

CLASSICAL REALIZABILITY WITH FORCING AND THE AXIOM OF COUNTABLE CHOICE

ALEXANDRE MIQUEL

ABSTRACT. We present a framework for classical realizability that contains both Krivine’s classical realizability and Cohen forcing as particular cases. For that, we consider an extension of Krivine’s language λ_c where processes are ternary entities formed with a term (the proof), a stack (the counter-proof) and a state (the forcing condition). We prove that the usual typing rules for classical second-order logic are sound in this extended framework, and show the existence of storage operators for natural numbers. Then we instantiate this framework to the case where the state is a natural number (representing the contents of a reference), and show how to realize the countable axiom of choice in this setting, using a counter or a clock.

1. INTRODUCTION

2. THE LANGUAGE OF REALIZERS

2.1. Terms and stacks. *Terms* of the language of realizers (notation: t, u , etc.) are pure λ -terms enriched with two kinds of constants:

- *instructions* (notation: κ, κ' , etc.) ranging over a fixed set \mathcal{K} , that contains at least an instruction written \mathfrak{c} (‘call-cc’).
- *continuation constants* k_π , where π ranges over all stacks (see below).

Stacks (notation: π, π' , etc.) are lists of closed terms terminated by the stack constant \diamond (‘stack bottom’). Formally, terms and stacks are thus defined by mutual induction as follows:

$$\begin{array}{ll} \mathbf{Terms} & t, u ::= x \mid \lambda x. t \mid tu \mid \kappa \mid k_\pi \quad (\kappa \in \mathcal{K}) \\ \mathbf{Stacks} & \pi ::= \diamond \mid t \cdot \pi \quad (t \text{ closed}) \end{array}$$

(taking care that stacks are only formed with closed terms so that continuation constants k_π are actually constant).

The set of *closed* terms is written Λ , and the set of stacks is written Π . The set of free variables of a (possibly open) term t is written $FV(t)$ and the usual operation of substitution is written $t\{x := u\}$.

As in [2] we call a *quasi-proof* (or a *proof-like term*) any term t that contains no continuation constant k_π . This terminology comes from the fact that all the ‘proofs’ (in the sense of the type system of Fig. 1) are of this form.

2.2. States and processes. We consider a nonempty set \mathcal{S} whose elements are called *states* (notation: s, s' , etc.) We assume that the set \mathcal{S} of states comes with a preorder $s \leq s'$ that reads: ‘the state s' is stronger than s ’.¹

A *process* is a triple written $t \star \pi \star s$, where $t \in \Lambda$, $\pi \in \Pi$ and $s \in \mathcal{S}$. The set of all processes is written $\Lambda \star \Pi \star \mathcal{S}$. We assume that this set is equipped with

¹In forcing, states are traditionally called *conditions* and the corresponding preorder is written symmetrically—that is $s' \leq s$ rather than $s \leq s'$ —following the logical intuition that conditions that are more likely to be true (i.e. weaker conditions) should stand above conditions that are more likely to be false (i.e. stronger conditions). In this presentation, we shall stick to the computer theoretic intuition, putting stronger states above weaker ones.

a binary relation of *evaluation* written $p \succ p'$, whose reflexive-transitive closure is written $p \succ^* p'$ as usual. We also introduce the shorthand $t \star \pi \succ t' \star \pi'$ to express that for all $s \in \mathcal{S}$ there is some state $s' \geq s$ such that $t \star \pi \star s \succ^* t' \star \pi' \star s'$.

In what follows, we shall assume that evaluation is such that:

(GRAB)	$\lambda x . t \star u \cdot \pi$	\succ	$t\{x := u\} \star \pi$
(PUSH)	$tu \star \pi$	\succ	$t \star u \cdot \pi$
(CALL/CC)	$\mathbf{c} \star t \cdot \pi$	\succ	$t \star k_\pi \cdot \pi$
(RESUME)	$k_\pi \star t \cdot \pi'$	\succ	$t \star \pi$

Note that the four operations above are not required to be atomic, and they may modify the current state provided the new state is stronger than the old one. On the other hand, remaining instructions may do anything to the current state; there is no global requirement of monotonicity for the relation of evaluation.

We say that evaluation is *deterministic* when for every process p , there is at most one process p' such that $p \succ p'$ (one step evaluation). Given a process p , we call the *thread* of p and write $\mathbf{thd}(p)$ the set of all processes the process p evaluates to (in zero, one or several steps), that is: $\mathbf{thd}(p) = \{p' : p \succ^* p'\}$.

2.3. Saturated sets of processes. A set of processes $\perp \subseteq \Lambda \star \Pi \star \mathcal{S}$ is said to be *saturated* when it is closed under anti-evaluation, that is, when the conditions $p \succ p'$ and $p' \in \perp$ together imply $p \in \perp$ for all $p, p' \in \Lambda \star \Pi \star \mathcal{S}$. Given a saturated set \perp and two pairs $(t, s_1) \in \Lambda \times \mathcal{S}$ and $(\pi, s_2) \in \Pi \times \mathcal{S}$ we write $(t, s_1) \perp (\pi, s_2)$ when $t \star \pi \star s \in \perp$ for all s such that $s \geq s_1$ and $s \geq s_2$. In symbols:

$$(t, s_1) \perp (\pi, s_2) \triangleq \forall s (s \geq s_1 \wedge s \geq s_2 \Rightarrow t \star \pi \star s \in \perp)$$

Note that this relation is monotonic w.r.t. both components, in the sense that $(t, s_1) \perp (\pi, s_2)$, $s_1 \leq s'_1$ and $s_2 \leq s'_2$ imply $(t, s'_1) \perp (\pi, s'_2)$. (On the other hand, the set \perp is not required to be monotonic, since the conditions $t \star \pi \star s \in \perp$ and $s \leq s'$ do not necessarily imply $t \star \pi \star s' \in \perp$.)

2.4. Truth values and falsity values. A *truth value* is a subset of $\Lambda \times \mathcal{S}$ whereas a *falsity value* is a subset of $\Pi \times \mathcal{S}$. In what follows we shall consider arbitrary falsity values $F \subseteq \Pi \times \mathcal{S}$ while only considering truth values of the form

$$F^\perp \triangleq \{(t, s_1) \in \Lambda \times \mathcal{S} : \forall (\pi, s_2) \in F (t, s_1) \perp (\pi, s_2)\}$$

where $F \subseteq \Pi \times \mathcal{S}$ is an arbitrary falsity value, and where \perp is a fixed saturated set of processes that we call the *pole*. Note that truth values of the form F^\perp are monotonic in the sense that $(t, s) \in F^\perp$ and $s \leq s'$ imply $(t, s') \in F^\perp$.

Finally, given a truth value T and a falsity value F , we write $T \cdot F$ the falsity value defined by

$$T \cdot F \triangleq \{(t \cdot \pi, s) : (t, s) \in T \text{ and } (\pi, s_0) \in F \text{ for some } s_0 \leq s\}$$

(The disymmetry of the definition comes from the fact that it will only be used with monotonic truth values $T \subseteq \Lambda \times \mathcal{S}$.)

3. INTERPRETING FORMULÆ

3.1. The language of second-order arithmetic. Arithmetic expressions (notation: e, e' , etc.)—a.k.a. first-order terms—are built from first-order variables (notation: x, y, z , etc.) and function symbols for (at least) all primitive recursive functions. Given a closed arithmetic expression e , we call the *value of e* and write $\downarrow e$ the natural number denoted by e (interpreting symbols in e the obvious way).

Formulæ of second-order arithmetic (notation: A, B , etc.) are built from second-order variables (notation: X, Y, Z , etc.) of all arities and from arithmetic expressions using the following grammar:

Formulæ $A, B ::= X(e_1, \dots, e_k) \mid A \Rightarrow B \mid \forall x A \mid \forall X A$

Parametric formulæ are defined by enriching the language of formulæ above with a kary predicate symbol \dot{F} for every falsity value function $F : \mathbb{N}^k \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})$:

Parametric formulæ $A, B ::= \dots \mid \dot{F}(e_1, \dots, e_k)$

In the language of parametric formulæ, one can define the propositional constant \top as the formula associated with the empty falsity value: $\top \triangleq \dot{\emptyset}$.

The other constructions of second-order predicate logic (falsity, conjunction, disjunction, first- and second-order existential quantification) are encoded as follows:

$$\begin{aligned} \perp &\equiv \forall Z Z \\ \neg A &\equiv A \Rightarrow \perp \\ A \wedge B &\equiv \forall Z ((A \Rightarrow B \Rightarrow Z) \Rightarrow Z) \\ A \vee B &\equiv \forall Z ((A \Rightarrow Z) \Rightarrow (B \Rightarrow Z) \Rightarrow Z) \\ \exists x A(x) &\equiv \forall Z (\forall x (A(x) \Rightarrow Z) \Rightarrow Z) \\ \exists X A(X) &\equiv \forall Z (\forall X (A(X) \Rightarrow Z) \Rightarrow Z) \end{aligned}$$

(where Z is a fresh variable).

3.2. Truth and falsity values of a formula. Let \perp be a fixed saturated set of processes (the ‘pole’). Every closed parametric formula A is interpreted as two sets: a truth value $|A| \subseteq \Lambda \times \mathcal{S}$ and a falsity value $\|A\| \subseteq \Pi \times \mathcal{S}$. The falsity value $\|A\| \subseteq \Pi \times \mathcal{S}$ is defined by induction on the size of A by

$$\begin{aligned} \|\dot{F}(e_1, \dots, e_k)\| &\triangleq F(\downarrow e_1, \dots, \downarrow e_k) \\ \|A \Rightarrow B\| &\triangleq |A| \cdot \|B\| \\ \|\forall x A\| &\triangleq \bigcup_{n \in \mathbb{N}} \|A\{x := n\}\| \\ \|\forall X A\| &\triangleq \bigcup_{F: \mathbb{N}^k \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})} \|A\{X := \dot{F}\}\| \end{aligned}$$

whereas the truth value $|A| \subseteq \Lambda \times \mathcal{S}$ is defined by $|A| \triangleq \|A\|^\perp$. From the definition above, it is clear that:

$$|\forall x A| = \bigcap_{n \in \mathbb{N}} |A\{x := n\}| \quad |\forall X A| = \bigcap_{F: \mathbb{N}^k \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})} |A\{X := \dot{F}\}|$$

We then write:

- $(t, s) \Vdash A$ for $(t, s) \in |A|$; and
- $t \Vdash A$ when $(t, s) \Vdash A$ for all $s \in \mathcal{S}$.

(Note that these notions depend on the pole \perp .) From this, it is clear that:

$$\begin{aligned} t \Vdash \forall x A &\quad \text{iff} \quad \text{for all } n \in \mathbb{N} \quad t \Vdash A\{x := n\} \\ t \Vdash \forall X A &\quad \text{iff} \quad \text{for all } F : \mathbb{N}^k \rightarrow \mathfrak{P}(\Pi \star \mathcal{S}) \quad t \Vdash A\{x := \dot{F}\} \end{aligned}$$

Lemma 1. — *Let A and B be closed formulæ of the enriched language. Then for all $\pi \in \Pi$ and $s \in \mathcal{S}$:*

- (1) *If $(\pi, s) \in \|A\|$ then $(k_\pi, s) \in |A \Rightarrow B|$.*
- (2) *$(\mathfrak{c}, s) \Vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$*

Proof. (1) Assume that $(\pi, s_1) \in \|A\|$. To show that $(k_\pi, s_1) \in |A \Rightarrow B|$, consider an element of $\|A \Rightarrow B\|$ —that is: a pair of the form $(u \cdot \pi', s_2)$ where $(u, s_2) \in |A|$ and $(\pi', s_0) \in \|B\|$ for some $s_0 \leq s_2$ —and take a state s such that $s \geq s_1$ and $s \geq s_2$. Let $s' \geq s$ such that

$$k_\pi \star u \cdot \pi' \star s \succ^* u \star \pi \star s'.$$

From the assumptions above we get $u \star \pi \star s' \in \perp$ hence $k_\pi \star u \cdot \pi' \star s \in \perp$ by anti-reduction.

(2) Let $s_1 \in \mathcal{S}$. To show that $(\mathfrak{c}, s_1) \in |((A \Rightarrow B) \Rightarrow A) \Rightarrow A|$, consider an element of $\|((A \Rightarrow B) \Rightarrow A) \Rightarrow A\|$ —that is: a pair of the form $(u \cdot \pi, s_2)$ where $(u, s_2) \in |(A \Rightarrow B) \Rightarrow A|$ and $(\pi, s_0) \in \|A\|$ for some $s_0 \leq s_2$ —and take a state s such that $s \geq s_1$ and $s \geq s_2$. Let $s' \geq s$ such that

$$\mathfrak{c} \star u \cdot \pi \star s \succ^* u \star k_\pi \cdot \pi \star s'.$$

From (1) we know that $(k_\pi, s_0) \in |A \Rightarrow B|$, hence we get $(k_\pi \cdot \pi, s_0) \in \|(A \Rightarrow B) \Rightarrow A\|$ and thus $u \star k_\pi \cdot \pi \star s' \in \perp$ (since $s_2, s_0 \leq s'$). By anti-reduction we conclude that $\mathfrak{c} \star u \cdot \pi \star s \in \perp$. \square

4. DEGENERATED CASES

Let us now investigate degenerated cases of the construction depicted above.

4.1. Krivine realizability. Krivine realizability [2] immediately comes as a particular case of the generalized realizability model construction, namely, as the particular case where \mathcal{S} is a singleton (equipped with the obvious preorder).

4.2. Cohen forcing. More interesting is the case of the empty pole $\perp = \emptyset$ (with an arbitrary nonempty set of states \mathcal{S}), that corresponds to Cohen forcing. To understand this point, let us recall that given a set of forcing conditions (\mathcal{S}, \geq) , the binary relation over \mathcal{S} defined by

$$s_1 \perp s_2 \triangleq \neg \exists s (s \geq s_1 \wedge s \geq s_2) \quad ('s_1 \text{ and } s_2 \text{ are incompatible}')$$

induces a complete Boolean algebra $(\mathbb{B}(\mathcal{S}), \subseteq)$ whose carrier is

$$\mathbb{B}(\mathcal{S}) \triangleq \{X \in \mathfrak{P}(\mathcal{S}) : X = X^{\perp\perp}\}.$$

Forcing consists in interpreting every formula A of the language (ZF, PA2, etc.) as an element $\llbracket A \rrbracket \in \mathbb{B}(\mathcal{S})$ following the correspondence

$$\begin{aligned} \llbracket \neg A \rrbracket &\triangleq \llbracket A \rrbracket^\perp & \llbracket A \Rightarrow B \rrbracket &\triangleq (\llbracket A \rrbracket \cap \llbracket B \rrbracket^\perp)^\perp \\ \llbracket A \wedge B \rrbracket &\triangleq \llbracket A \rrbracket \cap \llbracket B \rrbracket & \llbracket A \vee B \rrbracket &\triangleq (\llbracket A \rrbracket \cup \llbracket B \rrbracket)^{\perp\perp} \\ \llbracket \forall x A(x) \rrbracket &\triangleq \bigcap_{x \in D} \llbracket A(x) \rrbracket & \llbracket \exists x A(x) \rrbracket &\triangleq \left(\bigcup_{x \in D} \llbracket A(x) \rrbracket \right)^{\perp\perp} \end{aligned}$$

The relation of forcing $s \Vdash A$ (' s forces A ') is then defined as $s \Vdash A \triangleq s \in \llbracket A \rrbracket$.

A simple way to implement this definition in second-order logic is to interpret every k -ary second-order variable X as a function $G : \mathbb{N}^k \rightarrow \mathfrak{P}(\mathcal{S})$ mapping k -uples of individuals to arbitrary set of forcing conditions that intuitively force the negation of the corresponding proposition, so that the atomic formula $X(n_1, \dots, n_k)$ is actually interpreted as $\llbracket X(n_1, \dots, n_k) \rrbracket \triangleq G(n_1, \dots, n_k)^\perp$ (that constitutes an element of $\mathbb{B}(\mathcal{S})$). The rest of the interpretation is then defined according to the correspondence depicted above.

It is a simple exercise to check that in the case where $\perp = \emptyset$, the binary relation $(t, s_1) \perp (\pi, s_2)$ defined in 2.3 collapses to the relation $s_1 \perp s_2$, so that the generalized realizability interpretation defined in 3.2 actually mimics the forcing interpretation $A \mapsto \llbracket A \rrbracket$ described above:

Proposition 1. — Assume that $\perp = \emptyset$. Then for every closed formula A of second-order logic, one has $|A| = \Lambda \times \llbracket A \rrbracket$.

Proof. We first extend the definition of forcing to parametric formulæ of second-order logic (such as defined in 3.1) by letting

$$\llbracket \dot{F}(e_1, \dots, e_k) \rrbracket \triangleq \pi_2(F(\downarrow e_1, \dots, \downarrow e_k))^\perp$$

(writing $\pi_2 : \Pi \times \mathcal{S} \rightarrow \mathcal{S}$ the second projection) and then prove the desired result by induction on the size of the parametric formula A . \square

4.3. The standard 2-valued model. In the case where \mathcal{S} is a singleton, the Boolean algebra $\mathbb{B}(\mathcal{S})$ reduces to the 2-point algebra $\mathbb{B}(\mathcal{S}) = \{\emptyset; \mathcal{S}\}$, in which case the corresponding forcing model is isomorphic to the standard model of second-order logic. In our framework, this corresponds to the case where \mathcal{S} is a singleton and where the pole \perp is empty.

The situation is summarized in the following table:

	$\perp \neq \emptyset$	$\perp = \emptyset$
$\#\mathcal{S} > 1$	Generalized realizability	Cohen forcing
$\#\mathcal{S} = 1$	Krivine realizability	Standard 2-valued model

5. TYPING

5.1. The type system for second-order logic. To facilitate the construction of realizers, we introduce a type system based on a judgment $\Gamma \vdash t : A$ ('in context Γ , the term t has type A '), where A is an open formula of the enriched language and Γ a typing context mapping finitely many term-variables to open formulæ. In what follows, we write $\text{dom}(\Gamma)$ the domain of a context Γ , and given two contexts Γ and Γ' that coincide on the intersection of their domains we write Γ, Γ' their union.

The inference rules for second-order logic are depicted in Fig. 1

$\overline{\Gamma \vdash x : A} \quad (x:A) \in \Gamma$	$\overline{\Gamma \vdash t : \top} \quad FV(t) \subseteq \text{dom}(\Gamma)$
$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x. t : A \Rightarrow B}$	$\frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B}$
$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall x A} \quad x \notin FV(\Gamma)$	$\frac{\Gamma \vdash t : \forall x A}{\Gamma \vdash t : A\{x := e\}}$
$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall X A} \quad X \notin FV(\Gamma)$	$\frac{\Gamma \vdash t : \forall X A}{\Gamma \vdash t : A\{X(x_1, \dots, x_k) := B\}}$
$\overline{\Gamma \vdash \mathbf{c} : ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$	

FIGURE 1. Inference rules for second-order logic

5.2. Adequacy. Let us now define the realizability interpretation of typing judgments w.r.t. a fixed pole \perp . We call a *valuation* any function ρ that maps every first-order variable x to a natural number $\rho(x) \in \mathbb{N}$ and every kary second-order variable X to a falsity value function $\rho(X) : \mathbb{N}^k \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})$. Given a (possibly) open formula A and a valuation ρ , we write $A[\rho]$ the closed formula of the enriched language obtained by replacing each occurrence of a free (first- or second-order) variable by its value according to ρ . A *substitution* is a finite association list $\sigma = [x_1 := u_1; \dots; x_n := u_n]$, where x_1, \dots, x_n are pairwise distinct term-variables

and where u_1, \dots, u_n are closed terms. The domain of σ is written $\text{dom}(\sigma)$, and for all terms t we write $t[\sigma] \triangleq t\{x_1 := u_1; \dots; x_n := u_n\}$. Given a substitution σ , a state $s \in \mathcal{S}$ and a context Γ of closed formulæ, we write $(\sigma, s) \Vdash \Gamma$ when $\text{dom}(\sigma) = \text{dom}(\Gamma)$ and $(\sigma(x), s) \Vdash \Gamma(x)$ for every $x \in \text{dom}(\Gamma)$.

With these notations, we say that the judgment $\Gamma \vdash t : A$ is *valid* (w.r.t. \perp) when for all valuations ρ , for all states $s \in \mathcal{S}$ and for all substitutions σ such that $(\sigma, s) \Vdash \Gamma[\rho]$ we have $(t[\sigma], s) \Vdash A[\rho]$. We extend the notion of validity to inference rules, saying that the rule

$$\frac{P_1 \quad \dots \quad P_n}{C}$$

is *valid* (w.r.t. \perp) when the validity of the premises P_1, \dots, P_n implies the validity of the conclusion C . From the latter definition, it is clear that the conclusion of any typing derivation formed with only valid inference rules is valid.

Proposition 2 (Adequacy). — *The inference rules of Fig. 1 are valid w.r.t. all saturated sets \perp .*

Proof.

- Axiom and \top -intro. The validity of these rules is obvious.
- Introduction of implication. Assume that the judgment $\Gamma, x : A \vdash t : B$ is valid, and consider a valuation ρ , a substitution σ and a state s_1 such that $(\sigma, s_1) \Vdash \Gamma[\rho]$. To show that $((\lambda x. t)[\sigma], s_1) \Vdash A[\rho] \Rightarrow B[\rho]$, let us consider an element of $\|A[\rho] \Rightarrow B[\rho]\|$ —that is: a pair of the form $(u \cdot \pi, s_2)$ where $(u, s_2) \in |A[\rho]|$ and $(\pi, s_0) \in \|B[\rho]\|$ for some $s_0 \leq s_2$ —and take a state s such that $s \geq s_1$ and $s \geq s_2$. Let $s' \geq s$ such that

$$(\lambda x. t)[\sigma] \star u \cdot \pi \star s \succ^* t[\sigma; x := u] \star \pi \star s'.$$

It is clear that $(\sigma; x := u, s') \Vdash (\Gamma, x : A)[\rho]$. From our initial assumption, we get $(t[\sigma; x := u], s') \Vdash B[\rho]$, hence $t[\sigma; x := u] \star \pi \star s' \in \perp$ (using $s_0 \leq s'$). By anti-reduction we conclude that $(\lambda x. t)[\sigma] \star u \cdot \pi \star s \in \perp$.

- Elimination of implication. Assume that the judgments $\Gamma \vdash t : A \Rightarrow B$ and $\Gamma \vdash u : A$ are valid, and consider a valuation ρ , a substitution σ and a state s_1 such that $(\sigma, s_1) \Vdash \Gamma[\rho]$. To show that $((tu)[\sigma], s_1) \Vdash B[\rho]$, let us consider an element $(\pi, s_2) \in \|B[\rho]\|$ and take a state s such that $s \geq s_1$ and $s \geq s_2$. Let $s' \geq s$ such that

$$(tu)[\sigma] \star \pi \star s \succ^* t[\sigma] \star u[\sigma] \cdot \pi \star s'.$$

From the validity of $\Gamma \vdash u : A$ we get $(u[\sigma], s_1) \in |A|$ and thus $(u[\sigma], s) \in |A|$, so that $(u[\sigma] \cdot \pi, s) \in \|A[\rho] \Rightarrow B[\rho]\|$ (using $s_2 \leq s$). But since $(t[\sigma], s_1) \in |A[\rho] \Rightarrow B[\rho]|$, we get $t[\sigma] \star u[\sigma] \cdot \pi \star s' \in \perp$. By anti-reduction we conclude that $(tu)[\sigma] \star \pi \star s \in \perp$.

- Introduction and elimination of first-order quantification. The validity of both rules follows from the equality $|\forall x A| = \bigcap_{n \in \mathbb{N}} |A\{x := n\}|$.
- Introduction and elimination of second-order quantification. This case is similar to the case of first-order quantification.
- Peirce's law. This case immediately follows from Lemma 1. \square

5.3. Subtyping. Given two (possibly) open formulæ A and B of the enriched language, we say that A is a *subtype* of B and write $A \leq B$ when for all valuations ρ we have $\|B[\rho]\| \subseteq \|A[\rho]\|$. This relation obviously implies—but is not equivalent to—the inclusion $|A[\rho]| \subseteq |B[\rho]|$. We also write $A \equiv B$ when $A \leq B$ and $B \leq A$.

It is clear that the relation $A \leq B$ (resp. the relation $A \equiv B$) is a preorder (resp. an equivalence). Moreover:

Proposition 3. — *The relation $A \leq B$ enjoys the following rules:*

- (1) $\perp \leq A$ and $A \leq \top$ (where $\perp = \forall XX$ and $\top = \dot{\emptyset}$)

- (2) If $A' \leq A$ and $B \leq B'$, then $A \Rightarrow B \leq A' \Rightarrow B'$
- (3) $\forall x A \leq A\{x := e\}$
- (4) $\forall X A \leq A\{X(x_1, \dots, x_k) := B\}$
- (5) If $A \leq B$ and $x \notin FV(A)$, then $A \leq \forall x B$
- (6) If $A \leq B$ and $X \notin FV(A)$, then $A \leq \forall X B$
- (7) If $A \leq A'$, then $\forall x A \leq \forall x A'$ and $\forall X A \leq \forall X A'$.
- (8) $\forall x (A \Rightarrow B) \leq \forall x A \Rightarrow \forall x B$
- (9) $\forall X (A \Rightarrow B) \leq \forall X A \Rightarrow \forall X B$
- (10) If $x \notin FV(A)$, then $\forall x (A \Rightarrow B) \equiv A \Rightarrow \forall x B$
- (11) If $X \notin FV(A)$, then $\forall X (A \Rightarrow B) \equiv A \Rightarrow \forall X B$
- (12) $(A \Rightarrow \top) \equiv \top$
- (13) If $\Gamma \vdash t : A$ and $A \leq A'$ are valid, then so is $\Gamma \vdash t : A'$.

6. FROM SECOND-ORDER LOGIC TO SECOND-ORDER ARITHMETIC

6.1. Realizing equalities and inequalities. In second-order logic, the equality predicate $e_1 = e_2$ is defined by letting

$$e_1 = e_2 \triangleq \forall Z (Z e_1 \Rightarrow Z e_2)$$

We easily check that:

Lemma 2. — *Given two closed arithmetic expressions e_1 and e_2 , we have:*

$$e_1 = e_2 \equiv \begin{cases} \forall Z (Z \Rightarrow Z) & \text{if } \downarrow e_1 = \downarrow e_2 \\ \top \Rightarrow \perp & \text{if } \downarrow e_1 \neq \downarrow e_2 \end{cases}$$

where

$$\begin{aligned} \|\forall Z (Z \Rightarrow Z)\| &= \{(t \cdot \pi, s) : t \star \pi \star s \in \perp\} \\ \|\top \Rightarrow \perp\| &= \{(t \cdot \pi, s) : t \in \Lambda, \pi \in \Pi, s \in \mathcal{S}\} \end{aligned}$$

From this we deduce that:

Lemma 3. (1) *If the equality $\downarrow e_1(n_1, \dots, n_k) = \downarrow e_2(n_1, \dots, n_k)$ is true for all $n_1, \dots, n_k \in \mathbb{N}$, then*

$$\lambda z. z \Vdash \forall x_1 \dots \forall x_k e_1(x_1, \dots, x_k) = e_2(x_1, \dots, x_k)$$

In particular, the term $\lambda z. z$ realizes all the defining equations of all primitive recursive function symbols involved in arithmetic expressions.

- (2) $\lambda x. x \Vdash \forall x \forall y (s(x) = s(y) \Rightarrow x = y)$ (3rd Peano axiom)
 - (3) $\lambda x. xu \Vdash \forall x \neg(s(x) = 0)$ (4th Peano axiom)
- (where u is an arbitrary term, such that $FV(u) \subseteq \{x\}$).

6.2. Realizing the induction principle. In second-order arithmetic, the smallest class containing zero and closed under the successor function is defined by

$$\text{Nat}(x) \triangleq \forall Z [\forall y (Z(y) \Rightarrow Z(s(y))) \Rightarrow Z(0) \Rightarrow Z(x)]$$

It is easy to show that in general, the formula $\forall x \text{Nat}(x)$ is not realized by a quasi-proof (cf [2]). However, we can still benefit from induction provided we relativize all quantifications to the class Nat

$$\begin{aligned} \forall^{\text{Nat}} x A(x) &\triangleq \forall x (\text{Nat}(x) \Rightarrow A(x)) \\ \exists^{\text{Nat}} x A(x) &\triangleq \forall Z (\forall x (\text{Nat}(x) \Rightarrow A(x) \Rightarrow Z) \Rightarrow Z) \end{aligned}$$

so that the following (relativized) induction principle is immediately derivable:

$$\forall Z (X(0) \Rightarrow \forall^{\text{Nat}} y (Z(y) \Rightarrow Z(s(y))) \Rightarrow \forall^{\text{Nat}} x Z(x))$$

Note that the relativized quantification $\forall^{\text{Nat}} x A(x) \triangleq \forall x (\text{Nat}(x) \Rightarrow A(x))$ can be eliminated with an expression e only in the case where we have a realizer (or a proof) of the formula $\text{Nat}(e)$. In practice, this is always the case when e is an

expression formed from first-order variables x_i ($1 \leq i \leq k$) such that $\text{Nat}(x_i)$ using primitive recursive function symbols, since:

Lemma 4. (1) *For every kary primitive recursive function symbol f , the formula*

$$\forall^{\text{Nat}}x_1 \dots \forall^{\text{Nat}}x_k \text{Nat}(f(x_1, \dots, x_k))$$

is derivable (and thus realizable by a quasi-proof).

(2) *For every arithmetic expression $e(x_1, \dots, x_k)$ depending on k first-order variables x_1, \dots, x_k , the formula*

$$\forall^{\text{Nat}}x_1 \dots \forall^{\text{Nat}}x_k \text{Nat}(e(x_1, \dots, x_k))$$

is derivable (and thus realizable by a quasi-proof).

From this, we easily check that

Lemma 5 (Adequacy in PA2). — *If a closed formula A is provable in classical second-order arithmetic (PA2), then the formula A^{Nat} obtained by relativizing all first-order quantifications with the predicate Nat is realized by a quasi-proof (independently from the choice of the pole \perp).*

As an example we can find a quasi-proof that realizes the minimum principle

$$\forall z \forall Z [\exists^{\mathbb{N}}x A(x) \Rightarrow \exists^{\mathbb{N}}x (A(x) \wedge \forall^{\mathbb{N}}y (A(y) \Rightarrow x \leq y))]$$

(writing $x \leq y$ for $x - y = 0$).

6.3. Storage operators. Every natural number $n \in \mathbb{N}$ is represented in the language of realizers as the closed term \bar{n} defined by $\bar{n} = \bar{s}^n \bar{0}$ (Krivine numeral n), where $\bar{s} = \lambda n f x . f(n f x)$ and $\bar{0} = \lambda f x . x$. From the type system defined in the latter section, it is clear that

$$\bar{0} : \text{Nat}(0) \quad \text{and} \quad \bar{s} : \forall^{\text{Nat}}x \text{Nat}(s(x))$$

so that $\bar{n} : \text{Nat}(n)$ (and thus $\bar{n} \Vdash \text{Nat}(n)$) for all $n \in \mathbb{N}$. Note that although Krivine numeral \bar{n} is β -convertible (as a pure λ -term) to Church-numeral $\lambda f x . f^n x$, it is not equivalent to it when only considering weak head-reduction.

Let us enrich the language of formulæ with a new form of implication

$$A, B ::= \dots \mid \{e\} \Rightarrow B$$

(where e is an arithmetic expression and B a formula) that intuitively represents the set of all functions that produce a B when applied to the Krivine numeral representing the arithmetic expression e . Formally, we extend the interpretation of (closed) formulæ by defining the falsity value of $\{e\} \Rightarrow B$

$$\|\{e\} \Rightarrow B\| \triangleq \{(\bar{n} \cdot \pi, s) : n = \downarrow e, (\pi, s) \in \|B\|\}$$

From this definition combined with the fact that \bar{n} is always a realizer of $\text{Nat}(n)$, it is clear that $\text{Nat}(e) \Rightarrow B \leq \{e\} \Rightarrow B$, so that the corresponding implication $(\text{Nat}(e) \Rightarrow B) \Rightarrow \{e\} \Rightarrow B$ is realized by the identity term.

We say that a closed quasi-proof M is a *storage operator* (for natural numbers) when this term realizes the converse implication, that is, when

$$M \Vdash \forall x \forall Z (\{x\} \Rightarrow Z \Rightarrow \text{Nat}(x) \Rightarrow Z).$$

We can check that storage operators exist, since:

Lemma 6. — *The term $M \triangleq \lambda f n . n(\lambda h x . h(\bar{s} x)) h \bar{0}$ is a storage operator.*

By an obvious argument of subtyping (Prop. 3 item (7)), it is clear that when M is a storage operator we have

$$M \Vdash \forall x (\{x\} \Rightarrow A(x)) \Rightarrow \forall^{\text{Nat}} x A(x)$$

This suggests another way of defining relativized quantifications by letting:

$$\begin{aligned} \forall^{\text{N}} x A(x) &\triangleq \forall x (\{x\} \Rightarrow A(x)) \\ \exists^{\text{N}} x A(x) &\triangleq \forall Z (\forall x (\{x\} \Rightarrow A(x) \Rightarrow Z) \Rightarrow Z) \end{aligned}$$

From the existence of storage operators, it is clear that the formulæ $\forall^{\text{N}} x A(x)$ and $\exists^{\text{N}} x A(x)$ defined above are equivalent to the relativized formulæ $\forall^{\text{Nat}} x A(x)$ and $\exists^{\text{Nat}} x A(x)$ (respectively), since

$$\begin{array}{ll} \forall^{\text{Nat}} x A(x) \leq \forall^{\text{N}} x A(x) & M \Vdash \forall^{\text{N}} x A(x) \Rightarrow \forall^{\text{Nat}} x A(x) \\ \exists^{\text{N}} x A(x) \leq \exists^{\text{Nat}} x A(x) & M' \Vdash \exists^{\text{Nat}} x A(x) \Rightarrow \exists^{\text{N}} x A(x) \end{array}$$

where $M' \triangleq \lambda z. z (M \lambda x y z. z x y)$. In what follows, we shall prefer the more compact forms $\forall^{\text{N}} x A(x)$ and $\exists^{\text{N}} x A(x)$ to the formulæ $\forall^{\text{Nat}} x A(x)$ and $\exists^{\text{Nat}} x A(x)$.

6.4. Extensionality. Given two kary second-order variables X, Y , let us write

$$X \approx Y \triangleq \forall z_1 \cdots \forall z_k (X(z_1, \dots, z_k) \Leftrightarrow Y(z_1, \dots, z_k)).$$

Intuitively, this notation expresses that the sets (of kuples of individuals) represented by X and Y are extensionally equivalent. (Of course, extensional equality of sets as predicates does not mean that they are denoted by the same falsity value functions.) An interesting property of the formulæ of the language of second-order logic is that they are extensional w.r.t. second-order variables, in the sense that:

Lemma 7. — *For every formula $A(X)$ of second-order logic that (possibly) depends on a free second-order variable X , the following formula is derivable:*

$$X \approx Y \Rightarrow (A(X) \Leftrightarrow A(Y)).$$

Proof. By structural induction on the formula A . □

In particular this means that for every formula $A(X)$ of second-order logic we can build a quasi-proof $e_{A,X}$ such that

$$e_{A,X} \Vdash \forall \vec{z} \forall \vec{Z} (X \approx Y \Rightarrow (A(X) \Leftrightarrow A(Y)))$$

(where \vec{z} and \vec{Z} are the remaining free variables of A). Note that this property still holds if we consider parametric formulæ in the sense of 3.1, or even formulæ containing the second form of implication $\{e\} \Rightarrow B$ introduced in 6.3. On the other hand, the property of extensionality is no more automatically true when considering formulæ of third-order logic or of higher-order languages.

7. UNIVERSAL AND LOCAL REALIZERS

7.1. Universal realizers. Although the definition of the relation $t \Vdash A$ depends on the choice of the pole \perp , the reader may have noticed that none of the realizers we explicitly constructed in sections 5 and 6 depend on it.

Formally, we thus say that a closed term t is a *universal realizer* of a closed formula A and write $t \Vdash A$ when t realizes A for every choice of the pole \perp . From the property of adequacy (Prop. 2), it is clear that any closed term of type A (in the sense of Fig. 1) is a universal realizer of A , that is: $\vdash t : A$ implies $t \Vdash A$.

From the results of section 6, this can be generalized as follows:

Proposition 4. — *If a formula A is provable in second-order arithmetic, then the formula A^{Nat} has a universal realizer.*

Remember that when the pole is empty, the truth value of a formula is given by $|A| = \Lambda \times \llbracket A \rrbracket$ (Prop. 1), where $\llbracket A \rrbracket \in \mathbb{B}(\mathcal{S})$ is the set of conditions that force A (cf subsection 4.2). Consequently:

Proposition 5. — *Every closed formula A that has a universal realizer is true in the sense that it is forced by any condition, that is: $\llbracket A \rrbracket = \mathcal{S}$.*

In particular, the formula $\perp \triangleq \forall X X$ has no universal realizer (since $\llbracket \perp \rrbracket = \emptyset$).

In many situations however, the criterion of universal realizability is too strong and it is desirable to relax it by considering terms that realize a formula only w.r.t. the poles of a particular form, namely: local poles.

7.2. Local realizers. A pole \perp is said to be *local* when its complement is included into the thread of some process p_0 , namely: $\mathbb{C}\perp \subseteq \mathbf{thd}(p_0)$. We can notice that:

- (1) Every set of processes of the form $\perp = \mathbf{Cthd}(p_0)$ (for some process p_0) is saturated, and thus constitutes a local pole.
- (2) Conversely, it is easy to check that when evaluation is deterministic, every local pole \perp is of the form $\perp = \mathbf{Cthd}(p_0)$ for some process p_0 .

In what follows, we say that a closed term t is a *local* realizer of a closed (parametric) formula A and write $t \Vdash A$ when $t \Vdash A$ for every local pole \perp . Universal realizers of a formula are thus local realizers of it, but the converse is not true. Moreover, it is easy to check that the formula $\perp \triangleq \forall X X$ has no local realizer too.

The interest of the notion of a local realizer is that it is weaker than the notion of a universal realizer—thus allowing to realize more formulæ—while being sufficient to allow the prediction of the computational behaviour of realizers, which is essential for program extraction. The typical example is the extraction of a (reliable) witness from a local realizer of a Σ_1^0 -formula:

Proposition 6 (Σ_1^0 -extraction). — *Let us consider:*

- a unary (primitive recursive) function symbol f ;
- a closed term d that decides the nullity of f , in the sense that for all $n \in \mathbb{N}$, $u, v \in \Lambda$, $\pi \in \Pi$ and $s \in \mathcal{S}$, there is a state $s' \geq s$ such that

$$d \star \bar{n} \cdot u \cdot v \cdot \pi \star s \quad \succ^* \quad \begin{cases} u \star \pi \star s' & \text{if } f(n) = 0 \\ v \star \pi \star s' & \text{otherwise} \end{cases}$$

- a local realizer t of the formula $\exists^{\mathbb{N}} x f(x) = 0$.

Then for all $u \in \Lambda$, $\pi \in \Pi$ and $s \in \mathcal{S}$, there is a number $n \in \mathbb{N}$ such that $f(n) = 0$ and a state $s' \geq s$ such that

$$t \star (\lambda xy. d x (u x) (y x)) \cdot \pi \star s \quad \succ^* \quad u \star \bar{n} \cdot \pi \star s'.$$

Proof. Let us set $t' \triangleq \lambda xy. d x (u x) (y x)$, $p_0 \triangleq t \star t' \cdot \pi \cdot s$, $F \triangleq \{(\pi, s)\}$ and consider the local pole $\perp = \mathbf{Cthd}(p_0)$. Reasoning by contradiction, let us assume that for all $n \in \mathbb{N}$ such that $f(n) = 0$ and for all $s' \geq s$, we have

$$p_0 \not\succeq^* u \star \bar{n} \cdot \pi \star s'$$

that is: $u \star \bar{n} \cdot \pi \star s' \in \perp$. From this hypothesis, we get that $(u, s) \Vdash \{n\} \Rightarrow \dot{F}$ for all n such that $f(n) = 0$. Let us now show that $t' \Vdash \forall x (\{x\} \Rightarrow f(x) = 0 \Rightarrow \dot{F})$. For that, let us consider an arbitrary state $s_1 \in \mathcal{S}$ and an arbitrary element of $\|\forall x (\{x\} \Rightarrow f(x) = 0 \Rightarrow \dot{F})\|$ —that is: a pair of the form $(\bar{n} \cdot v \cdot \pi, s_2)$ where $(v, s_2) \Vdash f(n) = 0$ and $s \leq s_2$ —and take a state s' such that $s' \geq s_1$ and $s' \geq s_2$. We distinguish two cases, depending on whether $f(n) = 0$ or not.

(1) $f(n) = 0$. In this case we have

$$t' \star \bar{n} \cdot v \cdot \pi \star s' \succ^* d \star \bar{n} \cdot (u \bar{n}) \cdot (v \bar{n}) \cdot \pi \star s'' \succ^* u \star \bar{n} \cdot \pi \star s'''$$

for some $s''' \geq s'' \geq s' (\geq s_2 \geq s)$. Since $u \star \bar{n} \cdot \pi \star s''' \in \perp$ (from our initial hypothesis) we get $t' \star \bar{n} \cdot v \cdot \pi \star s' \in \perp$ by anti-reduction.

(2) $f(n) \neq 0$. In this case we have

$$t' \star \bar{n} \cdot v \cdot \pi \star s' \succ^* d \star \bar{n} \cdot (u \bar{n}) \cdot (v \bar{n}) \cdot \pi \star s'' \succ^* v \star \bar{n} \cdot \pi \star s'''$$

for some $s''' \geq s'' \geq s'$. But since $(v, s_2) \in |f(n) = 0| = |\top \Rightarrow \perp|$, $s''' \geq s_2$ and $(\bar{n} \cdot \pi, s''') \in \|\top \Rightarrow \perp\|$ we get $v \star \bar{n} \cdot \pi \star s''' \in \perp$ from which we deduce that $t' \star \bar{n} \cdot v \cdot \pi \star s' \in \perp$ by anti-reduction.

We have shown that $t' \star \bar{n} \cdot v \cdot \pi \star s' \in \perp$. Hence $t' \Vdash \forall x (\{x\} \Rightarrow f(x) = 0 \Rightarrow \dot{F})$. Thus $(t' \cdot \pi, s) \in \|\forall x (\{x\} \Rightarrow f(x) = 0 \Rightarrow \dot{F}) \Rightarrow \dot{F}\| \subseteq \|\exists^{\mathbb{N}} x f(x) = 0\|$, and since $t \Vdash \exists^{\mathbb{N}} x f(x) = 0$ we get $p_0 \triangleq t \star t' \cdot \pi \star s \in \perp$. But this contradicts the fact that $p_0 \notin \mathbf{Cthd}(p_0) = \perp$. Therefore our initial hypothesis is wrong, which means that $p_0 \succ^* u \star \bar{n} \cdot \pi \star s'$ for some $n \in \mathbb{N}$ such that $f(n) = 0$ and for some $s' \geq s$. \square

8. REALIZING THE COUNTABLE AXIOM OF CHOICE

8.1. The principle of countable selection. Let \perp be a fixed pole. We say that a closed term $\chi \in \Lambda$ realizes the *principle of countable selection* (w.r.t. \perp) when for every parametric formula $A(Y)$ that only depends on a kary second-order variable Y , there exists a falsity value function $\Phi_A : \mathbb{N}^{1+k} \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})$ such that

$$\chi \Vdash \forall^{\mathbb{N}} y A(\dot{\Phi}_A(y)) \Rightarrow \forall Y A(Y),$$

writing $\forall^{\mathbb{N}} y B(y)$ for $\forall y (\{y\} \Rightarrow B(y))$. (We shall see in 8.3 several examples of terms that realize the principle of countable selection.) It is easy to check that any closed term that realizes the principle of countable selection also realizes the following parametric form of this principle:

Proposition 7. — *Let χ be a closed term that realizes the principle of countable selection. Then more generally, for every formula with parameters $A(x_1, \dots, x_p, Y)$ that depends on p first-order variables x_1, \dots, x_p (hereafter abbreviated as \vec{x}) and on a kary second-order variable Y :*

(1) *There exists a falsity value function $\Phi_A : \mathbb{N}^{p+1+k} \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})$ such that*

$$\chi \Vdash \forall \vec{x} (\forall^{\mathbb{N}} y A(\vec{x}, \dot{\Phi}_A(\vec{x}, y)) \Rightarrow \forall Y A(\vec{x}, Y)).$$

(2) *Considering a fresh term variable z and a fresh second-order variable Z of arity $p + 1 + k$, we have*

$$\lambda z. z\chi \Vdash \exists Z \forall \vec{x} (\forall^{\mathbb{N}} y A(\vec{x}, Z(\vec{x}, y)) \Rightarrow \forall Y A(\vec{x}, Y)),$$

Proof. (1) Using the countable axiom of choice, we associate to each tuple of parameters $\vec{n} \in \mathbb{N}^p$ a function $\Phi_{A, \vec{n}} : \mathbb{N}^{1+k} \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})$ such that

$$\chi \Vdash \forall^{\mathbb{N}} y A(\vec{n}, \dot{\Phi}_{A, \vec{n}}(y)) \Rightarrow \forall Y A(\vec{n}, Y)$$

(applying the principle of countable selection with the formula $A(\vec{n}, Y)$ that only depends on the variable Y). Then we define $\Phi_A : \mathbb{N}^{p+1+k} \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})$ by letting $\Phi_A(\vec{n}) = \Phi_{A, \vec{n}}$ for all $\vec{n} \in \mathbb{N}^p$. By definition we have

$$\chi \Vdash \forall^{\mathbb{N}} y A(\vec{n}, \dot{\Phi}_A(\vec{n}, y)) \Rightarrow \forall Y A(\vec{n}, Y)$$

for all $\vec{n} \in \mathbb{N}^p$, hence

$$\chi \Vdash \forall \vec{x} (\forall^{\mathbb{N}} y A(\vec{x}, \dot{\Phi}_A(\vec{x}, y)) \Rightarrow \forall Y A(\vec{x}, Y)).$$

- (2) Let $B(Z) \triangleq \forall \vec{x} (\forall^{\mathbb{N}} y A(\vec{x}, Z(\vec{x}, y)) \Rightarrow \forall Y A(\vec{x}, Y))$ (where Z is a fresh second-order variable of arity $p + 1 + k$). Using the type system of Fig. 1, we easily derive the judgment $u : B(\Phi) \vdash \lambda z . zu : \exists Z B(Z)$. Then we conclude with Prop. 2 and (1) using the substitution $[u := \chi]$. \square

8.2. From the countable selection to the countable choice. In second-order arithmetic, the parametric principle of countable selection (classically) implies the axiom of countable choice as follows. Let $A(x, Y)$ be a formula depending a first-order variable x and on a k -ary second-order variable Y , and let us assume that

$$\exists Z \forall x (\forall^{\mathbb{N}} y \neg A(x, Z(x, y)) \Rightarrow \forall Y \neg A(x, Y)) \quad (1)$$

(where Z is a $(k + 2)$ -ary second-order variable). From Prop. 7 (2) applied to the formula $\neg A$ we know that the formula above is realized by $\lambda z . z\chi$ provided the principle of countable selection is realized by a suitable closed term χ . By contraposition, (1) becomes

$$\exists Z \forall x (\exists Y A(x, Y) \Rightarrow \exists^{\mathbb{N}} y A(x, Z(x, y))) \quad (2)$$

Let us now consider the formula $U_Z(x, \vec{z})$ defined by

$$U_Z(x, \vec{z}) \equiv \exists^{\mathbb{N}} y_0 [Z(x, y_0, \vec{z}) \wedge A(x, Z(x, y_0)) \wedge \forall^{\mathbb{N}} y (A(x, Z(x, y)) \Rightarrow y_0 \leq y)]$$

Intuitively, the k -ary predicate $\vec{z} \mapsto U_Z(x, \vec{z})$ is extensionally equivalent to the predicate $\vec{z} \mapsto Z(x, y_0, \vec{z})$ where y_0 denotes the smallest natural number such that $A(x, Z(x, y_0))$. (From the minimum principle, we know that y_0 exists provided $A(x, Z(x, y))$ holds for some number y .) Combining the minimum principle with the property of extensionality established in Lemma. 7 we get

$$\exists Z \forall x (\exists Y A(x, Y) \Rightarrow A(x, U_Z(x))), \quad (3)$$

hence

$$\exists U \forall x (\exists Y A(x, Y) \Rightarrow A(x, U(x))) \quad (4)$$

(now considering U as a $(k + 1)$ -ary second-order variable).

8.3. Realizing the principle of countable selection. Let us now instantiate the framework presented in sections 2–7 to the particular set of states $\mathcal{S} = \mathbb{N}$ equipped with the usual ordering (a natural number being considered as stronger when larger). We assume that the relation of evaluation fulfils the requirements of 2.2, with the following extra conditions:

- (1) Evaluation is deterministic.
- (2) No instruction can decrease the current value stored in the state, in the sense that $t \star \pi \star n \succ t' \star \pi' \star n'$ implies $n' \geq n$.
- (3) There is a special instruction χ such that for all $t \in \Lambda$, $\pi \in \Pi$ and $n \in \mathbb{N}$:

$$\chi \star t \cdot \pi \star n \succ^* t \star \bar{n} \cdot \pi \star n'$$

for some $n' > n$.

(note that we do not even have to assume that the evaluation of χ is atomic). There are several ways to implement these conditions:

The counter: No evaluation step (including abstraction, application, etc.) changes the current value of the state, but the evaluation of the instruction χ , that increments this value:

$$\chi \star t \cdot \pi \star n \succ t \star \bar{n} \cdot \pi \star (n + 1).$$

In this case, the value stored in the state can be understood as a counter whose value is read and incremented by the instruction χ .

The clock: Every evaluation step increments the current value of the state, that is: $t \star \pi \star n \succ t' \star \pi' \star n'$ implies $n' = n + 1$. In this case, the value stored in the state can be understood as the current time, whose value is read (and incremented) by the instruction χ .

(Of course, it is possible to imagine several variants of this design.)

An important consequence of the conditions given above is that within a given thread, the value of the current state determines the stack that faces the instruction χ each time it comes in head position, namely:

Fact 1. — *If $\chi \star \pi \star n \in \mathbf{thd}(p_0)$ and $\chi \star \pi' \star n \in \mathbf{thd}(p_0)$, then $\pi = \pi'$.*

Proposition 8. — *Any instruction χ that fulfils the conditions above locally realizes the principle of countable selection (i.e. χ realizes the principle of countable selection w.r.t. every local pole \perp).*

Proof. Let \perp be a pole such that $\mathbf{C}\perp \subseteq \mathbf{thd}(p_0)$ for some process p_0 . For every natural number n , let us write

$$P_n = \{(\pi, m) \in \Pi \times \mathbb{N} : m \leq n \wedge \exists t \in \Lambda (\chi \star t \cdot \pi \star n \notin \perp)\}.$$

From Fact 1 combined with the hypothesis that the pole \perp is local, the assertions $\chi \star t \cdot \pi \star n \notin \perp$ and $\chi \star t' \cdot \pi' \star n \notin \perp$ imply $t = t'$ and $\pi = \pi'$. Consequently, the first projection of P_n contains at most one stack π . Let us now consider a parametric formula $A(Y)$ that only depends on a k -ary second-order variable Y . For every $n \in \mathbb{N}$, let us choose $\Phi(n) : \mathbb{N}^k \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})$ as follows

- $\Phi(n)$ is taken as some falsity value function $F : \mathbb{N}^k \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})$ such that $\|A(\dot{F})\|$ intersects P_n , if there exists such a function.
- $\Phi(n) : \mathbb{N}^k \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})$ is any falsity value function otherwise.

We have thus defined a function $\Phi : \mathbb{N}^{1+k} \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})$. To show that χ realizes the formula $\forall^{\mathbb{N}} y A(\dot{\Phi}(y)) \Rightarrow \forall Y A(Y)$, let us consider a state $n_1 \in \mathbb{N}$, an arbitrary element of $\|\forall^{\mathbb{N}} y A(\dot{\Phi}(y)) \Rightarrow \forall Y A(Y)\|$ —that is, a stack of the form $(t \cdot \pi, n_2)$ where $(t, n_2) \in \|\forall^{\mathbb{N}} y A(\dot{\Phi}(y))\|$ and $(\pi, n_3) \in \|A(\dot{F})\|$ for some $F : \mathbb{N}^k \rightarrow \mathfrak{P}(\Pi \times \mathcal{S})$ and for some $n_3 \leq n_2$ —and take a state n such that $n \geq n_1$ and $n \geq n_2$. To conclude, it suffices to show that $\chi \star t \cdot \pi \star n \in \perp$. For that, let us reason by contradiction and assume that $\chi \star t \cdot \pi \star n \notin \perp$. By definition of P_n we have $(\pi, n_3) \in \|A(\dot{F})\| \cap P_n$, so that $\|A(\dot{\Phi}(n))\| \cap P_n$ is not empty (by construction of Φ). Let (π, n_4) be an element of $\|A(\dot{\Phi}(n))\| \cap P_n$ (using the remark above about the uniqueness of π), with $n_4 \leq n$. We have $(\bar{n} \cdot \pi, n_4) \in \|\{n\} \Rightarrow \dot{\Phi}(n)\| \subseteq \|\forall^{\mathbb{N}} y \dot{\Phi}(y)\|$, hence $t \star \bar{n} \cdot \pi \star (n + 1) \in \perp$ (since $n + 1 \geq n_4$ and $n + 1 \geq n_2$). By anti-reduction we get $\chi \star t \cdot \pi \star n \in \perp$, hence the assumption $\chi \star t \cdot \pi \star n \notin \perp$ is absurd. \square

Remark. This example seems to indicate that generalized realizability is more expressive than forcing when considering the same set of conditions. Indeed, in ordinary forcing with the set of conditions (\mathbb{N}, \geq) , the induced complete boolean algebra is trivial: $\mathbb{B}(\mathbb{N}) = \{\emptyset; \mathbb{N}\} \approx \{0; 1\}$ (due to the fact that natural numbers are pairwise compatible). Consequently, the corresponding model extension adds nothing to the initial 2-valued model.

REFERENCES

- [1] J.-L. Krivine. Dependent choice, ‘quote’ and the clock. *Th. Comp. Sc.*, 308:259–276, 2003.
- [2] J.-L. Krivine. Realizability in classical logic. Unpublished lecture notes (available on the author’s web page), 2005.
- [3] J.-L. Krivine. Structures de réalisabilité, RAM et ultrafiltre sur \mathbb{N} . Manuscript, available on the author’s web page, 2008.