Constructing Unprejudiced Extensional Type Theories with Choices via Modalities

Liron Cohen
Ben-Gurion University, Israel

Vincent Rahli
University of Birmingham, UK

Abstract

Time-progressing expressions, i.e., expressions that compute to different values over time such as Brouwerian choice sequences or reference cells, are a common feature in many frameworks. For type theories to support such elements, they usually employ sheaf models. In this paper, we provide a general framework in the form of an extensional type theory incorporating various time-progressing elements along with a general possible-worlds forcing interpretation parameterized by modalities. The modalities can, in turn, be instantiated with topological spaces of bars, leading to a general sheaf model. This parameterized construction allows us to capture a distinction between theories that are "agnostic", i.e., compatible with classical reasoning in the sense that classical axioms can be validated, and those that are "intuitionistic", i.e., incompatible with classical reasoning in the sense that classical axioms can be proven false. This distinction is made via properties of the modalities selected to model the theory and consequently via the space of bars instantiating the modalities. We further identify a class of time-progressing elements that allows deriving "intuitionistic" theories that include not only choice sequences but also simpler operators, namely reference cells.

2012 ACM Subject Classification Theory of computation → Type theory; Theory of computation → Constructive mathematics

Keywords and phrases Intuitionism, Extensional Type Theory, Constructive Type Theory, Realizability, Choice sequences, References, Classical Logic, Theorem proving, Agda

Digital Object Identifier 10.4230/LIPIcs.FSCD.2022.17

1 Introduction

Time-progressing elements are a common feature in many frameworks. These are elements whose value can change over time. Examples include mutable reference cells which are pervasive in programming languages, and free-choice sequences which are key components in logical systems such as Brouwer's intuitionistic logic [26; 3; 40; 41; 28; 43; 32]. A free-choice sequence is a primitive concept of a sequence that is never complete and can always be extended over time, and whose choices are allowed to be made freely, i.e., not generated by a predefined procedure. Capturing the non-deterministic, time-progressing behavior of such elements in a formal setting often relies on sheaf models, which logical formulas can interact with through a forcing interpretation, e.g., [21; 42].

The inclusion of such elements in a logical system has far reaching consequences. In particular, many works have used the existence of choice-sequences to show incompatibility with classical reasoning. For example, Kripke's Schema, which relies on the notion of choice sequences, is inconsistent with Church's Thesis [18, Sec.5]. They have also been used to refute classical results such as *"any real number different from* 0 *is also apart from* 0*"* [24, Ch.8]. Similarly, a weak counterexample of the Law of Excluded Middle (LEM) was provided by defining a choice sequence of numbers in which the value 1 can only be picked once an undecided conjecture has been resolved (proved or disproved), and then by showing that one could resolve this undecided conjecture using LEM [9, Ch.1, Sec.1]. Kripke [30, Sec.1.1] also used choice sequences to refute other classical results, namely Kuroda's conjecture and

7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022).
Editor: Amy P. Felty; Article No. 17; pp. 17:1–17:25
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Markov's Principle (MP) in Kreisel's FC system [27]. This technique was later generalized using sheaf models [21; 42] to refute classical axioms. For example, in [15] the independence of MP with Martin-Löf's type theory was proven using a forcing method where the forcing conditions capture the unconstrained nature of free-choice sequences in Kripke's proof. However, using a concrete sheaf model, it was shown in [6] that choice sequences can be made compatible with classical reasoning. This was however done by committing to a particular model, disabling the ability to derive "purely" intuitionistic theories.

This paper goes one step further by providing a general framework in the form of an extensional type theory that incorporates a notion of time progression through a Kripke frame, as well as elements that progress over time. The framework uses a general possible-worlds forcing interpretation parameterized by a modality, which, in turn can be instantiated with topological spaces of bars, leading to a general sheaf model. Thus, our generic type theory, denoted by $\mathrm{TT}_{\mathcal{C}}^{\square}$, is modeled through an abstract modality $\square$ and is parameterized by a type of time-progressing choice operators $\mathcal{C}$, which can both be instantiated to derive theories that are either compatible or incompatible with classical logic. $\mathrm{TT}_{\mathcal{C}}^{\square}$'s syntax and operational semantics are presented by first describing its time-independent core in Sec. 2.2, and then its time-progressing components in Sec. 3. In particular, $\mathrm{TT}_{\mathcal{C}}^{\square}$ can be instantiated with different choice operators described in Sec. 3.2. $\mathrm{TT}_{\mathcal{C}}^{\square}$'s inference rules are standard and are presented in Appx. A. They reflect the semantics of the types, which are given meaning through a forcing interpretation [11; 12; 4, Ch.15] parameterized by a modality $\square$ presented in Sec. 4.

We call $\mathrm{TT}_{\mathcal{C}}^{\square}$ an "unprejudiced" type theory since we can tune the parameters to obtain theories that are either "agnostic", i.e., compatible with classical reasoning (in the sense that classical axioms can be validated), or that are "intuitionistic", i.e., incompatible with classical reasoning (in the sense that classical axioms can be proven false). Concretely, we identify classes of choice operators and modalities that are sufficient to derive the negation of classical axioms, as well as classes that are sufficient to validate classical axioms in Sec. 5. We further show that $\mathrm{TT}_{\mathcal{C}}^{\square}$ can be validated w.r.t. standard sheaf models in Sec. 6, which presents classes of sheaf models over topological spaces of bars that are used to instantiate the modalities. We provide examples of classes of bar spaces B and choice operators $\mathcal{C}$ that allow proving the consistency of $\mathrm{TT}_{\mathcal{C}}^{B}$ with LEM, and classes that allow proving the consistency of $\mathrm{TT}_{\mathcal{C}}^{B}$ with the negation of classical axioms such as LEM. In particular, we show that even though choice sequences can be used to validate the negation of classical axioms, they are not necessary, and in fact much simpler choice operators, e.g. mutable references, are enough.

2 Background

2.1 Metatheory

Our metatheory is Agda's type theory [1]. The results presented in this paper have been formalized in Agda, and the formalization is available here: `https://github.com/vrahli/opentt/`. We use $\forall, \exists, \wedge, \vee, \rightarrow, \neg$ in place of Agda's logical connectives in this paper. Agda provides an hierarchy of types annotated with universe labels which we omit for simplicity. Following Agda's terminology, we refer to an Agda type as a *set*, and reserve the term *type* for $\mathrm{TT}_{\mathcal{C}}^{\square}$'s types. We use $\mathbb{P}$ as the type of sets that denote propositions; $\mathbb{N}$ for the set of natural numbers; and $\mathbb{B}$ for the set of Booleans true and false. We use induction-recursion to define the forcing interpretation in Sec. 4, where we use function extensionality to interpret universes. We do not discuss this further here and the interested reader is referred to forcing.lagda in the Agda code for further details. Classical reasoning is only used once in Lem. 19 to establish the compatibility of instances of $\mathrm{TT}_{\mathcal{C}}^{\square}$ with LEM.

Figure 1 Core syntax (above) and small-step operational semantics (below)

$v \in$ Value ::=	vt	(type)	$\mid \lambda x.t$	(lambda)	$\mid \star$	(constant)
	$\mid \underline{n}$	(number)	$\mid \texttt{inl}(t)$	(left injection)	$\boxed{\mid \delta}$	$\boxed{\text{(choice name)}}$
	$\mid \langle t_1, t_2 \rangle$	(pair)	$\mid \texttt{inr}(t)$	(right injection)		
$vt \in$ Type ::=	$\mathbf{\Pi} x{:}t_1.t_2$	(product)	$\mid \{x : t_1 \mid t_2\}$	(set)	$\mid t_1{+}t_2$	(disjoint union)
	$\mid \mathbf{\Sigma} x{:}t_1.t_2$	(sum)	$\mid t_1{=}t_2{\in}t$	(equality)	$\boxed{\mid \downzigzagarrow t}$	$\boxed{\text{(time truncation)}}$
	$\mid \mathbb{U}_i$	(universe)	$\mid \mathsf{Nat}$	(numbers)		
$t \in$ Term ::=	x	(variable)	$\mid t_1 \; t_2$			(application)
	$\mid v$	(value)	$\mid \texttt{let}\ x, y = t_1\ \texttt{in}\ t_2$			(pair destructor)
	$\mid \texttt{fix}(t)$	(fixpoint)	$\mid \texttt{case}\ t\ \texttt{of}\ \texttt{inl}(x) \Rightarrow t_1 \mid \texttt{inr}(y) \Rightarrow t_2$			(injection destructor)

$$
\begin{array}{ll}
(\lambda x.t)\ u \ \mapsto_{\boxed{w}}\ t[x\backslash u] & \texttt{let}\ x, y = \langle t_1, t_2 \rangle\ \texttt{in}\ t \mapsto_{\boxed{w}} t[x\backslash t_1; y\backslash t_2] \\
\texttt{fix}(v) \ \mapsto_{\boxed{w}}\ v\ \texttt{fix}(v) & \texttt{case}\ \texttt{inl}(t)\ \texttt{of}\ \texttt{inl}(x) \Rightarrow t_1 \mid \texttt{inr}(y) \Rightarrow t_2 \mapsto_{\boxed{w}} t_1[x\backslash t] \\
\boxed{\delta(\underline{n}) \mapsto_w \mathsf{choice?}(w, \delta, n)} & \texttt{case}\ \texttt{inr}(t)\ \texttt{of}\ \texttt{inl}(x) \Rightarrow t_1 \mid \texttt{inr}(y) \Rightarrow t_2 \mapsto_{\boxed{w}} t_2[y\backslash t]
\end{array}
$$

2.2 $\text{TT}_{\mathcal{C}}^{\square}$'s Core Syntax and Operational Semantics

$\text{TT}_{\mathcal{C}}^{\square}$'s core syntax and operational semantics are presented in Fig. 1, which for presentation purposes also includes the additional components introduced in Sec. 3, highlighted in blue boxes. Fig. 1's upper part presents the syntax of $\text{TT}_{\mathcal{C}}^{\square}$'s core computation system, where x belongs to a set of variables Var. For simplicity, numbers are considered to be primitive. The constant $\star$ is there for convenience, and is used in place of a term, when the particular term used is irrelevant. Terms are evaluated according to the operational semantics presented in Fig. 1's lower part. In what follows, we use all letters as metavariables for terms. Let $t[x\backslash u]$ stand for the capture-avoiding substitution of all the free occurrences of x in t by u.

Types are syntactic forms that are given semantics in Sec. 4 via a forcing interpretation. The type system contains standard types such as dependent products of the form $\mathbf{\Pi} x{:}t_1.t_2$ and dependent sums of the form $\mathbf{\Sigma} x{:}t_1.t_2$. For convenience we write $t_1 \to t_2$ for the non-dependent $\mathbf{\Pi}$ type; $\texttt{True}$ for $\underline{0}{=}\underline{0}{\in}\mathsf{Nat}$; $\texttt{False}$ for $\underline{0}{=}\underline{1}{\in}\mathsf{Nat}$; $\neg T$ for $(T \to \texttt{False})$; Bool for $\texttt{True}{+}\texttt{True}$; tt for $\texttt{inl}(\star)$; ff for $\texttt{inr}(\star)$; and $\uparrow(t)$ for $t{=}\mathsf{tt}{\in}\mathsf{Bool}$ (a Bool to type coercion).

Our computation system includes a *space-squashing* mechanism, which we use (among other things) to validate some of the axioms in Secs. 5.1 and 5.2. It erases the evidence that a type is inhabited by truncating it to a subsingleton type using set types: $\downarrow T \coloneqq \{x : \texttt{True} \mid T\}$. While $\texttt{True}$ is a contractible type (because equality types are subsingleton types — see Sec. 4), $\downarrow T$ is either empty or inhabited by all (closed) terms in Term, and all its inhabitants are equal to each other. Therefore, $\downarrow T$ is inhabited iff T is inhabited.

Fig. 1's lower part presents $\text{TT}_{\mathcal{C}}^{\square}$'s core small-step operational semantics, where $t_1 \mapsto t_2$ expresses that the term t_1 reduces to t_2 in one computation step. We omit the congruence rules that allow computing within terms such as: if $t_1 \mapsto t_2$ then $t_1(u) \mapsto t_2(u)$. We denote by $\Downarrow$ the reflexive transitive closure of $\mapsto$, i.e., $a \Downarrow b$ states that a computes to b in ≥ 0 steps.

3 $\text{TT}_{\mathcal{C}}^{\square}$'s Time-Progressing Choice Operators

In addition to the core described in Sec. 2.2, $\text{TT}_{\mathcal{C}}^{\square}$ includes time-progressing notions which we now describe. We capture these notions via the concept of worlds (Sec. 3.1). Then, we provide a formal, abstract definition of choice operators and add corresponding components to the core system (Sec. 3.2). These time-progressing choice operators cover standard operators such as Brouwerian choice sequences or references (Sec. 3.2.1). We further enrich our system with a notion of time-truncation, used to capture time-sensitive expressions (Sec. 3.3).

3.1 Worlds

To capture the time progression notion, the core computation system presented in Sec. 2.2 is parameterized by a Kripke frame [31; 30] defined as follows:

▶ **Definition 1** (Kripke Frame). *A Kripke frame consists of a set of* worlds $\mathcal{W}$ *equipped with a reflexive and transitive binary relation* $\sqsubseteq$.

Let w range over $\mathcal{W}$. We sometimes write $w' \sqsupseteq w$ for $w \sqsubseteq w'$. Let $\mathcal{P}_w$ be the collection of predicates on world extensions, i.e., functions in $\forall w' \sqsupseteq w.\mathbb{P}$. Note that due to $\sqsubseteq$'s transitivity, if $P \in \mathcal{P}_w$ then for every $w' \sqsupseteq w$ it naturally extends to a predicate in $\mathcal{P}_{w'}$. We further define the following notations for quantifiers. $\forall^{\sqsubseteq}_{w}(P)$ states that $P \in \mathcal{P}_w$ is true for all extensions of w, i.e., $P\ w'$ holds in all worlds $w' \sqsupseteq w$. $\exists^{\sqsubseteq}_{w}(P)$ states that $P \in \mathcal{P}_w$ is true at an extension of w, i.e., $P\ w'$ holds for some world $w' \sqsupseteq w$. For readability, we sometime write $\forall^{\sqsubseteq}_{w}(w'.P)$ (or $\exists^{\sqsubseteq}_{w}(w'.P)$) instead of $\forall^{\sqsubseteq}_{w}(\lambda w'.P)$ (or $\exists^{\sqsubseteq}_{w}(\lambda w'.P)$), respectively.

The operational semantics is parameterized by a frame in the sense that the relation $t_1 \mapsto t_2$ is generalized to a ternary relation between two terms and a world, $t_1 \mapsto_w t_2$, which expresses that t_1 reduces to t_2 in one step of computation *w.r.t. the world* w. Similarly, $a \Downarrow_w b$ generalizes $a \Downarrow b$. We also write $a \Downarrow\!\!\Downarrow_w b$ if a computes to b in all extensions of w, i.e., if $\forall^{\sqsubseteq}_{w}(w'.a \Downarrow_{w'} b)$. We write $\sim_w$ for the symmetric and transitive closure of $\Downarrow\!\!\Downarrow_w$.

3.2 Time-Progressing Choice Operators

This section introduces the general notion of time-progressing choices into our system. We rely on worlds to record choices and provide operators to access the choices stored in a world. Choices are referred to through their names. A concrete example of such choices are reference cells in programming languages, where a variable name pointing to a reference cell is the name of the corresponding reference cell. To introduce an abstract notion of such choice operators, we assume our computation system contains a set $\mathcal{N}$ of *choice names*, that is equipped with a decidable equality, and an operator that given a list of names, returns a name not in the list. This can be given by, e.g., nominal sets [39]. In what follows we let δ range over $\mathcal{N}$, and take $\mathcal{N}$ to be $\mathbb{N}$ for simplicity. We introduce further abstract operators and properties in Defs. 2, 4, 8, 10–12, 14, 15, and 18 which our framework is parameterized over, and which we show how to instantiate in Exs. 5, 6, 13, 26, 27, and 29 below. Definitions such as Def. 2 provide axiomatizations of operators, and in addition informally indicate their intended use. Choices are defined abstractly as follows:

▶ **Definition 2** (Choices). *Let* $\mathcal{C} \subseteq \mathsf{Term}$ *be a set of* choices,[1] *and let* κ *range over* $\mathcal{C}$. *We say that a computation system contains* $\langle\mathcal{N},\mathcal{C}\rangle$*-choices if there exists a partial function* $\mathsf{choice?} \in \mathcal{W} \to \mathcal{N} \to \mathbb{N} \to \mathcal{C}$. *Given* $w \in \mathcal{W}$, $\delta \in \mathcal{N}$, $n \in \mathbb{N}$, *the returned choice, if it exists, is meant to be the* n^{th} *choice made for* δ *according to* w. $\mathcal{C}$ *is said to be* non-trivial *if it contains two values* $\boldsymbol{\kappa_0}$ *and* $\boldsymbol{\kappa_1}$, *which are computationally different, i.e., such that* $\neg(\boldsymbol{\kappa_0} \sim_w \boldsymbol{\kappa_1})$ *for all* w.

Thus, to introduce choices into the computation system, we extend the core computation system with a new kind of value for a choice name δ (as shown in Fig. 1) that can be used to access choices from a world. To facilitate making use of choices extracted from worlds and computing with them, the operational semantics is also extended with the following

[1] To guarantee that $\mathcal{C} \subseteq \mathsf{Term}$, one can for example extend the syntax to include a designated constructor for choices, or require a coercion $\mathcal{C} \to \mathsf{Term}$. We opted for the latter in our formalization.

clause: $\delta(\underline{n}) \mapsto_w \mathsf{choice?}(w, \delta, n)$ (as shown in Fig. 1). This allows applying a choice name δ to a number $\underline{n}$ to get a choice from the current world w. Note that the $\mathbb{N}$ component in this definition enables providing a general notion of choice operators. In some cases, e.g. the case for free-choice sequences, the history is recorded and so the notion of an n's choice is extracted from the history of the choice element. In simpler choice concepts, e.g. references, one only maintains the latest update and so the $\mathbb{N}$ component becomes moot.[2]

We next introduce the notion of a *restriction*, which allows assuming that the choices made for a given choice name all satisfy a pre-defined constraint.

▶ **Definition 3** (Restrictions). *A restriction $r \in \mathsf{Res}$ is a pair $\langle res, d\rangle$ consisting of a function $res \in \mathbb{N} \to \mathcal{C} \to \mathbb{P}$ and a default choice $d \in \mathcal{C}$, such that $\forall(n : \mathbb{N}).(res\ n\ d)$ holds. Given such a pair r, we write $r._{\mathsf{d}}$ for d; $(r\ n\ \kappa)$ for $(res\ n\ \kappa)$; and $r(\kappa)$ for $\forall(n : \mathbb{N}).r\ n\ \kappa$.*

Intuitively, *res* specifies a restriction on the choices that can be made at any point in time and d provides a default choice that meets this restriction (e.g., for reference cells, this default choice is used to initialize a cell). For example, the restriction $\langle \lambda n.\lambda\kappa.\kappa \in \mathbb{N}, 0\rangle$ requires choices to be numbers and provides 0 as a default value. To reason about restrictions, we require the existence of a "compatibility" predicate as follows.

▶ **Definition 4.** *We further assume the existence of a predicate $\mathsf{compatible} \in \mathcal{N} \to \mathcal{W} \to \mathsf{Res} \to \mathbb{P}$, intended to guarantee that restrictions are satisfied, and which is preserved by $\sqsubseteq$: $\forall(\delta : \mathcal{N})(w_1, w_2 : \mathcal{W})(r : \mathsf{Res}).w_1 \sqsubseteq w_2 \to \mathsf{compatible}(\delta, w_1, r) \to \mathsf{compatible}(\delta, w_2, r)$.*

3.2.1 Standard Examples of Choice Operators

The abstract notion of choice operators has many concrete instances. This section provides a high-level description of two such instances: a theoretically-oriented one, based on the notion of free-choice sequences, and a programming-oriented one, based on mutable references.

▶ **Example 5** (Free-Choice Sequences). Free choices are fundamental objects introduced by Brouwer [10] that lay at the heart of intuitionistic mathematics. They are there described as *"new mathematical entities... in the form of infinitely proceeding sequences, whose terms are chosen more or less freely from mathematical entities previously acquired"*. Thus, free-choice sequences are never-finished sequences of objects created *over time* by continuously picking elements from a previously well-defined collection, e.g., the natural numbers. Even though free-choice sequences are ever proceeding, at any point in time the sequence of choices made so far is finite. Therefore, the current state of a choice sequence can be implemented as a list of choices. We use worlds to capture the state of all the choice sequences started so far, and the $\sqsubseteq$ relation on worlds captures the fact that an extension of a world can contain additional choices. In that respect, a choice sequence can be seen as a reference cell that maintains the complete history of values that were stored in the cell. Formally, we define choice sequences of terms, Fcs, as follows (see worldInstanceCS.lagda for details):

Non-Trivial Choices Let $\mathcal{N} := \mathbb{N}$ and $\mathcal{C} := \mathsf{Term}$, which is non-trivial, e.g., take $\boldsymbol{\kappa_0} := \underline{0}$ and $\boldsymbol{\kappa_1} := \underline{1}$. Other examples of $\mathcal{C}$s that would be suitable for the results presented in this paper are $\mathbb{N}$, with $\boldsymbol{\kappa_0} := 0$ and $\boldsymbol{\kappa_1} := 1$ (which can be mapped to the terms $\underline{0}$ and $\underline{1}$); or $\mathbb{B}$ with $\boldsymbol{\kappa_0} := \mathsf{true}$ and $\boldsymbol{\kappa_1} := \mathsf{false}$ (which can be mapped to the terms $\mathtt{tt}$ and $\mathtt{ff}$).

[2] Technically, this can be captured by instantiating $\mathcal{C}$ with a function type from $\mathbb{N}$ when records are kept. For simplicity, we here opt to make $\mathbb{N}$ explicit.

Worlds Worlds are instantiated as lists of entries, where an entry is either (1) a pair of a choice name and a restriction, indicating the creation of a choice sequence; or (2) a pair of a choice name δ and a choice κ indicating the extension of the choice sequence δ with the new choice κ. $\sqsubseteq$ is the reflexive transitive closure of these extension operations. Given an entry list w and a name δ, the state of the choice sequence δ in w is then the list of extensions made to δ starting from the point δ was created in w, which allows us to define choice? by looking up the n^{th} choice in that list. This enables starting multiple choice sequences in parallel, which is crucial in the proof of Lem. 16.

Compatibility compatible(δ, w, r) states that a choice sequence named δ with restriction r was started in the world w (using the first kind of entry described above), and that all the choices made for δ in w satisfy r.

▶ **Example 6** (References)**.** Reference cells, which are values that allow a program to indirectly access a particular object, are also choice operators since they can be pointed to different objects over their lifetime. As opposed to a choice sequence, with a reference cell, the history of previous choices is not kept, and the old recorded value is discarded when a new value is stored in a reference cell. In this paper, we will make use of a particular class of reference cell, that are mutable, but can be made immutable at any given point, i.e., the reference cell can be "frozen" so that new values cannot be stored anymore. Formally, we define references to terms, Ref, as follows (see worldInstanceRef.lagda for details):

Non-trivial Choices $\mathcal{N}$ and $\mathcal{C}$ are defined as for free-choice sequences.

Worlds Worlds are lists of cells, where a cell is a quadruple of (1) a choice name, (2) a restriction, (3) a choice, and (4) a Boolean indicating whether the cell is mutable. $\sqsubseteq$ is the reflexive transitive closure of two operations that allow (1) creating a new reference cell, and (2) updating an existing reference cell. We define choice?(w, δ, n) so that it simply accesses the content of the δ cell in w, irrespective of what n is. Again, this allows for maintaining multiple reference cells, which is crucial in the proof of Lem. 16.

Compatibility compatible(δ, w, r) states that a reference cell named δ with restriction r was created in the world w (using the first kind of operation described above), and that the current value of the cell satisfies r.

3.3 Time-Truncation

While some computations are *time-invariant*, in the sense that they compute to the same value at any point in time, others, such as references, are *time-sensitive*. These two kinds of computations have different properties, e.g., a time-invariant term t that computes to a number $\underline{n}$ in a world w, will compute to $\underline{n}$ in all $w' \sqsupseteq w$. However, if t is a time-sensitive number, t might compute to numbers different from $\underline{n}$ in extensions of w, e.g., $\underline{n+1}$ in $w' \sqsupseteq w$ and $\underline{n+2}$ in $w'' \sqsupseteq w'$. To capture this distinction at the level of types, we further enrich $\mathrm{TT}_{\mathcal{C}}^{\Box}$ by a time-truncation operator $\wr$. The type $\wr T$ contains T's members as well as the terms that behave like members of T at a particular point in time, i.e., in a particular world.

In this paper, we make use in particular of the type $\wr$Nat, which as opposed to Nat, is not required to only be inhabited by time-invariant terms, and allows for terms to compute to different numbers in different world extensions. For example, $\wr$Nat is allowed to be inhabited by a term t that computes to $\underline{3}$ in some world w, and to $\underline{4}$ in $w' \sqsupseteq w$. A reference cell that holds numbers is then essentially of type $\wr$Nat but not of type Nat, as its content can change over time. This distinction between Nat and $\wr$Nat will be critical when validating the negation of classical axioms in Sec. 5.1, where we make use of time-sensitive references (in particular in Ex. 13). Note that as we only need a type with two different inhabitants, we could have equally used $\lightning\mathsf{Bool}$, whose inhabitants compute to either $\mathtt{tt}$ or $\mathtt{ff}$ in a given world, but might compute to different Booleans in different extensions.

4 The Modality-based Forcing Interpretation

Now that we have defined $\mathrm{TT}^{\square}_{\mathcal{C}}$'s computation system that includes choice operators, we provide a semantic for it. $\mathrm{TT}^{\square}_{\mathcal{C}}$ is interpreted via a forcing interpretation in which the forcing conditions are worlds. This interpretation is defined using induction-recursion as follows: (1) the inductive relation $w \vDash T_1 \equiv T_2$ expresses type equality in the world w; (2) the recursive function $w \vDash t_1 \equiv t_2 \in T$ expresses equality in a type. We further use the following abstractions: $w \vDash \mathtt{type}(T)$ for $w \vDash T \equiv T$, $w \vDash t \in T$ for $w \vDash t \equiv t \in T$, and $w \vDash T$ for $\exists(t : \mathsf{Term}).w \vDash t \in T$.

This forcing interpretation is parameterized by a family of abstract modalities $\square$, which we sometimes refer to simply as a modality, which is a function that takes a world w to its modality $\square_w \in \mathcal{P}_w \to \mathbb{P}$. We often write $\square_w(w'.P)$ for $\square_w \lambda w'.P$. To guarantee that this interpretation yields a standard type system in the sense of Thm. 9, we require in Def. 8. that the modalities satisfy certain properties reminiscent of standard modal axiom schemata [17].

The inductive relation $w \vDash T_1 \equiv T_2$ has one constructor per type plus one additional constructor expressing when two types are equal in a world w using the $\square_w$ modality. Consequently, the recursive function $w \vDash t_1 \equiv t_2 \in T$ has as many cases as there are constructors for $w \vDash T \equiv T'$, requiring a dependent version $\boxdot^i_w$ of $\square_w$ to recurse over i, which is a proof that T is given meaning using the $\square_w$ modality. Indeed, technically, $\square$ induces two abstract modalities for a world w: the modality $\square_w \in \mathcal{P}_w \to \mathbb{P}$, and a dependent version $\boxdot^i_w$, where $P \in \mathcal{P}_w \to \mathbb{P}$ and $i \in \square_w P$. However, to avoid the technical details involved with the dependent modality $\boxdot^i_w$, we opt here for a slightly informal presentation where we slid the technical details concerning the dependent modality to Appx. B.

▶ **Definition 7** (Forcing interpretation). *Given modality $\square$, the forcing interpretation of $\mathrm{TT}^{\square}_{\mathcal{C}}$ is given in Fig. 2. There, we write R^+ for R's transitive closure, and $\mathsf{Fam}_w(A_1, A_2, B_1, B_2)$ for $w \vDash A_1 \equiv A_2 \wedge \forall^{\sqsupseteq}_w(w'.\forall(a_1, a_2 : \mathsf{Term}).w' \vDash a_1 \equiv a_2 \in A_1 \to w' \vDash B_1[x \backslash a_1] \equiv B_2[x \backslash a_2])$.*[3]

There are some standard properties expected for a semantics such as this forcing interpretation to constitute a type system [2; 16]. These include the monotonocity and locality properties expected for a possible-world semantics [44; 20; 19, Sec.5.4] (here monotonicity refers to types, and not to computations). In order to obtain a type system satisfying such standard, useful properties, we must impose some conditions on the modality. Thus, we next identify a set of conditions for the underlying modality that is sufficient for proving these type system properties.

▶ **Definition 8** (Equality modality). *The modality $\square$ is called an* equality modality *if it satisfies the following properties:*

- $\square_1$ *(monotonicity of $\square$): $\forall(w : \mathcal{W})(P : \mathcal{P}_w).\forall w' \sqsupseteq w.\, \square_w P \to \square_{w'} P$.*
- $\square_2$ *(K, distribution axiom): $\forall(w : \mathcal{W})(P, Q : \mathcal{P}_w).\, \square_w (w'.P\ w' \to Q\ w') \to \square_w P \to \square_w Q$*
- $\square_3$ *(C4, i.e., $\square$ follows from $\square\square$): $\forall(w : \mathcal{W})(P : \mathcal{P}_w).\, \square_w (w'.\square_{w'} P) \to \square_w P$*
- $\square_4$*: $\forall(w : \mathcal{W})(P : \mathcal{P}_w).\forall^{\sqsupseteq}_w(P) \to \square_w P$*
- $\square_5$ *(T, reflexivity axiom): $\forall(w : \mathcal{W})(P : \mathbb{P}).\, \square_w (w'.P) \to P$*

As detailed in Appx. B, we further require that the dependent modality $\boxdot$ satisfies similar properties to the ones listed above, as well as properties relating the two modalities.

[3] For readability, we adopt a slightly different presentation here compared to the Agda formalization. See Appx. B for a faithful presentation, which in addition covers universes.

Figure 2 Forcing Interpretation

Numbers:
- $w \vDash \mathsf{Nat}{\equiv}\mathsf{Nat} \iff \mathtt{True}$
- $w \vDash t{\equiv}t'{\in}\mathsf{Nat} \iff \square_w(w'.\exists(n:\mathbb{N}).t \Downarrow_{w'} \underline{n} \wedge t' \Downarrow_{w'} \underline{n})$

Products:
- $w \vDash \mathbf{\Pi}x{:}A_1.B_1{\equiv}\mathbf{\Pi}x{:}A_2.B_2 \iff \mathsf{Fam}_w(A_1,A_2,B_1,B_2)$
- $w \vDash f{\equiv}g{\in}\mathbf{\Pi}x{:}A.B \iff \square_w(w'.\forall(a_1,a_2:\mathsf{Term}).w' \vDash a_1{\equiv}a_2{\in}A \rightarrow w' \vDash f\ a_1{\equiv}g\ a_2{\in}B[x\backslash a_1])$

Sums:
- $w \vDash \mathbf{\Sigma}x{:}A_1.B_1{\equiv}\mathbf{\Sigma}x{:}A_2.B_2 \iff \mathsf{Fam}_w(A_1,A_2,B_1,B_2)$
- $w \vDash p_1{\equiv}p_2{\in}\mathbf{\Sigma}x{:}A.B \iff \square_w(w'.\exists(a_1,a_2,b_1,b_2:\mathsf{Term}).w' \vDash a_1{\equiv}a_2{\in}A \wedge w' \vDash b_1{\equiv}b_2{\in}B[x\backslash a_1] \wedge p_1 \Downarrow_{w'} \langle a_1,b_1\rangle \wedge p_2 \Downarrow_{w'} \langle a_2,b_2\rangle)$

Sets:
- $w \vDash \{x:A_1 \mid B_1\}{\equiv}\{x:A_2 \mid B_2\} \iff \mathsf{Fam}_w(A_1,A_2,B_1,B_2)$
- $w \vDash a_1{\equiv}a_2{\in}\{x:A \mid B\} \iff \square_w(w'.\exists(b_1,b_2:\mathsf{Term}).w' \vDash a_1{\equiv}a_2{\in}A \wedge w' \vDash b_1{\equiv}b_2{\in}B[x\backslash a_1])$

Disjoint unions:
- $w \vDash A_1{+}B_1{\equiv}A_2{+}B_2 \iff w \vDash A_1{\equiv}A_2 \wedge w \vDash B_1{\equiv}B_2$
- $w \vDash a_1{\equiv}a_2{\in}A{+}B \iff \square_w(w'.\exists(u,v:\mathsf{Term}).(a_1 \Downarrow_{w'} \mathtt{inl}(u) \wedge a_2 \Downarrow_{w'} \mathtt{inl}(v) \wedge w' \vDash u{\equiv}v{\in}A) \vee (a_1 \Downarrow_{w'} \mathtt{inr}(u) \wedge a_2 \Downarrow_{w'} \mathtt{inr}(v) \wedge w' \vDash u{\equiv}v{\in}B))$

Equalities:
- $w \vDash (a_1{=}b_1{\in}A){\equiv}(a_2{=}b_2{\in}B) \iff w \vDash A{\equiv}B \wedge \forall^{\sqsubseteq}_w(w'.w' \vDash a_1{\equiv}a_2{\in}A) \wedge \forall^{\sqsubseteq}_w(w'.w' \vDash b_1{\equiv}b_2{\in}B)$
- $w \vDash a_1{\equiv}a_2{\in}(a{=}b{\in}A) \iff \square_w(w'.w' \vDash a{\equiv}b{\in}A)$ *(note that a_1 and a_2 can be any term here)*

Time-Quotiented types:
- $w \vDash \wr A{\equiv}\wr B \iff w \vDash A{\equiv}B$
- $w \vDash a{\equiv}b{\in}\wr A \iff \square_w(w'.(\lambda a,b.\exists(c,d:\mathsf{Value}).a \sim_w c \wedge b \sim_w d \wedge w \vDash c{\equiv}d{\in}A)^+\ a\ b)$

Modality closure:
- $w \vDash T_1{\equiv}T_2 \iff \square_w(w'.\exists(T_1',T_2':\mathsf{Term}).T_1 \Downarrow_{w'} T_1' \wedge T_2 \Downarrow_{w'} T_2' \wedge w' \vDash T_1'{\equiv}T_2')$
- $w \vDash t_1{\equiv}t_2{\in}T \iff \square_w(w'.\exists(T':\mathsf{Term}).T \Downarrow_{w'} T' \wedge w' \vDash t_1{\equiv}t_2{\in}T')$

▶ **Theorem 9.** *Given a computation system with choices $\mathcal{C}$ and an equality modality $\square$, $TT^{\square}_{\mathcal{C}}$ is a standard type system in the sense that its forcing interpretation induced by $\square$ satisfy the following properties (where free variables are universally quantified):*

transitivity:	$w \vDash T_1{\equiv}T_2 \rightarrow w \vDash T_2{\equiv}T_3 \rightarrow w \vDash T_1{\equiv}T_3$	$w \vDash t_1{\equiv}t_2{\in}T \rightarrow w \vDash t_2{\equiv}t_3{\in}T \rightarrow w \vDash t_1{\equiv}t_3{\in}T$
symmetry:	$w \vDash T_1{\equiv}T_2 \rightarrow w \vDash T_2{\equiv}T_1$	$w \vDash t_1{\equiv}t_2{\in}T \rightarrow w \vDash t_2{\equiv}t_1{\in}T$
computation:	$w \vDash T{\equiv}T \rightarrow T \Downarrow_w T' \rightarrow w \vDash T{\equiv}T'$	$w \vDash t{\equiv}t{\in}T \rightarrow t \Downarrow_w t' \rightarrow w \vDash t{\equiv}t'{\in}T$
monotonicity:	$w \vDash T_1{\equiv}T_2 \rightarrow w \sqsubseteq w' \rightarrow w' \vDash T_1{\equiv}T_2$	$w \vDash t_1{\equiv}t_2{\in}T \rightarrow w \sqsubseteq w' \rightarrow w' \vDash t_1{\equiv}t_2{\in}T$
locality:	$\square_w(w'.w' \vDash T_1{\equiv}T_2) \rightarrow w \vDash T_1{\equiv}T_2$	$\square_w(w'.w' \vDash t_1{\equiv}t_2{\in}T) \rightarrow w \vDash t_1{\equiv}t_2{\in}T$
consistency:	$\neg w \vDash t{\in}\mathtt{False}$	

Proof. The proof relies on the properties of the equality modality. For example: $\square_{\mathbf{1}}$ is used to prove monotonicity when $w \vDash T_1{\equiv}T_2$ is derived by closing under $\square_w$; $\square_{\mathbf{2}}$ and $\square_{\mathbf{4}}$ are used, e.g., to prove the symmetry and transitivity of $w \vDash t{\equiv}t'{\in}\mathsf{Nat}$; $\square_{\mathbf{3}}$ is used to prove locality; and $\square_{\mathbf{5}}$ is used to prove consistency. See props3.lagda for further details. ◀

5 Compatibility with Classical Axioms

To study the compatibility of $\mathrm{TT}^{\square}_{\mathcal{C}}$ with classical reasoning, this section identifies two subclasses of the family of type theories $\mathrm{TT}^{\square}_{\mathcal{C}}$, specified through conditions on the choices and modalities. Sec. 5.1 provides conditions that are sufficient to derive the negation of classical axioms such as LEM, while Sec. 5.2 provides conditions that are sufficient to derive LEM. We further give concrete instantiations for such choices and modalities (the modalities are instantiated only in Sec. 6.2 based on the notion of bars).

5.1 Intuitionistic Theories

This section identifies a set of general properties of choices and modalities that enables proving the negation of classical axioms such as LEM. We call theories based on such choices and modalities "intuitionistic", in the sense that they are incompatible with classical reasoning.

The proof of the negation of classical axioms provided below (Cor. 17) captures intuitionistic counterexamples [24; 9] abstractly. Briefly, we prove that, given a non-trivial choice structure, (A) if the only choice made so far is $\boldsymbol{\kappa_0}$, then it is not possible to decide whether $\boldsymbol{\kappa_1}$ will ever be made. More precisely, we prove that: (B) it is not the case that $\boldsymbol{\kappa_1}$ will be made because there are extensions where it won't; and (C) it is not the case that $\boldsymbol{\kappa_1}$ is not made in all extensions because there are extensions where it is made. To capture this, we require some additional properties from the underlying choices and modalities. To ensure that (A) holds, we introduce an *extendability* property in Def. 10, which allows creating a fresh choice name δ and a world w where the only choice made for δ in w is $\boldsymbol{\kappa_0}$. (B) is proved thanks to the properties introduced in Defs. 14 and 15, which guarantee the existence of an extension where the n^{th} choice made for δ is $\boldsymbol{\kappa_0}$, for any $n \in \mathbb{N}$. (C) is proved using the *immutability* property in Def. 11, which allows exhibiting a world where $\boldsymbol{\kappa_1}$ is made.

▶ **Definition 10** (Extendability). *We say that C is extendable if there exists a function $\nu\mathcal{C} \in \mathcal{W} \to \mathcal{N}$, where $\nu\mathcal{C}(w)$ is intended to return a new choice name not present in w, and a function $\mathsf{start}\nu\mathcal{C} \in \mathcal{W} \to \mathsf{Res} \to \mathcal{W}$, where $\mathsf{start}\nu\mathcal{C}(w,r)$ is intended to return an extension of w with the new choice name $\nu\mathcal{C}(w)$ with restriction r, satisfying the following properties:*

- *Starting a new choice extends the current world:* $\forall(w:\mathcal{W})(r:\mathsf{Res}).w \sqsubseteq \mathsf{start}\nu\mathcal{C}(w,r)$
- *Initially, the only possible choice is the default value of the given restriction, i.e.:* $\forall(n:\mathbb{N})(r:\mathsf{Res})(w:\mathcal{W})(\kappa:\mathcal{C}).\mathsf{choice?}(\mathsf{start}\nu\mathcal{C}(w,r),\nu\mathcal{C}(w),n)=\kappa \to \kappa = r_{.\mathsf{d}}$
- *A choice is initially compatible with its restriction:* $\forall(w:\mathcal{W})(r:\mathsf{Res}).\mathsf{compatible}(\nu\mathcal{C}(w),\mathsf{start}\nu\mathcal{C}(w,r),r)$

If only one choice κ was made so far for a name δ, then to prove (C) above we exhibit an extension where another choice κ' is made. Thus, we require a way to make a choice $\kappa' \neq \kappa$, as well as a way to make κ' immutable in the sense that no other choice than κ' can be made in the future. This is necessary because $\mathrm{TT}_{\mathcal{C}}^{\square}$ is a monotonic theory (see Lem. 16's proof). Consequently, we further rely on the ability to, at any point in time, be able to constrain the choices to be the same forever. This does not prevent making different choice before a choice is made immutable, and the ability to make different choices over time is indeed necessary as we just highlighted. To capture this, we define the immutability property.

▶ **Definition 11** (Immutability). *We say that C is immutable if there exist a function $\mathsf{freeze} \in \mathcal{N} \to \mathcal{C} \to \mathcal{W} \to \mathcal{W}$ (where $\mathsf{freeze}(\delta,\kappa,w)$ is intended to return a world w' that extends the world w with the choice κ for the choice name δ, and such that κ can be retrieved in any extension of w'), and a predicate $\mathsf{mutable} \in \mathcal{N} \to \mathcal{W} \to \mathbb{P}$ (intended to hold iff the choice name is mutable in the world, i.e., different choices can be made), satisfying the following properties:*

- *Making an immutable choice extends the current world:* $\forall(\delta:\mathcal{N})(w:\mathcal{W})(\kappa:\mathcal{C})(r:\mathsf{Res}).\mathsf{compatible}(\delta,w,r) \to r(\kappa) \to w \sqsubseteq \mathsf{freeze}(\delta,\kappa,w)$
- *A choice is initially mutable:* $\forall(w:\mathcal{W})(r:\mathsf{Res}).\mathsf{mutable}(\nu\mathcal{C}(w),\mathsf{start}\nu\mathcal{C}(w,r))$
- *Immutable choices stay immutable:* $\forall(\delta:\mathcal{N})(w:\mathcal{W})(\kappa:\mathcal{C})(r:\mathsf{Res}).\mathsf{compatible}(\delta,w,r) \to \mathsf{mutable}(\delta,w) \to \exists(n:\mathbb{N}).\forall^{\sqsubseteq}_{\mathsf{freeze}(\delta,\kappa,w)}(w'.\mathsf{choice?}(w',\delta,n)=\kappa)$

In addition, to state properties about non-trivial choices within $\mathrm{TT}_{\mathcal{C}}^{\square}$, such as the fact that it is not always decidable whether a choice will be made in the future (see $\Sigma\mathsf{choice}$ in Lem. 16), we assume the existence of a term ($\in \mathsf{Term}$) denoting a type that contains the two distinct choices $\boldsymbol{\kappa_0}$ and $\boldsymbol{\kappa_1}$, capturing Def. 2 at the level of the theory $\mathrm{TT}_{\mathcal{C}}^{\square}$.

▶ **Definition 12** (Reflection). *We say that* $\mathcal{C}$ *is* reflected *if there exists a term* $\mathsf{Type}\mathcal{C} \in \mathsf{Term}$ *such that the following hold for all worlds* w*:*

- $\mathsf{Type}\mathcal{C}$ *is a type inhabited by* $\boldsymbol{\kappa_0}$ *and* $\boldsymbol{\kappa_1}$*:* $w \vDash \mathtt{type}(\mathsf{Type}\mathcal{C})$, $w \vDash \boldsymbol{\kappa_0} \in \mathsf{Type}\mathcal{C}$, $w \vDash \boldsymbol{\kappa_1} \in \mathsf{Type}\mathcal{C}$.
- *The choices that inhabit* $\mathsf{Type}\mathcal{C}$ *are related w.r.t.* $\sim$*:* $\forall(w : \mathcal{W})(a, b : \mathsf{Term}).w \vDash a{\equiv}b \in \mathsf{Type}\mathcal{C} \rightarrow \square_w(w'.\forall^{\sqsubseteq}_{w'}(w''.\forall(\kappa_1, \kappa_2 : \mathcal{C}).a \Downarrow_{w''} \kappa_1 \rightarrow b \Downarrow_{w''} \kappa_2 \rightarrow \kappa_1 \sim_{w''} \kappa_2))$
- *Choices obtained from worlds that compute to either* $\boldsymbol{\kappa_0}$ *or* $\boldsymbol{\kappa_1}$ *inhabit* $\mathsf{Type}\mathcal{C}$*:* $\forall(w : \mathcal{W})(n : \mathbb{N})(\delta : \mathcal{N}).\ \square_w\ (w'.(\mathsf{choice?}(w', \delta, n) \Downarrow_{w'} \boldsymbol{\kappa_0} \lor \mathsf{choice?}(w', \delta, n) \Downarrow_{w'} \boldsymbol{\kappa_1})) \rightarrow w \vDash (\delta(\underline{n})) \in \mathsf{Type}\mathcal{C}$

Crucially, these properties allow $\mathsf{Type}\mathcal{C}$'s inhabitants to be time-sensitive, i.e., to compute to different choices in different extensions, which allows implementing choices with either references or choice sequences. As shown in Ex. 13, we can then instantiate $\mathsf{Type}\mathcal{C}$ with $\wr$-truncated types, which references inhabit.

Building up on the examples of choice operators presented in Exs. 5 and 6, we next provide examples for the aforementioned properties of choices.

▶ **Example 13.** Both free-choice sequences, Fcs, and references, Ref, are extendable, immutable and reflected choices.

Extendable $\nu\mathcal{C}(w)$ returns a choice name not occurring in w. For Fcs, $\mathsf{start}\nu\mathcal{C}(w, r)$ adds a new entry to w that creates a choice sequence with name $\nu\mathcal{C}(w)$ and restriction r (using the first kind of entry mentioned in Ex. 5). For Ref, $\mathsf{start}\nu\mathcal{C}(w, r)$ adds a new reference cell to w with name $\nu\mathcal{C}(w)$ and restriction r (using the first kind of operation mentioned in Ex. 6). In both cases, the properties are straightforward.

Immutable For Fcs, $\mathsf{freeze}(\delta, \kappa, w)$ extends w with a new entry (of the second kind from Ex. 5) that adds a new choice κ to the choice sequence δ. $\mathsf{mutable}(\delta, w)$ is always true since it is always possible to extend choice sequences with new choices. For Ref, $\mathsf{freeze}(\delta, \kappa, w)$ updates w by changing the content of the reference cell δ to κ if it is mutable and marking it as immutable; and $\mathsf{mutable}(\delta, w)$ checks that δ is still mutable in w.

reflected $\mathsf{Type}\mathcal{C}$ is $\wr\mathsf{Nat}$ in both cases, which is inhabited by $\boldsymbol{\kappa_0} \coloneqq \underline{0}$ and $\boldsymbol{\kappa_1} \coloneqq \underline{1}$. The other properties follow from the semantics of $\wr\mathsf{Nat}$. The use of $\wr$ is crucial because without it we would not be able to prove that choices obtained from worlds that compute to either $\boldsymbol{\kappa_0}$ or $\boldsymbol{\kappa_1}$ inhabit $\mathsf{Type}\mathcal{C}$, as reference cells can change value over time.

Next, we define the following two properties, which among other things allow proving (B) above. Sec. 6.2.1 shows how those properties can be proved for concrete instances of $\square$ with Beth bars. The first property requires that the choices corresponding to a name on which a restriction r is imposed, can always eventually be retrieved and that they satisfy r.

▶ **Definition 14** (Retrieving). *The modality* $\square$ *is called* retrieving *if:*
$\forall(w : \mathcal{W})(\delta : \mathcal{N})(n : \mathbb{N})(r : \mathsf{Res}).\mathsf{compatible}(\delta, w, r) \rightarrow \square_w(w'.r\ n\ \mathsf{choice?}(w', \delta, n))$

The second property states that if $\square_w P$ then P is true in an extension of w, and this for a specific class of worlds, namely those where only one choice has been made so far (possibly multiple times) and is still mutable. This property allows following a sequence of worlds where the same choice is picked for a given choice name.

▶ **Definition 15** (Choice-following). *The modality* $\square$ *is called* choice-following *if:*
$\forall(\delta : \mathcal{N})(w : \mathcal{W})(P : \mathcal{P}_w)(r : \mathsf{Res}).\mathsf{Sat}(w, \delta, r) \rightarrow \square_w P \rightarrow \exists^{\sqsubseteq}_w(w'.P\ w' \land \mathsf{Sat}(w', \delta, r))$
where $\mathsf{Sat}(w, \delta, r) \coloneqq \mathsf{compatible}(\delta, w, r) \land \mathsf{mutable}(\delta, w) \land \mathsf{OnlyChoice}(w, \delta, r_{.\mathsf{d}})$
and $\mathsf{OnlyChoice}(w, \delta, \kappa) \coloneqq \forall(n : \mathbb{N})(\kappa' : \mathcal{C}).\mathsf{choice?}(w, \delta, n) = \kappa' \rightarrow \kappa' = \kappa$.

Before we prove the negation of classical axioms, we first prove the following general result. Note the use of $\downarrow$ in Lem. 16, where $\downarrow(T{+}U)$ captures a classical reading of "or".

▶ **Lemma 16.** *Let $TT_{\mathcal{C}}^{\square}$ be a type system where $\mathcal{C}$ is a non-trivial, extendable, immutable and reflected set of choices and $\square$ is a retrieving, choice-following equality modality. Then, the followings hold (see not_lem.lagda for details):*

- $\forall(w:\mathcal{W}).\neg\,\square_{\mathsf{start}\nu\mathcal{C}(w,r)}\,(w'.(w' \vDash \Sigma\mathcal{C}(w)) \vee \forall_{w'}^{\sqsupseteq}(w''.\neg w'' \vDash \Sigma\mathcal{C}(w)))$
- $\forall(w:\mathcal{W}).\neg\mathsf{start}\nu\mathcal{C}(r,w) \vDash \downarrow(\Sigma\mathcal{C}(w){+}\neg\Sigma\mathcal{C}(w))$

where (1) $\Sigma\mathsf{choice}(\delta,\kappa) \coloneqq \boldsymbol{\Sigma} k{:}\mathsf{Nat}.((\delta(k)){=}\kappa{\in}\mathsf{Type}\mathcal{C})$; (2) $\Sigma\mathcal{C}(w) \coloneqq \Sigma\mathsf{choice}(\nu\mathcal{C}(w),\boldsymbol{\kappa_1})$; and (3) $r \coloneqq \langle res, d\rangle$ is the restriction where $res \coloneqq \lambda n,\kappa.(\kappa = \boldsymbol{\kappa_0} \vee \kappa = \boldsymbol{\kappa_1})$ and $d \coloneqq \boldsymbol{\kappa_0}$.

Proof. As the second statement is a straightforward consequence of the first, we only sketch a proof of the first. Let $w \in \mathcal{W}$. By extendability, we derive a new choice name δ, namely $\nu\mathcal{C}(w)$, and an extension $\mathsf{start}\nu\mathcal{C}(w,r)$ of w, where the only choice made so far for δ is $\boldsymbol{\kappa_0}$, and such that $\mathsf{mutable}(\delta,\mathsf{start}\nu\mathcal{C}(w,r))$, by immutability. We assume $\square_{\mathsf{start}\nu\mathcal{C}(w,r)}(w'.(w' \vDash \Sigma\mathcal{C}(w)) \vee \forall_{w'}^{\sqsupseteq}(w''.\neg w'' \vDash \Sigma\mathcal{C}(w)))$, and by the choice-following property we can derive a world $w' \sqsupseteq \mathsf{start}\nu\mathcal{C}(w,r)$, where the only choice made so far for δ is $\boldsymbol{\kappa_0}$, and such that $w' \vDash \Sigma\mathcal{C}(w)$ or $\forall_{w'}^{\sqsupseteq}(w''.\neg w'' \vDash \Sigma\mathcal{C}(w))$. We now derive a contradiction in both cases:

- $w' \vDash \Sigma\mathcal{C}(w)$: By the choice-following property and the meaning of $\Sigma\mathcal{C}(w)$, we derive that there exists $k \in \mathbb{N}$ such that $\delta(\underline{k})$ and $\boldsymbol{\kappa_1}$ are equal members of the type $\mathsf{Type}\mathcal{C}$ in some world $w'' \sqsupseteq w'$, where the only choice so far associated with δ is $\boldsymbol{\kappa_0}$. Since the modality is retrieving and choice-following, we can further derive a world $w''' \sqsupseteq w''$ where $\delta(\underline{k})$ computes to a choice κ satisfying r (therefore, either $\kappa = \boldsymbol{\kappa_0}$ or $\kappa = \boldsymbol{\kappa_1}$), and again where the only choice so far associated with δ is $\boldsymbol{\kappa_0}$. We derive that $\delta(\underline{k})$ computes to $\boldsymbol{\kappa_0}$, which cannot be equal to $\boldsymbol{\kappa_1}$, from which we obtain a contradiction.
- $\forall_{w'}^{\sqsupseteq}(w''.\neg w'' \vDash \Sigma\mathcal{C}(w))$: By immutability, we build the world $w'' = \mathsf{freeze}(\delta,\boldsymbol{\kappa_1},w') \sqsupseteq w'$, and get to assume $\neg w'' \vDash \Sigma\mathsf{choice}(\delta,\boldsymbol{\kappa_1})$. The reflected choice and retrieving modality entail $w'' \vDash \Sigma\mathsf{choice}(\delta,\boldsymbol{\kappa_1})$, from which we conclude a contradiction. Let us comment on the use of freeze. Assume that when "freezing" $\boldsymbol{\kappa_1}$, it is the n^{th} choice being made for δ in w''. Then, $(\delta\ \underline{n})$ computes to $\boldsymbol{\kappa_1}$ in w''. To derive $w'' \vDash \Sigma\mathsf{choice}(\delta,\boldsymbol{\kappa_1})$ we must prove that $(\delta\ \underline{n})$ computes to $\boldsymbol{\kappa_1}$, which using $\square_{\mathbf{3}}$, we must do in a $w''' \sqsupseteq w''$. Now, as some computations are time-sensitive (such as those involving references), without immutability it might not be that $(\delta\ \underline{n})$ computes to $\boldsymbol{\kappa_1}$ in w'''. ◀

Using Lem. 16, we can derive the negation of classical axioms such as LEM, or the Limited Principle of Omniscience (LPO) [7, p.9] (the above examples showed how to prove some of the assumptions in this lemma for instances of $\mathcal{C}$ and $\square$, and the others are described in Sec. 6.2.1, as they rely on a concrete instance of $\square$ with Beth bars).

▶ **Corollary 17** (Incompatibility with Classical Principles)**.** *Let $TT_{\mathcal{C}}^{\square}$ be a type system where $\mathcal{C}$ is a non-trivial, extendable, immutable and reflected set of choices and $\square$ is a retrieving, choice-following modality. Then, the following hold (see not_lem.lagda and not_lpo.lagda):*

- *¬LEM:* $\forall(w:\mathcal{W}).\neg w \vDash \boldsymbol{\Pi} P{:}\mathbb{U}_i.\downarrow(P{+}\neg P)$
- *¬LPO:* $\forall(w:\mathcal{W}).\neg w \vDash \boldsymbol{\Pi} f{:}\mathsf{Nat} \to \mathsf{Bool}.\downarrow\boldsymbol{\Sigma} n{:}\mathsf{Nat}.\uparrow(f\ n){+}\boldsymbol{\Pi} n{:}\mathsf{Nat}.\neg\uparrow(f\ n)$

For LPO, we further assume that choices are Booleans, i.e., that $\mathsf{Type}\mathcal{C}$ from Def. 12 is Bool*, that $\boldsymbol{\kappa_0}$ is* tt *and that $\boldsymbol{\kappa_1}$ is* ff *(see Appx. D for further details).*

5.2 Agnostic Theories

This section introduces the following general property of modalities that enables proving LEM, leading to "agnostic" instances of $\mathrm{TT}_{\mathcal{C}}^{\square}$, in the sense that they support classical reasoning.

▶ **Definition 18** (Jumping). *The modality $\Box_w$ is called* jumping *if:* $\forall(w:\mathcal{W})(P:\mathcal{P}_w).\forall^{\sqsubseteq}_{w}(w_1.\exists^{\sqsubseteq}_{w_1}(w_2.\,\Box_{w_2} P)) \to \Box_w P$

Note that, classically, the negation of the choice-following property can be read as: $\exists(\delta:\mathcal{N})(w:\mathcal{W})(P:\mathcal{P}_w)(r:\mathsf{Res}).\mathsf{Sat}(w,\delta,r) \wedge \Box_w P \wedge \forall^{\sqsubseteq}_{w}(w'.\mathsf{Sat}(w',\delta,r) \to \neg(P\ w'))$. Reading $\Box$ as "always eventually" this says that there exists a property P, which is always eventually true but there is no extension of the current world that satisfies Sat where P is true. Thus, not all possible futures have to be covered for a property to be "always eventually" true. The jumping property captures a similar behavior only requiring to prove that for all $w_1 \sqsupseteq w$ it is enough to exhibit one world $w_2 \sqsupseteq w_1$ where P is "always eventually" true, to derive that P is "always eventually" true. We now prove that $\mathrm{TT}^{\Box}_{\mathcal{C}}$ is compatible with LEM when instantiated with jumping modalities.

▶ **Lemma 19** (Compatibility with LEM). *Let $TT^{\Box}_{\mathcal{C}}$ be a type system where $\Box_w$ is a jumping equality modality. Then, the following holds (classically): $\forall(w:\mathcal{W}).w \vDash \mathbf{\Pi}P{:}\mathbb{U}_i.{\downarrow}(P{+}\neg P)$.*

Proof. By the semantics of the $\mathbf{\Pi}P{:}\mathbb{U}_i.{\downarrow}(P{+}\neg P)$, it is enough to prove that for all $w \in \mathcal{W}$ and $p \in \mathsf{Term}$ such that $w \vDash p{\in}\mathbb{U}_i$, then $\Box_w(w'.w' \vDash p \vee \forall^{\sqsubseteq}_{w'}(w''.\neg w'' \vDash p))$. By the jumping property, it is enough to prove $\forall^{\sqsubseteq}_{w}(w_1.\exists^{\sqsubseteq}_{w_1}(w_2.\,\Box_{w_2}(w_3.w_3 \vDash p \vee \forall^{\sqsubseteq}_{w_3}(w_4.\neg w_4 \vDash p))))$. Let $w_1 \sqsupseteq w$, and we prove $\exists^{\sqsubseteq}_{w_1}(w_2.\,\Box_{w_2}(w_3.w_3 \vDash p \vee \forall^{\sqsubseteq}_{w_3}(w_4.\neg w_4 \vDash p)))$. Using classical logic, we can then prove this by cases (see lem.lagda for further details):

- $\exists^{\sqsubseteq}_{w_1}(w_2.w_2 \vDash p)$: We obtain a $w_2 \sqsupseteq w_1$ such that $w_2 \vDash p$. We instantiate our conclusion using w_2, and must prove $\Box_{w_2}(w_3.w_3 \vDash p \vee \forall^{\sqsubseteq}_{w_3}(w_4.\neg w_4 \vDash p))$. Using $\Box_{\mathbf{4}}$ it is enough to prove $\forall^{\sqsubseteq}_{w_2}(w_3.w_3 \vDash p \vee \forall^{\sqsubseteq}_{w_3}(w_4.\neg w_4 \vDash p))$, which we prove by monotonicity of $w_2 \vDash p$.
- $\neg\exists^{\sqsubseteq}_{w_1}(w_2.w_2 \vDash p)$: We instantiate our conclusion using w_1, and show that $\Box_{w_1}(w_3.w_3 \vDash p \vee \forall^{\sqsubseteq}_{w_3}(w_4.\neg w_4 \vDash p))$. Using $\Box_{\mathbf{4}}$, it is enough to prove $\forall^{\sqsubseteq}_{w_1}(w_3.w_3 \vDash p \vee \forall^{\sqsubseteq}_{w_3}(w_4.\neg w_4 \vDash p))$. Therefore, assuming $w_3 \sqsupseteq w_1$, it remains to show $w_3 \vDash p \vee \forall^{\sqsubseteq}_{w_3}(w_4.\neg w_4 \vDash p)$, and since the right disjunct is provable, this contradicts our assumption. ◀

6 Bars

The notion of topological spaces of bars is typically used in possible worlds semantics to capture the intuitive notion of time progression and provide a forcing interpretation. Therefore, this section provides an abstract definition of this notion and establishes the connection to the aforementioned equality modalities. Concretely, we offer a notion of monotone bars that we then use to instantiate the equality modalities with.

6.1 Bar Spaces

The opens of a topological bar space are collections of worlds. To define a topological space of bars, one needs to describe the "shape" of the opens in the space through a predicate, which specifies when an open belongs to the space. Given a bar space, a bar in that space is an open (a collection of worlds) that satisfies the predicate specifying the space.

▶ **Definition 20** (Bars). *Let $\mathcal{O} := \mathcal{W} \to \mathbb{P}$ be the set of predicates on worlds, which we call opens, and let $\mathsf{BarProp} := \mathcal{W} \to \mathcal{O} \to \mathbb{P}$ be the set of predicates on opens. An open o is said to be a* bar *in $B \in \mathsf{BarProp}$ w.r.t. a world w if: (1) it satisfies $(B\ w\ o)$, (2) all its elements extend w, and (3) it is upward closed w.r.t. $\sqsubseteq$ (i.e., if $w_1 \sqsubseteq w_2$ and $(o\ w_1)$ then $(o\ w_2)$). We denote the set of all bars in B w.r.t. w by $\mathcal{B}^{w}_{B}$.*

Intuitively, given $B \in \mathsf{BarProp}$, $(B\ w\ o)$ specifies whether o "bars" the world w. We write $w \lhd o \in B$ for $(B\ w\ o)$, and $w' \in o$ for $(o\ w')$.

▶ **Definition 21** (Bar Spaces). *$B \in \mathsf{BarProp}$ is called a* bar space *if it satisfies the followings:*

- $\mathsf{isect}(B) \coloneqq \forall(w : \mathcal{W})(o_1, o_2 : \mathcal{O}).w \lhd o_1 \in B \to w \lhd o_2 \in B \to w \lhd (o_1 \cap o_2) \in B$, *where* $o_1 \cap o_1 \in \mathcal{O} \coloneqq \lambda w_0.\exists(w_1, w_2 : \mathcal{W}).w_1 \in o_1 \land w_2 \in o_2 \land w_1 \sqsubseteq w_0 \land w_2 \sqsubseteq w_0$.
- $\mathsf{union}(B) \coloneqq \forall(w : \mathcal{W})(b : \mathcal{B}^w_B)(i : \forall w' \sqsupseteq w.w' \in b \to \mathcal{B}^{w'}_B).w \lhd (\cup(i)) \in B$, *where* $\cup(i) \in \mathcal{O} \coloneqq \lambda w_0.\exists w_1 \sqsupseteq w.\exists(j : w_1 \in b).w_0 \in (i\ w_1\ j)$, *given* $i \in \forall w' \sqsupseteq w.w' \in b \to \mathcal{B}^{w'}_B$.
- $\mathsf{top}(B) \coloneqq \forall(w : \mathcal{W}).w \lhd (\top(w)) \in B$, *where* $\top(w) \in \mathcal{O} \coloneqq \lambda w_0.w \sqsubseteq w_0$.
- $\mathsf{non}\varnothing(B) \coloneqq \forall(w : \mathcal{W})(b : \mathcal{B}^w_B).\exists^{\sqsubseteq}_w(w'.w' \in b)$.
- $\mathsf{sub}(B) \coloneqq \forall(w_1, w_2 : \mathcal{W})(o : \mathcal{O}).w_1 \sqsubseteq w_2 \to w_1 \lhd o \in B \to w_2 \lhd (o\downharpoonleft_{w_2}) \in B$, *where* $o\downharpoonleft_w \in \mathcal{O} \coloneqq \lambda w_0.\exists(w_1 : \mathcal{W}).w_1 \in o \land w_1 \sqsubseteq w_0 \land w \sqsubseteq w_0$.

We denote by $\mathsf{BarSpace}$ *the set of all bar spaces.*

That is, a bar space B is a set of opens that is closed under binary intersections (i.e., $\mathsf{isect}(B)$) and arbitrary unions (i.e., $\mathsf{union}(B)$), contains a top element (i.e., $\mathsf{top}(B)$), all its elements are non-empty (i.e., $\mathsf{non}\varnothing(B)$), and is closed under subsets (i.e., $\mathsf{sub}(B)$).

For $w \in \mathcal{W}$, $P \in \mathcal{P}_w$, $B \in \mathsf{BarSpace}$, and $b \in \mathcal{B}^w_B$, we write $P \in b$ for $\forall w' \sqsupseteq w.w' \in b \to P\ w'$, i.e., P holds at the bar b, i.e., for all elements in b. Let $\exists\mathcal{B}^w_B \in \mathcal{P}_w \to \mathbb{P}$ be defined as $\lambda P.(\exists(b : \mathcal{B}^w_B).P \in b)$, i.e., that P holds in some bar of the space B. Using this definition, we next show that any bar space B induces an equality modality.

▶ **Proposition 22.** *If $B \in \mathsf{BarSpace}$ and $w \in \mathcal{W}$, then $\exists\mathcal{B}^w_B$ is an equality modality.*

Proof. Given the properties of a bar space, we derive corresponding properties for bars in $\mathcal{B}^w_B$, and in turn, the properties of an equality modality. In particular, $\mathsf{sub}(B)$ allows deriving $\Box_1$, $\mathsf{isect}(B)$ allows deriving $\Box_2$, $\mathsf{union}(B)$ allows deriving $\Box_3$, $\mathsf{non}\varnothing(B)$ allows deriving $\Box_5$, and $\mathsf{top}(B)$ allows deriving $\Box_4$. See Appx. C and bar.lagda for further details. ◀

Let $\mathrm{TT}^B_{\mathcal{C}}$ be the theory $\mathrm{TT}^{\Box}_{\mathcal{C}}$, where $\Box$ is derived from $B \in \mathsf{BarSpace}$ using Prop. 22.

▶ **Corollary 23.** *For any choice operator $\mathcal{C}$ and $B \in \mathsf{BarSpace}$, $TT^B_{\mathcal{C}}$ is a type system in the sense of Thm. 9.*

6.2 Examples of Bar Spaces

We next present two examples of bar spaces, namely Beth bars in Def. 25 and open bars in Def. 28, and use them to provide concrete instances for intuitionistic and agnostic «««< HEAD theories. ======= theories. »»»> e5e99da5628a63d71b0915560173a98d94cb6bb0 In particular, we show that the choice-following property, which is key in proving compatibility with LEM, is satisfied by Beth bars but not by open bars.

6.2.1 Beth Bars

As presented below, a Beth bar is defined so that for any infinite sequence of worlds ordered by $\sqsubseteq$, there exists a world in that sequence belonging to the bar. However, for Beth bars to satisfy the retrieving property presented in Def. 14, we must also ensure that for any choice name δ occurring in a world w in a chain, there is a $w' \sqsupseteq w$ in that chain such that $\mathsf{choice?}(w', \delta, n)$ is defined. To this end we introduce a predicate $\mathsf{progress} \in \mathcal{N} \to \mathcal{W} \to \mathcal{W} \to \mathbb{P}$, which we show how to instantiate in Exs. 26 and 27, as well as the concept of (progressing) chains:

▶ **Definition 24** (Chains & Barred Chains). *Let* $\mathsf{chain}(w)$ *be the set of sequences of worlds in* $\mathbb{N} \to \mathcal{W}$ *such that* $c \in \mathsf{chain}(w)$ *iff (1)* $w \sqsubseteq c\ 0$*, (2) for all* $i \in \mathbb{N}$*,* $c\ i \sqsubseteq c\ (i+1)$*; and (3)* c *is* progressing*, i.e.,* $\forall(\delta : \mathcal{N})(n : \mathbb{N})(r : \mathsf{Res}).\mathsf{compatible}(\delta, (c\ n), r) \to \exists m > n.\mathsf{progress}(\delta, (c\ n), (c\ m))$*. We say that a chain* $c \in \mathsf{chain}(w)$ *is* barred *by an* $o \in \mathcal{O}$*, denoted* $\mathsf{barredChain}(o, c)$*, if there exists a world* $w' \sqsubseteq (c\ n)$ *for some* $n \in \mathbb{N}$ *such that* $w' \in o$*.*

Using chains, we define Beth bars as follows:

▶ **Definition 25** (Beth Bars). *Beth bars are defined by the following bar predicate* $\mathsf{Beth} \coloneqq \lambda w.\lambda o.\forall(c : \mathsf{chain}(w)).\mathsf{barredChain}(o, c)$*, which is a bar space due to the properties of chains.*[4]

We now show through the following two examples how to define Beth bars, and how they induce a retrieving (Def. 14) and choice-following (Def. 15) modality, as required by Cor. 17.

▶ **Example 26** (Beth Bars & Free-Choice Sequences). Building up on Ex. 13, we present here an example where choices are free-choice sequences and bars are Beth bars, yielding an intuitionistic theory $\mathrm{TT}^{\mathsf{Beth}}_{\mathsf{Fcs}}$ (see worldInstanceCS.lagda and modInstanceCS.lagda for details). This is the theory presented in [5].

Progress For Fcs, $\mathsf{progress}(\delta, w_1, w_2)$ states that the state of the choice sequence δ in w_1 is a strict initial segment of the state of the choice sequence δ in w_2.

Retrieving We prove this property by exhibiting a bar that given a choice name δ and a $n \in \mathbb{N}$, requires its n^{th} choice to exist. We can prove that this forms a Beth bar thanks to the fact that chains are required to always eventually make progress.

Choice-following This property is true about Beth bars because they require *all* possible chains of worlds extending a given world w to be "barred" by the bar. Given a choice name δ that satisfies $\mathsf{Sat}(w, \delta, r)$, we can therefore pick a chain that repeatedly makes the same choice for δ, and obtain a world along that chain, which is at the bar.

▶ **Example 27** (Beth Bars & References). Building up on Ex. 13, we present here an example where choices are references and bars as Beth bars, yielding an intuitionistic theory $\mathrm{TT}^{\mathsf{Beth}}_{\mathsf{Ref}}$ (see worldInstanceRef.lagda and modInstanceRef.lagda for details).

Progress For Ref, $\mathsf{progress}(\delta, w_1, w_2)$ states that if a reference cell named δ holds t in w_1, then it must also hold t' in w_2, such that $t = t'$ if the cell is not mutable in w_1.

Retrieving This property is trivial to prove for references because we need to exhibit a bar, which given $\delta \in \mathcal{N}$ and $n \in \mathbb{N}$, requires δ's n^{th} choice to exist, which necessarily does because $\mathsf{choice?}(w, \delta, n)$ disregards its argument n and returns δ's current content in w.

Choice-following This property is proved as for free-choice sequences.

6.2.2 Open Bars

Open bars [6] are more straightforwardly defined and do not require the concept of chains.

▶ **Definition 28** (Open Bars). *Open bars are defined by the following bar predicate:* $\mathsf{Open} \coloneqq \lambda w.\lambda o.\forall^{\sqsubseteq}_{w}(w_1.\exists^{\sqsubseteq}_{w_1}(w_2.w_2 \in o))$*, which forms a bar space.*

The choice-following property does not hold for open bars due to the existential quantification in their definition, which allows different choices to be made. In fact, we can prove the negation of the choice-following property for open bars. Given $w_0 \in \mathcal{W}$, $\exists(\delta : \mathcal{N})(w : \mathcal{W})(P :$

[4] To be precise, to prove that Beth bars satisfy the $\mathsf{non}\varnothing$ property, we further require a function $\mathit{Ch}\mathsf{of}\mathcal{W}$ from $w \in \mathcal{W}$ to $\mathsf{chain}(w)$.

$\mathcal{P}_w)(r : \mathsf{Res}).\mathsf{Sat}(w, \delta, r) \wedge \square_w P \wedge \forall^{\sqsubseteq}_{w}(w'.\mathsf{Sat}(w', \delta, r) \to \neg(P\ w'))$ holds by instantiating δ with $\nu\mathcal{C}(w_0)$, w with $\mathsf{start}\nu\mathcal{C}(w_0, r)$, and P with $\lambda w'.\neg\mathsf{mutable}(\delta, w')$, where r restricts the choices to be either $\boldsymbol{\kappa_0}$ or $\boldsymbol{\kappa_1}$. Next we show that open bars induce a jumping modality, which is required to prove Lem. 19.

▶ **Example 29** (Open bars)**.** The agnostic theory $\mathrm{TT}_{\mathcal{C}}^{\mathsf{Open}}$, built upon open bars and an arbitrary choice operator $\mathcal{C}$, is compatible with classical logic (see lem.lagda). In [6] this theory was presented specifically for Fcs. As choices are irrelevant to prove Lem. 19, we can instantiate them with any suitable type, such as Ref or Fcs, and $\mathcal{W}$ can be any poset. It remains to show that Open satisfies the jumping property, which follows from the definition of open bars in terms of the existence of extensions of all extensions of the current world.

7 Conclusions and Related Works

This paper provides a generic extensional type theory incorporating various time-progressing elements along with a possible-worlds forcing interpretation parameterized by modalities, which when instantiated with topological spaces of bars leads to a general sheaf model. We have opted for a general framework, both in terms of the choice operators it can embed and its modality-based semantics. This is so that our system is abstract enough to capture other general models from the literature, as well as for it to contain a wide class of theories, allowing us to reason collectively about their (in)compatibility with classical reasoning. Much remains to be explored to fully utilize our general framework to study the relation with classical reasoning. For one, the choice and modality properties presented in Sec. 5 provide sufficient conditions for determining the relation of the corresponding theories to classical reasoning. Further work is required to establish whether they are also necessary.

Other sheaf models for choice-like concepts have been proposed in the literature. We mention a few concrete examples that are most closely related to our general framework. In [21], the author provides a sheaf model of predicate logic extended with non-constructive objects such as choice sequences, where formulas are interpreted w.r.t. a forcing interpretation parameterized by a site. In [42], the authors provide sheaf models for the intuitionistic theories LS [41] and CS [29] featuring choice sequences, where formulas are essentially interpreted w.r.t. a forcing interpretation over the Baire space. In [14; 13], the authors prove the *uniform* continuity of a Martin-Löf-like intensional type theory using forcing, and extract an algorithm that computes a uniform modulus of continuity. In [25] the authors introduce a forcing translation for the Calculus of Inductive Constructions (CIC) [33] extended with effects, which crucially preserves definitional equality. In [15], the independence of MP with Martin-Löf's type theory is established through a forcing interpretation, with sequences of Booleans as forcing conditions, by following Brouwer's argument that it is not decidable whether a choice sequence of Booleans will remain true for ever or become eventually false.

Related to our work is also the line of work, starting from [35], on building syntactic models of CIC, by translating CIC extended with logical principles and effects into itself. Using this technique, in [8], the authors present syntactic models through which properties can be added to negative types, allowing them to prove independent results, e.g., the independence of function extensionality in intentional type theory. In [36], the authors present a translation, where the resulting type theory features exceptions, which is consistent if the target theory is when exceptions are required to be caught locally. The authors use this translation to exhibit syntactic models of CIC which validate the independence of premise axiom, but not MP. In [38], the authors solve the problem of the restriction on exceptions in [36] by introducing a layered type theory with exceptions, which separates the consistency and effectful programming concerns. In [34] the authors present a syntactic presheaf model of CIC, which solves issues with dependent elimination present in [25], and allows extending CIC with MP. In [37], the authors go back to these dependent elimination issues and present a new version of call-by-push-value which allows combining effects and dependent types.

Also connected to our work are the generic modal theories recently introduced in [23; 22]. In [23] presents a Martin-Löf type theory extended with an S4-style necessity modality, and prove that the resulting theory satisfies normalization and decidability of type checking properties. To guarantee that the modality is an S4 necessity modality, this theory imposes restrictions on the terms that inhabit modalities, which are enforced through a "locking" mechanism. While this paper focuses on normalization on particular, our main focus is on deriving a modal type theory, which in particular satisfies monotonicity and locality to capture properties of choice operators. The generic modal type theory MTT presented in [22] goes one step further by supporting multiple interacting modalities. Both theories share the same goal of generically capturing hand-crafted modal theories, while we in particular focus on modalities "compatible" with choice operators.

References

[1] *Agda Wiki*. http://wiki.portal.chalmers.se/agda/pmwiki.php.

[2] Stuart F. Allen. "A Non-Type-Theoretic Definition of Martin-Löf's Types". In: *LICS*. IEEE Computer Society, 1987, pp. 215–221.

[3] Mark van Atten and Dirk van Dalen. "Arguments for the continuity principle". In: *Bulletin of Symbolic Logic* 8.3 (2002), pp. 329–347.

[4] Michael J. Beeson. *Foundations of Constructive Mathematics*. Springer, 1985.

[5] Mark Bickford, Liron Cohen, Robert L. Constable and Vincent Rahli. "Computability Beyond Church-Turing via Choice Sequences". In: *LICS 2018*. ACM, 2018, pp. 245–254. DOI: 10.1145/3209108.3209200.

[6] Mark Bickford, Liron Cohen, Robert L. Constable and Vincent Rahli. "Open Bar - a Brouwerian Intuitionistic Logic with a Pinch of Excluded Middle". In: *CSL*. Vol. 183. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 11:1–11:23. DOI: 10.4230/LIPIcs.CSL.2021.11.

[7] E. Bishop. *Foundations of constructive analysis*. Vol. 60. McGraw-Hill New York, 1967.

[8] Simon Boulier, Pierre-Marie Pédrot and Nicolas Tabareau. "The next 700 syntactical models of type theory". In: *CPP 2017*. ACM, 2017, pp. 182–194. DOI: 10.1145/3018610.3018620.

[9] Douglas Bridges and Fred Richman. *Varieties of Constructive Mathematics*. London Mathematical Society Lecture Notes Series. Cambridge University Press, 1987.

[10] L. E. J Brouwer. "Begründung der mengenlehre unabhängig vom logischen satz vom ausgeschlossen dritten. zweiter teil: Theorie der punkmengen". In: *Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam* 12.7 (1919).

[11] Paul J. Cohen. "The independence of the continuum hypothesis". In: *the National Academy of Sciences of the United States of America* 50.6 (Dec. 1963), pp. 1143–1148.

[12] Paul J. Cohen. "The independence of the continuum hypothesis II". In: *the National Academy of Sciences of the United States of America* 51.1 (Jan. 1964), pp. 105–110.

[13] Thierry Coquand and Guilhem Jaber. "A Computational Interpretation of Forcing in Type Theory". In: *Epistemology versus Ontology*. Vol. 27. Logic, Epistemology, and the Unity of Science. Springer, 2012, pp. 203–213. DOI: 10.1007/978-94-007-4435-6_10.

[14] Thierry Coquand and Guilhem Jaber. "A Note on Forcing and Type Theory". In: *Fundam. Inform.* 100.1-4 (2010), pp. 43–52. DOI: 10.3233/FI-2010-262.

[15] Thierry Coquand and Bassel Mannaa. "The Independence of Markov's Principle in Type Theory". In: *FSCD 2016*. Vol. 52. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, 17:1–17:18. DOI: 10.4230/LIPIcs.FSCD.2016.17.

[16] Karl Crary. “Type-Theoretic Methodology for Practical Programming Languages”. PhD thesis. Ithaca, NY: Cornell University, Aug. 1998.

[17] M. J. Cresswell and G. E. Hughes. *A New Introduction to Modal Logic*. Routledge, 1996.

[18] Dirk van Dalen. “An interpretation of intuitionistic analysis”. In: *Annals of mathematical logic* 13.1 (1978), pp. 1–43.

[19] Michael A. E. Dummett. *Elements of Intuitionism*. Second. Clarendon Press, 2000.

[20] VH Dyson and Georg Kreisel. *Analysis of Beth’s semantic construction of intuitionistic logic*. Stanford University. Applied Mathematics and Statistics Laboratories, 1961.

[21] Michael P. Fourman. “Notions of Choice Sequence”. In: *The L. E. J. Brouwer Centenary Symposium*. Vol. 110. Studies in Logic and the Foundations of Mathematics. Elsevier, 1982, pp. 91–105. DOI: https://doi.org/10.1016/S0049-237X(09)70125-9.

[22] Daniel Gratzer, G. A. Kavvos, Andreas Nuyts and Lars Birkedal. “Multimodal Dependent Type Theory”. In: *LICS*. ACM, 2020, pp. 492–506. DOI: 10.1145/3373718.3394736.

[23] Daniel Gratzer, Jonathan Sterling and Lars Birkedal. “Implementing a modal dependent type theory”. In: *Proc. ACM Program. Lang.* 3.ICFP (2019), 107:1–107:29. DOI: 10.1145/3341711.

[24] Arend Heyting. *Intuitionism: an introduction*. North-Holland Pub. Co., 1956.

[25] Guilhem Jaber, Gabriel Lewertowski, Pierre-Marie Pédrot, Matthieu Sozeau and Nicolas Tabareau. “The Definitional Side of the Forcing”. In: *LICS ’16*. ACM, 2016, pp. 367–376. DOI: 10.1145/2933575.2935320.

[26] Stephen C. Kleene and Richard E. Vesley. *The Foundations of Intuitionistic Mathematics, especially in relation to recursive functions*. North-Holland Publishing Company, 1965.

[27] Georg Kreisel. “A Remark on Free Choice Sequences and the Topological Completeness Proofs”. In: *J. Symb. Log.* 23.4 (1958), pp. 369–388. DOI: 10.2307/2964012.

[28] Georg Kreisel and Anne S. Troelstra. “Formal systems for some branches of intuitionistic analysis”. In: *Annals of Mathematical Logic* 1.3 (1970), pp. 229–387. DOI: http://dx.doi.org/10.1016/0003-4843(70)90001-X.

[29] Georg Kreisel and Anne S. Troelstra. “Formal systems for some branches of intuitionistic analysis”. In: *Annals of mathematical logic* 1.3 (1970), pp. 229–387.

[30] Saul A. Kripke. “Semantical Analysis of Intuitionistic Logic I”. In: *Formal Systems and Recursive Functions*. Vol. 40. Studies in Logic and the Foundations of Mathematics. Elsevier, 1965, pp. 92–130. DOI: https://doi.org/10.1016/S0049-237X(08)71685-9.

[31] Saul A. Kripke. “Semantical Analysis of Modal Logic I. Normal Propositional Calculi”. In: *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik* 9.5-6 (1963), pp. 67–96. DOI: 10.1002/malq.19630090502.

[32] Joan R. Moschovakis. “An intuitionistic theory of lawlike, choice and lawless sequences”. In: *Logic Colloquium’90: ASL Summer Meeting in Helsinki*. Association for Symbolic Logic. 1993, pp. 191–209.

[33] Christine Paulin-Mohring. “Introduction to the Calculus of Inductive Constructions”. In: *All about Proofs, Proofs for All*. Vol. 55. Studies in Logic (Mathematical logic and foundations). College Publications, Jan. 2015.

[34] Pierre-Marie Pédrot. “Russian Constructivism in a Prefascist Theory”. In: *LICS*. ACM, 2020, pp. 782–794. DOI: 10.1145/3373718.3394740.

[35] Pierre-Marie Pédrot and Nicolas Tabareau. “An effectful way to eliminate addiction to dependence”. In: *LICS 2017*. IEEE Computer Society, 2017, pp. 1–12. DOI: 10.1109/LICS.2017.8005113.

[36] Pierre-Marie Pédrot and Nicolas Tabareau. “Failure is Not an Option - An Exceptional Type Theory”. In: *ESOP 2018*. Vol. 10801. LNCS. Springer, 2018, pp. 245–271. DOI: 10.1007/978-3-319-89884-1_9.

[37] Pierre-Marie Pédrot and Nicolas Tabareau. “The fire triangle: how to mix substitution, dependent elimination, and effects”. In: *Proc. ACM Program. Lang.* 4.POPL (2020), 58:1–58:28. DOI: 10.1145/3371126.

[38] Pierre-Marie Pédrot, Nicolas Tabareau, Hans Jacob Fehrmann and Éric Tanter. “A reasonably exceptional type theory”. In: *Proc. ACM Program. Lang.* 3.ICFP (2019), 108:1–108:29. DOI: 10.1145/3341712.

[39] Andrew M Pitts. *Nominal Sets: Names and Symmetry in Computer Science, volume 57 of Cambridge Tracts in Theoretical Computer Science*. 2013.

[40] Anne S. Troelstra. "Choice Sequences and Informal Rigour". In: *Synthese* 62.2 (1985), pp. 217–227.

[41] Anne S. Troelstra. *Choice sequences: a chapter of intuitionistic mathematics*. Clarendon Press Oxford, 1977.

[42] Gerrit Van Der Hoeven and Ieke Moerdijk. "Sheaf models for choice sequences". In: *Annals of Pure and Applied Logic* 27.1 (1984), pp. 63–107. DOI: https://doi.org/10.1016/0168-0072(84)90035-6.

[43] Wim Veldman. "Understanding and Using Brouwer's Continuity Principle". In: *Reuniting the Antipodes — Constructive and Nonstandard Views of the Continuum*. Vol. 306. Synthese Library. Springer Netherlands, 2001, pp. 285–302. DOI: 10.1007/978-94-015-9757-9_24.

[44] Beth E. W. "Semantic Construction of Intuitionistic Logic". In: *Journal of Symbolic Logic* 22.4 (1957), pp. 363–365.

A $TT_{\mathcal{C}}^{\Box}$'s Inference Rules

In $TT_{\mathcal{C}}^{\Box}$, sequents are of the form $h_1, \ldots, h_n \vdash t : T$. Such a sequent denotes that, assuming $h_1, \ldots, h_n$, the term t is a member of the type T, and that therefore T is a type. The term t in this context is called the *extract* of T. Extracts are sometimes omitted when irrelevant to the discussion. An hypothesis h is of the form $x : A$, where the variable x stands for the name of the hypothesis and A its type. A rule is a pair of a conclusion sequent S and a list of premise sequents, $S_1, \cdots, S_n$ (written as usual using a fraction notation, with the premises on top). Let us now provide a sample of $TT_{\mathcal{C}}^{\Box}$'s key inference rules for some of its types not discussed above. In what follows, we write $a{\in}A$ for $a{=}a{\in}A$.

A.1 Products

The following rules are the standard $\mathbf{\Pi}$-elimination rule, $\mathbf{\Pi}$-introduction rule, type equality for $\mathbf{\Pi}$ types, and λ-introduction rule, respectively.

$$\frac{H, f : \mathbf{\Pi}x{:}A.B, J \vdash a{\in}A \quad H, f : \mathbf{\Pi}x{:}A.B, J, z : f(a){\in}B[x\backslash a] \vdash e : C}{H, f : \mathbf{\Pi}x{:}A.B, J \vdash e[z\backslash\star] : C} \qquad \frac{H, z : A \vdash b : B[x\backslash z] \quad H \vdash A{\in}\mathbb{U}_i}{H \vdash \lambda z.b : \mathbf{\Pi}x{:}A.B}$$

$$\frac{H \vdash A_1{=}A_2{\in}\mathbb{U}_i \quad H, y : A_1 \vdash B_1[x_1\backslash y]{=}B_2[x_2\backslash y]{\in}\mathbb{U}_i}{H \vdash \mathbf{\Pi}x_1{:}A_1.B_1{=}\mathbf{\Pi}x_2{:}A_2.B_2{\in}\mathbb{U}_i} \qquad \frac{H, z : A \vdash t_1[x_1\backslash z]{=}t_2[x_2\backslash z]{\in}B[x\backslash z] \quad H \vdash A{\in}\mathbb{U}_i}{H \vdash \lambda x_1.t_1{=}\lambda x_2.t_2{\in}\mathbf{\Pi}x{:}A.B}$$

Note that the last rule requires to prove that A is a type because the conclusion requires to prove that $\mathbf{\Pi}x{:}A.B$ is a type, and the first hypothesis only states that B is a type family over A, but does not ensures that A is a type.

The following rule is the standard function extensionality rule:

$$\frac{H, z : A \vdash f_1(z){=}f_2(z){\in}B[x\backslash z] \quad H \vdash A{\in}\mathbb{U}_i}{H \vdash f_1{=}f_2{\in}\mathbf{\Pi}x{:}A.B}$$

The following captures that equalities are closed under β-reductions:

$$\frac{H \vdash t[x\backslash s]{=}u{\in}T}{H \vdash (\lambda x.t)\ s{=}u{\in}T}$$

A.2 Sums

The following rules are the standard $\mathbf{\Sigma}$-elimination rule, $\mathbf{\Sigma}$-introduction rule, type equality for the $\mathbf{\Sigma}$ type, and pair-introduction rule, respectively.

$$\frac{H, p : \mathbf{\Sigma}x{:}A.B, a : A, b : B[x\backslash a], J[p\backslash\langle a,b\rangle] \vdash e : C[p\backslash\langle a,b\rangle]}{H, p : \mathbf{\Sigma}x{:}A.B, J \vdash \texttt{let}\ a, b = p\ \texttt{in}\ e : C} \qquad \frac{H \vdash a{\in}A \quad H \vdash b{\in}B[x\backslash a] \quad H, z : A \vdash B[x\backslash z]{\in}\mathbb{U}_i}{H \vdash \langle a, b\rangle : \mathbf{\Sigma}x{:}A.B}$$

$$\frac{H \vdash A_1{=}A_2{\in}\mathbb{U}_i \quad H, y : A_1 \vdash B_1[x_1\backslash y]{=}B_2[x_2\backslash y]{\in}\mathbb{U}_i}{H \vdash \mathbf{\Sigma}x_1{:}A_1.B_1{=}\mathbf{\Sigma}x_2{:}A_2.B_2{\in}\mathbb{U}_i} \qquad \frac{H, z : A \vdash B[x\backslash z]{\in}\mathbb{U}_i \quad H \vdash a_1{=}a_2{\in}A \quad H \vdash b_1{=}b_2{\in}B[x\backslash a_1]}{H \vdash \langle a_1, b_1\rangle{=}\langle a_2, b_2\rangle{\in}\mathbf{\Sigma}x{:}A.B}$$

The following rule states that equalities are closed under spread-reductions:

$$\frac{H \vdash u[x\backslash s; y\backslash t]{=}t_2{\in}T}{H \vdash \texttt{let}\ x, y = \langle s, t\rangle\ \texttt{in}\ u{=}t_2{\in}T}$$

A.3 Equality

The following rules are the standard equality-introduction rule:, equality-elimination rule, hypothesis rule, symmetry and transitivity rules, respectively.

$$\frac{H \vdash A{=}B{\in}\mathbb{U}_i \quad H \vdash a_1{=}b_1{\in}A \quad H \vdash a_2{=}b_2{\in}B}{H \vdash (a_1{=}a_2{\in}A){=}(b_1{=}b_2{\in}B){\in}\mathbb{U}_i} \qquad \frac{H, z : a{=}b{\in}A, J[z\backslash\star] \vdash e : C[z\backslash\star]}{H, z : a{=}b{\in}A, J \vdash e : C}$$

$$\frac{}{H, x : A, J \vdash x{\in}A} \qquad \frac{H \vdash b{=}a{\in}T}{H \vdash a{=}b{\in}T} \qquad \frac{H \vdash a{=}c{\in}T \quad H \vdash c{=}b{\in}T}{H \vdash a{=}b{\in}T}$$

The following rule allows fixing the extract of a sequent:

$$\frac{H \vdash t : T}{H \vdash t{\in}T}$$

The following rule allows rewriting with an equality in an hypothesis:

$$\frac{H, x : B, J \vdash t : C \quad H \vdash A{=}B{\in}\mathbb{U}_i}{H, x : A, J \vdash t : C}$$

A.4 Universes

Let i be a lower universe than j. The following rules are the standard universe-introduction rule and the universe cumulativity rule, respectively.

$$\frac{}{H \vdash \mathbb{U}_i{=}\mathbb{U}_i{\in}\mathbb{U}_j} \qquad \frac{H \vdash T{\in}\mathbb{U}_j}{H \vdash T{\in}\mathbb{U}_i}$$

A.5 Sets

The following rule is the standard set-elimination rule:

$$\frac{H, z : \{x : A \mid B\}, a : A, \boxed{b : B[x\backslash a]}, J[z\backslash a] \vdash e : C[z\backslash a]}{H, z : \{x : A \mid B\}, J \vdash e[a\backslash z] : C}$$

Note that we have used a new construct in the above rule, namely the hypothesis $\boxed{b : B[x\backslash a]}$, which is called a hidden hypothesis. The main feature of hidden hypotheses is that their names cannot occur in extracts (which is why we “box” those hypotheses). Intuitively, this is because the proof that B is true is discarded in the proof that the set type $\{x : A \mid B\}$ is true and therefore cannot occur in computations. Hidden hypotheses can be unhidden using the following rule:

$$\frac{H, x : T, J \vdash \star : a{=}b{\in}A}{H, \boxed{x : T}, J \vdash \star : a{=}b{\in}A}$$

which is valid since the extract is $\star$ and therefore does not make use of x.

The following rules are the standard set-introduction rule, type equality for the set type, and introduction rule for members of set types, respectively.

$$\frac{H \vdash a{\in}A \quad H \vdash B[x\backslash a] \quad H, z : A \vdash B[x\backslash z]{\in}\mathbb{U}_i}{H \vdash a : \{x : A \mid B\}}$$

$$\frac{H \vdash A_1{=}A_2{\in}\mathbb{U}_i \quad H, y : A_1 \vdash B_1[x_1\backslash y]{=}B_2[x_2\backslash y]{\in}\mathbb{U}_i}{H \vdash \{x_1 : A_1 \mid B_1\}{=}\{x_2 : A_2 \mid B_2\}{\in}\mathbb{U}_i}$$

$$\frac{H, z : A \vdash B[x\backslash z]{\in}\mathbb{U}_i \quad H \vdash a{=}b{\in}A \quad H \vdash B[x\backslash a]}{H \vdash a{=}b{\in}\{x : A \mid B\}}$$

A.6 Disjoint Unions

The following rules are the standard disjoint union-elimination rule, disjoint union-introduction rules, type equality for the disjoint union type, and injection-introduction rules, respectively.

$$\frac{\begin{array}{c} H, d : A{+}B, x : A, J[d\backslash \texttt{inl}(x)] \vdash t : C[d\backslash \texttt{inl}(x)] \\ H, d : A{+}B, y : B, J[d\backslash \texttt{inr}(y)] \vdash u : C[d\backslash \texttt{inr}(y)] \end{array}}{H, d : A{+}B, J \vdash \texttt{case}\ d\ \texttt{of}\ \texttt{inl}(x) \Rightarrow t\ |\ \texttt{inr}(y) \Rightarrow u : C}$$

$$\frac{H \vdash a : A \quad H \vdash B {\in} \mathbb{U}_i}{H \vdash \texttt{inl}(a) : A{+}B} \qquad \frac{H \vdash b : B \quad H \vdash A {\in} \mathbb{U}_i}{H \vdash \texttt{inr}(b) : A{+}B}$$

$$\frac{H \vdash A_1{=}A_2{\in}\mathbb{U}_i \quad H \vdash B_1{=}B_2{\in}\mathbb{U}_i}{H \vdash A_1{+}B_1{=}A_2{+}B_2{\in}\mathbb{U}_i}$$

$$\frac{H \vdash a_1{=}a_2{\in}A \quad H \vdash B{\in}\mathbb{U}_i}{H \vdash \texttt{inl}(a_1){=}\texttt{inl}(a_2){\in}A{+}B} \qquad \frac{H \vdash b_1{=}b_2{\in}B \quad H \vdash A{\in}\mathbb{U}_i}{H \vdash \texttt{inr}(b_1){=}\texttt{inr}(b_2){\in}A{+}B}$$

The following rules state that PERs are closed under decide-reductions:

$$\frac{H \vdash t[x\backslash s]{=}t_2{\in}T}{H \vdash (\texttt{case}\ \texttt{inl}(s)\ \texttt{of}\ \texttt{inl}(x) \Rightarrow t\ |\ \texttt{inr}(y) \Rightarrow u){=}t_2{\in}T}$$

$$\frac{H \vdash u[y\backslash s]{=}t_2{\in}T}{H \vdash (\texttt{case}\ \texttt{inr}(s)\ \texttt{of}\ \texttt{inl}(x) \Rightarrow t\ |\ \texttt{inr}(y) \Rightarrow u){=}t_2{\in}T}$$

A.7 Time-Quotients

The following rules are the introduction and type equality rules for the time-quotienting type. Due to their nature, we do not provide an elimination rule. Note that in practice more terms that the ones in A can be shown to be in $\wr A$. For example, given a choice name δ with a restriction that constrains its choices to be elements of A, we can prove that $(\delta\ \underline{n})$, for $n \in \mathbb{N}$ is in $\wr A$, even though $(\delta\ \underline{n})$ might change over time. Devising such rules is left for future work.

$$\frac{H \vdash a : A}{H \vdash a : \wr A} \qquad \frac{H \vdash A{=}B{\in}\mathbb{U}_i}{H \vdash \wr A{=}\wr B{\in}\mathbb{U}_i} \qquad \frac{H \vdash a{=}b{\in}A}{H \vdash a{=}b{\in}\wr A}$$

B Equality Modalities

As mentioned in Sec. 4, our forcing interpretation relies on a pair of a modality $\square$ and a dependent modality $\boxdot$. The version of this interpretation presented there is a consequence of the formal definition, which involves both modalities. Let us now describe this definition in this section (see forcing.lagda for further details). We define in Fig. 3 an $w \vDash_l T_1{\equiv}T_2$ set, which compared to the one presented in Sec. 4, contains a universe level annotation l, which is simply here a $\mathbb{N}$. In addition, that figure defines a recursive function $w \vDash_l a{\equiv}b{\in}e$, which recurses over $e \in w \vDash_l T_1{\equiv}T_2$, and again contains a universe level annotation compared to the one presented in Sec. 4. This inductive-recursive definition is defined recursively over universe levels. The function $w \vDash a{\equiv}b{\in}T$ presented in Sec. 4 can then be defined as $\exists(l : \mathbb{N})(e : w \vDash_l T{\equiv}T).w \vDash a{\equiv}b{\in}e$.

Let us now formally introduce the dependent modality $\boxdot_w^i$, along with its properties. First, we introduce a dependent version of the set $\mathcal{P}_w$ as follows: the collection of predicates

Figure 3 Inductive-Recursive Forcing Interpretation

Inductive definition:

$$
\begin{array}{rl}
w \vDash_l T_1{\equiv}T_2 ::= & \texttt{NAT}{\equiv}(T_1 \Downarrow_w \mathsf{Nat} \wedge T_2 \Downarrow_w \mathsf{Nat}) \\
\mid \texttt{PI}{\equiv} & \left(\begin{array}{l} \exists(x:\mathsf{Var})(A_1,A_2,B_1,B_2:\mathsf{Term})(e:\forall_w^{\sqsubseteq}(w'.w' \vDash_l A_1{\equiv}A_2)). \\ \qquad T_1 \Downarrow_w \mathbf{\Pi}x{:}A_1.B_1 \wedge T_2 \Downarrow_w \mathbf{\Pi}x{:}A_2.B_2 \\ \quad \wedge\ \forall_w^{\sqsubseteq}(w'.\forall(a,b:\mathsf{Term}).w' \vDash_l a{\equiv}b{\in}(e\ w') \to w' \vDash_l B_1[x\backslash a]{\equiv}B_2[x\backslash b]) \end{array}\right) \\
\mid \texttt{SUM}{\equiv} & \left(\begin{array}{l} \exists(A_1,A_2,B_1,B_2:\mathsf{Term})(e:\forall_w^{\sqsubseteq}(w'.w' \vDash_l A_1{\equiv}A_2)). \\ \qquad T_1 \Downarrow_w \mathbf{\Pi}x{:}A_1.B_1 \wedge T_2 \Downarrow_w \mathbf{\Pi}x{:}A_2.B_2 \\ \quad \wedge\ \forall_w^{\sqsubseteq}(w'.\forall(a,b:\mathsf{Term}).w' \vDash_l a{\equiv}b{\in}(e\ w') \to w' \vDash_l B_1[x\backslash a]{\equiv}B_2[x\backslash b]) \end{array}\right) \\
\mid \texttt{SET}{\equiv} & \left(\begin{array}{l} \exists(A_1,A_2,B_1,B_2:\mathsf{Term})(e:\forall_w^{\sqsubseteq}(w'.w' \vDash_l A_1{\equiv}A_2)). \\ \qquad T_1 \Downarrow_w \mathbf{\Pi}x{:}A_1.B_1 \wedge T_2 \Downarrow_w \mathbf{\Pi}x{:}A_2.B_2 \\ \quad \wedge\ \forall_w^{\sqsubseteq}(w'.\forall(a,b:\mathsf{Term}).w' \vDash_l a{\equiv}b{\in}(e\ w') \to w' \vDash_l B_1[x\backslash a]{\equiv}B_2[x\backslash b]) \end{array}\right) \\
\mid \texttt{UNION}{\equiv} & \left(\begin{array}{l} \exists(A_1,A_2,B_1,B_2:\mathsf{Term}). \\ \qquad T_1 \Downarrow_w A_1{+}B_1 \qquad\qquad \wedge\ T_2 \Downarrow_w A_2{+}B_2 \\ \quad \wedge\ \forall_w^{\sqsubseteq}(w'.w' \vDash_l A_1{\equiv}A_2) \wedge \forall_w^{\sqsubseteq}(w'.w' \vDash_l B_1{\equiv}B_2) \end{array}\right) \\
\mid \texttt{EQ}{\equiv} & \left(\begin{array}{l} \exists(a_1,a_2,b_1,b_2,A,B:\mathsf{Term})(e:\forall_w^{\sqsubseteq}(w'.w' \vDash_l A{\equiv}B)). \\ \qquad T_1 \Downarrow_w a_1{=}a_2{\in}A \qquad\qquad \wedge\ T_2 \Downarrow_w b_1{=}b_2{\in}B \\ \quad \wedge\ \forall_w^{\sqsubseteq}(w'.w' \vDash_l a_1{\equiv}b_1{\in}(e\ w')) \wedge \forall_w^{\sqsubseteq}(w'.w' \vDash_l a_2{\equiv}b_2{\in}(e\ w')) \end{array}\right) \\
\mid & \texttt{QTIME}{\equiv}(\exists(A,B:\mathsf{Term}).T_1 \Downarrow_w \wr A \wedge T_2 \Downarrow_w \wr B \wedge \forall_w^{\sqsubseteq}(w'.w' \vDash_l A{\equiv}B)) \\
\mid & \texttt{MOD}{\equiv}(\Box_w(w'.w' \vDash_l T_1{\equiv}T_2)) \\
\mid & \texttt{UNIV}{\equiv}(\exists(j<l).T_1 \Downarrow_w \mathbb{U}_j \wedge T_2 \Downarrow_w \mathbb{U}_j)
\end{array}
$$

Recursive function:

$$
\begin{array}{l}
w \vDash_l t{\equiv}t'{\in}\texttt{NAT}{\equiv}(c_1,c_2) \\
\quad := \Box_w(w'.\exists(n:\mathbb{N}).t \Downarrow_{w'} \underline{n} \wedge t' \Downarrow_{w'} \underline{n}) \\
w \vDash_l t{\equiv}t'{\in}\texttt{PI}{\equiv}(x,A_1,A_2,B_1,B_2,e,c_1,c_2,f) \\
\quad := \Box_w(w'.\forall(a_1,a_2:\mathsf{Term})(i:w' \vDash_l a_1{\equiv}a_2{\in}(e\ w')).w' \vDash_l (t\ a_1){\equiv}(t'\ a_2){\in}(f\ w'\ a_1\ a_2\ i)) \\
w \vDash_l t{\equiv}t'{\in}\texttt{SUM}{\equiv}(x,A_1,A_2,B_1,B_2,e,c_1,c_2,f) \\
\quad := \Box_w\left(w'.\begin{array}{l} \exists(a_1,a_2,b_1,b_2:\mathsf{Term})(i:w' \vDash_l a_1{\equiv}a_2{\in}(e\ w')). \\ \quad w' \vDash_l b_1{\equiv}b_2{\in}(f\ w'\ a_1\ a_2\ i) \wedge t \Downarrow_{w'} \langle a_1,b_1\rangle \wedge t' \Downarrow_{w'} \langle a_2,b_2\rangle \end{array}\right) \\
w \vDash_l t{\equiv}t'{\in}\texttt{SET}{\equiv}(x,A_1,A_2,B_1,B_2,e,c_1,c_2,f) \\
\quad := \Box_w(w'.\exists(b_1,b_2:\mathsf{Term})(i:w' \vDash_l t{\equiv}t'{\in}(e\ w')).w' \vDash_l b_1{\equiv}b_2{\in}(f\ w'\ t\ t'\ i)) \\
w \vDash_l t{\equiv}t'{\in}\texttt{UNION}{\equiv}(A_1,A_2,B_1,B_2,c_1,c_2,e,f) \\
\quad := \Box_w\left(w'.\exists(u,v:\mathsf{Term}).\begin{array}{l} (t \Downarrow_{w'} \texttt{inl}(u) \wedge t' \Downarrow_{w'} \texttt{inl}(v) \wedge w' \vDash_l u{\equiv}v{\in}(e\ w')) \\ \vee\ (t \Downarrow_{w'} \texttt{inr}(u) \wedge t' \Downarrow_{w'} \texttt{inr}(v) \wedge w' \vDash_l u{\equiv}v{\in}(f\ w')) \end{array}\right) \\
w \vDash_l t{\equiv}t'{\in}\texttt{EQ}{\equiv}(a_1,a_2,b_1,b_2,A,B,e,c_1,c_2,i_1,i_2) \\
\quad := \Box_w(w'.w' \vDash_l a_1{\equiv}a_2{\in}(e\ w')) \\
w \vDash_l t{\equiv}t'{\in}\texttt{QTIME}{\equiv}(A,B,c_1,c_2,e) \\
\quad := \Box_w(w'.(\lambda a,b.\exists(c,d:\mathsf{Value}).a \sim_w c \wedge b \sim_w d \wedge w \vDash_l c{\equiv}d{\in}(e\ w'))^+\ t\ t') \\
w \vDash_l t{\equiv}t'{\in}\texttt{MOD}{\equiv}(i) \\
\quad := \boxdot_w^i(w'.\lambda(j:w' \vDash_l T_1{\equiv}T_2).w' \vDash_l t{\equiv}t'{\in}j), \text{where } i \text{ is a proof of } \Box_w(w'.w' \vDash_l T_1{\equiv}T_2) \\
w \vDash_l t{\equiv}t'{\in}\texttt{UNIV}{\equiv}(j,c_1,c_2) \\
\quad := w \vDash_j t{\equiv}t', \text{where } j<l
\end{array}
$$

in $\forall w' \sqsupseteq w.P\ w' \to \mathbb{P}$ for $P \in \mathcal{P}_w$, is denoted $\mathcal{P}_w^P$. The dependent modality $\boxdot_w^i \in \mathcal{P}_w^P \to \mathbb{P}$, where $P \in \mathcal{P}_w \to \mathbb{P}$ and $i \in \Box_w P$, is called a *dependent equality modality*

Note that as for members of $\mathcal{P}_w$, due to $\sqsubseteq$'s transitivity, if $Q \in \mathcal{P}^P_w$, where $P \in \mathcal{P}_w$, then for every $w' \sqsupseteq w$, it naturally extends to a predicate in $\mathcal{P}^P_{w'}$. Also, note that property $\Box_{\mathbf{1}}$ in Def. 8 can be viewed as defining a lifting operator $\uparrow_{w'} i$, which returns a $\Box_{w'} P$, given a $w' \sqsupseteq w$ and $i \in \Box_w P$ as specified there. This lifting operator will be used to state $\boxdot^i_w$'s properties.

We can now state $\boxdot^i_w$'s properties, which are counterparts of properties $\Box_{\mathbf{1}}, \Box_{\mathbf{2}}, \Box_{\mathbf{3}}$:

- $\boxdot_{\mathbf{1}}$: monotonicity of $\boxdot$:

$$\forall (w : \mathcal{W})(P : \mathcal{P}_w)(Q : \mathcal{P}^P_w)(i : \Box_w P).\forall w' \sqsupseteq w.\ \boxdot^i_w\, Q \to \boxdot^{\uparrow_{w'} i}_{w'} Q$$

  This property defines a lifting operator $\uparrow_{w'} j$, which returns a $\boxdot^{\uparrow_{w'} i}_{w'} Q$, given a $w' \sqsupseteq w$ and $j \in \boxdot^i_w Q$ as specified above.
- $\boxdot_{\mathbf{2}}$: A version of the distribution axiom:

$$\begin{aligned}&\forall (w : \mathcal{W})(P_1, P_2, P_3 : \mathcal{P}_w)(Q_1 : \mathcal{P}^{P_1}_w)(Q_2 : \mathcal{P}^{P_2}_w)(Q_3 : \mathcal{P}^{P_3}_w)(i_1 : \Box_w P_1)(i_2 : \Box_w P_2)(i_3 : \Box_w P_3).\\ &\quad (\forall^{\sqsubseteq}_w (w')\forall (p_1 : P_1\ w')(p_2 : P_2\ w')(p_3 : P_3\ w').Q_1\ w'\ p_1 \to Q_2\ w'\ p_2 \to Q_3\ w'\ p_3)\\ &\quad \to \boxdot^{i_1}_w Q_1 \to \boxdot^{i_2}_w Q_2 \to \boxdot^{i_3}_w Q_3\end{aligned}$$

- $\boxdot_{\mathbf{3}}$: $\boxdot$ follows from $\Box\boxdot$, i.e., a dependent version of C4:

$$\forall (w : \mathcal{W})(P : \mathcal{P}_w)(Q : \mathcal{P}^P_w)(i : \Box_w P).\ \Box_w\ (w'.\ \boxdot^{\uparrow_{w'} i}_{w'}\ Q) \to \boxdot^i_w Q$$

In addition, the two modalities $\Box$ and $\boxdot$ are required to satisfy the following properties that allow deriving one from other in some contexts:

- $\boxdot$ follows from $\Box$:

$$\forall (w : \mathcal{W})(P : \mathcal{P}_w)(Q : \mathcal{P}^P_w).\ \Box_w\ (w'.\forall (p : P\ w').Q\ w'\ p) \to \forall (i : \Box_w P).\ \boxdot^i_w\ Q$$

- $\Box$ follows from $\boxdot$:

$$\forall (w : \mathcal{W})(P, R : \mathcal{P}_w)(Q : \mathcal{P}^P_w)(i : \Box_w P).\forall^{\sqsubseteq}_w (w')\forall (p : P\ w').Q\ w'\ p \to R\ w' \to \boxdot^i_w Q \to \Box_w R$$

C Properties of the Bar Space

The properties of bar spaces presented in Def. 21 allow deriving corresponding bars as follows:

- Intersection of bars: Given a bar predicate $B \in \mathsf{BarProp}$ such that $\mathsf{isect}(B)$, and two bars $b_1, b_2 \in \mathcal{B}^w_B$ for some world w, then $b_1 \cap b_2 \in \mathcal{B}^w_B$.
  - $w \triangleleft (b_1 \cap b_2) \in B$ follows from $\mathsf{isect}(B)$
  - the two other properties of bars follow from the definition of $b_1 \cap b_2$.
- Union of bars: Given a bar predicate $B \in \mathsf{BarProp}$ such that $\mathsf{union}(B)$, and a family of bars $i \in \forall w' \sqsupseteq w.w' \in b \to \mathcal{B}^{w'}_B$ for some world w, then $\cup(i) \in \mathcal{B}^w_B$.
  - $w \triangleleft (\cup(i)) \in B$ follows from $\mathsf{union}(B)$
  - the two other properties of bars follow from the definition of $\cup(i)$.
- Top bar: Given a bar predicate $B \in \mathsf{BarProp}$, such that $\mathsf{top}(B)$, then $\top(w) \in \mathcal{B}^w_B$.
  - $w \triangleleft (\top(w)) \in B$ follows from $\mathsf{top}(B)$
  - the two other properties of bars follow from the definition of $\top(w)$.
- Sub-bar: Given a bar predicate $B \in \mathsf{BarProp}$ such that $\mathsf{sub}(B)$, and a bar $b \in \mathcal{B}^w_B$ for some world w, then $b \downharpoonright_{w'} \in \mathcal{B}^{w'}_B$ for any $w' \sqsupseteq w$.

- $w \triangleleft (b\downharpoonright_{w'}) \in B$ follows from $\mathsf{sub}(B)$
- the two other properties of bars follow from the definition of $b\downharpoonright_{w'}$.

As mentioned in Prop. 22, $\exists\mathcal{B}^{w}_{B}$, where $B \in \mathsf{BarSpace}$ and $w \in \mathcal{W}$, is an equality modality. We can derive the properties (see Def. 8) of this modality as follows:

- To prove $\square_{\mathbf{1}}$, we need to derive $\exists\mathcal{B}^{w'}_{B}(P)$ from $\exists\mathcal{B}^{w}_{B}(P)$, where $w' \sqsupseteq w$. As $\exists\mathcal{B}^{w}_{B}(P)$ gives us a bar $b \in \mathcal{B}^{w}_{B}$, we can instantiate our conclusion with $b\downharpoonright_{w'}$.
- To prove $\square_{\mathbf{2}}$, we need to derive $\exists\mathcal{B}^{w}_{B}(Q)$ from $\exists\mathcal{B}^{w}_{B}(\lambda w'.P\ w' \to Q\ w')$ and $\exists\mathcal{B}^{w}_{B}(P)$. Our first assumption gives us a bar $b_1 \in \mathcal{B}^{w}_{B}$ and our second assumption gives us a bar $b_2 \in \mathcal{B}^{w}_{B}$. We can then instantiate our conclusion with $b_1 \cap b_2$.
- To prove $\square_{\mathbf{3}}$, we need to derive $\exists\mathcal{B}^{w}_{B}(P)$ from $\exists\mathcal{B}^{w}_{B}(\lambda w'.\exists\mathcal{B}^{w'}_{B}(P))$. This assumption gives us a bar $b \in \mathcal{B}^{w}_{B}$ along with a function $i \in (\lambda w'.\exists\mathcal{B}^{w'}_{B}(P)) \in b$. We can then instantiate our conclusion with $\cup(i)$.
- To prove $\square_{\mathbf{4}}$, we need to derive $\exists\mathcal{B}^{w}_{B}(P)$ from $\forall^{\sqsubseteq}_{w}(P)$. We can then instantiate our conclusion using $\mathsf{T}(w)$, and have to prove $P \in \mathsf{T}(w)$, which trivially follows from $\forall^{\sqsubseteq}_{w}(P)$.
- To prove $\square_{\mathbf{5}}$, we need to derive P from $\exists\mathcal{B}^{w}_{B}(\lambda w'.P)$. This assumption gives us a bar b such that $(\lambda w'.P) \in b$. From $\mathsf{non}\varnothing(B)$, we obtain a $w' \sqsupseteq w$ such that $w' \in b$. We can then instantiate $(\lambda w'.P) \in b$ with w', and we obtain P since it does not depend on a world.

D Classical Axioms

As mentioned in Cor. 17, we can prove the negation of LEM and LPO assuming a non-trivial, extendable, immutable and reflected set of choices $\mathcal{C}$ and a retrieving, choice-following equality modality $\square_w$. For LPO, we further assume that choices are Booleans, i.e., that $\mathsf{Type}\mathcal{C}$ from Def. 12 is Bool, that $\boldsymbol{\kappa_0}$ is tt and that $\boldsymbol{\kappa_1}$ is ff. This is due to the fact that LPO is stated in terms of a function in $\mathsf{Nat} \to \mathsf{Bool}$, which we instantiate with a choice sequence whose choice are restricted to Booleans to prove its negation. A consequence of this is that choices can be instantiated using free-choice sequences but not using references. A free choices sequence name δ occurring in world with a restriction constraining its choices to be Booleans will be of type $\mathsf{Nat} \to \mathsf{Bool}$ because choices do not change over time. However, a reference name δ occurring in world with a restriction constraining its choices to be Booleans will be of type $\mathsf{Nat} \to \wr\mathsf{Bool}$ because its choices can change over time. However, we can prove the following alternative version of the negation of LPO (see not_lpo_qtbool.lagda for details):

$$\forall(w:\mathcal{W}).\neg w \vDash \mathbf{\Pi} f{:}\mathsf{Nat} \to \wr\mathsf{Bool}.\downarrow\mathbf{\Sigma} n{:}\mathsf{Nat}.\wr(f\ n) + \mathbf{\Pi} n{:}\mathsf{Nat}.\neg\wr(f\ n)$$

where $\wr(T) \coloneqq T{=}\mathsf{tt}{\in}\wr\mathsf{Bool}$.

Furthermore, using similar than the ones presented in Lem. 16, we can prove the negation of Markov's Principles (see not_mp.lagda for details):

$$\forall(w:\mathcal{W}).\neg w \vDash \mathbf{\Pi} f{:}\mathsf{Nat} \to \mathsf{Bool}.(\neg\mathbf{\Pi} n{:}\mathsf{Nat}.\neg\uparrow(f\ n)) \to \downarrow\mathbf{\Sigma} n{:}\mathsf{Nat}.\uparrow(f\ n)$$

In addition to requiring that choices are Booleans as for LPO, the proof also requires that $\mathsf{mutable}$ is always true (even if we had used $\wr\mathsf{Bool}$ instead of Bool), which only holds about free-choice sequences but not references.

E Further Bars & Modalities

Let us present here another bar space, which allows capturing traditional Kripke semantics. Let $\mathsf{Kripke} \coloneqq \lambda w.\lambda o.\forall^{\sqsubseteq}_{w}(w'.w' \in o)$, be the predicate that given a world w requires opens to contain all extensions of w. This also form a bar space as proved in barKripke.lagda. According to Prop. 22, this space leads in turn to an equality modality, which captures traditional a Kripke semantics. However, as proved in kripkeCsNotRetrieving.lagda, this modality is not retrieving when choices are free-choice sequences, and therefore does not allow deriving the negation of classical axioms using Cor. 17. It is however retrieving when choices are references because reference cells are always filled with a value. We can then prove that the resulting equality modality along with references as choices satisfy all the properties required for Cor. 17 (see modInstanceKripkeRefBool.lagda).