A first-order completeness result about characteristic Boolean algebras in classical realizability

***Abstract*—We prove the following completeness result about classical realizability: given any Boolean algebra with at least two elements, there exists a Krivine-style classical realizability model whose characteristic Boolean algebra is elementarily equivalent to it. This is done by controlling precisely which combinations of so-called "angelic" (or "may") and "demonic" (or "must") nondeterminism exist in the underlying model of computation.**

I. Introduction

Classical realizability: Realizability is an aspect of the propositions-as-types / proofs-as-programs correspondence in which each proposition is interpreted as a specification on the behaviour of programs: programs which satisfy this specification are said to *realize* the proposition. This interpretation defines a notion of truth value: a proposition counts as "true" if it is realized by a well-formed program. In particular, any provable proposition is true in that sense. Indeed, any proof, when be seen as a program through the correspondence, must realize the proposition that it proves. This fundamental result ensures that realizability is compatible with logical deduction.

Initially, this compatibility was restricted to intuitionistic deduction. Griffin's discovery of a link between control operators and classical reasoning [1] overcame this limitation. More precisely, Griffin proved that Peirce's law–a deductive principle that is valid in classical logic but not in intuitionistic logic–can be used as a specification (i.e. as a type) for Scheme's operator *call/cc* ("call with current continuation"), which allows a program to manipulate its own evaluation context as a first-class object.

Using this idea, Krivine developed a framework which could interpret all classical reasoning, first within second-order arithmetic [2], and then within Zermelo—Frænkel set theory with dependent choice [3]. Miquel then adapted this *classical realizability* to higher-order arithmetic and explored its connections with forcing [4]. Work on interpreting reasoning that uses the full axiom of choice is ongoing [5].

Characteristic Boolean algebras: Abstractly, a *classical realizability model* is the data of a *model of computation* (for example: a variant of the lambda-calculus enriched with the instruction *call/cc*), a *model of deduction* (for example: a first-order language, plus the rules of classical reasoning, and optionally a theory on this language, i.e. a set of axioms), and a *realizability relation* between *programs* (from the former) and *propositions* from the latter.

Each classical realizability model has a *characteristic Boolean algebra* (which Krivine calls ℷ2–"gimel 2"). More precisely, given a classical realizability model and a formula A in the language of Boolean algebras, there is a proposition "the characteristic Boolean algebra satisfies A", which may or may not be realized by any given program.

The set of all first-order formulas A such that the proposition "the characteristic Boolean algebra satisfies A" is realized by a well-formed program (i.e. "true") forms a first-order theory on the language of Boolean algebras: this is called the *first-order theory of the characteristic Boolean algebra of the realizability model*. This theory may or may not be consistent, but it is always closed under classical deduction, and it always contains the theory of Boolean algebras with at least two elements.

The characteristic Boolean algebra, and in particular the ability to "shape" it, plays a central role in classical realizability. For example, by constructing a realizability model whose characteristic Boolean algebra is atomless, Krivine obtained a model of set theory with remarkable combinatorial properties [6]. As an other example, Krivine's construction of a particular classical realizability model that satisfies the axiom of choice [5] depends crucially on the ability to reliably force a realizability model's characteristic Boolean algebra to be isomorphic to any given *finite* Boolean algebra with at least 2 elements (in that case, the Boolean algebra with 4 elements).

Contribution: The contribution of this paper is to prove that the characteristic Boolean algebra can in fact be made elementarily equivalent to *any* given Boolean algebra with at least two elements, finite or not. More precisely, we prove that for each first-order theory $\mathcal{T}$ over the language of Boolean algebras, the following two conditions are equivalent:

- The theory $\mathcal{T}$ is closed under classical deduction and contains the theory of Boolean algebras with at least two elements;
- There exists a classical realizability model whose characteristic Boolean algebra's theory is exactly $\mathcal{T}$.

In particular, given a first-order formula A over the language of Boolean algebra, the proposition "the characteristic Boolean algebra satisfies A" is universally realized (i.e. realized in all models) if and only if A is true in all Boolean algebras with at least two elements.

The proof we give is constructive: given a theory $\mathcal{T}$, we describe a concrete realizability model whose characteristic Boolean algebra's theory is $\mathcal{T}$. The construction works as follows: it has been pointed out [7] that the properties the characteristic Boolean algebra reflect the kinds of nondeterminism that exist in the underlying computational model; so for each formula A in $\mathcal{T}$, what we do is add to the computational model a nondeterministic instruction γ_A which has exactly the right combination of so-called "angelic" (or "may") and "demonic" (or "must") nondeterminism to realize the proposition "the characteristic Boolean algebra satisfies A".

Outline: Section II states well-known facts about classical realizability (including the fact that in the equivalence we want to prove, the second condition implies the first), and lays down the conventions that will be used throughout the paper. To keep the discussion focused and the notations simple, we restrict the language of propositions to the first-order language of Boolean algebras (rather than, say, the language of set theory, or the second-order language of Peano arithmetic). The main benefit is that, in this context, given any first-order formula A in the language of Boolean algebras, the proposition "the characteristic Boolean algebra satisfies A" is simply the formula A: no translation is needed.

Section III details the construction, given any first-order theory $\mathcal{T}$ that is closed under classical deduction and contains the theory of Boolean algebras with at least two elements, of a realizability model that satisfies $\mathcal{T}$.

Section IV proves that this model's characteristic Boolean algebra's theory does indeed contain $\mathcal{T}$, and Section V proves the converse inclusion, which concludes the proof of this paper's main result.

Finally, Section VI gives an example of application of this result to the problem of sequentialisation in a denotational model of the lambda-calculus with a control operator.

II. Conventions and reminders about classical realizability

A. First-order formulas on Boolean algebras

The *language of Boolean algebras* is the first-order language with equality over the signature $(0, 1, \vee, \wedge, \neg)$ (respectively: two constants, two binary function symbols with infix notation, and one unary function symbol). To make it clear which symbols we take as primitives, we spell out its grammar:

First-order terms:

$$\begin{aligned} a, b := \quad & z \text{ (first-order variable)} \\ & \mid 0 \mid 1 \mid a \vee b \mid a \wedge b \mid \neg a \end{aligned}$$

First-order formulas:

$$A, B \quad := \quad a \neq b \mid A \to B \mid \forall z\, A$$

Note that, as is customary in classical realizablity, we take non-equality rather than equality as a primitive symbol, because its realizability interpretation is simpler.

The other usual symbols can be encoded as follows:

- $\bot$ is $0 \neq 0$,
- for all first-order terms a, b, $a = b$ is $(a \neq b) \to \bot$,
- for all first-order formulas A, B, $A \wedge B$ is $(A \to B \to \bot) \to \bot$,
- for all first-order formulas A, B, $A \vee B$ is $(A \to \bot) \to (B \to \bot) \to \bot$,
- for all first-order formulas A and all first-order variables z, $\exists z\ A$ is $(\forall z\ (A \to \bot)) \to \bot$.

A set of closed first-order formulas is called a *first-order theory*. Over the signature we have chosen, the theory of Boolean algebras can be axiomatised by a finite set of equations. As a result, there exists a finite first-order theory $\mathcal{T}_{\mathrm{Bool}}$ consisting of:

- the first-order formula $0 \neq 1$,
- plus a finite number of closed first-order formulas of the form $\forall \overline{z}\ a = b$ (where $\overline{z}$ is a list of variables),

such that for each first-order structure $\mathbb{B}$ over the language of Boolean algebras, $\mathbb{B}$ satisfies $\mathcal{T}_{\mathrm{Bool}}$ if and only if $\mathbb{B}$ is a Boolean algebra with at least two elements.

First-order formulas are defined up to α-renaming. Given a first-order formula A (repectively, a first-order term a), a list $\overline{z}$ of variables and a list $\overline{b}$ of first-order terms of equal length, we denote by $A[\overline{z} := \overline{b}]$ (respectively, $a[\overline{z} := \overline{b}]$) the simultaneous, capture-avoiding substitution of $\overline{z}$ with $\overline{b}$ in A (respectively, in a).

B. The λ_c-calculus

Syntax: The λ_c-calculus consists of three kinds of syntactic objects: λ_c*-terms* (which represent programs), *stacks* (which represent execution environments), and processes (which represent a program running in a given environment). They are defined by the following grammars, up to α-renaming:

λ_c-terms:

$$\begin{aligned} A, B := \quad & x \mid tu \mid \lambda x.\, t \\ & \mid \mathrm{cc} \quad \text{(call with current continuation)} \\ & \mid k_\pi \quad (\pi \text{ stack}) \\ & \mid \zeta_n \quad (n \in \mathbb{N}\text{: unrestricted additional instructions}) \\ & \mid \eta_n \quad (n \in \mathbb{N}\text{: restricted additional instructions}) \end{aligned}$$

Stacks:

$$\begin{aligned} \pi, \pi' \quad := \quad & t \cdot \pi \quad (t \textit{ closed } \lambda_c\text{-term}) \\ \mid \quad & \omega \quad \text{(empty stack)} \end{aligned}$$

Processes:

$$p, q \quad := \quad t \star \pi \quad (t \textit{ closed } \lambda_c\text{-term})$$

Given a λ_c-term t, a list $\overline{x}$ of variables and a list $\overline{u}$ of λ_c-terms of equal length, we denote by $t[\overline{x} := \overline{u}]$ the simultaneous, capture-avoiding substitution of $\overline{x}$ with $\overline{u}$ in t.

Operational semantics: Processes are evaluated according to the rules of the *Krivine abstract machine*. Namely, we denote by $\succ_1$ ("evaluates in one step to") the least binary relation on the set of processes such that:

$$\begin{array}{rcll} tu \star \pi & \succ_1 & t \star u \cdot \pi & \text{(Push)} \\ \lambda x.\, v \star t \cdot \pi & \succ_1 & v[x := t] \star \pi & \text{(Grab)} \\ \mathrm{cc} \star t \cdot \pi & \succ_1 & t \star k_\pi \cdot \pi & \text{(Save)} \\ k_{\pi_2} \star t \cdot \pi_1 & \succ_1 & t \star \pi_2 & \text{(Restore)} \end{array}$$

for all closed terms $t, u, \lambda x.\, v$ and all stacks π, π_1, π_2.

Moreover, we denote by $\succ$ ("evaluates to") the reflexive and transitive closure of $\succ_1$.

The rules Push and Grab simulate weak head β-reduction, and the rules Save and Restore allow programs to manipulate continuations (cc stands for "call with current continuation"). In the context of realizability, the former pair will ensure compatibility with intuitionistic logic, and the latter with classical logic.

Note that there are no rules for the additional instructions: their purpose will be to help construct specific *poles* (see next subsection), and they can be ignored for the time being.

Typing: Typing judgements have the following form: $x_1 : A_1, \dots, x_n : A_n \vdash t : B$, where $x_1, \dots, x_n$ are pairwise distinct variables, t is a λ_c-term with no free variables other than $x_1, \dots, x_n$, and $A_1, \dots, A_n$ are first-order formulas (possibly with free variables).

Typing judgements are defined up to α-renaming (i.e. $x_1 : A_1, \dots, x_n : A_n \vdash t : B$ is the same as $y_1 : A_1, \dots, y_n : A_n \vdash t[\overline{x} := \overline{y}] : B$), and up to permutations of the context (i.e. $x_1 : A_1, \dots, x_n : A_n \vdash t : B$ is the same as $x_{\sigma(1)} : A_{\sigma(1)}, \dots, x_{\sigma(n)} : A_{\sigma(n)} \vdash t : B$ for all permutations σ).

A typing judgement is *valid* if it can be derived from the following rules:

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

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall z.\, A}\ (z \text{ not free in } \Gamma) \qquad \frac{\Gamma \vdash t : \forall z.\, A}{\Gamma \vdash t : A[z := b]}$$

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{}{\Gamma \vdash \mathrm{cc} : ((A \to B) \to A) \to A}$$

$$\frac{\Gamma[z := a] \vdash t : a \neq b}{\Gamma[z := b] \vdash t : a \neq b} \qquad \frac{\Gamma \vdash t : a \neq a}{\Gamma \vdash t : A}$$

The first five are the usual rules of natural deduction, and the sixth types cc with Peirce's law (which allows classical deduction). The last two reformulate the usual elimination and introduction rules for equality using the symbol $\neq$ instead:

Lemma 1. *The following rule is admissible:*

$$\frac{\Gamma \vdash t : A[z := a]}{\Gamma, x : a = b \vdash \mathrm{cc}(\lambda k.\, x(kt)) : A[z := b]}\ .$$

Proof. If the typing judgement $\Gamma \vdash t : A[z := a]$ is valid, then so is the judgement $\Gamma, x : a = b, k : A[z := a] \to a \neq b \vdash t : A[z := a]$. Then we can use the following derivation:

$$\begin{array}{c}
\Gamma, x : a = b, k : A[z := a] \to a \neq b \vdash t : A[z := a] \\ \hline
\vdots \\ \hline
\Gamma, x : a = b, k : A[z := a] \to a \neq b \vdash kt : a \neq b \\ \hline
\Gamma, x : a = b, k : A[z := b] \to a \neq b \vdash kt : a \neq b \\ \hline
\vdots \\ \hline
\Gamma, x : a = b, k : A[z := b] \to a \neq b \vdash x(kt) : \bot \\ \hline
\Gamma, x : a = b, k : A[z := b] \to a \neq b \vdash x(kt) : A[z := b] \\ \hline
\Gamma, x : a = b \vdash \lambda k.\, x(kt) : (A[z{:=}b] \to a \neq b) \to A[z{:=}b] \\ \hline
\vdots \\ \hline
\Gamma, x : a = b \vdash \mathrm{cc}(\lambda k.\, x(kt)) : A[z := b]
\end{array}\ .$$

□

C. Classical realizability

Poles: A *pole* is a set $\bot\!\!\!\bot$ of processes that is *saturated*, i.e. such that for all processes p, q, if $p \succ q$ and $q \in \bot\!\!\!\bot$, then $p \in \bot\!\!\!\bot$.

Falsity values and truth values: For each pole $\bot\!\!\!\bot$ and each closed first-order formula A, we define inductively its *falsity value* $\|A\|_{\bot\!\!\!\bot}$ (which is a set of stacks) and its *truth value* $|A|_{\bot\!\!\!\bot}$ (which is a set of closed λ_c-terms) with respect to $\bot\!\!\!\bot$:

- $|A|_{\bot\!\!\!\bot} = \{t;\ \forall \pi \in \|A\|_{\bot\!\!\!\bot}\ \ t \star \pi \in \bot\!\!\!\bot\}$,
- $\|a \neq b\|_{\bot\!\!\!\bot} = \begin{cases} \emptyset & \text{if } a \neq b \text{ is true (in the Boolean algebra } \{0,1\}), \\ \{\text{all stacks}\} & \text{if } a \neq b \text{ is false.} \end{cases}$
- $\|A \to B\|_{\bot\!\!\!\bot} = \{t \cdot \pi;\ t \in |A|_{\bot\!\!\!\bot}, \pi \in \|B\|_{\bot\!\!\!\bot}\}$,
- $\|\forall z\, A\|_{\bot\!\!\!\bot} = \big\|A[z := 0]\big\|_{\bot\!\!\!\bot} \cup \big\|A[z := 1]\big\|_{\bot\!\!\!\bot}$.

We say that a given closed λ_c-term t *realizes* a given closed first-order formula A *with respect to* a given pole $\bot\!\!\!\bot$ if $t \in |A|_{\bot\!\!\!\bot}$.

Adequacy: A key fact about classical realizability is that it is compatible with the above typing rules, and therefore with classical reasoning:

Lemma 2 (Adequacy lemma)**.** *Let $x_1 : A_1, \dots, x_n : A_n \vdash t : B$ be a valid typing judgement (with $A_1, \dots, A_n$ closed), and let $u_1, \dots, u_n$ be closed λ_c-terms. For all poles $\bot\!\!\!\bot$, if $u_1, \dots, u_n$ realize $A_1, \dots, A_n$ respectively with respect to $\bot\!\!\!\bot$, then $t[\overline{x} := \overline{u}]$ realizes B with respect to $\bot\!\!\!\bot$.*

D. Realizability theories

We would like to associate with each pole a first-order theory of "all first-order formulas that are realized with respect to that pole". However, given any non-empty pole $\bot\!\!\!\bot$, there is bound to be a closed λ_c-term which realizes $\bot$ (namely: take any $t \star \pi \in \bot\!\!\!\bot$ and consider tk_π). Therefore, in order to obtain a meaningful notion, we must put some restrictions on which terms are allowed as realizers.

We call *proof-like* any closed λ_c-term which contains no stack constants (k_π) and no restricted instructions (η_n).

Definition 3. Let $\bot\!\!\!\bot$ be a pole. The *first-order theory of* $\bot\!\!\!\bot$ is the set of all closed first-order formulas which are realized *by at least one proof-like term* with respect to $\bot\!\!\!\bot$. We denote it by $\mathrm{Th}(\bot\!\!\!\bot)$.

As stated in the introduction, the benefit of restricting the formulas to the language of Boolean algebra is that each pole $\bot\!\!\!\bot$ can be interpreted as a realizability model whose characteristic Boolean algebra's theory is simply $\mathrm{Th}(\bot\!\!\!\bot)$: no translation is needed.

We say that a pole $\bot\!\!\!\bot$ is *consistent* if its first-order theory is, i.e. if there exists a first-order structure which satisfies $\mathrm{Th}(\bot\!\!\!\bot)$.

Logical closure: As a consequence of the adequacy lemma and the completeness theorem of first-order logic, the first-order theory of a pole is always *closed under classical deduction*:

Lemma 4. *Let $\perp\!\!\!\perp$ be a pole and A a closed first-order formula. If $\mathrm{Th}(\perp\!\!\!\perp)$ implies A (in the sense that any first-order structure which satisfies $\mathrm{Th}(\perp\!\!\!\perp)$ also satisfies A), then $A \in \mathrm{Th}(\perp\!\!\!\perp)$.*

In particular, $\perp\!\!\!\perp$ is consistent if and only if no proof-like term realizes $\perp$ with respect to it.

Remark 5. In fact, because we chose a very restricted language for formulas, Lemma 4 would hold even in the absence of the instruction cc. However, the goal here is to describe a method which can be generalised to richer contexts, and that requires an instruction such as cc that is capable of altering the control flow: otherwise, all we get is closure under intuitionistic deduction.

Boolean algebras: In addition, the first-order theory of a pole is always an extension of $\mathcal{T}_{\mathrm{Bool}}$, and therefore any first-order structure which satisfies it is a Boolean algebra with at least two elements. This is a consequence of the following lemma:

Lemma 6. *Let A be a closed first-order formula that is true in the Boolean algebra $\{0,1\}$.*

- *If A is of the form $\forall \overline{z}\ a \neq b$, then A is realized by all closed λ_c-terms,* universally *(i.e. with respect to all poles);*
- *If A is of the form $\forall \overline{z}\ a = b$, then A is universally realized by $\lambda x.\, x$.*

Corollary 7. *For all poles $\perp\!\!\!\perp$, $\mathrm{Th}(\perp\!\!\!\perp)$ contains $\mathcal{T}_{\mathrm{Bool}}$.*

Proof of Lemma 6. Let $\perp\!\!\!\perp$ be a pole. First part: for all lists $\overline{w}$ of elements of $\{0,1\}$, we know that $a \neq b$ is true in $\{0,1\}$, therefore the falsity value of $\forall \overline{z}\ a \neq b$ is empty, and so this first-order formula is realized by all closed λ_c-terms.

Second part: let $\overline{\alpha}$ be a list of elements of $\{0,1\}$. We must prove that $\lambda x.\, x$ realizes $(a = b)[\overline{z} := \overline{\alpha}]$. Let $t \cdot \pi$ be in the falsity value of $(a = b)[\overline{z} := \overline{\alpha}]$, i.e. t realizes $a \neq b$ and π is any stack. Since $a \neq b$ is false, its falsity value contains all stacks, which means that $t \star \pi$ is in $\perp\!\!\!\perp$. Since $\lambda x.\, x \star t \cdot \pi$ evaluates to $t \star \pi$, it is also in $\perp\!\!\!\perp$. $\square$

Horn clauses: We have just seen that any (universally quantified) equation or non-equation that is true in the Boolean algebra $\{0,1\}$ is universally realized. In fact, this "transfer" property holds for all Horn clauses:

A *Horn clause* is a closed first-order formula of the form either $\forall \overline{z}\ (a_1 = a'_1 \to \ldots \to a_n = a'_n \to b = b')$ (*definite clause*) or $\forall \overline{z}\ (a_1 = a'_1 \to \ldots \to a_n = a'_n \to b \neq b')$ (*goal clause*).

Lemma 8. *Let A be a Horn clause. Then A is true in the Boolean algebra $\{0,1\}$ if and only if it is true in all Boolean algebras with at least two elements.*

Corollary 9. *Let A be a Horn clause. If A is true in the Boolean algebra $\{0,1\}$, then it is universally realized.*

Proof of Lemma 8. Assume that A is true in $\{0,1\}$, and let $\mathbb{B}$ be a Boolean algebra with at least two element. By Stone's representation theorem, there exists a nonempty set X and an injective morphism of Boolean algebras (i.e. a–non-necessarily elementary–embedding) φ from $\mathbb{B}$ to $\{0,1\}^X$. Since A is a universal (Π^0_1), it is sufficient to prove that A is true in the Boolean algebra $\{0,1\}^X$.

Let $\overline{\delta} \in \{0,1\}^X$. Let $\alpha_1, \alpha'_1, \ldots, \alpha_n, \alpha'_n, \beta, \beta'$ be respectively the values of $a_1[\overline{z} := \overline{\delta}], \ldots, b'[\overline{z} := \overline{\delta}]$ in $\{0,1\}^X$.

Assume that for all $i \leq n$, $\alpha_i = \alpha'_i$, i.e. for all $x \in X$, $\alpha_i(x) = \alpha'_i(x)$.

For all $x \in X$, evaluation at x is a morphism of Boolean algebras from $\{0,1\}^X$ to $\{0,1\}$. Therefore, the value of $a_1[\overline{z} := \overline{\delta}(x)]$ in $\{0,1\}$ is $\alpha_i(x)$, the value of $a'_1[\overline{z} := \overline{\delta}(x)]$ in $\{0,1\}$ is $\alpha'_i(x)$, etc.

Therefore, if A is definite, then for all x, we have $\beta(x) = \beta'(x)$, because A is true in $\{0,1\}$. In other words, $\beta = \beta'$, which means that A is true in $\{0,1\}^X$.

On the other hand, if A is a goal clause, then for all x, we have $\beta(x) \neq \beta'(x)$. Since X is non-empty, this means that $\beta \neq \beta'$, and so A is true in $\{0,1\}^X$. $\square$

With all these conventions written down, we can precisely state the main result of this paper:

Theorem 10. *Let $\mathcal{T}$ be a first-order theory. The following two statements are equivalent:*

- *$\mathcal{T}$ is closed under classical deduction and contains the theory of Boolean algebras with at least two elements;*
- *There exists a pole whose theory is exactly $\mathcal{T}$.*

In particular, a first-order formula is universally realized if and only if it is true in every Boolean algebra with at least two elements.

We have already seen that that the second point implies the first. The task of the remainder of this paper will be to prove the converse implication.

III. Constructing the pole

From now on, $\mathcal{T}$ will denote a fixed first-order theory which is closed under classical deduction and contains $\mathcal{T}_{\mathrm{Bool}}$. We will construct a pole $\perp\!\!\!\perp_{\mathcal{T}}$ whose theory is exactly $\mathcal{T}$.

For each first-order formula A (closed or not), let γ_A denote:

- one of the unrestricted instructions if $A \in \mathcal{T}$,
- one of the restricted instructions otherwise.

Furthermore, let the γ_A be pairwise distinct.

We will construct a pole $\perp\!\!\!\perp_{\mathcal{T}}$ in such a way that γ_A realizes A for all closed first-order formulas A: this will imply that its theory contains $\mathcal{T}$. Then, we will prove the converse inclusion.

A. The structure of first-order formulas

Any first-order formula A can be decomposed as:

$$\forall \overline{y}_1\ (B_1 \to \ldots \to \forall \overline{y}_m\ (B_m \to \forall \overline{y}_{m+1}\ b \neq b') \ldots),$$

with $n \geq 0$, $B_1, \ldots, B_m$ first-order formulas, each $\overline{y}_i$ a list of variables, and b, b' two first-order terms. Moreover, this decomposition is unique, up to renaming of the variables $\overline{y}_1$, $\ldots, \overline{y}_{m+1}$.

Each B_i can itself be decomposed as:

$$\forall \overline{z}_{i,1}\, (C_{i,1} \to \ldots \to \forall \overline{z}_{i,n_i}\, (C_{i,n_i} \to \forall \overline{z}_{i,n_i+1}\, c_i \neq c'_i) \ldots),$$

so that A is decomposed as:

$$\forall \overline{y}_1\ (\boxed{} \to \ldots \to \forall \overline{y}_m\ (\boxed{} \to \forall \overline{y}_{m+1}\ b \neq b')\ldots)$$

$$\forall \overline{z}_{1,1}\,(C_{1,1} \to \ldots \to \forall \overline{z}_{1,n_1}\,(C_{1,n_1} \to \forall \overline{z}_{1,n_1+1}\, c_1 \neq c_1')\ldots)$$

$$\forall \overline{z}_{m,1}(C_{m,1} \to \ldots \to \forall \overline{z}_{m,n_m}(C_{m,n_m} \to \forall \overline{z}_{m,n_m+1} c_m \neq c_m')\ldots)$$

Whenever we decompose a formula A in this way, we will denote by $\overline{\overline{y}}$ the concatenated list $\overline{y}_1, \ldots, \overline{y}_{m+1}$, and for all $1 \leq i \leq m$, we will denote by $\overline{\overline{z}}_i$ the concatenated list $\overline{z}_{i,1}, \ldots, \overline{z}_{i,n_i+1}$. Furthermore, we will assume that the variables $\overline{\overline{y}}, \overline{\overline{z}}_1, \ldots, \overline{\overline{z}}_m$ are chosen all different from one another and from the free variables of A.

B. The pole $\perp\!\!\!\perp_{\mathcal{T}}$

We define by induction an increasing sequence $(\perp\!\!\!\perp_{\mathcal{T},k})_{k\in\mathbb{N}}$ of sets of processes: $\perp\!\!\!\perp_{\mathcal{T},0}$ is empty, and for all k, $\perp\!\!\!\perp_{\mathcal{T},k+1}$ is the smallest set of processes such that:

- For all processes p, q, if $p \succ_1 q$ and $q \in \perp\!\!\!\perp_{\mathcal{T},k}$, then $p \in \perp\!\!\!\perp_{\mathcal{T},k+1}$;
- For each closed first-order formula A (decomposed as in section III-A), for all closed λ_c-terms $t_1, \ldots, t_m$, all stacks π and all lists $\overline{\overline{\beta}} \in \{0,1\}$ such that $(b = b')\left[\overline{\overline{y}} := \overline{\overline{\beta}}\right]$ is true, if the set of processes
$$\left\{ \begin{array}{c} t_i \star \gamma_{C_{i,1}\left[\overline{\overline{z_i}}:=\overline{\overline{\delta_i}},\overline{\overline{y}}:=\overline{\overline{\beta}}\right]} \cdot \ldots \cdot \gamma_{C_{i,n_i}\left[\overline{\overline{z_i}}:=\overline{\overline{\delta_i}},\overline{\overline{y}}:=\overline{\overline{\beta}}\right]} \cdot \pi; \\ i \leq m \text{ and } \overline{\overline{\delta_i}} \in \{0,1\} \text{ such that} \\ (c_i = c_i')\left[\overline{\overline{z_i}} := \overline{\overline{\delta_i}}, \overline{\overline{y}} := \overline{\overline{\beta}}\right] \text{ is true} \end{array} \right\}$$
is included in $\perp\!\!\!\perp_{\mathcal{T},k}$, then the process
$$\gamma_A \star t_1 \cdot \ldots \cdot t_m \cdot \pi$$
is in $\perp\!\!\!\perp_{\mathcal{T},k+1}$.

Then, we define the pole $\perp\!\!\!\perp_{\mathcal{T}}$ as the directed union $\bigcup_{k\in\mathbb{N}} \perp\!\!\!\perp_{\mathcal{T},k}$.

Remark 11. The rule for the instructions γ_A have the following general shape:

> If there exists $\overline{\overline{\beta}}$ such that for all $i, \overline{\overline{\delta}}_i$, $t_i \star \pi'_{\overline{\overline{\beta}},i,\overline{\overline{\delta}}_i}$ is in $\perp\!\!\!\perp_{\mathcal{T}}$, then $\gamma_A \star t_1 \cdot \ldots \cdot t_m \cdot \pi$ is in $\perp\!\!\!\perp_{\mathcal{T}}$.

It has been pointed out [7] that such a rule can be interpreted as saying that γ_A is a special kind of nondeterministic instruction: part "may" (because of the existential quantification), and part "must" (because of the universal quantification).

IV. The theory of $\perp\!\!\!\perp_{\mathcal{T}}$ contains $\mathcal{T}$

We wish to prove that $\mathrm{Th}(\perp\!\!\!\perp_{\mathcal{T}})$ contains $\mathcal{T}$. Since γ_A is proof-like whenever A is in $\mathcal{T}$, it is sufficient to prove the following result:

Proposition 12. *For all closed first-order formulas A, γ_A realizes A with respect to $\perp\!\!\!\perp_{\mathcal{T}}$.*

Proof. We proceed by induction on the height of A. Let A be decomposed as in III-A.

Let $t_1 \cdot \ldots \cdot t_n \cdot \pi$ be in the falsity value of A. In other words, let $\overline{\overline{\beta}}$ be a list of elements of $\{0,1\}$, let $t_1, \ldots t_n$ be closed λ_c-terms such that t_i realizes $B_i\left[\overline{\overline{y}} := \overline{\overline{\beta}}\right]$ for all i, and let π be in the falsity value of $(b \neq b')\left[\overline{\overline{y}} := \overline{\overline{\beta}}\right]$. All we need to do is prove that
$$\gamma_A \star t_1 \cdot \ldots \cdot t_m \cdot \pi$$
is in $\perp\!\!\!\perp_{\mathcal{T}}$.

Let $i \leq m$ and $\overline{\overline{\delta_i}} \in \{0,1\}$ be such that $(c_i = c_i')\left[\overline{\overline{z_i}} := \overline{\overline{\delta_i}}, \overline{\overline{y}} := \overline{\overline{\beta}}\right]$ is true. By the induction hypothesis, we know that for all $j \leq n_i$, $\gamma_{C_{i,j}\left[\overline{\overline{z_i}}:=\overline{\overline{\delta_i}},\overline{\overline{y}}:=\overline{\overline{\beta}}\right]}$ realizes $C_{i,j}\left[\overline{\overline{z_i}} := \overline{\overline{\delta_i}}, \overline{\overline{y}} := \overline{\overline{\beta}}\right]$. Since t_i realizes $B_i\left[\overline{\overline{y}} := \overline{\overline{\beta}}\right]$ and π is in the falsity value of $(c_i \neq c_i')\left[\overline{\overline{z_i}} := \overline{\overline{\delta_i}}, \overline{\overline{y}} := \overline{\overline{\beta}}\right]$ (because this inequality is false), we have that
$$t_i \star \gamma_{C_{i,1}\left[\overline{\overline{z_i}}:=\overline{\overline{\delta_i}},\overline{\overline{y}}:=\overline{\overline{\beta}}\right]} \cdot \ldots \cdot \gamma_{C_{i,n_i}\left[\overline{\overline{z_i}}:=\overline{\overline{\delta_i}},\overline{\overline{y}}:=\overline{\overline{\beta}}\right]} \cdot \pi$$
is in $\perp\!\!\!\perp_{\mathcal{T}}$.

Saying that π is in the falsity value of $(b \neq b')\left[\overline{\overline{y}} := \overline{\overline{\beta}}\right]$ is the same as saying that $(b = b')\left[\overline{\overline{y}} := \overline{\overline{\beta}}\right]$ is true. Therefore, by definition of $\perp\!\!\!\perp_{\mathcal{T}}$, this proves that $\gamma_A \star t_1 \cdot \ldots \cdot t_m \cdot \pi$ is in $\perp\!\!\!\perp_{\mathcal{T}}$.

□

V. The theory of $\perp\!\!\!\perp_{\mathcal{T}}$ is contained in $\mathcal{T}$

All that remains is to prove that The theory $\mathrm{Th}(\perp\!\!\!\perp_{\mathcal{T}})$ is contained in $\mathcal{T}$.

For all λ_c-terms t (respectively, all stacks π; all processes p), let $\mathcal{C}_t$ (respectively, $\mathcal{C}_\pi$; $\mathcal{C}_p$) denote the conjunction of all first-order formulas A such that γ_A appears anywhere in t (respectively, in π; in p)–including nested within a stack constant. Note that $\mathcal{C}_t$ is not necessarily closed even if t is (because the former notion of closure is about first-order variables, while the latter is about variables of the λ_c-calculus).

We are going to prove that for all processes p such that $\mathcal{C}_p$ is closed, if p is in $\perp\!\!\!\perp_{\mathcal{T}}$, then p must contain a contradiction, in the sense that $\mathcal{C}_p$ must be false in all Boolean algebras with at least two elements.

In order to prove this, we will need to state and prove a more general result that also covers the case when $\mathcal{C}_p$ is not closed. To that end, we will need the following notation: for all λ_c-terms t, all lists $\overline{z}$ of first-order variables, and all lists $\overline{b}$ of first-order terms, we denote by $t[\overline{z} := \overline{b}]$ the λ_c-term obtained by replacing each instruction of the form γ_A by $\gamma_{A[\overline{z}:=\overline{b}]}$ (including when they appear nested within a stack constant). Similarly, we define $\pi[\overline{z} := \overline{b}]$ when π is a stack, and $p[\overline{z} := \overline{b}]$ when p is a process.

Proposition 13. *Let p be a process, $\overline{a} = a_1, \ldots, a_r$ and $\overline{a}' = a_1', \ldots, a_r'$ two lists of first-order terms, and $\overline{w}$ a list of distinct* *first-order variables that contains all the free variables of $\mathcal{C}_p$, $\overline{a}$ and $\overline{a}'$.*

Assume that for all lists $\overline{\alpha}$ of elements of $\{0,1\}$ such that $(\overline{a} = \overline{a}')[\overline{w} := \overline{\alpha}]$ is true, $p[\overline{w} := \overline{\alpha}]$ is in $\perp\!\!\!\perp_{\mathcal{T}}$ (where $\overline{a} = \overline{a}'$ denotes the conjunction $(a_1 = a_1') \wedge \ldots \wedge (a_r = a_r')$).

Then the first-order formula $\exists \overline{w}\ (\mathcal{C}_p \wedge (\overline{a} = \overline{a}'))$ is false in all Boolean algebras with at least two elements .

Corollary 14. *Let t be a closed λ_c-term such that $\mathcal{C}_t$ is closed, and A a closed first-order formula. If t realizes A with respect to $\perp\!\!\!\perp_{\mathcal{T}}$, then the formula $\mathcal{C}_t \to A$ is true in all Boolean algebras with at least two elements.*

Proof. If t realizes A with respect to $\perp\!\!\!\perp_{\mathcal{T}}$, then $\gamma_{A\to\perp} \star t \cdot \omega$ is in $\perp\!\!\!\perp_{\mathcal{T}}$, therefore $\mathcal{C}_t \wedge (A \to \perp)$ is false in all Boolean algebras with at least two elements. $\square$

Corollary 15. *The theory $\mathrm{Th}(\perp\!\!\!\perp_{\mathcal{T}})$ is contained in $\mathcal{T}$.*

Proof. Let $A \in \mathrm{Th}(\perp\!\!\!\perp_{\mathcal{T}})$. Let t be a proof-like term which realizes A. The formula $\mathcal{C}_t$ is in $\mathcal{T}$ by construction, because t is proof-like. By the previous corollary, the formula $\mathcal{C}_t \to A$ is also in $\mathcal{T}$, therefore A is in $\mathcal{T}$. $\square$

We now prove the proposition:

Proof of Proposition 13. We will prove by induction that for all natural numbers k, for all p, $\overline{a}$, $\overline{a}'$, $\overline{w}$, if $p[\overline{w} := \overline{\alpha}]$ is in $\perp\!\!\!\perp_{\mathcal{T},k}$ for all $\overline{\alpha} \in \{0,1\}$, then the first-order formula $\exists \overline{w}\ (\mathcal{C}_p \wedge \overline{a} = \overline{a}')$ is false in all Boolean algebras with at least two elements.

The result is vacuously true for $k = 0$, because $\perp\!\!\!\perp_{\mathcal{T},0}$ is empty.

Assume the result holds for some k, and let p, $\overline{a}$, $\overline{a}'$, $\overline{w}$ be such that $p[\overline{w} := \overline{\alpha}]$ is in $\perp\!\!\!\perp_{\mathcal{T},k+1}$ for all $\overline{\alpha} \in \{0,1\}$ such that $(\overline{a} = \overline{a}')[\overline{w} := \overline{\alpha}]$ is true.

If we look back at the definition of $\perp\!\!\!\perp_{\mathcal{T}}$ in section III-B, we see that we must be in one of the following cases:

(i) The formula $\exists \overline{w}\ (\overline{a} = \overline{a}')$ is false in $\{0,1\}$. In that case, this formula is false in all Boolean algebras, because its negation is equivalent to a Horn clause (Lemma 8).

(ii) There exists a process q such that p evaluates in one step to q. In that case, for all $\overline{\alpha} \in \{0,1\}$, if $(\overline{a} = \overline{a}')[\overline{w} := \overline{\alpha}]$ is true, then $q[\overline{w} := \overline{\alpha}]$ must be in $\perp\!\!\!\perp_{\mathcal{T},k}$. Therefore, by the induction hypothesis, the formula $\exists \overline{w}\ (\mathcal{C}_q \wedge \overline{a} = \overline{a}')$ is false in all Boolean algebras with at least two elements. On the other hand, evaluation can only remove or copy the constants γ_A, and not add new ones. This means that the formula $\forall \overline{w}\ (\mathcal{C}_p \to \mathcal{C}_q)$ is a propositional tautology, which proves the result.

(iii) The process p is of the form $\gamma_A \star t_1 \cdot \ldots \cdot t_n \cdot \pi$. In that case, let A be decomposed as in section III-A. Then for all $\overline{\alpha}$ in $\{0,1\}$ such that $(\overline{a} = \overline{a}')[\overline{w} := \overline{\alpha}]$ is true, there exists a list $\overline{\overline{\beta}}_{\overline{\alpha}}$ in $\{0,1\}$ such that $(b = b')\left[\overline{w} := \overline{\alpha}, \overline{\overline{y}} := \overline{\overline{\beta}}_{\overline{\alpha}}\right]$ is true and that set of processes

$$\left\{ \begin{array}{c} t_i[\overline{w} := \overline{\alpha}] \star \gamma_{C_{i,1}\left[\overline{w} := \overline{\alpha}, \overline{\overline{z_i}} := \overline{\overline{\delta_i}}, \overline{\overline{y}} := \overline{\overline{\beta}}_{\overline{\alpha}}\right]} \cdot \ldots \cdot \pi[\overline{w} := \overline{\alpha}]; \\ i \leq m \text{ and } \overline{\overline{\delta_i}} \in \{0,1\} \text{ such that} \\ (c_i = c_i')\left[\overline{w} := \overline{\alpha}, \overline{\overline{z_i}} := \overline{\overline{\delta_i}}, \overline{\overline{y}} := \overline{\overline{\beta}}_{\overline{\alpha}}\right] \text{ is true} \end{array} \right\}$$

is included in $\perp\!\!\!\perp_{\mathcal{T},k}$.

Let $\overline{\overline{e}}$ be a list of first-order terms with no free variables other than $\overline{w}$ and such that for all $\overline{\alpha}$ in $\{0,1\}$, if $(\overline{a} = \overline{a}')[\overline{w} := \overline{\alpha}]$ is true, then the value of $\overline{\overline{e}}[\overline{w} := \overline{\alpha}]$ (in $\{0,1\}$) is $\overline{\overline{\beta}}_{\overline{\alpha}}$.

Let $i \leq m$. For all $\overline{\alpha}, \overline{\overline{\delta_i}}$ in $\{0,1\}$ such that $((\overline{a} = \overline{a}') \wedge (c_i = c_i')[\overline{\overline{y}} := \overline{\overline{e}}])[\overline{w} := \overline{\alpha}, \overline{\overline{z_i}} := \overline{\overline{\delta_i}}]$ is true, we know that the process

$$(t_i \star \gamma_{C_{i,1}[\overline{\overline{y}} := \overline{\overline{e}}]} \cdot \ldots \cdot \gamma_{C_{i,n_i}[\overline{\overline{y}} := \overline{\overline{e}}]} \cdot \pi)[\overline{w} := \overline{\alpha}]$$

is in $\perp\!\!\!\perp_{\mathcal{T},k}$. Therefore, by the induction hypothesis, we know that the formula

$$\begin{aligned} \exists \overline{w}\ \exists \overline{\overline{z}}_i\ (\mathcal{C}_{t_i} \wedge C_{i,1}[\overline{\overline{y}} := \overline{\overline{e}}] \wedge \ldots \wedge C_{i,n_i}[\overline{\overline{y}} := \overline{\overline{e}}] \\ \wedge (\overline{a} = \overline{a}') \wedge (c_i = c_i')[\overline{\overline{y}} := \overline{\overline{e}}]) \end{aligned}$$

is false in all Boolean algebras with at least two elements. In other words, for all $i \leq m$, the formula

$$\exists \overline{w}\ (\mathcal{C}_{t_i} \wedge (\overline{a} = \overline{a}') \wedge (B_i[\overline{\overline{y}} := \overline{\overline{e}}] \to \perp))$$

is false in all Boolean algebras with at least two elements.

This means that the formula

$$\begin{aligned} \exists \overline{w}\ (\mathcal{C}_{t_1} \wedge \ldots \wedge C_{t_m} \wedge (\overline{a} = \overline{a}') \\ \wedge (B_1 \to \ldots \to B_m \to \perp)[\overline{\overline{y}} := \overline{\overline{e}}]) \end{aligned}$$

is false in all Boolean algebras with at least two elements.

The formula $\forall \overline{w}\ (\overline{a} = \overline{a}' \to (b = b')[\overline{\overline{y}} := \overline{\overline{e}}])$ is true in $\{0,1\}$. Since it is a Horn clause, it is true in all Boolean algebras with at least two elements (Lemma 8). Therefore the formula

$$\begin{aligned} \exists \overline{w}\ (\mathcal{C}_{t_1} \wedge \ldots \wedge C_{t_m} \wedge (\overline{a} = \overline{a}') \\ \wedge (B_1 \to \ldots \to B_m \to b \neq b')[\overline{\overline{y}} := \overline{\overline{e}}]) \end{aligned}$$

is false in all Boolean algebras with at least two elements. This formula is a logical consequence of the formula

$$\exists \overline{w}\ (\mathcal{C}_{t_1} \wedge \ldots \wedge C_{t_m} \wedge (\overline{a} = \overline{a}') \wedge A),$$

which is itself a logical consequence of the formula

$$\exists \overline{w}\ (\mathcal{C}_p \wedge (\overline{a} = \overline{a}')).$$

Therefore, this last formula is false in all Boolean algebras with at least two elements. $\square$

This completes the proof of Theorem 10.

VI. Application: Sequentialisation in a denotational model of the λ_c-calculus

It is known [8] that, by performing Scott's construction D_∞ with $D_0 = \{\perp, \top\}$ (the two-elements lattice), one obtains a denotational model of the λ_c-calculus. As with any such model, one natural question [9] is: among its elements, which ones are sequentialisable. In other words, which ones are the denotation of an actual λ_c-term. We will show how the techniques developed in this paper can give a partial answer.

A. The construction of D_∞

We recall the construction of D_∞ [10]. The first step is to define a finite lattice D_n for all natural numbers n. We let:

- $D_0 = \{\bot, \top\}$ (the two-elements lattice, with $\bot < \top$),
- $D_{n+1} = [D_n \to D_n]$ (the complete lattice of all Scott-continuous functions from D_n to D_n, which is the same as the lattice of all non-decreasing functions, because D_n is finite).

Then for all n we define an injection $\varphi_n \in [D_n \to D_{n+1}]$ and a projection $\psi_n \in [D_{n+1} \to D_n]$:

- for all $\alpha \in D_0$, $\varphi_0(\alpha)$ is the function $\beta \mapsto \alpha$,
- for all $f \in D_1$, $\psi_0(f) = f(\bot)$
- for all $n \geq 0$ and all $\alpha \in D_{n+1}$, $\varphi_{n+1}(\alpha) = \varphi_n \circ \alpha \circ \psi_n$,
- for all $n \geq 0$ and all $f \in D_{n+2}$, $\psi_{n+1}(f) = \psi_n \circ f \circ \varphi_n$.

Finally, we define D_∞ as the limit of the diagram $(D_n, \psi_n)_{n\in\mathbb{N}}$ in the category of complete lattices and Scott-continuous functions, namely:

$$D_\infty = \{\alpha = (\alpha_{[n]} \in D_n)_{n\in\mathbb{N}};\ \forall n\ \alpha_{[n]} = \psi_n(\alpha_{[n+1]})\}.$$

In fact, D_∞ is also a colimit of the diagram $(D_n, \varphi_n)_{n\in\mathbb{N}}$ [10]; for all n, the injection from D_n into D_∞ is given by:

$$\begin{array}{rcl} \alpha_{[n]} & \mapsto & (\psi_0 \circ \ldots \circ \psi_{n-1}(\alpha_{[n]}), \ldots, \psi_{n-1}(\alpha_{[n]})), \\ & & \alpha_{[n]}, \\ & & \varphi_n(\alpha_{[n]}), \varphi_{n+1} \circ \varphi_n(\alpha_{[n]}), \ldots). \end{array}$$

As is customary with colimits, we identify each D_n with the corresponding subset of D_∞.

This defines an extensional reflexive object in the category of complete lattices and Scott-continuous functions, because we can define two inverse isomorphisms $\Phi : D_\infty \to [D_\infty \to D_\infty]$ and $\Psi : [D_\infty \to D_\infty] \to D_\infty$:

$$\begin{array}{rcl} \Phi((\alpha_{[n]})_{n\in\mathbb{N}}) & = & (\beta[n])_{n\in\mathbb{N}} \mapsto (\alpha_{[n+1]}(\beta_{[n]}))_{n\in\mathbb{N}} \\ \Psi(f) & = & (\gamma_{[n]})_{n\in\mathbb{N}}, \text{ where} \\ & & \gamma_{[0]} = f(\bot)_{[0]} \in D_0, \\ & & \gamma_{[n+1]} = (\alpha_{[n]} \mapsto f(\alpha_{[n]})_{[n]}) \in D_{n+1}. \end{array}$$

A model of the λ_c-calculus: It is known [8] that D_∞ can be equipped with the structure of a model of the λ_c-calculus. One way to present this structure is as follows: for each λ_c-term t and each list $x_1, \ldots, x_n$ of pairwise distinct variables containing at least all the free variables of t, define a Scott-continuous function $[\![t]\!] : D_\infty^n \to D_\infty$:

$$\begin{array}{rcl} [\![x_k]\!](\alpha_1, \ldots, \alpha_n) & = & \alpha_k \\ [\![\lambda x_{n+1}.\, t]\!](\alpha_1, \ldots, \alpha_n) & = & \Psi(\alpha_{n+1} \mapsto [\![t]\!](\alpha_1, \ldots, \alpha_{n+1})) \\ [\![tu]\!](\overline{\alpha}) & = & \Phi([\![t]\!](\overline{\alpha}))([\![u]\!](\overline{\alpha})) \\ [\![\zeta_m]\!](\overline{\alpha}) = [\![\eta_m]\!](\overline{\alpha}) & = & \bot \\ [\![k_{t_1\cdot\ldots\cdot t_m\cdot\omega}]\!](\overline{\alpha}) & = & [\![\lambda y.\, y\, t_1 \ldots t_n]\!](\overline{\alpha}) \\ [\![\mathrm{cc}]\!](\overline{\alpha}) & = & \bigvee_{\beta,\gamma\in D_\infty}((\beta \to \gamma) \to \beta) \to \beta, \\ & & \text{where } \delta \to \varepsilon \text{ is the least} \\ & & \text{element of } D_\infty \text{such that} \\ & & \Phi(\delta \to \varepsilon)(\delta) \geq \varepsilon. \end{array}$$

This structure is compatible with evaluation in the Krivine abstract machine [8]: for all close λ_c-terms $t, t', u_1 \ldots, u_n, u'_1 \ldots, u'_{n'}$, if

$$t \star u_1 \cdot \ldots u_n \cdot \omega \succ t' \star u'_1 \cdot \ldots u'_{n'} \cdot \omega,$$

then $[\![t\, u_1 \ldots u_n]\!] = [\![t'\, u'_1 \ldots u'_{n'}]\!]$.

In addition, this model charaterises solvability [8]. Namely, for each closed term t, we have $[\![t]\!] > \bot$ if and only if there exist $k \leq n \in \mathbb{N}$ such that for each stack $u_1 \cdot \ldots \cdot u_n \cdot \pi$, there exists a stack π' such that

$$t \star u_1 \cdot \ldots \cdot u_n \cdot \pi \succ u_k \star \pi'.$$

B. Sequentialisation

In this context, the problem of sequentialisation can be formulated as follows: given $\alpha \in D_\infty$, is there a closed λ_c-term t such that $[\![t]\!] = \alpha$? We will show how the techniques developed in this paper can answer a simplified version of this problem. Namely, whenever α is in D_n for some finite n, we give a necessary and sufficient condition for the existence of a closed λ_c-term t such that $[\![t]\!] \geq \alpha$.

Remark 16. Alternatively, one might also ask whether there exists a *proof-like* term t such that $[\![t]\!] \geq \alpha$. However, due to how $[\![\eta_m]\!]$ and $[\![k_\pi]\!]$ are defined, we have that for each closed λ_c-term t, there exists a proof-like term t' such that $[\![t]\!] = [\![t']\!]$, so that is in fact the same question.

Remark 17. One can prove that the set $\bigcup_{n\in\mathbb{N}} D_n$ is in fact the least subset of D_∞ that contains $\bot$ and $\top$ and is closed under the binary operations $\vee$ and $\to$ (where $\delta \to \varepsilon$ is defined as the least element of D_∞ such that $\Phi(\delta \to \varepsilon)(\delta) \geq \varepsilon$).

Interpreting first-order formulas in D_∞: For each closed first-order formula A, we can define an element $[\![A]\!] \in D_\infty$:

- $[\![a \neq b]\!] = \begin{cases} \bot & \text{if } a \neq b \text{ is true,} \\ \top & \text{if } a \neq b \text{ is false} \end{cases}$
- $[\![A \to B]\!] = [\![A]\!] \to [\![B]\!]$,
- $[\![\forall x\, A]\!] = [\![A]\!](0) \vee [\![A]\!](1)$

In fact, for each A, there exists n such that $[\![A]\!]$ is in D_n. Conversely, thanks to Remark 17, for all n and all $\alpha \in D_n$, one can construct inductively a closed formula Θ_α such that $[\![\Theta_\alpha]\!] = \alpha$.

True formulas give sequentialisable elements: These two translations, from λ_c-terms and first-order formulas into D_∞, are linked by a variant of the adequacy lemma (Lemma 2), which can be proved using the same standard techniques:

Lemma 18. *Let $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ be a valid typing judgement (with $A_1, \ldots, A_n$ closed), and let $u_1, \ldots, u_n$ be closed λ_c-terms. If $[\![u_1]\!] \geq [\![A_1]\!], \ldots, [\![u_n]\!] \geq [\![A_n]\!]$, then $[\![t]\!]([\![u_1]\!], \ldots, [\![u_n]\!]) \geq [\![B]\!]$.*

In addition, a variant of Lemma 6 also holds in D_∞ (with essentially the same proof):

Lemma 19. *Let A be a closed first-order formula that is true in the Boolean algebra $\{0, 1\}$.*

- *If A is of the form $\forall \overline{z}\ a \neq b$, $[\![A]\!] = \bot \leq [\![t]\!]$ for all closed λ_c-terms t;*
- *If A is of the form $\forall \overline{z}\ a = b$, then $[\![A]\!] = (\top \to \top) \leq [\![\lambda x.\, x]\!]$.*

As a result, for each closed first-formula A, if A is true in all Boolean algebras with at least two elements, then there exists a (proof-like) closed λ_c-term t such that $[\![t]\!] \geq [\![A]\!]$.

Sequentialisation gives universal realizers: For each closed λ_c-term t and each closed first-order formula A, one can prove by induction on the structure of t that if $[\![t]\!] \geq [\![A]\!]$, then t realizes A universally. Thanks to Theorem 10 (and Remark 17), this means that for all closed first-order formulas A, if there exists a closed λ_c-term t such that $[\![t]\!] \geq [\![A]\!]$, then A is true in all Boolean algebras with at least two elements. In addition, for all closed first-order formula A, B, if $[\![A]\!] = [\![B]\!]$, then A and B are equivalent in all Boolean algebras with at least two elements (because $[\![A \to B]\!] = [\![B \to A]\!] \leq [\![\lambda x.\, x]\!]$, so both $A \to B$ and $B \to A$ are universally realized by $\lambda x.\, x$).

Combining all these results, we get:

Proposition 20. *Let $n \in \mathbb{N}$ and $\alpha \in D_n$. The following two statements are equivalent:*

- *There exists a closed λ_c-term t such that $[\![t]\!] \geq \alpha$,*
- *The formula Θ_α is true in all Boolean algebras with at least two elements (where Θ_α is any closed first-order formula such that $[\![\Theta_\alpha]\!] = \alpha$. Such a Θ_α does exist for all α and it can be obtained effectively. The choice of Θ_α does not matter).*

VII. Concluding remarks

We have proved that the only *first-order* formulas that are true in the characteristic Boolean algebra ($\gimel 2$) of every classical realizability model are those that are true in all Boolean algebras with at least two elements. In a sense, as far as the first order is concerned, the only thing we always know about $\gimel 2$ is that it is a Boolean algebra with at least two elements. This does not extend to the second order: indeed, for example, Krivine [11] has proved that there always exists an ultrafilter on $\gimel 2$, even though the axiom of choice does not necessarily hold in a realizability model. This raises the question: what are the second- and higher-order properties of $\gimel 2$ that are true in all realizability models? And what about $\gimel \mathbb{N}$?

In a different direction, it would be interesting to know if and to what extent the technique presented in section VI can be adapted to other denotational models of the lambda-calculus, and notably to non-lattice and non-continuations-based models.

References

[1] T. G. Griffin, "A formulae-as-type notion of control," in *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '90. New York, NY, USA: ACM, 1990, pp. 47–58. [Online]. Available: http://doi.acm.org/10.1145/96709.96714

[2] J.-L. Krivine, "Dependent choice, 'quote' and the clock," *Theor. Comput. Sci.*, vol. 308, no. 1-3, pp. 259–276, Nov. 2003. [Online]. Available: http://dx.doi.org/10.1016/S0304-3975(02)00776-4

[3] J. Krivine, "Realizability algebras: a program to well order R," *Logical Methods in Computer Science*, vol. 7, no. 3:02, pp. 1–47, 2011. [Online]. Available: https://doi.org/10.2168/LMCS-7(3:2)2011

[4] A. Miquel, "Forcing as a Program Transformation," in *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011.* Toronto, Canada: IEEE Computer Society, Jun. 2011, pp. 197–206. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00800558

[5] J.-L. Krivine, "A program for the full axiom of choice," *Logical Methods in Computer Science*, vol. Volume 17, Issue 3, Sep 2021. [Online]. Available: http://dx.doi.org/10.46298/lmcs-17(3:21)2021

[6] ——, "Realizability algebras II : new models of ZF + DC," *Logical Methods in Computer Science*, vol. 8, no. 1:10, pp. 1–28, Feb. 2012. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00497587

[7] G. Geoffroy, "Classical realizability as a classifier for nondeterminism," *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, Jul 2018. [Online]. Available: http://dx.doi.org/10.1145/3209108.3209140

[8] T. Streicher and B. Reus, "Classical logic, continuation semantics and abstract machines," *Journal of Functional Programming*, vol. 8, no. 6, p. 543–572, 1998.

[9] D. S. Scott, "A type-theoretical alternative to iswim, cuch, owhy." *Theor. Comput. Sci.*, vol. 121, no. 1&2, pp. 411–440, 1993, first written 1969. [Online]. Available: http://dblp.uni-trier.de/db/journals/tcs/tcs121.html#Scott93

[10] H. P. Barendregt, *The lambda calculus - its syntax and semantics*, ser. Studies in logic and the foundations of mathematics. North-Holland, 1985, vol. 103.

[11] J. Krivine, "On the structure of classical realizability models of ZF," in *Proceedings TYPES 2014 - LIPIcs*, vol. 39, 2015, pp. 146–161. [Online]. Available: http://arxiv.org/abs/1408.1868