Formalizing Implicative Algebras in Coq

Étienne Miquey

Équipe Gallinette
Inria, LS2N (CNRS)
etienne.miquey@inria.fr

Abstract. We present a Coq formalization of Alexandre Miquel's *implicative algebras* [18], which aim at providing a general algebraic framework for the study of classical realizability models. We first give a self-contained presentation of the underlying *implicative structures*, which roughly consist of a complete lattice equipped with a binary law representing the implication. We then explain how these structures can be turned into models by adding separators, giving rise to the so-called implicative algebras. Additionally, we show how they generalize Boolean and Heyting algebras as well as the usual algebraic structures used in the analysis of classical realizability.

1 Introduction

Krivine classical realizability It is well-known since Griffin's seminal work [10] that a classical Curry-Howard correspondence can be obtained by adding control operators to the λ-calculus. Several calculi were born from this idea, amongst which Krivine λ_c-calculus [13], defined as the λ-calculus extended with Scheme's `call/cc` operator (for *call-with-current-continuation*). Elaborating on this calculus, Krivine's developed in the late 90s the theory of *classical realizability* [13], which is a complete reformulation of its intuitionistic twin. Originally introduced to analyze the computational content of classical programs, it turned out that classical realizability also provides interesting semantics for classical theories. While it was first tailored to Peano second-order arithmetic (*i.e.* second-order type systems), classical realizability actually scales to more complex classical theories, *e.g.* ZF [14], and gives rise to surprisingly new models. In particular, its generalizes Cohen's forcing [14,17] and allows for the direct definition of a model in which neither the continuum hypothesis nor the axiom of choice hold [16].

Algebraization of classical realizability During the last decade, the study of the algebraic structure of the models that classical realizability induces have been an active research topic. This line of work was first initiated by Streicher, who proposed the concept of *abstract Krivine structure* [24], followed by Ferrer, Frey, Guillermo, Malherbe and Miquel who introduced other structures peculiar to classical realizability [6,7,5,8,9]. In addition to the algebraic study of classical realizability models, these works had the interest of building the bridge with the algebraic structures arising from intuitionistic realizability. In particular,

Streicher showed in [24] how classical realizability could be analyzed in terms of *triposes* [21], the categorical framework arising from intuitionistic realizability models, while the later work of Ferrer *et al.* [6,7] connected it to Hofstra and Van Oosten's notion of *ordered combinatory algebras* [12]. More recently, Alexandre Miquel introduced the elegant concepts of *implicative structure* and *implicative algebra*[18][1], which appear to encompass the previous approaches and which we present in this paper.

Implicative structures In addition to providing an algebraic framework conducive to the analysis of classical realizability, an important feature of implicative structures is that they allow us to identify *realizers* (*i.e.* λ-terms) and *truth values* (*i.e.* formulas). Concretely, implicative structures are complete lattices equipped with a binary operation $a \to b$ satisfying properties coming from the logical implication. As we will see, they indeed allow us to interpret both the formulas and the terms in the same structure. For instance, the ordering relation $a \leq b$ will encompass different intuitions depending on whether we regard a and b as formulas or as terms. Namely, $a \leq b$ will be given the following meanings:

- the formula a is a *subtype* of the formula b;
- the term a is a *realizer* of the formula b;
- the realizer a is *more defined* than the realizer b.

The last item corresponds to the intuition that if a is a realizer of all the formulas of which b is a realizer, a is more precise than b, or more powerful as a realizer.

In terms of the Curry-Howard correspondence, this means that not only do we identify types with formulas and proofs with programs, but *we also identify types and programs*.

Implicative Algebras Because we consider formulas as realizers, any formula will be at least realized by itself. In particular, the lowest formula $\bot$ is realized. While this can be dazzling at first sight, it merely reflects the fact that implicative structures do not come with an intrinsic criterion of consistency. To overcome this, we will introduce the notion of *separator*, which is similar to the usual notion of filter for Boolean algebras. *Implicative algebras* will be defined as implicative structures equipped with a separator. As we shall see, they capture the algebraic essence of classical realizability models. In particular, we will embed both the λ_c-calculus and its second-order type system in such a way that the adequacy is preserved. Implicative algebras therefore appear to be the adequate algebraic structure to study classical realizability and the models it induces.

Coq formalization The formalization of implicative algebras that we present in this paper has been written using the Coq proof assistant. It was written during the author's PhD, as a way of (1) checking the correctness of implicative algebras

[1] Independently, very similar structures can be found in Frédéric Ruyer's Ph.D. thesis [22] under the name of *applicative lattices*.

properties (which, at the time, were neither published nor formally written with their proofs), and (2) easing the further study of similar structures[2].

Technically, it relies on Charguéraud's *locally nameless representation* of λ-terms [2]. and the corresponding LN library[3], which was developed at the occasion of the POPLmark challenge [1]. As for the different algebraic structures evoked in the paper, we systematically represent them as classes using Sozeau-Oury's `Class` mechanism [23]. Interestingly, apart from the technical details mentioned above to define terms (and types), the formalization of the different results mostly follows the corresponding pen and paper proofs.

Outline of the paper We begin by briefly recalling the structures of classical realizability models in Section 2. We then present in Section 3 the concept of implicative structures and explain how it generalizes well-known algebraic structures[4]. We then show in Section 4 how λ_c-terms and second-order types can be adequately embedded within implicative structures. Finally, we introduce implicative algebras in Section 5. We study their internal logic and finally explain how they give rise to models. *It should be clear to the reader that the notion of implicative algebra and its properties are due to Alexandre Miquel [18].*

The theorems in the paper are hyperlinked with their formalizations in the Coq development[5]. Detailed proofs can be found in [19, Chapter 10] from which this paper is partially taken.

2 Krivine classical realizability

Due to the lack of space, it is not possible to fully introduce here Krivine classical realizability and its models defined using the machinery of the λ_c-calculus[6]. Rather than that, we choose to present it through the lenses of Streicher's *abstract Krivine structures* (AKS), which are merely an axiomatization of the Krivine abstract machine for the λ_c-calculus viewed as an algebraic structure:

Definition 1. (AKS) *An* abstract Krivine structure *is given by a septuple* $(\boldsymbol{\Lambda}, \boldsymbol{\Pi}, \mathtt{app}, \mathtt{push}, \mathtt{k}_{\_}, \boldsymbol{k}, \boldsymbol{s}, \boldsymbol{cc}, \mathbf{PL}, \perp\!\!\!\perp)$ *where:*

1. $\boldsymbol{\Lambda}$ *and* $\boldsymbol{\Pi}$ *are non-empty sets, called the* terms *and the* stacks *of the AKS;*
2. $\mathtt{app} : t, u \mapsto tu$ *is a function (called* application*) from* $\boldsymbol{\Lambda} \times \boldsymbol{\Lambda}$ *to* $\boldsymbol{\Lambda}$*;*
3. $\mathtt{push} : t, \pi \mapsto t \cdot \pi$ *is a function (called* push*) from* $\boldsymbol{\Lambda} \times \boldsymbol{\Pi}$ *to* $\boldsymbol{\Pi}$*;*
4. $\mathtt{k}_{\_} : \pi \mapsto \mathtt{k}_\pi$ *is a function from* $\boldsymbol{\Pi}$ *to* $\boldsymbol{\Lambda}$ *(*$\mathtt{k}_\pi$ *is called a* continuation*);*
5. $\boldsymbol{k}$*,* $\boldsymbol{s}$ *and* $\boldsymbol{cc}$ *are three distinguished terms of* $\boldsymbol{\Lambda}$*;*

[2] Namely, one goal of the author's PhD work was to define similar algebras based on the decomposition of the implication as $\neg A \vee B$ and $\neg(A \wedge \neg B)$ (see [19]).

[3] In doing so, our development implicitly relies on assumptions of functional and propositional extensionnality, which we do not need nor use.

[4] We will not recall the definition of lattices, Heyting algebras and so on, for a more detailed introduction we refer the reader to [19, Chapter 9].

[5] Available at `https://gitlab.com/emiquey/ImplicativeAlgebras/`.

[6] For a detailed introduction on this topic, we refer the reader to [13] or [19].

6. $\perp\!\!\!\perp \subseteq \boldsymbol{\Lambda} \times \boldsymbol{\Pi}$ *(called the* pole*) is a relation between terms and stacks, also written* $t \star \pi \in \perp\!\!\!\perp$*. This relation fulfills the following axioms for all terms* $t, u, v \in \boldsymbol{\Lambda}$ *and all stacks* $\pi, \pi' \in \boldsymbol{\Lambda}$*:*

$$\begin{array}{rl|rl} t \star u \cdot \pi \in \perp\!\!\!\perp & \Rightarrow tu \star \pi \in \perp\!\!\!\perp & t \star \boldsymbol{k}_\pi \cdot \pi \in \perp\!\!\!\perp & \Rightarrow \boldsymbol{cc} \star t \cdot \pi \in \perp\!\!\!\perp \\ t \star \pi \in \perp\!\!\!\perp & \Rightarrow \boldsymbol{k} \star t \cdot u \cdot \pi \in \perp\!\!\!\perp & t \star \pi \in \perp\!\!\!\perp & \Rightarrow \boldsymbol{k}_\pi \star t \cdot \pi' \in \perp\!\!\!\perp \\ tv(uv) \star \pi \in \perp\!\!\!\perp & \Rightarrow \boldsymbol{s} \star t \cdot u \cdot v \cdot \pi \in \perp\!\!\!\perp & & \end{array}$$

7. $\mathbf{PL} \subseteq \boldsymbol{\Lambda}$ *is a subset of* $\boldsymbol{\Lambda}$ *(whose elements are called the* proof-like terms*), which contains* $\boldsymbol{k}, \boldsymbol{s}, \boldsymbol{cc}$ *and is closed under application.*

Given any subset of stacks $X \subseteq \boldsymbol{\Pi}$ (which we call a *falsity value*), we write $X^{\perp\!\!\!\perp}$ for its orthogonal set with respect to the pole:

$$X^{\perp\!\!\!\perp} \quad \triangleq \quad \{t \in \boldsymbol{\Lambda} : \forall \pi \in X, t \star \pi \in \perp\!\!\!\perp\}$$

Orthogonality for subsets $X \subseteq \boldsymbol{\Lambda}$ (*i.e.* a *truth value*) is defined identically. Intuitively, classical realizability models are mainly given by the choice of the sets $\perp\!\!\!\perp$ and $\mathbf{PL}$ together with the interpretation of formulas as falsity values. Valid formulas are the one admitting a proof-like *realizer*, that is to say a term $t \in \mathbf{PL}$ such that $t \in \|A\|^{\perp\!\!\!\perp}$ where $\|A\| \in \mathcal{P}(\boldsymbol{\Pi})$ is the falsity value of A.

3 Implicative structures

3.1 Definition

Intuitively, *implicative structures* are tailored to represent both the formulas of second-order logic and realizers arising from Krivine's λ_c-calculus. We shall see in the sequel how they indeed allow us to define λ-terms, but let us introduce them by focusing on their logical facet. We are interested in formulas of second-order logic, that is to say of system F, which are defined by a simple grammar:

$$A, B ::= X \mid A \Rightarrow B \mid \forall X.A$$

Implicative structures are therefore defined as meet-complete lattices (for the universal quantification) with an internal binary operation satisfying the properties of the implication:

Definition 2. *An* implicative structure *is a complete meet-semilattice* $(\mathcal{A}, \preccurlyeq)$ *equipped with a binary operation* $(a, b) \mapsto (a \to b)$*, called the* implication *of* $\mathcal{A}$ *, that fulfills the following axioms:*

1. *Implication is anti-monotonic with respect to its first operand and monotonic with respect to its second operand, in the sense that for all* $a, a_0, b, b_0 \in \mathcal{A}$*:*

   (*Variance*) *If* $a_0 \preccurlyeq a$ *and* $b \preccurlyeq b_0$ *then* $(a \to b) \preccurlyeq (a_0 \to b_0)$.

2. *Arbitrary meets distribute over the second operand of implication, in the sense that for all* $a \in \mathcal{A}$ *and for all subsets* $B \subseteq \mathcal{A}$ *:*

   (*Distributivity*) $$\bigwedge_{b \in B} (a \to b) = a \to \bigwedge_{b \in B} b$$

Remark 3. *In the particular case where* $B = \emptyset$*, the axiom of distributivity states that* $a \to \top = \top$ *for all* $a \in \mathcal{A}$.

3.2 Examples of implicative structures

Complete Heyting algebras The first example of implicative structures is given by complete Heyting algebras. Indeed, the axioms of implicative structures are intuitionistic tautologies verified by any complete Heyting algebra. Therefore, every complete Heyting algebra induces an implicative structure with the same arrow:

Proposition 4. *Every complete Heyting algebra is an implicative structure.*

Proof. Since $\mathcal{H}$ is complete, by definition we have $a \to b = \bigvee\{x \in \mathcal{H} : a \wedge x \preccurlyeq b\}$, from which we deduce that $a \curlywedge c \preccurlyeq b \Leftrightarrow a \preccurlyeq c \to b$. The axioms defining implicative structures are straightforward to prove using these observations.

The converse is obviously false, since the implication of an implicative structure $\mathcal{A}$ is in general not determined by the lattice structure of $\mathcal{A}$. Besides, since any (complete) Boolean algebra is in particular a (complete) Heyting algebra, *a fortiori* any complete Boolean algebra induces an implicative structure:

Proposition 5. *If $\mathcal{B}$ is a complete Boolean algebra, then $\mathcal{B}$ induces an implicative structure where the implication is defined for all $a, b \in \mathcal{B}$ by $a \to b \triangleq \neg a \curlyvee b$.*

Dummy structures Given a complete lattice $\mathcal{L}$, it is easy to check that the following definitions induce dummy implicative structures:

Proposition 6. *If $\mathcal{L}$ is a complete lattice, the following definitions give rise to implicative structures: 1. $a \to b \triangleq \top$ 2. $a \to b \triangleq b$ (for all $a, b \in \mathcal{L}$)*

Both definitions lead to implicative structures which are meaningless from the point of view of logic. Nonetheless, they will provide us with useful counter-examples.

Ordered combinatory algebras We recall the notion of *ordered combinatory algebra*, abbreviated in OCA, which is a variant[7] of Hofstra and Van Oosten's notion of ordered partial combinatory algebras [12]. Ferrer *et al.* structures to represent Krivine realizability, called ${}^{\mathcal{I}}$OCA or ${}^{\mathcal{K}}$OCA, are particular cases of OCA [6,7,5].

Definition 7. (OCA) *An* ordered combinatory algebra *is given by a quintuple $(\mathcal{A}, \leq, \mathtt{app}, \boldsymbol{k}, \boldsymbol{s})$, where:*

- *$\leq$ is a partial order over $\mathcal{A}$,*
- *$\mathtt{app} : (a, b) \mapsto ab$ is a monotonic function[8] from $\mathcal{A} \times \mathcal{A}$ to $\mathcal{A}$,*
- *$\boldsymbol{k} \in \mathcal{A}$ is such that $\boldsymbol{k}ab \leq a$ for all $a, b \in \mathcal{A}$,*
- *$\boldsymbol{s} \in \mathcal{A}$ is such that $\boldsymbol{s}abc \leq ac(bc)$ for all $a, b, c \in \mathcal{A}$.*

[7] In partial combinatory algebras, the application is defined as a partial function.

[8] Observe that the application, which is written as a product, is neither commutative nor associative in general.

Given any ordered combinatory algebra, we can define an implication on the complete lattice $\mathcal{P}(\mathcal{A})$ which give rise to an implicative structure:

Proposition 8. *If $\mathcal{A}$ is an ordered combinatory algebra, then the complete lattice $\mathcal{P}(\mathcal{A})$ equipped with the implication*[9] *:*

$$A \to B \quad \triangleq \quad \{r \in \mathcal{A} : \forall a \in A. ra \in B\} \qquad (\forall A, B \subseteq \mathcal{A})$$

is an implicative structure.

Proof. Both conditions (variance/distributivity) are trivial from the definition.

Implicative structure of classical realizability Our final example of implicative structure—which is the main motivation of this work—is given by classical realizability. As we saw in Section 2, the construction of classical realizability models, whether it be from Krivine's realizability algebras [14,15,16] in a set-theoretic like fashion or in Streicher's AKS [24], takes place in a structure of the form $(\boldsymbol{\Lambda}, \boldsymbol{\Pi}, \cdot, \perp\!\!\!\perp)$ where $\boldsymbol{\Lambda}$ is the set of realizers; $\boldsymbol{\Pi}$ is the set of stacks; $(\cdot) : \boldsymbol{\Lambda} \times \boldsymbol{\Pi} \to \boldsymbol{\Pi}$ is a binary operation for pushing a realizer onto a stack and $\perp\!\!\!\perp \subseteq \boldsymbol{\Lambda} \times \boldsymbol{\Pi}$ is the pole. Given such a quadruple, we can define for all $a, b \in \mathcal{A}$:

$$\mathcal{A} \triangleq \mathcal{P}(\boldsymbol{\Pi}) \qquad a \preccurlyeq b \triangleq a \supseteq b \qquad a \to b \triangleq a^{\perp\!\!\!\perp} \cdot b = \{t \cdot \pi : t \in a^{\perp\!\!\!\perp}, \pi \in b\}$$

where as usual $a^{\perp\!\!\!\perp}$ is $\{t \in \boldsymbol{\Lambda} : \forall \pi \in a, (t, \pi) \in \perp\!\!\!\perp\} \in \mathcal{P}(\boldsymbol{\Lambda})$, the orthogonal set of $a \in \mathcal{P}(\boldsymbol{\Pi})$ with respect to the pole $\perp\!\!\!\perp$. It is easy to verify that:

Proposition 9. *The triple $(\mathcal{A}, \preccurlyeq, \to)$ is an implicative structure.*

Proof. The proof is again trivial. Variance conditions correspond to the usual monotonicity of truth and falsity values in Krivine realizability [13], while the distributivity follows directly by unfolding the definitions.

4 Interpreting the λ-calculus

4.1 Interpretation of λ-terms

We motivated the definition of implicative structures with the aim of obtaining a common framework for the interpretation both of types and programs. We shall now see how λ-terms can indeed be defined in implicative structures.

From now on, let $\mathcal{A} = (\mathcal{A}, \preccurlyeq, \to)$ denotes an arbitrary implicative structure.

[9] This definition is related with the consttruction of a realizability tripos from an OCA $\mathcal{A}$. Indeed, given a set X, the ordering on predicates of $\mathcal{P}(\mathcal{A})^X$ is defined by:

$$\varphi \vdash_X \psi \quad \triangleq \quad \exists r \in \mathcal{A}. \forall x \in X. \forall a \in \mathcal{A}. (a \in \varphi(x) \Rightarrow ra \in \psi(x))$$

where r is broadly a *realizer* of $\forall x \in X. \varphi(x) \Rightarrow \psi(x)$. See [12] for further details.

Definition 10. (Application) *Given two elements $a, b \in \mathcal{A}$, we call the* application *of a to b and write ab the element of $\mathcal{A}$ that is defined by:*

$$ab \triangleq \bigwedge\!\!\!\!\!\!\curlywedge \{c \in \mathcal{A} : a \preccurlyeq (b \to c)\}.$$

If we think of the order relation $a \preccurlyeq b$ as "*a is more precise than b*", the above definition actually defines the application ab as the meet of all the elements c such that $b \to c$ is an approximation of a. This definition fulfills the usual properties of the λ-calculus:

Proposition 11. (Properties of application) *For all $a, a', b, b', c \in \mathcal{A}$:*

1. *If $a \preccurlyeq a'$ and $b \preccurlyeq b'$, then $ab \preccurlyeq a'b'$* *(Monotonicity)*
2. *$(a \to b)a \preccurlyeq b$* *($\beta$-reduction)*
3. *$a \preccurlyeq (b \to ab)$* *($\eta$-expansion)*
4. *$ab = \min\{c \in A : a \preccurlyeq (b \to c)\}$* *(Minimum)*
5. *$ab \preccurlyeq c \quad \Leftrightarrow \quad a \preccurlyeq (b \to c)$* *(Adjunction)*

Proof. Simple lattice manipulations using the properties of the arrow.

Remark 12. (Galois connection) *The adjunction $ab \preccurlyeq c \Leftrightarrow a \preccurlyeq (b \to c)$ expresses the existence of a family of Galois connections $f_b \dashv g_b$ indexed by all $b \in \mathcal{A}$, where the left and right adjoints $f_b, g_b : \mathcal{A} \to \mathcal{A}$ are defined by:*

$$f_b : a \mapsto ab \qquad \text{and} \qquad g_b : c \mapsto (b \to c) \qquad\qquad (\text{for all } a, b, c \in \mathcal{A})$$

Recall that in a Galois connection, the left adjoint is fully determined by the right one (and vice-versa). In the particular case of a complete Heyting algebra $(\mathcal{H}, \preccurlyeq, \to)$, this implies that the application is characterized by $ab = a \curlywedge b$ for all $a, b \in \mathcal{H}$. Indeed, in any Heyting algebra, the adjunction $a \curlywedge b \preccurlyeq c \Leftrightarrow a \preccurlyeq (b \to c)$ holds for all $a, b, c \in \mathcal{H}$, by uniqueness of the left adjoint, ab and $a \curlywedge b$ are thus equal.

Definition 13. (Abstraction) *Given a function $f : \mathcal{A} \to \mathcal{A}$, we call* abstraction *of f and write λf the element of $\mathcal{A}$ defined by:*

$$\lambda f \triangleq \bigwedge_{a \in \mathcal{A}} (a \to f(a))$$

Once again, if we think of the order relation $a \preccurlyeq b$ as "*a is more precise than b*", the meet of the elements of a set S is an element containing the union of all the informations given by the elements of S. With this in mind, the above definition sets λf as the union of all the step functions $a \to f(a)$. This definition, together with the definition of the application, fulfills again properties expected from the λ-calculus:

Proposition 14. (Properties of the abstraction) *The following holds for any $f, g : \mathcal{A} \to \mathcal{A}$:*

1. *If for all $a \in \mathcal{A}$, $f(a) \preccurlyeq g(a)$, then $\lambda f \preccurlyeq \lambda g$.* *(Monotonicity)*

2. *For all* $a \in \mathcal{A}$, $(\lambda f)a \preccurlyeq f(a)$. *(β-reduction)*
3. *For all* $a \in \mathcal{A}$, $a \preccurlyeq \lambda(x \mapsto ax)$. *(η-expansion)*

Proof. Again, the proof consists in easy lattices manipulations.

We call *λ-term with parameters (in* $\mathcal{A}$*)* any term defined from the following grammar[11]:

$$t, u ::= x \mid a \mid \lambda x.t \mid tu$$

where x is a variable and a is an element of $\mathcal{A}$. We can thus associate to each closed λ-term with parameters t an element $t^{\mathcal{A}}$ of $\mathcal{A}$, defined by induction on the size of t as follows (where $a \in \mathcal{A}$):

$$a^{\mathcal{A}} \triangleq a \qquad (tu)^{\mathcal{A}} \triangleq t^{\mathcal{A}} u^{\mathcal{A}} \qquad (\lambda x.t)^{\mathcal{A}} \triangleq \lambda(a \mapsto (t[a/x])^{\mathcal{A}})$$

Thanks to the properties of the application and of the abstraction in implicative structures that we proved, we can check that the embedding of λ-term is sound with respect to the β-reduction:

Proposition 15. *For all closed* λ*-terms* t *and* u *with parameters, if* $t \longrightarrow_\beta u$*, then* $t^{\mathcal{A}} \preccurlyeq u^{\mathcal{A}}$*.*

Proof. By induction on the reduction $t \longrightarrow_\beta u$ using Propositions 11 and 14.

Again, if we think of the order relation $a \preccurlyeq b$ as "*a is more precise than b*", it makes sense that the β-reduction $t \longrightarrow_\beta u$ is reflected in the ordering $t^{\mathcal{A}} \preccurlyeq u^{\mathcal{A}}$: the result of a computation contains indeed less information than the computation itself[10].

4.2 Adequacy

We now dispose of a structure in which we can interpret types and λ-terms. We saw that the interpretation of terms was intuitively sound with respect to the β-reduction. We shall now prove that the typing rules of System F are adequate with respect to the interpretation of terms, that is to say that if t is a closed λ-term of type T, then $t^{\mathcal{A}} \preccurlyeq T^{\mathcal{A}}$. The last statement can again be understood as the fact that a term (*i.e.* a computation) carries more information than its type, just like a realizer of a formula is more informative about the formula than the formula itself.

Adequacy of the interpretation We shall now sketch the formalization of the former result. First, we extend the usual formulas of System F by defining second-order formulas with parameters as:

$$A, B ::= a \mid X \mid A \Rightarrow B \mid \forall X.A \qquad (a \in \mathcal{A})$$

[10] For instance, 0 contains less information than $15 - (3 \times 5)$ or than $\mathbb{1}_{\mathbb{Q}}(\sqrt{(2)})$.

We can then embed closed formulas with parameters into the implicative structure $\mathcal{A}$. The embedding is trivially defined by:

$$a^{\mathcal{A}} \triangleq a \qquad (A \Rightarrow B)^{\mathcal{A}} \triangleq A^{\mathcal{A}} \to B^{\mathcal{A}} \qquad (\forall X.A)^{\mathcal{A}} \triangleq \bigwedge_{a \in \mathcal{A}} (A\{X := a\})^{\mathcal{A}}$$

where $a \in \mathcal{A}$. We define a type system for the λ_c-calculus with parameters[11] (that is λ-terms with parameter plus an instruction $\boldsymbol{cc}$). Typing contexts are defined as usual by finite lists of hypotheses of the shape $(x : A)$ where x is a variable and A a formula with parameters. The inference rules, given in Figure 1, are the same as in System F (with the extended syntaxes of terms and formulas with parameters), plus the additional rules for *cc*.

In order to prove the adequacy of the type system with respect to the embedding, we define substitutions, which we write σ, as functions mapping variables (of terms and types) to element of $\mathcal{A}$:

$$\sigma ::= \varepsilon \mid \sigma[x \mapsto a] \mid \sigma[X \mapsto a] \qquad (a \in \mathcal{A},\ x, X \text{ variables})$$

In the spirit of the proof of adequacy in classical realizability, we say that a substitution σ realizes a typing context Γ, which we write $\sigma \Vdash \Gamma$, if for all bindings $(x : A) \in \Gamma$ we have $\sigma(x) \preccurlyeq (A[\sigma])^{\mathcal{A}}$.

Theorem 16. (Adequacy) *The typing rules of Figure 1 are adequate with respect to the interpretation of terms and formulas: if t is a λ_c-term with parameters, A a formula with parameters and Γ a typing context such that $\Gamma \vdash t : A$ then for all substitutions $\sigma \Vdash \Gamma$, we have $(t[\sigma])^{\mathcal{A}} \preccurlyeq (A[\sigma])^{\mathcal{A}}$.*

Proof. The proof resembles the usual proof of adequacy in classical realizability (see [13,19]), namely by induction on typing derivations.

Corollary 17. *For all λ-terms t, if $\vdash t : A$, then $t^{\mathcal{A}} \preccurlyeq A^{\mathcal{A}}$.*

4.3 Combinators

The previous results indicate that any closed λ-term is, through the interpretation, lower than the interpretation of its principal type. We give here some examples of closed λ-terms which are in fact equal to their principal types through the interpretation in $\mathcal{A}$. Let us now consider the following combinators:

$$\mathbf{i} \triangleq \lambda x.x \qquad \mathbf{k} \triangleq \lambda xy.x \qquad \mathbf{s} \triangleq \lambda xyz.xz(yz) \qquad \mathbf{w} \triangleq \lambda xy.xyy$$

[11] In practice, we use Charguéraud's locally nameless representation [2] for terms and formulas. Without giving too much details, we actually define pre-terms and pre-types which allow both for names (for free variables) and De Bruijn indices (for bounded variables). Terms and types are then defined as pre-terms and pre-types without free De Bruijn indices. As for the embedding from pre-terms (resp. pre-types) into an implicative structure, we define them by means of inductive predicates: `Inductive translated : trm →X → Prop := ...` for which we proved the expected properties.

$$\frac{(x:A)\in\Gamma}{\Gamma\vdash x:A} \qquad \frac{\Gamma,x:A\vdash t:B}{\Gamma\vdash \lambda x\,.\,t:A\to B} \qquad \frac{\Gamma\vdash t:A\to B \qquad \Gamma\vdash t:A}{\Gamma\vdash tu:B}$$

$$\frac{\Gamma\vdash t:A}{\Gamma\vdash t:\forall X.A}\,(X\notin FV(\Gamma)) \qquad \frac{\Gamma\vdash t:\forall X.A}{\Gamma\vdash t:A\{X:=B\}} \qquad \frac{}{\Gamma\vdash \mathbf{cc}:((A\to B)\to A)\to A}$$

Fig. 1. Second-order type system for the λ_c-calculus

It is well-known that these combinators can be given the following polymorphic types:

$\mathbf{i} : \forall X.X \Rightarrow X$	$\mathbf{s} : \forall XYZ.(X \Rightarrow Y \Rightarrow Z) \Rightarrow (X \Rightarrow Y) \Rightarrow X \Rightarrow Z$
$\mathbf{k} : \forall XY.X \Rightarrow Y \Rightarrow X$	$\mathbf{w} : \forall XY.(X \Rightarrow X \Rightarrow Y) \Rightarrow X \Rightarrow Y$

Through the interpretation these combinators are identified with their types:

Proposition 18. *The following equalities hold in any implicative structure $\mathcal{A}$:*

1. $\mathbf{i}^{\mathcal{A}} = \bigwedge_{a\in\mathcal{A}}(a\to a)$
2. $\mathbf{k}^{\mathcal{A}} = \bigwedge_{a,b\in\mathcal{A}}(a\Rightarrow b\Rightarrow a)$
3. $\mathbf{s}^{\mathcal{A}} = \bigwedge_{a,b,c\in\mathcal{A}}((a\to b\to c)\to(a\to b)\to a\to c)$
4. $\mathbf{w}^{\mathcal{A}} = \bigwedge_{a,b,c\in\mathcal{A}}((a\to a\to b)\to a\to b)$

Proof. The inequality from left to right are consequences of the adequacy. The converse inequalities are proved by hands, using the properties of application and abstraction in implicative structures (Propositions 11 and 14).

Finally, in the spirit of the previous equalities, we define the interpretation of ***cc*** by the interpretation of its principal type, that is:

$$\mathbf{cc}^{\mathcal{A}} \triangleq \bigwedge_{a,b}(((a\to b)\to a)\to a)$$

Remark 19. *It is not always the case that a term is equal to its principal type. Consider for instance a dummy implicative structure $\mathcal{A}$ where $a\to b=\top$ for all elements $a,b\in\mathcal{A}$. Suppose in addition that $\mathcal{A}$ has at least two distinct elements, so that $\bot\neq\top$. Then the following holds:*

1. *For any $a,b\in\mathcal{A}$, we have $ab=\bigwedge\{c : a\preccurlyeq b\to c\}=\bigwedge\mathcal{A}=\bot$.*
2. *For any $f:\mathcal{A}\to\mathcal{A}$, we have $\lambda f=\bigwedge_{a\in\mathcal{A}}(a\to f(a))=\bigwedge_{a\in\mathcal{A}}\top=\top$.*
3. *$\mathbf{ii}:\forall X.X\to X$, yet $(\mathbf{ii})^{\mathcal{A}}=\bot\neq\top=(\forall X.X\to X)^{\mathcal{A}}$.*
4. *$\mathbf{i}^{\mathcal{A}}=\top\neq\bot=(\mathbf{skk})^{\mathcal{A}}$.*

4.4 The problem of consistency

The last remark shows us that not all implicative structures are suitable for interpreting intuitionistic or classical logic. We thus need to introduce a criterion of consistency.

Definition 20. (Consistency) *We say that an implicative structure is:*

- intuitionistically consistent *if* $t^{\mathcal{A}} \neq \bot$ *for all closed* λ*-terms;*
- classically consistent *if* $t^{\mathcal{A}} \neq \bot$ *for all closed* λ_c*-terms.*

We shall now relate the previous definition to the usual definition of consistency in classical realizability models. Recall that any abstract Krivine structure $\mathcal{K} = (\boldsymbol{\Lambda}, \boldsymbol{\Pi}, \mathtt{app}, \mathtt{push}, \mathtt{k}_{\_}, \boldsymbol{k}, \boldsymbol{s}, \boldsymbol{cc}, \mathbf{PL}, {}^{\perp\!\!\perp})$ induces an implicative structure $(\mathcal{A}, \preccurlyeq, \to)$ where $\mathcal{A} = \mathcal{P}(\boldsymbol{\Pi})$, $a \preccurlyeq b \Leftrightarrow a \supseteq b$ and $a \to b = a^{\perp\!\!\perp} \cdot b$. A realizability model is said to be consistent when there is no proof-like term realizing $\bot$. In terms of abstract Krivine structures, the consistency can then be expressed by this simple criterion:

$$\mathcal{K} \text{ is consistent} \quad \text{if and only if} \quad \{\bot\}^{\perp\!\!\perp} \cap \mathbf{PL} = \boldsymbol{\Pi}^{\perp\!\!\perp} \cap \mathbf{PL} = \emptyset$$

We thus need to check that this criterion of consistency for the AKS implies the consistency of the induced implicative structure, *i.e.* that if t is a closed λ_c-term, then $t^{\mathcal{A}} \neq \bot$. By definition of the implicative structure $\mathcal{A}$ induced by $\mathcal{K}$, we have that $t^{\mathcal{A}} \in \mathcal{A} = \mathcal{P}(\boldsymbol{\Pi})$. Therefore, $t^{\mathcal{A}}$ is a falsity value from the point of view of the AKS. To ensure that it is not equal to $\bot$ (*i.e.* $\boldsymbol{\Pi}$), it is enough to find a realizer of $t^{\mathcal{A}}$ in $\mathcal{K}$. The consistency of $\mathcal{K}$ precisely states that $\bot$ does not have any realizer.

Our strategy to find a realizer for $t^{\mathcal{A}}$ in $\mathcal{K}$ is to use t itself. First, we reduce the problem to the set of terms that are identifiable with the combinatory terms of $\mathcal{K}$. We call a *combinatory term* any term that is obtained by combination of the combinators $(\mathbf{k}, \mathbf{s}, \mathbf{cc})$. To each combinatory term t we associate a term $t^{\boldsymbol{\Lambda}}$ in $\boldsymbol{\Lambda}$, whose definition by induction is trivial:

$$\mathbf{k}^{\boldsymbol{\Lambda}} \triangleq \boldsymbol{k} \qquad \mathbf{s}^{\boldsymbol{\Lambda}} \triangleq \boldsymbol{s} \qquad \mathbf{cc}^{\boldsymbol{\Lambda}} \triangleq \boldsymbol{cc} \qquad (tu)^{\boldsymbol{\Lambda}} \triangleq \mathtt{app}(t^{\boldsymbol{\Lambda}}, u^{\boldsymbol{\Lambda}})$$

Since the set $\mathbf{PL}$ is closed under application, for any combinatory term t, its interpretation $t^{\boldsymbol{\Lambda}}$ is in $\mathbf{PL}$. The combinatory completeness of $(\mathbf{k}, \mathbf{s}, \mathbf{cc})$ with respect to closed λ_c-terms ensures us that there exists a combinatory term t_0 (viewed as a λ-term) such that $t_0 \longrightarrow_\beta t$. By Proposition 15, we thus have $t_0^{\mathcal{A}} \preccurlyeq t^{\mathcal{A}}$. It is thus enough to show that $t_0^{\mathcal{A}} \neq \bot$: we reduced the original problem for closed λ_c-terms to combinatory terms.

It only remains to show that for any combinatory term t_0, its interpretation $t_0^{\mathcal{A}}$ is not $\bot$. For the reasons detailed above, it is sufficient to prove that $t_0^{\mathcal{A}}$ is realized. We prove that $t_0^{\mathcal{A}}$ is in fact realized by $t_0^{\boldsymbol{\Lambda}}$:

Lemma 21. *For any combinatory term* t*,* $t^{\boldsymbol{\Lambda}}$ *realizes* $t^{\mathcal{A}}$*,* i.e. $t^{\boldsymbol{\Lambda}} \Vdash t^{\mathcal{A}}$

Proof. By induction on the structure of t, by combining usual results of classical realizability and properties of implicative structures.

We can thus conclude that the consistency of $\mathcal{K}$ induces the one (in the sense of Definition 20) of the associated implicative structure:

Proposition 22. *If* $\mathcal{K}$ *is a consistent abstract Krivine structure, then the implicative structure it induces is classically consistent.*

Proof. Let t be any closed λ_c-term. We want to show that $t^{\mathcal{A}} \neq \bot = \boldsymbol{\Pi}$. We show that $t^{\mathcal{A}}$, which belongs to $\mathcal{P}(\boldsymbol{\Pi})$ is realized by a proof-like term.

It is worth noting that the criterion of consistency is defined with respect to the set **PL** together with the pole. These sets are already at the heart of the definition of Krivine's realizability models, where valid formulas are precisely the formulas realized by a proof-like term. We shall then introduce the corresponding ingredient for implicative structures.

5 Implicative algebras

5.1 Separation

Definition 23. (Separator) *Let $(\mathcal{A}, \preccurlyeq, \to)$ 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:*

1. $\mathbf{k}^{\mathcal{A}} \in \mathcal{S}$, *and* $\mathbf{s}^{\mathcal{A}} \in \mathcal{S}$. *(Combinators)*
2. *If $a \in \mathcal{S}$ and $a \preccurlyeq b$, then $b \in \mathcal{S}$.* *(Upwards closure)*
3. *If $(a \to b) \in \mathcal{S}$ and $a \in \mathcal{S}$, then $b \in \mathcal{S}$.* *(Closure under modus ponens)*

A separator $\mathcal{S}$ is said to be classical *if $\mathbf{cc}^{\mathcal{A}} \in \mathcal{S}$ and* consistent *if $\bot \notin \mathcal{S}$.*

Remark 24. (Alternative definition) *In presence of condition (2), it is easy to show that condition (3) is equivalent to the following condition:*

(3') If $a \in \mathcal{S}$ and $b \in \mathcal{S}$ then $ab \in \mathcal{S}$. *(Closure under application)*

Intuitively, thinking of elements of an implicative structure as truth values, a separator should be understood as the set which distinguishes the valid formulas. Considering the elements as terms, it should rather be viewed as the set of valid realizers. Indeed, conditions (1) and (3') ensure that all closed λ-terms are in any separator. Reading $a \preccurlyeq b$ as "*the formula a is a subtype of the formula b*", condition (2) ensures the validity of semantic subtyping. Thinking of the ordering as "*a is a realizer of the formula b*", condition (2) states that if a formula is realized, then it is in the separator.

Definition 25. (Implicative algebra) *We call* implicative algebra *any quadruple $(\mathcal{A}, \preccurlyeq, \to, \mathcal{S})$ where $(\mathcal{A}, \preccurlyeq, \to)$ is an implicative structure and $\mathcal{S}$ is a separator over $\mathcal{A}$. We say that an implicative algebra is* classical *if its separator is.*

Example 26. (Complete Boolean algebras) *It is easy to verify that for any complete Boolean algebra $\mathcal{B}$, combinators are interpreted by the maximal element in the induced implicative structure: $\mathbf{k}^{\mathcal{B}} = \mathbf{s}^{\mathcal{B}} = \mathbf{cc}^{\mathcal{B}} = \top$. Therefore, the singleton $\{\top\}$ is a classical separator for the induced implicative structure. Any non-degenerated complete Boolean algebras thus induces a classically consistent implicative algebra.*

Example 27. (Abstract Krivine structure) *Recall that any* AKS *induces an implicative structure $(\mathcal{A}, \preccurlyeq, \to)$ where $\mathcal{A} = \mathcal{P}(\boldsymbol{\Pi})$, $a \preccurlyeq b \Leftrightarrow a \supseteq b$ and $a \to b = a^{\perp\!\!\!\perp} \cdot b$. The set of realized formulas, namely $\mathcal{S} = \{a \in \mathcal{A} : a^{\perp\!\!\!\perp} \cap \mathbf{PL} \neq \emptyset\}$, defines a valid separator.*

5.2 λ_c-terms

The first property that we shall state about classical separators is that they contain the interpretation of all closed λ_c-terms. This follows again from the combinatory completeness of the basis $(\mathbf{k}, \mathbf{s}, \mathbf{cc})$ for the λ_c-calculus[12]. Indeed, if $\mathcal{S}$ is a classical separator over an implicative structure $(\mathcal{A}, \preccurlyeq, \to)$, it is clear that any combinatory term is in the separator. Again, by combinatory completeness, if t is a closed λ_c-term, there exists a combinatory term t_0 such that $t_0 \longrightarrow_\beta t$, and therefore $t_0^{\mathcal{A}} \preccurlyeq t^{\mathcal{A}}$ (by Proposition 15). By upward closure of separators, we deduce that:

Proposition 28. *If $(\mathcal{A}, \preccurlyeq, \to, \mathcal{S})$ is a (classical) implicative algebra and t is a closed λ-term (resp. λ_c-term), then $t^{\mathcal{A}} \in \mathcal{S}$.*

From the previous proposition and the adequacy of second-order typing rules for the λ_c-calculus (Theorem 16), we obtain that:

Corollary 29. *If $(\mathcal{A}, \preccurlyeq, \to, \mathcal{S})$ is a (classical) implicative algebra, t is a closed λ-term (resp. λ_c-term) and A is a formula such that $\vdash t : A$, then $A^{\mathcal{A}} \in \mathcal{S}$.*

Remark 30. *The latter corollary provides us with a methodology for proving that an element of a given implicative algebra is in the separator. In the spirit of realizability, where the standard methodology to prove that a formula is realized consists in using typed terms and adequacy as much as possible, we can use typed terms to prove automatically that the corresponding formulas belongs to the separator. We shall use this methodology abundantly in the sequel. In the Coq development, this corresponds to a tactic called* `realizer` *which allows us to prove that an element belongs to the separator simply by furnishing a realizer:*

```
Lemma composition: ∀ a b c, (a ↦ b) ↦ (b ↦ c) ↦ a ↦ c ∈ S
Proof. intros. realizer ((λ⁺ λ⁺ λ⁺([$1] ([$2] $0)))). Qed. (** λxyz.y(xz) *)
```

5.3 Internal logic

In order to be able to define triposes from implicative algebras, the first step is to equip them with a structure of Heyting algebra. To this end, we begin with defining an entailment relation in the spirit of filtered OCAs [12]. We then define quantifiers and connectives as usual in classical realizability (see [13]), and we verify that they satisfy the usual logical rules. In the rest of this section, we work within a fixed implicative algebra $(\mathcal{A}, \preccurlyeq, \to, \mathcal{S})$.

Definition 31. (Entailment) *For all $a, b \in \mathcal{A}$, we say that a* entails *b and write $a \vdash_{\mathcal{S}} b$ if $a \to b \in \mathcal{S}$. We say that a and b are equivalent and write $a \cong_{\mathcal{S}} b$ if $a \vdash_{\mathcal{S}} b$ and $b \vdash_{\mathcal{S}} a$.*

Proposition 32. (Properties of $\vdash_{\mathcal{S}}$) *For any $a, b, c \in \mathcal{A}$, the following holds:*

1. *$a \vdash_{\mathcal{S}} a$* *(Reflexivity)*

[12] In order to avoid the certification of the corresponding compilation function, we state this well-known fact as an axiom (the only one) in our development.

2. *If* $a \vdash_{\mathcal{S}} b$ *and* $b \vdash_{\mathcal{S}} c$ *then* $a \vdash_{\mathcal{S}} c$. *(Transitivity)*
3. *If* $a \preccurlyeq b$ *then* $a \vdash_{\mathcal{S}} b$. *(Subtyping)*
4. *If* $a \cong_{\mathcal{S}} b$, *then* $a \in \mathcal{S}$ *if and only if* $b \in \mathcal{S}$. *(Closure under* $\cong_{\mathcal{S}}$*)*
5. *If* $a \vdash_{\mathcal{S}} b \to c$ *then* $a \curlywedge b \vdash_{\mathcal{S}} c$. *(Half-adjunction property)*
6. $\bot \vdash_{\mathcal{S}} a$ *(Ex falso quod libet)*
7. $a \vdash_{\mathcal{S}} \top$ *(Maximal element)*

Proof. Straightforward from the definitions, using $\lambda xyz.y(xz)$ to realize the second item and $\lambda xy.xyy$ to realize the fifth.

Besides, the entailment relation behaves like Heyting's arrow with respect to the preorder relation $\vdash_{\mathcal{S}}$ in terms of monotonicity:

Proposition 33. (Compatibility with $\to$) *For all* $a, a', b, b' \in \mathcal{A}$, *we have:*

1. *If* $b \vdash b'$ *then* $a \to b \vdash a \to b'$.
2. *If* $a \vdash a'$ *then* $a' \to b \vdash a \to b$.

Proof. Direct using $\lambda xyz.x(yz)$ and $\lambda xyz.y(xz)$ as realizers.

Negation We define the negation by $\neg a \triangleq a \to \bot$. If the separator is classical, we can prove that for any $a \in \mathcal{A}$, we have:

Proposition 34. (Double negation) *If* $\mathcal{S}$ *is a classical separator, for any* $a \in \mathcal{A}$ *we have:* 1. $a \vdash_{\mathcal{S}} \neg\neg a$ 2. $\neg\neg a \vdash_{\mathcal{S}} a$

Proof. The first item is realized by $\lambda xk.kx$, while the second follows from the inequality $((a \to \bot) \to a) \to a \preccurlyeq ((a \to \bot) \to \bot) \to a$, whose left member is realized by **cc**.

Quantifiers Following the usual definitions in classical realizability (see [13,19]), the universal quantification of a family of truth values is naturally defined as its meet while the existential quantification is defined through a negative encoding:

$$\bigvee_{i \in I} a_i \triangleq \bigwedge_{i \in I} a_i \qquad \exists_{i \in I} a_i \triangleq \bigwedge_{c \in A} \Big(\bigwedge_{i \in I} (a_i \to c) \Big) \to c$$

While it could have seemed more natural to define existential quantifiers through joins, we should recall that the arrow does not commute with joins in general[13]. It is clear that these definitions are compatible with the expected semantic rules:

Proposition 35. (Universal quantifier) *The following semantic typing rules are valid in any implicative structures:*

$$\frac{\Gamma \vdash t : a_i \quad \text{for all } i \in I}{\Gamma \vdash t : \bigvee_{i \in I} a_i} \qquad \frac{\Gamma \vdash t : \bigvee_{i \in I} a_i \quad i_0 \in I}{\Gamma \vdash t : a_{i_0}}$$

[13] When it does, the realizability tripos actually collapses to a forcing tripos, see [18,19].

Proposition 36. (Existential quantifier) *The following semantic typing rules are valid in any implicative structures:*

$$\frac{\Gamma \vdash t : a_{i_0} \quad i_0 \in I}{\Gamma \vdash \lambda x.xt : \exists_{i\in I}\, a_i} \qquad \frac{\Gamma \vdash t : \exists_{i\in I}\, a_i \quad \Gamma, x : a_i \vdash u : c \quad (\textit{for all } i \in I)}{\Gamma \vdash t(\lambda x.u) : c}$$

Proof. Straightforward using the adjunction of the application (Proposition 11).

Sum and product We define it by the usual encodings in System F:

$$a \times b \triangleq \bigwedge_{c\in\mathcal{A}} ((a \to b \to c) \to c) \qquad a + b \triangleq \bigwedge_{c\in\mathcal{A}} ((a \to c) \to (b \to c) \to c)$$

Recall that the pair $\langle a, b\rangle$ is encoded by the λ-term $\lambda x.xab$, while first and second projections are respectively defined by $\pi_1 \triangleq \lambda xy.x$ and $\pi_2 \triangleq \lambda xy.y$. We can check that the expected semantic typing rules are valid

Proposition 37. (Product) *The following semantic typing rules are valid:*

$$\frac{\Gamma \vdash t : a \quad \Gamma \vdash u : b}{\Gamma \vdash \lambda z.ztu : a \times b} \qquad \frac{\Gamma \vdash t : a \times b}{\Gamma \vdash t\pi_1 : a} \qquad \frac{\Gamma \vdash t : a \times b}{\Gamma \vdash t\pi_2 : b}$$

Proposition 38. (Sum) *The following semantic typing rules are valid:*

$$\frac{\Gamma \vdash t : a}{\Gamma \vdash \lambda lr.lt : a + b} \qquad \frac{\Gamma \vdash t : b}{\Gamma \vdash \lambda lr.rt : a + b} \qquad \frac{\Gamma \vdash t : a_1 + a_2 \quad \Gamma, x_i : a_i \vdash u_i : c}{\Gamma \vdash t(\lambda x_1.u_1)(\lambda x_2.u_2) : c}$$

Proof. Straightforward lattices manipulations, similar to the proof for the existential quantifier.

The natural candidate to computationally represents a "meet" of a and b is the product type $a \times b$. We can verify that it satisfies the expected property (in Heyting algebras) with respect to the arrow:

Proposition 39. (Adjunction) *For any $a, b, c \in \mathcal{A}$, we have $a \vdash_{\mathcal{S}} b \to c$ if and only if $a \times b \vdash_{\mathcal{S}} c$.*

Proof. Both directions are proved using the expected realizer and subtyping: from left to right, we use $\lambda xy.yx$ to realize $(a \to b \to c) \to a \times b \to c$; from right to left, we realize $(a \times b \to c) \to a \to b \to c$ with $\lambda pxy.p(\lambda z.zxy)$.

5.4 Implicative tripos

It is clear from the properties of implicative algebras presented in the last sections that the entailment relation together with the sum and products induce a structure of Heyting prealgebra (indeed, the entailment relation only defines a preorder). By considering the quotient $\mathcal{A}/_{\cong_{\mathcal{S}}}$ of the former Heyting prealgebra by the relation $\cong_{\mathcal{S}}$, and lifting the previous definitions of connectives and quantifiers

to equivalence classes, we thus obtain a Heyting algebra[14]. This construction is actually the main step towards the definition of the implicative tripos [18,19], which allows us to recover the usual categorical interpretation of realizability models. In particular, it provides us with a framework in which simple criteria allows us to compare classical realizability and forcing models.

6 Conclusion & future work

6.1 Conclusion

We presented in this paper Miquel's concept of *implicative algebra* [18], that relies on the primitive notion of implicative structure. These structures are defined as a particular class of meet-complete lattices equipped with an arrow, where this arrow satisfies commutations with arbitrary meets which are the counterpart of the logical commutation between the universal quantification and the implication. We showed that implicative algebras are a generalization Streicher's AKSs [24] and Ferrer *et al.*'s $^{\kappa}$OCAs [6,7]. Besides, they provide us with a framework in which both λ_c-terms and their types can be interpreted. This has the nice consequence that we really consider the elements of the implicative structure as λ_c-terms and that we can compute with truth values. Through the formalization, this is reflected by a tactic allowing us to prove that elements belong to the separator simply by furnishing realizers.

6.2 Future work

For future work, it would be interesting to push the formalization further to be able to represent implicative triposes. However, this poses the challenge of manipulating quotients and equivalent classes. The safe definition of quotients within CIC (and thus Coq) is indeed a tricky question [11,3,4], and as for now, we do not know which solution (reasoning modulo setoids, quotient as types classes, etc.) would be the more adapted to our situation.

In a more theoretic perspective, implicative algebras take position on a presentation of logic through universal quantification and the implication. The computational counterpart of this choice is that the presentation relies on the call-by-name λ_c-calculus. This raises the question of knowing whether it is possible to have alternative presentations with similar structures based on different connectives (and thus different calculi). We partially undertook this investigation in [19] by studying different presentations based on disjunctive and conjunctive connectives and related Munch-Maccagnoni's system L [20]. Yet, the equivalence between all presentations still remains to prove.

Acknowledgments The author wishes to thank Assia Mahboubi for pushing him to write the current paper.

[14] If the implicative algebra is classical, for all $a \in \mathcal{A}$ we saw that $\neg\neg a \cong_{\mathcal{S}} a$. Through the same quotient, this implies that $\neg\neg[a] = [a]$ for all $a \in \mathcal{A}$, and that the induced Heyting algebra is actually a Boolean algebra.

References

1. The poplmark challenge, https://www.seas.upenn.edu/~plclub/poplmark/
2. Charguéraud, A.: The locally nameless representation. Journal of Automated Reasoning **49**(3), 363–408 (Oct 2012). https://doi.org/10.1007/s10817-011-9225-2
3. Chicli, L., Pottier, L., Simpson, C.: Mathematical Quotients and Quotient Types in Coq. In: Geuvers, H., Wiedijk, F. (eds.) Types for Proofs and Programs. pp. 95–107. Springer Berlin Heidelberg, Berlin, Heidelberg (2003). https://doi.org/10.1007/3-540-39185-1_6
4. Cohen, C.: Types quotients en coq. In: Hermann (ed.) Actes des 21ème journées francophones des langages applicatifs (JFLA 2010). INRIA, Vieux-Port La Ciotat, France (2010), http://jfla.inria.fr/2010/actes/PDF/cyrilcohen.pdf
5. Ferrer, W., Malherbe, O.: The category of implicative algebras and realizability. ArXiv e-prints (Dec 2017), https://arxiv.org/abs/1712.06043
6. Ferrer Santos, W., Guillermo, M., Malherbe, O.: Realizability in OCAs and AKSs. ArXiv e-prints (2015), https://arxiv.org/abs/1512.07879
7. Ferrer Santos, W., Frey, J., Guillermo, M., Malherbe, O., Miquel, A.: Ordered combinatory algebras and realizability. Mathematical Structures in Computer Science **27**(3), 428–458 (2017). https://doi.org/10.1017/S0960129515000432
8. Frey, J.: Realizability Toposes from Specifications. In: Altenkirch, T. (ed.) 13th International Conference on Typed Lambda Calculi and Applications (TLCA 2015). Leibniz International Proceedings in Informatics (LIPIcs), vol. 38, pp. 196–210. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2015). https://doi.org/10.4230/LIPIcs.TLCA.2015.196
9. Frey, J.: Classical realizability in the cps target language. Electronic Notes in Theoretical Computer Science **325**(Supplement C), 111 – 126 (2016). https://doi.org/10.1016/j.entcs.2016.09.034, the Thirty-second Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXII)
10. Griffin, T.G.: A formulae-as-type notion of control. In: Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 47–58. POPL '90, ACM, New York, NY, USA (1990). https://doi.org/10.1145/96709.96714
11. Hofmann, M.: Extensional Concepts in Intensional Type Theory. Ph.D. thesis, University of Edinburgh (1995)
12. Hofstra, P., Van Oosten, J.: Ordered partial combinatory algebras. Mathematical Proceedings of the Cambridge Philosophical Society **134**(3), 445–463 (2003). https://doi.org/10.1017/S0305004102006424
13. Krivine, J.L.: Realizability in classical logic. In Interactive models of computation and program behaviour. Panoramas et synthèses **27** (2009)
14. Krivine, J.L.: Realizability algebras: a program to well order r. Logical Methods in Computer Science **7**(3) (2011). https://doi.org/10.2168/LMCS-7(3:2)2011
15. Krivine, J.L.: Realizability algebras II : new models of ZF + DC. Logical Methods in Computer Science **8**(1), 10 (Feb 2012). https://doi.org/10.2168/LMCS-8(1:10)2012, 28 p.
16. Krivine, J.L.: Quelques propriétés des modèles de réalisabilité de ZF (Feb 2014), http://hal.archives-ouvertes.fr/hal-00940254
17. Miquel, A.: Existential witness extraction in classical realizability and via a negative translation. Logical Methods in Computer Science **7**(2), 188–202 (2011). https://doi.org/10.2168/LMCS-7(2:2)2011

18. Miquel, A.: Implicative algebras: a new foundation for realizability and forcing. ArXiv e-prints (2018), https://arxiv.org/abs/1802.00528
19. Miquey, É.: Classical realizability and side-effects. Theses, Univ.é Paris Diderot ; Univ. de la República, Uruguay (Nov 2017), https://hal.inria.fr/tel-01653733
20. Munch-Maccagnoni, G.: Focalisation and Classical Realisability. In: Grädel, E., Kahle, R. (eds.) Computer Science Logic '09. Lecture Notes in Computer Science, vol. 5771, pp. 409–423. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04027-6_30
21. Pitts, A.M.: Tripos theory in retrospect. Mathematical Structures in Computer Science **12**(3), 265–279 (2002). https://doi.org/10.1017/S096012950200364X
22. Ruyer, F.: Proofs, Types and Subtypes. Ph.D. thesis, Université de Savoie (Nov 2006), https://tel.archives-ouvertes.fr/tel-00140046
23. Sozeau, M., Oury, N.: First-class type classes. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) Theorem Proving in Higher Order Logics. pp. 278–293. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71067-7_23
24. Streicher, T.: Krivine's classical realisability from a categorical perspective. Mathematical Structures in Computer Science **23**(6), 1234–1256 (2013). https://doi.org/10.1017/S0960129512000989