A modal deconstruction of Löb induction

Anonymous Author(s)

ABSTRACT

We present a novel analysis of the fundamental Löb induction principle from guarded recursion. Taking advantage of recent work in modal type theory and univalent foundations, we derive Löb induction from a simpler and more conceptual set of primitives. We then capitalize on these insights to present Gatsby, the first guarded type theory capturing the rich modal structure of the topos of trees alongside Löb induction without immediately precluding canonicity or normalization. We show that Gatsby can recover many prior approaches to guarded recursion and use its additional power to improve on prior examples. We crucially rely on homotopical insights and Gatsby constitutes a new application of univalent foundations to the theory of programming languages.

ACM Reference Format:
Anonymous Author(s). 2024. A modal deconstruction of Löb induction. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recursive definitions have long been both a hallmark of the theory of programming languages and a sore point for type theory. Topics as varied as the denotational semantics of λ-calculus or logical relations for higher-order mutable references all prominently feature complex recursive definitions at their heart. Those techniques which construct solutions for such recursive equations (domain theory, step-indexing, etc.) are among the most commonly used within programming language theory.

On the other hand, recursive definitions in type theory introduce two serious complications. First, and most notably, they are simply unsound in general. A type theory that includes an unrestricted fixed-point operator is easily seen to be unsound with $\mathsf{fix}(x.x) : \bot$. This problem is usually addressed by restricting the fixed-point operator to apply only to types and operations where we can guarantee the existence of a fixed-point. A typical such example is present in proof assistants like Coq or Agda, where only structural recursion is permitted. This discipline is sufficient to accommodate inductive arguments and similar, but insufficient for the equations arising in programming languages which often fail to even induce a monotone operator. A successful line of work (*guarded domain theory*) has focused on replacing recursive equations with *guarded* recursive equations and applying the results to programming languages [2, 7, 8, 30].

Conference'17, July 2017, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.1 Guarded type theory

Guarded domain theory extends type theory with a new connective $\blacktriangleright$ (pronounced *later*) where intuitively $\blacktriangleright A$ classifies computations which will eventually produce an element of A but only after work has been done. Crucially, while there is a natural map $\mathsf{next} : A \to \blacktriangleright A$ there is no natural map in the reverse direction. It is therefore sound to add a restricted version of fix, Löb induction:

$$\mathsf{loeb} : (\blacktriangleright A \to A) \to A$$

Even after making such a restriction, however, a second problem emerges: decidability. Modern type theories strive to maintain a decidable type-checking in order to facilitate an implementation and even adding the more restrictive operator loeb with the computation rule $\mathsf{loeb}(f) = f(\mathsf{next}(\mathsf{loeb}(f)))$ is sufficient to render type-checking undecidable. This problem, along with other complications with integrating $\blacktriangleright$ into dependent type theory, has motivated over a decade of proposals for guarded type theory.

Thesis 1. *We can summarize the aspirations for an ideal guarded type theory as the following four goals:*

1. *Include* $\blacktriangleright$ *along with e.g., the* always *comonad.*
2. *Include* loeb *with a propositional equality stating it unfolds.*
3. *Ensure that closed elements of* **Nat** *compute to numerals.*
4. *Ensure that type-checking is decidable.*[1]

Example 1.1. We illustrate how these requirements may be used simultaneously. We use 1 and 2 to define the type of guarded streams alongside a propositional equality $\mathsf{GStr} = \mathbf{Nat} \times \blacktriangleright\mathsf{GStr}$. Using 2 again, we then construct e.g., fibs the guarded stream of Fibonacci numbers. With the always comonad from 1, we can pass under n copies of $\blacktriangleright$ to extract the nth element of fibs. Finally, 3 ensures that this natural number computes to a genuine numeral while 4 implies that the entire program could be implemented in a proof assistant.

Historically, even the first of these goals was a serious challenge since the integration of multiple interacting modalities proved to be difficult. For instance, much of the work on guarded recursion uses the global sections or *always* comonad $\square$, which cannot be included as an operation $\mathbf{U} \to \mathbf{U}$ in type theory [34]. Early attempts to incorporate both $\blacktriangleright$ and $\square$ simultaneously precluded the final two desiderata without even considering Löb induction [11, 13].

Recently, Gratzer et al. [18] proposed MTT as a convenient framework for dependent type theories supporting arbitrarily many modalities, including all those necessary for guarded recursion. Later work [15] has further shown that MTT enjoys both decidable type-checking and canonicity, ensuring that this general framework can be instantiated to automatically yield a type theory satisfying Goals 1, 3 and 4. Unfortunately, balancing these requirements with Goal 2 has proven to be another substantial challenge.

Two flavors of MTT have been proposed which include Löb induction [17, 18]. Both involve adding loeb as an axiom but thereafter diverge: either adding an axiom ensuring that it unfolds up to

[1]This requirement is tantamount to requiring a normalization algorithm.

propositional equality or simply adding a definitional equality to this effect. Adding only propositional unfolding will preclude validating Goal 3 in the resulting type theory. Gratzer and Birkedal [17] showed, however, that adding definitional unfolding will address Goal 3 at the cost of refuting Goal 4. Indeed, the "no-go" theorem of op. cit. caused type theorists to weaken the second aspiration of guarded type theory from "include loeb with a definitional unfolding equation" to the 2 we listed above.

In this work, we propose a novel alternative approach. We do not start by axiomatizing Löb induction and attempting to balance its computation rule with decidability. Instead, we enrich our modal framework to obtain our new type theory **Gatsby** and *derive* Löb induction from these more basic principles.

1.2 Guarded accessible type theory: Gatsby

Following prior work, we also build Gatsby atop MTT. To do so, we must choose a mode theory—a 2-category—specifying the collection of modalities we wish to use. Roughly, each object m represents a different type theory which are connected by a modality $\langle \mu \mid - \rangle$ for each morphism μ in $\mathcal{M}$. Finally, the 2-cells in a mode theory introduce transformations between modalities. However, instantiating MTT on its own is insufficient. We prove that it is impossible to derive Löb induction just from modalities:

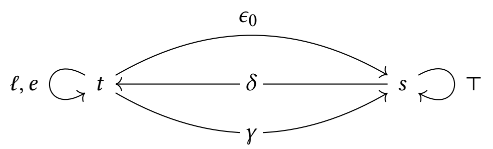
Theorem 2.7 (No-go)**.** *For any $\mathcal{M}$ and $\mu \in \mathcal{M}$, there is no term $(\langle \mu \mid \mathbf{Void} \rangle \to \mathbf{Void}) \to \mathbf{Void}$ in cubical MTT instantiated with $\mathcal{M}$.*

This is because nothing in MTT prevents modalities from being trivial: there is always a model of the system which realizes each modality by the identity. To rule out such degenerate models, we then isolate a simple reasoning principle, similar to existing rules in cubical type theory, which enriches the entire system enough to rule out trivial models and thereby derive Löb induction.

1.2.1 From modalities to Löb induction. While no mode theory is sufficient on its own, we must first choose a particular mode theory with which to instantiate MTT. In our case, we have two modes t and s. The first t represents the guarded type theory and will intuitively model the topos of **t**rees $\mathbf{PSh}(\omega)$. The second s is meant to represent ordinary non-guarded type theory and will attempt to model **s**ets. These modes are then linked by a collection of modalities, and the modalities are equipped with a partial order. We give the entire mode theory in Fig. 1. Many aspects of this mode theory were already explored by Gratzer et al. [18]. For instance, ℓ and e correspond to ► and its left adjoint ◄. The composites $\delta \circ \epsilon_0$ and $\delta \circ \gamma$ correspond to the possibly and always modalities $\Diamond$ and $\Box$, though we have chosen to break them into adjunctions.

The crucial novelty to our system is in the final modality: $\top$. Ironically, this modality represents the simplest possible modality, the one which sends every type to **Unit**. However, while Theorem 2.7 shows that it is impossible to realize loeb just from modalities, if we force $\langle \top \mid A \rangle \simeq \mathbf{Unit}$, this impacts the behavior of the other modalities enough to make loeb induction derivable.

Concretely, we isolate a particular proposition acc at mode t and show that under the assumption of acc both Löb induction and its unfolding principle are derivable. While this proposition is true in our intended model, it is not derivable in Gatsby. Indeed, the no-go theorem above Gratzer and Birkedal [17] shows that it



$$\begin{tikzcd} t \arrow[loop left, "\ell, e"] \arrow[rr, bend left, "\epsilon_0"] \arrow[rr, bend right, "\gamma"'] & & s \arrow[ll, "\delta"'] \arrow[loop right, "\top"] \end{tikzcd}$$

$$\mathrm{id} \le \ell \qquad \mathrm{id} \le \ell \circ e \qquad e \circ \ell = \mathrm{id} \qquad \gamma \circ \delta = \mathrm{id}$$

$$\gamma \circ \ell = \gamma \circ e = \gamma \qquad \delta \circ \gamma \le \mathrm{id} \qquad \epsilon_0 \circ \delta = \mathrm{id} \qquad \mathrm{id} \le \delta \circ \epsilon_0$$

$$\xi \le \nu \circ \top \circ \mu \qquad \top \circ \epsilon_0 = \epsilon_0 \circ \ell$$

Figure 1: $\mathcal{M}_{\mathbf{Gatsby}}$: the mode theory for Gatsby

must not be. However, this is where the more sophisticated modal framework and special behavior of $\top$ comes into play. While acc is not provable, $\langle \mu \mid \mathrm{acc} \rangle$ holds for a large class of modalities μ.

One can then write a program which assumes acc and thereby has access to Löb induction and guarded reasoning. Once the program is completed, a user can apply $\langle \mu \mid - \rangle$ to the entire closed term and discharge the acc assumption to actually compute a result. In this manner, the choice of μ plays the role of type-based "fuel", allowing a user to extract arbitrary but finite prefixes from a guarded type without requiring the type-checker to compare infinite types.

1.2.2 Working with accessible types. While theoretically sufficient, the prospect of carrying acc through every computation is unpleasant. Moreover, in a theory like Gatsby with modalities, real problems could possibly emerge. For instance, we may pass under $\Box$ and lose access to the acc assumption we desired. Both problems can be resolved simultaneously by restricting to types which are *accessible*, i.e., where A is equivalent to $\mathrm{acc} \to A$. Intuitively, these are types for which there is a canonical and best way to discharge an acc assumption and so it can be done automatically.

When working with an accessible type, there is no need to carry around a proof of acc as it can always be obtained from the goal itself. We show that that the subuniverse of accessible types supports a model of guarded type theory closed under all the connectives of type theory, including all modal operators and the universe. Remarkably, even types like booleans, natural numbers, and $\Box A$ are automatically accessible. The result is that essentially any standard guarded recursive algorithm will need to mention only accessible types, freeing the user from any obligation to think about acc when programming. Formally, we have the following result:

Theorem 3.13 (Completeness)**.** *Any program written in MLTT with ►, $\Box$, and* loeb *with propositional unfolding can be encoded in Gatsby.*

More informally, our strategy is complete with respect to a more standard formulation of guarded recursion.

1.3 Closely related approaches

While comparison to related work is carried out in Section 6, two approaches are sufficiently close to warrant earlier mention: stratified guarded type theory [17] and clocked cubical type theory [3, 21].

Stratified guarded type theory (GuTT) [17] balances the tensions of Thesis 1 by proposing two separate but related type theories: one satisfying Goals 1, 2 and 4 and one satisfying Goals 1 to 3. Gatsby sharpens this idea by careful unifying these theories through a more sophisticated modal analysis of Löb induction. In particular, by discarding GuTT's Löb induction axioms, we are able to recover GuTT's type-based notion of fuel through the class of modalities for which $\langle \mu \mid \mathsf{acc} \rangle$. We then use this to derive Löb induction. Gatsby also offers a richer class of modalities than GuTT and seamlessly includes $\square$.

Clocked cubical type theory (CloTT$_\square$) [21] also attempts to satisfy Thesis 1 but through very different means. Firstly, rather than including multiple interacting modalities, CloTT$_\square$ includes only $\blacktriangleright$ but then *indexes* it by a clock κ which can be quantified over. The presence of clocks allows CloTT$_\square$ to add Löb induction as an axiom which unfolds only in specific circumstances; roughly after the particular clock κ has been bound to prevent any further occurrences of $\blacktriangleright^\kappa$. Like Gatsby, CloTT$_\square$ conjecturally satisfies canonicity and decidable type-checking and, like Gatsby, it therefore provisionally satisfies Goals 1 to 4.[2] However, the approach is very different.

CloTT$_\square$'s indexed $\blacktriangleright$ modality offers a richer framework for guarded programming. However, clock quantification does not replicate all uses of $\square$ and forces a more complex semantics [24]. Moreover, clock quantification necessitates considering *clock-irrelevant* types, complicating the theory of universes. On the other hand, while Gatsby enjoys a simpler semantics, richer modalities, and better-behaved universes, programming with modalities can be less intuitive than with clocks and clocks are required for nested guarded types. The biggest difference is in the approach Gatsby and CloTT$_\square$ take for Löb induction. While CloTT$_\square$ adds in Löb induction as an axiom and restricts its computation to decide type-checking, Gatsby *derives* Löb induction from modalities. In addition to the intrinsic interest of this decomposition, this approach arguably makes it easier to extend Gatsby with additional reasoning principles.

Despite these differences, Gatsby has crucially benefited from the work behind CloTT$_\square$. The CloTT$_\square$ approach to clock-irrelevant types motivates our exploration of accessible types which substantially improves the usability of Gatsby. Moreover, the cubical variant of MTT [1] which Gatsby extends is inspired by CloTT$_\square$.

1.4 Contributions

We present *guarded accessible type theory* or Gatsby: a modal type theory built on MTT extended with a new primitive computational modality $\top$. From this purely modal extension, we derive Löb induction and offer a complete solution to the problems of the original guarded dependent type theory proposed by Birkedal et al. [7].

- We show that Gatsby satisfies Goals 1 and 2 above and offer evidence for Goals 3 and 4.
- We show that more standard presentations of guarded recursion can be compiled faithfully into Gatsby.
- We demonstrate that Gatsby is usable by providing a case-study extending an example given by Birkedal et al. [7].

Gatsby is the first guarded type theory to support computational Löb induction alongside the rich modal structure of $\mathbf{PSh}(\omega)$ $(\blacktriangleright, \blacktriangleleft, \square, \diamond)$ without immediately precluding decidable type-checking.

[2]The non-cubical variant of CloTT$_\square$ [3] is known to enjoy normalization and satisfies Goals 1, 3 and 4. However, it does not include any means of proving $\mathsf{loeb}(f) = f(\mathsf{next}(\mathsf{loeb} f))$ and therefore does not satisfy Goal 2.

Moreover, the good behavior of universes and propositions in univalent foundations is critical to Gatsby, making it a novel application of homotopy type theory to programming languages.

Organization. In Section 2 we introduce *cubical MTT*, the foundation of Gatsby. Section 3 introduces Gatsby itself and tours through its most essential features. Section 4 uses Gatsby to improve upon a logical-relations argument from Birkedal et al. [7]. Finally, Section 5 discusses the semantics of Gatsby and proves its soundness.

2 CUBICAL MTT AND GUARDED RECURSION

Gatsby depends on a computational account of univalence and a well-behaved theory of modalities. In this section we recall some of the details of a system which contains both: Cubical MTT [1]. Cubical MTT is a variant on ordinary MTT in which every mode contains a copy of cubical type theory, as opposed to the ordinary copy of intensional Martin-Löf type theory. Fortunately, the details of cubical type theory are not essential to understanding Gatsby. The same cannot be said of the theory of modalities used by Gatsby where details do matter.

Accordingly, we will give a brief overview of MTT only, as the reader will be able to follow all the constructions of Sections 3 and 4 simply by assuming that Gatsby is built atop ordinary MTT extended with univalence; the only import of the cubical implementation of univalence is that it is computational.[3]

For a comprehensive review of MTT we refer the reader to Gratzer [16, Chapter 6] or Gratzer et al. [19]. For cubical type theory and computational univalence, we suggest Cohen et al. [14].

A glossary for homotopy type theory. We will take advantage of a variety of standard notations and definitions from homotopy type theory. The standard reference for these is the HoTT Book [40].

For convenience, we recall that if $p : x = y$ then $\mathsf{transport}_B\, p : B\,x \longrightarrow B\,y$. In the special case where $B = \mathsf{id}$, then we write p_* for the induced map. If $f : A \longrightarrow B$ and $p : a_0 = a_1$, we write $\mathsf{ap}_f\, p$ for the induced path $f\,a_0 = f\,a_1$. We shall have frequent occasion to use the univalent version of propositions: homotopy proposition or hprop. An hprop is a type A equipped with a (necessarily unique) function $(a_0, a_1 : A) \longrightarrow a_0 = a_1$. We write hProp_i for the subtype of the universe $\mathbf{U}_i$ spanned by hprops. We write $\|A\|$ for the unique homotopy proposition equipped with a map $\eta : A \longrightarrow \|A\|$ such that $\eta^* : P^{\|A\|} \longrightarrow P^A$ is an equivalence for all propositions P.

Finally, for convenience, we shall generally omit subscripts around universes and avoid discussing size issues. So we shall write $\mathbf{U} : \mathbf{U}$ and leave it to the reader to insert the subscripts $\mathbf{U}_i : \mathbf{U}_{i+1}$. This also applies to hProp_i for which we will simply write hProp.

2.1 A summary of multimodal type theory

To a first approximation, MTT is a machine which accepts as input an abstract specification of modalities (a *mode theory* [23]) and produces a modal type theory as output satisfying canonicity and normalization [15]. While generally, a mode theory is allowed to be a strict 2-category, for our purposes it suffices to consider a 1-category *enriched in partial orders*. That is, a mode theory is a

[3]We conjecture that an alternative account of computational univalence—e.g., the work by Shulman, Altenkirch, and Kaposi [35]—could be adapted for a similar purpose.

category $\mathcal{M}$ where every hom-set $\hom_{\mathcal{M}}(x, y)$ is a partial order such that composition is monotone.

Suppose MTT is instantiated with some mode theory $\mathcal{M}$. The objects of $\mathcal{M}$ are called *modes* (ranged over by m, n, o) and the morphisms are *modalities* (ranged over by μ, ν, ξ). In MTT, each mode m yields a separate copy of type theory with its attendant set of judgments ($\vdash \Gamma\ \mathsf{cx} \ @\ m$, $\Gamma \vdash A\ \mathsf{type}\ @\ m$, etc.). Each type theory is then extended with several operations to reflect the modalities which link the modes:

$$\begin{array}{llcl} (Cx) & \Gamma, \Delta & ::= & \dots \mid \Gamma.(\mu \mid A) \mid \Gamma.\{\mu\} \\ (Ty) & A, B & ::= & \dots \mid \langle \mu \mid A \rangle \\ (Tm) & M, N & ::= & \dots \mid \mathbf{mod}_\mu(M) \mid \mathbf{let}_\nu\ \mathbf{mod}_\mu(-) \leftarrow M\ \mathbf{in}\ N \end{array}$$

The idea behind MTT is that each modality $\mu : n \longrightarrow m$ induces (1) a type former $\langle \mu \mid - \rangle$ sending mode n types to mode m types along with (2) a context former $-.\{\mu\}$ sending m contexts to n contexts. We add equations ensuring that $\Gamma.\{\mathsf{id}\} = \Gamma$ and $\Gamma.\{\nu\}.\{\mu\} = \Gamma.\{\nu\mu\}$.

Roughly, $-.\{\mu\}$ is the *left adjoint* to $\langle \mu \mid - \rangle$; modalities in MTT are (weak) *dependent right adjoints* [5]. Hence, an element $\Gamma \vdash M : \langle \mu \mid A \rangle$ is roughly equivalent to an element $\Gamma.\{\mu\} \vdash N : A$. Converting N to M is the role of the introduction rule:

$$\frac{\Gamma.\{\mu\} \vdash A\ \mathsf{type}\ @\ n}{\Gamma \vdash \langle \mu \mid A \rangle\ \mathsf{type}\ @\ m} \qquad \frac{\Gamma.\{\mu\} \vdash M : A\ @\ n}{\Gamma \vdash \mathbf{mod}_\mu(M) : \langle \mu \mid A \rangle\ @\ m}$$

The passage from M to N is more fraught. The elimination principle ought to give such an inverse. Directly adding such an operation, however, disrupts the substitution property of the type theory. We instead add a "pattern-matching" elimination rule similar to the elimination principle used for booleans or natural numbers.

It is here that the modified form of context $\Gamma.(\mu \mid A)$ is used, so we postpone the elimination rule and first explain how this novel form of context extension works. Roughly, $\Gamma.(\mu \mid A)$ is a context with a variable of type A *annotated with* μ. While morally the same as an element of $\langle \mu \mid A \rangle$, the force of the annotation is that it can be used in the variable rule to pull a variable out from behind $\{\mu\}$:

$$\frac{\Gamma.\{\mu\} \vdash A\ \mathsf{type}\ @\ n \qquad \mu \leq \nu : n \longrightarrow m}{\Gamma.(\mu \mid A).\{\nu\} \vdash \mathbf{v} : A[\mathbf{p}.\{\nu\}]\ @\ m}$$

The variable rule is where the partial order on modalities enters the type theory. The most basic form of the variable rule states that a μ-annotated variable can be used behind $\{\mu\}$. We enhance this rule slightly by not requiring the annotation μ to precisely match the restriction $\{\nu\}$. Instead, we require only that the annotation implies the restriction: $\mu \leq \nu$.

Notation 2.1. We have written this rule using de Bruijn indices and explicit substitutions. This is convenient when specifying modal type theories, as substitution is often subtle and non-standard. In future examples we shall opt for more readable named variables.

Notation 2.2. When writing MTT terms "informally", we will write $x :_\mu A$ to indicate the variable x exists in the context with annotation μ and type A. If $\mu = \mathsf{id}$, we suppress it to recover the traditional notation. See Gratzer [16, Section 6.1] for a detailed discussion.

We now return to the original question of the elimination rule for $\langle \mu \mid A \rangle$. Intuitively, this rule ensures that it suffices to assume every element of $\langle \mu \mid A \rangle$ is of the form $\mathbf{mod}_\mu(-)$:

$$\frac{\begin{array}{c} \nu : m \longrightarrow o \\ \Gamma.\{\nu\} \vdash M : \langle \mu \mid A \rangle\ @\ m \qquad \Gamma.(\nu \mid \langle \mu \mid A \rangle) \vdash B\ \mathsf{type}\ @\ o \\ \Gamma.(\nu \circ \mu \mid A) \vdash N : B[\mathsf{id}.\mathbf{mod}_\mu(\mathbf{v})]\ @\ o \end{array}}{\Gamma \vdash \mathbf{let}_\nu\ \mathbf{mod}_\mu(-) \leftarrow M\ \mathbf{in}\ N : B[\mathsf{id}.M]\ @\ o}$$

$$\mathbf{let}_\nu\ \mathbf{mod}_\mu(-) \leftarrow \mathbf{mod}_\mu(M)\ \mathbf{in}\ N = N[\mathsf{id}.M]$$

We will take occasional advantage of a convenience feature of MTT. It is common to accept $x : \langle \mu \mid A \rangle$ as an argument and immediately pattern-match on it. This is simplified by modal functions:

$$\frac{\Gamma.\{\mu\} \vdash A\ \mathsf{type}\ @\ n \qquad \Gamma.(\mu \mid A) \vdash M : B\ @\ m}{\Gamma \vdash \lambda M : (\mu \mid A) \to B\ @\ m}$$

$$\frac{\Gamma.\{\mu\} \vdash N : A\ @\ n \qquad \Gamma \vdash M : (\mu \mid A) \to B\ @\ m}{\Gamma \vdash M(N) : B[\mathsf{id}.N]\ @\ m}$$

We use $(x :_\mu A) \to B(x)$ for the informal version of $(\mu \mid A) \to B$.

2.2 Instantiating MTT with $\mathcal{M}_{\mathsf{Gatsby}}$

We now turn from MTT in general to its instantiation with the specific mode theory required for Gatsby as detailed in Fig. 1. The mode theory is a combination of mode theories previously used to adapt MTT to study guarded recursion [16–18].

Unlike many presentations of guarded recursion, it contains multiple modes. The types at mode t are intended to behave like objects from $\mathbf{PSh}(\omega)$ and can exhibit guarded behavior, while s is intended to capture $\mathbf{Set}$. Each modality μ is intended to capture a particular right adjoint F_μ on or between these categories:

$$\begin{array}{lll} F_\ell(X) = \blacktriangleright X & F_e(X) = \blacktriangleleft X & F_\delta(S) = \lambda n.\, S \\ F_\gamma(X) = \lim\limits_{\omega} X & F_{\epsilon_0}(X) = X(0) & F_\top(X) = \{\star\} \end{array}$$

See Birkedal et al. [7] for the definitions of $\blacktriangleleft$ and $\blacktriangleright$ on $\mathbf{PSh}(\omega)$. Inspired by the above, we will write $\blacktriangleright A \triangleq \langle \ell \mid A \rangle$, $\blacktriangleleft A \triangleq \langle e \mid A \rangle$, $\square A \triangleq \langle \delta \circ \gamma \mid A \rangle$, and $\Diamond A \triangleq \langle \delta \circ \epsilon_0 \mid A \rangle$.

Remark 2.3. For the sake of simplicity, we have chosen to ignore the complications imposed by modeling cubical MTT and interpreting univalence above. In Section 5 we shall return to this point and show that the preceding sketch can be turned into a valid construction by replacing sets with cubical sets and presheaves with cubical presheaves. For the next two sections we shall ignore univalence when providing semantic intuitions and simply note that all arguments can be adapted to cubical sets with minimal changes.

Several modal combinators are definable for any modality in MTT and, when appropriately instantiated, these yield familiar operations in guarded recursion. In particular, we shall have use for the following general MTT combinators:

$$\circledast : \langle \mu \mid A \to B \rangle \to \langle \mu \mid A \rangle \to \langle \mu \mid B \rangle \qquad \mathsf{triv} : \langle \mathsf{id} \mid A \rangle \simeq A$$

$$\mathsf{comp} : \langle \nu \mid \langle \mu \mid A \rangle \rangle \simeq \langle \nu \circ \mu \mid A \rangle \qquad \mathsf{coe}_{\mu \leq \nu} : \langle \mu \mid A \rangle \to \langle \nu \mid A \rangle$$

For instance, the standard map $\mathsf{next} : A \to \blacktriangleright A$ is defined as $\mathsf{coe}_{\mathsf{id} \leq \ell} \circ \mathsf{triv}^{-1}$ and the equivalence $\mathsf{now} : \square \blacktriangleright A \simeq \square A$ is realized by comp. We record these and more in an omni-bus theorem:

Theorem 2.4.

(1) *$\blacktriangleright$ is a well-pointed applicative functor,*
(2) *$\square$ is an idempotent comonad and $\square\blacktriangleright \simeq \square \simeq \square\blacktriangleleft$,*
(3) *the following pairs of modalities are adjoint: $\blacktriangleleft \dashv \blacktriangleright$, $\diamond \dashv \square$, $\langle \delta \mid - \rangle \dashv \langle \gamma \mid - \rangle$, and $\langle \epsilon_0 \mid - \rangle \dashv \langle \delta \mid - \rangle$.*

We emphasize that certain modalities, $\blacktriangleleft$ and $\langle \delta \mid - \rangle$, are *left adjoint* modalities. These modalities enjoy particularly good behavior in MTT in the form of a variety of *crisp induction principles* [16, 34]. For instance, $\langle \delta \mid \mathbf{Nat} \rangle$ is equivalent to **Nat** and we may therefore perform induction on an element of the former as if it were the latter. This is not the case for every modality $\langle \mu \mid - \rangle$. In general, inductive types are preserved by left adjoint modalities, and we shall capitalize on this fact several times.

Finally, we note two specific consequences of working with cubical MTT rather than ordinary MTT. First, as already mentioned, each mode supports the univalence axiom. Second, each modality preserves identity types:

Theorem 2.5. *The map $\mathbf{mod}_\mu(a) = \mathbf{mod}_\mu(b) \to \langle \mu \mid a = b \rangle$ sending* **refl** *to $\mathbf{mod}_\mu(\mathbf{refl})$ is an equivalence. We denote the inverse map $\iota : \langle \mu \mid a = b \rangle \to \mathbf{mod}_\mu(a) = \mathbf{mod}_\mu(b)$.*

Corollary 2.6. *The map $\langle \mu \mid A \to B \rangle \to (\langle \mu \mid A \rangle \to \langle \mu \mid B \rangle)$ restricts to a map $\langle \mu \mid A \simeq B \rangle \to (\langle \mu \mid A \rangle \simeq \langle \mu \mid B \rangle)$.*

2.3 A no-go theorem for Löb induction

Thus far we have avoided any mention of Löb induction. Indeed, as an instantiation of cubical MTT, our theory thus far cannot possibly validate Löb induction without additional axioms. We now give a proof of this no-go theorem which implies that no choice of mode theory makes Löb induction derivable in an instantiation of MTT without additional axioms.

Theorem 2.7 (No-go). *For any $\mathcal{M}$ and $\mu \in \mathcal{M}$, there is no term $(\langle \mu \mid \mathbf{Void} \rangle \to \mathbf{Void}) \to \mathbf{Void}$ in cubical MTT instantiated with $\mathcal{M}$.*

Proof. MTT with mode theory $\mathcal{M}$ has a model where each mode is interpreted by ordinary (cubical) type theory and each modality is realized by the identity. If $(\langle \mu \mid \mathbf{Void} \rangle \to \mathbf{Void}) \to \mathbf{Void}$ was derivable in (cubical) MTT, then interpreting this term into this model would yield an element of $(\llbracket \mathbf{Void} \rrbracket \to \llbracket \mathbf{Void} \rrbracket) \to \llbracket \mathbf{Void} \rrbracket$ in (cubical) type theory for each A in MTT. Since $\llbracket \mathbf{Void} \rrbracket = \mathbf{Void}$ (the empty type), we would have $M : (\mathbf{Void} \to \mathbf{Void}) \to \mathbf{Void}$ and so $M(\mathrm{id}) : \mathbf{Void}$; this contradicts the soundness of cubical type theory. □

3 INTRODUCING Gatsby

In Section 2 we introduced (cubical) MTT and discussed its instantiation to $\mathcal{M}_{\mathrm{Gatsby}}$. As it stands, this instantiation must be extended with a new rule in order to validate Löb induction (Theorem 2.7). We now present this new rule (Rule 1) and show that it forces $\langle \top \mid A \rangle = \mathbf{Unit}$. We term the system extended with it Gatsby.

A priori, constraining $\top$ in this manner is entirely unrelated to Löb induction. We prove, however, that the interactions between $\top$ and other modalities in the system give us the ability to *define* Löb induction on a large class of types. We carry out this derivation in stages; we define a homotopy proposition which entails Löb induction (Theorem 3.5) and then show that this homotopy proposition holds in certain circumstances (Theorem 3.7). In Sections 3.3 and 3.4, we then show how to parlay this new reasoning principle into a workable guarded type theory.

3.1 The new rule

As shown in Theorem 2.7, the key obstruction to deriving Löb induction is that we cannot guarantee that $\blacktriangleright$ (or indeed any of our modalities) is not actually the identity. In order to rule this out, we shall add a new rule forcing one modality, $\top$, to be not the identity. To do this, note that $\langle \top \mid A \rangle$ is intended to encode **Unit** and so its left adjoint action $\Gamma.\{\top\}$ corresponds to extending Γ by the empty type. In particular, $\Gamma.\{\top\}$ will be interpreted by the empty set. With this in mind, we add the following rule:

$$\frac{\Gamma \vdash r : \mathbf{1}.\{\top\} \;@\; s}{\Gamma \vdash \mathcal{J}} \tag{1}$$

In the above, $\mathcal{J}$ ranges over any judgment. Intuitively, if there exist a substitution from Γ to $\mathbf{1}.\{\top\}$, then Γ will be interpreted by the empty set as well and so the principle of explosion ought to let us conclude any judgment we wish in this context.

Remark 3.1. This rule is similar to the rule in cubical type theory stating that $\Gamma \vdash \mathcal{J}$ holds if Γ proves the false cofibration [14].

While we phrased this rule in an algebraic way by asking for a substitution from the context to $\mathbf{1}.\{\top\}$, we can prove that it is decidable whether such a substitution exists (see Proposition 7.1). Such a metatheorem is crucial for showing that Gatsby admits a normalization algorithm. Indeed, similar rules yield undecidable type theories because of the absence of such a result.

Lemma 3.2. *Rule 1 implies $\langle \top \mid A \rangle = \mathbf{Unit}$.*

Proof. First, we note that $\langle \top \mid A \rangle$ is inhabited. Indeed, to construct an element of $\langle \top \mid A \rangle$, it suffices to construct an element of A in a context restricted by $\{\top\}$ but Rule 1 ensures this is true regardless of A. By univalence, it then suffices to argue that all elements in $\langle \top \mid A \rangle$ are equal. While one can argue directly, this is an immediate consequence of Theorem 2.5 and the argument above after replacing A by $a_0 =_A a_1$. □

In light of this result, we say that $\langle \top \mid - \rangle$ is a *computational* modality [20]. On its own, constraining $\langle \top \mid A \rangle$ to be **Unit** is not a useful change. After all, if we only cared about constraining $\langle \top \mid A \rangle$ we could simply discard the modal apparatus and take this equivalence to be an equality. The force of this extension comes through the interactions between $\top$ and other modalities. In particular, $\epsilon_0 \circ \ell = \top \circ \epsilon_0$ together with Lemma 3.2 implies the following:

$$\langle \epsilon_0 \mid \blacktriangleright A \rangle = \mathbf{Unit}$$

While this is obviously true in the model (the first stage of $\blacktriangleright X$ is $\{\star\}$ by definition), it is only after forcing $\langle \top \mid A \rangle = \mathbf{Unit}$ that we can derive this result purely within the theory. Having some circumstances under which $\blacktriangleright A$ trivializes opens up a new avenue to encoding Löb induction.

3.2 The accessible proposition

Contrary to prior guarded type theories, we will not endeavor to make Löb induction directly available on each type at mode t. We will instead split things up into two steps:

(1) Define an (h-)proposition acc which implies Löb induction.
(2) Prove that $\langle \mu \mid \mathsf{acc} \rangle$ holds for a large class of modalities μ.

This two-step approach offers a way to include Löb induction in Gatsby without directly postulating it. Instead, when working with guarded recursion we will *assume* acc holds, such that the type of the end result will be $\mathsf{acc} \longrightarrow A$. In order to actually run such a program, we then choose an appropriate modality μ and switch to considering $\langle \mu \mid \mathsf{acc} \longrightarrow A \rangle$. The general principles of MTT modalities together with our proof of $\langle \mu \mid \mathsf{acc} \rangle$ then enables us to conclude $\langle \mu \mid A \rangle$. By choosing μ appropriately, we are then able to extract arbitrary finite approximations of A and compute with Löb induction without ever directly postulating it. While the first step can be carried out in ordinary MTT, showing that $\langle \mu \mid \mathsf{acc} \rangle$ is true for any modality requires Gatsby's computational $\top$ modality.

The proposition acc is defined using the truncation modality $\|{-}\|$ from homotopy type theory:

Definition 3.3. We define $\mathsf{acc} = \left\|\sum_{n:\mathsf{Nat}} \blacktriangleright^n \mathbf{Void}\right\|$ at mode t.

Remark 3.4. Note that acc is globally true when t is interpreted into $\mathbf{PSh}(\omega)$ where it is realized by $\exists n. \blacktriangleright^n \bot$. In this model, $\blacktriangleright^n \bot$ has a more recognizable form as the representable presheaf $\mathbf{y}(n)$. From this viewpoint, acc is equivalent to the join of all $\mathbf{y}(n)$ which is $\mathbf{1}$ by construction. However, while it holds in the intended model, it is not derivable within Gatsby. For instance, one could interpret mode t using $\mathbf{Sh}(\omega^2)$ where acc is not globally true; it circumscribes the open embedding of $\mathbf{PSh}(\omega)$ into $\mathbf{Sh}(\omega^2)$.

We now set out to prove that assuming acc is sufficient to derive Löb induction. Intuitively, as acc states that there exists some n such that $\blacktriangleright^n \mathbf{Void}$ holds, we ought to be able to define $\mathsf{loeb}(f) : A$ where $f : \blacktriangleright A \longrightarrow A$ by induction on n; if $n = 0$ then we use the now modality-free assumption of $\mathbf{Void}$ and otherwise we apply f to an element of $\blacktriangleright A$ which we obtain by induction. Unfortunately, this naïve proof strategy does not work: the propositional truncation used to define acc means that we cannot directly scrutinize n when constructing an element of A. However, we may strengthen our goal slightly to work around this issue.

Instead of constructing $\mathsf{loeb}(f) : A$, we could instead construct an element $\mathsf{loeb}(f) : A$ together with a proof that $\mathsf{loeb}(f)$ is a guarded fixed-point i.e., $f(\mathsf{next}(\mathsf{loeb}\, f)) = \mathsf{loeb}\, f$. Indeed, while there are potentially many distinct elements of A, we prove that f has at most one guarded fixed-point. We will therefore prove that the type of guarded fixed-points of f is *contractible* (inhabited and all inhabitants are equal). This type is a hprop and so it is valid to scrutinize an element of acc when proving it. We emphasize that this strategy relies heavily on the good behavior of homotopy propositions in univalent foundations.

Theorem 3.5. *In mode t, for any $A : \mathbf{U}$ and $f : \blacktriangleright A \longrightarrow A$ there exists a term $\mathsf{gfix} : \mathsf{acc} \longrightarrow \mathsf{isContr}(\mathsf{GFix}\, A\, f)$ where $\mathsf{GFix}\, A\, f$ is the type of guarded fixed-points $\sum_{a:A} f(\mathsf{next}\, a) = a$.*

Proof. We take advantage of the fact that isContr is a proposition to ignore the truncation on acc and actually scrutinize the underlying natural number. We define gfix as follows:

$$
\begin{aligned}
&\mathsf{gfix}\, \|(0, p)\| = \mathsf{absurd}\, p \\
&\mathsf{gfix}\, \|(1 + n, \mathbf{mod}_\ell(p))\|\, f = \ ?0 : \mathsf{isContr}(\mathsf{GFix}\, f) \\
&\quad \textbf{where} \\
&\quad\quad C :_\ell \mathsf{isContr}(\mathsf{GFix}\, f) \\
&\quad\quad C = \mathsf{gfix}\, \|(n, p)\|\, f
\end{aligned}
$$

To fill this hole, we start with a preliminary observation: given any element of $\blacktriangleright \mathsf{GFix}\, f$, we may promote it to an element of $\mathsf{GFix}\, f$.

$$
\begin{aligned}
&\mathsf{promote} : (r_\ell :_\ell \mathsf{GFix}\, f) \longrightarrow \textstyle\sum_{r:\mathsf{GFix}\, f} \blacktriangleright(r = r_\ell) \\
&\mathsf{promote}\, (a, p) = \textbf{let}\ r = (f(\mathbf{mod}_\ell(a)), \mathsf{ap}_f(\iota(\mathbf{mod}_\ell(p))))\ \textbf{in} \\
&\quad (r, \mathbf{mod}_\ell(\ \{\ell\} \vdash ?1 : r = (a, p)\ ))
\end{aligned}
$$

To fill $?1$, it suffices to fill the following:

$$\{\ell\} \vdash ?2 : f(\mathsf{next}(a)) = a$$

$$\{\ell\} \vdash ?3 : \mathsf{ap}_f(\iota(\mathsf{next}\, p)) = ?2 \bullet p \bullet \mathsf{ap}_{f \circ \mathsf{next}}(?2^{-1})$$

The choice for $?2$ is clear enough: p^{-1}. Plugging this in, it suffices to construct a term of the following type:

$$\{\ell\} \vdash \mathsf{ap}_f(\iota(\mathsf{next}\, p)) = \mathsf{ap}_{f \circ \mathsf{next}}(p)$$

This follows immediately from the definition of ι.

Next, we note that $\pi_1(\mathsf{promote}\, r) = r$. We again have two holes to fill if we let $r = (a, p)$:

$$?4 : f(\mathsf{next}(a)) = a$$

$$?5 : \mathsf{ap}_f(\iota(\mathsf{next}\, p)) = ?2 \bullet p \bullet \mathsf{ap}_{f \circ \mathsf{next}}(?2^{-1})$$

A similar argument fills these two holes with $?4 = p^{-1}$.

We now return to $?1$. It suffices to implement the following:

$$?6 : \mathsf{GFix}\, f \qquad ?7 : (r : \mathsf{GFix}\, f) \longrightarrow\ ?6\ = r$$

We fill $?6 = \mathsf{promote}(\pi_1 C)$. For the second hole, we use our remark that $r = \mathsf{promote}\, r$ together with $\pi_2 C$, transitivity, and ι. □

We can derive Löb induction from gfix:

$$
\begin{aligned}
&\mathsf{loeb} : (A : \mathbf{U}) \longrightarrow (\blacktriangleright A \longrightarrow A) \longrightarrow \mathsf{acc} \longrightarrow A \\
&\mathsf{loeb}\, A\, f\, z = \pi_1(\pi_1(\mathsf{gfix}\, A\, f\, z)) \\
\\
&\mathsf{unfold} : (A : \mathbf{U})(f : \blacktriangleright A \longrightarrow A) \longrightarrow \mathsf{acc} \longrightarrow \\
&\quad f(\mathsf{next}(\mathsf{loeb}\, f)) = \mathsf{loeb}\, f \\
&\mathsf{unfold}\, A\, f\, z = \pi_2(\pi_1(\mathsf{gfix}\, A\, f\, z))
\end{aligned}
$$

Notation 3.6. We shall often work with $\mathsf{loeb} : ((\ell \mid A) \longrightarrow A) \longrightarrow A$ (where the argument type is a modal function type) rather than $(\blacktriangleright A \longrightarrow A) \longrightarrow A$ as the two types are equivalent.

While this is certainly interesting, if we can never discharge the acc assumption from gfix, this brings us no closer to ever being able to actually use these combinators. While we cannot prove acc holds on its own, we can show that $\langle \epsilon_0 \circ e^n \mid \mathsf{acc} \rangle$ holds. We break this statement apart into two results:

Theorem 3.7. *The following hold in modes t and s respectively:*

- $\blacktriangleleft \mathsf{acc} = \mathsf{acc}$
- $\langle \epsilon_0 \mid \mathsf{acc} \rangle = \mathbf{Unit}$

Proof. For the first point, we note that $\blacktriangleleft$ is a left adjoint and therefore commutes with both propositional truncation and natural numbers [16, Lemmas 6.4.16 and 6.4.15]. By propositional univalence, it therefore suffices to show that acc implies the following:

$$\left\|\textstyle\sum_{m:\mathsf{Nat}} \blacktriangleleft\blacktriangleright^m \mathbf{Void}\right\|$$

Scrutinizing our assumption of acc, we know that there exists some k such that $\blacktriangleright^k\mathbf{Void}$; we then choose $m = k+1$ to obtain the goal.

For the second claim, we switch to working in mode s. As all modalities in Gatsby preserve identity types and the unit, they preserve homotopy levels and, in particular, h-props. It therefore suffices to show that $\langle \epsilon_0 \mid \mathrm{acc}\rangle$ holds at all and we shall accomplish this by arguing that $\left\langle \epsilon_0 \mid \sum_{n:\mathbf{Nat}} \blacktriangleright^n\mathbf{Void}\right\rangle$ holds. Commuting this modality with the dependent sum, it suffices to argue that $\langle \epsilon_0 \mid \blacktriangleright\mathbf{Void}\rangle$ holds. However, we have shown in the prior subsection that $\langle \epsilon_0 \mid \blacktriangleright\mathbf{Void}\rangle = \langle \top \mid \langle \epsilon_0 \mid \mathbf{Void}\rangle\rangle = \mathbf{Unit}$. □

Corollary 3.8. *For $n : \mathbf{Nat}$ we have $\langle \epsilon_0 \mid \blacktriangleleft^n\mathrm{acc}\rangle = \mathbf{Unit}$.*

In particular, if we confine ourselves to just working in mode t, we will not be able to (dis)prove acc and so loeb cannot be run. However, if we write a closed program e.g. $M : \mathrm{acc} \to \blacktriangleright^n\mathbf{Nat}$, we can place the entire program under $\langle \epsilon_0 \mid \blacktriangleleft^n - \rangle$ and discharge the acc assumption to obtain a program of type $\langle \epsilon_0 \mid \blacktriangleleft^n \blacktriangleright^n\mathbf{Nat}\rangle \simeq \mathbf{Nat}$. Intuitively, the choice of n allows us to internally recover a type-based "fuel" discipline for running guarded programs. We explore this process in the next two sections.

3.3 The universe of accessible types

The previous subsection outlined an approach for guarded recursion in Gatsby: hypothesize acc when constructing a program and discharge it for closed terms using modalities. However, this simple strategy has two major drawbacks. First, it is hardly convenient to manually thread the acc assumption through every program we write. Second, it may not even be possible to do so! Working in a modal type theory, some connectives restrict access to the context and so we may write a program which assumes acc in the beginning and yet find ourselves unable to rely on this assumption midway through the program having descended beneath □ or ▶.

The solution to both of these problems is the same: rather than manually passing around acc, we should consistently work with the acc-null types. That is, types A such that the canonical map $A \to (\mathrm{acc} \to A)$ is an equivalence. Informally, these are the types that "do not care" if a finite number of steps remain. By restricting attention to these types, we free ourselves of the requirement to pass acc around manually; whenever we require a witness for acc we can simply "pull it out" of the goal A by replacing it with $\mathrm{acc} \to A$.

This has another startling consequence: if we limit ourselves to working with accessible types, we never need to mention or explicitly discharge acc. Each accessible type knows how to remove the acc and we shall see that this allows the process to fade into the background.

Definition 3.9. We say a type is accessible if it is acc-null. The universe of accessible types $\mathbf{U}_{\mathrm{acc}}$ is therefore written as follows:

$$\mathbf{U}_{\mathrm{acc}} = \textstyle\sum_{A:\mathbf{U}} \mathsf{isEquiv}(\mathrm{const} : A \to (\mathrm{acc} \to A))$$

This is a subuniverse of $\mathbf{U}$; the map $\mathbf{U}_{\mathrm{acc}} \to \mathbf{U}$ is an embedding.

If we restrict gfix, loeb, and unfold to applications where A belongs to $\mathbf{U}_{\mathrm{acc}}$ rather than $\mathbf{U}$, we may dispense with the acc hypothesis e.g., $\mathrm{loeb} : (A : \mathbf{U}_{\mathrm{acc}}) \to (\blacktriangleright A \to A) \to A$. This leads us to ask what structure $\mathbf{U}_{\mathrm{acc}}$ possess and which types are accessible.

The universe of accessible types is a *reflective subuniverse* [33]. We therefore conclude a number of results about $\mathbf{U}_{\mathrm{acc}}$ from op. cit.

Proposition 3.10 (Rijke et al. [33])**.**

1. *$\mathbf{U}_{\mathrm{acc}}$ is spanned by ○-modal types where $\bigcirc A = \mathrm{acc} \to A$.*
2. *$\mathbf{U}_{\mathrm{acc}}$ is closed under dependent products and sums, the unit type, and identity types.*
3. *$\mathbf{U}_{\mathrm{acc}}$ and $\mathsf{hProp}_{\mathrm{acc}}$ are accessible types.*
4. *○ is a lex idempotent monad (in fact, an* open modality[4]*).*

Moreover, $\mathbf{U}_{\mathrm{acc}}$ is a model of HoTT and ○ a morphism of models.

We will not reproduce the entire proof for reasons of space, but we will show the argument for why $\mathbf{U}_{\mathrm{acc}}$ is accessible. We highlight this because it relies crucially upon univalence and is one place where working with cubical MTT over ordinary MTT is vital.

Proof. We claim that $\eta : \mathbf{U}_{\mathrm{acc}} \to \bigcirc\mathbf{U}_{\mathrm{acc}}$ is an equivalence. By another result of Rijke et al. [33, Lemma 1.20], it is sufficient to construct a left inverse to η. That is, we must define $r : \bigcirc\mathbf{U}_{\mathrm{acc}} \to \mathbf{U}_{\mathrm{acc}}$ such that for all $A : \mathbf{U}_{\mathrm{acc}}$, $r(\eta\, A) = A$. We define r as follows:

$$r\,\tilde{A} = (z : \mathrm{acc}) \to \tilde{A}\,z$$

To show that this is a left inverse, we must show that if $A : \mathbf{U}_{\mathrm{acc}}$ then $A = \mathrm{acc} \to A$. By univalence, it suffices to show that there is an equivalence $A \simeq \mathrm{acc} \to A$. We conclude by noting that const is such an equivalence because $A : \mathbf{U}_{\mathrm{acc}}$. □

The universe of accessible types $\mathbf{U}_{\mathrm{acc}}$ enjoys a number of additional closure properties, specifically due to the behavior of acc. In particular, acc is closed under both □ and ▶, and we can prove these facts purely within Gatsby without additional assumptions.

Theorem 3.11.

1. *If $A :_\ell \mathbf{U}_{\mathrm{acc}}$ then $\blacktriangleright A$ is accessible.*
2. *If $A :_\delta \mathbf{U}$ then $\langle \delta \mid A\rangle$ is accessible.*

The last statement implies that, in particular, $\square A : \mathbf{U}_{\mathrm{acc}}$.

Proof. We begin with the first statement. Suppose that $A :_\ell \mathbf{U}_{\mathrm{acc}}$. We note that ▶ is a right adjoint modality and employ the transposition equivalence proven by Gratzer [16, Lemma 6.4.2]:

$$\mathrm{acc} \to \blacktriangleright A \simeq \blacktriangleright(\blacktriangleleft\mathrm{acc} \to A)$$

By Theorem 3.7, $\blacktriangleleft\mathrm{acc} = \mathrm{acc}$ and we therefore continue:

$$\blacktriangleright(\blacktriangleleft\mathrm{acc} \to A) \simeq \blacktriangleright(\mathrm{acc} \to A) \simeq \blacktriangleright A$$

The last step follows from Corollary 2.6 applied to $\mathbf{mod}_\ell(\pi_2 A) : \blacktriangleright(A \simeq (\mathrm{acc} \to A))$. This proves that $\blacktriangleright A \simeq (\mathrm{acc} \to \blacktriangleright A)$ and computation shows that the induced equivalence is the constant map.

The second statement is similar, since $\langle \delta \mid - \rangle$ is right adjoint to ϵ_0 and we have already shown that $\langle \epsilon_0 \mid \mathrm{acc}\rangle = \mathbf{Unit}$ (Corollary 3.8):

$$\mathrm{acc} \to \langle \delta \mid A\rangle \simeq \langle \delta \mid \langle \epsilon_0 \mid \mathrm{acc}\rangle \to A\rangle \simeq \langle \delta \mid A\rangle$$

Again, computation shows that the inverse map induced by this chain of equivalences is the constant map. □

Corollary 3.12. *Both $\mathbf{Nat}$ and $\mathbf{Bool}$ are accessible and if $A, B : \mathbf{U}_{\mathrm{acc}}$ then $A + B$ is also accessible.*

Proof. As δ is a left adjoint, $\langle \delta \mid \mathbf{Nat}\rangle = \mathbf{Nat}$ and $\langle \delta \mid \mathbf{Bool}\rangle = \mathbf{Bool}$. The second statement follows by noting that dependent sums and $\mathbf{Bool}$ suffices to define + and $\mathbf{U}_{\mathrm{acc}}$ is closed under both. □

[4]There is an unfortunate terminological clash here; Rijke et al. [33] use the term modality to refer to lex idempotent monads while Gratzer et al. [18] use it to refer to $\langle \mu \mid - \rangle$. For us, modality is meant in the latter sense.

This corollary, which is a direct consequence of Lemma 3.2, is essential for ensuring that $\mathbf{U}_{\mathsf{acc}}$ is usable. Absent this result, $\mathbf{U}_{\mathsf{acc}}$ would still be closed under a large collection of operations but contain no non-trivial base types. However, since essentially all connectives and base types do land in $\mathbf{U}_{\mathsf{acc}}$, we are able to perform all standard operations in mode t and stay within $\mathbf{U}_{\mathsf{acc}}$, without ever mentioning non-accessible types.

For instance, we may construct the type of guarded streams featured in the introduction without explicitly threading acc through the construction. We simply replace $\mathbf{U}$ with $\mathbf{U}_{\mathsf{acc}}$ to ensure that we are applying Löb induction to an accessible type:

$$\mathsf{GStr} = \mathsf{loeb}(\lambda S.\mathbf{Nat} \times \blacktriangleright S)$$

Here we have taken advantage of the fact that $\mathbf{U}_{\mathsf{acc}}$ is closed under both $\mathbf{Nat}$ and $\times$. We have also used $\mathsf{loeb} : ((\ell \mid A) \to A) \to A$ to avoid immediately having to pattern-match upon S in the above example. The propositional unfolding associated with Löb induction also ensures that $\mathsf{GStr} = \mathbf{Nat} \times \blacktriangleright \mathsf{GStr}$.

In fact, we can codify this procedure more generally:

Theorem 3.13 (Completeness)**.** *Any program written in MLTT with* $\blacktriangleright$, $\square$, *and* loeb *with propositional unfolding can be encoded in Gatsby.*

Proof. We must construct a model of such a type theory in Gatsby. We do so by interpreting types as accessible types—thereby interpreting Löb induction. We then use Proposition 3.10, Theorem 3.11, and Corollary 3.12 to interpret all connectives. □

3.3.1 Inaccessible types. Care is required when working with types such as propositional truncation $\|-\|$ which do not preserve accessibility. Indeed, acc itself is not accessible in all models; if it were, then acc would simply be true. However, acc is the propositional truncation of the type $\sum_{n:\mathsf{Nat}} \blacktriangleright^n \mathbf{Void}$ which *is* accessible in all models. Phrased differently, $\|-\|$ does not restrict to a map $\mathbf{U}_{\mathsf{acc}}$ to $\mathbf{U}_{\mathsf{acc}}$.

Consequently, if we were to use $\|A\|$ in a guarded program, it would be necessary to explicitly replace it by the accessible type $\|A\|' = \mathsf{acc} \to \|A\|$. This cannot cause issues within the program itself: mapping out of $\|A\|'$ to an accessible type is the same as mapping out of $\|A\|$. However, after the construction is completed and one wishes to inspect the results as an ordinary type, it is necessary to record that the replacement has taken place. In such situations we must use Corollary 3.8 ourselves. After obtaining a closed term $M : \mathsf{acc} \to \|A\|$, we must choose some natural number n and consider $\mathbf{mod}_{\epsilon_0 \circ e^n}(M) \circledast \star : \langle \epsilon_0 \circ e^n \mid \|A\| \rangle$. Choosing different numbers n enables us to extract different finite approximations of $\|A\|$. This may be important if e.g., $A = \blacktriangleright^k A_0$ so that the first stages of $\|A\|$ are trivial. This corresponds to the idea of type-based fuel introduced by Gratzer and Birkedal [17].

Thus, it is possible, if more complex, to apply guarded reasoning handle types which are not accessible. Fortunately, the standard operations of type theory and guarded recursion ($\square$, $\blacktriangleright$) *do* land in $\mathbf{U}_{\mathsf{acc}}$ so this occurs infrequently.

3.4 First examples in Gatsby

We begin by working through some elementary constructions in guarded type theory in order to give a flavor of working within the system. We begin by filling in the example sketched in Example 1.1:

$$\mathsf{GStr} : \mathbf{U}_{\mathsf{acc}} \to \mathbf{U}_{\mathsf{acc}}$$
$$\mathsf{GStr}\,A = \mathsf{loeb}(S.A \times \blacktriangleright S)$$

$$\mathsf{GStrEq} : (A : \mathbf{U}_{\mathsf{acc}}) \to \mathsf{GStr}\,A = A \times \blacktriangleright \mathsf{GStr}\,A$$
$$\mathsf{GStrEq}\,A = \mathsf{unfold}(S.A \times \blacktriangleright S)$$

We demonstrate how one might carry out small but complete guarded program which first (1) calculates an infinite stream of Fibonacci numbers and then (2) extracts the third number. We begin by defining the stream of numbers using Löb induction:

$$\begin{array}{l}
\mathsf{fibs} : \mathsf{GStr}\,\mathbf{Nat} \\
\mathsf{fibs} = \mathsf{go}\,0\,1 \\
\quad \mathbf{where} \\
\qquad \mathsf{go} : \mathbf{Nat} \to \mathbf{Nat} \to \mathsf{GStr} \\
\qquad \mathsf{go} = \\
\qquad\quad \mathsf{loeb}(f.\,\lambda m, n.\,(\mathsf{GStrEq}\,\mathbf{Nat})_*^{-1}(m, \mathbf{mod}_\ell(f\,n\,(m+n))))
\end{array}$$

Notice that in this example, $(\mathsf{GStrEq}\,A)_*^{-1}$—coercing backwards along the equation $\mathsf{GStr}\,A = A \times \blacktriangleright \mathsf{GStr}\,A$—plays the role of cons. We record this: $\mathsf{cons}_A = (\mathsf{GStrEq}\,A)_*^{-1} : A \times \blacktriangleright \mathsf{GStr}\,A \to \mathsf{GStr}\,A$. Deconstructing a stream uses the inverse coercion i.e., $\mathsf{hd} = \pi_1 \circ (\mathsf{GStrEq}\,A)_*$ and $\mathsf{tl} = \pi_2 \circ (\mathsf{GStrEq}\,A)_* : \mathsf{GStr}\,A \to \blacktriangleright \mathsf{GStr}\,A$.

To extract the third element, we must bring $\square$ into play. In particular, in order to obtain an element of type $\mathbf{Nat}$ rather than $\blacktriangleright^2 \mathbf{Nat}$, we will use the equivalence $\square \blacktriangleright A \simeq \square A$. We note that both $\mathsf{GStr}\,\mathbf{Nat}$ and fibs are closed terms and we may form the following:

$$\begin{array}{l}
\mathsf{fibs}' : \square(\mathsf{GStr}\,\mathbf{Nat}) \\
\mathsf{fibs}' = \mathbf{mod}_{\delta \circ \gamma}(\mathsf{fibs})
\end{array}$$

We then use $\mathsf{now} : \square \blacktriangleright A \simeq \square A$ and $\mathsf{extract} : \square A \to A$ to pull out the third element:

$$\begin{array}{l}
\mathsf{third} : \mathbf{Nat} \\
\mathsf{third} = \mathbf{let}\ \mathbf{mod}_{\delta \circ \gamma}(s) \leftarrow \mathsf{fibs}'\ \mathbf{in}\ \mathsf{extract}(\mathsf{now}(\mathsf{now}(\mathsf{go}\,s))) \\
\quad \mathbf{where} \\
\qquad \mathsf{go} : (\delta \circ \gamma \mid \mathsf{GStr}\,\mathbf{Nat}) \to \square \blacktriangleright \blacktriangleright \mathbf{Nat} \\
\qquad \mathsf{go}\,s = \mathbf{mod}_{\delta \circ \gamma}(\mathsf{next}(\mathsf{next}\,\mathsf{hd}) \circledast (\mathsf{next}\,\mathsf{tl} \circledast \mathsf{tl}\,s))
\end{array}$$

We note that under the assumption of canonicity for Gatsby (Section 7.1), we can compute third and obtain a closed natural number. Thus, in particular, purely through careful use of modalities, we are able to derive Löb induction and use it to compute a closed result. Even without the assumption of canonicity, we are able to use unfold to prove that third = 1.

To cultivate intuition, let us consider a pair of counterfactuals.

Suppose that we had worked within a theory without Rule 1, what would have changed in the above example? The main difference would be in the behavior of $\mathbf{Nat}$: without Rule 1, $\mathbf{Nat}$ would not land in $\mathbf{U}_{\mathsf{acc}}$. We could replace it with $\mathsf{acc} \to \mathbf{Nat}$ and the rest of the calculation would proceed along the same lines. However, the final result would have type $\mathsf{acc} \to \mathbf{Nat}$. Without Theorem 3.7, there would be no way to discharge that hypothesis.

Suppose instead that we considered a type theory without univalence but with a good theory of propositions, such as extensional type theory. In this case, we would not be able to show $\mathbf{U}_{\mathsf{acc}}$ to be accessible. More ad-hoc replacements are possible, for instance $\mathsf{acc} \to \mathbf{U}$, but these alternatives will (1) not form a cumulative hierarchy and (2) not admit an embedding into $\mathbf{U}$ without additional axioms. This hinders the construction of elements of GStr.

4 CASE STUDY WITH LOGICAL RELATIONS

In this section, we present a case-study of Gatsby by using it to construct a (synthetically) step-indexed logical relation for an ML-like language $\lambda_{\mathsf{ref},\forall}$ with general references (pointers to complex structures) and parametric polymorphism. We then use it to deduce semantic type-safety. This $\lambda_{\mathsf{ref},\forall}$ language and the logical relation is based on the account given by Birkedal et al. [7, Section 3] but our more advanced guarded type theory allows us to improve upon their results in two respects:

1. We do not need to worry about local contractibility of domain equations because we have access to a universe and solve the domain equation as an ordinary fixed point [6].
2. We have no need to carry out a challenging external argument after constructing the logical relation because Gatsby is multimodal. Accordingly, we work internally to Gatsby throughout the entire proof.

We have chosen this example to "complete the story" started by Birkedal et al. [7], but many other applications of guarded type theory to denotational semantics and logical relations exist [10, 11, 26, 27, 29, 32, 38, 39, 41]. Many of these applications can also be simplified in light of Gatsby's richer modal apparatus.

For reasons of space, we have included only selected details of this case study. In particular, where there is little difference from working in Gatsby versus the framework of Birkedal et al. [7], we have avoided duplicating their work.

Convention 4.1. In order to construct a logical relation for parametric polymorphism, we require an impredicative universe of propositions. Accordingly, in this section we assume propositional resizing i.e., that the maps $\mathsf{hProp}_i \to \mathsf{hProp}_{i+1}$ are equivalences.

4.1 Static and dynamic semantics of $\lambda_{\mathsf{ref},\forall}$

We begin by defining $\lambda_{\mathsf{ref},\forall}$, the language under consideration. We specify this language in mode s, i.e., with no guarded recursion whatever. As we shall argue in Section 5, this ensures that our definition of $\lambda_{\mathsf{ref},\forall}$ adequately represents the standard definition that one might formalize in Coq or similar.

We define the syntax of (untyped) terms and types as inductive types. We then define term and type contexts (TCxt, Cxt) and heaps (Heap) on top of these. The typing judgments as well as the operational semantics are realized by inductively-defined propositions:

$$\begin{aligned}
&\mathsf{isCx} : \mathsf{TCxt} \to \mathsf{Cxt} \to \mathsf{hProp}\\
&\mathsf{isTy} : \mathsf{TCxt} \to \mathsf{Ty} \to \mathsf{hProp}\\
&\mathsf{hasTy} : (\Xi : \mathsf{TCxt}) \to \mathsf{Cxt}^{\mathsf{wf}}\,\Xi \to \mathsf{Tm} \to \mathsf{Ty}^{\mathsf{wf}}\,\Xi \to \mathsf{hProp}\\
&\mathsf{isVal} : \mathsf{Tm} \to \mathsf{hProp}\\
&(\mapsto) : \mathsf{Tm} \times \mathsf{Heap} \to \mathsf{Tm} \times \mathsf{Heap} \to \mathsf{hProp}
\end{aligned}$$

We sketch all of these definitions in Fig. 2, using informal BNF grammars to specify the syntax of terms and types and inference rules for the judgments. We leave implicit many details as they are orthogonal to our case study and refer the reader to Birkedal et al. [7]. We have also assumed the existence of various standard operations on lists, such as snoc, lookup, etc. We do note that we formalize heaps as a list of values and that allocation is deterministic. Moreover, primitive locations $\mathtt{loc}(\ell)$ are never well-typed; they arise only in intermediate stages of execution for programs and are never written by a user.

$$\begin{aligned}
\tau : \mathsf{Ty} &\quad ::= \quad \mathtt{tvar}(i) \mid \mathtt{Ref}(\tau) \mid \mathtt{forall}(\tau) \mid \tau \times \tau \mid \ldots\\
e : \mathsf{Tm} &\quad ::= \quad \mathtt{var}(i) \mid {!e} \mid \mathtt{new}(e) \mid \mathtt{set}(e,e) \mid \Lambda e \mid e[\tau] \mid \ldots
\end{aligned}$$

$$\mathsf{TCxt} = \mathbf{Nat} \qquad \mathsf{Cxt} = \mathsf{List}\,\mathsf{Ty} \qquad \mathsf{Cxt}^{\mathsf{wf}}\,\Xi = \textstyle\sum_{\Gamma:\mathsf{Cxt}} \mathsf{isCxt}\,\Xi\,\Gamma$$

$$\mathsf{Ty}^{\mathsf{wf}}\,\Xi = \textstyle\sum_{\tau:\mathsf{Ty}} \mathsf{isTy}\,\Xi\,\tau \qquad \mathsf{Val} = \textstyle\sum_{e:\mathsf{Tm}} \mathsf{isVal}\,e \qquad \mathsf{Heap} = \mathsf{List}\,\mathsf{Val}$$

$$\frac{\mathsf{isTm}(\Xi,\Gamma,e,\tau)}{\mathsf{isTm}(\Xi,\Gamma,\mathtt{new}(e),\mathtt{Ref}(\tau))} \qquad \frac{\mathsf{isTm}(\Xi,\Gamma,e,\mathtt{Ref}(\tau))}{\mathsf{isTm}(\Xi,\Gamma,!e,\tau)}$$

$$\frac{\mathsf{isTm}(\Xi,\Gamma,e_1,\mathtt{Ref}(\tau)) \qquad \mathsf{isTm}(\Xi,\Gamma,e_2,\tau)}{\mathsf{isTm}(\Xi,\Gamma,\mathtt{set}(e_1,e_2),\tau)}$$

$$\begin{aligned}
(\mathtt{new}(v),h) &\mapsto (\mathtt{loc}(\mathsf{len}\,h),\mathsf{snoc}(h,v))\\
(!\mathtt{loc}(\ell),h) &\mapsto (\mathsf{lookup}(h,\ell),h) && (\ell < \mathsf{len}\,h)\\
(\mathtt{set}(\mathtt{loc}(\ell),v),h) &\mapsto (v,\mathsf{replace}(h,\ell,v)) && (\ell < \mathsf{len}\,h)
\end{aligned}$$

Figure 2: Selected rules of $\lambda_{\mathsf{ref},\forall}$

We will write $\mapsto^*$ for the reflexive transitive closure of $\mapsto$. As in Birkedal et al. [7], we have opted to present the operational semantics with deterministic allocation as this simplifies a number of technical details. We shall write $\mathsf{Good} : \mathsf{Tm} \times \mathsf{Heap} \to \mathsf{hProp}$ for the following:

$$\begin{aligned}
&\mathsf{canStep}(e,h) = \exists e',h'.\,(e,h) \mapsto (e',h')\\
&\mathsf{Good}\,e\,h = \forall (e',h').\,((e,h) \mapsto^* (e',h')) \to\\
&\quad \mathsf{isVal}\,e' \vee \mathsf{canStep}(e',h')
\end{aligned}$$

In other words, a configuration consisting of an expression and a heap is good if one can execute the pair for an arbitrary number of steps and the resulting expression and heap has either resulted in a value or can be run further. Our goal for the remainder of this section is to prove the following version of type-safety.

Theorem 4.2 (Safety of $\lambda_{\mathsf{ref},\forall}$). *If* $\mathsf{isTm}\,0\,\mathsf{nil}\,e\,\tau$ *then* $\mathsf{Good}\,e\,\mathsf{nil}$

We note that the presence of $\mathtt{loc}(\ell)$ makes this theorem difficult to prove: the intermediate stages are not typed and so progress and preservation does not apply.

4.2 A unary logical relation interpretation

In order to prove Theorem 4.2, we will construct a model of $\lambda_{\mathsf{ref},\forall}$ which interprets $\lambda_{\mathsf{ref},\forall}$ types as elements of a construction in mode t. Morally, the construction is a Kripke logical relation, but the presence of higher-order references necessitates a rich and highly-recursive type of worlds. It is for this reason that we must pass to mode t, where Löb induction is available; we shall use it to define the semantic universe of types alongside the worlds indexing them.

$$\begin{aligned}
&\mathcal{P}\,A = A \to \mathsf{hProp}_{\mathsf{acc}}\\
&\mathsf{W} : \mathbf{U}_{\mathsf{acc}} \qquad (\sqsubseteq) : \mathsf{W} \to \mathsf{W} \to \mathsf{hProp}_{\mathsf{acc}}\\
&(\mathsf{W},\sqsubseteq) = \mathsf{loeb}(A,\leq.\,(\mathsf{X}(A,\leq),\mathsf{R}(A,\leq)))\\
&\quad \textbf{where}\\
&\qquad \mathsf{X}\,(A,\leq) = \mathbf{Nat} \rightharpoonup_{\mathsf{fin}} ((\blacktriangleright A,\leq^{\dagger}) \to_{\mathsf{mon}} (\mathcal{P}\langle\delta \mid \mathsf{Val}\rangle,\subseteq))\\
&\qquad \mathsf{R}\,(A,\leq)\,w_1\,w_2 = \forall \ell.w_1(\ell)\!\downarrow\, \to w_1\ell = w_2\ell\\
&\mathsf{T} : \mathbf{U}_{\mathsf{acc}}
\end{aligned}$$

$\mathsf{T} = \mathsf{W} \to_{\mathsf{mon}} \mathcal{P}\langle \delta \mid \mathsf{Val}\rangle$

Notation 4.3. We follow Gratzer [16] in writing $\mathsf{foo}^\dagger$ to lift a function or relation foo to apply to a sequence of modal arguments similar to *idiom brackets* for applicative functors [25].

A few words of discussion are in order. First, $\to_{\mathsf{mon}}$ refers to the subtype of monotone maps where $\mathcal{P}\langle \delta \mid \mathsf{Val}\rangle$ is ordered by subset inclusion. To use W in the domain of $\to_{\mathsf{mon}}$, we must define W simultaneously with its ordering relation $\sqsubseteq$. For this reason, we use Löb induction to compute an element of $\sum_{A:\mathbf{U}_{\mathsf{acc}}} A \to A \to \mathsf{hProp}_{\mathsf{acc}}$ rather than merely $\mathbf{U}_{\mathsf{acc}}$. Second, $\rightharpoonup_{\mathsf{fin}}$ refers to finitely-supported partial maps. We note that this procedure is simpler than its cousin in Birkedal et al. [7]. In particular, we have access to a universe and therefore have no need to use an external construction.

The remaining details of the unary logical relation proceed along similar lines to Birkedal et al. [7], though we are able to simplify several definitions by continuing to exploit the internal language. We define a path WEq witnessing the unfolding of W to its definition using unfold. Next, we note that since $\langle \delta \mid - \rangle$ is a left adjoint, we have a crisp induction principle available and we use this to define a map $[\![-]\!]_{\_} : \langle \delta \mid \mathsf{Ty}^{\mathsf{wf}}\,\Xi\rangle \to (\sum_{n \le \Xi} \mathsf{T}) \to \mathsf{T}$ for each $\Xi : \langle \delta \mid \mathsf{TCxt}\rangle = \langle \delta \mid \mathbf{Nat}\rangle = \mathbf{Nat}$. This map sends a type with Ξ free variables to a map from Ξ semantic types to a single semantic type. The details of $[\![-]\!]_{\_}$ are not vital, but the following auxiliary definitions will be important later:

$$
\begin{aligned}
&\mathsf{sat}\; w\; h = \\
&\quad \mathsf{dom}(h) \subseteq \mathsf{dom}(w) \\
&\quad \wedge \forall \ell \in \mathsf{dom}(w).\mathsf{WEq}_{*}\, w\, \ell\, w\, (\mathsf{lookup}^\dagger\; h\; \ell)
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{comp} : \mathsf{T} \to \mathsf{W} \to \mathcal{P}\langle \delta \mid \mathsf{Exp}\rangle \\
&\mathsf{comp}\; \phi\; w\; e = \\
&\quad (\mathsf{isVal}^\dagger\; e \wedge \phi\; w\; e) \\
&\quad \vee \forall h : \langle \delta \mid \mathsf{Heap}\rangle.\, \mathsf{sat}\; w\; h \to \\
&\qquad \exists e' h' w'.\, (e,h) \mapsto^\dagger (e',h') \wedge w \sqsubseteq w' \wedge \mathsf{sat}\; w'\; h' \\
&\qquad\quad \wedge \blacktriangleright \mathsf{comp}\; \phi\; e'\; w'
\end{aligned}
$$

Informally, comp lifts a semantic type ϕ to a predicate on expressions which evaluates the expression and, if it ever reaches a value, insists the result satisfies ϕ.[5] The following is proven as in Birkedal et al. [7].

Lemma 4.4 (Fundamental lemma). *If* $\mathsf{isTm}^\dagger\; 0\; \mathsf{nil}\; e\; \tau$ *holds in mode* t *then so does* $\mathsf{comp}\; [\![\tau]\!]\; (\mathsf{WEq}_{*}\emptyset)\; e$.

4.3 Adequacy

At this point, we substantially deviate from Birkedal et al. [7]. It remains to argue that Lemma 4.4 implies Theorem 4.2. In op. cit., the authors were forced to unfold various definitions externally and argue that Lemma 4.4 externally implied the desired type-safety result. Our richer modal apparatus allows us to proceed internally. First, we note that Lemma 4.4 can be placed under $\langle \gamma \mid - \rangle$ (it is closed). Fixing e and τ in mode s, general properties of modalities yield the following implication:

$$
\begin{aligned}
&\langle \gamma \mid \mathsf{isTm}^\dagger\; 0\; \mathsf{nil}\; \mathbf{mod}_\delta(e)\; \mathbf{mod}_\delta(\tau)\rangle \\
&\quad \to \langle \gamma \mid \mathsf{comp}\; [\![\mathbf{mod}_\delta(\tau)]\!]\; (\mathsf{WEq}_{*}\emptyset)\; \mathbf{mod}_\delta(e)\rangle
\end{aligned}
$$

[5] We note that we did not need to define comp using Löb induction, though we certainly could have. Its definition is positive, however, and so Tarski's fixed-point theorem applies. The unicity of guarded fixed points, morover, ensures both approaches agree.

Inspecting the definitions, we note that the domain of this function is equal to $\mathsf{isTm}\; 0\; \mathsf{nil}\; e\; \tau$. It therefore remains only to show that the codomain implies $\mathsf{Good}\; e\; \mathsf{nil}$. In fact, in our case we do not need to worry about the actual properties of the value e runs to, so we will consider $\langle \gamma \mid \mathsf{comp}\; (\lambda\_.\,\mathbf{Unit})\; (\mathsf{WEq}_{*}\emptyset)\; \mathbf{mod}_\delta(e)\rangle$ instead.

We prove the following helper lemma in mode t.

Lemma 4.5. *If* $\mathsf{comp}\; (\lambda\_.\,\mathbf{Unit})\; w\; e$, *fix an* $h : \langle \delta \mid \mathsf{Heap}\rangle$ *such that* $\mathsf{sat}\; w\; h$ *and, moreover, if* $(e,h)\; (\mapsto^n)^\dagger\; (e',h')$ *then* $\blacktriangleright^n(\mathsf{isVal}^\dagger\; e' \vee \mathsf{canStep}^\dagger(e',h'))$.

Proof. We proceed by induction on n—which we can due to crisp induction. The base case where $(e,h) = (e',h')$ is trivial. For the inductive step, we may apply the induction hypothesis to reduce to the case that $(e,h) \mapsto (e',h')$. Noting that e must not be a value, we deduce that the following holds:

$$\exists e_1 h_1 w'.\, (e,h) \mapsto^\dagger (e_1,h_1) \wedge w \sqsubseteq w' \wedge \mathsf{sat}\; w'\; h_1 \wedge \blacktriangleright \mathsf{comp}\; \phi\; e_1\; w'$$

As our operational semantics are deterministic, we may replace e_1 and h_1 with e' and h' and simplify the above:

$$\exists w' \sqsupseteq w.\, \mathsf{sat}\; w'\; h' \wedge \blacktriangleright \mathsf{comp}\; \phi\; e'\; w'$$

The goal then follows by unfolding the definition of comp. □

We may now return to mode s to complete the theorem.

Theorem 4.2 (Safety of $\lambda_{\mathsf{ref},\forall}$). *If* $\mathsf{isTm}\; 0\; \mathsf{nil}\; e\; \tau$ *then* $\mathsf{Good}\; e\; \mathsf{nil}$

Proof. Fix e', h', n such that $(e,h) \mapsto^n (e',h')$. We must show that either $\mathsf{isVal}\; e'$ or $\mathsf{canStep}(e',h')$. By Lemma 4.4, we know that $\langle \gamma \mid \mathsf{comp}\; (\lambda\_.\,\mathbf{Unit})\; w\; \mathbf{mod}_\delta(e)\rangle$ holds. Using Lemma 4.5, we therefore obtain the following:

$$\langle \gamma \mid \blacktriangleright^n(\mathsf{isVal}^\dagger\; e' \vee \mathsf{canStep}^\dagger(e',h'))\rangle$$

Using the equation $\gamma \circ \ell = \gamma$, we may replace this:

$$\langle \gamma \mid (\mathsf{isVal}^\dagger\; e' \vee \mathsf{canStep}^\dagger(e',h'))\rangle$$

$\langle \gamma \mid - \rangle$ does not commute with disjunctions. However, $\langle \delta \mid - \rangle$ does commute with disjunctions as it is a left adjoint. We then replace the type under $\langle \gamma \mid - \rangle$ with $\langle \delta \mid \mathsf{isVal}\; e' \vee \mathsf{canStep}(e',h')\rangle$. We conclude using the equation $\gamma \circ \delta = \mathsf{id}$. □

We note that such a proof is impossible without $\langle \gamma \mid - \rangle$ or something equivalent to it; without such a feature we would have no means to remove the $\blacktriangleright^n$ appearing in Lemma 4.5.

5 SEMANTICS OF Gatsby

Thus far we have shown that Gatsby is usable but we not yet shown it to be sound. In this section, we prove this, among other results, by developing the model theory of Gatsby. This is seemingly daunting: constructing models of cubical type theory is already a challenging task. Fortunately Gatsby is built atop cubical MTT, which already has a well-developed denotational semantics [1].

At a high level, we wish to interpret mode t as $\mathbf{PSh}(\omega)$ and s as $\mathbf{Set}$, but this will not serve if we wish to interpret univalence. Instead, we must replace $\mathbf{Set}$ with the category of cubical sets $\mathbf{cSet}$. Beyond this change, we are able to interpret the modalities as right adjoints between these two categories without change. We thereby obtain a pseudofunctor $F : \mathcal{M}_{\mathsf{Gatsby}} \to \mathbf{Cat}$ which sends

$F(s) = \mathbf{cSet}$ and $F(t) = \mathbf{PSh}_{\mathbf{cSet}}(\omega)$ (cubical presheaves on ω). We show the definitions of F on the generating 1-cells below:

$$F(\ell)\,X\,n = \text{if } n = 0 \text{ then } \mathbf{1} \text{ else } X(n-1) \qquad F(e)\,X = X(n+1)$$

$$F(\gamma)\,X = \lim_{\omega} X \qquad F(\epsilon_0)\,X = X(0) \qquad F(\delta)\,S\,n = S \qquad F(\top)\,S = \mathbf{1}$$

In general, cubical presheaves $\mathbf{PSh}_{\mathbf{cSet}}(C)$ (presheaves valued in cubical sets) support a model of cubical type theory [22]. The following result of Aagaard et al. [1] shows that these models of cubical type theory can be combined into a model of cubical MTT:

Proposition 5.1 (Theorem 4.15 [1]). *Fix a strict 2-functor $f : \mathcal{M} \longrightarrow \mathbf{Cat}$ and for each $\mu : n \longrightarrow m$, write $F^*(\mu) \dashv F_*(\mu)$ for the adjunction between $\mathbf{PSh}_{\mathbf{cSet}}(f(n))$ and $\mathbf{PSh}_{\mathbf{cSet}}(f(m))$ induced by precomposition and right Kan extension. There is a model of cubical MTT with mode theory $\mathcal{M}$ which interprets $\langle \mu \mid - \rangle$ as $F_*(\mu)$.*

Ideally, we would instantiate this theorem by taking $\mathcal{M} = \mathcal{M}_{\text{Gatsby}}$ and define f in such a way that F is induced by right Kan extending f. Unfortunately, not every functor described by F should be interpreted using right Kan extension. In particular, ϵ_0 and $\top$ do not arise in this way. Fortunately, $F(\epsilon_0)$ and $F(\top)$ "almost" arise from right Kan extension: their left adjoints preserve connected limits. Moreover, the pseudofunctor induced by taking left adjoints to F is a strict 2-functor. This, along with similar strictness properties of F, ensures that the coherence conditions required for a model of cubical MTT are trivially satisfied.

We can therefore extend Proposition 5.1:

Lemma 5.2. *There exists a model of cubical MTT with mode theory $\mathcal{M}_{\text{Gatsby}}$ which interprets $\langle \mu \mid - \rangle$ using $F(\mu)$.*

The details of the proof of Lemma 5.2 require some knowledge of cubical MTT and we have therefore deferred them to Appendix A.

Theorem 5.3. *F supports a model of Gatsby.*

Proof. Lemma 5.2 almost suffices, but we must interpret Rule 1. However, since $\langle \top \mid - \rangle$ is interpreted using $\_ \mapsto \mathbf{1}$, we must have $[\![\mathbf{1}.\{\top\}]\!] = \mathbf{0}$. We note that $\mathbf{cSet}$ is a topos and so $\mathbf{0}$ is a strict initial object. It therefore follows that any object X for which there is a map $X \longrightarrow \mathbf{0}$ must itself be the initial object. The interpretation of Rule 1 is then immediate; it is the universal property of $\mathbf{0}$. □

Remark 5.4. The same proof strategy suffices to interpret Gatsby into cubical presheaves on an arbitrary limit ordinal.

Corollary 5.5. *Gatsby is consistent.*

Beyond merely ensuring consistency, interpreting mode s as into the standard model of cubical type theory, ensures a degree of *adequacy* for constructions carried out in Gatsby. In particular, Theorem 5.3 shows that any construction in mode s induces a construction element in a model already accepted by cubical type theorists. Consequently, there is no need to care about e.g. the topos of trees or modalities when evaluating the content of Theorem 4.2.

6 RELATED WORK

While many variations on guarded type theory have been proposed [3, 4, 7, 11, 13, 28], these failed to meet at least one of the four goals raised in Thesis 1. Only two prior type theories offer a reasonably complete solution: stratified guarded type theory [17] and clocked cubical type theory [21]. We discussed both in Section 1.3 and we now sharpen our prior comparison.

Stratified guarded type theory. Recall from Section 1.3 that stratified guarded type theory is actually a pair of type theories: one in which Löb computes and one in which it does not. In the type theory where Löb induction computes, Gratzer and Birkedal [17] introduce a notion of guarded canonicity where canonical forms are judged in a special context $\mathbf{0}[\ell^n]$. All terms trivialize when $\mathbf{0}[\ell^n]$ is placed under $\{\ell^n\}$ and so the canonicity result enables one to extract a finite approximation to an infinite canonical form. In Gatsby, $\mathbf{0}[\ell^n]$ can be *defined* as $\mathbf{1}.\{\epsilon_0 \circ e^n\}$. Guarded canonicity becomes a special case of ordinary canonicity.

For example, if we assume acc to prove $M : A$, we may place M under $\langle \epsilon_0 \circ e^n \mid - \rangle$ to discharge acc and obtain an element of $\langle \epsilon_0 \circ e^n \mid A \rangle$. Just as in stratified guarded type theory, we are able to extract information about A from this term but we are only able to descend "beneath n iterations of ▶". In this way, Gatsby takes the idea of guarded canonicity in stratified guarded type theory and recasts it as a modal discipline. The result is an internalization of guarded canonicity as normal canonicity and, moreover, Gatsby does not rule out normalization in the process.

We have also isolated the universe of accessible types where there is a canonical and optimal choice of fuel and used this to avoid requiring the user to choose a fuel supply each time they wish to calculate a result. Thus, by enriching type theory with modalities and Rule 1, we are able to essentially recover stratified guarded type theory without splitting our theory into two.

Clocked cubical type theory. As described in Section 1.3, clocked cubical type theory (CloTT$_\square$) is an alternative approach to guarded recursion built around indexing ▶ by a clock. In this way, CloTT$_\square$ allows Löb induction to compute only when the clock indexing the relevant ▶ modality has been bound. This essentially limits computation to occurring at the top-level and thereby conjecturally preserves canonicity and normalization. A more substantial difference between CloTT$_\square$ and Gatsby is in the approach they take to Löb induction. In CloTT$_\square$—and all other proposed guarded type theories—Löb induction is a primitive while in Gatsby it is derived.

At a high-level, Gatsby provides a richer set of modalities and a simpler semantics, but does not support multi-clock guarded recursion. This means that Gatsby, unlike CloTT$_\square$, can internally express notions such as "constant types" but cannot directly encode a coinductive stream of non-constant types. Despite these differences, both theories conjecturally satisfy the goals of Thesis 1 and so both provide adequate foundations for guarded recursion.

Interestingly, just as our approach to Löb induction necessitates consideration of *accessible types*, the use of clocks in CloTT$_\square$ requires users to frequently restrict to *clock-irrelevant types*. Roughly, these are types which are "clock-null". However, accessible types form a better-behaved class than clock-irrelevant types; accessible types form an open reflective subuniverse [33]. Consequently, we are not only able to show important type operations respect accessibility but also prove that the universe of accessible types is accessible. We are even able to replace a non-accessible type by a universal accessible counterpart. This machinery is not available for clock-irrelevant types; the sort of clocks is not presented as a type but, more fundamentally, it is not a homotopy proposition. Consequently, for instance, the question of whether or not a suitable clock-irrelevant universe of clock-irrelevant types within $\mathsf{CloTT}_{\square}$ [12] remains open.

Other occurrences of accessibility. Finally, we note that variants of the accessibility proposition acc have appeared before in the literature. In Palombi and Sterling [31], for instance, it is used to isolate the universal property of $\mathbf{PSh}(\omega)$ as a model of guarded recursion. That $\exists n. \blacktriangleright^n \bot$ holds in $\mathbf{PSh}(\omega)$ is also an important motivation for *transfinite Iris* [36] which uses higher-ordinal models of guarded recursion precisely to *avoid* having acc $= \top$ hold. Amin Timany has further proposed taking adding the axiom acc $= \top$ to Iris, as a more intuitive but equivalent formulation of Löb induction [9].

7 CONCLUSIONS AND FUTURE WORK

We have presented Gatsby,[6] a univalent multimodal type theory based on cubical MTT. Proceeding from the observation that Löb induction is an ill-behaved primitive for guarded recursion, Gatsby uses additional modalities to essentially *derive* Löb induction.

Concretely, we isolate a homotopy proposition acc which suffices to imply Löb induction. We then show that the collection of accessible types A—those which are acc-null and therefore support Löb induction—is closed under numerous standard constructions. Gatsby also constrains the modality $\langle \top \mid - \rangle$ to be equivalent to $A \mapsto \mathbf{Unit}$ and this ensures that all constant types are accessible. Using these results, we show that it is possible to encode any program in a standard guarded type theory within Gatsby. We have further exploited Gatsby's rich multimodal structure to improve upon case studies considered already in guarded recursion.

Gatsby constitutes a canonical point in the space of guarded type theories satisfying Thesis 1 as it captures much of the behavior of the important model of guarded recursion in $\mathbf{PSh}(\omega)$ while still maintaining a well-behaved metatheory.

We summarize several directions for future work below.

7.1 Normalization and canonicity for Gatsby

At present, we have not proven that Gatsby enjoys either normalization or canonicity. We conjecture that it in fact enjoys both. We offer preliminary evidence in support of this conclusion.

We note that Gatsby is built on top of cubical MTT and we expect to be able to adapt a proof of normalization and canonicity for the latter to apply to the former. At present no such proof for cubical MTT exists, so we begin by discussing prospects for this result.

Cubical MTT is a fusion of two type theories, MTT and cubical type theory, which both enjoy canonicity and normalization [15, 37]. Normalization and canonicity are not modular properties, so this does not necessarily mean that cubical MTT enjoys either. However, given that there are no meaningful interactions between the two theories in cubical MTT, we expect both to hold.

In order to adapt such a proof of normalization and canonicity for cubical MTT to apply to Gatsby, we must show that Rule 1 does not introduce stuck terms and does not disrupt the decidability of normal forms. We expect the techniques used by Sterling and Angiuli [37] to handle the false cofibration should suffice for our situation. In particular, we can show that the crucial lemma of op. cit. stating that it is decidable whether a given context proves the false cofibration can be adapted to Gatsby. That is, it is decidable whether or not there exists a substitution from Γ to $\mathbf{1}.\{\top\}$; it is equivalent to whether one of the following two conditions hold (1) Γ proves the false cofibration or (2) the composite of the modalities in Γ contains $\top$. We give a version of this theorem below which deals with MTT extended by Rule 1 rather than cubical MTT, as the latter involves essentially unrelated details of cubical type theory.

Proposition 7.1. *A substitution $\Gamma \vdash r : \mathbf{1}.\{\top\}$ @ s exists if and only if the composite of all modalities within Γ is $\nu \circ \top$ for some ν.*

Proof Sketch. We do not present the full details of the proof because it requires a more thorough explanation of the substitution calculus of MTT and Gatsby. We begin by noting that there is a trivial model of Gatsby in which every type is interpreted by **Unit**. The category of contexts of this model is a reflective subcategory of the ordinary syntactic category of contexts and substitutions. The reflection sends Γ to a new context $|\Gamma|$ obtained by weakening away all variables in Γ so $|\Gamma|$ is of the form $\mathbf{1}.\{\mu\}$ for some μ.

As $\mathbf{1}.\{\top\}$ lies within this subcategory already, a substitution from Γ to $\mathbf{1}.\{\top\}$ exists just when one there is one from $|\Gamma| = \mathbf{1}.\{\mu\}$ to $\mathbf{1}.\{\top\}$. An inductive argument shows that this occurs just when $\mu \geq \nu \circ \top$ for some ν. However, if $\mu \geq \nu \circ \top$ then $\mu = \nu \circ \top$. □

While the above observations give strong support to our conjecture, normalization and canonicity proofs for any type theory are complex and cubical MTT and Gatsby are both sophisticated type theories. We therefore leave the normalization and canonicity of Gatsby to future work.

7.2 Extensions to Gatsby

Aside from extending our knowledge of the metatheory of Gatsby, we hope to study the behavior of other concepts from univalent foundations within this framework. In particular, it remains to isolate which *higher inductive types (HITs)* naturally land within the universe of accessible types. The work on HITs within the context of $\mathsf{CloTT}_{\square}$ [21] suggests that this may hold for a broad class, though Section 3.3.1 demonstrates that the situation is subtle. That accessible types form a reflective subuniverse does mean that while this may improve convenience, it is not as vital as the corresponding question for clock-irrelevance.

We have focused on capturing the behavior of guarded recursion within the topos of trees and, in particular, indexing over ω. In the future, we intend to explore whether the idea of isolating accessible types can be adapted to account for indexing over higher ordinals with the goal of modeling $\mathbf{PSh}_{\mathbf{cSet}}(\alpha)$ for at least countable α.

Finally, we intend to explore the behavior of Gatsby further by implementing it. Both MTT and cubical type theories have been implemented in proof assistants and, as discussed in Section 7.1, such implementations should be possible to extend to cubical MTT and Gatsby. Such an implementation would provide a better setting to explore what definitional equalities are possible to achieve in Gatsby, as checking such calculations is subtle and error-prone.

[6]The authors defer to the reader on whether or not Gatsby is great.

REFERENCES

[1] Frederik Lerbjerg Aagaard, Magnus Baunsgaard Kristensen, Daniel Gratzer, and Lars Birkedal. 2022. Unifying cubical and multimodal type theory. arXiv:2203.13000 [cs.LO]

[2] A. Arnold and M. Nivat. 1980. Metric interpretations of infinite trees and semantics of non deterministic recursive programs. *Theoretical Computer Science* 11, 2 (1980), 181–205. https://doi.org/10.1016/0304-3975(80)90045-6

[3] Patrick Bahr, Hans Bugge Grathwohl, and Rasmus Ejlers Møgelberg. 2017. The clocks are ticking: No more delays!. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. https://doi.org/10.1109/LICS.2017.8005097

[4] Lars Birkedal, Aleš Bizjak, Ranald Clouston, Hans Bugge Grathwohl, Bas Spitters, and Andrea Vezzosi. 2019. Guarded Cubical Type Theory. *Journal of Automated Reasoning* 63 (2019), 211–253.

[5] Lars Birkedal, Ranald Clouston, Bassel Mannaa, Rasmus Ejlers Møgelberg, Andrew M. Pitts, and Bas Spitters. 2020. Modal dependent type theory and dependent right adjoints. *Mathematical Structures in Computer Science* 30, 2 (2020), 118–138. https://doi.org/10.1017/S0960129519000197 arXiv:1804.05236

[6] Lars Birkedal and Rasmus Ejlers Møgelberg. 2013. Intensional Type Theory with Guarded Recursive Types qua Fixed Points on Universes. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '13)*. IEEE Computer Society, USA, 213–222. https://doi.org/10.1109/LICS.2013.27

[7] Lars Birkedal, Rasmus Møgelberg, Jan Schwinghammer, and Kristian Støvring. 2012. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *Logical Methods in Computer Science* 8, 4 (2012). https://doi.org/10.2168/LMCS-8(4:1)2012

[8] Lars Birkedal, Kristian Støvring, and Jacob Thamsborg. 2010. The Category-Theoretic Solution of Recursive Metric-Space Equations. *Theoretical Computer Science* 411, 47 (10 2010), 4102–4122. https://doi.org/10.1016/j.tcs.2010.07.010

[9] Aleš Bizjak and Lars Birkedal. 2022. Lecture Notes on Iris: Higher-Order Concurrent Separation Logic. Online. https://iris-project.org/tutorial-pdfs/iris-lecture-notes.pdf.

[10] Ales Bizjak, Lars Birkedal, and Marino Miculan. 2014. A Model of Countable Nondeterminism in Guarded Type Theory. In *Rewriting and Typed Lambda Calculi – Joint International Conference, RTA-TLCA 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8560)*, Gilles Dowek (Ed.). Springer, 108–123. https://doi.org/10.1007/978-3-319-08918-8_8

[11] Aleš Bizjak, Hans Bugge Grathwohl, Ranald Clouston, Rasmus E. Møgelberg, and Lars Birkedal. 2016. Guarded Dependent Type Theory with Coinductive Types. In *Foundations of Software Science and Computation Structures*, Bart Jacobs and Christof Löding (Eds.). Springer Berlin Heidelberg, 20–35.

[12] Aleš Bizjak and Rasmus Ejlers Møgelberg. 2020. Denotational semantics for guarded dependent type theory. *Mathematical Structures in Computer Science* 30, 4 (2020), 342–378. https://doi.org/10.1017/S0960129520000080

[13] Ranald Clouston, Aleš Bizjak, Hans Bugge Grathwohl, and Lars Birkedal. 2015. Programming and Reasoning with Guarded Recursion for Coinductive Types. In *Foundations of Software Science and Computation Structures* (Berlin, Heidelberg), Andrew Pitts (Ed.). Springer Berlin Heidelberg, 407–421.

[14] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2017. Cubical Type Theory: a constructive interpretation of the univalence axiom. 4, 10 (2017), 3127–3169. arXiv:1611.02108 [cs.LO]

[15] Daniel Gratzer. 2022. Normalization for Multimodal Type Theory. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science* (Haifa, Israel) *(LICS '22)*. Association for Computing Machinery, New York, NY, USA, Article 2, 13 pages. https://doi.org/10.1145/3531130.3532398

[16] Daniel Gratzer. 2023. *Syntax and semantics of modal type theory*. Ph. D. Dissertation. Aarhus University.

[17] Daniel Gratzer and Lars Birkedal. 2022. A Stratified Approach to Löb Induction. In *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)* (Dagstuhl, Germany) *(Leibniz International Proceedings in Informatics (LIPIcs), Vol. 228)*, Amy Felty (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. https://doi.org/10.4230/LIPIcs.FSCD.2022.3

[18] Daniel Gratzer, G.A. Kavvos, Andreas Nuyts, and Lars Birkedal. 2020. Multimodal Dependent Type Theory. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20)*. ACM. https://doi.org/10.1145/3373718.3394736

[19] Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal. 2021. Multimodal Dependent Type Theory. *Logical Methods in Computer Science* Volume 17, Issue 3 (07 2021). https://doi.org/10.46298/lmcs-17(3:11)2021

[20] Astra Kolomatskaia and Michael Shulman. 2023. Displayed Type Theory and Semi-Simplicial Types. arXiv:2311.18781

[21] Magnus Baunsgaard Kristensen, Rasmus Ejlers Møgelberg, and Andrea Vezzosi. 2022. Greatest HITs: Higher inductive types in coinductive definitions via induction under clocks. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3531130.3533359

[22] Daniel R. Licata, Ian Orton, Andrew M. Pitts, and Bas Spitters. 2018. Internal Universes in Models of Homotopy Type Theory. In *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018) (Leibniz International Proceedings in Informatics (LIPIcs))*, H. Kirchner (Ed.). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 22:1–22:17. https://doi.org/10.4230/LIPIcs.FSCD.2018.22 arXiv:1801.07664

[23] Daniel R. Licata and Michael Shulman. 2016. Adjoint Logic with a 2-Category of Modes. In *Logical Foundations of Computer Science*, Sergei Artemov and Anil Nerode (Eds.). Springer International Publishing, 219–235. https://doi.org/10.1007/978-3-319-27683-0_16

[24] Bassel Mannaa, Rasmus Ejlers Møgelberg, and Niccolò Veltri. 2020. Ticking clocks as dependent right adjoints: Denotational semantics for clocked type theory. *Logical Methods in Computer Science* Volume 16, Issue 4 (12 2020). https://doi.org/10.23638/LMCS-16(4:17)2020

[25] Conor McBride and Ross Paterson. 2008. Applicative programming with effects. *Journal of Functional Programming* 18, 1 (2008). https://doi.org/10.1017/S0956796807006326

[26] Rasmus Ejlers Møgelberg and Marco Paviotti. 2016. Denotational semantics of recursive types in synthetic guarded domain theory. In *LICS '16 Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*. Association for Computing Machinery, United States, 317–326. https://doi.org/10.1145/2933575.2934516

[27] Rasmus Ejlers Møgelberg and Andrea Vezzosi. [n. d.]. Two Guarded Recursive Powerdomains for Applicative Simulation. In *Proceedings 37th Conference on Mathematical Foundations of Programming Semantics* (2021-12), Vol. 351. Electronic Proceedings in Theoretical Computer Science, 200–217. https://doi.org/10.4204/EPTCS.351.13

[28] Rasmus Ejlers Møgelberg. 2014. A Type Theory for Productive Coprogramming via Guarded Recursion. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (CSL-LICS '14)*. https://doi.org/10.1145/2603088.2603132

[29] Rasmus Ejlers Møgelberg and Niccolò Veltri. 2019. Bisimulation as Path Type for Guarded Recursive Types. *Proceedings of the ACM on Programming Languages* 3, POPL (12 2019). https://doi.org/10.1145/3290317

[30] H. Nakano. 2000. A modality for recursion. In *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No.99CB36332)*. IEEE Computer Society, 255–266.

[31] Daniele Palombi and Jonathan Sterling. 2023. Classifying topoi in synthetic guarded domain theory. In *Proceedings 38th Conference on Mathematical Foundations of Programming Semantics, MFPS 2022*. https://doi.org/10.46298/entics.10323

[32] Marco Paviotti, Rasmus Ejlers Møgelberg, and Lars Birkedal. 2015. A Model of PCF in Guarded Type Theory. *Electronic Notes in Theoretical Computer Science* 319, Supplement C (2015), 333–349. https://doi.org/10.1016/j.entcs.2015.12.020 The 31st Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXI).

[33] Egbert Rijke, Michael Shulman, and Bas Spitters. 2020. Modalities in homotopy type theory. *Logical Methods in Computer Science* 16, 1 (2020). arXiv:1706.07526

[34] Michael Shulman. 2018. Brouwer's fixed-point theorem in real-cohesive homotopy type theory. *Mathematical Structures in Computer Science* 28, 6 (2018), 856–941. https://doi.org/10.1017/S0960129517000147

[35] Michael Shulman. 2023. Towards third generation HoTT. Joint work with Thorsten Altenkirch and Ambrus Kaposi. Slides available at https://home.sandiego.edu/~shulman/papers/hott-cmu-day1.pdf.

[36] Simon Spies, Lennard Gäher, Daniel Gratzer, Joseph Tassarotti, Robbert Krebbers, Derek Dreyer, and Lars Birkedal. 2021. *Transfinite Iris: Resolving an Existential Dilemma of Step-Indexed Separation Logic*. Association for Computing Machinery, New York, NY, USA, 80–95. https://doi.org/10.1145/3453483.3454031

[37] Jonathan Sterling and Carlo Angiuli. 2021. Normalization for Cubical Type Theory. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '21)*. ACM, New York, NY, USA.

[38] Jonathan Sterling, Daniel Gratzer, and Lars Birkedal. 2023. Denotational semantics of general store and polymorphism. arXiv:2210.02169 [cs.PL]

[39] Jonathan Sterling, Daniel Gratzer, and Lars Birkedal. 2024. Towards univalent reference types. In *Computer Science Logic (CSL 2018)*.

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

[41] Niccolò Veltri and Andrea Vezzosi. 2020. Formalizing π-calculus in guarded cubical Agda. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. 270–283.

A THE STANDARD MODEL OF Gatsby

Unlike the rest of the paper, in this appendix we will presuppose familiarity with the contents of Aagaard et al. [1]. We further assume the same notation as Section 5.

As mentioned in Section 5, we cannot simply apply the results of op. cit. as-is to Gatsby to interpret it into **cSet** and $\mathbf{PSh}_{\mathbf{cSet}}(\omega)$, because ϵ_0 and $\top$ are not interpreted by right Kan extension.

We begin by explicitly defining the left adjoints for the various functors as a strict 2-functor $\bar{F} : \mathcal{M}_{\mathsf{Gatsby}}{}^{\mathsf{coop}} \longrightarrow \mathbf{Cat}$:

$$\bar{F}(\ell)\, X\, n = X(n+1) \qquad \bar{F}(e)\, X\, n = X(\mathsf{pred}(n)) \qquad \bar{F}(\gamma)\, S\, n = S$$

$$\bar{F}(\delta)\, X = X(0) \qquad \bar{F}(\epsilon_0)\, S = \text{if } n = 0 \text{ then } S \text{ else } \mathbf{0} \qquad \bar{F}(\top)\, S = \mathbf{0}$$

This organizes into a strict 2-functor because each functor is either determined by precomposition or uses sends an object to $\mathbf{0}$; all relevant coherences are therefore necessarily realized by id.

Combining this with the model of cubical type theory in $\mathbf{PSh}_{\mathbf{cSet}}(\omega)$ [22], we are therefore to directly interpret all the rules of cubical MTT as in Proposition 5.1 *except* for those involving the interactions between modal and cubical features. Specifically, we must show that there exists a set of coherent natural isomorphisms between the interpretations of the following:

- The collections $\{\phi \mid \Gamma \vdash \phi\}$ and $\{\phi' \mid \Gamma.\{\mu\} \vdash \phi'\}$
- $\Gamma.\{\mu\}.\mathbb{I}$ and $\Gamma.\mathbb{I}.\{\mu\}$
- $\Gamma.\{\mu\}.\phi$ and $\Gamma.\bar{\phi}.\{\mu\}$ where $\bar{\phi}$ is induced by the first point.

Aagaard et al. [1] notes that for any modality μ interpreted by right Kan extension, all of these isomorphisms may be realized by the identity. While in general, this is not the case for modalities defined by precomposition like $F(\epsilon_0)$ and $F(\top)$, it is the case for these two functors. For instance, $[\![\mathbb{I}_t]\!]$ in $\mathbf{PSh}_{\mathbf{cSet}}(\omega)$ is defined by $[\![\mathbb{I}_t]\!]\, n = [\![\mathbb{I}_s]\!]$. Accordingly, for $\mu = \epsilon_0$ the second isomorphism above is equivalent to requiring an isomorphism between the following presheaves:

$$[\![\Gamma.\{\epsilon_0\}.\mathbb{I}_t]\!]\, n = \begin{cases} [\![\Gamma.\{\mu\}]\!] \times [\![\mathbb{I}_s]\!] & n = 0 \\ \mathbf{0} \end{cases}$$

$$[\![\Gamma.\mathbb{I}_s.\{\epsilon_0\}]\!]\, n = \begin{cases} [\![\Gamma.\{\mu\}]\!] \times [\![\mathbb{I}_s]\!] & n = 0 \\ \mathbf{0} \end{cases}$$

In the above, we have capitalized on the fact that $\mathbf{0} \times X = \mathbf{0}$ in the category of (cubical) sets. We can therefore realize this (and all other necessary) isomorphisms by the identity. This also ensures that all necessary coherence conditions are satisfied. Accordingly, the argument given for Proposition 5.1 holds without real adaptation.