Evidenced Frames: A Unifying Framework Broadening Realizability Models

Liron Cohen, Étienne Miquey, Ross Tate

Evidenced Frames: A Unifying Framework Broadening Realizability Models

Liron Cohen
Ben-Gurion University
cliron@cs.bgu.ac.il

Étienne Miquey
ÉNS de Lyon
etienne.miquey@ens-lyon.fr

Ross Tate
Cornell University
ross@cs.cornell.edu

***Abstract*—Constructive foundations have for decades been built upon realizability models for higher-order logic and type theory. However, traditional realizability models have a rather limited notion of computation, which only supports non-termination and avoids many other commonly used effects. Work to address these limitations has typically *overlaid* structure on top of existing models, such as by using powersets to represent non-determinism, but kept the realizers themselves deterministic. This paper alternatively addresses these limitations by making the structure *underlying* realizability models more flexible. To this end, we introduce *evidenced frames*: a general-purpose framework for building realizability models that support diverse effectful computations. We demonstrate that this flexibility permits models wherein the realizers themselves can be *effectful*, such as λ-terms that can manipulate state, reduce non-deterministically, or fail entirely. Beyond the broader notions of computation, we demonstrate that evidenced frames form a unifying framework for (realizability) models of higher-order dependent predicate logic. In particular, we prove that evidenced frames are complete with respect to these models, and that the existing completeness construction for implicative algebras—another foundational framework for realizability—factors through our simpler construction. As such, we conclude that evidenced frames offer an ideal domain for unifying and broadening realizability models.**

I. Introduction

Constructive foundations have for decades been built upon realizability models for logic and type theory [1], [2]. Realizability captures the ability to extract (computable) content from proofs; realizers are "codes" that represent programs, and a proposition is "true" if and only if it has a realizer. Most realizability models are based on partial combinatory algebras (PCA) [3], [4], which formalize the key components of computation that support the proofs-as-programs correspondence [5] (most famously exhibited by the BHK interpretation). Concretely, from a PCA one can construct a realizability model of higher-order dependent logic—namely, a realizability *tripos*—via a standard construction [2], and from there on one can construct a realizability model of set theory and (extensional, impredicative) dependent type theory—namely, a realizability topos—via the standard tripos-to-topos construction [6].[1] These realizability models are intrinsically bound to a limited notion of computation. First, they assume that all internal computations are deterministic; and second, they can only support non-terminating computations, thus hindering our ability to reason about the role of other computational effects in realizability.

This paper addresses these limitations by presenting a general-purpose framework for building realizability models, which, as we demonstrate, can soundly accommodate diverse effectful computations. In a previous work [8], the underlying notion of a PCA was generalized into *relational* combinatory algebras (RCA) and *stateful* combinatory algebras (SCA) to directly model non-deterministic and stateful computation, respectively. It was there shown that these more general structures of computation give rise to a tripos providing a non-traditional realizability model of (dependent) higher-order logic. While these models are still constructive in that every proof is evidenced by some computable realizer, they can differ substantially from traditional realizability models in that they negate properties often associated with realizability and constructivism despite still being computational in nature. For example, while every PCA-based tripos models Countable Choice, this fact relies implicitly on the assumption that the computational system is deterministic, and, as such, a non-deterministic computational system whose corresponding model internally refutes Countable Choice was presented in [8]. These were first steps towards better understanding the role of effectful computation in realizability models. Exploring this broader notion of realizability has the potential to provide a deeper understanding of the role of computation in mathematical foundations. As noted by Bauer in [9]:

> "A comprehensive account of computable mathematics should encompass a rich spectrum of models of computability. ... By focusing on just a couple of models in the name of simplicity and historical importance of Turing machines, an opportunity for the view of a larger picture is missed."

In the current paper, we identify a unifying abstract construct—which we call *evidenced frames*—underlying the construction of traditional realizability triposes derived from PCAs and non-traditional realizability triposes such as in [8]. The goal of this framework is to capture the concept of realizability without making assumptions about the computational system. As such, evidenced frames have no notion of reduction—they just have a notion of when an abstract piece of *evidence* serves as proof that one abstract proposition entails

[1]To be precise, obtaining models of set theory, i.e. IZF, requires some additional properties [7].

another, plus constructors for combining pieces of evidence together (but with no equational theory). This structure is sufficient to build models such as triposes and toposes with the same key property of PCA-based models: for every entailment or function there is some piece of evidence supporting the proof or operation. This abstraction streamlines the development of models that support various effectful computations.

To demonstrate the flexibility of the framework, we illustrate how different effectful notions of computation—namely, state, non-deterministic, and failure—can be used as evidence in corresponding evidenced frames. To illustrate the utility of the framework, we identify various differences between the resulting models and PCA-based models, despite all being evidenced by computational systems.

Lastly, to highlight the unifying nature of evidenced frames, we prove that they are complete with respect to **Set**-based triposes [6]. That is, any **Set**-based tripos (under common metatheoretic assumptions to be discussed in detail) can be described as and reconstructed from an evidenced frame. However, the reverse is not true, as the evidence within an evidenced frame cannot, in general, be reconstructed from its corresponding tripos. We furthermore prove that the existing completeness constructions for *implicative algebras*—a generalization of Boolean algebra tailored for realizability models [10]–[12]—factor through our simpler constructions for evidenced frames. However, whereas evidenced frames for computational system can use codes of the system directly as evidence, implicative algebras requires one to indirectly represent such evidence through powersets of codes. Thus evidenced frames offer an ideal domain for realizability models, with triposes effectively being evidenced frames that have forgotten their evidence, and with implicative algebras imposing much more structure than necessary.

Our contributions. In this paper we introduce a novel general-purpose framework, which we term *evidenced frame*, for building broader realizability models that can accommodate diverse effectful computations. Concretely, this paper

- introduces evidenced frames and provides a framework for constructing realizability models;
- embeds several common effectful notions of computation into evidenced frames, and in turn, realizability models;
- proves the completeness of evidenced frames with respect to **Set**-based triposes;
- and proves that the constructions connecting implicative algebras and realizability models factors cleanly through evidenced frames.

Outline. Section II provides background on PCAs and triposes and their role as models of higher-order dependent logic. Section III presents the main concept of evidenced frame. Section IV illustrates the flexibility of evidenced frame by showing how it captures various notions of effectful computations, namely, computation with state, non-determinism, and failure. Section V presents the construction of a realizability model from an evidenced frame (Section V-B), discusses some of the effectful models presented (Section V-C), and proves completeness of evidenced frames with respect to **Set**-based triposes (Section V-F). Section VI then shows how the connections between implicative algebras and triposes factor through evidenced frames. Section VII concludes with directions for future work. Due to space constraints, full definitions and proofs are elided throughout the paper but can be found, together with a corresponding Coq mechanization, at https://www.cs.bgu.ac.il/~cliron/EvidencedFrame.

II. Background

The main goal of the paper is to introduce evidenced frames as a general-purpose framework for building realizability models that can soundly accommodate diverse effectful computations. There are realizability models of many kinds of logic, and in this paper we target specifically dependent higher-order (predicate) logic, whose category-theoretic models are *triposes* [6], [13]. Accordingly, this section provides background on triposes and their role realizability models.

A. Triposes

A tripos is a kind of a model of higher-order *dependent* logic over *dependent* type theory [14]. In this paper we focus specifically on **Set**-based triposes, meaning the dependent type theory is comprised of the type constructors that can be modeled by (the slice category of) **Set**, such as unit, dependent functions, and dependent sums. The dependent logics over this type theory that we are considering are comprised of $\top$, $\bot$, conjunction, disjunction, implication, equality, and universal and existential quantification. And *higher-order* dependent logic furthermore includes a type Ω of propositions, whose (dependent) terms can be used as (dependent) predicates in the logic and vice versa. Thus, higher-order logic provides a means of discussing (higher-order) relations while simultaneously enabling impredicative quantification over propositions. Definition V.1 provides a full formal definition of triposes.

Although in this paper we focus in logic, the tripos-to-topos construction [15] provides a standard method for converting models of higher-order logic into models of (extensional, impredicative) dependent type theory (and set theory). In particular, a *realizability topos* is a topos that is constructed from a *realizability tripos*, where a realizability tripos is a tripos that is constructed through a standard process from a partial combinatory algebra.

B. Partial combinatory algebra (PCA)

The formalization of PCAs is twofold. First, the concepts of codes and partial application of codes are introduced through a *partial applicative structure*.

Definition II.1 (Partial Applicative Structure)**.** A partial applicative structure is a set C of “codes” c and a *partial* binary “application” operator $\cdot$ on C. We use $c_f \cdot c_a \downarrow c_r$ to denote c_r being the (successful) result of the application $c_f \cdot c_a$.

Second, a PCA is defined as a partial applicative structure that is “functionally complete”, meaning there is a way to encode application expressions with n free variables as individual codes accepting n arguments through applications.

This ensures the necessary expressiveness for modeling computational systems like the λ-calculus. To present the formal definition, we first formalize expressions e with numbered free variables $i \in \mathbb{N}$, substitution $e[c_a]$, and the extension of reduction to expressions $e \downarrow c_r$.

$$e ::= i \in \mathbb{N} \mid c \in C \mid e \cdot e \qquad E_n = \{e \mid \text{all } i\text{s in } e \text{ are } < n\}$$

e	$e[c_a]$
0	c_a
$i+1$	i
c	c
$e_f \cdot e_a$	$e_f[c_a] \cdot e_a[c_a]$

$$\frac{}{c \downarrow c} \qquad \frac{e_f \downarrow c_f \qquad e_a \downarrow c_a \qquad c_f \cdot c_a \downarrow c_r}{e_f \cdot e_a \downarrow c_r}$$

Definition II.2 (Partial Combinatory Algebra)**.** A PCA is a partial applicative structure with an assignment of every expression $e \in E_{n+1}$ to a code $c_{\lambda^n.e} \in C$ that conceptually embodies the λ-calculus term binding the $n+1$ free variables in e, as formalized by the following requirements:

$$\begin{array}{ll} \forall n. \forall e \in E_{n+2}. \forall c_a. & c_{\lambda^{n+1}.e} \cdot c_a \downarrow c_{\lambda^n.e[c_a]} \\ \forall e \in E_1. \forall c_a, c_r. & c_{\lambda^0.e} \cdot c_a \downarrow c_r \iff e[c_a] \downarrow c_r \end{array}$$

Perhaps the more standard definition of PCAs is as partial applicative structures with **s** and **k** *combinators* satisfying certain behaviors [2]. These combinators are simply encodings of particular expressions that are sufficient to ensure that all expressions can be encoded. In our formalization, the **s** and **k** combinators are simply the codes $c_{\lambda^2.(0\cdot 2)\cdot(1\cdot 2)}$ and $c_{\lambda^1.0}$ modeling the λ-calculus terms $\lambda x.\lambda y.\lambda z.\ (x\ z)\ (y\ z)$ and $\lambda x.\lambda y.x$, respectively.

C. From PCAs to realizability triposes

Given a PCA one can construct its corresponding realizability tripos, in which the set of propositions Ω is defined as $\mathcal{P}(C)$, via a standard construction [2]. The core intuition is that a predicate on a set I specifies for each element i of I which codes (if any) from the partial combinatory algebra "realize" that the predicate holds for i. A common example defines the predicate $\mathbb{n}$ on the natural numbers $\mathbb{N}$ such that $\mathbb{n}(n)$ is realized solely by the Church encoding of n, thereby connecting a value from the meta-theory to a (in this case singular) computational representation of that value in the PCA. A code c^λ_n that Church-encodes n can be defined as

$$c^\lambda_0 = c_{\lambda^1.1} \qquad c^\lambda_{n+1} = c_{\lambda^1.0\cdot((c^\lambda_n \cdot 0)\cdot 1)}$$

One predicate ϕ_1 entails another ϕ_2 in a realizability tripos when there is a *uniform* code that converts all the realizers of $\phi_1(i)$ to realizers of $\phi_2(i)$ for every i in I. Uniformity means that the code does not itself depend on i; that is, the same code must work for all elements of I. Uniformity is critical for ensuring entailment corresponds to computation. To see why, suppose for a function $f : \mathbb{N} \to \mathbb{N}$ (in **Set**) we define a predicate on the natural numbers, call it ϕ_f, whose sole realizer for a given n is $c^\lambda_{f(n)}$. Consider what it means for $\mathbb{n}$ to entail ϕ_f. If entailment could be evidenced by a different code c_n for each $n \in \mathbb{N}$, then $\mathbb{n}$ entails ϕ_f for any function f since c_n could be the constant computation that returns $c^\lambda_{f(n)}$. However, requiring a uniform code that works for all indices $n \in \mathbb{N}$ ensures the predicate $\mathbb{n}$ entails ϕ_f if and only if f is computable according to the PCA at hand. Thus uniformity ensures that entailment actually has computational significance. This is why realizability toposes (i.e. toposes derived from triposes derived from PCAs) have the property that, internally, all functions $\mathbb{N} \to \mathbb{N}$ are computable (according to the corresponding PCA).

With these intuitions in mind, we can informally describe how the various propositional connectives are modeled by realizability triposes. The realizers of a conjunction $\phi_1 \wedge \phi_2$ are simply the Church-encoded pairs of realizers of ϕ_1 and ϕ_2. The realizers of an implication $\phi_1 \supset \phi_2$ are simply the codes that, when applied to a realizer of ϕ_1, necessarily produce a realizer of ϕ_2. There are no realizers for $\perp$, and the realizers of a disjunction $\phi_1 \vee \phi_2$ are the Church-encoded tagged unions of realizers of ϕ_1 and realizers of ϕ_2. A realizer of a universal quantification $\forall i{:}I.\phi_i$ (for inhabited I) is anything that is a realizer of ϕ_i for *every* $i \in I$, whereas a realizer of an existential quantification $\exists i{:}I.\phi_i$ is anything that is a realizer of ϕ_i for *some* $i \in I$. Lastly, any code is a realizer of $\top$, and any code is a realizer of $i =_I i'$ if and only if i and i' are equal in I (in **Set**). Notice that the realizers for the quantifiers are themselves uniform. That is, a realizer of $\exists i{:}I.\phi_i$ has no *computational* way of knowing *which* i it is a realizer for, and similarly a realizer of $\forall i{:}I.\phi_i$ *cannot* computationally depend on the index i.

III. Evidenced Frames

Traditional realizability models are generated from PCAs. This definition of realizability bakes in an expectation that PCAs encompass computation. However, because a PCA defines application to be functional (albeit partial), even something as simple as non-determinism cannot be directly expressed in a PCA; one has to use indirect representations such as powersets [16]. This bounds realizability models to pure (non-effectful) computations (apart from non-termination).[2]

Previous work [8] explored the impact of lifting these restrictions by generalizing PCAs into *relational* combinatory algebras (RCAs) and *stateful* combinatory algebras (SCAs) directly modeling non-deterministic and stateful computation, respectively. These more expressive models of computation were then shown to also give rise to models of higher-order dependent logic. Generalizing that work, this section introduces the uniform, abstract framework of evidenced frames, wherein "evidence" generalizes the role of computational codes but without any equational theory or notion of reduction. This abstraction allows for the introduction of various forms of effectful computation, from which corresponding models of higher-order dependent logic, i.e. triposes, can be easily constructed. The key insight is the importance that *uniformity* has in traditional realizability triposes for ensuring their correspondence between entailment and computability.

[2]In fact, it also enforces intuitionistic realizability, though combinatory algebras can also be used to construct, e.g., Krivine classical realizability [12].

Intuitively speaking, an evidenced frame can be regarded as a frame [17], or (more accurately) a complete Heyting algebra, in which one provides evidence that an element is smaller than another, and there can be multiple forms of evidence for the same fact. In Kleene's realizability model [1], the idea is that the elements of the frame represent the propositions in the model, and evidence that ϕ_1 is smaller than ϕ_2 (i.e., ϕ_1 entails ϕ_2) is a computation that can convert proofs of ϕ_1 into proofs of ϕ_2. This computation is not necessarily typed though, in that the same computation can be used as evidence of a variety of entailment relationships. For example, the evidence $e_{\mathtt{id}}$ in the following definition serves as evidence that any ϕ entails itself.

Definition III.1 (Evidenced Frame)**.** An *evidenced frame* is a triple $(\Phi, E, \cdot \xrightarrow{\cdot} \cdot)$, where Φ is a set of propositions, E is a collection of evidence, and $\phi_1 \xrightarrow{e} \phi_2$ is a ternary evidence relation on $\Phi \times E \times \Phi$, along with the following:

Reflexivity There exists evidence $e_{\mathtt{id}} \in E$:

- $\forall \phi.\ \phi \xrightarrow{e_{\mathtt{id}}} \phi$

Transitivity There exists an operator $; \in E \times E \to E$:

- $\forall \phi_1, \phi_2, \phi_3, e, e'.\ \phi_1 \xrightarrow{e} \phi_2 \wedge \phi_2 \xrightarrow{e'} \phi_3 \implies \phi_1 \xrightarrow{e\,;\,e'} \phi_3$

Top A proposition $\top \in \Phi$ such that there exists evidence $e_\top \in E$:

- $\forall \phi.\ \phi \xrightarrow{e_\top} \top$

Conjunction An operator $\wedge \in \Phi \times \Phi \to \Phi$ such that there exists an operator $\langle\!| \cdot, \cdot |\!\rangle \in E \times E \to E$ and evidence $e_{\mathtt{fst}}, e_{\mathtt{snd}} \in E$:

- $\forall \phi_1, \phi_2.\ \phi_1 \wedge \phi_2 \xrightarrow{e_{\mathtt{fst}}} \phi_1$
- $\forall \phi_1, \phi_2.\ \phi_1 \wedge \phi_2 \xrightarrow{e_{\mathtt{snd}}} \phi_2$
- $\forall \phi, \phi_1, \phi_2, e_1, e_2.\ \phi \xrightarrow{e_1} \phi_1 \wedge \phi \xrightarrow{e_2} \phi_2 \implies \phi \xrightarrow{\langle\!| e_1, e_2 |\!\rangle} \phi_1 \wedge \phi_2$

Universal Implication An operator $\supset \in \Phi \times \mathcal{P}(\Phi) \to \Phi$ such that there exists an operator $\lambda \in E \to E$ and evidence $e_{\mathtt{eval}} \in E$:

- $\forall \phi_1, \phi_2, \vec{\phi}, e.\ (\forall \phi \in \vec{\phi}.\ \phi_1 \wedge \phi_2 \xrightarrow{e} \phi) \implies \phi_1 \xrightarrow{\lambda e} \phi_2 \supset \vec{\phi}$
- $\forall \phi_1, \vec{\phi}, \phi \in \vec{\phi}.\ (\phi_1 \supset \vec{\phi}) \wedge \phi_1 \xrightarrow{e_{\mathtt{eval}}} \phi$

In the above definition, we write $\vec{\phi}$ for an element of $\mathcal{P}(\Phi)$, i.e., a subset of Φ.

Remark III.2. To simplify the construction of evidenced frames, in the above definition we use universal implication as a connective that unifies the standard implication and universal quantification. In fact, it is equivalent to the combination of implication and universal quantification. For example, we can define the standard connectives as:

$$\begin{array}{lcl} \phi_1 \to \phi_2 & \triangleq & \phi_1 \supset \{\phi_2\} \\ \prod \vec{\phi} & \triangleq & \top \supset \vec{\phi} \\ \bot & \triangleq & \prod \Phi \\ \phi_1 \vee \phi_2 & \triangleq & \prod\{(\phi_1 \supset \phi') \wedge (\phi_2 \supset \phi') \supset \phi' \mid \phi' \in \Phi\} \\ \coprod \vec{\phi} & \triangleq & \prod\{(\prod\{\phi \supset \phi' \mid \phi \in \vec{\phi}\}) \supset \phi' \mid \phi' \in \Phi\} \end{array}$$

Definition III.1 directly links logical operations with program operations. Reflexivity is program identity, and transitivity is program composition. Conjunction is program pairing with pair projections. As mentioned, the universal implication unifies the standard implication and universal quantification. Implication is program currying with closure evaluation. Quantification, though, has no programmatic counterpart and instead corresponds to uniformity—the quantification states that the same code serves as evidence for each proposition in the subset *without* the code examining the proposition itself. Note that evidenced frames enable us to model impredicativity since the subset can be taken to be the set of all propositions Φ of the frame, as used in the definitions of $\bot$, $\vee$, and $\coprod$ above.

Definition III.3. Given an evidenced frame $(\Phi, E, \cdot \xrightarrow{\cdot} \cdot)$, we say that $e \in E$ is *evidence* of $\phi \in \Phi$ if $\top \xrightarrow{e} \phi$ holds. The evidenced frame is said to *model* ϕ if it has evidence of ϕ. An evidenced frame is *consistent* if it does not model $\bot$.

Example III.4 (PCA as an evidenced frame)**.** As a first example, notice that a PCA can be naturally embedded into an evidenced frame. Given a PCA $(C, \cdot)$, its corresponding evidenced frame is defined by the triple $(\mathcal{P}(C), C, \cdot \xrightarrow{\cdot} \cdot)$, where a proposition in $\Phi = \mathcal{P}(C)$ is defined by its set of realizers in C, an evidence in $E = C$ is a code of the PCA, and the evidence relation $\phi_1 \xrightarrow{e} \phi_2$ is defined so that for any code $e_1 \in \phi_1$, $e \cdot e_1$ terminates and if $e \cdot e_1$ reduces to e_2, then $e_2 \in \phi_2$. The connectives and corresponding evidences are defined as usual in realizability models.

Evidenced frames require relationships between the operations on propositions and the operations on evidence. However, there are no requirements relating the operations on evidence to each other. Thus, this minimal structure ensures that uniform realizers can be extracted from proofs *without* imposing any constraints on how those realizers compute. There is no equational theory whatsoever in an evidenced frame, and as such evidenced frames do not prescribe any particular theory of computation; they simply demand that there is enough expressive power available. Accordingly, even though evidenced frames make no reference to state, non-determinism, or failure, all these forms of computation can be captured in evidenced frames, as the next section demonstrates.

IV. Effectful Computation via Evidenced Frames

This section demonstrates the generality and uniformity of the framework of evidenced frames. It does so by illustrating how various forms of effectful computation, namely shared mutable state, non-deterministic computation, and computation with a failure mechanism, can all be captured naturally as instances of evidenced frame. This, in turn, enables, by Theorem V.5, for a general, uniform construction of triposes that internally supports these effects.[3] Our examples further demonstrate the utility of the framework by showing the ease in which such computational elements can be augmented.[4]

[3]While there are various ways to *model* non-deterministic computation (e.g. [18]) and stateful computation (using, e.g., the state monad [19]), most of them do so *indirectly* and not embed them into the underlying computational system. Notable exceptions include [20] for non-deterministic computation, [21] for stateful computation or [22] for failure and exceptions.

[4]The evidenced-frame structure presented in this section is a generalization of the notion of stateful combinatory algebras (SCAs) developed in [8].

A. Framework for common effectful computational systems

This section describes abstract computational systems $\mathscr{C}$ with shared mutable state, non-deterministic computation, and potential for computational failure. Expressions e and substitution $e[c]$ are defined as in a PCA (see Section II-B). The other notions from PCAs are adapted to take the additional computational effects into account as follows.

Let Σ be an inhabited set of states σ with a "possible future" preorder $\sigma \leq \sigma'$, and let $e \downarrow^{\sigma}_{\sigma'} c$ and $e{\downarrow^{\sigma}}$ define a reduction relation and a termination relation (respectively) satisfying the following rules and properties:

$$\frac{}{c \downarrow^{\sigma}_{\sigma} c} \qquad \frac{e_f \downarrow^{\sigma}_{\sigma'} c_f \qquad e_a \downarrow^{\sigma'}_{\sigma''} c_a \qquad c_f \cdot c_a \downarrow^{\sigma''}_{\sigma'''} c_r}{e_f \cdot e_a \downarrow^{\sigma}_{\sigma'''} c_r} \qquad \frac{}{c{\downarrow^{\sigma}}}$$

$$\frac{e_f{\downarrow^{\sigma}} \qquad \begin{array}{l}\forall \sigma', c_f.\ e_f \downarrow^{\sigma}_{\sigma'} c_f \\ \quad \implies e_a{\downarrow^{\sigma'}} \wedge \forall \sigma'', c_a.\ e_a \downarrow^{\sigma'}_{\sigma''} c_a \implies c_f \cdot c_a{\downarrow^{\sigma''}}\end{array}}{e_f \cdot e_a{\downarrow^{\sigma}}}$$

Preservation: $\forall \sigma, c_f, c_a, \sigma', c_r.\ c_f \cdot c_a \downarrow^{\sigma}_{\sigma'} c_r \implies \sigma \leq \sigma'$

The concept of "possible futures" captures the fact that, even in a system with mutable state, the system can maintain certain invariants about its state and how it progresses, as enforced by the preservation property. Note, though, that the reduction and termination relations are not themselves necessarily preserved by futures: an application is permitted to reduce to a code in a given state that it cannot reduce to in a future state (as opposed to the case of a standard possible-worlds structure, e.g. [23]).

We further assume "functional completeness" of the calculus.[5] That is, we assume the existence of an assignment of every expression $e \in E_{n+1}$ to a code $c_{\lambda^n.e} \in C$ satisfying the following properties in all states $\sigma, \sigma' \in \Sigma$:

$$\begin{array}{llcl}
\forall n.\forall e \in E_{n+2}.\forall c_a, c_r. & c_{\lambda^{n+1}.e} \cdot c_a \downarrow^{\sigma}_{\sigma'} c_r & \implies & \begin{array}{c}\sigma' = \sigma \\ \wedge \\ c_r = c_{\lambda^n.e[c_a]}\end{array} \\
\forall e \in E_1.\forall c_a, c_r. & c_{\lambda^0.e} \cdot c_a \downarrow^{\sigma}_{\sigma'} c_r & \implies & e[c_a] \downarrow^{\sigma}_{\sigma'} c_r \\
\forall n.\forall e \in E_{n+2}.\forall c_a. & & & c_{\lambda^{n+1}.e} \cdot c_a{\downarrow^{\sigma}} \\
\forall e \in E_1.\forall c_a. & e[c_a]{\downarrow^{\sigma}} & \implies & c_{\lambda^0.e} \cdot c_a{\downarrow^{\sigma}}
\end{array}$$

Definition IV.1. A *computational system* (i.e., a concrete implementation), denoted by $\mathscr{C}$, is a system with the above components which satisfies all the above rules and properties.[6]

For example, the standard notion of a PCA can be obtained by taking Σ to be singleton, requiring reduction to be deterministic and to imply termination, and requiring an additional *progress* property to be discussed shortly.

B. Common effectful combinators

Before turning to the general construction of an effectful evidenced frame, this section provides concrete examples of computation systems for the common effects considered here.

[5]This notion is taken from PCAs, and here it is adapted to incorporate stateful and nondeterministic application.

[6]Note that the rules and properties above only describe a general behavior of the reduction and termination relations that is *assumed to hold*. Thus, they do not provide inductive definitions of termination and reduction, and the preservation property is not proven.

1) Non-determinism: As an example introducing a simple form of non-determinism into a computational system, we take $\mathscr{C}_{\mathsf{flip}}$ to be the computational system generated from a combinator flip that models a coin flip, i.e., non-deterministically evaluating to the Church encoding of either false or true. Formally, the flip token behaves in the following way:

$$\frac{}{\mathsf{flip} \cdot c{\downarrow^{\sigma}}} \qquad \frac{}{\mathsf{flip} \cdot c \downarrow^{\sigma}_{\sigma} c_{\lambda^1.0}} \qquad \frac{}{\mathsf{flip} \cdot c \downarrow^{\sigma}_{\sigma} c_{\lambda^1.1}}$$

$\mathscr{C}_{\mathsf{flip}}$ is a simplified version of the Flip-RCA example given in [8]—we refer the interested reader there for more details.

2) Mutable state: As an example of introducing a simplified notion of state into a computational system we take $\mathscr{C}_{\mathsf{lookup}}$ to be the computational system generated from a countable number of "lookup" terms to some location heaps. The set of states Σ is defined to be finite partial maps from the natural numbers to the codes of the computational system, ordered by inclusion. The codes are generated from the combinators lookup_n that model retrieving (and possibly initializing) the code stored at the address $n \in \mathbb{N}$ in the heap. The lookup combinator is formalized as follows, where we use $n \mapsto c \in \sigma$ to denote that the code c is stored at the address n in the heap σ, and $\sigma, n \mapsto c$ for the updated heap.

$$\frac{}{\mathsf{lookup}_n \cdot c{\downarrow^{\sigma}}} \qquad \frac{n \mapsto c' \in \sigma}{\mathsf{lookup}_n \cdot c \downarrow^{\sigma}_{\sigma} c'} \qquad \frac{\nexists c'.\ n \mapsto c' \in \sigma}{\mathsf{lookup}_n \cdot c \downarrow^{\sigma}_{\sigma, n \mapsto c} c}$$

$\mathscr{C}_{\mathsf{lookup}}$ is a simplified version of the Mem-SCA example given in [8] and so we here elide details that are irrelevant to our current discussion (e.g. allocation method, heap ordering)—we refer the interested reader there for full details.

3) Failure: As an example of introducing a failure mechanism into the computation system, we take $\mathscr{C}_{\mathsf{fail}}$ to be the computational system generated from a combinator fail that terminates on all inputs but does not reduce to anything. Formally, the semantics of fail is the following:

$$\frac{}{\mathsf{fail} \cdot c{\downarrow^{\sigma}}}$$

C. Separating computational failure

The fail combinator presents a challenge for logical realizability, since naively employed it can realize entailment between any two propositions. To develop a logically consistent model, we need to separate fail from other terms so that its effects can be employed in a contained manner. To this end, we introduce the notion of a *separator*.[7]

Definition IV.2. Given a computational system $\mathscr{C}$, a *separator* $\mathcal{S}$ is a functionally complete subset of C closed under reduction for which the following *progress* property holds.

$$\forall \sigma, c_f \in \mathcal{S}, c_a \in \mathcal{S}.\ c_f \cdot c_a{\downarrow^{\sigma}} \implies \exists \sigma', c_r.\ \ c_f \cdot c_a \downarrow^{\sigma}_{\sigma'} c_r$$

A PCA, in particular, has the (sub)set C as a valid separator.

In our evidenced frame, we will restrict *evidence* to be codes in the separator. Thus, by keeping codes such as fail out of

[7]We borrow the terminology from implicative algebras (see Definition VI.2) insofar as in both settings the separator is used to discriminate valid realizers.

the separator, we can build a consistent evidenced frame. But we will still define propositions using *arbitrary* codes, so that a proposition can still be realized by fail.

For the models we will consider, the two common classes of separators are the separator comprised of all codes (when progress holds for all codes), which we will denote with $\mathcal{S}_\top$, and the separator comprised of codes generated solely from functional completeness, which we will denote with $\mathcal{S}_\lambda$.

D. Evidenced frames for common effects

This section demonstrates how one can construct evidenced frames from a computational system $\mathscr{C}$ with a separator $\mathcal{S}$. The propositions ϕ will be subsets of state-code pairs, denoted $\phi^\sigma(c)$, indicating that the proposition is realized by certain codes in certain states. Evidence will be elements of the separator—rather than arbitrary codes—and conceptually the evidence relation holds when the separator maps realizers to realizers. More formally, it turns out that there are at least two useful interpretations of what it means to map realizers to realizers depending on how we interpret non-determinism. We can interpret non-determinism *demonically* (D) or *angelically* (A) [24]. With the demonic interpretation, a mapping of realizers is valid only when *all* possible results of the reduction are realizers. With the angelic interpretation, a mapping of realizers is valid when *a* possible result of the reduction is a realizer. We formalize these two interpretions as follows:

$$c_f \cdot c_a \downdownarrows^\sigma_D \phi \triangleq c_f \cdot c_a {\downarrow^\sigma} \wedge \forall \sigma', c_r.\ c_f \cdot c_a \downarrow^\sigma_{\sigma'} c_r \Longrightarrow \phi^{\sigma'}(c_r)$$
$$c_f \cdot c_a \downdownarrows^\sigma_A \phi \triangleq c_f \cdot c_a {\downarrow^\sigma} \wedge \exists \sigma', c_r.\ c_f \cdot c_a \downarrow^\sigma_{\sigma'} c_r \ \wedge\ \phi^{\sigma'}(c_r)$$

Intuitively, $c_f \cdot c_a \downdownarrows^\sigma_{\mathfrak{D}} \phi$ has two components (where $\mathfrak{D} ::= D \mid A$). First, it ensures termination, which in turn ensures the existence of *some* result if c_f and c_a are in the separator. Second, it ensures that reduction is sound, either under all possible future (D) or for some possible future (A). Accordingly, the definition of an evidenced frame below can be adjusted to either of these interpretations of non-determinism $\mathfrak{D}$.

Definition IV.3. Given a computational system $\mathscr{C}$, a separator $\mathcal{S}$, and an interpretation of non-determinism $\mathfrak{D}$, the evidenced frame $\mathcal{EF}^{\mathscr{C},\mathcal{S}}_{\mathfrak{D}}$ is defined as $\langle \Phi, E, \cdot \dot{\rightarrow} \cdot \rangle$ where:

- A proposition in $\Phi \subset \mathcal{P}(\Sigma \times C)$ is a "stateful" predicate on codes $\phi^\sigma(c)$ that is "future-stable":
$$\forall \sigma, \sigma', c.\ \ \sigma \leq \sigma' \wedge \phi^\sigma(c) \implies \phi^{\sigma'}(c)$$
- E is the set of codes in the separator $\mathcal{S}$.
- The evidence relation $\phi_1 \xrightarrow{e} \phi_2$ is defined as
$$\forall \sigma, c.\ \phi_1^\sigma(c) \Longrightarrow e \cdot c \downdownarrows^\sigma_{\mathfrak{D}} \phi_2$$

Top The predicate $\top^\sigma(c)$ holds for any c and σ.
Conjunction The predicate $(\phi_1 \wedge \phi_2)^\sigma(c)$ is defined as

$$\forall \sigma'.\ \sigma \leq \sigma' \Longrightarrow c \cdot c_{\lambda^1.0} \downdownarrows^{\sigma'}_{\mathfrak{D}} \phi_1 \wedge c \cdot c_{\lambda^1.1} \downdownarrows^{\sigma'}_{\mathfrak{D}} \phi_2$$

Universal Implication The predicate $(\phi_1 \supset \vec{\phi})^\sigma(c)$ is

$$\forall \sigma', c_1, \phi \in \vec{\phi}.\ \ \sigma \leq \sigma' \wedge \phi_1^{\sigma'}(c_1) \implies c \cdot c_1 \downdownarrows^{\sigma'}_{\mathfrak{D}} \phi$$

Any combination of the examples in Section IV-B induces evidenced frames via the above definition. When the separator is $\mathcal{S}_\top$, we denote the evidenced frame simply as $\mathcal{EF}^{\mathscr{C}}_{\mathfrak{D}}$. When the computational system is a PCA, the demonic and angelic evidenced frames coincide for the $\mathcal{S}_\top$ separator, in which case we simply denote either evidenced frame as $\mathcal{EF}^{\mathscr{C}}$. Section VI will offer yet another set of examples of computations that exceed PCAs and can be embedded into evidenced frame. One particularly interesting example is computations with continuations (i.e. `call/cc`), which enables an evidenced frame for a classical calculus, replicating preexisting computational models of classical logic [25]–[27].

V. Evidenced Frames and Realizability Models

As our terminology suggests, we designed evidenced frames to have a logical interpretation. To that end, we use (**Set**-based) triposes, which correspond to models of dependent higher-order logic over the **Set** model of dependent type theory. We provide a way to construct a (consistent) tripos from a (consistent) evidenced frame, which is conservative over traditional realizability models in that the evidenced frame for a PCA results in the standard realizability tripos for that PCA. We illustrate, however, that the triposes resulting from our other non-standard computational systems exhibit interesting properties that no traditional realizability tripos exhibits despite entailment still being realized by computation. We then demonstrate that every tripos can in turn be represented as an evidenced frame such that converting the resulting evidenced frame back to a tripos results in the same tripos. However, in converting an evidenced frame to a tripos and back, the evidence is lost; evidence that can be countable and concrete such as codes in a computational system gets replaced by its extensional characterization as uncountable relationships between propositions. Thus evidenced frames more directly and flexibly capture the structure underlying realizability without limiting the models it can describe.

A. Triposes

The precise meaning of *tripos* varies across settings. In broad lines, here we define a tripos $\mathcal{T}$ in terms of a function assigning to each set[8] Γ a Heyting prealgebra[9] $\mathcal{T}(\Gamma)$, which conceptually represent predicates on Γ, and to each function $s : \Gamma \to \Gamma'$ a morphism of Heyting prealgebras $s^* : \mathcal{T}(\Gamma') \to \mathcal{T}(\Gamma)$, which is intuitively the substitution induced by s over predicates.

[8] Our Coq proof models sets as the types of a `Type` universe in which we assume Uniqueness of Identity Proofs. The category **Set** is comprised of such sets and (Coq) functions between them (which have η and β equivalence).

[9] We use prealgebras to avoid requiring the Axiom of Choice of our metatheory while keeping our constructions straightforward. The carrier of these prealgebras belongs to a `Type` universe that may or may not be **Set**. The category **pHA** is comprised of Heyting algebras and *weak* morphisms between them, meaning the algebraic operations are preserved but only up to equivalence. We also equip **pHA** with a preorder between parallel morphisms defined pointwise, making **pHA** a locally preordered bicategory.

Definition V.1 (A **Set**-based Tripos)**.** A **Set**-*based tripos* is a pseudofunctor[10] $\mathcal{T} : \mathbf{Set}^{op} \to \mathbf{pHA}$ with the following:

Quantifiers. For each function $s : \Gamma \to \Gamma'$, s^* has both a left and a right adjoint $\coprod_s / \prod_s$ in $\mathcal{T}(\Gamma) \to \mathcal{T}(\Gamma')$, that is:

$$\begin{array}{ccc} \varphi \le s^*(\psi) & \Leftrightarrow & \coprod_s(\varphi) \le \psi \\ s^*(\varphi) \le \psi & \Leftrightarrow & \varphi \le \prod_s(\psi) \end{array}$$

Compatibility. Given any pullback square:

$$\begin{tikzcd} \Gamma \arrow[r, "r"] \arrow[d, "v"'] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma' \arrow[d, "u"] \\ \Gamma'' \arrow[r, "s"'] & \Gamma''' \end{tikzcd}$$

we have $s^* \circ \coprod_u \le \coprod_v \circ r^*$ and $\prod_v \circ r^* \le s^* \circ \prod_u$.

Generic predicate. A set Ω, a predicate $\mathsf{holds} \in \mathcal{T}(\Omega)$, and, for any set Γ and any predicate $\phi \in \mathcal{T}(\Gamma)$, a function $\chi_\phi : \Gamma \to \Omega$ such that $\chi_\phi^*(\mathsf{holds}) \lesseqgtr \phi$.

Reflective surjective substitions. Given any surjective function $s : \Gamma \to \Gamma'$, one has $s^*\phi \le s^*\phi' \implies \phi \le \phi'$.

Let us pause here to give a few intuitions to this definition, and let us informally write $\varphi(\vec{x})$ for a predicate in $\mathcal{T}(\Gamma)$ and $\varphi(s(\vec{y}))$ for the image of φ by the substitution s^*. Since both quantifiers $\exists x : X$ and $\forall x : X$ turn any formula ranging over $\Gamma \times X$ into a formula ranging over Γ, it is natural to interpret them as morphism from $\mathcal{T}(\Gamma \times X)$ to $\mathcal{T}(\Gamma)$. We can thus define them as left and right adjoints of the first projection $\pi_{\Gamma,X} : \Gamma \times X \to X$; the adjunctions' properties then correspond to the logical equivalences that characterize them. For instance, we have for the coproduct/existential quantifier: $\big(\forall \vec{y}, x. \varphi(\vec{y}, x) \Rightarrow \psi(\vec{y})\big) \Leftrightarrow \big(\forall \vec{y}. \exists x. \varphi(\vec{y}, x)) \Rightarrow \psi(\vec{y})\big)$.

Compatibility, often referred to as the Beck-Chevaley condition, expresses that quantifiers are compatible with substitution. For instance, if we consider the pullback:

$$\begin{tikzcd} \Gamma' \times X \arrow[r, "s \times \mathrm{id}_X"] \arrow[d, "\pi_{\Gamma',X}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma \times X \arrow[d, "\pi_{\Gamma,X}"] \\ \Gamma' \arrow[r, "s"'] & \Gamma \end{tikzcd}$$

the condition for the coproduct/existential quantifier requires that for any $\varphi \in \mathcal{T}(\Gamma' \times X)$ and any $\vec{y}' \in \Gamma'$, the equality $(\exists (x : X). \varphi(\vec{y}, x))[\vec{y} := s(\vec{y}')] = \exists (x : X). \varphi(s(\vec{y}'), x)$ holds.

The set Ω is an internal representation of the tripos's propositions, holds indicates when a proposition holds internally, and the function χ_φ conceptually represents the predicate φ as a term (i.e. $\forall \vec{x} \in \Gamma.\ \mathsf{holds}(\chi_\varphi(\vec{x})) \lesseqgtr \varphi(\vec{x})$). It is important to notice that, while most of the structure of a tripos is uniquely determined up to equivalence, the generic predicate is not. Different choices of generic predicate can change which higher-order-logic sentences hold in the model. Yet, despite these differences, the resulting *toposes* are necessarily (logically) isomorphic regardless of the choice of generic predicate.

The final property captures that entailment is propositional and so, if for every entailment in Γ' there *exists* a corresponding proof of entailment in Γ, we can transport that proof to Γ. Indeed, substitution along any section that splits the surjection would transport the proof. As such, works assuming a stronger metatheory wherein the Axiom of Choice holds (externally) for **Set** gain this property for all **Set**-based triposes implicitly. For example, the sole use of the Axiom of Choice in Miquel's completeness theorem for implicative algebras and triposes is to reflect a proof of entailment along a surjective substitution [11, Proposition 2.6]. As we will illustrate, the property makes for a cleaner relationship between the tripos's internal logic and the logic of its base category **Set**.

[10]A pseudofunctor is a relaxation of a functor wherein the mapping need only preserve identity and composition of morphisms only up to equivalence.

Example V.2. The canonical **Set**-based tripos has, as its predicates on a set Γ, the metatheoeritic predicates on Γ. The quantifiers $(\coprod_s \phi)(\vec{y})$ and $(\prod_s \phi)(\vec{y})$ correspond to $\exists \vec{x}.\ s(\vec{x}) = \vec{y} \wedge \phi(\vec{y})$ and $\forall \vec{x}.\ s(\vec{x}) = \vec{y} \supset \phi(\vec{y})$, respectively. The set Ω is the set of metatheoretic propositions, e.g. `Prop` in Coq. We abuse notation and denote this tripos also as **Set**.

Example V.3. A simple example of a tripos is given, starting from a complete Heyting (resp. Boolean) prealgebra $\mathcal{H}$, by the following functor: $\mathcal{T}(I) = \mathcal{H}^I$, $s^* = \lambda h.\, h \circ s$. Connectives are defined using the corresponding operations in $\mathcal{H}$ pointwise, while quantifiers are given by abritrary meets and joins [13]. The set Ω is $\mathcal{H}$, and holds is the identity function on $\mathcal{H}$. In the sequel, we refer to these as *forcing* triposes.

With a tripos, one can interpret dependent higher-order logic. A judgement is of the form $\Gamma \mid \phi_1, \ldots, \phi_n \vdash \phi$. A tripos models a judgement, denoted $\mathcal{T} \models \Gamma \mid \phi_1, \ldots, \phi_n \vdash \phi$, if in the Heyting prealgebra $\mathcal{T}(\Gamma)$ the conjunction of $\phi_1, \ldots, \phi_n$ is less than or equal to ϕ. The generic predicate of the tripos makes it possible to treat propositions ϕ in context Γ as expressions of type Ω in Γ and vice-versa.

B. Modeling higher-order logic with evidenced frames

Given an evidenced frame $\mathcal{EF}$, one can construct a tripos of *uniform families* wherein predicates are functions $\phi : \Gamma \to \Phi$, and entailment holds if there is a uniform evidence e that for every element of Γ serves as evidence between the corresponding propositions.

Definition V.4 (UFam construction)**.** For an evidenced frame $\mathcal{EF} = \langle \Phi, E, \cdot \xrightarrow{\cdot} \cdot \rangle$, the structure $\mathsf{UFam}(\mathcal{EF})$ is defined by:

Predicates. Any object $\Gamma \in \mathbf{Set}$ is mapped to $\Phi^\Gamma \in \mathbf{pHA}$ where the preorder $\phi \le \phi'$ is defined as $\exists e. \forall \gamma.\ \phi(\gamma) \xrightarrow{e} \phi'(\gamma)$, and the Heyting prealgebraic structure is defined pointwise via the corresponding operations on Φ provided by $\mathcal{EF}$.

Substitution. Any function $s : \Gamma \to \Gamma'$ is mapped to the **pHA**-morphism $\mathcal{T}(s) = \lambda h.\, h \circ s$.

Quantifiers. The quantifier $\coprod_s \in \Phi^I \to \Phi^J$ is defined as $\lambda \phi. \lambda j.\ \coprod \{\phi(i) \mid i \in I \wedge s(i) = j\}$, and the quantifier $\prod_s$ is defined as $\lambda \phi. \lambda j.\ \prod \{\phi(i) \mid i \in I \wedge s(i) = j\}$.

Generic predicate. The set of propositions Ω is simply Φ, $\mathsf{holds} \in \Omega \to \Omega$ is given by id_Ω, and χ_ϕ is ϕ.

Theorem V.5. $\mathsf{UFam}(\mathcal{EF})$ *is a tripos.*

The intent is that entailment holds only if it has (computational) evidence. Indeed, given a PCA defining a computational system $\mathscr{C}$, the tripos $\mathsf{UFam}(\mathcal{EF}^{\mathscr{C}})$ is precisely the corresponding traditional realizability tripos. But the UFam construction allows one to form a tripos from *any* evidenced frame, thus extending realizability models to broader notions of evidence and in particular broader notions of computation.

C. *Broadening realizability models*

While we ensure that the additions of state, non-determinism, and failure (see Section IV) do not make the resulting models inconsistent, they do have other impacts that make these models deviate from traditional, i.e. PCA-based, realizability triposes. This section presents some examples.

1) Models with demonic non-determinism: Traditional realizability triposes are well known for modeling the Axiom of Countable Choice [2].[11] However, in [8] it was demonstrated that $\mathsf{UFam}(\mathcal{EF}_D^{\mathscr{C}_{\mathsf{flip}}})$ models the *negation* of Countable Choice. In short, the proof that traditional realizability triposes model Countable Choice takes the realizer that the given relation is total and restricts that relation to the image of the realizer for each given natural number. When the realizer is deterministic—as in any PCA—that restriction is deterministic as well, but in $\mathsf{UFam}(\mathcal{EF}_D^{\mathscr{C}_{\mathsf{flip}}})$ such a realizer might be *demonically* non-deterministic, and from that insight it was shown that $\mathsf{UFam}(\mathcal{EF}_D^{\mathscr{C}_{\mathsf{flip}}})$ negates Countable Choice.

2) Models with state: It was further shown in [8] that adding state could restore Countable Choice even in the presence of demonic non-determinism. This was achieved via memoization, where the high-level intuition is to use the state to dynamically determinize the realizer of totality—only supplying a natural-number input to the realizer the first time its corresponding output is requested, and storing that output in the heap to reuse for subsequent requests of the same input.

3) Models with angelic non-determinism: The angelic coin flip in $\mathcal{EF}_A^{\mathscr{C}_{\mathsf{flip}}}$ provides an evidence constructor $e_{\mathsf{flip}}(\cdot,\cdot)$ with some special properties. Conceptually, $e_{\mathsf{flip}}(e_1,e_2)$ conducts a coin flip and reduces to e_1 or e_2 depending on the result, and since the coin flip is angelic it can explore both options concurrently. In particular, $e_{\mathsf{flip}}(e_{\mathtt{fst}},e_{\mathtt{snd}})$ is evidence of both that $\phi_1 \wedge \phi_2$ entails ϕ_1 and that $\phi_1 \wedge \phi_2$ entails ϕ_2. As such, it holds in $\mathcal{EF}_A^{\mathscr{C}_{\mathsf{flip}}}$ that $\exists e. \forall \phi_1, \phi_2.\ \phi_1 \wedge \phi_2 \xrightarrow{e} \prod_{i\in\{1,2\}} \phi_i$. When this property holds we say that the evidenced frame is *finitely forced*. This property is meaningful in that the treatment of the conjunction is the main difference between realizability and forcing models (Example V.3): the former give a computational content to conjunction, usually by means of pairs and projections, while the latter interpret it as an intersection (as for the universal quantification). When the set of evidence is finitely generated, being finitely forced amounts to being equivalent to a forcing tripos where the computational meaning is lost.[12] On the other hand, traditional realizability triposes model its *negation* because $\wedge$ corresponds to pairs and PCA-based evidence has to *deterministically* pick which component of the pair to project. With angelic non-determinism, one can flip a coin and project either component depending on how the coin lands.

[11] Assuming Countable Choice in the metatheory.

[12] Following insights from [10], in $\mathcal{EF}_A^{\mathscr{C}_{\mathsf{flip}}}$ one can, e.g., repeatedly flip a coin to generate all possible combination of evidences. The resulting evidence is *universal* as it relates any two propositions that any other evidence relates.

However, neither being finitely forced nor exhibiting angelic non-determinism necessarily result in a forcing tripos. One counterexample can be constructed by adding state. In $\mathsf{UFam}(\mathcal{EF}_A^{\mathscr{C}_{\mathsf{flip,lookup}}})$, take $\phi_{\mapsto}$ to be the predicate on $\mathbb{N} \times C$ that maps $\langle n, c\rangle$ to the proposition that is realized by any code *in states mapping address n to the code c*. In this case, the realizer itself has no computational content, but the existence of the realizer guarantees a property of the current state of the system. In addition, take ϕ_{π_C} to be the predicate that maps $\langle n, c\rangle$ to the proposition realized by c in all states. For each n and c, $\phi_{\mapsto}(n,c)$ entails $\phi_{\pi_C}(n,c)$ in $\mathsf{UFam}(\mathcal{EF}_A^{\mathscr{C}_{\mathsf{flip,lookup}}})$ by using lookup_n. However, there is no single deterministic evidence that uniformly entails these predicates, and flip is not able to enumerate all the lookup_n combinators because n is fixed into the combinator rather than passed as an argument to a single lookup combinator.

As another example, one can use a separator to disallow the usage of flip within evidence. Then $\mathsf{UFam}(\mathcal{EF}^{\mathscr{C}_{\mathsf{flip}},\mathcal{S}_\lambda})$ is not finitely forced, but it also does not model the negation of this property (unlike traditional realizability triposes). Intuitively, this is because the separator is not part of the definition of Φ or the type constructors for UFam—it only affects entailment in UFam. As such, $\forall \phi_1, \phi_2.\ \phi_1 \wedge \phi_2 \supset \prod_{i\in\{1,2\}} \phi_i$ still has a realizer in $\mathsf{UFam}(\mathcal{EF}^{\mathscr{C}_{\mathsf{flip}},\mathcal{S}_\lambda})$; there just is no code in the separator that can construct this realizer, which is why it is not modeled by $\mathsf{UFam}(\mathcal{EF}^{\mathscr{C}_{\mathsf{flip}},\mathcal{S}_\lambda})$. But its negation is also not modeled by $\mathsf{UFam}(\mathcal{EF}^{\mathscr{C}_{\mathsf{flip}},\mathcal{S}_\lambda})$ for reasons we discuss next.

4) Models with failure: Traditional realizability triposes have many predicates (on a given set) that model $\top$, the most obvious ones being the predicates realized by all codes or the predicates uniformly realized by just one code. Yet the same is not true for $\bot$ which is modeled by only one predicate: the predicate with no realizers. While in a constructive model one might expect the realizers of the translation of $(\exists n. \mathbb{n}(n)) \wedge (\forall n. \mathbb{n}(n) \supset \bot)$ to similarly involve computations on the natural numbers, but instead it simply has no realizers. The same property holds for all the effectful models discussed so far. For any computational system $\mathscr{C}$, if $\mathcal{S}_\top$ is a valid separator, meaning progress holds for all codes, then the only predicate in $\mathsf{UFam}(\mathcal{EF}_{\mathfrak{D}}^{\mathscr{C}})$ that models $\bot$ is the one with no realizers in any state.

However, when it is possible for computations to fail, then $\mathcal{S}_\top$ is no longer a valid separator. As such, $\mathsf{UFam}(\mathcal{EF}_D^{\mathscr{C}_{\mathsf{fail}},\mathcal{S}_\lambda})$ has many predicates that model $\bot$. For example, the translation of $(\exists n. \mathbb{n}(n)) \wedge (\forall n. \mathbb{n}(n) \supset \bot)$ is realized by (Church-encoded) pairs of natural numbers and computations that will necessarily fail when given a natural number. Because fail is not itself in the separator, evidence that $(\exists n. \mathbb{n}(n)) \wedge (\forall n. \mathbb{n}(n) \supset \bot)$ entails the predicate with no realizers cannot directly invoke fail. Instead, it must be a computation that takes a pair realizing $(\exists n.\, \mathbb{n}(n)) \wedge (\forall n.\, \mathbb{n}(n) \supset \bot)$ and passes a natural number to the second component, possibly computing it from the natural number in the first component.

If one thinks of fail as a computational representation of contradiction, evidence that a predicate is (internally) false is a failure-free computation that can extract a contradiction from the realizers of the predicate. Thus, just as proofs of verity in traditional realizability models correspond to computations that can *construct* a realizer of the target predicate, proofs of falsity in $\mathsf{UFam}(\mathcal{EF}_D^{\mathscr{C}_{\mathsf{fail}}, \mathcal{S}_\lambda})$ correspond to computations that can *extract* contradictions from realizers of the source predicate, making falsity more symmetric with verity.

D. *Predicate reflection in* **Set**-*based triposes*

Now that we have illustrated the breadth of realizability models enabled by evidenced frames, we next move to demonstrating the completeness of evidenced frames as models.

First, we must discuss a property that specifically **Set**-based triposes exhibit. Recall the canonical **Set**-based tripos (also denoted **Set**) of Example V.2. For a context-interpreted-as-a-set Γ, the interpretation of a predicate $\Gamma \vdash \psi$ corresponds to a metatheoretic predicate of Γ. Given such a predicate, we can define the (dependent) type $\Gamma \vdash \{\langle\rangle \in 1 \mid \psi\}$ representing the subset of the unit type that is inhabited if and only if ψ holds. So because **Set**-based triposes model *dependent* higher-order logic over **Set**, we can define the following.

Definition V.6. Given a **Set**-predicate ψ in context Γ we define the corresponding $\mathcal{T}$-predicate $(\!|\psi|\!)$ in context Γ as $\coprod_{\iota_\psi : \{\Gamma \mid \psi\} \to \Gamma} \top$, i.e. the predicate $\Gamma \vdash \exists i : \{\langle\rangle \in 1 \mid \psi\}.\, \top$.

Thus any **Set**-based tripos $\mathcal{T}$ has a predicate constructor $(\!|\cdot|\!)$ where the contained predicate is from **Set** rather than in $\mathcal{T}$. The interpretation of $(\!|\psi|\!)$ is essentially that ψ holds *externally* within the current context, thereby *reflecting* an external notion of truth as an internal predicate. For example, in **Set** we have the predicate $x : X, \vec{x} : \mathcal{P}(X) \vdash x \in \vec{x}$, and the corresponding $\mathcal{T}$-predicate $x : X, \vec{x} : \mathcal{P}(X) \vdash (\!|x \in \vec{x}|\!)$ indicates that $x \in \vec{x}$ holds externally.

The constructor $(\!|\cdot|\!)$ exhibits a number of useful properties.[13] $(\!|\psi|\!)$ entails $(\!|\psi'|\!)$ if ψ entails ψ'; $(\!|\cdot|\!)$ respects substitution; $(\!|e_1 = e_2|\!)$ entails $e_1 = e_2$ (and vice versa); $(\!|\psi_1|\!) \wedge (\!|\psi_2|\!)$ entails $(\!|\psi_1 \wedge \psi_2|\!)$ (and vice versa); $(\!|\psi_1 \supset \psi_2|\!)$ and $(\!|\psi_1|\!) \supset (\!|\psi_2|\!)$ (though *not* vice versa); $(\!|\forall a : A.\, \psi|\!)$ and $\forall a : A.\, (\!|\psi|\!)$ (though *not* vice versa); and $(\!|\exists a : A.\, \psi|\!)$ entails $\exists a : A.\, (\!|\psi|\!)$[14] (and vice versa). In other words, $(\!|\cdot|\!)$ is a morphism of models of *regular* logic from the **Set** tripos to any **Set**-based tripos.

E. *Triposes as evidenced frames*

Lastly, we illustrate that for any **Set**-based tripos there is a corresponding evidenced frame—with propositions Φ given directly by Ω—that UFam maps back to the given tripos. The key challenge is defining the correct notion of evidence for an arbitrary tripos, and to this end we introduce the concept of *reflected axiom schemas*. Generally speaking, a reflected axiom schema indicates a collection of premise-conclusion pairs that it entails, and here we define what it means for a tripos to model such a collection.[15]

Definition V.7. A *reflected axiom schema* of a tripos $\mathcal{T}$ is a **Set**-relation $R \in \mathcal{P}(\Omega \times \Omega)$ for which the following holds:

$$\mathcal{T} \models \phi : \Omega, \phi' : \Omega \mid (\!|\phi \; R \; \phi'|\!), \phi \vdash \phi'$$

The following Lemma shows that reflected axiom schemas provide a means to convert *external* quantification into *internal* quantification, effectively bridging the gap between evidenced frames and triposes.

Lemma V.8. *Given a set Γ and functions $\phi, \phi' : \Gamma \to \Omega$ for a tripos $\mathcal{T}$, if there exists a reflected axiom schema R (externally) satisfying $\forall \gamma \in \Gamma.\ \phi(\gamma) \; R \; \phi'(\gamma)$ (in* **Set***), then (internally) $\mathcal{T} \models \gamma : \Gamma \mid \phi(\gamma) \vdash \phi'(\gamma)$.*

Proof. By assumption, $\mathcal{T} \models \phi : \Omega, \phi' : \Omega \mid (\!|\phi \; R \; \phi'|\!), \phi \vdash \phi'$. By substitution, $\mathcal{T} \models \gamma : \Gamma \mid (\!|\phi(\gamma) \; R \; \phi'(\gamma)|\!), \phi(\gamma) \vdash \phi'(\gamma)$. By assumption, **Set** $\models \Gamma \mid \varnothing \vdash \phi(\gamma) \; R \; \phi'(\gamma)$, and so $\mathcal{T} \models \gamma : \Gamma \mid \varnothing \vdash (\!|\phi(\gamma) \; R \; \phi'(\gamma)|\!)$. By cut, we conclude that $\mathcal{T} \models \gamma : \Gamma \mid \phi(\gamma) \vdash \phi'(\gamma)$ holds. ☐

Definition V.9 (RAS construction)**.** Given a tripos $\mathcal{T}$, the structure $\mathsf{RAS}(\mathcal{T})$ is defined as $(\Phi, E, \cdot \xrightarrow{\cdot} \cdot)$ where:

- Φ is the set Ω
- E is the set of reflected axiom schemas of $\mathcal{T}$
- The evidence relation $\phi_1 \xrightarrow{e} \phi_2$ is defined as $\langle \phi_1, \phi_2 \rangle \in e$

Top Given by χ of $\vdash \top$
Conjunction Given by χ of $\phi_1 : \Omega, \phi_2 : \Omega \vdash \phi_1 \wedge \phi_2$
Universal Implication Given by χ of

$$\phi_1 : \Omega, \vec{\phi} \in \mathcal{P}(\Omega) \vdash \phi_1 \supset (\forall \phi : \Omega.\ (\!|\phi \in \vec{\phi}|\!) \supset \phi)$$

Notice that the definition of universal implication employs predicate reflection. $\mathcal{P}(\Omega)$ corresponds to predicates in **Set**, not predicates in $\mathcal{T}$. As such, the tripos has no direct way to reason about the relationship between Ω and $\mathcal{P}(\Omega)$. Here we use predicate reflection to indirectly represent this relationship through the **Set**-predicate $(\in)$. As such, universal implication ensures that ϕ_1 implies ϕ for all ϕ (externally) in $\vec{\phi}$.

Theorem V.10. $\mathsf{RAS}(\mathcal{T})$ *is an evidenced frame.*

Proof. The key challenge is proving that the required premise-conclusion collections form reflected axiom schemas in $\mathcal{T}$. In particular, given evidence—i.e. reflected axiom schema—e, the corresponding premise-conclusion collection for the required evidence λe must contain all $\phi_1, \phi' \in \Omega$ satisfying

$$\exists \phi_2, \vec{\phi}.\ (\phi' = \phi_2 \supset \vec{\phi}) \wedge (\forall \phi.\ \phi \in \vec{\phi} \implies \langle \phi_1 \wedge \phi_2, \phi \rangle \in e).$$

Notice that λe starts with an existential quantifier. Thus, to prove that λe is a reflected axiom schema of $\mathcal{T}$—i.e. that

[13]This is where the Unique Identity Proofs assumption for **Set** is needed.

[14]This one direction of entailment is exclusively where we use the requirement that surjective substitutions reflect entailment in **Set**-based triposes.

[15]The notion of reflected axiom schema is closely related to the notion of uniform preorders introduced in [28], in that the (sub)set of reflected axiom schemas forms a uniform preorder (or, more specifically, a basic relational object) on Ω. Indeed, our completeness theorem is an extension of Lemma 4.2.5 in [28] to quantifiers and explicit evidence.

$$\mathcal{T} \models \phi_1 : \Omega, \phi' : \Omega \mid (\!|\langle \phi_1, \phi' \rangle \in \lambda e|\!), \phi_1 \vdash \phi'$$

holds—we first need to reflect that external existential as an internal existential so that we can reason about the ϕ_2 and $\vec{\phi}$ that ϕ' was constructed from. The rest of the proof proceeds through using the internal language of $\mathcal{T}$, exploiting the aforementioned properties of $(\!|\cdot|\!)$. □

F. Logical equivalence of evidenced frames and triposes

We now have conversions from evidenced frames to triposes and vice versa, and we would like to say something about the round trips of these conversions. But although the result of these round trips are *essentially* the same as the originals, they are not *exactly* the same. To bridge this gap, we introduce both morphisms between triposes and between evidenced frames, as well as a preorder on such morphisms (so we can formalize when two morphisms are essentially the same). Technically, these components comprise *locally preordered bicategories*, but in this paper we refer to them simply as categories.

Definition V.11 (Categories $\mathbf{Trip}_{\mathsf{ext}}$ and $\mathbf{Trip}_{\mathsf{int}}$)**.** Objects of $\mathbf{Trip}_{\mathsf{ext}}$ and $\mathbf{Trip}_{\mathsf{int}}$ are **Set**-based triposes. A morphism of $\mathbf{Trip}_{\mathsf{ext}}$ from $\mathcal{T}_1$ to $\mathcal{T}_2$ is a set-indexed collection of functions $F_\Gamma : \mathcal{T}_1(\Gamma) \to \mathcal{T}_2(\Gamma)$ satisfying the following properties:

- F is a pseudonatural transformation $\mathcal{T}_1 \Rightarrow \mathcal{T}_2 : \mathbf{Set} \to \mathbf{pHA}$
- $\forall s \in \Gamma \to \Gamma'.\ \forall \phi_1 \in \mathcal{T}_1(\Gamma).\ \prod_s F(\phi_1) \leq F(\prod_s \phi_1)$
- $\forall s \in \Gamma \to \Gamma'.\ \forall \phi_1 \in \mathcal{T}_1(\Gamma).\ F(\coprod_s \phi_1) \leq \coprod_s F(\phi_1)$
- $\exists f \in \Omega_{\mathcal{T}_2} \to \Omega_{\mathcal{T}_1}.\ \mathsf{holds}_{\mathcal{T}_2} \lesseqgtr f^*(F(\mathsf{holds}_{\mathcal{T}_1}))$

A morphism of $\mathbf{Trip}_{\mathsf{int}}$ further requires the function f to have an inverse. Both $\mathbf{Trip}_{\mathsf{ext}}$ and $\mathbf{Trip}_{\mathsf{int}}$ define the preorder $F \leq G$ on morphisms from $\mathcal{T}_1$ to $\mathcal{T}_2$ as $\forall \Gamma.\ F_\Gamma \leq_{\mathbf{pHA}} G_\Gamma$.

Here we defined *two* categories of triposes that coincide except that $\mathbf{Trip}_{\mathsf{ext}}$, which is more common in the literature, has a more relaxed definition than $\mathbf{Trip}_{\mathsf{int}}$. This relaxed definition is sufficient to, e.g., induce a logical functor between the toposes derived from the respective triposes. However, it is insufficient to ensure that, given a higher-order predicate $\Gamma \vdash \phi$ that can be interpreted in any tripos ($[\![\phi]\!]_{\mathcal{T}} \in \mathcal{T}(\Gamma)$), the mapping $F : \mathcal{T}_1 \to \mathcal{T}_2$ preserves the interpretation of ϕ, i.e. $F([\![\phi]\!]_{\mathcal{T}_1}) \lesseqgtr_\Gamma [\![\phi]\!]_{\mathcal{T}_2}$. For example, while the final requirement of $\mathbf{Trip}_{\mathsf{ext}}$ morphisms ensures that *extensional* equality on predicates—$\phi, \phi' : \mathcal{P}(\tau) \vdash \forall t : \tau.\ \mathsf{holds}(\phi(t)) \supset\subset \mathsf{holds}(\phi'(t))$—is preserved, it does not ensure that *intensional* equality on predicates—$\phi, \phi' : \mathcal{P}(\tau) \vdash \phi =_{\mathcal{P}(\tau)} \phi'$—is preserved. The additional requirement on $\mathbf{Trip}_{\mathsf{int}}$ morphisms ensures that intensional equality, and in fact all higher-order-logic predicates, are preserved. For this reason, we refer to $\mathbf{Trip}_{\mathsf{ext}}$ morphisms as *extensional*, and to $\mathbf{Trip}_{\mathsf{int}}$ morphisms as *intensional*, and likewise for concepts such as equivalences.

Definition V.12 (Categories $\mathbf{EF}_{\mathsf{ext}}$ and $\mathbf{EF}_{\mathsf{int}}$)**.** Objects of $\mathbf{EF}_{\mathsf{ext}}$ and $\mathbf{EF}_{\mathsf{int}}$ are evidenced frames. A morphism of $\mathbf{EF}_{\mathsf{ext}}$ from $\mathcal{EF}_1 = \langle \Phi_1, E_1, \cdot \xrightarrow{\cdot}_1 \cdot \rangle$ to $\mathcal{EF}_2 = \langle \Phi_2, E_2, \cdot \xrightarrow{\cdot}_2 \cdot \rangle$ is a function $F : \Phi_1 \to \Phi_2$ satisfying the following properties:

- $\forall e_1.\ \exists e_2.\ \forall \phi_1, \phi_1'.\ \phi_1 \xrightarrow{e_1}_1 \phi_1' \implies F(\phi_1) \xrightarrow{e_2}_2 F(\phi_1')$
- $\exists e_2.\ \top_2 \xrightarrow{e_2}_2 F(\top_1)$
- $\exists e_2.\ \forall \phi_1, \phi_1'.\ F(\phi_1) \wedge_2 F(\phi_1') \xrightarrow{e_2}_2 F(\phi_1 \wedge_1 \phi_1')$
- $\exists e_2.\ \forall \phi_1, \vec{\phi}_1'.\ F(\phi_1) \supset_2 \{F(\phi_1') \mid \phi_1' \in \vec{\phi}_1'\} \xrightarrow{e_2}_2 F(\phi_1 \supset_1 \vec{\phi}_1')$
- $\exists f \in \Phi_2 \to \Phi_1.\ \bigwedge \begin{array}{l} \exists e_2.\ \forall \phi_2.\ \phi_2 \xrightarrow{e_2}_2 F(f(\phi_2)) \\ \exists e_2.\ \forall \phi_2.\ F(f(\phi_2)) \xrightarrow{e_2}_2 \phi_2 \end{array}$

A morphism of $\mathbf{EF}_{\mathsf{int}}$ further requires the function f to have an inverse. Both $\mathbf{EF}_{\mathsf{ext}}$ and $\mathbf{EF}_{\mathsf{int}}$ define the preorder $F \leq G$ on morphisms from $\mathcal{EF}_1$ to $\mathcal{EF}_2$ to hold when there exists evidence $e_2 \in E_2$ satisfying $\forall \phi_1 \in \Phi_1.\ F(\phi_1) \xrightarrow{e_2}_2 G(\phi_1)$.

As with triposes, we have both extensional and intensional morphisms of evidenced frames.

Theorem V.13 (Completeness)**.** *$\mathbf{EF}_{\mathsf{ext}}$ and $\mathbf{Trip}_{\mathsf{ext}}$ are equivalent, as are $\mathbf{EF}_{\mathsf{int}}$ and $\mathbf{Trip}_{\mathsf{int}}$, in the sense that* UFam *and* RAS *extend to biadjoint biequivalences between them.*

Proof. A biadjoint biequivalence includes an equivalence between a tripos $\mathcal{T}$ and the round-trip construction $\mathsf{UFam}(\mathsf{RAS}(\mathcal{T}))$. The counit $\varepsilon_{\mathcal{T}} : \mathsf{UFam}(\mathsf{RAS}(\mathcal{T})) \to \mathcal{T}$ for this equivalence uses Lemma V.8 to map reflected axiom schemas to entailments. Its inverse (up to equivalence) maps an entailment $\phi \leq \phi'$ over Γ to the reflected axiom schema $\{\langle \chi_\phi(\gamma), \chi_{\phi'}(\gamma) \rangle \mid \gamma \in \Gamma\}$. A biadjoint biequivalence also includes an equivalence between an evidenced frame $\mathcal{EF}$ and the round-trip construction $\mathsf{RAS}(\mathsf{UFam}(\mathcal{EF}))$. The unit $\eta_{\mathcal{EF}} : \mathcal{EF} \to \mathsf{RAS}(\mathsf{UFam}(\mathcal{EF}))$ for this equivalence maps evidence e to the reflected axiom schema $\{\langle \phi, \phi' \rangle \mid \phi \xrightarrow{e} \phi'\}$. Its inverse uses the fact that a reflected axiom schema of $\mathsf{RAS}(\mathsf{UFam}(\mathcal{EF}))$ is a relation on propositions for which uniform evidence of entailment exists. UFam and RAS both reuse the given set of propositions, making intensionality of the above morphisms trivial. □

While the unit $\eta_{\mathcal{EF}}$ is able to provide a functional mapping of evidence, its inverse (up to equivalence) is not.[16] Thus a tripos is conceptually an evidenced frame that has forgotten its evidence, forgoing countability and computability properties for an extensional logical description of that evidence. As such, we consider evidenced frames to be the ideal domain for reasoning about the computational properties of realizability models, such as the composition of their computational systems, whereas triposes are specifically suited for reasoning about logical properties of such models, such as the forcing properties that can result from angelically non-deterministic computational systems, as we will discuss again within our connections to implicative algebras.

VI. Relation to Implicative Algebras

Implicative algebras are a simple algebraic structures tailored to factorize the model-theoretic constructions underlying

[16]One could require evidenced-frame morphism to provide explicit function converting evidence. Such morphisms conceptually preserve not just *that* an entailment holds but also *why* it holds, which tripos morphisms cannot do.

forcing and realizability (in both intuitionistic and classical logic) [10]. Miquel gives a simple translation from implicative algebras to triposes [10], and an impressive translation from triposes to implicative algebras [11]. This section demonstrates that both of these translations can be factored through the ones between evidenced frames and triposes.

While implicative algebras offer a useful and concise framework for reasoning about the algebraic foundation of realizability and forcing, we argue that evidenced frames are more convenient for models arising from effectful computations. Indeed, having a particular realizability interpretation at hand, evidenced frames allow us to abstract its core structure while directly reflecting the relation between evidence (the expressions) and formulas (the propositions). Implicative algebras, on the other hand, blur the distinction between proofs and propositions, which supports the concise representation but complicates the computational structure underlying the model.

A. *Implicative algebras*

This section reviews the notion of an implicative algebra. One key feature of implicative algebras is that they capture both the formulas and the realizers arising from (classical) realizability [10]. For their logical facet, they are defined using complete meet-semilattices (for universal quantification) with an internal binary operation (for implication).

Definition VI.1 (Implicative Structure)**.** An *implicative structure* is a triple $\mathcal{A} = \langle A, \preccurlyeq, \rightarrow \rangle$ where $\langle A, \preccurlyeq \rangle$ is a preordered[17] complete meet-semilattice and $\rightarrow$ is a binary operation, called the *implication* of $\mathcal{A}$, that fulfills the following axioms:

- if $a' \preccurlyeq a$ and $b \preccurlyeq b'$ then $(a \rightarrow b) \preccurlyeq (a' \rightarrow b')$.
- $\curlywedge_{b\in\vec{b}}(a \rightarrow b) \preccurlyeq a \rightarrow \curlywedge_{b\in\vec{b}} b$.

An implicative structure $\mathcal{A}$ provides a trivial embedding (using the meet for the universal quantifier and the arrow for the implication) of System F formulas [10]. More interestingly, we can also encode λ-terms with $\mathcal{A}$-valued parameters as follows:

$$ab \triangleq \curlywedge\{c \mid a \preccurlyeq b \rightarrow c\} \qquad \lambda f \triangleq \curlywedge_{a\in\mathcal{A}}(a \rightarrow f(a))$$

We denote by $t^{\mathcal{A}}$ (resp. $A^{\mathcal{A}}$) the interpretation of a term t (resp. System-F formula A) in $\mathcal{A}$, which are at the same time:

- sound with respect to β-reduction, in the sense that $t \rightarrow_\beta u$ implies $t^{\mathcal{A}} \preccurlyeq u^{\mathcal{A}}$;
- adequate with respect to typing, insofar as if t is of type T, then we have $t^{\mathcal{A}} \preccurlyeq T^{\mathcal{A}}$ (*i.e.* t *"realizes"* T).

Implicative structures are thus suited to interpret both λ-terms and their types. To give an account for realizability models, one then has to define a notion of validity.

Definition VI.2 (Separator)**.** Let $\langle \mathcal{A}, \preccurlyeq, \rightarrow \rangle$ be an implicative structure. We call a *separator* over $\mathcal{A}$ any set $\mathcal{S} \subseteq \mathcal{A}$ such that, for all $a, b \in A$, the following conditions hold:

- $\mathbf{K}^{\mathcal{A}} = \curlywedge_{a,b}\, a \rightarrow b \rightarrow a \in \mathcal{S}$
- $\mathbf{S}^{\mathcal{A}} = \curlywedge_{a,b,c}(a \rightarrow b \rightarrow c) \rightarrow (a \rightarrow b) \rightarrow a \rightarrow c \in \mathcal{S}$,
- If $a \in \mathcal{S}$ and $a \preccurlyeq b$, then $b \in \mathcal{S}$,
- If $(a \rightarrow b) \in \mathcal{S}$ and $a \in \mathcal{S}$, then $b \in \mathcal{S}$.

A separator $\mathcal{S}$ is said to be *consistent* if $\perp \notin \mathcal{S}$ and *classical* if $\mathbf{cc}^{\mathcal{A}} = \curlywedge_{a,b}((a \rightarrow b) \rightarrow a) \rightarrow a \in \mathcal{S}$.

Definition VI.3 (Implicative Algebra)**.** An *implicative algebra* is a quadruple $\langle A, \preccurlyeq, \rightarrow, \mathcal{S} \rangle$ where $\mathcal{A} = \langle A, \preccurlyeq, \rightarrow \rangle$ is an implicative structure and $\mathcal{S}$ is a separator over $\mathcal{A}$. An implicative algebra is called *classical* if its separator is.

Intuitively, thinking of elements of an implicative structure as truth values, a separator comprises the set that distinguishes the valid formulas (similar to a filter in a Boolean algebra). Considering the elements as terms, the separator should rather be viewed as the set of valid realizers. In particular, it contains any λ-terms with $\mathcal{S}$-valued parameters [10, Prop. 3.4].

Given a separator, we write $a \vdash_{\mathcal{S}} b$, and say that *a entails* b, if $a \rightarrow b \in \mathcal{S}$. As the next section shows, this relation is reminiscent of the evidence relation in Definition III.1. By defining the connectives using products $a \times b$ and sums $a + b$ through their usual impredicative encodings[18] we obtain a Heyting prealgebra with the preorder $\vdash_{\mathcal{S}}$.

Given any set I, we can then endow the product $\mathcal{A}^I$ (which is an implicative structure) with the *uniform* separator

$$\mathcal{S}[I] \triangleq \{a \in \mathcal{A}^I \mid \exists s \in S.\, \forall i \in I.\, s \preccurlyeq a_i\}$$

This provides us with a Heyting prealgebra and leads to the definition of the so-called implicative tripos $\mathcal{T}^{\mathcal{A}}$:

$$\mathcal{T}^{\mathcal{A}}(I) = \langle \mathcal{A}^I, \vdash_{\mathcal{S}[I]} \rangle \qquad \mathcal{T}^{\mathcal{A}}(f) = \lambda \langle a_i \rangle_{i\in I}.\, \langle a_{f(j)} \rangle_{j\in J}$$

B. *From implicative algebras to evidenced frames*

Following the intuition that the separator is the set of valid realizers, we easily show the following.

Definition VI.4. For $\mathcal{A} = \langle A, \preccurlyeq, \rightarrow, \mathcal{S} \rangle$ an implicative algebra, we define $\mathsf{UEF}(\mathcal{A}) \triangleq \langle A, \mathcal{S}, \cdot \xrightarrow{\cdot} \cdot \rangle$ where the entailment relation is defined by $a \xrightarrow{e} b \triangleq e \preccurlyeq a \rightarrow b$.

Theorem VI.5. *Given an implicative algebra $\mathcal{A}$,* $\mathsf{UEF}(\mathcal{A})$ *is an evidenced frame.*

Theorem VI.6. *The implicative-algebra-to-tripos construction in [10] is both intensionally and extensionally equivalent (as pseudofunctors) to the composition* $\mathsf{UEF}\,;\mathsf{UFam}$.

The construction of an evidenced frame out of any implicative algebra allows us to retrieve all the examples from [10]: Heyting and Boolean algebras, partial equivalence relations, reducibility candidates, etc. Implicative algebras also encompass several presentations of Krivine classical realizability, for instance by means of *ordered combinatory algebras* [12] or *abstract Krivine structure* [29]. In particular, composing the definitions of these implicative algebras with the definition of UEF, results in consistent, classical evidenced frames which are not finitely forced. These include, for instance, a model of ZF in which neither the Axiom of Choice nor the Continuum Hypothesis hold [30], or a class of models of ZF accounting for a hierarchy of parallel computations [20].

[17]Normally antisymmetry is further required.

[18]That is to say that we define $a \times b \triangleq \curlywedge_{c\in\mathcal{A}}((a \rightarrow b \rightarrow c) \rightarrow c)$ and $a + b \triangleq \curlywedge_{c\in\mathcal{A}}((a \rightarrow c) \rightarrow (b \rightarrow c) \rightarrow c)$.

C. *The case of total combinatory algebras*

So far, we have shown how any implicative algebra induces a canonical evidenced frame. This provides us with two different constructions of an evidenced frame from a total combinatory algebra: the direct one (see Example III.4) and via an intermediate implicative algebra. While the obtained evidenced frames are different, they induce equivalent triposes.

Let us consider a computational system $\mathscr{C}$ defining a PCA $(C, \cdot)$. The quadruple $\mathcal{A}^{\mathscr{C}} = \langle \mathcal{P}(C), \subseteq, \rightarrow, \mathcal{S}^C \rangle$, where $\mathcal{S}^C$ is $\mathcal{P}(C) \setminus \emptyset$ and $\rightarrow$ is the Kleene arrow $a \rightarrow b \triangleq \{e \in C : \forall x \in a.\ e \cdot x{\downarrow} \wedge\ e \cdot x \in b\}$, defines an implicative algebra [10, Fact 2.5]. As shown in Theorem VI.5, it induces an evidenced frame $\mathsf{UEF}(\mathcal{A}^{\mathscr{C}}) = \langle \mathcal{P}(C), \mathcal{S}^C, \cdot \xrightarrow{\cdot} \cdot \rangle$. While the definition of $\mathcal{EF}^{\mathscr{C}}$ and $\mathsf{UEF}(\mathcal{A}^{\mathscr{C}})$ are slightly different, with the evidence for the former being C and the evidence for the latter being $\mathcal{S}^C$, they are nonetheless equivalent.

Theorem VI.7. *The evidenced frames $\mathcal{EF}^{\mathscr{C}}$ and $\mathsf{UEF}(\mathcal{A}^{\mathscr{C}})$ are intensionally equivalent.*

D. *From evidenced frames to implicative algebras*

We shall now see how we can translate an evidenced frame $\mathcal{EF}$ to an implicative algebra $\mathsf{FIA}(\mathcal{EF})$ whose induced evidenced frame is extensionally equivalent to $\mathcal{EF}$. To define FIA, we follow the guidelines of the construction of an implicative algebra, which we write $\mathcal{A}^{\mathcal{T}}$, from a tripos $\mathcal{T}$ defined by Miquel to prove the (extensional) completeness of implicative triposes w.r.t. **Set**-based triposes [11].

Let us a fix an evidenced frame $\mathcal{EF} = \langle \Phi, E, \cdot \xrightarrow{\cdot} \cdot \rangle$. Regrettably the ordering given by $\exists e.\, \phi \xrightarrow{e} \phi'$ lacks the algebraic structure required by implicative algebras. Thus, as in [11], we build from Φ a larger preordered set $\langle A_0, \leq_0 \rangle$:

$$\frac{\phi \in \Phi}{\phi \in A_0} \qquad \frac{\vec{\phi} \in \mathcal{P}(\Phi) \quad \alpha \in A_0}{\vec{\phi} \mapsto \alpha \in A_0} \qquad \Bigg| \qquad \frac{}{\phi \leq_0 \phi} \qquad \frac{\vec{\phi} \subseteq \vec{\phi}' \quad \alpha \leq_0 \beta}{\vec{\phi} \mapsto \alpha \leq_0 \vec{\phi}' \mapsto \beta}$$

We then define $A = \mathcal{P}_{\uparrow}(A_0)$—the upwards-closed subsets—which forms a complete lattice $(A, \supseteq)$. To obtain an implicative structure, we first define the function $\varphi_0 : A_0 \rightarrow \Phi$ recursively as $\varphi_0(\phi) = \phi$ and $\varphi_0(\vec{\phi} \mapsto \alpha) = (\prod \vec{\phi}) \supset \{\varphi_0(\alpha)\}$. We then equip $\mathcal{P}_{\uparrow}(A_0)$ with the arrow given by:

$$a \rightarrow b \triangleq \{\vec{\phi} \mapsto \beta \mid (\forall \alpha \in a.\ \varphi_0(\alpha) \in \vec{\phi}) \wedge \beta \in b\}$$

Definition VI.8. For $\mathcal{EF}$ an evidenced frame, we define $\mathsf{FIA}(\mathcal{EF}) \triangleq \langle A, \supseteq, \rightarrow, \mathcal{S} \rangle$, where the separator $\mathcal{S}$ is given by $\{a \in A \mid \exists e \in E.\ \forall \alpha \in a.\ \top \xrightarrow{e} \varphi_0(\alpha)\}$.

Theorem VI.9. *$\mathsf{FIA}(\mathcal{EF})$ defines an implicative algebra.*[19]

E. *Extensional equivalence with evidenced frames*

Now we consider the relationships between the above constructions, but to do so we must first define the categorical structure of implicative algebras.

[19]As with the other constructions, FIA extends to a pseudofunctor $\mathsf{FIA} : \mathbf{EF}_{\mathsf{ext}} \rightarrow \mathbf{IA}_{\mathsf{ext}}$; however, unlike the other constructions, surprisingly FIA does not appear to extend to a pseudofunctor from $\mathbf{EF}_{\mathsf{int}}$ to $\mathbf{IA}_{\mathsf{int}}$ due to Miquel's propositional encoding φ_0 losing the computational structure of A_0.

Definition VI.10 (Categories $\mathbf{IA}_{\mathsf{ext}}$ and $\mathbf{IA}_{\mathsf{int}}$)**.** Objects of $\mathbf{IA}_{\mathsf{ext}}$ and $\mathbf{IA}_{\mathsf{int}}$ are implicative algebras. A morphism of $\mathbf{IA}_{\mathsf{ext}}$ from $\mathcal{A}_1 = \langle A_1, \preccurlyeq_1, \rightarrow_1, \mathcal{S}_1 \rangle$ to $\mathcal{A}_2 = \langle A_2, \preccurlyeq_2, \rightarrow_2, \mathcal{S}_2 \rangle$ is a function $F : A_1 \rightarrow A_2$ satisfying the following properties:

- $\forall s_1.\ F(s_1) \in \mathcal{S}_2$
- $\forall s_1.\ \curlywedge_{s_1 \preccurlyeq_1 a_1 \rightarrow_1 a_1' \rightarrow_1 a_1''} F(a_1) \rightarrow_2 F(a_1') \rightarrow_2 F(a_1'') \in \mathcal{S}_2$
- $\curlywedge_{\vec{a}_1 \subseteq A_1} (\curlywedge_{a_1 \in \vec{a}_1} F(a_1)) \rightarrow_2 F(\curlywedge_{a_1 \in \vec{a}_1} a_1) \in \mathcal{S}_2$
- $\curlywedge_{a_1, a_1'} (F(a_1) \rightarrow_2 F(a_1')) \rightarrow_2 F(a_1 \rightarrow_1 a_1') \in \mathcal{S}_2$
- $\exists f \in A_2 \rightarrow A_1.\ \bigwedge \begin{matrix} \curlywedge_{a_2} a_2 \rightarrow_2 F(f(a_2)) \in \mathcal{S}_2 \\ \curlywedge_{a_2} F(f(a_2)) \rightarrow_2 a_2 \in \mathcal{S}_2 \end{matrix}$

A morphism of $\mathbf{IA}_{\mathsf{int}}$ further requires the function f to have an inverse. Both $\mathbf{IA}_{\mathsf{ext}}$ and $\mathbf{IA}_{\mathsf{int}}$ define the preorder $F \leq G$ on morphisms from $\mathcal{A}_1$ to $\mathcal{A}_2$ as $\curlywedge_{a_1} F(a_1) \rightarrow_2 G(a_1) \in \mathcal{S}_2$.

With this we can formally factor Miquel's tripos constructions through evidenced frames.

Theorem VI.11. *The tripos-to-implicative-algebra construction in [11] is extensionally equivalent (as pseudofunctors) to the composition* $\mathsf{RAS}\,;\mathsf{FIA}$.

Furthermore, we can formalize both the completeness and incompleteness of implicative algebras.

Theorem VI.12. $\mathbf{EF}_{\mathsf{ext}}$ *and* $\mathbf{IA}_{\mathsf{ext}}$ *are equivalent in the sense that* UEF *and* FIA *extend to a biadjoint biequivalence between them. However,* $\mathbf{EF}_{\mathsf{int}}$ *and* $\mathbf{IA}_{\mathsf{int}}$ *are not equivalent through* UEF *and* FIA.

This states that, although implicative algebras are closely related to evidenced frames, the conversions between them change the set representing propositions too much to retain (intensional) equality on predicates. In particular, FIA increases the cardinality of the set of propositions, enabling distinctions that did not previously exist. This suggests that, while implicative algebras are complete with respect to triposes as an intermediate step to toposes, they are not complete with respect to triposes as models of higher-order logic themselves.

This biequivalence can provide more insight on the relationship between evidenced frames and implicative algebras. In particular, the counit $\varepsilon_{\mathcal{A}} : \mathsf{FIA}(\mathsf{UEF}(\mathcal{A})) \rightarrow \mathcal{A}$ maps separators of the elaborate construction to separators of $\mathcal{A}$, and it does so despite the fact that the seperator of $\mathcal{A}$ was not used in the construction of the elements of $\mathsf{FIA}(\mathsf{UEF}(\mathcal{A}))$. This shows that an implicative algebra is able to reconstruct each element of its separator from its extensional behavior as evidence. Thus, implicative algebras are conceptually evidenced frames that have been endowed with additional structure to reconstruct evidence from its extensional behavior on propositions.

VII. Conclusion

This paper presented evidenced frames and showed that they serves as a unifying framework for building realizability models supporting broader notions of computation. We demonstrated the framework's *flexibility* by illustrating how different effectful notions of computation can be naturally embedded into it; its *utility* by showing how, via uniform construction, it

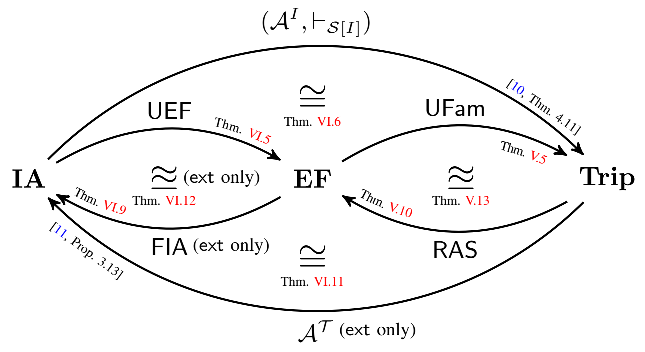

Figure 1. Implicative algebras, evidenced frames, and triposes

induces models that go beyond traditional realizability models; its *generality* by proving its completeness; and its *uniformity* by showing it subsumes the connections between implicative algebras and triposes (as summarized in Figure 1).

Triposes were first identified by Hyland, Johnstone and Pitts as a common generalization of the effective topos and H-valued sets, realizability models being "*the spur for developing that theory*" [6]. This paper identifies a simpler structure, namely evidenced frames, which factorizes the construction of a tripos from realizability models while being complete with respect to **Set**-based triposes. However, while evidenced frames are complete with respect to **Set**-based triposes, triposes are *not* complete with respect to models of higher-order logic, as noted by Pitts himself [13, Example 4.8]. Higher-order logic requires that for every predicate there exists some internal representation of that predicate. For a tripos, that existential is interpreted externally, but in general it need only internally. We plan to explore the potential connections between evidenced frames and such weaker models.

Evidenced frames also emphasize the *evidence* that an entailment holds, rather than just the fact that it holds. This highlights the role of computation in realizability models, where evidence is an actual code rather than predicates on codes. As such, it would be interesting to explore variations of evidenced frames where evidence has explicit computational (e.g. combinatory) structure and where morphisms preserve such computational structure. For example, while Countable Choice (CC) is commonly assumed across constructive discourse, adding demonic non-determinism to the computational system negates it [8], which suggests a certain fragility to CC as a constructive principle. Then again, various works have used memoization techniques to prove CC (e.g., [31]–[34]), and [8] also showed that extending the computational system with memoization restores CC even in the presence of demonic non-determinism, suggesting that it is somehow a more robust model of CC. Perhaps one can show that morphisms preserving computational structure necessarily preserve such an explicit implementation of CC. Such computational morphisms of computational evidenced frames might provide more robust constructive foundations, where one can prove that not only does an axiom hold within a particular realizability model, but it also holds in all extensions of that model regardless of what additional effects might be introduced.

References

[1] S. C. Kleene, "On the interpretation of intuitionistic number theory," *The Journal of Symbolic Logic*, vol. 10, no. 4, 1945.
[2] J. Van Oosten, *Realizability: an Introduction to its Categorical Side*. Elsevier, 2008, vol. 152.
[3] S. Feferman, "A language and axioms for explicit mathematics," in *Algebra and Logic*, 1975.
[4] P. J. Hofstra, "Partial combinatory algebras and realizability toposes," 2004, https://mysite.science.uottawa.ca/phofstra/Kananaskis.pdf.
[5] J. L. Bates and R. L. Constable, "Proofs as programs," *TOPLAS*, vol. 7, no. 1, 1985.
[6] J. M. E. Hyland, P. T. Johnstone, and A. M. Pitts, "Tripos theory," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 88, no. 2, 1980.
[7] D. C. McCarty, "Realizability and recursive mathematics," Ph.D. dissertation, The University of Edinburgh, 1985.
[8] L. Cohen, S. A. Faro, and R. Tate, "The effects of effects on constructivism," in *MFPS*, 2019.
[9] A. Bauer, "Realizability as connection between constructive and computable mathematics," in *Computability and Complexity in Analysis*, 2005.
[10] A. Miquel, "Implicative algebras: A new foundation for realizability and forcing," *Mathematical Structures in Computer Science*, vol. 30, no. 5, 2020.
[11] A. Miquel, "Implicative algebras II: Completeness w.r.t. Set-based triposes," 2020, https://arxiv.org/abs/2011.09085v1.
[12] W. F. Santos, J. Frey, M. Guillermo, O. Malherbe, and A. Miquel, "Ordered combinatory algebras and realizability," *Mathematical Structures in Computer Science*, vol. 27, no. 3, 2017.
[13] A. M. Pitts, "Tripos theory in retrospect," *Mathematical Structures in Computer Science*, vol. 12, no. 3, 2002.
[14] B. Jacobs, *Categorical Logic and Type Theory*. Elsevier, 1999.
[15] J. M. E. Hyland, "The effective topos," *Studies in Logic and the Foundations of Mathematics*, vol. 110, 1982.
[16] D. Scott, "Lambda calculus and recursion theory (preliminary version)," in *Scandinavian Logic Symposium*, ser. Studies in Logic and the Foundations of Mathematics, 1975, vol. 82.
[17] P. T. Johnstone, "The point of pointless topology," *Bulletin of the American Mathematical Society*, vol. 8, no. 1, 1983.
[18] D. Scott, "Completeness and axiomatizability in many-valued logic," in *Proceedings of the Tarski Symposium*, vol. 25, 1974.
[19] E. Moggi, "Notions of computation and monads," *Information and Computation*, vol. 93, no. 1, 1991.
[20] G. Geoffroy, "Classical realizability as a classifier for nondeterminism," in *LICS*, 2018.
[21] E. Miquey and H. Herbelin, "Realizability interpretation and normalization of typed call-by-need λ-calculus with control," in *FoSSaCS*, 2018.
[22] P.-M. Pédrot and N. Tabareau, "Failure is not an option an exceptional type theory," in *ESOP*, 2018.
[23] S. A. Kripke, "Semantical considerations on modal logic," *Acta Philosophica Fennica*, vol. 16, no. 1963, 1963.
[24] N. F. W. Voorneveld, "Non-deterministic effects in a realizability model," *ENTCS*, vol. 336, 2018.
[25] T. G. Griffin, "A formulae-as-type notion of control," in *POPL*, 1990.
[26] P.-L. Curien and H. Herbelin, "The duality of computation," in *ICFP*, 2000.
[27] J.-L. Krivine, "Realizability in classical logic. In Interactive models of computation and program behaviour," *Panoramas et Synthèses*, vol. 27, 2009.
[28] J. Frey, "A fibrational study of realizability toposes," Ph.D. dissertation, Université Paris Diderot, 2014.
[29] T. Streicher, "Krivine's classical realisability from a categorical perspective," *Mathematical Structures in Computer Science*, vol. 23, no. 6, 2013.
[30] J.-L. Krivine, "Realizability algebras II : New models of ZF + DC," *LMCS*, 2012.
[31] S. Berardi, M. Bezem, and T. Coquand, "On the computational content of the axiom of choice," *Journal of Symbolic Logic*, vol. 63, no. 2, 1998.
[32] H. Herbelin, "A constructive proof of dependent choice, compatible with classical logic," in *LICS*, 2012.
[33] V. Blot and C. Riba, "On bar recursion and choice in a classical setting," in *Programming Languages and Systems*, 2013.
[34] E. Miquey, "A sequent calculus with dependent types for classical arithmetic," in *LICS*, 2018.