Classical Extraction in Continuation Models

Valentin Blot

Classical Extraction in Continuation Models*

Valentin Blot

Department of Computer Science, University of Bath, Bath, United Kingdom
v.blot@bath.ac.uk

Abstract

We use the control features of continuation models to interpret proofs in first-order classical theories. This interpretation is suitable for extracting algorithms from proofs of Π^0_2 formulas. It is fundamentally different from the usual direct interpretation, which is shown to be equivalent to Friedman's trick. The main difference is that atomic formulas and natural numbers are interpreted as distinct objects. Nevertheless, the control features inherent to the continuation models permit extraction using a special "channel" on which the extracted value is transmitted at toplevel without unfolding the recursive calls. We prove that the technique fails in Scott domains, but succeeds in the refined setting of Laird's bistable bicpos, as well as in game semantics.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages: Denotational semantics, F.4.1 Mathematical Logic

Keywords and phrases Extraction, Classical Logic, Control Operators, Game Semantics

Digital Object Identifier 10.4230/LIPIcs.FSCD.2016.13

1 Introduction

Game semantics appeared simultaneously in [13, 21, 2] and provides precise models for PCF. These seminal works started a new line of research which led among other things to models of languages with higher-order references [1] and control operators [18]. These interesting features were obtained by removing some requirements of the original model: innocence for references, and well-bracketing for control operators. This interesting property means that the model is at first a model of a language with side effects, that we can then constrain to eliminate non-functional behaviors. In this work, we are interested into giving computational content to formulas through realizability, so it is an interesting feature to have an expressive language in which we can write simple realizers for complicated formulas.

Realizability is a way to relate programs and formulas invented by Kleene [14]. To each formula A in a given language $\mathcal{L}$ we associate a set $|A|$ of realizers in a given programming language $\mathcal{P}$. The realizability interpretation $|\_|$ is usually defined by induction on the formula, for example $M \in |A \Rightarrow B|$ is typically defined as: for all $N \in |A|$, $M\,N \in |B|$, where $M\,N$ is the application of the argument N to the program M in $\mathcal{P}$. The property of adequacy is obtained by the definition of a theory for $\mathcal{L}$ which is correct with respect to the realizability interpretation. Adequacy allows the mapping of any proof π of a formula A in the chosen theory to some $[\pi] \in |A|$. Finally, if the realizability interpretation is sufficiently well behaved, then a realizer M of a formula $\forall x\, \exists y\, A\{x, y\}$ is such that for any element a in the model, $A\{a, M\,a\}$ holds. This property combined with adequacy gives extraction: from a proof of a formula A one can obtain a program M such that $A\{a, M\,a\}$ holds for any element a.

* Research supported by the UK EPSRC grant EP/K037633/1.

In the context of classical logic, there are mainly two extraction methods. The first one relies on Gödel's negative translation which maps a proof of $\forall x\, \exists y\, P\,\{x, y\}$ (where P is an atomic predicate) in classical arithmetic to a proof of $\forall x\, \neg\forall y\, \neg P\,\{x, y\}$ in intuitionistic arithmetic. Then Friedman's trick [8] replaces $\bot$ with $\exists y\, P\,\{x, y\}$, to obtain an intuitionistic proof of $\forall x\, \exists y\, P\,\{x, y\}$. This turns any extraction method in intuitionistic arithmetic into an extraction method for Π^0_2 formulas in classical arithmetic. Refinements of this technique have been studied in [5], and bar recursive interpretations of the negative translation of the axiom of choice in this setting have been given in [4, 6].

The second method uses control operators [9] like scheme's call/cc. If $\mathcal{P}$ has control features, then adequacy can hold for a classical theory. Extraction with this method is obtained by taking a non-empty set of realizers of $\bot$. The first realizability models for classical logic with control operators were built by Krivine [16] and use an untyped λ-calculus extended with call/cc as programming language, and second order Peano arithmetic as logical theory. They have also been later extended with the axiom of dependent choice, by the addition of particular instructions to the language of realizers [15]. Krivine's realizability has also been related to Friedman's A-translation in [22, 20], and several extraction methods in this setting have been studied [20, 24].

In this work we define a variant of the second method and introduce control features in $\mathcal{P}$ by working in simply-typed $\lambda\mu$-calculus [23]. Precisely, we take $\mathcal{P}$ to be a model of $\lambda\mu$-calculus, that is, a category of continuations [12]. Working in a model rather than in a language can be interesting when one wants features that fit more naturally in a model than in the syntax. An example is the realization of the axiom of choice using the bar recursion operator [7], which requires the existence of all the first-class functions in $\mathcal{P}$. Taking the game semantics model for $\mathcal{P}$ also gives access to references in the realizers.

In Section 2 we first fix the logical framework, that is first-order logic. We then describe the mapping of intuitionistic (resp. classical) proofs to λ-calculus (resp. $\lambda\mu$-calculus) and the realizability interpretation. We describe Friedman's trick which turns extraction for intuitionistic theories into extraction for classical theories, then we describe how control operators can be used to interpret directly classical logic, and finally we explain why the two methods are equivalent when working in a model. In Section 3 we present our method of extraction for classical logic in a category of continuations and explain how it provides a simpler interpretation. Finally we explain why this technique fails in models based on Scott domains, but works in the model of unbracketed Hyland-Ong-style game semantics.

2 Friedman's trick and direct interpretation

In this section, we fix the logical system we will be working with. Then we present on one hand the indirect interpretation through negative translation and Friedman's trick and on the other hand the direct interpretation with control operators. Finally we explain why these two interpretations are the same when we work in a model.

2.1 The logical system

The logical systems under consideration in this work are first-order minimal logic, $\mathfrak{M}$, and first-order classical logic, $\mathfrak{C}$. $\mathfrak{C}$ is simply an extension of $\mathfrak{M}$ with double-negation elimination. The common syntax of first-order terms and formulas of $\mathfrak{M}$ and $\mathfrak{C}$ is the following:

$$t, u \;::=\; x \mid \mathsf{f}\left(\vec{t}\right) \qquad A, B \;::=\; P\left(\vec{t}\right) \mid \bot \mid A \Rightarrow B \mid A \wedge B \mid \forall x\, A \mid A \vee B \mid \exists x\, A$$

$$\frac{}{\Gamma, p : A \vdash p : A} \qquad \frac{\Gamma, p : A \vdash M : B}{\Gamma \vdash \lambda p.M : A \Rightarrow B} \qquad \frac{\Gamma \vdash M : B \Rightarrow A \qquad \Gamma \vdash N : B}{\Gamma \vdash MN : A}$$

$$\frac{\Gamma \vdash M_1 : A_1 \qquad \Gamma \vdash M_2 : A_2}{\Gamma \vdash \langle M_1, M_2 \rangle : A_1 \wedge A_2} \qquad \frac{\Gamma \vdash M : A_1 \wedge A_2}{\Gamma \vdash \pi_i\, M : A_i}$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \Lambda x.M : \forall x\, A}\ (x \notin \mathrm{FV}(\Gamma)) \qquad \frac{\Gamma \vdash M : \forall x\, A}{\Gamma \vdash M\,\lfloor t \rfloor : A\{t/x\}}$$

$$\frac{\Gamma \vdash M : A_i}{\Gamma \vdash \mathtt{in}_i\, M : A_1 \vee A_2} \qquad \frac{\Gamma \vdash M : A_1 \vee A_2 \qquad \Gamma, p : A_1 \vdash N_1 : B \qquad \Gamma, p : A_2 \vdash N_2 : B}{\Gamma \vdash \mathtt{case}\, M\, [\mathtt{in}_1\, p \mapsto N_1, \mathtt{in}_2\, p \mapsto N_2] : B}$$

$$\frac{\Gamma \vdash M : A\{t/x\}}{\Gamma \vdash \mathtt{ex}\, [t, M] : \exists x\, A} \qquad \frac{\Gamma \vdash M : \exists x\, A \qquad \Gamma, p : A \vdash N : B}{\Gamma \vdash \mathtt{dest}\, M\, \mathtt{as}\, \mathtt{ex}\, [x, p]\ \mathtt{in}\, N : B}\ (x \notin \mathrm{FV}(\Gamma, B))$$

Figure 1 Rules for $\mathfrak{M}$.

where f (resp. P) ranges over a set of function (resp. predicate) symbols given with their arity. Quantification has precedence over other connectives and negation is encoded as usual: $\neg\, A \triangleq A \Rightarrow \bot$. The proofs and proof terms of $\mathfrak{M}$ are defined by the rules of Figure 1, where Γ stands for a sequence of pairs $p : A$ where p is a proof variable, for which we allow implicit re-ordering.

For $\mathfrak{C}$, we simply add the rules:

$$\frac{}{\Gamma \vdash \mathsf{dne} : \neg\, \neg\, A \Rightarrow A}$$

Provability in $\mathfrak{M}$ (resp. $\mathfrak{C}$) will be denoted $\vDash_{\mathfrak{m}}$ (resp. $\vDash_{\mathfrak{c}}$). We will sometimes write $\Gamma \vDash_{\mathfrak{m}} A$ (resp. $\Gamma \vDash_{\mathfrak{c}} A$) if we're not interested in the proof term, and we may write $X \vDash_{\mathfrak{m}} A$ (resp. $X \vDash_{\mathfrak{c}} A$) for some set X, which means $\Gamma \vDash_{\mathfrak{m}} A$ (resp. $\Gamma \vDash_{\mathfrak{c}} A$) for some finite sequence Γ of formulas of X. Since contraction is derivable and weakening is admissible, we will use these implicitly.

2.2 Mapping proofs to terms

We will use two programming languages that share a common ground to interpret $\mathfrak{M}$ and $\mathfrak{C}$. This common ground is simply-typed λ-calculus with products and unit types, and one base type ι. The set of variables of this λ-calculus is the union of first-order variables $x, y, \ldots$ and proof variables $p, q, \ldots$, this union being ranged over with $e, f, \ldots$. There is also one constant f of type $\iota \to \ldots \to \iota \to \iota$ ($n+1$ times) for each first-order constant f of arity n. The syntax of types and terms of this common ground is as follows:

$$T, U \ ::= \ \iota \mid T \to U \mid 1 \mid T \times U \qquad M, N \ ::= \ \mathsf{f} \mid \lambda e.M \mid M\, N \mid \langle\rangle \mid \langle M, N \rangle \mid \pi_1\, M \mid \pi_2\, M$$

From this common ground, the first language we consider is λ^+, in which we will interpret $\mathfrak{M}$. λ^+ is obtained by adding sum types to the common ground:

$$T, U \ ::= \ \ldots \mid T + U \qquad M, N \ ::= \ \ldots \mid \mathsf{in}_1\, M \mid \mathsf{in}_2\, M \mid \mathsf{case}\, M\, \{\mathsf{in}_1\, e \mapsto N_1 \mid \mathsf{in}_2\, e \mapsto N_2\}$$

The second language is $\lambda\mu$, in which we will interpret $\mathfrak{C}$, and which is obtained by adding to the common ground an empty type, control features and μ-variables:

$$T, U \ ::= \ \ldots \mid \diamond \qquad M, N \ ::= \ \ldots \mid \mu\alpha.M \mid [\alpha]\, M$$

$$(A \Rightarrow B)^* \triangleq A^* \to B^* \qquad (A \wedge B)^* \triangleq A^* \times B^* \qquad (\forall x\, A)^* \triangleq \iota \to A^*$$

$$\bot^{*m} \triangleq 1 \qquad P\left(\vec{t}\right)^{*m} \triangleq 1 \qquad (A \vee B)^{*m} \triangleq A^{*m} + B^{*m} \qquad (\exists x\, A)^{*m} \triangleq \iota \times A^{*m}$$

$$\bot^{*c} \triangleq \diamond \qquad P\left(\vec{t}\right)^{*c} \triangleq \diamond \to \diamond \qquad (A \vee B)^{*c} \triangleq (\neg\,(\neg\, A \wedge \neg\, B))^{*c} \qquad (\exists x\, A)^{*c} \triangleq (\neg\, \forall x \neg\, A)^{*c}$$

Figure 2 Mapping formulas to types.

$$p^* \triangleq p \qquad (\lambda p.M)^* \triangleq \lambda p.M^* \qquad (M\ N)^* \triangleq M^*\ N^* \qquad (\Lambda x.M)^* \triangleq \lambda x.M^*$$

$$(M\ \lfloor t \rfloor)^* \triangleq M^*\ t^* \qquad \langle M, N\rangle^* \triangleq \langle M^*, N^*\rangle \qquad (\pi_i\ M)^* \triangleq \pi_i\ M^* \qquad \mathsf{dne}^{*c} \triangleq \lambda p.\mu\alpha.p\,(\lambda q.\,[\alpha]\,q)$$

$$(\mathtt{in}_i\ M)^{*m} \triangleq \mathsf{in}_i\ M^{*m} \qquad (\mathtt{ex}\,[t, M])^{*m} \triangleq \langle t^*, M^{*m}\rangle$$

$$(\mathtt{in}_i\ M)^{*c} \triangleq \lambda p.\pi_i\ p\ M^{*c} \qquad (\mathtt{ex}\,[t, M])^{*c} \triangleq \lambda p.p\ t^*\ M^{*c}$$

$$(\mathtt{case}\ M\ [\mathtt{in}_1\ p \mapsto N_1, \mathtt{in}_2\ p \mapsto N_2])^{*m} \triangleq \mathtt{case}\ M^{*m}\ [\mathtt{in}_1\ p \mapsto {N_1}^{*m}, \mathtt{in}_2\ p \mapsto {N_2}^{*m}]$$

$$(\mathtt{case}\ M\ [\mathtt{in}_1\ p \mapsto N_1, \mathtt{in}_2\ p \mapsto N_2])^{*c} \triangleq \mathsf{dne}^{*c}\ (\lambda q.M^{*c}\ \langle \lambda p.q\ {N_1}^{*c}, \lambda p.q\ {N_2}^{*c}\rangle)$$

$$(\mathtt{dest}\ M\ \mathtt{as}\ \mathtt{ex}\,[x, p]\ \mathtt{in}\ N)^{*m} \triangleq (\lambda xp.N^{*m})\ (\pi_1\ M^{*m})\ (\pi_2\ M^{*m})$$

$$(\mathtt{dest}\ M\ \mathtt{as}\ \mathtt{ex}\,[x, p]\ \mathtt{in}\ N)^{*c} \triangleq \mathsf{dne}^{*c}\ (\lambda q.M^{*c}\ (\lambda xp.q\ N^{*c}))$$

Figure 3 Mapping proof terms to $\lambda^+/\lambda\mu$-terms.

We work in a call-by-name setting and refer to [12, 25] for the typing rules and equational theory of $\lambda\mu$.

Note that terms of λ^+ and $\lambda\mu$ are not the same as the proof terms of Figure 1. In particular, proof terms have no operational or denotational semantics, but provide a convenient way to manipulate proofs. Proofs terms will now be mapped to terms of λ^+ or $\lambda\mu$. First, we map each formula A to a type A^* of either λ^+ or $\lambda\mu$. When the interpretation is different in $\mathfrak{M}$ and $\mathfrak{C}$ we will write A^{*m} or A^{*c}. The mapping is defined in Figure 2.

In the case of $\mathfrak{C}$, since positive connectives are known to raise issues on the computational level when combined with classical logic [11] (we also give a concrete example at the end of Section 2.3) we use a negative encoding of connectives $\vee$ and $\exists$. First-order terms (which are common to $\mathfrak{M}$ and $\mathfrak{C}$) are mapped to terms of $\lambda^+ \cap \lambda\mu$ by $x^* \triangleq x$ and $(\mathsf{f}\,(t_1, \ldots, t_n))^* \triangleq \mathsf{f}\,{t_1}^*\ \ldots\ {t_n}^*$, and proof terms of $\mathfrak{M}$ (resp. $\mathfrak{C}$) are mapped to terms of λ^+ (resp. $\lambda\mu$) as in Figure 3. The elimination of the encoded connectives in $\mathfrak{C}$ is made possible by classical logic and control features of $\lambda\mu$.

A proof term $p_1 : A_1, \ldots, p_n : A_n \vDash_m M : A$ (resp. $p_1 : A_1, \ldots, p_n : A_n \vDash_c M : A$) with $\mathrm{FV}\,(A_1, \ldots, A_n, A) = \{x_1, \ldots, x_m\}$ is therefore mapped to a term $M^{*m} : A^{*m}$ (resp. $M^{*c} : A^{*c}$) of λ^+ (resp. $\lambda\mu$) with free variables among $p_1 : {A_1}^*, \ldots, p_n : {A_n}^*, x_1 : \iota, \ldots, x_m : \iota$, and no free μ-variable (in the case of $\mathfrak{C}$).

$$\left|P\left(\vec{t}\right)\right|_m \triangleq \begin{cases} \{\langle\rangle\} & \text{if } \vec{t}^* \in \{\!|P|\!\} \\ \emptyset & \text{otherwise} \end{cases} \qquad |A_1 \wedge A_2| \triangleq \{\phi |\ \pi_1\,\phi \in |A_1| \text{ and } \pi_2\,\phi \in |A_2|\}$$

$$\left|P\left(\vec{t}\right)\right|_c \triangleq \begin{cases} |\bot \Rightarrow \bot|_c & \text{if } \vec{t}^* \in \{\!|P|\!\} \\ \{\phi |\ \text{if } \psi \in \mathcal{C}\left(1, \diamond\right) \text{ then } \phi\,\psi \in |\bot|_c\} & \text{otherwise} \end{cases}$$

$$|A \Rightarrow B| \triangleq \{\phi |\ \text{if } \psi \in |A| \text{ then } \phi\,\psi \in |B|\} \qquad |\forall x\, A| \triangleq \{\phi |\ \text{if } \psi \in \langle\!|\iota|\!\rangle \text{ then } \phi\,\psi \in |A\{\psi/x\}|\}$$

$$|A_1 \vee A_2|_m \triangleq \{\mathsf{in}_1\,\phi |\ \phi \in |A_1|_m\} \cup \{\mathsf{in}_2\,\phi |\ \phi \in |A_2|_m\} \qquad |A_1 \vee A_2|_c \triangleq |\neg\left(\neg A_1 \wedge \neg A_2\right)|_c$$

$$|\exists x\, A|_m \triangleq \{\phi |\ \pi_1\,\phi \in \langle\!|\iota|\!\rangle \text{ and } \pi_2\,\phi \in |A\{\pi_1\,\phi/x\}|_m\} \qquad |\exists x\, A|_c \triangleq |\neg \forall x \neg A|_c$$

Figure 4 The realizability interpretation.

2.3 Typed realizability

We now describe the realizability semantics of $\mathfrak{M}$ and $\mathfrak{C}$. Let $\mathcal{C}$ be a cartesian closed category. We also suppose that $\mathcal{C}$ has coproducts in the case of $\mathfrak{M}$, or that it is a category of continuations (see [12, 25] and Section 2.6) in the case of $\mathfrak{C}$. Every type of λ^+ (for $\mathfrak{M}$) or $\lambda\mu$ (for $\mathfrak{C}$) is interpreted as an object of $\mathcal{C}$ in the standard way, from a chosen interpretation of ι. We will therefore consider types as objects of $\mathcal{C}$. Similarly, we suppose given a morphism for each constant f of $\lambda^+ \cap \lambda\mu$ (remember from Section 2.2 that there is one such constant for each first-order constant f), so that every term of λ^+ or $\lambda\mu$ is interpreted as a morphism in the homset corresponding to its context. We will therefore also consider terms as morphisms in $\mathcal{C}$, and we will use the syntax of λ^+ and $\lambda\mu$ to manipulate morphisms with domain 1. In order to define the realizability values of the formulas, we first fix a set $\langle\!|\iota|\!\rangle \subseteq \mathcal{C}\left(1, \iota\right)$ such that for every closed first-order term t, $t^* \in \langle\!|\iota|\!\rangle$. This set represents the elements of the model on which we quantify. Typically, if $\mathcal{C}\left(1, \iota\right)$ is a domain of natural numbers with a bottom element, $\langle\!|\iota|\!\rangle$ would be the set of natural numbers (the domain minus the bottom element). We also fix for every predicate P of arity n a set $\{\!|P|\!\} \subseteq \langle\!|\iota|\!\rangle^n$ of n-tuples satisfying the predicate. The set of realizers of a closed formula with parameters in $\langle\!|\iota|\!\rangle$ is then a set of morphisms $|A| \subseteq \mathcal{C}\left(1, A^*\right)$. When the interpretation is different in $\mathfrak{M}$ and $\mathfrak{C}$ we will again write $|A|_m$ or $|A|_c$. We take $|\bot|_m = \emptyset$, but $|\bot|_c$ is a parameter of the realizability interpretation. The interpretation is defined in Figure 4.

It is worth noting here that taking $|\bot|_c = \emptyset$ gives a degenerated model, in which $|A|_c$ is either empty or the full homset $\mathcal{C}\left(1, A^*\right)$. Indeed, suppose that $|A|_c \neq \emptyset$, and let $\phi \in \mathcal{C}\left(1, A^*\right)$. First, since $|A|_c \neq \emptyset$ and $|\bot|_c = \emptyset$, we have $|\neg A|_c = \emptyset$, and therefore $|\neg\neg A|_c = \mathcal{C}\left(1, \left(\neg\neg A\right)^*\right)$. But since $\lambda p.p\,\phi \in \mathcal{C}\left(1, \left(\neg\neg A\right)^*\right)$, we get $\lambda p.p\,\phi \in |\neg\neg A|_c$, and then $\mathsf{dne}^*\left(\lambda p.p\,\phi\right) = \phi \in |A|_c$ by adequacy (Lemma 1 below). Therefore, if we want to get computational content from classical proofs, we need to consider $|\bot|_c \neq \emptyset$.

We can already state an adequacy lemma for $\mathfrak{M}$ and $\mathfrak{C}$:

▶ **Lemma 1** (Adequacy). *Suppose $\vec{p} : \vec{A} \vdash M : B$ is a proof in $\mathfrak{M}$ or $\mathfrak{C}$ and write $FV\left(\vec{A}, B\right) = \vec{x} = \{x_1, \ldots, x_n\}$. Then for any $\vec{\phi}$ in $\langle\!|\iota|\!\rangle^n$ and any realizers $\vec{\psi} \in \left|\vec{A}\left\{\vec{\phi}/\vec{x}\right\}\right|$, we have $M^*\left\{\vec{\phi}/\vec{x}, \vec{\psi}/\vec{p}\right\} \in \left|B\left\{\vec{\phi}/\vec{x}\right\}\right|$.*

Proof. By induction on M. An interesting case is that of $\mathsf{dne} : \neg\neg A \Rightarrow A$ in the case of $\mathfrak{C}$ (we will write dne_A in this proof), which is proved by induction on A. If A is atomic, then it

is a case analysis. In the other cases, it follows from the observation that in $\mathcal{C}$:

$$\mathsf{dne}_{A\Rightarrow B}{}^{*_c} = \lambda pq.\mathsf{dne}_B{}^{*_c}\left(\lambda r.p\left(\lambda s.r\left(s\,q\right)\right)\right) \qquad \mathsf{dne}_{\forall x\,A}{}^{*_c} = \lambda px.\mathsf{dne}_A{}^{*_c}\left(\lambda q.p\left(\lambda r.q\left(r\,x\right)\right)\right)$$

$$\mathsf{dne}_{A\wedge B}{}^{*_c} = \lambda p.\left\langle \mathsf{dne}_A{}^{*_c}\left(\lambda q.p\left(\lambda r.q\left(\pi_1\,r\right)\right)\right), \mathsf{dne}_A{}^{*_c}\left(\lambda q.p\left(\lambda r.q\left(\pi_2\,r\right)\right)\right)\right\rangle$$ ◀

We can now explain by reformulating the ideas of [11] why we encoded $\exists x\,A$ in $\mathfrak{C}$ instead of interpreting it as in $\mathfrak{M}$. Suppose there is a binary predicate $P\left(t,u\right)$ and some $\phi,\psi_1,\psi_2 \in \left|\!\left|\iota\right|\!\right|$ such that $\left(\phi,\psi_1\right) \notin \left\{\!\left|P\right|\!\right\}$ and $\left(\phi,\psi_2\right) \in \left\{\!\left|P\right|\!\right\}$ (for example, P may be equality and $\psi_1 \neq \phi = \psi_2$). Consider the proof:

$$r : P\left(x,y_2\right) \vdash_{\mathrm{c}} \mathsf{dne}\left(\lambda p.p\,\mathsf{ex}\left[y_1,\mathsf{dne}\left(\lambda q.p\,\mathsf{ex}\left[y_2,r\right]\right)\right]\right) : \exists y\,P\left(x,y\right)$$

If $\left(\mathsf{ex}\left[\_,\_\right]\right)^{*_c}$ was a pair like in $\mathfrak{M}$, then this proof would be translated to a term of $\lambda\mu$ which is equal in $\mathcal{C}$ to $\mu\alpha.\left[\alpha\right]\left\langle y_1,\mu\beta.\left[\alpha\right]\left\langle y_2,r\right\rangle\right\rangle$. Adequacy would imply that for any $\xi \in \left|P\left(\phi,\psi_2\right)\right|_c$:

$$\zeta = \mu\alpha.\left[\alpha\right]\left\langle \psi_1,\mu\beta.\left[\alpha\right]\left\langle \psi_2,\xi\right\rangle\right\rangle \in \left|\exists y\,P\left(\phi,y\right)\right|_c$$

so $\pi_2\,\zeta \in \left|P\left(\phi,\pi_1\,\zeta\right)\right|_c$ (if $\left|\exists\,y\,\_\right|_c$ was as in $\mathfrak{M}$). But since $\pi_1\,\zeta = \psi_1$ and $\pi_2\,\zeta = \xi$, this would mean that $\xi \in \left|P\left(\phi,\psi_1\right)\right|_c$. The other inclusion being easy, we would get $\left|P\left(\phi,\psi_1\right)\right|_c = \left|P\left(\phi,\psi_2\right)\right|_c$, that is, $\left|P\left(\phi,\psi\right)\right|_c$ would not depend on whether $\left(\phi,\psi\right) \in \left\{\!\left|P\right|\!\right\}$ or not and the model would be degenerated.

2.4 Extraction for minimal logic

We fix now a theory $\mathcal{A}x$, that is, a set of closed formulas, or axioms. We also suppose that for each $A \in \mathcal{A}x$ we have some realizer $\zeta_A \in \left|A\right|_m$. Extraction is an immediate consequence of adequacy:

▶ **Theorem 2** (Extraction). *Let $\vec{p_A} : \vec{A} \Vdash_m M : \forall x\,\exists y\,B$ be a proof, where $\vec{A} \subseteq \mathcal{A}x$ and $FV(B) \subseteq \{x;y\}$. From that proof we can extract some $\phi \in \mathcal{C}\left(1,\iota\to\iota\times B^{*_m}\right)$ such that for any $\psi \in \left|\!\left|\iota\right|\!\right|$, $\pi_1\left(\phi\,\psi\right) \in \left|\!\left|\iota\right|\!\right|$ and:*

$$\pi_2\left(\phi\,\psi\right) \in \left|B\left\{\psi/x,\pi_1\left(\phi\,\psi\right)/y\right\}\right|_m$$

Proof. Immediate from adequacy, with $\phi = M^{*_m}\left\{\vec{\zeta_A}/\vec{p_A}\right\}$. ◀

Let's look at the particular example of $\mathcal{A}x$ being the set of axioms of arithmetic. Suppose we have realizers of these axioms in $\mathcal{C}$ (which usually means that $\mathcal{C}$ is a model of Gödel's system T) and $\left|\!\left|\iota\right|\!\right|$ is isomorphic to $\mathbb{N}$ with the constants interpreted accordingly. The extraction result tells us that from a proof of $\forall x\,\exists y\left(t=0\right)$ in $\mathcal{A}x$ we can extract an element:

$$\phi' = \lambda x.\pi_1\left(\phi\,x\right) \in \mathcal{C}\left(1,\iota\to\iota\right)$$

such that for any $n \in \mathbb{N}$, $\left|t\left\{n/x,\phi'\,n/y\right\} = 0\right|_m \neq \emptyset$, and so $\left(n,\phi'\,n\right) \in \left\{\!\left|=\right|\!\right\}$ by definition of the realizability interpretation in $\mathfrak{M}$. If moreover $\left\{\!\left|=\right|\!\right\}$ is equality on $\left|\!\left|\iota\right|\!\right| \simeq \mathbb{N}$ then it tells us that $t^*\left\{n/x,\phi'\,n/y\right\} = 0$ in $\mathbb{N}$.

$$p^R \triangleq p \qquad (\lambda p.M)^R \triangleq \lambda q.\left(\lambda p.M^R\right)(\pi_1\, q)\,(\pi_2\, q) \qquad (M\, N)^R \triangleq \lambda p.M^R\left\langle N^R, p\right\rangle$$

$$\langle M_1, M_2\rangle^R \triangleq \lambda p.\mathtt{case}\, p\left[\mathtt{in}_1\, q \mapsto M_1^R\, q, \mathtt{in}_2\, q \mapsto M_2^R\, q\right] \qquad (\pi_i\, M)^R \triangleq \lambda p.M^R\,(\mathtt{in}_i\, p)$$

$$(\Lambda x.M)^R \triangleq \lambda p.\mathtt{dest}\, p\, \mathtt{as}\, \mathtt{ex}\,[x,q]\ \mathtt{in}\, M^R\, q \qquad (M\ \lfloor t\rfloor)^R \triangleq \lambda p.M^R \mathtt{ex}\,[t,p]$$

$$(\mathtt{in}_i\, M)^R \triangleq \lambda p.\pi_i\, p\, M^R \qquad (\mathtt{ex}\,[t,M])^R \triangleq \lambda p.p\,\mathtt{ex}\left[t, M^R\right]$$

$$(\mathtt{case}\, M\,[\mathtt{in}_1\, p \mapsto N_1, \mathtt{in}_2\, p \mapsto N_2])^R \triangleq \lambda q.M^R\left\langle \lambda p.N_1^R\, q, \lambda p.N_2^R\, q\right\rangle$$

$$(\mathtt{dest}\, M\, \mathtt{as}\, \mathtt{ex}\,[x,p]\ \mathtt{in}\, N)^R \triangleq \lambda q.M^R\left(\lambda r.\mathtt{dest}\, r\, \mathtt{as}\, \mathtt{ex}\,[x,p]\ \mathtt{in}\, N^R\, q\right)$$

$$\mathsf{dne}^R \triangleq \lambda p.\pi_1\, p\left\langle \lambda q.\pi_1\, q\,(\pi_2\, p)\,, \lambda r.r\right\rangle$$

Figure 5 Lafont-Reus-Streicher translation of proofs from $\mathfrak{C}$ to $\mathfrak{M}$.

2.5 Negative translation and Friedman's trick

In this section we explain the indirect interpretation of classical theories via negative translation and Friedman's trick. Negative translation has been defined by Gödel to study the relationship between intuitionistic and classical provability. A formula A is translated to $A^\neg$ by inductively replacing every positive formula by a (classically) equivalent negative one. Negative translation turns classical provability into intuitionistic provability, in the sense that if $A_1, \ldots, A_n \vdash_{\mathfrak{c}} B$, then $A_1^\neg, \ldots, A_n^\neg \vdash_{\mathfrak{m}} B^\neg$.

Adapting Friedman's original translation [8], we can parameterize negative translation by an arbitrary formula R playing the role of $\bot$. R will be instantiated later on with a carefully chosen formula. This is known as Friedman's trick and can be used to prove conservativity of classical arithmetic over intuitionistic arithmetic for Π_2^0 formulas. Here we use a slightly different version: Lafont-Reus-Streicher (LRS) translation [17]. This version makes explicit the connection with the direct interpretation that we will present in Section 2.6.

For each formula A we define an intermediate translation $A^{\overline{R}}$ in order to then define $A^R \equiv A^{\overline{R}} \Rightarrow R$. $A^{\overline{R}}$ and A^R are parameterized by the arbitrary formula R:

$$\left(P\left(\vec{t}\right)\right)^{\overline{R}} \triangleq P\left(\vec{t}\right) \Rightarrow R \qquad \bot^{\overline{R}} \triangleq \bot \Rightarrow \bot \qquad (\forall x\, A)^{\overline{R}} \triangleq \exists x A^{\overline{R}} \qquad (\exists x\, A)^{\overline{R}} \triangleq \exists x A^R \Rightarrow R$$

$$(A \Rightarrow B)^{\overline{R}} \triangleq A^R \wedge B^{\overline{R}} \qquad (A \wedge B)^{\overline{R}} \triangleq A^{\overline{R}} \vee B^{\overline{R}} \qquad (A \vee B)^{\overline{R}} \triangleq \left(A^R \Rightarrow R\right) \wedge \left(B^R \Rightarrow R\right)$$

This translation turns provability in classical logic into provability in minimal logic:

▶ **Lemma 3.** *If $\Gamma \vdash_{\mathfrak{c}} A$ then $\Gamma^R \vdash_{\mathfrak{m}} A^R$ (modulo α-conversion to avoid capture of the free variables of R).*

Proof. The translation $M \mapsto M^R$ on proof terms is given in Figure 5 ◀

We now show how we can transpose the extraction Theorem 2 into an extraction theorem for a classical theory through LRS translation. We fix $\mathcal{C}$ to be a cartesian closed category with coproducts as in Section 2.3, with a chosen object ι, a set $\langle\!|\iota|\!\rangle \subseteq \mathcal{C}(1,\iota)$, interpretations of the constants f and interpretations of the predicates $\{\!|P|\!\} \subseteq \langle\!|\iota|\!\rangle^n$. We also fix a theory $\mathcal{A}x$, and we suppose that for each $A \in \mathcal{A}x$ we have some realizer $\zeta_A \in |A|_m$. Extraction from classical proofs of Π_0^2 formulas is obtained through Friedman's trick combined with the extraction Theorem 2:

▶ **Theorem 4** (Extraction). *Suppose that for every $A \in \mathcal{A}x$ we have $\mathcal{A}x \vDash_m A^R$. From a proof of $\mathcal{A}x \vdash_c \forall x\, \exists y\, P\left(\vec{t}\right)$ where $FV\left(\vec{t}\right) \subseteq \{x; y\}$, we can extract a morphism $\phi \in \mathcal{C}\left(1, \iota \to \iota\right)$ such that for any $\psi \in \langle\!|\iota|\!\rangle$, $\phi\, \psi \in \langle\!|\iota|\!\rangle$ and:*

$$\vec{t}^{\,*}\left\{\psi/x, \phi\, \psi/y\right\} \in \{\!|P|\!\}$$

Proof. Elimination of the $\forall$ quantification gives $\mathcal{A}x \vdash_c \exists y\, P\left(\vec{t}\right)$, and LRS translation combined with the proofs $\mathcal{A}x \vDash_m A^R$ gives some proof term:

$$p_{\vec{A}} : \vec{A} \vDash_m M : \left(\exists y\, P\left(\vec{t}\right)\right)^R \equiv \left(\exists y\, \left(\left(P\left(\vec{t}\right) \Rightarrow R\right) \Rightarrow R\right) \Rightarrow R\right) \Rightarrow R$$

for some $\vec{A} \subseteq \mathcal{A}x$. We apply now Friedman's trick: take $R \equiv \exists\, y\, P\left(\vec{t}\right)$ (its only free variable is x so there is no capture of variables). Then we have:

$$p_{\vec{A}} : \vec{A} \vDash_m \Lambda x.M\left(\lambda p.\mathtt{dest}\, p\, \mathtt{as}\, \mathtt{ex}\left[y, q\right]\, \mathtt{in}\, q\left(\lambda r.\mathtt{ex}\left[y, r\right]\right)\right) : \forall x\, \exists\, y\, P\left(\vec{t}\right)$$

to which we apply Theorem 2 and get some ϕ_0 such that for any $\psi \in \langle\!|\iota|\!\rangle$:

$$\pi_2\left(\phi_0\, \psi\right) \in \left|P\left(\vec{t}\{\psi/x, \pi_1\left(\phi_0\, \psi\right)/y\}\right)\right|_m$$

so $\left|P\left(\vec{t}\{\psi/x, \pi_1\left(\phi_0\, \psi\right)/y\}\right)\right|_m \neq \emptyset$, and $\vec{t}^{\,*}\left\{\psi/x, \phi\, \psi/y\right\} \in \{\!|P|\!\}$ with $\phi = \lambda x.\pi_1\left(\phi_0\, x\right)$. ◀

We discuss now about the assumption that for every $A \in \mathcal{A}x$ we have $\mathcal{A}x \vDash_m A^R$.

In the case of arithmetic, since equality is decidable in minimal logic, all the axioms $A \in \mathcal{A}x$ but induction are such that $\mathcal{A}x \vDash_m A^R$. For induction it is even simpler since its translation is itself an instance of induction. Therefore, the extraction in minimal arithmetic presented in Section 2.4 can be turned into extraction for classical arithmetic.

This assumption however doesn't hold for every theory. For example, consider the axiom of dependent choice DC (that we can formulate in a multi-sorted version of first-order logic). DC fails to prove DC^R in minimal logic. However, DC^R is a consequence of $DNS + DC$ in minimal logic, where DNS is the double-negation shift:

$$\forall x\, \left(\left(A \Rightarrow R\right) \Rightarrow R\right) \Rightarrow \left(\forall x\, A \Rightarrow R\right) \Rightarrow R$$

where the sort of x is that of natural numbers. In a single-sorted setting, one can even show that DNS proves $A \Rightarrow A^R$ for any formula A in minimal logic. Historically, this technique has been used to give computational content to the axiom of choice in a classical setting, interpreting DNS intuitionistically with Spector's operator of bar recursion [4, 6].

2.6 Direct interpretation

Since Griffin's discovery [9], we can directly interpret proofs of classical logic in functional programming languages with control operators, as was done in Section 2.2. In order to interpret $\lambda\mu$ (in which classical proofs are mapped), we fix a category of continuations $\mathcal{C} = \diamond^{\mathcal{D}}$ (see [12, 25]). This means that $\mathcal{D}$ is a distributive category, $\diamond$ is an object of $\mathcal{D}$ such that all exponents $\diamond^X$ exist in $\mathcal{D}$, and $\diamond^{\mathcal{D}}$ is the full subcategory of $\mathcal{D}$ consisting of the objects $\diamond^X$. As suggested by the notation, $\diamond$ is the object interpreting the type $\diamond$ of $\lambda\mu$. In order to perform extraction we suppose $\iota = \diamond$ in $\mathcal{C}$. We fix a theory $\mathcal{A}x$ and we suppose that for each $A \in \mathcal{A}x$ we have some realizer $\zeta_A \in |A|_c$. Extraction requires a clever choice of the parameter $|\bot|_c$ of the realizability interpretation:

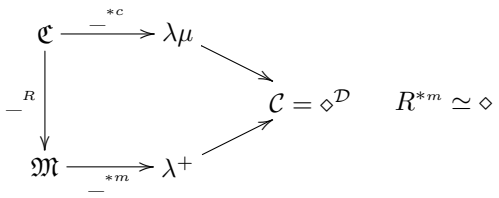


Figure 6 Correspondence of direct and indirect realizability interpretations.

▶ **Theorem 5** (Extraction). *From a proof of* $\mathcal{A}x \vdash_c \forall x \exists y\, P\left(\vec{t}\right)$ *where* $FV\left(\vec{t}\right) \subseteq \{x; y\}$*, we can extract a morphism* $\phi \in \mathcal{C}\left(1, \iota \to \iota\right)$ *such that for any* $\psi \in \langle\!|\iota|\!\rangle$*,* $\phi\,\psi \in \langle\!|\iota|\!\rangle$ *and:*

$$\vec{t}^{\,*}\left\{\psi/x, \phi\,\psi/y\right\} \in \{\!|P|\!\}$$

Proof. Write $\vec{p_A} : \vec{A} \vdash_c M : \forall x \exists y\, P\left(\vec{t}\right)$ where $\vec{A} \subseteq \mathcal{A}x$. By adequacy we have for any $\psi \in \langle\!|\iota|\!\rangle$:

$$M^{*_c}\left\{\vec{\zeta_A}/\vec{p_A}\right\}\psi \in \left|\exists y\, P\left(\vec{t}\{\psi/x\}\right)\right|_c = \left|\neg\forall y \neg P\left(\vec{t}\{\psi/x\}\right)\right|_c$$

Remember now that $|\bot|_c \subseteq \mathcal{C}\left(1, \diamond\right) = \mathcal{C}\left(1, \iota\right)$ is a parameter that we can choose freely. Take now:

$$|\bot|_c = \left\{\zeta \in \langle\!|\iota|\!\rangle \;\middle|\; \vec{t}^{\,*}\left\{\psi/x, \zeta/y\right\} \in \{\!|P|\!\}\right\}$$

By a simple disjunction of cases we prove that $\lambda yp.p\,y \in \left|\forall y \neg P\left(\vec{t}\{\psi/x\}\right)\right|_c$ for this choice of $|\bot|_c$. Note that this term is well-typed precisely because $\iota = \diamond$. The following morphism then has the required property:

$$\phi = \lambda x.M^{*_c}\left\{\vec{\zeta_A}/\vec{p_A}\right\} x\left(\lambda yp.p\,y\right) . \qquad \blacktriangleleft$$

2.7 Correspondence of the two methods

The indirect and direct methods presented in the previous two sections share some similarities, which we shall now make explicit. More precisely, we will prove that the diagram of Figure 6 commutes.

Fix a category of continuations $\mathcal{C} = \diamond^{\mathcal{D}}$. First, observe that the LRS translation of Section 2.5 is such that for any formula A, A^R belongs to a restricted syntax of formulas where implications are all of the form $B \Rightarrow R$. Moreover, A^R doesn't contain $\forall$, therefore, the simple type $\left(A^R\right)^{*_m}$ is itself in a restricted syntax where arrow types are all of the form $T \to R^*$. This means that $\mathcal{D}$ has enough structure to interpret the LRS translations of $\mathfrak{C}$-proofs if we suppose $R^{*_m} \simeq \diamond$ (remember that $\mathcal{D}$ has all exponentials $\diamond^X$). Finally, since $\left(A^R\right)^{*_m} = \left(A^{\overline{R}}\right)^{*_m} \to R^{*_m}$, LRS translations of $\mathfrak{C}$-proofs can be interpreted in the full subcategory $\mathcal{C} = \diamond^{\mathcal{D}}$. We claim now that this interpretation is the same as the direct interpretation of Section 2.6.

The following lemma, proved by induction on formulas and proofs, connects the two interpretations in $\mathcal{C}$ at the level of formulas, proof terms and realizers:

▶ **Lemma 6.** *If* $R^{*_m} \simeq \diamond$*, then:*

- *for any formula* A*,* $\left(A^R\right)^{*_m} \simeq A^{*_c}$ *in* $\mathcal{C}$
- *for any* $\mathfrak{C}$*-proof term* M*,* $\left(M^R\right)^{*_m}$ *and* M^{*_c} *are equal in* $\mathcal{C}$*, up to the previous isomorphism*

- *if* $|R|_m$ *and* $|\bot|_c$ *are equal up to the isomorphism* $R^{*m} \simeq \diamond$*, then for any formula* A*,* $\left|A^R\right|_m$ *is equal to* $|A|_c$*, up to the isomorphism* $\left(A^R\right)^{*m} \simeq A^{*c}$

To conclude this section, we show that in the extraction Theorems 4 and 5, we indeed have $R^{*m} \simeq \diamond$ and $|R|_m$ equal to $|\bot|_c$ up to this isomorphism. On one hand, in the extraction Theorem 4, we fixed $R \equiv \exists y\, P\left(\vec{t}\right)$, and therefore:

$$R^{*m} = \left(\exists y\, P\left(\vec{t}\right)\right)^{*m} = \iota \times 1 \simeq \iota$$

And on the other hand, in the extraction Theorem 5 we supposed that $\iota = \diamond$. Therefore we have indeed $R^{*m} \simeq \diamond$. Also, in Theorem 4 we have for any $\psi \in \langle\!|\iota|\!\rangle$:

$$|R\{\psi/x\}|_m = \left|\exists y\, P\left(\vec{t}\{\psi/x\}\right)\right|_m = \left\{\phi \middle|\ \pi_1\,\phi \in \langle\!|\iota|\!\rangle \text{ and } \pi_2\,\phi \in \left|P\left(\vec{t}\{\psi/x, \pi_1\,\phi/y\}\right)\right|_m\right\}$$

but since $\pi_2\,\phi \in \mathcal{C}(1,1)$ which is a singleton, $|R\{\psi/x\}|_m$ is isomorphic to:

$$\left\{\zeta \in \langle\!|\iota|\!\rangle \middle|\ \left|P\left(\vec{t}\{\psi/x, \zeta/y\}\right)\right|_m \neq \emptyset\right\} = \left\{\zeta \in \langle\!|\iota|\!\rangle \middle|\ \vec{t}^{\,*}\{\psi/x, \zeta/y\} \in \{\!|P|\!\}\right\}$$

but this last set is exactly the one chosen for $|\bot|_c$ in Theorem 5. Finally, the requirement of Theorem 4 that for every $A \in \mathcal{A}x$ we have $\mathcal{A}x \vDash_m A^R$ provides through adequacy a method to turn a set of realizers $\{\zeta_B \in |B|_m \mid B \in \mathcal{A}x\}$ into a realizer $\xi_A \in |A|_c \simeq \left|A^R\right|_m$.

Before presenting another method for direct extraction, we discuss the reason for the correspondence between the two methods. This correspondence comes from the fact that we chose $\diamond = \iota$ in the direct interpretation so we could take elements of ι as realizers of $\bot$. This choice was motivated by the fact that taking $|\bot|_c = \emptyset$ gives a degenerated model. However, this is an unnatural interpretation, since the object $\diamond$ should represent an empty type. We will see that even though this natural interpretation is not possible in Scott domains, it is possible in bistable bicpos and game semantics.

3 Another direct interpretation

In this section we present another direct realizability interpretation of $\mathfrak{C}$ in a category of continuations $\mathcal{C} = \diamond^{\mathcal{D}}$. The interpretation of $\mathfrak{C}$-proofs as terms of $\lambda\mu$ is almost identical as in Section 2.2, the only difference being that a proof term $\vec{p} : \vec{A} \vdash_c M : B$ with $\mathrm{FV}\left(\vec{A}, B\right) = \{\vec{x}\}$ is now mapped to a term $M^{*c} : B^{*c}$ of $\lambda\mu$ with free λ-variables among $\vec{p} : \vec{A}^{*c}$, $\vec{x} : \vec{\iota}$, and a special free μ-variable $\kappa : \iota$, which doesn't appear in M^{*c} but may appear in an arbitrary realizer. This free μ-variable will only be used for extraction, as a channel to transmit the extracted value. In categories of continuations, a term $M : T$ of $\lambda\mu$ with λ-context Γ and μ-context Δ is interpreted as a morphism in $\mathcal{C}(\Gamma, T \parr \Delta)$ where $\parr$ is the pretensor defined by $\diamond^X \parr \diamond^Y \triangleq \diamond^{X \times Y}$, see [25]. Therefore, we adapt the realizability semantics, with the realizability value of a formula A being now $|A|_c \subseteq \mathcal{C}(1, A^* \parr \iota)$. The parameter $|\bot|_c$ is now a subset of $\mathcal{C}(1, \diamond \parr \iota)$, which is isomorphic to the homset $\mathcal{C}(1, \iota)$ used in the previous direct interpretation. The difference is that now we can choose $\diamond \neq \iota$ and take $\diamond$ to be a "truly" empty object. The new realizability value for atomic predicates is:

$$\left|P\left(\vec{t}\right)\right|_c \triangleq \begin{cases} |\bot \Rightarrow \bot|_c & \text{if } \vec{t}^{\,*} \in \{\!|P|\!\} \\ \{\phi \mid \text{if } \psi \in \mathcal{C}(1, \diamond \parr \iota) \text{ then } \phi\,\psi \in |\bot|_c\} & \text{otherwise} \end{cases}$$

and the other definitions go through easily, the "$\parr\,\iota$" part being carried over transparently as a "semantic" free μ-variable. For the definition of $|\forall x\, A|_c$, the morphism $\psi \in \langle\!|\iota|\!\rangle \subseteq \mathcal{C}(1, \iota)$ is viewed as a morphism in $\mathcal{C}(1, \iota \parr \iota)$ by adding the semantic μ-variable κ with weakening, that is, post-composing ψ with $w^l_{\iota,\iota}$ (with the notations of [25], r emark 2.6). Adequacy still holds and the extraction theorem is very similar to the previous one:

▶ **Theorem 7** (Extraction). *From a proof of* $\mathcal{A}x \vdash_c \forall x\, \exists y\, P\left(\vec{t}\right)$ *where* $FV\left(\vec{t}\right) \subseteq \{x; y\}$*, we can extract a morphism* $\phi \in \mathcal{C}\left(1, \iota \to \iota\right)$ *such that for any* $\psi \in \langle\!|\iota|\!\rangle$*,* $\phi\,\psi \in \langle\!|\iota|\!\rangle$ *and:*

$$\vec{t}^{\,*}\left\{\psi/x, \phi\,\psi/y\right\} \in \{\!|P|\!\}$$

Proof. Write $\vec{p_A} : \vec{A} \vdash_c M : \forall x\, \exists y\, P\left(\vec{t}\right)$ where $\vec{A} \subseteq \mathcal{A}x$. Adequacy gives for any $\psi \in \langle\!|\iota|\!\rangle$:

$$M^{*_c}\left\{\vec{\zeta_A}/\vec{p_A}\right\}\psi \in \left|\exists y\, P\left(\vec{t}\{\psi/x\}\right)\right|_c = \left|\neg\forall y\, \neg P\left(\vec{t}\{\psi/x\}\right)\right|_c$$

We fix now the parameter $\left|\bot\right|_c \subseteq \mathcal{C}\left(1, \diamond \parr \iota\right)$:

$$\left|\bot\right|_c = \left\{\zeta \in \mathcal{C}\left(1, \diamond \parr \iota\right) \middle|\ \mu\kappa.\zeta \in \langle\!|\iota|\!\rangle \text{ and } \vec{t}^{\,*}\left\{\psi/x, \mu\kappa.\zeta/y\right\} \in \{\!|P|\!\}\right\}$$

Here, the morphism $\zeta \in \mathcal{C}\left(1, \diamond \parr \iota\right)$ is viewed as a term of $\lambda\mu$ of type $\diamond$ with the special free μ-variable κ of type ι, and therefore $\mu\kappa.\zeta \in \mathcal{C}\left(1, \iota\right)$. We prove that $\lambda yp.p\left([\kappa]\, y\right) \in \left|\forall y\, \neg P\left(\vec{t}\{\psi/x\}\right)\right|_c$ by taking $\varphi \in \langle\!|\iota|\!\rangle$ and $\xi \in \left|P\left(\vec{t}\{\psi/x, \varphi/y\}\right)\right|_c$ and showing $\xi\left([\kappa]\,\varphi\right) \in \left|\bot\right|_c$. We distinguish two cases. If $\vec{t}^{\,*}\left\{\psi/x, \varphi/y\right\} \in \{\!|P|\!\}$, then $\xi \in \left|\bot \Rightarrow \bot\right|_c$ and we are left to prove $[\kappa]\,\varphi \in \left|\bot\right|_c$, which is true since $\mu\kappa.\,[\kappa]\,\varphi = \varphi$ (because the semantic μ-variable κ doesn't appear in φ which comes from weakening), $\varphi \in \langle\!|\iota|\!\rangle$ and $\vec{t}^{\,*}\left\{\psi/x, \varphi/y\right\} \in \{\!|P|\!\}$. In the other case $\vec{t}^{\,*}\left\{\psi/x, \varphi/y\right\} \notin \{\!|P|\!\}$, and $\xi\left([\kappa]\,\varphi\right) \in \left|\bot\right|_c$ by definition of $\left|P\left(\vec{t}\{\psi/x, \varphi/y\}\right)\right|_c$, since $[\kappa]\,\varphi \in \mathcal{C}\left(1, \diamond \parr \iota\right)$. Therefore we get:

$$\phi_0 = M^*\left\{\vec{\zeta_A}/\vec{p_A}\right\}\psi\left(\lambda yp.p\left([\kappa]\, y\right)\right) \in \left|\bot\right|_c$$

so $\mu\kappa.\phi_0 \in \langle\!|\iota|\!\rangle$ and $\vec{t}^{\,*}\left\{\psi/x, \mu\kappa.\phi_0/y\right\} \in \{\!|P|\!\}$ by definition of $\left|\bot\right|_c$. Finally, the following morphism has the required property:

$$\phi = \lambda x.\mu\kappa.M^*\left\{\vec{\zeta_A}/\vec{p_A}\right\}x\left(\lambda yp.p\left([\kappa]\, y\right)\right)\ .$$ ◀

The computational behavior of the extracted term is different from that of the extracted term of Section 2.6, because as soon as the argument $\left(\lambda yp.p\left([\kappa]\, y\right)\right)$ is called inside M^{*_c}, y receives a value which is directly transmitted over channel κ and redirected to toplevel. Conversely, in the previous direct interpretation, once the argument $\left(\lambda yp.p\, y\right)$ was called, the value given to y had to go through all the call stack before returning to toplevel. This interpretation corresponds to the meaning of the control features of $\lambda\mu$-calculus.

This new direct interpretation's improvement relies heavily on the fact that we do not require $\diamond = \iota$ anymore, and we can choose $\diamond$ to be a "truly" empty object. We will see in the next sections that the ability to choose $\diamond \neq \iota$ is very dependent on the particular model that we choose.

3.1 Failure in Scott domains

In this section, we explain why in Scott domains, we have no choice but to take $\diamond = \iota$ if we want ι to be in the category of continuations. Recall that a Scott domain is a partial order with a least element, least upper bounds of directed subsets, least upper bounds of non-empty upper-bounded subsets, and which is algebraic (see e.g. [3] for the definitions and basic properties). The standard domain interpretation of a base type ι with set-theoretic interpretation $[\![\iota]\!]$ is $[\![\iota]\!]_\bot = \left([\![\iota]\!] \cup \{\bot\}, \leq\right)$ with $x \leq y$ if and only if $x = y$ or $x = \bot$. First, we should ask ourselves what should $\mathcal{D}$ be if $\diamond^{\mathcal{D}}$ is a category of continuations of domains. The category of Scott domains is cartesian closed and has fixpoints: for any morphism $\phi : X \to X$ there is a morphism $\psi : \mathbf{1} \to X$ such that $\psi; \phi = \psi$. It is well-known that a bicartesian closed

category with fixpoints has to be trivial (since in that case $\mathbf{1} \simeq \mathbf{0}$). Therefore, since $\mathcal{D}$ should have coproducts, we should relax one of the conditions of Scott domains. The most natural choice is to drop the requirement of existence of a least element and define $\mathcal{D}$ as the category of unpointed Scott domains. In that case, the set-theoretic disjoint union provide $\mathcal{D}$ with a bicartesian closed structure, at the expense of not having fixpoints. We can now state the following failure lemma:

▶ **Lemma 8.** *If $\mathcal{D}$ is a category of unpointed Scott domains, if $\mathcal{C} = \diamond^{\mathcal{D}}$ is a category of Scott domains, if $[\![\iota]\!] \neq \emptyset$ and if $[\![\iota]\!]_\perp$ is an object of $\mathcal{C}$, then $\diamond \simeq [\![\iota]\!]_\perp$.*

Proof. Suppose $[\![\iota]\!]_\perp$ is an object of $\mathcal{C} = \diamond^{\mathcal{D}}$. Then there is some unpointed domain X in $\mathcal{D}$ such that $[\![\iota]\!]_\perp$ is the domain $\diamond^X$ of functions from X to $\diamond$. If X is empty then $\diamond^X$ has only one element, which is impossible since $[\![\iota]\!] \neq \emptyset$. If X has only one element, then $X \simeq \mathbf{1}$ and $\diamond \simeq [\![\iota]\!]_\perp$, which is the conclusion of the lemma. Suppose now that X has at least two elements $a \neq b$. We will derive a contradiction. Since X is non-empty and $\diamond^X = [\![\iota]\!]_\perp$ is pointed, $\diamond$ is also pointed and we write $\perp_\diamond$ for its least element. If $\diamond = \{\perp_\diamond\}$ then $\diamond^X$ has only one element, which is impossible since $[\![\iota]\!] \neq \emptyset$. Therefore there must be some $c \neq \perp_\diamond$ in $\diamond$. Since X is algebraic, we can suppose without loss of generality that a and b are compact (a non-compact element dominates infinitely many compact elements), and since $a \neq b$, we can also suppose without loss of generality that $a \not\geq b$. Define now monotone continuous functions f, g and h from X to $\diamond$ by:

$$f(x) = \perp_\diamond \qquad g(x) = \begin{cases} c \text{ if } x \geq b \\ \perp_\diamond \text{ otherwise} \end{cases} \qquad h(x) = c$$

From the above assumptions we have $f < g < h$ ($a \not\geq b$ ensures $g \neq h$), but $[\![\iota]\!]_\perp$ has no chain of length > 2, so $\diamond^X \not\simeq [\![\iota]\!]_\perp$. ◀

3.2 Bistable bicpos

In this section we prove that the category of bistable bicpos [19] is a category of continuations $\diamond^{\mathcal{D}}$ in which the natural interpretation of ι is in general different from $\diamond$. First we recall the definition of bistable biorders and bistable functions:

▶ **Definition 9** (Bistable Biorder)**.** A bistable biorder is a partial order $(X, \leq)$ together with an equivalence relation $\updownarrow$ on X such that for each $\updownarrow$-equivalence class E, $\left(E, \leq_{|E}\right)$ is a distributive lattice and the inclusion $E \subseteq X$ preserves meets and joins.

▶ **Definition 10** (Bistable Function)**.** A bistable function from $(X, \leq_X, \updownarrow_X)$ to $(Y, \leq_Y, \updownarrow_Y)$ is a monotone function $f : X \to Y$ such that for any $x, y \in X$, $x \updownarrow_X y$ implies:

$$f(x) \updownarrow_Y f(y) \qquad f(x \wedge y) = f(x) \wedge f(y) \qquad f(x \vee y) = f(x) \vee f(y)$$

The set Y^X of bistable functions from $(X, \leq_X, \updownarrow_X)$ to $(Y, \leq_Y, \updownarrow_Y)$ is itself a bistable biorder:

▶ **Lemma 11.** *If f, g are bistable functions from $(X, \leq_X, \updownarrow_X)$ to $(Y, \leq_Y, \updownarrow_Y)$, define:*

$$f \leq_{Y^X} g \equiv \forall x \in X, f(x) \leq_Y g(x)$$

$$f \updownarrow_{Y^X} g \equiv \begin{cases} \forall x \in X, f(x) \updownarrow_Y g(x) \\ \forall x, y \in X, x \updownarrow_X y \Rightarrow \begin{cases} f(x) \wedge g(y) = f(y) \wedge g(x) \\ f(x) \vee g(y) = f(y) \vee g(x) \end{cases} \end{cases}$$

(note that if $x \updownarrow_X y$ then $f(x) \updownarrow_Y f(y) \updownarrow_Y g(x) \updownarrow_Y g(y)$)

$\left(Y^X, \leq_{Y^X}, \updownarrow_{Y^X}\right)$ is a bistable biorder.

As shown in [19], the category of bistable biorders is bicartesian closed, the product and coproduct of bistable biorders being defined pointwise. Bistable bicpos are then defined by adding notions of completeness and continuity to bistable biorders:

▶ **Definition 12.** Let $(X, \leq, \updownarrow)$ be a bistable biorder. If E, F are directed subsets of X, write $E \updownarrow F$ if for any $x \in E$ and $y \in F$ there exists $x' \in E$ and $y' \in F$ such that $x \leq x'$, $y \leq y'$ and $x' \updownarrow y'$. $(X, \leq, \updownarrow)$ is a bistable bicpo if $(X, \leq)$ has least upper bounds of directed subsets and for any $E \updownarrow F$, $\bigvee E \updownarrow \bigvee F$ and $\bigvee E \wedge \bigvee F = \bigvee \{x \wedge y \,|\, x \in E, y \in F, x \updownarrow y\}$.

Similarly to the case of Scott domains, we say that a bistable bicpo is pointed if it has both a least and a greatest element which are $\updownarrow$-equivalent. The category of unpointed bistable bicpos and bistable continuous functions is bicartesian closed, while the full subcategory of pointed bistable bicpos is cartesian closed and has fixpoints. The standard interpretation of a base type ι with set-theoretic interpretation $[\![\iota]\!]$ is the bistable bicpo $[\![\iota]\!]_\perp^\top = ([\![\iota]\!] \cup \{\perp; \top\}, \leq, \updownarrow)$ with $x \leq y$ if and only if $x = y$ or $x = \perp$ or $y = \top$, and $x \updownarrow y$ if and only if $x = y$ or $\{x; y\} = \{\perp; \top\}$. If we choose $\mathcal{D}$ to be the category of unpointed bistable bicpos and $\diamond = \emptyset_\perp^\top$, then $\mathcal{C} = \diamond^{\mathcal{D}}$ is a category of continuations of bistable bicpos and all $[\![\iota]\!]_\perp^\top$ are objects of $\mathcal{C}$. Indeed, it was proved in [19] (for $[\![\iota]\!] = \mathbb{N}$, but the extension to arbitrary $[\![\iota]\!]$ is straightforward) that if $[\![\iota]\!]$ denotes the unpointed bistable bicpo $([\![\iota]\!], \leq, \updownarrow)$ with $x \leq y$ if and only if $x = y$ and $x \updownarrow y$ if and only if $x = y$, then $[\![\iota]\!]_\perp^\top$ is isomorphic to the space of bistable continuous functions from $\left(\emptyset_\perp^\top\right)^{[\![\iota]\!]}$ to $\emptyset_\perp^\top$.

One may wonder if the failure in Scott domains was not simply because we interpreted the datatype with values in $[\![\iota]\!]$ as $[\![\iota]\!]_\perp$ rather than $[\![\iota]\!]_\perp^\top$, which is also a Scott domain. This is not the case, since the space of monotone continuous functions from $\emptyset_\perp^{\top\,[\![\iota]\!]}$ (where $[\![\iota]\!]$ is the unpointed Scott domain with $x \leq y$ iff $x = y$) to $\emptyset_\perp^\top$ is isomorphic to the set $\{\mathcal{E} \subseteq \mathcal{P}_{fin}([\![\iota]\!]) \,|\, E \in \mathcal{E} \wedge E \subseteq F \Rightarrow F \in \mathcal{E}\}$ ordered with inclusion, which is clearly not isomorphic to $[\![\iota]\!]_\perp^\top$. A careful analysis shows that if we restrict ourselves to bistable continuous functions, then the sets $\mathcal{E} \subseteq \mathcal{P}_{fin}([\![\iota]\!])$ above also have to satisfy: $E \in \mathcal{E} \wedge F \in \mathcal{E} \Rightarrow E \cap F \in \mathcal{E}$ and $E \cup F \in \mathcal{E} \Rightarrow E \in \mathcal{E} \vee F \in \mathcal{E}$. The only possibilities are then $\mathcal{E} = \emptyset$, $\mathcal{E} = \mathcal{P}_{fin}([\![\iota]\!])$ and $\mathcal{E} = \{E \in \mathcal{P}_{fin}([\![\iota]\!]) \,|\, v \in E\}$ for $v \in [\![\iota]\!]$, and we indeed get back $[\![\iota]\!]_\perp^\top$.

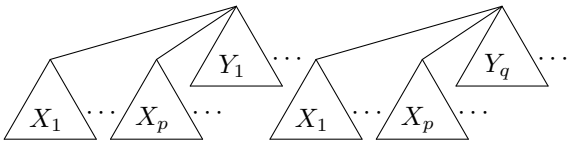
3.3 Game semantics

In this section, we take $\mathcal{C}$ to be the category of unbracketed non-innocent but single-threaded Hyland-Ong games. Hyland-Ong game semantics provide precise models of various programming languages such as PCF [13, 21, 2], also augmented with control operators [18] and higher-order references [1]. In game semantics, plays are interaction traces between a program (player P) and an environment (opponent O). A program is interpreted by a strategy for P which represents the interactions it can have with any environment. We will only define what is necessary for our result, and we refer to e.g. [10] for the full definitions and properties. In the category $\mathcal{C}$, objects are arenas and morphisms are strategies:

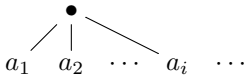
▶ **Definition 13** (Arena)**.** An *arena* is a countably branching, finite depth forest of *moves*. Each move is given a polarity O (for Opponent) or P (for Player or Proponent): a root is of polarity O and a move which is not a root has the inverse polarity than that of his parent. A root of an arena is also called an initial move.

▶ **Definition 14** (Play, Strategy)**.** A play on an arena X is a justified sequence of moves of X with alternating polarities, starting with an O-move. A *strategy* on X is a non-empty even-prefix-closed set of even-length plays on X which is deterministic and single-threaded.

A play on an arena is the trace of an interaction between a program and a context, each one performing an action alternately, and a strategy represents all the interactions that a given program can have with its environment. The definitions of justified sequence, determinism and single-threadedness are standard and can be found for example in [10]. $\mathcal{C}$ is cartesian closed and has countable products, the I-indexed product of $\{X_i \mid i \in I\}$ being the juxtaposition of arenas X_i, and if X and Y are arenas consisting of the trees $X_1 \dots X_p \dots$ and $Y_1 \dots Y_q \dots$, then the arena Y^X can be represented as follows (roots are at the top):



The standard interpretation of a base type ι in $\mathcal{C}$ is the flat arena $[\![\iota]\!]^\dagger$ associated to the set-theoretic interpretation $[\![\iota]\!]$ of ι. This flat arena is the tree with one root move and a child move for each element of $[\![\iota]\!] = \{a_1; a_2; \dots; a_i; \dots\}$:



As in the cases of Scott domains and bistable bicpos, if we want to get a category of continuations $\diamond^{\mathcal{D}}$, we must first find out what $\mathcal{D}$ should be. In game semantics, however, there is no natural notion of unpointed arena, since the strategy consisting of only the empty play is always a least element. We will therefore simply take $\mathcal{D}$ to be the countable coproduct completion $\mathrm{Fam}(\mathcal{C})$ of $\mathcal{C}$:

▶ **Definition 15** ($\mathrm{Fam}(\mathcal{C})$)**.** The objects of $\mathrm{Fam}(\mathcal{C})$ are families of objects of $\mathcal{C}$ indexed by countable sets, and a morphism from $\{X_i \mid i \in I\}$ to $\{Y_j \mid j \in J\}$ is a function $f : I \to J$ together with a family of morphisms of $\mathcal{C}$ from X_i to $Y_{f(i)}$, for $i \in I$.

$\mathrm{Fam}(\mathcal{C})$ is a distributive category, the empty product being the singleton family $\{\mathbf{1}\}$, the product of $\{X_i \mid i \in I\}$ and $\{Y_j \mid j \in J\}$ being $\{X_i \times Y_j \mid (i,j) \in I \times J\}$, the empty coproduct being the empty family $\{\}$, and the coproduct of two families being the disjoint union of the two families. $\mathrm{Fam}(\mathcal{C})$ is not cartesian closed, but has exponentials of all singleton families: the object of functions from $\{X_i \mid i \in I\}$ to the singleton family $\{Y\}$ is the singleton family $\left\{\Pi_{i \in I} Y^{X_i}\right\}$. This property implies that if $\diamond$ is a singleton family, then $\diamond^{\mathcal{D}} = \diamond^{\mathrm{Fam}(\mathcal{C})}$ is a category of continuations. We now prove that if we choose $\diamond$ carefully, we can completely reveal the continuation structure of $\mathcal{C}$:

▶ **Lemma 16.** *If $\diamond = \left\{\emptyset^\dagger\right\}$, then the category $\diamond^{\mathcal{D}} = \diamond^{\mathrm{Fam}(\mathcal{C})}$ is isomorphic to the category $\mathcal{C}$.*

Proof. Since $\left\{\emptyset^\dagger\right\}$ is a singleton family, the objects of $\diamond^{\mathcal{D}}$ are all singleton families and we have a functor from $\diamond^{\mathcal{D}}$ to $\mathcal{C}$ which is full, faithful and strictly injective on objects and which maps $\{X\}$ to X and $(Id, \phi) \in \diamond^{\mathcal{D}}(\{X\}, \{Y\})$ to $\phi \in \mathcal{C}(X, Y)$. We now show that it is also surjective on objects, so it is an isomorphism of categories. This amounts to show that any arena X can be written as $\Pi_{i \in I}\left(\emptyset^\dagger\right)^{Y_i}$, but this is immediate if one takes I to be the set of roots of X and Y_i to be the forest under root $i \in I$ in X. ◀

Therefore $\mathcal{C}$ is (isomorphic to) a category of continuations $\diamond^{\mathcal{D}}$ where $\diamond \neq \iota$, which was our goal. Note that this result is stronger than the one on bistable bicpos, since $\diamond^{\mathcal{D}}$ is not only a category of continuations of arenas which has the natural interpretation of ι as an object, but $\diamond^{\mathcal{D}}$ is isomorphic to the category $\mathcal{C}$ of arenas.

We now examine the extraction technique presented at the beginning of Section 3 in the particular case of game semantics. In order to have a realizability model, we define $\langle\!|\iota|\!\rangle$ to be the set of all strategies on $[\![\iota]\!]^\dagger$ but $\{\epsilon\}$, so we have $\langle\!|\iota|\!\rangle \simeq [\![\iota]\!]$. We write $\underline{a}$ for the strategy on $[\![\iota]\!]^\dagger$ corresponding to $a \in [\![\iota]\!]$ (it answers a to the unique initial move), so $\langle\!|\iota|\!\rangle = \{\underline{a} \mid a \in [\![\iota]\!]\}$. The extraction technique for classical proofs presented at the beginning of Section 3 becomes in this setting:

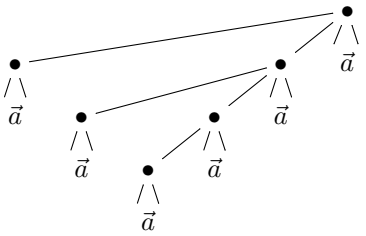
▶ **Theorem 17** (Extraction). *Suppose given a strategy $\zeta_A \in |A|_c$ for each $A \in \mathcal{A}x$. We can extract from a proof of $\mathcal{A}x \vdash_c \forall x\, \exists y\, P\left(\vec{t}\right)$ where $FV\left(\vec{t}\right) \subseteq \{x; y\}$ a strategy ϕ on the arena $\left([\![\iota]\!]^\dagger\right)^{[\![\iota]\!]^\dagger}$ such that for any $a \in [\![\iota]\!]$, $\phi\, \underline{a} = \underline{b}$ for some $b \in [\![\iota]\!]$, and $\vec{t}^*\, \{\underline{a}/x, \underline{b}/y\} \in \{\!|P|\!\}$.*

Let us compare now the arena where realizers of the formula $\forall x\, \exists y\, P\left(\vec{t}\right)$ live in the standard and new interpretations. First, we have:

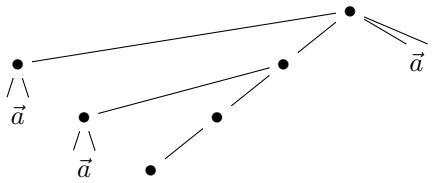
$$\left(\forall x\, \exists y\, P\left(\vec{t}\right)\right)^{*c} = \left(\forall x\, \neg\, \forall y\, \neg\, P\left(\vec{t}\right)\right)^{*c} = \iota \to (\iota \to (\diamond \to \diamond) \to \diamond) \to \diamond$$

In the standard interpretation, $\diamond = \iota$ is $[\![\iota]\!]^\dagger$, the flat arena for $[\![\iota]\!]$, so the arena of realizers is $[\![\iota]\!]^\dagger \Rightarrow \left([\![\iota]\!]^\dagger \Rightarrow \left([\![\iota]\!]^\dagger \Rightarrow [\![\iota]\!]^\dagger\right) \Rightarrow [\![\iota]\!]^\dagger\right) \Rightarrow [\![\iota]\!]^\dagger$. In the new interpretation however, $\diamond$ is the one-move arena $\emptyset^\dagger$, and a "$\parr\, \iota$" is added, so the arena of the realizers of the same formula is $\left([\![\iota]\!]^\dagger \Rightarrow \left([\![\iota]\!]^\dagger \Rightarrow (\emptyset^\dagger \Rightarrow \emptyset^\dagger) \Rightarrow \emptyset^\dagger\right) \Rightarrow \emptyset^\dagger\right) \parr [\![\iota]\!]^\dagger$. The two arenas are as follows:

standard interpretation:



new interpretation:



where $\vec{a}$ represents the sequence of all elements of $[\![\iota]\!]$. We can observe that the flat arenas for the atomic formulas in the standard interpretation are replaced with one-move arenas in the new interpretation, and a set of moves $\vec{a}$ is added, but only under the root (this corresponds to the "$\parr\, \iota$"). On the computational level, when in the standard interpretation a realizer gives a value for an atomic formula, this value is copied to the various parts of the arena which are in the call stack. Conversely, in the new interpretation the realizer writes the value directly under the root and the computation stops (this corresponds to the interpretation of μ-variables as "channels").

4 Conclusion

We defined a method to extract strategies of game semantics from classical proofs. This method uses peculiarities of the games model allowing the interpretation of the $\bot$ formula as an empty type while still being able to extract computational content, through the use of an external "output channel" on which the extracted value is transmitted, without going through all the call stack. It would be interesting to compare these results with the technique described in [20], which also reduces the amount of recursive calls, but takes place in an untyped, syntactic setting. Working with non-innocent games also has the advantage that we have access to higher-order references which could be used to write efficient realizers.

References

1. Samson Abramsky, Kohei Honda, and Guy McCusker. A Fully Abstract Game Semantics for General References. In *13th Annual IEEE Symposium on Logic in Computer Science*, pages 334–344. IEEE Computer Society, 1998.
2. Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full Abstraction for PCF. *Information and Computation*, 163(2):409–470, 2000.
3. Roberto Amadio and Pierre-Louis Curien. *Domains and Lambda-Calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998.
4. Stefano Berardi, Marc Bezem, and Thierry Coquand. On the Computational Content of the Axiom of Choice. *Journal of Symbolic Logic*, 63(2):600–622, 1998.
5. Ulrich Berger, Wilfried Buchholz, and Helmut Schwichtenberg. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114(1-3):3–25, 2002.
6. Ulrich Berger and Paulo Oliva. Modified bar recursion and classical dependent choice. In *Logic Colloquium 2001, Proceedings of the Annual European Summer Meeting of the Association for Symbolic Logic*, volume 20 of *Lecture Notes in Logic*, pages 89–107. A K Peters, Ltd., 2005.
7. Valentin Blot and Colin Riba. On Bar Recursion and Choice in a Classical Setting. In *11th Asian Symposium on Programming Languages and Systems*, volume 8301 of *Lecture Notes in Computer Science*, pages 349–364. Springer, 2013.
8. Harvey Friedman. Classically and intuitionistically provably recursive functions. In Gert Müller and Dana Scott, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–27. Springer, 1978.
9. Timothy Griffin. A Formulae-as-Types Notion of Control. In *17th Symposium on Principles of Programming Languages*, pages 47–58. ACM Press, 1990.
10. Russ Harmer. *Games and full abstraction for non-deterministic languages*. PhD thesis, Imperial College London (University of London), 1999.
11. Hugo Herbelin. On the Degeneracy of Sigma-Types in Presence of Computational Classical Logic. In *7th International Conference on Typed Lambda Calculi and Applications*, Lecture Notes in Mathematics, pages 209–220. Springer, 2005.
12. Martin Hofmann and Thomas Streicher. Completeness of Continuation Models for $\lambda\mu$-Calculus. *Information and Computation*, 179(2):332–355, 2002.
13. Martin Hyland and Luke Ong. On Full Abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000.
14. Stephen Cole Kleene. On the Interpretation of Intuitionistic Number Theory. *Journal of Symbolic Logic*, 10(4):109–124, 1945.
15. Jean-Louis Krivine. Dependent choice, 'quote' and the clock. *Theoretical Computer Science*, 308(1–3):259–276, 2003.
16. Jean-Louis Krivine. Realizability in classical logic. *Panoramas et synthèses*, 27:197–229, 2009.
17. Yves Lafont, Bernhard Reus, and Thomas Streicher. Continuations Semantics or Expressing Implication by Negation. Technical Report 93-21, Ludwig-Maximilians-Universität, München, 1993.
18. James Laird. Full Abstraction for Functional Languages with Control. In *12th Annual IEEE Symposium on Logic in Computer Science*, pages 58–67. IEEE Computer Society, 1997.
19. James Laird. Bistable Biorders: A Sequential Domain Theory. *Logical Methods in Computer Science*, 3(2), 2007.
20. Alexandre Miquel. Existential witness extraction in classical realizability and via a negative translation. *Logical Methods in Computer Science*, 7(2), 2011.

21 Hanno Nickau. Hereditarily Sequential Functionals. In *Third International Symposium on Logical Foundations of Computer Science*, Lecture Notes in Computer Science, pages 253–264. Springer, 1994.

22 Paulo Oliva and Thomas Streicher. On Krivine's Realizability Interpretation of Classical Second-Order Arithmetic. *Fundamenta Informaticae*, 84(2):207–220, 2008.

23 Michel Parigot. $\lambda\mu$-Calculus: An Algorithmic Interpretation of Classical Natural Deduction. In *3rd International Conference on Logic Programming and Automated Reasoning*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer, 1992.

24 Christophe Raffalli. Getting results from programs extracted from classical proofs. *Theoretical Computer Science*, 323(1-3):49–70, 2004.

25 Peter Selinger. Control categories and duality: on the categorical semantics of the $\lambda\mu$ calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, 2001.