.

Example 2.4. The empty type `empty` is a proposition, and the unit type `unit` is contractible. The booleans `bool` and naturals $\mathbb{N}$ are both sets but not propositions. All propositions are also sets, all sets are also 1-groupoids, and all k-groupoids are also $(k+1)$-groupoids.

Write $\mathtt{iscontr}(A)$, $\mathtt{isaprop}(A)$ and $\mathtt{isaset}(A)$ for the types of A respectively being contractible, a proposition, and a set. Assuming function extensionality, it can be proven that these are propositions.

As a special case, when A is a dependent sum $A \equiv \sum_{x:B} E[x]$ we shall write $\exists! x : B, E[x]$ for $\mathtt{iscontr}(\sum_{x:B} E[x])$. Using this notation, given a function $f : A \to B$, we write $\mathtt{isweq}(f) \stackrel{\text{def}}{\equiv} \prod_{y:B} \exists! x : A, f\ x =_B y$ for the type of f being an *equivalence*. Write $A \simeq B$ for the type of equivalences, that is $A \simeq B \stackrel{\text{def}}{\equiv} \sum_{f:A\to B} \mathtt{isweq}(f)$.

Sometimes the data of a type (which need not be a proposition) A is not so important, only whether it is inhabited. Or to use set-theoretic terminology, whether such an element *exists*. The *propositional truncation* of A, written $\|A\|$, is the proposition which captures this idea. It comes with a constructor

$$\frac{t : A}{|t| : \|A\|} \tag{2.5}$$

and for any proposition P an eliminator

$$\frac{h : \|A\| \qquad f : A \to P}{\mathtt{u}_P(h, f) : P} \quad \text{satisfying} \quad \frac{t : A \qquad f : A \to P}{\mathtt{u}_P(|t|, f) \equiv f\ t : P}. \tag{2.6}$$

When A is a dependent sum $A \equiv \sum_{x:B}, E[x]$, we shall write $\exists x : B, E[x]$ for $\left\|\sum_{x:B} E[x]\right\|$.

[1] This is orthogonal to concerns about size.

Universes

A *universe* $(\mathcal{U}, \mathsf{El})$ is a type $\mathcal{U}$ of *codes* along with, for each element $A : \mathcal{U}$, an interpretation $\mathsf{El}(A)$ type. We shall abuse notation and use simply $\mathcal{U}$ to refer to the universe. For a universe $(\mathcal{U}, \mathsf{El})$, we say that a type A is $\mathcal{U}$*-small* if there is an element $\mathsf{code}(A) : \mathcal{U}$ such that $\mathsf{El}(\mathsf{code}(A)) \equiv A$ type.[2] By contrast, a type is *large* with respect to some universes $\mathcal{U}, \mathcal{U}', \ldots$ when it is not small in any of them. We shall typically omit El and code, so that $A : \mathcal{U}$ implies A type, and vice-versa when A is $\mathcal{U}$-small.

Convention 2.5. We shall often simply say that a type is "small" or "large" when the universes it is small or large with respect to are obvious or have not given a name. This rule will also apply to other things that may be "small" or "large."

We will typically assume, unless otherwise mentioned, that a universe $\mathcal{U}$ is closed under all of the type constructors mentioned so far (products, sums, identifications, finite types, …). This holds of most of the universes we shall consider.

Example 2.6. Given a universe $(\mathcal{U}, \mathsf{El})$, there is a universe $(\mathtt{hSet}_{\mathcal{U}}, \mathsf{El}')$ of $\mathcal{U}$-small sets. The codes of $\mathtt{hSet}_{\mathcal{U}}$ are the pairs $\sum_{A:\mathcal{U}} \mathtt{isaset}(A)$, and the interpretation is given by $\mathsf{El}'\langle A, h_A\rangle \stackrel{\text{def}}{\equiv} \mathsf{El}(A)$. Likewise, $\mathtt{hProp}_{\mathcal{U}}$ is the universe consisting of $\mathcal{U}$-small propositions $\sum_{A:\mathcal{U}} \mathtt{isaprop}(A)$, defined in much the same way. The universe $\mathtt{hProp}_{\mathcal{U}}$ is closed under dependent products, binary products, and identifications, but **not** dependent sums or binary sums. It also does not include $\mathtt{bool}$ or $\mathbb{N}$.

A *univalent universe* is a universe $\mathcal{U}$ where the canonical map

$$\begin{aligned} &\mathtt{id\_to\_weq} : \prod_{A,B:\mathcal{U}} (A =_{\mathcal{U}} B) \to (A \simeq B) \\ &\mathtt{id\_to\_weq}\ A\ A\ \mathsf{refl}_A \stackrel{\text{def}}{\equiv} \mathsf{id}_A \end{aligned} \tag{2.7}$$

given by identification induction is an equivalence for all types $A, B : \mathcal{U}$. For a given universe $\mathcal{U}$, the *univalence axiom* states that $\mathcal{U}$ is univalent. In short, the univalence axiom states that the identifications $A =_{\mathcal{U}} B$ between two types are characterized by the equivalences $A \simeq B$ between those types. These, in turn, depend on the structure of the identifications within A and B themselves.

[2] TODO: should this be only an equivalence?

Example 2.7. If a universe $\mathcal{U}$ is univalent, then so too is $\mathtt{hSet}_{\mathcal{U}}$ and $\mathtt{hProp}_{\mathcal{U}}$. The converse need not be true in general.

The type of equivalences between types, say $\mathtt{bool} \simeq \mathtt{bool}$, is not in general a proposition. Thus, the univalence axiom notably provides a way of constructing identifications which are provably **not** identifiable with refl. This property is perfectly consistent with J [10]; indeed there are models including the univalence axiom that are as consistent as ZFC with some strongly inaccessible cardinals [11, 12].

We will at times the existence of a tower of universes $\mathcal{U}_0, \mathcal{U}_1, \cdots$. These are such that each $\mathcal{U}_k$ is $\mathcal{U}_{k+1}$-small. Moreover, we assume the universes are cumulative, if $A : \mathcal{U}_k$ then $A : \mathcal{U}_{k'}$ for any $k' \geq k$. We will need only the first three universes[3]. We shall explicitly mention the univalence axiom when assumed. UniMath further assumes a propositional resizing axiom,[4] but we will not be using it explicitly.

Classical Mathematics and Logic

By contrast to "univalent foundations," I use the term "set-theoretic foundations" to include both ZF(C) and type theories where all types are sets (in the sense above). Many theorems and definitions of set-theoretic foundations may be recovered in univalent foundations by assuming certain types to be sets, such as by reasoning about $\mathtt{hSet}_{\mathcal{U}}$ instead of $\mathcal{U}$ for a fixed universe $\mathcal{U}$. The main expressive power lost by moving from set-theoretic foundations to univalent foundations is the ability to express statements that are invariant under equivalence (of types, categories, or any other mathematical object).

It is an occasional misconception is that univalent foundations is incompatible with non-constructive reasoning such as the *law of excluded middle* (LEM) or the *axiom of choice* (AxiomOfChoice). However, these are perfectly admissible so long as they are stated correctly in terms of sets and propositions. They are also often rendered unnecessary by univalence. We shall use the law of excluded middle on one such occasion.

[3] TODO: I think? Assuming all our duploids are $\mathcal{U}_0$-small and therefore live in $\mathcal{U}_1$, the largest type we use is the ($\mathcal{U}_2$-small) reflexive graph of $\mathcal{U}_1$-small reflexive graphs.

[4] For technical and historical reasons, UniMath asks Rocq not to check universes, and so has $\mathcal{U} : \mathcal{U}$. As such, checking universe levels is the responsibility of us humans.

Further Conventions

Table 2.1 describes English language examples and their type-theoretic interpretation. In particular, I use a propositionally truncated interpretation of "there exists" and "or," for similarity with set-theoretic mathematics. In addition to the informal English language interpretations, the final column suggests topological interpretations. These may be realized more formally by models of univalent foundations such as Voevodsky's in simplicial sets [11, 12].

Convention 2.8. I use upper case for types (A, B, C, E, P), calligraphic font for categories and the like $(\mathcal{C}, \mathcal{D}, \mathcal{N}, \mathcal{P})$, lower case for objects and morphisms within categories $(a, b : \mathcal{C},\ f, g : a \to b)$, upper case for functors $(F, G, H : \mathcal{C} \to \mathcal{D})$, and Greek letters for natural transformations $(\alpha, \beta : F \Rightarrow G)$.

Notation 2.9 (infix). Where a type contains a binary operator such as $\to$, $=$, or $\simeq$, we may use the infix notation $a \xrightarrow{f} b$ to mean $f : a \to b$. Sharing expressions in such infix notation, as in $t_1 \overset{p}{=} t_2 \overset{q}{=} t_3$, denotes the independent judgements $p : t_1 = t_2$ and $q : t_2 = t_3$. It may additionally denote the appropriate composition of p and q of the type $t_1 = t_3$ too, depending on the context.

Convention 2.10. A word or phrase written in *italics* is a technical term introduced at that point, and may be found in the index. When a theorem or definition has been formalized in UniMath, we shall write the name of its definition in `monospace` font, hyperlinked to the file and line number in the UniMath repository. An example is "a *category* (`category`) is …" In anonymized versions, these links will necessarily be broken.

2.3 ROUGH Univalent Categories and Reflexive Graphs

Beyond the foundation of mathematics in which we work, the content of this dissertation is about non-associative generalizations of categories. Namely, Munch-Maccagnoni's *duploids* [21]. Before we can talk about univalent duploids, we ought understand categories in the context of univalent foundations.[5]

[5] TODO: This paragraph is slightly out of place now that I have moved things around.

Table 2.1: Interpretations of mathematical English. $E[x]$ denotes a type indexed by x, and P denotes a proposition.

Mathematical English	Homotopy Type Theory	Topology
Utterance	Judgement	Interpretation
description of a ***type*** *as below*	A type	A is a topological space
desc. of a ***type*** *mentioning* $x : B$	$x : B \vdash E[x]$ type	E is a fiber bundle over B
description of a ***term***	t or $t : A$	$t \in A$
$t_1 : A$ is $t_2 : A$	$t_1 \equiv t_2 : A$	$t_1 = t_2$
let x be an A	$\dfrac{x : A}{\ldots}$	let $x \in A$
assume P	$\dfrac{h : P}{\ldots}$	assume P
Definition *xyz* is $t : A$.	$\mathsf{xyz} \overset{\text{def}}{\equiv} t : A$	$\mathsf{xyz} \overset{\text{def}}{=} t$
Theorem P. *Proof.* t. $\square$	$t : P$	$t \in P$
Description	Type	Space
[pair/triple/... of] $x : B$, $E[x]$	$\sum_{x:B} E[x]$	the total space E
$x : B$ such that $P[x]$	$\sum_{x:B} P[x]$	subspace $\{x \in B \mid P[x] \text{ inhabited}\}$
function A to B	$A \to B$ or $\prod_{x:A} B$	function space $A \to B$
for all $x : B$, $E[x]$	$\prod_{x:B} E[x]$ or $\forall x : B, E[x]$	sections of $E \to B$
A is a proposition	$\mathtt{isaprop}(A)$	A is contractible-if-inhabited
there exists a unique A	$\mathtt{iscontr}(A)$	A is contractible
there exists a unique $x : B$ such that $P[x]$	$\mathtt{iscontr}(\sum_{x:B} P[x])$ or $\exists! x : B, P[x]$	
there exists A	$\lVert A \rVert$ (propositional truncation)	A is inhabited
there exists $x : B$ such that $P[x]$	$\left\lVert \sum_{x:B} P[x] \right\rVert$ or $\exists x : B, P[x]$	
A is a set	$\mathtt{isaset}(A)$	A is a homotopy 0-type
$t_1 : A$ and $t_2 : A$ are identified	$t_1 =_A t_2$	path space t_1 to t_2
(if A is a set) $t_1 : A$ and $t_2 : A$ are equal	$t_1 =_A t_2$	path space t_1 to t_2
either A or B	$A \coprod B$	disjoint union of A and B
A or B	$A \vee B$ or $\lVert A \coprod B \rVert$	
$f : A \to B$ is an equivalence	$\mathtt{isweq}(f)$ or $\prod_{y:B} \exists! x : A, f(x) =_B y$	f is a homotopy equivalence
A and B are equivalent	$A \simeq B$ or $\sum_{f:A\to B} \mathtt{isweq}(f)$	A and B are homotopy equivalent

The authoritative paper on categories in univalent foundations is by Ahrens et al. [1], whose formalisation in Rocq is today part of UniMath. An expanded version of that paper is included as Chapter 9 of the *HoTT Book* [24]. The development has since diverged from the paper: we use the names "`category`" and "`univalent_category`" for what used to be "precategory" and "category" respectively.

Intuition

It is well-known that in a category $\mathcal{C}$, the correct notion of "sameness" between two objects $a, b : \mathcal{C}$ is *isomorphism* $a \cong b$, and not (necessarily) the foundations' notion of "equality" $a = b$. In set-theoretic foundations, equality is overly restrictive: there may be objects $a \cong b$ that are not equal,[6] and there may be isomorphisms $a \cong a$ that are not the identity.[7] In univalent foundations, the type of identifications $a =_{\mathsf{ob}\ \mathcal{C}} b$ may very well correspond to isomorphisms $a \cong b$. A *univalent category* is one where they do.

An informative use-case of univalence is in relation to weak equivalences between categories. It is a theorem that for a univalent category $\mathcal{C}$ and a category $\mathcal{D}$, a fully-faithful and essentially surjective functor $F : \mathcal{C} \to \mathcal{D}$ is an adjoint equivalence. The analogous statement in set-theoretic foundations amounts to the axiom of choice. More generally, objects determined by a universal property in a univalent category $\mathcal{C}$ become literally unique, as do left- and right- adjoints of a fixed functor $F : \mathcal{C} \to \mathcal{D}$ to another category $\mathcal{D}$. In other words, universal properties become mere properties and, when they hold, contractible. Thus, univalence has the practical benefit of being able to talk about *the* object (functor, category, etc.) satisfying a universal property, with the representative being canonical and choice-free.[8]

When the univalence axiom holds, many naturally-arising categories are univalent. For example, for a univalent universe $\mathcal{U}$, the category $\mathbf{Set}_{\mathcal{U}}$ of $\mathcal{U}$-small sets $A, B : \mathtt{hSet}_{\mathcal{U}}$ and functions $A \to B$ is univalent (`HSET_univalent_category`). As an extension, it was shown by Coquand et al. [5] that the univalence axiom extends to a "large class of algebraic structures." This implies that categories of set-based algebraic structures such as monoids and rings are univalent too.

[6] Say, the sets $\{0, 1\}$ and $\{1, 2\}$.
[7] Say, the set $\mathbb{B}$ under boolean negation.
[8] TODO: How much does this even matter lol

Every category $\mathcal{C}$ is weakly equivalent to a univalent category $\mathfrak{R}\mathcal{C}$. This is in the sense that there is a fully-faithful and essentially surjective functor $\mathfrak{r}_{\mathcal{C}} : \mathcal{C} \to \mathfrak{R}\mathcal{C}$. We call the category $\mathfrak{R}\mathcal{C}$ the *Rezk completion* of $\mathcal{C}$. One way to construct the Rezk completion is as the image of the Yoneda embedding $\mathsf{y} : \mathcal{C} \to [\mathcal{C}^{\mathsf{op}}, \mathbf{Set}_{\mathcal{U}}]$ [1].

The univalence of a category is sensitive to how the category is constructed. A relevant example is the *Kleisli category* of a monad

$$(\mathcal{C} \xrightarrow{T} \mathcal{C}, \quad \mathsf{Id}_{\mathcal{C}} \overset{\eta}{\Rightarrow} T, \quad T^2 \overset{\mu}{\Rightarrow} T)$$

on a category $\mathcal{C}$. One way to construct the Kleisli category of T is as follows. The objects are those of $\mathcal{C}$, and, for objects $a, b : \mathcal{C}$, the morphisms are the Kleisli arrows $a \to Tb$ in $\mathcal{C}$. Identities and composition are through η and μ respectively. Even when $\mathcal{C}$ is univalent, this construction of the Kleisli category is **not** generally univalent (see example 2.11). Let us therefore call this non-univalent Kleisli category $\mathbf{Kleisli}_{\not{\mathsf{univ}}}(\mathcal{C}, T)$. There is a univalent construction of the Kleisli category, however, as the subcategory of free T-algebras in the Eilenberg-Moore category of T (`univalent_kleisli_cat`). We shall call this category $\mathbf{Kleisli}_{\mathsf{univ}}(\mathcal{C}, T)$, or simply $\mathbf{Kleisli}(\mathcal{C}, T)$. Explicitly, this amounts to considering a suitable subcategory of $\mathcal{C}$ whose objects are of the form Ta for some $a : \mathcal{C}$.

Example 2.11 (Levy [16])**.** Let $\mathcal{U}$ be a universe. Consider the state monad $Ta \overset{\text{def}}{\equiv} (\mathbb{N} \Rightarrow a \times \mathbb{N})$ on $\mathbf{Set}_{\mathcal{U}}$. Then the Kleisli arrows $a \to Tb$ of T correspond to set-functions $a \times \mathbb{N} \to b \times \mathbb{N}$. From this it is easy to see that there is an isomorphism $\mathtt{unit} \cong \mathtt{bool}$ in $\mathbf{Kleisli}_{\not{\mathsf{univ}}}(\mathbf{Set}_{\mathcal{U}}, T)$: this is simply an isomorphism $\mathtt{unit} \times \mathbb{N} \cong \mathtt{bool} \times \mathbb{N}$ in $\mathbf{Set}_{\mathcal{U}}$. However, the type `unit` is certainly not identified with `bool`!

Definitions

TODO: I might move these to the unital magmoids section and define categories/functors/natural transformations in parallel with the respective things for unital magmoids?

Definition 2.12 (`category` [1])**.** A *category* $\mathcal{C}$ consists of the following:

1. A type of *objects*, written $\mathsf{ob}\ \mathcal{C}$ or just $\mathcal{C}$.

2. For each pair of objects $a, b : \mathcal{C}$, a type of *morphisms* from a to b, written $\mathcal{C}[\![a, b]\!]$ or $a \to b$.

   a) $\mathcal{C}$ *has hom-sets*: each $\mathcal{C}[\![a, b]\!]$ is a set.

3. For each object $a : \mathcal{C}$, a distinguished *identity* morphism $\mathsf{id}_a : \mathcal{C}[\![a, a]\!]$.

4. A *composition* operator ; assigning to each pair of morphisms $a \xrightarrow{f} b \xrightarrow{g} c$ a composite $a \xrightarrow{f;g} c$ such that:

   a) Composition is *unital*: for $a \xrightarrow{f} b$ we have $\mathsf{id}_a ; f = f$ and $f ; \mathsf{id}_b = f$.

   b) Composition is *associative*: for $a \xrightarrow{f} b \xrightarrow{g} c \xrightarrow{h} d$ we have $f ; (g ; h) = (f ; g) ; h$.

Definition 2.13. Let $\mathcal{U}$ be a universe. We say that a category $\mathcal{C}$ is $\mathcal{U}$*-small* if the types $\mathsf{ob}\ \mathcal{C}$ and $\mathcal{C}[\![a, b]\!]$ (for all $a, b : \mathcal{C}$) are $\mathcal{U}$-small. A category is *large* with respect to some universes $\mathcal{U}, \mathcal{U}', \ldots$ when it is not small in any of them.

Most of definition 2.12 will be familiar to a mathematician who has come across categories. Indeed, putting aside the hom-set law for now (definition 2.12.2.a), one obtains the familiar set-theoretic definition by substituting "class" for "type" in the definition of objects and morphisms. The hom-set law simply makes identitification $f =_{a \to b} g$ of morphisms $f, g : a \to b$ be a proposition.

Definition 2.14 (`is_z_isomorphism`, `z_iso`)**.** Let $\mathcal{C}$ be a category. A morphism $p : a \to b$ between objects $a, b : \mathcal{C}$ *has an inverse* if there is a morphism $p^{-1} : b \to a$ such that

$$p ; p^{-1} = \mathsf{id}_a \quad \text{and} \quad p^{-1} ; p = \mathsf{id}_b \,. \tag{2.8}$$

In this case, we say that p is an *isomorphism* $p : a \cong b$.

Remark 2.15 (`isaprop_is_z_isomorphism`). It is well known that inverses of morphisms are unique in a category. That is, if we have a morphism $p : a \to b$ in a category $\mathcal{C}$ and any two morphisms $q, q' : b \to a$ satisfying equation 2.8, then we have $q = q'$. I spell out the proof, because it depends on associativity:

$$q = q ; (p ; q') = (q ; p) ; q' = q' \,.$$

In combination with the hom-set law, this makes the type of "having an inverse" be a proposition.

Definition 2.16 (`is_univalent` [1])**.** Let $\mathcal{C}$ be a category. Then there is a canonical map from identifications to isomorphisms

$$\begin{aligned} &\texttt{id\_to\_iso} : \prod_{a,b:\mathsf{ob}\ \mathcal{C}} (a =_{\mathsf{ob}\ \mathcal{C}} b) \to (a \cong b) \\ &\texttt{id\_to\_iso}\ a\ a\ \mathsf{refl}_a \overset{\text{def}}{\equiv} \mathsf{id}_a \end{aligned} \tag{2.9}$$

defined by identification induction (reminiscent of equation 2.7). The category $\mathcal{C}$ is *univalent* if `id_to_iso` $a\ b$ is an equivalence for all $a, b : \mathcal{C}$.

TODO: narrative to bring in functors?

Definition 2.17. A *functor* F between categories $\mathcal{C}$ and $\mathcal{D}$, written $\mathcal{C} \to \mathcal{D}$, consists of:

1. A mapping on objects $F_0 : \mathsf{ob}\ \mathcal{C} \to \mathsf{ob}\ \mathcal{D}$.
2. For all objects $a, b : \mathsf{ob}\ \mathcal{C}$, a mapping on morphisms $F_1 : \mathcal{C}[\![a, b]\!] \to \mathcal{C}[\![F_0\ a, F_0\ b]\!]$.
3. F preserves:
   a) Identities:
   $$F_1(a \xrightarrow{\mathsf{id}_a} a) = F_0\ a \xrightarrow{\mathsf{id}_{F_0 a}} F_0\ a$$
   for all $a : \mathcal{C}$.
   b) Composition:
   $$F_1\left(a \xrightarrow{f} b \xrightarrow{g} c\right) = F_0\ a \xrightarrow{F_1\ f} F_0\ b \xrightarrow{F_1\ g} F_0\ c$$
   for all $a \xrightarrow{f} b \xrightarrow{g} c$ in $\mathcal{C}$.

Both F_0 and F_1 may simply be written F.

Definition 2.18. A *natural transformation* α between functors $F, G : \mathcal{C} \to \mathcal{D}$, written $\alpha : F \Rightarrow G$, consists of:

1. For each object $a : \mathcal{C}$, a morphism $\alpha_a : Fa \to Ga$.

2. *Naturality*: for all objects $a, b : \mathcal{C}$ and morphisms $f : a \to b$, the equation $\alpha_a ; Gf = Ff ; \alpha_b$ holds. That is, the *naturality square*

$$\begin{array}{ccc} Fa & \xrightarrow{Ff} & Fb \\ {\scriptstyle \alpha_a}\downarrow & & \downarrow{\scriptstyle \alpha_b} \\ Ga & \xrightarrow{Gf} & Gb \end{array}$$

in $\mathcal{D}$ commutes.

Example 2.19. Let $\mathcal{C}$ and $\mathcal{D}$ be categories with functors $F, G : \mathcal{C} \to \mathcal{D}$ between them. Then the type of natural transformations $F \Rightarrow G$ is a set because $\mathcal{D}$ has hom-sets. Thus, there is a category $[\mathcal{C}, \mathcal{D}]$ (`functor_category`) whose objects are functors $F, G : \mathcal{C} \to \mathcal{D}$ and whose morphisms are natural transformations $\alpha : F \Rightarrow G$. If $\mathcal{D}$ is univalent, then the functor category $[\mathcal{C}, \mathcal{D}]$ is univalent. That is, the identifications $F = G$ between two functors $F, G : \mathcal{C} \to \mathcal{D}$ correspond to the natural isomorphisms $F \cong G$ between them (`is_univalent_functor_category`).

Non-example 2.20. Let $\mathcal{U}$ be a universe. There is in general no category $\mathbf{Cat}_\mathcal{U}$ of $\mathcal{U}$-small categories and functors between them. This is because, in general, the functors $\mathcal{C} \to \mathcal{D}$ between categories $\mathcal{C}$ and $\mathcal{D}$ do not form a set.[9] One must instead consider $\mathbf{Cat}_\mathcal{U}$ as a bicategory. Another approach, more reminiscent of set-theoretic foundations, is to restrict ourselves to *strict categories*, or $\mathtt{hSet}_\mathcal{U}$-small categories when $\mathcal{U}$-small. These are categories $\mathcal{C}$ whose objects $\mathtt{ob}\ \mathcal{C}$ form a set. Then we may perfectly well consider the category $\mathbf{Cat}_{\mathtt{hSet}_\mathcal{U}}$ of strict $\mathcal{U}$-small categories.

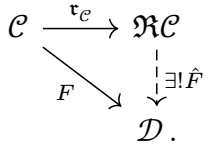
TODO: narrative to bring in Rezk completion?

Definition 2.21. The *Rezk completion* of a category $\mathcal{C}$ is a univalent category $\mathfrak{R}\mathcal{C}$ and a functor $\mathfrak{r}_\mathcal{C} : \mathcal{C} \to \mathfrak{R}\mathcal{C}$ which is fully-faithful and essentially surjective on objects.

Remark 2.22. The Rezk completion of categories $\mathfrak{R}$ enjoys the following universal property as a consequence of $\mathfrak{r}_\mathcal{C}$ being fully-faithful and essentially surjective. Let $\mathcal{C}$ be a category, $\mathcal{D}$ be a univalent category, and $F : \mathcal{C} \to \mathcal{D}$ be a functor. Then there exists

[9] Consider the two natural isomorphisms from $\mathbf{1} \xrightarrow{\mathtt{bool}} \mathbf{Set}$ to itself, when $\mathbf{Set}$ is univalent.

a unique functor $\hat{F} : \mathfrak{R}(\mathcal{C}) \to \mathcal{D}$ such that $\mathfrak{r}_{\mathcal{C}}; \hat{F} = F$. Graphically, this is the following commutative diagram in the bicategory **Cat**:[10]



Another way to say this is that $\mathfrak{R}$ extends to a pseudofunctor $\mathfrak{R} : \mathbf{Cat} \to \mathbf{Cat}_{\mathsf{univ}}$ which is the left-biadjoint of the forgetful pseudofunctor $U : \mathbf{Cat}_{\mathsf{univ}} \to \mathbf{Cat}$, witnessing $\mathbf{Cat}_{\mathsf{univ}}$ as a reflective sub-bicategory of **Cat**.

[10] This **Cat** should be $\mathbf{Cat}_{\mathcal{U}}$ for a universe $\mathcal{U}$ in which both $\mathcal{C}$ and $\mathfrak{R}\mathcal{C}$ are small. We make no assumption here about the size of $\mathfrak{R}\mathcal{C}$ in relation to $\mathcal{C}$. (NOTE: In the UniMath construction via the Yoneda embedding, $\mathcal{C}$ is large even when $\mathcal{C}$ is small. There is a HIT variant too which is the same size. I don't know how much I care to include these references.)

3 ROUGH Unital Magmoids

Definition 3.1. A *unital magmoid* or *non-associative category* [17] (`unital_magmoid`) is a generalization of a category that need not be associative (definition 2.12.4.b). More generally, a *reflexive graph*[1] (`rxgraph`) is a category that omits composition (definition 2.12.4) entirely and need not have hom-sets (definition 2.12.2.a). For exposition only, these can further be generalized to *magmoids* and *graphs*[1] respectively by omitting identities (definition 2.12.3, 2.12.4.a) entirely. These are summarized in table 3.1.

NOTE: I introduce reflexive graphs for the purpose of giving a name to funtors-that-preserve-only-identities, but I am not sure how to recast their purpose as a tool for looking at univalence of duploids and categories. This will likely materialize once I write the background chapter for univalent categories.

Table 3.1: Attributes of category-like structures.

Attribute	Structure				
	Category	Unital Magmoid	Magmoid	Reflexive Graph	Graph
Objects	✓	✓	✓	✓	✓
Morphisms	✓	✓	✓	✓	✓
Hom-sets	✓	✓	✓	✗	✗
Identities	✓	✓	✗	✓	✗
Composition	✓	✓	✓	✗	✗
Unital	✓	✓	N/A	N/A	N/A
Associative	✓	✗	✗	N/A	N/A

We import any definitions of structures in and on categories for unital magmoids and reflexive graphs directly, as much as they are well-defined. As such, a *unital*

[1] Our "graph" is similar to what others may call a "quiver" or "directed multigraph."

magmoid functor is a mapping on objects and morphisms that preserves identities and composition; a *reflexive graph functor* need only preserve identities, and a *graph functor* need preserve neither. A *natural transformation* $F \overset{\alpha}{\Rightarrow} G$ between (unital) magmoid functors $F, G : \mathcal{M} \to \mathcal{M}'$ is still a family of $\mathcal{M}'$-morphisms $\alpha_a : \mathcal{M}'[\![Fa, Ga]\!]$ such that the naturality square

$$\begin{array}{ccc} a & Fa & \xrightarrow{Ff} Fb \\ {\scriptstyle \forall f}\downarrow & {\scriptstyle \alpha_a}\downarrow & \downarrow{\scriptstyle \alpha_b} \\ b & Ga & \xrightarrow[Gf]{} Gb \end{array} \tag{3.1}$$

commutes.

The failure of associativity causes many imported notions to become weaker in unital magmoids, however. For instance, the imported notion of composition for natural transformations defined componentwise is ill-defined in general as it fails to preserve naturality (equation 3.1). The hom-set functor $\mathcal{M}[\![\cdot, \cdot]\!] : \mathcal{M}^{\mathsf{op}} \times \mathcal{M} \to \mathbf{Set}$ is only a reflexive graph functor, because it does not necessarily preserve composition of morphisms in either argument. Moreover, the imported notion of isomorphism $a \cong b$ becomes too weak to render the related objects indiscernible to $\mathcal{M}$ as they would in an associative category, which I will make precise in definition 5.1.

Many theorems about categories which fail to generalize to unital magmoids can be salvaged by hypothesizing associativity for certain special cases.

Definition 3.2 (`is_thunkable`, `is_negative`, … [21])**.** We say that a path of morphisms

$$a_0 \overset{f_1}{\longrightarrow} a_1 \overset{f_2}{\longrightarrow} a_2 \to \cdots \to a_{n-1} \overset{f_n}{\longrightarrow} a_n \tag{3.2}$$

in a unital magmoid *associates* when all ways of parenthesising their composite $f_1; f_2; \cdots; f_n$ are equal. In the simplest case, a triple of morphisms $a \overset{f}{\to} b \overset{g}{\to} c \overset{h}{\to} d$ associates if and only if we have $f; (g; h) = (f; g); h$.

We say that:

- f is *thunkable* when all triples (f, g, h) associate.
- h is *linear* when all triples (f, g, h) associate.
- g is *intermediate* when all triples (f, g, h) associate.
- b is (*semantically*) *negative* when all morphisms into b are thunkable.

- c is (*semantically*) *positive* when all morphisms out of c are linear.

This is depicted pictorially in figure 3.1.

Notation 3.3. When a path of morphisms as in equation 3.2 is known to associate, we will freely write its composite without parentheses as $f_1; f_2; \cdots; f_n$. Occasionally, we will use the diagram $a_0 \xrightarrow{f_1} a_1 \xrightarrow{f_2} a_2 \to \cdots \to a_{n-1} \xrightarrow{f_n} a_n$ itself to denote the composite.

Remark 3.4. It is easy to see that thunkability/linearity and negativity/positivity are dual, while intermediateness is self-dual. This is in the usual sense that being thunkable in some unital magmoid $\mathcal{C}$ is the same as being thunkable in $\mathcal{C}^{\mathsf{op}}$, and being negative in $\mathcal{C}$ is the same as being positive in $\mathcal{C}^{\mathsf{op}}$, up to inversion of the identification $f; (g; h) = (f; g); h$.

TODO: the names "linear" and "thunkable" are due to Munch-Maccagnoni's [21] observation that the associativity conditions coincide with the existing notions of purity by e.g. Führmann, and that the polarities correspond to polarization in logic.

Example 3.5. In any unital magmoid, the identity morphisms are thunkable, linear, and intermediate by unitality (definition 2.12.4.a). Composites of thunkable, linear, or intermediate morphisms are also thunkable, linear, or intermediate respectively.

Definition 3.6 (`linear_category`, `positive_category`, …)**.** Let $\mathcal{M}$ be a unital magmoid. $\mathcal{M}$ has the following subcategories.

- The *linear subcategory* $\mathcal{M}_l$ by restricting to linear morphisms.
- The *thunkable subcategory* $\mathcal{M}_t$ by restricting to thunkable morphisms.
- The *intermediate subcategory* $\mathcal{M}_i$ by restricting to intermediate morphisms.

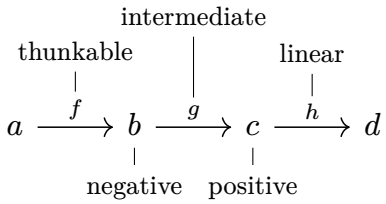


Figure 3.1: The triple (f, g, h) associates whenever any of the labelled properties holds.

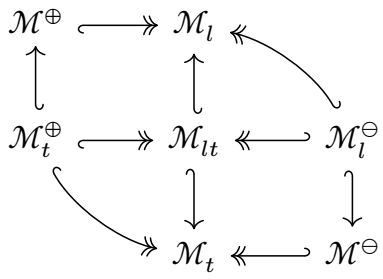


Figure 3.2: Inclusion functors between the subcategories of a unital magmoid $\mathcal{M}$. The double arrow head ($\twoheadrightarrow$) means full, and the hook ($\hookrightarrow$) means faithful.

- The *linear-and-thunkable subcategory* $\mathcal{M}_{lt}$ by restricting to linear-and-thunkable morphisms.
- The *negative subcategory* $\mathcal{M}^{\ominus}$: the full subcategory of semantically negative objects.
- The *positive subcategory* $\mathcal{M}^{\oplus}$: the full subcategory of semantically positive objects.
- The *negative-linear subcategory* $\mathcal{M}_l^{\ominus}$: the full subcategory of $\mathcal{M}_l$ of semantically negative objects.
- The *positive-thunkable subcategory* $\mathcal{M}_t^{\oplus}$: the full subcategory of $\mathcal{M}^{\oplus}$ of semantically negative objects.

By the property of polarization, all morphisms of $\mathcal{M}^{\ominus}$ and $\mathcal{M}_l^{\ominus}$ are automatically thunkable, and all morphisms of $\mathcal{M}^{\oplus}$ and $\mathcal{M}_t^{\oplus}$ are automatically linear. Figure 3.2 depicts the inclusion functors between the subcategories.

Notation 3.7. Let $a, b : \mathcal{M}$ be objects in a unital magmoid $\mathcal{M}$. We may write the subscripts from definition 3.6 onto the binary operators $a \to b$ (morphisms) or $a \cong b$ (isomorphisms) to denote that the morphism or isomorphism resides in the corresponding subcategory of the unital magmoid. For example, $a \to_l b$ denotes a linear morphism; $a \cong_{lt} b$ denotes a *linear-and-thunkable isomorphism*: an isomorphism which is linear-and-thunkable, and has a linear-and-thunkable inverse.

TODO: this is where I should talk about isomorphisms, or after preduploids maybe?

Lemma 3.8 (is_positive_of_lt_iso)**.** *In a unital magmoid $\mathcal{M}$, let $a, b : \mathcal{M}$ be objects and $p : a \cong_{lt} b$ be a linear-and-thunkable isomorphism. Then a is positive if and only if b is positive, and a is negative if and only if b is negative.*

Proof. Without loss of generality, we need only prove that a being positive implies that b is positive. By definition, this is that for any object c, any morphism $f : b \to c$ is linear. We have

$$\begin{aligned} b \overset{f}{\to} c &= \left(b \overset{p^{-1}}{\longrightarrow}_{lt} a \overset{p}{\to}_{lt} b \right) \overset{f}{\to} c \\ &= b \overset{p^{-1}}{\longrightarrow}_{lt} \left(a \overset{p}{\to}_{lt} b \overset{f}{\to} c \right) \end{aligned}$$

which is linear because p^{-1} is linear and a is positive. □

Lemma 3.9 (intermediate_z_iso_interpose)**.** *In a unital magmoid $\mathcal{M}$, let $a, b, b', c : \mathcal{M}$ be objects and $p : b \cong_i b'$ an intermediate isomorphism. Then the equation $(f; p); (p^{-1}; g) = f; g$ holds for any $a \overset{f}{\to} b \overset{g}{\to} c$.*

Proof. We have

$$\begin{aligned} &\left(a \overset{f}{\to} b \overset{p}{\to}_i \right) \left(b' \overset{p^{-1}}{\longrightarrow}_i b \overset{g}{\to} c \right) \\ &= a \overset{f}{\to} \left(b \overset{p}{\to}_i \left(b' \overset{p^{-1}}{\longrightarrow}_i b \overset{g}{\to} c \right) \right) \\ &= a \overset{f}{\to} \left(\left(b \overset{p}{\to}_i b' \overset{p^{-1}}{\longrightarrow}_i b \right) \overset{g}{\to} c \right) \\ &= a \overset{f}{\to} b \overset{g}{\to} c \,. \end{aligned}$$

□

4 ROUGH Duploids

Definition 4.1 (`preduploid` [4])**.** A *preduploid* (or *single-sorted preduploid* for emphasis) is a unital magmoid such that each object is semantically negative or semantically positive.

Lemma 4.2 (`preduploid_lt_iso_is_intermediate`)**.** *Every linear-and-thunkable isomorphism $p : a \cong_{lt} b$ in a preduploid $\mathcal{D}$ is intermediate.*

Proof. By the preduploid property (definition 4.1), we have that a is negative or positive. If it is negative, then p is automatically intermediate. If a is positive, then b must be positive by lemma 3.8, and so p is automatically intermediate. □

Definition 4.3 (`duploid`)**.** A *duploid* (or *single-sorted duploid*) is a preduploid $\mathcal{D}$ equipped with *polarity shifts*:

1. (`has_negative_shifts`) For every object $a : \mathcal{D}$,
   a) its *upshift*, a semantically negative object written $\Uparrow a : \mathcal{D}$,
   b) a linear morphism $\text{force}_a : \Uparrow a \to a$,
   c) a linear inverse for force_a called $\text{delay}_a : a \to \Uparrow a$.
2. (`has_positive_shifts`) Dually, for every object $a : \mathcal{D}$,
   a) its *downshift*, a semantically positive object written $\Downarrow a : \mathcal{D}$,
   b) a thunkable morphism $\text{wrap}_a : a \to \Downarrow a$,
   c) a thunkable inverse for wrap_a called $\text{unwrap}_a : \Downarrow a \to a$.

Remark 4.4. The above definitions make the polarities in a (pre)duploid be a *property* on unital magmoids, rather than *structure*. This differs from the original construction [21] which defined (pre)duploids in terms of separate negative and positive categories, with additional mixed composition operators, or as explicit structure. I therefore call the (pre)duploids from the original definition *two-sorted (pre)duploids* to distinguish them from the single-sorted (pre)duploids defined above. It is not always immediate how the theorems of single-sorted (pre)duploids transfer over to their single-sorted counterparts, and some of this dissertation consists of developing this theory.

We subsume two-sorted (pre)duploids into the theory of single-sorted (pre)duploids by considering them as extra structure on a single-sorted (pre)duploid. A *split preduploid* (`split_preduploid`) is a unital magmoid $\mathcal{D}$ equipped with a chosen *polarity mapping* $\omega : \mathsf{ob}\ \mathcal{D} \to \{\ominus, \oplus\}$. We say that an object a is *syntactically negative* if $\omega(a) = \ominus$ and *syntactically positive* if $\omega(a) = \oplus$, and require that every **syntactically** negative/positive object is also **semantically** negative/positive respectively. If $\mathcal{D}$ is a single-sorted duploid and a split preduploid, then $\mathcal{D}$ is a *split duploid* (`split_duploid`) when $\Uparrow a$ and $\Downarrow a$ are also syntactically negative and positive respectively. The original two-sorted duploids are precisely the split duploids where $\Uparrow$ and $\Downarrow$ are trivial (identity-on-objects and wrap/unwrap/force/delay identities) for syntactically negative and syntactically positive objects respectively.

TODO: Mention that this is a universal property? Probably derive how we can even get there from this definition. Should probably ask Éléonore or Guillaume as well.

An important result of duploids is their correspondence with adjunctions satisfying an equalization property [21] (NOTE: and adjunction models for CBPV in particular, which generalize adjunction/monadic models for CBV/CBN…). Every adjunction gives rise to a duploid, in the manner that we shall construct in chapter 6. Moreover, every duploid arises from an adjunction; up to isomorphism for two-sorted duploids, and up to (weak) equivalence for single-sorted duploids.

5 TODO Univalence of Duploids

TODO: I'm not really sure how to get into justifying the univalence condition for duploids nicely. I should probably revisit this once the univalence background chapter is written (and move some stuff from here to there).

Definition 5.1. Let $\mathcal{G}$ be a reflexive graph. We call $\mathcal{G}$ *univalent (as a reflexive graph)* (`is_rxgraph_univalent`) if for all morphisms $e : \mathcal{G}[\![a, b]\!]$, any property of e, or type indexed by e, can be derived "by induction on e" from the case where $a \equiv b$ and $e \equiv \mathsf{id}_a$. Type theoretically this suggests, given a type family $E[a, b, e]$ indexed by $a, b : \mathcal{G}, e : \mathcal{G}[\![a, b]\!]$, the inference rules:

$$\frac{t : E[a, a, \mathsf{id}_a] \qquad e : \mathcal{G}[\![a, b]\!]}{\mathsf{Ind}_{\mathcal{G}}(t, e) : E[a, b, e]}$$

$$\frac{t : E[a, a, \mathsf{id}_a]}{\mathsf{IndId}_{\mathcal{G}}(t) : \mathsf{Ind}_{\mathcal{G}}(t, \mathsf{id}_a) =_{E[a,a,\mathsf{id}_a]} t\,.}$$

Remark 5.2. A more convenient definition of the above is that a reflexive graph $\mathcal{G}$ is univalent if

$$\texttt{id\_to\_edge}\ a\ b : (a =_{\mathcal{G}} b) \to \mathcal{G}[\![a, b]\!]$$

defined by identification induction as

$$\texttt{id\_to\_edge}\ a\ a\ \mathsf{refl}_a \overset{\text{def}}{\equiv} \mathsf{id}_a\,.$$

is a weak equivalence for all $a, b : \mathcal{G}$.

NOTE: I believe there may be universe size differences between the two definitions, but I have been ignoring size issues and intend to keep doing so.

Example 5.3. Every univalent reflexive graph with hom-sets (definition 2.12.2.a) is a groupoid under pasting of identifications modulo `id_to_edge`.

Definition 5.4. When a reflexive graph has extra structure, univalence as a reflexive graph is typically too strong a condition. For example, a category which is univalent as a reflexive graph would have to be a groupoid by example 5.3. Instead, it makes sense to consider some subset/subtype of the morphisms, and ask for the sub-reflexive-graph containing those to be univalent. We call the morphisms in this sub-reflexive-graph the *indiscernibilities*.

Example 5.5. The indiscernibilities of a category $\mathcal{C}$ are its isomorphisms. Thus, $\mathcal{C}$ is *univalent (as a category)* [1] (`is_univalent`) when the subcategory of $\mathcal{C}$'s isomorphisms is univalent as a reflexive graph. Every category is weakly equivalent to a univalent category [1].

TODO: motivate the usefulness of indiscernibility induction?

The induction principle from definition 5.1 can also be used "backwards" to infer what properties the indiscernibilities in a *univalent* reflexive-graph-with-extra-structure must have. In a univalent category $\mathcal{C}$, indiscernibilities $p : \mathcal{C}[\![a, b]\!]$ *must* be isomorphisms, since we can find the inverse

$$p^{-1} : \mathcal{C}[\![b, a]\!]$$

by induction on p as

$$\mathsf{id}_a^{-1} \stackrel{\text{def}}{=} \mathsf{id}_a \,.$$

Any additional structure on top of the category must be respected too. An indiscernibility $p : \mathcal{C}[\![a, b]\!]$ in a univalent dagger category $\mathcal{C}$ (whatever that may be) must have that

$$p^\dagger =_{\mathcal{C}[\![b,a]\!]} p^{-1} \,,$$

since by induction on p and functoriality of the dagger we have

$$\mathsf{id}_a^\dagger = \mathsf{id}_a = \mathsf{id}_a^{-1} \,.$$

Remark 5.6. Using this reasoning, we can deduce from example 3.5 that the indiscernibilities of a unital magmoid $\mathcal{M}$ must be linear, thunkable, and intermediate, and have an inverse which is linear, thunkable, and intermediate.

NOTE: Ahrens et al. [2] provide a general procedure for fully determining what the indiscernibilities ought to be. Unfortunately I have yet managed to fully intuit the procedure itself.

Lemma 5.7. *Given a single-sorted preduploid $\mathcal{D}$, the following are equivalent:*

1. *$\mathcal{D}$ is univalent.*
2. *The linear-and-thunkable subcategory $\mathcal{D}_{lt}$ is univalent.*
3. *The subcategories $\mathcal{D}_t^{\oplus}$ and $\mathcal{D}_l^{\ominus}$ are univalent.*

Proof. $(1 \leftrightarrow 2)$: 2 is just a restatement of what it is to be a linear-and-thunkable isomorphism.

$(2 \to 3)$: the inclusion functor $\mathcal{D}_t^{\oplus} \to \mathcal{D}_{lt}$ is fully faithful, so in particular the isomorphisms $a \cong_t^{\oplus} b$ in $\mathcal{D}_t^{\oplus}$ are precisely the isomorphisms in $a \cong_{lt} b$ in $\mathcal{D}_{lt}$.

$(3 \to 2)$ (`is_duploid_univalent_from_positive_thunkable_and_negative_linear_categories`): we show that for any $a : \mathcal{D}$ the fan $\sum_{b:\mathcal{D}} a \cong_{lt} b$ is a proposition. By the preduploid property, assume that a is positive. Then fans $\sum_{b:\mathcal{D}} a \cong_{lt} b$ are equivalent to fans $\sum_{b:\mathcal{D}^{\oplus}} a \cong_t^{\oplus} b$ by lemma 3.8. The latter type is a proposition by assumption. The case is dual if a is negative instead. □

6 TODO The Univalent Duploid arising from an Adjunction

TODO: part of this will need to be merged with the construction of the oblique duploid, potentially in the previous chapter?

Let $L : \mathcal{P} \to \mathcal{N}$ and $R : \mathcal{N} \to \mathcal{P}$ be functors taking part in an adjunction $L \dashv_{(\eta,\epsilon)} R$ with unit η and counit ϵ, variously written

$$L \dashv_{(\eta,\epsilon)} R : \mathcal{N} \to \mathcal{P} \quad \text{or} \quad \mathcal{P} \underset{R}{\overset{L}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathcal{N} \ .$$

One way to view such an adjunction is as an isomorphism of sets

$$\mathcal{N}[\![Lp, n]\!] \overset{\varphi_{p,n}}{\simeq} \mathcal{P}[\![p, Rn]\!] \qquad \text{natural in } p : \mathcal{P}^{\mathsf{op}} \text{ and } n : \mathcal{N}. \tag{6.1}$$

Explicitly, φ may be given in terms of η and ϵ as

$$\varphi_{p,n}(Lp \overset{f}{\to} n) \overset{\text{def}}{\equiv} p \overset{\eta_p}{\longrightarrow} RLp \overset{Rf}{\longrightarrow} Rn \tag{6.2}$$

$$\varphi^{-1}_{p,n}(p \overset{g}{\to} Rn) \overset{\text{def}}{\equiv} Lp \overset{Lg}{\longrightarrow} LRn \overset{\epsilon_n}{\longrightarrow} n \,. \tag{6.3}$$

Following Levy [14], we may call the morphisms on either side of φ (6.1) the *oblique morphisms*, adopting the following definition.

Definition 6.1 (`oblique_mor`)**.** Let $\varphi_{p,n} : \mathcal{N}[\![Lp, n]\!] \cong \mathcal{P}[\![p, Rn]\!]$ be the natural isomorphism witnessing the adjunction $L \dashv R$. An *oblique morphism* in $L \dashv R$ from $p : \mathcal{P}$ to $n : \mathcal{N}$, written $f : \mathcal{O}_{L \dashv R}[\![p, n]\!]$, is defined as a pair of mates: a morphism $f^\flat : \mathcal{N}[\![Lp, n]\!]$ and a morphism $f^\sharp : \mathcal{P}[\![p, Rn]\!]$ such that $\varphi_{p,n}(f^\flat) = f^\sharp$. I will adopt the convention of drawing diagrams involving oblique morphisms in two columns, one for

the $\mathcal{N}$ mate and one for the $\mathcal{P}$ mate. For example, f on its own may be depicted as

$$Lp \xrightarrow{f^\flat} n \qquad\qquad p \xrightarrow{f^\sharp} Rn \quad .$$

It is sometimes convenient to depict an oblique morphism on just one diagram, as

$$p \xrightarrow{f} n \qquad \text{(a } \mathbf{coll}_{L \dashv R}\text{-diagram (6.2))}.$$

Where the adjunction is obvious I will freely omit it from $\mathcal{O}$. $\mathcal{O}$ extends to a functor $\mathcal{P}^{\mathsf{op}} \times \mathcal{N} \to \mathbf{Set}$ by pre- and post-composition in the respective categories, as

$$\begin{array}{ccc} Lp & & \\ {\scriptstyle Lf}\downarrow & & \\ Lq & \xrightarrow{g^\flat} & n \\ & & \downarrow{\scriptstyle h} \\ & & m \end{array} \qquad\qquad \begin{array}{ccc} p & & \\ {\scriptstyle f}\downarrow & & \\ q & \xrightarrow{g^\sharp} & Rn \\ & & \downarrow{\scriptstyle Rh} \\ & & Rm\,, \end{array}$$

with the two sides being mates by naturality of φ.

The projections $\flat$ and $\sharp$ have the obvious inverses

$$\flat^{-1} : \mathcal{N}[\![Lp, n]\!] \to \mathcal{O}[\![p, n]\!] \quad \text{and} \quad \sharp^{-1} : \mathcal{P}[\![p, Rn]\!] \to \mathcal{O}[\![p, n]\!]\,,$$

so that we have

$$\mathcal{N}[\![Lp, n]\!] \overset{\flat}{\simeq} \mathcal{O}[\![p, n]\!] \overset{\sharp}{\simeq} \mathcal{P}[\![p, Rn]\!] \qquad \text{natural in } p : \mathcal{P}^{\mathsf{op}} \text{ and } n : \mathcal{N}.$$

We would like to consider a category-like structure obtained from $L \dashv R$ which includes $\mathcal{P}$ and $\mathcal{N}$ as subcategories, as well as all oblique morphisms.

Definition 6.2. Define the *collage category* $\mathbf{coll}_{L \dashv R}$ to have:

- Objects those either from $\mathcal{P}$ or from $\mathcal{N}$, written $\iota^\oplus(p)$ and $\iota^\ominus(n)$ respectively. The injections $\iota^\oplus$ and $\iota^\ominus$ may be elided, particularly in diagrams.
- Morphisms obtained by taking the corresponding morphisms in $\mathcal{P}$, $\mathcal{N}$, or $\mathcal{O}_{L \dashv R}$

for the three well-defined pairs as follows

$$\mathbf{coll}_{L\dashv R}[\![\iota^{\oplus}(p), \iota^{\oplus}(q)]\!] \stackrel{\text{def}}{\equiv} \mathcal{P}[\![p, q]\!]$$
$$\mathbf{coll}_{L\dashv R}[\![\iota^{\ominus}(n), \iota^{\ominus}(m)]\!] \stackrel{\text{def}}{\equiv} \mathcal{N}[\![n, m]\!]$$
$$\mathbf{coll}_{L\dashv R}[\![\iota^{\oplus}(p), \iota^{\ominus}(n)]\!] \stackrel{\text{def}}{\equiv} \mathcal{O}[\![p, n]\!]\,.$$

There are no morphisms of the form $\mathbf{coll}_{L\dashv R}[\![\iota^{\ominus}(n), \iota^{\oplus}(p)]\!]$.

- Composition is given in either $\mathcal{P}$ or $\mathcal{N}$ by the functors $\mathcal{P}[\![\cdot,\cdot]\!]$, $\mathcal{N}[\![\cdot,\cdot]\!]$ or $\mathcal{O}[\![\cdot,\cdot]\!]$. This works because any morphisms of the form $x \to \iota^{\oplus}(p)$ or $\iota^{\ominus}(n) \to y$ *must* be of the form $\iota^{\oplus}(q) \to \iota^{\oplus}(p)$ or $\iota^{\ominus}(n) \to \iota^{\ominus}(m)$ respectively, and hence must lie fully in $\mathcal{P}$ or $\mathcal{N}$ rather than being oblique.

Duploids provide a natural way to extend $\mathbf{coll}_{L\dashv R}$ to include morphisms $\iota^{\ominus}(n) \to \iota^{\oplus}(p)$ that are currently absent, in the following way.

Definition 6.3. Observe that any object $p : \mathcal{P}$ gives rise to an oblique morphism $p^{\blacktriangleright} : \mathcal{O}[\![p, Lp]\!]$ with components

$$Lp \xrightarrow{\mathsf{id}_{Lp}} Lp \qquad\qquad p \xrightarrow{\eta_p} RLp \quad , \tag{6.4}$$

and any $n : \mathcal{N}$ to one $n^{\blacktriangleright} : \mathcal{O}[\![Rn, n]\!]$ with components

$$LRn \xrightarrow{\epsilon_n} n \qquad\qquad Rn \xrightarrow{\mathsf{id}_{Rn}} Rn \quad . \tag{6.5}$$

Taken together, for any object $a : \mathbf{coll}_{L\dashv R}$, define the *positive projection* $a^{\oplus} : \mathcal{P}$, *negative projection* $a^{\ominus} : \mathcal{N}$, and *cross morphism* $a^{\blacktriangleright} : \mathcal{O}[\![a^{\oplus}, a^{\ominus}]\!]$ of the object as

$$\left(a^{\oplus} \xrightarrow{a^{\blacktriangleright}} a^{\ominus}\right) \stackrel{\text{def}}{\equiv} \begin{cases} Rn \xrightarrow{n^{\blacktriangleright}} n & \text{if } a = \iota^{\ominus}(n) \\ p \xrightarrow{p^{\blacktriangleright}} Lp & \text{if } a = \iota^{\oplus}(p)\,. \end{cases} \tag{6.6}$$

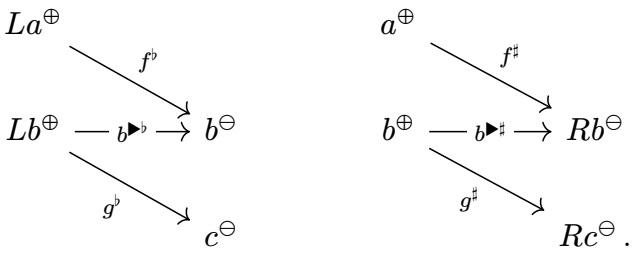
Definition 6.4. The *oblique duploid* over an adjunction $L \dashv R : \mathcal{N} \to \mathcal{P}$, written $\mathcal{O}[L \dashv R]$ is the duploid given by the following data.

- Objects are the objects of $\mathbf{coll}_{L\dashv R}$.

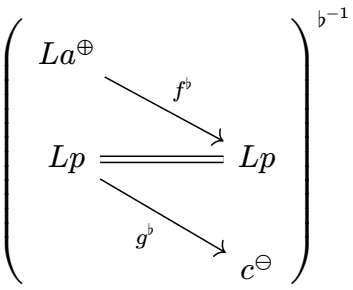
- Morphisms are the oblique morphisms between the positive and negative projections:
$$\mathcal{O}[L \dashv R][\![a, b]\!] \stackrel{\text{def}}{\equiv} \mathcal{O}_{L\dashv R}[\![a^{\oplus}, b^{\ominus}]\!] .$$

- Identities id_a are given by the cross morphisms $a^{\blacktriangleright}$ (6.6).

- Composition in $\mathcal{O}[L \dashv R]$ amounts to finding, given morphisms $f : \mathcal{O}[\![a^{\oplus}, b^{\ominus}]\!]$ and $g : \mathcal{O}[\![b^{\oplus}, c^{\ominus}]\!]$, an oblique morphism $f ; g : \mathcal{O}[\![a^{\oplus}, c^{\ominus}]\!]$. That is, some morphism $(f ; g)^{\flat} : \mathcal{N}[\![La^{\oplus}, c^{\ominus}]\!]$ or $(f ; g)^{\sharp} : \mathcal{P}[\![a^{\oplus}, Rc^{\ominus}]\!]$, given the data

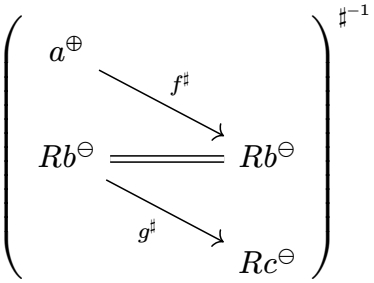


Composition must be such that $f = \mathsf{id}_b = b^{\blacktriangleright}$ or $g = \mathsf{id}_b$ would be cancelled. This would be easy if $b^{\blacktriangleright}$ had an inverse in either $\mathcal{N}$ or $\mathcal{P}$ – and it does: either $b^{\blacktriangleright^{\flat}}$ or $b^{\blacktriangleright^{\sharp}}$ is the identity depending on whether $b = \iota^{\oplus}(b^{\oplus})$ (6.4) or $b = \iota^{\ominus}(b^{\ominus})$ (6.5). So define the composite $a \xrightarrow{f} b \xrightarrow{g} c$ as



if $b = \iota^{\oplus}(p)$ (6.7)

and



if $b = \iota^{\oplus}(n)$. (6.8)

- Shifts are given by

$$\Downarrow a \stackrel{\text{def}}{\equiv} \iota^{\oplus}(a^{\oplus}) \quad \text{and} \quad \Uparrow a \stackrel{\text{def}}{\equiv} \iota^{\ominus}(a^{\ominus}) \quad \text{on objects} \tag{6.9}$$

$$\Downarrow a \xrightarrow{\mathrm{force}_a \stackrel{\text{def}}{\equiv} a^{\ominus\blacktriangleright}} a \tag{6.10}$$

$$a \xrightarrow{\mathrm{delay}_a \stackrel{\text{def}}{\equiv} a^{\blacktriangleright}} \Downarrow a \tag{6.11}$$

$$a \xrightarrow{\mathrm{wrap}_a \stackrel{\text{def}}{\equiv} a^{\oplus\blacktriangleright}} \Uparrow a \tag{6.12}$$

$$\Uparrow a \xrightarrow{\mathrm{unwrap}_a \stackrel{\text{def}}{\equiv} a^{\blacktriangleright}} a\,. \tag{6.13}$$

Lemma 6.5. *Let $\mathcal{P}$ and $\mathcal{N}$ be univalent categories, and $L \dashv R : \mathcal{P} \to \mathcal{N}$ a fully equalizing adjunction between them. The following categories characterize the subcategories of the envelope duploid $\mathcal{E}[L \dashv R]$.*

- *The positive thunkable category $\mathcal{E}[L \dashv R]^{\oplus}_{t}$ is identified with $\mathcal{P}$.*
- *The negative linear category $\mathcal{E}[L \dashv R]^{\ominus}_{l}$ is identified with $\mathcal{N}$.*
- *The positive subcategory $\mathcal{E}[L \dashv R]^{\oplus}$ is identified with the non-univalent Kleisli category $\mathbf{Kleisli}_{\not{\text{univ}}}((,\mathcal{P}), RL)$.*
- *The negative subcategory $\mathcal{E}[L \dashv R]^{\ominus}$ is identified with the non-univalent Co-Kleisli category $\mathbf{CoKleisli}_{\not{\text{univ}}}((,\mathcal{N}), LR) \equiv \mathbf{Kleisli}_{\not{\text{univ}}}((,\mathcal{N})^{\mathsf{op}}, (LR)^{\mathsf{op}})$.*

7 TODO Related Work

Bibliography

[1] B. Ahrens, K. Kapulkin, and M. Shulman, "Univalent categories and the rezk completion," *Mathematical Structures in Computer Science*, vol. 25, no. 5, pp. 1010–1039, 2015. DOI: 10.1017/S0960129514000486

[2] B. Ahrens, P. R. North, M. Shulman, and D. Tsementzis, *The Univalence Principle* (Memoirs of the American Mathematical Society). 2025, vol. 305. DOI: 10.1090/memo/1541 arXiv: 2102.06275.

[3] A. Church, "A formulation of the simple theory of types," *Journal of Symbolic Logic*, vol. 5, no. 2, pp. 56–68, 1940. DOI: 10.2307/2266170

[4] P. Clairambault and G. Munch-Maccagnoni, "Duploid situations in concurrent games," in *12th Workshop on Games for Logic and Programming Languages (GaLoP XII)*, Juha Kontinen, Marina Lenisa, Uppsala, Sweden, Apr. 2017. [Online]. Available: https://inria.hal.science/hal-01991555

[5] T. Coquand and N. A. Danielsson, "Isomorphism is equality," *Indagationes Mathematicae*, vol. 24, no. 4, pp. 1105–1120, 2013, In memory of N.G. (Dick) de Bruijn (1918–2012), ISSN: 0019-3577. DOI: 10.1016/j.indag.2013.09.002 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0019357713000694

[6] M. H. Escardó, *A self-contained, brief and complete formulation of voevodsky's univalence axiom*, 2018. arXiv: 1803.02294 [math.LO]. [Online]. Available: https://arxiv.org/abs/1803.02294

[7] C. Führmann, "Direct models of the computational lambda-calculus," *Electronic Notes in Theoretical Computer Science*, vol. 20, pp. 245–292, 1999, MFPS XV, Mathematical Foundations of Progamming Semantics, Fifteenth Conference, ISSN: 1571-0661. DOI: 10.1016/S1571-0661(04)80078-1 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571066104800781

[8] D. R. Grayson, "An introduction to univalent foundations for mathematicians," *Bull. Amer. Math. Soc.*, vol. 55, pp. 427–450, 2018, ISSN: 1088-9485. DOI: 10.1090/bull/1616

[9] D. R. Grayson et al., *Unimath/unimath: V20250923*, version v20250923, Sep. 2025. DOI: 10.5281/zenodo.17186647

[10] M. Hofmann and T. Streicher, "The groupoid model refutes uniqueness of identity proofs," in *Proceedings Ninth Annual IEEE Symposium on Logic in Computer Science*, 1994, pp. 208–212. DOI: 10.1109/LICS.1994.316071

[11] C. Kapulkin, P. L. Lumsdaine, and V. Voevodsky, *Univalence in simplicial sets*, 2018. arXiv: 1203.2553 [math.AT]. [Online]. Available: https://arxiv.org/abs/1203.2553

[12] K. Kapulkin and P. L. Lumsdaine, "The simplicial model of univalent foundations (after voevodsky)," *Journal of the European Mathematical Society*, vol. 23, no. 6, pp. 2071–2126, Mar. 2021, ISSN: 1435-9863. DOI: 10.4171/jems/1050

[13] J. Lambek and P. J. Scott, *Introduction to higher order categorical logic*. USA: Cambridge University Press, 1986, ISBN: 0521246652.

[14] P. B. Levy, "Adjunction models for call-by-push-value with stacks," *Electronic Notes in Theoretical Computer Science*, vol. 69, pp. 248–271, 2003, CTCS'02, Category Theory and Computer Science, ISSN: 1571-0661. DOI: 10.1016/S1571-0661(04)80568-1

[15] P. B. Levy, "Call-by-push-value: A subsuming paradigm," in *Typed Lambda Calculi and Applications*, J.-Y. Girard, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 228–243, ISBN: 978-3-540-48959-7.

[16] P. B. Levy, "Contextual isomorphisms," *SIGPLAN Not.*, vol. 52, no. 1, pp. 400–414, Jan. 2017, ISSN: 0362-1340. DOI: 10.1145/3093333.3009898

[17] É. Mangel, P.-A. Melliès, and G. Munch-Maccagnoni, "Classical notions of computation and the hasegawa-thielecke theorem," *Proc. ACM Program. Lang.*, vol. 10, no. POPL, Jan. 2026. DOI: 10.1145/3776715

[18] P. Martin-Löf, "Constructive mathematics and computer programming," in *Logic, methodology and philosophy of science, VI (Hannover, 1979)*, ser. Stud. Logic Found. Math. Vol. 104, North-Holland, Amsterdam, 1982, pp. 153–175. DOI: 10.1016/S0049-237X(09)70189-2

[19] E. Moggi, "Computational lambda-calculus and monads," in *[1989] Proceedings. Fourth Annual Symposium on Logic in Computer Science*, 1989, pp. 14–23. DOI: 10.1109/LICS.1989.39155

[20] E. Moggi, "Notions of computation and monads," *Information and Computation*, vol. 93, no. 1, pp. 55–92, 1991, Selections from 1989 IEEE Symposium on Logic in Computer Science, ISSN: 0890-5401. DOI: 10.1016/0890-5401(91)90052-4

[21] G. Munch-Maccagnoni, "Models of a non-associative composition," in *Foundations of Software Science and Computation Structures*, A. Muscholl, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 396–410, ISBN: 978-3-642-54830-7. DOI: 10.1007/978-3-642-54830-7_26

[22] G. Munch-Maccagnoni, "Syntax and Models of a non-Associative Composition of Programs and Proofs," Theses, Université Paris-Diderot - Paris VII, Dec. 2013. [Online]. Available: https://theses.hal.science/tel-00918642

[23] E. Rijke, *Introduction to Homotopy Type Theory* (Cambridge Studies in Advanced Mathematics). Cambridge University Press, 2025.

[24] T. Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013. [Online]. Available: https://homotopytypetheory.org/book

Index