On the Semantics of Intensionality and Intensional Recursion

G. A. Kavvos
St John's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Trinity Term 2017

This thesis is dedicated to my father, John G. Kavvos (1948–2015), who taught me to persist in the face of all adversity.

Acknowledgements

A thesis in the making is a process of controlled deconstruction of its author's character. This fact in itself suffices to warrant a rich list of acknowledgees.

First and foremost, I would like to thank my doctoral supervisor, Samson Abramsky, *sine qua non*. Not only did he suggest the—admittedly unusual—topic of this thesis, but his unfailingly excellent advice and his unparalleled wit were vital ingredients in encouraging me to plough on, even when it seemed futile to do so. I shall never forget the time he quoted the experience of J. E. Littlewood on the development of chaos theory:

> "For something to do we went on and on at the thing with no earthly prospect of "results"; suddenly the whole vista of the dramatic fine structure of solutions stared us in the face."

It was a pleasure to work for a few short years next to a scientist of his stature, who also happens to be a kind man with a wonderful sense of humour.

I also want to thank my examiners, Luke Ong and Martin Hyland, who examined this work in detail, and provided many enlightening comments and suggestions.

I am grateful to the UK Engineering and Physical Sciences Research Council (EPSRC) for their financial support. I am also greatly indebted to the Department of Computer Science, the European Cooperation in Science and Technology (COST) framework, and St John's College, for generously funding the trips involved in presenting my work to the wider community.

I would also like to thank all those who reviewed parts of this work before its completion. These include quite a few anonymous reviewers; John Longley, who toiled over the first version of the manuscript on which this thesis is based; Neil Jones, with whom I have had many interesting discussions during and after his visits to Oxford, and who also read many

parts of this thesis in detail; and my interim assessors within the department, Sam Staton and Hongseok Yang. This thesis could hardly have been completed without their support and encouragement.

Many thanks to the fellow scientists around me: Kohei Kishida, for many interesting discussions, for providing help and advice at a moment's notice, and for translating and transliterating the title of [Kiselyov, 2015]; Martin Berger, for inviting me to Sussex to present my work, and for the numerous exchanges that followed; and Sean Moss, who was eager to discuss all sorts of interesting categorical diversions. Many thanks to Geraint Jones for allowing me to use his macros for Eindhoven-style calculations.

I also want to thank my fellow students: Mario Alvarez, to whom I wish the best of luck in his ongoing attempt to 'debunk' this thesis; Amar Hadzihasanovic, for many discussions on higher category theory and its applications; and Matthijs Vákár and Norihiro Yamada, who were always eager to discuss categories, logic, and semantics.

Finally, I would like to thank Andrew Ker, who was my tutor during my time as an undergraduate at University College, and my employer thereafter. I learned an enormous amount from him as a student, and even more whilst teaching under his supervision. I am indebted to him for his wise advice, and all the things that he has inadvertently taught me over a cup of coffee in the Senior Common Room. One would be hard-pressed to find better guidance or more generosity.

Abstract

Intensionality is a phenomenon that occurs in logic and computation. In the most general sense, a function is intensional if it operates at a level finer than (extensional) equality. This is a familiar setting for computer scientists, who often study different programs or processes that are interchangeable, i.e. extensionally equal, even though they are not implemented in the same way, so intensionally distinct. Concomitant with intensionality is the phenomenon of intensional recursion, which refers to the ability of a program to have access to its own code. In computability theory, intensional recursion is enabled by Kleene's Second Recursion Theorem.

This thesis is concerned with the crafting of a logical toolkit through which these phenomena can be studied. Our main contribution is a framework in which mathematical and computational constructions can be considered either *extensionally*, i.e. as abstract values, or *intensionally*, i.e. as fine-grained descriptions of their construction. Once this is achieved, it may be used to analyse intensional recursion.

To begin, we turn to type theory. We construct a modal λ-calculus, called Intensional PCF, which supports non-functional operations at modal types. Moreover, by adding Löb's rule from provability logic to the calculus, we obtain a type-theoretic interpretation of intensional recursion. The combination of these two features is shown to be consistent through a confluence argument.

Following that, we begin searching for a semantics for Intensional PCF. We argue that 1-category theory is not sufficient, and propose the use of P-categories instead. On top of this setting we introduce *exposures*, which are P-categorical structures that function as abstractions of well-behaved intensional devices. We produce three examples of these structures, based on Gödel numberings on Peano arithmetic, realizability theory, and homological algebra.

The language of exposures leads us to a P-categorical analysis of intensional recursion, through the notion of *intensional fixed points*. This, in turn, leads to abstract analogues of classic intensional results in logic and computability, such as Gödel's Incompleteness Theorem, Tarski's Undefinability Theorem, and Rice's Theorem. We are thus led to the conclusion that exposures are a useful framework, which we propose as a solid basis for a *theory of intensionality*.

In the final chapters of the thesis we employ exposures to endow Intensional PCF with an appropriate semantics. It transpires that, when interpreted in the P-category of assemblies on the PCA K_1, the Löb rule can be interpreted as the type of Kleene's Second Recursion Theorem.

Contents

List of Figures

This is the ORA version of this thesis (identical to build 7049), compiled on February 1, 2018.

Chapter 1

Introduction

This thesis concerns the computational phenomenon of intensionality.

1.1 Intensionality

To be *intensional* is to contain not only *reference*, but also *sense*. This philosophical distinction was drawn by Frege; see Fitting [2015]. An intensional sign denotes an external *referent*, yet inherently connotes more information—its elusive *sense*. The classic example is that of the planet Venus, which may be referred to as either the morning star, or the evening star.

In the most mathematically general sense, to be 'intensional' is to somehow operate at a level finer than some predetermined 'extensional' equality. Intensionality is omnipresent in constructive mathematics, where the question of equality is non-trivial, see e.g. Beeson [1985]. An example that dates back to the work of Bishop [1967] on constructive analysis is that of real numbers, and their construction as Cauchy sequences of rationals: two different Cauchy sequences of rationals may stand for the same real number, thus being *extensionally* equal, yet *intensionally* distinct.

Most mainstream mathematics is extensional: we usually reason about some underlying, 'ideal' mathematical object, and not its concrete descriptions. The latter is only a way to refer to the former. In this light, intensionality is merely a nuisance, so common set theories assume some *axiom of extensionality*: sets are identified by their members, and functions by their graphs. Glimpses of intensionality appear very rarely, and usually only because we are interested in proving that some extensionality axiom is independent from the rest of some logical theory: see e.g. Streicher [2015] for a recent example.

This is a difficult-to-work-in setting for Computer Science, where intensionality is the norm. In fact, the present author believes that it would be fair to say that many

branches of computer science are in essence the study of programs and processes seen under some appropriate notion of equality. At one end of the spectrum, correctness of programs is discussed in the context of some relation of *observational equivalence*, i.e. indistinguishability of programs. Intermediately, complexity theory requires a slightly stronger notion, which we could call *complexity equivalence*: this is observational equivalence strengthened by some account of the resources the program consumes. At the other extreme, computer viruses sometimes make decisions based strictly on patterns of object code they encounter, disregarding the actual function of what they are infecting; one could say they operate up to α*-equivalence*, i.e. syntactic identity. Each of the aforementioned notions of equality is more intensional than the one preceding it, and each level is interesting in its own right.

Thus, depending on our point of view, there are always two ways in which we can see a computational process. There is an extensional level, which corresponds to *what may be computed*. But there is also an intensional level, corresponding to the *programs and processes that carry out the computation*. The shift between these two viewpoints has been discussed by Moschovakis [1993] and Abramsky [2014]: computational processes may be understood by the ideal objects that they refer to (e.g. functions), or their internal characteristics: length, structure, and, ultimately, the *algorithm* they embody.

This thesis concerns the difficulties that arise when we are trapped between *intension* and *extension*, or *description* and *behaviour*.

1.2 Objectives

The main objective of this thesis is to answer the following question:

> Is there a consistent, logical universe where the same mathematical objects can be viewed both as intension, and extension?

This research programme is a suggestion of Abramsky [2014]; in his words:

> The notions of intensionality and extensionality carry symmetric-sounding names, but this apparent symmetry is misleading. Extensionality is enshrined in mathematically precise axioms with a clear conceptual meaning. Intensionality, by contrast, remains elusive. It is a "loose baggy monster" into which all manner of notions may be stuffed, and a compelling and coherent general framework for intensional concepts is still to emerge.

Our discussion in the previous section hints at the fact that the choice of what is extensional, and thereby what is intensional is entirely up to us. There are many shades on the spectrum between ideal mathematical object and full symbolic description, and the choice of perspective depends on what we wish to study. However, the design of a mathematical framework or universe where both the extensions and intensions of our choice co-exist harmoniously is the difficult and well-defined problem with which this thesis is concerned.

The present author believes that such a framework will be instrumental in making progress in providing answers to the following questions:

1. What is the meaning of *intensional programming*? Is there a logical interpretation of it?

2. What is the meaning of *reflective programming*? Is it possible to program reflectively in a consistent way?

3. What is the meaning of *intensional operations* in a *higher-order setting*? How can we have a non-functional operation without resorting to first-order manipulations of Gödel numbers?

4. How can we add recursion to type theory?

The theory developed in this thesis fully addresses the first two of these questions. Considered as a toolkit, it is likely to be useful in answering the third one. The fourth one is left for future work. Nevertheless, we shall now briefly consider all of them.

1.2.1 Intensional Programming

In the realm of *functional computation*, we can immediately distinguish two paradigms:

- **The Extensional Paradigm**. It has been exactly 50 years since Christopher Strachey articulated the notion of *functions as first-class citizens* in his influential notes on programming [Strachey, 2000]. In a purely functional world, a higher-order program can use a functional argument extensionally by evaluating it at a finite number of points: this leads to a form of *continuity*, which is the basis of *domain theory* [Abramsky and Jung, 1994].

- **The Intensional Paradigm**. This way of computing originated in *computability theory* [Cutland, 1980, Jones, 1997]: a program can compute with the *source code*—or *intension*—of another program as an argument. It can edit, optimise, call, or simulate the running of this code.

Whereas the first paradigm led to a successful research programme on the *semantics of programming languages*, the second is often reduced to *symbolic evaluation*. This is one of the reasons for which the intensional paradigm has not reached the sophistication of its extensional counterpart. Yet, the question remains: what can the intensional paradigm contribute to programming? What is the additional expressivity or programming power afforded by intensionality?

This is a theme that is often discussed by the LISP community. Indeed, certain dialects of LISP are the closest we have to a true paradigm of intensional programming, both through LISP *macros* [Graham, 1993] as well the construct of *quoting* [Bawden, 1999]. Either of these features can also be used for *metaprogramming*, which is the activity of writing programs that write other programs.

The notion of intensional programming is also central to the work of the *partial evaluation community*, which uses a rather extreme form of intensionality that is better known under the name *programs-as-data*: see Jones et al. [1993], Jones [1996, 1997]. Their work uses insights from computability theory and Gödel numberings to build a metaprogramming-oriented methods for the automatic generation of compilers, all based on the power of the *s-m-n theorem* of computability theory: see §2.1.4 for more details and references. Even though LISP-inspired, and never seriously considering non-functional operations, this is perhaps the closest anybody has come to what we mean by intensional programming.

But, to this day, no satisfying theoretical account of intensionality in programming has been produced. The present author believes that this has to do with the fact that LISP is an unstructured, untyped language. Hence, it is not amenable to any kind of analysis, other than maybe that of the untyped λ-calculus [Barendregt, 1984]. See also Wadler [1987] for an early critique of SCHEME in programming methodology.

We shall introduce a new approach to intensional programming that is fundamentally *typed* in §3.

1.2.2 Reflective Programming

For a very long time, programmers and theoreticians have sought to understand *computational reflection*, a concept introduced by Brian Cantwell Smith [1982, 1984]. Computational reflection is an obscure idea that has been used in a range of settings: see Demers and Malenfant [1995] for a (slightly dated) survey. In broad strokes, it refers to the ability of a program to *access its own code*, *refer to its own description*, or *examine its own internals*.

The kind of reflection envisaged by Brian Cantwell Smith was to be implemented by using *reflective towers*. The idea is that a program could be understood as running on the topmost level of an infinity of interpreters. Reflective constructs then accept code that is to be injected to the interpreter situated one level below, hence having access to the complete state of the top-level interpreter, all its registers and variables, and so on—to infinity. This is a rather mysterious construction with unclear semantics that have been the subject of investigation by Friedman and Wand [1984], Wand and Friedman [1988], and Danvy and Malmkjaer [1988]. This line of work eventually concluded with a theorem of Wand [1998], which shows that there are no useful semantic descriptions of the reflective tower. We will discuss Wand's result in §1.3.2.

Despite its being poorly understood, computational reflection seems to be a recurring and useful concept. Demystifying it is a pressing problem, as many modern programming languages have reflective or 'introspective' facilities that lead to pernicious bugs. The author suspects that a central theme of this thesis, the notion of *intensional recursion*, will prove fundamental in obtaining a logical foundation for reflection. To quote Polonsky [2011]:

> Of course, they [the questions posed by Smullyan] are only a sliver in the more global puzzle of understanding reflection as a distinct phenomenon. There is still lacking a general concept, an all-inclusive definition through which the common features of the constructions in Gödel's theorem, computability, number theory (systems of arithmetic), and set theory could be related. Finding such a concept remains a fascinating open problem.

We believe that the notion of *intensional fixed points*, to which we devote §6, is precisely an abstract definition of (well-behaved) reflection. Our framework thereby provides a candidate solution to the problem of Polonsky.

1.2.3 Higher-Order Non-Functional Computation

Over the past years, a new field of theoretical computer science has emerged under the name of *higher-order computability*. As a subject, higher-order computability has its roots in the 1950s, but the work of Longley [2005] has produced a unifying account that overlaps significantly with the study of logic, λ-calculus, category theory, realizability theory, and the semantics of programming languages; see the recent book by Longley and Normann [2015].

In classical computability theory [Rogers, 1987, Cutland, 1980, Odifreddi, 1992] there was a *confluence of ideas* [Gandy, 1988] in the 1930s, culminating in the *Church-Turing thesis*, viz. that all 'effectively calculable' functions—whatever that might mean—are partial recursive. In contrast, higher-order computability suffers from a *Church-Turing anti-thesis*: there are multiple notions of computation at higher order, and some of them can be shown to be strongly incompatible with each other. This situation generates a lot of debate regarding which notion of higher-order computability is 'more natural' than the others: see the discussion in [Longley, 2005, §1]

Perhaps one of the most difficult challenges in higher-order computability is to clarify what intensional, or *non-functional higher-order computation* is. This does not only pertain to computation with the usual effects, like memory, exceptions, or first-class continuations: such effects are well-understood, either through *game semantics* [Abramsky and McCusker, 1996, Abramsky et al., 1998], or more abstractly through the various theories of effects: see [Moggi, 1989, 1991, Plotkin and Power, 2004, Hyland and Power, 2007, Levy, 2003]. Instead, we consider more general non-functional computation, where the non-functional aspect arises from the ability of a device to read the description or code of its higher-order argument.

A known example that is impossible to accommodate in an extensional setting is the *modulus of continuity functional*. This is a type 3 functional,

$$\Phi : ((\mathbb{N} \to \mathbb{N}) \to \mathbb{N}) \to \mathbb{N}$$

Intuitively, when given a type 2 functional $F : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$, $\Phi(F)$ returns an *upper bound* n on the range of values F would examine of any argument given to it, so that if $f, g : \mathbb{N} \to \mathbb{N}$ agree on those values, i.e.

$$\forall i \in \{0, \ldots, n\}.\ f(i) = g(i)$$

then $Ff = Fg$. It can be shown that one cannot define a modulus of continuity functional that is extensional. To compute $\Phi(F)$ it is necessary to examine the internals of F: we simulate its run on some function, and see what the maximum argument it examines is. This can be computed in a language with side-effects: we call $F(f_\spadesuit)$, where $f_\spadesuit$ uses side-effects to record the maximum value at which it is called, and $\Phi(F)$ then returns this recorded value. We can see that this is highly dependent on the way F is implemented, and thus not extensional: see the blog post of Andrej Bauer [2011]. The question we want to ask in this setting is this: can we make the modulus of continuity computable, perhaps by admitting it at *certain intensional types only*, but *without* generally violating neither extensionality nor freedom from side effects?

The main ways to understand such computation are either through a 'highly intensional,' essentially untyped, first-order formulation, which suffers from a lack of logical structure and useful properties; or through computational effects, which is even less clear in terms of higher-order computability. One way out of this impasse has been suggested by Longley [Longley, 2005, §6], and it employs realizability. Longley argues that posing the question of whether a non-functional operation is definable amounts to writing down a logical formula that specifies it, and then examining whether that formula is *realizable*: see [Longley, 1999a] and the note [Longley, 1999b].

Whereas this is an informative approach, it does not smoothly lead us toward the design of programming languages that harness these non-functional powers. We propose, instead, the following research programme: we should first use the type-theoretic techniques proposed in this thesis to add intensionality to a λ-calculus. Then, we shall be able to vary the intensional operations available, and study the expressivity of the resulting systems.

1.2.4 Recursion in Type Theory

General recursive definitions are prohibited in most type theories, including *Martin-Löf type theory (MLTT)* [Martin-Löf, 1998, 1984, Nordström et al., 1990]. This is not an arbitrary design choice: MLTT features *dependent types*, i.e. types depend on terms. Thus, its terms ought to be strongly normalising, so that the types themselves are.

In that way, MLTT is logically well-behaved, but not ideal for programming. In fact, writing down the definitions of many ordinary programs becomes a difficult exercise. The intuitive reason is that every program we want to write must somehow contain its own proof of termination. A lot of work has gone into regaining the lost expressivity, from the results of Constable and Smith [1993] on *partial types*, to the method of Bove and Capretta [2005], and all the way to the coinductive types of Capretta [2005] and the *delay monad*. There is also recent work on the *partiality monad* and higher inductive types: see Altenkirch et al. [2017].

In any case, working within MLTT has a fundamental disadvantage: an old theorem of Blum [1967] ensures that, no matter what kind of 'blow-up factor' we choose, we will be able to write a program (in a partial, Turing-complete language) whose shortest equivalent in a total programming language is larger by the given factor: see the blog post of Harper [2014]. Thus, ultimately, there is no working around this theorem: at some point we might have to add general recursion to type theory.

In the setting of *simple types* adding recursion is relatively easy. It was first done by Scott [1993] and Plotkin [1977], who defined the prototypical fixed point language PCF, a simply-typed λ-calculus of booleans, integers, and a fixed point combinator $\mathbf{Y} : (A \to A) \to A$. There is a long and rich literature on the models of PCF: the reader may consult any of [Plotkin, 1977, Gunter, 1992, Mitchell, 1996, Abramsky et al., 1996, Hyland and Ong, 2000, Streicher, 2006, Longley, 1995].

Merely throwing $\mathbf{Y}$ into the mixture is not advisable in more expressive type theories, e.g. in System F or MLTT. If sum or Σ types are available (or are definable, e.g. in System F), a theorem of Huwig and Poigné [1990] guarantees that the resulting theory will be inconsistent: even the existence of the coproduct $\mathbf{1} + \mathbf{1}$ is enough to make a cartesian closed category with fixpoints degenerate to a *preorder*.

A short abstract by Plotkin [1993] proposes that we forget the cartesian setting, and work in either a (second-order) *intuitionistic linear type theory*, or a *relevant type theory*. The full manuscript for that abstract never appeared, but the linear type system to which it refers (but without recursion) was studied in detail by Barber [1996].

Further efforts went into defining a *Linear PCF*, based on these intuitions. Invariably, the type of $\mathbf{Y}$ or the rule for recursion has one of the following forms:

$!(!A \multimap A) \multimap A$	Braüner [1995, 1997]
$\dfrac{!(!A \multimap A)}{A}$	Maraist et al. [1995], Bierman [2000]
$\dfrac{!(!A \multimap A)}{!A}$	Maraist et al. [1995]

We will shortly see that this pattern is not accidental. We will develop a type-theoretic approach to intensionality that will admit a slightly unusual kind of recursion, namely *intensional recursion*. Its type-theoretic interpretation will be strongly reminiscent of the pattern of these rules, but linearity will prove to be a red herring.

1.3 Quoting is Impossible

The first impression that one usually acquires regarding intensional phenomena is that they can only spell trouble. After all, encoding programs or logical formulae as data *in the same language* is the typical function of *Gödel numbering*. Such constructs quickly lead to *negative* theorems that pinpoint the inherent limitations of logical systems, e.g. Gödel's *First Incompleteness Theorem* [Smullyan, 1992].

Our aim in this thesis is to tell a *positive story*, but we shall first recount the negative tales of the past. We shall not engage in a foundational debate regarding Gödel's theorems and related results here, but a more in-depth discussion can be found in §6, in the book of Smullyan [1992], and in [Girard, 2011, §2]. Instead, we will focus on the negative repercussions on programming.

1.3.1 Tale 1: Quoting is Not Definable

Let Λ be the set of untyped λ-terms [Barendregt, 1984], and let

$$\ulcorner \cdot \urcorner : \Lambda \to \Lambda$$

be a map on λ-terms. The intention is that, for each λ-term M, the term $\ulcorner M \urcorner$ represents the *program* M as a *datum* in the λ-calculus. We call $\ulcorner M \urcorner$ the *quote of* M. The properties of such (external) quoting functions $\ulcorner \cdot \urcorner : \Lambda \to \Lambda$ have been systematically studied by Polonsky [2011], who also broadly surveys the sporadic literature on such 'meta-encodings' of the λ-calculus into itself.

In standard accounts, e.g. [Barendregt, 1984], the quote function is a *Gödel numbering*, as known from the literature on Gödel's theorems. To each term M one assigns a number $\#M$, and defines $\ulcorner M \urcorner$ to be the Church numeral for $\#M$.

A fundamental question then arises: *is quoting internally definable?* The answer is, of course, negative, as internal definability of quoting would lead to inconsistency. The following argument is due to Barendregt [1991]: suppose there is a term $\mathbf{Q} \in \Lambda$ such that

$$\mathbf{Q}\, M =_\beta \ulcorner M \urcorner$$

It is a fact that Church numerals are in normal form. Hence, both $\ulcorner \mathbf{I} \urcorner$ and $\ulcorner \mathbf{I}\, \mathbf{I} \urcorner$ are in normal form, where $\mathbf{I} \stackrel{\text{def}}{=} \lambda x.\, x$ is the identity combinator. We have that

$$\ulcorner \mathbf{I} \urcorner =_\beta \mathbf{Q}\, \mathbf{I} =_\beta \mathbf{Q}\, (\mathbf{I}\, \mathbf{I}) =_\beta \ulcorner \mathbf{I}\, \mathbf{I} \urcorner$$

This amounts to equating two distinct normal forms. But it is known that the λ-calculus is confluent, hence consistent! It follows that such a term $\mathbf{Q}$ cannot exist.

It is very important to notice that the crux of the argument essentially rests on the confusion between the extension M, the intension $\ulcorner M \urcorner$, and the ability to pass *from extension to intension*. To obtain a consistent account of intensionality we ought to forbid this possibility. However, let us seize the opportunity to remark that the opposite direction is not only attainable, but a result of historical importance for computability theory [Kleene, 1936, 1981]:

Theorem 1 (Kleene [1936])**.** *There exists a term* $\mathbf{E} \in \Lambda^0$ *such that*

$$\mathbf{E} \ulcorner M \urcorner \twoheadrightarrow_\beta M$$

for all $M \in \Lambda^0$*.*

This is essentially the same result as the one obtained by Turing [1937], but for the λ-calculus instead of the Turing Machine. See [Barendregt, 1984, §8.1.6] for a proof.

1.3.2 Tale 2: Quoting Collapses Observational Equivalence

Let us suppose that the above result does not deter us in our plans to add reflection to the λ-calculus. All else failing, we can do so by postulating constants **eval** and **fexpr**, along with the following reductions:

$$\mathbf{eval} \ulcorner M \urcorner \longrightarrow M$$
$$(\mathbf{fexpr}\ V)\ M \longrightarrow V \ulcorner M \urcorner$$

where V is some notion of value (e.g. a weak head normal form). Then **Q** would be definable as **fexpr** $(\lambda x.\, x)$.

One of the most interesting and well-studied notions in λ-calculus is that of *observational equivalence*. Two terms M, N are observationally equivalent, written $M \cong N$, if they are interchangeable in all possible contexts, without any 'observable changes.' The notion of observable change is up for debate, and the exploration of different options leads to interesting variations—see e.g. Bloom and Riecke [1989] and Abramsky [1990]. The usual choice is that we can observe normal forms at ground type (i.e. different numerals and boolean values), or—equivalently—termination of evaluation at ground type. If $M =_\beta N$, then certainly $M \cong N$, but the converse is not normally true: terms can be observationally equivalent, but the theory of equality (which is only computably enumerable) is not strong enough to show that.

In this context, Wand [1998] showed the following theorem:

Theorem 2 (Wand)**.** $M \cong N$ *if and only if* $M \equiv_\alpha N$

Wand's definition of observation is termination at a weak head normal form, and his quoting function $\ulcorner \cdot \urcorner$ is a Scott-Mogensen encoding: see [Polonsky, 2011, Mogensen, 1992]. However, the result is strong and general, and puts the last nail in the coffin: there can be no semantic study of functional programming languages that are so strongly reflective that they can internally define quoting. Such a language can internally distinguish any two terms in the language, as long as they are not equal up to renaming. This result concludes the long line of research on Smith's reflective towers that we mentioned in §1.2.2.

1.4 Intensionality and Types: Modality-as-Intension

The two negative tales we have just told are, fortunately, not the final word. If we take a close look at the existing literature, e.g. at the chapters of [Barendregt, 1984] that concern computability and diagonal arguments, we may see that some intensional operations are indeed definable. For example, there are λ-terms $\mathbf{gnum}, \mathbf{app}$ and $\mathbf{E}$ such that

$$\begin{aligned} \mathbf{gnum}\ \ulcorner M \urcorner &=_\beta \ulcorner \ulcorner M \urcorner \urcorner \\ \mathbf{app}\ \ulcorner M \urcorner \ulcorner N \urcorner &=_\beta \ulcorner M\, N \urcorner \\ \mathbf{E}\ \ulcorner M \urcorner &=_\beta M \end{aligned}$$

We mentioned $\mathbf{E}$ in the previous section, but the other two might come as a surprise: they are indeed pure operations on syntax. The pattern that seems to be at work here is that the only true restriction is quoting, and that everything else is admissible.

The standard way to enforce restriction in computation is to use *types*. Indeed, the main contribution of this thesis is a detailed investigation and analysis of the following idea:

> Intensionality adheres to a typing discipline.

This is an old but not so well-known observation. To the best of our knowledge, it was first formulated by Neil Jones,[1] and led to his work on *underbar types*: see [Jones et al., 1993, §16.2].

1.4.1 Modality-as-Intension

Given a type A, let us write $\mathsf{Code}(A)$ for the *type of code of type A*. We must certainly have that, if $M : A$, then $\ulcorner M \urcorner : \mathsf{Code}(A)$. Thus, if $\mathbf{gnum}\ \ulcorner M \urcorner =_\beta \ulcorner \ulcorner M \urcorner \urcorner$, then it must have the type

$$\mathbf{gnum} : \mathsf{Code}(A) \to \mathsf{Code}\,(\mathsf{Code}(A))$$

This looks familiar! If we drop all pretense and write $\Box A$ for $\mathsf{Code}(A)$, we obtain the following types:

$$\begin{aligned} \mathbf{gnum} &: \Box A \to \Box\Box A \\ \mathbf{app} &: \Box(A \to B) \to \Box A \to \Box B \\ \mathbf{E} &: \Box A \to A \end{aligned}$$

[1] Personal communication.

Surprisingly, there is an underlying Curry-Howard correspondence: the types of these operations correspond to the axioms $\mathsf{4}$, K and T of the modal logic $\mathsf{S4}$. This connection to modal logic was drawn by Davies and Pfenning [1996, 2001], who used it to embed two-level λ-calculi [Nielson and Nielson, 1992] in a modal λ-calculus, and to perform binding-time analysis.

We will thus introduce and study the *modality-as-intension* interpretation. This is an idea that pervades [Davies and Pfenning, 1996, 2001], and is even mentioned in name in the conclusion of [Pfenning and Davies, 2001]. To quote:

> One particularly fruitful interpretation of $\Box A$ is as the intensional type for expressions denoting elements of type A. Embedding types of this form in a programming language means that we can compute with expressions as well as values. The term $\mathsf{box}\, M$ quotes the expression M, and the construct $\mathsf{let\ box}\, u \Leftarrow M\, \mathsf{in}\, N$ binds u to the expression computed by M and then computes the value of N. The restrictions placed on the introduction rule for $\Box A$ mean that a term $\mathsf{box}\, M$ can only refer to other expression variables u but not value variables x. This is consistent with the intensional interpretation of $\Box A$, since we may not know an expression which denotes a given value and therefore cannot permit an arbitrary value as an expression.

The above excerpt refers to the modal type system introduced and used by Davies and Pfenning, which is a *dual-context λ-calculus*, with judgements of the form

$$\Delta\,;\,\Gamma \vdash M : A$$

where Δ and Γ are two ordinary but disjoint contexts. The variables that occur in Δ are to be thought of as *modal variables*, or variables that carry *intensions* or *codes*, whereas the variables in Γ are ordinary (intuitionistic) variables. In this system, the evaluator $\mathbf{E} : \Box A \to A$ exists as a variable rule, i.e. the ability to use a code variable as if it were a value:

$$\frac{}{\Delta, u{:}A, \Delta'\,;\,\Gamma \vdash u : A}$$

The canonical terms of modal type are of the form $\mathsf{box}\, M$, and they largely mimic the Gödel numbering $\ulcorner M \urcorner$. The shape of the introduction rule guarantees that all the variables that occur in the 'boxed term' M are code variables, and not value variables; we write $\cdot$ for the empty context:

$$\frac{\Delta\,;\,\cdot \vdash M : A}{\Delta\,;\,\Gamma \vdash \mathsf{box}\, M : \Box A}$$

And, as mentioned above, there is also a $\mathsf{let\ box}\ u \Leftarrow (-)\ \mathsf{in}\ (-)$ construct that allows for substitution of quoted terms for code variables:

$$\frac{\Delta\,;\Gamma \vdash M : A \quad \Delta, u{:}A\,;\Gamma \vdash N : C}{\Delta\,;\Gamma \vdash \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N : C}$$

along with the reduction

$$\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N \longrightarrow N[M/u]$$

The $\mathsf{let\ box}\ u \Leftarrow (-)\ \mathsf{in}\ (-)$ construct secretly requires $\Box(A \times B) \cong \Box A \times \Box B$, which is just enough to define the $\mathbf{app} : \Box(A \to B) \to \Box A \to \Box B$ constant.

Finally, the $\mathsf{4}$ axiom is inherent in the system:

$$\frac{\cdot\,;x : \Box A \vdash x : \Box A \qquad \dfrac{\dfrac{u{:}A\,;\cdot \vdash u : A}{u{:}A\,;\cdot \vdash \mathsf{box}\ u : \Box A}}{u{:}A\,;x : \Box A \vdash \mathsf{box\ box}\ u : \Box A}}{\cdot\,;x : \Box A \vdash \mathsf{let\ box}\ u \Leftarrow x\ \mathsf{in\ box\ box}\ u : \Box A}$$

The author has previously studied such dual-context systems in [Kavvos, 2017b].

1.4.2 A Puzzle: Intensional Recursion

We touched upon the subject of recursion in type theory in §1.2.4. To obtain recursion in simple types, we have to add a fixed point combinator $\mathbf{Y} : (A \to A) \to A$, and obtain PCF. In contrast, recursion is definable in the untyped setting. This is the conclusion of the *First Recursion Theorem* [Barendregt, 1984, §2.1, §6.1]:

Theorem 3 (First Recursion Theorem)**.** $\forall f \in \Lambda.\ \exists u \in \Lambda.\ u = fu$.

Proof. Let

$$\mathbf{Y} \stackrel{\text{def}}{=} \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

Then $\mathbf{Y}f =_\beta f(\mathbf{Y}f)$. □

The FRT corresponds to *extensional recursion*, which is what most programming languages support. When defining a recursive function definition, a programmer may make a finite number of calls to the definiendum itself, in the same vein as our description of the functional-extensional paradigm in §1.2.1. Operationally, this leads a function to examine its own values at a finite set of points at which it has—hopefully—already been defined.

However, as Abramsky [2014] notes, in the *intensional paradigm*, which we also described in §1.2.1, a stronger kind of recursion is attainable. Instead of merely examining the result of a finite number of recursive calls, the definiendum can recursively have access to a *full copy of its own source code*. This is embodied in the *Second Recursion Theorem (SRT)*, which was proved by Kleene [1938]. Here is a version of the SRT in the untyped λ-calculus [Barendregt, 1984, §6.5]:

Theorem 4 (Second Recursion Theorem)**.** $\forall f \in \Lambda.\ \exists u \in \Lambda.\ u = f\ulcorner u \urcorner$.

Proof. Given $f \in \Lambda$, set $u \stackrel{\text{def}}{=} W\ulcorner W\urcorner$, where

$$W \stackrel{\text{def}}{=} \lambda x.\ f(\mathbf{app}\ x\ (\mathbf{gnum}\ x))$$

where **app** and **gnum** are as above. Then

$$\begin{aligned} u &\equiv W\ulcorner W\urcorner \\ &=_\beta f(\mathbf{app}\ \ulcorner W\urcorner\ (\mathbf{gnum}\ \ulcorner W\urcorner)) \\ &=_\beta f(\mathbf{app}\ \ulcorner W\urcorner\ \ulcorner\ulcorner W\urcorner\urcorner) \\ &=_\beta f(\ulcorner W\ulcorner W\urcorner\urcorner) \\ &\equiv f\ \ulcorner u\urcorner \end{aligned}$$

□

It is not hard to see that, using Kleene's interpreter for the λ-calculus (Theorem 1), the SRT implies the FRT. It is not at all evident whether the converse holds. This is because the FRT is a theorem that concerns higher-order computation, whereas the SRT is very much grounded on first-order, diagonal constructions. The exact relationship between these two theorems is the subject of §2.

The point that we wish to make is that, in the presence of intensional operations, the SRT affords us with a much stronger kind of recursion. In fact, it allows for exactly the sort of *computational reflection* that we discussed in §1.2.2.

Perhaps the greatest surprise to be found in this thesis is that the SRT *admits a type*. Indeed, suppose that $u : A$. Then certainly $\ulcorner u\urcorner : \Box A$, and it is forced that

$$f : \Box A \to A$$

The Curry-Howard reading of the SRT is then the following: for every $f : \Box A \to A$, there exists a $u : A$ such that $u = f\ulcorner u\urcorner$. This corresponds to *Löb's rule* from *provability logic* [Boolos, 1994]:

$$\frac{\Box A \to A}{A}$$

Löb's rule is equivalent to adding the Gödel-Löb axiom,

$$\Box(\Box A \to A) \to \Box A$$

to a modal logic. One of the punchlines of this thesis will be that

The type of the Second Recursion Theorem is the Gödel-Löb axiom.

Thus, to obtain reflective features, all we have to do is add a version of Löb's rule to the Davies-Pfenning modal λ-calculus.

1.5 A Road Map

The rest of this thesis consists of a thorough discussion of the above observations, and the development of appropriate syntax and categorical semantics that capture the aforementioned intuitions.

In §2, we dive back to find the origins of Kleene's two Recursion Theorems. As it happens, these correspond to the two ways in which we can define a mathematical object by recursion: one can either use diagonalisation, or some kind of infinite (or transfinite) iteration. The origin of both of these methods is lost in the mists of time, but both find their first documented expressions in the work of Kleene [1952, 1938]. §2 states and proves these two theorems, and engages in a thorough discussion regarding their similarities and differences. Whereas the SRT works by diagonalising and is applicable to a first-order setting, the FRT requires a higher-order perspective. In some cases both theorems are applicable, but one is stronger than the other: the FRT always produces least fixed points, but this is not always the case with the SRT. Nevertheless, an old result by Hartley Rogers Jr. bridges this gap. We also find the opportunity to engage in some speculation regarding the possible applications of the reflective features provided by the SRT.

In §3, we revisit the system of Pfenning and Davies [2001] that we described in §1.4. After noting that it does not feature any actual intensional operations, we add some to the system. We also take the hint from §1.4.2, and add a form of Löb's rule as well. The result is a programming language that is (a) *intensional*, exactly in the sense described in §1.2.1, and (b) *reflective*, in the sense described in §1.2.2. A confluence proof ensures that the resulting system, which we call *Intensional PCF*, is consistent. It is evident that the central device by which everything comes together is the modal types, which separate the two worlds of intension and extension, by way of containing the former under the modality.

Following that, in §4 we begin to seek categorical semantics for intensionality. We argue that 1-category theory is ill-suited for modelling intensionality. We are thus led to consider the *P-categories* of Čubrić, Dybjer and Scott [Čubrić et al., 1998], which are categories only up to a *partial equivalence relation* (PER). In this setting, we introduce a new P-categorical construct, that of *exposures*. Exposures are very nearly functors, except that they do not preserve the PERs of the P-category, but *reflect* them instead. Inspired by the categorical semantics of S4 [Bierman and de Paiva, 2000], we begin to develop the theory of exposures.

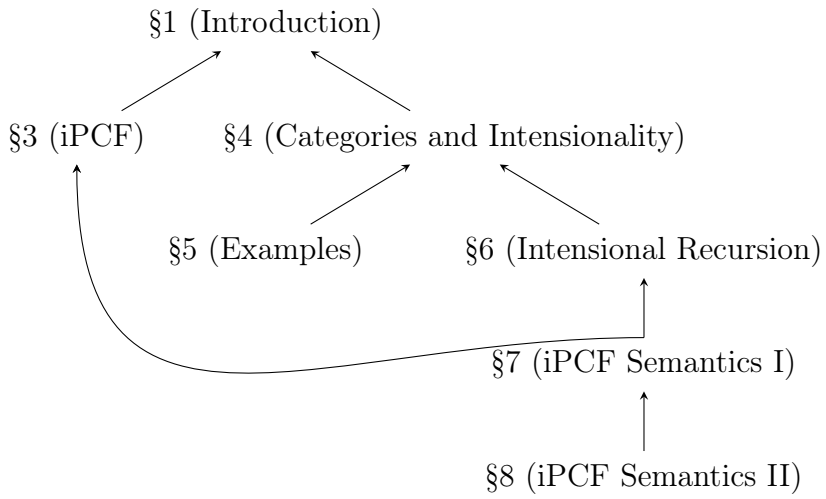
To substantiate the discussion, §5 builds on the previous chapter by presenting three concrete examples of exposures. The first example shows that, when built on an appropriate P-category, an exposure is really an abstraction of the notion of a well-behaved intensional device, such as a *Gödel numbering*. The second example is based on *realizability theory*; it is also the motivating example for exposures, and is later used to show that Kleene's SRT is a form of intensional recursion. The final example illustrates that intensionality and exposures may occur outside logic and computability, and is related to basic homological algebra.

Then, in §6, we put on our P-categorical spectacles and examine intensional recursion. We find that it has a simple formulation in terms of exposures. We then show that classic theorems of logic that involve intensional recursion, such as Gödel's *First Incompleteness Theorem*, Tarski's *Undefinability Theorem*, and *Rice's Theorem*, acquire concise, clear formulations in the unifying framework of exposures. Moreover, our theory ensures that each logical device or assumption involved in their proofs can be expressed in the same algebraic manner. The chapter concludes by using exposures to generalise a famous categorical fixed-point theorem of Lawvere [1969, 2006], which roughly corresponds to a restricted version of the FRT. The resulting Intensional Recursion Theorem is a categorical analogue of Kleene's SRT.

At last, in §7 and §8, we bring Intensional PCF (§3) and exposures (§4) together, by using the second to provide a semantics for the first. Intensional PCF is too expressive for this endeavour, so we discuss a restriction of it, which we call Intensional PCF v2.0. We find that a sound semantics for it consists of a *cartesian closed P-category* equipped with a *product-preserving, idempotent comonadic exposure*. We then discuss in which cases we may lift some of these new restrictions of Intensional PCF v2.0. We conclude by showing that $\mathfrak{Asm}(K_1)$ is a model of Intensional PCF v2.0, with the intensional fixed points being interpreted by Kleene's SRT.

Finally, in §9, we conclude our investigation by evaluating our contribution, and delineating a number of future directions towards which this thesis seems to point.

Figure 1.1: Chapter dependencies



The figure presents a rough outline of the way the chapters of this thesis depend on each other. §2 is not included in the diagram, as it functionally independent of the developments in the thesis. Nevertheless, we recommend that the reader consult it in order to understand the origins and importance of intensional recursion. The sequence §1, §4, §6 may be read independently, hence constituting the basics of our 'theory of intensionality.' Finally, the diagram does not capture two small further dependencies: that of §6.4 on the examples developed in §5, and that of the construction of IFPs in $\mathfrak{Asm}(K_1)$ in §8.6 on the definition of that P-category in §5.2.

Chapter 2

Kleene's Two Kinds of Recursion[1]

It is well known that there are two ways to define a function by recursion.

One way is through a *diagonal construction*. This method owes its popularity to Cantor, and forms the backbone of a large number of classic diagonalisation theorems. Diagonal constructions are a very concrete, syntactic, and computational method of obtaining fixed points, which we in turn use to obtain recursion.

Another way is through a *least fixed point* that is obtained as a result of some kind of infinite (or even transfinite) iteration. This kind of construction is more abstract and mathematical in style. It is a very common trope in the study of denotational semantics of programming languages, particularly those based on *domain theory* [Abramsky and Jung, 1994]. The origins of this lattice-theoretic argument are lost in the mists of time, but see Lassez et al. [1982] for a historical exposition.

Both of these ways were famously used by Stephen C. Kleene [Kleene, 1981] to define functions by recursion in computability theory. The least fixed point construction is the basis of Kleene's *First Recursion Theorem (FRT)* [Kleene, 1952], whereas the diagonal construction is found at the heart of his *Second Recursion Theorem (SRT)* [Kleene, 1938].

Nevertheless, it is not so well known that there is a slight mismatch between these two theorems. This is mainly due to the context in which they apply: the FRT is essentially a theorem about *computation at higher types*, whereas the SRT is a *first-order theorem* of a syntactic nature.

Modulo the above mismatch, it so happens that *the SRT is more general than the FRT*. Indeed, the SRT allows for a computationally 'stronger' kind of recursion—namely *intensional recursion*—whereas the FRT has a more *extensional* flavour. However, there is a close yet slightly mysterious relationship between these two theorems,

[1] A preprint is available as arXiv:1602.06220

the particulars of which we shall examine.

The SRT has numerous applications in computability, but it is deafeningly absent in computer science. Abramsky [2014] has suggested further investigation, likening the SRT to "the dog that didn't bark" in the Sherlock Holmes story, and has also discussed several related issues.

In the sequel we shall investigate both of these types of recursion, as well as their intricate relationship. In §2.1 we discuss the notion of intensionality and its relationship to computability; we state and prove the SRT, and discuss the extra generality afforded by what we call intensional recursion; and we sketch a number of speculative applications of intensional recursion. Subsequently, in §2.2, we move to the discussion of higher types: we look at two slightly different notions of computation at higher types, discuss their interaction, and prove the FRT for each one. Finally, in §2.3, we investigate when each of the recursion theorems applies, and when the resulting recursive constructions match each other.

2.1 Intensionality and Computability

In loose philosophical terms, to be *intensional* is to contain not only *reference*, but also *sense*. The distinction between these two notions is due to Frege, see e.g. Fitting [2015]. An intensional sign denotes an external *referent*, yet inherently connotes more information—its elusive *sense*. The classic example is that of the planet Venus, which may be referred to as either the morning star, or the evening star.

Most mainstream mathematics is rather *extensional*: we normally reason about underlying, 'ideal' mathematical objects, and not their concrete descriptions; the latter are, in a way, only there for our referential convenience. In most presentations of set theory, for example, the *axiom of extensionality* equates any two sets whose members are the same. Thus, in the mathematical sense, *to be intensional is to be finer than some presupposed 'extensional equality.'*

It is not difficult to argue that this setting is most inadequate for Computer Science. On a very rough level, extensions correspond to *what may be computed*, whereas intensions correspond to the *programs and processes that carry out the computation*, see e.g. Moschovakis [1993]. Once more, there is a distinction to be made: programs may be understood by the ideal objects that they refer to (e.g. functions), or their internal characteristics: length, structure, and—ultimately—the *algorithm* they express.

The former aspect—viz. the study of ideal objects behind programs—is the domain of *Computability Theory* (previously known as *Recursion Theory*), where the object of study is 'effectively computable' functions over the natural numbers. Computability Theory began with the 'confluence of ideas' [Gandy, 1988] of multiple researchers in the late 1930s in their attempt to characterise 'automatic' or 'mechanical' calculability. Remarkably, all roads led to Rome: different notions were shown to coincide, leading to the identification of the class of *partial recursive functions*. Subsequent to this fortuitous development, things took a decisive turn, as further developments mostly concerned the study of the *incomputable*.[2]

So much for the extensional side. What about intensions? Here, we encounter a diverse ecosystem. On one side, fixing the *Turing Machine* as one's model of computation leads to *Complexity Theory*, which attempts to classify algorithmic problems with a view to identifying the exact resources that one needs to solve a a problem. This aspect is largely reliant on more *combinatorial reasoning*. Alternatively, adopting the λ-calculus as a point of reference leads to the study of *programming languages*, which includes—amongst other things—type systems, semantics, program analysis, and program logics. Here, the emphasis is on logical aspects. Finally, the subject of concurrent and interactive computation unfolds as a bewildering, obscure, and diverse landscape: see e.g. Abramsky [2006].

2.1.1 The Space of All Programming Languages

It is, however, a curious state of affairs that standard computability theory—as presented, for example, in the classic textbooks of Rogers [1987], Cutland [1980] and Odifreddi [1992]—begins with a small set of abstract results that concern Gödel numberings of partial recursive functions. Even though they are the central pillar of an extensional theory, these results have a decidedly intensional flavour.

These results begin by putting some very mild conditions on Gödel numberings. Indeed, if one thinks of a Gödel numbering as a 'programming language,' then these conditions comprise the absolute minimum that intuitively needs to hold if that programming language is 'reasonable.' A clear presentation of this part of the theory can be found in the classic textbook of Odifreddi [1992, §II.5]. A more modern account that is informed by programming language theory is that of Neil Jones [1997].

[2]Harvey Friedman [1998] once made the following tongue-in-cheek recommendation to the Foundations of Mathematics (FOM) mailing list: "Why not rename recursion theory as: noncomputability theory? Maybe that would make everybody happy."

The story begins to unfold as soon as we encode *programs-as-data*, by assigning partial recursive functions to natural numbers. Following tradition, we write ϕ for an arbitrary numbering, and $\phi_p : \mathbb{N} \rightharpoonup \mathbb{N}$ for the partial recursive function indexed by $p \in \mathbb{N}$ under the numbering ϕ.

From a programming perspective, we may consider p to be a 'program,' and $\phi : \mathbb{N} \to (\mathbb{N} \rightharpoonup \mathbb{N})$ a semantic function that maps programs to the functions they compute. In practice, $p \in \mathbb{N}$ is usually a Gödel number that encodes the syntax of a Turing Machine, or the instructions for a register machine, or even a λ-term.

Let us write $e_1 \simeq e_2$ for *Kleene equality*, viz. to mean that expressions e_1 and e_2 are both undefined, or both defined and equal in value. Of the numbering ϕ we shall require the following conditions:

Turing-Completeness That for each partial recursive function f there exists a p such that $f = \phi_p$.

Universal Function That there is a program U such that $\phi_U(x, y) \simeq \phi_x(y)$ for all $x, y \in \mathbb{N}$.

S-m-n That there is a *total* recursive function S such that $\phi_{S(p,x)}(y) \simeq \phi_p(x, y)$ for all $x, y \in \mathbb{N}$.

That the first and second conditions are achievable was popularised by Turing [1937], and the third is a result of Kleene [1938]. The first condition corresponds, by the *Church-Turing thesis*, to the fact that our programming language is as expressive as possible (in extensional terms). The second corresponds to the ability to write a *self-interpreter*, under suitable coding. And the third allows us to computably 'fix' an argument into the source code of a two-argument program, i.e. that substitution is computable (c.f. one-step β-reduction in the λ-calculus).

In logical terms, we may regard these as sanity conditions for Gödel numberings of the partial recursive functions. The 'sane' numberings that satisfy them are variously known as *acceptable numberings* [Rogers, 1987, Ex. 2.10], *acceptable programming systems* [Machtey and Young, 1978, §3.1.1], or *systems of indices* [Odifreddi, 1992, §II.5.1].

It was first shown by Rogers [1958] that acceptable numberings have very pleasant properties.

Definition 1. For numberings ϕ and ψ, define

$$\phi \leq_R \psi \quad \text{iff} \quad \exists t : \mathbb{N} \rightharpoonup \mathbb{N} \text{ total recursive. } \forall p \in \mathbb{N}.\ \phi_p = \psi_{t(p)}$$

We then say that ϕ Rogers-reduces to ψ, and $\leq_R$ is a preorder.

Hence, thinking of ϕ and ψ as different programming languages, $\phi \leq_R \psi$ just if every ϕ-program may be effectively translated—or *compiled*—to a ψ program. Then $\equiv_R \stackrel{\text{def}}{=} \leq_R \cap \geq_R$ is an equivalence relation. Quotienting by it yields the *Rogers semilattice* under the extension of $\leq_R$ to the equivalence classes (see *op. cit*). More specifically,

Theorem 5. *The following are equivalent:*

1. *ψ is an acceptable numbering, as above.*
2. *ψ is a member of the unique top element of the Rogers semilattice.*
3. *ψ is an enumeration for which there is a universal function, and a total recursive $c : \mathbb{N} \rightharpoonup \mathbb{N}$ such that*
$$\psi_{c(i,j)} = \psi_i \circ \psi_j$$

The first two equivalences are due to Rogers, see *op. cit*, and Odifreddi [1992, §II.5.3], and the third is due to Machtey et al. [1978, Theorem 3.2]. Note that one should exercise great caution with these equivalences, for their proofs liberally invoke pairing tricks, loops, iterations, and other programming constructs. Finally, the equivalence between (1) and (2) above was strengthened by Rogers [1958] to

Corollary 1 (Rogers' Isomorphism Theorem)**.** *Any two acceptable enumerations are recursively isomorphic.*

The possible numberings of the partial recursive functions, whether acceptable or pathological, as well as the various forms of SRTs that may or may not hold of them, have been investigated by the school of John Case and his students: David Riccardi [1980, 1981], James Royer [1987] and, more recently, Samuel Moelius III [Case and Moelius, 2007, 2009a,b, Moelius, 2009]. For example, this community has shown that there are numberings where certain known theorems, just as the s-m-n theorem or Kleene's second recursion theorem, are not 'effective,' or simply do not hold. Earlier work on this front seems to have concentrated on enumerations of subrecursive classes of functions, see e.g. Royer and Case [1994], whereas the later work of Case and his students concentrated on the study of what they called *control structures*, i.e. constructs which provide "a means of forming a composite program from given constituent programs and/or data."

2.1.2 The Second Recursion Theorem

The central intensional result of elementary computability theory is Kleene's *Second Recursion Theorem (SRT)*, also first shown in [Kleene, 1938].

Theorem 6 (Kleene)**.** *For any partial recursive $f : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$, there exists $e \in \mathbb{N}$ such that*

$$\forall y \in \mathbb{N}.\ \phi_e(y) \simeq f(e, y)$$

Proof. Consider the function defined by

$$\delta_f : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$$
$$\delta_f(y, x) \simeq f(S(y, y), x)$$

Since f is partial recursive, simple arguments concerning the computability of composition and substitution yield that δ_f is partial recursive. Hence $\delta_f = \phi_p$ for some $p \in \mathbb{N}$. Consider $e \stackrel{\text{def}}{=} S(p, p)$; then

$$\phi_e(y) \simeq \phi_{S(p,p)}(y) \simeq \phi_p(p, y) \simeq \delta_f(p, y) \simeq f(S(p, p), y) \simeq f(e, y)$$

The second Kleene equality follows by the s-m-n theorem, and the others are simply by definition or construction. □

In the above theorem, consider $f(x, y)$ as a function that treats its first argument as code, and its second argument as data. The equation $\phi_e(y) \simeq f(e, y)$ implies that e is a program which, when run on some data, will behave like f with e being its first argument. In slogan form,

> We can always write a program in terms of its own source.

Indeed, the trick has become standard: $f(e, y)$ is a 'blueprint' that specifies what to do with its own code e, and we take its fixed point (c.f. the functionals in the untyped λ-calculus to which we apply the $\mathbf{Y}$ combinator). Moreover, in much the same way that the $\mathbf{Y}$ combinator is itself a term of the untyped λ-calculus, it so happens that the construction used in the proof of the SRT is itself computable:

Theorem 7 (Constructive Kleene SRT)**.** *There is a total recursive $h : \mathbb{N} \rightharpoonup \mathbb{N}$ such that, for every $p \in \mathbb{N}$,*

$$\phi_{h(p)}(y) \simeq \phi_p(h(p), y)$$

That is: in the original statement, we may calculate a code for δ_f from a code for f, and hence obtain e in an effective manner.

The Kleene SRT lies at the heart many proofs in computability, especially those involving diagonalisation, or results pertaining to fixed points, self-reference and the like. A smattering of more theoretical applications of this 'amazing' theorem has been compiled by Moschovakis [2010].

The SRT is perhaps more familiar in the form popularised by Hartley Rogers Jr. in his book [Rogers, 1987]. We state a slightly generalised version, and prove it from Kleene's version. We write $e \downarrow$ to mean that the expression e has a defined value.

Theorem 8 (Rogers SRT)**.** *For partial recursive $f : \mathbb{N} \rightharpoonup \mathbb{N}$, there is a $e \in \mathbb{N}$ such that, if $f(e) \downarrow$, then*

$$\phi_e = \phi_{f(e)}$$

Proof. [Jones, 1997, Lemma 14.3.7] Define

$$d_f(x, y) \simeq \phi_U(f(x), y)$$

Again, by standard arguments, d_f is partial recursive. By Kleene's SRT, there is a $e \in \mathbb{N}$ such that, for all $y \in \mathbb{N}$,

$$\phi_e(y) \simeq d_f(e, y) \simeq \phi_U(f(e), y) \simeq \phi_{f(e)}(y)$$

which is to say $\phi_e = \phi_{f(e)}$. □

This result is equivalent to the previous formulation [Rogers, 1987, Ex. 11-4]. We may summarise it in the following slogan:

> Every computable syntactic program transformation
> has a semantic fixed point.

Moreover, this version of the SRT comes in a 'constructive' variant as well—see Rogers [1987, §11.2-II] or Cutland [1980, §11-3.1]:

Theorem 9 (Constructive Rogers SRT)**.** *There is a total recursive $n : \mathbb{N} \rightharpoonup \mathbb{N}$ such that, for any $z \in \mathbb{N}$ such that ϕ_z is total,*

$$\phi_{n(z)} = \phi_{\phi_z(n(z))}$$

All of the above variants of the SRT are *equivalent* under the assumption that ϕ is an acceptable enumeration. Nevertheless, if we relax the assumption of acceptability, there are ways to compare and contrast them. Riccardi [1980] showed that there are enumerations of the partial recursive functions for which there exist Rogers-type fixed points, but not Kleene-type fixed points. In a rather technical section of his thesis, Moelius [2009, §3] painstakingly compares the various entailments between different formulations of the recursion theorem, and concludes that Kleene's is the one that is more natural and general.

No matter how enticing it may be, we shall not dwell on this particular line of discussion. In fact, we shall avoid it as much as possible, because it does not fit our view of programming. Any numbering that is *not* acceptable is somehow pathological: by contraposition, it follows that either substitution is not computable (and the s-m-n theorem does not hold), or that there is no self-interpreter (universal function)—which, by Turing-completeness, means that the interpreter as a function is not computable. Things are even more subtle if the language is *not* Turing-complete: we then have a *subrecursive* indexing, as in [Royer and Case, 1994], and this is not a path we would like to tread presently.

2.1.3 Intensional Recursion

From the point of view of programming languages, the very general and still rather strange application of the SRT is the ability to define functions by *intensional recursion*. This means that a function can not only call itself on a finite set of points during its execution—which is the well-known extensional viewpoint—but it may also examine its own *intension*, insofar as the source code for a program is a finite and complete representation of it.

Indeed, the SRT is the only basic tool available in standard (non-higher-order) accounts of computability that enables one to make any 'unrestricted' recursive definition whatsoever. For example, if we define

$$f(x, y) \simeq \begin{cases} 1 & \text{if } y = 0 \\ y \cdot \phi_U(x, y - 1) & \text{otherwise} \end{cases}$$

and apply Kleene's SRT, we obtain a code $e \in \mathbb{N}$ such that ϕ_e is the factorial function.

However, the above use is slightly misleading, in that it is *extensional*: x is only used as an argument to the universal function ϕ_U. Hence, the resulting behaviour does not depend on the code x itself, but only on the values of the function ϕ_x for which it stands. The following definition captures that phenomenon.

Definition 2. A total recursive $f : \mathbb{N} \rightharpoonup \mathbb{N}$ is *extensional* just if

$$\phi_a = \phi_b \quad \Longrightarrow \quad \phi_{f(a)} = \phi_{f(b)}$$

for any $a, b \in \mathbb{N}$.

That is, $f : \mathbb{N} \rightharpoonup \mathbb{N}$ is extensional just if the program transformation it effects depends solely on the extension of the program being transformed. By a classic result of Myhill and Shepherdson [1955], such transformations correspond to a certain class of functionals which we discuss in §2.2.1.

However, even if $f : \mathbb{N} \rightharpoonup \mathbb{N}$ is extensional, the paradigm of intensional recursion strictly increases our expressive power. For example, suppose that we use the SRT to produce a program e satisfying a recursive definition of the form

$$\phi_e(x, y) \simeq \ \ldots \phi_U(e, g(x), y) \ldots$$

for some $g : \mathbb{N} \rightharpoonup \mathbb{N}$. We could then use the s-m-n function to replace this recursive call by a specialisation of e to $g(x)$, thereby obtaining another program e' of the form

$$\phi_{e'}(x, y) \simeq \ \ldots \phi_{S(e', g(x))}(y) \ldots$$

This is an equivalent program, in the sense that $\phi_e = \phi_{e'}$. But if the s-m-n function $S(e, x)$ performs some *optimisation* based on the argument x—which it may sometimes do—then e' may provide a more efficient definition, in that it the code for the recursive call may be optimised for this particular recursive call. This line of thought is the driving force of the *partial evaluation* community: there, the s-m-n function is called a *specialiser* or a *partial evaluator*, and it is designed so that it optimises the programs it is called to specialise; see §2.1.4 for more details.

But the SRT also allows for recursive definitions which are *not functional*, in that the 'blueprint' of which we take a fixed point may not be extensional. For example, one may be as daft as to define

$$f(x) \simeq \begin{cases} x + 1 & \text{if } x \text{ is even} \\ x - 1 & \text{if } x \text{ is odd} \end{cases}$$

This $f : \mathbb{N} \rightharpoonup \mathbb{N}$ is total recursive, but decidedly not extensional. However, we may still use Rogers' theorem to obtain a fixed point $e \in \mathbb{N}$ such that $\phi_e = \phi_{f(e)}$, and the resulting behaviour will depend on the parity of e (!). To this day, it is unclear what the use of this is, except of course its underlying rôle in powerful diagonal arguments—see e.g. Cutland [1980, §11.2]—as well as many kinds of *reflection*.

2.1.4 Applications of Intensional Recursion

Abramsky [2014] observed that the SRT, as well as other simple results on program codes, are strangely absent from Computer Science. He comments:

> "This reflects the fact which we have already alluded to, that while Computer Science embraces wider notions of processes than computability theory, it has tended to refrain from studying intensional computation, despite its apparent expressive potential. This reluctance is probably linked to the fact that it has proved difficult enough to achieve software reliability even while remaining within the confines of the extensional paradigm. Nevertheless, it seems reasonable to suppose that understanding and harnessing intensional methods offers a challenge for computer science which it must eventually address."

In many ways, we empathise with this programme. Consequently, we catalogue some applications of such intensional 'results about program codes,' both within and on the fringes of Computer Science, and then engage in some speculation regarding various future directions.

Partial Evaluation

Kleene's s-m-n theorem allows one to 'specialise,' or 'partially evaluate' a certain program by fixing some of its arguments. It may appear simple and innocuous, but this is deceptive: the s-m-n theorem is an essential result in computability.

The power afforded by the s-m-n theorem bestowed considerable success upon the *partial evaluation* community, which began with the work of Futamura [1999] and Ershov [1977, 1982] in the 1970s. Futamura observed that the ability to write an interpreter for a language (i.e. a universal function), as well as the ability to 'specialise' an argument of a program (s-m-n function) followed by source-level optimisation yields an easy approach to generate a compiler, thus leading to the three *Futamura projections*. Writing $S = \phi_s$, where S is the s-m-n function, and U for the program corresponding to the universal function, we have:

$$\begin{aligned} \mathsf{target\ code} &\stackrel{\text{def}}{=} \phi_s(U, \mathsf{source\ code}) \\ \mathsf{compiler} &\stackrel{\text{def}}{=} \phi_s(s, U) \\ \mathsf{compiler\ generator} &\stackrel{\text{def}}{=} \phi_s(s, s) \end{aligned}$$

One can then verify that these equations do yield the desired behaviour, as shown in an elementary fashion by Jones [1997]. This led to a successful programme of automatic generation of compilers, first realised in Copenhagen. The results are documented in Jones [1996], and the book of Jones et al. [1993].

Contrasting the simplicity of the s-m-n theorem with its success in the partial evaluation community also led Jones to ponder whether the SRT, which is a much more powerful result, could have interesting practical applications. Quoting from [Jones, 1992]:

> "While this theorem has many applications in mathematics, it is not yet clear whether it will be as constructively useful in computer science as the s-m-n theorem has turned out to be."

Taking this as a point of departure, he has posed two significant questions:

- What is the exact relationship between the First Recursion Theorem (FRT) and the Second Recursion Theorem (SRT)?
- If one implements the SRT, how can the accumulating layers of self-interpretation be avoided?

Looking back at the literature on computability, we find that the first question has been answered in a mostly satisfactory way. Regarding the second question, some recent progress is documented in Kiselyov [2015].

A series of experiments with the SRT and further discussion are documented in Hansen et al. [1989], and Jones [1992, 2013].

Abstract Computer Virology

Computer viruses rely heavily on the ability to propagate their own code, which is a kind of reflection. This was noticed by Cohen [1989], who was the first to introduce the term *computer virus* alongside an early Turing Machine model of viruses. A few years later, his supervisor—Leonard Adleman [1990]—concocted his own model of viruses that is based on elementary computability theory. In that model, viruses are program transformations that 'infect' ordinary programs. This is also where the connection with the SRT was made explicit, as Adleman invokes it to construct a program that—under his own definition—is classified as a virus. He also proved a result on his model that makes crucial use of the SRT for a diagonal construction.

These were the two cornerstones that laid the foundation of *abstract computer virology*. In more recent years there have been further developments, owing to the work of Bonfante et al. [2005, 2006, 2007], who discuss and classify different types of viruses that correspond to multiple variants of the SRT. See also Marion [2012].

Computational Reflection & Reflective Towers

The concept of *computational reflection* was introduced in the context of programming languages by Brian Cantwell Smith [1982, 1984] in his voluminous thesis. The underlying intuition is that a program may be considered to be running on an interpreter, which interpreter itself is running on another copy of this interpreter, and so on A special construct that makes use of this structure is available in the programming language: it allows one to inject code in the interpreter that lies one level below. Hence, a program has access not only to its code, but the entire state of the interpreter that runs it, and even the interpreter that runs the interpreter. This is embodied by the language 3-LISP, introduced in *op. cit.*

The resulting structure of *reflective towers* has captured the imagination of many, but—while couched in colourful imagery—its construction is logically and computationally mysterious. A series of publications [Friedman and Wand, 1984, Wand and Friedman, 1988, Danvy and Malmkjaer, 1988] have made partial attempts to explain this 'tower' in more concrete terms.

Nevertheless, the concept of *computational reflection* seems to be a general recurring theme with broader scope, but is not well-understood. For the state of affairs up to the mid-1990s, see the short comparative survey Demers and Malenfant [1995]. Demystifying reflection is a pressing concern, as many modern programming languages have reflective or 'introspective' facilities, which are infamous for pernicious bugs, and generally wreaking havoc.

Some ideas regarding reflection seem to be experiencing a resurgence of interest, mainly because of the appearance of a new candidate foundation for reflection, namely Barry Jay's *factorisation calculus*: see Jay and Given-Wilson [2011], Jay and Palsberg [2011], and Jay [2016].

As the SRT is a result with fundamental connections to reflection, we believe that a better understanding of intensional recursion must be instrumental in laying a logical foundation for reflective constructs.

Economics

This section concerns a speculative application of the SRT to economic modelling. Historically, there has been a lot of discussion—especially in the years following the Great Recession—regarding the foundational principles of economics, and whether these are an adequate substrate for the science. Some of these critical approaches touch on the *self-referential* aspects that are ignored by (neoclassical) economic theory. Winrich [1984] offers an early example of these criticisms; we give him the floor, for the argument is compelling:

> In order for preferences to be complete the choice set must include preferences themselves! But, as soon as you allow preferences within the choice set you have preferences "talking" to preferences. In a world of preference self-reference we can, if you will, produce a neoclassical liar. As an example let our liar be a smoker. In such a situation it is not uncommon to hear a smoker say, "I dislike my desire to smoke." What are we to make of such a statement? In the static framework of neoclassical choice theory this is a contradiction. But at the same time, it cannot be prevented. Not only is the "act of smoking" an element in the choice set, but the "desire to smoke" is itself an element in the choice set and also subject to discretion.
>
> Continuously expanding the choice set by including not only commodities but also social conditions in an attempt to explain invidious distinctions, other individuals in an attempt to explain altruism, and so forth, neoclassicism has been able to "absorb" heterodox attacks within the atomic individualistic perspective. However, the inclusion of choice itself is devastating to its own premise of consistency.
>
> Let me make the point clear. Preference functions, preference orderings, or preference rankings do not exist. [...]

Indeed, changes to an economic process originating from within an economic process have been rather difficult to model in a reductionist fashion. Some early work that attempted to model the 'change of institutions' in game theory was carried out by Vassilakis [1989, 1992]. Moreover, there have even been approaches—even predating *algorithmic game theory*—that approach economics with computational feasibility in mind, see e.g. the work of Velupillai [2000] on *computable economics*. See also Blumensath and Winschel [2013] for a recent attempt involving coalgebra.

2.2 Extensional Recursion and the FRT

2.2.1 Effective Operations

Suppose that we have some $f : \mathbb{N} \rightharpoonup \mathbb{N}$ that is total and extensional. It is not hard to see that—by the definition of extensionality—f uniquely induces a very specific type of functional. Let us write $\mathcal{PR}$ for the set of partial recursive functions.

Definition 3. A functional $F : \mathcal{PR} \to \mathcal{PR}$ is an *effective operation*, if there exists a total, extensional $f : \mathbb{N} \rightharpoonup \mathbb{N}$ such that

$$F(\phi_x) = \phi_{f(x)}$$

This is well-defined precisely because f is extensional. We are entering the realm of *higher types*, by computing functions from functions. Nevertheless, we are doing so in a computable and finitary sense: functions may be infinite objects, but this computation occurs, or *is tracked*, on the level of program codes.

The Myhill-Shepherdson Theorem

Functionals such as the one defined above are perhaps one of the most straightforward ways to define computability at higher order, namely as effective code transformations. Surprisingly, Myhill and Shepherdson [1955] showed that the same functionals can be defined in a much more abstract manner that dispenses with code transformations entirely. In fact, anyone familiar with the domain-theoretic semantics of the λ-calculus will recognise it immediately.

We first need to discuss the simple order-theoretic structure of the set $\mathcal{P}$ of unary partial functions: its elements may indeed be ordered by *subset inclusion*:

$$\psi \subseteq \chi \quad \text{iff} \quad \forall x, y \in \mathbb{N}.\ \psi(x) \simeq y \Longrightarrow \chi(x) \simeq y$$

This makes $\mathcal{P}$ into a ω-complete partial order (ω-cpo), in that least upper bounds of increasing chains always exist, and they are unions. We can now make the following definition.

Definition 4. A functional $F : \mathcal{P} \to \mathcal{P}$ is *effectively continuous* just if it satisfies the following properties:

Monotonicitity $\psi \subseteq \chi \implies F(\psi) \subseteq F(\chi)$

Continuity For any increasing sequence of partial functions,

$$f_0 \subseteq f_1 \subseteq f_2 \cdots$$

we have

$$F\left(\bigsqcup_i f_i\right) = \bigsqcup_i F(f_i)$$

Effectivity on Finite Elements Given an encoding $\hat{\cdot}$ of the graph of every finite function $\theta : \mathbb{N} \rightharpoonup \mathbb{N}$ as a number $\hat{\theta} \in \mathbb{N}$, there is a partial recursive $g_F : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$ such that, for every finite $\theta : \mathbb{N} \rightharpoonup \mathbb{N}$, and for all $x \in \mathbb{N}$,

$$F(\theta)(x) \simeq g_F(\hat{\theta}, x)$$

These are called *recursive functionals* by Cutland [1980] and Rogers [1987]: we beseech the reader to exercise caution, as terminology varies *wildly*.

Let us restate continuity, by using the following equivalent formulation, for which see Odifreddi [1992, §II.2.23]:

Lemma 1 (Compactness)**.** *$F : \mathcal{P} \to \mathcal{P}$ is continuous if and only if*

$$F(f)(x) \simeq y \quad \Longleftrightarrow \quad \exists \text{ finite } \theta \subseteq f.\ F(\theta)(x) \simeq y$$

for all $x, y \in \mathbb{N}$.

Putting these together, we see that effectively continuous functionals are indeed very strongly *computational* and *effective*: the value of $F(f)(x)$ only depends on a finite part of the graph of f. In fact, we can show that the behaviour of F on finite elements completely determines its behaviour on f:

Lemma 2 (Algebraicity)**.** *Let $F : \mathcal{P} \to \mathcal{P}$ be continuous. Then*

$$F(f) = \bigsqcup \{\, F(\theta) \mid \theta \text{ finite } \wedge \theta \subseteq f \,\}$$

So, how do these functionals, which are computable in finite approximations, relate to the aforementioned effective operations, which are based on computations on indices? The answer is astonishingly simple:

Theorem 10 (Myhill-Shepherdson)**.** *An effective operation $F_{\mathit{eff}} : \mathcal{PR} \to \mathcal{PR}$ can be uniquely extended to an effectively continuous functional $F : \mathcal{P} \to \mathcal{P}$ (with $F_{\mathit{eff}} \subseteq F$).*

Conversely, any effectively continuous functional, when restricted to the partial recursive functions $\mathcal{PR}$, is an effective operation.

See Cutland [1980, 10-§2], Rogers [1987, §15.3, XXIX], or Odifreddi [1992, §II.4.2] for proofs.

A Turing Machine characterisation

Our discussion of functionals began with extensional operations on codes, as the most natural definition of higher-order computation.

There is, however, an alternative, which some would argue is even more simple: one may envisage the implementation of any functional $F : \mathcal{P} \to \mathcal{P}$ as a Turing Machine which has access to an *oracle* for the argument of the functional. During a computation, the machine may write a number x on a separate tape, and then enter a special state, in order to query the oracle. The oracle then replaces x with $f(x)$ if f is defined at x. If it is not, the oracle does not respond, forcing the machine to eternally wait for an answer, and the computation diverges.

Computation with oracles was first considered by Turing [1939], leading to the intricate theory of *Turing reducibility* and *relative computability*. However, Turing-type oracles are *fixed* in relative computability, whilst we consider them as *arguments to a computation*. This shift in perspective, as well as the first concrete results involving higher types, are due to Kleene [1952].

In this context, subtle issues arise with *non-determinism*. It is well-known that non-deterministic Turing Machines are equivalent to deterministic Turing machines at the first order, but at higher order this is no longer true. In fact, the following theorem was first shown—to the best of our knowledge—by Moschovakis [2010, §3], even though it was simply labelled as the Myhill-Shepherdson theorem:

Theorem 11 (Moschovakis)**.** *A functional $F : \mathcal{PR} \to \mathcal{PR}$ is an effective operation if and only if $F(f)$ is computable by a non-deterministic Turing machine with an oracle for f.*

Great care has to be taken in combining non-determinism with oracles: the machine should be designed so as to avoid the presence of two halting branches in the same computation tree with different candidate outputs. Similar restrictions occur in defining what it means to non-deterministically compute a polynomial time function—see e.g. Lewis and Papadimitriou [1997, §4.5.2]. However, in this case, non-halting branches do not harm anyone; in fact, they are in some sense necessary for the expressive power afforded by this model of computation.

The First Recursion Theorem

We have seen that effectively continuous functionals exhibit an impressive amount of inherent order-theoretic structure. By the Myhill-Shepherdson theorem, this struc-

ture emerges automatically once a functional can be 'realised' on codes by an extensional operation on codes.

This order-theoretic structure is the basis of the proof of the First Recursion Theorem (FRT). This fact was discovered by Dana Scott, who was the first one to notice that the core argument in the proof of the FRT applies to all so-called *simple types*.[3] This discovery of Scott led to the development of *domain theory*, and the study of PCF [Plotkin, 1977], both of which were the firstfruits of the field of programming language semantics. Indeed, Scott acknowledged his debts; quoting from [Scott, 1975]:

> "[...] It is rather strange that the present model was not discovered earlier, for quite sufficient hints are to be found in the early paper of Myhill and Shepherdson and in Rogers' book (especially §§9.7-9.8). These two sources introduce effective enumeration operators and indicate that there is a certain amount of algebra about that gives these operators a pleasant theory, but no one seemed ever to take the trouble to find out what it was."

The central result underlying the FRT is therefore this:

Theorem 12 (The Fixpoint Theorem)**.** *Let $(D, \sqsubseteq)$ be a ω-cpo with a least element $\bot \in D$, and let $F : D \to D$ be a continuous function. Then F has a* least fixed point*, defined explicitly by*

$$\mathbf{lfp}(F) = \bigsqcup_i F^i(\bot)$$

This is largely considered a folk theorem: its origins are difficult to trace, and many variants of it have been proved and used widely in Logic and Computer Science—see Lassez et al. [1982] for a historical view. In *op. cit* the authors note that Kleene knew of it at least as early as 1938; see also [Kleene, 1981].

In fact, it is high time that we show to the reader how it constitutes the first half of Kleene's FRT. But first, a caveat: the version of the FRT that we will presently prove pertains to effective operations (à la Myhill and Shepherdson), and a proof similar to ours may be found in the book by Cutland [1980, §10-3]. The original statement, found in Kleene's book [Kleene, 1952, §66], concerns partial recursive functionals, which we discuss in §2.2.2; see also Odifreddi [1992, §II.3.15].

[3] The simple types of PCF, as defined by Scott [1993], are generated by $\sigma ::= \mathbb{B} \mid \mathbb{N} \mid \sigma \to \sigma$. $\mathbb{B}$ is supposed to connote the *booleans*, and $\mathbb{N}$ the type of natural numbers.

Theorem 13 (First Recursion Theorem)**.** *Every effectively continuous functional $F : \mathcal{P} \to \mathcal{P}$ has a least fixed point $\mathbf{lfp}(F) : \mathbb{N} \rightharpoonup \mathbb{N}$, which is partial recursive.*

Furthermore, let $\phi_q : \mathbb{N} \rightharpoonup \mathbb{N}$ be an extensional function that realises F on the partial recursive functions, i.e. $F(\phi_x) = \phi_{\phi_q(x)}$. Then, a $p \in \mathbb{N}$ such that $\phi_p = \mathbf{lfp}(F)$ may be computed effectively from any such $q \in \mathbb{N}$.

Proof. The existence of the least fixed point follows from the fact $(\mathcal{P}, \subseteq)$ is a ω-cpo, and the Fixpoint Theorem.

By the Myhill–Shepherdson theorem, the functional F corresponds to some extensional $\phi_q : \mathbb{N} \rightharpoonup \mathbb{N}$. The proof of the Fixpoint Theorem constructs a chain,

$$f_0 = \emptyset \subseteq f_1 \subseteq f_2 \dots$$

where $f_{i+1} = F(f_i)$, and the least fixed point is $\mathbf{lfp}(F) = \bigsqcup_i f_i$. The f_i may be construed as increasingly defined 'approximations' to f. The key to this proof lies in using the extensional ϕ_q to obtain indices that *track* each element of this chain:

$$\begin{aligned} p_0 &= \text{(some index for the nowhere defined function)} \\ p_1 &= \phi_q(p_0) \\ &\vdots \end{aligned}$$

so that $p_{i+1} = \phi_q(p_i)$ and hence, by induction, $f_i = \phi_{p_i}$ for all $i \in \mathbb{N}$. Then, by Church's Thesis, we define $p \in \mathbb{N}$ by writing a program that performs the following steps: on input n,

1. Set $i := 0$.
2. Begin a *simulation* of the program p_0 running on n.
3. Loop:
   (a) For each $j \leq i$, simulate one step of p_j on n.
   (b) If any of these simulations have halted and produced a value m, halt and output m.
   (c) Otherwise, compute $p_{i+1} = \phi_q(p_i)$, and begin a simulation of p_{i+1} on input n. Set $i := i + 1$.

Since the f_i's are a chain, this program cannot accidentally produce two contradictory values. But since $\mathsf{lfp}(F) = \bigsqcup_i f_i$, if $f(n) \simeq m$, then $f_i(n) \simeq m$ for some i, so that $f = \phi_p$. □

In the books of both Cutland [1980] and Odifreddi [1992], the above proof is obtained after the recursively enumerable sets are characterised as the Σ_1^0 sets in the arithmetical hierarchy. We prefer the more primitive, 'algorithmic version' above, because we can isolate the expressive power needed.

So, what do expressive power do we use? The program in the proof curiously calls for *countable dovetailing*, i.e. the simulation of a slowly expanding yet potentially infinite set of computations, of which we perform a few steps at each stage. This requires access to the code $q \in \mathbb{N}$ of the relevant extensional function, so that we can run it to obtain the rest of the codes p_i. Furthermore, we require a handle on a number of simulations we spawn, so that we can pause them, schedule some steps of each, and possibly even discard some.

It is worth remarking once more that the output of this procedure is obviously deterministic, but there is inherent non-determinism and parallelism in the method we use to compute it. This is very much in line with Moschovakis' version of the Myhill–Shepherdson theorem (Theorem 11).

2.2.2 Partial Recursive Functionals and Pure Oracles

We have only discussed effective operations up to this point, and shown that they correspond to effectively continuous functionals.

In our discussion leading to Theorem 11 we also mentioned a slightly different paradigm, that of *oracle computation*. Theorem 11, however, guaranteed that non-deterministic oracle computation coincided with effective continuity. If, in contrast, we adopt deterministic oracle computation as our notion of higher-order computation, we are led to a different notion of computable functional. This kind of functional was first discussed by Kleene [1952]:[4]

Definition 5. A functional $F : \mathcal{P} \to \mathcal{P}$ is a *partial recursive functional* if $F(f)$ can be obtained from $f : \mathbb{N} \rightharpoonup \mathbb{N}$ and the initial functions by composition, primitive recursion, and minimalisation. If its domain is restricted to the set $\mathcal{F}$ of total functions, such a functional $F : \mathcal{F} \to \mathcal{P}$ is called a *restricted partial recursive functional*.

[4]However, his definition was not identical to ours. Kleene defined his functionals through an equation calculus. In this framework, even if the semantics of composition were understood to be strict, multiple (and possibly inconsistent) defining equations were still allowed, leading to the non-determinism observed by Platek [1966], which allowed parallel-or to be computable. We use the definition that is widely believed that Kleene really intended to use, and found in later textbooks. That this is the common interpretation I learned from John Longley, in personal communication.

Thus, if $F(f) = g$, then we say that g is partial recursive *uniformly in* f.

An implementation of such a functional F would resemble a deterministic Turing machine with an oracle for its argument. But notice that there is no non-determinism in this case, and hence calls to the oracle have to happen in a predetermined way. As soon as we decide to make a query at an undefined point, the computation diverges: there is no other branch of the computation to save the day! Informally, we may say that *calls to the oracle may not be dovetailed*. In effect, partial recursive functionals deal with their arguments as *pure extensions*, whereas effectively continuous functionals interact in a more involved manner with the phenomenon of non-termination.

In the case of total inputs, the above connection was made precise by Kleene:

Theorem 14. *[Kleene, 1952, §68, XXVIII] A functional $F : \mathcal{F} \to \mathcal{P}$ is a restricted partial recursive functional if and only if it is computed by a deterministic Turing machine with an oracle.*

We are not aware of a plausible analogue of this theorem for partial recursive functionals and deterministic Turing machines.

The definition of partial recursive functionals has a lot of undesirable consequences: see the discussion in the thesis of Platek [1966, p. 128-130]. Thus, the definition is often restricted to total inputs, for which the above characterisation through Turing Machines exists. The underlying reason seems to be that, for total inputs, we may *enumerate* the graph of the oracle, as no call to it will diverge.

Let us not forget this trivial but pleasant consequence:

Lemma 3. *Let $F : \mathcal{P} \to \mathcal{P}$ be a partial recursive functional. If $g \in \mathcal{PR}$, then $F(g) \in \mathcal{PR}$.*

It is in this setting that Kleene obtained the First Recursion Theorem, which first appeared in Kleene [1952, §66]:

Theorem 15 (FRT for Partial Recursive Functionals)**.** *Let $F : \mathcal{P} \to \mathcal{P}$ be a partial recursive functional. Then F admits a partial recursive least fixed point.*

Proof. See Odifreddi [1992, §II.3.15]. As before, the existence of the least fixed point follows from the Fixpoint Theorem. The fact it is partial recursive follows from Lemma 3: we conclude by induction that all the f_i's are partial recursive; as the least fixed point is $f = \bigsqcup_i f_i$, we have that

$$f(x) \simeq y \quad \iff \quad \exists i \in \mathbb{N}.\ f_i(x) \simeq y$$

As the f_i's are partial recursive, the predicate on the RHS of this equivalence is recursively enumerable, and hence so is the graph of f. □

Notice that the proof was rather abstract, and that all references to indices have disappeared completely.

The following was shown by Uspenskii and Nerode, see Odifreddi [1992, §II.3.19] for a proof:

Theorem 16. *Every partial recursive functional is effectively continuous.*

In particular, if we restrict a partial recursive functional to $\mathcal{PR}$, it is an effective operation. The converse was shown to fail by Sasso—see Odifreddi [1992, §II.3.20]:

Theorem 17. *The functional*

$$F(f) = \lambda x. \begin{cases} 0 & \text{if } f(2x) \simeq 0 \text{ or } f(2x+1) \simeq 0 \\ \text{undefined} & \text{otherwise} \end{cases}$$

is effectively continuous, but not partial recursive.

This clearly demonstrates, once more, that there is inherent *parallelism* or *non-determinism* in effective operations, whilst partial recursive functionals are purely *sequential*. In more detail, to compute the above functional we would have to concurrently query the argument f at two points by dovetailing the computations. A deterministic Turing machine would have to either query f at either $2x$ or $2x+1$ first; if the first call were to an undefined point, it would diverge and never examine the second. A non-deterministic Turing machine would deal with the same difficulty by branching at the point where a choice between $2x$ and $2x+1$ is to be made.

2.3 FRT vs. SRT

2.3.1 Effective Operations and the SRT

Suppose we would like to construct a fixed point in a more simplistic manner than the one employed in the proof of Theorem 13. All we need to do is use the Myhill-Shepherdson theorem to restrict an effectively continuous functional to an effective operation and extract an extensional function from it, followed by applying the SRT.

Lemma 4. *Given an effective operation $F : \mathcal{PR} \to \mathcal{PR}$ defined by an extensional $\phi_p : \mathbb{N} \rightharpoonup \mathbb{N}$, we may effectively obtain a code for one of its fixed points from $p \in \mathbb{N}$.*

Proof. By Theorem 9, that code is $n(p)$; for then,

$$\phi_{n(p)} = \phi_{\phi_p(n(p))} = F(\phi_{n(p)})$$

so that $\phi_{n(p)}$ is a fixed point of F. □

So far, so good; but what sort of fixed point have we obtained? In particular, is it minimal? The following construction, due to Rogers [1987, §11-XIII] demonstrates that it is not.

Theorem 18. *There is an extensional $\phi_m : \mathbb{N} \rightharpoonup \mathbb{N}$ such that the fixed point obtained by the SRT as in Lemma 4 is not minimal.*

Proof. Use Church's Thesis, Kleene's SRT, and the function n from 9 to define $m \in \mathbb{N}$ such that

$$\phi_m(x) \simeq \begin{cases} x & \text{if } x \neq n(m) \\ t & \text{if } x = n(m) \end{cases}$$

where t is an index for the constant zero function, i.e. $\phi_t(x) \simeq 0$ for all $x \in \mathbb{N}$. Observe that, as n is total recursive, ϕ_m is total recursive. It is also extensional. Essentially, ϕ_m asks: is the input my own Rogers fixed point? If yes, output code for the constantly zero function; otherwise, echo the input. Thus, if $x \neq n(m)$, we have that $\phi_m(x) \simeq x$, so that $\phi_{\phi_m(x)} \simeq \phi_x$. Otherwise,

$$\phi_{\phi_m(n(m))} \simeq \phi_{n(m)}$$

as $n(m)$ is a Rogers-style fixed point. In either case, ϕ_m is extensional, and defines the identity functional. The least fixed point of it is the empty function. However, the fixed point that results from the SRT has code $n(m)$, and

$$\phi_m(n(m)) \simeq t$$

so that $\phi_{n(m)} = \phi_t$, which is equal to the constantly zero function. □

The key aspect of this construction seems to be that, unlike oracle computation, an extensional function is able to syntactically inspect its input, thus creating a 'singularity' at one point. We maintain extensionality by arranging that the point at which the 'singularity' is to be found is—incidentally—the extensional function's own fixed point! This is more evidence that effective operations really hide something more than 'pure extension' under the hood.

The Standard Form

Can this mend this situation? The answer is positive: any extensional function can be rewritten in a 'standard form,' which guarantees that the SRT really defines a minimal fixed point. This is the exact sense in which the SRT implies the FRT.

The construction is due to Rogers [1987, §11-XIV]. The original statement is horribly complicated, and involves multiple layers of enumeration; there is a lot of concurrency happening here, and we cannot do much better than keep the description informal.

For the following, we assume that there is also a standard way to enumerate the graph of a partial recursive function given its index. This may be done by dovetailing simulations of

$$\phi_x(0),\ \phi_x(1), \ldots$$

and emitting pairs $(i, \phi_x(i))$ as soon as the ith simulation halts. We do not care about the exact details, but we do care that the exact same construction is used throughout.

Thus, let there be a effective operation $F : \mathcal{PR} \to \mathcal{PR}$, and let $f : \mathbb{N} \rightharpoonup \mathbb{N}$ be total and extensional, such that $F(\phi_x) = \phi_{f(x)}$. We will define a total and extensional $h_f : \mathbb{N} \rightharpoonup \mathbb{N}$, which is *co-extensional* with f, in the sense that

$$\forall x \in \mathbb{N}.\ \phi_{f(x)} = \phi_{h_f(x)}$$

This h_f will be in 'standard' form. Moreover, we may effectively compute an index for h_f from an index for f.

We use Church's Thesis to define h_f so that, on input y, it outputs a program that performs the following instructions:

$h_f(y) \simeq$ "On input x, run the following processes in parallel:

1. One process enumerates the graphs of all finite functions $\mathbb{N} \rightharpoonup \mathbb{N}$, encoded as numbers:

   $$\hat{\theta_0} = \emptyset,\ \hat{\theta_1},\ \hat{\theta_2},\ \ldots$$

   This may be done in many ways, but it is necessary that we begin with the empty function (in order to include the covert base case in the strong induction of the following theorem).

2. There is a total recursive $d : \mathbb{N} \rightharpoonup \mathbb{N}$ which turns a graph of a finite function into an index for that function (by writing code that simply checks if the input is in the graph, outputting the relevant value if so, and diverging otherwise). The second process receives the encoded graphs of the finite functions above, and enumerates codes,

   $$d(\hat{\theta_0}),\ d(\hat{\theta_1}),\ d(\hat{\theta_2}),\ \ldots$$

   with $\phi_{d(\hat{\theta_i})} = \theta_i$.

3. A third process receives messages from the second process, and applies f to those codes, outputting

$$f(d(\hat{\theta_0})),\ f(d(\hat{\theta_1})),\ \ldots$$

with $\phi_{f(d(\hat{\theta_i}))} = F(\phi_{d(\hat{\theta_i})}) = F(\theta_i)$. We thus obtain codes for all the applications of the effective operation F on all the finite functions. Beware: these functions $\phi_{f(d(\hat{\theta_i}))}$ may now be infinite!

4. (This is the process where partiality enters the construction.) Enumerate, simultaneously, all the pairs in the graph of the function ϕ_y, as well as the pairs in the graphs of $\phi_{f(d(\hat{\theta_i}))} = F(\theta_i)$. This can be done using the method postulated above.

5. As soon as we find some t such that

$$F(\theta_i)(x) \simeq t \quad \text{and} \quad \theta_i \subseteq \phi_y$$

we halt and output that t. This may be done by periodically checking whether x is defined in the enumeration of some $F(\theta_i)$, and then confirming that the entire graph of that θ_i is contained in the enumeration of ϕ_y."

Notice that, in this construction, the code for f may be abstracted away. Using the s-m-n theorem, we can then effectively produce code for it from any index of f. Trivially, h_f is total. Furthermore, notice that we needed to enumerate the $\phi_{f(d(\hat{\theta_i}))}$, for—in general—they will not be finite functions.

By the compactness of F, which follows from the theorem of Myhill and Shepherdson, we know that, $F(\phi_y)(x) \simeq t$ if and only if $F(\theta_i)(x) \simeq y$ for some finite $\theta_i \subseteq \phi_y$. This construction will always find such a θ_i if there exists one. Hence h_f defines the same functional as f.

Now, using the SRT on h_f will produce a minimal fixed point:

Theorem 19. *If $\phi_v = h_f$, then $n(v)$ is a code for the least fixed point of the effective operation $F : \mathcal{PR} \to \mathcal{PR}$ defined by $f : \mathbb{N} \rightharpoonup \mathbb{N}$.*

Proof. We have that

$$\phi_{n(v)} = \phi_{\phi_v(n(v))} = \phi_{h_f(n(v))}$$

so that $n(v)$ behaves exactly as $h_f(y)$ would if y were fixed to be its own code. That is to say, we can read $\phi_{n(v)}$ wherever ϕ_y is occurs in the definition of h_f, and the check

$$\theta_i \subseteq \phi_y$$

becomes

$$\theta_i \subseteq \phi_{n(v)}$$

which is to say that the program checks whether each finite function is a subset of its own graph! By Lemma 4, this defines a fixed point for the functional F.

To prove that this fixed point is least, we proceed by *strong induction on the number of steps taken to enumerate the graph of* $\phi_{n(v)}$. That is, we shall show that if we begin enumerating $\phi_{n(v)}$ using our standard enumeration procedure on the code $n(v)$, all the pairs produced will belong to the least fixed point, hence $\phi_{n(v)} \subseteq \mathbf{lfp}(F)$, whence $\phi_{n(v)} = \mathbf{lfp}(F)$.

Begin enumerating $\phi_{n(v)}$. This involves running the code $n(v)$. One of the subprocesses in that code involves enumerating $\phi_{n(v)}$ itself, using the same procedure as we are. Since this is a *sub-computation* of our enumeration, it is always shorter in length. Hence, by the inductive hypothesis, we assume that the enumeration in the sub-computation produces the least fixed point. Hence, if the check

$$\theta_i \subseteq \phi_{n(v)}$$

succeeds, we know that

$$\theta_i \subseteq \mathbf{lfp}(F)$$

By monotonicity, it follows that

$$F(\theta_i) \subseteq F(\mathbf{lfp}(F)) = \mathbf{lfp}(F)$$

Hence, when the check $F(\theta_i)(x) \simeq t$ succeeds and the pair (x, t) is output by the enumeration, we know it belongs to the least fixed point. It follows that every pair produced by the enumeration is in the least fixed point. □

2.3.2 Partial Recursive Functionals and the SRT

In contrast with effective operations, the situation is simpler in the case of oracle computation: because partial recursive functionals are decidedly extensional in their behaviour, the problems that arose in the preceding section vanish. There is no 'inherent' parallelism in computing such a functional, and the SRT immediately yields least fixed points.

The following theorem was shown by Odifreddi [1992, §II.3.16], who wrongly attributes it to Rogers:[5]

[5]Rogers instead sketched the proof to our Theorem 19, which strictly concerns effective operations.

Theorem 20 (Odifreddi)**.** *Let $F : \mathcal{P} \to \mathcal{P}$ be a partial recursive functional, and define*

$$f(e, x) \simeq F(\phi_e)(x)$$

Then f is partial recursive. Moreover, there exists $q \in \mathbb{N}$ such that $f = \phi_q$, and the function $h : \mathbb{N} \rightharpoonup \mathbb{N}$ of Theorem 7 produces a code $h(q)$ such that $\phi_{h(q)} = \mathbf{lfp}(F)$.

Proof. As F is a partial recursive functional, it is also an effective operation on the partial recursive functions, by Theorem 16. We define $q \in \mathbb{N}$ by Church's thesis: on input (e, x), process the code of e with the total extensional function associated to F by Myhill-Shepherdson, and call the resulting code on x.

Let $g = \phi_{h(q)}$. We have that

$$g(x) \simeq \phi_{h(q)}(x) \simeq f(h(q), x) \simeq F(\phi_{h(q)})(x) \simeq F(g)(x)$$

for any $x \in \mathbb{N}$, so that g is a fixed point of F.

It remains to show minimality. The proof is by *strong induction on the length of computations of $F(g)$ on its arguments.* Suppose $F(g)(x) \simeq t$. F is effectively continuous, so there exists a finite $\theta \subseteq g$ such that

$$F(\theta)(x) \simeq t$$

by compactness. Choose a minimal such θ, and let

$$\theta = \{\, (x_1, y_1), \ldots, (x_n, y_n) \,\}$$

By construction, the x_i are exactly the 'questions' with which a Turing machine that computes F would query the oracle, on input x.

By using Kleene's SRT, we have replaced calls to the oracle by recursive calls to another copy of itself. It follows that the computation of each $F(g)(x_i) \simeq y_i$ is strictly shorter in length than the overall computation of $F(g)(x) \simeq y$. Hence, by the induction hypothesis, $(x_i, y_i) \in \mathbf{lfp}(F)$ for all i, and $\theta \subseteq \mathbf{lfp}(F)$.

By monotonicity, $F(\theta) \subseteq \mathbf{lfp}(F)$, and since $F(\theta)(x) \simeq t$, we have that $(x, t) \in \mathbf{lfp}(F)$. Hence $g \subseteq \mathbf{lfp}(F)$, and as g is also a fixed point, equality holds. □

Chapter 3

iPCF: An Intensional Programming Language[1]

This chapter concerns the elaboration of the modality-as-intension interpretation that we introduced in §1.4. Our starting point will be the Davies-Pfenning calculus for S4, which is a typed λ-calculus with *modal types*. The intuitive meaning of the modal type $\Box A$ will be that of *code*, that—when evaluated—yields a value of type A. We wish to use this calculus for intensional and reflective programming.

The Davies-Pfenning calculus already supports a notion of *programs-as-data*: to each term $M : A$ that uses only 'code' variables there corresponds a term $\mathsf{box}\, M : \Box A$ that stands for the term M considered as a datum. This is already considerably stronger than ordinary higher-order functional programming with 'functions as first-class citizens,' as it also entails a kind of *homoiconicity*, similar to the one present in dialects of Lisp. But we want to go even further than that: in Lisp, a program is able to process code by treating it as mere symbols, thereby disregarding its observable behaviour.

The true spirit of intensionality is the ability to support operations that are, according to the extensional viewpoint, *non-functional*. This was not the case in the work of Davies and Pfenning, who merely used their calculus for *staged metaprogramming*, which did not require non-functional operations. In this chapter we shall mend this. We shall augment their calculus by adding *intensional operations*, and *intensional recursion*. We shall call the resulting calculus *Intensional PCF*, after the simply-typed λ-calculus with (extensional) fixed points studied by Scott [1993] and Plotkin [1977].

[1]This chapter is based on the paper [Kavvos, 2017d], which was presented at the 7th workshop on Intuitionistic Modal Logics and Applications (IMLA 2017). A preprint is available as arXiv:1703.01288

There has been some previous work on adding intensional operations to the Davies-Pfenning calculus. A complicated system based on nominal techniques that fleshed out those ideas was presented by Nanevski [2002]. The notions of intensional and extensional equality implicit in this system were studied using logical relations by Pfenning and Nanevski Nanevski and Pfenning [2005]. However, none of these papers studied whether the induced equational systems are consistent. We show that, no matter the intensional mechanism at use, modalities enable consistent intensional programming.

To our knowledge, this chapter presents (a) the first consistency proof for type-safe intensional programming, and (b) the first type-safe attempt at reflective programming.

3.1 Introducing Intensional PCF

Intensional PCF (iPCF) is a typed λ-calculus with modal types. As discussed before, the modal types work in our favour by separating intension from extension, so that the latter does not leak into the former. Given the logical flavour of our observations in §1.4 we shall model the types of iPCF after the *constructive modal logic* $\mathsf{S4}$, in the dual-context style pioneered by Pfenning and Davies [Pfenning and Davies, 2001, Davies and Pfenning, 2001]. Let us seize this opportunity to remark that (a) there are also other ways to capture $\mathsf{S4}$, for which see the survey [Kavvos, 2016], and that (b) dual-context formulations are not by any means limited to $\mathsf{S4}$: they began in the context of *intuitionistic linear logic*, but have recently been shown to also encompass other modal logics; see Kavvos [2017b].

iPCF is *not* related to the language Mini-ML that is introduced by Davies and Pfenning [2001]: that is a call-by-value, ML-like language, with ordinary call-by-value fixed points. In contrast, ours is a call-by-name language with a new kind of fixed point, namely intensional fixed points. These fixed points will afford the programmer the full power of *intensional recursion*. In logical terms they correspond to throwing the Gödel-Löb axiom $\Box(\Box A \to A) \to \Box A$ into $\mathsf{S4}$. Modal logicians might object to this, as, in conjunction with the T axiom $\Box A \to A$, it will make every type inhabited. We remind them that a similar situation occurs in PCF, where the $\mathbf{Y}_A : (A \to A) \to A$ combinator allows one to write a term $\mathbf{Y}_A(\lambda x{:}A.\ x)$ at every type A. As in the study of PCF, we care less about the logic and more about the underlying computation: *it is the terms that matter, and the types are only there to stop type errors from happening.*

The syntax and the typing rules of iPCF may be found in Figure 3.1. These are largely the same as Pfenning and Davies' S4, save the addition of some constants (drawn from PCF), and a rule for intensional recursion. The introduction rule for the modality restricts terms under a box $(-)$ to those containing only modal variables, i.e. variables carrying only intensions or code, but never 'live values:'

$$\frac{\Delta\,;\cdot \vdash M : A}{\Delta\,;\Gamma \vdash \mathsf{box}\; M : \Box A}$$

There is also a rule for intensional recursion:

$$\frac{\Delta\,;z : \Box A \vdash M : A}{\Delta\,;\Gamma \vdash \mathsf{fix}\; z\; \mathsf{in}\; M : A}$$

This will be coupled with the reduction $\mathsf{fix}\; z\; \mathsf{in}\; M \longrightarrow M[\mathsf{box}\; (\mathsf{fix}\; z\; \mathsf{in}\; M)/z]$. This rule is actually just *Löb's rule* with a modal context, and including it in the Hilbert system of a (classical or intuitionistic) modal logic is equivalent to including the Gödel-Löb axiom: see Boolos [1994] and Ursini [1979b]. We recommend the survey Litak [2014] for a broad coverage of constructive modalities with a provability-like flavour. Finally, let us record a fact noticed by Samson Abramsky, which is that erasing the modality from the types appearing in either Löb's rule or the Gödel-Löb axiom yields the type of $\mathbf{Y}_A : (A \to A) \to A$, as a rule in the first case, or axiomatically internalised as a constant in the second (both variants exist in the literature: see Gunter [1992] and Mitchell [1996].)

3.2 Metatheory

This section concerns the basic metatheoretic properties of iPCF. The expected structural rules are admissible. We also prove a theorem regarding the behaviour of free variables, similar to the ones in [Kavvos, 2017b], which demonstrates how the different layers of intension and extension are separated by the type system.

3.2.1 Structural Theorems & Cut

iPCF satisfies the expected basic results: structural and cut rules are admissible. This is no surprise given its origin in the well-behaved Davies-Pfenning calculus. We assume the typical conventions for λ-calculi: terms are identified up to α-equivalence, for which we write $\equiv$, and substitution $[\cdot/\cdot]$ is defined in the ordinary, capture-avoiding manner. Bear in mind that we consider occurrences of u in N to be bound in

Figure 3.1: Syntax and Typing Rules for Intensional PCF

$$\begin{array}{rrcl}
\textbf{Ground Types} & G & ::= & \mathsf{Nat} \mid \mathsf{Bool} \\
\textbf{Types} & A, B & ::= & G \mid A \to B \mid \Box A \\
\textbf{Terms} & M, N & ::= & x \mid \lambda x{:}A.\ M \mid MN \mid \mathsf{box}\ M \mid \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N \mid \\
 & & & \widehat{n} \mid \mathsf{true} \mid \mathsf{false} \mid \mathsf{succ} \mid \mathsf{pred} \mid \mathsf{zero?} \mid\ \supset_G\ \mid \mathsf{fix}\ z\ \mathsf{in}\ M \\
\textbf{Contexts} & \Gamma, \Delta & ::= & \cdot \mid \Gamma, x : A
\end{array}$$

$$\frac{}{\Delta\,;\Gamma \vdash \widehat{n} : \mathsf{Nat}} \qquad \frac{}{\Delta\,;\Gamma \vdash b : \mathsf{Bool}}\ (b \in \{\mathsf{true}, \mathsf{false}\})$$

$$\frac{}{\Delta\,;\Gamma \vdash \mathsf{zero?} : \mathsf{Nat} \to \mathsf{Bool}} \qquad \frac{}{\Delta\,;\Gamma \vdash f : \mathsf{Nat} \to \mathsf{Nat}}\ (f \in \{\mathsf{succ}, \mathsf{pred}\})$$

$$\frac{}{\Delta\,;\Gamma \vdash \supset_G : \mathsf{Bool} \to G \to G \to G}$$

$$\frac{}{\Delta\,;\Gamma, x{:}A, \Gamma' \vdash x : A}\ (\mathsf{var}) \qquad \frac{}{\Delta, u{:}A, \Delta'\,;\Gamma \vdash u : A}\ (\Box\mathsf{var})$$

$$\frac{\Delta\,;\Gamma, x{:}A \vdash M : B}{\Delta\,;\Gamma \vdash \lambda x{:}A.\ M : A \to B}\ (\to \mathcal{I}) \qquad \frac{\Delta\,;\Gamma \vdash M : A \to B \quad \Delta\,;\Gamma \vdash N : A}{\Delta\,;\Gamma \vdash MN : B}\ (\to \mathcal{E})$$

$$\frac{\Delta\,;\cdot \vdash M : A}{\Delta\,;\Gamma \vdash \mathsf{box}\ M : \Box A}\ (\Box\mathcal{I}) \qquad \frac{\Delta\,;\Gamma \vdash M : \Box A \qquad \Delta, u{:}A\,;\Gamma \vdash N : C}{\Delta\,;\Gamma \vdash \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N : C}\ (\Box\mathcal{E})$$

$$\frac{\Delta\,;z : \Box A \vdash M : A}{\Delta\,;\Gamma \vdash \mathsf{fix}\ z\ \mathsf{in}\ M : \Box A}\ (\Box\mathsf{fix})$$

$\mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N$. Contexts Γ, Δ are lists of type assignments $x : A$. Furthermore, we shall assume that whenever we write a judgement like $\Delta\,;\Gamma \vdash M : A$, then Δ and Γ are *disjoint*, in the sense that $\textsc{Vars}\,(\Delta) \cap \textsc{Vars}\,(\Gamma) = \emptyset$, where $\textsc{Vars}\,(x_1 : A_1, \ldots, x_n : A_n) \stackrel{\text{def}}{=} \{x_1, \ldots, x_n\}$. We write Γ, Γ' for the concatenation of disjoint contexts. Finally, we sometimes write $\vdash M : A$ whenever $\cdot\,;\cdot \vdash M : A$.

Theorem 21 (Structural & Cut)**.** *The following rules are admissible in iPCF:*

1. *(Weakening)*

$$\frac{\Delta\,;\Gamma,\Gamma' \vdash M : A}{\Delta\,;\Gamma, x{:}A, \Gamma' \vdash M : A}$$

2. *(Exchange)*

$$\frac{\Delta\,;\Gamma, x{:}A, y{:}B, \Gamma' \vdash M : C}{\Delta\,;\Gamma, y{:}B, x{:}A, \Gamma' \vdash M : C}$$

3. *(Contraction)*

$$\frac{\Delta\,;\Gamma, x{:}A, y{:}A, \Gamma' \vdash M : A}{\Delta\,;\Gamma, w{:}A, \Gamma' \vdash M[w, w/x, y] : A}$$

4. *(Cut)*

$$\frac{\Delta\,;\Gamma \vdash N : A \qquad \Delta\,;\Gamma, x{:}A, \Gamma' \vdash M : A}{\Delta\,;\Gamma,\Gamma' \vdash M[N/x] : A}$$

Proof. All by induction on the typing derivation of M. Verified in the proof assistant Agda: see Appendix A.2. □

Theorem 22 (Modal Structural & Cut)**.** *The following rules are admissible:*

1. *(Modal Weakening)*

$$\frac{\Delta, \Delta'\,;\Gamma \vdash M : C}{\Delta, u{:}A, \Delta'\,;\Gamma \vdash M : C}$$

2. *(Modal Exchange)*

$$\frac{\Delta, x{:}A, y{:}B, \Delta'\,;\Gamma \vdash M : C}{\Delta, y{:}B, x{:}A, \Delta'\,;\Gamma \vdash M : C}$$

3. *(Modal Contraction)*

$$\frac{\Delta, x{:}A, y{:}A, \Delta'\,;\Gamma \vdash M : C}{\Delta, w{:}A, \Delta'\,;\Gamma \vdash M[w, w/x, y] : C}$$

4. *(Modal Cut)*

$$\frac{\Delta\,;\cdot \vdash N : A \quad \Delta, u{:}A, \Delta'\,;\Gamma \vdash M : C}{\Delta, \Delta'\,;\Gamma \vdash M[N/u] : C}$$

Proof. All by induction on the typing derivation of M. Verified in the proof assistant Agda: see Appendix A.2. □

3.2.2 Free variables

In this section we prove a theorem regarding the occurrences of free variables in well-typed terms of iPCF. It turns out that, if a variable occurs free under a $\mathsf{box}\,(-)$ construct, then it has to be in the modal context. This is the property that enforces that *intensions can only depend on intensions.*

Definition 6 (Free variables)**.**

1. The *free variables* $\mathrm{FV}(M)$ of a term M are defined by induction on the structure of the term:

$$\begin{aligned}
\mathrm{FV}(x) &\stackrel{\text{def}}{=} \{x\} \\
\mathrm{FV}(MN) &\stackrel{\text{def}}{=} \mathrm{FV}(M) \cup \mathrm{FV}(N) \\
\mathrm{FV}(\lambda x{:}A.\ M) &\stackrel{\text{def}}{=} \mathrm{FV}(M) - \{x\} \\
\mathrm{FV}(\mathsf{box}\ M) &\stackrel{\text{def}}{=} \mathrm{FV}(M) \\
\mathrm{FV}(\mathsf{fix}\ z\ \mathsf{in}\ M) &\stackrel{\text{def}}{=} \mathrm{FV}(M) - \{z\} \\
\mathrm{FV}(\mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N) &\stackrel{\text{def}}{=} \mathrm{FV}(M) \cup (\mathrm{FV}(N) - \{u\})
\end{aligned}$$

2. The *unboxed free variables* $\mathrm{FV}_0(M)$ of a term are those that do *not* occur under the scope of a $\mathsf{box}\,(-)$ or $\mathsf{fix}\ z\ \mathsf{in}\ (-)$ construct. They are formally defined by replacing the following clauses in the definition of $\mathrm{FV}(-)$:

$$\begin{aligned}
\mathrm{FV}_0(\mathsf{box}\ M) &\stackrel{\text{def}}{=} \emptyset \\
\mathrm{FV}_0(\mathsf{fix}\ z\ \mathsf{in}\ M) &\stackrel{\text{def}}{=} \emptyset
\end{aligned}$$

3. The *boxed free variables* $\mathrm{FV}_{\geq 1}(M)$ of a term M are those that *do* occur under the scope of a $\mathsf{box}\,(-)$ construct. They are formally defined by replacing the following clauses in the definition of $\mathrm{FV}(-)$:

$$\begin{aligned}
\mathrm{FV}_{\geq 1}(x) &\stackrel{\text{def}}{=} \emptyset \\
\mathrm{FV}_{\geq 1}(\mathsf{box}\ M) &\stackrel{\text{def}}{=} \mathrm{FV}(M) \\
\mathrm{FV}_{\geq 1}(\mathsf{fix}\ z\ \mathsf{in}\ M) &\stackrel{\text{def}}{=} \mathrm{FV}(M) - \{z\}
\end{aligned}$$

Theorem 23 (Free variables)**.**

1. *For every term* M, $\mathrm{FV}(M) = \mathrm{FV}_0(M) \cup \mathrm{FV}_{\geq 1}(M)$.

2. *If and* $\Delta\,;\Gamma \vdash M : A$, *then*

$$\begin{aligned}
\mathrm{FV}_0(M) &\subseteq \mathrm{VARS}(\Gamma) \cup \mathrm{VARS}(\Delta) \\
\mathrm{FV}_{\geq 1}(M) &\subseteq \mathrm{VARS}(\Delta)
\end{aligned}$$

Proof.

1. Trivial induction on M.

2. By induction on the derivation of $\Delta\,;\Gamma \vdash M : A$. We show the case for $(\Box\mathcal{I})$; the first statement is trivial, so we show the second:

$$\begin{aligned}
& \textsc{fv}_{\geq 1}\,(\mathsf{box}\ M) \\
= \;& \{\,\text{definition}\,\} \\
& \textsc{fv}\,(M) \\
= \;& \{\,(1)\,\} \\
& \textsc{fv}_0\,(M) \cup \textsc{fv}_{\geq 1}\,(M) \\
\subseteq \;& \{\,\text{IH, twice}\,\} \\
& (\textsc{Vars}\,(\Delta) \cup \textsc{Vars}\,(\cdot)) \cup \textsc{Vars}\,(\Delta) \\
= \;& \{\,\text{definition}\,\} \\
& \textsc{Vars}\,(\Delta)
\end{aligned}$$

□

3.3 Consistency of Intensional Operations

In this section we shall prove that the modal types of iPCF enable us to consistently add intensional operations on the modal types. These are *non-functional operations on terms* which are not ordinarily definable because they violate equality. All we have to do is assume them as constants at modal types, define their behaviour by introducing a notion of reduction, and then prove that the compatible closure of this notion of reduction is confluent. A known corollary of confluence is that the equational theory induced by the reduction is *consistent*, i.e. does not equate all terms.

There is a caveat involving extension flowing into intension. That is: we need to exclude from consideration terms where a variable bound by a λ occurs under the scope of a $\mathsf{box}\,(-)$ construct. These will never be well-typed, but—since we discuss types and reduction orthogonally—we also need to explicitly exclude them here too.

3.3.1 Adding intensionality

Pfenning and Davies [2001] suggested that the $\Box$ modality can be used to signify intensionality. In fact, in [Davies and Pfenning, 1996, 2001] they had prevented reductions from happening under $\mathsf{box}\,(-)$ construct, " [...] since this would violate its intensional nature." But the truth is that neither of these presentations included any genuinely non-functional operations at modal types, and hence their only use was for

homogeneous staged metaprogramming. Adding intensional, non-functional operations is a more difficult task. Intensional operations are dependent on *descriptions* and *intensions* rather than *values* and *extensions*. Hence, unlike reduction and evaluation, they cannot be blind to substitution. This is something that quickly came to light as soon as Nanevski [2002] attempted to extend the system of Davies and Pfenning to allow 'intensional code analysis' using nominal techniques.

A similar task was also recently taken up by Gabbay and Nanevski [Gabbay and Nanevski, 2013], who attempted to add a construct $\mathsf{is\text{-}app}$ to the system of Davies and Pfenning, along with the reduction rules

$$\begin{aligned} \mathsf{is\text{-}app}\ (\mathsf{box}\ PQ) &\longrightarrow \mathsf{true} \\ \mathsf{is\text{-}app}\ (\mathsf{box}\ M) &\longrightarrow \mathsf{false} \qquad \text{if } M \text{ is not of the form } PQ \end{aligned}$$

The function computed by $\mathsf{is\text{-}app}$ is truly intensional, as it depends solely on the syntactic structure of its argument: it merely checks if it syntactically is an application or not. As such, it can be considered a *criterion of intensionality*, albeit an extreme one: its definability conclusively confirms the presence of computation up to syntax.

Gabbay and Nanevski tried to justify the inclusion of $\mathsf{is\text{-}app}$ by producing denotational semantics for modal types in which the semantic domain $[\![\Box A]\!]$ directly involves the actual closed terms of type $\Box A$. However, something seems to have gone wrong with substitution. In fact, we believe that their proof of soundness is wrong: it is not hard to see that their semantics is not stable under the second of these two reductions: take M to be u, and let the semantic environment map u to an application PQ, and then notice that this leads to $[\![\mathsf{true}]\!] = [\![\mathsf{false}]\!]$. We can also see this in the fact that their notion of reduction is *not confluent*. Here is the relevant counterexample: we can reduce like this:

$$\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ (PQ)\ \mathsf{in\ is\text{-}app}\ (\mathsf{box}\ u) \longrightarrow \mathsf{is\text{-}app}\ (\mathsf{box}\ PQ) \longrightarrow \mathsf{true}$$

But we could have also reduced like that:

$$\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ (PQ)\ \mathsf{in\ is\text{-}app}\ (\mathsf{box}\ u) \longrightarrow \mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ (PQ)\ \mathsf{in\ false} \longrightarrow \mathsf{false}$$

This example is easy to find if one tries to plough through a proof of confluence: it is very clearly *not* the case that $M \longrightarrow N$ implies $M[P/u] \longrightarrow N[P/u]$ if u is under a $\mathsf{box}\ (-)$, exactly because of the presence of intensional operations such as $\mathsf{is\text{-}app}$.

Perhaps the following idea is more workable: let us limit intensional operations to a chosen set of functions $f : \mathcal{T}(A) \to \mathcal{T}(B)$ from terms of type A to terms

of type B, and then represent them in the language by a constant $\tilde{f}$, such that $\tilde{f}(\mathsf{box}\, M) \longrightarrow \mathsf{box}\, f(M)$. This set of functions would then be chosen so that they satisfy some sanity conditions. Since we want to have a let construct that allows us to substitute code for modal variables, the following general situation will occur: if $N \longrightarrow N'$, we have

$$\mathsf{let\ box}\, u \Leftarrow \mathsf{box}\, M\ \mathsf{in}\ N \longrightarrow N[M/u]$$

but also

$$\mathsf{let\ box}\, u \Leftarrow \mathsf{box}\, M\ \mathsf{in}\ N \longrightarrow \mathsf{let\ box}\, u \Leftarrow \mathsf{box}\, M\ \mathsf{in}\ N' \longrightarrow N'[M/u]$$

Thus, in order to have confluence, we need $N[M/u] \longrightarrow N'[M/u]$. This will only be the case for reductions of the form $\tilde{f}(\mathsf{box}\, M) \to \mathsf{box}\, f(M)$ if $f(N[M/u]) \equiv f(N)[M/u]$, i.e. if f is *substitutive*. But then a simple naturality argument gives that $f(N) \equiv f(u[N/u]) \equiv f(u)[N/u]$, and hence $\tilde{f}$ is already definable by

$$\lambda x : \Box A.\ \mathsf{let\ box}\, u \Leftarrow x\ \mathsf{in\ box}\, f(u)$$

so such a 'substitutive' function is not intensional after all.

In fact, the only truly intensional operations we can add to our calculus will be those acting on *closed* terms. We will see that this circumvents the problems that arise when intensionality interacts with substitution. Hence, we will limit intensional operations to the following set:

Definition 7 (Intensional operations)**.** Let $\mathcal{T}(A)$ be the set of (α-equivalence classes of) closed terms such that $\cdot\,;\cdot \vdash M : A$. Then, the set of *intensional operations*, $\mathcal{F}(A, B)$, is defined to be the set of all functions $f : \mathcal{T}(A) \to \mathcal{T}(B)$.

We will include all of these intensional operations $f : \mathcal{T}(A) \to \mathcal{T}(B)$ in our calculus, as constants:

$$\Delta\,;\Gamma \vdash \tilde{f} : \Box A \to \Box B$$

with reduction rule $\tilde{f}(\mathsf{box}\, M) \to \mathsf{box}\, f(M)$, under the proviso that M is closed. Of course, these also includes operations on terms that might *not be computable*. However, we are interested in proving consistency of intensional operations in the most general setting. The questions of which intensional operations are computable, and which primitives can and should be used to express them, are both still open.

Figure 3.2: Reduction for Intensional PCF

$$\frac{}{(\lambda x{:}A.\ M)N \longrightarrow M[N/x]}\ (\longrightarrow \beta) \qquad \frac{M \longrightarrow N}{\lambda x{:}A.\ M \longrightarrow \lambda x{:}A.\ N}\ (\mathsf{cong}_\lambda)$$

$$\frac{M \longrightarrow N}{MP \longrightarrow NP}\ (\mathsf{app}_1) \qquad \frac{P \longrightarrow Q}{MP \longrightarrow MQ}\ (\mathsf{app}_2)$$

$$\frac{}{\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N \longrightarrow N[M/u]}\ (\Box\beta)$$

$$\frac{}{\mathsf{fix}\ z\ \mathsf{in}\ M \longrightarrow M[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z]}\ (\Box\mathsf{fix})$$

$$\frac{M\ \text{closed}}{\tilde{f}(\mathsf{box}\ M) \longrightarrow \mathsf{box}\ f(M)}\ (\Box\mathsf{int})$$

$$\frac{M \longrightarrow N}{\mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ P \longrightarrow \mathsf{let\ box}\ u \Leftarrow N\ \mathsf{in}\ P}\ (\mathsf{let\text{-}cong}_1)$$

$$\frac{P \longrightarrow Q}{\mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ P \longrightarrow \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ Q}\ (\mathsf{let\text{-}cong}_2)$$

$$\frac{}{\mathsf{zero?}\ \widehat{0} \longrightarrow \mathsf{true}}\ (\mathsf{zero?}_1) \qquad \frac{}{\mathsf{zero?}\ \widehat{n+1} \longrightarrow \mathsf{false}}\ (\mathsf{zero?}_2)$$

$$\frac{}{\mathsf{succ}\ \widehat{n} \longrightarrow \widehat{n+1}}\ (\mathsf{succ}) \qquad \frac{}{\mathsf{pred}\ \widehat{n} \longrightarrow \widehat{n \dot{-} 1}}\ (\mathsf{pred})$$

$$\frac{}{\supset_G\ \mathsf{true}\ M\ N \longrightarrow M}\ (\supset_1) \qquad \frac{}{\supset_G\ \mathsf{false}\ M\ N \longrightarrow N}\ (\supset_2)$$

Figure 3.3: Equational Theory for Intensional PCF

Function Spaces

$$\frac{\Delta\,;\Gamma \vdash N : A \qquad \Delta\,;\Gamma, x{:}A, \Gamma' \vdash M : B}{\Delta\,;\Gamma \vdash (\lambda x{:}A.M)\,N = M[N/x] : B}\,(\to \beta)$$

Modality

$$\frac{\Delta\,;\cdot \vdash M : A \qquad \Delta, u : A\,;\Gamma \vdash N : C}{\Delta\,;\Gamma \vdash \mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N = N[M/x] : C}\,(\Box\beta)$$

$$\frac{\Delta\,;z : \Box A \vdash M : A}{\Delta\,;\Gamma \vdash \mathsf{fix}\ z\ \mathsf{in}\ M = M[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z] : A}\,(\Box\mathsf{fix})$$

$$\frac{\cdot\,;\cdot \vdash M : A \quad f \in \mathcal{F}(A, B)}{\Delta\,;\Gamma \vdash \tilde{f}(\mathsf{box}\ M) = \mathsf{box}\ f(M) : \Box B}\,(\Box\mathsf{int})$$

$$\frac{\Delta\,;\Gamma \vdash M = N : \Box A \qquad \Delta\,;\Gamma \vdash P = Q : C}{\Delta\,;\Gamma \vdash \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ P = \mathsf{let\ box}\ u \Leftarrow N\ \mathsf{in}\ Q : B}\,(\Box\mathsf{let\text{-}cong})$$

Remark. In addition to the above, one should also include (a) rules that ensure that equality is an equivalence relation, (b) congruence rules for λ-abstraction and application, and (c) rules corresponding to the behaviour of constants, as in Figure 3.2.

3.3.2 Reduction and Confluence

We introduce a notion of reduction for iPCF, which we present in Figure 3.2. Unlike many studies of PCF-inspired languages, we do not consider a reduction strategy but ordinary 'non-deterministic' β-reduction. We do so because are trying to show consistency of the induced equational theory.

The equational theory induced by this notion of reduction is the one alluded to in the previous section: it is a symmetric version of it, annotated with types. It can be found in Figure 3.3. Note the fact that, like in the work of Davies and Pfenning, we do *not* include the congruence rule for the modality:

$$\frac{\Delta\,;\cdot \vdash M = N : A}{\Delta\,;\Gamma \vdash \mathsf{box}\ M = \mathsf{box}\ N : \Box A}\ (\Box\mathsf{cong})$$

In fact, the very absence of this rule is what will allow modal types to become intensional. Otherwise, the only new rules are intensional recursion, embodied by the rule ($\Box\mathsf{fix}$), and intensional operations, exemplified by the rule ($\Box\mathsf{int}$).

We note that it seems perfectly reasonable to think that we should allow reductions under fix, i.e. admit the rule

$$\frac{M \longrightarrow N}{\mathsf{fix}\ z\ \mathsf{in}\ M \longrightarrow \mathsf{fix}\ z\ \mathsf{in}\ N}$$

as M and N are expected to be of type A, which need not be modal. However, the reduction $\mathsf{fix}\ z\ \mathsf{in}\ M \longrightarrow M[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z]$ 'freezes' M under an occurrence of $\mathsf{box}\ (-)$, so that no further reductions can take place within it. Thus, the above rule would violate the intensional nature of boxes. We were likewise compelled to define $\mathrm{FV}_0\,(\mathsf{fix}\ z\ \mathsf{in}\ M) \stackrel{\text{def}}{=} \emptyset$ in the previous section: we should already consider M to be intensional, or under a box.

We can now show that

Theorem 24. *The reduction relation $\longrightarrow$ is confluent.*

We will use a variant of the proof in [Kavvos, 2017b], i.e. the method of *parallel reduction*. This kind of proof was originally discovered by Tait and Martin-Löf, and is nicely documented in Takahashi [1995]. Because of the intensional nature of our $\mathsf{box}\ (-)$ constructs, ours will be more nuanced and fiddly than any in *op. cit.* The method is this: we will introduce a second notion of reduction,

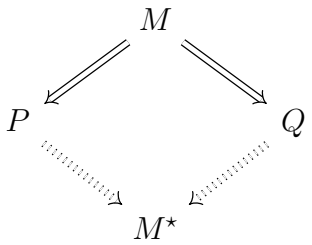
$$\Longrightarrow\ \subseteq \Lambda \times \Lambda$$

which we will 'sandwich' between reduction proper and its transitive closure:

$$\longrightarrow \quad \subseteq \Longrightarrow \subseteq \longrightarrow^*$$

We will then show that $\Longrightarrow$ has the diamond property. By the above inclusions, the transitive closure $\Longrightarrow^*$ of $\Longrightarrow$ is then equal to $\longrightarrow^*$, and hence $\longrightarrow$ is Church-Rosser.

In fact, we will follow Takahashi [1995] in doing something better: we will define for each term M its *complete development*, $M^\star$. The complete development is intuitively defined by 'unrolling' all the redexes of M at once. We will then show that if $M \Longrightarrow N$, then $N \Longrightarrow M^\star$. $M^\star$ will then suffice to close the diamond:



The parallel reduction $\Longrightarrow$ is defined in Figure 3.4. Instead of the axiom (refl) we would more commonly have an axiom for variables, $x \Longrightarrow x$, and $M \Longrightarrow M$ would be derivable. However, we do not have a congruence rule neither for $\mathsf{box}\,(-)$ nor for Löb's rule, so that possibility would be precluded. We are thus forced to include $M \Longrightarrow M$, which slightly complicates the lemmas that follow.

The main lemma that usually underpins the confluence proof is this: if $M \Longrightarrow N$ and $P \Longrightarrow Q$, $M[P/x] \Longrightarrow N[Q/x]$. However, this is intuitively wrong: no reductions should happen under boxes, so this should only hold if we are substituting for a variable *not* occurring under boxes. Hence, this lemma splits into three different ones:

- $P \Longrightarrow Q$ implies $M[P/x] \Longrightarrow M[Q/x]$, if x does not occur under boxes: this is the price to pay for replacing the variable axiom with (refl).
- $M \Longrightarrow N$ implies $M[P/u] \Longrightarrow N[P/u]$, even if u is under a box.
- If x does not occur under boxes, $M \Longrightarrow N$ and $P \Longrightarrow Q$ indeed imply $M[P/x] \Longrightarrow N[Q/x]$

But let us proceed with the proof.

Lemma 5. *If $M \Longrightarrow N$ then $M[P/u] \Longrightarrow N[P/u]$.*

Figure 3.4: Parallel Reduction

$$\frac{}{M \Longrightarrow M}\,(\mathsf{refl}) \qquad \frac{M \Longrightarrow N \qquad P \Longrightarrow Q}{(\lambda x{:}A.\ M)P \Longrightarrow N[Q/x]}\,(\to \beta)$$

$$\frac{M \Longrightarrow N}{\lambda x{:}A.\ M \Longrightarrow \lambda x{:}A.\ N}\,(\mathsf{cong}_\lambda) \qquad \frac{M \Longrightarrow N \qquad P \Longrightarrow Q}{MP \Longrightarrow NQ}\,(\mathsf{app})$$

$$\frac{P \Longrightarrow P'}{\supset_G\ \mathsf{true}\ P\ Q \Longrightarrow P'}\,(\supset_1) \qquad \frac{Q \Longrightarrow Q'}{\supset_G\ \mathsf{false}\ P\ Q \Longrightarrow Q'}\,(\supset_2)$$

$$\frac{M \Longrightarrow N}{\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ P\ \mathsf{in}\ M \Longrightarrow N[P/u]}\,(\Box\beta)$$

$$\frac{M \Longrightarrow M'}{\mathsf{fix}\ z\ \mathsf{in}\ M \Longrightarrow M'[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z]}\,(\Box\mathsf{fix})$$

$$\frac{M\ \text{closed}}{\tilde{f}(\mathsf{box}\ M) \Longrightarrow \mathsf{box}\ f(M)}\,(\Box\mathsf{int})$$

$$\frac{M \Longrightarrow N \qquad P \Longrightarrow Q}{\mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ P \Longrightarrow \mathsf{let\ box}\ u \Leftarrow N\ \mathsf{in}\ Q}\,(\Box\mathsf{let\text{-}cong})$$

Remark. In addition to the above, one should also include rules for the constants, but these are merely restatements of the rules in Figure 3.2.

Proof. By induction on the generation of $M \Longrightarrow N$. Most cases trivially follow, or consist of simple invocations of the IH. In the case of $(\to \beta)$, the known substitution lemma suffices. Let us look at the cases involving boxes.

CASE($\Box\beta$). Then $M \Longrightarrow N$ is $\mathsf{let\ box}\ v \Leftarrow \mathsf{box}\ R\ \mathsf{in}\ S \Longrightarrow S'[R/v]$ with $S \Longrightarrow S'$. By the IH, we have that $S[P/u] \Longrightarrow S'[P/u]$, so

$$\mathsf{let\ box}\ v \Leftarrow \mathsf{box}\ R[P/u]\ \mathsf{in}\ S[P/u] \Longrightarrow S'[P/u][R[P/u]/v]$$

and this last is α-equivalent to $S'[R/v][P/u]$ by the substitution lemma.

CASE($\Box\mathsf{fix}$). A similar application of the substitution lemma.

CASE($\Box\mathsf{int}$). Then $M \Longrightarrow N$ is $\tilde{f}(\mathsf{box}\ Q) \Longrightarrow \mathsf{box}\ f(Q)$. Hence

$$\left(\tilde{f}(\mathsf{box}\ Q)\right)[P/u] \equiv \tilde{f}(\mathsf{box}\ Q) \Longrightarrow \mathsf{box}\ f(Q) \equiv (\mathsf{box}\ f(Q))\,[P/u]$$

simply because both Q and $f(Q)$ are closed.

□

Lemma 6. *If $P \Longrightarrow Q$ and $x \notin \mathrm{FV}_{\geq 1}(M)$, then $M[P/x] \Longrightarrow M[Q/x]$.*

Proof. By induction on the term M. The only non-trivial cases are those for M a variable, $\mathsf{box}\ M'$ or $\mathsf{fix}\ z\ \mathsf{in}\ M'$. In the first case, depending on which variable M is, use either (refl), or the assumption $P \Longrightarrow Q$. In the latter two, $(\mathsf{box}\ M')[P/x] \equiv \mathsf{box}\ M' \equiv (\mathsf{box}\ M')[Q/x]$ as x does not occur under a box, so use (refl), and similarly for $\mathsf{fix}\ z\ \mathsf{in}\ M'$. □

Lemma 7. *If $M \Longrightarrow N$, $P \Longrightarrow Q$, and $x \notin \mathrm{FV}_{\geq 1}(M)$, then*

$$M[P/x] \Longrightarrow N[Q/x]$$

Proof. By induction on the generation of $M \Longrightarrow N$. The cases for most congruence rules and constants follow trivially, or from the IH. We prove the rest.

CASE(refl). Then $M \Longrightarrow N$ is actually $M \Longrightarrow M$, so we use Lemma 6 to infer $M[P/x] \Longrightarrow M[Q/x]$.

CASE($\Box\mathsf{int}$). Then $M \Longrightarrow N$ is actually $\tilde{f}(\mathsf{box}\ M) \Longrightarrow \mathsf{box}\ f(M)$. But M and $f(M)$ are closed, so $\left(\tilde{f}(\mathsf{box}\ M)\right)[P/x] \equiv \tilde{f}(\mathsf{box}\ M) \Longrightarrow \mathsf{box}\ f(M) \equiv (\mathsf{box}\ f(M))\,[Q/x]$.

CASE($\supset_i$). Then $M \Longrightarrow N$ is $\supset_G\ \mathsf{true}\ M\ N \Longrightarrow M'$ with $M \Longrightarrow M'$. By the IH, $M[P/x] \Longrightarrow M'[Q/x]$, so

$$\supset_G\ \mathsf{true}\ M[P/x]\ N[P/x] \Longrightarrow M'[Q/x]$$

by a single use of $(\supset_1)$. The case for false is similar.

CASE($\to \beta$). Then $(\lambda x'{:}A.\ M)N \Longrightarrow N'[M'/x']$, where $M \Longrightarrow M'$ and $N \Longrightarrow N'$. Then

$$((\lambda x'{:}A.\ M)N)\,[P/x] \equiv (\lambda x'{:}A.\ M[P/x])(N[P/x])$$

But, by the IH, $M[P/x] \Longrightarrow M'[Q/x]$ and $N[P/x] \Longrightarrow N'[Q/x]$. So by $(\to \beta)$ we have

$$(\lambda x'{:}A.\ M[P/x])(N[P/x]) \Longrightarrow M'[Q/x]\,[N'[Q/x]/x']$$

But this last is α-equivalent to $(M'[N'/x'])\,[Q/x]$ by the substitution lemma.

CASE($\Box\beta$). Then $\mathsf{let\ box}\ u' \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N \Longrightarrow N'[M/u']$ where $N \Longrightarrow N'$. By assumption, we have that $x \notin \mathrm{FV}\,(M)$ and $x \notin \mathrm{FV}_{\geq 1}\,(N)$. Hence, we have by the IH that $N[P/x] \Longrightarrow N'[Q/x]$, so by applying $(\Box\beta)$ we get

$$\begin{aligned}(\mathsf{let\ box}\ u' \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N)[P/x] &\equiv \mathsf{let\ box}\ u' \Leftarrow \mathsf{box}\ M[P/x]\ \mathsf{in}\ N[P/x] \\ &\equiv \mathsf{let\ box}\ u' \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N[P/x] \\ &\Longrightarrow N'[Q/x][M/u']\end{aligned}$$

But this last is α-equivalent to $N'[M/u'][Q/x]$, by the substitution lemma and the fact that x does not occur in M.

CASE($\Box\mathsf{fix}$). Then $\mathsf{fix}\ z\ \mathsf{in}\ M \Longrightarrow M'[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z]$, with $M \Longrightarrow M'$. As $x \notin \mathrm{FV}_{\geq 1}\,(\mathsf{fix}\ z\ \mathsf{in}\ M)$, we have that $x \notin \mathrm{FV}\,(M)$, and by Lemma 9, $x \notin \mathrm{FV}\,(M')$ either, so

$$(\mathsf{fix}\ z\ \mathsf{in}\ M)[P/x] \equiv \mathsf{fix}\ z\ \mathsf{in}\ M$$

and

$$M'[\mathsf{fix}\ z\ \mathsf{in}\ M/z][Q/x] \equiv M'[Q/x][\mathsf{fix}\ z\ \mathsf{in}\ M[Q/x]/z] \equiv M'[\mathsf{fix}\ z\ \mathsf{in}\ M/z]$$

Thus, a single use of $(\Box\mathsf{fix})$ suffices.

□

We now pull the following definition out of the hat:

Definition 8 (Complete development)**.** The *complete development* $M^\star$ of a term M is defined by the following clauses:

$$
\begin{aligned}
x^\star &\stackrel{\text{def}}{=} x \\
c^\star &\stackrel{\text{def}}{=} c \qquad (c \in \{\tilde{f}, \widehat{n}, \mathsf{zero?}, \dots\}) \\
(\lambda x{:}A.\ M)^\star &\stackrel{\text{def}}{=} \lambda x{:}A.\ M^\star \\
\left(\tilde{f}(\mathsf{box}\ M)\right)^\star &\stackrel{\text{def}}{=} \mathsf{box}\ f(M) \qquad \text{if } M \text{ is closed} \\
((\lambda x{:}A.\ M)\ N)^\star &\stackrel{\text{def}}{=} M^\star[N^\star/x] \\
(\supset_G\ \mathsf{true}\ M\ N)^\star &\stackrel{\text{def}}{=} M^\star \\
(\supset_G\ \mathsf{false}\ M\ N)^\star &\stackrel{\text{def}}{=} N^\star \\
(MN)^\star &\stackrel{\text{def}}{=} M^\star N^\star \\
(\mathsf{box}\ M)^\star &\stackrel{\text{def}}{=} \mathsf{box}\ M \\
(\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N)^\star &\stackrel{\text{def}}{=} N^\star[M/u] \\
(\mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N)^\star &\stackrel{\text{def}}{=} \mathsf{let\ box}\ u \Leftarrow M^\star\ \mathsf{in}\ N^\star \\
(\mathsf{fix}\ z\ \mathsf{in}\ M)^\star &\stackrel{\text{def}}{=} M^\star[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z]
\end{aligned}
$$

We need the following two technical results as well.

Lemma 8. $M \Longrightarrow M^\star$

Proof. By induction on the term M. Most cases follow immediately by (refl), or by the IH and an application of the relevant rule. The case for $\mathsf{box}\ M$ follows by (refl), the case for $\mathsf{fix}\ z\ \mathsf{in}\ M$ follows by ($\Box$fix), and the case for $\tilde{f}(\mathsf{box}\ M)$ by ($\Box$int). □

Lemma 9 (BFV antimonotonicity)**.** *If* $M \Longrightarrow N$ *then* $\mathrm{FV}_{\geq 1}(N) \subseteq \mathrm{FV}_{\geq 1}(M)$.

Proof. By induction on $M \Longrightarrow N$. □

And here is the main result:

Theorem 25. *If* $M \Longrightarrow P$*, then* $P \Longrightarrow M^\star$.

Proof. By induction on the generation of $M \Longrightarrow P$. The case of (refl) follows by Lemma 8, and the cases of congruence rules follow from the IH. We show the rest.

> Case($\to\ \beta$). Then we have $(\lambda x{:}A.\ M)N \Longrightarrow M'[N'/x]$, with $M \Longrightarrow M'$ and $N \Longrightarrow N'$. By the IH, $M' \Longrightarrow M^\star$ and $N' \Longrightarrow N^\star$. We have that $x \notin \mathrm{FV}_{\geq 1}(M)$, so by Lemma 9 we get that $x \notin \mathrm{FV}_{\geq 1}(M')$. Hence, by Lemma 7 we get $M'[N'/x] \Longrightarrow M^\star[N^\star/x] \equiv ((\lambda x{:}A.\ M)\ N)^\star$.

CASE($\Box\beta$). Then we have

$$\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N \Longrightarrow N'[M/u]$$

where $N \Longrightarrow N'$. By the IH, $N' \Longrightarrow N^\star$, so it follows that

$$N'[M/u] \Longrightarrow N^\star[M/u] \equiv (\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N)^\star$$

by Lemma 5.

CASE($\Box\mathsf{fix}$). Then we have

$$\mathsf{fix}\ z\ \mathsf{in}\ M \Longrightarrow M'[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z]$$

where $M \Longrightarrow M'$. By the IH, $M' \Longrightarrow M^\star$. Hence

$$M'[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z] \Longrightarrow M^\star[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z] \equiv (\mathsf{fix}\ z\ \mathsf{in}\ M)^\star$$

by Lemma 5.

CASE($\Box\mathsf{int}$). Similar.

□

As a result,

Corollary 2. *The equational theory of iPCF (Figure 3.3) is consistent.*

3.4 Some important terms

Let us look at the kinds of terms we can write in iPCF.

From the axioms of S4 First, we can write a term corresponding to axiom K, the *normality axiom* of modal logics:

$$\mathsf{ax_K} \stackrel{\text{def}}{=} \lambda f : \Box(A \to B).\ \lambda x : \Box A.\ \mathsf{let\ box}\ g \Leftarrow f\ \mathsf{in\ let\ box}\ y \Leftarrow x\ \mathsf{in\ box}\ (g\,y)$$

Then $\vdash \mathsf{ax_K} : \Box(A \to B) \to (\Box A \to \Box B)$. An intensional reading of this is the following: any function given as code can be transformed into an *effective operation* that maps code of type A to code of type B.

The rest of the axioms correspond to evaluating and quoting. Axiom T takes code to value, or intension to extension:

$$\vdash \mathsf{eval}_A \stackrel{\text{def}}{=} \lambda x : \Box A.\ \mathsf{let\ box}\ y \Leftarrow x\ \mathsf{in}\ y : \Box A \to A$$

and axiom 4 quotes code into code-for-code:

$$\vdash \mathsf{quote}_A \stackrel{\text{def}}{=} \lambda x : \Box A.\ \mathsf{let\ box}\ y \Leftarrow x\ \mathsf{in\ box}\ (\mathsf{box}\ y) : \Box A \to \Box\Box A$$

Undefined The combination of eval and intensional fixed points leads to non-termination, in a style reminiscent of the term $(\lambda x.\, xx)(\lambda x.\, xx)$ of the untyped λ-calculus. Let

$$\Omega_A \stackrel{\text{def}}{=} \mathsf{fix}\ z\ \mathsf{in}\ (\mathsf{eval}_A\ z)$$

Then $\vdash \Omega_A : A$, and

$$\Omega_A \longrightarrow\ \mathsf{eval}_A\ (\mathsf{box}\,\Omega_A)\ \longrightarrow^*\ \Omega_A$$

The Gödel-Löb axiom: intensional fixed points Since ($\Box\mathsf{fix}$) is Löb's rule, we expect to be able to write down a term corresponding to the Gödel-Löb axiom of provability logic. We can, and it is an *intensional fixed-point combinator*:

$$\mathbb{Y}_A \stackrel{\text{def}}{=} \lambda x : \Box(\Box A \to A).\ \mathsf{let\ box}\ f \Leftarrow x\ \mathsf{in\ box}\ (\mathsf{fix}\ z\ \mathsf{in}\ f\ z)$$

and $\vdash \mathbb{Y}_A : \Box(\Box A \to A) \to \Box A$. We observe that

$$\mathbb{Y}_A(\mathsf{box}\ M) \longrightarrow^*\ \mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ (M\ z))$$

Notice that, in this term, the modal variable f occurs free under a $\mathsf{fix}\ z\ \mathsf{in}\ (-)$ construct. This will prove important in §8, where this occurrence will be prohibited.

Extensional Fixed Points Perhaps surprisingly, the ordinary PCF $\mathbf{Y}$ combinator is also definable in the iPCF. Let

$$\mathbf{Y}_A \stackrel{\text{def}}{=} \mathsf{fix}\ z\ \mathsf{in}\ \lambda f : A \to A.\ f(\mathsf{eval}\ z\ f)$$

Then $\vdash \mathbf{Y}_A : (A \to A) \to A$, so that

$$\begin{aligned}\mathbf{Y}_A &\longrightarrow^*\ \lambda f : A \to A.\ f(\mathsf{eval}\ (\mathsf{box}\ \mathbf{Y}_A)\ f)) \\ &\longrightarrow^*\ \lambda f : A \to A.\ f(\mathbf{Y}_A\, f)\end{aligned}$$

Notice that, in this term, the modal variable z occurs free under a λ-abstraction. This will prove important in §8, where this occurrence will be prohibited.

3.5 Two intensional examples

No discussion of an intensional language with intensional recursion would be complete without examples that use these two novel features. Our first example uses intensionality, albeit in a functional way, and is drawn from the study of PCF and issues related to sequential vs. parallel (but not concurrent) computation. Our second example uses intensional recursion, so it is slightly more adventurous: it is a computer virus.

3.5.1 'Parallel or' by dovetailing

In [Plotkin, 1977] Gordon Plotkin proved the following theorem: there is no term $\mathsf{por} : \mathsf{Bool} \to \mathsf{Bool} \to \mathsf{Bool}$ of PCF such that $\mathsf{por}\ \mathsf{true}\ M \twoheadrightarrow_\beta \mathsf{true}$ and $\mathsf{por}\ M\ \mathsf{true} \twoheadrightarrow_\beta \mathsf{true}$ for any $\vdash M : \mathsf{Bool}$, whilst $\mathsf{por}\ \mathsf{false}\ \mathsf{false} \twoheadrightarrow_\beta \mathsf{false}$. Intuitively, the problem is that por has to first examine one of its two arguments, and this can be troublesome if that argument is non-terminating. It follows that the *parallel or* function is not definable in PCF. In order to regain the property of so-called *full abstraction* for the *Scott model* of PCF, a constant denoting this function has to be manually added to PCF, and endowed with the above rather clunky operational semantics. See [Plotkin, 1977, Gunter, 1992, Mitchell, 1996, Streicher, 2006].

However, the parallel or function is a computable *partial recursive functional* [Streicher, 2006, Longley and Normann, 2015]. The way to prove that is intuitively the following: given two closed terms $M, N : \mathsf{Bool}$, take turns in β-reducing each one for a one step: this is called *dovetailing*. If at any point one of the two terms reduces to true, then output true. But if at any point both reduce to false, then output false.

This procedure is not definable in PCF because a candidate term por does not have access to a code for its argument, but can only inspect its value. However, in iPCF we can use the modality to obtain access to code, and intensional operations to implement reduction. Suppose we pick a reduction strategy $\longrightarrow_r$. Then, let us include a constant $\mathsf{tick} : \Box\mathsf{Bool} \to \Box\mathsf{Bool}$ that implements one step of this reduction strategy on closed terms:

$$\frac{M \longrightarrow_r N,\ M, N \text{ closed}}{\mathsf{tick}\ (\mathsf{box}\ M) \longrightarrow \mathsf{box}\ N}$$

Also, let us include a constant $\mathsf{done?} : \Box\mathsf{Bool} \to \mathsf{Bool}$, which tells us if a closed term under a box is a normal form:

$$\frac{M \text{ closed, normal}}{\mathsf{done?}\ (\mathsf{box}\ M) \longrightarrow \mathsf{true}} \qquad \frac{M \text{ closed, not normal}}{\mathsf{done?}\ (\mathsf{box}\ M) \longrightarrow \mathsf{false}}$$

It is not hard to see that these two intensional operations can be subsumed under our previous scheme for introducing intensional operations: our proof still applies, yielding a consistent equational system.

The above argument is now implemented by the following term:

$$\begin{aligned}
\mathsf{por} :\equiv\ & \mathbf{Y}(\lambda \mathsf{por}.\ \lambda x : \Box\mathsf{Bool}.\ \lambda y : \Box\mathsf{Bool}. \\
& \supset_{\mathsf{Bool}}\ (\mathsf{done?}\ x) \qquad\qquad (\mathsf{lor}\ (\mathsf{eval}\ x)(\mathsf{eval}\ y)) \\
& \qquad\qquad\qquad\quad (\supset_{\mathsf{Bool}}\ (\mathsf{done?}\ y) \qquad (\mathsf{ror}\ (\mathsf{eval}\ x)(\mathsf{eval}\ y)) \\
& \qquad\qquad\qquad\qquad\qquad\quad (\mathsf{por}\ (\mathsf{tick}\ x)(\mathsf{tick}\ y)))
\end{aligned}$$

where $\mathsf{lor}, \mathsf{ror} : \mathsf{Bool} \to \mathsf{Bool} \to \mathsf{Bool}$ are terms defining the left-strict and right-strict versions of the 'or' connective respectively. Notice that the type of this term is $\Box\mathsf{Bool} \to \Box\mathsf{Bool} \to \mathsf{Bool}$: we require *intensional access* to the terms of boolean type in order to define this function!

3.5.2 A computer virus

Abstract computer virology is the study of formalisms that model computer viruses. There are many ways to formalise viruses. We will use the model of Adleman [1990], where files can be interpreted either as data, or as functions. We introduce a data type F of files, and two constants

$$\mathsf{in} : \Box(F \to F) \to F \quad \text{and} \quad \mathsf{out} : F \to \Box(F \to F)$$

If F is a file, then $\mathsf{out}\ F$ is that file interpreted as a program, and similarly for in. We ask that $\mathsf{out}\ (\mathsf{in}\ M) \longrightarrow M$, making $\Box(F \to F)$ a retract of F.[2] This might seem the same as the situation where $F \to F$ is a retract of F, which yields models of the (untyped) λ-calculus, and is not trivial to construct [Barendregt, 1984, §5.4]. However, in our case it is not nearly as worrying: $\Box(F \to F)$ is populated by programs and codes, not by actual functions. Under this interpretation, the pair $(\mathsf{in}, \mathsf{out})$ corresponds to a kind of Gödel numbering—especially if F is $\mathbb{N}$.

Now, in Adleman's model, a *virus* is a given by its infected form, which either *injures*, *infects*, or *imitates* other programs. The details are unimportant in the present discussion, save from the fact that the virus needs to have access to code that it can use to infect other executables. One can hence construct such a virus from its *infection routine*, by using Kleene's SRT. Let us model it by a term

$$\vdash \mathsf{infect} : \Box(F \to F) \to F \to F$$

which accepts a piece of viral code and an executable file, and it returns either the file itself, or a version infected with the viral code. We can then define a term

$$\vdash \mathsf{virus} \stackrel{\text{def}}{=} \mathsf{fix}\ z\ \mathsf{in}\ (\mathsf{infect}\ z) : F \to F$$

so that

$$\mathsf{virus} \longrightarrow^* \mathsf{infect}\ (\mathsf{box}\ \mathsf{virus})$$

which is a program that is ready to infect its input with its own code.

[2] Actually, in §8.5 and §8.6 we will see it is very easy to construct examples for the apparently more natural situation where $\mathsf{in} : \Box(\Box F \to F) \to F$, $\mathsf{out} : F \to (\Box F \to F)$, and $\mathsf{out}\ (\mathsf{in}\ M) \longrightarrow \mathsf{eval}_{F \to F}\ M$. Nevertheless, our setup is slightly more well-adapted to virology.

3.6 Open Questions

We have achieved the desideratum of an intensional programming language, with intensional recursion. There are two main questions that result from this development.

Firstly, does there exist a good set of *intensional primitives* from which all others are definable? Is there perhaps *more than one such set*, hence providing us with a choice of programming primitives?

Secondly, what is the exact kind of programming power that we have unleashed? Does it lead to interesting programs that we have not been able to write before? We have outlined some speculative applications for intensional recursion in §2.1.4. Is iPCF a useful tool when it comes to attacking these?

We discuss some more aspects of iPCF in the concluding chapter (§9.2).

Chapter 4

Categories and Intensionality[1]

We turn now to the discussion of the categorical modelling of intensionality.

We have discussed the importance of intensionality for Computer Science in §1.1. One might therefore ask why the concept has not led to many exciting developments. Abramsky [2014] suggests that it may simply be that the extensional paradigm is already sufficiently challenging:

> " [...] while Computer Science embraces wider notions of processes than computability theory, it has tended to refrain from studying intensional computation, despite its apparent expressive potential. This reluctance is probably linked to the fact that it has proved difficult enough to achieve software reliability even while remaining within the confines of the extensional paradigm. Nevertheless, it seems reasonable to suppose that understanding and harnessing intensional methods offers a challenge for computer science which it must eventually address."

But we believe that there is also a deeper reason: once we step outside extensionality, there are no rules: one might even say that 'anything goes:' this is what Abramsky means by the phrase "loose baggy monster" (quoted at the start of §1.2). A natural reaction to this state of affairs is to turn to category theory in an attempt to find some structure that can put things into perspective, or simply provide a language for studying the interplay between the extensional and the intensional.

Surprisingly, very little has been said about intensionality in the categorical domain. Lawvere [1969, 2006] observes that there are categories which are *not well-pointed*, hence—in some sense—'intensional.' But if we only allow for slightly more generality, this intensionality vanishes: there is only one notion of equality.

[1] A preliminary form of the results in this chapter was first published as [Kavvos, 2017a], which is available at Springer: `https://dx.doi.org/10.1007/978-3-662-54458-7_32`

We shall take the hint regarding the relationship between modal logic and intensionality from our discussion in §1.4. We already know that there is a deep connection between logic, type theory, and category theory, namely the *Curry-Howard-Lambek correspondence*. This makes it likely that attempting to transport the reading of modality-as-intension to the realm of category theory could lay the foundation for a basic theory of intensionality. We hence revisit the appropriate categorical semantics of type theories in the spirit of $\mathsf{S4}$, which was introduced by Bierman and de Paiva [2000]. We find that it is not appropriate for our purposes, and we argue to that effect in §4.1.

Taking all of these points into account, one can only surmise that there has to be a radical shift in our perspective: *we need to step outside classical 1-category theory*. Fortunately, this necessary groundwork has been laid by Čubrić et al. [1998] and their discovery of *P-category theory*. We introduce P-category theory in §4.1.2, and discuss its use in modelling intensionality.

All that remains is to introduce a new concept that ties intensionality and extensionality together under the same roof. This is the notion of an *exposure*, which we introduce and study in §4.2.

4.1 Categories are not intensional

At the outset, things look promising: let there be a category $\mathcal{C}$ with a terminal object $\mathbf{1}$. Arrows of type $x : \mathbf{1} \to C$ are called *points of the object* C. An arrow $f : C \to A$ introduces a map

$$x : \mathbf{1} \to C \quad \longmapsto \quad f \circ x : \mathbf{1} \to A$$

from points of C to points of A. In this setting, Lawvere [1969, 2006] observes that we may have two distinct arrows $f, g : C \to A$ that induce the same map, i.e. such that

$$\forall x : \mathbf{1} \to A.\ f \circ x = g \circ x$$

all whilst $f \neq g$. In this case, we say that the category $\mathcal{C}$ *does not have enough points*, or *is not well-pointed*.

Nevertheless, lack of enough points is not enough to have 'intensionality,' and the discussion in [Awodey, 2010, §2.3] provides the necessary intuition. Up to now, we have focused on points $x : \mathbf{1} \to C$. These can be construed as *'tests'*, in the sense that we can look at each $f \circ x$ and infer some information about the arrow $f : C \to A$, e.g. its value at some 'argument' $x : \mathbf{1} \to C$. However, we can conceive of more involved

test objects T, and significantly more comprehensive 'tests' of type $x : T \to C$ which we can call *generalised points*. Arrows $f : C \to A$ are completely distinguishable if given arbitrary generalised points. The argument is dumbfoundingly trivial: the most thorough 'test object' is C itself, and for that one there's the generalised point $id_C : C \to C$ with the unfortunate property that $f \circ id_C = f \neq g = g \circ id_C$.

We can therefore draw the conclusion that *categories cannot model intensionality*: there cannot be two distinct arrows f and g that are indistinguishable within the category. Of course, we can always quotient $\mathcal{C}$ by some compatible equivalence relation $\sim$ to obtain $\mathcal{C}/\sim$. Then the intensional universe would be $\mathcal{C}$, and its extensional version would be $\mathcal{C}/\sim$. But that would entail that we are no longer operating within a single mathematical universe, i.e. a single category!

4.1.1 Intension, Modality, and Categories

We thus return to the *modality-as-intention* interpretation of §1.4 to look for the answer. From a categorical perspective, all is well with the intuitions we have developed there and in §3, save the punchline: the categorical semantics of the $\mathsf{S4}$ box modality, due to Bierman and de Paiva [2000] and Kobayashi [1997], specifies that

$$\Box : \mathcal{C} \longrightarrow \mathcal{C}$$

is part of a *monoidal comonad* $(\Box, \epsilon, \delta)$ on a cartesian closed category $\mathcal{C}$. Let us define some of these notions for the sake of completeness.

Definition 9. A *comonad* (Q, ϵ, δ) consists of an endofunctor

$$Q : \mathcal{C} \longrightarrow \mathcal{C}$$

and two natural transformations,

$$\epsilon : Q \Rightarrow \mathsf{Id}_{\mathcal{C}}, \quad \delta : Q \Rightarrow Q^2$$

such that the following diagrams commute:

$$\begin{array}{ccc} QA & \xrightarrow{\delta_A} & Q^2A \\ {\scriptstyle \delta_A}\downarrow & & \downarrow{\scriptstyle \delta_{QA}} \\ Q^2A & \xrightarrow[Q\delta_A]{} & Q^3A \end{array} \qquad \begin{array}{ccc} QA & \xrightarrow{\delta_A} & Q^2A \\ {\scriptstyle \delta_A}\downarrow & \searrow^{id_A} & \downarrow{\scriptstyle \epsilon_{QA}} \\ Q^2A & \xrightarrow[Q\epsilon_A]{} & QA \end{array}$$

In the cartesian case, *monoidality* requires the provision of morphisms

$$m_{A,B} : QA \times QB \to Q(A \times B)$$
$$m_0 : \mathbf{1} \to Q\mathbf{1}$$

natural in each pair of objects $A, B \in \mathcal{C}$, which must also make certain diagrams commute. We will be particularly interested in the case where the functor is *strong monoidal*, i.e. $m_{A,B}$ and m_0 are natural *isomorphisms*. It has been shown in the technical report [Kavvos, 2017c] that this is the same as a *product-preserving functor*.

The transformations $\delta : Q \Rightarrow Q^2$ and $\epsilon : Q \Rightarrow \mathsf{Id}_{\mathcal{C}}$ are *monoidal* if the following diagrams commute:

$$\begin{array}{ccc} QA \times QB & \xrightarrow{\epsilon_A \times \epsilon_B} & A \times B \\ {\scriptstyle m_{A,B}}\downarrow & & \| \\ Q(A \times B) & \xrightarrow[\epsilon_{A\times B}]{} & A \times B \end{array} \qquad \begin{array}{ccc} \mathbf{1} & & \\ {\scriptstyle m_0}\downarrow & \searrow & \| \\ Q\mathbf{1} & \xrightarrow[\epsilon_{\mathbf{1}}]{} & \mathbf{1} \end{array}$$

$$\begin{array}{ccc} QA \times QB & \xrightarrow{\delta_A \times \delta_B} & Q^2A \times Q^2B \\ {\scriptstyle m_{A,B}}\downarrow & & \downarrow{\scriptstyle m_{QA,QB}} \\ Q(A \times B) & & Q(QA \times QB) \\ & & \downarrow{\scriptstyle Qm_{A,B}} \\ & \xrightarrow[\delta_{A\times B}]{} & Q^2(A \times B) \end{array} \qquad \begin{array}{ccccc} \mathbf{1} & & & & \\ {\scriptstyle m_0}\downarrow & \searrow^{m_0} & & & \\ Q\mathbf{1} & & Q\mathbf{1} & \xrightarrow{Q(m_0)} & \\ & & & & Q^2\mathbf{1} \end{array}$$

Please refer to [Mac Lane, 1978, §XI.2] or [Melliès, 2009, §5] for more details, and the missing commuting diagrams.

Now, as $\Box : \mathcal{C} \longrightarrow \mathcal{C}$ is a functor, it will unfortunately trivially satisfy

$$f = g \quad \Longrightarrow \quad \Box f = \Box g$$

and will hence necessarily validate the congruence rule for the modality:

$$\frac{\Delta\,;\cdot \vdash M = N : A}{\Delta\,;\Gamma \vdash \mathsf{box}\ M = \mathsf{box}\ N : \Box A}\ (\Box\mathsf{cong})$$

This does not conform to the 'no reductions under $\mathsf{box}\ (-)$' restriction, and hence disrupts the intensional nature of the modal types in iPCF.

As if this were not enough, we will now present another argument that provides the last straw for the monoidal comonad interpretation. Intuitively, if $\Box A$ is to represent code of type A, then there should be many more points $\mathbf{1} \to \Box A$ than points $\mathbf{1} \to A$: there is more than one expression corresponding to the same value in any interesting

logical system. Under a very mild assumption, this desideratum fails. To show that, suppose we have a monoidal comonad $(\Box, \epsilon, \delta)$. Furthermore, suppose the components of $\delta : Q \Rightarrow Q^2$ satisfy the following definition:

Definition 10. The component $\delta_A : \Box A \to \Box^2 A$ is *reasonable quoting device at* A just if the following diagram commutes:

$$\begin{array}{ccc} \mathbf{1} & \xrightarrow{a} & \Box A \\ {\scriptstyle m_0}\downarrow & & \downarrow{\scriptstyle \delta_A} \\ \Box\mathbf{1} & \xrightarrow[\Box a]{} & \Box^2 A \end{array}$$

This equation may be type-theoretically expressed in iPCF as the following equation for any $\vdash M : \Box A$:

$$\vdash \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in\ box}\ (\mathsf{box}\ u) = \mathsf{box}\ M : \Box\Box A$$

Then,

Proposition 1. *If each component* $\delta_A : \Box A \to \Box^2 A$ *of a monoidal comonad* $(\Box, \epsilon, \delta)$ *is a reasonable quoting device, then there is a natural isomorphism*

$$\mathcal{C}(\mathbf{1}, -) \cong \mathcal{C}(\mathbf{1}, \Box(-))$$

Proof. We can define maps

$$\begin{aligned} \mathsf{in} : \mathcal{C}(\mathbf{1}, A) &\to \mathcal{C}(\mathbf{1}, \Box A) \\ x &\mapsto \Box x \circ m_0 \\ \mathsf{out} : \mathcal{C}(\mathbf{1}, \Box A) &\to \mathcal{C}(\mathbf{1}, A) \\ a &\mapsto \epsilon_A \circ a \end{aligned}$$

and then calculate:

$$\mathsf{out}\,(\mathsf{in}(x)) = \epsilon_A \circ \Box x \circ m_0 = x \circ \epsilon_{\mathbf{1}} \circ m_0 = x$$

where the last equality is because $\mathbf{1}$ is a terminal object. Similarly,

$$\mathsf{in}\,(\mathsf{out}(a)) = \Box(\epsilon_A \circ a) \circ m_0 = \Box\epsilon_A \circ \Box a \circ m_0 = \Box\epsilon_A \circ \delta_A \circ a = a$$

where we have only used our 'reasonable' condition, and one of the comonad laws. Naturality of the isomorphism follows by functoriality of $\Box$ and naturality of ϵ. $\Box$

Hence, in these circumstances, there are no more codes than values!

This definition of a reasonable quoting device is slightly mysterious. In fact, it follows from the commutation of the following more general diagram, for each $f : QA \to QB$:

$$\begin{array}{ccc} QA & \xrightarrow{f} & QB \\ {\scriptstyle \delta_A}\downarrow & & \downarrow{\scriptstyle \delta_B} \\ Q^2A & \xrightarrow[Qf]{} & Q^2B \end{array}$$

Indeed, given $a : \mathbf{1} \to QA$, consider $a \circ m_0^{-1} : Q\mathbf{1} \to QA$. Commutation of the diagram them yields

$$\delta_A \circ a \circ m_0^{-1} = Q(a \circ m_0^{-1}) \circ \delta_{\mathbf{1}}$$

Taking the m_0 to the other side, we have

$$\delta_A \circ a = Qa \circ Qm_0^{-1} \circ \delta_{\mathbf{1}} \circ m_0 = Qa \circ Qm_0^{-1} \circ Qm_0 \circ m_0 = Qa \circ m_0$$

by the monoidality of δ. In turn, commutation of this diagram for any $f : A \to B$ corresponds to the comonad being *idempotent*.

Theorem 26 (Idempotence)**.** *Given a comonad (Q, ϵ, δ), the following are equivalent:*

1. *$\delta : Q \Rightarrow Q^2$ is an isomorphism.*
2. *$\delta \circ \epsilon_Q : Q^2 \Rightarrow Q^2$ is the identity natural transformation.*
3. *For all $f : QA \to QB$, $Qf \circ \delta_A = \delta_B \circ f$.*

If any one of these holds, we say that (Q, ϵ, δ) is idempotent*.*

Proof. We prove (2) ⇒ (3) ⇒ (1) ⇒ (2).

CASE(2 ⇒ 3). We have

$$\begin{array}{ll} & Qf \circ \delta_A \\ = & \{ \text{by (2)} \} \\ & \delta_B \circ \epsilon_{QB} \circ Qf \circ \delta_A \\ = & \{ \text{naturality of } \epsilon \} \\ & \delta_B \circ f \circ \epsilon_{QA} \circ \delta_A \\ = & \{ \text{comonad equation} \} \\ & \delta_B \circ f \end{array}$$

CASE($3 \Rightarrow 1$). We already know that $\epsilon_Q \circ \delta = \mathsf{Id}$ from the comonad equations, so it remains to prove that $\delta \circ \epsilon_Q$ is the identity natural transformation on Q.

$$\begin{aligned} & \delta_A \circ \epsilon_{QA} \\ = & \quad \{ \text{by (3)} \} \\ & Q\epsilon_{QA} \circ \delta_{QA} \\ = & \quad \{ \text{comonad equation} \} \\ & id_{QA} \end{aligned}$$

Hence $\delta^{-1} = \epsilon_Q$.

CASE($1 \Rightarrow 2$). We have

$$\begin{aligned} & \delta_A \circ \epsilon_{QA} \\ = & \quad \{ \text{by (1)} \} \\ & \delta_A \circ \epsilon_{QA} \circ \delta_A \circ \delta_A^{-1} \\ = & \quad \{ \text{comonad equation} \} \\ & \delta_A \circ \delta_A^{-1} \\ = & \quad \{ \delta \text{ iso} \} \\ & id_{Q^2A} \end{aligned}$$

□

The behaviour of code will often be *idempotent* in the above sense: once something is quoted code, it is in the realm of syntax, and more layers of quoting do not change its quality as sense. Thus, the above argument delivers a fatal blow to the monoidal comonad approach.

4.1.2 PERs and P-categories

The way out of this seeming impasse is to step outside 1-category theory. We shall use the notion of *P-category*, which was introduced by Čubrić et al. [1998] precisely so the authors could deal with a form of intensionality.[2]

P-category theory is essentially category theory *up to a partial equivalence relation.*

[2]For the record, the gist of [Čubrić et al., 1998] is that the Yoneda embedding on the categorical term model of typed λ-calculus is a key ingredient in *normalisation by evaluation*—with the proviso that terms are not strictly identified up to $\beta\eta$ equality!

Definition 11. A *partial equivalence relation (PER)* is a symmetric and transitive relation.

Partial equivalence relations were introduced to Theoretical Computer Science by Turing in an unpublished manuscript, and brought to the study of semantics by Girard [1972] and Scott [1975]. Broadly speaking, we can view an equivalence relation on a set as a *notion of equality* for that set. However, partial equivalence relations might not be reflexive. Elements that are not related to themselves can be understood as *not being well-defined*. We can, for example, define a PER $\sim$ between sequences of rationals as follows: $\{x_i\} \sim \{y_i\}$ just if both sequences are Cauchy and converge to the same real number. Sequences that are *not Cauchy* cannot be real numbers.

Definition 12. A *P-set* is a pair $A = (|A|, \sim_A)$ consisting of a set $|A|$ and a PER $\sim_A$ on A.

We will formally distinguish between *elements* $x \in |A|$ of the P-set A, and *points* $x \in A$ of A: for the latter we will also require that $x \sim_A x$, i.e. that they be well-defined. Given a P-set A, its *domain* $\mathrm{dom}(A)$ is the set of its points.

The notion of *operation* will be instrumental in the development of our theory. An operation is a transformation between the elements of P-sets that is not functional, in that it need not respect the PERs on P-sets.

Definition 13 (Operation)**.** Given two P-sets $A = (|A|, \sim_A)$ and $B = (|B|, \sim_B)$, an *operation*, written

$$f : A \dashrightarrow B$$

is a function $f : |A| \to |B|$ such that $x \sim_A x$ implies $f(x) \sim_B f(x)$.

I.e. an operation takes elements to elements, but when given a point (well-defined element) also returns a point. Some operations are more well-behaved:

Definition 14. An operation $f : A \dashrightarrow B$ is a *P-function*, written

$$f : A \to B$$

just if $x \sim_A y$ implies $f(x) \sim_B f(y)$.

We will later see that P-sets and P-functions form a cartesian closed P-category (Theorem 27). The main ingredients needed for that theorem are the subject of the first three of the following examples.

Example 1.

1. The P-set $\mathbf{1}$ is defined to be $(\{*\}, \{(*,*)\})$.

2. Given P-sets $A = (|A|, \sim_A)$ and $B = (|B|, \sim_B)$, we define the P-set $A \times B$ by $|A \times B| \stackrel{\text{def}}{=} |A| \times |B|$ and $(a,b) \sim_{A\times B} (a',b')$ just if $a \sim_A a'$ and $b \sim_B b'$.

3. Given P-sets $A = (|A|, \sim_A)$ and $B = (|B|, \sim_B)$, the P-functions $f : A \to B$ form a P-set $B^A = (\left|B^A\right|, \sim_{B^A})$ as follows: $\left|B^A\right| \stackrel{\text{def}}{=} \{\, f \mid f : A \dashrightarrow B \,\}$, and $f \sim_{B^A} g$ just if $a \sim_A a'$ implies $f(a) \sim_B g(a')$.

4. Given P-sets $A = (|A|, \sim_A)$ and $B = (|B|, \sim_B)$, the P-operations $f : A \dashrightarrow B$ form a P-set $A \succ B = (\left|B^A\right|, \sim_{A\succ B})$ with the same underlying set $\left|B^A\right|$, but with $f \sim_{A\succ B} g$ just if $f = g$.

In the definition of the P-set of P-functions, it is evident that all operations $f : A \dashrightarrow B$ are present as elements in $\left|B^A\right|$. However, they are only in $\mathrm{dom}(B^A)$ if they are P-functions. This pattern of 'junk' being present amongst elements is characteristic when it comes to PERs, and it is the reason we need the notion of points. A *P-point* of the P-set A would be a P-function $x : \mathbf{1} \to A$, which is determined by the point $x(*) \in \mathrm{dom}(A)$. We will systematically confuse a P-point $x : \mathbf{1} \to A$ with the point $x(*) \in \mathrm{dom}(A)$.

We can finally define what it means to be a

Definition 15 (P-category)**.** A *P-category* $(\mathfrak{C}, \sim)$ consists of:

- a set of objects $ob(\mathfrak{C})$;
- for any two objects $A, B \in ob(\mathfrak{C})$, a P-set $\mathfrak{C}(A,B) = \left(|\mathfrak{C}(A,B)|, \sim_{\mathfrak{C}(A,B)}\right)$;
- for each object $A \in ob(\mathfrak{C})$, a point $id_A \in \mathfrak{C}(A,A)$;
- for any three objects $A, B, C \in ob(\mathfrak{C})$, a P-function

$$c_{A,B,C} : \mathfrak{C}(A,B) \times \mathfrak{C}(B,C) \to \mathfrak{C}(A,C)$$

  for which we write $g \circ f \stackrel{\text{def}}{=} c_{A,B,C}(f,g)$,

such that, for any point $f \in \mathfrak{C}(A,B)$ we have

$$f \circ id_A \sim_{\mathfrak{C}(A,B)} f$$
$$id_B \circ f \sim_{\mathfrak{C}(A,B)} f$$

and for any $f \in \mathfrak{C}(A,B)$, $g \in \mathfrak{C}(B,C)$ and $h \in \mathfrak{C}(C,D)$, we have

$$h \circ (g \circ f) \sim_{\mathfrak{C}(A,D)} (h \circ g) \circ f$$

In a P-category we have an ordinary set of objects, but only a P-set of morphisms. We will say that f is *an arrow of* $\mathfrak{C}$ *with domain* A *and codomain* B, and write $f : A \to B$, only whenever f is a well-defined morphism, i.e. $f \in \mathrm{dom}(\mathfrak{C}(A, B))$.

Furthermore, we will variously refer to P-categories by Fraktur letters $\mathfrak{B}, \mathfrak{C}, \ldots$, without mentioning the family of relations $\{\sim_{\mathfrak{C}(A,B)}\}_{A,B \in \mathfrak{C}}$. Sometimes we might use the same sets of morphisms $|\mathfrak{C}(A, B)|$, but with a different PER; we will then indicate which PER we are using, by writing e.g. $(\mathfrak{C}, \sim)$ or $(\mathfrak{C}, \doteq)$. When the types of two morphisms f and g are evident, we will write $f \sim g$ without further ado.

We can think of the morphisms as intensional, and of the PER on them as describing extensional equality. That composition is a P-function encodes the requirement that *composition respects extensional equality*: if $f \sim f'$ and $g \sim g'$, then $g \circ f \sim g' \circ f'$.

As is expected, P-categories come with associated notions of functor and natural transformation.

Definition 16 (P-functor)**.** Let $\mathfrak{C}, \mathfrak{D}$ be P-categories. A *P-functor* $F : \mathfrak{C} \longrightarrow \mathfrak{D}$ from $\mathfrak{C}$ to $\mathfrak{D}$ consists of a map assigning an object $FX \in \mathfrak{D}$ for each object $X \in \mathfrak{C}$, and a family of P-functions,

$$F_{A,B} : \mathfrak{C}(A, B) \to \mathfrak{D}(FA, FB)$$

for each pair of objects $A, B \in \mathfrak{C}$, such that

$$F(id_A) \sim id_{FA}$$
$$F(g \circ f) \sim Fg \circ Ff$$

for all pairs of arrows $f : A \to B$ and $g : B \to C$.

Definition 17 (P-natural transformation)**.** Let $F, G : \mathfrak{C} \longrightarrow \mathfrak{D}$ be P-functors. A *P-natural transformation* $\theta : F \Rightarrow G$ consists of an arrow $\theta_A : FA \to GA$ in $\mathfrak{D}$ for each $A \in \mathfrak{C}$, such that for each $f : A \to B$ we have

$$\theta_B \circ Ff \sim Gf \circ \theta_A$$

Finally, the definitions of finite products and exponentials carry over smoothly. The various components of the definitions are required to be unique with respect to their universal property, but only up to the PERs.

Definition 18 (Terminal object)**.** Let $\mathfrak{C}$ be a P-category. An object $\mathbf{1} \in \mathfrak{C}$ is *terminal* just if for every $C \in \mathfrak{C}$ there is an arrow

$$!_C : C \to \mathbf{1}$$

such that $!_C \sim !_C$, and for every arrow $h : C \to \mathbf{1}$ we have $h \sim !_C$.

Definition 19 (Binary Product)**.** Let $\mathfrak{C}$ be a P-category, and let $A, B \in \mathfrak{C}$. The object $A \times B \in \mathfrak{C}$ is a *product* of A and B if there are arrows

$$A \xleftarrow{\pi_1} A \times B \xrightarrow{\pi_2} B$$

and for any object $C \in \mathfrak{C}$, there is a P-function

$$\langle \cdot, \cdot \rangle : \mathfrak{C}(C, A) \times \mathfrak{C}(C, B) \to \mathfrak{C}(C, A \times B)$$

such that for every $f : C \to A$ and $g : C \to B$ we have

$$\pi_1 \circ \langle f, g \rangle \sim f$$
$$\pi_2 \circ \langle f, g \rangle \sim g$$

and, for any $h : C \to A \times B$, we have $\langle \pi_1 \circ h, \pi_2 \circ h \rangle \sim h$.

Of course, the usual calculational rules of products still hold, e.g.

$$\langle f, g \rangle \circ h \sim \langle f \circ h, g \circ h \rangle$$
$$(f \times g) \circ \langle h, k \rangle \sim \langle f \circ h, g \circ k \rangle$$

and so on.

Definition 20. A *cartesian P-category* is a P-category that has a terminal object and binary products.

Definition 21 (Exponential)**.** Let $\mathfrak{C}$ be a cartesian P-category, and let $A, B \in \mathfrak{C}$. The object $B^A \in \mathfrak{C}$ is an *exponential* of A and B just if there is an arrow

$$\mathsf{ev}_{A,B} : B^A \times A \to B$$

such that, for each $C \in \mathfrak{C}$, there is a P-function

$$\lambda_C(-) : \mathfrak{C}(C \times A, B) \to \mathfrak{C}(C, B^A)$$

such that, for any $h : C \times A \to B$ and $k : C \to B^A$,

$$\mathsf{ev}_{A,B} \circ (\lambda_C(h) \times id_A) \sim h$$
$$\lambda_C \left(\mathsf{ev}_{A,B} \circ (k \times id_A) \right) \sim k$$

Definition 22 (P-ccc)**.** A *cartesian closed P-category*, or *P-ccc*, is a P-category that has a terminal object, binary products, and an exponential for each pair of objects.

Theorem 27 (Čubrić et al. [1998])**.** *The P-category $\mathfrak{PSet}$ of P-sets and P-functions is a P-ccc.*

We warn the reader that we might be lax with the prefix "P-" in the rest of this thesis, as the overwhelming majority of it will solely concern P-categories.

4.2 Exposures

In this section we introduce a new P-categorical notion, that of an *exposure*. The aim of exposures is to P-categorically capture the modality-as-intension interpretation.

An exposure is *almost a functor*: it is a map of objects and arrows of a P-category into another, and it preserves identities and composition. It is *not* a functor because it does not preserve the PERs on the hom-sets of the source P-category. Instead, it *reflects* them. In that sense, it may *expose* the structure of a particular arrow by uncovering its inner workings, irrespective of the extensional equality represented by the PER. The inner workings are then represented as a well-defined arrow of some, possibly the same, P-category.

Definition 23. An exposure $Q : (\mathfrak{B}, \sim) \looparrowright (\mathfrak{C}, \sim)$ consists of

(i) a map assigning to each object $A \in \mathfrak{B}$ an object $QA \in \mathfrak{C}$, and

(ii) for each $A, B \in \mathfrak{C}$, an operation

$$Q_{A,B} : \mathfrak{B}(A, B) \dashrightarrow \mathfrak{C}(QA, QB)$$

for which we simply write Qf when the source and target of f are known

such that

(i) $Q(id_A) \sim id_{QA}$;

(ii) $Q(g \circ f) \sim Qg \circ Qf$, for any arrows $f : A \to B$ and $g : B \to C$, and

(iii) $Q_{A,B}$ reflects PERs: if $Qf \sim Qg$ then $f \sim g$.

Like functors, exposures compose: it suffices to use reflection of PERs twice. There is a close relationship between functors and exposures. In fact, if exposures were functors, they would be faithful functors.

Lemma 10. *A (P-)functor $Q : (\mathfrak{B}, \sim) \longrightarrow (\mathfrak{C}, \sim)$ is an exposure if and only if it is (P-)faithful.*

Proof. If $Q : (\mathfrak{B}, \sim) \longrightarrow (\mathfrak{C}, \sim)$ is also an exposure, then the morphism map $Q_{A,B} : \mathfrak{B}(A, B) \to \mathfrak{C}(QA, QB)$ is a P-function which also reflects PERs, hence Q is a (P-)faithful (P-)functor. The converse is similar. □

Thus, the identity functor is an exposure $\mathsf{Id}_{\mathfrak{B}} : (\mathfrak{B}, \sim) \nrightarrow (\mathfrak{B}, \sim)$. This lemma also enables us to compose an exposure $Q : (\mathfrak{B}, \sim) \nrightarrow (\mathfrak{C}, \sim)$ with a faithful functor in either direction: if $F : (\mathfrak{A}, \sim) \longrightarrow (\mathfrak{B}, \sim)$ is faithful then we can define the exposure $Q \circ F : (\mathfrak{A}, \sim) \nrightarrow (\mathfrak{C}, \sim)$ by

$$\begin{aligned}
&(Q \circ F)(A) \stackrel{\text{def}}{=} Q(FA) \\
&(Q \circ F)_{A,B} : \mathfrak{A}(A, B) \dashrightarrow \mathfrak{C}(Q(FA), Q(FB)) \\
&(Q \circ F)_{A,B} \stackrel{\text{def}}{=} f \mapsto Q_{A,B}(F_{A,B}(f))
\end{aligned}$$

and similarly for pre-composition.

The notion of natural transformations also naturally carries over to the setting of exposures:

Definition 24. A *natural transformation of exposures* $t : F \overset{\bullet}{\nrightarrow} G$ between two exposures $F, G : \mathfrak{B} \nrightarrow \mathfrak{C}$ consists of an arrow $t_A : FA \to GA$ of $\mathfrak{C}$ for each object $A \in \mathfrak{B}$, such that, for every arrow $f : A \to B$ of $\mathfrak{B}$, the following diagram commutes up to $\sim$:

$$\begin{array}{ccc}
FA & \xrightarrow{Ff} & FB \\
{\scriptstyle t_A}\downarrow & & \downarrow{\scriptstyle t_B} \\
GA & \xrightarrow[Gf]{} & GB
\end{array}$$

Nevertheless, we must not be cavalier when adopting practices from 1-category theory. For example, we cannot arbitrarily compose natural transformations of exposures with other exposures. Let $t : F \overset{\bullet}{\nrightarrow} G$ be a natural transformation between two exposures $F, G : \mathfrak{B} \nrightarrow \mathfrak{C}$. If $R : \mathfrak{A} \nrightarrow \mathfrak{B}$ is an exposure, then we can define

$$(t_R)_A \stackrel{\text{def}}{=} t_{RA} : FRA \to GRA$$

These components form a natural transformation $t_R : FR \overset{\bullet}{\nrightarrow} GR$. The defining diagram

$$\begin{array}{ccc}
FRA & \xrightarrow{FRf} & FRB \\
{\scriptstyle t_{RA}}\downarrow & & \downarrow{\scriptstyle t_{RB}} \\
GRA & \xrightarrow[GRf]{} & GRB
\end{array}$$

commutes, as it is the naturality square of $t : F \overset{\bullet}{\nrightarrow} G$ at $Rf : RA \to RB$. But if we instead have a P-functor $L : \mathfrak{C} \longrightarrow \mathfrak{D}$, we can define

$$\begin{aligned}
&Lt : LF \overset{\bullet}{\nrightarrow} LG \\
&(Lt)_A \stackrel{\text{def}}{=} L(t_A) : LFA \to LGA
\end{aligned}$$

We must have that L be a P-functor for this to be natural: the diagram we want is commutative only if

$$L(t_B) \circ LFf \sim L(t_B \circ Ff) \sim L(Gf \circ t_A) \sim LGf \circ L(t_A)$$

Whereas the first and last step would hold if L were merely an exposure, the middle step requires that we can reason equationally 'under L,' which can only happen if L preserves the PERs.

4.2.1 Intensional Equality

As exposures give a handle on the internal structure of arrows, they can be used to define intensional equality: if the images of two arrows under the same exposure Q are extensionally equal, then the arrows have the same implementation, so they are intensionally equal. This is an exact interpretation of the slogan of Abramsky [2014]: *intensions become extensions.*

Definition 25 (Intensional Equality)**.** Let $Q : (\mathfrak{B}, \sim) \nrightarrow (\mathfrak{C}, \sim)$ be an exposure. Two arrows $f, g : A \to B$ of $\mathfrak{B}$ are *intensionally equal (up to Q)*, written

$$f \approx_Q^{A,B} g$$

just if $Qf \sim Qg$.

We often drop the source and target superscripts, and merely write $f \approx_Q g$. The fact that exposures *reflect* PERs guarantees that

Lemma 11. *Intensional equality implies extensional equality.*

Proof. $f \approx_Q g : A \to B$ is $Qf \sim Qg$, which implies $f \sim g$. □

In some cases the converse is true, but not for general arrows $A \to B$: we often need some restrictions on A, perhaps that it is an *intensional context* i.e. of the form $\prod_{i=1}^{n} QA_i$, or that it is simply a point. In that case, we use the following definition.[3]

Definition 26. Let $Q : (\mathfrak{B}, \sim) \nrightarrow (\mathfrak{C}, \sim)$ be an exposure.

[3]Not to be confused with Voevodsky's univalence axiom. When examining this thesis, Martin Hyland strongly recommended that the name be changed. This will most likely happen before subsequent publications.

1. Let $\mathcal{A}$ be a class of objects of $\mathfrak{B}$. An object $U \in \mathfrak{B}$ is *$\mathcal{A}$-univalent (up to Q)* just if extensional equality implies intensional equality for arrows with domain in $\mathcal{A}$ and codomain U, i.e. for every $f, g : A \to U$ with $A \in \mathcal{A}$,

$$f \sim g \quad \Longrightarrow \quad f \approx_Q g$$

2. If $\mathfrak{B}$ has a terminal object $\mathbf{1}$, and U is $\{\mathbf{1}\}$-univalent, we say that U is *point-univalent*.

Intensional equality is a PER, as we have defined it through Q and extensional equality, which is a PER itself. Replacing $\sim$ with $\approx_Q$ yields another P-category, which is possibly 'more intensional' than $(\mathfrak{B}, \sim)$.

Definition 27. Given an exposure $Q : (\mathfrak{B}, \sim) \nrightarrow (\mathfrak{C}, \sim)$, we define the *x-ray category of* $(\mathfrak{B}, \sim)$ *up to* Q by replacing the hom-P-sets with

$$\left(\left|\mathfrak{B}(A, B)\right|, \approx_Q^{A,B}\right)$$

We denote the x-ray category by $(\mathfrak{B}, \approx_Q)$.

To show that this is a valid definition, we have to check that composition respects intensional equality, and that the necessary axioms hold. If $f \approx_Q k$ and $g \approx_Q h$, then

$$Q(g \circ f) \sim Qg \circ Qf \sim Qh \circ Qk \sim Q(h \circ k)$$

because exposures preserve composition; hence $\circ$ is indeed a well-defined P-function

$$\circ : (\mathfrak{B}, \approx_Q)(A, B) \times (\mathfrak{B}, \approx_Q)(B, C) \to (\mathfrak{B}, \approx_Q)(A, C)$$

Similarly, the PER $\approx_Q$ hereditarily satisfies associativity of composition, and—as Q preserves identities—also satisfies the identity laws.

4.2.2 Cartesian and Product-Preserving Exposures

Bare exposures offer no promises or guarantees regarding intensional equality. For example, it is not a given that $\pi_1 \circ \langle f, g \rangle \approx_Q f$. However, from a certain viewpoint one may argue there is no grand intensional content in projecting a component: it is merely a structural operation and not much more. Requiring this of an exposure strengthens the notion of intensional equality, and further reinforces the point that exposures offer a stratified and modular view of equality.

Definition 28. Let $\mathfrak{B}$ be a cartesian P-category, and let $Q : \mathfrak{B} \nrightarrow \mathfrak{C}$ be an exposure. We say that Q is *cartesian* just if for any arrows $f : C \to A$, $g : C \to B$, $h : C \to A \times B$, and $k : D \to \mathbf{1}$ we have

$$\begin{aligned} \pi_1 \circ \langle f, g \rangle &\approx_Q f \\ \pi_2 \circ \langle f, g \rangle &\approx_Q g \\ \langle \pi_1 \circ h, \pi_2 \circ h \rangle &\approx_Q h \\ k &\approx_Q !_D \end{aligned}$$

However, this is not enough to formally regain standard equations like $\langle f, g \rangle \circ h \approx_Q \langle f \circ h, g \circ h \rangle$. This is because we cannot be certain that the pairing function $\langle \cdot, \cdot \rangle$ preserves the intensional equality $\approx_Q$. We need something quite a bit stronger, which is to ask for full extensional preservation of products.

Definition 29. A cartesian exposure $Q : \mathfrak{B} \nrightarrow \mathfrak{C}$ of a cartesian P-category $\mathfrak{B}$ in a cartesian P-category $\mathfrak{C}$ is *product-preserving* whenever the canonical arrows

$$\begin{aligned} \langle Q\pi_1, Q\pi_2 \rangle &: Q(A \times B) \xrightarrow{\cong} QA \times QB \\ !_{Q\mathbf{1}} &: Q\mathbf{1} \xrightarrow{\cong} \mathbf{1} \end{aligned}$$

are (P-)isomorphisms. We write $m_{A,B} : QA \times QB \to Q(A \times B)$ and $m_0 : \mathbf{1} \to Q\mathbf{1}$ for their inverses.

In essence, the isomorphism $Q(A \times B) \xrightarrow{\cong} QA \times QB$ says that *code for pairs is a pair of codes*, and vice versa. Amongst the exposures, the product-preserving are the only ones that interact harmoniously with the product structure. In fact, preservation of products forces the pairing function to preserve intensional quality, thus regaining all the standard equations pertaining to products up to $\approx_Q$. To show all that, we first need the following proposition.

Proposition 2. *In the above setting, the following diagram commutes up to $\sim$:*

$$\begin{array}{ccc} QC & \xrightarrow{\langle Qf, Qg \rangle} & QA \times QB \\ & \searrow^{Q\langle f, g \rangle} & \downarrow{\scriptstyle m_{A,B}} \\ & & Q(A \times B) \end{array}$$

i.e. $m_{A,B} \circ \langle Qf, Qg \rangle \sim Q\langle f, g \rangle$ for $f : C \to A$ and $g : C \to B$.

Proof. We compute

$$\begin{aligned}
& \langle Q\pi_1, Q\pi_2\rangle \circ Q\langle f, g\rangle \\
\sim\ & \{\,\text{naturality}\,\} \\
& \langle Q\pi_1 \circ Q\langle f, g\rangle, Q\pi_2 \circ Q\langle f, g\rangle\rangle \\
\sim\ & \{\,Q \text{ is an exposure}\,\} \\
& \langle Q(\pi_1 \circ \langle f, g\rangle), Q(\pi_2 \circ \langle f, g\rangle)\rangle \\
\sim\ & \{\,Q \text{ is a cartesian exposure}\,\} \\
& \langle Qf, Qg\rangle
\end{aligned}$$

and hence $m_{A,B} \circ \langle Qf, Qg\rangle \stackrel{\text{def}}{=} \langle Q\pi_1, Q\pi_2\rangle^{-1} \circ \langle Qf, Qg\rangle \sim Q\langle f, g\rangle$. □

One can easily easy to compute that the $m_{A,B}$'s satisfy a naturality property, similar to the one for strong monoidal categories. That is,

Proposition 3. *In the above setting, the following diagram commutes up to* $\sim$*:*

$$\begin{array}{ccc}
QC \times QD & \xrightarrow{Qf \times Qg} & QA \times QB \\
{\scriptstyle m_{C,D}}\downarrow & & \downarrow{\scriptstyle m_{A,B}} \\
Q(C \times D) & \xrightarrow[Q(f \times g)]{} & Q(A \times B)
\end{array}$$

i.e. $m_{A,B} \circ (Qf \times Qg) \sim Q(f \times g) \circ m_{C,D}$ *for* $f : C \to A$ *and* $g : D \to B$.

Proof. Simple calculation as above, using the inverses of both $m_{A,B}$ and $m_{C,D}$, as well as the fact that $Q : \mathfrak{B} \nrightarrow \mathfrak{C}$ is cartesian. □

In monoidal 1-category theory this would simply be a natural isomorphism between the functors $Q(- \times -)$ and $Q(-) \times Q(-)$. However, the product functor $- \times -$ is not necessarily an exposure: we may have $f \times g \sim h \times k$, yet it may be that $f \not\sim h$ or $g \not\sim k$. However, if the category is *connected* (all hom-P-sets are nonempty), then the projections are epic, and hence $- \times -$ is *faithful*. By Lemma 10, this would then allow us to compose it with Q to make an exposure. But since this is not the case in general, we do not. However, the requisite naturality property follows from the fact the m's are the inverses of canonical arrows, so we are not seriously hampered.

We can also show that the following relationship holds between the projection arrows and their 'exposed' version, given product-preservation:

Proposition 4. *In the above setting, let*

$$A \xleftarrow{\pi_1^{A,B}} A \times B \xrightarrow{\pi_2^{A,B}} B$$

and

$$QA \xleftarrow{\pi_1^{QA,QB}} QA \times QB \xrightarrow{\pi_2^{QA,QB}} QB$$

be product diagrams in $\mathfrak{B}$ and $\mathfrak{C}$ respectively. Then

$$Q\pi_i^{A,B} \circ m_{A,B} \sim \pi_i^{QA,QB}$$

Proof. $\pi_i \circ m_{A,B}^{-1} \stackrel{\text{def}}{=} \pi_i \circ \langle Q\pi_1, Q\pi_2 \rangle \sim Q\pi_i$ □

The product-preserving structure of $Q : \mathfrak{B} \nrightarrow \mathfrak{C}$ then suffices to guarantee that taking the mediating morphism $\langle f, g \rangle$ preserves not just extensional equality in f and g, as it does by the definition of products in P-categories, but also intensional equality. Hence, $\langle \cdot, \cdot \rangle$ also induces products in the x-ray category $(\mathfrak{B}, \approx_Q)$.

Proposition 5. *If $Q : \mathfrak{B} \nrightarrow \mathfrak{C}$ is a product-preserving exposure, then the function*

$$\langle \cdot, \cdot \rangle : \mathfrak{B}(C, A) \times \mathfrak{B}(C, B) \to \mathfrak{B}(C, A \times B)$$

preserves intensional equality $\approx_Q$, and is thus a function

$$\langle \cdot, \cdot \rangle : (\mathfrak{B}, \approx_Q)(C, A) \times (\mathfrak{B}, \approx_Q)(C, B) \to (\mathfrak{B}, \approx_Q)(C, A \times B)$$

Proof. If $f \approx_Q h : C \to A$ and $g \approx_Q k : C \to B$, then

$$\begin{array}{ll} & Q\langle f, g \rangle \\ \sim & \{ \text{ Proposition 2 } \} \\ & m \circ \langle Qf, Qg \rangle \\ \sim & \{ \text{ assumptions } \} \\ & m \circ \langle Qh, Qk \rangle \\ \sim & \{ \text{ Proposition 2 } \} \\ & Q\langle h, k \rangle \end{array}$$

and hence $\langle f, g \rangle \approx_Q \langle h, k \rangle$. □

Note that in the above proof we used the 'monoidality' $m_{A,B}$ to 'shift' the exposure Q exactly where we want it to be to use the assumptions $f \approx_Q h$ and $g \approx_Q k$.

This result also implies that the standard equations for products hold up to $\approx_Q$.[4]

Lemma 12.

1. *If $Q : \mathfrak{B} \nrightarrow \mathfrak{C}$ is a product-preserving exposure, then*

$$\langle f, g\rangle \circ h \approx_Q \langle f \circ h, g \circ h\rangle$$

for $f : C \to A$, $g : C \to B$, and $h : D \to C$.

2. *If Q is a product-preserving exposure, then*

$$(f \times g) \circ \langle h, k\rangle \approx_Q \langle f \circ h, g \circ k\rangle$$

for $f : C \to A$, $g : D \to B$, $h : E \to C$, $k : F \to D$.

4.2.3 Comonadic Exposures

We can now revisit the failed categorical approach to modality-as-intension that we discussed in §4.1.1. It turns out that all the categorical equipment used for strong monoidal (= product-preserving) comonads have direct analogues in exposures. Throughout the rest of this section we fix a product-preserving endoexposure $Q : \mathfrak{B} \nrightarrow \mathfrak{B}$.

If we have an interpreter that maps code to values at each type, then we can present it as a (well-behaved) natural transformation from our selected exposure to the identity exposure.

Definition 30. An *evaluator* is a transformation of exposures,

$$\epsilon : Q \overset{\bullet}{\nrightarrow} \mathsf{Id}_{\mathfrak{B}}$$

such that the following diagrams commute up to $\sim$:

$$\begin{array}{ccc} QA \times QB & \xrightarrow{\epsilon_A \times \epsilon_B} & A \times B \\ {\scriptstyle m_{A,B}}\downarrow & & \| \\ Q(A \times B) & \xrightarrow[\epsilon_{A\times B}]{} & A \times B \end{array} \qquad \begin{array}{ccc} \mathbf{1} & & \\ {\scriptstyle m_0}\downarrow & \searrow & \| \\ Q\mathbf{1} & \xrightarrow[\epsilon_{\mathbf{1}}]{} & \mathbf{1} \end{array}$$

[4]In fact, even if we exclude $\langle \pi_1 \circ h, \pi_2 \circ h\rangle \approx_Q h$ from the definition of a cartesian exposure, we can then regain it through product-preservation: in this sense, product-preservation is a strong extensionality principle that even implies the 'η-rule' for the x-ray category. One might even entertain the idea that they can show the 'β-rule' $\pi_1 \circ \langle f, g\rangle \approx_Q f$ simply by the existence of the isomorphism $m_{A,B}$, and without explicitly assuming Q to be cartesian, thus ostensibly reducing cartesian exposures to product-preserving ones. But the derivation of this requires $Q\pi_1 \circ m \sim \pi_1$, which seems to only follow if the exposure is cartesian, making the apparently simple argument circular.

How about quoting, then? Given a point $a : \mathbf{1} \to A$, we define its *quote* to be the point

$$Qa \circ m_0 : \mathbf{1} \to QA$$

The fact that $\epsilon : Q \overset{\bullet}{\to} \mathsf{Id}_{\mathfrak{B}}$ lets us then calculate that

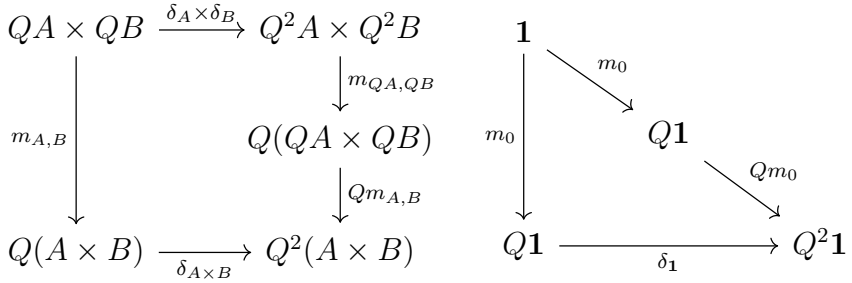
$$\epsilon_A \circ Qa \circ m_0 \sim a \circ \epsilon_{\mathbf{1}} \circ m_0 \sim a$$

So, post-composing a component of an evaluator to a quoted point yields back the point itself! The naturality is there to guarantee that the evaluator is defined 'in the same way' at all objects. The two diagrams that are required to commute as part of the definition would—in the context of monoidal functors—ensure that ϵ is a *monoidal* natural transformation. In the setting of exposures they are not only necessary in the final step of the above calculation, but they also ensure that the evaluators are compatible with products.[5]

Definition 31. A *quoter* is a transformation of exposures,

$$\delta : Q \overset{\bullet}{\to} Q^2$$

for which the following diagrams commute up to $\sim$:



If we post-compose a component of a quoter to a quoted point, we get

$$\delta_A \circ Qa \circ m_0 \sim Q^2a \circ \delta_{\mathbf{1}} \circ m_0 \sim Q^2a \circ Qm_0 \circ m_0 \sim Q(Qa \circ m_0) \circ m_0$$

So a quoter maps a quoted point to its doubly quoted version. In this instance, the diagram that would correspond to the transformation being monoidal is crucial in obtaining this pattern.

All of these ingredients then combine to form a comonadic exposure.

[5]Nevertheless, notice that—since we are in a cartesian setting and $\mathbf{1}$ is a terminal object—the second diagram commutes automatically.

Definition 32. A *comonadic exposure* (Q, ϵ, δ) consists of an endoexposure

$$Q : (\mathfrak{B}, \sim) \nrightarrow (\mathfrak{B}, \sim),$$

along with an evaluator $\epsilon : Q \overset{\bullet}{\nrightarrow} \mathsf{Id}_{\mathfrak{B}}$, and a quoter $\delta : Q \overset{\bullet}{\nrightarrow} Q^2$, such that the following diagrams commute up to $\sim$:

$$\begin{array}{ccc} QA & \xrightarrow{\delta_A} & Q^2A \\ {\scriptstyle \delta_A}\downarrow & & \downarrow{\scriptstyle \delta_{QA}} \\ Q^2A & \xrightarrow[Q\delta_A]{} & Q^3A \end{array} \qquad \begin{array}{ccc} QA & \xrightarrow{\delta_A} & Q^2A \\ {\scriptstyle \delta_A}\downarrow & \searrow^{id_{QA}} & \downarrow{\scriptstyle \epsilon_{QA}} \\ Q^2A & \xrightarrow[Q\epsilon_A]{} & QA \end{array}$$

Comonadic exposures are the analogue of product-preserving (= strong monoidal) comonads in the categorical semantics of $\mathsf{S4}$. They will prove instrumental in our analysis of intensional recursion (§6), and in the semantics of iPCF (§7, §8).

4.2.4 Idempotent Comonadic Exposures

If the components of $\delta : Q \overset{\bullet}{\nrightarrow} Q^2$ are isomorphisms, we shall call the comonadic exposure (Q, ϵ, δ) *idempotent*.

As we discussed before, if one is to take the interpretation of Q as 'code' seriously, then there are clearly two 'regions' of data: that of *static code*, always found under an occurrence of Q, and that of *dynamic data*. Intuitively, the notion of 'code of code of A,' namely $Q(QA)$, should be the same as 'code of A.' If something is code, it is already *intensional* in a maximal sense: it can certainly be taken 'one level up' $(Q(QA))$, but that should not amount to very much.

We have seen that exposures are a very weak setting in calculational terms, as they do not preserve equality. It is for this reason that we are forced to externally impose equations, such as those for cartesian products in §4.2.2. However, we will shortly see that the idempotence of a comonadic exposure is a particularly powerful tool that immediately allows us to infer a lot about equality, especially intensional.

It follows, as in §4.1.1, that following diagram commutes for each $f : QA \to QB$:

$$\begin{array}{ccc} QA & \xrightarrow{f} & QB \\ {\scriptstyle \delta_A}\downarrow & & \downarrow{\scriptstyle \delta_B} \\ Q^2A & \xrightarrow[Qf]{} & Q^2B \end{array}$$

The proof is the same as before: nowhere in Theorem 26 did we use the 'forbidden principle'

$$f = g \quad \Longrightarrow Qf = Qg$$

Furthermore, the argument we produced just before that theorem also still holds: each component of $\delta : Q \overset{\bullet}{\nrightarrow} Q^2$ is a 'reasonable quoting device,' in the sense that the diagram

$$\begin{array}{ccc} \mathbf{1} & \xrightarrow{a} & QA \\ {\scriptstyle m_0}\downarrow & & \downarrow{\scriptstyle \delta_A} \\ Q\mathbf{1} & \xrightarrow[Qa]{} & Q^2A \end{array}$$

commutes: the only time we used the 'forbidden principle,' namely the demonstration that $Qm_0 \circ Q(m_0^{-1}) \sim id_{Q\mathbf{1}}$, can be 'simulated' with idempotence as follows: we have

$$Qm_0 \circ Q(m_0^{-1}) \circ \delta_{\mathbf{1}} \sim \delta_{\mathbf{1}} \circ m_0 \circ m_0^{-1} \sim \delta_{\mathbf{1}} \circ id_{Q\mathbf{1}} \sim Q(id_{Q\mathbf{1}}) \circ \delta_{\mathbf{1}}$$

by using idempotence twice, so that cancelling the iso $\delta_{Q\mathbf{1}}$ then yields the result. We will later see that this is due to a more general theorem.

So, even if this diagram commutes, where does the 'degeneracy' argument (Proposition 1) break down? The key lies precisely in the fact that Q does not respect the PERs, and hence $\mathsf{in} : \mathfrak{C}(\mathbf{1}, A) \dashrightarrow \mathfrak{C}(\mathbf{1}, QA)$ is now only an operation, not a P-function. Hence, there is no natural isomorphism $\mathfrak{C}(\mathbf{1}, -) \cong \mathfrak{C}(\mathbf{1}, Q-)$.

Let us, however, take a closer look: the function in is defined by

$$x : \mathbf{1} \to A \quad \longmapsto \quad Qf \circ m_0 : \mathbf{1} \to QA$$

That is, the only occurrence of f is under Q, and the rest is simply pre-composition with m_0. If $f \approx_Q f'$, i.e. if $Qf \sim Qf'$, then we have that $\mathsf{in}(f) \sim \mathsf{in}(f')$. Hence, in changes $\approx_Q$ to $\sim$, so it is actually more than an operation: it is a map

$$\mathsf{in} : (\mathfrak{C}, \approx_Q)(\mathbf{1}, A) \to (\mathfrak{C}, \sim)(\mathbf{1}, QA)$$

Similarly, out is defined by

$$a : \mathbf{1} \to QA \quad \longmapsto \quad \epsilon_A \circ a : \mathbf{1} \to A$$

If $a \sim a'$, then $\delta_A \circ a \sim \delta_A \circ a'$, so $Qa \circ m_0 \sim Qa' \circ m_0$. Cancelling the isomorphism m_0 gives us $a \approx_Q a'$. Thus, if we have a reasonable quoting device at A, QA is point-univalent. For the time being, notice that this means that out takes $\sim$ to $\approx_Q$, i.e. it is a map

$$\mathsf{out} : (\mathfrak{C}, \sim)(\mathbf{1}, QA) \to (\mathfrak{C}, \approx_Q)(\mathbf{1}, A)$$

If we combine these facts with the previous calculations and naturality of Q, we obtain a natural isomorphism

Proposition 6. $(\mathfrak{C}, \approx_Q)(\mathbf{1}, -) \cong (\mathfrak{C}, \sim)(\mathbf{1}, Q-)$

That is: the intensional structure of the points, now visible in the x-ray category $(\mathfrak{C}, \approx_Q)$, is represented by the points $\mathbf{1} \to QA$ under extensional equality.

In the rest of this section let us fix a product-preserving idempotent comonadic exposure (Q, ϵ, δ) on $\mathfrak{B}$.

Equal Intensional Transformations are Intensionally Equal

A central result is the generalisation of the argument we used to define `out`, and it is the following. Think of arrows $f : QA \to QB$ as *intensional operations*: these transform code of type A to code of type B. It therefore should transpire that, if $f \sim g : QA \to QB$, then f and g represent the same code transformation, and in fact should be intensionally equal. If Q is idempotent, then this is exactly what happens.

Theorem 28. *For any $f, g : QA \to QB$,*

$$f \sim g \quad \Longrightarrow \quad f \approx_Q g$$

Proof. We have

$$Qf \circ \delta_A \sim \delta_B \circ f \sim \delta_B \circ g \sim Qg \circ \delta_A$$

Pre-composing with the inverse of δ_A yields $Qf \sim Qg$, and hence $f \approx_Q g$. □

Idempotence also implies another crucial piece of information regarding the product-preserving isomorphism $m_{A,B} : QA \times QB \xrightarrow{\cong} Q(A \times B)$. Even though we know $m_{A,B}$ is an isomorphism up to $\sim$, we ostensibly do not have any information on its behaviour up to $\approx_Q$: the hom-operations of Q do not preserve the PERs. However, in the idempotent setting it always iso, even up to $\approx_Q$.

Lemma 13. *$m_{A,B} : QA \times QB \xrightarrow{\cong} Q(A \times B)$ is an isomorphism up to $\approx_Q$, i.e. it is an isomorphism in the x-ray category $(\mathfrak{B}, \approx_Q)$.*

Proof. Calculate that

$$\begin{aligned}
& Qm \circ Q\langle Q\pi_1, Q\pi_2\rangle \circ \delta \\
\sim\ & \{\,\text{Proposition 2}\,\} \\
& Qm \circ m \circ \langle Q^2\pi_1, Q^2\pi_2\rangle \circ \delta \\
\sim\ & \{\,\text{naturality of product bracket, } \delta \text{ natural}\,\} \\
& Qm \circ m \circ \langle \delta \circ Q\pi_1, \delta \circ Q\pi_2\rangle \\
\sim\ & \{\,\text{product equation}\,\} \\
& Qm \circ m \circ (\delta \times \delta) \circ \langle Q\pi_1, Q\pi_2\rangle \\
\sim\ & \{\,\delta \text{ monoidal}\,\} \\
& \delta \circ m \circ \langle Q\pi_1, Q\pi_2\rangle \\
\sim\ & \{\, m^{-1} \stackrel{\text{def}}{=} \langle Q\pi_1, Q\pi_2\rangle \,\} \\
& \delta
\end{aligned}$$

Since δ is an isomorphism, we can cancel it on both sides to yield $m \circ m^{-1} \approx_Q id$. The calculation is similar in the opposite direction, and relies on post-composing with the isomorphism $m \circ (\delta \times \delta)$, and then cancelling it. □

The same trick with $m \circ (\delta \times \delta)$ also shows that Proposition 4 holds intensionally.

Lemma 14. $Q\pi_i^{A,B} \circ m_{A,B} \approx_Q \pi_i^{QA,QB}$.

These lemmas show that

Corollary 3. *For any* $f, g : \prod_{i=1}^n QA_i \to \prod_{j=1}^m QB_j$,

$$f \sim g \quad \Longrightarrow \quad f \approx_Q g$$

Proof. Pre-and-post-compose with the appropriate isomorphisms $m^{(n)}$ (see §7.1), use Theorem 28, and then use Lemma 13 to cancel the $m^{(n)}$'s. □

Some more lemmas

In this section we prove some more lemmas that hold in the idempotent case. Firstly, we can show that the comonadic diagrams commute intensionally.

Lemma 15. *The comonadic diagrams*

$$\begin{array}{ccc}
QA & \xrightarrow{\delta_A} & Q^2A \\
{\scriptstyle \delta_A}\downarrow & & \downarrow{\scriptstyle \delta_{QA}} \\
Q^2A & \xrightarrow[Q\delta_A]{} & Q^3A
\end{array}
\qquad
\begin{array}{ccc}
QA & \xrightarrow{\delta_A} & Q^2AT \\
{\scriptstyle \delta_A}\downarrow & \searrow^{id_{QA}} & \downarrow{\scriptstyle \epsilon_{QA}} \\
Q^2A & \xrightarrow[Q\epsilon_A]{} & QA
\end{array}$$

commute up to $\approx_Q$.

Proof. Simple calculations that mainly follow by pre-composing δ_A and then cancelling it. For example,

$$Q^2\epsilon_A \circ Q\delta_A \circ \delta_A \sim Q^2\epsilon_A \circ \delta_{QA} \circ \delta_A \sim \delta_A \circ Q\epsilon_A \circ \delta_A \sim \delta_A$$

by the equations and the naturality of δ, which gives that $Q\epsilon_A \circ \delta_A \approx_Q id_{QA}$. □

Moreover, we have that

Lemma 16. $\epsilon_A : QA \to A$ *is epic up to* $\approx_Q$.

Proof. If $f \circ \epsilon_A \approx_Q g \circ \epsilon_A : QA \to B$, then $Qf \circ Q\epsilon_A \sim Qg \circ Q\epsilon_A$. Pre-composing δ_A yields $f \approx_Q g$. □

The following lemma is also quite useful.

Lemma 17 (Quotation-Evaluation)**.** *For any* $f : QB \to QA$*, the following diagram commutes up to* $\sim$*:*

$$\begin{array}{ccc} QB & \xrightarrow{\delta_B} & Q^2B \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle Qf} \\ QA & \xleftarrow[Q\epsilon_A]{} & Q^2A \end{array}$$

Proof. We may calculate

$$Q\epsilon_A \circ Qf \circ \delta_B \sim Q\epsilon_A \circ \delta_A \circ f \sim f$$

by idempotence and the comonadic equations. □

This has a simple corollary when it comes to quoted points:

Corollary 4. *If* (Q, δ, ϵ) *is a product-preserving idempotent comonadic exposure, then*

$$Q(\epsilon_A \circ a) \circ m_0 \sim a$$

for any $a : \mathbf{1} \to QA$.

Proof. We may calculate

$$Q\epsilon \circ Qa \circ m_0 \sim Q\epsilon \circ \delta_A \circ a \sim a$$

by δ being reasonable and Q comonadic. □

Coalgebras, Idempotence, and Univalence

Recall the definition of *point-univalent objects* (Definition 26): U is point-univalent if $x \sim y : \mathbf{1} \to U$ implies $x \approx_Q y : \mathbf{1} \to U$. Intuitively, U is point-univalent if extensional and intensional equality coincide for its points, i.e. if none of them contain intensional information. Now, the objects QA are supposed to contain intensions/codes corresponding to the 'elements' of A. It is no surprise then that we can prove that

Lemma 18. *QA is point-univalent.*

Proof. Suppose $x \sim y : \mathbf{1} \to QA$. Then $x \circ \epsilon_{\mathbf{1}} \sim y \circ \epsilon_{\mathbf{1}} : Q\mathbf{1} \to QA$. Invoking Theorem 28 yields $x \circ \epsilon_{\mathbf{1}} \approx_Q y \circ \epsilon_{\mathbf{1}}$. If only we establish that $\epsilon_{\mathbf{1}} \circ m_0 \approx_Q id_{\mathbf{1}}$, then it would suffice to pre-compose m_0. But Q is product-preserving, hence cartesian, with the result that

$$\epsilon_{\mathbf{1}} \circ m_0 \approx_Q !_{\mathbf{1}} \approx_Q id_{\mathbf{1}}$$

□

In fact, this is a special case of a more general

Theorem 29. *Let $\mathcal{Q} \stackrel{def}{=} \{ QA \mid A \in \mathfrak{B} \}$. Then QA is $\mathcal{Q}$-univalent.*

This is just another way to state Theorem 28.

We would like to prove a kind of converse to this theorem, viz. that, with idempotence, every $\mathcal{Q}$-univalent object A is closely related to QA. Unfortunately, there is no systematic way to obtain an arrow $A \to QA$ from the $\mathcal{Q}$-univalence of A. But if we assume the existence of such an arrow—with appropriate equations—then it is easy to show that it is an isomorphism. This arrow is, of course, an old friend:

Definition 33. A *Q-coalgebra* is an arrow

$$\alpha : A \to QA$$

such that the following diagrams commute up to $\sim$:

$$\begin{array}{ccc} A & \xrightarrow{\alpha} & QA \\ & \searrow^{id_A} & \downarrow{\scriptstyle \epsilon_A} \\ & & A \end{array} \qquad \begin{array}{ccc} A & \xrightarrow{\alpha} & QA \\ {\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle \delta_A} \\ QA & \xrightarrow[Q\alpha]{} & Q^2A \end{array}$$

In the modality-as-intension interpretation a Q-coalgebra has an intuitive meaning: the equation $\epsilon_A \circ \alpha \sim id$ states that α can 'quote' the elements of A, producing an element of QA which—when evaluated—takes us back to where we started. The second equation merely states that α cooperates well with the quoter $\delta : Q \overset{\bullet}{\to} Q^2$. Thus, *a Q-coalgebra exists when we can internally quote the elements of an object.*

Lemma 19. *If A is $\mathcal{Q}$-univalent, then a Q-coalgebra $\alpha : A \to QA$ is an isomorphism, with inverse $\epsilon_A : QA \to A$.*

Proof. This is the classic proof that given idempotence all coalgebras are isomorphisms. However, a crucial step of that proof would rely on Q preserving equality; in this case, this is where $\mathcal{Q}$-univalence steps in.

We calculate that

$$\delta_A \circ (\alpha \circ \epsilon_A) \sim Q(\alpha \circ \epsilon_A) \circ \delta_A \sim Q\alpha \circ Q\epsilon_A \circ \delta_A \sim Q\alpha$$

by idempotence and the comonadic equations. If we post-compose $Q\epsilon_A$ to both sides of this equation, we obtain

$$\alpha \circ \epsilon_A \sim Q\epsilon_A \circ Q\alpha \sim Q(\epsilon_A \circ \alpha)$$

So, if we show that $\epsilon_A \circ \alpha \approx_Q id_A$, then we would obtain $\alpha \circ \epsilon_A \sim id_{QA}$. As we already know that $\alpha \circ \epsilon_A \sim id_{QA}$, we would conclude that $\alpha^{-1} \sim \epsilon_A$. Since $\epsilon_A \circ \alpha \sim id_A$, we obtain $\epsilon_A \circ \alpha \circ \epsilon_A \sim \epsilon_A : QA \to A$. But A is $\mathcal{Q}$-univalent, so

$$\epsilon_A \circ \alpha \circ \epsilon_A \approx_Q \epsilon_A$$

But ϵ_A is epic up to $\approx_Q$ (Lemma 16), so $\epsilon_A \circ \alpha \approx_Q id_A$. □

So, when is A univalent? Suppose that $\epsilon_A \circ \alpha \approx_Q id_A$, i.e. quoting and then evaluating returns the same intensional construction with which one began. Then it must be that A has no real intensional structure, and conversely.

Lemma 20. *Let $\alpha : A \to QA$ be a Q-coalgebra. Then A is $\mathcal{Q}$-univalent if and only if $\epsilon_A \circ \alpha \approx_Q id_A$.*

Proof. The 'only if' part was shown in the preceding proof. As for the 'if' part, let $f \sim g : QB \to A$. Then $\alpha \circ f \sim \alpha \circ g : QB \to QA$. By Theorem 29/28, $\alpha \circ f \approx_Q \alpha \circ g$. Post-composing with ϵ_A and using the assumption gives $f \approx_Q g$. □

To sum, we can combine these facts to show the following

Theorem 30. *If there is a Q-coalgebra $\alpha : A \to QA$ such that $\epsilon_A \circ \alpha \approx_Q id_A$, then $\alpha : A \xrightarrow{\cong} QA$ is an isomorphism.*

There is a partial converse, which is that

Theorem 31. *If Q is idempotent and there is an isomorphism $\alpha : A \xrightarrow{\cong} QA$ such that $\delta_A \circ \alpha \sim Q\alpha \circ \alpha$, then $\alpha : A \to QA$ is a Q-coalgebra, with $\epsilon_A \circ \alpha \approx_Q id_A$.*

Proof. We compute that

$$\alpha \sim Q\epsilon_A \circ \delta_A \circ \alpha \sim Q\epsilon_A \circ Q\alpha \circ \alpha$$

by the comonadic equations and the assumption. Cancelling α yields $\epsilon_A \circ \alpha \approx_Q id$, and hence α is a Q-coalgebra. □

Thus, either of the following data suffice to make A $\mathcal{Q}$-univalent:

1. an isomorphism $A \xrightarrow{\cong} QA$ with the second Q-coalgebra equation; or
2. a Q-coalgebra with the first equation holding intensionally.

4.2.5 Weakly Cartesian Closed Exposures

We close this section with a notion of cartesian closure for exposures. Unlike the situation with products, the relevant notion that will allow calculations under the exposure will be *weak*. We pick the weak notion because our motivating example of an exposure (see §5.2) is a weakly cartesian closed one. Whereas there is a good argument for an exposure to distribute over products—i.e. products do not contain any true intensional nature, they are simply pairs—exposures truly reveal the internal structure of morphisms: it makes sense that η does not hold, cf. the age-old discussion of the ordinary λ-theory $\lambda\beta$ and the extensional theory $\lambda\beta\eta$ in e.g. Barendregt [1984].

Definition 34. A product-preserving exposure $Q : \mathfrak{B} \to \mathfrak{C}$ from a P-ccc $\mathfrak{B}$ to a cartesian P-category $\mathfrak{C}$ is *weakly cartesian closed* just if

$$\mathsf{ev} \circ (\lambda(f) \times id) \approx_Q f$$

for all $f : C \times X \to Y$.

These work as expected. For example, if for $f : A \to B$ we let

$$\ulcorner f \urcorner \stackrel{\text{def}}{=} \lambda\left(\mathbf{1} \times A \xrightarrow{\pi_2} A \xrightarrow{f} B\right)$$

as before, then

Lemma 21. *For any* $a : C \to A$*,*

$$\mathsf{ev} \circ \langle \ulcorner f \urcorner, a \rangle \approx_Q f \circ a$$

Proof. All the necessary equations hold intensionally:

$$\mathsf{ev} \circ \langle \ulcorner f \urcorner, a \rangle \approx_Q \mathsf{ev} \circ (\ulcorner f \urcorner \times id) \circ \langle id_C, a \rangle \approx_Q f \circ \pi_2 \circ \langle id_C, a \rangle \approx_Q f \circ a$$

□

Chapter 5

Three Examples of Exposures[1]

Having introduced the basics of P-categories and exposures in the previous chapter, we now seek to prove that they are indeed useful abstractions in the study of intensional phenomena. Towards this goal, we shall present three examples of a P-category and an endoexposure on it.

In §5.1 we will construct a P-category based on a *first-order classical theory*. In the particular case of *Peano Arithmetic (PA)*, it will become apparent that a well-behaved Gödel numbering comprises an endoexposure.

Following that, in §5.2 we turn to *realizability theory* in order to obtain a handle on intensionality in settings related to higher-order computability theory. The example of a comonadic exposure constructed therein is particularly well-behaved, and is the motivating example for the entire development of this thesis.

The third example is, we hope, somehow unfamiliar. Prompted by Abramsky [2014], a natural question arises: is the phenomenon of intensionality only relevant to logic and computation? We are prepared to entertain the idea that it may be a more general mathematical pattern, which has hitherto been either a nuisance or—more often—invisible, and whose categorical formulation will enable us to recognise it in more settings. In §5.3, we present a simple example of intensionality found in *homological algebra*, and submit it to the reader for further discussion.

5.1 Exposures as Gödel Numbering

Our first example of an exposure substantiates the claim that exposures can be considered as abstract analogues of Gödel numberings.

[1] A preliminary form of the results in this chapter was first published as [Kavvos, 2017a], which is available at Springer: `https://dx.doi.org/10.1007/978-3-662-54458-7_32`

Following Lawvere [1969, 2006], we will construct a P-category from a first-order theory T. Then, we will sketch the proof that any sufficiently well-behaved Gödel numbering defines an exposure over this theory. We omit the details, leaving them as a (probably rather complicated) exercise in coding.

5.1.1 The Lindenbaum P-category

The construction in this section is called the *Lindenbaum P-category* of a first-order theory T, and it is simpler than what one might imagine: we begin with three basic objects,

$$\begin{array}{ll} \mathbf{1}, & \text{the terminal object} \\ \mathbf{2}, & \text{the object of truth values} \\ A, & \text{the universe} \end{array}$$

The objects of the category will be the *formal products* of these three objects; we write Γ for a generic product $A_1, \ldots, A_n$ of these objects. The arrows of type

$$\Gamma \to \mathbf{2}$$

will be construed as *formulas*, with free variables in Γ. Similarly, the arrows

$$\Gamma \to A$$

will be construed as *terms*, once again with free variables in Γ. In this light, a formula $\phi(x, y, z)$ in three free variables will be an arrow

$$\phi(x, y, z) : A \times A \times A \to A$$

with the three copies of A in the domain representing each free variable in a fixed order—and similarly for terms.

However, this idea will be complicated by the presence of $\mathbf{2}$ in the domain. This will represent a *Boolean hole* that may be filled by a formula. For example, consider the expression

$$\phi(x, P) \stackrel{\text{def}}{=} \forall y.\ \exists z.\ (f(x, y) = z \wedge P)$$

is a formula with one free variable x, and one Boolean hole P, for which a formula can be substituted. Hence, it will be an arrow

$$\phi(x, P) : A \times \mathbf{2} \to \mathbf{2}$$

This is a generalisation that Lawvere introduced, and of which we shall be making use. The interesting thing about it is that the Boolean connectives now appear as arrows, e.g. $\wedge : \mathbf{2} \times \mathbf{2} \to \mathbf{2}$, and $\neg : \mathbf{2} \to \mathbf{2}$. Of course, arrows $\Gamma \to A$ with $\mathbf{2}$ occurring in Γ do not make particularly good sense: how could one have a Boolean hole in a term?

Finally, we have the cases of $\mathbf{1}$ and products appearing in the codomain. In the first case, there will be a unique arrow $!_\Gamma : \Gamma \to \mathbf{1}$. In the case of a product $\Delta \stackrel{\text{def}}{=} B_1 \times \cdots \times B_n$, arrows $f : \Gamma \to \Delta$ will be freely generated by brackets, i.e. they will be

$$\langle f_1, \ldots, f_n \rangle : \Gamma \to \Delta$$

where $f_i : \Gamma \to B_i$. Almost everything in sight will act component-wise on these.

We shall say that

$$f \sim g : \Gamma \to \mathbf{2} \quad \text{just if} \quad \mathsf{T} \vdash f \leftrightarrow g$$

That is, two arrows that are predicates are extensionally equal if and only if they can be proven equivalent in the theory. The details are slightly more complicated than in most presentations of first-order logic, since we also have to treat Boolean holes. Similarly, two terms are extensionally equal if they can be proven equal in the theory, i.e.

$$t \sim s : \Gamma \to A \quad \text{just if} \quad \mathsf{T} \vdash t = s$$

On brackets, $\sim$ acts component-wise.

It is not hard to see that we obtain the following theorem, which—excluding the P-categorical twist—is essentially due to Lawvere.

Theorem 32. *The above construction is a cartesian P-category, called the* Lindenbaum P-category $\mathfrak{Lind}(\mathsf{T})$ *of the first-order theory* T.

5.1.2 Numbering as Exposure

Let us now concentrate on the case of Peano Arithmetic, which we denote by PA. A *Gödel numbering* [Boolos, 1994, Smullyan, 1992] is obtained when one assigns to each formula $\phi(\overrightarrow{x})$ and each term $t(\overrightarrow{x})$ a *Gödel number*, denoted by

$$\ulcorner \phi(\overrightarrow{x}) \urcorner, \quad \ulcorner t(\overrightarrow{x}) \urcorner$$

respectively. In the case of a first-order theory of arithmetic, the Gödel number of a term or formula is supposed to represent the term or formula *within the theory*.

The notion of representation-within-the-theory is exactly what exposures are meant to capture. Hence, we need to define the action of an exposure,

$$Q : \mathfrak{Lind}(\mathsf{PA}) \nrightarrow \mathfrak{Lind}(\mathsf{PA})$$

on the Lindenbaum P-category of PA. This is where we need that the Gödel numbering be well-behaved. Let us suppose that we have a formula $\phi(x, y) : A \times A \to A$ in two free variables. We would like to map this to a formula $Q\phi(x, y)$, also in two free variables, that respects substitution. Let us presume that we have functions

$$sub_{\text{wff},n}(x, z_1, \ldots, z_n)$$
$$sub_{\text{t},n}(x, z_1, \ldots, z_n)$$

that are definable in PA by a term (denoted by the same name), and moreover that these functions *define substitution for the Gödel numbering*, in the sense that, for example, if given—say—two closed terms t, s, we have

$$\mathsf{PA} \vdash sub_{\text{wff},2}\left(\ulcorner \phi(x, y) \urcorner, \ulcorner t \urcorner, \ulcorner s \urcorner\right) = \ulcorner \phi(t, s) \urcorner$$

We require more than most presentations of Gödel numberings. In particular, we require that these behave well under substitution, e.g. we require that the terms

$$sub_{\text{wff},2}\left(\ulcorner \phi(x, y) \urcorner, sub_{\text{t},n}(\ulcorner t(\overrightarrow{z}) \urcorner, \overrightarrow{w}), sub_{\text{t},n}(\ulcorner s(\overrightarrow{z}) \urcorner, \overrightarrow{w})\right)$$

and

$$sub_{\text{wff},n}(\ulcorner \phi\left(t(\overrightarrow{z}), s(\overrightarrow{z})\right) \urcorner, \overrightarrow{w})$$

be provably equal in PA. We can then define

$$Q\phi(x, y) \stackrel{\text{def}}{=} sub_{\text{wff},2}(\ulcorner \phi(x, y) \urcorner, x, y)$$

Finally, in the case of a *sentence* $\psi : \mathbf{1} \to \mathbf{2}$, i.e. a closed formula, we shall define $Q\psi : \mathbf{1} \to A$ to simply be the (numeral of the) Gödel number, i.e.

$$Q\psi \stackrel{\text{def}}{=} \ulcorner \psi \urcorner$$

This certainly respects substitution! We have elided the concept of Boolean holes, but we believe that to be a not so difficult exercise. Identities are a little stranger; the identity at A is the term that is a single free variable, i.e.

$$id_A \stackrel{\text{def}}{=} x : A \to A$$

and this is mapped to $sub_{t,1}(\ulcorner x \urcorner, x) : A \to A$, which somehow needs to be provably equal to x itself. This is a also strange requirement for a Gödel numbering, but we are mostly willing to believe that it is a feasible desideratum. A similar situation occurs when we examine the arrow

$$\langle Q\pi_1, Q\pi_2 \rangle = \langle sub_{t,2}(\ulcorner x \urcorner, x, y), sub_{t,2}(\ulcorner y \urcorner, x, y) \rangle$$

For the exposure Q to be product-preserving, we would like this to be an isomorphism, or—even better—the identity. And it would indeed be, if we could prove that, in general,

$$\mathsf{PA} \vdash sub_{t,n}(\ulcorner z_i \urcorner, \overrightarrow{z}) = z_i$$

More rigorously, we define

$$\begin{aligned} Q\mathbf{1} &\stackrel{\text{def}}{=} \mathbf{1} \\ Q(A) &\stackrel{\text{def}}{=} A \\ Q(\mathbf{2}) &\stackrel{\text{def}}{=} A \\ Q(B_1 \times \cdots \times B_n) &\stackrel{\text{def}}{=} QB_1 \times \cdots \times QB_n \end{aligned}$$

and, of course, $Q\langle f_1, \ldots, f_n \rangle \stackrel{\text{def}}{=} \langle Qf_1, \ldots, Qf_n \rangle$.

To complete the construction, we have to check the last axiom of exposures, namely reflection of PERs. For that we need that

$$\mathsf{PA} \vdash sub_{\text{wff},n}(\ulcorner \phi(\overrightarrow{x}) \urcorner, \overrightarrow{z}) = sub_{\text{wff},n}(\ulcorner \psi(\overrightarrow{x}) \urcorner, \overrightarrow{z})$$

implies

$$\mathsf{PA} \vdash \phi \leftrightarrow \psi$$

By substituting the Gödel numbers of variables $\overrightarrow{y}$ in the antecedent, we obtain that

$$\mathsf{PA} \vdash \ulcorner \phi(\overrightarrow{y}) \urcorner = \ulcorner \psi(\overrightarrow{y}) \urcorner$$

and so it suffices for the Gödel numbering to be injective, in the sense that

$$\ulcorner \phi(\overrightarrow{y}) \urcorner = \ulcorner \psi(\overrightarrow{y}) \urcorner \quad \Longrightarrow \quad \phi(\overrightarrow{y}) = \psi(\overrightarrow{y})$$

viz. that equality of Gödel numbers implies syntactic equality. In conclusion,

Theorem 33. *If all of the above desiderata on Gödel numberings are feasiable, then the construction is a product-preserving endoexposure,*

$$Q : \mathfrak{Lind}(\mathsf{PA}) \looparrowright \mathfrak{Lind}(\mathsf{PA})$$

on the Lindenbaum P-category of PA.

5.2 Exposures in Realizability

In this section we study our central example of an exposure, which hails from realizability theory.

The basic objects in realizability are *assemblies*, i.e. sets of which every element is associated with a *set of realizers*. The elements of the set can be thought of as the elements of an *abstract datatype*, whereas the set of realizers of each element contains its multiple *machine-level representations*. For example, realizers can range over the natural numbers; then taking functions between assemblies which can be 'tracked' on the level of realizers by partial recursive functions yields a category where 'everything is computable.'

In practice, the generalisation from natural numbers to an arbitrary *partial combinatory algebra (PCA)* is made. A PCA is an untyped 'universe' corresponding to some notion of computability or realizability. There are easy tricks with which one may encode various common first-order datatypes (such as booleans, integers, etc.) in a PCA.[2] Moreover, it is easy to show that, up to a simple representation of integers, one may represent all partial recursive functions in a PCA. Before proceeding with the construction, we recap in §5.2.1 the basics of PCAs. For the interested reader let us mention that detailed discussions may be found in [Beeson, 1985, Longley, 1995, Longley and Simpson, 1997, van Oosten, 2008, Longley and Normann, 2015].

Once a PCA—and hence a notion of computability—is fixed, defining a P-category $\mathfrak{Asm}(A)$ is reasonably straightforward: objects are assemblies, and morphisms are functions which are 'computable' on the level of realizers. The arrows are pairs (f, r) where f is a function on the underlying set of each assembly, and r is an element of the PCA that tracks the function f. Two arrows are related if and only if they define the same function on the underlying sets. Finally, to define an endoexposure all we need to do is display the effect of each tracking element r on the realizers. It then transpires that this exposure is a product-preserving idempotent comonadic exposure on $\mathfrak{Asm}(A)$.

5.2.1 Partial Combinatory Algebras

The definition of a PCA is deceptively simple:

[2] Most of these tricks have their origins in untyped λ-calculus, and are hence found in Barendregt [1984].

Definition 35. A *partial combinatory algebra (PCA)* $(A, \cdot)$ consists of a carrier set A and a partial binary operation $\cdot : A \times A \rightharpoonup A$ such that there exist $\mathbf{K}, \mathbf{S} \in A$ with the properties that

$$\mathbf{K} \cdot x \downarrow, \quad \mathbf{K} \cdot x \cdot y \simeq y, \quad \mathbf{S} \cdot x \cdot y \downarrow, \quad \mathbf{S} \cdot x \cdot y \cdot z \simeq x \cdot z \cdot (y \cdot z)$$

for all $x, y, z \in A$.

The paradigmatic example comes from ordinary computability theory, as presented by e.g. Cutland [1980], Odifreddi [1992], Rogers [1987]. It is not very difficult to use the s-m-n theorem to cook up indices that behave like $\mathbf{S}$ and $\mathbf{K}$, to the effect that

Theorem 34 (Kleene's First Model)**.** *The applicative structure* $K_1 = (\mathbb{N}, \cdot)$*, where*

$$x \cdot y \simeq \phi_x(y)$$

is a partial combinatory algebra.

Combinatory Completeness

Even though the definition of a PCA is remarkably simple, there is more to it than meets the eye. The structure of $\mathbf{S}$ and $\mathbf{K}$ suffice to obtain a property known as *combinatory completeness*: every syntactic function on the PCA formed by variables, applications and constants can be 'internalised' as an element of the PCA. This results originates from combinatory logic, and is well-known in the study of untyped λ-calculus—see Barendregt [1984].

Once combinatory completeness is obtained, standard tricks from untyped λ-calculus can be used to represent first-order data, but also greatly simplify calculations. Nevertheless, let us remind the reader that *not all the common rules of the untyped λ-calculus hold in a PCA*, so caution is advised. The particular presentation in this section is due to Longley [1995].

Definition 36. Let V be an infinite set of variables. The set $\mathcal{E}(A)$ of *terms* or *formal expressions* over a PCA $(A, \cdot)$ is defined as the least set that satisfies the following conditions:

$$A \subseteq \mathcal{E}(A), \quad V \subseteq \mathcal{E}(A), \quad \frac{s \in \mathcal{E}(A) \quad t \in \mathcal{E}(A)}{(s \cdot t) \in \mathcal{E}(A)}$$

Conventionally, we will use $e, s, t, u, v, \ldots$ as metavariables ranging over $\mathcal{E}(A)$, and we will write $s[t/x]$ for the formal expression obtained by substituting $t \in \mathcal{E}(A)$ for every occurrence of the variable x in the formal expression $s \in \mathcal{E}(A)$.

A formal expression is *closed* if it contains no variables. Write $\mathrm{FV}(e)$ for the set of free variables of expression $e \in \mathcal{E}(A)$. Also, write $e\downarrow$ ("e denotes"), where e is a closed formal expression, to mean that that, if the formal expression is interpreted as an actual algebraic expression in the standard way, composition throughout is defined and it denotes an element. This implies that all subexpressions also denote: if $s \cdot t\downarrow$, then $s\downarrow$ and $t\downarrow$. Otherwise, we write $e\uparrow$ to mean that partiality kicks in, and the expression does not denote an element in the PCA.

We also notationally distinguish two equalities: the strict equality, $s = t$, where both $s\downarrow$ and $t\downarrow$ and they denote the same element; and the Kleene equality, $s \simeq t$, which holds when s and t are both undefined, or both defined and denote the same element.

Finally, the above notions are straightforwardly extended to open terms, by substituting for all variables: let the variables of $s, t \in \mathcal{E}(A)$ be amongst $x_1, \ldots, x_n$; then, for example

$$\begin{aligned} e\downarrow \qquad & \text{just if} \quad \forall a_1, \ldots, a_n \in A.\ e[\vec{a}/\vec{x}]\downarrow \\ s \simeq t \qquad & \text{just if} \quad \forall a_1, \ldots, a_n \in A.\ s[\vec{a}/\vec{x}] \simeq t[\vec{a}/\vec{x}] \end{aligned}$$

for the obvious generalisation to simultaneous substitution. We can now λ-abstract:

Theorem 35 (Combinatory Completeness)**.** *Let $(A, \cdot)$ be a PCA. For any $e \in \mathcal{E}(A)$, there exists a formal expression $\lambda^* x.e \in \mathcal{E}(A)$, where $\mathrm{FV}(\lambda^* x.e) = \mathrm{FV}(e) - \{x\}$, such that*

$$\lambda^* x.e\downarrow$$

and

$$(\lambda^* x.e)a \simeq e[a/x]$$

for all $a \in A$.

Proof. Define

$$\begin{aligned} \lambda^* x.\ x &\stackrel{\text{def}}{=} \mathbf{S} \cdot \mathbf{K} \cdot \mathbf{K} \\ \lambda^* x.\ t &\stackrel{\text{def}}{=} \mathbf{K} \cdot t && \text{if } t \in A \cup V \text{ and } t \not\equiv x \\ \lambda^* x.\ s \cdot t &\stackrel{\text{def}}{=} \mathbf{S} \cdot (\lambda^* x.s) \cdot (\lambda^* x.t) \end{aligned}$$

Then $\lambda^* x.e\downarrow$ by the definedness conditions for $\mathbf{S}$ and $\mathbf{K}$. The rest follows by induction. □

Let us be pedantic and reiterate the warnings: Kleene equality is *not* respected by the $\lambda^* x$ operation on terms, and the obvious β-rules do not hold—observe that the operand above has to be a constant! Longley carefully develops correct β-rules for this language of terms in [Longley, 1995, §1.1.2].

Some common encodings

We will need the following combinators:

$$\mathbf{I} \stackrel{\text{def}}{=} \mathbf{SKK}$$
$$\mathbf{B} \stackrel{\text{def}}{=} \lambda^* f.\lambda^* g.\lambda^* x.f(gx)$$

It is easy to define selection and pairs. Let

$$\mathsf{true} \stackrel{\text{def}}{=} \lambda^* ab.a$$
$$\mathsf{false} \stackrel{\text{def}}{=} \lambda^* ab.b$$
$$\mathsf{if} \stackrel{\text{def}}{=} \lambda^* xyz.xyz$$
$$\mathsf{pair} \stackrel{\text{def}}{=} \lambda^* xyz.zxy$$
$$\mathsf{fst} \stackrel{\text{def}}{=} \lambda^* p.p(\mathsf{true})$$
$$\mathsf{snd} \stackrel{\text{def}}{=} \lambda^* p.p(\mathsf{false})$$

We always have $\mathsf{if}\ x\ y \downarrow$, and $\mathsf{pair}\ x\ y \downarrow$. Furthermore, the following equalities hold:

$$\mathsf{fst}\ (\mathsf{pair}\ x\ y) = x$$
$$\mathsf{snd}\ (\mathsf{pair}\ x\ y) = y$$
$$\mathsf{if}\ \mathsf{true}\ y\ z = y$$
$$\mathsf{if}\ \mathsf{false}\ y\ z = z$$

Encoding numbers is not more difficult, and—like [Longley, 1995] and [Longley and Simpson, 1997]—we use a trick due to Curry. Let

$$\overline{0} = \mathbf{I}$$
$$\overline{n+1} = \mathsf{pair}\ \mathsf{false}\ \overline{n}$$

Then we may let $\mathsf{succ} \stackrel{\text{def}}{=} \lambda^* x.\ \mathsf{pair}\ \mathsf{false}\ x$, so that $\mathsf{succ}\ \overline{n} = \overline{n+1}$. To check if a number is zero, use $\mathsf{iszero} \stackrel{\text{def}}{=} \mathsf{fst}$ so that

$$\mathsf{iszero}\ \overline{0} = \mathbf{I}\ (\mathsf{true}) = \mathsf{true}$$

whereas

$$\mathsf{iszero}\ \overline{n+1} = \mathsf{fst}\ (\mathsf{pair}\ \mathsf{false}\ \overline{n}) = \mathsf{false}$$

Finally, we can define the predecessor function by

$$\mathsf{pred} \stackrel{\text{def}}{=} \lambda^* x.\ \mathsf{if}\ (\mathsf{iszero}\ x)\ \overline{0}\ (\mathsf{snd}\ x)$$

5.2.2 Assemblies and Modest Sets

Definition 37. An *assembly* X on A consists of a set $|X|$ and for each $x \in |X|$ a non-empty subset $\|x\|_X \subseteq A$. If $a \in \|x\|_X$, we say that a *realizes* x.

Definition 38. For two assemblies X and Y, a function $f : |X| \to |Y|$ is said to be *tracked by* $r \in A$ just if, for all $x \in |X|$ and $a \in \|x\|_X$, we have

$$r \cdot a \downarrow \quad \text{and} \quad r \cdot a \in \|f(x)\|_Y$$

Definition 39. An assembly X on A is a *modest set* just if no element of A realizes two elements of $|X|$. That is,

$$x \neq x' \quad \Longrightarrow \quad \|x\|_X \cap \|x'\|_X = \emptyset$$

It is not hard to see that for each PCA A we can define a category $\mathbf{Asm}(A)$, with objects all assemblies X on A, and morphisms $f : X \to Y$ being functions $f : |X| \to |Y|$ that are tracked by some $r \in A$. In fact,

Theorem 36. *Assemblies and trackable morphisms between them form a category* $\mathbf{Asm}(A)$ *that is cartesian closed, has coproducts, as well as a natural numbers object.*

It is not clear who originated the—admittedly very intuitive—definition of $\mathbf{Asm}(A)$, and who first proved the above theorem. The identification of assemblies as the $\neg\neg$-separated objects of the effective topos can be found in the work of Hyland [1982]. Longo and Moggi [1991] refer to $\mathbf{Asm}(A)$ as the category of ω-sets, and so does Jacobs [1999]. For more details, see [Longley, 1995] or [Longley and Simpson, 1997].

A special subcategory of assemblies is of particular interest:

Theorem 37. *The full subcategory of* $\mathbf{Asm}(A)$ *consisting only of objects which are modest sets, which we denote by* $\mathbf{Mod}(A)$*, inherits the cartesian closed, coproduct, and natural number object structure from the category of assemblies.*

The category of modest sets—or its equivalent presentation in terms of PERs on the PCA A—seems to have originated in unpublished work by Turing, and later used by Gandy [1956, 1959]: see [Hyland, 2016]. In semantics, PERs were used independently by Scott [1976] and Girard [1972]. Accessible presentations may be found in Mitchell [1996] or Crole [1993].

5.2.3 Passing to a P-category

The lack of intensionality in the category $\mathbf{Asm}(A)$ is blatant: to elevate a function $f : |X| \to |Y|$ to a morphism $f : X \to Y$, we only require that *there exists* a $r \in A$ that tracks it: as soon as this is witnessed, we throw away the witness. For all the reasons discussed in §4.1 we have to move to P-categories to mend this.

The P-category of assemblies on A, denoted $\mathfrak{Asm}(A)$, is defined as follows. Its objects are once more all assemblies X on A. Given assemblies X and Y, the P-set $\mathfrak{Asm}(X, Y)$ is defined by having underlying set

$$|\mathfrak{Asm}(X,Y)| \stackrel{\text{def}}{=} \{\, (f : |X| \to |Y|\,, r \in A) \mid r \text{ tracks } f \,\}$$

and

$$(f, r) \sim_{\mathfrak{Asm}(X,Y)} (g, s) \quad \text{just if} \quad f = g$$

For $(f, r) : X \to Y$ and $(g, s) : Y \to Z$, we define composition by

$$(g, s) \circ (f, r) \stackrel{\text{def}}{=} (g \circ f, \mathbf{B} \cdot s \cdot r)$$

Notice that for any $x \in |X|$, $a \in \|x\|_X$ implies $p \cdot a \in \|f(x)\|_Y$, which implies $q \cdot (p \cdot a) \in \|g(f(x))\|_Z$, and as $\mathbf{B} \cdot q \cdot p \cdot a \simeq q \cdot (p \cdot a)$ it follows that $(g \circ f, \mathbf{B} \cdot p \cdot q)$ is an arrow $X \to Z$. It is easy to see that composition is a P-function: $\sim$ only refers to the underlying 'extensional' functions. Composition of set-theoretic functions is associative and the identity function is its unit. That said, we define the identity $id_X : X \to X$ to simply be $(id_{|X|}, \mathbf{I})$.

Finite Products

It is not hard to show that

Proposition 7. *The category $\mathfrak{Asm}(A)$ has binary products.*

Proof. The construction essentially follows the underlying structure of products in the category of sets, but augments it with tracking elements. For assemblies X and Y, we define

$$|X \times Y| \stackrel{\text{def}}{=} |X| \times |Y|\,, \qquad \|(x,y)\|_{X\times Y} \stackrel{\text{def}}{=} \{\, \mathsf{pair}\, a\ b \mid a \in \|x\|_X\,, b \in \|y\|_Y \,\}$$

The projections are the following arrows:

$$\pi_1 \stackrel{\text{def}}{=} (\pi_1 : |X| \times |Y| \to |X|\,, \mathsf{fst})$$
$$\pi_2 \stackrel{\text{def}}{=} (\pi_2 : |X| \times |Y| \to |Y|\,, \mathsf{snd})$$

Define the P-function $\langle -, - \rangle$ by

$$\langle (f, r), (g, s) \rangle \stackrel{\text{def}}{=} (\langle f, g \rangle, \lambda^* c.\ \mathsf{pair}\ (r\ c)\ (s\ c))$$

It is easy to see that this is a P-function. We compute that

$$\pi_1 \circ \langle (f, r), (g, s) \rangle \stackrel{\text{def}}{=} (\pi_1 \circ f, \mathbf{B} \cdot \mathsf{fst} \cdot (\lambda^* c.\ \mathsf{pair}\ (r\ c)\ (s\ c))) \sim (f, r)$$

and similarly for the other two equations. □

Proposition 8. *The P-category $\mathfrak{Asm}(A)$ has a terminal object.*

Proof. Define $\mathbf{1} \in \mathfrak{Asm}(A)$ by

$$|\mathbf{1}| \stackrel{\text{def}}{=} \{*\}, \qquad \|*\|_{\mathbf{1}} \stackrel{\text{def}}{=} \{\overline{0}\}$$

and, for $A \in \mathfrak{Asm}(A)$, let $!_A \stackrel{\text{def}}{=} (a \mapsto *, \mathbf{K} \cdot \overline{0}) : A \to \mathbf{1}$, which is unique (up to $\sim$). □

Hence,

Theorem 38. *$\mathfrak{Asm}(A)$ has finite products.*

Exponentials

Given assemblies X and Y, we define

$$\left|Y^X\right| \stackrel{\text{def}}{=} \{\, f \mid (f, r) : X \to Y \,\}, \qquad \|f\|_{Y^X} \stackrel{\text{def}}{=} \{\, r \mid r \text{ tracks } f \,\}$$

Let

$$\mathsf{ev}_{X,Y} \stackrel{\text{def}}{=} ((f, x) \mapsto f(x), \lambda^* p.\ (\mathsf{fst}\ p)\ (\mathsf{snd}\ p)) : Y^X \times X \to Y$$

Define a P-function

$$\lambda_C : \mathfrak{Asm}(A)(C \times X, Y) \to \mathfrak{Asm}(A)(C, Y^X)$$

by

$$(f, r) \mapsto (z \mapsto (x \mapsto f(z, x)), \lambda^* c.\ \lambda^* a.\ r(\mathsf{pair}\ c\ a))$$

It is again easy to see that this is a P-function, and one can verify that this is the exponential. Therefore,

Theorem 39. *$\mathfrak{Asm}(A)$ is cartesian closed.*

The Lifted Assembly for K_1

We only mention one other indispensable construction, which embodies partiality in the computable setting of assemblies on K_1. Given an assembly $X \in \mathfrak{Asm}(K_1)$, the *lifted assembly* $X_\perp$ is defined to be

$$|X_\perp| \stackrel{\text{def}}{=} |X| + \{\perp\}, \qquad \|x\|_{X_\perp} \stackrel{\text{def}}{=} \begin{cases} \{\, r \mid r \cdot \overline{0}\downarrow \text{ and } r \cdot \overline{0} \in \|x\|_X \,\} & \text{for } x \in |X| \\ \{\, r \mid r \cdot \overline{0}\uparrow \,\} & \text{for } x = \perp \end{cases}$$

for some chosen element of the PCA $\overline{0}$. Elements of $X_\perp$ are either elements of X, or the undefined value $\perp$. Realizers of $x \in |X|$ are 'computations' $r \in A$ which, when run (i.e. given the dummy value $\overline{0}$ as argument) return a realizer of x. A computation that does not halt when run represents the undefined value.

Bear in mind that this definition of the lifted assembly is only useful in K_1. In particular, it does not work at all if the PCA is total. There are other, more involved ways of defining the lifted assembly: see Longley and Simpson [1997] for a rather elegant and uniform method.

5.2.4 The Exposure

Theorem 40. *There is an exposure*

$$\Box : \mathfrak{Asm}(A) \nrightarrow \mathfrak{Asm}(A)$$

for any PCA A.

Proof. For an assembly $X \in \mathfrak{Asm}(A)$, let $\Box X$ be the assembly defined by

$$|\Box X| \stackrel{\text{def}}{=} \{\, (x, a) \mid x \in |X|\,, a \in \|x\|_X \,\}$$
$$\|(x, a)\|_{\Box X} \stackrel{\text{def}}{=} \{\, a \,\}$$

Given $(f, r) : X \to Y$, we define $\Box(f, r) = (f_r, r) : \Box X \to \Box Y$ where

$$f_r : |\Box X| \longrightarrow |\Box Y|$$
$$(x, a) \longmapsto (f(x), r \cdot a)$$

Each element $(x, a) \in |\Box X|$ has a unique realizer, a. As $a \in \|x\|_X$, and f is tracked by r, we see that $r \cdot a \downarrow$ and $r \cdot a \in \|f(x)\|_Y$, so that $(f(x), r \cdot a) \in |\Box Y|$. It follows that r tracks f_r, and so (f_r, r) is an arrow $\Box X \to \Box Y$.

To prove that $\Box$ preserves composites, observe that for arrows $(f, r) : A \to B$ and $(g, s) : B \to C$, we have

$$(x, a) \overset{f_r}{\longmapsto} (f(x), r \cdot a) \overset{g_s}{\longmapsto} (g(f(x)), s \cdot (r \cdot a))$$

and as $\mathbf{B} \cdot s \cdot r \cdot a \simeq s \cdot (r \cdot a)$, we have

$$g_s \circ f_r = (g \circ f)_{\mathbf{B} \cdot s \cdot r}$$

which is of course tracked by $\mathbf{B} \cdot s \cdot r$. Hence $\Box(g \circ f) \sim \Box g \circ \Box f$. Regarding identities, notice that if $id_X : X \to X$ is the identity arrow, then

$$(x, a) \overset{(id_X)_{\mathbf{I}}}{\longmapsto} (x, \mathbf{I} \cdot a)$$

and as $\mathbf{I} \cdot a \simeq a$, the latter is equal to (x, a), so $(id_X)_{\mathbf{I}} = id_{|\Box X|}$. Hence $\Box id_X \sim id_{\Box X}$.

Finally, we need to show that intensional equality implies extensional equality. Suppose $\Box(f, r) \sim \Box(g, s)$. That is equivalent to $f_r = g_s$, which in turn gives us both $f = g$ and also $r \cdot a \simeq s \cdot a$ for all $a \in \|x\|_X$. The first implies $(f, r) \sim (g, s)$. $\Box$

Intensional Equality

When showing that $\Box$ is an exposure, we inadvertently characterised intensional equality up to $\Box$:

Lemma 22. *$(f, r) \approx_{\Box} (g, s) : X \to Y$ precisely when $f_r = g_s$, i.e.*

$$f = g \quad \textit{and} \quad \forall x \in |X| \,.\, \forall a \in \|x\|_X \,.\, r \cdot a \simeq s \cdot a$$

This is indeed the meaning we expected of intensional equality: (f, r) and (g, s) are not only equal extensionally, but they also have *the same effect on realizers*. Notice that, unless we are in a highly extensional environment (which not all PCAs are), this is very far from *strict* equality. Instead, it is something in between.

We can also use this characterisation of intensional equality to speak about the realizer structure of assemblies, in particular by characterising which objects of $\mathfrak{Asm}(A)$ are univalent (recall Definition 26).

Definition 40. An assembly X has *unique realizers* just if $\|x\|_X$ is a singleton, for each $x \in |X|$.

Lemma 23. *The univalent objects of $\mathfrak{Asm}(A)$ are precisely those which have unique realizers.*

Proof. Suppose X is univalent. To each realizer $a \in \|x\|_X$, there corresponds an arrow

$$\hat{x}_a \stackrel{\text{def}}{=} (* \mapsto x, \lambda^* c.\ a) : \mathbf{1} \to X$$

But if $a, b \in \|x\|_X$, then $\hat{x}_a \sim \hat{x}_b$, as they share the same function component $(* \mapsto x)$. As X is univalent, it follows that $\hat{x}_a \approx_\square \hat{x}_b$, so

$$a \simeq (\lambda^* c.\ a) \cdot \overline{0} \simeq (\lambda^* c.\ b) \cdot \overline{0} \simeq b$$

Conversely, suppose X has unique realizers. For any two extensionally equal arrows $(f, r), (f, s) : Y \to X$, it is not very hard to see that $f_r = f_s$: we have that $\pi_1(f_r(y, b)) = f(y) = \pi_1(f_s(y, b))$ for any $b \in \|y\|_Y$. Thus, as $f(y)$ is uniquely realized by only one a, their second components are equal too, and $(f, r) \approx_\square (f, s)$. $\square$

$\square$ is cartesian

It so happens that projections behave nicely under exposures.

Theorem 41. $\square : \mathfrak{Asm}(A) \nrightarrow \mathfrak{Asm}(A)$ *is a cartesian exposure.*

Recall that $\pi_1 \circ \langle (f, r), (g, s) \rangle = (f, d)$, where

$$d \stackrel{\text{def}}{=} \mathbf{B} \cdot \mathsf{fst} \cdot (\lambda^* c.\ \mathsf{pair}\ (r\ c)(s\ c))$$

so that the function component of $\square(\pi_1 \circ \langle (f, r), (g, s) \rangle)$ is $f_d(z, c) = (f(z), d \cdot c)$. We compute that

$$d \cdot c = \mathbf{B} \cdot \mathsf{fst} \cdot (\lambda^* c.\ \mathsf{pair}\ (r\ c)(s\ c)) \cdot c = r \cdot c$$

which is to say that $f_d = f_r$, and hence $\square(\pi_1 \circ \langle (f, r), (g, s) \rangle) \sim \square(f, r)$. The calculation is similar for the other projection.

For the third equation, it is easy to calculate that, for any arrow $h : C \to X \times Y$, the function component of $\square \langle \pi_1 \circ (h, r), \pi_2 \circ (h, r) \rangle$ is h_s, where

$$s = \lambda^* c.\ \mathsf{pair}\ (\mathbf{B} \cdot \mathsf{fst} \cdot r \cdot c)\ (\mathbf{B} \cdot \mathsf{snd} \cdot r \cdot c)$$

We proceed by calculating that

$$s \cdot c = \mathsf{pair}\ (\mathbf{B} \cdot \mathsf{fst} \cdot r \cdot c)\ (\mathbf{B} \cdot \mathsf{snd} \cdot r \cdot c) = \mathsf{pair}\ (\mathsf{fst}\ (r \cdot c))\ (\mathsf{snd}\ (r \cdot c))$$

so that

$$h_s(z, c) = (h(z), \mathsf{pair}\ (\mathsf{fst}\ (r \cdot c))\ (\mathsf{snd}\ (r \cdot c)))$$

The argument would now be complete were our pairing surjective, but this is not so in general PCAs. However, we know that $c \in \|z\|_C$ for some $z \in |C|$, so $r \cdot c \in \|(x,y)\|_{X\times Y}$ with $h(z) = (x,y)$. Hence,

$$r \cdot c = \mathsf{pair}\ a\ b$$

for some $a \in \|x\|_X$ and $b \in \|y\|_Y$, and hence

$$s \cdot c = \mathsf{pair}\ a\ b$$

as well. It follows that $h_s = h_r$, where r was the original tracking element of (h,r). We have finally obtained that

$$\Box\langle \pi_1 \circ (h,r), \pi_2 \circ (h,r)\rangle \sim \Box(h,r)$$

The final thing to check is that any arrow into the terminal object $\mathbf{1}$ is intensionally equal to the canonical one; this follows, as $\mathbf{1}$ has only one element with a unique realizer.

$\Box$ is weakly cartesian closed

It is also the case that $\Box : \mathfrak{Asm}(A) \nrightarrow \mathfrak{Asm}(A)$ is *weakly cartesian closed* (§4.2.5), in the sense that the equation

$$\mathsf{ev} \circ (\lambda(f,r) \times id_X) \approx_\Box (f,r)$$

holds for any $(f,r) : C \times X \to Y$. To prove this one needs another tiring but easy calculation like the one showing that $\Box$ is cartesian: it is of no interest, save another use of the fact that it uses the trick that any $z \in \|(c,x)\|_{C\times X}$ is always of the form $z = \mathsf{pair}\ i\ j$ for $i \in \|c\|_C$ and $j \in \|x\|_X$.

Preservation of Products

Define

$$m_0 \stackrel{\text{def}}{=} (* \mapsto (*, \overline{0}), \mathbf{I}) : \mathbf{1} \to \Box\mathbf{1}$$

which maps the unique element of $\mathbf{1}$ to the pair of itself and its unique realizer, and is realized by the identity combinator. This is a P-isomorphism with inverse

$$!_{\Box\mathbf{1}} \stackrel{\text{def}}{=} ((*, \overline{0}) \mapsto *, \mathbf{I}) : \Box\mathbf{1} \to \mathbf{1}$$

Also, define $m_{A,B} : \Box A \times \Box B \to \Box(A \times B)$ by $m_{A,B} \stackrel{\text{def}}{=} (w_{A,B}, \mathbf{I})$, where

$$\begin{aligned} w_{A,B} : |\Box A| \times |\Box B| &\longrightarrow |\Box(A \times B)| \\ ((x,a),(y,b)) &\longmapsto ((x,y), \mathsf{pair}\ a\ b) \end{aligned}$$

The only realizer for the pair $((x,a),(y,b))$ is $\mathsf{pair}\ a\ b$, so the identity combinator tracks $w_{A,B}$. Then $\langle \Box\pi_1, \Box\pi_2 \rangle : \Box(A \times B) \to \Box A \times \Box B$ is equal to $(v_{A,B}, \mathbf{I})$, where

$$\begin{aligned} v_{A,B} : |\Box(A \times B)| &\longrightarrow |\Box A| \times |\Box B| \\ ((x,y),r) &\longmapsto ((x, \mathsf{fst} \cdot r), (y, \mathsf{snd} \cdot r)) \end{aligned}$$

and, as before, it is easy to see that this is an inverse to $m_{A,B}$, as r is necessarily of the form $\mathsf{pair}\ a\ b$. Hence,

Theorem 42. *The exposure $\Box : \mathfrak{Asm}(A) \looparrowright \mathfrak{Asm}(A)$ is product-preserving.*

Comonadicity and Idempotence

Proposition 9. *There exists a natural transformation of exposures,*

$$\epsilon : \Box \stackrel{\bullet}{\looparrowright} \mathsf{Id}_{\mathfrak{Asm}(A)}$$

Proof. Define $\epsilon_X = (u_X, \mathbf{I})$, where

$$\begin{aligned} u_X : |\Box X| &\to |X| \\ (x,a) &\longmapsto x \end{aligned}$$

If $b \in \|(x,a)\|_{\Box X} = \{a\}$, then $b = a$ and $\mathbf{I} \cdot b \simeq b = a \in \|x\|_X$, so u_X is indeed tracked by $\mathbf{I}$. To show naturality for $(f,r) : X \to Y$, we chase around the diagram:

$$\begin{array}{ccc} (x,a) & \xmapsto{f_r} & (f(x), r \cdot a) \\ {\scriptstyle u_X}\downarrow & & \downarrow{\scriptstyle u_Y} \\ x & \xmapsto[f]{} & f(x) \end{array}$$

Hence $u_Y \circ f_r = f \circ u_X$, so the square commutes up to $\sim$. Bear in mind that the tracking element along one composite is $\mathbf{B} \cdot r \cdot \mathbf{I}$, whereas along the other composite it is $\mathbf{B} \cdot \mathbf{I} \cdot r$. □

Let us now investigate the structure of $\Box^2 X$, for any assembly X. For each $(x,a) \in |\Box X|$, we have that $\|(x,a)\|_{\Box X} = \{a\}$. Thus, we can infer that

$$\left|\Box^2 X\right| = \{\, ((x,a),a) \mid a \in \|x\|_X \,\}$$

and that, for any $(f, r) : X \to Y$,

$$\begin{aligned} \Box^2 f : \left|\Box^2 X\right| &\longrightarrow \left|\Box^2 Y\right| \\ ((x,a),a) &\longmapsto ((f(x), r \cdot a), r \cdot a) \end{aligned}$$

Proposition 10. *There exists a natural transformation of exposures,*

$$\delta : \Box \overset{\bullet}{\looparrowright} \Box^2$$

Proof. Define $\delta_X = (v_X, \mathbf{I})$, where

$$\begin{aligned} v_X : |\Box X| &\longrightarrow \left|\Box^2 X\right| \\ (x,a) &\longmapsto ((x,a),a) \end{aligned}$$

If $a \in \|x\|_X$, then $\mathbf{I} \cdot a \simeq a \in \{a\} = \|((x,a),a)\|_{\Box^2 X}$, so $\mathbf{I}$ indeed tracks v_X. To show naturality for a given arrow $(f, r) : X \to Y$, we chase around the diagram:

$$\begin{array}{ccc} (x,a) & \xmapsto{f_r} & (f(x), r \cdot a) \\ {\scriptstyle v_X}\downarrow & & \downarrow{\scriptstyle v_Y} \\ ((x,a),a) & \xmapsto[(f_r)_r]{} & ((f(x), r \cdot a), r \cdot a) \end{array}$$

and so the diagram commutes up to $\sim$. Note that the tracking elements along the composites are respectively $\mathbf{B} \cdot \mathbf{I} \cdot r$ and $\mathbf{B} \cdot r \cdot \mathbf{I}$. □

It is not difficult to see that the components of $\delta : \Box \overset{\bullet}{\looparrowright} \Box^2$ are actually isomorphisms. Hence, putting everything together:

Theorem 43. $(\Box, \epsilon, \delta)$ *is a product-preserving idempotent comonadic exposure.*

Proof. It suffices to verify the coherence conditions. Regarding the first one:

$$\begin{array}{ccc} (x,a) & \xmapsto{v_X} & ((x,a),a) \\ {\scriptstyle v_X}\downarrow & & \downarrow{\scriptstyle v_{\Box X}} \\ ((x,a),a) & \xmapsto[(v_X)_{\mathbf{I}}]{} & (((x,a),a), \mathbf{I} \cdot a) = (((x,a),a),a) \end{array}$$

which commutes, since $\mathbf{I} \cdot a \simeq a$. Both the tracking elements along each composite are $\mathbf{B} \cdot \mathbf{I} \cdot \mathbf{I}$, so the diagram actually commutes on the nose. Regarding the second one:

$$\begin{array}{ccc} (x,a) & \xmapsto{v_X} & ((x,a),a) \\ {\scriptstyle v_X}\downarrow & & \downarrow{\scriptstyle u_{\Box X}} \\ ((x,a),a) & \xmapsto[(u_X)_{\mathbf{I}}]{} & (x, \mathbf{I} \cdot a) = (x,a) \end{array}$$

which commutes, since $\mathbf{I} \cdot a \simeq a$. Once more, both tracking elements are $\mathbf{B} \cdot \mathbf{I} \cdot \mathbf{I}$, so commutation is again on the nose. □

5.2.5 Weak Extensionality and Naturality

We have shown that nearly everything in sight behaves well, even with respect to intensional equality $\approx_\Box$: Proposition 5 necessitates that, when we have a product-preserving (and hence cartesian) exposure, the function

$$\langle \cdot, \cdot \rangle : \mathfrak{Asm}(A)(C, X) \times \mathfrak{Asm}(C, Y) \to \mathfrak{Asm}(A)(C, X \times Y)$$

that is implicated in the definition of products respects intensional equality $\approx_\Box$. The glaring exception, of course, is the cartesian closed structure: it is not necessarily that

$$\lambda_C : \mathfrak{Asm}(A)(C \times X, Y) \to \mathfrak{Asm}(A)(C, Y^X)$$

preserves intensional equality.

However, it is interesting to investigate when this might happen. Let $(f, r) \approx_\Box (f, s) : C \times X \to Y$ be two intensionally equal morphisms; we have

$$\forall d \in \|(c, x)\|_{C \times X} \,.\, r \cdot d \simeq s \cdot d \tag{5.1}$$

We can calculate that

$$\lambda(f, r) \stackrel{\text{def}}{=} (\lambda(f), \lambda^* c\, a.\ r(\mathsf{pair}\ c\ a))$$
$$\lambda(f, s) \stackrel{\text{def}}{=} (\lambda(f), \lambda^* c\, a.\ s(\mathsf{pair}\ c\ a))$$

Thus, to prove that $\lambda(f, r) \approx_Q \lambda(f, s)$, all we need to check is that

$$\forall d \in \|c\|_C \,.\, \lambda^* a.\ r(\mathsf{pair}\ c\ a) \simeq \lambda^* a.\ s(\mathsf{pair}\ c\ a)$$

By (5.1), it is indeed the case that $r(\mathsf{pair}\ c\ a) \simeq s(\mathsf{pair}\ c\ a)$, because the realizers $d \in \|(c, x)\|_{C \times X}$ are exactly of the right form. Nevertheless, *we are not allowed to use that equation* under an occurrence of λ^*! The situation that allows this is the one where the PCA A is *weakly extensional*.

Definition 41. A PCA $(A, \cdot)$ is *weakly extensional* if it satisfies the rule

$$\frac{M \simeq N}{\lambda^* x.\ M \simeq \lambda^* x.\ N}$$

for any two expressions M, N and any variable x.

Weak extensionality, also known as rule (ξ), is a notorious thorn in the study of the correspondence between combinatory logic and untyped λ-calculus. [Barendregt, 1984, §7.3.5(iii)] sets out a finite set of equational axioms (due to Curry) that suffice to ensure it.

The original plan for this thesis was that $\mathfrak{Asm}(K_1)$ would be a model of Intensional PCF, and Löb's rule would directly correspond to Kleene's Second Recursion Theorem. Unfortunately, K_1 is very likely *not* weakly extensional. In §8 we will develop a slight restriction on Intensional PCF, which will remove the requirement that $\lambda(-)$ preserve intensional equality, which is otherwise necessary.

Since we have come this far, let us also investigate the *naturality* of $\lambda(-)$, i.e. the equation

$$\lambda\left(f \circ (g \times id)\right) \approx_{\Box} \lambda(f) \circ g$$

under intensional equality. Given $(f, r) : C \times X \to Y$ and $(g, s) : C' \to C$, we compute that

$$\lambda\left((f,r) \circ ((g,s) \times id)\right) \stackrel{\text{def}}{=} \left(\lambda\left(f \circ (g \times id)\right), \lambda^* c'\, a.\ (\mathbf{B} \cdot r \cdot h) \cdot (\mathsf{pair}\ c'\ a)\right)$$
$$\lambda(f,r) \circ (g,s) \stackrel{\text{def}}{=} (\lambda(f) \circ g, \mathbf{B} \cdot (\lambda^* c'\ a.\ r \cdot (\mathsf{pair}\ c'\ a)) \cdot s)$$

where $h \stackrel{\text{def}}{=} \lambda^* d.\ \mathsf{pair}\ (\mathbf{B} \cdot s \cdot \mathsf{fst} \cdot d)\ (\mathbf{B} \cdot \mathbf{I} \cdot \mathsf{snd} \cdot d)$. It would suffice to prove that these two realizers, when applied to anything, would return the same c', namely

$$\lambda^*.\ r \cdot (\mathsf{pair}\ (s \cdot c')\ a)$$

This is easy to check with weak extensionality, but it seems impossible to ensure without it. Therefore, $\lambda(-)$ is natural up to $\approx_{\Box}$ if A is weakly extensional.

Finally, let us seize the opportunity to mention that if A is *extensional*, in the sense that the η-rule

$$\lambda^* x.\ e\ x \simeq e$$

holds for any expression $e \in \mathcal{E}(A)$, then

$$\mathsf{ev} \circ ((f,r) \circ id) \approx_Q (f,r) : C \to Y^X$$

In that case we would say that $\Box : \mathfrak{Asm}(A) \to\!\!\!\!\!\to \mathfrak{Asm}(A)$ is *cartesian closed*.

5.3 Exposures in Homological Algebra

This section aims to support the claim that, even though inspired by logic and computability, exposures are to be found in other contexts as well. This lends credibility to the idea that, even if within logic exposures are a sort of abstract yet well-behaved Gödel numbering, the phenomenon of *intensionality*, as discussed in §1.1 is more general, and can be found in other areas of mathematics.

We will draw our example from *homological algebra*. Homological algebra begins once we have *chain complexes* $C(X)$, i.e. sequences of abelian groups C_i with homomorphisms

$$\ldots \xrightarrow{\partial_{n+1}} C_n \xrightarrow{\partial_n} C_{n-1} \xrightarrow{\partial_{n-1}} \ldots \xrightarrow{\partial_1} C_1 \xrightarrow{\partial_0} C_0$$

such that $\partial_d \circ \partial_{n+1} = 0$. One then forms the groups of *boundaries* and *cycles*, namely

$$B_n(X) \stackrel{\text{def}}{=} im(\partial_{n+1}) \qquad Z_n(X) \stackrel{\text{def}}{=} ker(\partial_n)$$

The objects of study are then the *homology groups* H_n, defined by

$$H_n(X) \stackrel{\text{def}}{=} Z_n(X)/B_n(X)$$

A natural question arises: what if we make it so we never *actually* have to take quotients?

5.3.1 The P-category $\mathfrak{Grp}$

Instead of taking the quotient G/N of a group G by one of its normal subgroups N, we will instead merely keep the tuple

$$(G, N)$$

and work with it: we will consider this as G/N, even though the two components are kept separately.

Often in homology one considers maps between homology groups, which are group homomorphisms of type

$$f_* : G_1/H_1 \to G_2/H_2$$

Nevertheless, rarely does one work out G_1/H_1 exactly before defining such a f_*. More commonly, one picks out a representative of the equivalence class, defines f on that,

and then proves that the outcome is invariant under the choice of representative. This amounts to defining some $f : G_1 \to G_2$ such that

$$f(H_1) \subseteq H_2$$

Thus, we pick the morphisms $f : (G_1, H_1) \to (G_2, H_2)$ to be exactly those homomorphisms. Each one of them induces a homomorphism $f_* : G_1/H_1 \to G_2/H_2$ as is customary.

To prove that two such maps $f_*, g_* : G_1/H_1 \to G_2/H_2$ are equal, it suffices to prove that they are pointwise *homologous*, i.e. that that $f - g$ takes values only in H_2. This will be exactly our definition of extensional equality:

$$f \sim g : (G_1, H_1) \to (G_2, H_2) \quad \text{just if} \quad im(f - g) \subseteq H_2$$

A classic result of basic group theory, viz.

$$G_1/N_1 \times G_2/N_2 \cong (G_1 \times G_2)/(N_1 \times N_2)$$

also implies that we can define

$$(G_1, N_1) \times (G_2, N_2) \stackrel{\text{def}}{=} (G_1 \times G_2, N_1 \times N_2)$$

and use it to prove that

Theorem 44. *The P-category* $\mathfrak{Grp}$ *is cartesian.*

We can now think of the n-th homology functor as taking values in this P-category, or—even better—its subcategory $\mathfrak{Ab}$ of *abelian groups*,

$$\begin{aligned} H_n : \mathbf{Top} &\longrightarrow \mathfrak{Ab} \\ X &\longmapsto (Z_n(X), B_n(X)) \\ f : X \to Y &\longmapsto f_\# : (Z_n(X), B_n(X)) \to (Z_n(Y), B_n(Y)) \end{aligned}$$

5.3.2 Intensionality and Homomorphisms

The point of not forcing f to be f_* is that the action of $f : (G_1, H_1) \to (G_2, H_2)$ on each cycle of G_1 is 'visible,' *even if that cycle is a boundary.* This is because $f : G_1 \to G_2$ is still an actual homomorphism, which only happens to 'respect' a normal subgroup. The way we will define an exposure on this category is precisely by 'exposing' the action of f on individual cycles, *even if they are boundaries.*

We shall then define an endoexposure,

$$C : \mathfrak{Grp} \nrightarrow \mathfrak{Grp}$$

by

$$\begin{aligned} C(G,H) &\stackrel{\text{def}}{=} (G, \{e_G\}) \\ Cf &\stackrel{\text{def}}{=} f : (G_1, \{e_{G_1}\}) \to (G_2, \{e_{G_2}\}) \end{aligned}$$

It is not at all difficult to prove that this is indeed an exposure: it preserves composition and identities, and indeed if $Cf \sim Cg$ then f and g are equal, so $im(f-g)$ is just the identity element.

In fact, this exposure has comonadic structure, which comes for free. For evaluators, we notice that $\epsilon_{(G,H)} : C(G,H) \to (G,H)$ is actually of type $(G, \{e_G\}) \to (G,H)$, so it suffices to take the identity, which—in this context—is a kind of quotient map. Similarly, $\delta_{(G,H)} : (G, \{e_G\}) \to (G, \{e_G\})$, so again it suffices to take the identity. It is trivial that every single diagram in the definition of evaluator and quoter, as well as that of comonadic exposure, commutes: all the arrows are identities.

Furthermore, it is not hard to see that C preserves products: the candidate isomorphisms

$$m : Q(G_1, H_1) \times Q(G_2, H_2) \to Q(G_1 \times G_2, H_1 \times H_2)$$

have the same source and target, namely $(G_1 \times G_2, \{(e_{G_1}, e_{G_2})\})$, so it suffices to take the identity.

Chapter 6

Intensional Recursion in P-Categories[1]

Armed with the framework of exposures, we can now speak of both extensional and intensional recursion in categorical terms.

The case of *extensional fixed points (EFPs)* was first treated by Lawvere [1969, 2006] in the late 1960s. However, we will argue that his notion of fixed point is far too coarse for most applications in logic and computer science.

Instead, we will use exposures to replace that definition with one that captures intensional recursion, namely that of *intensional fixed points (IFPs)* (§6.1). We begin our investigation by showing that our framework allows for clear and concise formulations of the classic theorems of Gödel, Tarski, and Rice. The relevant arguments are entirely algebraic, and it is very clear what logical devices or assumptions each one requires. The conclusion to be drawn is that, despite their common use of IFPs, these three arguments have a fundamentally different flavour. In (§6.2) we discuss the relationship between IFPs and Löb's rule in provability logic.

Then, in §6.3, we then ask the natural question: where do IFPs come from? We recall in detail a theorem of Lawvere which guarantees the existence of EFPs under certain assumptions. We use exposures to prove the Intensional Recursion Theorem, a similar theorem that pertains to IFPs instead.

Finally, we examine the nature of both EFPs and IFPs in the three examples that we presented at length in §5. In particular, when viewed through the lens of the exposure on assemblies (§5.2), Lawvere's theorem and our Intensional Recursion Theorem are revealed to be categorical versions of the First and Second Recursion Theorems of Kleene respectively, as discussed in §2.

[1] A preliminary form of the results in this chapter was first published as [Kavvos, 2017a], which is available at Springer: `https://dx.doi.org/10.1007/978-3-662-54458-7_32`

6.1 Extensional and Intensional Fixed Points

Lawvere [1969, 2006] famously proved a theorem which guarantees that, under certain assumptions, which we discuss in §6.3, there exist fixed points of the following sort.

Definition 42. An *extensional fixed point (EFP)* of an arrow $t : Y \to Y$ is a point $y : \mathbf{1} \to Y$ such that

$$t \circ y = y$$

If every arrow $t : Y \to Y$ has a EFP, then we say that Y *has EFPs.*

In Lawvere's paper EFPs are a kind of fixed point that, for logical purposes, *oughtn't* exist. After constructing a category based on a logical theory (e.g. PA), in a manner that we have quite closely followed in §5.1, he argues that there can be no

$$\mathrm{sat} : A \times A \to \mathbf{2}$$

such that for every formula $\phi : A \to \mathbf{2}$ there is a point $c_\phi : \mathbf{1} \to A$ such that

$$\begin{array}{ccc} \mathbf{1} & \xrightarrow{a} & A \\ {\scriptstyle \langle a, c_\phi \rangle}\downarrow & & \downarrow{\scriptstyle \phi} \\ A \times A & \xrightarrow[\mathrm{sat}]{} & \mathbf{2} \end{array}$$

for every point $a : \mathbf{1} \to A$. In logical terms, this would amount to the existence of a two-variable predicate $\mathrm{sat}(-,-)$, and a Gödel number $\ulcorner \phi \urcorner$ for each unary predicate $\phi(x)$, such that

$$\mathsf{T} \vdash \mathrm{sat}(\ulcorner \phi(x) \urcorner, n) \leftrightarrow \phi(n)$$

for each n. If such a predicate existed, then *'satisfaction would be definable,'* and we would obtain a EFP of the arrow $\mathbf{2} \to \mathbf{2}$ encoding the logical 'not' function. In logical terms, we could obtain a closed formula ψ such that $\mathsf{T} \vdash \psi \leftrightarrow \neg\psi$. This leads to a categorical version of Tarski's *Undefinability Theorem*: if 'truth were definable,' then substitution would be too, and $\neg$ would have a fixed point.

Finally, Lawvere obtains a version of Gödel's *First Incompleteness Theorem* as follows: given an (external) relation between closed formulas and points, relating closed formulas to their 'Gödel number,' if we assume that provability is 'internally decidable' on these Gödel numbers, and if we assume that all 'truth values' are either true or false, then truth would be definable, which—by the categorical version of Tarski's undefinability theorem—it is not.

We can already see that Lawvere's notion of EFPs do not encompass fixed points that *ought* to exist. For example, the *diagonal lemma* for *Peano Arithmetic* (henceforth PA) manufactures a closed formula $\mathbf{fix}(\phi)$ for every formula $\phi(x)$, such that

$$\mathsf{PA} \vdash \mathbf{fix}(\phi) \leftrightarrow \phi(\ulcorner \mathbf{fix}(\phi) \urcorner)$$

The formula $\mathbf{fix}(\phi)$ occurs asymmetrically: on the left hand side of the bi-implication it appears as a truth value, but on the right hand side it appears under a *Gödel numbering*, i.e. an assignment $\ulcorner \cdot \urcorner$ of a numeral to each term and formula of PA. Taking our cue from the exposure on Peano arithmetic (§5.1), we can generalise this idea to the following, which encompasses this kind of 'asymmetric' fixed point.

Definition 43. Let $Q : \mathfrak{B} \nrightarrow \mathfrak{B}$ be a product-preserving endoexposure. An *intensional fixed point (IFP)* (w.r.t. to Q) of an arrow $t : QA \to A$ is a point $a : \mathbf{1} \to A$ such that the following diagram commutes up to $\sim$:

$$\begin{array}{ccc} \mathbf{1} & \xrightarrow{a} & A \\ {\scriptstyle m_0}\downarrow & & \uparrow{\scriptstyle t} \\ Q\mathbf{1} & \xrightarrow[Qa]{} & QA \end{array}$$

An object A *has IFPs (w.r.t. Q)* if every arrow $t : QA \to A$ has a IFP.

This makes intuitive sense: $a : \mathbf{1} \to A$ is extensionally equal to t 'evaluated' at the point $Qa \circ m_0 : \mathbf{1} \to Qa$, which is the 'quoted' version of a.

6.1.1 Consistency, Truth and Provability: Gödel and Tarski

We are now in a position to argue that the two well-known theorems that were discussed by Lawvere can be reduced to very simple algebraic arguments involving exposures. In fact, the gist of both arguments relies on the existence of IFPs for an 'object of truth values' in a P-category. For background in Gödel's First Incompleteness Theorem and Tarski's *Undefinability Theorem*, see Smullyan [1992] and Boolos [1994].

Suppose that we have some sort of object $\mathbf{2}$ of 'truth values.' This need not be fancy: we require that it has two points,

$$\top : \mathbf{1} \to \mathbf{2}$$
$$\bot : \mathbf{1} \to \mathbf{2}$$

standing for truth and falsehood respectively. We also require an arrow $\neg : \mathbf{2} \to \mathbf{2}$ that encodes the logical negation, satisfying

$$\neg \circ \top \sim \bot$$
$$\neg \circ \bot \sim \top$$

and

$$\neg \circ f \sim \bot \quad \Longrightarrow \quad f \sim \top$$

A simplified version of Gödel's First Incompleteness theorem for PA is this:

Theorem 45 (Gödel)**.** *If PA is consistent, then there are sentences ϕ of PA such that neither $\mathsf{PA} \vdash \phi$ nor $\mathsf{PA} \vdash \neg\phi$.*

The proof relies on two constructions: the diagonal lemma, and the fact that provability is definable within the system. The definability of provability amounts to the fact that there is a formula $\mathrm{Prov}(x)$ with one free variable x such that

$$\mathsf{PA} \vdash \phi \quad \text{if and only if} \quad \mathsf{PA} \vdash \mathrm{Prov}(\ulcorner \phi \urcorner)$$

That is: modulo Gödel numbering, the system can internally 'talk' about its own provability. It is not then hard to sketch the proof to Gödel's theorem.

Proof of Theorem 45. Use the diagonal lemma to construct ψ such that

$$\mathsf{PA} \vdash \psi \leftrightarrow \neg\mathrm{Prov}(\ulcorner \psi \urcorner)$$

Then ψ is provable if and only if it is not, so if either $\mathsf{PA} \vdash \psi$ or $\mathsf{PA} \vdash \neg\psi$ we would observe inconsistency. Thus, if PA is consistent, neither ψ nor $\neg\psi$ are provable. □

It follows that ψ is not equivalent to either truth value. In a way, ψ has some other eerie truth value, which is neither $\top$ nor $\bot$. Classical logicians would say that it is *undecidable*.

Let us represent the provability predicate as an arrow $p : Q\mathbf{2} \to \mathbf{2}$ such that $y \sim \top$ if and only if $p \circ Qy \circ m_0 \sim \top$. Consistency is captured by the following definition:

Definition 44. An object of truth values $\mathbf{2}$ as above is *simply consistent* just if

$$\top \not\sim \bot$$

Armed with this machinery, we can now transport the argument underlying Gödel's proof to our more abstract setting.

Theorem 46. *If a $p : Q\mathbf{2} \to \mathbf{2}$ is as above, and $\mathbf{2}$ has IFPs, then one of the following things is true:*

- *either there are points of $\mathbf{2}$ other than $\top : \mathbf{1} \to \mathbf{2}$ and $\bot : \mathbf{1} \to \mathbf{2}$, or*
- *$\mathbf{2}$ is* not *simply consistent, i.e. $\top \sim \bot$.*

Proof. As $\mathbf{2}$ has IFPs, take $y : \mathbf{1} \to \mathbf{2}$ such that

$$y \sim \neg \circ p \circ Qy \circ m_0$$

Now, if $y \sim \top$, then by the property of p above, $p \circ Qy \circ m_0 \sim \top$, hence $\neg \circ p \circ Qy \circ m_0 \sim \bot$, hence $y \sim \bot$. So either $y \not\sim \top$ or $\mathbf{2}$ is not simply consistent. Similarly, either $y \not\sim \bot$ or $\mathbf{2}$ is not simply consistent. □

Tarski's *Undefinability Theorem*, on the other hand is the result that *truth cannot be defined in arithmetic* [Smullyan, 1992].

Theorem 47 (Tarski)**.** *If PA is consistent, then there is no predicate True(x) such that*

$$PA \vdash \phi \leftrightarrow True(\ulcorner \phi \urcorner)$$

for all sentences ϕ.

Proof. Use the diagonal lemma to obtain a closed ψ such that

$$\mathsf{PA} \vdash \psi \leftrightarrow \neg \mathrm{True}(\ulcorner \psi \urcorner)$$

Then $\mathsf{PA} \vdash \psi \leftrightarrow \neg\psi$, which leads to inconsistency. □

A truth predicate would constitute an evaluator $\epsilon : Q \overset{\bullet}{\nrightarrow} \mathsf{Id}_{\mathfrak{B}}$. If we had one, we would have that

$$\epsilon_{\mathbf{2}} \circ Q(y) \circ m_0 \sim y \circ \epsilon_{\mathbf{1}} \circ m_0 \sim y$$

where the last equality is because $\mathbf{1}$ is terminal. This is actually a more general

Lemma 24. *Let $Q : \mathfrak{B} \nrightarrow \mathfrak{B}$ be an endoexposure, and let $\epsilon : Q \overset{\bullet}{\nrightarrow} Id_{\mathfrak{B}}$ be an evaluator. Then, if A has IFPs then it also has EFPs.*

Proof. Given $t : A \to A$, consider $t \circ \epsilon_A : QA \to A$. A IFP for this arrow is a point $y : \mathbf{1} \to A$ such that $y \sim t \circ \epsilon_A \circ Qy \circ m_0 \sim t \circ y$. □

In proving Tarski's theorem, we constructed a sentence ψ such that $\mathsf{PA} \vdash \psi \leftrightarrow \neg\psi$. This can be captured abstractly by the following definition.

Definition 45. An object $\mathbf{2}$ as above is *fix-consistent* just if the arrow $\neg : \mathbf{2} \to \mathbf{2}$ has no EFP: that is, there is no $y : \mathbf{1} \to \mathbf{2}$ such that $\neg \circ y \sim y$.

Putting these together, we get

Theorem 48. *If* $\mathbf{2}$ *has IFPs in the presence of an evaluator, then it is* not *fix-consistent.*

6.1.2 Rice's theorem

To further illustrate the applicability of the language of exposures, we state and prove an abstract version of *Rice's theorem.* Rice's theorem is a result in computability which states that no computer can decide any non-trivial property of a program by looking at its code. A short proof relies on the SRT.

Theorem 49 (Rice)**.** *Let* $\mathcal{F}$ *be a non-trivial set of partial recursive functions, and let* $A_{\mathcal{F}} \stackrel{def}{=} \{\, e \in \mathbb{N} \mid \phi_e \in \mathcal{F} \,\}$ *be the set of indices of functions in that set. Then* $A_{\mathcal{F}}$ *is undecidable.*

Proof. Suppose $A_{\mathcal{F}}$ is decidable. The fact $\mathcal{F}$ is non-trivial means that there is some $a \in \mathbb{N}$ such that $\phi_a \in \mathcal{F}$ and some $b \in \mathbb{N}$ such that $\phi_b \notin \mathcal{F}$. Consequently, $a \in A_{\mathcal{F}}$ and $b \notin A_{\mathcal{F}}$.

Define $f(e, x) \simeq \textbf{if } e \in A_{\mathcal{F}} \textbf{ then } \phi_b(x) \textbf{ else } \phi_a(x)$. By Church's thesis, $f : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$ is partial recursive. Use the SRT to obtain $e \in \mathbb{N}$ such that $\phi_e(x) \simeq f(e, x)$. Now, either $e \in A_{\mathcal{F}}$ or not. If it is, $\phi_e(x) \simeq f(e, x) \simeq \phi_b(x)$, so that $\phi_e \notin \mathcal{F}$, a contradiction. Similarly if $e \notin A_{\mathcal{F}}$. □

Constructing the function f in the proof required three basic elements: (a) the ability to evaluate either ϕ_a or ϕ_b given a and b; (b) the ability to decide which one to use depending on the input; and (c) intensional recursion. For (a), we shall need evaluators, for (b) we shall need that the truth object $\mathbf{2}$ is a *weak coproduct* of two copies of $\mathbf{1}$, and for (c) we shall require IFPs.

Theorem 50. *Let* $\mathbf{2}$ *be a simply consistent truth object which also happens to be a a weak coproduct of two copies of* $\mathbf{1}$*, with injections*

$$\top : \mathbf{1} \to \mathbf{2}$$

$$\bot : \mathbf{1} \to \mathbf{2}$$

Suppose that A *has EFPs. If* $f : A \to \mathbf{2}$ *is such that for all* $x : \mathbf{1} \to A$*, either* $f \circ x \sim \top$ *or* $f \circ x \sim \bot$*. Then* f *is trivial, in the sense that either*

$$\forall x : \mathbf{1} \to A.\ f \circ x \sim \top \qquad or \qquad \forall x : \mathbf{1} \to A.\ f \circ x \sim \bot$$

Proof. Suppose that there are two such distinct $a, b : \mathbf{1} \to A$ such that $f \circ a \sim \top$ and $f \circ b \sim \bot$. Let

$$g \stackrel{\text{def}}{=} [b, a] \circ f : A \to A$$

and let $y : \mathbf{1} \to A$ be its EFP. Now, either $f \circ y \sim \top$ or $f \circ y \sim \bot$. In the first case, we can calculate that

$$\top \sim \ f \circ y \sim f \circ g \circ y \sim \ f \circ [b, a] \circ f \circ y \sim \ f \circ [b, a] \circ \top \sim \ f \circ b \sim \ \bot$$

so that $\mathbf{2}$ is *not simply consistent*. A similar situation occurs if $f \circ y \sim \bot$. □

Needless to say that the premises of this theorem are easily satisfied in our exposure on assemblies from §5.2 if we take $A = {\mathbb{N}_\bot}^{\mathbb{N}}$ and $\mathbf{2}$ to be the lifted coproduct $(\mathbf{1}+\mathbf{1})_\bot$.

6.2 The relationship to Löb's rule

In this chapter we have considered two sorts of fixed points, *extensional* and *intensional*. Figure 6.1 summarises the definition of these two types of fixed points, in 1-category theory and P-category theory respectively.

Figure 6.1: Types of Fixed Points (without parameters)

Type	Morphism	Fixed Point
Extensional	$t : A \to A$	$a : \mathbf{1} \to A$ $t \circ a = a$
Intensional	$t : QA \to A$	$a : \mathbf{1} \to A$ $t \circ Qa \circ m_0 \sim a$

Viewed through the lens of the Curry-Howard isomorphism, the existence of extensional fixed points at A can be written as the logical inference rule

$$\frac{A \to A}{A}$$

which is exactly the type of the Y combinator of PCF. In fact, this exact inference rule corresponds to an equivalent formulation of PCF that proceeds through the binding construct $\mu x{:}A.M$ with equation

$$\mu x{:}A.M = M[\mu x{:}A.M]$$

For more details, see Gunter [1992].

This rule is obviously logically catastrophic, as it produces a closed term $\mu x{:}A.\ x$ at each type A. Consequently, there is no honest Curry-Howard isomorphism for PCF: every type is inhabited, and thus every 'formula' is provable, leading to the trivial logic. However, *the terms do matter, because they have computational behaviour*: it might be that the types do not correspond to logical formulae, but they are there to stop basic programming errors. And, in the end, the purpose of PCF is simply typed general recursive programming.

Viewing the notion of intensional fixed points through the lens of Curry-Howard, the result is much more impressive: IFPs correspond to *Löb's rule*, namely

$$\frac{\Box A \to A}{A}$$

To see this, it suffices to read $\Box \stackrel{\text{def}}{=} Q$ and to look at the definition of intensional fixed points as an inference rule, i.e.

$$\frac{f : QA \to A}{f^\circ : \mathbf{1} \to A}$$

such that

$$\begin{array}{ccc} \mathbf{1} & \xrightarrow{Qf^\circ \circ m_0} & QA \\ {\scriptstyle f^\circ}\downarrow & \swarrow_{f} & \\ A & & \end{array}$$

commutes up to $\sim$.

Unfortunately, we have seen in this chapter that a key ingredient in the theory of exposures are the so-called evaluators, which are natural transformations $\epsilon : Q \overset{\bullet}{\looparrowright} \mathsf{Id}$. These encapsulate the modal axiom T, namely

$$\Box A \to A$$

Coupled with the above inference rule, this will also have the catastrophic consequence that every type is inhabited. We shall not be alarmed by this fact, for we will want general recursion, and there seems no way around partiality in that case, as we discussed in §1.2.4. We shall still use the Curry-Howard isomorphism, but only *heuristically*.

However, let us for the moment revert to the mindset of a purely categorical logician. In our parallel work [Kavvos, 2017b,c] we have investigated an extension of the Curry-Howard isomorphism to box modalities. Our methodology consisted of mimicking the rules of sequent calculus in natural deduction. A quick perusal suffices

to bring to light the fact that in that work we used a *stronger form of Löb's rule*, namely

$$\frac{\Box A \to A}{\Box A}$$

Historically speaking, this variant of Löb's rule is proof-theoretically well-behaved, and yields cut-free sequent calculi. It was introduced in the context of intuitionistic modal logic by Ursini [1979a]. If we were to use it to formulate iPCF,[2] we would obtain something akin to

$$\frac{\Delta\,;\, z : \Box A \vdash M : A}{\Delta\,;\,\Gamma \vdash \mathsf{fix}\ z\ \mathsf{in}\ \mathsf{box}\ M : \Box A}$$

with

$$\mathsf{fix}\ z\ \mathsf{in}\ \mathsf{box}\ M \longrightarrow \mathsf{box}\ M[\mathsf{fix}\ z\ \mathsf{in}\ \mathsf{box}\ M/z]$$

However, this does not have a clear intensional interpretation. If, as in §1.4.2, we try to devise a 'reading' of this in the untyped λ-calculus, this rule would require that for each term f there exists a u such that

$$u =_\beta \ulcorner f \ulcorner u \urcorner \urcorner$$

which is obviously nonsense. This is why in this thesis we have *weakened* our formulation to Löb's original rule.

Nevertheless, we can still 'transport' this version of Löb's rule over to P-categories: it amounts to the inference rule

$$\frac{f : QA \to A}{f^\dagger : \mathbf{1} \to QA}$$

such that

$$\begin{array}{ccc} \mathbf{1} & \xrightarrow{Qf^\dagger \circ m_0} & Q^2 A \\ {\scriptstyle f^\circ}\downarrow & \swarrow_{Qf} & \\ QA & & \end{array}$$

commutes up to $\sim$.

Now: can we use the framework of comonadic exposures to *prove* that this version of Löb's rule is stronger? What is the exact relationship between this and the original form?

Surprisingly, the answer is positive. But before we show that, let us give a name to these two kinds of IFPs so we can talk about them efficiently.

[2] In fact, the first versions of this thesis and [Kavvos, 2017d] both used this version of iPCF.

Definition 46. Let $Q : \mathfrak{B} \nrightarrow \mathfrak{B}$ be a product-preserving endoexposure.

1. A *meek intensional fixed point (meek IFP)* of an arrow $f : QA \to A$ is a point $a : \mathbf{1} \to a$ such that the following diagram commutes up to $\sim$:

$$\begin{array}{ccc} \mathbf{1} & \xrightarrow{a} & A \\ {\scriptstyle m_0}\downarrow & & \uparrow{\scriptstyle f} \\ Q\mathbf{1} & \xrightarrow[Qa]{} & QA \end{array}$$

   i.e. $a \sim f \circ Qa \circ m_0$.

2. A *vehement intensional fixed point (vehement IFP)* of an arrow $f : QA \to A$ is a point $a : \mathbf{1} \to QA$ such that the following diagram commutes up to $\sim$:

$$\begin{array}{ccc} \mathbf{1} & \xrightarrow{a} & QA \\ {\scriptstyle m_0}\downarrow & & \uparrow{\scriptstyle Qf} \\ Q\mathbf{1} & \xrightarrow[Qa]{} & Q^2A \end{array}$$

   i.e. $a \sim Qf \circ Qa \circ m_0$.

Thus the intensional fixed points we have been working with up to this point are *meek*, whereas the proof theory in [Kavvos, 2017b,c] use a pattern closer to *vehement* ones.

If we have a meek IFP whose defining equation holds up to intensional equality ($\approx_Q$), then that is also a vehement IFP. Conversely, if we have a vehement IFP, and our comonadic exposure is idempotent, then we obtain a meek IFP.

Theorem 51. *Let $f : QA \to A$.*

1. *If we have a meek IFP of $f : QA \to A$, which moreover is so intensionally, i.e.*

$$f^\circ \approx_Q f \circ Qf^\circ \circ m_0$$

   then we can obtain a vehement IFP of f, defined by

$$f^\dagger \stackrel{def}{=} Qf^\circ \circ m_0$$

2. *If we have a vehement IFP $f : QA \to A$, and moreover the comonadic exposure (Q, ϵ, δ) is idempotent, then we can obtain a meek IFP of of f, defined by*

$$f^\circ \stackrel{def}{=} \epsilon_A \circ f^\dagger$$

Proof.

1. We calculate:

$$\begin{array}{ll} & f^\dagger \\ \sim & \{\,\text{definition}\,\} \\ & Qf^\circ \circ m_0 \\ \sim & \{\,\text{assumption}\,\} \\ & Q(f \circ Qf^\circ \circ m_0) \circ m_0 \\ \sim & \{\,\text{exposure}\,\} \\ & Qf \circ Q(Qf^\circ \circ m_0) \circ m_0 \\ \sim & \{\,\text{definition}\,\} \\ & Qf \circ Qf^\dagger \circ m_0 \end{array}$$

   so that $f^\dagger$ is a vehement IFP.

2. We calculate

$$\begin{array}{ll} & f^\circ \\ \sim & \{\,\text{definition}\,\} \\ & \epsilon_A \circ f^\dagger \\ \sim & \{\,\text{definition}\,\} \\ & \epsilon_A \circ Qf \circ Qf^\dagger \circ m_0 \\ \sim & \{\,\epsilon \text{ natural}\,\} \\ & f \circ \epsilon_{QA} \circ Qf^\dagger \circ m_0 \\ \sim & \{\,\epsilon \text{ natural}\,\} \\ & f \circ f^\dagger \circ \epsilon_{\mathbf{1}} \circ m_0 \\ \sim & \{\,\epsilon \text{ monoidal}\,\} \\ & f \circ f^\dagger \end{array}$$

   But, by the corollary of the Quotation-Evaluation lemma (Lemma 4),

$$f^\dagger \sim Q(\epsilon_A \circ f^\dagger) \circ m_0 \sim Qf^\circ \circ m_0$$

   so $f^\circ \sim f \circ Qf^\circ \circ m_0$ is a meek IFP.

□

6.3 Whence fixed points?

6.3.1 Lawvere's Theorem

Lawvere [1969, 2006] proved a fixed point theorem that generalises a number of 'diagonal' constructions, including Cantor's theorem, Gödel's First Incompleteness Theorem, and the Tarski undefinability theorem. In order to state it, we will first need some notation for cartesian closed categories (CCCs). A cartesian closed category [Eilenberg and Kelly, 1966] is a great place: it is a mathematical universe where morphisms of the category correspond exactly to points of certain objects, the *exponentials*. Indeed, to every morphism $f : A \to Y$ there corresponds a point of the exponential object Y^A, namely

$$\ulcorner f \urcorner : \mathbf{1} \to Y^A$$

which is defined by

$$\ulcorner f \urcorner \stackrel{\text{def}}{=} \lambda\left(\mathbf{1} \times A \stackrel{\cong}{\to} A \stackrel{f}{\to} Y\right)$$

and to each point $y : \mathbf{1} \to Y^A$ there corresponds a morphism $y^\wr : A \to Y$, defined by

$$y^\wr \stackrel{\text{def}}{=} A \xrightarrow{\langle y \circ !_A, id_A \rangle} Y^A \times A \stackrel{\mathsf{ev}}{\to} Y$$

The above operations are mutually inverse:

$$\ulcorner f \urcorner^\wr = f, \qquad \ulcorner y^\wr \urcorner = y$$

For a classic exposition, see Lambek and Scott [1988].

The main idea in Lawvere's paper is this: a morphism

$$X \stackrel{r}{\to} Y^A$$

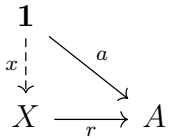
from an object to an exponential may be thought as 'indexing' morphisms of type $A \to Y$; for, given $x : \mathbf{1} \to X$, we have $(r \circ x)^\wr : A \to Y$. Such an indexing may be considered an *enumeration* if it is, in some sense, *surjective*. There are many ways in which an arrow $r : X \to A$ can be surjective. Here are four:

- It could a *retraction*; that is, there could exist $s : A \to X$ such that

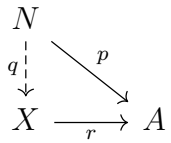
$$\begin{array}{ccc} A & & \\ {\scriptstyle s}\downarrow & \searrow^{id_A} & \\ X & \xrightarrow[r]{} & A \end{array}$$

  The arrow s may be thought of as 'choosing a preimage' of elements of A with respect to r. In the category of sets, surjective functions are always retractions (if one assumes the axiom of choice).

- It could be *point-surjective*: for each point $a : \mathbf{1} \to A$ there could be a point $x : \mathbf{1} \to X$ such that



- It could be *N-path-surjective*: it could be that, for each 'N-path' $q : N \to A$, there is a N-path $p : N \to X$ such that



  The name of the object N has been chosen to suggest the natural numbers, so that $p : N \to A$ can be thought of as tracing out a *discrete path* of points in A. A point-surjective arrow is, of course, a $\mathbf{1}$-path-surjective arrow.

- It could be *weakly point-surjective* (only if the codomain is an exponential): if $r : X \to Y^A$, then, for each $x : \mathbf{1} \to X$, we obtain $r \circ x : \mathbf{1} \to Y^A$, which corresponds to a morphism

$$(r \circ x)^\wr : A \to Y$$

  It could then be that *every* morphism $A \to Y$ is 'pointwise emulated' by $(r \circ x)^\wr$ for some x. That is, for each $f : A \to Y$, there exists $x_f : \mathbf{1} \to X$ such that

$$\forall a : \mathbf{1} \to A.\ (r \circ x_f)^\wr \circ a = f \circ a$$

  So a weak point-surjection is a bit like 'pointwise cartesian closure.'

Evidently,

$$\begin{aligned} r \text{ is a retraction} &\implies r \text{ is } N\text{-path-surjective} \\ &\overset{\text{if } N \text{ non-empty}}{\dashrightarrow} r \text{ is point-surjective} \\ &\implies r \text{ is weakly point-surjective} \end{aligned}$$

where the third implication only makes sense if codomain of r is an exponential, but may be skipped otherwise. For the second implication, notice that $N \xrightarrow{!} \mathbf{1} \xrightarrow{x} A$ is a N-path, factorise it through X, and pre-compose with any point $n : \mathbf{1} \to N$.

Lawvere then observed that, if the codomain of the weak point-surjection is an exponential Y^A, and the 'indexing object' X coincides with A, a curious phenomenon occurs.

Theorem 52 (Lawvere)**.** *If $r : A \to Y^A$ is a weak point-surjection, then then every arrow $t : Y \to Y$ has a fixed point.*

Proof. Let

$$f \stackrel{\text{def}}{=} A \xrightarrow{\langle r, id_A \rangle} Y^A \times A \xrightarrow{\mathsf{ev}} Y \xrightarrow{t} Y$$

As r is a weak point-surjection, there exists a $x_f : \mathbf{1} \to A$ such that, for all $a : \mathbf{1} \to A$, we have

$$\begin{aligned}
& (r \circ x_f)^\wr \circ a \\
= \quad & \{\, r \text{ is a weak point-surjection} \,\} \\
& f \circ a \\
= \quad & \{\, \text{definition of } f \,\} \\
& t \circ \mathsf{ev} \circ \langle r, id_A \rangle \circ a \\
= \quad & \{\, \text{naturality of product} \,\} \\
& t \circ \mathsf{ev} \circ \langle r \circ a, a \rangle \\
= \quad & \{\, \text{terminal object: } a \circ !_A = id_{\mathbf{1}} \,\} \\
& t \circ \mathsf{ev} \circ \langle r \circ a \circ !_A \circ a, a \rangle \\
= \quad & \{\, \text{naturality of product} \,\} \\
& t \circ \mathsf{ev} \circ \langle r \circ a \circ !_A, id_A \rangle \circ a \\
= \quad & \{\, \text{definition of } (-)^\wr \,\} \\
& t \circ (r \circ a)^\wr \circ a
\end{aligned}$$

Taking $a \stackrel{\text{def}}{=} x_f$ produces a fixed point. □

Lawvere also hinted at a 'cartesian' version of the above result that does not require exponentials. In this version, the diagonal nature of the argument is even more evident. To prove it, we need to introduce the following definition:

Definition 47. An arrow $r : X \times A \to Y$ is a *(cartesian) weak point-surjection* if for every $f : A \to Y$ there exists a $x_f : \mathbf{1} \to X$ such that

$$\forall a : \mathbf{1} \to A.\ r \circ \langle x_f, a \rangle = f \circ a$$

We will not bother to qualify weak point-surjections as ordinary or cartesian, as it will be clear by the context. We can now prove the

Theorem 53 (Lawvere)**.** *If $r : A \times A \to Y$ is a weak point-surjection, then every arrow $t : Y \to Y$ has a fixed point.*

Proof. Let

$$f \stackrel{\text{def}}{=} t \circ r \circ \langle id_A, id_A \rangle$$

Then there exists a $x_f : \mathbf{1} \to A$ such that

$$r \circ \langle x_f, a \rangle = f \circ a$$

for all $a : \mathbf{1} \to A$. We compute that

$$r \circ \langle x_f, x_f \rangle = t \circ r \circ \langle id_A, id_A \rangle \circ x_f = t \circ r \circ \langle x_f, x_f \rangle$$

so that $r \circ \langle x_f, x_f \rangle$ is a fixed point of of t. □

We have seen in §6.1 that the extensional kind of fixed points produced by this theorem are of a sort that *oughtn't* exist. We believe that this is one of the reasons that Lawvere's result has not found wider applications. Nevertheless, we will also see in §6.4.2 that—in a certain setting—this theorem corresponds to a very weak form of Kleene's *First Recursion Theorem*, which has been a central theorem in the semantics of programming languages (see §2).

6.3.2 An intensional Lawvere theorem

Can we adapt Lawvere's result to IFPs? The answer is positive: all we need is a cartesian P-category $\mathfrak{B}$, a product-preserving exposure $Q : \mathfrak{B} \nrightarrow \mathfrak{B}$, and a reasonable quoting device. What remains is to 'embellish' Lawvere's argument with appropriate instances of Q.

Theorem 54 (Intensional Recursion)**.** *Let $Q : \mathfrak{B} \nrightarrow \mathfrak{B}$ be a product-preserving exposure, and let $\delta_A : QA \to Q^2A$ be a reasonable quoting device. If $r : QA \times QA \to Y$ is a weak point-surjection, then every arrow*

$$t : QY \to Y$$

has an intensional fixed point.

Proof. Let

$$f \stackrel{\text{def}}{=} QA \xrightarrow{\langle \delta, \delta \rangle} Q^2A \times Q^2A \xrightarrow{m} Q(QA \times QA) \xrightarrow{Qr} QY \xrightarrow{t} Y$$

Then, there exists a $x_f : \mathbf{1} \to QA$ such that

$$r \circ \langle x_f, a \rangle \sim f \circ a$$

for all $a : \mathbf{1} \to QA$. We compute that

$$
\begin{aligned}
& r \circ \langle x_f, x_f \rangle \\
\sim \ & \{ \text{ definition } \} \\
& t \circ Qr \circ m \circ \langle \delta_A, \delta_A \rangle \circ x_f \\
\sim \ & \{ \text{ naturality } \} \\
& t \circ Qr \circ m \circ \langle \delta_A \circ x_f, \delta_A \circ x_f \rangle \\
\sim \ & \{ \ \delta_A \text{ is a reasonable quoting device } \} \\
& t \circ Qr \circ m \circ \langle Qx_f \circ m_0, Qx_f \circ m_0 \rangle \\
\sim \ & \{ \text{ naturality } \} \\
& t \circ Qr \circ m \circ \langle Qx_f, Qx_f \rangle \circ m_0 \\
\sim \ & \{ \text{ Proposition 2 } \} \\
& t \circ Qr \circ Q\langle x_f, x_f \rangle \circ m_0 \\
\sim \ & \{ \text{ exposures preserve composition } \} \\
& t \circ Q(r \circ \langle x_f, x_f \rangle) \circ m_0
\end{aligned}
$$

so that $r \circ \langle x_f, x_f \rangle$ is a IFP of t. □

6.4 Examples of Fixed Points

In this final section we shall briefly examine what extensional and intensional fixed points mean in the first two examples of exposures presented in §5, namely the Lindenbaum P-category and the P-category of assemblies.

Unfortunately, since the trivial group is a *zero object* in the P-category of groups, points do not carry any interesting structure, and thus neither notion of fixed point is interesting in the third example.

6.4.1 Fixed Points in Gödel numbering: the Diagonal Lemma

We will now carefully consider what we have hinted at throughout the development of the abstract analogues of Gödel and Tarski's results in §6.1.1, viz. the diagonal lemma of Peano arithmetic is precisely the existence of IFPs in the case of the exposure on arithmetic presented in §5.1.

Recall once more the diagonal lemma: for every formula $\phi(x)$ there exists a closed formula $\mathbf{fix}(\phi)$ such that

$$\mathsf{PA} \vdash \mathbf{fix}(\phi) \leftrightarrow \phi(\ulcorner \mathbf{fix}(\phi) \urcorner)$$

Take the formula $\phi(x)$. In $\mathfrak{Lind}(\mathsf{PA})$, it is an arrow

$$\phi(x) : A \to \mathbf{2}$$

Let there be a sentence $\psi : \mathbf{1} \to \mathbf{2}$. When the exposure acts on it, it produces (the numeral of) the Gödel number of ψ, which is a closed term. Suppose ψ is an IFP of ϕ, i.e

$$\psi \sim \phi \circ Q\psi \circ m_0$$

We can then see that the RHS simplifies by substitution to $\phi(\ulcorner\psi\urcorner)$, so this is precisely the conclusion of the diagonal lemma.

6.4.2 Fixed Points in Assemblies: Kleene's Recursion Theorems

We now turn to the consideration of fixed points in the P-category of assemblies $\mathfrak{Asm}(A)$ that we considered in §5.2, along with the paradigmatic exposure

$$\Box : \mathfrak{Asm}(A) \nrightarrow \mathfrak{Asm}(A)$$

EFPs are exactly what one would expect: given an arrow $(f, r) : X \to X$, a EFP of this arrow is a $x \in |X|$ such that

$$f(x) = x$$

We will shortly argue that EFPs in this setting are strongly reminiscent of *Kleene's First Recursion Theorem*, which we discussed at length in §2.

On the contrary, a IFP of an arrow

$$(f, r) : \Box X \to X$$

is more than what we had before: it is an element $x \in |X|$ along with a realizer $a \in \|x\|_X$ of it, such that

$$f(x, a) = x$$

That is, it is a kind of fixed point of f, but the computation of f also depends on the chosen realizer a rather than simply x.

It is worth pausing for a moment to ask what a *vehement* IFP (as discussed in §6.2) is in this case. It is not hard to compute that it consists once again of both an element $x \in |X|$ and a realizer $a \in \|x\|_X$ for it, but the pair now needs to satisfy not only $f(x, a) = x$, but also

$$r \cdot a \simeq a$$

That is, the two sides need to have the same realizer. The above theorem is not too surprising in light of Theorem 28 and Corollary 3: in the idempotent case, the fact that two arrows $\mathbf{1} \to QA$ are equal immediately yields that they are intensionally equal. It therefore follows that vehement IFPs are too strong a notion for what we consider to be intensional recursion.

Kleene's Recursion Theorems

We will now argue that, in the case of $\mathfrak{Asm}(K_1)$, the notions of EFPs and IFPs indeed match the conclusions of the two main theorems that are used to make recursive definitions in computability theory, namely the First and Second Recursion Theorems of Kleene. In fact, we will argue that Lawvere's fixed point theorem (Theorem 53) and our own Intensional Recursion theorem (Theorem 54) each correspond to abstract versions of them. We have discussed the relationship between the two theorems at length in §2, but—for the benefit of the reader—we recapitulate some basic points of this discussion here, in a form tailored to our needs in this chapter.

Let us fix some notation. We write $\simeq$ for *Kleene equality*: we write $e \simeq e'$ to mean either that both expressions e and e' are undefined, if either both are undefined, or both are defined and of equal value. Let ϕ_0, ϕ_1, ... be an enumeration of the partial recursive functions. We will also require the *s-m-n theorem* from computability theory. Full definitions and statements may be found in the book by Cutland [1980].

Theorem 55 (First Recursion Theorem)**.** *Let $\mathcal{PR}$ be the set of unary partial recursive functions, and let $F : \mathcal{PR} \to \mathcal{PR}$ be an* effective operation. *Then $F : \mathcal{PR} \to \mathcal{PR}$ has a fixed point.*

Proof. That $F : \mathcal{PR} \to \mathcal{PR}$ is an effective operation means that there is a partial recursive $f : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$ such that $f(e, x) \simeq F(\phi_e)(x)$. Let $d \in \mathbb{N}$ a code for the partial recursive function $\phi_d(y, x) \stackrel{\text{def}}{=} f(S(y, y), x)$, where $S : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$ is the s-1-1 function of the s-m-n theorem. Then, by the s-m-n theorem, and the definitions of $d \in \mathbb{N}$ and f,

$$\phi_{S(d,d)}(x) \simeq \phi_d(d, x) \simeq f(S(d, d), x) \simeq F(\phi_{S(d,d)})(x)$$

so that $\phi_{S(d,d)}$ is a fixed point of $F : \mathcal{PR} \to \mathcal{PR}$. □

Lawvere's theorem is virtually identical to a point-free version of this proof. Indeed, if we let

$$\begin{aligned} \mathbb{S} : \mathbb{N} \times \mathbb{N} &\longrightarrow \mathcal{PR} \\ (a, b) &\longmapsto \phi_{S(a,b)} \end{aligned}$$

We can then show that this is a weak point-surjection: any 'computable' $f : \mathbb{N} \to \mathcal{PR}$ is actually just an index $d \in \mathbb{N}$ such that

$$\phi_d(x, y) \simeq f(x)(y)$$

for all $x, y \in \mathbb{N}$. We can then show that

$$\mathbb{S}(d, x)(y) \simeq \phi_{S(d,x)}(y) \simeq \phi_d(x, y) \simeq f(x)(y)$$

so that $\mathbb{S}(d, x) = f(x) \in \mathcal{PR}$. Thus the resemblance becomes formal, and the argument applies to yield the FRT.

Yet, one cannot avoid noticing that we have proved more than that for which we bargained. The $f : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$ in the proof implemented a certain effective operation $F : \mathcal{PR} \to \mathcal{PR}$. It follows that f has a special property: it is *extensional*, in the sense that

$$\phi_e = \phi_{e'} \quad \Longrightarrow \quad \forall x \in \mathbb{N}.\ f(e, x) \simeq f(e', x)$$

Notice, however, that the main step that yields the fixed point in this proof also holds for any such f, not just the extensional ones. This fact predates the FRT, and was shown by Kleene [1938].

Theorem 56 (Second Recursion Theorem)**.** *For any partial recursive $f : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$, there exists $e \in \mathbb{N}$ such that $\phi_e(y) \simeq f(e, y)$ for all $y \in \mathbb{N}$.*

This is significantly more powerful than the FRT, as $f(e, y)$ can make arbitrary decisions depending on the source code e, irrespective of the function ϕ_e of which it is the source code. Moreover, it is evident that the function ϕ_e has *access to its own code*, allowing for a certain degree of reflection. Even if f is extensional, hence defining an effective operation, the SRT grants us more power than the FRT: for example, before recursively calling e on some points, $f(e, y)$ could 'optimise' e depending on what y is, hence ensuring that the recursive call will run faster than e itself would.

Lawvere's argument, as presented above, cannot account for the Second Recursion Theorem: $\mathbb{S} : \mathbb{N} \times \mathbb{N} \to \mathcal{PR}$ had $\mathcal{PR}$ in the codomain, and thus this argument only works to yield fixed points of effective operations $\mathcal{PR} \to \mathcal{PR}$. We do not see a meaningful way of replacing this with, say, $\mathcal{N}$. We could certainly replace it with an object $\mathbb{N}_\perp$ that accounts for non-termination, but that is not what we want. we wish to take here.

What we really want is to define a computable *operation* $\mathcal{PR} \dashrightarrow \mathcal{PR}$. In order to explain the meaning of this, a shift in perspective is required. To say that $F : \mathcal{PR} \to \mathcal{PR}$ is an effective operation, we need an extensional, total recursive function f, implemented by some index $d \in \mathbb{N}$ that tracks it on codes, along with a proof that f is extensional. But what about indices $d \in \mathbb{N}$ that describe a total ϕ_d, yet are *not* extensional? These still map every $e \in \mathbb{N}$, which is an index for ϕ_e, to $\phi_d(e)$, which is an index for $\phi_{\phi_d(e)}$. Hence, while f may map codes of partial recursive functions to codes of partial recursive functions, it may do so *without being extensional*. In that case, it defines a *non-functional operation* $G : \mathcal{PR} \dashrightarrow \mathcal{PR}$, which is exactly the case where IFPs will apply.

We can see this far more clearly in the setting of $\mathfrak{Asm}(K_1)$. Arrows

$$\mathbb{N} \to \mathbb{N}_\perp$$

are easily seen to correspond to partial recursive functions. The weak point-surjection $\mathbb{S} : \mathbb{N} \times \mathbb{N} \to \mathcal{PR}$ we produced above can now be seen as an arrow $r : \mathbb{N} \times \mathbb{N} \to \mathbb{N}_\perp^{\mathbb{N}}$ in $\mathfrak{Asm}(K_1)$, and invoking Lawvere's theorem indeed shows that every arrow

$$\mathbb{N}_\perp^{\mathbb{N}} \to \mathbb{N}_\perp^{\mathbb{N}}$$

has an extensional fixed point. Now, by Longley's generalised Myhill-Shepherdson theorem [Longley, 1995, Longley and Normann, 2015], these arrows exactly correspond to effective operations. Hence, in this context Lawvere's theorem states that each effective operation has a fixed point, and indeed corresponds to the simple diagonal argument above.[3]

However, let us look now look at arrows of type

$$\Box\mathbb{N}_\perp^{\mathbb{N}} \to \mathbb{N}_\perp^{\mathbb{N}}$$

These correspond to 'non-functional' transformations, mapping functions to functions, but without respecting extensionality. As every natural number indexes a partial recursive function, these arrows really correspond to all partial recursive functions. It is not hard to see that $\Box\mathbb{N}$ is P-isomorphic to $\mathbb{N}$: this fact can be used with r to immediately produce a weak point-surjection $q : \Box\mathbb{N} \times \Box\mathbb{N} \to \mathbb{N}_\perp^{\mathbb{N}}$, so that, by our Intensional Recursion theorem (Theorem 54), every arrow of type

$$\Box\mathbb{N}_\perp^{\mathbb{N}} \to \mathbb{N}_\perp^{\mathbb{N}}$$

has a IFP. It is thus evident that there is considerable formal similarity between this application of Theorem 54 and Kleene's Second Recursion Theorem!

[3]But note that this is not the complete story, as there is no guarantee that the fixed point obtained in *least*, which is what Kleene's original proof in Kleene [1952] gives. See also §2.

Chapter 7

Intensional Semantics of iPCF I

This is the first of the two chapters that address the issue of finding a truly intensional semantics for the language iPCF which we introduced in §3. Both of these chapters can be seen as evidence towards the claim that iPCF is a truly intensional language.

To make progress towards our goal, we will use the theory of exposures, as developed in §4 and 6. The desired outcome is that the category of assemblies $\mathfrak{Asm}(K_1)$ over the PCA K_1, which—as we showed in §5.2—is a cartesian closed P-category equipped with an idempotent comonadic exposure, is a model of iPCF.

However, before we delve into the details, we need to develop some algebraic machinery for interpreting the rules of iPCF. Because of the delicate behaviour of our two notions of equality—extensional ($\sim$) and intensional ($\approx_Q$)—this will be more involved than it sounds. Nevertheless, a lot of the ground work has been done before, and is still applicable to this setting.

We have discussed the categorical semantics of modal λ-calculi at length in previous work—see [Kavvos, 2017b,c]: therein we found that categorical semantics of a $\mathsf{S4}$ modality comprise a *Bierman-de Paiva category* [Bierman, 2000], viz. a cartesian closed category $(\mathcal{C}, \times, \mathbf{1})$ along with a product-preserving comonad (Q, ϵ, δ). Adapting this to the intensional setting involves exchanging $Q : \mathcal{C} \longrightarrow \mathcal{C}$ for an exposure $Q : (\mathfrak{C}, \sim) \nrightarrow (\mathfrak{C}, \sim)$ that is comonadic. The crucial factor that one should be aware of is that no calculation in our previous work depended on the 'forbidden principle' $f = g \Longrightarrow Qf = Qg$. Hence, a lot of the groundwork may be immediately transferred to the intensional setting.

Nevertheless, there are some differences, and we shall deal with them in this chapter. One of the main ones is our emphasis on *idempotence*. In the ensuing development it will become clear that the *idempotent* case is particularly elegant and useful. What is more, the argument made in §4.2.4, viz. that idempotence is the right notion for intensionality, will be corroborated in this chapter.

7.1 Setting the scene

First, we shall define certain basic ingredients that we will use for our semantics. These amount to algebraic short-hands that will prove incredibly useful for calculations.

We shall define the following arrows by induction (and be lax about the subscripts):

$$m^{(0)} \stackrel{\text{def}}{=} \mathbf{1} \xrightarrow{m_0} Q\mathbf{1}$$
$$m^{(n+1)} \stackrel{\text{def}}{=} \prod_{i=1}^{n+1} QA_i \xrightarrow{m^{(n)} \times id} Q\left(\prod_{i=1}^{n} A_i\right) \times QA_{n+1} \xrightarrow{m} Q\left(\prod_{i=1}^{n+1} A_i\right)$$

Then, the $m^{(n)}$'s are natural, in the sense that

$$m^{(n)} \circ \prod_{i=1}^{n} Qf_i \sim Q\left(\prod_{i=1}^{n} f_i\right) \circ m^{(n)}$$

The main contraption in the semantics of $\mathsf{S4}$ was a hom-set map that generalised the notion of *co-Kleisli lifting*. In our setting, this will be an operation:

$$(-)^* : \mathfrak{C}\left(\prod_{i=1}^{n} QA_i, B\right) \dashrightarrow \mathfrak{C}\left(\prod_{i=1}^{n} QA_i, QB\right)$$

which we define as follows:

$$\frac{f : \prod_{i=1}^{n} QA_i \to B}{f^* \stackrel{\text{def}}{=} \prod_{i=1}^{n} QA_i \xrightarrow{\prod_{i=1}^{n} \delta_{A_i}} \prod_{i=1}^{n} Q^2A_i \xrightarrow{m^{(n)}} Q\left(\prod_{i=1}^{n} QA_i\right) \xrightarrow{Qf} QB}$$

This operation is the categorical counterpart of an admissible rule of $\mathsf{S4}$, which from $\Box\Gamma \vdash A$ allows one to infer $\Box\Gamma \vdash \Box A$. In the weaker setting of K, we only have *Scott's rule*: from $\Gamma \vdash A$ infer $\Box\Gamma \vdash \Box A$. This is also categorically also an operation:

$$(-)^\bullet : \mathfrak{C}\left(\prod_{i=1}^{n} A_i, B\right) \dashrightarrow \mathfrak{C}\left(\prod_{i=1}^{n} QA_i, QB\right)$$

which is defined as follows:

$$\frac{f : \prod_{i=1}^{n} A_i \to B}{f^\bullet \stackrel{\text{def}}{=} \prod_{i=1}^{n} QA_i \xrightarrow{m^{(n)}} Q\left(\prod_{i=1}^{n} A_i\right) \xrightarrow{Qf} QB}$$

It might, of course, seem that both $(-)^\bullet$ and $(-)^*$ are operations, but what they have in common is that they both act by applying the exposure Q to their argument. It follows then that they are *functions* with respect to intensional equality, and hence well-defined on the x-ray category of $\mathfrak{C}$. We write

$$(-)^* : (\mathfrak{C}, \approx_Q)\left(\prod_{i=1}^n QA_i, B\right) \to \mathfrak{C}\left(\prod_{i=1}^n QA_i, QB\right)$$
$$(-)^\bullet : (\mathfrak{C}, \approx_Q)\left(\prod_{i=1}^n A_i, B\right) \to \mathfrak{C}\left(\prod_{i=1}^n QA_i, QB\right)$$

However, if (Q, ϵ, δ) is idempotent, we can show that $(-)^*$ actually preserves intensional equality. For this, we will need the following proposition.

Proposition 11. *If (Q, ϵ, δ) is idempotent, then for $f : \prod_{i=1}^n QA_i \to B$,*

$$Qf^* \sim \delta_B \circ Qf$$

Proof.

$$\begin{aligned}
& Qf^* \\
\sim\ & \{\,\text{definition, } Q \text{ exposure}\,\} \\
& Q^2 f \circ Qm^{(n)} \circ Q\left(\prod_{i=1}^n \delta_{A_i}\right) \\
\sim\ & \{\,\text{Proposition 2}\,\} \\
& Q^2 f \circ Qm^{(n)} \circ m^{(n)} \circ \langle \overrightarrow{Q\delta_{A_i} \circ Q\pi_{A_i}} \rangle \\
\sim\ & \{\, Q \text{ idempotent, so } Q\delta_{A_i} \sim \delta_{QA_i}\,\} \\
& Q^2 f \circ Qm^{(n)} \circ m^{(n)} \circ \langle \overrightarrow{\delta_{QA_i} \circ Q\pi_i} \rangle \\
\sim\ & \{\,\text{product equation, } \delta \text{ monoidal}\,\} \\
& Q^2 f \circ \delta \circ m^{(n)} \circ \langle \overrightarrow{Q\pi_i} \rangle \\
\sim\ & \{\, \delta \text{ natural, inverse of } m^{(n)} \text{ is } \langle \overrightarrow{Q\pi_i} \rangle\,\} \\
& \delta \circ Qf
\end{aligned}$$

□

This allows us to infer that

Corollary 5. *If Q is idempotent then $(-)^*$ preserves $\approx_Q$: it is a map*

$$(-)^* : (\mathfrak{C}, \approx_Q)\left(\prod_{i=1}^n QA_i, B\right) \to (\mathfrak{C}, \approx_Q)\left(\prod_{i=1}^n QA_i, QB\right)$$

Proof. If $f \approx_Q g$, then $Qf^* \sim \delta \circ Qf \sim \delta \circ Qg \sim Qg^*$. □

7.2 Distribution and naturality laws

We now look at the relationship between the two operations $(-)^\bullet$ and $(-)^*$, and we state and prove certain 'distribution' laws that apply to them, and describe their interaction. First, we note that it is definitionally the case that

$$f^* \stackrel{\text{def}}{=} f^\bullet \circ \prod_{i=1}^{n} \delta_{A_i}$$

for $f : \prod_{i=1}^n QA_i \to B$.

Proposition 12. *Given a comonadic exposure (Q, ϵ, δ), the following equations hold up to $\sim$. Furthermore, if Q is idempotent, they hold up to $\approx_Q$.*

(i) $id^*_{QA} \sim \delta_{QA}$

(ii) $\epsilon^*_A \sim id_{QA}$

(iii) For $k : \prod_{i=1}^n QA_i \to B$ and $l : QB \to C$,

$$(l \circ k^*)^* \sim l^* \circ k^*$$

(iv) For $k : \prod_{i=1}^n A_i \to B$ and $l : QB \to C$,

$$(l \circ k^\bullet)^* \sim l^* \circ k^\bullet$$

(v) Let $f : \prod_{i=1}^n B_i \to C$ and $g_i : \prod_{j=1}^k QA_j \to B_i$ for $i = 1, \ldots, n$. Then

$$(f \circ \langle \overrightarrow{g_i} \rangle)^* \sim f^\bullet \circ \left\langle \overrightarrow{g_i^*} \right\rangle$$

(vi) For $f : \prod_{i=1}^n QA_i \to B$ and $\langle \overrightarrow{\pi_j} \rangle : \prod_{i=1}^n QA_i \to \prod_{j \in J} QA_j$ for J a list with elements from $\{1, \ldots, n\}$,

$$(f \circ \langle \overrightarrow{\pi_j} \rangle)^* \sim f^* \circ \langle \overrightarrow{\pi_j} \rangle$$

(vii) If (Q, ϵ, δ) is idempotent then for $f : \prod_{i=1}^n QA_i \to B$ and $k : \prod_{j=1}^m QD_j \to \prod_{i=1}^n QA_i$ we have

$$(f \circ k)^* \approx_Q f^* \circ k$$

and hence $(f \circ k)^* \sim f^* \circ k$.

Proof. Straightforward calculations involving the comonadic equations. (i) and (ii) are standard from the theory of comonads and functional programming. In the case of idempotence we can use them alongside Proposition 11 to prove the intensional equation, e.g.

$$Q(id_A^*) \sim \delta_{QA} \circ Qid_A \sim \delta_{QA} \sim Q\delta_A$$

and hence $id_{QA}^* \approx_Q \delta_{QA}$. For (ii) use $\delta_A \circ Q\epsilon_A \sim id_{Q^2A}$. (iii) and (iv) are easy calculations; e.g. for (iv):

$$\begin{aligned}
& (l \circ k^\bullet)^* \\
\sim \quad & \{ \text{definitions} \} \\
& Ql \circ Q^2k \circ Qm^{(n)} \circ m^{(n)} \circ \prod_{i=1}^{n} \delta_{A_i} \\
\sim \quad & \{ \delta \text{ monoidal} \} \\
& Ql \circ Q^2k \circ \delta \circ m^{(n)} \\
\sim \quad & \{ \delta \text{ natural} \} \\
& Ql \circ \delta \circ Qk \circ m^{(n)}
\end{aligned}$$

which by definition is $l^* \circ k^\bullet$. Given idempotence it is simple to use Proposition 11 to prove (iii) and (iv) up to $\approx_Q$, without even using the non-idempotent result; e.g. for (iv):

$$Q\left(l \circ k^\bullet\right)^* \sim \delta_C \circ Ql \circ Qk^\bullet \sim Ql^* \circ Qk^\bullet$$

(v) is a simple but lengthy calculation. With idempotence we have

$$\begin{aligned}
& Q\left(f \circ \langle \overrightarrow{\pi_j} \rangle\right)^* \\
\sim \quad & \{ \text{Proposition 11} \} \\
& \delta \circ Qf \circ Q\langle \overrightarrow{g_i} \rangle \\
\sim \quad & \{ \delta \text{ natural, Proposition 2} \} \\
& Q^2f \circ \delta \circ m^{(n)} \circ \langle \overrightarrow{Qg_i} \rangle \\
\sim \quad & \{ \delta \text{ monoidal} \} \\
& Q^2f \circ Qm^{(n)} \circ m^{(n)} \circ \prod_{i=1}^{n} \delta \circ \langle \overrightarrow{Qg_i} \rangle \\
\sim \quad & \{ \text{product equation, Proposition 11} \} \\
& Q^2f \circ Qm^{(n)} \circ m^{(n)} \circ \langle \overrightarrow{Q(g_i^*)} \rangle \\
\sim \quad & \{ \text{Proposition 2} \} \\
& Q^2f \circ Qm^{(n)} \circ Q\langle \overrightarrow{(g_i^*)} \rangle
\end{aligned}$$

and hence $(f \circ \langle \overrightarrow{g_i} \rangle)^* \approx_Q f^\bullet \circ \langle \overrightarrow{g_i^*} \rangle$. (vi) is a corollary of (v), once we notice that $\pi_i^* \sim \delta_{A_i} \circ \pi_i$, and use the definition of $f^* \stackrel{\text{def}}{=} f^\bullet \circ \prod \delta$. For the idempotent case, notice that $\pi_i^* \approx_Q \delta_{A_i} \circ \pi_i$, or derive it as a corollary of (vii).

(v) is an easy calculation:

$$Q(f \circ k)^* \sim \delta \circ Qf \circ Qk \sim Qf^* \circ Qk$$

which yields $(f \circ k)^* \approx_Q f^* \circ k$, and hence $(f \circ k)^* \sim f^* \circ k$. It is worth noting that we know of no direct calculation that proves (7) up to $\sim$ without going through $\approx_Q$. □

In other news, $(-)^*$ interacts predictably with δ and ϵ.

Proposition 13.

(i) Let $f : \prod_{i=1}^n QA_i \to B$. Then $\delta_B \circ f^* \sim (f^*)^*$. Furthermore, if Q is idempotent then this equation holds intensionally.

(ii) Let $f : \prod_{i=1}^n QA_i \to B$. Then $\epsilon_B \circ f^* \sim f$. Furthermore, if Q is idempotent then this equation holds intensionally.

Proof.

1. Let $E \stackrel{\text{def}}{=} \prod_{i=1}^n QA_i$. Then

$$\begin{aligned}
& \delta_B \circ f^* \\
\sim\ & \{ \text{definition} \} \\
& \delta_B \circ Qf \circ m^{(n)} \circ \prod_{i=1}^n \delta_{A_i} \\
\sim\ & \{ \delta \text{ natural} \} \\
& Q^2 f \circ \delta_E \circ m^{(n)} \circ \prod_{i=1}^n \delta_{A_i} \\
\sim\ & \{ \delta \text{ monoidal} \} \\
& Q^2 f \circ Qm^{(n)} \circ m^{(n)} \circ \prod_{i=1}^n \delta_{QA_i} \circ \prod_{i=1}^n \delta_{A_i} \\
\sim\ & \{ \text{product is functorial, comonadic equation} \} \\
& Q^2 f \circ Qm^{(n)} \circ m^{(n)} \circ \prod_{i=1}^n Q\delta_{A_i} \circ \prod_{i=1}^n \delta_{A_i} \\
\sim\ & \{ Q \text{ product-preserving} \} \\
& Q^2 f \circ Qm^{(n)} \circ Q\left(\prod_{i=1}^n \delta_{A_i} \right) \circ m^{(n)} \circ \prod_{i=1}^n \delta_{A_i} \\
\sim\ & \{ \text{definitions} \} \\
& (f^*)^*
\end{aligned}$$

If Q is idempotent, we can also compute that

$$\begin{aligned} & Q\delta \circ Qf^* \\ \sim \; & \{ \text{Proposition 11} \} \\ & Q\delta \circ \delta \circ Qf \\ \sim \; & \{ \text{comonadic equation} \} \\ & \delta \circ \delta \circ Qf \\ \sim \; & \{ \text{Proposition 11} \} \\ & \delta \circ Qf^* \\ \sim \; & \{ \text{Proposition 11} \} \\ & Q\left((f^*)^*\right) \end{aligned}$$

and hence $\delta \circ f^* \approx_Q (f^*)^*$.

2. Straightforward calculation involving—amongst other things—the naturality and monoidality of ϵ. If Q is idempotent, then we calculate that

$$\begin{aligned} & Q\epsilon_B \circ Qf^* \\ \sim \; & \{ \text{Proposition 11} \} \\ & Q\epsilon_B \circ \delta_B \circ Qf \\ \sim \; & \{ \text{comonadic equation} \} \\ & Qf \end{aligned}$$

and hence $\epsilon_B \circ f^* \approx_Q f$.

□

To conclude this section, we note that a special case of Proposition 12(7) generalises the Quotation-Evaluation lemma (Lemma 17).

Corollary 6. *If (Q, ϵ, δ) is idempotent then for $k : \prod_{i=1}^m QB_i \to QA$ we have*

$$(\epsilon_A \circ k)^* \approx_Q k$$

and hence $(f \circ k)^* \sim f^* \circ k$.

Proof. By Proposition 12 (vii) & (ii), $(\epsilon_A \circ k)^* \approx_Q \epsilon_A^* \circ k \approx_Q k$ □

7.3 Fixed Points with Parameters

In §6 we discussed two types of fixed points, EFPs and IFPs. The first type concerned arrows $f : A \to A$, whereas the second pertained to arrows $f : QA \to A$. However, if we are to formulate a categorical semantics of PCF and iPCF, we are going to need a slightly more general notion of each fixed point, namely a *fixed point with parameters*. The parameters correspond to the *context of free variables* in the presence of which we are taking the relevant fixed point.

In the case of extensional fixed points, the context appears as a cartesian product in the domain of the morphism, which is of type $t : B \times Y \to Y$. B is usually of the form $\prod_{i=1}^{n} B_i$. Indeed, this is what happens in the categorical semantics of PCF, for which see Hyland and Ong [2000], Poigné [1992], or Longley [1995]. It is not at all difficult to generalise Lawvere's theorem to produce this kind of fixed point.

We then move on to intensional fixed points. The situation in this case is slightly more nuanced, for—as we saw in §6.2—we essentially need to model our fixed points after *Löb's rule*, viz.

$$\frac{\Box A \to A}{A}$$

Adapting this rule to a parametric version is not a trivial task, as it is almost equivalent to developing proof theory for it. However, we can look to our previous work on the proof theory of GL [Kavvos, 2017b,c] to find a good pattern. We briefly discussed that work in §6.2. The right form of the generalised rule is

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

However, since iPCF is based on an $\mathsf{S4}$-like setting, the $\mathsf{4}$ axiom is available. Thus, it suffices to consider a rule of the following shape:

$$\frac{\Box\Gamma \vdash \Box A \to A}{\Box\Gamma \vdash \Box A}$$

Indeed, this is very close to the rule we used for iPCF in §3: the only remaining step is the weaken the fixed point by dropping the box in the conclusion, thereby mimicking the form of Löb's rule more commonly found in the literature. Therefore, IFPs with parameters will pertain to arrows of type

$$f : \prod_{i=1}^{n} QB_i \times QA \to A$$

and yield

$$f^\dagger : \prod_{i=1}^{n} QB_i \to A$$

We thus arrive at a kind of context for IFPs that consists of a handful of intensional assumptions QB_i. We elaborate on that in §7.3.2.

But, before that, let us recall the extensional case.

7.3.1 Extensional Fixed Points with Parameters

Suppose we have a cartesian 1-category $\mathcal{C}$. An arrow

$$f : B \times A \to A$$

can be considered as a sort of endomorphism of A that is 'parameterised' by B. Type-theoretically, we will consider B to be the context, and we can take the EFP 'at A.'

Definition 48. Let $\mathcal{C}$ be a cartesian 1-category. A *parametric extensional fixed point* (parametric EFP) of $f : B \times A \to A$ is an arrow $f^\dagger : B \to A$ such that the following diagram commutes:

$$\begin{array}{ccc} B & \xrightarrow{\langle id_B, f^\dagger \rangle} & B \times A \\ {\scriptstyle f^\dagger}\downarrow & \swarrow_{f} & \\ A & & \end{array}$$

Definition 49 (Extensional Fixed Points with Parameters)**.** A cartesian category $\mathcal{C}$ has *parametric extensional fixed points* (parametric EFPs) at $A \in \mathcal{C}$ just if for all $B \in \mathcal{C}$ there exists a map

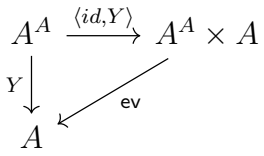
$$(-)^\dagger_B : \mathcal{C}(B \times A, A) \longrightarrow \mathcal{C}(B, A)$$

such that for each $f : B \times A \to A$, the morphism $f^\dagger : B \to A$ is a EFP of f.

This, of course, is an 'external view' of what it means to have parametric EFPs in a category. However, in typed λ-calculi we often include a fixed point combinator $Y_A : (A \to A) \to A$ in our calculus, which is rather more 'internal.' This fixed point combinator can be either *weak* or *strong*, and always occurs in the context of a CCC.

Definition 50 (Fixed Point Combinators)**.** Let $\mathcal{C}$ be a cartesian closed category.

1. A *strong fixed point combinator at* A is an arrow $Y : A^A \to A$ such that



2. A *weak fixed point combinator at* A is an arrow $Y : A^A \to A$ such that, for each $f : B \times A \to A$, the arrow

$$Y \circ \lambda(f) : B \to A$$

is a EFP of f.

Surprisingly, we have the following theorem.

Theorem 57. *In a cartesian closed category $\mathcal{C}$, the following are equivalent:*

1. *A strong fixed point combinator at A.*
2. *A weak fixed point combinator at A.*
3. *Extensional fixed points at A.*

Proof. We will prove a circular chain of implications.

Case$(1 \Rightarrow 2)$. If $Y : A^A \to A$ is a strong FPC, then

$$Y \circ \lambda(f) = \mathsf{ev} \circ \langle id, Y \rangle \circ \lambda(f) = \mathsf{ev} \circ \langle \lambda(f), Y \circ \lambda(f) \rangle = f \circ \langle id, Y \circ \lambda(f) \rangle$$

so $Y \circ \lambda(f)$ is a fixed point of f.

Case$(2 \Rightarrow 3)$. Trivial: define $(-)^\dagger$ by λ-abstraction and composition with Y.

Case$(3 \Rightarrow 1)$. A strong FPC Y at A is a fixed point of $\mathsf{ev} : A^A \times A \to A$. Hence, if we let

$$Y \stackrel{\text{def}}{=} A^A \xrightarrow{\mathsf{ev}^\dagger} A$$

then $Y = \mathsf{ev} \circ \langle id, Y \rangle$.

□

Finally, we note the following naturality property, which we learned from Simpson and Plotkin [2000].

Lemma 25. *EFPs induced by a (strong or weak) fixed point combinator satisfy the following equation: for any $f : B \times A \to A$ and $g : C \to B$,*

$$(f \circ (g \times id_A))^\dagger = f^\dagger \circ g$$

Proof. We have

$$(f \circ (g \times id_A))^\dagger = Y \circ \lambda\,(f \circ (g \times id_A)) = Y \circ \lambda(f) \circ g = f^\dagger \circ g$$

by naturality of the $\lambda(-)$ operation. □

It is easy to extend the cartesian version of Lawvere's theorem to one that also yields fixed points with parameters: we redefines weak point-surjections to include a parameter.

Definition 51 (Parametric weak point-surjection)**.** An arrow $r : X \times A \to Y$ is a *parametric weak point-surjection* if for every $f : B \times A \to Y$ there exists a $x_f : B \to X$ such that

$$\forall a : B \to A.\ r \circ \langle x_f, a\rangle = f \circ \langle id_B, a\rangle$$

The only change in the main theorem is that the fixed points of an arrow of type $B \times Y \to Y$ now have type $B \to Y$ instead of being points of Y.

Theorem 58 (Parametric Recursion)**.** *If $r : A \times A \to Y$ is a parametric weak-point surjection, then every arrow*

$$t : B \times Y \to Y$$

has a EFP.

Proof. Let

$$f \stackrel{\text{def}}{=} B \times A \xrightarrow{id_B \times \langle id_A, id_A\rangle} B \times (A \times A) \xrightarrow{id_B \times r} B \times A \xrightarrow{t} A$$

Then there exists a $x_f : B \to A$ such that

$$r \circ \langle x_f, a\rangle = f \circ a$$

for all $a : B \to A$. Then

$$\begin{aligned} & r \circ \langle x_f, x_f\rangle \\ = \ & \{\,\text{definition of parametric weak point-surjection}\,\} \\ & t \circ (id_B \times r) \circ (id_B \times \langle id_A, id_A\rangle) \circ \langle id_B, x_f\rangle \\ = \ & \{\,\text{various product equations}\,\} \\ & t \circ \langle id_B, r \circ \langle x_f, x_f\rangle\rangle \end{aligned}$$

so that $r \circ \langle x_f, x_f\rangle$ is a EFP of t. □

7.3.2 Intensional Fixed Points with Parameters

Without further ado, we generalise IFPs to include parameters.

Definition 52. Let (Q, ϵ, δ) be a product-preserving comonadic exposure. A *parametric intensional fixed point* (parametric IFP) of $f : \prod_{i=1}^{n} QB_i \times QA \to A$ (w.r.t. Q) is an arrow

$$f^\dagger : \prod_{i=1}^{n} QB_i \to A$$

such that the following diagram commutes up to $\sim$:

$$\begin{array}{ccc} \prod_{i=1}^{n} QB_i & \xrightarrow{\langle id, (f^\dagger)^* \rangle} & \prod_{i=1}^{n} QB_i \times QA \\ {\scriptstyle f^\dagger}\downarrow & \swarrow_{f} & \\ A & & \end{array}$$

If $f : \prod_{i=1}^{n} QB_i \times QA \to A$, then $f^\dagger : \prod_{i=1}^{n} QB_i \to A$, so $\left(f^\dagger\right)^* : \prod_{i=1}^{n} QB_i \to QA$, so this diagram has the right types.

This definition is a generalisation analogous to the one for EFPs. The context is intensional ($\prod_{i=1}^{n} QB_i$) instead of extensional ($B = \prod_{i=1}^{n} B_i$), and $f^\dagger$ appears under the co-Kleisli lifting, so essentially under an occurrence of Q. Notice that we have used both product-preservation and the quoter $\delta : Q \overset{\bullet}{\to} Q^2$ for this definition; this probably corresponds to the fact that that the $\mathsf{4}$ axiom ($\Box A \to \Box\Box A$) is a theorem of provability logic: see [Boolos, 1994, Kavvos, 2017b,c]. Since we are in a categorical logic setting, an appropriate gadget standing for the $\mathsf{4}$ axiom must be given as a primitive, alongside appropriate coherence conditions (e.g. the first comonadic equation).

Definition 53 (Intensional Fixed Points with Parameters)**.** Let (Q, ϵ, δ) be a product-preserving comonadic exposure on $\mathfrak{B}$. We say that $\mathfrak{B}$ has *parametric intensional fixed points at* A (parametric IFPs) just if for any objects $B_i \in \mathcal{C}$ there exists an operation

$$(-)^\dagger_{\overrightarrow{B_i}} : \mathfrak{B}\left(\prod_{i=1}^{n} QB_i \times QA, A\right) \dashrightarrow \mathfrak{B}\left(\prod_{i=1}^{n} QB_i, A\right)$$

such that for each arrow $f : \prod_{i=1}^{n} QB_i \times QA \to A$, the arrow $f^\dagger : \prod_{i=1}^{n} QB_i \to A$ is an intensional fixed point of f.

We often write $f^\dagger$ for succinctness, without specifying the context $\overrightarrow{B_i}$.

In analogy to EFPs, there is a similar 'internal' view of this definition, related to the *Gödel-Löb axiom*. However, in contrast to what we had before, it comes with certain caveats. But first, some more definitions:

Definition 54 (Intensional Fixed Point Combinators)**.** Let there be a cartesian closed P-category $\mathfrak{B}$, along with a product-preserving comonadic exposure (Q, ϵ, δ) on it.

1. A *strong intensional fixed point combinator at* A (w.r.t. Q) is an arrow

$$Y : Q(A^{QA}) \to QA$$

such that the following diagram commutes up to $\sim$:

$$\begin{array}{ccc} Q(A^{QA}) & \xrightarrow{\langle \epsilon, Y \rangle} & A^{QA} \times QA \\ {\scriptstyle Y}\downarrow & & \downarrow{\scriptstyle \mathsf{ev}} \\ QA & \xrightarrow[\epsilon]{} & A \end{array}$$

2. A *weak intensional fixed point combinator* (w.r.t. Q) is an arrow

$$Y : Q(A^{QA}) \to QA$$

such that, for each $f : \prod_{i=1}^{n} QB_i \times QA \to A$, the arrow

$$\prod_{i=1}^{n} QB_i \xrightarrow{(\lambda(f))^*} Q\left(A^{QA}\right) \xrightarrow{Y} QA \xrightarrow{\epsilon_A} A$$

is a IFP of f.

Theorem 59. *Let $\mathfrak{B}$ be a cartesian closed P-category, along with a product-preserving comonadic exposure (Q, ϵ, δ) on it.*

1. *If (Q, ϵ, δ) is idempotent then a strong fixed point combinator is also a weak fixed point combinator.*

2. *A weak fixed point combinator $Y : Q(A^{QA}) \to QA$ implies the existence of intensional fixed points at A.*

3. *The existence of intensional fixed points at A implies the existence of a strong intensional fixed point combinator at A.*

Proof.

1. Let $Y : Q(A^{QA}) \to QA$ be a strong FPC. Then, given $f : \prod_{i=1}^{n} QB_i \times QA \to A$, we may calculate that

$$\begin{aligned} & \epsilon \circ Y \circ (\lambda f)^* \\ \sim \; & \{ \text{ definition of strong FPC } \} \\ & \mathsf{ev} \circ \langle \epsilon, Y \rangle \circ (\lambda f)^* \\ \sim \; & \{ \text{ naturality of product, Proposition 13 (ii) } \} \\ & \mathsf{ev} \circ \langle \lambda f, Y \circ (\lambda f)^* \rangle \\ \sim \; & \{ \text{ cartesian closure } \} \\ & f \circ \langle id, Y \circ (\lambda f)^* \rangle \end{aligned}$$

   But as Q is idempotent and $Y \circ (\lambda f)^* : \prod_{i=1}^{n} QB_i \to QA$, we can use quotation-evaluation (Corollary 6) to conclude that $(\epsilon \circ Y \circ (\lambda f)^*)^* \approx_Q Y \circ (\lambda f)^*$, and hence that we have a IFP.

2. Trivial.

3. Let

$$g \stackrel{\text{def}}{=} Q(A^{QA}) \times QA \xrightarrow{\epsilon \times id} A^{QA} \times QA \xrightarrow{\mathsf{ev}} A$$

   Then we can show that $(g^\dagger)^* : Q(A^{QA}) \to QA$ is a strong FPC. It is easy to calculate that $g^\dagger \sim \mathsf{ev} \circ \langle \epsilon, \left(g^\dagger\right)^* \rangle$ and hence that

$$\epsilon \circ \left(g^\dagger\right)^* \sim g^\dagger \sim \mathsf{ev} \circ \langle \epsilon, \left(g^\dagger\right)^* \rangle$$

   by using Proposition 13 (ii).

□

It is interesting to examine if and when the map $(-)^\dagger$ might turn $\approx_Q$ to $\sim$, or even preserve it. In fact, it is easy to see that if $\lambda(-)$ preserves $\approx_Q$, then we can build a $(-)^\dagger$ that turns it into $\sim$: a weak FPC suffices. If Q is idempotent then $(-)^\dagger$ also preserves $\approx_Q$.

Lemma 26. *If $Y : Q(A^{QA}) \to QA$ is a weak FPC, and the map*

$$\lambda_C(-) : \mathfrak{C}(C \times A, B) \to \mathfrak{C}(C, B^A)$$

preserves the intensional equality $\approx_Q$, i.e. is a map

$$\lambda_C(-) : (\mathfrak{C}, \approx_Q)(C \times A, B) \to (\mathfrak{C}, \approx_Q)(C, B^A)$$

then the IFPs induced by $f^\dagger \stackrel{def}{=} \epsilon_A \circ Y \circ (\lambda(f))^*$ *turn* $\approx_Q$ *into* $\sim$*, i.e.* $(-)^\dagger$ *is a map*

$$(-)^\dagger : (\mathfrak{B}, \approx_Q)\left(\prod_{i=1}^n QB_i \times QA, A\right) \to \mathfrak{B}\left(\prod_{i=1}^n QB_i, A\right)$$

Moreover, if Q *is idempotent, then* $(-)^\dagger$ *preserves* $\approx_Q$*, and hence is a map*

$$(-)^\dagger : (\mathfrak{B}, \approx_Q)\left(\prod_{i=1}^n QB_i \times QA, A\right) \to (\mathfrak{B}, \approx_Q)\left(\prod_{i=1}^n QB_i, A\right)$$

Proof. Since $\lambda(-)$ preserves $\approx_Q$ (by assumption), and $(-)^*$ turns that into $\sim$ (§7.1) the result follows. In the case of idempotence observe that $(-)^*$ also preserves $\approx_Q$ (Corollary 5), and so does post-composition with $\epsilon \circ Y$. □

A similar argument will yield that, if $\lambda(-)$ is natural up to $\approx_Q$, then so are the fixed points in a certain sense.

Lemma 27. *If* (Q, ϵ, δ) *is idempotent,* $Y : Q(A^{QA}) \to QA$ *is a weak FPC, and*

$$\lambda_C(-) : \mathfrak{C}(C \times A, B) \to \mathfrak{C}(C, B^A)$$

is natural up to $\approx_Q$*, i.e.*

$$\lambda\left(f \circ (g \times id)\right) \approx_Q \lambda(f) \circ g$$

then the IFPs induced by $f^\dagger \stackrel{def}{=} \epsilon_A \circ Y \circ (\lambda(f))^*$ *are also natural, i.e. for* $f : \prod_{i=1}^n QB_i \times QA \to A$ *and* $k : \prod_{j=1}^k QC_j \to \prod_{i=1}^n QB_i$*, we have*

$$(f \circ (k \times id))^\dagger \approx_Q f^\dagger \circ k$$

Proof. Use naturality for $\lambda(-)$ up to $\approx_Q$, Corollary 5 for the preservation of $\approx_Q$ by $(-)^*$, and finally Proposition 12 (vii) to show that $(\lambda(f) \circ k)^* \approx_Q (\lambda(f))^* \circ k$. □

There is also a weaker version of this lemma that does not require idempotence: it states that if $f : \prod_{i=1}^n QB_i \times QA \to A$ and $g_i : C_i \to B_i$, then $(f \circ (\prod_{i=1}^n Qg_i \times id))^\dagger \sim f^\dagger \circ \prod_{i=1}^n g_i$. However, we do not find any use for it in the sequel.

We can summarise the above results by saying that if we have

- an idempotent comonadic exposure (Q, ϵ, δ);
- any of our three flavours of IFPs (strong, weak, $(-)^\dagger$);
- a $\lambda(-)$ that preserves $\approx_Q$ and is natural up to it

then we obtain IFPs at A, which preserve $\approx_Q$ and are natural up to $\approx_Q$. While this is nice and useful, we will see in §8.6 that in the most intensional of P-categories it will emphatically *not* be the case that $\lambda(-)$ preserves $\approx_Q$.

7.4 A Parametric Intensional Lawvere Theorem

In the final section of this chapter we shall generalise the Intensional Recursion Theorem (Theorem 54) to a version that admits parameters. The theorem will now guarantee the existence of parametric IFPs.

Accordingly, we will have to also change the antecedent: neither weak point-surjections (Def. 47) nor parametric weak point-surjections (Def. 51) are adequate anymore. We shall therefore introduce a variant, which—due to lack of imagination on the part of the author—we shall call an *intensional parametric weak point-surjection*. The only essential difference is the restriction of the arbitrary context B to an intensional one, viz. of the form $\prod_{i=1}^{n} QB_i$.

Definition 55 (IPWPS)**.** An arrow $r : X \times A \to Y$ is a *intensional parametric weak point-surjection (IPWPS)* if, for every $f : \prod_{i=1}^{n} QB_i \times A \to Y$, there exists a $x_f : \prod_{i=1}^{n} QB_i \to X$ such that

$$\forall a : \prod_{i=1}^{n} QB_i \to A.\ r \circ \langle x_f, a\rangle = f \circ \langle id, a\rangle$$

Armed with this, we can now prove a theorem analogous to the Parametric Recursion Theorem (Theorem 58), but yielding IFPs instead. Recall that in the Intensional Recursion Theorem (Theorem 54) we only needed a particular component of δ to be 'reasonable,' but in this case we will require full idempotence.

Theorem 60 (Parametric Intensional Recursion)**.** *Let (Q, ϵ, δ) be a product-preserving idempotent comonadic exposure. If $r : QA \times QA \to Y$ is a IPWPS, then every arrow*

$$t : \prod_{i=1}^{n} QB_i \times QY \to Y$$

has an intensional fixed point.

Proof. Let

$$\begin{aligned} f \stackrel{\text{def}}{=} \prod_{i=1}^{n} QB_i \times QA &\xrightarrow{id\times\langle id,id\rangle} \prod_{i=1}^{n} QB_i \times (Q^2A \times Q^2A) \\ &\xrightarrow{id\times r^*} \prod_{i=1}^{n} QB_i \times QY \\ &\xrightarrow{t} Y \end{aligned}$$

Then, there exists a $x_f : \prod_{i=1}^n QB_i \to QA$ such that

$$r \circ \langle x_f, a \rangle \sim f \circ \langle id, a \rangle$$

for all $a : \prod_{i=1}^n QB_i \to QA$. We compute that

$$\begin{array}{ll} & r \circ \langle x_f, x_f \rangle \\ \sim & \{ \text{ definition of IPWPS } \} \\ & t \circ (id \times r^*) \circ (id \times \langle id, id \rangle) \circ \langle id, x_f \rangle \\ \approx_Q & \{ \text{ product equation, naturality of brackets } \} \\ & t \circ \langle id, r^* \circ \langle x_f, x_f \rangle \rangle \\ \approx_Q & \{ \text{ idempotence: Prop. 12(vii) } \} \\ & t \circ \langle id, (r \circ \langle x_f, x_f \rangle)^* \rangle \end{array}$$

so that $r \circ \langle x_f, x_f \rangle$ is a IFP of t. □

Notice that we are allowed to use Proposition 12(vii) only because $\langle x_f, x_f \rangle : \prod_{i=1}^n QB_i \to QA \times QA$ is of the right type, i.e. there are occurrences of Q 'guarding' all the types.

Naturality

As we saw in §7.3.2, the main reason for introducing parametric IFPs is to have a context $\prod_{i=1}^n QB_i$. In turn, a context is useful because one can *substitute* for any objects in it, as they represent free variables. In this light, naturality is a key property: it states that substitution (composition) commutes with the (type-theoretic/categorical) construct in question; we will discuss that more in §8.

We showed in Lemmata 26 and 27 that, in the idempotent setting, if IFPs are induced by a weak FPC, and $\lambda(-)$ preserves $\approx_Q$, then so does $(-)^\dagger$, and moreover it is natural. But what about the IFPs produced by the Parametric Intensional Recursion Theorem?

A IPWPS $r : QA \times QA \to Y$ induces a map

$$f : \prod_{i=1}^n QB_i \times QA \to Y \longmapsto x_f : \prod_{i=1}^n QB_i \to QA$$

and then we can define $(-)^\dagger : \mathfrak{B}\left(\prod_{i=1}^n QB_i \times QY, Y\right) \dashrightarrow \mathfrak{B}\left(\prod_{i=1}^n QB_i, Y\right)$ to be

$$t \longmapsto f_t \longmapsto x_{f_t} \longmapsto r \circ \langle x_{f_t}, x_{f_t} \rangle :$$

where $f_t \stackrel{\text{def}}{=} t \circ (id \times r^*) \circ (id \times \langle id, id \rangle) : \prod_{i=1}^n QB_i \times QA \to Y$.

The final map $x_{f_t} \mapsto r \circ \langle x_{f_t}, x_{f_t} \rangle$ is clearly natural and preserves both $\sim$ and $\approx_Q$, as $\langle \cdot, \cdot \rangle$ does. But what about the rest? Suppose that we pre-compose t with $g \times id$, where $g : \prod_{j=1}^{m} QD_j \to \prod_{i=1}^{n} QB_i$. The map $t \mapsto f_t$ clearly preserves both $\sim$ and $\approx_Q$, and moreover it is natural, in the sense that

$$f_{t \circ (g \times id)} \approx_Q f_t \circ (g \times id)$$

because of the interchange law for cartesian products, which holds up to $\approx_Q$.

This leaves only the difficult case of $f \mapsto x_f$. For naturality, we would need[1]

$$x_{f_{t \circ (g \times id)}} \approx_Q x_{f_t} \circ g$$

which easily follows if $f \mapsto x_f$ preserves $\approx_Q$, and[2]

$$x_{h \circ (g \times id)} \approx_Q x_h \circ g$$

In that case $x_{f_{t \circ (g \times id)}} \approx_Q x_{f_t \circ (g \times id)} \approx_Q x_{f_t} \circ g$, and hence

$$(t \circ (g \times id))^\dagger \approx_Q r \circ \langle x_{f_{t \circ (g \times id)}}, x_{f_{t \circ (g \times id)}} \rangle \approx_Q r \circ \langle x_{f_t} \circ g, x_{f_t} \circ g \rangle \approx_Q r \circ \langle x_{f_t}, x_{f_t} \rangle \circ g \approx_Q t^\dagger \circ g$$

Hence, to obtain naturality we would need that the IPWPS be 'natural,' and that $f \mapsto x_f$ turn $\approx_Q$ to $\sim$.

Corollary 7. *If the IPWPS $r : QA \times QA \to Y$ of Theorem 60 is such that $f \mapsto x_f$ preserves $\approx_Q$ (or, equivalently, turns $\approx_Q$ into $\sim$), and is 'natural' insofar as*

$$x_{f \circ (g \times id)} \approx_Q x_f \circ g$$

(equivalently, with $\sim$) then the induced $(-)^\dagger$ is a map

$$(-)^\dagger : (\mathfrak{B}, \approx_Q) \left(\prod_{i=1}^{n} QB_i \times QY, Y \right) \to (\mathfrak{B}, \approx_Q) \left(\prod_{i=1}^{n} QB_i, Y \right)$$

which is natural up to $\approx_Q$, i.e.

$$(f \circ (g \times id))^\dagger \approx_Q f^\dagger \circ g$$

[1] Note that because of idempotence and the type of x_f, $\sim$ suffices.
[2] Ditto.

Chapter 8

Intensional Semantics of iPCF II

In this final chapter, we shall use all our results up to this point to attempt to produce an intensional semantics for iPCF. The intended—but, as it transpires, unachievable—result is that the category of assemblies $\mathfrak{Asm}(K_1)$ is a model of iPCF.

We have shown in previous chapters that a *product-preserving comonadic exposure* satisfies most of the standard equations of a product-preserving comonad. It should therefore be the case that, given such an exposure, one can use it to almost directly interpret the Davies-Pfenning fragment of iPCF. However, this is not so straightforward. The main ingredient of our soundness result is a *substituion lemma*, which relates the interpretation with substition. Since we are allowed to substitute terms under $\mathsf{box}\ (-)$ constructs—which we model by $(-)^*$—we need the substitution lemma to hold up to $\approx_Q$ at that location. Furthermore, since we may have multiple nested occurrences of boxes, we will need that $(-)^*$ preserve $\approx_Q$, which is the case if the comonadic exposure is *idempotent* (Cor. 5). Thus idempotence is essential.

Once this is decided, we run into a second issue. Suppose that there is a λ in a term, which we model by the function $\lambda(-)$ of a cartesian closed category. Since this λ might occur in the scope of a $\mathsf{box}\ (-)$, and a (modal) variable for which we want to substitute might occur under that λ, the above soundness result requires that $\lambda(-)$ be natural up to $\approx_Q$. Whereas in the case of $\mathfrak{Asm}(K_1)$, the exposure is idempotent, we have no guarantees at all about the naturality of $\lambda(-)$—see §5.2.5. We will therefore need to *forbid λ-abstractions with free variables under boxes.*

The above suffices to interpret the Davies-Pfenning fragment, in an intensional sense. The remaining piece of the puzzle pertains to the construction of IFPs. This is easy for booleans and naturals, but at higher types we are only able to use an inductive construction that only builds IFPs for a certain kind of *intensional exponential ideal*: if X is any object and Y is in the ideal, then Y^{QX} is in the ideal as well. We also face certain difficulties in showing that the IFPs are natural, as per §7.4, and lack of

naturality entails lack of substitution. It follows that *we must constrain the taking of IFPs to closed terms, at only at certain types.*

We are thus led to reformulate iPCF, or—more specifically—to admit only a subset of it. To reach this subset, we first *layer* it, so as to separate the *intensional layer*—where everything will hold up to $\approx_Q$—and the *extensional layer*, where things will hold up to $\sim$. Since the modality already implicitly enables this kind of layering, this will be a minor change. Nevertheless, it decisively quells the problem, as we will never need to substitute in a λ-abstraction under the box, so the desired lemma holds up to $\approx_Q$ at that location. In the case of IFPs, our only option is to alter the fixed point rule.

The resulting language is called *iPCF v2.0*. On the one hand, the new language has $\mathfrak{Asm}(K_1)$ as a model; on the other, a lot of expressiveness has been lost. We discuss what has been lost, and when one can regain it. In particular, we can model iPCF itself when given a *natural* iPCF v2.0 model. Furthermore, a *weakly extensional* model of iPCF v2.0 is already natural; the terminology comes from PCAs: if A is a weakly extensional PCA, then $\mathfrak{Asm}(A)$ is a weakly extensional model, and hence a model of iPCF; see also §5.2.5.

8.1 iPCF v2.0

We promptly introduce iPCF v2.0. We shall not prove any theorems in this section, because we have formally proven all of them in Agda: see Appendix A for the proofs.

Each typing judgement of iPCF v2.0 will be *annotated* by a $\mathcal{J}$, like so:

$$\Delta\,;\Gamma \vdash_{\mathcal{J}} M : A$$

The possible options for $\mathcal{J}$ will be "int." for intensional, and "ext." for extensional.

The revised system appears in Figure 8.1. The occurrence of a generic $\mathcal{J}$ in these rules is universally quantified. There is little else to say apart from the fact that these rules enforce the prohibition of free variables under a λ in the intensional judgements. In programming language terms this could be transliterated as the prohibition of the creation of *closures* (= λ-abstraction + environment for free variables). Of course, a term can then only be placed under a box if it is intensional, but the boxed term itself can be either intensional or extensional:

$$\frac{\Delta\,;\cdot \vdash_{\text{int.}} M : A}{\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{box}\ M : \Box A}$$

Figure 8.1: Syntax and Typing Rules for Intensional PCF v2.0

$$\begin{array}{rrcl}
\textbf{Ground Types} & G & ::= & \mathsf{Nat} \mid \mathsf{Bool} \\
\textbf{Types} & A, B & ::= & G \mid A \to B \mid \Box A \\
\textbf{Fixable Types} & A_{\mathsf{fix}} & ::= & G \mid \Box A \to A_{\mathsf{fix}} \\
\textbf{Terms} & M, N & ::= & x \mid \lambda x{:}A.\ M \mid MN \mid \mathsf{box}\ M \mid \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N \mid \\
 & & & \widehat{n} \mid \mathsf{true} \mid \mathsf{false} \mid \mathsf{succ} \mid \mathsf{pred} \mid \mathsf{zero?} \mid\ \supset_G\ \mid \mathsf{fix}\ z\ \mathsf{in}\ M \\
\textbf{Contexts} & \Gamma, \Delta & ::= & \cdot \mid \Gamma, x : A
\end{array}$$

$$\frac{}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} \widehat{n} : \mathsf{Nat}} \qquad \frac{}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} b : \mathsf{Bool}}\ (b \in \{\mathsf{true}, \mathsf{false}\})$$

$$\frac{}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} \mathsf{zero?} : \mathsf{Nat} \to \mathsf{Bool}} \qquad \frac{}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} f : \mathsf{Nat} \to \mathsf{Nat}}\ (f \in \{\mathsf{succ}, \mathsf{pred}\})$$

$$\frac{}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} \supset_G : \mathsf{Bool} \to G \to G \to G}$$

$$\frac{}{\Delta\,;\,\Gamma, x{:}A, \Gamma' \vdash_{\mathcal{J}} x : A}\ (\mathsf{var}) \qquad \frac{}{\Delta, u{:}A, \Delta'\,;\,\Gamma \vdash_{\mathcal{J}} u : A}\ (\Box\mathsf{var})$$

$$\frac{\Delta\,;\,\Gamma, x{:}A \vdash_{\text{ext.}} M : B}{\Delta\,;\,\Gamma \vdash_{\text{ext.}} \lambda x{:}A.\ M : A \to B}\ (\to \mathcal{I}) \qquad \frac{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} M : A \to B \quad \Delta\,;\,\Gamma \vdash_{\mathcal{J}} N : A}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} MN : B}\ (\to \mathcal{E})$$

$$\frac{\Delta\,;\,\cdot \vdash_{\text{int.}} M : A}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} \mathsf{box}\ M : \Box A}\ (\Box\mathcal{I}) \qquad \frac{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} M : \Box A \qquad \Delta, u{:}A\,;\,\Gamma \vdash_{\mathcal{J}} N : C}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N : C}\ (\Box\mathcal{E})$$

$$\frac{\cdot\,;\,z : \Box A_{\mathsf{fix}} \vdash_{\text{int.}} M : A_{\mathsf{fix}}}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} \mathsf{fix}\ z\ \mathsf{in}\ M : A_{\mathsf{fix}}}\ (\Box\mathsf{fix})$$

$$\frac{\cdot\,;\,x{:}A \vdash_{\mathcal{J}} M : B}{\Delta\,;\,\Gamma \vdash_{\text{int.}} \lambda x{:}A.\ M : A \to B}\ (\to \mathcal{I}_{\text{int.}})$$

We can only λ-abstract in an extensional term, yielding another extensional term:

$$\frac{\Delta\,;\,\Gamma, x{:}A \vdash_{\text{ext.}} M : B}{\Delta\,;\,\Gamma \vdash_{\text{ext.}} \lambda x{:}A.\ M : A \to B}$$

But if this term has no other free variables, then the result can be an intensional term. Of course, we must not forget to include the 'opportunity' to weaken the context:

$$\frac{\cdot\,;\,x{:}A \vdash_{\mathcal{J}} M : B}{\Delta\,;\,\Gamma \vdash_{\text{int.}} \lambda x{:}A.\ M : A \to B}$$

We shall also change the rule for intensional fixed points, which now reads

$$\frac{\cdot\,;\,z : \Box A_{\mathsf{fix}} \vdash_{\text{int.}} M : A_{\mathsf{fix}}}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} \mathsf{fix}\ z\ \mathsf{in}\ M = M[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z] : A_{\mathsf{fix}}}\ (\Box\mathsf{fix})$$

So $\mathsf{fix}\ z\ \mathsf{in}\ M$ recurses, but it can only do so when M is closed to everything else save the 'diagonal' variable z. What is more, we can only invoke this rule for types A_{fix} generated by the following grammar, where A is any type at all:

$$A_{\mathsf{fix}} \quad ::= \quad \mathsf{Nat} \quad | \quad \mathsf{Bool} \quad | \quad \Box A \to A_{\mathsf{fix}}$$

Theorem 61. *The following rules are admissible in iPCF v2.0:*

$$\frac{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} M : A}{\Delta\,;\,\Gamma \vdash_{ext.} M : A} \qquad \frac{\Delta\,;\,\Gamma \vdash_{int.} M : A}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} M : A}$$

The standard admissibility results for iPCF are also valid in iPCF v2.0, but they are now parametric in $\mathcal{J}$.

Theorem 62 (Structural)**.** *The standard structural rules of iPCF (as stated in Theorem 21) are admissible in iPCF v2.0, parametrically up to $\mathcal{J}$.*

The situation with the cut rule, however, is slightly more complicated, and this has to do with the nature of the term being substituted. If we substitute an intensional term for a variable, the resulting term will still retain its original disposition (int. or ext.). However, if we substitute an extensional term, we force the resulting term to be extensional. Similarly, and because of $(\Box\mathcal{I})$, we may only substitute an intensional term for a modal variable, and that leaves the disposition of the term invariant.

Theorem 63 (Cut for iPCF v2.0)**.** *The following rules are admissible in iPCF v2.0.*

1. *(Cut-Ext)*
$$\frac{\Delta\,;\,\Gamma \vdash_{ext.} N : A \qquad \Delta\,;\,\Gamma, x{:}A, \Gamma' \vdash_{\mathcal{J}} M : A}{\Delta\,;\,\Gamma, \Gamma' \vdash_{ext.} M[N/x] : A}$$

2. *(Cut-Int)*

$$\frac{\Delta\,;\Gamma \vdash_{int.} N : A \qquad \Delta\,;\Gamma, x{:}A, \Gamma' \vdash_{\mathcal{J}} M : A}{\Delta\,;\Gamma,\Gamma' \vdash_{\mathcal{J}} M[N/x] : A}$$

3. *(Modal Cut)*

$$\frac{\Delta\,;\cdot \vdash_{int.} N : A \quad \Delta, u{:}A, \Delta'\,;\Gamma \vdash_{\mathcal{J}} M : C}{\Delta,\Delta'\,;\Gamma \vdash_{\mathcal{J}} M[N/u] : C}$$

In light of those theorems we reformulate the equational theory of iPCF (Figure 3.3) for iPCF v2.0. The resulting theory can be found in Figure 8.2. Curiously, we see that *a form of the congruence rule for* $\Box$ *reappears*, even though the considerations of Davies and Pfenning led us to banish such rules in §3. When proving soundness, we will see that this rule reflects the fact shown in Corollary 5, viz. that $(-)^*$ preserves intensional equality $\approx_Q$.

Expressitivity of iPCF v2.0

It is easy to see that every typing judgment of iPCF v2.0 is also a typing judgment of iPCF: each rule of v2.0 is a special case of the rule for iPCF. Hence, iPCF v2.0 is in some sense a proper subset of iPCF.

We can thus conclude that, by moving to v2.0, we have lost some expressivity. This is centred around three limitations:

1. no free variables under λ-abstractions in the modal/intensional fragment, i.e. under a $\mathsf{box}\ (-)$;

2. IFPs can only be taken when there is exactly one free variable, the *diagonal* variable, and that must be of modal type; and

3. IFPs can only be taken at certain types

The first limitation is, in a way, double-edged. On the one hand, it is reasonable and familiar from someone coming from the computability theory, especially from the perspective of Jones [1997]. Indices do not have "free variables"; sometimes they are meant to have more than one argument, and in those cases we use the s-m-n theorem to substitute for one of those; this can be simulated here using λ-abstraction. On the other hand, this limitation invalidates *every single one of the original examples of* $\mathsf{S4}$*-typed staged metaprogramming of Davies and Pfenning [2001]* (*power*, *acker*, *ip*, etc.—see §7 of that paper): in almost all cases, a $\mathsf{box}\ (-)$ containing a λ-abstraction with free variables is implicated in the result.

Figure 8.2: Equational Theory for Intensional PCF v2.0

Function Spaces

$$\frac{\Delta\,;\Gamma \vdash_{\text{ext.}} N : A \qquad \Delta\,;\Gamma, x{:}A, \Gamma' \vdash_{\mathcal{J}} M : B}{\Delta\,;\Gamma, \Gamma' \vdash_{\text{ext.}} (\lambda x{:}A.M)\, N = M[N/x] : B}\,(\to \beta)$$

$$\frac{\Delta\,;\Gamma \vdash_{\mathcal{J}} M : A \to B \qquad x \notin \text{fv}(M)}{\Delta\,;\Gamma \vdash_{\text{ext.}} M = \lambda x{:}A.Mx : A \to B}\,(\to \eta)$$

Modality

$$\frac{\Delta\,;\cdot \vdash_{\text{int.}} M : A \qquad \Delta, u : A\,;\Gamma \vdash_{\mathcal{J}} N : C}{\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N = N[M/x] : C}\,(\Box\beta)$$

$$\frac{\Delta\,;\Gamma \vdash_{\text{int.}} M : \Box A}{\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in\ box}\ u = M : \Box A}\,(\Box\eta)$$

$$\frac{\Delta\,;\cdot \vdash_{\text{int.}} M = N : A}{\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{box}\ M = \mathsf{box}\ N : \Box A}\,(\Box\mathsf{cong})$$

$$\frac{\cdot\,; z : \Box A \vdash_{\text{int.}} M : A}{\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{fix}\ z\ \mathsf{in}\ M = M[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z] : A}\,(\Box\mathsf{fix})$$

$$\frac{\Delta\,;\Gamma \vdash_{\mathcal{J}} M = N : \Box A \qquad \Delta\,;\Gamma \vdash_{\mathcal{J}} P = Q : C}{\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ P = \mathsf{let\ box}\ u \Leftarrow N\ \mathsf{in}\ Q : B}\,(\Box\mathsf{let\text{-}cong})$$

Remark. In addition to the above, one should also include (a) rules that ensure that equality is an equivalence relation, (b) congruence rules for λ-abstraction and application, and (c) rules corresponding to the behaviour of constants, as e.g. in Figure 3.2.

The second limitation seems reasonably innocuous, but it is actually more severe than it looks. As an example, it renders the *intensional fixed-point combinator*—whose type is the Gödel-Löb axiom, see §3.4—untypable:

$$\mathbb{Y}_A \stackrel{\text{def}}{=} \lambda x : \Box(\Box A \to A).\ \mathsf{let\ box}\ f \Leftarrow x\ \mathsf{in\ box}\ (\mathsf{fix}\ z\ \mathsf{in}\ f\ z)$$
$$\not\vdash \mathbb{Y}_A : \Box(\Box A \to A) \to \Box A$$

Notice the characteristic occurrence of f within the $\mathsf{fix}\ z\ \mathsf{in}\ (-)$ construct. This particular limitation is severe, and increases the distance between iPCF and provability logic: Löb's rule does not even imply the Gödel-Löb axiom. This has to do with the 'loss of naturality' that occurs when we constrain the fixed point rule: the $\Box$ in the antecedent $\Box(\Box A \to A)$ means that 'only code variables can be found in the input data,' and this is too weak an assumption to use Löb's rule, which requires exactly one free variable, the *diagonal* one. This could be fixed by redefining $\mathsf{box}\ (-)$ to only enclose *completely closed* terms, but then we completely obliterate the expressive content of iPCF. If we did that, modal types would not be useful for anything at all, leading us to add more and more *combinators* that do specific things, e.g. something like $\mathbf{app} : \Box(A \to B) \to \Box A \to \Box B$ with a δ-reduction of the form $\mathbf{app}\ (\mathsf{box}\ F)(\mathsf{box}\ M) \to \mathsf{box}\ FM$. We are not sure that this approach is sustainable.

The relationship of this second limitation with computability theory is more subtle. Using the SRT on a program with a 'free variable' made no sense at all, as using it on a program with 'two arguments' was the norm; thus, not much seems to be lost. However, we remind the reader that *'constructive'* versions of the SRT are available in computability theory, e.g. there is a a partial recursive function $n(-)$ that, given an index e, returns $n(e)$ which would be a sort of IFP of e: see §2.1.2. These, one would expect, are *intensional fixed point combinators*. The fact that we cannot define an intensional fixed-point combinator with the Gödel-Löb axiom as type means that this is not possible to do *within* iPCF v2.0, not unless we redefine $\Box$ to mean *completely closed*. This is a serious limitation towards the goal of making iPCF v2.0 a typed 'language of indices' that works more or less in the style of Jones [1997]. The only way out of this impasse would be to show that our intended model is *natural*, so that we could model iPCF itself. We discuss this further in §8.4.

8.2 Interpreting iPCF v2.0

We have finally made it to the categorical interpretation of iPCF v2.0, which we will use the algebraic machinery developed in §7 to define. We assume that the reader

has some background on the categorical semantics of simply-typed λ-calculus. Useful expositions include the classics by Lambek and Scott [1988] and Crole [1993], as well as the detailed presentation of Abramsky and Tzevelekos [2011]. For the details of the categorical semantics of modal λ-calculi see [Kavvos, 2017b,c].

First, we define the notion of a iPCF v2.0 model.

Definition 56 (iPCF v2.0 model)**.** An iPCF v2.0 model consists of

(i) a cartesian closed P-category $(\mathfrak{C}, \sim, \times, \mathbf{1})$;

(ii) a product-preserving idempotent comonadic exposure (Q, ϵ, δ);

(iii) a choice of objects $\mathbb{N}$ and $\mathbb{B}$, suitable for interpreting the constants (see Hyland and Ong [2000]); and

(iv) maps $(-)^\dagger$ yielding IFPs at all the objects generated by

$$I \quad ::= \quad \mathbb{B} \quad | \quad \mathbb{N} \quad | \quad I^{QZ}$$

for all $Z \in \mathfrak{C}$, as per Definition 53.

Given an iPCF v2.0 model we define an object $[\![A]\!] \in \mathfrak{C}$ for every type A of iPCF, by induction:

$$\begin{aligned} [\![\mathsf{Nat}]\!] &\stackrel{\text{def}}{=} \mathbb{N} \\ [\![\mathsf{Bool}]\!] &\stackrel{\text{def}}{=} \mathbb{B} \\ [\![A \to B]\!] &\stackrel{\text{def}}{=} [\![B]\!]^{[\![A]\!]} \\ [\![\Box A]\!] &\stackrel{\text{def}}{=} Q\,[\![A]\!] \end{aligned}$$

Then, given a well-defined context $\Delta\,;\Gamma$ where $\Delta = u_1{:}B_1, \dots u_n{:}B_n$ and $\Gamma = x_1{:}A_1, \dots, x_m{:}A_m$, we let

$$[\![\Delta\,;\Gamma]\!] \stackrel{\text{def}}{=} QB_1 \times \cdots \times QB_n \times A_1 \times \cdots \times A_m$$

where the product is, as ever, left-associating.

We then extend the semantic map $[\![-]\!]$ to one that associates an arrow

$$[\![\Delta\,;\Gamma \vdash M : A]\!] : [\![\Delta\,;\Gamma]\!] \to [\![A]\!]$$

of the P-category $\mathfrak{C}$ to each derivation $\Delta\,;\Gamma \vdash M : A$. The full definition is given in Figure 8.3. The map

$$\pi_\Delta^{\Delta;\Gamma} : [\![\Delta\,;\Gamma]\!] \to [\![\Delta\,;\cdot]\!]$$

Figure 8.3: Categorical Semantics for Intensional PCF v2.0

$$[\![\Delta\,;\Gamma, x{:}A, \Gamma' \vdash_{\mathcal{J}} x : A]\!] \stackrel{\text{def}}{=} \pi : [\![\Delta\,;\Gamma, x{:}A, \Gamma']\!] \longrightarrow [\![A]\!]$$

$$[\![\Delta, u{:}A, \Delta'\,;\Gamma \vdash_{\mathcal{J}} u : A]\!] \stackrel{\text{def}}{=} \epsilon_A \circ \pi : [\![\Delta, u{:}A, \Delta'\,;\Gamma]\!] \to [\![\Box A]\!] \to [\![A]\!]$$

$$[\![\Delta\,;\Gamma \vdash_{\text{ext.}} \lambda x{:}A.M : A \to B]\!] \stackrel{\text{def}}{=} \lambda\left([\![\Delta\,;\Gamma, x : A \vdash_{\text{ext.}} M : B]\!]\right) : [\![\Delta\,;\Gamma]\!] \longrightarrow [\![B]\!]^{[\![A]\!]}$$

$$[\![\Delta\,;\Gamma \vdash_{\text{int.}} \lambda x{:}A.M : A \to B]\!] \stackrel{\text{def}}{=} \lambda\left([\![\cdot\,; x : A \vdash_{\text{int.}} M : B]\!] \circ \pi_2^{\mathbf{1},[\![A]\!]}\right) \circ ! : [\![\Delta\,;\Gamma]\!] \longrightarrow [\![B]\!]^{[\![A]\!]}$$

$$[\![\Delta\,;\Gamma \vdash_{\mathcal{J}} MN : B]\!] \stackrel{\text{def}}{=} \mathsf{ev} \circ \langle [\![\Delta\,;\Gamma \vdash_{\mathcal{J}} M : A \to B]\!], [\![\Delta\,;\Gamma \vdash_{\mathcal{J}} N : A]\!] \rangle$$

$$[\![\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N : C]\!] \stackrel{\text{def}}{=} [\![\Delta, u{:}A\,;\Gamma \vdash_{\mathcal{J}} N : C]\!] \circ \langle \overrightarrow{\pi_\Delta}, [\![\Delta\,;\Gamma \vdash_{\mathcal{J}} M : \Box A]\!], \overrightarrow{\pi_\Gamma} \rangle$$

Definitions for modal rules

$$[\![\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{box}\ M : \Box A]\!] \stackrel{\text{def}}{=} [\![\Delta\,;\cdot \vdash_{\text{int.}} M : A]\!]^* \circ \pi_\Delta^{\Delta;\Gamma}$$

$$[\![\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{fix}\ z\ \mathsf{in}\ M : A]\!] \stackrel{\text{def}}{=} [\![\cdot\,; z : \Box A \vdash_{\text{int.}} M : A]\!]^\dagger \circ !$$

is the obvious projection. Moreover, the notation $\langle \overrightarrow{\pi_\Delta}, f, \overrightarrow{\pi_\Gamma} \rangle$ stands for

$$\langle \overrightarrow{\pi_\Delta}, f, \overrightarrow{\pi_\Gamma} \rangle \stackrel{\text{def}}{=} \langle \pi_1, \ldots, \pi_n, f, \pi_{n+1}, \ldots, \pi_{n+m} \rangle$$

The first thing we need to observe is that there is no difference in the interpretation if the term is intensional or extensional: if a term can be both, it has the same interpretation.

Lemma 28. *If* $\Delta \,;\, \Gamma \vdash_{int.} M : A$*, then*

$$[\![\Delta \,;\, \Gamma \vdash_{int.} M : A]\!] = [\![\Delta \,;\, \Gamma \vdash_{ext.} M : A]\!]$$

where $=$ *stands for strict equality.*

8.3 Soundness

The main tools in proving soundness of our interpretation are (a) lemmas giving the categorical interpretation of various admissible rules, and (b) a fundamental lemma relating substitution of terms to composition in the category. In the sequel we often use informal vector notation for contexts: for example, we write $\vec{u} : \vec{B}$ for the context $u_1 : B_1, \ldots, u_n : B_m$. We also write $[\vec{N}/\vec{u}]$ for the simultaneous, capture-avoiding substitution $[N_1/u_1, \ldots, N_m/u_n]$.

First, we interpret weakening and exchange.

Lemma 29 (Semantics of Weakening)**.**

1. *Let* $\Delta \,;\, \Gamma, x{:}C, \Gamma' \vdash_{int.} M : A$ *with* $x \notin \mathrm{FV}(M)$*. Then*

   $$[\![\Delta \,;\, \Gamma, x{:}C, \Gamma' \vdash_{int.} M : A]\!] \approx_Q [\![\Delta \,;\, \Gamma, \Gamma' \vdash_{int.} M : A]\!] \circ \pi$$

   where $\pi : [\![\Delta \,;\, \Gamma, x{:}C, \Gamma']\!] \to [\![\Delta \,;\, \Gamma, \Gamma']\!]$ *is the obvious projection.*

2. *Let* $\Delta \,;\, \Gamma, x{:}C, \Gamma' \vdash_{ext.} M : A$ *with* $x \notin \mathrm{FV}(M)$*. Then*

   $$[\![\Delta \,;\, \Gamma, x{:}C, \Gamma' \vdash_{ext.} M : A]\!] \sim [\![\Delta \,;\, \Gamma, \Gamma' \vdash_{ext.} M : A]\!] \circ \pi$$

   where $\pi : [\![\Delta \,;\, \Gamma, x{:}C, \Gamma']\!] \to [\![\Delta \,;\, \Gamma, \Gamma']\!]$ *is the obvious projection. If the iPCF model is weakly extensional (see §8.4) then the result holds up to* $\approx_Q$*.*

3. *Let* $\Delta, u{:}B, \Delta' \,;\, \Gamma \vdash_{int.} M : A$ *with* $u \notin \mathrm{FV}(M)$*. Then*

   $$[\![\Delta, u{:}B, \Delta' \,;\, \Gamma \vdash_{int.} M : A]\!] \approx_Q [\![\Delta, \Delta' \,;\, \Gamma \vdash_{int.} M : A]\!] \circ \pi$$

   where $\pi : [\![\Delta, u{:}B, \Delta' \,;\, \Gamma]\!] \to [\![\Delta, \Delta' \,;\, \Gamma]\!]$ *is the obvious projection.*

4. Let $\Delta, u{:}B, \Delta' ; \Gamma \vdash_{ext.} M : A$ *with* $u \notin \mathrm{FV}(M)$*. Then*

$$[\![\Delta, u{:}B, \Delta' ; \Gamma \vdash_{ext.} M : A]\!] \sim [\![\Delta, \Delta' ; \Gamma \vdash_{ext.} M : A]\!] \circ \pi$$

where $\pi : [\![\Delta, u{:}B, \Delta' ; \Gamma]\!] \to [\![\Delta, \Delta' ; \Gamma]\!]$ *is the obvious projection. If the iPCF model is weakly extensional (see §8.4) then the result holds up to* $\approx_Q$*.*

Proof. By induction on the derivations. Most cases are straightforward.

The first one holds up to $\approx_Q$ because it essentially consists of projecting away components, which holds intensionally: the fact the judgement is intensional means no λ's are involved.

The second one holds only up to $\sim$, because of the occurrence of $\lambda(-)$'s in the semantics. If moreover the model is weakly extensional, $\lambda(-)$ preserves $\approx_Q$ (Cor. 5) so we can strengthen the inductive hypothesis to $\approx_Q$ and obtain the result up to intensional equality.

The third one also follows easily. In the case of a $\mathsf{box}\,(-)$, the term within it is intensional, so we use the induction hypothesis and the fact[1] $(-)^*$ preserves $\approx_Q$. We then know that $(-)^*$ is natural for projections (Prop. 12(vi)) up to $\approx_Q$ (due to idempotence). There is not much to show in the case of $\mathsf{fix}\, z\, \mathsf{in}\, (-)$, as no modal variables occur freely under it.

The fourth one is perhaps the most complicated, and only holds up to $\sim$, again because of the occurrence of $\lambda(-)$'s. In the case of $\mathsf{box}\,(-)$, the term within it is intensional, so we use the third result and the fact $(-)^*$ preserves $\approx_Q$, followed again by naturality for projections. The case of $\mathsf{fix}\, z\, \mathsf{in}\, (-)$ is again trivial. □

We can also show that the components of the interpretation interact in the expected way with the corresponding term formation rules in the language. These follows from the analogous lemmata in §7.1, which show that the necessary equations hold intensionally in a iPCF model, i.e. when the exposure Q is idempotent.

Lemma 30 (Double box)**.** *If* $\Delta ; \cdot \vdash_{\mathcal{J}} M : A$*, then*

$$[\![\Delta ; \Gamma \vdash_{\mathcal{J}} \mathsf{box}\,(\mathsf{box}\, M) : \Box\Box A]\!] \approx_Q \delta_A \circ [\![\Delta ; \Gamma \vdash_{\mathcal{J}} \mathsf{box}\, M : \Box A]\!]$$

[1] This is where idempotence is essential, otherwise this would hold only up to $\sim$ and the inductive hypothesis for $\mathsf{box}\,(-)$ would fail.

Proof. Let $f \stackrel{\text{def}}{=} [\![\Delta\,;\cdot \vdash_{\mathcal{J}} M : A]\!]$. Then

$$\begin{array}{ll} & [\![\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{box}\ (\mathsf{box}\ M) : \Box\Box A]\!] \\ \approx_Q & \{\,\text{definitions}\,\} \\ & (f^*)^* \circ \pi_{\Delta}^{\Delta;\Gamma} \\ \approx_Q & \{\,\text{Proposition 13}\,\} \\ & \delta_A \circ f^* \circ \pi_{\Delta}^{\Delta;\Gamma} \\ \approx_Q & \{\,\text{definitions}\,\} \\ & \delta_A \circ [\![\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{box}\ M : \Box A]\!] \end{array}$$

□

Lemma 31 (Identity Lemma)**.** *For* $(u_i : B_i) \in \Delta$*,*

$$[\![\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{box}\ u_i : \Box B_i]\!] \approx_Q \pi_{\Box B_i}^{\Delta;\Gamma}$$

Proof.

$$\begin{array}{ll} & [\![\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{box}\ u_i : \Box B_i]\!] \\ \approx_Q & \{\,\text{definition}\,\} \\ & \left(\epsilon_{B_i} \circ \pi_{\Box B_i}^{\Delta;\cdot}\right)^* \circ \pi_{\Delta}^{\Delta;\Gamma} \\ \approx_Q & \{\,\text{Proposition 12}\,\} \\ & \epsilon_{B_i}^* \circ \pi_{\Box B_i}^{\Delta;\cdot} \circ \pi_{\Delta}^{\Delta;\Gamma} \\ \approx_Q & \{\,\text{Proposition 12 (ii), projections}\,\} \\ & \pi_{\Box B_i}^{\Delta;\Gamma} \end{array}$$

□

Lemma 32 (Semantics of Substitution)**.** *Suppose that* $\Delta\,;\Gamma \vdash_{\mathcal{J}} M_i : A_i$ *for* $i = 1,\ldots,n$*, and that* $\Delta\,;\cdot \vdash_{int.} N_j : B_j$ *for* $j = 1,\ldots,m$*, and let*

$$\begin{aligned} \beta_j &\stackrel{def}{=} [\![\Delta\,;\Gamma \vdash_{int.} \mathsf{box}\ N_j : \Box B_j]\!] \\ \alpha_i &\stackrel{def}{=} [\![\Delta\,;\Gamma \vdash_{\mathcal{J}} M_i : A_i]\!] \end{aligned}$$

Then,

1. *if* $\vec{u} : \vec{B}\,;\vec{x} : \vec{A} \vdash_{int.} P : C$*, we have*

$$\left[\!\!\left[\Delta\,;\Gamma \vdash_{\mathcal{J}} P[\vec{N}/\vec{u}, \vec{M}/\vec{x}] : C\right]\!\!\right] \approx_Q \left[\!\!\left[\vec{u} : \vec{B}\,;\vec{x} : \vec{A} \vdash_{int.} P : C\right]\!\!\right] \circ \left\langle \overrightarrow{\beta}, \overrightarrow{\alpha} \right\rangle$$

2. *if* $\vec{u} : \vec{B} \,;\, \vec{x} : \vec{A} \vdash_{ext.} P : C$, *we have*

$$\left[\!\left[\Delta \,;\, \Gamma \vdash_{ext.} P[\vec{N}/\vec{u}, \vec{M}/\vec{x}] : C\right]\!\right] \sim \left[\!\left[\vec{u} : \vec{B} \,;\, \vec{x} : \vec{A} \vdash_{ext.} P : C\right]\!\right] \circ \left\langle \overrightarrow{\beta}, \overrightarrow{\alpha} \right\rangle$$

Proof. By induction on the derivation of $\vec{u} : \vec{B} \,;\, \vec{x} : \vec{A} \vdash_{\mathcal{J}} P : C$. Most cases are straightforward, and use a combination of standard equations that hold in cartesian closed categories—see [Crole, 1993, §2]—in order to perform calculations very close the ones detailed in [Abramsky and Tzevelekos, 2011, §1.6.5]. Because of the precise definitions we have used, we also need to make use of Lemma 29 to interpret weakening whenever variables in the context do not occur freely in the term. We only cover the modal cases.

Case($\Box$var). Then $P \equiv u_i$ for some u_i amongst the $\vec{u}$. Hence, the LHS is $\Delta \,;\, \Gamma \vdash_{\text{int.}} N_i : B_i$, whereas we calculate that the RHS, in either case, is

$$\begin{aligned}
& \left[\!\left[\vec{u} : \vec{B} \,;\, \vec{x} : \vec{A} \vdash_{\mathcal{J}} P : C\right]\!\right] \circ \langle \vec{\beta}, \vec{\alpha} \rangle \\
\approx_Q \;& \{ \text{ definition, projection } \} \\
& \epsilon_{B_i} \circ [\![\Delta \,;\, \Gamma \vdash_{\text{int.}} \mathsf{box}\, N_i : \Box B_i]\!] \\
\approx_Q \;& \{ \text{ definition } \} \\
& \epsilon_{B_i} \circ [\![\Delta \,;\, \cdot \vdash_{\text{int.}} N_i : B_i]\!]^* \circ \pi_{\Delta}^{\Delta;\Gamma} \\
\approx_Q \;& \{ \text{ Proposition 13 } \} \\
& [\![\Delta \,;\, \cdot \vdash_{\text{int.}} N_i : B_i]\!] \circ \pi_{\Delta}^{\Delta;\Gamma} \\
\approx_Q \;& \{ \text{ Semantics of Weakening (Lemma 29) } \} \\
& [\![\Delta \,;\, \Gamma \vdash_{\text{int.}} N_i : B_i]\!]
\end{aligned}$$

Case($\Box\mathcal{I}$). We have that $\overrightarrow{u} : \overrightarrow{B} \,;\, \overrightarrow{x} : \overrightarrow{A} \vdash_{\mathcal{J}} \mathsf{box}\, P : \Box C$, so that $\overrightarrow{u} : \overrightarrow{B} \,;\, \cdot \vdash_{\text{int.}} P : C$, with the result that none of the variables $\overrightarrow{x}$ occur in P. Hence

$P[\vec{N}/\vec{u}, \vec{M}/\vec{x}] \equiv P[\vec{N}/\vec{u}]$, and we calculate, in either case:

$$
\begin{aligned}
& \left[\!\left[\Delta \,;\, \Gamma \vdash_{\mathcal{J}} \mathsf{box}\,(P[\vec{N}/\vec{u}, \vec{M}/\vec{x}]) : \Box C \right]\!\right] \\
\approx_Q \;& \{\, \text{definition, and non-occurrence of the } \vec{x} \text{ in } P \,\} \\
& \left[\!\left[\Delta \,;\, \cdot \vdash_{\text{int.}} P[\vec{N}/\vec{u}] : C \right]\!\right]^* \circ \pi_{\Delta}^{\Delta;\Gamma} \\
\approx_Q \;& \{\, \text{IH, } (-)^* \text{ preserves } \approx_Q \text{ (Corollary 5)} \,\} \\
& \left(\left[\!\left[\vec{u} : \vec{B} \,;\, \cdot \vdash_{\text{int.}} P : C \right]\!\right] \circ \left\langle \overrightarrow{\left[\!\left[\Delta \,;\, \cdot \vdash_{\text{int.}} \mathsf{box}\, N_i : \Box B_i \right]\!\right]} \right\rangle \right)^* \circ \pi_{\Delta}^{\Delta;\Gamma} \\
\approx_Q \;& \{\, \text{Proposition 12} \,\} \\
& \left[\!\left[\vec{u} : \vec{B} \,;\, \cdot \vdash_{\text{int.}} P : C \right]\!\right]^{\bullet} \circ \left\langle \overrightarrow{\left[\!\left[\Delta \,;\, \cdot \vdash_{\text{int.}} \mathsf{box}\, N_i : \Box B_i \right]\!\right]^*} \right\rangle \circ \pi_{\Delta}^{\Delta;\Gamma} \\
\approx_Q \;& \{\, \text{naturality of product morphism, definition} \,\} \\
& \left[\!\left[\vec{u} : \vec{B} \,;\, \cdot \vdash_{\text{int.}} P : C \right]\!\right]^{\bullet} \circ \left\langle \overrightarrow{\left[\!\left[\Delta \,;\, \Gamma \vdash_{\text{int.}} \mathsf{box}\,(\mathsf{box}\, N_i) : \Box\Box B_i \right]\!\right]} \right\rangle \\
\approx_Q \;& \{\, \text{Double box (Theorem 30), } \langle \cdot, \cdot \rangle \text{ preserves } \approx_Q \,\} \\
& \left[\!\left[\vec{u} : \vec{B} \,;\, \cdot \vdash_{\text{int.}} P : C \right]\!\right]^{\bullet} \circ \left\langle \overrightarrow{\delta_{[\![B_i]\!]} \circ \left[\!\left[\Delta \,;\, \Gamma \vdash_{\text{int.}} \mathsf{box}\, N_i : \Box B_i \right]\!\right]} \right\rangle \\
\approx_Q \;& \{\, \text{product after angled brackets} \,\} \\
& \left[\!\left[\vec{u} : \vec{B} \,;\, \cdot \vdash_{\text{int.}} P : C \right]\!\right]^{\bullet} \circ \prod_{i=1}^{n} \delta_{[\![B_i]\!]} \circ \left\langle \overrightarrow{\left[\!\left[\Delta \,;\, \Gamma \vdash_{\text{int.}} \mathsf{box}\, N_i : \Box B_i \right]\!\right]} \right\rangle \\
\approx_Q \;& \{\, \text{some projections and definition of } (-)^* \text{ and } [\![-]\!] \,\} \\
& \left[\!\left[\vec{u} : \vec{B} \,;\, \vec{x} : \vec{A} \vdash_{\mathcal{J}} \mathsf{box}\, P : \Box C \right]\!\right] \circ \left\langle \overrightarrow{\beta}, \overrightarrow{\alpha} \right\rangle
\end{aligned}
$$

Case($\Box$fix). We have that $\overrightarrow{u} : \overrightarrow{B} \,;\, \overrightarrow{x} : \overrightarrow{A} \vdash_{\mathcal{J}} \mathsf{fix}\, z\, \mathsf{in}\, P : C$, so that $\overrightarrow{u} : \overrightarrow{B} \,;\, z : \Box C \vdash_{\text{int.}} P : C$, with the result that none of the variables $\overrightarrow{x}$ occur in P. Hence $P[\vec{N}/\vec{u}, \vec{M}/\vec{x}] \equiv P[\vec{N}/\vec{u}]$, and we calculate:

$$
\begin{aligned}
& \left[\!\left[\Delta \,;\, \Gamma \vdash_{\mathcal{J}} \mathsf{fix}\, z\, \mathsf{in}\, (P[\vec{N}/\vec{u}, \vec{M}/\vec{x}]) : C \right]\!\right] \\
\approx_Q \;& \{\, \text{definitions; only } z \text{ is free in } P \,\} \\
& \left[\!\left[\cdot \,;\, z : \Box C \vdash_{\text{int.}} P : C \right]\!\right]^{\dagger} \circ\, ! \\
\approx_Q \;& \{\, \text{definition} \,\} \\
& \left[\!\left[\cdot \,;\, \cdot \vdash_{\text{int.}} \mathsf{fix}\, z\, \mathsf{in}\, P : C \right]\!\right] \circ\, ! \\
\approx_Q \;& \{\, \text{projections, definitions} \,\} \\
& \left[\!\left[\vec{u} : \vec{B} \,;\, \vec{x} : \vec{A} \vdash_{\mathcal{J}} \mathsf{fix}\, z\, \mathsf{in}\, P : C \right]\!\right] \circ \left\langle \overrightarrow{\beta}, \overrightarrow{\alpha} \right\rangle
\end{aligned}
$$

□

Theorem 64 (Soundness)**.**

1. *If* $\Delta\,;\Gamma \vdash_{int.} M = N : A$*, then we have that*

$$[\![\Delta\,;\Gamma \vdash_{int.} M : A]\!] \approx_Q [\![\Delta\,;\Gamma \vdash_{int.} N : A]\!]$$

2. *If* $\Delta\,;\Gamma \vdash_{ext.} M = N : A$*, then we have that*

$$[\![\Delta\,;\Gamma \vdash_{ext.} M : A]\!] \sim [\![\Delta\,;\Gamma \vdash_{ext.} N : A]\!]$$

Proof. By induction on the derivation of $\Delta\,;\Gamma \vdash M = N : A$. The congruence cases are clear, as is the majority of the ordinary clauses. All of these even hold up to $\approx_Q$, with the exception of $(\to \beta)$ and $(\to \eta)$, which only hold up to $\sim$. Only the modal rules remain, which we prove with direct calculation.

For $(\Box\beta)$ in the case of $\mathcal{J} = \text{int.}$ we calculate:

$$\begin{aligned}
& [\![\Delta\,;\Gamma \vdash_{\text{int.}} \mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N : C]\!] \\
\approx_Q\ & \{\,\text{definition}\,\} \\
& [\![\Delta, u{:}A\,;\Gamma \vdash_{\text{int.}} N : C]\!] \circ \langle \overrightarrow{\pi_\Delta}, [\![\Delta\,;\Gamma \vdash_{\text{int.}} \mathsf{box}\ M : \Box A]\!]\,, \overrightarrow{\pi_\Gamma}\rangle \\
\approx_Q\ & \{\,\text{Lemma 31}\,\} \\
& [\![\Delta, u{:}A\,;\Gamma \vdash_{\text{int.}} N : C]\!] \circ \langle \overrightarrow{[\![\Delta\,;\Gamma \vdash_{\text{int.}} \mathsf{box}\ u_i : \Box B_i]\!]}, [\![\Delta\,;\Gamma \vdash_{\text{int.}} \mathsf{box}\ M : \Box A]\!]\,, \overrightarrow{\pi_\Gamma}\rangle \\
\approx_Q\ & \{\,\text{Lemma 32; } [\![\Delta\,;\Gamma \vdash_{\text{int.}} x_i : A_i]\!] \stackrel{\text{def}}{=} \pi_{A_i}^{\Delta;\Gamma}\,\} \\
& [\![\Delta\,;\Gamma \vdash_{\text{int.}} N[\vec{u_i}/\vec{u_i}, M/u, \vec{x_i}/\vec{x_i}] : C]\!]
\end{aligned}$$

In the case of $\mathcal{J} = \text{ext.}$, we use Lemma 28 to write $[\![\Delta\,;\Gamma \vdash_{\text{ext.}} \mathsf{box}\ M : \Box A]\!] = [\![\Delta\,;\Gamma \vdash_{\text{int.}} \mathsf{box}\ M : \Box A]\!]$, and then the same calculation works up to $\sim$.

There remains the case of the fixpoint; let

$$g \stackrel{\text{def}}{=} [\![\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{fix}\ z\ \mathsf{in}\ M : A]\!]$$

Then $g \stackrel{\text{def}}{=} f^\dagger \circ !$, where

$$f \stackrel{\text{def}}{=} [\![\cdot\,; z : \Box A \vdash_{\text{int.}} M : A]\!]$$

But we can easily calculate that

$$\begin{aligned}
& f^\dagger \\
\sim\ & \{\,\text{definition of fixpoint (Def. 52)}\,\} \\
& f \circ \left(f^\dagger\right)^* \\
\approx_Q\ & \{\,\text{definitions}\,\} \\
& f \circ [\![\cdot\,;\cdot \vdash_{\text{int.}} \mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M) : \Box A]\!] \\
\approx_Q\ & \{\,\text{Lemma 32}\,\} \\
& [\![\cdot\,;\cdot \vdash_{\text{int.}} M[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z] : A]\!]
\end{aligned}$$

and hence $g \sim [\![\Delta\,;\Gamma \vdash_{\mathcal{J}} M[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z] : A]\!]$ by weakening (Lemma 29). $\square$

8.4 Natural and Weakly Extensional Models

In iPCF v2.0 we effected three restrictions:

- No free variables when taking intensional fixed points (except the diagonal).
- No λ-abstractions with free variables under boxes.
- IFPs only at certain types generated by A_{fix}.

We discussed at the end of §8.1 the effect that these have on the expressivity of the language, and found that it was far too strong, so we would like to examine when these can be lifted.

A iPCF v2.0 model must satisfy certain requirements in order for these restrictions to be lifted. The first one can be lifted whenever a iPCF model is *natural*. The second one can be lifted whenever a iPCF model is *weakly extensional*. Unfortunately, we are still at a loss regarding the existence of IFPs at all objects.

8.4.1 Natural iPCF v2.0 models

The first restriction we want to lift is the occurrence of free variables when taking an intensional fixed point; that is, we want to generalise the ($\Box\mathsf{fix}$) rule to

$$\frac{\Delta\,;\, z : \Box A_{\mathsf{fix}} \vdash_{\text{int.}} M : A_{\mathsf{fix}}}{\Delta\,;\,\Gamma \vdash_{\mathcal{J}} \mathsf{fix}\ z\ \mathsf{in}\ M : A_{\mathsf{fix}}}$$

We will be able to do this in *natural* models of iPCF v2.0.

Definition 57. An iPCF v2.0 model is *natural* just if

1. $(-)^\dagger$ preserves $\approx_Q$; and
2. $(-)^\dagger$ is natural up to $\approx_Q$, in the sense that for any $f : \prod_{i=1}^n QB_i \times QA \to A$ and $k : \prod_{j=1}^k QC_j \to \prod_{i=1}^n QB_i$, it is the case that

$$(f \circ (k \times id))^\dagger \approx_Q f^\dagger \circ k$$

In this kind of iPCF model, we are free to have parameters in our IFPs, and we can interpret the Löb rule by

$$[\![\Delta\,;\,\Gamma \vdash_{\mathcal{J}} \mathsf{fix}\ z\ \mathsf{in}\ M : A]\!] \stackrel{\text{def}}{=} [\![\Delta\,;\, z : \Box A \vdash_{\text{int.}} M : A]\!]^\dagger \circ \pi_\Delta^{\Delta;\Gamma}$$

The lemmas for weakening (Lem. 29) and substitution (Lem. 32) directly carry over. Naturality is only used in the appropriate cases for ($\Box\mathsf{fix}$); e.g. here is the case for substitution:

$$
\begin{array}{ll}
 & \left[\!\left[\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{fix}\ z\ \mathsf{in}\ (P[\vec{N}/\vec{u},\vec{M}/\vec{x}]) : C\right]\!\right] \\
\approx_Q & \{\,\text{definition, and non-occurrence of the } \vec{x} \text{ in } P\,\} \\
 & \left[\!\left[\Delta\,; z:\Box C \vdash_{\text{int.}} P[\vec{N}/\vec{u}] : C\right]\!\right]^{\dagger} \circ \pi_{\Delta}^{\Delta;\Gamma} \\
\approx_Q & \{\,\text{IH, } (-)^{\dagger} \text{ preserves } \approx_Q\text{, definitions}\,\} \\
 & \left(\left[\!\left[\vec{u} : \vec{B}\,; z:\Box C \vdash_{\text{int.}} P : C\right]\!\right] \circ \left\langle \overrightarrow{\left[\!\left[\Delta\,; z:\Box C \vdash_{\text{int.}} \mathsf{box}\ N_i : \Box B_i\right]\!\right]}, \pi_{z:\Box C}^{\Delta;z:\Box C} \right\rangle\right)^{\dagger} \circ \pi_{\Delta}^{\Delta;\Gamma} \\
\approx_Q & \{\,\text{weakening, } \langle\cdot,\cdot\rangle \text{ and } (-)^{\dagger} \text{ preserve } \approx_Q\,\} \\
 & \left(\left[\!\left[\vec{u} : \vec{B}\,; z:\Box C \vdash_{\text{int.}} P : C\right]\!\right] \circ \left\langle \overrightarrow{\left[\!\left[\Delta\,;\cdot \vdash_{\text{int.}} \mathsf{box}\ N_i : \Box B_i\right]\!\right] \circ \pi_{\Delta;\cdot}^{\Delta;z:\Box C}}, \pi_{z:\Box C}^{\Delta;z:\Box C} \right\rangle\right)^{\dagger} \circ \pi_{\Delta}^{\Delta;\Gamma} \\
\approx_Q & \{\,\text{naturality of products, definition}\,\} \\
 & \left(\left[\!\left[\vec{u} : \vec{B}\,; z:\Box C \vdash_{\text{int.}} P : C\right]\!\right] \circ \left(\left\langle \overrightarrow{\left[\!\left[\Delta\,;\cdot \vdash_{\text{int.}} \mathsf{box}\ N_i : \Box B_i\right]\!\right]} \right\rangle \times id\right)\right)^{\dagger} \circ \pi_{\Delta}^{\Delta;\Gamma} \\
\approx_Q & \{\,\text{naturality of } (-)^{\dagger}\,\} \\
 & \left[\!\left[\vec{u} : \vec{B}\,; z:\Box C \vdash_{\text{int.}} P : C\right]\!\right]^{\dagger} \circ \left\langle \overrightarrow{\left[\!\left[\Delta\,;\cdot \vdash_{\text{int.}} \mathsf{box}\ N_i : \Box B_i\right]\!\right]} \right\rangle \circ \pi_{\Delta}^{\Delta;\Gamma} \\
\approx_Q & \{\,\text{naturality of products, Lemma 29, projections, definitions}\,\} \\
 & \left[\!\left[\vec{u} : \vec{B}\,; \vec{x} : \vec{A} \vdash_{\mathcal{J}} \mathsf{fix}\ z\ \mathsf{in}\ P : C\right]\!\right] \circ \left\langle \overrightarrow{\beta}, \overrightarrow{\alpha} \right\rangle
\end{array}
$$

We can also calculate as usual for the fixed point. Let $g \stackrel{\text{def}}{=} [\![\Delta\,;\Gamma \vdash_{\mathcal{J}} \mathsf{fix}\ z\ \mathsf{in}\ M : A]\!]$. Then $g \stackrel{\text{def}}{=} f^{\dagger} \circ \pi_{\Delta}^{\Delta;\Gamma}$, where

$$f \stackrel{\text{def}}{=} [\![\Delta\,; z:\Box A \vdash_{\text{int.}} M : A]\!]$$

But we can easily calculate that

$$
\begin{array}{ll}
 & f^{\dagger} \\
\sim & \{\,\text{definition of fixpoint (Def. 52)}\,\} \\
 & f \circ \langle id, \left(f^{\dagger}\right)^{*} \rangle \\
\approx_Q & \{\,\text{definitions}\,\} \\
 & f \circ \langle id, [\![\Delta\,;\cdot \vdash_{\text{int.}} \mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M) : \Box A]\!] \rangle \\
\approx_Q & \{\,\text{Lemma 32}\,\} \\
 & [\![\Delta\,;\cdot \vdash_{\text{int.}} M[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z] : A]\!]
\end{array}
$$

and hence $g \sim [\![\Delta\,;\Gamma \vdash_{\mathcal{J}} M[\mathsf{box}\ (\mathsf{fix}\ z\ \mathsf{in}\ M)/z] : A]\!]$, by weakening.

8.4.2 Weakly Extensional iPCF v2.0 models, or iPCF models

In some cases, we can even rid ourselves of the distinction between intensional and extensional judgements, eventually reaching a language very close to the one with which we begun our investigation in §3. The models in which this can occur are known as *weakly extensional*.

Definition 58 (iPCF model)**.** An iPCF v2.0 model is *weakly extensional*—or, more simply, a iPCF model—just if

1. $\lambda(-)$ preserves $\approx_Q$, and
2. $\lambda(-)$ is natural with respect to $\approx_Q$, i.e.

$$\lambda(f \circ (g \times id)) \approx_Q \lambda(f) \circ g$$

Before anything else, let us immediately remark that

Lemma 33. *A weakly extensional iPCF v2.0 model can be made natural.*

Proof. We can use Theorem 59(3) to define a strong intensional fixed point combinator at every object with IFPs given by $(-)^*$. Because Q is idempotent, we can then use Theorem 59(1) to yield weak fixed point combinators, and then Theorem 59(2) to induce IFPs anew, by setting

$$f^{\dagger'} \stackrel{\text{def}}{=} \epsilon_A \circ Y \circ (\lambda(f))^*$$

As Q is idempotent and $\lambda(-)$ preserves $\approx_Q$, we have—by Lemmata 26 and 27—that $(-)^{\dagger'}$ preserves $\approx_Q$, and is natural. Thus, we can replace $(-)^\dagger$ by $(-)^{\dagger'}$, which results in a weakly extensional and natural iPCF v2.0 model. □

We can use a weakly extensional model to interpret iPCF almost as presented in its original form in §3: we only need to hold back on the IFPs, and limit them to the intensional exponential ideal A_{fix}. Of course, we call these models weakly extensional as the category of assemblies $\mathfrak{Asm}(A)$ constitutes such a model whenever A is a weakly extensional PCA: see §5.2.5.

It is not a difficult calculational exercise to show that the main lemmata in this chapter, i.e. weakening (Lemma 29) and substitution (Lemma 32), hold up to $\approx_Q$, with no distinction between extensional and intensional judgements. This happens because the interpretation of all the main language constructs, i.e. $\lambda(-)$ and $(-)^*$, preserve $\approx_Q$.

The only exception is the soundness theorem, which only holds up to $\sim$, and it does so with good reason. Firstly, we do not expect the equational behaviour of the constants (naturals, booleans) to be *intensional*: on account of the language being partial, we expect both of these objects to have highly intensional structure. But even if they did not, the cartesian closed equations can realistically be expected to only hold extensionally (especially η).

However, if asked to name a weakly extensional model of iPCF, we might find ourselves at a loss. The paradigmatic example of a weakly extensional PCA is certainly $\Lambda/=_\beta$, the closed terms of the untyped λ-calculus quotiented by β-equivalence. The construction of $\mathfrak{Asm}(\Lambda/=_\beta)$ and the exposure $\Box$ on it (§5.2) are parametric in A, so all that remains is to construct IFPs. But even if we were to construct them, we might think that we have just engaged in an exercise in futility. The elements of $\Box A$ would be pairs (a, x) where $x \in \|a\|_A$ is a realizer of a point $x \in |A|$. But x would be an equivalence class $[P]_{=_\beta}$ of an untyped λ-term, which would be an object that is 'too extensional' for the level at which we have been aiming: IFPs would merely be ordinary fixed points of λ-terms!

That leaves us with three choices:

(i) Seek the Holy Grail PCA $\mathcal{HG}$ that is weakly extensional, yet sufficiently intensional for IFPs w.r.t. to $\Box$ in $\mathfrak{Asm}(\mathcal{HG})$ to be of interest: this seems rather difficult, perhaps to the point of being a contradiction in terms.

(ii) Try to redefine $\Box : \mathfrak{Asm}(A) \not\to \mathfrak{Asm}(A)$ in the case of a weakly extensional PCA A. In the previous case we were content to use realizers as the 'true intensions.' But how can we proceed this time? One attempt in the case of A consisting of equivalence classes would be to try to 'pick' a representative of each class. But then for $\Box$ to respect composition these would have to be 'compatible' up to composition, which seems impossible.

(iii) A third option would be to be in a position to accept that weakly extensional realizers, such as the terms of $\Lambda/=_\beta$, are sufficiently intensional. This could be the case of we require a lot of extensionality at the assembly level, perhaps to the point where a fixed point of an untyped λ-term seems a rather intensional affair. This is again in the spirit described in the introduction (§1.1), where intensionality is argued to only be defined *only relative to the extensional equality.*

It is slowly beginning to seem that, in the most intensional of settings, the restrictions we have demanded of iPCF v2.0 are somehow indispensable. We will produce some further evidence for that in the process of proposing a general method for constructing IFPs at the end of the next section (§8.5): the simplest naturality argument we can concoct already requires weak extensionality. This may not be proof, but nonetheless it is solid evidence, as the method described therein is particularly useful in constructing IFPs for very intensional PCA of classical computability K_1 (§8.6).

8.5 Building IPWPSs categorically

In this section we shall show how to build IPWPSs from more basic constructs. If given sufficiently many IPWPSs in a P-CCC which is equipped with an idempotent comonadic exposure, then we can use our Parametric Intensional Recursion Theorem (Theorem 60) to build a iPCF model.

The two main ingredients at our disposal will be a certain kind of *retraction*, and a certain kind of *enumeration*. Both of these are unlike the ones that have been considered before, and they make deep use of the theory of exposures as developed in this thesis. Consequently, they have a very intensional flavour.

Our enumerations will be arrows of the form $X \to A$, where A will be the object *enumerated*, and X the object of 'indices.' To this we will add a *factorisation property*, which will be evocative of, or even directly related to, the idea of *path surjections* as briefly discussed in §6.3.

To this, we will add a special notion of *intensional retraction*, which allows one to represent 'code' for objects of the form X^{QZ} (for any Z) as a sort of retract of X, but only up to extension. We will use these contraptions to formulate an *inductive argument* that constructs IPWPSs at all objects contained in the *intensional exponential ideal* I generated by the 'grammar'

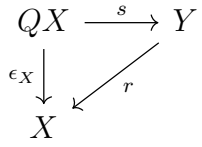
$$I \quad ::= \quad \mathbb{B} \quad | \quad \mathbb{N} \quad | \quad I^{QZ}$$

Throughout this section, let us fix a cartesian closed P-category $\mathfrak{C}$, and a comonadic exposure (Q, ϵ, δ) on it.

QX is an object which holds information about 'codes' of objects of type X. These 'codes' can often be encoded as very simple first-order data in an object Y; for example, Y could be the natural numbers object. Sometimes we might be able to retrieve the original 'code' in QX from Y, making QX a retract of Y. But, in some cases, we might not: we will only manage to 'interpret' the data in Y as data

in X, yielding the same extension—but not the same code—as the original one. This situation is precisely captured by intensional retractions.

Definition 59. X is an *intensional retract* of Y (w.r.t. Q) whenever there is a pair of arrows $s : QX \to Y$ and $r : Y \to X$ such that

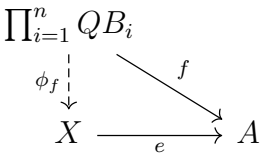


commutes up to $\sim$. We call X an *intensional retract* of Y.

The second concept that is central is that of *enumeration*. As hinted in §6.3 in the context of various forms of surjections, we can think of arrows $X \to A$ as 'enumerating' the elements of A by 'indices' in X. We will require the existence of a particular type of *path surjection* (see §6.3), namely one whose 'path' object (denoted N in §6.3) is an intensional context of the form $\prod_{i=1}^{n} QB_i$.

Definition 60. An object $A \in \mathfrak{C}$ is (Q, X)-enumerated by $e : X \to A$ just if it is a $(\prod_{i=1}^{n} QB_i)$-path surjection for every finite list of objects $\overrightarrow{B_i}$.

That is: for every arrow $f : \prod_{i=1}^{n} QB_i \to A$ there is at least one arrow $\phi_f : \prod_{i=1}^{n} QB_i \to X$, not necessarily unique, that makes the diagram



commute. We often call the arrow $e : X \to A$ a (Q, X)-enumeration, and say that A is (Q, X)-enumerable.

We are very fond of (Q, X)-enumerations for two reasons. The first reason is that they are quite easy to construct: the domain of ϕ_f provides enough intensional information in the QB_i's. It would be essentially impossible to construct something of the sort given merely a $\prod_{i=1}^{n} B_i$. Intuitively, the reason is that the B_i's are available extensionally, i.e. as a kind of oracle to which we can pose a (probably finite) number of questions. It would be unthinkable to *internally* extract an 'index' for the enumeration $e : X \to A$ in such a situation.

The second reason is simply because—under one mild assumption—they directly give rise to IPWPSs.

Lemma 34. *If*

- *A is (Q, X)-enumerated by $e : X \to A$; and*
- *X^{QX} is an intensional retract of X.*

then there is an intensional parametric weak-point surjection $p : QX \times QX \to A$.

Proof. Let (s, r) witness X^{QX} as an intensional retract of X. Define

$$p \stackrel{\text{def}}{=} QX \times QX \xrightarrow{\epsilon \times id} X \times QX \xrightarrow{r \times id} X^{QX} \times QX \xrightarrow{\mathsf{ev}} X$$

We want to show that this is a IPWPS. Let $f : \prod_{i=1}^{n} QB_i \times QX \to A$. Then f can be written as

$$\prod_{i=1}^{n} QB_i \times QX \xrightarrow{\phi_f} X \xrightarrow{e} A$$

as $e : X \to A$ is a (Q, X)-enumeration. We can λ-abstract the index, and take its co-Kleisli lifting to obtain $(\lambda(\phi_f))^* : \prod_{i=1}^{n} QB_i \to Q(X^{QX})$. Post-composing with the lifted section $s^* : Q(X^{QX}) \to QX$ yields an 'index'

$$x_f \stackrel{\text{def}}{=} s^* \circ (\lambda(\phi_f))^* : \prod_{i=1}^{n} QB_i \to QX$$

w.r.t to p:

$$\begin{aligned}
& p \circ \langle s^* \circ (\lambda(\phi_f))^*, a \rangle \\
\sim\ & \{ \text{definition of } p, \text{ products after brackets} \} \\
& e \circ \mathsf{ev} \circ \langle r \circ \epsilon \circ s^* \circ (\lambda(\phi_f))^*, a \rangle \\
\sim\ & \{ \text{Prop. 13, int. retract.} \} \\
& e \circ \mathsf{ev} \circ \langle \epsilon \circ (\lambda(\phi_f))^*, a \rangle \\
\sim\ & \{ \text{Prop. 13 again} \} \\
& e \circ \mathsf{ev} \circ \langle \lambda(\phi_f), a \rangle \\
\sim\ & \{ \text{cartesian closure} \} \\
& e \circ \phi_f \circ \langle id, a \rangle \\
\sim\ & \{ e \text{ is a } (Q, X)\text{-enumeration} \} \\
& f \circ \langle id, a \rangle
\end{aligned}$$

□

So much for the construction of IPWPSs given enumerations. What about higher types? In fact, the following lemma shows that, if X is sufficient to intensionally encode X^{QZ}, then we can 'lift' a (Q, X)-enumeration $e : X \to A$ to (Q, X)-enumerate A^{QZ}. This is where our previous notion of intensional exponential ideal comes from.

Lemma 35. *Suppose that*

- $A \in \mathfrak{C}$ *is* (Q, X)*-enumerated by* $e : X \to A$*, and*
- X^{QZ} *is an intensional retract of* X*.*

Then A^{QZ} *is* (Q, X)*-enumerable.*

Proof. Take $f : \prod_{i=1}^{n} QB_i \to A^{QZ}$. Then $f \sim \lambda(g)$ for some $g : \prod_{i=1}^{n} QB_i \times QZ \to A$. By IH, we have some $\phi_g : \prod_{i=1}^{n} QB_i \times QZ \to X$ such that

$$\begin{array}{ccc} \prod_{i=1}^{n} QB_i \times QZ & & \\ {\scriptstyle \phi_g}\,\vdots\downarrow & \searrow^{g} & \\ X & \xrightarrow[e]{} & A \end{array}$$

If we apply $\lambda(-)$ to this triangle, we obtain

$$\begin{array}{ccc} \prod_{i=1}^{n} QB_i & & \\ {\scriptstyle \lambda(\phi_g)}\downarrow & \searrow^{f} & \\ X^{QZ} & \xrightarrow[e^{QZ}]{} & A^{QZ} \end{array}$$

We can now rewrite $\lambda(\phi_f) \sim \epsilon \circ (\lambda(\phi_f))^*$ using Proposition 13:

$$\begin{array}{ccc} \prod_{i=1}^{n} QB_i & \xrightarrow{f} & A^{QZ} \\ {\scriptstyle (\lambda(\phi_g))^*}\downarrow & & \uparrow{\scriptstyle e^{QZ}} \\ Q\left(X^{QZ}\right) & \xrightarrow[\epsilon]{} & X^{QZ} \end{array}$$

But X^{QZ} is an intensional retract of X, so we can rewrite ϵ like so:

$$\begin{array}{ccccc} \prod_{i=1}^{n} QB_i & & \xrightarrow{\qquad f \qquad} & & A^{QZ} \\ {\scriptstyle (\lambda(\phi_g))^*}\downarrow & & & & \uparrow{\scriptstyle e^{QZ}} \\ Q\left(X^{QZ}\right) & & & & X^{QZ} \\ & \searrow^{s} & & \nearrow_{r} & \\ & & X & & \end{array}$$

Hence, by defining

$$e' \stackrel{\text{def}}{=} e^{QZ} \circ r : X \to E^{QZ}$$

we have that E^{QZ} is (Q, X)-enumerated by e', and the index of f is

$$\phi_f \stackrel{\text{def}}{=} s \circ (\lambda(\phi_g))^*$$

□

Theorem 65. *Suppose that*

- $\mathbb{B}, \mathbb{N} \in \mathfrak{C}$ *are* (Q, X)*-enumerable, and that*
- X^{QZ} *is an intensional retract of* X *for any* $Z \in \mathfrak{C}$*.*

Then any object in the intensional exponential ideal generated by

$$I \quad ::= \quad \mathbb{B}_\perp \quad | \quad \mathbb{N}_\perp \quad | \quad I^{QZ}$$

for any $Z \in \mathfrak{C}$ *has parametric IFPs.*

Proof. We can show by induction that every object generated by I is (Q, X)-enumerable: the base cases are assumptions, and the inductive step is provided by Lemma 35. Then by Lemma 34 there is an intensional weak-point surjection $p_I : QX \times QX \to I$ for every I. Finally, by the Parametric Intensional Recursion Theorem (Theorem 60) each I has IFPs. □

Naturality

It is again worth asking about the necessary premises that are sufficient for us to conclude that the IPWPSs build in this section are *natural* in the sense of §7.4. According to our previous results, we need $x_h \circ g \approx_Q x_{h \circ (g \times id)}$. We can show that, under certain assumptions, the construction of a IPWPS from an enumeration in Lemma 34 maintains it. Calculating suffices to elicit the necessary assumptions:

$$\begin{aligned}
& x_h \circ g \\
\approx_Q \quad & \{ \text{definition} \} \\
& s^* \circ (\lambda(\phi_h))^* \circ g \\
\approx_Q \quad & \{ \text{idempotence} \} \\
& s^* \circ (\lambda(\phi_h) \circ g)^* \\
\approx_Q \quad & \{ \lambda(-) \text{ natural up to } \approx_Q \} \\
& s^* \circ (\lambda(\phi_h \circ (g \times id)))^* \\
\approx_Q \quad & \{ \lambda(-) \text{ preserves } \approx_Q,\ \phi_h \circ (g \times id) \approx_Q \phi_{h \circ (g \times id)} \} \\
& s^* \circ (\lambda(\phi_{h \circ (g \times id)}))^*
\end{aligned}$$

which by definition is $x_{h\circ(g\times id)}$. Hence,

Corollary 8. *If $\lambda(-)$ preserves $\approx_Q$ and is natural up to it, and moreover the (Q,X)-enumeration $e : X \to A$ satisfies $\phi_h \circ (g \times id) \approx_Q \phi_{h\circ(g\times id)}$ then the IPWPS $p : QX \times QX \to A$ constructed in Lemma 34 is natural in the sense described at the end of §7.4.*

In turn, an easy calculation shows that if a (Q,X)-enumeration is 'natural' in the above sense then the inductive step of Lemma 35 maintains this property—if $\lambda(-)$ preserves $\approx_Q$! By induction, all the IPWPSs constructed in Theorem 65 are natural.

But we are—so to speak—already preaching to the choir: we have made use of the naturality and preservation of $\lambda(-)$ up to $\approx_Q$. Thus, the model is already *weakly extensional*, and Lemma 33 already provides a way to make it natural.

At this point, it is beginning to seem like there is no way around weak extensionality. On the one hand, generalising our results to yield natural IFPs seems to already already weak extensionality. On the other hand, we cannot conceive of a weakly extensional model in which IFPs really are more informative than EFPs. *Intensionality and weak extensionality seem to be at odds with each other.* If we take all of this into account, limiting the fixed point rule to admit no free variables other than the 'diagonal' one seems almost forced in the most intensional of settings.

8.6 $\mathfrak{Asm}(K_1)$ as a model of iPCF v2.0

Let us revisit the construction of the P-category of assemblies $\mathfrak{Asm}(K_1)$ on the PCA K_1 of classical computability, as described in §5.2. We shall prove that it is a iPCF model, as per Definition 56. It certainly comes with all the prerequisite structure, so all we need to check is whether it has (natural) intensional fixed points at all the relevant objects. We shall construct them using Theorem 65.

Before we proceed with the construction, we need to define some objects of interest. First, we define the assembly $\mathbb{N} \in \mathfrak{Asm}(K_1)$ of *natural numbers* by

$$|\mathbb{N}| \stackrel{\text{def}}{=} \mathbb{N}, \qquad \|n\|_{\mathbb{N}} \stackrel{\text{def}}{=} \{n\}$$

We also need to define the assembly $\mathbb{N} \in \mathfrak{Asm}(K_1)$ of *booleans* by

$$|\mathbb{B}| \stackrel{\text{def}}{=} \{\mathsf{ff}, \mathsf{tt}\}, \qquad \|b\| \stackrel{\text{def}}{=} \begin{cases} \{0\} & \text{for } b = \mathsf{ff} \\ \{1, 2, \dots\} & \text{for } b = \mathsf{tt} \end{cases}$$

At this point we urge the reader to recall the definition of the *lifted assemblies* $\mathbb{N}_\perp$ and $\mathbb{B}_\perp$ that we defined for K_1 in §5.2.3.

Proposition 14. *For any assembly $Z \in \mathfrak{Asm}(K_1)$ there is an intensional retraction*

$$\begin{array}{ccc} Q\left(\mathbb{N}_\perp{}^{QZ}\right) & \xrightarrow{\ s\ } & \mathbb{N}_\perp \\ {\scriptstyle \epsilon_X}\downarrow & \swarrow_{r} & \\ \mathbb{N}_\perp{}^{QZ} & & \end{array}$$

Proof. For the section part, we define

$$\begin{aligned} s : \left|Q\left(\mathbb{N}_\perp^{QZ}\right)\right| &\longrightarrow |\mathbb{N}_\perp| \\ (f, n) &\longmapsto n \end{aligned}$$

which is realized by $\lambda^* nx.n$. For the retraction we define

$$\begin{aligned} r : |\mathbb{N}_\perp| &\longrightarrow \left|\mathbb{N}_\perp^{QZ}\right| \\ n &\longmapsto f_n \\ \perp &\longmapsto ((z, a) \mapsto \perp) \end{aligned}$$

where

$$\begin{aligned} f_n : |QZ| &\longrightarrow |\mathbb{N}_\perp| \\ (z, a) &\longmapsto \begin{cases} m \text{ if } n \cdot a \cdot \overline{0} \simeq m \\ \perp \text{ if } n \cdot a \cdot \overline{0} \uparrow \end{cases} \end{aligned}$$

f_n is realized by $\lambda^* ax.\ n \cdot a \cdot \overline{0}$, and hence r itself is realized by $\lambda^* wax.\ w \cdot \overline{0} \cdot a \cdot \overline{0}$.

It remains to show that if $n \in \|f\|_{\mathbb{N}_\perp^{QZ}}$, then $f_n = f$. In order to show this, the first thing we have to note is that $\mathbb{N}_\perp$ is a *modest set*: a realizer can only realize one element of $|\mathbb{N}_\perp| \stackrel{\text{def}}{=} \mathbb{N} + \{\perp\}$. Thus, if $n \cdot a \in \|x\|_{\mathbb{N}_\perp}$, then necessarily $f(z, a) = x$: if $f(z, a) = x'$, then we must have $n \cdot a \in \|x'\|_{\mathbb{N}_\perp}$, so $x = x'$. We can thus set up a chain of equivalences,

$$f(z, a) = m \iff n \cdot a \in \|m\|_{\mathbb{N}_\perp} \iff n \cdot a \cdot \overline{0} \simeq \overline{m} \iff f_n(z, a) = m$$

and, similarly,

$$f(z, a) = \perp \iff n \cdot a \in \|\perp\|_{\mathbb{N}_\perp} \iff n \cdot a \cdot \overline{0} \uparrow \iff f_n(z, a) = \perp$$

□

Proposition 15. $\mathbb{N}_\perp$ *is* $(\square, \mathbb{N}_\perp)$*-enumerable.*

Proof. Suppose $(f, r) : \prod_{i=1}^{n} QB_i \to \mathbb{N}_\perp$. This means that, if $a_i \in \|b_i\|_{B_i}$ for all i then

$$r \cdot \langle a_1, \ldots, a_n \rangle \in \|f((b_1, a_1), \ldots, (b_n, a_n))\|_{\mathbb{N}_\perp}$$

where

$$\langle a \rangle \stackrel{\text{def}}{=} a$$
$$\langle a_1, \ldots, a_{m+1} \rangle \stackrel{\text{def}}{=} \mathsf{pair}\ \langle a_1, \ldots, a_m \rangle\ a_{m+1}$$

We can then define $\phi_{(f,r)} \stackrel{\text{def}}{=} (g_r, s)$ where

$$g_r : \left| \prod_{i=1}^{n} QB_i \right| \longrightarrow |\mathbb{N}_\perp|$$
$$((b_1, a_1), \ldots, (b_n, a_n)) \longmapsto \langle r, a_1, \ldots, a_n \rangle$$

which is obviously realizable. Then, define $e_{\mathbb{N}_\perp} : \mathbb{N}_\perp \to \mathbb{N}_\perp \stackrel{\text{def}}{=} (h, v)$ by

$$h : |\mathbb{N}_\perp| \longrightarrow \mathbb{N}_\perp$$
$$c \longmapsto \begin{cases} \perp, \text{ if } (c)_1 \cdot \langle (c)_2 \ \ldots (c)_{n+1} \rangle \cdot \overline{0} \uparrow \\ m, \text{ if } (c)_1 \cdot \langle (c)_2 \ \ldots (c)_{n+1} \rangle \cdot \overline{0} \simeq m \end{cases}$$
$$\perp \longmapsto \perp$$

where $(\langle a_1, \ldots, a_m \rangle)_i \stackrel{\text{def}}{=} a_i$. This is realizable, and it is easy to show that the required diagram

$$\begin{array}{ccc} \prod_{i=1}^{n} QB_i & & \\ {\scriptstyle \phi_f}\downarrow & \searrow^{f} & \\ \mathbb{N}_\perp & \xrightarrow[e_{\mathbb{N}_\perp}]{} & \mathbb{N}_\perp \end{array}$$

commutes. □

Proposition 16. $\mathbb{B}_\perp$ *is* $(\square, \mathbb{N}_\perp)$*-enumerable.*

Proof. The same proof as for $\mathbb{N}_\perp$ almost works: we only need to slightly alter the definition of $e_{\mathbb{N}_\perp} = (h, v) : \mathbb{N}_\perp \to \mathbb{N}_\perp$ to make it into an arrow $e_{\mathbb{B}_\perp} \stackrel{\text{def}}{=} (h', v) : \mathbb{N}_\perp \to \mathbb{B}_\perp$ where

$$h' : |\mathbb{N}_\perp| \longrightarrow \mathbb{B}_\perp$$
$$c \longmapsto \begin{cases} \perp, \text{ if } (c)_1 \cdot \langle (c)_2 \ \ldots (c)_{n+1} \rangle \cdot \overline{0} \uparrow \\ \mathsf{ff}, \text{ if } (c)_1 \cdot \langle (c)_2 \ \ldots (c)_{n+1} \rangle \cdot \overline{0} \simeq 0 \\ \mathsf{tt}, \text{ if } (c)_1 \cdot \langle (c)_2 \ \ldots (c)_{n+1} \rangle \cdot \overline{0} \simeq m \neq 0 \end{cases}$$
$$\perp \longmapsto \perp$$

The same realizer works. □

We thus conclude that

Theorem 66. *$\mathfrak{Asm}(K_1)$ has intensional fixed points at the intensional exponential ideal generated by the 'grammar'*

$$I \quad ::= \quad \mathbb{B} \quad | \quad \mathbb{N} \quad | \quad I^{QZ}$$

for any $Z \in \mathfrak{Asm}(K_1)$.

Proof. Use Propositions 14, 16, 15 to fulfil the premises of Theorem 65. □

Chapter 9

Conclusions & Future Work

We briefly peruse what has been achieved in this thesis:

(i) We first attempted to pin down the informal meaning of *intensionality* as the possibility of *non-functional operations* (§1), where non-functionality is understood in the presence of some ambient extensional equality.

(ii) Then, we carefully reviewed the distinction between *extensional and intensional recursion in computability theory* (§2).

(iii) This led us to the formulation of a *higher-order intensional and reflective programming language*, in the form of a modal λ-calculus called Intensional PCF. This language included genuinely 'non-functional' operations, and typed intensional recursion through Löb's rule. We showed that, if intensionality/'non-functionality' is limited to modal types, then iPCF is consistent (§3).

(iv) In §4 we began the search for a categorical semantics for that calculus. We first argued that 1-category theory is not the correct mathematical setting to speak of intensionality. As an alternative, we proposed P-category theory. We then proceeded to introduce *exposures*—a new P-categorical construct which abstracts the idea of intensional devices, e.g. Gödel numberings. Then, drawing inspiration from comonads, we developed the theory of exposures.

(v) The claim that exposures are abstractions of intensional devices was substantiated by carefully constructing three rather different examples in §5.

The first one comprises an actual Gödel numbering on Peano Arithmetic.

The second one was drawn from higher-order computability/realizability. If we think of realizers as machine code, this main example made clear the idea that *exposures expose the implementation.*

The third one is based on ideas from homological algebra, and constitutes a first attempt at recognising the occurrence of intensionality in fields beyond logic and computability.

(vi) Then, in §6, we reformulated to intensional recursion, and showed that it can be captured through exposures. We proved abstract analogues of classic intensional results, like Gödel's First Incompleteness Theorem, Tarski's Undefinability Theorem, and Rice's Theorem. These results lend credibility to the idea that exposures are a toolkit where the fine structure of results with an intensional flavour can be described.

The culmination of this chapter was the Intensional Recursion Theorem (Theorem 54), which set out conditions that guarantee the existence of intensional fixed points. The Intensional Recursion Theorem can be thought of as an abstract version of Kleene's Second Recursion Theorem.

(vii) In the final two chapters (§7, §8) we brought iPCF and exposures together. After some technical development, we showed that a restriction of iPCF, called iPCF v2.0, can be interpreted in a cartesian closed P-category equipped with a product-preserving comonadic exposure, and IFPs at appropriate objects.

We then discussed the cases in which the restrictions that plague iPCF v2.0 can be waived. However, we argued that there are good reasons indicating that lifting those restrictions is at odds with intensionality, at least in the way in which we understand it.

We closed the thesis by proving that $\mathfrak{Asm}(K_1)$, the P-category of assemblies on K_1, is a model of iPCF v2.0. As K_1 is a PCA based on classical computability, this means that iPCF v2.0 is adequate for constructing indices in classical computability theory: it is a typed 'intensional metaprogramming' language for writing programs in the style of Jones [1997], albeit with limited expressivity.

In the rest of this concluding chapter, we will try to evaluate these achievements. We would like to focus on four aspects in particular:

- Is intensionality really just the ability to have non-functional operations?
- Are iPCF and iPCF v2.0 adequate intensional and reflective languages?
- How do exposures compare with alternative 'theories of intensionality'?
- Have we managed to elucidate the mysterious Second Recursion Theorem of Kleene?

9.1 Is intensionality really just non-functionality?

The formalisation of Frege's ideas of sense and reference, which we discussed in §1.1, is an old problem. Even though many have tried, there does not seem to be a definitive account. Some even question whether such a definitive account *should* exist.

Moschovakis [1993] defines the sense of logical formula as the (possibly infinitary) algorithm that the syntax of a (first-order) formula suggests. This is formalised using Moschovakis' own *theory of recursive algorithms*.

Occupying some middle ground, Abramsky [2014] draws on a long tradition of *programming language semantics*. In a sense, he introduced what one could call the *spectrum of intensionality*: some kinds of semantics is more *intensional* than others, in that the mathematical objects that comprise it contain strictly more information about the computation that is being modelled, e.g. an account of the interactions that take place when a program is run. The gist is that, by moving to more refined semantics, more can be captured, even though a price might have to be paid, perhaps in the form of *quotienting the model*. However, we believe that *op. cit.* is permeated by a preliminary form of the ideas explicitly developed in this thesis.

A third opinion is given by Girard et al. [1989]: "the sense contains the denotation, at least implicitly." Girard proposes a study of the *invariants of syntax*, in a vein inspired by proof theory. Some interpret this statement as taking the extreme viewpoint that 'syntax = sense,' but it becomes clear in [Girard, 2011] that the author is simply proposing the study of new, unorthodox proof-theoretic structures.

In this thesis we avoided this lengthy debate by *defining* intensionality to mean 'anything finer than what we call extensional equality.' We like to view this as *purely mathematical*, and *philosophically agnostic*. We have merely demonstrated in §3 that modal types allow one to treat their elements as pure syntax.

In the development of exposures in §4, and then in the intensional semantics of §8, it became clear that this view of equality is *modular*. The modal types allow one to introduce equalities in a *controlled* fashion: we began with no equations in iPCF (§3), and gradually reached a set of equations in iPCF v2.0 (§8) which seem to mirror intensional equality, as defined by the exposure. Whether that is a precise reflection can be shown through a *completeness theorem*, which we conjecture to hold. In turn, our flavour of exposure (cartesian, product-preserving, weakly cartesian closed, etc.) determines which equations intensional equality will satisfy. We believe that the modularity of this framework is a serious advantage that makes it adaptable to all sorts of settings, and all levels of fine-grain intensional information.

9.2 How expressive is iPCF?

The first two objectives of this thesis that we discussed in §1.2 were the clarification of the *intensional and reflective programming* paradigms.

The development of iPCF elucidated the fact that we can indeed treat terms at modal types as if they were 'pure syntax' (up to α-equivalence), and make arbitrary decisions on them *in a typed manner*. This shows that, if we comprehend intensional programming as entailing non-functional behaviour, then there is a type theory in which this ability is provably consistent. As we remarked in the introduction, there have been many similar attempts in the past, but all were in one way or another problematic: some led to particularly complicated languages with unclear semantics, like those of Smith [1982, 1984]; others, like Gabbay and Nanevski [2013], were close to ours, but proposed semantics which are—unfortunately—inconsistent. A third class led to impossibility theorems, e.g. Wand [1998].

Our work decisively resolved many of these issues: once a modal typing discipline is established, then we are perfectly capable of accommodating both non-functional behaviour as well as reflexivity. If we limit these behaviours to the modal types, and use the typing system to stop the flow from the extensional region of the language to the modal types, then we get a consistent language. We also believe that we are the first to directly tie intensionality and reflexivity together, as we think they should be, if we are to use reflexivity in any interesting manner.

Nonetheless, as we hinted in §3.6, iPCF can only be considered a proof-of-concept. We have deliberately decided not to concern ourselves with the task of finding good sets of intensional primitives, but merely with the possibility of crafting a setting in which this is possible. We believe that finding good intensional primitives is a particularly hard problem, with close connections to both *metaprogramming* and *higher-order computability*. Furthermore, finding *other models* is likely to prove challenging.

9.2.1 Metaprogramming

As we discussed in the introduction, metaprogramming is a difficult task that dates back to the work of the Lisp community. The area has recently witnessed a resurgence of interest, leading to the *International Summer School on Metaprogramming* that took place at Robinson College, Cambridge (8-12 August 2016) around the time that the author began drafting this thesis. It is quite clear that a good foundation for metaprogramming is still lacking: see the work of Berger [2016] for a recent discussion.

At this point we should remark that intensional operations are *not* the main subject of that area: metaprogramming is about *constructing* code dynamically from its fragments, and not *destructing* it, as we are wont to do with intensionality.

A common issue in metaprogramming is that of *manipulating code with open variables (= free variables).* This is known to be a rather painful limitation of the $\mathsf{S4}$-based language of Davies and Pfenning [1996] on which iPCF is based. In order to overcome this, Davies [1995, 1996, 2017] developed a λ-calculus based on another modality that is reminiscent of the 'next' operator of Linear Temporal Logic (LTL). This language has explicit annotations of the *stage* at which each computation is taking place, and is also able to handle open code. This led to a flurry of developments, and in particular the very influential work of Taha and Sheard [1997, 2000] on MetaML, and then the environment classifiers of Taha and Nielsen [2003], which have recently been improved by Tsukada and Igarashi [2010]. See [Kavvos, 2016, §6] for further references, and for a discussion of the modal aspects used.

In iPCF, the restriction of *no open code* does not only occur as it did before (only modal variables under boxes), but it also vengefully reappears in the case of intensional operations: we saw in §3.3 that, unless the terms on which we operate intensionally are closed, we risk inconsistency. The author's colleague, Mario Alvarez-Picallo, has begun some preliminary work in intensionality in calculi like Davies', which we discuss further in §9.3.

However, to achieve any meaningful notion of intensional primitives, much more than simple modal types is required. Consider, for example, a constant **deapp** that attempts to take apart an application, in the sense that

$$\mathbf{deapp}\ (\mathsf{box}\ (M\ N)) = \langle \mathsf{box}\ M, \mathsf{box}\ N \rangle$$

(where we have also assumed products in the language). This cannot meaningfully be typed in the simple modal setting. In fact, it seems that we need *existential types* in order to assign a type to this, as the domain of M is unknown. For example, the type would be something like

$$\mathbf{deapp} : \Box B \to \exists A.\ \Box(A \to B) \times \Box A$$

However, this would take us deep into the waters of *second-order modal logic*, and we are not aware of any previous work on that front.

Similar second-order type systems like this have been suggested in the context of Barry Jay's *factorisation calculi.* These are combinatory calculi that admit intensional operations at 'face value.' The trick by which disaster is avoided is that of having

intensional operations act only on normal forms, e.g. a partially applied **S** combinator (**S***PQ*), which they are able to take apart (*factorise*) in order to reuse its immediate subterms P and Q: see Jay and Given-Wilson [2011]. In subsequent work by Jay and Palsberg [2011] it was shown that such a system admits a second-order Curry-style typing similar the one shown above, but without the modalities.

9.2.2 Higher-Order Computability

This is perhaps the point of view from which the present work originates, but also the least developed in this thesis. We discussed some possible applications of our ideas on intensional higher-order computation in §1.2.3. Unfortunately, the scope of this thesis could not be stretched to contain more material in this direction. Nevertheless, the author has a presentiment that any truly novel understanding of intensional higher-order computation will come simultaneously with the development of intensional primitives for a language like iPCF.

9.2.3 iPCF, iPCF v2.0, and their models

Even though we begun with the model of assemblies on K_1 in mind, it gradually became apparent that things were not quite that simple. In §8 we ran into severe difficulties when trying to force $\mathfrak{Asm}(K_1)$ to be a model of iPCF, which we related to three fundamental problems:

1. free variables in intensional fixed points;
2. free variables in abstractions of quoted terms, a.k.a *no higher-order intensional programming*; and
3. the construction of intensional fixed points at all types.

Whereas the third problem is largely technical, the other two are quite serious points against our argument that iPCF v2.0 is a good 'language of indices.' We argued extensively in §8.1 that, indeed, both of these problems seem more-or-less natural from a computability theory point-of-view. However, naturality and substitution are fundamental aspects of any λ-calculus, and the fact that we have to disallow them fundamentally reduces the expressiveness of what we have achieved.

It thus follows that we need to consider more candidate models of iPCF. We can go about this task in three ways:

(i) We can study $\mathfrak{Asm}(K_1)$ more closely; or

(ii) we can look for other PCAs A such that $\mathfrak{Asm}(A)$ is an interesting *natural/weakly extensional* model; or

(iii) we can look elsewhere.

Regarding the first option, we should remark that we *did not show* that $\mathfrak{Asm}(K_1)$ is *not* natural. We very strongly believe that it is not weakly extensional, although we have no evidence for that either. Disproving either of these assertions is likely to be a very cumbersome exercise, and it is perhaps best that it be automated/computer-assisted. If naturality is shown to not be the case, then we should perhaps redefine $\mathsf{box}\ (-)$ to enclose only *completely closed* terms, with no free variables at all, whether modal or intuitionstic. This would directly lead us to some kind of intensional combinator language, as also described at the end of §8.1, which would not be very far from what is already the case with indices in computability theory. It may be that the 'nature' of K_1 simply has this shape.

However, even if $\mathfrak{Asm}(K_1)$ were to unexpectedly prove to be natural, the key to Davies-Pfenning style staged metaprogramming—which is an expressive improvement over indices in computability theory—is precisely the free variables under quoted λ-abstractions. This very urgently requires *weak extensionality*, as remarked in §8.4. In that section, we also discussed the possibility of finding some weakly extensional PCA A that can be informative in terms of intensional programming. We concluded that the said task is likely very difficult.

It is not inconceivable that there could be another source of models for iPCF: perhaps a change in perspective is required, and this change may be one that involves moving away from ideas sourced from realizability, and more towards the emergent theory of metaprogramming. This is also related to some ideas that we will discuss in the next section, regarding alternative proposals for modelling the phenomenon of intensionality at large.

Finally, as soon as more models of iPCF are identified, there are many interesting questions that need to be answered. First, showing a *completeness theorem* for iPCF interpreted with exposures should be a primary goal. In a sense, a completeness theorem will demonstrate that our categorical structure indeed corresponds to what we type-theoretically understand as intensional types. This should be investigated as a priority. Secondly, a more careful approach to the available intensional operations should be taken, and some form of *adequacy* theorem for iPCF should be shown. We are not exactly sure what form this should take, but it should certainly be much more involved than the adequacy statements concerning PCF. This issue is likely

to be deeply intertwined with the *intensional primitives* we discussed this section. Thirdly, there must be many interesting results concerning the mismatch between iPCF and its various models. This is very much the case with PCF, beginning with the work of Plotkin [1977] on domain-theoretic models, and all the way to the work of Escardó [1999] on metric models. However, the intensional case is likely to be much richer, detailed, and tricky.

9.3 Exposures vs. other theories of intensionality

Exposures attempt to abstract the general notion of intensional devices, of which Gödel numberings or numberings of partial recursive functions are only particular cases, as seen in §5. As such, the ambition of the work in this thesis is slightly different when compared to previous attempts, which are mainly concerned with presenting a categorical account of computability theory.

We showed in §6 that exposures are particularly elegant settings for reproducing well-known diagonal arguments. It has been known for some time that there are common elements between these theorems, but the language of P-categories and exposures allows us to capture what is needed for each argument in very fine detail. For example, Rice's theorem necessitates an evaluator $\epsilon : Q \overset{\bullet}{\looparrowright} \mathsf{Id}$, whereas Tarski's Undefinability Theorem precisely states that merely having evaluators is catastrophic for one aspect of consistency (fix-consistency), as this causes $\neg$ to have a fixed point. On the other hand, Gödel's First Incompleteness Theorem states that we *must* have more truth values than simply true and false.

Contrary to the presentation of Lawvere [1969, 2006], all our theorems are in the same language of comonadic exposures. Our development is *positive*, in that it is predicated on the existence of IFPs, and not the absence of EFPs, as in Lawvere's paper. We have concentrated on the *oughts*, and not the *ought nots*. It is for this reason that we view our results as a refinement of those of Lawvere.

We believe that all these observations and results lend support to the idea that exposures can become a useful toolkit situated at the heart of a new *theory of intensionality*, which would be applicable in all sorts of settings, not necessarily related to logic and computation. Furthermore, unlike previous work in the same style, exposures draw inspiration from modal logic and the Curry-Howard correspondence. It is for that reason that the resulting framework is—unlike all its predecessors—*inherently typed*. Essentially all previous work on the subject relied on some 'universal'—in one way or another object—which contained *codes* for a whole class of arrows.

First, we want to mention the only two generalisations of the SRT of which we are aware. They both seem very similar, and they concern *effective Scott domains*: one is due to Kanda [1988], and the other one due to Case and Moelius [2012].

As for category-theoretic frameworks, the only previous work with a similar flavour consists of (a) a paper of Mulry, which uses the *recursive topos*; and (b) the line of work that culminates with Cockett and Hofstra's *Turing categories*, as well as their intellectual predecessors.

Mulry's Recursive Topos

Mulry [1982] constructed the *recursive topos* **Rec**, which is the Grothendieck topos of canonical sheaves on the monoid of recursive functions under composition. Broadly speaking, the notion of (higher-order) computation in the recursive topos is that of *Banach-Mazur* computability: a map is computable if it maps recursive sequences to recursive sequences.[1]

In [Mulry, 1989], the author aims to "contemplate a synthetic theory of computation, i.e. an intrinsic set of categorical axioms for recursion which captures both essential features of classical recursion theory but is also applicable over a wide range of applications in areas of effective mathematics and computer science."

Indeed, Mulry's work has some overlap with some of the material we presented in §6.3 on Lawvere's theorem. He identifies an enumeration of the partial recursive functions as a point surjective map, and then proceeds to interpret all the classic results of type 2 computability (Myhill-Shepherdson, Kreisel-Lacombe-Shoenfield, etc.) in the context of the recursive topos. This is followed by a discussion of the connections between the recursive topos, the *effective topos* of Hyland [1982], and *effective Scott domains*. He there points out that one can construct a $\mathbb{N}$-path surjection $\mathbb{N} \to D$ and a point surjection $\mathbb{N} \to D^{\mathbb{N}}$ in **Rec**, for any effective Scott domain D.

Fixed points are discussed in the final section of the paper. It is noted there that Lawvere's theorem corresponds to (a limited version of) Kleene's FRT, simply by considering an enumeration of (two-argument) partial recursive functions, alongside $\mathbb{N} \cong \mathbb{N} \times \mathbb{N}$. This last isomorphism is also used to prove versions of the FRT and SRT for effective Scott domains, based on the observations of the previous paragraph. It is remarked that the same can be shown in the effective topos, but not in the category of effective domains itself, as neither of their natural numbers objects are effective domains.

[1] This short explanation of Banach-Mazur is due to Andrej Bauer, and was sourced from mathoverflow question #21745.

Thus, Mulry's versions of the FRT and SRT seem to depend on the natural numbers object $\mathbb{N}$. The SRT seems to give a fixed point for $h : \mathbb{N} \to \mathbb{N}$, in the sense that both the fixed point n and $h \circ n$ are indices for the same element of the effective Scott domain. This is indeed a version of the SRT for all effective Scott domains, but it lacks the well-typed flavour of our approach.

The road to Turing categories

Inspired by previous work by Paola and Heller [1987], and some of the material in the thesis of Birkedal [2000], Cockett and Hofstra [2008] developed the notion of a *Turing category*.

Turing categories are cartesian restriction categories: that is, they are equipped with some structure to handle the *partiality* of their arrows, and there are cartesian products—up to partiality. Their defining feature is that they have a *Turing object* A, which is a domain that contains codes for all the arrows of the category: it has a *universal application*,

$$\tau_{X,Y} : A \times X \to Y$$

for each pair of objects X and Y, such that given any $f : Z \times X \to Y$, an *index* $h : Z \to A$ exists, not by any means unique, such that the following diagram commutes:

$$\begin{array}{ccc} A \times X & \xrightarrow{\tau_{X,Y}} & Y \\ {\scriptstyle h \times id_X}\uparrow & \nearrow_{f} & \\ Z \times X & & \end{array}$$

So A is a very weak exponential for all X, Y at the same time. In fact, every object X of a Turing category is a retract of A.

Whereas the work of Cockett and Hofstra is a beautiful and general account of settings where basic recursion theory can be done, and also give rise to an interesting theory of *categorical simulations* [Cockett and Hofstra, 2010], they are very far from the goals of our own work: we are interested in exploring intensionality, which is only one of the phenomena that occur in recursion theory. We are also interested in doing so in a stringent type-theoretic manner, and thus we perceive the reliance on Turing objects as a problem. Cockett and Hofstra themselves point out the untyped nature of their work:

> "A similar inherent limitation to Turing categories lies in its essential untypedness. Recently, Longley [...] has advocated the use of typed PCAs in

> order to clarify notions of computation at higher types; it is not unimaginable that the corresponding generalization of Turing categories can be of interest if we wish to handle such notions of computation in our setting. Such a generalization would essentially bring us back to Birkedal's weakly closed partial cartesian categories."

This is followed by the observation that in the Turing category corresponding to classical computability theory no real discussion of higher-order phenomena can occur, as one has no way of speaking about type-2 functionals.

In conclusion, the work of Cockett and Hofstra is untyped, fundamentally based on partiality, and mostly centred around recursion theory. We prefer typed, total, and intensional settings, without wanting to declare any particular allegiance to recursion-theoretic arguments.

9.3.1 An Idea for an Alternative Approach

At this point we ought to record another candidate approach for modelling intensionality as we see it in this thesis. This idea is due to Martin Hyland, who examined this thesis. However, a similar framework has also been put forward by my fellow student, Mario Alvarez-Picallo, in his work on the semantics of metaprogramming.

In many ways, the framework of exposures can be seen as *evil* or *pathological*, in that it requires a highly non-standard property, namely that PERs are reflected instead of preserved—as they would be for a P-functor. In that sense, it may indeed be *non-categorical* (this is another interesting idea that needs to be investigated).

Perhaps the following proposal would be more workable. Instead of insisting on a *single* mathematical universe, i.e. a single category, we could consider two: one that is *intensional*, and one that is *extensional*. The theory of exposures already gives us two rather good candidates for these: given an endoexposure $Q : (\mathfrak{C}, \sim) \nrightarrow (\mathfrak{C}, \sim)$, the intensional universe is the x-ray P-category $(\mathfrak{C}, \approx_Q)$, wheras the extensional one is the P-category $(\mathfrak{C}, \sim)$ itself—see §4.2 for definitions of these notions. Now, as intensional equality implies extensional equality, it is evident that there is a trivial P-functor,

$$(\mathfrak{C}, \approx_Q) \longrightarrow (\mathfrak{C}, \sim)$$

which is the identity on objects, the identity on morphisms, and full—but definitely not faithful![2] One could perhaps argue that we could even move away from P-categories, and back into ordinary categories: there could be a functor $\mathcal{C} \to \mathcal{D}$, which

[2]The first of these criteria is a recurrent theme of questionable categorical status, which nonetheless occurs reasonably often, e.g. in Freyd categories; see e.g. Staton [2014].

is the identity on objects and full. Perhaps $\mathcal{D}$ could be of the form $\mathcal{C}/\sim$, i.e. a quotient category of some sort. Thus, intensionality is obtained by having (P-)categories on *two levels*, so that one is included in the other.

Mario Alvarez has proposed some directions regarding the categorical modelling of non-homogeneous staged metaprogramming calculi, like the ones of Davies [1995, 1996, 2017], which bear a certain resemblance to the preceding idea. Put simply, he proposes a simple metaprogramming calculus which consists of essentially two 'stages,' one 'under the circle' (cf. under the box), and the ordinary one. Under the circle, affairs intensional, in much the same way that they are under our boxes, but otherwise things work up to ordinary equality. Using the syntax of this calculus, one can construct a *cartesian* category $\mathbb{SM}$, the *syntactic model*: this is a term model, but not quotiented up to equality. A model of this is then a cartesian closed category $\mathcal{C}$ along with a functor,

$$F : \mathbb{SM} \longrightarrow \mathcal{C}$$

which is *strictly product-preserving*. Furthermore, if *intensional operations* are to be soundly interpreted, this functor should be *faithful*. This is to be understood in the following way: the functor F *encodes* syntactic information *within the semantic model*, in a manner that does not collapse the syntax.

There is an odd tension between fullness and faithfulness in this. Which one should we choose for intensionality? Faithfulness guarantees that the syntax is accurately represented in the model, whereas fullness describes the idea that for every 'extensional' view, there is at least one intensional. Nevertheless, investigation of this idea of *two level-two category* paradigm of intensionality is likely to be a very interesting avenue for further research.

9.4 Kleene's mysterious Second Recursion Theorem

The basic impetus behind this thesis was not intensionality itself, but rather its rôle in Kleene's Second Recursion Theorem, which we exhaustively studied in §2. These questions were brought to the author's attention by Abramsky [2012, 2014], who—along with Jones [2013]—had identified the mysterious nature of this theorem in the 1980s. Abramsky [2012] also remarks that the theorem is very powerful, and—even though very simple—its proof is opaque, and provides no intuition. Our analysis of intensional recursion in §6, and the connection we have drawn with Lawvere's work on diagonal arguments, has shed some light on these issues.

First, we believe that the opacity in proof is not a real problem: all diagonalisation proofs have a quintessentially 'magical' element that often stimulates people's imagination, bringing them to the forefront of popular science books and scientific novels, see e.g. Hofstadter [1979], Doxiadis [2001]. What is more, Kleene [1981] explicitly states that the SRT was obtained quite straightforwardly by translating the FRT from the λ-calculus. He even dates this derivation before the 1st of July 1935.

Instead, we believe that the our analysis provides a new perspective on the *structure* of the situations in which Kleene's argument is fundamentally applicable. This amounts to a generalisation of intensional recursion beyond first-order computability: the Curry-Howard interpretation through Löb's rule, as well as the Intensional Recursion theorem stated in the language of comonadic exposures, reveal patterns in the statement and the proof of the theorem that were not known before.

9.5 Concluding remarks

It is evident from the previous sections that there are many very interesting questions that follow from our work in this thesis. Perhaps the most interesting theoretical direction is that of studying the expressiveness of iPCF, especially through the related themes of metaprogramming and non-functional higher-order computation. The present author believes that it is very likely that we could use modalities and intensionality to obtain access to computational power that we have both sclerotically rejected from theoretical analysis, or left to programmers of the untyped persuasion.

These developments—we hope—will proceed hand-in-hand with an improved understanding of metaprogramming, intensional or not. Progress is difficult to achieve in metaprogramming not only because we have been lacking a theoretical foundation, but also because industrial metaprogramming has progressed unabashedly in the meantime. The result is that multiple hard-to-shake ad-hoc habits have been established.

On the other hand, our intensional framework is as close to 1-category theory as one can get, and it is for this reason that we hope that it may be more widely applicable than just in the context of computability and logic. We have demonstrated at least one example of this in the case of homological algebra (§5.3). However, as the applications of category theory expand to include linguistics and philosophy—see e.g. the forthcoming volume [Landry, 2017]—we would like to believe that the concept of intensionality will reappear in many different settings, and our framework will be there to explain its structure.

Appendix A

iPCF v2.0 in Agda

A.1 Basics.agda

```
module Basics where

open import Relation.Binary.PropositionalEquality
open import Data.Nat using (ℕ ; zero ; suc)
open import Data.Sum renaming (_⊎_ to _+_)

infixr 1 _=>_
infixr 5 _∈_
infixl 1 _⊆_
infixl 4 _,_
infixl 3 _++_
infixl 2 _∧_

------------
- Types and Contexts -
------------

data Types : Set where
  simple : Types
  modal : Types

data Ty : Types → Set where
  P : ∀ {T} → ℕ → Ty T
  _=>_ : ∀ {T} → Ty T → Ty T → Ty T
```

```
  _∧_ : ∀ {T} → Ty T → Ty T → Ty T
  □_ :  Ty modal → Ty modal

data Cx : Types → Set where
  · : ∀ {T} → Cx T
  _,_ : ∀ {T : Types} (Γ : Cx T) (A : Ty T) → Cx T

data _∈_ : ∀ {T : Types} (A : Ty T) (Γ : Cx T) → Set where
  top : ∀ {T} {Γ : Cx T} {A : Ty T} → A ∈ (Γ , A)
  pop : ∀ {T} {Γ : Cx T} {A B : Ty T} (i : A ∈ Γ) → A ∈ (Γ , B)

_⊆_ : ∀ {T} (Γ Δ : Cx T) → Set
Γ ⊆ Δ = ∀ {A} → A ∈ Γ → A ∈ Δ

- Functions on contexts.

boxcx : Cx modal → Cx modal
boxcx · = ·
boxcx (Γ , A) = boxcx Γ , □ A

_++_ : ∀ {T} → Cx T → Cx T → Cx T
Δ ++ · = Δ
Δ ++ (Γ , A) = (Δ ++ Γ) , A

box∈cx : ∀ {Γ : Cx modal} {A : Ty modal} → A ∈ Γ → □ A ∈ boxcx Γ
box∈cx top = top
box∈cx (pop d) = pop (box∈cx d)

subsetdef :  ∀ {T} {Γ Δ : Cx T} {A} → A ∈ Γ → Γ ⊆ Δ → A ∈ Δ
subsetdef d f = f d

subsetempty : ∀ {T} {Γ : Cx T} → · ⊆ Γ
subsetempty ()

subsetid : ∀ {T} {Γ : Cx T} → Γ ⊆ Γ
subsetid = λ {Γ} {A} z → z
```

```
weakone : ∀ {T} {Γ Δ : Cx T} {A} → Γ ⊆ Δ → Γ ⊆ (Δ , A)
weakone p = λ {A} z → pop (p z)

weakboth : ∀ {T} {Γ Δ : Cx T} {A} → Γ ⊆ Δ → Γ , A ⊆ Δ , A
weakboth p top = top
weakboth p (pop x) = subsetdef x (weakone p)

weakmany : ∀ {T} (Γ Δ : Cx T) → Γ ⊆ Γ ++ Δ
weakmany Γ · x = x
weakmany Γ (Δ , A) x = pop (weakmany Γ Δ x)

concat-subset-1 : ∀ {T} (Γ Δ : Cx T) → Γ ⊆ Γ ++ Δ
concat-subset-1 Γ · x = x
concat-subset-1 Γ (Δ , A) x = subsetdef x (weakone (concat-subset-1 Γ Δ))

concat-subset-2 : ∀ {T} (Γ Δ : Cx T) → Δ ⊆ Γ ++ Δ
concat-subset-2 Γ · ()
concat-subset-2 Γ (Δ , A) x = subsetdef x (weakboth (concat-subset-2 Γ Δ))

incl-trans : ∀ {T} {Γ Γ' Γ'' : Cx T} → Γ ⊆ Γ' → Γ' ⊆ Γ'' → Γ ⊆ Γ''
incl-trans p q x = q (p x)

swap-last : ∀ {T} {Γ : Cx T} {A B} → Γ , A , B ⊆ Γ , B , A
swap-last {_} {·} top = pop top
swap-last {_} {·} (pop top) = top
swap-last {_} {·} (pop (pop x)) = pop (pop x)
swap-last {_} {Γ , A} top = pop top
swap-last {_} {Γ , A} (pop top) = top
swap-last {_} {Γ , A} (pop (pop x)) = pop (pop x)

cx-exch : ∀ {T} {Γ Δ : Cx T} {A B} → (Γ , A , B) ++ Δ ⊆ (Γ , B , A) ++ Δ
cx-exch {Δ = ·} d = swap-last d
cx-exch {Δ = Δ , A1} top = top
cx-exch {Δ = Δ , A1} (pop d) = subsetdef d (weakone (cx-exch {Δ = Δ}))
```

```
cx-contr : ∀ {T} {Γ Δ : Cx T} {A} → (Γ , A , A) ++ Δ ⊆ (Γ , A) ++ Δ
cx-contr {Δ = ·} top = top
cx-contr {Δ = ·} (pop d) = d
cx-contr {Δ = Δ , A1} top = top
cx-contr {Δ = Δ , A1} (pop d) = subsetdef d (weakone (cx-contr {Δ = Δ}))

is-in : ∀ {T} (Γ Γ' : Cx T) (A : Ty T) → A ∈ (Γ , A ++ Γ')
is-in Γ · A = top
is-in Γ (Γ' , A') A = pop (is-in Γ Γ' A)

ctxt-disj : ∀ {T} (Γ Γ' : Cx T) (A : Ty T) → A ∈ (Γ ++ Γ') → A ∈ Γ + A ∈ Γ'
ctxt-disj Γ · A x = inj1 x
ctxt-disj Γ (Γ' , A') .A' top = inj₂ top
ctxt-disj Γ (Γ' , A') A (pop x)
  with ctxt-disj Γ Γ' A x
ctxt-disj Γ (Γ' , A') A (pop x) | inj1 z = inj1 z
ctxt-disj Γ (Γ' , A') A (pop x) | inj₂ z = inj₂ (pop z)

swap-out : ∀ {T} (Δ Γ : Cx T) (A : Ty T) → (Δ , A) ++ Γ ⊆ (Δ ++ Γ) , A
swap-out Δ · A x = x
swap-out Δ (Γ , B) A x = swap-last (subsetdef x (weakboth (swap-out Δ Γ A)))

swap-in : ∀ {T} (Δ Γ : Cx T) (A : Ty T) → (Δ ++ Γ) , A ⊆ (Δ , A) ++ Γ
swap-in Δ Γ A top = is-in Δ Γ A
swap-in Δ Γ A (pop x)
  with ctxt-disj Δ Γ _ x
swap-in Δ Γ A (pop x) | inj1 y = concat-subset-1 (Δ , A) Γ (pop y)
swap-in Δ Γ A (pop x) | inj₂ y = concat-subset-2 (Δ , A) Γ y
```

A.2 iPCF.agda

```
module iPCF where

infixl 0 _/_⊢_

open import Basics

- Definition

data _/_⊢_ (Δ Γ : Cx modal) : Ty modal → Set where

  iPCF-var : ∀ {A}

    → A ∈ Γ
    -------
    → Δ / Γ ⊢ A

  iPCF-modal-var : ∀ {A}

    → A ∈ Δ
    ------
    → Δ / Γ ⊢ A

  iPCF-app : ∀ {A B}

    → Δ / Γ ⊢ A => B  → Δ / Γ ⊢ A
    -----------------
      → Δ / Γ ⊢ B

  iPCF-lam : ∀ {A B}

    → Δ / (Γ , A) ⊢ B
    ---------
    → Δ / Γ ⊢ A => B
```

iPCF-prod : $\forall\ \{A\ B\}$

$$\frac{\to \Delta\ /\ \Gamma \vdash A \quad \to \Delta\ /\ \Gamma \vdash B}{\to \Delta\ /\ \Gamma \vdash A \wedge B}$$

iPCF-fst : $\forall\ \{A\ B\}$

$$\frac{\to \Delta\ /\ \Gamma \vdash A \wedge B}{\to \Delta\ /\ \Gamma \vdash A}$$

iPCF-snd : $\forall\ \{A\ B\}$

$$\frac{\to \Delta\ /\ \Gamma \vdash A \wedge B}{\to \Delta\ /\ \Gamma \vdash B}$$

iPCF-boxI : $\forall\ \{A\}$

$$\frac{\to \Delta\ /\ \cdot \vdash A}{\to \Delta\ /\ \Gamma \vdash \Box\ A}$$

iPCF-boxE : $\forall\ \{A\ C\}$

$$\frac{\to \Delta\ /\ \Gamma \vdash \Box\ A \quad \to (\Delta\ ,\ A)\ /\ \Gamma \vdash C}{\to \Delta\ /\ \Gamma \vdash C}$$

iPCF-fix : $\forall\ \{A\}$

$$\frac{\to \Delta\ /\ (\cdot\ ,\ \Box\ A) \vdash A}{\to \Delta\ /\ \Gamma \vdash A}$$

```
- Weakening and exchange.


exch : ∀ {Δ Γ A B C} (Γ' : Cx modal)

      → Δ / (Γ , A , B) ++ Γ' ⊢ C
      ---------------
      → Δ / (Γ , B , A) ++ Γ' ⊢ C

exch Γ' (iPCF-var x) = iPCF-var (cx-exch {Δ = Γ'} x)
exch Γ' (iPCF-modal-var x) = iPCF-modal-var x
exch Γ' (iPCF-app d d1) = iPCF-app (exch Γ' d) (exch Γ' d1)
exch {C = A => B} Γ' (iPCF-lam d) = iPCF-lam (exch (Γ' , A) d)
exch Γ' (iPCF-prod d e) = iPCF-prod (exch Γ' d) (exch Γ' e)
exch Γ' (iPCF-fst d) = iPCF-fst (exch Γ' d)
exch Γ' (iPCF-snd d) = iPCF-snd (exch Γ' d)
exch Γ' (iPCF-boxI d) = iPCF-boxI d
exch Γ' (iPCF-boxE d e) = iPCF-boxE (exch Γ' d) (exch Γ' e)
exch Γ' (iPCF-fix d) = iPCF-fix d


exch-modal : ∀ {Δ Γ A B C} (Δ' : Cx modal)

      → (Δ , A , B) ++ Δ' / Γ ⊢ C
      ---------------
      → (Δ , B , A) ++ Δ' / Γ ⊢ C

exch-modal Δ' (iPCF-var x) = iPCF-var x
exch-modal Δ' (iPCF-modal-var x) =
  iPCF-modal-var (subsetdef x (cx-exch {Δ = Δ'}))
exch-modal Δ' (iPCF-app d e) =
  iPCF-app (exch-modal Δ' d) (exch-modal Δ' e)
exch-modal Δ' (iPCF-lam d) = iPCF-lam (exch-modal Δ' d)
exch-modal Δ' (iPCF-prod d e) =
  iPCF-prod (exch-modal Δ' d) (exch-modal Δ' e)
```

```
exch-modal Δ' (iPCF-fst d) = iPCF-fst (exch-modal Δ' d)
exch-modal Δ' (iPCF-snd d) = iPCF-snd (exch-modal Δ' d)
exch-modal Δ' (iPCF-boxI d) = iPCF-boxI (exch-modal Δ' d)
exch-modal Δ' (iPCF-boxE d e) =
  iPCF-boxE (exch-modal Δ' d) (exch-modal (Δ' , _) e)
exch-modal Δ' (iPCF-fix d) = iPCF-fix (exch-modal Δ' d)


weak : ∀ {Δ Γ Γ' A}

     → Δ / Γ ⊢ A  → Γ ⊆ Γ'
     -------------
       → (Δ / Γ' ⊢ A)

weak (iPCF-var x) f = iPCF-var (f x)
weak (iPCF-modal-var x) f = iPCF-modal-var x
weak (iPCF-app d e) f = iPCF-app (weak d f) (weak e f)
weak (iPCF-lam d) f = iPCF-lam (weak d (weakboth f))
weak (iPCF-prod d e) f = iPCF-prod (weak d f) (weak e f)
weak (iPCF-fst d) f = iPCF-fst (weak d f)
weak (iPCF-snd d) f = iPCF-snd (weak d f)
weak (iPCF-boxI d) f = iPCF-boxI d
weak (iPCF-boxE d e) f =
  iPCF-boxE (weak d f) (weak e f)
weak (iPCF-fix d) f = iPCF-fix d


weak-modal : ∀ {Δ Δ' Γ A}

     → Δ / Γ ⊢ A  → Δ ⊆ Δ'
     -------------
       → Δ' / Γ ⊢ A

weak-modal (iPCF-var p) x = iPCF-var p
weak-modal (iPCF-modal-var p) x = iPCF-modal-var (x p)
weak-modal (iPCF-app t u) x = iPCF-app (weak-modal t x)
```

```
            (weak-modal u x)
weak-modal (iPCF-lam t) x = iPCF-lam (weak-modal t x)
weak-modal (iPCF-prod t u) x = iPCF-prod (weak-modal t x)
              (weak-modal u x)
weak-modal (iPCF-fst t) x = iPCF-fst (weak-modal t x)
weak-modal (iPCF-snd t) x = iPCF-snd (weak-modal t x)
weak-modal (iPCF-boxI t) x = iPCF-boxI (weak-modal t x)
weak-modal (iPCF-boxE t u) x =
  iPCF-boxE (weak-modal t x)
    (weak-modal u (weakboth x))
weak-modal (iPCF-fix t) x = iPCF-fix (weak-modal t x)


- Cut.

cut : ∀ {Δ Γ A B} → (Γ' : Cx modal)

      → Δ / Γ ⊢ A  → Δ / Γ , A ++ Γ' ⊢ B
      --------------------
        → Δ / Γ ++ Γ' ⊢ B

cut · d (iPCF-var top) = d
cut · d (iPCF-var (pop x)) = iPCF-var x
cut (Γ' , B) d (iPCF-var top) = iPCF-var top
cut (Γ' , A') d (iPCF-var (pop x)) =
  weak (cut Γ' d (iPCF-var x)) (weakone subsetid)
cut Γ' d (iPCF-modal-var p) = iPCF-modal-var p
cut Γ' d (iPCF-app t u) = iPCF-app (cut Γ' d t) (cut Γ' d u)
cut Γ' d (iPCF-lam e) = iPCF-lam (cut (Γ' , _) d e)
cut Γ' d (iPCF-prod t u) = iPCF-prod (cut Γ' d t) (cut Γ' d u)
cut Γ' d (iPCF-fst e) = iPCF-fst (cut Γ' d e)
cut Γ' d (iPCF-snd e) = iPCF-snd (cut Γ' d e)
cut Γ' d (iPCF-boxI e) = iPCF-boxI e
cut Γ' d (iPCF-boxE t u) =
  iPCF-boxE (cut Γ' d t)
    (cut Γ' (weak-modal d (weakone (subsetid))) u)
```

```
cut Γ' d (iPCF-fix t) = iPCF-fix t


cut-modal : ∀ {Δ Γ A B} → (Δ' : Cx modal)

      → Δ / · ⊢ A   → Δ , A ++ Δ' / Γ ⊢ B
      ------------------
        → Δ ++ Δ' / Γ ⊢ B

cut-modal Δ' d (iPCF-var x) = iPCF-var x
cut-modal · d (iPCF-modal-var top) = weak d subsetempty
cut-modal · d (iPCF-modal-var (pop x)) = iPCF-modal-var x
cut-modal (Δ' , B) d (iPCF-modal-var top) = iPCF-modal-var top
cut-modal (Δ' , A') d (iPCF-modal-var (pop x)) =
  weak-modal (cut-modal Δ' d (iPCF-modal-var x)) (weakone subsetid)
cut-modal Δ' d (iPCF-app p q) =
  iPCF-app (cut-modal Δ' d p) (cut-modal Δ' d q)
cut-modal Δ' d (iPCF-lam e) = iPCF-lam (cut-modal Δ' d e)
cut-modal Δ' d (iPCF-prod p q) =
  iPCF-prod (cut-modal Δ' d p) (cut-modal Δ' d q)
cut-modal Δ' d (iPCF-fst e) = iPCF-fst (cut-modal Δ' d e)
cut-modal Δ' d (iPCF-snd e) = iPCF-snd (cut-modal Δ' d e)
cut-modal Δ' d (iPCF-boxI e) = iPCF-boxI (cut-modal Δ' d e)
cut-modal Δ' d (iPCF-boxE p q) =
  iPCF-boxE (cut-modal Δ' d p) (cut-modal (Δ' , _) d q)
cut-modal Δ' d (iPCF-fix e) = iPCF-fix (cut-modal Δ' d e)
```

A.3 iPCF2.agda

```
module iPCF2 where

infixl 0 _/_⊢_::_

open import Basics

- Definition

data Judgement : Set where
  int : Judgement
  ext : Judgement

data _/_⊢_::_ (Δ Γ : Cx modal) : Judgement → Ty modal → Set where

  iPCF-var : ∀ {J A}

      → A ∈ Γ
    --------------
    → Δ / Γ ⊢ J :: A

  iPCF-modal-var : ∀ {J A}

      → A ∈ Δ
    --------------
    → Δ / Γ ⊢ J :: A

  iPCF-app : ∀ {J A B}

    → Δ / Γ ⊢ J :: A => B    → Δ / Γ ⊢ J :: A
    ----------------------------
      → Δ / Γ ⊢ J :: B

  iPCF-lam-ext : ∀ {A B}
```

```
      → Δ / (Γ , A) ⊢ ext :: B
      ------------------------
      → Δ / Γ ⊢ ext :: A => B

    iPCF-lam-int : ∀ {J A B}

      → · / (· , A) ⊢ J :: B
      ----------------------
      → Δ / Γ ⊢ int :: A => B

    iPCF-boxI : ∀ {J A}

      → Δ / · ⊢ int :: A
      ------------------
      → Δ / Γ ⊢ J :: □ A

    iPCF-boxE : ∀ {J A C}

      → Δ / Γ ⊢ J :: □ A  → (Δ , A) / Γ ⊢ J :: C
      ----------------------------------------
        → Δ / Γ ⊢ J :: C

    iPCF-fix : ∀ {J A}

      → · / (· , □ A) ⊢ int :: A
      --------------------------
      → Δ / Γ ⊢ J :: A


- Weakening and exchange.


exch : ∀ {Δ Γ J A B C} (Γ' : Cx modal)

      → Δ / (Γ , A , B) ++ Γ' ⊢ J :: C
      --------------------------------
```

```
      → Δ / (Γ , B , A) ++ Γ' ⊢ J :: C

exch Γ' (iPCF-var x) = iPCF-var (cx-exch {Δ = Γ'} x)
exch Γ' (iPCF-modal-var x) = iPCF-modal-var x
exch Γ' (iPCF-app d d1) = iPCF-app (exch Γ' d) (exch Γ' d1)
exch {C = A => B} Γ' (iPCF-lam-ext d) = iPCF-lam-ext (exch (Γ' , A) d)
exch Γ' (iPCF-lam-int d) = iPCF-lam-int d
exch Γ' (iPCF-boxI d) = iPCF-boxI d
exch Γ' (iPCF-boxE d e) = iPCF-boxE (exch Γ' d) (exch Γ' e)
exch Γ' (iPCF-fix f) = iPCF-fix f


exch-modal : ∀ {Δ Γ J A B C} (Δ' : Cx modal)

      → (Δ , A , B) ++ Δ' / Γ  ⊢ J :: C
      ---------------
      → (Δ , B , A) ++ Δ' / Γ ⊢ J :: C

exch-modal Δ' (iPCF-var x) = iPCF-var x
exch-modal Δ' (iPCF-modal-var x) =
   iPCF-modal-var (subsetdef x (cx-exch {Δ = Δ'}))
exch-modal Δ' (iPCF-app d e) =
   iPCF-app (exch-modal Δ' d) (exch-modal Δ' e)
exch-modal Δ' (iPCF-lam-ext d) = iPCF-lam-ext (exch-modal Δ' d)
exch-modal Δ' (iPCF-lam-int d) = iPCF-lam-int d
exch-modal Δ' (iPCF-boxI d) = iPCF-boxI (exch-modal Δ' d)
exch-modal Δ' (iPCF-boxE d e) =
   iPCF-boxE (exch-modal Δ' d) (exch-modal (Δ' , _) e)
exch-modal Δ' (iPCF-fix f) = iPCF-fix f


weak : ∀ {Δ Γ Γ' J A}

      → Δ / Γ ⊢ J ::  A  → Γ ⊆ Γ'
      ----------------
        → Δ / Γ' ⊢ J :: A
```

```
weak (iPCF-var x) f = iPCF-var (f x)
weak (iPCF-modal-var x) f = iPCF-modal-var x
weak (iPCF-app d e) f = iPCF-app (weak d f) (weak e f)
weak (iPCF-lam-int d) f = iPCF-lam-int d
weak (iPCF-lam-ext d) f = iPCF-lam-ext (weak d (weakboth f))
weak (iPCF-boxI d) f = iPCF-boxI d
weak (iPCF-boxE d e) f =
  iPCF-boxE (weak d f) (weak e f)
weak (iPCF-fix d) f = iPCF-fix d


weak-modal : ∀ {Δ Δ' Γ J A}

     → Δ / Γ ⊢ J :: A  → Δ ⊆ Δ'
     -------------
       → Δ' / Γ ⊢ J :: A

weak-modal (iPCF-var p) x = iPCF-var p
weak-modal (iPCF-modal-var p) x = iPCF-modal-var (x p)
weak-modal (iPCF-app t u) x = iPCF-app (weak-modal t x)
          (weak-modal u x)
weak-modal (iPCF-lam-int t) x = iPCF-lam-int t
weak-modal (iPCF-lam-ext t) x = iPCF-lam-ext (weak-modal t x)
weak-modal (iPCF-boxI t) x = iPCF-boxI (weak-modal t x)
weak-modal (iPCF-boxE t u) x =
  iPCF-boxE (weak-modal t x)
    (weak-modal u (weakboth x))
weak-modal (iPCF-fix f) x = iPCF-fix f


- Including intensional into extensional.

incl : ∀ {Δ Γ A}

     → Δ / Γ ⊢ int :: A
```

```
      ----------
      → Δ / Γ ⊢ ext :: A

incl (iPCF-var x) = iPCF-var x
incl (iPCF-modal-var x) = iPCF-modal-var x
incl (iPCF-app d e) = iPCF-app (incl d) (incl e)
incl (iPCF-lam-int {int} d) =
  iPCF-lam-ext (weak (weak-modal (incl d) subsetempty) (weakboth subsetempty))
incl (iPCF-lam-int {ext} d) =
  iPCF-lam-ext (weak (weak-modal d subsetempty) (weakboth subsetempty))
incl (iPCF-boxI d) = iPCF-boxI d
incl (iPCF-boxE d e) = iPCF-boxE (incl d) (incl e)
incl (iPCF-fix f) = iPCF-fix f


incl-either-ext : ∀ {Δ J Γ A}

      → Δ / Γ ⊢ J :: A
      ----------
      → Δ / Γ ⊢ ext :: A

incl-either-ext {J = int} d = incl d
incl-either-ext {J = ext} d = d


incl-either-int : ∀ {Δ J Γ A}

      → Δ / Γ ⊢ int :: A
      ----------
      → Δ / Γ ⊢ J :: A

incl-either-int {J = int} d = d
incl-either-int {J = ext} d = incl d

- Cut.
```

```
cut-ext : ∀ {Δ Γ J A B} → (Γ' : Cx modal)

      → Δ / Γ ⊢ ext :: A   → Δ / Γ , A ++ Γ' ⊢ J :: B
      ----------------------
        → Δ / Γ ++ Γ' ⊢ ext :: B

cut-ext · d (iPCF-var top) = d
cut-ext · d (iPCF-var (pop x)) = iPCF-var x
cut-ext (Γ' , B) d (iPCF-var top) = iPCF-var top
cut-ext (Γ' , A') d (iPCF-var (pop x)) =
  weak (cut-ext {J = ext} Γ' d (iPCF-var x)) (weakone subsetid)
cut-ext Γ' d (iPCF-modal-var p) = iPCF-modal-var p
cut-ext Γ' d (iPCF-app t u) = iPCF-app (cut-ext Γ' d t) (cut-ext Γ' d u)
cut-ext Γ' d (iPCF-lam-int e) = incl (iPCF-lam-int e)
cut-ext Γ' d (iPCF-lam-ext e) = iPCF-lam-ext (cut-ext (Γ' , _) d e)
cut-ext Γ' d (iPCF-boxI e) = iPCF-boxI e
cut-ext Γ' d (iPCF-boxE t u) =
  iPCF-boxE (cut-ext Γ' d t)
    (cut-ext Γ' (weak-modal d (weakone (subsetid))) u)
cut-ext Γ' d (iPCF-fix f) = iPCF-fix f


cut-int : ∀ {Δ Γ J A B} → (Γ' : Cx modal)

      → Δ / Γ ⊢ int :: A   → Δ / Γ , A ++ Γ' ⊢ J :: B
      ----------------------
        → Δ / Γ ++ Γ' ⊢ J :: B

cut-int · d (iPCF-var top) = incl-either-int d
cut-int · d (iPCF-var (pop x)) = iPCF-var x
cut-int (Γ' , B) d (iPCF-var top) = iPCF-var top
cut-int (Γ' , A') d (iPCF-var (pop x)) =
  weak (cut-int Γ' d (iPCF-var x)) (weakone subsetid)
cut-int Γ' d (iPCF-modal-var p) = iPCF-modal-var p
cut-int Γ' d (iPCF-app t u) = iPCF-app (cut-int Γ' d t) (cut-int Γ' d u)
```

```
cut-int Γ' d (iPCF-lam-int e) = iPCF-lam-int e
cut-int Γ' d (iPCF-lam-ext e) = iPCF-lam-ext (cut-ext (Γ' , _) (incl d) e)
cut-int Γ' d (iPCF-boxI e) = iPCF-boxI e
cut-int Γ' d (iPCF-boxE t u) =
  iPCF-boxE (cut-int Γ' d t)
    (cut-int Γ' (weak-modal d (weakone (subsetid))) u)
cut-int Γ' d (iPCF-fix e) = iPCF-fix e


cut-modal : ∀ {Δ Γ J A B} → (Δ' : Cx modal)

      → Δ / · ⊢ int :: A    → Δ , A ++ Δ' / Γ ⊢ J :: B
      ------------------------
        → Δ ++ Δ' / Γ ⊢ J :: B

cut-modal Δ' d (iPCF-var x) = iPCF-var x
cut-modal · d (iPCF-modal-var top) = incl-either-int (weak d subsetempty)
cut-modal · d (iPCF-modal-var (pop x)) = iPCF-modal-var x
cut-modal (Δ' , B) d (iPCF-modal-var top) = iPCF-modal-var top
cut-modal (Δ' , A') d (iPCF-modal-var (pop x)) =
  weak-modal (cut-modal Δ' d (iPCF-modal-var x)) (weakone subsetid)
cut-modal Δ' d (iPCF-app p q) =
  iPCF-app (cut-modal Δ' d p) (cut-modal Δ' d q)
cut-modal Δ' d (iPCF-lam-ext e) = iPCF-lam-ext (cut-modal Δ' d e)
cut-modal Δ' d (iPCF-lam-int e) = iPCF-lam-int e
cut-modal Δ' d (iPCF-boxI e) = iPCF-boxI (cut-modal Δ' d e)
cut-modal Δ' d (iPCF-boxE p q) =
  iPCF-boxE (cut-modal Δ' d p) (cut-modal (Δ' , _) d q)
cut-modal Δ' d (iPCF-fix f) = iPCF-fix f
```

Bibliography

Samson Abramsky. The Lazy Lambda Calculus. In *Research Topics in Functional Programming*, pages 65–117. Addison Wesley, 1990. URL https://www.cs.ox.ac.uk/files/293/lazy.pdf.

Samson Abramsky. What are the Fundamental Structures of Concurrency? *Electronic Notes in Theoretical Computer Science*, 162:37–41, 2006. ISSN 15710661. doi: 10.1016/j.entcs.2005.12.075. URL http://linkinghub.elsevier.com/retrieve/pii/S1571066106004105.

Samson Abramsky. Notes on Intensional Recursion, 2012.

Samson Abramsky. Intensionality, Definability and Computation. In Alexandru Baltag and Sonja Smets, editors, *Johan van Benthem on Logic and Information Dynamics*, pages 121–142. Springer-Verlag, 2014. doi: 10.1007/978-3-319-06025-5_5. URL https://dx.doi.org/10.1007/978-3-319-06025-5_5.

Samson Abramsky and Achim Jung. Domain Theory. *Handbook of Logic in Computer Science*, 3:1–168, 1994. URL https://www.cs.bham.ac.uk/$\sim$axj/pub/papers/handy1.pdf.

Samson Abramsky and Guy McCusker. Linearity, Sharing and State: a fully abstract game semantics for Idealized Algol with active expressions. *Electronic Notes in Theoretical Computer Science*, 3:2–14, 1996. ISSN 15710661. doi: 10.1016/S1571-0661(05)80398-6. URL http://linkinghub.elsevier.com/retrieve/pii/S1571066105803986.

Samson Abramsky and Nikos Tzevelekos. Introduction to Categories and Categorical Logic. In Bob Coecke, editor, *New Structures for Physics*, pages 3–94. Springer-Verlag, 2011. doi: 10.1007/978-3-642-12821-9_1. URL http://arxiv.org/abs/1102.1313.

Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full Abstraction for PCF. *Information and Computation*, 163:409–470, 1996.

Samson Abramsky, Kohei Honda, and G McCusker. A fully abstract game semantics for general references. In *Proceedings of theThirteenth Annual IEEE Symposium on Logic in Computer Science*. IEEE Comput. Soc, 1998. ISBN 0-8186-8506-9. doi: 10.1109/LICS.1998.705669. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=705669.

Leonard M. Adleman. An Abstract Theory of Computer Viruses. In *Advances in Cryptology - CRYPTO' 88*, volume 403 of *Lecture Notes in Computer Science*, pages 354–374. Springer New York, New York, NY, 1990. ISBN 3-540-97196-3. doi: 10.1007/0-387-34799-2_28. URL https://dx.doi.org/10.1007/0-387-34799-2_28.

Thorsten Altenkirch, Nils Anders Danielsson, and Nicolai Kraus. Partiality, Revisited: The Partiality Monad as a Quotient Inductive-Inductive Type. In *Proceedings of the 20th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, oct 2017. URL http://arxiv.org/abs/1610.09254.

Steve Awodey. *Category Theory*. Oxford Logic Guides. Oxford University Press, 2010. ISBN 9780191612558. URL https://books.google.co.uk/books?id=zLs8BAAAQBAJ.

Andrew Graham Barber. Dual Intuitionistic Linear Logic. Technical report, ECS-LFCS-96-347, Laboratory for Foundations of Computer Science, University of Edinburgh, 1996. URL http://www.lfcs.inf.ed.ac.uk/reports/96/ECS-LFCS-96-347/.

Henk Barendregt. *Lambda Calculus: Its Syntax and Semantics*. North-Holland, Amsterdam, 1984. ISBN 978-0444875082.

Henk Barendregt. Self-Interpretation in Lambda Calculus. *Journal of Functional Programming*, 1(2):229–233, 1991. URL https://dx.doi.org/10.1017/S0956796800020062.

Andrej Bauer. Definability and extensionality of the modulus of continuity functional. *Mathematics and Computation (blog)*, 2011. URL http://math.andrej.com/2011/07/27/definability-and-extensionality-of-the-modulus-of-continuity-functional/.

Alan Bawden. Quasiquotation in LISP. In *Proceedings of the 6th ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation (PEPM '99)*, 1999. URL http://repository.readscheme.org/ftp/papers/pepm99/bawden.pdf.

Michael J. Beeson. *Foundations of Constructive Mathematics*. Springer Berlin Heidelberg, 1985. ISBN 978-3-642-68954-3. doi: 10.1007/978-3-642-68952-9. URL https://dx.doi.org/10.1007/978-3-642-68952-9.

Martin Berger. Foundations of meta-programming, 2016. URL http://users.sussex.ac.uk/$\sim$mfb21/publications/mp-slides/slides.pdf.

Gavin M. Bierman. Program equivalence in a linear functional language. *Journal of Functional Programming*, 10(2):167–190, 2000. ISSN 09567968. doi: 10.1017/S0956796899003639.

Gavin M. Bierman and Valeria de Paiva. On an Intuitionistic Modal Logic. *Studia Logica*, 65(3):383–416, 2000. doi: 10.1023/A:1005291931660. URL https://dx.doi.org/10.1023/A:1005291931660.

Lars Birkedal. Developing theories of types and computability via realizability. *Electronic Notes in Theoretical Computer Science*, 34:2, 2000. ISSN 15710661. doi: 10.1016/S1571-0661(05)80642-5. URL http://cs.au.dk/$\sim$birke/papers/devttc.pdf.

Errett Bishop. *Foundations of Constructive Analysis*. McGraw-Hill, 1967.

B Bloom and J G Riecke. LCF Should Be Lifted. 1989.

Manuel Blum. On the size of machines. *Information and Control*, 11(3):257–265, sep 1967. ISSN 00199958. doi: 10.1016/S0019-9958(67)90546-3. URL http://linkinghub.elsevier.com/retrieve/pii/S0019995867905463.

Achim Blumensath and Viktor Winschel. A Coalgebraic Framework for Games in Economics. 2013.

Guillaume Bonfante, Matthieu Kaczmarek, and Jean-Yves Marion. Toward an Abstract Computer Virology. In Dan Van Hung and Martin Wirsing, editors, *Theoretical Aspects of Computing – ICTAC 2005: Second International Colloquium, Hanoi, Vietnam, October 17-21, 2005. Proceedings*, volume 3722 of *Lecture Notes in Computer Science*, pages 579–593. Springer Berlin Heidelberg, 2005. ISBN

3540291075. doi: 10.1007/11560647_38. URL http://link.springer.com/10.1007/11560647_38.

Guillaume Bonfante, Matthieu Kaczmarek, and J.-Y. Marion. On Abstract Computer Virology from a Recursion Theoretic Perspective. *Journal in Computer Virology*, 1(3):45–54, 2006. ISSN 1772-9890. doi: 10.1007/s11416-005-0007-4. URL http://link.springer.com/10.1007/s11416-005-0007-4.

Guillaume Bonfante, Matthieu Kaczmarek, and Jean-Yves Marion. A Classification of Viruses Through Recursion Theorems. In S Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *Computation and Logic in the Real World: Third Conference on Computability in Europe, CiE 2007, Siena, Italy, June 18-23, 2007. Proceedings*, volume 4497 of *Lecture Notes in Computer Science*, pages 73–82. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi: 10.1007/978-3-540-73001-9_8. URL http://link.springer.com/10.1007/978-3-540-73001-9_8.

George S. Boolos. *The Logic of Provability*. Cambridge University Press, Cambridge, 1994. ISBN 9780511625183. doi: 10.1017/CBO9780511625183. URL https://dx.doi.org/10.1017/CBO9780511625183.

Ana Bove and Venanzio Capretta. Modelling general recursion in type theory. *Mathematical Structures in Computer Science*, 15(4):671–708, aug 2005.

Torben Braüner. The Girard Translation Extended with Recursion. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, number Lecture Notes in Computer Science 933, pages 31–45, 1995. ISBN 3-540-60017-5. doi: 10.1007/BFb0022245.

Torben Braüner. A general adequacy result for a linear functional language. *Theoretical Computer Science*, 177(1):27–58, 1997. ISSN 03043975. doi: 10.1016/S0304-3975(96)00233-2.

Venanzio Capretta. General recursion via coinductive types. *Logical Methods in Computer Science*, 1(2):1–28, jul 2005. ISSN 18605974. doi: 10.2168/LMCS-1(2:1)2005. URL http://www.lmcs-online.org/ojs/viewarticle.php?id=55.

John Case and Samuel E. Moelius. Properties Complementary to Program Self-reference. In Luděk Kučera and Antonín Kučera, editors, *Proceedings*

of the 32nd International Symposium on Mathematical Foundations of Computer Science 2007 (MFCS'07), volume 4708 of *Lecture Notes in Computer Science*, pages 253–263, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. doi: 10.1007/978-3-540-74456-6_24. URL https://dx.doi.org/10.1007/978-3-540-74456-6_24.

John Case and Samuel E. Moelius. Characterizing programming systems allowing program self-reference. *Theory of Computing Systems*, 45(4):756–772, 2009a. ISSN 14324350. doi: 10.1007/s00224-009-9168-8. URL https://dx.doi.org/10.1007/s00224-009-9168-8.

John Case and Samuel E. Moelius. Independence Results for n-Ary Recursion Theorems. In Miroslaw Kutylowski, Witold Charatonik, and Maciej Gebala, editors, *Fundamentals of Computation Theory: Proceedings of the 17th International Symposium, FCT 2009, Wrocław, Poland, September 2-4, 2009*, volume 5699 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2009b. ISBN 978-3-642-03408-4. doi: 10.1007/978-3-642-03409-1_5. URL https://dx.doi.org/10.1007/978-3-642-03409-1_5.

John Case and Samuel E. Moelius. Program Self-Reference in Constructive Scott Subdomains. *Theory of Computing Systems*, 51(1):22–49, 2012. ISSN 14324350. doi: 10.1007/s00224-011-9372-1. URL https://dx.doi.org/10.1007/s00224-011-9372-1.

J. R. B. Cockett and Pieter J. W. Hofstra. Introduction to Turing categories. *Annals of Pure and Applied Logic*, 156(2-3):183–209, 2008. doi: 10.1016/j.apal.2008.04.005. URL http://dx.doi.org/10.1016/j.apal.2008.04.005.

J. R. B. Cockett and Pieter J. W. Hofstra. Categorical simulations. *Journal of Pure and Applied Algebra*, 214(10):1835–1853, 2010. ISSN 00224049. doi: 10.1016/j.jpaa.2009.12.028. URL http://dx.doi.org/10.1016/j.jpaa.2009.12.028.

Fred Cohen. Computational aspects of computer viruses. *Computers and Security*, 8 (4):325–344, 1989. ISSN 01674048. doi: 10.1016/0167-4048(89)90094-1.

Robert L. Constable and Scott F. Smith. Computational foundations of basic recursive function theory. *Theoretical Computer Science*, 121(1-2):89–112, dec 1993. ISSN 03043975. doi: 10.1016/0304-3975(93)90085-8. URL https://dx.doi.org/10.1016/0304-3975(93)90085-8.

Roy L. Crole. *Categories for Types*. Cambridge University Press, 1993. ISBN 0 521 45701 7.

Djordje Čubrić, Peter Dybjer, and Philip J. Scott. Normalization and the Yoneda embedding. *Mathematical Structures in Computer Science*, 8(2):153–192, 1998. doi: 10.1017/s0960129597002508. URL `https://dx.doi.org/10.1017/s0960129597002508`.

Nigel Cutland. *Computability: An Introduction to Recursive Function Theory*. Cambridge University Press, 1980. ISBN 9780521294652.

Olivier Danvy and Karoline Malmkjaer. Intensions and extensions in a reflective tower. In *Proceedings of the 1988 ACM conference on LISP and functional programming (LFP '88)*, pages 327–341, New York, New York, USA, 1988. ACM Press. ISBN 089791273X. doi: 10.1145/62678.62725. URL `https://dx.doi.org/10.1145/62678.62725`.

Rowan Davies. A Temporal-Logic Approach to Binding-Time Analysis. Technical report, BRICS Report Series RS-95-51, 1995.

Rowan Davies. A temporal-logic approach to binding-time analysis. *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 184–195, 1996. ISSN 1043-6871. doi: 10.1109/LICS.1996.561317.

Rowan Davies. A Temporal Logic Approach to Binding-Time Analysis. *Journal of the ACM*, 64(1):1–45, mar 2017. ISSN 00045411. doi: 10.1145/3011069. URL `http://dx.doi.org/10.1145/3011069`.

Rowan Davies and Frank Pfenning. A modal analysis of staged computation. In *Proceedings of the 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'96)*, pages 258–270, 1996. ISBN 0897917693. doi: 10.1145/382780.382785. URL `https://dx.doi.org/10.1145/382780.382785`.

Rowan Davies and Frank Pfenning. A modal analysis of staged computation. *Journal of the ACM*, 48(3):555–604, 2001. ISSN 00045411. doi: 10.1145/382780.382785. URL `https://dx.doi.org/10.1145/382780.382785`.

François-Nicola Demers and Jacques Malenfant. Reflection in logic, functional and object-oriented programming: a Short Comparative Study. In *Proceedings of the IJCAI '95 Workshop on Reflection and Metalevel Architectures and their Applications in AI*, pages 29–38, 1995.

Apostolos Doxiadis. *Uncle Petros and Goldbach's Conjecture*. Faber and Faber, 2001.

Samuel Eilenberg and G. Max Kelly. Closed Categories. In *Proceedings of the Conference on Categorical Algebra*, pages 421–562. Springer Berlin Heidelberg, Berlin, Heidelberg, 1966. doi: 10.1007/978-3-642-99902-4_22. URL http://www.springerlink.com/index/10.1007/978-3-642-99902-4_22.

A. P. Ershov. On the partial computation principle. *Information Processing Letters*, 6(2):38–41, apr 1977. ISSN 00200190. doi: 10.1016/0020-0190(77)90078-3. URL https://dx.doi.org/10.1016/0020-0190(77)90078-3.

A. P. Ershov. Mixed computation: potential applications and problems for study. *Theoretical Computer Science*, 18(1):41–67, apr 1982. ISSN 03043975. doi: 10.1016/0304-3975(82)90111-6. URL http://linkinghub.elsevier.com/retrieve/pii/0304397582901116.

Martín Hötzel Escardó. A metric model of PCF, 1999. URL http://cs.bham.ac.uk/$\sim$mhe/papers/metricpcf.pdf.

Melvin Fitting. Intensional Logic. In Edward N Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer '15 edition, 2015. URL https://plato.stanford.edu/archives/sum2015/entries/logic-intensional/.

Daniel P. Friedman and Mitchell Wand. Reification: Reflection without metaphysics. In *Proceedings of the 1984 ACM Symposium on LISP and functional programming (LFP '84)*, pages 348–355, New York, New York, USA, 1984. ACM Press. ISBN 0897911423. doi: 10.1145/800055.802051. URL https://dx.doi.org/10.1145/800055.802051.

Harvey Friedman. FOM: renaming recursion theory, 1998. URL http://www.cs.nyu.edu/pipermail/fom/1998-August/002017.html.

Yoshihiko Futamura. Partial evaluation of computation process–an approach to a compiler-compiler. *Higher-Order and Symbolic Computation*, 12(4):381–391, 1999. ISSN 13883690. doi: 10.1023/A:1010095604496. URL http://link.springer.com/article/10.1023/A:1010095604496.

Murdoch J. Gabbay and Aleksandar Nanevski. Denotation of contextual modal type theory (CMTT): Syntax and meta-programming. *Journal of Applied Logic*, 11

(1):1–29, mar 2013. ISSN 15708683. doi: 10.1016/j.jal.2012.07.002. URL http://dx.doi.org/10.1016/j.jal.2012.07.002.

R Gandy. The Confluence of Ideas in 1936. In *A Half-century Survey on The Universal Turing Machine*, pages 55–111, New York, NY, USA, 1988. Oxford University Press, Inc. ISBN 0-19-853741-7. URL http://dl.acm.org/citation.cfm?id=57249.57252.

R. O. Gandy. On the axiom of extensionality – Part I. *The Journal of Symbolic Logic*, 21(01):36–48, mar 1956. ISSN 0022-4812. doi: 10.2307/2268484. URL https://www.cambridge.org/core/product/identifier/S0022481200085352/type/journal_article.

R. O. Gandy. On the axiom of extensionality, Part II. *The Journal of Symbolic Logic*, 24(04):287–300, dec 1959. ISSN 0022-4812. doi: 10.2307/2963897. URL https://www.cambridge.org/core/product/identifier/S0022481200123266/type/journal_article.

Jean-Yves Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.

Jean-Yves Girard. *The Blind Spot: Lectures on Logic*. European Mathematical Society, 2011. ISBN 978-3037190883.

Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge University Press, 1989.

Paul Graham. *On LISP: Advanced Techniques for Common LISP*. Prentice Hall, 1993. ISBN 0130305529.

Carl A. Gunter. *Semantics of programming languages: structures and techniques*. Foundations of Computing. The MIT Press, 1992.

Torben Amtoft Hansen, Thomas Nikolajsen, Jesper Larsson Träff, and Neil D. Jones. Experiments with Implementations of Two Theoretical Constructions. In *Proceedings of the Symposium on Logical Foundations of Computer Science: Logic at Botik '89*, pages 119–133, London, UK, 1989. Springer-Verlag. ISBN 3-540-51237-3. URL http://dl.acm.org/citation.cfm?id=646798.759646.

Robert Harper. Old Neglected Theorems Are Still Theorems, 2014. URL https://existentialtype.wordpress.com/2014/03/20/old-neglected-theorems-are-still-theorems/.

Douglas R Hofstadter. *Godel, Escher, Bach: An Eternal Golden Braid*. Basic Books, Inc., New York, NY, USA, 1979. ISBN 0465026850.

Hagen Huwig and Axel Poigné. A note on inconsistencies caused by fixpoints in a cartesian closed category. *Theoretical Computer Science*, 73(1):101–112, jun 1990. ISSN 03043975. doi: 10.1016/0304-3975(90)90165-E. URL http://linkinghub.elsevier.com/retrieve/pii/030439759090165E.

J. M. E. Hyland. The Effective Topos. In Anne S. Troelstra and Dick van Dalen, editors, *The L. E. J. Brouwer Centenary Symposium*, pages 165–216. North-Holland, 1982. URL https://webdpmms.maths.cam.ac.uk/$\sim$martin/Research/Oldpapers/hyland-effectivetopos.pdf.

J.M.E. Hyland. The Forgotten Turing. In S. Barry Cooper and Andrew Hodges, editors, *The Once and Future Turing*, pages 20–33. Cambridge University Press, Cambridge, 2016. doi: 10.1017/CBO9780511863196.005. URL http://ebooks.cambridge.org/ref/id/CBO9780511863196A012.

J.M.E. Hyland and C.-H.L. Ong. On Full Abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000. doi: 10.1006/inco.2000.2917. URL http://linkinghub.elsevier.com/retrieve/pii/S0890540100929171.

Martin Hyland and John Power. The Category Theoretic Understanding of Universal Algebra: Lawvere Theories and Monads. *Electronic Notes in Theoretical Computer Science*, 172:437–458, 2007. ISSN 15710661. doi: 10.1016/j.entcs.2007.02.019.

Bart Jacobs. *Categorical Logic and Type Theory*. Number 141 in Studies in Logic and the Foundations of Mathematics. North Holland, Amsterdam, 1999.

Barry Jay. Programs as Data Structures in λSF-Calculus. *Electronic Notes in Theoretical Computer Science*, 325:221–236, 2016. ISSN 15710661. doi: 10.1016/j.entcs.2016.09.040. URL http://dx.doi.org/10.1016/j.entcs.2016.09.040.

Barry Jay and Thomas Given-Wilson. A Combinatory Account of Internal Structure. *The Journal of Symbolic Logic*, 76(3):807–826, 2011. URL http://www.jstor.org/stable/23041848.

Barry Jay and Jens Palsberg. Typed self-interpretation by pattern matching. *ACM SIGPLAN Notices*, 46(9):247, sep 2011. ISSN 03621340. doi: 10.1145/2034574.2034808. URL `http://dl.acm.org/citation.cfm?doid=2034574.2034808`.

Neil D. Jones. Computer Implementation and Applications of Kleene's S-M-N and Recursion Theorems. In Yiannis N Moschovakis, editor, *Logic from Computer Science: Proceedings of a Workshop Held November 13-17, 1989 [at MSRI]*, volume 21 of *Mathematical Sciences Research Institute Publications*, pages 243–263. Springer New York, 1992. doi: 10.1007/978-1-4612-2822-6_9. URL `https://dx.doi.org/10.1007/978-1-4612-2822-6_9`.

Neil D. Jones. An introduction to partial evaluation. *ACM Computing Surveys*, 28(3):480–503, 1996. ISSN 03600300. doi: 10.1145/243439.243447. URL `http://doi.acm.org/10.1145/243439.243447`.

Neil D. Jones. *Computability and Complexity: From a Programming Perspective*. Foundations of Computing. MIT Press, 1997.

Neil D. Jones. A Swiss Pocket Knife for Computability. *Electronic Proceedings in Theoretical Computer Science*, 129:1–17, sep 2013. ISSN 2075-2180. doi: 10.4204/EPTCS.129.1. URL `http://arxiv.org/abs/1309.5128v1`.

Neil D. Jones, Carsten K. Gomard, and Peter Sestoft. *Partial Evaluation and Automatic Program Generation*. Prentice Hall International, 1993. ISBN 0-13-020249-5.

Akira Kanda. Recursion theorems and effective domains. *Annals of Pure and Applied Logic*, 38(3):289–300, 1988. ISSN 01680072. doi: 10.1016/0168-0072(88)90029-2.

G. A. Kavvos. The Many Worlds of Modal λ-calculi: I. Curry-Howard for Necessity, Possibility and Time. *CoRR*, 2016.

G. A. Kavvos. On the Semantics of Intensionality. In Javier Esparza and Andrzej S. Murawski, editors, *Proceedings of the 20th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 10203 of *Lecture Notes in Computer Science*, pages 550–566. Springer-Verlag Berlin Heidelberg, 2017a. doi: 10.1007/978-3-662-54458-7_32. URL `https://dx.doi.org/10.1007/978-3-662-54458-7_32`.

G. A. Kavvos. Dual-context calculi for modal logic. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 2017b. ISBN 978-1-5090-3018-7. doi: 10.1109/LICS.2017.8005089.

G. A. Kavvos. Dual-context calculi for modal logic (technical report). Technical report, University of Oxford, 2017c. URL http://www.lambdabetaeta.eu/papers/dualcalc.pdf.

G. A. Kavvos. Intensionality, Intensional Recursion, and the Gödel-Löb axiom. In *Proceedings of 7th Workshop on Intuitionistic Modal Logic and Applications (IMLA 2017)*, 2017d.

Oleg Kiselyov. In'yō to ichiokubai kōsokuka no monogatari: Kansū puroguramingu ni yoru Kleene daini saiki teiri no shōmei [A story of quotation and 10^8-fold speed-up: A proof of Kleene's second recursion theorem by functional programming]. In *The 17th Programming and Programming Language Workshop (PPL 2015)*, 2015. URL http://okmij.org/ftp/Computation/Kleene.pdf.

S. C. Kleene. λ-definability and recursiveness. *Duke Mathematical Journal*, 2(2):340–353, 1936. ISSN 0012-7094. doi: 10.1215/S0012-7094-36-00227-2. URL https://dx.doi.org/10.1215/S0012-7094-36-00227-2.

Stephen C. Kleene. On notation for ordinal numbers. *The Journal of Symbolic Logic*, 3(04):150–155, 1938. ISSN 0022-4812. doi: 10.2307/2267778. URL https://dx.doi.org/10.2307/2267778.

Stephen C. Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, 1952.

Stephen C. Kleene. Origins of Recursive Function Theory. *IEEE Annals of the History of Computing*, 3(1):52–67, 1981. ISSN 1058-6180. doi: 10.1109/MAHC.1981.10004. URL https://dx.doi.org/10.1109/MAHC.1981.10004.

Satoshi Kobayashi. Monad as modality. *Theoretical Computer Science*, 175(1):29–74, 1997. doi: 10.1016/S0304-3975(96)00169-7.

Joachim Lambek and Philip J. Scott. *Introduction to Higher-Order Categorical Logic*. Cambridge University Press, 1988. ISBN 9780521356534.

Elaine Landry, editor. *Categories for the Working Philosopher*. Oxford University Press, 2017. ISBN 9780198748991.

J.-L. Lassez, V.L. Nguyen, and E.a. Sonenberg. Fixed point theorems and semantics: a folk tale. *Information Processing Letters*, 14(3):112–116, 1982. ISSN 00200190. doi: 10.1016/0020-0190(82)90065-5.

F. William Lawvere. Diagonal arguments and cartesian closed categories. In *Category Theory, Homology Theory and their Applications II*, number 15, pages 134–145. Springer Berlin Heidelberg, 1969. ISBN 978-3-540-04611-0. doi: 10.1007/BFb0080769. URL http://link.springer.com/content/pdf/10.1007/BFb0080769.pdf.

F. William Lawvere. Diagonal arguments and cartesian closed categories. *Reprints in Theory and Applications of Categories*, 15:1–13, 2006. URL http://www.tac.mta.ca/tac/reprints/articles/15/tr15abs.html.

Paul Blain Levy. *Call-by-Push-Value: A Functional-Imperative Synthesis*. Semantic Structures in Computation. Springer, 2003. ISBN 978-94-010-3752-5. doi: 10.1007/978-94-007-0954-6.

Harry R Lewis and Christos H Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1997. ISBN 0132624788.

Tadeusz Litak. Constructive Modalities with Provability Smack. In Guram Bezhanishvili, editor, *Leo Esakia on duality in modal and intuitionistic logics*, pages 179–208. Springer, 2014. doi: 10.1007/978-94-017-8860-1_8. URL https://www8.cs.fau.de/ext/litak/esakiaarxivfull.pdf.

John R. Longley. *Realizability Toposes and Language Semantics*. PhD thesis, University of Edinburgh. College of Science and Engineering. School of Informatics., 1995. URL http://www.lfcs.inf.ed.ac.uk/reports/95/ECS-LFCS-95-332/.

John R. Longley. Matching typed and untyped realizability. *Electronic Notes in Theoretical Computer Science*, 23(1):74–100, 1999a. ISSN 15710661. doi: 10.1016/S1571-0661(04)00105-7. URL http://linkinghub.elsevier.com/retrieve/pii/S1571066104001057.

John R. Longley. Unifying Typed and Untyped Realizability, 1999b. URL http://homepages.inf.ed.ac.uk/jrl/Research/unifying.txt.

John R. Longley. Notions of computability at higher types I. In *Logic Colloquium 2000: Proceedings of the Annual European Summer Meeting of the Association for Symbolic Logic, held in Paris, France, July 23-31, 2000*, volume 19 of *Lecture Notes in Logic*, pages 32–142. A. K. Peters, 2005.

John R. Longley and Dag Normann. *Higher-Order Computability*. Theory and Applications of Computability. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. ISBN 978-3-662-47991-9. doi: 10.1007/978-3-662-47992-6. URL https://dx.doi.org/10.1007/978-3-662-47992-6.

John R. Longley and Alex K. Simpson. A uniform approach to domain theory in realizability models. *Mathematical Structures in Computer Science*, 7:469–505, 1997. doi: 10.1017/S0960129597002387. URL https://dx.doi.org/10.1017/S0960129597002387.

Giuseppe Longo and Eugenio Moggi. Constructive natural deduction and its 'omega-set' interpretation. *Mathematical Structures in Computer Science*, 1(02):215, 1991. ISSN 0960-1295. doi: 10.1017/S0960129500001298. URL http://www.journals.cambridge.org/abstract_S0960129500001298.

Saunders Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer New York, New York, NY, 1978. ISBN 978-1-4419-3123-8. doi: 10.1007/978-1-4757-4721-8. URL http://link.springer.com/10.1007/978-1-4757-4721-8.

Michael Machtey and Paul Young. *An introduction to the general theory of algorithms*. Theory of Computation Series. Elsevier North-Holland, New York, 1978. ISBN 9780444002266. URL http://books.google.co.uk/books?id=qncEAQAAIAAJ.

Michael Machtey, Karl Winklmann, and Paul Young. Simple Gödel Numberings, Isomorphisms, and Programming Properties. *SIAM Journal on Computing*, 7(1): 39–60, feb 1978. ISSN 0097-5397. doi: 10.1137/0207003. URL https://dx.doi.org/10.1137/0207003.

John Maraist, Martin Odersky, David N Turner, and Philip Wadler. Call-by-name, call-by-value, call-by-need, and the linear lambda calculus. *Electronic Notes in Theoretical Computer Science*, 1:370–392, 1995.

Jean-Yves Marion. From Turing machines to computer viruses. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370 (1971):3319–3339, 2012. ISSN 1364-503X. doi: 10.1098/rsta.2011.0332. URL http://rsta.royalsocietypublishing.org/cgi/doi/10.1098/rsta.2011.0332.

Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in Proof Theory*. Bibliopolis, 1984. ISBN 88-7088-105-9.

Per Martin-Löf. An intuitionistic theory of types. In Giovanni Sambin and Jan M Smith, editors, *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 127–172. Oxford University Press, 1998.

Paul-André Melliès. Categorical Semantics of Linear Logic. In Pierre-Louis Curien, Hugo Herbelin, Jean-Louis Krivine, and Paul-André Melliès, editors, *Panoramas et synthèses 27: Interactive models of computation and program behaviour*. Société Mathématique de France, 2009. ISBN 978-2-85629-273-0. URL `http://www.pps.univ-paris-diderot.fr/$\sim$mellies/papers/panorama.pdf`.

John C. Mitchell. *Foundations for programming languages*. Foundations of Computing. The MIT Press, 1996. ISBN 9780262133210.

Samuel E. Moelius. *Program Self-Reference*. PhD thesis, University of Delaware, 2009.

Torben Æ. Mogensen. Efficient self-interpretation in lambda calculus. *Journal of Functional Programming*, 2(03):345–364, jul 1992. ISSN 0956-7968. doi: 10.1017/S0956796800000423. URL `http://www.journals.cambridge.org/abstract_S0956796800000423`.

Eugenio Moggi. Computational lambda-calculus and monads. In *[1989] Proceedings. Fourth Annual Symposium on Logic in Computer Science*, pages 14–23. IEEE Comput. Soc. Press, 1989. ISBN 0-8186-1954-6. doi: 10.1109/LICS.1989.39155.

Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991. ISSN 08905401. doi: 10.1016/0890-5401(91)90052-4. URL `https://dx.doi.org/10.1016/0890-5401(91)90052-4`.

Yiannis N. Moschovakis. Sense and Denotation as Algorithm and Value. *Logic Colloquium '90: ASL Summer Meeting in Helsinki*, 2:210–249, 1993. URL `http://www.math.ucla.edu/$\sim$ynm/papers/frege.pdf`.

Yiannis N Moschovakis. Kleene's Amazing Second Recursion Theorem. *Bulletin of Symbolic Logic*, 16(2):189–239, 2010.

Philip S. Mulry. Generalized Banach-Mazur functionals in the topos of recursive sets. *Journal of Pure and Applied Algebra*, 26(1):71–83, oct 1982. ISSN 00224049. doi: 10.1016/0022-4049(82)90030-5. URL `http://linkinghub.elsevier.com/retrieve/pii/0022404982900305`.

Philip S. Mulry. A categorical approach to the theory of computation. *Annals of Pure and Applied Logic*, 43(3):293–305, aug 1989. ISSN 01680072. doi: 10.1016/0168-0072(89)90072-9. URL http://linkinghub.elsevier.com/retrieve/pii/0168007289900729.

J. Myhill and J. C. Shepherdson. Effective operations on partial recursive functions. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 1(4):310–317, 1955. ISSN 00443050. doi: 10.1002/malq.19550010407. URL http://doi.wiley.com/10.1002/malq.19550010407.

Aleksandar Nanevski. Meta-programming with names and necessity. *ACM SIGPLAN Notices*, 37:206–217, 2002. ISSN 03621340. doi: 10.1145/583852.581498.

Aleksandar Nanevski and Frank Pfenning. Staged computation with names and necessity. *Journal of Functional Programming*, 15(06):893, 2005. ISSN 0956-7968. doi: 10.1017/S095679680500568X.

Flemming Nielson and Hanne Riis Nielson. *Two-Level Functional Languages*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1992. ISBN 9780521018470.

Bengt Nordström, Kent Petersson, and Jan M. Smith. *Programming in Martin-Löf's Type Theory: an Introduction*. Oxford University Press, 1990. ISBN 0-19-853814-6. doi: 10.1016/0377-0427(91)90052-L.

Piergiorgio Odifreddi. *Classical recursion theory: The theory of functions and sets of natural numbers*. Elsevier, 1992. ISBN 9780444894830.

Robert A. Di Paola and Alex Heller. Dominical Categories: Recursion Theory without Elements. *The Journal of Symbolic Logic*, 52(3):594, sep 1987. ISSN 00224812. doi: 10.2307/2274352. URL http://www.jstor.org/stable/2274352?origin=crossref.

Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(4):511–540, 2001. ISSN 0960-1295. doi: 10.1017/S0960129501003322. URL https://dx.doi.org/10.1017/S0960129501003322.

Richard Alan Platek. *Foundations of Recursion Theory*. PhD thesis, Stanford University, 1966.

Gordon Plotkin and John Power. Computational Effects and Operations: An Overview. *Electronic Notes in Theoretical Computer Science*, 73(March 2002): 149–163, oct 2004. ISSN 15710661. doi: 10.1016/j.entcs.2004.08.008. URL http://linkinghub.elsevier.com/retrieve/pii/S1571066104050893.

Gordon D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223–255, 1977. ISSN 03043975. doi: 10.1016/0304-3975(77)90044-5.

Gordon D. Plotkin. Type theory and recursion. In *Proceedings Eighth Annual IEEE Symposium on Logic in Computer Science*, page 374. IEEE Comput. Soc. Press, 1993. ISBN 0-8186-3140-6. doi: 10.1109/LICS.1993.287571. URL https://dx.doi.org/10.1109/LICS.1993.287571.

Axel Poigné. Basic category theory. In *Handbook of Logic in Computer Science*. Clarendon Press, 1992. ISBN 9780198537359.

Andrew Polonsky. Axiomatizing the Quote. In Marc Bezem, editor, *Computer Science Logic (CSL'11) - 25th International Workshop/20th Annual Conference of the EACSL*, volume 12 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 458–469. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. doi: 10.4230/LIPIcs.CSL.2011.458. URL https://dx.doi.org/10.4230/LIPIcs.CSL.2011.458.

Gregory A. Riccardi. *The Independence of Control Structures in Abstract Programming Systems*. PhD thesis, State University of New York at Buffalo, 1980.

Gregory A. Riccardi. The independence of control structures in abstract programming systems. *Journal of Computer and System Sciences*, 22(2):107–143, apr 1981. ISSN 00220000. doi: 10.1016/0022-0000(81)90024-6. URL https://dx.doi.org/10.1016/0022-0000(81)90024-6.

Hartley Rogers. Gödel numberings of partial recursive functions. *The Journal of Symbolic Logic*, 23(03):331–341, sep 1958. ISSN 0022-4812. doi: 10.2307/2964292. URL http://www.journals.cambridge.org/abstract_S0022481200058011.

Hartley Rogers. *Theory of recursive functions and effective computability*. MIT Press, Cambridge, MA, USA, 1987. ISBN 0-262-68052-1.

James S. Royer. *A Connotational Theory of Program Structure*, volume 273 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987. ISBN 978-3-540-18253-5. doi: 10.1007/3-540-18253-5. URL http://link.springer.com/10.1007/3-540-18253-5www.cis.syr.edu/$\sim$royer/archive/ctps.ps.

James S. Royer and John Case. *Subrecursive Programming Systems*. Birkhäuser Boston, Boston, MA, 1994. ISBN 978-1-4612-6680-8. doi: 10.1007/978-1-4612-0249-3. URL http://link.springer.com/10.1007/978-1-4612-0249-3.

Dana Scott. Lambda Calculus and Recursion Theory (Preliminary Version). In Stig Kanger, editor, *Proceedings of the Third Scandinavian Logic Symposium*, volume 82 of *Studies in Logic and the Foundations of Mathematics*, pages 154–193. North-Holland, 1975. doi: 10.1016/S0049-237X(08)70730-4. URL http://linkinghub.elsevier.com/retrieve/pii/S0049237X08707304.

Dana S. Scott. Data Types as Lattices. *SIAM Journal on Computing*, 5(3):522–587, 1976. doi: 10.1137/0205037.

Dana S. Scott. A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121(1-2):411–440, 1993. ISSN 03043975. doi: 10.1016/0304-3975(93)90095-B.

Alex K. Simpson and Gordon D. Plotkin. Complete axioms for categorical fixed-point operators. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science (LICS 2000)*, pages 30–41. IEEE Comput. Soc, 2000. ISBN 0-7695-0725-5. doi: 10.1109/LICS.2000.855753. URL http://ieeexplore.ieee.org/document/855753/.

Brian Cantwell Smith. *Procedural reflection in programming languages*. PhD thesis, Massachusetts Institute of Technology, 1982. URL http://hdl.handle.net/1721.1/15961.

Brian Cantwell Smith. Reflection and Semantics in LISP. In *Proceedings of the 11th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '84)*, pages 23–35, New York, New York, USA, 1984. ACM Press. ISBN 0897911253. doi: 10.1145/800017.800513. URL https://dx.doi.org/10.1145/800017.800513.

Raymond M. Smullyan. *Gödel's Incompleteness Theorems*. Oxford University Press, 1992.

Sam Staton. Freyd categories are Enriched Lawvere Theories. *Electronic Notes in Theoretical Computer Science*, 303:197–206, 2014. ISSN 15710661. doi: 10.1016/j.entcs.2014.02.010. URL http://dx.doi.org/10.1016/j.entcs.2014.02.010.

Christopher Strachey. Fundamental Concepts in Programming Languages. *Higher-Order and Symbolic Computation*, 13:11–49, 2000. ISSN 13883690. doi: 10.1023/A:1010000313106. URL http://dx.doi.org/10.1023/A:1010000313106.

Thomas Streicher. *Domain-theoretic Foundations of Functional Programming*. World Scientific, 2006.

Thomas Streicher. How Intensional Is Homotopy Type Theory? In Maria del Mar González, Paul C. Yang, Nicola Gambino, and Joachim Kock, editors, *Extended Abstracts Fall 2013: Geometrical Analysis; Type Theory, Homotopy Theory and Univalent Foundations*, number September in Research Perspectives CRM Barcelona, pages 105–110. Birkhäuser Basel, Cham, 2015. ISBN 978-3-319-21283-8. doi: 10.1007/978-3-319-21284-5_20. URL https://dx.doi.org/10.1007/978-3-319-21284-5_20.

Walid Taha and Michael Florentin Nielsen. Environment classifiers. *ACM SIGPLAN Notices*, 38:26–37, 2003. ISSN 03621340. doi: 10.1145/640128.604134.

Walid Taha and Tim Sheard. Multi-stage programming with explicit annotations. In *Proceedings of the 1997 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation (PEPM '97)*, pages 203–217, New York, New York, USA, 1997. ACM Press. ISBN 0897919173. doi: 10.1145/258993.259019. URL https://dx.doi.org/10.1145/258993.259019.

Walid Taha and Tim Sheard. MetaML and multi-stage programming with explicit annotations. *Theoretical Computer Science*, 248(1-2):211–242, 2000. ISSN 03043975. doi: 10.1016/S0304-3975(00)00053-0. URL https://dx.doi.org/10.1016/S0304-3975(00)00053-0.

M. Takahashi. Parallel Reductions in λ-Calculus. *Information and Computation*, 118(1):120–127, apr 1995. ISSN 08905401. doi: 10.1006/inco.1995.1057. URL https://dx.doi.org/10.1006/inco.1995.1057.

Takeshi Tsukada and Atsushi Igarashi. A logical foundation for environment classifiers. *Logical Methods in Computer Science*, 6(4):1–43, 2010. ISSN 18605974. doi: 10.2168/LMCS-6(4:8)2010. URL https://dx.doi.org/10.2168/LMCS-6(4:8)2010.

Alan M Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, jan 1937. ISSN 0024-6115. doi: 10.1112/plms/s2-42.1.230. URL http://plms.oxfordjournals.org/cgi/doi/10.1112/plms/s2-42.1.230.

Alan M Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, 2(1):161–228, 1939.

Aldo Ursini. Intuitionistic diagonalizable algebras. *Algebra Universalis*, 9(1):229–237, 1979a. ISSN 00025240. doi: 10.1007/BF02488034.

Aldo Ursini. A modal calculus analogous to K4W, based on intuitionistic propositional logic. *Studia Logica*, 38(3):297–311, 1979b. ISSN 0039-3215. doi: 10.1007/BF00405387. URL https://dx.doi.org/10.1007/BF00405387.

Jaap van Oosten. *Realizability: An Introduction to its Categorical Side*, volume 152. Elsevier, 2008. ISBN 978-0-444-51584-1. URL http://www.sciencedirect.com/science/bookseries/0049237X/152.

Spyros Vassilakis. Economic Data Types. 1989.

Spyros Vassilakis. Some economic applications of Scott domains. *Mathematical Social Sciences*, 24(2-3):173–208, nov 1992. ISSN 01654896. doi: 10.1016/0165-4896(92)90061-9. URL http://linkinghub.elsevier.com/retrieve/pii/0165489692900619.

Kumaraswamy Velupillai. *Computable economics*. The Arne Ryde Memorial Lectures. Oxford University Press, 2000. ISBN 978 1 84376 239 3.

P Wadler. A critique of Abelson and Sussman or why calculating is better than scheming. *ACM SIGPLAN Notices*, 22(3):83–94, mar 1987. ISSN 03621340. doi: 10.1145/24697.24706. URL https://dx.doi.org/10.1145/24697.24706.

Mitchell Wand. The Theory of Fexprs is Trivial. *LISP and Symbolic Computation*, 10(3):189–199, 1998. ISSN 08924635. doi: 10.1023/A:1007720632734. URL https://dx.doi.org/10.1023/A:1007720632734.

Mitchell Wand and Daniel P. Friedman. The mystery of the tower revealed: A nonreflective description of the reflective tower. *Lisp and Symbolic Computation*, 1(1):11–38, jun 1988. ISSN 0892-4635. doi: 10.1007/BF01806174. URL `https://dx.doi.org/10.1007/BF01806174`.

Steven J Winrich. Self-Reference and the Incomplete Structure of Neoclassical Economics. *Journal of Economic Issues*, 18(4):987–1005, 1984.