

# Classical modal realizability and side effects

Alexandre Miquel  
U. Paris 7 (PPS) & ENS Lyon (LIP)  
46 Allée d’Italie, 69364 Lyon, France  
Alexandre.Miquel@ens-lyon.fr

## Abstract

*The proofs-as-programs paradigm—a.k.a. the Curry-Howard correspondence—is currently limited to functional programming with control. We propose here a way to extend it to imperative programming (yet in the call-by-name discipline) in connection with modal logic, using Kripke-style worlds to represent any concrete or abstract notion of state, such as machine stores or logic invariants.*

*For that, we present an extension of classical realizability to second-order modal logic by introducing a combined notion of Kripke-Krivine realizability model. We define an adequate type system for this realizability model, and show that the axioms of standard presentations of modal logics (S4, S5, etc.) are realized if and only if the underlying Kripke structure fulfils the expected frame condition.*

*As an application, we show how to type the access and assignment operations through a store using specific modal connectives.*

## 1. Introduction

For a long time the proofs-as-programs paradigm—a.k.a. the Curry-Howard correspondence—has been limited to intuitionistic logic and pure functional programming. In the 90’s, the correspondence has been extended to classical logic by the discovery of a possible interpretation of classical principles in terms of control primitives such as call/cc. Several deterministic and non-deterministic typed  $\lambda$ -calculi have been proposed to express the Curry-Howard correspondence extended to classical logic.

A particularly expressive framework to express the correspondence between programs and classical proofs is the classical realizability model introduced by Krivine [8]. This model provides realizers for all classical principles as well as for several forms of the axiom of choice [7, 8], and it has been recently extended to the calculus of constructions [11].

In this paper, we propose an extension of classical realizability to modal logic, by introducing a combined Kripke-

Krivine realizability model where the set of realizers of a given formula depends on a world  $w \in \mathcal{W}$ , writing  $\mathcal{W}$  an arbitrary nonempty set of ‘possible worlds’.

This combined framework does not only contain Krivine’s usual realizability model (taking a singleton as  $\mathcal{W}$ ) and the usual Kripke models (taking the empty set as a pole  $\perp$ ) as degenerated cases, but it also suggests a connection between Kripke’s worlds and side-effects through the Curry-Howard correspondence, as we shall see with the example of Section 7. The most noticeable feature of our interpretation of modal logic is that realizers do not content themselves to retrieve computational information from the current world, but they may also modify it.

Many connections have been investigated between realizability and Kripke semantics (for instance in [10] or [1]) on one hand, and between modal logic and side effects (for instance in [5]) on the other hand. But as far as we know, the combined realizability model presented here is the first one to provide realizers (written in direct style) for all provable formulæ of well-known systems such as S4 or S5 [6].

## 2. The calculus of realizers

### 2.1. Terms and stacks

We consider a denumerable set  $\mathcal{X}$  of (*term*) *variables* (notation:  $x, y, z$ , etc.) The definition of terms and stacks is parameterized by a set  $\mathcal{K}$  of *instructions* (notation:  $\kappa, \kappa'$ , etc.) containing at least an instruction written  $\mathfrak{c}$  (‘call-cc’). *Terms* (notation:  $t, u$ , etc.) and *stacks* (notation:  $\pi, \pi'$ , etc.) are defined by mutual induction as follows:

**Terms**  $t, u ::= x \mid \lambda x. t \mid tu \mid \kappa \mid k_\pi$

**Stacks**  $\pi ::= \diamond \mid t \cdot \pi \quad (t \text{ closed})$

Terms of the  $\lambda_c$ -calculus are the usual  $\lambda$ -terms enriched with constants that consist of all instructions  $\kappa \in \mathcal{K}$  plus a *continuation constant*  $k_\pi$  for every stack  $\pi$ .

Stacks are lists of *closed* terms.<sup>1</sup> Unlike terms (that may be open), stacks are closed objects—so that the continuation  $k_\pi$  associated to every stack  $\pi$  is really a constant.

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

## 2.2. Worlds and processes

The calculus is also parameterized by a nonempty set  $\mathscr{W}$  of *worlds* (notation:  $w, w', \text{etc.}$ )<sup>2</sup> We call a *process* (notation:  $p, q, \text{etc.}$ ) any triple written  $t \star \pi \star w$  formed by a closed term  $t$ , a stack  $\pi$  and a world  $w$ :

**Processes**  $p, q ::= t \star \pi \star w \quad (t \text{ closed})$

The set of all processes is written  $\Lambda \star \Pi \star \mathscr{W}$ .

We assume given a binary relation  $p \succ p'$  of *evaluation* that satisfies (at least) the following axioms:

$$\begin{aligned} \lambda x. t \star u \cdot \pi \star w &\succ t\{x := u\} \star \pi \star w \\ tu \star \pi \star w &\succ t \star u \cdot \pi \star w \\ \mathfrak{c} \star t \cdot \pi \star w &\succ t \star k_\pi \cdot \pi \star w \\ k_\pi \star t \cdot \pi' \star w &\succ t \star \pi \star w \end{aligned}$$

(for all  $t, u, \pi, \pi', w$ ). Although the evaluation rules given above do not interact with worlds, the evaluation relation may contain other rules (for extra instructions) that depend on the current world (e.g. a read operation) or that may even change the current world (e.g. a write operation).

Finally, we say that a set of processes  $S \subseteq \Lambda \star \Pi \star \mathscr{W}$  is *saturated* when it is closed under anti-evaluation, that is:  $p \succ p'$  and  $p' \in S$  imply  $p \in S$  for all processes  $p, p'$ .

## 3. The realizability model

### 3.1. The language of 2nd-order modal logic

We consider a fixed first-order term algebra of *expressions* (notation:  $e, e', \text{etc.}$ ) formed from a given first-order signature. First-order variables (that shall not be confused with term variables) are also denoted by  $x, y, z$ .<sup>3</sup>

An example is the algebra of *arithmetic expressions* that are formed from a constant symbol 0 ('zero'), a unary function symbol  $s$  ('successor'), two binary function symbols  $+$

<sup>1</sup>In [8], infinitely many stack constants are introduced to give the possibility of hiding extra information in the bottom of stacks. This is no more necessary here, since worlds can now carry on such an extra information.

<sup>2</sup>At first reading, it is convenient to think of worlds as some kind of machine states (typically: a store) although worlds may contain much richer contents (typically: logical invariants).

<sup>3</sup>There should be no ambiguity between term variables and first-order variables, since terms and expressions do not occur in the same contexts.

(‘addition’) and  $\times$  (‘multiplication’), and more generally function symbols for all primitive recursive functions.

We consider an infinite set of *second-order variables* (a.k.a. *predicate variables*) for every arity  $k \geq 0$  (notation:  $X, Y, Z, \text{etc.}$ ) Formulæ (notation:  $A, B, C, \text{etc.}$ ) of (minimal) second-order modal logic are defined by

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

Here, the necessity connective  $\Box_R A$  is indexed by a binary relation  $R \subseteq \mathscr{W} \times \mathscr{W}$ . We assume that  $R$  ranges over a fixed set of relations  $\mathfrak{R} \subseteq \mathfrak{P}(\mathscr{W} \times \mathscr{W})$ , thus allowing several notions of necessity to be expressed in the language.

We write  $FV(A)$  the set of free (1st- and 2nd-order) variables of the formula  $A$ , and both notions of substitution  $A\{x := e\}$  (1st-order) and  $A\{X(x_1, \dots, x_k) := B\}$  (2nd-order) are defined as expected [3, 8].

**Second-order encodings** Absurdity, negation, conjunction and disjunction are defined using 2nd-order encodings

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

as well as 1st- and 2nd-order existential quantification

$$\begin{aligned} \exists x A(x) &\equiv \forall Z (\forall x (A(x) \Rightarrow Z) \Rightarrow Z) \\ \exists X A(X) &\equiv \forall Z (\forall X (A(X) \Rightarrow Z) \Rightarrow Z) \end{aligned}$$

and Leibniz equality:

$$e = e' \equiv \forall Z (Z(e) \Rightarrow Z(e')).$$

The connective  $\Diamond_R A$  of possibility is then defined dually to the connective of necessity  $\Box_R A$  by letting

$$\Diamond_R A \equiv \neg \Box_R \neg A.$$

### 3.2. Truth values and falsity values

A (*simple*) *falsity value* is a set of stacks  $F_0 \in \mathfrak{P}(\Pi)$ , and a *modal falsity value* is a set of pairs  $F \in \mathfrak{P}(\Pi \times \mathscr{W})$ . Equivalently, a modal falsity value  $F \in \mathfrak{P}(\Pi \times \mathscr{W})$  can be considered as a function  $F : \mathscr{W} \rightarrow \mathfrak{P}(\Pi)$  that assigns a simple falsity value to every world, and given a modal falsity value  $F \in \mathfrak{P}(\Pi \times \mathscr{W})$  and a world  $w \in \mathscr{W}$  we shall write  $F_w = \{\pi \in \Pi : (\pi, w) \in F\}$ .

Symmetrically, a (*simple*) *truth value* is a set of closed terms  $T_0 \in \mathfrak{P}(\Lambda)$  whereas a *modal truth value* is a set of pairs  $T \in \mathfrak{P}(\Lambda \times \mathscr{W})$ . As for falsity values, we identify modal truth values with functions  $T : \mathscr{W} \rightarrow \mathfrak{P}(\Lambda)$ .

In what follows we will often define (simple and modal) truth values from (simple and modal) falsity values by orthogonality w.r.t. a given set of processes  $\perp \subseteq \Lambda \star \Pi \star \mathscr{W}$  (the ‘pole’) as follows:

- Given a simple falsity value  $F_0 \in \mathfrak{P}(\Pi)$  and a world  $w \in \mathscr{W}$  we denote by  $F_0^{(\perp_w)} \in \mathfrak{P}(\Lambda)$  the simple truth value defined by

$$\begin{aligned} F_0^{(\perp_w)} &= \{t \in \Lambda : \forall \pi \in F_0 \ (t \star \pi \star w) \in \perp\} \\ &= \{t \in \Lambda : \forall \pi \in F_0 \ (t, \pi) \in \perp_w\} \end{aligned}$$

writing  $\perp_w = \{(t, \pi) : t \star \pi \star w \in \perp\}$ .

- Given a modal falsity value  $F \in \mathfrak{P}(\Pi \times \mathscr{W})$  we denote by  $F^\perp$  the modal truth value defined by  $(F^\perp)_w = (F_w)^{(\perp_w)}$  for all worlds  $w \in \mathscr{W}$ .

Finally, we say that a set of processes  $\perp \subseteq \Lambda \star \Pi \star \mathscr{W}$  is *monotonic* w.r.t. a binary relation  $R \subseteq \mathscr{W} \times \mathscr{W}$  when  $(w, w') \in R$  implies  $\perp_w \subseteq \perp_{w'}$  for all  $w, w' \in \mathscr{W}$ . We then say that  $\perp$  is  $\mathfrak{R}$ -*monotonic* when  $\perp$  is monotonic w.r.t. all relations  $R \in \mathfrak{R}$ . In what follows, we shall require that the pole  $\perp$  is both saturated and  $\mathfrak{R}$ -monotonic.

### 3.3. Enriching the language

We now assume that closed expressions are interpreted as elements of a fixed nonempty set  $\mathcal{D}$ , writing  $\llbracket e \rrbracket \in \mathcal{D}$  the denotation of a closed expression  $e$ . We then enrich the language of expressions with a constant symbol  $\dot{d}$  for every element  $d \in \mathcal{D}$ , letting  $\llbracket \dot{d} \rrbracket = d$ :

**Expressions**  $e ::= \dots \mid \dot{d} \quad (d \in \mathcal{D})$

Similarly, we enrich the language of formulae with a predicate symbol  $\dot{F}$  of arity  $k \geq 0$  for every modal falsity function  $F : \mathcal{D}^k \rightarrow \mathfrak{P}(\Pi \star \mathscr{W})$ :

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

(Note that unlike 1st-order variables, the interpretation of 2nd-order variables depends on the world. We shall consider an alternative approach in Section 6.)

**Valuations** We call a *valuation* any function  $\rho$  that maps every first-order variable  $x$  to an element  $\rho(x) \in \mathcal{D}$  and every second-order variable  $X$  of arity  $k$  to a modal falsity function  $\rho(X) : \mathcal{D}^k \rightarrow \mathfrak{P}(\Pi \times \mathscr{W})$ .

Given an open expression  $e$  (resp. an open formula  $A$ ) of the enriched language and a valuation  $\rho$ , we write  $e[\rho]$  (resp.  $A[\rho]$ ) the closed expression (resp. the closed formula) of the enriched language obtained by replacing every free occurrence of a first-order variable  $x$  by the constant symbol  $\dot{\rho}(x)$  associated to the denotation  $\rho(x) \in \mathcal{D}$ , and by replacing every free occurrence of a second-order variable  $X$  of arity  $k$  by the predicate symbol  $\dot{\rho}(X)$  associated to the modal falsity function  $\rho(X) : \mathcal{D}^k \rightarrow \mathfrak{P}(\Pi \times \mathscr{W})$ .

### 3.4. Interpreting formulae

Given a  $\mathfrak{R}$ -monotonic saturated set  $\perp \subseteq \Lambda \star \Pi \star \mathscr{W}$ , we interpret every closed formula  $A$  of the enriched language of formulae as two sets: a modal truth value written  $|A| \in \mathfrak{P}(\Lambda \times \mathscr{W})$  and a modal falsity value written  $\|A\| \in \mathfrak{P}(\Pi \times \mathscr{W})$  that are defined by induction on the size of the formula  $A$  using the equations

$$\begin{aligned} \|\dot{F}(e_1, \dots, e_n)\|_w &= F(\llbracket e_1 \rrbracket, \dots, \llbracket e_n \rrbracket)_w \\ \|A \Rightarrow B\|_w &= |A|_w \cdot \|B\|_w \\ &= \{t \cdot \pi : t \in |A|_w, \pi \in \|B\|_w\} \\ \|\Box_R A\|_w &= \bigcup_{w' \in R\langle w \rangle} \|A\|_{w'} \\ \|\forall x A\|_w &= \bigcup_{d \in \mathcal{D}} \|A\{x := \dot{d}\}\|_w \\ \|\forall X A\|_w &= \bigcup_{F : \mathcal{D}^k \rightarrow \mathfrak{P}(\Pi \star \mathscr{W})} \|A\{X := \dot{F}\}\|_w \\ |A|_w &= \|A\|_w^{(\perp_w)} \end{aligned}$$

writing  $R\langle w \rangle = \{w' : (w, w') \in R\}$ . Note that by orthogonality we have

$$|\forall x A|_w = \bigcap_{d \in \mathcal{D}} |A\{x := \dot{d}\}|_w$$

(and similarly for 2nd-order quantification) while

$$|\Box_R A|_w = \bigcap_{w' \in R\langle w \rangle} (\|A\|_{w'})^{\perp_w} \subseteq \bigcap_{w' \in R\langle w \rangle} |A|_{w'}$$

(using the fact that  $(w, w') \in R$  implies  $\perp_w \subseteq \perp_{w'}$  and thus  $(\|A\|_{w'})^{\perp_w} \subseteq (\|A\|_{w'})^{\perp_{w'}} = |A|_{w'}$ ). In general, the set  $|\Box_R A|_w$  is thus *smaller* than the intersection of all the sets  $|A|_{w'}$  when  $w'$  ranges over  $R\langle w \rangle$ .

Since the truth value  $|A|$  and the falsity value  $\|A\|$  actually depend on the parameter  $\perp$ , we shall sometimes use the notations  $|A|_\perp$  and  $\|A\|_\perp$  to indicate this dependency explicitly. In what follows, we shall write

- $t \Vdash_w A$  (' $t$  realizes  $A$  in world  $w$ ') when  $t \in |A|_w$ ;
- $t \Vdash A$  (' $t$  uniformly realizes  $A$ ') when  $t \Vdash_w A$  for all worlds  $w \in \mathscr{W}$ ;
- $t \Vdash\!\!\! \Vdash A$  (' $t$  is a universal realizer of  $A$ ') when  $t \Vdash A$  for all  $\mathfrak{R}$ -monotonic saturated sets  $\perp \subseteq \Lambda \star \Pi \star \mathscr{W}$ .

(Note that only the first two notions depend on the choice of the pole  $\perp \subseteq \Lambda \star \Pi \star \mathscr{W}$ .)

### 3.5. Why call/cc does not save the world

The reader may have noticed that the call/cc instruction only saves the current stack  $\pi$ , but not the current world. Dually, continuation constants  $k_\pi$  only restore previously saved stacks without affecting the current world:

$$\begin{array}{lcl} \alpha \star t \cdot \pi \star w & \succ & t \star k_\pi \cdot \pi \star w \\ k_\pi \star t \cdot \pi' \star w & \succ & t \star \pi \star w \end{array}$$

This design choice that corresponds to the usual way of implementing call/cc in imperative languages is justified by the fact that call/cc still realizes Peirce's law in the extended framework of modal realizability:

**Lemma 1** — *Let  $A$  and  $B$  be closed formulae of the enriched language. For all  $\pi \in \Pi$  and  $w \in \mathscr{W}$ :*

1. *If  $\pi \in \|A\|_w$ , then  $k_\pi \Vdash_w A \Rightarrow B$  (negation of  $A$ )*
2.  $\alpha \Vdash_w ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$  (Peirce's law)

On the other hand, it is a simple exercise to check that a stronger version of call/cc that would produce modified continuation constants  $k_{(\pi,w)}$  encapsulating—and later restoring—both the current stack and the current world would simply not have the desired type in the model.

### 3.6. Kripke semantics as a degenerated case

Besides the construction depicted above, the language of formulae given in 3.1 has a simple Kripke-style semantics where individuals are interpreted as elements of  $\mathcal{D}$  and predicate variables of arity  $k \geq 0$  as functions  $f : \mathcal{D}^k \rightarrow \mathfrak{P}(\mathscr{W})$ .

Formally we define a binary relation  $w \models A$  between worlds and closed formulae. As usual, open formulae are closed using an assignment  $\varphi$  that maps every 1st-order variable  $x$  to an element  $\varphi(x) \in \mathcal{D}$  and every 2nd-order variable  $X$  of arity  $k \geq 0$  to a function  $\varphi(X) : \mathcal{D}^k \rightarrow \mathfrak{P}(\mathscr{W})$ . The relation  $w \models A[\varphi]$  is inductively defined as follows:

- $w \models X(e_1, \dots, e_n)[\varphi]$  if  $w \in \varphi(X)(\llbracket e_1[\varphi] \rrbracket, \dots, \llbracket e_n[\varphi] \rrbracket)$ ;
- $w \models (A \Rightarrow B)[\varphi]$  if  $w \models A[\varphi]$  implies  $w \models B[\varphi]$ ;
- $w \models \Box_R A[\varphi]$  if for all  $w' \in R(w)$ ,  $w' \models A[\varphi]$ ;
- $w \models (\forall x A)[\varphi]$  if for all  $d \in \mathcal{D}$ ,  $w \models A[\varphi, x \leftarrow d]$ ;
- $w \models (\forall X A)[\varphi]$  if for all  $f : \mathcal{D}^k \rightarrow \mathfrak{P}(\mathscr{W})$ ,  $w \models A[\varphi, X \leftarrow f]$ .

This interpretation extends to the language enriched with predicate symbols  $\dot{F}$  associated to all modal falsity functions  $F : \mathcal{D}^k \rightarrow \mathfrak{P}(\Pi \times \mathscr{W})$ , letting

- $w \models \dot{F}(e_1, \dots, e_n)[\varphi]$  if  $F(\llbracket e_1[\varphi] \rrbracket, \dots, \llbracket e_n[\varphi] \rrbracket)_w = \emptyset$ .

Given a closed formula  $A$  and a world  $w \in \mathscr{W}$  we write  $w \models A$  for  $w \models A[\varphi]$  where  $\varphi$  is an arbitrary assignment. (The definition does not depend on  $\varphi$ .)

It is straightforward to check that this simple Kripke-style semantics is mimicked by the realizability model in the particular case where the pole  $\perp$  is empty:

**Fact 1** — *If  $\perp = \emptyset$ , then for every closed formula  $A$  of the enriched language and for every  $w \in \mathscr{W}$ :*

1.  $\|A\|_w = \emptyset$  iff  $w \models A$ ;
2.  $|A|_w = \Lambda$  iff  $w \models A$ , and  $|A|_w = \emptyset$  otherwise.

Therefore, if a closed formula  $A$  has a universal realizer  $t \Vdash A$ , then  $A$  is true in the underlying Kripke model, in the sense that  $w \models A$  in all worlds  $w \in \mathscr{W}$ .

## 4 Typing

### 4.1 The judgments $A \leq B$ and $\Gamma \vdash t : A$

To facilitate the construction of realizers, we introduce a type system based on two judgments, namely:

- A subtyping judgment written  $A \leq B$  (' $A$  is a subtype of  $B$ '), where  $A$  and  $B$  are open formulae of the enriched language. We write  $A \sim B$  to express that  $A \leq B$  and  $B \leq A$  hold simultaneously.
- A typing judgment written  $\Gamma \vdash t : A$  ('in context  $\Gamma$ , the term  $t$  has type  $A$ '), where  $A$  is an open formula of the enriched language and  $\Gamma$  a typing context mapping finitely many term-variables to open formulae.

In what follows, we write  $\text{dom}(\Gamma)$  the domain of a context  $\Gamma$ , and given two contexts  $\Gamma$  and  $\Gamma'$  that coincide on the intersection of their domains we write  $\Gamma, \Gamma'$  their union.

The inference rules of both judgments are given in Fig. 1.

Typing rules contain the usual rules of minimal propositional logic (axiom, weakening, introduction and elimination of implication) as well as introduction rules for 1st- and 2nd-order quantification (the corresponding eliminations are performed via subtyping and subsumption). They also comprise a subsumption rule, a typing rule for call/cc (Peirce's law) plus a *necessitation typing rule*, the only typing rule directly involving a modal connective.

### Typing rules

$$\frac{}{x : A \vdash x : A} \quad \frac{\Gamma \vdash t : B}{\Gamma, x : A \vdash t : B}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x. t : A \Rightarrow B} \quad \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma' \vdash u : A}{\Gamma, \Gamma' \vdash tu : B}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall x A} \quad x \notin FV(\Gamma) \quad \frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall X A} \quad X \notin FV(\Gamma)$$

$$\Gamma \vdash \mathbf{c} : ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$$

$$\frac{\Gamma \vdash t : A}{\Box_R \Gamma \vdash t : \Box_R A} \quad \frac{\Gamma \vdash t : A \quad A \leq A'}{\Gamma \vdash t : A'}$$

### Subtyping rules

$$\frac{}{A \leq A} \quad \frac{A \leq B \quad B \leq C}{A \leq C}$$

$$\frac{A' \leq A \quad B \leq B'}{A \Rightarrow B \leq A' \Rightarrow B'} \quad \frac{A \leq A'}{\Box_R A \leq \Box_R A'}$$

$$\frac{A \leq B}{A \leq \forall x B} \quad x \notin FV(A) \quad \frac{A \leq B}{A \leq \forall X B} \quad X \notin FV(A)$$

$$\overline{\forall x A \leq A\{x := e\}} \quad \overline{\forall X A \leq A\{X(x_1, \dots, x_k) := B\}}$$

$$\overline{\forall x (A \Rightarrow B) \leq \forall x A \Rightarrow \forall x B}$$

$$\overline{\forall X (A \Rightarrow B) \leq \forall X A \Rightarrow \forall X B}$$

$$\overline{\Box_R (A \Rightarrow B) \leq \Box_R A \Rightarrow \Box_R B} \quad \overline{\Box_R A \leq \Box_S A} \quad S \subseteq R$$

$$\overline{\Box_{id} A \sim A} \quad \overline{\Box_{R,S} A \sim \Box_R \Box_S A}$$

$$\overline{\forall x \Box_R A \leq \Box_R \forall x A} \quad \overline{\forall X \Box_R A \leq \Box_R \forall X A}$$

Figure 1. Typing and subtyping rules

Subtyping rules contain administrative rules (reflexivity, transitivity and propagation rules across implication and necessity) as well as the (subtyping) introduction and elimination rules for 1st- and 2nd-order quantification. Then comes the *distributivity subtyping rule* followed by administrative rules describing how different necessity connectives combine (inclusion, identity, composition). These rules end

with 1st- and 2nd-order *Barcan subtyping rules* implementing the corresponding Barcan rules. (Note that the converse inclusions can be derived from the preceding rules.)

## 4.2 Interpreting judgments in the model

We call a *substitution* any finite association list of the form  $\sigma = [x_1 := u_1; \dots; x_n := u_n]$ , where  $x_1, \dots, x_n$  are pairwise distinct term-variables and where  $u_1, \dots, u_n$  are closed terms. The domain of  $\sigma$  is written  $\text{dom}(\sigma)$ , and for all terms  $t$  we write  $t[\sigma] = t\{x_1 := u_1; \dots; x_n := u_n\}$ . Given a substitution  $\sigma$ , a world  $w \in \mathscr{W}$  and a context  $\Gamma$  of closed formulæ, we write  $\sigma \Vdash_w \Gamma$  when  $\text{dom}(\sigma) \supseteq \text{dom}(\Gamma)$  and  $\sigma(x) \Vdash_w \Gamma(x)$  for every  $x \in \text{dom}(\Gamma)$ .

With these notations, both judgments  $A \leq B$  and  $\Gamma \vdash t : A$  are interpreted in the realizability model as follows: given a saturated set  $\perp$  we say that

- The subtyping judgment  $A \leq B$  is *valid* when for all valuations  $\rho$  and for all  $w \in \mathscr{W}$  we have  $\|A[\rho]\|_w \supseteq \|B[\rho]\|_w$  (reverse inclusion of falsity values). Which immediately implies that  $\|A[\rho]\|_w \subseteq \|B[\rho]\|_w$  (direct inclusion of truth values), but is not equivalent to.
- The typing judgment  $\Gamma \vdash t : A$  is *valid* when for all valuations  $\rho$ , for all worlds  $w \in \mathscr{W}$  and for all substitutions  $\sigma \Vdash_w \Gamma[\rho]$  we have  $t[\sigma] \Vdash_w A[\rho]$ .

We finally extend the notion of validity to inference rules, saying that the rule

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

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

**Proposition 1 (Adequacy)** — *The inference rules of Fig. 1 are valid for all saturated sets  $\perp$ .*

*Proof.* The validity of the rules that do not involve modalities is shown the same way as in ordinary classical realizability, whereas the validity of both the necessitation typing rule and the distributivity subtyping rule strongly relies on the condition of  $\mathfrak{R}$ -monotonicity. Validity of remaining subtyping rules is straightforward.  $\square$

**Example.** Using the typing rules of the simply typed  $\lambda$ -calculus we easily derive

$$f : A \Rightarrow B, y : \neg B \vdash \lambda z. y(f z) : \neg A$$

so that from the necessitation typing rule we get

$$f : \Box(A \Rightarrow B), y : \Box \neg B \vdash \lambda z. y(f z) : \Box \neg A.$$

From the latter we easily derive the following judgment

$$\vdash \lambda fxy . x (\lambda z . y (f z)) : \Box(A \Rightarrow B) \Rightarrow \Diamond A \Rightarrow \Diamond B$$

(writing  $\Diamond A \equiv \neg\Box\neg A$ ) expressing that the co-distributivity formula  $\Box(A \Rightarrow B) \Rightarrow \Diamond A \Rightarrow \Diamond B$  can be ‘proved’ with the term  $\lambda fxy . x (\lambda z . y (f z))$  in our system.

## 5 Logical interpretation

In this section, we fix a binary relation  $R \in \mathfrak{R}$  and write  $\Box A = \Box_R A$ . Dually, we write  $\Diamond A = \neg\Box_R\neg A$ .

The minimal system of modal logic—system **K**—is implemented in the type system described in Fig. 1 via the necessitation and distributivity rules:

$$\frac{\Gamma \vdash t : A}{\Box\Gamma \vdash t : \Box A} \quad \frac{}{\Box(A \Rightarrow B) \leq \Box A \Rightarrow \Box B}$$

In particular, every uniform realizer of  $A$  is also a uniform realizer of  $\Box A$  (necessitation), while the distributivity axiom is uniformly realized by the identity term  $\lambda z . z$ .

Depending on the properties of the relation  $R$ , we can realize the axioms of many systems of modal logic.

### 5.1 Interpreting S4 and S5

From the modality subtyping rules (inclusion, identity and composition) of Fig. 1 we immediately check that:

1. If  $R$  is reflexive (i.e.  $\text{id} \subseteq R$ ), then the subtyping judgment  $\Box A \leq A$  is derivable, and thus valid.
2. If  $R$  is transitive (i.e.  $R; R \subseteq R$ ), then the subtyping judgment  $\Box A \leq \Box\Box A$  is derivable, and thus valid.

This means that when  $R$  is a preorder, we can build universal realizers for all theorems of S4, realizing both axioms **T** and **4** by the identity term  $\lambda z . z$ .

To extend this result to S5, we first check that:

**Fact 2** — *If  $R$  is symmetric, then the subtyping judgment  $A \Rightarrow \Box B \leq \Box(\Box A \Rightarrow B)$  is valid.*

From this we get  $\neg\neg A \leq \neg A \Rightarrow \Box\perp \leq \Box(\Box\neg A \Rightarrow \perp)$ , that is:  $\neg\neg A \leq \Box\Diamond A$ , hence

**Fact 3** — *If  $R$  is symmetric, then axiom (B):  $A \Rightarrow \Box\Diamond A$  is universally realized by the term  $\lambda xy . yx$ .*

Which means that when  $R$  is an equivalence relation, we can build universal realizers for all theorems of S5.

## 5.2. Interpreting other axioms

More generally, many other conditions on the relation  $R$  (seriality, Euclidianness, etc.) are captured by a particular axiom of modal logic [6]. For some of them, we can extend the result to realizability by providing both a realizer for the axiom and a subtyping rule capturing the condition:

**Proposition 2** — *For every line of Table 1, the following assertions are equivalent:*

1. *The relation  $R$  fulfils the condition of column 1.*
2. *The axiom of column 2 is valid (for all  $A, B$ , etc.) in the underlying Kripke model (cf 3.6).*
3. *The term of column 3 is a universal realizer of the axiom of column 2.*
4. *The subtyping rule of column 4 is valid.*

*Proof.* The equivalence  $1 \Leftrightarrow 2$  is standard [6]. For  $1 \Rightarrow 4$ : check that the condition of column 1 implies the desired (reverse) inclusion of falsity values. For  $4 \Rightarrow 3$ : typecheck the realizer of column 3 using the subtyping rule of column 4. For  $3 \Rightarrow 2$ : set  $\perp = \emptyset$  (cf 3.6).  $\square$

### 5.3. Embedding intuitionistic logic into S4

It is well-known [6] that intuitionistic logic (LJ) can be embedded into S4 using a translation  $A \mapsto A^\Box$  mapping provable formulæ of LJ into provable formulæ of S4 in a conservative way (at least in the propositional case). We easily adapt this translation to our framework as follows, assuming that the relation  $R$  is a preorder so that all the axioms of S4 are derivable (cf 5.1).

**The syntactic translation**  $A \mapsto A^\Box$  To every formula  $A$  of the language of second-order minimal logic (i.e. without the connective  $\Box$ ) we associate a formula  $A^\Box$  of second-order modal logic defined by

$$\begin{aligned} (X(e_1, \dots, e_n))^\Box &= \Box X(e_1, \dots, e_n) \\ (A \Rightarrow B)^\Box &= \Box(A^\Box \Rightarrow B^\Box) \\ (\forall x A)^\Box &= \forall x A^\Box \\ (\forall X A)^\Box &= \forall X A^\Box \end{aligned}$$

Note that we take advantage of Barcan laws by not putting a box on the top of 1st- and 2nd-order quantifications.

This translation is extended to the formulæ of the enriched language (i.e. parametric formulæ) by letting

$$(\dot{F}(e_1, \dots, e_n))^\Box = \Box\dot{F}(e_1, \dots, e_n).$$

**Proposition 3** — *For every formula  $A$  without modalities, the judgment  $A^\Box \leq \Box A^\Box$  is derivable.*

1. Condition on $R$	2. Axiom	3. Realizer	4. Subtyping rule
Reflexivity: $\forall w (w R w)$	$\Box A \Rightarrow A$ (T)	$\lambda x . x$	$\Box A \leq A$
Transitivity: $\forall w_1 w_2 w_3 (w_1 R w_2 \wedge w_2 R w_3 \Rightarrow w_1 R w_3)$	$\Box A \Rightarrow \Box \Box A$ (4)	$\lambda x . x$	$\Box A \leq \Box \Box A$
Seriality: $\forall w \exists w' (w R w')$	$\Box A \Rightarrow \Diamond A$ (D)	$\lambda xy . yx$	$\Box \perp \leq \perp$
Symmetry: $\forall w w' (w R w' \Rightarrow w' R w)$	$A \Rightarrow \Box \Diamond A$ (B)	$\lambda xy . yx$	$A \Rightarrow \Box B \leq \Box(\Box A \Rightarrow B)$
Euclidianness: $\forall w w_1 w_2 (w R w_1 \wedge w R w_2 \Rightarrow w_1 R w_2)$	$\Diamond A \Rightarrow \Box \Diamond A$ (5)	$\lambda x . x$	$\Box A \Rightarrow \Box B \leq \Box(\Box A \Rightarrow B)$

**Table 1. Frame conditions, axioms of modal logic and their realizers**

A consequence of the above fact is that the translation  $A \mapsto A^\square$  is substitutive in the sense that

**Proposition 4 (Substitutivity)** — *For all formulæ  $A, B$  without modalities:*

1.  $(A\{x := e\})^\square \equiv A^\square\{x := e\}$  (syntactic identity)
2.  $(A\{X(\vec{x}) := B\})^\square \sim A^\square\{X(\vec{x}) := B^\square\}$

We now define the intuitionistic restriction of the type system defined in Fig. 1, by removing all the typing and subtyping rule involving modalities as well as the typing rule of  $\alpha$  (Peirce’s law). Formally, we write  $A \leq_{\text{LJ}} B$  (resp.  $\Gamma \vdash_{\text{LJ}} t : A$ ) when the judgment  $A \leq B$  (resp.  $\Gamma \vdash t : A$ ) can be derived by only using the typing and subtyping rules of the intuitionistic fragment defined above.

**Proposition 5** 1. *If  $A \leq_{\text{LJ}} B$ , then  $A^\square \leq B^\square$ .*

2. *If  $\Gamma \vdash_{\text{LJ}} t : A$ , then  $\Gamma^\square \vdash t : A^\square$ .*

In other words, the translation  $A \mapsto A^\square$  extends to our type system, becoming the identity on proof-terms.

**Realizability interpretation** Consider a fixed pole  $\perp\!\!\!\perp$  (i.e. a  $\mathfrak{R}$ -monotonic saturated set of processes).

Given a closed formula  $A$  of the language of 2nd-order minimal logic (enriched with symbols  $\dot{d}$  and  $\dot{F}$ ) and a closed term  $t$  we write  $t \Vdash_{\text{LJ}} A$  for  $t \Vdash A^\square$  (that is:  $t \in |A^\square|_w$  for all  $w \in \mathscr{W}$ ). Given a substitution  $\sigma$  and a context  $\Gamma$  of closed formulæ we write  $\sigma \Vdash_{\text{LJ}} \Gamma$  for  $\sigma \Vdash \Gamma^\square$ .

Combining the property of adequacy (Prop. 1) with the results of subsection Prop. 5 we immediately get:

**Proposition 6 (Intuitionistic adequacy)** — *If  $\Gamma \vdash_{\text{LJ}} t : A$  then for all valuations  $\rho$  and for all substitutions  $\sigma$  such that  $\sigma \Vdash_{\text{LJ}} \Gamma[\rho]$  one has  $t[\sigma] \Vdash_{\text{LJ}} A$ .*

In the same way as S4-Kripke structures can be used to interpret LJ via the embedding  $A \mapsto A^\square$ , we can thus use the combined Kripke-Krivine realizability model to interpret intuitionistic provability, the classical features of the model being disabled by the embedding  $A \mapsto A^\square$ .

Note that the translation  $A \mapsto A^\square$  (and the corresponding realizability interpretation) is useful to decompose Cohen’s forcing in terms of Kripke forcing (cf Section 8).

## 6. The restricted 2nd-order quantification

The choice to interpret 2nd-order variables using *modal* falsity value functions allows us to instantiate 2nd-order variables by arbitrary predicates with the elimination rule

$$\forall X A \leq A\{X(x_1, \dots, x_k) := B\}$$

without any restriction on the formulæ  $A$  and  $B$ .

Another design choice is to restrict the interpretation of 2nd-order variables to simple falsity value functions. For that, we introduce (in addition to the already existing ‘full’ 2nd-order variables) a second set of 2nd-order variables of all arities which we call *restricted 2nd-order variables* (notation:  $\alpha, \beta, \gamma$ , etc.)

### 6.1. Extending the syntax

Formally, we extend the language of formulæ as follows:

**Formulæ**  $A, B ::= \dots \mid \alpha(e_1, \dots, e_k) \mid \forall \alpha A$

We extend the notation  $FV(A)$  to include restricted 2nd-order variables and define the corresponding form of 2nd-order substitution written  $A\{\alpha(x_1, \dots, x_k) := B\}$  in the expected way. Finally we introduce the shorthands

$$\begin{aligned} A\{X := \alpha\} &\equiv A\{X(x_1, \dots, x_k) := \alpha(x_1, \dots, x_k)\} \\ A\{\alpha := X\} &\equiv A\{\alpha(x_1, \dots, x_k) := X(x_1, \dots, x_k)\} \end{aligned}$$

to express the substitutions exchanging both forms of 2nd-order variables (with the same arity).

### 6.2. Extending realizability and typing

Given a simple falsity value function  $G : \mathcal{D}^k \rightarrow \mathfrak{P}(\Pi)$  we write  $\tilde{G} : \mathcal{D}^k \rightarrow \mathfrak{P}(\Pi \times \mathscr{W})$  its *modal extension* defined by  $\tilde{G}(d_1, \dots, d_k)_w = G(d_1, \dots, d_k)$  for all  $w \in \mathscr{W}$  and  $d_1, \dots, d_k \in \mathcal{D}$ . We then extend the realizability interpretation defined in 3.4 to the new quantifier, letting

$$\|\forall \alpha A\|_w = \bigcup_{G : \mathcal{D}^k \rightarrow \mathfrak{P}(\Pi)} \|A\{X := \dot{G}\}\|_w$$

$$\begin{array}{c}
\frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall \alpha A} \quad \alpha \notin FV(\Gamma) \quad \frac{A \leq B}{A \leq \forall \alpha B} \quad \alpha \notin FV(A)}{\frac{\overline{\forall \alpha A \leq A\{\alpha(x_1, \dots, x_k) := B\}} \quad (A \text{ amodal w.r.t. } \alpha)}}{\overline{\forall \alpha (A \Rightarrow B) \leq \forall \alpha A \Rightarrow \forall \alpha B} \quad \overline{\forall \alpha \Box_R A \leq \Box_R \forall \alpha A}}
\end{array}$$

**Figure 2. Typing and subtyping rules for  $\forall \alpha A$**

for all formulæ  $A$  of the extended language (enriched with predicate constants  $\dot{F}$  for all modal falsity value functions  $F : \mathcal{D}^k \rightarrow \mathfrak{P}(\Pi \times \mathscr{W})$ ) and for all worlds  $w \in \mathscr{W}$ .

The restricted form of 2nd-order quantification shares all its typing rules with the non restricted form such as depicted in Fig. 2, except the 2nd-order elimination subtyping rule.

**Restricted 2nd-order elimination** From our interpretation, it is clear that the 2nd-order elimination rule

$$\forall \alpha A \leq A\{\alpha(x_1, \dots, x_k) := B\}$$

cannot be proved valid without any restriction on  $A$  and  $B$ .

A naive way to achieve this goal would be to forbid full 2nd-order variables and modalities in the formula  $B$ . Alas, such a restriction would be incorrect: even if the formula  $B$  contains no modality and no full 2nd-order variable, its falsity value  $\|B\|_w$  may still depend on  $w$  for the reason that the falsity value of implications (which  $B$  may contain) relies on the set  $\perp_w$  that generally depends on  $w$ .

However, we can still benefit from the 2nd-order elimination rule provided we introduce a restriction on the formula  $A$  based on the following notions of *amodality*:

Let  $A$  be a formula possibly depending on a 2nd-order variable  $X$ . We say that  $A$  is *semantically amodal* w.r.t. the variable  $X$  when for every valuation  $\rho$ , for every world  $w \in \mathscr{W}$  and for every pair of modal falsity value functions  $F, F' : \mathcal{D}^k \rightarrow \mathfrak{P}(\Pi \times \mathscr{W})$  that coincide in world  $w$  (i.e.  $F(d_1, \dots, d_k)_w = F'(d_1, \dots, d_k)_w$  for all  $d_1, \dots, d_k \in \mathcal{D}$ ) one has

$$\|A[\rho; X \leftarrow F]\|_w = \|A[\rho; X \leftarrow F']\|_w.$$

In other words, the formula  $A$  is amodal w.r.t.  $X$  when its falsity value in world  $w$  only depends on the falsity value interpreting the variable  $X$  in world  $w$  (for all  $w \in \mathscr{W}$ ).

By extension, we say that a formula  $A$  is *semantically amodal* w.r.t. a restricted 2nd-order variable  $\alpha$  when the formula  $A\{\alpha := X\}$  is semantically amodal w.r.t.  $X$ , where  $X$  is a fresh full 2nd-order variable. We easily check that:

**Fact 4 (Restricted 2nd-order elimination)** — *If the formula  $A$  is semantically amodal w.r.t.  $\alpha$ , then the subtyping judgment  $\forall \alpha A \leq A\{\alpha(x_1, \dots, x_k) := B\}$  is valid for every formula  $B$  (without any restriction on  $B$ ).*

*Proof.* We instantiate the variable  $\alpha$  differently depending on the world  $w$ , taking in world  $w \in \mathscr{W}$  the simple falsity value function that coincides with the interpretation of the predicate  $B(x_1, \dots, x_n)$  in world  $w$ .  $\square$

### 6.3. Syntactic amodality

The semantic notion of amodality can be approximated with a syntactic notion of amodality that we inductively define as follows: we say that  $A$  is (syntactically) amodal w.r.t. a variable  $X$  when either:

- $X \notin FV(A)$ ;
- $A$  is  $X(e_1, \dots, e_k)$ ;
- $A$  is  $B \Rightarrow C$ , where  $B$  and  $C$  are amodal w.r.t.  $X$ ;
- $A$  is  $\forall x B$ , where  $B$  is amodal w.r.t.  $X$ ;
- $A$  is  $\forall \alpha B$ , where  $B$  is amodal w.r.t.  $X$ ;
- $A$  is  $\forall Y B$  ( $Y \neq X$ ), where  $B$  is amodal w.r.t.  $Y$ .

(Intuitively, this definition expresses that no atomic subformula of  $A$  of the form  $X(e_1, \dots, e_k)$  occurs below a modal connective.) The syntactic notion of amodality w.r.t. a restricted 2nd-order variable  $\alpha$  is defined similarly.

**Fact 5** — *If  $A$  is syntactically amodal w.r.t.  $X$  (resp.  $\alpha$ ), then  $A$  is semantically amodal w.r.t.  $X$  (resp.  $\alpha$ ).*

We can now complete the proof of the following:

**Proposition 7** — *The typing rules of Fig. 2 are all valid.*

### 6.4. Logical consequences

In the general case, a full 2nd-order universal quantification is a subtype of the corresponding restricted 2nd-order universal quantification:  $\forall X A \leq \forall \alpha A\{X := \alpha\}$ . However, in the case where  $A$  is amodal w.r.t.  $X$ , then both quantifications are equivalent:  $\forall X A \sim \forall \alpha A\{X := \alpha\}$ .

An important consequence is that the standard 2nd-order encodings of connectives and quantifiers ( $\top$ ,  $\perp$ ,  $\wedge$ ,  $\vee$ ,  $\exists$ ) do not depend on whether we use the full or the restricted form of 2nd-order quantification. For instance, we have:

$$\begin{aligned}
A \wedge B &\equiv \forall Z ((A \Rightarrow B \Rightarrow Z) \Rightarrow Z) \\
&\sim \forall \gamma ((A \Rightarrow B \Rightarrow \gamma) \Rightarrow \gamma) \\
\exists v A(v) &\equiv \forall Z (\forall v (A(v) \Rightarrow Z) \Rightarrow Z) \\
&\sim \forall \gamma (\forall v (A(v) \Rightarrow \gamma) \Rightarrow \gamma)
\end{aligned}$$

(where  $v$  is a variable of any kind). And similarly for  $\perp$ ,  $\top$ ,  $A \wedge B$  and  $e = e'$ , without any restrictions on  $A$  and  $B$  (but the freshness condition on the variables  $Z$  and  $\gamma$ ).

**Existential quantification and comprehension** Dually, the (encoded) restricted existential quantification is a subtype of the non-restricted one:  $\exists\alpha A(\alpha) \leq \exists X A(X)$  (in the general case); and in the case where  $A(\alpha)$  is amodal w.r.t.  $\alpha$  this becomes an equivalence:  $\exists\alpha A(\alpha) \sim \exists X A(X)$ .

A counter-intuitive consequence of this fact is that both forms of the comprehension scheme

$$\exists X \forall \vec{y} (X(\vec{y}) \Leftrightarrow B(\vec{y}))$$

and 
$$\exists \alpha \forall \vec{y} (\alpha(\vec{y}) \Leftrightarrow B(\vec{y}))$$

are equivalent w.r.t. subtyping (since the existentially quantified formula is amodal w.r.t. the quantified variable) independently from the shape of  $B$ , and thus are both universally realized by the term  $\lambda z . z \langle \mathbf{I}; \mathbf{I} \rangle$  (using the usual encoding of pairing in  $\lambda$ -calculus and writing  $\mathbf{I} = \lambda x . x$ ).

However, the second (and counter-intuitive) form of the comprehension scheme cannot be eliminated to get a realizer of the formula  $\forall \alpha A \Rightarrow A\{\alpha(\vec{y}) := B(\vec{y})\}$  (which is wrong when  $A$  is not amodal w.r.t.  $\alpha$ ) unless we have a realizer of the following property of extensionnality:

$$\forall \alpha (\forall y (\alpha(\vec{y}) \Leftrightarrow B(\vec{y})) \Rightarrow (A \Leftrightarrow A\{\alpha(\vec{y}) := B(\vec{y})\}))$$

But to realize this property, we precisely need the condition that  $A$  is amodal w.r.t.  $\alpha$ , hence the paradox vanishes.

## 7. Realizability and side effects

### 7.1. Stores as worlds

We consider a set  $\mathcal{L}$  of *locations* and let  $\mathcal{W} = \mathbb{N}^{\mathcal{L}}$ , viewing each world  $w \in \mathcal{W}$  as a store mapping every location  $\ell \in \mathcal{L}$  to a natural number  $w(\ell) \in \mathbb{N}$ . We enrich the set  $\mathcal{K}$  with instructions  $\text{get}_\ell$  and  $\text{set}_\ell$  for all  $\ell \in \mathcal{L}$ , with the evaluation rules

$$\begin{array}{l} \text{get}_\ell \star t \cdot \pi \quad \star w \succ t \star \overline{w(\ell)} \cdot \pi \star w \\ \text{set}_\ell \star \bar{n} \cdot t \cdot \pi \star w \succ t \star \pi \quad \star w\{\ell := n\} \end{array}$$

writing  $\bar{n} = \overline{s^n 0}$  the canonical representation of the numeral  $n \in \mathbb{N}$  such as defined in [8].<sup>4</sup>

For every  $(\ell, n) \in \mathcal{W} \times \mathbb{N}$  we define two binary relations  $R_{\ell=n?}, R_{\ell:=n} \subseteq \mathcal{W} \times \mathcal{W}$  by

$$\begin{array}{l} R_{\ell=n?} = \{(w, w) : w(\ell) = n\} \subseteq \text{id}_{\mathcal{W}} \\ R'_{\ell:=n} = \{(w, (w; \ell \leftarrow n)) : w \in \mathcal{W}\} \end{array}$$

and let  $\mathfrak{R} = \{R_{\ell=n?}; R_{\ell:=n} : (\ell, n) \in \mathcal{L} \times \mathbb{N}\} \subseteq \mathfrak{P}(\mathcal{W}^2)$ .

<sup>4</sup>For an explanation why numbers are not represented using Church encoding in classical realizability, see [4].

## 7.2. The language of formulæ

We now consider the following language of formulæ

$$\begin{array}{l} \text{Formulæ} \quad A, B ::= X(e_1, \dots, e_k) \mid \forall X A \\ \quad \quad \quad \mid A \Rightarrow B \mid \{e\} \Rightarrow B \mid \forall x A \\ \quad \quad \quad \mid \{\ell = e\}A \mid [\ell := e]A \end{array}$$

introducing two modal constructions, namely:

- $\{\ell = e\}A$  (' $A$  holds provided  $\ell$  equals  $e$ '), and
- $[\ell := e]A$  ('after setting  $\ell$  to  $e$ ,  $A$  holds').

Moreover, we introduce the 'type'  $\{e\} \Rightarrow A$  of functions building a proof of  $A$  when applied to the canonical representation of  $e$  (see [8, 4] for more explanations).

We extend the realizability interpretation as follows:

$$\begin{array}{l} \|\{e\} \Rightarrow B\|_w = \{\llbracket e \rrbracket\} \cdot \|B\|_w \\ \|\{\ell = e\}A\|_w = \begin{cases} \|A\|_w & \text{if } w(\ell) = \llbracket e \rrbracket \\ \emptyset & \text{otherwise} \end{cases} \\ \|\llbracket \ell := e \rrbracket A\|_w = \|A\|_{w; \ell \leftarrow \llbracket e \rrbracket} \end{array}$$

In the realizability model, the construction  $\{\ell = e\}A$  (resp.  $[\ell := e]A$ ) is thus interpreted exactly as the necessity connective  $\Box_{R_{\ell=e?}} A$  (resp.  $\Box_{R_{\ell:=e}} A$ ), with this subtlety that the underlying relation now depends on the value of  $e$ —and thus on the interpretation of the free variables of  $e$ . Despite these differences, the property of adequacy (Prop. 1) still holds, and all the inference rules of Fig. 1 are still valid (taking care of restricting 1st-order Barcan subtyping rule to the case where the modality does not depend on  $x$ ).

In this modal language, the type system of Fig. 1 can be enriched with the rules of Fig. 3 that give typings for  $\text{get}_\ell$  and  $\text{set}_\ell$  as well as subtyping and commutation rules for both modalities  $\{\ell = e\}A$  and  $[\ell := e]A$ , since:

**Proposition 8** — *For every  $\mathfrak{R}$ -monotonic saturated set  $\perp\!\!\!\perp$ , the judgments of Fig. 3 are valid.*

## 8 Future work

### 8.1 More on side effects

The language described in Section 7 to typecheck access and assignment in a store is still very rudimentary—and puts severe conditions on the set of possible poles  $\perp\!\!\!\perp$  via the condition of  $\mathfrak{R}$ -monotonicity. The expressivity of this system should be more investigated, as well as alternative presentations.

Also note that the modality  $[\ell := e]A$  is close to the modalities used in Dynamic Logic [5], and further work should be done to clarify the possible connections.

---


$$\begin{aligned} &\vdash \text{get}_\ell : (\{e\} \Rightarrow A) \Rightarrow \{\ell = e\}A \\ &\vdash \text{set}_\ell : \{e\} \Rightarrow [\ell := e](A \Rightarrow A) \\ &A \leq \{\ell = e\}A \\ &\{\ell = e\}(A \Rightarrow B) \sim \{\ell = e\}A \Rightarrow \{\ell = e\}B \\ &\{\ell = e\}\{\ell' = e'\}A \sim \{\ell' = e'\}\{\ell = e\}A \\ &[\ell := e][\ell' := e']A \sim [\ell' := e'][\ell := e]A \quad (\ell \neq \ell') \\ &[\ell := e][\ell := e']A \sim [\ell := e']A \\ &\{\ell = e\}[\ell' := e']A \sim [\ell' = e']\{\ell = e\}A \quad (\ell \neq \ell') \\ &\{\ell = e\}[\ell := e]A \sim \{\ell = e\}A \\ &[\ell := e]\{\ell = e\}A \sim [\ell := e]A \end{aligned}$$


---

**Figure 3. Typing/subtyping rules for access and assignment operations**

## 8.2 A modal decomposition of forcing

D. Scott remarked that P. Cohen’s forcing can be decomposed in terms of Kripke forcing by combining a particular  $A$ -translation with the embedding of LJ into S4:

$$w \Vdash^{\text{Cohen}} F \quad \Leftrightarrow \quad w \Vdash^{\text{Kripke}} (F^{\neg\neg})^\square$$

where  $\square$  is defined from the binary relation  $w \geq w'$  ( $w$  is a weaker condition than  $w'$ ), and where  $F \mapsto F^{\neg\neg}$  is an  $A$ -translation with a suitable formula  $A$ .

In terms of realizability, this seems to indicate that realizers of  $w \Vdash F$  (in Cohen’s sense) could be obtained from proofs of  $F$  (in the sense of Fig. 1) using a CPS-translation (the embedding from LJ to S4 being the identity on proof-terms, as shown in 5.3).

Such a decomposition could provide a better understanding of the connections between classical realizability and forcing, and constitutes a promising tool to analyze the method described in [9] to realize the axiom stating the existence of a selective ultrafilter over  $\mathbb{N}$ .

## References

[1] Steven Awodey, Lars Birkedal, and Dana S. Scott. Local realizability toposes and a modal logic for computability. *Mathematical Structures in Computer Science*, 12(3):319–334, 2002.

[2] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and The Foundations of Mathematics*. North-Holland, 1984.

[3] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[4] M. Guillermo. *Jeux de réalisabilité en arithmétique classique*. PhD thesis, Université Paris 7, 2008.

[5] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[6] G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic*. Routledge, 1968.

[7] J.-L. Krivine. Dependent choice, ‘quote’ and the clock. *Th. Comp. Sc.*, 308:259–276, 2003.

[8] J.-L. Krivine. Realizability in classical logic. Unpublished lecture notes (available on the author’s web page), 2005.

[9] J.-L. Krivine. Structures de réalisabilité, RAM et ultrafiltre sur  $\mathbb{N}$ . Manuscript, available on the author’s web page, 2008.

[10] J. Lipton and M. J. O’Donnell. Some intuitions behind realizability semantics for constructive logic: Tableaux and läuchli countermodels. *Ann. Pure Appl. Logic*, 81(1-3):187–239, 1996.

[11] A. Miquel. Classical program extraction in the calculus of constructions. In J. Duparc and T. A. Henzinger, editors, *CSL*, volume 4646 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 2007.

## A Proofs

*Proof of Lemma 1 p. 4 (Peirce's law).* 1. Assume that  $\pi \in \|A\|_w$  and take an arbitrary element of  $\|A \Rightarrow B\|_w$ , that is: a stack of the form  $u \cdot \pi'$  where  $u \in |A|_w$  and  $\pi' \in \|B\|_w$ . We have  $k_\pi \star u \cdot \pi' \star w \succ u \star \pi \star w \in \perp$  (since  $u \in |A|_w$ ) hence  $k_\pi \Vdash_w A \Rightarrow B$ .

2. Consider an element of  $\|((A \Rightarrow B) \Rightarrow A) \Rightarrow A\|_w$ , that is: a stack  $u \cdot \pi$ , where  $u \in \|(A \Rightarrow B) \Rightarrow A\|_w$  and  $\pi \in \|A\|_w$ . We have  $\mathfrak{c} \star u \cdot \pi \star w \succ u \star k_\pi \cdot \pi \star w \in \perp$  (since  $k_\pi \cdot \pi \in \|(A \Rightarrow B) \Rightarrow A\|_w$  by item 1) therefore  $\mathfrak{c} \Vdash_w ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$ .  $\square$

*Proof of Prop. 1 p. 5 (Adequacy).* We first consider the validity of the typing rules of Fig. 1.

- Axiom and weakening. Both cases are obvious.
- Introduction of implication. Assume that the judgment  $\Gamma, x : A \vdash t : B$  is valid, and consider a valuation  $\rho$ , a world  $w \in \mathscr{W}$  and a substitution  $\sigma \Vdash_w \Gamma[\rho]$ . To show that  $(\lambda x . t)[\sigma] \Vdash_w (A \Rightarrow B)[\rho]$ , consider an arbitrary element of  $\|(A \Rightarrow B)[\rho]\|_w$ , that is, a stack of the form  $u \cdot \pi$  where  $u \in |A[\rho]|_w$  and  $\pi \in \|B[\rho]\|_w$ . We have

$$(\lambda x . t)[\sigma] \star u \cdot \pi \star w \succ t[\sigma; x := u] \star \pi \star w \in \perp$$

(by hypothesis), hence  $\Gamma \vdash \lambda x . t : A \Rightarrow B$  is valid.

- Elimination of implication. Assume that both judgments  $\Gamma \vdash t : A \Rightarrow B$  and  $\Gamma' \vdash u : A$  are valid, and consider a valuation  $\rho$ , a world  $w \in \mathscr{W}$  and a substitution  $\sigma \Vdash_w (\Gamma, \Gamma')[\rho]$ . To show that  $(tu)[\sigma] \Vdash_w B[\rho]$ , consider an arbitrary stack  $\pi \in \|B[\rho]\|_w$ . We have

$$(tu)[\sigma] \star \pi \star w \succ t[\sigma] \star u[\sigma] \cdot \pi \in \perp$$

since  $t[\sigma] \Vdash_w A \Rightarrow B$  and  $u[\sigma] \cdot \pi \in \|A \Rightarrow B\|_w$  (from our initial assumptions), hence  $\Gamma \vdash tu : B$  is valid.

- Introduction of first-order universal quantification. Assume that the judgment  $\Gamma \vdash t : A$  is valid (with  $x \notin FV(\Gamma)$ ) and consider a valuation  $\rho$ , a world  $w$ , and a substitution  $\sigma \Vdash_w \Gamma[\rho]$ . To show that  $t[\sigma] \Vdash_w (\forall x A)[\rho]$ , let us consider an arbitrary element of  $\|(\forall x A)[\rho]\|_w$ , that is, an arbitrary stack  $\pi \in \|A[\rho; x \leftarrow d]\|_w$  for some  $d \in \mathcal{D}$ . Since  $x \notin FV(\Gamma)$ , we have  $\Gamma[\rho] = \Gamma[\rho; x \leftarrow d]$  and thus  $\sigma \Vdash_w \Gamma[\rho; x \leftarrow d]$ . From our initial assumption we get  $t[\sigma] \Vdash_w A[\rho; x \leftarrow d]$ , hence  $t[\sigma] \star \pi \in \perp$ . Therefore the judgment  $\Gamma \vdash t : \forall x A$  is valid.
- Introduction of second-order universal quantification. Assume that the judgment  $\Gamma \vdash t : A$  is valid (with  $X \notin FV(\Gamma)$ ) and consider a valuation  $\rho$ , a world  $w$ , and a substitution  $\sigma \Vdash_w \Gamma[\rho]$ . To show that  $t[\sigma] \Vdash_w$

$(\forall X A)[\rho]$ , let us consider an arbitrary element of  $\|(\forall X A)[\rho]\|_w$ , that is, any stack  $\pi \in \|A[\rho; X \leftarrow F]\|_w$  for some  $F : \mathcal{D}^k \rightarrow \mathfrak{P}(\Pi \times \mathscr{W})$ . Since  $X \notin FV(\Gamma)$ , we have  $\Gamma[\rho] = \Gamma[\rho; X \leftarrow F]$  and thus  $\sigma \Vdash_w \Gamma[\rho; X \leftarrow F]$ . From our initial assumption we get  $t[\sigma] \Vdash_w A[\rho; X \leftarrow F]$ , hence  $t[\sigma] \star \pi \in \perp$ . Therefore the judgment  $\Gamma \vdash t : \forall X A$  is valid.

- Peirce's law. This case is given by lemma 1.
- Necessitation rule. Assume that the judgment  $\Gamma \vdash t : A$  is valid, and consider a valuation  $\rho$ , a world  $w \in \mathscr{W}$  and a substitution  $\sigma \Vdash_w \square_R \Gamma[\rho]$ . To show that  $t[\sigma] \Vdash_w \square_{RA} A[\rho]$ , take an arbitrary element of  $\|\square_{RA} A[\rho]\|_w$ , that is, a stack  $\pi \in \|A[\rho]\|_{w'}$  for some world  $w' \in \mathscr{W}$  such that  $R(w, w')$ . From the assumption  $t[\sigma] \Vdash_w \square_{RA} A[\rho]$  we get  $t[\sigma] \Vdash_{w'} A[\rho]$ , hence  $t[\sigma] \star \pi \in \perp$ . Therefore the judgment  $\square_R \Gamma \vdash t : \square_{RA} A$  is valid.
- Subsumption rule. Let us assume that both judgments  $\Gamma \vdash t : A$  and  $A \leq A'$  are valid, and consider a valuation  $\rho$ , a world  $w \in \mathscr{W}$  and a substitution  $\sigma \Vdash_w \Gamma[\rho]$ . To show that  $t[\sigma] \Vdash_w A'[\rho]$ , consider an arbitrary stack  $\pi \in \|A'[\rho]\|_w$ . From the validity of the judgment  $A \leq A'$  we get  $\pi \in \|A[\rho]\|_w$  hence  $t[\sigma] \star \pi \in \perp$  (from the validity of the judgment  $\Gamma \vdash t : A$ ). Therefore the judgment  $\Gamma \vdash t : A'$  is valid.

We then consider the validity of the subtyping rules of Fig. 1. Most cases are obvious, and we only deal with the cases concerning modalities:

- Covariance of necessity. Assume that  $A \leq A'$  is valid, and consider a valuation  $\rho$  and a world  $w \in \mathscr{W}$ . We have:

$$\begin{aligned} \|\square_{RA} A[\rho]\|_w &= \bigcup_{w' \in R(w)} \|A[\rho]\|_{w'} \\ &\supseteq \bigcup_{w' \in R(w)} \|A'[\rho]\|_{w'} = \|\square_{RA'} A'[\rho]\|_w \end{aligned}$$

- Distributivity. For all valuations  $\rho$  and for all worlds  $w \in \mathscr{W}$  we have:

$$\begin{aligned} \|\square_R(A \Rightarrow B)[\rho]\|_w &= \bigcup_{w' \in R(w)} (|A[\rho]\|_{w'} \cdot \|B[\rho]\|_{w'}) \\ &\supseteq \bigcap_{w'' \in R(w)} |A[\rho]\|_{w''} \cdot \bigcup_{w' \in R(w)} \|B[\rho]\|_{w'} \\ &\supseteq |\square_{RA} A[\rho]\|_w \cdot \bigcup_{w' \in R(w)} \|B[\rho]\|_{w'} \\ &\supseteq \|(\square_{RA} A \Rightarrow \square_{RB} B)[\rho]\|_w \end{aligned}$$

- Relation inclusion. Assume that  $S \subseteq R \subseteq \mathscr{W} \times \mathscr{W}$ .

For all valuations  $\rho$  and for all worlds  $w \in \mathcal{W}$  we have:

$$\begin{aligned} \|\Box_S A\|_w &= \bigcup_{w' \in S\langle w \rangle} \|A\|_{w'} \\ &\subseteq \bigcup_{w' \in R\langle w \rangle} \|A\|_{w'} = \|\Box_R A\|_w. \end{aligned}$$

- Identity modality. For all valuations  $\rho$  and for all worlds  $w \in \mathcal{W}$  we have:

$$\|\Box_{\text{id}} A\|_w = \bigcup_{w' \in \text{id}\langle w \rangle} \|A\|_{w'} = \|A\|_w.$$

- Sequence modality. For all valuations  $\rho$  and for all worlds  $w \in \mathcal{W}$  we have:

$$\begin{aligned} \|\Box_{R;S} A\|_w &= \bigcup_{w'' \in (R;S)\langle w \rangle} \|A\|_{w''} \\ &= \bigcup_{w' \in R\langle w \rangle} \bigcup_{w'' \in S\langle w' \rangle} \|A\|_{w''} \\ &= \|\Box_R \Box_S A\|_w. \end{aligned}$$

- First- and second-order Barcan laws. For all valuations  $\rho$  and for all worlds  $w \in \mathcal{W}$  the equality  $\|\forall x \Box_R A[\rho]\|_w = \|\Box_R \forall x A[\rho]\|_w$  amounts to a straightforward commutation of unions. The same holds for second-order Barcan law.  $\square$