Realising Intensional S4 and GL Modalities

Liang-Ting Chen
Institute of Information Science, Academia Sinica, Taipei, Taiwan

Hsiang-Shang Ko
Institute of Information Science, Academia Sinica, Taipei, Taiwan

Abstract

There have been investigations into type-theoretic foundations for metaprogramming, notably Davies and Pfenning's (2001) treatment in **S4** modal logic, where code evaluating to values of type A is given the modal type $\texttt{Code}\,A$ ($\Box A$ in the original paper). Recently Kavvos (2017) extended PCF with $\texttt{Code}\,A$ and intensional recursion, understood as the deductive form of the **GL** (Gödel-Löb) axiom in provability logic, but the resulting type system is logically inconsistent. Inspired by staged computation, we observe that a term of type $\texttt{Code}\,A$ is, in general, code to be evaluated in a next stage, whereas **S4** modal type theory is a special case where code can be evaluated in the current stage, and the two types of code should be discriminated. Consequently, we use two separate modalities $\boxtimes$ and $\Box$ to model **S4** and **GL** respectively in a unified categorical framework while retaining logical consistency. Following Kavvos' (2017) novel approach to the semantics of intensionality, we interpret the two modalities in the $\mathscr{P}$-category of assemblies and trackable maps. For the **GL** modality $\Box$ in particular, we use guarded type theory to articulate what it means by a 'next' stage and to model intensional recursion by guarded recursion together with Kleene's second recursion theorem. Besides validating the **S4** and **GL** axioms, our model better captures the essence of intensionality by refuting congruence (so that two extensionally equal terms may not be intensionally equal) and internal quoting (both $A \to \Box A$ and $A \to \boxtimes A$). Our results are developed in (guarded) homotopy type theory and formalised in Agda.

2012 ACM Subject Classification Theory of computation → Type theory

Keywords and phrases provability, guarded recursion, realisability, modal types, metaprogramming

Digital Object Identifier 10.4230/LIPIcs.CSL.2022.4

Supplementary Material The results were formalised in Agda 2.6.2 in the guarded cubical mode, and the source code is available at `https://doi.org/10.5281/zenodo.5602771`.

Funding This work was partially supported by EPSRC grant number EP/N028139/1 and supported by the Ministry of Science and Technology of Taiwan under grant MOST 109-2222-E-001-002-MY3.

Acknowledgements We are grateful to Alex Kavvos and Tsung-Ju Chiang for insightful discussions. We would also like to thank Jacques Carette, Tom de Jong, Churn-Jung Liau, Rasmus Ejlers Møgelberg, Chad Nester, Anton Setzer, Andrea Vezzosi, Ren-June Wang, and Zhixuan Yang for useful exchanges. Finally, we thank the anonymous reviewers for their thoughtful comments.

1 Introduction

Metaprogramming is the activity of writing metaprograms that manipulate program code. Executing a metaprogram can result in another program to be executed, and these successive executions are abstractly referred to as computation *stages*. A particular form of metaprogramming is *staged computation*, where fragments of a program are *internally* marked to be evaluated in multiple stages, so that the program can be partially evaluated to produce more efficient code. The stratification of computation stages forms possible worlds and can be ideally reasoned about by modal logic. Therefore, there have been investigations into type-theoretic foundations for staged computation with modalities [9, 10, 16, 22], which have influenced the design of practical implementations [17, 27, 28] to varying degrees.

Let `Code` A denote the type of code that evaluates to values of type A in a next stage. In Davies and Pfenning's analysis of staged computation [10], `Code` corresponds to the modality $\boxtimes$ in the intuitionistic modal logic **S4**. More specifically, the **4** axiom $\boxtimes A \to \boxtimes\boxtimes A$ corresponds to the use of code in the stage after the next, so code can be shared across all stages, a.k.a. *cross-stage persistence*; the **T** axiom $\boxtimes A \to A$ corresponds to the *evaluation* of code to its value in the same stage. For instance, in λ-calculus, it is well-known [4] that there are terms encoding the Gödel code of a code and evaluating a code respectively.

Recently, Kavvos proposed *intensional recursion* [16] for `Code` to construct a value recursively from its own code (intension). Logically, intensional recursion amounts to the **G**ödel-**L**öb axiom $\Box(\Box A \to A) \to \Box A$ for the modal logic **GL** [5], where $\Box A$ stands for 'A is provable'. Computationally, the behaviour of the **GL** axiom mirrors Kleene's second recursion theorem. In contrast to general recursion $(A \to A) \to A$, which constructs a value recursively from its value (extension), intensional recursion alone does not lead to logical inconsistency. Kavvos explored the computational capabilities of a variant of PCF extended with `Code` viewed as both **S4** and **GL** modalities. Unfortunately, when `Code` is designed in this way, the type system is inconsistent so that it cannot be treated as a logical foundation.

To make **S4** and **GL** coexist in a single system while maintaining logical consistency, our approach is to keep the two modalities $\boxtimes$ and $\Box$ separate. Intuitively, while both modalities model code and appear similar, there are crucial differences: In general, programs are to be evaluated in a *next* stage, but **S4** is a special case where next stages include the current one, so the result of evaluating a program can be immediately used in the current stage, as witnessed by **T**. On the other hand, a program constructed with **GL** can recursively refer to its own code, which must not be evaluated within the same stage or risk non-termination computationally and inconsistency logically, so stage distinction has to be kept for **GL**.

To illuminate the difference, we present in this paper a denotational semantics of two types of code—a type $\boxtimes A$ of code that can be evaluated in the current stage and a type $\Box A$ of code to be evaluated in a next stage. To distinguish between intensions and extensions, we build upon the previous work using $\mathscr{P}$-categories [8, 15], which have an additional partial equivalence relation on morphisms that models extensional equality, while the underlying equality on morphisms models intensional equality. We revisit elements of realisability and construct a $\mathscr{P}$-category of assemblies on λ-calculus. Roughly speaking, an assembly X on λ-calculus is a set $|X|$ of extensions associated with at least an intension in the form of a λ-term (the existence of such an intension is merely a property that holds for the extension), while trackable maps are pairs of a function and one of its intensions, and can be equipped with both extensional and intensional equalities (taking only the function part or both parts into account). The denotations $\boxtimes X$ and $\Box X$ of the two types of code both consist of pairs (M, x) of an extension x and an associated λ-term M whose associated intensions are λ-terms that are reducible to the Gödel code $\ulcorner M \urcorner$. The choice of using $\ulcorner M \urcorner$ (rather than M as chosen by Kavvos [15]) as the intensions of (M, x) prevents $X \to \boxtimes X$ from having the meaning of generic quoting. The difference between $\boxtimes X$ and $\Box X$ is what extensions are.

- For $\boxtimes X$, the extension part x in (M, x) coincides with the values of the set $|X|$. We can then validate the **S4** axioms, for example $\boxtimes X \to X$ (natural in X) by just projection.
- For $\Box X$, the extension x does not come from $|X|$ but a different set $\triangleright|X|$ whose elements denote values of $|X|$ that are available in a next, or *later*, stage. This set $\triangleright|X|$ can be expressed directly in a *guarded type theory* which features Nakano's later modality $\triangleright$ and *guarded recursion* [21]. By working in guarded type theory, $\Box X \to X$ can no longer be validated by projection (because x from (M, x) is available later rather than now) and is actually not possible; also intensional recursion can be modelled by guarded recursion.

The above constructions do not give rise to $\mathscr{P}$-functors but 'exposures' [15] so that congruence (the preservation of extensional equality) is not required by definition and is actually false. Hence, by refuting both congruence and generic quoting, which previous work [12, 15] did not achieve, our denotational semantics is more intensional in the sense that it provides a finer-grained equality which allows us to distinguish computationally equivalent intensions.

We use homotopy type theory (HoTT) [29] as our metalanguage, but this is only for access to a small subset of convenient HoTT features, which in particular does not include univalence. Specifically, our work is built upon the reformulated set theory and logic within HoTT, which enable us to be precise about notions such as sets and propositions, and existence with explicit witnesses versus 'mere' existence (for example, within HoTT we can easily distinguish between 'pairs of an extension and an intension' and 'extensions associated with at least an intension'). Moreover, implementations of HoTT are readily available, so we are able to formalise and verify our constructions, for which HoTT provides essential features that some other type theories lack, notably function extensionality: $(\forall(x : A).\, f\, x =_B g\, x) \to f =_{A\to B} g$. Indeed, our work has been formalised in Agda [11], which implements HoTT and guarded type theory respectively in the forms of cubical type theory [7, 32] and ticked cubical type theory [31] with clock quantification [18]. However, we do not use cubical arguments.

Plan of the paper

After recalling preliminaries on homotopy type theory, untyped λ-calculus, and $\mathscr{P}$-categories in Section 2, we present the $\mathscr{P}$-category of assemblies on λ-calculus and trackable maps developed in HoTT in Section 3 and the denotational semantics of $\boxtimes A$ and $\square A$ in Sections 4 and 5 respectively, and discuss related work in Section 6 and future work in Section 7.

2 Preliminaries

Most, if not all, of the materials in this section are standard, so we do not go into details.

2.1 Homotopy type theory

In type theory, between every two inhabitants x and y of a type A, there is a type $x =_A y$ of proofs that x and y are (propositionally) equal; given an equality proof $p : x =_A y$, any $z : B(x)$ can be converted to $\mathsf{transport}(p, z) : B(y)$. Between equality proofs p, $q : x =_A y$ there is again a higher-dimensional equality type $p =_{x=_A y} q$, and so on. HoTT identifies this infinite-dimensional structure of equality types as an abstract form of homotopy theory, where types are interpreted as spaces and equality proofs as paths; in particular, equality types are path spaces, and paths between paths are homotopies. We do not make use of the full generality of HoTT but work exclusively with *propositions* and *sets*, whose equality structure degenerates at higher dimensions.

A *proposition* P is a type whose equality types $x =_P y$, for x and $y : P$, are all inhabited—all inhabitants of P are deemed equal, so all we care about P is whether it is inhabited, not its specific inhabitants. Hence, we will simply write $\star$ when referring to an inhabitant of any proposition. If P and Q are propositions, then so is their product $P \times Q$. Similarly, if a family $R(x)$ of types indexed by $x : A$ (where A is any type) are all propositions, then so is the product $\prod_{(x:A)} R(x)$. Logically these give us conjunction and universal quantification; therefore we may write $\forall(x : A).\, R(x)$ in place of $\prod_{(x:A)} R(x)$. Function types $A \to P$ are a special case of $\prod$, so we have implication too. On the other hand, the disjoint sums $P + Q$ and $\sum_{(x:A)} R(x)$ are usually not propositions. This reflects the fact that they are constructive

disjunction and existential quantification. In particular, from a proof of $\sum_{(x:A)} R(x)$ we can project a witness $x : A$ and a proof of $R(x)$.

For any type A, the *propositional truncation* of A is a type $\|A\|$ for which there is an introduction rule that wraps any $a : A$ into $|a| : \|A\|$, and a higher one that introduces an inhabitant of $|a| =_{\|A\|} |b|$ for any $a, b : A$, equating all inhabitants of $\|A\|$ and making $\|A\|$ a proposition; its recursion principle maps $\|A\|$ to a proposition P provided that $A \to P$. With truncation, we can obtain a classical version of existential quantification by defining $\exists(x : A).\, R(x) :\equiv \left\|\sum_{(x:A)} R(x)\right\|$, from which we can no longer project a witness x and a proof of $R(x)$, but can only derive another proposition P provided that $\left(\sum_{(x:A)} R(x)\right) \to P$. Following the HoTT convention, if an existential quantification in an informal statement is truncated, we will use the words 'mere' or 'merely' to make it clear.

Sets are types whose equality types are all propositions, so for any two inhabitants of a set, there is at most one way for them to be equal. It is easy to check that the types we work with in this paper are sets: types constructed from sets and the type formers $\times$, $+$, $\to$, $\prod$, and $\sum$ are all sets, and *Hedberg's theorem* is useful for proving that a base type A is a set—if A has decidable equality, that is, $\prod_{(x,y:A)}(x =_A y) + \neg(x =_A y)$, then A is a set (which we will write as $A : \mathsf{Set}$ for short).

2.2 λ-calculus, Gödel encoding, and the second recursion theorem

For λ-calculus we only fix notations. The details are left to, for example, the classic textbook by Barendregt [4]. Terms are defined informally by

$$M :\equiv \mathtt{x} \mid M\, N \mid \lambda\mathtt{x}.\, M$$

where variables $\mathtt{x}$'s are in the typewriter font. Λ denotes the type of terms and Λ_n the type of terms with at most n free variables. In particular, Λ_0 is the type of *closed terms*. Our formalisation uses the de Bruijn representation, so the α-equivalence $=_\alpha$ coincides with the equality type $=_\Lambda$ by construction. For the presentation in this paper, the variable with index i is written $\mathtt{x}_i$, and given $F : \Lambda_{n+1}$ we write $F[M]$ instead of $F[M/\mathtt{x}_0]$ for the substitution for the first free variable $\mathtt{x}_0$. The type $M \longrightarrow\!\!\!\!\!\rightarrow_\beta N$ of reductions from M to N consists of sequences of reduction rules such as $\beta : (\lambda\mathtt{x}.\, M)\, N \longrightarrow_\beta M[N/\mathtt{x}]$; as a special case, the type $M \longrightarrow\!\!\!\!\!\rightarrow_\beta M$ has exactly one inhabitant $\mathsf{refl}_{\twoheadrightarrow} M$, or just $\mathsf{refl}_{\twoheadrightarrow}$, which can be understood as either the empty sequence or (the proof of) the reflexivity of reduction. The types Λ and $M \longrightarrow\!\!\!\!\!\rightarrow_\beta N$ have decidable equality, so they are sets by Hedberg's theorem.

There is a function between λ-terms $\ulcorner\cdot\urcorner : \Lambda \to \Lambda_0$ such that $\ulcorner M \urcorner$ is normal and $M =_\alpha N$ whenever $\ulcorner M \urcorner =_\alpha \ulcorner N \urcorner$. Moreover, there are $\mathtt{ap}, \mathtt{subst} \in \Lambda_2$ and $\mathtt{quote}, \mathtt{eval} \in \Lambda_1$ satisfying

$$\mathtt{ap}[\ulcorner M \urcorner][\ulcorner N \urcorner] \longrightarrow\!\!\!\!\!\rightarrow_\beta \ulcorner M\, N \urcorner \qquad \mathtt{subst}[\ulcorner F \urcorner][\ulcorner N \urcorner] \longrightarrow\!\!\!\!\!\rightarrow_\beta \ulcorner F[N] \urcorner$$
$$\mathtt{quote}[\ulcorner M \urcorner] \longrightarrow\!\!\!\!\!\rightarrow_\beta \ulcorner\ulcorner M \urcorner\urcorner \qquad \mathtt{eval}[\ulcorner M \urcorner] \longrightarrow\!\!\!\!\!\rightarrow_\beta M.$$

This function $\ulcorner\cdot\urcorner$ is called a *Gödel encoding*. Traditionally, a quoted term $\ulcorner M \urcorner$ is called a Gödel *number* since the encoding $\ulcorner\cdot\urcorner$ assigns to every term M a Church numeral $\mathbf{c}_{\#M}$. An encoding needs not be a number at all, however, so we simply call $\ulcorner M \urcorner$ a *code* of M rather than a number. For details on the axiomatic characterisation of encoding, see Polonsky [24].

Note that the term $\mathtt{quote}$ can only compute the code of a term $\ulcorner M \urcorner$ which is already in quoted form. Indeed, no term can compute the code of any arbitrary closed term.

▶ **Proposition 2.1.** *There is no* $\mathtt{Q} : \Lambda_1$ *such that* $\mathtt{Q}[M] \longrightarrow\!\!\!\!\!\rightarrow_\beta \ulcorner M \urcorner$ *for all* $M : \Lambda_0$.

Contrary to the well-known first recursion theorem, Kleene's second recursion theorem works for *code* instead of values and will be used to model the **GL** modality.

▶ **Theorem 2.2** (SRT). *For every $F : \Lambda_n$ there exists $M : \Lambda_n$ such that $M \twoheadrightarrow_\beta F\ulcorner M\urcorner$.*

2.3 $\mathscr{P}$-Categories and exposures

Instead of ordinary categories, we work with $\mathscr{P}$-categories pioneered by Čubrić et al. [8], where morphisms are equipped with an additional partial equivalence relation (PER) as another level of equality between morphisms. Kavvos [15] recently advocated its use and introduced a construct called *exposure*, which is similar to a ($\mathscr{P}$-)functor but does not enforce the preservation of PERs of $\mathscr{P}$-categories, to manifest the essence of intensionality.

▶ **Definition 2.3.** A *partial equivalence relation* is a symmetric and transitive relation. A *$\mathscr{P}$-set* $(X, \sim_X)$ is a set X with a PER $\sim_X$. A *$\mathscr{P}$-function* from $(X, \sim_X)$ to $(Y, \sim_Y)$ is a function $f\colon X \to Y$ which respects the relation $\sim$ in the sense that $f\,x \sim_Y f\,y$ whenever $x \sim_X y$. An element $x \in X$ is *well-defined* (with respect to $\sim$) if $x \sim x$.

The identity function id_X is a $\mathscr{P}$-function, and the composite of $\mathscr{P}$-functions is also a $\mathscr{P}$-function. Then we recall the notion of $\mathscr{P}$-categories as follows.

▶ **Definition 2.4** ([8, Definition 2.4])**.** A $\mathscr{P}$-category $\mathbf{C}$ consists of a class of objects, a $\mathscr{P}$-set $(\mathbf{C}(X, Y), \sim)$ for each pair of objects X and Y, and an identity morphism $id_X\colon X \to X$ for each object X satisfying the associativity and identity laws up to $\sim$ in the sense that

- **(i)** $id_X \sim id_X$ always,
- **(ii)** $g \circ f \sim g' \circ f'$ whenever $g \sim g'$ and $f \sim f'$,
- **(iii)** $id \circ f \sim f'$ and $f \circ id \sim f'$ whenever $f \sim f'$,
- **(iv)** $h \circ (g \circ f) \sim (h' \circ g') \circ f'$ whenever $h \sim h'$, $g \sim g'$, and $f \sim f'$.

A $\mathscr{P}$-category has two kinds of equality for morphisms—the underlying equality $=$ and the PER $\sim$, where the former can be used to model the *intensional equality* and the latter the *extensional equality* akin to the structure of multiple judgemental equalities in the modal type theory by Pfenning [23]. Having two different equalities $=$ and $\sim$ reflects the fact that, for example, α-equivalent terms are β-equivalent but not vice versa. For categorical semantics, where terms are interpreted as morphisms, an interpretation into a $\mathscr{P}$-category is able to discriminate these two kinds of equality, enabling us to model intensionality. To emphasise the categorical notions up to the extensional equality $\sim$, the '$\mathscr{P}$-' prefix are added so that we have $\mathscr{P}$-functors, $\mathscr{P}$-initiality, etc.

We recall the notion of exposures, which are like $\mathscr{P}$-functors but are intended to 'expose' intensional differences at the extensional level: if an exposure is applied to intensionally different morphisms (which may or may not be identified extensionally), the resulting morphisms may be distinguished extensionally. Consequently, exposures are not required to preserve the extensional equality. Moreover, exposures are only supposed to refine the extensional equality and do not eliminate existing extensional differences, that is, exposures are faithful with respect to $\sim$. Put differently, intensionally equal morphisms, with respect to an exposure, should be extensionally equal. The precise definition is given as follows.

▶ **Definition 2.5.** Given $\mathscr{P}$-categories $\mathbf{C}$ and $\mathbf{D}$, an *exposure* $Q\colon \mathbf{C} \nrightarrow \mathbf{D}$ consists of *(a)* a mapping Q from objects X of $\mathbf{C}$ to objects QX of $\mathbf{D}$ and *(b)* from well-defined morphisms $f\colon X \to Y$ to well-defined morphisms $Qf\colon QX \to QY$ satisfying the following properties:

- **(i)** $Qid_X \sim id_{QX}$,

(ii) $Q(g \circ f) \sim Qg \circ Qf$, and
(iii) $f \sim g$ whenever $Qf \sim Qg$ for any two well-defined morphisms $f, g \colon X \to Y$.
The *identity exposure* $\mathcal{I}$ maps every object or morphism to itself. Composing two exposures in the usual way clearly gives us an exposure.

Similarly, the notion of natural transformations is introduced for exposures, sharing the same idea with ordinary natural transformations but only up to $\sim$.

▶ **Definition 2.6.** Given exposures $P, Q \colon \mathbf{C} \nrightarrow \mathbf{D}$, a *natural transformation of exposures* $t \colon P \to Q$ is a family of well-defined morphisms $t_X \colon PX \to QX$ such that $Qf \circ t_X \sim t_Y \circ Pf$ for every well-defined morphism $f \colon X \to Y$.

An *evaluator* for an endo-exposure Q is a natural transformation from Q to $\mathcal{I}$, modelling the **T** axiom $\boxtimes A \to A$. To model the **S4** modality, we may define *comonadic exposures* introduced by Kavvos [15] as an endo-exposure equipped with an evaluator and a natural transformation $\delta \colon Q \to Q^2$, modelling the **4** axiom $\boxtimes A \to \boxtimes\boxtimes A$, subject to comonad laws. In the presence of intensionality, however, we observe that the naturality is not always appropriate as discussed later in Remark 4.5.

3 $\mathscr{P}$-Category of assemblies on λ-calculus

Assemblies are used to accommodate the information of how extensions are *realised by* intensions. Accordingly an appropriate notion of morphisms between assemblies is introduced to form a $\mathscr{P}$-category, laying the technical foundation for Sections 4 and 5.

3.1 Assembly and trackable map

Traditionally, an assembly on natural numbers is a set $|X|$ with a realisability relation $\Vdash \subseteq \mathbb{N} \times |X|$ such that for every x in $|X|$ there exists some a with $a \Vdash x$, where a is said to *realise* x or a is a *realiser* of x. The modern notion of assemblies [30] is often defined on a partial combinatory algebra $(A, \cdot)$, called *PCA* for short, where $\cdot$ is a partial binary operation. For the sake of formalisation and potential applications in programming language design, we base our definition on λ-calculus subject to α-equivalence, which is more akin to the one based on an ordered PCA [14].

▶ **Definition 3.1.** An *assembly* X on λ-calculus consists of a *carrier* set $|X| : \mathsf{Set}$ and a family $\Vdash_X$ of sets indexed by Λ_0 and $|X|$ as its *realisability relation* such that *(a)* there is merely a realiser $M : \Lambda_0$ of every $x : |X|$, and *(b)* $M \Vdash_X x$ whenever $M \longrightarrow\!\!\!\!\!\rightarrow_\beta N$ and $N \Vdash_X x$. In other words, an assembly is a quadruple $(|X|, \Vdash_X, r_X, t_X)$ of type

$$\mathsf{Asm}_0 :\equiv \sum_{(|X| : \mathsf{Set})} \sum_{(\Vdash_X : \Lambda_0 \to |X| \to \mathsf{Set})} \mathsf{Respects}\,(\Vdash_X, \twoheadrightarrow_\beta) \times \mathsf{RightTotal}(\Vdash_X)$$

where

$$\mathsf{Respects}\,(\Vdash, \twoheadrightarrow_\beta) :\equiv \prod_{(M N : \Lambda_0)} \prod_{(x : |X|)} (M \twoheadrightarrow_\beta N) \to (N \Vdash x) \to (M \Vdash x) \tag{3.1}$$

$$\mathsf{RightTotal}(\Vdash) :\equiv \forall (x : |X|).\, \exists (M : \Lambda_0).\, M \Vdash x \tag{3.2}$$

Our type-theoretic formulation is almost a direct translation from the set-theoretic formulation except that the realisability relation $\Vdash$ is not really a relation but an indexed family of sets. As we would like to account for intensional equality in addition to extensional

equality between terms, computationally equivalent terms should not be identified *a priori*. It turns out that formulating the interaction with reduction $\longrightarrow\!\!\!\!\!\rightarrow_\beta$ as (3.1) in line with the definition on an ordered PCA suffices to derive familiar properties.

▶ **Example 3.2.** The type Λ_0 of closed terms with $\longrightarrow\!\!\!\!\!\rightarrow_\beta$ as its realisability relation is an assembly $(\Lambda_0, \longrightarrow\!\!\!\!\!\rightarrow_\beta, r_{\Lambda_0}, t_{\Lambda_0})$ where r_{Λ_0} and t_{Λ_0} are given by the transitivity and the reflexivity of $\longrightarrow\!\!\!\!\!\rightarrow_\beta$. That is, each term M is realised by those reducible to M.

Note that the assembly Λ_0 does *not* yet model code. Indeed, in such case, M should be realised by its *code* $\ulcorner M \urcorner$ instead. This is exactly the point of forthcoming sections.

▶ **Example 3.3.** Every natural number $n : \mathbb{N}$ is realised by terms reducible to its Church numeral c_n. That is, the type $\mathbb{N}$ of natural numbers with $M \Vdash_{\mathbb{N}} n$ whenever $M \longrightarrow\!\!\!\!\!\rightarrow_\beta \mathsf{c}_n$ is an assembly where $r_{\mathbb{N}}$ and $t_{\mathbb{N}}$ are given by the transitivity and the reflexivity of $\longrightarrow\!\!\!\!\!\rightarrow_\beta$.

A morphism between assemblies on a PCA $(A, \cdot)$ is defined as a function f merely *tracked by* some $b \in A$ in the sense that there merely exists some b such that $b \cdot a \Vdash f\,x$ whenever $a \Vdash x$. In this case, b is called the *tracker* of f. It is noted by Kavvos [15] that to bring out intensionality the tracker should be considered as part of the structure instead of a property.

▶ **Definition 3.4.** Given assemblies X and Y, a *trackable map* f from X to Y consists of a function $|f| : |X| \to |Y|$ and a term $F : \Lambda_1$ such that $F[M] \Vdash |f|\,x$ whenever $M \Vdash x$. That is, the type $\mathsf{Asm}_1(X, Y)$ of trackable maps is $\sum_{(|f| : |X| \to |Y|)} \sum_{(F:\Lambda_1)} \mathsf{Tracks}_{X,Y}(F, |f|)$ where

$$\mathsf{Tracks}_{X,Y}(F, |f|) :\equiv \prod_{(M:\Lambda_0)} \prod_{(x:|X|)} (M \Vdash_X x) \to (F[M] \Vdash_Y |f|\,x)\,.$$

A *merely trackable map* is an inhabitant of $\sum_{(|f| : |X| \to |Y|)} \exists (F : \Lambda_1).\, \mathsf{Tracks}_{X,Y}(F, |f|)$.

By definition, a trackable map $f \equiv (f, F, \mathfrak{f})$ consists of not only a function $|f|$ between carriers but also its tracker F and a transformation $\mathfrak{f}$ of realisability.

▶ **Example 3.5.** Every assembly X has an identity map $id_X :\equiv (\mathsf{id}_{|X|}, \mathtt{x}_0, \mathsf{pr}_3)$ where

$$\mathsf{pr}_3 :\equiv \lambda M.\, \lambda x.\, \lambda r.\, r : \prod_{(M:\Lambda_0)} \prod_{(x:|X|)} (M \Vdash_X x) \to (M \Vdash_X x)$$

since $\mathtt{x}_0[M]$ is judgementally equal to M.

Now we proceed with defining the composition of trackable maps. Let $f \colon X \to Y$ and $g \colon Y \to Z$ be trackable maps. Then, the term substitution $(G, F) \mapsto G[F]$ can be thought of as (intensional) function composition, since $G[F[M]] =_{\Lambda_0} G[F][M]$ holds for any term M. Given any $r : M \Vdash_X x$, the inhabitant $\mathfrak{g}(\mathfrak{f}\,r)$ has type $G[F[M]]$ and its transportation along a witness $p : G[F[M]] =_{\Lambda_0} G[F][M]$ has type $G[F][M]$, defining a function $\lambda M\, x\, r.\, \mathsf{transport}(p, \mathfrak{g}(\mathfrak{f}\,r))$. The above discussion amounts to defining a composition operation $(g, f) \mapsto g \circ f$.

3.2 Extensional equality and $\mathscr{P}$-category of assemblies

We define the partial equivalence relation $\sim$, referred to as the extensional equality, on trackable maps by $f_1 \sim f_2$ (f_1 is *extensionally equal* to f_2) if $|f_1| = |f_2|$.

▶ **Proposition 3.6.** *The type* Asm_0 *of assemblies and the family of types* $\mathsf{Asm}_1(X, Y)$ *for any two assemblies* X *and* Y *with the extensional equality form a* $\mathscr{P}$*-category* $\mathsf{Asm}(\Lambda)$.

We now investigate some of its basic properties.

▶ **Example 3.7** ($\mathscr{P}$-Terminal object)**.** The unit $\top :\equiv (\mathbf{1}, \Vdash_\top, r_\top, t_\top)$ is $\mathscr{P}$-terminal where
(i) $\mathbf{1}$ is the unit type,
(ii) $\Vdash_\top$ a relation defined by $M \Vdash_\top \star :\equiv M \longrightarrow\!\!\!\!\!\rightarrow_\beta \mathtt{I}$ where $\mathtt{I} :\equiv \lambda \mathtt{x}.\, \mathtt{x}$,
(iii) $r_\top : (M \longrightarrow\!\!\!\!\!\rightarrow_\beta N) \to (N \longrightarrow\!\!\!\!\!\rightarrow_\beta \mathtt{I}) \to (M \longrightarrow\!\!\!\!\!\rightarrow_\beta \mathtt{I})$ given by the transitivity of $\longrightarrow\!\!\!\!\!\rightarrow_\beta$,
(iv) and $t_\top$ the fact that the only inhabitant $\star : \mathbf{1}$ has a realiser $\mathtt{I}$ (by reflexivity).
The finality follows from function extensionality.

The construction of binary $\mathscr{P}$-products is also typical—the carrier of a product is the cartesian product and a pair (x, y) is realised by M if its Church-encoded projections realise x and y. It follows that $\mathsf{Asm}(\Lambda)$ has finite $\mathscr{P}$-products.

Every inhabitant of an assembly X corresponds to a merely trackable map to X from the terminal object $\top$, which are called *(global) elements* of X, and distinct merely trackable maps can be separated by elements of X. In $\mathsf{Asm}(\Lambda)$, as trackers are part of trackable maps, an element has to be constructed with an intension.

▶ **Lemma 3.8.** *Let X be an assembly. Then the following statements hold:*
1. *Every inhabitant $x : |X|$ corresponds to a merely trackable map from $\top$ to X.*
2. *Every pair of $x : |X|$ and $M : \Lambda_0$ with $r : M \Vdash_X x$ defines a* closed *element of X, i.e. a trackable map $(\lambda\_.\, x, M, \lambda\_.\, \lambda\_.\, \lambda\_.\, r)$ from $\top$ to X.*

As expected, the $\mathscr{P}$-terminal object $\top$ in $\mathsf{Asm}(\Lambda)$ is a $\mathscr{P}$-separator in the sense that for any two trackable maps f_1 and f_2 we have $f_1 \sim f_2$ if $f_1 \circ x \sim f_2 \circ x$ for every element x of X. Even further, we can restrict to closed elements.

▶ **Proposition 3.9.** *Two trackable maps $f_1, f_2 \colon X \to Y$ are extensionally equal if and only if $f_1 \circ x \sim f_2 \circ x$ for every closed element $x \colon \top \to X$. In particular, the $\mathscr{P}$-terminal object $\top$ is a $\mathscr{P}$-separator in $\mathsf{Asm}(\Lambda)$.*

Proof sketch. The proof from left to right is trivial. For the proof from right to left, let f_1 and f_2 be two trackable maps. By function extensionality, to prove that $|f_1| = |f_2|$, we define for any inhabitant $x : |X|$ a closed element $\hat{x}$ of X constructed by Lemma 3.8 using M_x and $r : M_x \Vdash x$ given by the right totality t_X. By the recursion principle of propositional truncation and $|Y|$ being a set, it follows from our assumption that there exists a path $|f_1|\, x = |f_2|\, x$ independent of the choice of M_x and $\mathfrak{M}_x$. ◀

▶ **Example 3.10** (Initial object)**.** The empty assembly $\bot$ is $\mathscr{P}$-initial consisting of the empty type $\mathbf{0}$ and a relation $\Vdash_\bot \colon \Lambda_0 \to \mathbf{0} \to \mathsf{Set}$ given by the elimination rule for the empty type. The other two components $r_\bot$ and $t_\bot$ are trivial.

In addition, one can show that $\bot$ is even a *strict* $\mathscr{P}$-initial object. That is,

▶ **Proposition 3.11.** *Any trackable map from some assembly X to $\bot$ is a $\mathscr{P}$-isomorphism.*

From the strictness of the initial object, no morphism from $\top$ to $\bot$ could exist.

The construction of $\mathscr{P}$-exponential $X \Rightarrow Y$ is a bit laborious and, perhaps surprisingly, $X \Rightarrow Y$ has the type of *merely* trackable maps as its carrier.

▶ **Example 3.12** ($\mathscr{P}$-Exponential)**.** Given assemblies X and Y, define

$$|X \Rightarrow Y| :\equiv \sum_{f : |X| \to |Y|} \exists (F : \Lambda_1).\, \mathsf{Tracks}_{X,Y}(F, f) \equiv \sum_{f : |X| \to |Y|} \left\| \sum_{F : \Lambda_1} \mathsf{Tracks}_{X,Y}(F, f) \right\|$$

with $L \Vdash_{X \Rightarrow Y} (f, \star) :\equiv \prod_{(M:\Lambda_0)} \prod_{(x:|X|)} (M \Vdash_X x) \to (L\,M \Vdash_Y f\,x)$.

It remains to construct $r_{X \Rightarrow Y}$ and $t_{X \Rightarrow Y}$: We know that $L' \longrightarrow\!\!\!\!\!\rightarrow_\beta L$ implies $L'\,M \longrightarrow\!\!\!\!\!\rightarrow_\beta L\,M$, so L' realises $(f, \star)$ whenever L realises $(f, \star)$ and $L' \longrightarrow\!\!\!\!\!\rightarrow_\beta L$ by r_Y. For every $(f, \star) : |X \Rightarrow Y|$, there merely exists a tracker of f, say F. We see that $L :\equiv \lambda \mathtt{x}.\, F$ realises $(f, \star)$, since $(\lambda \mathtt{x}.\, F)\ M \longrightarrow_\beta F[M]$ for any M and $F[M] \Vdash_Y f\,x$ whenever $M \Vdash_X x$. By applying the recursion principle of the truncated type $\left\| \sum_{(F:\Lambda_1)} \mathsf{Tracks}_{X,Y}(F, f) \right\|$ to the second component of $(f, \star)$, there merely exists a realiser of $(f, \star)$ for the right totality.

The *evaluation map* $(X \Rightarrow Y) \times X \xrightarrow{ev_{X,Y}} Y$ natural in X and Y consists of a function $((f, \star), x) \mapsto f\,x$ and its tracker $(\mathtt{proj}_1\,\mathtt{x}_0)\,(\mathtt{proj}_2\,\mathtt{x}_0) : \Lambda_1$ where $\mathtt{x}_0$ is the free variable (thought of as a pair of realisers for a function and its argument) and $\mathtt{proj}_i$ the projection function between λ-terms.

The *curried map* $(f^*, F^*, \mathfrak{f}^*)$ of a trackable function $(f, F, \mathfrak{f})$ from $Z \times X$ to Y consists of a function $f^* :\equiv \lambda z.\, ((\lambda x.\, f\,(z, x)), \star_z)$, where by the recursion principle on the mere existence of a realiser $L_z :\equiv t_Z z$ there is merely a tracker $F[\langle L_z, \mathtt{x}_0 \rangle]$ of $\lambda x.\, f\,(z, x)$, and a term $F^* :\equiv \lambda \mathtt{x}_0.\, F[\langle \mathtt{x}_1, \mathtt{x}_0 \rangle]$ with a witness $\mathfrak{f}^*$ of $\prod_{(L:\Lambda_0)} \prod_{(z:|Z|)} (L \Vdash_Z z) \to (F^*[L] \Vdash_{X \Rightarrow Y} f\,z)$ because of the reduction

$$(\lambda \mathtt{x}_0.\, F[\langle L, \mathtt{x}_0 \rangle])\ M \longrightarrow_\beta F[\langle L, M \rangle]$$

and that F is indeed a tracker of $f : |Z \times X| \to |Y|$. It is routine to verify remaining details.

▶ **Corollary 3.13.** $\mathsf{Asm}(\Lambda)$ *is a cartesian closed* $\mathscr{P}$*-category with a strict* $\mathscr{P}$*-initial object.*

4 Realisability semantics for the S4 modality

We are now ready to introduce an exposure $\boxtimes : \mathsf{Asm}(\Lambda) \nrightarrow \mathsf{Asm}(\Lambda)$ modelling the **S4** modality (validating the **K** axiom $\boxtimes(A \to B) \to \boxtimes A \to \boxtimes B$, the **4** axiom $A \to \boxtimes\boxtimes A$, and the **T** axiom $\boxtimes A \to A$) and show that a generic quoting $X \to \boxtimes X$ cannot exist.

4.1 An exposure for the S4 modality

Given an assembly X, which describes a set of extensions merely realised by some intensions (i.e. terms), we can expose the intensions at the level of extensions by constructing an assembly $\boxtimes X$ where an inhabitant $(M, x, r) : |\boxtimes X|$ is an extension $x : |X|$ and a term $M : \Lambda_0$ that realises x, witnessed by $r : M \Vdash_X x$. This term M becomes the main part of the extension (with respect to $\boxtimes X$) and should be (merely) realised by some intensional representation of M; a natural choice of such representation is $\ulcorner M \urcorner$, or indeed any term β-reducible to $\ulcorner M \urcorner$. In short, the carrier and the realisability relation of $\boxtimes X$ are defined as

$$|\boxtimes X| :\equiv \sum_{(M:\Lambda_0)} \sum_{(x:|X|)} M \Vdash_X x \quad \text{and} \quad (N \Vdash_{\boxtimes X} (M, x, r)) :\equiv N \longrightarrow\!\!\!\!\!\rightarrow_\beta \ulcorner M \urcorner$$

respectively. It turns out that $\boxtimes X :\equiv (|\boxtimes X|, \Vdash_{\boxtimes X}, r_{\boxtimes X}, t_{\boxtimes X})$ is indeed an assembly where $r_{\boxtimes X}$ and $t_{\boxtimes X}$ are the transitivity and the reflexivity of $\longrightarrow\!\!\!\!\!\rightarrow_\beta$.

To make $\boxtimes$ an exposure, we should also define the mapping on morphisms. Consider any trackable map f from X to Y and define $\boxtimes f :\equiv (|f|^{\boxtimes}, F^{\boxtimes}, \mathfrak{f}^{\boxtimes}) : \boxtimes X \to \boxtimes Y$ as follows. First define a function from $|\boxtimes X|$ to $|\boxtimes Y|$ by

$$|f|^{\boxtimes} : (M, x, r) \mapsto (F[M], |f|\,x, \mathfrak{f}\,M\,x\,r).$$

To give a tracker of $\boxtimes f$, recall that there is a term $\mathtt{subst}$ performing term substitution on codes (Section 2.2), and then the term $F^{\boxtimes} :\equiv \mathtt{subst}\ulcorner F\urcorner \mathtt{x}$ tracks $|f|^{\boxtimes}$ because

$$\mathtt{subst}\ulcorner F\urcorner N \longrightarrow\!\!\!\!\rightarrow_\beta \mathtt{subst}\ulcorner F\urcorner\ulcorner M\urcorner \longrightarrow\!\!\!\!\rightarrow_\beta \ulcorner F[M]\urcorner \Vdash_{\boxtimes Y} |f|^{\boxtimes}(M,x,r)$$

completing the definition of $\mathfrak{f}^{\boxtimes}$. In short, $\boxtimes f :\equiv (|f|^{\boxtimes}, F^{\boxtimes}, \mathfrak{f}^{\boxtimes})$ is a trackable map.

▶ **Definition 4.1.** By $\bigstar\colon \top \to \boxtimes\top$ we denote a closed element of $\boxtimes\top$ given by Lemma 3.8 with $(\mathtt{I}, \star, \mathsf{refl}_{\twoheadrightarrow}) : |\boxtimes\top|$ and its realiser $\ulcorner \mathtt{I}\urcorner$.

▶ Remark 4.2. Given elements a, b of X with $a \sim b$ but with different trackers, it follows that by definition $\boxtimes a \circ \bigstar \not\sim \boxtimes b \circ \bigstar$ are two extensionally different elements. That is, $\boxtimes$ does not preserve extensional equality.

The assembly $\boxtimes\top$ cannot be $\mathscr{P}$-isomorphic to $\top$, since there are countably many inhabitants of $|\boxtimes\top|$ while there is exactly one inhabitant of $|\top| \equiv \mathbf{1}$. Similarly, there are trackable maps from $\boxtimes(X \times Y)$ to $\boxtimes X \times \boxtimes Y$ and vice versa, but they are not $\mathscr{P}$-isomorphic. It follows that the exposure $\boxtimes$ does not preserve finite $\mathscr{P}$-products.

▶ **Theorem 4.3.** $\boxtimes\colon \mathsf{Asm}(\Lambda) \looparrowright \mathsf{Asm}(\Lambda)$ *is an exposure of assemblies. Moreover, there is an evaluator* ϵ *for* $\boxtimes$*, i.e. a natural transformation* ϵ *from* $\boxtimes$ *to* $\mathcal{I}$*.*

Proof sketch. It is routine to prove the preservation of identities and composition. For example, it follows by definition that $\mathsf{id}^{\boxtimes}_{|X|}(M,x,r) \equiv (\mathtt{x}[M], x, \mathsf{pr}_3\, M\, x\, r) \equiv (M,x,r)$.

Now we show that $\boxtimes$ reflects the extensional equality. Let f and g be trackable maps from X to Y. By assumptions that $\boxtimes f \sim \boxtimes g$ and that there is merely $M : \Lambda_0$ with $r : M \Vdash_X x$, we can apply the recursion principle of propositional truncation to derive

$$\boxtimes f(M,x,r) = (F[M], |f|\, x, \mathfrak{f}\, M\, x\, r) = (G[M], |g|\, x, \mathfrak{g}\, M\, x\, r) = \boxtimes g(M,x,r)$$

since the equality type on $\boxtimes Y$ is a proposition. Therefore, we have $\prod_{(x:|X|)} |f|\, x =_Y |g|\, x$. By function extensionality it then follows that $\big(|f| =_{|X|\to|Y|} |g|\big) \equiv f \sim g$.

As for the evaluator $\epsilon_X\colon \boxtimes X \to X$, recall the term $\mathtt{eval}$ which evaluates a code (Section 2.2). We simply define $|\epsilon_X|$ by $(M,x,r) \mapsto x$. Then, given $N : \Lambda_0$ with $N \longrightarrow\!\!\!\!\rightarrow_\beta \ulcorner M\urcorner$, we have $\mathtt{eval}[N] \longrightarrow\!\!\!\!\rightarrow_\beta \mathtt{eval}[\ulcorner M\urcorner] \longrightarrow\!\!\!\!\rightarrow_\beta M$ where $M \Vdash_X x$ is witnessed by r. That is, $|\epsilon_X|$ is tracked by $\mathtt{eval}$. The naturality of ϵ follows by definition. ◀

Given an element a of X, define its *quotation* as the element $\boxtimes a \circ \bigstar$ of $\boxtimes X$. The choice of $\bigstar$ does not matter if a is closed, since $\boxtimes a \circ \bigstar' \sim \boxtimes a \circ \bigstar$ for any element $\bigstar'$ of $\boxtimes\top$. We say that a trackable map $q\colon X \to \boxtimes X$ *quotes* an element a of X whenever $q \circ a \sim \boxtimes a \circ \bigstar$. The (**4**) axiom can be realised by a family of trackable maps which quote closed elements:

▶ **Proposition 4.4.** *There is a family of functions* $|\delta_X|(M,x,r) :\equiv (\ulcorner M\urcorner, (M,x,r), \mathsf{refl}_{\twoheadrightarrow})$ *indexed by objects* X *from* $|\boxtimes X|$ *to* $|\boxtimes\boxtimes X|$ *and tracked by* $\mathtt{quote}$*, which quote closed elements of* $\boxtimes X$*.*

The fact that δ_X quotes closed elements justifies the computational meaning categorically. Yet, δ_X may fail to quote an element a if a is not closed, since in general the intension part of $\boxtimes a$ is applied only verbatim. That is, $|\boxtimes a|(\mathtt{I}, \star, \mathsf{refl}_{\twoheadrightarrow})$ is $(F[\mathtt{I}], (M,x,r), s)$ where $F[\mathtt{I}]$ is not necessarily α-equivalent to $\ulcorner M\urcorner$. The subtlety goes on:

▶ Remark 4.5. One may expect that $(\boxtimes, \epsilon, \delta)$ is comonadic in the sense that δ is a natural transformation up to $\sim$ satisfying comonad laws, but these maps δ_X are *not* natural in X. In detail, for each trackable map $f\colon X \to Y$ the inhabitant

$$\delta_Y(\boxtimes f(M,x,r)) \equiv (\ulcorner F[M]\urcorner, (F[M], f\, x, \mathfrak{f}\, M\, x\, r), \mathsf{refl}_{\twoheadrightarrow}) : \boxtimes\boxtimes Y$$

is not equal to

$$\boxtimes\boxtimes f(\delta_X(M,x,r)) \equiv (\mathsf{subst}\ulcorner F\urcorner\ulcorner M\urcorner, (F[M], f\,x, \mathfrak{f}\,M\,x\,r), \mathsf{subst}_{\twoheadrightarrow}) : \boxtimes\boxtimes Y$$

despite that their extensions are the same, where $\mathsf{subst}_{\twoheadrightarrow}$ is the witness of the reduction sequence $\mathsf{subst}\ulcorner F\urcorner\ulcorner M\urcorner \longrightarrow\!\!\!\!\!\rightarrow_\beta \ulcorner F[M]\urcorner$. Let us define $(M,x,r) \leq (N,y,s)$ if $M \longrightarrow\!\!\!\!\!\rightarrow_\beta N$ and $x = y$ and $f \leq g$ if $|f|\,x \leq |g|\,x$ for all x. Then we only have $|\boxtimes\boxtimes f \circ \delta_X| \leq |\delta_Y \circ \boxtimes f|$. In general, the *lax naturality* appears more appropriate in the presence of intensionality.

The normality condition is realised exactly by $\mathtt{ap}$ without naturality:

▶ **Proposition 4.6.** *There is a family of trackable maps from* $\boxtimes(X \Rightarrow Y)$ *to* $\boxtimes X \Rightarrow \boxtimes Y$ *tracked by* $\lambda \mathtt{x}_0.\, \mathtt{ap}[\mathtt{x}_1]\ \mathtt{x}_0$.

On the other hand, it is impossible for the rule $A \to \boxtimes A$ to compute quotations for arbitrary A, since this is already impossible for the particular case of $\Lambda_0 \to \boxtimes\Lambda_0$ (where Λ_0 was given in Example 3.2).

▶ **Theorem 4.7.** *No trackable map from* Λ_0 *to* $\boxtimes\Lambda_0$ *quotes closed elements of* Λ_0.

Proof. Assume $\eta\colon \Lambda_0 \to \boxtimes\Lambda_0$ with $\eta \circ a \sim \boxtimes a \circ \bigstar$ for any $a\colon \top \to \Lambda_0$ given by Lemma 3.8. Every closed term M defines an element $\widehat{M} :\equiv (\lambda\_.\, M, M, \lambda\_.\, \lambda\_.\, \lambda\_.\, \mathsf{refl}_{\twoheadrightarrow})$ of Λ_0 and thus $\left|\boxtimes\widehat{M}\right|(N,y,s) = (M, M, \mathsf{refl}_{\twoheadrightarrow})$ for any $(N,y,s) : \boxtimes\top$ by definition. By assumption

$$|\eta|\,M \equiv |\eta|\left(\left|\widehat{M}\right|\star\right) = \left|\boxtimes\widehat{M}\right|(|\bigstar|\,\star) = (M, M, \mathsf{refl}_{\twoheadrightarrow}),$$

so the tracker $\mathtt{Q}$ of η should satisfy $\mathtt{Q}[N] \longrightarrow\!\!\!\!\!\rightarrow_\beta \ulcorner M\urcorner$ whenever $N \longrightarrow\!\!\!\!\!\rightarrow_\beta M$. In particular, it follows that $\mathtt{Q}[M] \longrightarrow\!\!\!\!\!\rightarrow_\beta \ulcorner M\urcorner$. By Proposition 2.1 such $\mathtt{Q}$ cannot exist. ◀

As the choice of $\bigstar$ does not matter for closed elements a, the above theorem shows that even a very limited form of naturality for any two morphisms η_{Λ_0} and $\eta_\top$ satisfying $\eta_{\Lambda_0} \circ a \sim \boxtimes a \circ \eta_\top$ for any closed element a remains impossible. It is unclear how to state 'parametricity' for $\boxtimes$ so that any family of morphisms from A to $\boxtimes A$, satisfying a reasonable naturality, can be rejected.

5 Realisability semantics for the GL modality

Kavvos [15] advocated that the provability modality $\Box$ and the **GL** axiom $\Box(\Box A \to A) \to \Box A$ can also be understood as the type of code of type A and as intensional recursion respectively from the computational perspective. Since we already have an exposure $\boxtimes$ modelling typed code, a natural approach is to extend $\boxtimes$ to model the **GL** axiom in addition to **S4**. However, it is known that the **GL** axiom is incompatible with the reflection principle $\Box A \to A$. Indeed, let A be the falsity $\bot$ for both laws. Then, we have $\Box(\Box\bot \to \bot) \to \Box\bot$ and $\Box\bot \to \bot$. By the necessitation rule we can derive $\Box(\Box\bot \to \bot)$ and thus by modus ponens we can derive $\Box\bot$ and finally $\bot$. Therefore, by Proposition 3.11 and Theorem 4.3, we cannot expect the exposure $\boxtimes$ to model the **GL** axiom if we want the type system to be logically consistent.

To untie the knot and retain consistency and the understanding of $\Box A$ as code of type A, we observe that in general $\Box A$ and $\boxtimes A$ are types for different kinds of code and should be kept separate: code constructed with intensional recursion can only be expanded in stages (or otherwise may result in non-termination), whereas code supporting **S4** is only a special case where next stages include the current one, so that $\epsilon_X\colon \boxtimes X \to X$ is allowed. Therefore, separately from $\boxtimes$, we give an exposure $\Box\colon \mathsf{Asm}(\Lambda) \nrightarrow \mathsf{Asm}(\Lambda)$ modelling the **GL** modality

in a staged setting (Section 5.2). The construction refutes both $X \to \Box X$ (by the same argument for $\boxtimes$) and the reflection principle $\Box X \to X$. We also derive the **GL** axiom as well as its deductive form. To express staged constructions more conveniently (without stage indexing), we work within guarded type theory.

5.1 Digression: Clocked cubical type theory

We use a particular version of guarded type theory—*clocked cubical type theory* [18], CCTT for short. It extends HoTT with a later modality $\triangleright^\kappa$, parametrised by a 'clock' κ, and guarded recursion. Here we only intuitively introduce the constructs and properties of CCTT that are necessary for the informal presentation of the constructions in Section 5.2, but the formal details are all checked in Agda in the guarded cubical mode.

CCTT features a new type $\triangleright(\alpha : \kappa).\, A$ of suspended computations that take in a tick α on a clock κ to produce an inhabitant of A in a next stage. (Clocks will be discussed towards the end and can be ignored for now.) An inhabitant of $\triangleright(\alpha : \kappa).\, A$ therefore resembles a function computationally, and can be introduced as a λ-expression $\lambda(\alpha : \kappa).\, t$, often abbreviated to $\lambda\alpha.\, t$, where $t : A$, or eliminated by application to a tick, denoted by $f[\alpha] : A$ where $f : \triangleright(\alpha : \kappa).\, A$ and $\alpha : \kappa$. For brevity, $\triangleright(\alpha : \kappa).\, A$ is written as $\triangleright^\kappa A$ if α is not referred to in A. For example, the following term implements the normality axiom for $\triangleright^\kappa$:

$$\mathsf{ap}^\kappa :\equiv \lambda f.\, \lambda x.\, \lambda\alpha.\, f[\alpha]\,(x[\alpha]) : \triangleright^\kappa(A \to B) \to \triangleright^\kappa A \to \triangleright^\kappa B.$$

Viewed as a function, ap^κ takes $f : \triangleright^\kappa(A \to B)$ and $x : \triangleright^\kappa A$ as arguments, both of which are values that can be used in a next stage, and should produce a result of type $\triangleright^\kappa B$, that is, a value of type B in a next stage; this result is constructed by first taking in a tick α—after which the rest of the term describes a construction in the next stage—and then applying both f and x to α to produce $f[\alpha] : A \to B$, $x[\alpha] : A$, and eventually $f[\alpha]\,(x[\alpha]) : B$ in the next stage. Another important example is delaying a value to a next stage:

$$\mathsf{next}^\kappa :\equiv \lambda x.\, \lambda\alpha.\, x : A \to \triangleright^\kappa A.$$

In contrast to next^κ, there is no term of type $\triangleright^\kappa A \to A$, matching our intuition about a series of stages happening in order: in the current stage we should not be able to obtain a value that is only available in the next stage. Also there is no term of type $\triangleright^\kappa\triangleright^\kappa A \to \triangleright^\kappa A$—it might be tempting to write $\lambda x.\, \lambda\alpha.\, x[\alpha][\alpha]$, but the two consecutive applications to α are prohibited in CCTT.

An important primitive is *guarded recursion*, also known as *Löb induction*: every function $f : \triangleright^\kappa A \to A$ has a delayed fixed point $\mathsf{dfix}^\kappa\, f : \triangleright^\kappa A$ with the fixed point equation $(\mathsf{dfix}^\kappa\, f)[\alpha] =_A f\,(\mathsf{dfix}^\kappa\, f)$ where the right-hand side can be seen as a fixed point of f without delay.

We now list some properties needed for Section 5.2. The first one helps to assure that we are still working with propositions and sets even when $\triangleright^\kappa$ is involved.

▶ **Lemma 5.1.** *If $A[\alpha]$ is a proposition/set for arbitrary $\alpha : \kappa$, then so is $\triangleright(\alpha : \kappa).\, A[\alpha]$.*

The later modality distributes over a Σ-type.

▶ **Lemma 5.2.** *Let $B(x)$ be a family of types indexed by $x : A$. Then there are functions from $\triangleright^\kappa \sum_{(x:A)} B(x)$ to $\sum_{(x:\triangleright^\kappa A)} \triangleright(\alpha : \kappa).\, B(x[\alpha])$ and vice versa. In fact, the two types are equivalent.*

Therefore guarded recursion has a specialised form for Σ-types.

▶ **Corollary 5.3.** *Let $B(x)$ be a family of types indexed by $x : A$. Then*

$$\sum_{x:\triangleright A} \triangleright(\alpha : \kappa).\, B(x[\alpha]) \to \sum_{x:A} B(x) \quad \textit{implies} \quad \sum_{x:A} B(x).$$

The final property states that if two values delayed to a next stage are equal, then they are equal in the current stage. It may be tempting to formulate the property as $\mathsf{next}^\kappa\, x =_{\triangleright^\kappa A} \mathsf{next}^\kappa\, y \to x =_A y$, but this is in fact invalid, since (analogously to function extensionality) the antecedent equality is equivalent to $\triangleright^\kappa(x =_A y)$ rather than $x =_A y$ [20]. The correct formulation is the following, where the antecedent includes a *clock quantification* '$\forall\kappa$'.

▶ **Lemma 5.4.** *Let x and $y : A$. Then $x =_A y$ if $\forall\kappa.\, \mathsf{next}^\kappa\, x =_{\triangleright^\kappa A} \mathsf{next}^\kappa\, y$.*

This lemma holds because, with clock quantification, it is possible to write Atkey and McBride's [3] operator $\mathsf{force} : (\forall\kappa.\triangleright^\kappa A) \to (\forall\kappa.\, A)$, which can then be applied to the equality $\forall\kappa.\triangleright^\kappa(x =_A y)$ equivalent to the antecedent and yield $\forall\kappa.\, x =_A y$, which is equivalent to $x =_A y$. One way to think about (fully) clock-quantified types is that they are independent of the choice of clocks and can be viewed as the types of pure, completed descriptions of staged computation rather than ongoing computations that are taking effect in stages with respect to a particular clock in scope. We can manipulate such descriptions at will, irrespective of our current timeline—in particular, it is perfectly fine to take a description of a staged computation that produces results from the second stage onwards and make it produce the results right from the first stage instead, which is what force does.

5.2 An exposure for the GL modality

First we adapt the definition of exposures to the setting of CCTT.

▶ **Definition 5.5** (Clocked exposure)**.** Given $\mathscr{P}$-categories $\mathbf{C}$ and $\mathbf{D}$, a *clocked exposure* $Q\colon \mathbf{C} \nrightarrow \mathbf{D}$ consists of *(a)* a mapping Q^κ for each clock κ from objects X of $\mathbf{C}$ to objects $Q^\kappa X$ of $\mathbf{D}$ and *(b)* for each clock κ from well-defined morphisms $f\colon X \to Y$ to well-defined morphisms $Q^\kappa f\colon Q^\kappa X \to Q^\kappa Y$ satisfying following properties

(i) $Q^\kappa id_X \sim id_{QX}$,
(ii) $Q^\kappa(g \circ f) \sim Q^\kappa g \circ Q^\kappa f$, and
(iii) $f \sim g$ whenever $\forall\kappa.\, Q^\kappa f \sim Q^\kappa g$ for any two well-defined morphisms $f, g\colon X \to Y$.

Notably, the faithfulness of a clocked exposure mirrors the form of Lemma 5.4, and is the main reason that we need CCTT.

Now we introduce the clocked exposure $\Box\colon \mathsf{Asm}(\Lambda) \to \mathsf{Asm}(\Lambda)$ modelling **GL**. For an assembly X, the carrier $|\Box^\kappa X|$ and the realisability relation $\Vdash_{\Box^\kappa X}$ are defined as

$$|\Box^\kappa X| :\equiv \sum_{(M:\Lambda_0)} \sum_{(x:\triangleright^\kappa |X|)} \triangleright(\alpha : \kappa).\, M \Vdash_X x[\alpha] \quad \text{and} \quad (N \Vdash_{\Box^\kappa X} (M, x, r)) :\equiv N \longrightarrow\!\!\!\!\!\rightarrow_\beta \ulcorner M \urcorner$$

where $\Vdash_{\Box^\kappa X}$ is defined in the same way as the exposure $\boxtimes$. The main difference between the carriers $|\boxtimes X|$ and $|\Box^\kappa X|$ is that the extension part $|X|$ becomes $\triangleright^\kappa|X|$. That is, the extension x is available in a next stage but not earlier (with respect to the clock κ), but the intension M remains the same type. Similarly, $\Box^\kappa X :\equiv (|\Box^\kappa X|, \Vdash_{\Box^\kappa X}, r_{\Box^\kappa X}, t_{\Box^\kappa X})$ is an assembly where $r_{\Box^\kappa X}$ and $t_{\Box^\kappa X}$ are given by the transitivity and the reflexivity of $\longrightarrow\!\!\!\!\!\rightarrow_\beta$.

For any trackable map f from X to Y, also define $\Box^\kappa f$ in the same way as $\boxtimes f$ except that a later modality is involved:

$$|\Box^\kappa f|(M, x, r) :\equiv (F[M], \triangleright^\kappa|f|\, x, \lambda\alpha.\, \mathfrak{f}\, M\, (x[\alpha])\, (r[\alpha]))$$

where $\triangleright^\kappa|f| : \triangleright^\kappa|X| \to \triangleright^\kappa|Y|$ is given by the functoriality of the later modality $\triangleright^\kappa$. The very same argument for $\boxtimes f$ shows that $\Box^\kappa f$ is indeed a trackable map from $|\Box^\kappa X|$ to $|\Box^\kappa Y|$.

▶ Remark. By Lemma 5.2, the carrier of $\Box^\kappa X$ and the type $\sum_{(M:\Lambda_0)} \triangleright^\kappa \sum_{(x:|X|)} M \Vdash_X x$ are interchangeable. The latter form is more convenient when using guarded recursion.

It is straightforward to show that $\Box$ is a clocked exposure by Lemma 5.4.

▶ **Theorem 5.6.** $\Box\colon \mathsf{Asm}(\Lambda) \looparrowright \mathsf{Asm}(\Lambda)$ *is a clocked exposure.*

▶ **Proposition 5.7.** *There is a family of trackable maps from* $\Box^\kappa(X \Rightarrow Y)$ *to* $\Box^\kappa X \Rightarrow \Box^\kappa Y$ *tracked by* $\lambda \mathtt{x}_0.\, \mathtt{ap}[\mathtt{x}_1]\, \mathtt{x}_0$.

Similar to Theorem 4.7, no morphism $\Lambda_0 \to \Box^\kappa \Lambda_0$ can be quoting.

▶ **Theorem 5.8.** *There is no trackable map from* Λ_0 *to* $\Box^\kappa \Lambda_0$ *which quotes closed elements.*

It is also not possible to have a family of trackable maps ϵ_X from $\Box^\kappa X$ to X natural in X, since the extension of (M, x, r) can only be projected in a time step away from now.

▶ **Theorem 5.9.** *There is no function from* $|\Box^\kappa \bot|$ *to* $|\bot|$. *In particular, there is no natural transformation from* $\Box^\kappa$ *to* $\mathcal{I}$ *for any* κ.

Proof. Assume $\epsilon_\bot : |\Box^\kappa \bot| \to |\bot|$ exists. We show that there is $\mathsf{bang} : \triangleright\mathbf{0} \to \mathbf{0}$, so by guarded recursion a contradiction $\mathsf{fix}\,\mathsf{bang} : \mathbf{0}$ is derivable. Let x be an inhabitant of $\triangleright\mathbf{0}$. We construct an inhabitant (M, x, r) of $\Box^\kappa \bot$ so that the function $|\epsilon_\bot|$ from $|\Box^\kappa \bot|$ to $|\bot| \equiv \mathbf{0}$ can be applied. Choose an arbitrary closed term M, say $\lambda \mathtt{x}.\, \mathtt{x}$, and apply the recursion principle $\mathsf{rec}_\mathbf{0}$ of the empty type to x in a time step to get $r :\equiv \lambda\alpha.\, \mathsf{rec}_\mathbf{0}\,(M \Vdash_\bot x[\alpha])\, x[\alpha]$, which is an inhabitant of $\triangleright(\alpha : \kappa).\, M \Vdash_\bot x[\alpha]$, so (M, x, r) is of type $|\Box^\kappa \bot|$. ◀

The pay-off for disallowing evaluation is to be able to derive intensional recursion, which is logically the **GL** axiom and its deductive form(s).

▶ **Theorem 5.10** (Intensional recursion)**.** *For every trackable map* $f : \Box^\kappa X \to X$, *there are*
1. *an element* $f^\dagger$ *of* $\Box^\kappa X$ *realised by* $\ulcorner \mathsf{fix}\, F \urcorner$ *and*
2. *an element* $f^\ddagger$ *of* X *realised by* $\mathsf{fix}\, F$ *satisfying* $f^\ddagger \sim f \circ \Box^\kappa f^\ddagger \circ \bigstar$

where $\mathsf{fix}\, F : \Lambda_0$ *is a term that can be reduced to* $F[\ulcorner \mathsf{fix}\, F \urcorner]$.

Proof. Let f be a trackable map from $\Box^\kappa X$ to X tracked by $F : \Lambda_1$. Applying Theorem 2.2 to $\lambda \mathtt{x}_0.\, F : \Lambda_0$, we obtain a term $\mathsf{fix}\, F : \Lambda_0$ with $\mathsf{fix}\, F \longrightarrow\!\!\!\!\rightarrow_\beta (\lambda \mathtt{x}_0.\, F) \ulcorner \mathsf{fix}\, F \urcorner \longrightarrow_\beta F[\ulcorner \mathsf{fix}\, F \urcorner]$. Now we construct the first element by Löb induction on a Σ-type (Corollary 5.3): assuming

$$x : \triangleright|X| \quad \text{and} \quad r : \triangleright(\alpha : \kappa).\, F[\ulcorner \mathsf{fix}\, F \urcorner] \Vdash x[\alpha]$$

we show an inhabitant of type $|X|$ realised by $\ulcorner \mathsf{fix}\, F \urcorner$ as follows.
1. First, $\triangleright(\alpha : \kappa).\, (\mathsf{fix}\, F \Vdash x[\alpha])$ has an inhabitant, say r', since $\mathsf{fix}\, F$ reduces to $F[\ulcorner \mathsf{fix}\, F \urcorner]$.
2. Then, we derive an inhabitant of type $|X|$ realised by $F[\ulcorner \mathsf{fix}\, F \urcorner]$ as witnessed by

   $$\mathfrak{f}(\mathsf{refl}_{\to\!\!\!\!\rightarrow} \ulcorner \mathsf{fix}\, F \urcorner) : F[\ulcorner \mathsf{fix}\, F \urcorner] \Vdash |f|(\mathsf{fix}\, F, x, r')$$

   since F tracks $|f|$ and the set $\ulcorner \mathsf{fix}\, F \urcorner \Vdash_{\Box^\kappa X} (\mathsf{fix}\, F, x, r')$ is judgementally equal to $\ulcorner \mathsf{fix}\, F \urcorner \longrightarrow\!\!\!\!\rightarrow_\beta \ulcorner \mathsf{fix}\, F \urcorner$, which is inhabited by $\mathsf{refl}_{\to\!\!\!\!\rightarrow}$.
3. By Löb induction, $\sum_{(x:\triangleright|X|)} \triangleright(\alpha : \kappa).\, F[\ulcorner \mathsf{fix}\, F \urcorner] \Vdash x[\alpha]$ has an inhabitant (x_0, r_0).
4. By $\mathsf{fix}\, F \longrightarrow\!\!\!\!\rightarrow_\beta F[\ulcorner \mathsf{fix}\, F \urcorner]$, we have $(\mathsf{fix}\, F, x_0, r_0') : |\Box^\kappa X|$ where $r_0' : \triangleright(\alpha : \kappa).\, \mathsf{fix}\, F \Vdash x_0[\alpha]$.

Clearly $(\mathsf{fix}\, F, x_0, r_0')$ is realised by $\ulcorner \mathsf{fix}\, F \urcorner$, and by Lemma 3.8 it gives an element $f^\dagger$ of $\Box X$.

To construct the second element $f^\ddagger$, we follow the same steps except the third: we conclude $(x_0, r_0) : \sum_{(x:|X|)} F[\ulcorner \mathsf{fix}\, F \urcorner] \Vdash x[\alpha]$ without any delay and thus $x_0 : |X|$ is realised by $r_0' : \mathsf{fix}\, F \Vdash x_0$. The equation $f^\ddagger \sim f \circ \Box^\kappa(f^\ddagger) \circ \bigstar$ follows from the fixed point equation for guarded recursion. ◀

From the above proof, we can see that the intensional information available in the trackable map $\Box^\kappa X \to X$, i.e. the tracker F, does matter, since it is essential for constructing the fixpoint $\mathsf{fix}\, F$. To internalise the inductive form as the **GL** axiom, we also need the intension:

▶ **Theorem 5.11.** *There is a family of trackable maps from* $\Box^\kappa(\Box^\kappa X \Rightarrow X)$ *to* $\Box^\kappa X$.

The reader may wonder whether the strong Löb axiom, interpreted as a map from $\Box^\kappa X \Rightarrow X$ to X, can also be realised by the SRT in a similar way, but from $\Box^\kappa X \Rightarrow X$, which amounts to a merely trackable map, we do not get the tracker F explicitly needed by the SRT.

6 Related work

Kavvos introduced a comonadic exposure [15, Theorem 11], which we denote by $\boxtimes_K$ here, on $\mathscr{P}$-category of assemblies on a PCA (instead of λ-calculus) to model the intensional **S4** modality. For an assembly X on a PCA $(A, \cdot)$, the assembly $\boxtimes_K X$ is defined by

$$|\boxtimes_K X| :\equiv \{\, (a, x) \mid a \Vdash x \,\} \quad \text{and} \quad b \Vdash_{\boxtimes_K X} (a, x) :\equiv a = b,$$

without the use of Gödel encoding. The morphism mapping is similar to $\boxtimes$. The difference between $\boxtimes_K$ and $\boxtimes$ mainly comes from the chosen notion of realisability and the use of Gödel encoding. First, the exposure $\boxtimes_K$ preserves finite products, while $\boxtimes$ does not. Some β-equivalent intensions have to be identified to satisfy equations of PCA. For example $\star : |\top|$ has only one realiser $\mathtt{I}$, so $|\boxtimes_K \top|$ has only one element $(\star, \mathtt{I})$, too. Second, $\boxtimes_K$ is comonadic by a similar reason, while our δ is not even natural. Third, $\boxtimes_K$ is idempotent, i.e. δ_K are isomorphisms, which is impossible for $\boxtimes$ because of Gödel encoding. Finally, a generic quoting for $\boxtimes_K$ *can* be defined:

▶ **Observation 6.1.** *For the one-element PCA, there exists a natural transformation from the identity exposure to* $\boxtimes_K$.

Further, assuming the axiom of choice, there is (merely) a function q from $|X|$ to $|\boxtimes_K X|$ because of the right totality of $\Vdash$. It is likely that q could be realised in the sense of Krivine's classical realisability [19], which would establish a non-trivial example of generic quoting for $\boxtimes_K$ or alike. On the other hand, using λ-terms subject to α-equivalence as realisers and $\ulcorner M \urcorner$ as realisers for (M, x, r) allows the exposure $\boxtimes$ to distinguish β-equivalent intensions, so $\boxtimes$—being more intensional than $\boxtimes_K$—lacks well-behaved extensional properties, as expected.

As a reviewer pointed out, Remark 4.5 is reminiscent of *categorical simulation* studied by Cockett and Hofstra [6].

Artemov and Beklemishev [2] pointed out that Gödel attempted to use classical **S4** modal logic to capture provability for Peano Arithmetic (**PA**) but realised that $\mathsf{Prov}(A) :\equiv \exists x.\, \mathsf{Proof}(x, A)$ cannot be **S4**. Löb found the well-known **GL** axiom and Solovay showed that **GL** is complete with respect to **PA**. On the other hand, Artemov [1] proposed *logic of proofs* extending **S4** and argued that **S4** is for *explicit proofs*. Goris [13] discussed two modalities **GL** and **S4** over classical logic by presenting a bi-modal logic and provides a Kripke semantics for both. We took Goris' notation for the **S4** modality $\boxtimes$. Our work is partly inspired by Shamkanov's [25, 26] provability semantics for **GL** using circular proofs.

7 Conclusion

In this paper we follow the principle of modality-as-intension [10] and the $\mathscr{P}$-categorical semantics [15] to manifest the concepts of denotations, extensions, and intensions. We have studied the $\mathscr{P}$-category of assemblies on λ-calculus developed and formalised in HoTT as a semantic foundation for intensionality, on which we modelled **S4** and **GL** modalities. Notably, our denotational semantics of the **S4** modality $\boxtimes$ is more intensional than that of Kavvos' $\boxtimes_K$. For the **GL** modality $\Box$, we have given the *first* denotational semantics, which is shown (Theorem 5.10 and Theorem 5.11) to satisfy the Gödel-Löb axiom and its deductive form—the intensional recursion—using guarded type theory.

As future work, an important issue we have not discussed yet is the connection between $\boxtimes$ and $\Box$—for example, it is easy to construct a family of trackable maps from $\boxtimes X$ to $\Box^\kappa X$ natural in X by deferring the extension part of (M, x) to a later stage, and Goris' bi-modal logic [13] suggests that there are more rules to discover. In the long term, based on the $\mathscr{P}$-category of assemblies and the denotational semantics of modalities, we intend to design a type theory with two modes of `Code` and prove its meta-properties such as consistency, confluence, and decidability of type checking by interpreting judgements into $\mathsf{Asm}(\Lambda)$.

References

1 Sergei N. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, 2001. doi:10.2307/2687821.

2 Sergei N. Artemov and Lev D. Beklemishev. Provability logic. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd Edition*, volume 13 of *Handbook of Philosophical Logic*, pages 189–360. Springer, Dordrecht, 2005. doi:10.1007/1-4020-3521-7_3.

3 Robert Atkey and Conor McBride. Productive coprogramming with guarded recursion. In *Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming (ICFP)*, page 197, New York, New York, USA, 2013. ACM Press. doi:10.1145/2500365.2500597.

4 Henk Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic*. North Holland, 1984.

5 George S. Boolos. *The Logic of Provability*. Cambridge University Press, 1994. doi:10.1017/cbo9780511625183.

6 J.R.B. Cockett and Pieter J.W. Hofstra. Categorical simulations. *Journal of Pure and Applied Algebra*, 214(10):1835–1853, 2010. doi:10.1016/j.jpaa.2009.12.028.

7 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: A constructive interpretation of the univalence axiom. In Tarmo Uustalu, editor, *21st International Conference on Types for Proofs and Programs (TYPES)*, volume 69 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–34, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TYPES.2015.5.

8 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.

9 Rowan Davies. A temporal logic approach to binding-time analysis. *Journal of the ACM*, 64(1):1–45, 2017. doi:10.1145/3011069.

10 Rowan Davies and Frank Pfenning. A modal analysis of staged computation. *Journal of the ACM*, 48(3):555–604, 2001. doi:10.1145/382780.382785.

11 Agda development team. Agda 2.6.2 documentation. Accessed: 2021-06-20. URL: https://agda.readthedocs.io/en/v2.6.2/.

12 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, 2013. doi:10.1016/j.jal.2012.07.002.

13 Evan Goris. A modal provability logic of explicit and implicit proofs. *Annals of Pure and Applied Logic*, 161(3):388–403, 2009. doi:10.1016/j.apal.2009.07.020.

14 Pieter Hofstra and Jaap van Oosten. Ordered partial combinatory algebras. *Mathematical Proceedings of the Cambridge Philosophical Society*, 134(3):445–463, 2003. doi:10.1017/S0305004102006424.

15 G. A. Kavvos. On the semantics of intensionality. In Javier Esparza and Andrzej S. Murawski, editors, *Proceedings of the 20th International Conference the Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 10203 of *Lecture Notes in Computer Science*, pages 550–566. Springer, Berlin, Heidelberg, 2017. doi:10.1007/978-3-662-54458-7_32.

16 G. A. Kavvos. Intensionality, intensional recursion, and the Gödel-Löb axiom. *Journal of Applied Logics - The IfCoLog Journal of Logics and their Applications*, 8(8):2287–2311, 2021. Special Issue: Intuitionistic Modal Logic and Applications. URL: http://collegepublications.co.uk/ifcolog/?00050.

17 Oleg Kiselyov. The design and implementation of BER MetaOCaml. In Michael Codish and Eijiro Sumii, editors, *Proceedings of the 12th International Symposium on Functional and Logic Programming (FLOPS)*, volume 8475 of *Lecture Notes in Computer Science*, pages 86–102. Springer, Cham, 2014. doi:10.1007/978-3-319-07151-0_6.

18 Magnus Baunsgaard Kristensen, Rasmus Ejlers Møgelberg, and Andrea Vezzosi. Greatest HITs: Higher inductive types in coinductives via induction under clocks. *ArXiv preprint*, 2021. arXiv:2102.01969.

19 Jean-Louis Krivine. A program for the full axiom of choice. *Logical Methods in Computer Science*, 17(3):21:1–22, 2021. doi:10.46298/lmcs-17(3:21)2021.

20 Rasmus Ejlers Møgelberg and Niccolò Veltri. Bisimulation as path type for guarded recursive types. *Proceedings of the ACM on Programming Languages*, 3(POPL):4:1–29, 2019. doi:10.1145/3290317.

21 Hiroshi Nakano. A modality for recursion. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society, 2000. doi:10.1109/LICS.2000.855774.

22 Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. Contextual modal type theory. *ACM Transactions on Computational Logic*, 9(3):1–49, 2008. doi:10.1145/1352582.1352591.

23 Frank Pfenning. Intensionality, extensionality, and proof irrelevance in modal type theory. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 221–230. IEEE Computer Society, 2002. doi:10.1109/LICS.2001.932499.

24 Andrew Polonsky. Axiomatizing the quote. In Marc Bezem, editor, *Computer Science Logic (CSL)—25th International Workshop/20th Annual Conference of the EACSL*, volume 12 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 458–469, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2011.458.

25 Daniyar S. Shamkanov. Circular proofs for the Gödel-Löb provability logic. *Mathematical Notes*, 96(3–4):575–585, 2014. doi:10.1134/S0001434614090326.

26 Daniyar S. Shamkanov. A realization theorem for the Gödel-Löb provability logic. *Sbornik: Mathematics*, 207(9):1344–1360, 2016. doi:10.1070/SM8667.

27 Tim Sheard and Simon Peyton Jones. Template meta-programming for Haskell. In *Proceedings of the 2002 ACM SIGPLAN Workshop on Haskell*, pages 1–16, New York, New York, USA, 2002. ACM Press. doi:10.1145/581690.581691.

28 Walid Taha and Tim Sheard. MetaML and multi-stage programming with explicit annotations. *Theoretical Computer Science*, 248(1–2):211–242, 2000. doi:10.1016/S0304-3975(00)00053-0.

29 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

30 Jaap van Oosten. *Realizability: An Introduction to its Categorical Side*. Studies in Logic and the Foundations of Mathematics. Elsevier, 2008. doi:10.1016/S0049-237X(08)80001-8.

31 Niccolò Veltri and Andrea Vezzosi. Formalizing π-calculus in guarded cubical Agda. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*, pages 270–283, New York, NY, USA, 2020. ACM. doi:10.1145/3372885.3373814.

32 Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical Agda: A dependently typed programming language with univalence and higher inductive types. *Journal of Functional Programming*, 31:e8:1–47, 2021. doi:10.1017/s0956796821000034.