Classical realizability as a classifier for nondeterminism

Guillaume Geoffroy
guillaume.geoffroy@univ-amu.fr
Institut de mathématiques de Marseille
Aix-Marseille Université
France

Abstract

We show how the language of Krivine's classical realizability may be used to specify various forms of nondeterminism and relate them with properties of realizability models. More specifically, we introduce an abstract notion of multi-evaluation relation which allows us to finely describe various nondeterministic behaviours. This defines a hierarchy of computational models, ordered by their degree of nondeterminism, similar to Sazonov's degrees of parallelism. What we show is a duality between the structure of the characteristic Boolean algebra of a realizability model and the degree of nondeterminism in its underlying computational model.

ACM Reference Format:
Guillaume Geoffroy. 2018. Classical realizability as a classifier for nondeterminism. In *LICS '18: LICS '18: 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, July 9–12, 2018, Oxford, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209108.3209140

1 Introduction

Classical realizability Realizability is an instance of the formulas-as-types/proofs-as-programs correspondence in which each formula is interpreted as the specification of a certain computational behaviour. Initially, this correspondence (and hence realizability) was limited to intuitionistic reasoning, until Griffin noted a connection between control operators and classical reasoning [4], which lifted this limitation. Classical realizability is an extension of Kleene's realizability to accommodate classical reasoning: using control operators, Krivine developed a theory capable of interpreting all classical reasoning within *second-order arithmetic* ([11, 12]) and *Zermelo–Fraenkel set theory* with dependent choice ([6–9, 13]). Subsequently, Miquel adapted this framework to higher-order arithmetic [15].

Following intuitionistic realizability, Krivine's framework is made of three ingredients:

- A computational model (in Krivine's setting, it is called a *realizability algebra*). In general, it is a set of programs (with an operational semantics);
- A logical language (for arithmetic, higher-order arithmetic or set theory), together with a *realizability relation* expressing the fact that a given program realizes a given formula. The essential result is the *adequacy lemma* which states that classical realizability is compatible with deduction rules;
- A realizability model (in the usual sense of model theory), which satisfies all formulas that are realized. Such a model must exist by the completeness theorem. Krivine noted that this construction may be seen as an extension of Cohen's forcing construction [2]. Here, we will not look at it directly, nor even define it explicitly: we will look at realizability theories (*i.e.* consistent sets of realized formulas) rather than realizability models.

In Krivine's framework, each realizability model comes with a *characteristic Boolean algebra* $\beth 2$ ("gimel 2") [13] whose structure encodes interesting properties of the model. In particular, forcing models correspond to the degenerate case where $\beth 2 = \{0, 1\}$.

In order to emphasise the central role of the characteristic Boolean algebra, let us recall that classical realizability gives rise to surprising models of set theory (such as the *model of threads* [13]), whose strange set-theoretic properties are mostly direct consequences of the structure of their characteristic Boolean algebras.

As noted above, classical realizability can be seen as an extension of forcing. However, while there are plenty of theoretical results connecting the properties of a forcing model to the structure of the underlying forcing set, there is currently a severe lack of general results connecting the properties of a classical realizability model to the properties of the underlying computational model. The goal of this paper is to start addressing this state of affairs by two means. First, by proving new results of this kind (namely, results which connect the presence of nondeterminism in the computational model with the size of the characteristic Boolean algebra in the realizability model), and second, by introducing a new framework which should make it easier, in the future, to find such results.

Nondeterminism The logical language can be extended by adding realizability connectives (such as union "$\cup$" and intersection "$\cap$"), which may in particular be used to express various forms of nondeterminism in the computational model.

For example, we will see that the formula $\forall X \forall Y\ X \to Y \to X \cap Y$ specifies the *may*-nondeterministic choice operator "fork", which takes two programs as arguments and does whatever *either* does. Dually the formula $\forall X \forall Y\ X \to Y \to X \cup Y$ specifies the *must*-nondeterministic choice operator "choose", which takes two programs as arguments and only does what *both* do.

It is known that adding a must-nondeterministic choice operator does not change the realizability model (no new formulas are realized), and that adding a may-nondeterministic choice operator collapses it into a forcing model.

However, we show that adding more subtle nondeterministic instructions corresponding to different mixes of *may* and *must* has the effect of altering the properties of $\beth 2$. To this end, we define an abstract notion of *multi-evaluation relation* which can express arbitrary mixes of may- and must-nondeterminism such as Kleene's

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
LICS '18, July 9–12, 2018, Oxford, United Kingdom
© 2018 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN 978-1-4503-5583-4/18/07…$15.00
https://doi.org/10.1145/3209108.3209140

"*parallel or*" [5], Berry's "*Gustave's function*" [1], and Trakhtenbrot's "*voting function*" [18], as well as generalisations thereof.

Outline We recall Krivine's formalisation of classical realizability – in the context of second-order arithmetic – (sections 2 and 3). Our presentation follows closely those given by Krivine and Miquel, except that, instead of individual poles, we consider sets of poles (which we call *realizability structures*), a formula being realized if it has a common realizer for all the poles in the structure. The rationale behind this is that realizability structures are a notion dual to multi-evaluation relations.

Then, we define the notion of *multi-evaluation relation* (section 4) and show that each multi-evaluation relation defines a unique realizability structure.

Next, we analyse in detail particular examples of such mixes, to notice that they are organised in a hierarchy, in terms of both their computational expressiveness and the properties of $\daleth 2$ to which they correspond (sections 6 and 5). This hierarchy is reminiscent of Sazonov's *degrees of parallelism* [17].

Finally, we prove that this hierarchy does not collapse: each level is indeed strictly more expressive than the levels below (section 7). To this end, we prove that $\daleth 2$ can be made elementarily equivalent to any finite Boolean algebra with at least two elements. The case of Boolean algebras with 4 elements was stated and proved by Krivine [7], but the method used here is new.

The contributions of this paper are:

- The dual notions of *realizability structures* and *multi-evaluation* relations,
- The results connecting the size of $\daleth 2$ to voting functions, *parallel or* and Gustave's function (the result about *fork* is due to Krivine),
- The method presented in section 7, which fits nicely in the framework developed here, and will be used in some future work to prove the same result about all Boolean algebras, not just the finite ones.

2 Classical realizability semantics

Following Krivine [13], we define the model of computation and the logical language which will be used throughout this paper, and we connect them through the classical realizability interpretation.

2.1 The computational model: the λ_c-calculus

The λ_c-calculus is a model of computation which extends pure λ-calculus (with a specific evaluation strategy, namely: weak head β-reduction) by the addition of a *control operator* cc (*call-with-current-continuation*). It is made up of three kinds of syntactic entities: *λ_c-terms*, *stacks* and *processes*.

Definition 1. Let us fix a countably infinite set of variables. The sets of *λ_c-terms*, *stacks* and *processes* are defined by the following grammars, modulo α-equivalence (free and bound variables are defined as usual, abstraction being the only binding construction):

λ_c-terms:

$$\begin{array}{rcll} t,u & ::= & x & (x \text{ variable}) \\ & | & tu & (t,u\ \lambda_c\text{-terms} - \text{application}) \\ & | & \lambda x.t & (x \text{ variable}, t\ \lambda_c\text{-term} - \text{abstraction}) \\ & | & cc & (\textit{call-with-current-continuation}) \\ & | & k_\pi & (\pi \text{ stack} - \text{continuation constants}) \\ & | & \xi_n & (n \in \mathbb{N} - \text{nonrestricted instructions}) \\ & | & \eta_n & (n \in \mathbb{N} - \text{restricted instructions}), \end{array}$$

Stacks:

$$\begin{array}{rcll} \pi & ::= & \omega_n & (n \in \mathbb{N} - \text{stack bottoms}) \\ & | & t \cdot \pi & (t \textit{ closed } \lambda_c\text{-term}, \pi \text{ stack}), \end{array}$$

Processes:

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

The additional instructions (*i.e.* ξ_n and η_n) will serve as customizable instructions: the *evaluation relation of the λ_c-calculus*, which we will define in this section, gives no evaluation rule for them, and therefore treats them as inert constants. On the other hand, in section 4, we will define *multi-evaluation relations*, which can specify evaluation rules for these instructions: thus, the meaning of a given additional instruction will depend on the particular evaluation relation which we are considering at the moment. Typically, they will represent nondeterministic choice operators.

Intuitively, restricted instructions correspond to *privileged instructions* of real-world processors (which cannot be called directly be the user), while unrestricted instructions correspond to *system calls* (through which the user can access privileged instructions, in a controlled fashion).

From now on, closed λ_c-terms will be simply called *terms*.

The set of terms is denoted by Λ, the set of stacks by Π and the set of processes by $\Lambda \star \Pi$.

If $t \star \pi$ is a process, we call t its *head term* and π its *stack*.

Application is left-associative (so $tuvw$ means $((tu)v)w$) and has higher priority than abstraction (so $\lambda x.tu$ means $\lambda x.(tu)$). We write $t^n u$ for t applied n times to u.

Substitutions [10]. Given any list of λ_c-terms $t, u_1, \ldots, u_n$ and any list of distinct variables $x_1, \ldots, x_n$, we will denote by $t[x_1 := u_1, \ldots, x_n := u_n]$ the λ_c-term obtained by replacing *simultaneously* each free occurrence of x_i by u_i in t for $i = 1, \ldots, n$.

Definition 2. The *evaluation relation* of the λ_c-calculus, denoted by $\succ_{\mathrm{K}}$, is the smallest *preorder* on $\Lambda \star \Pi$ satisfying the following rules:

$$\begin{array}{rcll} tu \star \pi & \succ_{\mathrm{K}} & t \star u \cdot \pi & \text{(push)} \\ \lambda x.t \star u \cdot \pi & \succ_{\mathrm{K}} & t[x := u] \star \pi & \text{(grab)} \\ cc \star t \cdot \pi & \succ_{\mathrm{K}} & t \star k_\pi \cdot \pi & \text{(save)} \\ k_{\pi'} \star t \cdot \pi & \succ_{\mathrm{K}} & t \star \pi' & \text{(restore)} \end{array}$$

The one-step evaluation relation of λ_c-calculus, denoted by $\succ_{\mathrm{K}}^1$, is the smallest *binary relation* on $\Lambda \star \Pi$ satisfying these rules.

The *push* and *grab* rules simulate *weak head β-reduction*; they will make the realizability interpretation compatible with *intuitionistic logic*. The *save* and *restore* rules allow a program (*i.e.* a term) to save its evaluation context (*i.e.* the stack), and restore it later; they will make the realizability interpretation compatible with *classical logic*.

Remark. *The one-step evaluation relation is deterministic: for all p, q, q', if $p \succ_{\mathrm{K}}^1 q$ and $p \succ_{\mathrm{K}}^1 q'$, then $q = q'$.*

2.2 The realizability language

The realizability language is a second-order logical language, whose first-order terms are intended to represent integers.

Definition 3. We fix a set of first-order variables, and for each natural number n, we fix a set of n-ary propositional variables. Those sets are taken pairwise disjoint and countably infinite. The sets of *first-order terms* and of *formulas* are described by the following grammars, modulo α-equivalence (the binding constructions being first- and second-order universal quantifications).

First order terms:

$$\begin{array}{rcll} a, b & ::= & x & (x \text{ first-order variable}) \\ & | & f(a_1, \ldots, a_n) & (f : \mathbb{N}^n \to \mathbb{N}), \end{array}$$

Formulas:

$$\begin{array}{rcl} A, B & ::= & X(a_1, \ldots, a_n) \quad (X \ n\text{-ary relational variable}) \\ & | & \top \mid \bot \mid A \to B \mid \forall x\, A \mid \forall X\, A \\ & | & (a = b) \hookrightarrow A \mid A \cap B \mid A \cup B \\ & | & F(a_1, \ldots, a_n) \quad (F : \mathbb{N}^n \to \mathcal{P}(\Pi)). \end{array}$$

All formulas can be interpreted as describing program behaviours. In addition, formulas which only contain first-order terms, propositional variables, $\top$, $\bot$, $\to$ and $\forall$ also have a logical meaning: they can be interpreted in $\mathbb{N}$ (or any model of second-order arithmetic). These formulas are called *arithmetic formulas*.

The construction $(a = b) \hookrightarrow A$ is called *equational implcation*. As we will see later, it is logically equivalent to regular implication ($(a = b) \to A$), and its interest lies in the fact that realizers of $(a = b) \hookrightarrow A$ are easier to read, write and understand than realizers of $(a = b) \to A$ (they involve less "red tape").

We write $a \neq b$ for $(a = b) \hookrightarrow \bot$.

Caution: we use the same letters for first-order variables and variables of λ_c-calculus. It will always be clear from context which is which.

Additional connectives. Equality and additional logical connectives are defined by the usual second-order encodings:

- $a = b$ means $\forall Z,\ Z(a) \to Z(b)$;
- $A \wedge B$ means $\forall Z,\ (A \to B \to Z) \to Z$;
- $A \vee B$ means $\forall Z,\ (A \to Z) \to (B \to Z) \to Z$;
- $\neg A$ means $A \to \bot$;
- $A \leftrightarrow B$ means $(A \to B) \wedge (B \to A)$;
- $\exists x\, A$ means $\forall Z,\ (\forall x,\ A \to Z) \to Z$;
- $\exists X\, A$ means $\forall Z,\ (\forall X,\ A \to Z) \to Z$.

Connectives are, from highest to lowest precedence: $\neg$, $\neq$, $=$, $\cap$, $\cup$, $\wedge$, $\vee$, $\to$, $\hookrightarrow$, $\leftrightarrow$, $\exists$ and $\forall$. In addition, $\wedge$ $\vee$, $\cap$ and $\cup$ are left-associative, and $\to$ is right-associative.

First-order substitutions. Given any formula or first-order term α, any list of first-order terms $b_1, \ldots, b_n$ and any list of distinct first-order variables $x_1, \ldots, x_n$ we denote by $\alpha[x_1 := u_1, \ldots, x_n := u_n]$ the formula or the first-order term obtained by replacing simultaneously each free occurrence of x_i by b_i in α for $i = 1, \ldots, n$.

Second-order substitutions. Given any two formulas A, B, any n-ary propositional variable X and any list of distinct first-order variables $y_1, \ldots, y_n$, we denote by $A[X(y_1, \ldots, y_n) := B]$ the formula obtained by replacing X by B in A. Specifically, each free occurence of X of the form $X(a_1, \ldots, a_n)$, is replaced by $B[y_1 := a_1, \ldots, y_n := a_n]$.

Formulas and terms with parameters. Given any formula or first-order term α and any list of distinct first-order variables $x_1, \ldots, x_n$ containing at least all the free variables of α, we will sometimes denote α by $\alpha(x_1, \ldots, x_n)$ (the point of this notation is to order the free variables of α). In that case, given any list of first-order terms $a_1, \ldots, a_n$ we will write $\alpha(a_1, \ldots, a_n)$ for $\alpha[x_1 := a_1, \ldots, x_n := a_n]$.

When writing first-order terms, we will use all sorts of abuses of notation, such as writing $a + b$ instead of $+(a, b)$, or $\sum_{i=1}^{n} a_i$ instead of $+(a_1, \ldots, a_n)$, etc.

In addition, if $A_1, \ldots A_n$ are formulas and $\odot$ is $\wedge$, $\vee$, $\cap$ or $\cup$, we will write $\bigodot_{i=1}^{n} A_i$ for $A_1 \odot \ldots \odot A_n$.

2.3 Classical realizability

Definition 4. A *pole* is a set of processes which is *closed by anti-evaluation*, that is to say, a set $\perp\!\!\!\perp \subseteq \Lambda \star \Pi$ such that for all $p, q \in \Lambda \star \Pi$, if $p \succ_K q$ and $q \in \perp\!\!\!\perp$, then $p \in \perp\!\!\!\perp$. The set of all poles is denoted by $\mathcal{S}_0$.

Definition 5. Let a be a closed first-order term. The *value* of a, written $v(a)$ is the integer defined (inductively) by $v(f(a_1, \ldots, a_n)) = f(v(a_1), \ldots, v(a_n))$.

Definition 6. Let $\perp\!\!\!\perp$ be a pole and X a subset of Π. The *dual of X with respect to* $\perp\!\!\!\perp$, denoted by $X^{\perp\!\!\!\perp}$, is the set $\{t \in \Lambda; \forall \pi \in X,\ t \star \pi \in \perp\!\!\!\perp\}$.

Definition 7. Let $\perp\!\!\!\perp$ be a pole and A a closed formula. The *falsity value of A with respect to* $\perp\!\!\!\perp$, denoted by $\|A\|_{\perp\!\!\!\perp}$, is the subset of Π defined below, and the *truth value of A with respect to* $\perp\!\!\!\perp$, denoted by $|A|_{\perp\!\!\!\perp}$, is defined as $(\|A\|_{\perp\!\!\!\perp})^{\perp\!\!\!\perp}$.

Falsity values of closed formulas are defined as follows:

- $\|\top\|_{\perp\!\!\!\perp} = \emptyset$, $\|\bot\|_{\perp\!\!\!\perp} = \Pi$;
- $\|A \to B\|_{\perp\!\!\!\perp} = \{t \cdot \pi; t \in |A|_{\perp\!\!\!\perp}, \pi \in \|B\|_{\perp\!\!\!\perp}\}$;
- $\|\forall x\, A\|_{\perp\!\!\!\perp} = \bigcup_{n \in \mathbb{N}} \|A[x := n]\|_{\perp\!\!\!\perp}$;
- $\|\forall X\, A\|_{\perp\!\!\!\perp} = \bigcup_{F : \mathbb{N}^n \to \mathcal{P}(\Pi)} \|A[X(y_1, \ldots, y_n) := F(y_1, \ldots, y_n)]\|_{\perp\!\!\!\perp}$
- $\|(a = b) \hookrightarrow A\|_{\perp\!\!\!\perp} = \begin{cases} \|A\|_{\perp\!\!\!\perp} & \text{if } v(a) = v(b) \\ \|\top\|_{\perp\!\!\!\perp} & \text{otherwise;} \end{cases}$
- $\|A \cap B\|_{\perp\!\!\!\perp} = \|A\|_{\perp\!\!\!\perp} \cup \|B\|_{\perp\!\!\!\perp}$;
- $\|A \cup B\|_{\perp\!\!\!\perp} = \|A\|_{\perp\!\!\!\perp} \cap \|B\|_{\perp\!\!\!\perp}$;
- $\|F(a_1, \ldots, a_n)\|_{\perp\!\!\!\perp} = F(v(a_1), \ldots, v(a_n))$.

Notation. Let A be a closed formula. We denote by $\|A\|$ the function which maps $\perp\!\!\!\perp$ to $\|A\|_{\perp\!\!\!\perp}$, and by $|A|$ the function which maps $\perp\!\!\!\perp$ to $|A|_{\perp\!\!\!\perp}$.

Definition 8. Let A be a closed formula, t a term and $\perp\!\!\!\perp$ a pole. If $t \in |A|_{\perp\!\!\!\perp}$, we say that t *realizes A with respect to* $\perp\!\!\!\perp$, and we write $t \Vdash_{\perp\!\!\!\perp} A$.

Definition 9. Let A be a closed formula and t a term. We say that t *realizes A universally* if t realizes A with respect to every pole.

Remark. *Let $\perp\!\!\!\perp$ be a pole. For all $X, Y \subseteq \Pi$, if $X \subseteq Y$, then $X^{\perp\!\!\!\perp} \supseteq Y^{\perp\!\!\!\perp}$. Therefore, for all closed formulas A and B, if $\|A\|_{\perp\!\!\!\perp} \subseteq \|B\|_{\perp\!\!\!\perp}$, then $|A|_{\perp\!\!\!\perp} \supseteq |B|_{\perp\!\!\!\perp}$ – in particular, the identity term $\lambda x.x$ realizes $B \to A$.*

Semantic equivalence and semantic subtyping [16]. Let A and B be two closed formulas. If $\perp\!\!\!\perp$ is a pole, we write $A \approx_{\perp\!\!\!\perp} B$ if $\|A\|_{\perp\!\!\!\perp} = \|B\|_{\perp\!\!\!\perp}$ and $A \lesssim_{\perp\!\!\!\perp} B$ if $\|A\|_{\perp\!\!\!\perp} \supseteq \|B\|_{\perp\!\!\!\perp}$. Moreover, we write $A \approx B$ if $A \approx_{\perp\!\!\!\perp} B$ for all $\perp\!\!\!\perp$, and $A \lesssim B$ if $A \lesssim_{\perp\!\!\!\perp} B$ for all $\perp\!\!\!\perp$.

2.4 Adequacy

2.4.1 Typing λ_c-terms

We type λ_c-terms with (non-necessarily closed) formulas of the realizability language.

A *context* is a finite set of *hypotheses* of the form $x : B$ (with x a variable of the λ_c-calculus and B a formula) such that no variable of the λ_c-calculus appears more than once. Variables of the λ_c-calculus which appear on the left of an hypothesis are said to be *declared* in the context, and variables of the realizability language which appear freely on the right of at least one hypothesis are said to be *free* in the context.

Typing judgments are sequents of the form $\Gamma \vdash t : A$, where Γ is a context, t is a λ_c-term whose free variables are all declared in Γ, and A is a formula.

A *typing derivation* is a tree formed with the following *typing rules*. Its root is called its *conclusion*:

$$\text{(Axiom)}\ \frac{}{\Gamma, x : A \vdash x : A}$$

$$\text{(Peirce)}\ \frac{}{\Gamma \vdash cc : ((A \to B) \to A) \to A}$$

$$(\top\text{-intro})\ \frac{}{\Gamma \vdash t : \top} \quad (\bot\text{-elim})\ \frac{\Gamma \vdash t : \bot}{\Gamma \vdash t : A}$$

$$(\to\text{-intro})\ \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}$$

$$(\to\text{-elim})\ \frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash tu : B}$$

$$(\forall_1\text{-intro})\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall x\, A}\ (x \text{ not free in } \Gamma)$$

$$(\forall_1\text{-elim})\ \frac{\Gamma \vdash t : \forall x\, A}{\Gamma \vdash t : A[x := a]}$$

$$(\forall_2\text{-intro})\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall X\, A}\ (X \text{ not free in } \Gamma)$$

$$(\forall_2\text{-elim})\ \frac{\Gamma \vdash t : \forall X\, A}{\Gamma \vdash t : A[X(y_1, \ldots, y_n) := B]}$$

These are the usual rules of intuitionistic natural deduction, plus the rule that cc is typed by Peirce's law.

A sequent is said to be *derivable* if it is the conclusion of some derivation.

The following lemma states the compatibility of classical realizability with respect to deduction rules:

Proposition 10 (Adequacy lemma)**.** *Let $\perp\!\!\!\perp$ be a pole, $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ a derivable sequent with $A_1, \ldots A_n, B$ closed, and $u_1, \ldots, u_n$ terms such that u_i realizes A_i w.r.t. $\perp\!\!\!\perp$ for all i. The term $t[x_1 := u_1, \ldots, x_n := u_n]$ realizes B w.r.t. $\perp\!\!\!\perp$.*

The proof can be found in [13] (although in a set-theoretical rather than second-order-arithmetical setting).

2.5 Realizability structures

So far, we have considered realizability with respect to a fixed pole and uniform realizability across all poles (*i.e.* universal realizablity). However, in order to state the results of this paper, we will need to consider uniform realizability across arbitrary sets of poles:

Definition 11. A *realizability structure* is a set of poles.

Definition 12. Let A be a closed formula, t a term and $\mathcal{S}$ a realizability structure. If $t \Vdash_{\perp\!\!\!\perp} A$ for all $\perp\!\!\!\perp \in \mathcal{S}$, we say that t *realizes A with respect to $\mathcal{S}$*, and we write $t \Vdash_{\mathcal{S}} A$.

Remark. *To simplify notation, when we have fixed a realizability structure $\mathcal{S}$ but no individual pole, we shall say that t realizes A and write $t \Vdash A$ to mean $t \Vdash_{\mathcal{S}} A$, and when we have fixed a pole $\perp\!\!\!\perp$, we shall say that t realizes A and write $t \Vdash A$ to mean $t \Vdash_{\perp\!\!\!\perp} A$.*

Now, we might be tempted to define the *theory* of a realizability structure as the set of all formulas which have a realizer with respect to this structure. However, an important feature of classical (as opposed to intuitionistic) realizability is that, given any non-empty pole $\perp\!\!\!\perp$, there are terms which realize $\bot$ with respect to $\perp\!\!\!\perp$. Indeed, take $t \star \pi \in \perp\!\!\!\perp$ and form the term $k_\pi t$: when evaluated, this term ignores its context and replaces it with the "winning" context π, therefore it realises $\bot$ with respect to $\perp\!\!\!\perp$. As a result, to avoid getting inconsistent theories, we will consider the following restriction:

Definition 13. The set of *proof-like terms*, denoted by PL, is the set of all terms which do not contain any continuation constant (k_π) nor any restricted instruction (η_n).

We also exclude *restricted* instructions because in some cases (*e.g.* in section 7), it will be convenient to require some of them to realize $\bot$ (or other inconsistent formulas), and we want to do so without breaking the logic.

Definition 14. Let $\mathcal{S}$ be a realizability structure. The *theory generated by $\mathcal{S}$*, denoted by $\mathrm{Th}(\mathcal{S})$, is the set of all closed formulas A such that there exists a *proof-like term t* which realizes A with respect to $\mathcal{S}$.

Intuitively, the adequacy lemma means that $\mathrm{Th}(\mathcal{S})$ is *closed by the rules of classical deduction*. Therefore, we will write $\mathrm{Th}(\mathcal{S}) \models A$ for $A \in \mathrm{Th}(\mathcal{S})$, and we will say that $\mathrm{Th}(\mathcal{S})$ is *inconsistent* if it contains $\bot$, and *consistent* if it does not. Likewise, we will say that the structure $\mathcal{S}$ is consistent (respectively, inconsistent) if $\mathrm{Th}(\mathcal{S})$ is.

From now on, we will say that a formula is *universally realized* if it is universally realized *by a proof-like term*.

Definition 15. Two formulas $A(X_1, \ldots, X_m, y_1, \ldots, y_n)$ and $B(X_1, \ldots, X_m, y_1, \ldots, y_n)$ are *universally equivalent* if the formula $\forall X_1 \ldots X_m\ \forall y_1 \ldots y_n\ A \leftrightarrow B$ is universally realized.

Remark. *A realizability structure $\mathcal{S}$ is consistent if and only if there is no proof-like term t such that $t \star \pi \in \perp\!\!\!\perp$ for all $\perp\!\!\!\perp \in \mathcal{S}$ and all $\pi \in \Pi$.*

Remark. *It is well-known that if $\perp\!\!\!\perp = \emptyset$, then the truth value of an arithmetic formula A is Λ if A is true in $\mathbb{N}$, and $\emptyset$ else [14]. In particular, an arithmetic formula which is universally realized must be true in $\mathbb{N}$. However, the converse is false, as we will shortly see.*

Theories of the form $\mathrm{Th}(\mathcal{S})$, which we will call *realizability theories*, will be our main object of study: what are their properties, and how do these relate to the properties of the generating realizability structures?

2.6 Equations, inequations and equational implications

Equations and inequations are preserved by classical realizability, in the following sense:

Lemma 16. *Let a and b be closed first-order terms.*

- $\|a \neq b\| = \begin{cases} \|\top\| & \text{if } v(a) \neq v(b), \\ \|\bot\| & \text{otherwise.} \end{cases}$
- $\|a = b\| = \begin{cases} \|\forall X\ X \to X\| & \text{if } v(a) = v(b), \\ \|\top \to \bot\| & \text{otherwise.} \end{cases}$

Proof. Both facts can be checked by expanding the definitions on either side of the equality. □

Moreover, as mentioned above, equational implication is equivalent to regular implication:

Lemma 17. *Let A be a formula and a and b two first-order terms. The formulas $(a = b) \hookrightarrow A$ and $(a = b) \to A$ are universally equivalent.*

Proof. By the previous lemma and the definition of $\|(a = b) \hookrightarrow a\|$, we see that $\lambda a.\lambda e.ea$ universally realizes the left-to-right implication and $\lambda b.b(\lambda x.x)$ the right-to-left implication. □

In particular, formulas which contain the construction $\hookrightarrow$ can also be read as arithmetic formulas.

2.7 Horn clauses

An other important class of formulas which are preserved by classical realizability is the class of Horn clauses:

Proposition 18. *Let H be a* Horn clause*, i.e. a closed formula of the form $\forall x_1 \dots x_m\ E_1 \to \dots \to E_n \to G$, with E_i of the form $a_i = b_i$ for all i, and G of the form either $c = d$ (*definite clause*) or $\bot$ (*goal clause*). In particular, H is arithmetic. If H is true in $\mathbb{N}$, then H is universally realized, and if not, then $\neg H$ is universally realized.*

Proof. Let us first assume that H is false in $\mathbb{N}$. Let $p_1, \dots, p_m$ be values for $x_1, \dots, x_m$ which invalidate it (*i.e.* the $E_i(p_1, \dots, p_m)$ are true and $G(p_1, \dots, p_m)$ is false). Then $I = \lambda y.y$ universally realizes $E_i(p_1, \dots, p_m)$ for all i, so $\lambda f.fI \dots I$ universally realizes either $H \to \bot$ (if H is a goal clause) or $H \to \top \to \bot$ (if H is a definite clause).

Now, let us assume that H is true, and let us fix a pole $\perp\!\!\!\perp$ and some $p_1, \dots, p_m \in \mathbb{N}$. To simplify notation, we will write E_i for $E_i(p_1, \dots, p_m)$, etc. Let $t_1, \dots, t_n$ be terms such that t_i realizes E_i for all i. If all the E_i are true, then G is true (and therefore, of the form $c = d$), so $u = \lambda y.t_1(\dots(t_n(y))\dots)$ realizes $G \approx \forall X\ X \to X$. If any of the E_i is false, then u realizes $\top \to \bot$, therefore, if G is of the form $c = d$, it is realized by u, and if it is $\bot$, then it is realized by $u(\lambda y.y)$. □

However, as we will see in section 7, a universal (Π^0_1) formula which is not a Horn clause (*i.e.* which is not of the form described in proposition 18) is generally not universally realized, even if it is true in $\mathbb{N}$.

2.8 The axioms of arithmetics in the realizability model

Let s denote the successor function: $s(n) = n + 1$ for all $n \in \mathbb{N}$. The first two axioms of Peano, which can be written as $\forall x\ s(x) \neq 0$ and $\forall x \forall y\ s(x) = s(y) \to x = y$, are universally realized (by proposition 18). However, the axiom of induction, which can be written as $\forall x\ \forall Z\ (\forall y\ Z(y) \to Z(s(y))) \to Z(0) \to Z(x)$, is not, as we will see later.

Notwithstanding, we can define a predicate $Nat(x)$ such that all three axioms are realized when relativized to Nat: let $Nat(x)$ denote the formula $\forall Z\ (\forall y\ Z(y) \to Z(s(y))) \to Z(0) \to Z(x)$ Intuitively, $Nat(x)$ says "x is a natural number".

Proposition 19. *The following formulas are universally realized:*

- $Nat(0)$;
- $\forall x\ \ Nat(x) \to Nat(s(x))$;
- $\forall x\ \ Nat(x) \to s(x) \neq 0$;
- $\forall x \forall y\ \ Nat(x) \to Nat(y) \to s(x) = s(y) \to x = y$;
- $\forall x\ \ Nat(x) \to \forall Z\ (\forall y\ Z(y) \to Z(s(y))) \to Z(0) \to Z(x)$.

Proof. The third and fourth are consequences of their non-relativized versions, and the other three are provable in second-order logic (and therefore realized, by the adequacy lemma). □

3 The characteristic Boolean algebra ⅁2

In this section, we define the characteristic Boolean algebra ⅁2 and prove some general properties about it.

Intuitively, ⅁2 is a unary predicate (*i.e.* a set of individuals) present in each realizability model. Since we are dealing with models of second-order logic, ⅁2 must be thought of as a "first-class" object in the realizability model. However, for simplicity, we will define it as a formula with one parameter, *i.e.* a purely syntactic object.

The predicate ⅁2(x) is designed so that ⅁2(0) and ⅁2(1) are both true (*i.e.* universally realized) and ⅁2(2), ⅁2(3), ... are all false. Moreover, and crucially, it is designed so that ⅁2(0) and ⅁2(1) are true *in the same way, i.e.* $\|⅁2(0)\| = \|⅁2(1)\|$. In that respect, it is very different from the predicate $(x = 0 \vee x = 1)$, which represents the set $\{0, 1\}$.

In fact, we can define a predicate ⅁n for all n, although only ⅁2 is a Boolean algebra:

Notation. For all integers m and n, we denote by $\max(m, n)$ the greater of the two, and by $\min(m, n)$ the lesser. For all first order terms a and b, we write $a \leq b$ for $\min(a, b) = a$.

Definition 20. Let n be a natural number and a a first-order term: we denote by ⅁$n(a)$ the formula $a + 1 \leq n$.

By lemma 16, ⅁$n(k)$ is universally realized if $k < n$, and $\neg$ ⅁$n(k)$ is universally realized otherwise.

However, whenever $n > 1$, the formula $\forall x$ ⅁$n(x) \to (x = 0) \vee \dots \vee (x = n - 1)$ is not universally realized, even though it is true in $\mathbb{N}$. This means that the predicate ⅁n does not represent the set $\{0, \dots, n - 1\}$. We will prove this for $n = 2$ in section 7.

Relativized quantifiers. Let n be an integer, x a first-order variable and A a formula. We write $\forall x^{⅁n} A$ for $\forall x$ ⅁$n(x) \hookrightarrow A$.

Now, we show how to equip ⅁2 with the structure of a Boolean algebra, inherited from the set $\{0, 1\}$:

Notation. For all natural numbers m and n, let

- $m \vee n = 1$ if $m > 0$ or $n > 0$, $m \vee n = 0$ otherwise,
- $m \wedge n = 1$ if $m > 0$ and $n > 0$, $m \wedge n = 0$ otherwise,
- $\neg m = 1$ if $m = 0$, $\neg m = 0$ otherwise,

Definition 21. The *language of Boolean algebras* is the subset of the realizability language defined by the following grammar:

Terms: $a, b ::= x \mid 0 \mid 1 \mid a \vee b \mid a \wedge b \mid \neg a$

Formulas: $A, B ::= \top \mid \bot \mid a \neq b \mid a = b \mid A \to B \mid A \vee B \mid A \wedge B \mid \forall x\ A$

Terms and formulas of this language can be interpreted in any Boolean algebra. In particular, they can be interpreted in the Boolean algebra $\{0,1\}$.

Note that we use the same notations $\vee$, $\wedge$ and $\neg$ for operations in Boolean algebras and for logical connectives: the context will always make it clear which is which.

Notation. If A is a formula of the language of Boolean algebras, we denote by $\daleth 2 \models A$ the formula A relativized to $\daleth 2$ (*i.e.*, A with all $\forall x$ turned into $\forall x^{\daleth 2}$).

We choose this notation, rather than the more conventional $A^{\daleth 2}$, in order to emphasise that we think of $\daleth 2$ as a first-class object of the realizability model which, *from the point of view of the realizability model*, happens to be a Boolean algebra (*i.e.* a first-order structure on the language of Boolean algebras satisfying all the axioms of Boolean algebras).

Note that the formula $\daleth 2 \models A$ is true in $\mathbb{N}$ if and only if A is true in $\{0,1\}$.

The following proposition establishes that $\daleth 2$ is indeed a Boolean algebra (with at least two elements):

Proposition 22. *Let A be a closed formula of the language of Boolean algebras. If A is true in every Boolean algebra with at least two elements, then $\daleth 2 \models A$ is universally realized.*

Proof. The theory of Boolean algebras with at least two elements can be axiomatised by a list $E_1, \dots, E_n$ of formulas, where each E_i is of the form $\forall x_{i,1} \dots x_{i,m_i}\ G_i$ and G_i is of the form $a_i = b_i$ or $a_i \neq b_i$.

For all i, denote by H_i the formula $\forall x_{i,1} \dots \forall x_{i,m_i}\ \daleth 2(x_{i,1}) \to \dots \to \daleth 2(x_{i,m_i}) \to G_i$: it is a Horn clause, and it is universally equivalent to the formula $\daleth 2 \models E_i$, which is true in $\mathbb{N}$ (because E_i is true in $\{0,1\}$). Therefore, by proposition 18, $\daleth 2 \models E_i$ is universally realized.

The proposition follows by the adequacy lemma and the completeness theorem of first-order logic. □

The cardinality of $\daleth 2$

An important property of a realizability model is the cardinality of its characteristic Boolean algebra. This subsection develops the basic tools that we will need in order to state and prove facts about this cardinality.

If $\mathbb{B}$ is a Boolean algebra and n a positive integer, then formula $\forall x_1 \dots x_n\ \bigvee_{i \neq j}(x_i = x_j)$ is true in $\mathbb{B}$ if and only if $\mathbb{B}$ has fewer than n elements. This justifies the following notation:

Notation. Let n be a positive integer.

- We denote by $|\daleth 2| < n$ (or $|\daleth 2| \leq n+1$) the formula
$$\daleth 2 \models \forall x_1 \dots x_n\ \bigvee_{i \neq j}(x_i = x_j),$$
- We denote by $|\daleth 2| \geq n$ the formula $(|\daleth 2| < n) \to \bot$,
- We denote by $|\daleth 2| = n$ the formula $(|\daleth 2| \geq n) \wedge (|\daleth 2| < n+1)$.

Remark. *The cardinality of a finite Boolean algebra is always a power of two. Therefore, for all n, the formulas $|\daleth 2| \leq 2^n$ and $|\daleth 2| < 2^{n+1}$ are universally equivalent, and in particular, the formula $|\daleth 2| = 2^n$ is universally equivalent to $|\daleth 2| \geq 2^n \wedge |\daleth 2| < 2^{n+1}$.*

In general, a Boolean algebra has at least 2^n element if and only if it contains a sequence of n pairwise-disjoint non-zero elements. Therefore, by proposition 22, the sentences "$\daleth 2$ has at least 2^n elements" and "$\daleth 2$ contains a sequence of n pairwise-disjoint elements" are equivalent (see [3], chapter 15). We will favour the latter, because because its formal statement (as a formula of the realizability language) has a better-behaved falsity value:

Notation. For all positive integers n, let us denote by A_n the following formula of the language of Boolean algebras:

$$\forall x_1 \dots x_n\ x_1 \neq 0 \to \dots \to x_n \neq 0 \to \left(\bigvee_{i \neq j} x_i \wedge x_j\right) \neq 0.$$

If $\mathbb{B}$ is a Boolean algebra and n a positive integer, then the formula A_n is true in $\mathbb{B}$ if and only if it is not possible to find n pairwise-disjoint non-zero elements in $\mathbb{B}$.

The following lemma describes the falsity value of $\daleth 2 \models A_n$ and shows that it is indeed quite simple:

Lemma 23. *Let n be a positive integer, $\perp\!\!\!\perp$ a pole, $t_1, \dots, t_n \in \Lambda$ and $\pi \in \Pi$. Then $t_1 \cdot \dots \cdot t_n \cdot \pi \in \|\daleth 2 \models A_n\|_{\perp\!\!\!\perp}$ if and only if at most one of the t_i does not realize $\bot$.*

Proof. For all $x_1, \dots, x_n$, one can check that $\left(\bigvee_{i \neq j} x_i \wedge x_j\right) = 0$ if and only if at most one of the x_i is not equal to 0. Therefore, $t_1 \cdot \dots \cdot t_n \cdot \pi \in \|\daleth 2 \models A_n\|_{\perp\!\!\!\perp}$ if and only there exist $x_1, \dots, x_n$ such that t_i realize $x_i \neq 0$ for all i and at most one of the x_i is not equal to 0. □

The following notation provides a third way of expressing the cardinality of $\daleth 2$. In a way, it is the most important one, since it is the one which provides the connection with non-determinism (as we will see in section 5):

Notation. Let n be a positive integer. We denote by B_n the formula

$$\forall X \begin{cases} \top \to X \to \dots \to X \to X & (1) \\ \cap\ X \to \top \to \dots \to X \to X & (2) \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots & \vdots \\ \cap\ X \to X \to \dots \to \top \to X & (n) \end{cases}.$$

Its falsity value is quite similar to that of $\daleth 2 \models A_n$ (which is why they are equivalent):

Lemma 24. *Let n be a positive integer, $\perp\!\!\!\perp$ a pole, $t_1, \dots, t_n \in \Lambda$ and $\pi \in \Pi$. Then $t_1 \cdot \dots \cdot t_n \cdot \pi \in \|B_n\|_{\perp\!\!\!\perp}$ if and only if there is at most one i such that $t_i \star \pi \notin \perp\!\!\!\perp$.*

Proposition 25. *Let n be a positive integer. The formulas $|\daleth 2| < 2^n$, $\daleth 2 \models A_n$ and B_n are universally equivalent.*

Proof of proposition 25. The formulas $|\daleth 2| < 2^n$ and $\daleth 2 \models A_n$ are universally equivalent by proposition 22.

By lemmas 23 and 24, $B_n \to (\daleth 2 \models A_n)$ is universally realized by $\lambda t.t$ (because $B_n \lesssim (\daleth 2 \models A_n)$) and $(\daleth 2 \models A_n) \to B_n$ is universally realized by $\lambda t.\ \lambda u_1 \dots . \lambda u_n.\ cc\ (\lambda k.\ t\ (ku_1) \dots (ku_n))$ (because for all $\perp\!\!\!\perp$, π and i, $u_i \star \pi \in \perp\!\!\!\perp$ implies $k_\pi u_i \Vdash_{\perp\!\!\!\perp} \bot$). □

As a side note, we would like to bring the reader's attention on the following result (which is due to Krivine, but does not seem to appear in any published work):

Proposition 26. *The formulas $|\daleth 2| = 2$ and $\forall x\ Nat(x)$ are universally equivalent.*

Proof. By proposition 25, it suffices to realize $B_2 \leftrightarrow \forall x, Nat(x)$.

Let us denote by δ the λ_c-term $\lambda d.x(dd)$ and by Y the λ_c-term $\delta\delta$. For all $t \in \Lambda$ and all $\pi \in \Pi$, $Y[x := t] \star \pi \succ_K t \star Y[x := t] \cdot \pi$.

Let $\overline{0}$ denote the term $\lambda f.\lambda x.\ x$ (*i.e* the Church numeral 0) and $\overline{s}$ the term $\lambda n.\lambda f.\lambda x.\ n\ f\ (f\ x)$ (*i.e.* the Church successor function). Using the adequacy lemma, one can check that $\overline{0}$ universally realizes $Nat(0)$ and that $\overline{s}$ universally realizes $\forall n.\ Nat(n) \to Nat(n+1)$.

To simplify notation, let us denote $Y[x := \lambda y.\ \psi\ \overline{0}\ (\overline{s}y)]$ by Y_ψ. We prove that the left-to-right implication is universally realized by $l = \lambda\psi.\ Y_\psi$. Let us fix a pole $\bot\!\!\bot$, and let $\psi \Vdash B_2$. For all $\pi \in \Pi$, $Y_\psi \star \pi \succ_K \psi \star \overline{0} \cdot \overline{s} Y_\psi \cdot \pi$. Therefore, $Y_\psi \Vdash Nat(0)$, and for all n such that $Y_\psi \Vdash Nat(n)$, $Y_\psi \Vdash Nat(n+1)$ (because $\overline{s}$ realizes $Nat(n) \to Nat(n+1)$). By induction, Y_ψ realizes $Nat(n)$ for all n.

Let us prove that the right-to-left implication is universally realized by $r = \lambda t.\ \lambda u.\ \lambda v.\ t\ (\lambda z.u)\ v$. First, note that $Nat(0) \approx \forall Z,\ \top \to Z \to Z$ and $Nat(1) \approx \forall Z_0, Z_1,\ (Z_0 \to Z_1) \to Z_0 \to Z_1$. Let us fix a pole $\bot\!\!\bot$, and let $t \Vdash \forall x,\ Nat(x)$, $X \in \mathcal{P}(\Pi)$ and $u \cdot v \cdot \pi \in \|B_2\|_{\bot\!\!\bot}$, which means that $\pi \in X$, and u or v realizes X. If v realizes X, then, since t realizes $Nat(0)$, $t\ (\lambda z.u)\ v$ realizes X. If u realizes X, then $\lambda z.u$ realizes $\top \to X$, t realizes $Nat(1)$ and v realizes $\top$, so again $t\ (\lambda z.u)\ v$ realizes X. In both cases, $r \star t \cdot u \cdot v \cdot \pi \in \bot\!\!\bot$. □

4 Multi-evaluation relations

This section defines a correspondence between realizability structures and a certain notion of *multi-evaluation relations* which extend the evaluation relation of λ_c-calculus: this notion is powerful enough to express arbitrary mixes of may- and must-nondeterminism.

Then, we will explore some of the connections between the properties of the theory generated by a structure and the properties of the corresponding evaluation relation.

While the evaluation relation of λ_c-calculus is a relation between processes, *multi-evaluation relations* are relations between *sets of processes* which satisfy certain axioms. Using "$\succ$" generic symbol for evaluation relations, the following intuitions should be kept in mind:

The statement "$\{p\} \succ \{q_1\}$ and $\{p\} \succ \{q_2\}$" specifies that, under "$\succ$", p reduces nondeterministically to q_1 or q_2, according to an oracle which "favours" being in the pole: this is *may*-nondeterminism. One can also imagine that the program tries both branches in parallel, to find out if one eventually meets the condition for being in the pole. Meanwhile, "$\{p\} \succ \{q_1, q_2\}$" specifies that p reduces nondeterministically to q_1 or q_2, according to an oracle which "favours" being out of the pole: this is *must*-nondeterminism.

Additionally, we can mix may- and must-nondeterminism by writing, say, "$\{p\} \succ \{q_1, q_2\}$ and $\{p\} \succ \{q_2, q_3\}$ and $\{p\} \succ \{q_3, q_1\}$". This kind of two-out-of-three nondeteministic choice will be studied in detail in the next section, under the name of 3-voting instruction.

Finally, statements like "$\{p_1, p_2\} \succ \{q_1, q_2\}$" allow interaction between different branches of execution: if one branch evaluates to p_1 and another to p_2, then they merge and evaluate *must*-nondeterministically to q_1 or q_2.

Definition 27. A *multi-evaluation relation*, or simply *evaluation relation* is a binary relation on $\mathcal{P}(\Lambda \star \Pi)$ such that

- For all $p, q \in \Lambda \star \Pi$ such that $p \succ_K^1 q$, $\{p\} \succ \{q\}$,
- For all $p \in \Lambda \star \Pi$, $\{p\} \succ \{p\}$ *(identity)*,
- For all $P, Q, P', Q' \in \mathcal{P}(\Lambda \star \Pi)$, for all $r \in \Lambda \star \Pi$, if $P \succ Q \cup \{r\}$ and $P' \cup \{r\} \succ Q'$, then $P \cup P' \succ Q \cup Q'$ *(cut)*,
- For all $P, Q, P', Q' \in \mathcal{P}(\Lambda \star \Pi)$ such that $P \succ Q$, if $P \subseteq P'$ and $Q \subseteq Q'$, then $P' \succ Q'$ *(weakening)*.

The intuition behind the cut rule is the following: if on the one hand $p_1, \ldots, p_m$, together, evaluate to r or one of $q_1, \ldots, q_n$, and the other hand $p'_{1'}, \ldots, p'_{m'}$, together with r, evaluate to one of $q'_{1'}, \ldots, q'_{n'}$, then, together, $p_1, \ldots, p_m$ and $p'_{1'}, \ldots, p'_{m'}$ must evaluate to one of $q_1, \ldots, q_n$ (directly) or one of $q'_{1'}, \ldots, q'_{n'}$ (*via* r).

Definition 28. Let $\succ$ be a binary relation on $\mathcal{P}(\Lambda \star \Pi)$. The *realizability structure generated by* $\succ$, denoted by $\mathcal{S}_\succ$, is the of set all poles $\bot\!\!\bot$ such that, for all P, Q such that $P \succ Q$, if $Q \subseteq \bot\!\!\bot$, then $P \cap \bot\!\!\bot \neq \emptyset$.

Definition 29. Let $\mathcal{S}$ be a realizability structure. The *evaluation relation associated with* $\mathcal{S}$, written "$\succ_\mathcal{S}$", is the binary relation on $\mathcal{P}(\Lambda \star \Pi)$ defined by the following: for all $P, Q \in \mathcal{P}(\Lambda \star \Pi)$, $P \succ_\mathcal{S} Q$ if and only if for all $\bot\!\!\bot \in \mathcal{S}$, if $Q \subseteq \bot\!\!\bot$, then $P \cap \bot\!\!\bot \neq \emptyset$.

Proposition 30. *Let $\mathcal{S}$ be a realizability structure: $\succ_\mathcal{S}$ is an evaluation relation, and $\mathcal{S}_{\succ_\mathcal{S}} = \mathcal{S}$.*

This means that any realizability structure is fully described by its associated evaluation relation.

Proof. Let $\bot\!\!\bot \in \mathcal{S}$. By definition of $\succ_\mathcal{S}$, $\bot\!\!\bot \in \mathcal{S}_{\succ_\mathcal{S}}$. Let $\bot\!\!\bot \in \mathcal{S}_0 \setminus \mathcal{S}$. Then for all $\bot\!\!\bot' \in \mathcal{S}$, $\bot\!\!\bot' \neq \bot\!\!\bot$, which means that if $\bot\!\!\bot \subseteq \bot\!\!\bot'$, there exists $p \in \bot\!\!\bot'$ such that $p \notin \bot\!\!\bot$. In other words, $(\Lambda \star \Pi \setminus \bot\!\!\bot) \succ_\mathcal{S} \bot\!\!\bot$, so $\bot\!\!\bot \notin \mathcal{S}$. □

Lemma 31. *Let $\succ$ be a binary relation on $\mathcal{P}(\Lambda \star \Pi)$. There exists a smallest evaluation relation containing $\succ$. Moreover, if we denote this relation by $\succ^*$, then $\mathcal{S}_{\succ^*} = \mathcal{S}_\succ$.*

Proof. The set of evaluations relations containing $\succ$ is stable by arbitrary intersections, so it has a smallest element.

Since $\succ \subseteq \succ^*$, $\mathcal{S}_\succ \supseteq \mathcal{S}_{\succ^*}$. Moreover, $\succ_{\mathcal{S}_\succ}$ is an evaluation relation which contains $\succ$, so it contains $\succ^*$. Therefore, $\mathcal{S}_{\succ^*} \subseteq \mathcal{S}_{\succ_{\mathcal{S}_\succ}} = \mathcal{S}_\succ$, by proposition 30. □

If $\succ$ is an evaluation realation, it is true in general that $\succ \subseteq \succ_{\mathcal{S}_\succ}$, but not that $\succ = \succ_{\mathcal{S}_\succ}$.[1]

5 Voting instructions and fork

Using the language of evaluation relations, one can state the following specification result:

Definition 32. Let n be a positive integer and $\succ$ an evaluation relation. An *n-voting instruction modulo $\succ$* is a term ϕ such that for all terms $t_1, \ldots, t_n$ and all stacks π, for all $j \in \{1, \ldots, n\}$, $\{\phi \star t_1 \cdot \ldots \cdot t_n \cdot \pi\} \succ \{t_i \star \pi; i \neq j\}$.

Theorem 33. *Let $\mathcal{S}$ be a realizability structure and n a positive integer. A term ϕ realizes B_n with respect to $\mathcal{S}$ if and only if ϕ is an n-voting instruction modulo $\succ_\mathcal{S}$.*

Proof. First, we notice that, by lemma 23, for all poles $\bot\!\!\bot$, $t_1 \cdot \ldots \cdot t_n \cdot \pi \in \|B_n\|_{\bot\!\!\bot}$ if and only if there exists $j \in \{1, \ldots, n\}$ such that for all $i \neq j$, $t_i \star \pi \in \bot\!\!\bot$. Therefore, both sides of the equivalence amount to saying that for all $\bot\!\!\bot \in \mathcal{S}$, for all $t_1 \cdot \ldots \cdot t_n \cdot \pi \in \|B_n\|_{\bot\!\!\bot}$, $\phi \star t_1 \cdot \ldots \cdot t_n \cdot \pi \in \bot\!\!\bot$, and so they are equivalent. □

[1] This becomes true if $\succ$ is *compact* (*i.e.* for all $P \succ Q$, there exist two finite sets $P_0 \subseteq P$ and $Q_0 \subseteq Q$ such that $P_0 \succ Q_0$) or if $\succ$ satisfies the following infinitary cut-rule (which is implied by compactness): if there exists R such that, for all $S \subseteq R$, $P \cup S \succ (R \setminus S) \cup Q$, then $P \succ Q$.

Corollary 34. *Let $\mathcal{S}$ be a realizability structure and n a positive integer. We have* $\mathrm{Th}(\mathcal{S}) \models |\beth 2| \leq 2^n$ *if and only if there is a* proof-like *n-voting instruction modulo $\succ_{\mathcal{S}}$.*

We use the name n-voting instruction because whenever all but one of the t_i "agree on" a certain behaviour, $\phi\ t_1 \dots t_n$ "chooses" it. For example, if one of the t_i is the church numeral $\overline{0}$ and all the others are $\overline{1}$, then $\phi\ t_1 \dots t_n$ behaves like $\overline{1}$. (The notion of 3-voting instruction was introduced by Trakhtenbrot [18].)

Interestingly, the important point turns out to be, not the number of "votes for", but rather their *proportion*. Indeed, one can define a notion of (n,k)-voting instruction, which takes n arguments and requires at least $n-k$ "votes" (so that an n-voting instruction is the same thing as an $(n,1)$-voting instruction). Then, using the same method, one can prove that there is a proof-like (n,k)-voting instruction if and only if $\mathrm{Th}(\mathcal{S}) \models |\beth 2| < 2^{\lceil \frac{n}{k} \rceil}$.

Now, let us consider the following definitions:

Definition 35. Let $\succ$ be an evaluation relation.

- A *fork instruction* (or *may*-nondeterministic choice operator) modulo $\succ$ is a term ψ such that for all terms u, v and all stacks π, $\{\psi \star u \cdot v \cdot \pi\} \succ \{u \star \pi\}$ *and* $\{\psi \star u \cdot v \cdot \pi\} \succ \{v \star \pi\}$ (*i.e.* a 2-voting instruction).
- A *must*-nondeterministic choice operator modulo $\succ$ is a term χ such that for all terms u, v and all stacks π, $\{\chi \star u \cdot v \cdot \pi\} \succ \{u \star \pi, v \star \pi\}$.

Proposition 36. *Let $\mathcal{S}$ be a realizability structure and t a term. Then t is a* may*-nondeterministic choice operator modulo $\succ_{\mathcal{S}}$ if and only if it realizes $\forall X \forall Y\ X \to Y \to X \cap Y$ with respect to $\mathcal{S}$, and it is a* must*-nondeterministic choice operator $\succ_{\mathcal{S}}$ if and only if it realizes $\forall X \forall Y\ X \to Y \to X \cup Y$ with respect to $\mathcal{S}$.*

Proof. If t is a fork instruction, let $\perp\!\!\!\perp \in \mathcal{S}$, $X, Y \subseteq \Pi$, $u \Vdash_{\perp\!\!\!\perp} X$, $v \Vdash_{\perp\!\!\!\perp} Y$ and $\pi \in X \cup Y$. If $\pi \in X$, then $\{t \star u \cdot v \cdot \pi\} \succ_{\mathcal{S}} \{u \star \pi\} \subseteq \perp\!\!\!\perp$, and similarly if $\pi \in Y$.

Assume t is not a fork instruction. Without loss of generality, one can assume that there exist u, v and π such that $\{t \star u \cdot v \cdot \pi\} \not\succ_{\mathcal{S}} \{u \star \pi\}$. Let $\perp\!\!\!\perp \in \mathcal{S}$ such that $u \star \pi \in \perp\!\!\!\perp$ and $t \star u \cdot v \cdot \pi \notin \perp\!\!\!\perp$, and let $X = \{\pi\}$ and $Y = \emptyset$: then $u \cdot v \cdot \pi \in \|X \to Y \to X \cap Y\|_{\perp\!\!\!\perp}$, so t does not realize $X \to Y \to X \cap Y$ with respect to $\perp\!\!\!\perp$.

The proof of the second point is similar. □

Proposition 37. *Let $\mathcal{S}$ be a realizability structure. We have* $\mathrm{Th}(\mathcal{S}) \models |\beth 2| = 2$ *if and only if there exists a* proof-like *fork instruction modulo $\succ_{\mathcal{S}}$.*

Proof. Consequence of corollary 34 and proposition 22. □

6 Parallel Or

Definition 38. Let $\mathrm{Bool}(y)$ denote the formula $\forall X\ X(0) \to X(1) \to X(y)$.

Lemma 39. *We have* $\mathrm{Bool}(0) \approx \forall X\ X \to \top \to X$, $\mathrm{Bool}(1) \approx \forall X,\ \top \to X \to X$, *and for all $n > 1$,* $\mathrm{Bool}(n) \approx \top \to \top \to \bot$.

Remark. *The formula* $\mathrm{Bool}(0)$ *is universally realized by* false $= \lambda x.\lambda y.x$, *and* $\mathrm{Bool}(1)$ *by* true $= \lambda x.\lambda y.y$.

Proposition 40. *Let $\mathcal{S}$ be a realizability structure and t a term: t realizes* $\mathrm{Bool}(0)$ *in $\mathcal{S}$ if and only if for all terms u, v and all stacks π, $\{t \star u \cdot v \cdot \pi\} \succ_{\mathcal{S}} \{u \star \pi\}$, and t realizes* $\mathrm{Bool}(1)$ *in $\mathcal{S}$ if and only if for all terms u, v and all stacks π, $\{t \star u \cdot v \cdot \pi\} \succ_{\mathcal{S}} \{v \star \pi\}$.*

Proof. Similar to theorem 33 □

In other words, realizers of $\mathrm{Bool}(0)$ are terms which behave like false, and realizers of $\mathrm{Bool}(1)$ are terms which behave like true.

One can say that a term "computes the *or* function" (modulo some realizability structure $\mathcal{S}$) if it realizes the formula $\forall xy\ \mathrm{Bool}(x) \to \mathrm{Bool}(y) \to \mathrm{Bool}(x \vee y)$, which we denote by Or. In general, there are two different ways of doing that:

- left-first: $\mathrm{or}_l = \lambda x.\lambda y.\ x\ y$ true,
- right-first: $\mathrm{or}_r = \lambda x.\lambda y.\ y\ x$ true.

The left-first version "returns true", *i.e.* behaves like true, as soon as its first argument does, even if its second argument diverges. The right-first version does the symmetric.

Let Or_l denote the formula $(\forall x\ \mathrm{Bool}(0) \to \mathrm{Bool}(x) \to \mathrm{Bool}(x)) \cap (\mathrm{Bool}(1) \to \top \to \mathrm{Bool}(1))$, and let Or_r denote the formula $(\forall x\ \mathrm{Bool}(x) \to \mathrm{Bool}(0) \to \mathrm{Bool}(x)) \cap (\top \to \mathrm{Bool}(1) \to \mathrm{Bool}(1))$. Note that $\mathrm{Or}_l \lesssim \mathrm{Or}$ and $\mathrm{Or}_r \lesssim \mathrm{Or}$.

Lemma 41. *The term* or_l *universally realizes* Or_l, *and the term* or_r *universally realizes* Or_r.

How different are or_l and or_r? Can we come up with a (proof-like) term which captures the behaviour of both? That is to say, a *parallel or*, which returns true as soon as either of its arguments is true, even if the other diverges, and false if both arguments are false. This turns out to depend on the realizability structure and to be related to the properties of $\beth 2$.

First, let us denote by $\mathrm{Or}_{\|}$ the formula $(\mathrm{Bool}(1) \to \top \to \mathrm{Bool}(1)) \cap (\top \to \mathrm{Bool}(1) \to \mathrm{Bool}(1)) \cap (\mathrm{Bool}(0) \to \mathrm{Bool}(0) \to \mathrm{Bool}(0))$.

Definition 42. Let $\mathcal{S}$ be a realizability structure and t a term. The term t *computes parallel or modulo* $\mathcal{S}$ if it realizes $\mathrm{Or}_{\|}$ with respect to $\mathcal{S}$.

Theorem 43. *The formulas* $\mathrm{Or}_{\|}$ *and* $|\beth 2| \leq 4$ *are universally equivalent.*

Proof. In order to realize the left-to-right implication, by proposition 25, it suffices to realize $\mathrm{Or}_{\|} \to (\beth 2 \models A_3)$. Let us prove that this formula is universally realized by $l = \lambda t.\lambda u_1.\lambda u_2.\lambda u_3.\ (t\ u_1\ u_2)\ u_1\ u_3$. Let us fix a pole $\perp\!\!\!\perp$, and let $t \Vdash_{\perp\!\!\!\perp} \mathrm{Or}_{\|}$ and $u_1 \cdot u_2 \cdot u_3 \cdot \pi \in \| \beth 2 \models A_3 \|_{\perp\!\!\!\perp}$. By lemma 23, at least two of the u_i realize $\perp$. If u_1 and u_2 realize $\perp$, in particular, they realize $\mathrm{Bool}(0)$, which means that $t\ u_1\ u_2$ realizes $\mathrm{Bool}(0)$. Since u_1 realizes $\perp$, $(t\ u_1\ u_2)\ u_1\ u_3$ realizes $\perp$, so $l \star t \cdot u_1 \cdot u_2 \cdot u_3 \cdot \pi \in \perp\!\!\!\perp$. If u_1 and u_3 realize $\perp$, in particular, u_1 realizes $\mathrm{Bool}(1)$, so $t\ u_1\ u_2$ realizes $\mathrm{Bool}(1)$, $(t\ u_1\ u_2)\ u_1\ u_3$ realizes $\perp$, and $l \star t \cdot u_1 \cdot u_2 \cdot u_3 \cdot \pi \in \perp\!\!\!\perp$. The last case ($u_2$ and u_3 realize $\perp$) is similar to the second.

In order to realize the right-to-left implication, by proposition 25, it suffices to realize $B_n \to \mathrm{Or}_{\|}$. Let us prove that this formula is universally realized by $r = \lambda t.\lambda u.\lambda v.\ t\ (\mathrm{or}_l\ u\ v)\ (\mathrm{or}_r\ u\ v)$ true. Indeed, let us fix a pole $\perp\!\!\!\perp$, and let $t \Vdash_{\perp\!\!\!\perp} B_3$, $u, v \in \Lambda$ and $\pi \in \Pi$.

If u realizes $\mathrm{Bool}(1)$ and $\pi \in \|\mathrm{Bool}(1)\|_{\perp\!\!\!\perp}$, then $\mathrm{or}_l\ u\ v$ and true both realize $\mathrm{Bool}(1)$, so $t\ (\mathrm{or}_l\ u\ v)\ (\mathrm{or}_r\ u\ v)$ true realizes $\mathrm{Bool}(1)$ (because t realizes $\forall X\ X \to \top \to X \to X$), so $r \star t \cdot u \cdot v \cdot \pi \in \perp\!\!\!\perp$.

If v realizes $\mathrm{Bool}(1)$ and $\pi \in \|\mathrm{Bool}(1)\|_{\perp\!\!\!\perp}$, then $r \star t \cdot u \cdot v \cdot \pi \in \perp\!\!\!\perp$ for similar reasons.

If u and v realize $\mathrm{Bool}(0)$ and $\pi \in \|\mathrm{Bool}(0)\|$, then $\mathrm{or}_l\ u\ v$ and $\mathrm{or}_r\ u\ v$ both realize $\mathrm{Bool}(0)$, so $t\ (\mathrm{or}_l\ u\ v)\ (\mathrm{or}_r\ u\ v)$ true realizes $\mathrm{Bool}(0)$ (because t realizes $\forall X\ X \to X \to \top \to X$), so $r \star t \cdot u \cdot v \cdot \pi \in \perp\!\!\!\perp$. □

Corollary 44. *Let $\mathcal{S}$ be a realizability structure. We have* $\mathrm{Th}(\mathcal{S}) \models |\exists 2| \leq 4$ *if and only if there is a proof-like term which computes* parallel or *modulo* $\mathcal{S}$.

As a corollary, there is a proof-like term which computes *parallel or* if and ony if there is a proof-like 3-voting instruction: a similar result was stated by Sazonov in [17], though in a very different setting.

Formulas of the form $|\exists 2| \leq 2^n$ create a hierarchy on realizability theories (which is included in the hierarchy of all formulas of the form $\exists 2 \models A$). Through the realizability interpretation, this gives a hierarchy on models of computation. This hierarchy compares models of computation on their ability to "try" different execution paths using nondeterminism (and, consequently, to perform computations on "partial data", the way *parallel or* can compute even when one of its arguments is undefined or diverges): the bigger $\exists 2$, the more deterministic the computation model.

This hierarchy is reminiscent of the *degrees of parallelism* of [17]. Indeed, the first three levels correspond to *fork*, *parallel or* and 4-voting functions respectively.

Interestingly, the third level also corresponds to Gérard Berry's "Gustave's function". More precisely, the behaviour of Gustave's function can be encoded by the formula $(0 \to 1 \to \top \to 1) \cap (\top \to 0 \to 1 \to 1) \cap (1 \to \top \to 0 \to 1) \cap (0 \to 0 \to 0 \to 0) \cap (1 \to 1 \to 1 \to 0)$ (where 0 stands for Bool(0) and 1 for Bool(1)), and it can be proved that this formula is universally equivalent to $|\exists 2| \leq 8$ (similar to theorem 43).

7 Models with $|\exists 2| = 2^n$ for any $n > 0$

In order to prove that the hierarchy described above (of formulas of the form $|\exists 2| \leq 2^n$) does not collapse, we have to prove the following:

Theorem 45. *Let n be a positive integer. There exists a consistent realizability structure $\mathcal{S}$ such that* $\mathrm{Th}(\mathcal{S}) \models |\exists 2| = 2^n$.

The cardinality of a finite Boolean algebra is always a power of two. Moreover, for all n, the theory of Boolean algebras with 2^n elements is complete, so this theorem means $\exists 2$ can be made elementarily equivalent to any finite Boolean algebra with at least two elements.

Let us fix a positive integer n.

Because of the results from section 3, it suffices to find a consistent realizability structure where both $\exists 2 \models A_{n+1}$ and $\exists 2 \models A_n \to \bot$ are realized.

Let $\beta_\top$ and $\beta_\bot$ be two restricted instructions and ϕ and χ two nonrestricted instructions. Let $>_1$ be the smallest binary relation on $\mathcal{P}(\Lambda \star \Pi)$ such that:

- For all $p, q \in \Lambda \star \Pi$, if $p >^1_{\mathrm{K}} q$, then $\{p\} >^1 \{q\}$,
- For all $t_0, \ldots, t_n \in \Lambda$, $\pi \in \Pi$ and $j \in \{0, \ldots, n\}$,
$$\{\phi \star t_0 \cdot \ldots \cdot t_n \cdot \pi\} >^1 \{t_i \star \pi; i \neq j\},$$
- For all $\pi \in \Pi$, $u \in \Lambda$ and $k \in \{1, \ldots, n\}$,
$$\{\chi \star u \cdot \pi\} >^1 \{u \star \beta_{1=k} \cdot \ldots \cdot \beta_{n=k} \cdot \pi; 1 \leq k \leq n\},$$
where $\beta_{i=k}$ means $\beta_\top$ if $i = k$ and $\beta_\bot$ if $i \neq k$,
- For all $\pi \in \Pi$, $\{\beta_\bot \star \pi\} >^1 \emptyset$.

Let $\succ$ be the smallest evaluation relation which contains $>_1$, and let $\mathcal{S} = \mathcal{S}_\succ$.

Note that ϕ is an n-voting instruction with respect to $\succ$.

Lemma 46. *The term $\beta_\bot$ realizes $\bot$ with respect to $\mathcal{S}$.*

Proposition 47. *The term ϕ realizes $\exists 2 \models A_{n+1}$ with respect to $\mathcal{S}$.*

Proof. From lemma 24 and theorem 33. □

Proposition 48. *The term χ realizes $\exists 2 \models A_n \to \bot$ with respect to $\mathcal{S}$.*

Proof. Let us fix a pole $\perp\!\!\!\perp \in \mathcal{S}$. Let $u \Vdash_{\perp\!\!\!\perp} (\exists 2 \models A_n)$, $\pi \in \Pi$ and $k \in \{1, \ldots, n\}$. There is at most one i such that $\beta_{i=k}$ does not realize $\bot$, so $\beta_{1=k} \cdot \ldots \cdot \beta_{n=k} \cdot \pi \in \|\exists 2 \models A_n\|_{\perp\!\!\!\perp}$ (by lemma 24). Therefore, $\{u \star \beta_{1=k} \cdot \ldots \cdot \beta_{n=k} \cdot \pi; 1 \leq k \leq n\} \subseteq \perp\!\!\!\perp$, so $\chi \star u \cdot \pi \in \perp\!\!\!\perp$. □

Note that in particular, the formula $\forall x^{\exists 2} \forall y^{\exists 2}\ x = 0 \vee x = 1$, which is universally equivalent to $|\exists 2| = 2$, is a disjunction of two positive literals which is true in $\mathbb{N}$, and whose *negation* is realized in $\mathcal{S}$ if $n > 1$ (compare this with proposition 18).

Lemma 49. *For all P, Q, $P \succ Q$ if and only if there exists $p \in P$ such that $\{p\} \succ Q$.*

Proof. Let $\widetilde{\succ}$ be the binary relation defined by $P \mathrel{\widetilde{\succ}} Q$ if and only if there exists $p \in P$ such that $\{p\} \succ Q$. It is contained in $\succ$, it contains $\succ^1$, and it is an evaluation relation, therefore it is equal to $\succ$. □

Lemma 50. *The evaluation relation $\succ$ is* compact*: for all P, Q such that $P \succ Q$, there exist two finite sets $P_0 \subseteq P$ and $Q_0 \subseteq Q$ such that $P_0 \succ Q_0$.*

Proof. Similar to the previous lemma. □

It remains to prove that the structure $\mathcal{S}$ is consistent. It suffices to find one pole $\perp\!\!\!\perp \in \mathcal{S}$ such that no proof-like term realizes $\bot$ with respect to $\perp\!\!\!\perp$.

Let $\perp\!\!\!\perp = \{p \in \Lambda \star \Pi, \{p\} \succ \emptyset\}$.

Lemma 51. *We have $\perp\!\!\!\perp \in \mathcal{S}$, and for all $\perp\!\!\!\perp' \in \mathcal{S}$, $\perp\!\!\!\perp \subseteq \perp\!\!\!\perp'$: in other words, $\perp\!\!\!\perp$ is the smallest pole in $\mathcal{S}$.*

Proof. Let us prove that $\perp\!\!\!\perp \in \mathcal{S}$: let $Q \subseteq \perp\!\!\!\perp$ and $P \succ Q$. By lemma 50, one can assume without loss of generality that P and Q are finite. By lemma 49, there exists $p \in P$ such that $\{p\} \succ Q$. Since $\{q\} \succ \emptyset$ for all $q \in Q$, one can prove by repeatedly applying the cut rule (once per element of Q) that $\{p\} \succ \emptyset$: in other words, $p \in \perp\!\!\!\perp$.

Now, let $\perp\!\!\!\perp' \in \mathcal{S}$ and let $p \in \perp\!\!\!\perp$. Since $\{p\} \succ \emptyset$ and $\emptyset \subseteq \perp\!\!\!\perp$, necessarily, $\{p\} \cap \perp\!\!\!\perp' \neq \emptyset$, so $p \in \perp\!\!\!\perp'$. □

We say that a process is *sound* if it can be written without the instruction $\beta_\bot$.

Let us fix a sequence $(\gamma_i)_{i \in \mathbb{N}}$ of pairwise distinct restricted instructions all different from $\beta_\top$ and $\beta_\bot$.

Notation. Let p be a process and K a subset of $\mathbb{N}$. We denote by $p[K]$ the process p where, for all $i \in \mathbb{N}$, each occurence of γ_i has been replaced by $\beta_\top$ if $i \in K$, and by $\beta_\bot$ if $i \notin K$.

Definition 52. Let p be a process. The *content* of p, denoted by $C(p)$, is the set of all $K \subseteq \mathbb{N}$ such that $p[K] \in \perp\!\!\!\perp$.

Remark. *Let p be a process and $K \subseteq L \subseteq \mathbb{N}$. One can prove that if $L \in C(p)$, then $K \in C(p)$.*

Proposition 53. *Let p be a sound process. For all $K_1, \ldots, K_n \in C(p)$, $K_1 \cup \ldots \cup K_n \neq \mathbb{N}$.*

In other words, whenever p is sound, it is impossible to span all of $\mathbb{N}$ with only n elements of $C(p)$.

Proof of proposition 53. By induction, we let $\perp\!\!\!\perp_0 = \emptyset$, and for all $r \in \mathbb{N}$, $\perp\!\!\!\perp_{r+1} = \{p \in \Lambda \star \Pi;$ there exists $Q \subseteq \perp\!\!\!\perp_r$ such that $\{p\} \succ^1 Q\}$.

For all $r \in \mathbb{N}$, let $C_r(p)$ be the set of all $K \subseteq \mathbb{N}$ such that $p[K] \in \perp\!\!\!\perp_r$.

The sequences $(\perp\!\!\!\perp_r)_r$ and $(C_r(p))_r$ are increasing. Moreover, $\perp\!\!\!\perp = \bigcup_{r\in\mathbb{N}} \perp\!\!\!\perp_r$ (because $\bigcup_{r\in\mathbb{N}} \perp\!\!\!\perp_r$ can also be proved to be the smallest pole in $\mathcal{S}$), and for all p, $C(p) = \bigcup_{r\in\mathbb{N}} C_r(p)$, so we can prove the result by induction on r.

Let $r \in \mathbb{N}$, and let us assume that for all sound p, for all $K_1, \ldots, K_n \in C_r(p)$, $K_1 \cup \ldots \cup K_n \neq \mathbb{N}$.

Let p be a sound process and $K_1, \ldots, K_n \in C_{r+1}(p)$. By contradiction, let us assume that $K_1 \cup \ldots \cup K_n = \mathbb{N}$. By definition of $C_{r+1}(p)$, for all $i \in \{1, \ldots, n\}$, there exists $Q_i \subseteq \perp\!\!\!\perp_r$ such that $p[K_i] \succ^1 Q_i$. By a case analysis on the structure of the head term of p (and by considering the definition of $\succ^1$), we must be in one of the following cases:

- p is of the form $tu \star \pi$, $\lambda x.t \star u \bullet \pi$, $cc \star t \bullet \pi$ or $k_{\pi'} \star t \bullet \pi$. In that case, there is one and only one q such that $p \succ^1_{\mathrm{K}} q$, and one can check that necessarily q is sound and for all i, $Q_i = \{q[K_i]\}$ (because $\{q[K_i]\}$ is the only R such that $\{p[K_i]\} \succ^1 R$). Therefore, for all i, $q[K_i] \in \perp\!\!\!\perp_r$, so $K_i \in C_r(q)$. This contradicts the induction hypothesis, because q is sound and $K_1 \cup \ldots \cup K_n = \mathbb{N}$.
- p is of the form $\phi \star t_0 \bullet \ldots \bullet t_n \bullet \pi$. In that case, for all $i \in \{1, \ldots, n\}$, there exists $j(i) \in \{0, \ldots, n\}$ such that $Q_i = \{(t_k \star \pi)[K_i]; k \neq j(i)\}$. Therefore, for all $i \in \{1, \ldots, n\}$, for all $k \in \{0, \ldots, n\}$ such that $k \neq j(i)$, $K_i \in C_r(t_k \star \pi)$.
  Let l be an element of $\{0, \ldots, n\}$ such that for all $i \in \{1, \ldots, n\}$, $j(i) \neq l$. Then for all i, $K_i \in C_r(t_l \star \pi)$. This contradicts the induction hypothesis, because $t_l \star \pi$ is sound and $K_1 \cup \ldots \cup K_n = \mathbb{N}$.
- p is of the form $\chi \star u \bullet \pi$. In that case, for all i, $Q_i = \{(u \star \beta_{1=k} \bullet \ldots \bullet \beta_{n=k} \bullet \pi)[K_i]; 1 \leq k \leq n\}$.
  Let $a_1, \ldots a_n \in \mathbb{N}$ such that for all i, there is no occurence of γ_{a_i} in p, let $q = u \star \gamma_{a_1} \bullet \ldots \bullet \gamma_{a_n} \bullet \pi$, and for all i, let $L_i = (K_i \setminus \{a_1, \ldots, a_n\}) \cup \{a_i\}$.
  Then for all i, $q[L_i] = (u \star \beta_{1=i} \bullet \ldots \bullet \beta_{1=k} \bullet \pi)[K_i] \in \perp\!\!\!\perp_r$. This contradicts the induction hypothesis because q is sound and $L_1 \cup \ldots \cup L_n = \mathbb{N}$.
- p is of the form $\gamma_j \star \pi$, with $j \in \mathbb{N}$. Since $K_1 \cup \ldots \cup K_n = \mathbb{N}$, there exists $i \in \{1, \ldots, n\}$ such that $j \in K_i$. Then the head term of $p[K_i]$ is $\beta_\top$, which contradicts $p[K_i] \succ^1 Q_i$, since there is no evaluation rule for $\beta_\top$.

Now, since $\perp\!\!\!\perp_0 = \emptyset$, for all sound p, $C_0(p) = \emptyset$, and so for all $K_1, \ldots, K_n \in C_0(p)$, $K_1 \cup \ldots \cup K_n \neq \mathbb{N}$: the result follows by induction. □

Corollary 54. *If p is a sound process, then $C(p) \neq \mathcal{P}(\mathbb{N})$.*

Corollary 55. *The realizability structure $\mathcal{S}$ is consistent.*

Proof. If t is proof-like and realizes $\bot$ with respect to $\mathcal{S}$, then in particular, $t \star \omega_0$ is sound and for all $K \subseteq \mathbb{N}$, $(t \star \omega_0)[K] = t \star \omega_0 \in \perp\!\!\!\perp$, which contradicts the previous corollary. □

8 Concluding remarks

We hope to have made apparent the connection between the different forms of nondeterminism and the hierarchy of "formulas about $\exists 2$". In this paper, we only studied this connection on Π^0_1 formulas: further work would include considering more complex first-order formulas, higher-order formulas, and formulas about the whole $\exists\mathbb{N}$. Moreover, it would be interesting to take a more serious look at the analogy with the hierarchy of parallelism: currently, little is known about it, and the methods presented here could help understand it better.

References

[1] G. Berry. 1976. Bottom-up computation of recursive programs. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications* 10, R1 (1976), 47–82. http://eudml.org/doc/92028

[2] P.J. Cohen. 1966. *Set theory and the continuum hypothesis.* W. A. Benjamin. https://books.google.fr/books?id=ZKc-AAAAIAAJ

[3] S. Givant and P. Halmos. 2008. *Introduction to Boolean Algebras.* Springer New York. https://books.google.fr/books?id=ORILyf8sF2sC

[4] Timothy G. Griffin. 1990. A Formulae-as-type Notion of Control. In *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '90).* ACM, New York, NY, USA, 47–58. https://doi.org/10.1145/96709.96714

[5] S. Kleene. 1952. Introduction to Metamathematics. (1952).

[6] Jean-Louis Krivine. 2011. Realizability algebras: a program to well order R. *Logical Methods in Computer Science* 7, 3 (2011). https://doi.org/10.2168/LMCS-7(3:2)2011

[7] Jean-Louis Krivine. 2012. Realizability algebras III: some examples. (2012). http://arxiv.org/abs/1210.5065

[8] Jean-Louis Krivine. 2014. On the structure of classical realizability models of ZF. (2014). http://arxiv.org/abs/1408.1868

[9] Jean-Louis Krivine. 2001. Typed lambda-calculus in classical Zermelo-Frænkel set theory. *Archive for Mathematical Logic* 40, 3 (01 Apr 2001), 189–205. https://doi.org/10.1007/s001530000057

[10] Jean-Louis Krivine. 2002. Lambda-calculus types and models. (Feb. 2002), 208 pages. https://cel.archives-ouvertes.fr/cel-00574575 Lecture.

[11] Jean-Louis Krivine. 2003. Dependent Choice, 'Quote' and the Clock. *Theor. Comput. Sci.* 308, 1-3 (Nov. 2003), 259–276. https://doi.org/10.1016/S0304-3975(02)00776-4

[12] Jean-Louis Krivine. 2009. Realizability in classical logic. *Panoramas et synthèses* 27 (2009), 197–229. https://hal.archives-ouvertes.fr/hal-00154500

[13] Jean-Louis Krivine. 2012. Realizability algebras II : new models of ZF + DC. *Logical Methods in Computer Science* 8, 1 (Feb. 2012), 10. https://doi.org/10.2168/LMCS-8(1:10)2012 28 p.

[14] Alexandre Miquel. 2009. Relating Classical Realizability and Negative Translation for Existential Witness Extraction. In *Typed Lambda Calculi and Applications*, Pierre-Louis Curien (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 188–202.

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

[16] Lionel Rieg. 2014. *On Forcing and Classical Realizability.* Theses. École normale supérieure de Lyon. https://tel.archives-ouvertes.fr/tel-01061442

[17] V Yu Sazonov. 1976. Degrees of parallelism in computations. In *International Symposium on Mathematical Foundations of Computer Science.* Springer, 517–523.

[18] Mark B Trakhtenbrot. 1975, Russian version 1974. On representation of sequential and parallel functions. In *International Symposium on Mathematical Foundations of Computer Science.* Springer, 411–417.