Do CPS translations also translate realizers?

Samuel Gardelle, Étienne Miquey

Do CPS translations also translate realizers?

Samuel Gardelle[1] and Étienne Miquey[2]

[1] ÉNS de Lyon, France
samuel.gardelle@ens-lyon.fr
[2] Aix-Marseille Université, CNRS, I2M, Marseille, France
etienne.miquey@univ-amu.fr

Abstract

In the realm of the proofs-as-programs correspondence, continuation-passing style (CPS) translations are known to be twofold: they bring both a program translation and a logical translation. In particular, when using the former to compile a language with a control operator, the latter ensures the soundness of the compilation with respect to types.

This work is inspired by [OS08], in which Oliva and Streicher explained how Krivine realizability could be rephrased as the composition of a CPS and an intuitionistic realizability model. In this paper, we propose to push one step forward the analysis of the relation between realizability models and CPS translations to investigate the following question: assume that two realizability models are defined using the source and the destination of a CPS translation, is it the case that the CPS translates realizers of a given formula into realizers of the translated formulas?

1 Introduction

Continuation-passing style translations, which where first introduced by Sussman and Steel [SS75], constitute a great tool when it comes to studying operational semantics of calculi: by making explicit the order in which reduction steps are computed, CPS translations indirectly specify an evaluation strategy for the translated calculus. In particular, continuation-passing style translations have a lot of applications in terms of compilation and have been widely studied for call-by-name and call-by-value strategies of the λ-calculus [Plo75, App92, SF93].

From a logical perspective, CPS translations are also very informative insofar as they induce a translation at the level of types that mostly amounts to a syntactical model allowing to transfer logical properties (coherence, normalization) from the target calculus [BPT17]. For instance, standard CPS translations are known to correspond to embeddings of classical logic into intuitionistic logic through variants of Gödel's negative translation [Gri90, Mur90]. Computationally, the latter corresponds to Griffin's seminal observation that a classical Curry-Howard correspondence can be obtained by extending the λ-calculus with control operators, *e.g.* Scheme's `call/cc`. These operators provides a *direct* handle on continuations (allowing in particular the definition of backtracking programs), as opposed to the *indirect* one provided by CPS translations. Several calculi were born from this idea, amongst which Krivine's λ_c-calculus [Kri04].

Elaborating on this calculus, Krivine developed in the late 90s the theory of classical realizability, which is a complete reformulation of its intuitionistic twin. This theory has shown to be particularly fruitful, both to analyze the computational content of proofs [Kri03, Miq11b, Miq18] or to define new models of classical theories [Kri12, Kri21].

Studying the structure of Krivine's classical realizability, Oliva and Streicher showed how the latter could in fact be viewed as the composition of a CPS with a traditional intuitionistic realizability interpretation [OS08]. This observation unveils a somewhat surprising situation: the very nature of a CPS translation is *syntactical* and as such, it is quite unexpected that this

turns out to be well-behaved with respect to realizability, a *semantic* notion. In fact, taking a closer look at Oliva and Streicher's work [OS08], their (classical) realizers are defined through the computational behavior of their CPS translation. In line with this work, Frey also defines a notion of classical realizability directly within the target of a CPS translation [Fre16]. In both cases, realizers are thus compatible with the CPS *by definition*. In a slightly different setting, Miquel studied the witness extraction mechanism of classical realizability for Σ^0_1 formulas through the CPS translation, but his results only apply to typed terms [Miq11b].

Therefore, none of these works tackle the following question: *do CPS translations also translate realizers?* To phrase it a bit more precisely, let us consider a CPS translation $[\cdot]$ from a (classical) source calculus, say Krivine λ_c-calculus, to an intuitionistic calculus, say the λ-calculus, and let us write $[\![\cdot]\!]$ for the translation on types it induces. Assume besides that these two calculi serve as the underlying language of realizers for two realizability interpretations, say for second-order classical and intuitionistic arithmetic (**PA2** and **HA2**) respectively. Now, if t is a term of type A (in the source), then $[t]$ is of type $[\![A]\!]$ (in the destination); but *is it the case that if t realizes A then $[t]$ realizes $[\![A]\!]$*?

To investigate this question, we will use a very convenient tool to reason about realizability models: *evidenced frames* [CMT21] which, as we will see in Section 2.2, capture the algebraic structure of realizability interpretations. In fact, this work was also an excuse to put this recent notion in practice and to test the companion notion of morphism against a concrete candidate. One can think of an evidenced frames morphism as a functional embedding of a realizability interpretation into another one, and as such, CPS translations provide us with very natural nontrivial candidates. The main question of this paper could indeed be rephrased in these terms: *do CPS translations define evidenced frame morphisms*? As we shall see in Section 3.2, the answer is nuanced in that in general a CPS translation does not induces an evidences frame morphism, but we can nonetheless introduce another evidenced frame, which corresponds to the image of the source calculus through the translation and can serve as the codomain of an evidenced frame morphism.

Outline of the paper We start by giving a standard example of a realizability interpretation for **HA2** based on the λ-calculus in Section 2, which we will later use as the destination of the CPS translation we will consider. We take advantage of this section to recall the definition of evidenced frames, and illustrate how the interpretation for **HA2** naturally induces an evidenced frame $\mathcal{EF}_{\mathrm{HA2}}$. We then introduce (a call-by-value presentation of) Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus in Section 3, together with its CPS translation into the (pure) λ-calculus. Finally, in Section 4 we define an evidenced frame $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$ corresponding to a Krivine realizability interpretation based on the $\lambda\mu\tilde{\mu}$-calculus, and we investigate the CPS translation in terms of evidenced frame morphisms. In particular, we show that in general it does not define a morphism from $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$ to $\mathcal{EF}_{\mathrm{LJ2}}$, but that another evidenced frame $\mathcal{EF}_{\mathsf{fw}}$(which also defines a realizability interpretation for **HA2**) can be introduced for the CPS to define an appropriate morphism $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}} \to \mathcal{EF}_{\mathsf{fw}}$.

Due to the page limit, some proofs are only sketched and others have been omitted, an extended version with complete proofs is accessible at: `https://hal.inria.fr/hal-03910311`.

2 Evidenced frames

Before introducing evidenced frames, we give a first example of a (very standard) realizability interpretation that will serve both as the destination of the CPS translation considered in the

$$\frac{(x:A)\in\Gamma}{\Gamma\vdash x:A}\ (\text{Ax}) \qquad \frac{\Gamma\vdash t:A\to B \quad \Gamma\vdash u:A}{\Gamma\vdash t\,u:B}\ (\to_E) \qquad \frac{\Gamma,x:A\vdash t:B}{\Gamma\vdash \lambda x\,.\,t:A\to B}\ (\to_I)$$

$$\frac{\Gamma\vdash t:A \quad \Gamma\vdash u:B}{\Gamma\vdash (t,u):A\wedge B}\ (\wedge_I) \qquad \frac{\Gamma\vdash t:A\wedge B \quad \Gamma,x:A,y:B\vdash u:C}{\Gamma\vdash \textbf{let}\ (x,y)=t\ \textbf{in}\ u:C}\ (\wedge_E) \qquad \frac{\Gamma\vdash t:A[x:=n]}{\Gamma\vdash t:\exists x.A}\ (\exists^1_I)$$

$$\frac{\Gamma\vdash t:\forall x.A}{\Gamma\vdash t:A[x:=n]}\ (\forall^1_E) \qquad \frac{\Gamma\vdash t:A \quad x\notin FV(\Gamma)}{\Gamma\vdash t:\forall x.A}\ (\forall^1_I) \qquad \frac{\Gamma\vdash t:A[X(x_1,\dots,x_n):=B]}{\Gamma\vdash t:\exists X.A}\ (\exists^2_I)$$

$$\frac{\Gamma\vdash t:\forall X.A}{\Gamma\vdash t:A[X(x_1,\dots,x_n):=B]}\ (\forall^2_E) \qquad \frac{\Gamma\vdash t:A \quad X\notin FV(\Gamma)}{\Gamma\vdash t:\forall X.A}\ (\forall^2_I) \qquad \frac{\Gamma\vdash t:A' \quad A\cong A'}{\Gamma\vdash t:A}\ (\cong)$$

Figure 1: A type system for **HA2**

sequel and as an introducing example for evidenced frames.

2.1 Realizability interpretation of HA2

2.1.1 Heyting second-order arithmetic

We start by introducing the terms and formulas of Heyting second-order arithmetic, for which we mostly follow Miquel's presentation [Miq11a]. Second-order formulas are built on top of first-order arithmetical expressions, by means of logical connectives, first- and second-order quantifications and primitive predicates. We use upper case letters for second-order variables and lower case letters for first-order ones.

We consider the usual λ-calculus terms extended with (positive) pairs and the corresponding destructor (written $\lambda(x,y).t$). In the last sections of the paper, we will also include primitives for booleans for technical purposes. The syntax of formulas and terms is given by

1st-order exp.	e	$::=$	$x \mid f(e_1,\dots,e_n)$
Formulas	A,B	$::=$	$X(e_1,\dots,e_n) \mid A\to B \mid A\wedge B \mid \forall x.A \mid \exists x.A \mid \forall X.A \mid \exists X.A$
Terms	t,u	$::=$	$\lambda x.t \mid t\,u \mid (t,u) \mid \textbf{let}\ (x,y)=t\ \textbf{in}\ u$

where $f:\mathbb{N}^n\to\mathbb{N}$ is any primitive recursive function. We write Λ for the set of all closed λ-terms, and we may use the following usual shorthands: $\top \triangleq \exists X.X$, $\bot \triangleq \forall X.X$ and $\neg A \triangleq A\to\bot$.

To simplify the use of existential quantifiers, as in [Miq11a], we introduce the following congruence rules, where the variables x, X are not free in B

$$(\exists x.A)\to B \cong \forall x.(A\to B) \qquad\qquad (\exists X.A)\to B \cong \forall X.(A\to B) \tag{1}$$

These congruences allow us to avoid having elimination rules for the existential quantifiers, thus simplifying the resulting type system. The type system, which is given in Figure 1, corresponds to the usual rules of natural deduction. The reader may observe that in particular, no computational content is given to quantifiers in the type system.

The one-step (weak head-) reduction over terms is defined by the following rules:

$$\frac{}{(\lambda x.t)u \triangleright_\beta t[u/x]} \qquad \frac{}{\textbf{let}\ (x,y)=(u,v)\ \textbf{in}\ t \triangleright_\beta t[u/x][v/y]} \qquad \frac{t\triangleright_\beta t'}{C[t]\triangleright_\beta C[t']}$$

where $C[\,] ::= [\,] \mid C[[\,]\,u] \mid C[\mathbf{let}\ (x,y) = [\,]\ \mathbf{in}\ t]$. We write $\to_\beta$ for the reflexive-transitive closure of $\triangleright_\beta$, which is known to be deterministic[1], type-preserving and normalizing on typed terms [Bar92].

2.1.2 Realizability interpretation

We will now see how to define a realizability interpretation relying on the type system defined in Figure 1. Formulas are interpreted as *saturated sets of terms*, i.e. as sets of closed terms $S \subseteq \Lambda$ such that $t \to_\beta t'$ and $t' \in S$ imply that $t \in S$. We write $\mathbf{SAT}$ to denote the set of all saturated sets and, given a formula A, we call *truth value* its realizability interpretation.

Definition 1 (Valuation)**.** A *valuation* is a function ρ that associates a natural number $\rho(x)$ to every first-order variable x and a *truth value function* $\rho(X)$, i.e. a function in $\mathbb{N}^k \to \mathbf{SAT}$ to every second-order variable X of arity k.

1. Given a valuation ρ, a first-order variable x and a natural number n, we denote by $\rho, x \mapsto n$ the valuation defined by $(\rho, x \mapsto n) \triangleq \rho_{|\,\mathrm{dom}(\rho)\setminus\{x\}} \cup \{x \mapsto n\}$.
2. Given a valuation ρ, a second-order variable X of arity k and a truth value function $F : \mathbb{N}^k \to \mathbf{SAT}$, the valuation defined by $(\rho, X \mapsto F) \triangleq \rho_{|\,\mathrm{dom}(\rho)\setminus\{X\}} \cup \{X \mapsto F\}$ will be denoted by $\rho, X \mapsto F$.

We say that a valuation ρ is *closing* the formula A if $FV(A) \subseteq \mathrm{dom}(\rho)$.

Definition 2 (Realizability interpretation)**.** We interpret closed arithmetical expressions e in the standard model of first-order Peano arithmetic $\mathbb{N}$. Given a valuation ρ and a first-order expression e (whose variables are in the domain of ρ) we denote its interpretation by $[\![e]\!]_\rho$. The interpretation of a formula A together with a valuation ρ closing A is the set $|A|_\rho$ defined inductively according to the following clauses:

$$|X(e_1,\dots,e_n)|_\rho \triangleq \rho(X)([\![e_1]\!]_\rho,\dots,[\![e_n]\!]_\rho)$$
$$|A \to B|_\rho \triangleq \{t \in \Lambda : \forall u \in |A|_\rho.(t\,u \in |B|_\rho)\}$$
$$|A \wedge B|_\rho \triangleq \{t \in \Lambda : \exists u \in |A|_\rho.\exists v \in |B|_\rho.t \to_\beta (u,v)\}$$

$$|\forall x.A|_\rho \triangleq \bigcap_{n\in\mathbb{N}} |A|_{\rho,x\mapsto n}$$
$$|\exists x.A|_\rho \triangleq \bigcup_{n\in\mathbb{N}} |A|_{\rho,x\mapsto n}$$
$$|\forall X.A|_\rho \triangleq \bigcap_{F:\mathbb{N}^k\to\mathbf{SAT}} |A|_{\rho,X\mapsto F}$$
$$|\exists X.A|_\rho \triangleq \bigcup_{F:\mathbb{N}^k\to\mathbf{SAT}} |A|_{\rho,X\mapsto F}$$

Observe that in the previous definition, the universal quantification cannot be seen as a generalized conjunction. Indeed, the conjunction is given computational content through pairs, while the universal quantifications are defined as intersections of truth values.

It is easy to see that for any formula A and any valuation ρ closing A, one has $|A|_\rho \in \mathbf{SAT}$. As it turns out, the congruences defined by Equation (1) are sound w.r.t. the interpretation.

Proposition 3 ([Miq11a])**.** *If A and A' are two formulas of* **HA2** *such that $A \cong A'$, then for all valuations ρ closing both A and A' we have $|A|_\rho = |A'|_\rho$.*

To express that the realizability interpretation is sound with respect to the type system we need the following preliminary notions.

[1] We also could have considered a non-deterministic reduction relation (*i.e.*. without enforcing any evaluation strategy) without altering the forecoming definition of the realizability interpretation. Nonetheless, this choice will provide us with a tighter preservation of reduction through the CPS translation.

Definition 4 (Substitution)**.** A *substitution* is a finite function σ from λ-variables to closed λ-terms. Given a substitution σ, a λ-variable x and a closed λ-term u, we denote by $(\sigma, x := u)$ the substitution defined by $(\sigma, x := u) \triangleq \sigma_{|\,\mathrm{dom}(\sigma)\setminus\{x\}} \cup \{x := u\}$.

Definition 5. Given a context Γ and a valuation ρ closing the formulas in Γ, we say that a substitution σ *realizes* $\rho(\Gamma)$ and write $\sigma \Vdash \rho(\Gamma)$ if $\mathrm{dom}(\Gamma) \subseteq \mathrm{dom}(\sigma)$ and $\sigma(x) \in |A|_\rho$ for every declaration $(x : A) \in \Gamma$.

Definition 6. A typing judgement $\Gamma \vdash t : A$ is *adequate* if for all valuations ρ closing A and Γ and for all substitutions $\sigma \Vdash \rho(\Gamma)$ we have $\sigma(t) \in |A|_\rho$. More generally, we say that an inference rule $\dfrac{J_1 \quad \cdots \quad J_n}{J_0}$ is adequate if the adequacy of all typing judgements $J_1, \dots, J_n$ implies the adequacy of the typing judgement J_0.

Theorem 7 (Adequacy [Miq11a])**.** *The typing rules of Figure 1 are adequate.*

The adequacy theorem is the key result when defining realizability interpretations in that fundamental properties stem from it. For example, we have the following corollary.

Corollary 8 (Consistency)**.** *There is no proof term t such that $\vdash t : \bot$.*

Proof. The proof is by *reductio ad absurdum*. It follows from Theorem 7 that if $\Gamma \vdash t : A$ is derivable, then it is adequate. In this case, this entails $t \in |\bot|_\rho = |\forall X.X|_\rho = \bigcap_{S \in \mathbf{SAT}} S = \emptyset$. □

While a complete introduction to realizability interpretations and their benefits to prove properties such as soundness or normalization of typed calculi is out of the scope of this paper[2], we would like to point out nonetheless that the proof of adequacy is very flexible. Indeed, if one wants to add a new instruction to the language of terms via its typing rule, it is enough to check that this typing rule is adequate while the remainder of the proof is exactly the same. For instance, to extend the present setting with booleans (as we shall do later on) it is enough to introduce terms $\mathsf{tt}, \mathsf{ff}, \mathsf{if}\ b\ \mathsf{then}\ t\ \mathsf{else}\ u$ with their typing rules and to prove that the latter are adequate with the realizability interpretation.

2.2 Evidenced frames

In the previous section, we have seen an example of a proof system labelled with proof terms derived from the λ-calculus. Note that if proof terms are indeed realizers, there exists realizers that are not typable. Take for example a non-typable term Ω, and observe that $(\lambda\_.\lambda x.x)\ \Omega$ is not typable but realizes $\forall X.X \to X$ because $\lambda x.x$ does. The realizability interpretation generalizes this proof system in that it is only concerned with the behavior (semantic) of proof terms and not their syntax. Note that we loose decidability but we are not concerned about it since we only want to build interpretations. The essence of a realizability interpretation lies between the interaction of a programming language and a language of formulas. The formalism of evidenced frames seeks to abstract in a unified way this structure. It is composed of a triple (with axioms) that contains two languages: evidences (programs) and proposition (formulas) as well as a relation $\cdot \dot{\to} \cdot$ that connects them.

[2] We refer the reader interested in this to the existing literature on classical realizability, *e.g.* [Rie14, Lep17, Miq17].

Definition 9 ([CMT21]). An *evidenced frame* is a triple $(\Phi, E, \cdot \dot{\to} \cdot)$, where Φ is a set of propositions, E is a collection of evidences, and $\phi_1 \overset{e}{\to} \phi_2$ is a ternary evidence relation on $\Phi \times E \times \Phi$, along with the following[3]:

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

- $\forall \phi.\ \phi \overset{e_{\mathtt{id}}}{\longrightarrow} \phi$

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

- $\forall \phi_1, \phi_2, \phi_3, e, e'.\ \phi_1 \overset{e}{\to} \phi_2$ and $\phi_2 \overset{e'}{\to} \phi_3 \implies \phi_1 \overset{e\,;e'}{\longrightarrow} \phi_3$

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

- $\forall \phi.\ \phi \overset{e_\top}{\longrightarrow} \top$

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

- $\forall \phi_1, \phi_2.\ \phi_1 \wedge \phi_2 \overset{e_{\mathtt{fst}}}{\longrightarrow} \phi_1$
- $\forall \phi_1, \phi_2.\ \phi_1 \wedge \phi_2 \overset{e_{\mathtt{snd}}}{\longrightarrow} \phi_2$
- $\forall \phi, \phi_1, \phi_2, e_1, e_2.\ \phi \overset{e_1}{\to} \phi_1$ and $\phi \overset{e_2}{\to} \phi_2 \implies \phi \overset{\langle\!\langle e_1, e_2 \rangle\!\rangle}{\longrightarrow} \phi_1 \wedge \phi_2$

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 $e_{\mathtt{eval}} \in E$:

- $\forall \varphi_1, \varphi_2, \overrightarrow{\phi}, e.\ (\forall \phi \in \overrightarrow{\phi}.\ \phi_1 \wedge \phi_2 \overset{e}{\to} \phi) \implies \phi_1 \overset{\lambda e}{\to} \phi_2 \supset \overrightarrow{\phi}$
- $\forall \phi_1, \overrightarrow{\phi}, \phi \in \overrightarrow{\phi}.\ (\phi_1 \supset \overrightarrow{\phi}) \wedge \phi_1 \overset{e_{\mathtt{eval}}}{\longrightarrow} \phi$

Given an evidenced frame $(\Phi, E, \cdot \dot{\to} \cdot)$, we say that $e \in E$ is *evidence* of $\phi \in \Phi$ if $\top \overset{e}{\to} \phi$ 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 $\bot$.

Note that contrary to cartesian closed categories, this formalism does not enforce any equation between arrows, in fact it does not allow for the axiomatization of reductions: we only require that the languages of propositions and evidences are expressive enough.

Remark 10. If there exists a huge literature describing realizability interpretations for different theories based on different notions of computations, the notion of "*realizability interpretation*" itself does not have a formal definition. The best approximation that one could come up with would probably be *an interpretation of formulas as sets of computing terms* plus some extra intuitions on how *terms should compute accordingly to the connectives they realize*. The study of its categorical counterpart gives a more precise picture: the interpretation should induce a tripos [Pit02, vO08]. As shown in [CMT21], evidenced frames are complete with respect to triposes, and reflect the structure of a realizability interpretations in a much more faithful way than triposes do. As such, "*it is an evidenced frame*" is probably the best definition one could give of a realizability interpretation, the definition of an evidenced frame and of its different components specifying how formulas are interpreted and what "*to realize*" means. In the next section, we will show how the realizability interpretation given for **HA2** indeed defines an evidenced frame, but in the sequel of the paper, the reader should understand the existence of an evidenced frame as the definition of a realizability interpretation.

[3] Observe that the different construct on propositions and evidences are actually part of the definition of an evidenced frame (in particular, several different evidenced frames may be induced from one given triple $(\Phi, E, \cdot \dot{\to} \cdot)$). In the sequel, for conciseness we may nonetheless only state the existence of evidenced frames through this triple, giving the other defining constructs in the proofs.

Definition 11. A *morphism*[4] from $\mathcal{EF}_1 = \langle \Phi_1, E_1, \cdot \dot{\to}_1 \cdot \rangle$ to $\mathcal{EF}_2 = \langle \Phi_2, E_2, \cdot \dot{\to}_2 \cdot \rangle$ is a function $F : \Phi_1 \to \Phi_2$ satisfying the following properties:

1. $\forall e_1.\, \exists e_2.\, \forall \varphi_1, \varphi_1'.\, \varphi_1 \overset{e_1}{\to}_1 \varphi_1' \implies F(\varphi_1) \overset{e_2}{\to}_2 F(\varphi_1')$
2. $\exists e_2.\, \top_2 \overset{e_2}{\to}_2 F(\top_1)$
3. $\exists e_2.\, \forall \varphi_1, \phi_1'.\, F(\varphi_1) \wedge_2 F(\varphi_1') \overset{e_2}{\to}_2 F(\varphi_1 \wedge_1 \varphi_1')$
4. $\exists e_2.\, \forall \varphi_1, \overrightarrow{\phi}_1'.\, F(\varphi_1) \supset_2 \{F(\varphi_1') \mid \varphi_1' \in \overrightarrow{\phi}_1'\} \overset{e_2}{\to}_2 F(\varphi_1 \supset_1 \overrightarrow{\phi}_1')$
5. $\exists f \in \Phi_2 \to \Phi_1.\ \Big(\big(\exists e_2.\, \forall \varphi_2.\, \varphi_2 \overset{e_2}{\to}_2 F(f(\varphi_2))\big) \bigwedge \big(\exists e_2.\, \forall \varphi_2.\, F(f(\varphi_2)) \overset{e_2}{\to}_2 \varphi_2\big)\Big)$

In broad lines, an evidenced frame morphism F from $\mathcal{EF}_1$ to $\mathcal{EF}_2$ mostly ensures (first item) that if any two propositions that are logically connected by an evidence, so are their images through F, and guarantees (second to fourth item) the existence of a *uniform* evidence that witnesses that the image of the conjunction of two propositions (and similarly for other connectives) is the logical consequence of the conjunction of images of these propositions. The last condition in turns provides us with a constructive mean to relate any proposition in the codomain Φ_2 of the morphism with the image of a proposition in Φ_1 that is logically equivalent to it (which is, again, witnessed by a pair of *uniform* evidences that do not depend of the considered proposition).

This notion of morphism provides us with the definition of a category **EF**, whose objects are evidenced frames and whose structure can be further enriched, for instance to equip morphisms from $\mathcal{EF}_1$ to $\mathcal{EF}_2$ with a preorder relation $F \preccurlyeq G$, defined to hold when there exists evidence $e_2 \in E_2$ satisfying $\forall \varphi_1 \in \Phi_1.\, F(\varphi_1) \overset{e_2}{\to}_2 G(\varphi_1)$. Again, for a more detailed introduction on evidenced frame we refer the reader to the corresponding paper [CMT21], but it is worth mentioning that any evidenced frame induces a tripos (via a uniform construction that does not depend on the considered evidenced frame), and that evidenced frames are complete with respect to triposes in the sense that the category **EF** is actually equivalent to the category **Trip** of triposes.

2.3 The induced evidenced frame $\mathcal{EF}_{\mathrm{HA2}}$

The interpretation of **HA2** given above induces an evidenced frame $\mathcal{EF}_{\mathrm{HA2}}$ whose definition simply reflects the structure of the interpretation: propositions are defined by saturated sets of terms, evidences are just λ-terms and the evidence relation is given by:

$$\psi \overset{t}{\to} \varphi \iff \forall u \in \psi.(t\ u) \in \varphi$$

This definition is reminiscent of the ordering relation on predicates induced by realizability interpretations, for instance to define triposes [Pit02].

Proposition 12 ($\mathcal{EF}_{\mathrm{HA2}}$). *The triple* $(\mathbf{SAT}, \Lambda, \cdot \dot{\to} \cdot)$ *defines an evidenced frame.*

Proof. As hinted by the realizability interpretation in Definition 2, one can simply define

$$\rho \wedge \theta \triangleq \{t \in \Lambda : \exists u \in \rho. \exists v \in \theta. t \to_\beta (u, v)\} \qquad \rho \supset \overrightarrow{\Theta} \triangleq \{t \in \Lambda : \forall u \in \rho. \forall \theta \in \overrightarrow{\Theta}. (t\,u) \in \theta\}$$

which both define saturated sets. Proving the existence of the required evidences is then an easy exercise of λ-calculus, which amounts to proving that the corresponding realizers exist. □

[4] For simplicity reasons, we adopt here an extensional presentation of evidenced frames and thus of morphisms, see [CMT21] for further discussion on intensional/extensional aspects of evidenced frames and their morphisms.

Remark 13. Even though we started from an interpretation of second-order Heyting arithmetic, we should insist on the fact that an evidenced frame always provides us with a model of higher-logic. Indeed, since propositions are viewed through their semantic counterpart (here as saturated sets of terms), this allows us to define a (semantic) quantification over any set of propositions regardless of what the syntax of the language accounts for. In fact, should we have considered a concrete example for an even simpler theory (say an interpretation of intuitionistic propositional logic based on the simply-typed λ-calculus), the induced evidenced frame would still give us an interpretation of higher-order logic.

Conversely, the definition of an evidenced frame only specifies the minimal requirements for the interpretation to give such a model, but the language of propositions and evidences can actually be richer. For instance, in Section 4.3 we shall use a primitive data type for booleans, while in [CMT21] an evidenced frame is build from a computational system allowing for stateful computations.

3 Classical logic, the $\lambda\mu\tilde{\mu}$-calculus and CPS translation

We present Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus [CH00], which we use afterwards as the source calculus of a CPS translation. To illustrate the flexibility of evidenced frame, we chose on purpose to pick a call-by-value source calculus. Besides, to ease the later definition of the CPS translation, we opted for a sequent calculus as advocated by [DMMZ10,DMAJ16,MM13], hence the choice of Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus[5].

We first recall the syntax and operational semantics of the $\lambda\mu\tilde{\mu}$-calculus, before defining a well-behaved CPS that translates it to the λ-calculus defined in Section 2.1.1. We will then study the corresponding evidenced frame $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$ and its relation to the CPS in Section 4. For the sake of simplicity, we will use the simply-typed version with pairs of the $\lambda\mu\tilde{\mu}$-calculus. Quantifications (both first and second order) will be implicitly taken care of when we define the corresponding evidenced frame $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$ in the next section.

3.1 Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus

We recall here the spirit of the Curien-Herbelin $\lambda\mu\tilde{\mu}$-calculus [CH00]. The key notion of the $\lambda\mu\tilde{\mu}$-calculus is the notion of *command*. A command $\langle t \parallel e\rangle$ can be understood as a state of an abstract machine, representing the evaluation of a *term* t (the program) against a co-proof e (the stack) that we call *context*. The syntax and reduction rules (parameterized over a subset of terms $\mathcal{V}$ and a subset of evaluation contexts $\mathcal{E}$) are given in Figure 2, where $\tilde{\mu}x.c$ can be read as a context $\mathbf{let}\ x = [\cdot]\ \mathbf{in}\ c$. The μ operator comes from Parigot's $\lambda\mu$-calculus [Par97], $\mu\alpha$ binds a context to a context variable α in the same way that $\tilde{\mu}x$ binds a proof to some proof variable x.

The $\lambda\mu\tilde{\mu}$-calculus can be seen as a proof-as-program correspondence between sequent calculus and abstract machines. Right introduction rules correspond to typing rules for

[5] In fact, an even better choice when it comes to defining operational semantics and CPS translations of calculus could have been to rely upon Munch-Maccagnoni's system L [MM13], which can be seen as a finer-grained variant of $\lambda\mu\tilde{\mu}$-syntax where the evaluation order is driven by the polarity of terms. In such a syntax, continuation-passing style translations are really easy to define in that the operational semantics is that of an abstract machine specifying at each step whether the term or the evaluation context is given the priority. We mostly chose to stick to Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus for the simplicity of the presentation, as its types system shares the same connectives than the one for **HA2** (rather than their decompositions into linear logic). In fact, call-by-value/call-by-name variants of the $\lambda\mu\tilde{\mu}$-calculus are expressible and correspond to a fixed choice of polarities when decomposing the different connectives.

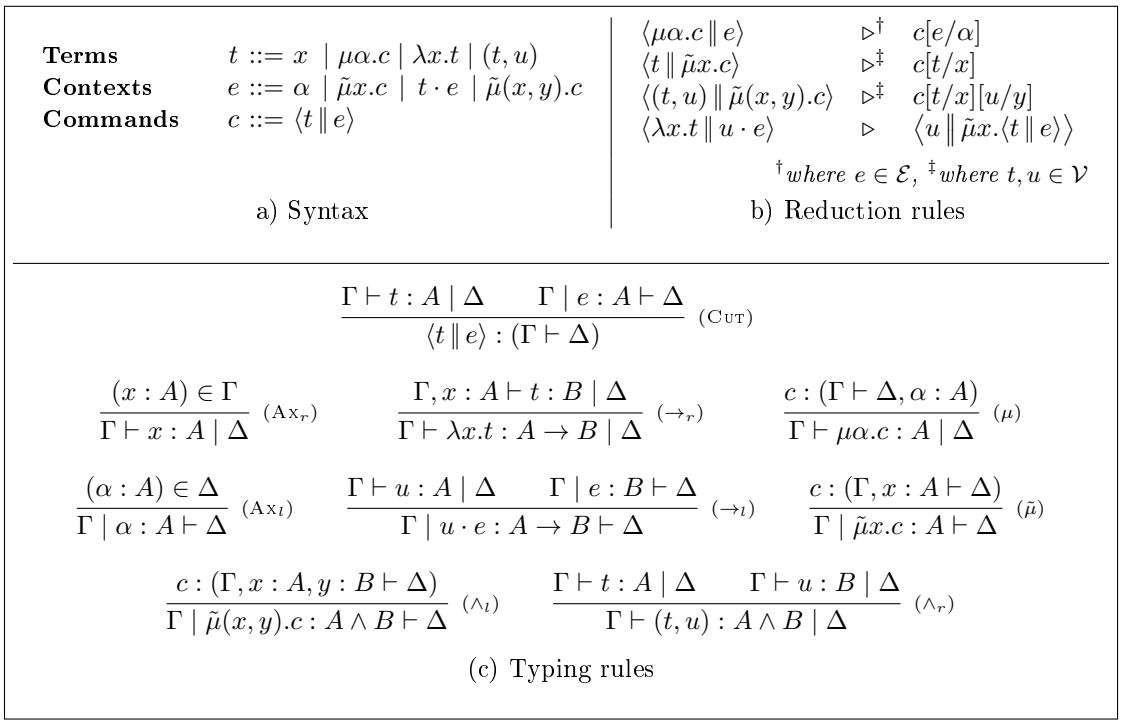

Figure 2: The simply-typed $\lambda\mu\tilde{\mu}$-calculus with pairs

proofs, while left introduction are seen as typing rules for evaluation contexts. In contrast with Gentzen's original presentation of sequent calculus, the type system of the $\lambda\mu\tilde{\mu}$-calculus explicitly identifies at any time which formula is being worked on. In a nutshell, this presentation distinguishes between three kinds of sequents: sequents of the form $\Gamma \vdash t : A \mid \Delta$ for typing terms, where the focus is put on the (right) formula A; sequents of the form $\Gamma \mid e : A \vdash \Delta$ for typing contexts, where the focus is put on the (left) formula A; sequents of the form $c : (\Gamma \vdash \Delta)$ for typing commands, where no focus is set. In a right (resp. left) sequent $\Gamma \vdash t : A \mid \Delta$, the singled out formula[6] A reads as the conclusion *"where the proof shall continue"* (resp. hypothesis *"where it happened before"*).

Regarding the reduction rules, observe that if $\mathcal{V}$ and $\mathcal{E}$ are not restricted enough, these rules admit a critical pair:

$$c[\tilde{\mu}x.c'/\alpha] \quad \triangleleft \quad \langle \mu\alpha.c \parallel \tilde{\mu}x.c' \rangle \quad \triangleright \quad c'[\mu\alpha.c/a].$$

The difference between call-by-name and call-by-value can be characterized by how this critical pair is solved, by defining $\mathcal{V}$ and $\mathcal{E}$ in such a way that the two rules do not overlap. This justifies the definition of a subcategory V of proofs, that we call *values*, and of the dual subset E of contexts that we call *co-values* (following Downen and Ariola's denomination [DA14]):

Values $V ::= x \mid \lambda x.t \mid (V_1, V_2)$ **Co-values** $E ::= \alpha \mid t \cdot e$

The call-by-name evaluation strategy amounts to the case where $\mathcal{V} \triangleq$ *Terms* and $\mathcal{E} \triangleq$ *Co-values*, while call-by-value corresponds to $\mathcal{V} \triangleq$ *Values* and $\mathcal{E} \triangleq$ *Contexts*. In the sequel, we

[6] This formula is often referred to as the formula in the *stoup*, a terminology due to Girard.

will focus on the latter and we write $\triangleright_{\mathsf{bv}}$ for the corresponding reduction steps and $\to_{\mathsf{bv}}$ for its reflexive-transitive closure. Since the by-value reduction rule for $\tilde{\mu}(x,y).c$ only computes in front of a pair of values, we restrict the syntax to these pairs and define pairs of (non-evaluated) terms through the shorthand (which simulate the left-to-right opening of such pairs):

$$(t,u) \triangleq \mu\alpha.\Big\langle t \,\Big\|\, \tilde{\mu}x.\langle u \,\|\, \tilde{\mu}y.\langle (x,y) \,\|\, \alpha\rangle\rangle\Big\rangle$$

Readers more accustomed to the λ-calculus may wonder why the syntax of terms does not include the usual application (which we write $t@u$ below), but this can be expressed as a macro, as well as the $\mathbf{let}\ \cdot = \cdot\ \mathbf{in}\ \cdot$ construct:

$$t@u \triangleq \mu\alpha.\langle t \,\|\, u\cdot\alpha\rangle \qquad\qquad \mathbf{let}\ x = t\ \mathbf{in}\ u \triangleq \mu\alpha.\Big\langle t \,\Big\|\, \tilde{\mu}x.\langle u \,\|\, \alpha\rangle\Big\rangle$$

In particular, the β-reduction is simulated by the reduction of $\lambda\mu\tilde{\mu}$ commands, since if $t,u \in \Lambda$ are such that $t \longrightarrow_\beta u$, then for any stack e we have $\langle t \,\|\, e\rangle \to_\beta^* \langle u \,\|\, e\rangle$. In our setting, we can observe that the relation $\triangleright_{\mathsf{bv}}$ induces a (weak-head) call-by-value evaluation strategy for the application, since if u reduces (in front of any context) to value V, we have:

$$\langle \lambda x.t@u \,\|\, e\rangle \quad \to_{\mathsf{bv}} \quad \Big\langle u \,\Big\|\, \tilde{\mu}x.\langle t \,\|\, e\rangle\Big\rangle \quad \to_{\mathsf{bv}} \quad \Big\langle V \,\Big\|\, \tilde{\mu}x.\langle t \,\|\, e\rangle\Big\rangle \quad \to_{\mathsf{bv}} \quad \langle t[V/x] \,\|\, e\rangle$$

that is analogous to the expected reduction thread in a call-by-value evaluated λ-calculus:

$$(\lambda x.t)\,u \quad \longrightarrow_\beta \quad \mathbf{let}\ x = u\ \mathbf{in}\ t \quad \longrightarrow_\beta \quad \mathbf{let}\ x = V\ \mathbf{in}\ t \quad \longrightarrow_\beta \quad t[V/x]$$

Readers familiar with control operators may observe that we can implement `call/cc` as well as continuations in this calculus as follows:

$$\texttt{call/cc} \triangleq \lambda x.\mu\alpha.\langle x \,\|\, \mathbf{k}_\alpha \cdot \alpha\rangle \qquad\qquad \mathbf{k}_e \triangleq \lambda y.\mu\_.\langle y \,\|\, e\rangle$$

Forgetting a minute about the call-by-value restriction, we can check that these definitions yield the expected computational behavior of `call/cc`: in front of a stack $t\cdot e$, it catches the context e thanks to the $\mu\alpha$ binder and reduces as follows:

$$\langle \texttt{call/cc} \,\|\, t\cdot e\rangle \;=\; \Big\langle \lambda x.\mu\alpha.\langle x \,\|\, \mathbf{k}_\alpha\cdot\alpha\rangle \,\Big\|\, t\cdot e\Big\rangle \quad \triangleright \quad \Big\langle \mu\alpha.\langle t \,\|\, \mathbf{k}_\alpha\cdot\alpha\rangle \,\Big\|\, e\Big\rangle \quad \triangleright \quad \langle t \,\|\, \mathbf{k}_e\cdot e\rangle$$

In turns, in front of a stack $u\cdot e'$, the continuation $\mathbf{k}_e$ will now catch the context e' and throw it away to restore the former context e:

$$\langle \mathbf{k}_e \,\|\, u\cdot e'\rangle \;=\; \Big\langle \lambda y.\mu\_.\langle y \,\|\, e\rangle \,\Big\|\, u\cdot e'\Big\rangle \quad \triangleright \quad \Big\langle \mu\_.\langle u \,\|\, e\rangle \,\Big\|\, e'\Big\rangle \quad \triangleright \quad \langle u \,\|\, e\rangle$$

From a logical perspective, it is an easy exercise to check that `call/cc` can be typed with Peirce's law.

3.2 Continuation-passing style translation of the call-by-value $\lambda\mu\tilde{\mu}$-calculus

We define a CPS translation for the call-by-value variant of the $\lambda\mu\tilde{\mu}$-calculus introduced previously which is analogous to the translations in [CH00, Miq17]. The definition of the translation can be mechanically derived from the operational semantics of the calculus, following the methodology of Danvy's semantic artifacts described in [ADH+12, Miq17], or the decomposition of the $\lambda\mu\tilde{\mu}$-calculus into Munch-Maccagnoni's system L [MM13]. As is usual in

call-by-value, the translation is defined on three layers, reflecting the three syntactic categories at play: $[\cdot]_{\mathsf{t}}$ on terms, $[\cdot]_{\mathsf{e}}$ on contexts and $[\cdot]_{\mathsf{V}}$ on values.

$$
\begin{array}{ll|ll}
[\langle t \parallel e\rangle]_{\mathsf{c}} & \triangleq [t]_{\mathsf{t}}\ [e]_{\mathsf{e}} & [\tilde{\mu}(x,y).c]_{\mathsf{e}} & \triangleq \lambda V.\mathbf{let}\ (x,y) = V\ \mathbf{in}\ [c]_{\mathsf{c}} \\
[\mu\alpha.c]_{\mathsf{t}} & \triangleq \lambda e.(\lambda\alpha.[c]_{\mathsf{c}})\, e & [\alpha]_{\mathsf{e}} & \triangleq \alpha \\
[V]_{\mathsf{t}} & \triangleq \lambda e.e\,[V]_{\mathsf{V}} & [x]_{\mathsf{V}} & \triangleq x \\
[u\cdot e]_{\mathsf{e}} & \triangleq \lambda V.V\ [u]_{\mathsf{t}}\ [e]_{\mathsf{e}} & [(V_1,V_2)]_{\mathsf{V}} & \triangleq ([V_1]_{\mathsf{V}}, [V_2]_{\mathsf{V}}) \\
[\tilde{\mu}x.c]_{\mathsf{e}} & \triangleq \lambda V.(\lambda x.[c]_{\mathsf{c}})\, V & [\lambda x.t]_{\mathsf{V}} & \triangleq \lambda u e.u\,(\lambda x.[t]_{\mathsf{t}}\ e)
\end{array}
$$

The computational translation induces the following translation on types:

$$
\begin{array}{ll|ll}
[\![A]\!]_{\mathsf{t}} & \triangleq [\![A]\!]_{\mathsf{e}} \to \mathcal{R} & [\![A \to B]\!]_{\mathsf{V}} & \triangleq [\![A]\!]_{\mathsf{t}} \to [\![B]\!]_{\mathsf{e}} \to \mathcal{R} \\
[\![A]\!]_{\mathsf{e}} & \triangleq [\![A]\!]_{\mathsf{V}} \to \mathcal{R} & [\![A \wedge B]\!]_{\mathsf{V}} & \triangleq [\![A]\!]_{\mathsf{V}} \wedge [\![B]\!]_{\mathsf{V}} \\
 & & [\![X]\!]_{\mathsf{V}} & \triangleq X
\end{array}
$$

where $\mathcal{R}$ is the return type of continuations, usually defined as $\mathcal{R} \triangleq \bot$. This translation extends naturally to contexts, where the translation of Γ is defined in terms of values while Δ is translated in terms of contexts:

$$[\![\Gamma, x : A]\!]_{\mathsf{V}} \triangleq [\![\Gamma]\!]_{\mathsf{V}}, x : [A]_{\mathsf{V}} \qquad\qquad [\![\Delta, \alpha : A]\!]_{\mathsf{e}} \triangleq [\![\Delta]\!]_{\mathsf{e}}, \alpha : [\![A]\!]_{\mathsf{e}}$$

Remark 14. Usually, the return type $\mathcal{R}$ is chosen to be some specific formula of the target language (here **HA2**). Actually, we can even consider an even more general settings, where we extend the language of formulas with a new constant $\mathcal{R}$ for which we only have to provide its realizability interpretation as a saturated set of terms (*i.e.* $|\mathcal{R}| \in \mathbf{SAT}$). Since no typing rule is provided for $\mathcal{R}$, it can be understood as a generalization of $\bot$: any derivation of $\mathcal{R}$ can be turned into a derivation of $\bot$ (and vice-versa), but contrarily to $|\bot| = |\forall X.X| = \emptyset$, the semantic interpretation of $\mathcal{R}$ can be chosen not to be empty.

The translation of terms, contexts and commands is sound with respect both to types and computations[7], as shown by the following propositions:

Proposition 15 ([CH00, Miq17]). *For any contexts Γ and Δ, we have*

1. *if* $\Gamma \vdash t : A \mid \Delta$ *then* $[\![\Gamma]\!]_{\mathsf{V}}, [\![\Delta]\!]_{\mathsf{e}} \vdash [t]_{\mathsf{t}} : [\![A]\!]_{\mathsf{t}}$
2. *if* $\Gamma \mid e : A \vdash \Delta$ *then* $[\![\Gamma]\!]_{\mathsf{V}}, [\![\Delta]\!]_{\mathsf{e}} \vdash [e]_{\mathsf{e}} : [\![A]\!]_{\mathsf{e}}$
3. *if* $c : \Gamma \vdash \Delta$ *then* $[\![\Gamma]\!]_{\mathsf{V}}, [\![\Delta]\!]_{\mathsf{e}} \vdash [c]_{\mathsf{c}} : \mathcal{R}$

Proposition 16 (Simulation). *For any c, c', we have: $c \to_{\mathsf{bv}} c'$ if and only if $[c]_{\mathsf{c}} \longrightarrow_{\beta} [c']_{\mathsf{c}}$.*

Proof. The direct implication is standard and proven by induction on $\triangleright_{\mathsf{bv}}$ [Miq17]. The "only if" part is proven by contradiction, by considering c and c' with the shortest reduction path $[c]_{\mathsf{c}} \longrightarrow_{\beta} [c']_{\mathsf{c}}$ possible, then reasoning by induction on c. □

4 CPS translation of realizers

We are now ready to examine Oliva and Streicher result through the lens of evidenced frames, and investigate the main question of this paper.

[7] To be even more precise, we could restrict ourselves to a weak-head call-by-name evaluation strategy with the same result.

4.1 A call-by-value classical realizability interpretation

We begin by defining the evidenced frame induced by the realizability interpretation that the call-by-value $\lambda\mu\tilde{\mu}$-calculus yields. The structure of this interpretation differs from the (intuitionistic) interpretation introduced in Section 2.1.2 mostly for two reasons: a) it is a classical (*à la* Krivine) interpretation, and b) it is based on a call-by-value calculus.

As in intuitionistic realizability, every formula A is interpreted in classical realizability as a set $|A|_{\mathsf{t}}$ of terms (the realizers) that share a common computational behavior determined by the structure of the formula A [Kri04]. However the difference between intuitionistic and classical realizability is that in the latter, the set of realizers of A is defined indirectly, that is from a set $\|A\|_{\mathsf{e}}$ of contexts that are intended to challenge the truth of A. Intuitively, the set $\|A\|_{\mathsf{e}}$ (which we shall call the *falsity value* of A) can be understood as the set of all possible counter-arguments to the formula A. In this framework, a program realizes the formula A if and only if it is able to defeat all the attempts to refute A by a context in $\|A\|_{\mathsf{e}}$.

When defining such an interpretation on a call-by-value calculus, the falsity value itself is in fact defined in terms of a more primitive notions of truth values of values [MM09]. This set, which we write $|A|_{\mathsf{V}}$, can be understood as the values that any test challenging A should accept as a valid answer.

The last ingredient peculiar to Krivine realizability is the fact that realizability interpretations are parameterized by a set of commands, the *pole*, which intuitively represents the valid computations.

Definition 17 (Pole)**.** A pole $\perp\!\!\!\perp$ is a saturated set of commands, *i.e.* a set such that if $c \to_{\mathrm{bv}} c'$ and $c' \in \perp\!\!\!\perp$ then $c \in \perp\!\!\!\perp$. It should be seen as a set of commands whose computations end with success.

Definition 18 (Orthogonal)**.** Given a pole $\perp\!\!\!\perp$, we define $A^{\perp\!\!\!\perp}$ to be the orthogonal of A: if A is a set of terms (resp. contexts), it is the set of contexts e (resp. terms t) such that $\langle t \,\|\, e\rangle \in \perp\!\!\!\perp$.

The complete definition of the realizability interpretation based on the call-by-value $\lambda\mu\tilde{\mu}$-calculus introduced before would require once again to define the appropriate notions of valuations, adequate judgements, etc... Such definitions can be found for analogous interpretation in [MM09, Lep16, Miq20], and will be somewhat hidden here between the lines of the definition of the induced evidenced frame $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$. Let us however explain how connectives are interpreted in terms of values. The aforementioned sets of truth and falsity values are then defined by orthogonality to each others, *i.e.* $|A|_{\mathsf{t}} = \|A\|_{\mathsf{e}}^{\perp\!\!\!\perp}$ and $\|A\|_{\mathsf{e}} = |A|_{\mathsf{V}}^{\perp\!\!\!\perp}$, which in particular implies that $|A|_{\mathsf{V}} \subseteq |A|_{\mathsf{t}}$. For A and B two closed formulas, values realizing implication (resp. the conjunction) are the expected ones, namely functions (resp. pairs):

$$|A \to B|_{\mathsf{V}} = \{\lambda x.t : \forall V \in |A|_{\mathsf{V}} : t[V/x] \in |B|_{\mathsf{t}}\} \qquad |A \wedge B|_{\mathsf{V}} = \{(V_1, V_2) : V_1 \in |A|_{\mathsf{V}} \wedge V_2 \in |B|_{\mathsf{V}}\}$$

Recall that even if the type system introduced earlier does not include quantifiers, we can nevertheless define them through their semantic interpretation in terms of values, namely as an intersection of primitive truth values (where ρ should be the appropriate notion of valuation for this setting, and X ranges over propositions):

$$|\forall x.A|^{\rho}_{\mathsf{V}} = \bigcap_{n \in \mathbb{N}} |A|^{\rho, x \mapsto n}_{\mathsf{V}} \qquad |\forall X.A|^{\rho}_{\mathsf{V}} = \bigcap_{F \in \mathcal{P}(V)} |A|^{\rho, X \mapsto F,}_{\mathsf{V}}$$

The corresponding evidenced frame thus uses sets of values as propositions, terms as evidences (we write $\mathcal{T}$ for the set of closed terms) and the following evidence relation

$$\varphi \overset{t}{\to} \psi \iff \forall V \in \varphi : t@V \in \psi^{\perp\!\!\!\perp\perp\!\!\!\perp}$$

Theorem 19 ($\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$). *The triple* $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}} \triangleq (\mathcal{P}(V), \mathcal{T}, \cdot \dot{\to} \cdot)$ *defines an evidenced frame.*

4.2 CPS as a morphism

We now wish to investigate whether the CPS translation defined in Section 3.2 defines an evidenced frame morphism from $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$ to $\mathcal{EF}_{\mathrm{HA2}}$. To be precise, recall that so far our definitions for the interpretations of the source and destination of the translation leave us two degrees of liberty: the choice of pole in $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$, which we shall write $\perp\!\!\!\perp_{\mathsf{s}}$; and the realizability interpretation of the return type $\mathcal{R}$, which we write $|\mathcal{R}| = \perp\!\!\!\perp_{\mathsf{d}}$. As we shall see in Section 4.3, we cannot build a morphism that works for any choice for these parameters and these two evidenced frames. However, we investigate the following questions: given an interpretation $\perp\!\!\!\perp_{\mathsf{s}}$, can we find a pole $\perp\!\!\!\perp_{\mathsf{d}}$ for $\mathcal{R}$ such that $[\cdot]_{\mathsf{V}}$ defines an evidenced frame morphism? Reciprocally, given a pole $\perp\!\!\!\perp_{\mathsf{d}}$, can we find an appropriate $\perp\!\!\!\perp_{\mathsf{s}}$?

4.2.1 The forward evidenced frame

We first tackle the first problem, that is to define from a fixed pole $\perp\!\!\!\perp_{\mathsf{s}}$ in the source an interpretation $\perp\!\!\!\perp_{\mathsf{d}}$ for $\mathcal{R}$ such that the CPS induces a morphism, we can restrict ourselves to consider instead the image of $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$ through the CPS translation as an evidenced frame itself (as is done in [OS08]). To begin with, we define the interpretation of the return type $\mathcal{R}$ as the saturation of the image of the pole $\perp\!\!\!\perp_{\mathsf{s}}$ through the CPS:

$$\perp\!\!\!\perp_{\mathsf{d}} \triangleq \{t : \exists c \in \perp\!\!\!\perp_{\mathsf{s}}. t \longrightarrow_{\beta} [c]_{\mathsf{c}}\}$$

We take as propositions the images of sets of values through the translation of values $[\cdot]_{\mathsf{V}}$, *i.e.* $\Phi_{\mathsf{fw}} \triangleq [\mathcal{P}(\text{values})]_{\mathsf{V}}$. Similarly, we take evidences to be translated terms: $E_{\mathsf{fw}} \triangleq [\text{terms}]_{\mathsf{t}}$. As for the evidence relation, we can see it as the image of the evidence relation in $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$ through the translation (up to some technical details), that is:

$$\varphi \xrightarrow{t}_{\mathsf{fw}} \psi \triangleq \forall V \in \phi. \forall e \in \text{contexts}. [(\forall [V']_{\mathsf{V}} \in \psi : [V']_{\mathsf{t}}\ [e]_{\mathsf{e}} \in \perp\!\!\!\perp_{\mathsf{d}}) \implies (t \# V)\ [e]_{\mathsf{e}} \in \perp\!\!\!\perp_{\mathsf{d}}]$$

where $a \# b \triangleq \lambda\alpha. a\ (\lambda V. V\ (\lambda e. e\ b)\ \alpha)$. This operator is solely motivated by technical reasons, in order to satisfy the equation $[t@V]_{\mathsf{t}} = [t]_{\mathsf{t}} \#[V]_{\mathsf{V}}$.

Theorem 20. $\mathcal{EF}_{\mathsf{fw}} = (\Phi_{\mathsf{fw}}, E_{\mathsf{fw}}, \cdot \dot{\to}_{\mathsf{fw}} \cdot)$ *defines an evidenced frame and the map* $F : V \mapsto [V]_{\mathsf{V}}$ *induces a morphism from* $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$ *to* $\mathcal{EF}_{\mathsf{fw}}$.

Proof. The first part mostly relies on the observation that $[\cdot]_{\mathsf{t}}$ is injective and admits an inverse (see the proof of Prop. 22). The evidences that make the morphism well-behaved w.r.t. the connectives are just obtained as the translation $[\cdot]_{\mathsf{t}}$ of the corresponding evidences in $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$. □

As the proof of Theorem 20 shows, not only does the CPS translation induce a morphism, but it does so while preserving evidences. In other words, in this setting, we can actually conclude that the CPS translation does preserve realizers. Observe nonetheless that this is almost a tautology, in that evidences in $\mathcal{EF}_{\mathsf{fw}}$ are by definition translated terms. In particular, if $\mathcal{EF}_{\mathsf{fw}}$ defines a model for **HA2**, it is by nature very different from the one that $\mathcal{EF}_{\mathrm{HA2}}$ provides.

4.2.2 The backward evidenced frame

We now tackle the other question, that is, we consider a fixed interpretation $\perp\!\!\!\perp_{\mathsf{d}} \in \mathbf{SAT}$ of the return type $\mathcal{R}$ in the realizability interpretation. To define a pole in the source of the

translation, we simply pick the sets $\bot\!\!\!\bot_{\mathsf{s}} = \{c : [c]_{\mathsf{c}} \in \bot\!\!\!\bot_{\mathsf{d}}\}$ of commands whose translations belong to $\bot\!\!\!\bot_{\mathsf{d}}$. This indeed defines a saturated set of commands, since the translation $[\cdot]_{\mathsf{c}}$ preserves computation and $\bot\!\!\!\bot_{\mathsf{d}}$ is itself a saturated set of terms.

For technical reasons, we now take propositions to be pairs made of the translation of a proposition in $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$ and the translation of its orthogonal set: $\Phi_{\mathsf{bw}} = \{([S]_{\mathsf{V}}, [S^{\bot\!\!\!\bot}]_{\mathsf{e}}) : S \in \mathcal{P}(\mathcal{V})\}$. Evidences are again defined as translation of terms $E_{\mathsf{bw}} = [\text{terms}]_{\mathsf{t}}$ while the evidence relation is given by:

$$(\varphi^V, \varphi^e) \xrightarrow{t}_{\mathsf{bw}} (\psi^V, \psi^e) \iff \forall V \in \varphi^V . \forall e \in \psi^e . (t\#V)\ e \in \bot\!\!\!\bot_{\mathsf{d}}$$

Theorem 21. *The triple $\mathcal{EF}_{\mathsf{bw}} = (\Phi_{\mathsf{bw}}, E_{\mathsf{bw}}, \cdot \dot{\to}_{\mathsf{bw}} \cdot)$ defines an evidenced frame and the map $F : V \mapsto ([V]_{\mathsf{V}}, [V^{\bot\!\!\!\bot}]_{\mathsf{e}})$ induces a morphism from $\mathcal{EF}^{\mathsf{bv}}_{\mu\tilde{\mu}}$ to $\mathcal{EF}_{\mathsf{bw}}$.*

The proof is analogous to the one of Theorem 20 (in particular, realizers are also trivially preserved in this case), and actually unveils that $\mathcal{EF}_{\mathsf{fw}}$ and $\mathcal{EF}_{\mathsf{bw}}$ are essentially the same.

Proposition 22. *For any term t, we have: $(\varphi^V, \varphi^e) \xrightarrow{[t]_{\mathsf{t}}}_{\mathsf{bw}} (\psi^V, \psi^e) \iff \varphi^V \xrightarrow{[t]_{\mathsf{t}}}_{\mathsf{fw}} \psi^V$.*

Proof. We prove that both statement are equivalent to ${[\varphi]_{\mathsf{V}}}^{-1} \xrightarrow{t}_{\mathsf{bv}} {[\psi]_{\mathsf{V}}}^{-1}$, where $[\cdot]_{\mathsf{V}}^{-1}$ is the inverse of the injective map $[\cdot]_{\mathsf{V}}$. This relies in turn on the definitions of the poles $\bot\!\!\!\bot_{\mathsf{d}}$, $\bot\!\!\!\bot_{\mathsf{s}}$ and on the fact that the CPS translation simulates computations (Proposition 16). □

4.3 A counter-example

We shall now give an example of a pole $\bot\!\!\!\bot_{\mathsf{s}}$ together with a term that is a realizer in the interpretation based on the $\lambda\mu\tilde{\mu}$-calculus, but whose translation is not a realizer of the translated formula. We follow here the lines of Oliva-Streicher's presentation of Krivine realizability through a CPS translation, in particular we assume that an interpretation of the return type $\bot\!\!\!\bot_{\mathsf{d}}$ is given and we define the pole $\bot\!\!\!\bot_{\mathsf{s}} \triangleq \{c : [c]_{\mathsf{c}} \in \bot\!\!\!\bot_{\mathsf{d}}\}$ consequently. In the rest of this section, we write $t \Vdash_{\mathsf{s}} A$ when t is a $\lambda\mu\tilde{\mu}$-term which realizes A in the source, and $t \Vdash_{\mathsf{d}} A$ when t is a λ-term realizing A in the destination.

To give a simple example, we will extend thereafter the definitions of the source and destination of the CPS to include a type $\mathbb{B}$ of booleans, whose translation at the level of types will be given by: $[\![\mathbb{B}]\!]_{\mathsf{t}} = (\mathbb{B} \to \mathcal{R}) \to \mathcal{R}$, and we will present a term t such that $t \Vdash_{\mathsf{s}} \mathbb{B}$ but $[t]_{\mathsf{t}} \not\Vdash_{\mathsf{d}} [\![\mathbb{B}]\!]_{\mathsf{t}}$. Let us briefly guide the reader through the rationale of our construction. To prove that $[t]_{\mathsf{t}} \not\Vdash_{\mathsf{d}} [\![\mathbb{B}]\!]_{\mathsf{t}}$, we shall exhibit a continuation $k \Vdash_{\mathsf{d}} \mathbb{B} \to \mathcal{R}$ such that $[t]_{\mathsf{t}}\ k \notin \bot\!\!\!\bot_{\mathsf{d}}$. By definition, if $t \Vdash_{\mathsf{s}} \mathbb{B}$ in particular for any context $e \in \|\mathbb{B}\|_{\mathsf{e}}$, the fact that $\langle t \parallel e \rangle \in \bot\!\!\!\bot_{\mathsf{s}}$ means that $[t]_{\mathsf{t}}\ [e]_{\mathsf{e}} \in \bot\!\!\!\bot_{\mathsf{d}}$. Therefore we should look for a continuation that is *not* in the image of the CPS translation. Besides, $[t]_{\mathsf{t}}\ k$ should *not* reduce to $k\,b$ where b is a boolean, since otherwise the fact that $k \Vdash_{\mathsf{d}} \mathbb{B} \to \mathcal{R}$, $b \Vdash_{\mathsf{d}} \mathbb{B}$ and antireduction would also entail that $[t]_{\mathsf{t}}\ k \in \bot\!\!\!\bot_{\mathsf{d}}$. For these reasons, we will pick an inert constant κ for t, define the pole $\bot\!\!\!\bot_{\mathsf{d}}$ to enforce $\kappa \Vdash_{\mathsf{d}} \mathbb{B}$ and select a continuation k that will be syntactically discriminated for any translated context.

4.3.1 Extension with booleans

We briefly review the extensions of the $\lambda\mu\tilde{\mu}$-calculus and of the λ-calculus to include a type $\mathbb{B}$ of booleans in the corresponding languages of formulas. We first extend the syntax and reduction rules of the $\lambda\mu\tilde{\mu}$-calculus to account for boolean values tt and ff, a context to eliminate booleans and an inert term κ:

Terms	$t ::= \ldots \mid \kappa$	$\langle \mathsf{tt} \parallel \tilde{\mu}\mathbb{B}.[c_1 \mid c_2] \rangle \triangleright_{\mathsf{bv}} c_1$
Values	$V ::= \ldots \mid \mathsf{tt} \mid \mathsf{ff}$	$\langle \mathsf{ff} \parallel \tilde{\mu}\mathbb{B}.[c_1 \mid c_2] \rangle \triangleright_{\mathsf{bv}} c_2$
Contexts	$e ::= \ldots \mid \tilde{\mu}\mathbb{B}.[c_1 \mid c_2]$	

These terms can be typed with the following rules:

$$\frac{c_1 : (\Gamma \vdash \Delta) \qquad c_1 : (\Gamma \vdash \Delta)}{\Gamma \mid \tilde{\mu}\mathbb{B}.[c_1 \mid c_2] : \mathbb{B} \vdash \Delta}\ (\mathbb{B}_l) \qquad \frac{}{\Gamma \vdash \mathsf{tt} : \mathbb{B} \mid \Delta}\ (\mathsf{tt}) \qquad \frac{}{\Gamma \vdash \mathsf{ff} : \mathbb{B} \mid \Delta}\ (\mathsf{ff})$$

Similarly, the λ-calculus that serves as a destination of the CPS translation can be extended with booleans and an inert constant, for which we use the same notations:

$$t, u ::= \ldots \mid \kappa \mid \mathsf{ff} \mid \mathsf{tt} \mid \mathsf{if}\ t\ \mathsf{then}\ u\ \mathsf{else}\ v \quad \Big| \quad \overline{\mathsf{if}\ \mathsf{tt}\ \mathsf{then}\ u\ \mathsf{else}\ v \triangleright_\beta u} \qquad \overline{\mathsf{if}\ \mathsf{ff}\ \mathsf{then}\ u\ \mathsf{else}\ v \triangleright_\beta v}$$

Once again, we can give typing rules to the terms computing with booleans:

$$\frac{\Gamma \vdash b : \mathbb{B} \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{if}\ b\ \mathsf{then}\ t\ \mathsf{else}\ u : A}\ (\mathsf{if}) \qquad \frac{}{\Gamma \vdash \mathsf{tt} : \mathbb{B}}\ (\mathsf{tt}) \qquad \frac{}{\Gamma \vdash \mathsf{ff} : \mathbb{B}}\ (\mathsf{ff})$$

The CPS translation is then naturally extended (following what is already defined for values and base types):

$$[\![\mathbb{B}]\!]_{\mathsf{V}} \triangleq \mathbb{B} \quad \Big| \quad [\mathsf{tt}]_{\mathsf{V}} \triangleq \mathsf{tt} \qquad [\mathsf{ff}]_{\mathsf{V}} \triangleq \mathsf{ff} \quad \Big| \quad [\tilde{\mu}\mathbb{B}.[c_1 \mid c_2]]_{\mathsf{e}} \triangleq \lambda b.\, \mathsf{if}\ b\ \mathsf{then}\ [c_1]_{\mathsf{c}}\ \mathsf{else}\ [c_2]_{\mathsf{c}} \quad \Big| \quad [\kappa]_{\mathsf{t}} \triangleq \kappa$$

It is straightforward to verify that the different properties of the CPS translation still hold for the extended calculi.

4.3.2 An untranslatable realizer

It now only remains to extend the realizability interpretation, in the $\lambda\mu\tilde{\mu}$-calculus we simply define the primitive truth value of $\mathbb{B}$ as the set containing the two boolean values

$$|\mathbb{B}|_{\mathsf{V}} \triangleq \{\mathsf{tt}, \mathsf{ff}\}$$

while for the intuitionistic interpretation based on the target calculus, we consider the set of terms reducing to a boolean value:

$$|\mathbb{B}| \triangleq \{t \longrightarrow_\beta \mathsf{tt}\} \cup \{t \longrightarrow_\beta \mathsf{ff}\}$$

These definitions make the typing rules for booleans adequate with the corresponding realizability interpretations.

Finally, we define the pole (and thus the realizability interpretations), which will allow us to complete the construction of the counter-example:

$$\perp\!\!\!\perp_{\mathsf{d}} \triangleq \{u \in \Lambda : u \longrightarrow_\beta [t]_{\mathsf{t}}\ \lambda x.v \text{ for some term } v\} \qquad \perp\!\!\!\perp_{\mathsf{s}} \triangleq \{c : [c]_{\mathsf{c}} \in \perp\!\!\!\perp_{\mathsf{d}}\}$$

Lemma 23. *We have* $\kappa \Vdash_{\mathsf{s}} \mathbb{B}$

Proof. Since any context e translates into a function of the shape $\lambda x.v$, for any context $e \in \|\mathbb{B}\|_{\mathsf{e}}$, we have in particular $[\langle \kappa \,\|\, e\rangle]_{\mathsf{c}} = \kappa\, [e]_{\mathsf{e}} \in \perp\!\!\!\perp_{\mathsf{d}}$, and then $\langle \kappa \,\|\, e\rangle \in \perp\!\!\!\perp_{\mathsf{s}}$. □

Lemma 24. *Let us fix* $p \in \perp\!\!\!\perp_{\mathsf{d}}$*, and define* $\mathbf{k} \triangleq (\lambda x.x)\lambda b.p$*. Then it holds that* $\mathbf{k} \Vdash_{\mathsf{d}} \mathbb{B} \to \mathcal{R}$*.*

Proof. Recall that we define $|\mathcal{R}| = \perp\!\!\!\perp_{\mathsf{d}}$ and $|\mathbb{B}| = \{t : t \longrightarrow_\beta b \text{ with } b \in \mathbb{B}\}$. Let $b \in \mathbb{B}$, and t be such that $t \longrightarrow_\beta b$. We have

$$\mathbf{k}\, t = ((\lambda x.x)\lambda b.p)t \ \longrightarrow_\beta \ (\lambda b.p)t \ \longrightarrow_\beta \ p \in \perp\!\!\!\perp_{\mathsf{d}}$$

hence the fact that $\mathbf{k} \Vdash_{\mathsf{d}} \mathbb{B} \to \mathcal{R}$ by antireduction. □

We can now check that :

Proposition 25. *We have* $[\kappa]_{\mathsf{t}} \not\Vdash_{\mathsf{d}} [\![\mathbb{B}]\!]_{\mathsf{t}}$.

Proof. We have $[\kappa]_{\mathsf{t}}\ \mathbf{k} = \kappa\,\mathbf{k}$. Since this term does not reduce (recall that β is a weak-head reduction relation) and $\mathbf{k}$ is not of the shape $\lambda x.v$, it does not belong to $\perp\!\!\!\perp_{\mathsf{d}}$. The result then follows from Lemma 24. □

Nonetheless, this example is not entirely satisfying in that the subsequent realizability interpretation is not coherent:

Proposition 26. *The closed term* $t_\perp \triangleq \mu\alpha.\langle \kappa \,\|\, \tilde{\mu}x.\langle x \,\|\, \alpha\rangle\rangle$ *satisfies* $t_\perp \Vdash_{\mathsf{s}} \perp$.

Proof. Indeed, for any context e we have:

$$\langle t_\perp \,\|\, e\rangle \ \triangleright_{\mathsf{bv}}\ \langle \kappa \,\|\, \tilde{\mu}x.\langle x \,\|\, e\rangle\rangle$$

Since $[\tilde{\mu}x.\langle x \,\|\, e\rangle]_{\mathsf{e}} = \lambda x.(\lambda k.k\,x)[e]_{\mathsf{e}}$, in particular we have $\kappa\,[\tilde{\mu}x.\langle x \,\|\, e\rangle]_{\mathsf{e}} \in \perp\!\!\!\perp_{\mathsf{d}}$ and thus $\langle t_\perp \,\|\, e\rangle \in \perp\!\!\!\perp_{\mathsf{s}}$ by antireduction. □

5 Conclusion

As mentioned in the introduction, the motivation behind this work was twofold. On the one hand, from a methodological perspective, this work also was a pretext to experiment with evidenced frames as a tool to reason on realizability interpretation. As such, the result of this experimentation turns out to be pretty satisfactory, for they have shown to be very helpful during the research process to identify the key ingredients necessary for the different results exposed. In particular, they provided us with a precious algebraic viewpoint to avoid loosing ourselves in the implementation details of realizers and translations. In particular, we could easily reproduce Oliva-Streicher's construction while using different calculi.

On the other hand, from a logical perspective, we were actually really interested in the question raised in the title. In that regards, we gave here partial answers, by giving some sufficient conditions, namely the restriction to the backward and forward evidenced frame, for CPS translations to be well-behaved with respect to the realizability interpretations; as well as an example of two realizability interpretations where the translation does not preserve realizers. Many interesting questions remain to be explored in that direction: are there counter-examples that do not require an incoherent pole? What can be said in general of realizers that are compatible with the CPS translation? In particular, which are the terms that are *always* compatible (*i.e.* regardless of the choice of $\perp\!\!\!\perp_{\mathsf{d}}$ and $\perp\!\!\!\perp_{\mathsf{s}}$) with a CPS? In particular, it would be very interesting to wonder whether *universal* realizers (*i.e.* compatible with any pole) are soundly CPS translated. This can be shown for simple data types (for instance boolean or natural numbers) by means of specification techniques [GM16a, GM16b] (but more details on this would drive us out of the scope of this paper), and is still to be investigated for more complex types (functions, etc.).

It is worth mentioning that all these questions go beyond the sole case of CPS translations and also extend to other syntactic translations, and in particular to a wide range of effects accounted for by monads. On a long-term perspective, a lot is yet to be learned on the syntactic translations that are compatible with semantics interpretations, that is, which yield EF morphisms preserving evidences.

References

[ADH+12] Zena M. Ariola, Paul Downen, Hugo Herbelin, Keiko Nakata, and Alexis Saurin. Classical call-by-need sequent calculi: The unity of semantic artifacts. In Tom Schrijvers and Peter Thiemann, editors, *Functional and Logic Programming - 11th International Symposium, FLOPS 2012, Kobe, Japan, May 23-25, 2012. Proceedings*, Lecture Notes in Computer Science, pages 32–46. Springer, 2012.

[App92] Andrew W. Appel. *Compiling with Continuations*. Cambridge University Press, New York, NY, USA, 1992.

[Bar92] Henk Barendregt. Lambda calculi with types. In S. Abramsky, Dov M. Gabbay, and S. E. Maibaum, editors, *Handbook of Logic in Computer Science (Vol. 2)*, pages 117–309. Oxford University Press, Inc., New York, NY, USA, 1992.

[BPT17] Simon Boulier, Pierre-Marie Pédrot, and Nicolas Tabareau. The next 700 syntactical models of type theory. In *Proceedings of CPP 2017*, pages 182–194, New York, NY, USA, 2017. ACM.

[CH00] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In *Proceedings of ICFP 2000*, SIGPLAN Notices 35(9), pages 233–243. ACM, 2000.

[CMT21] Liron Cohen, Étienne Miquey, and Ross Tate. Evidenced frames: A unifying framework broadening realizability models. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, 2021.

[DA14] Paul Downen and Zena M. Ariola. The duality of construction. In Zhong Shao, editor, *Programming Languages and Systems*, pages 249–269, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[DMAJ16] Paul Downen, Luke Maurer, Zena M. Ariola, and Simon Peyte Jones. Sequent calculus as a compiler intermediate language. In *ICFP 2016*, 2016.

[DMMZ10] Olivier Danvy, Kevin Millikin, Johan Munk, and Ian Zerny. *Defunctionalized Interpreters for Call-by-Need Evaluation*, pages 240–256. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[Fre16] Jonas Frey. Classical Realizability in the CPS Target Language. *Electronic Notes in Theoretical Computer Science*, 325(Supplement C):111 – 126, 2016. The Thirty-second Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXII).

[GM16a] Mauricio Guillermo and Alexandre Miquel. Specifying Peirce's law in classical realizability. *Mathematical Structures in Computer Science*, 26(7):1269–1303, 2016.

[GM16b] Mauricio Guillermo and Étienne Miquey. Classical realizability and arithmetical formulæ. *Mathematical Structures in Computer Science*, page 1–40, 2016.

[Gri90] Timothy Griffin. A formulae-as-type notion of control. In *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '90, pages 47–58, New York, NY, USA, 1990. ACM.

[Kri03] Jean-Louis Krivine. Dependent choice, 'quote' and the clock. *Th. Comp. Sc.*, 308:259–276, 2003.

[Kri04] Jean-Louis Krivine. A call-by-name lambda-calculus machine. In *Higher Order and Symbolic Computation*, 2004.

[Kri12] Jean-Louis Krivine. Realizability algebras II : new models of ZF + DC. *Logical Methods in Computer Science*, 8(1):10, February 2012. 28 p.

[Kri21] Jean-Louis Krivine. A program for the full axiom of choice. *Logical Methods in Computer Science*, Volume 17, Issue 3, September 2021.

[Lep16] Rodolphe Lepigre. A classical realizability model for a semantical value restriction. In Peter Thiemann, editor, *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016,*

Proceedings, volume 9632 of *Lecture Notes in Computer Science*, pages 476–502. Springer, 2016.

[Lep17] Rodolphe Lepigre. *Semantics and Implementation of an Extension of ML for Proving Programs. (Sémantique et Implantation d'une Extension de ML pour la Preuve de Programmes)*. PhD thesis, Grenoble Alpes University, France, 2017.

[Miq11a] Alexandre Miquel. Existential witness extraction in classical realizability and via a negative translation. *Logical Methods in Computer Science*, 7(2):188–202, 2011.

[Miq11b] Alexandre Miquel. Forcing as a program transformation. In *Proceedings of the 2011 IEEE 26th Annual Symposium on Logic in Computer Science*, LICS '11, page 197–206, USA, 2011. IEEE Computer Society.

[Miq17] Étienne Miquey. *Classical realizability and side-effects*. Ph.D. thesis, Université Paris Diderot ; Universidad de la República, Uruguay, November 2017.

[Miq18] Étienne Miquey. A sequent calculus with dependent types for classical arithmetic. In *LICS 2018*, pages 720–729. ACM, 2018.

[Miq20] Étienne Miquey. Revisiting the Duality of Computation: An Algebraic Analysis of Classical Realizability Models. In Maribel Fernández and Anca Muscholl, editors, *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*, volume 152 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:18, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[MM09] Guillaume Munch-Maccagnoni. Focalisation and Classical Realisability. In Erich Grädel and Reinhard Kahle, editors, *Computer Science Logic '09*, volume 5771 of *Lecture Notes in Computer Science*, pages 409–423. Springer, Heidelberg, 2009.

[MM13] Guillaume Munch-Maccagnoni. *Syntax and Models of a non-Associative Composition of Programs and Proofs*. PhD thesis, Univ. Paris Diderot, 2013.

[Mur90] Chetan Murthy. *Extracting constructive content from classical proofs*. Ph.D. thesis, Cornell University, 1990.

[OS08] P. Oliva and T. Streicher. On Krivine's realizability interpretation of classical second-order arithmetic. *Fundam. Inform.*, 84(2):207–220, 2008.

[Par97] M. Parigot. Proofs of strong normalisation for second order classical natural deduction. *J. Symb. Log.*, 62(4):1461–1479, 1997.

[Pit02] Andrew M. Pitts. Tripos theory in retrospect. *Mathematical Structures in Computer Science*, 12(3):265–279, 2002.

[Plo75] Gordon D. Plotkin. Call-by-name, call-by-value and the lambda-calculus. *Theor. Comput. Sci.*, 1(2):125–159, 1975.

[Rie14] Lionel Rieg. *On Forcing and Classical Realizability*. Theses, Ecole normale supérieure de lyon - ENS LYON, June 2014.

[SF93] Amr Sabry and Matthias Felleisen. Reasoning about programs in continuation-passing style. *Lisp and Symbolic Computation*, 6(3-4):289–360, 1993.

[SS75] Gerald J. Sussman and Guy L. Steele, Jr. An interpreter for extended lambda calculus. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1975.

[vO08] Jaap van Oosten. *Realizability: an introduction to its categorical side*, volume 152 of *Studies in Logic and the Foundations of Mathematics*. Elsevier B. V., Amsterdam, 2008.